Spectrum Detector for Cognitive Radios

Andrew Tolboe

Motivation

Currently in the United States the entire radio spectrum has already been reserved for various applications by the FCC. Therefore, if someone needs to use part of that spectrum for any reason (development of new hardware/idea, some sort of short range information transfer, etc) they can't do it at all without violating FCC regulations. However, there is a new FCC supported technology that can be used to overcome such a problem called cognitive radio.

In the context of this proposal, all a cognitive radio does is it scans the available radio spectrum looking for a frequency band that is not currently being used. Once it finds a band that is open, it will use that frequency until the owner of the band begins using it. Then the cognitive radio is forced to rescan the spectrum and use another open frequency band.

Introduction

This project will implement only the part of the cognitive radio that scans the available frequencies. It will use a filter that has the capability to filter out everything except a specified section of the spectrum. This process will then be applied to all of the frequency bands that we are interested in. The data will then be sent to a computer so a list of available frequencies can be displayed on a screen in some sort of graphical manner. To implement this part of the cognitive radio a software radio will be used.

A software radio is a radio in which all of the capabilities of the radio are defined in software, whereas with a traditional radio everything was defined in hardware and cannot be changed once manufactured. The ideal software radio contains an antenna that would pick up all of the frequencies that are needed. Second there is an analog to digital converter (ADC), to convert signals to the digital domain from the analog domain, and a digital to analog converter (DAC), to convert signals to the analog domain from the digital domain. Finally there is extremely flexible hardware that can be programmed to do the digital signal processing on these signals. Most software radios have a combination of hardware for this, such as a digital signal processor (DSP) and a field programmable gate array (FPGA). The DSP is a programmable processor specially designed for doing digital signal processing and the FPGA is a device that can be programmed to implement digital circuits.

To demonstrate the functionality of the frequency scanner another radio device with the capability to transmit a radio signal, possibly another software radio, will be placed next to the frequency scanner. This other device should have the ability to change the frequency it is broadcasting on, such as another software radio or something simple like a citizens' band radio. A citizens' band radio, or CB radio, is a short distance radio with 40 channels in the 27 MHz band. So, as the other radio device broadcasts on different frequencies the frequency scanner can show that those frequencies are being used.

Related Work

The Wireless Communications Lab at the University of Utah has developed cognitive radios in the past and is currently working on improving many different aspects of a cognitive radio. However, currently nobody in the lab is using a technology called software communication architecture (SCA) to develop cognitive radios. SCA is an operating environment that has been standardized by the Joint Tactical Radio System (JTRS) for the U.S. Military. The power of using SCA is that a single SCA application can easily be adapted to work with any SCA implementation as long as it is compliant with SCA specifications. For example, a company writes an application for their SCA certified implementation, then that application will work on all of the software radios that have that SCA implementation. If one of those applications needs to be moved to another SCA certified implementation it can be done without difficulty.

This project will take a little bit from both SCA and cognitive radio technology. The project will develop enough of a SCA so a spectrum sensor can be developed on top of the SCA. For this project the SCA will be written as close to SCA version 2.2 specification as possible so it can possibly be used in future projects.

Other groups outside of the Wireless Communications Lab have developed cognitive radios using SCA. However, this work is being done specifically for the Wireless Communications Lab at the University of Utah as one of the first projects to use SCA which makes the project important.

Effort Details

The Software Radio

The Software Radio that will be used is the TMDSSFFSDRR from Texas Instruments which will be provided by the Wireless Communications Lab at the University of Utah. This software radio has three physical modules. The first module is the RF module which basically handles the RF signals, both transmit and receive. Then the RF module communicates with the Data conversion module. This is the module where the DAC and ADC convert the signal from the digital domain to analog domain and from the analog domain to the digital domain respectively. This module then communicates with the last module via two first in first out (FIFO) queues. A FIFO queue is just like a checkout line at the grocery store, the first person in line is the first person to get out of line or first person helped. There is one queue for outgoing signals and one queue for incoming signals. The final module is the digital processing module which is where all of the digital processing modules are. On the digital processing module are two main chips for control, one chip is a DSP chip and the other one is a Vertex4 FPGA. This allows for both digital programmable logic as well as programmable logic which can be a powerful combination. The FIFO queues from the last section feed into the FPGA and there is another FIFO bus between the FPGA and the DSP for communicating between the two units (1).

The communication protocols for this software radio are already built in for loading the programs onto the chips and communicating with the software radio while it is in use. The software radios have an ethernet port and all of the needed software for computer networking built right onto the board. They have a small FTP server that can be accessed (with a password) to load the FPGA bit stream and the DSP program. The ethernet port also has a communication protocol that can be used in programming languages to read and write directly to the memory of the DSP chip (2).

The Display Device

A computer will need to be used that is capable of running a C/C++ program and a graphical user interface, and has an ethernet port for communication with the software radio. A laptop has been obtained with the Microsoft Windows operating system on it which should be sufficient for the task.

A program will be written that is capable of communicating with the software radio and displaying the information on a screen in a graphical manner. A protocol has been developed for handling the communication between the display interface program and the software running on the software radio, this is described in the next section.

Interfaces

There will only be a single interface that connects the software radio to the computer for displaying information about the current state of the spectrum. Because a computer can directly access the memory on the DSP chip a simple protocol can be implemented for communication. The first bit of the memory location that will store information for the computer application will be a valid bit. When the DSP chip is writing spectrum data to that memory location it will make the valid bit zero. When it is done writing data to that memory location then it will change the valid bit to one. Then all the computer application needs to do is check to see if the valid bit is a one. If it is one then the data is valid, if it is not, then the data is invalid and should be thrown away.

For the actual spectrum data, there will be a single bit to symbolize each spectrum interval that will be represented. If there is a one for that bit then that spectrum interval will be open, if there is a zero then it will not be open.

Software Communication Architecture

The project's operating environment will be the software communication architecture (SCA). SCA is basically an operating system that runs on the software radio as an abstraction layer. So instead of writing software for the software radio you write software for SCA. Then any software radios that have a SCA implementation that you want to run your software on you can with little to no hassle. Some SCA implementations may be a little different, but if the SCA model was followed, they should be similar. The idea is much like the Microsoft Windows operating system on a computer. A company must only write one program that runs on Windows. Then the burden of dealing with all the different hardware differences lies on Windows (3).

Because writing a SCA can be such a large job a tool is going to be used called Spectra SDR Power Tools. This tool has features that will ease the process of designing a SCA implementation for the software radio. The designer starts by defining the business logic (how everything will fit together) for the SCA implementation. Then once a given module is complete the tool will generate code and tests for the module. The module should then be tested and once fully tested, it is ready for implementation on the software radio. This testing needs to be done for all required modules (4).

SCA Application

The application that will run on top of the SCA framework will be a simple application. All it will need to do is tie together multiple SCA interfaces. For example, it will read in spectrum data from a SCA interface, correctly format the data, and send it to another SCA interface to be sent to the computer application.

Testing and Verification

To test the software written for this project (both the SCA application and graphical display program) a set of program stubs will be written to test the application's reaction to expected cases, unexpected cases, and extreme cases. Assertions will also be used to verify conditions that should never happen.

The Spectra SDR Power Tools suite provides a test harness that should be sufficient for testing the SCA code produced. However, as the code for each module is produced and tested by the tool, assertions and function stubs will also be used to verify compliance and expected reactions.

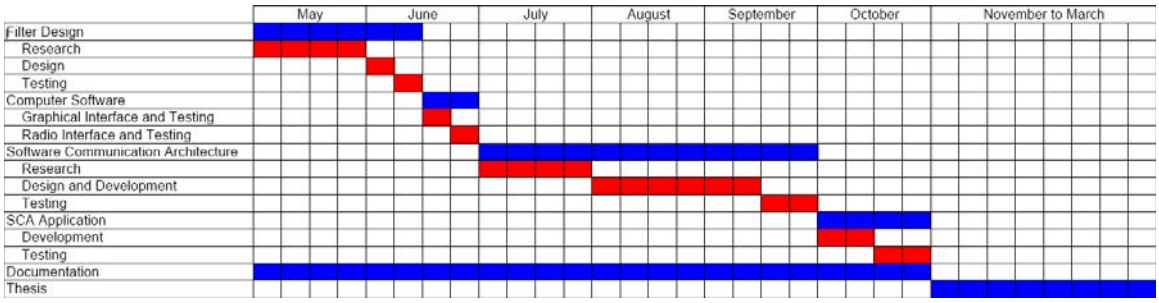
Risks

Software communication architecture may be hard to understand and properly implement. This will first require a lot of reading and research on the subject and nobody currently in the Wireless Communications Lab has seen a SCA implementation all the way through. The second fall back to this will be to abandon the software communication architecture and write code from scratch for everything, which is how the software radios are currently being programmed in the Wireless Communications Lab.

Currently the Wireless Communications Lab does not have any active licenses for its copy of the Spectra SDR Power Tools suite. However, last year the lab was able to get a license and they are confident that a license can be obtained again this year. If one cannot be obtained for the Spectra SDR Power Tools then there is another SCA suite that the Wireless Communications Lab has that could be used. However, again a license will need to be requested for this software as well. If no software can be obtained then it would be possible to write everything from scratch or just not use SCA.

Another possible risk associated with a license is the license may expire before the work has been completed. However, if this does take place another license could be requested.

Schedule and Milestones



Filter Design

The project will require the use of a filter to detect which frequency bands are available. The first step is going to be to research exactly how to properly design a filter or filter bank that can be implemented for this project. Then Matlab will be used to actually design and test the filter, and finally Verilog will be used to implement the filter in hardware on the FPGA. The result of this milestone is a filter that can be used on the software radio. The resolution of the scanner and range will be known so that information will be available when writing the computer software and finalizing how the interface will work (how many bits are needed to transfer all the required information).

Computer Software

The software that runs on the computer will be written in this milestone. The first step will be to find an API that can be used for interfacing with the Windows operating system for displaying the information that needs to be displayed. Then this graphical interface needs to be tested by changing values that currently reside in computer memory, representing different frequency bands, and making sure the display updates properly.

Second the interface to the actual software radio will be written using the C library included with the software radio on top of the graphical interface code. Then a set of function stubs will be written to test that this program can handle both expected values and unexpected values from the DSP's memory.

The result of this milestone will be software that can run on a computer networked with the software radio that has the capability of displaying spectrum information.

Software Communication Architecture

The first component of this milestone is research. A large amount of time in the schedule has been reserved for researching this topic and it is the most important research topic in this project because it is what makes this project unique. Because SCA is what makes this project important, it is imperative that the implementation is done correctly. Towards the end of the research a license for Spectra SDR Power Tools needs to be requested. The result of this section of this milestone is a greater understanding of the SCA and its components.

Next the business module of the SCA implementation using Spectra SDR Power Tools needs to be completed. This includes instructing the tools how to interface with the hardware of the software radio and the display interface that will reside on the computer and the filter implemented on the FPGA chip. If any modules need to be written by hand then that will also be addressed for this milestone. As the different modules are merged together testing will be done as progress is made to assure functionality. The result of this section should be a SCA that is ready for the software radio.

Finally, testing will be done on this SCA on the software radio to assure that communication correctly works between all of the different modules and components on the radio itself. The result of this will be a fully functional SCA that has everything needed for this project.

SCA Application

The application that runs on top of the SCA is the last milestone for this project. This is the software that will run on top of the SCA software and because everything has already been tested to work, the software must only communicate with different modules of the SCA and put everything together, which should not be difficult. Once the software is written it will be tested using function stubs before being put on the software radio. Once the previous tests have passed, the application will be transferred to the software radio for final testing using a radio device that is able to broadcast on different frequency bands. The result of this milestone will be the complete project.

Documentation

Documentation will be written as each step is taken so that the information will be available in the end without having to try to recall what was done. The result of this milestone will be complete documentation of the entire project.

The Thesis

Finally the thesis will be written. The result of the completion of this milestone will be the final project.

References

1. Small Form Factor SDR Evaluation Module/Development Platform user's guide
2. Lyrtech API guide
3. Brad, John & Kovarik, Vincent. Software Defined Radio: The Software Communications Architecture. John Wiley & Sons Ltd, England. 2007
4. Spectra SDR Power Tools documentation