SISO and MIMO OFDM Physical Layer Algorithms on a Heterogeneous Multiprocessor Platform - Implementations and Mapping Exploration

Venkatesh Ramakrishnan*, Marc Adrat†, Gerd Ascheid* and Markus Antweiler†

*Institute for Integrated Signal Processing Systems, RWTH Aachen University, Templergraben 55, 52062 Aachen, Germany

†Fraunhofer Institute for Communication, Information Processing & Ergonomics (FKIE), Wachtberg, Germany

***Abstract*—This paper presents the implementation and mapping exploration of the physical layer algorithms in a single input single output (SISO) and multiple input multiple output (MIMO) orthogonal frequency division multiplexing (OFDM) systems on a commercial heterogeneous multiprocessor hardware platform. We focus our investigations on the algorithms for implementing two computation-intensive components of a MIMO receiver, FFT and MIMO demapping. Several implementation variants for FFT and MIMO-preprocessing (sorted QR Decomposition) are evaluated in terms of performance properties like bit error rate (BER), latency, etc. The cascading effects due to the finite word length of the fixed point implementations on the receiver performance, in terms of BER, are studied. The generic SISO-OFDM system is implemented on a commercial heterogeneous MPSoC platform. Finally, the maximum achievable throughput as well as the latency issues for different spatial mapping configurations are presented.**

I. Introduction

Computation-intensive techniques like multiple input multiple output (MIMO), iterative processing, etc. are getting popular in order to meet the high throughput and low latency requirements of the emerging applications in the wireless communications. In parallel, the flexibility requirements to support new standards like LTE, WiMax, etc. in addition to the legacy standards like GSM, UMTS, etc. on a single mobile device are also growing strongly.

Efficiency in mobile systems is sought in several forms, e.g., energy, area, spectrum, etc. Algorithmic flexibility is a key factor to maximize spectrum and energy efficiency in wireless systems. For instance, energy efficiency can be improved if an option to choose sophisticated, complex algorithms in a bad channel and simple algorithms in a good channel exists. The contradictory nature of flexibility and efficiency requirements, when coupled with low-cost and reduced time-to-market constraints, make the development of a mobile device highly complicated. Software defined radios (SDRs) are getting prominence as potential candidates to meet these requirements of the mobile devices.

On the one hand, high computation and low energy needs cannot be achieved by using entirely general purpose processors (GPPs) as processing elements (PEs) for implementing future SDRs. On the other hand, pure application specific integrated circuit (ASIC)-based solutions, which are highly energy efficient, do not offer flexibility needed by SDRs. This makes heterogeneous multi-processor system-on-chips (MPSoCs), with PEs like digital signal processors (DSPs), application-specific instruction-set processors (ASIPs) and field programmable gate arrays (FPGAs), etc., in addition to GPPs and ASICs, as good candidates for implementing SDRs.

However, designing such a system is a challenging task. In order to decrease the system development-time, tools and methods for quick design space exploration and verification at an early stage are needed. This can be achieved by creating an executable specification and tools for performing constraint-aware mapping and evaluation. In essence, a standardized method for developing SDRs which can enhance reusability, among wireless standards and vendors, is needed to decrease the design costs.

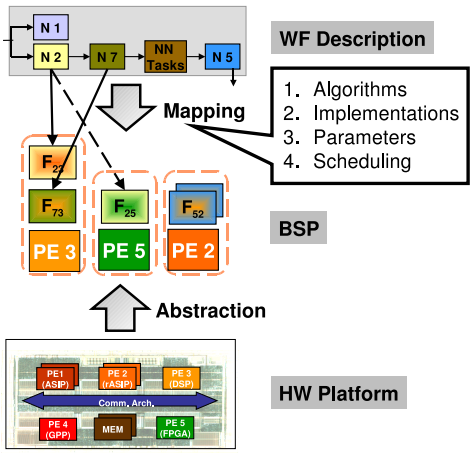


Fig. 1. *Mapping* in the Nucleus Methodology. NN, F and WF denote non-Nuclei, Flavor and waveform respectively.

Considering the above aspects, a novel library-based methodology for developing SDRs, known as the Nucleus methodology, that targets both flexibility and efficiency has been proposed [1]. The basis of the concept (Figure 1) is a library that is independent of waveforms[1] consisting of *Nuclei*[2] kernels. As one of the goals of the Nucleus approach is to

[1]In this context, the term *waveform* is used to refer a complete wireless standard like UMTS, GSM, etc.

[2]A Nucleus is a critical, computation-intensive, algorithmic kernel.

standardize the Nucleus library, i.e. the components and the interface of the components will be available to all the vendors. This can enable the vendors to provide *Flavors*[3] as a board support package (BSP) for a hardware (HW) platform.

As shown in Figure 1, a main feature of the Nucleus methodology is the abstraction of the HW platform to the Nucleus level. This is made possible due to the tight coupling of a Flavor to a PE. Therefore, mapping is reduced to identifying the best efficient implementation (i.e. a Flavor) for a Nucleus such that the waveform implementation not only meets the specification of the waveform but also maximizes developer requirements of a system like energy efficiency, etc. Note that in traditional approaches, mapping is from a waveform description to a PE. Tools are used for the complete development flow. However, in order to perform mapping and evaluation with tool-assistance, detailed information on the waveform and HW platform, among others, have to be provided to the tool. More details on the Nucleus methodology is given in [1].

The contributions of the paper are summarized as follows.

- A generic SISO-OFDM system and a MIMO-OFDM system are analyzed as case studies for the Nucleus methodology. In the case of the SISO-OFDM system, the focus of our investigations is given on the Flavors that are available as a BSP for performing fast Fourier transform (FFT). In the case of the MIMO-OFDM system, the emphasis of our analysis is given on the FFT and the MIMO processing. The implementations on the sorted QR decomposition (SQRD) in [2] are extended to cover the complete MIMO processing.
- Mapping exploration is done by using the Flavors available on the PEs (self implemented in case of MIMO processing) of a commercial heterogeneous HW platform, the small form factor (SFF) SDR development platform (DP) [3] from Lyrtech. Note that the term *mapping* is used in a special context in this paper (explained in Section V).
- The influence of the cascading effects due to the finite word length (FWL) on the bit error rate (BER) performance is studied.
- Real-time throughput is measured by implementing a selective combination of the SISO-OFDM implementation on the SFF SDR DP. Performance bottlenecks in the HW platform are identified.

Note that the goal of the case studies is not to showcase the highest throughput or the lowest latency that can be achieved with our implementation, but to identify the issues and bottlenecks while implementing a waveform on a heterogeneous HW platform, in the context of the Nucleus methodology.

The rest of the paper is structured as follows. Related work is presented in Section II. The system models of the SISO- and MIMO-OFDM system, used in our analysis, are explained in Section III. The algorithms that are used for implementing FFT and SQRD are detailed in Section IV-A followed by the implementation variants in Section IV-B. In Section V, the term *mapping* is defined, HW platform that is used for mapping is presented and the mapping exploration is explained. Performance of the algorithms in terms of BER and processing time is presented in Section VI. Finally, conclusions are drawn.

[3]A Flavor is an optimized and efficient implementation of a Nucleus.

II. Related Work

In [1], we have presented the Nucleus methodology. Recently, we have implemented two matrix decomposition algorithms for performing the SQRD on a DSP [2]. However, the focus in [2] has been entirely on the implementation of the SQRD. In this paper, we build on the prior work in [2] by covering the implementation of the complete MIMO processing/demapping.

Implementations of different waveforms on the SFF SDR DP already exist [4–8]. An experimental cognitive radio has been implemented in [4] and a throughput of 19.2 kilo bits per second (kbps) and 16 kbps for data and voice services respectively has been demonstrated. Though BER simulation results using the SFF SDR DP have been presented in [6], throughput analysis is missing. The authors in [8] report a latency of 43 millisec for looping back data from the receiver antenna to the transmitter antenna on the SFF SDR DP. However, throughput measurements nor mapping exploration have been performed in [8]. In [5], the physical layer of the terrestrial trunked radio (TETRA) waveform has been generated (from Simulink blocks) using automatic-tools, and implemented on the SFF SDR DP. In [5], a throughput of 72 kbps has been reported. As stated in [5], the overhead due to the automatic generation of the source code is unclear. Later, in [7], the same authors have presented the processing time measurements for the components of a SISO-OFDM system, with a focus on DSP. However, throughput measurements and mapping exploration are missing in [7], as well.

Several publications also exist on the implementations of some or all components in the physical layer of SISO- and MIMO-OFDM systems. For example, the authors in [9] have implemented the SISO-OFDM and MIMO-OFDM acoustic modems on a floating-point and a fixed point DSP and have performed timing measurements in real-time. Several matrix decomposition techniques like LU decomposition and QRD along with different MIMO detection schemes like zero forcing (ZF) and minimum mean squared error (MMSE) have been implemented on a real-time test bed consisting of DSPs and FPGAs in [10]. Algorithms, implementation complexities and inherent challenges for LTE terminal implementation are presented in [11] by dividing the receiver into inner and outer parts.

Due to the consideration of the Nucleus methodology [1], the requirements of our work is unique and different when compared to and therefore does not match exactly to the analysis done in the above mentioned works. Even though [4], [5] and [8] can be used for comparing throughput and latency, a fair comparison to our achieved results is still elusive due to the unavailability of the exact configuration details of the SFF SDR DP (including the communication link).

III. System Model

Figure 2 illustrates the block diagram of the SISO-OFDM system that is used in our analysis. The system is kept generic and consists of components present in a typical SISO-OFDM system. The transmitter is made up of the encoder, bit-wise interleaver, mapper and inverse FFT (IFFT). Though the insertion and the removal of the cyclic prefix (CP) is shown in Figure 2, the channel delay spread is assumed smaller than the guard interval. Perfect channel state information (CSI) is assumed at the receiver. The receiver is composed of FFT, demapper, de-interleaver and viterbi decoder.

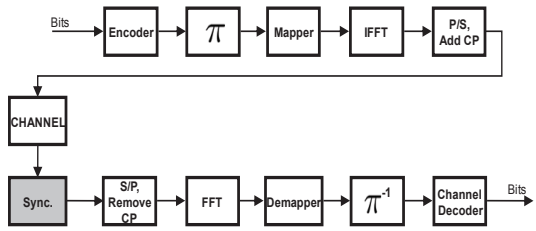


Fig. 2. Block diagram of a generic SISO-OFDM system simulation model. S/P and P/S represent serial-to-parallel and parallel-to-serial conversion respectively.

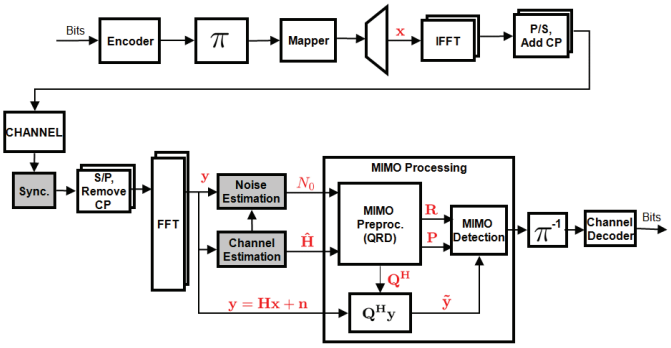


Fig. 3. Block diagram of a generic MIMO-OFDM system simulation model. S/P and P/S represent serial-to-parallel and parallel-to-serial conversion respectively.

Similarly, a generic MIMO-OFDM system with n_T transmitting and n_R receiving antennas is used for our analysis. Figure 3 illustrates the block diagram of the system. The received $n_R \times 1$ dimensional signal vector $\mathbf{y}$ can be expressed as

$$\mathbf{y} = \mathbf{Hx} + \mathbf{n} \tag{1}$$

where $\mathbf{H}$ is the $n_R \times n_T$ channel matrix, $\mathbf{x}$ is the $n_T \times 1$ transmitted signal vector and $\mathbf{n}$ is the $n_R \times 1$ noise vector with zero mean and variance σ_n^2 respectively. With the QRD of $\mathbf{H}$,

$$\mathbf{H} = \mathbf{QR} \tag{2}$$

and (1), the estimated value of $\mathbf{y}$ becomes

$$\tilde{\mathbf{y}} = \mathbf{Q^H y} = \mathbf{Rx} + \tilde{\mathbf{n}} \tag{3}$$

$\mathbf{Q}$ and $\mathbf{R}$ are unitary and upper triangular matrices respectively. $\mathbf{Q^H}$ represents the Hermitian transpose of $\mathbf{Q}$. The noise in the system is also taken into account for estimating the transmitted signal vector at the receiver side. This system, known as the MMSE solution, offers BER performance when compared to ZF solution, which does not consider the noise n [12]. Therefore, an *augmented channel matrix* $\bar{\mathbf{H}}$, is obtained by

$$\bar{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sqrt{\frac{n_T}{E_s}}\sigma_n \mathbf{I}_{n_T} \end{bmatrix} \tag{4}$$

where E_s is the transmitted signal power and $\mathbf{I}_{n_T}$ denotes a $n_T \times n_T$-dimensional identity matrix.

The task of the MIMO processing block is to retrieve the spatially independent parallel streams from the received super-imposed stream at the MIMO receiver. The functionality of MIMO processing can be divided into two parts: MIMO preprocessing and MIMO processing. As SQRD provides better BER performance and reduces the complexity of MIMO processing, when compared to QRD without sorting, it is used in this work. Therefore, the SQRD of $\bar{\mathbf{H}}$ now becomes

$$\bar{\mathbf{H}} = \bar{\mathbf{Q}}\mathbf{R}\mathbf{P}^T = \begin{bmatrix} \mathbf{Q}_a \\ \mathbf{Q}_b \end{bmatrix} \mathbf{R}\mathbf{P}^T \tag{5}$$

including $\mathbf{P}^T$, the transpose of the permutation matrix $\mathbf{P}$. $\mathbf{Q}_a$ and $\mathbf{Q}_b$ are the sub-matrices of $\mathbf{Q}$. Successive interference cancelation (SIC) detection is performed in n_T iterations. In each iteration, the first detected symbol is used for subsequent detection. Perfect channel state information is assumed.

IV. Analysis

In this section, the algorithms and implementations of FFT and SQRD on the PEs, C64x+ DSP [13] and Virtex-4 FPGA [14] are presented. Note that the PEs, C64x+ and Virtex-4, are present in the SFF SDR DP, the HW platform used for our case study.

A. Algorithms

*1) **FFT**:* Two algorithms are used for implementing the FFT: radix-4 and mixed-radix. In general, mixed-radix offers a combination of radix-2 and radix-4 for implementing the FFT in the sub-stages. The main operation for implementing the FFT is the so-called butterfly operation. As multiplication and addition are the main operations for performing FFT, it is well suitable for implementation on a DSP.

*2) **SQRD**:* Two algorithms are used for implementing the MMSE-SQRD (referred as SQRD): modified Gram Schmidt (MGS) and Givens rotation (GR). The basic operations that are required for performing SQRD using MGS are: square root, inverse and multiply. Therefore, it is suitable to implement the MGS algorithm on a DSP.

Vectoring and rotation are two main operations of implementing the SQRD operation using GR. In a *vectoring* operation, a complex-valued matrix element is rotated in such a way that the y-component becomes zero. In the *rotation* operation, a complex-valued matrix element is rotated by a

Case	PE	Name	Impl. Algo.	Input width (bits)	Twiddle width (bits)	Internal scaling	Processing time (cycles)	
							$n_{FFT} = 64$	$n_{FFT} = 1024$
1	C64x+	DSP_fft16x16	Mixed	16	16	No	1989	15489
2		DSP_fft16x16_imre	Mixed	16	16	No	2606	17682
3		DSP_fft16x16r	Mixed	16	16	Yes	1564	16695
4		DSP_fft16x32	Mixed	16	32	No	6399	60859
5		DSP_fft32x32	Mixed	32	32	No	5521	67663
6		DSP_fft32x32s	Mixed	32	32	Yes	5371	67561
7	Virtex 4	Stream	Radix-4	24	8	Yes	414	4324

TABLE I
LIST OF THE INVESTIGATED FFT FLAVORS ALONG WITH THEIR KEY PARAMETERS.n_{FFT} REPRESENTS THE SIZE OF THE FFT.

specific angle. Note that MMSE-SQRD is performed in a series of steps (both in the cases of MGS and GR). Detailed explanation on obtaining the SQRD using the MGS and GR algorithms can be found in [2].

B. Implementations

1) ***FFT****:* Table I lists the Flavors that are considered in our analysis. They are a part of the library consisting of efficient implementations from TI [15] and IP cores from Xilinx [16]. All the Flavors for C64x+ use mixed-radix, a combination of radix-4 and radix-2 (depending on the FFT size, n_{FFT}), for implementation. Scaling is done by 2 at each radix-4 stage (referred as internal scaling) for two Flavors of C64x+ in Table I, DSP_fft16x16r and DSP_fft32x32s. For other FFT Flavors in C64x+ DSP, there is no internal scaling. The input data has to be scaled depending on the internal scaling in order to prevent overflow. As the rest of the FFT-Flavors listed in Table I, except the above two mentioned FFT Flavors, do not support internal scaling, input has to be scaled by 10 bits when $n_{FFT} = 1024$. This results in very bad BER performance for similar FFT Flavors, as can be seen in the Section VI. All the FFT-Flavors of C64x+ use rounding. The FFT IP core that is considered for our investigations has internal scaling and is used with a twiddle width of 8 bits. Note that the FFT implementation is also used for implementing the IFFT (conjugation of the input data and scaling of the output data is done, in addition).

2) ***SQRD****:* In contrast to the Flavors for FFT which is readily available as a library, Flavors for performing SQRD had to be implemented. The hand-optimized assembly code for SQRD in [2] is taken and extended to implement the complete MIMO processing block. In other words, two more components seen in Figure 3 other than MIMO processing, namely, SIC detection and $Q^H y$ are implemented for this paper. Table II provides an overview on the different Flavors that have been implemented for performing MIMO processing. Note that 16-bits are sufficient for obtaining close-to floating-point BER performance. Since 32-bit variants consume more processing cycles, only the 16-bit variants are considered for the implementation of the MIMO processing. Furthermore, other components of the MIMO-OFDM system is implemented by duplicating the reusable components from the SISO-OFDM, e.g. FFT, IFFT, etc. Note that a 2×2 MIMO system is considered.

For implementing the square root and inverse operations in the MGS algorithm, optimized assembly code which is part of the IQMath library [13] from TI is used. IQMath library is a collection of highly-optimized and high-precision mathematical functions for C64x+ DSP. As listed in Table II, three variants for performing SQRD using GR are implemented. The first variant uses co-ordinate rotation digital computer (CORDIC) kernel for both vectoring and rotation operations. CORDIC algorithm, which performs a vector rotation in a sequence of micro rotations with fixed step size, is used widely for several applications in SDRs [17]. As the implementation that uses CORDIC for rotation consumes more processing time, a hybrid variant which uses multiplication for rotation operation is implemented. The last variant, GR_plain, uses the functions: sine, cosine and arctangent, available in the IQMath library for implementing the SQRD. Note that the coefficients in Table II denote the look-up tables needed by the implementations; in case of CORDIC, the step size for microrotations of the angle is stored as a look-up table and in case for MGS, the IQMath library uses look-up tables for implementing the used functions [13].

V. MAPPING EXPLORATION

Mapping, in the context of the Nucleus methodology, is the process of selecting a suitable Flavor for a Nucleus that meets the requirements of a waveform. However, for performing this selection, several factors like the limitations of the HW platform have to be considered. For example, let us take the scenario in Figure 1, where two Nuclei, N_2 and N_7, are neighbors and exchange data with each other. If a communication link does not exist between PE_5 and PE_3, then the combination of the Flavors, F_{25} and F_{73}, which are coupled to PE_5 and PE_3 respectively, cannot be chosen. This is indicated by dotted arrow in Figure 1. Therefore, F_{23} may have to be selected instead of F_{25}. This leads to define the term *mapping*, in our context, representing the following tasks:

- **algorithm mapping** i.e. the choice of an algorithm for the implementation of the functionality, e.g. the selection of MGS or CORDIC algorithm for implementing the SQRD.
- **implementation mapping** i.e. the selection of a Flavor (bound to a PE) depending on the method for implementing an algorithm of a component, e.g. the selection of the CORDIC only or the hybrid approach for implementing SQRD using GR.

Case	Alg.	Implementation Variant			Input width (bits)	Coeff. width (bits)	Processing time (cycles)	
		Name	Vectoring	Rotation			SQRD	MIMO Proc.
1	GR	GR_co_32x32	CORDIC	CORDIC	32	32	3043	-
2		GR_co_16x16			16	16	2627	4361
3		GR_hy_32x32		Mult.	32	32	1833	-
4		GR_hy_16x16			16	16	1563	3300
5		GR_plain_16x32	Atan, Sine, Cosine		16	32	2365	4136
6		GR_plain_32x32			32	32	2530	-
7	MGS	MGS_16x32	-	-	16	32	340	1941
8		MGS_32x32	-	-	32	32	598	-

TABLE II
LIST OF THE IMPLEMENTATION VARIANTS USED FOR SQRD IMPLEMENTATION OF MIMO PREPROCESSING ON DSP ALONG WITH THEIR KEY PARAMETERS; PROCESSING TIME INDICATES THE CYCLES FOR PERFORMING THE SQRD OF ONE OFDM SUB-CARRIER IN A 2×2 MIMO SYSTEM; ALG., COEFF., PROC. AND MULT. REPRESENT ALGORITHM, COEFFICIENTS, PROCESSING AND MULTIPLIER RESPECTIVELY.

- **parameter mapping**, i.e. the selection of the parameters of a Flavor, e.g. the selection of 16- or 32-bit as input data-width.
- **temporal mapping**, i.e. the execution order or scheduling, if more Flavors are run on a PE, e.g. static or dynamic.

For performing the above tasks, several considerations have to be made in the process of *mapping*. Some key considerations are listed in the next section.

A. Considerations

Broadly, the properties that affect the performance of a waveform implementation can be divided into three: processing properties like latency, throughput and energy efficiency; implementation properties like area and memory; and algorithmic properties like BER. Care must be taken in order to minimize the effects of the following items on the above mentioned performance properties:

- FWL effects due to the use of fixed point implementations and their associated influence on the BER performance.
- The cascading effects of a Flavor on the other components of the waveform implementation. Though it is easy to isolate and calculate the effects of Flavors on properties like latency (which is additive), it is very difficult to isolate the effects of Flavors on the algorithmic properties like the BER.
- Constraints like the throughput and latency in a communication link between two PEs must be analyzed in order to identify the bottlenecks in the system.
- Due to the optimized nature, Flavors exhibit interface related constraints like specific data width and *Q format*. [4] When the interface of two neighbouring Flavors do not match, additional logic in the form of glue-code is needed to remove the incompatibility. However, the overhead on latency due to the glue-code need to be considered while mapping.

The presence of numerous critical loops with varying waveform constraints, several Flavors and performance properties like BER make the "mapping" process challenging.

[4]Q notation is used to represent fixed point numbers. For example, Q0.15 or Q15 represents the format where a 16-bit data contains zero integer bit, one sign bit and 15 fractional bits.

B. HW Platform

The SFF SDR DP is used as the HW platform for our case study. It consists of a TMS320DM6446 [18] system-on-chip (SoC) with a ARM926 core [19] running at 297MHz and a TMS320C64x+ DSP [15] core from Texas Instruments running at 594MHz and a Virtex-4 SX35 FPGA [14] from Xilinx. Figure 4 illustrates the baseband processing part of the SFF SDR development platform. The ARM9 GPP core hosts the Green hills INTEGRITY real time operating system [20]. The GPP loads the application on the DSP and the FPGA and manages the host-to-DSP communication. As shown in the Figure 4, the DSP and the FPGA communicate using the video processing sub-system (VPSS) port, consisting of the video processing back end (VPBE) and the video processing front end (VPFE) from TI. The DSP subsystem has the following memory hierarchy:

- 32KB L1 program (L1P)/cache (up to 32KB)
- 80KB L1 data (L1D)/cache (up to 32KB)
- 64KB unified mapped RAM/cache (L2)

In addition, TMS320DM6446 can access a DDR2 SDRAM bank of 128 MB. This DDR2 SDRAM runs at 324 MHz and is connected to the DDR2 bus of the DMP SoC [18].

As the focus of our analysis is only on the modem part of the waveforms, the transmission and reception over-the-air interface is avoided. The throughput (and the latency) measurements are computed in real-time using the operational clock frequency of the FPGA.

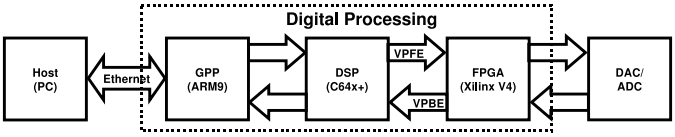


Fig. 4. Block diagram of the baseband processing part of the SFF SDR DP.

A main motivation of using the SFF SDR DP for our analysis is the heterogeneous nature of the HW. Theoretically, with the presence of a GPP, DSP and FPGA, the SFF SDR DP offers a possibility to perform mapping exploration and identify the bottlenecks like computation and communication.

However, due to the special software requirements of the GPP, it is not used for running the physical layer implementation. The host PC is used instead.

VI. Results

In this section, we present the performance of the implementations in terms of processing time, BER and throughput that is achieved in real-time on the prototyping HW. The cascading effects due to FWL are presented. Finally, the results are discussed. For all the measurements in this section, the TMS320C6000 C Compiler v6.08 with "-o3" compiler option has been used. Furthermore, the processing time has been measured by mapping the program and data code onto the SDRAM and with the cache memory enabled. Note that a better performance can be achieved by mapping the program and data directly onto the L1 cache memory. However, the size of the program and data code, when considering the complete modem of the SISO-OFDM or MIMO-OFDM system, does not permit this scenario. In case of a MIMO system, we have considered two transmit and two receive antennas.

A. Processing Time

Processing time in terms of the processing cycles for the FFT Flavors are listed in Table I. A considerable difference in the processing time is seen between the 16-bit to 32-bit Flavors. It is important to note that the processing time in [15] represents the CPU cycles only, without considering memory overhead. Table II illustrates the processing time, including the memory overhead, for the MIMO processing using the different SQRD implementations. Note that the two operations of MIMO processing other than SQRD, namely, the SIC detection and $Q^H y$ consume 770 and 570 cycles respectively. However, the implementation of these two operations is in fixed-point C. As mentioned in Section IV-B2, only the 16-bit implementations are used for implementing the MIMO processing due to the same BER performance when compared to the 32-bit implementations.

The overhead due to the glue-code is calculated using the Flavors listed in Table I as an example. Note that in practical systems a direct connection between IFFT and FFT does not exist. However, interface constraints are prevalent in existing systems. For example, it is typical in efficient implementations on DSP to expect the real and imaginary parts of the input data in a consecutive order (referred as IR). Also, it is typical for IP cores on FPGAs to have independent streams for real and imaginary components (referred as I-R). This conversion is shown in case 1 in Table III. The reordering of data from real followed by imaginary (RI) to imaginary followed by real parts (IR), as required for FFT Flavor in Table I is represented in case 2 in Table III. Similarly the requirements of input scaling, bit-width conversion and Q-format conversion can be seen in all the FFT Flavors listed in Table I. Thanks to the in-house development of the implementation variants for MIMO processing, the glue-code is avoided.

Table III gives an overview on the overhead in terms of processing cycles for implementing the glue-code. Note that the measurements use the stand-alone version of the implementation, where the data is read from the memory and written back after the glue-code operations. While looking at only the CPU cycles, the overhead due to memory read/write operations seems high. In general, this overhead can be reduced by combining the glue-code with the other computation logic thereby reducing the memory operations. This reduction effect can be well seen while combining several glue logic operations into one (case 5 in Table III). The reduction, in percentage, when compared to the sum of the processing cycles due to the individual overheads, is as well listed.

Case	Glue logic	Processing time (cycles)			
		only CPU		CPU and Memory	
		64	1024	64	1024
1	IR to I-R	141	2061	2247	21880
2	IR to RI	44	524	1344	15834
3	I/P scaling	49	529	727	7981
4	Q-Format	75	1035	1378	1660
5	1+3+4	111 (41%)	1551 (43%)	2477 (57%)	26086 (83%)

TABLE III
PROCESSING TIME IN TERMS OF CYCLES FOR THE GLUE-LOGIC; ONE OFDM SYMBOL WITH $n_{FFT} = 64$ AND 1024. P.T. AND I/P REPRESENT PROCESSING TIME AND INPUT RESPECTIVELY.

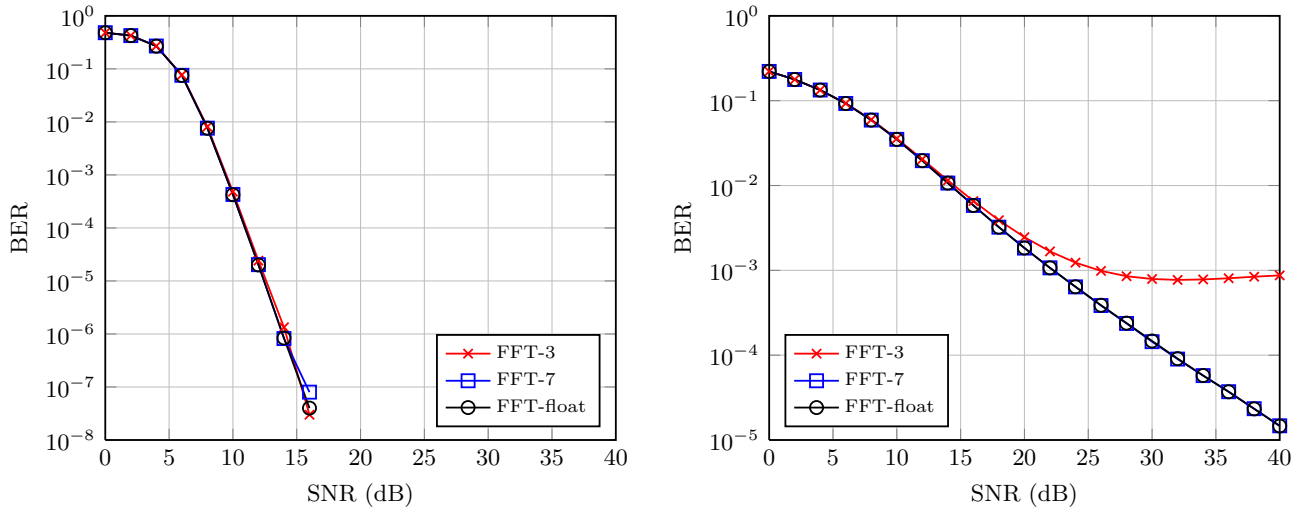
B. BER

Additive white Gaussian noise (AWGN) with Rayleigh fading is used for the simulations of the systems. A non-systematic $\{171, 133\}_8$ convolutional code with code rate r = 1/2 is used. QPSK with Gray mapping is chosen. Every subcarrier is used for data symbol. The same Flavor is used in both FFT and IFFT for simulating a case in Table I. A 2×2 MIMO with a MMSE based MIMO processing and SIC as the MIMO detection scheme is implemented.

Figure 5 illustrates the BER performance of the Flavors for FFT with the different cases listed in Table I and the floating point implementation of the GR. It is important to note that, with a floating point FFT implementation, the floating-point SQRD implementations using MGS and GR yield the same BER performance.

To analyze the performance of the Flavors without the influence of coding, we illustrate both the coded and uncoded BER performance in Figure 5(b). The floating point implementations of FFT and SQRD are used as the base-line for comparison in both coded and uncoded BER curves.

Figure 6 illustrates the BER performance using a selected list of the FFT Flavors in Table I and using the Flavor, GR_co_16x16 in Table II, with different CORDIC rotations. As the rest of the components in the MIMO-OFDM simulation model yield close to floating-point BER performance, their influence of BER can be ruled out. Both coded and uncoded versions are illustrated in Figure 6. The floating point implementations of FFT and SQRD are used as the base-line for comparison in both coded and uncoded BER curves. Note that with 6 CORDIC iterations, close to floating-point performance is achieved.



(a) Coded; $r = 1/2$, Non-systematic conv. code $\{171, 133\}_8$.

(b) Uncoded.

Fig. 5. BER vs. E_b/N_o for a few FFT Flavors listed in Table I and the floating point implementation of the GR; QPSK, Gray mapping, $n_{FFT} = 1024$.

1) Cascading Effects: The error propagation due to the FWL effect from the FFT Flavor to the MIMO processing can be clearly seen between Figure 5 and Figure 6. In both the figures, the BER error floor is predominantly determined by the FFT Flavors and by the number of CORDIC iterations. However, the influence of FFT Flavors is remarkable. Note that in Figure 6, more CORDIC iterations do not improve the BER performance.

C. Hardware Prototype

Table IV lists the throughput that is achieved while mapping the implementation of the SISO-OFDM system on the SFF SDR DP. The implementation on the DSP of the other components in the system, except FFT and IFFT, is in C. For FFT and IFFT, the Flavors listed in Table I are used. For the implementation on the FPGA, the IP cores from Xilinx are used. Note that the channel synchronization (marked grey in Figure 2) is not implemented.

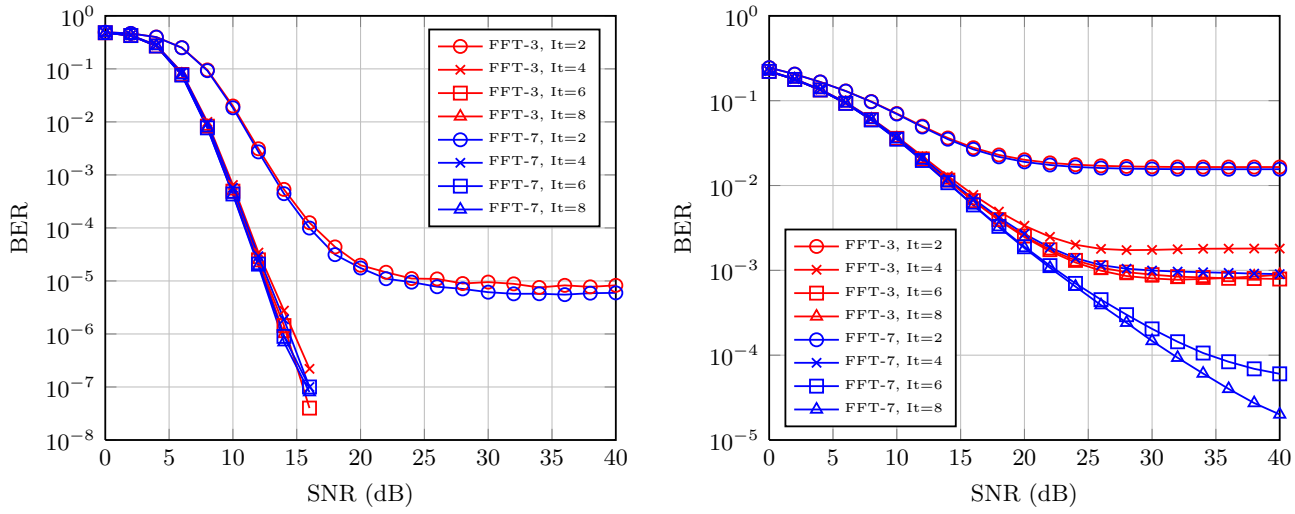
The first experiment is to perform a loop back, where data is transferred from the host to the FPGA via the DSP. The same data is routed back to the host from the FPGA through the DSP. Note that the communication between the host and the DSP is facilitated by ARM9 GPP core. However, GPP is treated as a black box and abstracted as a part of communication between the host and the DSP. The SFF SDR DP is controlled by the host using a state machine that runs on both the host and the DSP. The FPGA is controlled by the DSP via registers.

Since the throughput for the host-DSP-FPGA is found very low (see Table IV), the throughput between the DSP and the FPGA is calculated to identify the bottleneck. From the analysis, the bottleneck is found to be the communication between the host and the DSP. Therefore, host is not considered further in our mapping exploration. When both the transmitter and receiver of the SISO-OFDM system is implemented on the FPGA, the throughput is reduced slightly when compared to the unloaded scenario. As seen in Table IV, the DSP becomes the bottleneck when it is loaded with the functionalities. The throughput worsens, when compared to the unloaded DSP, by approximately 2.5 times when the transmitter is running on the DSP and by 56 times when both the transmitter and the receiver are running on the DSP. When the computation intensive kernels, FFT and IFFT are mapped onto the FPGA, instead of DSP, a throughput of approximately 70 kbps is gained. Similarly, a throughput of 90 kbps can be gained if the Viterbi decoder is mapped onto FPGA.

S.No.	Host	DSP	FPGA	Throughput (Mbps)
1	S/R	-	-	0.93
2	-	S/R	-	5.71
3	-	S/R	Tx+Rx	5.05
5	-	S/R+Tx (rest)	IFFT + Rx	2.10
4	-	S/R+Tx+FFT	Rx(rest)	2.03
6	-	S/R+Tx+Rx(rest)	Rx(Viterbi)	0.99
7	-	S/R+Tx+Rx	-	0.09

TABLE IV
REAL-TIME THROUGHPUT MEASUREMENTS OF THE SISO-OFDM SYSTEM. S/R, TX AND RX DENOTE SENDING/RECEIVING PAYLOAD, TRANSMITTER AND RECEIVER RESPECTIVELY. $n_{FFT} = 64$.

Note that our implementation of the SISO-OFDM system offers full flexibility and allows the mapping each of the components shown in Figure 2 individually onto the host, the DSP or the FPGA. However, as the main goal of the analysis is to identify the bottlenecks between computation and communication, exhaustive mapping exploration is not done. The real-time measurement numbers for a simple SISO-OFDM system are not satisfactory enough to perform a similar exploration for MIMO-OFDM.



(a) Coded; $r = 1/2$, Non-systematic conv. code $\{171, 133\}_8$.

(b) Uncoded.

Fig. 6. BER vs. E_b/N_o for a few FFT Flavors listed in Table I and for the Flavor GR_co_16x16 in Table II with different CORDIC rotations; QPSK, Gray mapping, $n_{FFT} = 1024$.

D. Discussion

From our results on the cascading effects (Figure 5 and Figure 6), it is interesting to see that the losses due to the FWL effects of the implementations add (approximately) when measured in the dB scale. As seen in our BER results, if we select a FFT Flavor that performs bad in terms of BER, the error due to the FWL is propagated to the entire system. Due to this, in spite of more CORDIC iterations, a better BER is not reached. However, the FWL effects due to the CORDIC iterations are additive to the effects of the FFT Flavor. This allows to segregate the effects from the individual components in the mapping process, at least in non-iterative systems.

The implementation of the waveform on a HW platform for real-time demonstration has brought several critical issues into the lime light. In general, such issues cannot be easily seen or even impossible to see while using simulation models. One key issue is the ambiguity in the user manual of the prototyping platform leading to misinterpretation. For example, there is a huge mismatch between the latency and throughput numbers that are presented in [21] and our results shown in Table IV. Even though the black-box approaches for the interface between the host and the DSP and the DSP and the FPGA, makes the communication simpler, it also makes the identification and isolation of the bottleneck complicated. For example, it is normal to enable the cache memory in the DSP to improve the performance. However, the DSP to the FPGA communication in SFF SDR DP did not work, initially, when the cache memory is enabled in the DSP while using the default functions from the BSP for the communication. After some efforts, the special way with which the DSP and FPGA communication can be made to work with enabled cache has been found. Also, the throughput is drastically reduced while performing the loop-back between the host, the DSP and the FPGA. Due to the non-availability of the source code and the usage of the black-box approach, the cause of this reduction cannot be isolated and solved. Therefore, more transparent and clear approaches are needed, as highlighted in [1].

Accurate or even high-abstraction models with test cases can be very useful to avoid the misinterpretation of the user manuals. Such models should clearly distinguish the PEs, communication and memories to separate and identify the bottlenecks in a HW platform.

Though the real-time throughput measurements are far better when compared to the numbers reported in Section II, a one-to-one comparison is very difficult due to the lack of exact configuration details of the HW platform.

The extent of the performance loss, in terms of the processing time, that can be expected while targeting full reusability in implementations, can be seen in Table II. For example, the implementation of SQRD using CORDIC offers full flexibility, in terms of CORDIC iterations. Furthermore, due to the application of CORDIC algorithm for implementing several functionalities [17], the implementation offers high reusability. However, when compared to a dedicated SQRD implementation using MGS, the CORDIC implementations is slow by a factor of 2. Depending on the requirements of a waveform, e.g. latency, fully reusable implementations or hybrid variants can be attractive as well. Therefore, the trade-offs between reusability and performance need to be considered while developing waveform implementations and mapping.

In the context of Nucleus methodology, implementation details of a Flavor, like the FWL effects, are hidden from the system developer. Only the key parameters for configuring a kernel like the length of FFT, number of micro-iterations in CORDIC, implementation algorithm, etc. are made visible. For a given scenario of desired BER, throughput, etc., the mapping tool should consider aspects like FWL effects while identifying

suitable Flavors for implementing a waveform. In order to do this automatically, the critical issues that can be present in a BSP, like shown above using SFF SDR DP as a case study, should be addressed. Furthermore, the different categories in the mapping exploration should seen as the division of the mapping problem into manageable sub-problems. It simplifies the mapping process and also aids in the efficient development of tools that are modular.

VII. Conclusions

The physical layers of a generic SISO- and MIMO-OFDM system have been implemented on a heterogeneous hardware platform, SFF SDR DP, as a case study for the Nucleus methodology. Emphasis has been given on the two computation-intensive kernels in a MIMO-OFDM system: FFT and MIMO-processing. Different implementations that vary in algorithms and methods have been studied in terms of performance like the processing time and the BER (using the FWL effects). The cascading nature of the error propagation due to the FWL effects has been analyzed. The overhead in terms of processing time due to the glue-code, which is needed for meeting the interface-related constraints of the efficient implementations, is calculated. Finally, the modem part of the physical layer is mapped onto the SFF SDR DP in several ways and the maximum achievable throughput in real-time is obtained. Bottlenecks of the implementation on the SFF SDR DP have been identified. The issues hampering the overall waveform development have been discussed and suggestions have been provided for improvement.

VIII. Acknowledgement

This research project was performed in the Ultra High-Speed Mobile Information and Communication (UMIC) research centre under the support of the Technical Center for Information Technology and Electronics (WTD-81), Germany.

References

[1] V. Ramakrishnan, E. M. Witte, T. Kempf, D. Kammler, G. Ascheid, R. Leupers, H. Meyr, M. Adrat, and M. Antweiler, "Efficient and portable SDR waveform development: The nucleus concept," in *IEEE Military Communications Conference (MILCOM 2009)*, 2009.

[2] V. Ramakrishnan, T. Veerkamp, G. Ascheid, M. Adrat, and M. Antweiler, "Matrix Decomposition Algorithms for MIMO receivers: Flexibility vs. Efficiency Tradeoffs in a Library-based Tool-Assisted SDR Development," in *2011 Wireless Innovation Forum European Conference on Communications Technologies and Software Defined Radio*, June 2011.

[3] Lyrtech, "Small form factor SDR development platforms," November 2009. http://www.lyrtech.com/

[4] P. Amini, E. Azarnasab, S. Akoum, and B. Farhang-Boroujeny, "An experimental cognitive radio for first responders," in *3rd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2008. DySPAN 2008.*, oct. 2008, pp. 1 –6.

[5] S. Nagel, D. Epple, and F. K. Jondral, "Implementing the tetra physical layer on lyrtechs sff sdr development platform," in *2008 Software Defined Radio Technical Conference and Product Exposition*, October 2008.

[6] M. Kadhim and W. Ismail, "Implementation of WIMAX IEEE802.16d baseband transceiver on multi-core software-defined radio platform," *WTOC*, vol. 9, pp. 301–311, May 2010.

[7] S. Nagel, M. Schwall, and F. K. Jondral, "Performance overhead with high level waveform development," in *2010 European Reconfigurable Radio Technologies Workshop*, June 2010.

[8] Z. Chen, N. Guo, Z. Hu, and R. Qiu, "Channel state prediction in cognitive radio, part I: Response delays in practical hardware platforms," in *2011 Proceedings of IEEE Southeastcon*, march 2011, pp. 45 –49.

[9] H. Yan, S. Zhou, Z. Shi, J.-H. Cui, L. Wan, J. Huang, and H. Zhou, "DSP implementation of SISO and MIMO OFDM acoustic modems," in *OCEANS 2010 IEEE - Sydney*, may 2010, pp. 1 –6.

[10] T. Haustein, A. Forck, H. Gäbler, V. Jungnickel, and S. Schiffermüller, "Real-time signal processing for multiantenna systems: Algorithms, optimization, and implementation on an experimental test-bed," *EURASIP Journal on Applied Signal Processing*, vol. 2006, 2006.

[11] J. Berkmann, C. Carbonelli, F. Dietrich, C. Drewes, and W. Xu, "On 3G LTE Terminal Implementation - Standard, Algorithms, Complexities and Challenges," in *International Wireless Communications and Mobile Computing Conference, 2008. IWCMC '08.*, aug. 2008, pp. 970 –975.

[12] D. Wubben, R. Bohnke, V. Kuhn, and K.-D. Kammeyer, "Near-maximum-likelihood detection of MIMO systems using MMSE-based lattice reduction," in *2004 IEEE International Conference on Communications*, vol. 2, june 2004, pp. 798 – 802 Vol.2.

[13] *TMS320C64x+ IQmath Library User's Guide*, Texas Instuments, December 2008.

[14] Xilinx, "Virtex-4 family overview," August 2010. http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf

[15] Texas Instruments, "TMS320C64x+ DSP little-endian DSP library programmer's reference," March 2008. http://focus.ti.com/lit/ug/sprueb8b/sprueb8b.pdf

[16] Xilinx, "Fast Fourier transform v6.0," September 2008. http://www.xilinx.com/support/documentation/ip_documentation/xfft_ds260.pdf

[17] J. Valls, T. Sansaloni, A. Perez-Pascual, V. Torres, and V. Almenar, "The use of CORDIC in software defined radios: A tutorial," *Communications Magazine, IEEE*, vol. 44, no. 9, pp. 46 –50, 2006.

[18] Texas Instruments, "TMS320DM6446 digital media system-on-chip: Data sheet," September 2010. http://focus.ti.com/docs/prod/folders/print/tms320dm6446.html

[19] *ARM926EJ-S: Technical Reference Manual*, E ed., ARM Limited, June 2008.

[20] Green Hills Software, "INTEGRITY RTOS, The MULTI Integrated Development Environment," http://www.ghs.com/, April 2011.

[21] Lyrtech, "Small form factor SDR evaluation module/ development platform - User's guide," October 2007.