Cognitive Radio Platforms: Wireless Open-Access Research Platform for Networks

Siddharth Gupta, Patrick Murphy, Christopher Hunter and Ashutosh Sabharwal

Abstract Cognitive radio has received considerable interest as a communication paradigm to provide dynamic spectrum allocation. However, there are still no real-time, at-scale validations of the key concepts in cognitive wireless. In this chapter we present the Wireless Open-Access Research Platform for Networks (WARPnet), a platform aimed at making cognitive radio a practical reality. The platform is designed from ground-up to be deployable and provides all the functionality needed to control and observe large-scale networks while enabling iteration on low-level algorithms. In addition, several design flows have been developed for isolated as well as integrated experiments. The platform supports both traditional research and cognitive protocol research, and thus provides uniform hardware and software frameworks to compare traditional approaches to the new clean-slate designs. During the course of the chapter, we will identify the difference between cognitive radio and traditional wireless networks to motivate the various features of the platform itself. We will also explore platform architecture to enable quick prototyping of next generation algorithms. Finally, we shall showcase WARPnet with an example cognitive protocol which exercises every part of the platform.

Siddharth Gupta
Rice University, Houston, TX e-mail: sgupta@rice.edu

Patrick Murphy
Rice University, Houston, TX e-mail: murphpo@rice.edu

Christopher Hunter
Rice University, Houston, TX e-mail: chunter@rice.edu

Ashutosh Sabharwal
Rice University, Houston, TX e-mail: ashu@rice.edu

1 Overview

Spectrum continues to be one of the most expensive and scarce resources for commercial wireless systems. As a result, constant innovation has ensured that each new generation of wireless networks (e.g cellular, Wi-Fi) is spectrally more efficient than the previous. Most current wireless systems employ one of the two forms of spectrum management: centrally managed dynamic allocation or nearly-static allocations. In the case of centrally managed dynamic allocation, active nodes can change their frequency bands but rely completely on a centralized decision maker that has a more complete view of the network. This model is employed by cellular networks, for example. In near-static allocations, nodes find a suitable frequency band for their operation and then use it for long time durations. The most dominant example is Wi-Fi access points, where the center frequency of the AP is often chosen dynamically during setup or, at best, managed at long time-scales by a central server.

Recent measurement studies [1–3] demonstrated that while most of the prime spectrum is allocated, nearly 50% is not in active use even in metropolitan areas. To enable a more agile use of spectrum, the idea of *cognitive* wireless has emerged as a strong candidate for spectrum management because it is both dynamic and distributed. As is evident from the contents of this book, there are many flavors of cognitive wireless. Thus, as the field continues to evolve, one question is repeatedly asked: “Will cognitive wireless work?”

While the above question is asked for any new technology, skepticism is especially high for cognitive wireless since it has the potential to impact policy, legislation and a vast array of existing and future wireless services. Thus, it is now commonly believed that unless there are clear technical demonstrations of the technology [4], cognitive wireless may never realize its envisioned potential. Driven by this urgent need for at-scale and at-speed verification of clean-slate designs, we describe the design and use of a flexible cognitive wireless platform in this chapter.

1.1 From Concepts to Platforms

One of the challenges in building a platform for experiments is in defining its specifications, which must not only accommodate all current ideas but also ideas which have not yet been conceived. The platform can guarantee longevity by adopting a modular architecture, with high performance components in each module. To lay out the specifications for such a platform, we start by noting that central to all cognitive wireless protocol innovations there are three functional components.

1. *Sense*: At the heart of cognitive wireless is the ability of nodes to sense the environment for ongoing traffic. Nodes can use the sensed spectral usage at multiple time-scales to adjust their behavior accordingly. Furthermore, the nodes can report their spectral measurements to a server which can execute large spatial-scale

functionality, like spectrum auctions [5], create spectral maps [6], keep nodes in the same network synchronized or simply manage nodes in the conventional manner.

2. *Adapt*: Based on the sensed environment, nodes can adapt any part of their communication stack. For example, they can adapt their physical layer [7] or medium access layer [8–10] to be more conservative if they sense a primary network with which they must not interfere.
3. *Share*: Finally, nodes can best communicate when they have synchronized spectrum knowledge at sources and destinations. Since nodes in a wireless network see a different environment (due to propagation effects of the wireless medium), their local knowledge is different. As a result, they have a different view of the network and spectrum utilization. For cognitive nodes to maximize the system wide utilization, they need to agree on their collective actions.

In addition to enabling the above basic functionality for cognitive protocol innovations, an experimentation platform which is completely programmable and deployable has to enable three additional features.

1. *Isolated and Integrated Experimentation*: The platform should enable isolated experiments, which are essential for systematically testing sub-components of a larger system. Yet at the same time, the platform design flow should enable seamless integration of sub-components into fully functional networks of nodes.
2. *Programming and Control of Deployed Networks*: To continue innovation in a deployed network of programmable nodes, it is crucial that rapid reprogramming of each node is possible. Better still, the nodes should be controllable in real-time to allow complete exploration of the design space of implemented protocols.
3. *Deep Observability of Operational Network*: Finally, each node should be able to report measurements from *any* layer to enable a deep inspection of any part of the network stack. Such deep observability is essential to provide insights into the operation of clean-slate, cross-layer designs and crucial to optimize networks for highest performance.

1.2 WARPnet

In this chapter, we describe the Wireless Open-Access Research Platform for Networks (WARPnet), which is an open-access hardware and software platform built for experimentation in at-scale deployed networks and has been engineered from the ground up for cognitive wireless networks. Below we detail how WARPnet addresses each of the six platform requirements: three for clean-slate protocol development, and three for experimentation on deployed and operational networks.

Clean-slate Protocol Requirements Mapped to Hardware Specifications

1. *Sensing Requires Agile and Wideband Radios*: Since cognitive radios must operate opportunistically over a large bandwidth, a wideband radio is essential. Fur-

thermore, RF interfaces should be modular to allow integration of future radios, which may operate on larger bandwidths centered on different frequencies. Additionally, future radios can have faster switching times, sharper front-end filters and faster scan rates. In WARPnet, we use two custom radio boards. The first is based on the a widely available commercial transceiver, MAX2829, which is dual-band and MIMO-capable. The second is based on the highly integrated AD9352, that operates in both licensed and unlicensed bands around 2.5 GHz.
2. *Adaptation Requires Full Programmability at All Layers*: To enable clean-slate designs, accommodate highly complex designs and to allow for fast time-scale in-operation adaptation, the processing core should be capable of high-performance signal processing tasks with a flexible software interface. We employ FPGAs with embedded processor cores to allow high-performance computation along with a high-bandwidth link to processors. The WARPnet FPGA Board supports up to four independent radio chains.
3. *Sharing Requires a Control Channel*: To ensure that nodes can stay synchronized, nodes should be able to collaborate with each other via a secure and orthogonal control channel. WARPnet uses multiple dedicated control interfaces, included in hardware called the Backdoor Board, which can be used for in-band and out-of-band coordination channels.

Experimentation Requirements Mapped to Platform Specifications

1. *End-to-end Design Flows*: WARPnet supports two major design flows: WARPLab, for offline processing of over-the-air data, and a real-time flow targeting standalone wireless designs. WARPLab enables prototyping of new physical layers while the real-time flow allows for network level performance measurement.
2. *Programming and Control Layer*: Once nodes are deployed in a large network, having fine-grained control of nodes and their system parameters is crucial to experimentation. WARPnet provides this functionality through the Backdoor Board.
3. *Measurement Framework*: Reviewing the performance of systems in a large scale network allows for network-level debugging. This requires the retrieval of data from every node, which is again supported through the Backdoor Board.

2 WARPnet Hardware Architecture

In this section, we discuss three major components of the WARPnet hardware platform. First is the base FPGA Board, which houses a large FPGA, multiple Radio Boards and the Backdoor Board. Next we describe the two Radio Boards, which are MIMO-capable and operate in several frequency ranges. Finally, we detail the Backdoor Board which provides multiple highly-reliable communication interfaces which can operate both over wireline infrastructure or orthogonal wireless bands. In addition, the Backdoor Board is capable of local processing and storage.

2.1 FPGA Board

The foundation of WARPnet is a Field Programmable Gate Array (FPGA) from Xilinx. It belongs to the Virtex-4 FX series [11] and is one of the largest Virtex-4 devices available on the market. FPGAs provide direct access to a large number of I/O pins, have reprogrammable logic resources and an onboard processor to implement OS functionality.

Slices	XtremeDSPs	Block RAM	PowerPCs	Ethernet MACs	RocketIO Transceivers	User I/O
42,176	160	376	2	2	20	768

Table 1 Resources available on Virtex-4 FX100

The resources available on the Virtex-4 are shown in Table 1. The flexibility of the FPGA is realized in the slice resources. Each slice is composed of two 4-input look-up tables (LUTs) and associated logic. These can implement arithmetic functions or act as distributed RAM. The slices are laid out in an array-like structure and each can be reconfigured to form larger complex systems. Low-level architectural systems can be implemented for the fine-grained control of not only the design, but also the placement and routing.

Other resources such as XtremeDSP slices, Block RAM, Ethernet MACs and RocketIO Transceivers are hardened logic for specific functions. For example, the XtremeDSP slice is an 18 x 18 bit multiplier followed by an accumulator. If a design requires the use of a multiplier, one of these will be instantiated. All these resources are spread out across the chip easing the load on the interconnect that links all the resources.

The FPGA also includes two PowerPC processors. This is extremely useful in rendering the node as a standalone device. Embedded operating systems, which can execute in the PowerPC, provide easy access to all the layers of the OSI network stack.

The hardware peripherals attached to the FPGA are shown in the block diagram in Figure 1. The FPGA provides the raw processing power to tackle real-time wireless algorithms, while the peripherals are the conduits for the data to be processed. Each peripheral plays a specific role in the complete picture of a standalone node.

2.1.1 Daughtercard Slots

To support multiple standards and especially in the evolving portions of spectrum, RF interfaces are built as modules. A very general daughtercard interface is provided, which has an open specification to encourage the community to design custom boards to suit their needs. Currently, several Rice-designed daughtercards are available, like the Radio Boards and the Analog Board (digital-to-analog converter board). We describe two Radio Boards in the next section; details of others boards can be found at WARP website [12]. The interface itself is a 124 bit parallel bus

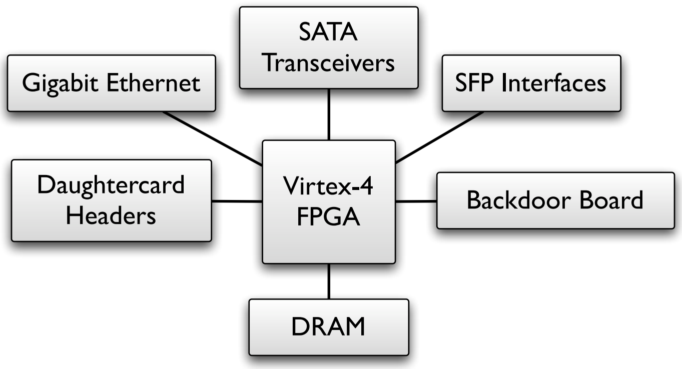

Fig. 1 Block Diagram of the WARPnet FPGA Board

connected directly to the FPGA general I/O. Additionally, the FPGA Board provides dedicated 5 V and -5 V supplies. Significant high-speed bypass on the FPGA Board reduces noise on the power planes generated by the daughtercards.

2.1.2 Gigabit Ethernet

To take advantage of the embedded Tri-mode Ethernet MAC in the Virtex-4, the board is built with the Marvell Alaska 88e1111 gigabit Ethernet transceiver. This is important in developing higher performance nodes that bridge wired and wireless domains. The hardened MAC also reduces the load on user designs as it requires considerably less logic than when implemented in fabric. The transceiver supports standard features like auto-negotiation and auto-crossover, thus easing intersection with existing Ethernet installations.

2.1.3 DRAM

As the two primary goals of the platform are deployment and experimentation, taking measurements and storing data are important. To sustain such data collection the FPGA Board has a DDR2 SO-DIMM slot that can support up to 2 GB of dynamic RAM. Xilinx tools seamlessly handle interaction with the RAM using a memory controller provided by the Virtex-4 development tools. Other than data storage, operating systems, such as Linux, can utilize this RAM for program and data.

2.1.4 Multi-Gigabit Transceivers

The MGTs are high-speed serial data links that support a variety of interfaces capable of up to 6.5 Gbps each. Two MGTs are routed to SATA jacks that enable hard drives to be connected for data storage. Two additional gigabit Ethernet ports connect via Small Form-factor Pluggable (SFP) jacks. And finally, some user designs may require two or more FPGAs to split the processing between them. Large amounts of data need to be transferred with low latency to sustain such a setup. Four HSSDC2 jacks provide this functionality. Additionally, a flexible clocking interface allows two boards to share MGT reference clocks. This can reduce the board-to-board latency down to a few clock cycles.

2.2 Radio Boards

The FPGA Board does not tackle the RF aspect of a wireless network. Its primary goal is to provide the processing power to implement wireless algorithms. Thus there is no analog conversion on the FPGA Board. The daughtercards that implement the RF chain have a digital interface to the FPGA on one end and antennas on the other. This division of roles is ideal for a cognitive radio scenario as different daughtercards orthogonal in frequency can be used by the same node, giving it access to a much wider bandwidth.

The two Radio Boards currently in use operate in two different frequency ranges. The first radio (Section 2.2.1) that operates in the 2.4 GHz and 5 GHz ISM/UNII bands. The second operates in frequencies from 2.3 GHz to 2.7 GHz, covering both licensed (especially WiMAX) and unlicensed bands.

2.2.1 Radio Board: 2.4/5 GHz

This radio daughtercard is based around a Maxim MAX2829 RF transceiver. This device does not implement any particular standard. Instead, it provides analog baseband interfaces and handles translation to RF. A block diagram of the daughtercard is shown in Figure 2.

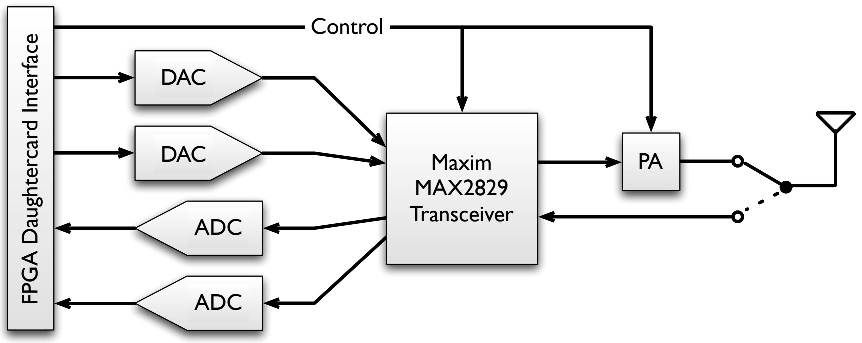

Fig. 2 Block Diagram of Radio Board based on MAX2829 Transceiver

There are three chains from the daughtercard header to the radio: transmit, receive and control.

The transmit chain passes through a 16-bit dual input Analog Devices AD9777 digital-to-analog converter. The dual channel converters reduces imbalances between the real and imaginary chains of the transmitted signal. The MAX2829 also includes adjustable baseband transmit filters.

The receive chain employs similar dual 14-bit analog-to-digital (AD9248) converters. In order to fill the dynamic range of the received signal, the MAX2829 has two stages of signal amplification, at RF and at baseband. Both stages are controlled by user designs in the FPGA, allowing flexible and fast automatic gain control (AGC).

The FPGA can control the various parameters of the radio, such as transmit channel and power, input and output filters, etc., via an SPI interface. This is a direct connection between the transceiver and the host FPGA.

The RF front-end encompasses the power amplifier on the output and the antenna switch that selects between the transmit and receive chain. Both are controlled by the FPGA.

2.2.2 Radio Board: 2.5 GHz

The second radio daughtercard has a similar structure but includes a highly integrated RF transceiver. It is based around Analog Devices' AD9352 chip. The block diagram is shown in Figure 3.

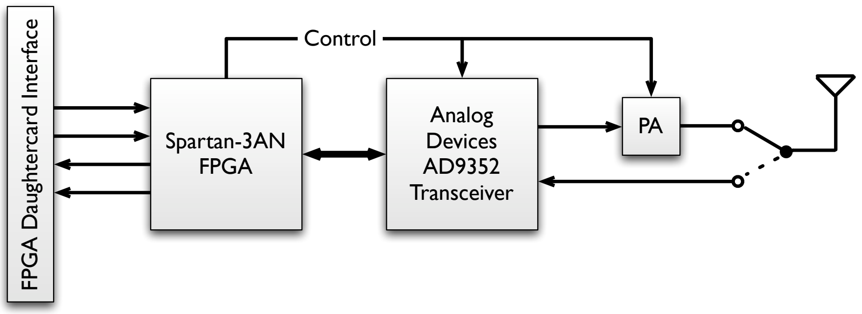
FPGA Daughtercard Interface
Control
Spartan-3AN FPGA
Analog Devices AD9352 Transceiver
PA

Fig. 3 Block Diagram of Radio Board based on AD9352 Transceiver

The AD9352 integrates the analog-to-digital and digital-to-analog converters along with the RF transceiver, filters and amplifiers. The data interface to the chip is a dual-datarate bidirectional bus. The Spartan-3AN FPGA in the chain is used to translate the data to a single-datarate unidirectional interface back to the Virtex-4. Similar to the previous radio, the AD9352 has an extensive register bank to control its various parameters. While ordinarily the Virtex-4 would have to perform the tasks, the presence of the Spartan-3AN splits the processing load. Much of fine-grained radio control can be handled by the local processor while the PowerPC in the Virtex-4 can continue to perform MAC layer functionality.

2.3 Backdoor Board

The third part of the platform is the Backdoor Board. As mentioned, it serves two primary purposes: (1) a dedicated control channel and (2) the control and monitoring hub. The control channel problem can be addressed in many ways. One could add an additional radio to the system and utilize the same band as the primary network or use a dedicated control channel orthogonal to the experimental network itself. WARPnet supports both, as up to four radios can be connected to the FPGA Board. The Backdoor Board provides dedicated control channels that are also available to the FPGA. Next, having the ability to control every board in a network from a central server is important in scheduling experiments and observing network-level performance. Physical access to the boards may be difficult in deployed networks and hence remote reprogramming is an integral part of WARPnet. All the above functionality is provided by the Backdoor Board.

The block diagram of the Backdoor Board is shown in Figure 4. There are two main processors on the board each serving a different purpose. The first is a Linux SoC, the Axis ETRAX 100LX, and the second is a Xilinx Spartan-3AN FPGA.

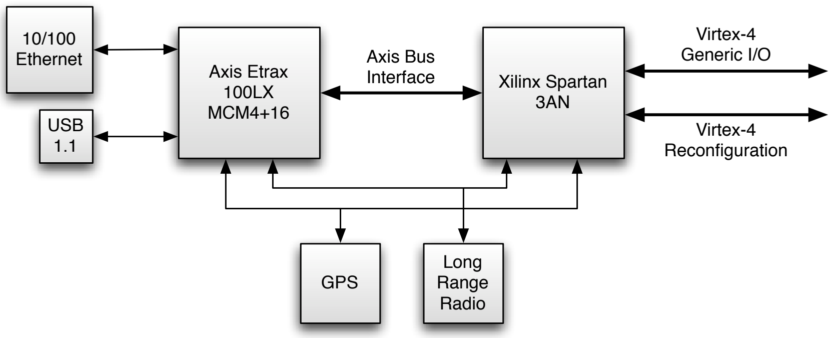

Fig. 4 Block Diagram of Backdoor Board

2.3.1 Axis Etrax 100LX Multi-Chip Module

The design of the board is centered around an embedded Linux device server, the Axis Etrax 100LX MCM4+16 [13]. This is a multi-chip module with four major components: an Axis Etrax 100LX processor, 4 MB flash memory, 16 MB RAM and an Ethernet transceiver (discussed in *Backdoor Interfaces*). When the chip is powered up, it automatically boots from a local flash image. This has two major benefits: it eliminates the need for users to reprogram the device, and eases deployment, since a stable image can be saved and used as many times as required.

The Etrax uses a custom Linux distribution. It includes all the tools that are expected from an operating system that are useful for debugging and development. Logging into its terminal is an option with an onboard serial port. It also reduces design cycle time as standard C/C++ programs can be used and no additional language needs to be learned. Additionally, the development environment is open source.

2.3.2 Xilinx Spartan-3AN FPGA

Connecting the Axis Etrax chip to the main Virtex-4 on the FPGA Board is a Xilinx Spartan-3AN (XC3S700AN) FPGA [14]. This is a reconfigurable device with plenty of logic blocks and embedded multipliers. Unlike the Virtex-series it has no embedded processor but a soft processor (e.g. MicroBlaze) can be implemented using the Xilinx design flow. Since it is programmable, it is ideal for data translation. It seamlessly interfaces the Etrax and Virtex-4 devices. Any data transfer from either side is a simple bus transaction through the Spartan.

2.3.3 Backdoor Interfaces

We have implemented three backdoor communication interfaces. The type of deployment dictates the interface that is used in the network.

Consider the scenario where the deployment is within a building and wired Ethernet connections are available at each node. Here a wired backdoor link would be ideal. To fulfill this scenario a Ethernet interface is provided giving direct access to the Etrax chip. The physical layer is part of the Etrax multi-chip module.

The other and more flexible implementation involves a long-range radio. This is tailored to outdoor deployments that spread over larger areas without wired connectivity at every node. The implementation uses a 900 MHz radio that can communicate over long distances with a highly reliable but low data rate link. This avoids interference between the backdoor interface and the experimental wireless links.

The other implementations utilize the USB or Ethernet interfaces. A Wi-Fi card connected to the USB 1.1 port provides similar functionality as a laptop's wireless interface. The user must ensure that the channel usage is orthogonal to the experimental network. This solution is in between the long range and wired connection in terms of distance and speed. Any other external device connecting over USB or Ethernet can be utilized to provide the backdoor functionality.

2.3.4 Other Peripherals

Other than the main backdoor links, this board also has a GPS receiver. This is extremely useful for MAC layer algorithms as it can provide information such as topology and precise globally-synchronized time. To perform scheduled access algorithms all the nodes in the system need a common time reference. The GPS time is accurate enough for this application [15]. This is also useful when the user is trying to conduct large-scale experiments where events need to occur at certain precise times.

3 WARPnet Experimentation Framework

Figure 5 shows an example deployment of WARPnet. Each WARPnet node implements a full cognitive radio protocol stack and is deployable. Every node is also connected to a central server that can monitor the system characteristics and other parameters. The centralization provides the experimenter with a system-wide view of network performance. In addition, it allows arbitrary manipulation of node parameters. Therefore each node has the infrastructure for cognitive protocols under test, and network control and measurement for managing the experiments.

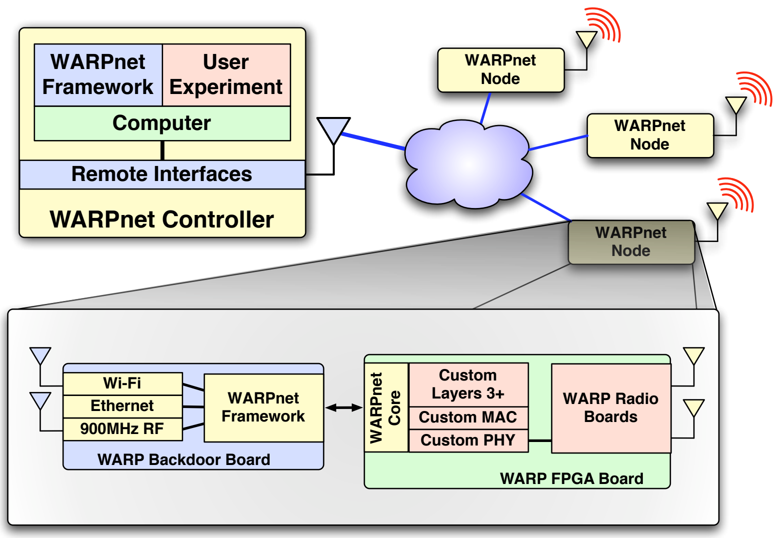

Fig. 5 Deployed Network of WARPnet Nodes

3.1 Design Flows

Now that we have described the hardware itself, it is important to understand how all the pieces play together. Each WARPnet node consists of an FPGA Board, up to four Radio Boards and a Backdoor Board. An example kit can be seen in Figure 6.

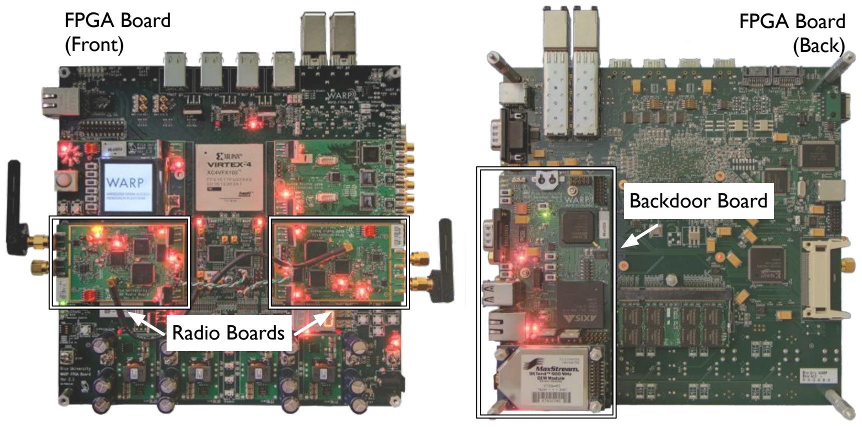

Fig. 6 Front and Back of a WARPnet Node

As mentioned previously, the Radio Boards have a direct bus connection to the FPGA. The physical layer design has access to this bus and hence to the data the radio produces. Using the received waveforms, the physical layer can do the baseband processing, from the receive decimation filters to the decoding and demodulation. Once complete, the MAC layer in the PowerPC can direct the data to the appropriate interface. The Backdoor Board also has a direct connection to the FPGA. Custom protocols implemented in the Spartan-3AN and the Virtex-4 are needed to maximally utilize the interface and share information between the Etrax processor and the PowerPC.

Revisiting the hardware requirements for cognitive protocols, we have a highly reconfigurable processor that is capable of implementing complex and novel physical and MAC layers. It has access to two different radio interfaces that can sense both licensed and unlicensed spectrum. The presence of a dedicated control channel completes the loop of cognitive radio to share information with other nodes.

Designing for the platform can be a daunting task with the number of devices in the chain. However, some serve as translators to connect between processor domains. The Spartan-3AN on the Backdoor Board connects the bus of the Etrax device and the FPGA PowerPC (each adhering to its own standard) by allowing each to appear as a peripheral to the other. Similarly, the Spartan-3AN on the 2.5 GHz Radio Board interfaces the radio's digital interface with the Virtex-4. The radio has a dual data rate bidirectional port for the receive and transmit waveforms. The Spartan FPGA splits the data into four individual single data rate unidirectional buses. This greatly simplifies user designs in the Virtex-4.

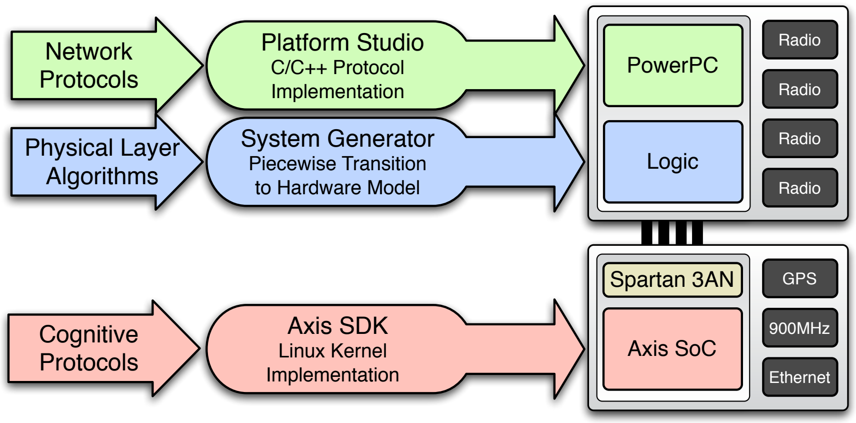

Fig. 7 Design Flows

The three processors of interest are the FPGA logic, the PowerPC and the Etrax SoC. The role of each device can be easily delineated based on the design environment, as illustrated in Figure 7. The FPGA with its high speed low-level hardware logic is ideal for implementing physical layers. It can support extremely complex designs in HDL while the Xilinx tools help with placement and routing. MAC layers are traditionally written in C/C++. Thus the PowerPCs in the FPGA, with direct access to the physical layer in the FPGA fabric, can be utilized for MAC layer design. And the Etrax device, also programmed in C/C++, has hooks into the physical and MAC layer and can therefore provide the functionality of the dedicated control channel and the monitoring network.

Not every user has expertise in each of the above design environments but would like to research protocols in only a particular area. To meet this demand, standard physical and MAC layers have been provided and available online [12]. These enable quick iteration in a sub-field without having to recreate the entire system from scratch for each new protocol.

3.2 WARPnet Experiment Framework

Having looked at the hardware resources, we can see that there are plenty of resources to implement a cognitive algorithm. The next issue to consider is in-system validation. How do these algorithms behave once they are in a large network with several nodes interacting? Will the addition of a once-per-second control packet severely affect performance in another region of the network?

Consider the example of a network of nodes deployed in a large building as a new mesh-based physical layer algorithm is tested. As the interference ranges of the nodes is discovered and bottlenecks are resolved, we would like to iterate on system parameters to maximize system performance. At that point, running experiments with the nodes while iterating through parameters will help us debug the algorithm and stress it to the limit. Once tested, we can make changes but downloading the new firmware to the node would require physical access to it as the CompactFlash card connected to the board that stores the bitstream needs to be updated. In a large network gaining physical access to all nodes is non-trivial. Thus, ability to download new firmware, modify system parameters and run experiments remotely is important to maintaining research deployments.

To address the issue of remote management, the platform has an experiment management framework. This framework will handle the control and observation of nodes, such as firmware upgrades, system performance measurements and data storage for packet traces, all using the current platform without any other hardware additions. The flexibility of the Backdoor Board is leveraged in the development of the framework. A central server connects to all the nodes over the backdoor interfaces. In addition to collecting data, the server can also send firmware upgrades.

The framework can tap into the complete network stack, and primarily uses the Etrax device to do so. Simple cores are required in the Virtex-4 to complete the link.

3.3 Remote Control

A critical requirement in setting up an experimental network is control. Ideally, each node can be controlled from a single location. WARPnet provides this remote control functionality with the help of the Backdoor Board. Any of the three interfaces (Ethernet, Wi-Fi, 900 MHz radio) can be configured as remote control interfaces.

The system is structured such that there is a shared memory interface between the Backdoor Board and the Virtex-4 FPGA, as illustrated in Figure 8. As the bus standards of the two devices are not the same, the Spartan-3AN is required to translate the communication between the processors. The shared memory is located in the Virtex-4 where both the bus and custom logic cores can access it. The Etrax device sees the shared memory as a peripheral on its bus with a particular memory address that it can access directly. For the data in the shared memory to be meaningful, the Virtex-4 and Etrax must agree on types of parameters stored and their corresponding locations.

Keeping the shared memory interface in mind, there are several levels of control that can be exercised. At the lowest level, this control can be applied to parameters of an algorithm. The physical and MAC layers can read their system parameters, such as modulation rate, transmit power and carrier frequency, from the shared memory. The Backdoor Board can push values from either the central controller or from a local cognitive protocol. At the highest level, the central controller can restart a node if it has fallen into a bad state. Such high-level functions do not communicate

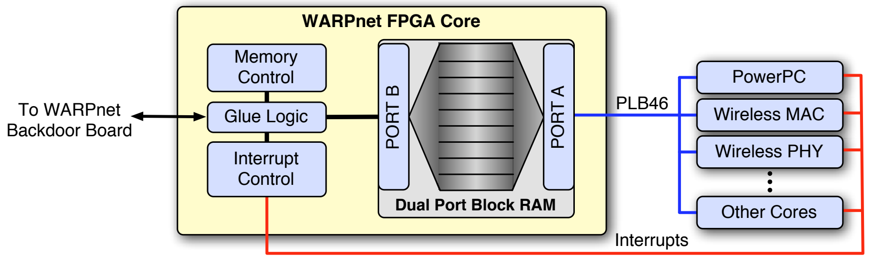

Fig. 8 Shared Memory Interface for Control and Observation Data

with the Virtex-4 FPGA; but instead take control of power circuits on the FPGA Board itself.

The capacity of the control channel will depend on the interface used and the size of the network deployed. If the nodes are connected with wired Ethernet, the control information can be sent as often as Ethernet can support, while if using the 900 MHz radio the data is limited to 115200 Kbps.

3.4 Remote Observation

Building on the control framework, we can also gather data from each node in a large system. The Etrax device can pull data out of the shared memory interface and send it to the central controller. For example at a low level, AGC gain values for every receive packet can be extracted to reveal channel variations. At a higher level, users can collect the total good and bad packets counts received at every node. Observations relating to node state, such as GPS location, are also accessible by the central controller.

Just like control information, the observations are limited by the scale and nature of the deployed network. If the Backdoor Boards are wired together with Ethernet, receiving packet-level data is feasible. For slower interfaces the user is restricted to long-term statistics.

Even though some observations would be useful to receive in real-time, most can be analyzed at the conclusion of the experiment. Thus each node is equipped to store data locally. A hard drive can also be connected to each board using a standard SATA interface, so if the user wishes to collect packet traces, representative channels or actual received samples, these can be stored on the external drive. Once an experiment is complete, the data can be streamed back over the backdoor network or the hard drives can be gathered at a later date.

3.5 Remote Reprogramming

Controlling an existing algorithm and collecting performance data can point to limitations or bugs in the algorithm. As the goal of the platform is to allow users to experiment with new technologies, implementations will rarely be perfect. Hence iterating on user designs needs to be made part of the monitoring network. For this purpose the platform builds in a methodology to reprogram nodes in a network from the central location without the need to physically access every board.

The Etrax device has control of the slave serial reconfiguration input for the Virtex-4 FPGA. The Etrax streams the received bitstream to the FPGA configuration port. Alternatively, the bitstream can be stored on the CompactFlash card connected to the FPGA. This allows the FPGA to be restarted from that bitstream at any time.

The process of remotely reprogramming the nodes is similar to what is used in some sensor networks [16, 17]. Both the 900 MHz radio and Wi-Fi support multihopping, which eases the reprogramming of entire networks.

3.6 Experiment Scheduling

Often one data point on the failure of an algorithm is not enough to pinpoint its cause. Several experiments with varied topologies and system settings must be run to determine the cause. After every experiment the data must be collated and analyzed. This can be automated, and the framework builds in the ability to manage and schedule large experiments that cycle through system parameters. This allows for checking corner cases and behavior not anticipated during the design.

The framework can execute experiments on a per-node basis. The parameters for every node as well as the traffic patterns can be adjusted prior to the start of the experiment. The GPS provides an accurate time-basis for the nodes to start executing their particular function and run for as long as required. Finally, traffic traces, errors, system performance numbers and other data can be collected by the central node on completion. This can be repeated for various patterns and parameters.

4 System Deployment and Example Experiment

In this section, we show how the WARPnet architecture enables rapid prototyping of clean-slate cognitive network protocols. Due to space limitations, only a basic form of dynamic spectrum access is demonstrated, where nodes coordinate to improve performance. Further results from extensive experimentation will be reported elsewhere.

4.1 Problem Setup

Let us consider a scenario with licensed and unlicensed users. The primary licensed users have the rights to and hence priority in the frequency bands. Such networks are generally legacy systems that do not expect other users to exist in the network and therefore have no active avoidance mechanisms. The secondary unlicensed users, on the other hand, is trying to "steal" as much bandwidth as possible without affecting the primary links. Unlike the primary, they are intelligent and spectrum aware, with all the abilities discussed above.

In this particular example, the secondary network is a two node link attempting to communicate at a constant traffic rate. Examples of such a scenario could be streaming audio or video, etc. The nodes must pick a channel that can sustain their rate as well as avoid interfering with the primary flow. In addition to a wireless stack, each node has a dedicated control channel available to it. The problem of just picking a wireless channel is simplistic but can be easily extended to considerably more complex cognitive protocols.

The solution we implement uses the FPGA and PowerPC for the custom wireless link built with physical and MAC layers, and the Backdoor interfaces to coordinate on selecting the best channel (Figure 9).

4.2 System Implementation

There are three main pieces of the implementation to consider: physical layer, MAC layer and cognitive protocol.

The physical layer on the boards is a custom 64-subcarrier OFDM system. As mentioned in the hardware description of WARPnet, the Virtex-4 FPGA has direct access to the digital waveforms from the radio. Hence, all the blocks of the OFDM algorithm are implemented in the FPGA: packet detection, automatic gain control, decimation, FFT and demodulation. This real-time implementation is developed for a random access system. The packet detector and automatic gain control blocks are needed to allow the node to receive packets from any transmitter, near or far. The physical layer has a large packet buffer as the interface to the MAC layer. As each received packet is processed, it is stored and the MAC layer is notified of its presence. Finally, to simplify the link to the MAC layer, there are PowerPC drivers that can access all the parameters of the physical layer.

The MAC layer implementation largely resides in the PowerPC of the FPGA. It is a carrier-sense, collision avoidance medium access protocol. The nodes are in wired-to-wireless mode; they forward any data received over the air to Ethernet and vice-versa. Hence the MAC does not have to interface with any higher layers but instead determine ideal times for transmission and copy data to the Ethernet.

For efficient cognitive protocol performance, the parameters of the system that cause a secondary user to modify its behavior must be determined. We can take advantage of the fact that the nodes are sending data at a constant rate. As the rate

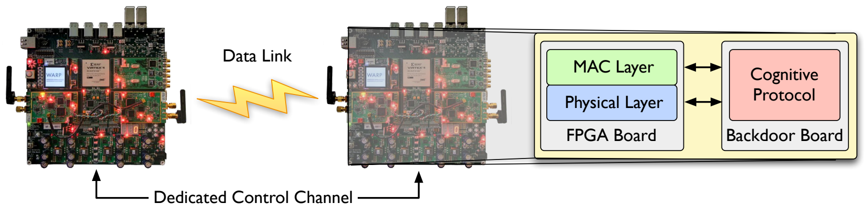

Fig. 9 Experimental Setup of Secondary Network

is fixed, the expected number of packets over the link can be determined. Hence one metric for link failure is if the achieved throughput falls too low. As the goal is to pick the best channel, knowing the quality of the alternate channel is important. Therefore, the received signal strength of alternate channels is tracked. This is a good measure of the utilization of the frequency. These two metrics must then be used to decide whether the link must switch its frequency.

To provide the sensing functionality, there is one major addition to the physical layer. A second radio interface is enabled to monitor other channels on the network. For this example, if the primary interface is using channel 3 for data transmission, the second interface will monitor channel 9. This is the first key trait of cognitive radio: sensing the environment. Therefore, a custom core is added to the physical layer in the FPGA that averages the received signal strength (RSSI) of the alternate channel over one second. The packet throughput data is available at the MAC layer and this is stored in the memory shared with the Backdoor Board.

The cognitive protocol itself is implemented on the Backdoor Board as a user application that makes a decision on link performance. The real-time link data (good and bad packets received) is captured from the MAC layer and average RSSI values are read from the physical layer. If the number of received good packets is above a threshold, the channel is good and no change is needed. However, if less than a certain number of good packets are received there is possibility that the primary user is transmitting and affecting the secondary throughput. But the secondary link can switch to the alternate channel only if it is under-utilized. The averaged RSSI value gives the protocol this information. Again if that is below a threshold, the alternate channel is good and node can switch. This is the second key trait of cognitive radio: adapting to changing conditions.

Once it has been determined that the alternate channel is better that the current one, the transmitter must be notified. Using the backdoor interface, the new channel value is communicated to the transmitter and both nodes can switch the frequency channel. This illustrates cognitive radio's last trait: sharing local information with other nodes.

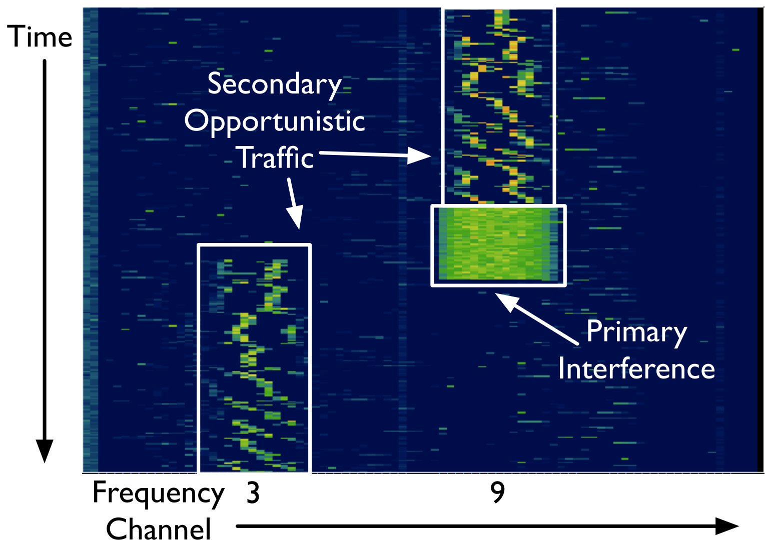

Fig. 10 Spectrogram of Channel Usage During Cognitive Experiment

4.3 System Performance

The system is implemented on the WARPnet hardware described previously. All the nodes are real-time systems and a constant rate link is setup between the two secondary nodes. The presence of the primary user is simulated as wideband noise injected onto the channel with the help of a signal generator. This accurately models the primary as it does not carrier sense or expect any other user on the air, but transmits its data as needed. The behavior of the primary and secondary links can be observed by sensing the environment with a third device. In Figure 10 we see the real-time channel trace from that third device. The graph is a spectrogram of the wireless channel. Along the y-axis is time, x-axis is frequency while the brightness of each point determines the power of the transmission. The transmissions of the secondary user can be clearly distinguished as the constant traffic on the two available channels. As soon as the primary user started transmitting, the secondary switched its channel. The speed at which the secondary reacts and the agility of the protocol are run-time parameters which can be customized by the experimenter.

We have a seen a complete cognitive protocol implementation on WARPnet. The hardware and software frameworks that were developed give us the real opportunity to try novel protocols in short amounts of time. We were able to go from idea to implementation in just a couple of days by leveraging on all the open-source designs available [12] for the platform. The experiment demonstrates all the key features of a cognitive system and will serve as the starting point for more complex implementations.

5 Conclusion

In this chapter we showcased a wireless prototyping platform that not only meets the needs of cognitive radio but provides all the features required of large-scale deployed experimental networks. We noted the differences between traditional wireless and cognitive radio networks, identifying specific features that the cognitive platform must provide. We described the hardware platform itself and the design flows for building experiments. Finally, we implemented a basic cognitive protocol on the hardware in order to demonstrate the full capabilities of the platform.

References

[1] S. Geirhofer, L. Tong, and B. M. Sadler, "Dynamic spectrum access in the time domain: Modeling and exploiting white space," in *IEEE Communications Magazine*, May 2007.

[2] D. Cabric, A. Tkachenko, and R. W. Broderson, "Experimental study of spectrum sensing based on energy detection and network cooperation," in *TAPAS*, 2006.

[3] C. Cordeiro, K. Challapali, K. Birru, and S. Shankar, "IEEE 802.22: the first worldwide wireless standard based on cognitive radios," in *New Frontiers in Dynamic Spectrum Access Networks*, 2005.

[4] J. M. Peha, "Sharing spectrum through spectrum policy reform and cognitive radio," in *Proceedings of the IEEE*, April 2009.

[5] X. Zhou, S. Gandhi, S. Suri, and H. Zheng, "eBay in the sky: Strategy-proof wireless spectrum auctions," in *International Conference on Mobile Computing and Networking*, 2008.

[6] Y. Zhao, L. Morales, J. Gaeddert, K. K. Bae, J.-S. Um, and J. H. Reed, "Applying radio environment maps to cognitive wireless regional area networks," in *New Frontiers in Dynamic Spectrum Access Networks*, 2007.

[7] D. Dash and A. Sabharwal, "Secondary transmission profile for a single-band cognitive interference channel," in *Asilomar Annual Conference on Signals, Systems and Computers*, 2008.

[8] Q. Zhao, L. Tong, A. Swami, and Y. Chen, "Decentralized cognitive MAC for opportunistic spectrum access in ad hoc networks: A POMDP framework," in *IEEE Journal On Selected Areas in Communications*, April 2007.

[9] S. Novich and A. Sabharwal, "Stealing from an ongoing flow," in *Mobile Adhoc Networking and Computing*, 2009.

[10] C. Cordeiro and K. Challapali, "C-MAC: A cognitive MAC protocol for multichannel wireless networks," in *New Frontiers in Dynamic Spectrum Access Networks*, 2007.

[11] "Xilinx Virtex-4 FPGA," http://xilinx.com/virtex4.

[12] "WARPnet online repository," http://warp.rice.edu/trac.

[13] "Axis Etrax 100LX MCM4+16," http://www.axis.com/products/dev_etrax_100lx_mcm/.

[14] "Xilinx Spartan-3AN FPGA," http://www.xilinx.com/spartan3an.
[15] J. Wang, Y. Fang, and D. Wu, "SYN-DMAC: a directional MAC protocol for ad-hoc networks with synchronization," in *Military Communications Conference*, 2005.
[16] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *ACM SenSys*, 2004.
[17] S. S. Kulkarni and L. Wang, "MNP: Multihop network programming service for sensor nodes," in *IEEE ICDCS*, 2005.