Software Defined Radio in the Electrical and Computer Engineering Curriculum

Ladimer S. Nagurney
Department of Electrical and Computer Engineering
University of Hartford
West Hartford, CT 06117 nagurney@hartford.edu
and
NSF Engineering Research Center for the Collaborative Sensing of the Atmosphere, CASA
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003

Abstract - **The development of Software Defined Radio systems and their extension to Cognitive Radio Systems and Smart Radio Systems have introduced a plethora of topics and examples that can be included in the curriculum. The design of these software defined radio systems has less in common with traditional radio design and more in common with the design of Embedded Systems and Software Engineering. This purpose of this paper is to overview software defined radio from the simplest form to its most complicated form and giveexamples on how software defined radio concepts can be used as examples and exercises in a variety of Electrical Engineering and Computer Engineering courses and labs.**

Index Terms – Software Defined Radio, Electrical Engineering Laboratories, Computer Engineering Laboratories, Digital Signal Processing Applications

INTRODUCTION

For over ¾ of a century radio transmitters and receivers have been designed using analog components to implement frequency generation, modulation, amplification, demodulation, and any additional circuitry regardless of whether the information to be transmitted was analog or digital. Since the development of Small Scale Integrated Circuits, Large Scale Integrated Circuits, and the Microprocessor, various digital circuits have been added to the radio transmitters and receivers. The initial use of digital circuitry was primarily for frequency generation and control and for coding and decoding of digital information transmitted using analog modulation.

The complete implementation of a radio system using digital processing of sampled analog signals was proposed about 2 decades ago. However, at that time, it was not possible to develop such digital radio systems that could operate in real time, except for a few limited applications that were unconstrained by cost and size. As technology has progressed the development of DSP processors and the use of FPGA's for signal processing allows the implementation of a radio system.

Many of these Software Defined Radio developments have been fueled by the need for reconfigurable radio receivers and transmitters that can be updated remotely. For example, the radio system at a cell site might need to be reconfigured to implement new standards or protocols, bypass failed hardware, or reconfigure the network due to a change in the traffic pattern. By using Software Defined Radio, these changes can be performed remotely, thereby reducing the need for visits to the site and increasing cost efficiency. Another application of Software Defined Radio is in the development of multi-standard communications systems needed for interoperability among users where a new modulation method, frequency range, coding scheme, etc could be downloaded. A future example is projected for Automotive Radios where additional features such as satellite reception (XM or Sirius) could be remotely added to the receiver.

Non-communications examples of Software Defined Radio include radar systems where new pulse schemes can be easily implemented and the received RF echoes may be digitally down converted to baseband and digitally correlated with the transmitted pulses. Another use is in RFID tag readers where Software Defined Radio techniques may be used to configure RFID readers for multiple RFID protocols [1].

The development of these so-called Software Defined Radio systems have shifted radio development topics from Communication Engineering and RF Design courses to courses in FPGA implementation and software engineering.

The purpose of this paper is to describe many of the topics related to Software Defined Radio and illustrate where these topics might be included in the Electrical and Computer Engineering Curriculum. In subsequent sections, I describe the fundamentals of Software Defined Radio, outline both complex and simple implementations, and show how these implementations can be used as examples, exercises, and projects in various courses.

SOFTWARE DEFINED RADIO FUNDAMENTALS

Software Defined Radio, SDR, is defined as the hardware/software implementation of radio systems. While it was suggested that SDR system could be built almost 2 decades ago, it was not until the development of fast DSP processors and easily configurable FPGA hardware that SDR became practical. The basic architecture of an SDR Receiver or Transmitter is illustrated in Figure 1. Although the block entitled processor, might imply the use of a computer, the actual implementations, to be described later, include Standard CPU, DSP processor, FPGA, or even hardwired discrete logic.

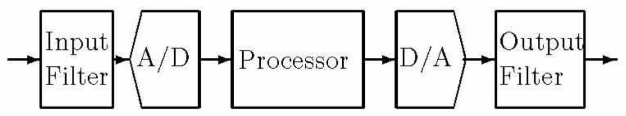


FIGURE 1
BASIC ARCHITECTURE OF A SOFTWARE DEFINED RADIO

In Figure 1 to implement a receiver the Input Filter should be a bandpass filter centered near the received frequency and the sample rate of the input Analog-to-Digital converter will be at least twice the received frequency. The output Digital-to-Analog converter should be sampled at a frequency that is at least twice the bandwidth of the baseband signal. The Output Filter should be a low pass filter to band limit the baseband signal. To implement at transmitter, the specifications of the filters and sampling rates should be reversed. Then, the Input Filter will band limit the baseband signal and the Output Filter will be a bandpass filter centered at the transmitted frequency.

In many applications, the baseband signal, itself, is digital. Then the baseband A/D and D/As and their associated filters can be eliminated and the digital signals fed into the processor. In effect, the processor would also become a modem.

Note that this SDR implementation is independent of the final modulation type, thus any modulation mode for the transmitted signal, analog (AM or FM0 or digital (BPSK or QPSK) may be implemented with the same hardware architecture. This is a key advantage of using Software Defined Radio.

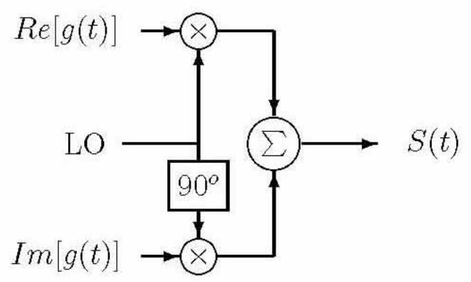
While the discussion of this overarching architecture could continue, it is clear that the processor must be fast enough to process the information in real-time. This fundamental requirement has lead to various choices for implementing the processor. It is these implementations that illustrate how SDR may be included in the curricula.

It is often clear that the conversion speed required for the receiver A/D and the transmitter D/A might not be attainable at reasonable cost. Thus, alternative implementations have been developed to allow use of slower converters. In particular, quadrature modulation and demodulation have become standard in implementing SDR. The quadrature modulator, illustrated in Figure 2, can be used to generate the signal s(t) as follows. Following [2]

$$S(t) = \mathrm{Re}[g[m(t)]e^{j\omega_c t}] \qquad (1)$$

$$= \mathrm{Re}[g[m(t)]] \cos \omega_c t + \mathrm{Im}[g[m(t)]] \sin \omega_c t \qquad (2)$$

The requirements for the two baseband D/A converters to generate the in phase and quadrature components of g[m(t)] only require that the converters operate at the Nyquist Frequency for the baseband signal. While the



system performance is dependent on the amplitude and phase balance of the quadrature modulator, the baseband processor can be adapted to correct these errors as in [3].

FIGURE 2
QUADRATURE MODULATOR

On the receiver side, the sampling rate can be reduced by using a quadrature demodulator. By mixing the input signal with both a Local Oscillator Signal and its 90 degree phase shift, baseband I and Q signals may be generated. While only the I signal is required to demodulate AM, both the I and Q signals are required to demodulate other forms of modulation. Since the baseband signal for audio is often in the 0-20KHz range, relatively slow and inexpensive A/D converters may be used in this application. The actual demodulation will be performed in the processor.

As an alternative to the quadrature demodulator that is both simple and less sensitive to component variations, a Tayloe Detector was developed and is described in [4]. As shown in Figure 3, this Quadrature Sampling Detector requires a local oscillator operating at 4 times the carrier frequency. This LO signal is decoded to drive 4 RF switches. The outputs of the switches are fed into operational amplifiers that are configured as differential sample and hold circuits to generate the baseband I and Q signals. The simplicity of the Tayloe Detector is evident because it does not require any analog quadrature circuitry and thus may be easily broad banded. In addition, its upper carrier frequency limit is constrained by the speed of the digital circuitry. Another advantage is that by changing the LO signal frequency the received frequency can be easily changed and thus a tunable receiver implemented.

It is important to note that the I and Q signals themselves might have much broader bandwidths than the baseband signal allowing the processing and detection to occur in the processor.

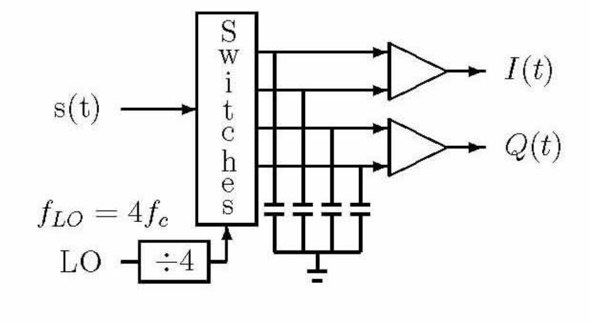


FIGURE 3
TAYLOE DETECTOR

While this paper does not have space for more SDR hardware design, more information on SDR and its implementation can be found in [5].

An extension of Software Defined Radio is Cognitive Radio. [6] A cognitive radio receiver receives a signal and from its spectral/temporal characteristics determines the type of modulation, data rate, data format, etc. Cognitive radio does not need to know all the parameters of the communications network. An example of a cognitive radio might be a television receiver that automatically determines whether the received signal is NTSC, PAL, SECAM, or ATSC and automatically processes the received signal.

LINKING SDR TO DSP

To implement a real-time communications system, the processor must operate in real time, thus the performance from a DSP processor is often required to implement an SDR. The implementation of an SDR receiver/transmitter on a DSP processor evaluation board is an excellent example of to illustrate DSP processor implementation..

Figure 4 illustrates a typical DSP processor evaluation board, such as those from Texas Instruments or Analog Devices. It is straightforward to implement a transmitter using the input A/D Converter to sample the baseband (audio) signal and the stereo D/A converters to output the I and Q waveforms to the quadrature modulator. For simplicity, in Figure 4 the analog band limiting filters have not been included.

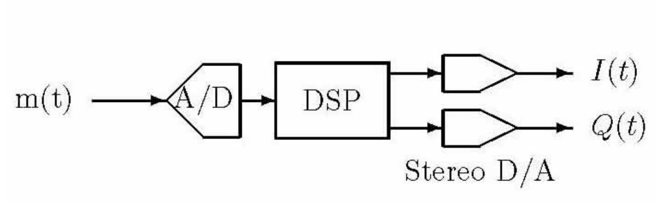


FIGURE 4
SAMPLE DSP PROCESSOR BOARD ARCHITECTURE

As a concrete example of a typical student project, consider the implementation of a single-sideband suppressed carrier transmitter. Following [2] the I and Q signals that must be generated are:

$$I(t) = A_c\ m(t) \quad (3)$$

$$Q(t) = A_c\ m^*(t) \quad (4)$$

Where m*(t) is the Hilbert Transform of m(t).

While the Hilbert Transformer of a signal is difficult to implement using analog components, it is straightforward to implement an FIR filter to create the Hilbert Transformer m*(t). Once the coefficients for this FIR filter are calculated, the DSP processor can be programmed to implement the filter. Most floating point DSP processors operate at a high enough rate to implement this system. If the ultimate D/A speed is high enough, the Hilbert Transformer can be combined with an interpolation filter to increase the sampling rate to one where only a single D/A is needed to output s(t).

This simple example may be implemented using either DSP machine language or C. However, the programs for DSP boards may be created using a Block Diagram Language such as Simulink or Labview. The block diagram is converted to C-code which is then compiled, linked, and loaded to the DSP processor. Thus, no actual C programming is necessary. An advantage of this method is that the block diagrams can be simulated to ensure proper system performance before being loaded to the DSP Board.

Because the communications system must operate in real-time, it is important that in implementing equations (3) and (4), care must be taken so that the data from each arrives simultaneously at the D/A. Thus, a delay element with the same time delay as the Hilbert Transform FIR filter must be inserted in the branch of the program representing equation (3) as illustrated in Figure 5. Note that the overall gain, Ac, may be included in the Hilbert Transformer/Delay Specification. This is often a student's first opportunity to realize that timing of the data does matter.

One constraint on most DSP evaluation boards is that they have no more than 2 channels of A/D and D/A converters, thus not allowing simultaneous transmission and reception. This problem may be overcome by having the student design and construct a daughter board containing an A/D and D/A. or by being very clever and designing a half-duplex switching circuit to switch the DSP Processor between transmit and receive.

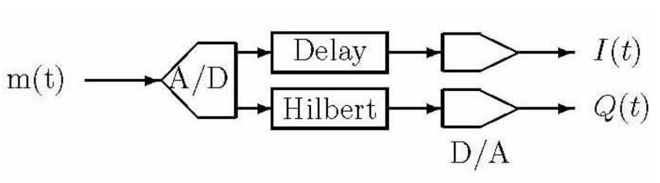


FIGURE 5
SSB-SC IMPLEMENTATION

The development of an SDR system in this way is no more complicated that any assignment in an Undergraduate

DSP course and can serve as a course project or capstone design project.

FPGA IMPLEMENTATION OF SDR

With the development of Intellectual Property (IP) based FPGA design, SDR design using FPGAs has become more popular. The SDR algorithms are implemented as blocks within an FPGA. This gives the student familiarity with how multiple data streams may be processed simultaneously in an FPGA. Indeed, most high end transceiver boards have an architecture similar to Figure 6. An overview of high end FPGA based transceiver boards can be found in [7] and [8].

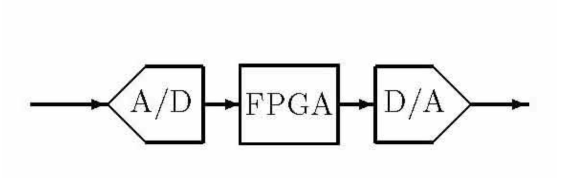


FIGURE 6
BLOCK DIAGRAM OF A TYPICAL FPGA BASED SDR BOARD

Since most Electrical and Computer Engineering students have been introduced to FPGA design in digital logic courses, it is possible to ask the students to apply these techniques to SDR. Implementing an SDR using an FPGA is a good example of a discrete time system derived from an analog signal may be implemented using an FPGA and illustrates that FPGA's may be used for more than just processors or inherently digital systems.

To illustrate this use of FPGAs, consider an AM transmitter implementation. The transmitted signal s(t) will be:

$$s(t) = A_c\ (1+m(t)) \cos w_c t \qquad (5)$$

or following the analysis in (1) and (2)

$$g[m(t)] = Ac\ (1+m(t)). \qquad (6)$$

It is straightforward to implement on an FPGA board such as a Digilent BASYS using a Xilinx Spartan 3E FPGA[9]. While some FPGA boards include the A/D and D/A converters, others must have them added as daughter boards. The block diagram of the implementation is illustrated in

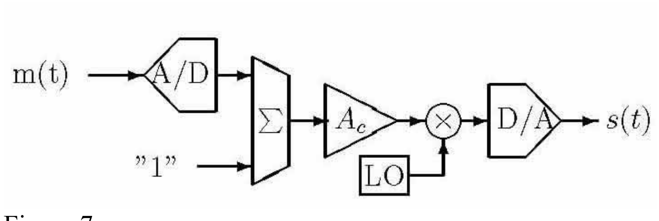


Figure 7.

FIGURE 7
BLOCK DIAGRAM OF AN IMPLEMENTATION OF AN AM TRANSMITTER

While the blocks are an summer and a multiplier are needed, judicious choice of values, the gain block might be eliminated and the adder block may be simplified. When faced with theses choices the students will often begin to understand why the concepts taught in logic and computer architecture are related to the system world. Additionally, the students are exposed to how number representation affects system design and implementation.

Because the ultimate speed of the FPGA is faster than typical sequential processors, one may use a output Digital-to-Analog converter that can sample at a rate twice the carrier frequency. Therefore, one can implement a transmitter that operates at true RF frequencies using this technology.

While the simplest FPGA implementations usually use a USB interface to copy the program from the host, many FPGAs have sufficient room to implement a small processor so that code updates may be uploaded via a network connection.

SIMPLE SDR RECEIVERS

The radio amateur and hobbiest communities have developed a series of hardware designs for SDR receivers that are straightforward to construct and can be easily used to add SDR into the curriculum. The design, as illustrated in Figure 8 is often referred to as the SOFTROCK design [10], [11] uses a crystal oscillator, divide-by-4 counter and a Tayloe Detector. A bandpass filter conditions the input signal.

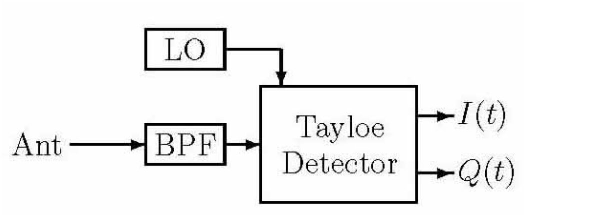


FIGURE 8
BLOCK DIAGRAM OF THE SOFTROCK RECEIVER

The outputs of the Tayloe Detector, which are I and Q signals, can then be fed into the sound card input of a PC. By using a soundcard with a sampling rate of 48 or 96 kHz, the I and Q signals can be tuned and with a fixed LO, a range of signal frequencies can be tuned and detected. There are several public domain programs, including ROCKY [12], Flex-Radio PowerSDR [13] WINRAD [14], and SDRADIO [14], which have been developed to convert the I and Q signals to baseband. While each of these packages is complete in itself, each provides slightly different functionality.

A weakness of the Softrock design for the classroom is that most of the IC's are surface mount limiting how the students may investigate the circuit. Recently, Coyle [15] has published a simple SDR design using only DIP components that can be easily fabricated on protoboards. This allows the straightforward construction, as well as, ne format.

testing of a SDR receiver. Cole [15] also explains the used of a low-cost commercially available Digital Digital Synthesis frequency synthesizer module to expand the frequency range of the receiver.

A search of the web will find many other projects that can easily be adapted to course or capstone projects.

SOFTWARE DEVELOPMENT

The SDR software, described in previous sections of this paper, is straightforward to implement and could be used as a project in several courses. The basic algorithms necessary to generate the transmitted I(t) and Q(t) from m(t) and to recover m(t) from the received I(t) and Q(t) can be found in [16]. The software development project can be divided into several pieces, since it is more than just the sequential processing of I and Q.

A big piece of the development is understanding how to interface with a sound card and the variabilities of sound card software design as described in [17]. The key issue is to insure that the sound card is programmed so that the sampling rate is programmed to be 48 kHz or, preferably, 96 kHz, in the stereo mode. While the examples in [17] are coded in Visual Basic under Windows, C may also be used.

The development of the detection algorithms are described in sequential form in [16] with a more detailed description of the background in [18]. However, by using FFT's there are more efficient algorithms that can be implement not only to decode I(t) and Q(t) , but use the spectrum of I(t) and Q(t) to tune around the carrier frequency using software. While these techniques are often mentioned in Digital Signal Processing Courses, implementing an SDR gives examples of their use. An overview of these techniques is given in [19].

OTHER PROJECTS

While this paper presents a brief overview of the types of assignments and projects related to software defined radio that can be performed in the Undergraduate Electrical and Computer Engineering Curriculum, it is important to note that SDR technology is not used in just communications but radar, magnetic resonance and any other system that requires a pulse to be transmitted and it reflection received [19].

Here are some ideas (not exhaustive) for additional projects involving SDR and its techniques.

- Simulation Using Labview – Labview may be used to simulate an SDR receiver or transmitter. For non-realtime prototyping of SDR, Labview might be software of choice.
- Signal generator development using SDR techniques – It would be straightforward to develop a signal generator or an arbitrary waveform generator using SDR techniques. In fact, using SDR one can implement a high quality signal generator.
- Generation of sample I(t) and Q(t) signals – using a variety of techniques, sample I(t) and Q(t) signals can be generated and stored as a file. Most SDR software allows the recording and playing back of the I(t) and Q(t) signals. This is also an efficient way to verify performance of the software. This would introduce students to the notion of using sample data to test a program.
- Hardware simulation using SPICE or other appropriate software – the simulation of a the hardware part of the system is an important part of designing the system. This way trade offs and effects of the Op Amp parameters in a Tayloe Detector may be investigated.

SUMMARY

This paper has outline a variety of exercises and projects that may be used to include Software Defined Radio in the Electrical and Computer Engineering Curriculum. Each of these can be performed as a course assignment, course project, or capstone project. The use of SDR will include the latest technology in the courses. This work is intended to outline various possibilities and provide a framework for further investigation.

ACKNOWLEDGMENT

The portion of this work performed while the author was on sabbatical at the Engineering Research Center for the Collaborative Adaptive Sensing of the Atmosphere was supported in part by the Engineering Research Centers Program of the National Science Foundation under NSF Cooperative Agreement No. EEC-0313747. Any Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

REFERENCES

[1] Wasserman, M., “SDR-Based Readers Keep Pace With Changing RFID Technology,” RTC Magazine, January 2007, pp 42-45.

[2] Couch, L.E, Digital and Analog Communications Systems, 4th edition, Macmillian, NY, 1993, p 252 (Also in later editions)

[3] Simoneau, J.S., and L.W. Pearson, “Digital Augmentation of RF Component Performance in Software Defined Radio,” IEEE Transactions on Microwave Theory and Techniques, Vol. 57, No 3., March 2009, pp 573-581.

[4] Youngblood, G, “A Software Defined Radio for the Masses, Part 1”, *QEX*, July/August 2002, pp 1-9

[5] Frerking, M. E., *Digital Signal Processing in Communications Systems*, Van Nostrand, Reinhold, NY, NY, 1994

[6] Mitola, III, J., “Cognitive Radio Architecture,” in C*ognitive Radio, Software Defined Radio and Adaptive Wireless Systems*, Hüseyin Arslan, editor, Springer, Dordrecht, Netherlands, 2007, pp 43-105

[7] Hosking, R. H., *Putting FPGAs to Work in Software Radio Systems,* 3rd edition, Pentek, Upper Saddle River, NJ, 2006, available from http://www.pentek.com

[8] Hosking, R. H., *Digital Receiver Handbook: Basics of Software Radio*, 6th edition, Pentek, Upper Saddle River, NJ, 2006, available from http://www.pentek.com

[9] http://www.digilent.com

[10] http://www.softrock.org

[11] http://www.amqrp.org/kits/softrock40/

[12] http://www.dxatlas.com/Rocky/

[13] http://ewjt.com/kd5tfd/sdr1k-notebook/sr40/sw.html

[14] Both WINRAD and SDRADIO can be downloaded from http://www.weaksignals.com

[15] Coyle, L, "A Modular Receiver for Exploring the LF/VLF Bands – Part 2", *QST,* Vol 92, No12., December 2008, pp 33-37.

[16] Johnson Jr., C. R. and W.A. Sehares, "Telecommunication Breakdown" *Concepts of Communication Transmitted via Software-Defined Radio*," Pearson-Prentice Hall, Upper Saddle River, NJ,.

[17] Youngblood, G., "A Software Defined Radio for the Masses: Part 2," *QEX*, September/October 2002, pp 10-18.

[18] Agilent, Digital Modulation in Communications Systems — An Introduction Application Note, 2001, http://cp.literature.agilent.com/litweb/pdf/5965-7160E.pdf

[19] Youngblood, G., "A Software Defined Radio for the Masses: Part 3," QEX, November/December 2002, pp 1-10.

[20] D. McLaughlin,D., et al " Short-Wavelength Technology and the Potential for Distributed Networks of Small Radar Systems," Submitted to the *Bulletin of the American Meteorological Society (BAMS) 2008.*