Architectures for Cognitive Radio Testbeds and Demonstrators – An Overview

Oscar Gustafsson*, Kiarash Amiri§, Dennis Andersson¶, Anton Blad*, Christian Bonnet‖, Joseph R. Cavallaro§, Jeroen Declerck‡, Antoine Dejonghe‡, Patrik Eliardsson¶, Miguel Glasse‡, Aawatif Hayar‖, Lieven Hollevoet‡, Chris Hunter§, Madhura Joshi†, Florian Kaltenberger‖, Raymond Knopp‖, Khanh Le†, Zoran Miljanic†, Patrick Murphy§, Frederik Naessens‡, Navid Nikaein‖, Dominique Nussbaum‖, Renaud Pacalet‖**, Praveen Raghavan‡, Ashutosh Sabharwal§, Onkar Sarode†, Predrag Spasojevic†, Yang Sun§, Hugo M. Tullberg¶, Tom Vander Aa‡, Liesbet Van der Perre‡, Michelle Wetterwald‖, and Michael Wu§

*Department of Electrical Engineering, Linköping University, Sweden. Email: oscarg@isy.liu.se

†WINLAB (Wireless Information Network Laboratory), Rutgers University, USA

‡IMEC, Belgium

§Center for Multimedia Communication, Rice University, USA

¶Swedish Defence Research Agency (FOI), Linköping, Sweden

‖Mobile Communications Department, Eurecom, Sophia Antipolis, France

**System on Chip Laboratory, Telecom ParisTech, Sophia Antipolis, France

***Abstract*—Wireless communication standards are developed at an ever-increasing rate of pace, and significant amounts of effort is put into research for new communication methods and concepts. On the physical layer, such topics include MIMO, cooperative communication, and error control coding, whereas research on the medium access layer includes link control, network topology, and cognitive radio. At the same time, implementations are moving from traditional fixed hardware architectures towards software, allowing more efficient development. Today, field-programmable gate arrays (FPGAs) and regular desktop computers are fast enough to handle complete baseband processing chains, and there are several platforms, both open-source and commercial, providing such solutions. The aims of this paper is to give an overview of five of the available platforms and their characteristics, and compare the features and performance measures of the different systems.**

I. Introduction

Two of the trends within future wireless communication to allow higher data rates and lower probabilities of outage are more aggressive use of the spectrum as well as nodes cooperating in transmission, allowing relaying of data. One approach for increased spectrum utilization is cognitive radio [1], [2], where the nodes can adapt their spectrum usage to what is currently available, even if these unused frequencies are within licensed bands. The cooperative approaches are based around nodes that allow relaying of data. While this approach has been known for several decades it has had a recent resurgence [3].

For cognitive radio communication techniques to get closer to actual implementation and in the end deployment, an important step is to demonstrate it using a fully working over-the-air implementation. Furthermore, a testbed can significantly speed up simulation and evaluation. By using existing architectures the development time is reduced. However, there exist several different architectures that a potential research entity would like to consider. The objective of this work is to provide an overview of five of these architectures. Three are already openly available, namely USRP/GNU Radio, WARP, and OpenAirInterface, while the other two, WiNC2R and COBRA, are currently in internal use only. We aim at providing an overview of these architectures and for the reader to be able to compare pros and cons in a simple way.

II. USRP/USRP2 and GNU Radio

GNU Radio and the Universal Software Radio Peripheral (USRP) are the software and hardware parts, respectively, of a complete low-cost SDR platform that has gained widespread use [4]–[7]. The USRP is a product developed by Ettus Research, and follows a basic concept with a motherboard containing ADC/DAC, an FPGA performing sampling rate conversion, host interface, and plug-in daughterboards containing frequency-specific RF front-ends. USRP2, shown in Fig. 1, was developed to improve on the limited communication bandwidth of USRP1. In addition, USRP2 provides improvements in ADC/DAC resolution, an SD card for FPGA image and firmware, a MIMO connector and synchronization between units. The USRP 2 connects to the host PC using Gigabit Ethernet where USRP1 uses USB 2.0.

The USRP1 has slots for two TX and two RX daughterboards with synchronization capabilities allowing 2×2 MIMO applications. In contrast, the USRP2 has slots for only one TX and one RX daughterboard, but allows MIMO with up to 8 antennas using external synchronization with other USRP2 units. The available daughterboards are compatible with both of the USRP versions. Schematics and source code for the FPGA firmware are provided for both units, and both schematics and PCB layouts are provided for the daughterboards, allowing users to easily adjust the hardware

Digital Object Identifier: 10.4108/ICST.CROWNCOM2010.9290
http://dx.doi.org/10.4108/ICST.CROWNCOM2010.9290

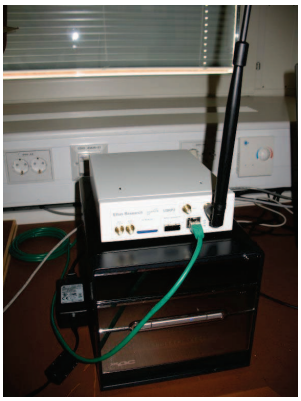

Fig. 1. The USRP2 perched on top of its host computer.

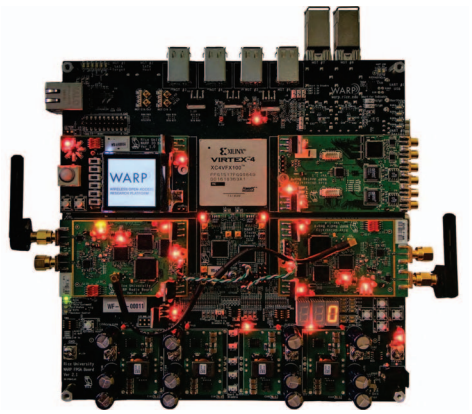

Fig. 2. WARP board with radio daughtercards.

to specific needs. Performance measurements for one of the more widely used daughter boards are provided in [8].

All signal processing is done using the open source GNU Radio software on the host PC. GNU Radio is a hybrid software package using signal processing primitive blocks implemented in C++, and applications defined as flow graphs. Performance can be ensured by a low-level implementation of computations, whereas applications can be simplified by the high-level development in Python.

One of the main drawbacks of the platform is that the data processing latency is severe. The delays are mainly imposed by the flow graph block structure by which the software is designed, which leads to large buffers between each block. In addition, the host interface also imposes delays, especially for the USRP1, which can not easily be overcome. Typical turn-around time from reception to transmission can reach several hundreds of microseconds. However, a stand-alone version of the USRP2 featuring a larger FPGA, where all the signal processing is done in the hardware, may solve the problem, although at an increased application development complexity.

III. WARP: Wireless Open-Access Research Platform

Rice University's WARP is a scalable, extensible and programmable wireless platform, built from the ground up, to prototype wireless networks. The platform architecture consists of four key components: custom hardware, platform support packages, open-access repository and research applications; all together providing a reconfigurable wireless testbed for students and faculty. Figure 2 shows the WARP board along with radio daughtercards.

A Xilinx Virtex-4 FPGA is the primary communication processor on the main board. The PowerPC processors embedded in the FPGA provide a complete embedded programming environment for MAC and network layer design [11], [13]. The dedicated multi-gigabit transceivers (MGTs) provide high speed board-to-board connections which make the WARP platform scalable and extendable.

For physical layer design, the platform supports different levels of design flows from low level VHDL/Verilog RTL coding to system level MATLAB modeling. Xilinx System Generator is one of the system-level modeling tools integrated in MATLAB that provides abstractions for building and debugging high-performance DSP systems in MATLAB/Simulink using the Xilinx Blockset. Moreover, the WARP board supports Simulink "hardware co-simulation" that expedites the simulation and debugging steps.

For MAC and network layer design, the WARP platform supports C based applications on the PowerPC while interfacing the physical layer implementations in the FPGA fabric. The Xilinx Platform Studio tool is an integrated programming environment that is used to control both the physical layer and MAC layer implementations as shown in Figure 3.

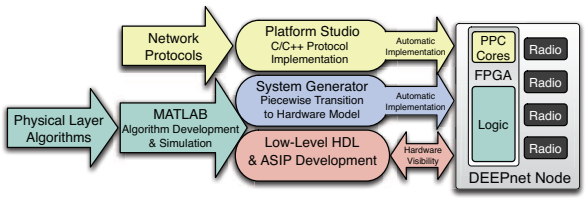

Fig. 3. WARP system tools and design flow.

The Xilinx FPGAs deployed in WARP boards provide significant processing resources to implement and test complicated physical (PHY) layers. Currently, two different PHY have been implemented and fully verified in over-the-air tests. The single-input single-output (SISO) Orthogonal Frequency Division Multiplexing (OFDM) transceiver uses the FPGA for all the baseband processing. The up-conversion to the RF band is carried out using one radio daughtercard [10] in each WARP node. Furthermore, a 2×2 MIMO OFDM transceiver, i.e. two daughtercard radio boards for each WARP node, has been designed for the WARP hardware.

Digital Object Identifier: 10.4108/ICST.CROWNCOM2010.9290
http://dx.doi.org/10.4108/ICST.CROWNCOM2010.9290

In addition to the full real-time physical layer, a second design flow, called WARPLab, can be used for system exploration, as in the case of cooperative communication algorithms [12]. WARPLab allows rapid prototyping of physical layer algorithms over the air, by exposing WARP hardware to MATLAB.

IV. OpenAirInterface.org

OpenAirInterface.org provides open-source hardware and software solutions for experimental radio network experimentation. The platform is developed by Eurecom. The activity consists of two hardware platforms, one targetting air interface experimentation which is mostly software and PC-oriented for maximum flexibility [14]. The second is FPGA-based and aims at innovation in system-on-chip architectures [15] for multi-modal baseband subsystems.

The OpenAirInterface software-based platform currently aligns its air-interface development with the evolving LTE standard but provides extensions for mesh networking, particularly in the MAC and Layer 3 protocol stack. It can be seen as a mock standard which retains the salient features of a real radio system, without all the required mechanisms one would find in a standard used in deployment of commercial networks. Deployment of tens of nodes with two-way real-time communication in both cellular and mesh topologies has been demonstrated in the context of several collaborative projects. The aim is to study techniques such as multi-cell cooperative techniques, distributed synchronization, interference coordination and cancellation.

OpenAirInterface features an open-source software modem written in C comprising physical and link layer functionalities for cellular and mesh network topologies. This software modem can be used either for extensive computer simulations using different channel models or it can be used for real-time operation with the available hardware. In the latter case, it is run under the control of the real-time application interface (RTAI) which is an extension of the Linux operating system. A picture of the software-based platform (CBMIMO1) is shown in Fig. 4. It has a native dual RF chain (2×2 MIMO or multiple-frequency) with TDD multiplexing comprising 5 MHz channels at 1900 MHz.

The SoC experimentation platform (ExpressMIMO) is flexible enough to use the same baseband processing resources for multiple standards. The control is in the software part of the design, which passes the relevant paremeters to hardware for specific functionalities. The challenge in the design is to synchronize all the processing at air-interface in an efficient manner with minimum resource utilization and high accuracy. The identified operations are implemented as seven independent processing blocks, and can be called as hardware accelerators:

- Pre-processor
- Frontend processor
- Mapper
- Detector
- Channel encoder

(a) (b)

Fig. 4. (a) CBMIMO1 card and (b) CBMIMO1 card with PC.

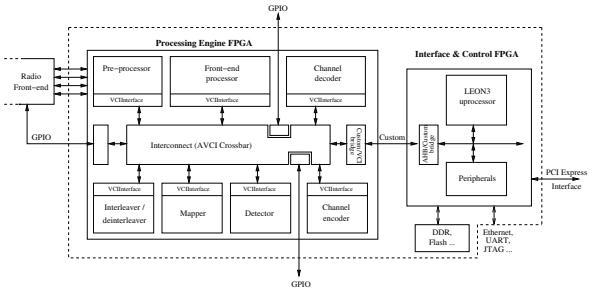

Fig. 5. Architecture of ExpressMIMO.

- Channel decoder
- Interleaver/de-interleaver

The pre-processor block is used as an interface with the external A/D and D/A converters (I/Q multiplexing, control signaling). It also provides several basic signal processing functionalities like filtering, sample rate adjustment, carrier frequency adjustment. The mapper and the detector implement all the modulation schemes ranging from BPSK to QAM256. The (de)interleaver block, apart from (de)interleaving the data streams with all options in the different standards, performs the frame equalization and rate matching operations. The frontend processor provides the digital signal processing operations at the air-interface, like channel estimation, data detection, carrier phase offset (CPO) estimation etc. The channel encoder implements convolutional encoding, block cyclic codes and m-sequences. The channel decoder IP block realizes trellis-based decoding algorithms; Viterbi and Turbo, to decode convolutional and turbo codes, respectively.

The proposed hardware architecture is subdivided in two main parts: a high level control module and a digital signal processing engine. They are implemented in high end Xilinx Virtex- V FPGAs. Figure 5 depicts the architectural overview of the system.

a) The control module is based on a SPARC CPU (LEON3 from Gaisler Research) surrounded by its usual pe-

Digital Object Identifier: 10.4108/ICST.CROWNCOM2010.9290
http://dx.doi.org/10.4108/ICST.CROWNCOM2010.9290

ripherals, external memories (DRAM, Flash), a PCI-Express/ExpressCard interface and a dedicated interface with the DSP engine. The control module is in charge of controlling the DSP engine, implements some low-demanding processing (PHY and MAC) and interfaces the system with the host PC through the PCI-Express / ExpressCard interface. Most of the MAC layer processing runs on the host PC while the DSP engine executes most of the PHY layer processing tasks.

b) The DSP engine is a collection of data processing IP blocks plugged on a crossbar interconnect. Each IP block is a highly configurable and parametrizable processing unit dedicated to one class of algorithms (Fourier transforms, channel coding, channel decoding, modulation/ demodulation, etc.) The chosen interface between the IP blocks and the interconnect is a 64 bits Advanced VCI interface. Each IP block embeds a DMA engine and an 8 bits microcontroller. The synchronization (inter or intra-blocks) is based on a set of interrupts signaling the end of memory transfers and of data processing. The control module programs the DSP engine by configuring the parameters and local software routines of the IP blocks.

The baseband architecture is separated into two FPGAs which can function as stand alone (i.e. without host PC). This helps to design and implement the architecture, and is also fruitful to test the design later on. The Interface and Control FPGA (control module) is responsible to transfer MAC requests to the Processing-Engine FPGA and control data direction flow. The Processing-Engine FPGA (DSP Engine) is responsible for all up-link/downlink signal processing.

The prototype card contains four high-speed dual data converters (dual A/D and dual D/A) which can be connected to an external RF. Eurecom can provide a 200MHz–8GHz flexible RF chain which interconnects with SoC system.

V. WiNC2R: The WINLAB Network Centric Cognitive Radio Hardware Platform

The architecture of the WiNC2R cognitive radio platform, developed by WINLAB at Rutgers University, is based on the recognition of the workload characteristic of multi-layer wireless communication protocol processing, which are quite different from the embedded computing applications, and even more so from the ones of the general purpose computing applications. To name a few, the context switching is very frequent (it is needed for every packet, or even for every packet processing unit), there is no spatial and temporal locality of data, workload size is small, processing time constraints are very stringent, and the data input and output is very intensive. Most importantly the processing flow is driven by the events rather than by the program counter as in the stored program paradigm. The events driving the flow can be time or data based, and both can be generated by the environment, or as the result of internal processing. Given these sharp differences of the SDR applications and the traditional application workload, we need a different architectural framework to address both functional requirements and workload characteristics of programmable radio applications. The WiNC2R architecture framework addresses the workload characteristics of wireless communication protocols with programmable control mechanisms that engage both hardware and software modules in a uniform manner in order to satisfy both functional and performance requirements.

We observe that processing consists of a number of functional modules operating and generating the events consisting of signals and data. The run time control has to respond rapidly to the event by detecting and decoding it and activating the processing function in charge of handling it. The sequencing of events and their causal relationships need to be maintained by the control mechanisms. Given the short processing time for the packet, or parts of it, the fast context switching needs to be supported by both software function execution entities (CPUs), as well as hardware functional modules, otherwise the utilization of the units will be low. For the systems targeting hundreds of megabits per second data throughput the unit processing time is sub-microsecond, dictating the use of hardware assisted and controlled context switching.

The Virtual Flow Pipelining (VFP) [16] architecture approach is designed to satisfy the workload characteristics and functional requirements of radio communication protocol processing applications. The following are the key characteristics of the VFP framework:

a) System level control structure implemented in hardware will support the protocol processing flow requirements: function synchronization, scheduling, performance guaranties, sequencing, and communication;
b) The set of functional units will be comprised of generic hardware modules and software programmable processors. Both will be controlled by the system level Virtual Flow Pipeline controller;
c) Function switching will be fast, on a per packet basis, regardless whether it is performed in hardware or software;
d) Hardware based central processing units scheduling will enable fast context switching and, hence, obviate the need for an embedded operating system.

A fully hardware based solution lacks flexibility for the future changes, and quite often generalization to effectively accommodate multiple protocols. On the other hand such solutions do accommodate the processing flow functional and performance requirements. Our Virtual Flow Pipeline architecture adds the required flexibility to the hardware pipelining approach, while retaining performance guaranties. Figure 6 illustrates the VFP processing flow for the OFDM transmitter implemented in a first version of WiNC2R platform.

The experimental test setup at WINLAB with two back-to-back WiNC2R boards is shown in Fig. 7.

VI. COBRA: COgnitive Baseband RAdio

One of the key requirements towards a truly cognitive radio is that the digital and analog parts of the radio are flexible. In other words the radio has to be software defined. IMEC's first generation baseband radio platform called BEAR [18] was developed to be software defined and support different

Digital Object Identifier: 10.4108/ICST.CROWNCOM2010.9290
http://dx.doi.org/10.4108/ICST.CROWNCOM2010.9290

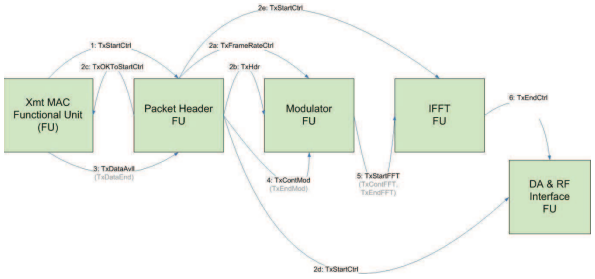

Fig. 6. VFP processing flow for the OFDM transmitter.

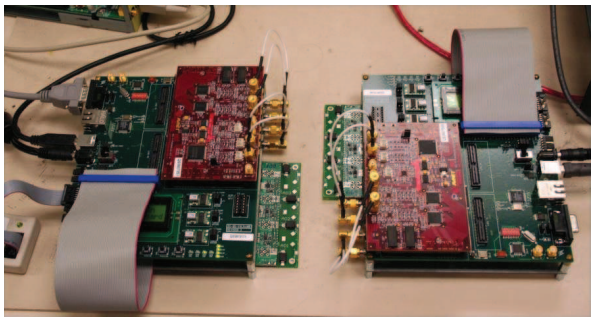

Fig. 7. Two-node WiNC2R setup used for prototyping.

standards including cellular (LTE), connectivity (WLAN) and also broadcasting (different DVB standards). An instance of this platform was fabricated and is shown in Fig. 8. In such software platform, there is a different flexibility need in the digital front end, inner modem and the outer-modem across different standards.

The second generation IMEC's platform COBRA is a much more advanced platform template. One instance of a COBRA platform is shown in Fig. 9. The COBRA platform can be customized to not only handle very high data rates, but also low throughputs in a scalable way. This platform largely consists of four types of cores:

a) DIFFS: A digital front end capable of sensing and synchronization;
b) ADRES [18]: A reconfigurable core capable of doing inner modem processing;
c) FlexFEC [19]: A flexible forward error correction capable of doing different outer modem processing and;

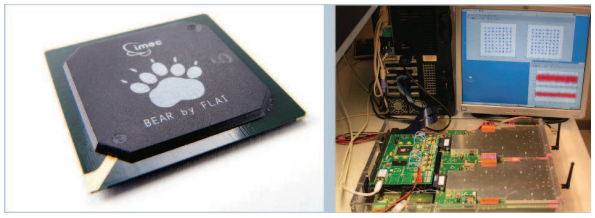

Fig. 8. IMEC's first generation BEAR platform.

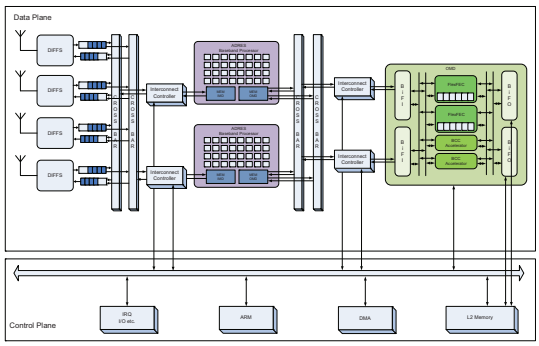

Fig. 9. IMEC's second generation COBRA platform.

d) ARM core for controlling the tasks on the platform.

All cores of this digital platform can be customized based on the requirements and the target standards that need to be supported. Also all parts of the platform are programmable in C or high level assembly.

VII. Comparison Overview

To provide an easier comparison between the different platforms, we have summarized some key facts in Table I.

VIII. Conclusion

In this work we have provided an overview of five different platform suitable for testbeds and demonstrators in the area of cognitive and cooperative communication. This overview should be seen as an aid for selecting a platform for proving current and future algorithms in an over-the-air setting.

References

[1] J. Mitola III and G. Q. Maguire, Jr., "Cognitive radio: making software radios more personal," *IEEE Personal Comm. Mag.*, vol. 6, no. 4, pp. 13–18, Aug. 1999.
[2] S. Haykin, "Cognitive Radio: Brain-empowered Wireless Communications," *IEEE J. Sel. Areas Comm.*, vol. 23, no. 2, pp. 201–220, Feb. 2005.
[3] A. Nosratinia, T. E. Hunter, and A. Hedayat, "Cooperative communication in wireless networks," *IEEE Comm. Mag.*, vol. 42, no. 10, pp. 74–80, Oct. 2004.
[4] P. Eliardsson and D. Andersson, "A modular cognitive radio testbed architecture for dynamic spectrum access," FOI Memo 3040, FOI, Linköping, Sweden, 2009.
[5] T. R. Newman and T. Bose, "A cognitive radio network testbed for wireless communication and signal processing education," in *Proc. DSP Workshop and Signal Processing Education Workshop*, Marco Island, FL, USA, Jan. 2009, pp. 757–761.
[6] Z. Yan, Z. Ma, H. Cao, G. Li, and W. Wang, "Spectrum sensing, access and coexistence testbed for cognitive radio using USRP," in *Proc. IEEE Int. Conf. Circuits Syst. Comm.*, 2008, pp. 270–274. [
[7] R. Dhar, G. George, A. Malani, and P. Steenkiste, "Supporting integrated MAC and PHY software development for the USRP SDR," in *Proc. IEEE Workshop SDR*, 2006, pp. 68–77.
[8] P. Eliardsson, U. Uppman, A. Johansson, and H. Tullberg, "Properties of USRP2 in Communication Applications," https://hig.se/download/18.51c1f7d3123eb47ebf5800015043/1-1+Hugo+Tullberg_Patrik+Eliardsson+slides.pdf

Digital Object Identifier: 10.4108/ICST.CROWNCOM2010.9290
http://dx.doi.org/10.4108/ICST.CROWNCOM2010.9290

TABLE I
FEATURES OF THE DIFFERENT CONSIDERED PLATFORMS FOR COGNITIVE RADIO TESTBEDS AND DEMONSTRATORS.

Platform	**GNU Radio/USRP**	**WARP**	**OpenAirInterface.org**	**WiNC2R**	**COBRA**
Website	http://gnuradio.org/ http://www.ettus.com/	http://warp.rice.edu/	http://www. openairinterface.org/	http://www.winlab. rutgers.edu/	http://www.imec.be/
Hardware platform	FPGA with USB (USRP1) or ethernet (USRP2) with (up to) 2 RF interfaces	FPGA with ethernet, memory, I/O, with (up to) 4 RF interfaces	CPU/FPGA/RF		CPUs and ASICs
License	GPL	BSD	GPL/Cecill	N/A	White-box IP
Sample rate	Input: 64 MSps, 12 bits (USRP1), 100 MSps, 14bits (USRP2), Output: 128 MSps, 14 bits (USRP1), 400 MSps, 16 bits (USRP2)	14-bit I/Q at 40MSps typical; supports up to 65MSps	7.68 MSps (PF1 = CB-MIMO1), 61.44 MSps (PF2 = ExpressMIMO)	Input: 125 MSps, 14 bits Output: 500 MSps	Up to 65 MSps, 9.5 bits
RF Bandwidth	8 Mhz (USRP1), 25 MHz (USRP2)	40MHz per RF interface; interfaces are tuned independently	5 MHz (PF1), 20 MHz (PF2)	Up to 40 MHz	200 kHz–40 MHz
Frequency bands (MHz ranges)	Receive: 0.1–300, DC–30, 50–860, 800–2400, Transmit: 0.1–200, DC–30, Transceive: 50–2200, 400–500, 800–1000, 1150–1400, 1500–2100, 2250–2900, 2400–2500/4900–5850	2400–2500, 4900–5875	1900 (PF1), 200–8000 (PF2)	2400–2500, 4900–5300, 5400–5875	100–5000
MIMO support	2×2 (USRP1), 8×8 (USRP2)	4×4	2×2 (PF1), 4×4 (PF2)	No	4×4
Programmable hardware	Xilinx Spartan 3-2000 FPGA (USRP2, mainly LUTs left)	Xilinx Virtex-4 FX100 FPGA: Example full 2x2 MIMO OFDM + CSMA MAC design leaves about 50%	20% Virtex 2 3000 (PF1), 50% of Virtex 5 LX330, 50% of Virtex 5 LX110T (PF2)	Yes, Virtual Flow Pipelining API	Yes
Approximate cost	$700 (USRP1), $1400 (USRP2), $75–450 (RF boards)	$6500 (academic)	2500 euros (PF1), 8000 euros, 10000 with RF (PF2)	N/A	N/A
Programming languague	Python/C++ (GNU Radio)	Real-time designs: FPGA for PHY, C for MAC, WARPLab designs: MATLAB	C	C/C++; Setting up control tables	C and assembly
Requirement on host computer	- (USRP2 can be stand alone)	No host required; FPGA can implement full wireless design	PC with native-Intel CPU	Can be stand alone	None (stand alone)
Interface to host computer	USB 2.0 (USRP), Gigabit Ethernet (USRP2)	Gigabit Ethernet, when host PC is used	PCI, PCI Express, ExpressCard (PF1), PCI Express (PF2)	PCI Express	N/A
Max. data transfer rate		FPGA can saturate gigabit Ethernet	30.72 Mbytes/s RX, 15.36 Mbytes/s TX (PF1)	10 Mbps	up to 1 Gbps
Public common code base	GIT/wiki	SVN/wiki	SVN/wiki	Non-public SVN	No

[9] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, and P. Steenkiste, "Enabling MAC Protocol Implementations on Software-Defined-Radios," in *Proc. USENIX Symp. Networked Systems Design Implementation*, 2009.

[10] P. Murphy, A. Sabharwal, and B. Aazhang, "Design of WARP: A flexible wireless open-access research platform," in *Proc. EUSIPCO*, 2006.

[11] P. Murphy, C. Hunter, and A. Sabharwal, "Design of a cooperative OFDM transceiver," in *Proc. Asilomar Conf. Comp. Signals Syst.*, 2009.

[12] K. Amiri, M. Wu, M. Duarte, and J. R. Cavallaro, "Physical layer algorithm and hardware verification of MIMO relays using cooperative partial detection," in *Proc. IEEE ICASSP*, 2010.

[13] C. Hunter, J. Camp, P. Murphy, A. Sabharwal, and C. Dick, "A flexible framework for wireless medium access protocols," in *Proc. Asilomar Conf. Comp. Signals Syst.*, 2006.

[14] F. Kaltenberger, R. Ghaffar, R. Knopp, H. Anouar, and C. Bonnet, "Design and Implementation of a Single-frequency Mesh Network using OpenAirInterface," *EURASIP Journal on Wireless Communications and Networking*, Article ID 719523, 16 pages, 2010.

[15] N. Muhammad, R. Rasheed, R. Pacalet, R. Knopp, and K. Khalfallah, "Flexible baseband architectures for future wireless systems," in *Proc. Euromicro Conf. Digital System Design Arch. Methods Tools*, Parma, Italy, Sept. 3–5, 2008, pp. 39–46.

[16] Z. Miljanic, I. Seskar, K. Le and D. Raychaudhuri, "The WINLAB Network Centric Cognitive Radio Hardware Platform – WiNC2R," *Mobile Networks and Applications*, vol. 13, no. 5, Oct. 2008.

[17] Z. Miljanic and P. Spasojevic, "Resource Virtualization with Programmable Radio Processing Platform," in *Proc. WICON*, 2008.

[18] V. Derudder et al "A 200Mbps+ 2.14nJ/b digital baseband multi processor system-on-chip for SDRs," *Proceedings of VLSI*, June 2009.

[19] F. Naessens et al, "A 10.37 mm^2 675 mW reconfigurable LDPC and Turbo encoder and decoder for 802.11n, 802.16e and 3GPP-LTE," *Proceedings of VLSI*, June 2010.

Digital Object Identifier: 10.4108/ICST.CROWNCOM2010.9290
http://dx.doi.org/10.4108/ICST.CROWNCOM2010.9290