Implementation of an ASIP based SDR platform for MIMO OFDM transceivers

T. Kempf*, D. Guenther*, U. Deidersen*, A. Yarmuch*, G. Ascheid*, M. Adrat† and M. Antweiler†
*Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, Germany
†Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE, Wachtberg, Germany
kempf@ice.rwth-aachen.de

Supporting several wireless communication standards with a single hardware device demands for a flexible hardware platform that allows the implementation of these standards in software while still meeting the tight real-time requirements. Furthermore, these Software Defined Radio (SDR) platforms have to achieve a viable trade-off between the contradicting requirements of flexibility on one side and area, power as well as energy efficiency on the other side.

One promising approach is to utilize programmable Application Specific Instruction Set Processors (ASIPs) for the computationally intensive functions of the baseband transceiver. Thanks to their programmability they provide sufficient flexibility while being optimized for the targeted application.

In the course of this paper we highlight an ASIP design based on the LISA 2.0 language and Synopsys' Processor Designer. The primary target of the ASIP is the computational complex MIMO OFDM preprocessing and equalization, e.g. found in the IEEE 802.11n standard. Furthermore, a Virtual Platform prototype of a Software Defined Radio (SDR) platform has been designed to investigate the real-time performance of the complete hardware platform. This Virtual Platform allows for detailed system-level performance analysis and area estimation at an early design stage prior to finalizing the time-intensive RTL design.

I. Introduction

SDR platforms provide a significant benefit for the development of wireless communication devices ranging from a single handset to infrastructure systems, such as femtocells or complete base-stations. For example, low volume markets like satellite or military communication systems can benefit from reduced hardware development cost due to a common hardware platform. Other markets like base-stations and mobile handsets can profit by multi-standard support on a single hardware platform as well as reduced maintenance costs due to remote software updates.

In general, these SDR platforms are based on a heterogeneous Multi-Processor System-on-Chip (MPSoC) that has the potential to provide a viable trade-off between flexibility and efficiency. Based on their definition, heterogeneous MPSoC platforms consist of multiple processor cores of different types. These processor cores range from simple RISC-based processors over Digital Signal Processors (DSPs) to highly specialized Application Specific Instruction set Processors (ASIPs). Apart from these processor cores, a wide variety of different hardware components needs to be integrated. Examples are memories, interrupt controller, DMAs and HW accelerators which implement dedicated functions due to superior performance and/or better energy efficiency. As a result, development of a complete SDR platform is one of the most complex assignments today. This includes the design of the individual HW IP components, the integration of the hardware components on the system level as well as the development of the distributed software that is executed on the various processing elements of the platform.

This work has been supported by the UMIC Research Centre, RWTH Aachen University.

In this work we present a platform design of an SDR for baseband operations of MIMO OFDM transceivers that includes the development of individual IP components and a Virtual Platform for system-level design. In the computational intensive data plane processing, developed IP components are optimized for the target functions of the application. Hence, in traditional designs, these have been developed as hardwired architectures with only limited configuration support. For functions that do not require more flexibility, like OFDM demodulation (basically FFT/iFFT processing) this principle is kept for the sake of improved implementation efficiency. Other parts of the physical layer processing like channel estimation, MIMO preprocessing and spatial equalizing can benefit from software programmable architectures to support a large range of different wireless communication standards such as WLAN, WIMAX and LTE. However, for acceptance of system designers a performance comparable to hardwired solutions is mandatory.

In order to implement the control plane processing, that occurs in modern wireless communication standards we incorporated our in-house developed RISC processor core called IRISC that comes with an optimized C-compiler based on ACE's CoSy compiler design environment [1]. Both processor cores (ASIP for MIMO processing and IRISC) have been developed with the LISA 2.0 language and Synopsys' Processor Designer [2]. For efficient investigation of the system-level performance and the development of the firmware software, we developed a Virtual Platform (VP) of the complete heterogeneous MPSoC platform in SystemC using of Synopsys Platform Architect [2]. This VP contains cycle accurate models of all processor cores based on the Transaction Level Modeling standard 2.0 (TLM2) [3] that allows efficient software development and debugging on the system-level. Furthermore, additional SystemC components for all required hardware IP blocks have been designed for platform integration and timing measurements. Based on the quick modeling and simulation capabilities this Virtual Platform is available prior to the availability of the RTL implementation and hence fastens the overall design process.

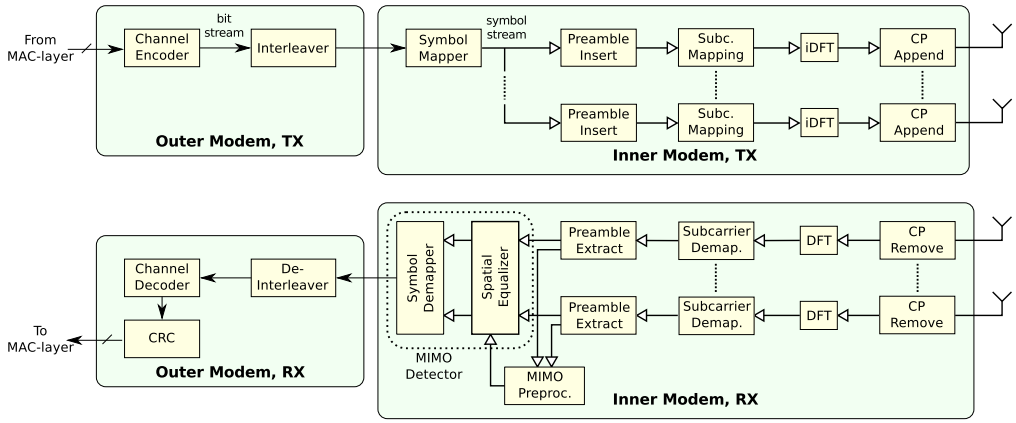


Fig. 1. MIMO OFDM Transceiver Overview

II. Related Work

SDR platforms provide a significant benefit for the development of wireless communication devices. Already today the market of base-stations and femtocells is dominated by SDR platforms e.g. from Texas Instruments (TMS320TCI6618 [4]) or Freescale (MSC8156 DSP [5]) that support a wide variety of standards such as WCDMA/HSPA/HSPA+, TD-SCDMA, GSM, LTE and WiMAX. In addition, latest platforms for handsets like the Nvidia Icera [6] platform and Qualcomm Gobi [7] platform adopt the concept of SDR.

Common feature of all these platforms is that they are build on the concept of a heterogeneous MPSoC. While the control-plane processing is typically executed on a RISC based architecture e.g. ARM9 or ARM11 [8], the processor cores of the data plane processing are equipped with SIMD (single instruction multiple data) instructions. Examples for these data plane processors are Tensilica ConnX [9], CEVA-X/XC [10] and EVP [11] architectures. Other processor cores can be extended with additional vector units like the STxP70 [12] and hence can be used for such operations. In addition, these platforms incorporate dedicated hardware circuits for improved execution of dedicated transceiver functions. For example, most platforms have a hardware accelerator for fast fourier transformation that is an efficient way to implement orthogonal frequency-division multiplexing (OFDM).

One common drawback of these heterogeneous SDR platforms is that software development gets extremely challenging compared to single core development or on a homogeneous architecture. For example, porting an application from one processor to the other requires in a homogeneous environment basically no effort, while in a heterogeneous architecture the software needs to be ported to the target processor. In addition, the DSPs and ASIPs in the data plane processing have to be programmed typically in Assembly or a low-level C with the use of intrinsic and compiler known functions (CKFs). Clearly this requires a significant effort and it is of practical interest to limit or to avoid such time intensive software development.

Our previously introduced Nucleus methodology [13] and tooling [14] target this issue. They guarantee an effective design approach that allows simple porting of a transceiver implementation to another platform while maintaining high efficiency [15]. The key idea is to first analyze the transceiver to be implemented based on it's computationally intensive algorithmic kernels. Since these Nuclei account for most of the computational effort, efficient implementations - Flavors - are implemented for each specific hardware architecture. Thanks to the effective tooling and mapping process porting of a given transceiver from one to the other platform can be carried out quickly and with limited design effort.

III. MIMO OFDM Transceiver Application

Before introducing the details of our SDR platform, a common transceiver structure (Fig. 1) is highlighted, whereas a more extensive overview can be found in [16]. Within this work we will concentrate on the WLAN 802.11n communication standard as an example for a MIMO OFDM transceiver. The related physical layer application is depicted in Figure 1 that shows a single MIMO OFDM communication link and an open-loop receiver architecture with configurable transmit (N_t) and receive (N_r) antennas.

The incoming bits from the Medium Access Control (MAC) layer are first encoded and interleaved to cope with channel impairments and to protect them against burst errors. Next, the coded bits are mapped to complex symbols and distributed among the multiple data streams, which are sent via the corresponding transmit antennas. Each transmit path contains a complete OFDM transmitter that first maps the symbols to the related data subcarriers and pilots are added to their subcarriers. The time signal is obtained by applying an inverse Discrete Fourier Transform (iDFT/iFFT) and a cyclic prefix is inserted to guard the OFDM symbol from inter-symbol and inter-carrier interference. Finally, the signal is windowed and passed to the analog domain by D/A conversion.

The lower part of Figure 1 illustrates the receiver architecture that performs the inverse operations of the transmitter and attempts to eliminate the impact of the wireless channel. After A/D conversion the received signal is represented in digital state and the cycle prefix is removed. With the help of a DFT(FFT) the signal is transformed back to frequency domain and the guard carriers are removed from the OFDM symbols. Before reconstructing the actual data payload, MIMO

preprocessing is carried out. This comprises first the estimation of the channel matrices for each subcarrier. Second an equalizer matrix needs to be obtained[1], which requires a matrix inversion. In general, matrix decompositions methods are preferable to direct inversion techniques thanks to their numerical stability, hence most implementations use QR, Cholesky, or Divide-and-Conquer decomposition schemes. By applying spatial equalizing the channel impact is mitigated on the transmitted baseband data. The corrected symbols are mapped from their complex baseband representation back to a bitwise representation by means of soft symbol demapping. The received soft bits are de-interleaved and decoded by exploiting the redundancy added during channel encoding. Finally data is passed to the CRC check and back to the MAC-layer.

A. Application Analysis

For data transmission over the wireless communication channel, we use a common system model where the symbol stream is represented by a vector $\mathbf{x}$ of dimension $N_t \times 1$ whereas each symbol is selected from a given signal constellation with zero mean and transmit power of E_s per antenna. The transmission coefficients between each antenna pair are modeled to be normalized independent and identically distributed (i.i.d.) complex circular Gaussian distributions with zero-mean, and remain invariant for all time slots within a frame, but may change to another independent state in the next frame. The wireless channel can be modelled as a matrix $\mathbf{H}$ of dimension $N_r \times N_t$ with the transmission coefficients as its elements. Assuming perfect synchronization during reception of the signals, the system can be expressed for a quasi-static Rayleigh flat-fading MIMO channel as

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

with $\mathbf{n}$ being an additive space-time noise vector that is assumed to be temporally and spatially white and is of complex circular Gaussian distribution with zero mean and variance σ_n^2.

Following our previous work based on the Nucleus methodology [17] a detailed analysis of state-of-the-art receiver algorithms has been carried out in the past [15]. Based on this analysis, we selected transceiver algorithms that provide a suitable trade-off between computational complexity and algorithmic performance. Please note that a vast variety of algorithms exist, hence only a limited set has been investigated here that will be briefly introduced next.

a) OFDM Modulation and Demodulation: OFDM (de-)modulation can be efficiently realized by using the iFFT and FFT algorithm [18] if the number of subcarriers is a power of two. Both iFFT and FFT implementations differ only in the utilized twiddle factors, hence can be efficiently implemented by the same hardware architecture. Thanks to the well known and frequently utilized FFT algorithm a vast variety of implementation exists, e.g. radix-2 or mixed-radix implementations.

[1]Other approaches exist like sphere decoding, that do not require an equalizer matrix but are not investigate in the course of this work.

Within the targeted MIMO OFDM transceiver application for each OFDM symbol up to four FFTs/iFFTs have to be carried out. Accordingly, an efficient hardware support for this part of the receiver is requested, that will be active always while receiving or transmitting data. Hence, we implemented this function on a dedicated hardware accelerator that will be highlighted in Section IV-B.

b) Channel Estimation: To mitigate the channel impact on the received symbol vector $\mathbf{y}$, the channel matrix $\mathbf{H}$ has to be estimated. In conformance with the IEEE 802.11n standard, a block type channel estimation over training symbols inserted at the beginning of a frame has been applied [19]. With such pilot structure the channel condition is estimated once at the start of a frame and is used for all data symbols in the frame.

Block type channel estimation requires a minimum of N_t preamble symbol vectors $\mathbf{p}_i$ of dimension $N_t \times 1$ to estimate the channel condition. Accordingly, the entire preamble can be expressed as a matrix $\mathbf{P} = [\ \mathbf{p}_1\ ...\ \mathbf{p}_{N_t}\ ]$. After transmission over the wireless channel, the receiver obtains the signal

$$\mathbf{S} = \mathbf{H}\mathbf{P} + \mathbf{N} \tag{2}$$

with the received symbol vectors $\mathbf{s}_i$ of dimension $N_r \times 1$ that are combined into a matrix $\mathbf{S} = [\ \mathbf{s}_1\ ...\ \mathbf{s}_{N_t}\ ]$ and $\mathbf{N} = [\mathbf{n}_1\ ..\ \mathbf{n}_{N_t}]$ is the additive noise matrix.

A simple, low complexity solution is the least squares method which can be formulated as follows if matrix $\mathbf{P}$ is selected as an orthogonal matrix (common for most communication standards including IEEE 802.11n)

$$\hat{\mathbf{H}}_{\mathbf{LS}} = \mathbf{S}\mathbf{P}^{\mathbf{H}}\left(\mathbf{P}\mathbf{P}^{\mathbf{H}}\right)^{-1} = \frac{1}{N_t}\mathbf{S}\mathbf{P}^{\mathbf{H}} \text{ with } \mathbf{P}\mathbf{P}^{\mathbf{H}} = N_t\mathbf{I}_{\mathbf{N_t}}. \tag{3}$$

Here $(.)^{\mathbf{H}}$ stands for the Hermitian transpose and matrix $\mathbf{I}_{\mathbf{N_t}}$ denotes an identity matrix of dimension N_t.

Channel estimation requires complex valued matrix operations for the selected least square (eq. 2) and minimum mean square error ([15]) methods. In the likely case of a BPSK modulated preamble this can be further reduced to additions and subtractions of complex values since all entries of matrix $\mathbf{P}$ are either +1 or -1. Using an MMSE based approach computation of a reciprocal is required to obtain channel estimates. One efficient approach for computation is the Newton-Raphson method [20] that allows sufficiently accurate approximation of the reciprocal with a few iterations (typically 2-3 iterations should be sufficient for limited fixed point precision).

c) Spatial Equalizing: Having obtained the channel estimate $\hat{\mathbf{H}}$, the impact of the MIMO channel on the transmitted data can be mitigated by spatial equalizing. Apart from a large variety of algorithmic options, so-called linear equalizers are often employed as suboptimal but computationally attractive solutions. The output signal of a linear equalizer can be expressed as:

$$\hat{\mathbf{x}} = \mathbf{G}\mathbf{y} \tag{4}$$

where the design of the $N_t \times N_r$ filter matrix $\mathbf{G}$ depends on the equalizing criterion. Well known criteria are zero forcing and minimum mean square error (MMSE) approaches,

whereas the latter one achieves superior performance with minor complexity increase. Accordingly, we apply an MMSE based equalization scheme that computes the equalizer matrix $\mathbf{G}$ by minimizing the expected mean square error between $\hat{\mathbf{x}}$ and the original payload data $\mathbf{x}$.

$$\mathbf{G} = \underset{\mathbf{G}}{\operatorname{argmin}} \, \mathrm{E}\left\{\|\mathbf{x} - \hat{\mathbf{x}}\|^2\right\} = \left(\hat{\mathbf{H}}^{\mathbf{H}}\hat{\mathbf{H}} + \frac{\sigma_n^2}{E_s}\mathbf{I}_{\mathbf{N_t}}\right)^{-1}\hat{\mathbf{H}}^{\mathbf{H}}. \quad (5)$$

As a side note, the MMSE approach offers future extension to iterative receiver architectures based on the MMSE-PIC [21] and other iterative LMMSE [22] algorithms that achieve similar algorithmic performance but yield far less computational complexity compared to algorithms like sphere decoding [23].

Implementing the above discussed spatial equalization requires the inversion of a complex valued matrix of dimensions depending on the number of transmit and receive antennas. For high data-rate modes of WLAN and LTE antenna constellations up to 4x4 are supported. Accordingly inversion of a 4x4 matrix is requested. Direct matrix inversions with limited fixed point precision suffers from numerical stability, accordingly more robust matrix decomposition schemes e.g. QR-, LDL^T- and LU-decomposition [24] are typically used. Following a flexible SDR approach we do not limit our implementation to one or the other decomposition algorithm. For example, iterative receiver architectures based on soft-in-soft-out sphere decoding require a QR-decomposition while the MMSE-PIC [21] can benefit from a LU-decomposition. Therefore, when targeting a flexible SDR implementation common operations for matrix-multiplications, matrix-additions and matrix-decompositions should be supported.

d) Soft-Symbol Demapping: Applying soft-symbol demapping a soft bit representation is obtained from the received complex symbols. This computes a log-likelihood ratio (LLR) for each received bit. With application of the max-log approximation and assuming that $p(\mathbf{y}|\mathbf{H}, \mathbf{x})$ is Gaussian, the noise is white Gaussian, the LLR for stream k with the equalized symbol vector $\mathbf{z}$ is computed as:

$$L(b_{k,i}) \approx \rho_k \left(\min_{s \in A_i^0} |z_k - s|^2 - \min_{s \in A_i^1} |z_k - s|^2 \right) \quad (6)$$

$$\rho_k = \frac{1}{\sigma_n^2 \left[\left(\mathbf{H}^{\mathbf{H}}\mathbf{H} + \frac{\sigma_n^2}{E_s}\mathbf{I}_{\mathbf{N_t}} \right)^{-1} \right]_{k,k}} - 1 \quad (7)$$

The scalar ρ_k denotes the signal to interference and noise ratio (SINR) for stream k [25]. Whereas, A_i^1 represents a subset of all constellation symbols that contain a one at position i of their bitwise representation and A_i^0 denote the ones with a zero at position i. An efficient implementation of such LLR computation can be found in [26] when a Gray-coded constellation is utilized (typically done).

Such an implementation of a soft-symbol demapping requires computation of a piecewise linear function. Due to the large amount of required soft-symbol demappings, special instructions should be added to the ASIP to compute them efficiently.

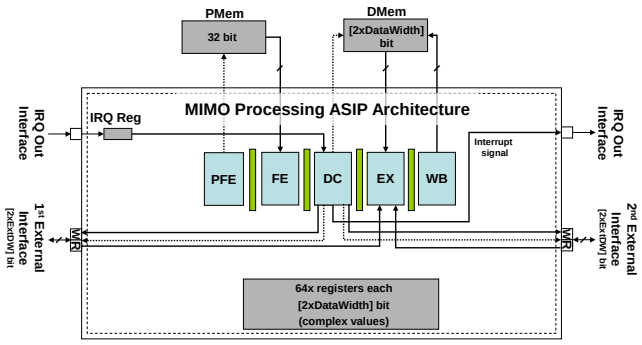


Fig. 2. ASIP for MIMO processing architecture overview

IV. Flexible HW components

In this section, we present the three main HW components of the envisioned SDR platform. First, the ASIP for MIMO processing is highlighted, followed by a discussion of the FFT HW accelerator and the IRISC control processor. For all components synthesizable RTL implementations were developed as well as a SystemC based simulation model for Virtual Platform development.

A. ASIP for MIMO Processing

After the analysis of the application and the identification of the computational kernels, we briefly summarize the selected hardware components for the different transceiver functions. As mentioned above, the FFT/iFFT components are active for each received data symbol. Therefore, it is obvious that an efficient hardware with limited flexiblility is an appropriate implementation choice, which is introduced in Section IV-B.

The remaining transceiver functions require mainly complex valued matrix/vector operations. Accordingly, we have designed a more flexible ASIP that can efficiently compute these functions, but still remains a programmble architecture.

An overview of the ASIP HW architecture is given in Figure 2. As previously analyzed, key operations that have to be carried out are vector- and matrix- computations with complex values. Since the dimension of vectors and matrices depends on the number of receive and transmit antennas (typically up to 4x4) the ASIP HW has been designed with the following features:

- Relatively large register file (64 registers) to minimize memory accesses during computations. The area of register file requires 88kGate that is approx. 43% of the total area of the ASIP.
- Short 5-stage pipeline to avoid long stall cycles or nop operations.
- Native support for complex data-types and all common operations, such as addition/subtraction, (conjugate) multiplication, (conjugate) multiply accumulate/subtract.
- 2-way SIMD for all instructions operating on complex values including random register access.
- Special instructions for computation of reciprocal and inverse square root are available. Computation is based

	Area (kgate)	Prog. Mem. (words)	Data Mem. (words)	Freq. (MHz)
ASIP (24/24bit)	205	1024 (32bit)	512 (48bit)	400
FFT HW Accel.	46.18	-	included	335
IRISC	22.2	2048 (32bit)	2048 (32bit)	435

TABLE I. Synthesis Results of ASIP for MIMO Processing, FFT HW Accelerator, IRISC (Synopsys Design Compiler for the Faraday 90 nm standard cell library under typical conditions (supply voltage 1.0 V, temperature 25 C))

on the Newton-Raphson approximation method. This approximation is carried out by first finding an appropriate start value and then performing several iterations. These two stages are reflected in the two special instuctions *initialize approximation* and *perform iteration*.

- Special instrcutions are available for computing the log-likelihood-ratio (LLR) computation of Gray-coded modulations by piecwise linear functions [26].
- Design time flexible data path, e.g. high precision fix point format Q8.16 bit (=24 bit, one sign bit, seven integer bits and sixteen fractional bits) for real/imaginary data.
- Two external simple handshake protocol interfaces with separate read and write ports.
- Small data memory (1k, words with same data width as data path) to store look-up tables and intermediate results.
- Small program memory (2k, words each 32 bit) due to limited number of computational kernels and program size
- Instructions for simple control mechanisms, e.g. loops and if-else statements.
- Efficient IRQ In/Out interfaces that can be accessed via special instructions.

Thanks to this rather simple ASIP architecture, implementation of the most common matrix-/vector-operations can be efficiently programmed by a few lines of assembly. The same applies to all common decomposition algorithms such as QR, LDL^T, LU and Strassen algorithm.

A.1 Implementation Results

In this subsection the implementation results of our ASIP for MIMO processing are presented. First, the synthesis results are highlighted, followed by a detailed analysis of the execution characteristic of individual functions.

a) Synthesis Results: The ASIP was developed using the Language for Instruction Set Architectures (LISA) and Processor Designer by Synopsys [2]. In order to build an efficient processor core, the design was started from scratch with the required instructions that were analyzed in Section III-A. From the LISA description of the ASIP, the Register Transfer Level (RTL) description was automatically generated as well as the simulator, assembler and linker.

The ASIP with a data format of Q8.16 bit was synthesized with Synopsys Design Compiler [2] for the Faraday 90 nm standard cell library under typical conditions (supply voltage 1.0 V, temperature 25 C). Table I shows the synthesis results in terms of area and frequency.

b) Benchmark of MIMO Processing Functions: In order to evaluate the performance of the developed ASIP, the individual functions of channel estimation, MIMO preprocessing, spatial equalizing and soft symbol dempping (LLR computation) have been implemented and benchmarked. Please note that not only one golden implementation of each function exists, e.g. spatial equalizing (eq. 5) requires the inversion of a matrix that can be implemented in several ways. Accordingly, multiple implementation results are given in Table II.

For comparison of the developed ASIP, we consider hardware architectures found in literature. Since most of the traditional implementations make use of hardwired accelerators for computing the equalizer matrix we compare the measured performance with dedicated hardwired solutions which provide some configuration support, e.g. number of transmit and receive antennas, but *cannot* be programmed like the developed ASIP.

Table III depicts the comparison results. Obviously the first three HW accelerators achieve a higher throughput of a factor of approx. 2 up to 10. In general, this trade-off of up to one order of magnitude matches well with other ASIP results when compared against a specialised non programmable architecture. Compared to other weakly programmable architectures like [27] the developed ASIP achieves an approximately 2 times higher throughput. Please note that our ASIP can also efficiently support spatial equalization and LLR computation (see. Table II). Hence, compared to all other architectures these transceiver functions do not require additional hardware.

B. FFT HW accelerator

Besides the previously discussed MIMO processing, OFDM modulation is another key feature of modern communication standards. Thanks to its structure, OFDM modulation can be efficiently implemented via a Fast Fourier Transformation [31]. FFT is a well known algorithm and many implementation variants exist. We selected a radix-2 decimation-in-frequency memory-based FFT implementation with conflict free memory addressing (see Fig. 3). All supported FFT sizes can be selected at run-time by setting the appropriate configuration of the HW accelerator.

Since the primary goal of this work is an implementation of a flexible MIMO OFDM transceiver that can execute the 802.11n WLAN standard, the number of subcarrier, respectively of the FFT size, is limited to 128. Accordingly, the maximum FFT size has been set to 128 at design-time. During

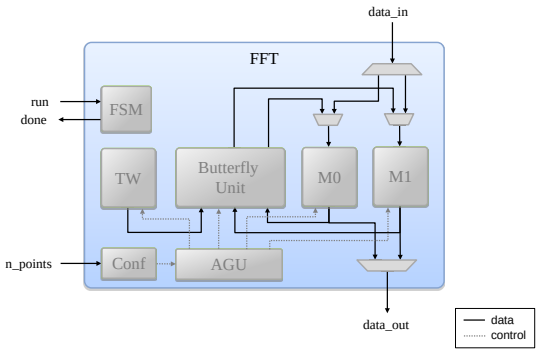


Fig. 3. Radix-2 Decimation-In-Frequency memory-based FFT HW accelerator

Function	Subfunction executed per subcarrier	Cycles	Exec. Time @ 400MHz
Channel Estimation	Least squares method	46	115 ns
Spatial Equalizing using QR decomposition	Computation of Equalizer Matrix G	169	422.5 ns
	SINR computation ρ_k	38	95.0 ns
	Spatial equalization	28	70.0 ns
Spatial Equalizing using LDL^T decomposition	LDL^T decomposition	108	270.0 ns
	SINR computation ρ_k	64	160.0 ns
	Spatial equalization via backsubstitution	40	100.0 ns
Spatial Equalizing using Strassen algorithm	Computation of Equalizer Matrix G	145	362.5 ns
	SINR computation ρ_k	30	75.0 ns
	Spatial equalization	28	70.0 ns
LLR computation per symbol vector	4QAM	13	32.5 ns
	16QAM	19	47.5 ns
	64QAM	24	60.0 ns

TABLE II. Benchmark of transceiver functions executed on ASIP for 4x4 antenna case

Core	Luethi [28]	Patel [29]	Senning [30]	Wu [27]	ASIP	ASIP
Algorithm	QRD/SQRD GR	QRD GR	QRD MGS	Strassen Alg.	Reg. QRD SW	Strassen SW
Technology	180 nm	130 nm	130 nm	65 nm	90 nm	90nm
Area [kGate]	48.7	36.0	182	120	205	205
Frequency [MHz]	167	270	320	400	400	400
Processing Time (scaled)	201 ns^a	103 ns^a	657 ns^{ab}	706 ns^c	375 ns	363 ns^c
Throughput (scaled)	4.98 M QRD/s^a	9.75 M QRD/s^a	28.89 M QRD/s^{ab}	1.42 M EqMat/s^c	2.67 M QRD/s	2.67 M EqMat/s^{cd}

TABLE III. Comparison of ASIP for MIMO Processing
(^{a} Simplified technology scaling applied. ^{b} includes SINR calculation. ^{c} includes computation of equalizer matrix (eq. 5) ^{d} includes external memory accesses)

run-time, the HW accelerator can be configured to FFT sizes of 2^N with $N = \{1, 2, ..., 7\}$. The HW architecture was implemented in VHDL at RTL. Table I shows the synthesis results in terms of area and frequency.

Execution time of each possible FFT computation is deterministic, since one radix-2 operation is computed per cycle. Hence, the resulting computation time is determined by

$$\text{cycles} = log_2(n) * \frac{n}{2} * \frac{1}{f_{max}}, \text{ with } n = \text{FFT size}.$$

Please note that this denotes only the computation time and does not contain reading and writing of input and output values.

When synthesized for a Faraday 90 nm standard cell library under typical conditions (supply voltage 1.0 V, temperature 25 C) our architecture executes a 64 point FFT in 567 ns running at a frequency of 335 MHz. Compared to other architectures in literature, similar performance is achieved. For example, the architecture described in [32] executes a 64 point FFT in 640 ns when running on it's maximum frequency of 100MHz. Considering a simplified technology scaling (130nm to 90nm) this compares to 443 ns for our technology, while our architecture should have a smaller gate count[2].

C. RISC processor for control path operations

For building a complete SDR platform control-path processing is of vital importance. In contrast to the processing characteristics of the data plane, the control plane requires more flexible hardware solutions and demands software programmability on a high-level language. Accordingly, other solutions [33], [34] incorporate ARM or other RISC like architectures with rich C-compiler support.

Thus, we make use of our internal IRISC processor core. The IRISC comes with a rich instruction set and a compiler designed with ACE CoSy compiler designer. Key features of the IRISC are:

- Harvard architecture
- RISC instruction set with conditional execution and multiplier support
- 5 pipeline stages
- 32 bit data width
- 16 general purpose register
- Fully bypassed and interlocked architecture

An RTL description has been generated with Synopsys Processor Designer and synthesis has been carried out. The implementation results are depicted in Table I.

D. Additional HW components

The challenging requirements of the physical-layer processing requests optimized and efficient components for data exchange and synchronization along with the hardware components for the computational part. Accordingly, a set of hardware components and related SystemC components has been developed that follows a common interface approach and hence can be easily connected to the other hardware components. For the SDR platform we have developed the following IP components as functional and timing approximate SystemC components that will be utilized for system-level design.

- Communication architectures
  - Simple handshake based point-to-point communication link

[2]Giving precise numbers is difficult since only area requirement and no gate count equivalent is given for the architecture in [32])

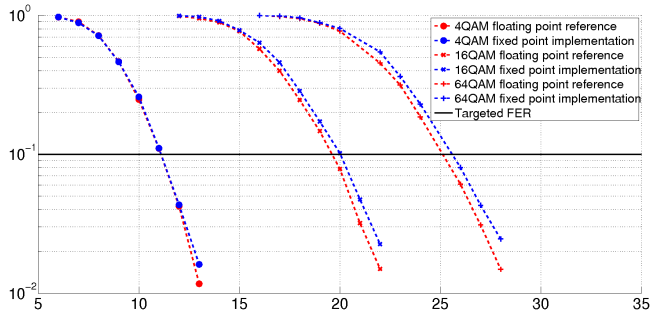


Fig. 4. FER comparison for 4x4 MIMO OFDM transceiver (Simulation parameters: detector: MMSE-QRD, ch. est.: LS, BCJR decoder: $r = 1/2$)

  - FIFO based point-to-point communication link
  - Bus based communication architecture
- Direct Memory Access (DMA) controller
- Interrupt controller
- Cyclic Redundancy Check (CRC) hardware component
- Channel decoder (LDPC) functional SystemC model based on component available from literature [35]
- Memories

E. Algorithmic Verification of individual IP components

Before inspecting the system level performance, it needs to be evaluated that quality of service (QoS) constraints in terms of bit-error-rate (BER) and frame-error-rate (FER) are kept by the targeted hardware implementations. For efficient verification, a simulation framework has been designed that supports evaluation of each single IP component within a complete system simulation. This becomes necessary as reliable information about QoS can only be obtained by a physical layer simulation. Accordingly, a framework for testing individual hardware components has been developed based on a C++ simulation library.

This allows for simple and quick analysis of individual functions of a transceiver or of the complete transceiver implementation by a plug & play fashion. To obtain a lower bound of the BER and FER, a reference C++ implementation based on the EigenLib [36] and scientific libraries such as the GNU scientific library [37] is co-simulated so that the effect of the hardware implementation can be easily analyzed.

The ASIP operates on a fixed point implementation hence sacrifices some precision. Therefore, an extensive algorithmic performance evaluation has been carried out by making use of the above described simulation testbed.

A channel simulation has been carried out featuring an AWGN and an i.i.d. Rayleigh Fading channel. Similar to the TGn-C channel model of WLAN 802.11n, channel condition with a 150ns power delay spread is assumed and a power delay profile that has an exponential 20dB drop. Both BER and FER of both fixed point implementation and floating point reference have been measured, while the more important FER is illustrated in Figure 4 that has been obtained by simulation of the ASIP and QRD based spatial equalizing for 4x4 antenna case and 16QAM constellation.

The obtained simulation results show only a minor degradation of the QoS performance. These simulation results are comparable to results found in literature [18].

V. System-Level Design and Virtual Platform

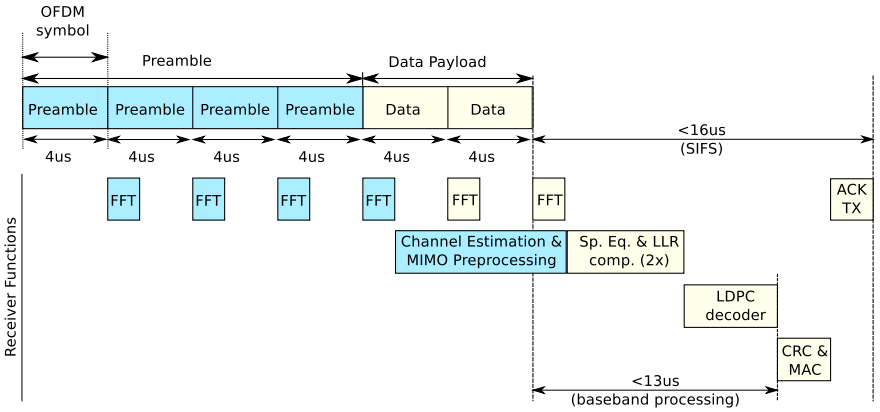
With the measured algorithmic QoS in range of a floating point implementation, a virtual platform (VP) was developed in parallel to the RTL implementation. Thanks to the quick development cycle of the SystemC based VP, which was developed with Synopsys Platform Architect (PA) [2], the following main objectives were targeted with the VP.

1) *Firmware development of data and control path.* High simulation speed and superior debug capabilities of the VP compared to traditional RTL simulation techniques allows for quick software development.
2) *Latency and throughput measurements prior to finalized RTL implementation.* Early availability within the design cycle has enabled us to measure the latency and throughput of the complete hardware architecture.
3) *Multicore debugging and performance investigation.* Extensive debug and tracing of the VP features allow efficient analysis of performance within the system. Especially, when it comes to synchronization and memory access conflicts this becomes of vital importance.
4) *Design space exploration.* Modular and advanced scripting features of PA support efficient design space exploration, e.g. evaluation of the number of required processor cores in the data path processing.

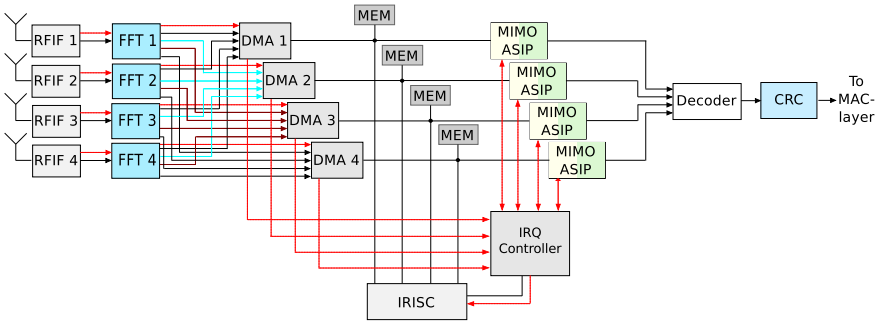
As a side note, the most important hardware components such as the ASIP, the FFT HW accelerator and the IRISC have been verified also on RTL, but verification of the complete system is not yet finished. Currently, we are confident that the developed software can be used with no or minor modifications in the final implementation.

Thanks to the efficient simulation speed of the cycle approximate VP, the system architecture as illustrated in Figure 5(b) has been developed. To achieve the WLAN 802.11n latency and throughput requirements, we have designed the architecture to cope with latency constraints for a minimum frame with two OFDM data symbols. As the Short Inter-Frame Space (SIFS) packet [38] needs to be send after 16 μs, this latency constraint is most severe during the SoC architecture design (see Fig. 5(a)). Accordingly, a system architecture with parallel processing in the data plane is mandatory. As illustrated, four FFT HW accelerators are operating in parallel for the four data streams that are received. To simplify the ASIP subsystem four configurable DMA components retrieve the data from the FFT components and move the data ordered by subcarriers to the ASIPs for MIMO processing. In total four ASIPs have been instantiated that operate on different subcarriers (each 13 subcarriers) to achieve the latency constraint for the 4x4 antenna mode. The computed LLR values are passed to the decoder and interleaver interface which decodes the received bits on the basis of codewords. Finally, after the cyclic redundancy check the received data is forwarded to the MAC layer.

The system level architecture was derived on basis of the individually measured implementations of the different



(a) Latency and throughput requirements for baseband processing for a frame (2 data symbols) with most demanding timing constraints



(b) System-level design of MIMO OFDM transceiver architecture for operation of up to 4x4 antenna constellation

Fig. 5. Execution time requirements and SoC platform including application mapping

transceiver functions. To confirm these theoretical considerations, which might underestimate the impact of synchronization and/or memory access conflicts, a system level simulation based on the VP has been carried out.

The results shown in Figure 6 verify that the computed execution behavior is reflected in the simulation environment. Compared to the primary computation of the individual transceiver functions, a comparable large overhead exists in accessing data memories. However, the measured execution characteristic for a worst case frame with only two data slots, shows that the throughput and latency constraints are sufficiently achieved by the current implementation. The obtained latency for passing the decoded bit stream back to the MAC layer is $12.13\mu s$. This leaves in total $3.87\mu s$ for the RF receive interface, the MAC layer and the sending of the SIFS packet, which is a common valid assumption [39].

VI. Conclusions and Outlook

In this paper we highlighted a complete SDR platform for a MIMO OFDM transceiver. The underlying hardware architecture is a heterogeneous MPSoC. Data plane processing is efficiently handled by a programmable ASIP that achieves significantly higher performance than general purpose DSPs, but surely has some performance loss compared to dedicated non-programmable solutions. For efficient support of FFT and LDPC decoding, specialized HW circuits are considered. For flexible control-plane processing we have included our in house designed RISC processor core. Furthermore, efficient support for inter-core synchronization is achieved by an interrupt controller and advanced interrupt handling of the ASIP. Our future work will concentrate on finalizing the RTL design and to optimize the performance of thee individual IP components and the complete system.

VII. Acknowledgement

The work described in this paper is based on many discussion within ICE and uses IP components, e.g. the IRISC processor, that have been developed at ICE. In particular, we would like to acknowledge contributions from Dominik Auras, David Kammler, Andreas Minwegen and Martin Witte.

References

[1] ACE Associated Computer Experts. *http://www.ace.nl/.*
[2] Synopsys Inc. *http://www.synopsys.com.*
[3] Transaction-level Modeling Standard (TLM-2.0) Accellera Systems Initiative. *http://www.accellera.org/.*
[4] Texas Instruments Inc. TMS320TCI6618, 2012.
[5] Freescale Semi. Inc. MSC8156: Six Core High Performance DSP, 2011.
[6] Nvidia Corporation. Icera soft-modem chipsets, 2012.
[7] Qualcomm. Gobi modems, 2012.
[8] ARM Ltd. ARM Processor Cores, 2012.
[9] Tensilica Inc. ConnX Baseband DSP and Xtensa Customizable Processor, May 2011.
[10] CEVA Inc. CEVA-X/XC DSP Cores for Advanced Wireless Communication, 2012.

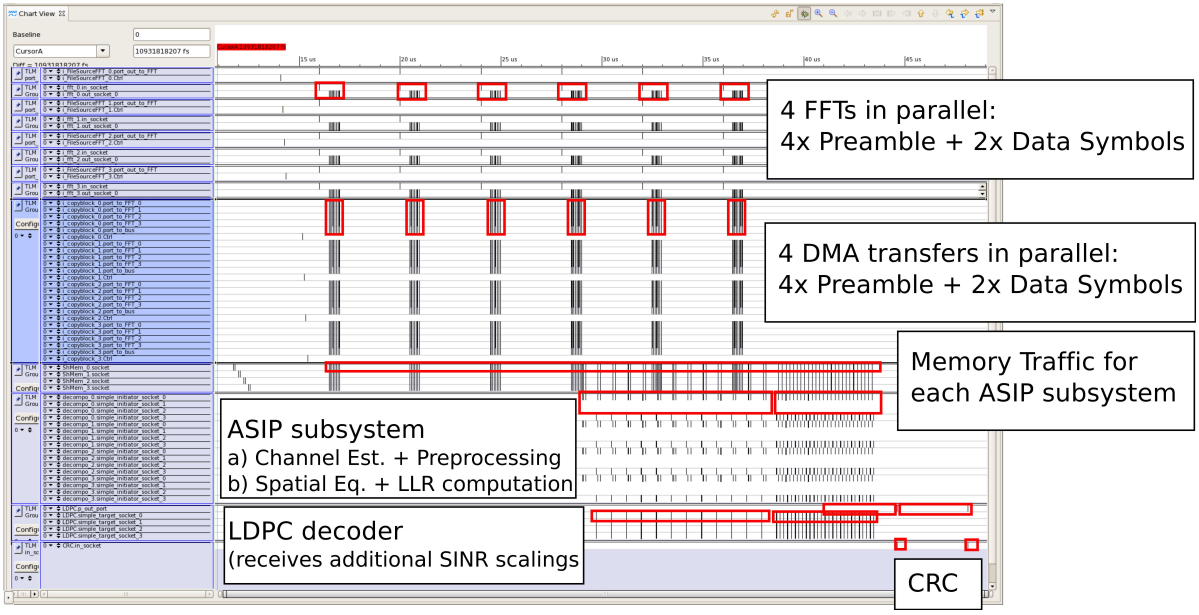


Fig. 6. Execution Trace of the Multi-Core Implementation

[11] STEricsson. Low-power Embedded Vector DSP, EVP VD32041 32-bit Embedded-Vector Processor for SoCs, Feb. 2009.

[12] STMicroelectronic. *STxP70-4 Core and Instruction Set Architecture. 8154580 Rev. B*. http://www.st.com/, June 2010.

[13] Ramakrishnan, V., et al. Efficient and Portable SDR Waveform Development: The Nucleus Concept. In *IEEE Military Communications Conference (MILCOM 2009)*, Boston, USA, Oct 2009.

[14] Castrillon, J., et al. Component-based Waveform Development: the Nucleus Tool Flow for Efficient and Portable SDR. In *2010 Wireless Innovation Conference and Product Exposition (SDR'10).*, Washington D.C., USA, Dec 2010.

[15] Torsten Kempf, Daniel Guenther, Aamir Ishaque and Gerd Ascheid. Mimo ofdm transceiver for a many-core computing fabric - a nucleus based implementation. In *SDR'11 - The Wireless Innovation Forum Conference on Communications Technologies and Software Defined Radio*, Washington D.C., USA, dec 2011.

[16] A.J. Paulraj, D.A. Gore, R.U. Nabar, and H. Bolcskei. An Overview of MIMO Communications - a Key to Gigabit Wireless. *Proceedings of the IEEE*, 92(2):198 – 218, feb 2004.

[17] Venkatesh Ramakrishnan, E. M. Witte, Torsten Kempf, David Kammler, Gerd Ascheid, Heinrich Meyr, Marc Adrat and Markus Antweiler. Efficient and portable sdr waveform development: The nucleus concept. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, pages 918–924, Boston, USA, oct 2009.

[18] E. Perahia and R. Stacey. *Next Generation Wireless LANs: Throughput, Robustness, and Reliability in 802.11n*. Cambridge University Press, Cambridge, UK, 2010.

[19] S. Coleri, M. Ergen, A. Puri, and A. Bahai. Channel estimation techniques based on pilot arrangement in ofdm systems. *Broadcasting, IEEE Transactions on*, 48(3):223 – 229, sep 2002.

[20] M.J. Flynn. On division by functional iteration. *Computers, IEEE Transactions on*, C-19(8):702 – 706, aug. 1970.

[21] C. Studer, S. Fateh, and D. Seethaler. Asic implementation of soft-input soft-output mimo detection using mmse parallel interference cancellation. *Solid-State Circuits, IEEE Journal of*, 46(7):1754 –1765, july 2011.

[22] M. Senst and Gerd Ascheid. How the framework of expectation propagation yields an iterative ic-lmmse mimo receiver. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)*, Houston, TX, USA, 2011.

[23] B. Hassibi and H. Vikalo. On the sphere-decoding algorithm i. expected complexity. *Signal Processing, IEEE Transactions on*, 53(8):2806 – 2818, aug. 2005.

[24] Gene H. Golub and Charles F. van Van Loan. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)(3rd Edition)*. The Johns Hopkins University Press, 3rd edition, October 1996.

[25] I.B. Collings, M.R.G. Butler, and M. McKay. Low Complexity Receiver Design for MIMO Bit-Interleaved Coded Modulation. In *International Symposium on Spread Spectrum Techniques and Applications, 2004*.

[26] F. Tosato and P. Bisaglia. Simplified soft-output demapper for binary interleaved cofdm with application to hiperlan/2. In *Communications, 2002. ICC 2002. IEEE International Conference on*, volume 2, pages 664 – 668 vol.2, 2002.

[27] Di Wu, Johan Eilert, and Dake Liu. Implementation of a high-speed mimo soft-output symbol detector for software defined radio. *Journal of Signal Processing Systems*, 63:27–37, 2011. 10.1007/s11265-009-0369-9.

[28] P.J. Lüthi. *VLSI Circuits for MIMO Preprocessing*. Series in microelectronics, v. 203. Hartung-Gorre, 2009.

[29] Dimpesh Patel, Mahdi Shabany, and P. Glenn Gulak. A low-complexity high-speed qr decomposition implementation for mimo receivers. In *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, pages 33–36, 24 2009-Yearly 27 2009.

[30] C. Senning, A. Staudacher, and A. Burg. Systolic-array based regularized qr-decomposition for ieee 802.11n compliant soft-mmse detection. In *Microelectronics (ICM), 2010 International Conference on*, pages 391 –394, dec. 2010.

[31] E. O. Brigham and R. E. Morrow. The fast fourier transform. *Spectrum, IEEE*, 4(12):63 –70, dec. 1967.

[32] Yu-Wei Lin and Chen-Yi Lee. Design of an fft/ifft processor for mimo ofdm systems. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 54(4):807 –815, april 2007.

[33] Optimum Semiconductor Technologies (OST) Sandblaster SB3500. *http://www.optimumsemi-tech.com/*.

[34] D. Liu, A. Nilsson, E. Tell, D. Wu, and J. Eilert. Bridging dream and reality: Programmable baseband processors for software-defined radio. *Communications Magazine, IEEE*, 47(9):134 –140, september 2009.

[35] C. Roth, P. Meinerzhagen, C. Studer, and A. Burg. A 15.8 pj/bit/iter quasi-cyclic ldpc decoder for ieee 802.11n in 90 nm cmos. In *Solid State Circuits Conference (A-SSCC), 2010 IEEE Asian*, pages 1 –4, nov. 2010.

[36] EigenLib: C++ template library for linear algebra. *http://eigen.tuxfamily.org/*.

[37] GSL GNU Scientific Library. *http://www.gnu.org/software/gsl/*.

[38] IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999) - Wireless LAN MAC and PHY Specifications. 12 2007.

[39] Raffaele Riva, STMicroelectronics, Agrate Brianza, Italy. LDPC Codes for WLAN IEEE 802.11n: an Overview. In *International Workshop on VLSI Architectures for LDPC Decoders, Pisa, Italy*, Oct. 2006.