WARPLab: A Flexible Framework for Rapid Physical Layer Design

Narendra Anand, Ehsan Aryafar, and Edward W. Knightly
Rice University, Houston, TX, USA
{nanand, ehsan, knightly}@rice.edu *

ABSTRACT

In this paper, we present WARPLab, a framework for the rapid prototyping and implementation of physical layer algorithms for wireless LANs. WARPLab is based on WARP, an FPGA-based, wireless experimental platform. We first give an overview of WARPLab and then present an example of a physical layer algorithm that we implemented on the framework for our work. We then discuss the features, limitations, and future modifications as they pertain to our research.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design- Wireless Communication

General Terms

Experimentation, Design

Keywords

WARPLab, Multi-User Beamforming, Wireless LAN

1. WARPLAB OVERVIEW

Rice University's **W**ireless Open-**A**ccess **R**esearch **P**latform (or WARP) [1] is a framework developed for the design and implementation of physical and network layer protocols. The hardware, as seen in Fig. 1, is comprised of a Xilinx Virtex-II FPGA with a PowerPC processor connected to up to four interchangeable daughter cards including radio boards, ADC/DAC boards, and user I/O boards.

New physical layer designs can be developed in the FPGA while new network layer protocols can be developed in the PowerPC. When these new designs are coupled with the radio daughter cards, researchers are able to easily evaluate the performance of new wireless protocols in a real-time, over-the-air (OTA) environment.

WARPLab is an extension of this core idea that allows for a researcher to more rapidly prototype and evaluate new physical layer protocols by interfacing WARP nodes directly with MATLAB. WARPLab allows for physical layer processing to be implemented in MATLAB instead of in the FPGA itself, a far more complex route. In addition to performing all baseband processing in MATLAB, the researcher can also coordinate up to 16 WARP nodes from the same host PC in order to easily perform complex, multi-node, OTA experiments. The resulting experimental system's overall topology would be as shown in Fig. 2.

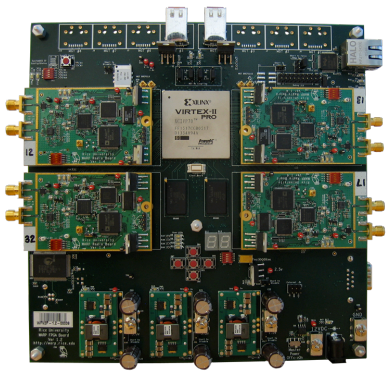

Figure 1: WARP Node.

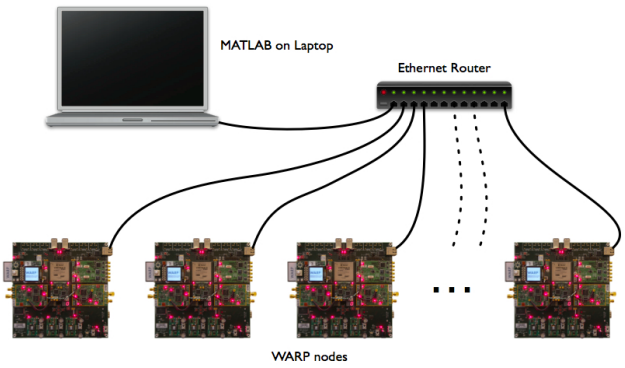

Figure 2: WARPLab experimental topology.

1.1 WARPLab Design Flow

In a WARPLab system, each node's FPGA consists of four large buffers (one per antenna) that each hold 2^{14} samples. The PowerPC on the FPGA is used to facilitate the communication between MATLAB and the FPGA buffers by transferring data between the host PC and the buffers along with handling control signals that instruct the node to transmit or receive (along with setting other

*We would like to thank Rice University's Center for Multimedia Communication (CMC) Lab for their invaluable support with the WARP platform.

S3'10, September 20, 2010, Chicago, Illinois, USA.
ACM 978-1-4503-0144-2/10/09.

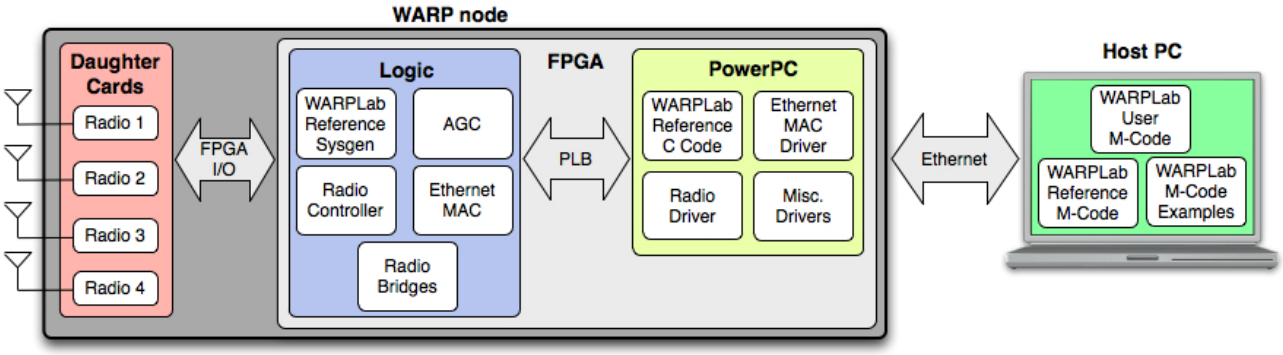

Figure 3: WARPLab design flow.

configuration parameters). The complete design flow is as shown in Fig. 3.

One WARPLab experiment cycle is as follows:

1. Samples to transmit are generated in MATLAB.
2. Baseband processing of signal to be transmitted is performed in MATLAB (i.e. the PHY layer protocol to be tested is applied to the transmitting signal).
3. MATLAB downloads the processed signal to the buffers on the transmitting WARP node over ethernet.
4. MATLAB sends the "Enable Transmit" and "Enable Receive" control packets to the appropriate WARP nodes to prime the nodes for an OTA transmission.
5. MATLAB then transmits a "Sync" packet to all of the transmitting and receiving nodes simultaneously.
6. Once the nodes receive this "Sync" packet, the transmitting node imeadiately flushes its buffers through its radios while the receiving nodes imeadiately loads its buffers with data streamed in through its radios.
7. After the OTA transmission is complete, the receiving WARP nodes upload the received signals along with RSSI readings (if requested) to the host PC where the resulting data is post-processed in MATLAB to take the desired measurements.

2. PHYSICAL LAYER DESIGN EXAMPLE: MULTI-USER BEAMFORMING

Multi-User Beamforming (MUBF) is a method of serving multiple users simultaneously from a single transmitter. Recent advances in hardware technology have finally allowed for the design and prototyping of such systems. In [2], we have thoroughly and experimentally evaluated the performance of MUBF. The complexity of implementing an appropriate system for the OTA evaluation of this protocol necessitated a research platform that allowed for rapid prototyping of physical layer (PHY) algorithms while still allowing for accurate OTA experimentation. For this reason, we elected to use WARP. The WARP hardware coupled with the WARPLab design flow allowed for us to implement and experimentally evaluate MUBF in an accurate and timely manner. In addition to being able implement complex PHY layer algorithms in MATLAB, WARPLab's ability to coordinate experiments between multiple nodes from a single host PC made this framework an even more desirable choice. This section describes how WARPLab framework was used to perform the experimental evaluation described in [2].

2.1 Implementation

In [2], we used WARPLab to experimentally evaluate MUBF and constructed an experimental cycle based on the list above. Our experimental topology is as shown in Fig. 4 where the connection through the host PC serves as the feedback link.

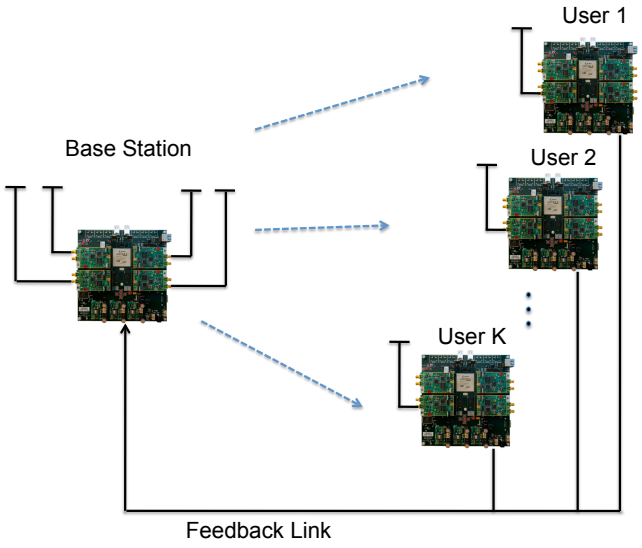

Figure 4: Experimental setup.

Because MUBF requires channel information (through the feedback link) to properly steer simultaneous beams toward the intended receivers, two WARPLab transmission cycles are required to perform one experimental iteration. First, training data is transmitted to all of the receiving nodes who then report the received signals back to the host PC. Then, the channel matrix is computed, the beam weights are calculated and then used to phase twist the data streams towards the intended receivers. (A more in depth discussion of this process can be found in [2].) This measurement process raises an obvious concern–the key WARPLab tradeoff.

2.2 Limitations: The WARPLab Tradeoff

Because of the added latency of communication between the host PC and each of the WARP nodes, the experimental setup is not real time. Specifically, the time between the calculation of the channel estimate and the use of this estimate for beamforming (~60 ms) could result in an outdated channel estimate due to rapidly varying channels. This is why, in [2], OTA experiments had to be performed in stable environments and any channel variation-type experiments had to be performed with a channel emulator. WARPLab allowed us to implement the baseband processing in MATLAB more efficiently than if we were to do so in an FPGA with the downside of a non-real time system.

2.3 Future Work

For future work, we would like to leverage the complete node management and control of WARPLab while introducing modifications to the FPGA design to create a more real time system. We are currently working on shifting more of the baseband processing from MATLAB to the FPGA so that this complete measurement cycle can run in real time (i.e. the latency between channel estimate and beamformed data transmission more closely resembles that of a real wireless card).

3. REFERENCES

[1] Rice University WARP project. Available at: `http://warp.rice.edu`.

[2] E. Aryafar, N. Anand, T. Salonidis, and E. Knightly. Design and experimental evaluation of multi-user beamforming in wireless LANs. In *Proceedings of ACM MobiCom*, September 2010.