Progressive Simulation-based Design: A Case Study Example on Software Defined Radio

Ehsan Azarnasab, Xiaolin Hu, Peiman Amini and Behrouz Farhang-Boroujeny

***Abstract*— Complex systems and manufacturing processes have unique characteristics that make any general modeling unrealistic, however they are inherently hybrid systems that can be attacked with divide and conquer. Progressive design, using simulation for cumulative tests, defines a systematic methodology for development of such systems. In this paper we apply this methodology to implement a cognitive radio network on software defined radio, and show the progress from modeling and simulation point of view.**

I. INTRODUCTION

System design and implementation has been influenced by the increasing demand of new products as well as recent advances in technology. The complexity and multidisciplinary nature of modern 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.

Hardware in the Loop (HIL) simulation 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. In previous work, we have proposed a progressive simulation based design (PSBD) methodology that extends HIL 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]. This methodology has been successfully applied to a robotic convoy system [4], [5].

This paper presents how the PSBD methodology is applied to another application in a significantly different context: 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 [6]. 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 our simulation-based approach, as the individual system pieces are code modules to be developed. The advantage of PSBD becomes more clear 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. Through this case study example, this work aims to show that the progressive simulation-based design is an effective methodology that can be applied to a wide range of engineering systems and automation applications.

The modeling and simulation environment that supports this work is based on DEVS (Discrete Event System Specification) [7]. 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 is not just a theoretical framework, as it has been widely implemented and used as a practical simulation tool in a variety of implementations [8], [9]. The DEVSJava environment [8] is used in this work. More information about the DEVS formalism can be found in [7].

The remainder of the paper is organized as follows. Section II gives an overview of the progressive simulation-based design methodology. Section III describes the Cognitive Radio case study example and highlights some of the design challenges of the system. Cognitive Radio is a recent application of SDR technology. Section IV describes the progressive simulation-based design procedure when applied to the software defined radio system. The design starts with

Xiaolin Hu, Computer Science Department, Georgia State University, 34 Peachtree Street, Suite 1438, Atlanta, GA, USA xhu@acs.gsu.edu

Ehsan Azarnasab and Peiman Amini and Behrouz Farhang, Department of Electrical and Computer Engineering, University of Utah, Salt Lake, UT 84112, USA azarnasa,pamini,farhang@ece.utah.edu

a single radio system, and then proceeds to the design of a cognitive radio network. Some design results are presented. Section V concludes this work.

II. Overview of the progressive simulation-based design

This section gives an overview of the progressive simulation-based design (PSBD) framework that is developed in [3]. 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 Fig. 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, where simulation is carried out using all models. The second stage is virtual environment simulation, where simulation-based study is carried out with combined real system components and simulation models. The final stage is real system test, where all real system components are tested in a physical environment. Along this design process, the framework 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 overlooked before. 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. It is important to note that each design stage is a dynamical evolving process by itself. In particular, the virtual environment simulation stage that involves combined models and real system components typically includes multiple stages by itself. For example, as will be shown in Section IV, the design of the software defined radio starts from the design of a single radio system, and then proceeds to the design of a cognitive radio network. Even the single radio system is designed in progressive manner, where different functional modules are gradually implemented/tested in the real hardware while other modules are provided by simulation models.

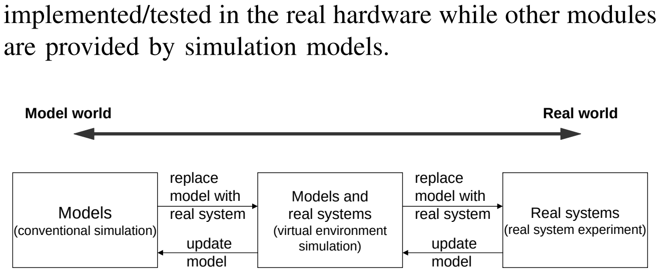
Model world
Real world
Models
(conventional simulation)
replace model with real system
update model
Models and real systems
(virtual environment simulation)
replace model with real system
update model
Real systems
(real system experiment)

Fig. 1. Progressive Simulation-Based design (PSBD) methodology.

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 to designers. It also increases designers confidence about how the final real system is going to work. However, the PSBD goes beyond that by emphasizing a systematical design process which gradually adding more real system components to replace simulation models. It is thus targeted especially for networked engineering systems with tight coupling between software and hardware. 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 for these systems.

III. 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 (PU), however licensed frequency bands are rarely used everywhere and overtime [10], [11] (leading to spectrum holes in time and space). To address this underutilization FCC has recently loosened the regulation to allow secondary user (SU) to share some previously dedicated bands subject to minimal interference to legacy devices of the band [12]. Based on this definition of coexistence, SU tries to dynamically fill the spectrum holes over time and space [11]. For example when a TV station (which is a PU of some frequency band) is not broadcasting or in a location far from any TV broadcasting, SU can instead use the spectrum in an opportunistic fashion. At the same time SU should give up the spectrum when PU begins transmission. SU thus should form a Cognitive Radio (CR) [13] network to sense the presence of active PU and intelligently adapt to a suitable frequency resulting in little or no interference with PU. Spectrum access for first responders in disaster scenario, where many wireless devices are active, is an application of CR [14]. In addition it is desirable to have a radio operating among radios from different vendors and modulations and protocols. Coexistence and interoperability are two major goals expected from a CR. SDR best satisfies the required flexibility of CR, and thus is often used to implement CR [15], [16]. The CR network system thus is composed of many SUs working in the presence of multiple PU each occupying a portion of the spectrum for random amount of time. SU hardware is a CR modem. CR modem is an implementation of CR on SDR board.

The complexity of CR modem is due to the abundance of comprising components and their complicated mutual interaction. These components can be implemented in parallel to expedite the manufacturing process. Parallelism can be achieved by simulating the whole system for a component which is being developed, here the simulation closes the loop of system integration as a regression testing mechanism. In addition, similar to other system engineering problems, not all hardware is available at the beginning of the project. Therefore, parallelism is necessary to finish the project within a time frame. For example 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 legacy users. By the time we developed our fixed point sensing method, we received one RF frontend and applied sensing on real wireless channel. Furthermore, we could be able to test a CR network of many SU when only one complete SDR was available, by simulating the rest of the system before more hardware was available to us. While Simulink, using Realtime Workshop, is capable of handling both real and simulated models (also called co-simulation), we realized that the automatically generated code is not efficient when dealing with realtime requirements of embedded system, and writing custom codes for Realtime Workshop is an extra burden. Therefore we implemented our code and then using optimization level of Code Composer Studio compiled it to an output to be uploaded to Small Form Factor (SFF) SDR [17] hardware platform which is provided by Lyrtech and Texas Instruments. For co-simulation engine we chose DEVSJava platform.

IV. PSBD of Cognitive Radio

A. Simulation model and design procedure

Simulated PU generates a traffic based on a model of the application layer for realistic PUs on particular frequency channels. Depending on the channel this traffic model can be a constant bit rate (CBR), bursty 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 SU not only generates data traffic, but also is responsible of handling signalling packets over control channel of the network.

In the first stage of PSBD, we implement a single transceiver. To start with the traditional modeling and simulation we begin with MATLAB simulation of generic transceiver modem and a generic channel, to find out the initial parameters for required bit error rate (BER) of the system. A complete transceiver is first simulated in MATLAB, then individual and combined signal processing and communications algorithms are substituted by those running on the board. To include the functionalities of Simulink in our system, we also combine MATLAB with DEVSJava, and test some parts of the radio in DEVSJava We use realtime DEVS [7] modeling of DEVSJava [8] to model and then simulate the CR system in all stages of PSBD. We use compiled MATLAB algorithms inside DEVSJava whenever a mathematical model for a subsystem was required. MATLAB Builder for Java, can build Java libraries from the MATLAB functions and have them ready to use in DEVSJava. The MATLAB m-files that were already developed for MATLAB simulation of the channel, some MATLAB visualization methods, and also the hardware-interfacing MEX files are compiled to Java methods that are called later inside our DEVSJava simulation. Note that embedded MATLAB code inside DEVSJava helps in generation of 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 are able to avoid implementation of many mathematical libraries inside Java and instead focus on higher level of modeling with DEVSJava. The integration of DEVS with MATLAB gives designers the opportunity of decoupling the CR model and underlying (often very complicated) math.

While MATLAB is good at modeling a very simple scenario of two nodes, it can not mimic the event-driven nature of interconnected components, on the other hand network modeling inside DEVSJava environment is done naturally. This property of DEVS is further utilized in modeling the network of the CR nodes. After the initial modem is implemented (as one SU), a network simulation with PU and more SU is performed to co-simulate a network of cognitive SUs. At first, a complete simulation of a cognitive radio system with SUs and PUs is developed in DEVSJava. The central simulation engine keeps track of the activities of channel assignments for SUs and PUs. Secondary Base (SB) station 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, CR nodes are modeled with DEVS coupled models of a few atomic DEVS models, while the scheduled messages passing between these DEVS models carry packets with different lengths. Based on the propagation model, the channel model reschedules the messages to the receiver. DEVS modeling of CR system is homomorphic. The hierarchical and network based structure of DEVS models help in modeling the internal interaction of subsystems of a single CR modem as well as interaction of the CR modems in a network. During the course of PSBD, we have replaced two, and are going to replace more, of the simulated SUs with the real cognitive radios in order to test the whole system. In our setup, the real nodes use filterbanks to detect the presence of the PUs and send their sensing information to the SB which is simulated at a PC. In this way, we are able to develop our network, for example, carrying out system-wide test, without waiting for all hardware to be available. As mentioned, the hierarchy of progressive design of the CR network suggests implementing a single modem and its model using PSBD before constructing the network upon this intermediate result. Next, we describe this two phase PSBD process.

B. Single radio implementation

Based on PSBD methodology the abstraction level is chosen according to the required details and accuracy, and then the system is broken into interworking components. First,

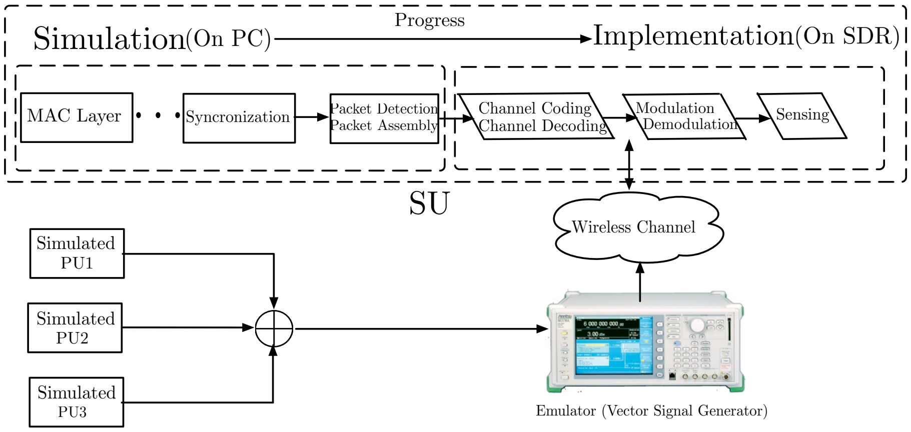

Fig. 2. Progressive Simulation-Based design (PSBD) of a single cognitive modem. The implementation starts from the sensing component and progressively more of the simulated models (left dotted box) are implemented (right dotted box). The rectangles are DEVS models simulated, and parallelograms are implemented components on SDR

we start from a traditional pure simulation of the system in a simulated environment. In this example the environment is defined by frequency usage and channel and is changed as a result of both PU and SU transmissions or different channel fading parameters. The cognitive radio itself is also simulated in Simulink to test for required bit error rate [15]. The PSBD of the modem however is done using DEVS in the course of implementation when real components are added to the model. It is a challenging issue to what extent simulation assisted design should be involved and to what level the system should be decomposed. A single real modem itself comprises many individual modules each a candidate for co-simulation along with other simulated objects and the environment. Figure 2 shows the PSBD approach of designing a single modem including packet generation, coding, decoding, sensing, channel and PU effects.

As we proceed with PSBD, we start using one SDR board. Sensing the medium is the most critical part of cognitive radio and is implemented in the earliest stage as shown in Fig. 2. To test 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 Agilents Waveform Download Assistant via ethernet connection with simulator on a PC. 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.

C. CR network

In the last step we can design our CR network using the developed cognitive modem. Similar to single modem PSBD, we start from traditional simulation of the cognitive network by using the model of a single modem developed in the previous stage and then putting the model in a network

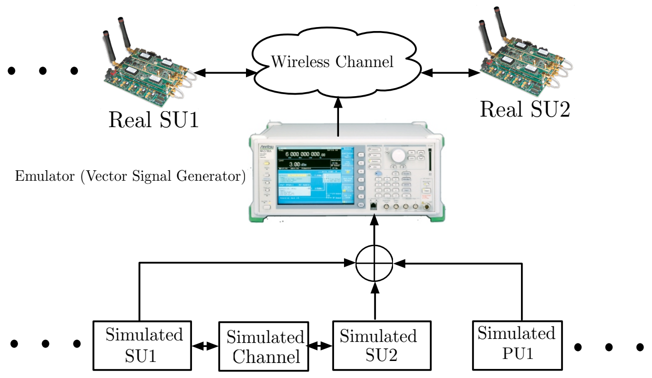

Fig. 3. Progressive Simulation-Based design (PSBD) of a Cognitive Radio (CR) network. The rectangles are DEVS models.

of cognitive nodes. The intermediate stage of PSBD of CR network includes real SU, simulated SU, simulated PU and simulated channel (only between simulated SU). Both simulated SUs and PUs are emulated using a vector signal generator to generate spectral energy on the bands of the simulated agents. Therefore, any real SU would see the channel as if the simulated users exist. To test our CR against generic PU channel usage pattern, the traffic of PU (being simulated) is known to the simulation engine. The emulation of PU for real SU is required to test the sensing mechanism of real SU, however the simulated SU uses the simulated channel and simulated traffic to simulate the PU. The simulated channel is also used between two simulated SU 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 use inside DEVSJava. Figure 3 shows the PSBD of CR network and the interconnection of simulated, emulated, and real components. Note that the emulator and the wireless channel are considered as the environment and do not belong to the system. Similar to the robot-in-the-loop (RIL) simulation [4], [5] this real environment has a shadow model inside the simulator among simulated agents. The emulator conveys the simulated environment to the real real objects of the simulation, much similar to the wireless link in RIL testbed [4].

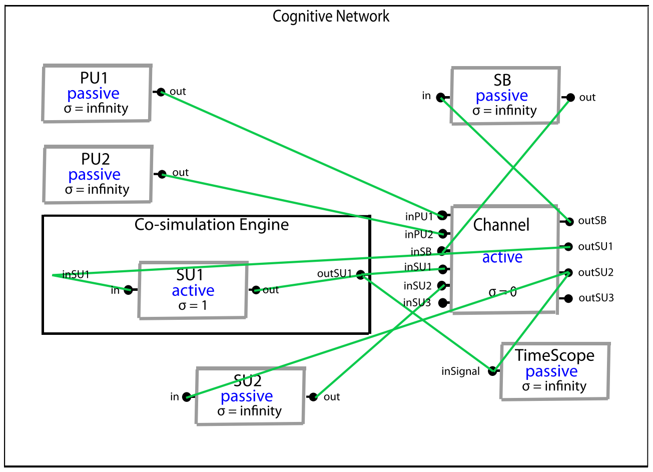

Fig. 4. A cognitive network DEV model with 2 Primary Users (PU) and 2 Secondary Users (SU) and one Secondary Base station (SB), SU1 is implemented inside an SFF SDR board and emulated along with the other nodes

Next step after setting up the relation and hierarchy of the model is to implement higher layers of the network. We consider a centralized network of cognitive nodes thus some SU are assigned to act as SB stations. This approach leads to a global common plane architecture for signaling and control channel that avoids hidden PU node problem, which is an essential property in any CR network to avoid interfering existing PU. SB is in charge of synchronization and channel assignment. SB compiles the channel state information based on sensing results of SU and uses this global knowledge to rank different part of spectrum and blacklist some active channels. Now that the role of each CR node is assigned, we can model the CR network.

PSBD concept in Fig. 3 is realized as successive DEVS models of intermediate design. Figure 4 shows a DEVS model of CR network of two PUs, two SUs and one SB. One of the secondary users (SU2), two primary users (PU1 and PU2) and also the secondary base (SB) station are simulated in the host PC. Only one of the secondary users (SU1) has hardware implementation thus is emulated using the co-simulation engine. 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 were used 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 DEVS model *Channel* 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, down-sampling, etc. At the receiver, also the necessary functions are developed in MATLAB and the data which might have error is passed to the node in DEVS. The real cognitive nodes rely on their embedded software for data transmission among themselves and the co-simulation engine handles their interface to the simulation.

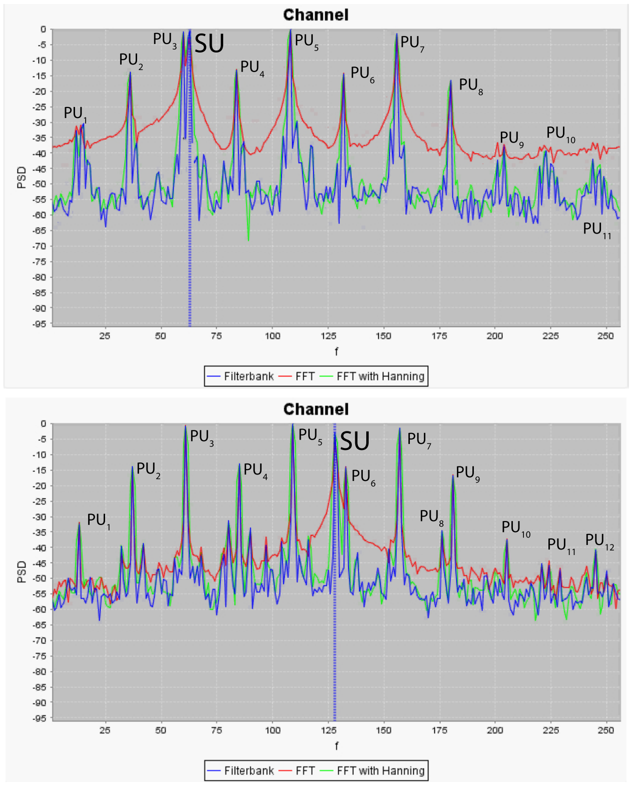

Fig. 5. One SU among many PU: SU changes its band, shown by the vertical dotted line, when PU_3 power is sensed in nearby carrier. Three sensing methods including filterbank, and FFT are shown. The top and bottom figures show before and after PU_3 detection respectively.

Based on PSBD approach we designed and implemented three cognitive SU as follows. After initial simulation of one SU in MATLAB, we agree on certain technical parameters, then inside DEVSJava along with a generic simulation of one PU, we improve our SDR implementation of one SU along with its model. In the next step, we add one more SDR based real SU and more PU nodes along with one simulated SB to our network [15], [18]. The simulated SB is in charge of channel assignment and network management. SB combines the sensing result of the SU and assigns channels to them upon their request. Figure 5 shows the power spectral density (PSD) sensed from a densely populated spectrum for 256 subcarriers in an experiment, this figure shows that SB reassigns a new frequency band to the only SU when PU_3 is detected in the adjacent frequency of the previous carrier (the dotted vertical line). Continuous transmission of voice during this action show the success of the frequency hopping. The cognition is capable of locating a less active band in the spectrum where more than 10 active PUs are transmitting. In addition, as shown in this figure, the fast fourier transform (FFT) sensing can not detect two PUs at the right side of the frequency spectrum, while our developed filterbank sensing easily finds them.

V. CONCLUSIONS AND FUTURE WORKS

Here we presented an example of applying PSBD methodology to build a CR network. A single CR modem was first developed in stepwise manner, and then CR network was built in the same fashion. The cognitive nodes in CR network have shown the required coexistence with legacy devices, while our sensing method prevents collision with low power PUs. The system responds dynamically to the changing environment to avoid active PUs on realtime basis, while continuing the functionality of a wireless radio. The system was presented successfully at 2007 smart radio challenge held by SDR forum. The systematic design of PSBD enabled us to achieve the project goals within one year while hardware and software architecture were prone to constant change. The implemented system is an embedded solution which requires little effort to setup or deploy to disaster filed. Unlike conventional simulation, PSBD surpasses HIL in defining a framework and ongoing strategy all during the implementation process. In future we are going to improve the realtime requirements and implement higher levels of networking protocol stack. The goal is a cross-layer approach for improving QoS among cognitive SU while keeping the interference temperature at minimum. We plan to continue PSBD method in the later stages of the SDR development as well. We are also looking forward to use PSBD in other applications.

VI. ACKNOWLEDGMENTS

The authors gratefully acknowledge the hardware and software support received from SDR Forum, Lyrtech, Texas Instruments, Xilinx, Mathworks, Greenhills, Prismtech, Zeligsoft, and Syncplicity.

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 and E. Azarnasab, "From virtual to real – A progressive simulation-based design framework," *submitted to IEEE Transactions On Systems, Man And Cybernetics Part A: Systems And Humans*, 2007.

[4] 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.

[5] 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.

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

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

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

[9] 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.

[10] 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.

[11] 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.

[12] F. C. Commission, "Spectrum Policy Task Force," ET Docket 02-135, Nov 2002.

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

[14] 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.

[15] P. Amini, E. Azarnasab, S. Akoum, X. Mao, H. I. Rao, and B. Farhang-Boroujeny, "Implementation of a cognitive radio modem," *2007 Software Defined Radio Technical Conference*, November 2007.

[16] 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.

[17] "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.

[18] E. Azarnasab, P. Amini, and B. Farhang-Boroujeny, "Hardware in the Loop: A development strategy for software radio," *2007 Software Defined Radio Technical Conference*, November 2007.