

Development of a Software Defined Radar Environment Simulator

ANTON FRANCOIS JOUBERT



*Thesis presented in partial fulfilment of the requirements for the degree
Master of Science in Electronic Engineering
at the University of Stellenbosch*

SUPERVISOR: Mr G-J van Rooyen

Co-SUPERVISOR: Prof. J.G. Lourens

April 2005

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Abstract

The development of a radar system is a complex process, requiring a substantial amount of testing. In general, these tests are performed using field trials. Such trials are expensive, and their exact conditions can never be repeated. A radar environment simulator allows for repeatable testing of the majority of a radar's functionality; however, most commercial products are not cost effective. In this thesis we investigate the various approaches to modelling targets and the littoral environment, and then develop a low-cost, largely software defined simulator. This simulator is capable of generating real-time video signals for a monopulse doppler tracking radar. The core simulation routines reside in an extensible software framework which is populated with simple target and clutter models for a prototype simulator. Closed loop tracking tests verify the operation of the prototype simulator.

Opsomming

Die ontwikkeling van 'n radarstelsel is a ingewikkeld proses wat verskeie stelseltoetse benodig. Veldtoetse, wat gewoonlik gebruik word, is 'n duur proses, en die toestande is nooit presies dieselfde nie. 'n Radar-omgewingsimulator kan 'n groot deel van 'n radar se funksionaliteit herhaaldelik toets. Daar is egter weinig koste-effektiewe kommersile produkte vir hierdie doel beskikbaar. Hierdie tesis ondersoek verskeie teiken- en omgewingsmodelle, waarna 'n lae-koste, meestal sagteware-gedefinieerde simulator ontwikkel word. Hierdie simulator genereer intydse videoseine vir 'n enkelpuls-dopplervolgingsradar. Die kernfunksionaliteit van die simulator is in sagteware gesetel, en basiese teiken- en omgewingsmodelle is geïmplementeer. Die prototipe-simulator word getoets deur dit aan die radar self te koppel, en dan die teiken-opsporing en -volging te verifieer.

Acknowledgments

I would like to thank those who helped make this thesis possible:

- My university supervisor, Gert-Jan van Rooyen, whose suggestions and feedback greatly improved the end result. Thanks for the exuberance and energy you showed every time you visited “radar land”!
- Hennie Jordaan, my mentor at Reutech. Thank you for suggesting not only such an interesting topic, but also for suggesting Stellenbosch — it has been a awesome experience, so far.
- Cornell Leibbrandt, thanks for wading through one of the early drafts — your input was invaluable.
- The DSP Lab “inwoners” — thanks to the new guns, the old guns, and the coffee machine! The MSP battlefield was always entertaining!
- My parents, thanks for never holding me back, and always providing such loving support.
- My Heavenly Father, for blessing me with my talents, and giving me such wonderful opportunities to use them.

Contents

Nomenclature	xii
1 Introduction	1
1.1 The need for a radar environment simulator	1
1.2 Background	1
1.3 Project objectives	2
1.4 System requirements	2
1.5 Contributions	4
1.6 Thesis overview	4
2 Modelling radar target and environmental returns	6
2.1 Introduction	6
2.2 Radar system	6
2.2.1 Positioner and antenna	7
2.2.2 Radar signal processor	9
2.3 Atmospheric effects	10
2.3.1 Refraction	10
2.3.2 Retardation	13
2.3.3 Attenuation	14
2.4 Surface reflections and multipath propagation	16
2.4.1 Propagation mechanisms	17
2.4.2 Modelling	20
2.5 Scattering	24
2.5.1 Target models	24
2.5.2 Sea clutter	32
2.5.3 Ground clutter	40
2.5.4 Volume clutter	45
2.6 Electronic warfare	47
2.6.1 Jamming	47
2.6.2 Chaff	48
2.6.3 Model selection	49
2.7 Simulators	49
2.7.1 Functions	50

CONTENTS

v

2.7.2	Architecture	51
2.7.3	Interfacing	52
2.7.4	Example systems	53
2.8	Summary	55
3	System design	57
3.1	Introduction	57
3.2	Functional analysis and system architecture	57
3.2.1	Functional units	57
3.2.2	Interfaces	59
3.3	Conceptual design	60
3.3.1	System-wide design decisions	60
3.3.2	Software architecture	61
3.3.3	Concept of execution	62
3.3.4	Waveform generator control unit	67
3.3.5	Target generator	67
3.3.6	Clutter generator	73
3.3.7	RSP interface	74
3.3.8	RSP synchroniser	74
3.3.9	Interface design	75
3.4	Summary	77
4	Software implementation	78
4.1	Introduction	78
4.2	Implementation methodology	78
4.3	Simulator core	79
4.3.1	Implementation decisions	79
4.3.2	Reference frame	79
4.3.3	Scenario objects	80
4.3.4	Terrain	82
4.3.5	Scenario	83
4.3.6	Pre-processing	83
4.3.7	Real-time operation	86
4.4	Embedded software	88
4.5	Graphical user interface	89
4.5.1	Communications engine	90
4.5.2	Class structure	90
4.5.3	Logging	90
4.5.4	Modes of operation	91
4.5.5	Message handling	91
4.5.6	Current boresight angle	91

4.5.7	GUI description	92
4.6	Summary	94
5	Hardware implementation	95
5.1	Introduction	95
5.2	Hardware	95
5.2.1	Development board	95
5.2.2	DAC board	96
5.3	Firmware	97
5.3.1	Implementation methodology	98
5.3.2	RSP synchroniser	98
5.3.3	Target generation	99
5.3.4	Clutter generation	103
5.3.5	Nios SOPC	104
5.3.6	Timing	104
5.3.7	Top level	104
5.4	Summary	105
6	System evaluation	106
6.1	Introduction	106
6.2	Test equipment	106
6.3	Target returns	108
6.3.1	Test set up	108
6.3.2	Pulse width	109
6.3.3	Uncompressed pulse range	109
6.3.4	Amplitude	110
6.3.5	Phase	113
6.3.6	Pulse compression	115
6.3.7	Doppler	116
6.3.8	Co-located targets	118
6.3.9	Comparison to real-world target returns	119
6.4	Clutter	120
6.4.1	Test set up	120
6.4.2	Clutter map	120
6.4.3	Range extent	122
6.4.4	Range dependence	124
6.4.5	Mean clutter level	126
6.4.6	Doppler	127
6.5	Tracking	130
6.5.1	Test set up	130
6.5.2	Stationary target	131

CONTENTS

vii

6.5.3 Receding target	134
6.5.4 Cross-range target	137
6.6 Summary	140
7 Conclusion	142
7.1 Research results	142
7.2 Further work	143
7.3 Summary	144
Bibliography	145
A Frequency band letters	153
B Detailed models	154
B.1 Introduction	154
B.2 Radar system	154
B.3 Atmospheric effects	154
B.3.1 Refraction	154
B.3.2 Retardation	154
B.3.3 Attenuation	154
B.4 Surface reflections and multipath	158
B.4.1 Modelling	158
B.5 Scattering	163
B.5.1 Target models	163
B.5.2 Sea clutter	165
B.5.3 Ground clutter	169
B.5.4 Volume clutter	171
B.6 Electronic warfare	172
B.7 Simulators	172
B.8 Summary	172
C Design and implementation details	173
C.1 DAC prototype board	173
C.2 Interface description	176
C.2.1 Message type definitions	176
C.2.2 Message details	177
C.3 Waveform generator block diagrams	183
C.3.1 Top level	183
C.3.2 Return generator	183
C.3.3 Target generator	183
C.3.4 DDS engine	183
C.3.5 Clutter generator	183

List of Figures

2.1	Simplified block diagram of the RTS6400 radar system.	7
2.2	Monopulse antenna patterns for one plane showing the two individual beams, and the resulting sum and difference channels.	8
2.3	Effect of atmospheric refraction - the solid line shows the actual path travelled by the ray. The dotted line indicates a straight line path.	11
2.4	Gaseous absorption versus frequency for oxygen and water vapour (absolute humidity of 7,75g/m ³).	15
2.5	Simulated PPF coverage diagrams showing the lobed structure generated by optical interference at various frequencies and elevation angles. The intensity of the graphs represents the power, in decibels, of a return from that region.	18
2.6	Geometry used for the flat earth approximation model.	21
2.7	Maximum antenna height versus elevation angle for which the flat earth model approximation is valid.	22
2.8	Geometry used for the spherical earth propagation model.	22
2.9	The four spatial regions defined for the RPO and APM hybrid models [9]. .	23
2.10	Geometry for the definition of the scattering coefficients.	34
2.11	Simple precipitation clutter spectrum for rain approaching at 10m/s, measured at X-band.	47
3.1	Functional block diagram of the simulator system, showing system boundaries.	58
3.2	Mapping of functional units to software packages.	61
3.3	System use case diagram.	62
3.4	System sequence diagram.	63
3.5	Pre-processing engine's activity diagram.	65
3.6	Real-time engine's activity diagram.	66
3.7	Waveform generator's activity diagram.	67
3.8	Target generator functional diagram.	68
3.9	Example of an ambiguity diagram for a linear FM chirp.	72
4.1	Class hierarchy for <i>clsScenarioObject</i>	80
4.2	Collaboration diagram for <i>clsSensor</i>	81
4.3	Collaboration diagram for <i>clsScatterer</i>	81
4.4	Truncated collaboration diagram for <i>clsScenario</i>	83

LIST OF FIGURES

ix

4.5	Include dependency graph for waveform generator main program.	88
4.6	Truncated collaboration diagram for the graphical user interface.	90
4.7	Screenshot of the graphical user interface.	92
4.8	Other tabs available in the graphical user interface.	93
5.1	FPGA development board, with custom DAC prototype board (bottom right).	96
5.2	Block diagram of the waveform generator firmware.	97
5.3	State diagram for the RSP pattern synchroniser.	99
5.4	State diagram for the <i>ReturnGenDelay</i> module.	101
5.5	Measured frequency accuracy for the DDS pipeline, simulated in <i>MATLAB</i> . .	102
6.1	Simplified block diagram of the RSP's processing chain.	107
6.2	Annotated example of the DCM software's waterfall trace.	107
6.3	DCM's AR trace showing the result for the uncompressed pulse range test (Section 6.3.3).	110
6.4	DCM's 2-D waterfall trace showing the result for the amplitude test (Section 6.3.4). Test target amplitude is 0,1.	111
6.5	DCM's 2-D waterfall trace showing the result for the amplitude test (Section 6.3.4). Test target amplitude is 0,05.	112
6.6	DCM's surface trace showing the I and Q channel output for the phase test (Section 6.3.5).	114
6.7	Measured pulse-to-pulse phase increment and error for the phase test (Section 6.3.5).	114
6.8	Output for the pulse compression test (Section 6.3.6).	116
6.9	Doppler test output for all doppler bins (Section 6.3.7). The numbers in the top left corners indicate the doppler bin in which the target should be. . . .	117
6.10	Waterfall trace for co-located targets experiment (Section 6.3.8).	118
6.11	Waterfall trace showing the returns from both a real target and a simulated target.	119
6.12	The scenario map with the wind speed in parenthesis (top), and a contour plot showing the mean scattering coefficients after pre-processing (bottom). .	121
6.13	Scenario set up for clutter range extent test (the cross-hatched region represents forest, which results in the clutter).	122
6.14	2-D waterfall trace for clutter range extent test (Section 6.4.3).	123
6.15	Scenario set up for clutter range dependence test (the hatched region on the right represents sea).	124
6.16	2-D waterfall trace for clutter range dependence test (Section 6.4.4), overlayed with a curve proportional to $1/R^2$	125
6.17	Scenario set up for the mean clutter level test (the cross-hatched region at the top represents farmland, and the lower hatched region, sea).	126

LIST OF FIGURES

x

6.18	Two-dimensional waterfall trace for clutter means test (Section 6.4.5), with curves showing $1/R$ and $1/R^2$ amplitude drop off.	127
6.19	Waterfall trace for clutter doppler test (Section 6.4.6).	128
6.20	Waterfall trace for real sea clutter (there is a target at 1700 m).	129
6.21	Scenario map showing the trajectories followed by the three tracking verification targets (all targets remain at the same height as the radar).	130
6.22	Comparison of simulated and measured range and velocity tracking results for the static target (Section 6.5.2).	132
6.23	Comparison of simulated and measured angular tracking results for the static target (Section 6.5.2).	133
6.24	Comparison of simulated and measured range and velocity tracking results for the receding target (Section 6.5.3).	135
6.25	Comparison of simulated and measured angular tracking results for the receding target (Section 6.5.3).	136
6.26	Comparison of simulated and measured range and velocity tracking results for the cross-range target (Section 6.5.3).	138
6.27	Comparison of simulated and measured angular tracking results for the cross-range target (Section 6.5.3).	139
B.1	Geometry for far-field measurements.	164
C.1	Schematic for DAC prototype board.	174
C.2	DAC prototype board layout. The upper image shows the top layer, and lower one the bottom layer (mirrored).	175
C.3	Waveform generator top level block diagram.	184
C.4	Return generator top level block diagram.	185
C.5	Target generator block diagram.	186
C.6	Block diagram for the DDS engine.	187
C.7	Block diagram for the clutter generator.	188

List of Tables

2.1	Values for the surface refractivity N_s and the constant c_e for various models.	12
2.2	The Swerling cases for target RCS fluctuation.	29
3.1	Target descriptor fields with bit lengths and the corresponding ranges.	73
3.2	Comparison of various communication protocols.	77
5.1	Burst buffer address decoding	100
A.1	British and United States radar frequency bands, from [57, pp. 553]	153
B.1	Two-way atmospheric attenuation coefficients for Barton model, from [54, p. 242].	156
B.2	Values for the water content M in fog, in terms of the visibility	158
B.3	Relative permittivity ϵ_r and conductivity σ_e , from [54, p. 184].	159
B.4	The Beaufort wind scale, from [57, p. 195].	165
B.5	Minimum slant range to surface for a grazing angle $\Psi = 10^\circ$, with $k = 4/3$ for various radar heights.	169
B.6	Weibull shape parameter b for various terrains, from [57, p. 193].	170
B.7	Composite K-distributed terrain model parameters for various terrains measured at X-band, from [45].	171
B.8	Empirically determined constants for relating the reflectivity factor Z to the precipitation rate R_r .	171
B.9	Description of precipitation rates.	172

Nomenclature

Acronyms

ACF	Autocorrelation function
APM	Advanced propagation model
AR	Auto-regressive
CAD	Computer-aided design
CCIR	Comité Consultif des International Radiocommunications
CDF	Cumulative distribution function
CPU	Central processor unit
CFAR	Constant false alarm rate
CORDIC	Co-ordinate rotation digital computer
CRPL	Central Radio Propagation Laboratory
CW	Continuous wave
DDS	Direct digital synthesis
DFM	Doppler filter module
DMA	Direct memory access
DPC	Digital pulse compressor
DSP	Digital signal processor
DRFM	Digital RF memory
ECCM	Electronic counter-countermeasures
ECM	Electronic countermeasure
EM	Electromagnetic
FDGN	Fractionally differenced Gaussian noise
FFT	Fast Fourier transform
FM	Frequency modulation
FPGA	Field programmable gate array
FU	Functional unit
GIT	Georgia Institute of Technology
GUI	Graphical user interface
HH	Horizontal-horizontal
HRR	High resolution radar
I	In-phase
IF	Intermediate frequency

IFS	Intermediate frequency sampler
I/O	Input/output
IP	Internet protocol
LAN	Local area network
LUT	Lookup table
ML	Maximum likelihood
NATO	North Atlantic Treaty Organisation
NG	Non-central chi-squared gamma
PC	Personal computer
PCB	Printed circuit board
PDF	Probability density function
PE	Parabolic equation
PIO	Parallel input/output
PPF	Pattern propagation factor
PPI	Plan position interface
PRI	Pulse repetition interval
PRF	Pulse repetition frequency
Q	Quadrature
RAM	Random access memory
RBF	Radial basis function
RCS	Radar cross-section
RES	Radar environment simulator
REX	Receiver/exciter
RF	Radio frequency
RMS	Root-mean square
RPO	Radio physical optics
RRS	Reutech Radar Systems
RSP	Radar signal processor
SDK	Software development kit
SDP	System data processor
SI	Système International [d'Unités]
SIRP	Spherically invariant random processes
SOPC	System on programmable chip
SRAM	Static RAM
SDRAM	Synchronous dynamic RAM
STL	Standard template library
TCP	Transmission control protocol
TDM	Time division multiplexing
TPEM	Terrain Parabolic Equation Model
UDP	User datagram protocol

VCL	Visual component library
VHDL	VHSIC hardware design language
VHSIC	Very high speed integrated circuit
VV	Vertical-vertical
Δ	Difference
Σ	Sum (channel)
Δ_{AZ}	Azimuth difference (channel)
Δ_{EL}	Elevation difference (channel)

Variables

a_e	Effective earth's radius
c	Speed of light in free space
h_r	Height of the radar antenna above the earth's surface
h_t	Height of the target above the earth's surface
t	Time
ω	Angular frequency
$\log x$	$\log_{10} x$
$\ln x$	$\log_e x$
R_r	Rainfall rate
R	Slant range
f	RF frequency
λ	RF wavelength
θ	Elevation angle
θ_d	Depression angle
Ψ	Grazing angle
ϕ	Azimuth angle
$\Delta\theta$	Elevation beamwidth
$\Delta\phi$	Azimuth beamwidth
τ	Pulse width

Operations

$[x]$ ceiling function (smallest integer $\geq x$)

Chapter 1

Introduction

1.1 The need for a radar environment simulator

The problem addressed by this project is that of testing a specific radar signal processor (RSP) in a simulated scenario. Such scenarios will include a number of targets following certain trajectories as well as environmental clutter from terrain and meteorological conditions. Use of a simulator allows the RSP to be tested repeatedly in complex scenarios without the expense and unpredictability of field trials.

The specific system with which this simulator will be used is the RTS6400, developed by Reutech Radar Systems (RRS). The RTS6400 system is a monopulse doppler tracking radar.

The simulator will overcome the existing shortcomings with the RSP's built-in testing:

- Only two point test-targets, with specific ranges and velocities, can be generated.
- The targets' range rates and velocities cannot be matched, they are thus considered false targets by higher level processing.
- The test targets are ideal — amplitude and phase variations, as seen in the real-world, are absent.

Many commercial simulators are available, with digital radio frequency memories (DRFMs) at the top-end. The problem with these simulators is that they are prohibitively expensive.

1.2 Background

Radar operate on the following principle: a radio frequency (RF) pulse is transmitted from the radar's antenna and is reflected by any object in the radar's beam (e.g. an aircraft or a mountain). A small amount of the reflected RF energy returns to the receiving antenna. The round trip time is proportional to the range of the target. The returning signal is frequency shifted in proportion to the target's velocity (the doppler effect). A classic text on the subject of radar is by Skolnik [77]. A more readable introduction is found in [50].

Simulating a radar return requires that a number of different real-world effects be modelled. Atmospheric effects include refraction and attenuation. The transmitted energy is scattered by the facets of a target, sea waves (sea clutter), the ground and objects on it (ground clutter). The jamming signals generated by electronic warfare (EW) systems are another source of radar returns.

Models for these effects, which are typically of a statistical nature, are well documented in the literature. Chapter 2 will expound on the various models and important aspects thereof.

Radar environment simulators (RESSs) have a variety of applications, each requiring different levels of return signal fidelity [53]. The simplest models are applicable to operator training, while the most complex are needed for system evaluation and EW simulation. Simulators used for in-house radar system testing require a medium degree of model fidelity.

1.3 Project objectives

At the outset of this study, it was required that a number of goals be fulfilled:

- A theoretical study of the existing models for radar target and environmental returns, and the necessary processing techniques to generate them, must be done.
- The pertinent models for the specified radar system must be selected, according to the requirements set out in the following section.
- A functional analysis and conceptual design of the simulator must be performed.
- A prototype simulator must be successfully implemented.

It is important to note that the prototype system is not meant to be a fully featured, high fidelity radar environment simulator. Rather, the emphasis is on creating a software framework into which models of all the effects can be added, with any desired level of fidelity. The design of the prototype must thus consider all the possible environmental effects, but the implementation need only use simple models for proof of concept.

1.4 System requirements

The system requirements are listed below.

- The platform dynamics (i.e. trajectories) for all objects in the scenario will be known prior to simulation.
- The maximum speed for an object is $1715 \text{ m/s} \approx \text{Mach } 5$ (the majority of jets and missiles fly slower than this).
- The radar will not move during the simulation (in order to simplify the clutter implementation).

- The radar’s boresight angle will be provided at run-time via a TELNET link over Ethernet [68]. The simulator must use this information to determine the parameters of the targets and clutter currently in the beam.
- There can be a maximum of eight targets visible to the radar at one time, with at most two targets in the same range cell (at least two targets are necessary to test the radar’s ability to distinguish between co-located targets).
- Environmental modelling must be contained in software as far as possible. Simple models may be used for the prototype system, but an allowance must be made for implementing more complex models.
- Normal targets generated by the simulator must lie in the correct range and doppler bins, and the range rate must be correlated to the doppler speed.
- The simulator must be capable of generating static test targets at an arbitrary range and in an arbitrary doppler bin.
- A target’s maximum range will be limited to the unambiguous range defined by the current pulse repetition interval (PRI).
- The simulator’s maximum range must be at least 200 km, i.e. a maximum PRI of 1,33 ms (for applicability to a wide range of terrestrial radar systems).
- The RSP will use a single waveform pattern for the duration of the simulation.
- The waveform pattern’s details — number of bursts, number of pulses per burst, and PRI per burst — will be provided. There will be a maximum of four bursts in a pattern, and a maximum of 32 pulses in a burst (for an explanation of the waveform details see Section 2.2.2, p. 9).
- The pulse is a phase modulated signal with a maximum length of $12,8 \mu\text{s}$ — the base-band phase modulation will be provided. The system must be capable of changing the phase modulation at run-time.
- The system interfaces directly with the RSP’s intermediate frequency sampler — it must provide three analogue output channels (Σ , Δ_{AZ} and Δ_{EL} — see Section 2.2.1). All three channels must be amplitude and phase matched, and have an output impedance of 50Ω .
- The maximum output frequency is 15 MHz, and the peak power must be less than 30 dBm. Filtering will be provided by the RSP.
- The RSP’s clock and pulse-start timing signals will be provided as digital logic inputs to the system.

- The simulator must allow a scenario to be set up. The scenario must be visible in the user interface, and must be updated during playback of the simulation.
- The user must be able to preview the scenario via the user interface.
- The user interface must allow the simulator's settings to be changed at run-time.
- Test target set up must be available via the user interface.
- The software must run on a personal computer (PC) under Microsoft Windows XP (a popular and widely available operating system).
- In order to reduce the simulator's hardware cost, an existing field programmable gate array (FPGA) development board (*Nios Development Board, Stratix Edition*, from Altera) must be used to interface to the RSP.

1.5 Contributions

In accomplishing the objectives, a number of contributions were envisaged:

- A summary of the state of the art in radar environment simulation and target generation.
- The development of a generic, largely software defined, simulator — thus enabling models to be easily modified, exchanged and added. It also allows the simulator to be adapted to wide range of radar systems.
- A solution which provides proof of concept for low cost, repeatable testing of a specific radar system.

1.6 Thesis overview

An extensive literature study is provided in Chapter 2. We consider the various aspects affecting the generation of target and clutter returns. The most important factors are identified and relevant models are selected. The majority of the model details are given in Appendix B.

In Chapter 3, the system design presented. A functional analysis is done, followed by the conceptual design. The system architecture and functional allocation are discussed in detail.

The implementation starts with the simulator's core software and then moves outward to the user and radar interfaces — this is presented in Chapter 4. The hardware implementation is detailed in Chapter 5.

A number of tests were run to verify that the simulator works with the RTS6400 tracking radar system. Target and clutter returns are considered separately. The final closed loop

tests show tracking of a simulated target. All of these results are presented and discussed in Chapter 6.

The concluding chapter, Chapter 7, summarises the achievements of this work and suggests further research topics.

Chapter 2

Modelling radar target and environmental returns

2.1 Introduction

This chapter discusses the various effects which need to be considered when simulating radar returns. Before delving too deeply into the literature, we discuss the basics of the target radar system. This provides a context in which to review the various modelling schemes.

We consider the effects due to the atmosphere, scattering from various sources and finally, electronic warfare. For each effect, the principle is discussed; a synopsis of the literature is presented; and then the most applicable model is selected. The chapter ends with an examination of the current trends in simulator system design.

2.2 Radar system

The RTS6400, developed by Reutech Radar Systems, is a monopulse doppler tracking radar system. A simplified system block diagram is shown in Figure 2.1.

Under control of the system data processor (SDP), the positioner aims the antenna beam at the target. An analogue pulse from the RSP is passed to the receiver/exciter (REX) where the pulse is upconverted from the intermediate frequency (IF) to X-band (8–12 GHz) (Appendix A lists the frequency band letters). Prior to transmission, the X-band signal is amplified by the transmitter. The transmitted pulse then interacts with the environment and any returning RF energy is captured by the antenna. The echoes are downconverted to IF by the REX and then sampled by the RSP to generate in-phase (I) and quadrature (Q) samples. The RES aims to reproduce the returns that would be received by the RSP — as such, the RES's analogue output is injected directly into the RSP's IF sampler (IFS). The subsystems are discussed in more detail in the following sections.

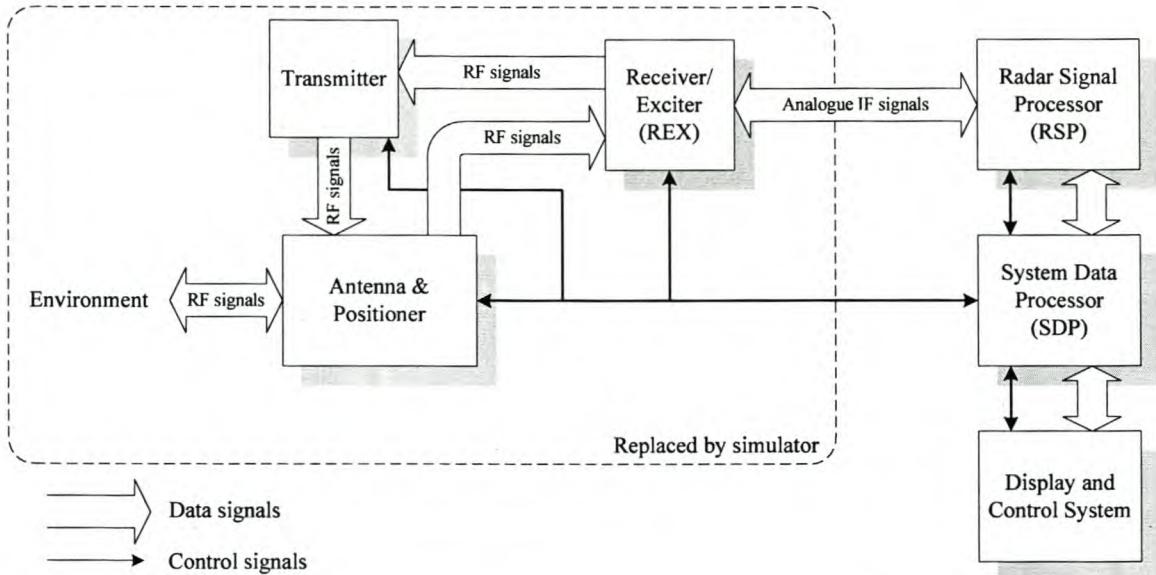


Figure 2.1: Simplified block diagram of the RTS6400 radar system.

2.2.1 Positioner and antenna

A monopulse tracking radar transmits multiple beams simultaneously so that the angular position of the target within the beam can be accurately determined with a single pulse. The antenna patterns for one plane are depicted in Figure 2.2.

On reception, the two beams are added together to generate the sum (Σ) channel. Addition causes the resultant pattern to have a broad peak. The Σ channel is used to detect the presence of a target and measure its range and speed (the wide beam makes it unsuitable for angular estimation). The difference (Δ) channel's gain has a steep slope with a zero-crossing at the beam's centre. The Δ channel generates an error signal that tends to zero when the target is positioned exactly in the middle of the beam. The sign of the error signal indicates the direction that the antenna should move in.

In order to determine both the azimuth and elevation angles, a total of four beams are transmitted (two for each plane). When receiving, the antenna has three output channels: Σ , the azimuth difference (Δ_{AZ}) and the elevation difference (Δ_{EL}).

The antenna is relatively simple to model using the measured antenna pattern for each channel. Every return is multiplied by the complex gain corresponding to its angle of arrival. For range calculations, the usual approach is to include the antenna's pattern in the pattern-propagation factor (see Section 2.4). The RTS6400's antenna is a Gregorian Cassegrain double reflector which generates a pencil-beam with very low sidelobes.

The positioner has two degrees of freedom — azimuth and elevation — whereby the radar beam can be directed towards any point of interest. When tracking, the system drives the positioner so that the target stays approximately in the centre of the beam. Very tight

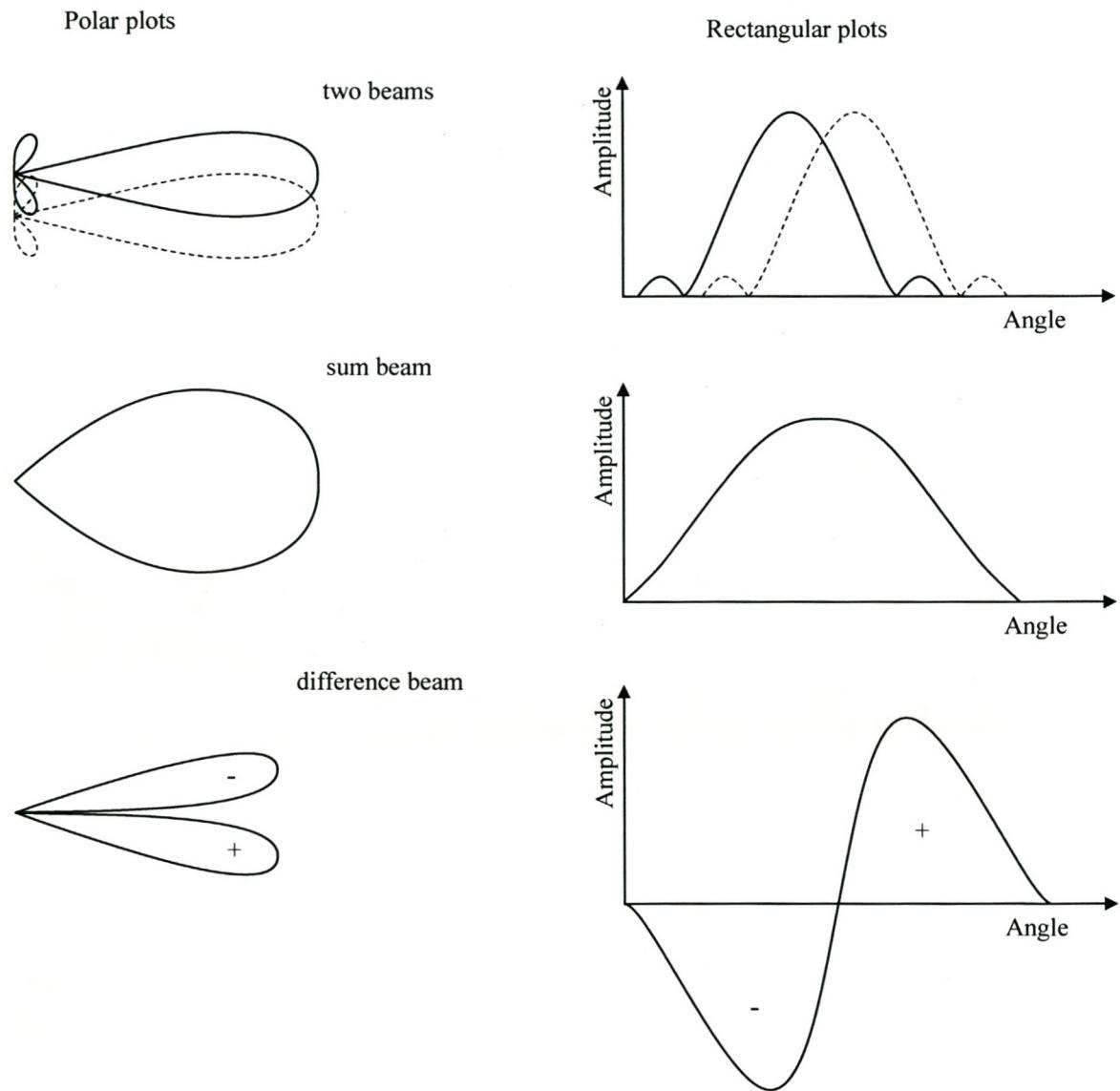


Figure 2.2: Monopulse antenna patterns for one plane showing the two individual beams, and the resulting sum and difference channels.

boresight control is not needed, as the boresight position is accurately known (feedback from the positioner), and the monopulse measurements accurately indicate how far off boresight the target is. An advantage of this approach is the radar's reduced susceptibility to angle pull-off (an electronic warfare technique — see Section 2.6.1). Another very important function of the positioner and SDP is stabilisation — the antenna is aimed in the same direction (in earth relative co-ordinates) regardless of the radar platform's motion. This simplifies the modelling of the platform as its swaying can be neglected.

2.2.2 Radar signal processor

The radar signal processor estimates the target position from the radar returns. This information is used by the SDP to track the desired target. The exact processing performed by the RSP is not important in the design of the simulator. However, it must be noted that the RSP is a coherent processor. This means that the sampling of the return signal must be synchronised to the generation of the transmitted signal. The RSP uses both the phase and amplitude of the return, representing signals as complex, or I and Q, data.

In order for the simulator to generate coherent data, the RSP's timing signals are required for synchronisation. The length of each PRI is defined by the RSP's waveform pattern. The signal transmitted during one PRI is termed a pulse. A pattern typically consists of a number of pulses grouped together in bursts (the pulses in a burst have the same PRI). The RSP is capable of using a number of different waveform patterns — these are chosen according to the current scenario. To simplify the simulator's implementation, the RSP will be limited to a single pattern for the duration of the simulation run.

As an example, consider a pattern that contains two bursts. The first burst generates ten pulses at a certain pulse repetition frequency (PRF). The second burst, containing say fifteen pulses, is then transmitted at a different PRF. This technique has a number of important advantages [78, ch. 3], [90, ch. 9].

- The pulses can be integrated, per burst, to help reduce noise effects (this is known as integration gain).
- Doppler information can be obtained from a burst's pulses.
- Range and doppler ambiguity can be resolved by comparing measurements from both bursts.

The simulator requires knowledge of the pattern, including the pulse characteristics (i.e. phase modulation) currently being used by the RSP, so that the returns can be generated correctly. As the pulses are generated digitally, it is possible for the simulator to use an exact replica of the transmitted signal. The specific characteristics available to the RTS6400 will not be discussed.

2.3 Atmospheric effects

Once the pulse has been transmitted, the first effect to be modelled is that of the atmosphere. The atmosphere consists of various layers, each with different characteristics.

- The troposphere extends from the surface to about 14 km — this is where the weather occurs.
- Next is the stratosphere, up to about 50 km — this is a relatively stable region that includes the ozone layer.
- The mesosphere and then ionosphere are found above 50 km — here EM propagation is significantly affected by the presence of free electrons.

We are primarily concerned with short to medium range systems (excluding those used for meteorology), and will not consider the mesosphere and ionosphere further.

The two most important factors to be considered are refraction and attenuation — we discuss these and a third factor, retardation, in the following sections. Dispersion is also present, but the effect is negligible in all but specialist wideband systems [50, ch. 8].

An important simplification occurs due to the reciprocity principle of electromagnetic waves. If the same antenna is used for transmission and reception (monostatic radar), then the path of the RF energy only has to be analysed in one direction.

2.3.1 Refraction

Principle

Refraction refers to the process whereby an electromagnetic wave changes direction at the interface between two media in which its velocity of propagation differs. Initially it may seem that this does not apply to an RF pulse that only travels through the air, however the atmosphere does not have a constant index of refraction n . The changes are small, with n near unity, so that it is more convenient to use the refractivity N , measured in N units, and defined as

$$N = (n - 1) \cdot 10^6. \quad (2.1)$$

Generally, the atmosphere's refractivity decreases with altitude. This leads to a bending of the beam, back towards earth — see Figure 2.3. If a radar assumes that the beam travelled in a straight line (free-space propagation), the target's position will be calculated incorrectly. By the same token, the simulator must model this effect in order to depict the environment correctly.

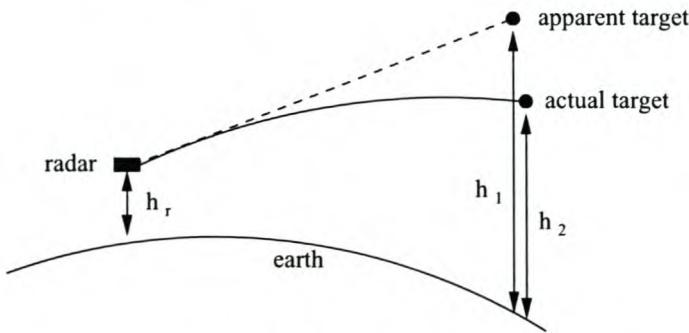


Figure 2.3: Effect of atmospheric refraction - the solid line shows the actual path travelled by the ray. The dotted line indicates a straight line path.

Modelling

The variation in refractivity is determined by a number of environmental factors: the air temperature, pressure and water content. The relationship is as follows ([78], p. 24-2)

$$N = 77,6 \left(\frac{P}{T} \right) + 3,73 \cdot 10^5 \left(\frac{e}{T^2} \right), \quad (2.2)$$

where the temperature T is in degrees Kelvin, and both the total atmospheric pressure P , and the partial pressure of water vapour e are in millibars (1 millibar = 100 Pa). This has been shown to be accurate up to 30 GHz, with a standard error of 0,5 percent [15, 79].

It is seldom practical to measure the values of P , T and e at different altitudes. Attempts have been made to find simpler, more generally applicable models.

Linear or effective-earth's-radius model. By increasing the apparent size of the earth, the curvature of the rays can be absorbed into the curvature of the larger earth's surface. With this larger earth, straight lines can be used to trace the radio rays. Before the widespread availability of fast digital processors, this approach was extremely popular for its computational simplicity.

The effective earth radius a_e is proportional to the actual radius a :

$$a_e = ka. \quad (2.3)$$

Skolnik [78, p. 24-7] has shown that the scaling factor should be $k = 4/3$.

The principal assumption that leads to this model (also known as the constant k model) is that the atmosphere's refractivity varies linearly with altitude. Physically, this is a poor assumption and the result is that this model is only applicable at altitudes below 1 km [78, p. 24-9].

Exponential model. A more physically realistic model of refractivity is one which assumes an exponential decay with altitude:

$$N = N_s \exp(-c_e h), \quad (2.4)$$

Table 2.1: Values for the surface refractivity N_s and the constant c_e for various models.

Model	N_s	c_e [km $^{-1}$]	Region
CRPL	313	0.14386	United States of America
NATO	320	0.1217	Northern Europe
CCIR	289	0.136	Default

where N_s is the refractivity at the earth's surface, c_e is a constant [km $^{-1}$], and h is the height above the surface [km]. Various organisations have determined values for N_s and c_e . In the United States of America, the Central Radio Propagation Laboratory (CRPL) values are used. The North Atlantic Treaty Organisation (NATO) model represents Northern Europe. The Comité Consultif des Radiocommunications (CCIR) has published maps for the world. Table 2.1 shows the default values [57].

When using the exponential model, the path followed by a radio ray must be traced out in an incremental fashion. For each step the refractivity is calculated and used to determine the refraction at the shell boundary. Such a ray trace is a computationally intensive process.

Parabolic wave equation model. Recently, interest in the use of a parabolic approximation to the Helmholtz wave equation has been renewed due to improved measurement and computational ability [30, 40]. The approach was first suggested by Fock in 1946 [35], but an efficient numerical technique — the split-step Fourier algorithm — was only proposed in 1977 [82, 29]. Using this parabolic equation (PE) model, refractivity profiles that vary in both range and altitude can be specified. The model includes the transmission frequency, antenna pattern, altitude, elevation angle and polarization. The earth is assumed to be a finitely conducting sphere.

The effects of a rough surface have been included by Thews and Dockery [84]. Fabbro, *et al.* [32] have extended the technique to include shadowing above Gaussian rough surfaces and modelling of forest cells.

Other methods such as geometric optics [49], coupled mode analysis and combinations thereof can be used with vertically and horizontally dependant refractivity profiles. All of these methods have shortcomings [30]. Geometric optics has problems associated with caustics and is only applicable at high frequencies. Coupled-mode analysis requires a large number of modes and coupling coefficients and is thus very complicated.

The PE model includes all diffractive and refractive effects, and is thus very widely applicable. The PE method has the following advantages:

- Very high fidelity.
- Arbitrary refractivity profiles.
- Implicitly includes multipath effects caused by the earth's surface.

- Simpler to process than coupled-mode analysis.

The disadvantages of the PE method are listed below:

- Much more computationally expensive than geometric optics.
- The angle of elevation is limited to within $10^\circ - 20^\circ$ of the horizontal.
- The increased accuracy is only useful if measured refractivity profiles are available.

The PE model is thus best suited to offline processing of data, when measured refractivity data is available and very high fidelity is required [84]. If anomalous propagation effects such as ducting and tropospheric scattering are to be simulated, then the PE model is also a good candidate [40].

Hybrid models. Using the PE algorithm, the most accurate results can be obtained. However, the model is only applicable for small elevation angles. This suggests a hybrid approach, using different models for different spatial regions. Another advantage of hybrid models is that the use of simpler models, where applicable, reduces the processing time required. These models are discussed in detail in Section 2.4.

Anomalous propagation

In normal conditions, the refractivity gradient decreases at a rate of about $40 N$ units per kilometre. Unusual atmospheric conditions can result in steeper gradients. When the gradient in a layer is more than $157 N$ units/km, the RF energy will experience total internal reflection. The energy becomes trapped in a duct which brings extra coverage at this altitude — the process is thus known as ducting. The unusual condition that leads to ducting is a temperature inversion — that is, the temperature increases with height, instead of decreasing. An example is hot, dry air from a desert that is blown out over the sea. A more detailed description is given by Patterson *et al.* [64].

A refraction model that allows a refractivity versus height profile can model ducting. The exponential and PE models meet this criteria.

Model selection

The prototype simulator only requires simple models, so the simplest model — a constant refractive index, $n = 1$ — is selected. In this case, there is no refraction and trivial geometry can be used for the ray tracing algorithm. By implementing the ray tracing in a single module, a more complex model can be added at a later stage.

2.3.2 Retardation

The refractive index n of the atmosphere, discussed at length in the previous section, is not exactly unity. By the definition of the refractive index, this means that the speed of the

radio waves will not equal the free space speed of light c . By considering all values of n along the path, a range dependent range error ΔR , can be calculated. An approximation can be found by using the path's average refractive index \bar{n} . For a target at a range of R metres, the error is approximately

$$\Delta R = R(\bar{n} - 1) \quad (2.5)$$

As the refractivity decreases with altitude, the error is largest for low altitude targets. Near the earth's surface, the atmosphere's refractive index is about $n = 1,0003$ — the error is thus less than 0,03 percent of the range, and is usually ignored.

For simulation, it would be simple to calculate ΔR . For a linear refractivity gradient $\frac{dN}{dh}$ (assumed by the constant- k model), a radar at height h_r where the refractivity is N_0 , and a target at height h_t , the range error is [50, p. 178]

$$\Delta R = 10^{-6}R \left(N_0 + \frac{(h_t - h_r)}{2000} \cdot \frac{dN}{dh} \right) \quad (2.6)$$

For the more complicated refractivity models that require an iterative ray tracing approach, ΔR could easily be calculated for each step and then summed.

Model selection

As the refractive model selected assumes a constant refractivity of unity, the effect of retardation can be ignored in the simulator. If a more complex refractive model was implemented, then the retardation model would be added to the ray tracing algorithm.

2.3.3 Attenuation

In this section we consider the attenuation of electromagnetic energy as it travels through the atmosphere. Attenuation is due to two effects: absorption and scattering. The two areas of interest are gaseous absorption and weather effects.

Gaseous absorption

Between 100 MHz and 50 GHz, the chief contributors to gaseous absorption are oxygen and water vapour. Their effects at lower frequencies are negligible; higher frequencies are not of interest in a radar context. Figure 2.4 shows the frequency dependence — oxygen absorption peaks at about 60 GHz, and water vapour around 22 GHz. The absorption is also a function of the temperature and pressure for both gasses. In addition, the water vapour absorption is proportional to the absolute humidity.

Constant attenuation model. The simplest way to account for atmospheric absorption is to use a constant attenuation rate, regardless of pressure and temperature changes along the path travelled by the RF energy. As the attenuation is relatively small, and thus the approximation error small, this approach is often used in simulators [64, 53].

The CCIR model is given in Appendix B, p. 154.

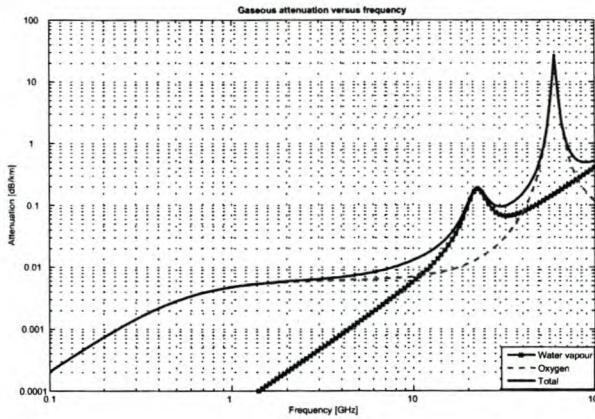


Figure 2.4: Gaseous absorption versus frequency for oxygen and water vapour (absolute humidity of $7.75\text{g}/\text{m}^3$).

Height dependent models. For accurate calculation of the attenuation, a model that depends on pressure, temperature and absolute humidity must be used. Details of a model attributed to Van Vleck, Bean and Abbott is given in Appendix B, p. 154.

Skolnik examines the total gaseous absorption for measured data from two stations in North America [78, p. 24-13]. Frequencies ranging from 100 MHz to 50 GHz and heights up 75 km are considered. For frequencies away from the absorption peaks, the seasonal variation is generally less than 0,001 dB/km. For frequencies above 3 GHz, the absorption decreases exponentially with height.

Zhevankin and Troitskii proposed the biexponential model for attenuation (given in [78]). This model requires empirical factors known as the *scale heights*. For oxygen absorption, the scale height is a simple expression, but for water vapour this is not the case.

To calculate the total radio-path absorption, the absorption model should be evaluated at each step of the RF path. This calculation requires the pressure, temperature and absolute humidity at each step.

A simplification by Barton using an exponential model, is detailed in Appendix B, p. 154.

More advanced models. More advanced models are available for calculating the gaseous absorption of radiation. For example, Warner and Ellingson's model [93], which calculates a number absorption lines very accurately (not just the 1,35-cm line). These lines are then used to determine the attenuation. Such accuracy is unnecessary for radar environment simulation, but is important for studies in climatology.

Weather effects

Meteorological conditions can have a notable effect on the performance of a radar system. The main problems are reduced range due to signal attenuation, and reduced true detection

ability due to increased clutter (i.e. an increase in false alarms). We consider the attenuation caused by clouds, fog and precipitation. These effects are only appreciable at frequencies above 3 GHz [78, p. 24-23]. There is attenuation because water and ice particles absorb and scatter a fraction of the incident RF energy. We discuss the backscatter from hydrometeors in the section on volume clutter (Section 2.5.4, p. 45).

Attenuation by rain. Suspended water droplets have a greater attenuation than predicted by the gaseous absorption discussed in Section 2.3.3. Models for rain absorption are generally given in terms of the rainfall rate R_r . The rate is a function of the liquid-water content of the air and the size of the drops — the latter is proportional to the fall velocity.

A model by Ryde, discussed in [78, p. 24-23], gives the attenuation in decibels per kilometre K_R as

$$K_R = K \int_0^r [R_r(r)]^\alpha dr \quad (2.7)$$

with K a function of frequency, $R_r(r)$ the rainfall rate along the propagation path r , and α a function of frequency (taken as unity). More details are given in Appendix B, p. 156.

Thomson and Riseborough [85] recently developed a high fidelity weather clutter simulator. The precipitation is split into three layers: rain, mixed phase particles, and snow. Clouds, turbulence and wind are also defined at specific heights. Attenuation is determined from the thermodynamic phases, and the size and concentration of precipitation particles. Gaseous attenuation is also included. The details of the models used are not given in the reference.

Attenuation by hail, snow, clouds and fog. Hail and snow were originally thought to have a negligible contribution to attenuation, however the assumption made by Ryde was that the particles were solid. Once the ice particles start melting, they can cause more attenuation than purely liquid particles [78, p. 24-28]. The increased water content in clouds and fog attenuates RF energy.

Models exist for these effects — a discussion can be found in Appendix B, p. 156.

Model selection

As with the refractive effects, attenuation is not considered an important effect for the baseline system and need not be implemented. However, to allow for more complicated models in future, the ray tracing algorithm must return a parameter indicating the amount of attenuation experienced along the path of the ray.

2.4 Surface reflections and multipath propagation

Surface reflections and the resulting multipath propagation effects can have a significant effect on a radar system's coverage. The following sections will clarify the effects. We

first give a physical description of the propagation methods, before examining a number of possible models for these effects.

2.4.1 Propagation mechanisms

A standard atmosphere is a theoretical model, derived from average measurements, that describes the atmosphere's refractivity profile. The standard propagation mechanisms are discussed below. Other mechanisms exist, such as subrefraction, superrefraction and trapping, but these are due to anomalous propagation. These anomalous mechanisms are generally only included in the highest fidelity simulators and will not be examined in detail.

Free-space propagation

A region that is homogeneous, isotropic and loss-free can be termed free space. In free space the EM wavefront spreads uniformly from the antenna — it is thus the simplest form of propagation.

Optical interference and surface reflection

At ranges short of the optical horizon, there may be two paths to a point. The first path is directly through the atmosphere, while the second, slightly longer path, includes a reflection off the earth's surface (see Figure 2.6, p. 21). Due to the difference in path length, and a phase and amplitude change on reflection, the two arriving rays will have different phases. The energy from the two rays add as vectors at the point, resulting in either constructive or destructive interference. As the elevation angle changes, so will the path length difference. As the path length difference goes through multiples of a wavelength, lobes are generated on the coverage diagram. Figure 2.5 shows simulated coverage diagrams (using the pattern propagation factor (PPF) ray optics model — see p. 21).

The surface reflection characteristics are captured in the complex reflection coefficient, defined as [78, p. 2-34]

$$\Gamma = \rho e^{-j\alpha} , \quad (2.8)$$

with ρ the amplitude and α the phase. For the sea, Γ is typically near $1/180^\circ$ in calm conditions, but the amplitude decreases as the sea gets rougher. More detail will be given in the section on modelling (see p. 20).

Diffraction

When opaque or refractive objects shadow a region from an approaching wavefront, part of the energy tends to follow the curve of the surface and is thus diffracted into the shadowed region [64, p. 19]. The lower the frequency, the more the wave is diffracted. In the case of radar, diffraction occurs most notably at the radar horizon. In other words, when the propagating ray's path is just tangential to the earth's surface.

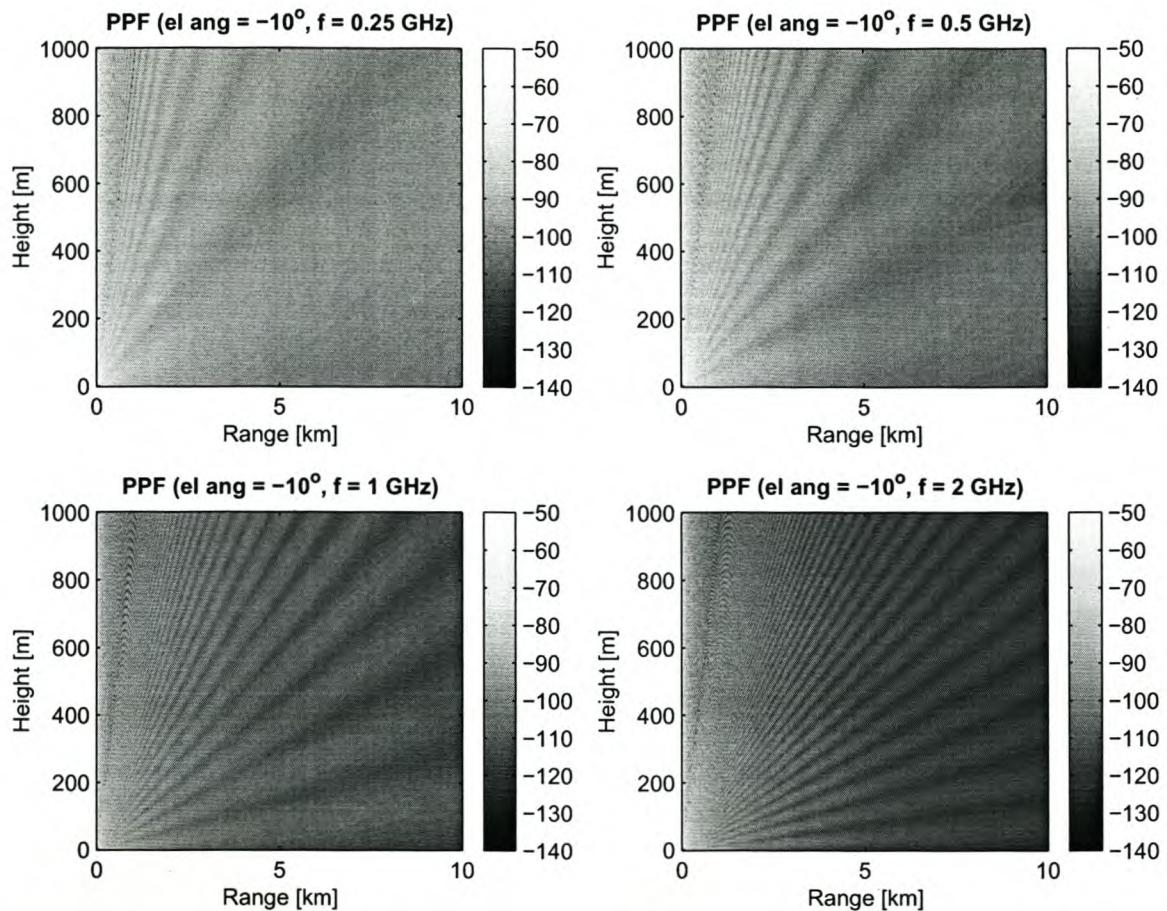


Figure 2.5: Simulated PPF coverage diagrams showing the lobed structure generated by optical interference at various frequencies and elevation angles. The intensity of the graphs represents the power, in decibels, of a return from that region.

Tropospheric scatter

A limited portion of energy may reach far beyond the radar horizon due to tropospheric scatter. The scattering is caused by small inhomogeneities in the atmosphere's refractive index. Tropospheric scatter is understandably weak, and more so at high frequencies.

Anomalous propagation

Subrefractive layers cause an upward bend in the propagating energy, thus decreasing the radar's detection range. In contrast, superrefractive layers result in increased range, often well beyond the radar horizon. More details are available in [64, p. 20].

Pattern propagation factor

The pattern propagation factor (PPF), F , is normally included in the radar range equation (see eq. (2.18), p. 25) to account for the antenna pattern and multipath propagation effects. We will only consider the monostatic case, i.e. $F_t = F_r = F$.

F is defined [78, p. 2-33] as the ratio of the actual field strength E at a point, to the field strength that would occur at the same range in free space in the direction of the antenna's maximum gain, E_o . Symbolically, this is

$$F = \frac{|E|}{|E_o|}. \quad (2.9)$$

Note that F is a voltage ratio, thus it occurs as F^2 in the radar range equation for either transmission or reception. For monostatic radars, the transmission and reception factors are the same, thus the factor appears at F^4 .

When only the propagation mechanisms are taken into account, and not the antenna pattern, we refer to just the *propagation factor*. In the case of an isotropic antenna, the *pattern propagation factor* and the *propagation factor* will be equal. Some authors do not make this distinction.

In a simulator for a coherent radar, the phase of the return is also important. We define the complex value F_c as

$$F_c = f_d + \rho D f_r e^{-j\gamma}, \quad (2.10)$$

- with f_d = antenna pattern factor in direction of direct ray,
- f_r = antenna pattern factor in direction of reflected ray,
- ρ = amplitude of Γ ,
- γ = total phase difference between the direct and reflected paths, [rad]
- D = divergence factor.

It is simple to obtain f_d and f_r , as shown in the following section. For radar range calculations, only the amplitude of F_c is required (this is the usual definition of the PPF we mentioned previously), i.e.

$$F = |F_c|. \quad (2.11)$$

We define F_ϕ as the phase angle of F :

$$F_\phi = \angle F_c = \arctan \left(\frac{\text{Im}\{F_c\}}{\text{Re}\{F_c\}} \right) \quad [\text{rad}]. \quad (2.12)$$

The phase of the resultant signal is that of the direct ray plus F_ϕ . As the signal follows the same path back to the radar, the total additional phase shift, due to multipath, is $2F_\phi$.

Antenna pattern factor. The antenna's pattern factor, $p(\epsilon)$, is the ratio of the radiated electric field at the angle ϵ (relative to the beam axis), to the maximum electric field radiated.

Given that the boresight, which is assumed to be aligned with the beam axis, has an elevation angle of θ_{bs} , the angle of the direct ray relative to the beam axis is then

$$\epsilon_d = \theta_e - \theta_{bs} \quad [\text{rad}] , \quad (2.13)$$

with θ_e the elevation angle [rad]. The angle of the reflected ray relative to the beam axis is

$$\epsilon_r = -\theta_d - \theta_{bs} \quad [\text{rad}] , \quad (2.14)$$

with θ_d the depression angle [rad]. The relative angles ϵ_i , are then used with the antenna's pattern factor to obtain f_d and f_r , i.e.

$$f_i = p(\epsilon_i) . \quad (2.15)$$

2.4.2 Modelling

In the following section we discuss a number of models that can be used to calculate the PPF. The simplest is the flat earth model, next we examine a ray optics approach, and lastly we look at more advanced methods.

Reflection coefficients

A model reported by Leonov and Leonov [54, pp. 184–186], splits the calculation of the reflection coefficient into three parts. The first is the complex Fresnel reflection coefficient $\rho_0 \angle \alpha$. Second is the specular scattering coefficient ρ_s which accounts for the loss of energy due to a rough surface. Third is the vegetation coefficient ρ_v which accounts for the additional losses due to the vegetation. The last two only affect the amplitude of the reflected ray. The three coefficients are combined to obtain the complex reflection coefficient

$$\Gamma = \rho_0 \rho_s \rho_v e^{-j\alpha} . \quad (2.16)$$

In order to calculate the combined complex reflection coefficient, a number of steps must be followed — we defer the details to Appendix B, p. 158.

Flat earth model

The flat earth model applies only to the interference region. The geometry is shown in Figure 2.6. Two criteria must be met in order to use the flat earth approximation. The first [78, p. 2-35] is that the target range R must be much greater than the radar antenna height h_r . This allows the approximation $\theta_e \approx \theta_d$, to be made. The second requirement [78, p. 2-38] is that the following inequality is true

$$\tan \theta_e > 0,001 \sqrt{h_r/3} . \quad (2.17)$$

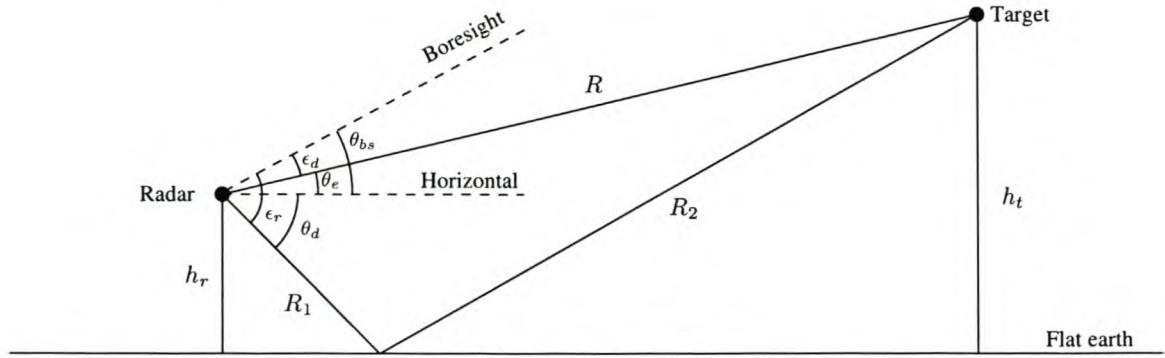


Figure 2.6: Geometry used for the flat earth approximation model.

It is clear from the equation that the target must be at a positive elevation angle. The inequality ensures that the actual curved earth reflection point is close to the plane tangential to the earth at the antenna site. Also, the divergence factor can be approximated as $D \approx 1$. The limit on antenna height is shown in Figure 2.7, as a function of elevation angle. Details of the model are given in Appendix B.

Ray optics (spherical earth)

When taking the earth's curvature into account, the model must consider propagation in the interference, intermediate and diffraction regions. This type of model is far more general than the flat earth model, and provides a medium degree of fidelity. The model we review is given by Leonov and Leonov [54, pp. 180–190]. It is similar to the one given by Patterson *et al.* [64, pp. 83–108], used in the Engineer's Refractive Effects Prediction System (EREPS). However, the EREPS model accounts for anomalous propagation and tropospheric scatter. Both models use the constant k model for refraction. The geometry used is shown in Figure 2.8.

The assumptions for Leonov and Leonov's model are listed below (details are given in Appendix B).

- Absorption losses and anomalous propagation are ignored.
- The target's range is much greater than the path length difference between the reflected and direct rays.
- The distance to the reflection point is much less than the target range, so that the rays are parallel. The elevation angle is thus equal to the grazing angle.
- In the diffraction region: the target is further than the radar horizon, and the path is blocked.

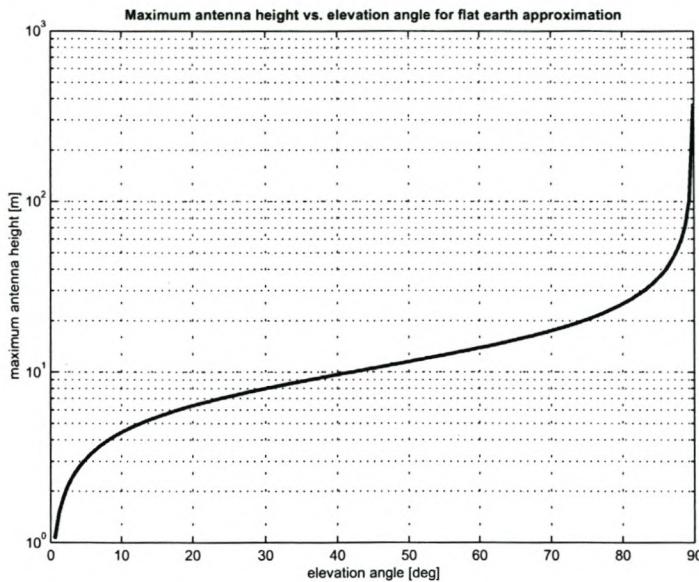


Figure 2.7: Maximum antenna height versus elevation angle for which the flat earth model approximation is valid.

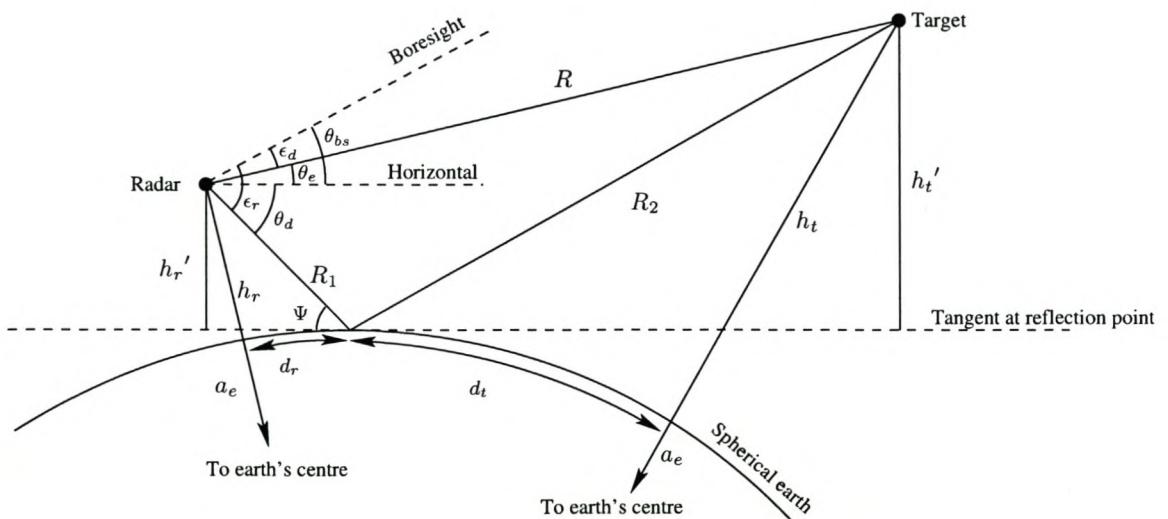


Figure 2.8: Geometry used for the spherical earth propagation model.

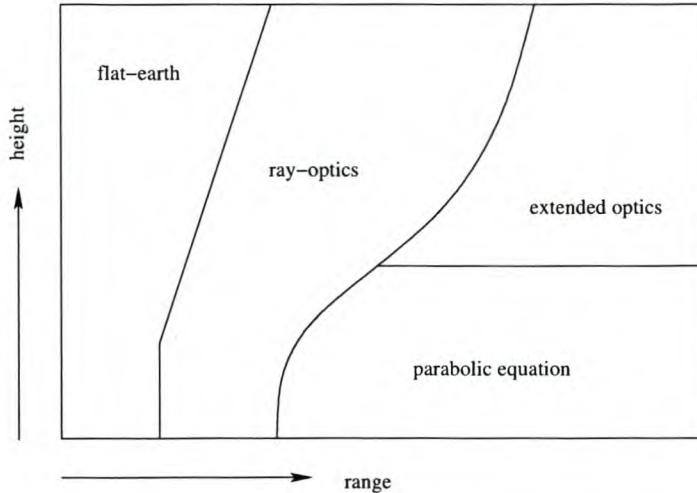


Figure 2.9: The four spatial regions defined for the RPO and APM hybrid models [9].

Advanced methods

A major shortcoming of the flat earth and ray optics techniques is the simplified refractivity profile assumed. The flat earth model ignores refractivity completely, while the ray optics model uses the constant k approximation. The EREPS approach [64], based on the ray optics model, includes additions that allow it to approximate various anomalous propagation mechanisms.

More advanced methods, for essentially arbitrary refractivity profiles, include a ray tracing procedure, coupled-mode analysis, or the parabolic wave equation (PE). These methods were discussed in the section on refraction (p. 11). Reflection effects would need to be included in the ray tracing approach, but they are intrinsic to coupled-mode analysis and the PE algorithm. Advanced models allow for more accurate results, but at the expense of much more processing time.

Hybrid models. Using the PE algorithm, the most accurate results can be obtained, however the model is only applicable for small elevation angles. This suggests the use of a hybrid model that applies different techniques to different spatial regions. Another advantage of hybrid models is that the use of simpler models, where applicable, reduces the processing time required.

The radio physical optics (RPO) model described by Hitney [39, 40] is an example of such a hybrid model. The model uses four spatial regions (see Figure 2.9) with the boundaries chosen so that the solution is continuous. An improved PE method, that includes troposcatter, is applied at low elevations and altitudes. Next an extended-optics approach is used — this is essentially a ray-optics algorithm, initialised by the PE solution. At still higher elevation angles, standard ray-optics are applied. Finally, for short ranges and angles above 5°, a flat-earth model is used.

An improved technique is used by the Advanced Propagation Model (APM) [9]. The APM incorporates the RPO model and the Terrain Parabolic Equation Model (TPEM). Using the TPEM, the terrain structure, and the corresponding dielectric ground constants, can vary with range. This model represents a very good compromise between speed and accuracy.

Model selection

There are essentially two types of models to choose from: the simple PPF models, or the more advanced parabolic equation models. In keeping with the simple approach to the simulator, the PPF method is selected. The software thus requires a module that will calculate the PPF given the radar and scatterer parameters. For implementation it was decided that the PPF would simply be set to unity — i.e. the prototype need not determine the effects of multipath propagation.

If the ray tracing algorithm were to be re-implemented using a PE model, then the PPF calculation would become unnecessary. This is because the PE algorithm implicitly includes multipath propagation effects.

2.5 Scattering

Whenever the radar's RF energy is incident on an object, part of the energy is scattered. The most important scatterers are the facets of the target, the surface of the sea, the ground and volume clutter (e.g. rain). We will examine each of these scatterers, and the modelling thereof, in the sections that follow. It is noted that we only consider models applicable to pulsed radar.

2.5.1 Target models

When modelling radar target returns, the four parameters of interest are the return's amplitude, phase, frequency and range delay. The first two are complex functions of the target's geometry, construction materials, position, aspect angle and the radar frequency. The return frequency is determined from the doppler effect, and the range delay is calculated from the slant range.

Our analysis of target modelling has two parts. Firstly, we discuss the theoretical return parameters, including target radar cross-section (RCS). Secondly, we consider the inherent fluctuations of a target's measured RCS (target noise).

Target returns

The single most important target characteristic for detection and tracking is the target's RCS. This parameter indicates how much of the RF energy incident on the target is reflected

back to the radar. In this section we will discuss the target return parameters and their relationship to the RCS. Then we will examine methods of determining the target RCS.

Return amplitude. The amplitude of the target's return is represented by its RCS σ in the radar transmission equation. The basic transmission equation [78, p. 2-4] is

$$\frac{P_r}{P_t} = \frac{G_t G_r \sigma \lambda^2 F_t^2 F_r^2}{(4\pi)^3 R_t^2 R_r^2 L_s}, \quad (2.18)$$

with	P_r	= received power, [W]
	P_t	= transmitted power, [W]
	G_t	= transmitting antenna gain, []
	G_r	= receiving antenna gain, []
	σ	= radar target cross-section, [m^2]
	λ	= RF wavelength, [m]
	F_t	= pattern-propagation factor for transmitting-antenna-to-target path, []
	F_r	= pattern-propagation factor for target-to-receiving-antenna path, []
	R_t	= target to transmitter range, [m]
	R_r	= target to receiver range, [m]
	L_s	= system losses, []

The pattern-propagation factors include the antenna pattern and propagation effects (a discussion is given in Section 2.4.1, p. 19). Equation (2.18) is for a bistatic radar. In the monostatic case, $G_t G_r$ can be replaced by G^2 , $F_t^2 F_r^2$ can be replaced by F^4 , and $R_t^2 R_r^2$ by R^4 . For a simulator, the system losses L_s would include any attenuation due to gaseous absorption or weather effects, as well as losses in the front-end of the radar receiver, up to the point of injection.

Return phase and delay. When generating returns for a coherent radar, it is critical that the relative phase of the returns are maintained pulse to pulse. For a scatterer at a range R [m], the phase change of the return $\Delta\phi$ for a direct ray path is [57, p. 142]

$$\Delta\phi = \frac{4\pi R}{\lambda} \quad [\text{rad}] \quad (2.19)$$

with λ the RF wavelength [m]. The time rate of change of $\Delta\phi$ is proportional to the doppler frequency shift.

If the ray path includes reflection off the ground or sea, then there will be an additional phase shift. This effect was discussed in Section 2.4.

The range delay τ_d can also be calculated in a very straight forward manner:

$$\tau_d = \frac{2R}{c} \quad [\text{s}] \quad (2.20)$$

with c the speed of light [m/s]. If retardation (see Section 2.3.2) is to be accounted for, then the retardation range error ΔR must be added to the true slant range R .

Return frequency. The frequency of the target return is shifted from the transmission frequency according to the doppler effect. The doppler frequency shift is given [57, p. 143], [50, p. 20] as

$$f_D = \frac{2v_r f_0}{c} = \frac{2v_r}{\lambda} \quad (2.21)$$

with f_D = doppler frequency shift, [Hz]
 f_0 = RF transmission frequency, [Hz]
 v_r = target's relative radial velocity towards radar, [m/s]
 c = free space speed of light, [m/s]
 λ = RF transmission wavelength, [m]

It is noted that eq. (2.21) is an approximation that holds only if $v_r \ll c$ — this is almost always true [50, p. 20]. If the target is modelled by a number of point scatterers, v_r and f_D should be calculated for each one independently. A lower fidelity model can be obtained by using the velocity of the target's centroid for all the point scatterers — errors will only be present when the target's flight path is curved. For practical implementation, Leighty and Perkins [53] note that problems can occur if the simulator does not track the transmit frequency closely.

Far-field. In most field radar applications, the target is generally distant and can be assumed to be in the antenna's far-field. If the path length from a point to all parts of the antenna are in phase, or nearly so, then the point is in the far-field [51, p. 329].

At 10GHz, for a 2m diameter antenna, the far field distance is approximately 167m (see p. 154 in Appendix B). Thus the far-field approximation is very reasonable for our application, for all but the closest targets.

Polarisation effects. The polarisation of the transmit and receive antennas affect the amplitude of the return. With linear polarisation, the majority of the RF energy returned from a complex target will be polarised in the same sense as the illuminating energy. With circular polarisation, the return energy from a complex target is split roughly evenly between left and right circular polarisation. Nearly isotropic scatterers such as rain drops return most of the circularly polarised energy with the sense reversed — this phenomenon can be used to reduce rain clutter [78, p. 28-17].

Polarisation effects need only be considered when polarisation-dependent simulation models are available. An example of statistical polarimetric modelling is given by Sandhu [71]. Typically, this level of model fidelity is only required for radar and EW evaluation [53]. The effects of polarisation are sometimes considered as part of target noise — this is known as polarisation modulation [78, p. 28-2].

Simple RCS models. Very simple models for determining the mean RCS of a target exist. In general, these models empirically relate the cross-section to the target's physical

extent. Examples for aircraft and ships are given in Appendix B.

Real-world RCS measurements. A common method of determining the RCS of a complex real-world target is measurement — either of the actual target or a scale model [47]. The majority of existing measurements have been performed on stationary targets using synthetic aperture radar [63].

Direct measurement was the only option before digital computers capable of synthetic RCS measurements were widely available. While real-world measurements may provide good results for the particular radar system used for the measurements, they cannot easily be translated to other systems. Another problem is that the measured datasets are usually incomplete as it is difficult to measure data at all aspect angles, and over a wide band of frequencies.

Synthetic RCS measurements. Simulated full-wave EM analysis on a computer-aided design (CAD) model is currently the most popular method of RCS determination [11, 25, 63, 74, 76]. It is sometimes referred to as EM scattering prediction in the literature. The CAD simulations use techniques such as physical optics and the uniform theory of diffraction [48, 97]; or the generalised forward-backward method [66].

Using the CAD approach has a number of advantages over actual measurements. Firstly, it is easier to obtain a complete set of data for all aspect angles. Secondly, it is far simpler logistically and more cost effective. Thirdly, high resolution models can be used at a variety of frequencies and are thus applicable to a wider range of radar systems [76].

Model optimisation. None of the techniques mentioned above are feasible for a real-time simulator as a massive amount of measurement or processing time is required. The amount of data generated from these measurements or simulations is enormous, and so cannot be used directly in a real-time simulator. The amount of data must be radically reduced by some means. A first-order approach is a single isotropic point scatterer. A better method is the use of multiple isotropic scatterers — a valid approach according to RCS theory [47]. The point scatterers are generally located near features such as the nose and wing tips. Multiple point scatterers model the RCS aspect angle dependence well.

At long ranges, where the individual scattering elements of the target cannot be resolved, a single point target may suffice. This is the lowest fidelity model, but even so, it is applicable to radar testing and evaluation level simulation — as long as the target is much smaller than the radar resolution cell [53]. The mean RCS of such a point target should still vary according to the target's aspect angle. When a target spans multiple radar resolution cells, it must be modelled by a number of individual point scatterers.

In order to calculate the positions of the multiple point scatterers, a high resolution radar return can be analysed along with the model that generated it. Zwart *et al.* [97] propose an estimation-maximisation approach, which includes occlusion and interference effects, to find the scatterers. Using the modelled scatterers, they are capable of accurately predicting the

locations of peaks in the radar return for arbitrary aircraft poses. Charrier and Delilse [18] suggest using wavelet analysis on the RCS data — they report relatively good success with simple targets. Details of real-world targets are still to be reported. Simpson and Galloway [76] investigate super-resolution inverse synthetic aperture radar. Their simulated results are very promising.

If the number of point scatterers determined is relatively high, reduction can be achieved by various approaches, according to Hughes and Leyland [43]. An exhaustive approach always results in an optimal solution, but requires extensive processing (2^{20} iterations for 20 scatterers). An iterative method only requires 209 iterations (for 20 scatterers), but the solution is far from optimal. Using a genetic algorithm, Hughes and Leyland show that a near optimal solution can be obtained with far less processing than is needed for an exhaustive search (50 000 iterations for 20 scatterers).

Neither the single nor multiple point target approaches are ideal, but they provide a medium level of model fidelity. A more accurate approach known as cluster modelling has been developed [11]. This technique is a method of compressing the raw radar image, by selecting clusters of pixels — generally located at the same position as where the isotropic scatterers would be. The additional information from a group of pixels improves model accuracy. Experiments show the clustering approach to work very well; however, the added fidelity is at the expense of more processing time.

Another way of reducing the amount of data is to average the RCS over sectors of aspect angle. The simulator's memory resources and the radar application determine the sector resolution.

Target noise

Amplitude scintillation. The RCS determination discussed above is an average RCS — real radar returns will fluctuate around the average by 30dB and more [78, p. 27-2]. This is known as amplitude scintillation. The amount of received energy is equal to the vector sum of the returns from each of the target's reflective surfaces. Small changes in the relative positions of these scatterers cause small phase changes in the received echo from each one. The resulting interference can cause a large change in the resultant vector's magnitude. This effect is more pronounced at shorter wavelengths.

The positions of the reflective surfaces relative to the receiving antenna can change due to a number of effects [78, p. 28-2]:

- Airframe vibration.
- Random yaw, pitch and roll motion (even when flying on a “fixed” course).
- Weather conditions.
- Polarisation modulation.

The classic method for modelling amplitude scintillation is by generating the RCS with a statistical process. We will discuss the various probability density functions (PDFs) and their applicability to certain real-world targets below. An important consideration with this type of model is the target aspect angle. The average RCS is dependent on the target's pose. E.g. the broadside returns from an aircraft will be larger on average than the head-on returns [78, pp. 27-9–27-12].

When modelling the amplitude scintillation of aircraft and missiles, the earliest and most common models are those by Marcum and Swerling. The original work was done by Marcum on non-fluctuating targets in order to determine the probability of detection for a pulsed radar. Swerling [80] extended the work to fluctuating targets and defined four cases which depend on the target and radar configuration. The four cases depend on two different PDFs.

The first PDF characterises the RCS via a negative exponential distribution (although it applies to a Rayleigh target because the received voltage is Rayleigh distributed) [78, p. 2-18]

$$\text{Swerling I and II, exponential PDF: } p(\sigma) = \frac{1}{\bar{\sigma}} e^{-\sigma/\bar{\sigma}} \quad (2.22)$$

with $\bar{\sigma}$ the mean RCS. This distribution is suitable for targets with more than about five independent, equally sized scattering centres. Most aircraft fit this profile at microwave frequencies [57, p. 161], [78, p. 2-19].

The second distribution is more suited to targets where there is a dominant scatterer and many smaller independent scatterers. Practical examples include missiles [57, p. 161]. In this case, the PDF of the radar cross-section is Chi-squared with four degrees of freedom

$$\text{Swerling III and IV, } \chi^2_4 \text{ PDF: } p(\sigma) = \frac{4\sigma}{\bar{\sigma}^2} e^{-2\sigma/\bar{\sigma}} \quad (2.23)$$

with $\bar{\sigma}$ the mean RCS, as before.

The four Swerling cases and the fifth (Marcum) case are summarised in Table 2.2. For a search radars, slow fluctuations mean that the target's RCS is constant during a scan, but fluctuates from scan-to-scan. For pulse doppler tracking radars, slow fluctuations would occur between patterns. A fast fluctuating target's RCS changes from pulse to pulse.

Table 2.2: *The Swerling cases for target RCS fluctuation.*

Case	Fluctuation rate	PDF
I	slow	Eq. (2.22)
II	fast	Eq. (2.22)
III	slow	Eq. (2.23)
IV	fast	Eq. (2.23)
V	non-fluctuating	

For high pulse repetition frequency (PRF) radars, at gigahertz frequencies, most complex targets would be classified as fast fluctuating. This is especially true if the radar uses pulse-

to-pulse frequency diversity. Changing frequencies changes the relative phase differences between the various scatterers, thus resulting in RCS fluctuation.

Weinstock showed that the Swerling models did not cover all types of targets — satellites in particular. Swerling's response [81] generalises the five models above and the Weinstock models to χ^2 PDFs, but is mostly focused on the probability of detection curves. The Weinstock model uses a modified Gamma distribution [57, p. 165]

$$\text{Modified Gamma PDF: } p(\sigma) = \frac{\eta^\eta}{\Gamma(\eta)} \sigma^{\eta-1} e^{-\eta\sigma} \quad (2.24)$$

where η is the shape parameter (it has been set equal to the scale parameter in the standard Gamma distribution). For the Weinstock model, η must be between 0,6 and 4.

The RCS for ships can be modelled using a log-normal distribution [57, p. 161]. The log-normal distribution is given as

$$\text{Log-normal PDF: } p(\sigma) = \frac{1}{\sigma \sigma_s \sqrt{2\pi}} e^{-\frac{1}{2} \left[\frac{\ln(\sigma/\sigma_m)}{\sigma_s} \right]^2} \quad (2.25)$$

with σ_s = standard deviation of $\ln \sigma$

σ_m = median

$e^{\sigma_s^2/2}$ = mean-to-median ratio.

When the mean-to-median ratio is small, the distribution is characteristic of one large scatterer and a number of smaller ones. Large mean-to-median ratios are used when the dominant scattering is due to a few strong reflectors. Most ships would have a number of strong reflecting surfaces illuminated by the radar and should be modelled with a large mean-to-median ratio.

A problem with trying to characterise measured RCS data with analytical PDFs, such as those above, is the limited number of parameters available. Xu and Huang [96] propose a new method of non-parametric statistical modelling whereby Legendre orthogonal polynomials are used to reconstruct the first n moments of the PDF of the target's RCS. The polynomial representation matches the true PDF and cumulative distribution function (CDF) very well. The reason why Xu and Huang approximate the measured CDF by polynomials is unclear. One possibility is that it may not be feasible to store the full CDF for all aspect angles. Using the polynomials, the PDF or CDF can be calculated from a few polynomial coefficients. Typically, between 15 and 20 coefficients are required.

Doppler scintillation. There are two major phenomena which lead to this effect [78, p. 28-17]

- Continuous doppler spread due to the motion of the aircraft.
- Doppler spectral lines due to rotating parts such as jet turbine blades, propellers and rotors.

The continuous doppler spread has the same causes as amplitude scintillation, and is also modelled by a statistical process. From [78, p. 28-20], the probability density distribution of the doppler frequency for aircraft is

$$\text{Hankel PDF: } p(f) = \frac{1}{2\pi\sigma_\psi\sigma_\omega} K_0\left(\frac{f}{2\sigma_\psi\sigma_\omega}\right) \quad (2.26)$$

with K_0 = modified Hankel function

f = frequency, [Hz]

σ_ψ = standard deviation of angle scintillation, []

σ_ω = standard deviation of yaw rate, [Hz].

Typical values are given in Appendix B.

Rotating parts frequency modulate the doppler spectrum, resulting in spectral lines displaced from the airframe's average doppler frequency [78, p. 28-20]. When the target is viewed head-on, the spectral lines will be symmetric around the airframe's doppler frequency. For arbitrary aspect angles the spectral lines can be located anywhere in the doppler spectrum. The amplitude and frequency spacing of these lines is dependent on the engine's type and speed [53].

Modelling the modulation effects is very important if the radar has either jet engine modulation rejection or recognition capabilities.

Glint. When measuring the position and doppler frequency of a complex target, there will be errors due to the interference between the various scatterers' returns. These errors are known as glint. The same effects that cause scintillation are responsible for variations in glint.

Positional glint is manifested as range and angle errors in the radar. The location of scatterers can sometimes appear outside of the physical extent of the target. Leighty and Perkins [53] suggest modelling positional glint in a very straightforward manner:

- The target's apparent position varies normally around its true position.
- The apparent position must lie inside the target's extremities 80 to 90 percent of the time.

According to Chen *et al.* [19], other modelling approaches include the *t*-distribution, generalised hypergeometric functions and even an empirical approach. Chen *et al.* propose a new technique using fractionally differenced Gaussian noise (FDGN). The discrete FDGN process is given as

$$w_d[n] = \sum_{k=0}^{\infty} \frac{(k+d-1)!}{k!(d-1)!} w[n-k] \quad (2.27)$$

where d is a parameter dependent on the target type, and $w[n]$ is a discrete zero-mean white Gaussian process with standard deviation σ_w . Typical values for d are between 0,3 and 0,5.

The FDGN process matches the long term correlation properties of measured glint well, and the spectral density of the two are similar. Chen *et al.* also compare a time sequence of measured and simulated glint, which do not appear to be that similar. Nevertheless, a discrete FDGN process is a simple and effective way of simulating glint.

Glint of the doppler frequency can be generated through similar statistical techniques, however we will model it as doppler scintillation.

Model selection

The return power, phase, range delay and doppler shift of the target return will be determined using the equations discussed in Section 2.5.1. For each type of target that can occur in the scenario, the software must provide a software object that implements the target’s scattering model. To increase the system’s extensibility, the software framework must allow new types of objects to be added.

The baseline system will only include a software object for aircraft. To keep the prototype simple, aircraft will be modelled as non-polarimetric, isotropic scatterers with a mean RCS value entered by the user. The actual target RCS for a certain pulse will be taken from a PDF that describes the amplitude scintillation. The classic Swerling PDFs are selected as they characterise aircraft returns well. All five cases must be implemented. Doppler scintillation is not implemented, but the necessary methods can be added to the target’s object at a later stage, if desired. Each target object should have a method that returns its position — glint is not implemented, but could be added to this method, if necessary.

2.5.2 Sea clutter

Sea clutter is the term used to describe the backscattered RF energy from the sea’s surface. The sea clutter can obscure returns from small vessels on or near the sea’s surface, thus degrading the performance of a radar system. For this reason, sea clutter simulation is important for all marine and airborne radar systems.

We will first consider a phenomenological description of sea clutter. Thereafter we review constant, then deterministic, and finally probabilistic sea clutter models.

Phenomenological description

Sea clutter consists of two major components [61, p. 9]. The first, termed speckle, is due to the small (< 2.5 cm high), fast changing capillary waves. The velocity of propagation of the capillary waves is primarily determined by the water’s surface tension. Capillary waves are generated by local gusts of wind and disappear soon after the wind dies down [78, p. 26-4].

The second component is due to the larger waves known as swell. These waves are also generated by wind, but because of their larger height (> 5 cm), their propagation velocity is controlled by gravity. The gravity dependence allows these waves to propagate very long distances without the help of the wind [78, p. 26-4].

Sea clutter is mostly due to the reflections from the capillary waves, especially at frequencies above 8 GHz [78, p. 26-4], [61, p. 9]. The reason is that the majority of the scattered energy is due to water waves with a wavelength equal to half the RF wavelength. Reflections from these waves add coherently, thus giving a larger return. This process of selective reflection is known as Bragg scattering [50, p. 228]. At microwave frequencies, the resonant water wavelength is on the order of centimetres — the same size as the capillary waves. At longer RF wavelengths, e.g. HF, the dominant reflections are from the swell.

The capillary waves ride on the swell, thus the sea clutter power is modulated by the swell. According to Ward *et al.* [92], the speckle decorrelates on the order of a few milliseconds, while the modulating component requires a few seconds to decorrelate. With pulse-to-pulse frequency diversity, the speckle component can be made to decorrelate from pulse to pulse. The modulating component is unaffected by frequency agility. The independence of the two effects suggests the use of compound models.

Recently, Lamont-Smith and Walker proposed a 3-component model [52]. The first component of this model is in fact the compound effect of the modulating and speckle components discussed above. The second component is due to the scattering from the very rough white-caps of broken waves. The third component is associated with the specular scattering from a wave crest just before it spills. According to Lamont-Smith and Walker, the latter two components are largely responsible for the spikes observed in sea clutter.

Returns measured with vertical-vertical (VV) polarisation differ slightly from those measured with horizontal-horizontal (HH) polarisation. The reason is that different scatterers are responsible for the returns in the two polarisation modes [61, p. 10]. Polarimetric effects are generally accounted for by adjusting the model parameters [64, p. 110] [52, 92]. Some authors claim that the returns require different models [33].

The doppler spectrum of both VV and HH sea clutter is very similar. It is largely determined by the modulating component. The reason for this is that the instantaneous velocity of the wave front changes periodically with the rise and fall of the swell — this modulates both the doppler spectrum and the intensity of the returns [61, p. 11].

Sea clutter is correlated both temporally and spatially — mostly due to the continuous nature of the swell. Swells generally cover more than one range cell and thus adjacent cells cannot be independent. Temporal correlations are easily explained by the fact that the waves do not change shape instantaneously. The correlation is especially evident when using high resolution radar. Environmental conditions, look angle and radar parameters strongly affect the measured correlation structure [92].

Simple probability distributions cannot adequately model both the temporal and spatial correlation. The former is simple to model, but the spatial correlation much less so.

Scattering coefficients

The scattering cross section is proportional to the area illuminated by the radar beam, i.e. the radar resolution cell's area. As the area of a radar resolution cell is dependant on the

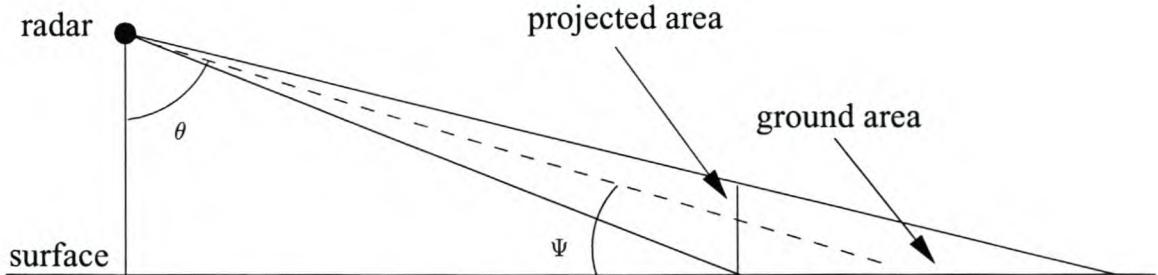


Figure 2.10: Geometry for the definition of the scattering coefficients.

range, it is preferable to characterise clutter using a scattering coefficient defined as the scattering cross section per unit area. There are two differing definitions, depending on which area is used — the geometry, with the radar some height above the surface, is shown in Figure 2.10. When dividing by the ground area, the scattering coefficient is denoted as σ^0 , and when dividing by the projected area, the symbol γ is used. The relationship between the two is derived from simple trigonometry as

$$\sigma^0 = \gamma \sin \Psi \quad (2.28)$$

where Ψ is the grazing angle. The area of a radar resolution cell is also simple to calculate

$$A_{res} = R \Delta\theta c \frac{\tau}{2} \text{ [m}^2\text{]} \quad (2.29)$$

with R = range, [m]

$\Delta\theta$ = azimuth beam width, [rad]

c = speed of light, [m/s]

τ = pulse width, [s].

Some references measure the scattering coefficient in $[\text{m}^2/\text{m}^2]$, while others use [dB]. The conversion is $\sigma_{dB}^0 = 10 \log \sigma^0$. The differentiation will be clear.

Constant models

If only the average sea clutter power is required, then a constant model can be used. Such models are applicable to first order radar coverage calculations. For radar environment simulation, the clutter should invariably fluctuate. In this case, the average power can be used as the mean of a probability distribution — probabilistic models will be discussed on p. 36.

Appendix B (p. 165) reviews two empirically determined, constant models. The first model, given by Barton and presented in [57, p. 195], is very simple, considering only the wind speed and grazing angle . The second, known as the Georgia Institute of Technology (GIT) model, is more sophisticated. It includes effects due to polarisation, surface roughness, interference, and wind direction. A modified version of the GIT model is used in the EREPS program [64, p. 109].

Deterministic models

We consider two broad fields of deterministic models. The first is derived from chaos theory, and the second from the theory of fractals.

Chaotic modelling. With enough information about the sea surface and the illuminating radar beam, it is theoretically possible to predict the exact radar return. However, the backscattering process is extremely sensitive to the angle of incidence. This sensitivity suggests the use of chaotic modelling for the scattering process [38],[61, p. 14].

Haykin and Puthusserypady [38] point out that there is no mathematical or physical reason for modelling sea clutter as a stochastic process. It is purely the apparent randomness of the waveform that has led to probabilistic modelling. They report that there is strong experimental evidence to support the claim that sea clutter is indeed chaotic.

Unsworth *et al.* [87] studied the nature of sea clutter and found it to be statistical, rather than chaotic. Noga [61, p. 19] also doubts the chaotic nature of sea clutter, which he says has not been adequately established. He argues that the tests used by some authors to diagnose chaos, can be misleading when applied to sea clutter. Noga reaches no conclusion as to whether or not sea clutter is a chaotic process. He also accedes that the results obtained using dynamical models can indeed increase the performance of radars. Thus, the chaotic models must have some validity.

In order to model a chaotic system, a non-linear predictive model is required. A neural network is an example of such a model. A one-dimensional multilayer perceptron has been shown to perform well as both a clutter simulator and suppressor [38]. Other techniques include memory-based methods, adaptive rational function filters, and radial basis function neural networks (RBFs) [95].

Xie *et al.* [95] have recently used the RBF network, which is both accurate and robust, in a multiple model approach. The multiple model RBFs characterise the different effects such Bragg reflection and sea spikes. They propose an expectation-maximisation algorithm to train the models. Their results show that the multiple model RBFs are capable of modelling the different dynamics of sea clutter successfully. Their multiple model consistently outperforms a single model. Multiple model RBFs can be applied in real-time — assuming the training has already been performed. This makes the technique suitable for both radar signal processing and simulation purposes.

Fractal modelling. An alternative to chaos theory, is the closely linked field of fractals. Using fractal functions, natural surfaces can be modelled. Fractal-type modelling falls between purely periodic and purely random surface models. The natural roughness of the sea can be well represented by a fractal as it has a number of adjustable parameters.

Berizzi and Dalle Mese [10] propose a sea surface model based on bandlimited multiscaled fractal functions from the Weierstrass-Mandelbrot class. Their model uses one-dimensional roughness and includes dynamical evolution of the surface.

The scattering coefficient γ is the ratio of the actual scattered electrical field to that scattered, in the specular direction, from a smooth perfectly conducting surface. The return signal is then proportional to γ . Berizzi and Dalle Mese give a closed form solution for γ . The solution is in the form of an n -dimensional discrete Fourier transform, thus making calculation a relatively processor intensive application. The parameter n is the number of periodic functions (or tones) summed to create the sea surface. They show that for a simplified sea surface model, the fractal dimension of the scattering coefficient is the same as that of the sea surface.

According to Berizzi and Dalle Mese, other work has shown agreement between fractal models and experimental data.

Probabilistic models

Whenever the underlying principles of a process are not well understood, probabilistic models are a good option. While the physical mechanisms of sea clutter are now relatively well understood [52], full wave EM analysis is still not feasible for a real-time simulator. The simplicity with which probabilistic models can be applied, and the reasonable accuracy that can be obtained, justify their consideration.

Simple processes. When using low resolution radars, incapable of resolving the structure of the surface, sea clutter has a quadrature Gaussian characteristic [92]. According to Noga [61, p. 22], this is because the large number of scattering centres illuminated allow the application of the Central Limit Theorem. The quadrature Gaussian scattering results in the sea clutter amplitude being Rayleigh distributed. The clutter cross-section σ_c (i.e. the return power) will thus be negatively exponentially distributed

$$\text{Negative exponential PDF: } p(\sigma_c) = \frac{1}{\bar{\sigma}_c} e^{-\sigma_c/\bar{\sigma}_c} \quad (2.30)$$

with $\bar{\sigma}_c$ the average value of the clutter cross section [78, p. 26-19].

With high resolution radars and low grazing angles, the Rayleigh distribution does not have a heavy enough tail to match measured data [61, p. 22]. The log-normal distribution — equation (2.25) — is an example of a heavier tailed PDF. Another is the Weibull distribution, given as

$$\text{Weibull PDF: } p(r) = abr^{b-1} e^{-ar^b} \quad (2.31)$$

with r the amplitude, a a scale parameter, and b a shape parameter. The Weibull distribution does match some experimental data [37]. Farina *et al.* [33] report that the log-normal model is good for HH polarised radars.

Nohara and Haykin [62] have considered using autoregressive (AR) processes to model high resolution sea clutter, specifically for the detection of small icebergs (known as growlers). AR processes are shown to characterise sea clutter well over short time periods (one to two seconds). However, they do not discuss their model's long term evolution.

Compound processes. When considering single range cells, Ward *et al.* [92] have shown the clutter is locally Rayleigh distributed. In other words, the speckle component is Rayleigh distributed. They conclude that the overall non-Rayleigh nature of the total sea clutter amplitude must be due to the bunching of the scatterers caused by the larger waves, rather than there being a small number of effective scatterers in a range cell. The resulting model must be one of a modulated Gaussian process.

Ward *et al.* show that the modulation should be Gamma distributed. According to Noga, this can be explained by a birth-death migration process model for the evolution of the sea surface scatterers. However, Noga shows experimentally [61, pp. 48–52] that using a log-normal distribution for the modulating component provides more accurate results for target detection analysis. He concludes that models other than the Gamma distribution should be sought for the modulating component.

In any event, an extensive amount of study has been done using the Gamma distribution as the modulating component [12, 13, 26, 27, 45, 86, 92, 94]. The resulting compound model is the K-distribution. It is given by Ward *et al.* [92] as

$$\text{K-distribution} \quad p(r) = \frac{2\beta}{\Gamma(\alpha)} \left(\frac{\beta r}{2} \right)^\alpha K_{\alpha-1}(\beta r) \quad (2.32)$$

with r the clutter amplitude, β the scale parameter, α the shape parameter, $\Gamma(\cdot)$ the gamma function, and $K_\alpha(\cdot)$ the modified Bessel function. Ward *et al.* state that the K-distribution is a considerably better model than the simple Gaussian model. The K-distribution's separation of the speckle and the modulating component has good physical grounds, thus it generally performs better than the Weibull and log-normal distributions [61, p. 24]. The compound approach also allows the differing correlation properties of the speckle and swell to be characterised. Tough and Ward [86] provide more detail on the origins of the K-distribution.

Noga [61, p. 140] proposes a conditionally Gaussian state-space model for high resolution sea clutter. Like the K-distribution, he models the speckle as a Rayleigh distributed process. However, the modulating component is based on a conditional heteroscedastic (non-constant prediction error variance) regression model. His own work shows that there is no compelling evidence for the heteroscedasticity of the modulating component.

Shnidman's general clutter model [75] is based on a non-central chi-squared gamma (NG) distribution. He models the modulating component with a gamma distribution. The speckle is no longer assumed Gaussian, and instead modelled with an NG distribution. This speckle density is most suited to returns dominated by a single scatterer. Shnidman shows that the NG density fits an example of sea clutter measurements (VV and HH) well. The NG distribution has three parameters: the number of clutter sources, the average of the total normalised power of the clutter sources, and a fluctuation parameter. Values for these parameters can be determined from measured data. The NG distribution is basically an extension of the K-distribution, allowing for more complex speckle. In the case of sea clutter this is unnecessary due to the, widely agreed upon, Gaussian nature of speckle.

In Section 2.5.3 we discuss ground clutter, which is a more likely candidate for the NG distribution.

Single point distribution parameter estimation. Before the PDFs discussed previously can be applied, the single point parameters must be estimated. Noga [61, pp. 24–26] provides a good summary of the available estimators. In Appendix B we discuss these estimators and also include an empirical model due to Ward *et al.* for the K-distribution’s shape parameter [92]. As mentioned previously, the constant sea clutter models (see p. 34) provide a relatively simple method of determining $\bar{\sigma}_c$, based on the scenario conditions.

Simulating sea clutter. Generating data with specific single point statistics is a relatively straight forward process. Unfortunately, when generating sea clutter, the problem of matching the correlation and spectral properties have to be considered as well. The correlation properties become important when measuring or calculating radar detection performance, especially with constant false alarm rate (CFAR) modules. When generating non-coherent data, the spectrum need not be considered.

If using a simple probability distribution (e.g. Rayleigh), the clutter spectrum can be modelled as Gaussian [57, p. 196]. It should be centred on the mean radial speed of the waves, which is between about an eighth and a quarter of the wind speed. The standard deviation is taken as one eighth of the wind speed. However, Ward *et al.* [92] show that the sea clutter spectrum is not truly Gaussian, and that its shape is time variant.

The single statistic nature of log-normal and Weibull distributions make them unable to capture both spatial and temporal correlations. Methods exist to generate correlated data from both of these distributions, either coherently or non-coherently. Noga [61, pp. 26–27] provides a summary of these methods.

Using the K-distribution, spatially and temporally correlated data can be generated with some difficulty. K-distributed data can be obtained by multiplying Rayleigh distributed data with Gamma distributed data. The Rayleigh data is trivial to generate, while the Gamma data, because of the required correlation and spectral characteristics, is not.

Spatial correlation is required for the modulating component, because this component is due to the swell which generally extends over multiple range cells. The fast decorrelation of the speckle makes it essentially uncorrelated spatially. According to Denny [26], the Gamma distributed data must be filtered, prior to multiplication with the speckle, in order to simulate spatial correlation. The filtering can be performed in two dimensions, if necessary. The filter coefficients must be chosen carefully in order for the data to be correctly correlated. Denny also points out that the filtered data’s shape factor will be scaled by the filtering process.

Temporal correlation, like the spatial correlation, is more critical for the modulating component because of the speckle’s short decorrelation time. With frequency agile radar, the speckle decorrelates pulse to pulse. Measured data [26] shows that the autocorrelation function of sea clutter decreases from 1 to about 0,3 in one pulse, but then decreases much

more slowly to zero. This is indicative of the differing timescales of the modulating and speckle components.

Denny [26] reviews three methods of generating the Gamma data. His simulation methods are only applicable for incoherent K-distributed data, as the spectrum is not considered. The first method only generates uncorrelated data and is thus of minimal use. The second method has the correct spatial correlation property (exponential), but Denny is unsure as to whether or not the temporal correlation is correct (MATLAB code is given in [24]). The third method [27], while producing temporally correlated data correctly, has a triangular rather than exponential spatial autocorrelation function.

Blacknell [13] suggests another method of simulating random numbers belonging to a Gamma distribution. He proposes that a number of independent Gamma distributed variables are passed through a variety of moving average filters and then summed to obtain the necessary correlation properties. This technique allows a much wider range of autocorrelation functions to be generated than previous methods [61, pp. 27], e.g. using bivariate Gamma distributions, or spherically invariant random processes (SIRPs) [21]. The difficulty with Blacknell's technique is finding the parameters for the filters and the Gamma distributions. The complexity increases almost exponentially with the highest non-zero lag of the autocorrelation function (ACF). Blacknell only presents an explicit solution for the case where there are two non-zero lags in the ACF. He suggests using a computer aided search when attempting to match longer ACFs. Once the Gamma distributed samples have been generated, he proposes that each one be replaced with a sample from an exponential distribution with the same average as the original sample. In this way, the speckle is added to produce K-distributed data.

Tough and Ward [86] discuss a technique whereby two-dimensional coherent sea clutter, from a K-distribution, can be simulated with the correct correlation properties. It is in fact a generalisation of the second method examined by Denny above. The key to the technique is a non-linear transformation of Gaussian distributed random variables into Gamma distributed random variables, with the required correlation properties. Tough and Ward use Hermite polynomials to form a power series representation of the relationship between the correlation functions of the input Gaussian and output Gamma processes. By inverting this relationship, the required correlation function of the input Gaussian process can be found. Arbitrary correlation of the Gaussian process can be obtained by suitable filtering [46].

Model selection

Sea clutter returns are never constant, so a purely constant model would be a very poor representation of the effect, rather a deterministic, or probabilistic model should be used. The deterministic models are invariably complicated to implement, so a probabilistic model is preferred. When using such a statistical model, the single point distribution parameters are required — this is a use for the constant models. Our approach is thus to determine the mean clutter cross-section using a constant model — Barton's simple model (see p. 165), is

preferred to the more complex GIT model (p. 166). A large body of literature indicates that the K-distribution is the best PDF for the fluctuations. It is straightforward to calculate the shape parameter using the model due to Ward *et al.* (see p. 168). Controlling the clutter’s correlation properties is complicated and thus not implemented in the prototype system.

2.5.3 Ground clutter

The unwanted echoes from the earth’s surface and objects on it are collectively known as ground clutter. As with any form of clutter, the performance of the affected radar system is degraded. If sub-clutter target visibility is to be tested via a simulator, then the generation of clutter is an important aspect.

This section will proceed similarly to the previous one on sea clutter. First the physical basis of the clutter returns are discussed, thereafter we review some of the available models for ground clutter simulation.

Phenomenological description

Ground clutter is caused by the scattering of the incident RF energy by a wide variety of scattering elements, dependent of the terrain. Possible scattering elements include soil, vegetation and man-made structures. Different types of terrain, e.g. desert, forest, urban, etc. can be defined, each with their own mix of scattering elements.

Some authors [45] differentiate between three categories of scatterers — these are listed below with the strength of the returns in parenthesis.

- Noise-like returns from smooth flat regions, or those in the radio shadow (weak).
- Returns from the terrain with a rough surface or extensive vegetation cover (intermediate).
- Discrete clutter from elevated sources such as trees and man-made structures (strong).

More recently, only the last two types of scattering have been considered by authors [20, 73] — they term these weak diffusive backscattering, and strong specular backscattering, respectively.

The exact amount of energy in the radar return is a function of a number of parameters. The radar system’s wavelength, power, polarisation, direction and area of illumination are important. These are combined with ground parameters such as the conductivity and permittivity, surface roughness, and depth of penetration.

As with sea clutter, ground clutter can be analysed spatially and temporally [73]. The different types of terrain, and corresponding scattering elements, over a large area lead to a spatial variation of the observed ground clutter. This can be likened to the modulating component of sea clutter seen in Section 2.5.2. Unlike sea clutter, this modulating component does not vary on a timescale significant to radar measurement or simulation. Short-term

temporal fluctuations generally need only be considered for the scattering elements in each range cell. Physical phenomena such as the waving of the trees in the wind can lead to these fluctuations.

The spectra resulting from these temporal fluctuations are often taken to be Gaussian. They can be modelled very simply as having a standard deviation of 0,3 m/s [57, p. 193]. The mean of the clutter can be determined from the velocity of the radar platform. Skolnik [78, p. 25-14] gives a simplified model for the width of the doppler spectrum Δf_D , for a moving, high PRF radar at low grazing angles, as

$$\Delta f_D \approx \frac{2v}{\lambda} \Delta\theta \cos \theta_d \quad (2.33)$$

with v = forward speed of the radar platform, [m/s]

λ = RF wavelength, [m]

$\Delta\theta$ = elevation beamwidth, [rad]

θ_d = depression angle, [rad].

The spectral spread due to radar platform movement should be added to that due to the movement of the individual scatterers (e.g. leaves blowing in the wind).

Constant models

A very simple way of simulating ground clutter is by assuming that the scattering coefficient γ is constant, thus the return power only varies with range and grazing angle. This type of model has been implemented by Leighty and Perkins [53]; however, few details are given. They differentiate between mainlobe clutter and sidelobe clutter. The mainlobe clutter is assumed to have a constant amplitude per pulse. Depending on the pulselength and PRF, the return may be separate pulses or a continuous wave. The return for the difference channel is the same length as the return for the sum channel, but normally requires a phase reversal halfway through the pulse (see Figure 2.2).

Sidelobe clutter is generally of longer duration than a single pulse due to reflections from different ranges. For this reason, the returns are given an exponential decay for low PRFs. At higher PRFs, the returns overlap, giving rise to a continuous wave.

EM analysis

One approach to modelling the returns from terrain is by theoretical analysis of the EM scattering properties of the surface and objects on it. In theory, a geometrical model of the entire radar environment, accurate to the nearest centimetre (for X-band), would be required. This is understandably complex, and would tax even state of the art technology in terms of measurement, processing and storage requirements.

Digital elevation maps. With the proliferation of satellites mapping the earth, there is a large amount of topographical data available. An example is the digital terrain elevation

data (DTED) published by the National Imagery and Mapping Agency. Current technology does not allow mapping down to centimetre resolution. The DTED databases have a height accuracy of 1 m and are available in varying resolutions:

- Level 0: ground resolution of 30 arc seconds — approximately 1 km (public access).
- Level 1: ground resolution of 3 arc seconds — approximately 100 m (limited access).
- Level 2: ground resolution of 1 arc second — approximately 30 m (limited access).

With a reasonable model of the terrain topography available, the clutter level can be determined much more accurately by considering the grazing angle for each DTED facet illuminated. Reilly and Lin [70] showed that by incorporating DTED in their model, relatively accurate results could be obtained for regions that were not shadowed. Davidson *et al.* [23] have also used DTED with some success. Money *et al.* [60], and later Branson *et al.* [16], used DTED as part of their coastal environment modelling — their models perform well, in both standard and anomalous propagation.

Simplified EM models. In an effort to make EM analysis feasible, simplifying assumptions must be made. One of the earliest attempts is the use the Kirchhoff-Huygens model which assumes that the current flowing in a rough or curved surface is the same as that which would flow in a flat surface tangential to the rough one [78, p. 25-9]. This model requires the spatial ACF of the surface heights — this information is not readily available. The rough surfaces can also be approximated by simple shapes such as cylinders, spheres and corner reflectors, whose EM scattering characteristics are known. The latter approach proves too complicated for all but the coarsest of models.

A much more popular approach is the use of the parabolic wave equation (PE). This approximation was already discussed in the section on refractive modelling (see p. 12). The Tropospheric Electromagnetic Parabolic Equation Routine (TEMPER) [84] includes the earth's surface as a boundary condition. TEMPER accounts for the conductivity, permittivity and roughness of the surface, the grazing angle and the incident polarisation. Reilly and Lin's work used the TEMPER model combined with a DTED database to define the earth's surface [70]. The combination of a DTED database with the PE results in a far more accurate model, and a similar approach is applied in other simulation programs [9, 60].

According to Popov *et al.* [67] the time-domain solution of the parabolic equation is only applicable in two limiting cases: narrowband signals and ultra-wideband carrierless bursts. As most radars are narrowband, this is not a significant drawback. Popov *et al.* propose an efficient frequency-domain version of the parabolic equation for use with high frequency, broadband radar. They report good agreement of their method with a measured experiment.

One problem with the PE/DTED method is that complicated terrain (e.g. urban areas) is not as well characterised as is simple terrain (e.g. desert), when just the elevation data is used. No doubt, the future will see the inclusion of much more complex surface models combined with the power of the PE method.

Probabilistic models

As with sea clutter, probabilistic modelling is useful when there are such a large number of unknowns. By empirically matching the statistical parameters to the terrain, at least a medium level of fidelity can be obtained.

Simple processes. Single distributions are capable of describing low resolution land clutter, but the models tend to have a low fidelity. The log-normal distribution, eq. (2.25), is sometimes used. A large mean-to-median ratio is descriptive of ground clutter where the mean is determined by only a few strong specular reflectors [57, p. 162]. This is most likely when there is a low grazing angle or the radar has a high resolution, because there will be fewer scatterers contributing to a cell's return [75].

The Weibull distribution, eq. (2.31), described in the section on sea clutter is more commonly used than the log-normal PDF. One of the reasons is the difficulty of mathematically manipulating the latter [57, p. 192].

There are no underlying physical reasons for using the Weibull or log-normal distributions — they are only used because they fit measured data reasonably well [75, 45].

Compound processes. As discussed in the phenomenological description, ground and sea clutter are quite similar. Thus it is not surprising that the best probabilistic models are those which model the local fluctuations and the widespread mean power variations separately [73]. Likewise, Davidson *et al.* [23] report that high resolution ground clutter is unlikely to be characterised by any distribution that assumes spatial homogeneity (at the CFAR level). Their clutter measurements appear to be locally exponentially distributed with spikes due to what they term “edge pollution”. The edge pollution is associated with the specular returns from objects such as power pylons.

The NG distribution, proposed by Shnidman [75], was already discussed in the section on sea clutter (see p. 36). Ground clutter from a cell is more likely to have a single dominant scatterer than sea clutter, thus the NG distribution's non-centralised chi-squared speckle is more suited to ground clutter. The added analytical complexity of the NG distribution, compared to the K-distribution, is a disadvantage for simulation purposes.

The K-distribution is a very good model for ground clutter. Sarno [73] shows that experimental forest clutter can be modelled spatially as Gamma distributed, thus suggesting the use of the K-distribution. The K-distribution is generally used in a composite form. This will be discussed in the paragraphs below.

Composite models. Work in this field was done by Jao [45] and more recently it has been reiterated by Choong [20]. Different terrain types are characterised by different scatterer types. Jao and Choong propose that each of these scatterers should be modelled by a random process, i.e. a composite model. The simplest example is sea clutter which can be modelled with only one scatterer type. Forest is well characterised by two scattering types —

a weak return from the leaves and a strong return from the trunks and large branches. The type of random process applied to each elementary scatterer depends on the grazing angle (for high resolution radar). For high grazing angles (e.g. for airborne radar) Rayleigh PDFs are sufficient; however, low grazing angles (e.g. surface based radar) require K-distributions.

Jao fits his composite model to various types of data, including desert, rural suburbs and water, forests and farmland. In all cases the composite models characterise the statistics of the measured clutter very well. Choong describes a composite model of farmland using Gaussian distributed soil returns and K-distributed specular returns — it too fits measured data accurately.

Jao discusses the phenomenological basis of the K-distribution for an elementary scatter type. He relates the non-Rayleigh characteristics of this PDF to a spatial birth, death, and migration process of the scatterer population. This derivation is applied to the situation where the elementary scatterers are clustered together in Poisson distributed groups. Using this clustered hypothesis, Jao obtains expressions relating the K-distribution parameters to some physical parameters, although the exact relationships have yet to be determined. Details can be found on p. 169, in Appendix B.

Parameter estimation. In order to estimate the parameters of a particular distribution, measured data is required. Skolnik summarises the results from various scattering coefficient measurement programs [78, p. 25-24]. These measurements include returns from varying terrains, e.g. urban areas, farmland, and forest, at a number of different frequencies from about 400 MHz to 35 GHz. Unfortunately the data presented is representative rather than comprehensive. No data is given for grazing angles below 10°.

When fitting measured data to the various distributions, the techniques discussed in the section on sea clutter can be applied (see p. 38). In the case of composite models, a number of PDFs must be fitted to the data — this is done by Jao [45], although he gives no details as to how he obtains the best fit.

Examples of ground clutter models and parameter estimates are given in Appendix B, p. 169.

Simulation. As the PDFs used to characterise ground clutter are the same as those discussed in the section on sea clutter, the simulation considerations remain the same (see p. 38).

Model selection

As with sea clutter, the fidelity of the ground clutter for the prototype system is not a major concern. In order to parallel the sea clutter, we choose K-distributed ground clutter, with parameters taken from Jao's work (see p. 169). However, to simplify implementation we only use a single K-distribution, rather than the compound processes suggested by Jao. The elevation of the earth will be simplified to a smooth sphere, rather than using DTED

elevation maps. However, the terrain must be defined in a grid whose size corresponds to the DTED level 1 specification (see Section 2.5.3). This will smooth future use of DTED elevation maps (DTED level 1 is chosen as maps at this resolution are available). As with the sea clutter, the ground clutter’s correlation properties are ignored for the prototype simulator.

2.5.4 Volume clutter

Volume clutter refers to those airborne phenomena that generate unwanted radar returns. As they do not lie on the earth’s surface, a three-dimensional description is required. We will consider two types of volume clutter: natural phenomena (hydrometeors) will be discussed in this section, while man-made phenomena (chaff) will be deferred to the section on electronic warfare (Section 2.6).

The most important radar parameter for volume clutter is the volume of a radar resolution cell V_{res} . A number of definitions exist for V_{res} , in general

$$V_{res} = k(R\Delta\theta)(R\Delta\phi)(c\tau/2) \quad [\text{m}^3] \quad (2.34)$$

- with k = beamshape factor
- R = range, [m]
- $\Delta\theta$ = azimuth beam width, [rad]
- $\Delta\phi$ = elevation beam width, [rad]
- τ = pulse width, [s]

The beamshape factor is nominally one, but can be $k = \frac{4}{\pi} = 1,273$, or $k = \frac{4}{\pi \cdot 2 \ln 2} = 0,918$, from [77]. The correction factor accounts for the error induced in approximating the volume boundaries by the half power beamwidths. Errors in other factors such as rainfall measurement tend to mask the correction factor, thus the nominal value of unity [78, p. 24-30].

Hydrometeors

Mean RCS. Meteorologists refer to all precipatory scattering particles (rain, hail, etc.) as hydrometeors. From [50, ch. 9], a single droplet’s RCS is

$$\sigma_{particle} = C\pi^5 D^6 / \lambda^4 \quad [\text{m}^2] \quad (2.35)$$

with C a dimensionless constant determined from the particle’s complex dielectric constant [78, p. 24-30], and D the drop’s diameter. The fact that the drops are small compared to the radar’s wavelength, means that there is Rayleigh scattering (i.e. a λ^{-4} dependence) — this holds for frequencies up to X-band. The RCS of a cloud is obtained by summing the returns from all the particles in the radar’s volume resolution cell. A model relating the cross-section to the rainfall rate is given in Appendix B.

Rain storms are generally only found in a small portion of the volume covered by a radar. Their size is inversely proportional to the rainfall rate, with a simple model [57, p. 198] giving the relationship as

$$D_r = 41,595 - 23,608 \log R_r \quad [\text{km}] , \quad (2.36)$$

where D_r is the diameter of the rain storm. According to the same reference, the CCIR model is approximately three times smaller than predicted by eq. (2.36). In practice, rainfall clutter is limited to a height of about 2 km.

Fluctuations and spectra. The total radar return from precipitation is due to the backscatter from a large number of particles. The relative position (and velocity) of the hydrometeors is random, consequently the return will fluctuate. As we have seen in the sections on sea and ground clutter, returns due to a large scatterer population can be modelled as a complex Gaussian process, i.e. a Rayleigh amplitude distribution.

Thomson and Riseborough [85] recently developed a weather clutter simulator. They generate I and Q data by filtering Gaussian noise. The exact properties of the filter depend on the parameters of their precipitation model. If the radar volume being examined only contains rain, then the spectrum can be characterised as roughly Gaussian. However, if the volume includes the 0° isotherm, then it will contain rain, snow and mixed phased particles. In this case, the spectrum becomes more exotic, e.g. asymmetry or multiple peaks. Thomson and Riseborough compare their simulator to measured data (X-band, HH polarisation) with very convincing results. The technical details of the precipitation models used are not provided in the reference.

A simpler model in [57, p. 200] shows the spectrum to be flat between the extremities of the wind speed to which the precipitation is subjected. What could be termed the rising and falling edges have a width proportional to the turbulence. As an example, assume the ground wind speed is 10 m/s (i.e. doppler shift of 600 Hz at X-band). With rain up to a height of 2 km, and windshear of 4 m/s/km, the maximum wind speed will be 18 m/s (i.e. a doppler shift of 1080 Hz). Turbulence of 1 m/s corresponds to a doppler shift of 60 Hz. The resulting spectrum is shown in Figure 2.11. The sharp edges are not required, and such a spectrum can easily be obtained with a simple bandpass filter.

Model selection

Volume clutter is not considered an important effect to be modelled by the prototype simulator. Nevertheless, the software framework must include code stubs for volume scatterers.

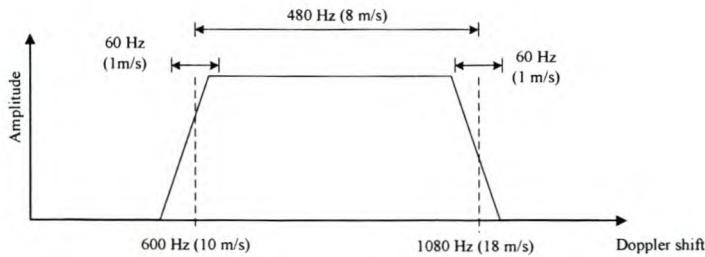


Figure 2.11: Simple precipitation clutter spectrum for rain approaching at 10 m/s, measured at X-band.

2.6 Electronic warfare

Besides all energy reflected from the various scatterers in the radar environment, EM signals can be generated directly by enemy systems for electronic countermeasure (ECM) purposes — this is known as jamming. Another source of “false” signals are the returns from chaff clouds and flares. We briefly discuss a few common ECM techniques and the modelling thereof. We are not concerned with electronic counter-countermeasures (ECCM), as these are part of the radar system for which we are simulating data.

2.6.1 Jamming

Jamming can be used to either deny detections or deceive a radar system — we will look at examples of both cases.

Noise jamming

In order to deny the use of the radar system, the enemy simply swamps the radar receiver with high-power noise. If the radar receiver bandwidth is known, spot jamming can be used to concentrate all the noise energy in the same frequency range. Problems with spot jamming occur when the radar uses frequency diversity. If a powerful enough, wideband transmitter is available, the complete radar band can be jammed — this is known as barrage jamming. In either case, modelling is very simple as the jamming signal is simply Gaussian noise filtered to a certain bandwidth.

The important principle to note is that the jamming signal only has to travel in one direction to the radar receiver. Assuming that the normal receiver noise power is much less than the jamming noise power, the power received at the radar can be calculated from [78, p. 2-62] as

$$P_r = \frac{P_j G_j G_r \lambda^2 F_r^2}{(4\pi R_j)^2 L_s} , \quad (2.37)$$

with P_r = received power, [W]
 P_j = jammer's transmitted power in the radar bandwidth, [W]
 G_j = jammer's antenna gain, []
 G_r = receiving antenna gain, []
 λ = jammer's RF wavelength, [m]
 F_r = pattern-propagation factor for jammer-to-receiving-antenna path, []
 R_j = target to jammer's transmitter range, [m]
 L_s = system losses, [].

Note that the reference gives the equation in terms of the transmitted and received spectral power density. We have accounted for this by stating that P_j is the power transmitted in the radar receiver's bandwidth.

Deception jamming

There are a number of ways to deceive a radar system and they are generally a more popular approach than noise jamming [50, p. 291]. False echoes can be generated by a repeater jammer, which simply plays back a delayed copy of the received radar signal. A slightly more advanced approach is used by the range gate stealer. At first, the range gate stealer returns a signal stronger than the target's, but at the same range. The false echo is then slowly shifted in range in the hope that the radar will follow it. The jammer is then switched off and track may be lost. This technique can be extended to the doppler domain, in which case it is known as a velocity gate stealer. Similarly, the technique can also be used to make the target appear to be at a different angular position in the radar beam — this is known as angle pull-off.

All of these effects are reasonably easy to simulate — the whole point of the simulator is in fact to deceive the radar signal processor into believing that there are really targets flying around. In light of this fact, deception jamming will not be pursued further.

2.6.2 Chaff

Dispensing chaff clouds is a useful way to clutter the radar environment and is especially effective for radars that do not use doppler processing. The proliferation of pulse-doppler radars has therefore lessened its usefulness.

Chaff clouds are composed of strips of metal foil or wire that are tuned to a specific radar frequency. As these strips fall to the ground, they generate a very large return to the radar. Their horizontal velocity is determined by the prevailing wind, and thus they can easily be ignored if the radar is tracking a fast moving target.

Chaff modelling

Simple models. A simple model for the chaff cross section is given in [57, p. 199] as

$$\sigma_{chaff} = \frac{6600W}{f} = 22000 \lambda W \quad [\text{m}^2], \quad (2.38)$$

where W is the weight of the chaff [kg], f is the RF frequency [GHz], and λ is the RF wavelength [m].

Leonov and Leonov [54, p. 363] also give a simple chaff model, in terms of the number of dipoles N , as

$$\sigma_{chaff} = 0,18 \lambda^2 N \quad [\text{m}^2]. \quad (2.39)$$

Advanced models. A new chaff cloud model was recently proposed by Marcus [56]. The model is very complex and includes aerodynamic and EM modelling; however, a minor shortcoming is the exclusion of wind effects. EM modelling of a dipole is simple and well understood, so the model complexity lies in the aerodynamics.

The fall velocity of each wire is calculated according an aerodynamic equation, known as the Stokes equation. Marcus also gives PDFs that govern the vertical and horizontal distribution of the fibres. Studies of chaff clouds have shown that they fall with a helical motion, although the elevation angle of each fibre remains constant. Marcus proposes a helical motion model to account for this phenomenon. Once the position of each fibre is known, the RCS is calculated using EM analysis. The EM calculations allow for any frequency or polarisation mode.

Marcus provides approximations to the model to allow for real-time simulation. He also reports results which show excellent agreement with measured data.

2.6.3 Model selection

The whole simulator is essentially a deception jammer — the simulated targets are in fact false targets, so implementing deception jammers for targets in the scenario is almost completely unnecessary. Noise jamming attempts to reduce the signal-to-noise ratio of the target return — as the amplitude of the simulated targets can be controlled, and made as small as desired, an explicit noise jammer is not required.

Chaff is another form of volume clutter, and as with the meteorological effects, it need not be implemented for the prototype simulator. The inclusion of volume scatterer code stubs allows a chaff model to be added to the software framework at a later stage.

2.7 Simulators

In this section, we will review the current trends in simulator design as well as a few example systems. Hollands [41] provides a good summary of these trends, partitioning his review into

simulator functions, architecture and interfacing. We also select the type of architecture to use for our simulator and discuss how the simulator will interface with the radar system.

2.7.1 Functions

In order to generate meaningful signals for the radar under test, all simulators must perform a number of basic functions, within the framework of a hypothetical scenario. Platforms (friend and foe), must be moved along given trajectories, the propagation effects must be modelled, and pulse data must be generated according to the radar's current parameters.

Prior to simulation, the desired scenario must be input and stored in some way. By definition, the scenario includes all the parameters necessary to characterise the radar system, the targets, and the natural environment, for the duration of the simulation. For the simulation itself, Hollands [41] suggests the use of three functions: data preparation, scenario modelling, and pulse generation. Leighty and Perkins [53] prefer to lump the first two together, as general processing.

Data preparation

Before the simulation begins, data needs to be generated in a format suitable for the simulator. Data preparation includes all calculations that are independent of the radar parameters that can be changed during run-time (e.g. boresight angle). Examples of these calculations include target RCS, range, range-rate, and position and velocity vectors.

The target and clutter cross-sections are the most complex part of the data preparation. The complex models needed for high fidelity simulation may require considerable processing time. If target fluctuations are assumed to be statistical, then a consideration is whether to include them in the offline process, or generate them in real-time.

Scenario modelling

Scenario modelling is the link between the prepared data and the pulse generation hardware. Using the prepared data, the scenario modeller performs in-beam checking to generate a list of target and clutter returns. These returns can include amplitude, phase, delay, and doppler shift information. The list must be prepared every pulse repetition interval (PRI), and is then passed on to the waveform generator.

Pulse generation

Most radar systems generate and process raw radar pulses at a phenomenal rate, in order to extract a little information (normally target detections) from the radar environment. By requiring that the simulated signals are input as raw pulses, the demands placed on the data throughput of the pulse generator are very high. The result is that dedicated hardware is usually used for this function.

Essentially, the pulse generator just replays distorted, possibly overlapping versions of the original pulse. This makes the hardware relatively simple, provided the original pulse is available. The only information required from the scenario modeller is when to generate a pulse, and how to distort it. Had this not been the case, it would be very difficult to achieve the desired data throughput directly from a computer running the scenario modeller application.

2.7.2 Architecture

Many modern simulators are based on a similar architecture. There are basically three ways in which the simulation can be approached, depending on the processing time, storage and model fidelity requirements.

- i) All functions are simulated by software off-line, and the resultant data stream is saved in a time tagged format. The data is then played back using dedicated hardware at a later stage. The limitations of this approach include the inability to respond to changes in the radar (e.g. such a simulation would be inapplicable to a tracking radar) and storage problems for long simulation runs. An advantage is that very high fidelity models can be used, as the processing time is not a major issue.
- ii) An off-line software simulation of the platform dynamics and propagation effects, but real-time digital pulse generation using digital hardware. This approach allows for limited system interaction, e.g. the radar can now change waveforms during the simulation. Far less information need be stored, thus allowing for longer simulations. An alternative is to store the information for all possible look angles, in which case functions such as radar tracking can be performed. However, this requires a large amount of stored data. Either way, the off-line processing again allows for high fidelity models. Another disadvantage is that all object trajectories must be known prior to the simulation (e.g. an aircraft which realises that it has been spotted by the radar could not suddenly change course to avoid further detection).
- iii) Real-time software simulation of platform dynamics and radar wave propagation, controlling the real-time waveform generation hardware. This is the most versatile solution, as any run-time decisions by the scenario objects (e.g. changing waveforms or altering course) can be accounted for immediately. No data, other than the scenario description, needs to be generated or stored prior to a simulation run (which can be very long). The disadvantage of this approach is the requirement of a very high performance computer for real-time simulation. The processing power can, however, be traded-off against model fidelity.

We will refer to the above simulators as type *I*, *II* and *III*, respectively.

Architecture selection

The system's architecture is a fundamental design consideration, and must be based on the system requirements (see Section 1.4). Firstly, the application is a tracking radar, so type *I* (completely off-line processing) is not applicable — the simulator cannot know beforehand how the radar will move its antenna during the simulation. Secondly, we are required to allow for relatively complicated models, and still wish to run the simulator from a standard PC. Thus a fully real-time (type *III*) simulator is also not applicable. The best option is thus a type *II* system — off-line processing of the platform dynamics and propagation effects, with real-time pulse generation. This allows the benefits of real-time feedback from the radar tracker combined with the possibility of complex models. The details of the design are presented in Chapter 3.

2.7.3 Interfacing

The waveform generator has to interface with the radar system under test in some manner. This can be done digitally, at baseband, or with RF signals, as described below.

Digital simulation

As the scenario modeller is a software application, its output is inherently digital. If we have access to the RSP's data bus, then a digital waveform generator can simply inject its data directly, without the need for extra analogue conversion hardware. Digital simulators are relatively cheap and small, which can even allow them to be included as part of the radar system.

Digital simulators are incapable of testing the antenna and receiver. However, if these modules are well modelled, then their exclusion will not affect a performance analysis of the system. It also simplifies in-house testing, where it may not be feasible to mount antennas near the laboratory. Another advantage is that the purely digital simulation can model targets with arbitrary motion, in any desired environment.

Video simulation

The only difference between digital and video simulation is that the latter injects analogue baseband signals into the radar. This requires extra hardware in the form of digital-to-analogue converters and filters. There is also a minor noise penalty for such an approach. An advantage of video simulation is that the baseband analogue inputs to a RSP may be more easily accessible than the data bus.

Injected RF

The signal can also be injected further up the radar module chain. RF injection takes place behind the antenna, and can thus test the receiver, but not the antenna. Many expensive RF

components are required to convert the digital data to RF signals for each receiver channel. The hardware may also require the capability of switching frequency bands.

If the RF hardware is available, then the benefit of arbitrary target and environmental profiles can be realised with an injected RF simulator.

Radiated RF

Another RF option is to radiate signals into the receiving antenna. The interfacing is thus exceptionally simple. This approach also allows the full radar chain, from the antenna onwards, to be tested. However, there are major drawbacks: since all signals must come from transmitting antennas, arbitrary target and clutter profiles cannot be simulated. It may in fact be very difficult to move the transmitting antennas realistically, without the use of an aircraft — this would defeat the object of the simulator. A phased array could be used for faster beam steering [72]. The transmitting RF hardware would also be very expensive and may even result in dangerous levels of radiation.

Interfacing selection

According to the system requirements, the simulator must interface with the radar at the video level. The interfacing hardware will thus need to generate three analogue IF signals in order to drive the sum and two difference channels.

2.7.4 Example systems

The design of radar target and environment simulators has come a long way. Early systems, in the 1960s, known as Digital Radar Landmass Simulators required a huge amount of technology: databases were stored on large, rigid disk drives; mainframes were used for control; radar modelling was performed by dedicated hardware signal processors; and the displays were generated using custom frame buffers [7]. A major problem with these systems was their limited fidelity and reconfigurability. Nowadays, the processing power of a modern workstation allows far more accurate simulations to be performed. The software intensive implementation of most modern simulators also makes systems versatile and easily reconfigurable.

High level simulations, for an integrated sensor suite have been reported by Janowiak in 1990 [44], and Mobsby *et al.* in 1993 [59]. Janowiak's work gives an overview of a combat system level simulator. The sensors simulated are radar, sonar and communications links. No details of the lower level sensor simulation is given. The objective of the work by Mobsby *et al.* is to determine the detection performance of the sensor suite. The high level simulator is software based and the sensor performance (signal-to-noise ratio or probability of detection) is taken from lookup tables populated by low level simulators. The low level models include sea and rain clutter, thermal and jammer noise, and multipath effects. Both the simulators by Janowiak and by Mobsby *et al.* allow for complex scenario definition.

Metz and Phelps [58] present a high fidelity, detailed, video simulator for an air-to-air radar in the presence of ECM (reported in 1994). The majority of the processing is software-based. Targets may fly any type of profile, and can carry jammers. Target RCS can be either a fixed value or taken from an attitude dependent lookup table. In either case, Swerling fluctuations can be applied to the mean RCS. The ECM model allows for noise and deception jamming and can be air- or ground-based. Parameters include the jammer's power, bandwidth and waveform. Environmental modelling is relatively simple: attenuation must be specified explicitly, and only ground clutter is available. The clutter can be either homogeneous, in which case the returns are given a Rayleigh distributed envelope; or discrete clutter, in which case one of four models are used to define the returns. The total clutter return is the sum of the discrete and homogeneous parts. A limitation is that doppler shifting is not provided for clutter patches. The simulator includes a fully featured radar processor model. If the radar processor model is not used, the simulator's output is either analogue or digital I and Q data.

A very different approach is taken by Phu *et al.* [65]. Their test target generator is implemented mostly in hardware, using optic fibre to generate delayed returns. The advantage of this simulator is that it can cater for very wideband radar systems. Up to two targets can be generated with independent (steady) RCS, doppler and range. One target has a fixed range, while the other's range can vary slightly around the fixed range. The doppler shift is not linked to the range, and thus higher level processing may disregard the targets if their doppler information does not correspond to their range rates. An incoherent noise source can also be coupled to the system.

Details of a radiated RF simulator for a monopulse radar are given by Sarkar *et al.* [72]. A phased array, using four horn antennas, simulates a moving target. The array's coverage is 11° , with a resolution of 10 mrad. The advantage of using an array is that fast moving targets can be simulated without mechanically shifting the transmitting antenna. No environmental effects are simulated.

BangKui *et al.* [8] describe a real-time radar video signal simulator. The basic system design of the relatively simple simulator is given. Data preparation and scenario modelling are performed on PC, while a Digital Signal Processor (DSP) microprocessor is used for the real-time waveform generation. Point targets, with Swerling fluctuations are used, along with simple pseudo-independent correlated clutter. A simple signal processing algorithm for generating pseudo-independent processes, with a certain correlation time, is given.

An FPGA-based real-time digital radar environment simulator is presented by Andraka and Phelps [6]. Their application is a helicopter-mounted 2D search radar. The simulator is capable of flying hundreds of targets, and includes sea clutter, land masses, weather, jamming and thermal noise. The target models are steady point targets, with a limit of two per range cell. Sea clutter is generated using the compound K-distribution, with correlation provided through a novel random cell replacement process, as the radar scans. The source of the weather, landmass and jamming noise profiles are not given. FGPAs are used for

the time critical tasks: target sorting, waveform generation, clutter and thermal noise generation. Target and radar movement is performed using software on DSP microprocessors. Andraka and Phelps provide a detailed explanation of the FPGA implementation, including the system design, generation of various random processes, and the use of these processes in the clutter generation. Their chosen level of fidelity is also substantiated.

Mahafza *et al.* [55] report a real-time digital target simulator, for use with a ground-based phased array radar. A basic system design is given, which conforms to the architecture we have discussed previously. The whole system is software based, with the scenario modeller running on a Digital Equipment Corporation (DEC) VAX 7000 workstation, and the waveform generator on a Convex S-class system. Target data is generated offline using a complex EM simulation that determines the effective scattering centres for each target at each stage of the scenario. RCS data is then stored for use with narrow, medium and wide-band waveforms. The scenario modeller then uses the RCS data and the current system settings to generate return information. Polarimetric, frequency and atmospheric effects are also accounted for, although no details are given as to exactly what effects are considered. The scenario modeller returns any health messages that would normally be generated by the modules it replaces (receiver/exciter and antenna equipment), thus allowing straightforward integration with the radar system. The digital waveform generator then creates the necessary I and Q data by relating the current transmission waveform to the scenario modeller's information.

Palomino and Schmitz [63] describe a purely software, digital simulator used to generate high resolution radar (HRR) profiles for ground based models. Complex scenarios can easily be represented, as the simulator includes DTED and accurate vehicle dynamics. In order to generate the range profiles, an airborne search radar is modelled. Very high resolution CAD models, accurate to a few centimetres and capable of articulated motion, e.g. movable turrets and hatches, are used. EM scattering prediction codes are then used on the CAD models, along with a scenario defined ground plane, to produce scattering information. The ground plane can be either smooth or have Gaussian roughness. In the latter case, various surface types such as sand and tar can be specified. The combined clutter and target data is then passed to the radar model to generate the HRR range and doppler profiles. This simulator does not interface directly with a real radar signal processor.

2.8 Summary

After a brief discussion of the simulator's target radar system, the bulk of the chapter examined the current state of the art of radar environment simulation. Atmospheric effects, surface reflections and multipath were considered, as was the scattering of RF energy from targets, sea and ground clutter, and meteorological phenomena. Electronic warfare, specifically ECM, provides another source of radar echoes. The chapter concluded with a review of a number of simulators implemented by other authors — the functions, architecture and

interfacing used by these systems were of interest.

Throughout the chapter we considered which models and effects to allow for in the simulator's software framework, and also which models to implement in the prototype system. Invariably, very simple models were chosen for the implementation. This was because the prototype's objective is not to be a fully featured, high fidelity simulator, but rather to prove the basic concept. The design choices made in this chapter are summarised below:

- The antenna pattern will be modelled using measured data of the complex gain for various angular positions in the beam (Section 2.2.1, p. 7).
- Refraction will be ignored (i.e. a refractive index of unity will be used) by the ray tracing algorithm (Section 2.3.1, p. 13).
- Retardation will also be ignored by the ray tracing algorithm (Section 2.3.2, p. 14).
- The ray tracing algorithm must allow for attenuation, but will assume there is none (Section 2.3.3, p. 16).
- The simulator allows for, but does not implement, the PPF model for surface reflections and multipath propagation (Section 2.4.2, p. 24).
- A target model for aircraft will be implemented as a point target with Swerling amplitude fluctuations. Doppler scintillation and glint will not be implemented (Section 2.5.1, p. 32).
- Sea clutter will have K-distributed amplitude fluctuations with the mean determined using Barton's constant model, and the shape parameter determined via the model due to Ward *et al.* (Section 2.5.2, p. 39).
- Ground clutter will be modelled using a single K-distribution with the parameters taken from Jao's work (Section 2.5.3, p. 44).
- The software framework will allow for volume scatterers, but does not implement them (Section 2.5.4, p. 46).
- Jamming is not catered for explicitly, and chaff clouds (being volume scatterers) are not implemented (Section 2.6.3, p. 49).
- The simulator will be based on a type *II* simulator — off-line processing of the platform dynamics and propagation effects, with real-time pulse generation (Section 2.7.2, p. 52).
- The simulated returns will be injected into the radar as video signals (Section 2.7.3, p. 53).

The remainder of this thesis is concerned with the design, implementation and testing of the prototype simulator.

Chapter 3

System design

3.1 Introduction

The basic system architecture and model considerations were discussed in Chapter 2. In the current chapter, the implications of these early design decisions are seen, as the detailed design progresses. The chapter starts with a functional analysis of the simulator and its interfaces, and then moves on to the conceptual design.

3.2 Functional analysis and system architecture

As discussed in Section 2.7.2 (p. 51), a type *II* architecture — off-line processing of the platform dynamics and propagation effects, with real-time pulse generation — was selected. Figure 3.1 shows the functional block diagram for this system.

The two main components of the type *II* architecture are evident in the diagram. The majority of the processing takes place on a computer, after which the data are sent to the waveform generator's dedicated hardware. This hardware generates the pulses which are fed into the RSP in real-time. It is noted that while the PC software may easily be generalised to many radar systems, the waveform generator is far more application specific.

3.2.1 Functional units

The description of each functional unit (FU) is as follows:

User interface (FU1). Allows the user to view and change the current scenario, and the system's settings. It also provides feedback to the user during real-time playback.

Scenario (FU2). Stores a description of the scenario, including the radar sensor and targets' parameters and trajectories. Volume scatterers and terrain and other environmental details are also included.

Pre-processing engine (FU3). Calculates all the parameters of the simulation that are independent of the radar's look angle. All complicated model processing, e.g. ray

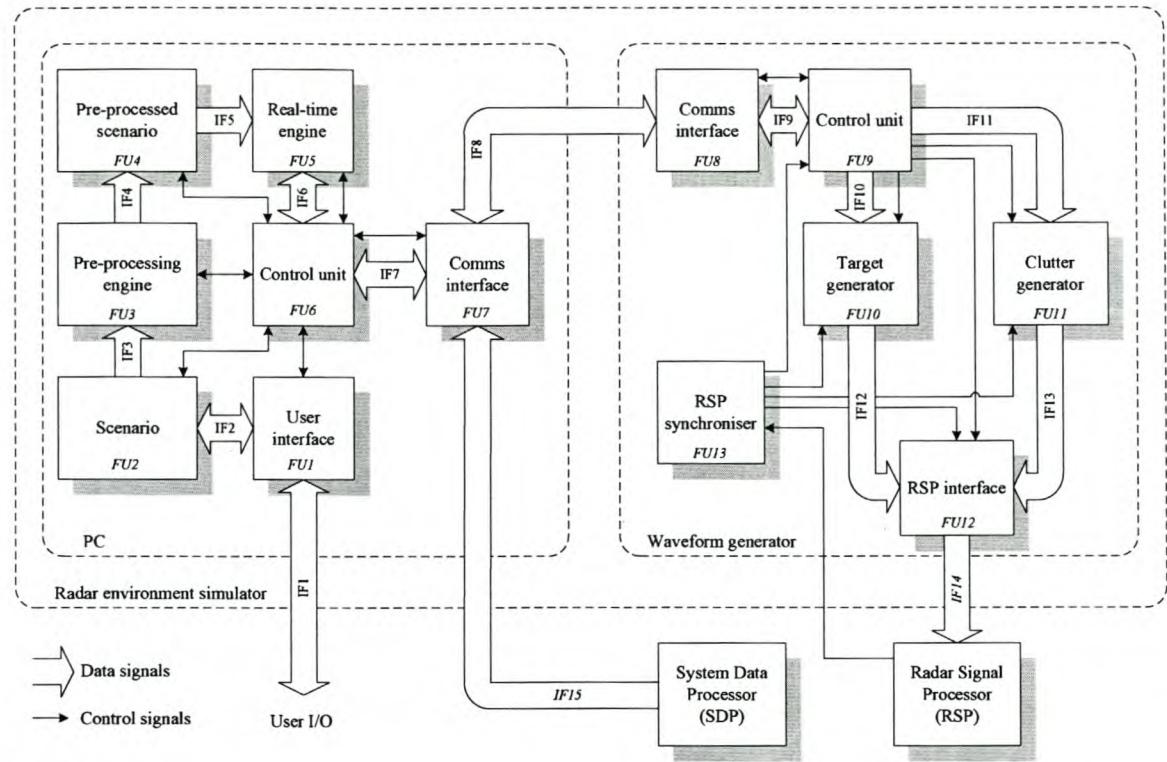


Figure 3.1: Functional block diagram of the simulator system, showing system boundaries.

tracing, is performed prior to the simulation run.

Pre-processed scenario (FU4). Stores the pre-processed information for later playback.

Real-time engine (FU5). This engine uses the radar's current look angle to determine which scatterers are in the beam. The radar's antenna patterns are then applied to the pre-processed information in order to generate the descriptors for the real-time returns.

Control unit (FU6). All system functions on the PC are co-ordinated by this control unit. The required parameters and control signals are passed to all modules, dependent on the system's current mode of operation.

Communications interfaces (FU7 & FU8). Handle the sending and receiving of data between the PC and the waveform generator.

Control unit (FU9). Co-ordinates all system functions on the waveform generator. Processes incoming data from the PC and sets up the waveform generator accordingly. A synchronisation message is periodically sent to the PC.

Target generator (FU10). Uses the descriptors of the targets currently in the beam to generate phase, frequency and amplitude modulated versions of the RSP's waveform.

These are output, at IF, after a range dependent delay.

Clutter generator (FU11). Generates samples from various PDFs for the in-beam clutter sources.

RSP interface (FU12). Combines and converts the digital signals from the target and clutter generators to analogue IF signals.

RSP synchroniser (FU13). Synchronises the waveform generator with the RSP — the start of each pattern, burst and PRI is derived from the RSP's timing signals.

3.2.2 Interfaces

The interfaces (**IF x**) are described below (unless otherwise stated, interfaces use digital data).

- IF1.** Mouse and keyboard input from the user and visual feedback to the user.
- IF2.** Information about the scenario — can be read in order to display the scenario via the user interface, or written in order to change the scenario.
- IF3.** Information about the scenario is read in order to do the pre-processing.
- IF4.** The pre-processed data is written to a data store prior to the simulation run.
- IF5.** Pre-processed target and clutter returns are read in by the real-time engine.
- IF6.** The current boresight angle is written to the real-time engine, and the real-time returns are read by the control unit.
- IF7.** The real-time return information is written to the communications interface. The boresight angle, status and synchronisation information is read from the interface.
- IF8.** The status, synchronisation and real-time return information passes over the communications channel.
- IF9.** The real-time return information is read from the communications interface, and status and synchronisation information is written to it.
- IF10.** Real-time target return information is written to the target generator.
- IF11.** Real-time clutter return information is written to the clutter generator.
- IF12.** The target return pulses are written to the RSP interface.
- IF13.** The clutter return signal is written to the RSP interface.
- IF14.** The combined clutter and target returns are output as analogue signals to the RSP's Σ , Δ_{AZ} and Δ_{EL} channels, at IF.

IF15. The current boresight angle is sent over an Ethernet link, using the TELNET protocol [68].

3.3 Conceptual design

We lead in with the system-wide design decisions, and then consider the partitioning of the system's software and hardware. Most of the software is analysed as part of the concept of execution, while the firmware and hardware are discussed separately.

3.3.1 System-wide design decisions

Prior to the details of the conceptual design, we discuss our fundamental system-wide design decisions.

Programming language. It was decided to use C++ to code the core of the simulator engine and the user interface. C++ allows for an object-orientated design, combined with efficient code (more so than Java or Python, for example). Speed of execution is important in this system, so the efficiency of the code must be considered. In addition, C++ tools and experts were easily accessible. C was chosen for the embedded software as a number of design examples were already available in this language.

Coding standards. The RRS coding standards were chosen, as the work was commissioned by them. The standards encompass naming and layout conventions, and some simple coding rules.

Inter-system communication. It was decided to use the standard RRS messaging scheme to communicate between the two main system components (PC and waveform generator). In general, each command or request message is acknowledged with a response message. More details will be given when describing the software implementation (Chapter 4).

Hardware. A field programmable gate array (FPGA) development board (*Nios Development Board, Stratix Edition*, from Altera) was the only development kit readily available, and thus was chosen for implementing the waveform generator. A development kit was preferred, as hardware development was not a major objective of this project. Conceivably, a DSP board could be used to fulfil the same task.

The FPGA board includes Joint Test Action Group (JTAG), serial (RS232) and Ethernet (100 Mbps) communications links. It has no digital-to-analogue converters, but does have prototyping headers.

Sampling rate. The maximum output frequency is specified at 15 MHz, thus the minimum theoretical sampling rate is 30 MHz. We choose to run the waveform generator at a sampling rate of 40 MHz, which meets this requirement and also matches the RSP's

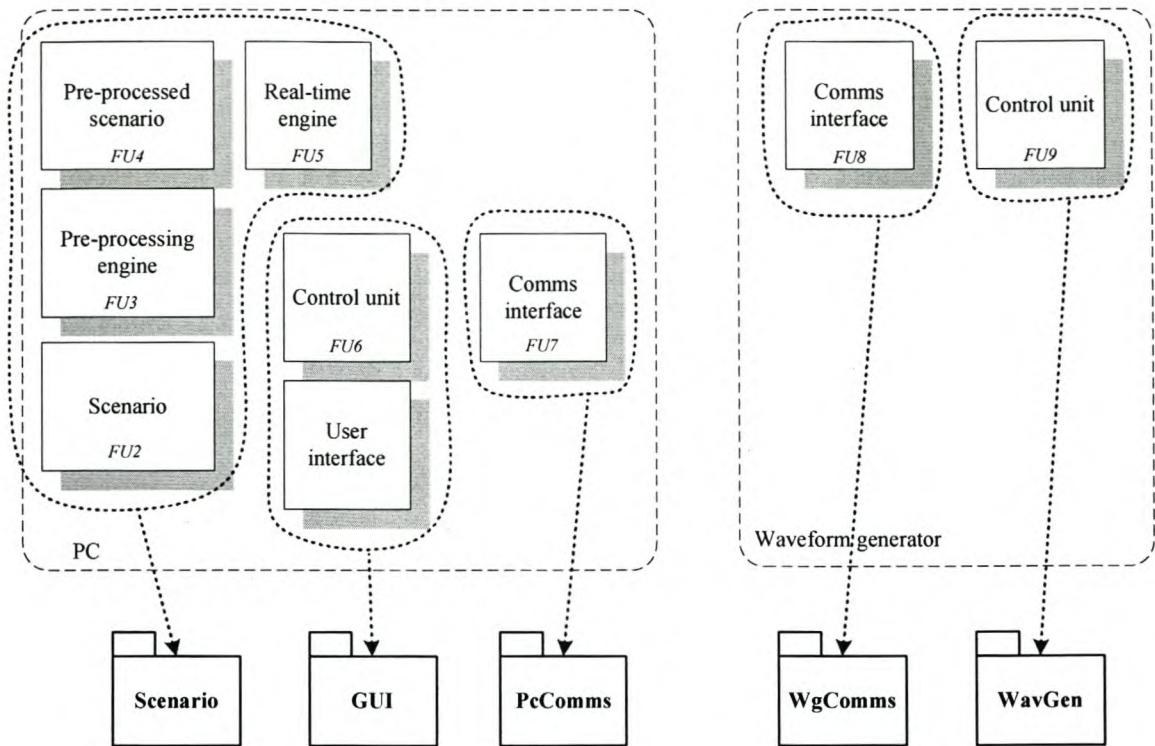


Figure 3.2: Mapping of functional units to software packages.

IFS sampling rate. Given that the maximum output frequency is 15 MHz, the DAC's alias signals will not occur at frequencies lower than 25 MHz. The IFS has filters prior to its ADCs which must provide sufficient suppression for any signal above the Nyquist frequency (20 MHz). As the DAC aliasing only occurs at higher frequencies, no additional filtering is necessary on the DAC's output.

Modelling. In order to meet the requirement of a largely software defined simulator, all models must be implemented in the simulator's core software. I.e. no modelling effects are applied by the waveform generator. This creates a flexible design where models can be changed with minimal knock-on effects.

3.3.2 Software architecture

Figure 3.2 shows the mapping of the functional units to software packages. The core of the simulator is contained in the *Scenario* package. The control unit is combined with the graphical user interface package in order to reduce the number of pass-through functions and thus simplify the implementation.

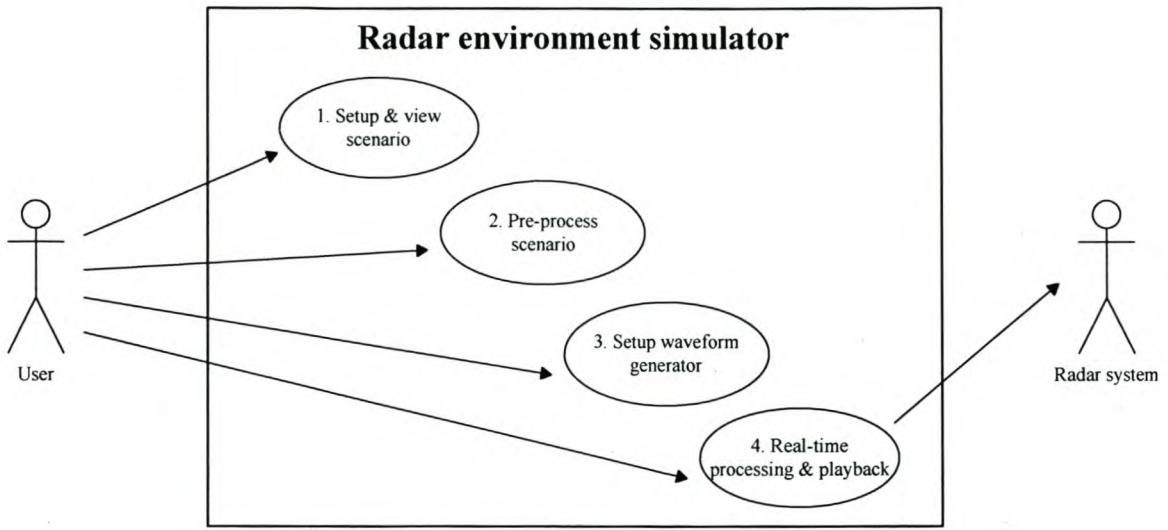


Figure 3.3: System use case diagram.

3.3.3 Concept of execution

In order to understand the system's execution we use a number of Unified Modelling Language (UML) diagrams. First the functions are defined, and then the sequencing and program flow are considered in more detail.

Use cases

All four use cases are shown in a single UML use case diagram (Figure 3.3), for brevity. It is clear from the diagram that the first two use cases require no interaction with the radar system. This is an advantage as the time consuming pre-processing can be performed on any available machine. Allowing the user to modify the waveform generator's default settings makes the system more versatile, and simplifies testing.

Sequencing

With the use cases defined, it is enlightening to view the time-dependent interaction of the various system components. This is shown in the UML sequence diagram (Figure 3.4). Note that the PC and waveform generator communications interfaces have been lumped together in *itsCommsEngine* to simplify the diagram. We note the following:

- The radar system continuously sends timing signals to the waveform generator. In line with the use case diagram, these are ignored, unless the system is busy with real-time playback.
- Modifying the scenario and pre-processing require very simple sequencing.

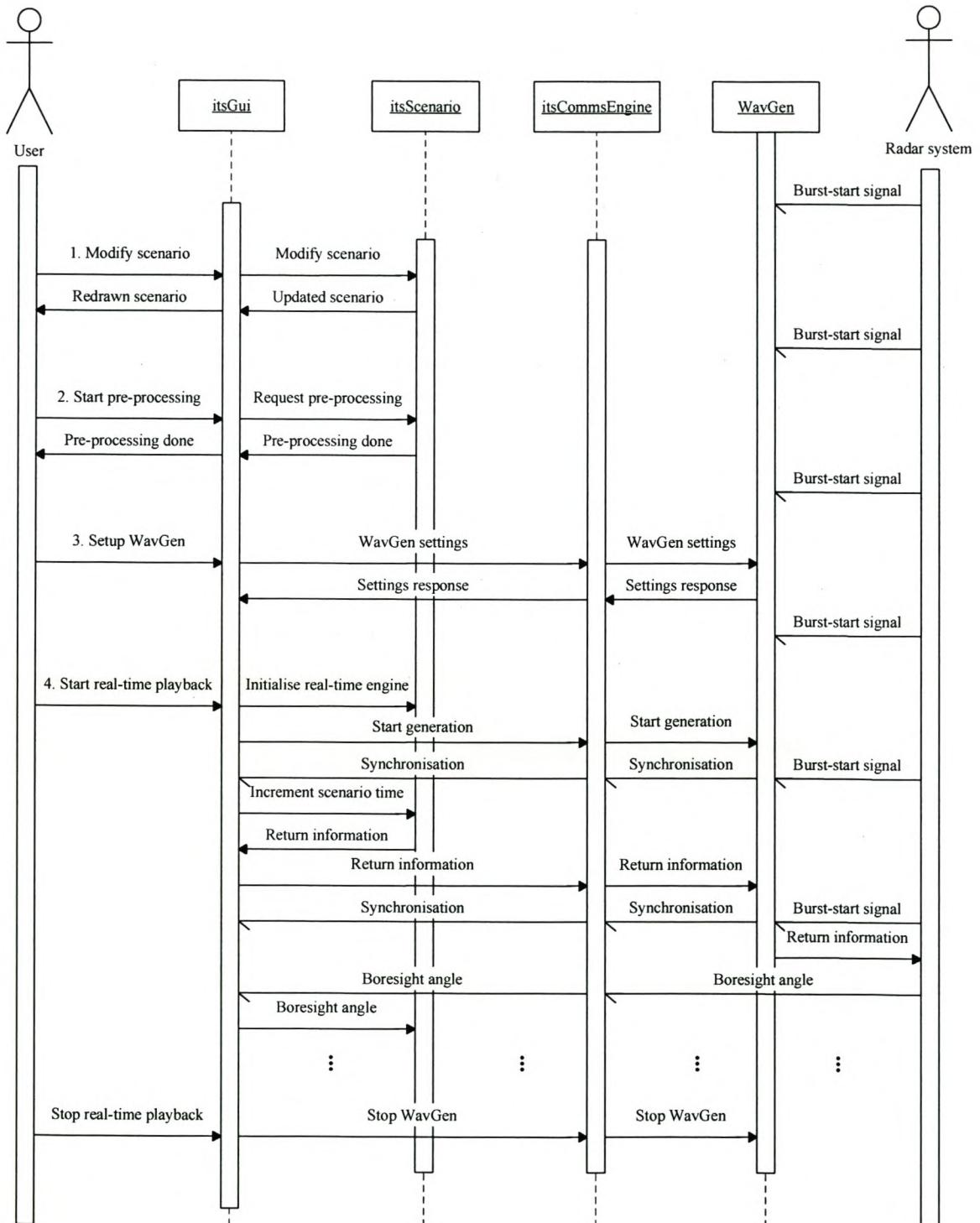


Figure 3.4: System sequence diagram.

- During real-time playback, the scenario is driven by the synchronisation message from the waveform generator. This ensures that the simulation stays in step with real-time.
- On reception of the synchronisation message, the GUI requests that the simulation time is incremented. The scenario responds with return information for the next step.
- This new return information is sent to the waveform generator, which plays it out after the next burst-start signal. Note that the simulator must be synchronised with the RSP waveform pattern (see Section 3.3.8, p. 74).
- The boresight angle is sent from the radar system periodically (but asynchronously to the burst-start signal). On reception, the GUI informs the scenario of the new angle.
- Real-time playback continues until stopped by the user.

Program flow

The basic program flows of the pre-processing engine, real-time engine and the waveform generator are shown, using UML activity diagrams. The first use case, setting up and viewing the scenario, is straightforward and will not be discussed.

The pre-processing engine’s activity diagram is shown in Figure 3.5 — we note the following design decisions:

- The radar is static, so the surface clutter map need only be calculated once.
- The scenario’s terrain is described by a grid of terrain blocks.
- The surface scatterers (i.e. the terrain) are found by sweeping in azimuth at each range — this generates a sectorised surface clutter map.
- The visible angles are the locus of boresight angles in which the radar’s beam illuminates the scatterer.
- The environmental parameters include the range, delay, phase angle, atmospheric attenuation, and the scatterer’s relative azimuth and elevation angle.
- The pattern propagation factor (PPF) depends on the scatterer’s position in the beam, but a number of the coefficients in eq. (2.10) can be predetermined (see the implementation details, p. 86). The final calculation must be done by the real-time engine.

Figure 3.6 shows the real-time engine’s program flow. We note the following design decisions:

- The pre-processed clutter map is loaded as part of the real-time engine’s initialisation (not shown).

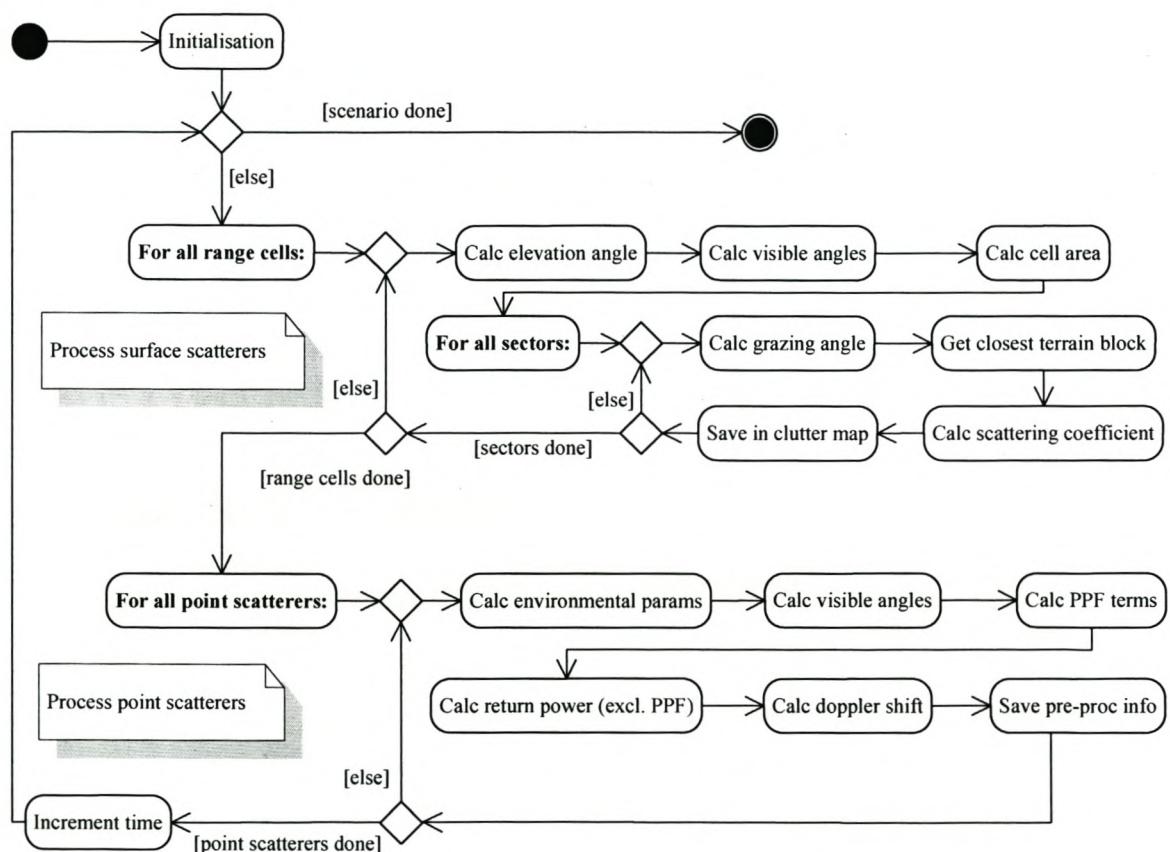


Figure 3.5: Pre-processing engine's activity diagram.

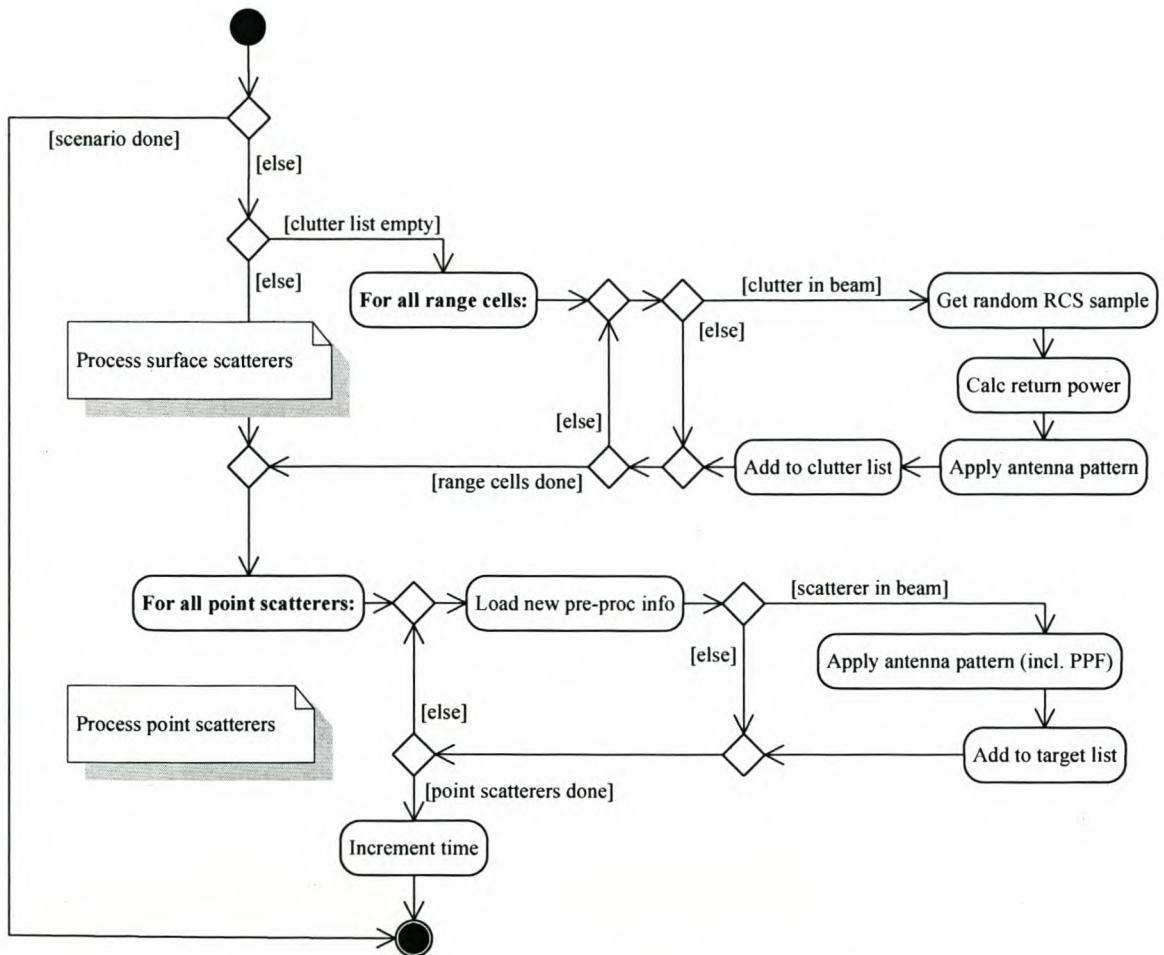


Figure 3.6: Real-time engine's activity diagram.

- To simplify the implementation, clutter returns cannot change pulse to pulse (thus the clutter can have no doppler spread). The implication is that clutter will appear static (relative to the radar), and its correlation properties cannot be controlled. In line with this implementation, new clutter return information (stored in the clutter list) is only calculated once the previous information has been sent to the waveform generator (list will be empty).
- The clutter map includes only the PDF parameters for the clutter — the real-time engine generates the corresponding random samples. No allowance has been made for temporal or spatial correlation — this could be added by filtering the clutter returns, however it may prove too processor intensive for the real-time engine.

The waveform generator's program flow (Figure 3.7) has an action for modifications — this includes changing parameters and copying new return information and pulse waveforms to the return generators.

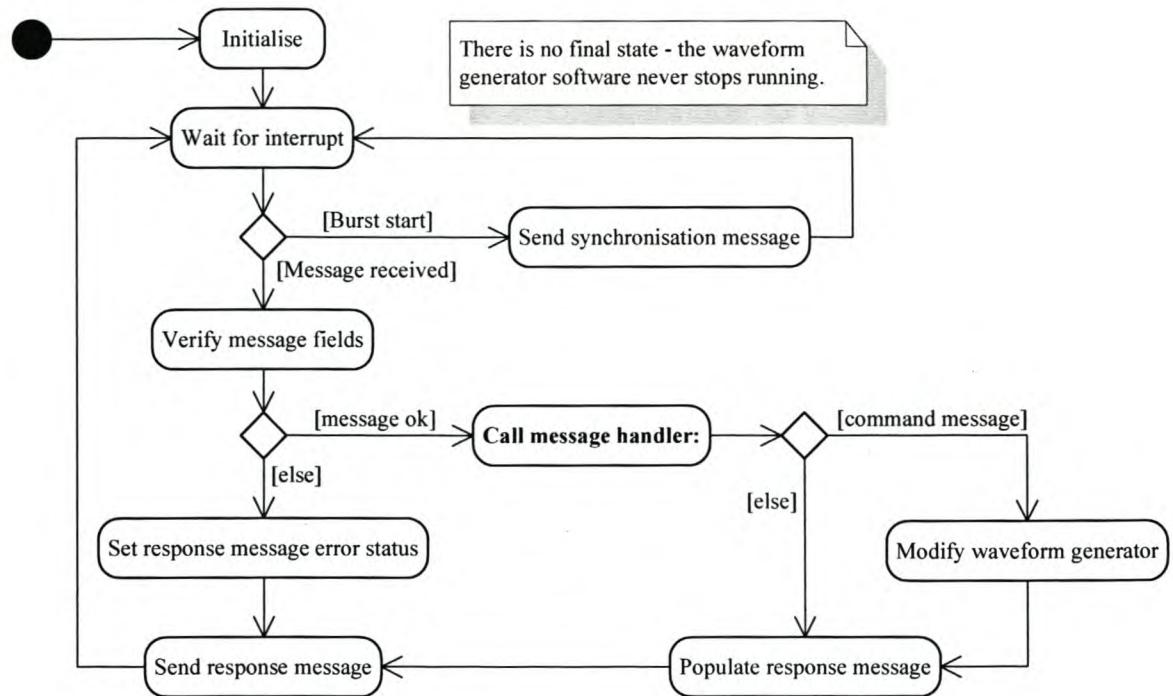


Figure 3.7: Waveform generator's activity diagram.

3.3.4 Waveform generator control unit

Using the selected development kit, the obvious choice for the waveform generator's control unit was the *Nios* central processor unit (CPU). It can be synthesised as part of the FPGA design, using Altera's *System on Programmable Chip Builder (SOPC Builder)*. The most critical design decision was whether to use the 16-bit or 32-bit variant. As the waveform generator's main task is transferring a large amount of data, and the amount of data shifted each clock cycle is proportional to the data bus width, the 32-bit processor was chosen. It was also decided to include a data and instruction cache — this is important, as the data and instruction memory is shared (Von Neumann architecture).

Copying the return information into the target and clutter generators is time critical and a direct memory access (DMA) architecture was considered. However, as the Ethernet controller shares the CPU's data bus, DMA was not used. The CPU cannot send or receive any messages, or even execute uncached instructions, until the copying is done, so DMA provides an insignificant performance improvement.

3.3.5 Target generator

The target generator is the most complicated part of the FPGA logic. It consists of a large number of modules, but we only discuss those at the top level — Figure 3.8 shows the functional diagram. The functions are explained in the sections that follow.

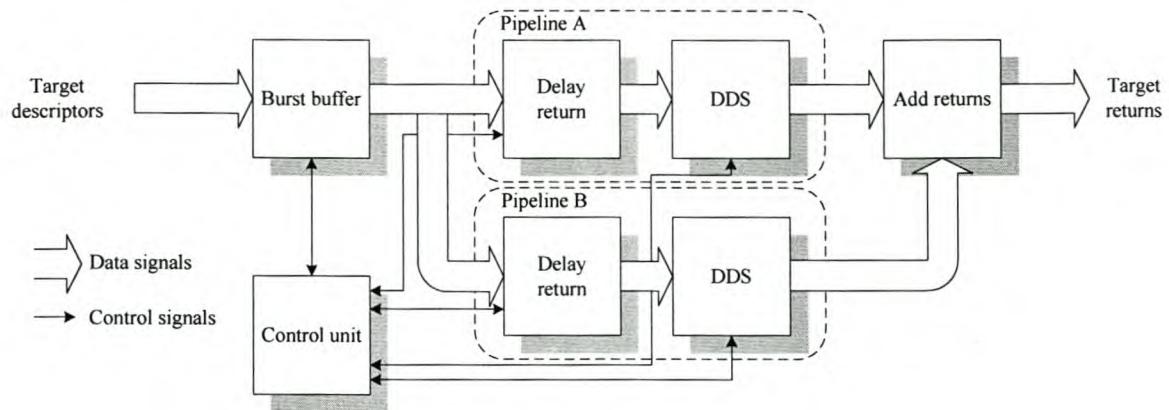


Figure 3.8: Target generator functional diagram.

Buffering

The link between the PC and waveform generator uses UDP/IP over Ethernet (see Section 3.3.9). This medium's latency and inherently bursty nature requires that the target returns are buffered in some way. For illustrative purposes, we use a PRF of 5 KHz (reasonable for this type of pulse doppler radar). At this rate, buffering only a single pulse requires new data every 200 μ s. Under Windows XP, a non-real time operating system, with a minimum time-slice granularity of 1 ms [91], single pulse buffering is not feasible. Buffering a very large amount of data reduces the responsiveness of the system which will hinder closed loop tracking. A reasonable compromise, which fits logically into the radar's timing schedule, is to buffer one burst of pulses. For the majority of waveform patterns, a burst will be a few milliseconds long, thus providing the time in which to transfer the necessary data.

We use two burst buffers accessed via a ping-pong scheme — the control unit writes to one buffer while the target generator plays out the burst from the other. The bursty nature of the link means that occasionally the new buffer may not be full when the next burst starts. If the target generator were to switch buffers at this point then the next burst's returns would not be fully correlated pulse to pulse (the doppler phase shift especially). To prevent this problem, the control unit must assert a data valid signal once copying is complete.

Pipelines

There is a requirement for the ability to generate two targets in the same range cell. As the targets will invariably have different parameters, two separate pulse generating pipelines are needed. The output from these pipelines must then be summed to generate the total return.

A pipeline consists of a return delaying module and a direct digital synthesis (DDS) module. The former waits until the number of clock ticks elapsed in the current PRI is equal to the number of cycles specified by the target's descriptor delay. Range ambiguity is not allowed. The DDS module is used to generate a frequency, phase and amplitude modulated

version of the pulse waveform.

The pipeline is busy until the DDS has finished generating the pulse. The return delaying module can only accept a new target descriptor once the pipeline is free again. The software must ensure that the targets are sufficiently far apart, or otherwise the pipeline will fail to generate all the target returns.

The maximum delay length limits the simulator's maximum range. At a sampling rate of $f_s = 40\text{ MHz}$, the range resolution is $\Delta R = c/(2f_s) = 3.75\text{ m}$. The maximum range R_{max} is simply the product of ΔR and the number of delay cycles d_{max} . For the required 200 km range, $d_{max} \approx 53\,000$. This requires 16 bits to represent the delay, and gives $R_{max} \approx 246\text{ km}$. This is more than adequate for a large number of radar systems.

Loading returns

The control unit is needed to load the returns from the burst buffer into the pipelines. This module must wait until a pipeline is free before fetching the next return from the buffer. To simplify implementation, it is required that the software provide a range-sorted target descriptor list. If there are less than eight targets in the beam, then the unused descriptors can be effectively ignored by setting their amplitude to zero. This requires that all eight descriptors are sent for every pulse — it is slightly inefficient, but allows the system to perform consistently regardless of the number of targets in the beam.

DDS engine

The DDS engine's architecture is obtained by analysing the equation for target returns. We then study the DDS process a little more closely, discuss doppler shifting, and finally select the necessary design parameters.

Target returns. A detailed analysis of target returns is given in [90, pp. 238–242]. The most important result is that the compression or stretching of the return's complex envelope, due to the target's radial speed is negligible for radar systems. For our application, the target return is a sinusoidal pulse that must be generated for each channel (Σ , Δ_{AZ} and Δ_{EL}). For the duration of the pulse, the continuous time signal (excluding noise), is

$$p_{ch}(t) = A_{ch} \sin [2\pi(f_c + f_D)t + \theta(t) + \phi_{ch}] , \quad (3.1)$$

- with A_{ch} = channel dependent amplitude [V],
 f_c = pulse's centre frequency at IF [Hz],
 f_D = pulse's doppler frequency shift [Hz],
 t = time since the start of the pulse [s],
 $\theta(t)$ = pulse's phase modulation [rad],
 ϕ_{ch} = channel dependent phase shift [rad].

The channel dependent parameters are due to the different antenna patterns for each channel. From eq. (3.1), the design of the DDS is relatively clear. For a pipeline, a single lookup table

for the phase modulation, and a single phase accumulator is needed. For each channel, ϕ_{ch} must be added to the accumulated phase; the sine of the angle generated; and then multiplied by the scaling factor A_{ch} . Most of these functions map into an FPGA relatively well — the exception is the sine function.

Two common methods of implementing trigonometric functions in programmable logic are lookup tables (LUTs) or co-ordinate rotation digital computers (CORDICs) [5]. Using CORDICs, two-dimensional vectors can be rotated using only shift and add operations. Rotation through a given angle is an iterative process analogous to a binary search, with the result gaining approximately one bit of accuracy per cycle.

LUTs are fast, but memory intensive, while CORDICs have more latency and use more logic, but require minimal memory. As there is more logic on the FPGA than memory, and most of the memory is required for the target and clutter return buffers, it was decided to use the CORDIC approach. The additional latency reduces the minimum target range slightly (tens of metres), but this is not an important design parameter.

DDS. DDS is a conceptually simple process where a phase accumulator is incremented at a rate of f_s Hz. If the phase increment $\Delta\phi$ is equivalent to $2\pi/f_s$ radians per second, then the output frequency (taking the sine of the accumulated phase angle) will be 1 Hz. Smaller or larger increments can be used to synthesise different frequencies. In general, the phase increment is related to the desired output frequency f_o by [89, p. 8]

$$\Delta\phi = 2\pi \left(\frac{f_o}{f_s} \right) . \quad (3.2)$$

The number of bits P required for a binary phase accumulator (representing angles from 0 to 2π radians), is [89, p. 10]

$$P = \left\lceil \log_2 \left(\frac{2(f_c + f_D)}{\Delta f} \right) \right\rceil , \quad (3.3)$$

where Δf is the required frequency resolution, and $\lceil \cdot \rceil$ is the ceiling function (rounds its argument to the nearest integer, towards $+\infty$).

In general, the phase accumulator's value is truncated to Q bits before generating the sine of the angle. I.e. the angle resolution is $2\pi/2^Q$ rad. The phase accumulator can be viewed as a fixed point number with the Q most significant bits the integer part and the remaining $P - Q$ bits, the fractional part. Truncation reduces the size of the lookup table (or the width of the CORDIC data buses) needed, at the expense of more spurious noise. As a rule-of-thumb, the spurious free dynamic range will be about $6 \cdot Q$ dB [89, p. 11].

A problem arises when using eq. (3.3) with sinusoidal pulses, as opposed to continuous wave signals. With only a limited number of samples in the pulses, arbitrary frequency resolution cannot be obtained. Consider two pulses of length n , with distinct phase increments

$\Delta\phi_1$ and $\Delta\phi_2$. The samples output by the DDS for the two phase increments will only be different if

$$|\Delta\phi_1 - \Delta\phi_2| > \frac{2\pi}{2^Q n} \text{ rad.} \quad (3.4)$$

If the difference is too small, then after each of the n additions of either $\Delta\phi_1$ or $\Delta\phi_2$, the integer part of the phase accumulator will be the same. The signals will be identical and correspondingly their frequencies.

Substituting eq. (3.2) into eq. (3.4), with $\Delta\phi_1$ set to generate a frequency of f_o and $\Delta\phi_2$ set to generate a frequency of $f_o + \Delta f$, we get

$$\left| 2\pi \left(\frac{f_o}{f_s} \right) - 2\pi \left(\frac{f_o + \Delta f}{f_s} \right) \right| > \frac{2\pi}{2^Q n} \text{ rad.} \quad (3.5)$$

which can be simplified to

$$|\Delta f| > \frac{f_s}{2^Q n} \text{ Hz.} \quad (3.6)$$

We see that Q has the largest impact on the DDS's frequency resolution.

Doppler shift. It is important to note that the RSP does not measure a return's doppler frequency shift directly, but rather derives this information from the pulse-to-pulse phase difference. This means that the DDS need not apply a doppler shift to a target's return pulse. However, the frequency shift does have an effect on the output, discussed below, so we include it.

The frequency shift causes a decorrelation between the return pulse and the RSP's stored pulse. The peak output of the matched filter (or pulse compressor) is reduced and shifted slightly from zero lag [57, p. 256]. Figure 3.9 shows an ambiguity diagram for a linear frequency modulated (FM) chirp (an ambiguity diagram plots the pulse compressor's peak output versus doppler shift and apparent delay).

Design parameters. As the pulse-to-pulse phase shift is the critical factor in determining a target's doppler speed, we are more concerned with the output's instantaneous fidelity than with its history-dependent frequency.

The RSP generates pulses using a 12-bit DAC — the waveform generator's output need not be more accurate than this, so we also use 12-bit DACs. The output of the CORDICs must thus be accurate to 12 bits. After the CORDIC, the data is multiplied by the channel amplitude and then added to the second pipeline's output and the clutter. To limit the loss of precision during the multiplication and two additions, two extra bits are kept after the multiplication.

If the CORDIC output need only be accurate to 12 bits, then the input angle need not be more accurate than 12 bits. The CORDIC function's input angle range is only $-\pi/2$ to $+\pi/2$ radians [5], so this is the range represented by the 12 bits. Using an extra bit and

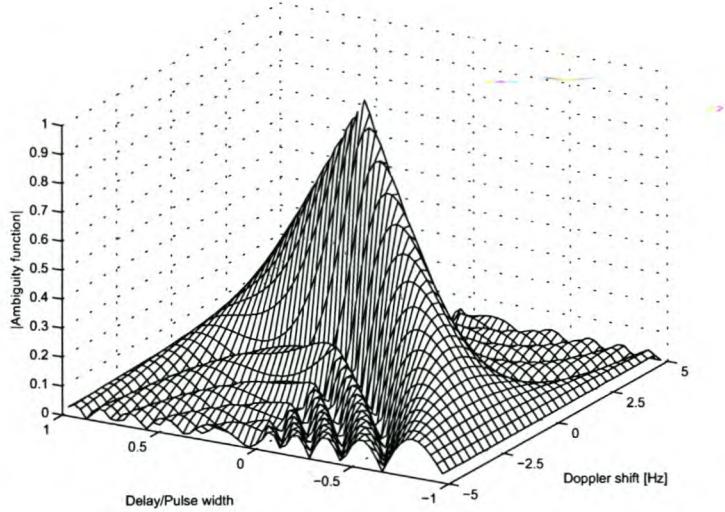


Figure 3.9: Example of an ambiguity diagram for a linear FM chirp.

pre-rotation, the range can be extended to cover a complete revolution. Our design choice for the phase angle's integer part is $Q = 13$ bits.

The maximum number of samples needed to store the pulse can be calculated by multiplying the maximum pulse length of $12,8\mu\text{s}$ (see Section 1.4) by the sampling rate of 40MHz . This calculation gives $n = 512$ samples, so the DDS's theoretical frequency resolution is $\Delta f = 9,54\text{Hz}$, according to eq. (3.6). The doppler effect — see eq. (2.21), p. 26 — can be used to relate the target speed resolution Δv_r to Δf . The lower the transmission frequency, the more coarse Δv_r . At the lowest X-band frequency ($f = 8\text{GHz}$), $\Delta v_r = \Delta f c/2f = 0,179\text{m/s}$.

The last parameter of interest is the total width of the phase accumulator P , for which the maximum doppler shift is needed. From the requirements, the maximum target speed, relative to the radar, is 1715m/s . The maximum doppler shift in X-band, will be $f_D = 137\text{kHz}$, at a transmission frequency of 12GHz . Substituting f_D into eq. (3.3), gives the minimum width as $P = 23$ bits.

Target descriptors The information needed by the waveform generator to create a target return is contained in a target descriptor. Table 3.1 shows the fields. *Frac.* 2's comp. refers to two's complement signed fractional representation, and *unsigned frac.* to unsigned fractional representation — see [1] for a description. The following paragraphs explain the choices that have not already been covered.

Using eq. (3.2) and the fact that a phase accumulator value of 2^P corresponds to 2π radians, $\Delta\phi = 1$ represents a frequency of $4,77\text{Hz}$. This is half of the achievable Δf , and means that the doppler phase increment $\Delta\phi_d$ should only increase in steps of two. The lowest bit will always be zero and need not be included in the target descriptor. For the

maximum required doppler shift, $\Delta\phi_d = 28\,731$. Dropping the least significant bit, and adding a sign bit for positive and negative shifts, 15 bits are needed to represent the doppler phase increment.

The width of the phase angles are chosen to match the 13 bit input to the CORDIC sine function, and the width of the amplitudes are chosen to match the 12-bit output.

The total size of a descriptor is 106 bits = 13,25 bytes . The CPU has a 32-bit data path, so for maximum speed when copying, the target descriptor should be a multiple of 32 bits. Were this not the case, the target descriptors would have to be masked out of the message payload — a slower process. Aligning the fields to 16 bits in the target return list message, gives a 128-bit, or 16 byte, descriptor.

Table 3.1: Target descriptor fields with bit lengths and the corresponding ranges.

Field	Bits	Representation	Minimum	Maximum
Delay	16	Unsigned	0 km	246 km
Doppler phase increment	15	Frac. 2's comp	-156 kHz	156 kHz
Σ amplitude	12	Unsigned frac.	0	0.9998
Δ_{AZ} amplitude	12	Unsigned frac.	0	0.9998
Δ_{EL} amplitude	12	Unsigned frac.	0	0.9998
Σ phase angle	13	Frac. 2's comp.	0°	359,96°
Δ_{AZ} phase angle	13	Frac. 2's comp.	0°	359,96°
Δ_{EL} phase angle	13	Frac. 2's comp.	0°	359,96°

3.3.6 Clutter generator

In line with the software-defined modelling approach, the clutter generator hardware is very simple. All random samples are generated by the PC and streamed to the waveform generator. The clutter generator is essentially a block of memory than plays out the clutter samples. As with the target generator's buffer, we use a ping-pong approach so that new data can be loaded without affecting the current output. Buffers are switched every burst.

As discussed in the software design (Section 3.3.3), this approach allows for a relatively simple implementation. However, the clutter cannot change pulse to pulse, and thus cannot have any doppler spread.

The clutter memory is loaded in blocks, each corresponding to one clutter return list message from the PC. As shown in the software design, only one clutter message is sent per burst. Thus the update rate of the clutter is equal to the number of messages needed to fill the memory divided by the burst frequency.

The FPGA on the development board has one large block of memory (64 kB) and many smaller ones (4 kB) — the large one is selected for the clutter buffers. Half of this can be used for each of the ping-pong buffers. Taking the 32-bit CPU's data path into account, it

is logical to use 32 bits to represent the clutter per range cell. This allows 8192 cluttered range cells, giving a maximum range of approximately 30 km (the range resolution is 3,75 m due to the 40 MHz sampling rate).

The amount of clutter cell information per message is limited by the maximum payload size for the selected protocol (see Section 3.3.9). Sending the information for 320 cells per message (1,2 km range), gives a reasonable message payload length of 1280 bytes. Thus 26 messages are required to update one complete clutter buffer.

As previously mentioned, each range cell has a 32-bit descriptor. For symmetry between the three channels, 10 bits are allocated to each, with the remaining 2 bits unused. This does reduce the fidelity of the clutter, compared to the 12-bit target returns. As mentioned in the project objectives (Section 1.3, p. 2) the fidelity of the prototype simulator is not a critical design requirement, so the reduction in clutter fidelity is not a major concern.

3.3.7 RSP interface

The RSP interface includes a section in the FPGA that prepares the data for the DACs, and a 3-channel DAC board that attaches to a prototype header on the development board.

FPGA module. The FPGA module has two simple functions. The first, is to add the signals from the target and clutter generators together. The second, is to convert the result from fractional two's complement data to straight binary.

DAC board. A commercial DAC board could have been purchased for the development kit, however it was much more cost effective to design and build one.

The requirements for the DACs were a 12-bit input and a sampling rate in excess of 40 MHz. Only current steering DACs provide the necessary speed. No high-speed, 3-channel DACs were available, so it was decided to use one dual-channel DAC and one single channel DAC — both from the same series. The design choice was the DACx902 series from Texas Instruments. These are 165 megasample-per-second devices, with a 0,1 % settling time of 30 ns, and rise and fall times on the order of 2 ns — more than adequate for this application.

The design is simple: the DACs' control and data lines must be connected to the prototype headers, decoupling added, and the output signals converted from differential to single ended outputs. RF transformers provide the necessary conversion, level shifting and impedance matching to 50Ω . The design is based on the application notes in the DAC datasheets. The schematic design is shown in Appendix C.

3.3.8 RSP synchroniser

In order for the simulator's targets to appear coherent to the RSP, the simulator must be synchronised with the RSP's clock and pulse-start signals. The RSP transmits pulses in patterns, each pattern consisting of a number of bursts, and each burst, a number of pulses.

As the PRI used for the pulses in each burst can differ, it is critical that the simulator is synchronised with the start of the pattern as well. This is achieved using two modules — a pattern synchroniser and a pattern counter.

Pattern synchroniser

The last PRI in each burst is slightly longer than the others (specifically for the RSP we are using — it is not a general characteristic of radar systems). Given the nominal PRI, the start of the next burst can easily be detected by measuring the length of each PRI. Pattern synchronisation can be achieved if the PRIs in each burst are different. For a simple implementation, we require that the last burst in the pattern has the longest PRI. In this case, the synchroniser can detect the start of the pattern, given only the longest nominal PRI.

Pattern counter

The pattern counter keeps track of the current burst and pulse number. The pattern counter requires the number of pulses in each burst (the information must be provided by the user, via the GUI). Using this information, combined with the pattern- and pulse-start signals, only a few counters are needed for the implementation.

3.3.9 Interface design

In this section we only consider the link between the PC and the waveform generator (IF8, Figure 3.1). The other external links are already defined by the radar system, and the internal interfaces are trivial.

In order to determine the physical and logical layers required for the link, we consider the following aspects:

- The throughput and direction of the data sent over the link.
- The available hardware and software.
- The data integrity requirements.

A small part of data sent over the link will consist of synchronisation and setup messages and the corresponding acknowledgements. The majority of the bandwidth requirements stem from the target and clutter return information that must be streamed to the waveform generator.

Messaging

One of the system-wide design decisions was to use messages for inter-system communication. Appendix C gives a detailed description of each message sent over the link.

Physical layer selection

There is a relatively large amount data to be transferred, so the FPGA board's fastest link should be used. The 100 Mbps Ethernet link is thus preferred to the JTAG or RS232 links. Another advantage of Ethernet is the wide availability of local area network (LAN) connectivity, allowing for easy physical interfacing.

Ethernet uses a multiple access contention protocol, so we cannot rely on the full bandwidth being available continuously. We use 1,25 MB/s for our design (10% of the maximum). Before the achievable burst rate and PRF can be determined, the protocol must be selected.

Protocol selection

The data integrity requirements are not very stringent. The success of setup messages can easily be determined from the acknowledgements. As the return information is continually streamed to the waveform generator, a lost or corrupted message will cause a momentary glitch in the output (e.g. the returns in a certain pulse may be lost). The number of glitches can be reduced if the waveform generator only changes its output on successful reception of the new return information.

Three protocols were considered for the Ethernet link, and are summarised in Table 3.2. The payload field is the maximum number of bytes that can be sent without fragmenting the packet. The efficiency relates the size of the payload to the maximum raw Ethernet packet size [3, p. 17]). Efficiency is a concern because handling the layers in the protocol will take a significant amount of the waveform generator's processing time. Maximising the efficiency minimises the number of messages needed.

Another consideration is the link integrity — TCP/IP is the only one of the protocols that guarantees packet delivery. However, this requires a substantial amount of overhead. The real-time nature of the simulator means that the data is not critical. This fact allows the simulator to handle data loss in a graceful fashion: the last valid data received is used repeatedly, until newer information is available. In normal conditions, the high data rate on the link means that the output will only experience a momentary glitch. If the link integrity deteriorates, the fidelity of the simulated output will deteriorate similarly, with the number and duration of the glitches increasing.

Considering the link efficiency and integrity, TCP/IP is a poor solution. The final consideration is the ease of implementation. As a UDP/IP implementation is already available as part of the development kit, we conclude that UDP/IP is the protocol best suited to our application.

Limitations

For each burst the target descriptors for every pulse must be sent — this is done using the 1418 byte `WavGenScattererReturnListSnd` message (see Appendix C). For a 32 pulse burst, three messages are sent (11 pulses per message), thus $3 \times 1418 = 4254$ bytes of data is

Table 3.2: Comparison of various communication protocols.

Protocol	Payload size	Efficiency	Integrity	Implementation
Raw Ethernet	1500 bytes	100%	Not guaranteed	Requires additional work
TCP/IP	536 bytes	36%	Guaranteed	Already available
UDP/IP	1476 bytes	98%	Not guaranteed	Already available

required per burst. The clutter descriptor message, `WavGenClutterReturnListSnd`, requires 1294 bytes per burst. The total data per burst (excluding the message overhead), is 5548 bytes.

The maximum achievable burst rate is simple to calculate as the quotient of the available bandwidth and data per burst, multiplied by the protocol efficiency: $(1,25 \cdot 1024^2)/5548 \cdot 0,68 = 161 \text{ Hz}$ (where the factor 1024^2 converts from megabytes to bytes). With 32 pulses per burst the maximum (data rate limited) PRF will be $161/32 = 5,0 \text{ kHz}$. Higher PRFs can be achieved in patterns with fewer pulses per burst.

3.4 Summary

This chapter documented the system design at a relatively high level. The design approach followed standard engineering norms: the requirements were stated, a functional analysis was performed, followed by the conceptual design.

A type *II* architecture was selected for the system. The core of the simulator is implemented in the *Scenario* software package, with the majority of processing done on a PC, off-line. During real-time playback, the PC uses the current boresight angle to determine the scatterers in the beam and also generates the random clutter samples. Dedicated hardware is used for the time-critical pulse generation. The hardware consists of an FPGA development kit, supplemented with a custom DAC board.

Chapter 4

Software implementation

4.1 Introduction

Details of the software implementation are described in this chapter. The first step is to describe the methodology used for the implementation. Next, the core simulator software is analysed in detail. A brief discussion of the embedded software is given. Finally, we describe the graphical user interface and its implicit control function.

4.2 Implementation methodology

The RRS coding standards were used as a guideline for the implementation. This allowed for consistent layout and consistent variable and function naming, across all software packages. The standard advises the use of unit testing. Unit testing is a part of the Extreme Programming paradigm [31, pp. 77–80]. The traditional “code first, test later” approach is reversed. In order to implement a software unit, the process below is used.

1. Analyse the functions that the unit should perform.
2. Set up test cases to check the functionality (the desired output for each test case must be determined).
3. Code a set of automated tests that compare the actual output from all the unit’s methods with the desired output.
4. Implement one of the unit’s methods and re-run the tests. Repeat this until all the tests pass.
5. If additional functionality must be added, first modify the tests, then the software unit.

Software exists to simplify the testing process — Michael Feathers’ *CppUnit* [34] was used.

Unit testing has two major benefits — it forces the programmer to focus on functionality, and allows code to be verified simply. Automated unit tests are particularly important in large projects, where the effects of a code change is not always fully apparent (especially

with multiple programmers on the project). Running the tests will immediately indicate any problems.

The unit testing approach was used for the simulator’s core functionality, but not for the user interface or embedded code. The latter packages are very simple, so unit testing is unnecessary.

As a final note, the *CVS* [36] version control system was used in order to help track changes. The *Doxxygen* documentation system [88] was used to simplify documentation. *Doxxygen* generates documentation, in a variety of formats, directly from the source code.

4.3 Simulator core

The simulator’s core is implemented in the *Scenario* package — it includes the scenario, and the pre-processing and real-time engines (see Figure 3.2, p. 61).

4.3.1 Implementation decisions

A number of important implementation decisions were made prior to coding.

- All variables storing physical values must work in the base Système International (SI) units.
- Angular values must always be stored in radians.
- All classes containing information about the scenario should have streaming operators to simplify the loading, saving and displaying of their data.

4.3.2 Reference frame

In order to define the scenario, the first thing needed is a reference frame. We use a 3-axis Cartesian system, with its origin on the earth’s surface. The X-axis is positive to the East, the Y-axis positive to the North, and the Z-axis positive away from the earth’s centre. Cartesian co-ordinates are preferable to latitude, longitude and altitude co-ordinates, as they simplify the relative range and angle calculations.

Position in the Cartesian reference frame is handled by *clsXYZPosition*. For geographical co-ordinates the *clsLLAPosition* class was created. No DTED or other geographical data was needed for the prototype system, so the geographical co-ordinates were unnecessary. Thus the conversion functions between the two classes were not fully implemented. Besides positional information, the rate of positional change is also of interest. For this need, the *clsDynamics* class was implemented as a descendent of *clsXYZPosition*.

Just as important as spatial position, is temporal position. For this, *clsSimTime* was implemented. It includes comparison operators (as do the spatial position classes).

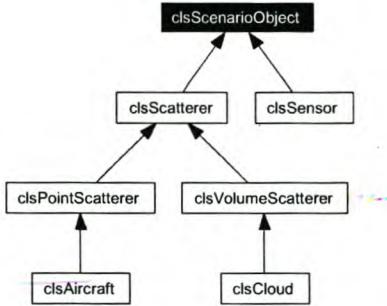


Figure 4.1: Class hierarchy for *clsScenarioObject*.

4.3.3 Scenario objects

A scenario can contain a number of different types of objects: planes, ships, the radar itself, etc. All of these objects share common attributes such as position and trajectories, but also have fundamental differences. This is a clear use for class inheritance. The class hierarchy is shown in Figure 4.1. We describe each object in the following sections.

clsScenarioObject

The base class includes a unique key for each object, the object's trajectory, its current position and functions to move it to another time. Another important function is the implementation of string-based attributes, taken from the University of Stellenbosch Software Defined Radio project [22, pp. 30–33]. The fact that the attributes are string-based, allows easy access from a user interface.

clsWayPoints

Each scenario object includes an instance of *clsWayPoints* in order to describe its trajectory. The way points are simply a list of positions (in time and space). Given a certain time, *clsWayPoints* interpolates the object's position between the way points, and calculates the correct velocity. Linear interpolation is used as a first order implementation. If the way points are not set up carefully, then there will be discontinuities in the object's dynamics. The software's current implementation does not aid the user in this process.

clsSensor

One of the most important classes in the scenario, *clsSensor* represents a radar sensor. It includes the waveform and antenna patterns, current boresight look angle, functions to implement the radar range equation, functions to apply the antenna pattern to pre-processed information, and other parameters such as the transmission frequency and power. The other classes required by *clsSensor* are clearly depicted in the collaboration diagram (Figure 4.2).

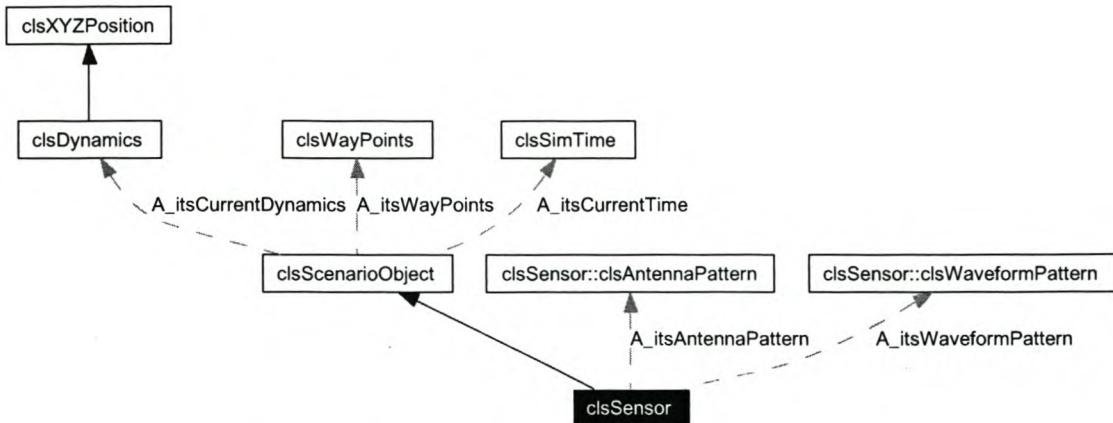


Figure 4.2: Collaboration diagram for `clsSensor`.

clsScatterer

The scatterer branch of the hierarchy contains all objects that could generate radar returns. There are functions for loading and saving the pre-processed information — each scatterer has a stream object (`clsPreProcStream`) to simplify the file input/output (I/O). The object can check whether it is in the sensor's beam — this is simple to perform, as the pre-processed information includes the locus of azimuth and elevation angles in which the target is visible to the sensor. The last function is one to calculate the object's current RCS. The RCS function is a pure virtual function which must be implemented by each of the leaf classes. The collaboration diagram is shown in Figure 4.3.

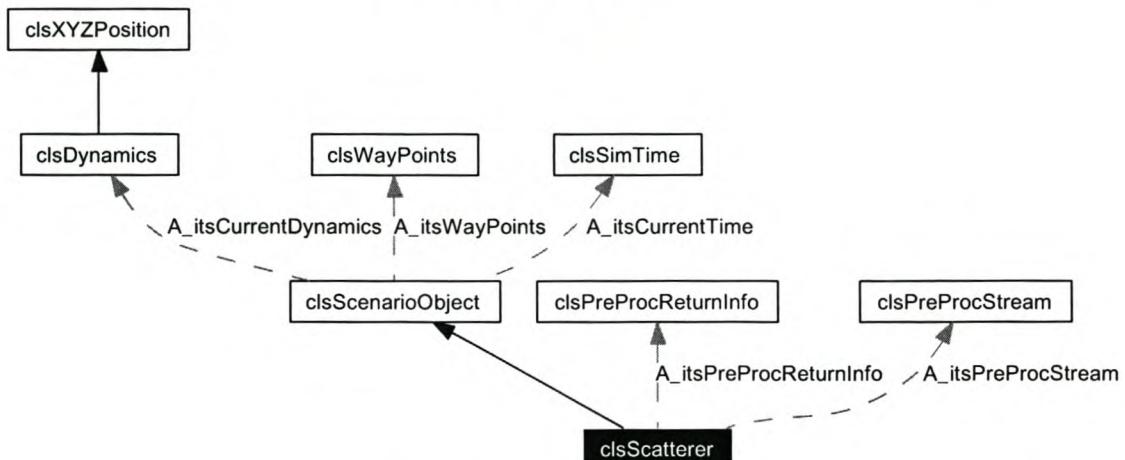


Figure 4.3: Collaboration diagram for `clsScatterer`.

clsPointScatterer

This is an abstraction layer that groups all the point targets (isotropic scatterers with no volume). No attributes or functions are added.

clsAircraft

An example of a point scatterer, objects of type *clsAircraft* implement the RCS function. The model for aircraft returns was selected in Section 2.5.1 (p. 32). The mean RCS must be provided by the user (the simple model in Appendix B, p. 163, could be used). Amplitude scintillation is generated using the Swerling fluctuations (see p. 28) — the user can select between any of the five types. The *clsAircraft* model does not model doppler scintillation or glint.

clsVolumeScatterer

This is also an abstraction layer, it groups all the scatterers that can have volume. In addition to isotropic scattering, these objects can attenuate the rays that pass through them. Attributes are added to define the object's extent, as well as a pure virtual function that calculates the amount of attenuation caused by the volume.

clsCloud

The *clsCloud* class is included as an example of a volume scatterer, but was not implemented. The reason for this was explained in Section 2.5.4 (p. 46).

4.3.4 Terrain

The terrain map (*clsTerrainMap*) is a grid of terrain blocks (*clsTerrainBlock*), we describe these two classes below.

clsTerrainMap

The terrain map is very simple, and publishes all of its attributes (there is no benefit to abstracting the access). Its attributes include the position of the top left corner of the map, the size of a block, the number of rows and columns, and a two-dimensional array of *clsTerrainBlock*. It has an initialisation function which sets the height of the blocks to follow to the curve of a spherical earth.

By default, and as per the system requirements, the map's maximum size is 200 km × 200 km. The default block size is 100 m × 100 m, to match the DTED level 1 specification (see Section 2.5.3).

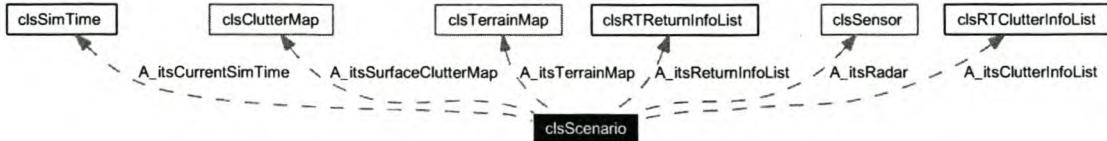


Figure 4.4: Truncated collaboration diagram for `clsScenario`.

clsTerrainBlock

As with the terrain map, `clsTerrainBlock` publishes all of its attributes. These include the type of terrain, and the Z-axis co-ordinate of the block. The surface permittivity and conductivity, surface roughness, and vegetation coefficient are needed to calculate the pattern propagation factor. Attributes relating to clutter include: wind speed, reflectivity, the mean scattering coefficient, and the shape parameter for the K-distribution. The sea clutter model was discussed on p. 39, and the ground clutter model on p. 44.

4.3.5 Scenario

The scenario is encapsulated in the `clsScenario` class — the simulator's main workhorse. A truncated collaboration diagram is shown in Figure 4.4. The scenario includes the current simulation time, the terrain map, and the radar sensor. The clutter map is described in Section 4.3.6, and the two lists in Section 4.3.7. Not shown are three lists containing the point scatterers, the volume scatterers and the test targets.

The point scatterer and volume scatterer lists benefit from the polymorphism of the `clsScenarioObject`. Instead of requiring a list for each type of vehicle and each type of volume scatterer, only two are needed. As point scatterers are processed differently to volume scatterers, we cannot make do with a single scatterer list.

Each `clsScenario` object must have its own project directory (set via an attribute) — this is where all the pre-processed data will be stored. In normal usage, there will only be one instance of a `clsScenario` object in existence.

The majority of the methods needed for the pre-processing and real-time engines are contained in `clsScenario`. The details are discussed in the following two sections.

4.3.6 Pre-processing

The pre-processed information is handled by three classes in particular; these are discussed first. Thereafter we detail the pre-processing engine's implementation, considering user interaction, surface scatterer processing, and finally point and volume scatterer processing.

Pre-processed information classes

clsPreProcReturnInfo. All the target returns (point and volume scatterers) are ultimately derived from the information stored by this class. A block of *clsPreProcReturnInfo* is stored for every scatterer, for every time step in the simulation.

clsPreProcReturnInfo is quite simple and publishes all of its attributes. An advantage of using a class rather than a C structure, is that streaming operators can be included. The return parameters stored include the range delay, phase angle, and doppler frequency shift. The return power (excluding the PPF) is also stored, along with the PPF terms — these are all needed by the real-time engine to determine the return’s amplitude. For in-beam checking, the range of visible angles are saved.

clsClutterMap. The clutter map is a sectorised representation of the clutter in the scenario — analogous to a plan position interface (PPI) display. The map is stored in a two-dimensional array. The number of sectors is calculated as the quotient of 2π and the azimuthal beamwidth. The number of range cells is 8192, as described in Section 3.3.6. The information for each clutter cell is contained in a *clsPreProcSurfaceClutterInfo* object.

The clutter map also includes a stream object (*clsPreProcStream*) to facilitate file I/O.

clsPreProcSurfaceClutterInfo. The *clsPreProcSurfaceClutterInfo* object stores the surface area of the clutter cell; information about which terrain block’s clutter parameters to use; and the locus of elevation angles in which the clutter cell is visible. Streaming operators are also implemented.

User interaction

Prior to pre-processing, the user must initialise the engine by calling *PPInitialise()*. This creates all the necessary files and ensures that all objects are at time zero. After initialisation, the user must continually call *PPProcessCurrentTimeStep()*, until the whole scenario has been pre-processed. A number of private functions are used to process the current time step, ending with *PPStepTime()*, which moves all the objects forward in time. The remaining methods are described below.

Surface scatterers

The surface clutter is processed in a relatively simplistic way, using the *PPSurfaceScatterers()* method. A number of simplifications are made. Firstly, the radar is static, so the surface clutter parameters are only generated once. Secondly, at a given range, the terrain’s height must be the same at all azimuth angles. Thirdly, there is no interpolation if the radar beam illuminates more than one clutter cell.

Ray tracing (discussed in the next section) is used to determine which terrain block lies closest to the centre of the current clutter cell. This block is then requested to calculate

its clutter parameters given the radar's current settings, the range and the grazing angle. These clutter parameters remain in the terrain map, rather than the clutter map. This is why *clsPreProcSurfaceClutterInfo* must contain the details of which terrain block to use. After processing is complete, both the clutter map and the terrain map are saved using *clsPreProcStream*.

Problems with this implementation include the following:

- Only a single transmission frequency is used for clutter generation, so frequency dependent models cannot be implemented.
- The sector resolution is very low and there is no interpolation, so the clutter will not change smoothly as the beam moves through sector boundaries.
- A clutter cell will generally include a number of terrain blocks, but only one is used — this reduces the fidelity of the clutter returns.

Point scatterers

The point scatterer information is calculated by determining the parameters below.

Environmental parameters. The top level functionality is implemented in *PPGetEnvironmentalParams()*. The range, azimuth and elevation angles from the radar to the point scatterer are determined by tracing a ray between the two using *PPRayTrace()*. The delay and phase shift are then calculated from the target's range, using eq. (2.20) and eq. (2.19), respectively.

As discussed in Sections 2.3.1–2.3.3, the current implementation of *PPRayTrace()* is trivial, assuming free space propagation, but can be modified to use more complex models if desired. The simple ray optics excludes refraction, so neither anomalous propagation nor retardation effects are modelled. The ray trace result also includes a field for atmospheric attenuation, however the calculation was not implemented. The path loss can be determined by adding the gaseous absorption losses to those caused by any volume scatterers intersected (i.e. losses due to weather effects).

Visible angles. It is straightforward to determine the range of angles in which a point scatterer is illuminated. *PPGetVisibleAngles()* simply adds half the beam width to the object's azimuth and elevation angle, returned by the ray tracing algorithm.

PPF terms. The pattern propagation factor requires a number of calculations, but is not particularly complicated. The general definition for the PPF was given in eq. (2.11) and is repeated here for convenience.

$$F_c = f_d + \rho D f_r e^{-j\gamma} , \quad (4.1)$$

where f_d and f_r are the antenna dependent parameters that can only be determined by the real-time engine. Expanding eq. (4.1), we get

$$\begin{aligned} F_c &= f_d + \rho D f_r [\cos(\gamma) - j \sin(\gamma)] \\ &= f_d + \rho D f_r \cos(\gamma) - j \rho D f_r \sin(\gamma) . \end{aligned} \quad (4.2)$$

Excluding f_d and f_r , the terms are 1, $\rho D \cos(\gamma)$, and $\rho D \sin(\gamma)$, where the first one is trivial and need not be saved. Despite our development here, the *PPGetPpfTerms()* method was only implemented trivially ($F_c = 1$), as explained in Section 2.4.2, p. 24. The implication is that the surface scattering multipath propagation effects are not modelled — an especially poor assumption for targets flying low over the sea.

Partial return power. The ubiquitous radar range equation, eq. (2.18), is implemented in *clsSensor*, taking the monostatic simplifications into account. It is termed “partial return power” because the PPF is not included. The *PPGetPartialReturnPower()* method simply requests this calculation from the radar sensor.

Doppler shift. Before the doppler shift can be determined, the scatterer’s radial velocity, relative to the radar must be calculated — this is the function of *PPGetRadialVelocity()*. Next, the *PPGetDopplerShift()* method implements eq. (2.21) to determine the frequency shift.

Save pre-processed information. Once all the calculations have been done, the pre-processed information can be saved — this is handled by *clsPreProcStream*.

Volume scatterers

Processing of the volume scatterers is essentially the same as the point scatterers, with a slight modification to the calculation of the visible angles. Pre-processing of the volume scatterers was outside the scope of the prototype implementation, and thus not included.

4.3.7 Real-time operation

As with the pre-processing, a number of classes are used to store the real-time information. These are discussed, and then the real-time engine’s implementation is exposed.

Real-time information classes

clsRTReturnInfo. All the information needed for a target descriptor is contained in *clsRTReturnInfo*. Specifically, the amplitude and phase for each channel, as well as the delay and doppler shift. The class implements a comparison operator, based on the target delay — this is required by *clsRTReturnInfoList*.

clsRTReturnInfoList. The information for all the returns received in the current PRI is stored in *clsRTReturnInfoList*. It provides a level of abstraction for the storage method. The implementation uses a Standard Template Library (STL) *std::multiset*, of type *clsRTReturnInfo*. The *std::multiset* automatically sorts the returns by increasing delay (a requirement for the waveform generator).

clsRTCutterInfo. The real-time clutter information contains the amplitude of the return for each channel, and the range cell in which it belongs. As with *clsRTReturnInfo*, a range-based comparison operator is implemented.

clsRTCutterInfoList. This list is similar to *clsRTReturnInfoList*, however it includes additional attributes. These are the minimum and maximum ranges of the visible clutter. The ranges are used to speed up the packing of the `WavGenClutterReturnListSnd` message.

User interaction

The user must call the *RTInitialise()* method before starting real-time playback. This opens the necessary files for reading, loads the pre-processed clutter, and moves all objects to time zero. Once initialised, the simulation is stepped one PRI using the top level method *RT-ProcessCurrentTimeStep()*. The user (i.e. the GUI's control unit) must ensure that all PRIs in the current burst are stepped through after receiving a `WavGenBurstStartSnd` message.

Surface clutter

The real-time clutter processing is only performed when the clutter list is empty. All the range cells in the sector currently being viewed by the radar are examined to see if any are in the beam. If so, a random RCS sample is generated according the terrain-defined PDF. The radar range equation is then used to determine the return power (as with *PPGetPartialReturnPower()*). Next, the radar sensor applies its antenna patterns — in this case the PPF is not considered, and the average amplitude of the each channel's antenna pattern is used. The clutter amplitude is then stored in the clutter list.

Target returns

Scenario-based targets. After the clutter processing, it is the turn of the point scatterers. For each scatterer, the current time step's pre-processed information is loaded. If the scatterer is in the beam, then the antenna pattern is applied by *clsSensor* (the PPF would be added here). The final return amplitude and phase for each channel is then stored in the target return list.

The antenna patterns are stored in a number of matrices. The rows represent the azimuth angle, and the columns the elevation angle (offset from boresight). The step size is one milliradian. The matrix elements store the normalised amplitude and the phase shift,

measured at the given angles. When applying the antenna patterns, the information in the element nearest the desired angle is used. I.e. there is no interpolation of the patterns. This simplistic implementation does affect the tracking performance of simulated targets, since the radar will measure relatively large jumps in the position of the target as the beam moves relative to the target. The tracking tests in Section 6.5 (p. 130) highlight the problem.

Test targets. Test targets are provided as a debugging tool, allowing returns to be generated in an arbitrary range and doppler cell, independent of the current scenario. The targets exist only in the real-time engine and, as they are not part of the scenario, have no pre-processed information. The parameters that can be set for test targets are: range, amplitude, phase and doppler frequency shift.

The implementation assumes that the test targets are static and always in the radar’s beam. The antenna patterns and range equation are not applied — the return has the same amplitude and phase for all three channels.

Theoretically, a static target can have no doppler shift, but by varying the return’s phase from pulse-to-pulse, an apparent doppler shift can be simulated. The phase increment $\Delta\phi$ for each pulse can be calculated. For a time step of Δt seconds, the phase change is simply the number of revolutions that the wave goes through due to the doppler frequency f_D , i.e.

$$\Delta\phi = 2\pi f_D \Delta t \text{ [rad].} \quad (4.3)$$

Volume scatterer returns

The volume scatterers can be handled in the same way as the point scatterers. As mentioned in the discussion of the pre-processing engine, the volume scatterers are outside the prototype’s scope, so the implementation was not coded.

4.4 Embedded software

The embedded software was implemented in C, and thus the code partitioning is explained using files, rather than classes. An include dependency graph is shown in Figure 4.5.

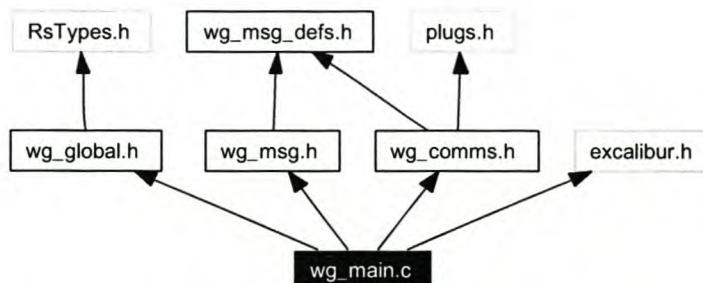


Figure 4.5: Include dependency graph for waveform generator main program.

A description of the software units implemented in each of the files, is given below.

wg_main.c The main program is implemented here. It performs the initialisation: the *Nios* CPU is set up; interrupt handlers installed; and the communications initialisation routine called.

RsTypes.h Type definitions used by RRS.

wg_global.h All the global constant and variable declarations.

wg_msg_defs.h The messaging declarations, including enumerations and structures.

wg_msg.h Along with wg_msg.c, the expected messages are defined, and the message handlers implemented. The setup message handlers either modify the CPU’s parallel I/O (PIO) outputs, or return their current state. The return list handlers simply copy the received data into the appropriate target or clutter generator buffer.

plugs.h The Plugs Ethernet library is part of the *Nios* software development kit (SDK) provided by Altera. It contains a TCP/IP stack implementation (UDP is also included).

wg_comms.h The communications engine is implemented in this file and wg_comms.c. It includes the communications initialisation and shut-down routines, as well as functions to verify the packets and messages received.

excalibur.h This header file is generated by the *SOPC Builder* for the specific *Nios* system (more details of the *SOPC Builder* are given in Section 5.3.5). It includes all the constants and structures needed to access the *Nios*’s peripherals.

The embedded software’s main task is processing, and then replying to, the messages received from the PC. Invariably, the processing consists of copying the message data to a memory location in the waveform generator. One other important task is to send a message to the PC, whenever the RSP starts a new burst. The implementation is straightforward and will not be discussed further.

4.5 Graphical user interface

The GUI package has two important functions: interfacing with the user, and controlling the PC side of the simulator. The latter function includes handling all messages received and sent via the communications interface.

The graphical user interface was implemented using Borland’s *C++ Builder 5.0* [14]. This program was chosen since the Borland development tools are the standard at RRS. A disadvantage of *C++ Builder 5.0* is that the GUI is not portable — it will only run in a 32-bit Windows environment. As the control unit is embedded in the GUI, the portability of the simulator is reduced. Portability is not a system requirement (the simulator need only run under Windows XP), so this is not a major concern.

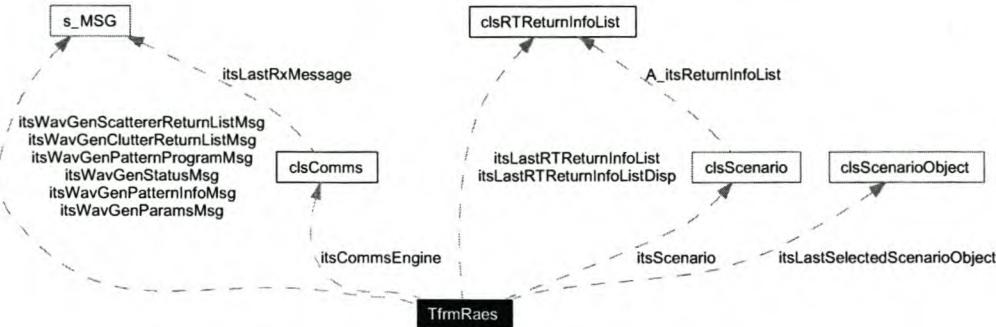


Figure 4.6: Truncated collaboration diagram for the graphical user interface.

4.5.1 Communications engine

The *PcComms* package is realised in the *clsComms* class. The communications engine does not provide much processing, but serves as a layer of abstraction between the messages and the communications medium. It implements two logical links, a UDP/IP link to the waveform generator and a TELNET link to the SDP. Both links are implemented using components from the Internet Direct (a.k.a. Indy) component suite [42].

The UDP/IP link uses a UDP server (*TIdUDPServer*), that provides a callback function whenever it receives a UDP packet. *clsComms* automatically executes a user defined function whenever this event occurs. A function is also provided to send a message to the UDP client.

A TELNET client, *TIdTelnet*, is used for the interface to the SDP. The data received over the TELNET link is parsed by *clsComms*, and the current boresight angle extracted. As with the UDP link, a user defined event is executed whenever a new angle is available.

4.5.2 Class structure

The GUI is implemented using a single class (*TfrmRaes*), a descendent of Borland's Visual Component Library (VCL) *TForm* class. The collaboration diagram is shown in Figure 4.6.

The basic message structure is contained in *s_MSG* — the GUI has an *s_MSG* structure for each message that it needs to handle. The two most important objects are the instance of *clsCommsEngine* and the instance of *clsScenario*. There are also a couple of real-time information lists, used for packing the *WavGenScattererReturnListSnd* message; and a *clsScenarioObject* object to store the item clicked by the user.

4.5.3 Logging

In order to facilitate debugging, some basic logging functions were implemented. These functions simply write the given log messages to a file.

4.5.4 Modes of operation

The GUI has three modes of operation:

Setup scenario. Allows the user to set up the scenario, e.g. object attributes can be modified. Playback moves the objects, but does no pre-processing.

Pre-processing. Allows the user to pre-process the scenario. While pre-processing, the objects will move along their trajectories.

Real-time. If the scenario has been pre-processed, then real-time mode can be started. This sets up the waveform generator, and then streams the real-time information to it. Objects will move along their trajectories in real-time.

In each mode, the user can play, pause or stop playback.

4.5.5 Message handling

The event fired by *clsComms* on reception of a UDP packet is linked to the GUI's message handler — *OnMessageReceived()*. The message status is checked and any errors indicated to the user. *OnMessageReceived()* examines the message identifier to determine which message was received and then calls the appropriate message-specific handler. The setup messages require trivial processing, so only the burst-start message will be discussed.

If the burst-start synchronisation message is received during real-time playback, the appropriate action is to step the scenario through all the pulses in the next burst. After each pulse, the target return information is packed into the *WavGenScattererReturnListSnd* message (it is sent as soon as it is full, or the end of the burst is reached). Finally the next block of clutter information is packed and sent using *WavGenClutterReturnListSnd*.

The processing in the burst-start message handler is the most time critical part of the PC-based code. It must be finished before the next burst is finished, or else the simulation will fall behind real time. The software implementation is not allowed to skip bursts, so in the case where the PC is too slow, there is no way to catch up again. Simulation time will just lag further and further behind real-time. Skipping bursts is a poor solution since the simulated returns would then only change every few bursts. In addition, it would only work in a single burst waveform pattern (otherwise the number of pulses in successive bursts would be different). Better solutions would be to optimise the code, use a faster PC, or increase the PRI for each pulse (i.e. a longer time between bursts).

4.5.6 Current boresight angle

The *OnBoresightAngleReceived()* event handler is linked to the communication engine. Its processing is trivial — it simply updates the radar's boresight angle using a function provided by *clsScenario*.

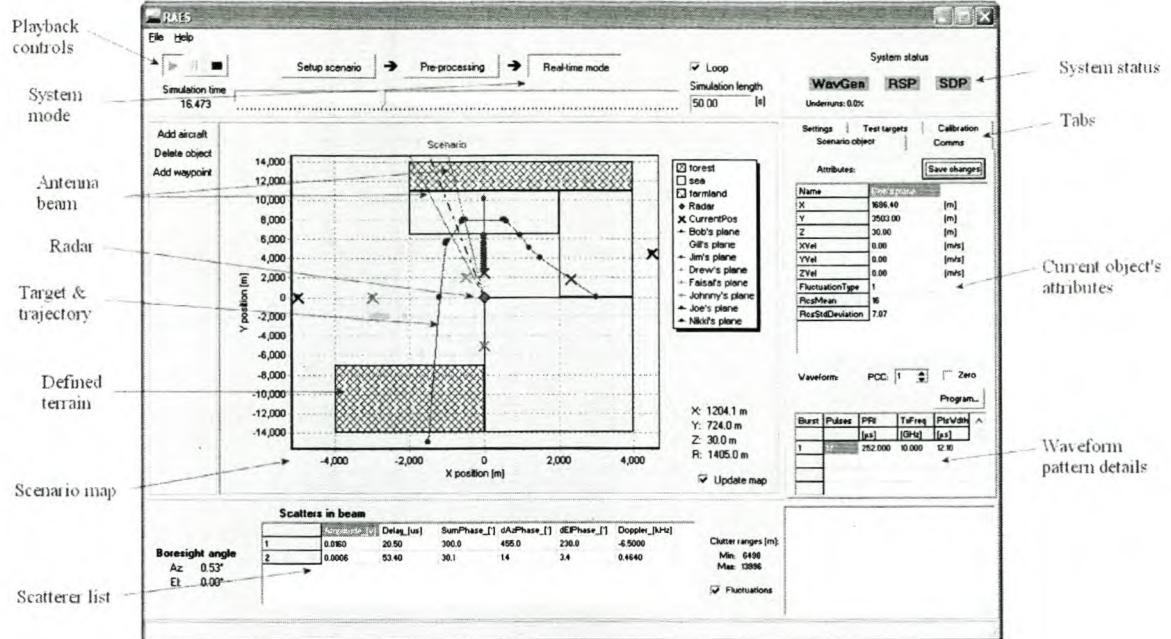


Figure 4.7: Screenshot of the graphical user interface.

4.5.7 GUI description

We first discuss the general layout of the user interface, and then look at the settings available to the user.

Layout

A screenshot of the graphical user interface is shown in Figure 4.7. The more cryptic annotations are described below.

Antenna beam. The “V” centred on the radar indicates the current volume that can be viewed by the radar.

Target and trajectory. The current position of each target is indicated by an “X”, way points are dots, and the trajectory a line joining the dots. Only three of the objects in this scenario are moving.

Scenario map. It provides a top down view of the scenario, with the radar in the middle.

Scatterer list. During scenario setup and pre-processing, this list shows all the targets in the scenario. During real-time playback, only the scatterers in the beam, along with their real-time return parameters, are shown.

System status. The status of the three modules external to the PC are shown here, using colour coding. Grey indicates unknown status; red, an error; yellow, degraded status; and green means that there are no problems.

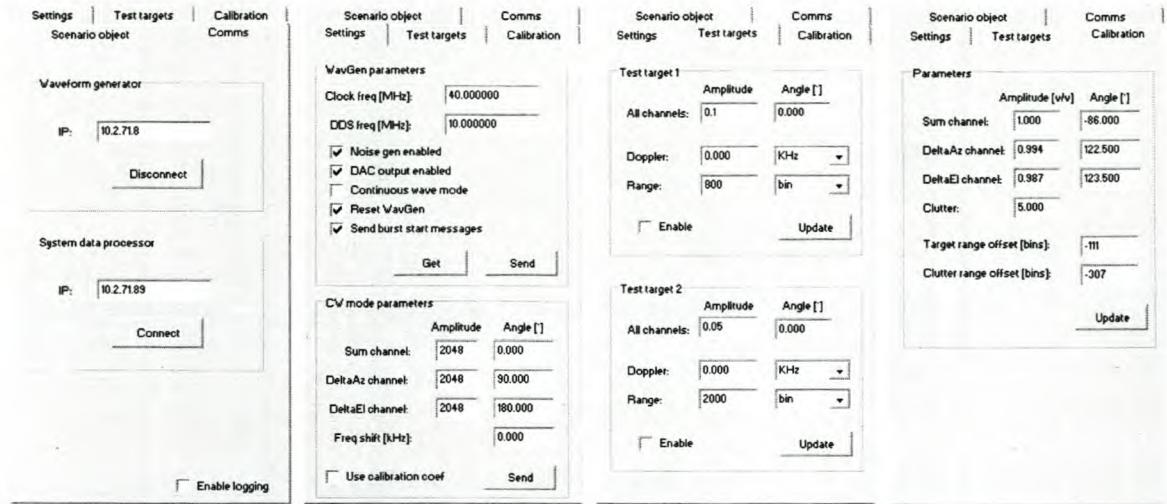


Figure 4.8: Other tabs available in the graphical user interface.

Tabs. This is where most of the simulator’s settings are changed — the tabs are discussed in detail in the following section.

Current object’s attributes. The attributes of the last object selected by the user are displayed here. All string-based attributes can be edited (this excludes the object’s position).

Waveform pattern details. The settings of the current waveform pattern are displayed here. The *Program...* button can be used to download a new pulse code to the waveform generator.

Settings tabs

Figure 4.8 shows the tabs that were not visible in the previous screenshot. We describe the available settings in the paragraphs that follow.

Comms. This allows the IP addresses of the waveform generator and SDP to be set up, and the connections opened or closed. There is also a check box that can be used to enable the logging.

Settings. The top group box can be used to view or modify the waveform generator’s parameters. The nominal DDS frequency (the IF) can be changed — note that the clock frequency is only used to calculate the correct phase increment. A number of other options can be enabled or disabled: clutter output (noise generator), DAC output, continuous wave mode, the waveform generator’s reset state, and the sending of burst start messages.

Continuous wave mode is a debugging and testing feature. The lower group box can be used to set up the continuous wave parameters, which are self-explanatory.

Test targets. The user interface only allows two test targets to be set up, although the *Scenario* package does not have a hard limit. Doppler can be set as either a frequency or a speed (calculated at 10 GHz). Units for the range are range bins, microseconds, or metres.

Calibration. The calibration parameter tab allows the modification of the system’s calibration values. The amplitude factors are multiplicative, while the phase and range offsets are added to the nominal values. The calibration parameters have a number of functions:

- Differences in the gain and phase of the three analogue channels can be corrected.
- The phase measurements for the antenna patterns includes a bias due to the test setup — this must also be calibrated out, if the RSP is to measure the target’s position in the beam correctly.
- The target and clutter range offsets allow the delays incurred by the waveform generator’s latency to be calibrated out.
- The calibration parameters can be used as a debugging tool — target parameters can be adjusted on the fly (i.e. without re-processing the scenario).

4.6 Summary

Defining a methodology prior to coding allowed for consistent system wide implementation. The unit testing approach used for the simulator’s core functionality helped improve the quality, and most importantly the testability, of the code.

The *Scenario* package implementation was more concerned with establishing a generic framework, than with implementing detailed target and environmental models. An indication was given as to how more complicated models could be included.

The embedded software is conceptually simple, which led to a straightforward implementation. Only a brief overview was given.

The *GUI* package linked the communications and user interfaces to the simulator’s core software. Event handlers were used to manage the information received on the communications interface, allowing for an uncomplicated implementation. A basic description of the functionality provided by the GUI was also given.

Chapter 5

Hardware implementation

5.1 Introduction

This chapter is divided into two parts, the first discusses the hardware used, including the custom DAC board. The second part, by far the larger, examines the FPGA-based firmware. As with the software, the implementation methodology is presented prior to the implementation details.

5.2 Hardware

The external hardware consists of the *Nios Development Board, Stratix Edition* from Altera, and a custom DAC board. A picture is shown in Figure 5.1: the DAC board is in the bottom right corner, with analogue output via the three rightmost SMA connectors. The following sections describe these two pieces of hardware.

5.2.1 Development board

The FPGA on the development board is the smallest device in the *Stratix* family (a 1S10), but nevertheless still provides enough logic (10 570 logic elements) and RAM (112 kB).

For communications and debugging, the board has two RS232 ports, a JTAG port, a Mictor logic analyser port, and a 100 Mbps Ethernet port. On-board memory includes 1 MB of static random access memory (SRAM), 16 MB of synchronous dynamic RAM and an 8 MB flash chip. There is also a configuration device which loads the user or factory configuration from the flash chip. There are two prototype headers each providing 40 I/Os, clock and power signals. Information can be conveyed using an LCD display, 7-segment displays, and the inevitable LEDs. User input can be provided using the four push buttons. Extensive detail can be found in [2].

The FPGA board fast tracks development by providing a reference design capable of accessing all the board's peripherals. It includes all the necessary interface logic and software. Interfacing is facilitated by Altera's *SOPC Builder* — described in Section 5.3.5.



Figure 5.1: *FPGA development board, with custom DAC prototype board (bottom right).*

The development kit is comprehensively documented and includes a number of design examples, besides the reference design, this reduced the development time significantly.

5.2.2 DAC board

As has been mentioned before, the development board has no analogue outputs, thus necessitating the DAC board. *Protel 99 SE*, developed by Protel [69], was used for the schematic editing and printed circuit board (PCB) layout.

As stated in the project objectives (Section 1.3), the fidelity of the prototype system is not critical, so a simple two layer board was used. In addition, the digital and analogue ground planes were not separated. Substantial decoupling was provided, in line with the DAC manufacturer's application notes.

The PCB runs at 40 MHz, which does require high speed layout principles to be considered.

- The most important is clock line termination, which minimises the reflections and thus reduces the clock glitches. A 50Ω parallel resistor was placed at the end of the clock line.

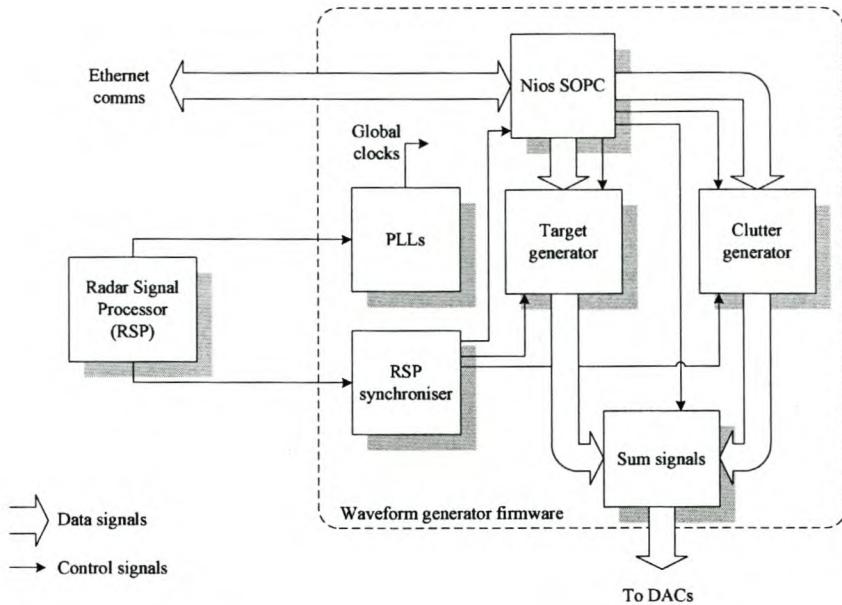


Figure 5.2: Block diagram of the waveform generator firmware.

- Component placement was kept as compact as possible.
- Decoupling capacitors were placed as close to the device pins as possible (most were placed directly underneath the DACs).
- A ground fill was used to cover unused areas of the board — this helps to reduce the coupling between signal lines.

It is noted that serial resistors on the data lines (especially the least significant bits, which change more often) would help reduce glitches in the data. The prototype board performed adequately without these resistors.

The layout is shown in Figure C.2, on p. 175 of Appendix C. The prototype PCB was manufactured at the University of Stellenbosch, and the components soldered on by hand.

5.3 Firmware

The waveform generator's firmware was developing using *Quartus II 4.0*, from Altera. This includes the *SOPC Builder 4.0* program. A block diagram is shown in Figure 5.2 — the implementation will be detailed in the following sections, however we provide a brief overview here.

- The control unit and communications interface is implemented using a Nios System on Programmable Chip (SOPC).
- The phase-locked loops (PLLs) generate clock signals based on the RSP's clock.

- The RSP synchroniser keeps track of the radar's waveform pattern.
- The target and clutter generators create the digital data that represents the simulated returns. These data are summed and then fed to the DAC board.

5.3.1 Implementation methodology

As with the software implementation, it is important to follow a design methodology throughout the implementation. The principal decisions are listed below.

- Implementation proceeds in a bottom up fashion.
- Simulations are run on each module to check the basic functionality, as far as is feasible.
- Parametrisable modules (in terms of bus widths or memory depths) are developed where possible.
- Block diagram design, using Altera's parametrisable functions (*Megafunctions*), is the preferred approach. This speeds the implementation and makes the design more easily understandable. Very High Speed Integrated Circuit Hardware Design Language (VHDL) is only used for special cases, such as state machines.
- Latency is not a major concern as it can be accounted for in software by offsetting the range delay. It does however, increase the simulator's minimum range.

5.3.2 RSP synchroniser

Synchronisation to the RSP is done using two modules: the pattern synchroniser and the pattern counter. The interfacing of the two blocks can be seen in the return generator's top level block diagram (Figure C.4 in Appendix C).

Pattern synchroniser

The pattern synchroniser uses a state machine implementation and is coded in VHDL. The state diagram is shown in Figure 5.3. Note that the RESET signal has the highest priority. The machine starts in the *Idle* state, and stays there until the PULSESTART signal is asserted (the signal is only asserted for one clock cycle). After this event, the machine moves to the *Counting* state where the number of clock cycles in the current PRI are counted. If the PRI is longer than the maximum PRI expected, then the *Pattern start* state is entered. For PRIs shorter than the maximum, the next PULSESTART assertion will reset the counter, so that the new PRI's length can be measured (simple logic, rather than the state machine itself, implements the reset).

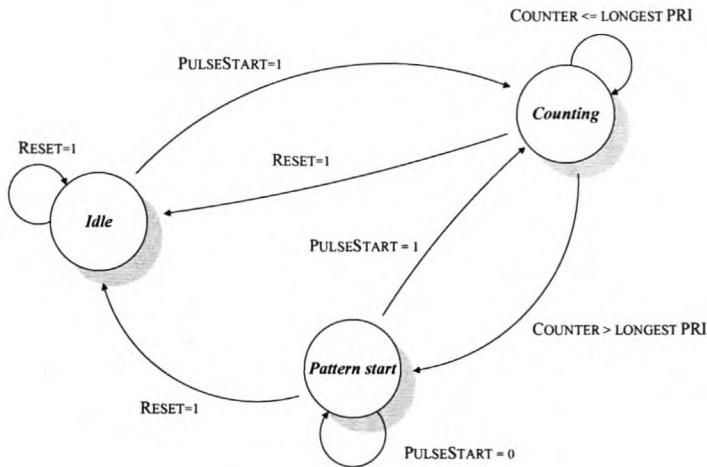


Figure 5.3: State diagram for the RSP pattern synchroniser.

Pattern counter

As can be seen in Figure C.4, the pattern counter's inputs include the number of pulses for each burst, and the pulse code length (i.e. the pulse width) for each burst. Using the PATTERNSTART and PULSESTART signals, it is simple to keep track of the current burst number. During each burst, the pattern counter outputs the current burst number, pulse number and pulse code length.

The implementation consists of a number of multiplexers and counters, and proved to be simplest to realise using VHDL.

5.3.3 Target generation

As is evident from the design chapter, the target generator is the most complicated part of the waveform generators's firmware. A functional block diagram was given in Figure 3.8 (p. 68). Our discussion of the implementation starts with the burst buffers, then looks at the logic needed to load the returns, and finally, the DDS engine is explained. The *Quartus* block diagram can be found in Appendix C, Figure C.5.

Buffering

A single block of dual port RAM is used for the two burst buffers. Using dual port RAM simplifies the implementation as the address bus does not need to be multiplexed. In order to implement the ping-pong buffering scheme, the RAM is divided in half. A T-flip flop is used to toggle the most significant bit of the RAM address lines. The generator will only switch buffers at the start of a burst if the data in the new buffer is valid. The embedded software asserts the BURSTBUFFERVALID signal once the new target descriptors have been successfully written (see Section 3.3.5).

The RAM has a 32-bit input data bus and a 128-bit output data bus. The former matches the Nios CPU’s data path width, while the latter matches the width of a target descriptor. A total of 8 kB of RAM is used to store the 512 target descriptors needed for two bursts. The memory map was designed to allow for straightforward address decoding, as shown in Table 5.1.

Table 5.1: *Burst buffer address decoding*

Address line bits	Description
Addr[8]	Which of the two burst buffers to use.
Addr[7..3]	The pulse number (up to 32 pulses).
Addr[2..0]	Up to eight target descriptors.

The write addressing, including byte enables, is generated by the Nios CPU’s interface logic. The read addresses are generated by concatenating the current pulse number with the current target descriptor number. The latter is provided by the *ReturnGenCtrl* module (discussed in the next section).

Loading returns

Three modules are required to load returns into the DDS engine: one *ReturnGenCtrl* module, and two *ReturnGenDelay* modules (one per pipeline).

ReturnGenCtrl. The *ReturnGenCtrl* block requests the next target descriptor from the RAM every time that a pipeline is free. It also controls the “load enable” signals for the two *ReturnGenDelay* modules, ensuring that only one is loaded at a time.

The *ReturnGenCtrl* block is implemented in VHDL, as it requires a reasonably complex interaction of flags and counters.

ReturnGenDelay. Each *ReturnGenDelay* module is associated with a DDS pipeline. The module’s purpose is to present a target descriptor to its DDS pipeline at the correct time.

The implementation was coded in VHDL, and uses a state machine approach. Figure 5.4 depicts the state diagram. Note that the RESET signal has the highest priority. The unconnected arrow in the figure indicates that the machine will return to the *Idle* state whenever the condition is true. The CWMODE signal refers to continuous wave (CW) mode.

The pipeline starts in the *Idle* state. In normal mode (i.e. CWMODE=0), the LOAD signal enables the machine to move to the *Waiting* state. Simultaneous to this transition, the input target descriptor is sampled.

While waiting (and also when idle), the DDS pipeline is held in a reset state. The delay module remains in the *Waiting* state until the range delay matches the number of ticks since

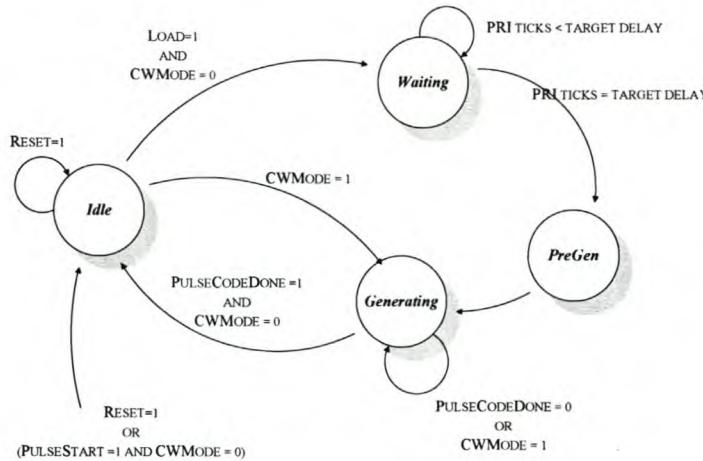


Figure 5.4: State diagram for the *ReturnGenDelay* module.

the start of the PRI. When the delay is over, the *PreGen* state is entered. This state is required so that the DDS pipeline can be released from reset one cycle before it must sample its input target descriptor.

Finally, the state machine moves to the *Generating* state. It remains here until the DDS pipeline has played out all the pulse code samples, or until the start of a new PRI.

For CW mode, the *ReturnGenDelay* module simply ignores the delay, going directly to the *Generating* state. Thus the first two target descriptors must contain the CW mode settings, as they will be loaded into the DDS pipelines.

DDS engine

The DDS engine implements two DDS pipelines. The main reason for combining two pipelines into one module was because the pattern phase LUT was implemented using dual port RAM. The engine thus consists of one pattern LUT, two phase accumulators, two CORDIC sine blocks and a number of adders and multipliers. The block diagram is shown in Figure C.6, in Appendix C.

Simulation. Prior to implementation, the entire DDS pipeline was simulated in *MATLAB* (from the Mathworks [83]). The simulation used fixed point values for all calculations, and included the CORDIC sine functions. The experiment was to generate sine waves (512 samples each) at a number of doppler shifted frequencies, in steps of 1 Hz. The output frequencies were measured and the average step size, maximum error and root-mean square (RMS) error calculated.

With only 512 samples, simple FFT peak picking could not provide accurate enough frequency measurement, so a more sophisticated approach was applied. Measurement was performed by fitting a third order auto-regressive (AR) model to the output sine wave. The modified covariance forward-backward method was used to estimate the AR parameters. For

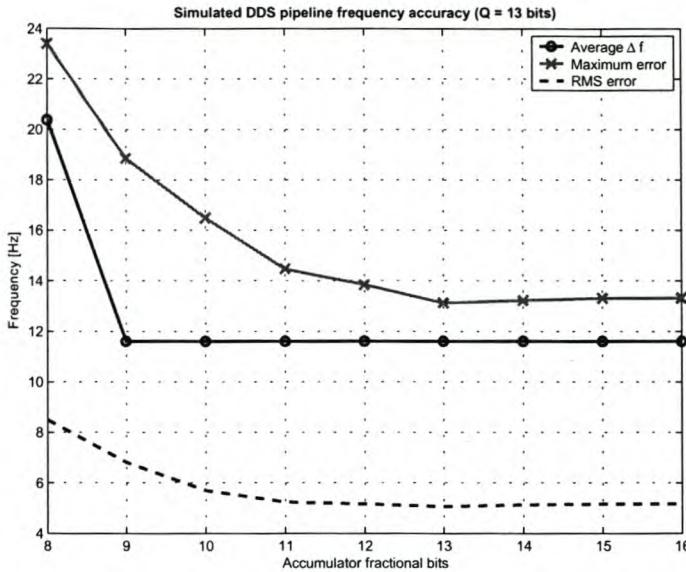


Figure 5.5: Measured frequency accuracy for the DDS pipeline, simulated in MATLAB.

real-valued sine waves, the AR process will always have two conjugate poles near the unit circle. These poles were used to calculate the sine wave frequency.

Figure 5.5 shows the results for various accumulator lengths. The number of integer bits Q , was kept constant at the designed 13 bits. The original design required 10 fractional bits, for a total accumulator width of $P = 23$ bits.

The best frequency resolution can be obtained with at least 9 fractional bits. The measured value was $\Delta f = 11.61$ Hz — 22% larger than the 9.54 Hz predicted by the theory. The deviation is attributed to the imperfect output from the CORDIC sine functions (this will be explained shortly, when their implementation is discussed).

From the graph we observe that frequency resolution is not the only factor affected by the number of fractional accumulator bits. The maximum and RMS errors continue to benefit from a wider accumulator, up to about 13 bits. It was thus decided that the phase accumulator should be widened to $P = 26$ bits. The only implementation implication of this decision is that the phase increment values will need to be shifted left by three bits.

Phase accumulators. The phase accumulator blocks perform two functions. The main function is the obvious summing of phase angles. There are three terms that are common to all channels: the nominal phase increment (to generate the carrier), the doppler shift phase increment, and the waveform pattern phase. In CW mode, the pattern phase is ignored. The secondary function is to generate the pattern phase LUT addresses, and assert a signal when the whole pulse code has been played out.

The phase accumulator was implemented as a block diagram.

Pattern LUT. The pattern LUT is a block of dual port RAM, that can be accessed by either of the pipelines. The RAM can store up to four different pulse codes — one for each burst. The maximum length for the simulator’s pulse codes is 512 samples. One of the input ports is shared by the CPU to allow different pulse codes to be programmed.

CORDIC sine. Time division multiplexing (TDM) is used to allow the sine values for all three channels to be calculated by the same CORDIC sine block. The CORDIC sine function is clocked at 120 MHz, three times the normal clock rate. This allows all three channels to be handled in the same 40 MHz cycle.

The sine function uses a parallel, online CORDIC processor. In other words, the CORDIC iterations are unrolled to create a pipelined implementation, with all the data bits handled in parallel. In general $n + 1$ iterations are needed for an n -bit CORDIC [28]. However, we only use a 12 CORDIC pipeline to calculate the 12-bit result. The reason being that 12 cycles fits into the TDM scheme, while 13 does not. This results in a slight error — simulation showed that skipping the 13th iteration increases RMS error (over one revolution) by 7.33%.

Another source of error is overflow in the CORDIC’s fixed point calculations — widening the internal data path helps minimise this. For an n -bit CORDIC, the recommendation is to use $\ln(n)$ guard bits [28]. For a 12-bit CORDIC, $\ln(12) = 2.48$ bits are required. We used two guard bits, even though the accuracy could be improved marginally by using three. The output accuracy is already reduced by using one too few iterations, not to mention the noise introduced by the DAC board, so the slight loss here is not critical.

Multipliers and adders. The *Stratix* contains dedicated multipliers as part of DSP blocks — these multipliers were used. The adders were implemented in normal logic.

5.3.4 Clutter generation

The block diagram approach was used to implement the clutter generator (see Figure C.7). The most important part of the generator is a 64 kB block of dual port RAM. The ping-pong access scheme is implemented in a similar fashion to the target generator’s burst buffer. The lower bits of the read address are taken directly from the pattern counter’s PRI cycle count output. Writing to the RAM is handled by Nios CPU.

In order to allow the clutter generator to be disabled, the output can be switched between the RAM output or a grounded signal. Output can either be disabled completely, or the minimum and maximum range inputs used to limit clutter generation to a certain region only.

5.3.5 Nios SOPC

SOPC Builder

Altera's *SOPC Builder* is the tool used to generate the system module that contains the Nios CPU. Besides the CPU, the system module also includes all the interface logic for accessing the boards peripherals. *SOPC Builder* allows components to be added, removed and modified, with a focus on the bus interfacing between them. A wizard is available which simplifies the addition of user designed logic. For more details see [4].

Once the system is designed, *SOPC Builder* generates an HDL description of the system that can be compiled by *Quartus*. Besides the hardware description, a software development kit (SDK) is also generated for the user's system. This is the origin of the plugs.h and excalibur.h files used in the embedded software (Section 4.4).

CPU system module

The implementation of the CPU system module required a few minor changes to Altera's reference design. The SDRAM is unused and was disabled. A number of output ports were added for the return generator and DAC control signals. For synchronisation, interrupt generating inputs were created for the pulse-start and burst-start signals.

The waveform pattern LUT, and the target and clutter buffers required the addition of interfaces to custom logic — a trivial process with the *SOPC Builder*. The system generation automatically takes care of the data and address buses, write signal timing and even arbitration. In software, data simply needs to be copied to the address of the desired buffer.

5.3.6 Timing

The required clock signals are generated using two PLLs. The first PLL is used to synthesise a 40 MHz clock from the RSP's 10 MHz signal. A limitation of the *Stratix* 1S10 device prevents generation of a 120 MHz clock from the same PLL. Thus the need for the second PLL, which uses the first PLL's output to create another 40 MHz clock and the 120 MHz clock. The second PLL's outputs drive the waveform generator. The interconnections can be seen in the waveform generator's top level block diagram — Figure C.3.

5.3.7 Top level

The top level block diagram was based on Altera's reference design, which provided all the necessary pin assignments. As can be expected, the top level file is used to set up all the connections between the Nios CPU, the return generator, the PLLs and a number of I/O pins. Figure C.3 shows the layout.

5.4 Summary

The majority of this chapter was concerned with the firmware implementation. As with the software, the implementation methodology was defined early on — a bottom-up approach was chosen. The development kit's comprehensive reference design and documentation speeded up the process.

In addition to the firmware development, some custom hardware was manufactured. For the prototype DAC PCB, simplicity was preferred to signal fidelity.

Chapter 6

System evaluation

6.1 Introduction

This chapter analyses the performance of the simulator — the majority of the tests are conducted with the simulator connected to a real radar system. Before delving into the tests and results, the test equipment is discussed. Thereafter the simulator is evaluated in three different ways: firstly, the returns from a single target are analysed throughout the radar's processing chain. Secondly, the clutter generation is tested. The chapter culminates with a number of closed-loop tracking experiments using simulated targets in the presence of clutter.

6.2 Test equipment

The tests were performed using the following equipment

- A Hewlett Packard HP54501A 100 MHz oscilloscope.
- A Reutech Radar Systems RTS6400 tracking radar.
- The simulator software (*RAES*) was run on a PC with a 2 GHz AMD Athlon XP processor, and 512 MB of RAM .

For the majority of the tests, only the RTS6400's radar signal processor was needed. Using a custom data acquisition device known as the Data Capture Module (DCM), the RSP's output can viewed and logged via a PC. The DCM's user interface program is simply known as the DCM software.

In order to understand the RSP's output, its processing chain is briefly described (Figure 6.1 shows a simplified version). The IFS performs quadrature demodulation to generate baseband I and Q samples, used for coherent processing. The DPC compresses the return pulses using a matched filter. Doppler information is extracted by the DFM, which effectively performs an FFT on the compressed samples, for every range cell. As can be seen from

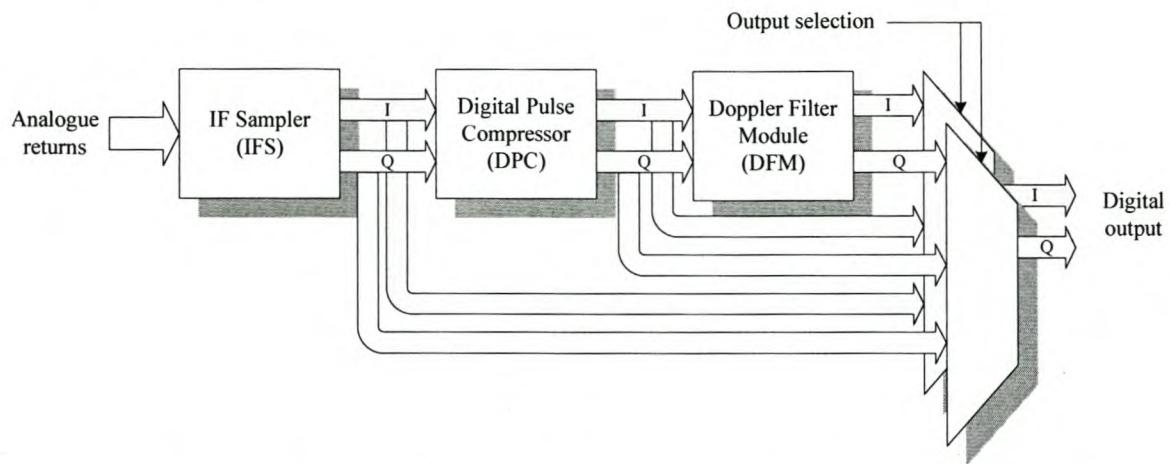


Figure 6.1: Simplified block diagram of the RSP's processing chain.

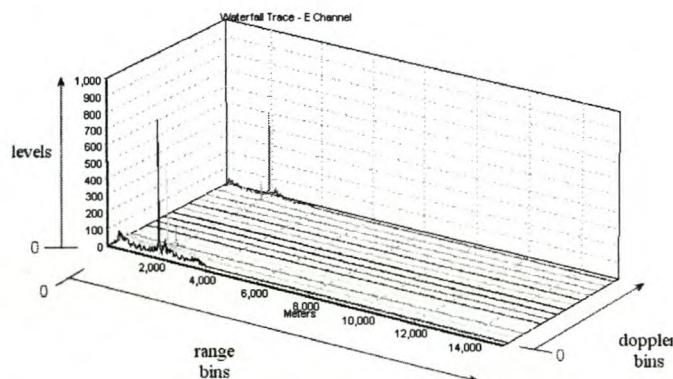


Figure 6.2: Annotated example of the DCM software's waterfall trace.

the diagram, the RSP's output I and Q samples can be sourced from any of the processing blocks.

Part of the DCM software's GUI is shown in Figure 6.2, as an example. The user can choose to plot either the I samples, the Q samples, or the magnitude of the complex sum. The DCM software terms these the I-channel, the Q-channel, and the E-channel (i.e. envelope), respectively. The RSP's complete processing chain was used for the example, so the waterfall trace shows a range/doppler map. The annotations in Figure 6.2 indicate which axes represent range, doppler, and the level of the samples. Units for the range axis can be toggled between metres and range bins. The example shows a single stationary target at a range of 2000 m, which can be identified by the large spikes. The target is static, so it is centred in the zero doppler bin.

When using the IFS or DPC's output, the waterfall trace's depth axis is slightly different. The plots in the doppler bins now show the range cell samples taken for each PRI in the

burst. The first PRI is in doppler bin zero. For a non-fluctuating jtarget with no doppler shift, the returns for each pulse should be identical (e.g. Figure 6.8, p. 116).

The DCM software can also display other plots such as a standard AR (amplitude-range) trace, and a surface trace. We refer to a “2-D waterfall trace” when viewing the waterfall trace looking down the doppler axis. This is similar to an AR trace, except that all the pulses are superimposed on each other (the AR trace only shows the peak values).

In order to analyse the tracking performance of the system, two additional programs are needed. The first is the SDP Logger which logs information from the SDP — we are concerned with the tracking data. The second program is the Offline Tracking Visualisation Software, which provides a graphical view of the logged data. We are only concerned with the graphs generated with these two programs — the details of the applications will not be discussed.

6.3 Target returns

The simulator’s principal function is to generate target returns with parameters determined from its environmental models. The tests in this section serve to verify that the return parameters can in fact be correctly controlled.

The experiments only use the test target feature of the real-time engine, so operation of the pre-processing engine is not verified here.

6.3.1 Test set up

A large part of the test set up is common to all the target return tests, and is described here.

1. Using the GUI, program a $5\ \mu\text{s}$ continuous wave pulse (PCC 1, on the GUI).
2. Set up the RSP so that it also uses a $5\ \mu\text{s}$ CW pulse.
3. Both the RSP and simulator must be set to use a single burst pattern consisting of 14 pulses. The PRI is $252\ \mu\text{s}$.
4. Set up the RSP so that output is taken directly from the IFS.
5. Set up a test target in range bin 800, with zero doppler shift, zero phase shift, and an amplitude of 0,1.
6. Turn off the clutter generator.
7. Start real-time playback of the scenario.
8. Position the radar beam so that only the test target is visible.

All measurements will be taken using only the sum channel. For tests where a measured value v_{meas} is compared to a nominal value v_{nom} , the relative error e_{rel} is calculated as

$$e_{rel} = \frac{v_{nom} - v_{meas}}{v_{nom}} . \quad (6.1)$$

6.3.2 Pulse width

Objective

The objective of the experiment is to verify the width of the pulse output by the waveform generator.

Protocol

1. Perform the common set up in Section 6.3.1.
2. Measure the pulse width v_{meas} using the oscilloscope.
3. The nominal pulse width is $5\ \mu s$.

Results

The measured pulse width was $v_{meas} = 5,000\ 00\ \mu s$, so the relative error was $e_{rel} = 0,000\%$.

Discussion

The oscilloscope showed the pulse width to be correct to within 10 ps of the nominal value. It is critical that the pulses generated by the simulator can be controlled exactly. From this test we conclude that the pulse width can be controlled correctly.

6.3.3 Uncompressed pulse range

Objective

The objective of the experiment is to verify that an uncompressed target is generated at the correct range.

Protocol

1. Perform the common set up in Section 6.3.1.
2. Set the DCM software to display an AR trace.
3. The AR trace will show a very sharp drop near the end of the pulse. The end of the pulse is the last sample before the sharp drop off.

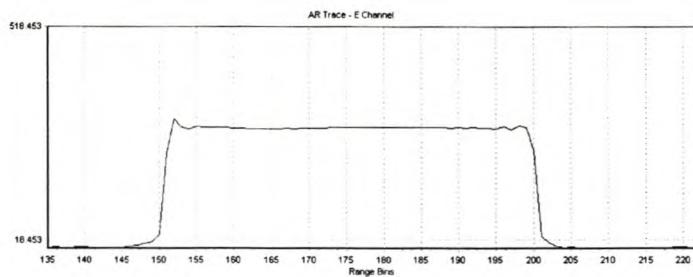


Figure 6.3: DCM’s AR trace showing the result for the uncompressed pulse range test (Section 6.3.3).

4. The radar’s I and Q samples are generated at 10 MHz, so one range bin on the DCM plot corresponds to four of the simulator’s range cells (it runs at 40 MHz). The end of the pulse should be in range bin $800/4 = 200$.

Results

Figure 6.3 shows the output. The end of the pulse was measured to be in range bin $v_{meas} = 200$, giving a relative error of $e_{rel} = 0\%$.

Discussion

Using the DCM software, we saw that the end of the uncompressed pulse was located in the correct bin. It is not surprising, as the simulator was calibrated prior to running all the experiments. Nevertheless the target range is one of the most important return parameters, so it is crucial that it is correct.

While not related to this test, we observe that the pulse is relatively flat. This is as expected — the I and Q components of the baseband signal should not be changing independently, so the envelope must be constant.

6.3.4 Amplitude

Objective

The objective of this experiment is twofold: firstly, we verify that the relative amplitude of the target return can be controlled and secondly, measure the amplitude fluctuations from pulse to pulse.

Protocol

1. Perform the common set up in Section 6.3.1.
2. Set the DCM software to display the waterfall trace in 2-D mode, viewing the envelope.

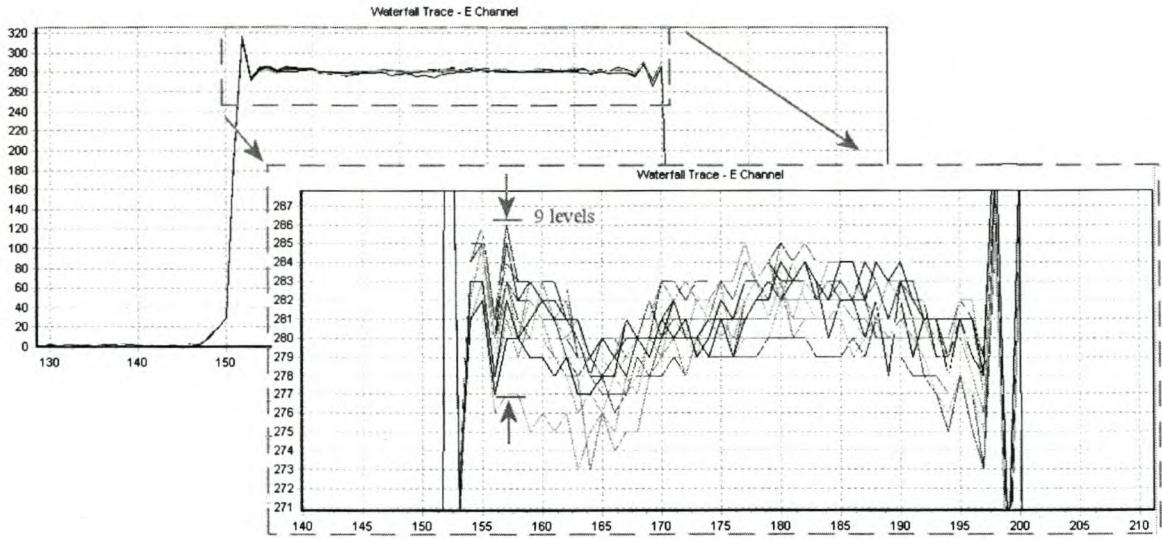


Figure 6.4: DCM's 2-D waterfall trace showing the result for the amplitude test (Section 6.3.4). Test target amplitude is 0,1.

3. Measure the mean amplitude of the pulse, v_{meas1} .
4. Measure the maximum peak-to-peak difference in the pulse amplitude over all the PRIs in a burst, Δv_{meas1} . The test target generates a non-fluctuating return, so ideally the measured amplitude should be constant.
5. Set the test target's amplitude to 0,05 (half of what it was originally).
6. Measure the mean amplitude of the pulse, v_{meas2} .
7. Measure the maximum peak-to-peak difference in the pulse amplitude over all the PRIs in a burst, Δv_{meas2} .
8. The ratio of the mean amplitudes v_{meas1}/v_{meas2} should be two.
9. Measure the highest level of the noise floor, Δv_{meas3} .

Results

Figure 6.4 shows the output for a test target amplitude of 0,01. The mean is $v_{meas1} = 281,28$ levels, and the maximum deviation as $\Delta v_{meas1} = 9$ levels. With the test target amplitude set to 0,05, the result in Figure 6.5 was obtained. The mean is $v_{meas2} = 140,70$ levels, and the maximum deviation is $\Delta v_{meas2} = 5$ levels. The maximum level of the noise floor was measured as $\Delta v_{meas3} = 4$ levels

The ratio of the mean amplitudes is $281,28/140,70 = 1,9991$, giving a relative error of $e_{rel} = (2 - 1,9991)/2 = 0,0450\%$.

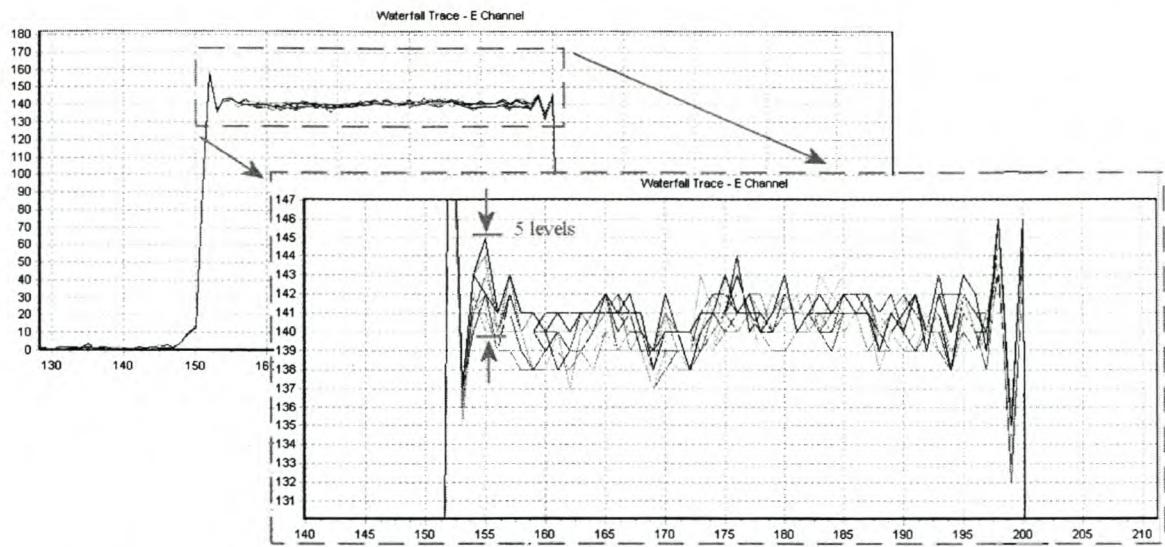


Figure 6.5: DCM’s 2-D waterfall trace showing the result for the amplitude test (Section 6.3.4). Test target amplitude is 0,05.

Discussion

The error is surprisingly small, considering that 12-bit DAC outputs theoretically have 2048 different amplitudes, allowing for a minimum change of $1/2048 = 0,05\%$. This is attributed to the fact the mean measurements average out the noise induced by the DAC quantisation and non-linearities (these effects are well described in [89, ch. 5]). Amplitude is one of the four most important characteristics of a radar return (others are the delay, phase, and frequency shift), so accurate control is essential.

The pulse-to-pulse fluctuations are larger than the maximum variations seen in the noise floor (essentially quantisation noise from the IFS’s ADCs). This indicates that the pulses’ fluctuations have a source other than the noise in the RSP’s front end. The most likely source is glitches on the DAC board’s data lines. This hypothesis is supported by the noise’s amplitude dependence. More of the data lines need to be switched for a larger signal, thus there are more signals that can be affected, and that can affect others. If this is indeed the problem, then termination resistors on the data lines would help reduce the fluctuations. DAC non-linearities were considered as an additional noise source, but later discounted. While the non-linearities may result in the generation of a distorted signal, they would not cause the pulse-to-pulse fluctuations seen here.

6.3.5 Phase

Objective

This experiment's objective is to verify that the target phase can be controlled coherently, pulse to pulse.

Protocol

1. Perform the common set up in Section 6.3.1.
2. Set the test target's doppler speed to 3,451 m/s (the doppler phase increment will then be 21,18° every PRI, giving a rotation of approximately 360° each burst).
3. Set the DCM software to display I channel samples on the waterfall trace.
4. Measure the average level of the pulse in each PRI.
5. Repeat for the Q channel samples.
6. Normalise the results, and then find the angle of the complex value $I + jQ$ for each pulse. This angle should increment by the value stated above, every PRI.

Results

Figure 6.6 shows the DCM's output for the I and Q channels using surface plots. The expected sinusoidal modulation is clear, as is the 90° phase difference between the two channels.

The doppler speed of 3,451 m/s does not result in exactly one phase revolution per burst, so the phase drifts with time. The measurements were made with the phase of the first pulse close to, but not exactly, 0°, thus introducing a phase offset into the measurements. By subtracting the mean phase error, the measurement artefact is effectively removed.

The phase angles derived from the measurements are shown in Figure 6.7. The phase error is not easy to interpret directly, instead we consider the effects on range and doppler speed measurement. We derive the necessary equations below.

The phase induced range error can be determined by rearranging eq. (2.19), as

$$R_{err} = \frac{\Delta\phi_{err} \lambda}{4\pi} \quad [\text{m}]. \quad (6.2)$$

In one PRI, the doppler speed error $v_{dop\ err}$ is calculated as the quotient of the range error and the PRI length, or alternatively

$$v_{dop\ err} = R_{err} \cdot f_{PRF} = \frac{\Delta\phi_{err} \lambda f_{PRF}}{4\pi} \quad [\text{m/s}] , \quad (6.3)$$

where f_{PRF} is the pulse repetition frequency [Hz].

This can be compared to the doppler speed resolution Δv_{dop} (the change in speed represented by one doppler bin). The doppler sample rate is equal to the PRF, so for n bursts,

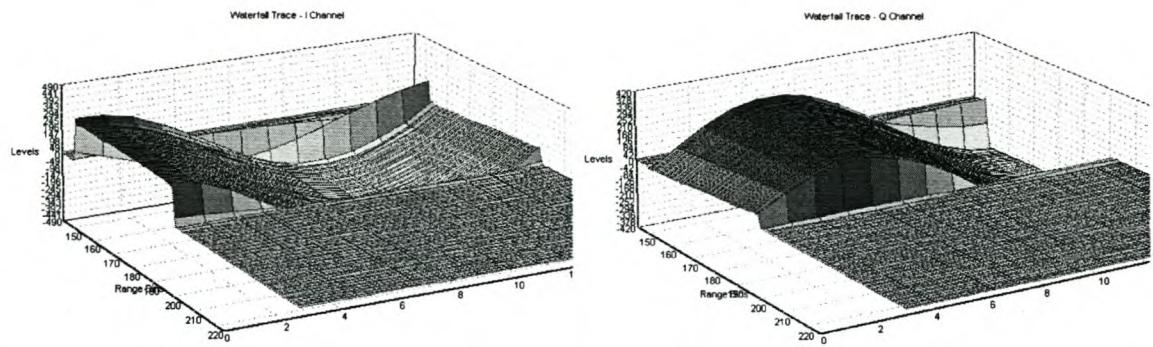


Figure 6.6: DCM's surface trace showing the *I* and *Q* channel output for the phase test (Section 6.3.5).

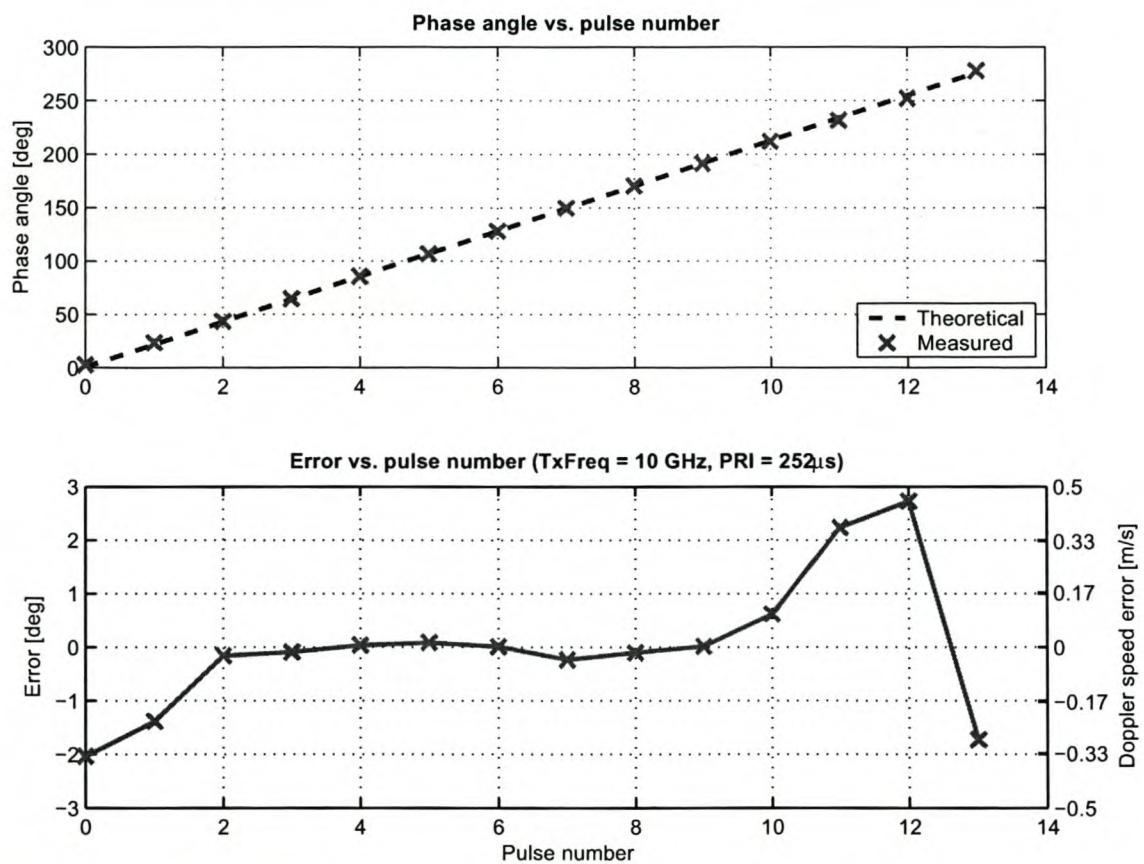


Figure 6.7: Measured pulse-to-pulse phase increment and error for the phase test (Section 6.3.5).

the doppler frequency resolution is $\Delta f_d = f_{PRF}/n$. This can be substituted into the doppler equation, eq. (2.21), to determine the doppler speed resolution, i.e.

$$\Delta v_{dop} = \frac{\Delta f_d \lambda}{2} = \frac{f_{PRF} \lambda}{2n} \text{ [m/s]} . \quad (6.4)$$

Discussion

The maximum phase error is $2,73^\circ$. At 10 GHz, the wavelength is $\lambda \approx 3 \text{ cm}$, so from eq. (6.2) the maximum pulse-to-pulse range error will be $114 \mu\text{m}$. A negligible amount, in terms of a tracking radar. The RSP does not derive range from the phase information, so a more important measurement is the doppler speed error. Using eq. (6.3), the maximum error is 0,452 m/s. The doppler speed resolution for the test waveform pattern is 4,25 m/s. Thus $v_{dop\ err}$ is an order of magnitude smaller than the RSP's doppler resolution, and can be considered insignificant, in engineering terms.

6.3.6 Pulse compression

Objective

Verify that the return pulse is compressed by the DPC, and that the compressed target range is correct.

Protocol

1. Perform the common set up in Section 6.3.1.
2. Add the DPC to the RSP's processing chain.
3. Set the DCM software to display the envelope channel on a waterfall trace.
4. Verify that the target return is compressed — there should be a triangular pulse for the target, in each PRI.
5. The nominal width of the pulse is 99 samples, with the peak in the centre of the pulse (this is explained below).
6. Measure the range bin that the peak occurs in. As with the uncompressed pulse test, the DCM software should show the peak in range bin 200, for a test target in the simulator's 800th bin. The peak should also be at the same range for all the PRIs.

In theory, the baseband I and Q samples for a pure, continuous wave pulse will be constant, as there is no phase modulation. Thus the target return can be seen as a rectangular pulse. In general, when convolving two sequences, the result's non-zero width is one less than the sum of the sequences' widths [17, p. 412]. A $5 \mu\text{s}$ pulse is 50 samples wide at 10 MHz. If we consider samples equal to or below the noise floor as “zero” samples, then the convolved pulse should be 99 samples wide. When the matched filter convolves two rectangular pulses of equal width, the output will have a symmetrical, triangular shape.

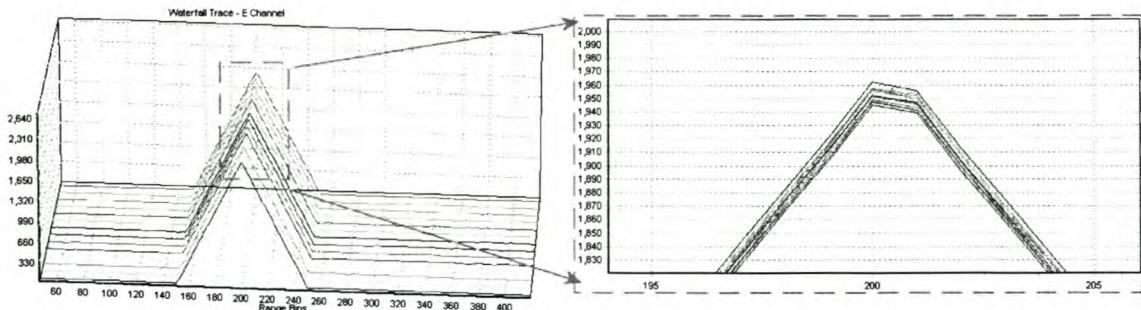


Figure 6.8: Output for the pulse compression test (Section 6.3.6).

Results

The compressed pulses can be seen in Figure 6.8. Triangular pulses are generated, all peaking in range bin 200. The width of the triangular pulses is 100 samples, with the first and last range samples (above the noise floor) in bins 151 and 250, respectively.

The peaks were aligned in the correct range bin, so the target range is correct. The relative error in the compressed pulse width is $(99 - 100)/99 = -1,01\%$.

Discussion

The range of the compressed target was correct — this is not surprising as the uncompressed pulse's range was correct (Section 6.3.3). From the graph we see that the peak of the compressed target is slightly offset from the centre of range bin 200. As the error is less than half a range bin, it is of no significance to the RSP. A possible cause for the error is a fractional delay between the IFS's sampling clock edge, and the edges that clock the DACs.

The width of the compressed pulse is not exactly as predicted, but the error is relatively small. The RSP uses the peak of the compressed target to detect targets, so the inexact width of the compressed pulse is not a major concern. Again, the error can be attributed to the marginal time difference between the IFS and DAC clock edges.

6.3.7 Doppler

Objective

The objective of this experiment is to verify that a target can be generated in any doppler bin.

Protocol

1. Perform the common set up in Section 6.3.1.
2. Add both the DPC and the DFM to the RSP's processing chain.

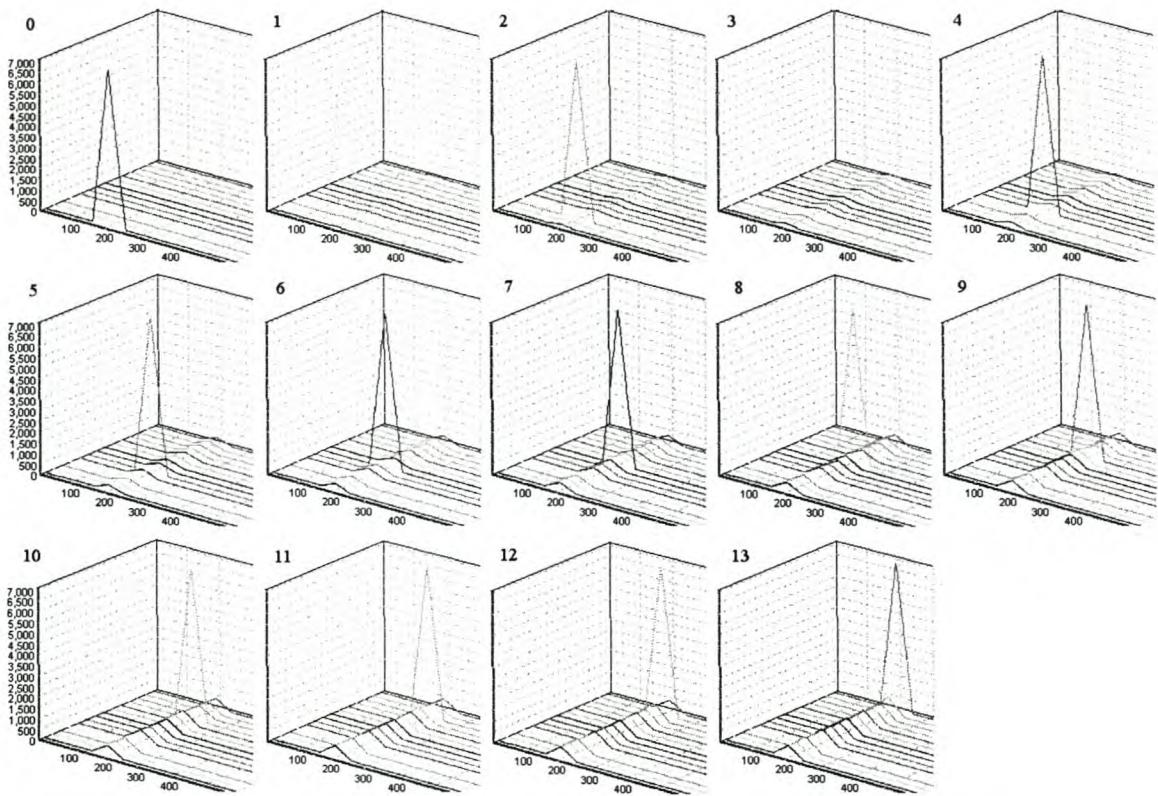


Figure 6.9: Doppler test output for all doppler bins (Section 6.3.7). The numbers in the top left corners indicate the doppler bin in which the target should be.

3. Set the DCM software to display the envelope channel on a waterfall trace.
4. Start with a doppler speed of 0 m/s, and then increase it in steps of 4,24 m/s (approximately equal to the doppler speed resolution) 13 times.
5. Verify that the target peak shifts one doppler bin for each speed increment. The target should start in doppler bin 0 and end in doppler bin 13.

Results

In Figure 6.9 the waterfall traces are shown for all 14 steps. All the peaks occurred in the expected doppler bins.

Discussion

The experiment shows that the simulator can successfully generate a target in any doppler bin. We observe that as the target speed increases, the doppler sidelobes also increase. The reason is that the speed increment of 4,24 m/s is not exactly equal to the doppler speed resolution (4,2487 m/s). After each increment, the test target's speed moves further from

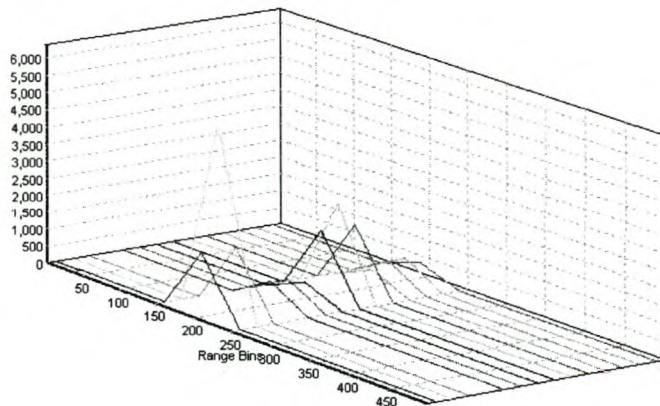


Figure 6.10: Waterfall trace for co-located targets experiment (Section 6.3.8).

a multiple of the doppler resolution. If the doppler speed is not a multiple of the doppler resolution, then some of the energy will leak into the other doppler bins.

6.3.8 Co-located targets

Objective

This experiment's objective is to verify that the simulator can simultaneously generate two co-located targets.

Protocol

1. Perform the common set up in Section 6.3.1.
2. Add both the DPC and the DFM to the RSP's processing chain.
3. Set the first test target's doppler speed to 4,249 m/s (doppler bin one), and set its amplitude to 0,1.
4. Enable the second test target, also in range bin 800, a doppler speed of 33,99 m/s (doppler bin eight), and an amplitude of 0,05.
5. View a waterfall trace (envelope channel) using the DCM software.
6. Verify that there are two target peaks, and that they are in the correct doppler bins.

Results

The resultant waterfall trace is visible in Figure 6.10. There are two target peaks in range bin 200. The larger target is in doppler bin one, and the smaller target in doppler bin eight.

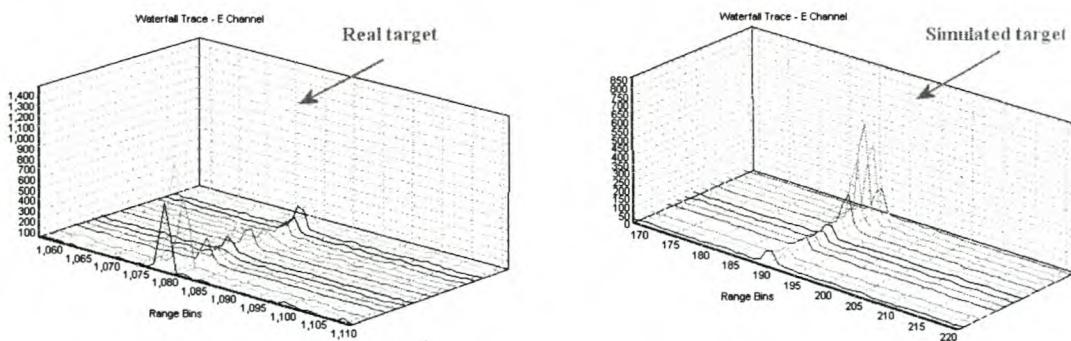


Figure 6.11: Waterfall trace showing the returns from both a real target and a simulated target.

Discussion

The simulator is capable of generating co-located targets. Of interest in this test, is the high sidelobes, especially in the doppler bins adjacent to the targets. The reason for this is actually an incorrect test set up. For the previous tests, the DFM's windowing function was disabled (i.e. uniform windowing), however for this test it was enabled. The windowing function is responsible for the high sidelobes.

6.3.9 Comparison to real-world target returns

This section is not an experiment, as such, but rather serves as a subjective comparison between returns from a real-world target and a simulated target. The real-world target is a jet aircraft, and the simulated target also an aircraft. The exact settings used for the simulated target are unimportant. The RSP's full processing chain is used, with non-uniform DFM windowing. A chirped waveform was used in both instances; however, the waveform used for the real target had 18 pulses per burst, compared to the 14 pulses used for the simulated target. Figure 6.11 shows the comparative waterfall traces.

The shape of the two returns look very similar — a narrow triangular peak with relatively high sidelobes. Compared to the compressed pulses in the previous three sections, these returns are much narrower (about five range bins, as compared to 100). The reason being the use of a chirp waveform — the chirp's higher bandwidth-time product improves the range resolution.

The lack of a doppler scintillation model (see p. 28) is apparent. The simulated target has a relatively smooth doppler spectrum, while real-world target's is more spiky. We do not compare the temporal properties of the target returns.

6.4 Clutter

The clutter implementation is relatively simplistic, so the verification thereof is likewise. We only consider the fundamental characteristics of the clutter in relative terms. These traits are the clutter's range extent, its range dependence, its mean level, and its mapping into doppler space.

6.4.1 Test set up

The parts of the test set up common to all the clutter return tests are described below.

1. Using the GUI, program a $5\ \mu\text{s}$ continuous wave pulse (PCC 1, on the GUI).
2. Set up the RSP so that it also uses a $5\ \mu\text{s}$ CW pulse.
3. Both the RSP and simulator must be set to use a single burst pattern consisting of 14 pulses. The PRI is $252\ \mu\text{s}$.
4. Set up the RSP so that output is taken directly from the IFS.
5. All test targets should be disabled.
6. Turn on the clutter generator.
7. Start real-time playback of the scenario.

The GUI has an option to turn off the clutter fluctuations. When this option is selected, the return's amplitude is determined from the mean scattering coefficient, rather than taking a sample from the PDF. However, the IFS will filter out a DC signal, so we randomise the phase. The resultant signal still fluctuates, but the envelope gives an indication of the mean clutter level.

As with the target return tests, all measurements will be taken using only the sum channel.

6.4.2 Clutter map

In this section the pre-processed clutter map is compared to the terrain defined in the scenario. This is a subjective, rather than an objective evaluation. Figure 6.12 shows both the scenario map and the resultant pre-processed clutter map. One of the parameters stored in the pre-processed clutter map is the mean clutter level — the contours indicate the amplitude of the mean, in decibels.

Overall, the pre-processed clutter map approximates the defined terrain well. The sectorised approach to the clutter processing is clear. The jagged edges are due to the large width (in azimuth) of the range cells and the fact that the terrain type at the centre of a range cell defines the terrain throughout that cell. The area of the cells increases with range,

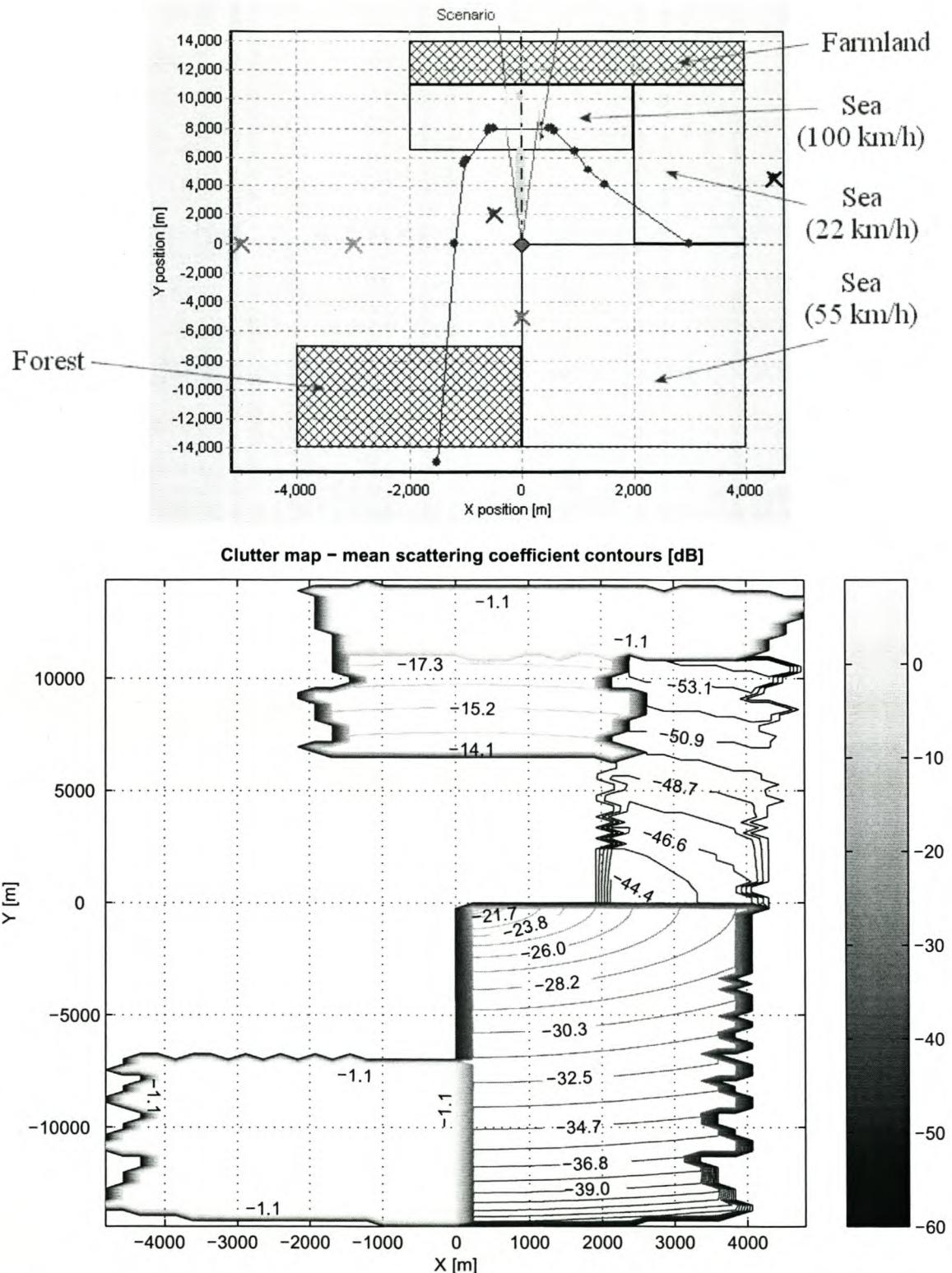


Figure 6.12: The scenario map with the wind speed in parenthesis (top), and a contour plot showing the mean scattering coefficients after pre-processing (bottom).

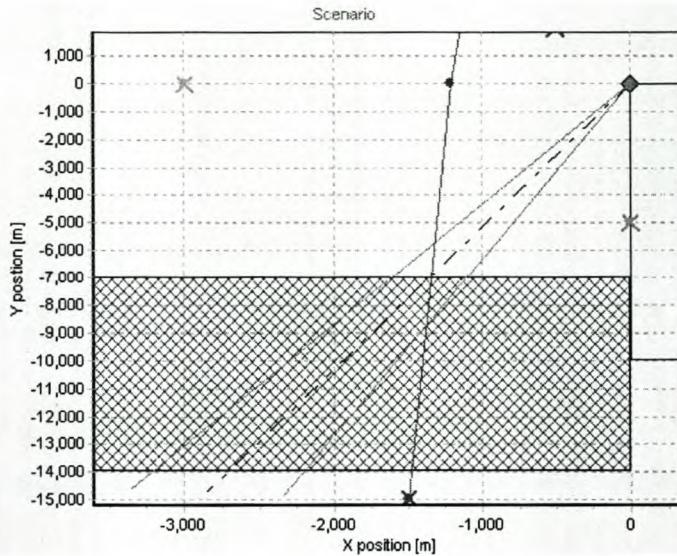


Figure 6.13: Scenario set up for clutter range extent test (the cross-hatched region represents forest, which results in the clutter).

which causes the edges of the pre-processed clutter to get more and more ragged. The mean level of the ground clutter (forest and farmland) is constant, while the sea clutter shows a decreasing amplitude with range. The reason for this is that the ground clutter model (see p. 171) ignores the grazing angle, while the sea clutter model (p. 165) does not. The effect of the wind speed on the sea clutter model can be seen — the stronger the wind, the larger the clutter returns (because the waves will be bigger).

6.4.3 Range extent

Objective

This experiment's objective is to verify that the surface clutter is generated at the correct range (as determined by the real-time engine).

Protocol

1. Perform the common set up in Section 6.3.1.
2. Set the radar's azimuth and elevation angles to $191,0^\circ$ and $0,0^\circ$, respectively (Figure 6.13). At this angle the real-time engine reports clutter in the radar beam from range bins 1921 to 3638. For the RSP, with a four times lower range resolution, the clutter should extend from range bin 480 to range bin 967.

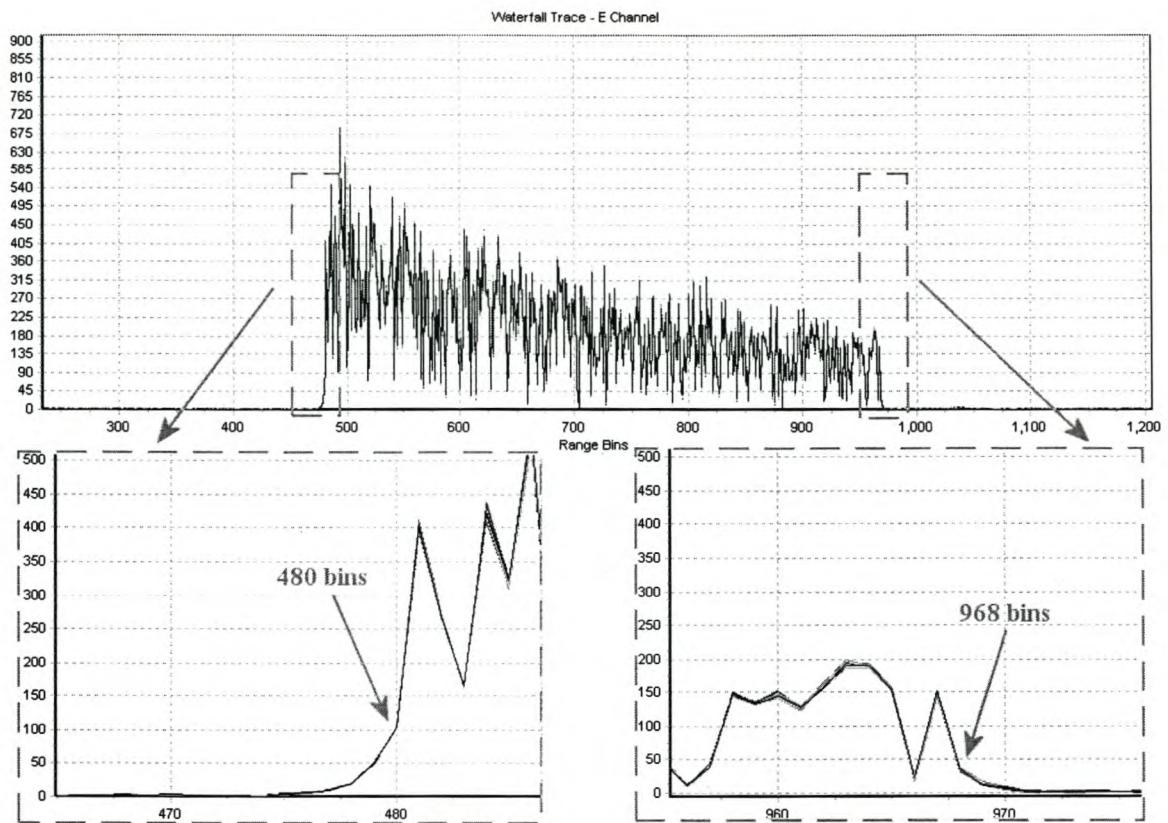


Figure 6.14: 2-D waterfall trace for clutter range extent test (Section 6.4.3).

Results

Figure 6.14 shows the resultant 2-D waterfall trace. The zoomed in sections of the image show the minimum clutter range to be 480 bins, and the maximum range 968 bins. The relative error for the minimum range is zero, and for the maximum range it is $(967 - 968)/968 = -0,1\%$.

Discussion

As discussed at length in Chapter 2, clutter returns are noise-like signals. In light of this, an extra range bin of clutter will be negligible in all but the highest fidelity simulators.

There is clearly a tapering effect at the beginning and the end of the clutter region. We attribute this to the filtering in the IFS which tends to smear fast changing signals such as the clutter.

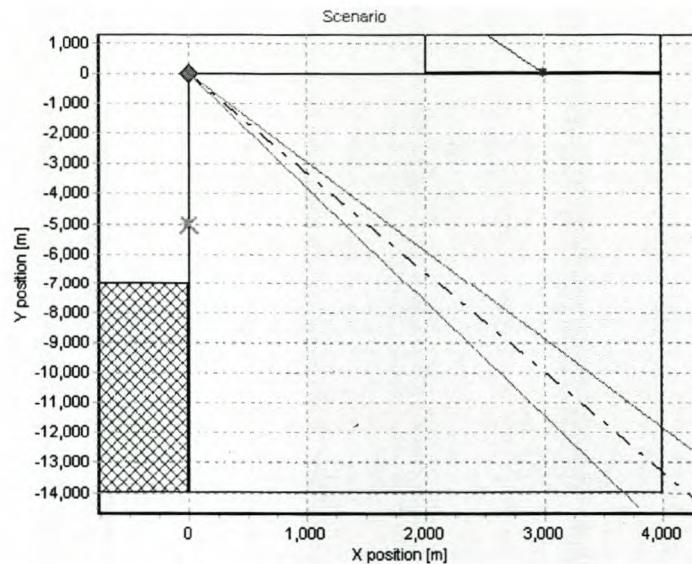


Figure 6.15: Scenario set up for clutter range dependence test (the hatched region on the right represents sea).

6.4.4 Range dependence

Objective

This experiment's objective is to verify that the clutter's mean amplitude exhibits the correct range dependence.

Protocol

1. Perform the common set up in Section 6.3.1.
2. Turn off the clutter fluctuations (using the GUI), so that the mean level can be viewed.
3. Set the radar's azimuth and elevation angles to $163,3^\circ$ and $0,0^\circ$, respectively (see Figure 6.15).
4. Compare the clutter envelope to the simulated range profile (discussed in more detail below).

The trend in the range profile is determined by a number of factors. Firstly, a radar's return power shows a range (R) dependence of $1/R^4$, giving an amplitude dependence of $1/R^2$ (eq. (2.18)). Secondly, a range cell's area is directly proportional to R , due to the spreading of the beam (see eq. (2.29)). Combining these two effects, gives a range dependence of $1/R$. Finally, the range dependent grazing angle Ψ must be considered, as it affects the scattering coefficient — according to eq. (B.63), $\sigma^0 = \gamma \sin(\Psi)$. From eq. (B.64) on p. 166, the range dependence due to the grazing angle can be determined as follows:

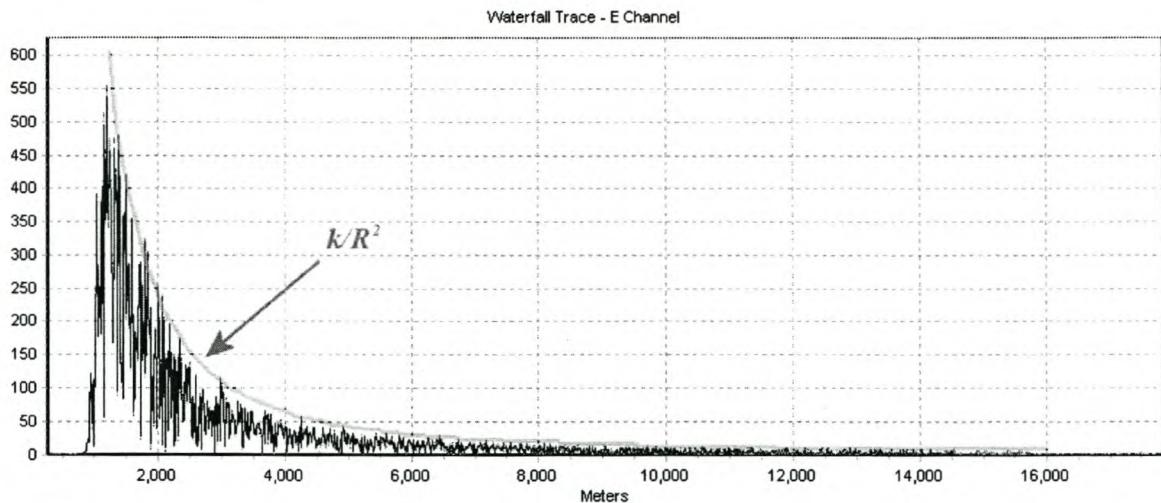


Figure 6.16: 2-D waterfall trace for clutter range dependence test (Section 6.4.4), overlayed with a curve proportional to $1/R^2$.

$$\begin{aligned}
 \sin(\Psi) &= \frac{a_e^2 + R^2 - (a_e + h_r)^2}{2a_e R} \\
 &= \frac{R^2}{2a_e R} - \frac{2a_e h_r}{2a_e R} - \frac{h_r^2}{2a_e R} \\
 &\approx \frac{h_r}{R},
 \end{aligned} \tag{6.5}$$

where the assumptions $a_e \gg R$ and $a_e \gg h_r^2$ were made. They are true for most radars concerned with earth-based objects, including the RTS6400. The grazing angle's range dependence effectively cancels out the effect of the range cell's area, thus the clutter's mean amplitude should be proportional to $1/R^2$.

Results

The resultant 2-D waterfall trace is visible in Figure 6.16. A curve inversely proportional to R^2 has been superimposed on the clutter amplitude.

Discussion

The peaks of the clutter signal clearly have a $1/R^2$ range dependence, as expected. We conclude that, bar a scaling factor, the pre-processing engine, the real-time engine, and the clutter generator are correctly handling the amplitude of the signal.

The clutter is very spiky, even though the K-distribution fluctuations are disabled. This is due to the random phase change applied to the signal, as discussed in Section 6.4.1.

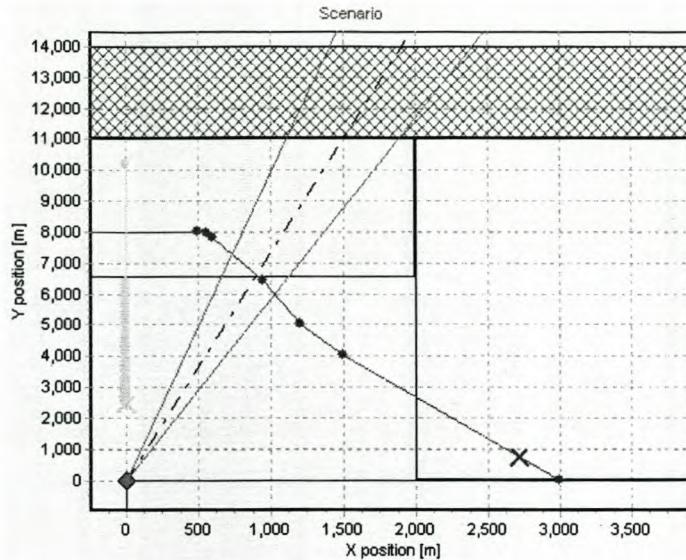


Figure 6.17: Scenario set up for the mean clutter level test (the cross-hatched region at the top represents farmland, and the lower hatched region, sea).

6.4.5 Mean clutter level

Objective

The objective of this experiment is to qualitatively verify that the simulator generates different returns for different types of terrain.

Protocol

1. Perform the common set up in Section 6.3.1.
2. Set the radar's azimuth and elevation angles to $7,73^\circ$ and $0,0^\circ$, respectively (see Figure 6.17). This setup illuminates a section of sea followed by a section of agricultural land (there are no targets in the beam).

Results

The clutter returns are shown in Figure 6.18 using a 2-D waterfall trace.

Discussion

This is a very simple test, and the results clearly show that strengths of the clutter returns, from the sea and the farmland, differ.

Of interest in this test is the fact that the farmland's amplitude-range dependence is more pronounced than that of the sea clutter, following a $1/R$ drop off, as opposed to the

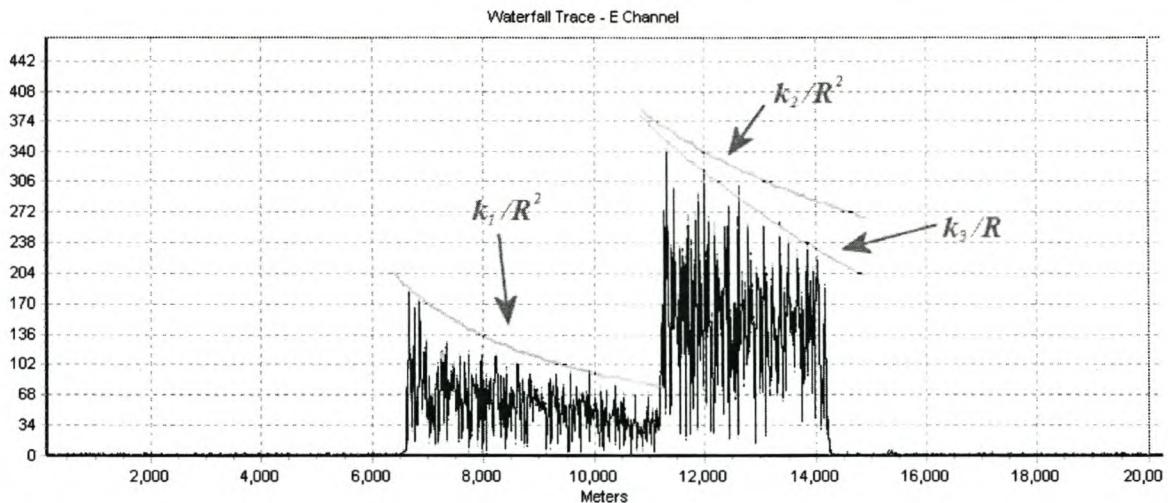


Figure 6.18: Two-dimensional waterfall trace for clutter means test (Section 6.4.5), with curves showing $1/R$ and $1/R^2$ amplitude drop off.

$1/R^2$ rate seen previously. The reason for this is that the model is not dependent on the grazing angle. Without the $\sin(\Psi)$ factor, the clutter amplitude's range drop off should be proportional to $1/R$.

6.4.6 Doppler

Objective

This experiment's objective is to verify that the fluctuating clutter is centred in zero doppler.

Protocol

1. Perform the common set up in Section 6.3.1.
2. Add both the DPC and the DFM (uniform windowing) to the RSP's processing chain.
3. Ensure that the clutter fluctuations are on (using the GUI).
4. Set the radar's azimuth and elevation angles to 163.3° and 0.0° , respectively (see Figure 6.15).
5. Measure the attenuation of the non-zero doppler sidelobes relative to the zero doppler peak.

Results

The two- and three-dimensional waterfall traces in Figure 6.19 show the range-doppler map for the clutter returns. The peak in zero doppler is measured at 885 levels, and the

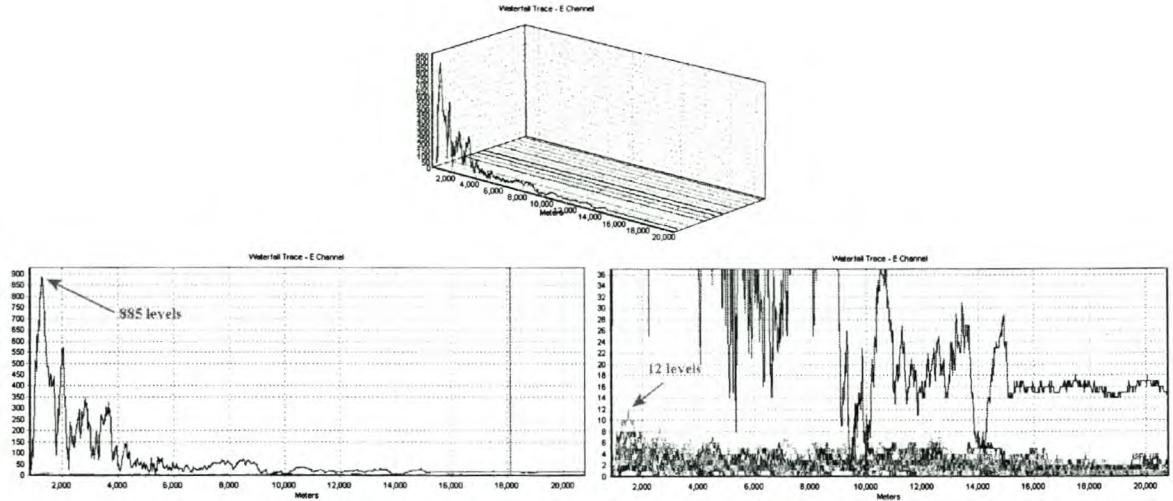


Figure 6.19: Waterfall trace for clutter doppler test (Section 6.4.6).

non-zero doppler peak at 12 levels. The amplitude of the sidelobes is thus suppressed by $20 \log(12/885) = 37,4 \text{ dB}$.

Discussion

From the waterfall traces, we observe that the clutter is indeed centred in the zero doppler bin, with very little energy in any of the other doppler bins. The sidelobe suppression of 37,4 dB confirms this.

There are large fluctuations in the clutter, but the profile is much smoother than in the previous clutter tests. The reason for this is the filtering performed by the DPC. The CW pulse has a constant phase, so the matched filter's coefficients are all the same — effectively a moving average filter over $5 \mu\text{s}$.

The fact that the simulator cannot generate doppler shifted clutter is a major limitation. Most clutter, even for a static ground-based radar will have some doppler spread. This is especially true for sea clutter, due to the water's flow. Figure 6.20 shows the output for real sea clutter, measured with a chirped pulse. Note that there is a target at a range of about 1700 m which creates the row of spikes in all the doppler bins.

The real sea clutter is centred in the second doppler bin, indicating relative movement between the sea and radar. There is more of a doppler spread than with the simulated clutter. Real sea clutter should spread slightly, but the main cause is the DFM's non-uniform windowing function used for the real-world measurements.

The chirped pulse has a finer range resolution than the CW pulse, so the real clutter is more spiky than the simulated clutter. The large spikes are most probably due to the Bragg scattering from capillary waves (see Section 2.5.2). Some spatial correlation is evident, as well as the expected amplitude-range dependence.

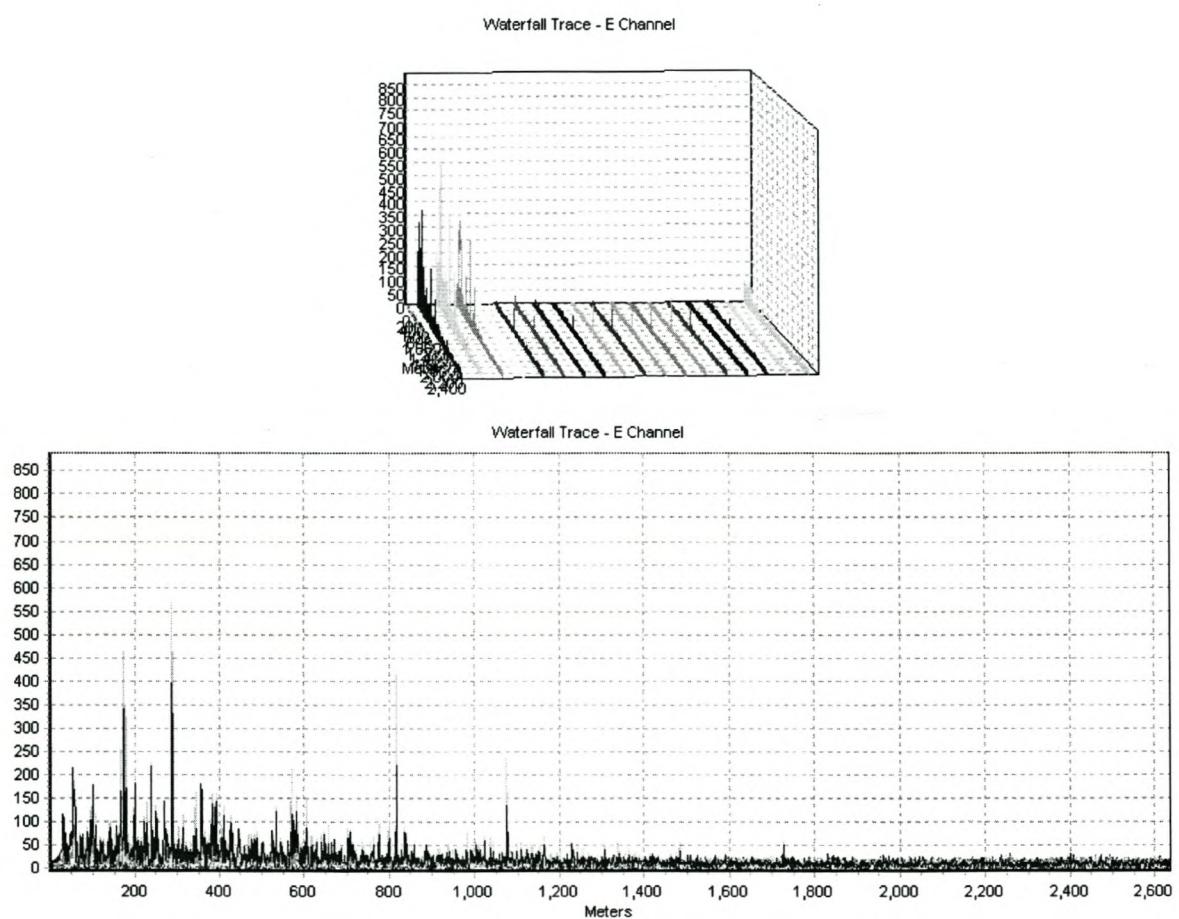


Figure 6.20: Waterfall trace for real sea clutter (there is a target at 1700 m).

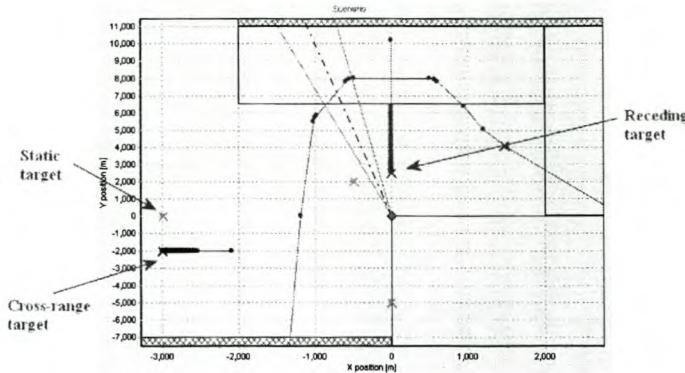


Figure 6.21: Scenario map showing the trajectories followed by the three tracking verification targets (all targets remain at the same height as the radar).

6.5 Tracking

The final section of the system verification evaluates closed-loop tracking. Three targets are tracked — a static target, a receding target (range changing, angles static), and a cross-range target (both range and angle changing). Figure 6.21 shows the trajectories followed by the three targets. The Offline Tracking Visualisation Software is used to generate plots of the radar measurements, which are then compared to the simulated values.

6.5.1 Test set up

The parts of the test set up common to all the tracking tests are described below.

1. Using the GUI, program a $12,1\mu\text{s}$ chirp pulse (PCC 2, on the GUI).
2. Set up the RSP so that it also uses a $12,1\mu\text{s}$ chirp pulse.
3. Both the RSP and simulator must be set to use a single burst pattern consisting of 14 pulses. The PRI is $252\mu\text{s}$.
4. Set up the RSP so that the full processing chain is enabled.
5. All test targets should be disabled.
6. The targets are of type *clsAircraft*, with no amplitude fluctuations (Swerling V).
7. The two moving targets are initially stationary, and then slowly accelerate.
8. Turn on the clutter generator.
9. Log the SDP's tracking data (using the SDP Logger).
10. Start real-time playback of the scenario.

The closed-loop tracking tests require the use of, and thus verify, all three receive channels (the sum and two difference channels). These tests also verify the pre-processing and real-time engines, for point scatterers (of type *clsAircraft*).

6.5.2 Stationary target

Objective

The objective of this experiment is to evaluate the system's closed-loop tracking performance with a stationary target.

Protocol

1. Perform the common set up in Section 6.5.1.
2. Designate the target at a range of 3000 m (the exact range is 2997,9 m), velocity of 0 m/s, azimuth of 270° and elevation of 0°.
3. Compare the target's simulated trajectory to the measured trajectory.

Results

The logged target position is compared to the simulated target position in Figures 6.22 and 6.23. The radar's worst range estimate is approximately 3001 m, giving a maximum absolute error of 3 m. The velocity estimation is made from the doppler information — the maximum error is 0,01 m/s (the range determined dx/dt rate's maximum error is -0,11 m/s). The maximum azimuth error is: $-90^\circ - (-89,93^\circ) = -0,07^\circ$, and the maximum elevation error: $0^\circ - 0,13^\circ = -0,13^\circ$. The mean azimuth angle is $-89,97^\circ$, giving an offset of 0,03°. For the elevation channel, the mean is also offset by 0,03°.

In the velocity plot, the doppler decorrelation jamming strobe is never asserted.

Discussion

Most notable in the results are the oscillations in the azimuth and elevation measurements. The oscillations have two main causes: the low resolution of the stored antenna tables, and the delay that the simulator adds to the radar's tracking control loop, resulting in an underdamped system. The delay has a number of sources: the ping-pong buffer adds a delay of one burst; boresight position updates are only received every few bursts (close to six bursts per angle update, for this waveform); and lost waveform generator data packets. The largest delay is that due to the boresight angle update rate, so increasing this rate would be the best way to improve performance. As discussed in Section 4.3.7 (p. 87), the antenna tables are stored with an angular resolution of 1 mrad (0,06°). This is thus a lower bound on the uncertainty of the radar's angular measurements, for a simulated target. Near the end of the simulation, the approximate amplitude of the angular oscillations are indeed 0,06°.

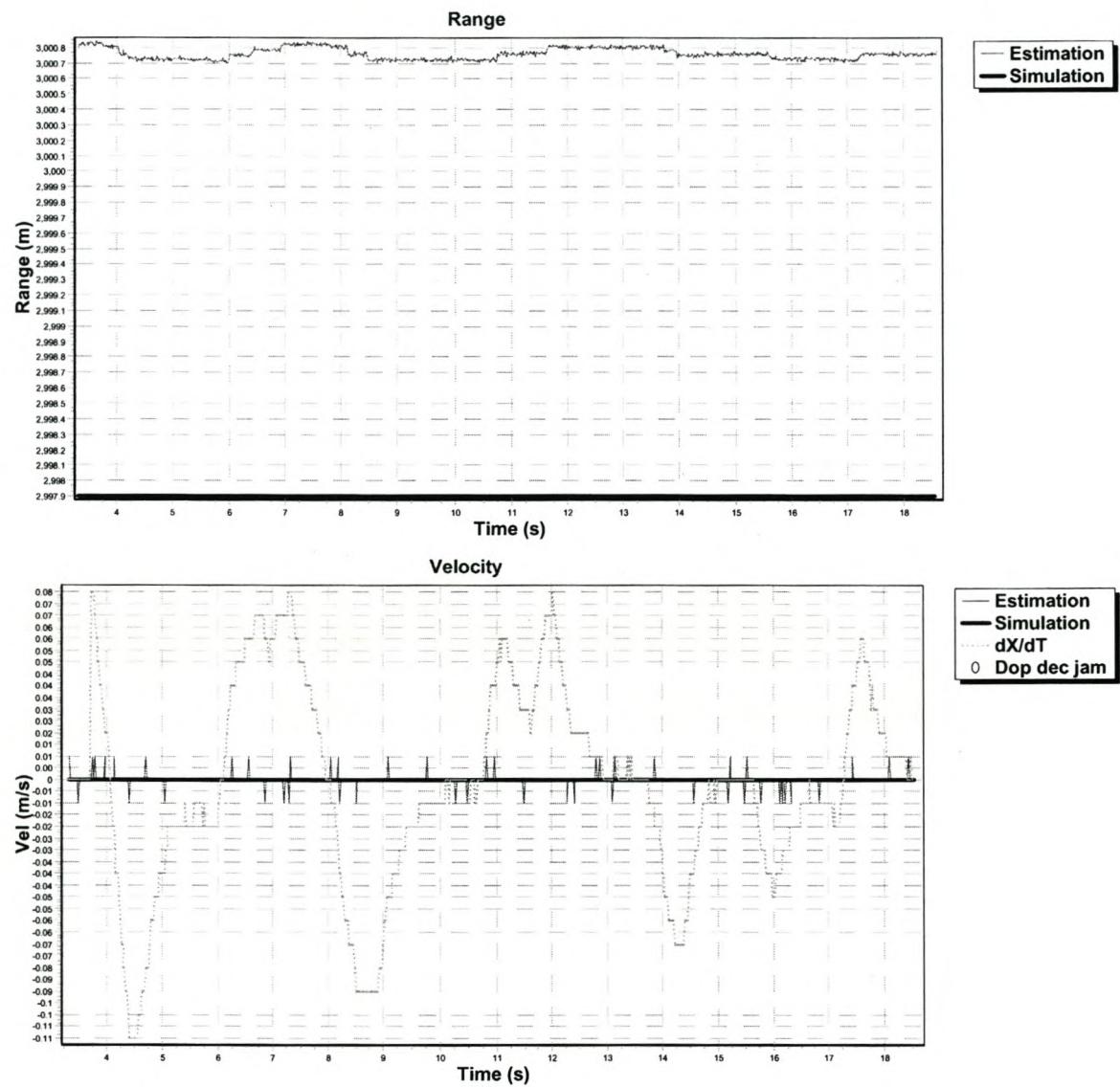


Figure 6.22: Comparison of simulated and measured range and velocity tracking results for the static target (Section 6.5.2).

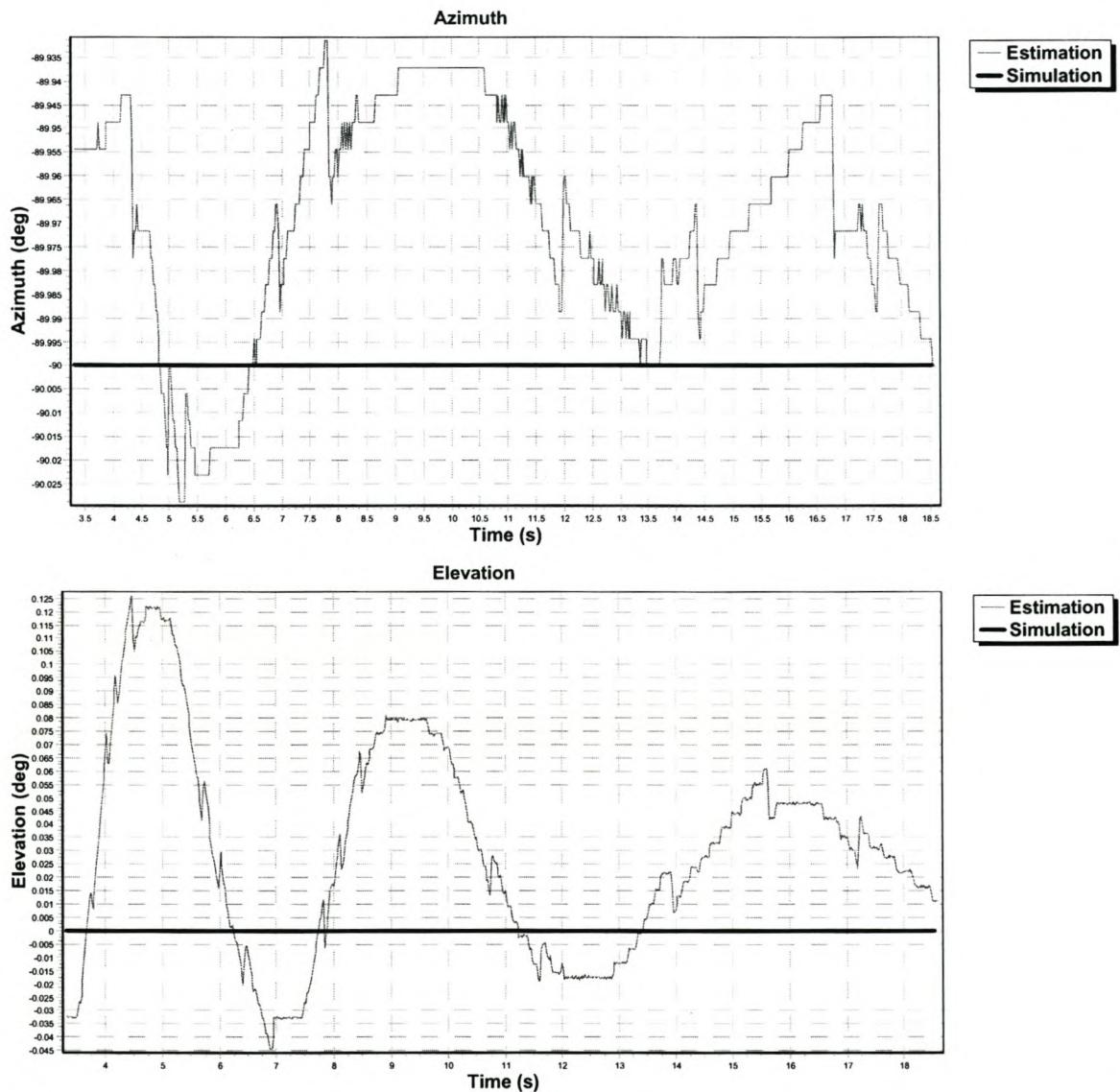


Figure 6.23: Comparison of simulated and measured angular tracking results for the static target (Section 6.5.2).

The maximum range error is well within the radar's range cell resolution (approximately 15 m), as expected from the pulse compression range test (Section 6.3.6). The measured range shows small oscillations which appear to be correlated to the angular oscillations. While the positioner converges on the target's true angle, the return's phase and amplitude changes (due to variations in the channel's antenna pattern). The range is determined from the Σ channel whose broad pattern (see Figure 2.2, p. 8) results in only minor phase and amplitude variations and correspondingly small range oscillations.

Variations in the estimated velocity are attributed to the imperfect per-pulse phase seen in Section 6.3.5. The log file limits the resolution of the velocity estimations to 0,01 m/s. The range rate (dx/dt) highlights the slight oscillations in the range measurement. The range rate correlates very well to the doppler velocity estimation, so the radar system does not assert its doppler decorrelation jamming strobe.

The mean errors in the measured angles are very small ($0,03^\circ$), and smaller than the $0,06^\circ$ resolution of the stored antenna pattern. The errors have two possible causes, the first is the error in the CORDIC sine function. The second is the fact that the simulator and the RSP used slightly different antenna patterns — the two sets of patterns were measured from similar, but not identical antennas.

6.5.3 Receding target

Objective

This experiment's objective is to evaluate the system's performance when tracking a target moving in range only.

Protocol

1. Perform the common set up in Section 6.5.1.
2. Designate the target at a range of 2500 m, velocity of 0 m/s, an azimuth of 0° and an elevation of 0° . Note that the target is initially at rest and then accelerates away from the radar.
3. Compare the target's simulated trajectory to the measured trajectory.

Results

The logged target position is compared to the simulated target position in Figures 6.24 and 6.25. The track proceeds well up to about 46 seconds. Up to this point, the mean angular error is approximately 0° . The measured range after 46 s is 5487 m, while the simulated trajectory reaches this range after only 36,80 s. The simulation playback is thus $|36,80 - 46,00|/36,80 = 25\%$ slower than real-time.

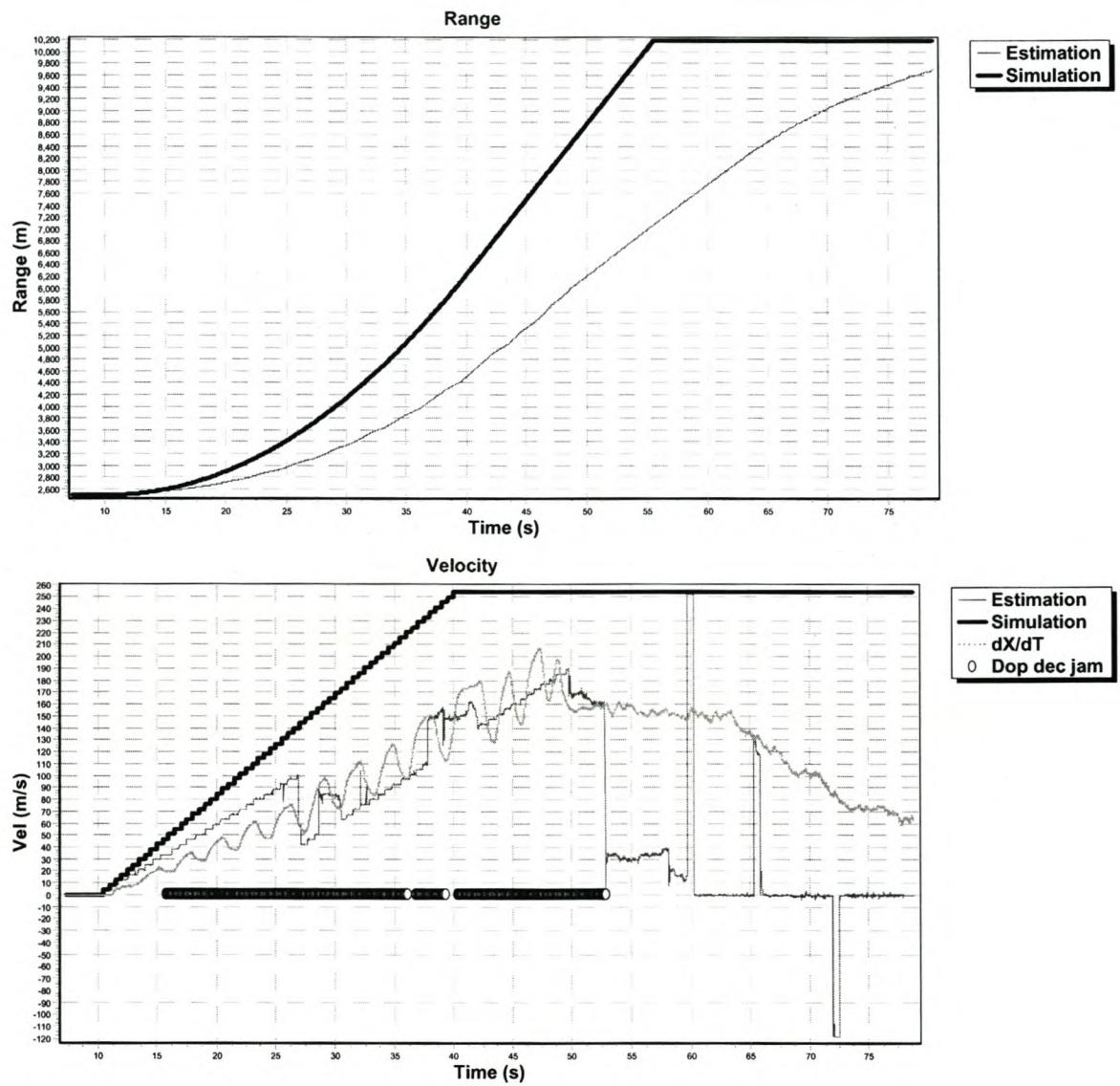


Figure 6.24: Comparison of simulated and measured range and velocity tracking results for the receding target (Section 6.5.3).

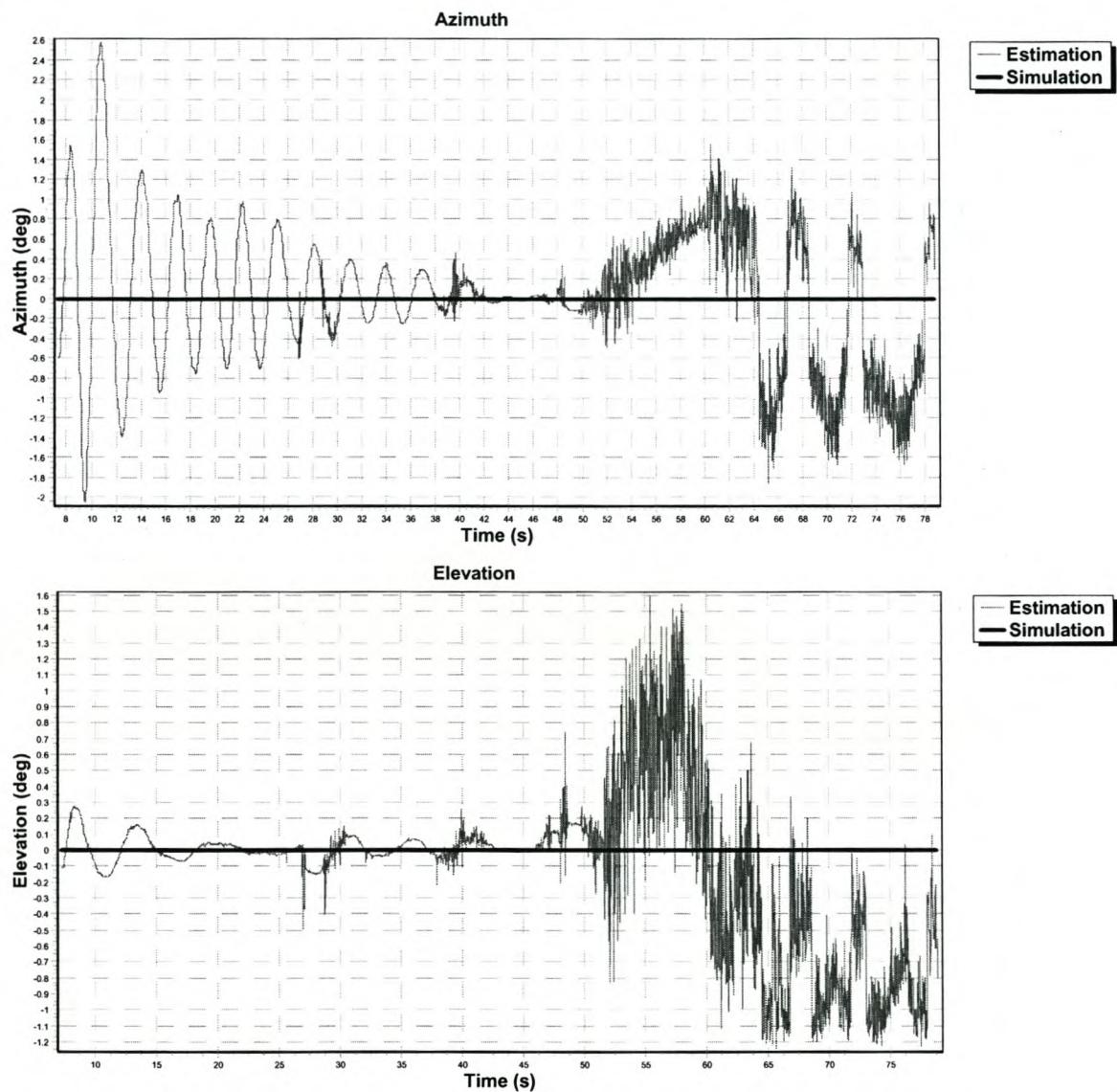


Figure 6.25: Comparison of simulated and measured angular tracking results for the receding target (Section 6.5.3).

Discussion

The range track test shows a fundamental problem with the simulator — it is too slow for this waveform. For the most part, the range and velocity plots have the correct shape, albeit stretched. The short PRI, and correspondingly short time per burst does not allow the simulator enough time to generate and update the information for the next burst. Tests using the same waveform pattern, but with a longer PRI ($396\ \mu\text{s}$, as opposed to $252\ \mu\text{s}$) reduced the buffer underruns from 49.4% to a much more reasonable 5.6%. The bottleneck is in fact the PC — it takes too long to do the real-time processing. The rate at which faster PCs become available reduces our concern of this problem.

The fact that the simulator is too slow can also be seen by the decorrelation between the doppler speed (shown by the estimations), and the range rate (dx/dt). The radar notes this and asserts the doppler decorrelation jamming strobe, for most of the simulation. The large difference between the two rates is also the cause of the discontinuities in the estimated speed (the estimator essentially unwraps the ambiguous doppler measurements by considering the range rate).

The underdamped response seen in the previous test is again evident, although the target movement significantly worsens the oscillations. The initial azimuth error is larger than the initial elevation error, resulting in the larger oscillations in the azimuth channel. After about 52 seconds, the radar essentially loses track of the target. The reason being that the target has entered a region of clutter (starting at 6500 m). The addition of a large amount of clutter to the target's small return results in a signal that is very poorly correlated with the transmitted pulse.

6.5.4 Cross-range target

Objective

This experiment evaluates the system's ability to track a target moving in both range and angle. The target's azimuth angle changes, while its elevation remains constant.

Protocol

1. Perform the common set up in Section 6.5.1.
2. Designate the target at a range of 3620 m, velocity of 0 m/s, azimuth of 236° and elevation of 0° .
3. Compare the target's simulated trajectory to the measured trajectory.

Results

The logged target position is compared to the simulated target position in Figures 6.26 and 6.27. The simulated trajectory reaches 3400 m after 31,33 s, while the measurement only

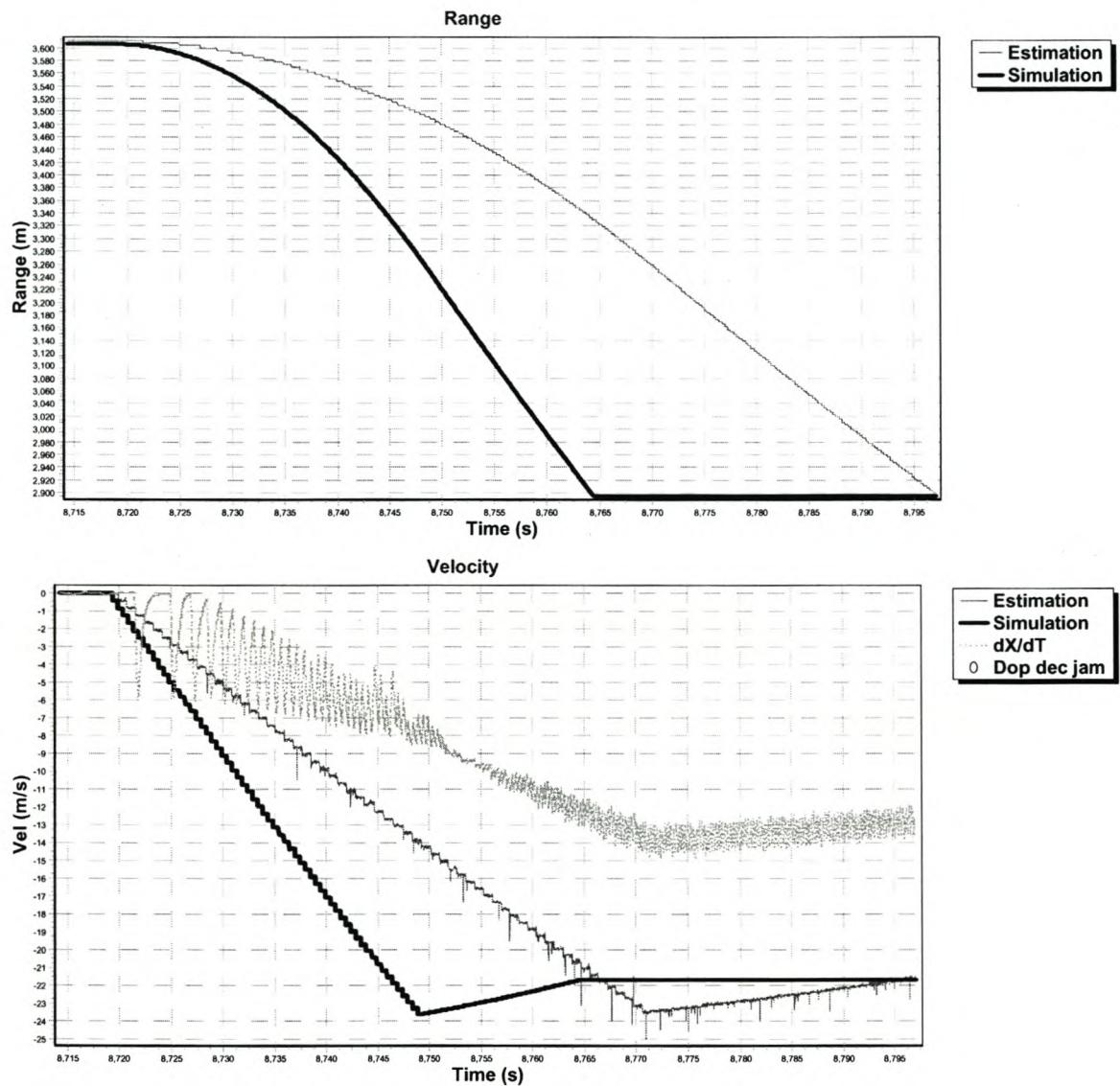


Figure 6.26: Comparison of simulated and measured range and velocity tracking results for the cross-range target (Section 6.5.3).

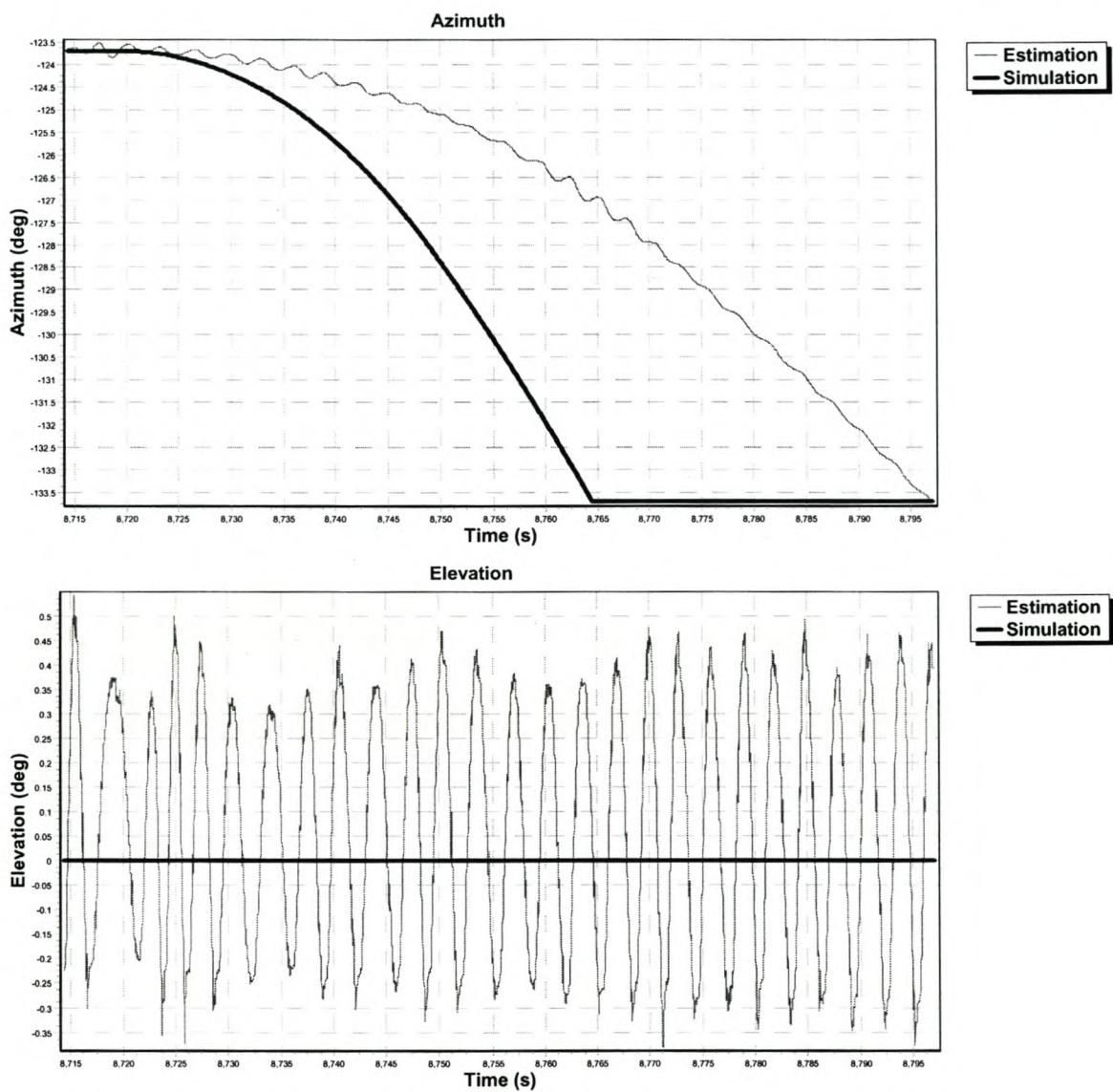


Figure 6.27: Comparison of simulated and measured angular tracking results for the cross-range target (Section 6.5.3).

reaches this range after 48,33 s. The simulation is thus $|31.33 - 48.33|/31.33 = 54\%$ slower than real-time.

Discussion

As with the receding target test, this test also shows the simulator's inability to keep up with real-time. In this test, the simulator is even slower than in the previous one (54% compared to 25%). When this test was performed, the scenario actually included more objects than in the previous test. More scatterering objects require more of the PC's processing time, thus slowing the simulator down. As mentioned previously, a faster PC or marginally longer PRI will overcome this problem.

Except for the time scaling factor, all the measurements correspond well to the simulated values. As with the previous test, the slight oscillatory nature of the range measurement is again visible in the velocity's range rate. The target's radial velocity is much lower than in the previous test, so the doppler speed remains quite close to the range rate — the doppler decorrelation strobe is never asserted. Unlike the previous test, there are no large jumps in the unambiguous doppler speed — this is also due to the closer correlation between the range rate and doppler speed.

In the velocity and azimuth plots, we see how the control loop converges on the nominal position. The elevation angle measurement exhibits an undamped response — the simulator is slower in this test, and the extra delay added to the elevation control loop's dynamics results in an unstable response. We note that the dynamics of the azimuth and elevation control loops must be different. This is reasonable, considering that the positioner uses different motors to drive the antenna in elevation and azimuth.

The test shows that the simulator is able to track a target moving through a large angle. However, the tracker's control loop just barely manages to remain locked onto the target because of the large delay added by the simulator.

6.6 Summary

The target return tests showed that the simulator was able to generate targets with the correct range, amplitude, phase and doppler speed. Both the amplitude and phase did fluctuate slightly, thus reducing the fidelity of the generated targets. This was attributed to the simplistic design used for the DAC board. The simulated target compared well to a real-world target, except for the lack of doppler scintillation.

Clutter was generated successfully — it was shown to be present at the correct range, and the amplitude had the expected range dependence. The clutter's constant phase during a burst was also verified, by showing that the clutter was centred in the zero doppler bin. Comparison to real sea clutter highlighted a major drawback of the simple clutter implementation — surface clutter invariably has a doppler shift.

The radar system was able to perform closed-loop tracking of simulated targets. However, a fundamental problem was observed — the simulator was too slow for the high PRF waveform used for testing. Either a faster PC, or a lower PRF would be needed to overcome this problem. The antenna tables' low resolution and the additional delay added to the radar's control loop, by the simulator, resulted in underdamped responses. Nevertheless, the radar measurements still converged on the correct target position.

Chapter 7

Conclusion

7.1 Research results

The need for a radar environment simulator

During the development of a radar system, a large amount of money is spent performing field trials. Much of a radar system's functionality can be tested using a radar environment simulator, so use of one goes a long way in reducing the number of field trials required. Other advantages of simulators include the ability to repeat scenarios exactly (allowing parameters in the radar system to be optimised more easily), and safe testing of dangerous scenarios (e.g. a missile attack).

The more accurate the simulator, the more field trials can be replaced with virtual tests. As discussed in Chapter 2, even state-of-the-art simulators need to make many simplifying assumptions in order to model the radar environment. Thus field trials will remain a necessary part of radar development for the near future, at least.

Project objectives

This project had four main objectives. The first was a theoretical study of the existing models for radar target and environmental returns. Chapter 2 examined the literature at length. In the same chapter, the pertinent models for a simple prototype simulator were selected, thus achieving the second objective. The prototype was not required to be a fully featured, high fidelity simulator, but rather served to prove the concept. The most important outcome of the prototype development, was an extensible software framework into which higher fidelity models could be added at a later stage. The simulator's chosen architecture performs off-line processing of the platform dynamics and propagation effects, with real-time pulse generation. This allows the benefits of real-time feedback from the radar tracker combined with the possibility of complex models.

The third objective was the functional analysis and conceptual design of the simulator — Chapter 3 detailed the design process. The core of the simulator consisted of two interrelated software units: the pre-processing engine and the real-time engine. The former performs the

complex modelling and analysis of the input scenario prior to the actual simulation run. During a simulation, the real-time engine selects the relevant pre-processed information according to the current boresight angle, applies the antenna pattern, and then streams the data to the waveform generator.

The fourth and final objective was the successful implementation of a prototype simulator. Chapter 4 discussed the software implementation — the majority of which was done in object-orientated C++. Interaction with the user was provided through a graphical user interface. In Chapter 5, the hardware implementation was described. The waveform generator consisted of an FPGA development board supplemented with a custom made three-channel DAC board. The output signal is the sum of two target return generators and the clutter generator.

In order to evaluate the successfulness of the prototype implementation, a number of tests were performed — they are documented in Chapter 6. The prototype was shown to performed satisfactorily for static targets and clutter. Once the target's started to move, a fundamental problem was observed — the simulator was too slow to keep up with the high PRF waveform used for testing. Unless a much faster PC is used, the current prototype is only useful at lower PRFs.

7.2 Further work

Software improvements

As the prototype simulator only implemented simple target and environmental models, a large amount of work can still be done to improve the simulator's fidelity. Refraction and multipath propagation should be implemented. A refraction-based ray tracing algorithm, combined with the calculation of the pattern propagation factor, could be used. The current software framework was designed with this approach in mind. Attenuation is simple to model, and should not prove a problem to implement as part of the ray tracing algorithm.

Only aircraft were modelled, and only with amplitude scintillation. A more realistic target model should include doppler scintillation, glint, and an attitude dependent mean radar cross-section. Other types of targets such as ships and ground vehicles could also be added.

A major limitation of the implemented surface clutter models is the lack of correlation properties. While these properties are difficult to realise, doing so would greatly increase the simulator's fidelity. The clutter generator firmware would need to be changed slightly as well (see below). Another limitation is the fact that the radar must remain static throughout the scenario. The software framework would require very little modification to overcome this problem; however, the pre-processing time and the amount of data generated would increase dramatically.

Much work still needs to be done concerning volume scatterers. The prototype did not implement any volume scatterers, but the framework does include the necessary objects.

Two applications of volume scatterers are: volume clutter (such as rain, or chaff clouds), and distributed targets (such as ships, which occupy a number of range cells). The attenuation of volume clutter should be considered by the ray tracing algorithm.

The user interface requires some work in order to allow scenarios to be set up and changed more easily. The current implementation only allows the object parameters to be modified, and not their trajectories. A three-dimensional view of the scenario would also be useful.

The real-time engine's code could be optimised further, to make the simulator usable with higher PRF waveforms. However, the rate at which faster PCs become available suggests that the optimisation would soon be unnecessary.

Hardware improvements

The CORDIC implementation is not as accurate as a comparable width lookup table (LUT) would be. The number of bits and iterations used by the CORDICs could be increased, but it would be simpler to use a LUT. If a bigger FPGA is not available, then some of the memory used for storing the pulse compression code's phase information could be used. The current implementation allows up to four different codes in a pattern; however, this could easily be reduced to one. The requirement is that the RSP's pattern always uses the same pulse code.

The clutter generator plays out the same samples for all pulses in a burst. In order to allow correlation and doppler properties to be controlled, the implementation should be modified so that different clutter samples are used for each pulse. If a larger FPGA is not available, memory for the extra per-pulse information could be gained by reducing the clutter range coverage.

The DAC board layout could be improved by separating the analogue and digital ground planes, and termination resistors should be added to the data lines. A four layer board would also help improve the signal fidelity.

7.3 Summary

The objectives were achieved, with the project culminating in the successful implementation of a prototype simulator. The core of the simulator is largely applicable to the majority of terrestrial radar systems, although the frequency dependence of the models needs to be considered. The simulator could be used with other pulse doppler radars with only minimal changes to the interfacing, and the waveform and antenna patterns. For continuous wave radars, the simulator (especially the waveform generator) would require substantial modifications.

In achieving the objectives, the envisioned contributions were also realised. Firstly, the state of the art in radar environment simulation and target generation was summarised. Secondly, a generic, largely software defined, simulator was developed. Thirdly, inexpensive, repeatable testing of a specific radar system was shown to be feasible.

Bibliography

- [1] Altera, "Binary Numbering Systems."
http://www.altera.com/literature/an/an083_01.pdf. April 1997.
- [2] Altera, "Nios Development Board, Stratix Edition: Data sheet."
http://www.altera.com/literature/ds/ds_nios_board_stratix_1s10.pdf. January 2003.
- [3] Altera, "Plugs Ethernet Library Reference Manual."
http://www.altera.com/literature/manual/mnl_plugs.pdf. July 2003.
- [4] Altera, "SOPC Builder: Data sheet."
http://www.altera.com/literature/ds/ds_sopc.pdf. January 2003.
- [5] ANDRAKA, R., "A survey of CORDIC algorithms for FPGAs." in *Proceedings of the ACM/SIGDA sixth international symposium on field programmable gate arrays*, pp. 191–200, February 1998.
- [6] ANDRAKA, R. J. and PHELPS, R. M., "An FPGA based processor yields a real-time high fidelity radar environment simulator."
<http://www.andraka.com/files/mapld.pdf>. September 1998.
- [7] BAIR, G. and JOHNSTON, D., "Advances in realtime radar simulation." in *IEEE Region 5 Conference, 1988: 'Spanning the Peaks of Electrotechnology'*, pp. 23–31, March 1988.
- [8] BANGKUI, F., ERKE, M., YUEQIU, H., and YIFANG, Z., "A real-time radar video signal simulator." in *3rd International Conference on Signal Processing*, pp. 1102–1105, October 1996.
- [9] BARRIOS, A. E. and PATTERSON, W. L., "Advanced Propagation Model (APM) Ver 1.3.1 Computer Software Configuration Item (CSCI)." <http://www.spawar.navy.mil/sti/publications/pubs/td/3145/td3145.pdf>. August 2002.
- [10] BERIZZI, F. and MESE, E., "Fractal theory of sea scattering." in *Proceedings - CIE International Conference of Radar*, pp. 661–665, October 1996.

- [11] BETTINI, G., BICCI, A., CIONI, R., COSTA, F., and DE PALO, M., "Cluster Model: A New Procedure for the Computation of E. M. Scattering From Radar Targets." in *IEEE International Radar Conference*, pp. 351–356, May 1996.
- [12] BLACKNELL, D., "Comparison of parameter estimators for K-distribution." in *IEE Proceedings - Radar, Sonar and Navigation*, no. 1, pp. 45–52, February 1994.
- [13] BLACKNELL, D., "New method for the simulation of correlated K-distributed clutter." in *IEE Proceedings - Radar, Sonar and Navigation*, no. 1, pp. 53–58, February 1994.
- [14] Borland Software, "C++ Builder." <http://www.borland.com/cbuilder/>. February 2000.
- [15] BOUDOURIS, G., "On the Index of Refraction of Air, the Absorption and Dispersion of Centimeter Waves by Gases." *Journal of Research of the National Bureau of Standards*, 1963, Vol. 67D (Radio Propagation), No. 6, pp. 631–684.
- [16] BRANSON, J., WOODING, S., and DAWBER, W., "Modelling of the littoral environment for real-time radar performance assessment." in *IEE Conference Proceedings - RADAR 2002*, pp. 1102–1105, 2002.
- [17] CARLSON, G. E., *Signal and Linear System Analysis*. Second edition. New York: John Wiley & Sons, 1998.
- [18] CHARRIER, C. and DELISLE, G., "Determination of characteristic features of a RCS using Wavelet Analysis." in *Antennas and Propagation Society, IEEE International Symposium*, vol. 4, pp. 2166–2169, 2000.
- [19] CHEN, G., HUANG, P., and YIN, H., "A statistical model of radar target glint based on discrete differenced gaussian noise." in *Antennas and Propagation Society International Symposium, AP-S. Digest*, vol. 2, pp. 1418–1421, July 1996.
- [20] CHOONG, P., "Composite terrain clutter modelling." in *Proceedings of the Fifth International Symposium on Signal Processing and its Applications (ISSPA)*, pp. 515–518, August 1999.
- [21] CONTE, E., LONGO, M., and LOPS, M., "Modelling and simulation of non-Rayleigh radar clutter." in *IEE Proceedings F*, no. 2, pp. 121–130, April 1991.
- [22] CRONJÉ, J., "Software Architecture Design of a Software-Defined Radio System." Master's thesis, University of Stellenbosch, 2004.
- [23] DAVIDSON, G., GRIFFITHS, H. D., and ABLETT, S., "Statistical analysis of high resolution land clutter." in *IEEE Radar Conference*, pp. 434–438, October 2002.

BIBLIOGRAPHY

147

- [24] DAVIDSON, G., "Radar Toolbox." <http://www.radarworks.com>. March 2003.
- [25] D'ELIA, U. F., DEL GAUDIO, M. G., and PISTOIA, D., "Electromagnetic Modelling of Targets in the Near Zone and Comparison with Experimental Results." in *IEE Radar Conference*, pp. 463–467, October 1997.
- [26] DENNY, M., "Simulating sea clutter via the compound-k distribution." in *IEE Colloquium on Radar System Modelling (Ref. No. 1998/459)*, pp. 16/1–16/5, October 1998.
- [27] DENNY, W. M., "K-distributed Sea Clutter: Performance Predictions Made Easy." in *IEE Radar Conference*, pp. 209–213, October 1997.
- [28] Digital Design Solutions, Inc., "CORDIC – COordinate Rotation DIgital Computer." <http://www.digital-designs.com/CORDIC.htm>. June 2004.
- [29] DINAPOLI, F. R. and DEAVENPORT, R. L., "Numerical methods of underwater acoustic propagation." in *Ocean Acoustics* (DESANTO, J. A. (Ed.)), New York: Springer-Verlag, 1977.
- [30] DOCKERY, G. D., "Modeling Electromagnetic Wave Propagation in the Troposphere Using the Parabolic Equation." *IEEE Transactions on Antennas and Propagation*, October 1988, Vol. 36, No. 10, pp. 1464–1470.
- [31] ECKEL, B., *Thinking in C++*. Upper Saddle River: Prentice Hall, 2000.
- [32] FABBRO, V., GUILLET, N., and COMBES, P., "Innovative improvements of the parabolic wave equation method for radiowave propagation modeling." in *Twelfth International Conference on Antennas and Propagation (ICAP 2003)*, vol. 2, pp. 654–656, March 2003.
- [33] FARINA, A., GINI, F., GRECO, M., and VERRAZZANI, L., "High resolution sea clutter data: statistical analysis of recorded live data." in *IEE Proceedings - Radar, Sonar and Navigation*, vol. 144, pp. 121–130, June 1997.
- [34] FEATHERS, M., "CppUnit." <http://cppunit.sourceforge.net/cgi-bin/moin.cgi/FrontPage>. December 1999.
- [35] FOCK, V. A., "Solution of the problem of propagation of electromagnetic waves along the earth's surface by method of parabolic equations." *USSR Journal of Physics*, 1946, Vol. 10, No. 1.
- [36] Free Software Foundation, "CVS - Concurrent Versions System." <http://www.gnu.org/software/cvs/>. July 2000.

- [37] GYU-HWAN, H., KYUNG-BIN, B., JONG-MIN, P., SUNG-HYUCK, Y., and JONG-SE, P., "Sea clutter measurement on the Yellow Sea using an instrumentation radar." in *Proceedings - 5th International Symposium on Antennas, Propagation and EM Theory. ISAPE 2000*, pp. 33–36, 2000.
- [38] HAYKIN, S. and PUTHUSERYPADY, S., "Chaotic dynamics of sea clutter: an experimental study." in *IEE Radar Conference*, pp. 75–79, October 1997.
- [39] HITNEY, H. V., "Hybrid ray optics and parabolic equation methods for radar propagation modelling." in *Radar 92*, no. 365, (Brighton, UK), pp. 58–61, IEE Conference Publication, October 1992.
- [40] HITNEY, H. V., "A Practical Tropospheric Scatter Model Using the Parabolic Equation." *IEEE Transactions on Antennas and Propagation*, July 1993, Vol. 41, No. 7, pp. 905–909.
- [41] HOLLANDS, P., "The use of radar environment simulation for operator training and EW system test and evaluation." in *Electronic Warfare Systems, IEE Colloquium on*, pp. 6/1–6/5, January 1991.
- [42] HOWER, C. Z., "Internet Direct." <http://www.nevrona.com/indy>. October 2002.
- [43] HUGHES, E. J. and LEYLAND, M., "Radar Cross Section Model Optimisation Using Genetic Algorithms." in *IEE Radar Conference*, pp. 458–462, October 1997.
- [44] JANOWIAK, T., "A simulation for combat systems development and acceptance testing." in *Proceedings - Winter Simulation Conference, 1990*, pp. 210–213, December 1990.
- [45] JAO, J., "Amplitude distribution of composite terrain radar clutter and the K-Distribution." *IEEE Transactions on Antennas and Propagation*, October 1984, Vol. 32, No. 10, pp. 1049–1062.
- [46] JOHNSON, G., "Constructions of particular random processes." in *Proceedings of the IEEE*, no. 2, pp. 270–285, February 1994.
- [47] JOHNSTON, S. L., "Target Model Pitfalls (Illness, Diagnosis and Prescription)." *IEEE Transactions on Aerospace and Electronic Systems*, April 1997, Vol. 33, No. 2, pp. 715–720.
- [48] KASHYAP, S., STANIER, J., PAINCHAUD, G., and LOUIE, A., "Radar response of missile-shaped targets." in *Antennas and Propagation Society International Symposium, AP-S. Digest*, vol. 4, pp. 1910–1913, June 1995.
- [49] KERR, D. E., *Propagation of short radio waves*. New York: McGraw-Hill, 1951.

- [50] KINGSLEY, S. and QUEGAN, S., *Understanding Radar Systems*. Mendham: SciTech, 1999.
- [51] KRAUS, J. D. and FLEISCH, D. A., *Electromagnetics with applications*. International edition. Singapore: WCB/McGraw-Hill, 1999.
- [52] LAMONT-SMITH, T. and WALKER, D., "A Comparison of EM Scattering Results and Radar Sea Clutter." in *IEE Radar Conference*, pp. 439–443, October 2002.
- [53] LEIGHTY, B. D. and PERKINS, M. D., "Fidelity aspects of radar target and environment simulation." in *Telesystems Conference, NTC '91*, vol. 1, pp. 247–256, 1991.
- [54] LEONOV, S. A. and LEONOV, A. I., *Handbook of computer simulation in radio engineering, communications, and radar*. Boston: Artech House, 2001.
- [55] MAHAFZA, B. *et al.*, "Real-time radar signal simulation for the ground based radar for national missile defense." in *Proceedings of the IEEE Radar Conference. RADARCON 98*, pp. 62–67, May 1998.
- [56] MARCUS, S., "The dynamics and radar cross section density of chaff clouds." in *Proceedings of the IEEE Radar Conference*, pp. 278–285, May 2002.
- [57] MEIKLE, H., *Modern Radar Systems*. Boston: Artech House, 2001.
- [58] METZ, R. and PHELPS, R., "AECM, an AF tool to generate I/Q for simulators or real radar processors." in *Proceedings of the IEEE Aerospace and Electronics Conference. NAECON 1994*, pp. 97–104, May 1994.
- [59] MOBSBY, C., NEWTON, D., PYMM, M., and YOUERN, K., "Scenario modelling [naval engagement, radar].". in *IEE Colloquium on Computer Modelling and Simulation of Radar Systems*, pp. 2/1–2/4, February 1993.
- [60] MONEY, D., BRANSON, J., HOOKER, M., and MABOGUNJE, A., "Radar littoral environment predictions and measurements." in *Radar 97 (Conf. Publ. No. 449)*, pp. 164–168, October 1997.
- [61] NOGA, J. L., *Bayesian State-Space Modelling of Spatio-Temporal Non-Gaussian Radar Returns*. PhD thesis, University of Cambridge, <http://www-sigproc.eng.cam.ac.uk/publications/archive/jln.phdthesis.pdf>, 1998.
- [62] NOHARA, T. J. and HAYKIN, S., "AR-Based Growler Detection in Sea Clutter." *IEEE Transactions on Signal Processing*, March 1993, Vol. 41, No. 3, pp. 1259–1271.

- [63] PALOMINO, A. and SCHMITZ, J., "Model-based RF ground moving target signature synthesis and analysis tool." in *Proceedings - IEEE National Aerospace and Electronics Conference, NAECON*, pp. 158–163, 2000.
- [64] PATTERSON, W. L. *et al.*, "Engineer's Refractive Effects Prediction System Advanced Propagation Model — Version 3.0."
<http://www.spawar.navy.mil/sti/publications/pubs/td/2648/td2648.pdf>. May 1994.
- [65] PHU, P., ADLER, E., INNOCENTI, R., and PAOLELLA, A., "A test target generator for wideband pulsed Doppler radars." in *IEEE Conference Proceedings - Microwave Systems Conference. NTC '95*, pp. 109–111, May 1995.
- [66] PINO, M., BURKHOLDER, R., RODRIGUEZ, J., and OBELLEIRO, F., "Monte Carlo study and statistical description of the radar scattering from 2D ships on rough sea surfaces." in *Antennas and Propagation Society, IEEE International Symposium*, vol. 4, pp. 352–355, 2001.
- [67] POPOV, A., KOPEIKIN, V., and LANDSTORFER, F., "Full-wave simulation of overland radar pulse propagation." *Electronics Letters*, March 2003, No. 6, pp. 550–552.
- [68] POSTEL, J. and REYNOLDS, J., "RFC 854: TELNET Protocol Specification." <http://www.rfc-editor.org/rfc/rfc854.txt>. May 1983.
- [69] Protel International Ltd., "Protel 99 SE." <http://www.protel.com>. 1999.
- [70] REILLY, J. and LIN, C., "A radar land clutter model and its verification." in *International Geoscience and Remote Sensing Symposium. IGARSS. 'Surface and Atmospheric Remote Sensing: Technologies, Data Analysis and Interpretation'*, pp. 2319–2321, August 1994.
- [71] SANDHU, G., "A real-time statistical polarimetric target model." *IEEE Transactions on Aerospace and Electronic Systems*, January 1988, Vol. 24, No. 1, pp. 51–67.
- [72] SARKAR, B., KAKATKAR, S., and AGARWAL, A., "Target movement simulation for testing monopulse radar." in *Antennas and Propagation Society International Symposium. AP-S. Digest*, vol. 4, pp. 1822–1825, June 1995.
- [73] SARNO, G., "Modelling of radar clutter." in *IEE Colloquium on Computer Modelling and Simulation of Radar Systems*, pp. 6/1–6/9, February 1993.
- [74] SHIRMAN, G. D., GORSHKOV, S. A., LESHENKO, S. P., and ORLENKO, V. M., "Aerial Target Backscattering Simulation and Study of Radar Recognition, Detection and Tracking." in *IEEE International Radar Conference*, pp. 521–526, May 2000.

- [75] SHNIDMAN, D., "Generalized radar clutter model." *IEEE Transactions on Aerospace and Electronic Systems*, July 1999, No. 3, pp. 857–865.
- [76] SIMPSON, S. H. W. and GALLOWAY, P. E. R., "Complex target signature generation." in *IEEE Signal Processing Conference*, pp. 485–488, October 1996.
- [77] SKOLNIK, M. I., *Introduction to Radar Systems*. International student edition. Tokyo: McGraw-Hill, 1962.
- [78] SKOLNIK, M. I., *Radar Handbook*. New York: McGraw-Hill, 1970.
- [79] SMITH, E. K. and WEINTRAUB, S., "The Constants in the Equation for Atmospheric Refractive Index at Radio Frequencies." in *Proceedings of the IRE*, vol. 41, pp. 1035–1037, August 1953.
- [80] SWERLING, P., "Probability of detection for fluctuating targets." *IEEE Transactions on Information Theory*, April 1960, Vol. 6, No. 2, pp. 269–308.
- [81] SWERLING, P., "Radar probability of detection for some additional fluctuating target cases." *IEEE Transactions on Aerospace and Electronic Systems*, April 1997, Vol. 33, No. 2, pp. 698–709.
- [82] TAPPERT, F. D., "The parabolic approximation method." in *Wave Propagation and Underwater Acoustics* (KELLER, J. B. and PAPADAKIS, J. S. (Eds)), New York: Springer-Verlag, 1977.
- [83] The MathWorks, "MATLAB 6.5." <http://www.mathworks.com>. 2002.
- [84] THEWS, E. R. and DOCKERY, G. D., "Scattering and Propagation Impacts on Shipboard Radar Systems." in *IEEE International Radar Conference*, pp. 401–408, May 1990.
- [85] THOMSON, A. and RISEBOROUGH, E., "Evaluation of a weather clutter simulation." in *IEE Proceedings - Radar, Sonar and Navigation*, vol. 148, pp. 119–129, June 2001.
- [86] TOUGH, R. J. A. and WARD, K. D., "Diffusive and related models of some properties of sea clutter." *Journal of Physics: Condensed Matter*, 2002, Vol. 14, No. 33, No. 33, pp. 7737–7748.
- [87] UNSWORTH, C., COWPER, M., MCCLAUGHLIN, S., and MULGREW, B., "The dynamics and radar cross section density of chaff clouds." in *IEE Proceedings - Radar, Sonar and Navigation*, June 2002.
- [88] VAN HEESCH, D., "Doxygen documentation system." <http://www.doxygen.org>. February 2004.

BIBLIOGRAPHY

152

- [89] VAN ROOYEN, G.-J., "An analysis of Quadrature-Baseband Direct Digital Synthesis." Master's thesis, University of Stellenbosch, 2000.
- [90] VAN TREES, H. L., *Detection, Estimation and Modulation Theory: Part III*. New York: McGraw-Hill, 1968–1971.
- [91] Venturcom Inc., "Hard Real-Time with Venturcom RTX on Microsoft Windows XP and Windows XP Embedded." <http://www.windowsforddevices.com/articles/AT2503923807.html>. October 2003.
- [92] WARD, K., BAKER, C., and WATTS, S., "Maritime surveillance radar. Part 1: Radar scattering from the ocean surface." in *IEE Proceedings - Radar and Signal Processing*, vol. 137, pp. 51–62, April 1990.
- [93] WARNER, J. X. and ELLINGSON, R. G., "A New Narrowband Radiation Model for Water Vapor Absorption." *Journal of Atmospheric Sciences*, May 2000, Vol. 57, No. 10, pp. 1481–1496.
- [94] WATTS, S., "The modelling of sea clutter and its application to the specification and measurement of radar performance." in *Proceedings - CIE International Conference on Radar*, pp. 431–435, 2001.
- [95] XIE, N., LEUNG, H., and CHAN, H., "A multiple-model prediction approach for sea clutter modeling." *IEEE Transactions on Geoscience and Remote Sensing*, June 2003, Vol. 41, No. 6, pp. 1491–1502.
- [96] XU, X. and HUANG, P., "A new RCS statistical model of radar targets." *IEEE Transactions on Aerospace and Electronic Systems*, April 1997, Vol. 33, No. 2, pp. 710–714.
- [97] ZWART, J. P., KRØSE, B., and GELSEMA, S., "Aircraft Classification from Estimated Models of Radar Scattering." www.science.uva.nl/pub/computer-systems/aut-sys/reports/IAS-UVA-03-02.pdf. January 2003.

Appendix A

Frequency band letters

In this paper, we use the United States frequency band letters. Table A.1 compares these to those used by the United States.

Table A.1: *British and United States radar frequency bands, from [57, pp. 553].*

British band letters		United States band letters	
Frequency [GHz]	Band	Frequency [GHz]	Band
		0,42 – 0,45	P
1 – 2	L	1 – 2	L
2 – 4	S	2 – 4	S
4 – 8	C	4 – 8	C
8 – 12	X	8 – 12	X
12 – 18	J	12 – 18	K _u
18 – 26	K	18 – 27	K
26 – 40	Q	27 – 40	K _a
40 – 60	V	40 – 75	V
60 – 90	O	75 – 110	W

Appendix B

Detailed models

B.1 Introduction

This appendix includes details of the models discussed in Chapter 2, where available. In order to simplify referencing between the two chapters, the section headings are the same in both.

B.2 Radar system

There are no additional details for this section.

B.3 Atmospheric effects

B.3.1 Refraction

There are no additional details for this section.

B.3.2 Retardation

There are no additional details for this section.

B.3.3 Attenuation

Gaseous absorption

Constant attenuation model. The CCIR model gives the water vapour absorption loss L_{wv} as ([64], p. 125)

$$L_{wv} = r\alpha_{wv} \tag{B.1}$$

The water vapour attenuation rate α_{wv} is given, in dB/km, as

$$\alpha_{wv} = (0,05 + 0,0021H_a + \alpha_{wv1} + \alpha_{wv2} + \alpha_{wv3}) f^2 H_a 10^{-10} \tag{B.2}$$

with H_a the absolute humidity, f the RF frequency [MHz], and

$$\alpha_{wv1} = \frac{3,6}{(0,001f - 22,2)^2 + 8,5} \quad (\text{B.3})$$

$$\alpha_{wv2} = \frac{10,6}{(0,001f - 183,3)^2 + 9} \quad (\text{B.4})$$

$$\alpha_{wv1} = \frac{8,9}{(0,001f - 325,4)^2 + 26,3} \quad (\text{B.5})$$

The losses with this model are negligible below about 10 GHz. From Figure 2.4 the loss due to oxygen at these frequencies is less than 0,01 dB/km, and is thus not included. At frequencies above 22 GHz, oxygen absorption is dominant and should be included.

Height dependent models. A model attributed to Van Vleck, Bean and Abbott is given in [78, p. 24-13]. The oxygen absorption in decibels per kilometre, γ_1 , at a temperature of 293°K and standard atmospheric pressure is

$$\gamma_1 = \frac{0,34}{\lambda^2} \left[\frac{\Delta\nu_1}{1/\lambda^2 + \Delta\nu_1^2} + \frac{\Delta\nu_2}{(2+1/\lambda)^2 + \Delta\nu_2^2} + \frac{\Delta\nu_2}{(2-1/\lambda)^2 + \Delta\nu_2^2} \right] \quad (\text{B.6})$$

with $\Delta\nu_1$ and $\Delta\nu_2$ line-width factors in cm^{-1} and λ the RF wavelength. The water vapour absorption, due to the peak near 22 MHz (also known as the 1,35-cm line), at 293°K, is given as

$$\frac{\gamma_2}{H_a} = \frac{3,5 \cdot 10^{-3}}{\lambda^2} \left[\frac{\Delta\nu_3}{(1/\lambda - 1/1,35)^2 + \Delta\nu_3^2} + \frac{\Delta\nu_3}{(1/\lambda + 1/1,35)^2 + \Delta\nu_3^2} \right] \quad (\text{B.7})$$

where $\Delta\nu_3$ is a line-width factor. The line-width factors have been empirically determined and are available in [78, p. 24-14]. Factors to account for the temperature and pressure dependence are also listed.

A simplified model by Barton, that uses an exponential model, is given by Leonov and Leonov [54, pp. 241–248]. First an effective elevation angle is calculated from

$$\theta_{eff} = \theta + \frac{2,5 \cdot 10^{-4}}{\theta + 0,028} \quad [\text{rad}], \quad (\text{B.8})$$

where θ is the radar elevation angle [rad]. Next an effective range must be determined:

$$R_{eff} = \frac{3}{\sin(\theta_{eff})} \quad [\text{km}]. \quad (\text{B.9})$$

The effective range is then used as a scaling factor in the final equation:

$$L = k_\alpha R_{eff} \left[1 - e^{\left(-\frac{R_{km}}{R_{eff}} \right)} \right] \quad [\text{dB}], \quad (\text{B.10})$$

- with L = total two-way attenuation, [dB]
- k_α = attenuation coefficient, [dB/km]
- R_{km} = target range, [km].

Table B.1: Two-way atmospheric attenuation coefficients for Barton model, from [54, p. 242].

Frequency [GHz]	k_α clear air	k_α/R_r rain	k_α/R_r snow
0,4	0.100	0.000	0.000
1,3	0.012	0.0003	0.0003
3,0	0.015	0.0013	0.0013
5,5	0.017	0.008	0.008
10	0.024	0.037	0.002
15	0.055	0.083	0.004
22	0.030	0.230	0.008
35	0.140	0.570	0.015
60	35.00	1.300	0.030
90	0.800	2.000	0.060
140	1.000	2.300	0.060
240	15.00	2.200	0.080

Values for k_α are listed in Table B.1. Note that the rain and snow values must be multiplied by the rainfall rate R_r [mm/h]. The precipitation is assumed to be present all along the ray's path. Interestingly, there is no 22 GHz water vapour peak, as would be expected from the Van Vleck *et al.* model, instead there is one at 35 GHz. The 60 GHz oxygen peak is obvious. Leonov and Leonov advise that the attenuation at intermediate frequencies be obtained by logarithmic interpolation.

Weather effects

Attenuation by rain. A model by Ryde, discussed in [78, p. 24-23], gives the attenuation in decibels per kilometre K_R as

$$K_R = K \int_0^r [R_r(r)]^\alpha dr \quad (\text{B.11})$$

with K a function of frequency; $R_r(r)$ the rainfall rate along the propagation path r ; and α a function of frequency, taken as unity. Values for K_R are tabulated in [78, p. 24-24], for frequencies from 3 GHz to 100 GHz. The attenuation is proportional to the temperature — the same reference also includes tabulated correction factors. As an example, at 9,4 GHz and 30°C, the attenuation due to rain at 0,25 mm/hour is $0,0019 \cdot 0,79 = 0,015$ dB/km; and at 2,5 mm/hour, the attenuation increases to $0,0317 \cdot 0,82 = 0,026$ dB/km. The first factor in each equation is the attenuation at 18°C, which is then multiplied by the corresponding correction factor.

The tabulated values for the above model assume that the rainfall rate is constant all along the path of the ray. A more appropriate model for a widespread storm is one in which the rate slows exponentially as [78, p. 24-26],

$$R_r = R_{r0} e^{-ch^2} \quad (\text{B.12})$$

where R_{r0} is the rainfall rate at the surface, c is a constant taken as 0, 2, and h is the altitude.

Attenuation by hail and snow. Gunn and East's model calculates the attenuation due to the wet snow K_S , and is given in [57] as

$$K_S = \frac{0,00349 R_r^{1,6}}{\lambda^4} + \frac{0,0022 R_r}{\lambda} \quad [\text{dB/km}] \quad (\text{B.13})$$

with λ the wavelength in centimetres, and R_r the rainfall rate [mm/h]

Attenuation by clouds. A model for the attenuation by cloud droplets is given in [78, p. 24-22] as

$$K_C = K_1 M \quad (\text{B.14})$$

with K_C in dB/km, K_1 is the attenuation coefficient in dB/km/g/m³, and M is the liquid water content in g/m³. Expressions exist for M and K_1 , but they are generally not required. M is usually between 1 and 2,5 g/m³ and K_1 is tabulated for frequencies between 9,4 and 33 GHz for both water and ice clouds [78, p. 24-22]. Over this frequency range, ice clouds have an attenuation coefficient of less than 0,009 and are neglected in practice.

Attenuation by fog. The approach to modelling fog is simpler than that used for clouds. The attenuation in decibels per kilometre for frequencies between 3 and 24 GHz was derived by Saxton and Hopkins and is given in [78, p. 24-28]. Values are listed for visibilities from 30 m to 300 m, at 0°C — these values must be multiplied by 0,6 and 0,4 for temperatures of 15°C and 25°C, respectively.

Another model, by Goldstein [49, 57] gives the attenuation K_F as

$$K_F = 4,97 \cdot 10^{-4} M f^2 \quad [\text{dB/km}] \quad (\text{B.15})$$

with M the liquid water content [g/m³], and f the operating frequency in GHz. Equation (B.15) is for a temperature of 18°C — it must be doubled near 2°C, and multiplied by 0,7 for temperatures around 30°C. Table B.2 shows some values of M .

Table B.2: Values for the water content M in fog, in terms of the visibility

Visibility [m]	M [g/m ³]
30	2,3
122	0,32
610	0,032

B.4 Surface reflections and multipath

B.4.1 Modelling

Reflection coefficients. A model reported by Leonov and Leonov [54, pp. 184–186], splits the calculation of the reflection coefficient into three parts. The first is the complex Fresnel reflection coefficient $\rho_0 \angle \alpha$, second is the specular scattering coefficient ρ_s , and third is the vegetation coefficient ρ_v . The last two only affect the amplitude of the reflected ray. The three coefficients are combined to obtain the complex reflection coefficient

$$\Gamma = \rho_0 \rho_s \rho_v e^{-j\alpha}. \quad (\text{B.16})$$

In order to calculate the complex Fresnel reflection coefficient, the complex dielectric constant ϵ_c must be found. It is given as [57, p. 172], [54, p. 185]

$$\epsilon_c = \epsilon_r - j 60\sigma_e \lambda \quad (\text{B.17})$$

with ϵ_r = relative permittivity of the reflecting surface, []

σ_e = conductivity of the surface layer, [mho/m]

λ = RF wavelength, [m].

Table B.3 lists typical frequency dependent values for ϵ_r and σ_e .

The complex Fresnel reflection coefficient Γ_F is polarisation dependent. For horizontal polarisation, it is given as [57, p. 172], [54, p. 185]

$$\Gamma_{F,H} = \frac{\sin(\Psi) - \sqrt{\epsilon_c - \cos^2(\Psi)}}{\sin(\Psi) + \sqrt{\epsilon_c - \cos^2(\Psi)}}, \quad (\text{B.18})$$

and for vertical polarisation as

$$\Gamma_{F,V} = \frac{\epsilon_c \sin(\Psi) - \sqrt{\epsilon_c - \cos^2(\Psi)}}{\epsilon_c \sin(\Psi) + \sqrt{\epsilon_c - \cos^2(\Psi)}}. \quad (\text{B.19})$$

The specular reflection coefficient ρ_s accounts for the loss of energy due to a rough surface. The model, attributed to Ament, is given in [54, p. 185] as

$$\rho_s = e^{-2(2\pi \frac{\sigma_h}{\lambda} \sin \Psi)^2}, \quad (\text{B.20})$$

where σ_h is the RMS roughness of the surface [m].

Table B.3: Relative permittivity ϵ_r and conductivity σ_e , from [54, p. 184].

Surface	Relative permittivity	Conductivity [mho/m]	Frequency range [GHz]
Good soil (wet)	25	0,02	
Average soil	15	0,005	
Bad soil (dry)	3	0,001	
Salt water	80 80 – 7,33($f - 1,5$) 69 – 2,43($f - 3,0$)	4,3 4,3 + 1,48($f - 1,5$) 6,52 + 1,314($f - 3,0$)	$f \leq 1,5$ $1,5 < f \leq 3,0$ $3,0 < f \leq 10$
Fresh water	80 80 – 7,33($f - 1,5$) 69 – 2,43($f - 3,0$)	1,0 1,0 + 1,06($f - 1,5$) 1,0 + 1,06($f - 1,5$)	$f \leq 1,5$ $1,5 < f \leq 3,0$ $3,0 < f \leq 10$
Snow or ice	3,2 3,2	$5,7 \cdot 10^{-5}$ $5,7 \cdot 10^{-5} +$ $6,79 \cdot 10^{-8}(f - 2,0)$	$f \leq 2,0$ $2,0 < f \leq 10$

The vegetation coefficient ρ_v , which accounts for the additional losses due to the vegetation, is given as [54, p. 186]

$$\rho_v = e^{-\frac{K}{\lambda} \sin \Psi}, \quad (\text{B.21})$$

where K is a constant dependent on the type of vegetation. For thin grass, $K = 1$; for dense weeds or brush, $K = 3$; and for dense trees, $K = 10$.

Flat earth model

The flat earth model applies only to the interference region. The geometry is shown in Figure 2.6, p. 21.

Two criteria must be met in order to use the flat earth approximation — these were given on p. 20. The result of these criteria is that the amplitude of F_c , eq. (2.11) and eq. (4.2), can be simplified to [78, p. 2-35]

$$F_{fe} = f_d \sqrt{1 + \left(\frac{f_r}{f_d}\rho\right)^2 + 2\frac{f_r}{f_d}\rho \cos(\gamma)}. \quad (\text{B.22})$$

The path length difference between the direct and reflected paths is given as

$$\delta = R_1 + R_2 - R \quad [\text{m}]. \quad (\text{B.23})$$

Using the first approximation and simple geometry, δ can be calculated as [78, p. 2-37]

$$\delta = 2h_r \sin \theta_e \quad [\text{m}]. \quad (\text{B.24})$$

The total phase difference between the two paths γ is then determined by adding the path length phase difference to the phase of the complex reflection coefficient

$$\gamma = \frac{2\pi\delta}{\lambda} + \alpha \quad [\text{rad}] . \quad (\text{B.25})$$

Ray optics (spherical earth)

The model we review is given by Leonov and Leonov [54, pp. 180–190]. The model assumptions and geometry are given on p. 21.

Path difference and propagation region. In order to calculate the ground range from the radar site to the point of reflection, d_r , two coefficients are needed. They are

$$p_c = \sqrt{\frac{4a_e(h_r + h_t) + R^2}{3}} , \quad (\text{B.26})$$

and

$$\phi_c = \arccos \left[\frac{2a_e(h_r + h_t)R}{p_c^3} \right] . \quad (\text{B.27})$$

The ground range can then be calculated from

$$d_r = \frac{R}{2} - p_c \cos \left(\frac{\phi_c + \pi}{3} \right) \quad [\text{m}] . \quad (\text{B.28})$$

The heights of the radar and target above a plane tangent to the point of reflection, h_r' and h_t' respectively, are calculated from

$$h_r' = h_r - \frac{d_r^2}{2a_e} \quad [\text{m}] , \quad (\text{B.29})$$

and

$$h_t' = \frac{(R - d_r)h_r'}{d_r} \quad [\text{m}] . \quad (\text{B.30})$$

The path length difference between the reflected and direct rays is then

$$\delta_0 = \frac{2h_r'h_t'}{R} \quad [\text{m}] . \quad (\text{B.31})$$

Finally, the region of propagation can be determined:

- If $h_r' > 0$ then, the radar horizon is not in the way, so
 - if $\delta_0 > \lambda/8$, then use the interference region model, otherwise
 - $\delta_0 \leq \lambda/8$, so use the intermediate region model.
- If $h_r' \leq 0$ then, the radar horizon is in the way, so use the diffraction region model.

Interference region. The reflection coefficients are needed and can be calculated as for the flat earth model, using equations B.17–B.21. Next the divergence factor, that accounts for the spherical earth is calculated:

$$D = \sqrt{\frac{1}{3} \left(1 + \frac{2u}{\sqrt{u^2 + 3}} \right)}, \quad (\text{B.32})$$

where u is a parameter given by

$$u = \sqrt{\frac{a_e}{2h_r}} \tan \theta_e. \quad (\text{B.33})$$

In order to calculate the PPF, Leonov and Leonov use a slightly different equation for the path length difference:

$$\delta = 2h_r \sin \theta_e \quad [\text{m}]. \quad (\text{B.34})$$

As, a path length difference has already been calculated in eq. (B.31), and it's formulation is similar to that used by Patterson *et al.*, we will not recalculate it. The total phase difference, using δ_0 , is then

$$\gamma = \frac{2\pi}{\lambda} \delta_0 + \alpha \quad [\text{rad}], \quad (\text{B.35})$$

with α to phase change caused by the reflection. The amplitude of the PPF is then found from

$$|F_{se,r}| = f_d \left| 1 + \frac{f_r}{f_d} D \rho_0 \rho_s \rho_v e^{-j\gamma} \right|, \quad (\text{B.36})$$

or equivalently,

$$|F_{se,r}| = f_d \sqrt{1 + \left(\frac{f_r}{f_d} D \rho_0 \rho_s \rho_v \right)^2 + 2 \left(\frac{f_r}{f_d} D \rho_0 \rho_s \rho_v \right) \cos \gamma}, \quad (\text{B.37})$$

with f_d = antenna pattern factor in direction of direct ray

f_r = antenna pattern factor in direction of reflected ray.

In decibels, the amplitude of the PPF is

$$F_{se,r,\text{dB}} = 20 \log |F_{se,r}|. \quad (\text{B.38})$$

Diffraction region. In this region the diffraction is assumed to be either due to a smooth sphere, or a knife-edge — the RMS surface roughness is the deciding factor. First we calculate the distance to the radar horizon as

$$R_h = \sqrt{2a_e} \left(\sqrt{h_r} + \sqrt{h_t} \right) \quad [\text{m}]. \quad (\text{B.39})$$

The critical roughness is determined from

$$\sigma_{h,\text{cr}} = \frac{\sqrt{\lambda R_h}}{2} \quad [\text{m}]. \quad (\text{B.40})$$

If the RMS roughness of the surface at the radar horizon, σ_h [m], is larger than $\sigma_{h, cr}$, then knife-edge diffraction occurs.

The smooth-sphere diffraction model requires a number of height-gain factors to be calculated. The first is h_{min} , determined from the permittivity ϵ_r and conductivity σ_e of the surface as

$$h_{min} = \frac{\lambda}{2\pi} \cdot \frac{[\epsilon_r^2 + (60\sigma_e\lambda)^2]^{b/2}}{[(\epsilon_r - 1)^2 + (60\sigma_e\lambda)^2]^{1/4}} \quad [\text{m}] , \quad (\text{B.41})$$

where b is a parameter taken as $b = 0$ for HH, and $b = 1$ for VV polarisation. Another three height parameters are needed:

$$h_1 = \begin{cases} h_{min} & , h_r < h_{min} \\ h_r & , \text{ otherwise} \end{cases} \quad (\text{B.42})$$

$$h_2 = \begin{cases} h_{min} & , h_t < h_{min} \\ h_t & , \text{ otherwise} \end{cases} \quad (\text{B.43})$$

$$h_c = 30\lambda^{\frac{2}{3}} . \quad (\text{B.44})$$

There are two gain factors:

$$g_1 = \begin{cases} 1 & , h_r \leq h_c \\ 0,1356 \left(\frac{h_r}{h_c}\right)^{-0,904} \cdot 10^{0,948 \cdot \sqrt{\frac{h_r}{2h_c}}} & , \text{ otherwise} \end{cases} \quad (\text{B.45})$$

$$g_2 = \begin{cases} 1 & , h_t \leq h_c \\ 0,1356 \left(\frac{h_t}{h_c}\right)^{-0,904} \cdot 10^{0,948 \cdot \sqrt{\frac{h_t}{2h_c}}} & , \text{ otherwise} \end{cases} \quad (\text{B.46})$$

The smooth sphere diffraction PPF is then

$$F_{se, sm} = 9,29 \cdot 10^{-6} \sqrt{R} \cdot \lambda^{-\frac{3}{2}} \cdot e^{-7,12 \cdot 10^{-5} R \lambda^{-\frac{1}{3}}} \cdot g_1 h_1 g_2 h_2 \quad (\text{B.47})$$

Knife-edge diffraction is slightly simpler to calculate. First three parameters are needed:

$$a = \frac{2}{\lambda} \left(\frac{1}{R_h} + \frac{1}{R - R_h} \right) \quad (\text{B.48})$$

$$b = \frac{R_h}{R} \left[\frac{(R - R_h)^2}{2a_e} + \sigma_h \right] \sqrt{a} \quad (\text{B.49})$$

$$p = \begin{cases} b & , a > 0 \\ 1 & , \text{ otherwise} \end{cases} \quad (\text{B.50})$$

Now the knife-edge diffraction PPF, in decibels, can be calculated from:

$$F_{se, ke_dB} = \begin{cases} -(6 + 8p) & , p \leq 1 \\ -[6,4 + 20 \log(p + \sqrt{p^2 + 1})] & , \text{ otherwise} \end{cases} \quad (\text{B.51})$$

To convert from decibels, use

$$F_{se, ke} = 10^{F_{se, ke_dB}/10}. \quad (\text{B.52})$$

Finally, the diffraction PPF is obtained as

$$F_{se, d} = f_0 F_{se, i}, \quad (\text{B.53})$$

where f_0 is the pattern factor at an elevation angle of 0° , and $F_{se, i}$ is either $F_{se, sm}$ or $F_{se, ke}$, depending on the surface roughness.

Intermediate region. Propagation in the intermediate region is not determined directly, instead logarithmic interpolation between the interference region PPF and the diffraction region PPF is used — this was first proposed by Kerr [49]. The transition region PPF is given as

$$F_{se, t_dB} = F_{se, r_dB} \frac{\delta_0}{Z} + F_{se, d_dB} \left(1 - \frac{\delta_0}{Z}\right), \quad (\text{B.54})$$

where Z is the threshold used to determine whether propagation takes place in the interference or intermediate region (see p. 160). The threshold is set to $Z = \lambda/8$.

B.5 Scattering

B.5.1 Target models

When modelling radar target returns, the four parameters of interest are the return's amplitude, phase, frequency and range delay. The first two are complex functions of the target's geometry, construction materials, position, aspect angle and the radar frequency. The return frequency is determined from the doppler effect, and the range delay is easily calculated from the slant range.

Our analysis of target modelling has two parts. Firstly, we discuss the target radar cross-section and methods of determining it. Secondly, we consider the inherent fluctuations of a target's measured RCS (target noise).

Target radar cross-section

Far-field. Using the geometry in Figure B.1, the following equation holds [51, p. 329]

$$x^2 + 2dx + d^2 = x^2 + \frac{y^2}{4} \quad (\text{B.55})$$

with x the far-field distance, y the antenna height, and d the maximum path length difference.

Taking d as a tenth of a wavelength, we can assume that $x \gg d$ for all practical purposes. This leads to a simplified equation for the far-field distance x

$$x = \frac{y^2}{8d} \quad (\text{B.56})$$

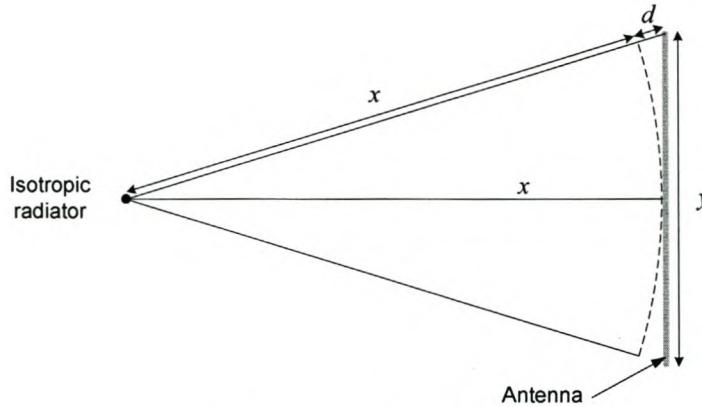


Figure B.1: Geometry for far-field measurements.

Simple RCS models. Leonov and Leonov give some very simple, empirical models for the mean RCS calculation [54, p. 160]. For an aircraft,

$$\bar{\sigma}_{aircraft} = 0,01L_a^2 \quad [\text{m}^2] \quad (\text{B.57})$$

where L_a is the length of the aircraft [m].

For a ship, the displacement is needed:

$$D = 2,5 \cdot 10^{-6}L_s^3 \quad [\text{kilotons}], \quad (\text{B.58})$$

with L_s the length of the ship [m]. Then the cross-section can be determined as

$$\bar{\sigma}_{ship} = \begin{cases} 1644f^{\frac{1}{2}}D^{\frac{3}{2}} & \text{for low elevation angles} \\ 1000D & \text{for high elevation angles} \end{cases} \quad [\text{m}^2], \quad (\text{B.59})$$

where f is RF frequency [GHz].

Target noise

Doppler scintillation. We repeat eq. (2.26) here for convenience,

$$\text{Hankel PDF: } p(f) = \frac{1}{2\pi\sigma_\psi\sigma_\omega} K_0\left(\frac{f}{2\sigma_\psi\sigma_\omega}\right) \quad (\text{B.60})$$

with K_0 = modified Hankel function

f = frequency, [Hz]

σ_ψ = standard deviation of angle scintillation, []

σ_ω = standard deviation of yaw rate, [Hz]

Typical values for σ_ω in a clear medium-turbulence atmosphere are: 2,2 mHz for a commerical airplane; 3,7 mHz for a bomber; and between 4,2 and 7,2 mHz for a fighter [78, p. 28-18].

Table B.4: *The Beaufort wind scale, from [57, p. 195].*

Beaufort scale	Description	Wind speed [km/h]
0	Calm	<1,9
1	Light air	1,9 to 5,6
2	Light breeze	7,4 to 11,1
3	Gentle breeze	13,0 to 18,5
4	Moderate breeze	20,4 to 29,6
5	Fresh breeze	31,5 to 38,9
6	Strong breeze	40,8 to 50,0
7	Near gale	51,9 to 61,1
8	Gale	63,0 to 74,1
9	Strong gale	76,0 to 87,1
10	Storm	87,1 to 101,9
11	Violent storm	103,8 to 116,7
12	Hurricane	>118,6

The angle scintillation σ_{ang} [m] can be used to determine values for σ_ψ using

$$\sigma_\psi = \frac{2\pi\sigma_{ang}}{\lambda} \quad (\text{B.61})$$

with λ the RF wavelength [m]. Typical values of σ_{ang} are between $0,15L$ and $0,25L$ (where L is half the target's wingspan). For a nose-on view of a small aircraft, $\sigma_{ang} \approx 0,1L$. For side views of any aircraft or nose-on views of larger aircraft, σ_{ang} tends to $0,3L$ [78, p. 28-10].

B.5.2 Sea clutter

Constant models

Barton model. This empirically determined model for the average clutter cross-section is given in [57, p. 195]. First the normal reflectivity γ , must be determined from the wind speed

$$10 \log \gamma = 6K_B - 10 \log \lambda - 64 \quad [\text{dB}/\text{m}^2] \quad (\text{B.62})$$

with K_B the wind speed on the Beaufort scale (see Table B.4), and λ the RF wavelength [m].

The sea's scattering coefficient σ^0 , which must be multiplied by the area of the corresponding radar resolution cell to determine the clutter's average RCS $\overline{\sigma_c}$, is then

$$\sigma^0 = \gamma \sin \Psi \quad [\text{m}^2/\text{m}^2] \quad (\text{B.63})$$

with Ψ the grazing angle [rad]. The grazing angle can be calculated [57, p. 195], using the constant- k refraction model (see Section 2.3.1 on p. 154), as

$$\Psi = \arcsin \left[\frac{a_e^2 + R^2 - (a_e + h_r)^2}{2a_e R} \right] \quad (\text{B.64})$$

- with Ψ = grazing angle, [rad]
- a_e = effective earth's radius, [m]
- h_r = radar's height above the surface, [m]
- R = slant range to earth's surface, [m]

The error incurred by using the constant k model instead of a more accurate ray-tracing approach is unimportant, considering the poor accuracy inherent in this empirical model.

GIT model. For the Georgia Institute of Technology model [64, p. 109], the average clutter power in dBW is given as

$$P_c = -123 + 10 \log \left(P_t \lambda^2 R^{-4} p(\alpha)^4 \right) + 2G + \sigma_c - L_s \quad (\text{B.65})$$

- with P_t = transmitted power, [kW]
- λ = RF wavelength, [m]
- R = slant range, [km]
- $p(\alpha)$ = antenna pattern factor
- α = launch angle that intercepts surface
- G = antenna gain, [dB]
- σ_c = average clutter cross-section, [dB]
- L_s = miscellaneous system losses, [dB]

The clutter cross section σ_c in decibels relative to 1 m^2 is

$$\sigma_c = \sigma^0 + A_c \quad (\text{B.66})$$

with σ^0 the average clutter cross section per area [dB], and A_c the radar resolution cell's area [dB]. The latter is determined from

$$A_c = 10 \log \left(\frac{1000 R c \Delta\theta \tau_c}{4 \ln 2} \right) \quad (\text{B.67})$$

- with R = slant range, [km]
- c = speed of light, [m/s]
- $\Delta\theta$ = antenna azimuth beamwidth, [rad]
- τ_c = radar compressed pulse width, [s]

Calculations of the average clutter cross section depend on the polarisation. For horizontal polarisation

$$\sigma^0_H = 10 \log \left(3,9 \cdot 10^{-6} \lambda \Psi^{0,4} A_i A_u A_w \right) \quad (\text{B.68})$$

where Ψ is the grazing angle [rad]. The other factors will be discussed below. For vertical polarisation the equation takes one of two forms. For frequencies above 3 GHz, it is

$$\sigma^0_V = \sigma^0_H - 1,05 \ln(h_{avg} + 0,02) + 1,09 \ln(\lambda) + 1,27 \ln(\Psi + 10^{-4}) + 9,7 \quad (\text{B.69})$$

and for frequencies below 3 GHz, it is

$$\sigma^0_V = \sigma^0_H - 1,73 \ln(h_{avg} + 0,02) + 3,76 \ln(\lambda) + 2,46 \ln(\Psi + 10^{-4}) + 22,2 \quad (\text{B.70})$$

If using circular polarisation, the maximum of σ^0_H and σ^0_V must be reduced by six decibels, i.e.

$$\sigma^0_C = \sigma^0_{max} - 6 \quad (\text{B.71})$$

where h_{avg} is the average wave height, given in terms of the wind speed W_s [m/s] as

$$h_{avg} = \left(\frac{W_s}{8,67} \right)^{2,5} \quad (\text{B.72})$$

The wind speed factor A_w is given as

$$A_w = \left(\frac{1,9425 W_s}{1 + \frac{W_s}{15}} \right)^{1,1(\lambda+0,02)^{-0,4}} \quad (\text{B.73})$$

The interference term A_i as

$$A_i = \frac{\sigma_\phi^4}{1 + \sigma_\phi^4} \quad (\text{B.74})$$

with σ_ϕ a roughness parameter calculated as

$$\sigma_\phi = \frac{(14,4\lambda + 5,5)\Psi h_{avg}}{(\lambda + 0,02)} \quad (\text{B.75})$$

The upwind/downwind factor A_u is given as

$$A_u = e^{0,2(1-2,8\Psi)(\lambda+0,02)^{-0,4} \cos(\phi)} \quad (\text{B.76})$$

The GIT model is valid below 2 GHz, and from grazing angles between $0,1^\circ$ and 10° . The maximum applicable range is given as

$$R_{lim} = 0,5 \left(-2\Psi a_e + \sqrt{(2\Psi a_e)^2 + 0,008 a_e h_r} \right) \quad (\text{B.77})$$

where a_e is the effective earth's radius [km] (see eq. (2.3) on p. 11), and h_r is the height of the radar above the ground [m]. The grazing angle for smaller ranges can then be found using

$$\Psi = \frac{h_r}{1000R} - \frac{R}{2a_e} \quad (\text{B.78})$$

In this case, the launch angle α will be the same as the negative of the grazing angle Ψ . A modified version of the GIT model is used in the EREPS program [64, p. 109]. The modifications are applied for frequencies above 2 GHz, and also account for ducting conditions. Equations for the launch angle and grazing angle, as well as clutter beyond R_{lim} are given in [64, p. 113] — they will not be stated.

Deterministic models

There are no additional details for this section.

Probabilistic models

Single point distribution parameter estimation. The estimators given below are a summary of the details reported by Noga [61, pp. 24–26]. The amplitude of the returned signal is defined as r , and the intensity or power of the signal, as $\sigma_c = r^2$. The sample mean of a process is denoted as $\langle \cdot \rangle$ below.

For the negative exponential distribution, eq. (2.30), the maximum likelihood (ML) estimator of the average clutter cross section is simply

$$\overline{\sigma_c} = \frac{4}{\pi} \langle r^2 \rangle = \langle \sigma_c \rangle \quad (B.79)$$

The ML parameter estimates for the Weibull distribution, eq. (2.31), can be calculated by solving for a and b in

$$\begin{aligned} -\frac{1}{b} + \frac{\langle r^b \ln r \rangle}{\langle r^b \rangle} &= \langle \ln r \rangle \\ a &= \frac{1}{\langle r^b \rangle} \end{aligned} \quad (B.80)$$

From [57, p. 197], b varies between about 0,67 for a rough sea, and 1,59 for a calm sea.

According to Blacknell [12], the best overall estimation performance for single look K-distributed data is obtained by using the data's mean and the natural logarithm of the data's mean. The following equations, adapted from [61, p. 26],

$$\begin{aligned} \psi^{(0)}(\alpha) - \ln(\alpha) &= \langle \ln \sigma_c \rangle - \ln \langle \sigma_c \rangle - \psi^{(0)}(1) \\ \beta &= 2\sqrt{\frac{\alpha}{\langle \sigma_c \rangle}} \end{aligned} \quad (B.81)$$

must be solved for α and β . $\psi^{(0)}(\cdot)$ denotes the digamma function. The parameter α is known as the order or shape parameter. Typically, it varies between 0,1 for very spiky sea clutter to 10 for near Rayleigh distributed clutter [61, p. 26]. The mean of the distribution is $4\alpha/\beta^2$.

An empirical model for the shape parameter is given by Ward *et al.* [92] as

$$\log \alpha = \frac{2}{3} \log \Psi + \frac{5}{8} \log l + k_a - k_f \quad (B.82)$$

with α = shape parameter

l = across range resolution, [m]

Ψ = grazing angle, [deg], with $(0,1^\circ < \phi < 10^\circ)$

k_a = aspect dependency constant

k_f = polarisation dependency constant

The aspect dependency is defined as follows:

$$k_a = \begin{cases} -\frac{1}{3} & \text{for up or down swell conditions} \\ +\frac{1}{3} & \text{for across swell conditions} \\ 0 & \text{for intermediate directions or when no swell exists} \end{cases} \quad (B.83)$$

Table B.5: Minimum slant range to surface for a grazing angle $\Psi = 10^\circ$, with $k = 4/3$ for various radar heights.

h_r [m]	R [m]
10	58
20	115
30	173
40	230
50	288

The polarisation dependency requires $k_f = 1$ for VV polarisation and $k_f = 1, 7$ for HH polarisation. This empirical model was derived from X-band measurements. Ward *et al.* note that the value of α so determined was independent of the sea state, wind speed and aspect angle relative to the wind direction. According to Andraka and Phelps [6], the sea state can be accounted for by using an empirical model for the scattering coefficient, e.g. the one in eq. (B.62). The power of the normalised K-distribution is then multiplied by the average clutter cross-section in the desired range cell.

The limitation on the grazing angle is not a major problem for shipboard radar. The grazing angle was given in eq. (B.64). Solving for the slant range R , results in

$$R = a_e \cos\left(\Psi + \frac{\pi}{2}\right) \pm \sqrt{a_e^2 \cos^2\left(\Psi + \frac{\pi}{2}\right) + (2a_e h_r + h_r^2)} \quad (\text{B.84})$$

where the positive square root is the only valid solution. Using equation (B.84), the values in Table B.5 were calculated, for a grazing angle of $\Psi = 10^\circ$, with a 4/3 effective earth radius. From the table it is clear that the grazing angle limitation is only a problem at very short ranges, or for radars that are mounted high above the surface. Airborne radar may be problematic.

B.5.3 Ground clutter

Probabilistic models

Composite models. Jao [45] obtains expressions relating the K-distribution parameters to some physical parameters by applying his clustered hypothesis. The mean clutter cross section $\bar{\sigma}_c$ is given as

$$\bar{\sigma}_c = \bar{N}_C E A_c S(\theta_d) \bar{\sigma}_a \quad (\text{B.85})$$

Table B.6: Weibull shape parameter b for various terrains, from [57, p. 193].

Terrain	Shape parameter b	Frequency band
Rolling hills	0,626	L-Band
Rocky mountains	0,512	S-Band
Forest	0,250	X-Band

with \bar{N}_C = average scatterer population per cluster
 E = population density
 A_c = radar resolution cell area
 θ_d = depression angle
 $S(\theta_d)$ = shadowing function
 $\bar{\sigma}_a$ = RCS of an individual scatterer

The shadowing function is the probability that a point is not shadowed when viewed at the given depression angle. The shape parameter can be calculated from

$$\alpha = \frac{E}{n_C} A_c S(\theta_d) \quad (\text{B.86})$$

where n_C determines the rate of scatterer population growth. It can be related to the average population per cluster as $N_C = \frac{1}{n_C} (e^{n_C} - 1)$. The major uncertainty in these relationships lies in the shadowing function and individual scatterer RCS.

Parameter estimation. Meikle [57] reports a simple model for ground clutter from rural land given by Nathanson. The scattering coefficient is calculated as

$$\sigma^0 = \frac{0.00032}{\lambda} \quad [\text{m}^2/\text{m}^2] \quad 0..30 \text{ km} \quad (\text{B.87})$$

In this model, the horizon is assumed to be at $R = 30$ km, after which diffraction occurs. Values at longer ranges are accounted for by scaling the clutter cross-section on the horizon by $1/R^4$. Meikle suggests modelling the fluctuations with either a log-normal or Weibull distribution. For the former, a mean-to-median ratio (see eq. (2.25) on p. 30) of 100 should be used. For the Weibull distribution, the scale parameter a is related to the mean clutter power $\bar{\sigma}_c$ by

$$a = \frac{1}{\bar{\sigma}_c} . \quad (\text{B.88})$$

To determine $\bar{\sigma}_c$ for a certain range cell, σ^0 is simply multiplied by the area of the range cell A_c — see eq. (B.67) on p. 166. Meikle gives examples of the Weibull scale parameter for different terrains — these are listed in Table B.6. Strangely, each terrain is given at a different frequency band.

Jao reports the parameters for K-distributed clutter from two terrain types (see Table B.7), measured at X-band with HH polarisation [45]. The level of the background

Table B.7: Composite K-distributed terrain model parameters for various terrains measured at X-band, from [45].

Terrain	σ_b^0 [dB]	α	σ_1^0 [dB]	σ_2^0 [dB]
Rolling forest	-47	1,5	-25	-9
Level and agricultural land	-48	0,1	-28	-3

radiation is given as σ_b^0 , and is modelled as a complex Gaussian process. Both terrain types require two K-distributed PDFs — they have a common shape parameter α , but different mean scattering coefficients denoted by σ_1^0 and σ_2^0 . The scale parameter β can be determined from the n th mean as $\sigma_n^0 = 4\alpha/\beta^2$.

B.5.4 Volume clutter

Hydrometeors

Mean RCS. The RCS of a cloud is obtained by summing the returns from the N particles in the radar's volume resolution cell:

$$\sigma_{cloud} = \frac{C\pi^5}{\lambda^4} \sum_{i=1}^N D_i^6 [\text{m}^2], \quad (\text{B.89})$$

with C a dimensionless constant between 0,197 (ice) and 0,931 (water) and D the drop's diameter (see p. 45). By defining a radar reflectivity factor Z as

$$\begin{aligned} Z &= \sum_{i=1}^N D_i^6 / \text{unit volume } [\text{mm}^6 \text{m}^{-3}] \\ &= a R_r^b \end{aligned} \quad (\text{B.90})$$

where a and b empirically relate the rainfall rate R_r [mm/h] to Z , we can evaluate equation B.89. Common values for a and b are shown in Table B.8, and the description of the rainfall rates in Table B.9.

Table B.8: Empirically determined constants for relating the reflectivity factor Z to the precipitation rate R_r .

Precipitation	a	b
Rain	200	1,60
Thunderstorm	486	1,37
Ice crystal	500	1,66
Wet snowflakes ($t > 0^\circ\text{C}$)	2000	2,00
Dry snowflakes ($t < 0^\circ\text{C}$)	1050	2,00

Table B.9: *Description of precipitation rates.*

Description	Rainfall rate R_r [mm/h]
Drizzle	0,25
Light rain	1
Moderate rain	4
Heavy rain	16
Excessive rain	40
Light snowfall	0,5 – 4
Heavy snowfall	4 – 30

For a volume resolution cell V_{res} , see eq (2.34), the total RCS of a cloud is

$$\begin{aligned}\sigma &= V_{res} \sum \sigma_{particle}/\text{unit volume} \\ &= \frac{C\pi^5}{\lambda^4} ZV_{res} \quad [\text{m}^2]\end{aligned}\tag{B.91}$$

B.6 Electronic warfare

There are no additional details for this section.

B.7 Simulators

There are no additional details for this section.

B.8 Summary

Details of a large number of radar environment simulation models were provided in this chapter. Combined with the discussion in Chapter 2, we have covered the majority of the environmental effects that need to be considered for simulation.

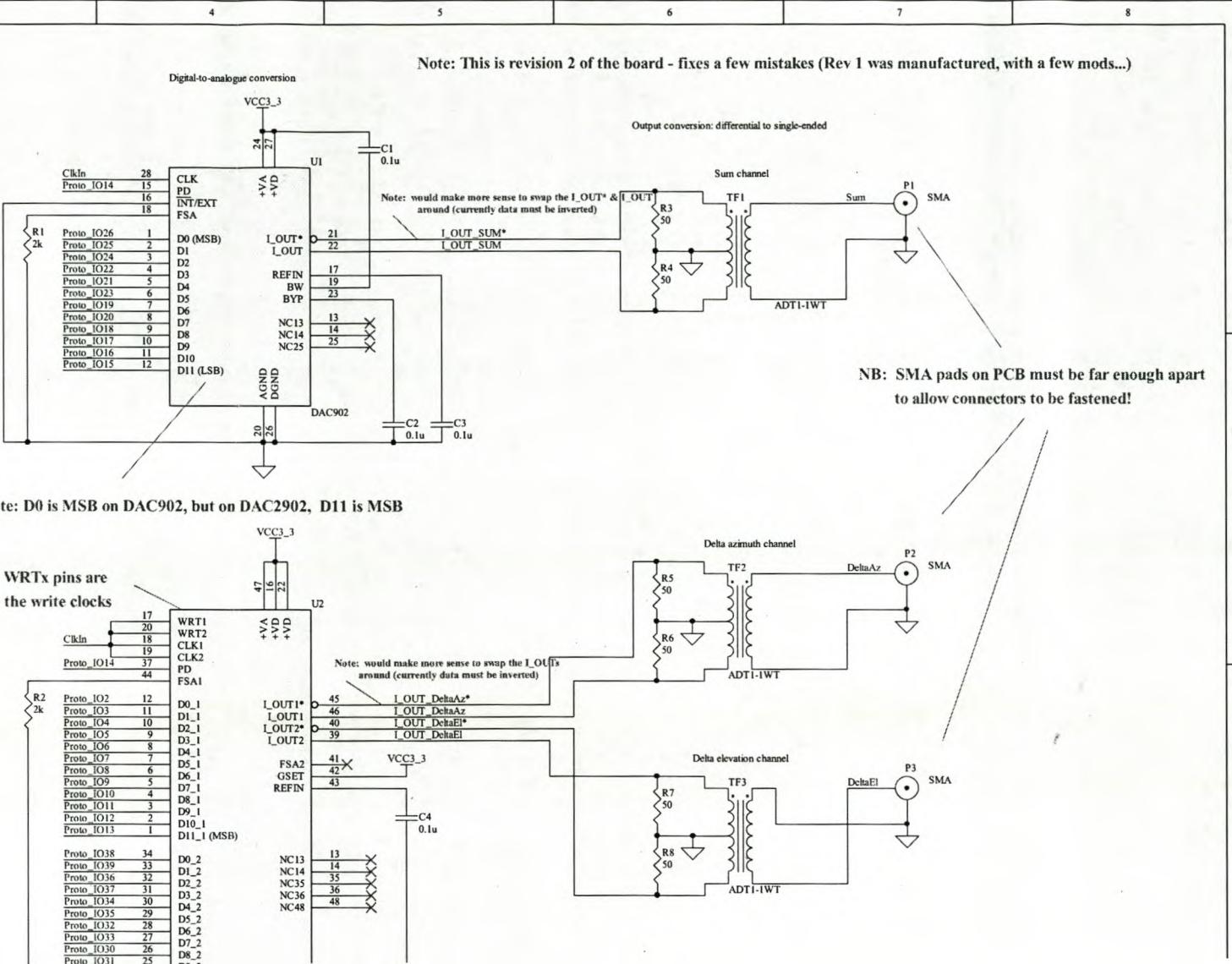
Appendix C

Design and implementation details

C.1 DAC prototype board

The schematic for the digital-to-analogue converter board is shown in Figure C.1, and the PCB layout in Figure C.2.

CHAPTER C – DESIGN AND IMPLEMENTATION DETAILS



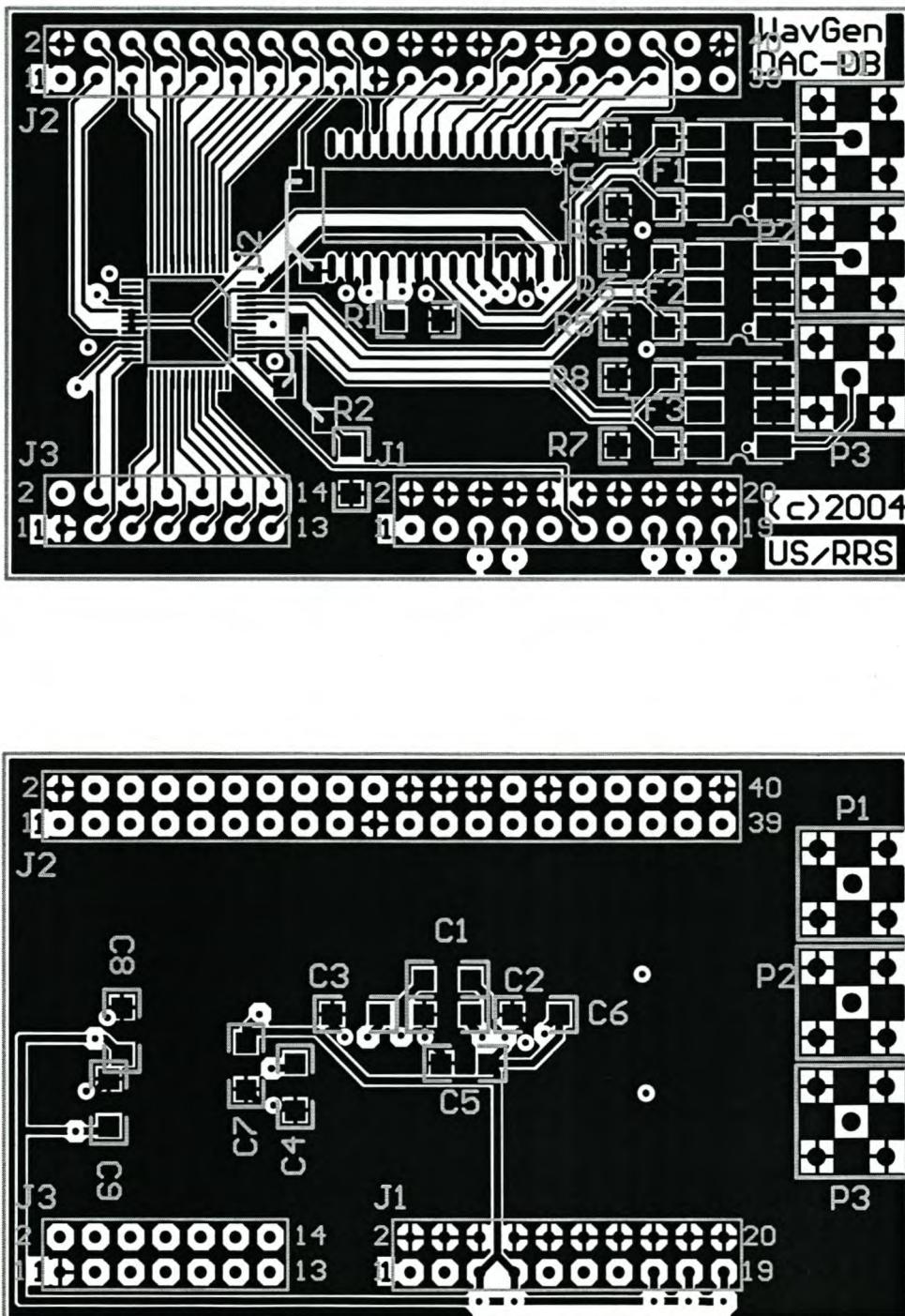


Figure C.2: DAC prototype board layout. The upper image shows the top layer, and lower one the bottom layer (mirrored).

C.2 Interface description

The details of the messages sent between the PC and waveform generator are stored in an extensible mark-up language (XML) file. We first list the type definitions used in some of the message fields (Section C.2.1), and then give the definitions of the messages themselves (Section C.2.2).

C.2.1 Message type definitions

Table C.1: *E2_MSG_STATUS*

Table C.2: *E1b_ERROR_STATUS*

Table C.3: *E1b_WG_MODE.*

Table C.4: *E1b_DISABLE*

Table C.5: *E/b_RESET*.

C.2.2 Message details

Table C.6: *WavGenStatusSnd*.

Name	SrcNode	DestNode	ReplyMsg
WavGenStatusSnd	RAES	WavGen	WavGenStatusRsp

Table C.7: *WavGenStatusRsp.*

Name	SrcNode	DestNode	ReplyMsg
WavGenStatusRsp	WavGen	RAES	

CHAPTER C — DESIGN AND IMPLEMENTATION DETAILS

178

Table C.8: *WavGenParametersSnd*

Table C.9: *WavGenParametersRsp*

CHAPTER C — DESIGN AND IMPLEMENTATION DETAILS

179

Table C.10: *WavGenPatternInfoSnd*

Name	SrcNode	DestNode	ReplyMsg
WavGenPatternInfoSnd	RAES	WavGen	WavGenPatternInfoRsp

Table C.11: *WavGenPatternInfoRsp*

Name	SrcNode	DestNode	ReplyMsg
WavGenPatternInfoRsp	WavGen	RAES	

CHAPTER C — DESIGN AND IMPLEMENTATION DETAILS

180

Table C.12: *WavGenPatternProgramSnd*

Name	SrcNode	DestNode	ReplyMsg
WavGenPatternProgramSnd	RAES	WavGen	WavGenPatternProgramRsp

Table C.13: *WavGenPatternProgramRsp.*

Name	SrcNode	DestNode	ReplyMsg
WavGenPatternProgramRsp	WavGen	RAES	

CHAPTER C — DESIGN AND IMPLEMENTATION DETAILS

181

Table C.14: *WavGenClutterReturnListSnd*

Name	SrcNode	DestNode	ReplyMsg
WavGenClutterReturnListSnd	RAES	WavGen	WavGenClutterReturnListRsp

Table C.15: *WavGenClutterReturnListRsp*

Name	SrcNode	DestNode	ReplyMsg
WavGenClutterReturnListRsp	WavGen	RAES	

CHAPTER C — DESIGN AND IMPLEMENTATION DETAILS

182

Table C.16: *WavGenScattererReturnListSnd*

Table C.17: *WavGenScattererReturnListRsp*

Table C.18: *WavGenBurstStartSnd*.

C.3 Waveform generator block diagrams

This section shows the block diagrams for main components of the waveform generator's firmware.

C.3.1 Top level

Figure C.3 shows the waveform generator's top level block diagram.

C.3.2 Return generator

The *Quartus* block diagram for the top level of the return generator is shown in Figure C.4. Note that the *ReturnGen* block is the target generator.

C.3.3 Target generator

The target generator's *Quartus* block diagram is shown in Figure C.4.

C.3.4 DDS engine

In Figure C.6, we show the block diagram that implements the DDS engine's dual return pulse generating pipelines.

C.3.5 Clutter generator

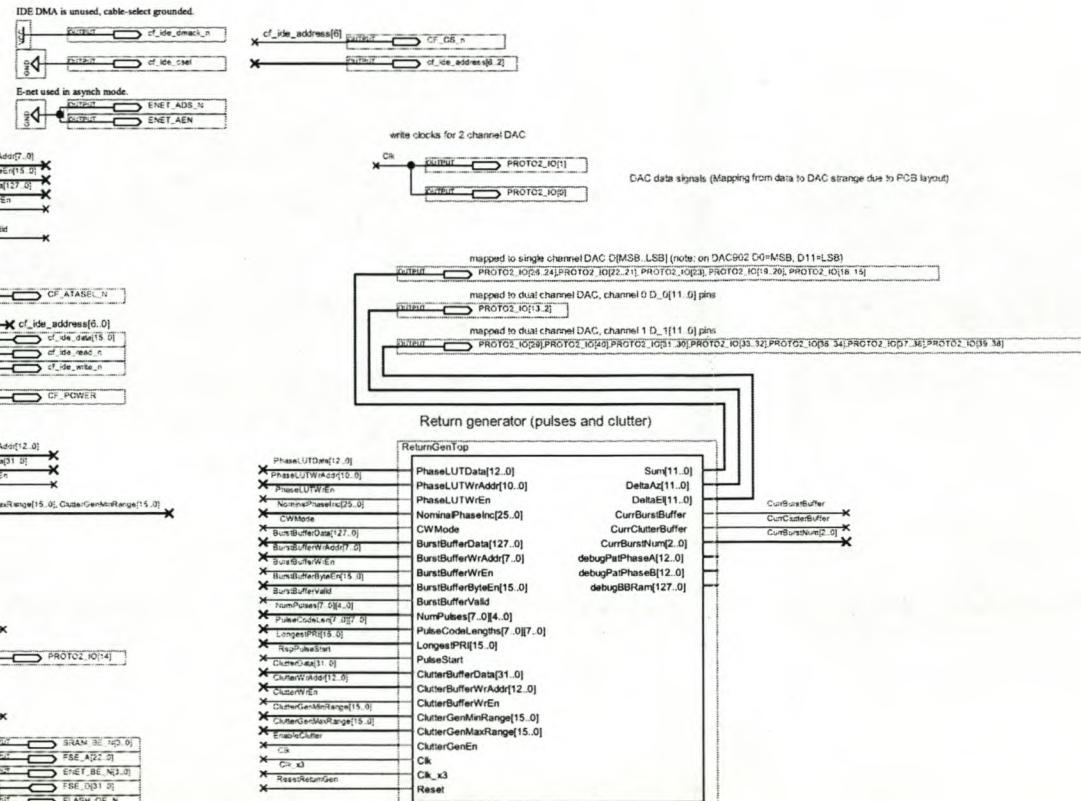
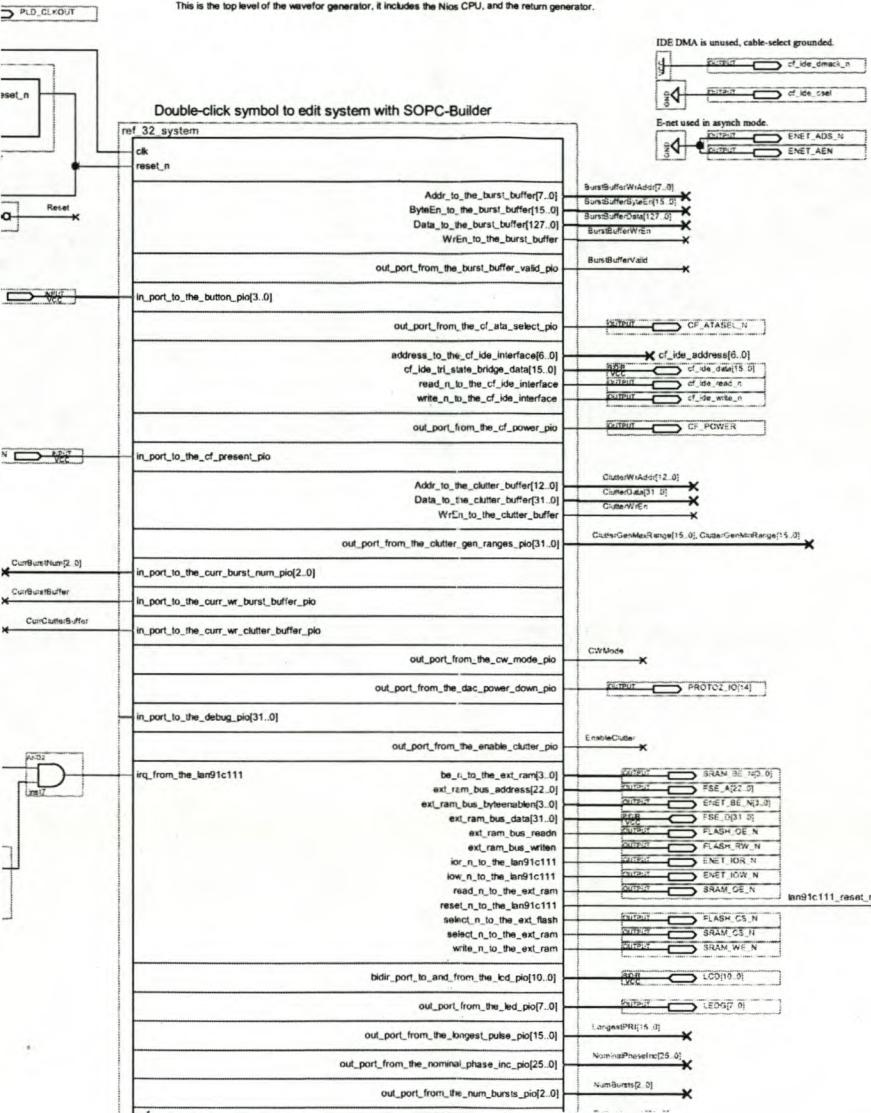
Figure C.7 shows the clutter generator's block diagram.

CHAPTER C — DESIGN AND IMPLEMENTATION DETAILS

WavGen - top level

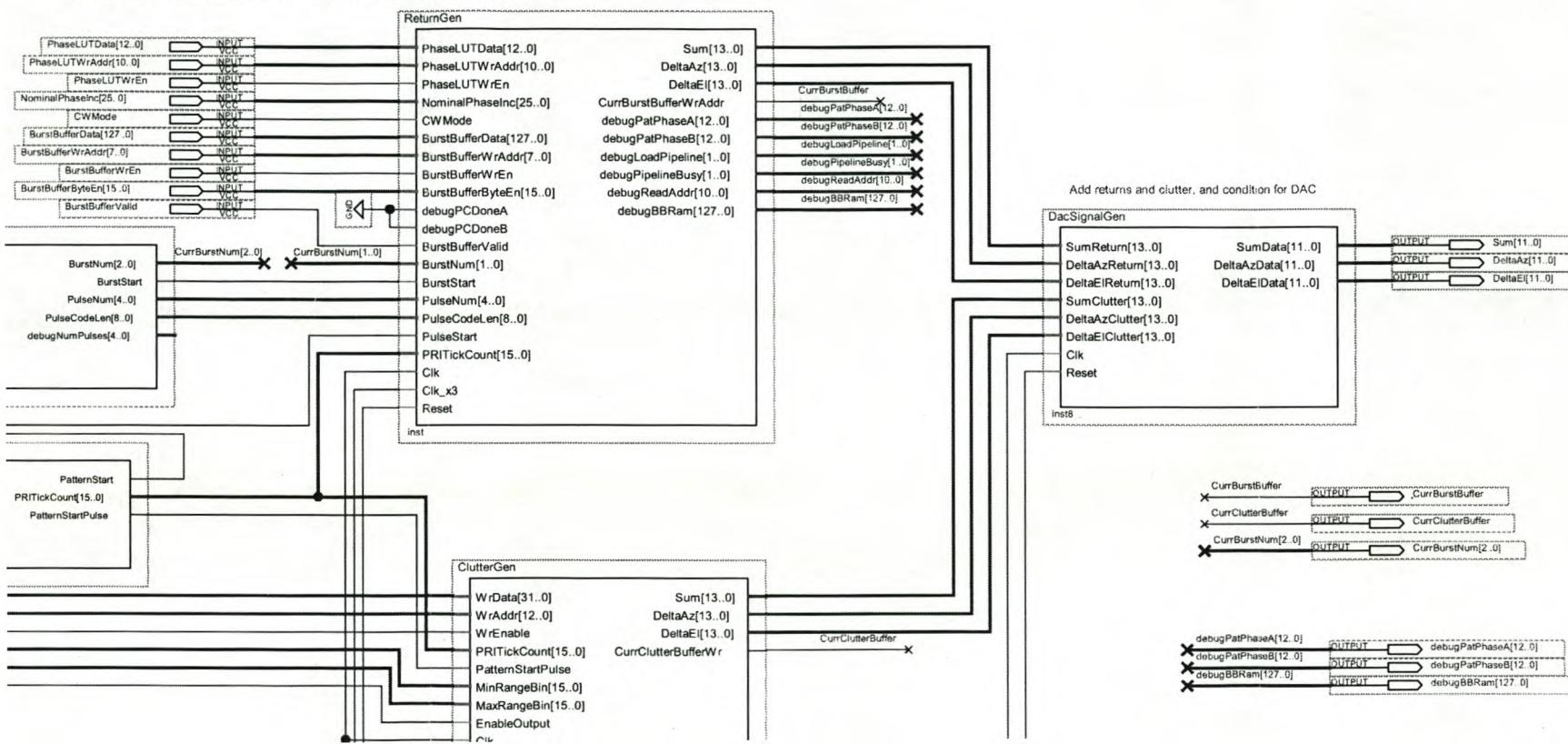
Based on reference design: standard_32.v3.0 -- Stratix 1510.

This is the top level of the waveform generator, it includes the Nios CPU, and the return generator.

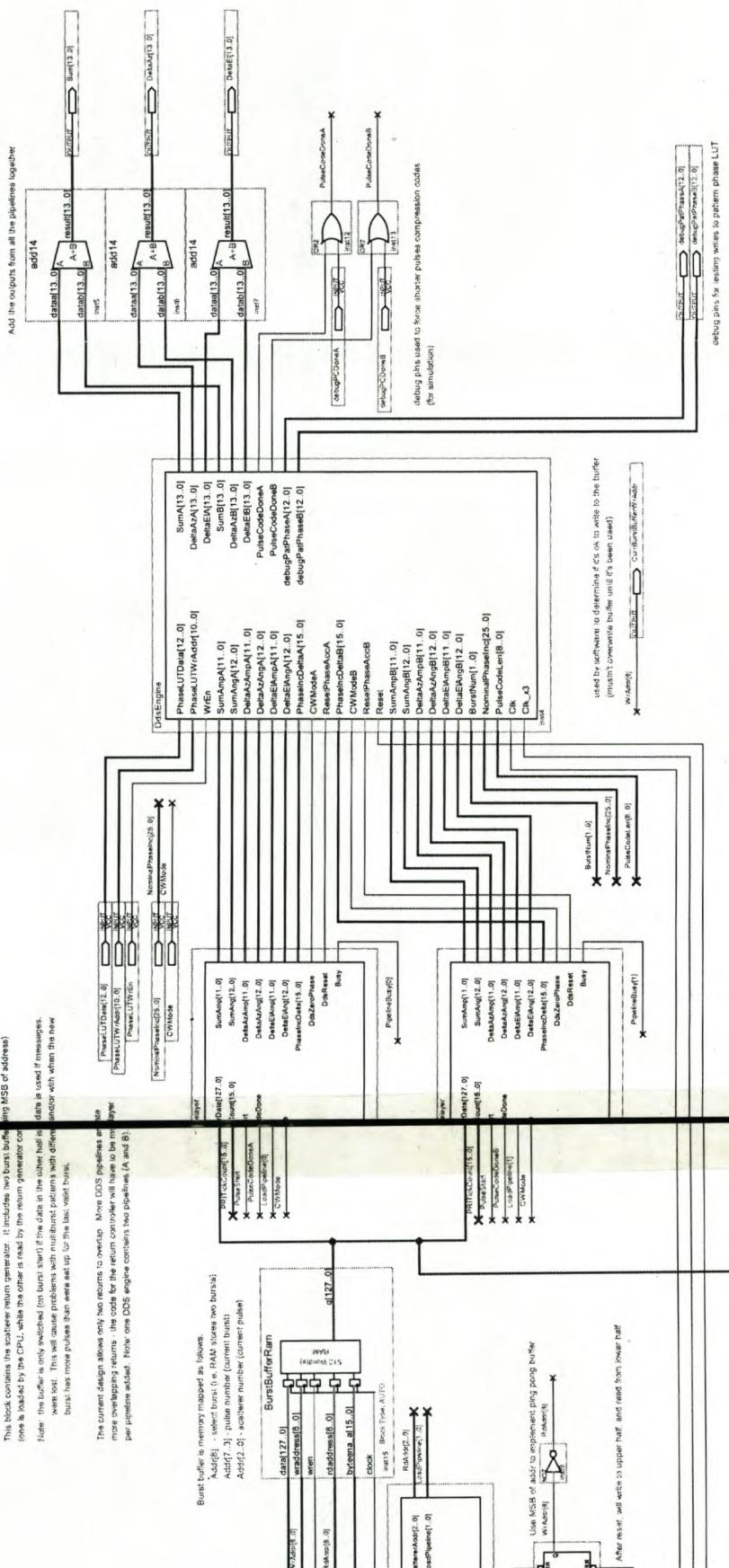


CHAPTER C — DESIGN AND IMPLEMENTATION DETAILS

top level test for the return generator (excluding the Nios)
 combines the pattern synching/counting, the return (pulse) generator and the clutter generator.



CHAPTER C — DESIGN AND IMPLEMENTATION DETAILS



CHAPTER C — DESIGN AND IMPLEMENTATION DETAILS

The DDS engine generates up to 8 returns by synthesising Doppler shifted versions of the base pattern (stuttered data in the pattern LUT). The phase and amplitude for each channel (delta el) can be controlled independently. added to the another phase term (Doppler shifted baseband carrier). $2^N(F + \text{Doppler})$ phase. The Dier is the term generated by DDS. Both of these phase terms are 13-bit values between -pi and pi, as fractional two's complement numbers.

Latency is 7 cycles.

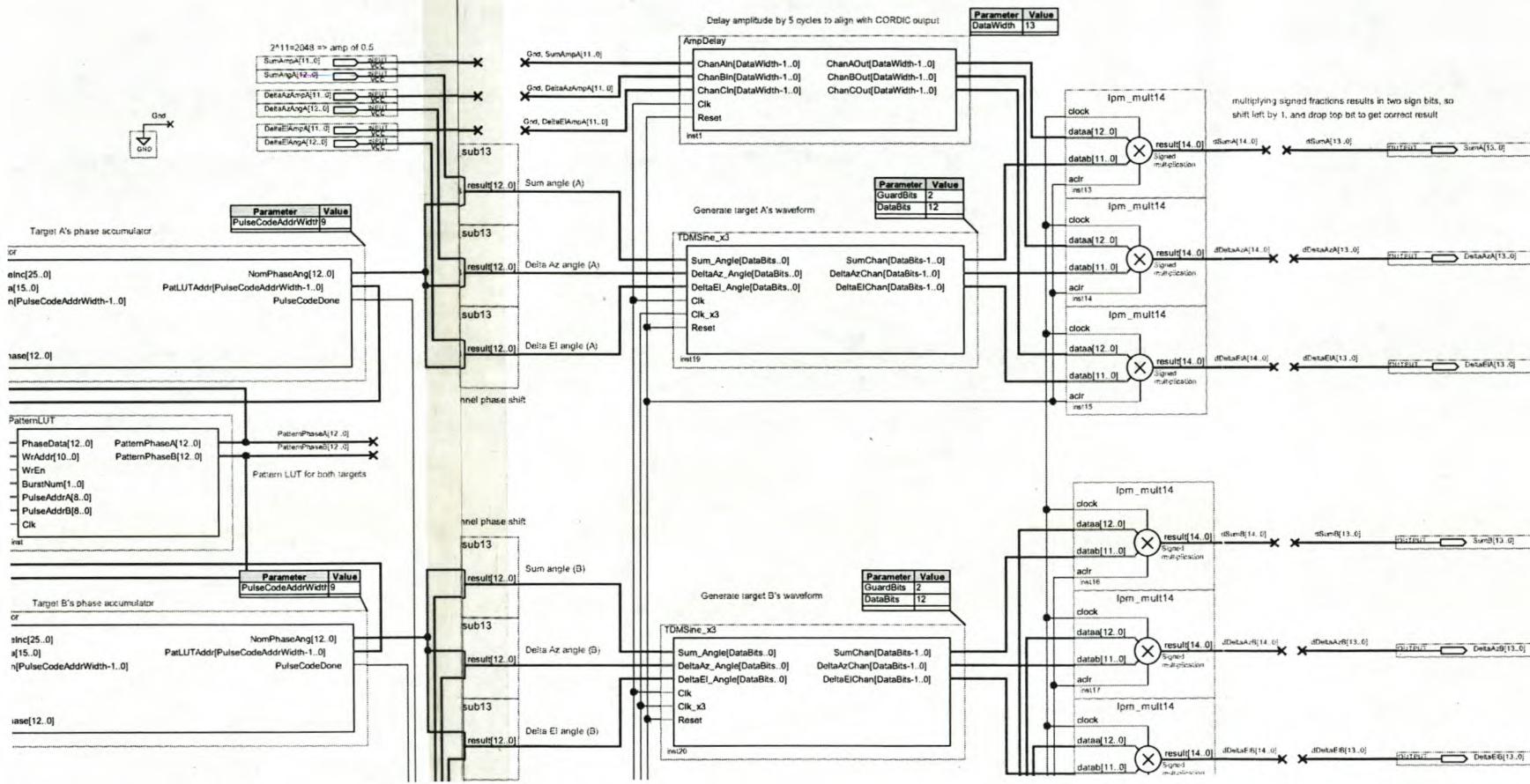
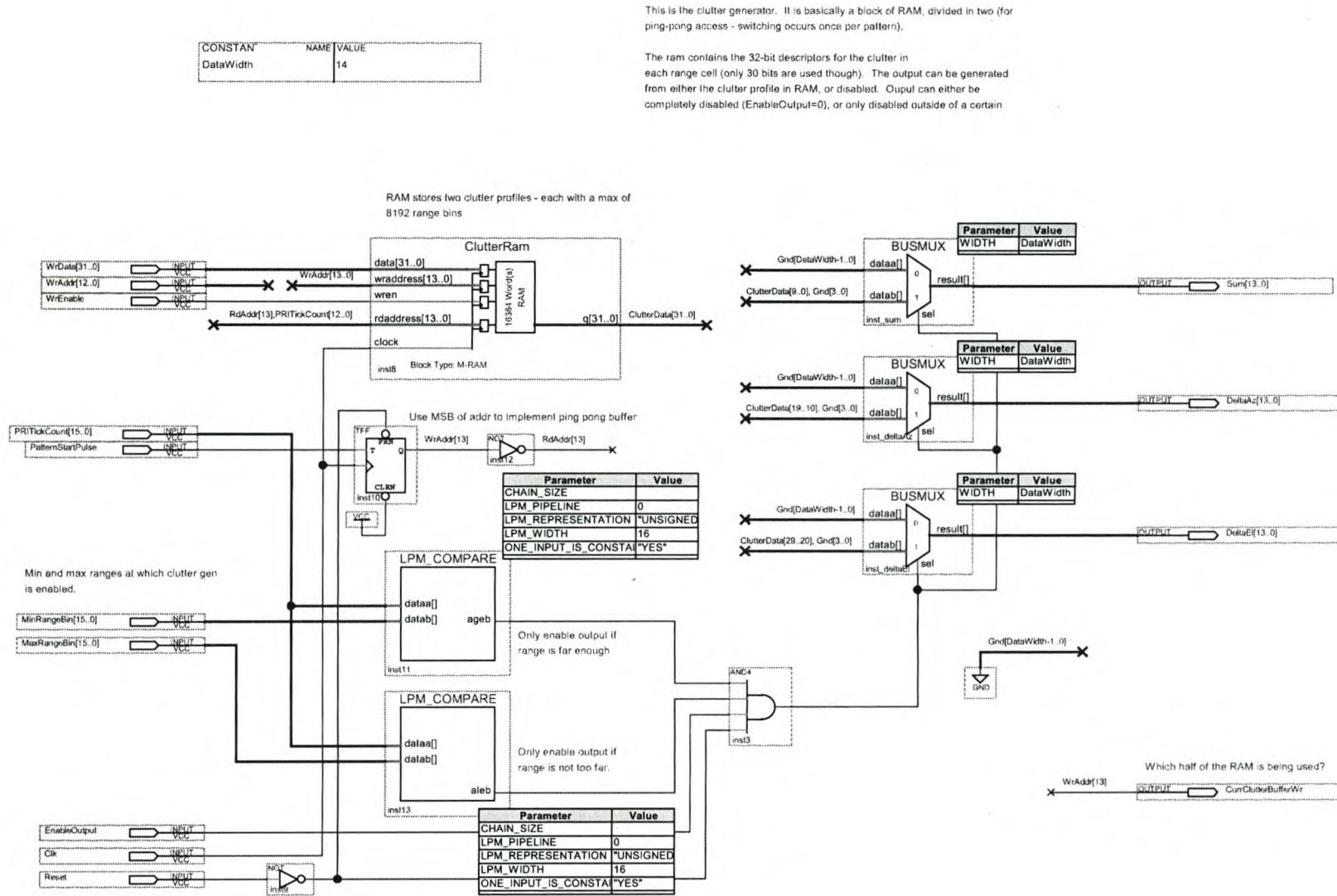


Figure C.7: Block diagram for the clutter generator.



VAK:

TITELNO: 632880

DATUM:

SKENKER/HERKOMS:

BESIT: JA

DUPLIKAATOPNAME: JA

HANDTEKENING:

BESTEMMING: KOM