DESIGN OF WARP: A WIRELESS OPEN-ACCESS RESEARCH PLATFORM

Patrick Murphy, Ashu Sabharwal and Behnaam Aazhang

Rice University
Department of Electrical and Computer Engineering
6100 Main St., Houston, TX 77005, USA
{murphpo, ashu, aaz}@rice.edu

ABSTRACT

This paper presents the design of WARP, a custom platform for research in advanced wireless algorithms and applications. The platform consists of both custom hardware and FPGA implementations of key communications blocks. The hardware consists of FPGA-based processing boards coupled to wideband radios and other I/O interfaces; the algorithm implementations already include a flexible OFDM physical layer. Both the hardware specifications and algorithm implementations will be freely available to academic researchers to enable the development of a widely disseminated, highly capable platform for wireless research.

1. INTRODUCTION

We present the design of a highly capable, scalable and extensible platform for advanced wireless research. The platform goals can be described at a high level through four key requirements. First, the platform is *capable* of implementing advanced wireless algorithms. Wireless algorithms are generally quite complex, requiring substantial processing resources to implement in real time. The computational resources must be vast but flexible enough to allow the exploration of algorithms which have not yet been developed. Second, the platform must be *scalable*, allowing additional processing resources to be allocated when the computational power of a single processor proves insufficient. Third, it must be *extensible*, providing peripheral and interface expansion options to accommodate future applications. Fourth, in order to preserve its utility in the long term, the platform must support the evolution of some components without compromising the functionality of others. For example, the platform hardware must be updated as more advanced processors become available while assuring compatibility for designs implemented on previous generations.

Finally, the design elements which address these requirements must be integrated into a complete, open-access platform, providing the tools and resources required for research at all levels of wireless systems. This integration must enable researchers to focus on improving one aspect of a wireless system without having to implement, or even understand, everything required to enable real-world communications.

2. PLATFORM ARCHITECTURE

Our platform is designed from the ground up to meet the demands of high performance wireless systems research. The platform architecture, depicted in Figure 1, consists of four key components:

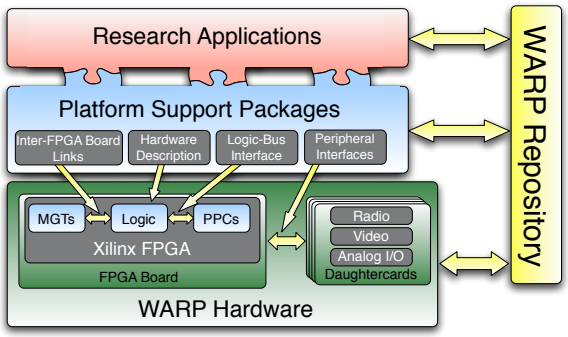

Figure 1: *Components of the platform design*

1. **Custom Hardware**: Custom designed hardware tailored to the needs of high-performance wireless communications. The hardware provides resources *capable* of high speed DSP-intensive algorithm implementations, *scalable* interconnects for allocating more processing power as needed, and *extensible* peripheral options for radios, specialized processors and other expansion interfaces.
2. **Platform Support Packages**: Design tools and low-level interface modules designed to enable seamless use of the hardware by researchers at all layers of wireless network design.
3. **Open-Access Repository**: Central archive for all source code, models and hardware design files. The full contents will be available under a BSD-like open-source license.
4. **Research Applications**: Implementations of novel algorithms, enabled by the custom hardware and platform support packages. Full systems incorporating novel algorithms can be rapidly built using standard building block modules provided in the repository.

The platform's modular and layered architecture ensures that the hardware components can benefit from advances in processor technology, following the exponential increases in performance famously described by Moore's Law. The approach also enables algorithm implementations to be used on future hardware revisions without being re-designed, maximizing the impact of researchers' implementation efforts.

2.1 Hardware Design

The first component of the platform design is custom hardware which provides the processing, interface and peripheral resources required to meet the goals described above. The initial revision of the hardware design, shown in Figure 2, was completed in the last year and is being actively for algorithmic development efforts.

2.1.1 FPGA Board

The platform hardware must provide substantial processing resources to meet the computational needs of wireless systems operating at 100s of Mb/sec. Wireless algorithms require a large number of DSP-centric computations, yet DSPs and other similarly structured processors do not provide nearly enough processing power. We chose FPGAs as the primary processor. Large FPGAs provide tremendous processing resources composed largely of parallel, programmable logic blocks which can be interconnected to form complex functional units. FPGAs are also extremely well suited for DSP-intensive operations, especially in applications where algorithms can be parallelized. For example, the front-end processing for many wireless applications requires high-throughput operations like filters and correlators replicated for each wireless interface. Implementations of these operations can exploit parallel structures in hardware to improve performance. Further, each instance of a functional unit operates in parallel with all others. Such multi-level parallelism is a key way FPGAs provide performance far beyond the capabilities of even the most powerful DSP.

The main board in the platform is the FPGA board. At the heart of this design is a Xilinx Virtex-II Pro FPGA. This family of FPGAs is very well suited for the kinds of DSP-intensive operations required by wireless algorithms. For example, the FPGA provides dedicated multipliers, hardware blocks which implement efficient and fast multiplication, a critical operation in signal processing designs. The FPGAs also provide flexible and fast interconnect options for interfacing peripherals and creating multi-processor systems, two primary requirements of the platform design. The Virtex-II Pro also includes embedded PowerPC processor cores, providing an ideal resource for implementing higher layer algorithms better suited for general purpose processors than programmable logic.

While the FPGA itself provides substantial processing power, its connections to other devices and boards enable the variety of applications targeted by the platform. The FPGA board provides a 10/100 Ethernet interface for connections to standard wired networks. This connection enables real-time communication between existing wired network nodes and custom wireless network nodes implemented on WARP.

In addition, the FPGA board has four daughtercard slots, each wired to a large number of dedicated FPGA I/O pins. These slots house peripheral cards (three example daughtercard designs are described below). The slots are flexible enough to support a wide variety of future peripheral designs, including multimedia interfaces and specialized auxiliary processors. The four slots are functionally identical, allowing users to mount peripheral cards that best suit their application. The slot interface is documented in the repository, allowing users to design custom daughtercards.

A major challenge in designing an FPGA-based platform for computationally intensive applications is connecting multiple FPGAs together to accommodate algorithms which are too complex to fit on a single chip. A common approach is to design hardware with multiple FPGAs on a single board [1]. Such designs have the benefit of proximity of computational resources, guaranteeing low-latency communication between processors. Two major drawbacks to this approach, however, are the added hardware design complexity and potential for wasted resources in applications which require fewer FPGAs than those mounted on a single board. We expect such applications to be common, especially in the early stages of algorithmic implementation. Thus, the FPGA board is built around a single large FPGA.

This presents the challenge of connecting multiple FPGAs together when the need arises. We address this scalability requirement by using the multi-gigabit transceivers built into the Xilinx FPGAs. Each MGT provides a full duplex 3+ Gb/sec connection between two FPGAs; multiple MGTs can be used in parallel to provide even more throughput between two boards. Eight MGTs are be routed to off-board connectors on each FPGA board, providing substantial ($\sim$ 24 Gb/s) inter-FPGA communications capabilities.

2.1.2 Peripheral Daughtercards

The FPGA board provides four daughtercard slots. Each slot is connected to a dedicated bank of I/O pins on the FPGA, providing a flexible, high-throughput interface. The daughtercard slots also provide a combined 30 watts of power isolated from the FPGA supplies. These daughtercard slots provide the opportunity to design a wide variety of custom daughtercards, ranging from peripheral interfaces like radios and Ethernet

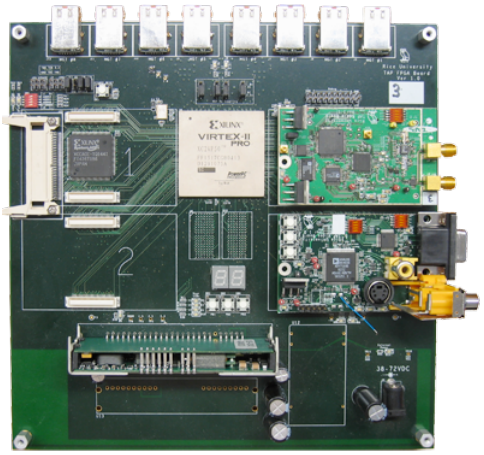

Figure 2: *First revision of the platform's custom hardware, including the Virtex-II Pro FPGA board, radio daughtercard (top right) and video daughtercard (bottom right).*

to auxiliary processing modules, built around DSPs or specialized processors.

We have already designed three daughtercards in the first revision of the hardware. The first is a radio interface, shown in Figure 2, built around a single chip direct-conversion radio transceiver from Maxim Integrated Products [2]. The Maxim radio supports both the full 2.4GHz and 5GHz ISM/UNII bands, including support for channels in these bands which are not available in the USA but are open in other countries. The radio is designed for MIMO applications, guaranteeing the phase coherency of carriers generated by chips which share a reference clock. The first revision of the radio board has been completed and is fully functional; Figure 3 shows the board's output at 2.4GHz.

The second daughtercard provides analog video capture and playback capabilities (see Figure 2). The FPGA interface modules for this card have already been completed and provide a simple means for algorithm designers to acquire and display image and video data. The third daughtercard provides six channels of fast analog I/O (2 A/D and 4 D/A). These interfaces enable the implementation of wireless algorithms in real-time at baseband frequencies, decoupling the processes of algorithmic and RF interface debugging. The specification for the daughtercard interface is part of the open-access repository. We envision that a whole range of daughtercards will be designed over time, both at Rice and by other users of the platform.

2.2 Platform Support Packages

The platform requires a number of tools and supporting packages in order to enable the functionality of all its interfaces and peripherals. These support packages expose the platform's full capabilities to researchers

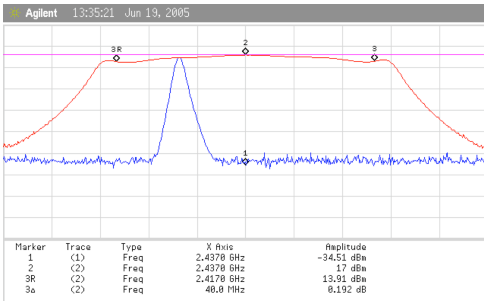

Figure 3: *Spectrum analyzer view of an RF signal transmitted by our custom radio board.*

working at various layers. Well designed interfaces help mask the intricacies of the processing at layers above and below the one of interest, allowing users to access the full platform without having to manage its full complexity.

Low-Level Support: Platform support packages must be developed to enable functionality at four levels. The first level is support for designs implemented solely in logic. Work at this level requires only Xilinx's ISE design tools to synthesize hand-coded HDL and map the designs to hardware. The hardware support for such designs consists largely of timing and I/O specifications specific to our custom hardware.

Logic-PowerPC Support: The second level of support enables the interface between designs running in FPGA logic and code running in the embedded PowerPC (PPC) cores. The PPC processors rely on their standard buses for all interaction with the surrounding programmable logic. Thus, designs running in logic must implement an interface to one of these buses in order to exchange data and control information with the PPC core. This bus interface is not trivial, and is especially challenging for users working with high level FPGA design tools. We have developed a tool to address this need for designs implemented in Xilinx's System Generator. Our tool translates the top-level ports of a System Generator model into a bank of registers. Logic implementing address decoding and bus multiplexing is then added, resulting in a seamless interface between the PPC bus and System Generator design. We have successfully used this tool to interface an OFDM transceiver designed in System Generator with a high level control program running in the PowerPC. The OFDM transceiver operates as a standard memory-mapped peripheral, appearing as a simple network interface to the control program.

Peripherals Support: The third layer of support packages is responsible for the interfaces between the FPGA and the various off-chip peripherals included in the custom hardware. Included at this layer is the library of interface modules which bridge designs running in the FPGA with peripheral cards mounted in the

FPGA board's daughtercard slots. The interface module for the radio daughtercard is already complete and exists in two forms. The first is a System Generator library, providing gateways in the simulation environment which map automatically to the radio card's analog data converters and control ports. The second is a set of Verilog modules which provide the same functionality for lower level designs implemented in raw HDL. Both libraries are complete and are being used in our development efforts. Similar interface modules will be completed for future daughtercards.

Board-to-Board Support: The final layer of support packages enables board-to-board communications for applications which require multiple FPGAs working in concert. Inter-board links utilize the multi-gigabit transceivers discussed above. Each transceiver is a low level device integrated into the FPGA. The transceivers are very flexible, with support for a wide variety of data rates, buffering and coding options. Unfortunately, this sophistication is more a problem than a feature from the perspective of a user working on high level designs. Our support packages mask this complexity by wrapping each MGT and connecting it as a memory-mapped peripheral on PowerPC core's bus. Our custom interface automatically handles the buffering required for clock compensation at both the transmitter and receiver. We have also designed a similar interface for use by low level designs which do not utilize the PowerPC cores.

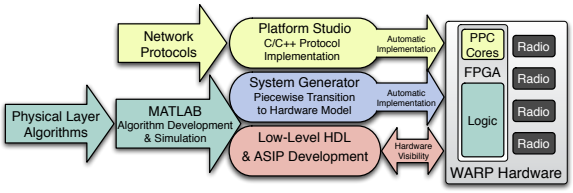

Figure 4: *WARP Design Flows*

2.3 WARP Design Flows

Figure 4 shows how researchers design the various layers of a custom wireless network while using the platform interface tools to integrate different layer implementations. To highlight the power of WARP's design flow, we give an example how a new network like a spectrum sniffing IEEE 802.11 network can be built with relative ease. Using custom spectrum sniffing and management protocols, combined with a standard IEEE 802.11 MAC and OFDM physical layer from the WARP Repository, a full network is immediately realizable. The resulting network can actively seek channels with the least interference, easily outperforming a standard IEEE 802.11 network with static channel allocation. The sniffing protocol will be implemented in C/C++ on one of the platform's PowerPC processors, which views the physical layer as a network interface

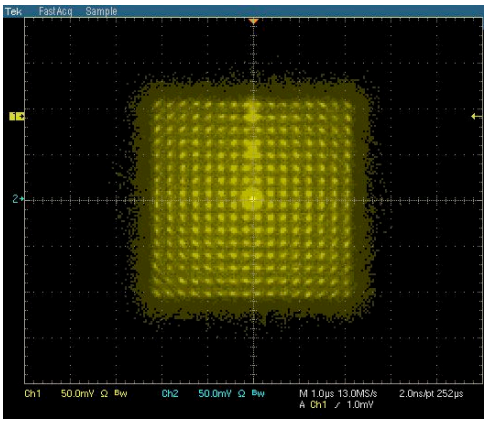

Figure 5: *256-QAM constellation viewed at the output of our OFDM transceiver, running in real time in the FPGA.*

card using the platform support packages described in Section 2.2.

Not only can a network be glued layer by layer, each layer can be closely integrated with others to enable cross-layer optimizations. Such cross-layer designs are only possible due to the completely open-access nature of the WARP repository, which allows designers to adapt each layer's design to fit their objectives.

2.4 Open-Access Repository

The third primary component of the platform infrastructure is the open-access repository. The repository serves as the archive for all aspects of the platform, including designs, libraries, documentation and application examples. It implements version control for all code, models and documentation and provides basic project management features to enable collaboration by many users on one project. The repository contents are available under a BSD-derived open-source license. All users will also be encouraged, but not required, to contribute their own designs, examples and documentation back to the repository.

The contents of the repository can be classified into five major categories:

1. **Hardware Designs:** schematics, layout and documentation for all custom printed circuit boards.
2. **Platform Support Packages:** tools and libraries providing interface support, as described in Section 2.2.
3. **Application Building Blocks:** optimized and documented designs implementing various standard functional units. These include both generic blocks, like high speed correlators and multi-rate filters, as well as application specific blocks like OFDM packet detection and symbol synchronization. Platform users can utilize these known-good blocks to speed the design of their own systems, concentrat-

ing their effort on implementations of novel algorithms.

4. **Research Applications:** fully functional, ready-for-hardware designs. Current designs include implementations of full network layers, like an OFDM PHY (see Figure 5) and a simple MAC protocol. Future contributions will include MIMO-OFDM, LDPC, network coding and advanced multi-hop network protocols.
5. **Tutorials, Application Notes & Student Projects:** documentation and design examples which will serve as an introduction to the platform for new users. This section will also contain student projects completed in courses that use the platform which are suitable for use as design examples.

3. RELATED WORK

A number of other wireless testbeds and commercial products exist which provide some capabilities similar to WARP. However, none of these addresses all of the requirements we identified when designing our platform. For example, the GNU Radio project provides a flexible wireless development platform which includes an open-source framework of wireless algorithms implemented in software [3]. But with the bulk of the processing offloaded to a host PC, current GNU Radio systems cannot achieve the kind of high-throughput, wide-band communications provided by WARP. Commercial platforms, such as those from Sundance [4] and Lyrtech [5], provide some RF capabilities similar to WARP. However, these systems do not provide an open framework for algorithmic implementation at both the physical and MAC layers, a key provision of our platform. To the best of our knowledge, WARP is unique in its combination of high-performance programmable radio interfaces with open-access frameworks which enable wireless architecture, physical layer and network protocol research.

4. CONCLUSIONS

This paper has describes the design of an open-access platform for advanced wireless research. Initial versions of all the platform components are completed, including fully functional FPGA and radio boards and implementations of an OFDM transceiver and medium access protocol. Development continues on all levels of the platform, extending the capabilities of both the hardware architecture and wireless networking frameworks. Ongoing updates and public access to the repository are available online [6].

REFERENCES

[1] C. Chang, K. Kuusilinna, B. Richards, A. Chen, N. Chan, R.W. Brodersen, and B. Nikolic, "Rapid design and analysis of communication systems using the BEE hardware emulation environment," in *IEEE International Workshop on Rapid Systems Prototyping*, June 2003.

[2] *http://maxim-ic.com/MAX2829*.

[3] *http://www.gnu.org/software/gnuradio/*.

[4] *http://www.sundance.com/*.

[5] *http://www.lyrtech.com/*.

[6] *http://warp.rice.edu/*.

ACKNOWLEDGEMENTS

Special thanks to both Maxim and Xilinx for their support throughout this project. This project is funded by NSF grants ANI-0325971, EIA-0224458 and EIA-0321266.