IMPLEMENTATION OF A COGNITIVE RADIO MODEM

Peiman Amini (University of Utah, Salt Lake City, UT, USA[+]; pamini@ece.utah.edu)
Ehsan Azarnasab([+]; azarnasa@ece.utah.edu), Salam Akoum ([+]; akoum@ece.utah.edu)
Xuehong Mao([+]; mao@ece.utah.edu), Harsha Rao ([+]; hrao@ece.utah.edu)
Behrouz Farhang-Boroujeny ([+]; farhang@ece.utah.edu)

ABSTRACT

Design and implementation of a cognitive radio modem on the Small Form Factor (SFF) Software Defined Radio (SDR) platform, provided by Lyrtech and Texas Instruments (TI) is reported. Filterbanks were used for spectrum sensing and this method is shown to exhibit superior performance in terms of the spectral dynamic range when compared to the conventional Fast Fourier Transform (FFT) techniques, i.e. periodogram method. To ensure reliability, a distributed sensing method is considered. Packet detection and channel equalization are performed using a cyclic preamble. A fractionally spaced equalizer is used for both channel and timing recovery. The different processing tasks are divided between a TI c64x+ DSP and a Xilinx Virtex IV FPGA while an ARM9 core is used for interfacing and running the Greenhills operating system. Measurement results from channel sensing are presented to show the high spectral dynamic range obtained using the filterbank sensing method.

1. INTRODUCTION

In disaster situations, it is absolutely crucial for law enforcement, rescue agencies, and other first responders to have the ability to communicate and exchange information quickly and reliably. Since wired networks cannot survive all types of disasters and are often impractical, communication through wireless networks is the ideal choice. Given the relative scarcity of available spectrum, coupled with the amount of time required to approve new users, the issue of spectrum access is the hardest task in the development of wireless networks geared towards first responders. Cognitive radio technology is the natural choice to overcome this access problem. First responders can use the idle portions of the spectrum to communicate with each other [1]. In this presentation, the Family Radio Service (FRS) band is chosen to implement a cognitive network of nodes able to transmit voice and data traffics with different Quality of Service (QoS) requirements [2].

Construction of a cognitive network for first responders presents many difficult challenges, the most obvious of which is how to fulfill the "awareness" requirement. Each node must be able to *sense* the channel for primary users, *i.e.* to identify the presence of *licensed* communications over the portions of spectrum, and share this information with the other nodes to allow the cognitive nodes to communicate reliably while avoiding the legacy devices. In our problem setup, 200 carriers (25KHz bandwidth each) are used by the cognitive radio to transmit voice or data provided that the legacy users are not using the channel. The method used for sensing should feature a high spectral dynamic range to enable the detection of the low power users. We choose the filterbanks sensing method [3], [4]. Filterbanks perform better than FFT in terms of detecting low power users when users with high power and low power are present.

The sensing is done in each node in the network and the results are passed to a base station which combines all the sensing information to compile a channel state information (CSI), which is then transmitted to all the nodes. The CSI is also used by the base station for channel allocation. Control channels are used for exchanging sensing information and control messages such as channel assignment between the leaf nodes and the base station.

Packet assembly is performed such that the voice and data traffics pass through the same signal processing blocks at the transmitter and at the receiver. Cyclic preamble is appended to the payload to be used in packet detection and carrier recovery. A fractionally spaced equalizer is used for channel and timing recovery. The equalizer is trained with the cyclic preamble and tuned with a decision directed algorithm [5].

The cognitive radio modem is implemented on the Lyrtech Software Defined Radio Platform [6]. Design decisions such as dividing the tasks between the FPGA and the DSP, and choosing the appropriate methods to implement each block are made in order to optimize the usage of the resources on the hardware. We simulate the functional modem in MATLAB and Simulink. We then use System generator for DSP for the implementation of the FPGA blocks and SIMULINK and Real Time Workshop to develop the individual modules for the DSP. TI Code Composer Studio is used to combine the DSP subsystem on the board. Networking simulations are performed in DEVSJava and the Hardware in the loop (HIL) technique that we have developed, [7] is used to implement the networking algorithm, when hardware resources are not available.

The rest of the paper is organized as follows. The system design, comprising of channel sensing, transceiver setup, and the MAC Layer are presented in Section 2. In Section 3, we describe and analyze the design decisions we have made for the implementation of the system. The implementation in DSP and FPGA are then described in Section 4, followed by the sensing measurement results in Section 5. We finally talk about the implementation status and draw our conclusions in Section 6.

2. PROBLEM SETUP

In this section, we present our sensing methodology which is the most important part of the cognitive radio. Then we describe the signal processing algorithms that we have chosen to implement a functional cognitive radio with minimal complexity. The system is broken down into four parts: (i) Channel sensing, (ii) Transmitter, (iii) Receiver, and (iv) MAC layer.

A. Channel Sensing

Choosing an appropriate channel sensing method is a key element in developing a functional cognitive radio system. This method should take into consideration the discrepancy in the received power levels from the different users and needs to feature a high dynamic range in order to reliably detect the spectrum holes.

While FFT has been long suggested as one possible sensing method [8], it suffers from spectrum leakage caused by the large side lobes in the frequency response of the filters that characterize each sub-carrier [9]. This results in significant inaccuracy and low dynamic range. On the other hand, the multi-taper method (MTM), suggested by Haykin as the optimal channel sensing method [10], comes at the expense of extra computational complexity. In our solution, we propose using filterbanks to sense the channel. By using filterbanks, the side lobes of the filters associated with each sub carrier can be made arbitrarily small by adjusting the filter length and other design specifications [3], [4]. The signal power of the output of the filterbank is used to estimate the signal spectrum.

In our system, we sense the channel 10 times per second. To sense the channel, each node halts its transmission for a specific amount of time, collects the necessary samples and runs the filterbanks sensing algorithm. The sensing information is next reported to the base station. The base station collects the sensing results from all the cognitive nodes, including itself, and broadcasts the compiled CSI results to all the leaf nodes in the network.

B. Transmitter

The software defined modem provides two modes of operation to process two different types of services. One service is a 19.2 kbps computer-to-computer data stream while the other service is a 16 kbps Continuously Variable Slope Delta Modulation (CVSD) vocoded voice. The data stream is encoded using a rate $1/2$, constraint length 7, convolutional encoder. The encoder generates two outputs using the two generator polynomials $x^6 + x^5 + x^4 + x^3 + 1$ and $x^6 + x^4 + x^3 + x + 1$. The data stream, constructed from the outputs of the encoder, is interleaved using a simple row-column block interleaver. The output of the interleaver is divided into groups of three coded bits and each group is mapped to an 8-PSK constellation using Gray mapping.

For the voice stream, we use a Reed-Solomon (RS)(63, 51, t=6) encoder having a generator polynomial $p(x) = x^6 + x + 1$. This has the ability to correct up to 6 random byte errors and does not need to be followed by an interleaver. The encoded voice stream is divided into groups of two coded bits and mapped using a Gray code to a QPSK constellation.

To use only one digital upconverter from baseband to Intermediate Frequency (IF) for both services, the packet assembly is done such that the symbol rate at the input of the pulse shaping filter is 20 kbps for both data and voice streams. The transmitted packet consists of two major parts: a 192-sample cyclic preamble, generated using three identical Binary Phase Shift Key (BPSK) modulated pseudo noise (PN) sequences of length 64, and a payload constructed using the data or voice streams output of the symbol mappers.

Upconversion is done using Cascaded-Integrator-Comb (CIC) filters [11] and a novel pulse-shaping filter (PSF) whose coefficients are chosen to achieve the Nyquist-M property while at the same time compensating for the CIC passband drop [12]. This combined CIC and pulse shaper (CPSCIC), when compared with other methods available in the literature, has been proven to achieve less passband ripple, more stopband attenuation and less ISI. Moreover, combining the pulse-shaping filter and the CIC compensator filter allowed us to save a great amount of space on the hardware. The upconverted signal is finally modulated to Intermediate Frequencies (IF) for transmission over the channel.

C. Receiver

The received signal is first down-converted by means of a CPSCIC matched to its transmitter counterpart. The baseband signal is then passed to the synchronization and channel equalization modules, both of which implemented in the fractional space. The fractional spacing between the samples is chosen to be $T_s/2$, where T_s is the symbol interval. The portion of the receiver performing the synchronization is shown in Fig. 1.

The received signal $y(t)$ can be modeled as:

$$y(t) = x(t - \tau - (nT_s/2))e^{j2\pi((f+\Delta f_c)t+\theta)} \quad (1)$$

where *x(t)* is the transmitted signal, τ is the time offset, Δf_c is the carrier offset and θ is the phase offset. We have ignored channel noise for brevity.

Synchronization is performed using a cyclic preamble. Cyclic preamble is chosen in our model because it can serve the dual purpose of estimating the timing and carrier offsets while at the same time equalizing the channel effects when coupled with a cyclic equalizer [13]. The repetition structure of the cyclic preamble allows us to detect the start of the packet as well as the carrier offset. This method exhibits good performance and is easy to implement. Packet detection is performed by computing the autocorrelation of the received signal. We correlate the signal with a shifted version of itself and find the position of the preamble by identifying

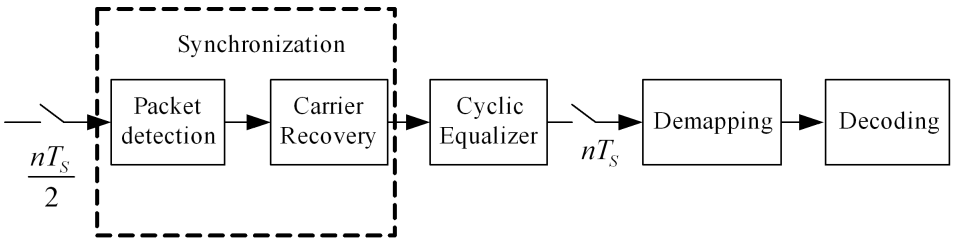

Fig. 1. Receiver block diagram after digital down conversion.

the interval over which the autocorrelation is significantly large [5].

On the other hand, the carrier offset is estimated using the following equation, [5]:

$$\Delta f_c = \frac{\angle\left\{\sum_n x(n')\cdot x^*(n)\right\}}{\pi(N+1)T_s} \tag{2}$$

where the summation is over one cycle of the preamble, $\angle(\cdot)$ represents the phase angle in radians, and $*$ denoted the conjucate.

After compensating for the carrier offset, we make use of a fractionally-spaced adaptive equalizer to compensate for the channel distortion, any residual carrier offset and to obtain the correct timing phase [5]. The equalizer coefficients obtained using this algorithm are further fine-tuned using a decision-directed adaptive scheme. The half symbol-spaced cyclic equalizer is shown in Fig. 2.

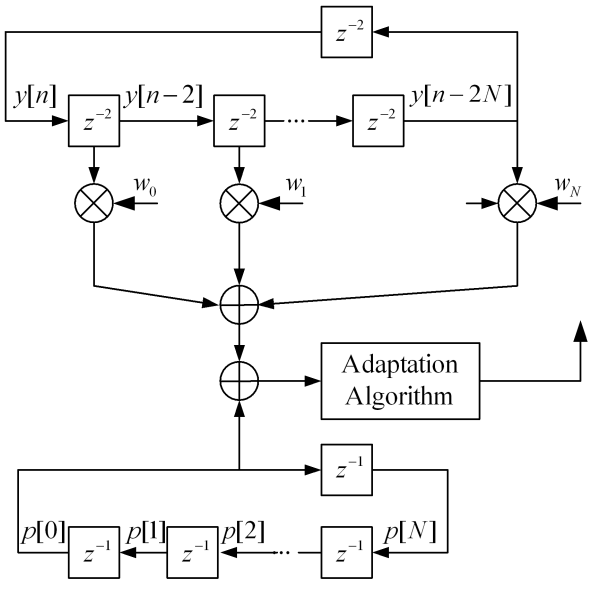

Fig. 2. Half symbol-spaced cyclic equalizer

In this figure, $p[n], n = 0, 1, \ldots, N$ represents a cycle of the preamble. The received signal is given by $\mathbf{y}_0 = [y[n], y[n-2], \ldots, y[n-2N]$. $\mathbf{w} = [w[o], w[1], \ldots, w[N]$ denotes the equalizer coefficient vector. The following equations describe the adaptation algorithm at step i.

$$e[i] = p[i \bmod \mathrm{N}] - \mathbf{w}^{\mathrm{H}}[\mathrm{i}]\mathbf{y}_{\mathrm{i}} \tag{3}$$

$$\mathbf{w}[i+1] = \mathbf{w}[i] + 2\mu e^{\star}[i]\mathbf{y}_i \tag{4}$$

where H denotes the Hermitian operators, and $\mathbf{y}_i$ is a cyclically rotated version of $\mathbf{y}_0$, delayed by 2 samples for each shift. The adaptive algorithm minimizes the mean-square error denoted by $\|\mathbf{e}\|^2$. $\mathbf{e} = [e[0], e[1], \ldots, e[N]]^T$ is used to obtain the optimal equalizer coefficients $\mathbf{w}$. We choose N equal to 64. Fig. 3 depicts the eye diagram of the received signal after packet detection, carrier recovery and cyclic equalization, respectively. Synchronization was simulated in MATLAB and Simulink. The signal to noise (SNR) was set equal to 9 dB. The normalized carrier offset is assumed to be 0.001.

D. MAC Layer

The MAC layer is designed to manage the medium access so as to co-exist with primary users. Coexistence is the primary goal in this design as well as in other CR systems, such as IEEE 802.22 [14]. The medium access method, as defined in the smart radio challenge problem statement [2], is frequency-division multiple access (FDMA). Secondary users (SUs) use frequency-division duplexing (FDD) to communicate with each other.

Control channels are used for coordinating sensing information, controlling leaf node communications and other management tasks. Our setup features uplink and downlink control channels. The uplink is based on Aloha while the downlink is a base station controlled broadcast channel used to broadcast sensing information and other management packets. We choose Aloha over carrier sense multiple access collision avoidance (CSMA/CA) because in the latter, channel sensing is not always possible; it is governed by the channel environment and the propagation delay. The base station receives and compiles the channel state data from all of the leaf nodes, including itself. The base station then broadcasts the compiled channel allocation table to all the leaf nodes. This allocation table is retained by the base

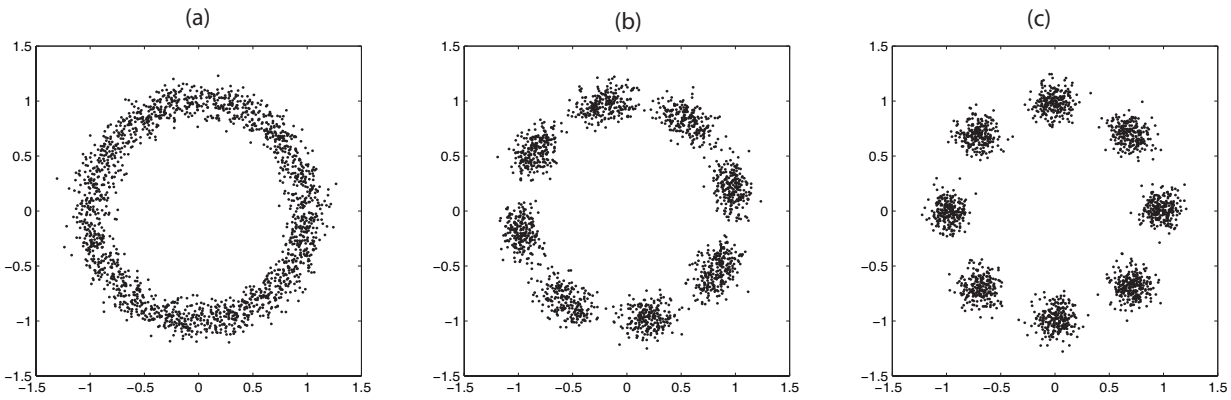

Fig. 3. Eye diagram of the received signal (a) after packet detection (b) after carrier recovery (c) after equalization

station and the leaf nodes until the conclusion of the next sensing interval. Each cognitive node stops transmitting and then senses the entire channel periodically. If the current sensing information differs from the last compiled channel state information, following a random delay, the channel state data is transmitted by each leaf node to the base station. More detailed description of our MAC Layer design and implementation can be found in [1] and [7].

3. System Layout and Hardware requirements

The cognitive radio modem was implemented on a Small Form Factor (SFF) Software Defined Radio (SDR) development platform provided by Lyrtech in collaboration with Texas Instruments (TI) and Xilinx. The SFF SDR is a self-contained platform consisting of three separate modules: the digital processing module, the data conversion module and the RF module.

The base of the platform is the digital processing module. It is designed around the TMS320DM6446 (also called DM6446) Digital Media Processor (DMP) System on Chip (SoC) [15] from TI and Virtex-IV XC4VX35 FPGA from Xilinx. DM6446 combines an Advanced Very Long Instruction Word (VLIW) 64x+ DSP and Reduced Instruction Set Computer (RISC) ARM926J-S cores, where the ARM microcontroller is mainly set to run the INTEGRITY Real-Time Operating System (RTOS) while DSP performs complex data processing. The data conversion module is equipped with a 125 MSPS, 14-bit dual channel ADC and a 500 MSPS 16-bit dual channel interpolating DAC provided by TI. The RF module is configured to have either 5 or 20 MHz bandwidth with working frequencies of 200-930 MHz for the transmitter and 30-928 MHz for the receiver. The SDR modem implementation is divided into different tasks each consisting of several modules. The SFF SDR platform gives the designer the option to choose the silicon device that is most suitable to the task being developed. We use the INTEGRITY and SMSHELL API provided by Lyrtech to target the board while we develop our signal processing tasks on the DSP core and the FPGA. The division of tasks between the DSP and the FPGA was made based on the availability of resources, the inherent characteristics of these cores, and the extra functionalities offered by TI and Xilinx. We make use of the already available Xilinx Logicore Blocksets for FPGA and the optimized DSP libraries written for vectors of complex numbers for C64x+ core. For instance, a Viterbi Algorithm was first written and optimized for the DSP, but considering the realtime required rate in the problem definition, the algorithm alone took 40% of the available time for processing. On the other hand, using the Viterbi decoder in the Xilinx Logicore performs the same task much faster. Traditionally, the algorithms that require fast and complex calculations but can be parallelized are best fit for the FPGA, while algorithms that require sequential analysis and decision making such as cognition and networking are usually implemented on the DSP. We note that it is hard to draw such a distinction between the functionalities of FPGA and DSP since DSPs are nowadays offering more pipelining and FPGAs are able to run sequential processing.

Interfacing between the DSP and FPGA is done using the Video Processing Sub-system (VPSS) data port. The DSP VPSS is a DM6446 DSP 16-bit synchronous video transfer port modified to support transfer of non-video data to and from the DSP. The VPSS consists of a Video Processing Front End (VPFE) and a Video Processing Back End (VPBE). The VPFE is used as an input interface to the DSP and the VPBE as an output interface from the DSP to FPGA. In the FPGA, a VPSS data port module, also consisting of a VPBE and a VPFE, is implemented to interface with the DSP VPSS. The data bus inside the FPGA is a 32-bit and the VPSS of DM6446 DSP bus is a 16-bit [6]. On the other hand, custom registers, a shared memory of eight 32-bit words between the DSP and the FPGA On-Chip Peripheral Bus (OPB), are used as configuration registers. As a result, the fast VPSS 32-bit bus, is our gateway between DSP and FPGA.

The tasks developed for the FPGA are implemented using

the System Generator for DSP. System Generator, an add-on to Simulink provided by Xilinx, produces a highly optimized FPGA realization, since each module used in the architecture maps to an FPGA library component that has been carefully constructed and optimized for the FPGA target device. Moreover, the System Generator provides us with a visual representation of the system that not only serves as the design specification, but as the behavioral simulation model and the source definition for the hardware. The system Generator implementation also facilitates the rapid investigation of various design options in the system [16], [17].

To develop the DSP subsystem, the algorithms targeting the DSP processor are first implemented in Simulink blocks. The Real-Time Workshop (RTW) is next used to produce the first version of the code for the individual Simulink blocks. Each block is then individually tested in Simulink external simulation. Although RTW is able to generate stand-alone C code for the Simulink blocks, it can only be used for rapid prototyping and testing since the code it generates is not optimized for a specific DSP or GPP target. The RTW generated code often needs extra memory and processing power and the optimization burden is put on the compiler only. The RTW thus cannot be used to implement the complete DSP subsystem whose requirements include realtime performance in terms of memory and speed and special data alignment. To overcome this problem, a Target Language Compiler (TLC) file is developed to customize the code generation. In writing the TLC, it is feasible to use optimized TI DSP libraries DSPLib [18] and compiler optimization techniques such as giving feedback to the compiler. Furthermore, the wrapper TLC (unlike inline TLC) saves a single version of each algorithm and therefore simplifies code maintenance. The wrapper TLC code (written for the individual blocks) can be reused in the independent compilation of the complete DSP subsystem project, without involving the RTW. Finally, the C code of the complete DSP subsystem (either generated by RTW or written as wrapper TLC is compiled by TI Code Composer Studio (CCS). CCS makes use of the high performance VelociTI architecture of DM6446 to optimize the code down to the programming level optimization (using the *-pm* switch).

4. Implementation of the Design

The distribution of the SDR modem components between the DSP core and the FPGA is shown in Figures 4. The VPBE and VPFE are used to transfer the data streams back and forth between the two modules while the custom registers are used for handshaking.

Since FPGA is logic based, any changes we make to the values saved in the custom registers are monitored and responded inside the FPGA. We use the custom register R_f to indicate the services required by DSP to be done by FPGA. Depending on the type of data processing required from the FPGA, the DSP specifies a command number inside the custom register R_f and then transfers the data through the VPSS. Similarly, the custom register R_d is used by the FPGA to inform the DSP about the characteristics of the bitstream arriving at the VPFE in the DSP subsystem. The DSP, being sequence based and often running an endless loop, uses the content of the R_d register to select the appropriate DSP function to be applied on the incoming data. The transmission is initiated, as shown in Figure 4, in the DSP core. The binary source is an arbitrary bitstream of voice or data incoming to the DSP. The voice data is input from the pcm3008 stereo audio codec at 48 KHz and encoded by the Continuously Variable Slope Delta Modulation (CVSD) having a data rate of 16kbps. The binary voice stream is zero padded to achieve a data rate of 20kbps. The data traffic, on the other hand, is a fixed computer-to-computer data stream. This data stream has a lower bit error rate than the voice stream and is retransmitted in the MAC layer (if missed or corrupted) to ensure data integrity of critical information.

The transmit packet is sent to the FPGA through the VPSS. The custom register R_f is set to zero if the source is audio and one if non-audio. In the FPGA, depending on the data content, either Reed Solomon (RS) coding or convolutional coding combined with interleaving is performed. Following the source coding in the FPGA, the binary vectors are retransferred to the DSP ($R_d = 0$) to perform modulation and framing. 8-PSK modulation is used for the data stream while QPSK is used for voice as per the problem statement. The binary vector is finally sent back to the FPGA ($R_f = 2$) to upconvert the signal to intermediate frequency (IF) and eventually transmit it over the air. Digital up conversion in the FPGA consists of three blocks. A novel combined CIC and pulse shaper (CPSCIC) that follow the Nyquist-M criterion, a CIC integrator, and a Direct Digital Synthesizer (DDS) to modulate the signal to the IF frequency. The CPSCIC filter we used is a 80-tap FIR filter generated using the Xilinx FIRCOMPILER provided by the System Generator for DSP. The baseband signal was modulated to an IF frequency of 30 MHz at a sampling rate of 80 MHz. Note that in the SFF SDR platform, the FPGA has access to IO, Data Conversion (DConv) and RF modules.

At the receiver side, two separate functionalities are first performed in the FPGA, digital down conversion (DDC) and sensing. To downconvert the received signal to baseband, a CIC differentiator and a CPSCIC filter are used. The resulting signal is then passed to the DSP ($R_d = 1$) where synchronization tasks and symbol demapping are performed. After demapping, the signal is passed back to the FPGA for decoding ($R_f = 2$ for voice and $R_f = 3$ for data). The decoded signal is finally passed to the DSP ($R_d = 2$). Sensing is activated in the FPGA every 100 ms by means of a timer. This timer activates the sensing module 10 times a second for almost 4 μs. The timer control circuit disables the transmitter functionality while the sensing is performed. The sensing data is first demodulated to baseband by means

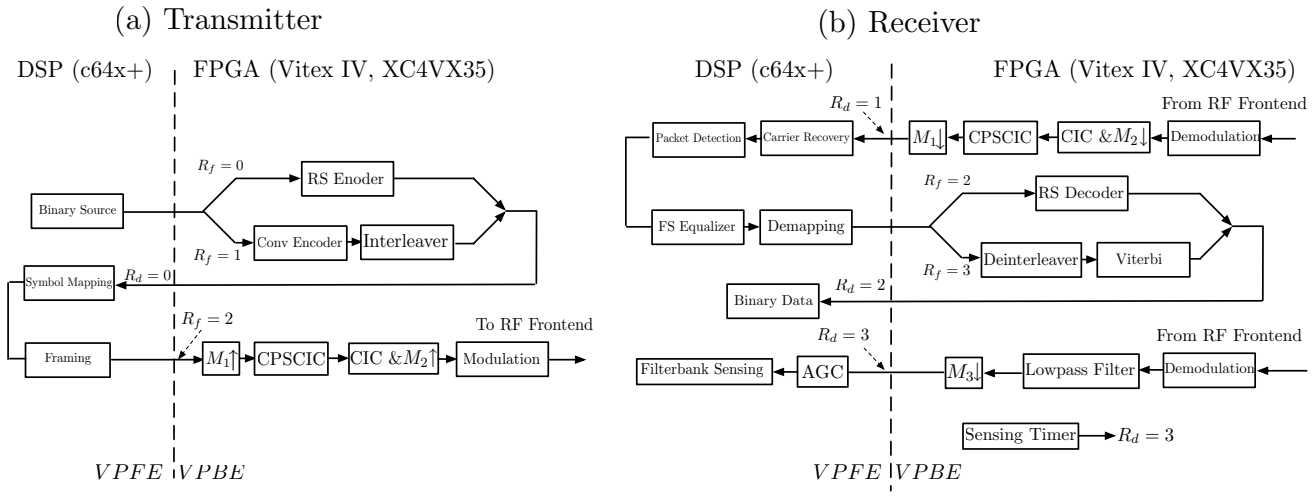

Fig. 4. System data flow

of a DDS whose frequency is centered at the IF frequency. A lowpass polyphase decimator is used to filter the required signal band and bring the sampling rate down to 5 MHz. Note that in order to make use of the maximum dynamic range of the ADC, An automatic gain amplifier (AGC) is developed in the DSP to control the gain of the analogue amplifiers available on the data conversion module before the signal is digitized. The resulting signal is then sent to the DSP ($R_d = 3$) for further processing.

In the DSP, the DSPLib library for C64x+ is used for efficient implementation of the filterbank sensing. A filterbank is implemented in polyphase structure using 256 8-tap polyphase elements which are the decimated coefficients of the described prolate filter in section 2. The output of the polyphase elements are then passed through FFT. The output energy of the filterbanks is then averaged over three decimated samples. The sensing information is compared with a tunable threshold to locate possible active primary users and create a 32-byte channel state information. This information is then transmitted to the basestation. The basestation compiles the sensing information from all of the users and creates a common CSI to be used for channel assignment.

5. Measurement Results

We have used a vector signal generator to generate a signal that emulates the effect of the primary users on the channel. 10000 samples are generated in MATLAB and are used as input to the signal generator. The signal generator modulates the signal and transmits over the frequency range 462 MHz to 467 MHz. Six sinewaves are added and then passed to the function generator. The function generator modulates these sinewaves to 463.278MHz, 463.367MHz, 463.456MHz, 463.545MHz, 463.624MHz, and 463.985MHz. The sinwaves at 463.278MHz, 463.456MHz, and 463.624MHz are transmitted at 25dbm power. The ones at 463.367MHz, and 463.545MHz are transmitted at -15dbm. 463.985MHz sinewave is transmitted at -9dbm.

The Power Spectral Density (PSD) of the received signal is presented in Fig. 5. 2048 samples are used for the filterbank with prototype filter of length 2048. The results of FFT, FFT with a Hanning window and filterbank are averaged over three decimated sample. The calculated PSD from 462MHz to 467MHz for these three methods are depicted in Fig. 5. As we can see filterbank sensing is able to show all the transmitted sinewaves clearly. FFT, on the other hand, has a considerable spectrum leakage which results in missing the three sinewaves at 463.367MHz, 463.545, and 463.985MHz. FFT with hanning while having better dynamic range than FFT, also misses two of the sinewaves.

6. Status of Implementation, Conclusion and Future Research

A single cognitive radio transceiver is developed to perform both cognition and transceiver tasks. We first simulated our modem in MATLAB and then migrated the simulation to Simulink. We used System Generator for DSP to implement the FPGA modules of our system. Real-time workshop and wrapper TLCs are used to develop the individual modules for the DSP. Code Composer Studio was used to combine all the DSP modules.

At present, we have only one RF frontend available to us. Using an arbitrary signal generator, the interference of primary users over 462 MHz to 467 MHz is emulated. We are able to show that our cognitive node can move to an unoccupied frequency band when a primary user starts transmitting on the carrier that is currently in use. Our measurement results show that filterbank sensing method has less spectrum leakage than FFT, as predicted in [3], [4].

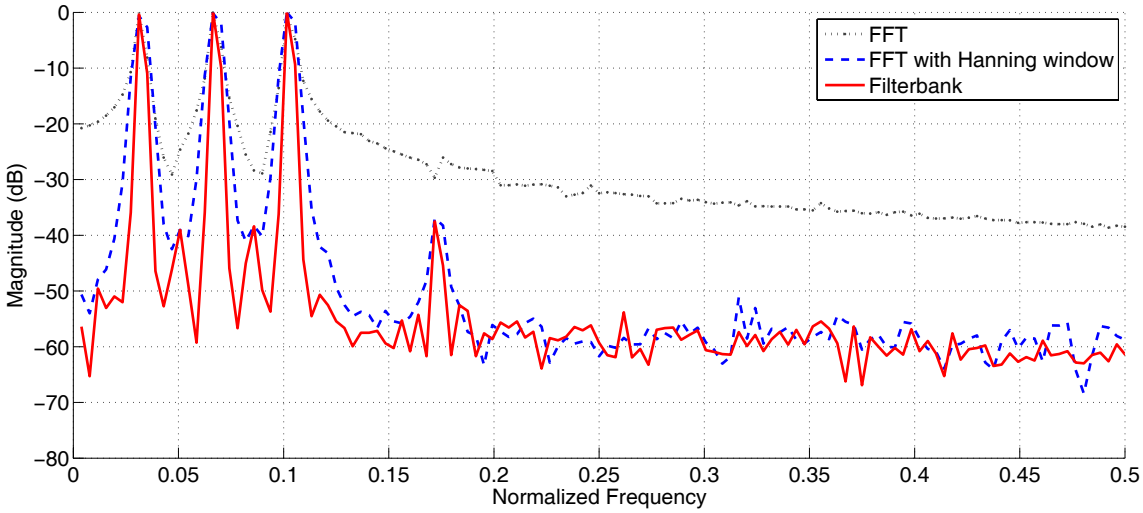

Fig. 5. Power Spectral Density (PSD) measurements by FFT, FFT with hanning window, and filterbank

Our network functionality is simulated in DEVSJava. We are able to implement a part of our networking algorithms using the HIL technique [7]. We are currently awaiting more SFF SDR platforms to progress towards the complete functional cognitive network. We will then be able to test the complete functionality of our transceiver as well as the cognition and the MAC layer implementations.

After implementing our network, we plan to use the available wireless testbed of the Flux Group [19] as a primary user network and study the effect of different types of primary user traffics on our cognitive radio as well as the effect of possible interference from the cognitive network on the legacy networks. Our proximity with the FLUX group provides us with this exciting opportunity.

ACKNOWLEDGMENTS

We would like to thank the hardware and software support that we have received from SDR Forum, Lyrtech, Texas Instruments, Xilinx, Mathworks, Greenhills, Prismtech, Zeligsoft, and Synplicity. It has been a challenge to learn this extraordinary collection of hardware and software, but indeed, a rewarding one. We are truly grateful to all supporting companies.

7. REFERENCES

[1] P. Amini, D. Palchak, X. Mao, S. Talbod, S. Akoum, and S. Abbasi, "Smart radio challenge proposal: Spectrum access for first responders," Smart Radio Challenge, Available at `http://www.ece.utah.edu/~pamini/proposal.pdf`, Tech. Rep., September 2006.

[2] "Smart radio challenge," *http://www.radiochallenge.org/*.

[3] P. Amini, R. Kempter, R. R. Chen, L. Lin, and B. Farhang-Boroujeny, "Filter bank multitone: A physical layer candidate for cognitive radios," *2005 Software Defined Radio Technical Conference*, November 2005.

[4] P. Amini, R. Kempter, and B. Farhang-Boroujeny, "A comparison of alternative filterbank multicarrier methods in cognitive radio systems," *2006 Software Defined Radio Technical Conference*, November 2006.

[5] B. Farhang-Boroujeny, *Signal Processing Techniques for Software Radio*. avilable at `http://www.ece.utah.edu/~farhang`, 2007.

[6] "Lyrtech SFF SDR development platform technical specs," Lyrtech Inc., Available at `http://www.lyrtech.com/publications/sff_sdr_dev_platform_en.pdf`, Tech. Rep., February 2007.

[7] E. Azarnasab, P. Amini, and B. Farhang-Boroujeny, "Hardware in the loop, a developement strategy for software defined radios," *Software Defined Radio Conference*, 2007.

[8] T. Weiss and F. Jondral, "Spectrum pooling: and innovative strategy for the enhancement of spectrum efficiency," *IEEE Communications Magazine*, vol. 42, no. 3, pp. S8–S14, March 2004.

[9] T. Weiss, J. Hillenbrand, A. Krohn, and F. Jondral, "Mutual interference in ofdm-based spectrum pooling systems," *IEEE 59th Vehicular Technology Conference, VTC 2004-Spring*, vol. 4, pp. 1873–1877, May 17-19 2004.

[10] S. Haykin, "Cognitive radio: Brain-empowered wireless communications," *IEEE Journal on Selected Areas in Communications*, February 2005.

[11] E. Hogenauer, "An economical class of digital filters for decimation and interpolation," *IEEE Transaction on Acoustic, Speach, and Signal Processing*, vol. 29, April 1981.

[12] S. Talbot and B. Farhang-Bouroujney, "Pulse shape filter design in digital modems employing CIC filters," *SDR Forum Technical Conference*, November 2006.

[13] B. Farhang-Boroujeny, "Cyclic equalization options in software-based radios," *SDR Technical Conference*, November 2007.

[14] The IEEE LAN/MAN Standards Committee, "802.22 wg on wireless regional area networks (WRANs)," *http://www.ieee802.org/22*.

[15] "TMS320DM6446 digital media system-on-chip," Lyrtech Inc., Available at `http://focus.ti.com/docs/prod/folders/print/tms320dm6446.html`, Tech. Rep., March 2007.

[16] C. Dick, F. Harris, and M. Rice, "Synchronization in software radios - carrier and timing recovery using fpgas," *In Proceedings of the IEEE symposium on Field-Programmable Custom Computing Machines*, 2000.

[17] "System generator for dsp reference guide," Xilinx Inc., Tech. Rep., August 2007.

[18] "TMS320C64x+ DSP Big-Endian Library Programmer's Reference," Texas Instrument, Tech. Rep., March 2006.

[19] "Emulab network simulation testbed," *http://www.emulab.net/*.