Energy Efficient Architecture Using Hardware Acceleration for Software Defined Radio Components

Chen Liu*, Omar Granados*, Rolando Duarte* and Jean Andrian*

Abstract—In order to make cognitive radio systems a practical technology to be deployed in real-world scenarios, the core Software Defined Radio (SDR) systems must meet the stringent requirements of the target application, especially in terms of performance and energy consumption for mobile platforms. In this paper we present a feasibility study of hardware acceleration as an energy-efficient implementation for SDR. We identified the amplifier function from the Software Communication Architecture (SCA) for hardware acceleration since it is one of the functions called for most frequently and it requires intensive floating-point computation. Then, we used the Virtex5 Field-Programmable Gate Array (FPGA) to perform a comparison between compiler floating-point support and the on-chip floating-point support. By enabling the on-chip floating-point unit (FPU), we obtained as high as a 2X speedup and 50% of the overall energy reduction. We achieved this with an increase of the power consumption by no more than 0.68%. This demonstrates the feasibility of the proposed approach.

Keywords—Software Communication Architecture, Software Defined Radio, Energy Efficiency, FPGA, Cognitive Radio

1. INTRODUCTION

Researchers in the wireless communications community have continuously focused on investigating and developing technologies that are aimed at alleviating the issue of growing spectrum overcrowding and improving the power and spectral efficiency of mobile devices with the objective of meeting critical deployment and commercialization constraints such as size, weight, and power (SWaP).

Examples of solutions to these three issues can be readily found in literature. Among these solutions, adaptive modulation and link adaptation techniques have been commonly used to achieve an optimal trade-off between the spectral and power efficiency of mobile systems **[1]**. In

※ This paper is an extended version of the paper, "A Case Study on a Software Communications Architecture Component for Hardware Acceleration," which appeared in the Proceedings of the 4th International Conference on Ubi-media Computing (U-Media 2011) in Sao Paulo, Brazil during July 3-4, 2011. This work was supported in part by the National Science Foundation under Grant No.ECCS-1125762. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Manuscript received July 28, 2011; accepted September 5, 2011.

Corresponding Author: Chen Liu

* Dept. of Electrical and Computer Engineering, Florida International University, Miami, Florida, USA ({cliu; ogran002; rduar002; andrianj}@fiu.edu)

fact, it has been shown that optimal adaptation can lead to significant improvements in the wireless channel capacity and reliability. That is, based on the current channel characteristics, the radio is able to select the transmission mode and protocol that yields the best performance. In addition to this, since studies performed by the Federal Communications Commission (FCC) have shown that most of the spectrum crowding is caused by inefficient allocation **[2]**, a solution involving some kind of opportunistic spectrum allocation at the node and network levels is considered to be a viable alternative to ease concerns regarding spectrum usage [3].

Mitola [4] envisioned a system that was aimed at addressing the three aforementioned mobile system challenges. This solution, known as cognitive radio (CR), includes a combination of both of the adaptive approaches discussed above. Specifically, a cognitive radio is a system that senses its operating environment and uses this information along with usage patterns in a cognitive process, whose ultimate goal is to alter the radio's parameters (such as mode, protocol, center frequency, and transmission power) in order to obtain a highly reliable communication system and to achieve efficient utilization of the radio spectrum. This so called cognitive process involves observation, learning, optimization, and decision-making **[5]**.

From the above definition, it is clear that radios whose parameters can be reconfigured through software are crucial components of cognitive radio systems. These reconfigurable devices are generally known as software defined radios (SDR). In order to make cognitive radio systems a practical and commercial technology to be deployed in real-world scenarios, and not just in academic or experimental environments, the core SDR systems must meet the performance and SWaP requirements of the target application. This implies that the system should be able to support the communication protocols of different levels of complexity. Therefore, two main characteristics that the SDR system must have are high computational performance and low energy consumption. A promising approach that offers improvements in the aforementioned areas is the use of hardware acceleration techniques on the computation-intensive signal-processing components implemented under the SDR's software architecture. In [7], Lau et al. showed that increased execution speed could be achieved by offloading some of the signal-processing functions of a general SDR system onto an FPGA-based hardware accelerator. Since the Software Communications Architecture (SCA) is the most widely accepted software platform for SDR, in [8] Millage demonstrated how the hardware acceleration of signal-processing components implemented in the SCA can be achieved using a graphics-processing unit (GPU). There, it is observed that execution time is reduced compared to those of software components running only on a general-purpose processor (GPP). Although GPUs have considerable computational advantage due to parallelization, their use as hardware accelerators in mobile devices is not very practical because of their significantly high power consumption. Therefore, in contrast with [8], in this work we propose an FPGA-based hardware acceleration platform for SCA components in order to achieve a balance between computational performance and energy consumption. We present an initial feasibility study on the implementation of such platform. Since hardware acceleration is only performed on the signal-processing portion of the code that implements a waveform component, the resulting platform is still compliant with SCA standards, thus guaranteeing that the target cognitive radio system can be readily used in current military and commercial systems without backward compatibility issues.

In Section 2, an overview of Software Defined Radio (SDR) and the Software Communications Architecture (SCA) is presented. Section 3 introduces the proposed architecture using hardware acceleration and describes the specific FPGA platform used to implement it. A study

on the energy consumption and performance of an SCA component implemented on a soft-core processor with specific hardware support is presented in Section 4. Lastly, concluding remarks are given in Section 5.

2. BACKGROUND

In this section we introduce the basic concepts of software defined radio (SDR) and the software communications architecture (SCA).

2.1 Software Defined Radio

Software defined radio (SDR) is the main enabling technology of cognitive radio nodes and networks. The term SDR refers to a radio system where the majority of the transmission and reception-path components are realized through software. The purpose behind this type of system is to allow the modification of radio waveforms and protocols with zero or minimal hardware changes. That is, alterations to parameters such as modulation format, transmission power, or transmission frequency can be done entirely through software. The block diagram of the main hardware (HW) components of an SDR system is presented in Fig. 1. In an ideal cognitive radio node, based on the measured environmental parameters, the system should be able to reconfigure the operating behavior of each hardware element of the SDR through software in order to enable dynamic spectrum access and improve system reliability. Therefore, the system must support a large variety of wireless protocols [9]. Moreover, a cognitive radio node should have the ability to learn and make such parameter-adjusting decisions based on previous experience.

It can be observed that for this type of platform to be implemented, enabling technologies such as intermediate frequency (IF) and baseband processors, analog-to-digital and digital-to-analog converters (ADC and DAC), reconfigurable components of the radio frequency (RF) front-end, and antenna elements all need to be properly chosen to comply with the requirements of the waveforms they are aimed to support.

Although choosing the most appropriate hardware platform for the SDR is critical, it is almost as important to select the right software architecture. Such architecture should provide portability among different hardware platforms. Moreover, in order to facilitate the cognitive radio system's adoption in commercial applications, the SDR's software performance bottlenecks should be minimized. That is, one has to ensure that the baseband signal processing components and the communication protocols between them are as efficiently implemented as possible. The benefits of doing so can be reflected in lower computational intensity and less energy consumption.

Although recently there has been an emergence of many SDR software architectures including GNURadio [10], Vanu Radio [11], OMG's PIM and PSM for Software Radio Components Specification [12], and NASA's Space Telecommunication Radio System (STRS) architecture

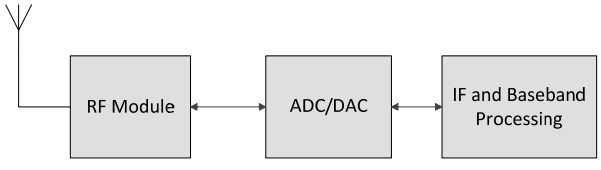

Fig. 1. Block diagram of SDR System

[13], the so-called software communications architecture (SCA) still remains as the de-facto standard of the SDR community [14]. Therefore, in this work our focus is on SCA architecture.

2.2 The Software Communications Architecture

The software communications architecture (SCA) was developed as the standard software framework for the Department of Defense's (DoD) military radio development program known as the Joint Tactical Radio System (JTRS) [15]. The introduction of this standard was mainly aimed at facilitating technology insertion, hardware abstraction, and hardware interoperability for SDR systems [16]. The structure of the SCA is presented in Fig. 2. Here, the elements labeled C1 through C4 represent the waveform components. In the SDR context, the function of each of these components is to perform a specific signal-processing task within a waveform. In SCA, all the services and interfaces available to each component are defined in the SCA core framework. From the figure it can be seen that waveform components interact with each other and with the core framework through the logical bus provided by the Common Object Request Broker Architecture (CORBA). One of the crucial features of this architecture is that it allows different applications (waveforms or waveform components) to run on distributed processing units and to communicate with each other. This is particularly useful when using only one processing unit does not satisfy the computational requirements of complex waveforms.

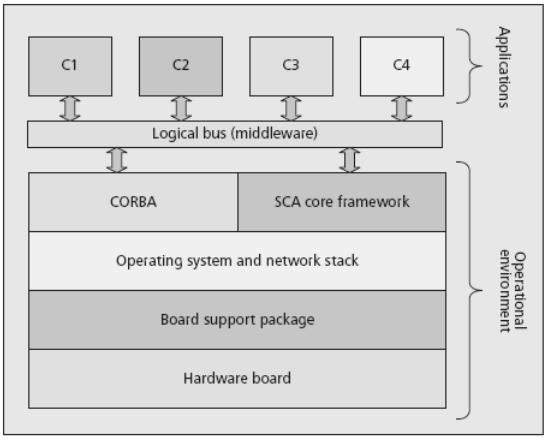

Fig. 2. Software Communications Architecture Structure [16]

3. SYSTEM ARCHITECTURE

As previously discussed, energy efficiency is of ultimate importance for mobile devices. Therefore, in this section we propose a SCA-compliant energy-efficient SDR system architecture using hardware acceleration. In addition to this, we also introduce the target hardware platform for the proposed architecture.

3.1 Proposed System Architecture for Hardware Acceleration

Because mobile devices have obvious limitations in size, weight, and power (SWaP), we pro-

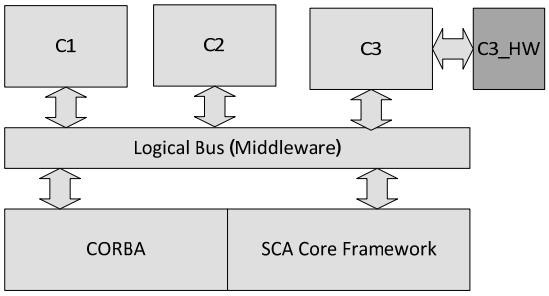

Fig. 3. Proposed SCA Architecture with Hardware Acceleration

pose the use of hardware acceleration in cognitive radio systems in order to improve their energy and computational efficiency. To take advantage of the benefits obtained through the hardware acceleration of software components, while still adhering to the SCA standard, only the code of the signal processing routine for each waveform component is accelerated in our proposed architecture as shown in Fig. 3. That is, all the core framework interfaces and CORBA services to these accelerated components remain unchanged, allowing the component to be initialized and also located by other waveform components. However, the signal-processing portion of the code is replaced by a subroutine, which calls upon the hardware-accelerated implementation. One of the implications of this process is that data stored in CORBA-type variables must be converted and loaded to the memory location of the hardware accelerator used to implement the signal-processing routine. In Fig. 3, the component to be accelerated is denoted as C3 and the accelerated signal-processing routine is C3_HW.

Since the proposed system is to be implemented using an FPGA board, in the following section a description of the characteristics of the hardware platform is provided.

3.2 Hardware Platform

We used the Virtex5 FPGA platform [17] to construct the system architecture shown in Fig. 4. FPGA is of great advantage since they can be reconfigured and reprogrammed. MicroBlaze [18] is a soft-core 32-bit RISC microprocessor operating at a clock rate of 125 MHz on Virtex5. It can be configured to have a 5-stage pipeline (performance-optimized) or 3 stage-pipeline (area-optimized), barrel shifter, integer multiplier/divider, floating-point operation, and cache support. For our design, we elect to use the 5-stage pipeline and floating-point unit (FPU) support for IEEE single-precision format. MicroBlaze can access the local memory Block RAM (BRAM) through the Local Memory Bus (LMB). Thus, it accesses the instruction and data through the Instruction Local Memory Bus (ILMB) and Data Local Memory Bus (DLMB) respectively. Every instruction or data access through the LMB takes 1 clock cycle and the LMB must only be connected from processor to local memory BRAM. In contrast, the Processor Local Bus (PLB) takes about 7 clock cycles per access. However, it allows any peripheral or custom hardware to connect to it. The PLB is bidirectional and is 32-bit wide. We also employed a timer to collect timing information; such as the number of clock cycles a specified operation takes on MicroBlaze or any other peripheral connected through the PLB.

Interrupt is used to make the system interruptible. If Microblaze needs to perform an urgent operation, then it can interrupt the current operation to serve the urgent one. RS232_UART is

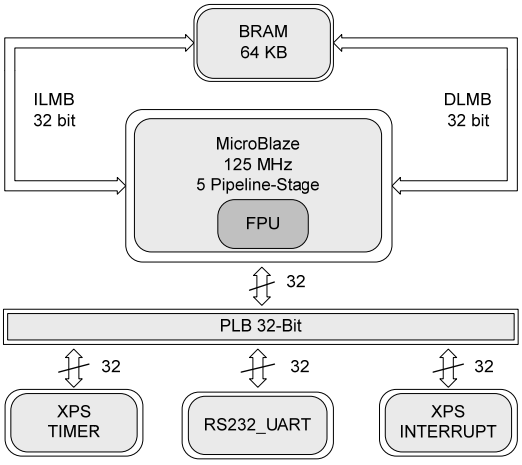

Fig. 4. System Architecture with FPU Support

used for displaying the output on the host machine. The FPGA platform is a memory-mapped system and every component is assigned a memory range. Therefore, if MicroBlaze needs to access any peripheral, it only needs to provide the address location and/or data. The address of MicroBlaze is 32-bit wide, which also defines the address space.

4. EXPERIMENTAL RESULTS

In this section we briefly go over our experiment setup and then present the experimental results.

4.1 Experimental Setup

For the following study, the loopback waveform shown in Fig. 5 was implemented in a SCA-based software architecture known as Open-Source SCA Implementation - Embedded (OSSIE). The waveform was executed and profiled using the GNU profiling tool OProfile. Based on the profiled information, it was observed that the components on which the processor spent more time (i.e. a higher percentage of samples were taken during execution) were the frame synchronizer and the IQ amplifier. This was inferred as an indication that the acceleration of these components can lead to significant performance improvement as the processor frequently accesses

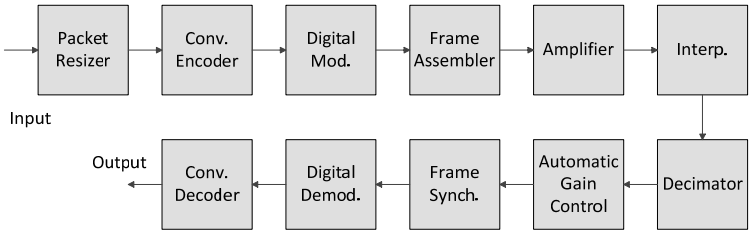

Fig. 5. SDR Test Waveform

them. Please keep in mind that purpose of this study is not only to accelerate a single amplifier function, which a modern ASIC chip will do. Instead, the choice of accelerating the amplifier component was based on noticing that this specific component is complex enough to demonstrate the feasibility of the proposed approach, which in the long run is to construct a hardware platform for the entire SCA, accelerating most of its major components.

The amplifier function accepts seven parameters to multiply the in_phase (I_in_0) and its gain (I_gain) as well as the quadrature (Q_in_0) and its gain (Q_gain). Now, I_gain and Q_gain are of floating-point, while the I_in_0 and Q_in_0 are of integer vectors. Therefore, our initial feasibility study consisted of executing the amplifier function from the SCA without the FPU incorporated in the MicroBlaze architecture over with FPU support. For the no FPU support case, the compiler translates the floating-point operations by emulating them using fixed-point operations.

4.2 Results

The experiment is conducted as follows: we execute the amplifier function on MicroBlaze with FPU support. The experiment data including the clock cycles, execution time, and energy consumption are collected with each time doubling the size of input data-sets consecutively. The data-set size ranges from 128 Bytes to 64 Kbytes (local memory BRAM maximum capacity). Next we follow the same procedure and collect the data, but this time without the floating point support.

We first enable the FPU support and run the amplifier with a data set size of 128 Bytes. It takes 17,842 clock cycles to perform the floating-point multiplication. Since the microprocessor runs at 125 MHz, the corresponding execution time is 0.1427 milliseconds. While for the case without FPU support, it takes 32,651 clock cycles to perform floating-point multiplication on a 128-Byte data set, and the execution time is 0.2612 milliseconds accordingly. As for the input size of the 64K case, it takes around 7.4 million cycles to finish the execution with FPU support and about 15 million cycles without.

The performance in terms of the number of clock cycles to finish the computation of various input sizes for two different configurations is shown in Fig. 6. The speedup is based on the ratio of the execution time of without FPU unit support over with FPU support. Hence, executing the

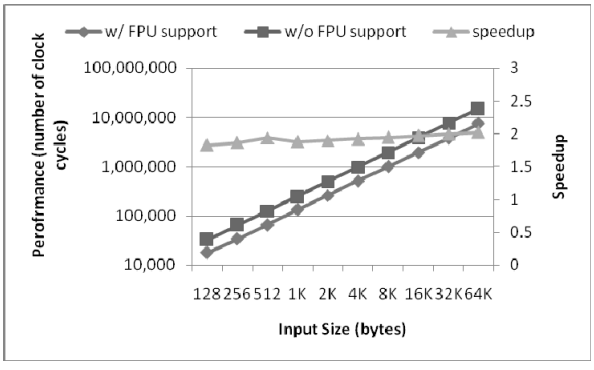

Fig. 6. Performance of FPU over no FPU Support

amplifier function with FPU support results in a speedup of 1.83 over no FPU support with an input size of 128-Byte, as shown in Fig. 6. If we continue doubling the input data set, the speedup slightly increases, reaching up to 2.02X faster. This is due to the fact the number of floating-point operations is directly proportional to the input data size. Thus, based on Amdahl's law, the percentage of the floating-point operations get increased, hence the speedup achieved over the pure software floating-point execution is more with the increase of input size.

Next, we utilized the XPower Analyzer [19] to get the power consumption of the system. Xpower Analyzer is a Xilinx tool dedicated to analyze the power consumption for a post-implemented place and routed design. It collects information about the hardware design and provides an accurate estimation about the power utilization. By using the Xpower Analyzer, the overall power consumption for the hardware system (as shown in Fig. 4) can be measured. Our results show that with FPU support the total power consumption of the design is 1.2650 watts, of which 1.1105 watts is of static power and 0.1545 watts is of dynamic power. If we disable the FPU unit, the static power is 1.1102 watts and dynamic power is 0.1463 watts, respectively, with a total power consumption of 1.2565 watts. Hence, the extra hardware corresponds to a 0.68% increase in power consumption. Adding the floating-point unit support requires more hardware and thus it will consume more power as well. Based on our analysis, the system architecture with FPU support requires 415-slice registers used as flip-flops (FFs) and 771 look-up tables (LUTs); which constitute to 21% and 40% more flip-flops and LUTs resources respectively, as compared to the system that does not use the FPU engine.

Since we get the power consumption reading, we can calculate the energy consumed during the execution period. Fig. 7 shows the energy consumption of two different configurations and the corresponding energy reduction with FPU over without FPU support. The 128-Byte data set has an energy consumption of 0.1805 mJoules with FPU support, while the energy consumed is 0.3270 mJoules without FPU support. From the figure we can observe that an energy reduction of 45% is achieved for a dataset of 128 Bytes. Recall that energy depends on both the power consumption and execution time. Since the speedup in performance we achieve is around 2X while the power overhead is less that 1%, overall we should expect the energy reduction to in-

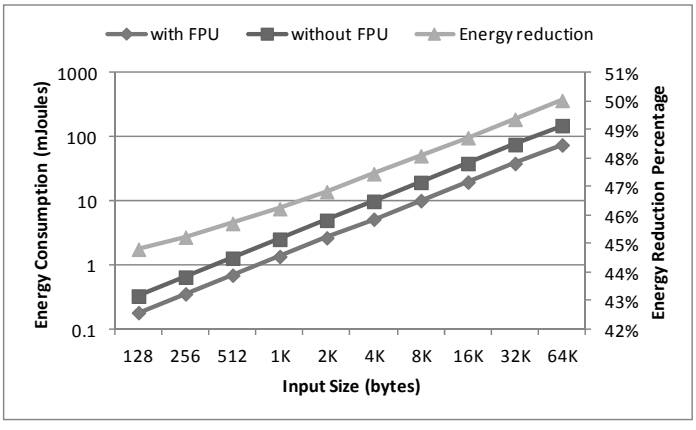

Fig. 7. Energy consumption corresponding to various input sizes and the energy reduction comparisons between with-FPU-support case and without-FPU-support case

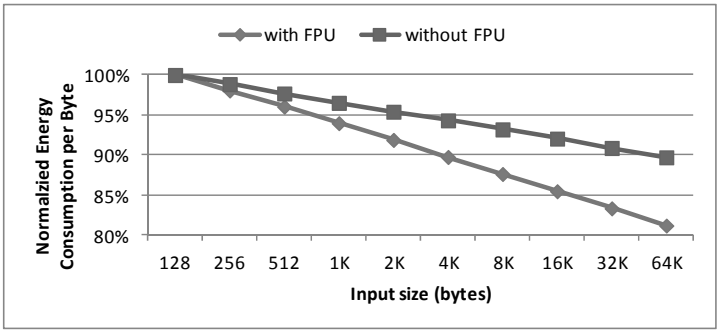

Fig. 8. Normalized Energy per Byte Consumption Comparison

crease accordingly. This observation matches the readings we get in Fig. 7. Therefore, increasing the size of dataset results in more energy reduction, reaching up to 50% when executing the amplifier function with 64 Kbytes dataset with the FPU over no FPU support, with the energy readings being 75 mJoules and 150 mJoules separately.

Fig. 8 demonstrates the comparison of the energy consumption per byte for two configurations, both of which are normalized to that of the 128-Byte input case, to evaluate the energy efficiency of the design. For the input size of 128-Bye, it takes 1.4106 uJoules for one byte with FPU support while it takes 2.5546 uJoules to do the same without FPU support. We can see that the energy per byte continuously decreases, reaching up to 20% less compared to that of 128-Byte case by increasing the input data size with FPU support. As for without FPU support, we only achieve at most 10% less energy per unit data by increasing the input size. This demonstrates that more energy efficiency per unit data is achieved for our hardware acceleration approach.

5. CONCLUSION

In this paper, a hardware acceleration technique to improve the performance of the overall SDR has been proposed. We employed a profiling tool and identified the amplifier function from the Software Communications Architecture (SCA) components as a candidate for hardware acceleration. Then using the Virtex5 FPGA platform, we compared the performance between floating-point support and without FPU support to gather information about the computational and power efficiency of the accelerated SCA component. Our results showed that execution was accelerated by up to 2X and the energy was reduced by up to 50% for an input set of 64 Kbytes. Although the extra hardware will correspond to a small increase in power consumption, in this case it will be no more than 0.68%. From our study, we can see that implementing the amplifier function using hardware acceleration can benefit the overall performance and energy efficiency of the system, making it feasible for a SDR platform, since energy efficiency is a major issue in mobile devices. As future work is concerned, we plan to extend our approach to other components of SCA and to build a hardware acceleration framework for cognitive radio systems. Our vision is to construct a heterogeneous many-core platform, with multiple hardware accelerators for most of the major components of SCA, while having several simple general-

purpose cores to handle the communication and task scheduling issues. In this way, we maximize the system performance and keep the optimal energy efficiency, thereby suiting the needs of the future mobile computing paradigm.

REFERENCES

[1] Falahati, S.; Svensson, A.; Ekman, T.; Sternad, M.; , "Adaptive modulation systems for predicted wireless channels," *Communications, IEEE Transactions on* , Vol.52, No.2, February, 2004, pp.307-316.

[2] "FCC - Notice of proposed rulemaking and order, facilitating opportunities for flexible, efficient and reliable spectrum use employing cognitive radio technologies," FCC Document ET Docket No.03-108, December, 2003.

[3] Feng Ge; Qinqin Chen; Ying Wang; Bostian, C.W.; Rondeau, T.W.; Bin Le; , "Cognitive Radio: From Spectrum Sharing to Adaptive Learning and Reconfiguration," *Aerospace Conference, 2008 IEEE* , Vol., No., 1-8 March, 2008, pp.1-10.

[4] I. Mitola, J. and J. Maguire, G. Q., "Cognitive radio: making software radios more personal," *IEEE Personal Commun. Mag.*, Vol.6, No.4, August, 1999, pp.13-18.

[5] Jondral, F.K.; , "Cognitive Radio: A Communications Engineering View," *Wireless Communications, IEEE ,* Vol.14, No.4, August, 2007, pp.28-33.

[6] Shukla, A.; Burbidge, E.; Usman, I.; , "Cognitive Radios - What are They and Why are the Military and Civil Users Interested in Them," *Antennas and Propagation, 2007. EuCAP 2007. The Second European Conference on* , Vol., No., 11-16 November, 2007, pp.1-10.

[7] D. Lau, J. Blackburn, and C. Jenkins, "Using c-to-hardware acceleration in FPGAs for waveform baseband processing," in *2006 Software Defined Radio Technical Conf. Product Exposition*, November, 2006.

[8] Millage, J.G., "GPU Integration into a Software Defined Radio Framework," Master's Thesis, Iowa State University, 2010.

[9] Haghighat, A.;, "A review on essentials and technical challenges of software defined radio," *MILCOM 2002. Proceedings* , Vol.1, No., Vol.1, 7-10 October, 2002, pp.377-382.

[10] Tucker, D.C.; Tagliarini, G.A.; , "Prototyping with GNU radio and the USRP - where to begin," *Southeastcon, 2009. SOUTHEASTCON '09. IEEE* , Vol., No., 5-8 March, 2009, pp.50-54.

[11] Chapin, J.; Lum, V.; Muir, S.; , "Experiences implementing GSM in RDL (the Vanu Radio Description Language™),"*Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE* , Vol.1, No., Vol.1, 2001, pp.213- 217.

[12] Gailliard, G.; Nicollet, E.; Sarlotte, M.; Verdier, F.; , "Transaction Level Modelling of SCA Compliant Software Defined Radio Waveforms and Platforms PIM/PSM," *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07* , Vol., No., 16-20 April, 2007, pp.1-6.

[13] Reinhart, R.C.; Johnson, S.K.; Kacpura, T.J.; Hall, C.S.; Smith, C.R.; Liebetreu, J.; , "Open Architecture Standard for NASA's Software-Defined Space Telecommunications Radio Systems," *Proceedings of the IEEE* , Vol.95, No.10, October, 2007, pp.1986-1993.

[14] P. Isomäki, N. Avessta, *An overview of software defined radio technologies*, Tech. Rep. 652, Turku Centre for Computer Science, Finland, 2004.

[15] Hayes, N.; , "The JTRS SCA specification-the past, the present, and the future," *Military Communications Conference, 2005. MILCOM 2005. IEEE* , Vol., No., Vol.5, 17-20, October, 2005, pp.2713-2719.

[16] Aguayo Gonzalez, C.R.; Dietrich, C.B.; Reed, J.H.; , "Understanding the software communications architecture," *Communications Magazine, IEEE* , Vol.47, No.9, September, 2009, pp.50-57.

[17] *Virtex-5 FPGA User Guide*. [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug190.pdf

[18] *MicroBlaze Processor Reference Guide.* [Online]. Available: http://www.xilinx.com/support/ documentation/ sw_manuals/ mb_ref_guide.pdf
[19] *Xilinx Xpower Analyzer* [Online]. Available: http://www.xilinx.com/products/design_tools/logic_ design/ verification/xpower_an.htm

Chen Liu

He received a B.E. degree in Electronics and Information Engineering from the University of Science and Technology of China in 2000, an M.S. degree in Electrical Engineering from the University of California in Riverside, California in 2002, and a Ph.D. degree in Electrical and Computer Engineering from the University of California in Irvine, California in 2008, respectively. He has been an assistant professor in the Department of Electrical and Computer Engineering at Florida International University since then. His research interests are in the area of multi-core multi-threading architecture, hardware acceleration for scientific computing, and the interaction between system software and micro-architecture.

Omar Granados

He received his B.S. and Ph.D. degrees in Electrical Engineering from Florida International University in 2005 and 2011, respectively. He is currently an adjunct professor at the same institution. His research interests include space-time coding, signal processing techniques for multiple antenna systems, software defined radio architectures, and spectrum sensing techniques for cognitive radio systems.

Rolando Duarte

He received a B.S and M.S. degree in Computer Engineering from Florida International University in 2009 and 2011, respectively. He is currently a hardware validation engineer at Intel. His research interests are in the area of Microprocessors, SoC (System-On-Chip), reconfigurable computing, and hardware acceleration for scientific computing.

Jean Andrian

He received a B.S. degree in Electrical Engineering and an M.S. degree in Engineering Physics from the Ecole Polytechnique in Montreal, QC, Canada in 1979 and 1982, separately. He received a Ph.D. degree in Electrical Engineering from University of Florida in 1985. Currently he is an associate professor in the Department of Electrical and Computer Engineering at Florida International University. His research interests are in the area of cognitive radio/software defined radio architectures, wireless communication, applications of wavelet transforms to stochastic process analysis, and image processing.