FPGAs Enable Next-Generation Multiple-Antenna Wireless Communications

Luis G. Barbero and John S. Thompson

***Abstract*— This paper shows how Alpha Data's field-programmable gate array (FPGA)-based hardware platforms can be used to investigate and implement receive algorithms for multiple input-multiple output (MIMO) wireless communications by means of a rapid prototyping methodology.**

I. Introduction

The use of multiple input-multiple output (MIMO) technology is rapidly becoming the new frontier of wireless communication systems increasing their capacity and spectral efficiency through the use of multiple antennas at both transmitter and receiver [1]. Nowadays, the prototyping of those multiple-antenna systems has become increasingly important to verify the enhancements advanced by analytical results. For that purpose, field-programmable gate arrays (FPGAs), with their high level of parallelism, high densities and embedded multipliers, are a suitable prototyping platform.

II. MIMO System

We consider a spatially-multiplexed MIMO system with M transmit and N receive antennas, denoted as $M \times N$. The main idea behind MIMO is that, at the transmitter, M independent signals are sent simultaneously over the wireless channel. At the receiver, each one of the N antennas receives a copy of the transmitted signals. Under favourable channel conditions, the receiver can use those multiple copies to obtain the original transmitted signals. Fig. 1 shows a simplified schematic of a MIMO system.

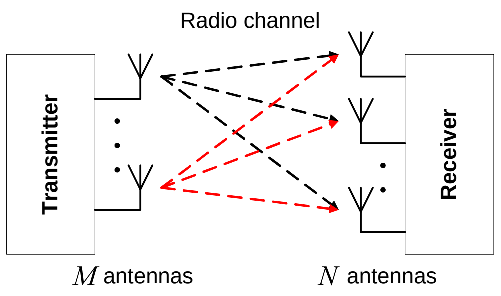

Fig. 1. MIMO schematic with M transmit and N receive antennas.

The optimum receiver for such MIMO systems is the maximum likelihood detector (MLD), although its exponential complexity makes it unrealizable in practical systems when a large number of antennas and higher order constellations are used. The sphere decoder (SD) has been proposed as an alternative, providing the same maximum likelihood (ML) performance with reduced complexity [2]. However, the SD has a variable complexity, depending on the channel conditions and the noise level making it difficult to integrate in an actual system where data needs to be processed at a constant rate. In addition, the algorithm performs a sequential tree search that makes it impossible to obtain a parallel fully-pipelined implementation.

The FPGA platform and the rapid prototyping methodology described here have been used to analyse the SD from an implementation point of view, identifying its drawbacks. Furthermore, a novel fixed-complexity sphere decoder (FSD) has been proposed to overcome the main problems of the SD [3]. The FSD achieves practically the same ML performance with a fixed complexity, while performing a parallel search that can be fully-pipelined in hardware.

III. Rapid Prototyping System

The algorithms have been implemented using a rapid prototyping system that has the simplicity and, at the same time, the flexibility required to move quickly from a computer-based implementation of an algorithm to its real-time implementation. It makes it possible to perform real-time hardware-in-the-loop testing of the algorithms embedded in a computer-simulated environment as shown in Fig. 2. Details can be found in [4].

- Hardware Platform: ADC-PMC peripheral component interconnect (PCI) adapter board that hosts two FPGA boards, an ADM-XRC-II with a Xilinx Virtex-II (XC2V4000) and an ADM-XP with a Xilinx Virtex-II-Pro (XC2VP70).
- Rapid Prototyping Methodology: It is based on The MathWork's MATLAB and Simulink and Xilinx's DSP System Generator tailored to Alpha Data's FPGA boards.

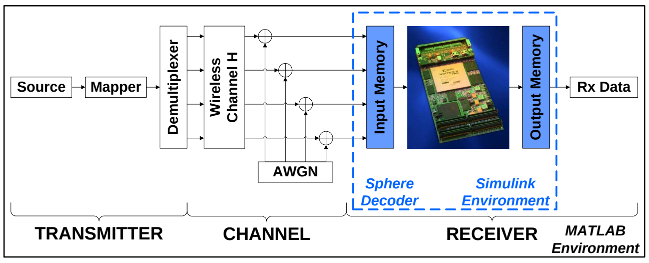

Fig. 2. Hardware-in-the-loop MIMO system diagram.

IV. FPGA Implementation Results

The SD and the FSD have been implemented for a 4×4 system using 16-quadrature amplitude modulation (QAM) modulation [5], [6]. In addition, the FSD has also been implemented for a $4{\times}4$ system using 64-QAM modulation [7].

The resource use of the implementations of both 4 parallel SDs and the FSD on the Xilinx Virtex-II-Pro FPGA board are summarized in Table I for the case of 16-QAM. The input symbols to the algorithms have been quantized using 16 bits per real component with both algorithms achieving effectively the same bit error ratio (BER) performance.

TABLE I
FPGA RESOURCE USE OF 4-SDS AND THE FSD

Xilinx XC2VP70 FPGA	4-SDs	FSD
Number of slices (33,088)	64% (21,467)	38% (12,721)
Number of flip-flops (66,176)	26% (17,691)	23% (15,332)
Number of 4-input LUTs (66,176)	54% (36,249)	24% (16,119)
Number of multipliers (328)	47% (156)	48% (160)
Number of block RAM (328)	55% (183)	25% (82)

It can be seen that the FSD uses significantly less resources than the 4-SDs with the exception of the flip-flops and the multipliers. The flip-flops are used in the design as delay nets to synchronize the different pipeline stages at the end of the detection process.

The number of multipliers used is slightly larger indicating that the computational complexity of the algorithm in terms of hardware resources is similar to that of the SD. However, this does not directly translate into a more complex hardware implementation. Factors like the regular structure of the algorithm and the possibility of pipelining also determine the suitability of the algorithm for a hardware implementation. The number of look-up tables (LUTs) has also been considerably reduced. The use of LUTs can be seen as an indicator of the control logic required for the algorithm. In the case of the FSD, the fixed number of operations and the possibility of pipelining greatly reduces the need for control blocks leaving the LUTs mainly to arithmetic operations.

Finally, the number of memory blocks has been more than halved, where most of them are due to the input and output buffers defined on the FPGA to synchronize the FPGA board and the Simulink interface.

Fig. 3 shows the real-time average throughput of the FSD and different versions of the SD as a function of the signal to noise ratio (SNR) per bit. The throughput in megabits per second (Mbps) is proportional to the clock frequency, f_{clock}, of the design in MHz and the number of clock cycles required to perform the detection per MIMO symbol. It can be seen how the FSD clearly outperforms the different SD alternatives and, more importantly, provides a constant throughput of 400 Mbps independent of the noise level. The maximum throughput achievable by the SD is 128 Mbps.

It should be noted that the SD runs at a lower clock frequency $f_{clock} = 50\,\text{MHz}$ compared to the $f_{clock} = 100\,\text{MHz}$ of the FSD. However, increasing f_{clock} to match that of the FSD would also increase the length of the critical path of the SD, without increasing the overall throughput of the system. On the other hand, given that the FSD has been fully pipelined, hardware optimizations to further increase f_{clock} would directly increase the throughput of the implementation. In particular, internally pipelining the multipliers in the FSD can increase the clock frequency of the design to $f_{clock} = 150\,\text{MHz}$, resulting in a throughput of 600 Mbps.

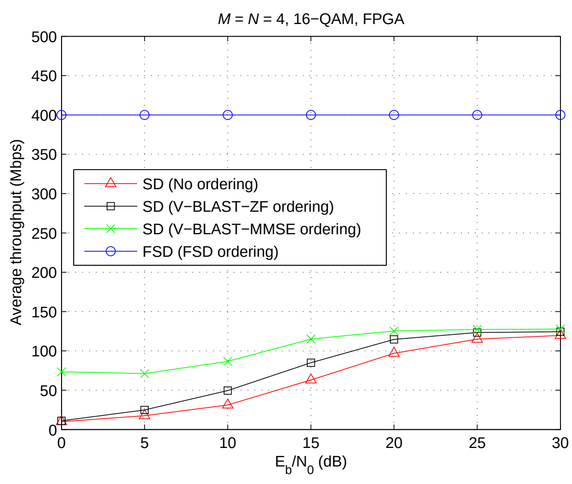

Fig. 3. Average throughput of the FSD and the SD with different orderings of the channel matrix as a function of the SNR per bit.

V. Conclusion

This paper has shown how Alpha Data's FPGA boards have been successfully used in the analysis and prototyping of detection algorithms for MIMO wireless communication systems. The rapid prototyping methodology described here has allowed the study of the SD algorithm from an implementation point of view in order to identify its disadvantages. That analysis has made it possible to propose a novel FSD algorithm that overcomes the practical problems of the SD. Thus, the use of FPGAs combined with a rapid prototyping methodology can significantly enhance the development of future wireless communication systems.

References

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

[2] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1639–1642, July 1999.

[3] L. G. Barbero and J. S. Thompson, "A fixed-complexity MIMO detector based on the complex sphere decoder," in *Proc. IEEE Workshop on Signal Processing Advances for Wireless Communications (SPAWC '06)*, vol. 1, Cannes, France, July 2006.

[4] ——, "Rapid prototyping system for the evaluation of MIMO receive algorithms," in *Proc. IEEE International Conference on Computer as a Tool (EUROCON '05)*, vol. 2, Belgrade, Serbia and Montenegro, Nov. 2005, pp. 1779–1782.

[5] ——, "Rapid prototyping of the sphere decoder for MIMO systems," in *Proc. IEE/EURASIP Conference on DSP Enabled Radio (DSPeR '05)*, vol. 1, Southampton, UK, USA, Sept. 2005, pp. 41–47.

[6] ——, "Rapid prototyping of a fixed-throughput sphere decoder for MIMO systems," in *Proc. IEEE International Conference on Communications (ICC '06)*, Istanbul, Turkey, June 2006.

[7] ——, "FPGA design considerations in the implementation of a fixed-throughput sphere decoder for MIMO systems," in *Proc. IEEE International Conference on Field Programmable Logic and Applications (FPL '06)*, Madrid, Spain, Aug. 2006.