2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing

Linux enriched design in third generation Wireless Open Access Research Platform

Muhammad Saad Saud, Anton Paatelma, Markku Jokinen, and Tuomo Hänninen
Centre for Wireless Communications
University of Oulu, Finland
Email: {muhammad.saud, anton.paatelma, markku.jokinen, tuomo.hanninen}@ee.oulu.fi

***Abstract*—This paper describes embedded Linux system implementation on third generation Wireless Open Access Research Platform (WARP v3). The purpose of this implementation is to integrate operating system (OS) with radio communication system to accomplish flexible and stand-alone communication node for testing different wireless communication protocols. Achieved Linux Enriched WARP v3 (LE-WARP v3) design is a dual (micro-processing) core implementation on field-programmable gate array (FPGA) with Linux running on one core, medium access control (MAC) implemented on other core directly interfacing radio peripherals. Inter-processor communication between both cores is implemented. Hence, achieved result is a versatile system which allows altering any layer of communication network stack, making it practical to test algorithms and prove concept solutions in real-time environment.**

***Keywords*-field-programmable gate array (FPGA); embedded systems; Linux; MicroBlaze; wireless communication;**

I. Introduction

In today's world communication is becoming a necessity in everyday life. People are used to being connected to others and sharing their experiences instantly and even on the move. This puts pressure on developing efficient wireless networks to cope with increasing traffic demand. One solution on acquiring more communication resources, i.e., spectrum, is from bands which are currently underutilized by their traditional users. These techniques are currently under heavy research, like Licensed Shared Access (LSA) [1] in Europe and Spectrum Access Systems (SAS) [2] in USA. These systems require more network control than traditional Fixed Spectrum Access (FSA) to avoid creating interference with users having primary status on the band. Researching these and even more advanced systems in practice requires a flexible test-bed. Our solution is to develop a test-bed that can be modified on all layers of open systems interconnection (OSI) model.

Wireless test-beds are designed to test and prototype latest algorithms for overcoming different real-world challenges such as congestion control, resource allocation, etc. Linux Enriched Wireless Open Access Research Platforms (LE-WARP v1 [3] and LE-WARP v2 [4]) are such systems implemented on the first and second generation Wireless Open Access Research Platform (WARP v1 and WARP v2) from open-source WARP Project [5]. WARPs with their reference system designs are well suited for research on MAC and physical (PHY) layers but lack upper layers of OSI model, therefore adding an OS to this platform will meet the need for adding remaining layers of the stack and providing a stand-alone system for wireless research.

Major changes between WARP v2 and WARP v3 require a complete redesign of LE-WARP environment. New FPGA, which has more resources available, but do no longer have embedded PowerPC processors for running software, makes it necessity to use Microblaze soft processor cores on FPGA. This means new architecture for running Linux OS. WARP v3 and its added hardware resources enables the release of IEEE 802.11 reference design [6]. Compared to old OFDM reference design, supported with WARP v2, the new design is fully compatible with 802.11 standard and can be connected with commercial devices. Whereas the old design, even though based on 802.11 standard, has such fundamental differences on the standard that only WARPs could connect to each other. We had a design choice to make, whether we should use this new reference design as a base to test 802.11 compliant devices or to use PHY and MAC partly for distinct need. We decided to modify and use the new 802.11 reference design with our LE-WARP v3.

LE-WARP design enables building a self-contained network out of WARP nodes. Previously we have been using mobile ad hoc network (MANET) type of heterogeneous network [7], but more recently studies have been focusing on centralized/backhaul networks imitating a real-world mobile networks which contain base stations (BSs) and user equipments (UEs). Figure 1 describes a real-time implementation example for Co-Primary Spectrum Sharing (CoPSS) with inter-operator device to device (D2D) communication from [8]. This is an example of a network test-bed achieved using LE-WARPs. In [9], software defined network (SDN) concept has been studied with cognitive radio (CR) using similar BS and UE scheme.

Building a network requires nodes with routing capabilities. The LE-WARP design offers an easy way for implementing this, because Linux OS can handle network layer functionalities. There are many routing applications available for Linux, optimized link state routing protocol (OLSR) and routing information protocol (RIP) have been used with previous versions of LE-WARP designs. Linux on board makes it possible to run applications for monitoring and measuring the network traffic in all the nodes. LE-

978-1-5090-0154-5/15 $31.00 © 2015 European Union
DOI 10.1109/CIT/IUCC/DASC/PICOM.2015.57

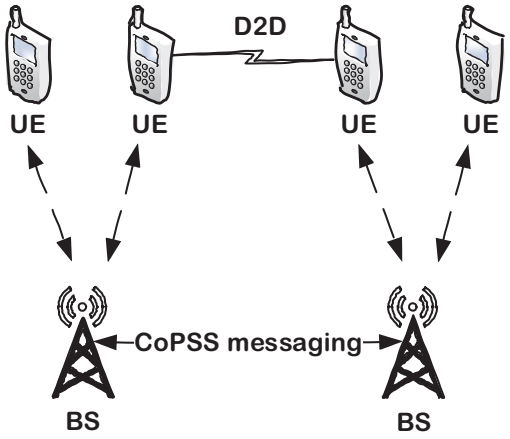

Figure 1. LE-WARP Co-Primary Spectrum Sharing demonstration using LE-WARPs.

WARP design enables the illustration of the network with a graphical user interface (GUI), by allowing us to route information packets about the network status to the machine running purpose built software. Illustration of the network status is a very practical feature for creating understandable demonstration.

An alternate approach of implementation for similar test-beds is to use software defined radios (SDRs) such as Ettus Research's Universal Software Radio Peripheral (USRP) [10]. For example, in [11] a test-bed for single and multi-relay cooperative communication schemes is implemented and in [12] a test-bed for cognitive radio (CR) to show abilities of different SDR and CR systems to coexist in common frequency bands is implemented using USRPs. In these implementations, a node consists of RF front-end implemented in USRP and rest of processing is taken in general purpose computers. Our proposed solution will remove the need of general purpose computer for processing, adding value to demonstrations due to stand-alone node feature and ease of application specific development and setup.

The rest of the paper is organized as follows. Section II gives a brief introduction to WARP v3 hardware. Section III gives implementation details of the system. It covers architecture of Linux side peripherals, MAC side peripherals, and the reference design which was used as base for MAC side peripherals. Section IV describes benchmark results for implemented system. Finally we conclude this paper in Section V.

II. Hardware Platform

LE-WARP v3 is implemented on Mango Communication's WARP v3, which is a FPGA based research platform as illustrated in Figure 2. FPGA is a programmable (semiconductor) device which has a matrix of Configurable Logic Blocks (CLBs) connected through programmable interconnects for implementing digital logic designs [13]. There are different kinds of available FPGAs, e.g., one-time program (OTP) FPGA and reprogrammable static random-access memory (SRAM) based FPGAs. WARP v3 consists of Xilinx's Virtex-6 (SRAM type [13]) FPGA, with two radio frequency (RF) interfaces, small outline dual in-line memory module double data rate type three random access memory (SO-DIMM DDR3 RAM) slot, two ethernet slots (ETH A, & ETH B), secure digital (SD) card slot, serial peripheral interface (SPI) flash and one FPGA mezzanine card high pin count(FMC HPC) slot so that one may add any high speed physical peripheral [5]. SPI flash and SD card are interfaced via complex programmable logic device (CPLD) which makes it possible to use same devices for differnt purposes.

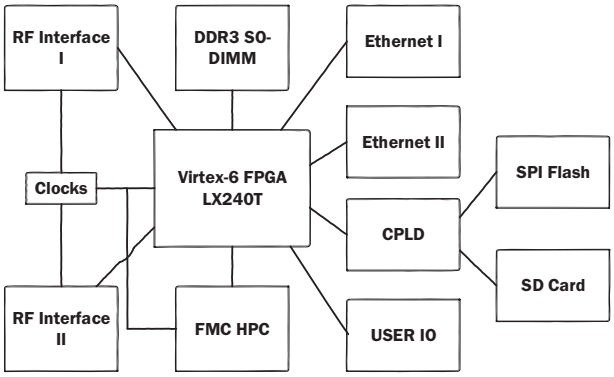

Figure 2. WARP v3 board overview.

III. Implementation Details

In our proposed system, there are two soft core Xilinx's Microblaze processors, one running Linux and other used for MAC. The system architecture is implied in Figure 3, where addressable intellectual property (IP) cores interfaced with buses and other control cores are shown. Details for all Xilinx Platform Studio (XPS) and advanced extensible interface (AXI) IP cores can be found in [13]. Other cores like System Reset and Clock Generators are used for providing reset and clocks to system. Universal asynchronous receiver/transmitter multiplexer (UART MUX) is used to multiplex UART console from Linux or MAC using a DIP switch on board.

In Section III-A, we discuss details of reference design for IEEE 802.11 standard implemented on WARP v3 by WARP Project which will provide a base for our design and better understanding for MAC peripherals. Linux side peripherals and other implementation details are discussed in Section III-B and MAC side details can be found in Section III-D. We have designed a system that allows LE-WARP v3 node to be configured as a BS or UE while booting. Differences between both modes have been mentioned

in Section III-D. Section III-C describes inter-processor communication.

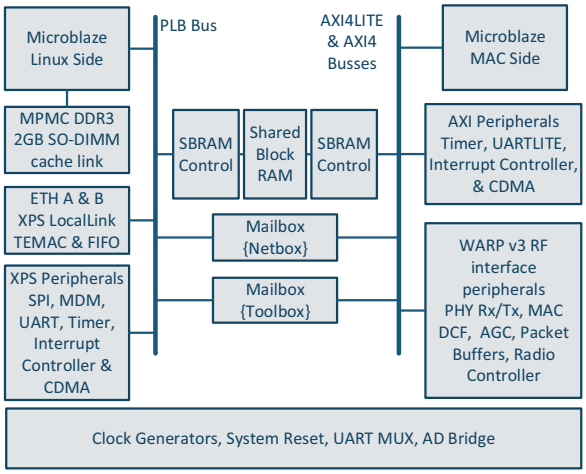

Figure 3. LE-WARP v3 system architecture overview.

A. 802.11 Reference Design

The 802.11 reference design is a real-time implementation of IEEE 802.11 orthogonal frequency division multiplexing (OFDM) PHY and distributed coordination function (DCF) MAC [6]. This design has capability to interact with commercial 802.11 devices, both ways, i.e., as access point (AP) or station (STA).

The detail of reference design architecture [5] is presented in Figure 4, which serves as a base for MAC side peripherals in our proposed design. The 802.11 reference design is itself a dual core design with microprocessors named as CPU High and CPU Low. CPU High in reference design executes top-level MAC and other high level functions. It constructs all non-control packets for transmission and for implementing handshakes between nodes, e.g., probe request/response, association handshakes, etc., but relies on lower-level DCF MAC for actual medium access.

One of the top-level MAC's functions is to integrate system with wired network, Linux side takes care for this functionality in the new design. All top-level MAC or low-level MAC functions in LE-WARP v3 are shifted to MAC side equivalent of CPU Low. CPU Low in reference design implements low-level functions for MAC DCF core. It is responsible for MAC-PHY interactions, and handling time critical intra-packet state. reference design provides softwares for both high and low CPUs. MAC DCF core (hardware) is FPGA core implementation (in System Generator) for interfacing MAC software and PHY Rx/Tx cores. This core implements timers needed for DCF, e.g., timeout, backoff, and various carrier sensing schemes, etc. It monitors and sequences Tx and Rx events via mechanism provided by MAC software. PHY Rx/Tx implements OFDM physical layer transceiver. Hardware support cores are platform support cores for WARP v3. These include analog/digital (AD) bridge, radio controller, and physical automatic gain control (AGC), etc., to enable control of various peripheral interfaces on the board from CPU Low. AD Bridge interfaces digital I/Q from analog-to-digital converters (ADCs) and to digital-to-analog converters (DACs) on radio board. Radio controller core implements interface to configure transceiver's mode and other radio front end settings, e.g., power amplifier and RF switch. AGC is used between PHY Rx and AD bridge for adjusting Rx gains.

Hence, reference design provides an extensible, configurable, and open design which has taken good care of PHY and MAC implementation for IEEE 802.11 standard. Since this standard is widely used today as WiFi products, it makes this design favorable to accomplish research with impact.

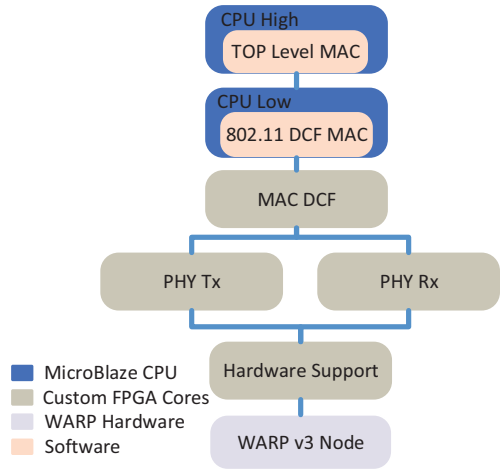

Figure 4. 802.11 reference design.

B. Linux Details

In order to run Linux over the WARP v3 board following considerations need to be taken into account. The first step is Linux configuration to run it on microprocessor architecture, in our case: Microblaze 8.40.b clocked at 160 MHz. Microblaze is Xilinx's FPGA-based soft processor with advanced architecture options like AXI or processor local bus (PLB) interface, memory management unit (MMU), instruction and data-side cache, configurable pipeline depth, and floating-point unit (FPU), etc. Xilinx provided minimum hardware requirements for MMU based Linux kernel, which are: (a) MMU in virtual mode and two memory protection zones, (b) a timer (XPS timer, with configuration of two programmable timers in it), and (c) a UART for console [14]. Latest Xilinx's Linux kernel can be found from Xilinx repository [14], which provides drivers for all the peripherals mentioned

in this section except inter-processor communication which was implemented for our distinct need as will be discussed in Section III-C.

Linux Microblaze uses PLB, which is why it is configured for big endianness. We added multi-port memory controller (MPMC) for interfacing DDR RAM. XPS MPMC uses chache link to communicate Microblaze processor and is not directly added to PLB. The two ethernet ports available on board (ETH A and ETH B) are connected to a Marvell 88E1121R dual ethernet PHY, implementing two tri-speed ethernet PHYs. We used Xilinx's hard tri-speed ethernet MAC (TEMAC) cores for interfacing both PHYs. Data packets are received via LocalLink first-in-first-out (FIFO) cores which receive packets from TEMAC core via LocalLink without any need for direct memory access (DMA).

XPS Microblaze debug module (MDM) IP core is added to PLB for debugging both processors, i.e., Linux side and MAC side. XPS SPI core is added to interface SD Card, which holds linux file-system. XPS central DMA (CDMA) is added to PLB as master and slave peripheral for future modifications. The two mailbox IP cores (Netbox and Toolbox) and shared block RAM (SBRAM) are used for inter-processor communication as covered in its respective subsection. XPS interrupt controller adds interrupts of both TEMAC cores and their respective FIFOs, timer, UART, SPI, CDMA, and both mailbox IP cores.

Xilinx also provides a pre-built root file-system and cross compilation toolchains for making the kernel, user space applications, etc. Another option can be to generate file system and cross compilation toolchain from Buildroot [15]. Buildroot is a tool for simplifying and automating process of building a complete embedded Linux system, using cross-compilation. It can generate cross-compilation toolchains, root filesystem, Linux kernel image, and bootloader or any combination of mentioned options. To have extra benefits of utilities and C library selection we used Buildroot and built root file system and cross compilation toolchain on embedded glibc (EGLIBC [16]). EGLIBC is a variant of well known GNU C library (GLIBC) designed for embedded systems with features of reduced footprint, configurable components and better support for cross compilation/testing.

Some considerations had to be taken for making this system bootable. A CPLD is present on board to interface SD card and SPI flash. CPLD is a combination of a fully programmable AND/OR (gate) array and a bank of macrocells [13]. The gate array is a re-programmable structure for implementing combinational logic functions. Macrocells are higher level structures in hierarchy than gate array which can be programmed both as combinational logic or sequential logic, e.g., flip flops. Unlike many FPGAs, CPLDs have non-volatile configuration, i.e., on power off it will not lose its configuration.

WARP Project provides CPLD design for reading FPGA configuration from SD card. WARP v3 supports 2 GB SD card for FPGA configuration. The design loads a binary format configuration file from SD card starting at an offset of 64 MB or address 131072. Up to 8 configuration files can be stored per card with a separation of 16 MB. An active configuration file is selected by a DIP switch present on WARP v3 board. This design does not support post-configuration access to SD card. Thus, CPLD design needs necessary modification to load Linux kernel from SD card to RAM and mount root filesystem from SD card.

The new design takes SPI connections from XPS SPI peripheral mentioned in Figure 3. It raises a signal called "bootloader signal", once bootloader runing on Linux processor tries to initialize SD card. It normally takes up-to 74 (dummy) clock cycles before SD card accepts any command while initialization in SPI mode [17]. The bootloader sends specific number of additional clock cycles. Those clock cycles are counted for raising "bootloader signal". On the basis of this signal, SPI connection to SD card is provided for SPI core and disconnected from FPGA configuration pins. Hence, a bootloader which tries to initialize SD card also get access to SD card via CPLD sending additional dummy clock cycles and we get rid of additional need to a general purpose input output (GPIO) core for raising "bootloader signal".

A light weight boot loader has to be placed in Microblaze's instruction memory, i.e., BRAM interfaced with local memory bus (LMB). We used S-Record (SREC) bootloader from Xilinx's application project which takes control on startup and can load kernel from non volatile memory. Motorola's SREC [18] is a file format for conveying binary information in ASCII hex text form. Originally it is used for encoding programs or data files in printable format for transportation between systems. SREC format is known for its ease of editing/visualizing. SRECs are character strings composed of several fields to identify record type, record length, memory address, code/data, and checksum as depicted in Figure 5. A byte in binary format is represented by two characters (each representing four bits). SREC type can hold some encoding/decoding information, e.g., how many address bytes are there or to mention termination block. Record length field holds length of the whole line including type, etc. Address field holds address of code/data to be loaded. Checksum is used for certainty of a valid record. SREC format is an overhead in terms of time, but it gives satisfaction for data validity.

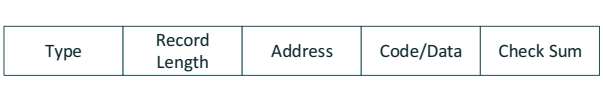

Figure 5. SREC line format.

Xilinx's SREC bootloader does not support SD card, therefore necessary changes have been made to boot from

SD card. In our proposed system, bootloader initializes SD card, expects SREC kernel image to be placed at starting address (2048). It takes image from the SD card, decodes the SREC format and starts placing kernel at memory locations beginning at entry point. Afterwards, it passes control to kernel. Kernel then mounts 64 MB second extended (EXT2) Linux primary partition from SD card positioned on non-conflicting section with kernel image and configuration binary file. In Figure 6, an example scheme of block addresses and respective space for kernel, configuration slots, and root filesystem is portrayed. One may use only one configuration slot for configuration file and start root filesystem partition at base block address of second slot. Vast space available allows us to make root filesystem partition bigger but in LE-WARP v3 64 MB partition proved to be sufficient.

16 Mb	47 Mb	16 Mb	16 Mb	80 Mb	16 Mb	64 Mb	1619 Mb
Kernel	Free	Slot 0	Slot 1	. . .	Slot 7	Root FileSystem	Free
2048 to 34815	34816 to 131071	131072 to 163839	163840 to 196607	196608 to 360447	360448 to 393215	393216 to 524287	524287 to 3839999

Figure 6. SD card fields, length, and addresses.

C. Inter-Processor Communication

Functional overview of LE-WARP v3 has been given in Figure 7. Linux and MAC side processors has to communicate for internet protocol traffic, control messaging and other log information. Communication between Linux and MAC processors is implemented using one 128 KB SBRAM and two Xilinx's mailbox IP Cores with FIFO depth of 16 words.

SBRAM is memory which is shared between both processors, so that both can read and write any chunk of that memory. SBRAM is divided into 76 data slots of size 1600 bytes for internet protocol traffic and information/control messages between Linux and MAC. We used 66 down slots (from Linux to MAC) and 10 up slots (from MAC to Linux). Down slots are more than up slots to buffer some traffic in case of Linux trying to send more than PHY rate. After slots for internet protocol traffic and information/control messages, there are slots for logging. There are 3 up slots and 0 down slots of size 2048 bytes reserved for this purpose.

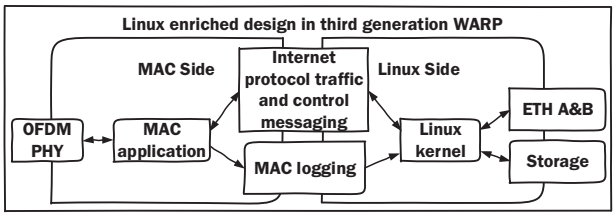

Figure 7. LE-WARP v3 functional overview.

Mailbox is used for bidirectional inter-processor communication. It has two bus interfaces to access internal resources, i.e., FIFO buffers. Communication between these two processors works with mailbox messages. It raise interrupt on Linux processor, while MAC processor poll for mailbox messages. The message provides SBRAM slot and data length/type for both internet protocol traffic data and logging. Both processors check available slots for data copy to SBRAM. Slot is kept busy unless a processor does not get ACK from the other side. After receiving ACK from the other side, slot is released and ready to be used again. In this design the two ports of mailbox are interfacing two different bus standards, i.e., Linux side has PLB and MAC side has AXI (AXI4Lite and AXI4) bus standards. PLB and AXI busses define Microblaze processor either big endian or little endian. At the MAC side, we have taken care of interpreting endianness of mailbox messages. The endianness has no effect on internet protocol traffic because end result is interpreted by OS.

From the Linux side two drivers were implemented. The first driver is network driver named as WARPNET and is used for internet protocol traffic via ethernet device and configuration of MAC using input/output control (IOCTL). Information or control messaging on the same channel as internet protocol traffic includes: time slot allocations, Tx power, coding, frequency, number of good/bad packets, beacon signals, and acknowledgement (ACK) signals, etc. Linux can configure MAC and MAC can feedback its state, etc., through information/control messages. It uses Netbox to raise interrupts on Linux side, informs data slot and length. Another character (CHAR) driver named WARPTOOL was implemented for getting log information only for debug purposes. In this setup, we only use one way messaging, i.e., MAC to Linux for providing MAC side logs which can be useful in case of some rectification.

Mailbox setup for this inter-processor communication is efficient and reliable. It takes an average of 28.23 µs time to send slot and data length information from Linux processor to MAC processor and get back ACK from the MAC processor while it is offloaded from MAC functions. The overall setup however, is even more efficient since Linux can fill the buffer upto 16 data slots (total length of mailbox FIFO) without having single ACK from MAC.

When MAC layer is under high load, latency can variate, because MAC prioritizes its functions. Latency can be around 1.1 ms for 12 Mbps physical rate example setup discussed in Section III-D, because it is set as reception time of maximum length packet. While packet is being received from physical layer, incoming data packet from Linux is ignored. However, it does not affect the performance, because system tolerance of maximum delay, e.g., 1.1 ms is much greater than actual ACK time, i.e., 28.23 µs. Hence, latency of the mailbox setup is low enough to support our inter-processor communication as mentioned in Section IV.

D. MAC Details

A Microblaze processor version 8.40.b clocked at 160 MHz is used for MAC processing. All RF interface peripherals mentioned in Section III-A are interfaced either using AXI4 or AXI4LITE busses. Additionally, we used AXI timer for timing purposes, UART Lite (UARTLITE) for console, interrupt controller for providing interrrupts to processor, and CDMA peripheral for transfering packets from SBRAM to packet buffers.

As mentioned previously, we have freedom of protocol selection at every layer. To achieve mobile network like behaviour, we implemented time division multiple access (TDMA) software MAC. It interacts with PHY using reference design MAC DCF core and with Linux using mailboxes and SBRAM. All the packet transfers inside MAC layer are done using CDMA to save prosessing resources. MAC software implementation can be roughly divided into two parts, interrupt driven and mainloop code.

Interrupt driven part handles all the timing critical functions. There are different functions for BS and UE mode for this part. In BS mode it sends syncronisation signal and beacons with slot allocations. In BS and UE mode it also handles data packet sending in appropriate times. Interrupt driven code also handles some time dependent packet transfers to PHY buffer after control slots. Uplink and downlink control slots are used for low rate data transfer between nodes, e.g., may be used for discovering link quality. In UE mode, Acknowledgement (ACK) is used for UE presence detection and its registration with BS.

Mainloop code is mainly handling packet transfers between PHY Rx/Tx buffers and SBRAM interfacing Linux processor. Because PHY Tx buffer has only 8 packet slots, 2 of them reserved for control packets in our implementation, we need to fill up PHY Tx buffer continuously to prevent interrupt driven scheduling to starve out of packets. Mainloop code is also responsible for taking care of packet reception and control traffic between Linux and MAC. Because mainloop code receives also syncronisation signal we are using MAC DCF core's timestamp feature to determinate when reception has begun to achieve better syncronisation.

Our TDMA frame starts with synchronization (SYNC). It consists of 32 data slots, one beacon slot, one downlink control data slot, one ACK slot, one uplink control data slot and at end of the frame one guard slot. Data slot allocation supports downlink, uplink and device to device (D2D) allocations. TDMA frame is illustrated in Figure 8. In our implementation we used quadrature phase shift keying (QPSK) modulation with code rate $\frac{1}{2}$ and time slot length of 1.1 ms but the parameters can be easily changed.

IV. Benchmark

We have taken few network performance benchmarks for LE-WARP v3 including maximum network throughput for ETH A and ETH B, wireless network throughput between

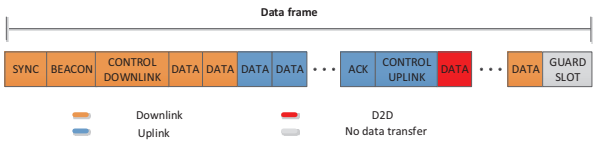

Figure 8. MAC frame structure.

LE-WARP v3 nodes. These benchmarks are taken without any performance optimization of architecture, hence better throughput results are expected. For these measurements kernel version 3.8.0 is used.

In Table I ethernet performance benchmarks for our system are shown. We used IPERF version 2 tool from [19] for network throughput measurement. In this test, IPERF server is run on Linux processor and IPERF client is run on personal computer (PC). Packets are generated on PC and ACKs are generated on Linux processor. Bandwidth for user datagram protocol (UDP) with datagram size 9 KB and transmission control protocol (TCP) is tested. In this test hard TEMAC's FIFO size is configured to be 8 KB and LocalLink FIFO adds 2 KB of more capacity in data path.

Table I
Performance Benchmarks for Linux CPU

Performance Measure	Result
TCP Throughput for ETH A&B	19.3 Mbps
UDP Throughput for ETH A&B	21.5 Mbps

Figure 9 shows wireless UDP throughput using IPERF tool between two LE-WARP v3 nodes. One LE-WARP v3 node is configured as BS and other as UE to establish wireless link. In this test setup, either of the nodes can be set as IPERF client and other node as IPERF server.

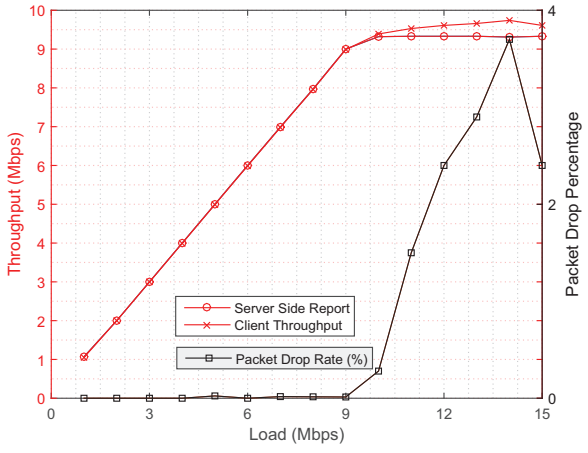

Figure 9. LE-WARP v3 wireless performance

In this test case, we consider 12 Mbps physical rate modulation/coding (QPSK/$\frac{1}{2}$) scheme and TDMA MAC with allocation of $32\frac{1}{3}$ (all data slots and every third control slot) out of 37 total slots from IPERF client to server node. Server side measurement is restricted to 9.33 Mbps, while client tries to send with higher throughput. As client tries to send above this threshold, packet drop rate increases. The achieved throughput is quite close to theoretically calculated value, i.e., 9.3425 Mbps, with such TDMA slot allocation, 1470 bytes datagram size using IPERF, and time slot length of 1.1 ms. Hence, inter-processor communication and Linux setup have supported 12 Mbps physical rate in this example setup.

V. Conclusion

In this work, LE-WARP v3 has been implemented which provides a basic embedded Linux system for complete communication network stack on FPGA. It is a dual-core implementation which runs an OS on one processor and puts MAC processing load on other. This system enables implementation and testing wireless communication algorithms. We have implemented TDMA based multiple access system (as potential application) so the LE-WARP v3 nodes imitate real-world BSs and UEs. It is also possible to use DCF MAC and integrate this system with 802.11 compliant nodes.

Acknowledgment

The authors would like to thank Wireless Network Lab facility at the Centre for Wireless Communications (CWC) for funding this project.

References

[1] M. Matinmikko, M. Palola, H. Saarnisaari, M. Heikkila, J. Prokkola, T. Kippola, T. Hanninen, M. Jokinen, and S. Yrjola, "Cognitive radio trial environment: First live authorized shared access-based spectrum-sharing demonstration," *Vehicular Technology Magazine, IEEE*, vol. 8, no. 3, pp. 30–37, Sept 2013.

[2] "Amendment of the commissions rules with regard to commercial operations in the 3550- 3650 mhz band," Federal Communications Commission (FCC), Tech. Rep., April 2015, uRL: https://www.fcc.gov/document/citizens-broadband-radio-service-ro.

[3] 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

[4] J. Niemela, M. Jokinen, and T. Hanninen, "Linux enriched design in second generation wireless open-access research platform," in *9th International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM)*, June 2014, pp. 311–316.

[5] "WARP Project, (accessed 21.6.2015)," uRL: http://warpproject.org/.

[6] "Mango Communications 802.11 Reference Design, (accessed 30.6.2015)," uRL: http://mangocomm.com/802.11.

[7] H. Tuomivaara, M. Raustia, and M. Jokinen, "Demonstration of distributed tdma mac protocol implementation with olsr on linux enriched warp," in *Proceedings of the 4th ACM International Workshop on Experimental Evaluation and Characterization*, ser. WINTECH '09. New York, NY, USA: ACM, 2009, pp. 85–86. [Online]. Available: http://doi.acm.org/10.1145/1614293.1614312

[8] M. Jokinen, M. Mäkeläinen, and T. Hänninen, "Demo: Co-primary spectrum sharing with inter-operator d2d trial," in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '14. New York, NY, USA: ACM, 2014, pp. 291–294. [Online]. Available: http://doi.acm.org/10.1145/2639108.2641749

[9] S. Namal, I. Ahmad, S. Saud, M. Jokinen, and A. Gurtov, "Implementation of openflow based cognitive radio network architecture: Sdn&r," *Wireless Networks*, pp. 1–15, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11276-015-0973-5

[10] "Ettus Research USRP Product Line, (accessed 02.09.2015)," uRL: http://www.ettus.com/product/.

[11] J. Zhang, J. Jia, Q. Zhang, and E. Lo, "Implementation and evaluation of cooperative communication schemes in software-defined radio testbed," in *INFOCOM, 2010 Proceedings IEEE*, March 2010, pp. 1–9.

[12] Z. Yan, Z. Ma, H. Cao, G. Li, and W. Wang, "Spectrum sensing, access and coexistence testbed for cognitive radio using usrp," in *4th IEEE International Conference on Circuits and Systems for Communications (ICCSC)*., May 2008, pp. 270–274.

[13] "Xilinx: All Programmable, (accessed 30.6.2015)," uRL: http://www.xilinx.com.

[14] "Xilinx Wiki, (accessed 30.6.2015)," uRL: http://www.wiki.xilinx.com/.

[15] "Buildroot: making Embedded Linux easy, (accessed 05.07.2015)," uRL: http://buildroot.uclibc.org.

[16] "Embedded GLIBC, (accessed 05.07.2015)," uRL: http://www.eglibc.org/home.

[17] "Sd specifications, part 1, physical layer, simplified specification," SD Association, Tech. Rep., January 2013, uRL: https://www.sdcard.org/downloads/pls/simplified_specs/part1_410.pdf.

[18] *Appendix C of M68000 Family Programmer's Reference Manual*, Motorola Inc., 1992, uRL: http://www.freescale.com/files/archives/doc/ref_manual/M68000PRM.pdf.

[19] "iperf2 Web Site, (accessed 30.6.2015)," uRL: http://sourceforge.net/projects/iperf2/.