Linux enriched design in second generation wireless open-access research platform

Jaakko Niemelä, Markku Jokinen, Tuomo Hänninen
Centre for Wireless Communications
Department of Communication Engineering
University of Oulu
P.O.Box 4500, 90014-Oulu Finland
Email: {jaakko.niemela, markku.jokinen, tuomo.hanninen}@ee.oulu.fi

***Abstract*—Wireless research infrastructure is a central tool in performing applied research. It enables the verification and demonstration of theoretical results in practice. Wireless research infrastructure at the Centre for Wireless Communications (CWC) has been built using second generation second generation wireless open access research platform (WARPv2). The WARPv2s are well suited for wireless research on both physical (PHY) and medium access control (MAC) layers but they lack the capability for handling network layer operations without an external PC. An on-board operating system is needed for a stand-alone solution.**

This paper describes Linux system porting to the WARPv2 board. The Linux system was combined with the radio communication system to achieve a flexible MAC interface for building custom protocols. The field-programmable gate array (FPGA) hardware design was created for this combination and needed software of the system was defined and implemented. The developed system consists of the Linux side and the MAC side which together implement all the layers of the open systems interconnection (OSI) model. Both sides use a dedicated processor core of the FPGA. Interprocessor communication between Linux and MAC sides was implemented to enable Linux to control the whole system.

***Keywords*—fpga, embedded system, porting Linux, development board, wireless network**

I. Introduction

The radio spectrum is a limited natural resource. Due to increased traffic in wireless networks, there is a need to use the radio spectrum more efficiently. Therefore, advanced wireless networks, such as cognitive radio networks (CRNs), are researched actively nowadays. The CRN is a radio system employing technology that allows the system to obtain knowledge of its operational environment and dynamically and autonomously adjust its operational parameters according to learning results it has obtained [1]. In order to test CRNs and other advanced wireless networking theories in practice, a real life test environment needs to exist.

The purpose of this work is to simplify management of devices in a wireless test network and make it easier to form the network. Therefore, the Linux operating system was ported to the second generation wireless open access research platform (WARPv2). The implementation is based on the previous Linux porting project done in [2]. Nevertheless, new porting was required because of updated hardware in our wireless research infrastructure. The first generation WARP uses Xilinx Virtex-2 field-programmable gate array (FPGA) while the second generation platform uses Virtex-4 FPGA.

The WARPv2 is a FPGA based research platform for wireless networks. It is developed by Rice University, USA. Linux was chosen for this work because it is an open source operating system, which means it is easy to modify for different purposes. Moreover, Linux is well documented and it has a wide software support. With Linux, the network layer and the layers above can be managed. It would demands a lot of work and knowledge to implement a network layer without an operating system. Linux implements network layer functionality very well. Hence, research scenarios are much easier to implement. There is no need to concentrate on an unrelevant low level functionality. It is possible to control all the layers of the open systems interconnection (OSI) model with the cooperation of Linux and the medium access control (MAC) implementation of the WARPv2. This is mandatory when researching advanced wireless networks. Furthermore, Linux reduces the complexity of the system by controlling the hardware resources of the system. It hides hardware from software running the top of it by providing common services, such as filesystems, networking, and interfaces to use hardware via drivers. Linux also enables more powerful advanced wireless network researching because it makes developing software for a platform much easier due to the higher abstraction level than without the operating system (OS). There is also a lot of ready-made software, which support researching efforts, available for popular operating systems such as Linux. For example, applications for monitoring and measuring the traffic of the network are available. These are reasons why porting an operating system to a research platform is an attractive idea when researching advanced wireless networks.

The remaining structure of this paper is organised as follows. The next section introduces the hardware platform used in this work. The wireless open access research platform (WARP) orthogonal frequency-division multiplexing (OFDM) reference design is described in Section III. Sections IV and V talk about hardware and software designs of the Linux porting. The building process of the implemented system is described in Section VI. Test results are shown in Section VII. The Final section concludes the paper.

The authors would like to acknowledge the valuable comments and funding from the CORE+ research consortium: VTT Technical Research Centre of Finland, University of Oulu, Centria University of Applied Sciences, Nokia Solutions and Networks, PehuTec, EXFO, Elektrobit, Anite, Rugged Tooling, Finnish Defence Forces, Finnish Communications Regulatory Authority, and Tekes.

CROWNCOM 2014, June 02-04, Oulu, Finland

DOI 10.4108/icst.crowncom.2014.255398

II. Hardware Platform

Wireless open access research platform (WARP) is a programmable wireless platform developed by Rice University, USA. It is being actively used for research in many areas like physical, MAC and network layer algorithms, routing, and cognitive radios as can be seen in [3], [4] and [5]. The main purpose of WARP is to provide a scalable and extensible platform for researching wireless networks. This work concentrates on the second generation WARP platform. This version is designed around a Xilinx Virtex-4 FPGA [6]. Figure 1 introduces the devices and input/output (I/O) ports of WARP board version 2.2. [7]

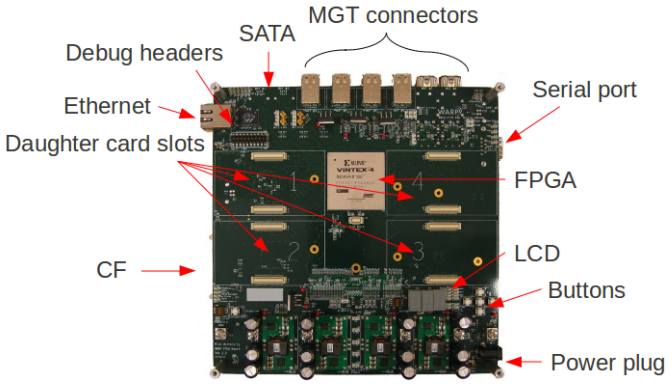

Fig. 1. WARP board overview.

III. The WARP OFDM reference design

Rice University provides an OFDM references design for the WARP board. This design implements a real-time network stack on it; it includes the OFDM physical layer (PHY) and customizable MAC interface. The purpose of the reference design is to demonstrate the full MAC/PHY capabilities of the WARP and provide a flexible base for additional applications. As seen in Figure 2, the design consists of a multiple in multiple out (MIMO) OFDM physical layer, a flexible MAC interface including a basic application implementing a wireless MAC protocol and low-level PHY control, and peripherals such as timers and radio bridges. All processing including hardware control, signal processing and the MAC protocol, is executed in real-time. [7] The OFDM reference design is shipped with a simple MAC application called *csmamac*. The algorithm of *csmamac* is simple. If there is a packet to send, the appilication sends it when the medium is idling and waits for an acknowledgment from the destination. When there is a packet received, the application checks if the packet is addressed to itself and sends an acknowledgment to a sender in that case. Otherwise, the packet is dropped. Moreover, *csmamac* has a resend functionality; if an acknowledgment is not received after fixed time, a packet is sent again. [7]

IV. Hardware design

The goal of this work was to port the Linux OS for the WARPv2 board. The implementation is based on the previous Linux porting project done in [2]. The purpose of the port was to get control about all the layers of the OSI model. Therefore, the functionality of the OFDM reference design version 16.1

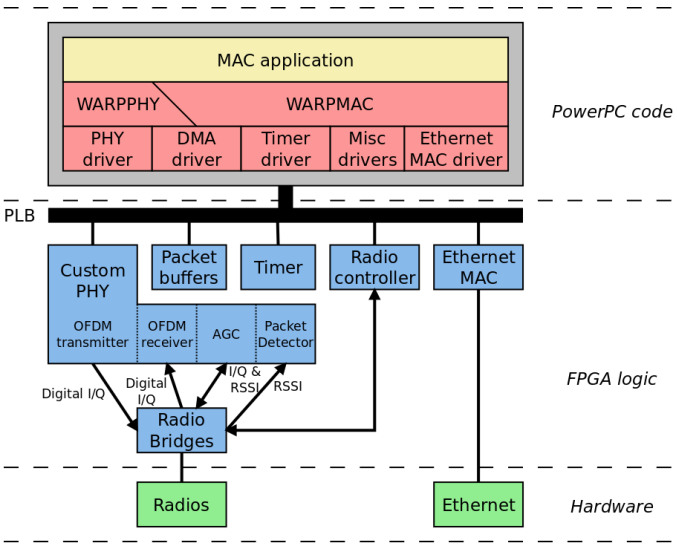

Fig. 2. The structure of the WARP OFDM reference design.

provided by Rice University was included in the final system. The reference design provides implementations for the physical layer and the MAC functionality of the datalink layer of the OSI model, so the role of Linux was to implement the Ethernet part of the datalink layer and the network layer. The OSI model implementation of the final Linux enriched second generation wireless open-access research platform (LE-WARPv2) system is illustrated in Figure 3.

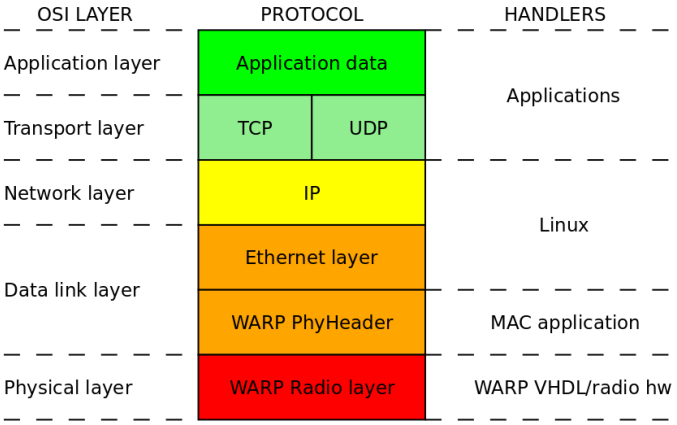

Fig. 3. The OSI model implementation.

The reference design contains the MAC functionality and the OFDM physical layer. It is designed to use only one of performance optimization with enhanced RISC performance computing (PowerPC) 405 cores present in the FPGA of the WARPv2 board. Therefore, the LE-WARPv2 system was divided into two parts, a MAC side and a Linux side, as seen in Figure 4. The MAC side uses central processing unit (CPU) 0 and it runs a simple OS called Standalone OS provided by Xilinx. Standalone OS offers low level software modules used to access hardware specific functions in order to run applications [8, p. 1]. The MAC side runs an application, which implements the MAC functionality and controls the physical layer. The Linux-side uses CPU 1. It has the Linux kernel running on it with a set of utilities and application programs.

Linux supports PowerPC architecture and the particular processor type used in the WARPv2 board. Therefore, main tasks in the porting process was to create a suitable FPGA hardware design for the combination of the Linux side and the MAC side, implement a method for communication between these sides, and configure the kernel to fit to the hardware design. The Linux system was implemented by using Buildroot for a root filesystem and cross-compilation toolchain generation, and Busybox for userspace utilities and applications.

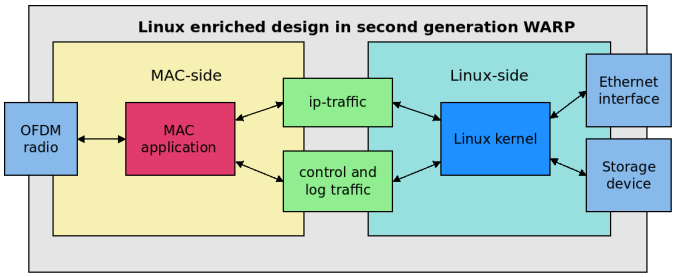

Fig. 4. Functional overview of the Linux enriched WARP system.

In order to port Linux to the WARPv2 board, the FPGA design has to contain necessary hardware. Linux requires at least a processor, a system bus, main memory, and a storage device for filesystems. However, the final goal in this work was to implement a LE-WARPv2 system with the functionality of the OFDM reference design version 16.1. Therefore, the final FPGA hardware design is based on the OFDM reference design. Hardware for Linux was added to this design. A PowerPC 405 core was added as a processor dedicated to Linux and processor local bus (plb) was added as a system bus. All peripherals controlled by Linux were connected to this system bus. Additionally, system advanced configuration environment (SystemACE) and Ethernet MAC intellectuel property (IP) cores were moved from the MAC side to the Linux side by connecting them to the system bus of Linux. SystemACE controls the usage of a CompactFlash (CF) card and the filesystem of Linux is stored to CF card. Therefore, it was important that Linux was able to control SystemACE.

The OFDM reference design does not use external double data rate (DDR) memory of the WARP board because the memory footprint of the MAC application is low and because external memory is too slow for the requirements of the physical layer. Thus, external memory was defined as a main memory for Linux. There are several methods to connect an external memory controller, multi-port memory controller (MPMC), to a processor. For example, MPMC can be connected directly to the processor or it can be connected via plb. In this work, MPMC was connected to Linux core via plb even using the direct connection would have offered better performance. The plb based architecture has been chosen due to simpler implementation and because performance provided by it was sufficient.

Hardware of the OFDM reference design needed to be modified to co-operate correctly with the hardware of the Linux side. A new bus for the MAC side was added to improve the performance of the MAC side. Memory for a heap and a stack of the MAC side was connected to this bus. Other peripherals of the MAC side remained intact.

Two mailboxes and shared memory was added to the final design. Other ends of mailboxes were connected to the system bus of the MAC side and other end to the system bus of Linux. Shared memory was connected as well between the buses of MAC and Linux sides. Simplified illustration of the final FPGA hardware design can be seen in Figure 5.

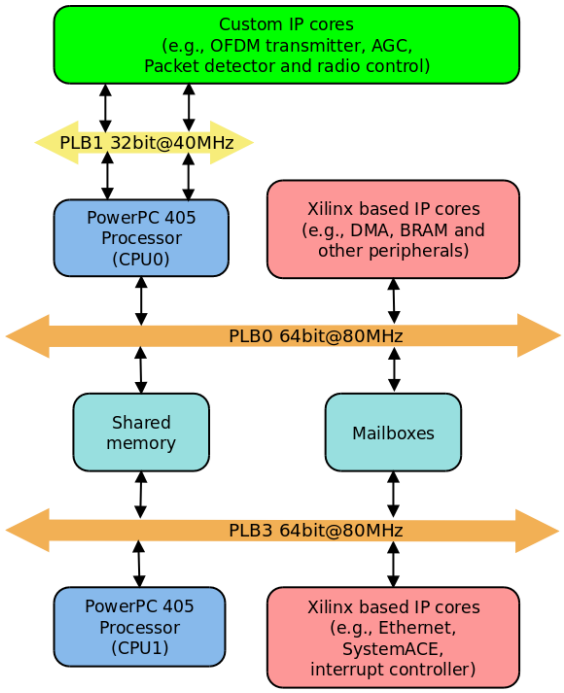

Fig. 5. Simplified FPGA hardware design.

A. *Interprocessor communication*

The communication between MAC and Linux sides was implemented with mailboxes and a shared memory block. Generally, the mailbox IP core implements a method to pass messages between one or more senders and receivers. It forms a transmission control protocol (TCP)/Internet protocol (IP)-like message channel where messages are queued in by senders and dequeued by receivers. Mailboxes support both synchronous and asynchronous methods for the reception of messages. In the synchronous method, polling is used to detect new data on the mailbox, whereas the asynchronous method uses interrupts.

The mailbox IP core provided by Xilinx uses two first in first out (FIFO) buffers; one for transmit and one for the reception. A buffer uses either distributed random access memory (RAM) or block RAM (BRAM) and the size of it is configurable. Mailboxes were used to inform other side about new data in shared memory.

The final FPGA design included shared BRAM memory and two mailboxes. One of the mailboxes was dedicated to IP-traffic and one for controlling and logging traffic between processor cores. The sizes of buffers of mailboxes were limited to 16 kB and they used the synchronous method to pass messages. Mailboxes were polled constantly to detect when new data was available in the shared memory. The shared memory was divided into fixed size slots, which held any large data that was to be transferred. The structure of this mailbox-based intercore communication solution is illustrated in Figure 6.

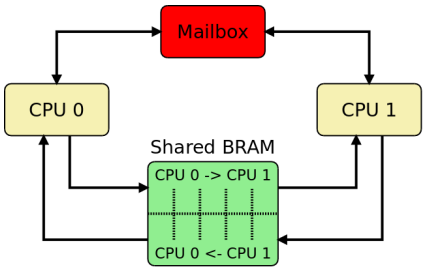

Fig. 6. Intercore communication structure.

V. Software design

Typically, software needed in the embedded system is highly tied to hardware. This was the case in this work; the kernel configuration depended on hardware, and drivers for communication between cores could not be written until it was known how the communication will be implemented at the hardware level. Although it was known at the beginning that there will be Linux running on top of the hardware. Therefore, when designing hardware, it was taken care that either Linux was supporting for chosen hardware solutions or it is possible to create Linux support with a reasonable workload. There was support in Linux for all IP cores used in the final FPGA hardware design except interprocessor communication. Thus, software design tasks were to configure the Linux kernel to match the FPGA hardware design, build a root filesystem and needed utilities and applications, and write drivers for interprocessor communication.

A. Kernel configuration

The Linux kernel used in this paper is based on 2.6.28 from Xilinx repository. Several patches and two custom drivers were added to achieve stability and interprocessor communication. The kernel was configured to support devices in the target board. The configuration was done without modules to optimize memory usage.

The PowerPC Linux uses device-tree to describe the hardware system such that the kernel can configure itself during the boot. Thus, the device-tree matching designed hardware had to generate before the kernel compilation. The device-tree was generated using the device-tree generator provided by Xilinx. The created device-tree file was added to the kernel source tree in order to build a working kernel image.

B. Communication between PowerPC cores

The communication interface was designed in order to enable communication between Linux and MAC sides. Two protocols using it was designed. One for handling communication between the Linux and the MAC sides and another for logging and controlling purposes.

The wireless interface device driver sets up a standard Linux Ethernet interface and send and receive packets to the MAC side using a mailbox and shared BRAM memory. The driver does not contain wireless extensions like wireless local area network (WLAN) drivers do; Linux sees this interface as a regular Ethernet interface. The driver probes mailboxes during the kernel boot up, and when the correct mailbox is found, it sets up the Ethernet interface. MAC and IP addresses are set by using standard Linux tools. The MAC address is sent to the MAC side in order to filter network packets by a MAC application.

Buffering between Linux and the Ethernet driver is done in the Xilinx platform studio (XPS) Locallink tri-mode Ethernet MAC (TEMAC) IP core and the Xilinx kernel driver for that IP core. The kernel driver is connected to the Linux IP stack. Buffering between cores consists of shared memory. The driver copies packets to the shared memory and notifies the MAC side with a mailbox message. Similarly, when receiving packets from the MAC side, packets are copied from shared memory and delivered to the kernel. The kernel informs the MAC side when it has copied a packet from the shared memory. On the MAC side, packets received from Linux are copied to the buffers of the PHY using direct memory access (DMA). Also packets send to Linux are delivered from PHY buffers to the shared memory using DMA.

The logging protocol is used to handle control communication and logging messages. The logging device driver is a character type driver which provides mechanisms to write data to the shared memory between Linux and MAC sides, and notify other side via mailbox that there is data in the shared memory. Two userspace applications were developed to use implemented device drivers.

C. Root filesystem

Linux requires a root filesystem in order to use all its services and features. The root filesystem is the filesystem mounted at the base of the filesystem hierarchy; any other filesystem is mounted under the root filesystem. A Linux system has special requirements for the root filesystem. It has to contain applications and utilities to boot a system, initialize services such as networking, load device drivers, and mount other filesystems. [9]

The root filesystem for Linux in the WARPv2 board was created by using Buildroot [10]. It was used because it provides a simple and efficient method to create a root filesystem with a variety of userspace applications and libraries. Additionally, Buildroot supports Busybox [11] which provides large number of small versions of common utilities. Buildroot was also used to generate the uClibc-based cross-compilation toolchain required for compiling userspace applications.

D. Applications

There are numerous applications and utilities needed in the embedded Linux system. In the custom Linux system, there is not any package manager available which provides method for installing application binaries. Therefore, all essential applications and utilities have to compile from source code to match the target system. Busybox provides an efficient manner to compile the most of the required applications and utilities. Additionally, with Busybox, sizes of application binaries are smaller than they are when compiled independently, because Busybox uses stripped versions of applications and utilities. Therefore Busybox is a reasonable choice for any embedded Linux project.

Using Busybox does not remove the option to compile additional applications or utilities and add them to the filesystem. Therefore it was possible to add applications not supported by Busybox to the system without issues. These applications were

compiled to PowerPC architecture using the cross-compiler provided by Buildroot, and binaries were moved to the filesystem of Linux.

The MAC application was modified to send and receive Ethernet packets to Linux via implemented interface. This was done by removing Ethernet MAC core references from the source code, and replacing them so that payloads of outgoing Ethernet packets were forwarded to the shared memory and headers of these packets were forwarded to the mailbox. The incoming Ethernet packets handling functionality was changed so that the status of the mailbox was polled in order to notice when new Ethernet packets will be available. Additionally, the original MAC application code was improved by adding control command handling and data gathering functionalities for logging purposes to the source code. These changes enabled Linux to take control about the MAC layer and to get logging data from the MAC side.

VI. Building the system

All required software for the LE-WARPv2 system can be installed in a single CF card. The system will bootup automatically when the board is powered up and the CF card is mounted to the board. When building a bootable system to a CF card, it is required that the CF card has to have a boot partition as the first partition on the card. It is mandatory that the boot partition contains a valid Microsoft disk operating system (MSDOS) filesystem, since it is the only filesystem type SystemACE supports. The advanced configuration environment (ACE) file containing hardware design, the Linux kernel and application executables is stored to the boot partition. Other partitions needed were ext2-type partition for Linux root filesystem and FAT32-type partition for data such as log files. However, many other filesystem types can be used as well.

The FPGA hardware design and MAC software was created using Xilinx XPS tools. These tools produced a bitstream file which contained the FPGA hardware design and the compiled MAC application. The Linux kernel image is required to combine with the bitstream file in order to produce a bootable ACE file. The cross-compilation toolchain of Xilinx XPS is not suitable for compiling the kernel due to lack of parts its version of GNU C compiler (gcc). Therefore, Linux had to be compiled with a toolchain which had a full support of gcc. Crosstool-NG was used to compile the kernel. Before Linux can be compiled for the WARPv2 board, the device-tree file has to be generated and added to the kernel source tree. In order to create the final ACE file, the Linux kernel image and the bitstream file have to be combined. This was done using SystemACE file generator (GenACE) script from Xilinx XPS. Created ACE file contained all information about how to boot up both CPUs from CF card, and it was copied to the boot partition of the CF card.

The building process was mitigated by scripts. These scripts automate the partioning of the CF card, producing the the bitstream with MAC application, combining the bitstream with the Linux kernel image, creating root filesystem, and copying needed files to the CF card. A configurable master script including all scripts listed earlier was also written to provide user friendly interface for building the LE-WARPv2 system.

VII. Test results

Several test cases were created to verify that the LE-WARPv2 system behaves as planned. Test cases were divided into two sections: hardware tests and performance tests. Hardware tests aimed to verify that hardware of the LE-WARPv2 system was properly configured and the performance of hardware components was decent, whereas the purpose of performance tests was to confirm that performance of the whole system enabled demands of realtime operation.

Throughput between two WARPv2 boards was measured using both the Linux enriched design and the OFDM reference design. The carrier sensing medium access (CSMA) MAC application was used in the measurement. The application was modified before measurements, because it had programming error which caused poor TCP throughput performance with the OFDM reference design. Boards were connected with coaxial cables and HP 8496B 60 dB attenuator instead of a wireless link. This was done in order to get more reliable and comparable results. Measured throughput values were compared. Measurement was done using user datagram protocol (UDP) and TCP protocols. *Iperf* program version 2.0.5 was used as measurement software.

Test results of UDP throughput are introduced in Figures 7 and 8. According to test results, the throughput performance of the Linux enriched design when using UDP was similar to the OFDM reference design. TCP results are shown in Table I. Hence, the throughput performance of the Linux enriched design is suitable for researching purposes.

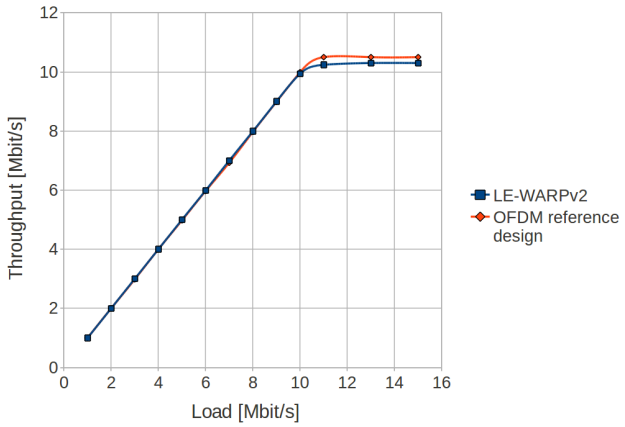

Fig. 7. UDP throughput.

TABLE I. Throughput comparison using TCP

Design	Transfer (Mbytes)	Bandwidth (Mbit/s)
The OFDM reference design	62.6	5.24
The Linux enriched design	48.7	4.09

A. FPGA resource usage analysis

Used FPGA resources in the OFDM reference design and the Linux enriched design were compared. The Linux enriched design uses only 10 percentage points more slices than the OFDM reference design. The Linux enriched design demands

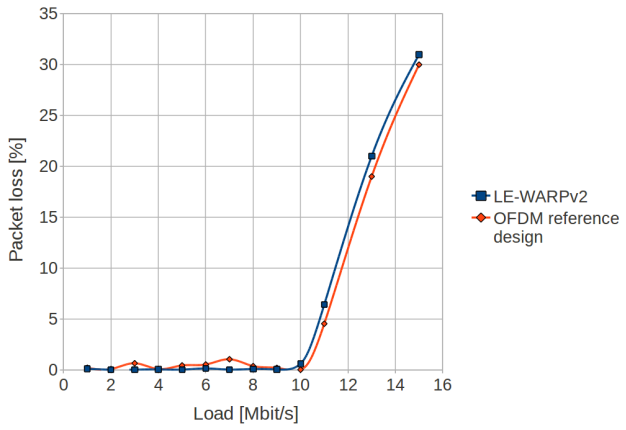

Fig. 8. UDP packet loss.

86% of available slices of the FPGA. Therefore, the FPGA can be considered as heavily utilized. The Linux side of the design might be difficult to reduce without loosing features. However, there is a lot of room for improving the MAC side design, since some functionalities, such as packet detector, have more than one implementation in the design.

VIII. Conclusion

LE-WARPv2 extends the features of the WARPv2 board. It provides a feature-rich platform for advanced wireless networks researching without a need to manage a full complexity of the platform; Linux offers interfaces for applications to use hardware of the WARPv2 board efficiently and it has wide variety of tools and utilities available required in the advanced wireless networks research. Furthermore, Linux makes possible to control the MAC layer of the WARPv2 board. LE-WARPv2 boards can be connected together in order to form a network. This makes possible to form a environment for advanced wireless networks researching. In other words, LE-WARPv2 enables the researchers of advanced wireless networks to set up a wireless test network to test and verify complex theoretical algorithms at different layers of OSI model and find out the practical limitations of theoretical results.

References

[1] J. Mitola and J. Maguire, G.Q., "Cognitive radio: making software radios more personal," *Personal Communications, IEEE*, vol. 6, no. 4, pp. 13–18, 1999.

[2] M. Jokinen and H. Tuomivaara, "LE-WARP: Linux enriched design for wireless open-access research platform," in *Proceedings of the 4th International Conference on Cognitive Radio and Advanced Spectrum Management*, ser. CogART '11. New York, NY, USA: ACM, 2011, pp. 16:1–16:5. [Online]. Available: http://doi.acm.org/10.1145/2093256.2093272

[3] H. Tuomivaara, M. Raustia, and M. Jokinen, "Demonstration of distributed TDMA MAC protocol implementation with OLSR on linux enriched WARP," in *WiNTECH'09 - Proc. 4th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, 2009, pp. 85–86. [Online]. Available: www.scopus.com

[4] A. Greco, T. Milcher, V. Kolar, P. Mähönen, and M. Petrova, "CSMA interaction detection and capacity estimation in cognitive radio networks," in *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '09. New York, NY, USA: ACM, 2009, pp. 321–322. [Online]. Available: http://doi.acm.org/10.1145/1530748.1530794

[5] Z. Chen, C. Zhang, F. Lin, J. Yu, X. Li, Y. Song, R. Ranganathan, N. Guo, and R. C. Qiu, "Towards a large-scale cognitive radio network: Testbed, intensive computing, frequency agility and security," in *2012 International Conference on Computing, Networking and Communications, ICNC'12*, 2012, pp. 556–562. [Online]. Available: www.scopus.com

[6] "Virtex-4 Family Overview," Xilinx, Tech. Rep., August 2010, uRL: http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf.

[7] "Rice University WARP Project, (accessed 3.1.2012)," uRL: http://warp.rice.edu.

[8] "OS and Libraries Document Collection," Xilinx, Tech. Rep., 2010, uRL: www.xilinx.com/support/documentation/sw_manuals/xilinx12_4/oslib_rm.pdf.

[9] C. Hallinan, *Embedded Linux Primer: A Practical Real-World Approach*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2006.

[10] "Buildroot: making Embedded Linux easy, (accessed 20.7.2012)," uRL: http://buildroot.uclibc.org.

[11] "BusyBox: The Swiss Army Knife of Embedded Linux, (accessed 20.7.2012)," uRL: http://www.busybox.net.