Progressive Simulation-based Design for Networked Real-time Embedded Systems

Xiaolin Hu[1], Ehsan Azarnasab[2]
[1] Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA
[2] Department of Electrical and Computer Engineering, University of Utah, Salt Lake City, UT 84112, USA

Abstract
The increasing complexity of networked real-time embedded systems asks for systematic design processes and methodologies. A progressive simulation-based design methodology that uses fast and real-time simulations at different stages of the design process is presented. This methodology views simulation as a driving force for designing and testing engineering systems and supports systematic transitions from simulation models to real system realization. We present this methodology for designing networked real-time embedded systems, and apply it to the development of a cognitive radio network on software defined radio. The design process and experiment results are presented.

1. Introduction

Simulation has long been used to support design and analysis of complex engineering systems. Fast simulations (simulations in the fast mode) allow designers to flexibly try out and analyze different design solutions without implementing the systems in hardware. Real-time simulations (simulations in the real-time mode) support designers to test the real-time features of a system that interacts with the real world and/or other hardware components. The later is especially useful for designing real-time embedded systems, such as mobile devices, manufactory automation sensors/actuators, and the networked software defined radio system presented in this book chapter. System design and implementation of these systems have been influenced by the increasing demand of new products as well as recent advances in technologies. The complexity and multidisciplinary nature of these systems make analytical modeling and analysis infeasible. However, system engineers need to assess a design before proceeding with implementation of an expensive solution. Although traditional modeling and simulation can help in this goal, its applicability has been limited due to the gap between simulation models and real implementation in hardware.

One type of real-time simulation is Hardware in the Loop (HIL) simulation, which is an advanced technique frequently used in embedded systems development [1], [2]. A HIL simulation refers to a system in which parts of a pure simulation have been replaced with actual hardware. This is based on the belief that once hardware is added to the loop, unmodeled characteristics can be investigated, and controls can be further refined. HIL is typically aimed at developing a single module in a larger system. Although it is a useful technique that can greatly support an engineering design, it does not offer a general methodology that can scale to more complex systems. Systematic design processes and methodologies are needed for designing complex systems that are characterized as large scale, networked, and tight couplings between software and hardware. Motivated by this need, we developed a progressive simulation based design (PSBD) methodology that goes beyond HIL simulation by gradually adding more hardware to the simulated system in a progressive manner and improving the co-simulated model in each level before continuing [3]. The design process of PSBD starts from all models that are simulated on computers, and proceeds by bringing real system components into the simulation to replace their virtual counterparts (models), and ends when all components are real and the final system is tested in a physical environment. Throughout this process, model continuity is emphasized and the simulation model is continually updated whenever new design details are revealed. Several distributed robotic systems have been developed following the principles of the PSBD methodology (see e.g., [4], [5], [6]).

This book chapter presents the progressive simulation-based design for networked real-time embedded systems. We give an overview of the PSBD methodology and show a bifurcated design process that implements progressive simulation-based design for individual embedded devices and the networked embedded system. We then apply the PSBD methodology to the design of a networked Software Defined Radio (SDR) system. SDR technology refers to a radio communication system capable of transmitting and receiving different modulated signals across a large frequency spectrum using software programmable hardware [7]. SDR boards often have a base-band processor for computation, field-programmable-gate-array (FPGA) for fast parallel processing, and radio frequency (RF) frontend for wireless communication. SDR gives modem designers a great opportunity to build complex modems by programming the, previously hardware, components of the radio. This replacement of hardware by software also intensifies the effectiveness of the model continuity approach of PSBD, as the individual system components are code modules to be developed and tested along the design process. The advantage of PSBD becomes more explicit when multiple SDR nodes should collaborate to form a network, hence the interaction of many complex subsystems should be engineered for best performance of the whole system. We show how the progressive simulation-based design is applied to a single cognitive modem as well as a cognitive radio network. This book chapter extends previous work on the PSBD methodology [3] and the case study example on SDR [8]. We aim to show that the progressive simulation-based design is an effective methodology that can be applied to a wide range of networked real-time embedded systems and engineering applications.

The modeling and simulation environment that supports this work is based on DEVS (Discrete Event System Specification) [9]. DEVS is a formalism derived from generic dynamic systems theory. It has well-defined concepts of coupling of components, hierarchical, modular model construction, and an object-oriented substrate supporting repository reuse. There are two kinds of models in DEVS: atomic model and coupled model. An atomic model is a basic component. It has input/output ports to represent its interface, state transition functions, time advance function, and output function to specify its dynamic behavior. A coupled model tells how to couple several component models together to form a new model. This latter model can itself be employed as a component in a larger coupled model, thus giving rise to hierarchical construction. DEVS models time explicitly. This makes it possible to study timeliness, which is an essential property of real-time systems. DEVS is not just a theoretical framework, as it has been widely implemented and used as a practical simulation tool in a variety of implementations [10], [11]. The DEVSJAVA environment [10] is used in this work to support fast and real-time simulations in the design process. More information about the DEVS formalism can be found in [9]. We note that although the DEVS simulation environment is used in this work, the presented PSBD methodology is general and can be realized using other modeling and simulation environments.

This book chapter is organized as follows. Section 2 gives an overview of the progressive simulation-based design methodology. Section 3 describes the cognitive radio case study example and highlights some of the design challenges of the system. Section 4 describes the progressive simulation-based design procedure when applied to the cognitive radio system. The design starts with a single radio system, and then proceeds to the design of the cognitive radio network. Some design results are presented. Section 5 concludes this work.

2. Progressive Simulation-based Design (PSBD) for Networked Real-time Embedded Systems

2.1 PSBD Overview

The progressive simulation-based design views simulation as the driving force for designing and testing engineering systems. It provides a design process that explicitly focuses on systematic transitions from simulation models to real system realization. As shown in Figure 1, the design process consists of three stages, each of which is characterized by the types of entities (virtual or real) that are involved. The first stage is *conventional simulation* (in fast simulation mode), where simulation is carried out using all models. A major task of this stage is to develop the system model based on knowledge and assumptions about the real system's hardware and operating environment. Often discrepancies between simulation models and real system components exist. These discrepancies cause the models to result in undesirable behavior when applied to real system in a physical environment. To reveal such design discrepancies, the next stage of the design process is *virtual environment simulation* (in real-time simulation mode), where simulation-based study is carried out in a virtual testing environment using combined models and real system components. This stage brings simulation-based study one step closer to the reality by including real system components into the simulation. The goal is to use real system components to reveal overlooked design details, and thus help designers to improve the control models/algorithms under development. The final stage is *real system experiment*, where the real system is tested in a physical environment.

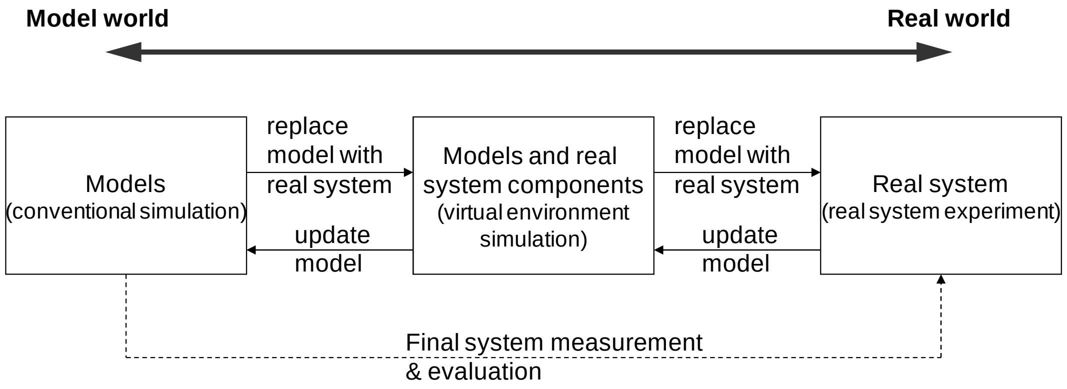

Model world
Real world
Models
(conventional simulation)
replace model with real system
update model
Models and real system components
(virtual environment simulation)
replace model with real system
update model
Real system
(real system experiment)
Final system measurement & evaluation

Figure 1: Progressive Simulation-Based design (PSBD) Methodology

Along this design process, the PSBD methodology emphasizes two parallel activities in a progressive manner: *replace models with real system components*, and *update models*. As the design moves forward, real system components are gradually brought into the simulation to replace models. Simulations with these real system components allow designers to validate their design assumptions and to reveal new design details. Such information is fed backward to the previous stages to update the models if needed. The updated model will then be used for follow-on design and test. This activity of model update allows designers to maintain a coherent model of the system under development. Thus at the end of the design, not only the system is realized and tested, but also a system model that faithfully represents the system is developed. This system model can support final system measurement and evaluation (shown by the dashed line in Figure 1), as well as serve other purposes such as system maintenance and future development. It is important to note that each design stage is a dynamical evolving process by itself. For example, during the conventional simulation stage, it is common for designers to start from high level models and then refine them to more detailed models. Similarly, the virtual environment simulation stage that involves combined models and real system components typically includes multiple phases too, e.g., to start with replacing one real system component first and then gradually add more.

Two important features of the PSBD methodology are *model continuity* and *virtual environment simulation* that supports simulation-based test with combined virtual and real system components. Model continuity refers to the ability to transition as much as possible of a model specification through the stages of a development process. For real-time embedded systems, we restrict model continuity to the models (software components) that implement the system's real-time control. This means the control models of a real-time embedded system are designed, analyzed, and tested by simulation methods, and then smoothly transitioned from simulation to hardware execution in the physical environment [4], [6]. To support model continuity it is necessary to develop system models and run simulation-based tests in a systematic way. A modular model design and well-defined interfaces are needed to ensure the control models to work with both real and simulated hardware at different stages of the design process [4], [6]. The virtual environment simulation provides a virtual testing environment by using combined real and virtual system components. It bridges the gap between conventional simulations that use all models and real system experiments that use all real system components. To support virtual environment simulation techniques need to be developed to synchronize the real and virtual system components. This includes allowing the real and virtual components to "sense" each others' existence (see [12] for an example of how real and virtual robots are synchronized with each other). Meanwhile, time synchronization is also important. Since real hardware components are included in the simulation-based study, real-time simulations are needed to support the virtual testing environment.

2.2 A Bifurcated Design Process for Networked Real-time Embedded Systems

Networked real-time embedded systems are characterized by a network of embedded devices interacting with each other and the tight couplings between software and hardware of those devices. Each of these devices is referred to as a *node* in this chapter. When commercial-off-the-shelf (COTS) nodes are not used, design of networked real-time embedded systems needs to design both the individual nodes and the networked system as a whole. Within this context, the PSBD methodology described above is elaborated to include a bifurcated design process as shown in Figure 2. The bifurcated design process explicitly differentiates the design of a single node (the bottom route in Figure 2) and the design of the networked system (the top route in Figure 2). The former mainly concerns designing the different functional modules, such as sensing, modulation/demodulation and channel coding, of the embedded device. The later focuses on how the multiple nodes work together as a whole, including designing and improving the communication protocols and the cooperative strategies among the nodes. Despite the different design focuses, both designs follow the progressive simulation-based design process that starts from models and gradually adding more real system components.

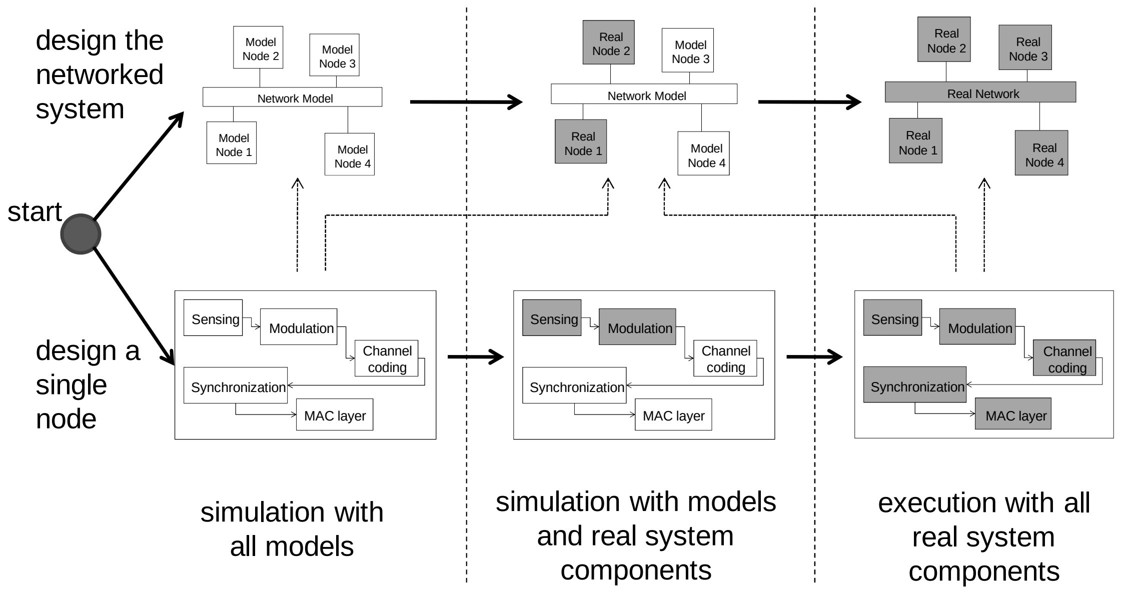


Figure 2: A Bifurcated Design Process for Networked Real-time Embedded System

In Figure 2, the models are shown as white boxes; the real system components are shown as grey boxes. The design starts and bifurcates into two routes: the design of a single node and the design of the networked system. For designing a single node, the first step is to model the functional modules of the embedded device and simulate how they work together to fulfill the functionality of the device. Then hardware components are brought into the design and hardware-in-the-loop simulations are conducted to test how well the designed modules and algorithmic code work with the hardware. This proceeds in a stepwise fashion as more and more hardware components are included. Eventually all the code modules are implemented in the hardware and the embedded node is tested with all hardware components. For designing the networked system, the first step is to develop individual node models, as well as a model of the communicating network. The individual node model can reuse the model from designing the single node (as indicated by the dotted arrow in Figure 2). Alternatively, a different node model at a higher abstraction level (e.g., without including all the details of the devices' functional modules) can be used. Simulations with these models allow designers to test the networked system in the model world. The next step is to gradually include real device nodes into the simulation-based study to conduct virtual environment simulations. The real device nodes are either the ones from the single node design or COTS nodes. This continues until the whole system is realized and all designed nodes are tested in a real network. It is important to note that the bifurcated design process provides a systematic view for designing general networked embedded systems. For a specific application, the design process can be tailored (e.g., some design stages are elaborated while others are skipped) to fit the specific design needs of the application. In Section 4, we provide an example of designing a cognitive radio system by starting from the design of a single radio modem, and then proceeding to the design of a cognitive radio network. The single radio modem is also designed in progressive manner, where different functional modules are gradually implemented/tested in the real hardware while other modules are provided by simulation models.

The progressive simulation-based design brings several advantages when designing complex networked engineering systems. Some of them are shared by the traditional HIL simulation. For example, it brings simulation-based study one-step closer to the reality to provide useful information for designers. It also increases designer’s confidence about how the final system is going to work. However, the PSBD goes beyond that by emphasizing a systematical design process that gradually adds more real system components to replace simulation models. The virtual environment simulation provides the flexibility for experimenting with a design in a virtual testing environment. It allows designers to use several, instead of all, real nodes to carry out system-wide test of a networked system. This is especially useful for large-scale networked real-time embedded systems whose complexity and scale severely limit experimentations in a physical environment using all real nodes. As the scale of these systems increases, so does their design and test complexity. It is the intension of PSBD to systematically handle such design complexity in a progressive manner.

3. Background on Cognitive Radio Design

The scarcity of unallocated frequency spectrum and the need for coexistence of different radios in the shared unlicensed bands highlight the importance of the next generation of radios with dynamic spectrum access. Traditionally spectrum is assigned to legacy devices, also called Primary Users (PUs). However licensed frequency bands are rarely used everywhere [13], [14], and overtime lead to spectrum holes in time and space. To address this underutilization Federal Communications Commission (FCC) has loosened the regulation to allow secondary users (SUs) to share some previously dedicated bands subject to minimal interference to legacy devices of the band [15]. Based on this definition of coexistence, SUs try to dynamically fill the spectrum holes over time and space [14], thus forming a cognitive radio (CR) [16]. For example when a TV station (which is a PU of some frequency band) is not broadcasting or is in a location far from any TV broadcasting, SUs can instead use the spectrum in an opportunistic fashion. Spectrum access for first responders in disaster scenario, where many wireless devices are active, is another application of cognitive radio [17].

A typical cognitive radio network consists of multiple secondary users (SUs) that coexist with primary users (PUs) of a shared spectrum. PUs have priority accesses to the spectrum over SUs, that is, SUs should give up the spectrum when PUs begin transmission. The SU network should be designed to aggregate more of the available bandwidth subject to minimum interference with the PUs. The hidden terminal problem should be solved to minimize interference. For this purpose, SUs should form a cognitive radio network to sense the presence of active PUs and dynamically adapt to a suitable frequency resulting in little or no interference with PUs. They collaboratively sense the spectrum and decide which part of the spectrum is available to them. Collaborative sensing involves signaling/reporting through a (possibly narrow band) control/reporting channel. To maximize the bandwidth efficiency of the SU network while minimizing interference with PUs, this signaling should be done in a reliable manner and in a minimum span of time. In addition to this requirement, it is desirable to have a system operating among radios from different vendors and modulations and protocols. Coexistence and interoperability are two major design goals for a cognitive radio network. Software defined radio (SDR) technology best satisfies the required flexibility of cognitive radio, and thus is often used to implement cognitive radio network [18], [19]. A SU that is realized on a SDR hardware board is called a cognitive modem. Design of a cognitive radio network needs to design the individual cognitive modems and the cognitive radio network as a whole.

The complexity of designing an individual cognitive modem is due to the many functional components and their complicated mutual interactions. These components can be implemented in parallel to expedite the manufacturing process. Parallelism can be achieved by simulating the whole system for a functional component that is being developed. Here the simulation closes the loop of system integration and provides a regression testing environment. In addition, as is common in system engineering design, not all hardware is available at the beginning of the project. Therefore, simulation is necessary in order to start the design with partial hardware that is available. The rest of the functional components are provided by simulation. In our project, at first we received only one baseband module of SDR, capable of processing only the Digital Signal Processor (DSP) algorithms such as polyphase implementation of filterbank for channel sensing. We tested our sensing algorithm using a simulated fading channel and optimized it to some extent. The high dynamic range of filterbank sensing better detects low power PU, thus it is less likely to interfere with legacy users. By the time we developed our fixed point sensing method, we received one RF frontend and applied sensing on real wireless channel to replace the previously simulated channel. The complexity of designing the cognitive radio network lies in the potentially large number of SUs and PUs that influence each other. By simulating the rest of the network before more hardware was available, we were able to test a cognitive radio network of many SUs when only one SDR board was available. Below we present how the progressive-based simulation design is applied to the design of both individual cognitive modems and the cognitive radio network. In this project, the code running on the hardware board was compiled by the Code Composer Studio (optimization level). The compiled code was then uploaded to the Small Form Factor (SFF) SDR [20] hardware platform provided by Lyrtech and Texas Instruments.

4. PSBD Of the Cognitive Radio Network

4.1 Design procedure and implementation environment

The design starts from a single cognitive modem, and then proceeds to the cognitive radio network. As part of the simulation-based testing environment, simulated PUs generate data traffic based on a model of the application layer for realistic PUs on particular frequency channels. Depending on the channel the traffic model can be a constant bit rate (CBR), burst data or a Poisson process. For example internet traffic is often modeled as bursts of data transfer, CBR is a good model for TDMA networks such as voice traffic, and Poisson process is a generic model of data arrival. Simulated SUs not only generate data traffic, but also are responsible for handling signaling packets over control channel of the network.

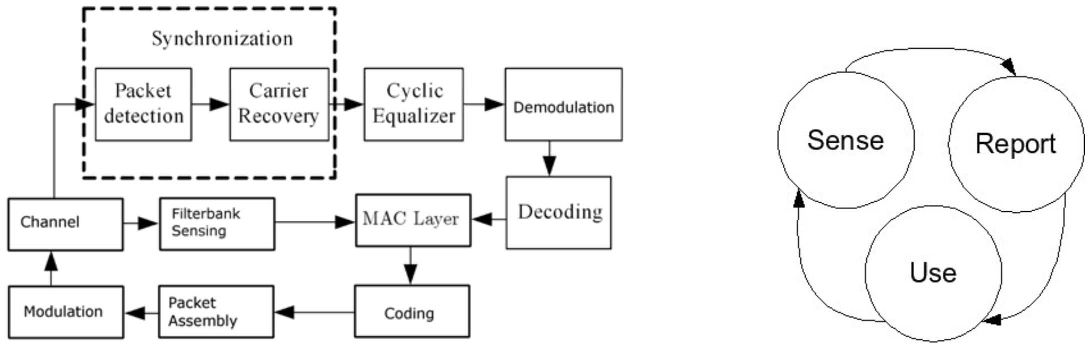
In the first stage of PSBD, we implement a single transceiver (transmitter and receiver radio). To start with the traditional modeling and simulation we begin with MATLAB simulation of generic transceiver modem and a generic channel. The goal is to find out the initial parameters for required bit error rate (BER) of the system. A complete transceiver is first simulated in MATLAB. Then the hardware board is included and the virtual environment simulation (also referred to as co-simulation) is carried out in DEVSJAVA. In this stage, individual and combined signal processing and communications algorithms are substituted by those running on the board. To take advantage of some utilities (such as the mathematical model of random fading of multipath channel) provided by Simulink, we reused some of the MATLAB code in DEVSJAVA. To do this, we used compiled MATLAB algorithms inside DEVSJAVA whenever a mathematical model for a subsystem was required. The MATLAB Builder for Java can build Java libraries from MATLAB functions and have them ready to use in DEVSJAVA. In our project, the MATLAB m-files developed for MATLAB simulation of the channel, some MATLAB visualization methods, and also the hardware-interfacing MEX files were compiled to

Java methods, which were invoked later inside the DEVSJAVA simulation. The embedded MATLAB code inside DEVSJAVA helps in generating precise communication-specific models such as traffic and random fading of multipath channel. Not only that MATLAB is very convenient and widely used when modeling complicated mathematical models of communication channels, but also we can reuse the MATLAB code written for this purpose beforehand. In this way, we were able to avoid re-implementing many the mathematical libraries inside Java, and instead we focus on higher level modeling in DEVSJAVA. The integration of DEVS with MATLAB gives designers the opportunity of decoupling the cognitive radio model from the underlying (often very complicated) math.

While MATLAB is good at mathematical modeling, it is not straightforward to model the event-driven nature of interconnected components. On the other hand DEVSJAVA can naturally support network modeling, and thus is used to model the network of cognitive radio nodes. After an initial cognitive modem is implemented (as one SU), the next step is to design the cognitive radio network. At first, a complete simulation of a cognitive radio network with SUs and PUs is developed in DEVSJAVA. The conventional simulation simulates the activities of channel assignments for SUs and PUs. An SU acting as Secondary Base (SB) station model receives the sensing information from all SUs and compiles channel state information. In our system, we have distributed sensing in order to avoid hidden node problem as much as possible. A vector signal generator emulates the traffic of simulated PUs and simulated SUs on the real channel. In modeling the network, cognitive radio nodes are modeled by DEVS atomic models and the network is modeled as a DEVS coupled model. The scheduled messages passing between the DEVS atomic models carry packets with different lengths. Based on the propagation rules, the channel model reschedules the messages to the receiver. Note that DEVS modeling is homomorphic. This means the cognitive radio node could be modeled as a coupled model itself including its functional component models. The hierarchical model construction of DEVS can help in modeling the internal interactions of subsystems of a single cognitive modem as well as the interactions of multiple cognitive modems in a network. During the course of PSBD, we have replaced two of the simulated SUs with real cognitive radios in order to carry out virtual environment test. In the setup of the virtual environment simulation, the real nodes use Filterbanks to detect the presence of the PUs and send the sensing information to the SB which is simulated on a PC. In this way, we were able to develop the network and to carry out system-wide test without waiting for all hardware to be available. Below we describe the two design routes in detail.

4.2 Design a Single Cognitive Modem

Based on the PSBD methodology a model of the cognitive modem is developed at the suitable abstraction level according to the required details and accuracy. This model includes the inter-connected functional modules of the cognitive modem. First, we start from a conventional simulation of the system in a simulated operating environment. In this example the operating environment is defined by the frequency usage and channel and is changed as a result of both PU and SU transmissions and different channel fading parameters. The cognitive modem model is simulated in Simulink to test for required bit error rate [18]. Figure 3(a) shows the model of the cognitive modem, which includes multiple functional modules. This model is a transceiver that interacts with a communication channel. The MAC layer module is the core of this model. The cognition algorithm inside the MAC layer module is responsible for compiling the spectral information sensed by the filterbank sensing module in order to provide a utility function. This utility function is utilized to determine the channels that should be used and the duration for which those channels are valid. The Filterbank sensing [7], [17] module is applied directly on the channel in order to detect the power spectral density. The synchronization module detects the incoming packets while correcting the residual carrier offset, before passing data to an equalizer to compensate for the effect of channel on data. The communication channel in the simulation is considered as an additive white Gaussian (AWGN). Conventional simulations based on this model allowed us to test the system logic and to check if the inter-connected modules effectively work together to fulfill the required functionality.



(a) Cognitive modem model (b) the three phases

Figure 3: Simulation model of a single cognitive modem and the three phases within each cycle of the cognitive radio.

The conventional simulation, however, is insufficient to design and test the features that are heavily influenced by the implementing hardware. For example, tuning algorithm/protocol parameters to achieve optimal result in singling and synchronization will depend on the hardware's real-time properties that cannot be pre-determined without using real hardware. In our design, the MAC layer protocol is decentralized and consists of three phases as shown in Figure 3(b). In the *sense* phase all of the nodes sense the channel (it takes Ts). In the *report* phase all the nodes report their cognition results over a control channel (it takes Tr). And in the *use* phase all the nodes start using available channels that are determined by cognition algorithm (for Td duration). The purpose of the reporting period is to allow all SUs to know exactly which channels are occupied by PUs, and how the available channels are assigned to the SUs. Note that only the transmission time (Td) contributes to the cognitive radio bandwidth and therefore Ts and Tr should be minimized. An important task in designing the cognitive modem is to optimize the MAC layer so that reporting time (Tr) is minimized [21]. Also we should note that Td cannot be very long because a PU may return back to the spectrum and thus SUs should leave the occupied channels to the primary users. The time that SUs are permitted to transmit (Td) is thus determined by statistics of PU arrivals in the bands of interest. To better design the features such as the MAC layer protocol, the next stage is to include real hardware components into the simulation for virtual environment simulation.

It is challenging to decide at what extent simulation-assisted design should be involved and to what level the system should be decomposed. A single cognitive modem comprises many individual modules each of which is a candidate for co-simulation along with other simulated objects and the environment. Figure 4 shows how a virtual environment simulation is carried out in designing a single modem. In this figure, one SDR hardware board is used. Since sensing the medium is the most critical part of cognitive radio we implemented this part in the earliest stage of the virtual environment simulation as shown in Figure 4. To test the sensing module we emulate PU traffic on different frequency bands using a wide-band vector signal generator, while adjusting some design parameters such as analogue to digital converter (ADC) gains, frequency axis margins, and power threshold of PU detection. The transmitted traffic of PU by the signal generator is a multi-band waveform which is generated using a MATLAB script and uploaded to the device with Agilent Waveform Download Assistant via Ethernet connection. All simulated objects are running on the same PC, while the SDR board (running implemented components) is connected to that PC via an Ethernet cable.

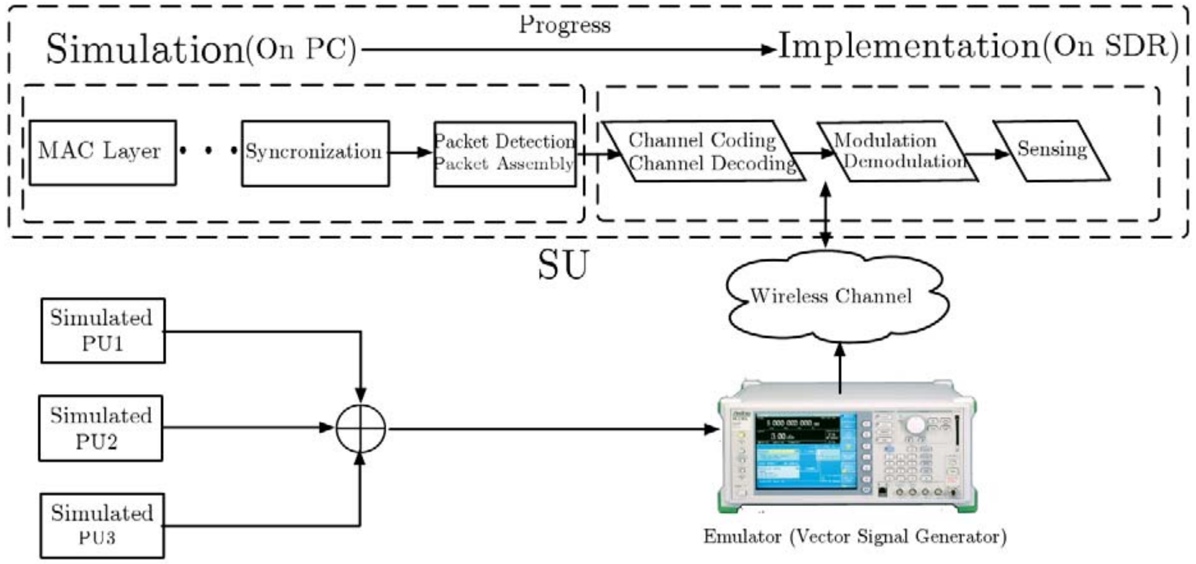


Figure 4: Progressive Simulation-Based design (PSBD) of a single cognitive modem. The implementation starts from the sensing module and progressively more of the simulated models (left dotted box) are implemented (right dotted box). The rectangles are DEVS models simulated on PC, and parallelograms are implemented modules on the SDR board.

4.3 Design a Cognitive Radio Network

After the cognitive modem is implemented, we design the cognitive radio network using the developed cognitive modem. Similar to the design of a single modem, we start from conventional simulation of the cognitive network using the model of cognitive modem developed in the previous stage. In our design, first we consider a centralized network where some SUs are assigned to act as Secondary Base (SB) stations. This approach leads to a global common plane architecture for signaling and control to avoid hidden PU node problem. This is essential in any cognitive radio network to avoid interfering with existing PUs. SB is in charge of synchronization and channel assignment. It compiles the channel state information based on sensing results of SUs and uses this knowledge to rank different part of spectrum and blacklist some active channels. After the role of each cognitive radio node is assigned, we model the cognitive network, which includes multiple PU and SU models, a Secondary Base (SB) model, and a Channel model that simulates the features of the wireless channel. Conventional simulations using this network model allowed us to test if the SUs can successfully detect non-used bands and to dynamically use them without interfering with the PUs. A sample simulation result is provided in Section 4.4.

The next stage is to introduce real cognitive modems (SUs) in the simulation. The virtual environment simulation includes real SUs, real PUs, simulated SUs, simulated PUs and simulated channel (used only between simulated SUs). Figure 5 shows the setup of a virtual environment simulation and the interconnections among simulated, emulated, and real components. In Figure 5, two real SUs (Lyrtech SFF SDR boards, denoted as *Real SU1* and *Real SU2*) and one real PU (two-way radio, denoted as *Real PU1*) are used. The simulated SUs and PUs are emulated using a vector signal generator to generate spectral energy on the bands of the simulated agents. Therefore, the real SUs would see the channel as if the simulated users exist. To test the cognitive modem against the generic PU channel usage pattern, the traffic of PU (being simulated) is known and programmed in the PU model. The emulation of PU for real SU is necessary to test the sensing mechanism of real SUs. The simulated SUs use the simulated channel and simulated traffic directly from the simulated PUs. The simulated channel is also used between two simulated SUs when transmitting a packet. A fading channel and different exponential PU traffics (with various mean for each channel) is implemented in MATLAB and compiled to be invoked by DEVSJAVA. Note that the emulator is considered as part of the testing environment and does not belong to the system model.

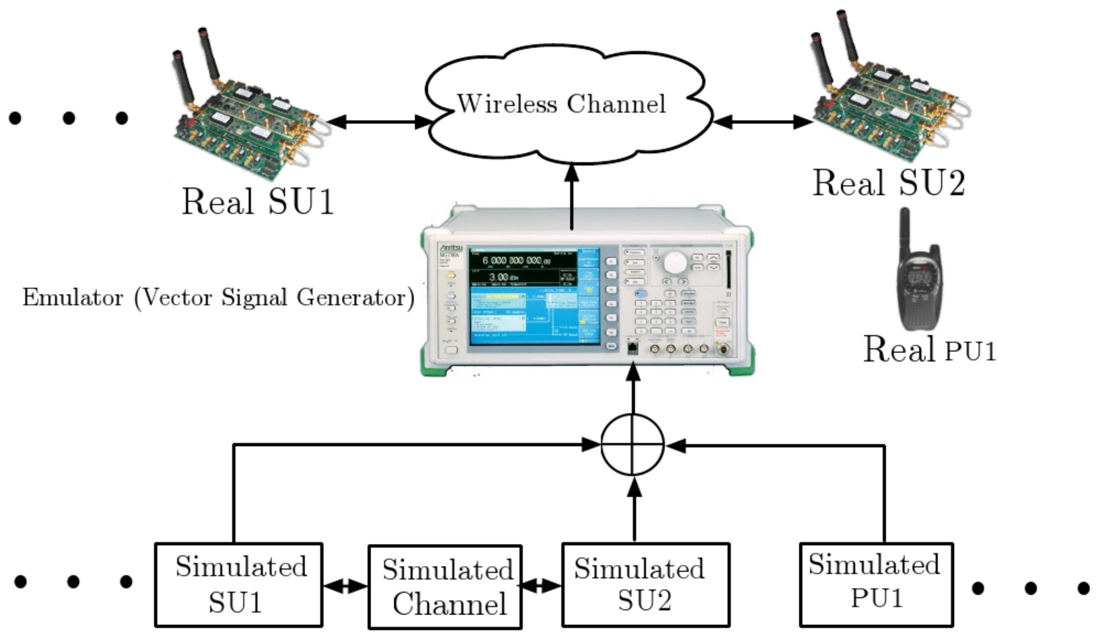


Figure 5: Virtual environment simulation of a cognitive radio network. The rectangles are DEVS models.

To carry out virtual environment simulation, it is important to set up the environment so the real and simulated nodes can “sense” each other’s existence. For example, when a real SU uses a band of the channel, the simulated SUs need to know the band has been occupied (by the real SU). The reverse is true too. Thus a two-way communication is needed between the real SUs and the PC that host the simulated SUs. In our system as shown in Figure 5, the emulator conveys the simulated environment and broadcasts the information sensed by the real SUs. To support the communication from a real SU to the PC, Ethernet cable (not shown in Figure 5) is used to connect the real SU with the PC. During the “report” stage of the MAC protocol, a real SU sends its information to the PC through the Ethernet cable. Note that the Ethernet cable is not needed in real system test, where wireless communication is used. To synchronize the real and simulated SUs (meaning to allow the real and simulated SUs to know each other's existence) in a systematic way, each real SU has a "shadow model" on the PC. This is similar to the robot-in-the-loop (RIL) simulation [4], [5], [6] where each real robot has a counterpart robot model in the simulation environment on the PC. The shallow model is responsible for receiving report information from the real SU and then passes that information to the Channel model in the same way as other simulated SUs do.

Figure 6 shows a DEVS model of cognitive radio network including two PUs, two SUs and one SB in the virtual environment simulation. One of the secondary users (SU2), the two primary users (PU1 and PU2) and also the secondary base (SB) station are simulated on the host PC. The other secondary user (SU1) has hardware implementation thus is emulated using the co-simulation engine. The SU1 shown in Figure 6 is the shadow model of the real SU1. Using the message passing mechanism of DEVS between the models, in which messages are external events, and also exploiting the time triggered message generation inside the modeled nodes, which are internal events, the simulation of the network was implemented. We used immediate messages for passing parameters between the models while time scheduled messages to pass the data-carrying binary signals (longer packet is scheduled for later time). For the simulated nodes, after a transmitter sends a packet of data, the Channel model passes the binary signal, along with the carrier frequency and other required parameters to a MATLAB code which simulates a complete transmitter, channel, and receiver. The MATLAB code for the transmitter includes source coding, base-band modulation, upsampling and RF modulation, channel, downsampling, etc. For the receiver, the necessary functions are also developed in MATLAB and the data which might have error is passed to the node in DEVS. The real cognitive modem SU1 relies on its embedded software for data transmission and the co-simulation engine handles the interface between the shadow model SU1 and the real SU1.

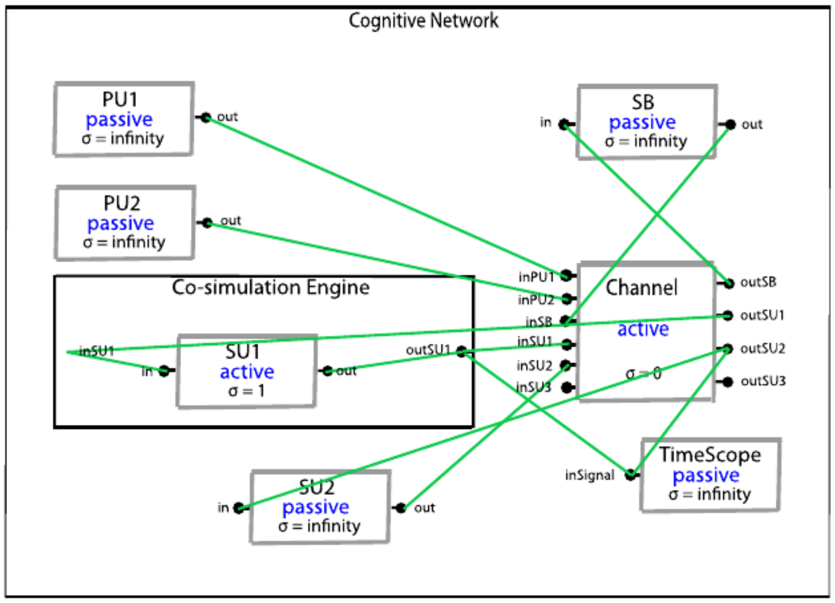


Figure 6: A cognitive network DEVS model with two Primary Users (PUs) and two Secondary Users (SUs) and one Secondary Base station (SB). SU1 is implemented on an SFF SDR board and emulated along with the other nodes.

4.4 Experiment Results

Based on the progressive simulation-based design methodology we designed and implemented three cognitive SUs as follows. After initial simulation of one SU in MATLAB, we agreed on certain technical parameters. Then inside DEVSJAVA along with a generic simulation of one PU, we improved our SDR implementation of one SU along with its model. In the next step, we added one more SDR-based real SU and more PU nodes along with one simulated SB to our network [18], [22]. The simulated SB is in charge of transmitting channel assignment over the reporting channel. It combines the individual sensing result of the SU and assigns channels to them upon their request.

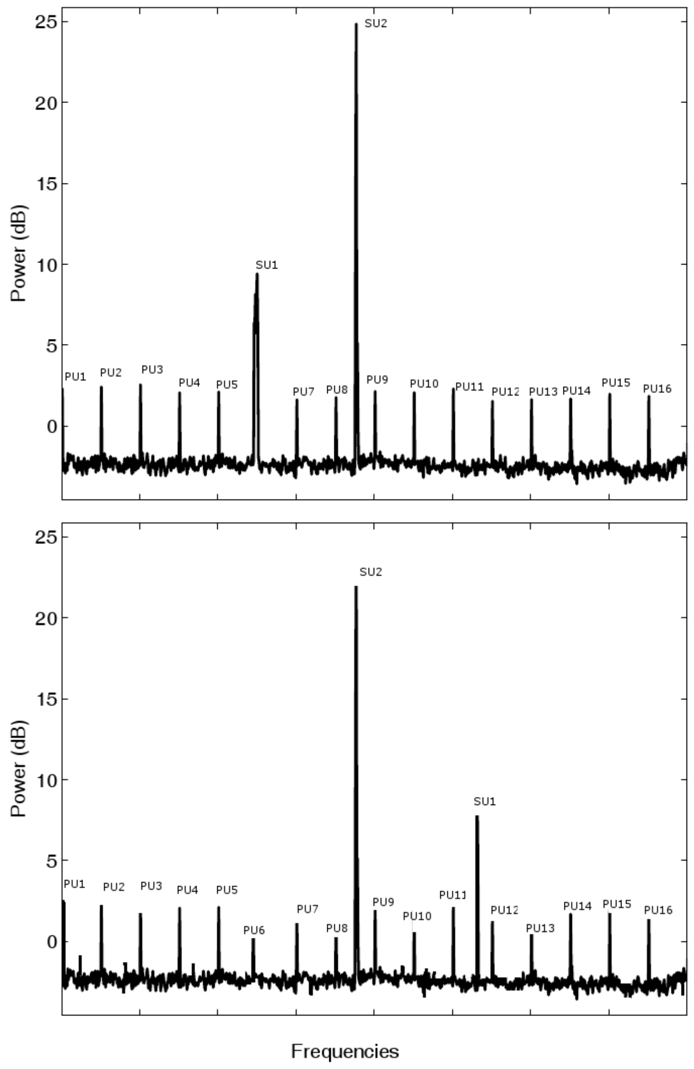
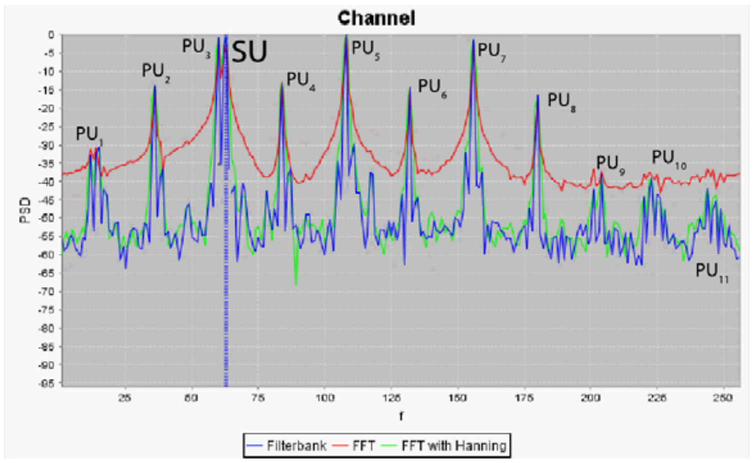


Figure 7: Simulation results using all models

Figure 7 shows the result of a conventional simulation with all-simulated SUs and PUs. In this simulation, 2 SUs and 16 PUs were used. A flat fading wireless channel was used for simulation with white noise, and filter-banks were used for sensing. The simulation demonstrates how a SU can dynamically detect and change its band in order to avoid interfering with a PU. As shown in the top part of Figure 7, initially PU6 was not present (its power was below the noise temperature) and SU1 was using its band. Then as shown in the bottom part of Figure 7, as soon as PU6 returned SU1 dynamically found another unused band that was less likely to have a PU any time soon as described in [18] and changed to the new band. In this simulation, SU2 used a band which no other PU was using. Thus SU2 did not need to change its band.



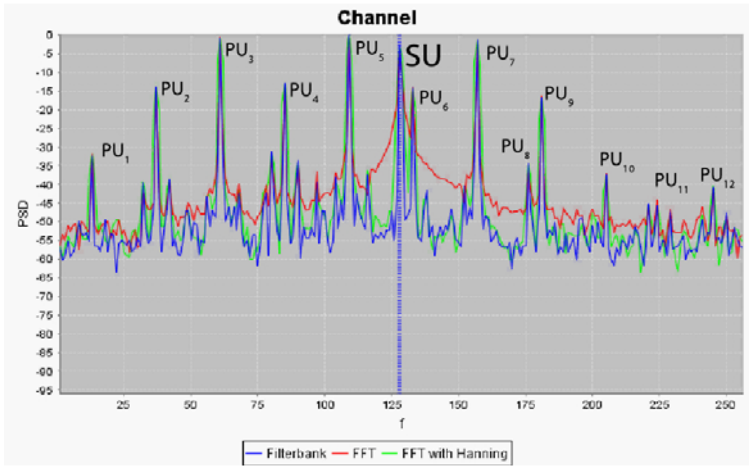


Figure 8: Simulation results using one real SU among many PUs

Figure 8 shows the result of a virtual environment simulation that included one real SU and 12 simulated PUs. In this experiment, we implemented three sensing methods including filterbank, FFT, and FFT with Hanning and compared them. The collected data was the power spectral density (PSD) sensed from a densely populated spectrum for 256 subcarriers in an experiment. This figure shows that SB reassigned a new frequency band to the only SU when PU3 was detected in the adjacent frequency of the previous carrier (the dotted vertical line in the top figure). As a result, the SU changed to a new band (shown by the vertical dotted line in the bottom figure). The top and bottom figures show before and after the PU3 detection respectively. The continuous transmission of voice during this action also proved the success of the frequency hopping. This experiment shows that the cognition modem was capable of locating a less active band in the spectrum where more than 10 active PUs were transmitting. In addition, as shown in this figure, the fast fourier transform (FFT) sensing could not detect two PUs on the right side of the frequency spectrum, while the developed filterbank sensing could easily find them.

5. Conclusions

We present a progressive simulation-based design methodology for designing networked real-time embedded systems. The methodology includes a bifurcated design process that implements progressive simulation-based design for designing both individual embedded devices and the networked embedded system as a whole. We apply this methodology to the development of a cognitive radio network. A single cognitive modem was first developed in a progressive manner, and then the cognitive radio network was built in the same fashion. During this process, the MAC layer model is fine tuned to increase data rate of cognitive radio, while minimizing interference with PUs. Experiment results show that the designed cognitive radio system was able to respond dynamically to the changing environment to avoid active PUs on a real-time basis, while continuing the functionality of a wireless radio. The system was presented successfully at 2007 smart radio challenge held by SDR forum. This case study example shows the effectiveness of the progressive simulation-based design methodology for developing complex networked real-time embedded systems.

Gradually including real system components into a design to replace the simulation models should consider factors such as hardware availability, cost, speed and required accuracy for a certain application. Strategies can be developed in the future to provide guidelines for including hardware in a stepwise fashion in the progressive simulation-based design. Characterizing different types of systems to take the best of the methodology is another interesting topic asking for further research.

References

[1] L. Li, T. Pearce, and G. Wainer, "Interfacing Real-Time DEVS models with a DSP platform," In Proceedings of the Industrial Simulation Symposium, 2003.

[2] J. Upton, Boeing 777 (AirlinerTech Series), 2nd ed. Voyageur Press, August 1998, vol. 2.

[3] X. Hu, From Virtual to Real – A Progressive Simulation-based Design Framework, *Discrete-Event Modeling and Simulation: Theory and Applications*, Chapter 10, Taylor and Francis, 2010 (to appear)

[4] X. Hu, N. Ganapathy, B. P. Zeigler, "Robots in the Loop: Supporting an Incremental Simulation-based Design Process", IEEE International Conference on Systems, Man, And Cybernetics, October, 2005

[5] E. Azarnasab and X. Hu, "An integrated multi-robot test bed to support incremental simulation-based design," In Proceedings of the IEEE International Conference on System of Systems Engineering, 2007.

[6] E. Azarnasab, "Robot-In-The-Loop Simulation To Support Multi- Robot System Development: A Dynamic Team Formation Example," Master's thesis, Georgia State University, Department of Computer Science, Atlanta, GA, 2007.

[7] B. Farhang-Boroujeny, Signal Processing Techniques for Software Radios. Morrisville, North Carolina: Lulu Publishing House, 2008.

[8] E. Azarnasab, X. Hu, P. Amini, B. Farhang-Boroujeny, "Progressive Simulation-based Design: A Case Study Example on Software Defined Radio", Proc. The 2008 IEEE Conference on Automation Science and Engineering (IEEE-CASE 2008), 2008

[9] B. Zeigler, H. Praehofer, and T. Kim, Theory of modeling and simulation, 2nd ed. Academic Press, 2000.

[10] B. P. Zeigler and H. S. Sarjoughian, Introduction to DEVS Modeling and Simulation with JAVA: Developing Component-Based Simulation Models, January 2005.

[11] B. P. Zeigler, Y. Moon, D. Kim, and J. G. Kim, "DEVS-C++ : A High Performance Modeling and Simulation Environment," HICSS (1), vol. 7, pp. 350–359, 1996.

[12] X. Hu, and B. P. Zeigler, "A Simulation-Based Virtual Environment to Study Cooperative Robotic Systems", Integrated Computer-Aided Engineering (ICAE), 12:4, pp. 353 – 367, 2005

[13] R. Brodersen, A. Wolisz, D. Cabric, S. Mishra, and D. Willkomm, “CORVUS: A cognitive radio approach for usage of virtual unlicensed spectrum,” White paper, Berkeley, Available from http://bwrc.eecs.berkeley.edu/ Research/MCMA/CR_White_paper_final1.pdf, Tech. Rep., July 2004.
[14] N. Shankar, C. Cordeiro, and K. Challapali, “Spectrum agile radios: utilization and sensing architectures,” First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN), pp. 160–169, November 2005.
[15] F. C. Commission, “Spectrum Policy Task Force,” ET Docket 02-135, Nov 2002.
[16] S. Haykin, “Cognitive radio: Brain-empowered wireless communications,” IEEE Journal on Selected Areas in Communications, February 2005.
[17] P. Amini, R. Kempter, and B. Farhang-Boroujeny, “A comparison of alternative filterbank multicarrier methods in cognitive radio systems,” Software Defined Radio Technical Conference, November 2006.
[18] P. Amini, E. Azarnasab, S. Akoum, X. Mao, H. I. Rao, and B. Farhang- Boroujeny, “Implementation of a cognitive radio modem,” Software Defined Radio Technical Conference, November 2007.
[19] H. Su and X. Zhang, “Cross-layer based opportunistic MAC protocols for QoS provisionings over cognitive radio wireless networks,” IEEE Journal on Selected Areas in Communications, vol. 26, no. 1, pp. 118–129, Jan 2008.
[20] “Lyrtech SFF SDR development platform technical specs,” Lyrtech Inc., Available from http://www.lyrtech.com/ publications/sff_sdr_dev_platform_en.pdf, Tech. Rep., February 2007.
[21] E. Azarnasab, R-R. Chen, K.H. Teo, Z. Tao and B. Farhang-Boroujeny “Medium Access Control Signaling for Reliable Spectrum Agile Radios”, IEEE Global Telecommunications Conference (GLOBECOM), ISSN: 1930-529X, pp.1-5, November 2009
[22] E. Azarnasab, P. Amini, and B. Farhang-Boroujeny, “Hardware in the Loop: A development strategy for software radio,” Software Defined Radio Technical Conference, November 2007.