Revolutionizing Software Defined Radio: Case Studies in Hardware, Software, and Education

Alexander M. Wyglinski, Don P. Orofino, Matthew N. Ettus, and Thomas W. Rondeau

SDR has increasingly become an invaluable research, development, and educational tool within the telecommunications sector with respect to rapidly prototyping new algorithms and paradigms in actual radio hardware and evaluating them in real-world over-the-air conditions. Due to advances in microprocessor technology, radio frequency hardware, and software, SDR has matured into a reliable tool that is now part of almost every communication engineer's toolbox.

ABSTRACT

SDR has increasingly become an invaluable research, development, and educational tool within the telecommunications sector with respect to rapidly prototyping new algorithms and paradigms in actual radio hardware and evaluating them in real-world over-the-air conditions. Due to advances in microprocessor technology, radio frequency hardware, and software, SDR has matured into a reliable tool that is now part of almost every communication engineer's toolbox, and it has changed the way the telecommunication sector produces innovative solutions to technical challenges. In this article, we explore four case studies that highlight SDR as a reliable tool in industry, academia, and government. Specifically, we study four examples that illustrate: advances in low-cost, reliable, and versatile SDR platforms, open source and universal SDR software development environments, powerful technical computing environments employing SDR hardware for real-world experimentation, and educational paradigms for synthesizing digital communications and digital processing concepts using SDR technology. By understanding the impact of these case studies, we intend to provide some insight on how the SDR revolution has changed the way the world designs and implements telecommunication systems.

20 YEARS OF SDR: A REVOLUTION IN THE MAKING

Software defined radio, or SDR, has increasingly captured the attention of the telecommunications sector over the past several decades with its promise of rapid design cycles, flexible real-time operations, reusable hardware for different transceiver implementations, ease of manufacturing, and upgrading, and accessibility to many communication system engineers, technologists, and researchers. This SDR vision, which has revolutionized the telecommunications sector over the past 20 years, has been fueled by significant advances in digital processing technologies, analog-to-digital and digital-to-analog converters, software tools, and radio hardware. Consequently, SDR technology is finally beginning to fulfill its promise, and become a mainstream, powerful, and accessible communication system and network prototyping tool.

In fact, at the time of the writing of this article, many companies, research laboratories, and universities are using SDR to support a wide range of communications-related activities.

Although SDR technology possesses significant potential to make communication system prototyping more efficient and accessible to the telecommunications community, there were several challenges that needed to be resolved in order to realize this potential. In order to enable the widespead use of SDR technology for prototyping communication systems and networks, the following conditions need to be achieved:

- *Affordable SDR hardware platforms* that possess sufficient computational horsepower, operate across a wide range of carrier frequencies and bandwidths, and are portable
- *Availability of SDR software development environments* that provide the communication technologist with a high level of control of the SDR platform, a rich set of modules, algorithms, and features, and a substantial level of accessibility to the software (i.e., shallow learning curve)
- *Support between SDR hardware and powerful technical computing software* that enables communication technologists to use reliable software models and tools in experiments using real-world SDR hardware in real time
- *Established SDR-based engineering undergraduate curricula* that introduce hands-on SDR design, prototyping, and experimentation to the next generation of communication technologists

Fortunately, the latest advances in SDR technology have recently achieved these conditions, making SDR more accessible to the telecommunications sector for use in prototyping communication systems and networks.

In this article, we present four case studies that highlight the SDR revolution by focusing on how it was fueled by advances in SDR hardware platforms, SDR software development environments, technical computing software solutions for SDR, and undergraduate educational pedagogy using SDR systems. It is expected that this article will provide the reader with a better understanding of current SDR technology, the many different layers that constitute these complex systems, and their capabilities with respect to prototyping communication systems.

The rest of this article is organized as fol-

Alexander M. Wyglinski is with Worcester Polytechnic Institute; Don P. Orofino is with MathWorks Inc.; Matthew N. Ettus is with Ettus Research; Thomas W. Rondeau is with Rondeau Research LLC

0163-6804/16/$25.00 © 2016 IEEE

	Ettus USRP N200/N210	ZedBoard w/ Xilinx Zynq-7000 FPGA & AD-FMCOMMS5-EBZ	NooElec NESDR Mini SDR USB Stick	Ettus USRP E300
Interface to host Computer	Gigabit Ethernet	Dual FMC Connectors	USB 2.0	AXI4-MM interface to an embedded dual-core ARM Cortex-A9 processor
RF front-end Instantaneous bandwidth RF frequency coverage	USRP daughterboards 25–50 MHz DC to 6 GHz (determined by daughterboard)	Integrated RFIC 56 MHz 70 MHz to 6 GHz	Integrated RFIC 3.2 MHz 24 to 1766 MHz	Integrated RFIC 56 MHz 70 MHz to 6 GHz
MIMO	1 × 1 per unit, up to 8 × 8 using multiple units	4 × 4	N/A	2 × 2
Full duplex	Yes	Yes	Rx only	Yes
ADC	Dual 14-bit 100 MS/s	Dual/quad 12-bit 61.44 MS/s	8-bit 3.2 MS/s	Dual/quad 12-bit 61.44 MS/s
DAC	Dual 16-bit 400 MS/s	Dual/quad 12-bit 61.44 MS/s	None	Dual/quad 12-bit 61.44 MS/s
FPGA RFNoC-compatible Cost	Xilinx Spartan 3A DSP No $$	Xilinx Zynq-7000 No $$$	None No $	Xilinx Zynq-7000 Yes $$$$

Table 1. Family of available SDR platforms.

lows. We provide an overview of commonly used SDR platforms, including the popular Universal Software Radio Peripheral (USRP) by Ettus Research LLC. Software development tools used to prototype SDR platforms are an essential element of the SDR revolution with respect to implementing digital transceiver designs and algorithms. Consequently, we examine several such tools, including the widely used GNU Radio project. We provide an overview of technical computing software used in enabling over-the-air experimentation via SDR technology, with a focus on the MATLAB technical computing software environment. Given all of these advances, the transformation of engineering undergraduate pedagogy with respect to digital communications by using SDR platforms for prototyping designs and algorithms is covered. Finally, several concluding remarks are given.

Building an SDR Platform for the Masses

The revolution in SDR hardware has always been tightly coupled to advances in computing technology, analog-to-digital converters (ADCs), and digital-to-analog converters (DACs). Specifically, the ability of SDR hardware to satisfactorily serve as a reliable interface between the digital world of bits, packets, and user applications and the surrounding analog RF environment of electromagnetic spectrum significantly depended on how quickly sample and continuous signal information can be converted between the two domains, as well as how fast the sample signal information can be processed. Although the technology for enabling SDR hardware has been around for decades, the cost of this technology only recently became affordable to most of the wireless community. Platforms such as the WARP Radio [1], various NUTAQ systems (formerly Lyrtech), and the Universal Software Radio Peripheral (USRP) by Ettus Research (now part of National Instruments) [2] have enabled real-time over-the-air experimentation in areas such as:

- Wireless networking
- Spectrum monitoring
- Dynamic spectrum access
- Global System for Mobile Communications (GSM), wideband code-division multiple access (WCDMA), and Long Term Evolution (LTE) mobile telephony base stations
- RADAR
- Radio astronomy and RADAR astronomy
- Wildlife tracking
- RF test equipment
- Magnetic resonance imaging (MRI)
- Motion tracking
- Radio navigation and global positioning
- Satellite communications
- RF identification (RFID)
- Wireless security research

To obtain some insight into the capabilities of present-day SDR platforms, let us study a range of systems and their technical specifications as shown in Table 1. The Ettus USRP N210 (Fig. 1a) is a widely used modular SDR platform consisting of an RF front-end (RFFE), a field programmable gate array (FPGA), and a general-purpose processor (GPP). This SDR platform is based on the initial framework devised by Matt Ettus in 2003 when he started work on the USRP in order to help lower the barrier of entry to SDR. The RFFE is tasked with bridging the digital world of sequences of timed samples and the antenna via a direct conversion architecture. A combination of low noise amplifiers (LNAs), switches, variable attenuators for gain control, local oscillators (LOs), and lowpass filters, coupled with ADCs and DACs, enabled the interfacing between the analog RF domain and the baseband digital domain. The FPGA performs all of the high-speed baseband digital signal processing (DSP) operation on the samples coming in and out, as well as all precision timing and synchronization functions, which allow

(a) (b) (c)

Figure 1. Three examples of widely-used SDR platforms: a) Ettus Research USRP N210 Platform; b) ZedBoard with Xilinx Zynq-7000 FPGA with analog devices FMC-based RF I/O; c) NooElec NESDR Mini SDR USB stick.

for such capabilities as time-division duplexing (TDD), time-division multiple access (TDMA), and multiple-input multiple-output (MIMO) operations across multiple devices. Finally, the FPGA contains the logic to interface with the GPP. To abstract away the interfacing and control of the USRP devices and instead present the user with a small set of primitives that can be used to build real-time communication systems, the USRP Hardware Driver (UHD) was developed in order to provide a single application program interface (API).

The combined ZedBoard-based Xilinx Zynq-7000 FPGA with AD-FMCOMMS5-EBZ SDR platform (Fig. 1b) is a joint venture between Analog Devices and Xilinx to provide a high-performance SDR evaluation system for the wireless community. The system-on-chip (SoC) solution available on the Xilinx Zynq-7000 FPGA enables powerful, versatile computational performance derived from both the onboard ARM processor and FPGA fabric. Thus, the Zynq-7000 can support a wide range of digital functions on this single platform. The AD-FMCOMMS5-EBZ from Analog Devices is the fifth generation of a family of high-speed analog RFFEs designed to showcase the latest generation of high-speed data converters, especially compute-intensive FPGA-based radio applications. In particular, the AD-FMCOMMS5-EBZ is designed around the AD9361 2 × 2 RF Agile Transceiver, which is capable of supporting instantaneous bandwidths of up to 56 MHz across 70 MHz to 6 GHz. As opposed to the USRP N210 SDR platform, the Xilinx Zynq-7000 FPGA/AD-FMCOMMS5-EBZ SDR platform is a standalone solution that does not require a GPP-based host (a Linux-based operating system can be supported on the Zynq-7000). To support standalone applications, Ettus Research released the USRP E300, which uses much of the same components as the Xilinx Zynq-7000 FPGA/AD-FMCOMMS5-EBZ SDR platform and possesses similar specifications and performance characteristics.

From a performance perspective, the USRP N210, Xilinx Zynq-7000 FPGA/AD-FMCOMMS5-EBZ, and USRP E300 SDR platforms are all very capable systems that can implement a wide range of solutions for different applications. On the other hand, these solutions range in cost from hundreds to thousands of dollars, which might be prohibitively expensive for relatively simple applications, such as satellite communication signal reception, wireless spectrum sensing, or applications requiring numerous SDR platforms. Consequently, the SDR market has also witnessed the advent of numerous low-cost low-complexity SDR platforms such as the NooElec NESDR Mini SDR USB Stick (Fig. 1c). On the order of tens of dollars, these simple SDR receivers plug into the USB port of a laptop computer and perform a wide range of operations based on the available software packages installed on the host computing platform.

As described previously, the USRP E300 SDR platform is a highly capable standalone wireless system that can support sophisticated digital communications and DSP functionality. The RFFE based on the AD9361 enables the USRP E300 to support wireless communications across a large part of the frequency spectrum, while both the ADC and DAC accurately interface the analog world with the digital baseband domain of the FPGA and the ARM processors on the Zynq-7000. Until recently, efficient and effective utilization of SDR computing hardware resources by the larger wireless community has always been a key technical challenge that prevented the widescale use of SDR within the wireless sector. Nevertheless, a new programming system called the RF Network-On-Chip (RFNoC) hopes to remedy this situation.

RF Network-On-Chip

RFNoC is a new programming system for FPGAs developed at Ettus Research with the goal of easing large SDR designs in FPGAs. This architecture allows users to easily integrate custom modules, such as modulators, demodulators, processors, and protocol stacks, without having to become experts on FPGA design. The basic concept behind RFNoC is that rather than treating the entire FPGA as a single monolithic sea of gates, users instead operate with a network of functional units called computation engines (CEs). This network dramatically reduces the complexity of large designs and allows for the dynamic runtime flexibility that many applications, especially cognitive radios, require.

Each computation engine has the exact same interface to the network, so they are easily interchangeable. This network of computation elements can scale across multiple FPGAs, including those of different types, and this allows for portability of CEs across all of the third generation USRP devices. It also makes dynamic reconfiguration of the FPGA a much easier task and makes it much easier to meet timing requirements in the FPGA.

The network fabric (a crossbar switch) connects the computation engines, radios, and external network interfaces. Figure 2 shows the FPGA internals of an X300 device, with external 10G Ethernet and PCIe interfaces. An E300 device would look the same, but instead of Ethernet and PCIe, it has interfaces to its on-chip ARM CPU.

The network exists inside the FPGA, but it also transparently routes across multiple FPGA devices that can be connected directly or through Ethernet switches and the like. This allows the user to easily create large systems consisting of many FPGAs and many MIMO-synchronized radios easily.

Open Source Solutions for SDR Prototyping

As discussed in the previous section, suitable digital/computing hardware combined with versatile RFFEs are some of the necessary ingredients for enabling the widescale adoption of SDR technology by the wireless community. Nevertheless, the implementation of a communication system also requires sufficient software support and tools that are accessible to the user in order to enable the rapid development and evolution of ideas and designs. A variety of open source SDR software projects exist in order to meet the needs of the community, including OSSIE [3], CubicSDR, ALOE [4], and the widely used GNU Radio project [5]. In order to obtain better understanding of these open source software frameworks for building and studying communications systems, we explore the revolution of open source software for SDR by examining the history and capabilities of GNU Radio.

GNU Radio

In 2001, Eric Blossom started the GNU Radio project with the goal of providing a framework for building SDR applications with free software. It has attracted a large community of users and developers from around the world, and has become the design environment of choice for much research in the area of SDR and cognitive radio. GNU Radio works by using a pluggable architecture, where blocks of signal processing algorithms are placed together into a graph such that samples flow through the graph, with each block operating independently on these samples to produce the radio application. With this concept of "drag-and-drop" signal processing, designs can easily be modified. Furthermore, GNU Radio comes with a variety of graphical plotting tools and simulation tools such as various channel models in order to provide straighforward ways to simulate and observe the behavior of a new design. In order to understand how software tools interface with the SDR hardware, as well as how information is passed between the hardware and software domains, let us study how GNU Radio operates in handling the design and implementation of various communication systems.

Development Models: When it comes to building communication and other signal processing systems, GNU Radio has a few modes that help enable the movement of data through the system. An intuitive way of handling the signal processing is to move the data as a stream of samples. GNU Radio has supported the streaming model since its inception, and this solves a number of communications issues. However, as interest in packet-based communications has been steadily increasing, handling packets as an infinite stream of samples has become problematic, with decreased efficiency experienced when trying to handle packet boundaries within the stream of samples. Consequently, GNU Radio also implements a message passing system, where messages of protocol data units (PDUs), which may represent a packet, frame, fragment, or similar, are transferred as a single unit. Boundary conditions

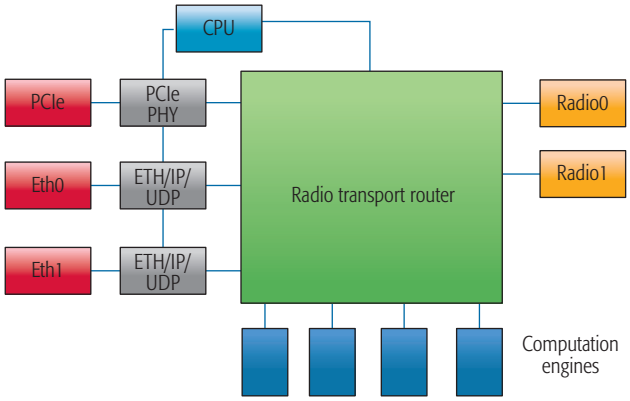

Figure 2. Schematic of the Ettus USRP E300 architecture.

are much more easily handled in this way and thus enable the focus to shift to efficient processing of the data within the PDU. The message passing system is also a useful way to signal and pass meta-data or control data between blocks.

GNU Radio has a third model for moving data around, called the tagged stream system, which is specifically designed for passing meta-data. A block can tag an item in the data stream with information about the sample, such as the time the sample was created, the signal-to-noise ratio measured at that sample, or even information about the state of the system such as the frequency and gain settings of the RFFE. The tags move synchronously with the data and are handled appropriately through sample rate changes. Each of the three models of moving data around have their uses and application spaces, and most of the real-world modems built in GNU Radio use each concept to some extent.

The radio communications research community has a number of problems that it is simultaneously addressing, such as examining existing standards to explore other use cases, improvements and efficiency issues, and security risks. Furthermore, the research community is looking at new models of communications that may or may not be tied to existing systems or methods. Finally, research is ever progressing to address the next level of challenges for wireless data communications, such as the rapidly developing push for 5G standards. GNU Radio itself as a project is not directly interested in specific standards. Instead, GNU Radio develops the architectural framework and API that enables the development and study of these issues through the out-of-tree (OOT) module project concept. On the other hand, there are several software efforts that focus on specific standards, implementations, and general technical computing research efforts, which are covered in the next section.

When SDR Meets Technical Computing Software

Once the hardware device and software framework for an SDR platform has been developed, wireless prototyping of various communication systems by the user can commence. Wireless prototyping is a workflow for the design, verification, and prototyping of radio systems. Traditionally,

Barriers to the wireless prototyping workflow are vanishing. Furthermore, the wireless community is at an inflection point in the pursuit of wireless prototyping given the availability of a wide range of affordable SDR platform options.

wireless prototyping required mastery of many distinct tools, languages, and interfaces, with very little tool integration and workflow usability. This posed a high cost of entry and limited wireless prototyping to highly motivated and well funded organizations. Technical computing software solutions, such as MATLAB [6] and SystemVUE [7], enable this workflow; often, model-based design is the preferred approach for design iteration. Technical computing software has made significant progress in recent years, with comprehensive support for cellular, wireless local area networks, and other wireless waveforms, enabling engineers to prototype commercially relevant systems in hours instead of months. Barriers to the wireless prototyping workflow are vanishing. Furthermore, the wireless community is at an inflection point in the pursuit of wireless prototyping given the availability of a wide range of affordable SDR platform options (refer to Table 1).

Wireless Prototyping Workflow

A compelling vision for the wireless prototyping workflow embodies four steps and places significant demands on hardware integration with technical computing software.

System Simulation: The initial step of the workflow executes all algorithms on the desktop in a convenient and interactive fashion with synthesized data. Desktop execution sidesteps the constraints of the target embedded system, making it easier to explore algorithm alternatives, identify execution errors, and tune parameters as a simulation is underway.

RF Integration: The second step configures the RF I/O so that the desktop simulation receives and transmits data using the target hardware. This enables the simulation to include sensor noise, quantization, fading, and power levels that typically influence design decisions. RF signals can be recorded for use during repeated testing and verification using real-world data, or streamed to the desktop in real time. This enables desktop testing of wireless systems and typically implicates multicore, GPU, and other acceleration techniques that increase simulation throughput.

Incremental Deployment: The third step generates code for elements of the design, replacing desktop simulation with streaming execution on target hardware. The highest-rate elements in the front-end of the radio are typically moved to target hardware first, while the rest of the design remains on the desktop. For an SDR, these elements are often destined for execution on an FPGA. Additional elements are transitioned to and validated on the target hardware iteratively, using bit error rate (BER) or other quality metrics. This step places high importance on automatic code generation and data transfer between desktop and target hardware.

System Validation: The final step executes the design on the target hardware and validates it for correctness relative to simulation results obtained in the first step. Synchronous (gated execution) and asynchronous (full-speed streaming) execution of hardware can be employed, using techniques such as FPGA-in-the-loop, to gain confidence in final system operation.

This workflow enables practitioners to minimize "time to next insight" throughout system design. The productivity afforded by each step has proven significant on its own, based on feedback from academic, research, and commercial wireless prototyping communities for the MathWorks technical computing environment. Gaining productivity from a technical computing environment should not require adopting the entire workflow.

An unexpected benefit of pursuing SDR with a technical computing platform is the use of RF modeling tools integrated into the platform. Predicting the imperfections of RF hardware is difficult, due in part to the complexity of nonlinear RF transceivers and the impact of antenna arrays; for example; the design of compensation and equalization algorithms requires simulation. RF modeling tools enable higher fidelity simulation of the complete wireless system.

An Example of Wireless Prototyping: 4G LTE Cellular Communications

As an illustration of how straightforward it is to perform the wireless prototyping workflow, let us consider the example of migrating a fourth generation (4G) LTE implementation from the MATLAB technical computing software environment to an actual SDR hardware platform. Referring to Fig. 3, the LTE test waveform is synthesized using an LTE resource grid that is initially designed and then analyzed in MATLAB. This figure also summarizes many details within this test signal. The LTE waveform is then transmitted using an SDR platform consisting of a Xilinx Zynq-7000 FPGA with an Analog Devices FMC-based RFFE, received by a second SDR platform, and processed using MATLAB in order to analyze the equalization of the physical downlink control channel (PDCCH) symbols and channel magnitude frequency response.

Note that throughout this experience, the communication systems engineer did not need to leave the MATLAB technical computing software environment. Instead, once an implementation for the 4G LTE system has been verified via computer simulation, it can readily be applied to actual SDR hardware for experimentation. Thus, this example highlights the capability of SDR technology with technical computing software. As shown in the following section, this functionality can also be leveraged as a powerful educational tool for teaching wireless communications.

Project-Based Learning with SDR

SDR technology has become a viable instructional resource for the teaching of undergraduate courses in digital communications. The relatively low cost of the SDR hardware can fit within the equipment budget of an academic department. Moreover, the increasing amount of software support for SDR systems, the reliability of SDR/software integration, and the growing familiarity of undergraduate students with the array of available technical/scientific software tools have all contributed to a decrease in the learning curve associated with the prototyping of communication systems using SDR. Finally, the availability of SDR-based instructional resources for educators in the form of ready-to-use models

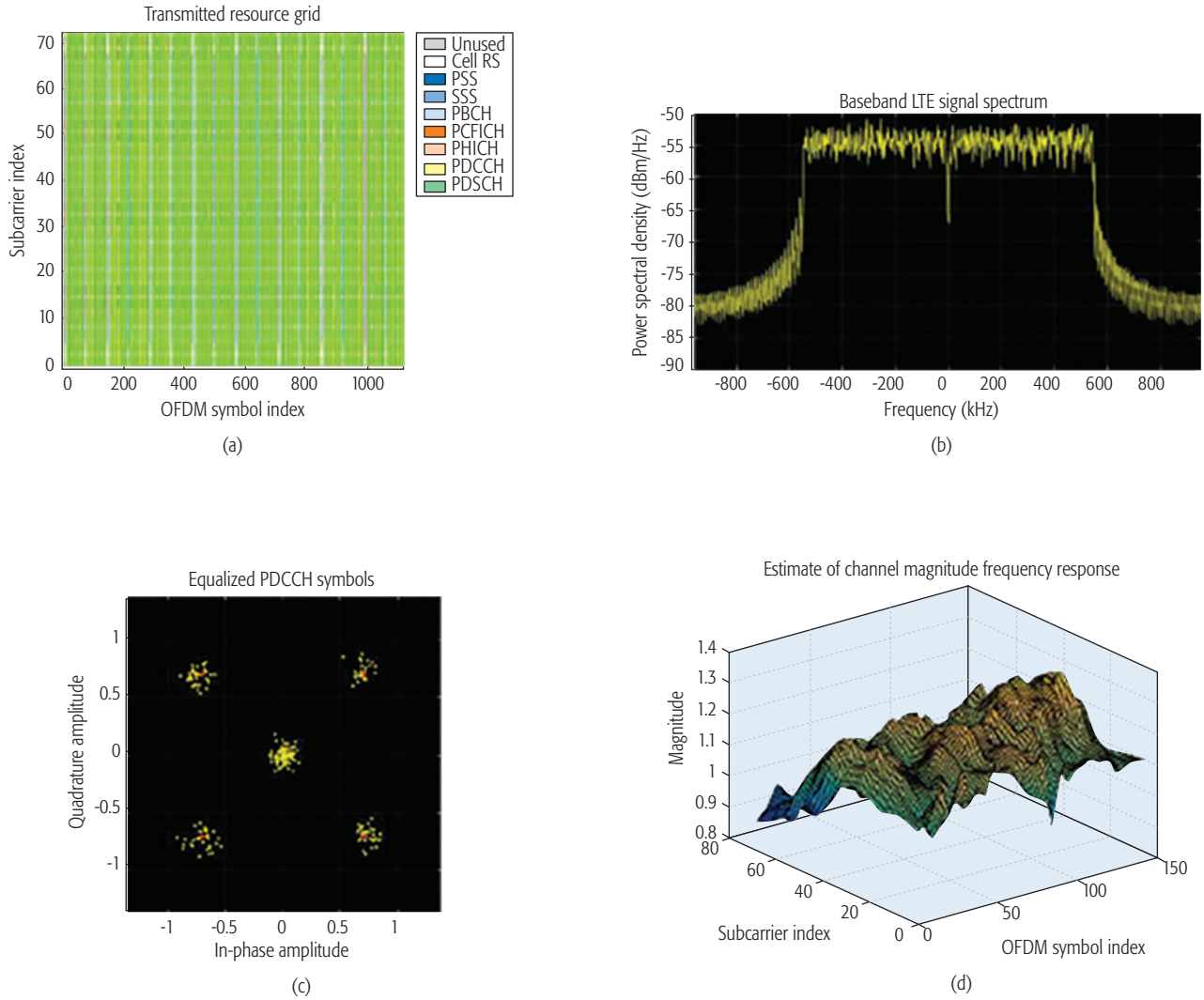

Figure 3. Results obtained using MATLAB, LTE system toolbox and Zynq SDR support from the communications system toolbox.

and demostrations, laboratory guides, and textbooks has enabled the introduction of SDR into the classroom [8].

Although the technology and resources for SDR-based education are finally becoming widely available to both instructors and students, it is very important to understand the context in which SDR can be deployed within a classroom environment in order to maximize the educational experience. Excellent coverage of lessons learned via the pedagogical usage of SDR technology at Penn State, Worcester Polytechnic Institute (WPI), the United States Naval Academy, Indiana University/Purdue University Fort Worth, University of Utah, and Virginia Tech was presented in a 2014 Feature Topic on Education of *IEEE Communications Magazine* [9]. Based on these different university experiences, the use of SDR in the classroom as a tool for teaching, reinforcing, and synthesizing concepts in digital communications can be decomposed into three steps (Fig. 4):

Step 1. Sample-based perspective of digital communications

Step 2. Insights on the fundamental building blocks of these systems

Step 3. Open-ended communication system design experiences

Understanding the Analog/Digital Divide

Most approaches for teaching undergraduate digital communications focus on the study of the transceiver from a binary perspective that gradually works toward the RFFE. Conversely, when it comes to working with SDR hardware, the key challenge in getting a digital communication system working is understanding how the ADC and DAC operate, since without the correct samples nothing else will function properly [10]. Thus, it has been observed that a digital communications course using SDR yields the best outcomes when students start the course from a sample-based perspective. Thus, students need to understand the functions of both the ADC and DAC, decimation and interpolation, and other discrete time signals and systems fundamentals.

Several key concepts with respect to sampling covered in SDR-based undergraduate courses include the following.

Signal Bandwidth: Given that the ADC and DAC on the SDR platform often operate at a fixed sampling rate, students need to understand

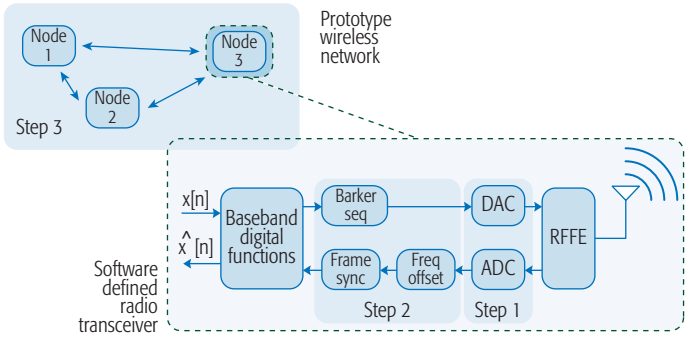

Figure 4. Illustration of the educational paradigm employed in teaching digital communications using SDR hardware.

how the digital data can be interpolated to or decimated from a target sample rate.

Appropriate Sampling Instants: Students must understand the physical issues associated with the ADC in terms of sampling at the correct instants, as well as performing decimation that results in the target samples being discarded.

Sampling Underflow and Overflow: There are several data bottlenecks in the overall setup of the SDR system, such as Gigabit Ethernet and available processing power. Thus, students are exposed to scenarios when too much data is received by an SDR system such that some of the information is lost (i.e., overflow) as well as when not enough data is provided to the USRP N210 such that gaps appear in the transmission (i.e., undeflow).

Open-Ended Design Experience

Once the sampling concept and fundamental modules needed for the construction of a functional digital communication transceiver have been covered by an SDR-based course, this provides several educational opportunities for presenting concepts in medium access control and wireless networking. One approach that could be employed is the *open-ended course design project*, where students work in teams of two or three on a task with specific objectives but loosely defined constraints on how to achieve them. The purpose of these projects is to synthesize the concepts already taught in class, combined with the aforementioned modules, in order to pursue a project-based experience that also promotes teamwork and hands-on learning in order to yield a real-world solution to a problem.

Step 3 of Fig. 4 shows how communication nodes can serve as the building blocks for relatively complex network architectures with sophisticated medium access protocols, such as:

- Wireless ad hoc networks capable of bootstrapping from scratch
- Jamming-resistant multihop wireless networks
- Scaled-down cellular networks

These kind of projects expose students to the challenges of coordinating multiple wireless transceivers in order to perform a variety of different operations. Some of these challenges include bidirectional communications, establishing contact between two wireless nodes, medium access control, and timing of network operations. Consequently, students obtain substantial insight into a wireless network consisting of digital communication transceivers from the bottom up. In particular, these projects using actual SDR hardware to prototype communication systems and wireless networks have the ability to help synthesize and reinforce digital communication concepts while exposing students to real-world issues encountered during transceiver and network prototyping.

Conclusion

Advances in SDR technology have revolutionized the way the telecommunications sector conducts research, development, and educational activities. Low-cost, accessible, and reliable SDR hardware coupled with open source SDR development environments and powerful technical computer software capable of interfacing with SDR platforms have significantly transformed the way we all think of prototyping new communication systems and networks. With hands-on SDR-based communications and networking pedagogy being introduced in engineering undergraduate curricula, the skill set needed to wield these SDR tools is becoming more widely available among the next generation of telecommunication technologists. Twenty years ago, many of the advantages and capabilities of SDR technology that we take for granted today were unrealizable. Given the rate at which advances are being made in this sector, it is expected that this revolution in SDR technology will continue for another 20 years.

References

[1] P. Murphy, A. Sabharwal, and B. Aazhang, "Design of WARP: A Wireless Open-Access Research Platform," *Proc. 2006 14th Euro. Signal Processing Conf.*, Sept. 2006, pp. 1–5.

[2] R. Dhar *et al.*, "Supporting Integrated MAC and PHY Software Development for the USRP SDR," *Proc. 1st IEEE Wksp. Networking Technologies for Software Defined Radio Networks*, Sept. 2006, pp. 68–77.

[3] M. Robert *et al.*, "OSSIE: Open source SCA for Researchers," *Proc. SDR Forum Tech. Conf.*, vol. 47, 2004.

[4] I. Gomez, V. Marojevic, and A. Gelonch, "Aloe: An Open-Source SDR Execution Environment with Cognitive Computing Resource Management Capabilities," *IEEE Commun. Mag.*, vol. 49, no. 9, Sept. 2011, pp. 76–83.

[5] E. Blossom, "GNU Radio: Tools for Exploring the Radio Frequency Spectrum," *Linux J.*, vol. 2004, no. 122, p. 4, 2004.

[6] J. Proakis, M. Salehi, and G. Bauch, *Contemporary Communication Systems Using MATLAB*, Cengage Learning, 2012.

[7] D. Silage, *Digital Communication System Using System VUE*, Firewall Media, 2006.

[8] D. Pu and A. M. Wyglinski, *Digital Communication Systems Engineering with Software-Defined Radio*, Artech House: Norwood, MA, USA, 2013.

[9] S. G. Bilen *et al.*, "Software-Defined Radio: A New Paradigm for integrated Curriculum Delivery," *IEEE Commun. Mag.*, vol. 52, no. 5, 2014, pp. 184–93.

[10] R. G. Machado and A. M. Wyglinski, "Software-Defined Radio: Bridging the Analog-Digital Divide," *Proc. IEEE*, vol. 103, no. 3, 2015, pp. 409–23.

Biographies

Alexander M. Wyglinski [S'99, M'05, SM'11] (alexw@wpi.edu) received his B.Eng. and Ph.D. degrees from McGill University, Montreal, Quebec, Canada, in 1999 and 2005, respectively, and his M.Sc.(Eng.) degree from Queens University, Kingston, Ontario, Canada, in 2000, all in electrical engineering. He is an associate professor of electrical and computer engineering with Worcester Polytechnic Institute, Massachusetts, and the director of the Wireless Innovation Laboratory (WI Lab). Throughout his academic career, he has published over 35 journal papers, over 75 conference papers, nine book chapters, and two textbooks. His current research activities include wireless communications, cognitive radio, software-defined radio, dynamic spectrum access, spectrum measurement and characterization, electromagnetic security, wireless system optimization and adaptation, and cyber-physical systems. He is currently or has been sponsored by organizations such as the Defense Advanced Research Projects Agency, the Naval Research Laboratory, the Office of Naval Research, the Air Force Research Laboratory Space Vehicles Directorate, MathWorks, Toyota InfoTechnology Center U.S.A., and the National Science Foundation. He is a member of Sigma Xi, Eta Kappa Nu, and the ASEE.

DON P. OROFINO is director of engineering for signal processing at MathWorks, leading product development for the analysis, design, and implementation of signal processing, communications, and computer vision systems. He holds 18 patents related to engineering system simulation and automatic code generation. Prior to joining MathWorks in 1995, he worked at Hewlett-Packard Imaging Systems (Andover, Massachusetts). He earned a Ph.D. in Electrical and Communications Engineering at Worcester Polytechnic Institute.

MATTHEW N. ETTUS is a core contributor to the GNU Radio project, a free framework for software radio, and is the creator of the Universal Software Radio Peripheral (USRP). In 2004, he founded Ettus Research to develop, support, and commercialize the USRP family of products. Ettus Research was acquired by National Instruments in 2010, and he continues as its president. USRPs are in use in over 100 countries for everything from cellular and satellite communications to radio astronomy, medical imaging, and wildlife tracking. In 2010, the USRP family won the Technology of the Year award from the Wireless Innovation Forum. In the past he has designed Bluetooth chips, GPS systems, and high-performance microprocessors. Before that, he received B.S.E.E. and B.S.C.S. degrees from Washington University, and an M.S.E.C.E. degree from Carnegie-Mellon University. In 2011, he was named an eminent member of Eta Kappa Nu. He is based in Mountain View, California.

THOMAS W. RONDEAU holds a Ph.D. from Virginia Tech in eectrical engineering, graduating in 2007. He is the current maintainer and lead developer of GNU Radio and is a visiting scholar with the University of Pennsylvania. He also works as a consultant on GNU Radio and wireless technology through his firm Rondeau Research, LLC. His Ph.D. dissertation on aritifical intelligence applied to wireless communications received the Council of Graduate Schools' Distinguished Dissertation for Math, Science, and Engineering. His research interests span areas of communications theory, signal processing, and software design, which are all part of his larger interests in software and cognitive radios.