Design, Simulation, and Implementation of a Digital Quadrature Demodulator for a Stepped Frequency Radar

by

Michael K. Cope

A dissertation submitted to the Department of Electrical Engineering, University of Cape Town, in partial fulfillment of the requirements for the Degree

Master of Science in Engineering

University of Cape Town

January 2003

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the degree of Master of Science in Engineering at the University of Cape Town. It has not been submitted before for any degree or examination at any other university.

_____

Cape Town, 3 June 2003

Abstract

The scope of this thesis project is the design and implementation of a digital quadrature demodulator for a stepped frequency ground penetrating radar. This dissertation presents a theoretical model of the demodulator, simulations characterising the demodulator performance, as well as the design, construction, and measurement of the prototype demodulator.

The demodulator estimates the amplitude and phase of the intermediate frequency signal of a time-interleaved dual-channel heterodyne radar receiver. A demodulator model is developed from a survey of the relevant literature, paying particular attention to errors introduced in sampling. Simulations predict the demodulator performance in the radar system, suggesting coherent integration improves accuracy by reducing the effect of random sampling errors. The design of the prototype and characterisation of its performance are briefly reported.

Measurements confirmed that coherent integration increased the demodulator accuracy. Timing jitter was found to be the most significant cause of error, due to phase noise in the IF signal. The simulations predicted that the demodulator would not meet the specified performance; measurement determined the prototype's accuracy to be within specification, although the test signals were of higher quality than the expected radar IF signal.

Acknowledgements

I would like to thank my supervisor, Professor Michael Inggs, for his advice. I also wish to thank Dr Alan Langman for his guidance on technical matters. Thanks also to OpenFuel (Pty) Ltd and the SA National Defence Force for their financial support. I wish to thank my fellow-members of the Radar Remote Sensing Group for their friendship and assistance, in particular Thomas Bennett, Grant Carter, and Leon Alexander. Finally, I am grateful to my family for the love and support extended to me over the last 25 years, which made it all worthwhile.

Nomenclature

ADC - Analogue-to-digital converter

AHDL - Analogue hardware design language

CPLD - Complex programmable logic device

Dacs - Data capture system

dc - direct current

DNL - differential nonlinearity

FPGA - field programmable gate array

GPR - ground penetrating radar

I - in-phase

IC - integrated circuit

IF - intermediate frequency

JTAG - Joint Test Access Group

Q - quadrature

LO - local oscillator

LOS - local oscillator synthesizer

Opamp - operational amplifier

PC - personal computer

RF - radio frequency

rms - root mean square

RRI - ramp repetition interval

SFCW - stepped frequency continuous wave

SNR - signal-to-noise ratio

SRAM - static random access memory

TXS - transmit synthesizer

VHDL - Very High Speed Integrated Circuit Design Language

Table of Contents

List of Figures vii
List of Tables viii

1 Introduction 1
1.1 Summary 1
1.2 Ground penetrating radar research at UCT 1
1.3 Dissertation outline 2

2 Literature review: digital quadrature demodulation 4
2.1 Architecture and operation of SFCW radar systems 4
2.2 Digital quadrature demodulation 5
2.3 Modelling the digital quadrature demodulator 10
2.4 Summary 17

3 Simulating the digital quadrature demodulator 18
3.1 The digital quadrature demodulator model 18
3.2 Simulating the demodulator with individual error sources 21
3.3 Simulating the complete demodulator 27
3.4 Summary 32

4 Design and implementation of the prototype 33
4.1 Dacs in context of radar architecture 33
4.2 Statement of key user requirements 34
4.3 System design 35
4.4 Hardware design and implementation 35
4.5 Programmable logic design and implementation 38
4.6 Software design and implementation 42
4.7 Measuring the performance of the demodulator prototype 44
4.8 Summary 51

5 Summary 54
5.1 Analysis of results 54
5.2 Summary 56
5.3 Conclusions 57
5.4 Recommendations 57

A Simulation of the demodulator 59

A.1 Digital quadrature demodulator model 59
A.2 Simulation parameters 59

B Microcontroller software tools **61**
B.1 Software libraries and tools 61
B.2 Cross-compiling the software 61
B.3 Programming the flash memory 62

C Testing the hardware and software **63**
C.1 Hardware tests 63
C.2 Firmware tests 64
C.3 Software tests 65

Bibliography **66**

List of Figures

1 Inputs and outputs of the digital quadrature demodulator 1
2 Block diagram of the Dacs board 2
3 Stepped frequency continuous radar waveform 4
4 Architecture of the Venus SFCW radar system 5
5 Signal flow in the digital quadrature demodulator 6
6 Errors in the I and Q components translate into amplitude and phase errors 8
7 The clock edge associated with the sample s[0] is ambiguous 8
8 High-level block diagram of the demodulator 10
9 Aperture delay introduces an error into the sampled voltage 13
10 Block diagram of the demodulator model 18
11 Simulated rms amplitude error versus coherent integration factor for 14-bit quantizer 22
12 Simulated rms phase error versus coherent integration factor for 14-bit quantizer 22
13 Simulated rms amplitude error versus coherent integration factor for SNR=40 dB 23
14 Simulated rms phase error versus coherent integration factor for SNR = 40 dB 23
15 Simulated rms amplitude error versus coherent integration factor for SNR=80 dB 25
16 Simulated rms phase error versus coherent integration factor for SNR=80 dB 25
17 Simulated rms amplitude error versus coherent integration factor for 10-bit quantizer 26
18 Simulated rms phase error versus coherent integration factor for 10-bit quantizer 26
19 Simulated rms amplitude error versus N for $\sigma_j = 1x10^{-9}$ 28
20 Simulated rms phase error versus N for $\sigma_j = 1x10^{-9}$ 28
21 Simulated rms amplitude error; amplitude = 999 mV_{pk-pk} and SNR = 74 dB 30
22 Simulated rms phase error; amplitude = 999 mV_{pk-pk} and SNR = 74 dB 30
23 Simulated rms amplitude error; amplitude = 316 μV_{pk-pk} and SNR = 10 dB 31
24 Simulated rms phase error; amplitude = 316 μV_{pk-pk} and SNR = 10 dB 31
25 Architecture of the Venus radar system 33
26 Function blocks of the data capture system 36
27 Block diagram of the Dacs hardware 36
28 Block diagram of the VHDL code for the FPGA 39
29 Timing of accumulator write signals 40
30 Timing of the accumulator clear and memory write signals 40
31 State machine representation of the FPGA system controller 42
32 Software is partitioned between the AT91 microcontroller and a PC 43
33 Sequence of software cooperation between PC and microcontroller 43
34 Diagram of the test setup, showing clock frequency for subsampling inactive 45
35 Ratio of demodulator output amplitude to input signal amplitude 47
36 Difference in output amplitude between consecutive input amplitude steps 47
37 Difference in phase output step size between consecutive phase steps 48
38 Measured rms amplitude error versus coherent integration factor 50
39 Measured rms phase error versus coherent integration factor 50
40 Rms amplitude error versus coherent integration factor, for $P = 5$ 52
41 Rms phase error versus coherent integration factor, for $P = 5$ 52
42 Mean output amplitude of 256 ensembles, N=1 64

List of Tables

1 Threshold SNR and quantization noise floor against number of bits 24
2 Simulated rms errors for 40 dB SNR 27
3 Rms error values for input amplitude of 499 mV and timing jitter std. dev. of 3 ns 29
4 Rms error values for input amplitude of 158 uV and timing jitter std. dev. of 3 ns 32
5 Effect of coherent integration on amplitude error for different input amplitudes 49
6 Effect of subsampling on amplitude error 51
7 Effect of subsampling on phase error 53
8 Simulated performance compared to specifications 55
9 Measured performance compared to specifications 56

Chapter 1
Introduction

1.1 Summary

The scope of this MSc project is the design and implementation of a *digital quadrature demodulator* for a new ground penetrating radar, known as Venus, which is an upgrade of an existing radar, MercuryB [1]. The demodulator is one module of the data capture system, abbreviated to *Dacs*, which is also responsible for controlling the radar and communicating with external computers.

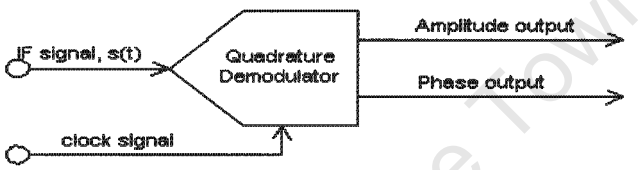

Figure 1. Inputs and outputs of the digital quadrature demodulator

A model of the quadrature demodulator was developed from the literature, and was simulated using the Monte Carlo technique. The simulations estimated the contribution of various error mechanisms and gauged the effectiveness of coherent integration in improving the accuracy of the demodulator output. Demodulator performance in the radar system was predicted by simulation.

A working prototype was produced; Figure 2 is a system block diagram. The prototype demodulator performance was tested, and found to satisfy the specifications, although the signal generators used in the tests limited the precision of the results.

The most significant conclusion is that phase noise in the radar receiver's IF signal limits the demodulator accuracy, which exceeds the specified error allowance.

1.2 Ground penetrating radar research at UCT

Ground penetrating radar (GPR) is a sensing technique that uses electromagnetic energy to produce an image of an underground target area [2]. The radar is usually in close contact with the earth's surface, looking into the ground, or it may be placed in a borehole or mineshaft. Ground penetrating radar is commonly used in landmine detection, mining, civil engineering, and archeology [2][3].

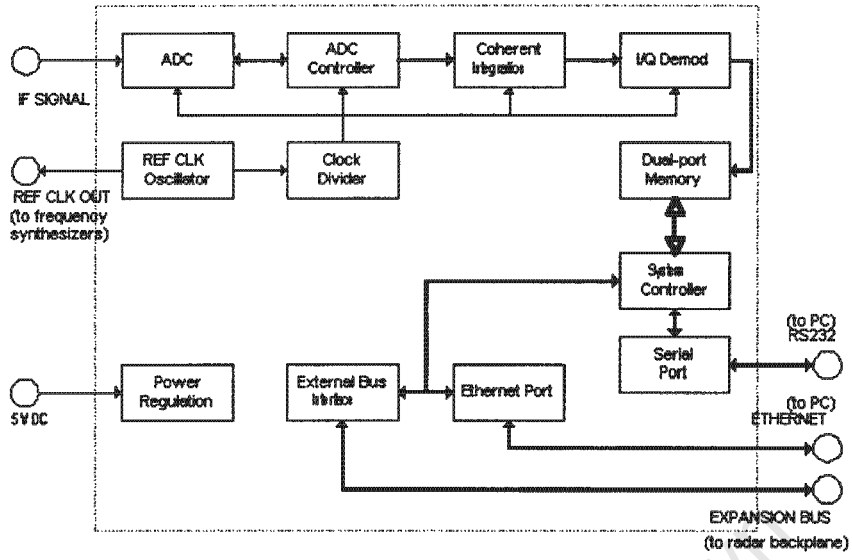

Figure 2. Block diagram of the Dacs board

The Radar Remote Sensing Group (RRSG) at the University of Cape Town began researching ground penetrating radar in 1988 [4], and has developed a radar system using a stepped frequency continuous wave architecture, housed in a portable enclosure [1][5]. The current version of this radar system is designated MercuryB, and is to be superceded by a new, upgraded design, Venus. The Dacs board, containing the digital quadrature demodulator, is one module of this radar.

The MercuryB radar hardware has a modular design, with synthesizer modules operating over the bands 200 to 1,600 MHz [1] and 800 to 3,200 MHz. A PC, communicating with the radar over a fibre-optic data link, acts as an operator interface, allowing for configuration of the the radar, measurement capture, and processing, display, and storage of the captured data. The radar data can be distributed over a network: observers at remote stations can receive the data stream for individual processing and interpretation [6].

The suitability of the MercuryB GPR system for various applications, such as surveying buried civil engineering structures and detection of non-metallic landmines has been examined in several field trials [1].

1.3 Dissertation outline

This dissertation describes two related work areas: the design, construction, and testing of the new Dacs circuit board, and the development and simulation of a theoretical model of the digital quadrature demodulator.

Chapter 2 is the literature review and shows the top-down development of a *behavioural model* for the digital quadrature demodulator. The demodulator is placed in the context of the Venus stepped frequency continuous wave radar system, emphasizing the measurement of relative phase using two time-interleaved channels. A simple architecture for a digital quadrature demodulator that exploits the SFCW waveform is shown, then the implementation of the demodulator using real components is examined. Coherent integration is examined as a technique to improve the demodulator accuracy.

Chapter 3 discusses the simulation of the demodulator model using the Monte Carlo technique. The significant contributors to demodulator error are examined individually and their effect on the output error estimated; the potential improvement available through coherent integration is assessed. The model parameters are then set to match the prototype demodulator and the actual radar signals. Simulations using this model predict the performance of the demodulator in the radar system.

Chapter 4 is an overview of the design and development of the circuit board, programmable logic, and software, and begins with a brief requirements analysis. The chapter then shows the conceptual design of the data capture system, and concisely explains the design of the hardware, programmable logic, and software. The results of performance tests on the demodulator complete the chapter - the imprecision of the signal generators limited the value of these tests.

The final chapter presents an analysis of the simulations and measurements of the prototype demodulator performance. Simulations of the demodulator in the radar system show that phase noise in the IF signal causes the demodulator error to exceed the specified limits. The measurements demonstrate achieved performance within the tolerable limits, although the test signals had lower phase noise than the radar IF signal. A brief summary of the main results is followed by the conclusions and recommendations for future work.

Appendix A shows the derivations of some of the simulation parameters, and lists the parameters used for each series of simulations. Appendix B explains the tools used to compile the microcontroller software and program the flash memory on the prototype. Appendix C describes the tests performed on the prototype hardware, software, and programmable logic.

Chapter 2
Literature review: digital quadrature demodulation

This chapter is an overview of the literature relevant to the design and implementation of a digital quadrature demodulator for a stepped frequency continuous wave radar system, and focusses on digitisation of the IF signal and the attendant imperfections.

2.1 Architecture and operation of SFCW radar systems

2.1.1 Stepped frequency continuous wave radar

The stepped frequency waveform consists of a sequence of discrete, usually equally-spaced frequencies, each transmitted for a fixed time, known as the dwell time, T_{dwell} [7]. The ramp repetition interval is simply the number of frequency steps n multiplied by the dwell time.

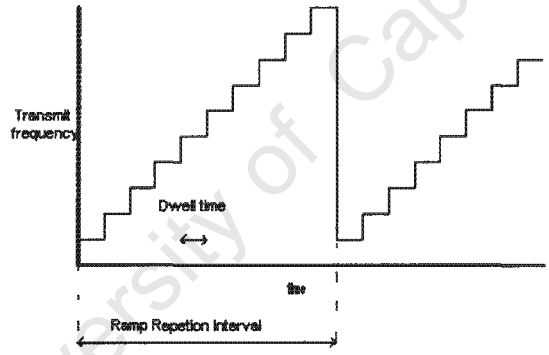

Figure 3. Stepped frequency continuous radar waveform

One method for the stepped frequency processor to construct a synthetic range profile is by performing an inverse Fast Fourier Transform on the received waveform [8]. The synthetic range profile is similar to the range display produced by conventional pulse radars.

2.1.2 The Mercury and Venus SFCW ground penetrating radar systems

The Venus radar was under construction at the time of writing, so this dissertation will refer to its predecessor, Mercury, as the same principles apply to both radars. The Mercury ground penetrating radar system detects the SFCW pulse using a heterodyne receiver [5]. The received

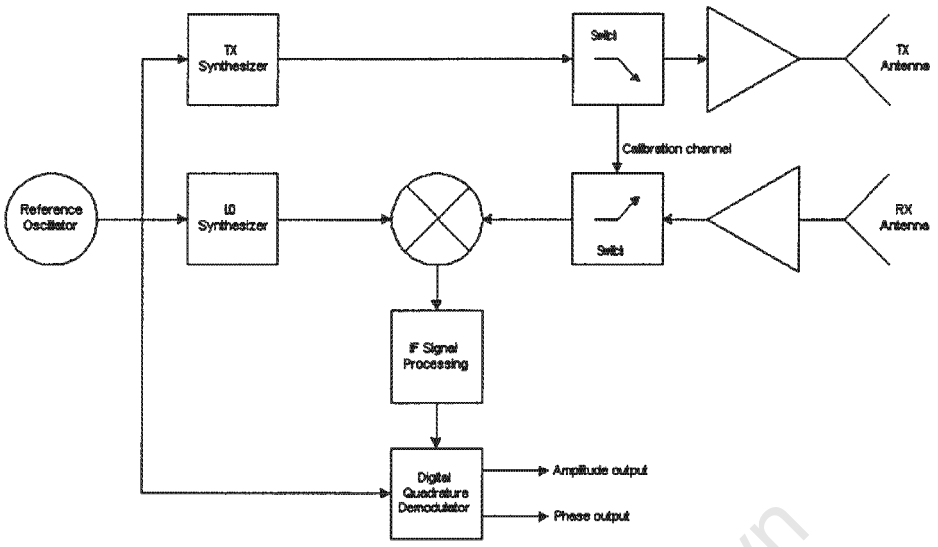

Figure 4. Architecture of the Venus SFCW radar system

signal is downconverted by mixing it with the local oscillator (LO) signal, producing an intermediate frequency (IF) signal at 2 MHz, with a bandwidth of 100 kHz [5].

The transmit frequency of the Mercury radar ranges from 200 to 1,600 MHz, or 800 to 3,200 MHz, depending on which synthesizer modules have been installed [1]. The RF components used in the transmitter and receiver do not have linear amplitude and phase responses over the operating bandwidth, corrupting the synthetic range profile [5]. The problem is solved by employing a time-interleaved heterodyne architecture: there is an extra channel between the transmitter and receiver. The signal through this extra channel contains the same corruptions as the signal through the transmit-receive channel, and is used as a calibration signal, allowing the amplitude and phase nonlinearities in the received signal to be removed [5]. The time interleaving is implemented by switches, as shown in Figure 4.

2.2 Digital quadrature demodulation

Traditional demodulation schemes derive the in-phase and quadrature (I and Q) components by demodulation to baseband by two oscillator signals separated by $\frac{\pi}{2}$ rad (i.e. in quadrature); each IF signal is then filtered individually and sampled by a separate ADC [9]. The components of the two channels are not perfectly matched and introduce an *imbalance* between the channels, which can cause phase errors as high as 2 to 3 degrees [10]. In direct digital demodulation, however, the downconverted carrier signal is digitised at a rate exceeding the Nyquist rate, after which the

digital signal is demodulated by two digital filters. The channel imbalances no longer exist, since the I and Q channel samples pass through the same analogue components [11].

2.2.1 Architecture of a simple digital quadrature demodulator

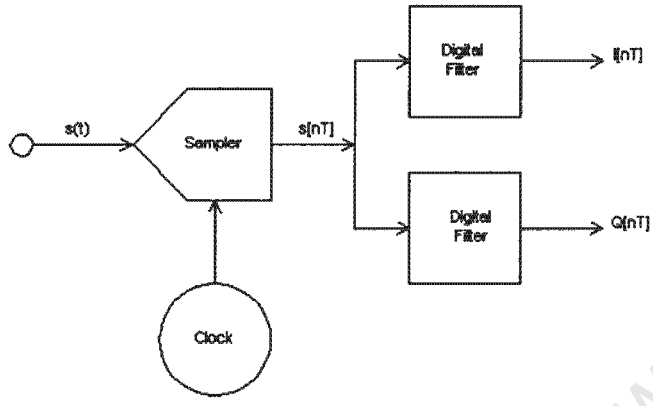

Figure 5. Signal flow in the digital quadrature demodulator

A simple system for digital quadrature demodulation of a modulated carrier signal is shown in Figure 5 [12]. The carrier signal has been mixed down to a suitable intermediate frequency, $\omega_{IF} = 2\pi f_{IF}$. We write the intermediate-frequency signal as

$$s(t) = A(t)\cos(\omega_{IF}t + \phi) \quad (1)$$

$$= I(t).\cos(\omega_{IF}t) - Q(t).\sin(\omega_{IF}t) \quad (2)$$

where

$$I(t) = A(t).\cos(\phi t) \quad (3)$$

$$Q(t) = A(t).\sin(\phi t) \quad (4)$$

are the in-phase and quadrature components respectively [12]. We now digitise the IF signal $s(t)$ by sampling at $f_s = 4f_{IF}$. Considering one period, this gives us a digital signal, $s[n]$, which is $s(t)$ sampled at

$$t = \frac{n\pi}{2} \quad ; n = 0, 1, 2, 3 \quad (5)$$

Digital filters process the sample sequence $s[n]$ to give the the I and Q components [12]:

$$I(t) = s[0] - s[2] \tag{6}$$

$$Q(t) = s[3] - s[1] \tag{7}$$

The amplitude and phase of the carrier signal are then just

$$A(t) = \sqrt{I(t)^2 + Q(t)^2} \tag{8}$$

$$\phi(t) = \arctan\left(\frac{Q(t)}{I(t)}\right) \tag{9}$$

Since the amplitude $A(t)$ and phase offset $\phi(t)$ of the IF signal are constant during each frequency step in the SFCW waveform, we will denote them as the constants A and ϕ for convenience.

2.2.1.1 Sampling errors introduce inaccuracies into the demodulator outputs

The sample sequence $s[n]$ in a real system contains errors since the electronic components employed are imperfect. We therefore introduce time-varying error terms $\epsilon_I(t)$ and $\epsilon_Q(t)$ into the in-phase and quadrature components derived from these samples, writing the components as $I'(t) = I + \epsilon_I(t)$ and $Q'(t) = Q + \epsilon_Q(t)$ respectively, where I and Q are the ideal values, so that the demodulator amplitude and phase outputs are given by

$$A'(t) = \sqrt{(I + \epsilon_I(t))^2 + (Q + \epsilon_Q(t))^2} \tag{10}$$

$$\phi'(t) = \arctan\left(\frac{Q + \epsilon_Q(t)}{I + \epsilon_I(t)}\right) \tag{11}$$

and now contain errors that change with time; Figure 6 depicts this as a vector diagram. It is easily seen that the significance of the errors depends on the amplitude of the input signal: error terms ϵ_I and ϵ_Q of a given magnitude will cause greater inaccuracies for smaller signal amplitudes. It is also worth noting that the sensitivity of the phase output depends on the input phase, in particular on the ratios $\frac{\epsilon_I}{I}$ and $\frac{\epsilon_Q}{Q}$.

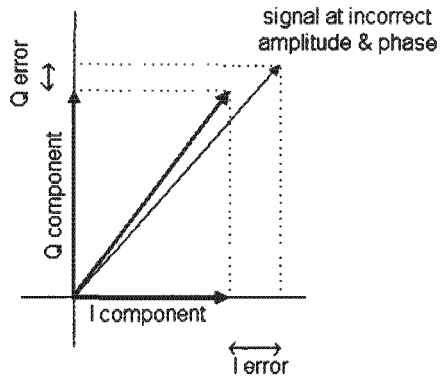

Figure 6. Errors in the I and Q components translate into amplitude and phase errors

2.2.2 Measuring the phase of the IF signal

The Mercury SFCW radar system has a 16 MHz oscillator which provides the reference for both the transmit and local-oscillator frequency synthesizers (abbreviated TXS and LOS respectively) and the quadrature demodulator. The IF signal, produced by mixing the received signal with the LO signal, is at a center frequency of 2 MHz. Quadrature demodulation requires a sampling frequency f_s to be four times the input frequency f_{in}: the reference oscillator is divided down to produce the 8 MHz sampling clock.

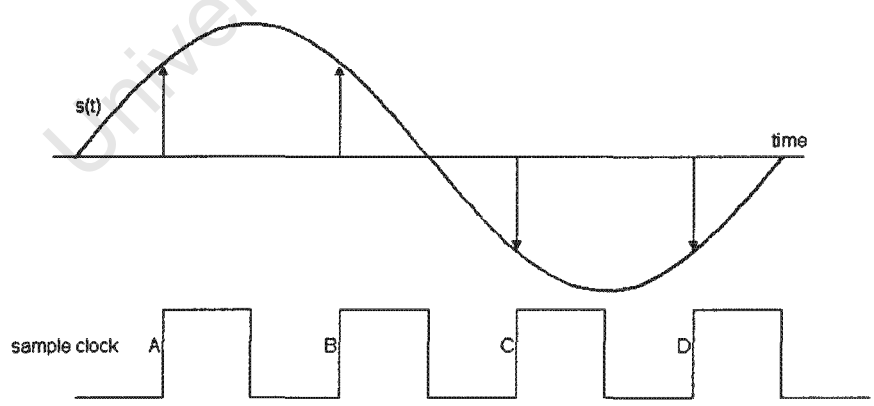

Figure 7. The clock edge associated with the sample s[0] is ambiguous

There are four clock edges per input signal period (Figure 7), and the choice of reference edge

is arbitrary, but the output phase depends on the reference edge choice. The problem is avoided by a fortunate result of the interleaved architecture: measuring the *relative* phase difference between the calibration and transmit-receive channels means the arbitrary reference edge is the same for both channels. Although the absolute phase values may change by a multiple of $\frac{\pi}{2}$ rad after a power-cycle, the relative phase difference remains constant.

2.2.3 Coherent integration improves demodulator performance

Coherent integration is the averaging of an ensemble of observations of a noisy signal with the same expected value, and reduces the *noise power* [13]. The variance of the noise signal, σ_1^2, is reduced by the factor N when successive observations are uncorrelated:

$$\sigma_N^2 = \frac{\sigma_1^2}{N} \tag{12}$$

where σ_N^2 is the variance of the noise signal after coherent integration over ensemble length N [13]. Since the signal-to-noise ratio of a sinusoid in the presence of white noise is simply the signal variance over the noise variance [14], the signal-to-noise ratio is also improved by the factor N. The standard deviation of the noise, or how far the *noise voltage* deviates from the mean, is only reduced by a factor of $\sqrt{N}$.

This "one-upon-N" rule does not hold for time averages in systems where the sampling rate is high enough to avoid aliasing, since the noise in successive observations is then correlated to some extent [13] - the variance σ_N^2 tends towards σ_1^2 as the sample-to-sample correlation of the noise increases.

2.2.4 Subsampling reduces the sampling rate

All four samples required for quadrature demodulation are usually taken in one period of the input waveform at a sample rate of $f_s = 4f_{in}$. Subsampling arises from the simple observation that the sampling points occur at identical phases in each period, and will therefore have identical values if the input signal is periodic, allowing the use of a lower sampling frequency and a cheaper ADC [1]; the penalty is that the samples must be taken over several periods of the input waveform. The actual sampling frequency required for subsampling is

$$f_{actual} = \frac{4f_{in}}{P} \tag{13}$$

where $P = 3, 5, 7, \ldots$ is the number of periods.

The prototype demodulator was constructed with an ADC capable of sampling at 8 MSa/s, so

subsampling is superfluous. The demodulator model is independent of the *sampling* frequency, and predicts that subsampling offers no performance improvement; this claim was verified empirically (Section 4.7.3.4.)

2.3 Modelling the digital quadrature demodulator

The digital quadrature demodulator is implemented using analogue and mixed-signal circuitry to digitise the input and digital logic to process the samples. Numbers in a sampled-data system are represented with finite precision, which can introduce errors due to truncation or rounding [15]; the word length in this application is sufficient that these errors are negligible in comparison to errors introduced in sampling; only multiplication (by a power of 2) is performed on the FPGA; the 32-bit results are converted to floating-point representation on a PC and then divided.

2.3.1 Top-level model of the demodulator

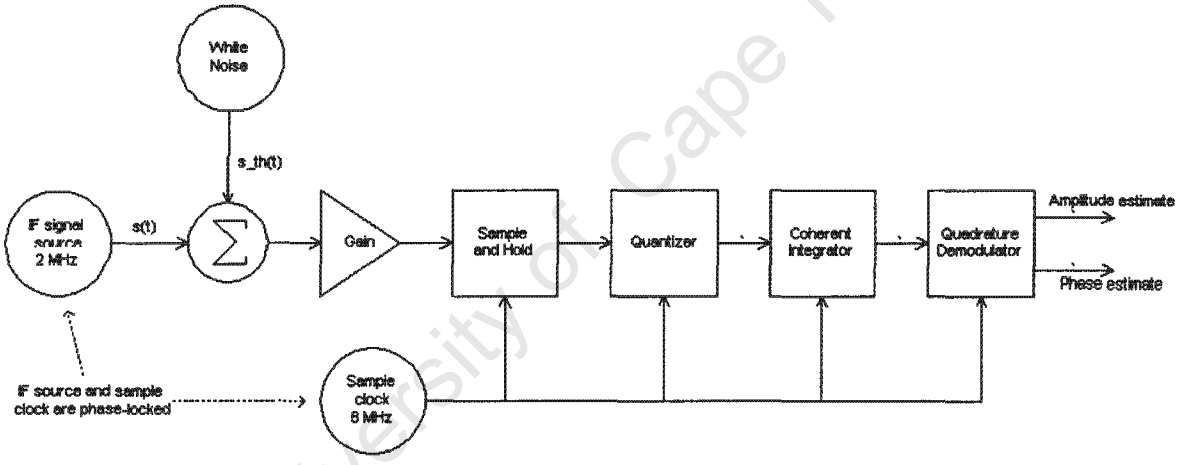

Figure 8. High-level block diagram of the demodulator

The demodulator structure is shown as a block diagram in Figure 8. The model for each analogue or mixed-signal component is expanded below; the coherent integration and quadrature demodulation operations are implemented digitally.

2.3.1.1 Approaches to modelling the analogue-to-digital converter in the demodulator

There is an extremely large body of literature on analogue-to-digital conversion, stretching as far back as the realisation that pulse-code modulation was in effect digitisation [16]. An important category within this literature is concerned with modelling ADCs and their error mechanisms. Aimed at chip *designers*, these papers generally produce analytical descriptions of

the converter. As ADCs become more and more integrated, including large amounts of digital circuitry, modelling and simulating the chip in analogue simulation programmes becomes very time-consuming [17]. One solution is to model the *behaviour* of the components in the time domain, using statistical descriptions for random phenomena. ADCs have been modelled and simulated in languages from Matlab [18], [19], [20] to AHDL and even VHDL [17], a language designed for describing digital logic.

Correction of systematic converter errors by processing the ADC output data is also a well-covered topic. A common approach is to model the ADC as an ideal converter with an error source described by a dynamic error function [21]. An error correction algorithm or table is programmed using parameters obtained from calibration, and then used to remove errors from the ADC output. No *a priori* knowledge of the converter structure is assumed.

The primary purpose of the following sections is to investigate the error mechanisms and how they affect the demodulator. This leads to a simplified model using a combination of analytical and behavioural descriptions in the time domain. The model is also independent of the converter architecture, useful since little information was available for the THS1408 selected for the prototype. Although modelling the demodulator and converter at an abstract level sacrifices some accuracy in predicting performance, the omitted errors are relatively small - the most significant sampling errors are caused by aperture jitter and thermal noise [22], which are included in the model.

2.3.2 Thermal noise in the input signal

Real devices introduce thermal noise into signals. The received radar signal is already noisy, and processing by the downconverter introduces more thermal noise [5].

2.3.2.1 Modelling thermal noise

Thermal noise $s_{th}(t)$ is modelled as a band-limited, Gaussian noise process having flat power spectral density, zero mean, and constant variance σ_{th}^2 - that is, as white noise. This is an adequate model for many significant noise sources in RF circuits [23]. The noise signal is superimposed on the ideal sinusoidal signal; the signal quality is indicated by the signal-to-noise ratio: $SNR_{th} = 10\log(\frac{P_s}{P_n})$ in decibels, where P_s and P_n are the signal and noise powers respectively [14].

2.3.3 RF transformer

The single-ended IF signal is converted into a differential signal by an RF transformer. We

simplify the demodulator model by representing the transformer as a single-ended component, modelling it as an ideal opamp representing the insertion loss (i.e. the gain is less than unity.)

2.3.4 Differential amplifier with programmable gain

The signal chain contains a differential amplifier with programmable gain, but this analysis will assume a constant gain setting. Opamp gain depends on frequency, but the input signal is at a constant frequency so the differential amplifier should have constant gain.

2.3.4.1 Model of the amplifier

Opamps generally have internal frequency compensation networks to ensure stability over the whole range of input frequencies, although these limit the rate at which the output can slew [24]; the minimum slew rate needed for a sinusoid of amplitude A and frequency f_{in} is $r_{slew} = 2\pi A f_{in}$ [24], giving a minimum requirement for this application of 25V/μs. Although the THS1408 datasheet does not specify the slew rate, a survey of recent PGA datasheets suggests that the opamp slew rate will not be a serious limitation - the slew rate of the AD603 PGA is at least 275V/μs [25], and Maloberti et al. use a slew rate of over 1000V/μs in their simulations [20].

The amplifier adds thermal noise to its output signal - this converter model refers all thermal noise to the input signal for simplicity, so the amplifier will be considered noiseless. Other errors are assumed to be either neglible or - especially gain and offset errors - removed through compensation [26][27]. The differential amplifier is represented as a single-ended component with fixed gain G_a and offset v_{os} set to 0.

2.3.5 Sample-and-hold circuit

Analogue-to-digital converters sample the input at regular intervals and then quantize the sampled voltage (supposedly held constant by the SNH.)

2.3.5.1 Model of the sample-and-hold circuit

The sample-and-hold model described here is a vastly simplified version, consisting of an ideal sample-and-hold with aperture jitter [28]. It excludes classic sources of dynamic error such as droop, limited slew rate, and feedthrough [29]. The limited slew rate of sample-and-hold circuits causes errors that worsen with increasing input frequency, but this is no longer a significant limitation in recent ADC designs [30].

Aperture delay and aperture jitter In an ideal sample-and-hold circuit the input will be sampled exactly on the rising edge of the clock. But practical converters have a small delay from the clock edge to the time the input is completely disconnected from the hold capacitor.

The average delay is called aperture delay, or aperture time [31]. Power supply induced noise and variation of digital thresholds by thermal noise cause aperture jitter, which is cycle-to-cycle variation in the aperture delay; power supply noise can be effectively eliminated by judicious filtering [28]. Aperture delay and aperture jitter cause the sampling instant to deviate from the ideal and thereby introduce an amplitude error into the sampled voltage (Figure 9.)

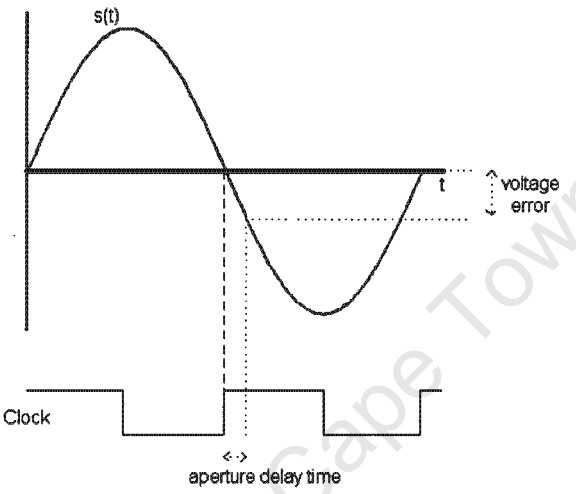

Figure 9. Aperture delay introduces an error into the sampled voltage

Modelling aperture delay and jitter Aperture delay $\epsilon_{aj}(t)$ is modelled as a stationary Gaussian random process with the mean corresponding to the aperture delay and a small variance corresponding to the aperture jitter. The delay is effective from the rising edge of the clock signal. The input signal is sampled at instants $t = nT_s + \epsilon_{aj}(t)|_{t=nT_s}$ where $n = 0, 1, 2, ...$and T_s is the clock period. The aperture delay is of course the same when the calibration and transmit/receive channels of the radar are selected; we are interested in measuring *relative* phase difference between the channels, and so set the mean delay to zero.

2.3.5.2 The effect of aperture delay and jitter on sampling

Kobayashi et. al. consider the case where a sinusoidal input $s(t) = A\sin(2\pi f_{in}t)$ is sampled at time $t = nT_s + \epsilon_{aj}(t)$, where T_s is the sampling period and $\epsilon_{aj}(t)$ is the aperture jitter, assumed to follow a Gaussian distribution with zero mean and a standard deviation of σ_{aj}. The amplitude error due to aperture jitter is then

$$\Delta V_{aj} \approx \epsilon_{aj}(t)\frac{ds(t)}{dt}|_{t=nT_s} \tag{14}$$

which has a maximum when the sampling instant occurs at a zero-crossing, where the input slope is maximum, giving

$$\Delta V_{aj} \approx \epsilon_{aj}(t)A2\pi f_{in} \tag{15}$$

where it is assumed that $2\pi f_{in}\sigma_{aj} << 1$ [32] [33].

2.3.6 Clock oscillator and timing jitter

An oscillator is an autonomous system producing a periodic output, $s_{osc}(t)$: an ideal oscillator's output depends only on time, but random events (noise) in real oscillators perturb the oscillator output, causing phase and amplitude deviations [34]. A sampled-data system using a square-wave oscillator output as its clock signal experiences these deviations as timing jitter: the transitions of the sampling clock occur at times deviating from the ideal. Timing jitter is exacerbated by mechanisms other than oscillator phase noise when the clock signal is passed through digital logic. In particular, the finite rise-time of logic signals and unstable digital logic thresholds worsen the jitter experienced at the sample-and-hold clock input [35].

2.3.6.1 Modelling timing jitter

One can write the oscillator output, including the phase and amplitude disturbances, as

$$s_{osc}(t + \alpha(t)) + y(t) \tag{16}$$

where $\alpha(t)$ is the phase deviation, and $y(t)$ is the amplitude deviation [36].

The amplitude deviations $y(t)$ are small and transient, and are compensated for by amplitude limiting mechanisms in the oscillator itself [37] - these deviations are therefore negligible and will be ignored.

The phase deviation $\alpha(t)$ is the most important effect of oscillator noise; $\alpha(t)$ will, in general, keep increasing with time as the effect of the phase perturbations is cumulative [36]. Further, $\alpha(t)$ is a Gaussian random variable with constant mean and a variance that increases linearly with time [36, p. 3]. The jitter variance of a square-wave oscillator therefore increases linearly with time as the oscillator output deviates further from the ideal the longer the oscillator runs [36]. In the quadrature demodulator, however, the input signal and the sampling clock are derived from the same oscillator, so the input and clock signals will only experience small, short-term phase

shifts, but will maintain a steady phase relationship over time. The delay between the received signal and the local oscillator signal is very short for ground penetrating radar, so the phase noise of the received signal is correlated with that of the LO signal [1]. Timing jitter is dependent on the target range: the received signal is delayed more for more distant targets, and the oscillator phase noise is integrated over a longer period of time, giving rise to worse jitter. The dependence of jitter on range was not explicitly examined.

Timing jitter can be derived from oscillator phase noise Timing jitter is the effect of oscillator phase noise observed in the time domain; the phase noise for a given oscillator is usually specified by a plot of single-sideband noise spectral density $L(f)$ in the frequency domain. Noise spectral density can be integrated over a particular frequency range to obtain the phase noise power over the chosen bandwidth, from which the rms timing jitter can be calculated by treating it as the equivalent sideband power of a phase modulator [38].

Model used for simulations Timing jitter $\epsilon_{tj}(t)$ was modelled as a Gaussian random variable with zero mean and constant variance, a reasonable approximation since the input and clock signals are derived from the same source. The standard deviation σ_j of the timing jitter was specified in units of time - typical values, gauged from the simulations performed by Maloberti et al., are standard deviations ranging from 1 to 10ns [20]. The effects of logic gate jitter are assumed to be allowed for by choosing a jitter variance larger than than that of the oscillator alone.

2.3.6.2 The effect of timing jitter on sampling

Timing jitter introduces an amplitude error into the sampled value in exactly the same way as aperture jitter - the error can be calculated by substituting ϵ_{tj} for ϵ_{aj} in (15).

2.3.7 Quantizer introduces quantization noise

The quantizer converts the analogue sample-and-hold output into a digital code: each code represents a unique reconstruction level. Since the quantizer has a fixed number of reconstruction levels and the input signal has an infinite number of possible values, the quantized output is necessarily an imperfect copy of the input signal.

2.3.7.1 Model of a mid-tread uniform quantizer

The quantizer is modelled as an ideal quantizer, which can be defined as consisting of a set of contiguous intervals or cells on a partition of the real line. Each cell has a reproduction level or code, placed in the middle of the cell. The quantizer output is then simply the reproduction level or code closest to the input [16]. The transfer function is a staircase function when plotted on

Cartesian axes, and can be described mathematically as

$$q(x) = \left\{ \begin{array}{c} \left(\frac{M}{2} - 1\right)\Delta; \quad \left(\frac{M}{2} - 1.5\right)\Delta \leq x \\ k\Delta; \quad (k - 0.5)\,\Delta \leq x < (k + 0.5)\,; \\ \left(-\frac{M}{2}\right)\Delta; \quad x < \left(-\frac{M}{2} + 0.5\right)\Delta \end{array} \right\} \tag{17}$$

where M is the number of reconstruction levels, and Δ is the width of the reconstruction cells, or the bin width, and the index k has the range $k = \left(-\frac{M}{2} + 1\right), ..., \left(\frac{M}{2} - 2\right)$ [39].

Quantizers in real ADCs are clocked at the sampling frequency, so that the output code changes only on clock edges. In practice, the output code is represented as a binary word of width b bits, so the number of levels is $M = 2^b$. A quantizer is *uniform* if the quantization levels are equally spaced, and is known as a mid-tread quantizer if a reconstruction level exists at exactly 0 volts, and each reconstruction level is a multiple of Δ from the x-axis.

2.3.7.2 Quantization noise

The quantizer error ϵ_q is defined as $\epsilon_q = Q(x) - x$. The quantizer output is expressed as the sum of the input signal and the quantizer noise. Since the quantizer input is generally a sequence of samples $x(n)$, the quantizer noise process is $\epsilon_q[nT_s] = Q(x[nT_s]) - x[nT_s]$, and is often characterised statistically [39].

Quantization error in synchronous sampling causes spurious signals The nominal ratio of signal to quantization noise is

$$SNR_{nom} = 6.02b + 1.76 \quad (dB) \tag{18}$$

where b is the number of quantizer bits [40]. Coherent sampling, or synchronous sampling, occurs when the ratio of the sampling frequency to the input frequency is an integer larger than 1 (i.e. $f_s = mf_{in}$; m a natural number >1) [40]. The quantization error due to synchronous sampling is deterministic, and (18) no longer holds. The deterministic quantization noise causes spurious signals, observable in FFT plots of sampled data [40]; the quantization noise process is distinctly periodic, and can be described by a Fourier series for simple inputs [39]. The third harmonic will usually have the highest amplitude since it is the odd harmonic with the lowest order, especially when the input amplitude is low [41].

2.3.7.3 Dithering improves data converter performance

Dithering is the addition of a random signal to the input of an ADC and can be achieved by simply averaging successive quantizer output values if the input signal already contains random noise- this is known as nonsubtractive dithering [42]. Dithering is implemented in this design

by coherent integration, which produces four averaged sample values for every N periods. The random signal is provided by the thermal noise in the input signal; Gaussian noise causes the quantization error to be approximately uniform even when the quantization step size Δ is as large as the standard deviation of the noise [43].

Dithering reduces quantization error Dither can, in theory, reduce quantization error to a negligible quantity and minimize coherent signal spurs caused by the nonlinear quantizer transfer function, at the cost of reduced converter bandwidth [44]. In practice, however, dither usefully removes small-scale (i.e. localised) linearity errors, but large-scale errors must be removed by a look-up table or compensation algorithm [45].

Dithering reduces differential nonlinearity error The cell width Δ varies slightly from code to code in practical converters, a property expressed by the differential linearity, the difference between each cell's actual width and the ideal value [46]; some codes are good and some codes are bad [30]. Differential nonlinearity causes spurious signals, which can be reduced by dithering, which effectively randomizes the errors so that the averaged output contains less error than if the input exercised only a bad code [30].

2.4 Summary

SFCW radars produce a series of discrete tones, measuring the amplitude and phase of the received signal at each frequency step. The Mercury and Venus SFCW radars use a time-interleaved heterodyne architecture to compensate for errors in the RF components; this also removes the ambiguity in the sampled phase. A digital quadrature demodulator measures the radar IF signal amplitude and phase by sampling at twice the Nyquist rate and filtering the samples to produce estimates for the in-phase and quadrature components, avoiding the imbalances which plague analogue demodulation schemes.

The IF signal samples contain errors, causing inaccuracy in the demodulator outputs. Coherent integration promises to reduce the effect of random errors, at the expense of increased conversion time.

A model of the demodulator was produced for use in computer simulation of the demodulator performance. The model shows that thermal noise and clock jitter are the most significant sources of random errors in sampling; systematic errors can be compensated for, and are also attenuated by dithering, implemented in this case by coherent integration.

Chapter 3
Simulating the digital quadrature demodulator

This chapter presents a behavioural model of the complete demodulator that includes analogue, mixed-signal, and digital components, and describes the simulations predicting the harmful effect of component errors on the demodulator outputs. These simulations assisted in defining the system design parameters (sampling rate, coherent integration factor.)

3.1 The digital quadrature demodulator model

3.1.1 Components of the model

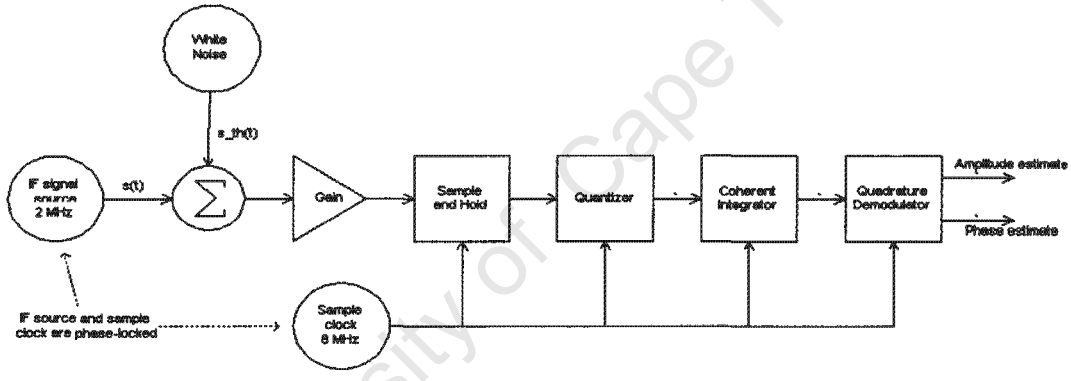

Figure 10. Block diagram of the demodulator model

The demodulator was simulated in the discrete-time domain, with timestep T_s equal to the clock period, 125 ns; the equations are written as functions of discrete time t_n, the timestep variable n ranging from 0 to $N_{\max}$.

3.1.1.1 Oscillators

The 2 MHz oscillator is an ideal oscillator, with output

$$s[t_n] = A\cos(2\pi f_{in} n T_s + \phi)\ ; n = 0, 1, 2, ..N_{\max}$$

where A and ϕ are the amplitude and phase parameters, and the frequency f_{in} is 2 MHz.

The 8 MHz clock oscillator is modelled as an ideal oscillator with random timing jitter, assumed to be Gaussian. Following the form used by Awad [47], we let the rising edges of the

clock signal occur at

$$t_{CLK}[t_n] = nT_s + \epsilon_j[n] \; ; n = 0, 1, 2, ..N_{\max}$$

where the timing jitter $\epsilon_j[n]$ is the jitter random variable associated with the nth sampling instant; $\epsilon_j[n]$ are identical and independent Gaussian random variables with zero mean and a variance of σ_j^2 [47].

3.1.1.2 Gaussian white noise

The white noise signal $s_{th}[t_n]$ is a Gaussian random process with zero mean and a variance of σ_{th}^2. The noise power is set directly by the signal-to-noise ratio, SNR_{th}, using the IF signal power as numerator (see Appendix A.)

3.1.1.3 Gain

The transformer and amplifier were combined and modelled as a single gain, G_a:

$$s_a[t_n] = G_a(s[t_n] + s_{th}[t_n])$$

3.1.1.4 Sample-and-hold

The sample-and-hold was an ideal sample-and-hold, and the output was calculated as the IF signal value at the sample time *including jitter*, including the noise voltage:

$$\begin{aligned} s_{snh}[t_n] &= s_a[nT_s + \epsilon_j[n]] \\ &= G_a(s[nT_s + \epsilon_j[n]] + s_{th}[t_n]) \; ; n = 0, 1, 2, ..N_{\max} \end{aligned}$$

3.1.1.5 Quantizer

The quantizer was as an ideal midtread quantizer $Q(x)$, with 2^B reconstruction levels; the number of bits B and the quantizer range were specified as parameters. The quantizer output was

$$x_q[t_n] = Q(\; s_{snh}[t_n] \;)$$

3.1.1.6 Coherent integrator

The coherent integrator is simply nonsubtractive dithering implemented individually on the sample streams corresponding to each of the four sampling phases, with the coherent integration factor N as the ensemble length. The accumulator outputs after N periods are therefore

$$
\begin{aligned}
x_{acc0}[i] &= (x_q[iNT_s] + x_q[(iN+4)T_s] + x_q[(iN+8)T_s] + ...)/N \\
x_{acc1}[i] &= (x_q[(iN+1)T_s] + x_q[(iN+5)T_s] + x_q[(iN+9)T_s] + ...)/N \\
x_{acc2}[i] &= (x_q[(iN+2)T_s] + x_q[(iN+6)T_s] + x_q[(iN+10)T_s] + ...)/N \\
x_{acc3}[i] &= (x_q[(iN+3)T_s] + x_q[(iN+7)T_s] + x_q[(iN+11)T_s] + ...)/N
\end{aligned}
$$

where the index $i = 0, 1, 2..L-1$ is the number of completed sets of N sinewave periods, and L is a parameter specifying how many contiguous measurements (each requiring N periods) should be taken.

3.1.1.7 Quadrature demodulator

The coherent integrator outputs are averaged sample values of the input signal at phases $\Phi = \frac{k\pi}{2}$; $k = 0, 1, 2, 3$. The in-phase and quadrature components are calculated as an intermediate step (the voltages are referred to the demodulator input by dividing by the amplifier gain):

$$I'[i] = \frac{x_{acc0}[i] - x_{acc2}[i]}{2G_a} \tag{19}$$

$$Q'[i] = \frac{x_{acc3}[i] - x_{acc1}[i]}{2G_a} \tag{20}$$

The amplitude and phase outputs are then:

$$A'[i] = \sqrt{I'[i]^2 + Q'[i]^2} \tag{21}$$

$$\phi'[i] = \arctan\left(\frac{Q'[i]}{I'[i]}\right) \tag{22}$$

A complete simulation therefore produces L estimates for the amplitude and phase, respectively $A'[i]$ and $\phi'[i]$, for each combination of parameters.

3.1.1.8 Error signals

The simulation aims to investigate the *errors* in the demodulator outputs, so the amplitude and phase errors were recorded to file, for statistical analysis in Matlab. The error signals were calculated as

$$
\begin{aligned}
\epsilon_A[i] &= A'[i] - A \\
\epsilon_\phi[i] &= \phi'[i] - \phi
\end{aligned}
$$

3.1.2 Simulation in the time domain using Python

The time-domain behaviour of the demodulator model was simulated on computer using Monte Carlo techniques in Python, an open-source, interpreted language [48]. Each simulation run was specified by supplying a set of values for each parameter. The parameters were varied in sequence, such that L experiments were performed for each parameter combination. The errors show some dependency on the input phase, so more than one value for ϕ was used; the mean of the rms error over the phases was taken, reducing the effect of input phase on the results.

3.2 Simulating the demodulator with individual error sources

3.2.1 Simulating thermal noise

Thermal noise was simulated in two phases: the first phase covered a wide set of parameters but used only one value for input phase ϕ to reduce computation time; the second phase examined a narrower set of parameters in more detail, using 11 different ϕ values (see Section A 2.1.)

3.2.1.1 Simulation results for a single phase value

Simulated rms amplitude and phase errors are shown in Figures 11 and 12 for a 14-bit quantizer. Coherent integration reduces the amplitude and phase errors by a constant ratio at lower signal-to-noise ratios; the error tends to an horizontal asymptote (typically for $N > 64$) at a threshold SNR; and coherent integration has no effect at the highest signal-to-noise ratios. Random thermal noise and deterministic quantization noise respond differently to averaging: coherent integration reduces the error contribution of white noise, but, at as the SNR rises, the noise voltage becomes much smaller than the quantizer step size and dithering has no effect; the output error is now due to quantization alone and will not be removed by averaging.

Coherent integration starts to show diminishing returns at a threshold SNR roughly corresponding to the theoretical noise floor due to quantization noise, as per (18). The relationship is illustrated in Table 1. This happens when the additive noise starts to become too small to cause dithering; the noise standard deviation, σ_{th}, is about one-quarter of the quantizer step size at the threshold.

3.2.1.2 Simulation results for several phase values

The second phase of the simulations used a larger set of input signal phase ϕ values, resulting in a larger data set in which the effects of input phase could be averaged out.

Figures 13 and 14 show the simulated rms amplitude and phase error versus coherent

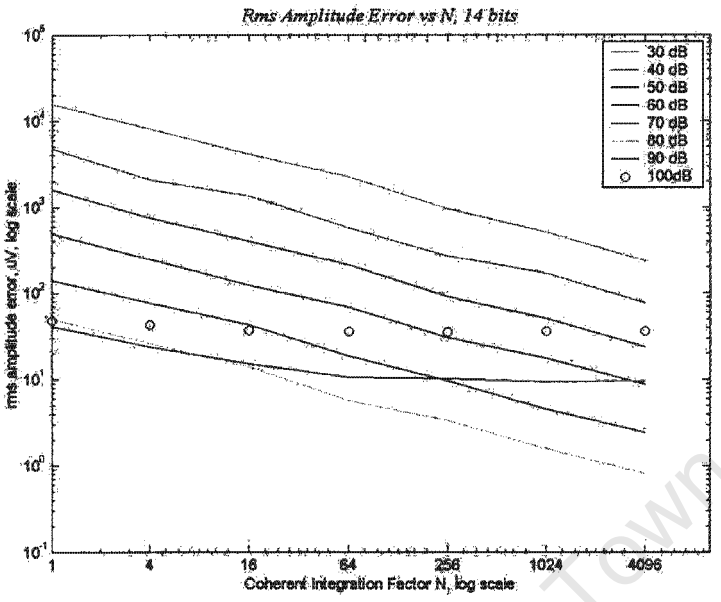

Figure 11. Simulated rms amplitude error versus coherent integration factor for 14-bit quantizer

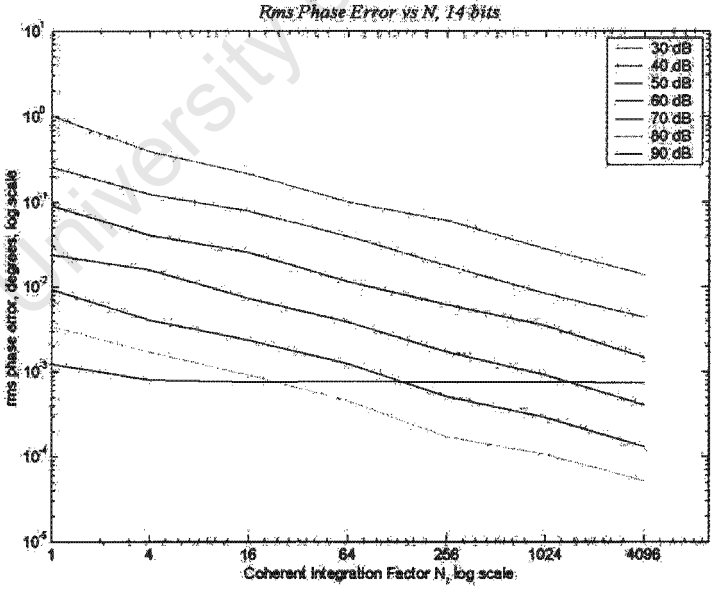

Figure 12. Simulated rms phase error versus coherent integration factor for 14-bit quantizer

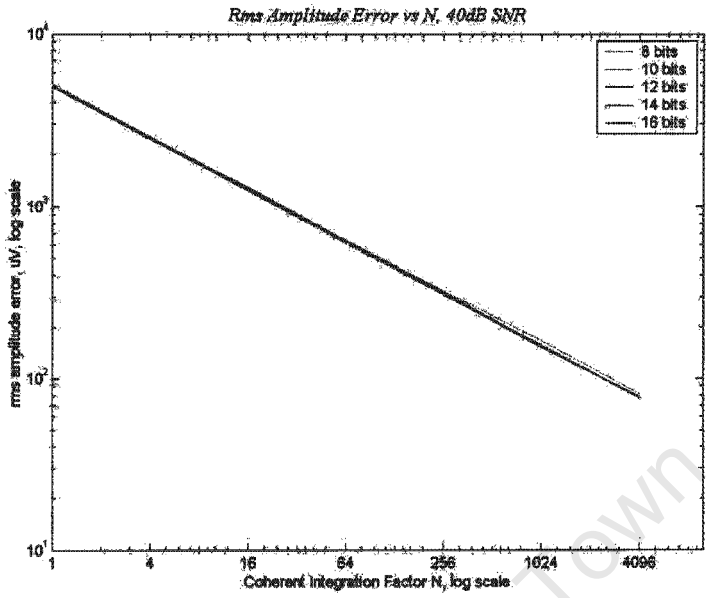

Figure 13. Simulated rms amplitude error versus coherent integration factor for SNR=40 dB

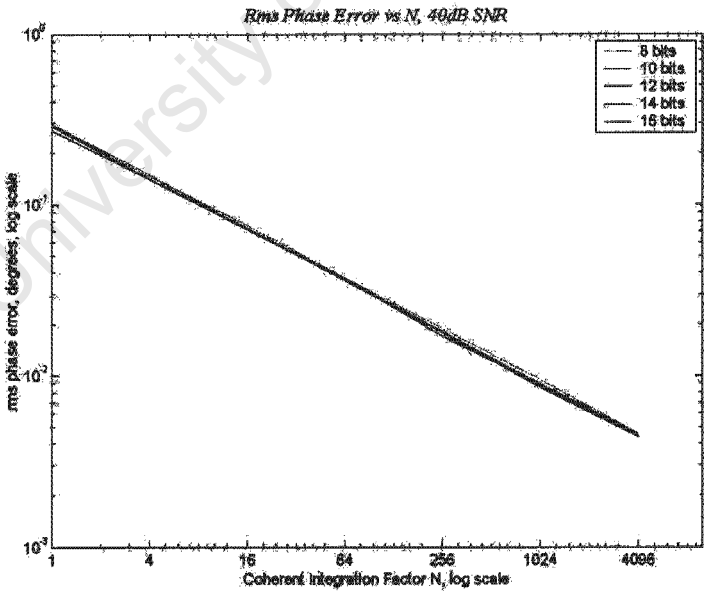

Figure 14. Simulated rms phase error versus coherent integration factor for SNR = 40 dB

Quantizer Bits	Quant. Noise Floor	Threshold SNR
8	-50 dB	50 dB
10	-62 dB	60 dB
12	-74 dB	≈75 dB
14	-86 dB	90 dB
16	-98 dB	100 dB

Table 1. Threshold SNR and quantization noise floor against number of bits

integration factor N for an input signal-to-noise ratio of 40 dB. The error is independent of the quantizer step size, indicating that the error is caused by thermal noise.

The rms amplitude and phase error, ϵ_A and ϵ_ϕ respectively, maintain a constant ratio of improvement as N increases logarithmically. The reduction in *standard deviation* of the additive noise predicted by (12) is $\frac{\sigma_1}{\sigma_2} = \sqrt{\frac{N_2}{N_1}}$, where σ_1 and σ_2 are at N_1 and N_2 respectively. The simulation results show $\frac{\epsilon_1}{\epsilon_2} \approx \sqrt{\frac{N_2}{N_1}}$, where the two rms error values, ϵ_1 and ϵ_2 are at N_1 and N_2 respectively (see Table 2). The root mean square of the variance of the amplitude and phase error ensembles also approximates the relationship predicted by (12), which is that the ratio of the two variances should equal to N.

Figures 15 and 16 show the simulated rms amplitude and phase errors for an input signal-to-noise ratio of 80 dB. Coherent integration provides no improvement for 8-, 10-, and 12-bit quantizers, indicating the predominance of quantization noise. However, the errors for 14- and 16-bit quantizers are very similar, suggesting that random noise is here the deciding factor. This is confirmed by the constant ratio of improvement due to coherent integration. The additive noise here has a standard deviation of 71μV, while the 14- and 16-bit quantizers have a maximum quantization error less than 61μV. The results for 14 and 16 bits have the property that two rms error values, ϵ_1 and ϵ_2 at N_1 and N_2 respectively, have the ratio $\frac{\epsilon_1}{\epsilon_2} \approx \sqrt{\frac{N_2}{N_1}}$.

3.2.2 Simulating timing jitter

Figures 17 and 18 are log-log plots of the rms amplitude and phase error signals for a 10-bit quantizer, with separate curves for each σ_j value. The results for 12-, 14-, and 16-bit quantizers are similar. The severity of the errors is notable.

The most striking feature is the different effect of coherent integration on the amplitude and phase errors. The amplitude error tends towards an asymptote as N increases, especially for larger σ_j values. The phase error, however, maintains a constant ratio of improvement as N rises. The amplitude and phase are calculated from the same samples, so the difference in the error signals response to coherent integration must be explained by the demodulator equations. The

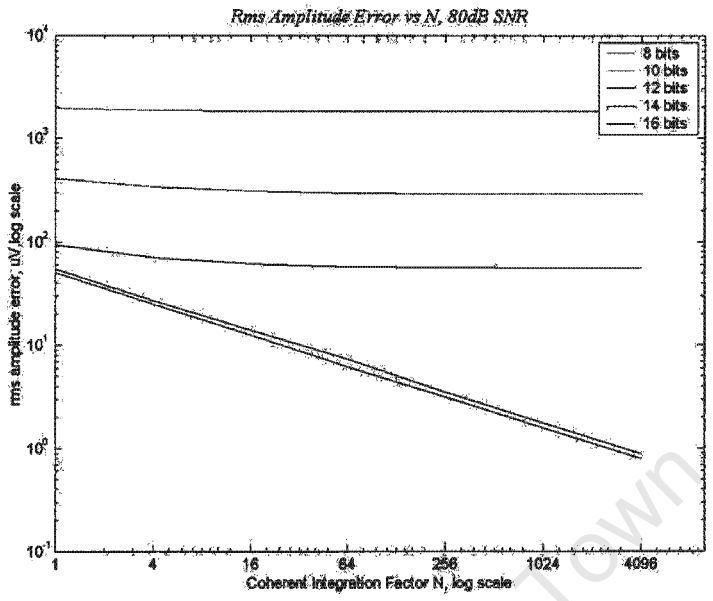

Figure 15. Simulated rms amplitude error versus coherent integration factor for SNR=80 dB

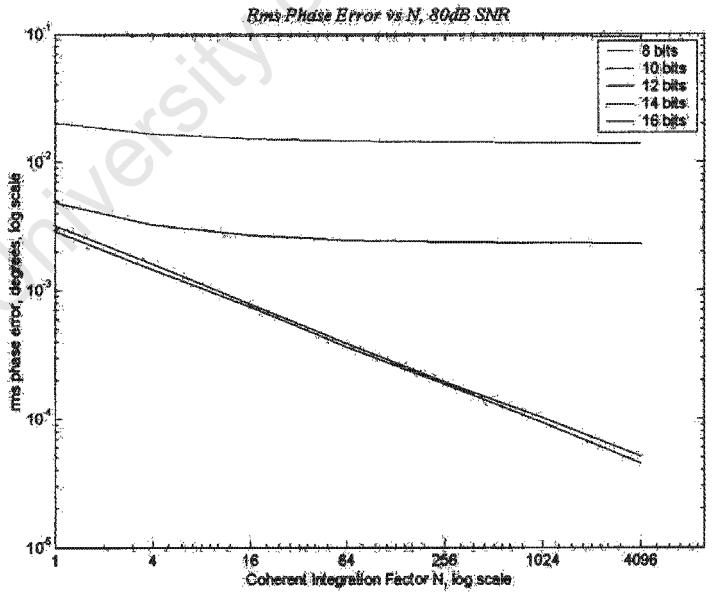

Figure 16. Simulated rms phase error versus coherent integration factor for SNR=80 dB

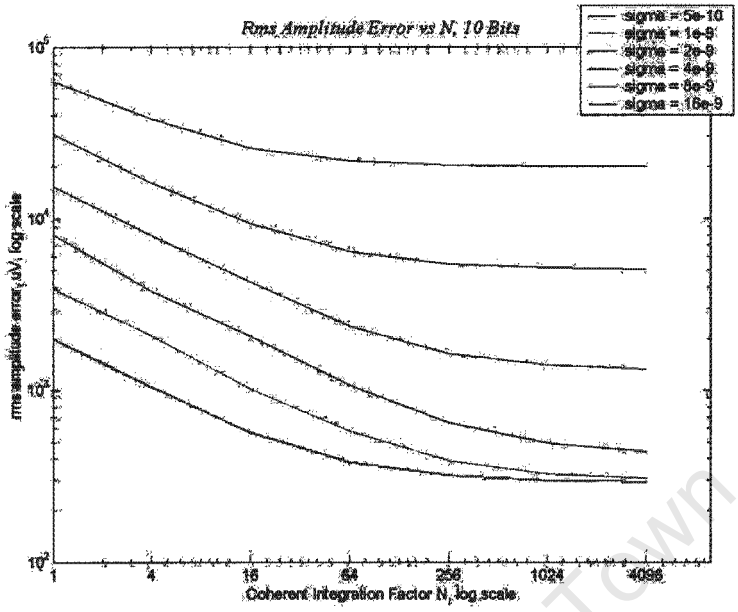

Figure 17. Simulated rms amplitude error versus coherent integration factor for 10-bit quantizer

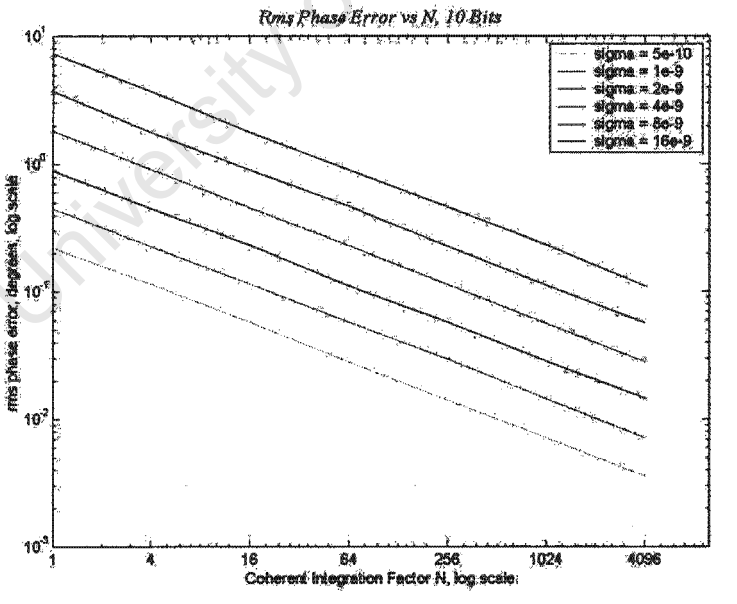

Figure 18. Simulated rms phase error versus coherent integration factor for 10-bit quantizer

B	$\frac{\epsilon_{A,N=1}}{\epsilon_{A,N=4096}}$	$\frac{\epsilon_{\phi,N=1}}{\epsilon_{\phi,N=4096}}$	$\frac{var.of\ \epsilon_{A,N=1}}{var.of\ \epsilon_{A,N=4096}}$	$\frac{var.of\ \epsilon_{\phi,N=1}}{var.of\ \epsilon_{\phi,N=4096}}$
8	63.3	64.3	4017	4107
10	63.8	65.3	4055	4273
12	64.4	65.8	4096	4302
14	63.7	66.0	4097	4404
16	64.0	59.7	4092	3540

Table 2. Simulated rms errors for 40 dB SNR

phase output is less sensitive at large input amplitudes.

The rms amplitude and phase errors are proportional to the timing jitter (the relationship is clearer for larger N), clearly indicating that the error is caused by timing jitter, although the effects of quantization can be seen for low σ_j values. Plotting the results for different quantizers with constant σ_j, as in Figures 19 and 20, shows that the rms amplitude error is independent of quantizer step size, except at higher N in this case - the error is completely independent of quantizer step size for larger timing jitter. The phase error is the same for all quantizers.

3.3 Simulating the complete demodulator

The simulations described above were performed with the quantizer number-of-bits parameter ranging from 8 to 16; unfortunately, a converter capable of sampling at 8 MSa/s with 16-bit precision was not available commercially. The 14-bit converter used in the prototype system has a dynamic range exceeding the specified 70 dB [49]. The following simulations aimed to investigate the performance of the prototype demodulator in the radar system. The simulation parameters are therefore set to match the prototype Dacs board.

3.3.1 Simulation parameters

3.3.1.1 Dynamic range

The demodulator should have a minimum dynamic range of 70 dB; this translates into an input voltage ranging from 1 V_{pk-pk} to 316.2 μV_{pk-pk}. The smallest input has an amplitude of only a few quantizer steps (Δ is 61 μV when referred to the transformer input.) Simulations were performed for the maximum and minimum input amplitudes.

3.3.1.2 Signal-to-noise ratio

The maximum signal-to-noise ratio at the transformer input was chosen as 80 dB, based on the radar receiver performance [5]. The thermal noise standard deviation was only σ_{th} =1.12 μV, much smaller than the quantizer step size (referred to transformer input.) The signal-to-noise ratio with the smallest input was then 10 dB. The simulation was also performed for a noise

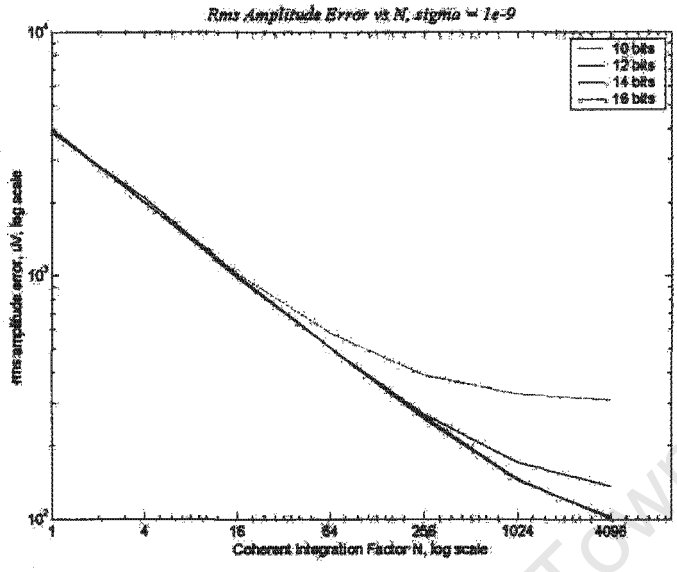

Figure 19. Simulated rms amplitude error versus N for $\sigma_j = 1x10^{-9}$

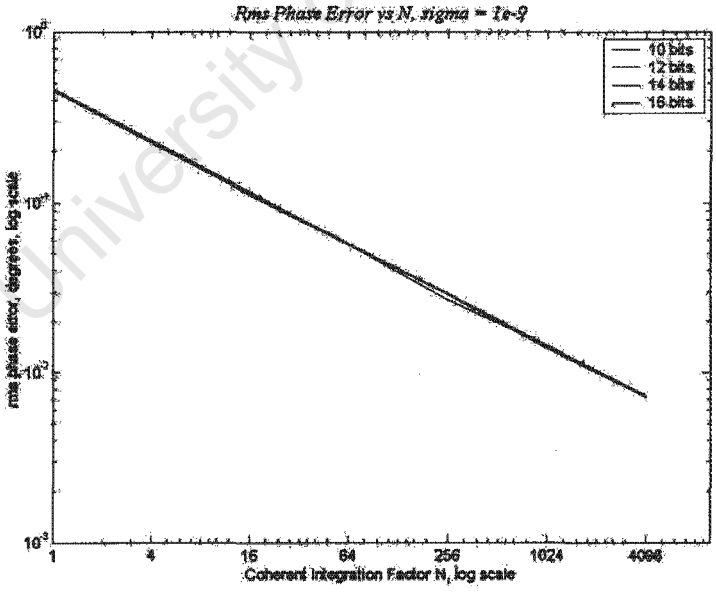

Figure 20. Simulated rms phase error versus N for $\sigma_j = 1x10^{-9}$

standard deviation of 70.5 μV_{pk-pk}, slightly larger than the quantizer step size; the signal-to-noise ratio was 74 dB with the maximum input and just 4 dB with the minimum input.

3.3.1.3 Timing jitter

The frequency synthesizers used to generate the transmit and local oscillator signals have a worst-case rms phase noise of 2.2° [5]. Assuming the same instability in the IF signal, i.e. 2.2°, the rms phase noise in the 2 MHz input signal is 3.3ns. The Dacs prototype contains a 16 MHz Vectron TO-330 temperature-compensated crystal oscillator; integrating the phase noise over the 10 kHz bandwidth for which the manufacturer specifies it, we find that the rms timing jitter is less than 60 ps (Appendix A.) We consider phase noise to be effectively timing jitter, so the simulations were performed with a timing jitter parameter of σ_j = 2 ns, 3 ns, and 4 ns - the simulation results should bound the achieved performance.

3.3.2 Simulation results

The errors are nearly identical for both signal-to-noise ratios at the maximum input amplitude (see Table 3.) Coherent integration reduces the rms amplitude error towards an asymptote, more pronounced for larger timing jitter; this behaviour indicates that timing jitter is the most significant contributor of error - even at N=1024, the rms amplitude error is several times larger than the standard deviation of the thermal noise. This is not surprising, as (15) predicts a maximum voltage error of 12.5 mV when the input signal is sampled with timing jitter ϵ_{tj} of 2ns; this is orders of magnitude greater than the magnitude of the thermal noise. The phase error also shows a clear dependence on the timing jitter, but is more reduced by coherent integration. Table 3 shows some of the results for A=999 mV_{pk-pk} and σ_{tj}=3 ns.

SNR	$\frac{\epsilon_{A,N=1}}{\epsilon_{A,N=1024}}$	$\frac{\epsilon_{\phi,N=1}}{\epsilon_{\phi,N=1024}}$	$\epsilon_{A,N=1}$	$\epsilon_{A,N=1024}$	$\epsilon_{\phi,N=1}$	$\epsilon_{\phi,N=1024}$
74 dB	14.8	32.8	5.9 mV	400 μV	1.4°	0.04°
80 dB	14.6	32.8	5.8 mV	396 μV	1.4°	0.04°

Table 3. Rms error values for input amplitude of 499 mV and timing jitter std. dev. of 3 ns

In contrast, the rms error at the minimum signal amplitude depends strongly on signal-to-noise ratio, the errors for a SNR of 4 dB being about twice the magnitude of the errors for a SNR of 10 dB. The errors are the same for all three timing jitter values, in accordance with the prediction from (15) that the sample error is about 8 μV when the input signal is sampled with timing jitter ϵ_{tj} of 4ns - this is much smaller than the quantizer step size. Coherent integration improves both the rms amplitude and phase error by approximately 32 as N increases from 1 to 1024, matching

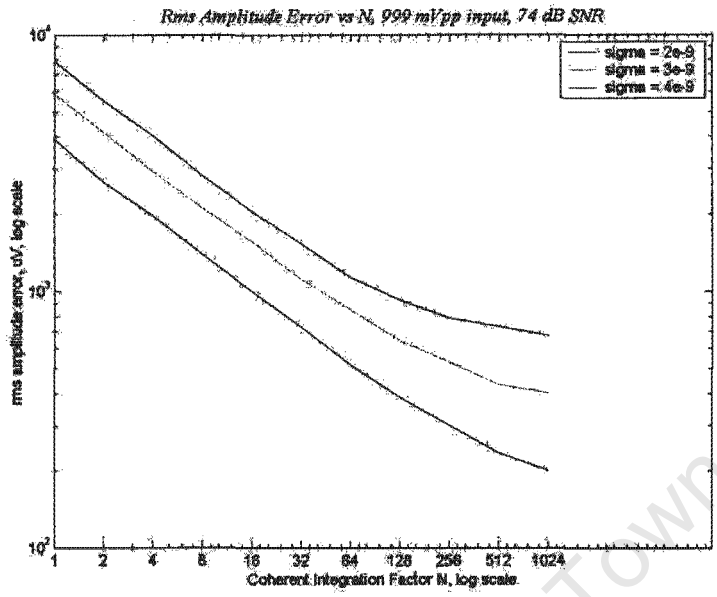

Figure 21. Simulated rms amplitude error; amplitude = 999 mV_{pk-pk} and SNR = 74 dB

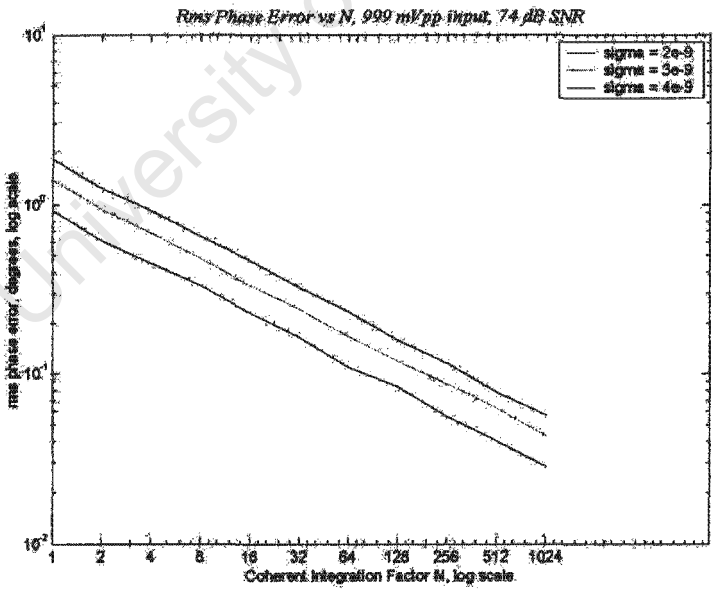

Figure 22. Simulated rms phase error; amplitude = 999 mV_{pk-pk} and SNR = 74 dB

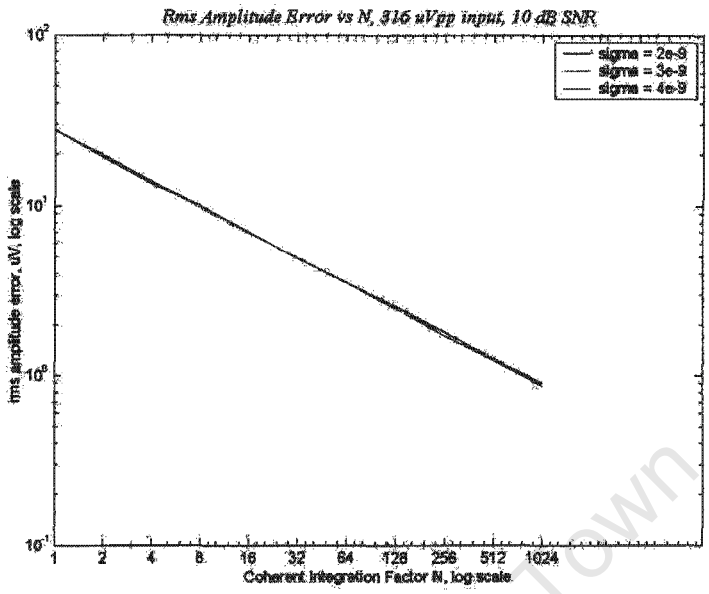

Figure 23. Simulated rms amplitude error; amplitude = 316 μV_{pk-pk} and SNR = 10 dB

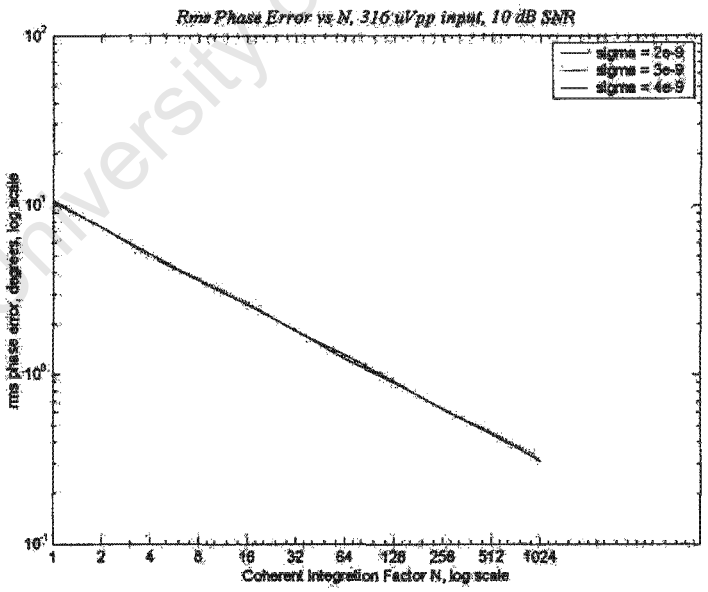

Figure 24. Simulated rms phase error; amplitude = 316 μV_{pk-pk} and SNR = 10 dB

SNR	$\frac{\epsilon_{A,N=1}}{\epsilon_{A,N=1024}}$	$\frac{\epsilon_{\phi,N=1}}{\epsilon_{\phi,N=1024}}$	$\epsilon_{A,N=1}$	$\epsilon_{A,N=1024}$	$\epsilon_{\phi,N=1}$	$\epsilon_{\phi,N=1024}$
4 dB	31.7	37.1	52 μV	1.6 μV	21°	0.6°
10 dB	31.9	32.4	28 μV	1 μV	10°	0.3°

Table 4. Rms error values for input amplitude of 158 uV and timing jitter std. dev. of 3 ns

the theoretical improvement due to dithering; the dithering is effective even for the lower noise voltage (10 dB SNR.) Table 4 shows some of the results for A=316 μV_{pk-pk} and σ_{tj}=3 ns. The rms phase error is very large without coherent integration, as much as 21°, illustrating the sensitivity of the phase output when the input amplitude is only a few Δ.

3.4 Summary

Computer simulations were performed to determine the relationship between sampling errors and the accuracy of the digital quadrature demodulator output and the effectiveness of coherent integration in improving the accuracy. The discrete-time simulations were performed in Python and the results were analysed using Matlab.

The first series of simulations examined the effects of thermal noise and timing jitter in isolation. Coherent integration was found to reduce the output errors due to thermal noise, provided the noise standard deviation was at least an eighth of the quantizer step size. The reduction corresponded well to that predicted by (12). Timing jitter caused more severe errors, and coherent integration failed to reduce the amplitude error by the factor predicted as N increased. The amplitude and phase errors, moreover, were independent of the quantizer step size for jitter with standard deviation greater than 1 ns. The amplitude and phase errors were proportional to the timing jitter, as expected.

Simulation of the prototype demodulator driven with the radar IF and clock signals was intended to predict whether the required performance would be achieved. Timing jitter caused the predicted performance to fall short of the specification at the maximum input amplitude; the phase error exceeded the tolerance at the lowest input amplitude due to the relatively large quantizer step size.

The link between simulations of the demodulator performance and the prototype system has already been established in the last simulation series, which matched the simulation parameters to the prototype. The following chapter describes the design and implementation of the hardware, programmable logic, and software used to implement the demodulator, and describes the measurement of the prototype performance.

Chapter 4
Design and implementation of the prototype

This chapter briefly describes the design and implementation of the Dacs circuit board and software. The Dacs board is placed in the context of the Venus radar system, and a brief requirements analysis is then elaborated. This is followed by a discussion of the design and implementation of the hardware, programmable logic, and software, which emphasizes the significant features. Finally, the performance of the digital quadrature demodulator is measured.

4.1 Dacs in context of radar architecture

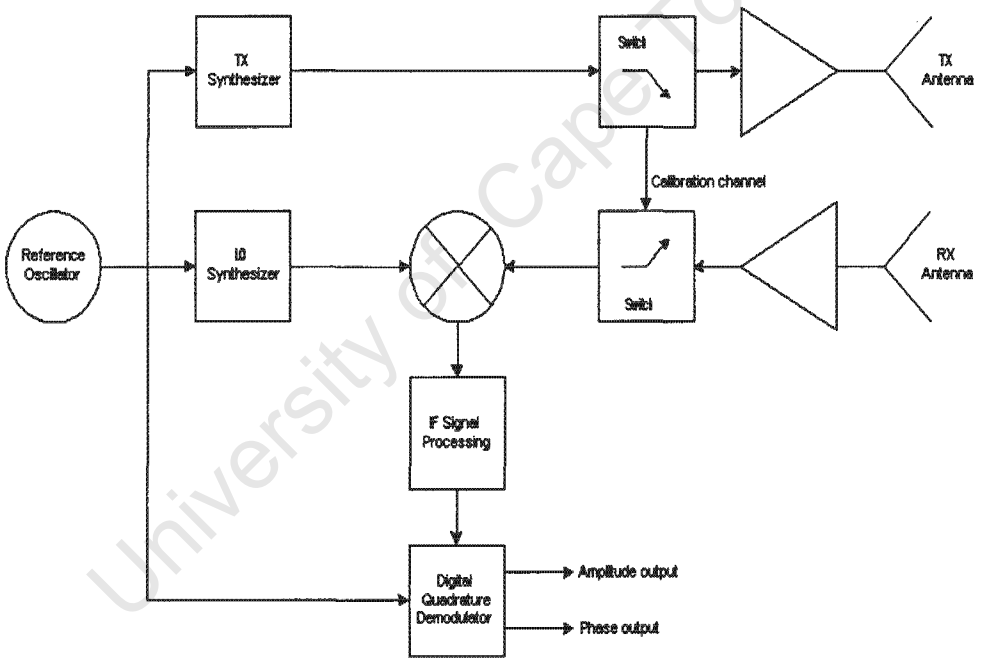

Figure 25. Architecture of the Venus radar system

The data capture system (Dacs) is one module of the Venus ground penetrating radar system. The Venus GPR uses a stepped frequency continuous wave architecture, described in more detail in Chapter 2. Most of the Venus radar's component systems have a single function. The Dacs unit, however, integrates several important functions onto one circuit card:

(1) *Digital quadrature demodulation.* The IF signal is digitised by synchronous sampling, and

the sample sequence is then processed by special algorithms. These suppress some of the noise in the digitised signal and perform digital quadrature demodulation to extract the input signal's amplitude and phase.

(2) *Radar control.* The Dacs controls the operation of the entire radar system by setting parameters of other systems in the radar, such as the frequency synthesizers and the downconverter. The Dacs also directs their operation.

(3) *Communication.* An operator controls the radar using a computer connected to the Dacs board; the computer sends commands to the Dacs and receives status information and captured data.

4.2 Statement of key user requirements

4.2.1 Digital quadrature demodulation

4.2.1.1 Profile rate

The data capture system must capture a minimum of 10 full profiles per second, with at least 100 frequency steps per profile. The ramp repetition interval (RRI) must therefore not exceed 100 ms, and the dwell time cannot exceed 1 ms. Amplitude and phase measurements of both the transmit-receive and the calibration channels must be taken at each frequency step.

4.2.1.2 Dynamic range

The dynamic range must be at least 70 dB (this specification is derived from the Mercury radar [5]); the maximum allowed input voltage is 1 V_{pk-pk}. The IF input impedance should be 500 Ω to ensure compatibility with the AD603 programmable gain amplifier which drives the IF signal input on the Dacs [25].

4.2.1.3 Accuracy

The phase output should have a rms output error no larger than 0.2°; this is the same as the error specification for the frequency synthesizers of the Mercury radar [5]. The rms amplitude error should be no larger than 120 μV, which is equivalent to 2Δ (quantizer step size) for a 14-bit analogue-to-digital converter.

4.2.2 System control

Dacs requires a microcontroller to implement system control and communications functions. The microcontroller software will execute from Flash memory and use SRAM as a temporary store. Although the microcontroller should eventually execute an operating system, targetting the microcontroller with code written in C is sufficient for test purposes. Only the functions necessary for testing the digital quadrature demodulator will be implemented.

4.2.3 Communication

The Dacs must communicate with other systems in the radar through a backplane interface. These systems include the two frequency synthesizers, the RF/IF board (downconverter), and the RF/IF motherboard. These functions will not be implemented on the prototype.

4.2.4 Electro-mechanical

Each system of the radar, including the Dacs, must be contained on an Eurocard-sized circuit board. The circuit board will be mounted onto a backplane in the enclosure.

4.3 System design

The data capture system consists of a printed circuit board populated with semiconductor and passive components. The functionality of the programmable logic devices is described in VHDL, and the software running on the microcontroller is written in C. Both the programmable logic devices and the microcontroller and its flash memory are programmed in-circuit.

The components can be grouped by function, as in Figure 26, which depicts only the major modules.

4.4 Hardware design and implementation

4.4.1 Design

4.4.1.1 Analogue-to-digital conversion

The ADC device was selected on the basis of resolution and sampling speed. A survey of available devices indicated that the fastest 16-bit converters, from Maxim and Analog Devices, ran at a sampling rate of 1 MSa/s - not fast enough for this application [50][51]. So a 14-bit analogue-to-digital converter was specified.

The Texas Instruments THS1408 is a 14-bit differential-input ADC and includes a programmable gain amplifier (PGA) with gain adjustable from 0 to 7 dB, and samples at speeds up to 8 MSa/s [49, p 5]. The ADC has a pipeline latency of 9.5 periods; ambiguity in dividing the reference clock makes the latency irrelevant. The THS1408 contains several control registers, which allow a programmable offset to be subtracted from the sample value and the PGA gain to be set.

A centre-tapped transformer converts the single-ended IF signal into a suitable differential signal. The maximum IF input is 1 V_{pk-pk} ac, producing a 4 V_{pk-pk} signal at the ADC input; the

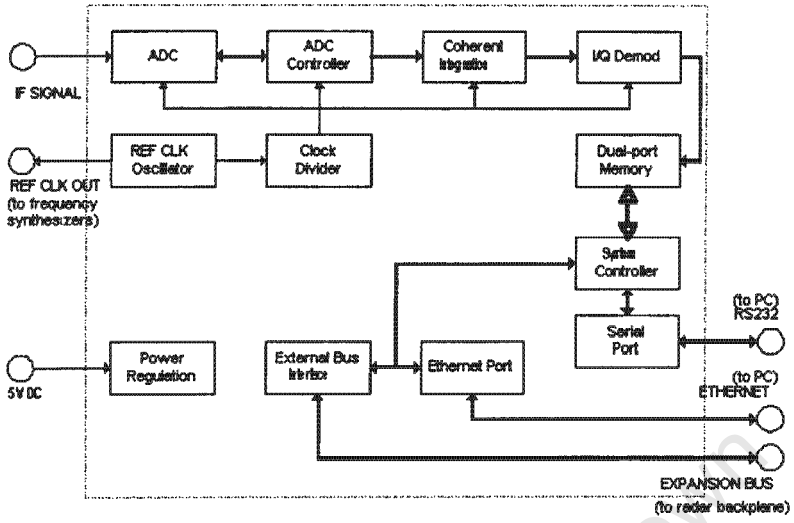

Figure 26. Function blocks of the data capture system

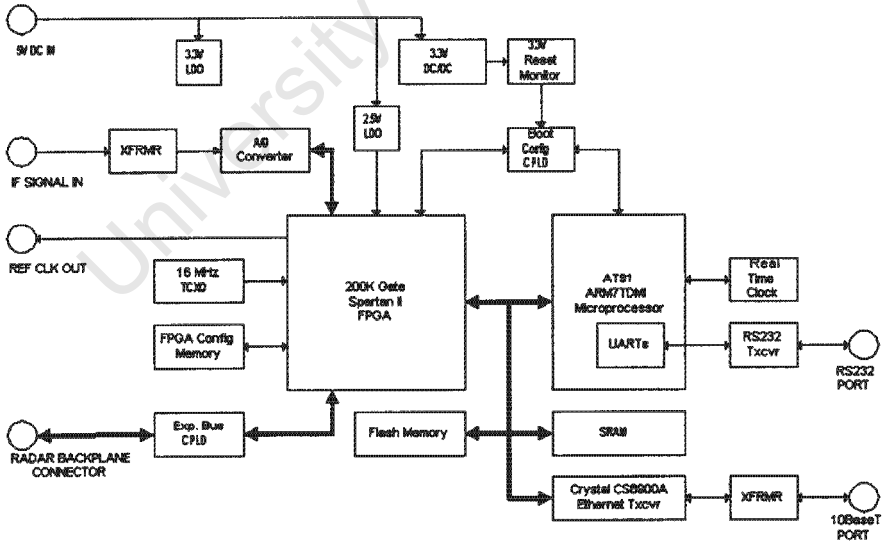

Figure 27. Block diagram of the Dacs hardware

quantizer step size is thus 61 μV referred to the transformer input. The datasheet for the THS1408 specifies a typical spurious-free dynamic range as 80 dB, for $f_{in} = 4$ MHz and $f_s = 8$ MHz [49].

4.4.1.2 Digital signal processing in the FPGA

The digital signal processing is performed in a Xilinx Spartan-II field-programmable gate array with 200,000 gates. The processing modules are coded in VHDL and loaded into the FPGA chip after power is applied.

4.4.1.3 Reference clock for digital quadrature demodulator

The Dacs design provides three sources of reference clock for the demodulator. The first two are on-board oscillators: a 16 MHz temperature-compensated crystal oscillator (TCXO) or a 32 MHz crystal oscillator. A connector provides an input port for an external oscillator, such as the HP 33120A signal generator used during testing.

4.4.1.4 Microcontroller controls system operation

The Dacs board contains an Atmel AT91M40400 microcontroller, which integrates an ARM7TDMI processor core, several on-chip peripheral devices, and a 16-bit external bus interface. The microcontroller boots off a 512K x 16-bit flash memory IC, and uses an 256K x 16-bit SRAM chip for temporary storage. The AT91M40400 executes 32-bit instructions, but the external bus interface is 16-bit so the processor executes instructions at half the bus rate.

4.4.1.5 Bootstrap configuration CPLD

A small CPLD (Xilinx XC9572) controls the boot mode of the microcontroller; this CPLD routes several signals between peripheral devices, the microcontroller, and the FPGA.

4.4.1.6 Communication interfaces

The Dacs board has several communication ports, linking it with other components of the radar system and with external devices. The Dacs board has two serial ports: one is implemented as an RS232 serial port, the other is a slot requiring a special add-on transceiver card. These serial ports provide a simple interface to external computers. A Crystal CS8900A Ethernet transceiver chip implements a 10BaseT Ethernet port, but was not placed on the prototype. The expansion bus interface is based on the 16-bit ISA synchronous bus; this is the link to other systems within the radar enclosure, which share a common backplane, and was also not placed on the prototype.

4.4.2 Implementation

The hardware design was captured as a set of schematic diagrams in Protel 99SE. The artwork for the printed circuit board was also drawn in Protel 99SE, and sent to a PCB production house, Trax

InterConnect (Pty) Ltd, who fabricated the circuit boards to specification - it is worth noting that the printed circuit board stretched the limits of the available fabrication technology, especially in terms of via size and track width/spacing in certain sections. The bare circuit board was populated by another contractor, Rhomco (Pty) Ltd.

4.5 Programmable logic design and implementation

The Dacs board contains three programmable logic devices: an FPGA used mainly for signal processing, a small CPLD assisting with boostrap configuration, and a large CPLD used for the expansion bus interface. Firmware was not produced for the expansion bus as this was outside the thesis scope.

4.5.1 Programmable logic implemented in the bootstrap configuration CPLD

The logic within the configuration CPLD is implemented as simple combinatorial logic. It configures the microcontroller boot mode at power-on, configures the FPGA programming mode at power-on, routes IO signals between the microcontroller, FPGA, and CS8900A Ethernet transceiver, and performs some simple logic functions on the reset signals.

4.5.2 Programmable logic implemented in the FPGA

The prototype board was used to test the demodulator, and was not integrated into the radar. The Dacs prototype was therefore programmed to take a sequence of measurements at a constant input amplitude and phase.

4.5.2.1 Structure

The functions implemented in the FPGA were decomposed into modules defined in terms of their behaviour and interfaces. This top-down design approach provided similar advantages to an object-oriented approach to software: reduced coupling between modules made modifications simpler, there was a clear mapping between modules and their functions, and the completed modules lend themselves to reuse [52].

Controller for analogue-to-digital converter The ADC controller is the interface between the FPGA and the ADC chip: it loads configuration parameters into the ADC, and reads sample values from the ADC, over a half-duplex, parallel bus. The ADC amplifier gain, the programmable offset, and the control register can be set.

Digital signal processor The digital signal processor acts as an accumulator, the first step in

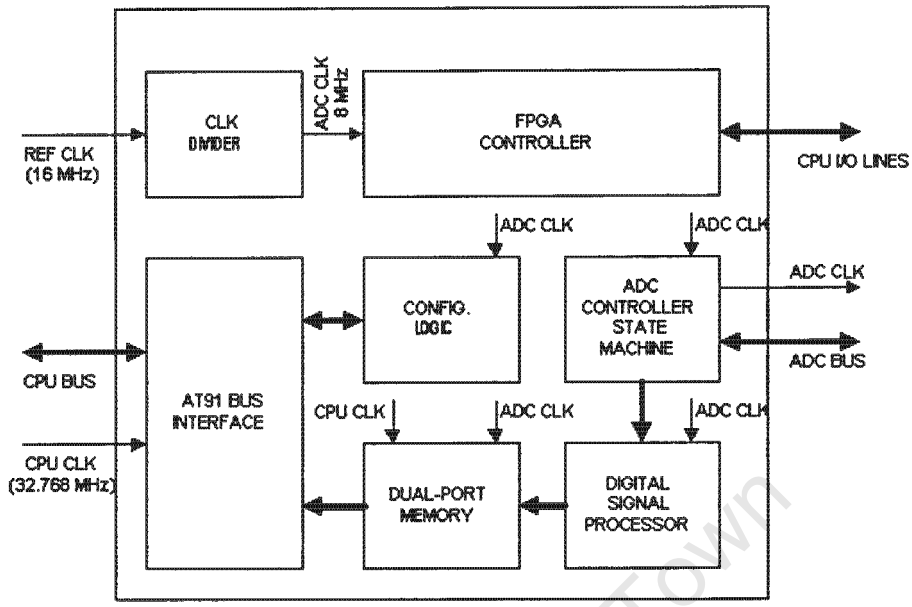

Figure 28. Block diagram of the VHDL code for the FPGA

coherent integration (division and demodulation were performed on the PC to avoid introducing round-off errors; it was also convenient to have the complete set of sample values available for analysis.) The number of samples to integrate, N, and the ensemble length, L, (the number of measurements to be taken in sequence) are parameters.

There are four accumulators, one for each sample phase needed for demodulation - the timing of the accumulator writes relative to the ADC controller output samples is shown in Figure 29; writes to accumulators 0 to 3 are represented by assertion of signals WR[0:3].

Once the full set of samples in a measurement have been loaded, the accumulator values are clocked into a set of registers and transferred into the dual-port memory while the accumulators are reset for the next measurement - the timing of this process is shown in Figure 30; the accumulators are cleared when the CLR signal is asserted, and the 8 writes to the dual-port memory occur while the MEM_WR signal is asserted. The coherent integration factor N must be at least 3 if more than one measurement is taken contiguously to allow enough clock cycles for the accumulator values to be written into memory.

Dual-port memory The 2048 x 16-bits dual-port memory is the bridge between the two clock domains: the sampling and signal processing modules run at the sample clock frequency and the host port interface runs at the microcontroller clock frequency (32.768 MHz.) This memory is

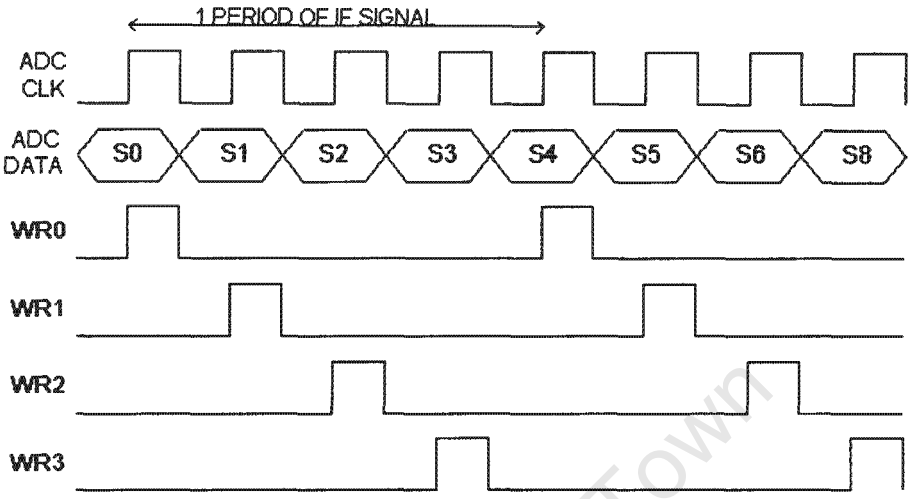

Figure 29. Timing of accumulator write signals

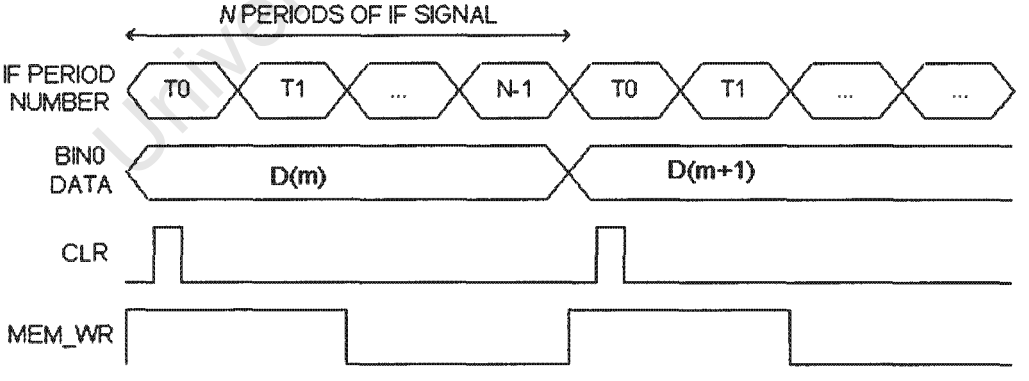

Figure 30. Timing of the accumulator clear and memory write signals

only used for storing the accumulator values, and has a capacity of 128 periods.

Host port interface to microcontroller The FPGA is memory-mapped on the microcontroller system bus. The microcontroller reads and writes configuration parameters from/to the configuration registers, and reads accumulator values from the dual-port memory. The microcontroller is bus master and initiates all transactions. The host-port interface implements the 16-bit bus protocol of the Atmel AT91M40400 microcontroller with 4 wait states.

Configuration registers Simple registers store the configuration parameters, namely the coherent integration factor N, ensemble length L (fixed at 128 for testing), ADC amplifier gain, ADC offset register value, and the ADC control register value (which should only be set to 0x0000.)

FPGA system controller The FPGA system controller directs the other modules' operation by asserting enable signals at the appropriate times and monitoring status signals from these modules. The FPGA system controller changes state under stimulation from the microcontroller.

The sampling and coherent integration must be synchronised to the IF signal if the phase estimate is to remain consistent across successive measurements. The first sample in each measurement must be taken at the same phase, which means that a measurement can be commenced only on every fourth clock edge. The FPGA system controller performs this synchronisation as the START signal from the microcontroller is asynchronous.

4.5.2.2 State machine logic

The FPGA is programmed as a state machine, expressed in Figure 31 - the diagram is simplified to enhance clarity. The FPGA system controller traverses this state machine in response to the RESET and START inputs from the microcontroller; the trigger conditions for each transition are shown. Microcontroller access to the configuration registers and dual-port memory is independent of the system controller state, with the exception that the dual-port memory cannot be read while the system controller is in the RESET state.

4.5.2.3 Implementing the programmable logic

The behaviour of the programmable logic for the FPGA and CPLD was specified in VHDL using a modular approach. The VHDL was synthesized and compiled in Xilinx ISE 4.1 on a Pentium II PC running Microsoft Windows 2000. The bit files were downloaded to the target devices using a special JTAG in-system programmer; all the programmable logic devices share a JTAG chain accessed via a single connector.

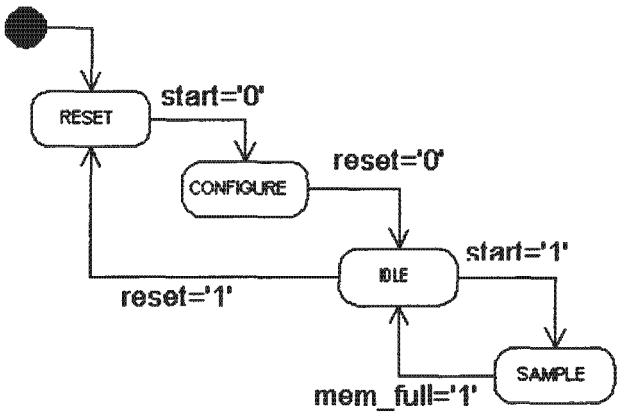

Figure 31. State machine representation of the FPGA system controller

4.6 Software design and implementation

The primary functions carried out in software are control of the FPGA state, configuration of the FPGA and ADC parameters, provision of a user interface, and digital quadrature demodulation. Software is executed on the Dacs board microcontroller and on a PC, the two computers linked by a high-speed serial port.

4.6.1 Software design

The software design partitions the functions according to which processor - microcontroller or PC - they will be executed on (Figure 32.) The microcontroller reads the sets of accumulator values from the memory-mapped FPGA and transmits them to the PC, which performs digital quadrature demodulation to estimate the amplitude and phase of the sampled signal.

The PC software also provides a simple user interface. Figure 33 shows a typical sequence.

4.6.2 Microcontroller software

The microcontroller code, written in C, has a simple structure. A few preliminary instructions configure the microcontroller peripherals, while the main routine is an infinite loop. The loop calls a few high-level functions to accomplish its tasks, namely configuring the FPGA, executing the measurement sequence, or running a test routine.

4.6.2.1 Behaviour

The microcontroller software idles in an infinite loop, waiting to receive a command over the serial port. The microcontroller jumps to the data reception, configuration, measurement, or a test procedure, contingent on the command. Control returns to the main loop once the procedure is

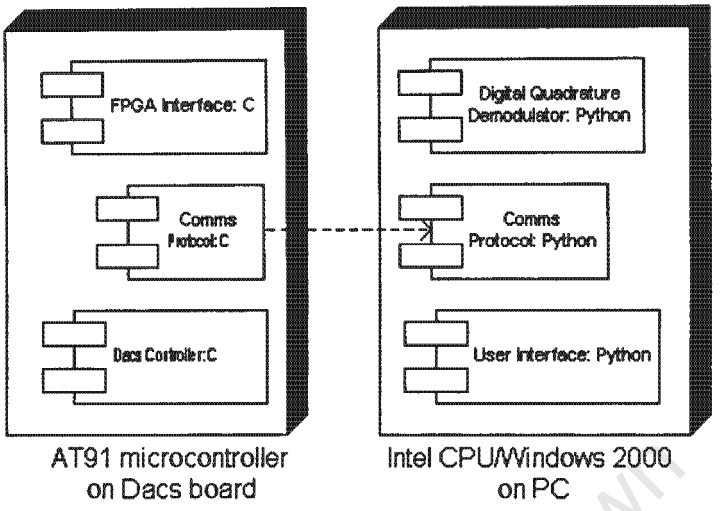

Figure 32. Software is partitioned between the AT91 microcontroller and a PC

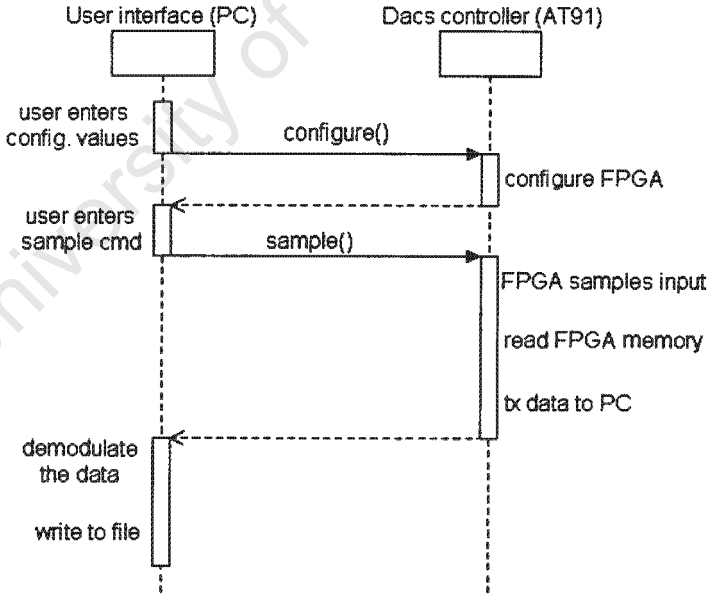

Figure 33. Sequence of software cooperation between PC and microcontroller

complete.

The PC software sends the configuration parameters to the microcontroller in a stream of bytes. Reception of the configuration command causes the microcontroller to place the FPGA into the RESET state. Once the parameters have been loaded into the FPGA, the FPGA is placed into CONFIGURE state; the FPGA enters the IDLE state after configuration. The FPGA state is set to SAMPLE if the sample command is received; once the FPGA indicates that sampling is complete, the microcontroller reads the data from the dual-port memory and transmits it to the PC.

4.6.3 PC software

The software on the PC is the user interface, and also performs digital quadrature demodulation on the data received from the Dacs. The software was implemented in Python [48], an object-oriented, open-source scripting language chosen for its ease of use. Python runs on GNU/Linux or Microsoft Windows 2000. The program for interfacing to the Dacs board runs in a MS-DOS window on Windows, but some of the features were ported to Linux during testing.

4.6.3.1 Behaviour

The PC software presents a command-line interface to the user, who can set the configuration parameters to be programmed into the FPGA, cause the FPGA to be configured, or have a set of measurements taken. Once an operation has been completed the user may choose another.

The FPGA is configured and measurements are taken in cooperation with the microcontroller on the Dacs, with communication over the serial port.

4.7 Measuring the performance of the demodulator prototype

The performance of the Dacs board was measured once the prototype hardware and software were working satisfactorily. The test equipment limited the precision of the performance tests, so the tests aimed to establish the effectiveness of coherent integration and subsampling in improving the demodulator output rather than measuring absolute accuracy. The demodulator performance was therefore measured in terms of statistical properties rather than absolute accuracy.

4.7.1 Test setup

The prototype Dacs board was connected to a pair of HP 33120A signal generators, phase-locked using the HP 33120-90005 kit. These provided the sampling clock and the 2 MHz IF signal.

The HP 33120A signal generators used digital-to-analogue converters with an amplitude resolution of 12-bits [53], whereas the device under test used a 14-bit analogue-to-digital

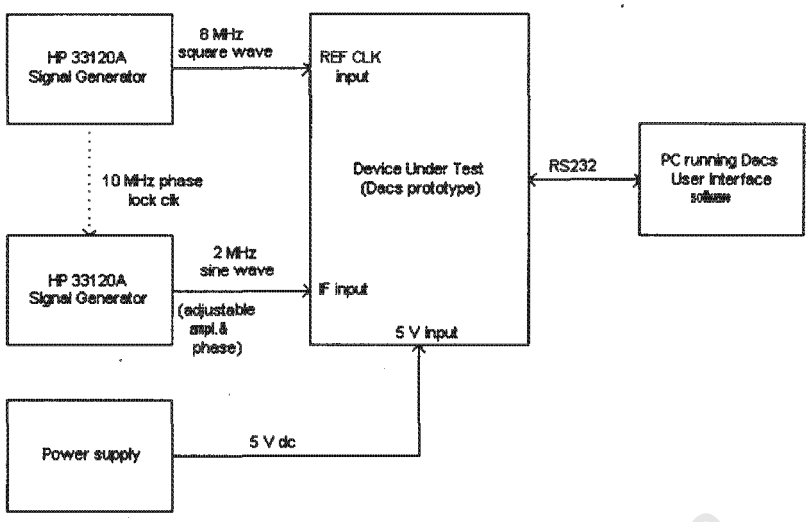

Figure 34. Diagram of the test setup, showing clock frequency for subsampling inactive

converter. The signal generator's harmonic distortion for a sinusoidal output at 2 MHz was specified at –35 dBc, and amplitude accuracy was given as no better than ±1%, with flatness of a sinusoidal output given as ±3.5% over the voltage range of interest [53]; in other words, measurements were limited by the *signal sources*. The generators had not been calibrated in years, so their performance was likely even worse than specified.

4.7.2 Test method

The device under test (DUT) is considered as a black box, with inputs for clock and IF signals, and outputs for amplitude and phase. The imprecision of the signal generators meant that the amplitude and phase of the input signal were not known accurately. Several ensembles of 128 contiguous amplitude and phase measurements were taken, then one of the parameters was changed and the process repeated. The recorded data was analysed in MATLAB: the root mean square errors of the amplitude and phase ensembles were calculated. The *exact* amplitude and phase of the input were unknown, so the rms errors were calculated by taking the mean of each amplitude and phase ensemble as a baseline and defining the error signal as the deviation from the mean.

4.7.3 Performance test results

4.7.3.1 Linearity of demodulator outputs

Amplitude linearity This experiment aimed to determine the linearity of the demodulator output, A'. The input signal amplitude A_{in} was stepped in equal increments of 25 mV, over the range 25

mV to 500 mV; the phase was fixed at 45 degrees. The coherent integration factor was 1024 and the ensemble length 128. Eight amplitude ensembles were recorded at each amplitude, and the average of the ensemble means was taken as the output amplitude.

Figure 35 is a plot of the ratio $\frac{A'}{A_{in}}$, and shows that the output is quite linear, although the exact ratio varies somewhat unpredictably. The HP 33120A signal generator changes its output amplitude by switching attenuators into and out of the signal path[53], and is likely the source of some of the nonlinearity. The average ratio of the output to input amplitude is 0.9085: this is less than 1 because of input impedance mismatch - the input impedance is not exactly 50 Ω - and insertion loss in the RF transformer.

The difference in output amplitude step size between consecutive input amplitude steps is plotted in Figure 36. The differential step size falls within a 200 μV range except for 3 points, errors attributed to the signal generator; an error band of 200 μV corresponds to slightly more than 3Δ, the quantizer step size - so the demodulator output amplitude is fairly linear. The differential step size is slightly larger than the rms amplitude error measurements reported below.

Phase linearity The purpose of this experiment was to measure the linearity of the demodulator phase output, ϕ'. The input signal phase was incremented in steps of 10° from 0° to 350°, at a constant amplitude of 1000 mV_{pk-pk}. The coherent integration factor was 1024 and the ensemble length 128. Four phase ensembles were recorded at each phase, and the average of the ensemble means was taken as the output phase.

The differential step size - the difference between consecutive phase outputs, minus the nominal step size of 10° - is plotted in Figure 37. The differential step size is within ±0.1° - that is, the demodulator phase output is linear within ±0.1°, with the exception of 1 data point, an error most likely due to the signal generator. However, the differential step size is an order of magnitude larger than the rms phase error measurement reported below, a discrepancy attributed to the phase linearity of the signal generator. The differential step size is also discernibly periodic, repeating four times over the 350° input range, again this is most likely caused by the signal generator.

4.7.3.2 Influence of coherent integration on estimated rms amplitude and phase errors

This experiment investigated the relationship between coherent integration and the rms demodulator output errors. The amplitude and phase of the IF signal were fixed, and 8 ensembles were recorded for each coherent integration factor; the experiment was repeated for amplitude settings of 100, 500, and 1000 mV_{pk-pk}, with the phase set to 45 degrees. The rms error signals

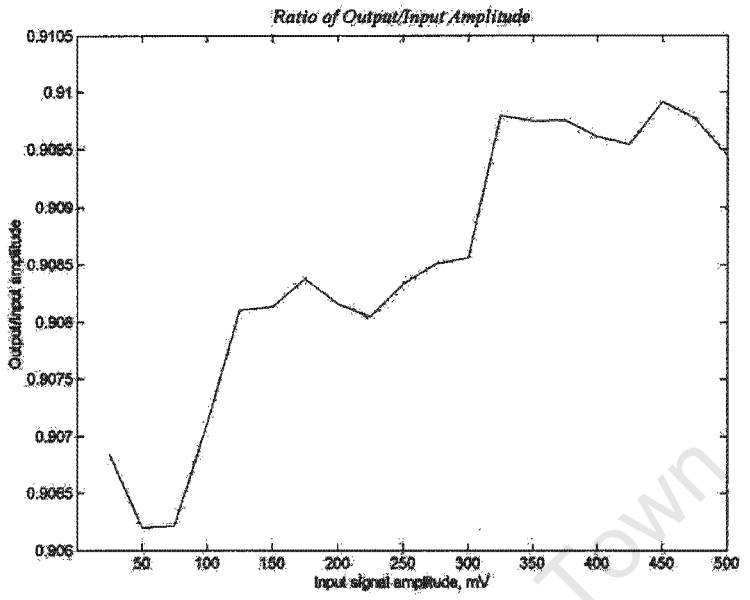

Figure 35. Ratio of demodulator output amplitude to input signal amplitude

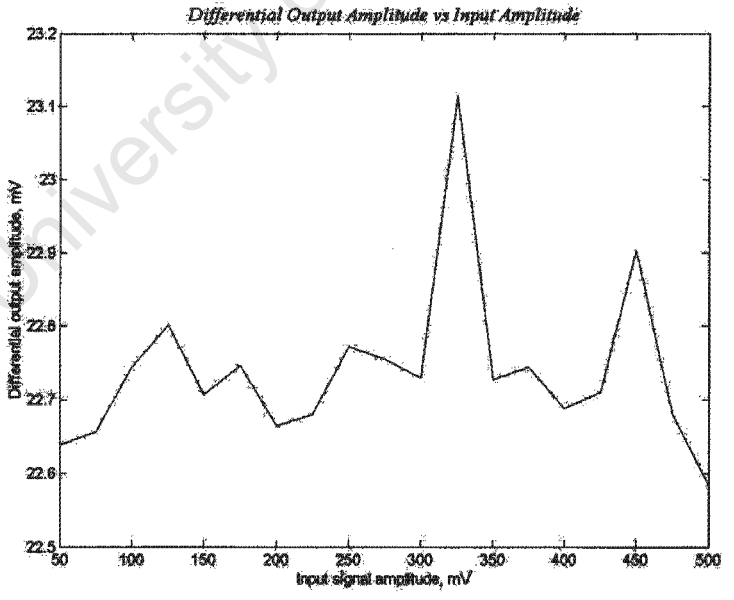

Figure 36. Difference in output amplitude between consecutive input amplitude steps

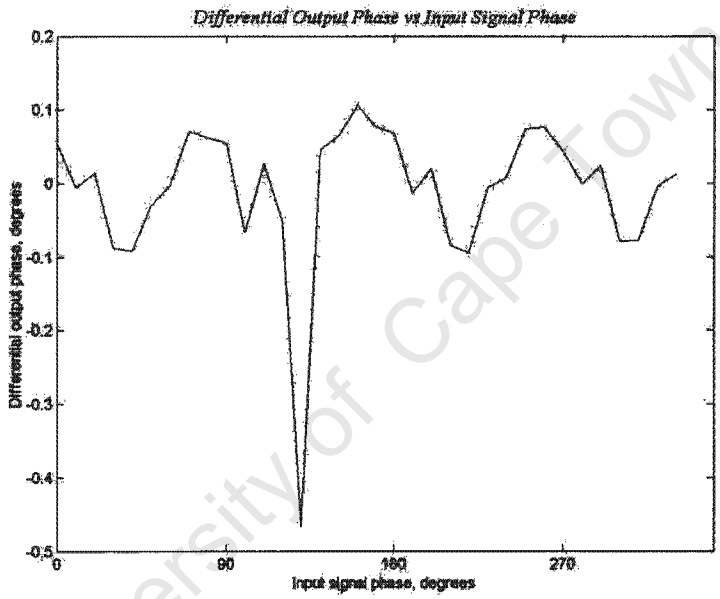

Figure 37. Difference in phase output step size between consecutive phase steps

are plotted on logarithmic axes in Figures 38 and 39, showing the effect of coherent integration.

The measured amplitude error is clearly improved by coherent integration: a sharper initial improvement flattens out as N increases, similar to the simulated results for significant timing jitter, although the improvement ratio for 100 mV_{pk-pk} amplitude remains more constant. The amplitude error is greatest for an input amplitude of 1000 mV_{pk-pk}, explained by the dependence of the voltage error due to timing jitter on the input amplitude, as shown in (15).

We expect coherent integration to reduce random errors: the rms amplitude errors ϵ_1 and ϵ_2 at N_1 and N_2 respectively should be related by $\frac{\epsilon_1}{\epsilon_2} \approx \sqrt{\frac{N_2}{N_1}}$, a relationship confirmed by the simulations. Table 5 shows that this relationship holds for an input amplitude of 100 mV_{pk-pk}, but not for the larger amplitudes.

N_1,N_2	$\sqrt{\frac{N_2}{N_1}}$	$\frac{\epsilon_{A,2}}{\epsilon_{A,1}}$	$\frac{\epsilon_{A,2}}{\epsilon_{A,1}}$	$\frac{\epsilon_{A,2}}{\epsilon_{A,1}}$
V_{in}		100mV_{pk-pk}	500mV_{pk-pk}	1000mV_{pk-pk}
4,16	2	1.99	1.88	1.86
4,64	4	4.03	3.43	2.93
4,256	8	8.20	5.68	4.84
4,1024	16	15.16	9.49	7.06
4,4096	32	27.11	12.22	9.15

Table 5. Effect of coherent integration on amplitude error for different input amplitudes

The rms phase error has a much less predictable behaviour. The error is largely independent of the input amplitude, except for the initial points on the 100 mV_{pk-pk} curve. Coherent integration has little effect on the rms error - in fact, the curves are nearly horizontal: only the region between $N = 256$ and $N = 1024$ shows much improvement. The error is worse for the first few points of the 100 mV_{pk-pk} curve. Simply estimating the rms timing jitter from the measurements, we obtain an answer of about 0.01°, or 14 ps, which seems incredibly small. Equation (15) suggests the resulting amplitude error for an input amplitude of 1000 mV_{pk-pk} is about 78 μV, which is not contradicted by the rms amplitude measurements shown in Figure 38, the largest of which is less than 50 μV.

4.7.3.3 Consistency of error measurements

The instability of the ADC output is noted in Appendix C (see Figure 42), and attributed to the signal generator. The rms amplitude and phase error measurements described above are quite small: the amplitude error is less than 50 μV and the phase error is less than 0.01°. Since the actual amplitude and phase were not known, the ensemble mean was used, and the deviation from the mean was used as the error signal. The results showed a high degree of repeatability.

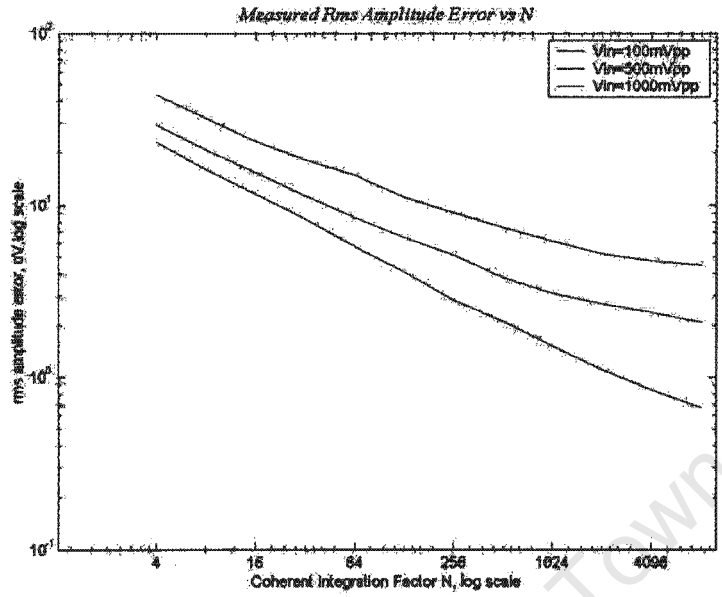

Figure 38. Measured rms amplitude error versus coherent integration factor

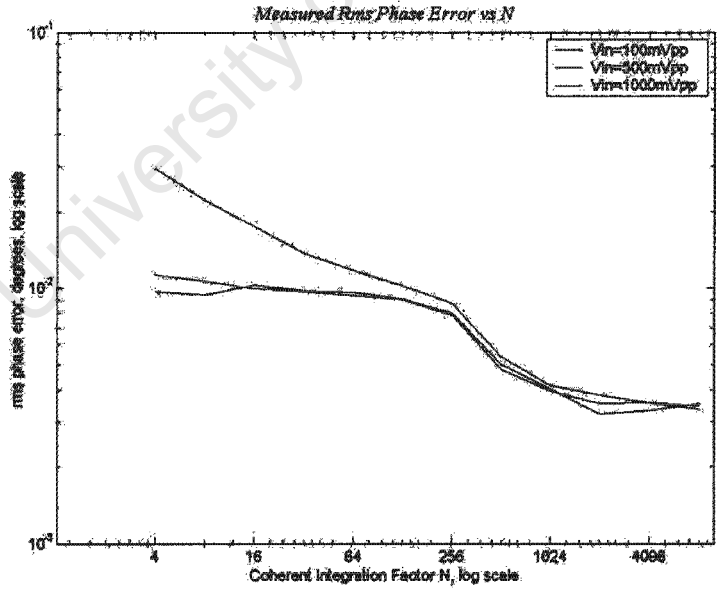

Figure 39. Measured rms phase error versus coherent integration factor

The experiment to measure the drift of the ADC output recorded 256 amplitude and phase ensembles, with an input amplitude of 1000 mV_{pk-pk} and coherent integration factor of 1. The mean rms amplitude error was 71 μV, with a standard deviation of 14 μV. The mean rms phase error was 0.017°, with a standard deviation of 0.002°. Both of these results agree with the measurements plotted in Figures 38 and 39, which were recorded with higher values of N.

4.7.3.4 Influence of subsampling on the amplitude and phase error

Subsampling Ratio	$\frac{\epsilon_{A,N=4}}{\epsilon_{A,N=4096}}$	$\frac{\epsilon_{A,N=4}}{\epsilon_{A,N=4096}}$	ϵ_A	ϵ_A
V_{in}	$100mV_{pk-pk}$	$1000mV_{pk-pk}$	$100mV_{pk-pk}$	$1000mV_{pk-pk}$
$P = 1$	27.11	9.15	23μV to <1μV	43μV to 5μV
$P = 5$	128.72	10.00	134μV to 1μV	57μV to 6μV

Table 6. Effect of subsampling on amplitude error

The signal generators maintained a very poor phase lock when the subsampling ratio P was 3, but the lock was significantly better with $P = 5$, so measurements were only taken for the latter value. The procedure was the same as for the above measurements. Eight ensembles were recorded for each coherent integration factor N, at fixed input amplitude and phase; the experiment was repeated for amplitude settings of 100, 500, and 1000 mV_{pk-pk}.

There were several differences between the performance obtained for $P = 1$ and $P = 5$, summarised in Tables 6 and 7, which show results obtained with input amplitudes of 100 and 1000 mV_{pk-pk}.

Coherent integration reduces the rms amplitude and phase errors to similar values whether subsampling is selected or not. Since the acquisition time is 5 times longer when P is 5 than when it is 1, it is more sensible to set P to 1, allowing a larger coherent integration factor given the limited dwell time of the radar - subsampling should be discounted in favour of coherent integration to obtain the lowest errors in a given acquisition time.

4.8 Summary

The prototype Dacs board contains a digital quadrature demodulator as well as several other modules required to perform its roles in the Venus radar system. The key demodulator requirements are a ramp repetition interval of 100 ms, a dynamic range of at least 70 dB, an amplitude accuracy of at least 120 μV, and a phase accuracy of at least 0.2°.

A microcontroller directs the system operation, implementing the control logic and the communications interface to an external computer. The IF signal is sampled at 8 MHz by a

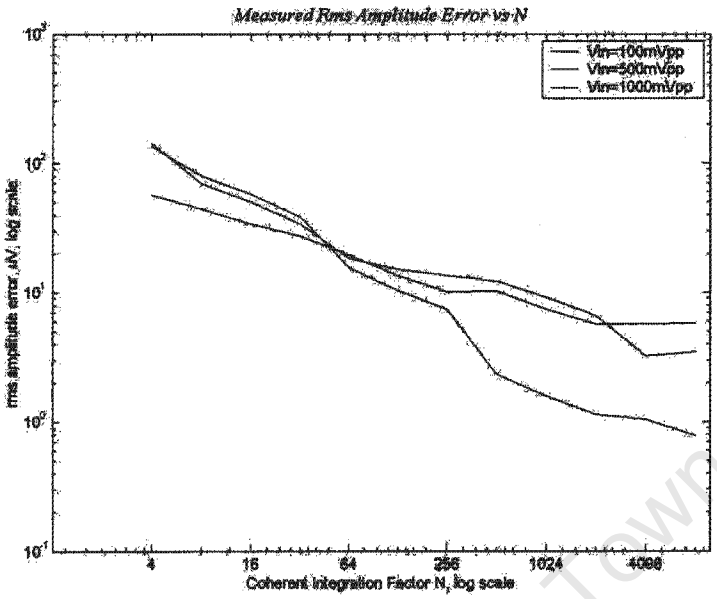

Figure 40. Rms amplitude error versus coherent integration factor, for $P = 5$

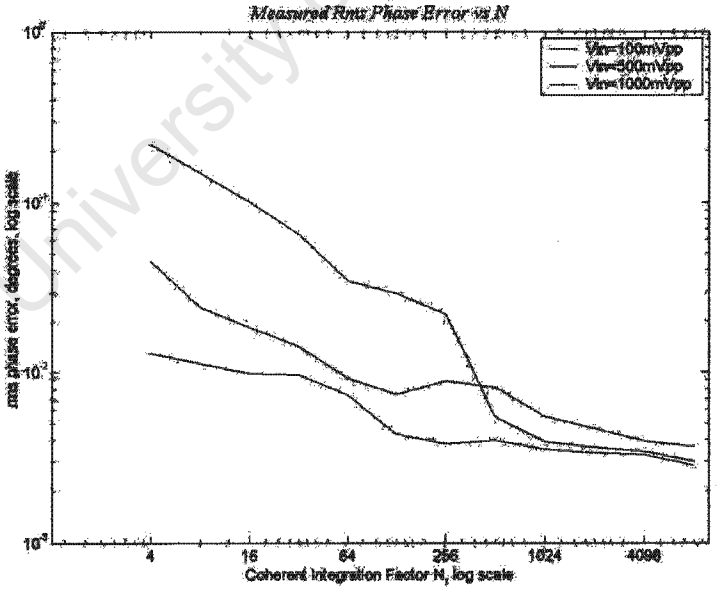

Figure 41. Rms phase error versus coherent integration factor, for $P = 5$

Subsampling Ratio	$\frac{\epsilon_\phi, N=4}{\epsilon_\phi, N=4096}$	$\frac{\epsilon_\phi, N=4}{\epsilon_\phi, N=4096}$	ϵ_ϕ	ϵ_ϕ
V_{in}	$100mV_{pk-pk}$	$1000mV_{pk-pk}$	$100mV_{pk-pk}$	$1000mV_{pk-pk}$
$P = 1$	8.35	2.92	0.03° to 0.004°	0.01° to 0.003°
$P = 5$	64.07	3.93	0.22° to 0.003°	0.01° to 0.003°

Table 7. Effect of subsampling on phase error

14-bit ADC, and the samples are processed in an FPGA. The microcontroller reads the processed samples from a dual-port memory in the FPGA and transmits them to a PC.

The demodulator performance was measured using phase-locked signal generators to provide the clock and IF signals, but the generators' imprecision limited the accuracy of the tests. The amplitude output was linear, with the differential error within a 3Δ band, while the phase output was linear within 0.1°. Timing jitter was found to be the most significant source of sampling error; coherent integration reduced the amplitude error considerably, while the phase error was less susceptible to improvement. Subsampling was found to offer no benefit, as expected from the theoretical model.

The simulation and measurement results are compared to the specifications in the next chapter, which also presents the conclusions and recommendations for future work.

Chapter 5
Summary

5.1 Analysis of results

5.1.1 Analysis of simulation results

5.1.1.1 Isolated error sources

Simulation of thermal noise in the input signal confirmed the benefit of coherent integration in improving the output accuracy. The simulations showed that dithering improved the output accuracy even more than the accuracy achieved when the noise was so small, relative to the quantizer step size, that no dithering occurred (Section 3.2.1.)

Timing jitter introduced significant errors into the demodulator outputs. The error was independent of quantizer step size for timing jitter with a standard deviation larger than 1 ns. Although both amplitude error and phase error improved with coherent integration, this was subject to diminishing returns in the case of amplitude error, which quickly approached a limit (Section 3.2.2.)

5.1.1.2 Simulated performance of the complete demodulator

Simulation also gauged the performance of a model representing the prototype Dacs hardware, using a 14-bit quantizer, driven by the radar signals. The simulations were performed for an input signal with 70 dB dynamic range (A=999 mV_{pk-pk} to 316 μV_{pk-pk}), for rms timing jitter σ_{tj} from 2 ns to 4 ns, and for thermal noise standard deviations of 70.5 μV and 1.12 μV (corresponding to SNR of 74 dB and 80 dB with the maximum input.) The amplitude and phase errors depended on the input amplitude, although the significance of the errors, especially the phase error, was much higher at small amplitudes. The dominant error contribution, for all but the smallest amplitudes, was attributed to timing jitter; the main contributor of timing jitter is frequency synthesizer phase noise, downconverted into the IF signal (Section 3.3.) The model includes only error sources that depend on the *input signal* frequency, so predicted that subsampling - using a lower *clock* frequency - should provide no benefit.

5.1.1.3 Predicted performance and the specifications

The specifications required a maximum rms amplitude error of 120 μV and a maximum rms phase error of 0.2° (Section 4.2.1.) The simulations predict that the specified performance will not be

A	SNR	$\epsilon_{A,specified}$	$\epsilon_{A,N=1024}$	$\epsilon_{\phi,specified}$	$\epsilon_{\phi,N=1024}$
999 mV_{pk-pk}	74 dB	120 μV	400 μV	0.2°	0.04°
316 μV_{pk-pk}	4 dB	120 μV	1.6 μV	0.2°	0.6°

Table 8. Simulated performance compared to specifications

achieved over the entire operating range. The dwell time limits the coherent integration factor N to 1000, closely approximated in the simulations by N=1024. At input amplitude A of 999 mV_{pk-pk} the rms amplitude error exceeds the specification due to the large timing jitter, while the phase jitter is acceptable; at an amplitude A of 316 μV_{pk-pk} the amplitude error is within the required tolerance, while the phase error now exceeds the specification due to the small ratio of input amplitude to quantizer step size (Section 3.3.)

5.1.2 Analysis of results from prototype demodulator

The performance of the prototype demodulator was measured in less than ideal conditions: the input amplitude was guaranteed to only 1%; the input amplitude was not especially stable - the signal generator amplitude drifted by several mV over periods of a few minutes; and the input amplitude range extended down to only 50 mV_{pk-pk}(Section 4.7.1.) A measurement campaign was performed nonetheless; the error signal was taken as the deviation from the ensemble mean, since the actual amplitude and phase of the input were not known precisely.

5.1.2.1 Performance of the demodulator

The demodulator outputs proved to be quite linear with a coherent integration factor N of 1024. The amplitude input was stepped in 25 mV increments, and the resulting output steps had a differential step size within $\pm$100 μV of the mean (the output step size was actually about 22.7 mV, showing the effect of transformer insertion loss.) The phase input was stepped in 10° increments, and the differential step size of the output was within $\pm$0.1° of the expected 10° (Section 4.7.3.1.)

Coherent integration was shown to reduce both the amplitude and phase error. The dominant source of error appeared to be timing jitter, in agreement with the simulations; the rms timing jitter was estimated from the measurements as less than 0.02° (Section 4.7.3.2.) Further in agreement with the theory, subsampling was shown to introduce no improvement in the output accuracy (Section 4.7.3.4.) Given the limited dwell time, subsampling should be discarded in favour of increasing the coherent integration factor.

5.1.2.2 Measured performance and the specifications

The performance measurements proved to be highly repeatable, despite the less than ideal test setup. Measurements could not be performed for input amplitudes less than 50 mV_{pk-pk}, so the full dynamic range of the demodulator was not exercised. Simulation of the demodulator and radar, however, predict that the phase error will exceed the specification (Section 3.3.)

A	$\epsilon_{A,specified}$	$\epsilon_{A,N=1024}$	$\epsilon_{\phi,specified}$	$\epsilon_{\phi,N=1024}$
1000 mV_{pk-pk}	120 μV	6.2 μV	0.2°	0.04°
100 mV_{pk-pk}	120 μV	1.5 μV	0.2°	0.04°

Table 9. Measured performance compared to specifications

Measurement shows that the prototype demodulator performance was within the specified limits (Sections 4.7.3.1, 4.7.3.2.) The dominant error contribution was from timing jitter, but the measurements do not represent the performance achievable in the radar, since the timing jitter of the test signals was much better than the expected phase noise of the radar IF signal. Reducing the IF phase noise, by reducing the synthesizer phase noise, will improve the demodulator performance.

5.2 Summary

This dissertation has described the design and implementation of a digital quadrature demodulator for a stepped frequency ground penetrating radar system; the demodulator is part of the data capture system board. The theoretical section began with an overview of the radar architecture and operation, and how the interleaved dual-channel design allows the demodulator to overcome ambiguity in the clock phase by making *relative* phase measurements. The demodulator design was approached from a theoretical perspective, with the intention of elucidating the primary causes of error in the demodulator output. After a suitable demodulator architecture employing direct sampling of the IF signal was described, the potential of coherent integration to improve the demodulator accuracy was examined. The theoretical section concluded by presenting models of the individual components from which the demodulator was constructed. The primary sources of error were found to be thermal noise and timing jitter. The short delay between the received signal and the local oscillator signal meant that the phase noise of the two signals was correlated, but the dependence of timing jitter on target range was not examined.

Computer simulations were performed on the demodulator model to determine the significance of timing jitter and thermal noise in corrupting the demodulator output and the efficacy of coherent integration in reducing these errors. Timing jitter was found to be the more serious error mechanism. The performance of the demodulator inside the radar was also simulated using signals with the signal-to-noise ratio and phase noise characteristics of Mercury radar system. The simulations predicted that the demodulator would not meet its performance specifications over the required dynamic range.

The design and construction of the prototype demodulator began with a review of the key user requirements, from which a design for the system was developed. The design of the hardware, programmable logic, and software was explained in some detail, showing the operation of the demodulator. The demodulator performance was measured and found to comply with the specification, for the region of the dynamic range that was tested; it was noted, though, that the clock and IF signals contained less phase noise or timing jitter than the signals expected in the actual radar.

5.3 Conclusions

The following is concluded from the results obtained from the simulations and measurements:

(1) Timing jitter introduces the most error into the demodulator outputs. This is an especially significant result since stepped frequency radars need frequency synthesizers which operate over a very wide bandwidth, making low timing jitter over the entire bandwidth difficult to achieve. Timing jitter, in the form of phase noise introduced by the frequency synthesizers, prevents the demodulator performance from reaching the specified accuracy.
(2) The demodulator performance also deteriorated for signal amplitudes that were not much greater than the quantizer step size (the smallest simulated input amplitude was less than 3Δ.) The demodulator performance would not meet the specifications under these conditions, even with coherent integration, as the demodulator phase output was sensitive to small errors.
(3) A prototype demodulator was constructed, and successful operation of the prototype was demonstrated, using signal generators to supply the clock and IF signals.
(4) The signal generators were not precise enough to characterise the demodulator accurately, but measurements showed the demodulator to be capable of meeting the performance specifications, provided the phase noise of the input signals was low enough.
(5) Coherent integration was found to reduce the errors in the amplitude and phase outputs in the presence of timing jitter and thermal noise. The simulations predicted that the error would be reduced by the square root of the coherent integration ratio. Measurements of the prototype performance found this was closely approximated by the amplitude error when the input amplitude was 100 mV_{pk-pk} (the error was reduced to less than $\frac{\Delta}{30}$); the phase error did not show as significant an improvement.
(6) Subsampling did not improve the demodulator performance, in accordance with theoretical prediction. The demodulator should therefore be operated at a sampling frequency of 8 MHz, allowing coherent integration of 1000 periods per channel (assuming the synthesizers have negligible settling time.)

5.4 Recommendations

The following recommendations are made:

(1) The performance of the digital quadrature demodulator should be measured using signal generators of greater accuracy than the HP 33120A generators.
(2) Calibration of the analogue-to-digital converter should be investigated, with the aim of

removing systematic errors.

(3) Integration of the Dacs board with the Venus radar system should be started. This involves implementing the Ethernet port and the radar backplane interface, porting an operating system to the microprocessor, implementing the radar control software, and performing the digital quadrature demodulation on the microprocessor rather than the PC which was used during testing.

Appendix A
Simulation of the demodulator

A.1 Digital quadrature demodulator model

A.1.1 Noise power calculations

The noise power is set directly by the signal-to-noise ratio, SNR_{th}. The signal power into 1 Ω is $\sigma_s^2 = \frac{A^2}{2}$ [14] and the noise power is calculated as

$$\sigma_{th}^2 = \frac{\sigma_s^2}{10^{\left(\frac{SNR_{th}}{10}\right)}}$$

which follows from the fact that the signal-to-noise ratio is the ratio of the mean-square signal to mean-square noise, which in the case of white noise is the simply the ratio of the variances [14]; we can calculate the noise variance using the 1 Ω powers since the signal-to-noise ratio is independent of impedance.

A.1.2 Vectron TO-330 rms timing jitter

The Vectron TO-330 single-sideband phase noise is specified as -90 dBc at 100 Hz, -110 dBc at 1 kHz, and -130 dBc at 10 kHz [54]. The phase noise at 10 Hz was interpolated as - 50 dBc.

The output is a 2.0 V square-wave at 16 MHz; the output power is 40 mW into 1 Ω. The SSB phase noise power, expressed in units of Watts, is then 4 x10^{-7} W at 10 Hz, 4 x10^{-11} W at 100 Hz, 4 x10^{-13} W at 1 kHz, and 4 x10^{-15} W at 10 kHz.

Following Adler [38], we integrate the phase noise power over the bandwidth, obtaining a total power of 1.336x10^{-6} W, giving an equivalent sideband level of -44.76 dB at 10 kHz offset. Treating this as phase modulation yields an rms jitter of 0.33°, which is equivalent to an rms jitter of 58 ps in a 16 MHz signal.

A.2 Simulation parameters

A.2.1 Simulating thermal noise

Phase one of the simulations used the following parameters:

(1) input signal amplitude A of 0.9987654 V,
(2) input signal phase ϕ of 0.1111 rad,
(3) signal-to-noise ratio SNR_{th} of {20dB, 30dB, 40dB, 50dB, 60dB, 70dB, 80dB, 90dB, 100dB},
(4) amplifier gain of 1,

(5) quantizer number of bits B of {8, 10, 12, 14, 16},
(6) quantizer range -1 V to 1 V,
(7) coherent integration factor N of {1, 4, 16, 64, 256, 1024, 4096},
(8) and ensemble length L of 128.

Phase two of the simulations used the following parameters:

(1) input signal amplitude A of 0.9987654 V,
(2) input signal phase ϕ of $\{(\frac{i}{11.211})2\pi;\ i = 0, 1, 2..10\}$ rad,
(3) signal-to-noise ratio SNR_{th} of { 40dB, 80dB},
(4) quantizer number of bits B of {8, 10, 12, 14, 16},
(5) coherent integration factor N of {1, 4, 16, 64, 256, 1024, 4096},
(6) and ensemble length L of 128.

A.2.2 Simulating timing jitter

The timing jitter simulations used the following parameters:

(1) input signal amplitude A of 0.9987654 V,
(2) input signal phase ϕ of $\frac{k}{5.11}2\pi$; $k = 0, 1, ..5$ rad,
(3) standard deviation of timing jitter σ_{tj} of {500ps, 1ns, 2ns, 4ns, 8ns, 16ns},
(4) amplifier gain of 1,
(5) quantizer number of bits B of {10, 12, 14, 16},
(6) quantizer range of -1 V to 1 V,
(7) coherent integration factor N of {1, 4, 16, 64, 256, 1024, 4096},
(8) and ensemble length L of 256.

A.2.3 Simulating the complete demodulator

The simulation parameter sets were chosen as

(1) input signal amplitude A of {499.5 mV, 158.113 μV},
(2) input signal phase ϕ of $\frac{k}{5.11}2\pi$; $k = 0, 1, ..5$ rad,
(3) thermal noise standard deviation σ_{th} of {1.12 μV, 70.5 μV},
(4) standard deviation of timing jitter σ_{tj} of { 2ns, 3ns, 4ns},
(5) amplifier gain of 4,
(6) quantizer number of bits B of 14,
(7) quantizer range of -2 V to 2 V,
(8) coherent integration factor N of {1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024},
(9) and ensemble length L of 256.

Appendix B
Microcontroller software tools

B.1 Software libraries and tools

The source code was compiled and linked against a set of C libraries providing basic functions including access to hardware peripherals. The linker also included an object file created from the bootstrap code, which was written in assembly. The libraries and bootstrap code were adapted from code used in a previous project in the Radar Remote Sensing Group [55].

The source code was compiled, linked, analysed, and converted using a set of programs provided as open-source software by the GNU collective [56]. The programs were downloaded as source code, compiled, and then installed on a PC running the Debian distribution of GNU/Linux. The installation process necessitated some patches and bug fixes.

A small, open-source program called *armtool* [57] communicated with the Test Access Port (TAP) on the AT91 microcontroller, allowing single instructions to be issued to the microcontroller core, the microcontroller's program counter to be set, and memory devices (including on-chip registers) to be read. The *armtool* software was modified to work with a customised in-system programmer: an Altera ByteBlaster modified to suit the Atmel AT91M40400's JTAG port and connected to the PC's parallel port.

B.2 Cross-compiling the software

The software was written in C, except the bootstrap code which was written in assembly language. The source code was cross-compiled and linked using the GNU compiler collection (*arm-elf-gcc*) and GNU linker [56]. The compilation and linking process was controlled by GNU *make* and a complicated makefile. This makefile was generated by writing a configuration script, then running *automake* and *autoconf* against the script to produce a configure file; executing the configure file in the build directory constructed the build tree and appropriate makefiles. The result of the compilation process was a binary file targeted to boot the microcontroller from flash memory. This file was in ELF format, and needed conversion into a raw binary file by the *arm-elf-objcopy* binary conversion utility. Software was targeted to SRAM during testing - once the software worked satisfactorily it was programmed into the flash memory.

B.3 Programming the flash memory

The hardware and software for accessing the microcontroller's Test Access Port via JTAG is described above, and allowed the PC to download a binary file into the SRAM. A small program to read a binary file from SRAM and write it to the flash memory IC was written to enable flash programming. The programming process is:

(1) Download the target binary file (linked to boot from ROM) into SRAM on the Dacs, at a certain base address.
(2) Download the flash programming application into SRAM (linked to run from SRAM), at another base address.
(3) Instruct the microcontroller to begin executing from the start address of the flash programming application; the target binary file will be programmed into the flash memory.
(4) Press the reset switch, causing the microcontroller to boot off the flash memory, which now contains the new software.

Appendix C
Testing the hardware and software

C.1 Hardware tests

C.1.1 Basic functions

The hardware was tested first, since a working circuit board was necessary to test the firmware and software. The first test checked the functioning of the power supply, which worked correctly.

C.1.2 Programmable logic devices

The programmable logic devices were tested individually. The boot CPLD was programmed with the firmware needed for normal operation and worked successfully. The FPGA was tested by downloading some simple test logic; programming failed. The IC was removed and replaced with one from another production batch, removing the problem. The expansion bus CPLD was programmed simply to pass inputs from the FPGA bus to the expansion bus connector - these signals were used in debugging the FPGA logic.

C.1.3 Microcontroller and memory

The microcontroller was first tested by connecting the microcontroller's JTAG port to the PC's parallel port with the special cable and using the *armtool* software on the PC to read some of the internal registers on the AT91 microcontroller. After communication was established between the PC and the microcontroller, a small program was downloaded into the AT91's internal SRAM and executed. The external memory ICs were tested next; a design flaw in the interface between the microcontroller and the SRAM IC was corrected. The flash memory IC was tested by downloading a compiled application into flash and then booting the microcontroller from flash. The microcontroller serial port successfully transmitted and received data.

C.1.4 Analogue-to-digital converter

The analogue-to-digital converter was tested once the firmware and software were working. The ADC output contained a small dc offset of -8Δ; this was measured by removing the input signal and averaging the ADC output codes. This offset was compensated by programming the Offset register on the THS1408.

C.1.4.1 Stability of the ADC output

The output of the ADC showed some drift over even short periods of time. The oscilloscope did not have the resolution to show small changes in the signal generator output, so it was not possible to ascertain whether this was due to the signal generator or to analogue-to-digital converter itself. The drift is therefore assumed to be caused by the signal generator.

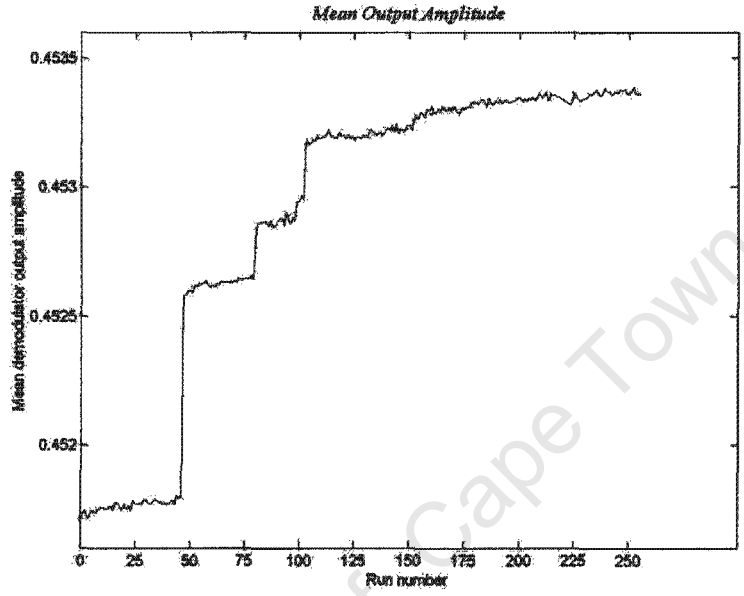

Figure 42. Mean output amplitude of 256 ensembles, N=1

Figure 42 clearly shows the instability of the ADC output - the output amplitude drifted by 2 mV over the few minutes required to capture the data. The plot is of the mean amplitude of 256 ensembles recorded consecutively; each ensemble consists of 128 amplitude measurements. The input signal amplitude was set to 1 V_{pk-pk}, and the coherent integration factor N was 1.

C.2 Firmware tests

The firmware for the boot CPLD and expansion bus CPLD was simple combinatorial logic, and was tested by toggling input signals and observing the output signals.

The FPGA firmware was more complicated. The logic was designed in discrete modules, each of which was tested as a unit before being combined with the other modules.

C.3 Software tests

The microcontroller software was developed in two stages. The first stage tested low-level functions, like booting, access to interrupts, using the serial port, and memory access. The second stage tested the application software itself. The major software functions - controlling the FPGA operation, reading data from the FPGA, and communicating with the PC over the serial link - were tested individually.

The library for writing data to the serial port had two serious flaws. The first was caused by a routine attempting to interpret numeric data as a string: writing 0x00 to the port caused the routine to interpret the character as the null character signifying the string end, and the character was not transmitted. The second was that the blocking, designed to pause transmission if the port was busy, did not work - the character to be transmitted was simply dropped. This problem was circumvented by inserting sufficient delay between successive writes to the serial port.

The Python software, running on the PC, was tested at the same time as the microcontroller software. Special attention was paid to checking the integrity of the data link over the serial port. Data from the microcontroller was transmitted in packets, every byte in the packet being duplicated; the PC software requests retransmission of any packets in which a transmission error is detected.

Bibliography

[1] A. Langman, *The Design of Hardware and Signal Processing for a Stepped Frequency Continuous Wave Ground Penetrating Radar*, Ph.D. thesis, University of Cape Town, 2002.

[2] D.J. Daniels, D.J. Gunton, and H.F. Scott, "Introduction to sub-surface radar," *IEE Proceedings Part F*, vol. 135, no. 4, pp. 277–320, 1988.

[3] L. Peters, J.J. Daniels, and J. Young, "Ground penetrating radar as a subsurface environmental sensing tool," *Proc. IEEE*, vol. 82, no. 12, pp. 1802–1822, December 1994.

[4] A.D.M. Garvin, "Range, resolution and imaging applications of a stepped-frequency continuous wave radar," *Proc. SAIEE*, 1991.

[5] G. Farquharson, "Design and implementation of a 200 to 1600 MHz, stepped frequency, ground penetrating radar transceiver," M.S. thesis, University of Cape Town, 1999.

[6] A.B. Wallis, "The design and implementation of a distributed data capture and processing framework for ground penetrating radar," M.S. thesis, University of Cape Town, 2001.

[7] L.A. Robinson, W.B. Weir, and L. Young, "Location and recognition of discontinuities in dielectric media using synthetic RF pulses," *Proc. IEEE*, vol. 62, no. 1, pp. 36–44, January 1974.

[8] G.F. Stickley, D.A. Noon, M. Cherniakov, and I.D. Longstaff, "Gated stepped-frequency ground penetrating radar," *Journal of Geophysics*, vol. 43, pp. 259–269, March 2000.

[9] G. Guirong, Z. Zhaowen, and W. Feixue, "Mixer-free all digital quadrature demodulation," in *Proc. ICSP '98*, 1998.

[10] W.M. Waters and B.R. Jarrett, "Bandpass signal sampling and coherent detection.," *IEEE Tr. Aerospace and Electronic Systems*, vol. 18, no. 4, pp. 731–736, November 1982.

[11] K.C. Ho, Y.T. Chan, and R. Inkol, "A digital quadrature demodulation system," *IEEE Tr. Aerospace and Electronic Systems*, vol. 32, no. 4, pp. 1218–1227, October 1996.

[12] J.E. Eklund and R. Arvidsson, "A 10b 120 MSample/s multiple sampling, single conversion CMOS A/D converter for I/Q demodulator," in *ISSCC'96 Digest of Technical Papers*. Internation Solid State Circuits Conference, 1996.

[13] J.F. Clare and D.R. White, "Noise in measurements obtained by sampling," *Meas. Sci. Technol.*, vol. 3, pp. 1–16, 1992.

[14] F.G. Stremler, *Introduction to Communication Systems*, Addison-Wesley, 3rd edition, 1990.

[15] A.V. Oppenheim and R.W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, 1989.

[16] R.M. Gray and D.L. Neuhoff, "Quantization," *IEEE Tr. Information Theory*, vol. 44, no. 6, October 1998.

[17] A.J. Acosta, E. Peralias, A. Rueda, and J.L. Huertas, "A VHDL behavioural model for pipeline ADCs," in *Proc. International Workshop on ADC Modelling and Testing*, September 1999, pp. 35–39.

[18] K. Folkesson, J.-E. Eklund, C. Svennson, and A. Gustafsson, "A matlab-based ADC model for RF system simulations," in *Proceedings GHz2000*, March 2000.

[19] D. Goren, E. Shamsaev, and I.A. Wagner, "A novel method for stochastic nonlinearity analysis of a CMOS pipeline ADC," in *DAC 2001*. June 2001, pp. 127 – 132, ACM.

[20] F. Maloberti, P. Estrada, P. Malcovati, and A. Valero, "Validation of data converter specifications with behavioural modeling simulations," *Measurement*, vol. 31, pp. 231 – 245, 2002.

[21] F.H. Irons, D.M. Hummels, I.N. Papantonopoulos, and C.A. Zoldi, "Analog-to-digital converter error diagnosis," in *IEEE Instr. Meas. Technology Conference*, June 1996.

[22] R.H. Walden, "Performance trends for analog-to-digital converters," *IEEE Communications*, pp. 96 – 101, February 1999.

[23] J. Roychowdhury and A. Demir, "Estimating noise in RF systems," Tech. Rep., Bell Laboratories.

[24] P. Horowitz and W. Hill, *The Art of Electronics*, Cambridge University Press, 2nd edition, 1989.

[25] Analog Devices, "AD603 low noise, 90 MHz variable-gain amplifier," Datasheet, Analog Devices, Inc., 2000.

[26] D. Hummels, "Linearization of ADCs and DACs for all-digital wide-bandwidth receivers.," Tech. Rep., Univ. Maine, 1999, IMEKO TC-4 Symposium on Development in Digital Measuring Instrumentation and 4th Workshop on ADC Modeling and Testing.

[27] P. Carbone and D. Petri, "Mean value and variance of noisy quantized data," *Measurement*, vol. 23, pp. 131 – 144, 1998.

[28] M. Koen, "High speed data conversion," Application Note AB-027A, Burr-Brown Corporation, 1991.

[29] D. Mirra, G.Pasini, P.A. Traverso, F. Filicori, and G. Iuculano, "A finite-memory discrete-time convolution approach for the nonlinear dynamic modelling of S/H-ADC devices," *Computer Standards and Interfaces*, , no. 2166, 2002, http://www.sciencedirect.com/.

[30] B. Brannon, "Overcoming converter nonlinearities with dither," Application Note AN-410, Analog Devices, Inc., http://www.analog.com/.

[31] L.G. Melkonian, "Dynamic specifications for sampling A/D converters.," Application Note 769, National Semiconductor, http://www.national.com/, 1991.

[32] H. Kobayashi, M. Morimura, K. Kobayashi, and Y. Onaya, "Aperture jitter effects in wideband ADC systems," in *26th IEEE Int. Conf. On Electronics, Circuits, and Systems*, 1999, pp. 1705 – 1708.

[33] B. Brannon, "Aperture uncertainty and ADC system performance," Application Note AN-501, Analog Devices, Inc.

[34] T.H. Lee and A. Hajimiri, "Oscillator phase noise: A tutorial," *IEEE J. Solid-State Circuits*, vol. 35, no. 3, pp. 326 – 336, March 2000.

[35] K. Kundert, "Modelling and simulation of jitter in pll frequency synthesizers," Tech. Rep., Cadence Design Systems, 1998.

[36] A. Demir, A. Mehrotra, and J. Roychowdhury, "Phase noise in oscillators: A unifying theory and numerical methods for characterisation," in *Proc. DAC98*, 1998.

[37] A. Hajimiri and T.H. Lee, "A general theory of phase noise in electrical oscillators," *IEEE J. Solid-State Circuits*, vol. 33, no. 2, pp. 179–194, 1998.

[38] J.V. Adler, "Clock-source jitter: A clear understanding aids oscillator selection," *EDN Magazine*, February 18 1999, http://www.ednmag.com/.

[39] R.M. Gray, "Quantization noise spectra," *IEEE Tr. Information Theory*, vol. 36, no. 6, pp. 1220 – 1244, November 1990.

[40] J. Tsui, *Digital Techniques for Wideband Receivers*, Artech House, 1995.

[41] D. Bellan, A. Brandolini, and A. Gandelli, "ADC nonlinearities and harmonic distortion in FFT test," in *IEEE Instr. Meas. Technology Conference*, May 1998.

[42] R.M. Gray and T.G. Stockham, "Dithered quantizers," *IEEE Tr. Information Theory*, vol. 39, no. 3, pp. 805 – 812, May 1993.

[43] B. Widrow, I. Kollar, and M-C. Liu, "Statistical theory of quantization," *IEEE Tr. Instr. Meas.*, vol. 45, no. 2, pp. 353–361, April 1996.

[44] P.J.B. Koeck, "Quantization errors in averaged digitized data," *Signal Processing*, vol. 81, pp. 345 – 386, 2001.

[45] F. Adamo, F. Attivissimo, N. Giaquinto, and M. Savino, "Measuring the static characteristic of dithered A/D converters," *Measurement*, vol. 32, pp. 231 – 239, 2002.

[46] T.C. Hofner, "Measure INL and DNL for high-speed ADCs," *Microwaves and RF*, August 2000.

[47] S. S. Awad, "Analysis of accumulated timing jitter in the time domain," *IEEE Tr. Instrumentation and Measurement*, vol. 47, no. 1, pp. 69 – 74, Feb 1998.

[48] "www.python.org," http://www.python.org/.

[49] Texas Instruments, "THS1408 8 MSPS DSP compatible analog-to-digital converter," Datasheet SLAS248A, Texas Instruments Inc., http://www.ti.com/, 1999.

[50] Maxim Integrated Products, "www.maxim-ic.com," 2002, http://www.maxim-ic.com/.

[51] Analog Devices, "www.analog.com," 2002, http://www.analog.com/.

[52] I. Sommerville, *Software Engineering*, Addison-Wesley, fourth edition, 1992.

[53] Hewlett-Packard, *User's Guide, Part Number 33120-90005*, Hewlett-Packard, 1997.

[54] Vectron, "TO-330," Datasheet, Vectron International, http://www.vectron.com/.

[55] C. Van Schaik, "Design, implementation and testing of a real-time microprocessor system for ground penetrating radar," 2000.

[56] GNU, "www.gnu.org," http://www.gnu.org/.

[57] E. Authried, "Armtool," 2000, http://kabel.home.at:5880/cgi-bin/viewcvs.cgi/arm-boot/.