

UNIVERSITY OF OKLAHOMA
GRADUATE COLLEGE

DIGITAL RADAR RECEIVER DESIGN BASED ON HIGHLY EFFICIENT
BANDPASS SAMPLING FPGA ARCHITECTURE

A THESIS
SUBMITTED TO THE GRADUATE FACULTY
in partial fulfillment of the requirement for the
Degree of
MASTER OF SCIENCE

By

JOHN MEIER
Norman, Oklahoma
2009

MEI
THESIS
cop. 2

DIGITAL RADAR RECEIVER DESIGN BASED ON HIGHLY EFFICIENT
BANDPASS SAMPLING FPGA ARCHITECTURE

A THESIS APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY:

[Redacted Name]

Dr. Mark Yearly, Chair

[Redacted Name]

Dr. Robert Palmer

[Redacted Name]

Dr. Monte Tull

© Copyright by JOHN MEIER 2009
All Rights Reserved.



ACKNOWLEDGMENTS

I would like to thank all those who made this research possible. Firstly, I offer many thanks to my advisor and mentor Dr. Mark Yeary for his recognition, encouragement, and support for me through all these years of research. Under his guidance I have learned and experienced more than I thought possible.

If research could be converted into the flow of electrons and researchers into copper traces on a printed circuit board, then Redmond Kelley is the complementary wire in our differential pair of a research team. I am supremely grateful to have such a perspicacious partner. His ingenuity is only exceeded by his unparalleled dedication, friendship, and humor.

I also appreciate Dr. Robert Palmer and Dr. Monte Tull, who complete my thesis committee and have also provided excellent foundations of knowledge for this research in the fields of weather radar and computer hardware design, respectively. Thanks to all of the students and faculty of the ARRC, especially the Radar Innovations Laboratory, for allowing me to draw from their combined skills and knowledge abundantly. Special thanks go to the staff of the Electrical and Computer Engineering and ARRC offices, who spent countless hours helping me navigate the waters of procurement so that the raw materials of research could be acquired. Their patience and helpfulness are boundless.


I am also thankful for the unending support and encouragement outside the lab from Stephenie Newton, who was always there for me when I needed help most. It was only through her motivation that I was able to achieve my goals. Finally, for those who have believed in me from my youngest years, I give thanks to my parents, brother, grandparents, cousins, and everyone else I can count as family.



TABLE OF CONTENTS

Acknowledgments	iv
Table of Contents	v
List of Tables	vii
List of Figures	viii
Abstract	x
1 Introduction	1
1.1 Digital Radar Receiver Design	1
1.2 Design Approach	2
2 Digital Receiver Design Considerations	4
2.1 Bandpass Sampling	4
2.2 Clock Jitter	7
2.3 Metastability in Bistable Circuits	8
3 Single-Channel Receiver Design	11
3.1 Hardware Design	11
3.1.1 Analog-to-Digital Converter	12
3.1.2 Field-Programmable Gate Array	13
3.1.3 Host Computer	16
3.1.4 Hardware Development	17
3.2 Software Design	18
3.2.1 FPGA Firmware Structure	18
3.2.2 Host Computer Software Operations	24
3.3 Experimental Results	25
3.3.1 Laboratory Testing	26

3.3.2	Field Testing	27
4	Eight-Channel Receiver Design	35
4.1	Hardware Design	35
4.1.1	Analog-to-Digital Converter	36
4.1.2	Field-Programmable Gate Array	42
4.1.3	Peripheral Devices	43
4.1.4	Clock Conditioner	45
4.1.5	Host Computer	46
4.1.6	Hardware Development	47
4.2	Software Design	49
4.2.1	FPGA Firmware Structure	49
4.2.2	Host Computer Software Operations	56
5	Conclusions and Future Work	58
5.1	Conclusions	58
5.2	Scope of Effort	59
5.3	Future Work	60
	List of References	62
	APPENDIX A VHDL Deserializer Code for the Eight Channel Digital Receiver	65
	APPENDIX B VHDL In-phase and Quadrature Mixing and Decimation Filtering Code for the Eight Channel Digital Receiver	73



LIST OF TABLES

1	Single-Channel Receiver Power Consumption Characteristics	27
2	Selected National Weather Radar Testbed Specifications	28
3	Interface Signal Characteristics and Requirements at the National Weather Radar Testbed	29
4	Control Packet Structure	54
5	Control Request Packet Opcodes	55
6	Control Response Packet Opcodes	56



LIST OF FIGURES

1	Bandpass Sampling Down-Conversion Block Diagram	5
2	Frequency Flow Diagram Depicting Down-Conversion Process	6
3	Graph of Theoretical SNR Achieved vs Clock Jitter	8
4	Bistable Device Experiencing a Metastable Condition	9
5	Second-order Synchronizer	10
6	Single-Channel Digital Receiver Interface Diagram	12
7	Single-Channel Digital Receiver ADC Differential Input Connections . .	13
8	Single-Channel Digital Receiver ADC Parallel Output Connections . .	14
9	Hand-soldered 100-pin FPGA Package on a Prototype Printed Circuit Board	14
10	Single-Channel Digital Receiver Printed Circuit Board Layer Stack-up .	15
11	Single-Channel Digital Receiver Printed Circuit Board Split Power Plane Layer	15
12	Single-Channel Digital Receiver FPGA Floorplan Utilization	16
13	Single-Channel Digital Receiver Prototype Setup During Field Testing .	17
14	Fabricated and Assembled Separable ADC and FPGA Digital Receiver Submodules	18
15	Signal Layout of the Combined Single-Channel Digital Receiver	19
16	Fabricated and Assembled Combined Single-Channel Digital Receiver .	19
17	Functional Block Diagram of the Single-Channel Digital Receiver FPGA Firmware	20
18	Bandpass Sampling Method of Simplification for I/Q Signal Generation	21
19	128-tap Low-Pass Decimation Filter Response	22
20	Falling Edge of the System Trigger at the NWRT	23
21	Zoomed-in View of Figure 20	23
22	Single-Channel Digital Receiver Input and Output Interface Timing Diagram	24

23	Labview VI Capturing Data Processed by the Digital Receiver	25
24	Graphical Labview Programming Code	26
25	FFT of Digital Output with +10 dBm 50.1 MHz Input Signal	27
26	Phased Array Radar at the National Weather Radar Testbed	28
27	Single-Channel Digital Receiver Clutter Test Data	30
28	NWRT Receiver Clutter Test Data	30
29	Single-Channel Digital Receiver Reflectivity Test Data	32
30	KTLX Level-II Reflectivity Data	32
31	Single-Channel Digital Receiver Radial Velocity Test Data	34
32	KTLX Level-II Radial Velocity Data	34
33	Eight-Channel Digital Receiver Interface Diagram	37
34	AD9252 ADC in a 64-pin LFCSP with Differential Input Filters	37
35	Analog Input Connection Layout Design	39
36	Analog Input Filter Schematic	39
37	Digital Receiver Ground Plane Layout	41
38	Eight-Channel Digital Receiver Printed Circuit Board Stackup	43
39	External Watchdog Schematic	44
40	Eight-Channel Digital Receiver Evaluation Board Setup	48
41	Eight-Channel Digital Receiver Printed Circuit Board 3D Model	50
42	Eight-Channel Digital Receiver Enclosure	50
43	Eight-Channel Digital Receiver Block Diagram	51
44	Minimal Eight-Channel Digital Receiver 128-tap Decimation Filter Re- sponse	52
45	Eight-Channel Digital Receiver Output Word Format	52
46	Eight-Channel Digital Receiver FPGA Floorplan Utilization	57



ABSTRACT

As digital electronics become faster and more efficient, it becomes possible to move the analog/digital interface in a radar downconversion system further towards the antenna. Instead of digitizing radar echoes at the end of the down-conversion process, digital logic can perform the same operations previously performed by analog components. Taking full advantage of this opportunity will result in a more highly integrated and reconfigurable design. By removing unnecessary analog components, the error from component variability and noise injected into the signal of interest is reduced, the size of the receiver and the power required for operation is minimized, and the overall cost of the system can be lowered. This research is focused on employing software defined radio concepts for weather observation, thus creating a low-cost digital radar receiver at the University of Oklahoma for use in radar projects as a way of obviating the need for commercial radar receivers, which can be many times more expensive. Software-defined radio techniques, such as bandpass sampling, are used to achieve a high data processing bandwidth and oversampling ratio with the smallest logic resource utilization.

Two novel digital receiver designs are discussed in this thesis. A prototype compact single-channel digital radar receiver based on a 14-bit analog-to-digital converter and a hand-solderable Xilinx FPGA was built and tested both in the laboratory and at the National Weather Radar Testbed (NWRT). Building on the lessons learned from testing the single-channel digital radar receiver, a second digital receiver was designed for expanded capabilities. Through the utilization of a low-power, simultaneous-sampling eight channel ADC with high-speed serial data links and a cost-efficient FPGA with integrated DSP slices, eight data channels can be digitized, processed and transferred at the same time in a compact form factor. An ethernet interface has been included which allows for a scalable control channel so that the digital receiver's operations can be quickly modified. This also makes it possible to remotely change the firmware of the FPGA in seconds, without the need for physical access. Development of host computer platforms to store and process each digital receiver's output data are also discussed.



CHAPTER 1: INTRODUCTION

1.1 Digital Radar Receiver Design

Digital radar receivers exhibit many advantages over analog radar receivers while also presenting many new challenges. In an analog radar receiver, all components in the design introduce additive noise and distortion. Each component must also be physically implemented, requiring an excessive amount of space for a complex design and introducing problems from interference and power dissipation. In a digital receiver, after the returned intermediate frequency (IF) signal has been converted into digital form, degradation from component interconnections and any adjacent signal interference is eliminated in a well-designed system. A significant improvement in signal-to-noise ratio (SNR) over analog receivers can be realized by transforming the IF signal into the digital domain as quickly as possible after the receive antenna through using these techniques.

As a unique aspect of his research program, the author has employed software defined radio (SDR) techniques for the design of the digital receiver modules that are devoted to environmental measurements. As described by Mitola [1], the receiver of a SDR uses an ADC to capture all of the channels of the software radio node. Then the receiver extracts, downconverts, and demodulates the channel waveform using software on a general purpose digital processor. SDRs initially began as a military concept for compact radio communication systems known as SpeakEasy [2]. These concepts have been recently adopted into commercial communication applications [3]-[11], and are quickly becoming the heart of new cognitive systems [12]-[14]. The utility of these concepts have however been untouched by the environmental radar community who routinely make atmospheric measurements. This thesis focuses on the creation of an SDR-based digital receiver which can be applied to a wide array of radar systems due to the low power consumption, small form factor, and flexible architecture of the design.




1.2 Design Approach

The overall design approach for this research is as follows:

- Select possible hardware components based on the design requirements.
- Acquire evaluation modules of hardware to determine capabilities, prove core receiver functionality, identify critical design parameters, and define ancillary functionality.
- Create a schematic diagram to define the connections between components.
- Place components and route connections to best adhere to identified design parameters and manufacturer recommended layout guidelines.
- Fabricate and assemble custom printed circuit board.
- Define the internal structure of the Field-Programmable Gate Array (FPGA) with very high-speed integrated circuits (VHSIC) hardware description language (VHDL) language.
- Perform a behavioral simulation in Modelsim to verify code design.
- Use Xilinx tools to synthesize, translate, map, place and route, and program the design into the internal logic of the selected FPGA and attached non-volatile memory.
- Verify successful end-to-end operation of all functionality through simulated in-lab and field testing.

The remainder of the thesis is organized as follows. Chapter 2 analyzes design considerations applicable to both the single and eight-channel receiver systems, including bandpass sampling, clock jitter effects, and metastable conditions. Chapter 3 details the single-channel digital receiver while Chapter 4 explains the eight-channel system design. Chapter 5 concludes the thesis, explains the scope of the research, and describes the future work which might be possible to improve the design. Two modules



of VHDL source code have been included. The code for the high-speed deserializer module has been included in Appendix A and is explained in Section 4.2.1. The code which performs in-phase and quadrature component mixing and low-pass decimation filtering on two input channels has been included in Appendix B. These two appendices together comprise approximately 1/10th of the code used in the eight-channel digital receiver design.



CHAPTER 2: DIGITAL RECEIVER DESIGN CONSIDERATIONS

Many of the digital receiver's characteristic parameters are largely decided by the conversion process to the digital domain. Because of this, the choice of analog-to-digital converter (ADC) and design of associated input signals and circuitry is critical to system performance. The following sections discuss high-level design considerations for both the analog-to-digital converter and the field-programmable gate array (FPGA) in order to achieve the best possible overall digital receiver specifications.

2.1 Bandpass Sampling

Both the single-channel and eight-channel digital receiver designs share a common method of analog to digital conversion known as bandpass sampling. Without the need to precisely reconstruct the input signal, the digital receiver may sample at less than twice the input frequency, so long as the sampling rate still meets the Nyquist criteria for the bandwidth of the baseband input signal. With this in mind, the bandpass sampling technique is a special form of undersampling which reduces resource utilization and clock frequency requirements. Because of the tradeoff that exists between an ADC's sampling rate and its overall performance, lowering the required sampling rate for a digital receiver has two primary effects: a slower, more precise ADC may be selected, and the dynamic power consumed by all logic when oscillating is reduced for the entire system.

Some limitations of the bandpass sampling method exist due to the finite analog input bandwidth of the analog to digital converter and the increasingly stringent phase noise requirements for the analog sampling clock in order to maintain a high signal-to-noise ratio. A discussion of the sampling clock limitations continues in Section 2.2.

Figure 1 shows the method of bandpass sampling used to digitize the input signal. Here the input signal spectrum is centered at an intermediate frequency (IF) f_{IF} and sampled at a frequency f_s . For a given sampling frequency, this method is valid for

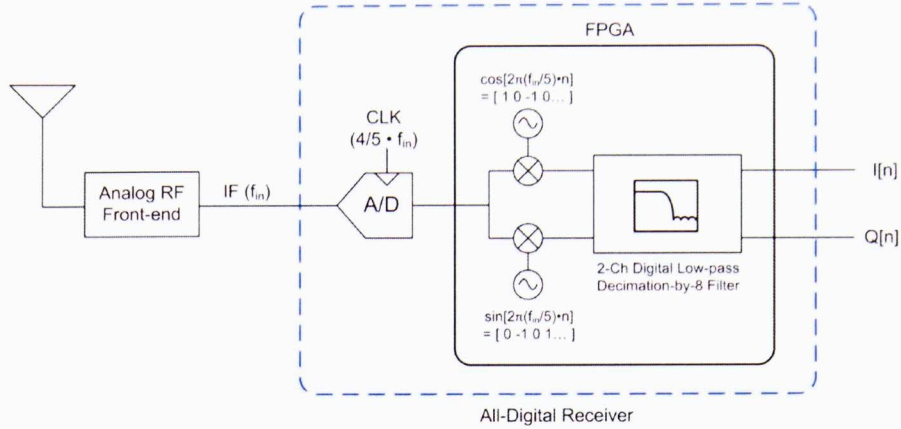


Figure 1: Bandpass Sampling Down-Conversion Block Diagram

many IF frequencies as shown in Equation (1).

$$f_{IF} = (f_s/4) * (1 + 4n) \quad n = 0, 1, 2, \dots \quad (1)$$

For this digital receiver design, n was chosen to be 1. This places the input signal in the third Nyquist zone at $f_{IF} = 5/4 * f_s$, where after digitization it will become aliased so that it is centered at $f_s/4$. In order to generate the baseband in-phase (I) and quadrature (Q) signals, the digitized input signal is multiplied with sine and cosine signals at frequency $f_s/4$. Generation of these signals can be simplified using Equations (2) and (3) so that trigonometric look-up tables need not be used, only a multiplexer and an inverter. Using this method thereby allows for faster throughput with less resource usage.

$$\sin(2\pi * f_s/4 * t) = \sin(2\pi * f_s/4 * (n/f_s)) = \{0, 1, 0, -1, \dots\} \quad (2)$$

$$\cos(2\pi * f_s/4 * t) = \cos(2\pi * f_s/4 * (n/f_s)) = \{1, 0, -1, 0, \dots\} \quad (3)$$

Figure 2 shows the spectrum of the input signal as it travels through each stage of the digital receiver. The input signal is defined as being centered at 50 MHz and having 5 MHz of bandwidth. This produces $f_{IF} = 50$ MHz, with a sampling rate $f_s = 4/5 * f_{IF} = 40$ MHz. After analog-to-digital conversion, the input signal's alias becomes

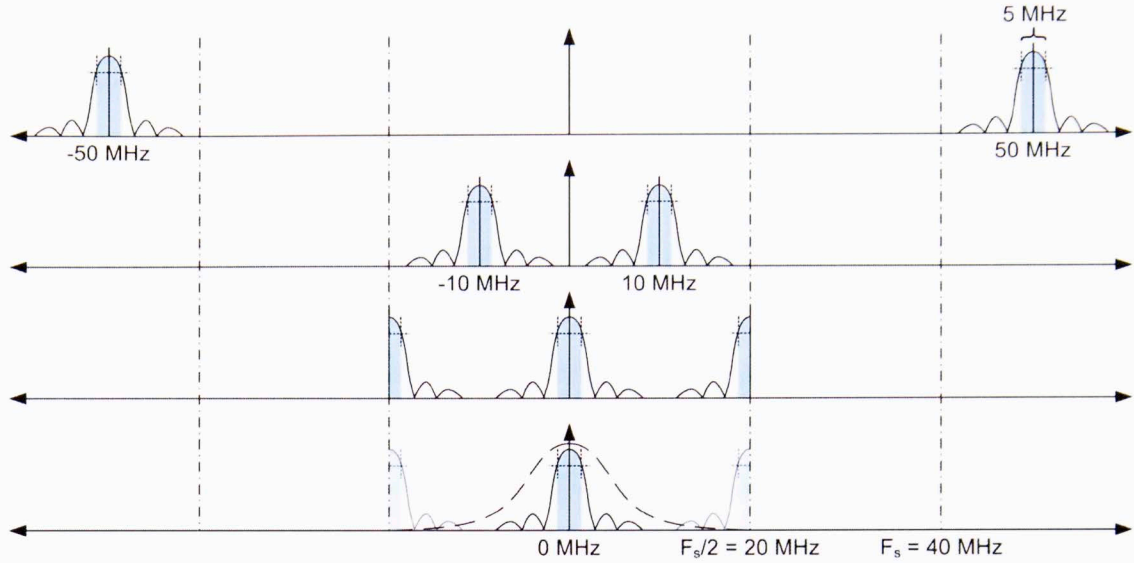


Figure 2: Frequency Flow Diagram Depicting Down-Conversion Process

centered at $f_s/4 = 10$ MHz. The I and Q signal generation is accomplished through multiplication with a sine and cosine signal, producing two images - one centered at baseband and one centered at 20 MHz. The unwanted image is removed by digitally low-pass filtering, leaving only the desired baseband I and Q signals.

While the input signal itself is undersampled, the bandwidth of the input signal is oversampled by a factor of $f_s/(2f_{max}) = 8$. Here the bandwidth of the input signal is divided over the two I and Q signals, resulting in a maximum frequency of 2.5 MHz. Because of this oversampling factor, the quantization noise produced by the finite resolution of the analog-to-digital converter is spread over a larger bandwidth which can be later filtered out and decimated to produce a signal with a higher SNR than would have been produced by merely sampling the input signal at the minimum required frequency. Equation (4) describes the maximum theoretical SNR for an r -bit ADC assuming random quantization noise that is uniformly distributed over one quantization step and a full-scale sinusoidal input [26].

$$SNR = 6.02r + 1.76 + 10 \log \left(\frac{f_s}{2f_{max}} \right) \quad (4)$$

For a 14-bit ADC with $f_s = 40$ MHz and $f_{max} = 2.5$ MHz, the maximum theoretical

SNR is approximately 95 dB, including an oversampling factor of $10 \log(8) \approx 9$ dB. Laboratory measurements of the digital receiver's SNR are given later in the thesis.

2.2 Clock Jitter

In bandpass undersampling systems, the effect of aperture uncertainty quickly becomes one of the limiting factors of the overall signal-to-noise ratio. Any jitter present in the ADC clock will be mixed with the input signal to degrade SNR performance [15, 25]. Furthermore, the effect of clock phase noise is directly related with the input frequency being sampled. The practical effect of clock jitter on the output signal from an ADC depends on the characteristics of the jitter exhibited by the oscillator source. Wideband phase noise present in the oscillator will raise the noise floor of the output, while phase noise near the clock fundamental will spread out a single-tone input signal across adjacent frequency bins [19].

Given the root-mean-square aperture uncertainty t_j , Equation (5) describes the theoretical SNR limit, where f_{in} is the analog input frequency.

$$SNR = -20 \log(2\pi * f_{in} * t_j) \quad (5)$$

Rearranging Equation (5) to solve for t_j results in Equation (6), the maximum rms clock jitter to achieve a specified theoretical SNR.

$$t_j = \frac{10^{\frac{-SNR}{20}}}{2\pi * f_{in}} \quad (6)$$

Figure 3 shows how clock jitter affects the total SNR of an ADC with 72.5 dB SNR for a 52.5 MHz input signal. If $t_j = 1$ ps, the maximum theoretical SNR of the system is reduced to 68.8 dB. With each doubling of the input frequency, the maximum theoretical SNR is decreased by a further 6 dB.

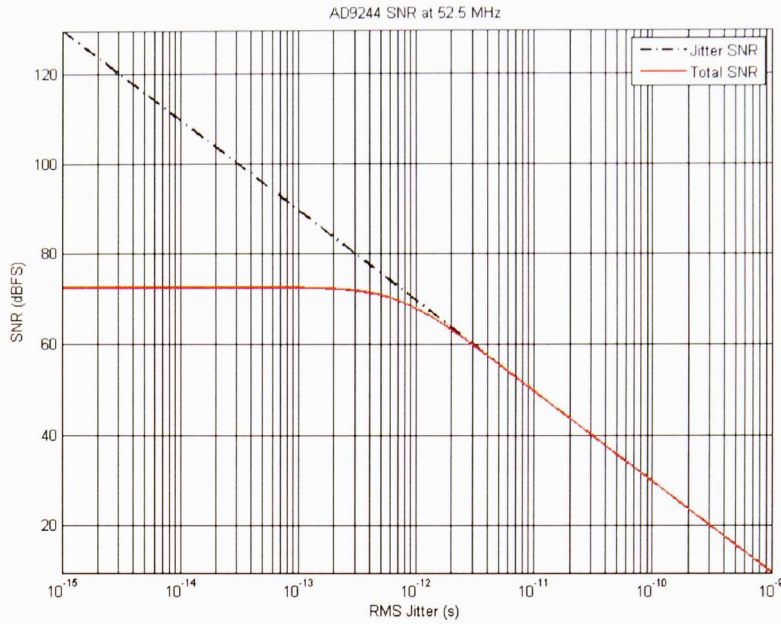


Figure 3: Graph of Theoretical SNR Achieved vs Clock Jitter

2.3 Metastability in Bistable Circuits

All bistable logic circuits such as the flip-flop memory elements present in FPGAs are vulnerable to a condition known as metastability, as shown in Figure 4. This condition manifests as a binary decision which takes an unbounded amount of time to resolve due to input signal collisions [20]. More specifically, in any case where a flip-flop’s setup or hold time is violated, the possibility for a metastable occurrence is created. Setup and hold time violations can occur when digital systems communicate asynchronously, so that the input signal is not synchronized with the sample clock [21]. Flip-flops are also more likely to become metastable when the device is operating at a higher temperature or a lower supply voltage. Due to the lack of correlation between the input signal and sample clock, metastability can only be defined statistically as a mean time between failures (MTBF). The probability that a flip-flop will take τ seconds to recover from a metastable condition is exponentially related to the failure rate and inversely proportional to the gain-bandwidth of the positive feedback system within the flip-flop [22]. Equation (7) defines the accepted method of determining the mean time between failures for a device vulnerable to metastability, where $K1$ is the

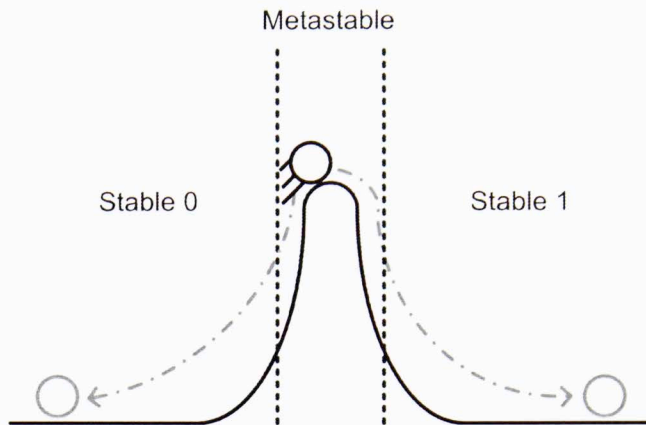


Figure 4: Bistable Device Experiencing a Metastable Condition

length of time during which a metastable condition can occur, K_2 is the metastability recovery speed which is proportional to the gain-bandwidth product of the device, F_1 and F_2 are the clock frequency and the average asynchronous data frequency, and τ is the time allowed for metastable resolution in addition to the propagation delay [23].

$$MTBF = \frac{e^{K_2 * \tau}}{K_1 * F_1 * F_2} \quad (7)$$

This equation assumes that the asynchronous input and the clock signal have no correlation so that the probability of a change on the input is randomly distributed across the period of the clock. To dramatically increase the MTBF, a second-order synchronization circuit such as the one shown in Figure 5 can be built so that the first flip-flop must be metastable for a long enough time that the metastable condition continues just until the setup and hold window on the second flip-flop. The chances of this occurring are much smaller. For example, a flip-flop may have a recovery rate K_2 of 10 GHz, a setup and hold time of 0.36 ns, and a 160 MHz clock, with an asynchronously switching input of 40 MHz. The amount of time allowed for metastability is the timing slack minus the setup and hold time. For 4 ns of timing slack this becomes $4 \text{ ns} - 0.36 \text{ ns} = 3.64 \text{ ns}$. Using Equation (7), the MTBF is then calculated to be approximately 88.5 years. Using an additional flip-flop applies a second factor of $e^{K_2 * \tau}$ [24] which increases the MTBF to $5.69 * 10^{17}$ years.

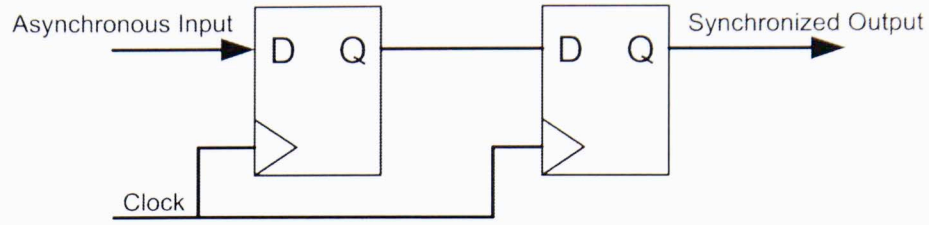


Figure 5: Second-order Synchronizer

Asynchronous inputs commonly occur when a logic element being driven with a clock of frequency f_1 and phase ϕ_1 has its output connected to a logic element being driven with a clock of frequency f_2 and phase ϕ_2 , where f_1 and f_2 are either non-harmonically related or are related, but with a phase offset. In these cases, the phase relationship between the clock domains can potentially cause a metastable event in the second logic element. Using a synchronizer will ensure that the output of the final flip flop is almost never metastable, but the time required for the signal from an unrelated clock domain to propagate through the synchronizer can vary by one clock cycle depending on the precise order of signal arrival. If two synchronizers are constructed and driven with the same asynchronous input, slight differences in path lengths to and within each synchronizer could cause one synchronizer to activate a cycle before the other. Because of this uncertainty, synchronizers should be used only as notifications in a handshaking protocol between clock domains which communicate the current status of the data to be transferred. For example, when transferring a data bus, a flag named *data_valid* could originate from the sending domain to signal when the bus is stable and wait until a flag named *data_ack* is received from the destination domain before changing the bus. One synchronizer is needed for each flag, but the data bus is guaranteed to be stable while *data_valid* is active and *data_ack* is inactive. While such protocols incur overhead cycles to complete a data transfer which may over-constrain the design, the inevitable result will be data corruption when their use is needed but not utilized. In summary, the second-order synchronizer is used in the digital receiver design to ensure high data reliability when crossing clock domains.



CHAPTER 3: SINGLE-CHANNEL RECEIVER DESIGN

This chapter discusses the design of the single-channel digital radar receiver. Section 3.1 details the hardware design of the digital receiver, Section 3.2 covers the firmware and software elements of the design, and finally Section 3.3 includes some experimental results obtained from the digital receiver.

3.1 Hardware Design

The single-channel digital receiver has several design constraints:

- The receiver must digitize a 50 MHz-centered IF with a bandwidth of 5 MHz.
- The receiver must accept an 80 MHz reference clock, from which all other clocks that drive synchronous data generation logic must be derived.
- The receiver must accept a TTL-level (0 to 5 V) system trigger.
- All components used in the design must be hand-solderable. No ball-grid array packages, no-lead packages, or packages with small lead spacings can be placed on the PCB. Resistor and capacitor packages smaller than 0805 (approximately 80 mils by 50 mils) cannot be used.
- The output connector must be a SCSI-type VHDCI connector in order to interface with the host computer's digital I/O card.
- The final PCB design must be as compact and lightweight as possible.
- To maintain a low fabrication cost, the PCB design should require no more than four total layers.
- The host computer must be able to store all data produced by the PCB to the disk storage system in real time, after being notified of a system trigger event.

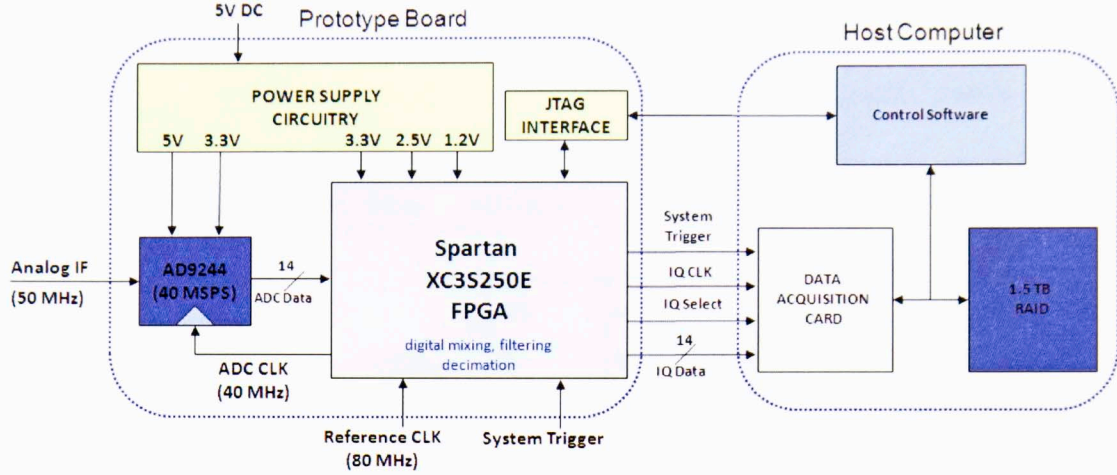


Figure 6: Single-Channel Digital Receiver Interface Diagram

The digital receiver design consists of two major subcomponents, as shown in the interface diagram in Figure 6. These components and other hardware aspects of the design will be discussed in detail in the following subsections.

3.1.1 Analog-to-Digital Converter

The first component is an Analog Devices AD9244 ADC. This ADC typically uses only 350 mW of power. The AD9244 also exhibits excellent bandpass sampling characteristics at the intermediate frequency of 50 MHz: 73 dBc SINAD (11.9 ENOB), 73.5 dBc SNR, and 86 dBc SFDR. The maximum input signal able to be digitized is $2 V_{pp}$, which corresponds to an input power of 10 dBm when loaded with 50Ω , through Equation (8). In order to maintain the specified SNR, the rms clock jitter must be less than 673 fs according to Equation (6).

$$P(\text{dBm}) = 10 \log \left[\frac{V_{rms}^2}{50 \Omega} \right] \quad (8)$$

The signal input to the ADC was chosen to provide the best noise performance possible while providing electrical isolation and AC-coupling through the 1:1 transformer. Figure 7 shows how the input signal is transformed from a single-ended to a differential

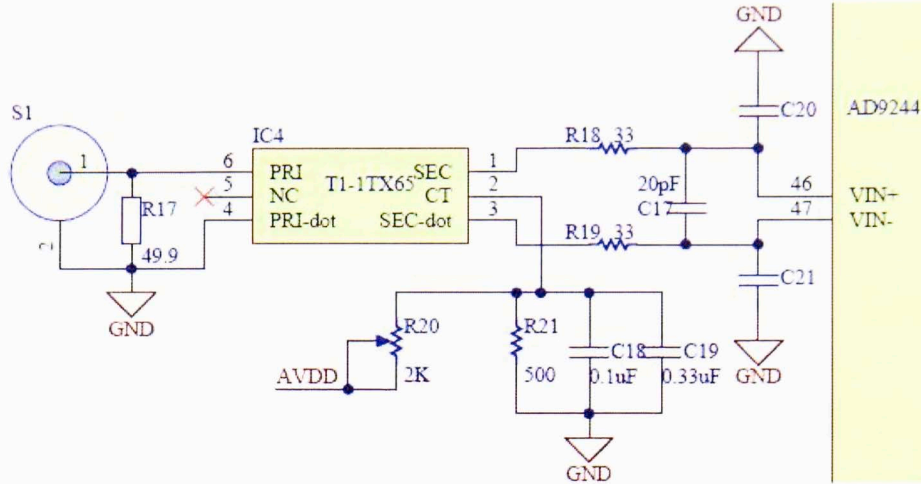


Figure 7: Single-Channel Digital Receiver ADC Differential Input Connections

trace with a nominal common-mode voltage of 2.5 V. The differential mode of signal distribution provides excellent power supply isolation, common mode noise rejection, and crosstalk immunity due to the inherent voltage reference and signal return path [28].

The digital output interface of the AD9244 is a parallel bus, as shown in Figure 8. The logic level depends on the digital supply voltage, which can be between 2.7 V and 5.25 V. The output bus includes an out-of-range OTR signal which is asserted when the input signal exceeds the ADC conversion window of $2 V_{pp}$. It is important to know when the conversion window has been exceeded because the out-of-range recovery time to re-acquire the analog input signal is 1-2 clock cycles, during which time the ADC's output is invalid.

3.1.2 Field-Programmable Gate Array

The second component is a Xilinx Spartan 3E XC3S250E FPGA. This FPGA holds all of the digital processing logic and controls all the other circuitry on the printed circuit board. The XC3S250E is logic-optimized with 250,000 system gates and was chosen because it has the highest logic density available for a hand-solderable package. The FPGA is available in a 100-pin, 144-pin, or 208-pin quad flat-pack (QFP). The 100-pin package can be seen in Figure 9 after being soldered on a printed circuit board.

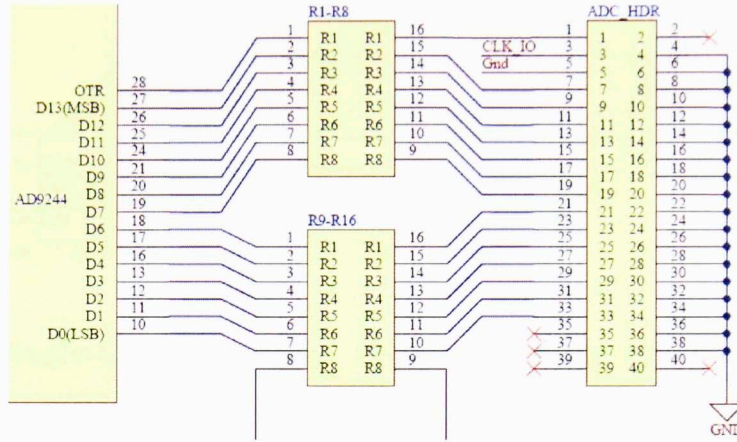


Figure 8: Single-Channel Digital Receiver ADC Parallel Output Connections

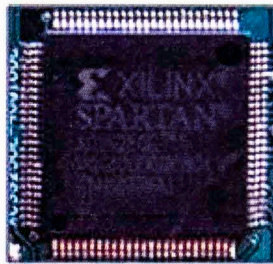


Figure 9: Hand-soldered 100-pin FPGA Package on a Prototype Printed Circuit Board

The higher pin count packages have more I/O pins available for designs that require many connections outside of the FPGA. The QFP package also facilitates using fewer layers than would be required for a package with a higher pin count or a non-hand-solderable package such as a ball-grid array. These denser packages require more layers to completely route all needed pins.

Figure 10 details the number and ordering of layers and the spacing between layers for the single-channel receiver design. Signals between components are routed on the two outer layers while the inner layers are used for low-impedance power and ground planes.

Due to the need for isolated analog and digital grounds as well as three different supply voltages, these planes must be split in order to reach all the connections that require each supply. Figure 11 shows how this split is accomplished for the three power supplies. The FPGA requires 2.5 V internally as well as 3.3 V for the input and output

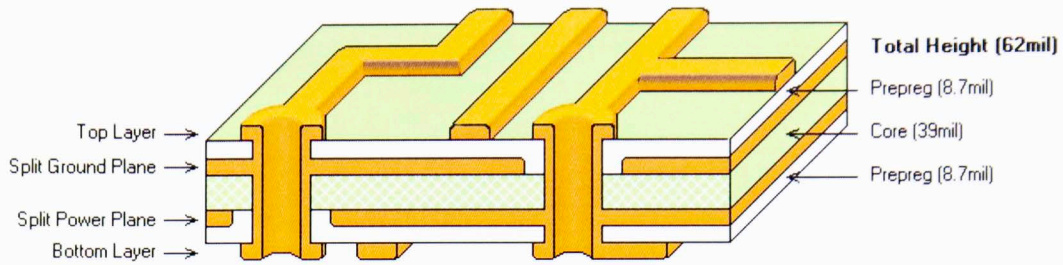


Figure 10: Single-Channel Digital Receiver Printed Circuit Board Layer Stack-up

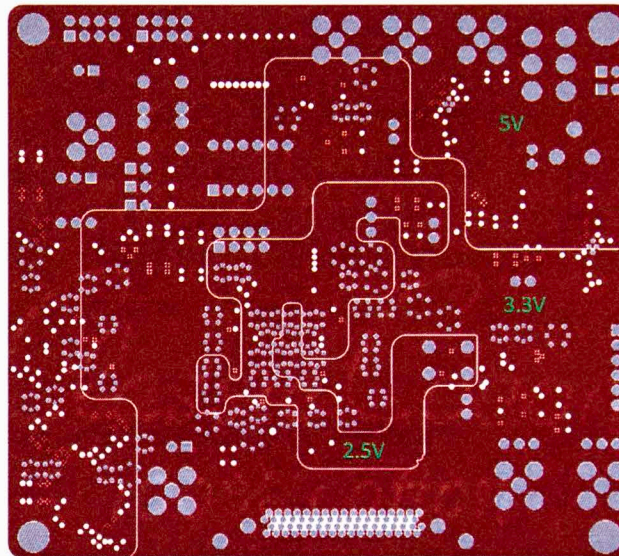


Figure 11: Single-Channel Digital Receiver Printed Circuit Board Split Power Plane Layer

buffer drivers to communicate with the surrounding logic. 5 V is used for the ADC's analog supply voltage as well as the voltage regulators which produce the 3.3 V and 2.5 V voltage sources.

The FPGA also features 12 integrated multipliers along with 216 kbit of block RAM and 38 kbit of distributed RAM. These multipliers are suitable for implementing high-speed digital filters which can operate near the maximum clock speed of the device because of their pre-optimized hardware structure. Building a multiplier out of the distributed logic of the FPGA would result in a lower maximum clock speed due to the multiplier being translated into the general logic of the FPGA, rather than custom-designed multiplier hardware and should therefore be avoided. The firmware

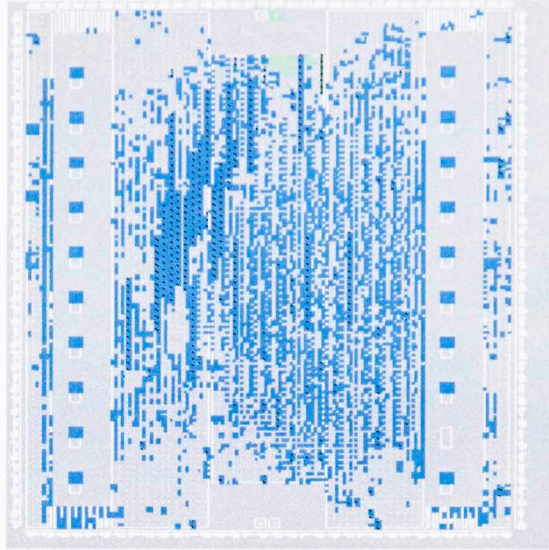


Figure 12: Single-Channel Digital Receiver FPGA Floorplan Utilization

logic utilization for the XC3S250E is shown in Figure 12. The light blue squares represent a configurable logic block (CLB), hardware multiplier, or block RAM being used in the design. Logic placement within the FPGA is controlled by the Xilinx software through user-programmed timing constraints.

Clock distribution is handled by the FPGA, where the 80 MHz reference clock is used to synchronize the internal logic. The 40 MHz sampling clock is also generated from the reference clock using a digital clock manager (DCM) inside the FPGA in order to maintain the coherency of the entire receiver.

The firmware for the FPGA may be loaded from a JTAG interface after power-up or stored more permanently in an on-board PROM chip. The PROM flash used is a 4 Mb Xilinx Platform Flash XCF04S in a thin shrink small outline package. The XC3S250E requires 1,322 kb of memory for each firmware image, so three complete firmware images could be stored in the XCF04S to reconfigure the FPGA as desired.

3.1.3 Host Computer

Once the input signal has been digitized, mixed into in-phase and quadrature components, and filtered, the data is transmitted from the custom printed circuit board to a host computer which contains a National Instruments PCIe-6537 digital I/O card

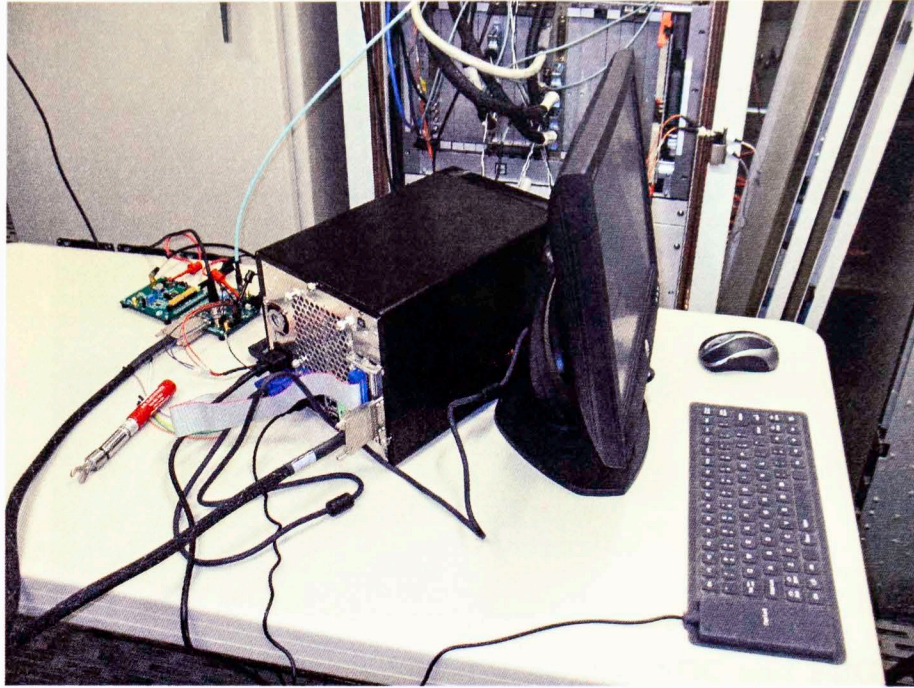


Figure 13: Single-Channel Digital Receiver Prototype Setup During Field Testing

to acquire the data. The host computer is a small form factor computer containing a RAID hard drive array to store the processed data from the digital receiver. As can be seen in Figure 13, the small size of the computer and digital receiver make the entire system extremely portable and can be transported and set up in minutes.

3.1.4 Hardware Development

The single-channel digital receiver proceeded through several stages of development before completion to aid debugging and to evaluate the effect of design changes with each iteration. Hardware testing began with two separate evaluation modules for the analog-to-digital converter and FPGA. The design was broken down into two submodules, one for the ADC and its associated circuitry and one for the FPGA and its associated circuitry. Figure 14 shows the fabricated submodules. Each of these submodules is backwards-compatible with the evaluation boards so they may be interchanged freely to test design changes.

Following the verification of functionality for the two submodules, the design was

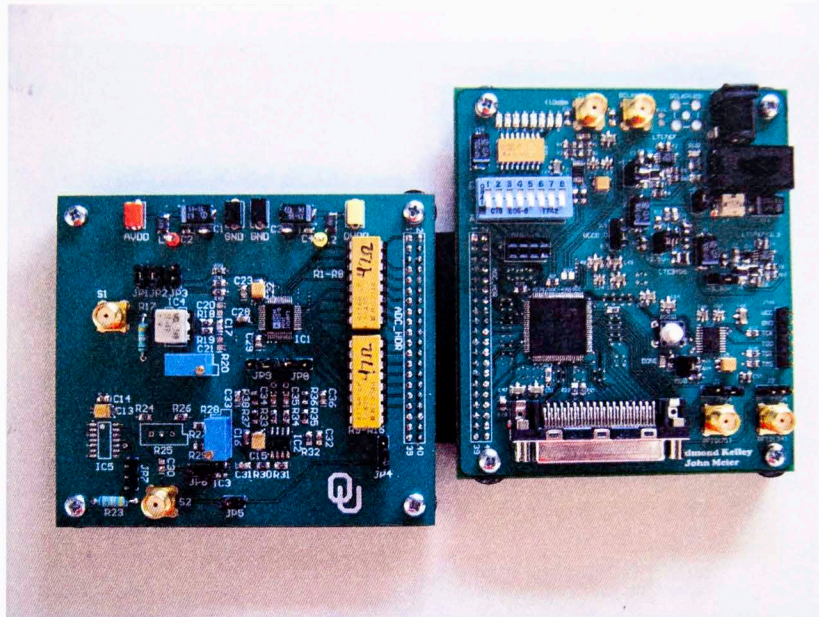


Figure 14: Fabricated and Assembled Separable ADC and FPGA Digital Receiver Submodules

quickly combined into a single PCB. Figure 15 shows the final circuit board layout for the combined digital receiver. The submodules functioned as expected so that minimal changes had to be made to the final design, accelerating the integration phase of the project. In total, combining the two submodules into one design required only one week of preparation until the layout was ready to be fabricated following the careful and diligent design work done for each submodule. The final single-channel fabricated and assembled design is shown in Figure 16.

3.2 Software Design

This section will discuss the method of operation for both the firmware loaded onto the FPGA and the software running on the host computer.

3.2.1 FPGA Firmware Structure

The method of bandpass sampling was discussed in Section 2.1, and is the primary function of the FPGA. Figure 17 shows the signal processing flow as the digitized IF signal proceeds through the FPGA onto the host computer for storage.

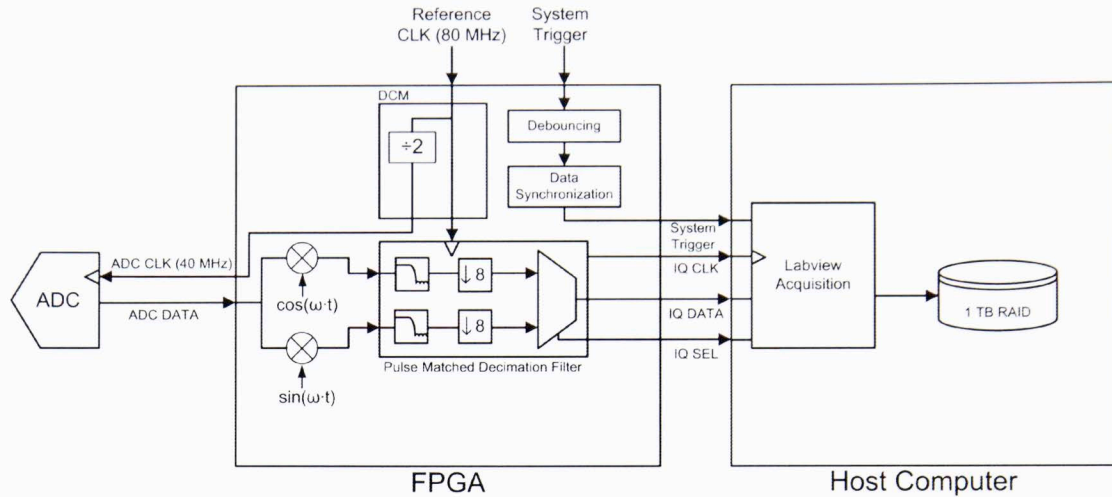


Figure 17: Functional Block Diagram of the Single-Channel Digital Receiver FPGA Firmware

Directly after digital conversion, the parallel bus output word from the ADC is multiplied by digital sine and cosine signals at 10 MHz. The sampling rate of 40 MHz allows these sine and cosine signals to be simplified into sequences of $\{0, 1, 0, -1\}$ and $\{1, 0, -1, 0\}$, respectively, so that digital multiplication is not required at all. Instead, as Figure 18 shows, a four-to-one multiplexer with a two's complement operation to invert the input data word on one of the inputs is all that is required. Furthermore, the two's complement operation can be activated at one-fourth of the input data rate, or 10 MHz, further enhancing power consumption and logic timing.

After the I and Q signals have been generated, they are both processed by a single low-pass filter to remove the unwanted spectrum image centered at 20 MHz. This low-pass filter also serves the second purpose of pulse-match filtering the input signal so that the signal-to-noise ratio is maximized. Briefly, if a signal is degraded with only Gaussian white noise, a filter with a frequency response that is the complex conjugate of the transmitted radar pulse spectrum is the most efficient at discriminating between the noise and desired echoes [29]. In the case of this digital receiver design, the input spectrum is degraded not only with Gaussian white noise, but also with the unwanted signal image centered at 20 MHz. A compromise must be found between filtering the unwanted image and preserving the transmitted pulse spectrum's energy in order to

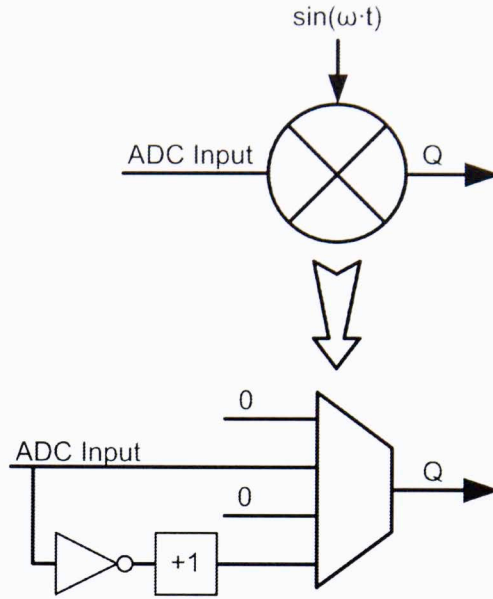


Figure 18: Bandpass Sampling Method of Simplification for I/Q Signal Generation

maintain the highest signal-to-noise ratio possible. Finding the optimum filter response for this purpose, however, is left for future work. The digital receiver's filter coefficients may easily be changed by loading a different firmware image that has been synthesized with the desired coefficients, so that when a more optimum filter response is found it will be straightforward to integrate the response into the digital receiver design.

Figure 19 shows a low-pass filter response that was used to collect the test results shown in Section 3.3. The filter's cutoff frequency was chosen to be the equivalent frequency of the pulse length. For the phased array radar at the NWRT, the pulse length is $1.57 \mu\text{s}$ which corresponds with a frequency of approximately 637 kHz. Because the filter decimates the sampling rate by eight down to 5 MHz, signals with a frequency greater than $f_s/2 = 2.5 \text{ MHz}$ or normalized frequency of 0.125 must also be removed or they will become aliased back into the passband after decimation. This includes the excess quantization noise distributed throughout the oversampled portion of the spectrum. The stop-band attenuation was chosen to be the dynamic range of the digital receiver, which is 74 dB, ensuring that any signals in the stop band will be attenuated below the noise floor.

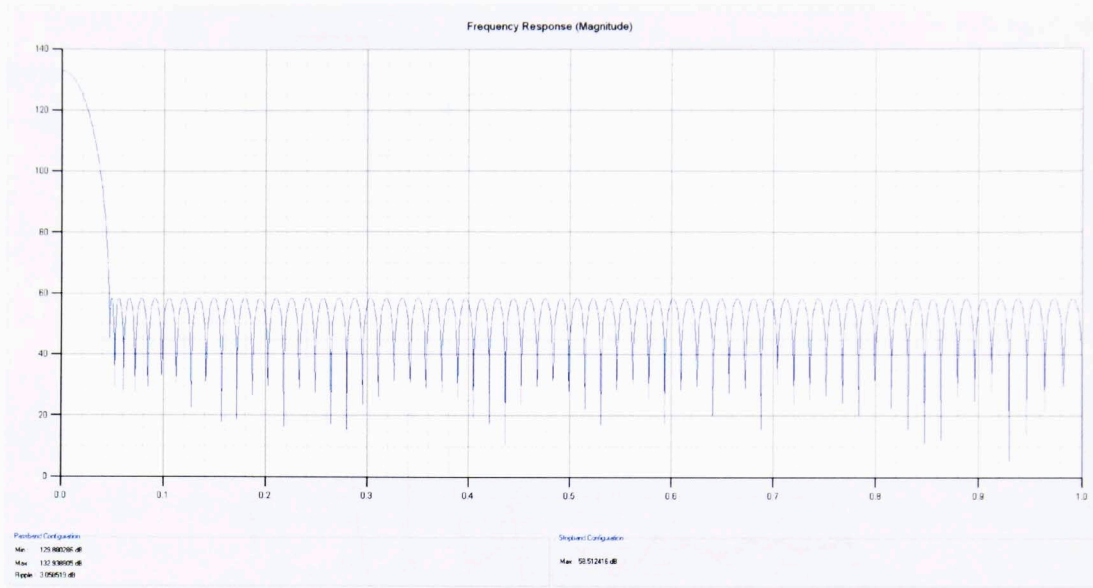


Figure 19: 128-tap Low-Pass Decimation Filter Response

The system trigger, which is asserted at the beginning of a new transmit pulse and signals when a new radial of data begins, is also passed through the digital receiver. This is done for two reasons. Figure 20 shows the long-scale time-domain activity of the system trigger at the NWRT. It can be seen that the system trigger initially has an extremely sharp edge followed by a slow capacitive charge time. Figure 21 shows a higher-resolution trace of the falling edge of the system trigger. The signal has so much ringing that it overshoots the starting quiescent high voltage, and contains several smaller peaks which could cause false system trigger activations to be sent. For this reason, the system trigger must be de-bounced before transmission to the host computer. Based on this specific system trigger, a recovery time of $5 \mu\text{s}$ or longer should ensure correct triggering functionality.

A second reason to route the system trigger through the digital receiver instead of directly to the host computer is to account for the sample delay of the digital receiver itself. There exists a pipeline delay for the logic of the ADC and FPGA to process the data samples. For the AD9244, the pipeline delay is fixed at eight clock cycles. The pipeline delay of the internal registers in the FPGA varies with the precise structure, which includes the delay introduced by the decimation filter to process the data. The

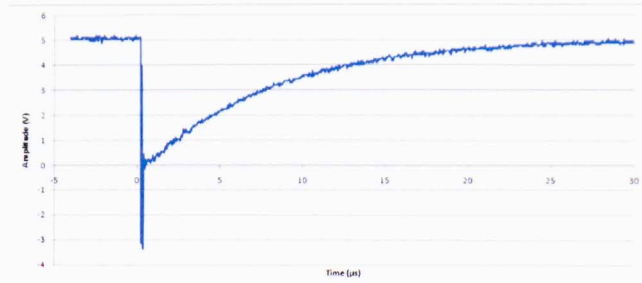


Figure 20: Falling Edge of the System Trigger at the NWRT

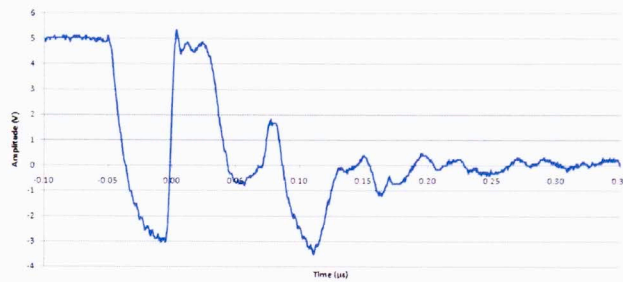


Figure 21: Zoomed-in View of Figure 20

sampling rate decimation that the filter performs must be taken into account for the final total delay. The single-channel digital receiver exhibits a total pipeline delay of 26 cycles. Once this delay has been established, it will not change unless the number of registers in the data path has been altered.

A timing diagram is shown in Figure 22 to explain the relative signal transition scheme for the input and output signals of the FPGA. The system trigger's analog waveform has been drawn to equivalent scale to show the internal de-bouncing signal's response. It can be seen that the reference clock of 80 MHz is the same clock that is used for the low-pass filter. The filter must be clocked at 80 MHz so that both the I and Q samples can be processed in one 40 MHz clock cycle. The 40 MHz ADC sampling clock is phase-aligned with the 80 MHz clock, and each data sample is updated between 3.5 ns and 7 ns after the rising edge.

After filtering and decimation, the baseband I and Q data words are output from the FPGA along with a 10 MHz IQ sample clock and an IQ select signal. The IQ select signal determines whether the current word is an I or Q sample. From here the data is captured by the digital I/O card on the host computer.

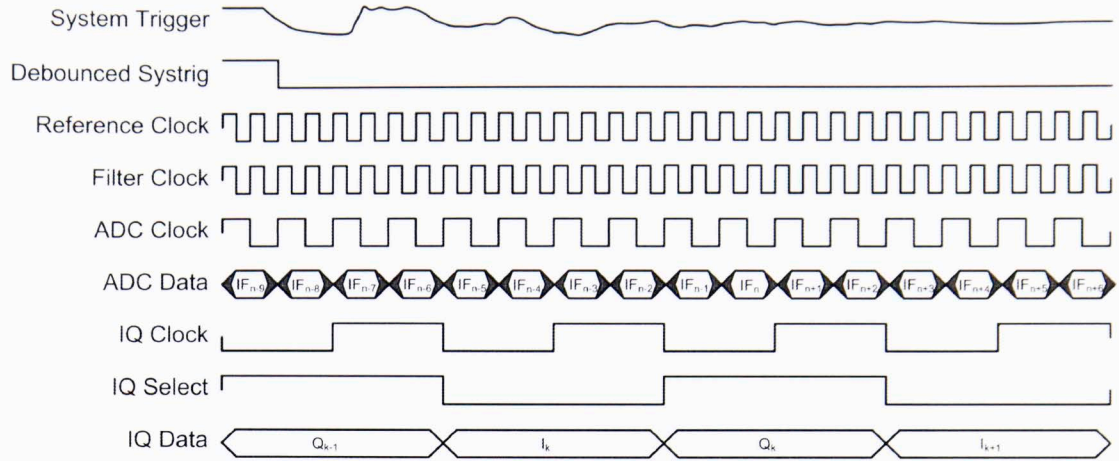


Figure 22: Single-Channel Digital Receiver Input and Output Interface Timing Diagram

3.2.2 Host Computer Software Operations

The host computer uses a Labview PCIe-6537 digital I/O card to capture the data coming from the digital receiver. This digital I/O card is capable of receiving a bus of 32 data bits clocked at up to 50 MHz along with five programmable function interface (PFI) signals, which can be used for clock and trigger signals, through a connecting 1 m long shielded VHDCI cable. A Labview virtual instrument (VI) program running on the computer controls the method of acquisition and any further data processing. Figure 23 shows a Labview VI that is used to display the data from the digital receiver and compute a single and double-sided FFT from the I and Q data and also to find the mean equivalent radial velocity for the spectrum. The radar data shown here are simulated in the lab by a waveform generator so that test conditions can be carefully controlled.

In order to properly receive data from the digital I/O card at the transmission rate, multi-threading must be employed in the Labview program. The graphical Labview code in Figure 24 shows a two loop structure. The first loop takes in data from the PCIe-6537 and enqueues the data in a FIFO buffer in RAM memory. The second loop dequeues the data and processes it for display and data storage. By decoupling these two operations, the loops can work in parallel without causing a stall when one task must wait upon another's work to be completed. Using this producer-consumer loop

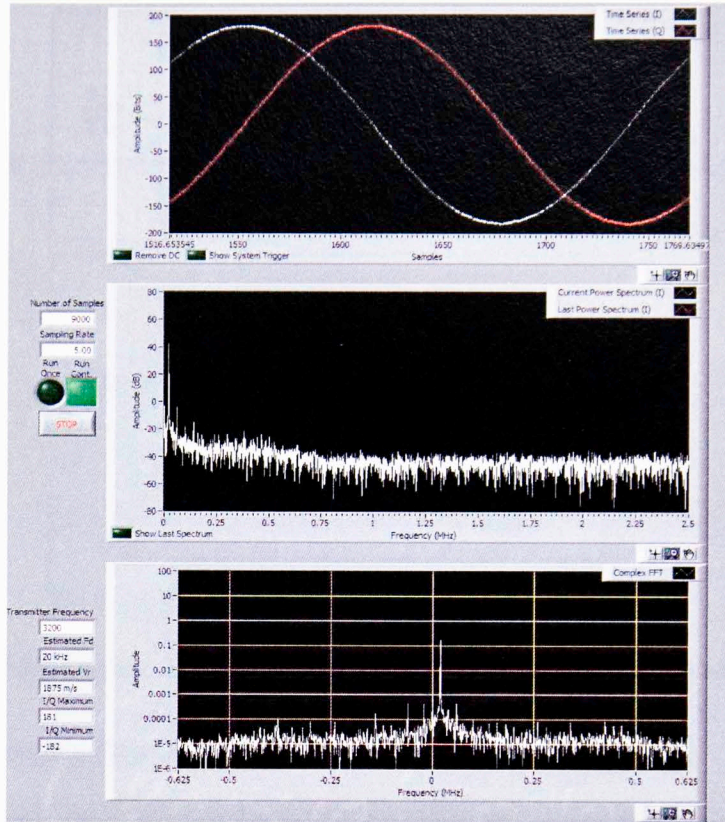


Figure 23: Labview VI Capturing Data Processed by the Digital Receiver

architecture allows the host computer to multi-task the read, processing, and write operations to store the full 160 Mbit/sec bandwidth of the digital receiver. The 1.5 TB RAID array on the host computer can then hold more than twenty hours of continuous data acquisition for later processing.

3.3 Experimental Results

Many experimental tests were performed to verify correct functionality of the hardware at each stage. The testing is divided into two parts: testing done inside the laboratory with controlled signal conditions and testing completed in the field under signal conditions which would be expected during normal use of the digital receiver.

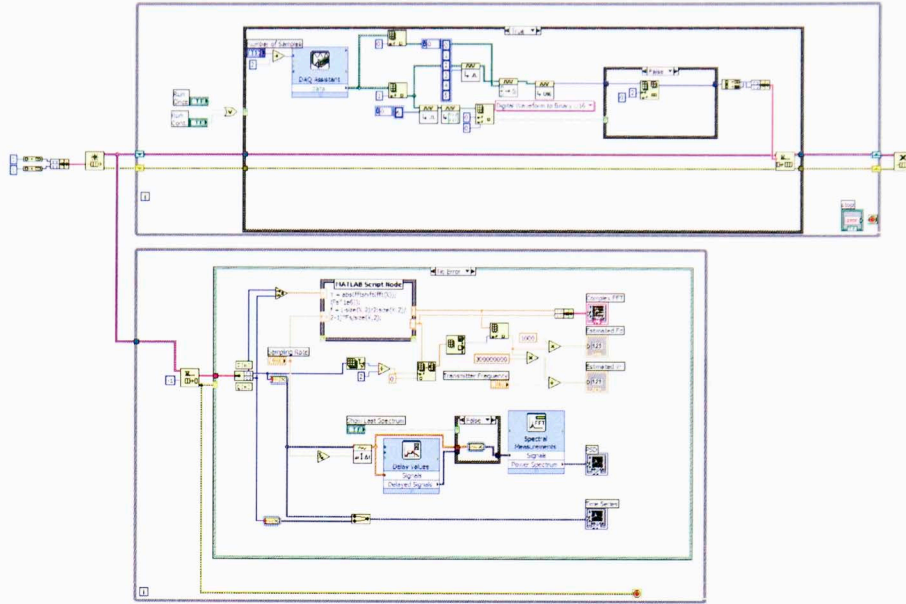


Figure 24: Graphical Labview Programming Code

3.3.1 Laboratory Testing

Testing completed in the laboratory under controlled conditions allows certain characteristics of the digital receiver to be found. The maximum and minimum input signal amplitude as well as the strength of the input signal amplitude relative to the amplitude of any distortion products was measured. The amount of power consumed during normal operation was also measured.

Signal characteristics of the single-channel receiver were found by applying an input signal and observing the digital output words either as time-series data or processed data after the application of a fast fourier transform (FFT). The maximum input signal amplitude that can be digitized is +11.4 dBm. This is higher than +10 dBm because of connector losses and losses inside the gain stage of the ADC. The lowest signal amplitude which can be digitized is approximately -65 dBm. Figure 25 shows the output spectrum when a +10 dBm 50.1 MHz input signal is applied. With a signal amplitude of 73.3 dBFS and a noise amplitude of 11.4 dBFS, the SNR at 50.1 MHz is 61.8 dB.

The current draw of each voltage supply was measured during normal operation with a reference clock and analog input signal applied. The results are shown in

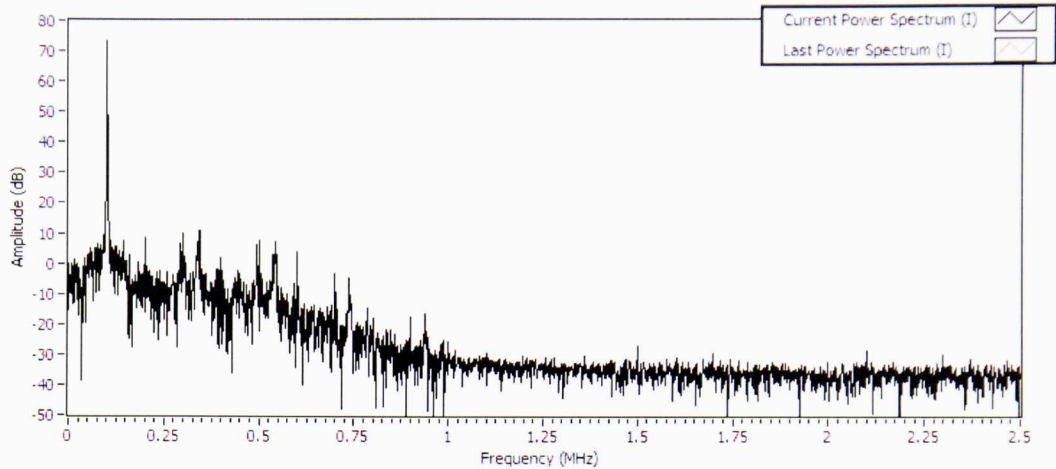


Figure 25: FFT of Digital Output with +10 dBm 50.1 MHz Input Signal

Device	Supply	Current
FPGA	1.2 V	27.5 mA
	2.5 V	24.5 mA
	3.3 V	95 mA
ADC	3.3 V	21 mA
	5 V	208 mA

Table 1: Single-Channel Receiver Power Consumption Characteristics

Table 1. The power consumption of the FPGA and related circuitry was found to be 408 mW, while the power consumption of the ADC and associated circuitry was 1.11 W. Including the losses of the voltage regulators by measuring the current draw at the input voltage, the total power consumption of the digital receiver was measured to be $5 \text{ V} * 357 \text{ mA} = 1.785 \text{ W}$.

3.3.2 Field Testing

Field testing of the single-channel digital receiver was performed at the phased array radar (PAR) at the NWRT, which can be seen in Figure 26. This environment proved to be an excellent testing ground for the digital receiver. The NWRT provides ready access to all necessary signals for the digital receiver, and it also serves as a convenient platform with which to sense weather phenomena in and around Norman, OK. Furthermore, all data collected at the NWRT can be compared with the data collected



Figure 26: Phased Array Radar at the National Weather Radar Testbed

Frequency	3200 MHz
Tx Polarization	Vertical
Maximum Peak Power	750 kW
Maximum Average Power	1.18 kW
Beamwidth	1.5° at broadside (2.1° at ±45°)
PRT	Variable, nominal 800 μs
Pulse length (short)	1.57 μs
Pulse length (long)	4.71 μs
WX Sensitivity	0 dB SNR is 5.89 dBZ at 50 km

Table 2: Selected National Weather Radar Testbed Specifications [30]

by the phased array radar’s own digital receiver using the same down-conversion chain. An overview of the capabilities of the NWRT system has been included in Table 2.

The single-channel digital receiver interfaces easily with the signals at the NWRT partially because it was designated as the primary test platform from early in the research program. Table 3 lists the interface signals used to interface the digital receiver to the NWRT so that data from the radar could be recorded. The 80 MHz reference clock at the NWRT is generated from a highly stable 10 MHz reference source which serves as the coherent oscillator for the entire radar system. Using this reference

NWRT Signal	Frequency	Amplitude	Cable Requirements
Reference clock	80 MHz	0 dBm sine	2 m low-loss SMA
IF weather signal	50 MHz	7 dBm maximum	2 m low-loss SMA
System trigger	1-2 kHz	TTL (0 to 5 Volts)	5 m low-loss BNC

Table 3: Interface Signal Characteristics and Requirements at the National Weather Radar Testbed

clock allows the digital receiver to become a coherent digital receiver. The maximum amplitude of the IF signal is 7 dBm, which is beneath the 10 dBm conversion window of the ADC sampling range. This means that digital conversion should be possible for all ranges and azimuths. This IF signal is the same signal normally attached to the NWRT digital receiver. To conduct a test, the NWRT receiver must be temporarily disconnected so that the single-channel receiver may use the signal.

In order to test how well the digital receiver acquired typical radar signals rather than simulated laboratory signals, a field test of the digital receiver was conducted at the NWRT on April 8, 2008. The type of scan performed was a single 90° sector scan at an elevation angle of 0.5054° with 16 pulses per azimuth angle. The results are shown in Figure 27. Weather conditions on April 8 were overcast becoming clear during time of the test. The returned echoes were composed of nearly all ground clutter from radio towers north of Norman, OK. Data taken from the NWRT's own digital receiver was also acquired for comparison, as shown in Figure 28. The data shown for the NWRT receiver were recorded on March 10, 2008 during clear conditions.

While the data were not recorded on the same day, the weather conditions are similar enough for comparison purposes because stationary clutter constituted the primary return signal. These plots were generated from raw I and Q data returned from both receivers in the same way. The relative amplitude of the I and Q signals in the returned datasets were different, so a calibration factor was estimated for the digital receiver in order to align the two datasets for comparison. Between the two figures, the NWRT receiver data appears to be more noisy because this data has not been processed with a matched filter, where the data from the receiver under test has been passed through the filter described in Section 3.2.1. Other than these differences, the data appears to

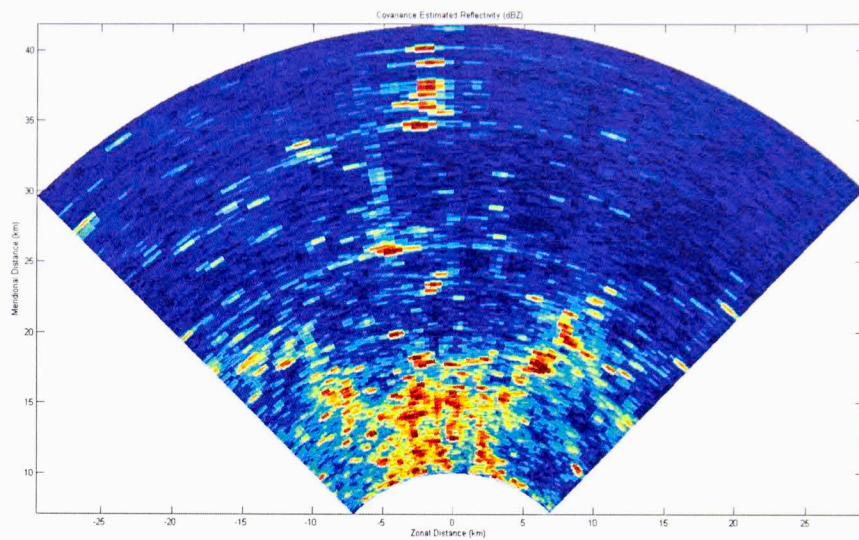


Figure 27: Single-Channel Digital Receiver Clutter Test Data

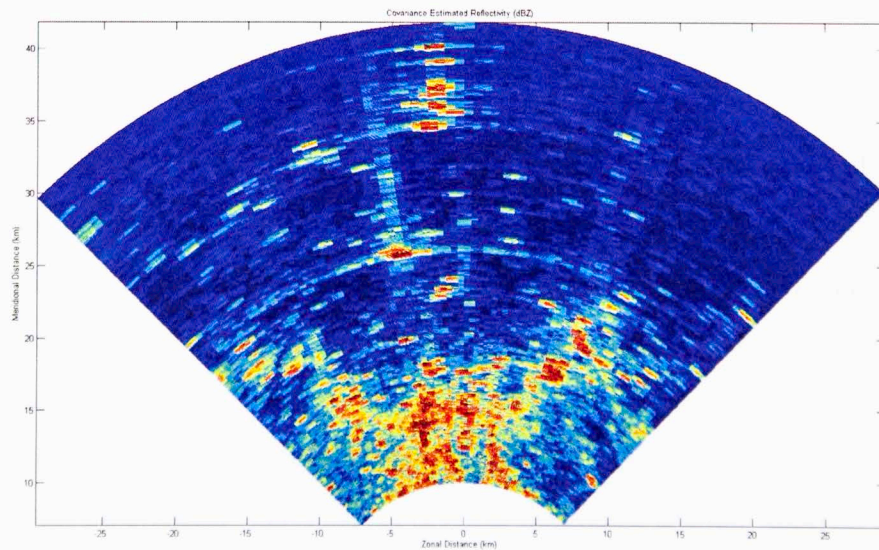


Figure 28: NWRT Receiver Clutter Test Data

be identical, confirming the correct operation of the digital receiver for narrow-band weather radar data near the IF frequency.

Following the clutter data collection success, the digital receiver would need to be tested with actual weather data. This weather data would show how the digital receiver acquires data with a larger bandwidth and also signals most similar to those which would be encountered under the intended operational conditions. A test was conducted during a storm on October 22, 2008 at 19:13 UTC. The storm observed was a squall line approximately 100-130 km to the east of the radar. Squall lines such as this one are severe thunderstorms which can form near cold fronts, bringing heavy precipitation, strong winds, and lightning. A VCP-12-type scan was performed with azimuth-interleaved pulses and two pulse repetition rates. The method of azimuth interleaving took the form of dividing the 90 degree sector into five segments of 18 azimuths. These 18 azimuths, when numbered from left to right beginning at zero, were illuminated with the number of pulses per azimuth in the order {9, 0, 10, 1, 11, 2, 12, 3, 13, 4, 14, 5, 15, 6, 16, 7, 17, 8} before moving to the next segment. Two PRTs were used, a long PRT of 3,104 μs and a short PRT of 896 μs . This gives unambiguous ranges $r_a = cT/2$ of 465.28 km and 134.3 km and unambiguous velocities $v_a = \pm\lambda/4T$ of approximately ± 7.5 m/s and ± 26 m/s. The processed reflectivity of the I and Q data from the longer 3,104 μs PRT is shown in Figure 29.

Comparison data of the same squall line was obtained from the National Climactic Data Center. The WSR-88D weather radar KTLX recorded a scan on October 22, 2008 at 19:19 UTC. KTLX is approximately 20 km northeast of the NWRT. The obtained level-II fully processed data can be seen in Figure 30. The KTLX dataset is in super resolution format with 0.5° angular resolution and 250 m range resolution.

Comparing the digital receiver reflectivity data with the weather data from KTLX reveals several differences. The most noticeable difference is what appears to be several contiguous azimuths missing from the digital receiver data. This is the result of the NWRT attenuating transmitted pulses so that the nearby operational WSR-88D weather radar KOUN does not become saturated. The declining signal to noise ratio

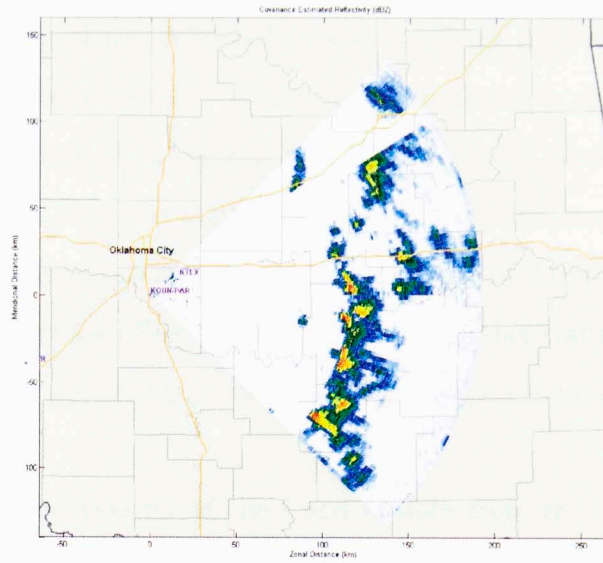


Figure 29: Single-Channel Digital Receiver Reflectivity Test Data

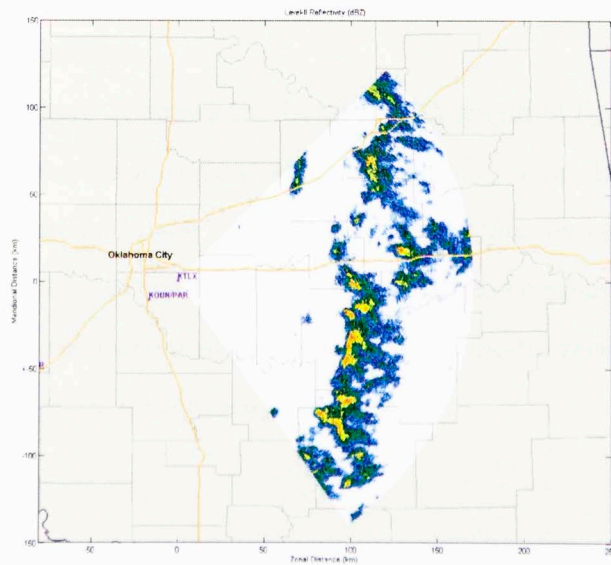


Figure 30: KTLX Level-II Reflectivity Data

can be seen as the clear weather background becomes speckled at the furthest ranges. This signal degradation is not nearly as detectable for data from KTLX because of the much higher sensitivity of the WSR-88D platform. Some differences can be seen for individual storm cells between each dataset, resulting from the slightly different angle the radar pulse takes through the storm. KTLX used a 0.57° elevation angle while the NWRT used a 0.505° elevation angle. The 20 km difference in geographic positions between the radar observation sites would also contribute to the change in precise cloud volume measured. The time offset of the observations further contributes to this effect, with some storm cells appearing to move several kilometers in the intervening time between observations.

The processed radial velocity of the I and Q data from the $896 \mu\text{s}$ PRT scan as computed from the covariance method is shown in Figure 31. With only minimal post-processing used, the velocity test dataset has many artifacts. Second-trip echoes can clearly be seen, with some echoes mixed in with first-trip echo returns. The squall line's radial velocity is also mostly outside the unambiguous velocity range of $\pm 26 \text{ m/s}$, causing aliasing to occur throughout the scan volume. Areas of low reflectivity show noisy estimated velocity measurements, as expected for the covariance method of velocity estimation.

For comparison, Figure 32 shows the Level-II data acquired by KTLX. Similarities can be seen in the far southeast region, where radial velocity reaches zero and reverses towards the radar slightly. In the case of this storm which exceeds both the unambiguous range and velocity of the NWRT, it is difficult to compare these two datasets without the benefit of extensive post-processing on the digital receiver data to remove the artifacts of folded range and velocity information.

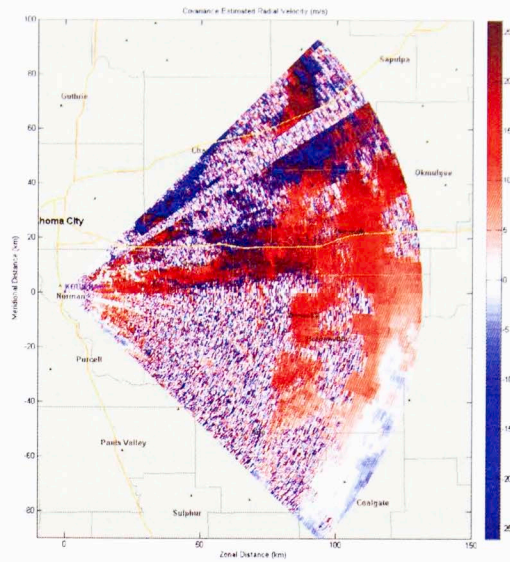


Figure 31: Single-Channel Digital Receiver Radial Velocity Test Data

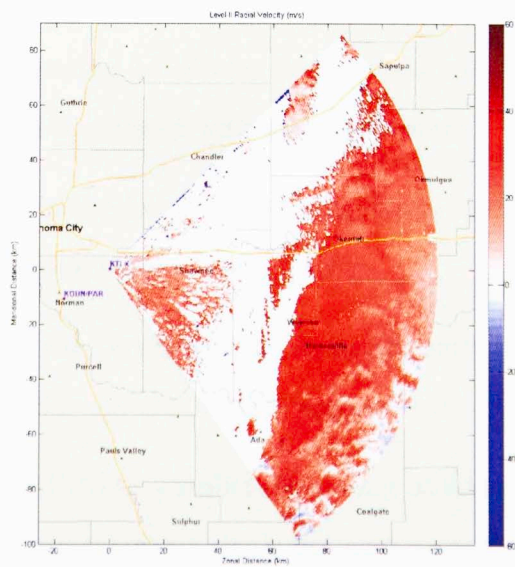


Figure 32: KTLX Level-II Radial Velocity Data

CHAPTER 4: EIGHT-CHANNEL RECEIVER DESIGN

This chapter discusses the design of the eight-channel digital radar receiver. The eight-channel digital receiver is based upon the design of the single-channel digital receiver while greatly expanding the functionality of the design by increasing the number of channels and hence overall complexity of the design, adding new peripheral and communication systems, and focusing on creating a more flexible and usable digital receiver. Despite the increased complexity, centralizing the design will decrease the aggregate cost per channel both for the receiver itself and the downstream processing computer over an implementation which uses multiple single-channel radar receiver units. This creates a much more cost-efficient receiver for radars which utilize more than one channel. Section 4.1 details the hardware design of the digital receiver while Section 4.2 covers the firmware and software elements of the design.

4.1 Hardware Design

Combining the functionality of eight one-channel digital receivers into a single unit has many motivations:

- Space is greatly conserved by placing connections and components much closer together than would be allowed with disparate units.
- Hardware overhead in the form of power regulation circuitry, peripheral components, and connectors is reduced considerably.
- Power consumption can be reduced by more efficiently utilizing a single component rather than under-utilizing several components.
- Complete use of an FPGAs parallel processing abilities can be made by time-multiplexing many signals into a single firmware instantiation.
- Communication bandwidth can be aggregated so that a single command may be sent where eight would be needed and communications overhead can be reduced by combining data packets and simplifying the protocol.

- Downstream processors can be combined more easily due to the fewer, higher-bandwidth communications links from the digital receivers.
- Using fewer, more powerful components can place the design in a more favorable technological position where a large increase in component complexity can be obtained for a relatively small increase in price and power consumption.

Some problems, however, must be carefully avoided which would nullify the above advantages. The increase in complexity may push the design to the technological limit of the components, where only marginal functionality can be achieved. For example, the analog-to-digital converter in this design outputs the digital words over eight serial links at 560 Mbit/s. If the I/O buffers in the FPGA cannot adequately sample the ADC data at a high enough clock rate, the data will become corrupted. A component may also be difficult to acquire due to the highly specific requirements of the design or it may be much more expensive. Designs based on such rare components will inevitably require extensive redesign when new technology becomes available and the older parts become obsolete. Finally, the design may need very small tolerances such as trace width or via size which may not be able to be fabricated without a highly specialized, expensive printed circuit board manufacturer. Only by paying special attention to the design and manufacturability constraints of all the system's components can these problems be averted. The interface diagram for the eight-channel digital receiver is shown in Figure 33. The individual components of this design, their interconnections, and their method of operation will be discussed in the following sections.

4.1.1 Analog-to-Digital Converter

The chosen analog-to-digital converter for this design is an Analog Devices AD9252 simultaneously sampling 14-bit resolution 50 MHz encode rate eight-channel ADC, shown in Figure 34. This ADC exhibits similar specifications with a 50 MHz input signal when compared with to the AD9244: 71.25 dBc SINAD (11.65 ENOB), 72 dBc SNR, and 81 dBc SFDR. The analog conversion window also remains the same at $2 V_{pp}$ or 10 dBm in a 50 Ω system. In order to maintain the specified SNR, the rms clock

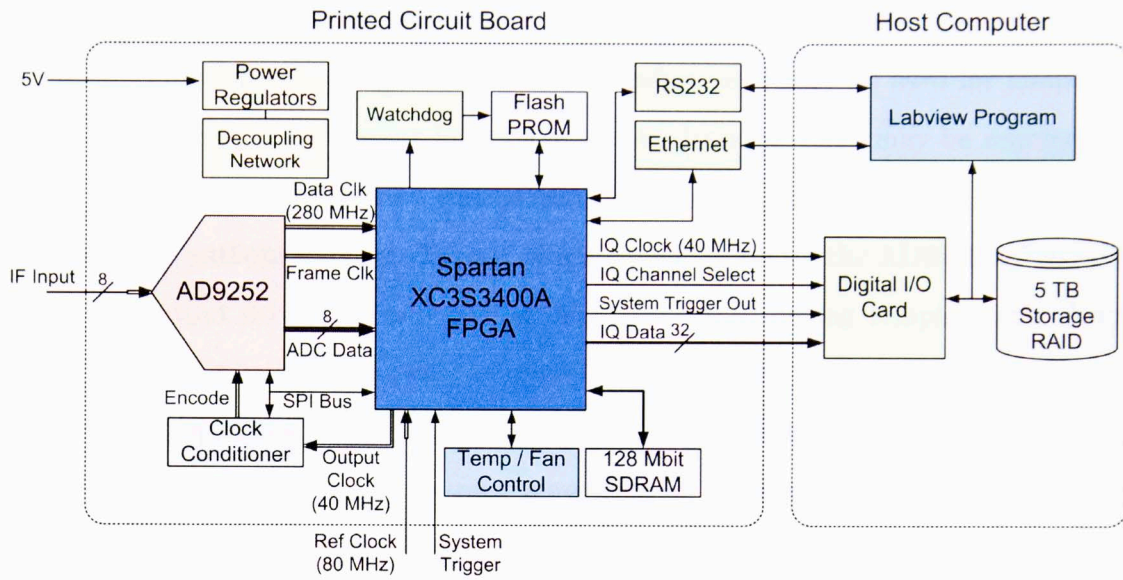


Figure 33: Eight-Channel Digital Receiver Interface Diagram

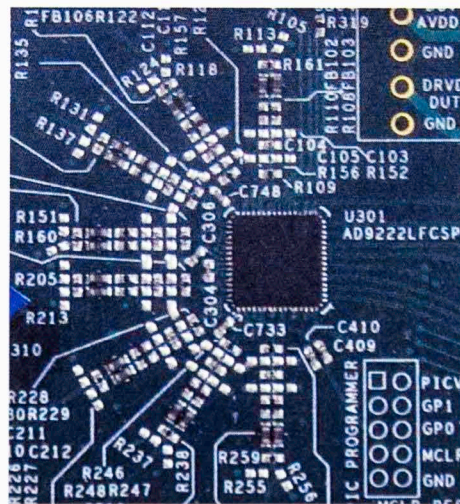



Figure 34: AD9252 ADC in a 64-pin LFCSP with Differential Input Filters



jitter must be less than 800 fs according to Equation (6). The AD9252 is available in a very small 64-pin lead frame chip scale package (LFCSP). This LFCSP requires the use of a thermal ground paddle underneath the package, therefore hand-soldering is not an option in this design. An infrared or reflow oven may be used for mounting the parts to the printed circuit board, but more advanced ovens may be required to meet the preheat, soak, reflow, and cooling profile of all components on the printed circuit board without damage. Typical power consumption for the AD9252 is 750 mW including output drivers, with 2 mW of power consumed during complete shut-down mode. The ADC also has an SPI interface which can be used to control all aspects of the ADC operation, including changing the output format of the data, powering down channels, executing test pattern generation on the digital outputs, and shifting the serial clock phase relationship with the data.

The eight analog input connections on the printed circuit board are arranged along one side in a linear configuration, as seen in Figure 35. The single-ended SMA connections are differentially coupled using a 1:1 RF transformer to maintain the highest SNR and distortion performance by reducing the common-mode swing of a single-ended input. All input channel trace lengths are made to be the same length through introducing curves of specific amplitudes and lengths in the differential lines, so that the SMA connectors appear electrically to be arranged along the circumference of a circle around the ADC. A differential analog 2-pole lumped filter is inserted directly before the pins of the ADC for three purposes.

The first purpose is to provide a better match with the input signal of 50 Ω . Because the ADC is an unbuffered, switched capacitance type, the ADC input impedance is time-varying between the two modes of operation [31]. The input impedance can be modeled as a capacitor and a resistor in parallel with the differential input pins, shown in Figure 36. When the ADC is charging the internal capacitor to the voltage of the input signal, it is in track mode, and displays a large capacitance. When the ADC is converting the voltage stored on the internal capacitor into a digital word, it is in hold mode with a larger resistance and a lowered capacitance. The track mode input

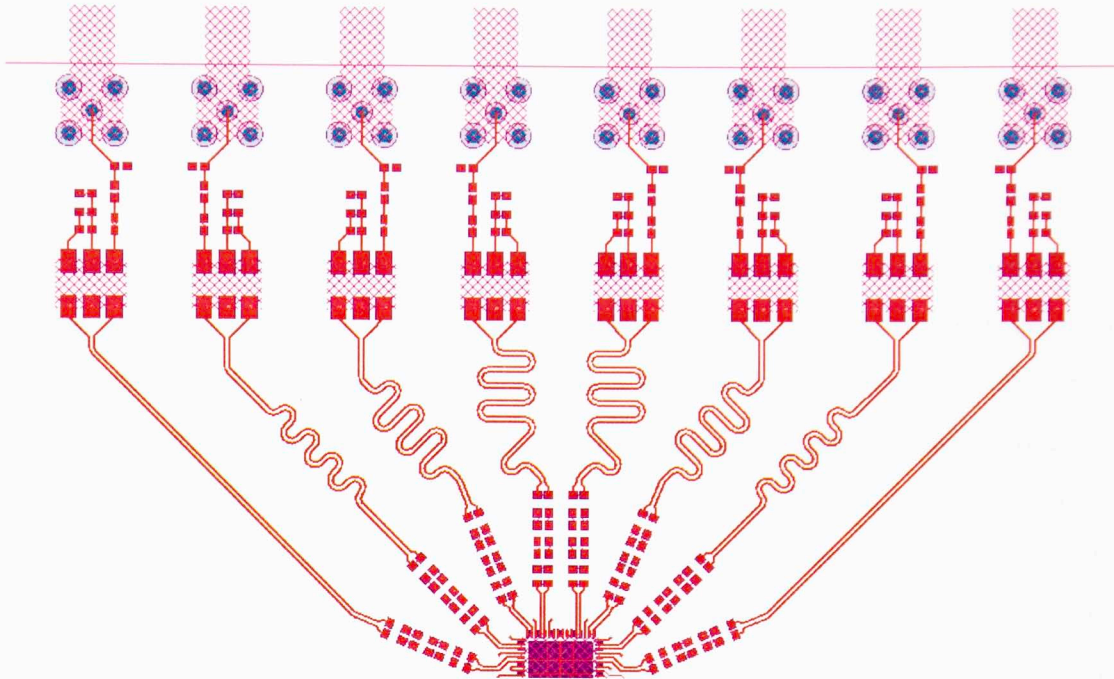


Figure 35: Analog Input Connection Layout Design

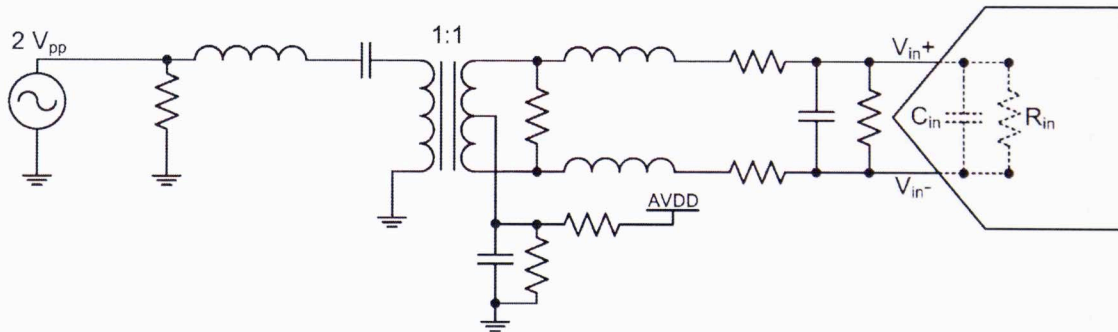



Figure 36: Analog Input Filter Schematic

impedance is used because the ADC is actively sampling during this time. Once this impedance is known, the differential filter can be simplified into a single impedance in parallel with the transformer which can then be brought across the transformer so an equivalent load can be found to match the source impedance.

The second purpose is to filter out higher-frequency signals or noise which may alias back into the Nyquist zone when sampled. Both the lumped filter and the analog input bandwidth of the ADC serve to limit the amplitude of input signals at frequencies above 55 MHz. By attenuating these signals as much as possible, the equivalent noise



bandwidth of the system is reduced. Furthermore, changing between track and hold mode by connecting and disconnecting the sampling capacitor creates an injection of charge on the input lines. To protect the input source, the charge can be dissipated with low-value series resistors which are present in the differential lumped filter.

One of the main concerns for the ADC design is to maintain a clean separation of analog and digital electronics as much as possible. Digital components can generate a large amount of noise from the many sudden voltage changes constantly occurring internal and external to each component. To this end, several methods are combined to minimize the effect of this noise on the analog inputs. The ADC has a dedicated linear voltage regulator kept away from the digital switching voltage regulators to ensure the most interference-free supply voltage. Physical separation of the components is also enforced to minimize any digital current return paths through the analog section of the printed circuit board. A ground plane notch is placed along the upper edge of the analog section of the PCB, shown in Figure 37, as a barrier to any high-frequency signals which may exist in the power supply or fan control components near the first ADC channel trace. A line of closely spaced, grounded vias on either side of differential traces were placed where possible to act as a shield from interference by creating a semi-enclosed tunnel around sensitive traces.

The output interface on the AD9252 consists of eight serial LVDS signals with a LVDS serial bit clock and frame clock. The frame clock determines when a new data byte starts, and has a frequency equal to the encode rate. The LVDS signals are dual-data rate, so the serial clock frequency is half the bit rate, where the bit rate is equal to the encode rate multiplied by the resolution. When digitizing with 14-bit resolution at a 40 MSPS encode rate, the bit rate on each serial link is $14 \text{ bits/sample} * 40 \text{ MSPS} = 560 \text{ Mbits/s}$, and the serial clock is a 280 MHz signal which is phase shifted 90° from the data streams. These differential signals between the FPGA and ADC are among the highest-frequency signals in the eight-channel digital receiver design and are carefully laid out to maintain data integrity. The length of the traces was kept to 520 mil on average by placing the ADC and FPGA very close to one another so that no trace

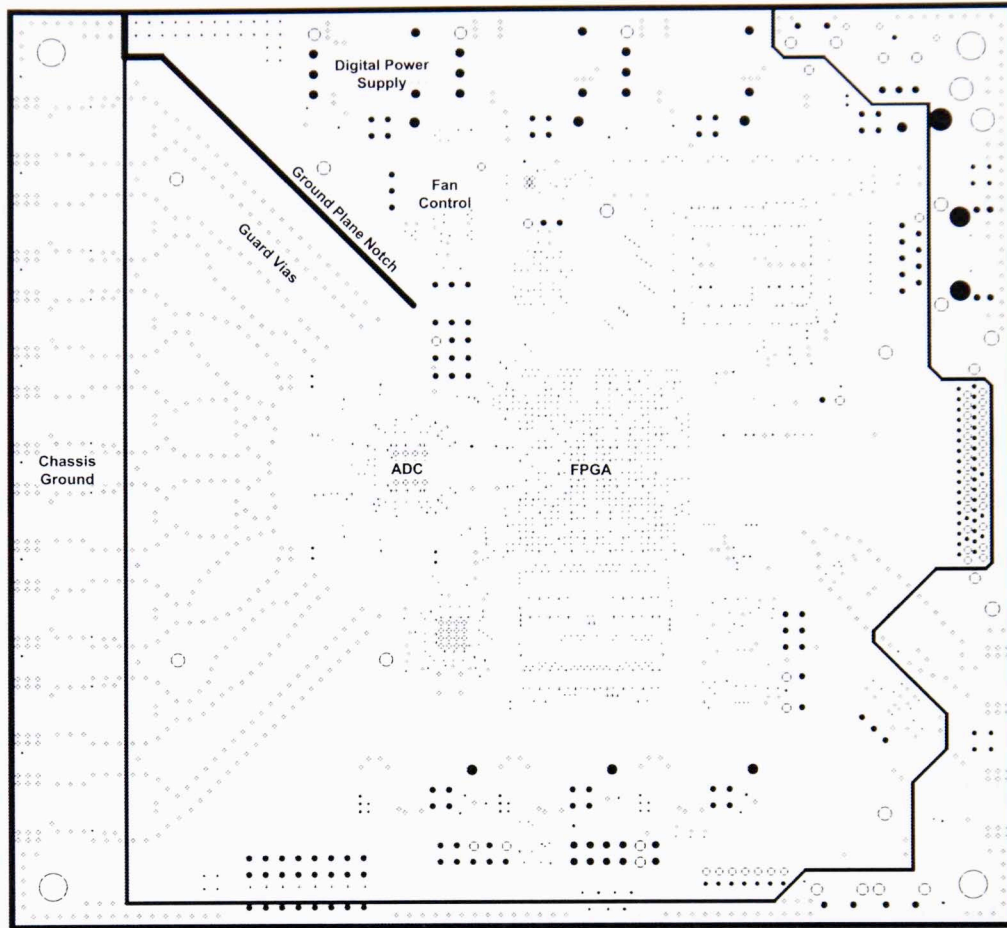



Figure 37: Digital Receiver Ground Plane Layout

is longer than 658 mil. Two methods of termination are available for the differential outputs. The first method is a resistor network package which can be soldered over the traces. 100 Ω terminations built this way have the advantage of precise termination impedance. Another method is to use the internal terminations available inside the FPGA. While the impedance of these differential terminations has a wide tolerance, the termination is placed directly in the input/output buffers (IOB) of the FPGA as close to the differential input buffer as possible. The internal terminations also do not require a physical component, and can be added or removed through a firmware change.



4.1.2 Field-Programmable Gate Array

The processing requirements of the eight-channel digital receiver present some unique challenges. Eight total channels must be processed simultaneously, necessitating a larger FPGA to perform the downconversion. Selecting only hand solderable components is not a requirement of this design as it was in the single-channel digital receiver, due to the need for smaller, more complex package types with better thermal characteristics and higher logic densities. The Spartan 3A DSP XC3SD3400A FPGA was chosen to be the digital processor. With 53,712 logic cells, 2,268 kbits of RAM, and 126 hardware DSP slices, this FPGA contains approximately ten times the amount of resources as the XC3S250E. An alternative XC3SD1800A FPGA contains about 33% fewer resources than the XC3S3400A and has been made pin-compatible with this FPGA in the design, allowing either Spartan 3A DSP version to be used in this digital receiver design. The smaller XC3SD1800A would be desirable for its smaller configuration memory footprint, larger number of user I/O pins, and slightly lower cost if the additional logic is not needed.

The FG676 package is a 27 mm x 27 mm ball-grid array package with 676 balls spaced 1 mm apart, and is available for both the XC3SD1800A and XC3SD3400A. Four signal layers are required to completely route all needed signals away from the FPGA package. The organization of the 10-layer printed circuit board stackup can be seen in Figure 38. The overall thickness of the PCB is a standard 63 mil thickness. The stackup consists of four signal routing layers which are separated by four ground planes. The ground planes reduce interference and crosstalk between signal layers and serve as a short low-impedance return path for high-speed signals. The two power planes lie in middle of the PCB structure and provide better separation between signals on the top half and bottom half of the PCB. The copper thickness of the interior layers is 0.5 oz in order to achieve a small layer separation of 5 mil, while the copper thickness of the two outer layers is 2 oz, creating a 50 Ω characteristic impedance for the 6 mil wide trace needed to escape the signals from the FPGA. In printed circuit board fabrication, the thickness of the copper layers used is specified by the weight of copper per square foot.

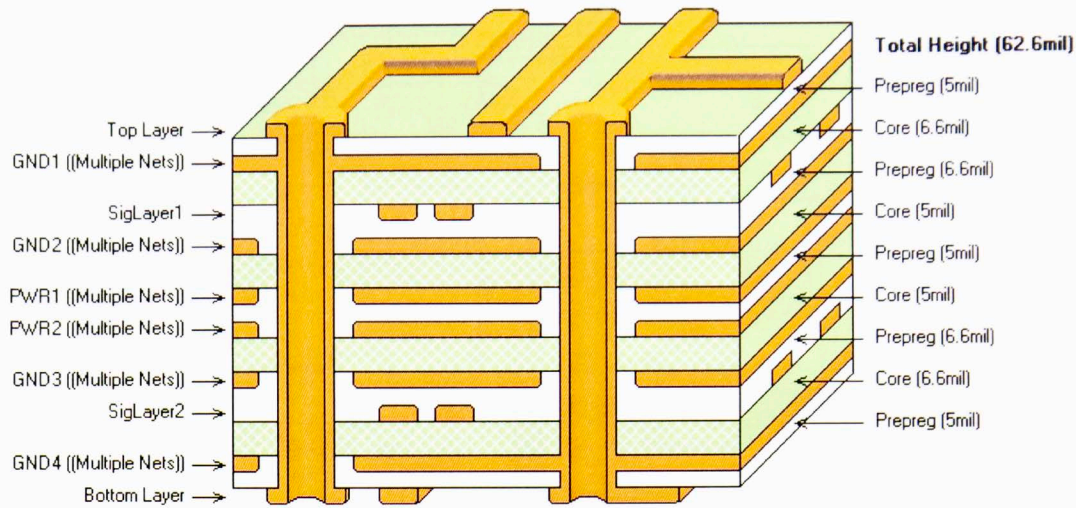


Figure 38: Eight-Channel Digital Receiver Printed Circuit Board Stackup


Using the copper's density of 8.9 g/cm^3 , the thickness of 0.5 oz of copper spread over 1 ft^2 is found to be $17.5 \text{ }\mu\text{m}$ and the thickness of 2 oz of copper is found to be $70 \text{ }\mu\text{m}$.

4.1.3 Peripheral Devices

The eight-channel digital receiver also includes a number of peripheral devices which are used to create a more robust and flexible digital receiver. There is a fan controller and temperature monitor, 128 Mbit of SDRAM, two PROM devices, a dedicated watchdog circuit, six power regulators, and an ethernet interface.

The fan controller and temperature monitor is a MAX6639. The MAX6639 is able to monitor one local and two remote temperature-sensing diodes and control two standard 3- or 4-pin case fans. The temperatures recorded are accessed through a serial bus from the FPGA and temperature limits can be set to activate interrupts in the event of overheating. The fans will ensure a large, steady airflow over all of the components to reduce the likelihood of overheating even in high ambient temperatures.

128 Mbit of high-speed SDRAM has been placed in the design for use with large Microblaze firmware processor programs instantiated in the FPGA. The SDRAM greatly expands the amount of RAM available so that the full IP stack may be implemented, more complex libraries may be used, or large data buffers may be created.




asserted-low pulse on FPGA's PROG_B pin, which begins the configuration process. The watchdog circuit can default to either PROM component through a jumper setting and the timeout can be changed to be a power of two between 2^1 and 2^{32} clock cycles. With a 32.768 kHz clock, this corresponds to timeout settings between 61 μ s and 131,072 seconds.

The eight-channel receiver design requires five distinct power supplies. There are four digital power supply voltages of 3.3 V, 2.5 V, 1.8 V, and 1.2 V and one analog supply voltage of 1.8 V. The analog supply is the sole linear supply while the rest are switching-type power supplies. Routing the many power supply voltages on the printed circuit board is made somewhat simpler through the use of two power supply planes rather than one, as was used in the single-channel receiver design.

Finally, an ethernet interface is included in this design. Using this interface as the primary data output interface is not possible due to hardware limitations in Spartan 3A FPGAs. However, the 5-10 Mbit/s link is very useful as a standard, scalable remote control interface. The physical layer of the ethernet specification is implemented with a DP83865DVH Gigabit ethernet device, although the highest achievable link speed is 100 Mbit/s half-duplex. Higher layers of the OSI model are implemented on the FPGA using a GMII-type interface.

4.1.4 Clock Conditioner

The clock of the ADC is driven by a special device known as a clock conditioner. The National Semiconductor LMK03000C clock conditioner is the combination of a phase-locked loop (PLL) and a voltage controlled oscillator (VCO), which can collectively be used to change an input clock with a high amount of jitter into several output clocks with much reduced jitter. The amount of jitter removed from the input clock is related to the bandwidth of the loop filter which controls the VCO tuning voltage. By designing a narrow loop filter bandwidth, the dominant noise source of the output clock becomes the VCO itself. The transfer function of the phase detector, N and R divider circuits, and reference noise are low-pass functions, while the VCO noise



contribution is a high-pass function. The cutoff for both the low-pass and high-pass transfer functions is the loop filter bandwidth, which should be carefully selected to balance the contribution of VCO noise and PLL noise in order to achieve the lowest total noise on the output waveform [33].


Secondary characteristics of the loop filter design include phase margin, which controls the amount of VCO control voltage damping and affects lock time, and γ , which controls where phase margin is maximized in the loop filter response and can further decrease lock time in certain cases. Phase margin may be between 0° and 90° , both of which will result in filter instability. A higher phase margin may be chosen to overdamp the tuning voltage at the expense of lock time, but will provide higher VCO noise suppression near the loop bandwidth cutoff frequency.

For the eight-channel digital receiver design, a loop bandwidth of approximately 10 kHz has been chosen because of the expected high jitter on the input reference clock, relatively low jitter of the integrated VCO, and deleterious spreading effect of close-in clock phase noise when digitizing radar data.

4.1.5 Host Computer

The host computer for this design has the following constraints:

- Digital I and Q data from the eight-channel receiver shall be read through a VHDCI connector with a 32-bit bus operating at 40 MHz.
- The digital time series data shall be stored to a redundant array of disks continuously for an extended period of time.
- Processing of the data shall be employed in tandem with data storage tasks in a way which does not interfere with their operation.
- The host computer shall allow a simple method of replacing defective hard drives and shall allow for future expansion of the storage array.
- An ethernet control interface shall be provided in order to verify operational readiness, status, and to transfer data to a central computer for post-processing.



These requirements and the need for long-term operation align directly with the requirements of server-style computer operational methodologies. Servers support many hard drives, fast parallel processors, large RAM caches, and are designed for high reliability. A Supermicro X7DWE motherboard was chosen to be the central system component, allowing for dual LGA-771 socket Xeon processors, four DIMMs of ECC RAM, dual gigabit ethernet ports, four PCIe x8 slots, and one PCI-X slot in an ATX form factor. To house this motherboard, a Supermicro SC743TQ-865B 4U rack-mountable chassis was chosen. This chassis includes an 865 W power supply with redundant cooling, eight hot-swappable SATA hard drive bays, two 5.25" peripheral drive bays, and one floppy drive. The processors were chosen to be two quad-core E5410 Xeon processors, giving the server eight total cores for parallel operation. To eliminate the bottleneck of RAID processing, a dedicated Areca ARC-1230 hardware RAID card was chosen, which can support up to 12 SATA drives in a RAID 6 configuration. Six 1 TB Samsung Spinpoint F1 hard drives were chosen to store the acquired digital receiver data, each of which is capable of more than 100 MB/s storage throughput. The chassis allows for an additional four hard drives to be mounted, allowing the RAID to be easily expanded. The National Instruments PCIe-6537 was chosen to be the digital I/O card in this design as it was in the single-channel receiver. These hardware selections allow the design to meet all of the current requirements, while retaining room for future expansion if operational expectations are increased.

4.1.6 Hardware Development

Although based upon the single-channel receiver, the eight-channel digital receiver includes many more advanced features and new components which necessitated a complete redesign. This redesign was completed in one step, rather than incrementally adding and validating new features with custom printed circuit boards at each step. Creating a simpler, feature-incomplete version of the digital receiver before moving to the final stage of the design could not be made an efficient use of design time because

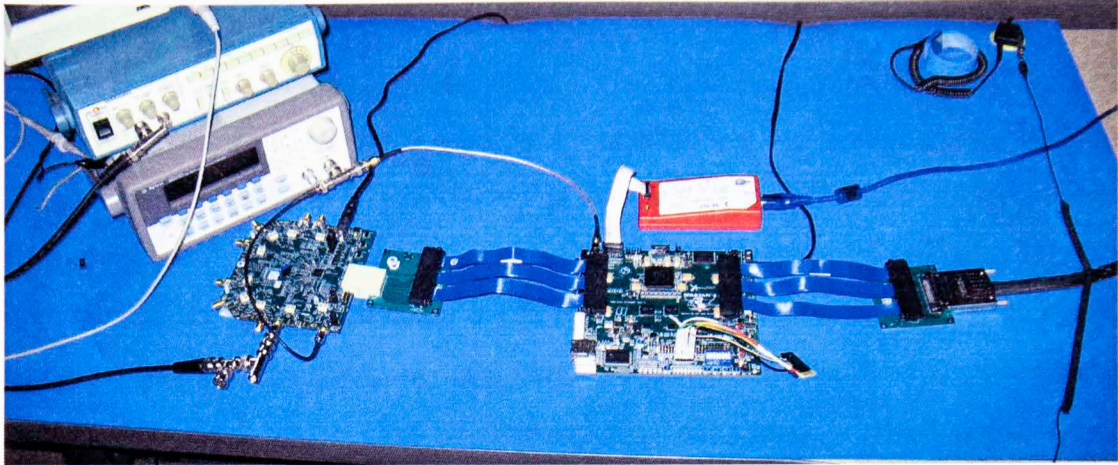



Figure 40: Eight-Channel Digital Receiver Evaluation Board Setup

many components require special considerations when being placed, resulting in a complete redesign at each stage. The complexity of the design also incurs an increased cost, further discouraging iterative development. In light of these constraints, a single full-featured prototype would be built while paying careful attention to each detail and testing all possible features before fabrication on evaluation boards or a simple custom PCB.

The primary platform for algorithm testing can be seen in Figure 40. The evaluation board on the left is the AD9252 ADC, from which the serial data streams cross over a custom PCB which connects the differential signal connector to a high-speed Samtec cable. The Samtec cable connects to a Spartan 3A DSP XC3SD1800A evaluation board, which outputs the digitized I and Q data on another Samtec cable to custom PCB which connects to the VHDCI cable attached to the host computer. This setup allows for a full test of the core digital receiver functionality from the bandpass sampling process to digitization and digital processing and finally to the post-processing on the host computer. The Spartan 3A DSP evaluation board also includes a gigabit ethernet interface so that network control functionality could be tested before fabrication of the custom PCB.

In parallel with the firmware design and testing, the hardware design proceeded by first creating the component interconnect schematic. Once the components and their



interconnections had been decided, each component was placed and the connections routed on the printed circuit board layout. The 3-dimensional model of the printed circuit board shown in Figure 41 provides an excellent method of viewing how the digital receiver would look before fabrication of the design. A model of the enclosure built to house the digital receiver after fabrication and assembly is shown in Figure 42. The approximate size of the enclosure is 6.4" x 6" x 1.5". The digital receiver was designed to be a standalone unit, capable of operating independently of other devices in the radar system. It also places low requirements on input signals, so that operation can continue even under extremely non-ideal conditions. Despite the independent model of operation, the digital receiver was also designed to respond to changes in inputs. For example, the temperature sensor can increase fan speed if the environmental temperature increases, or the FPGA can shut down parts of the system if their service is unneeded, saving power. The ethernet control interface allows the system to change drastically based on the user's needs. This design promotes a high level of flexibility so that the digital receiver can be used in a variety of contexts such as airborne radar systems, mobile radar systems, and ground based systems.

4.2 Software Design

This section will discuss the method of operation for both the firmware loaded onto the FPGA and the software running on the host computer, including descriptions of the communications protocols used in the design.

4.2.1 FPGA Firmware Structure

The firmware of the eight-channel digital receiver functions, at its core, much the same as the single-channel receiver, while broadening the number of conversion channels. A block diagram of the digital receiver's firmware structure is shown in Figure 43. The processing begins with eight parallel LVDS signals entering the FPGA, each operating at a bitrate of 560 Mbit/s. These signals are promptly buffered and sampled into two streams of bits generated from the rising and falling edges of the 280 MHz

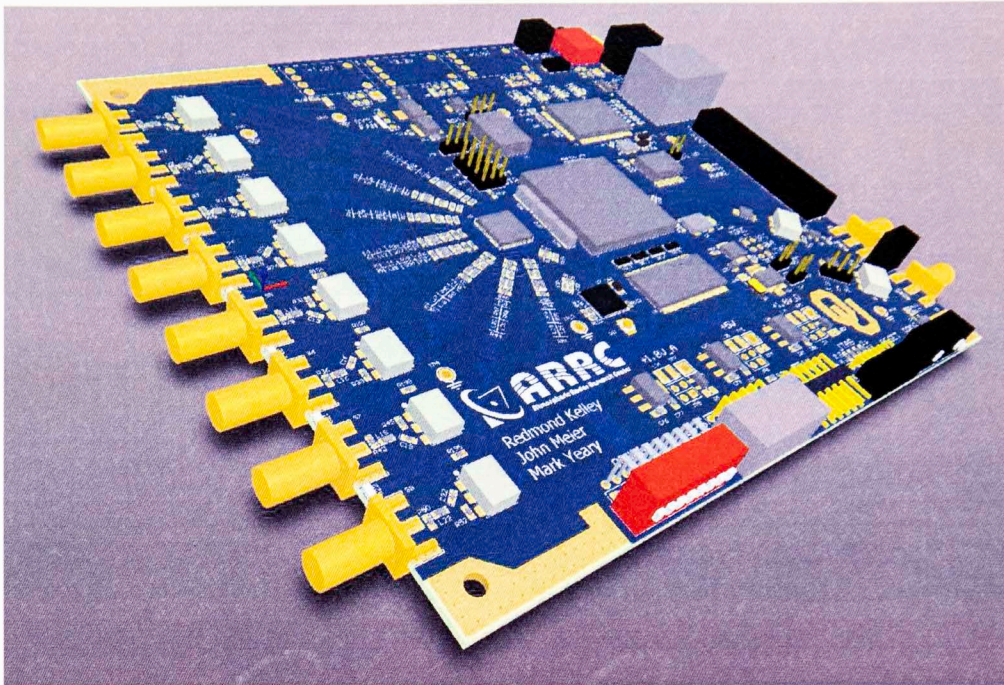


Figure 41: Eight-Channel Digital Receiver Printed Circuit Board 3D Model

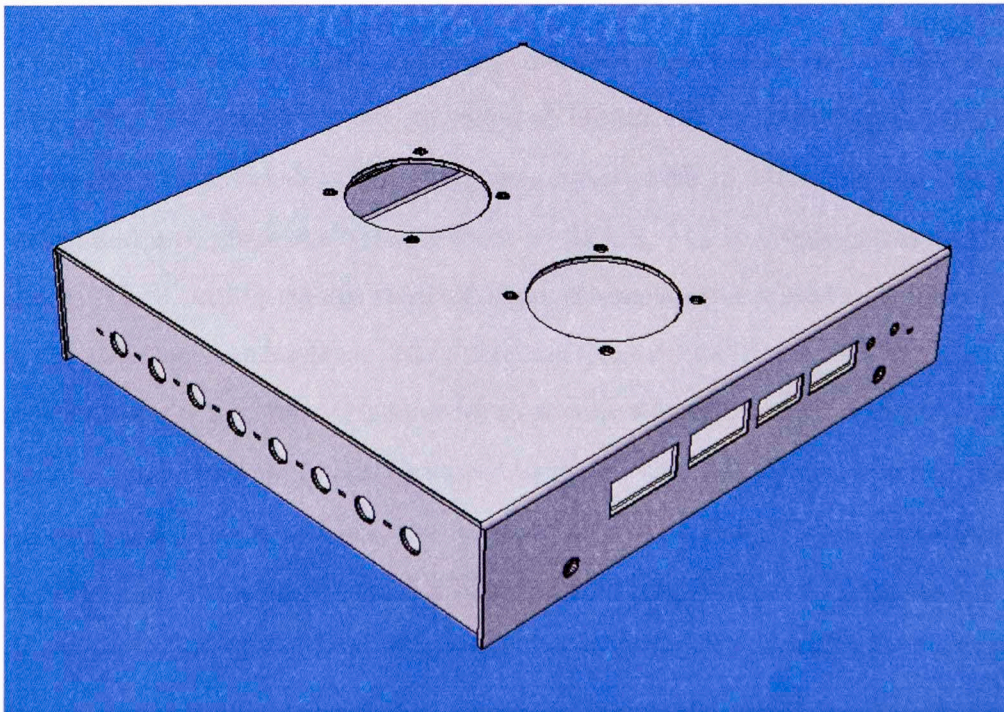


Figure 42: Eight-Channel Digital Receiver Enclosure

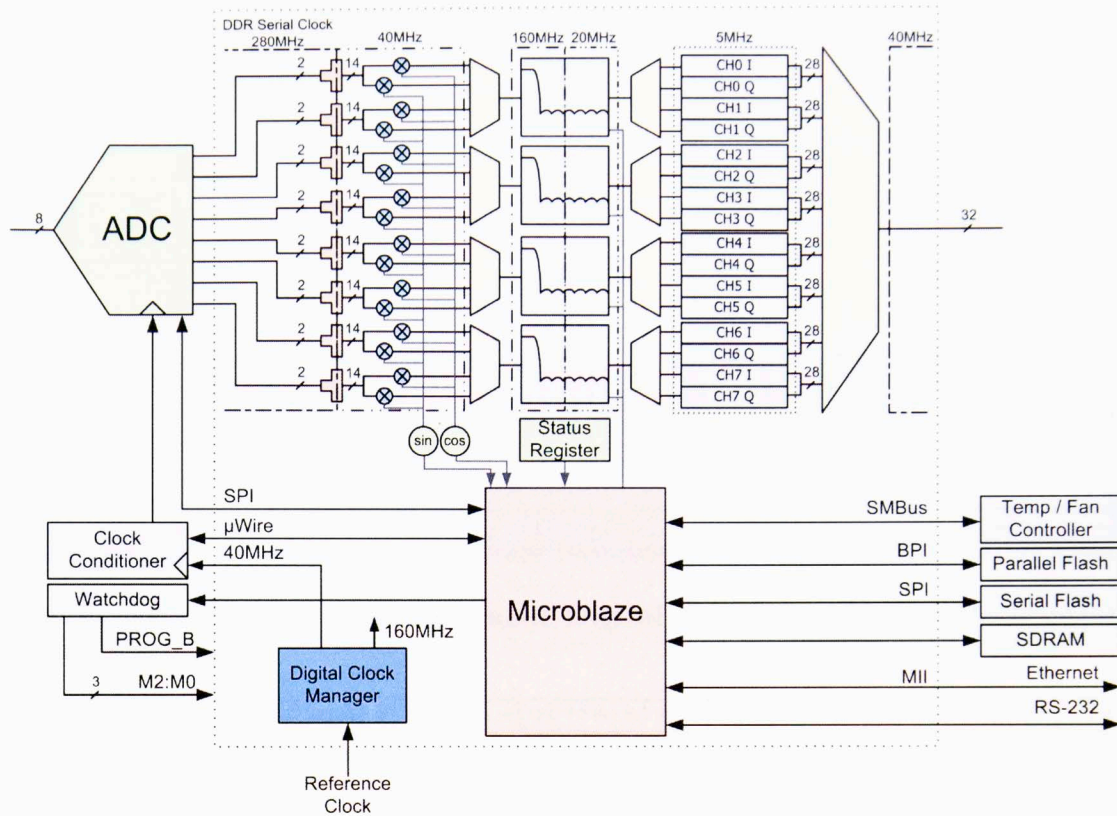


Figure 43: Eight-Channel Digital Receiver Block Diagram

serial clock. The even and odd bitstreams are then deserialized into 14-bit words at 40 MHz using relationally placed flip-flops to obtain the smallest signal path delay. After deserialization, the data words are then mixed with 10 MHz sine and cosine signals simplified into phase-shifted sequences of $\{0, 1, 0, -1\}$ to create quadrature I and Q signals. The I and Q signals from all eight channels are coupled into four streams which are then time-multiplexed into a four low-pass decimation-by-eight filters. This requires a filter clock of 160 MHz in order to acquire four data words every 25 ns. Multiple sets of filter coefficients can be stored in the FPGA at once and selected between during run time. At a minimum, the filter would need a flat response from baseband to 2.5 MHz and a stopband from 2.5 MHz to 20 MHz, as shown in Figure 44. This response would filter the unwanted spectral image at 20 MHz as well as all intermodulations and quantization noise in the decimated bandwidth which would otherwise be folded back onto the desired spectrum. The generated I and Q samples can be further

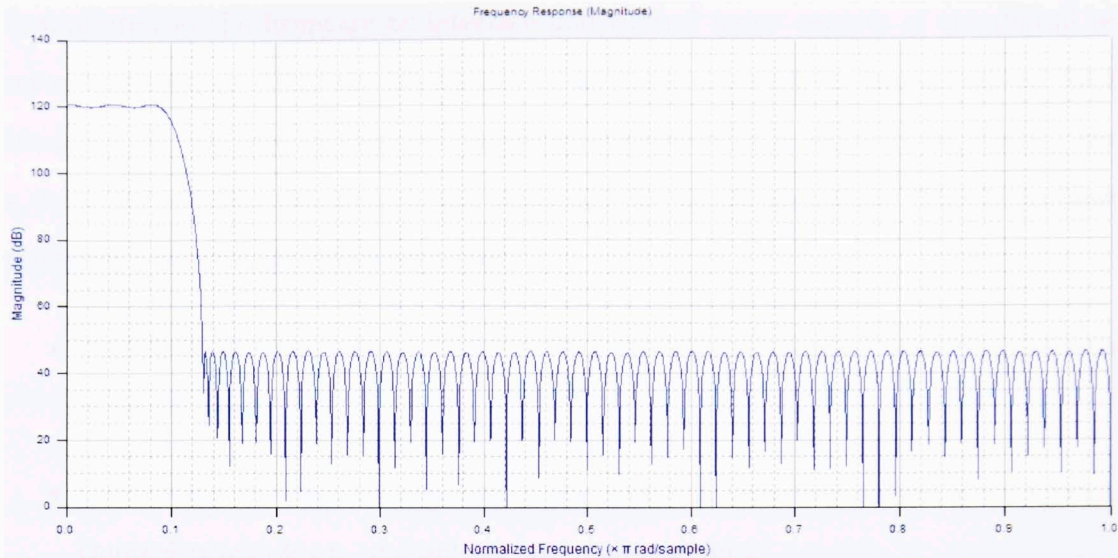


Figure 44: Minimal Eight-Channel Digital Receiver 128-tap Decimation Filter Response


Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	System Trigger	Channel	MSB	I Sample												LSB	MSB	Q Sample												LSB		

Figure 45: Eight-Channel Digital Receiver Output Word Format

filtered during post-processing if required.

Following decimation, the output words are generated at a rate of $5 \text{ MHz} * 4 = 20 \text{ MHz}$ and are reorganized into their respective I and Q channel banks. Each channel's data word is then produced in order from zero to seven onto a 32-bit bus every 25 ns along with its respective channel number and system trigger information formatted by the word structure shown in Figure 45. The presence of a high number of output pins on a single bank of the FPGA requires consideration of the maximum number of simultaneously switching outputs (SSO) per power and ground pin pair in order to avoid ground bounce. With ten power and ground pairs per bank, the highest recommended number of SSOs is 50 for fast slew rate LVTTTL outputs with a drive strength of 6 mA [34]. Based on this recommendation, the drive strength for the output pins has been reduced.

In parallel with the main digital receiver processes, a Microblaze processor is also



instantiated in the firmware to interface and control many aspects of the digital receiver's operation. All of the Microblaze code is stored on-chip in two initialized 64kB block RAMs. By storing the code on-chip, no bootloader is required for startup and a cache for off-chip resources is not necessary for quick operation. The Microblaze processor performs the following tasks:

- Disabling the watchdog circuit so that the digital receiver operation remains uninterrupted.
- Controlling the internal registers of the AD9252, including output word format, channel power-down, and output test mode settings.
- Initializing the LMK03000 clock conditioner so that a clean 40 MHz signal is generated for the ADC.
- Enabling and disabling the I and Q modulation digital processing.
- Reporting the contents of the internal status register of the FPGA on any communications interface.
- Changing the coefficients of the low-pass decimation filters.
- Monitoring the current temperatures across the board, increasing fan speed, issuing a clock conditioner recalibration if the temperature varies, and disabling digital receiver operation if the temperature becomes dangerously high.
- Updating the configuration stored in either the Parallel Flash PROM or Serial Flash PROM.
- Providing an ethernet and RS-232 communication control interface.

The SDRAM component in this design could be used for expansive buffering or program storage, but the currently unimplemented firmware interface remains for future work. A novel ethernet interface has been constructed which allows for simple remote

1 Byte	0-3 Bytes or String
Opcode	Operand

Table 4: Control Packet Structure

control of all of the Microblaze-controlled systems. This communication protocol is discussed below.

The ethernet interface uses solely the user datagram protocol (UDP) for communication and has two active listening processes. The first process is a general command interface which generates a reply for any packet sent after performing the requested action. The second process is a modified trivial file transfer protocol (TFTP) [35] server which mirrors disk access with reads and writes to the flash PROM. The structure of a control packet is shown in Table 4. The first byte of a control packet is a 1-byte opcode, which is then followed by a variable-size operand, dependent on the specific opcode. The length and format of the operand varies with the type of control packet. Control packets can be segmented into two groups: request packets and response packets.

Request packets initiate a transaction and are sent from a device other than the digital receiver as enumerated in Table 5. All of the request packets are issued with the UDP protocol to the digital receiver control port 10,000. A ping control packet requests an echo from the digital receiver and is used to verify communication link connectivity. Sending a ping request to the subnet broadcast IP will cause all active digital receivers to respond with their current IP address. A flash status request will return the current status of the flash as to whether a program or erase operation is in progress. Issuing an ADC register access request will begin a read or write operation on the ADC's internal registers. If a read is requested, operand byte 2 is ignored. The flash may be erased by two methods. Using method 0x00, the entire PROM is erased at once, requiring approximately 44 seconds. Method 0x01 erases only the first sixteen sectors, requiring approximately 6 seconds. A confirmation byte of 0xAA is required to verify that flash erasure is requested. The FPGA may also be immediately reset or forced to reload its configuration from flash memory with a request. The I and Q modulation may be halted or restarted with a request. Finally, the coefficient set for

Opcode	Meaning	Operand Byte 0	Operand Byte 1	Operand Byte 2
0	Ping	Unused	Unused	Unused
2	Flash Status Request	Unused	Unused	Unused
3	Status Register Request	Unused	Unused	Unused
4	ADC Register Access	R/W _Z	Address	Value
5	Flash Erase	Method	0xAA	Unused
6	FPGA Reset	Unused	Unused	Unused
7	FPGA Reload	Unused	Unused	Unused
8	I/Q Modulation	On/Off	Unused	Unused
9	Filter Coefficient Set	Value	Unused	Unused

Table 5: Control Request Packet Opcodes

the digital low-pass filters may be changed by requesting a new set.

Response packets are issued only by the digital receiver and are enumerated in Table 6. Several requests may merely issue acknowledgments after internally carrying out the command. A positive acknowledgment is given with a byte of 0x01 and a negative acknowledgment is denoted with a byte of 0x00. If a request is negatively acknowledged, then the request has failed. The digital receiver may also provide an output string for display to the user as a notification. When given a flash status request, the response is two bits with the LSB representing whether the flash is currently being erased and the second LSB representing whether the flash is currently being programmed. The status register response is four bytes long, with the first byte constituting the least significant byte of the status register and the last byte representing the most significant byte of the status register. Following any ADC register access request, the desired register of the ADC is read and the current value returned in a packet along with the byte denoting whether the previous command issued was a read or write request and the address accessed on the ADC.

The second process listening to the digital receiver ethernet interface is a TFTP server listening on port 69. The TFTP server responds to read requests by reading from the flash PROM and sending the data over the ethernet link. The TFTP server responds to write requests by writing to the flash PROM, always starting from address

Opcode	Meaning	Operand	Operand Type
0	Ping	Acknowledge	1 Bit
1	User Message	Message Text and CR/LF	String
2	Current Flash Status	Programming, Erasing	2 Bits
3	Status Register Response	Value	4 Bytes
4	ADC Register Access	R/W_z, Address, Value	3 Bytes
5	Flash Erase	Acknowledge	1 Bit
6	FPGA Reset	Acknowledge	1 Bit
7	FPGA Reload	Acknowledge	1 Bit
8	I/Q Modulation	Status	1 Bit
9	Filter Coefficient Set	Acknowledge	1 Bit

Table 6: Control Response Packet Opcodes

0, until the TFTP client signals an end of transaction. Both read and write requests ignore the filename given during the transaction so that any filename may be chosen at the convenience of the requesting terminal. The TFTP server does not re-issue unacknowledged packets after a timeout, making this implementation an unreliable one.

Figure 46 shows the conceptualized utilization of the digital receiver’s resources. This design utilizes approximately 40% of the logic slices, 50% of the DSP48A slices, and 80% of the BRAM resources on the FPGA. Approximately 85% of utilized BRAM in the design is consumed by the Microblaze processor code. By moving this code off-chip and executing in RAM, the utilization decreases to nearly 20%. This leaves a considerable amount of memory available for future expansion of capabilities or further processing.

4.2.2 Host Computer Software Operations

While the host computer of the eight-channel receiver design performs the same task as in the single-channel receiver, the data bandwidth which must be stored has been increased eight fold from 20 MB/s to 160 MB/s. Additional requirements have also been placed on the design to make it more robust to data loss and to allow for

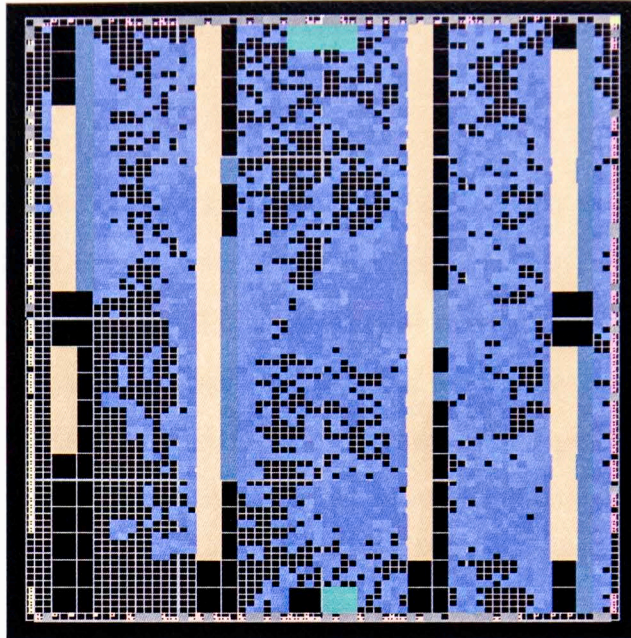


Figure 46: Eight-Channel Digital Receiver FPGA Floorplan Utilization

expansion of capabilities in the future with minimal time and materials investment.

The Labview program, which acquires and stores the data, needed modification in order to properly store the expanded data throughput. Because Labview is a dataflow language, copies of data variables may inadvertently be made even for trivial programs. In the case of large datasets, the creation time of these copies is non-negligible and may overrun the program's memory allocation. To overcome this issue, the program must work with smaller arrays by breaking up the large dataset into smaller pieces which can be copied and processed faster. For the eight-channel digital receiver host computer software, the data acquisition length was shortened to accommodate the bandwidth increase. Processing smaller arrays allowed the host computer to store data at 385 MB/s when five of the hard disks were combined in a RAID 0 configuration, far exceeding the required 160 MB/s throughput.




CHAPTER 5: CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

Two digital receiver designs have been presented which utilize software-defined radio techniques to achieve a lower power consumption, a smaller form factor, and a flexible architecture. The single channel digital radar receiver design digitizes one analog intermediate frequency input signal using a bandpass sampling technique to efficiently create I and Q time-series data streams. This prototype design can be made from a simple four-layer printed circuit board using only hand-solderable parts to achieve a 61.8 dB SNR while consuming only 1.785 W of power. The single channel receiver was tested at the National Weather Radar Testbed (NWRT) to collect radar data during clear air and squall line conditions. To elaborate, during the clear air conditions, the typical NWRT ground clutter test patterns were studied. The clutter echoes received from the clear air conditions matched well with the time series data acquired by digital receiver used at the NWRT. When observing a squall line, the reflectivity data recorded by the single channel digital receiver correlated well with the level-II reflectivity data despite some noise degradation from the lower sensitivity of the NWRT. The velocity data was not able to be adequately compared due to the large amount of range and velocity aliasing.

The eight channel receiver is a more complex 6.1" by 5.6" 10-layer printed circuit board which expands the design to accommodate eight input channels as well as several peripheral devices to create a more robust and flexible system. These devices include SDRAM to store extensive programs and process large arrays of data, two flash PROMs to safely allow for changes to the configuration image, a watchdog circuit to switch to the backup PROM in case of any firmware or software failure, a clock conditioner to remove as much phase noise from the ADC clock as possible so that a high SNR is achieved, temperature and dual fan control to maintain system integrity and to recalibrate the clock conditioner as the situation requires, and finally an ethernet interface so that all receiver functionality can be controlled remotely.



Host computers were built for both digital receiver designs to acquire and store the generated I and Q data. The eight channel digital receiver host computer required a hardware RAID controller and several hard drives in order to write the full bandwidth of 160 MB/s to disk without interruptions. The host computer is easily expandable up to ten hard drives and provides a quick method of recovery in the case of media failure.

5.2 Scope of Effort

The research described in this thesis was conducted by a team of researchers in a laboratory environment. While many design responsibilities were shared between researchers, the focus of my skill was placed on the design, development, and testing of the digital receiver firmware and the host computer software. In particular, I was responsible for the following items in the course of this research:

- Development of the VHDL firmware and output word communication protocols for the single-channel and eight-channel digital receivers.
- Experimentation with digital low-pass decimation filter designs to minimize resource utilization for the single-channel and eight-channel digital receivers.
- Realization of the Microblaze processor C algorithms for the eight-channel digital receiver.
- Design of the Labview storage and post-processing virtual instrument programs for the single-channel and eight-channel digital receivers.
- Definition of the ethernet communication protocol for the eight-channel digital receiver and design of the Labview ethernet controller virtual instrument program.
- Implementation of the radar moment post-processing code for the single-channel and eight-channel digital receivers.

5.3 Future Work

Many improvements could be made on the design of the eight channel digital receiver, if time allowed:

- Fabricate, assemble, and test the complete eight channel digital receiver both in the lab and in an operational radar.
- Increase the clock speed of the VHDCI bus from 40 MHz to 50 MHz for use as a bi-directional communication interface.
- Investigate alternative output connector interfaces. Few interfaces support both the data bandwidth and the physical length required for the receiver. A high-speed connector such as PCI mezzanine card (PMC) may be used with a PMC to PCI expansion card in the host computer to transfer the data. This would complicate the acquisition software, but would lower the cost of the host computer.
- Redesign the digital receiver into a peripheral card with a PCIe or PCI-X interface which can be housed directly inside the host computer.
- Implement adaptive power control over the ADC, parts of the FPGA, and peripheral devices based on current settings and inputs could be useful to lower power consumption.
- Find the optimal pulse matched filter shape for a decimation-by-eight filter to achieve a slight gain in SNR.
- Implement the SDRAM interface for the FPGA firmware so that a new configuration image loaded over ethernet may be completely buffered, checked, and treated as a transaction before committing the new data to PROM.
- Extend the dynamic range of the digital receiver by attaching one input signal to two channels, one of which is amplified by a multiple of 6 dB. This would lower the minimum detectable signal, yielding a moderate increase in overall dynamic range.

- Implement a sensitivity time control circuit. Given knowledge of when the transmit pulse is sent the gain of each channel could be increased over the PRT to account for losses in return signal strength from farther range gates. This would allow for higher SNR at distant ranges at the cost of a more complicated front-end and extra post-processing to account for the gain added.
- Create a frequency-agile IF by controlling the downconversion mixer's local oscillator signal with a direct digital synthesizer (DDS) frequency source. The DDS frequency can be changed quite rapidly and could be used to correct for magnetron drift or to quickly change frequency channels for a frequency-domain multiplexed input signal.

LIST OF REFERENCES

- [1] J. Mitola, "The software radio architecture," *IEEE Communications Magazine*, pp. 26-38, May 1995.
- [2] R. Lackey and D. Upmal, "Speakeasy: the military software radio," *IEEE Communications Magazine*, pp. 56-61, May 1995.
- [3] M. Dillinger, K. Madani, and N. Alonistioti, *Software Defined Radio: Architectures, Systems, and Functions*, Wiley, 2003.
- [4] P. Burns, *Software Defined Radio for 3G*, Artech House, 2003. ISBN:1-58053-347-7.
- [5] R. Bagheri, A. Mirzaei, M. Heidari, S. Chehrazi, M. Lee, M. Mikhemar, W. Tang, and A. Abidi, "Software-defined radio receiver: dream to reality," *IEEE Communications Magazine*, pp. 111-118, August 2006.
- [6] W. Tuttlebee, *Software Defined Radio: Enabling Technologies*, Wiley, 2002.
- [7] S. Srikanteswara, R. C. Palat, J. H. Reed, P. Athanas, "An overview of configurable computing machines for software radio handsets," *IEEE Communications Magazine*, July 2003.
- [8] J. Reed, *Software Radio: A Modern Approach to Radio Engineering*, Prentice Hall, 2002.
- [9] J. Chapin, *Overview of Vanu Software Radio*, Vanu, Inc., 2002.
- [10] Software Defined Radio Forum, www.sdrforum.org
- [11] H. Tsurumi and Y. Suzuki, "Broadband RF stage architecture for software-defined radio in handheld terminal applications," *IEEE Comm. Magazine*, vol 37, no. 2, pp. 90-95, 1999.
- [12] J. Mitola, "Cognitive radio for flexible mobile multimedia communications," *IEEE Int. Workshop on Mobile Multimedia Communications*, pp. 10-13, November 1999.
- [13] B. Fette, *Cognitive Radio Technology*, Elsevier Science & Technology Books, 2006.
- [14] H. Arslan, editor, *Cognitive Radio, Software Defined Radio, and Adaptive Wireless Systems (Signals and Communication Technology)*. Springer, January 2007.
- [15] Hai Tao, L. Toth, J.M. Khoury, "Analysis of timing jitter in bandpass sigma-delta modulators," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, no. 8, pp. 991-1001, August 1999.

- [16] D. Akos, M. Stockmaster, J. Tsui and J. Caschera, "Direct bandpass sampling of multiple distinct RF signals," *IEEE Transactions on Communications*, vol. 47, no. 7, pp. 983-988, July 1999.
- [17] N. Wong and T. Ng, "An efficient algorithm for downconverting multiple bandpass signals using bandpass sampling," *IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 910-914, June 2001.
- [18] C. Tseng and S. Chou, "Direct downconversion of multiband RF signals using bandpass sampling," *IEEE Transactions on Wireless Communications*, vol. 5, no. 1, pp. 72-76, January 2006.
- [19] Linear Technology, Design Note 1013.
- [20] J. Horstmann, H. Eichel, and R. Coates, "Metastability Behavior of CMOS ASIC Flip-Flops in Theory and Test," *IEEE Journal of Solid State Circuits*, vol. 24, no. 1, pp. 146, February 1989.
- [21] H. Veendrick, "The behaviour of flip-flops used as synchronizers and prediction of their failure rate," *IEEE Journal of Solid State Circuits*, vol. 15, no. 2, pp. 169, April 1980.
- [22] L. Kim and R. Dutton, "Metastability of CMOS latch/flip-flop," *IEEE Journal of Solid State Circuits*, vol. 25, no. 4, pp. 942, August 1990.
- [23] Xilinx, XAPP094, "Metastable Recovery in Virtex-II Pro FPGAs".
- [24] Texas Instruments, SCZA004a, "Metastability Performance Of Clocked FIFOs".
- [25] A. Hairapetian, "An 81-MHz IF Receiver in CMOS," *IEEE Journal of Solid-state Circuits*, vol. 31, no. 12, pp. 1981, December 1996.
- [26] J. Wepman, "Analog-to-digital converters and their applications in radio receivers," *IEEE Communications Magazine*, vol. 33, no. 5, pp.43, May 1995
- [27] Analog Devices, Inc., AD9244 Data Sheet, Rev. C.
- [28] H. Johnson and M. Graham, *High-Speed Digital Design: A Handbook of Black Magic*, Prentice Hall PTR, 1993, pp. 360.
- [29] M. Skolnik, *Radar handbook*, McGraw-Hill Professional, 1990, pp. 111.
- [30] D. Forsyth, J. F. Kimpel, D. S. Zrnic, R. Ferek, J. F. Heimmer, T. J. McNellis, J. E. Crain, A. M. Shapiro, R. J. Vogt, and W. Benner, "Progress report on the national weather radar testbed (phased array)," *21st Intl Conf. on Inter. Inf. Proc. Sys. (IIPS) for Meteor., Ocean., and Hydr., Amer. Meteor. Soc.*, 2005, pp. 1.
- [31] Analog Devices, Inc., Application Note AN935.

- [32] Analog Devices, Inc., AD9252 Data Sheet, Rev. A.
- [33] National Semiconductor, Clock Conditioner Owner's Manual, Winter 2006.
- [34] Xilinx, DS610 Spartan-3A DSP FPGA Family: Data Sheet, pp. 36, March 2009.
- [35] K. Sollins, The TFTP Protocol (Revision 2), IETF RFC 1350, July 1992.
- [36] M. Yeary, W. Zhang, J. Q. Trelewicz, "A computationally efficient decimation filter design for embedded systems," *IEEE-IMTC*, May 2004.
- [37] Uwe Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, Springer-Verlag Berlin Heidelberg, 2001.
- [38] Peter J. Ashenden, *The Design Guide to VHDL*, Morgan Kaufmann, San Francisco, California, 2000.
- [39] Sen M. Kuo and Bob H. Lee, *Real-Time Digital Signal Processing*, West Sussex, PO 19 1UD, England, 2001.
- [40] Sanjit K. Mitra, *Digital Signal Processing, Second Edition*, Tata McGraw-Hill, 2001.
- [41] Ralf Niemann, *Hardware/Software Co-Design for Data Flow Dominated Embedded Systems*, Kluwer Academic Publishers, P.O. Box 17, 3300 AA Dordrecht, The Netherlands, 1998.
- [42] F.J. Taylor, *Digital Filter Design Handbook*, Marcel Dekker, Inc., NYC, 1984.
- [43] R. Jain, P. T. Yang, and T. Yoshino, "FIRGEN: A computer-aided design system for high performance FIR filter integrated circuits", *IEEE Trans. Signal Processing*, vol. 39, pp. 1655-1668, July 1991.
- [44] R. A. Hawley et al., "Design techniques for silicon compiler implementations of high-speed FIR digital filters", *IEEE J. Solid-State Circuits*, vol. 31, pp. 656-667, May 1996.

APPENDIX A: VHDL DESERIALIZER CODE FOR THE EIGHT CHANNEL DIGITAL RECEIVER

The following code instantiates one deserializer module with relationally-placed flip flops to achieve the smallest possible signal delays between logic slices. The code expects a serial clock and bitstreams of the sampled even and odd data bits and frame clock bits synchronized with the falling edge of the serial clock.

```
-----  
-- Filename:      deserial.vhd  
-- Author:       John Meier  
-- Date Created:  Sep 21, 2008  
-- Description:   Deserializes a DDR serial input signal into a  
--               14-bit output word.  
-----  
  
library IEEE;  
use IEEE.NUMERIC_STD.ALL;  
use IEEE.STD_LOGIC_1164.ALL;  
use IEEE.STD_LOGIC_UNSIGNED.ALL;  
  
library work;  
use work.commonpkg.ALL;  
  
library unisim;  
use unisim.vcomponents.ALL;  
  
entity deserial is port (  
    reset          : IN  slv01; -- Synchronous reset  
    clkin          : IN  slv01; -- DDR clock (2 bits/period)  
    clkin_z        : IN  slv01; -- Inverted DDR clock (2 bits/period)
```

```

serfrm_even      : IN  slv01; -- Frame clock (14 bits/period),
                    -- captured on falling edge clk_in
serfrm_odd       : IN  slv01; -- Frame clock (14 bits/period),
                    -- captured on rising edge clk_in
datain_even     : IN  slv01; -- Serial input data, captured
                    -- on clk_in_z (even bits)
datain_odd      : IN  slv01; -- Serial input data,
                    -- captured on clk_in (odd bits)
dataout         : OUT slv14; -- Deserialized data OUT
datavalid       : OUT slv01 -- Toggles when new word is ready
);
end deserial;

```

architecture deserial_arch of deserial is

```

-----
-- Flip Flop Layout:
-- (siX = serial input FFs, oXY = output bits X&Y,
-- oe = output enable FFs, dv = data valid FFs)
-- Two FFs are available per slice (each (X,Y) coordinate)
--          <<TOP OF FPGA>>
--          -----
--      3 |si0|si1|si2|si3|si4|  |
--      ^ 2 |si5|si6|si7|si8|f01|dv |
--      | 1 |  |  |oe |o01|o23|o45|
--      | 0 |  |o67|o89|oAB|oCD|  |
--      Y  -----
--          0  1  2  3  4  5
--      X  - - >

```



```

signal datain_qe      : slv09; -- Even bits shift register
signal datain_qo      : slv09; -- Odd bits shift register
signal frmin_q        : slv01; -- Delayed frame clock
signal frmin_qz       : slv01; -- Delayed frame clock
signal frmin_qq       : slv01; -- Delayed frame clock
signal oe             : slv01; -- Clock enable for output bits
signal oe_d           : slv01; -- Combinational pre-registered CE
signal frmshift       : slv03; -- Shift register for frame clock
signal dv_toggle      : slv01; -- Toggled data_valid
signal datavalid_tog_i : slv01; -- Combinational feedback buffer

```

```

attribute rloc          : string;

```

```

-- Even bit FF locations

```

```

attribute rloc of fdr_e0 : label is "x0y3";
attribute rloc of fdr_e1 : label is "x1y3";
attribute rloc of fdr_e2 : label is "x2y3";
attribute rloc of fdr_e3 : label is "x3y3";
attribute rloc of fdr_e4 : label is "x4y3";
attribute rloc of fdr_e5 : label is "x0y2";
attribute rloc of fdr_e6 : label is "x1y2";
attribute rloc of fdr_e7 : label is "x2y2";
attribute rloc of fdr_e8 : label is "x3y2";

```

```

-- Odd bit FF locations

```

```

attribute rloc of fdr_o0 : label is "x0y3";
attribute rloc of fdr_o1 : label is "x1y3";
attribute rloc of fdr_o2 : label is "x2y3";
attribute rloc of fdr_o3 : label is "x3y3";

```

```
attribute rloc of fdr_o4    : label is "x4y3";
attribute rloc of fdr_o5    : label is "x0y2";
attribute rloc of fdr_o6    : label is "x1y2";
attribute rloc of fdr_o7    : label is "x2y2";
attribute rloc of fdr_o8    : label is "x3y2";
```

```
-- Framing clock FF locations
```

```
attribute rloc of fdr_fq0   : label is "x4y2";
attribute rloc of fdr_fq1   : label is "x4y2";
attribute rloc of fdr_fq2   : label is "x2y1";
```

```
-- Frameclock shift FF locations
```

```
attribute rloc of fdr_dv0   : label is "x5y2";
attribute rloc of fdr_dv    : label is "x5y2";
```

```
-- Output FF locations
```

```
attribute rloc of fdre_do0  : label is "x3y1";
attribute rloc of fdre_do1  : label is "x3y1";
attribute rloc of fdre_do2  : label is "x4y1";
attribute rloc of fdre_do3  : label is "x4y1";
attribute rloc of fdre_do4  : label is "x5y1";
attribute rloc of fdre_do5  : label is "x5y1";
attribute rloc of fdre_do6  : label is "x1y0";
attribute rloc of fdre_do7  : label is "x1y0";
attribute rloc of fdre_do8  : label is "x2y0";
attribute rloc of fdre_do9  : label is "x2y0";
attribute rloc of fdre_do10 : label is "x3y0";
attribute rloc of fdre_do11 : label is "x3y0";
attribute rloc of fdre_do12 : label is "x4y0";
```

```
attribute rloc of fdre_do13 : label is "x4y0";
```

```
attribute s: string; -- Save net (Do not optimize away)
```

```
attribute s of serfrm_odd : signal is "yes";
```

```
attribute s of serfrm_even : signal is "yes";
```

```
attribute s of oe : signal is "yes";
```

```
attribute s of reset : signal is "yes";
```

```
BEGIN
```

```
-- Odd bit FFs
```

```
fdr_o0 : fdr port map (c => clkin_z, r => reset,  
                      d => datain_odd, q => datain_qo(0));
```

```
fdr_o1 : fdr port map (c => clkin_z, r => reset,  
                      d => datain_qo(0), q => datain_qo(1));
```

```
fdr_o2 : fdr port map (c => clkin_z, r => reset,  
                      d => datain_qo(1), q => datain_qo(2));
```

```
fdr_o3 : fdr port map (c => clkin_z, r => reset,  
                      d => datain_qo(2), q => datain_qo(3));
```

```
fdr_o4 : fdr port map (c => clkin_z, r => reset,  
                      d => datain_qo(3), q => datain_qo(4));
```

```
fdr_o5 : fdr port map (c => clkin_z, r => reset,  
                      d => datain_qo(4), q => datain_qo(5));
```

```
fdr_o6 : fdr port map (c => clkin_z, r => reset,  
                      d => datain_qo(5), q => datain_qo(6));
```

```
fdr_o7 : fdr port map (c => clkin_z, r => reset,  
                      d => datain_qo(6), q => datain_qo(7));
```

```
fdr_o8 : fdr port map (c => clkin_z, r => reset,  
                      d => datain_qo(7), q => datain_qo(8));
```



```

-- Even bit FFs
fdr_e0 : fdr port map (c => clkin_z,      r => reset,
                      d => datain_even, q => datain_qe(0));
fdr_e1 : fdr port map (c => clkin_z,      r => reset,
                      d => datain_qe(0), q => datain_qe(1));
fdr_e2 : fdr port map (c => clkin_z,      r => reset,
                      d => datain_qe(1), q => datain_qe(2));
fdr_e3 : fdr port map (c => clkin_z,      r => reset,
                      d => datain_qe(2), q => datain_qe(3));
fdr_e4 : fdr port map (c => clkin_z,      r => reset,
                      d => datain_qe(3), q => datain_qe(4));
fdr_e5 : fdr port map (c => clkin_z,      r => reset,
                      d => datain_qe(4), q => datain_qe(5));
fdr_e6 : fdr port map (c => clkin_z,      r => reset,
                      d => datain_qe(5), q => datain_qe(6));
fdr_e7 : fdr port map (c => clkin_z,      r => reset,
                      d => datain_qe(6), q => datain_qe(7));
fdr_e8 : fdr port map (c => clkin_z,      r => reset,
                      d => datain_qe(7), q => datain_qe(8));

```

```

-- Frame Clock FFs
fdr_fq0 : fdr port map (c => clkin_z,      r => reset,
                      d => serfrm_even, q => frmin_q);
frmin_qz <= NOT frmin_q;
fdr_fq1 : fdr port map (c => clkin_z,      r => reset,
                      d => frmin_qz,      q => frmin_qq);
oe_d <= frmin_qq AND frmin_q;
fdr_fq2 : fdr port map (c => clkin_z, r => reset,

```

```

        d => oe_d,    q => oe);

-- Data Valid FFs
fdr_dv0 : fdr port map (c => clkin_z,    r => reset,
                        d => serfrm_odd, q => frmshift(0));
fdr_dv  : fdr port map (c => clkin_z,    r => reset,
                        d => dv_toggle,  q => datavalid_tog_i);
datavalid <= datavalid_tog_i;
dv_toggle <= datavalid_tog_i XOR oe;

-- Output FFs
fdre_do0 : fdre port map (c => clkin_z, r => reset, ce => oe,
                          d => datain_qe(2), q => dataout(0));
fdre_do1 : fdre port map (c => clkin_z, r => reset, ce => oe,
                          d => datain_qo(2), q => dataout(1));
fdre_do2 : fdre port map (c => clkin_z, r => reset, ce => oe,
                          d => datain_qe(3), q => dataout(2));
fdre_do3 : fdre port map (c => clkin_z, r => reset, ce => oe,
                          d => datain_qo(3), q => dataout(3));
fdre_do4 : fdre port map (c => clkin_z, r => reset, ce => oe,
                          d => datain_qe(4), q => dataout(4));
fdre_do5 : fdre port map (c => clkin_z, r => reset, ce => oe,
                          d => datain_qo(4), q => dataout(5));
fdre_do6 : fdre port map (c => clkin_z, r => reset, ce => oe,
                          d => datain_qe(5), q => dataout(6));
fdre_do7 : fdre port map (c => clkin_z, r => reset, ce => oe,
                          d => datain_qo(5), q => dataout(7));
fdre_do8 : fdre port map (c => clkin_z, r => reset, ce => oe,
                          d => datain_qe(6), q => dataout(8));

```

```
fdre_do9    : fdre port map (c => clk_in_z, r => reset, ce => oe,
                             d => data_in_qo(6), q => data_out(9));
fdre_do10   : fdre port map (c => clk_in_z, r => reset, ce => oe,
                             d => data_in_qe(7), q => data_out(10));
fdre_do11   : fdre port map (c => clk_in_z, r => reset, ce => oe,
                             d => data_in_qo(7), q => data_out(11));
fdre_do12   : fdre port map (c => clk_in_z, r => reset, ce => oe,
                             d => data_in_qe(8), q => data_out(12));
fdre_do13   : fdre port map (c => clk_in_z, r => reset, ce => oe,
                             d => data_in_qo(8), q => data_out(13));

end deserial_arch;
```


APPENDIX B: VHDL IN-PHASE AND QUADRATURE MIXING AND DECIMATION FILTERING CODE FOR THE EIGHT CHANNEL DIGITAL RECEIVER

The following code instantiates two deserializer modules and one 160 MHz decimation filter. The deserialized data is mixed into in-phase and quadrature components or passed straight through to the decimation filter based on an external signal. After filtering, the data is output from the module.

```
-----  
-- Filename:      IQ_2Channel.vhd  
-- Author:       John Meier  
-- Date Created:  Sep 21, 2008  
-- Description:   Deserializes, separates into I and Q, filters,  
--               and decimates two data streams.  
-----
```

```
LIBRARY IEEE;  
USE IEEE.STD_LOGIC_1164.ALL;  
USE IEEE.STD_LOGIC_UNSIGNED.ALL;  
USE IEEE.NUMERIC_STD.ALL;  
USE IEEE.STD_LOGIC_MISC.ALL;
```

```
LIBRARY WORK;  
USE WORK.commonpkg.ALL;
```

```
LIBRARY UNISIM;  
USE UNISIM.VComponents.ALL;
```

```
ENTITY IQ_2Channel IS
```

```

GENERIC (
    threshold : integer := 8 -- ADC signal activity threshold
);
PORT (
    clk160mhz : IN  slv01; -- Filter clock
    clk40mhz  : IN  slv01; -- Primary logic clock
    reset     : IN  slv01; -- Global reset (active HIGH)
    reset_ser : IN  slv01; -- Serial reset (active HIGH)
    iqactive  : IN  slv01; --
    serclk    : IN  slv01; -- DDR clock (1 period = 2 bits)
    serclk_z  : IN  slv01; -- DDR clock (1 period = 2 bits)
    serfrm_even : IN  slv01; -- 40MHz serial frame clock
                                -- falling edge captured
    serfrm_odd  : IN  slv01; -- 40MHz serial frame clock
                                -- rising edge captured
    datain_even : IN  slv02; -- Serial input data (even bits)
    datain_odd  : IN  slv02; -- Serial input data (odd bits)
    filter_sel  : IN  slv01; -- Current filter coefficient set
    dataout     : OUT slv14; -- Deserialized data out
    datavalid   : OUT slv01; -- Signals new data word ready
    chansel     : OUT slv02; -- Current output channel
    chanerr     : OUT slv02; -- HIGH when error detected
    chanactive  : OUT slv02  -- HIGH when threshold exceeded
);
END IQ_2Channel;

```

```

ARCHITECTURE IQ_2Channel_Arch OF IQ_2Channel IS

```

```

    COMPONENT deserial

```

```

        PORT (

```

```

reset      : IN  slv01; -- Synchronous reset
clkkin     : IN  slv01; -- DDR clock (2 bits/period)
clkkin_z   : IN  slv01; -- Inverted DDR clock (2 bits/period)
serfrm_even : IN  slv01; -- Frame clock (14 bits/period),
              -- captured on falling edge clkkin
serfrm_odd  : IN  slv01; -- Frame clock (14 bits/period),
              -- captured on rising edge clkkin
datain_even : IN  slv01; -- Serial input data, captured
              -- on clkkin_z (even bits)
datain_odd  : IN  slv01; -- Serial input data,
              -- captured on clkkin (odd bits)
dataout     : OUT slv14; -- Deserialized data OUT
datavalid   : OUT slv01 -- Toggles when new word is ready
);
END COMPONENT;

```

```

COMPONENT Filt_2chan IS

```

```

PORT (
    rfd      : OUT slv01;
    rdy      : OUT slv01;
    clk      : IN  slv01;
    sclr     : IN  slv01;
    din      : IN  slv14;
    filter_sel : IN  STD_LOGIC_VECTOR ( 0 DOWNT0 0 );
    dout     : OUT slv14;
    data_valid : OUT slv01;
    chan_in  : OUT slv02;
    chan_out  : OUT slv02
);

```



```
END COMPONENT;
```

```
TYPE mixer_stage IS ARRAY (1 DOWNTO 0) OF slv02;
```

```
TYPE shiftreg8 IS ARRAY (1 DOWNTO 0) OF slv08;
```

```
TYPE channeldata IS ARRAY (1 DOWNTO 0) OF slv14;
```

```
SIGNAL rfd : slv01; -- Filter ready for data
SIGNAL filtdatatin : slv14; -- Filter data input
SIGNAL filt_i : channeldata; -- I data samples
SIGNAL filt_q : channeldata; -- Q data samples
SIGNAL deserdata : channeldata; -- Deserializer output
SIGNAL serdatavalid : slv02; -- Deserializer data valid
SIGNAL chanactive_shft : shiftreg8; -- Activity shift register
SIGNAL filtchansel : slv02; -- Filter channel selection
SIGNAL mix_select : mixer_stage; -- I/Q mix state
SIGNAL filter_sel_q : slv01;
SIGNAL filter_sel_qq : slv01;
SIGNAL iqactive_q : slv01;
SIGNAL iqactive_qq : slv01;
SIGNAL serdatavalid_q : slv02;
SIGNAL serdatavalid_qq : slv02;
SIGNAL serdatavalid_pulse : slv02;
SIGNAL adcdata : channeldata; -- Synchronized data
SIGNAL filtdata_update : slv02; -- Update filter input signals
```

```
BEGIN
```

```
-- 160MHz 4-channel filter
```

Filter1 : Filt_2chan

PORT MAP (

```
rfd           => rfd,
rdy           => open,
clk           => clk160mhz,
sclr         => reset,
dout         => dataout,
filter_sel(0) => filter_sel_qq,
din          => filtdatain,
data_valid   => datavalid,
chan_in(1 DOWNTO 0) => filtchansel,
chan_out(1 DOWNTO 0) => chansel
```

);

-- Filter coefficient selection synchronizer

PROCESS (clk160mhz)

BEGIN

IF rising_edge(clk160mhz) THEN

IF reset = '1' THEN

filter_sel_q <= '0';

filter_sel_qq <= '0';

iqactive_q <= '0';

iqactive_qq <= '0';

chanerr <= "00";

ELSE

iqactive_q <= iqactive;

iqactive_qq <= iqactive_q;

filter_sel_q <= filter_sel;

filter_sel_qq <= filter_sel_q;

```

        END IF;
    END IF;
END PROCESS;

-- Multiplex input to filter as requested
filtdatain <= filt_i(0) WHEN filtchansel = "00" ELSE -- Channel 0
             filt_q(0) WHEN filtchansel = "01" ELSE -- Channel 1
             filt_i(1) WHEN filtchansel = "10" ELSE -- Channel 2
             filt_q(1) WHEN filtchansel = "11";      -- Channel 3

-- Update each filter input channel in sequence
filtdata_update(0) <= '1' WHEN filtchansel = "01" ELSE '0';
filtdata_update(1) <= '1' WHEN filtchansel = "11" ELSE '0';

deserial_reg_gen: FOR i IN 1 DOWNTO 0 GENERATE

-- Deserializer
deserial_inst: deserial
    PORT MAP (
        reset      => reset_ser,
        clk_in     => serclk,
        clk_in_z   => serclk_z,
        serfrm_even => serfrm_even,
        serfrm_odd  => serfrm_odd,
        data_in_odd => data_in_odd(i),
        data_in_even => data_in_even(i),
        data_out    => deserdata(i),
        data_valid  => serdata_valid(i)
    );

```



```

-- Quadrature and channel administration process
PROCESS (clk160mhz)
BEGIN
    IF rising_edge(clk160mhz) THEN
        IF reset = '1' THEN -- Synchronous reset
            filt_i(i) <= (OTHERS => '0');
            filt_q(i) <= (OTHERS => '0');
            chanactive(i) <= '0';
            chanactive_shft(i) <= (OTHERS => '0');
            mix_select(i) <= "00";
        ELSIF filtdata_update(i) = '1' THEN
            -----
            -- Quadrature Processing:
            -- Sample Number -->
            --      0      1      2      3
            --      -----
            -- I |Data | 0 |Data_z| 0 |
            -- Q | 0 |Data | 0 |Data_z|
            --      -----

            IF iqactive_qq = '0' THEN
                filt_i(i) <= adcddata(i);
                filt_q(i) <= adcddata(i);
            ELSE
                CASE mix_select(i) IS
                    WHEN "00" =>
                        filt_i(i) <= adcddata(i);
                        filt_q(i) <= ZERO14;
                    WHEN "01" =>

```

```

        filt_i(i) <= ZERO14;
        filt_q(i) <= adcddata(i);
    WHEN "10" =>
        filt_i(i) <= NOT (adcddata(i)) + 1;
        filt_q(i) <= ZERO14;
    WHEN "11" =>
        filt_i(i) <= ZERO14;
        filt_q(i) <= NOT (adcddata(i)) + 1;
    WHEN OTHERS =>
        END CASE;
    END IF;

    mix_select(i) <= mix_select(i) + 1;

    -- Activity detection for signed input data
    IF (signed(adcddata(i)) >= threshold) OR
        (signed(adcddata(i)) <= -threshold) THEN
        chanactive_shft(i) <=
            chanactive_shft(i)(6 DOWNT0 0) & '1';
    ELSE
        chanactive_shft(i) <=
            chanactive_shft(i)(6 DOWNT0 0) & '0';
    END IF;

    chanactive(i) <= or_reduce(chanactive_shft(i));

    END IF;
END IF;
END PROCESS;

```

```

PROCESS(clk160mhz)
BEGIN
    IF rising_edge(clk160mhz) THEN
        IF reset = '1' THEN
            serdatavalid_q(i) <= '0';
            serdatavalid_qq(i) <= '0';
            serdatavalid_pulse(i) <= '0';
            adcddata(i) <= (OTHERS => '0');
        ELSE
            serdatavalid_q(i) <= serdatavalid(i);
            serdatavalid_qq(i) <= serdatavalid_q(i);
            serdatavalid_pulse(i) <= serdatavalid_qq(i)
                XOR serdatavalid_q(i);
            IF serdatavalid_pulse(i) = '1' THEN
                adcddata(i) <= deserdata(i);
            END IF;
        END IF;
    END IF;
END PROCESS;

END GENERATE deserial_reg_gen;

END IQ_2Channel_Arch;

```


This volume is the property of the University of Oklahoma, but the literary rights of the author are a separate property and must be respected. Passages must not be copied or closely paraphrased without the previous written consent of the author. If the reader obtains any assistance from this volume, he must give proper credit in his own work.

I grant the University of Oklahoma Libraries permission to make a copy of my thesis upon the request of individuals or libraries. This permission is granted with the understanding that a copy will be provided for research purposes only, and that requestors will be informed of these restrictions.

NAME _____

DATE _____

A library which borrows this thesis for use by its patrons is expected to secure the signature of each user.

This thesis by JOHN MEIER has been used by the following persons, whose signatures attest their acceptance of the above restrictions.

NAME AND ADDRESS

DATE