A Mobile Programmable Radio Substrate for Any-layer Measurement and Experimentation

Kuang-Ching Wang

Department of Electrical and Computer Engineering

Clemson University, Clemson, SC 29634

kwang@clemson.edu

Version 1.0

September 27 2010

This work is sponsored by NSF grant CNS-0940805.

A Mobile Programmable Radio Substrate for Any-layer Measurement and Experimentation

A Whitepaper Developed for GENI Subcontract #1740

Executive Summary

Wireless networks have become increasingly pervasive in all aspects of the modern life. Higher capacity, ubiquitous coverage, and robust adaptive operation have been the key objectives for wireless communications and networking research. Latest researches have proposed a myriad of solutions for different layers of the protocol stack to enhance wireless network performance; these innovations, however, are hard to be validated together to study their combined benefits and implications due to the lack of a platform that can easily incorporate new protocols from any protocol layers to operate with the rest of the protocol stack.

GENI's mission is to create a highly programmable testbed for studying the future Internet. This whitepaper analyzes GENI's potentials and requirements to support any-layer programmable wireless network experiments and measurements. To date, there is a clear divide between the research methodology for the lower layers, i.e., the physical (PHY) and link layers, and that for the higher layers, i.e., from link layer above. For lower layer research, software defined radio (SDR) based on PCs and/or FPGAs has been the technology of choice for programmable testbeds. For higher layer research, PCs equipped with a range of commercial-off-the-shelf (COTS) network interfaces and standard operating systems generally suffice. The whitepaper studies the requirements needed to combine the two forms of testbeds into one while assuring seamless transition for the researchers – researchers will be more willing to conduct experiments on such a new platform if the learning threshold is sufficiently low.

To date, the key obstacles for SDR technologies include the sampling rate and resolution of the analog-digital conversion (ADC) which limits the signal bandwidth and dynamic range that can be received. The PC-to-SDR interface speed is another key bottleneck, since most of the SDR plugs into a host PC that runs the user applications. The use of an FPGA-based SDR is a certain requirement for experimenting with state-of-the-art commercial standards such as Wi-Fi or WiMAX at their full capacity (of course, even more so for technologies beyond the state of the art). FPGA's cell capacity dictates the complexity of lower layer protocols that can be supported, the additional line-rate measurement logic that can be programmed, and the user-friendliness for experimenter codes since they are usually not optimized.

It is relatively easy to experiment higher layer protocols on testbeds based on COTS radios under realistic scenarios with a range of environment and mobility conditions. This has not been the case for PHY testbeds which have typically been expensive to build with few prototypes available. One of the greatest values GENI can bring to the lower layer researchers is to help overcome the scale bottleneck by creating a shared use SDR testbed with a scale and setup sufficient to conduct realistic network experiments. The ready integration of SDR implementations with a standard yet programmable suite of higher layer protocols and the support for a range of meaningful scenarios will be top-priority tasks.

The GENI Cognitive Radio being developed has been a promising candidate to realize this goal with its extensible multi-FPGA system, high bandwidth PC-SDR interface, and the Cognitive Radio Kit (CRKit) programming environment. Nevertheless, more programming and experimentation support will be essential to realize its full potential.

1. Introduction

Future wireless communication devices must achieve high bandwidth, robust adaptive operation, and efficient spectrum usage to meet the exploding demands of wireless communication in a diverse range of conditions. Novel solutions are being studied in different protocol layers; in most cases, solutions for different protocol layers are studied and evaluated separately. Evaluation of the combined performance of solutions across all layers has been very costly and difficult, if not impossible. Traditionally, due to the very different knowledge needed for lower layer protocols, i.e., physical (PHY) and link layers, and higher layer protocols, i.e., from link layer above, they have been studied by different research communities using different analysis, simulation, and experimentation tools.

When it comes to prototype implementation, lower and higher layer protocols also have very different processing requirements. Lower layer protocols process, transmit, and receive physical waveforms and thus require fast processing and high precision analog circuits; higher layer protocols, on the other hand, process binary data frames at comparatively much lower rates. Prototypes for the lower layer protocols have often been implemented with custom circuitries that are highly optimized and less programmable; nevertheless, in recent years, FPGA-based platforms have been adopted for programmability. Due to the expense and efforts needed to build such platforms, they are usually made in small quantities and validated with one or few radio links.

On the contrary, prototypes for higher layer protocols can mostly be implemented on general-purpose computers with standard commercial off-the-shelf (COTS) wireless interfaces. Apparently of lower costs, these prototypes have been built in large quantities for studying complex network configurations involving large numbers of radio links, e.g., ORBIT [1] and many others [2-23].

To enable joint experimentation of solutions from both realms, integration is needed for not only *the platform*, but also *the people* and *the experiment methodologies*. A fully programmable platform must be based on a general-purpose computer (for higher layer protocols), FPGA (for lower layer protocols), and arbitrary waveform radio front ends. Protocols must be implemented, executed, and analyzed by researchers from both realms. This is possible only if the experiment methodologies can conform to the practice in both realms. This essentially creates a *loose federation among researchers* by contributing their efforts to develop mutually compatible modules for a common platform, such that they can expect to draw from each other the protocol pieces that make up a complete system for meaningful evaluation. In this way, lower layer researchers benefit from the opportunity to experiment with networks at practical scales with realistic network traffic, while higher layer researchers benefit from the incorporation of novel pre-standard wireless technologies. To further enhance cost effectiveness of realizing a sufficiently large network of such platforms, it makes even more sense to enable remote shared access to a large pool of these devices through a central control framework. This is exactly the National Science Foundation's vision for creating the Global Environment for Network Innovations (GENI) testbed, through which such programmable radio devices will be controlled for all layer experimentation.

This whitepaper follows one potential experimental workflow for a researcher to remotely access such a testbed for protocol development, deployment, and execution of their experiments to identify the testbed's required features. The whitepaper also reviews literature on experimental research in wireless communication and networking, so that the testbed can be designed with features meeting these needs.

The remaining of the whitepaper is organized as follows. Section 2 presents a survey of representative wireless testbeds in place today to identify their design goals and challenges. Section 3 gives an overview of the GENI Cognitive Radio hardware and software components. Section 4 analyzes the requirements and procedure for integrating custom any-layer code with the GENI Cognitive Radio and, at the same time, point out issues that have not been adequately addressed. A case study is presented based on the scenario of integrating student code from a coding theory class at Clemson University. Section 5 discusses the useful experiment supports for any-layer wireless experiments. The whitepaper concludes in Section 6.

2. Survey of Existing Wireless Testbeds

The survey was not meant to be comprehensive. Nevertheless, a representative set of 45 wireless testbeds were examined in four groups to identify their experimentation objectives, design features and challenges, and programming tools.

2.1 Experimentation Objectives

According to their focused protocol layers, the surveyed testbeds can be categorized into four categories: channel emulation testbeds [24-27], physical layer testbeds [28-42], COTS radio based higher layer testbeds [2-23], and full stack programmable testbeds [43-45]. A few testbeds such as TWINE [46] and Emulab [6] developed frameworks to also incorporate software emulated entities. The GENI Cognitive Radio, as introduced in Section 3, will be a full stack programmable testbed with a control framework supporting remote shared access over Internet.

2.1.1 Channel Emulation Testbeds

Channel emulation testbeds are designed to programmatically inject radio channel impairments to the RF signals sent between *radios under test* in a controlled environment, so that the impairments due to a variety of channel conditions expected in different realistic scenarios can be tested without uncontrolled disturbances and interferences. Channel emulation can be achieved by 1) recreating a close replica of the physical environment of interest in the testbed, such as that pursued by the Illinois Wireless Wind Tunnel project [24]; 2) connecting radios' antenna ports with shielded RF cables and analog signal conditioning elements (amplifier, attenuator, combiner, divider) [25]; 3) FPGA boards that digitally apply arbitrary channel effects between radio antenna ports, such as the CMU Emulator [26] and the JHUAPL ACTION testbed's Amplitude, Phase, and Frequency Control module [27].

Channel emulation testbeds address a critical aspect for wireless experimentation that is otherwise very hard to achieve, i.e., repeatability. The received signal of any radio sensitively depends on the surrounding environment – movement of any object (even very small ones), radiated signal from any source, and changing thermal noise can all result in significant differences in experiment results. Channel emulation testbeds isolate the radio from such variable conditions to the most extent and apply only controlled effects to the tested radio signal, such that reproducible experiments and valid comparison among different algorithms under same conditions are achievable. Hence, the objectives for such testbeds focus on repeatability of experiments, while their utility depends on the range and accuracies of producible channel effects.

To emulate any channel effect, the effect must be first recorded with separate channel sounding equipments. The recorded effects are then extracted into channel models that can be emulated. The two-step approach unavoidably loses a certain extent of accuracy in the radio test results.

2.1.2 Physical Layer Testbeds

Physical layer testbeds for MIMO [31-41], MIMO/OFDM [28, 30, 42], and cellular standards such as WCDMA [47], GSM [29] and LTE [31] are surveyed. Most of these testbeds are custom made for their specific target technology using a mixture of personal computers (PC), general purpose processors, DSP, FPGA, integrated circuit chips, signal generators, and RF attenuators, while several recent efforts have developed platforms for flexible reprogramming of a wider range of custom physical and link layer protocols [43, 45, 48-50]; among them, the Ettus USRP/USRP2 radio [49], the Rice University WARP radio [45], and the Lyrtech small form factor SDR [50], are now commercially available.

Physical layer testbeds implement the transmit and receive chains of a communication system. These testbeds' primary objectives are for validation of custom transmit and receive algorithms and measurement of signal and link level performance. Some attempts have been made to integrate such testbeds with protocols above the link layer [43, 44], though all are still in working progress. For example, since the Hydra radio testbed [44] is based on USRPs plugged into host PCs, it can be integrated with higher layer protocols and applications running on the PC. The end-to-end application experimentation capability with Hydra has been verified with a point-to-point link, while its current key limitation is the PC processing time for more complex PHY algorithms. The delay and buffer size for data transfer on the PC-USRP interface also presents an important limit on the turn-around time of PHY and link protocols implemented on PC; hence, the consensus has been to implement as much of these functions as possible on the radio's FPGA. However, USRP and the more recent USRP2 both have limited free space on their FPGAs (Altera Cyclone EP1C12Q240C8 [51] for USRP and Xilinx Spartan 3-2000 [52] for USRP2) to allow additional custom logic for this purpose [53].

The bandwidth between the PC and the USRPs also presents a limiting factor for the achievable signal bandwidth. USRP employs USB 2.0 (480 Mbps) for this interface, allowing it to send/receive a total of 16 MHz symbol rate (equivalent to 4 MHz baseband, half-duplex, complex I-Q signal); USRP2 employs 1 Gbps Ethernet to raise this limitation to 50 MHz symbol rate (12.5 MHz baseband, half-duplex, complex I-Q signal). It is useful to contrast this limit with the baseband signal bandwidth for various popular commercial standards: IEEE 802.11 a/b/g requires 20 MHz, IEEE 802.11n requires 20 or 40 MHz, and WiMAX requires 1.25~20 MHz. It is apparent that to experiment such standards and advancement beyond them require substantial down-scaling of their bandwidth with the current USRP boards. Moving as much of the PHY and link algorithm implementation to the FPGA can overcome the bandwidth and turn-around time limitations over the PC-USRP configuration, provided the implementation can fit in the FPGA onboard USRP.

The KU Agile Radio [43], on the other hand, adopts a fully embedded solution to implement the radio protocols on a Xilinx Virtex II FPGA (much higher cell counts than those for USRP and USRP2) and a PowerPC digital signal processor (DSP). The KU radio currently supports full-duplex transmit and receive of up to 30 MHz baseband signal [54]. Since the DSP runs Linux, the standard TCP/IP stack and existing research prototypes of higher layer protocols and applications can potentially be integrated for experimentation. This, however, has not been reported in the KU radio developers' publications.

2.1.3 COTS Radio Based Higher Layer Testbeds

COTS radio testbeds are the most proliferate to date. They are relatively *less costly* to build than other types of testbeds, *programmable* in the network layer and above (with a subset of tunable link layer parameters exported by COTS radios), and suitable for experimenting a wide range of protocols and applications. Most COTS radios, e.g., those based on IEEE 802.11 (Wi-Fi), IEEE 802.15 (Bluetooth), IEEE 802.15.4 (Zigbee), IEEE 802.16 (WiMAX), allow tuning only a subset of radio parameters such as the transmit power, channel, and modulation schemes; the rest of the physical and link layer protocols are inaccessible. Since these radios are mostly developed for use with general purpose computers, existing and new higher layer protocols and applications programmed to run on PCs can be quickly integrated for experimenting with the radios.

COTS radio testbeds have been used for experiments in laboratories and actual deployments for purposes from validating network properties (e.g., connectivity, link quality, and achievable throughput), monitoring network usage pattern (e.g., [12]), to protocol and application performance evaluation (link, network, transport, application, and cross-layer protocols).

2.1.4 Full Stack Programmable Testbeds

Despite the abundance of COTS radio testbeds, it is easy to notice their common limitation in incorporating advanced solutions at and below the link layer, which are crucial for a range of emerging applications. For example, recent studies of vehicle networks with COTS radio testbeds [55, 56] have identified severe link quality fluctuation and multipath effects as the main causes for throughput degradation. Prior theoretical physical and link layer research have suggested methods to mitigate such effects, such as using MIMO/OFDM to improve link quality and tolerate multipath effects, it is logistically expensive, if not impossible, to implement these methods on the same network to validate their potential strengths and limitations. The few that did set out to attempt the design of a solution for comprehensive system testing have resorted to the use of FPGA-based radios integrated with general purpose computers [43-45]. Clearly, their objectives have been focused on validation of comprehensive solutions with flexibly programmable components for each layer.

Key features of [43] and [44] have been discussed in Section 2.1.2. Rice University's Wireless open-Access Research Platform (WARP) [45] also adopts a fully embedded architecture, utilizing a FPGA board (Xilinx Virtex 4) and a variable number of RF daughterboards. Each RF board can accommodate up to 40 MHz wide signal, and multiple RF boards can be used at the same time for MIMO research. The WARP radio FPGA board also features a gigabit Ethernet interface and a few multi-gigabit interfaces that can be used to interface with external devices (e.g., PCs). WARP does not have a general processor onboard; hence, full stack experiments will require external PCs running higher layer protocols and applications to be tethered to the radio via the Ethernet interface.

2.2 Programmable Testbed Features and Challenges

The utility of any testbed lies in its *range of programmability* for studying different configurations, protocols and in its *modeling accuracy* for realistic conditions.

2.2.1 Channel Emulation Testbeds

Channel emulation testbeds condition the radio signal between the transmitting and receiving radios under test. According to the classification in Section 2.1.1, the first type (emulation facility) of such testbeds' programmability range is determined by the number of interesting scenarios that can be replicated in the controlled environment, and its accuracy depends on the proper scaling of radio power, injection of background noise and interference, and the geometry and mobility patterns of all modeled radios and obstacles. The Illinois Wireless Wind Tunnel [24] provides a good demonstration of this approach. It is expected that conducting experiments requires extensive investment and personnel support to design, create, and execute each specific scenario.

The second and third types (analog and digital signal conditioning) of such testbeds utilize, respectively, analog and digital radio conditioning elements, which define the range of programmable channel effects that can be accurately realized. The former can only model stationary channels (or snapshots of extracted mobile channels), while the latter can achieve the full range of possible time-variant channel effects. The usefulness of the digital channel emulation testbeds is determined in the available channel models for it to generate time-variant effects; acquiring channel models requires a *physical layer testbed* capable of capturing channel effects for each scenario under real conditions. The most commonly encountered channel models for narrowband mobile cellular communications are well known and easily available for emulation [57], while emerging multi-component multi-carrier (e.g., MIMO/OFDM) channels and urban vehicular network channels remain to be modeled for emulation.

2.2.2 Physical Layer Testbeds

Physical layer testbeds feature full implementation of the physical layer and, for most cases, its accompanying link layer protocols. Programmable physical layer testbeds are based either on software running on general purpose processors, FPGA, or a combination of both.

The software radio approach has been attractive for its low cost. For a moderate price (e.g., $2000 for an Ettus USRP board with a dual ISM band radio front end), researchers can develop their protocols on their PCs (desktop, laptop, or single-board PC) to conduct experiments. While the software complexity is virtually unlimited (as complex as the PC can handle), the achievable signal bandwidth is constrained by the PC-to-radio interface bandwidth and delay (see Section 2.1.2), and the signal fidelity (waveform error tolerance) is constrained by the transceiver board's analog-to-digital and digital-to-analog conversion (ADC/DAC) resolution.

The FPGA approach implements the bulk of the physical and link layers on the FPGA chip. In some cases an onboard DSP processor can be incorporated to share the processing load [43]; in such cases, decision on the placement of logic in FPGA or DSP must be carefully made to avoid delay and bandwidth bottlenecks posed by the DSP-FPGA interface bus.

While FPGA size is the utmost determinant on the complexity of algorithms it can afford, the availability of a repository of reusable protocol implementations is crucial to the wide adoption by a larger networking research community. For example, it will be utmost useful to have a reference open source implementation of the Wi-Fi and WiMAX PHY and link layers for research groups to expedite studies on improving or revolutionizing such techniques. Some efforts of building such repositories have been begun; e.g., the GNU radio archive [58] and the WARP repository [59].

Integration with higher layer protocols and applications has been envisioned [43-45] but not realized due to radio performance limits and the lack of an accessible programming platform.

2.2.3 COTS Radio Based Testbeds

COTS radio based testbeds have supported researches that require either none or limited radio control. An immense number of network performance measurement (connectivity, throughput, configuration comparison), protocol experimentation (routing, clustering, transport, distributed control, database/peer-to-peer data sharing, information processing), and field application (environmental monitoring, social networking, community Internet service) studies fall in this category. Most of them needed little or no visibility and programmability into the link layer beyond choosing the radio power and vendor provided modulation levels.

COTS radio testbeds have been used for studying network optimization for varied mobility, radio condition, traffic distribution, or energy efficiency by tuning radio and protocol parameters (power, modulation, scheduling parameters). The need to programmatically control these parameters has driven several open source radio driver projects such as MadWiFi [60] for Wi-Fi radios and TinyOS [61] for IEEE 802.15.4 radios. WINLAB's ORBIT testbed at Rutgers University is by far the largest COTS radio testbed, featuring 400 Wi-Fi radios and various other wireless radios (Bluetooth, Zigbee, SDR, and WiMAX). The recent GENI Open WiMAX base station project [62] will be the first of its kind that enables radio level programmability for WiMAX. In addition to their programmable features, their testbed management and control frameworks have provided valuable tools and experiences in managing large scale network resources, experiment execution, and data archiving.

2.2.4 Full Stack Programmable Testbeds

GNU radio, the WARP radio, the KU radio, and the GENI Cognitive Radio are four most recent hardware platforms that are provisioned to support full-stack programmability.

Several programming frameworks have been developed based on the GNU radio (with the Ettus USRP and radio boards). For example, Virginia Tech's OSSIE SDR software [63] is a GNU-radio based open source implementation of the US military's Software Communication Architecture (SCA). The OSSIE software provides a complete development suite for designing arbitrary and dynamic waveform applications (PHY and link layer protocols) based on a Linux based CORBA programming environment. Similar to Hydra [44], however, the framework has not integrated protocols above the link layer for experimentation. The source codes developed for GNU-radio based testbeds are mostly open source C codes under the GNU public license.

Contrary to the GNU radio testbeds, the WARP radios adopt a FPGA-based approach to develop all PHY and link layer protocols in FPGA. The WARP development environment is based on the Xilinx System Generator software that compiles researcher implementation with the required platform firmware to synthesize the FPGA bit stream for uploading. Most researcher codes are expected to be in VHDL, while it's possible to convert C source codes into equivalent VHDL for synthesis.

The KU radio, as reviewed in Section 2.1.2, combines the efficiency of FPGA/VHDL and flexibility of DSP/C design environment. KU radios, however, are currently not publicly available.

GENI's ORBIT Cognitive Radio Platform (OCRP) will be introduced separately in Section 3. Also a FPGA-centric platform, the OCRP radios adopt an innovative multi-FPGA configuration together with an integral compiler support to maximize parallel processing across multiple FPGAs to boost up processing speed of complex protocols. Full stack programmability is one of OCRP's envisioned objectives; the realization, however, still demands substantial development effort to enable flexible protocol integration and widely acceptable programming and experimentation supports.

The key challenges for realizing a new genre of full stack programmable testbed are beyond software integration. More importantly, such testbeds will attain their full utility only when people from across various traditionally separate fields of wireless communications and networking come together to jointly develop novel protocols in all layers and integrate them for experimentation. For example, researchers studying higher layer protocols and applications will benefit from experimenting with cutting edge radio protocols with superb performance, while researchers studying PHY and link layer protocols will benefit from an easily deployable network with real applications running on real network scales. Missing the participation from either side will limit the usefulness of such testbeds.

Frequency agility is another often overlooked challenge. The USRP, WARP, and GENI OCRP radios all feature wide-band (e.g., 50 MHz-2.2 GHz) radio transceivers. It is, however, very important to understand that the receivable signal frequency depends on the adopted antenna as well. Currently there is no commercial antenna that can receive signals at arbitrary frequencies across such a wide range. For experiments that focus on a specific narrow frequency band, choosing a matching antenna for the band will suffice. For experiments that wish to study agile frequency adaptation, new solutions must be developed to enable antenna frequency programmability.

2.3 Testbed Programming Methods and Tools

The testbed programming methods for the surveyed testbeds are summarized here to identify the different modes of programming support that researchers from different areas have been accustomed to.

2.3.1 Channel Emulation Testbeds

The CMU channel emulator is developed over the Emulab testbed control framework. Users first create an experiment by uploading an ns script at the testbed webpage to specify the number of nodes and the type of network to connect the nodes. Once the experiment is created, users can proceed to configure the channel model, node mobility, and test applications via xml scripts. Optionally, there are java utilities provided for network visualization and experiment dispatch.

2.3.2 Physical Layer Testbeds

Programming physical layer testbeds today typically requires C/C++ and/or hardware description languages (HDL). Matlab has been a popular environment/language as well for algorithm development and software verification. Matlab can be used for generating executable binaries for processors (e.g., PC or DSP) and FPGA; however, the former often suffers from substantial loss of efficiency, while the latter requires partnering FPGA vendor tools which require HDL codes as their default input for generating FPGA binaries. Python is also popular for linking C/C++ modules for running on processors. GNU radio and USRP based testbeds, embedded radio testbeds (e.g., KU radio [43], BYU MIMO radio [64]), and commercial SDR development boards (e.g., Lyrtech [50]) all require some or all of these programming methods.

For most PHY and link layer researchers, it remains a major hurdle to use HDL for developing advanced protocols (there are certainly those proficient of both, but they will be the relative few). The gap is not just due to less emphasis on HDL training; one major reason is the lack of open source HDL libraries for many standard functional blocks (e.g., coding/decoding algorithms). It is, on the other hand, relatively easy to find open source C/C++ libraries, or Matlab blocks that come with its digital signal processing and communications toolboxes.

It will be tremendously useful to enable flexible translation of C/C++ code into HDL. Various academic and commercial tools exist, though most of them have their respective constraints. Below is an example list of these tools and constraints:

- FpgaC is an open source initiative for an ANSI-C to HDL compiler.
- C-to-verilog.com is a commercial but free online tool. The tool requires uploading ANSI-C code to the website for HDL synthesis. The company can provide an off-line version for local installs on a case-by-case basis.
- All leading FPGA vendors provide their own C-to-hardware compilers. A few tools that are not platform specific are: Altium Designer (C/C++) [65], Nallatech (ANSI-C) [66], Impulse (ANSI-C) [67], Mentor (ANSI-C++, System-C) [68].
- There is a growing industrial trend to adopt C-like (but not C) languages for high level design and synthesis into FPGA. System-C and Handel-C are examples of two such languages. With reasonable effort, existing C code may be converted to these formats following certain guidelines.

While some of the above tools provide optimization features to accelerate the converted hardware performance, most of them expect the C source code has certain level of structural similarity to the anticipated FPGA implementation to result in efficient FPGA designs.

2.3.3 COTS Radio Testbeds

COTS radio testbeds are more often used to study protocols and applications above the link layer, and in some cases, the link layer protocol itself. Since COTS radio testbeds are based on PCs and standard operating systems (Linux, Windows, MAC, etc.), the full range of programming methods can be used.

In cases where the research interests lie inside the firmware (link layer protocols and configurations), open source firmware is needed. Madwifi [60] is one popular firmware for Linux platforms. Intel also releases the firmware source for its Wi-Fi chipsets to allow customization [69].

One important observation to be noted about COTS radio testbeds is their easy integration of any COTS radios to the platform without needing to modify any higher layer protocols and applications. The radio vendors will provide the drivers to install the radios as standard communication devices on the host machine, such that the protocols and applications can choose to access these radios through standard network configuration commands (ifconfig/ipconfig).

2.3.4 Full Stack Programmable Testbeds

Despite the potential of many existing testbeds to support full stack programmable experiments, none of them have been fully developed and published. This section discusses one possible future

programming environment that can most efficiently integrate the existing resources and encourage usage by researchers with different backgrounds to collaborate.

The key value of the full stack programmable testbed is not only its programmability but also the potential of resource sharing. For higher layer researchers to access advanced radio prototypes on this testbed, it is important that they see them as yet another communication device via a standard interface. For radio researchers to conduct experiments at network scale and with real application/protocol traffic, it is important that they come as a standard suite that can operate independent of the radios. For cross-layer researchers, they will expect the ability to access, revise, and recompile the source code of a selected subset of the radio, protocol, and application modules.

Attempting to come up with a whole new development environment will be a costly and time-consuming effort. The only way to lower the development threshold is to reuse as much as possible the existing resources. For example, if possible, it will be important to design the SDR hardware as a stand-alone radio that interfaces with a PC host as a standard communication device. This will not only allow all the higher layer protocols and applications be reused without any further efforts, but it will also simplify the design of the development and experiment control framework.

3. GENI Cognitive Radio: A Building Block for Any-layer Wireless Testbeds

The GENI Cognitive Radio, also known as the OCRP, is being developed at WINLAB, Rutgers University and University of Colorado, Boulder. The detailed specification can be referred to at [70].

3.1 OCRP Hardware

The OCRP radio hardware possesses a number of important design features that makes it a desirable building block for an any-layer programmable wireless testbed:

- Multi-FPGA platform with processor core: The project will leverage a system and network on chip builder (SNOC-builder) solution to program multiple FPGAs. The project proposes to specifically investigate tight time synchronization across the multiple FPGA cores. The gigabit interconnects among multiple FPGAs are critical for achieving accurate time synchronization and avoiding latency bottlenecks for real-time signal processing.
- High bandwidth PC-to-OCRP interface: Currently a gigabit interface is used for PC-to-OCRP interface. This is to be sufficient if the radio processing is kept onboard OCRP; thus, a host PC can treat an OCRP as a regular Ethernet device and direct higher layer packets to the Ethernet interface.
- Wide range of tunable carrier frequency: A number of radio front-ends can be used with OCRP. The widest tunable frequency range is from 100 MHz to 7.5 GHz. Note that proper antennas need to be selected for each radio range.
- Competitive ADC/DAC speed and resolution: ADC/DAC speed and resolution is a critical factor for waveform fidelity. While OCRP does not employ the most aggressive ADC/DAC available today, its top-of-the-line RF module's specification (250 MSps 12-bit ADC, 1 GSps 16-bit DAC) is competitive with other academic SDR platforms.

3.2 OCRP Software

The OCRP software has two main parts: the software for deploying VHDL designs on multiple FPGAs, and the software for experimental support (development, deployment, execution, and measurement).

The former is based on the SNOC-Builder tool, which in turn leverages various Xilinx tools. According to the GENI cognitive radio project proposal, currently the software assumes users to provide their own high level FPGA allocation schemes and the VHDL files for each FPGA to begin synthesis.

The latter is termed the Cognive Radio Kit (CR-Kit) [70]. The CR-Kit has just released its R3 architecture. The CR-Kit aims at providing a library of pre-programmed modules as building blocks for typical communication systems while allowing users to add their own customized blocks into the system. When completed, the CR-Kit will provide a number of "default blocks" that are standard to most users and need not be changed (unless a user chooses to), and it will provide an "application block" where users can plug in their own "radio applications". Note that "applications" in the radio context actually refers to the PHY and link layer protocols. The default blocks currently include the Ethernet interface to the host, the packet processor that handles outgoing/incoming packets from/to the host Ethernet interface before/after the "application block".

Like the SNOC Builder, the CR-Kit also assumes users to have the VHDL files for their custom radio application to be synthesized with the rest of the standard blocks for loading onto the OCRP FPGA. A trial use of the system was conducted at Clemson. A remote user at Clemson accessed two OCRP nodes at WINLAB with the following procedure (now available as a tutorial at [71]):

1. Install Xilinx ISE Design Suite with MATLAB Simulink plugin. The version of the Xilinx suite matters. For example, we experienced some command options that changed from v.10.1 to v.11.5, causing the OCRP partition script to have errors.
2. Install Mentor Graphics ModelSim.
3. Compile Xilinx libraries for ModelSim.
4. Download the OCRP source tree from http://svn.orbit-lab.org/.
5. Complete MATLAB path configuration.
6. Open OCRP model file in Simulink and use the Xilinx System Generator block to synthesize the net list and generate the bit file.
7. ModelSim will be called automatically after the previous step (co-simulation option in System Generator block). The ModelSim test waveform should be defined before this will work.
8. With the successfully generated FPGA bit file, it can be loaded onto the OCRP nodes for execution. According to the GENI Cognitive Radio team, the bit file loading will be integrated into the ORBIT management framework (OMF). It is under development now; hence, it was necessary to load the bit file through a remote desktop connection to the two Windows PCs that host the two OCRP nodes. In the remote desktop, Xilinx iMPACT tool is used to load the bit file. Execution begins automatically after loading.

3.3 Experiment Control Software

One of the planned goals for the GENI Cognitive Radio project is to integrate OCRP nodes with OMF. It is, however, not specified in the proposal the extent of integration that will take place. Through discussions with the project team, it was mentioned as desirable to integrate OCRP with Linux hosts (instead of Windows).

The latest R3 architecture for OCRP has defined the Ethernet packet format from the host [72]. The current definition can potentially allow two modes of packet handling:

1. OCRP as standard network device: In this mode, the host should send all packets as Ethernet frames with host derived source and destination MAC addresses, such that the packet is passed to RF after minor processing at the packet processor. It is necessary that OCRP implements a built-in ARP function. Currently the architecture shows a TCP/IP block inside the packet processor, which appears only needed for multiplexing control and data packets, since TCP and IP should already be running on the host.
2. OCRP as a stand-alone device without a host: In this case, it is expected that test applications, transport protocol (e.g., TCP), and network routing should be on implemented on OCRP. This, however, seems quite complex to enable. In the current architecture, no soft processing unit is exposed to the users (the proposal mentioned using the soft processing DSP units for internal scheduling and resource allocation).

The first mode seems more practical and flexible in reuse of higher layer protocols and applications. To complete the design, however, CR-Kit should provide an "Ethernet driver" to run on the host, such that higher layer protocols and applications will simply send and receive Ethernet packets via the driver exactly identical as on Ethernet, while the driver takes care of converting the standard Ethernet packet into the OCRP format as defined in the R3 architecture. Once this is done, the full stack integration can be considered complete.

4. Recommendations for Enabling Any-layer Programmable Experiments with OCRP

For researchers of diverse backgrounds to conduct any-layer programmable radio experiments, especially mobile scenarios, this section summarizes recommendations of key features with OCRP.

4.1 Any-layer Programmable Experiments: Definition and Requirements

We consider an any-layer programmable experiment being a network experiment that utilizes one or more programmable prototypes in any parts of the network protocol stack. To enable such experiments, the platform must facilitate loading, programming, and measuring metrics in any protocol layer(s).

While any platform that allows programming any layer protocols satisfies this definition, we are interested in only those that 1) require minimal effort to set up full stack experiments and 2) need changes only in the layers of interest to each specific researcher. Only so will researchers from different layers be attracted and justified to devote their efforts in developing research prototypes for the platform.

4.2 Maximal Application and Higher Layer Protocol Code Reuse

For PHY researchers to conduct a full stack experiment, they should ideally be able to take a number of "working images" of an operational network protocol stack and plug in their specialized PHY implementation. There should be a library of such images allowing users to specify, for example, their desirable application, transport, routing protocols, and network architecture. The images may also provide a set of default PHY and link layer protocols that the users can execute with no effort to verify the baseline performance of the network. Then, researchers can load their novel protocols to replace the default ones for comparative study. This approach not only allows the reuse of many existing well-known protocol modules (such as the base images provided in the ORBIT repository), but also provides a template to help new users to start using the platform and make only incremental changes to the testbed software, resulting in a least-resistance learning curve.

4.3 Maximal Physical and Link Layer Code and Development Environment Reuse

An existing and continuously growing resource of PHY and link layer implementations are now found in various academic repositories: GNU radio, WARP, OSSIE, and possibly others. It will be valuable to systematically integrate these resources to operate with the GENI OCRP nodes. The architectural similarity among these testbeds and OCRP makes it very possible to achieve full integration with relatively small interface supports. And, if C-toHDL converting tools can be made available to GENI, it will be even easier to port the existing software implementations onto the OCRP FPGA for major performance improvements.

For PHY researchers to study specific challenges in the PHY and link layer protocols, or for higher layer researchers to study the impacts of different lower layer protocols, it is useful to start with a working implementation and incorporate controlled changes into the protocols. Open source implementation of representative protocols, such as IEEE 802.x radio protocols, OFDM, or MIMO signaling schemes, will be ultimately useful. Such resources are readily available in the aforementioned open source SDR repositories. Establishing the utilities to link or convert these repositories to be compatible to OCRP will substantially enhance the GENI OCRP testbed's utilization.

These radio protocol repositories have been initiated by respective research projects. In addition to the open source resources they offer, these projects have also made efforts to develop graphical user interfaces for displaying the commonly useful protocol options, parameters, and performance metrics. It is recommended that GENI can incorporate the useful interfaces from these well established development environments but extend it with additional features that will ease the use by researchers from a wider range of diverse backgrounds.

4.4 Flexible Programming with Both FPGA and On-board Processor

As surveyed above, some existing testbeds feature FPGAs with on-board processors which allows Linux programming, e.g., [43]. OCRP's FPGA boards also feature on-board processors; on the other hand, the current architecture utilizes the processor for timing control and resource allocation without mentioning support for user programs. Enabling flexible programming and interfacing the two will have tremendous benefit in allowing fast prototyping by programming the processor while allowing performance optimization via FPGA programming.

4.5 Challenges and Recommendations for Experiment Monitoring and Debugging

Conducting wireless networking experiments faces a number of key challenges:

- RF monitoring for valid interpretation of measurement and debugging

  Wireless transmissions and protocol performances depend sensitively on the noise, interference, and signal strengths perceived by the nodes (transmitting and receiving nodes alike). The lower the protocol layer of interest is, the more accurate information of the radio environment is crucially required.

  RF spectrum analyzers have been used to date for experiment monitoring. For logging protocol-specific activities, commercial hardware and software packet sniffers have been developed for Wi-Fi networks. These functions can all be realized using an SDR node such as OCRP. However, monitoring operation must be carried out continuously and cannot be time shared with a node used for experimentation. Hence, it is recommended that dedicated experiment monitoring OCRP nodes be developed. It may also be recommended as a best practice experimental setup to always pair an experimental OCRP node with another collocated monitoring OCRP node.

  It is also noted that for debugging novel protocols, especially the lower layer ones, the timing and bandwidth requirements can be substantial. Since these protocols will most suitably be implemented in the FPGA, it is expected that data logging circuits will be useful for researchers. It is recommended that one or few standardized measurement blocks and conventions be developed and defined with the OCRP release, such that researchers can reuse a standardized measurement FPGA block that will all take care of the logging into onboard storage that the users can retrieve later with normal Linux commands.

  Of course, this does not preclude the need for researchers to log protocol specific data on the experimental nodes. Efficient logging requires substantial onboard resources (memory access and storage). It is recommended that the release of OCRP be accompanied with a number of measurement configuration and data archive guidelines and examples.

- Confirmation of radio configuration and surrounding setup

  It is foreseeable that circumstances exist when the experiments won't execute properly due to component failure or unexpected mis-configurations (e.g., wrong antenna attached or loose contact). While the software components can be easily verified remotely, physical components and surrounding conditions are harder to check. This is even more problematic if such disturbances occur during the experiment. In our own research experiences, when wireless measurements are conducted in an area not totally closed from moving parts (people, objects, doors), the resulting data could exhibit phenomena that are hard to deterministically validate without detailed recording of the environment.

  This is certainly a challenge that does not have one perfect solution. On potential recommendation is to install video cameras that can stream and record the experiment venue for a first order detection of any unexpected behavior. To verify the correct hardware configuration, such as the antenna, it is recommended to setup a standard logging system when the onsite staff

install and remove any parts of the hardware (for example, a mandatory sequence of webpage to enter the model number of the components on each device).

- On-site support for experiment setup and trouble-shooting:

  Unlike a wired network testbed that seldom needs to alter its physical and network topology, an important value of a wireless network testbed is the range of different scenarios that it can be tested on. It is possible to identify and maintain a number of scenarios using separate hardware at different sites to minimize the configuration overhead and chances for mis-configurations; it is, however, expected that such setups will require routine checkups to assure all software and hardware components are correct as specified and fully functional. On-site support is crucial, and it is recommended that such maintenance routines be standardized (fixed time, fixed list of check items, standard logs accessible by researchers) for efficiency and accuracy. Video or camera snapshots of up-to-date views of the devices may be useful as well.

4.6 Designing and Supporting Experiment Scenarios

The range of supported scenarios determines the core values of a wireless testbed. While there can be an infinite number of different scenarios possible, it is recommended that a representative set of scenarios be identified and maintained, such that most researcher needs are met.

Identifying useful scenarios can by itself be an open research question. It is possible to solicit community input to such scenarios (for example, calling for scenarios with justifications through GENI's website), and then identify the right site to realize it. Of course, the number and scale of scenarios require proportional investment on the hardware and personnel costs. Various means are possible to couple the creation of scenarios and the resources. For example, it is possible to solicit proposals through GENI spiral solicitations or other NSF programs for proposals describing one or few scenarios, justifying the focused wireless effects that can be supported, and offering to host a number of OCRP nodes for realizing the scenarios.

4.7 Education and Training

To utilize an any-layer programmable testbed effectively to advance research, a number of techniques deserve standardized training for the community or, even better, integration into the curriculum. The use of such as testbed is by itself transformative in research and education, as it naturally requires students and researchers to contemplate on wireless network problems with a holistic view. The learning curve for the setting can be, however, simplified if the platform is designed with clean separation of concerns – that is, a user should need to involve in "programming" only the protocols they choose to, and expect the other protocols above and below to be functional as given. Thus, the researchers will only need a minimal set of bootstrapping training sessions to learn the use of the testbed.

For example, the CR-Kit has planned to include a series of tutorials to introduce the OCRP architecture and the procedure for programming a radio module. To date, the tutorials are still under development and the full scope has not been set in stone. It is recommended that either the CR-Kit tutorials or a separate project can develop at least two such series of tutorials – one will be intended for the group of typical "higher layer researchers" who are more familiar with Linux network testbed operations but have less clues about FPGA, while the other will be intended for the group of typical

"lower layer researchers" who are more familiar with Matlab and C simulation environments but have limited knowledge about Linux and higher layer protocols and even FPGA programming. It is expected that FPGA programming may be the least familiar for most researchers in either group. It is recommended that such tutorials be based on a simple but reasonably complete example involving all protocol layers, such that the users can incrementally achieve the following learning goals:

- To set up the necessary remote system environment and launch the example experiment without modifying anything.
- To learn how to choose and configure protocols in different layers from a library of standard and contributed protocols.
- To learn how to modify, compile, and reload protocols in different layers and conduct new experiments.

It is understandable that it is impossible such tutorial covers the full basics of each programming languages and tools, but it is recommended that the narratives in the tutorial account for the diverse backgrounds of the researchers, i.e., provide at least sufficient details that explain the essential steps, and provide links to additional references or topical keywords when needed.

4.8 Software Licenses

Clemson has taken a trial exercise to install and configure the necessary software to identify the absolutely necessary procedure to enable remote use of the OCRP platform for experimentation. The exercise was discussed in Section 3.2. Through this exercise and discussion with the WINLAB team, it was found that the licensing issue was one of the top challenges to be addressed. Take the CR-Kit example, it is expected that:

1. If a remote user chooses to create/extend/modify the FPGA design, the user will need his/her own licensed Xilinx ISE software to develop and compile the FPGA design, and licensed ModelSim software for simulation validation.
2. For any user to use the CR-Kit's MATLAB development environment, the user must be licensed for MATLAB with Simulink.
3. For any user to load a compiled FPGA design onto OCRP using Xilinx tools, each user requires a Xilinx license.

While requirements 1 and 2 are clearly understood and easily achievable by most academic institutes to date through campus licensing, donation, or low-cost academic pricing purchases, the third requirement was not clearly understood so far. Based on the language in the Xilinx license terms, there is ambiguity in whether remote users leveraging a license held by the OCRP hosting site (e.g., WINLAB) to upload their compiled bit files to the OCRP FPGA is acceptable. The following lists three relevant statements from the Xilinx End User License Agreement [73]:

> 3.(1) Node-Locked (per-User) Seat. If Xilinx has issued to Licensee a (FLEXlm) node-locked Seat, then Licensee may allow the Software to be (a) installed on and accessed from only the specific machine(s) allowed by the applicable Authorization Codes, (b) used by only one User (at a time) for each one Seat for such Software that has been issued to Licensee by

Xilinx, and (c) used for the sole purposes of developing, synthesizing, testing and verifying designs only for Xilinx Devices.

3.(2) Floating (concurrent-User) Seat. If Xilinx has issued to Licensee a (FLEXlm) floating Seat, then Licensee may allow the Software to be (a) installed on and accessed from any number of machines, (b) used by up to the number of concurrent Users that is equal to the number of Seats for such Software that have been issued to Licensee by Xilinx as determined by the applicable Authorization Codes, and (c) used for the sole purposes of developing, synthesizing, testing and verifying designs only for Xilinx Devices.

4.(b) General Restrictions. Except only to the extent otherwise expressly allowed under Section 3 (License Grants) above (or under applicable laws notwithstanding these restrictions), Licensee is not licensed to, and agrees not to: … omitted … (vi) hypothecate, rent, lease, loan, lend, time-share, sublicense, distribute or otherwise transfer the Software to any other individual, corporation or other legal entity; … omitted …

Considering the current ORBIT control framework and other control frameworks (e.g., Emulab, Planetlab, etc.), it is expected that the FPGA programming and uploading will be done in three steps: 1) remote users program and compile their FPGA design using their own institute Xilinx software and license, 2) remote users upload their bit files to the control framework server, and 3) the server programs the FPGA using its local Xilinx license. According to the license terms above, these three steps do not appear to violate the agreement for either node-locked or floating licenses.

5. Conclusions

This whitepaper surveys the up-to-date wireless research testbeds' objectives, design features, and key challenges to identify the expectation and essential features to be enabled in GENI's cognitive radio testbed in support of any-layer programmable experimentation. It is concluded that the OCRP hardware platform is competitive and sufficient for a wide range of wireless research. At the same time, it is also pointed out that the current CR-Kit software architecture for OCRP does not have a number of needed features required for cleanly structured any-layer experiments. The importance of standardized experiment monitoring, measurement, and data collection support requires substantial on-board resource; it is recommended to develop a shared library of FPGA blocks and design templates to facilitate the ease of programming and efficient use of the OCRP hardware resources to support such standard functionalities.

Further efforts on education and training also play a crucial part in promoting the wide use of this resource in wide wireless research communities. The supported experiment scenarios will also be core to the testbed's value. Enabling creative ways to solicit and realize new and forward looking experiment scenarios, possibly at multiple sites by multiple teams will be ultimately useful.

6. Acknowledgements

This project is conducted with tremendous support from the GENI Cognitive Radio project team at WINLAB, Rutgers University. The author wishes to acknowledge the significant help by Ivan Seskar, Prasanti Maddala, and Khan Le. This project is funded by National Science Foundation through the GENI project contract #CNS-0940805.

References

[1] The ORBIT Network Testbed. http://www.orbit-lab.org/, last accessed in Sep. 2010.

[2] I. Broustis, J. Eriksson, S.V. Krishnamurthy, and M. Faloutsos, "A Blueprint for a Manageable and Affordable Wireless Testbed: Design, Pitfalls and Lessons Learned," in Proc. Tridentcom 2007, pp. 1-6, May 2007.

[3] J. Lei, R. Yates, L. Greenstein, and H. Liu, "Wireless Link SNR Mapping Onto An Indoor Testbed," in Tridentcom 2005, pp. 130-135, Feb. 2005.

[4] K. Lan, et al., "Implementation of a Wireless Mesh Network Testbed for Traffic Control," in Proc. ICCCN 2007, pp. 1022-1207, Aug. 2007.

[5] The Leipzig Wireless Mesh Testbed, http://rvs.informatik.uni-leipzig.de/en/forschung/testbed.php, last accessed in Sep. 2010.

[6] D. Johnson, T. Stack, R. Fish, D. M. Flickinger, L. Stoller, R. Ricci, and J. Lepreau, "Mobile Emulab: A Robotic Wireless and Sensor Network Testbed," in Proc. IEEE INFOCOM 2006, pp. 1-12, Apr. 2006.

[7] J. Etxaniz and G. Aranguren, "Bluetooth 2.0 Based Wireless Network Testbed Deployment," in Proc. UBICOMM 2008, pp. 150-153, Sep. 2008.

[8] The Virginia VineLab Wireless Testbed, http://www.cs.virginia.edu/~whitehouse/research/testbed/, last accessed in Sep. 2010.

[9] The Freie Universitat Berlin DES-Testbed, http://www.des-testbed.net/, last accessed in Sep. 2010.

[10] The Caltech Multi-Vehicle Wireless Testbed (MVWT), http://www.cds.caltech.edu/~murray/projects/durip01-mvwt/, last accessed in Sep. 2010.

[11] The WUSTL Wireless Sensor Network Testbed, http://www.cse.wustl.edu/wsn/index.php?title=Testbed, last accessed in Sep. 2010.

[12] S. Bratus, et al., "Dartmouth Internet Security Testbed (DIST): Building a Campus-wide Wireless Testbed", in Proc. USENIX Workshop on Cyber Security Experimentation and Test (CSET), pp. 1-6, August 2009.

[13] B. D. Walker, I. D. Vo, M. Beecher, and M. Seligman, "A Demonstration of the MeshTest Wireless Testbed for Delay-Tolerant Network Research," in Proc. ACM CHANTS Workshop, pp. 105-108, 2008.

[14] Y. Su and T. Gross, "Validation of a Miniaturized Wireless Network Testbed," in Proc. ACM WINTECH Workshop, pp. 25-32, 2008.

[15] B. Mathieu, F. Jan, D.-E. Meddour, Y. Gourhant, and M. Pilarski, "The Wireless Multihop-Based OneLab Testbed," in Proc. Real Overlays and Distributed System Workshop, pp. 1-2, 2007.

[16] M. Ramakrishnan and P. Vanaja Ranjan, "PICSENSE – A Wireless Sensor Network Testbed," International Journal of Recent Trends in Engineering, 1(4):59-63, May 2009.

[17] V. Naik, E. Ertin, H. Zhang, and A. Arora, "Wireless Testbed Bonsai," in Poc. International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, pp. 1-9, Apr. 2006.

[18] M. Hempel, H. Sharif, T. Zhou, and P. Mahasukhon, "A Wireless Test Bed for Mobile 802.11 and Beyond," in Proc. International Conference on Wireless Communications and Mobile Computing, pp. 1003-1008, 2006.

[19] A. Aziz, A. E. Fawal, J.-Y. L. Boudec, and P. Thiran, "Aziala-net: Deploying a Scalable Multi-hop Wireless Testbed Platform for Research Purposes." In Proc. ACM MobiHoc S^3 Workshop, pp. 1-3, 2009.

[20] P De, A. Raniwala, S. Sharma, and T. Chiueh, "MiNT: A Miniaturized Network Testbed for Mobile Wireless Research," in Proc. IEEE INFOCOM, vol. 4, pp. 2731-2742, Mar. 2005.

[21] Kansei: Sensor Testbed for At-Scale Experiments, http://ceti.cse.ohio-state.edu/kansei/, last accessed in Sep. 2010.

[22] CitySense – An Open, Urban-Scale Sensor Network Testbed, http://www.citysense.net/, last accessed in Sep. 2010.

[23] K. Farkas, T. Hossmann, F. Legendre, B. Plattner, and S. K. Das, "Link Quality Prediction in Mesh Networks," Computer Communications, 31:1497–1512, 2008.

[24] N. H. Vaidya, J. Bernhard, V. V. Veeravalli, P. R. Kumar, and R. K. Iyer, "Illinois Wireless Wind Tunnel: A Testbed for Experimental Evaluation of Wireless Networks," in Proc. ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis, pp. 64-69, 2005.

[25] Advanced wireleSS Environment Research Testbed (ASSERT), http://dslab.utdallas.edu/tnt.php, last accessed in Sep. 2010.

[26] P. Steenkiste, "The Use of a Controlled Wireless Testbed in Courses," in Proc. ACM SIGCSE Conference on Innovation and Technology in Computer Science Education, pp. 80-84, 2009.

[27] Adaptable Channel Testbed for Investigating On-the-move Wireless Nodes (ACTION), http://www.jhuapl.edu/ott/technologies/technology/articles/P02056.asp, last accessed in Sep. 2010.

[28] S. Häne, D. Perels, P. Lüthi, and A. Burg, "Real-Time MIMO-OFDM Testbed," http://www.iis.ee.ethz.ch/~mimo/Publications/MIMOResearchProjects05.pdf, last accessed in Sep. 2010.

[29] H. Sampath and A. Paulraj, "Space-Time Processing TDMA Wireless Testbed," in Proc. IEEE ICASSP'99, vol. 4, pp. 2203-2206, Mar. 1999.

[30] A. G. i Fàbregas, et al., "A MIMO-OFDM Testbed for Wireless Local Area Networks," EURASIP Journal on Applied Signal Processing, 2006(18083):1-20, 2006.

[31] D. Bates, S. Henriksen, B. Ninness, and S. R. Weller, "A 4x4 FPGA-based Wireless Testbed for LTE Applications," in Proc. IEEE PIMRC, pp. 1-5, 2008.

[32] The RACooN Lab, http://www.nari.ee.ethz.ch/wireless/research/projects/racoon/features.html, last accessed in Sep. 2010.

[33] A. Burg and H. Bölcskei, "Real-Time MIMO Testbed for Next Generation Wireless LANs," http://www.ercim.eu/publication/Ercim_News/enw59/burg.html, last accessed in Sep. 2010.

[34] S. Caban, C. Mehlfuhrer, R. Langwieser, A. L. Scholtz, and M. Rupp, "Vienna MIMO Testbed," EURASIP Journal on Applied Signal Processing, 2006(54868):1-13, 2006.

[35] P. Goud Jr., R. Hang, D. Truhachev, and C. Schlegel, "A Portable MIMO Testbed and Selected ChannelMeasurements," EURASIP Journal on Applied Signal Processing, 2006(51490):1-11, 2006.

[36] Gigabit MIMO OFDM Testbed, http://iaf-bs.de/projects/gigabit-mimo-ofdm-testbed.en.html, last accessed in Sep. 2010.

[37] P. Luethi, M. Wenk, T. Koch, W. Fichtner, M. Lerjen, and N. Felber, "Multi-user MIMO testbed," in Proc. ACM WINTECH Workshop, pp. 109-110, 2008.

[38] K.S. Bialkowski, A. Postula, A. Abbosh, and M.E. Bialkowski, "2x2 MIMO Testbed for Dual 2.4GHz/5GHz Band," in Proc. International Conference on Electromagnetics in Advanced Applications, pp 1-5, 2007.

[39] R. Spring, L. Zhou, N. Gogate, and A. S. Daryoush, "4x4 MIMO Experimental Test-bed using COTS at ISM Band," IEEE Transactions on Information Theory, 53(11):4127 – 4149, Jan. 2007.

[40] S. Zagriatski, et al., "MIMO Testbed Design at the University of Queensland," in Proc. 9th Australian Symposium on Antennas, pp. 1, Feb. 2005.

[41] B. Vandewiele and P. Mattheijssen, "An Experimental Broadband 4x4 MIMO Test-bed," in Proc. General Assembly URSI 2002, pp. 1-4, 2002.

[42] T. Horseman, J. Webber, M.K. Abdul-Aziz, R. Piechocki, A. Nix, M. Beach, and P. Fletcher, "A Testbed for Evaluation of Innovative Turbo MIMO-OFDM Architectures," in Proc. European Personal Mobile Communications Conference, pp. 453-457, Apr. 2003.

[43] G.J. Minden, et al., "Cognitive Radios for Dynamic Spectrum Access - An Agile Radio for Wireless Innovation," in IEEE Communications Magazine, 45(5):113-121, May 2007.

[44] K. Mandke, R. C. Daniels, S.-H. Choi, S. M. Nettles, and R. W. Heath, Jr., "A MIMO Demonstration of Hydra," in Proc. ACM WINTECH Workshop, pp. 101-102, 2007.

[45] WARP: Wireless Open-Access Research Platform, http://warp.rice.edu/index.php, last accessed in Sep. 2010.

[46] J. Zhou, Z. Ji, and R. Bagrodia, "TWINE: A Hybrid Emulation Testbed for Wireless Networks and Applications," in Proc. IEEE INFOCOM, pp. 1-13, 2006.

[47] A. Adjoudani, et al., "Prototype Experience for MIMO BLAST over Third-generation Wireless System," IEEE Journal on Selected Areas in Communications, 21(3):440-451, 2003.

[48] University of Florida Reconfigurable Multi-node Wireless Communication Testbed, http://www.wireless.ece.ufl.edu/testbed/, last accessed in Sep. 2010.

[49] GNU Radio, http://gnuradio.org/redmine/wiki/gnuradio, last accessed in Sep. 2010.

[50] Lyrtech Small Form Factor SDR Development Platforms, http://www.lyrtech.com/Products/SFF_SDR_development_platforms.php, last accessed in Sep. 2010.

[51] Altera Cyclone EP1C12Q240C8 FPGA, http://www.altera.com/literature/hb/cyc/cyc_c5v1_01.pdf, last accessed in Sep. 2010.

[52] Xilinx Spartan 3-2000 FPGA, http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf, last accessed in Sep. 2010.

[53] Matt Ettus, "SPARTAN FPGA on USRP2," http://old.nabble.com/SPARTAN-FPGA-on-USRP2-td26689507.html, last accessed in Sep. 2010.

[54] F. Weidling, A Design Workflow for Software Defined Radios, B.S. Thesis, University of Kansas, http://www.ittc.ku.edu/research/thesis/documents/Ted_Weidling_thesis.pdf, last accessed in Sep. 2010.

[55] J. Eriksson, H. Balakrishnan, and S. Madden, "Cabernet: Vehicular Content Delivery Using WiFi," in Proc. ACM MobiCom, pp. 199-210, 2008.

[56] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal, "Vehicular Opportunistic Communication Under the Microscope," in Proc. ACM Mobisys, pp. 206-219, 2007.

[57] T. S. Rappaport, Wireless Communication - Principles and Practice, 2nd edn, New Jersey: Prentice Hall, 1996.

[58] The Comprehensive GNU Radio Archive Network, https://www.cgran.org/, last accessed in Sep. 2010.

[59] Open Access WARP Repository, http://warp.rice.edu/trac, last accessed in Sep. 2010.

[60] The MadWifi Project, http://madwifi-project.org/, last accessed in Sep. 2010.

[61] TinyOS, http://www.tinyos.net/, last accessed in Sep. 2010.

[62] Open Virtualized WiMAX Base Station Node for GENI Wide-Area Wireless Deployments, http://groups.geni.net/geni/wiki/WiMAX, last accessed in Sep. 2010.

[63] OSSIE SCA-based Open Source Software Defined Radio, http://ossie.wireless.vt.edu/, last accessed in Sep. 2010.

[64] J. W. Wallace, B. D. Jeffs, and M. A. Jensen, "A Real-time Multiple Antenna Element Testbed for MIMO Algorithm Development and Assessment," 2004 IEEE AP-S International Symposium Digest, vol. 2, pp. 1716-1719, June 2004.

[65] Altium Designer, http://www.altium.com/Products/AltiumDesigner/, last accessed in Sep. 2010.

[66] Nallatech NAL - a C++ based Development Environment for FPGA Accelerators, http://www.nallatech.com/Development-Tools/nal-a-c-based-development-environment-for-fpga-accelerators.html, last accessed in Sep. 2010.

[67] Impulse C, http://www.impulseaccelerated.com/, last accessed in Sep. 2010.

[68] Mentor Catapult C, http://www.mentor.com/products/esl/high_level_synthesis/, last accessed in Sep. 2010.

[69] Intel Wireless WiFi Link Drivers for Linux, http://intellinuxwireless.org/, last accessed in Sep. 2010.

[70] GENI Cognitive Radio Kit, http://crkit.orbit-lab.org/, last accessed in Sep. 2010.

[71] CRKit Tutorials, http://crkit.orbit-lab.org/wiki/Software/Firmware/Framework/tutorial, last accessed in Sep. 2010.

[72] CRKit R3 Architecture, http://crkit.orbit-lab.org/wiki/Software/Firmware/Framework/R3_arch/outline, last accessed in Sep. 2010.

[73] Xilinx End User License Agreement, http://www.xilinx.com/ise/license/license_agreement.htm, last accessed in Sep. 2010.