Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 312614, 8 pages
doi:10.1155/2008/312614

Research Article

Design and Implementation of a Hardware Module for MIMO Decoding in a 4G Wireless Receiver

Alberto Jiménez-Pacheco,[1] Ángel Fernández-Herrero,[2] and Javier Casajús-Quirós[1]

[1] *Departamento de Señales, Sistemas y Radiocomunicaciones, Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, Ciudad Universitaria s/n, 28040 Madrid, Spain*
[2] *Departamento de Ingeniería Electrónica, Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, Ciudad Universitaria s/n, 28040 Madrid, Spain*

Correspondence should be addressed to Ángel Fernández-Herrero, angelfh@die.upm.es

Received 18 May 2007; Accepted 26 October 2007

Recommended by Jean-Baptiste Begueret

Future 4th Generation (4G) wireless multiuser communication systems will have to provide advanced multimedia services to an increasing number of users, making good use of the scarce spectrum resources. Thus, 4G system design should pursue both higher-transmission bit rates and higher spectral efficiencies. To achieve this goal, multiple antenna systems are called to play a crucial role. In this contribution we address the implementation in FPGAs of a multiple-input multiple-output (MIMO) decoder embedded in a prototype of a 4G mobile receiver. This MIMO decoder is part of a multicarrier code-division multiple-access (MC-CDMA) radio system, equipped with multiple antennas at both ends of the link, that is able to handle up to 32 users and provides raw transmission bit-rates up to 125 Mbps. The task of the MIMO decoder is to appropriately combine the signals simultaneously received on all antennas to construct an improved signal, free of interference, from which to estimate the transmitted symbols. A comprehensive explanation of the complete design process is provided, including architectural decisions, floating-point to fixed-point translation, and description of the validation procedure. We also report implementation results using FPGA devices of the Xilinx Virtex-4 family.

1. INTRODUCTION

The aim of the 4MORE Project (4G MC-CDMA Multiple Antenna System-on-Chip for Radio Enhancements) is to complement worldwide research efforts on MIMO systems, MC-CDMA, and other advanced signal processing techniques that will provide the high data rates and spectral efficiencies expected from 4G wireless multiuser communication systems. In order to investigate the real performance and feasibility of implementation of these technologies, a complete hardware demonstrator of a broadband mobile terminal (MT) has been designed and is being constructed within the 4MORE project [1]. The demonstrator will focus on an MT with two antennas, but a base station (BS) emulator with four antennas will also be built, since it is required for validation of the MT.

Multi-carrier CDMA, based on the serial combination of direct sequence CDMA and OFDM, has been considered for the physical layer in the downlink because it derives benefits from both technologies: OFDM, with appropriate carrier spacing and guard interval, provides robustness against multipath, avoiding intersymbol interference; whereas the use of CDMA with orthogonal spreading codes provides frequency diversity and multiple-user flexibility [2].

The use of multiple antennas is another enabling technology for 4G systems, which helps to exploit spatial diversity, to increase capacity and to mitigate the effects of fading. In our system the space-time block code for two transmit antennas designed by Alamouti [3] is employed. This option has been favoured over other MIMO technologies, such as beam-forming or layered space-time coding (BLAST) because it provides the maximum attainable diversity order for the number of antennas employed using a simple decoding algorithm.

To achieve good bit error rate (BER) performance, state-of-the-art channel coding techniques, including duo-binary

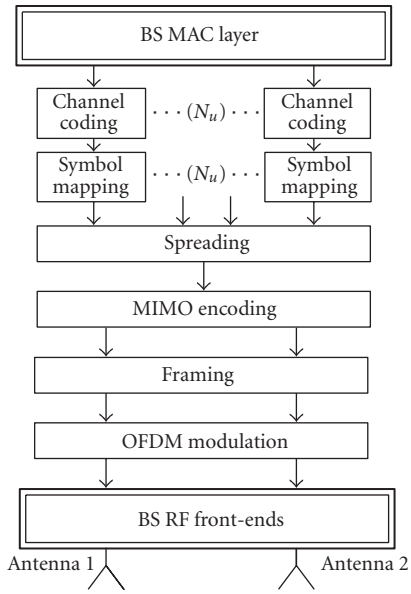

Figure 1: Simplified diagram of the BS transmitter.

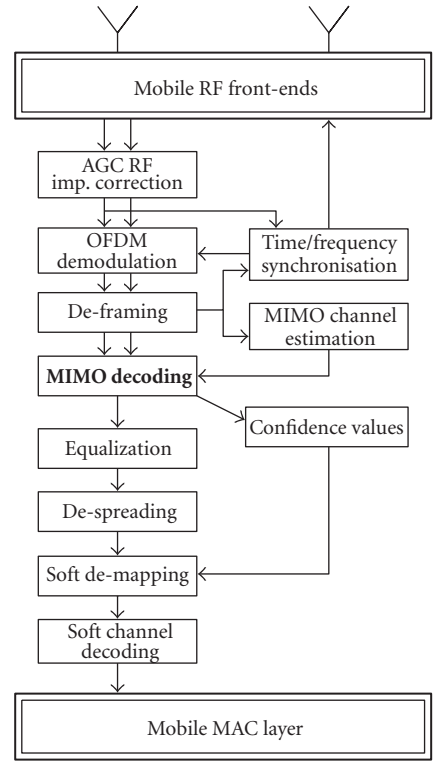

Figure 2: Simplified diagram of the MT receiver.

turbo codes [4] for the uplink, and convolutional and low density parity check codes [5] for the downlink, are employed in the 4MORE demonstrator.

The joint use of all these sophisticated technologies greatly increases the complexity of the transceiver. To deal with the constraints of VLSI design, the demonstrator includes ASICs as well as FPGAs. From the onset of the project it was clear that the demonstrator would make use of some well-established algorithms that could be implemented on ASICs, but the flexibility provided by FPGAs was required to accommodate to the more innovative algorithms to be investigated, bearing in mind that design and implementation tasks would partially overlap in time.

The rest of the paper describes the design and implementation in FPGAs of the hardware module that performs MIMO decoding in the MT, and is organized as follows. In Section 2 a brief overview of the complete downlink system is given, where focus is on the receiver. The basis of the Alamouti MIMO decoding scheme is reviewed in Section 3. Sections 4 and 5, respectively, describe the architecture of the MIMO decoder and detail its fixed-point translation. We discuss implementation details and results in Section 6, before we finally draw our conclusions.

2. OVERVIEW OF THE DOWNLINK SYSTEM

2.1. Transmitting base station

A simplified diagram of the transmitting BS is shown in Figure 1. Data bits to be transmitted to each active user are independently channel encoded and mapped onto symbols of the appropriate constellation (QPSK, 16-QAM or 64-QAM).

Each modulated symbol is multiplied by the spreading code of the corresponding user, and the spread symbols of the N_u active users are added together to be simultaneously transmitted over the same set of $S_f = 32$ subcarriers, which constitutes an MC-CDMA symbol. In our system, the spreading factor in frequency is $S_f = 32$, and the number of users must be in the range of $1 \leq N_u \leq S_f$.

An OFDM symbol consists of $N_s = 21$ contiguous MC-CDMA symbols, so that information is simultaneously transmitted over $N_d = N_s \times S_f = 672$ subcarriers.

Data is prepared for multiantenna transmission by the MIMO encoding module. According to the Alamouti scheme [3], a pair of OFDM symbols $\{\mathbf{x}(n), \mathbf{x}(n+1)\}$, also known as a space-time block, is transmitted employing two antennas over two consecutive symbol periods. During the first symbol period, $\mathbf{x}(n)$ is transmitted from the first antenna, and simultaneously $\mathbf{x}(n+1)$ is transmitted from the second one. During the next symbol interval, the first antenna outputs $-\mathbf{x}^*(n+1)$, while the second one transmits $\mathbf{x}^*(n)$, with $(\cdot)^*$ standing for complex conjugate and n for the symbol epoch. Small bold letters denote vectors with N_d elements, corresponding to the number of data subcarriers in an OFDM symbol.

Before OFDM modulation, the framing module interleaves pilot symbols in the data stream, in order to aid channel estimation at the receiver. One IFFT operation per transmit antenna is required for OFDM modulation, to convert data to the time domain. The IFFT size is 1024, and the sampling rate is 61.44 MHz.

Each stream of complex OFDM symbols is finally IQ-modulated, power amplified by independent RF front-ends, and radiated in the 5-GHz band.

2.2. Receiving mobile terminal

A simplified diagram of the MT receiver is depicted in Figure 2. Analog signals received by the two antennas of the MT are downconverted to baseband by twin zero-IF RF front-ends, and then sampled at 61.44 MHz. After automatic gain control (AGC) and correction of RF impairments caused by the zero-IF architecture of the front-ends, time and frequency synchronization must be performed in order to minimize misalignments with the transmitting BS.

One FFT operation per antenna branch is required to recover the symbols in the frequency domain (OFDM demodulation).

Next, pilots are split from information symbols by the deframing module. By interpolation of pilot symbols in time and frequency, the MIMO channel estimator provides the MIMO decoder with channel state information (CSI), which is combined with two contiguously received OFDM symbols to build the improved signal from which to estimate the modulated symbols.

However, the output stream of the MIMO decoder further requires module equalization [6] and despreading (separation of users by correlation with their spreading codes) before detection of the desired user can take place. The output of the soft demapper is finally sent to the channel decoder to make decisions about the transmitted information bits.

3. MIMO DECODING PRINCIPLE

The fact that during each symbol period both antennas simultaneously transmit different information implies that a linear combination of symbols, affected by the channel frequency response of the different paths, will be received at each antenna of the MT. Due to the intelligent way in which spatial diversity is introduced, a simple linear processing of the signals received by the two antennas during a space-time block eliminates the co-antenna interference (CAI) artificially created by MIMO transmission.

For each space-time block, the MIMO decoder must perform the following linear combination:

$$\begin{aligned}\widetilde{x}(n,l) &= \sum_{j=1}^{2}\left[h_{1,j}^{*}(n,l)\,y_j(n,l) + h_{2,j}(n+1,l)\,y_j^{*}(n+1,l)\right],\\ \widetilde{x}(n+1,l) &= \sum_{j=1}^{2}\left[h_{2,j}^{*}(n,l)\,y_j(n,l) - h_{1,j}(n+1,l)\,y_j^{*}(n+1,l)\right],\end{aligned} \tag{1}$$

where $h_{i,j}(n,l)$ is the estimated frequency response of the channel between transmit antenna i and receive antenna j at the lth subcarrier ($1 \le l \le N_d$) during the nth OFDM symbol period, y_j is the signal obtained after OFDM demodulation at antenna branch j, and $\widetilde{x}$ is the combined output signal. Assuming ideal channel estimation, and a constant channel response during one space-time block, it can be shown that this combining scheme provides full diversity order and cancels CAI [3], leading to this simple model for the combined signal:

$$\widetilde{x}(n,l) = \mathcal{H}(n,l)\,x(n,l) + \mathcal{N}(n,l), \tag{2}$$

where $x(n,l)$ is the lth element of vector $\mathbf{x}(n)$, and $\mathcal{N}(n,l)$ is a Gaussian noise term. Equation (2) is valid for all n, but the equivalent channel $\mathcal{H}(n,l)$ has slightly different expressions for even and odd n:

$$\begin{aligned}\mathcal{H}(n,l) &= \sum_{j=1}^{2}\left[|h_{1,j}(n,l)|^2 + |h_{2,j}(n+1,l)|^2\right],\\ \mathcal{H}(n+1,l) &= \sum_{j=1}^{2}\left[|h_{1,j}(n+1,l)|^2 + |h_{2,j}(n,l)|^2\right].\end{aligned} \tag{3}$$

According to (2), information $x(n,l)$ could be now recovered from $\widetilde{x}(n,l)$ by zero-forcing equalization (dividing by the real factor $\mathcal{H}(n,l)$) or by MMSE equalization [6].

4. ARCHITECTURE OF THE MIMO DECODER

The MIMO decoder must implement (1) to obtain the MIMO-combined signal $\widetilde{x}$, and (3) to obtain the equivalent channel $\mathcal{H}$, required by the equalizer.

The memory of the Alamouti scheme is one OFDM symbol. Throughout the paper we have used the pair (n,l) to refer to the OFDM symbol and subcarrier indices. After OFDM demodulation, information received on all subcarriers is converted from parallel to serial, so we recover a single (complex) stream per antenna branch, that is, the (n,l) pair of indices is equivalent to a single-time index $(n-1)N_d+l$. Hence, a straightforward implementation of the decoder would require the storage of a whole OFDM symbol for every input and output signal (real and imaginary parts of the received signal on each antenna, those of the estimates for the 2×2 MIMO channel, those of the combined output signal, and the equivalent channel), making a total of $15 \times N_d$ samples. However, if all complex signals in (1) are split in their real and imaginary parts (superscripts $(\cdot)^r$ and $(\cdot)^i$), after some algebra and intelligent grouping of terms, we arrive to expressions that suggest a much more efficient implementation. For example, for the real part of $\widetilde{x}$ we get:

$$\begin{aligned}\widetilde{x}^r(n,l) &= s_2(n,l) + s_1(n+1,l),\\ \widetilde{x}^r(n+1,l) &= s_1(n,l) - s_2(n+1,l),\end{aligned} \tag{4}$$

where we have defined:

$$\begin{aligned}s_i(n,l) &= \sum_{j=1}^{2} s_{i,j}(n,l),\\ s_{1,j}(n,l) &= h_{2,j}^{r}(n,l)\,y_j^{r}(n,l) + h_{2,j}^{i}(n,l)\,y_j^{i}(n,l),\\ s_{2,j}(n,l) &= h_{1,j}^{r}(n,l)\,y_j^{r}(n,l) + h_{1,j}^{i}(n,l)\,y_j^{i}(n,l).\end{aligned} \tag{5}$$

Equation (5) is valid for all n, and corresponds to memoryless arithmetic operators that will run continuously, while all memory effects have been included in (4). The architecture inferred from these equations is shown in Figure 3, where all signals are real. All arithmetic resources are disposed so as to make a 100% utilization of them, including the programmable adder/substractor A3 at the output of the module. The whole structure works as a pipeline running at

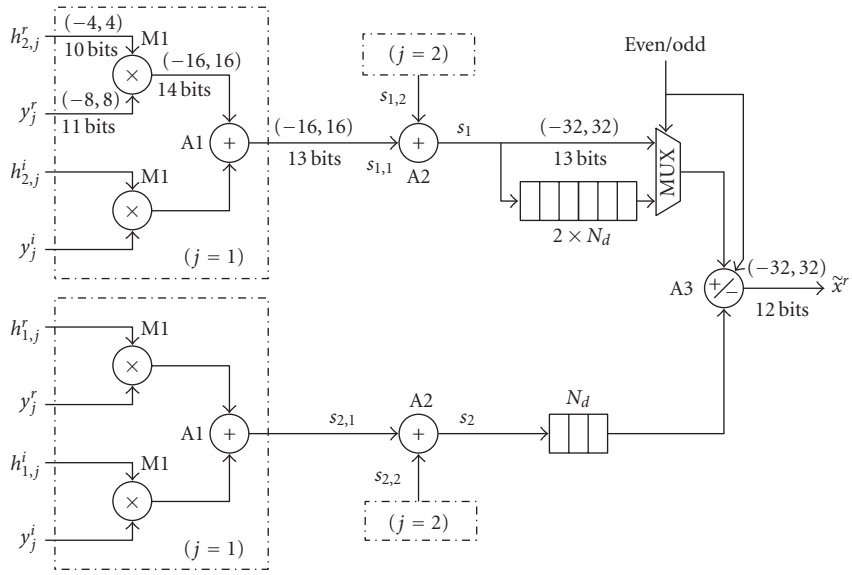

Figure 3: Architecture for the MIMO decoder (real part $\tilde{x}^r$). Signal ranges and wordlengths displayed are for the fixed-point implementation option Q2 (see Section 5 and Table 2).

Table 1: Parameters of the modes implemented in the demonstrator.

Modulation	Channel coding rate (R_{cc})	Number of users (N_u)
QPSK ($b = 2$)	1/2	1 to 32
16-QAM ($b = 4$)	2/3	1 to 32
64-QAM ($b = 6$)	3/4	1 to 32

clock speed and, although not explicitly shown in Figure 3, adders and multipliers have registered outputs. The even/odd signal indicates whether the current OFDM symbol is even or odd, and is used to control the multiplexer and to change between addition and substraction in the programmable adder A3. Slotted rectangles are used to represent multibit shift-registers, which do not need to be resettable. We observe that memory requirements for evaluation of $\tilde{x}^r$ are $3 \times N_d$ samples, and that the total latency is equal to $N_d + 4$ clock periods.

We do not show the full details of the architectures used to evaluate $\tilde{x}^i$ and $\mathcal{H}$ because they are very similar to that shown in Figure 3, just placing the appropriate signals at the inputs. For evaluation of $\tilde{x}^i$, the major difference is that first-level adders A1 are replaced by subtractors, while for $\mathcal{H}$, the programmable adder/substractor A3 is replaced by a simple unsigned adder, the rest of the adders being unsigned as well. Thus, the MIMO decoder comprises three submodules very much like the one shown in Figure 3, and we therefore reduce the total memory requirements of the complete module to $9 \times N_d$ samples.

This architecture can be easily and efficiently adapted to a different number of antennas at the receiver. To this end, the arithmetic blocks surrounded by dotted lines in Figure 3 should be replicated, both in the upper and lower branches of the architecture, and the two-input adders A2 should be replaced by cascaded adders to handle more than two inputs. While deploying more than two antennas at the MT is unpractical, this architecture could also be used for MIMO decoding in the uplink, where a BS with four or more receive antennas is feasible.

5. FIXED-POINT TRANSLATION

The fixed-point translation of the architectural design described in the previous section was accomplished following three steps.

(a) Determine the range of each input, output, and intermediate signal involved in the MIMO decoder.
(b) Obtain the number of bits (precision) required for each signal.
(c) Test the robustness of the design by performing BER simulations.

Following this process, similar to that described in [7], we seek to obtain a low-cost, performance-effective implementation for the hardware module.

5.1. Estimation of signal ranges

This task was accomplished with the help of the SystemC-based floating-point software simulator that has been developed within the 4MORE Project, which accurately models the behaviour of all the modules in the demonstrator and includes a realistic MIMO channel model. It is possible with this simulator to obtain traces of the signals at any point in the communication link.

We show in Table 1 the most important parameters of the different working modes that have been implemented in the demonstrator. While the range for the channel estimates $h_{i,j}$ is independent of the mode, the range for the

Table 2: Fixed-point quantization rules.

Signal		Q1		Q2		Q3	
		Range	Bits	Range	Bits	Range	Bits
Inputs	y_j^r, y_j^i	$(-8.0, 8.0)$	12	$(-8.0, 8.0)$	11	$(-8.0, 8.0)$	10
	$h_{i,j}^r, h_{i,j}^i$	$(-8.0, 8.0)$	12	$(-4.0, 4.0)$	10	$(-4.0, 4.0)$	9
Output of... (combined signal path)	M1	$(-16.0, 16.0)$	14	$(-16.0, 16.0)$	14	$(-16.0, 16.0)$	13
	A1	$(-16.0, 16.0)$	15	$(-16.0, 16.0)$	13	$(-16.0, 16.0)$	12
	A2	$(-32.0, 32.0)$	16	$(-32.0, 32.0)$	13	$(-32.0, 32.0)$	12
Output of... (equivalent channel path)	M1	$(0.0, 16.0)$	14	$(0.0, 16.0)$	12	$(0.0, 16.0)$	11
	A1	$(0.0, 16.0)$	15	$(0.0, 16.0)$	11	$(0.0, 16.0)$	10
	A2	$(0.0, 32.0)$	16	$(0.0, 16.0)$	10	$(0.0, 16.0)$	9
Global outputs	$\tilde{x}^r, \tilde{x}^i$	$(-32.0, 32.0)$	14	$(-32.0, 32.0)$	12	$(-32.0, 32.0)$	11
	$\mathcal{H}$	$(0.0, 32.0)$	14	$(0.0, 32.0)$	10	$(0.0, 32.0)$	9

received signals y_j depends on the modulation type and on the number of users. The widest signal range will be attained when 64-QAM modulation is combined with the maximum number of users. By careful examination of histograms of large records of data obtained running the SystemC simulator with these parameters, we found that the range for the real and imaginary parts of the received signals y_j lied with high probability in the interval $(-4.0, 4.0)$ while for the channel estimates $h_{i,j}$ the range was found to be $(-3.0, 3.0)$. The histograms observed for all signals were almost Gaussian in shape. To be on the safe side we decided to include an extra margin, and considered the ranges for y_j and $h_{i,j}$ to be $(-4.0, 4.0)$ for the design. By doing so we try to take outliers into account, and some of the variability of the channel which might have not been captured in our data records. Bear in mind that the channel variability greatly affects the amplitude of the received signals, and that the MIMO channel model is quite complex, its behaviour being influenced by many physical and statistical parameters.

Once the ranges for input signals were known, those of intermediate and output signals could be obtained taking into account the theoretical margins that result when operating with inputs whose range is already known. Nevertheless, this would lead to an overdimensioned module, due to the existence of hidden correlations between the inputs. After all, each of the received signals y_j is a linear combination of the data x multiplied by the channel paths $h_{i,j}$. Therefore, we resorted to histogram observation to determine those ranges. The results are all shown in parentheses in Figure 3 and also in Table 2.

5.2. Word-length optimization

To ease this task we developed a simple software model of the MIMO decoder, identical to the module included in the floating-point SystemC simulator of the whole chain, but much faster and practical, since all unnecessary burdens were removed. This new software model can be quickly modified to include fixed-point conversion effects in any of its parts.

As performance metric we used the signal-to-quantization noise ratio (SQNR) at the outputs of the MIMO decoder, measured by comparison of the outputs of the floating-point version of the module with that obtained after including quantization effects in some signal, or in all of them. By doing so we seek to keep the power of quantization noise much lower than that of additive white Gaussian (AWGN) noise, hence guaranteeing a negligible effect of the first one on performance.

Fixed-point conversion effects were introduced one signal at a time, and simulations were run in parallel with both versions of the MIMO decoder. The number of bits assigned to the fractional part of the signal under study was then adjusted and simulations repeated until a target value for the SQNR was reached.

Next, fixed-point effects were removed from that point, and we proceeded to optimize the word-length of another signal in the module.

Nevertheless, for those signals that share the same statistics, quantization effects were simultaneously analysed. For instance, optimization of the number of bits at the output of all multipliers M1 in Figure 3 was done simultaneously, running simulations with all multipliers substituted by their fixed-point counterparts, all of them with the same number of bits. For the same reason, all first-level adders A1 were simultaneously optimized, as well as all second-level adders A2.

Following this procedure we obtained, three sets of quantization rules, to which we will refer as Q1, Q2, and Q3 from now on, each of them established aiming at a different goal. The final parameters for these quantization rules are shown in Table 2 (and for Q2, they are also embedded in Figure 3). The number of bits displayed for all signals includes integer plus fractional part.

Quantization rule Q1 was conceived overdimensioned to ensure that it would work with every mode of the demonstrator. Quantization rule Q2, slightly less resource-consuming than Q1, was tried for 64-QAM, but final results were not good enough. As it will be shown in next section, the 64-QAM constellation is very sensitive to even small noise increments. Finally, Q3 was designed to work only with QPSK modulation, using the minimum number of resources.

Signal traces to run the tests were obtained from the complete SystemC simulator, always setting $N_u = 1$, since in this case the range of the inputs is the smallest and therefore the required precision is the highest. We used 64-QAM signals for Q1 and Q2, and QPSK for Q3. The target value for SQNR was set to be greater than 55 dB when designing Q1, 45 dB with Q2, and 35 dB with Q3.

As will be shown later (see Figure 4), the demonstrator may require values of the signal-to-noise ratio (SNR) per information bit (E_b/N_0) at the input of the receiver as high as 13 dB to obtain a low BER, the limiting case being that of 64-QAM modulation with 32 users. This is tantamount to a value of the per-carrier signal-to-noise ratio (SNR_c) of approximately 20 dB, since E_b/N_0 and SNR_c are related by [6] by the following equation:

$$\mathrm{SNR}_c(\mathrm{dB}) = E_b/N_0 + 10\log_{10}\left(b \cdot R_{cc} \cdot \frac{N_u}{S_f}\right). \quad (6)$$

Measurements with signal traces obtained running the simulator in this limiting case resulted in the higher value $\mathrm{SNR}_c = 22.1$ dB at the ouput of the MIMO decoder, the increase being due to the combining process.

At the end of the word-length optimization process we ran a final simulation to compare the floating-point version with the optimized fixed-point one, including all quantization effects simultaneously. The measured SQNR value was about 48 dB for Q1, safely bigger than 20 dB, and output SNR_c fell only from 22.11 dB to 22.10 dB when including quantization effects. For Q2, the final SQNR was about 40 dB, while SNR_c fell to 22.05 dB. For Q3, losses in SNR_c were negligible.

5.3. Validation in terms of BER performance

As final step, the SystemC simulator was used to validate in terms of BER performance the final decisions concerning signal ranges and word-length optimization. For this purpose a complete fixed-point software model of the MIMO decoder was developed, which is bit-accurate with the VHDL source code to be implemented in the FPGAs. By substitution of the original floating-point MIMO decoding module by its fixed-point counterpart in the complete SystemC simulation chain, and including appropriate floating/fixed-point interfaces to the neighbouring modules, we verified the degradation in BER performance introduced by the fixed-point MIMO decoder. This can be checked in Figures 4–6, where the BER versus E_b/N_0 performance has been evaluated for different modes of the demonstrator.

As it can be seen in Figure 4, quantization Q1 is suitable for every mode, with a maximum loss of about 0.14 dB at BER $= 10^{-4}$ for 64-QAM (negligible with 16-QAM and QPSK). From Figure 5, quantization Q2 can be considered for 16-QAM with a loss up to 0.14 dB, but not for 64-QAM, where losses reach 1 dB. Finally, according to Figure 6, Q3 is suitable for QPSK with negligible losses, while it worsens by 0.3 dB for 16-QAM, a loss double than that obtained using Q2.

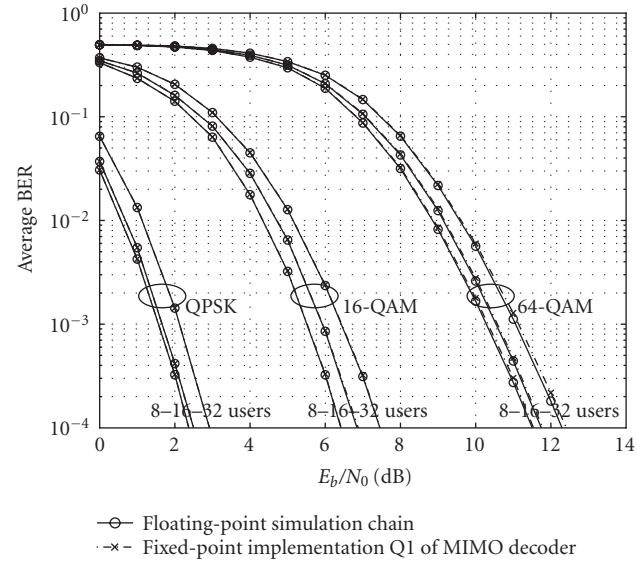

Figure 4: BER degradation comparing the floating-point version of the MIMO decoder (solid lines with marker "o") and its fixed-point counterpart implementation Q1 (dashed lines with marker "x").

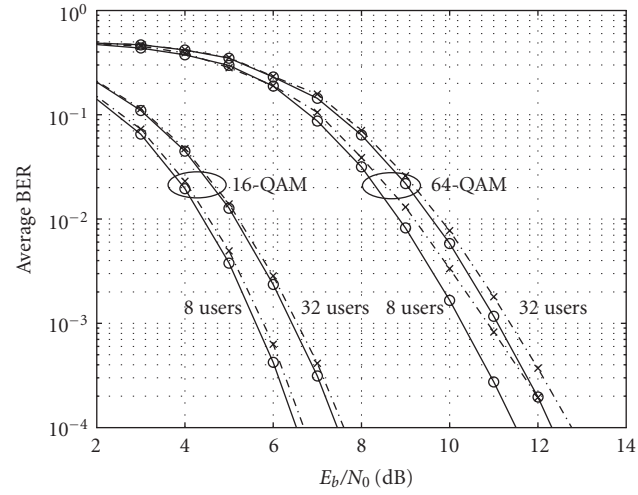

Figure 5: BER degradation comparing the floating-point version of the MIMO decoder (solid lines with marker "o") and its fixed-point counterpart implementation Q2 (dashed lines with marker "x").

6. IMPLEMENTATION AND RESULTS

The following tools were used during the design: Xilinx ISE 7.1 and the XST engine were used for VHDL synthesis and place-and-route, while Mentor ModelSim SE 6.0d was used to run functional and post place-and-route simulations. The target FPGAs considered for the implementation are Xilinx Virtex-4, since they are most suitable for implementation of wireless systems [8]. Specifically, model XC4VLX100-12 units are included in the demonstrator.

Table 3: Synthesis results for the MIMO decoding module.

	DSP48	Flip-flops	Slices	LUTs	Logic	Route-through	Shift registers	DSP slices	Min. clock cycle (ns)
Q1	Auto	599	3245	6337	704	5	5628	24	7.965
Q1	Yes	651	3405	6321	105	0	6216	49	9.554
Q2	Yes	419	2435	4544	92	0	4452	49	9.985
Q2	Auto	423	2495	4946	489	5	4452	24	6.577
Q2	No	759	3963	7628	3163	13	4452	0	5.524
Q3	Auto	390	2308	4515	436	5	4074	24	6.956

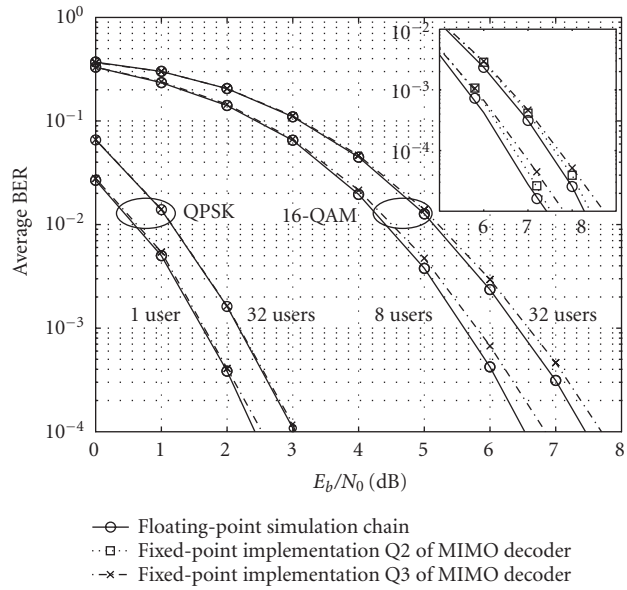

Figure 6: BER degradation comparing the floating-point version of the MIMO decoder (solid lines with marker "o") and its fixed-point counterpart implementation Q3 (dashed lines with marker "x"). In the zoomed area, results for the fixed-point implementation Q2 are also shown for comparison (dotted lines with marker "⊡").

Table 3 shows the synthesis results for the MIMO decoder using the three different fixed-point implementations discussed in Section 5 and summarized in Table 2.

The second column, labelled "DSP48," refers to an option of the synthesis tool which can take three different values: "no" means that no DSP blocks are allowed; "yes" tells the synthesis tool to use as many of them as required; and "auto" triggers a free use of the DSP blocks, depending on the best trade-off found by the tool.

The value of that option has a very significant effect on the column "DSP slices" since the architecture of MIMO decoder needs 24 multipliers. When using "auto" for the "DSP48" option, these are made available as DSP blocks by the synthesis tool, whereas when the "yes" option is selected, the tool also maps the 21 adders (including 15 adders, 4 substractors, and 2 programmable adders/substractors) and other elements in DSP blocks, finally getting 49 DSP slices used, and consequently reducing the number of LUTs in the column "Logic" (from 3163 to 92 for Q2, while shift registers keep the same size).

The column "LUTs" can be obtained by adding the following three: "Logic," LUTs used for logic functions and arithmetic; "Route-through" for routing paths between slices; and "Shift registers." The data in this last column are very relevant for our design, since shift registers are large components in the architecture and consume the greatest part of the resources (except in the case of value "no" for "DSP48"). They affect the slice count, since the width of the registers is reduced when changing to more severe quantizations (from Q1 to Q3).

Considering the total number of slices, there is a reduction of 23% from quantization Q1 to Q2 ("auto"), while it is only 7.5% from Q2 to Q3.

The column "Flip-flops" includes the registers needed in the control unit and also those used for the pipeline. This excludes the registers that follow the arithmetic units mapped to DSP blocks, since they are directly taken from the blocks, and not from the slices.

The last column is the minimum clock cycle inferred by the synthesis tool with a timing constraint of 100 MHz, which is the clock frequency available in the demonstrator. It can be emphasized that the use of DSP blocks results in a slower design, due to the additional routing needed to reach the (fixed) positions of those components in the FPGA. In this regard, the fastest implementation (and also the largest in area) is the one using quantization rules Q2 selecting "no" for the "DSP48" option.

Quantized outputs of the deframing and channel estimation modules (see Figure 2) obtained from the floating-point SystemC simulator were used as realistic input test patterns to perform the functional validation of the hardware implementation. The outputs of the VHDL simulations driven by these patterns were compared for equality with those obtained by the bit-accurate fixed-point software model of the MIMO decoder, when driven by those same input patterns.

7. CONCLUSIONS

We have presented the design methodology used in the implementation of a MIMO decoder within a 4G radio system. The architecture of the system has been optimized to comply with the throughput requirements while reducing implementation area.

Given the random nature of the inputs, the design of wireless systems demands a simulation-based fixed-point translation approach for word-length optimization. A robust simulation framework, able to deal both with floating-point

and fixed-point descriptions, has proven to be essential in the design.

Several quantization versions have been developed, synthesized with different options, in order to check the tradeoffs between accuracy and use of resources in different conditions.

Our implementation results using Xilinx Virtex-4 devices show that the MIMO decoder requires a limited number of FPGA resources, while achieving high performance.

ACKNOWLEDGMENTS

This work has been supported by European FP6 IST 2002 507039 Project 4MORE and by the Spanish Ministry of Science and Technology under Project TEC2006-13067-C03-03.

REFERENCES

[1] 4MORE IST project website, http://ist-4more.org.
[2] S. Hara and R. Prasad, "Overview of multicarrier CDMA," *IEEE Communications Magazine*, vol. 35, no. 12, pp. 126–133, 1997.
[3] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 8, pp. 1451–1458, 1998.
[4] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1261–1271, 1996.
[5] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, 1999.
[6] A. Fernández-Herrero, A. Jiménez-Pacheco, G. Caffarena, and J. Casajús-Quirós, "Design and implementation of a hardware module for equalisation in a 4G MIMO receiver," in *Proceedings of International Conference on Field Programmable Logic and Applications (FPL '06)*, pp. 1–4, Madrid, Spain, August 2006.
[7] W. Sung and K.-I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Transactions on Signal Processing*, vol. 43, no. 12, pp. 3087–3090, 1995.
[8] "Virtex-4 user guide," March 2006, http://www.xilinx.com/support/documentation/user_guides/ug070.pdf.