KUAR: A Flexible Software-Defined Radio Development Platform

G. J. Minden, J. B. Evans, L. Searl, D. DePardo, V. R. Petty, R. Rajbanshi, T. Newman, Q. Chen, F. Weidling, J. Guffey, D. Datla, B. Barker, M. Peck, B. Cordill, A. M. Wyglinski and A. Agah
Information Technology and Telecommunications Center
The University of Kansas, Lawrence, KS 66045
Email: {gminden,evans,alexw}@ittc.ku.edu

***Abstract*— In this paper, we present the details of a portable, powerful, and flexible software-defined radio development platform called the Kansas University Agile Radio (KUAR). The primary purpose of the KUAR is to enable advanced research in the areas of wireless radio networks, dynamic spectrum access, and cognitive radios. The KUAR hardware implementation and software architecture are discussed in detail. Radio configurations and applications are presented. Future research made possible by this flexible platform is also discussed.**

I. INTRODUCTION

Given the public and private sectors' insatiable desire for additional wireless bandwidth, new solutions are required to help address the burgeoning problem of "spectrum scarcity". Measurement studies have shown that licensed spectrum is relatively unused across time and frequency [1, 2]. This is particularly true (on a per market basis) for TV spectrum. A new concept has been proposed which would enable unlicensed devices to access this unused spectrum, thus solving the scarcity issue and spurring innovation in the wireless industry. Nevertheless, current government regulations prohibit unlicensed transmissions in these bands, constraining them instead to several heavily populated and interference-prone regions of spectrum. Despite these current prohibitions, the figurative "regulatory wheels" are beginning to roll towards the possibility of opening certain frequency bands to unlicensed usage. The Federal Communications Commission (FCC) has already commenced work on the concept of unlicensed users "borrowing" spectrum from spectrum licensees [3, 4]. Simultaneously, a similar approach has also been promoted through the Defense Advanced Research Projects Agency (DARPA) Next Generation (XG) program [5]. This form of spectrum allocation is known as dynamic spectrum access (DSA).

With the rapid evolution of microelectronics, wireless transceivers are becoming more versatile, powerful, and portable. This has enabled the development of software-defined radio (SDR) technology, where the radio transceivers perform the baseband processing entirely in software. The ease and speed of programming baseband operations in a SDR makes this technology a prime candidate for DSA networks. Software defined radios also represent an advancement in the rapid prototyping, testing and deployment of new radio hardware and communications systems. New modulation schemes or coding techniques can be rapidly implemented and tested without building expensive custom hardware. SDR system software can be designed to interface with communications system design programs, thus enabling designers to rapidly move from simulation to implementation.

SDR units that can rapidly reconfigure operating parameters due to changing requirements and conditions at the physical, network, and/or application layers of the system are known as cognitive radios [6]. With recent developments in cognitive radio technology, it is becoming possible for these systems to simultaneously respect the rights of incumbent license holders while providing additional flexibility and access to spectrum. When implementing a cognitive radio platform, there are several design goals to consider that would greatly assist research in this area, namely: (1) a very flexible RF front-end that can support both wide transmission bandwidths and a large center frequency range, (2) a self-contained, small form factor radio unit to enable portability, (3) powerful on-board digital processing to support a variety of cognitive functions and radio operations, and (4) a low cost build cycle to easily facilitate broad distribution of the radio units to other researchers within the community. Although several cognitive radio prototypes and testbeds have been implemented [7–10], each of these systems employs a set of design criteria that prevents them from realizing the aforementioned four design goals.

This work was supported in part by the National Science Foundation (NSF) under grants ANI-0230786 and ANI-0335272, and both the Defense Advanced Research Projects Agency (DARPA) and the Department of the Interior National Business Center via grant NBCHC050166. This work has been submitted in parts to the IEEE Communications Magazine Feature Topic on Cognitive Radios for Dynamic Spectrum Access.

In this paper, we present the Kansas University Agile Radio (KUAR) platform, a low cost, flexible RF, small form factor SDR implementation that is both portable and computationally powerful. The KUAR satisfies all four design goals discussed previously, making it an excellent platform to conduct cognitive radio and DSA network research. Section II provides an overview of the platform and its constituent parts. Section III covers the KUAR hardware components and discusses the design flexibility they provide. Section IV addresses the software architecture and tools provided to program the KUAR. A sensible design workflow that integrates design and simulation tools with the KUAR software architecture is discussed in Section V. Issues regarding configuration and adaptation specific to cognitive radio functionality are addressed in Section VI. Section VII enumerates current research projects related to the KUAR platform. Section VIII concludes the paper with a summary and the direction of future research regarding the platform.

II. KUAR OVERVIEW

The KUAR is a software-defined radio specifically designed to address the needs of wireless networking and radio frequency (RF) research. It features a modular design consisting of separate power supply, digital processing, and RF sections. The current version of the radio operates in the 5 - 6 GHz band and is capable of implementing numerous modulation algorithms, media access control (MAC) protocols, and adaptation mechanisms. As shown in Fig. 1, the KUAR consists of five major sub-systems on three printed circuit boards: a power supply, a control processor host (CPH), a digital board (DB) with a programmable signal processor, A/D, and D/A converters, an RF transceiver, and active antennas. With the exception of the antennas, the sub-systems are contained within a shielded box approximately 7 inches tall, 3 inches wide, and 6 inches deep, or roughly the size of a good dictionary. The antennas are separated to reduce interference and other antenna configurations are possible. The KUAR RF transmit and receive ports are standard SMA connectors, which allow the use of a variety of commercial and prototype antenna configurations.

A modular design was chosen so that sections of the KUAR platform can interoperate with other, third party prototypes for the purposes of experimentation and testing. For example, the KUAR CPH and DB could be connected to other RF transmitters or receivers in order to allow investigation of other frequency ranges or channel parameters (Fig. 2). Alternatively, the KUAR active antennas and RF transceiver could be used with existing signal processing systems.

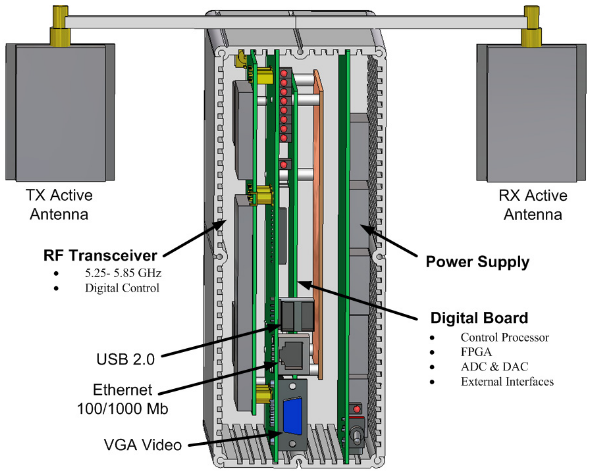


Fig. 1. KUAR Radio

III. KUAR HARDWARE

A. Power Board

The current version of the KUAR power board is designed to run off of a 12V battery and converts that to six independent supply voltages which supply power to the RF, Digital and CPH boards as well as the active antennas. The power board also features current sensors on each supply voltage that allows experimenters to evaluate the power efficiency of various communications systems.

B. Digital Board

The CPH of the KUAR is an embedded PC built on an industry standard ComExpress form factor and contains a 1.4 GHz Pentium M, 1 GB DDR2 SDRAM, and an 8 GB MicroDisk for storage. It connects to the Digital Processing board through a PCI Express connection. The CPH runs Linux and provides USB 2.0, VGA, PCI Express and Gigabit Ethernet (10/100/1000 Mbps) connections. The processing power of the CPH allows for significant signal processing, as well as rapid radio reconfiguration based on performance measurements of the current RF environment or the physical, network and application layers of the communications system. The addition of VGA and USB allow this mobile experimental platform to be used as a standard PC while on the test bench or in the field, decreasing the amount of equipment necessary for testing and experimentation.

The majority of digital communications components and digital signal processing operations in the KUAR are implemented in a Xilinx field-programmable gate array (FPGA) . The current KUAR hardware employs a Xilinx Virtex II Pro P30 FPGA, which possesses 30,816 logic cells, two PowerPC 405 cores, and operates at up to 350 MHz. The FPGA is programmed using command line utilities and software libraries available under Linux on

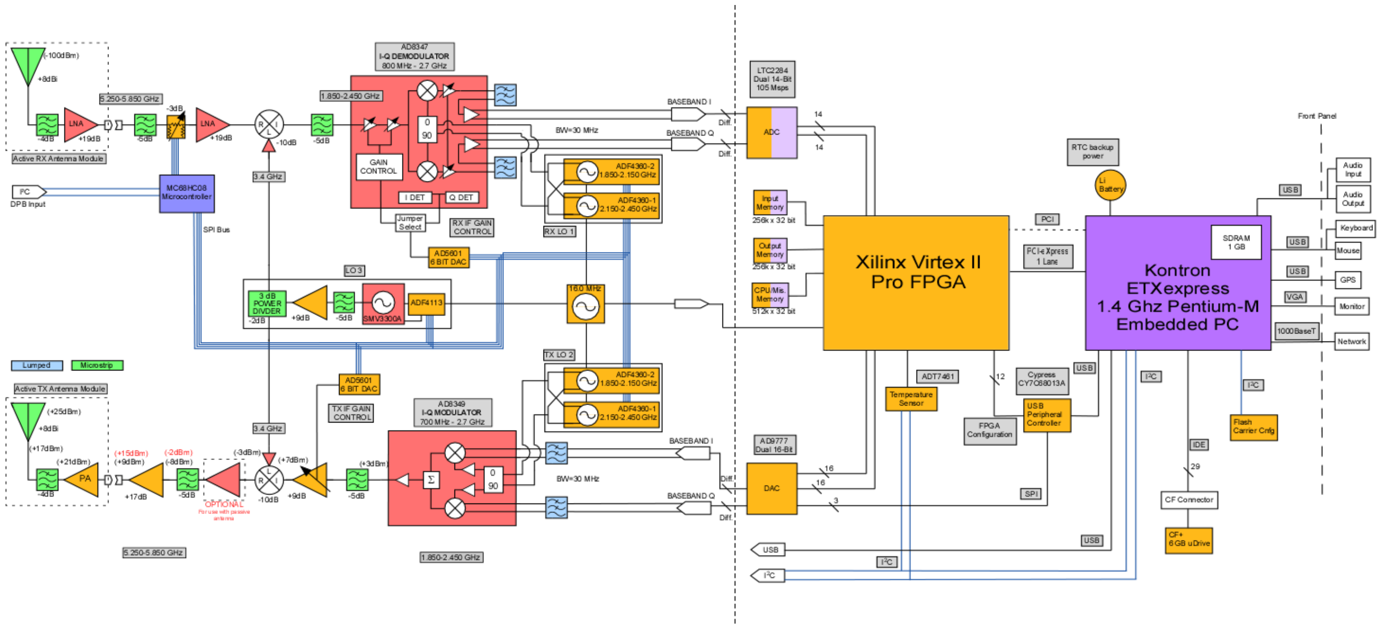

Xilinx Virtex II Pro FPGA
Kontron ETXexpress 1.4 Ghz Pentium-M Embedded PC
AD8347 I-Q DEMODULATOR 800 MHz - 2.7 GHz
AD8349 I-Q MODULATOR 700 MHz - 2.7 GHz
LTC2284 Dual 14-Bit 105 Msps
AD9777 Dual 16-Bit
ADC
DAC
Active RX Antenna Module
Active TX Antenna Module
MC68HC08 Microcontroller
USB Peripheral Controller
Front Panel

Fig. 2. KUAR System Diagram

the CPH. Programming and configuration commands are sent across the USB 2.0 bus to a Cypress USB peripheral controller. The controller sends the configuration data to the FPGA via a parallel bus connected to the configuration pins of the FPGA. The configuration data rate is at 48 Megabytes per second (MBps) which is near the 50 MBps limit of the FPGA. There are three possible physical interfaces between the CPH and the FPGA. The fastest is a connection over the PCI Express bus, which provides data rates around 250 MBps, but it takes around 7,000 logic slices to implement in the FPGA. To save logic space, a PCI bus connection may also be used. It provides a transfer rate of 130 MBps and only uses 500 logic slices. Finally, data can be transferred to and from the FPGA using the USB peripheral controller. This provides the slowest data rate at 48 MBps, but it requires only 10-20 logic slices. There is also a JTAG header on the Digital Board that connects to the FPGA for programming and debugging in the laboratory.

The current transceiver bandwidth is 30 MHz, although future designs will employ larger bandwidths. Received signals in an in-phase/quadrature modulation scheme are converted to baseband by the RF board and converted from an analog-to-digital representation by a Linear Technology LTC2284 dual analog to digital converter (ADC) at up to 105 megasamples per second (MSPS) with 14 bit resolution per sample. Processed baseband signals are converted from a digital to analog representation by an Analog Devices AD9777 DAC running at 100 MSPS with 16 bit resolution.

The KUAR has significant flexibility in locating signal processing functions in hardware logic, the embedded PowerPC processors or the CPH. This allows extremely parallel and time-sensitive operations to be moved into custom, reconfigurable hardware, and more complex operations to be implemented in software. Giving the system designer the fine-grained ability to determine whether an operation should occur in hardware or software allows for compact, efficient and innovative designs.

The design flexibility afforded by the KUAR can be demonstrated by the following three scenarios (Fig. 3):

1) Scenario 1 - Full Hardware Communications system: The communications system (BPSK, QPSK, etc.) is implemented entirely in hardware inside the FPGA. Data is fed from the CPH to the FPGA through the KUAR Memory Interface (discussed further in the Software section). This frees the CPH to perform various cognitive and control tasks. Placing the communications system in hardware allows for timing and performance requirements to be guaranteed and in general achieves a speed-up in comparison to the same system implemented in software.

2) Scenario 2 - Hybrid Hardware/Software Communications System: This scenario features a hybrid hardware/software implementation. Here, the two PowerPC cores can be used as general purpose RX and TX processors. These cores can execute software programs in much the same manner as the CPH. The PowerPC cores can also directly connect to any hardware accelerators or custom logic through the PowerPC's OPB and PLB buses which are standard PowerPC peripheral buses provided by Xilinx as pre-designed VHDL modules. This allows the cores to run synchronously with the

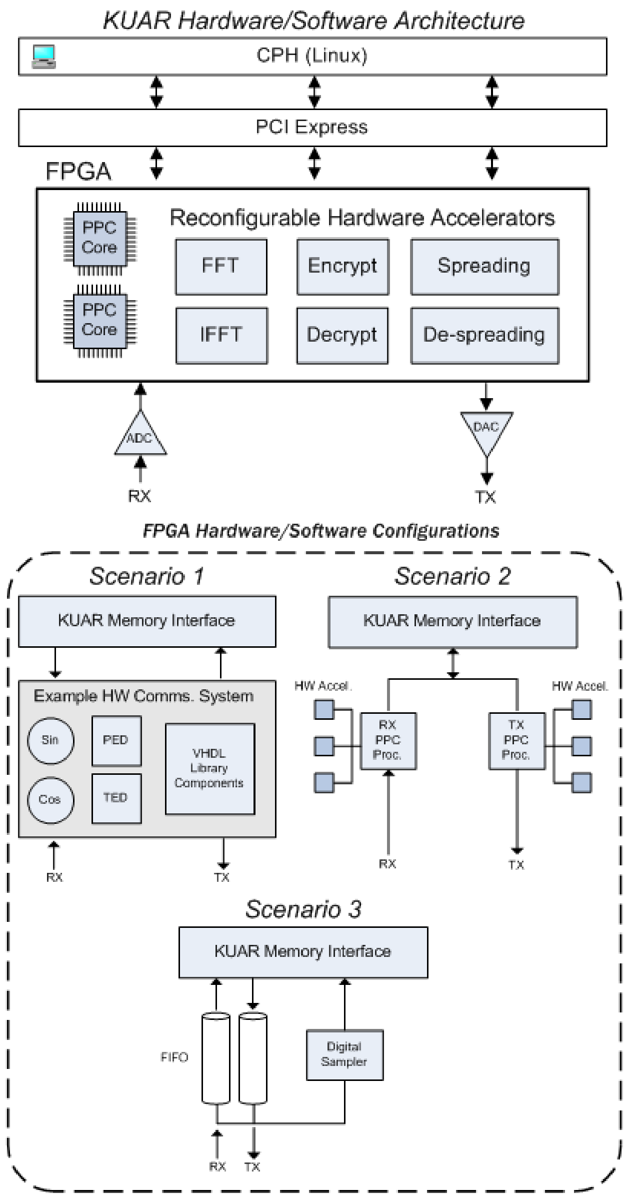


Fig. 3. KUAR Hardware/Software Flexibility

custom hardware, providing low latency data transfer. Designers can thus be extremely creative with regard to where signal processing operations are placed across this software/hardware boundary. Using this architecture the embedded processors may be dedicated to running a real-time operating system specially enhanced to use hardware accelerators, such as the KU Hybrid Threads project [11], which allows hardware accelerators to be controlled via the Posix Threads (pthreads) Application Programming Interface (API). More complex and user-interactive processing may then be moved into the CPH, alleviating the need for the same processor to meet strict real-time deadlines.

3) Scenario 3 - Full Software Communications System: In the final scenario, we demonstrate a minimal FPGA implementation. This would be used if a communications system was implemented entirely in software on the CPH. After all the signal processing is performed, the modulated data would then be sent to the FPGA. The KUAR Memory Interface and hardware first-in first-out (FIFO) buffers merely transfer this data to and from the ADC and DAC, where it is sent to the RF front-end. Because a minimal amount of logic slices are used in this design, it is easy to place other helper hardware in the FPGA such as a signal sampler, which can be used to adjust sampling rates, detect the presence of a signal, and perform general handshaking operations between the CPH and the data buffers. This is the implementation that is being used to test the GNU Radio software on the KUAR.

The aforementioned scenarios are just several of the possibilities afforded by the KUAR platform. Developers will be able to implement solutions regardless of whether they are more comfortable writing software or designing reconfigurable hardware.

C. RF Board

The RF module designs (left half of Fig. 2) have been tailored for experimental use. Features of the KUAR RF Board include the ability to set independent transmit and receive frequencies, as well as digitally control the transmit power output and receive gain levels. The RF modules incorporate standard SMA-style RF input and output connectors to allow the use of a variety of antenna types and configurations. Digital control of transmitter output power, receiver front end attenuation, and IF amplifier gain should prove to be useful for fading channel experiments, and also allows researchers to perform experiments in test environments. The RF modules currently in use offer a frequency range of 5.25-5.85 GHz, and are designed for operation in the 5 GHz Unlicensed National Information Infrastructure (UNII) and Industrial, Scientific and Medical (ISM) bands. An RF design for 2.05-2.70 GHz operation is currently being developed.

The 5 GHz RF module is a hybrid direct conversion design that makes use of a traditional superheterodyne frequency conversion to and from an intermediate frequency (IF) range of 1.85-2.45 GHz, which is directly converted to baseband using a quadrature demodulator, and from baseband using a direct conversion quadrature modulator. The RF modules are currently configured to select 30 MHz sections of the frequency band, in 4 MHz tuning steps.

1) Programming: A Freescale 8-bit microcontroller unit (MCU) is used to interface the digital processing section to the programmable components of the 5 GHz RF module. All frequency settings and amplifier gain controls are programmed using a Serial Peripheral Interface (SPI) bus, while the Rx chain variable attenuator is controlled with 3 V logic levels. The MCU is used to pass control register data and collect device status information, and is connected to the digital processing section using an inter-integrated circuit (I2C) bus.

2) Local Oscillators: The design incorporates three local oscillator (LO) sections; an IF receive (Rx) LO (Rx LO1), an IF transmit (Tx) LO (Tx LO2) and a common 3.4 GHZ fixed frequency LO (Rx+Tx LO3) which is supplied to separate receive and transmit front-end mixers. All three LOs share a buffered 16MHz reference frequency generated by a temperature compensated crystal oscillator (TCXO), and distributed using a dual 1:5 CMOS clock fanout buffer. The 16 MHz reference frequency is also provided to the digital processing section of the transceiver. Rx LO1 and Tx LO2 are two-stage differential output designs, with each LO comprised of a pair of integrated synthesizer / voltage controlled oscillator (VCO) devices; an Analog Devices 1.85-2.15 GHz ADF4360-2 is coupled with an Analog Devices 2.05-2.45 GHz ADF4360-1 to provide an effective tuning range of 600 MHz. RX+TX LO3 components consist of an Analog Devices ADF4113 PLL synthesizer device controlling the tuning port of a Z-Comm SMV3300A VCO. The SMV3300A RF output is passed through a 5th order Chebychev interdigital band-pass filter (BPF), centered at 3.4 GHz with a -3dB bandwidth (BW) of 200 MHz, to the input of an Agilent MGA-82563 (+10dB Gain, +17dBm P1dB, 2.4dB NF @ 4.0 GHz) amplifier, which feeds into a Wilkinson 3dB splitter, providing a 3.4 GHz LO to the Rx and Tx chain frequency mixers.

3) Receiver: Starting at the input SMA RF connector, the Rx chain consists of a 5th order Chebychev interdigital BPF, centered at 5.5 GHz with a -3dB BW of 600 MHz, followed by a 6-bit programmable GaAs 0-31.5dB Hittite HMC425LP3 variable attenuator, and an Agilent MGA-85676 (+19dB Gain, +4.3dBm P1dB, 1.8dB NF @ 6.0 GHz) Low Noise Amplifier (LNA). The output of the LNA feeds the RF input of a Hittite HMC488MS8G GaAs double balanced mixer, which features an integrated LO amplifier, and mixes the fixed frequency 3.4 GHz input from Rx+Tx LO3, down-converting frequencies from the 5.250-5.850 GHz range to the 1.850-2.450 GHz intermediate frequency (IF) range of the receive section.

Down-converted IF frequencies are passed through a 5th order Chebychev interdigital BPF centered at 2.15 GHz with a -3dB BW of 600 MHz, and are then fed into an Analog Devices AD8347 direct conversion quadrature demodulator. The AD8347 amplifies the IF signal with two stages of variable gain amplification before frequency conversion via two Gilbert-cell mixers, which perform a direct conversion to baseband using the differential 1.850-2.450 GHz output from Rx LO1.

The Rx LO1 inputs to the AD8347 are internally conditioned using a poly-phase filtered phase splitter, then connect to the Gilbert-cell mixer inputs. The baseband outputs of the mixers are followed by separate in-phase (I) and quadrature-phase (Q) channel variable gain amplifiers (VGA). A user may select either automatic gain control (AGC), which employs baseband level detectors integral to the AD8347, or manually control the RX IF VGA gain levels with the output from an Analog Devices 6-bit AD5601 Rx Digital-to-Analog converter (DAC).

The AD8347 internal IF and baseband VGAs provide a cumulative 69.5 dB of gain control. The baseband VGA outputs are brought out of the device to allow filtering before final amplification. Baseband I and Q signals are passed through a pair of 30 MHz -3dB BW low-pass filters (LPF), before being amplified and output as differential I and Q signals to a pair of Analog Devices AD6645 12-bit 80MSPS Analog-to-Digital converters (ADC) in the digital processing section.

4) Transmitter: The 5 GHz module Tx chain begins with differential I and Q inputs from an Analog Devices AD9777 16-bit 160 MSPS dual DAC located in the digital processing section, which are low-pass filtered with a pair of 30 MHz -3dB BW differential LPFs, then passed to I and Q inputs of an Analog Devices AD8349 direct conversion quadrature modulator. The quadrature modulator has an automatic gain control unit that can provide amplitude normalization.

The modulator uses the differential 1.850-2.450 GHz output of TX LO2 to up-convert baseband I and Q signals. The differential Tx LO2 input signal is buffered, and then split into I and Q signals using a poly-phase phase splitter. These two LO signals are amplified, then mixed with the corresponding I channel and Q channel baseband input signals in two Gilbert cell mixers. The mixer outputs are then summed together in the AD8349 output amplifier.

The 1.85-2.45 GHz output of the AD8349 is passed through the TX IF BPF; a 5th order Chebychev interdigital design centered at 2.15 GHz with a -3dB BW of 600 MHz. The Tx IF BPF output is then amplified by the programmable Tx IF VGA. The Tx IF VGA consists of a Phillips BGA2031/1 VGA (+23dB Gain, +11dBm P1dB @ 1.9 GHz) combined with an Analog Devices AD5601 6-bit DAC; the DAC output voltage sets the gain level of the BGA2031/1, which has a gain control range of 56dB.

The output of the Tx IF VGA is connected to the IF port of the Tx mixer (Hittite HMC488MS8G), which uses the 3.4 GHz input from Rx+Tx LO3 to up-convert Tx IF frequencies to the 5.25-5.850 GHz range. The RF output of the Tx mixer is amplified by a Mini-Circuits ERA-1SM (+6dB Gain, +12dBm P1dB, 4.3dB NF @ 6.0 GHz) RF amp, then passed through a 5th order Chebychev interdigital BPF, centered at 5.5 GHz with a -3dB BW of 600 MHz. The band-pass filtered signal is fed into the input of an Agilent MGA-83563 (+17dB G, +15dBm P1dB, +18dBm PSAT@6.0 GHz) amplifier, with the amplifier output connected to the Tx output SMA connector, providing an output of up to 15 dBm (32mW) of RF signal power in the 5.25-5.85 GHz frequency range.

D. Antennas

Three basic configurations of broadband 5 GHz directional planar antennas have been designed and constructed to complement the KUAR system; basic passive, active Rx, and active Tx. The passive antennas are intended for use in indoor or short range outdoor test environments, while the active versions utilize integrated RF amplification and filtering to provide longer range outdoor test performance.

The active and passive antennas share the same basic planar element design, consisting of an air dielectric patch element and feed structure that exhibits a 1.5:1 VSWR BW of 1.5 GHz, centered at 5.5 GHz. The element design provides 8.5dB of directive gain, with respective E and H plane -3dB beam-widths of 80 and 70.

The passive antenna element feed structure is directly connected to an SMA-style RF connector, and is suitable for use on either the KUAR Rx or Tx port, or both, depending upon testing needs. In the case of the active Rx antenna, the feed structure connects to a 5.5 GHz 3rd order Chebychev interdigital BPF with a -3dB BW of 600 MHz. The filtered signal is then passed through an Agilent MGA-86576 LNA to the antenna SMA-style RF output connector.

The active Tx antenna design uses an SMA-style RF connector as an RF input. The input signal is fed into an Agilent MGA-545P8 (+11.5 dB Gain, +21dBm P1dB, +22dBm PSAT) RF power amp, then passed through a 3rd order Chebychev interdigital BPF centered at 5.5 GHz, with a -3dB BW of 600 MHz. The output of the BPF is connected to the antenna element feed.

A KUAR 5 GHz transceiver, equipped with the previously described active Rx and Tx antennas, is capable of recovering signal levels as low as -100dBm, and can transmit an Effective Isotropically-Radiated Power (EIRP) level of up to +25dBm (354mW) .

IV. KUAR SOFTWARE

A. Architecture

Software-defined radio platforms are complex devices from a software point-of-view. The platform must be agile enough to support processing at multiple hardware and software layers. It must also allow researchers of various backgrounds, including communications, networking, system engineering and RF researchers, to perform research and experimentation at their locus of expertise while simultaneously not burdening them with the complex details of other layers of the platform. Software modules need to operate on and within a large number of different execution environments ranging from computer aided design tools to digital logic executing on the FPGA. These environments are listed in Table 1 starting at the signal processing level.

TABLE I
SDR SOFTWARE ENVIRONMENTS FOR RADIO DEVELOPMENT

Execution Environment	Environment Type / Tools	KUAR Use
FPGA	Digital logic described in schematics or VHDL. (Xilinx ISE, Xilinx command line tools; VHDL simulators; Signal simulation tools)	Signal processing; radio control; RF environment sensing; Design of communications signal processing modules and systems
FPGA embedded processor	Programs written in C or any language with PPC compiler; custom runtime (Xilinx EDK, GCC or any PPC compatible compiler)	Signal processing; radio control; RF environment sensing; Secondary Linux layer
RF Transceiver and MCU	Programs written in C; custom runtime (Code Warrior with I2C & SPI Hardware Beans, GCC, any HC08 compatible compiler)	Control and sensing of RF Transceiver functions
Control Processor	Linux Kernel / realtime code; Operating System; Linux userland programs; written in C or other common programming language (Any x86 compatible compiler, any software development tools)	Device drivers linking the CP, FPGA, and other hardware components; Network protocols; Control programs for loading and managing the FPGA; Radio services; Management programs; User applications; Running experiments; SDR network control scripts

Learning, configuring, managing, and integrating these environments can be a difficult task, especially given the reality that researchers may want to work on either a specific layer or across multiple layers. The KUAR Software Architecture is shown in Fig. 4. The Software Architecture incorporates radio module design and libraries depicted on the left of the figure, management and hardware abstraction in the middleware layer, drivers and signal processing modules on the bottom, and network protocol stacks and user applications on the right.

This architecture provides bootstrapping to researchers who want to perform targeted experiments while also allowing developers the flexibility to implement experimental designs. This will be further discussed in the KUAR Workflow section, but a library of pre-built

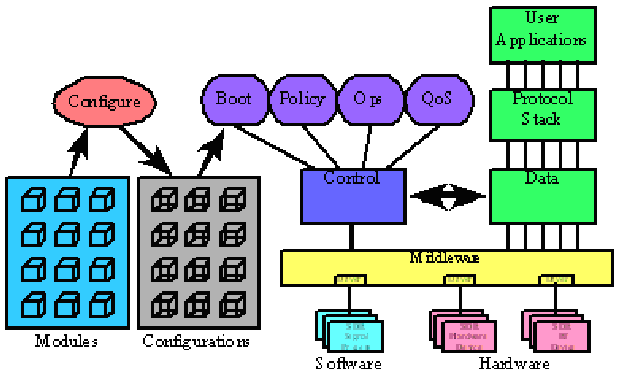


Fig. 4. KUAR Software Architecture

modules (both software programs and reconfigurable hardware) allow various radio configurations to be created (BPSK, QPSK, OFDM, etc.). This modular setup would allow for example, a networking researcher to perform multi-hop routing experiments on top of any number of physical layer designs without requiring an in-depth knowledge of RF design. Additionally, a communications engineer could design a new communications system and immediately test it with various cognitive radio programs (OSSIE, other SCA programs).

B. Software Architecture, Development and Tools

In order to simplify the complexity of programming the KUAR, we have developed a KUAR Control Library that is composed of various APIs. These interfaces allow radio and experiment software to be written at a high level with logical commands and syntax. They also shield the developer from extremely specific and nuanced details of implementation[1]. Rather than meticulously setting every register on each RF component (PLL's, DAC's, etc), the API abstracts these actions to simple calls that set the desired RF parameter, such as transmit frequency.

1) RF Control API: The RF Control API is part of the KUAR control library. The RF front-end consists of multiple SMBUS[2] controlled components including phase-locked loops (PLL's), quadrature demodulator chips, analog to digital converters (ADC's), and digital to analog components (DAC's). These devices work in unison to provide independent control of transmit and receive frequencies, transmit power and receive gain.

[1]These details are still accessible through the design and usage documentation generated during the design and assembly of the KUAR. Information regarding design choices and low-level programming is available by request or in a password protected Wiki on the KUAR website (https://agileradio.ittc.ku.edu/) that is accessible to third-party KUAR developers.

[2]System Management Bus. A communications bus created by Intel in 1995 that allows for various on-board components to exchange commands and data. http://www.smbus.org.

TABLE II
SAMPLE RF CONTROL API FUNCTIONS

Abstract RF Parameters	Related rfControl API function (written in C)
Transmit frequency	`KUAR_rf_status KUAR_rf_Tx_set_frequency( KUAR_rf_settings * settings, KUAR_frequency * frequency)`
Transmit gain	`KUAR_rf_status KUAR_rf_Tx_set_gain( KUAR_rf_settings * settings, gain_cdB gain_mB)`
Transmitter on/off	`KUAR_rf_status KUAR_rf_Tx_set_power_on( KUAR_rf_settings * settings, boolean power_on)`
Receive frequency	`KUAR_rf_status KUAR_rf_Rx_set_frequency( KUAR_rf_settings * settings, KUAR_frequency * frequency)`
Receive gain	`KUAR_rf_status KUAR_rf_Rx_set_gain( KUAR_rf_settings * settings, gain_cdB gain_mB)`

Automatic Gain Control (AGC), useful in many standard communications systems, can be enabled via a hardware jumper and will be controlable via software on future RF boards. The RF Control API abstracts this collection of related components into a structure that consists of five fields as shown in Table 2.

The previous table shows the functions for setting the various parameters of the RF Settings structure. There also exists a matching set of functions for retrieving the values, and a function call to configure the hardware based on the desired settings. Using this architecture, the RF Control API allows the user to create and store these specification parameters ahead of time, and then apply them with a single commit function call. In this manner an application which frequently changes between a set of pre-determined frequency ranges may rapidly hop between different frequencies. Alternatively, a set of RF Settings structures could be statically allocated for a specific operational frequency band such that radio operation is confined to that band. Finally, the RF Control

API includes a data structure to determine the current capabilities of the given radio (transmit frequency, receive frequency and gain ranges and fidelity), so that code may be written independently of the RF front-end attached to a given radio.

2) FPGA Control API: In addition to the RF Control API, there also exists an API for controlling the FPGA. The FPGA may be configured using a function call which takes the path to a Xilinx bit-file[3] and returns a status code. Once the FPGA has been configured, the software programmer may then use another API call to access an array which contains the memory mapped registers and data buses of the configuration loaded on the FPGA. From the point of view of the software programmer, two API calls result in a configured FPGA image and the necessary data streams to communicate with it. From the point of view of the hardware engineer, there are several pre-built modules that allow registers, FIFO's, and addressable memory elements to be connected to the radio memory bus. One of the integral problems of hardware/software co-design is creating interfaces to allow for data to be easily and accurately transferred from the software layer to the hardware layer. Data flow and access in these implemented interfaces is accomplished in hardware by the KUAR Memory Interface and in software by the FPGA Control API.

3) KUAR Memory Interface: The KUAR Memory Interface is a set of VHDL modules that provides Control Processor Host bus abstraction, and a set of predefined memory elements, including registers, FIFO's, and RAM's that may be accessed by both the CPH and the hardware module. This interface has already allowed the CPH-FPGA connection to be migrated from a direct connection to the memory bus (KUAR hardware version 2.1) to a PCI Express bus connection (KUAR hardware version 3.0). The KUAR Memory Interface provides a constant data and control interface to the hardware programmer regardless of the KUAR version.

Data Registers can be written in VHDL and synthesized to the FPGA, where they are accessible in much that same way that a programmer would store values in the register of a CPU. These FPGA registers are then memory mapped in the Linux operating system, allowing the programmer to simply write data to a specific memory location and know that the given data will be transferred into the hardware logic. The memory-mapped data elements are exposed through the FPGA Control API so that developers do not need an in-depth knowledge of the Linux memory system or hardware-specific constructs. This API allows for hardware to be accessed using standard bus techniques. The state of the hardware can be controlled and monitored through software by reading and writing to and from control registers. Hardware FIFO's and memory elements appear to the software programmer as buffers for sinks and sources. Overall, this system allows hardware programmers to easily connect existing components to the KUAR Memory Interface or easily write new components without extensive knowledge of the implemented CPH-FPGA bus.

[3]For Xilinx FPGAs, the binary-configuration file used has the file extension of bit, and is referred to as a bit-file.

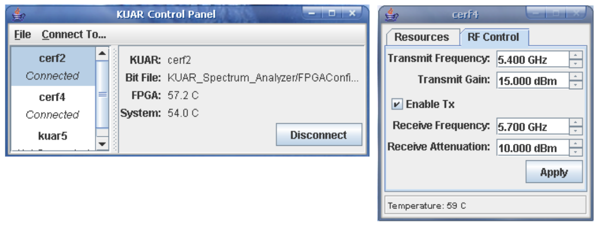
4) Additional Features API: The main purpose of the KUAR Control Library is to provide a simple interface to the RF front-end and the reconfigurable hardware. However, several additional features are included. These include simple logging routines, error handling codes specific to the agile radio, unit-based data-types (i.e. Hz, dB, sec), and radio status controls for thermal and power systems.

5) Software tools using the KUAR Control Library: All the current features of the KUAR Control Library are exposed via command line utilities. These include:

- rfControl - Controls the RF front-end.
- fpgaCnfg - Writes Xilinx bit-file to configure FPGA.
- fpgaRW - Read and write data both to and from hardware/logic in the FPGA. The program can be used to automate large data transfers and collection for experimentation. Complex experiments can be easily scripted using fpgaCnfg and fpgaRW.
- thermal - Determine the temperatures of vital components.
- power - Monitors current sensors on Power Board to measure radio power usage for experimentation purposes or to report such statistics to cognitive control software on CPH. Allows for control of power consumption by various devices.

There is also a graphical user interface (GUI) called the KUAR Control Panel, which allows for these API's to be controlled remotely or locally. The main program window displays the status of available radios on a given network and allows the user to control one or more KUARs.

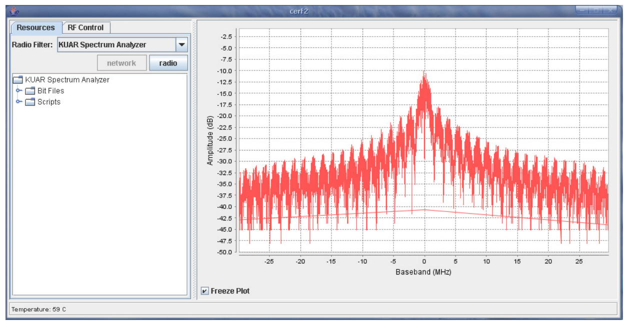
Once connected, each radio gets its own window which contains tabs for generic control parameters on the left, and data specific to the current FPGA image on the right. Fig. 5 shows the layout of the main window and the generic RF Control data. Fig. 6 shows the spectrum analyzer configuration, which uses the FPGA Control API to connect to the data stream of an FFT implemented in the FPGA. The implementation shown is a 16,383 point FFT with a real-time refresh rate of about a quarter of a second, with the main bottleneck being transmitting the data back to the control computer over the network. Several other plotters are also implemented for analyzing eye-diagrams, symbol constellations, and real-time error graphs. Each of these plotters may be configured to work with an arbitrary FPGA image through an XML



KUAR: cerf2
Bit File: KUAR_Spectrum_Analyzer/FPGAConfi...
FPGA: 57.2 C
System: 54.0 C

Transmit Frequency: 5.400 GHz
Transmit Gain: 15.000 dBm
[✓ Enable Tx]
Receive Frequency: 5.700 GHz
Receive Attenuation: 10.000 dBm
Temperature: 59 C

Fig. 5. KUAR Control Panel Radio Controls



Radio Filter: KUAR Spectrum Analyzer
[✓ Freeze Plot]
Temperature: 59 C

Fig. 6. Spectrum Analyzer Window with Data from a KUAR running the Spectrum Analyzer FPGA Image

file which defines the locations of data sources, control registers and status registers.

6) Distributed Radio Control / Management: "Radio Net" scripts have been developed that automate the process of setting up and executing multi-radio experiments. These scripts execute commands over Secure Shell (SSH) connections with each radio in the network that is participating in the current demonstration or experiment. On each radio, the FPGA is configured with the relevant communications system or other image. Any corresponding cognitive or control software on the Linux layer is also configured and started. For example, an experimenter could test a jamming/interference resistant design by configuring two radios to use the communications system under test, while a third radio could be configured to act as an interferer. A fourth radio might be configured to act as a spectrum analyzer so that RF activity can be observed and recorded for analysis. Radio Net scripts can be executed from either a command line prompt or the KUAR Control Panel on the experiment control computer[4].

7) Operating System: The CPH runs a Linux 2.6 kernel and can support a full Linux distribution such as Fedora Core or Ubuntu. FPGA firmware registers are addressable as PCI Express registers and exposed through the FPGA Control API discussed previously.

[4]The experiment control computer may be any computer with an SSH client and a network connection to the controlled radios. Therefore, any KUAR may be used as an experiment control computer.

8) KUAR VHDL Component Library / Communications System Library: We readily acknowledge that not every experimenter wants to implement an entire communications system on the KUAR. While it is possible to develop virtually any type of communications systems on the KUAR platform, we are developing a library of common components and systems that can be used for various experiments. Currently, we have implemented full, synthesizable VHDL designs for BPSK with phase and timing recovery, QPSK, M-QAM, and a simple multi-carrier system. By the time this paper is published we will also have a WiMax 802.16 physical layer reference design featuring a 256 subcarrier OFDM system. In addition to full communications systems, we have created a library of re-usable VHDL components which includes a signal sampler, energy detector, direct digital synthesizer (DDS), phase and timing error detectors, as well as data and system abstraction components such as CPH Processor and Bus abstraction blocks, Control and Status registers, and agile modulation blocks.

V. KUAR Design Workflow

The KUAR is designed to be used with industry standard design tools. As such, we have implemented an example workflow that we feel mimics design processes used in industry while providing certain levels of abstraction to various teams of designers. For example, engineers working on communications system design often use tools such as Matlab and Simulink. Once a design is complete, it can be implemented using components from the KUAR VHDL Component library if the designer is not familiar with reconfigurable hardware development using hardware description languages (HDLs) such as VHDL or Verilog. If the designer wishes to implement a custom VHDL design or the design team has access to HDL programmers, standard tools like Xilinx ISE can be used to design, simulate and implement hardware designs. If the designers wish to implement a higher level design, they can also use entire pre-built radio systems like the BPSK and QPSK systems discussed above.

After these designs have been flashed into the FPGA, other components allow for the design to be easily tested. One KUAR could be used to transmit a modulated signal to another KUAR which is testing an experimental receiver design. The signal sampler component could be implemented to detect the beginning of the transmitted packet and start writing data to a FIFO receive buffer in the FPGA. The KUAR Memory Interface handles the memory mapping of buffers and other control/status elements in Linux. This allows control and testing programs to easily read and write data to and from the FPGA. The fpgaRW control program is designed to accept input data and create output data formatted in raw text, Matlab, or Simulink format. Output from the FPGA can thus be easily fed back into Matlab/Simulink using a

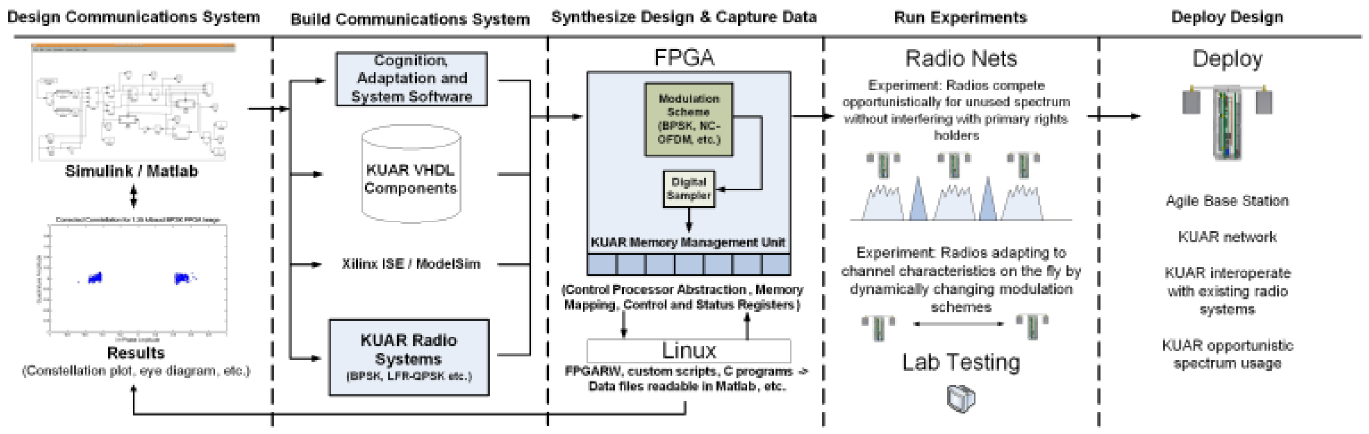

Design Communications System
Build Communications System
Synthesize Design & Capture Data
Run Experiments
Deploy Design
Simulink / Matlab
Results
(Constellation plot, eye diagram, etc.)
Cognition, Adaptation and System Software
KUAR VHDL Components
Xilinx ISE / ModelSim
KUAR Radio Systems
(BPSK, LFR-QPSK etc.)
FPGA
Modulation Scheme (BPSK, NC-OFDM, etc.)
Digital Sampler
KUAR Memory Management Unit
(Control Processor Abstraction, Memory Mapping, Control and Status Registers)
Linux
FPGARW, custom scripts, C programs -> Data files readable in Matlab, etc.
Radio Nets
Experiment: Radios compete opportunistically for unused spectrum without interfering with primary rights holders
Experiment: Radios adapting to channel characteristics on the fly by dynamically changing modulation schemes
Lab Testing
Deploy
Agile Base Station
KUAR network
KUAR interoperate with existing radio systems
KUAR opportunistic spectrum usage

Fig. 7. KUAR Design Workflow

series of test bench scripts. This allows the output of the implemented system to be easily compared to the output of the simulated system. This setup also allows for the designer to easily generate eye diagrams, BER curves and other diagnostic plots that provide information about the performance of the communications system under test.

By providing an abstraction layer that allows programs running in Linux to easily read and write data to and from the FPGA, another set of programmers and designers can work on network and application layer designs without having to have in-depth knowledge of lower level aspects of the communications system. This will allow cognition, adaptation and control software to be implemented and tested on a variety of communications systems.

While it is desirable to have an understanding of all layers of communications systems design and implementation, this is not always possible depending on the composition of the design team. We feel that the KUAR platform will enable research on various levels of the network stack that may have previously been impossible to perform because of the complexity involved in developing a system that provides the necessary underpinnings to support experiments at a specific, desired layer.

VI. KUAR CONFIGURATION & ADAPTATION

The process of setting up the appropriate communication modules and transmission parameter settings for the KUAR is separated into two stages. First, the configuration stage determines higher level communication settings that will be used by the radio. These settings are passed to the adaptation stage which uses this information to determine which lower level transmission parameters are available to modify and what the range of values is for each parameter.

A. *Configuration*

One of the major end goals of any software defined radio is high modularity of radio component functions. Ideally, software defined radio functions/techniques are swapped out as dictated by various environmental or user situations. However, there has been little discussion on what the initial configuration of the system should be upon startup and who or what determines this initial configuration.

In order to address these issues, our configuration phase is designed to allow users a means of setting constraints regarding expected radio performance. We have termed these constraints Mission-Oriented Communications (MOC) properties. Each MOC property is a formal description of a desired radio quality, attribute, or situation. For example, we define a MOC property for specifying the shape of a spread signal in terms of avoiding detection and interference from that detection. We formally define eight properties:

1) Low Probability of Detection / Interference
2) Avoidance / Rejection of Non-Intentional Interference
3) Multipath Mitigation
4) Information Assurance / Robustness
5) Jamming Resistance
6) Communication Range
7) Communication Capacity
8) Bandwidth Efficiency

Each of these properties will be discussed in detail in a future paper. These properties allow us to perform reasoning within a rule based system, analyzing user input about expected radio performance and various implementation techniques.

A set of these constraints, defined by a user, are given to a rule based engine which analyzes the constraints against a pre-defined set of radio component implementation techniques, various modulation techniques,

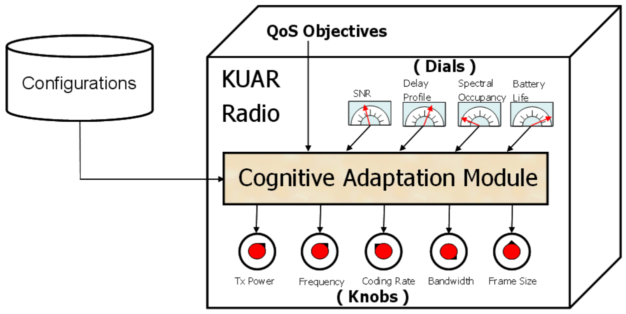


Fig. 8. Cognitive Adaptive Module

compression algorithms, error correcting codes, and spreading methods. The engine is comprised of rules which add or remove support for a given implementation technique based on the given constraints and then yields a specific configuration, which is then used as the initial configuration for all radios involved.

The rule engine was developed in Java and implemented on top of the Java Expert System Shell (JESS). The rules that the system uses were obtained from an "expert" on physical layer communications.

B. Adaptation

The cognitive adaptation module (CAM) uses the information from the configuration to understand the available transmission parameters and their possible ranges of operation. At the core of the cognitive adaptation module is an artificial intelligence (AI) engine that uses information sensed from the environment and quality of service (QoS) objectives to determine the appropriate lower level transmission parameters that may change more frequently than those in the configuration stage. Fig. 8 gives a visual representation of an instance of the KUAR and how the CAM is used.

The environmentally sensed parameters, also commonly referred to as "dials", are input to the CAM. The CAM reads these dials and sets the appropriate transmission parameters, also commonly referred to as "knobs". We have recently developed two cognitive engines for the KUAR. A genetic algorithm driven engine has been developed along with an expert system driven engine. Our current focus is identifying the hardware requirement trade offs for using each engine and determining the appropriate engine for the KUAR.

An important design decision for developing a cognitive engine is the selection of the "knobs" and "dials". Having a poorly selected set of parameters results in an uniformed CAM that outputs inaccurate decisions. We have identified a list of common wireless parameters that are essential to the operation of a cognitive radio.

Using these sets to control the operation of the radio is the primary task of the cognitive methods. In order for any cognitive method to perform its task, a relationship must be found between the "knobs", "dials", and performance objectives that give the cognitive engine the intelligence to understand how the environment is affected by the parameters.

Several challenges exist within the cognitive radio configuration and adaptation phase. Finding the correct objectives and parameters are among the most important. Developing the relationships that exist between several parameters and multiple objectives is the key to developing a well informed cognitive radio. We have implemented two cognitive engine methods and shown how each uses the derived relationships to find the appropriate operating parameters for a wireless environment. Along the way, several implementation trade-offs were identified that can be used to tailor the implementation to a specific hardware resource environment.

VII. KUAR APPLICATIONS

Several experiments are being developed for the KUAR and we briefly describe them in this section.

A. Agile Transmission

In a wide-band communications system, a large portion of frequency channels may be occupied by transmissions from incumbent or unlicensed users. Systems that desire to operate within these occupied channels must avoid placing subcarriers in occupied or licensed spectrum. Thus, to avoid interfering with these other transmissions, the subcarrier within the vicinity of the given transmission is turned off, or nulled. In the case of systems like OFDM, these null subcarriers are represented as zero-valued inputs to the FFT and IFFT blocks. When available spectrum is sparse, the number of zero-valued inputs in the FFT may be significant relative to the total number of the usable subcarriers. When the relative number of zero-valued inputs is quite large, significant hardware resources can be saved by pruning the FFT algorithm.

Channel conditions and incumbent spectrum occupancy (ISO) often vary over time so efficient FFT pruning algorithms should be able to generate an optimized FFT implementation every time the channel conditions and ISO changes. Given that the hardware resources of small form factor cognitive radios are limited, this FFT pruning algorithm would be very beneficial.

B. Distributed Radio Spectrum Survey

Determining whether a portion of the RF spectrum is in use is difficult. The common approach of measuring spectrum utilization with a spectrum analyzer only captures activity at a particular location for a particular time. This typical measurement approach can easily miss spectrum users who transmit intermittently, those with shaped antenna patterns, or those with very low-power signals. With a set of KUARs distributed over an area, coordinated observations can be collected and wideband signals can be captured for offline analysis. Given sufficiently long signals, offline analysis can elicit characteristics of time division multiplexed signals and detect weak signals in addition to conventional power analysis. Using distributed RF sensors enables us to better understand the RF environment over a region.

C. Channel Sounding Techniques

Channel sounding techniques are used to obtain the radio channel characteristics, such as the channel impulse response (CIR), the channel frequency response, the average delay, the delay spread and the coherence bandwidth among others. As the demand for high data rate wireless communications increases dramatically, radio channels are becoming more and more sophisticated (non-stationary transceivers and signal reflections) and the ability to characterize said channels is essential to the design of future radio systems.

In a Dynamic Spectrum Access network environment, the channel conditions might change rapidly due to random access to the channel by different types of users. Furthermore, radio channel characteristics can also change over a short period of time. Channel models built upon short-term measurement data may not accurately describe the channel. System designers require new techniques to model the long-term behavior of a given radio channel. Cognitive and software defined radios will allow for the development of channel sounding techniques for DSA networks. Cognitive radios will be capable of adjusting transmission parameters when channel conditions change and will be capable of capturing long-term channel characteristics.

VIII. Conclusion

We have presented the design details of the KUAR platform. The KUAR supports a very flexible RF front-end supporting wide transmission bandwidths and large center frequency ranges. The current version supports a 30 Mhz bandwidth anywhere within the 5-6 GHz, although auxilary RF designs will allow for operation in other bands, including the 2.4 GHz ISM band. The KUAR is extremely portable due to its small form factor, self-contained design, and on-board power supply. It also hosts powerful on-board processing to support a wide variety of complicated radio functions, network architectures and protocols, and cognitive algorithms. This highly configurable system has a robust set of both hardware and software tools to allow developers to work in their area of expertise without being encumbered by the other layers. Finally, a low cost build cycle will help facilitate wide distribution of KUAR units to researchers in the cognitive radios and DSA networks community, assisting them in implementing and validating new designs, algorithms and approaches.

References

[1] M. A. McHenry, “NSF Spectrum Occupancy Measurements Project Summary,” tech. rep., Shared Spectrum, http://www.sharedspectrum.com/?section=nsf_measurements, August 2005.

[2] M. J. Marcus, “Unlicensed Cognitive Sharing of TV Spectrum: The Controversy at the Federal Communications Commission,” *IEEE Communications Magazine*, vol. 43, pp. 24–25, May 2005.

[3] Federal Communications Commission, “Spectrum Policy Task Force Report ET Docket No. 02-135,” July 2002.

[4] Federal Communications Commission, “Unlicensed Operation in the TV Broadcast Bands ET Docket No. 04-186,” May 2004.

[5] DARPA, “XG Project.” http://www.darpa.mil/ato/programs/XG/.

[6] J. Mitola, “Cognitive Radio for Flexible Mobile Multimedia Communications,” in *Proceedings of the IEEE International Workshop of Mobile Multimedia Communications*, vol. 1, (San Diego, CA, USA), pp. 3–10, November 1999.

[7] R. J. DeGroot, D. P. Gurney, K. Hutchison, M. L. Johnson, S. Kuffner, A. Schooler, S. D. Silk, and E. Visotsky, “A Cognitive-Enabled Experimental System,” in *Proceedings of the First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks*, vol. 1, (Baltimore, MD, USA), pp. 556–561, November 2005.

[8] S. M. Mishra, D. Cabric, C. Chang, D. Willkomm, B. V. Schewick, A. Wolisz, and R. W. Brodersen, “A Real Time Cognitive Radio Testbed for Physical and Link Layer Experiments,” in *Proceedings of the First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks*, vol. 1, (Baltimore, MD, USA), pp. 562–567, November 2005.

[9] H. Harada, “Software Defined Radio Prototype Toward Cognitive Radio Communication Systems,” in *Proceedings of the First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks*, vol. 1, (Baltimore, MD, USA), pp. 539–547, November 2005.

[10] GNU Radio, “The GNU Software Radio Project.” http://www.gnu.org/software/gnuradio/.

[11] D. Andrews, W. Peck, J. Agron, K. Preston, E. Komp, M. Finley, and R. Sass, “hthreads: A Hardware/Software Co-Designed Multithreaded RTOS Kernel,” in *Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation Facolta’ di Ingegneria*, (Catania, Italy), September 2005.