An FPGA Based Rapid Prototyping Platform for MIMO Systems

Patrick Murphy, Feifei Lou, Ashutosh Sabharwal and J. Patrick Frantz
Rice University
Houston, TX 77005
{murphpo,fflou,ashu,jpfrantz}@rice.edu

Abstract

Multiple input multiple output (MIMO) techniques hold the potential of dramatically increasing the data rates and spectral efficiency of wireless communications systems. Even with extensive research on the design of transmission and reception algorithms, little is known as to how much of the predicted gains are actually achievable on real wireless channels. In this paper, we present a MIMO testbed which enables the rapid prototyping of MIMO transceivers for wideband channels. Such prototypes provide experimental quantification of achievable gains from MIMO algorithms. The testbed design allows real-time operation of baseband processing and RF up/down-conversion. The choice of testbed components is made to allow maximum flexibility for research purposes, including monitoring and control of all subsystems. In addition to discussing the testbed's design, we present the implementation of two wireless systems. The first is a spread-spectrum system based on IEEE 802.11b. The second is an implementation of Alamouti's transmit diversity scheme.

1 Introduction

Significant theoretical effort has addressed many aspects of MIMO techniques for narrowband systems. Though the literature boasts a deep understanding of MIMO algorithms for a small class of channels (flat-fading Rayleigh channels), little is known about the performance of MIMO systems in actual wireless channels. Even the limited experience with MIMO implementations has pointed to disagreement between theoretical predictions and practical systems [1]. The disagreement between theory and practice[1] is a good example of a common unwritten wisdom among wireless system builders, "the channel is Rayleigh only 25% of the time." As result, algorithms optimized for a specific channel class are bound to suffer in the presence of channel mismatch.

Channel mismatch is not the only effect which affects the system performance; other examples include nonlinearities in power amplifiers and finite bit precision of analog converters. We do not aim not to undermine the importance of simplified theoretical models and analyses, without which no progress may be possible. Instead, our goal is to suggest that the definitive word on the superiority of a scheme should always include a careful understanding of its performance on an actual system and interaction with other communication subsystems. Thus, one of the goals of this testbed is to refine the models used in theory to reflect the practical issues *without* making them intractable.

In order to facilitate an environment of cohesive theoretical and experimental development, we have assembled a MIMO testbed to allow rapid prototyping and evaluation over realistic wireless channels. The testbed allows flexibility and significant computational capabilities by using FPGAs as baseband engines. To allow testing on real wireless channels, we use the unlicensed 2.4GHz bands for actual transmissions. It is well known that real wireless channel conditions are not repeatable. To allow repeatability and testing in licensed bands, we have also incorporated multiple channel emulators.

As an example application of the proposed testbed, we present two implementations utilizing Xilinx's System Generator for DSP [2]. First, we discuss the implementation of synchronization of packets in direct sequence spread spectrum system in 802.11b wireless LAN standard [3]. The example clearly shows the tradeoff between hardware utilization (number of gates required for implementation and hence area of the chip) and clock rates. Second, we discuss the implementation of Alamouti's transmit diversity scheme [4] for two transmit and a single receive antenna. Our focus in this design is the development of systems to combat the effects of the transmitter and receiver running on independent, asynchronous. Again, the tradeoff between number of gates and clock frequency surfaces.

The remainder of this paper is organized as follows. Section 2 describes the testbed design and the

hardware chosen to construct it. Section 3 discusses two implementation examples using the testbed's resources. Finally, Section 4 offers some concluding remarks.

2 Testbed Design

The testbed consists of hardware which performs three major functions: baseband processing, conversion to and from radio frequencies and emulation of realistic wireless channels. The components we chose to fill each role are described below.

2.1 Baseband Hardware

We found early on that of all the options for baseband processing, FPGAs provided the greatest flexibility in algorithm design and visibility of resource utilization. FPGAs allow control over parallelism in resource utilization, and also the measurement of resource utilization and power consumption. These features are important as the computational requirements of a particular algorithm play a large role in gauging its suitability for real-world use. We acknowledge that FPGAs may not be suitable for deployment in large scale wireless communication systems. However, they are ideal for rapid prototyping and research use such as this testbed [5].

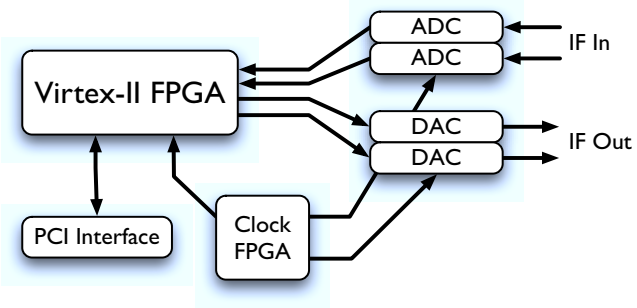

Figure 1: XtremeDSP Kit FPGA board

We decided on FPGA-based development systems designed by Nallatech of Glasgow, Scotland as our baseband processing platform. As shown in Figure 1, each board houses a multi-million gate Xilinx Virtex-II FPGA and two channels each of analog input and output. The analog channels are fast enough to interface directly to the testbed's RF hardware, allowing each FPGA board to act as a two antenna transceiver. Each board also contains a smaller FPGA dedicated to providing clocks to the larger FPGA and the external analog interfaces. The presence of a separate clock FPGA provides the desired flexibility in connecting to hardware from different designers with various interface requirements. Finally, the Nallatech board also has PCI and USB interfaces which provide a high throughput connection with host PC.

2.2 RF Hardware

We chose RF hardware from National Instruments which fits the needs of our testbed. The NI RF up/downconverters support signal bandwidth up to 20MHz, accommodating a large variety of wideband wireless communication schemes. They also operate with an intermediate frequency of just 15MHz or 25MHz which is easily within the range of baseband FPGA's sampling capabilities. Finally, they provide RF tunable from 9KHz to 2.7GHz which covers two ISM plus cellular bands to allow the testbed not tied to any particular standard. They also allow the control over the synchronization of the frequency translation stages of the radios. Such control allows variation in the carrier frequency offsets between antennas and is very useful in multi-antenna communication systems.

In order to maximize flexibility in choosing multiple antenna configurations, the testbed also uses a number of RF multiplexers between the RF transceivers and the channel emulators, described in Section 2.3. The multiplexers allow most configurations to be implemented programmatically without any changes to the equipment's physical connections.

2.3 Wireless Channel Emulation

In order to provide realistic wireless channels and repeatable experimental results in a laboratory setting, we chose RF channel emulators from Spirent Communications. Each device is capable of emulating two wireless channels with signal bandwidth up to 20MHz at RF from 900MHz to 2.6GHz, each channel with six or twelve independently configured paths. Figure 2 illustrates the capabilities of each emulated wireless channel. Each path can apply one of many fading models, including Rayleigh, Rician and Nakagami, plus programmed delay and attenuation. This flexibility allows the emulation of a large number of realistic line-of-sight and multi-path environments. The

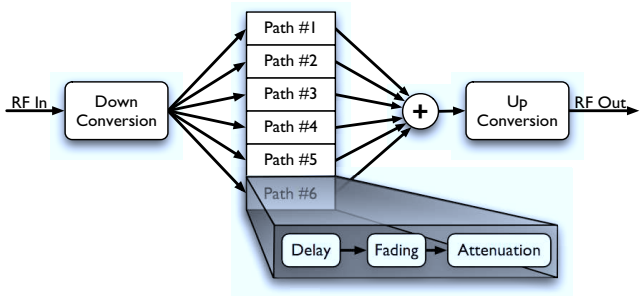

Figure 2: Emulated channel capabilities

emulator can also introduce arbitrary frequency shifts to each path, providing a reliable means to simulate the Doppler effects of various mobile scenarios.

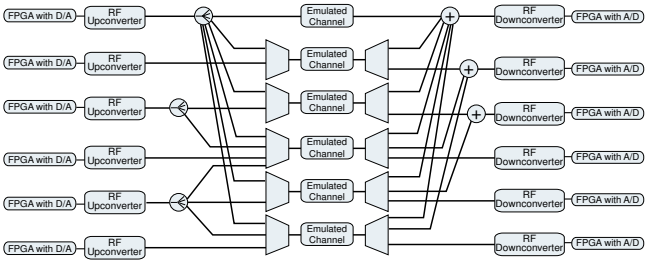

Figure 3: Testbed configurations

Our testbed currently houses three such emulators, providing up to six independent emulated channels. This fairly large number of channels provides significant flexibility in designing test configurations. In terms of transmit-to-receive antennas, the testbed can implement 6-to-1, 3-to-2, 2-to-3, 1-to-6 half-duplex systems or any smaller configurations. Figure 3 illustrates the full connectivity of multiple-antenna half-duplex systems using all the hardware described in this section. Figure 4 shows the testbed hardware in its current form.

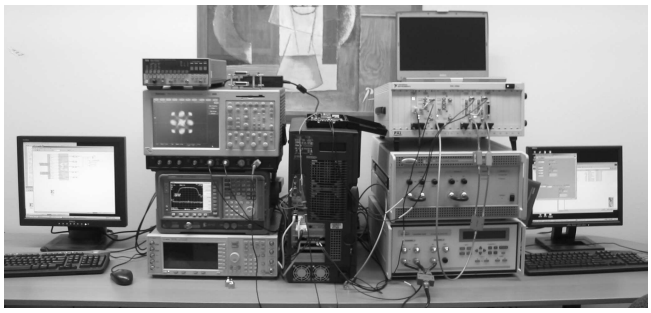

Figure 4: Testbed hardware

3 Example Applications

Two applications are presented in order to demonstrate testbed's functionality. First, we discuss the implementation of packet synchronization in direct sequence spread spectrum in IEEE 802.11b system. This example clearly shows one of the challenges of implementing a spread spectrum receiver, namely that computationally intensive task of chip level synchronization. Second, we discuss the implementation of Alamouti's transmit diversity scheme for two transmit and single receive antenna. Here we focus on the carrier recovery and timing synchronization subsystems. Many of the challenges we faced and the solutions we devised would be applicable in implementations of more complex MIMO systems.

These two examples are seemingly simple communication systems. However, their implementation highlights the challenges of prototyping, especially the challenges of optimizing models used in theory in terms of some practical aspects such as power consumption or hardware utilization.

3.1 DSSS System (802.11b) PHY Implementation

IEEE 802.11b provides high data rate specifications for transmission at up to 11Mbps with 20MHz bandwidth in the 2.4GHz ISM band. In the PHY layer it uses various modulation schemes for different data rates. We present an implementation of 2Mbps transceiver which uses the standard 11-bit Barker code as the spreading sequence with DQPSK modulation.

The implementation of the transmitter is straightforward. A block diagram of our transmitter is shown in Figure 5. The upconversion to the intermediate frequency of 25MHz is done in the FPGA. This IF was chosen for compatibility with the National Instruments NI5610 RF upconverter, the RF hardware we chose for this testbed. The transmitter design was targeted to a two million gate Xilinx Virtex-II FPGA (XC2V2000). The design consumes about 8% of the logic resources in the FPGA and runs with a system clock of 66MHz.

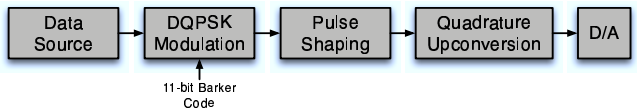

Figure 5: DSSS transmitter block diagram

The implementation of the receiver is more challenging. A block diagram of our receiver is shown in Figure 6. The carrier recovery block is based on a Costas loop, a scheme which functions well with our system's DQPSK modulation. The system has a chip rate of 11MHz and processes two samples per chip, resulting in a sample rate of at least 22MHz. If we were to use one large polyphase matched filter [6] matched to the preamble of each packet, the filter would operate at nearly 200MHz and would require a huge number of taps. Such a system is not practical in hardware. As a result, there must be some tradeoff between hardware utilization and operation rate. Also the receiver has to use multiple levels of synchronization to reduce filter taps.

The chip synchronization block utilizes a fully parallel eight branch interpolating polyphase matched filter running at the 22MHz sample rate. This block is used to track frequency and phase offsets between the transmitter and receiver's clocks. Compared to a fully serial filter structure, this system consumes 3 times the number of logic slices but runs at 1/8 the sample rate.

The symbol synchronization block correlates the received signal with the Barker code to identify each symbol boundary. A fully parallel correlator is applied. Compared to a fully serial filter structure, it consumes 2 times the number of logic slices but runs at 1/11 the sample rate. In this realization, we combine the symbol synchronization with despreading, demodulation and detection for a very hardware efficient design.

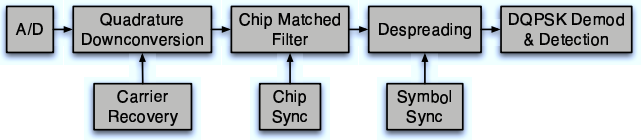

Figure 6: DSSS receiver block diagram

The receiver consumes 43% logic of a XC2V2000 and runs at 66MHz. At this clock frequency, it achieves a throughput of 2Mbps. The receiver was designed to interface to an NI5600 RF downconverter with an IF of 15MHz.

3.2 Alamouti Implementation

Alamouti's transmit diversity scheme [4] is the first example of a space-time block code which requires only linear processing in the encoding and decoding stages to achieve its diversity gain [7]. We present here an implementation of the simplest case of Alamouti's scheme which utilizes two transmit antennas and one receive antenna [8]. Many of the challenges we faced and the solutions we devised would be applicable in implementation of more complex multi-antenna systems.

The transmitter design is fairly simple, consisting of a QPSK modulator, Alamouti encoder, pulse shaping filters and digital upconverters for translation to IF. In fact, the hardware realization differs very little from the implementation of two QPSK transmitters.

The receiver is much more complicated. It includes subsystems for carrier recovery, symbol synchronization, channel estimation, decoding and demodulation. The carrier recovery block assumes the transmit antennas share a reference clock and have no frequency or phase offset. It then corrects for any frequency offset between the transmit and receive radios. The symbol synchronization block utilizes an eight branch interpolating polyphase matched filter described in [6] and [9]. It chooses the optimal time to sample the incoming symbol stream based only on a local estimate to the transmitter's symbol clock and the received data itself. The channel estimation block correlates the received data against a known training sequence to estimate each channel's phase offset and attenuation.

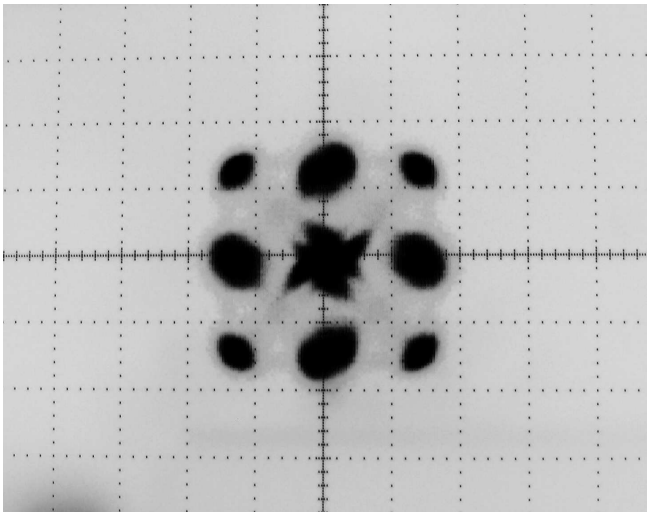

Figure 7: Received Alamouti QPSK constellation

The transmitter and receiver has been synthesized and implemented separately in three million gate FPGAs. The transmitter uses 20% of the logic resources in the FPGA running at 100MHz and the receiver uses 25% at 60MHz. The system has been verified as an end-to-end link with realized throughput of 7.5Mbps at 2.4GHz RF using the hardware described in Section 2.

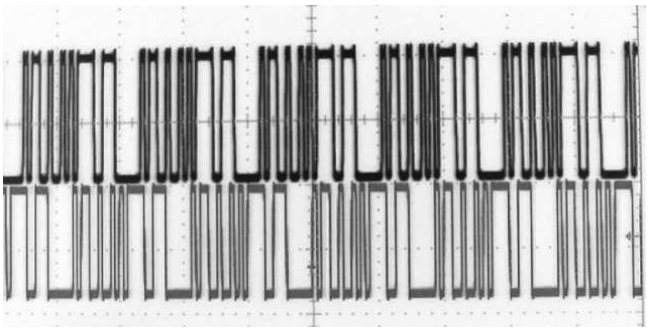

Figure 8: Transmitted vs. received bitstreams

With no common reference clock between the trans-

mitter and receiver, there was a definite offset in the symbol timing and generated carriers. This offset required the design of synchronization and carrier recovery loops, both of which are required to lock and track for extended periods. Figure 7 is the constellation at the receiver captured by oscilloscope. Our system uses two transmit antennas and QPSK modulation. As a result, the receiver must accept nine valid sums of two QPSK symbols. This constellation is very clean, demonstrating the performance of the carrier offset and symbol synchronization systems. Figure 8 shows the real-time transmitted and received bit sequences. Aside from a small delay representing the processing latency at both ends of the link, the bit sequences are identical.

4 Conclusion

This paper provides a description of an FPGA based rapid prototyping platform used to design, implement, and validate MIMO algorithms as a starting point for developing practical wireless MIMO systems. While the simulation of such algorithms is useful, it is often beneficial to verify these algorithms in real hardware. This process of implementation provides feedback to the algorithm designer regarding the practicality of the scheme, hopefully improving the suitability of future algorithms for use in real wireless systems.

Acknowledgements

The equipment for this testbed was purchased in part with NSF grants CISE-ANI-0224458, CISE-ANI-0325971 and CISE-MRI-0321266. Authors Murphy and Lou are each partly supported by Texas Instruments Fellowships. Special thanks to Xilinx, Nallatech, Spirent Communications and National Instruments for their generous support and equipment donations.

References

[1] P. Gupta, W. Zhu, and M. Fitz, "Field test results for space-time coding," in *Asilomar Conference on Signals, Systems, and Computers*, Asilomar, CA, November 2003.

[2] *http://www.xilinx.com/system_generator/.*

[3] *http://standards.ieee.org/getieee802/802.11.html.*

[4] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Select Areas in Communications*, vol. 16, pp. 1451–1458, October 1998.

[5] P. Murphy, F. Lou, and J. Patrick Frantz, "A hardware testbed for the implementation and evaluation of MIMO algorithms," in *Proceedings of the 2003 Conference on Mobile and Wireless Communications Networks*, October 2003.

[6] F. Harris and M. Rice, "Multirate digital filters for symbol timing synchronization in software defined radios," *IEEE Journal on Select Areas in Communications*, vol. 19, pp. 2346–2357, December 2001.

[7] V. Tarokh, H. Jafarkhani, and A.R. Calderbank, "Space-time block codes from orthogonal designs," *IEEE Transactions on Information Theory*, vol. 45, pp. 1456–1467, July 1999.

[8] P. Murphy, P. Frantz, and C. Dick, "An FPGA implementation of Alamouti's transmit diversity technique," in *University of Texas WNCG Wireless Networking Symposium*, Austin, TX, October 2003.

[9] C. Dick, F. Harris, and M. Rice, "Synchronization in software radios- carrier and timing recovery using FPGAs," in *Proceedings of 2000 IEEE Symposium on Field-Programmable Custom Computing Machines*, April 2000, pp. 195–204.