Addressing Next-Generation Wireless Challenges with Commercial Software-Defined Radio Platforms

George Sklivanitis, Adam Gannon, Stella N. Batalama, and Dimitris A. Pados

ABSTRACT

We review commercially available software-defined radio platforms and classify them with respect to their ability to enable rapid prototyping of next-generation wireless systems. In particular, we first discuss the research challenges imposed by the latest software-defined radio enabling technologies including both analog and digital processing hardware. Then we present the state-of-the-art commercial software-defined radio platforms, describe their software and hardware capabilities, and classify them based on their ability to enable rapid prototyping and advance experimental research in wireless networking. Finally, we present three experimental testbed scenarios (wireless terrestrial, aerial, and underwater) and argue that the development of a system design abstraction could significantly improve the efficiency of the prototyping and testbed implementation process.

INTRODUCTION

Since the early 1990s, the software-defined radio (SDR) or "software-radio" architecture, conceived by the seminal work in [1], changed the landscape of radio engineering by leveraging the flexibility provided by programmable software-reconfigurable hardware. Defining and programming radio communication functionalities in software that control heterogeneous hardware platforms such as general-purpose processors (GPPs), digital signal processors (DSPs), and field programmable gate arrays (FPGAs) was envisioned as a compelling solution for the development of flexible and reconfigurable wireless networks.

As of today, commercially available SDR platforms are capable of tuning in software critical physical layer communication parameters such as carrier frequency, bandwidth, modulation, and data rate. However, the ever growing demand for communication bandwidth, end-to-end reliability, and self-awareness under dynamically changing channel conditions requires SDR systems that exhibit self-organization and reconfiguration capabilities across all the layers of the network protocol stack [2]. Arguably, such systems would significantly benefit testbed developments in emerging research areas such as cognitive radio, multiple-input multiple-output (MIMO) communications (i.e., systems with multiple transmit/receive antennas), full-duplex multi-user MIMO, and massive MIMO (also known as large-scale antenna systems).

Existing hardware technologies such as FPGAs, DSPs, and GPPs enable individually, modular digital signal processing. However, rapid simulation, a necessary feature for modern SDR systems [3], rapid prototyping, and real-time experimental testing of next-generation/reconfigurable cross-layer optimized wireless network protocols require the development of comprehensive software environments that are able to i) adopt a holistic hardware-software approach to wireless system design; ii) provide heterogeneous multiprocessing hardware capabilities to modularize algorithmic deployment and match computational needs of signal processing software modules to computational capabilities of the available hardware platforms; iii) provide multiple layers of abstraction to tame hardware complexity and allow rapid iteration between simulation and prototyping.

In an attempt to address latency and throughput needs in software-defined wireless networking, recent software developments in both GPP-centric and FPGA-centric SDR architectures follow parallel paths with little or no cross-fertilization between the two software communities. Consequently, there are a plethora of software tools and pertinent libraries that are tightly integrated with specific hardware platforms, without offering consistent abstractions toward a unified SDR system design and simulation philosophy. Software tools are distinguished into open and closed source (proprietary) programming environments, each providing different levels of flexibility and integration with hardware. Furthermore, distributing processing across heterogeneous hardware platforms toward, for instance, optimizing the implementation of resource-demanding cross-layer networking protocols requires developers to master multiple software tools. At the same time, hardware configurations vary across different SDR vendors, while software portability is not guaranteed across different SDRs. Hence, facilitating the transition from concept to simulation and then to prototype can become quite a tedious

The authors review commercially available software-defined radio platforms and classify them with respect to their ability to enable rapid prototyping of next-generation wireless systems. They discuss research challenges imposed by the latest software-defined radio enabling technologies including both analog and digital processing hardware, and they present the state-of-the-art commercial software-defined radio platforms.

The authors are with the State University of New York at Buffalo.

0163-6804/16/$25.00

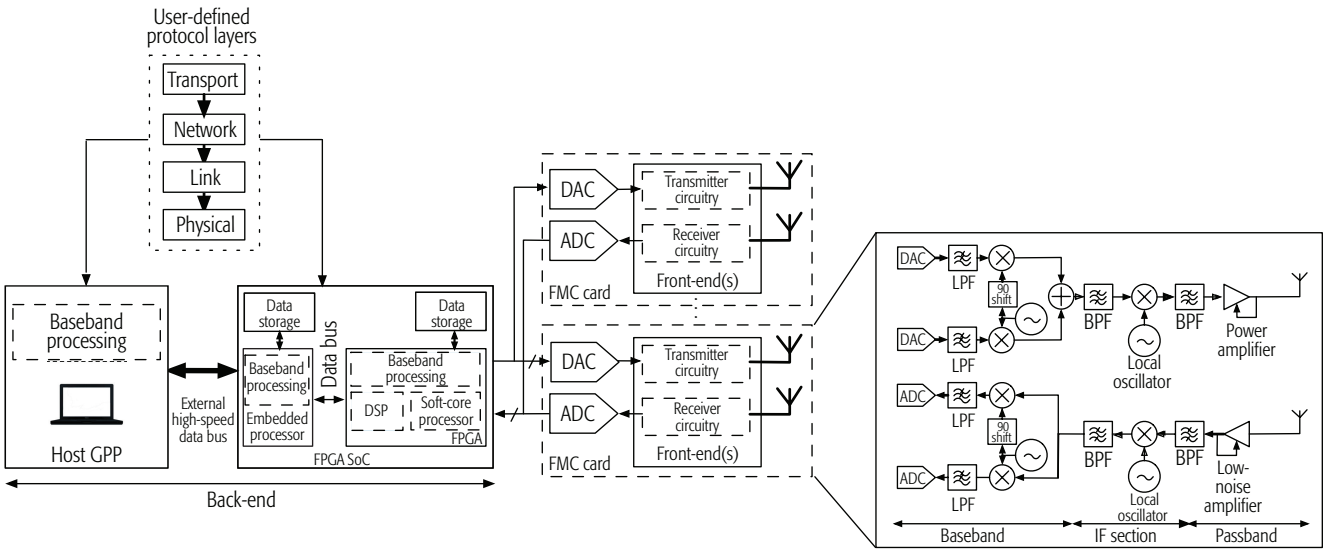

Figure 1. Generic software-defined radio architecture. Static analog transmit/receive circuitry (front-end) is interfaced with programmable hardware processing platforms (back-end) through ADCs/DACs. Analog-to-digital converters (ADCs)/digital-to-analog converters (DACs) and analog front-end radio circuitry may be featured under the same compact card, referred to as an FPGA mezzanine card (FMC).

and daunting process for developers who must trade off development time for system optimization and vice versa, until they finely tune to the wireless system's desired performance for different communication standards.

In light of the above, in this article we present current SDR challenges with respect to rapid prototyping and testing of reconfigurable software-defined wireless networks, review the existing state-of-the-art SDR platforms, and discuss their limitations with respect to projected challenges of the next-generation programmable wireless networking experimental testbeds and applications.

The rest of the article is organized as follows. We first provide an overview of a generic SDR architecture and discuss in detail the emerging challenges toward rapid experimental assessment and testing of novel wireless networking protocol proposals. Then we list and classify state-of-the-art commercial SDRs according to the capabilities and limitations of the underlying hardware and software platforms. Finally, we highlight key future challenges and directions dictated by the real-world experimental requirements of three diverse (i.e., terrestrial, aerial, underwater) wireless networking testbeds.

Software-Defined Radio Architecture

Software-defined radio proposes a paradigm shift from inherently inflexible dedicated-functionality hardware radio platforms by combining analog static or parameterizable (front-end) circuits and software reprogrammable digital hardware (back-end) components that are easily reconfigurable via software updates. Software-defined architectures are therefore ideal for rapid prototyping, and testing of new military applications and commercial standards. In the context of this work we use the term front-end to describe the analog signal processing stages between the antenna and analog-to-digital converters (ADCs) or digital-to-analog converters (DACs), and the term back-end to refer to software reprogrammable digital processing platforms such as GPPs, DSPs, and FPGAs.

Figure 1 illustrates a generic SDR architecture adopted by the majority of commercially available SDRs, consisting of an analog front-end interfaced with ADC/DAC converters, an FPGA, and a GPP. The SDR front-end consists of analog circuitry responsible for up/down-conversion of analog information signals directly to either the passband or baseband, respectively (homodyne, zero-intermediate frequency [IF] architecture) or an IF ([super-] heterodyne architecture). Bandpass and lowpass filters, and amplifiers in the front-end are used for signal conditioning. The analog front-end is interfaced with high sample rate/resolution ADCs that sample the baseband signal, and DACs that convert digital samples to analog waveforms for transmission. As a result, analog passband signals that arrive at the receiver antenna(s) are first bandpass filtered, then amplified via a low-noise-amplifier (LNA), down-converted directly to baseband or optionally to an IF, lowpass filtered, and finally amplitude leveled/power normalized by an automatic gain controller (AGC) before being sampled at the Nyquist rate by the ADC. The reverse process is followed at the transmit chain of the front-end, where incoming complex baseband signals from the DAC are filtered, up-converted, and amplified for passband transmission.

Baseband signal processing on the ADC output and DAC input digital samples is handled by the SDR back-end. User-defined protocol functionalities (Fig. 1) at different layers of the network protocol stack exhibit variable latency and memory requirements for multiple communication standards, so the wireless system designer may decide to split the execution of certain functionalities to heterogeneous hardware platforms (e.g., GPP, FPGA). The high parallelism offered by the FPGA is usually leveraged to accelerate the implementation of computationally demand-

ing signal processing operations (e.g., filters) on incoming/outgoing data from/to the ADC/DAC at the expense of increased implementation complexity. Digital data can be transferred to and stored long-term in an external storage medium such as a secure digital (SD) card or an onboard synchronous dynamic random access memory (SDRAM) for faster access, while upper layer (e.g., link and network layer) functionalities may be handled by either a software co-processor implemented in the FPGA, an embedded DSP, or a GPP (as depicted in Fig. 1). GPPs are well suited for the implementation of highly branching programs and offer short development times by exploiting high-level software programming languages. However, real-time operating systems at GPPs offer low resolution in strict real-time data flow constraints. Typical SDR designs implement physical layer functionalities and handle the data at the packet level at a GPP. Digital samples are then transferred to the FPGA through an external high-speed data bus connection (e.g., Gigabit Ethernet). GPPs are either external host PCs or embedded system-on-chip (SoC) processors, sometimes even incorporated in the same integrated circuit (IC) package with the FPGA.

Software-Defined Radio Challenges in Next-Generation Wireless Systems Prototyping

Next-generation wireless networking protocols and sophisticated network topologies such as multi-user MIMO are difficult to model and test in a software simulation environment. For example, software simulation mostly relies on simplified channel fading models that do not incorporate critical real-world networking conditions or hardware impairments. As a result, researchers can either loosely emulate in software the performance of new signal processing and wireless networking algorithms, or experimentally validate them in heterogeneous multiprocessing hardware platforms consisting of application-specific ICs (ASICs), FPGAs, GPPs, and DSPs. Figure 2 illustrates the trade-off between reconfigurability and development time for FPGA, ASIC, DSP, GPP, and hybrid GPP/FPGA-centric SDR architectures.

ASIC implementations have a static application-specific architecture and provide tailored processing units to optimize computational efficiency and power consumption for dedicated functionalities. On the other hand, FPGAs are field-programmable for different applications and provide rapid reconfiguration between different signal processing designs, thus enabling the development of software programmable SDR platforms at the expense of increased power consumption and circuit area. Reconfigurable SDR platforms aim to minimize the utilization of ASIC modules such as analog filters, amplifiers, digital down and up converters (DDC and DUC) due to their limited flexibility, or allow their parameterization and runtime reconfiguration through primitive functions that are activated by a processor such as a DSP or a GPP. DSPs offer the best trade-off between processing power and power consumption by providing optimized features especially targeted for digital signal processing operations (e.g., combined multiply-accumulate operations). They are usually embedded in FPGA hardware platforms to efficiently address signal processing tasks that can be pipelined (i.e., sequenced and repeated for each sample in a buffer). In addition, GPPs are real-time reprogrammable processing alternatives that can handle processing of a wide variety of applications at low implementation complexity. Furthermore, multi-core architectures of DSPs and GPPs can enhance processing performance by executing multiple operations in parallel.

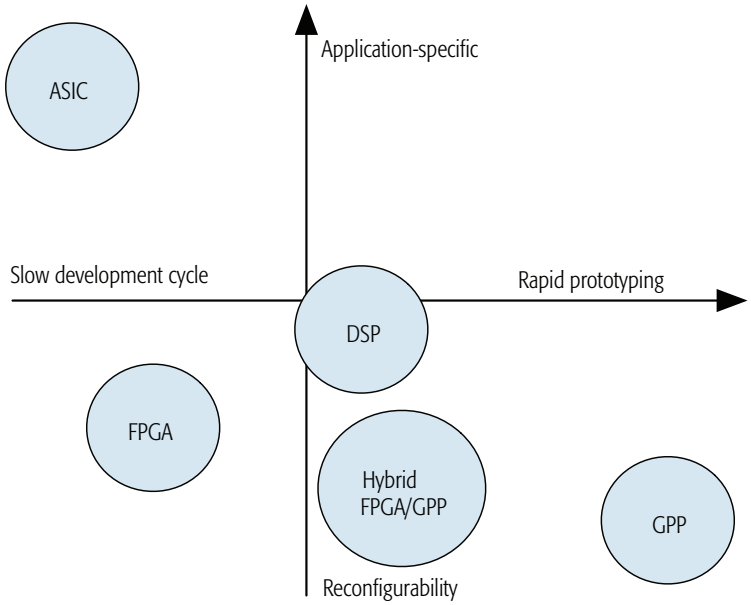

Figure 2. Trade-off between reconfigurability and development time for FPGA, ASIC, DSP, GPP, and hybrid GPP/FPGA-centric SDR architectures.

The rest of this section reviews and discusses the major emerging hardware and software challenges in existing commercially available SDRs toward accelerating experimental assessment and testing of novel wireless networking protocols. In this context, we study and classify SDR systems according to the following criteria:

- Level of flexibility and efficient interaction between analog front-end and digital back-end hardware technologies
- Multiprocessing capabilities in heterogeneous hardware platforms
- Level of abstraction between software environments and hardware platforms

Analog/Digital Hardware: The analog nature of the transmission medium (air, water, soil, etc.) as well as the requirements for multi-band support to accommodate multiple standards challenge the design of reconfigurable SDR architectures with respect to both the design of the analog front-end circuitry and their respective physical interface with back-end digital processing platforms. As an example, the (super)-heterodyne transceiver architecture depicted in the circuitry design of Fig. 1 fails to reconfigure its fixed narrowband components, such as channel selective filters, to meet the broadband requirements of multiple standards. On the other hand, the homodyne (zero-IF) transceiver architecture

The majority of existing SDR software tools are tightly integrated with specific hardware platforms with few or no abstractions available to the wireless system designer. As a result, optimization of the system design flow at a high level depends on software-hardware co-design for objects executed in processors and FPGAs, respectively.

minimizes the number of analog (fixed) components and is a flexible design alternative to the (super)-heterodyne architecture at the expense of increased baseband interference and phase noise (DC offset due to local oscillator leakage and IQ imbalance, respectively), which can be partially compensated for in the digital domain. Flexible receiver design implementation with minimal analog components requires ADCs with high dynamic range to be able to detect weak desired signals in the presence of possibly strong undesired in-band signal interference. Furthermore, the effective (useable) bandwidth of the SDR system of Fig. 1 is defined as the minimum of:

- The radio's analog bandwidth, which should not exceed the ADC/DAC sample rate
- The FPGA processing bandwidth, which depends on the FPGA clock rate
- The host-GPP bandwidth

Typical commercial SDR platforms either include a high-speed 1/10 Gigabit Ethernet (GigE) connection to address the host GPP-FPGA communication or implement FPGA-based soft-core processors or select wireless-focused embedded hard processors (e.g., ARM). In this way, the FPGA is able to handle GPP-like branching logic. Therefore, performance efficiency for multiple standards entails effective interaction and data transfer optimization between different hardware platforms at the SDR back-end.

Heterogeneous Multiprocessing: Multiprocessing is an SDR intrinsic requirement that aims to enhance SDR computational efficiency by off-loading complex signal processing operations such as finite impulse response (FIR) filtering or fast Fourier transform (FFT) to heterogeneous hardware platforms. However, software development of such architectures suffers from increased implementation complexity and lack of a standard methodology for partitioning the implementation of signal processing functionalities to heterogeneous hardware platforms [4]. As an example, GPPs and DSPs have memory architectures well suited for streaming data and rely on sequential execution of a single instruction stream. They are single-instruction multiple-data (SIMD) units where each processor is constrained to execute the same instruction at a single program but operate over multiple data streams in parallel. Therefore, GPPs and DSPs are suitable for simple computational tasks such as modulation/demodulation and encoding/decoding, higher-level logic applications, and network and medium access layer functions. On the other hand, FPGAs are well suited for implementation of logical functions that can be separated and run simultaneously or for execution of tasks that use non-standard data-type representations (i.e., 12- or 14-bit ADC/DAC); however, they exhibit increased development time and complexity. Data-type consistency, sampling rate adaptation, and resource mapping and scheduling of software applications for heterogeneous hardware are additional architectural considerations in the development of heterogeneous multiprocessing platforms. Exchanging a common data stream between multiple hardware architectures that use different data-type representations, say between a GPP that utilizes floating-point SIMD instruction sets and a DSP or FPGA that uses fixed-point representations, requires data-type conversions. Data-type conversions (e.g., conversion of a 12/14-bit ADC output to 16-bit) short at GPP input lead to increased processing load. Furthermore, sampling rate adaptation is necessary for interfacing multiple subsystems with different clock rates as well as for fine tuning to the sampling rate of multiple communication standards. Implementation of reconfigurable sampling rate adaptors suffers from the design of highly dynamically adaptive and computationally efficient FIR filters. Thus, existing SDR architectures assign the implementation of sampling rate adaptors to the FPGA. Finally, automatic instead of ad hoc resource mapping for heterogeneous multiprocessing may enable adaptation of protocol execution speeds that can be traded off during runtime to satisfy pertinent latency requirements of the specific communication standard.

Hardware Abstraction: The majority of existing SDR software tools are tightly integrated with specific hardware platforms with few or no abstractions available to the wireless system designer. As a result, optimization of the system design flow at a high level depends on software-hardware co-design for objects executed in processors and FPGAs, respectively. As an example, a researcher willing to develop and experimentally evaluate a complex modulation waveform or channel/source coding scheme is required to develop and optimize signal processing components, as well as decide on the optimal execution architecture according to the available hardware platforms and the respective software tools. Nevertheless, the implementation of abstract hardware application programming interfaces (APIs) that offer granular control and separate the execution architecture from signal processing may allow trading off prototyping time for optimization of platform resources and vice versa. In addition, the availability of dedicated software tools for particular hardware platforms in combination with hardware abstractions may give the wireless system designer the opportunity to integrate optimized intellectual property (IP-protected) processing components developed by domain specialists or third party developers, and rely only on abstract characteristics such as memory use and execution speed. Multiple levels of granularity in hardware control may also accelerate the transition from simulation to prototyping by providing bit level visibility into the design and the capability to quickly test its functional behavior. FPGA software development will definitely benefit from such abstractions as currently, long compile times, synthesize times, and place-and-route times are prohibitive for trial-and-error performance assessment of new wireless designs.

Commercial Software-Defined Radio Platforms

For small-scale laboratory testbed setups, commercially available SDR platforms offer low-cost hardware and software solutions for rapid experimental assessment of programmable wireless

	Ettus/NI USRP		Nutaq		Mango WARP		Per Vices		Epiq	
	GPP	FPGA	GPP	FPGA	GPP	FPGA	GPP	FPGA	GPP	FPGA
GNU Radio	•	N/A	•	N/A	○	N/A	•	N/A	•	N/A
Mathworks	•	○	•	•	•	•	○	○	○	○
NI LabVIEW	•	•	○	○	○	○	○	○	○	○

• Compatible option; ○: Incompatible option; N/A: option is not available.

Table 1. Software/hardware compatibility for state-of-the-art commercially available SDR platforms.

networks. In the following section, we discuss the strengths and limitations of both software frameworks and hardware architectures with respect to rapid prototyping and testing of next-generation wireless systems. We also provide a compatibility overview between the available software tools and heterogeneous SDR platforms in Table 1.

Software Frameworks

GNU Radio: GNU Radio is an open source software framework that follows a component-based design, where signal processing chains are broken into primitive components/blocks, therefore enabling code reusability and rapid block reconfiguration. Each block is assigned to a dedicated processor thread, while data exchange between blocks is achieved through shared memory buffers. GNU Radio applications are only supported by GPPs (Table 1) as well as embedded processors that support floating point SIMD instruction sets. New GNU Radio applications, called flow graphs, are programmed in Python and C++. The API of the GNU Radio framework enables integration of optimized signal processing blocks (GNU Radio IP) such as modulators and demodulators. The framework allows simulation of the performance of new physical layer communication designs through an XML-based graphical user interface (GUI), called GNU Radio Companion (GRC). With respect to performance acceleration of pure GPP-centric designs, GNU Radio provides a programming tool, named VOLK (vector-optimized library of kernels) [5], which enables vectorized mathematical operations and is independent of the processor's architecture. VOLK offers an abstraction layer to hardware-specific SIMD implementations, which vary across different processor families and vendors. Packet-based processing in upper layers is enabled by stream tagging and asynchronous message passing software APIs. More specifically, stream tags enable sample streams to carry metadata information (e.g., information on packet boundaries), while message passing enables asynchronous parameter reconfiguration in flow graph blocks regardless of their location in the data flow (upstream or downstream blocks). As a result, GNU Radio provides rapid software simulation and testing of new GPP-centric wireless system designs.

Mathworks/LabVIEW: Mathworks provides rapid radio prototyping solutions [6] by building a bridge between simulation at the Simulink environment and execution on heterogeneous hardware platforms, such as an FPGA interfaced with a DSP or GPP. More specifically, Simulink follows a graphical high-level modeling design approach that enables fast building and simulation of new designs based on Mathworks libraries. Simulation models are then linked to C-based DSPs or GPPs through the Mathworks RealTime Workshop tool, which enables automatic translation of Simulink simulation models into C-code. Simulink has the ability to interface with System Generator, a Xilinx DSP design tool, thus reducing the FPGA software development time. System Generator contains platform-specific sets of Xilinx FPGA IP blocks such as FFT, filters, and memory blocks that are guaranteed to exhibit equivalent cycle accuracy to the IP blocks available at Simulink. Mathworks therefore provides the required hardware abstractions to the SDR system designer to effectively integrate heterogeneous processing elements in the Simulink model-based environment and thus accelerate the transition from simulation to real-world testing. However, the available hardware APIs are constrained by the FPGA provider and technology. Another Mathworks tool that requires no prior experience with low-level FPGA or register-level (RTL) programming is the hardware-description-language (HDL) coder. The HDL coder enables automatic conversion of floating-point MATLAB or Simulink simulation models into fixed-point FPGA designs that are ready to be synthesized. Similar programming approaches are followed by the LabVIEW software tools, which target rapid development of heterogeneous multiprocessing system designs, that is, hybrid GPP-FPGA-centric architectures restricted to Ettus/National Instruments SDR platforms (Table 1).

Hardware Architectures

Ettus/NI: Ettus/National Instruments (NI) Universal Software Radio Peripherals (USRPs) are a family of heterogeneous hardware SDR platforms. They are classified into the Networked (N)-series, the Bus (B)-series, the Embedded (E)-series, and the X-series. USRP N- and X-series consist of an FPGA-based motherboard that is interfaced with a single-input single-output (SISO) daughterboard and multiple daughterboard(s) (MIMO capable), respectively, through high sample rate ADCs and DACs. Table 2 lists the frequency range (DC-6.0 GHz) and bandwidth capabilities (10–160 MHz) of Ettus daughterboards, which allow for flexible, interchangeable analog front-end circuitry for a variety of applications. FPGA-based signal processing designs are either controlled by host GPPs via external high-speed data bus connec-

GNU Radio is an open source software framework that follows a component-based design, where signal processing chains are broken into primitive components/blocks, therefore enabling code reusability and rapid block reconfiguration. Each block is assigned to a dedicated processor thread, while data exchange between blocks is achieved through shared memory buffers.

Daughterboard	Function	Low frequency	High frequency	Max RF Bandwidth per channel
UBX[1]	Transceiver	10 MHz	6 GHz	40 MHz 160 MHz
WBX[1]	Transceiver	50 MHz	2.2 GHz	40 MHz 120 MHz
CBX[1]	Transceiver	1.2 GHz	6.0 GHz	40 MHz 120 MHz
SBX[1]	Transceiver	400 MHz	4.4 GHz	40 MHz 120 MHz
XCVR	Transceiver[2]	2.4 GHz 4.9 GHz	2.5 GHz 5.9 GHz	36 MHz (RX) 48 MHz (TX)
Basic RX	Receiver	1 MHz	250 MHz	100 MHz
Basic TX	Transmitter	1 MHz	250 MHz	100 MHz
LFRX	Receiver	DC	30 MHz	30 MHz
LFTX	Transmitter	DC	30 MHz	30 MHz
DBSRX2	Receiver	800 MHz	2.3 GHz	60 MHz
TVRX2	Receiver	50 MHz	860 MHz	10 MHz

[1] The N-series family of USRPs is compatible only with 40 MHz bandwidth daughterboards, while the X-series can support both 40 MHz and 120/160 MHz daughterboards.
[2] TVRX2 supports half-duplex operation only.

Table 2. Ettus USRP daughterboards.

tions (Table 3, USRP X, N, and B-210), by soft-core processor cores implemented at the FPGA (e.g., 32-bit AeMB, MicroBlaze soft-processor cores), or by wireless-focused embedded (SoC) processors such as ARM Cortex-A9, that enable small-form-factor, portable SDR solutions (Table 4, USRP E310). Transceiver architectures such as homodyne or (super)-heterodyne vary across different daughterboards, while ADC, DAC sample rates and bit resolutions vary across different models of the USRP-series. The USRP family is compatible with the majority of software frameworks, as seen in Table 1, through the USRP-Hardware-Driver (UHD) software API that acts as a host communication driver for controlling Ettus SDRs. Furthermore, Ettus introduces a novel Network-on-Chip mechanism (RFNoC) [7], which enables integration of heterogeneous processing components into a GNU Radio flow graph by providing a standard method of consistently routing data and distributing processing throughout complex heterogeneous hardware platforms (i.e., FPGA and GPP). As a result, a researcher is able to minimize development time by integrating modular IP blocks that are executed in heterogeneous hardware. At the same time, she is able to maintain system design flexibility during runtime for performance demanding protocols by moving timing critical and computationally complex functionalities (e.g., lower medium access control and physical layer) at the FPGA, and implementing high-level control (e.g., upper medium access control) functionalities at the GPP. However, testbed requirements pertinent to specific applications entail proper selection and combination of daughterboards, USRP hardware back-end, and software.

Nutaq: Table 3 depicts two heterogeneous hardware SDR platforms provided by Nutaq, called PicoSDR (FPGA interfaced with an embedded PC or host PC) and ZeptoSDR (SoC-FPGA platform). Instead of using interchangeable analog front-ends or integrated radio chipsets, Nutaq SDRs use an FPGA mezzanine card (FMC) that features ADCs/DACs on the same compact card with analog front-end circuitry. The FMC card's transceiver follows a homodyne architecture with software selectable bandpass and baseband filters that enable easy runtime adaptation of the analog front-end to multiple communication standards. Furthermore, the separation of SDR back-end from the ADC/DAC and front-end circuitry enables hardware interoperability to deal with challenges imposed by different channel environments, on the condition that software APIs are backward compatible with different analog components. Nutaq provides an FPGA framework for embedded applications development (board-software-development kit [BSDK]) that includes custom-built hardware IPs and software APIs to enable efficient interaction with both Mathworks and GNU Radio software tools. In addition, Nutaq provides a set of custom blocks to control and handle real-time data exchange between the host or embedded processor and FPGA, thereby enabling easy synthesis and testing of heterogeneous multiprocessing designs through Simulink's model-based design approach.

Others: Tables 3 and 4 present additional tabletop and small-form-factor SDR platforms. In particular, the third revision of the wireless open access research platform (WARP) provides an FPGA-based SDR architecture with two integrated ADC/DAC and homodyne radio transceiver chipsets. The WARP platform allows expansion to four radio transceivers through FMC expansion ports, while rapid software development of new wireless designs is enabled through the WARPLab framework. WARPLab provides online and offline processing capabilities by allowing rapid physical layer prototyping with MATLAB at the host PC. Real-time online processing requires user modifications to existing WARPLab reference designs, which are implemented in MATLAB and System Generator. Epiq provides low-power portable SISO and MIMO-capable SDR platforms with proprietary IPs to reduce development time. More specifically, Maveriq's package features an FPGA interfaced with two homodyne wideband radio transceivers, an embedded Intel processor and an internal hard drive for recording and playback operations. Sidekiq and Matchstiq are small-form-factor SISO-capable alternatives that provide host and embedded multi-core processing capabilities, respectively. Per Vices offers both a SoC-based wideband SDR platform (Table 3, Crimson) equipped with four integrated homodyne radio transceivers, as well as a low-cost peripheral-component-interconnect-express (PCIe)-based alternative (Table 3, Noctar), which can be controlled and reprogrammed by custom web or Python-based GUIs. Both Epiq and Per Vices SDR platforms can be interfaced with GNU Radio; however, they currently lack the software abstractions to accelerate complex

SDR platform	Hardware	Number of TX/RX antennas	Low frequency	High frequency	Max. RF BW per IQ channel	ADC speed (MS/s, bits)	DAC speed (MS/s, bits)	Ext. data bus host connections	Maximum host throughput
Ettus USRP B-210	Xilinx Spartan 6 FPGA with integrated radio chipset	2/2	70 MHz	6 GHz	56 MHz	61.44,12	61.44,12	USB 3.0	1.96 Gb/s
Ettus USRP N-series[1]	Xilinx Spartan 3A-DSP FPGA	1/1	DC	6 GHz	40 MHz	100,14	400,16	GigE	0.8 Gb/s
Ettus USRP X-series[1]	Xilinx Kintex 7 FPGA	2/2	DC	6 GHz	160 MHz	200,14	800,16	Dual 1/10 GigE PCIex4	6.4 Gb/s
Nutaq PicoSDR[2]	Xilinx Virtex 6 FPGA	2/2	300 MHz	3.8 GHz	28 MHz	80,12	80,12	Dual GigE PCIex4	6.4 Gb/s
Nutaq ZeptoSDR	Xilinx Zynq 7020 SoC (Dual-core ARM Cortex-A9)	1/1	300 MHz	3.8 GHz	28 MHz	80,12	80,12	GigE	0.8 Gb/s
Mango WARP v3	Xilinx Virtex 6 FPGA	2/2	2.4 GHz 4.9 GHz	2.5 GHz 5.8 GHz	40 MHz	100,12	170,12	Dual GigE	0.8 Gb/s
Epiq Maveriq	Dual-core Intel Atom processor Xilinx Spartan 6 FPGA	2/2	70 MHz	6 GHz	50 MHz	50,12	50,12	GigE Dual USB	0.8 Gb/s
Epiq Sidekiq	Xilinx Spartan 6 FPGA	1/2	70 MHz	6 GHz	50 MHz	61.44,12	61.44,12	PCIex1 USB 2.0	1.6 Gb/s
Per Vices Noctar	Altera Cyclone IV FPGA	1/1	100 kHz	4.4 GHz	200 MHz	125,12	250,16	PCIex4	6.4 Gb/s
Per Vices Crimson	Altera Arria V ST SoC (Dual-core ARM Cortex-A9 MP)	4/4	100 kHz	6 GHz	322 MHz	370,16	2500,16	Dual 1/10 GigE USB	6.4 Gb/s

[1] Larger-scale antenna system setups with USRP X or N-series require additional hardware such as an OctoClock(-G), which is a clock distribution system for coherent operation of multiple SDRs under external clock reference.
[2] PicoSDR is also available in a 4 × 4 MIMO antenna configuration and a 2 × 2 embedded-PC (Quad-core i7) configuration.

Table 3. State-of-the-art tabletop SDR platforms.

algorithmic developments on either the FPGA or GPP.

Future Research Challenges and Next-Generation SDR Applications

A platform-independent system design flow can expedite rapid testing and fast standardization of novel signal processing algorithms and networking protocols. Such a high-level system design flow should also be capable of supporting the throughput and latency requirements of next-generation mobile communication protocols such as Long Term Evolution-Advanced (LTE-A) or large-scale MIMO systems, advances in the 802.11 family of networking standards as well as state-of-the-art programmable wireless networks (e.g., cognitive radio networks with dynamic spectrum access capabilities) that require fast and intelligent adaptation at all layers of the protocol stack. Existing software-hardware SDR architectures offer considerable flexibility and high performance for experimentation with new concepts at the physical layer, but lack adequate and coherently designed abstractions to define either networking protocols with cross-layer interactions across multiple layers of the protocol stack or decision making mechanisms to control such interactions [8]. Thus, state-of-the-art medium access protocols that need to comply with the IEEE 802.11 distributed coordination function (DCF) and enhanced distributed channel access (EDCA) timings, as well as routing protocols are implemented "from scratch" for different SDR testbed configurations. In this context, we describe the research challenges arising from three wireless networking testbeds (wireless terrestrial, aerial, and underwater) and argue for the potential future benefits of wireless network-specific APIs and abstractions to easily control network reconfiguration.

All-Spectrum Cross-Layer Optimized Cognitive Networks

In the context of multihop cognitive underlay networks, where primary spectrum licensees coexist with unlicensed secondary users, state-of-the-art proposals require reprogramming and optimization of the entire wireless protocol stack for every specific SDR platform. For example, the work in [9–11] implements a distributed algorithm that maximizes secondary network throughput, while at the same time avoiding interference to primary users. Implementation and deployment of the proposed cognitive algorithm at each SDR node requires complex optimization and low-latency interaction across the network and physical layers of the protocol stack to enable real-time adaptive decisions on the optimal channel waveform and routing path. Thus, implementation and deployment efforts would significantly bene-

SDR platform	Hardware	Number of TX/ RX antennas	Low freq.	High freq.	RF BW per IQ channel	ADC speed (MS/s, bits)	DAC speed (MS/s, bits)	Form factor
Ettus USRP E310	Xilinx Zynq 7020 SoC (Dual-core ARM Cortex-A9) with integrated radio chipset	2/2	70 MHz	6 GHz	56 MHz	61.44, 12	61.44, 12	133 × 68 × 26.4 mm
Epiq Matchstiq S10	Xilinx Spartan 6 FPGA Quad-core ARM Cortex-A9	1/1	70 MHz	6 GHz	50 MHz	61.44, 12	61.44, 12	114.3 × 40. × 27.9

Table 4. Portable small-form-factor SDR platforms.

fit from future SDR architectures that effectively separate decision and control from data processing so that the optimization of the protocol stack becomes SDR-platform-independent.

Software-Defined Autonomous Airborne Networks

The design and evaluation of airborne networks [12] in the context of network-centric warfare suffer from high network dynamics, such as bandwidth efficiency, link reliability, and security, across wireless nodes that are either geographically or hierarchically dispersed. Existing approaches rely on either unrealistic channel/network simulations or small-scale laboratory SDR setups with static wireless protocol designs. Thus, an SDR architecture with self-reconfigurable functionalities that can easily be controlled through a modular and declarative programming interface may enable rapid real-world deployment of novel complex communication techniques [13]. Such a radio architecture may also decouple network implementation from the underlying hardware through available wireless network APIs, and thus allow easy heterogeneous network integration and autonomous reconfiguration for next-generation airborne networks.

Real-Time Reconfigurable Underwater Acoustic Networks

Spectral efficiency of underwater acoustic networks is currently limited by the spatially and temporally variable characteristics of the underwater acoustic channel. As a result, research efforts focus on tailored protocol designs for different layers of the wireless protocol stack that are well suited to the deployment environment (e.g., ocean, lake). At the same time, existing commercially available underwater acoustic modems rely on fixed hardware designs that prevent real-time reconfiguration at any layer of the protocol stack or experimental testing of novel algorithmic developments [14, 15]. As a result, the definition of abstractions in a software defined radio architecture to handle cross-layer interactions will allow next-generation software-defined acoustic modems to decide intelligently and adapt their communication parameters to maximize spectral efficiency.

Conclusions

We have presented a comprehensive overview of the SDR design challenges in the context of next-generation programmable wireless networks, reviewed commercially available hardware and software platforms, and classified them according to their capability to promote rapid prototyping, testing, and evaluation. Finally, we have referred to three experimental testbed environments (wireless terrestrial, aerial, and underwater) and argued that although there is a wide variety of hardware and software frameworks for flexible SDR prototyping, the selection of the appropriate platform is still connected to the application and communication standard requirements. Thus, SDR-platform-independent system design flow is necessary to allow rapid prototyping, testing, evaluation, and standardization of novel wireless networking proposals.

Acknowledgments

This work was supported in part by the National Science Foundation under grants CNS-1422874, CNS-1126357, CNS-1117121, and CNS-1055945.

References

[1] J. Mitola, "Software Radio Architecture," *IEEE Commun. Mag.*, vol. 33, no. 5, May 1995, pp. 26–38.
[2] M. Mueck *et al.*, "ETSI Reconfigurable Radio Systems: Status and Future Directions on Software Defined Radio and Cognitive Radio Standards," *IEEE Commun. Mag.*, vol. 48, no. 9, Sept. 2010, pp. 78–86.
[3] T. Ulversoy, "Software Defined Radio: Challenges and Opportunities," *IEEE Commun. Surveys Tutorials*, vol. 12, no. 4, 4th qtr. 2010, pp. 531–50.
[4] D. Efstathiou, J. Fridman, and Z. Zvonar, "Recent Developments in Enabling Technologies for Software Defined Radio," *IEEE Commun. Mag.*, vol. 37, no. 8, Aug. 1999, pp. 112–17.
[5] T. Rondeau, N. McCarthy, and T. O'Shea, "SIMD Programming in GNU Radio, Maintainable und User-Friendly Algorithm Optimization with volk," *SDR, Wireless Innovation Forum Europe*, June 2012.
[6] C. Moy and R. M., "High-Level Design Methodology for Ultra-Fast Software Defined Radio Prototyping on Heterogeneous Platforms," *Advances in Electronics and Telecommun.*, vol. 1, no. 1, Apr. 2010, pp. 67–85.
[7] J. Malsbury and M. Ettus, "Simplifying FPGA Design with a Novel Network-on-chip Architecture," *Proc. 2nd Wksp. Software Radio Implementation Forum*, ser. SRIF '13, 2013, pp. 45–52.
[8] E. Demirors *et al.*, "RcUBe: Real-Time Reconfigurable Radio Framework with Self-Optimization Capabilities," *Proc. 12th IEEE Int'l. Conf. Sensing, Commun., and Networking*, June 2015.
[9] K. Gao *et al.*, "Cognitive Code-Division Channelization," *IEEE Trans. Wireless Commun.*, vol. 10, no. 4, Apr. 2011, pp. 1090–97.
[10] L. Ding *et al.*, "All-Spectrum Cognitive Networking through Joint Distributed Channelization and Routing," *IEEE Trans. Wireless Commun.*, vol. 12, no. 11, Nov. 2013, pp. 5394–5405.
[11] G. Sklivanitis *et al.*, "All-Spectrum Cognitive Channelization Around Narrowband and Wideband Primary Stations," *Proc. IEEE GLOBECOM*, Dec. 2015.
[12] K. Kwak *et al.*, "Airborne Network Evaluation: Challenges and High Fidelity Emulation Solution," *IEEE Commun. Mag.*, vol. 52, no. 10, Oct. 2014, pp. 30–36.
[13] W. Su *et al.*, "On the Capacity of Airborne MIMO Communications," *IEEE GLOBECOM*, Dec. 2012, pp. 4629–34.
[14] G. Sklivanitis *et al.*, "Receiver Configuration and Testbed Development for Underwater Cognitive Channelization," *Proc. 48th Asilomar Conf. Signals, Systems and Computers*, Nov. 2014.
[15] E. Demirors *et al.*, "Design of A Software-Defined Underwater Acoustic Modem with Real-Time Physical Layer Adaptation Capabilities," *Proc. ACM Int'l. Conf. Underwater Networks & Systems*, Nov. 2014.

Biographies

George Sklivanitis [S'11] (gsklivan@buffalo.edu) received his Diploma degree in electronic and computer engineering from the Technical University of Crete, Greece, in 2010. He is currently working toward his Ph.D. degree in electrical engineering at the State University of New York at Buffalo, and his research interests span the areas of signal processing, software-defined wireless communications and networking, cognitive radio, and underwater acoustic communications. In 2014 he was the winner of the Nutaq Software-Defined Radio Academic US National Contest and in 2015 he received the Graduate Student Excellence in Teaching award from SUNY Buffalo.

ADAM GANNON [S'14] (adamgann@buffalo.edu) received his Bachelor of Science degree in electrical engineering in 2013 from the State University of New York at Buffalo, where he is currently pursuing his Ph.D. degree in electrical engineering. His research interests include signal processing, software-defined and cognitive radio, and wireless communication in airborne and space environments.

STELLA N. BATALAMA [S'91, M'94] (batalama@buffalo.edu) received her Diploma degree in computer engineering and science from the University of Patras, Greece, in 1989 and her Ph.D. degree in electrical engineering from the University of Virginia, Charlottesville, in 1994. In 1995 she joined the Department of Electrical Engineering, State University of New York at Buffalo, where she is presently a professor. From 2009 to 2011, she served as the Associate Dean for Research of the School of Engineering and Applied Sciences, and since 2010, she has served as chair of the Electrical Engineering Department. During the summers of 1997–2002 she was visiting faculty at the U.S. Air Force Research Laboratory (AFRL), Rome, New York. From August 2003 to July 2004 she served as acting director of the AFRL Center for Integrated Transmission and Exploitation (CITE), Rome, New York. Her research interests include small-sample-support adaptive filtering and receiver design, cooperative communications, cognitive networks, underwater communications, covert communications, steganography, compressive sampling, adaptive multiuser detection, robust spread-spectrum communications, and supervised and unsupervised optimization. She was an Associate Editor for *IEEE Communications Letters* (2000–2005) and *IEEE Transactions on Communications* (2002–2008).

DIMITRIS A. PADOS [M'95, SM'15] (pados@buffalo.edu) received his Diploma degree in computer science and engineering from the University of Patras and his Ph.D. degree in electrical engineering from the University of Virginia, Charlottesville. Since August 1997, he has been with the Department of Electrical Engineering, State University of New York at Buffalo, where he currently holds the title of Clifford C. Furnas Professor of Electrical Engineering. His research interests are in the general areas of communication theory and systems and adaptive signal processing with applications to interference channels and signal waveform design, secure wireless communications, and cognitive acoustic-to-RF modems and networks.