

A SINGLE SEMESTER SOFTWARE DEFINED RADIO TRANSCEIVER IMPLEMENTATION IN A XILINX SPARTAN-3 FPGA

Steven M. Brown (Penn State University, University Park, PA, USA; smbrown@cse.psu.edu); Justin R. Ford (Penn State University, University Park, PA, USA; jrford@psu.edu); Kyusun Choi (Penn State University, University Park, PA, USA; kyusun@cse.psu.edu)

ABSTRACT

This paper describes a Tier 1 software-defined radio system that is space efficient and cost effective, integrating as many components as possible onto a single FPGA chip. We have implemented, demonstrated, and described a SDR based on straightforward concepts. This system is well suited for lecture discussions, demonstration of an AM digital radio, and practical laboratory implementation. We have targeted our design to be a real-world system design experience that allows students to put RF engineering, DSP, and communications theory into practice within the span of a single semester.

1. INTRODUCTION

In the next several years hardly anyone will escape being involved, in some way, with wireless communications. As wireless data communication increases in popularity, the demand for new college graduates competent in their understanding and design also increases. A Software-defined radio (SDR) course offers an excellent opportunity for students to understand and to apply wireless communications theory in a practical way. SDR systems support the modulation and demodulation of a wide variety of wireless communication standards. These communication schemes must be decoded in architecturally efficient manners, placing great demands on the designer's knowledge of digital signal processing and system design. A single SDR may have to decode cellular, radio, and television broadcasts, placing high demands on the design created. The challenges are not limited to simple software design, but to system integration and more traditional radio frequency design. These are the challenges placed before the next generation of engineers.

This paper describes a Tier 1 SDR system that is space efficient and cost effective, integrating as many components as possible onto a single FPGA chip. This system is well suited for lecture discussions, demonstration of an AM digital radio, and practical lab implementation. Adding a general-purpose processor to the digital radio to implement

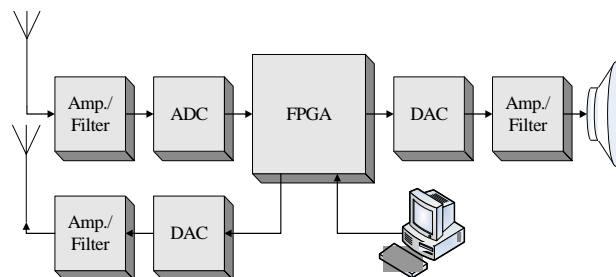


Figure 1. Proposed prototyping system

modulation and demodulation would promote this to a full SDR.

By the end of the semester students will implement a complete transceiver system, giving them practical experience in RF engineering, DSP, and communications. The construction of an RF receiver and transmitter provides an opportunity to discuss the costs and benefits of various designs. By implementing various types of filters (such as the infinite impulse response and peak detector to be presented), students will gain a greater understanding of the costs in both speed and space of these algorithms. Communications experience is obtained during the implementation of a simple communication system.

The components and construction of a digital radio, shown in Figure 1, will be described in three sections. The first section deals with the implementation details of the software-defined radio; this includes the basic hardware configuration, performance, and costs. The second section of this paper concerns the software details of the system. The third and final section of this paper addresses the educational tasks to be performed in a laboratory-oriented course based on the system described in this paper.

This paper assumes an elementary knowledge of software-defined radio concepts. These concepts include digital signal processing, RF engineering, and some basic communications theory.



A SINGLE SEMESTER SOFTWARE DEFINED RADIO TRANSCEIVER

Steven Brown

Penn State University
smbrown@cse.psu.edu

Justin Ford

Penn State University
jrford@psu.edu

Kyusun Choi

Penn State University
kyusun@cse.psu.edu

SDR Technology Commercial

PENNSSTATE



- AM and FM radio
- TV
- Walkie Talkie
- Cell phone
- Cordless phone
- Garage door opener
- Car door opener
- WiFi/802.11
- GPS
- Shortwave radio
- Remote control car/toy



SDR Technology Military



Image source: <http://www.deutschdao.com/> <http://www.nu-cast.com/> <http://www8.hawaii.edu/~mcaasw/Psde/Military/Military%20New/carter-1.jpg> <http://www.mathworks.com>
<http://www.shoppingathome.com/Military%20Wireless.htm>

How are They Similar?

PENNSSTATE



- **Engineers design them!**
- **The students of today are the engineers of tomorrow.**



Image source: <http://www.dsl.psu.edu/academics/>



Image source: http://www.engr.psu.edu/ae/ecc/contact_us.asp

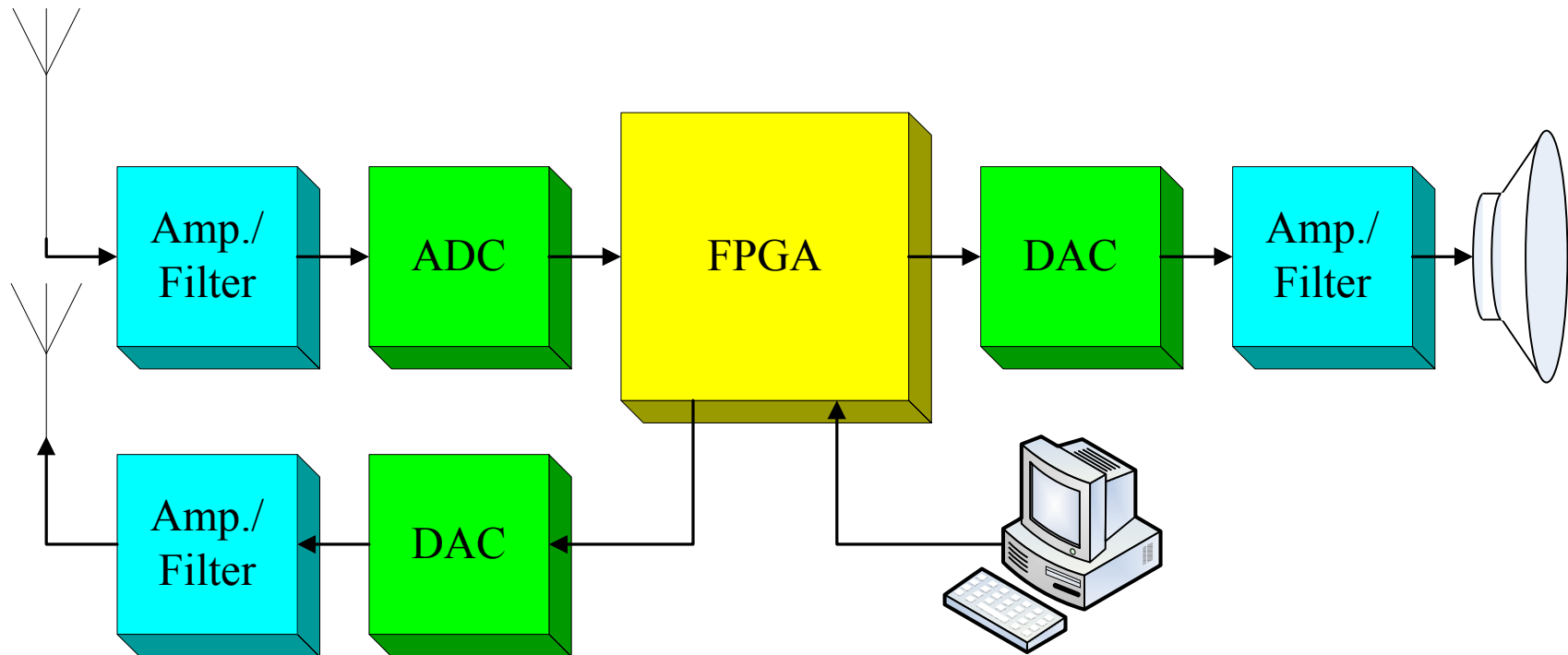


Why Build a Digital Radio?

- Provide a platform for practical experience
 - Radio Circuit Design
 - Digital Signal Processing
 - Communications Theory
- Demonstrate concepts
- Stimulate in-class discussion

Hardware Architecture

PENNSTATE



Budgetary Concerns

PENNSSTATE



- \$100 – FPGA board (1x = \$100)
- \$150 – ADC board (1x = \$150)
- \$3.31 – DAC (1x = \$3.31)
- \$1.95 – 3x potentiometers (1x = \$0.65)
- \$6 – 2x antennas (1x = \$3)
- \$4 – headphone jack, 1/8" (1x = \$4)
- \$20 – 4x amplifiers (1x = \$5)
- \$5 – 10x capacitors (1x = \$0.50)
- \$1 – 10x resistor (1x = \$0.10)
- \$7.95 – 3x 40 pin connectors (1x = \$2.65)
- \$11.63 – 96 pin connector (1x = \$11.63)
- \$9.50 – 10' of 40 conductor ribbon (1x = \$9.50)

- Total: \$320.34

Project 1

PENNSSTATE



- **Connect an ADC board to an FPGA board**
- **Recommended performance: 10 Msps**
- **Our reference design: 12 Msps**
- **Teaches practical system design**

Board Choice

PENNSSTATE



What FPGA board?

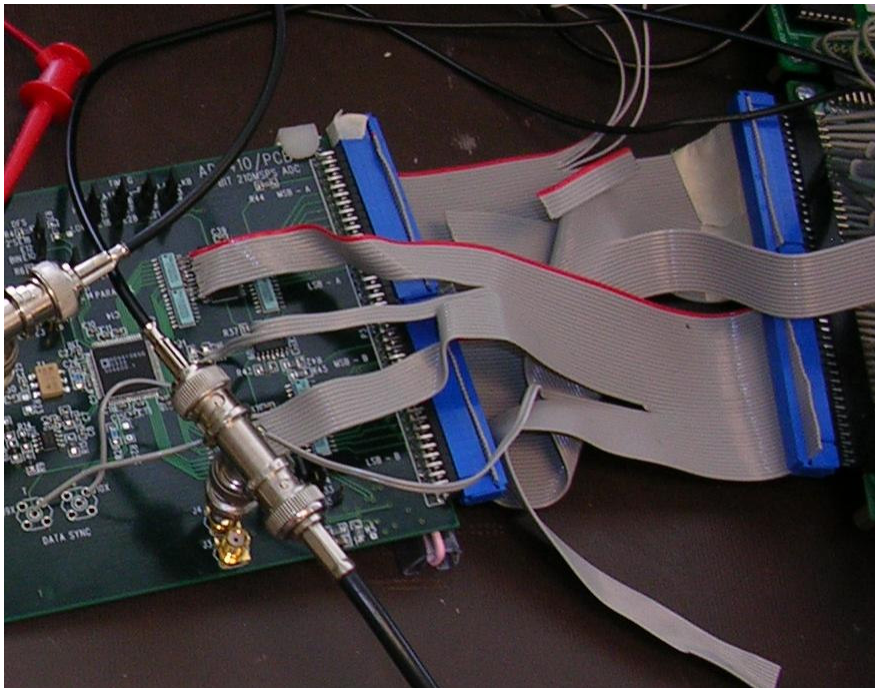
- FPGA size
- FPGA speed
- Board connectors
- Other features:
switches, LEDs, etc

What ADC board?

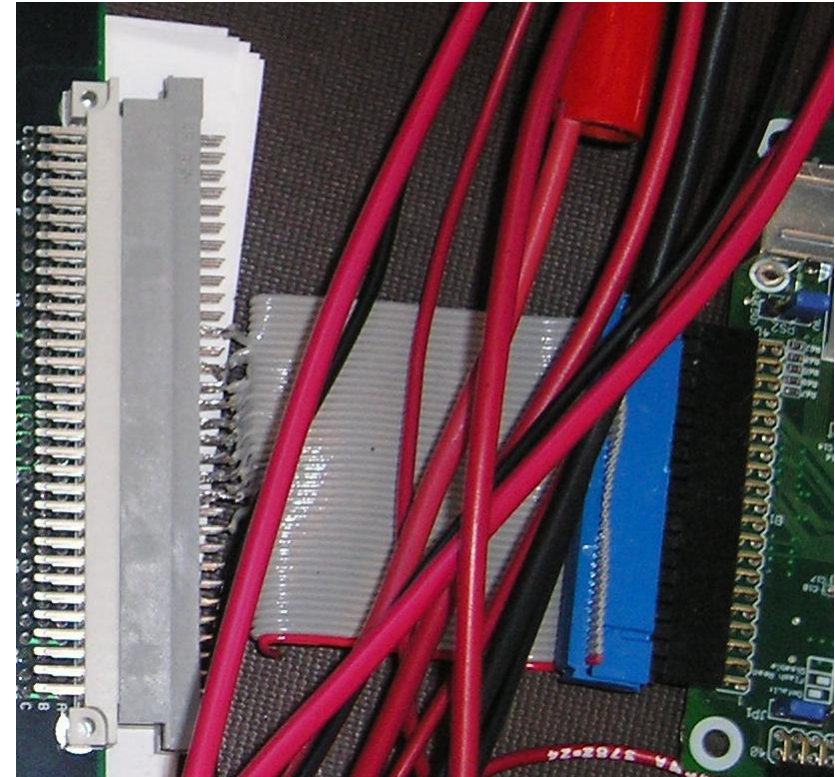
- Sampling rate
- Sampling precision
- Board noise
- Board connectors

Choose the Right ADC Board!

PENNSSTATE



- Cheap connectors
- Horrible wiring
- Complex initialization



- Expensive connectors
- Easy wiring
- Latch-based interface

Project 2

PENNSSTATE



- **Connect a DAC to the FPGA**
- **Create a Direct Digital Synthesizer**

- **Recommended performance: 10 Msps**

- **Our reference design: 6 Msps**

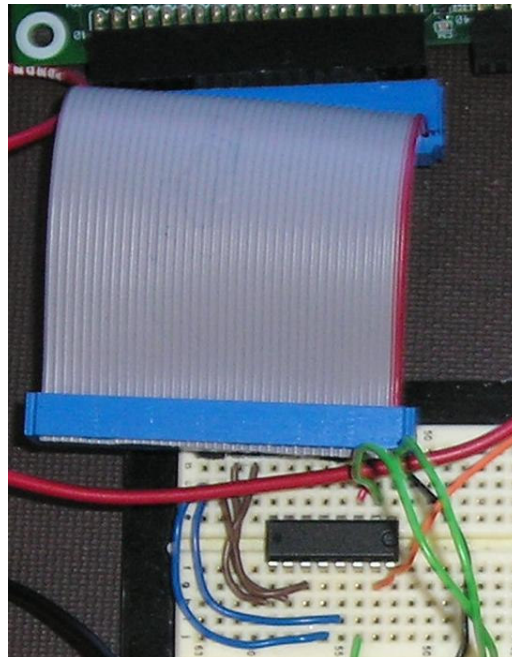
- **Teaches practical system design**

Which DAC?

PENNSSTATE

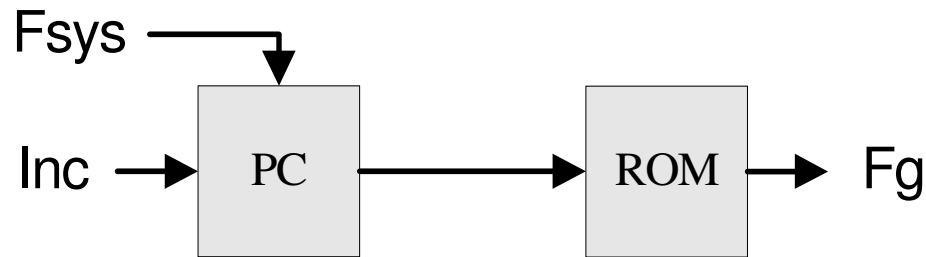


- High speed, but not blazing: 10 Msps is ok
- Low precision: 8 bit works
- Does not require an evaluation board
- Simple cabling (only 9 wires used below)



A Direct Digital Synthesizer

PENNSSTATE



- **P**rogram **C**ounter is a phase accumulator
- ROM stores a sine table
 - 127 entries
 - 8th bit of the PC is the sign of the output value

Project 3

PENNSSTATE



- **Create a filter**
- **Recommended performance: 10 Msps**
- **Our reference design: 12 Msps**
- **Teaches signal processing**

Finite Impulse Response (FIR)

PENNSSTATE



- Fixed point numbers will work
- Requires many cheap operations
- Pipelining (fast)
- Frequency hopping is expensive (>50 coefficients)
- Large



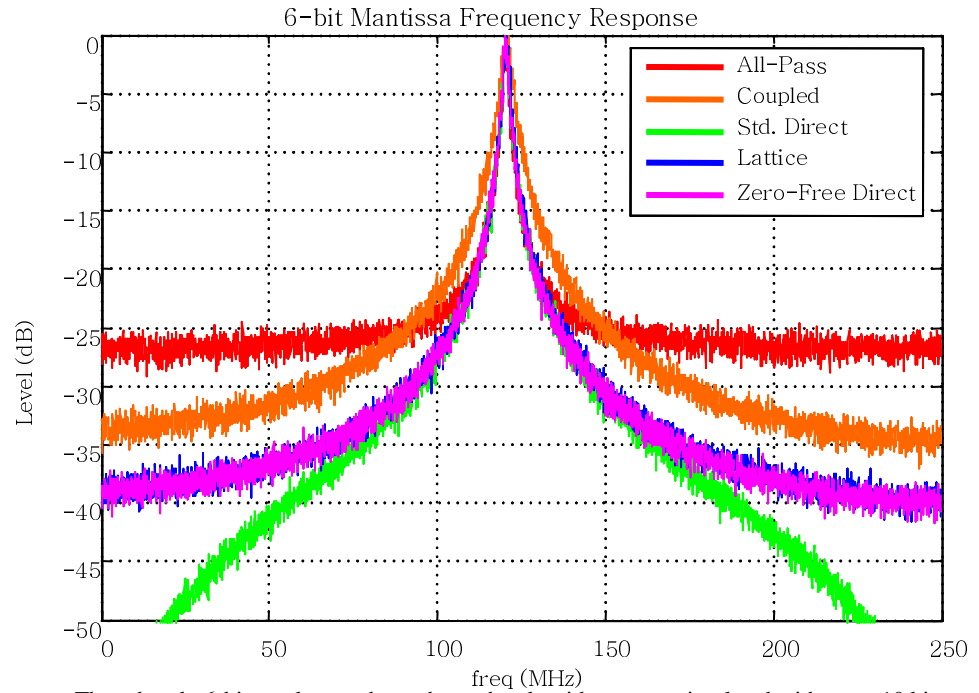
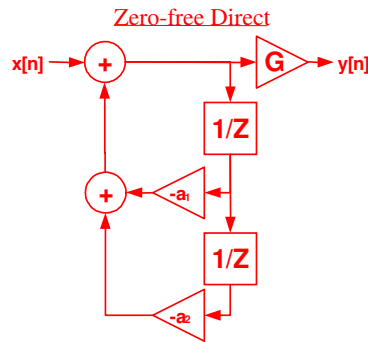
Infinite Impulse Response (IIR)

- Floating point numbers required
- Requires few but expensive operations
- Frequency hopping is cheap (requires updating of ~4 coefficients)
- No pipelining
- Speed limited by the performance of the FPGA, not the clock speed

Filter Design

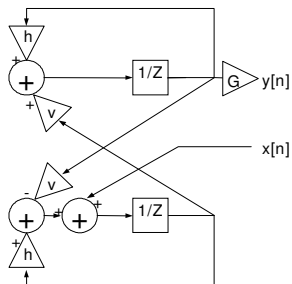


Filter Structure	# Add	# Mult	Worst case latency
Std. Direct	3	3	$2 \times ADLAT + 1 \times MULAT$
All-pass	7	2	$4 \times ADLAT + 2 \times MULAT$
Lattice	3	4	$3 \times ADLAT + 2 \times MULAT$
Coupled	3	5	$2 \times ADLAT + 1 \times MULAT$
Zero-Free Direct	2	3	$2 \times ADLAT + 1 \times MULAT$

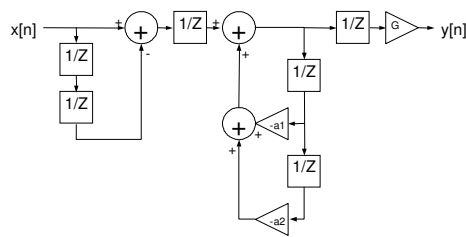


Though only 6-bit results are shown here, the algorithms were simulated with up to 10 bits of mantissa. The number of exponent bits was not as significant to the quality of the results.

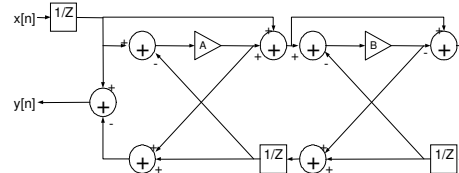
Coupled



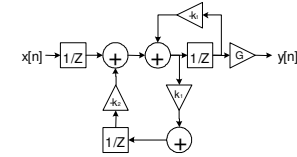
Standard Direct



All Pass



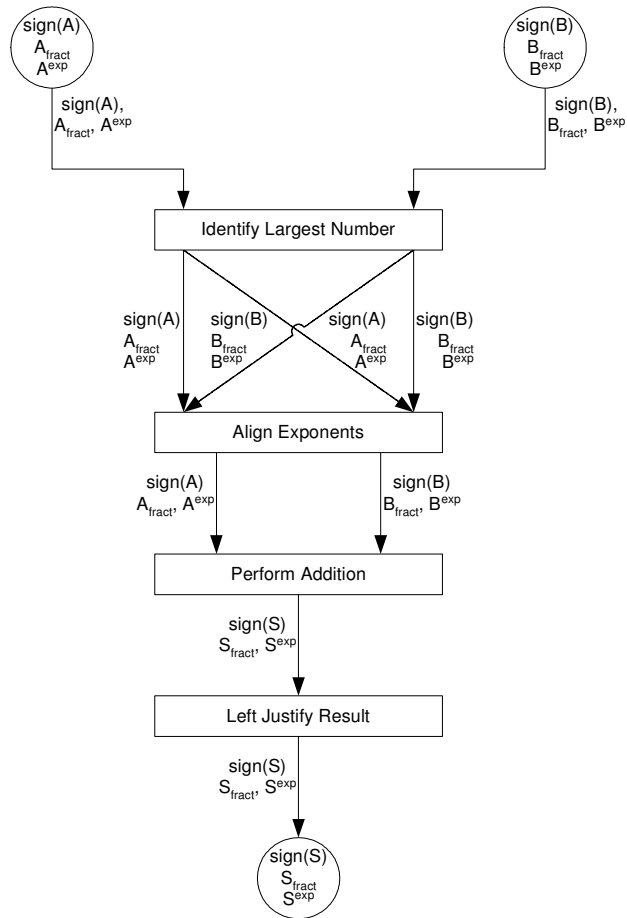
Lattice



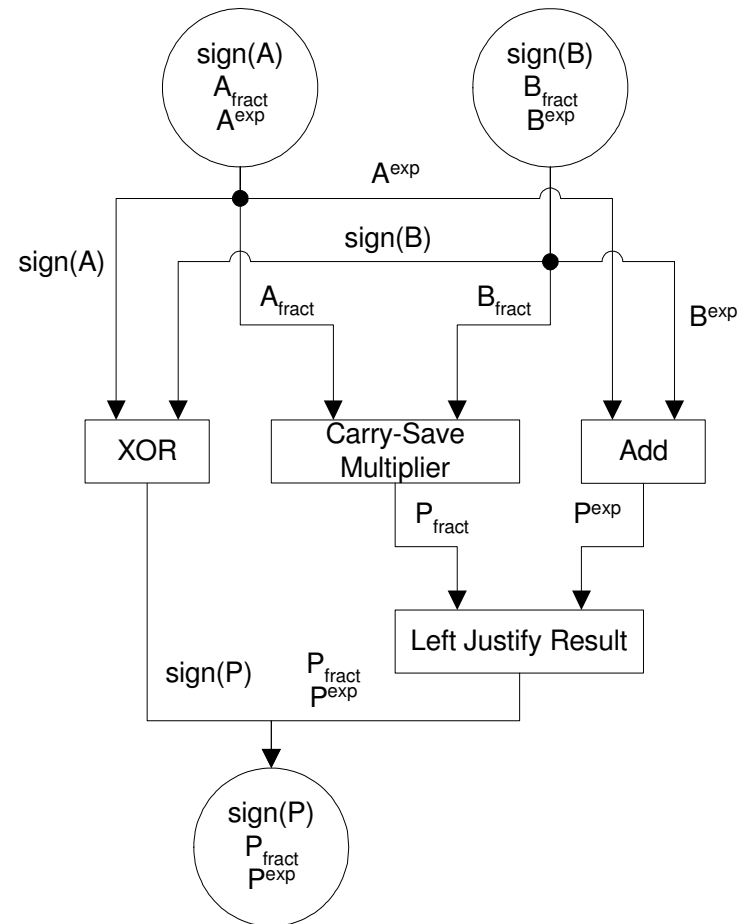
Floating Point



Floating Point Addition



Floating Point Multiplication



Project 4

PENNSSTATE



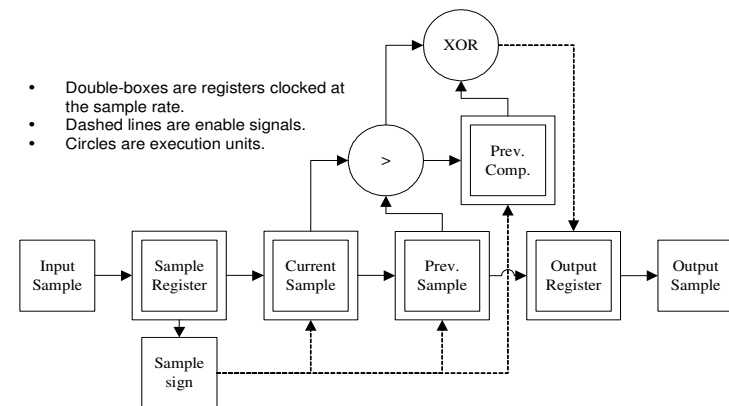
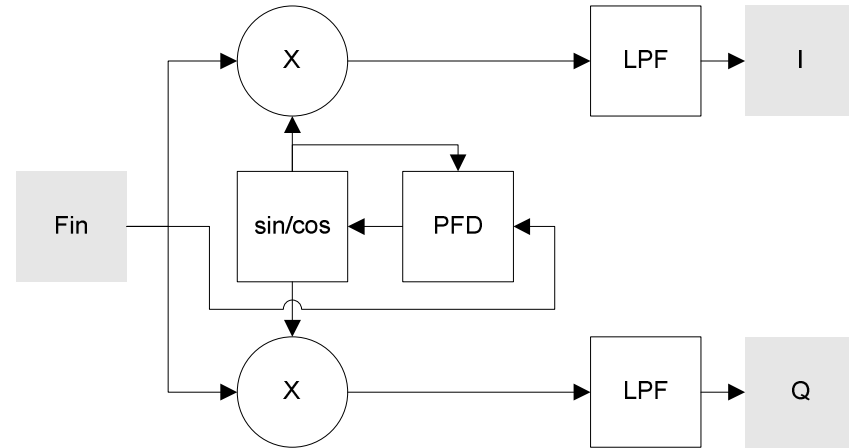
- **Create an AM demodulator**
- **Recommended performance: observable demodulation**
- **Our reference design: observable demodulation**
- **Teaches signal processing and communications theory**

AM Demodulation

PENNSSTATE

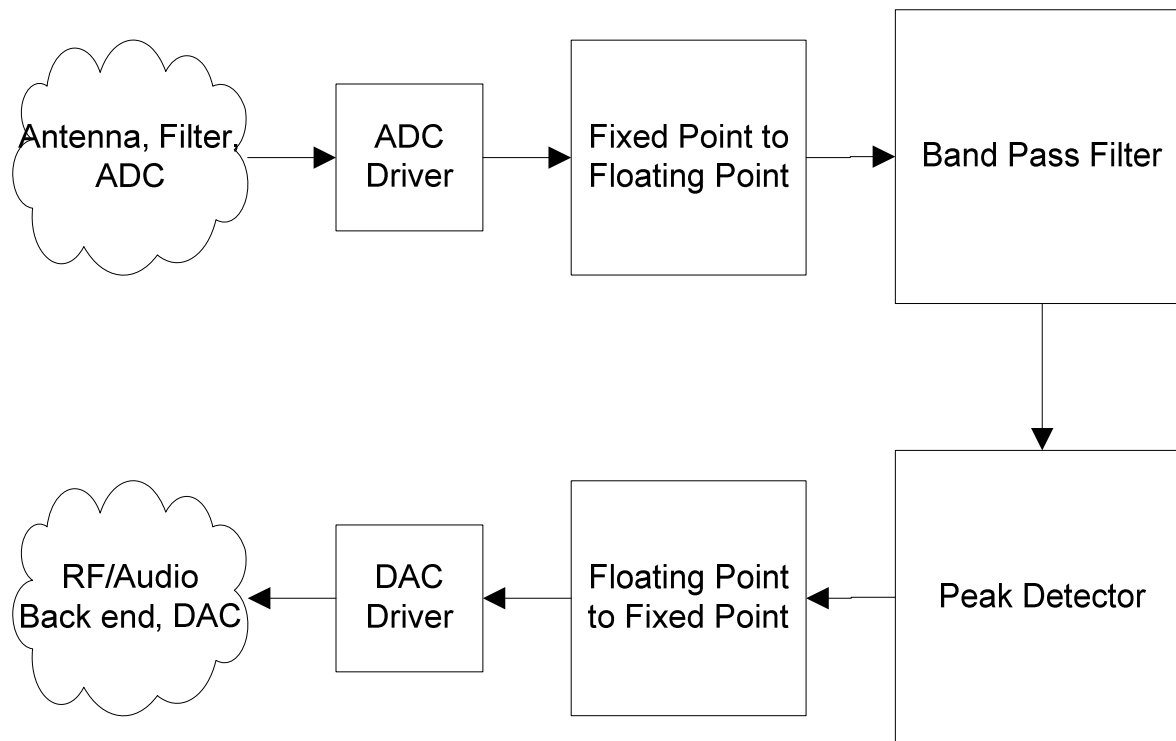


- Zero-IF, filter
 - Quadrature Down Converter
 - I/Q decomposition with carrier recovery
- Low pass filter (our approach)
 - Simple low pass, Peak detector
 - Higher quality low pass, FIR



AM Software Architecture

PENNSSTATE



Project 5

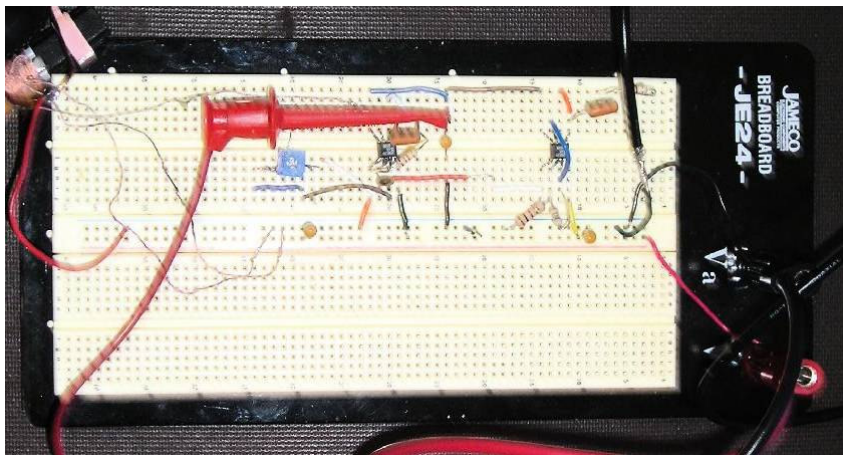
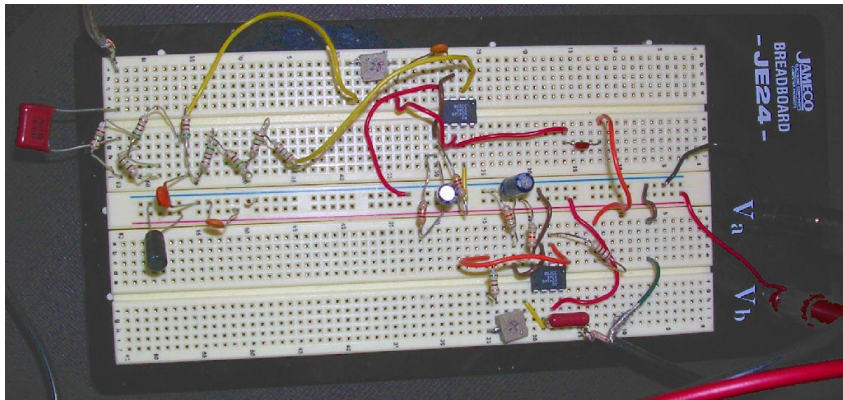
PENNSSTATE



- **Create an AM radio receiver**
- **Recommended performance:
audible radio reception**
- **Our reference design: audible radio
reception**
- **Teaches RF design**

RF Front-end

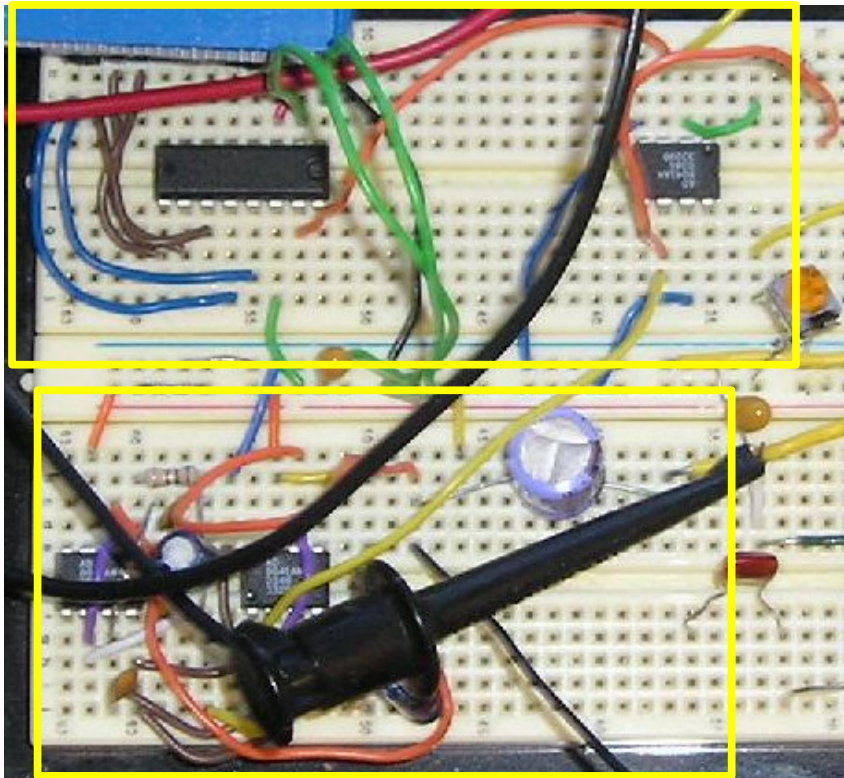
PENNSSTATE



- Broadband or narrowband?
- What about the antenna?

Audio Back-end

PENNSTATE



- DAC
- Audio range filter

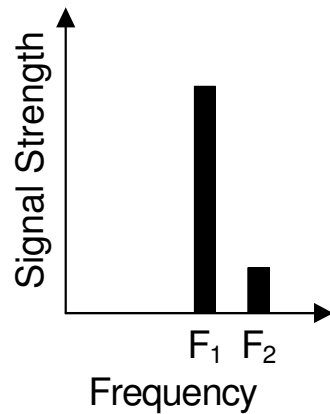
Project 6

PENNSSTATE

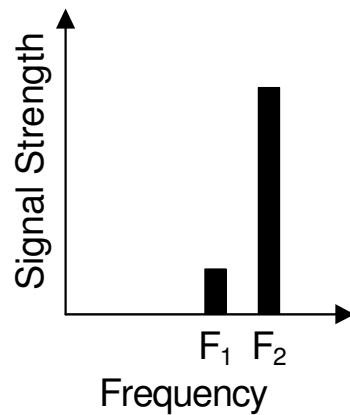


- **Create a BFSK transceiver**
- **Recommended performance: > 300 bits/second**
- **Our reference design: 4800 bits/second (peak)**
- **Teaches RF design and communications theory**

What is BFSK?



This figure shows a '0' transmission.

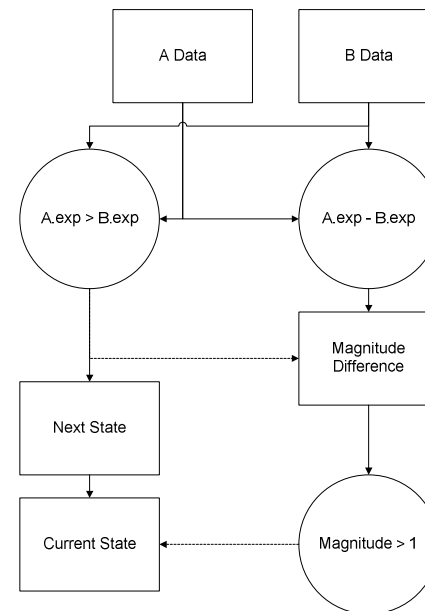
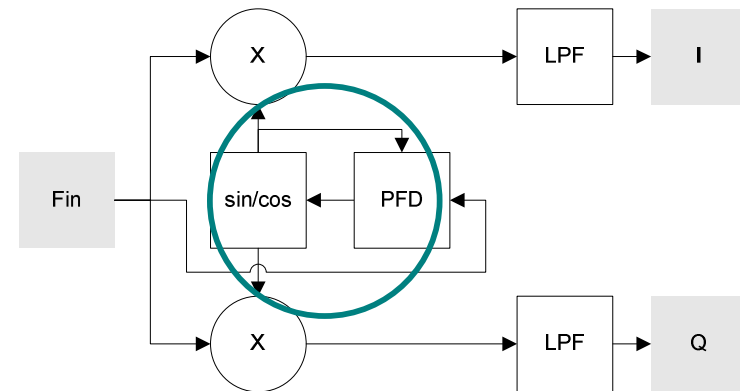


This figure shows a '1' transmission.

BFSK Demodulation



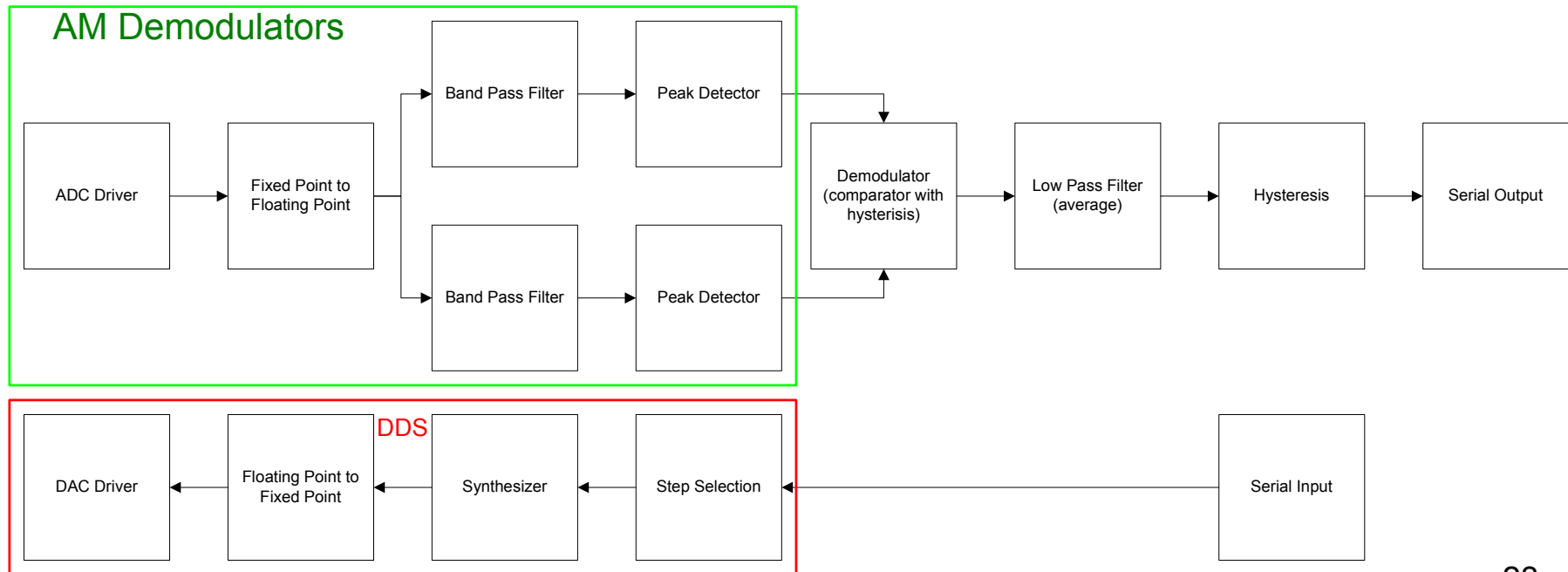
- Quadrature Down Converter
 - Full I/Q decomposition
 - PLL (half of the Down Converter)
- Heuristic method (our approach)
 - Use two AM demodulators and compare outputs



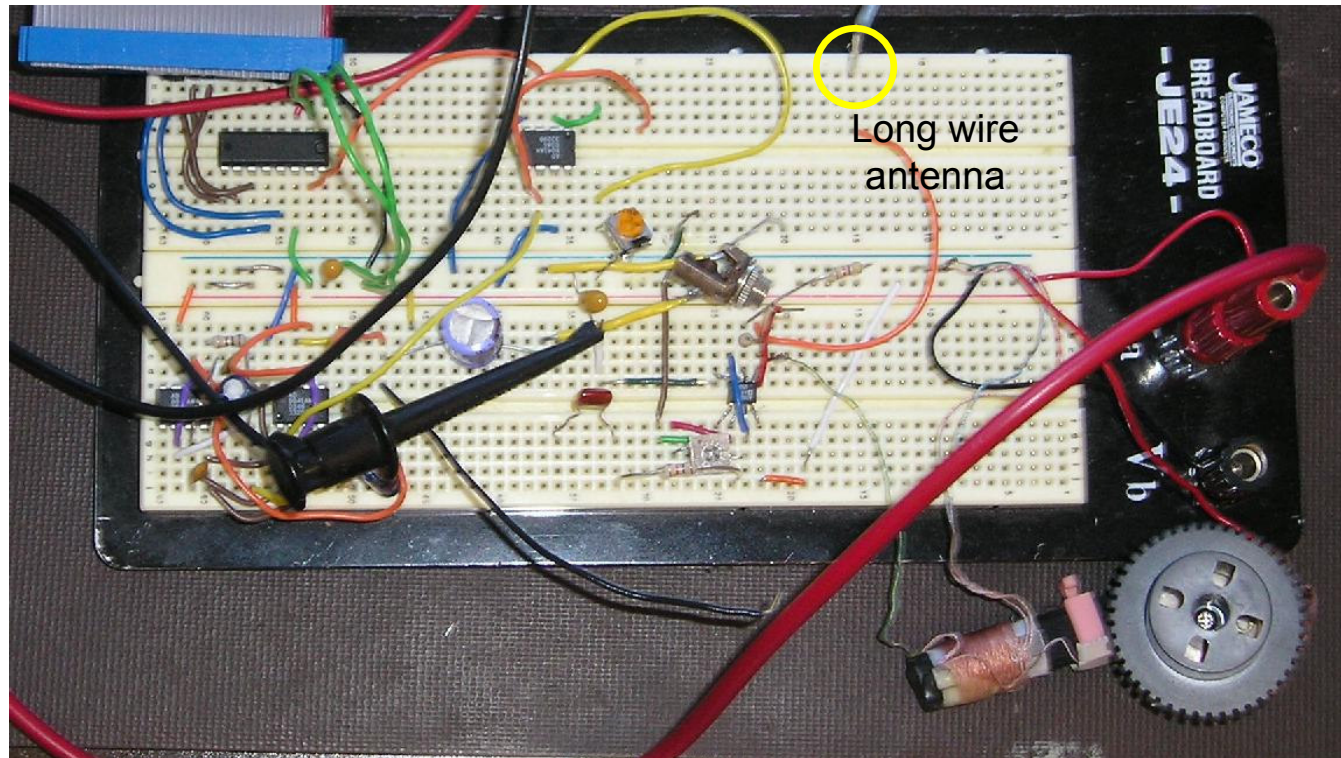
BFSK Software Architecture



- New blocks:
 - Demodulator
 - Low pass filter
 - Hysteresis



RF Back-end



- Monopole antenna
- Amplifier for RF signal buffer

Conclusion



- Objectives of the exercises:
 - Teach radio circuit design
 - Teach digital signal processing
 - Teach communications theory
- Demonstrate concepts
- Stimulate discussion

Acknowledgements and Disclaimers

PENNSTATE



This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), and administered by the Army Research Office under ESP MURI Award No. DAAD19-01-1-0504. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DAPRA), and Army Research Office.



Questions or comments?

2. HARDWARE IMPLEMENTATION

The inexpensive prototyping system shown in Figure 1 consists of a low cost field programmable gate array (FPGA) board, analog-to-digital converter (ADC), digital to analog converter (DAC), radio frequency (RF) front-end, audio frequency back-end, and RF back-end. At low carrier frequencies, below 10 MHz, our tests show that the analog components do not have stringent requirements. The prototyping system costs \$320.

The purpose of the RF front-end is to prepare the signal received by the antenna for processing by the ADC. The RF front-end must amplify and filter the received signal. Impedance matching with the antenna is important. If the chosen antenna is a broad band antenna, the demands on the filter are relatively high. On the other hand, a highly selective antenna such as an inductor-based antenna will place a high demand on the amplification needed in the front-end. In either case, high speed, low noise amplifiers are recommended. Proper impedance matching of the antenna output to the amplifier stage's input will dramatically increase the power output of the antenna [1]. A simple and proven design is invaluable for the construction of this portion of the design. Inductor-based antennas are readily available in commercial AM radios.

The ADC choice is usually a trade-off between precision and sampling rate, but when radio station reception and data communication are involved, the noise floor is also of major concern. A careful inspection of the datasheet for the ADC board chosen or created for the project is of paramount importance when radio reception is desired. A 20-dB difference in the noise floor will easily overcome a difference of 2 bits of ADC precision. Simulation and measurement of the system suggest that an 8-bit precision ADC is sufficient for short distance communication and radio reception.

The FPGA board chosen should host an FPGA of sufficient size and speed to implement the algorithms and logic needed for the system and should support enough input and output resources to drive the ADC, DAC, and other devices desired. The size of an FPGA specified in FPGA the datasheet relates indirectly to the amount of logic that can be utilized on an FPGA, due to optimizations performed during synthesis. The speed of the FPGA will determine the sampling rate supported by the FPGA. The filter may limit the sampling rate of the system more than the ADC because the filtering algorithm implemented may have a maximum operating rate that will be lower than the sampling rate of the ADC. ADCs and DACs that require complex initialization will likewise add complex logic in the FPGA; a simple interface to the FPGA will result in simple logic in the FPGA. An ADC that requires register

initialization prior to operation is more complex than an ADC that provides an output-only latch style interface.

The DAC does not have stringent requirements, unlike the ADC. Radio reception is quite audible with a low precision 8-bit DAC. Data transmission is possible with a 10 Msps or lower speed DAC.

The RF back-end, like the DAC, does not have stringent performance requirements. Due to the short-range nature of the communication, the digital harmonics from the low precision DAC will not prevent data communication. Note that the slack demands on the back-end of the system extend to the antenna used for transmission; a transmission range of only a few feet is sufficient for demonstration purposes and avoids regulatory issues [2].

The audio back-end used for radio reception has particularly low demands on the DAC. If greater audio quality is demanded, a low pass filter can be used to eliminate high frequency noise that remains after digital processing. A DAC capable of supporting data communication will have enough speed to generate audio data.

The prototyping system targets the 600 kHz to 1.6 MHz frequency band for low power communication using inexpensive components. This allows for the use of low cost resistors, capacitors, amplifiers, and DAC. The ADC board should support more than 10 mega-samples per second (Msps) in order to cover the target frequency range. Though this is also true of the DAC, data communication is more tolerant of digital noise on the transmit side of the system than the ADC is tolerant of digital noise when receiving an AM radio broadcast. With matched transmitter and receiver sampling rates, a simple comb filter is created when no other filtering is used. This asymmetric noise tolerance suggests a pre-built and properly designed ADC board, contributing significantly to the cost of this system. The FPGA board in the system also contributes significantly to the cost of the system.

3. SOFTWARE DETAILS

The software radio algorithms described in this paper perform frequency selection, baseband processing, and direct digital synthesis (DDS), all in an FPGA. The DSP algorithms to be described are efficient in terms of size and speed. The system is flexible enough that the complexity of the algorithms the students implement can be incrementally increased to encompass a tunable frequency selection filter and binary frequency shift keyed (BFSK) modulation for data communication. Students can create systems that interact with each other by also implementing the transmitter design using DDS based on a lookup table.

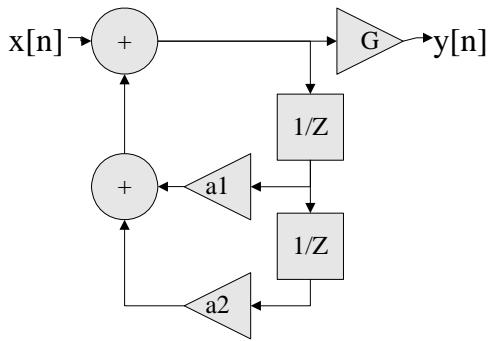


Figure 2. IIR Filter Structure

$$G = 2\sqrt{1 - (\cos^2 \omega_c)(1 + \alpha)^2 + \alpha^2 + 2\alpha \cos 2\omega_c}$$

$$a_1 = (\alpha + 1) \cos 2\pi\omega_c$$

$$a_2 = -\alpha$$

Equation 1. Calculation of Coefficients for the IIR Filter

Frequency selection can be performed via an Infinite Impulse Response (IIR) band pass filter. The filter structure, shown in Figure 2, implements an IIR filter using very few operations. The coefficient calculations to create a band pass filter are shown in Equation 1 for this structure. In this equation, α is the bandwidth parameter and ω_c is the center frequency (normalized by the sampling frequency, F_c/F_s). Though a Finite Impulse Response (FIR) filter can also implement a band pass filter, FIR filters have more complex structures than IIR filters.

Prior work has shown that this filter structure is tolerant of various floating point precisions [3]. By changing the mantissa size of the floating-point system, one can improve the speed of the operations. This also suggests that the sample width of the ADC will affect the performance of the filter but not cause complete failure. The gradual degradation of the filter's selectivity can be seen in Figure 3.

Though a large number of floating-point format implementations exist [4–7], a “light-weight” form such as that given in [8] is sufficient. These “light-weight” implementations generally support few (if any) rounding modes, smaller mantissas, and smaller exponents than the IEEE standard format. The creation of these floating-point operations requires an understanding of the number format, the ramifications of that format, and the efficiency trade-offs involved in the creation of the operations. If the floating-point implementation is in some way flawed, the filter itself may still appear to function to some degree; however, the filter characteristics will be distorted.

The baseband processing presented here is a non-coherent AM demodulator, a simple trough detector. This processor will allow high frequencies to pass and could be

followed either by a digital low pass filter or an analog low pass filter in the analog back-end to improve audio quality.

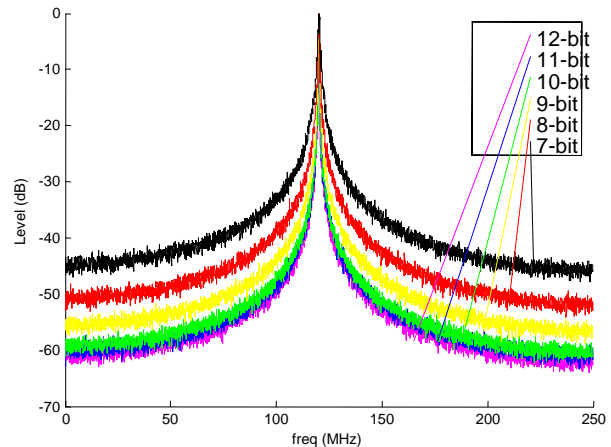


Figure 3. Filter Response at Various Precisions

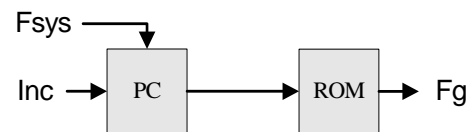


Figure 4. Structure of a Simple DDS

$$Inc = (Fd / F_{sys}) * PCw$$

$$Fg = Inc * F_{sys} / PCw$$

Equation 2. Calculation of Increment Value and Generated Frequency of the Synthesizer

The DDS proposed here consists of two components: a program counter and a read-only memory (ROM) [9]. The structure of this synthesizer is shown in Figure 4. A 128-sample ROM can be used for 8-bit audio by storing a hemisphere, rather than quadrant, of samples in the ROM. The most significant bit of the program counter is the sign of the floating-point number, generating both the positive and negative halves of the sine wave. The program counter itself should be wider than 8 bits. For a 16-bit program counter, the least significant 8 bits of the program counter can be thought of as an error accumulator (part of the phase accumulator in [9]) or as fractional bits in a fixed-point number. For example, an 8-bit program counter results in 0.2% frequency error (1-MHz system clock results in a maximum error of 2 kHz); while a 16-bit program counter results in 0.0008% frequency error (8-Hz error with a 1-MHz clock). To summarize the effects of table size from [9], the sample precision affects amplitude quantization; sample count affects the phase quantization; and, the phase accumulator size affects the frequency quantization. Equation 2 shows the calculations for calculating the increment value (the phase increment word in [9]) of the

program counter given the desired frequency (F_d), the system clock (F_{sys}), and the program counter bit-width (PC_w). In Equation 2, Inc is of the same bit-width as the program counter and must be rounded in order to maximize its accuracy (round to nearest). The value F_g in Equation 2 is the generated frequency.

4. EDUCATIONAL TASKS

The educational tasks consist of a series of laboratory exercises whose content can be tied into lecture material. Early exercises can be devoted to system assembly and integration of the core digital system, connecting the ADC and DAC to the FPGA. Later exercises can focus on higher level concepts related to communication theory, digital signal processing, and RF design as the antenna and its support circuitry and transmission amplifier are added. Six exercises are proposed here to cover these topics.

The first laboratory exercise would connect the ADC to the FPGA. In this exercise, the students would be provided with a datasheet for the ADC board and wiring diagram for the cabling between the two boards. The students would be required to create the necessary software to retrieve data from the ADC.

The second laboratory exercise would connect the DAC to the FPGA. In this exercise, the students would be provided with a datasheet for the DAC and wiring diagram for the cabling between the FPGA and a board hosting the DAC (a breadboard would suffice). The students would be required to create the necessary software to send data to the DAC. The students would be required to demonstrate that data can be transferred to the DAC from the FPGA.

The third laboratory exercise would be to create a filter within the FPGA. This assumes that the floating-point adder, multiplier, and conversion (floating-point to fixed-point and fixed-point to floating-point) modules are provided to the students, as these are non-trivial projects. The goal of this exercise would be to assemble the filter in software with proper register placement. The clock rate of the filter may need to be lowered because the IIR filter may be too slow to complete its calculation within a single clock cycle. Filter coefficients also must be encoded. These coefficients can either be hard coded, selected via switches, or selectable via an external computer (a personal computer and serial cable) with the addition of a UART, state machine, and programming interface. The functionality of the filter can be verified via function generator and oscilloscope by routing the filtered data out through the DAC.

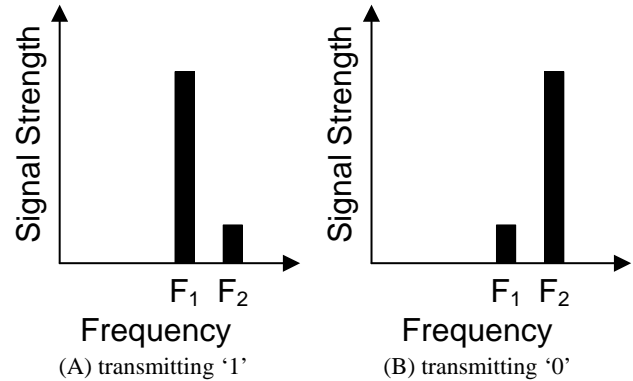


Figure 5. Output Level Encoding for BFSK

The fourth laboratory exercise in the series would be devoted to creating a non-coherent AM demodulator. A peak/trough detector is a conceptually simple means of accomplishing this task. This is a challenging task because this requires some knowledge of the floating-point number format. To test the results of the exercise, a frequency generator capable of AM modulation would be ideal. An AM modulator built from two function generators and a discrete transistor is an alternative.

The fifth laboratory exercise would be the creation of an AM receiver. The challenge of this lab is the construction of an antenna and amplifier system capable of driving the ADC's input. The recommended components for this part of the system are an inductor-based antenna with variable capacitor (easily obtainable from any dollar store AM/FM radio) and amplifier network. In order to test the system, the students may either try to tune in a sufficiently strong local station, or a kit-built AM transmitter may be used (this generates a very strong, though short ranged, signal).

The sixth and final laboratory exercise is the creation of a digital communication system. The system builds on all the previous projects and is therefore the most challenging. By instantiating two AM receivers and comparing their outputs, one can create a simple Binary Frequency Shift Keyed receiver. The two filters would be tuned to two frequencies. As shown in Figure 5, a transmitting unit would use the DDS module to generate one of those two frequencies. If received frequency F_1 is much stronger than frequency F_2 , a '1' would be received, and likewise, if received frequency F_2 is stronger than frequency F_1 , a '0' is received. A simple low pass filter can remove the high frequency noise within the received data. A simple way to generate and decode transmissions is to connect the transmit path to the serial port input to the FPGA evaluation board and the receive path to the serial port output from the FPGA (one can connect an RS-232 transceiver if the FPGA evaluation board has none).

These six exercises meet the goals of exposing students to RF engineering, digital signal processing (DSP), and communications theory. The third exercise exposes students to the implementation issues of DSP algorithms. The fourth exercise exposes students to communications theory. The fifth exercise exposes students to RF design. The sixth exercise uses all of these principles to create a communications system.

5. CONCLUSIONS

This paper has described a realizable software-defined radio. Modern signal processing techniques coupled with ADCs allow us to achieve a digital radio implementation. Both the algorithm and the arithmetic computation were designed to be efficient in terms of size and speed. The result was an AM radio receiver that was able to operate directly on the digitized carrier wave. An outline has been provided describing the system characteristics that should be considered when preparing the hardware needed for a course covering SDR. A software implementation has been shown that is capable of satisfying the needs of such a course. Finally, a series of exercises have been proposed that meet the objectives of a course on SDR.

6. ACKNOWLEDGEMENTS AND DISCLAIMER

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), and administered by the Army Research Office under ESP MURI Award No. DAAD19-01-1-0504. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the Defense Advanced

Research Projects Agency (DARPA), and Army Research Office.

7. REFERENCES

- [1] C. Hutchinson, et al., *The ARRL Handbook for Radio Amateurs*, ARRL – the national association for Amateur Radio, Newington, CT, 2000.
- [2] Office of Engineering and Technology, Federal Communications Commission, *Understanding The FCC Regulations for Low-power, Non-licensed Transmitters*, Federal Communications Commission, Columbia, MD, 1996.
- [3] L. M. Naji. *Comparison of Finite Precision Bandpass IIR Filter Structures for Software Radio Systems*. M.S. Paper: Pennsylvania State University, 2004.
- [4] A. A. Gaffar, O. Mencer, W. Luk, and P. Y.K. Cheung, “Unifying Bit-width Optimisation for Fixed-point and Floating-point Designs,” *Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th Annual IEEE Symposium on*, pp. 79–88, 20–23 April, 2004.
- [5] L. Louca, T. A. Cook, and W. H. Johnson, “Implementation of IEEE Single Precision Floating Point Addition and Multiplication on FPGAs,” *FPGAs for Custom Computing Machines, 1996. Proceedings. IEEE Symposium on*, IEEE, pp. 107–116, 17–19 April 1996.
- [6] K. Ralev and P. Bauer, “Asymptotic Errors of Floating Point Digital Filters,” *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, pp. 585–588, May 2004.
- [7] D. Williamson, S. Sridharan, and P. McCrea, “A New Approach for Block Floating-Point Arithmetic in Recursive Filters,” *Circuits and Systems, IEEE Transactions on*, vol. 32, pp. 719–722, July 1985.
- [8] C. Souani, M. Abid, and R. Tourki, “An FPGA Implementation of the Floating Point Addition,” *Industrial Electronics Society, 1998. IECON '98. Proceedings of the 24th Annual Conference of the IEEE*, vol. 3, pp. 1644–1648, 31 Aug. –4 Sept. 1998.
- [9] J. Vankka, “Methods of Mapping from Phase to Sine Amplitude in Direct Digital Synthesis,” *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, vol. 40, issue 9, pp. 1978–1982, Sept. 2005.