Interfacing USRP Kit With Zynq-7000 Evaluation Kit

Khadija Khaled[1], Chaymaa Osama[1], Mahetab Osama[1], Heba Magdy[1], Heba Mahmoud[1], Yara Hossam[1], Sherif Hosny[2], and Hassan Mostafa[1,3]

[1]Electronics and Communications Engineering Department, Cairo University, Egypt.
[2]Mentor Graphics.
[3]University of Science and technology, Nanotechnology and Nanoelectronics Program, Zewail City of Science and Technology, October Gardens, 6th of October, Giza 12578, Egypt.
{*khadija.k4a@gmail.com, chaymaa.ossama@gmail.com, mahetabo17@gmail.com, Heba.m.elgohary@gmail.com, heba.m.yassin@gmail.com, yarahossam7196@gmail.com, sherif_hosny@mentor.com, hmostafa@uwaterloo.ca*}

***Abstract*—Digital Front End Reconfiguration is a promising techniques to deploy the concept of Software Defined Radio (SDR). SDR is a radio communication system whose components are implemented by means of software using Universal Software Radio Peripheral (USRP) and GNU Radio. SDR not only provides cheap and flexible multi-standard-terminals for end users, but also offers a physical layer platform where functions are software defined. Dynamic Partial Reconfiguration (DPR) has proved itself in implementing SDR system on Field Programmable Gate Arrays (FPGAs). This work implements the physical layer of Wi-Fi transceiver chain on Xilinx Zynq board supporting DPR flow. Data is modulated via real communication channel using the USRP RF section connected to the FPGA. A new approach is proposed to enable running the DPR flow and USRP application simultaneously on the same FPGA. Bit Error Rate (BER) calculations are evaluated versus the distance for testing the system integrity.**

***Keywords*—Software Defined Radio, Universal Software Radio Peripheral, Dynamic Partial Reconfiguration, and Field Programmable Gate Array.**

I. Introduction

Over the past decade, modern wireless communication systems witnessed a new era of high data rates. The number of users of mobile devices has increased significantly. Different standards have been implemented not only to achieve high data rates, but also to compromise between area, power, quality and the large number of users. Each standard has its own transceiver which results in large area utilization and power consumption affecting the battery lifetime. The SDR is a proposed solution to tackle these challenges [1].

The SDR is a way to implement the physical layer of wireless communication standards using software which results in decreasing the utilized area. Different technologies can be used to implement the SDR such as Digital Signal Processors (DSPs) and FPGAs. The FPGA is suitable for high rate applications due to its short design cycle and low power consumption. The high flexibility of the FPGA achieved by Dynamic Partial Reconfiguration (DPR) makes it the best candidate for implementing SDR transceivers [2], [3].

USRP has become a popular platform for hardware based research in the field of SDR and cognitive radio (CR) [4]. The recent released versions of USRPs currently offer a scalable, simpler, and easier tools to use combined platforms. Integrating the GNU Radio software with the USRP enables the user to build a complete open software radio system that supports host-based signal processing on any platform.

This paper is organized as follows: Section II shows a list of related work. Section III gives details about the overall system implementation. Section IV illustrates the hardware implementation of the Wi-Fi transceiver followed by the way of interfacing the USRP with the FPGA in Section V. Simulation results are listed in Section VI. Ultimately, Section VII shows the paper conclusion and future work.

II. Background and Previous Work

The proposed approach mentioned in [5] provides an overview on the procedures of establishing a TCP/IP link between two NI-USRP-29xx kits. The internal FPGA implemented inside the USRP kit is used to communicate with the PC for implementing extra functionalities.

The authors in [6] used the Texas Instruments Beagleboard OMAP3 as a host for low-cost SDR platform implemented through the GNU Radio. Altera Cyclone III FPGA is used to perform signal processing activities such as filtering, down-sampling, digital down-conversion on the transmitted/received data from the RF section. Communication between the FPGA and the Beagleboard is performed using SPI interface.

The survey applied in [7] on different radio transport protocols shows that they can be used to build complex processing systems and potentially decrease development time for heterogeneous systems. The survey shows that routing architecture integrated in third-generation USRPs will increase performance in SDRs.

The authors in [8] use the USRPB200 kit as a host for implementing Wide Band Frequency Modulation (WBFM) mobile phone transceiver. The software model for each block in the design is implemented using GNU Radio. An audio file is transmitted and received via real channel using the USRP RF section. The design mentioned in [9] is focused on implementing the Wi-Fi 802.11a and 802.15.4 ZigBee

978-1-7281-1184-1/19/$31.00 ©2019 IEEE

transceiver as a soft model using GNU Radio for evaluating the performance of wireless networks.

The contribution of this work is implementing the physical layer of Wi-Fi transceiver chain on an external Zynq board supporting DPR technology. The RF section in the USRP kit is used for sending and receiving the modulated data via real communication channel. A synchronization approach is being deployed between each FPGA core to perform certain independent operations. Communication between the FPGAs and the USRPs is performed using GNU Radio. In order to test the system robustness, BER calculations are evaluated versus the distance.

III. SDR System Overview

The USRP Software Defined Radio Device is a tunable transceiver for prototyping wireless communication systems. It offers frequency ranges up to 6 GHz with instantaneous bandwidth up to 56 MHz. The USRP Hardware Driver (UHD) is connected to USB 3.0 whose data rate is up to 5Gbps. UHD is a hardware driver library serving all types USRPs and daughter-boards by providing unique APIs for software radio implementation. The driver can be standalone, or with third party applications such as: GNU Radio, Labview, and Simulink [10].

Figure 1 shows the overall system implementation. The connection between the USRP kit and Zynq FPGA is through USB cable. The GNU Radio provides a soft model in the form of flow graph that enables configuring the USRP easily [11]. Compiling the flow graph generates a python code that is executed from the Linux image through UART terminal.

The Linux image is created by using the Yocto project using Xilinx tools. The two main components of the project along with the open embedded project are: BitBake, the build engine, and Open Embedded-Core, the basic layer containing common functions and packages for all builds. The Linux image is customized to support the Zynq board and USRP by adding new layers offered by Xilinx and Yocto community. Theses Layers are:

- **Meta-Xilinx**: which provides packages and information for the Zynq board.
- **Meta-Xilinx-Tools**: which supports Xilinx tools on the Zynq board.
- **Meta-SDR**: which includes GNU Radio and UHD for the USRP.

The whole design flow is performed using Xilinx Vivado. The process invoked simulating the RTL design of the Wi-Fi transceiver, synthesis, placing, routing, and eventually bit stream file generation to program each FPGA. After building and configuring the Linux image, Xilinx SDK manages the following tasks:

- Configuring the PL side with the bitstream files generated by Xilinx Vivado. This is performed using a C-application that calls reserved APIs for configuring the PL side and the Partial Reconfiguration Controller PRC used to control the ICAP to support the DPR flow [12].
- Creating the bootable image from the Linux boot loader generated by the Yocto project.
- Propagating the test data through the Wi-Fi transceiver to generate the modulated symbols. Data is transferred from the DDR to the PL side through DMAs.
- Executing the python application generated by the GNU Radio to configure the USRP.

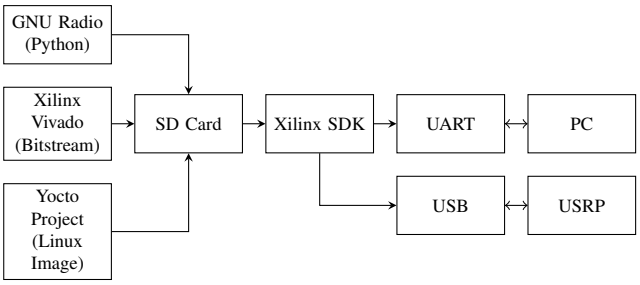


Fig. 1: System peripherals

The python application generated by the GNU Radio transmits the modulated symbols from the implemented Wi-Fi transmitter on the Zynq board to the USRP RF section as illustrated in Figure 2. The RF section uses Gaussian Minimum Shift Keying (GMSK) technique to transmit the data on high frequency range to the receiving antenna. The GNU Radio at the receiver side regenerates the modulated symbols and transmits them to the implemented Wi-Fi receiver on the other Zynq board. Ultimately, a comparison is being performed between the received demodulated data and the original reference model for error checking.

Distortion in the transmitted files is dependent on the distance between the antennas of the two USRPs. As the distance increases, the probability of losing packets increases. The maximum distance where the two USRPs can communicate is defined by the gain and bandwidth. Channel estimation techniques are used to repeal the effect of distortion and bit error rate.

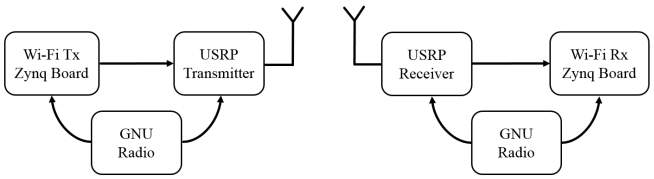


Fig. 2: Tranceiver block diagram

IV. WI-FI Tranceiver Design

The Wi-Fi 802.11a [13] is used as a case study for the proposed system. Figure 3 shows all implemented blocks in Wi-Fi chain. The transmitter includes all blocks from scrambler to preamble. The rest are the receiver chain. Scrambler is responsible for randomizing the MAC layer data in order to prevent the presence of long 1s or 0s sequences. Convolutional encoder with coding rate equals to 1/2 is used for data replication in order to decrease the bit error rate (BER) and enable the decoder to deduce the correct transmitted bits. Since both blocks uses generator polynomials, shift registers with XOR gates are used for hardware implementation.

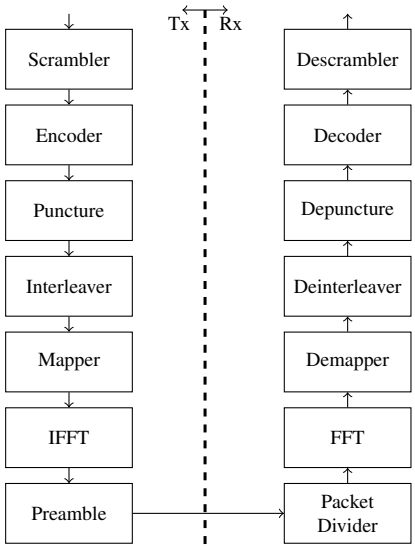


Fig. 3: Wi-Fi chain block diagram [13]

Puncturing technique of coding rate 3/4 is used to stealing specific encoded bits to increase the coding rate. Interleaver is used to get rid of burst errors by rearranging the data bits. Implementation of these blocks is performed using memory controllers for easily storing and reading the data.

Symbols are then modulated using 16-QAM modulation scheme. Since Wi-Fi is Orthogonal Frequency Division Multiplexing (OFDM) based, the 128-point IFFT is used to modulate the symbols on several sub-carriers instead of single carrier in order to reduce the channel fading. The cyclic prefix is added to eliminate the Inter Symbol Interference (ISI). Ultimately, the preamble is used to add short and long headers for synchronization purpose.

The packet divider receives the modulated symbols and removes the reserved preamble bits from the stored data. The demapper specifies the decision region of the received real and imaginary symbols from the FFT, then converts the symbols to a stream of bits.

Deinterleaving process is used to repeal the effect of the interleaver at the transmitter side. The depuncture pads dummy bits in the position of the removed punctured bits. Viterbi decoder is used because of its ability for error detection and correction. Descrambler uses the same equation used by the scrambler in the transmitter.

Figure 4 shows the established test environment on the PL side. The test data stored in the SD card is transferred to the DDR within the run-time. The data flow in the PL side is conducted using the following steps:

1) the input data is transferred from the DDR memory to be stored in the Direct Memory Access (DMA) that adjusts the rate according to the clock of each wireless communication system.
2) an intermediate block “Input Interface” is used to adjust the data input rate and system reset.
3) data is transferred through the system.
4) another intermediate block “Output Interface” is used to adjust output data rate for the DMA.
5) the output data is stored in a second DMA to be finally transferred to the DDR for verification.

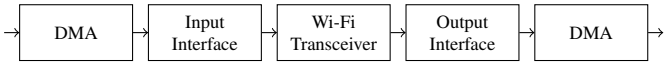


Fig. 4: Test environment on the PL side

V. USRP And FPGA Interfacing

Xilinx Zynq-7000 kit provides two Cortex-A9 processors that share common memory and other peripherals. Asymmetric multiprocessing (AMP) mechanism allows both processors to concurrently run independent OS or bare-metal applications allowing each one to communicate with the other through a shared memory called On-Chip Memory (OCM). OCM provides very high performance and low latency compared to the DDR.

As illustrated in Figure 5, the Linux image launched on CPU0, the system master is responsible for the following:

- Initializing the system.
- Controlling CPU1 startup.
- Communicating with CPU1 through the OCM.
- Running USRP applications using the USRP Hardware Driver (UHD) and GNU Radio installed on the file system.
- Interacting with the user through the UART terminal. The user has the ability to choose to configure the system or testing it.

Meanwhile, the bare-metal application running on CPU1 generated by compiling the C-application that uses the DPR APIs is responsible for:

- loading the bitstream files to configure the PL side.
- managing the test data for Wi-Fi tranceiver. This is performed by propagating the data from the SD card to the DDR then to the PL side through the DMA.
- communicating with CPU0 through OCM.

This technique manages to run the SDK bare metal and USRP python applications on the same Zynq board. Synchronization between the two processors is being deployed in order to prevent conflicts on shared hardware resources such as: OCM, global timer, and L2 cache.

The Linux image is configured as symmetric multiprocessing (SMP) with a single CPU in order to operate in the AMP configuration The device tree is modified to reduce the allocated memory for Linux image in order to provide some space for CPU1 to run its applications.

This technique manages to run the SDK bare metal and USRP python applications on the same Zynq board. Synchronization between the two processors is being deployed in order to prevent conflicts on shared Hardware resources such as: OCM, global timer, and L2 cache.

The AMP system is divided into three software sections:

1) Xilinx native First stage boot loader (FSBL).
2) The Linux OS and USRP operating on CPU0.
3) The Bare-metal OS and SDK operating on CPU1.

According to Figure 6, the following procedures are being performed:

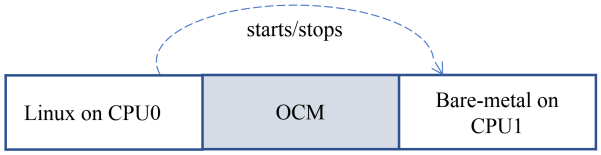


Fig. 5: AMP system configuration

- **System Initialization**: Initially, after power on, the FSBL is launched on CPU0 to reset the PS system and program the PL side using the bitstream file. The FSBL is also responsible for loading the ELF application from the DDR and executing it.
- **Linux Image Loading**: The Linux image is loaded on CPU0 which starts CPU1.
- **Modulated Symbols Generation**: The bare-metal on CPU1 loads the ELF file on the SD card to pass the Wi-Fi test data to the chain and generate the modulation symbols.
- **Configuring USRPs**: The python application is launched on the Linux image through the UART terminal to configure the USRP kit.
- **Sending Data To RF Section**: Eventually, The USRP sends the modulated data through the RF section.

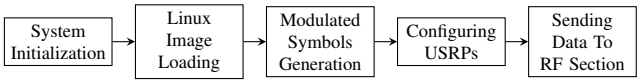
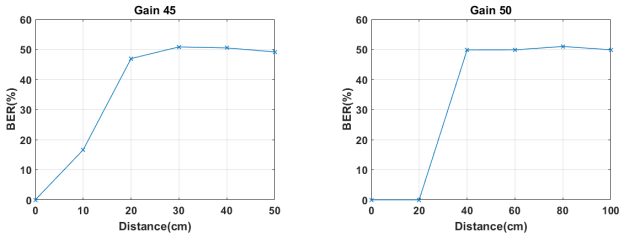


Fig. 6: System initialization procedures

VI. Simulation Results

Testing the system is performed by sending data through the implemented Wi-Fi 802.11a transceiver. Figure 7 shows the BER calculations versus distance between the two USRPs. As the gain increases the BER decreases. The BER percentage increases proportionally with the distance until reaching nearly 50% at 50 cm. Implementing the channel estimation RTL block will enable the receiver to correct the received data frames which will result in decreasing the BER percentage.



(a) BER for gain 45 (b) BER for gain 50

Fig. 7: BER percentage Vs Distance

VII. Conclusion And Future Work

Interfacing the Zynq board with the USRP kit, configured using GNU Radio, offers a complete solution for SDR implementation. The AMP technique is used to run bare-metal applications and load the Linux image on the same Zynq board. The Wi-Fi 802.11a physical layer implementation is performed on the board FPGA as a case study. Data is modulated through the air between the two USRPs and received with reasonable BER.

Connecting the Zynq board with the USRP via Ethernet cable instead of the USB to increase the data rate is a proposed future work. Reducing the BER can be achieved through implementing the channel estimation RTL blocks.

VIII. Acknowledgement

This work is partially funded by ONE Lab at Zewail City of Science and Technology and at Cairo University, NTRA, ITIDA, ASRT, and NSERC.

References

[1] G. Sklivanitis, A. Gannon, S. N. Batalama, and D. A. Pados, "Addressing next generation wireless challenges with commercial software-defined radio platform," in IEEE Communications Magazine, Vol. 54, No. 1, pp. 59-67, Jan 2016.

[2] S. Hosny, E. Elnader, M. Gamal, A. Hussien, A. H. Khalil, and H. Mostafa, "A Software Defined Radio Transceiver Based on Dynamic Partial Reconfiguration," *IEEE New Generation of Circuits and Systems (NGCAS 2018)*, Valletta, Malta, pp. 158-161, Nov. 2018.

[3] A. K. ELdin, S. Hosny, K. Mohamed, M. Gamal, A. Hussein, E. Elnader, A. Shalash, A. M. Obeid, Y. Ismail, and H. Mostafa, "A Reconfigurable Hardware Platform Implementation for Software Defined Radio using Dynamic Partial Reconfiguration on Xilinx Zynq FPGA," *IEEE International Midwest Symposium on Circuits and Systems (MWSCAS 2017)*, Boston, MA, USA, pp. 1540-1543, Aug. 2017.

[4] O. N. Samijayani, P. Gitomojati, D. Astharini, S. Rahmatia, and N. I. H. Pratama, "Implementation of SDR for video transmission using GNU radio and USRP B200," in International Conference on Cyber and IT Service Management (CITSM), Denpasar, Indonesia, pp. 1-4, Aug. 2017.

[5] B. A. Vithalapara1, V. D. Parmar2, S. Agrawal3, and S. P. Singh, "Experimental Study of USRP Radio Transport VITA Communication Protocol," in International Journal of Emerging Technology and Advanced Engineering, Vol. 5, No. 4, pp. 361-366, April 2015.

[6] C. Anderson, G. Schaertl, and P. Balister, "A Low-Cost Embedded SDR Solution for Prototyping and Experimentation," in Software Defined Radio Technical Conf.Digest of Papers, Washington, DC, Dec. 2009.

[7] J. Malsbury and M. Ettus, "Simplifying FPGA design with a novel network-on-chip architecture," in Proceedings of the second workshop on Software radio implementation forum, pp. 45-52, Aug. 2013.

[8] D. Kushnure, M. Jiniyawala, S. Molawade, and S. Patil, "Implementation of FM Transceiver using Software Defined Radio (SDR)," in IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), Beijing, China, Oct. 2015.

[9] I. Chao, K. B. Lee, R. Candell, F. Proctor, C. C. Shen, and S.Lin, "Software-defined Radio Based Measurement Platform for Wireless Networks," in The International journal of Engineering development and research (IJEDR), Vol. 5, No. 2, Aug. 2018.

[10] M. Fhnle, "Software-Defined Radio with GNU Radio and USRP/2 Hardware Frontend: Setup and FM/GSM Applications," Thesis in Hochschule Ulm University of Applied Sciences Institute of Communication Technology, Ulm, Germany, Oct. 2009.

[11] C. Y. Chen, F. Tseng, K. Chang, H. C. Chao, and J. L. Chen, "Reconfigurable Software Defined Radio and Its Applications," in Tamkang Journal of Science and Engineering, Vol. 13, No. 1, pp. 29-38, Feb. 2010.

[12] A. K. ELdin, A. Mohamed, A. Nagy, Y. Gamal, A. Shalash, Y. Ismail, and H. Mostafa, "Design Guidelines for the High-Speed Dynamic Partial Reconfiguration Based Software Defined Radio Implementations on Xilinx Zynq FPGA," *IEEE International Symposium on Circuits and Systems (ISCAS 2017)*, Baltimore, USA, pp. 1-4, May 2017.

[13] IEEE Computer Society, "IEEE Std 802.11-2012," March 2012.