



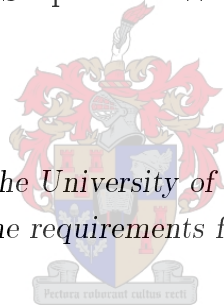
UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Upgrading of a radar system to implement a firmware based pulse compressor

by

Johannes Stephanus Warnich Rust

*Thesis presented at the University of Stellenbosch in partial
fulfilment of the requirements for the degree of*



Master of Science in Engineering

Department of Electrical and Electronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Study leaders:

Dr. M.M. Blanckenberg (US) Mr. P.J. Wolfaardt (RRS)

March 2007

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.

Signature:

J.S.W. Rust

Date:

Abstract

Upgrading of a radar system to implement a firmware based pulse compressor

J.S.W. Rust

*Department of Electrical and Electronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa*

Thesis: MScEng (Electric and Electronic)

March 2007

This project investigates the improvement of an existing radar system by means of currently available technologies and signal processing techniques. Upgrades are aimed to improve the system's specifications with focus on range resolution. Pulse compression and Doppler processing techniques are used to accomplish the improvement in range resolution. The use of pulse compression however resulted in numerous modifications to the RF sub-system required by the introduction of Direct Digital Synthesizer modules. A full replacement of the existing signal processor with a Doppler processing based signal processor implemented on a single programmable firmware device was designed. Certain sections of this signal processor were implemented and tested. Pulse compression was successfully implemented and tested in both the transceiver and signals processor sections.

Opsomming

Upgrading of a radar system to implement a firmware based pulse compressor

J.S.W. Rust

*Departement Elektriese en Elektroniese Ingenieurswese
Universiteit van Stellenbosch
Privaatsak X1, 7602 Matieland, Suid Afrika*

Tesis: MScIng (Elektries en Elektronies)

Maart 2007

Hierdie projek ondersoek die verbetering van 'n bestaande radar stelsel deur gebruik te maak van huidige tegnologie en seinverwerking tegnieke. Aanpasings is gemik om die radarstelselspesifikasies te verbeter met die fokus op afstandresolusie. Pulskompressie en Doppler-filter tegnieke is gebruik om 'n verbetering te maak aan die afstandresolusie van die stelsel. Die gebruik van pulskompressie het 'n groot aantal veranderings aan die RF-stelsel genoodsaak a.g.v. die gebruik van Direkte Digitale Sintetiseerder modules. Die huidige radarseinverwerker is volledig vervang met 'n Doppler gebaseerde verwerker. Hierdie verwerker is volledig ontwerp op 'n enkele programmeerbare fermatuureenheid maar slegs sekere gedeeltes is getoets. Pulskompressie is suksesvol toegepas en getoets in beide die RF-stelsel en in die gedeeltes van die radarseinverwerker wat getoets is.

Acknowledgements

Praise be to God.

I would like to thank the following persons and institutions for their help in making this project possible:

- My study leaders, Dr. M.M. Blanckenberg (US) and Mr. P.J. Wolfaardt (RRS) for their knowledge, time and effort contributed to this project.
- Misters Alex Pienaar and Werner Hattingh for their valuable help with this project.
- The University of Stellenbosch for the use of their academic infrastructure and facilities.
- The ARMSCOR/DefenceTek administrated LEDGER project for their financial support.
- Mr. James Verster, management and personnel of Reutech Radar Systems (Pty) Ltd. for generously providing their facilities, test equipment, expertise, help and support.
- Dr. Werner Steyn (RRS) for his interest in the project and invaluable RF knowledge and help provided.
- Mr. Albert Graham (RRS) for his design help and enthusiasm for the project.
- Mr. Johan Mostert (Armcor) for his interest and project guidance.
- Prof. Michael Inngs (UCT) for his academic insights.
- My wife, Annelie, for her love and support.
- My family, Lisa, Nicola, Elize and Lize, for their interest and motivation.
- My colleagues from the Electronic Systems Laboratory.
- Misters Ernest Lötter and Stéfan van der Walt for their LyX help.
- My friends, for their support and motivation.

For Nicola, Elize, Lize and Annelie.

Contents

Declaration	i
Abstract	ii
Opsomming	iii
Acknowledgements	iv
	v
Contents	vi
List of Figures	x
List of Tables	xiii
List of Abbreviations	xiv
List of Symbols	xvi
1 Project Overview	1
2 Existing Eekhorning Radar System	3
2.1 Introduction	3
2.2 Eekhorning Radar System Description	3
2.3 Eekhorning Radar System Modules	6
2.3.1 Antenna Module	6
2.3.2 Transceiver Module	7

2.3.2.1	RF Sub-system	9
2.3.2.2	Signal Processor Unit	16
2.3.2.3	RF performance measurements	18
2.3.3	Operator Module	18
2.4	Conclusion	20
3	Modified Eekhorning Radar System	21
3.1	Introduction	21
3.2	Transceiver Modifications	21
3.2.1	RF Subsystem	23
3.2.1.1	Transmitter chain	23
3.2.1.2	Receive Chain	36
3.2.1.3	RF Performance measurements	37
3.2.2	Signal Processor Modifications	38
3.2.2.1	Overview	38
3.2.2.2	Quadrature Sampling of the IF signal	40
3.2.2.3	Frequency Planning and Synchronization	41
3.2.2.4	Pulse Compression Implementation	42
3.2.2.5	RSP Design	51
3.2.2.6	RSP Implementation	55
3.3	Conclusion	59
4	Conclusions	60
4.1	RF Sub-system modifications	60
4.2	Signal Processing modifications	61
4.3	General	61
4.4	Future work	62
4.5	Personal note	62
	Bibliography	63

A ERS Technical Information	A-1
A.1 Antenna System	A-1
A.1.1 Antenna Information	A-1
A.1.2 Pedestal Information	A-1
A.2 Transceiver Unit	A-1
A.2.1 Transmitter Information	A-2
A.2.2 Receiver Information	A-2
B Upgrade Hardware Module Descriptions	B-1
B.1 9-time Frequency Multiplier Modules	B-1
B.2 Direct Digital Synthesizers Modules	B-3
B.3 RF Filter Modules	B-5
B.4 900MHz Amplifier Modules	B-6
B.5 RSP Clock Amplifier Module	B-7
B.6 IF Amplifier Module	B-7
B.7 Altera Digital Signal Processing Development Board	B-7
C Supplementary hardware information	C-1
D Calculated Radar Parameters	D-1
D.1 Range to Target	D-1
D.2 Maximum Unambiguous Range	D-1
D.3 Receiver Dead Range	D-2
E Mathematical Representation	E-1
E.1 Common Radar Parameters	E-1
E.1.1 Defined Radar Parameters for the ERS	E-1
E.2 Signal-to-Noise Ratio Calculations	E-2
E.2.1 SNR of the Existing Radar System	E-3
E.2.2 SNR of the Upgraded Radar System	E-9

F Firmware Code Listings	F-1
F.1 Clock Edge Detector Module	F-1
F.2 Clock Synchronization Module	F-3
F.3 DDS Setup Controller	F-4
F.4 DDS LUT Controller	F-9
F.5 Down Converter Module	F-11
F.6 Data Packer Module	F-13
F.7 FIFO Status Module	F-16
F.8 Zero-padder Module	F-18
F.9 Memory Map Controller	F-20
F.10 Alpha Filter Module	F-25

List of Figures

2.1	The Eekhorning Radar System (man-portable setup)	4
2.2	Eekhorning Radar System top level block diagram	5
2.3	Antenna beam elevation coverage diagram (+3°tilt)	6
2.4	The ERS antenna module	7
2.5	The ERS transceiver module	8
2.6	Functional block diagram of the transceiver module	8
2.7	Top-level RF block diagram	9
2.8	Transmit chain RF block diagram	11
2.9	Receive chain RF block diagram	14
2.10	Signal Processing block diagram	17
2.11	Block diagram of a single aircraft detection circuit	18
2.12	Original ERS Operator Module	19
3.1	Functional block diagram of the upgraded Transceiver Module	22
3.2	Photograph of the transceiver modification case	23
3.3	Functional block diagram of the Clock Generation sub-chain	24
3.4	Harmonic signal power levels of the system oscillator filter module	25
3.5	Measured output signal of the RSP clock amplifier module	26
3.6	Bandwidth of the system clock frequency multiplier module	27
3.7	Functional block diagram of the pulse generation sub-chain	28
3.8	Output frequency and signal power level of the transmit DDS module	30
3.9	Bandwidth of the transmit DDS filter module	30

<i>List of Figures</i>	xi
3.10 Harmonic signal power levels of the transmit DDS filter module	31
3.11 Bandwidth of the transmit DDS frequency multiplier module	31
3.12 Bandwidth of the transmit DDS amplifier module	32
3.13 Functional block diagram of the LO sub-chain	33
3.14 Output signal power level of the LO DDS module	34
3.15 Harmonic signal power levels of the LO DDS filter module	35
3.16 Bandwidth of the LO DDS frequency multiplier module	35
3.17 Output signal power level of the LO DDS amplifier module	36
3.18 Bandwidth of the IF filter module	37
3.19 Functional block diagram of the new RSP	39
3.20 Block diagram of the timing distribution in the transceiver	41
3.21 The output signal of the transmit DDS performing a 90° phase shift.	44
3.22 An example of a binary phase-coded signal.	45
3.23 Time-domain response of an optimized phase-coded compressed pulse	46
3.24 Time-domain response of a 32-bit optimal binary code obtained in the Radar Handbook (Merril I. Skolnik, page 10.18)	47
3.25 Figure of the phase-coded control signal waveform code	48
3.26 A picture of the dead zones at the output of the transmit frequency multiplier module (phase offset = 90°)	49
3.27 A zoomed picture of the dead zones	50
3.28 Implementation of a 10° phase offset in the DDS and resulting RF output.	51
3.29 Block diagram of the pulse compressor module	52
3.30 PRI Burst Map layout and processing directions	54
3.31 Block diagram of the hardware setup for the pulse compression test	56
3.32 Graph of the software vs. hardware simulated pulse compression	57
3.33 Graph of the error between the software and hardware simulated pulse compression	58
B.1 Block diagram of the 9-times frequency multiplier module	B-2
B.2 Photo of an open 9-times frequency multiplier module enclosure showing the PCB.	B-3

B.3	Photo of the 9-times frequency multiplier module enclosure with both lids secured.	B-3
B.4	A picture of the Analog Devices DDS DAC Output evaluation board (AD9858PCB)	B-4
B.5	Picture of an opened filter module	B-5
B.6	Picture of closed filter modules	B-6
B.7	Block diagram of the modified 9-times frequency multiplier module to function as an amplifier. . .	B-6
B.8	Block diagram of the IF amplifier module	B-7
B.9	Picture of the <i>Altera Stratix EP1S25 DSP development board</i> [5]	B-8
C.1	Summary information of the <i>SYNERG Microwave Corporation</i> power dividers (DSK-726S)	C-1
C.2	Functional block diagram of the Analog Devices AD9858 DDS	C-2
C.3	Summary information of the <i>Mini-Circuits</i> amplifier (MAN-1HLN)	C-2
E.1	ERS Aircraft Decision Circuit block diagram.	E-3
E.2	Frequency domain pass-band of the Doppler filters in the SPU.	E-4
E.3	Graph of the probability of detection for the ERS.	E-7
E.4	Effect of the non-ideal transmit and receiver chains on the compressed pulse.	E-10

List of Tables

D.1	Table of calculated radar parameters for the ERS	D-1
E.1	Table of defined radar parameters for the Eekhorning system	E-1
E.2	Table of relevant SNR parameters	E-2
E.3	Summary table of ERS pulse integration parameters	E-5
E.4	Table of the MDS parameters	E-7
E.5	Table of the MDS parameters for the upgraded system	E-10

List of Abbreviations

COTS	Commercial Of The Shelf
CFAR	Constant False Alarm Rate
dB	Decibel
°	degrees
DDS	Direct Digital Synthesizer
DP	Doppler Processor
DSP	Digital Signal Processing
ERS	<i>Eskoring</i> Radar System
FIFO	First-In, First-Out
FPGA	Field Programmable Gate Array
GPS	Global Positioning System
Hz	Hertz
I	In-phase
IAGC	Immediate Automatic Gain Control
IF	Intermediate Frequency
J	Joule
km	kilometer
LED	Light Emitting Diode
LNA	Low Noise Amplifier
LO	Local Oscillator
LTA	Long Time Average
Ltd	Limited
m	meter
MDS	Minimum Detectable Signal
MHz	Mega Hertz

List of Abbreviations

PC	Pulse Compressor
PCB	Printed Circuit Board
PLL	Phase Lock Loop
PPI	Plan Position Indicator
PRF	Pulse Repetition Frequency
PRI	Pulse Repetition Interval
Q	Quadrature-phase
RCS	Radar Cross-Section
RF	Radio Frequency
RFE	Radar Front End
RRS	<i>Reutech Radar Systems</i> (Pty) Ltd., Technopark, Stellenbosch, South Africa.
RSP	Radar Signal Processor
Pty	Properties
rpm	revolutions per minute
RMS	Root-Mean-Square
s	second(s)
SANDF	South African National Defense Force
SNR	Signal to Noise Ratio
SPU	Signal Processing Unit
STA	Short Time Average
STC	Sensitivity and Time Control
TLA	Three Letter Acronym
μ	micro
W	Watt

List of Symbols

π	3.1415926535897932384626433832795
B	pulse bandwidth
c	speed of light in a vacuum
f_{base}	RSP base frequency
f_{sample}	RSP sample frequency
f_{mod}	pulse-code modulation frequency
f_{IF}	intermediate frequency
f_{PRF}	pulse repetition frequency
f_{tx}	antenna transmission frequency
F	system noise figure
G_a	available gain
G_{ant}	antenna gain
G_r	receiver antenna gain
G_t	transmit antenna gain
k	Boltzmann's constant or a constant
λ_{tx}	transmission wave-length
L_{CFAR}	CFAR loss
L_{comp}	pulse compression loss
L_{int}	integration loss
L_m	matching loss
L_{rx}	receiver loss
L_s	system losses
n_{int}	integration gain
ψ_o	RMS noise power
P_d	probability of detection

List of Symbols

P_{fa}	probability of false alarm
P_t	peak transmitter power
R	range
R_{blind}	radar blind range
R_{max}	maximum detectable range
$R_{unambib}$	unambiguous range
σ	target radar cross-section
τ_{PRF}	pulse repetition interval
τ_{tx}	transmit pulse-width
T_{ant}	antenna noise temperature
v_m	target blind velocity
V_t	threshold voltage

Chapter 1

Project Overview

Most of the disciplines of electrical, electronic and mechanical engineering are applied during the design and implementation of a radar system. This makes the research field of radar systems very wide. One of the largest study areas in these systems is signal processing, better known as radar signal processing.

The aim of the project was to acquire some basic knowledge into the field of radar engineering and more in particular radar signal processing. REUTECH RADAR SYSTEMS (PTY) LTD. (RRS) proposed a project, to be done as a masters thesis project, which would investigate the implementation of a RADAR SIGNAL PROCESSOR (RSP) on a single FIELD PROGRAMMABLE GATE ARRAY (FPGA) device.

The research and design were carried out with the assistance of RRS at their facilities in Technopark, Stellenbosch, South Africa. RRS has many years of experience in radar system development which proved to be vital during the life of this project.

Radar signal processing is a very processing intensive task. Processing hardware such as Digital Pulse Compression and Doppler Processing consumes a large amount of dedicated processing units to meet real-time requirements. FPGA technology has improved significantly in the last few years. This resulted in FPGAs that are far more capable, bigger, faster and which include special features such as DIGITAL SIGNAL PROCESSING (DSP) algorithmic blocks. These powerful features make FPGA devices ideal for implementation in radar signal processing.

The project was originally defined with the following objectives in mind:

- To implement a RSP, with Digital Pulse Compression and Doppler Processing capabilities, on a single FPGA-based device.
- A working existing radar system was to be used for implementing the modifications on, thus acting as a development bench.
- The improvement of the limited range resolution of the existing system would be the main focal point. To improve this resolution, both Digital Pulse Compression and Doppler Processing were to be implemented and were the requirement for all the modifications done.
- COMMERCIAL OFF THE SHELF (COTS) components were to be used as far as possible to keep the implementation cost and time to a minimum.

The existing radar system chosen for this project was the EEKHORING RADAR SYSTEM (ERS) which is a small, short range radar system with limited capabilities due to design. These limitations include the limited range resolution of the signal processor. At first sight the simplicity of design and ease of use of this system made it seem an excellent choice to implement this project on.

However, the original aims of the project had to be adapted due to a large number of modifications that were needed to the RF sub-system of the existing radar system in order to implement pulse compression. In the end this resulted in more than 50% of the work that had to be done, where it was initially first thought to be minimal. A thorough study of the field of radar RF engineering was also required to design and implement the upgrades to the existing RF sub-system.

The project design time was consequently pushed far over the time originally projected and resulted in the newly designed RSP not being fully implemented due to the scope of the work. This is unfortunate, because ultimately this project was intended as a study of the radar on a system level.

The final aim of the project was to use an existing radar system and to implement pulse compression techniques to improve the range resolution of the system.

This document starts by giving an overview of the existing radar system in Chapter 2. The three (3) modules (Antenna, Transceiver and Display modules) that make up the ERS are then discussed in detail with regards to their design and capabilities.

Special depth is provided on the design of the Transceiver module sub-chains since this is the focus area for modifications. This detailed study was needed so as to be able to identify the areas which would be modified and gives valuable insight in the system. The specifications of the transceiver are given in Appendix (A) at the end of this document.

Finally the existing signal processing is discussed to provide information on the detection methods and capabilities implemented in the existing system.

The document proceeds in Chapter 3 with an overview of the modifications made to the system in order to implement pulse compression. An in depth discussion of the RF changes to the transmit and receive chains of the RF sub-system is given and the newly implemented hardware modules and results achieved are discussed. The functional design of each new hardware module is given in Appendix (B).

The newly implemented firmware based RSP is discussed next and starts off with an overview of the signal processing techniques to be implemented. Focus is put on IF sampling, the planning of frequencies and timing signals as well as the synchronization of the radar system. A full discussion of the design and implementation of a pulse code used for pulse compression is given. The results achieved in the RF sub-system during the modulation implementation are presented and discussed.

The designed and implemented firmware RSP is discussed on a functional level, followed by the testing and results of the pulse compressor section. These results show the design of a 32-bit pulse compression code by means of a software program. A test signal containing the designed pulse code was used to test both a software and hardware simulated version of the pulse compressor section. Both versions successfully implemented pulse compression and results were found to be almost identical. This is also presented in Chapter 3.

Extensive information not necessarily directly related to the project is given in the Appendices. These include technical information, hardware module descriptions, supplement hardware information, various calculated radar parameters and firmware code listings of the RSP modules.

It is important to understand the mathematics behind radar systems and radar signal processing. Appendix E gives an overview of the relevant radar mathematical calculations providing a way to quantify the difference between the existing and upgraded systems.

Chapter 2

Existing Eekhorning Radar System

2.1 Introduction

In this chapter the existing EEKHORING RADAR SYSTEM (ERS) will be discussed. Firstly an overview of the system will be presented giving a short history of the system, the operational use and a brief system overview design. Next the three system modules (Antenna, Transceiver and Operator) will be discussed in detail. The chapter is completed with a detail discussion on the operation of the existing signal processor in the ERS.

2.2 Eekhorning Radar System Description

The ERS (figure (2.1)) is a man-portable, short range radar system operating in the L-band (1-2 GHz) of radar frequencies. The ERS was preceded by an earlier version known as the *Tupperware Systems* at the end of the 1970s. This system was developed in the mid 1980s and has since been operational in the SOUTH AFRICAN NATIONAL DEFENCE FORCE (SANDF) at the 10 Air Defense Regiment stationed in Kimberley, South Africa.



Figure 2.1: The Eekhorng Radar System (man-portable setup)

The unit can operate at two defined maximum range settings of 10km and 20km. Selecting the first improves the range resolution to allow the operator to designate a target with higher accuracy. The range resolution is 1km per range bin for the 10km setting and 2km per range bin for the 20km setting and is determined by the ten (10) range filter circuits in the signal processor.

The rotational speed of the antenna can also be selected: Options of 10rpm or 30rpm are available with the latter giving a higher update rate due to the higher sweeping rate, but the detection range is degraded as less reflected power off a target is received.

A selection of eight (8) operation transmission frequencies are available which makes this system less susceptible to radar jamming attempts since the operational frequency bands of both the transmit and receive chains can be switched instantly. Filter banks in the receive chain are used to accommodate different transmission frequencies.

The radar system is relatively small in size and is designed to be man-portable. Its rugged design is a large plus point in the battlefield environment. The system was also later mounted on a vehicle which allowed it to be deployed in more harsh landscapes than other radar systems in use. Training of personnel to operate the system is also very easy due to its simple design and ease of use.

This radar system is very well suited to be used as a development- and test setup for the development of Pulse-Doppler radar signal processors[1].

Figure (2.2) shows the top level block diagram of the ERS indicating the three (3) main modules of the system. The inter-connections between the modules are also indicated.

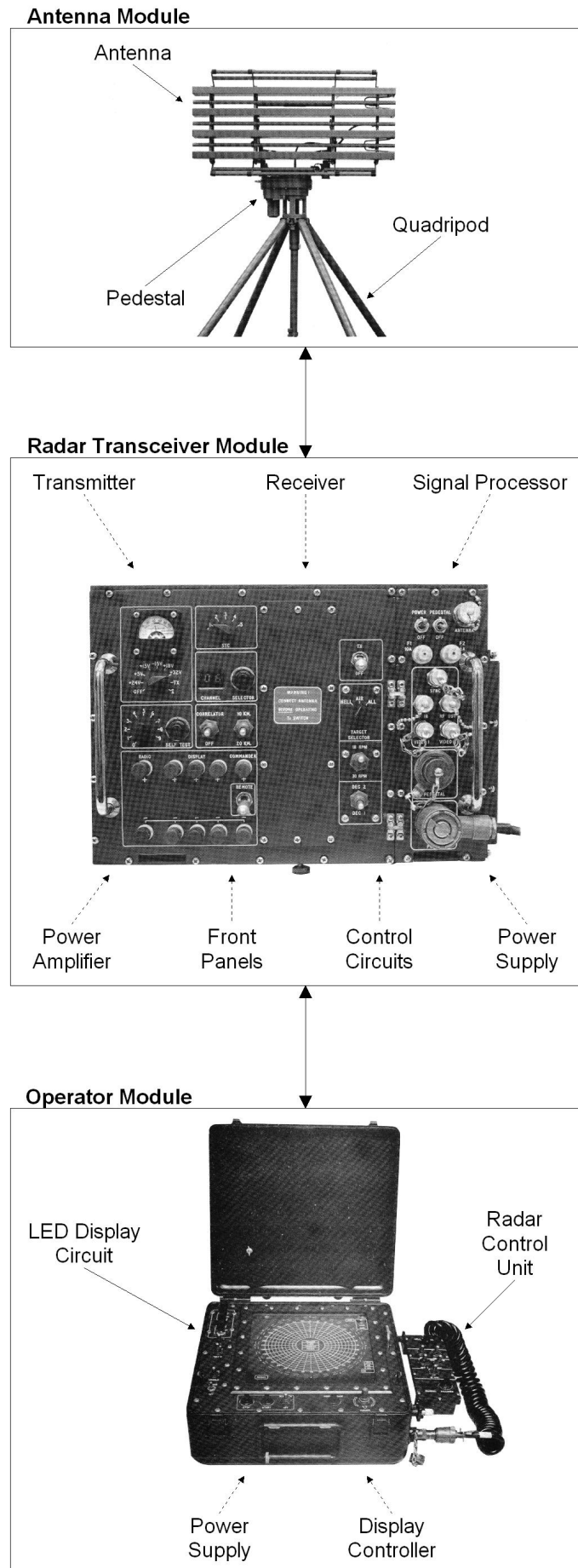


Figure 2.2: Eekhorng Radar System top level block diagram

The ERS consists of three (3) main modules: The antenna, transceiver and operator modules. These will be discussed in the next section.

2.3 Eekhorng Radar System Modules

2.3.1 Antenna Module

The antenna module comprises of the antenna, the rotating pedestal and the quadripod used to mount the radar system.

The antenna is made up of four (4) horizontal planks which produce an azimuth beam width of 9° at the -3dB points. The elevation of the antenna is adjustable from -5° to $+5^\circ$ allowing the antenna mean to be moved up or down depending on the surrounding terrain. The elevation coverage diagram of the antenna beam is shown in figure (2.3)

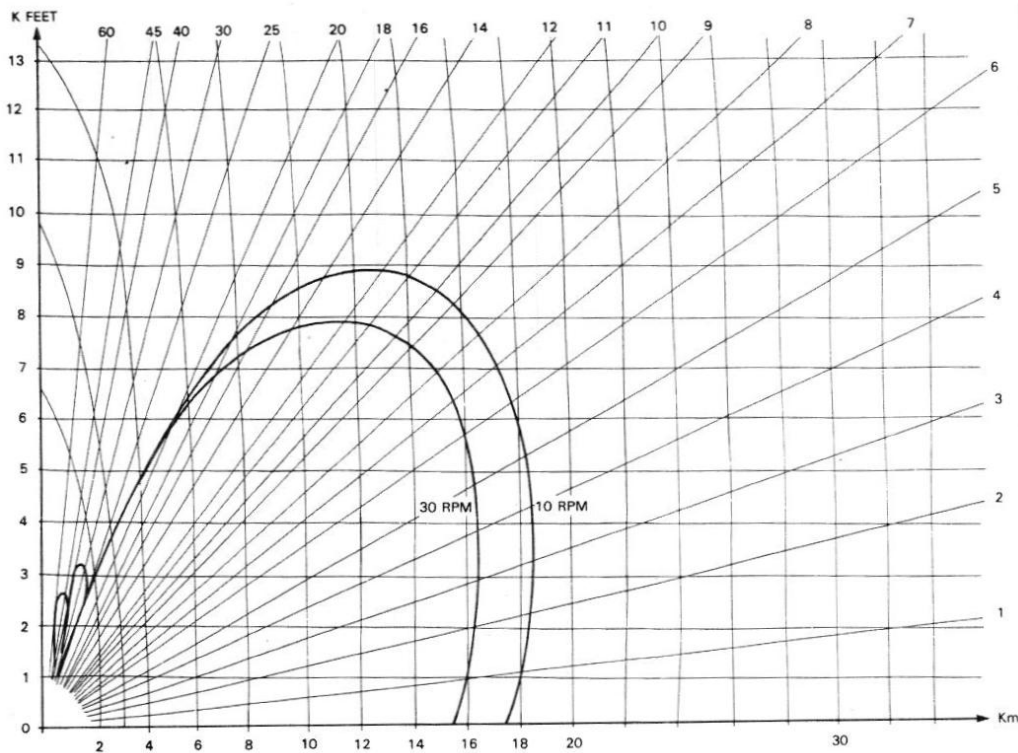


Figure 2.3: Antenna beam elevation coverage diagram ($+3^\circ$ tilt)

The pedestal can operate at two (2) preset speeds: 10rpm and 30rpm. The RF feed connection to the antenna is done by means of a rotary joint in the middle of the pedestal. The antenna is secured on the rotating pedestal while the pedestal itself is secured onto the quadripod.

The quadripod is a rugged and light weight stand for the radar system (refer to figure (2.1)). The transceiver module is mounted at the base of the quadripod which allows the system to have a very compact setup and short RF cable to minimize cable loss.

Antenna turning data of a north-pulse and two (2) antenna position pulses are routed from the antenna module to the transceiver module by means of a cable connection. Refer to figure (2.4) for an image of the antenna module.



Figure 2.4: The ERS antenna module

2.3.2 Transceiver Module

The radar transceiver module is of particular interest and is discussed in more detail. It includes the transmit and receive RADIO FREQUENCY (RF) chains, a Doppler RADAR SIGNAL PROCESSOR (RSP), timing and control circuits and the power supply unit. The transceiver module is shown in figure (2.5).

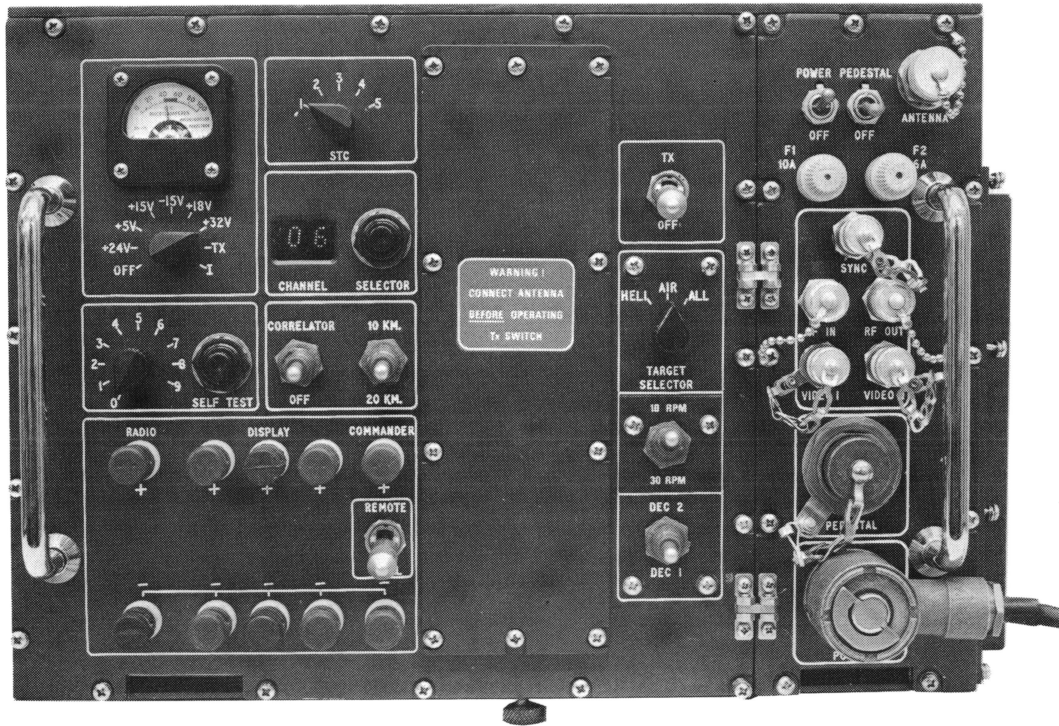


Figure 2.5: The ERS transceiver module

Figure (2.6) shows the functional block diagram of the transceiver module as discussed in the next sections.

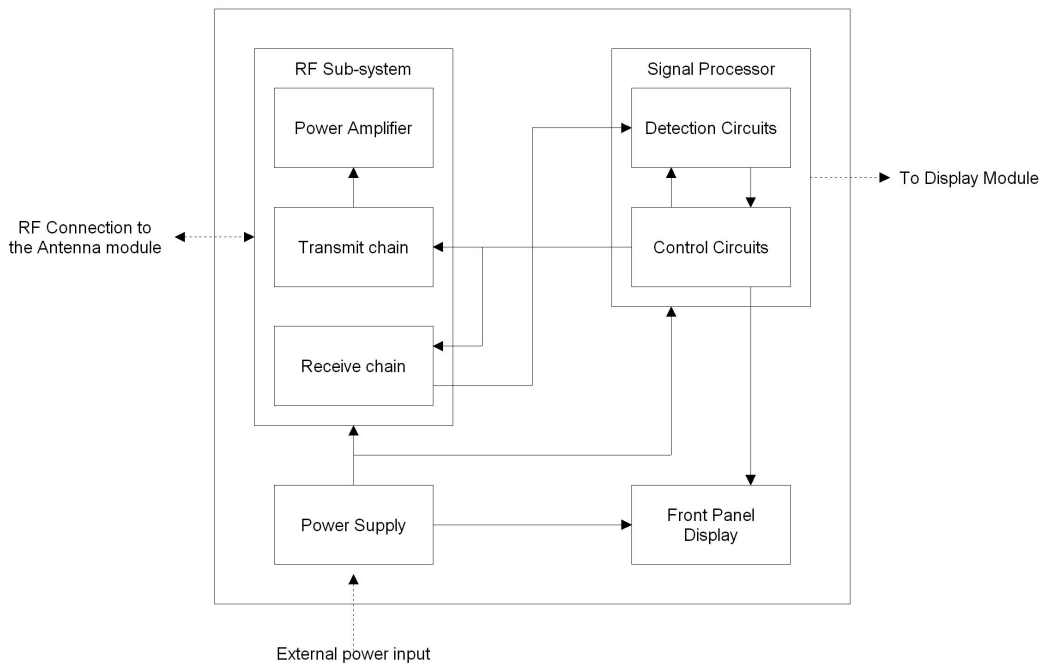


Figure 2.6: Functional block diagram of the transceiver module

2.3.2.1 RF Sub-system

Figure (2.7) shows the simplified block diagram of the RF sub-systems. The RF subsystem consists of the transmit and receiver chains which are interconnected and also connected to the Antenna module by means of three (3) circulator modules used to route outgoing and incoming signals to the respective RF chains and the antenna.

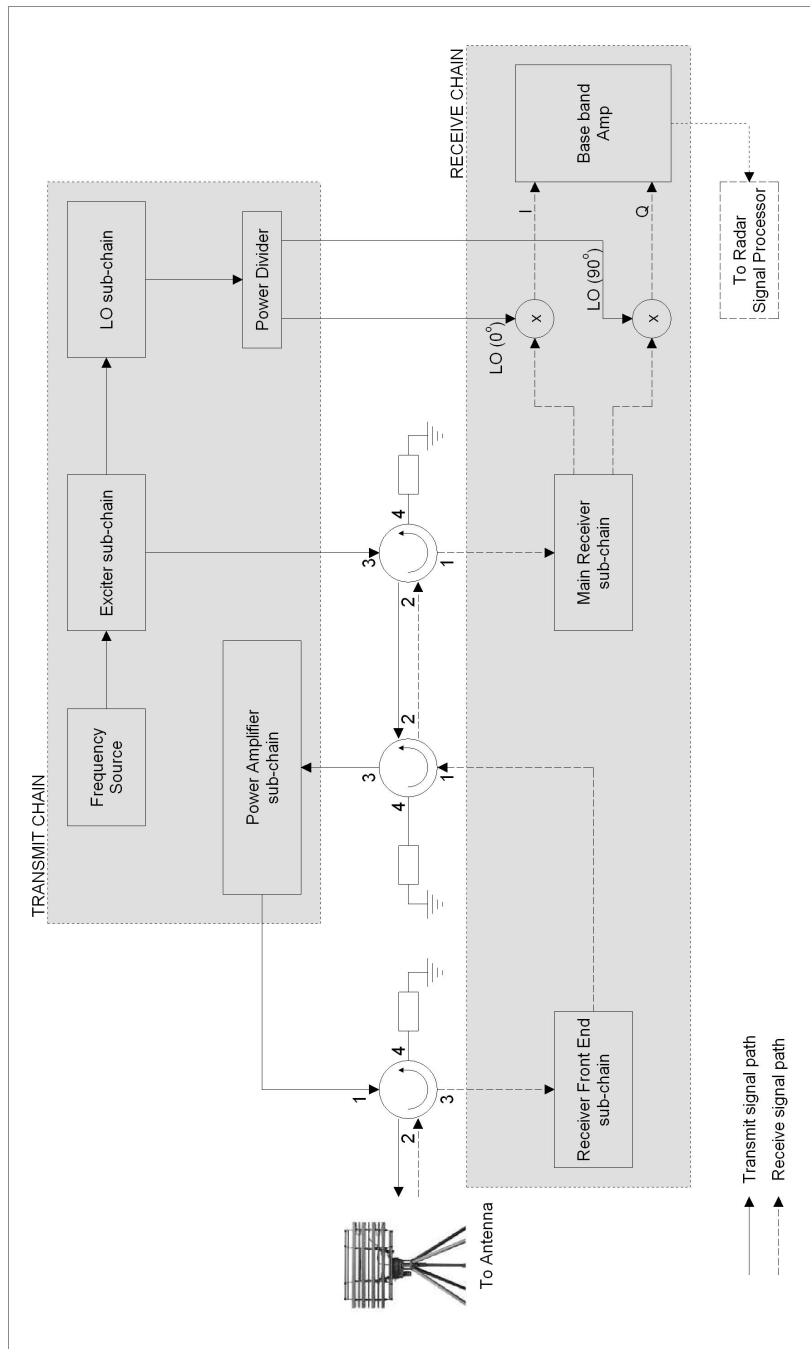


Figure 2.7: Top-level RF block diagram

The signal flow through the RF subsystem is as follows:

The frequency source generates the base frequency which is routed to the transmit chain. The transmit pulse is generated, multiplied and amplified to the correct frequency and power level in the transmit chain before being transmitted by the antenna module. The signal is routed through three (3) circulators during this process.

The returning echo is received by the antenna module and routed to the receiver chain by means of the same circulators as above. In the receiver chain the returning signal is amplified and mixed down to base band by means of the local oscillator signal generated in the transmit chain.

Finally the base band signal is routed to the signal processor in the ERS transceiver.

An interesting aspect of the design of the the RF sub-system is the use of three (3) circulator modules for signal routing. The two circulators used to connect the exciter to the power amplifier sub-chains and the receiver front end to the main receiver sub-chains are clearly not needed since these sub-chains could have been connected directly.

This is possibly due to a previous version where the power amplifier and receiver front end sub-chains were located in the antenna module. The antenna rotary joint was thus located between these sections of the RF sub-system and moved in this design.

2.3.2.1.1 Transmit Chain

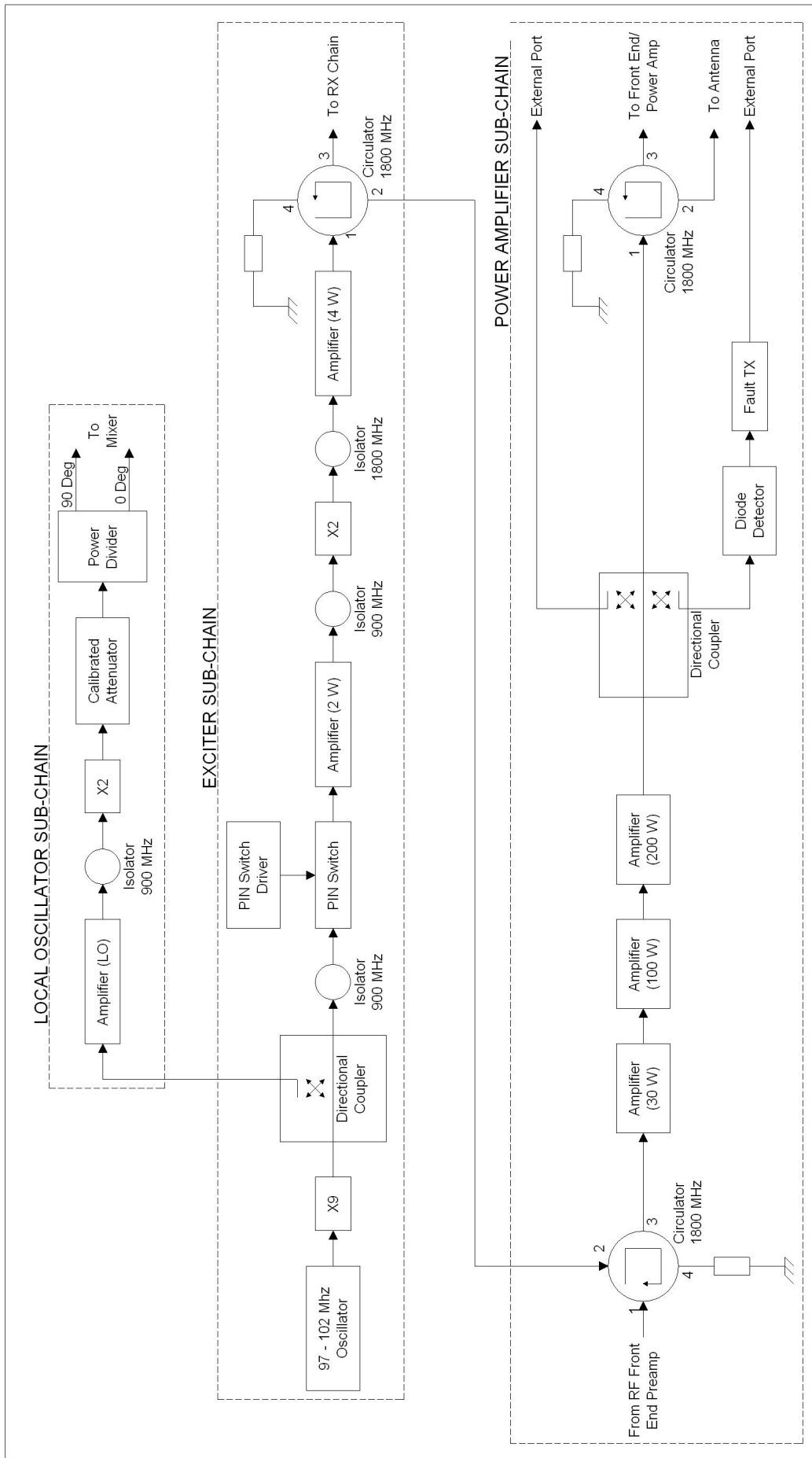


Figure 2.8: Transmit chain RF block diagram

The transmitter RF block diagram is shown in figure (2.8) (adapted from [1]). The transmitter chain consists of three (3) sub-chains:

2.3.2.1.1.1 Exciter sub-chain

Refer to the Exciter sub-chain section of figure (2.8).

In this chain the RF source, a crystal oscillator with 8 selectable frequencies, is multiplied nine (9) times and fed into a directional coupler. This coupler outputs the signal via the main port to the power amplifier sub-chain while the second attenuated port is connected to the LO sub-chain.

The actual transmit pulse width of either $6,6\mu s$ or $13,3\mu s$ (depending on the operational range setting) is generated by means of the PIN switch and amplified before being routed to the two (2) times multiplier. The transmit signal, now at the actual transmit frequency of between 1,75-1,85GHz passes through a second internal amplification stage. Three (3) isolators are implemented to protect the preceding modules in the event that power should be reflected. The isolators allow forward transmission of the signal, but largely attenuates it in the reverse direction.

The signal is physically connected from the output of the exciter chain to the power amplifier chain by means of two circulators[1].

2.3.2.1.1.2 Local Oscillator sub-chain

Refer to the LO sub-chain section of figure (2.8).

The LO sub-chain is used to generate a signal for the receive chain to be able to mix the received signal down to base band. The LO frequency is derived from the exciter chain, amplified and multiplied by two (2) to produce the same frequency as the transmitted pulse. The signal then passes through a calibration attenuator and splits into two signals (in-phase (I) and quadrature-phase (Q)) by a power divider. These two outputs are then directly fed into the receiver mixers where the received signal is mixed down to base band[1].

2.3.2.1.1.3 Power Amplifier sub-chain

Refer to the Power Amplifier sub-chain section of figure (2.8).

The transmit pulse generated in the exciter chain is amplified here to the transmit power level and routed to the antenna. The amplification is done in three (3) stages: Firstly, a 30W amplifier stage, followed by a 100W stage and finally a 200W amplifier producing a minimum transmit power of 115W at the RF output connector. The high power signal is fed to the antenna through a dual directional coupler. One port is connected to a detector diode to continuously monitor the level of the transmitter power and provide information in the event it should fail. The second port is used for test purposes[1].

The power amplifier chain is connected to the antenna by means of a circulator to route the outgoing signal to the antenna.

2.3.2.1.2 Receive Chain

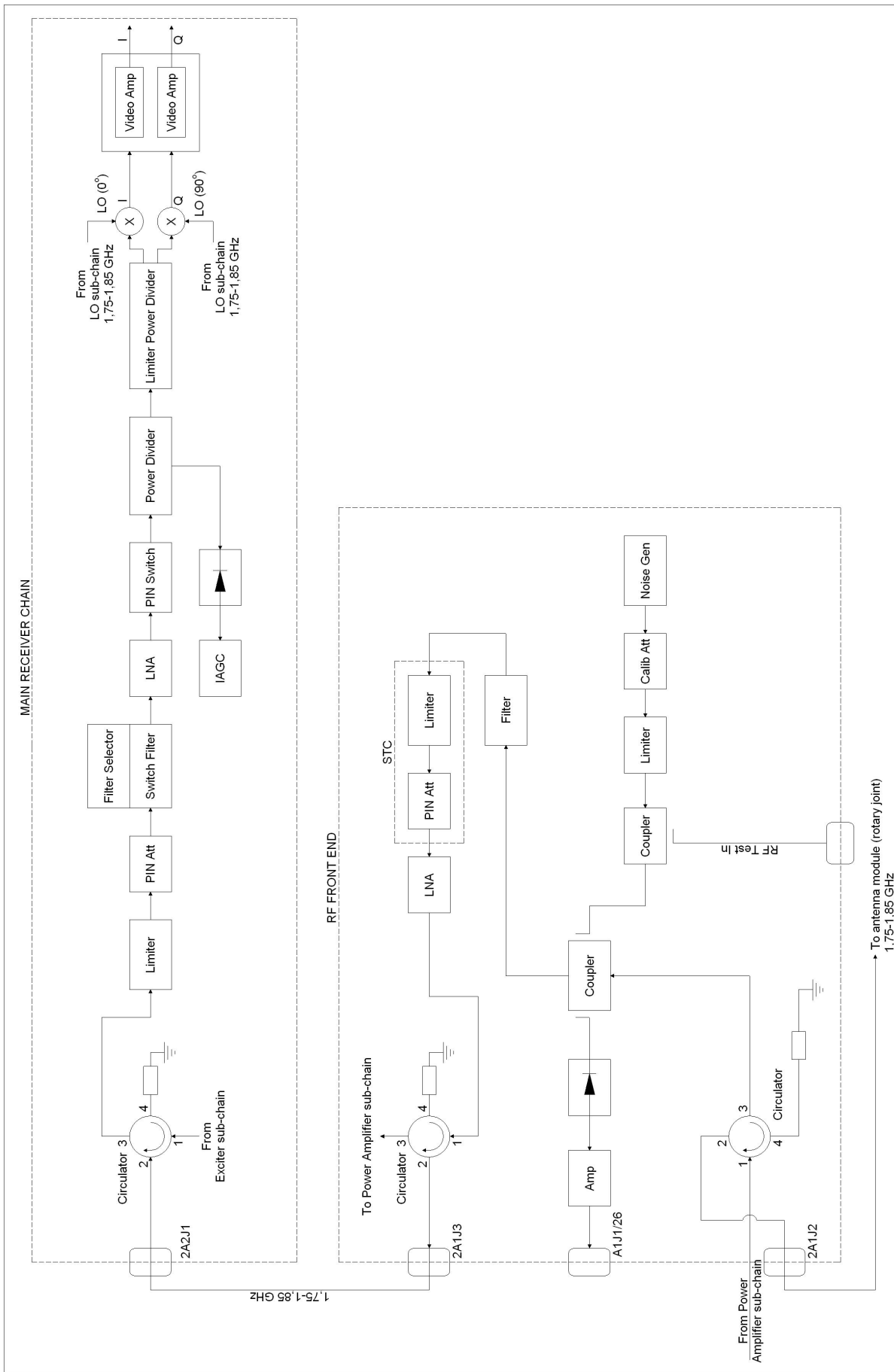


Figure 2.9: Receive chain RF block diagram

The receiver as shown in figure (2.9) (adapted from [1]) consists of two (2) sub-chains:

2.3.2.1.2.1 Receiver Front End sub-chain

Refer to the Receiver Front End sub-chain section section of figure (2.9).

The RECEIVER FRONT END (RFE) chain is used to amplify the small returning signal whilst adding minimal noise to the signal. It is fed from the antenna via the same circulator which connects the transmit chain to the antenna module. The received signal is passed through a dual port directional coupler which is used for maintenance functionality (self test injection and VSWR fault detection). Next the signal passes through a band pass filter bank and limiter modules.

Next a PIN attenuator module is used for the implementation of the INSTANTANEOUS AUTOMATIC GAIN CONTROL (IAGC) and SENSITIVITY TIME CONTROL (STC) features of the receiver. The IAGC is used to prevent the receiver from saturation as result of large received signals due to clutter. The STC on the other hand varies the sensitivity of the receiver as a function of time (and thus range). Reflected signals of objects close to the transmitter will be attenuated more than those of objects further away. This effectively removes excessive clutter close to the receiver. The level of the STC is also selectable as one of five (5) positions.

Lastly, the signal is amplified by a LOW NOISE AMPLIFIER (LNA) and passed to the main receiver sub-chain via a circulator [1].

2.3.2.1.2.2 Main Receiver Chain

Refer to the Main Receiver sub-chain section section of figure (2.9).

The function of this chain is to mix the received signal down to base band and add sufficient amplification to the signal before it is routed to the signal processor.

The received signal is firstly limited due to possible high power leakage into the receiver. A PIN attenuator is implemented to minimize leakage during the transmission period.

A filter bank band passes the signal according to the selected frequency. This is done to limit the noise in the system. The signal is now amplified again by means of a LNA unit.

The signal is then passed through a PIN switch which operates as the inverse of the PIN switch in the transmitter and so prevents the mixers from damage during the transmission period. Thus if the transmitter is on, the receiver is switched off.

The signal is then split by the power detector. One half is used to operate the IAGC while the other is split again (by the power divider) and used to feed the mixers which down converts the received signal to base band I and Q signals. Lastly the I and Q signals are amplified in the video amplifier and routed to the signal processor[1].

2.3.2.2 Signal Processor Unit

The purpose of the SIGNAL PROCESSOR UNIT (SPU) is to convert the raw received radar signal into usable information for the operator by means of the radar display. The SPU consists of eleven (11) analogue and digital cards consisting of the Sync card, Pulse Repetition Frequency (PRF) Filters, Decision Correlator and Line Transceivers. All these cards plug into a motherboard back-plane which provides communication channels between the cards.

The SPU receives all the returning radar signals which includes all clutter and target information. The signal is firstly processed by means of PRF Filters to eliminate all clutter returns and returns of targets with velocities less than 180km/h. These filtered signals are then routed to the aircraft decision circuits. A further parallel filter is used to filter all Doppler returns below 400km/h so to eliminate helicopter body returns. The filtered signal now contains blade flash returns of speeds higher than 400km/h and is routed to the helicopter decision circuit.

The PRF Filters (also known as Doppler Filters) consist of four (4) cards - two for the I channel and two for the Q channel. These remove clutter and produce ten (10) range gate sampling circuits per filter card which result in the range resolution of 1km/bin for the 10km range setting or 2km/bin for the 20km range setting.

Decision making is done per range cell. The energy in each cell is subjected to an energy based decision and followed by the summation of the I and Q channels. Positive target decisions are indicated by means of a positive logic signal for each target detected. Mid-range decisions are subject to the same process and all output signals are fed to the correlator unit.

The correlator combines both helicopter and aircraft decision channels and performs scan to scan correlation on this data to eliminate possible false positives.

Finally all the processed information is send to the Line Transceiver which modulates the data and routes it to the display unit[1]. The SPU block diagram is shown in figure (2.10) (adapted from [1]).

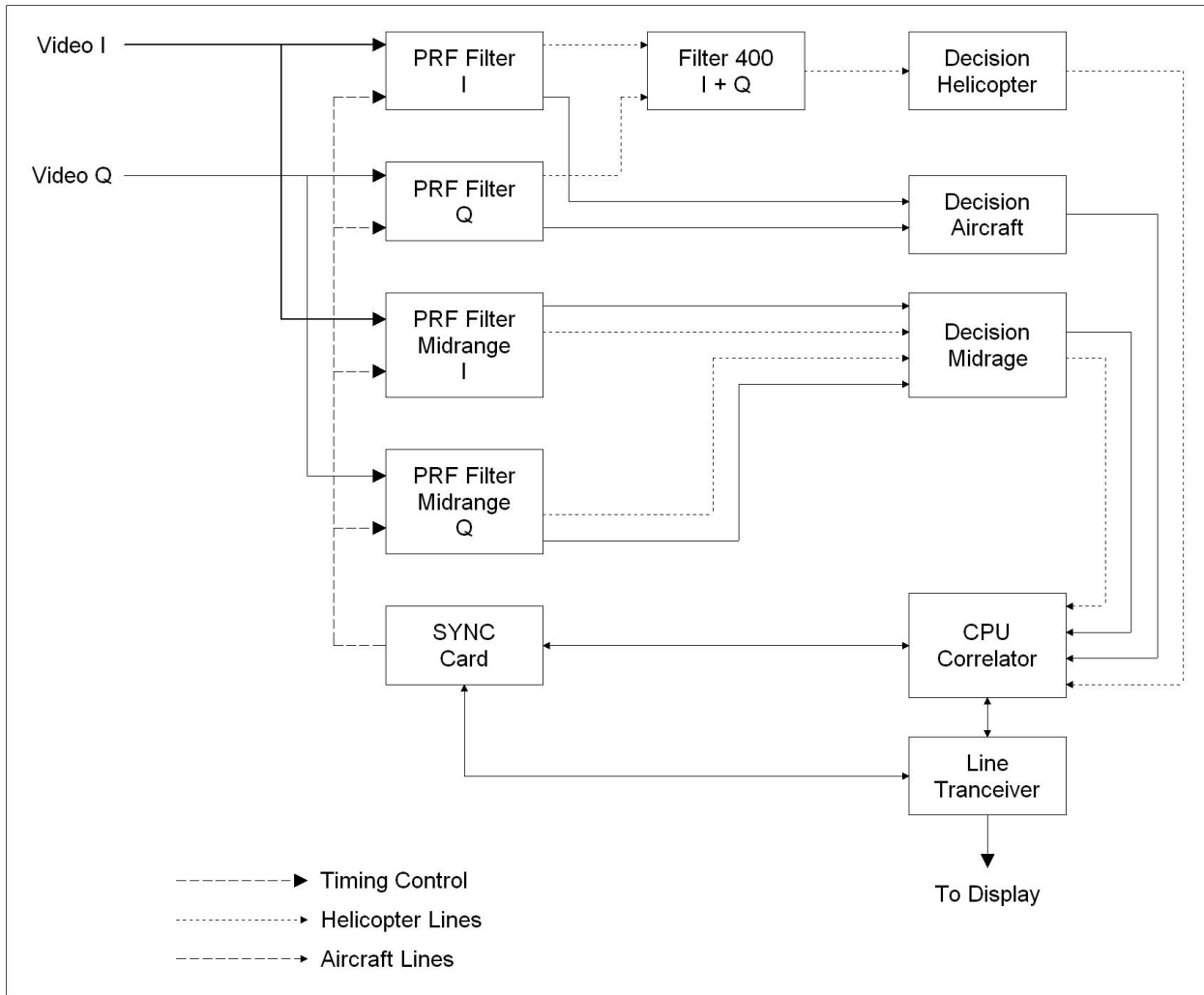


Figure 2.10: Signal Processing block diagram

2.3.2.2.1 Target Detection Target detection is done in the signal processor by means of 20 parallel and identical detection circuits called "Decision Circuits". Two (2) circuits per range bin are used (and triggered accordingly): One for I-channel data and the other for Q-channel data. These 20 detection circuits are duplicated to result in detection circuits for both fixed wing aircraft and helicopters resulting in 40 circuits.

Figure (2.11) shows the block diagram one of these detection circuits used in the aircraft detection channel.

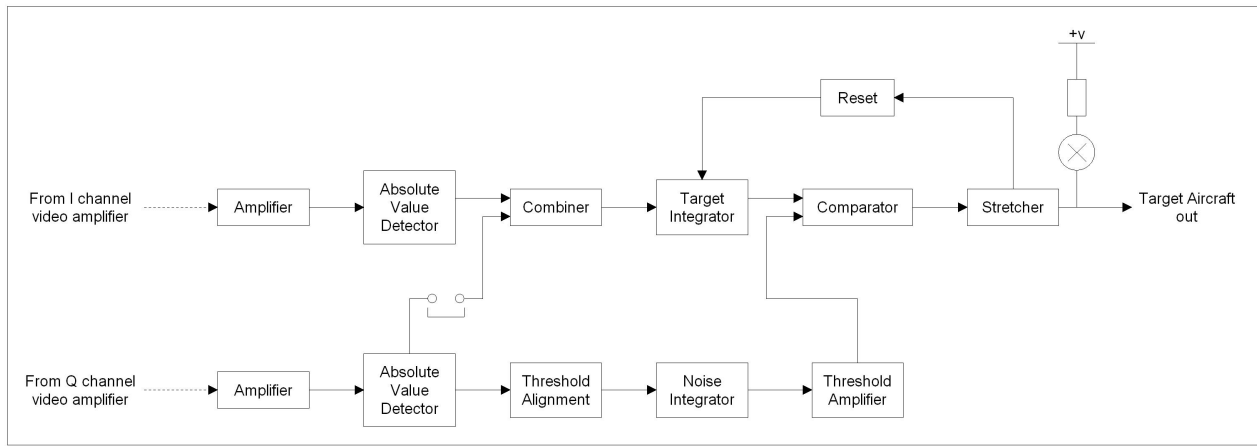


Figure 2.11: Block diagram of a single aircraft detection circuit

The absolute value from both the I-channel and Q-channel are combined to produce an approximation of the magnitude, i.e. $|I| + |Q| \sim \sqrt{I^2 + Q^2}$. Integration of this magnitude is done in the target integrator. This integrator is cleared at the end of each sector break of 10° .

Noise is also continuously integrated using the noise integrator which is supplied by means of the Q-channel data. This gives a measurement of the noise present in the system. This integrator is also not reset as is the case in the target integrator.

When a target is encountered, both the target and noise integrators rise due to the returning signal. The value of the target integrator rises quicker than the value of the noise integrator due to a shorter rise time. The output of the target and noise integrators are compared and if the target integrator value is higher than the value of the noise integrator, a target is declared in the particular range bin. The detection is routed via the signal processor to the Operator module where the corresponding light will be lit in azimuth and range to indicate a target.

This process is identical in all ten (10) range bin detection circuits for both the aircraft and helicopter channels.

This design is very complex since a large number of detection circuits are used. It could easily happen that some of these circuits are faulty or biased incorrectly without the user knowing this.

2.3.2.3 RF performance measurements

Various RF measurements were planned for the ERS. These included the measurement of the MINIMUM DETECTABLE SIGNAL (MDS) of the receiver which is specified to be -115dBm . RRS attempted this measurement, but it was not possible with the available test equipment. The value of -115dBm however seems rather high and a mathematical approximation of the MDS was calculated to be -128dBm (refer to Appendix (E.2.1)).

2.3.3 Operator Module

The operator module includes the PLAN-POSITION INDICATOR (PPI) and remote control units.

The PPI consists of a display with concentric circles of LIGHT EMITTING DIODES (LED) to indicate the target bearing and range. The PPI was previously replaced by RRS with an integrated PPI and remote control unit which is personal computer based.

This upgraded PPI runs on a laptop computer and has the same functionality as both the original PPI and remote control units. As an additional feature, a GLOBAL POSITIONING SYSTEM (GPS) unit was added by RRS to fix the radar system's location. Using the remote operation setting, the transceiver module can be fully controlled and configured from the operator module.

Figure (2.12) shows the Operator module.



Figure 2.12: Original ERS Operator Module

2.4 Conclusion

In this chapter the existing ERS was discussed on modular level to provide a base for the modified version discussed in Chapter 3. Emphasis was placed on the transceiver module and more in particular the transmit and receiver chains as well as the signal processor since all the modifications needed to implement pulse compression were done in this module.

Chapter 3

Modified Eekhorng Radar System

3.1 Introduction

This project aims to use the existing ERS as a development bench and implement advanced digital signal processing techniques with the main improvement being the improvement of the limited range resolution of the system. In the existing system, the range resolution is defined by ten (10) fixed range bin detection circuits of the signal processor. This results in a fixed range resolution per operational range setting.

The existing signal processor is replaced with a new firmware based signal processor running on a single FPGA device. The main upgrade feature of the new signal processor is the implementation of Digital Pulse Compression and Doppler Processing techniques to improve the range resolution.

Implementing pulse compression however required a relative large number of modifications to the RF subsystem of the ERS. The amount of RF modification work required was largely underestimated and resulted in the project taking much longer than what was anticipated.

This chapter firstly describes the RF subsystem modifications and the RF modules required to implement these modifications. Secondly an overview of the newly designed RSP for the implementation of the techniques above is given. This RSP is designed, but only partially tested due to time constraints. The pulse compression section implemented will be discussed with results.

3.2 Transceiver Modifications

Only the transceiver was modified in this project leaving the antenna and operator modules unchanged. (The operator module was to be an existing PC based moving target indicator (MTI) processor to indicate detections and show target movement and information. This was not implemented.)

At first the changes to the RF subsystem of the transceiver was envisaged to be minimal. However as the project progressed it was found that a large number of modifications had to be made to the transmit and receive chains to be able to implement pulse compression techniques and thus improve the range resolution of the ERS.

The existing signal processor was replaced entirely with a FPGA based digital signal processing board. Some existing timing signals were utilized to keep the changes to the RF subsystem to a minimum. Frequency planning was done to ensure that the system is synchronous.

Figure (3.1) shows a block diagram of the interconnections of the upgraded ERS.

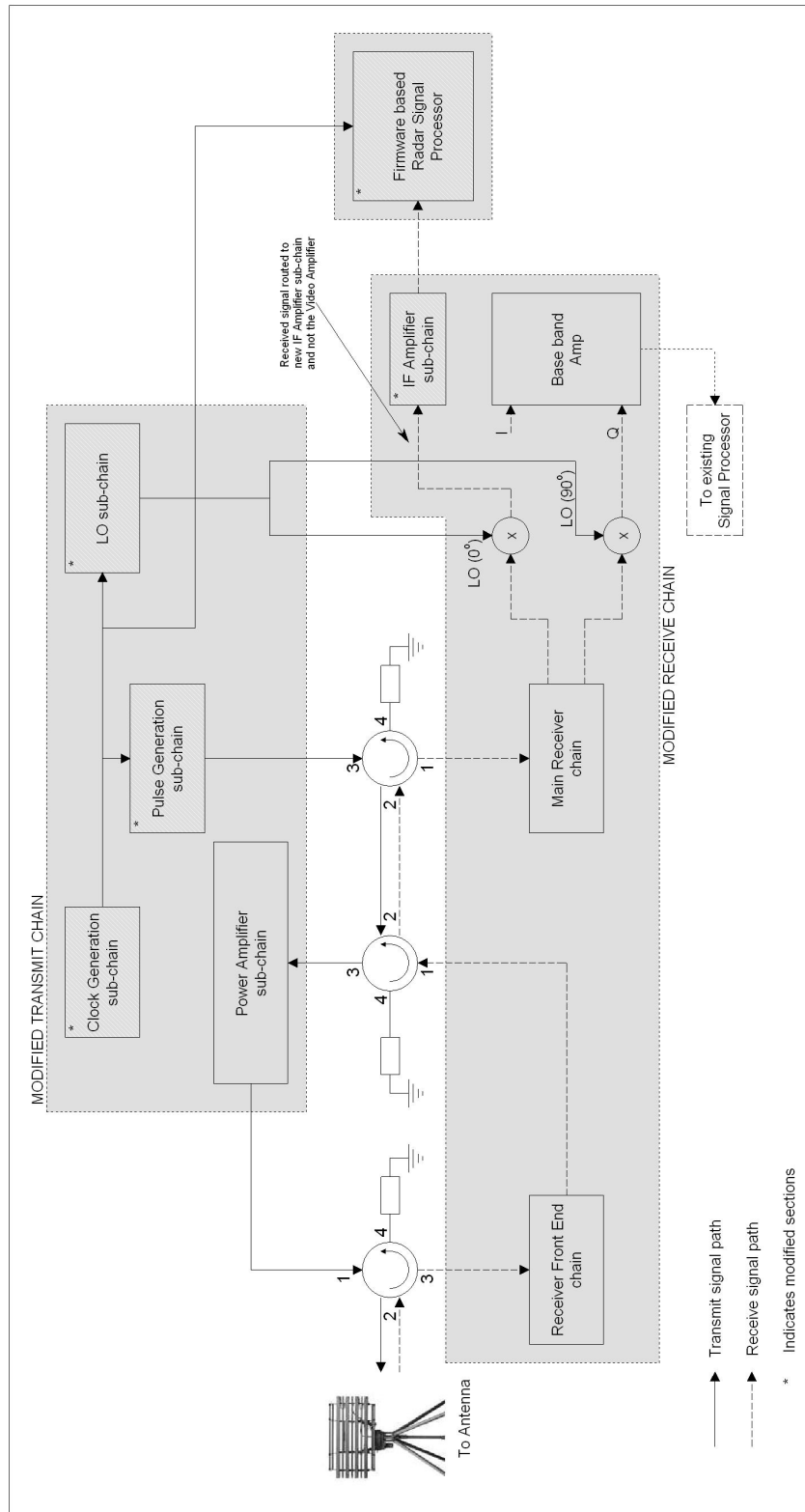


Figure 3.1: Functional block diagram of the upgraded Transceiver Module

The modifications were implemented in an metallic case allowing for the easy connection of the relevant modules. Figure (3.2) shows a photograph this case.

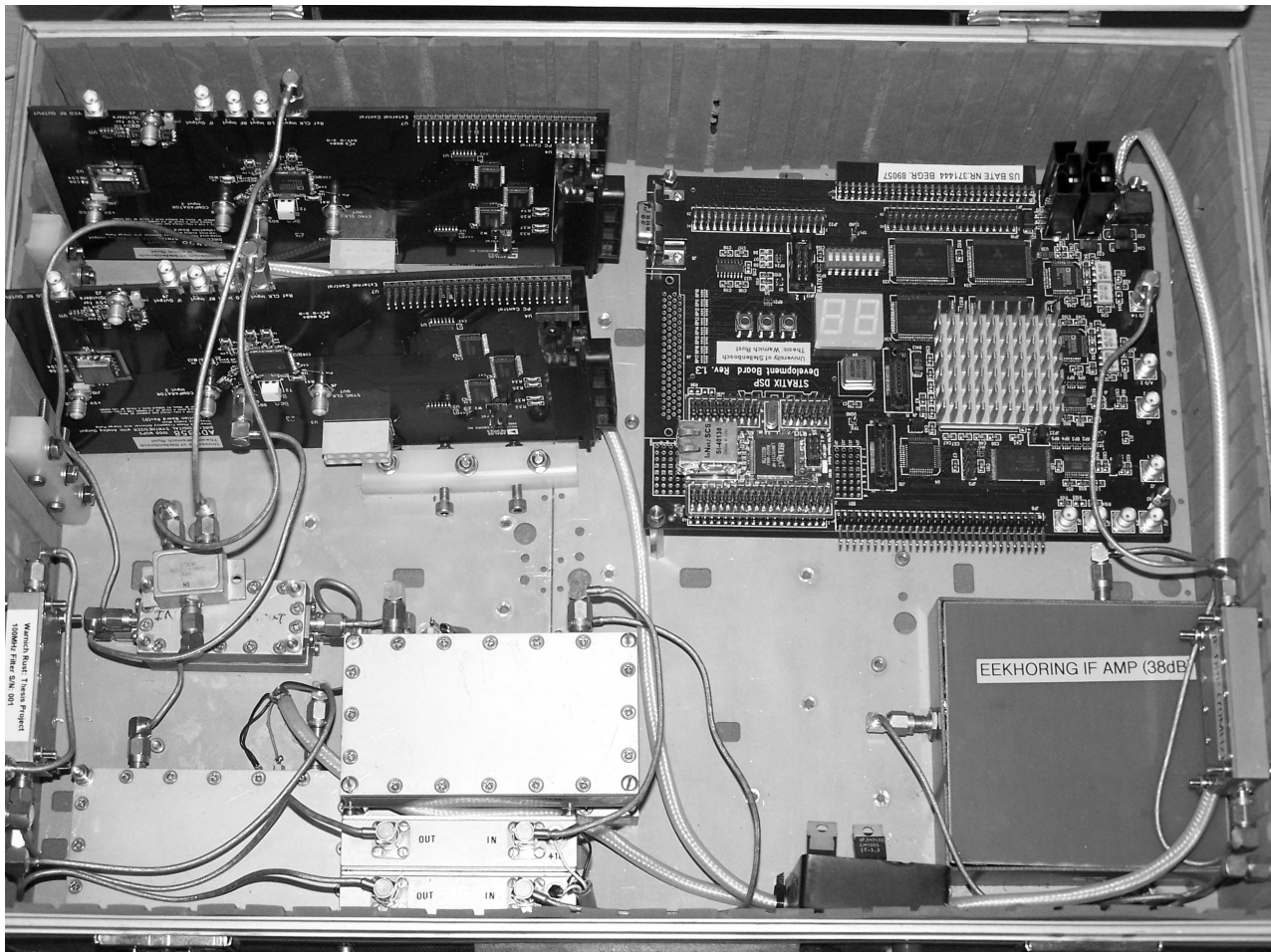


Figure 3.2: Photograph of the transceiver modification case

3.2.1 RF Subsystem

The modifications made to the RF subsystem are:

3.2.1.1 Transmitter chain

Modifications to the transmitter module were done in the clock generation, exciter and LO chains. Refer to figure (3.1) which shows a simplified block diagram of the modified transmit chain.

The existing exciter chain was modified to produce two (2) chains: The one the clock generation sub-chain and the other the pulse modulation sub-chain.

The existing 9-time frequency multiplier modules were replaced due to these modules having a high failure rate. This replacement also required an amplifier module at the end of the exciter chain to produce an equivalent

chain output power level to the unmodified chain. Two (2) filters were also introduced to suppress harmonic and spurious signals.

The main modification however was the introduction of a Direct Digital Synthesizer (DDS) module in the existing exciter chain. This DDS module was used to phase modulate the transmission pulse with a pulse compression code to improve range resolution in the RSP.

The modifications to the various chains in the transmitter module with results of the newly introduced components are given in the following sections.

3.2.1.1.1 Clock Generation sub-chain

The Clock Generation chain is used to distribute various clock signals, generated from a single frequency source, to the RSP board, LO and Transmit sub-chains.

The output of the existing variable frequency source was fixed to 98.8MHz and band pass filtered. The resulting signal was split to provide signal paths for the RSP Clock and the rest of the Transmit chain. The one splitter output to the RSP Clock is amplified to 15-16 dBm required by the RSP board while the other is output is frequency multiplied nine (9) times. This is implemented by means of a new 9-times frequency multiplier module replacing the existing one.

The output of the 9-times frequency multiplier module is split again and routed to both the LO and pulse modulation chains.

Figure (3.3) shows the block diagram of the Clock Generation chain with applicable signal levels.

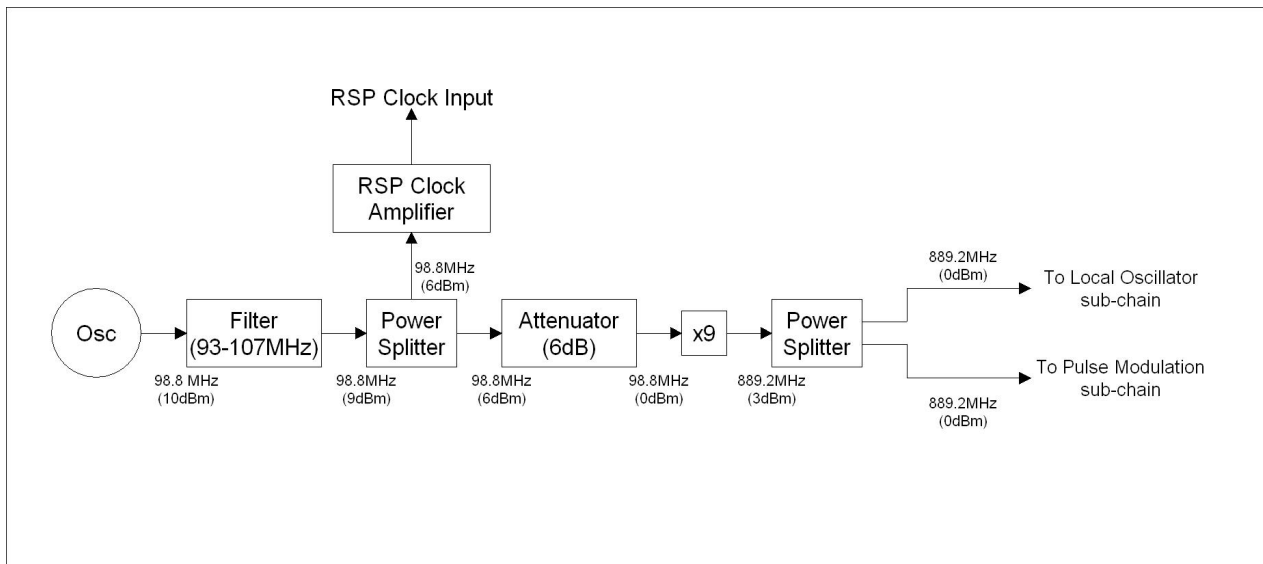


Figure 3.3: Functional block diagram of the Clock Generation sub-chain

The three modules introduced into this chain and the results achieved are:

3.2.1.1.1.1 Clock Filter

Refer to Appendix B.3 for a full functional description of this module.

This module is used to band pass filter the output of the system oscillator. Figure (3.4) shows the first, second and third order harmonic signal suppression. The centre frequency is 98MHz and the filter bandwidth is $\pm 14,5$ MHz. The signal loss at the centre frequency is 1,8dB.

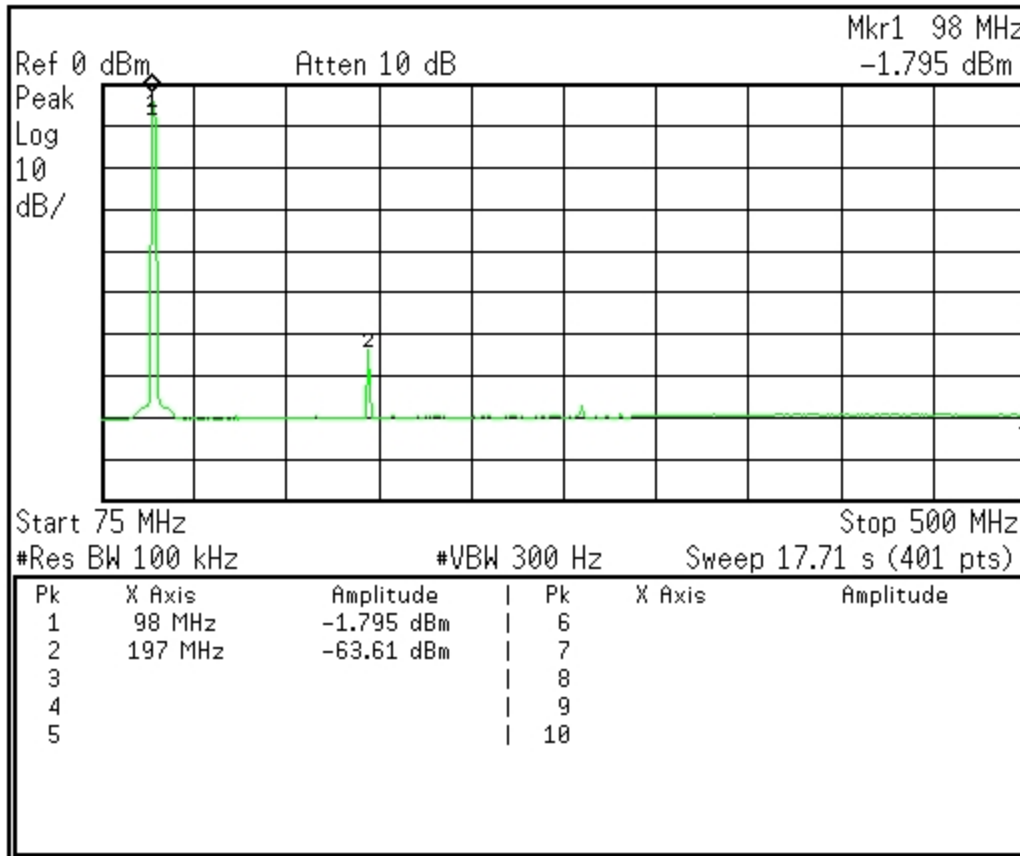


Figure 3.4: Harmonic signal power levels of the system oscillator filter module

3.2.1.1.1.2 RSP Clock Amplifier

Refer to Appendix B.5 for a full functional description of this module.

With an input frequency of 98,8MHz at a signal power level of 5,7dBm, the output power level was measured to be 15,7dBm (the minimum required level is 13dBm). This results in the amplifier providing a gain of 10dB. Figure (3.5) shows the output of the amplifier module. Spurious signals are at a level of 65,3dBc away from the fundamental.

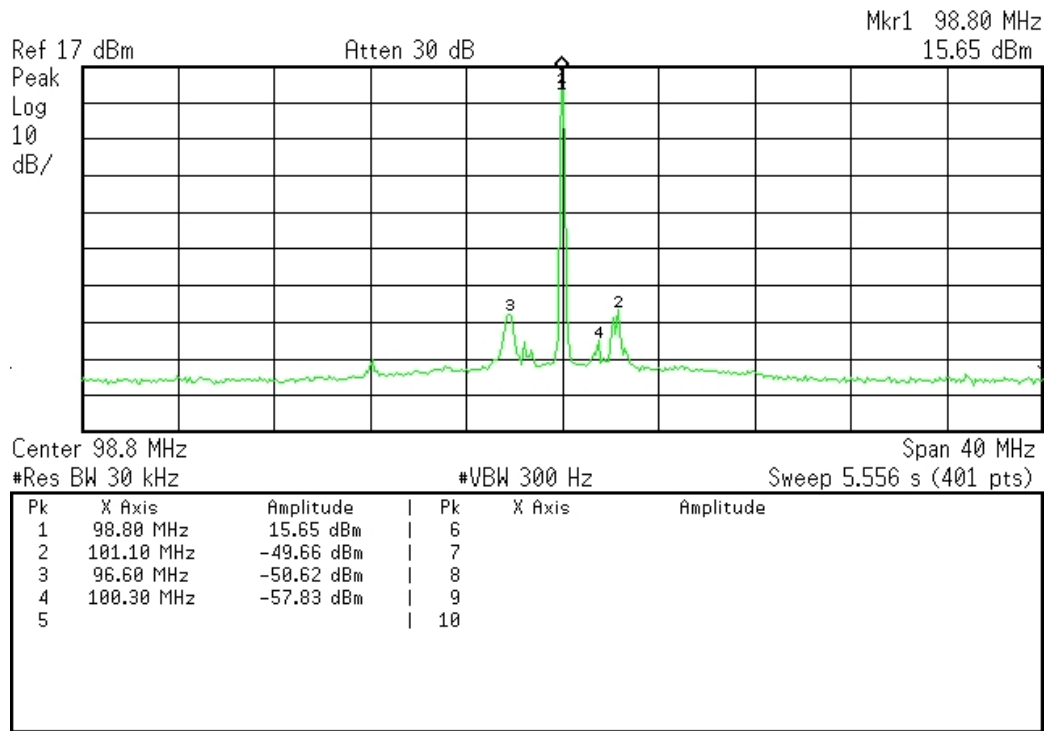


Figure 3.5: Measured output signal of the RSP clock amplifier module

3.2.1.1.1.3 9-Time Frequency Multiplier

Refer to Appendix B.1 for a full functional description of this module.

This module is used to multiply the system clock of 98,8MHz to 889,2MHz, the reference frequency used by the two (2) DDS modules. Given an input frequency of 98,8MHz at a power level of 0dBm the output was found to be 889,2MHz with a power level of 6,7dBm. Figure¹ (3.6) shows the bandwidth of the multiplier at the -3dB points to be 11,5MHz.

¹Note that these graphs were taken directly from the spectrum analyzer instrument. Frequency is on the x-axis with far-left the start and far-right the stop frequencies. The centre frequency is in the middle of the image. The logarithmic y-axis shows the signal power level in dBm. Sometimes a peak table is available showing the marker values.

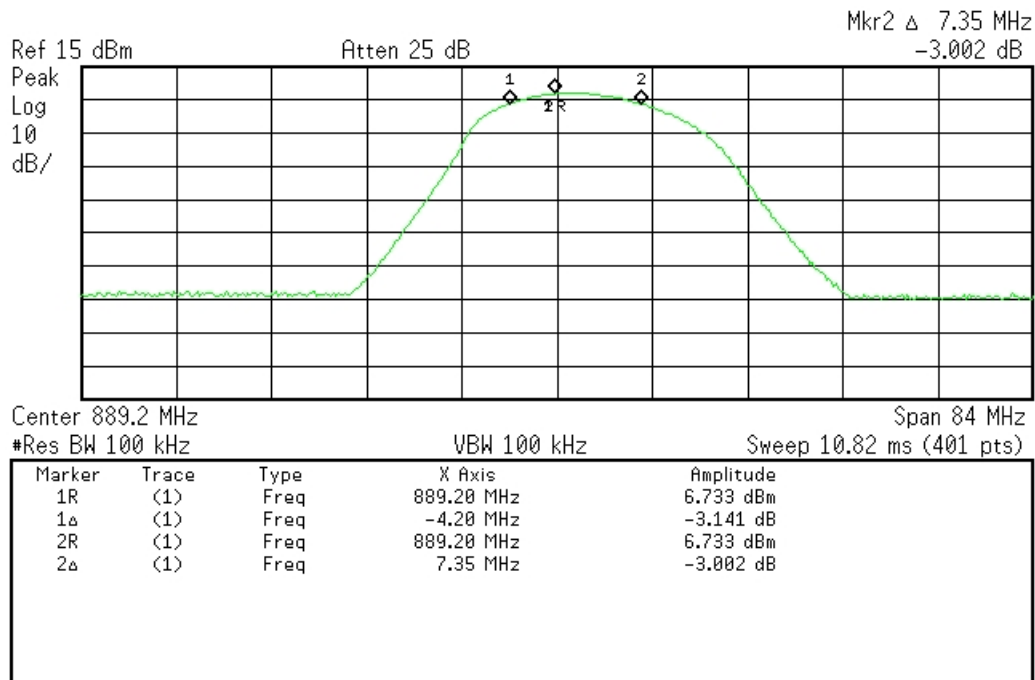


Figure 3.6: Bandwidth of the system clock frequency multiplier module

3.2.1.1.2 Pulse Generation sub-chain

Derived from the existing exciter chain, the pulse generation sub-chain is used to generate the coded transmission pulse which is compressed in the RSP on reception. A modulation frequency of $2,53125\text{MHz}^2$ is implemented and this bandwidth was designed for throughout the RF subsystem. The transmission pulse is phase modulated by means of a Direct Digital Synthesizer (DDS) module.

The Transmit DDS module, using the 900MHz output from the clock generation sub-chain, outputs a modulated transmit pulse at $\frac{1}{9}$ th the input frequency. Thus effectively returning to the frequency output by the frequency source in the clock generation sub-chain. The pulse modulation code is controlled by the RSP and is discussed later in this chapter (refer to section (3.2.2.4)).

This results in the need for a times-9 frequency multiplier unit to produce the 900MHz signal needed in the power amplification sub-chain. The output of the Transmit DDS module is bandpass filtered to suppress harmonic and spurious signals before the times-9 multiplication. The signal level needs to be amplified to 25dBm before being routed back into the rest of the transmit chain.

The existing times-9 multiplier modules included an amplifier with a 25dB gain. This amplification in the modified system was achieved by using the existing problematic modules, but with the times-9 multiplier section disabled thus directly routing the new times-9 module to the amplifier section of the old module. The output frequency is 889,2MHz at a signal power level of 25dBm. The rest of the existing exciter chain is used in which the transmit pulse is multiplied and amplified before being routed to the power amplification chain by means of two (2) circulators as per Chapter 2.

The output of the pulse generation chain is a modulated transmit pulse of $13,3\mu\text{s}$ repeated every $160\mu\text{s}$ (the PRI) which is generated by means of the PIN switch.

Figure (3.7) shows the block diagram of the pulse generation sub-chain with relevant frequencies and signal power levels.

²This frequency selection is discussed in the RSP section of this chapter.

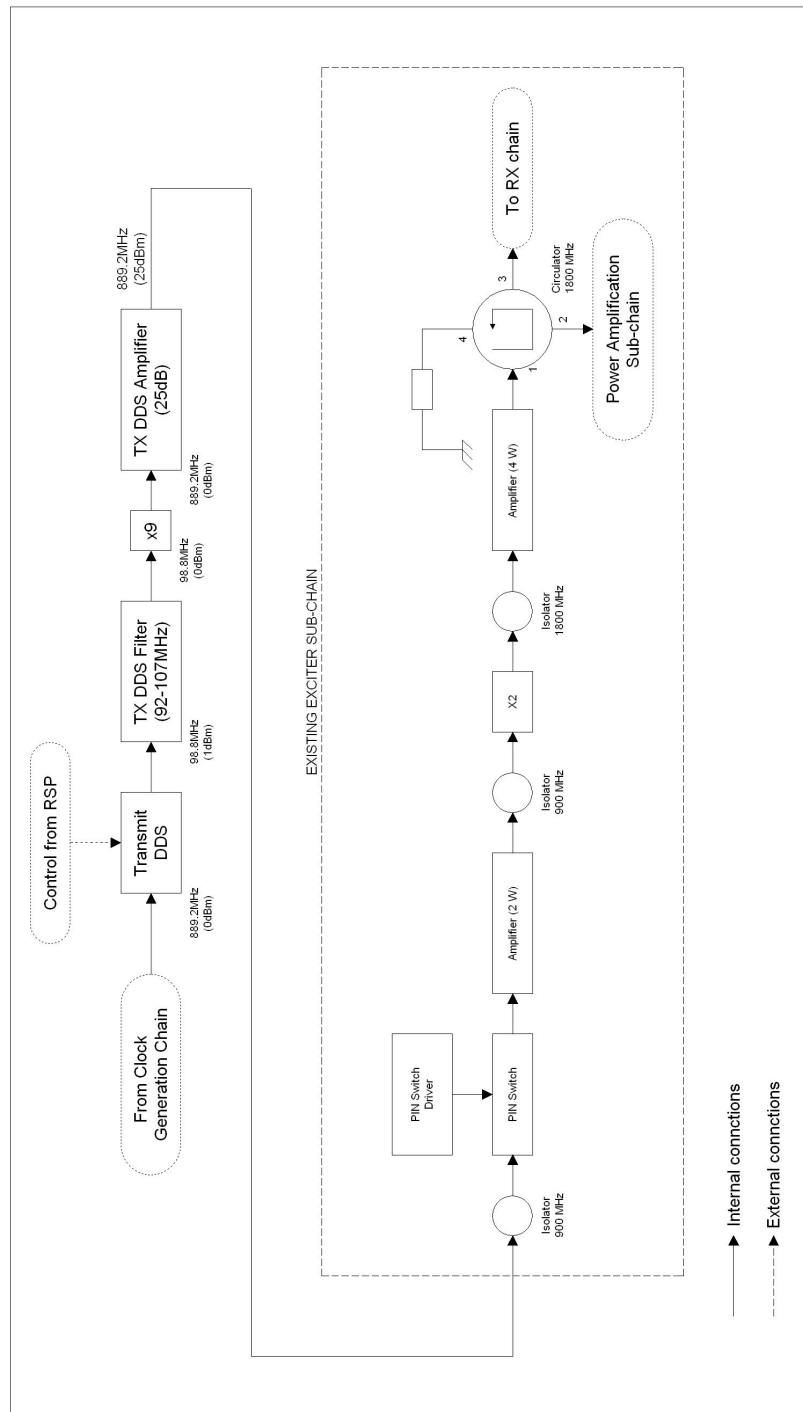


Figure 3.7: Functional block diagram of the pulse generation sub-chain

3.2.1.1.2.1 Pulse Generation DDS Module

Refer to Appendix B.2 for a full functional description of this module.

The DDS module receives a reference clock signal of 889,2MHz and generates the modulated output frequency of 98,8MHz. The actual pulse modulation code is controlled by the RSP and implemented by selecting two (2) preset frequency and phase offset profiles of the DDS module. The setup values for the DDS module is calculated below.

Using equation (B.1) the FREQUENCY TUNING WORD (FTW) can be calculated as:

$$\begin{aligned} 98,8 \times 10^6 &= \frac{(FTW)(889,2 \times 10^6)}{2^{32}} \\ FTW &= \frac{(98,8 \times 10^6)(2^{32})}{889,2 \times 10^6} \\ &= 477218588.44444400 \end{aligned}$$

The FREQUENCY TUNING WORD (FTW) is an integer value and rounded to $FTW = 477218588$. Using the rounded FTW the actual output frequency is 98799999,91MHz. Two profiles (3 and 4) were loaded with the FTW as above. The PHASE OFFSET WORD (POW) for profile 3 was loaded as $POW = 0$ while the POW for profile 4 was calculated for a 90° phase offset to be:

$$\begin{aligned} POW_{profile\ 4} &= \frac{90}{360} \times 2^{14} \\ &= 4096 \end{aligned}$$

Care must be taken during frequency planning since the DDS module generates a $-63dBc$ spurious signal at $\frac{REFCLK}{8}$.

For the transmit DDS module this spurious signal is located at:

$$\begin{aligned} f_{spur} &= \frac{889,2}{8} \\ &= 111,15 [MHz] \end{aligned}$$

This is out of the operational range of the transmitter chain.

Figure (3.8) shows the output frequency and signal power level of the transmit DDS module when profile 3 was selected.

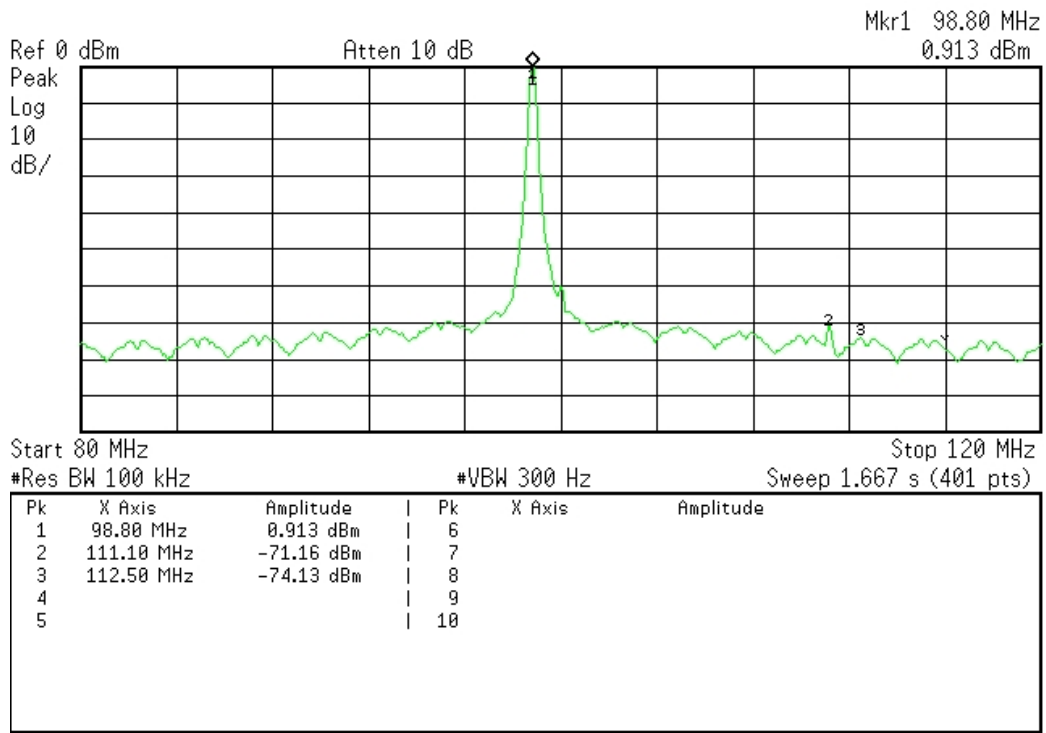


Figure 3.8: Output frequency and signal power level of the transmit DDS module

3.2.1.1.2.2 DDS Output Filter

Refer to Appendix B.3 for a full functional description of this module.

The DDS filter is used to filter the harmonics signals at the output of the DDS module. Figure (3.9) shows the bandwidth of the DDS filter module while figure (3.10) shows the harmonic suppression of the filter. The centre frequency is 99,1MHz and the filter bandwidth is $\pm 15,5$ MHz. The signal loss at the centre frequency is 1,59dB.

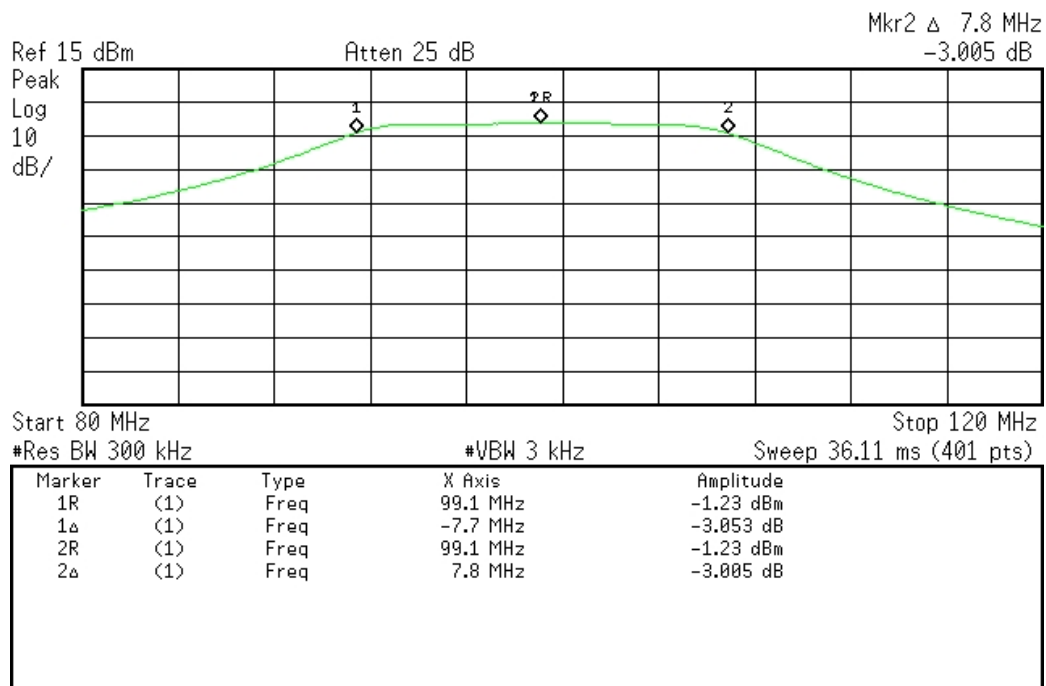


Figure 3.9: Bandwidth of the transmit DDS filter module

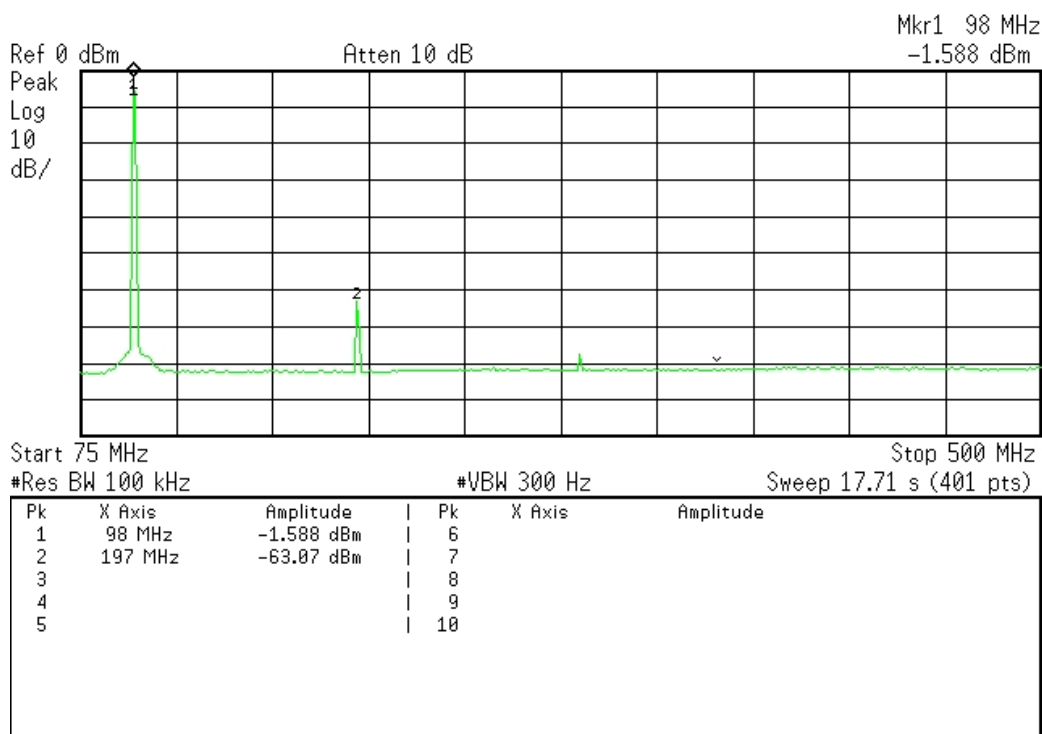


Figure 3.10: Harmonic signal power levels of the transmit DDS filter module

3.2.1.1.2.3 Times-9 Frequency Multiplier

Refer to Appendix B.1 for a full functional description of this module.

The times-9 multiplier module is used to multiply the output of the DDS filter back to 889,2MHz allowing the output to be routed to the rest of the existing exciter chain. Given an input frequency of 98,8MHz at a power level of 0dBm the output was found to be 889,2MHz with a output power level of 5,7dBm. Figure (3.11) shows the bandwidth of the multiplier at the -3dB points to be ± 37 MHz.

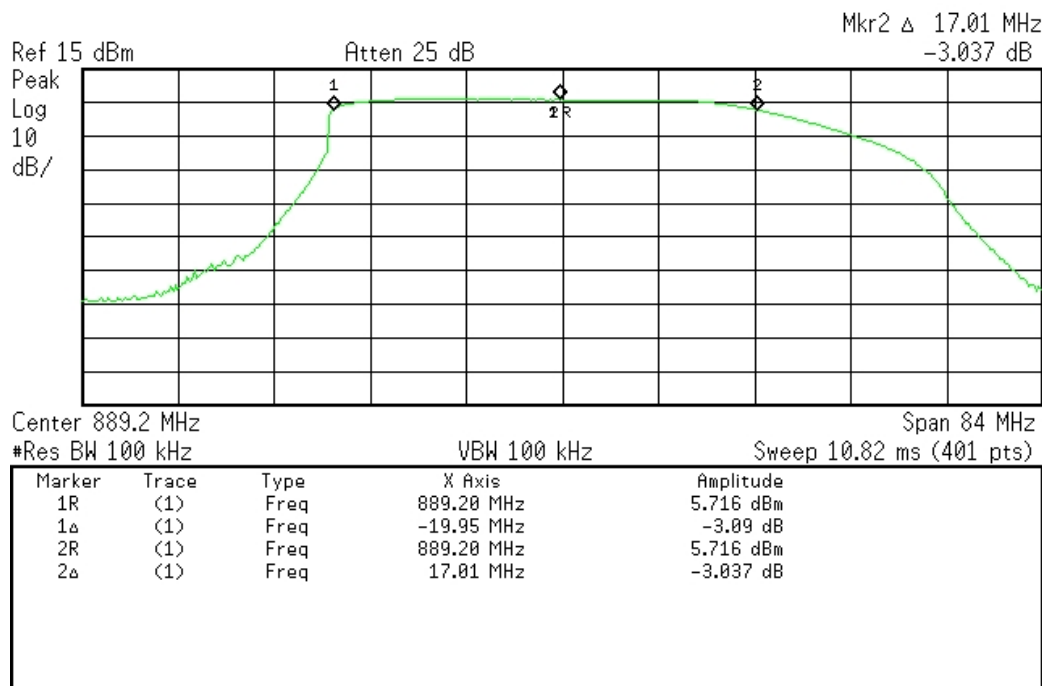


Figure 3.11: Bandwidth of the transmit DDS frequency multiplier module

3.2.1.1.2.4 DDS Amplifier

Refer to Appendix B.4 for a full functional description of this module.

The DDS amplifier is used to amplify the output of the frequency multiplier module to the required power level of ± 25 dBm. With a centre frequency of 889,2MHz and a signal power level of 0dBm the gain was found to be 23,49dB. Figure (3.12) shows the bandwidth of the amplifier at the -3dB points to be ± 82 MHz.

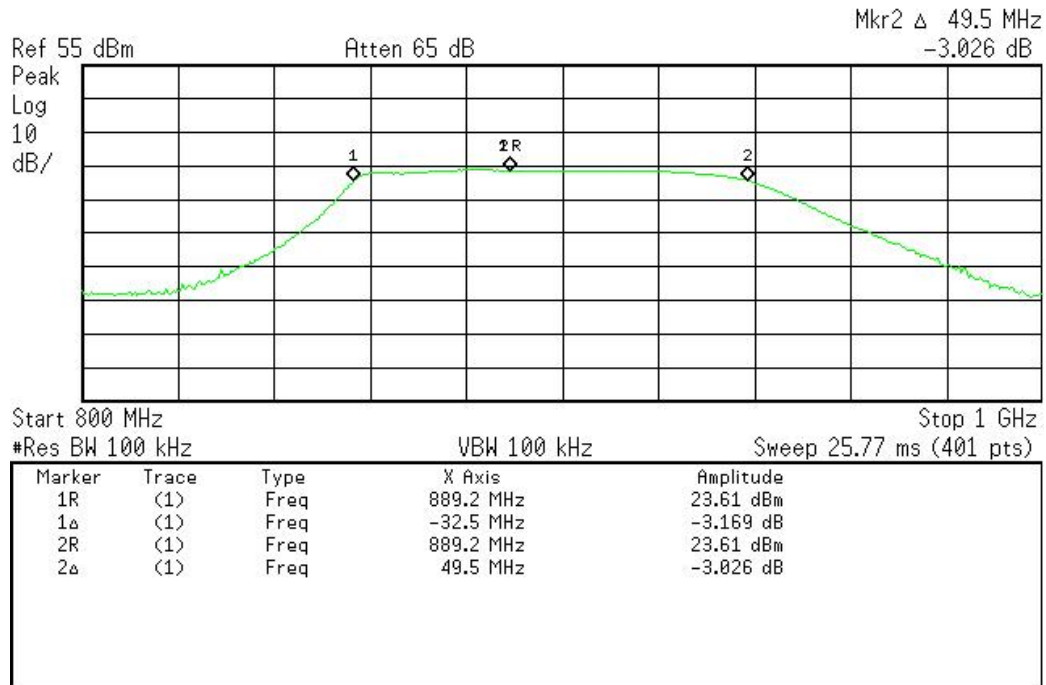


Figure 3.12: Bandwidth of the transmit DDS amplifier module

3.2.1.1.3 Local Oscillator (LO) sub-chain

In the existing ERS transmit chain, the same frequency used for the transmit pulse is used in the receive chain to mix down the returning signal to base band. The Analogue to Digital (ADC) converter used in the upgraded RSP performs IF sampling.

It was thus needed to modify the existing LO chain to generate an IF signal at the output of the receive chain mixer. This requires the LO chain to produce a frequency which is offset to the transmission frequency. The same DDS design setup as in the pulse generation sub-chain is used, but the DDS module outputs an offset frequency to the actual pulse generation sub-chain as per section (3.2.1.1.2).

The LO sub-chain receives a reference clock of 889,2MHz from the clock generation sub-chain as input to the DDS module. The offset frequency output of the DDS module is band pass filtered and routed to a times-9 module. This times-9 module is the same design as the other times-9 modules, but the internal filters have been tuned with respect to the offset frequency. Finally the signal is amplified to the correct power level before being routed back into the existing LO sub-chain where it is multiplied and amplified again to be routed to the receiver mixer.

The offset frequency is calculated to be an $\frac{1}{18}$ th of the IF frequency of 70,875MHz. The $\frac{1}{18}$ th factor is due to multiplication factor of 18 in the transmit chain.

$$\begin{aligned}
 f_{LO\ DDS} &= f_{PM\ DDS} + \frac{f_{IF}}{18} \\
 &= 98,9 + \frac{70,875}{18} \\
 &= 102,7375\text{MHz}
 \end{aligned}$$

Figure (3.13) shows the block diagram of the modified LO sub-chain. The applicable frequencies and power levels are indicated for the modified section.

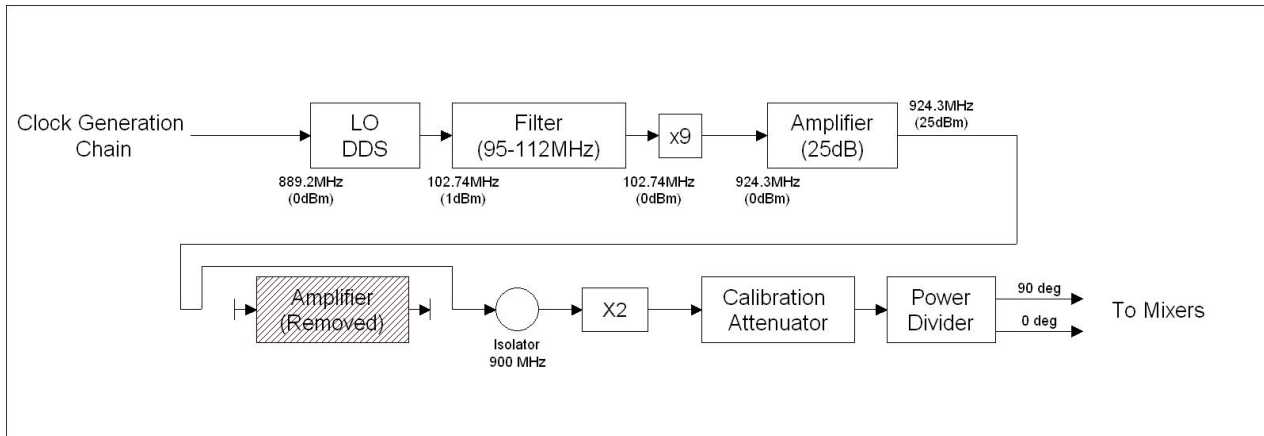


Figure 3.13: Functional block diagram of the LO sub-chain

3.2.1.1.3.1 LO DDS Module

Using equation (B.1) the FTW can be calculated as:

$$\begin{aligned}
 f_{LO\ DDS} &= \frac{(FTW)(889,2 \times 10^6)}{2^{32}} \\
 FTW &= \frac{(102,7375 \times 10^6)(2^{32})}{889,2 \times 10^6} \\
 &= 496237294,9
 \end{aligned}$$

When rounded, the $FTW = 496237295$ and this was implemented. Using the rounded FTW the actual output frequency is 102,7375MHz. Only one profile was needed and the POW was set to zero.

The spurious signal for the LO DDS module is located at the same frequency as for the transmit DDS module.

$$\begin{aligned}
 f_{spur} &= \frac{889,2}{8} \\
 &= 111,15\text{ [MHz]}
 \end{aligned}$$

This is also out of the operational range of the LO chain.

Figure (3.14) shows the output frequency and signal power level of the LO DDS module.

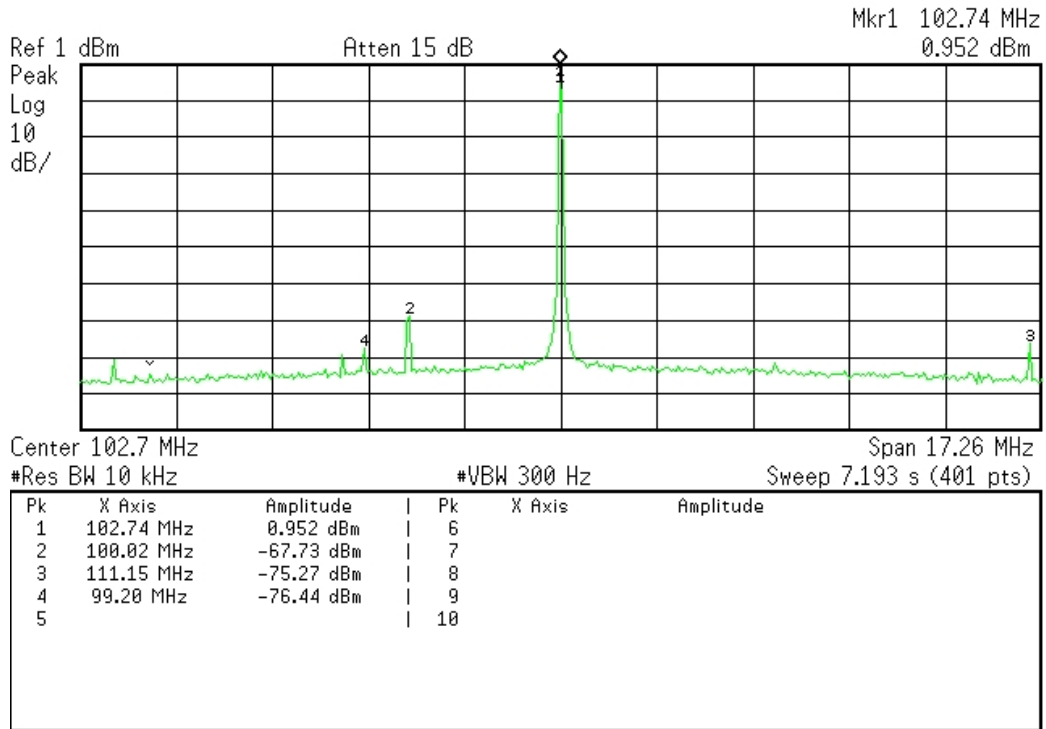


Figure 3.14: Output signal power level of the LO DDS module

3.2.1.1.3.2 DDS Output Filter

Refer to Appendix B.3 for a full functional description of this module.

This module has the same functionality as the previous filter, but is used at the output of the LO DDS module. Figure (3.15) shows the harmonic suppression of the LO DDS filter module. The centre frequency is 102,3MHz and the filter bandwidth is $\pm 15,8$ MHz. The signal loss at the centre frequency is 1,9dB.

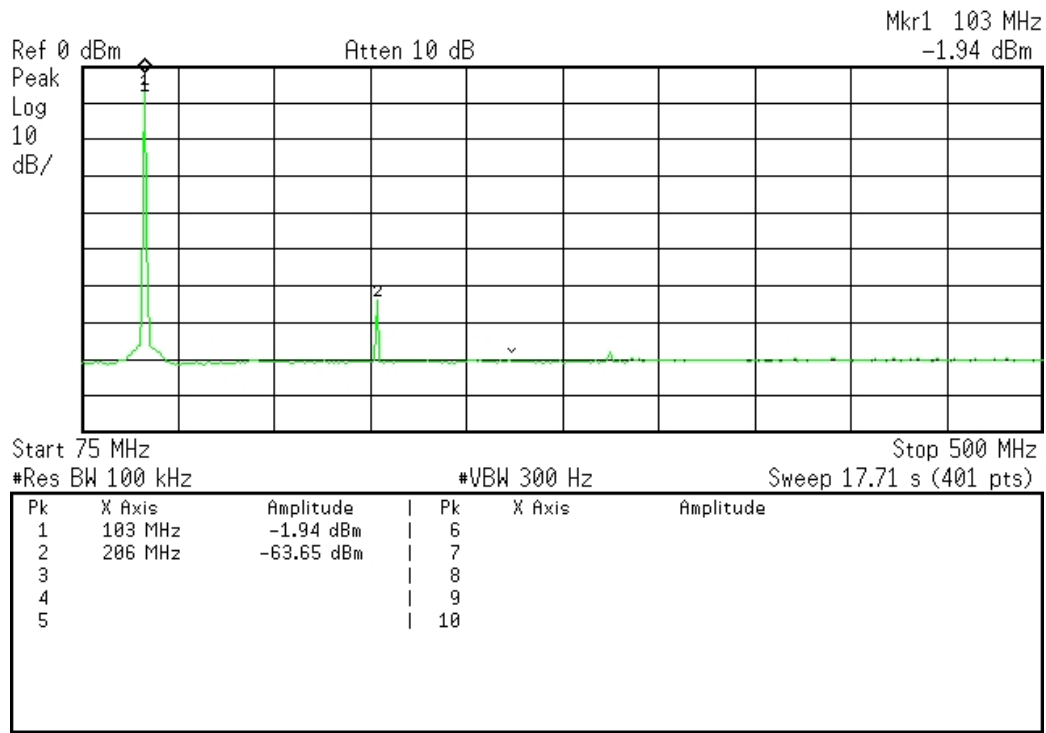


Figure 3.15: Harmonic signal power levels of the LO DDS filter module

3.2.1.1.3.3 9-Time Frequency Multiplier

Refer to Appendix B.1 for a full functional description of this module.

This module is used to multiply the output of LO DDS filter up from 102,74MHz to 924,64MHz. The output is then routed back into the existing LO sub-chain. Figure (3.16) shows the bandwidth at the -3dB points of the LO DDS frequency multiplier module to be ± 27 MHz. An input signal power level of 0dBm produced a 3dBm signal at the output.

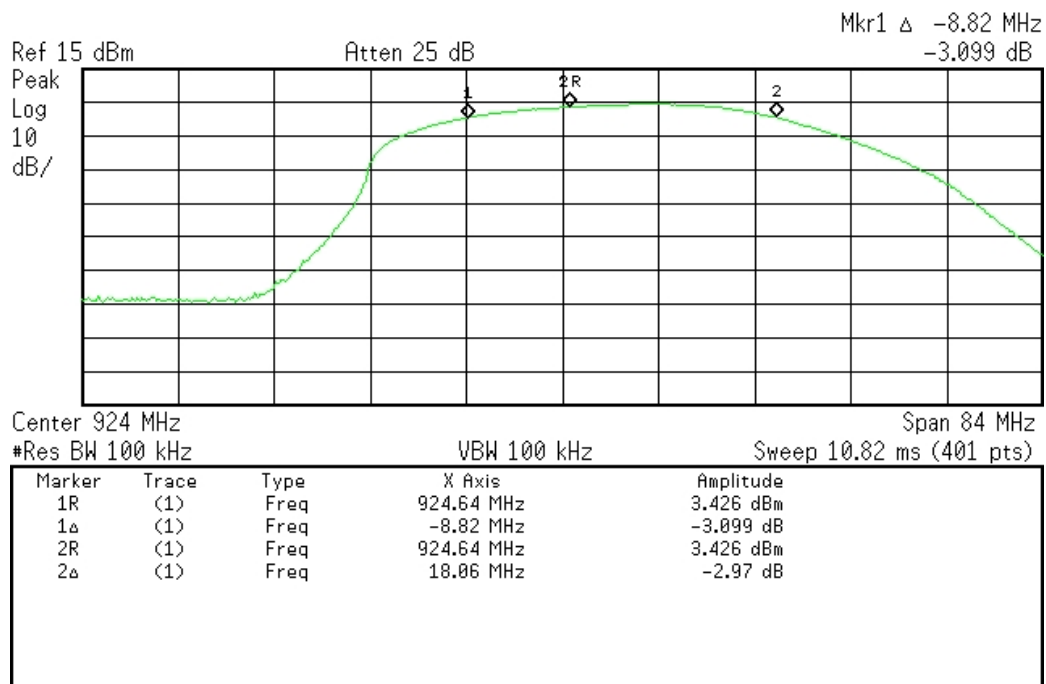


Figure 3.16: Bandwidth of the LO DDS frequency multiplier module

3.2.1.1.3.4 DDS Amplifier

Refer to Appendix B.4 for a full functional description of this module.

The LO DDS amplifier is used to amplify the output of the LO DDS frequency multiplier module to the required power level of ± 25 dBm. With a centre frequency of 924,64MHz and a signal power level of 0dBm the gain was found to be ± 20 dB. Figure (3.17) shows the output power level of the module (the bandwidth is not needed since the LO frequency does not change).

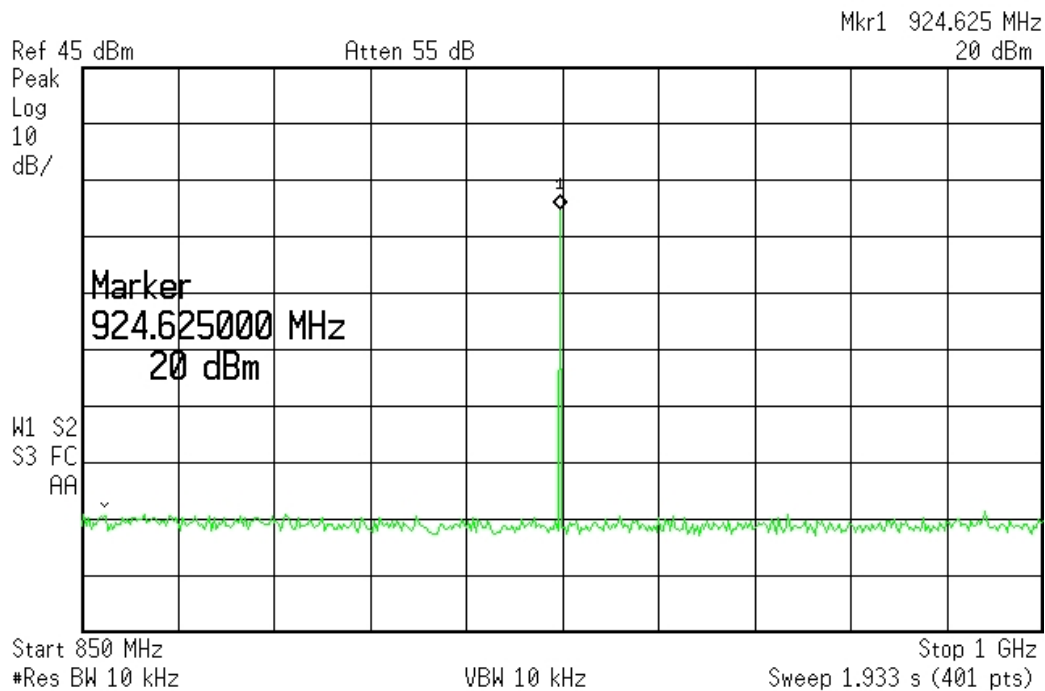


Figure 3.17: Output signal power level of the LO DDS amplifier module

(Note that the full range of the testing instrument was not used when this measurement was generated.)

3.2.1.2 Receive Chain

This receive chain was only modified at the end of the main receiver sub-chain to accommodate the IF sampling requirement of the ADC module in the RSP.

Refer to the receive chain section of figure (3.1) for a block diagram of the receive chain.

3.2.1.2.1 Main Receiver sub-chain

The I-channel feed to the Video (or Base Band) amplifier module in the existing system was routed via an IF amplifier module to amplify the mixed down received signal for sampling by means of the ADC in the RSP. A band pass filter module was introduced at the output of the IF amplifier to band pass filter the signal mainly to minimize noise and to filter harmonics signals.

3.2.1.2.1.1 IF Amplifier

The IF amplifier is used to amplify the output of the receiver mixer to produce the 10dBm signal power level for the ADC to operate optimally. The measured gain at the IF frequency of 70.875MHz was 40dB.

3.2.1.2.1.2 IF Filter

To minimize the noise at the input of the ADC in RSP, the output of the IF amplifier module passes through an IF filter module. Figure (3.18) shows the bandwidth of the IF filter module. The centre frequency is 71,2MHz and the filter bandwidth is ± 10 MHz. The signal loss at the centre frequency is 0,9dB.

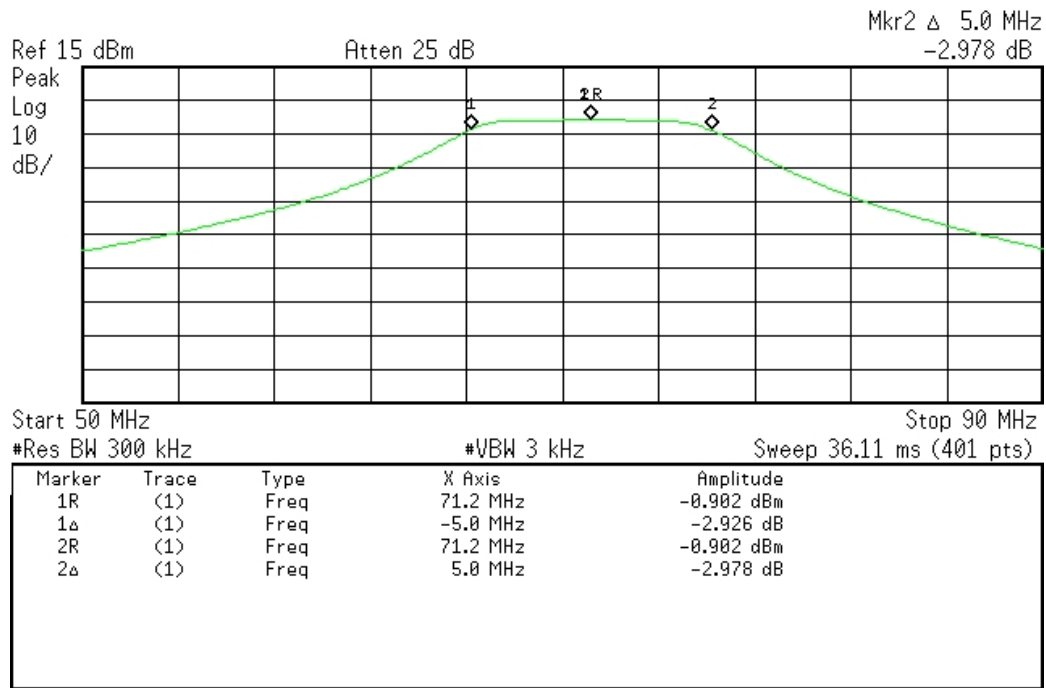


Figure 3.18: Bandwidth of the IF filter module

3.2.1.3 RF Performance measurements

Various RF measurements were planned for the modified transceiver. These included the MDS and an end-to-end measurement of the system. These measurements however were never done due to the system not being in a testable state.

3.2.2 Signal Processor Modifications

3.2.2.1 Overview

The existing ERS has a range resolution of 2km which is determined by the ten (10) PRF filter circuits in the signal processor. The primary aim of this project was to improve on this limiting factor. To improve the range resolution of the existing radar system, two (2) signal processing techniques were implemented: Digital Pulse Compression and an increased number of range bin target detection circuits.

The implementation of a pulse compression, as explained later in chapter (3.2.2.4), produces a compressed version of the transmit pulse. This improves the range resolution, while keeping the higher detection capabilities of a long pulse system.

Range resolution is improved by defining more range bins target detection circuits in the RSP than the existing ten (10) bins. Doppler processing is implemented on these increased number of range bins and the Doppler-shift generated by a moving target is computed which allows for the elimination of clutter.

The entire signal processor was replaced by a digital signal processor running on a dedicated development board. All signal processing is done on a single FPGA device. Refer to Appendix B.7 for a detailed description of the Altera Digital Signal Processing Evaluation Board used to implement the new RSP.

The RSP consists of the sections as shown in figure (3.19).

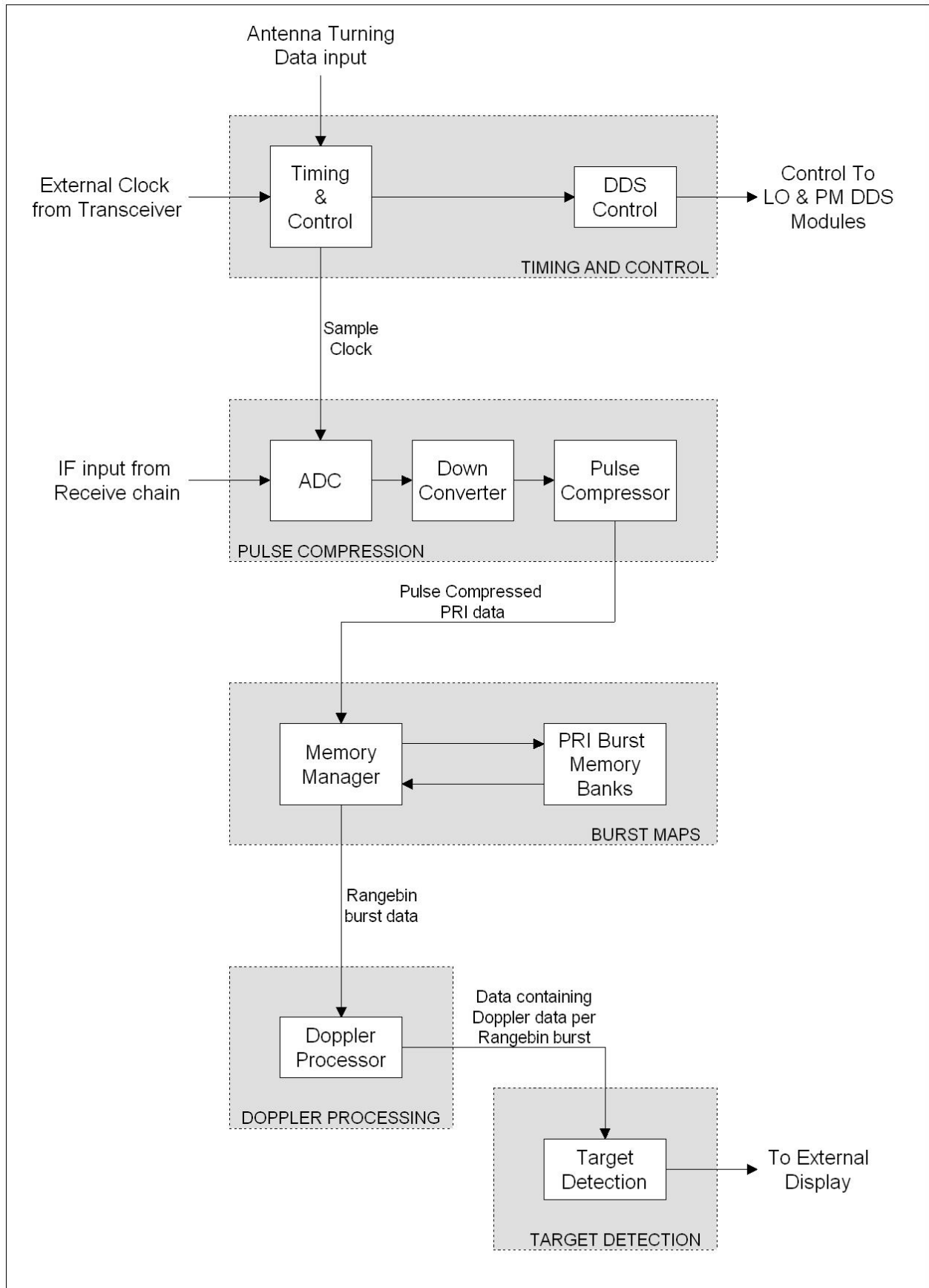


Figure 3.19: Functional block diagram of the new RSP

The RSP was fully designed but not fully implemented and tested due to time constraints. The sections implemented and tested will be discussed later. After implementation of almost the complete RSP (which included a 32-bit micro controller), it was found that only 25% of the available capacity was utilized. The amount of on-chip memory was the limiting factor, but this can be solved using external memory and creative techniques.

This section gives an overview design of the RSP and discusses the design and implementation of the RSP with regards to timing generation to ensure the radar system is synchronous, Pulse Compression design and implementation as well as implementation and results achieved by the down conversion chain and pulse compressor modules.

3.2.2.2 Quadrature Sampling of the IF signal

Since both the in-phase (I) and quadrature (Q) received signals are required for processing in this RSP, while using a single ADC, quadrature was implemented.

Quadrature sampling refers to the oversampling of a carrier signal (here the IF frequency) at four (4) times the frequency of the carrier and is shown in the following equation:

$$f_{sample} = 4 f_{IF}$$

This results in the signal being sampled every 90° . By combining the 0° and 180° samples into a data stream and the 90° and 270° into a second, two outputs are generated containing the I-channel data and the Q-channel data respectively. It is noted that the 180° and the 270° sampled results are negative and this should be compensated for.

The output of the I-channel data is of the form "I", "-I", "I", "-I", etc. while the Q-channel produces data of the form "Q", "-Q", "Q", "-Q", etc. Should the signal not be split, the output of the ADC results in giving "I", "Q", "-I", "-Q", "I", "Q", "-I", "-Q" sample outputs.

The bandwidth of the pulse modulation implemented in this project is 2,53125MHz resulting in a Nyquist frequency of 5,0625MHz for a single channel and thus a minimum frequency of 10,125MHz is required to be able to represent both the I-channel and Q-channel data once sampled.

The representation of the complete IF signal is not of interest and thus down sampling can be used where the sampling frequency does not comply to the Nyquist theorem. The equation[2] of

$$f_{sample} = \frac{4f_{IF}}{2n + 1} \quad (3.1)$$

(where $n = 1, 2, 3, 4, \dots$) can be used to select the an IF and sampling frequency pair. A maximum value for n is thus selected to minimize the sampling frequency while still complying to the Nyquist requirement for the bandwidth of the signal.

Originally a base RSP clock frequency of 81MHz was selected to allow for the use in the RSP was well as to replace the system clock of 9MHz in the existing ERS which generates the transmit trigger pulse. This resulted

in the selection of the sampling frequency of 40,5MHz. Using equation (3.1) and $n = 3$, the corresponding IF frequency was calculated to be:

$$\begin{aligned} f_{IF} &= \frac{7f_{sample}}{4} \\ &= 70,875MHz \end{aligned}$$

This is an easy frequency to implement in the transmit and receive chains by means of LO sub-chain frequency offset setup and IF amplification and filtering.

3.2.2.3 Frequency Planning and Synchronization

It is fundamental that the radar system must be synchronous and coherent. To achieved this a single frequency source must be used to generate all frequencies and timing signals and these signals must be synchronized to a particular event.

In the upgraded system the frequency source of 98,8MHz was used throughout the system to derive and generate all frequencies and timing signals. The synchronization of the system is achieved by triggering events on the Transmit Trigger Pulse supplied by the existing transceiver. This pulse indicates when the transceiver is generating the transmit pulse.

Figure (3.20) shows a block diagram for the timing distribution of the upgraded transceiver. Fundamentally the system is synchronized on the “Radar Synchronization Clock” of 10.125MHz which is derived from the frequency source by means of a Phase Lock Loop (PLL).

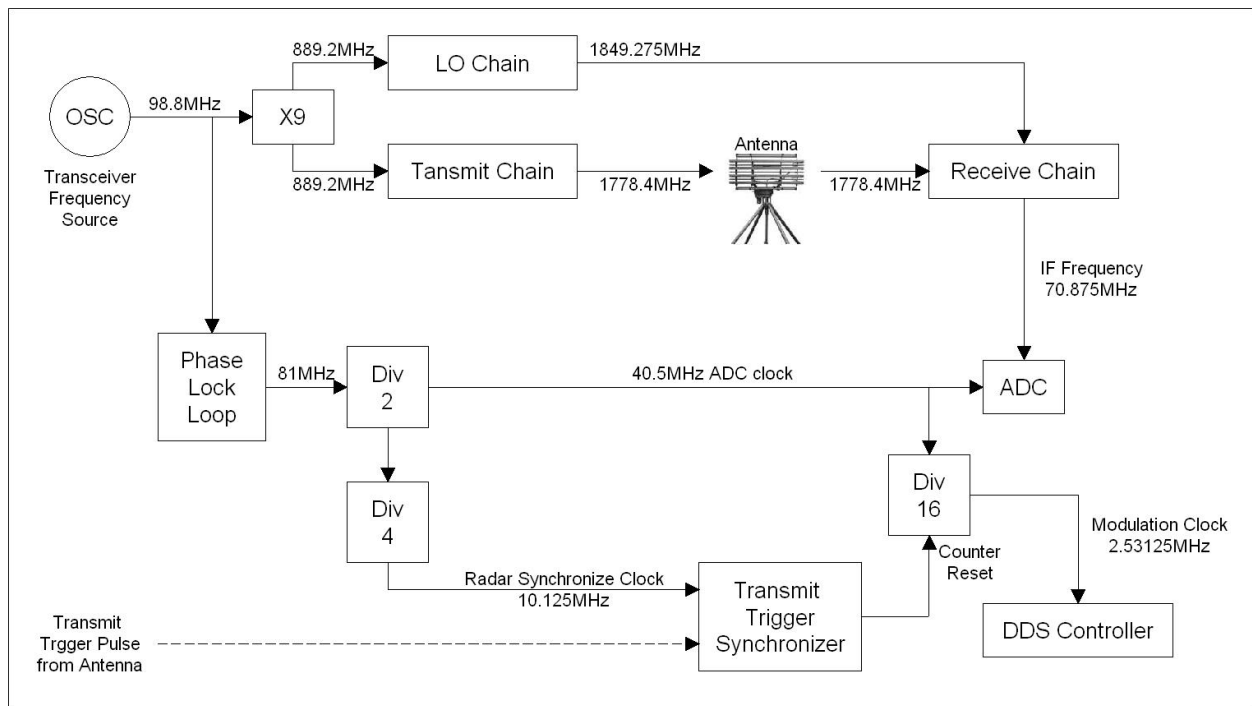


Figure 3.20: Block diagram of the timing distribution in the transceiver

In this setup the output of the LO and Transmit chains, generated by the two (2) DDS modules are regarded as two (2) free running clocks and are both driven by the frequency source. At the output of the Receiver chain, the frequency of the IF signal produced is 70,875MHz (the reason will become clear next).

In the RSP, the frequency source is used to drive the PLL which produces an output frequency of 81MHz. This output is divided by two (2) to derive the Sample Clock frequency of 40.5MHz. The Sample Clock is in turn divided by four (4) to give the RSP Synchronization Clock of 10.125MHz.

Having a closer look at the IF and Sampling frequencies, it is found that these two are factors of the RSP Synchronization Clock of 10.125MHz:

$$f_{IF} = 7 \times f_{RSP\ SYNC\ CLK}$$

and

$$f_{SAMPLE} = 4 \times f_{RSP\ SYNC\ CLK}$$

Referring to the IF Sampling section of this chapter (section (3.2.2.2)), it is required that the IF and sampling frequencies have a relationship of $\frac{7}{4}$. Thus, if this system is synchronized to the RSP Synchronization Clock of 10,125MHz, the system is regarded to be synchronous.

The system synchronization is done by means of the Transmit Trigger Synchronizer module. This module is used to sample the external transmit trigger event, and synchronize this to the RSP Synchronization Clock. The output of this module is used to reset the clock divider that generates the Modulation Clock of 2,53125MHz used to control the Pulse Generation DDS module which in turn generates the modulated transmission pulse in the Transmit chain.

3.2.2.4 Pulse Compression Implementation

3.2.2.4.1 Pulse Compression Overview

The ERS has a limited range resolution of 2km This equates to ten (10) range bins for the 20km operating range setting. When the operating range of 10km is selected, the range resolution improves to 1km This limited resolution is due to the fixed number of hardware range filters in the system. Each filter represents a range bin in which a target may be present.

One solution to increase the range resolution is to shorten the transmit pulse width, but this requires an increase in the peak power of the transmitter to maintain the radar's operational range. Increasing the peak transmit power on the other hand requires upgrades to the transmitter power amplifier sections which will increase the radar in size, design difficulty, and most importantly, in price.

Another technique is to modulate the transmit pulse into sub-pulses and spread it over time, which in turn will increase the average transmitter power and lower the peak power. This however increases the processing complexity. The technique is called Digital Pulse Compression[13, 8, 17] (or just pulse compression).

Pulse compression entails the transmission of a long modulated pulse and the processing of the returned signal to produce a relatively narrow pulse. Pulse compression is implemented in both the transceiver and the RSP

of a radar system, where the transceiver modulates (under the control of the RSP) the outgoing pulse and the RSP processes the returned echo.

The requirements for pulse compression in this project, the selection of a suitable pulse code and results achieved are discussed next.

3.2.2.4.2 Project Pulse Compression design

Pulse compression was done by convolution of the encoded received pulse with a copy of the encoded transmit pulse. When these two match up precisely, a large peak is produced. The range to the target is then determined by means of where this peak is located in a particular PRI. Since convolution in the time-domain equates to multiplication in the frequency-domain, FAST FOURIER TRANSFORM (FFT) methods were used to compress the pulse.

RRS provided a VHDL based Pulse Compressor module to be used for this project.

3.2.2.4.2.1 Modulation of the transmit pulse

The transmit pulse was 180° phase modulated by using the transmit DDS module to generate the phase offset by means of selecting the correct preset profiles at the modulation frequency of 2,53125MHz. Due to the two-times multiplier in the transmit chain of the transceiver the actual phase offset implemented by the DDS module was 90° which, when multiplied by two (2), will produce the required phase offset of 180° in the signal transmitted by the antenna.

Figure (3.21) shows the phase offset generated at the output of the DDS module as a 90° phase offset is added. This is implemented by selecting the correct profile of the DDS (refer to the detailed description of the DDS module in Appendix B.2).

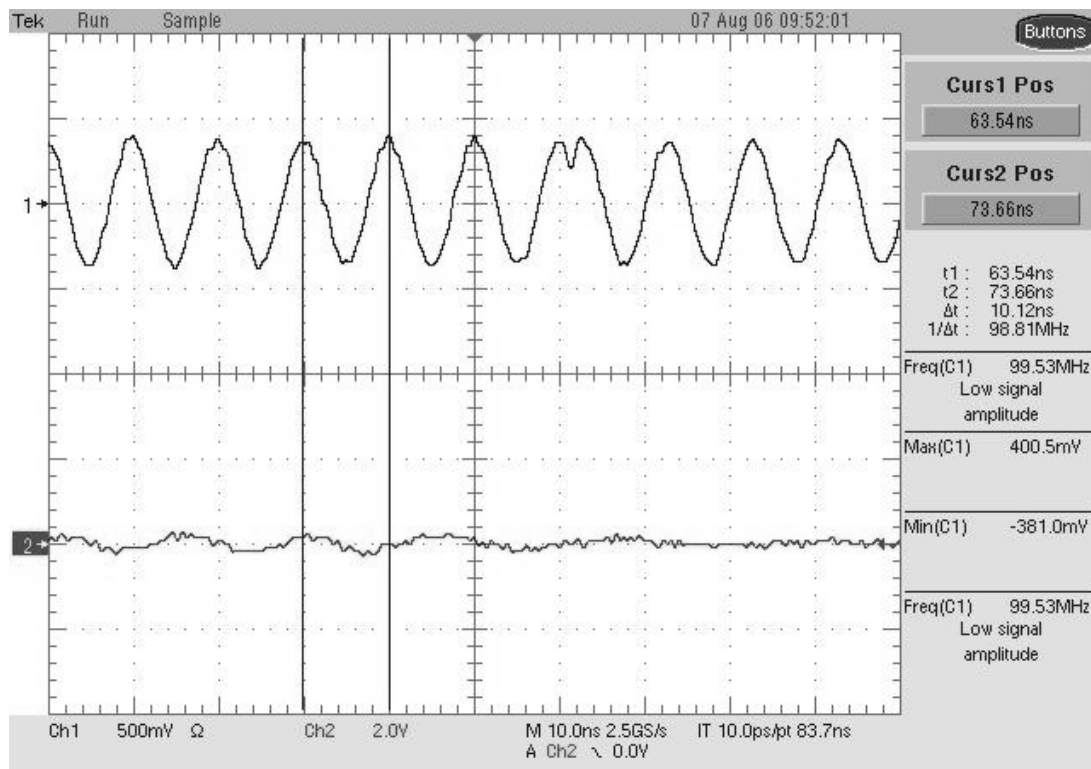


Figure 3.21: The output signal of the transmit DDS performing a 90° phase shift.

The transmit pulse was divided into sub-pulses. The number of sub-pulses were calculated as follows:

$$\begin{aligned}
 N &= \frac{\tau_{tx}}{\tau_{mod}} \\
 &= \tau_{tx} f_{mod}
 \end{aligned}$$

$$\begin{aligned}
 &= (13,3 \times 10^{-6})(2,53125 \times 10^6) \\
 &= 33,666 \\
 &\simeq 32
 \end{aligned}$$

where N is the number of sub-pulses, τ_{tx} the transmit pulse width and $\tau_{mod} = \frac{1}{f_{mod}}$ the modulation period and frequency.

Since only integer values can be implemented, the number of sub-pulses were rounded to $N = 32$ for easy implementation in digital systems.

By switching between these two DDS profiles, a phase-code is modulated on the transmit signal. The particular code pattern and design is discussed in the next section.

3.2.2.4.2.2 Phase-coded waveform design

Other than with FM waveform modulation, phase-coded waveforms sub-divide the transmit pulse into a number of sub-pulses[14, 9]. These sub-pulses carry a particular phase and is of equal length. Many types of phase-coding exist of which the most popular are bi-phase (or binary) and poly-phase coding.

Binary phase-coding implements a sequence of "0"s and "1"s resembling a 0° or 180° phase shift respectively in the transmit pulse as shown in figure (3.22) as an example.

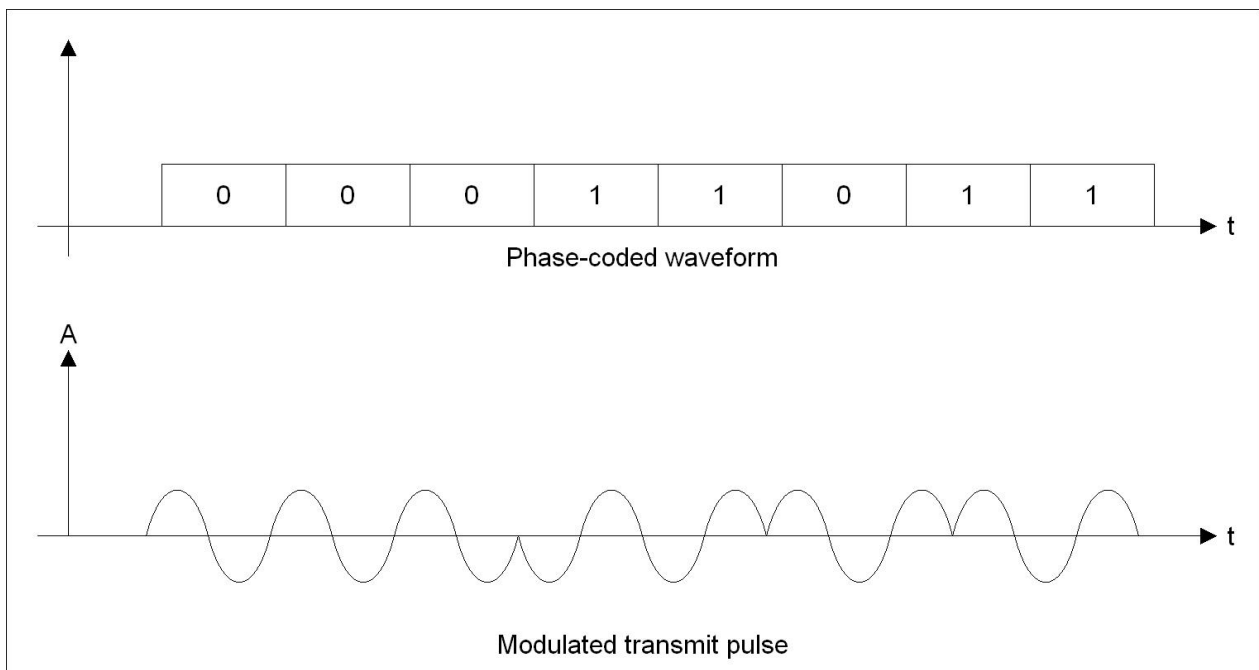


Figure 3.22: An example of a binary phase-coded signal.

Side-lobes are generated when the returning pulse is compressed, but with proper waveform design these side-lobes can be kept to a minimum. Side-lobes generated images of detected targets and produce false information. Side-lobe suppression must be implemented to prevent false detection but is not implemented here.

A special class of binary codes, called Barker codes, has the property that the peak of the auto-correlation function is equal to the number of sub-pulses while the maximum peak side-lobe is 1. Only thirteen (13) of these Barker codes exist resulting in a limited maximum compression ratio of thirteen (13).

An existing phase-code optimizing Matlab program was adapted³ and used to produce the 32-sub-pulse code needed. The software generates random 32-bit phase-codes, compresses each and compares it to a previous result in having the smallest side-lobes. Compression is done by taking the FFT of the random phase-code, multiplying it with itself (thus doing a time-domain convolution) and taking the IFFT to convert back to the time-domain. After numerous runs the best waveform found during simulation can be used. Figure (3.23) shows a compressed pulse and the generated side-lobes which are located at around -16dB. Note that the result is symmetrical around the origin.

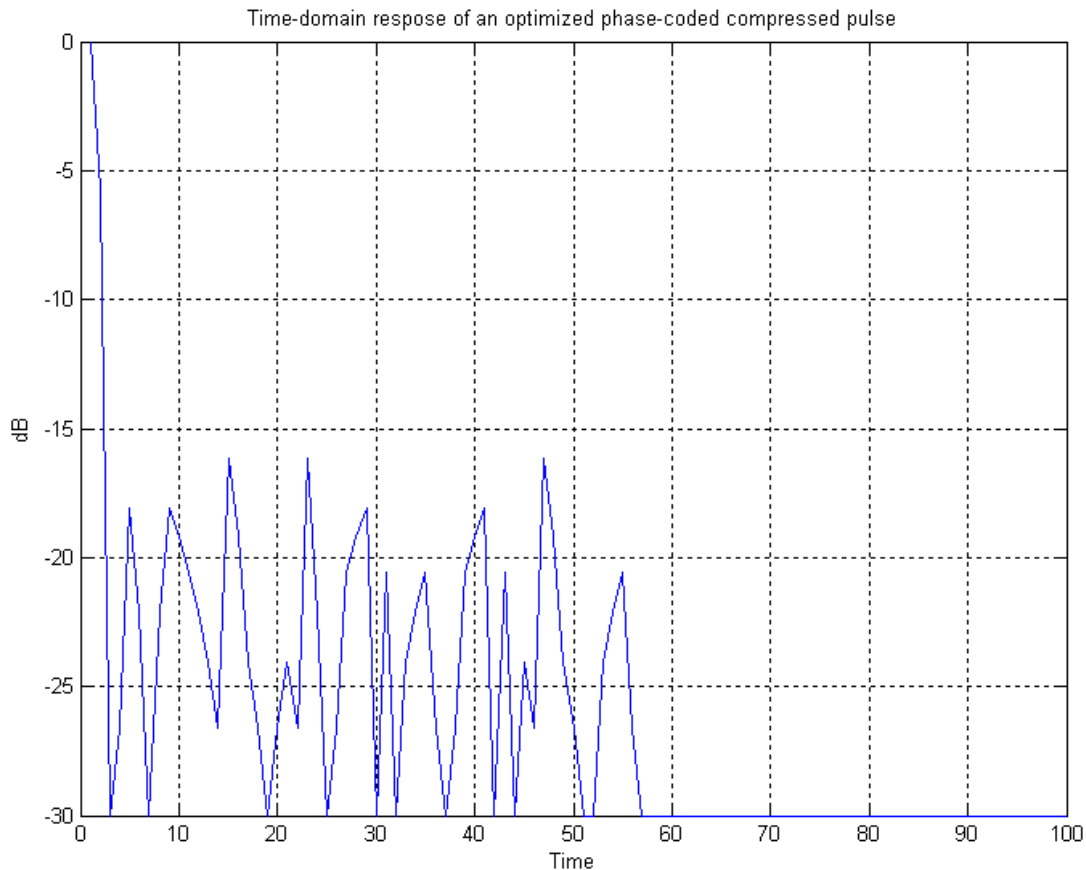


Figure 3.23: Time-domain response of an optimized phase-coded compressed pulse

The waveform selected for this project by means of simulation was: "1 0 1 0 0 0 0 1 0 0 1 0 0 1 1 0 0 1 1 1 1 0 1 0 0 0 1 1 1 1 0 0". The software code can be adapted to generate a number of these low side-lobe codes. The radar system will then be able to implement these codes randomly from pulse to pulse.

To compare this result, an optimal binary code with 32-sub-pulses was used as per the Radar Handbook by Merrill I. Skolnik (page 10.18) in the same Matlab program. Figure (3.24) shown the compressed pulse of this particular code. Side-lobes are located below -20dB which would make this a better selection. The computed code was used for further computations.

³Original software code was provided by Mr. P.J. Wolfaardt, RRS. Copyright is reserved.

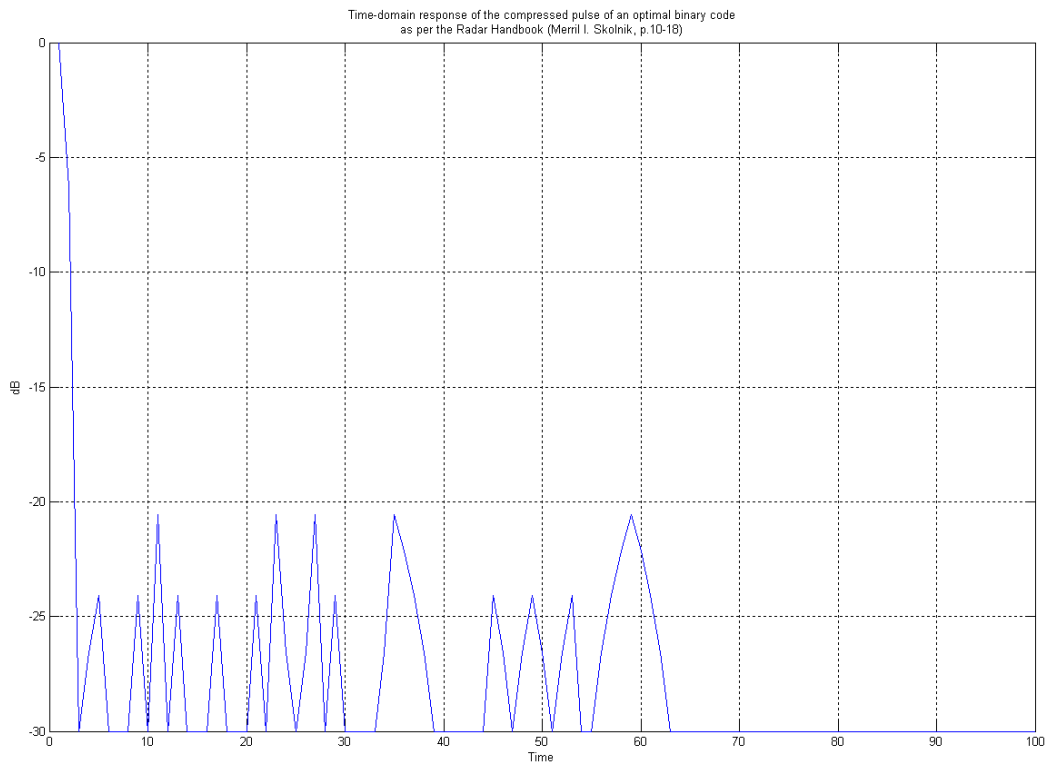


Figure 3.24: Time-domain response of a 32-bit optimal binary code obtained in the Radar Handbook (Merril I. Skolnik, page 10.18)

3.2.2.4.2.3 Implementation The RSP contains both the controls for the transmission of the phase-code via the DDS module as well as the Pulse Compressor module⁴. On the up-going flank of the transmit trigger pulse, the RSP starts to implement the selected modulation sequence. Figure (3.25) shows the phase-coded output control signal. The top trace is the sync-pulse, middle the control signal and bottom the output signal of the transmit DDS module. The last is however of too high a frequency to see the phase shift.

⁴The working of the DDS Controller Module and Pulse Compression Module is discussed in detail in the next chapter.

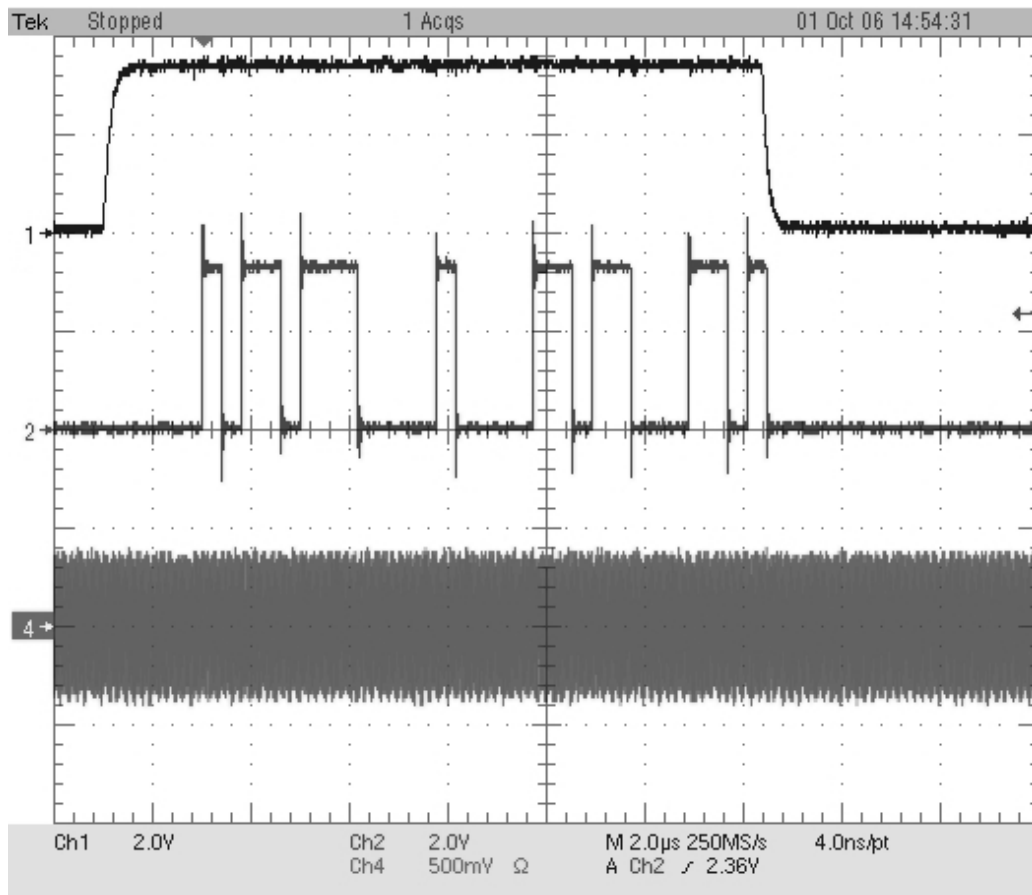


Figure 3.25: Figure of the phase-coded control signal waveform code

The actual compression of the pulse by means of the RSP was confirmed by means of a software vs. hardware simulation result. The results are given in section (3.2.2.6) of this chapter where the pulse compressor section was tested.

3.2.2.4.2.4 Bandwidth problems

To test the modulation of the transmit chain, the transmit signal was routed back into the receiver chain. The signal was routed out of the transmitter chain just after the two-times multiplier module and before the 4W amplifier. It was routed back into the receiver chain, with sufficient attenuation to protect the receiver modules, at the mixers.

The mixed signal, now at the IF frequency, showed dead zones after each phase change. After further investigation, the problem was traced back to the transmit chain 9-times frequency multiplier module. By means of a simulation it was found that on doing a phase-change, higher frequency components of up to 110MHz were generated while the input bandwidth of the module was only later calculated to be only ($\frac{37}{9} = 4,1\text{MHz}$).

When this signal passes through the transmit 9-times frequency multiplier, the internal filters attenuate the signal due to the limited input bandwidth. This lower signal power causes the tripler sections of the 9-times frequency multipliers to switch off and thus no output signal is generated. These triplers require a minimum input power level of -2dBm to operate. Figure (3.26) shows the dead zones as measured by means of a diode-detector on the bottom trace. Figure (3.27) is a zoomed picture.

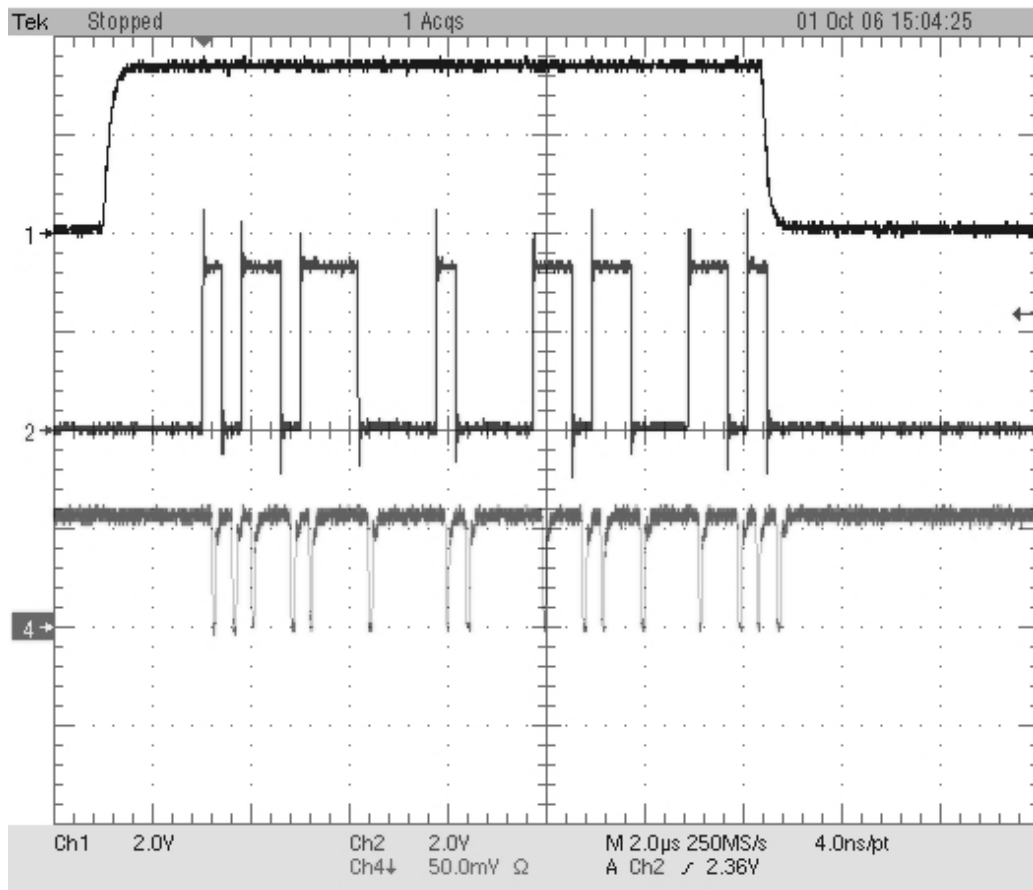


Figure 3.26: A picture of the dead zones at the output of the transmit frequency multiplier module (phase offset = 90°)

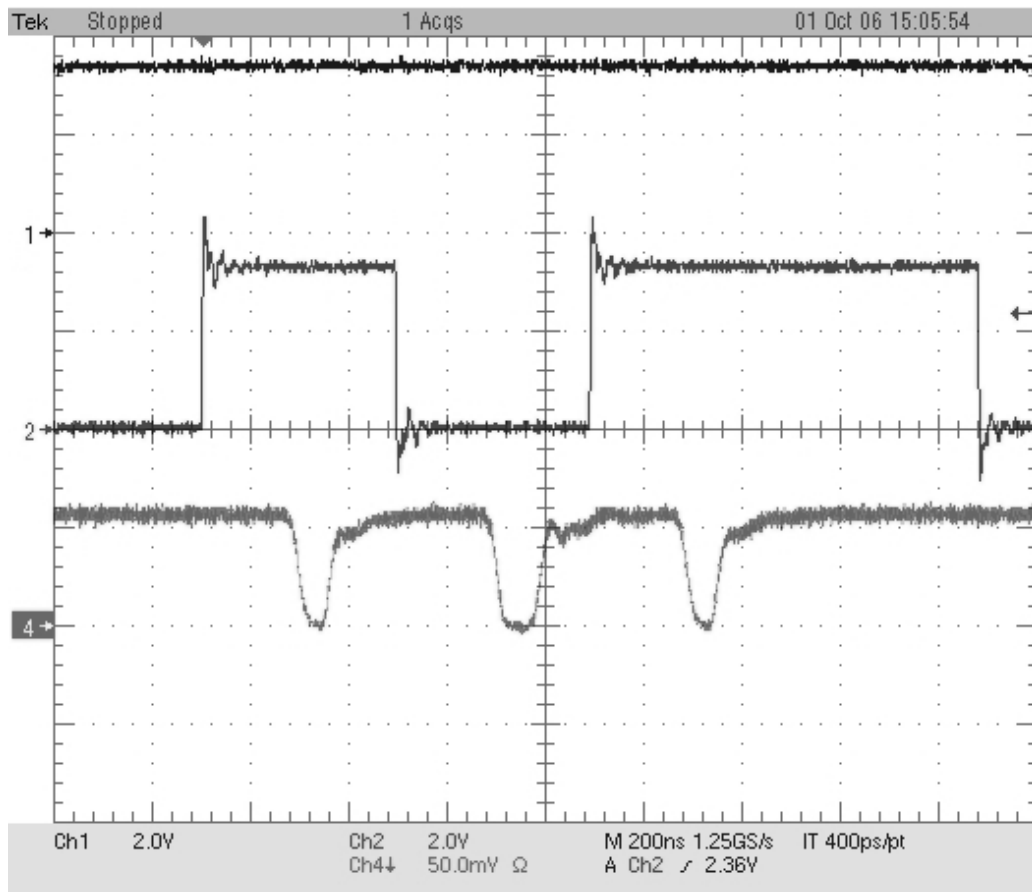


Figure 3.27: A zoomed picture of the dead zones

A few possible solutions were investigated, but only one implemented. In an attempt to lower the higher frequency components generated by the phase modulation, the phase-offset generated by the DDS module was reduced to 10° . This would still produce a 180° phase offset in the transmit signal due to the nine (9) and two (2) multiplication factors in the transmit chain. The problem was solved as shown in see figure (3.28). The bottom trace here is the output of the diode-detector indicating the RF signal.

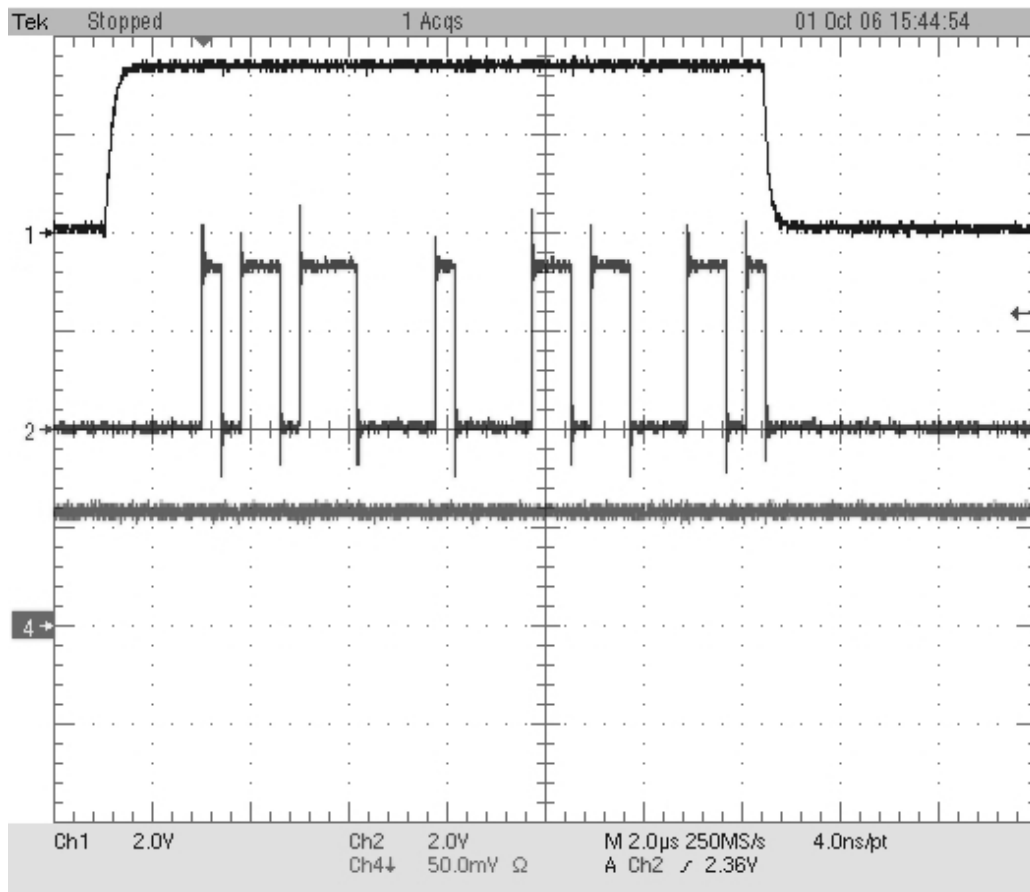


Figure 3.28: Implementation of a 10° phase offset in the DDS and resulting RF output.

During the concept design phase of this project, it was decided to implement the phase-code modulation by means of a quadrature modulator, but this was later changed since the DDS modules provided an easier solution. This would be the best solution, since the modulation could be done at the transmission frequency of $1,8GHz$ and so eliminate almost all the bandwidth issues.

3.2.2.5 RSP Design

This section gives a functional description of the RSP operation. The RSP is made up of four (5) sections as in figure (3.19). These are the “Timing and Control”, “Pulse Compression”, “Burst Maps”, “Doppler Processing” and “Target Detection” sections. The operation of the RSP is discussed on the hand of these sections.

3.2.2.5.1 Timing and Control

Synchronized timing signals are continuously generated in this section to clock various operations which include the RSP synchronization, Sampling and Pulse Modulation. These timing signals are derived from the external transceiver frequency source and governed by the transmit trigger pulse and turning data from the antenna pedestal.

On power-up, the section enters an initialization state where the two (2) DDS modules in the Transmit Chain are preloaded with frequency and phase offset profile settings. After this initialization state an operational state is entered where the DDS Control module will control the Pulse Generation DDS to implement the phase modulation on the transmit signal. This modulation is triggered by the transmit trigger pulse.

3.2.2.5.2 Pulse Compression

The ADC continuously samples the IF output signal of the Receiver chain at a sample frequency of 40,5MHz. This produces an output sample stream in the form of “I”, “Q”, “-I”, “-Q”, “I”, “Q”, “-I”, “-Q”, etc. where “I” is the in-phase data channel and “Q” the quadrature-phase channel data.

The Down Converter is used to correct the sign of every third and fourth sample and produce two (2) output data streams: One for “I”-channel data and the other for “Q”-channel data. These two (2) streams are then passed through digital Finite Impulse Response (FIR) filters to minimize noise in the data after which the result is decimated.

These decimated I-channel and Q-channel data streams are then repacked in a data packer to ensure that the data width and length are correct for the Pulse Compressor module. The output is buffered in a First In First Out (FIFO) memory bank since the sampling and pulse compression operations are not synchronized.

Finally the Pulse Compressor module reads a full PRI from the FIFO memory bank and performs pulse compression. This is done by transforming the read PRI to the frequency domain and performing a multiplication of this PRI data with a pre-calculated weighting function. Figure (3.29) shows a block diagram of the Pulse Compressor.

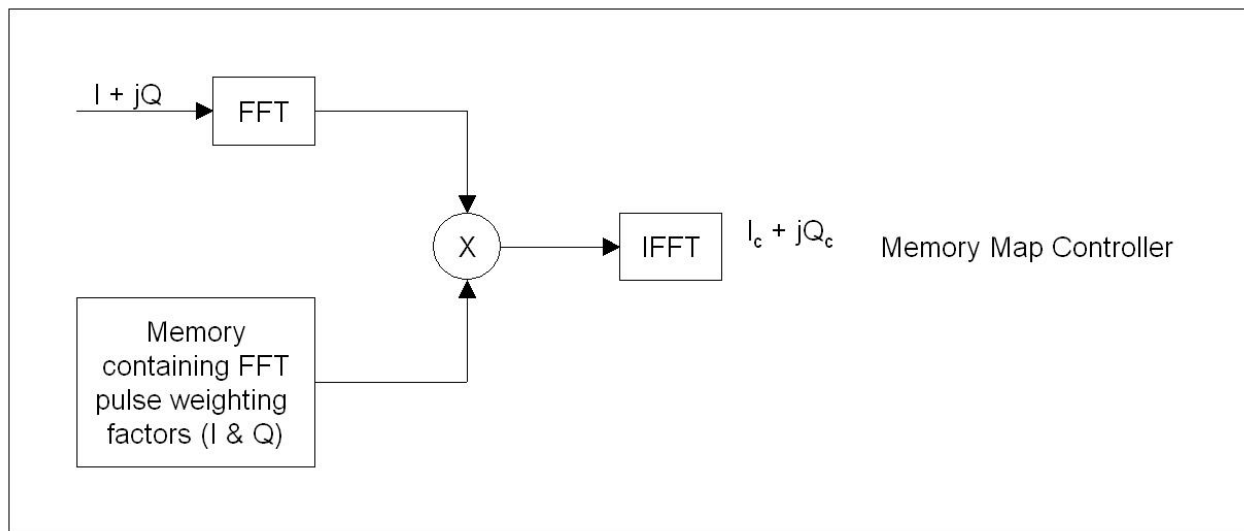


Figure 3.29: Block diagram of the pulse compressor module

The pulse compressor operates using five (5) stages:

1. Firstly the I and Q PRI data is clocked in respectively and stops when an end of PRI marker is received. This marker is embedded in the PRI by the module populating the FIFO memory bank.
2. The module then regards the first three (3) samples as header information which could contain PRI counting data for example. These three values are output immediately. The rest of the data is stored in bit-reversed format.
3. The FFT of is then computed.

4. The result is multiplied with the weighting functions from the pulse code which are already in the frequency domain and stored bit-reversed.
5. The output of the multiplication is then transformed back to the time-domain by an IFFT procedure.
6. Finally the result is exported on request.

The result is transformed back to the time domain and clocked to the Burst Maps.

3.2.2.5.3 Burst Maps

Two (2) PRI Burst Maps are generated, each containing data of 64 pulse compressed PRIs. As soon as a single Burst map is full the second will be populated while the first is being clocked out to the Doppler Processor. The process then switches back to the state where the first Burst map is being loaded with PRI data while the second is routed to the Doppler Processor.

A burst map is loaded horizontally with compressed data from the Pulse Compressor and unloaded vertically, thus 64 samples on a per range bin basis, to the Doppler Processor.

To manage this writing and reading of data in and out of the Burst Maps, a Memory Manager module is used to control the process. The process is run on a priority basis where the Doppler Processor has a higher priority than the Pulse Compressor, but Doppler processing can only start once there exists a full Burst Map. Also, once a write or read sequence of either the Pulse Compressor or Doppler Processor has started, that sequence has to be completed before another process can be started. This is implemented by means of a “Request” and “Grant” scheme where either the Pulse Compressor will request to write to the Burst Map or the Doppler Processor will request to read from the Burst Map.

Figure (3.30) shows the layout of a Burst Map and the processing directions.

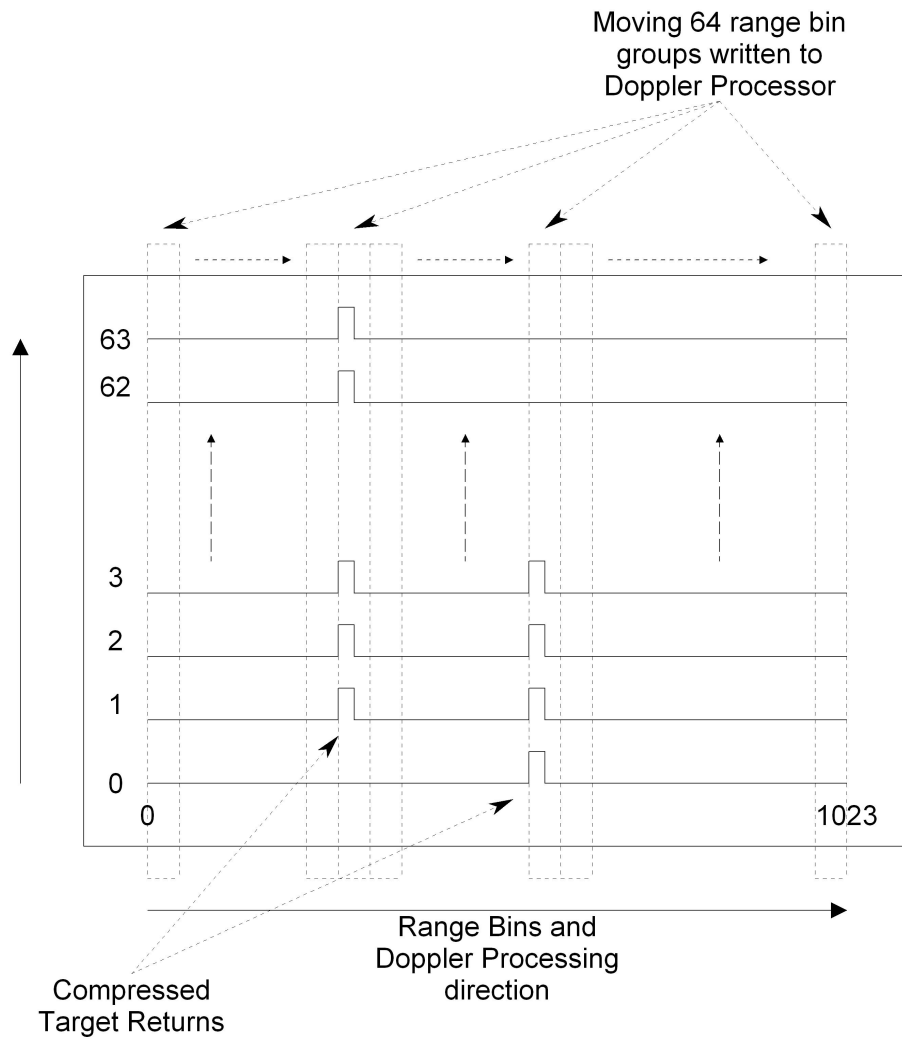


Figure 3.30: PRI Burst Map layout and processing directions

3.2.2.5.4 Doppler Processing

This module integrates 64 PRI range bins to compute the velocity information contained in these range bins. This is done by calculating the FFT (64-point) of the input data. The same module as for the Pulse Compressor is used, but the inverse FFT operation is not performed. Again the module is buffered by means of a FIFO module since the operation is not synchronized to other operations but started once sufficient data is available in the Burst Maps.

The magnitude of the velocity output of the Doppler Processor is calculated by means of the following equation and routed to the Target Detection section.

$$Mag = \sqrt{I^2 + Q^2}$$

3.2.2.5.5 Target Detection

Target detection is done by means of two (2) filter modules that average the input magnitudes time. By comparing these outputs it can be determined if a target is present in the clutter or not.

The two (2) filter sections implement alpha filters which effectively average the clutter the radar has detected over the past. The difference between the filters is the time period over which these filters integrate clutter. Once the filter output value with the shorter time period exceeds that of the longer period, a target is declared.

The results are then routed to an external display system for the radar operator to evaluate.

3.2.2.6 RSP Implementation

Although the RSP was designed and implemented in full, only the Timing and Control and Pulse Compression sections were tested. This was done by means of hardware simulation using the Quartus II design software. The results were compared with Matlab simulated results of the same problem.

The Timing and Control section however was physically tested and this provided the DDS results in the Transmit chain modifications of this chapter.

The Pulse Compression section was evaluated by means of providing the section with stimulus of a single PRI containing the transmission pulse code. This PRI data was routed into the Down Converter by means of a memory bank. This simulated the output of the ADC.

An IF signal of 70,875MHz containing the pulse code which was eight (8) times oversampled (due to the decimation factor of eight (8) in this section) was generated. This signal was sampled at 40,5MHz, captured and placed in the memory bank. The simulated data was then clocked out of the memory bank into the Down Converter at the sample frequency of 40,5MHz.

The PRI data was then sign corrected in the Down Converter and routed to a Decimator module where it was decimated by a factor of eight (8) thus producing two (2) output streams (I-data and Q-data) of 5,0625MHz each.

The result was buffered in the FIFO and on request passed to the pulse compressor via a zero-padder module which added zero values to the end of the PRI data to produce a trace of 1024 range bins. A FIFO Level module was used to indicate that a complete PRI was sampled and ready to be routed to the Pulse Compressor.

The Pulse Compressor received the data, performed the compression and the output was graphed together with the Matlab simulated values.

Figure (3.31) shows the hardware pulse compression section of the RSP that was used to evaluate the functioning of the section compared to Matlab simulated results.

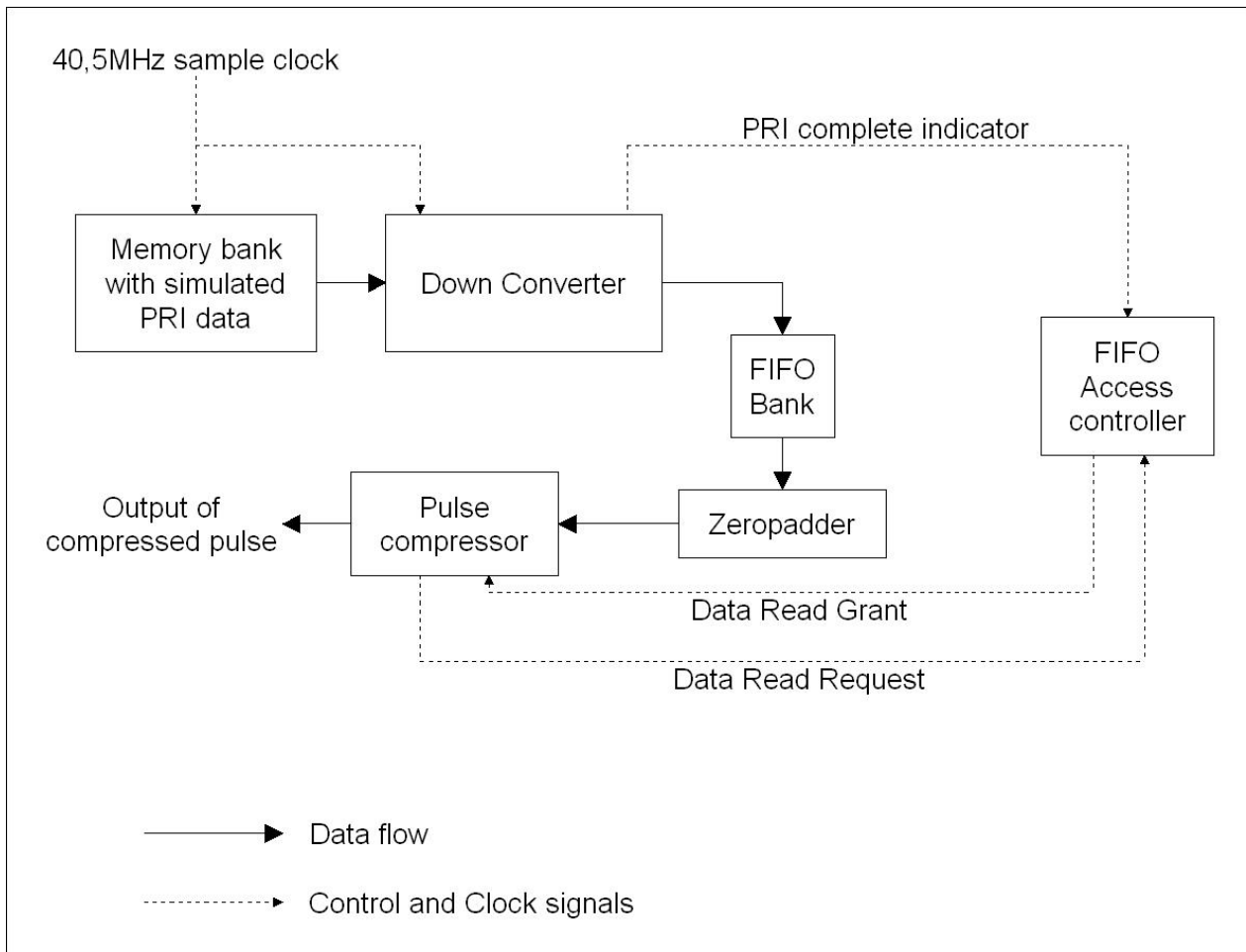


Figure 3.31: Block diagram of the hardware setup for the pulse compression test

Figure (3.32) shows the compressed pulse located at the start of the PRI. The solid line represents the compressed pulse as was calculated by Matlab, while the circular indicators show the result that was achieved by means of the Quartus II simulation of this section. Note that the values are normalized and shown in dB values relative to the largest value.

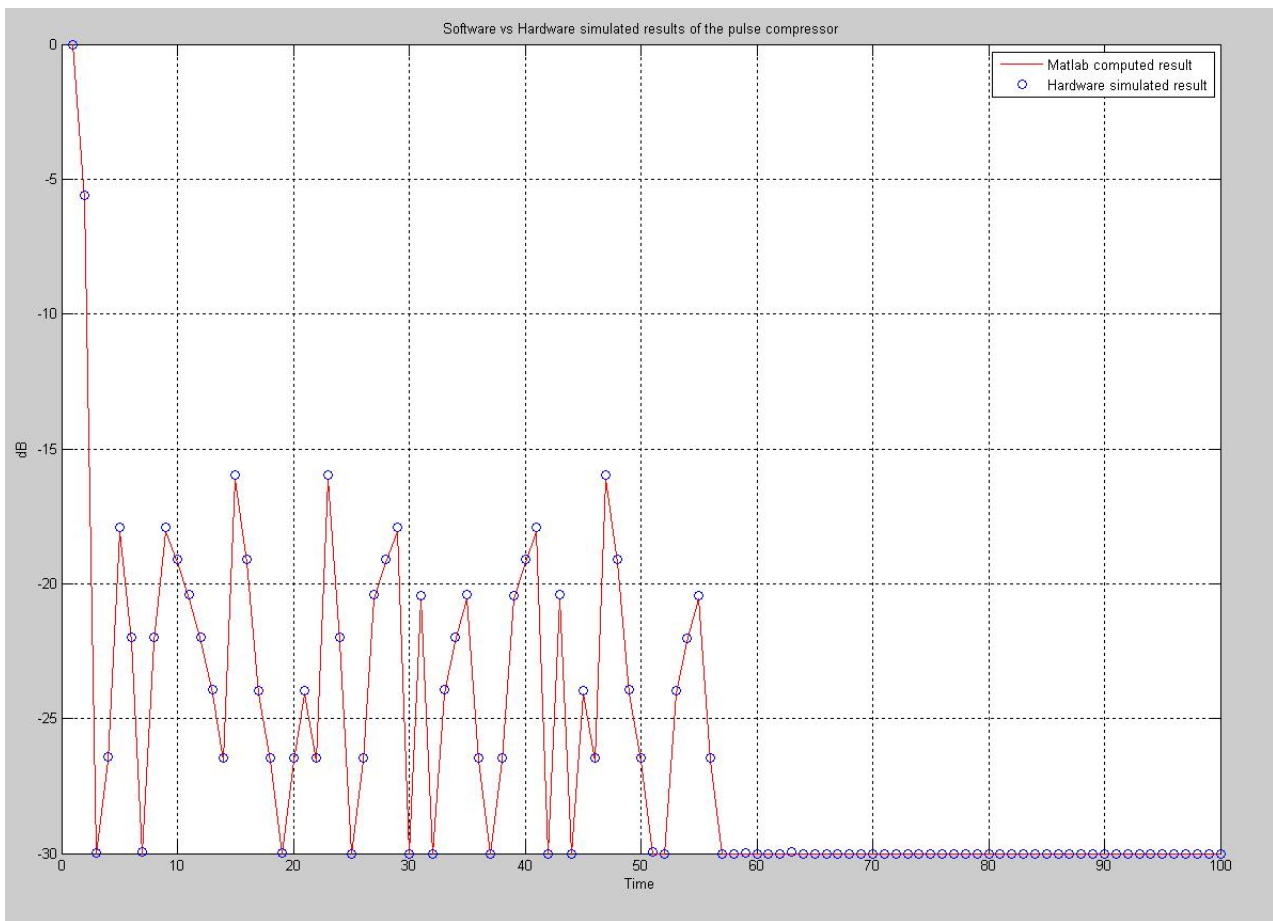


Figure 3.32: Graph of the software vs. hardware simulated pulse compression

Figure (3.33) shows the normalized error of the Quartus II simulation to the calculated Matlab simulation. The error is due to rounding errors in the Pulse Compressor since floating point multiplication and division are not possible. Given this fact, there is a close match between the two sets of results.

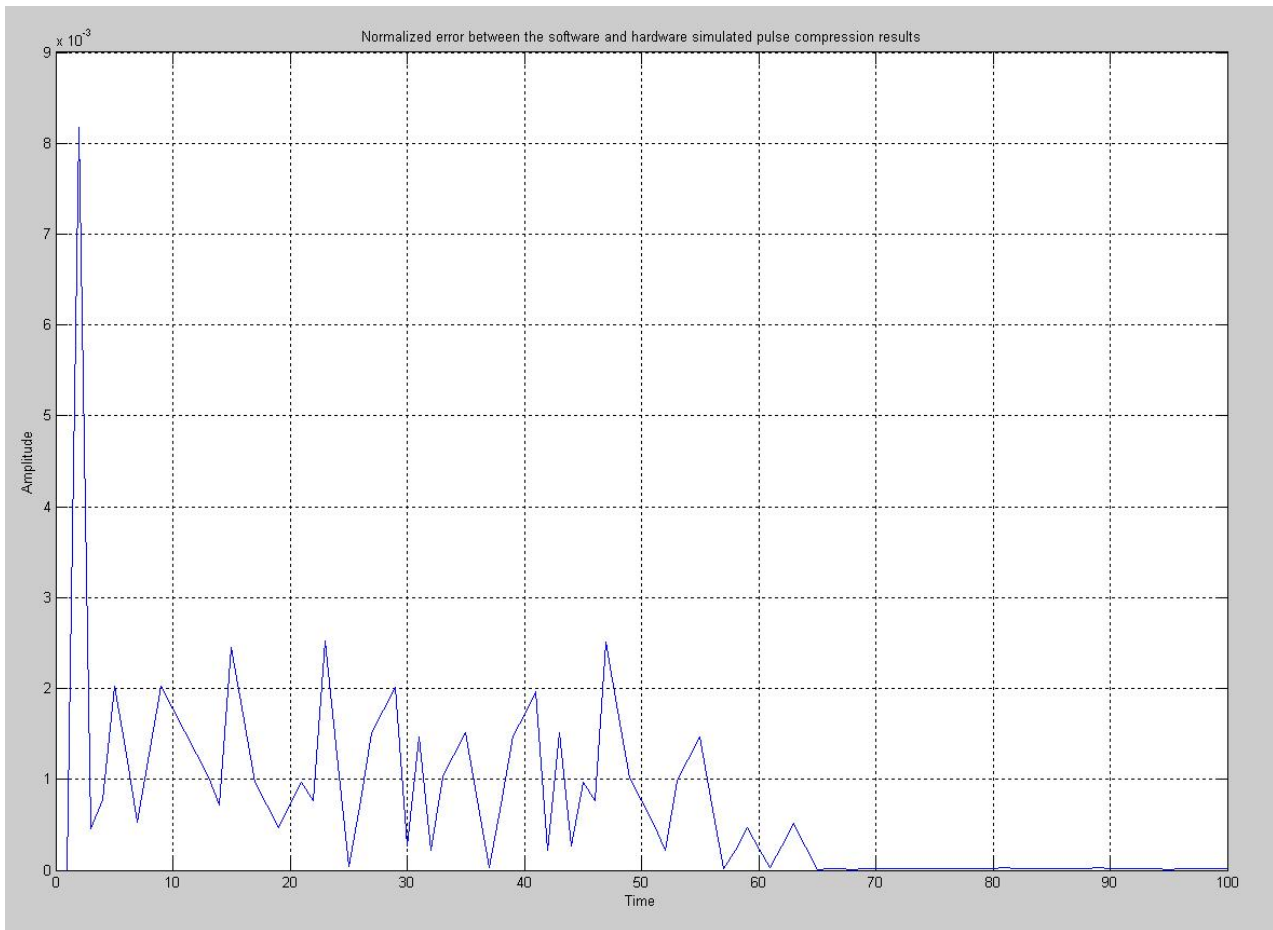


Figure 3.33: Graph of the error between the software and hardware simulated pulse compression

3.3 Conclusion

In this chapter the modifications to both the RF-subsystem and RSP were discussed.

The introduction of DDS modules used to generate the modulated pulse, as well as the offset frequency, were shown. All the needed modules required for the complete implementation of the DDS modules were shown as well as measured results of each component.

The implementation of the RSP was discussed. Focus was put on the IF sampling scheme used as well as the design and implementation of the pulse code used during pulse compression. Measured results were shown of the modulation of the transmit pulse. It was found that the bandwidth in the system was a crucial aspect of the design when implementing phase modulation.

A functional description of the newly designed RSP was given as well as the implementation and results achieved for the timing and control and pulse compression sections. It was found that these two sections function as designed.

Chapter 4

Conclusions

This project entailed the modification of an existing radar system to increase performance, particularly range resolution. Pulse compression was implemented to achieve this.

4.1 RF Sub-system modifications

It was found that implementing pulse compression requires major modifications to both the RF sub-system and to the RSP of the radar. This resulted in the project taking far longer than anticipated and consequently RSP could not be fully tested.

DDS modules were used to synthesize the modulated transmission pulse and also to create an IF signal for the RSP ADC to sample. These DDS modules were installed in the existing LO and Exciter chain of the Transmitter. The Exciter chain was subsequently renamed to the Pulse Generation chain.

Introduction of these DDS modules required other modules to allow seamless integration with the rest of the existing chains. Replacement of the times-9 frequency multiplier modules which were notorious for introducing errors into the system was also required which in turn required amplification modules. Available broken modules were used to construct the amplifiers needed in the replacement modules.

Various filters were introduced to filter harmonic and spurious signals in the case of the sections where the DDS modules were implemented and noise considerations in the case of the IF output to the RSP ADC.

A phase modulation code was successfully designed by means of software simulation with sufficiently low side-lobes of -16dB. However it was found that phase modulation required higher bandwidth than the modulation frequency due to the high rate of change of the phase. This resulted in the time-9 modules turning off due to the lower power levels introduced by the filters and thus effectively turning off the transmit pulse.

A workaround was found by lowering the phase offset from 90° to 10° . This still produced a phase offset of 180° at the antenna due to the multiplication factor of 18 in the system. A better solution would be to use frequency modulation or implement quadrature modulation of the final transmit frequency rather than modulation earlier in the Transmit chain. It was shown that the upgraded RF sub-system was fundamentally capable of implementing the designed phase modulation.

4.2 Signal Processing modifications

A new RSP which included sections for Timing and Control, Pulse Compression, Burst Map controlling, Doppler Processing and Target Detection was designed but of which only the Pulse Compression section was tested.

The RSP was capable of driving and controlling the DDS modules in the RF sub-system to phase modulate the transmit pulse and produce an IF signal of 70,875MHz while being synchronous and running off the same clock as the RF-subsystem. Various synchronous timing signals were supplied to the other sections of the RSP.

The Pulse Compression section was implemented and tested by means of verifying the results achieved to theoretical results generated using Matlab. It was found that the Pulse Compressor module down converted and fed the Pulse Compressor module with simulated data and produced a compressed pulse nearly identical to the theoretical one. Errors are due to the theoretical results being achieved using a floating point numerical system whereas the hardware simulated results used mathematical operations of a binary system which excludes fractions.

Due to many of the RSP processes running in parallel, the largest limiting factor of the RSP development board is the limited amount of individually addressable memory blocks. Newer development boards have a single large memory block with firmware controlling the access of memory. This creates timing problems for RSP processes which have to wait for memory access. A solution would be to design a custom board with a large number of individually addressable memory blocks. This however could be very expensive on a small scale.

4.3 General

The number of hardware components needed added to the complexity of this project. Obtaining the needed electronic components and samples from suppliers, at the amounts needed, proved time consuming and sometimes impossible. An example is the expensive surface mounted capacitors that were needed in the transceiver upgrade modules. These required to be of high quality from a specific manufacturer. An example are the minimum order quantities are in the thousands while only three (3) were needed for this project.

Construction and testing of these hardware components used time which would have been spend on the system itself. The insight into the field of radar RF engineering was the trade off.

The RSP of this project, although not fully exploited, shows that the implementation of a firmware based RSP on a single FPGA device is possible. The FPGA device used had 25 000 logic element blocks and only 25% was reported in use. Block sizes of 180 000 (and growing) are not uncommon today making possibilities endless. This however creates a support problem, i.e.

Radar systems are primarily used for military defense. Due to the high cost of these system it must operate and be maintained over decades in the field. With the pace at which FPGA technology advances, component obsolescence becomes a major problem for radar development houses.

This results in an increase in development cost due to the continuous redesign of hardware and firmware required since a particular device is not available on the market anymore and must be replaced by a newer variant. Using platform independent software therefore does not solve the problem. This is true for both firmware and software architectures and obsolescence issues will always be an part of engineering work.

4.4 Future work

The ERS was, and will be, a very useful development bench for future projects to modify and implement newly designed hardware and firmware. Future work may include:

- The full implementation and testing of the present project's RSP which would largely include the processing of velocities using Doppler Processing.
- Various target detection techniques can also be evaluated. These may include the suggested alpha-filters or CFAR filters.
- The implementation of antenna beam-shape interpolation in the RSP to increase the azimuth resolution of the ERS. The current antenna beam-width is 9° and this could be reduced by a factor of 10 using the correct techniques.
- The implementation of the RSP on transceivers of other radar systems such as the *EDR120 L-band*¹ radar system. The *EDR120* has a superior antenna and transceiver to that of the ERS. This will result in performance increase in operational range and azimuth resolution when compared to the ERS.
- The use of multiple upgraded ERSs to investigate the design of a matrix radar system where one system can use another system's transmit pulse to detect targets.
- A full redesign of the RF sub-system. The existing ERS Transceiver-module is rather complex and with the technology advances of the past years this design could be simplified and improved.
- The upgrade of ERS to a 3D radar system while still using only one firmware device. This would entail a large amount of work since multiple receivers would be required as well as multiple signal processing chains.

4.5 Personal note

This project started as a study into the digital field of FPGAs. It quickly also incorporated RF design and signal processing. It finally turned out as system engineering project where a large amount of work was done to integrate various engineering disciplines.

This project also provided the means to understand the functioning of a total radar system and representing this system and other radar parameters mathematically. Engineering principles were applied to design modifications to the RF sections both mathematically and physically.

Knowledge was obtained in the field of radar signal processing and the implementation of special processing techniques such as Digital Pulse Compression and Doppler Processing. This was achieved during the design of this RSP itself.

Difficulty was experienced in the writing of this document to give a complete picture of the amount of work which went into a project of this scope. Not everything could be captured and this document will never be fully complete. Then there is, as with every other project, always just a small addition that would be "nice" to include.

Finally, the knowledge obtained in the field of radar systems will continue to be very valuable in my future career.

¹The *EDR120* radar system was developed by RRS.

Bibliography

- [1] *Eekhorning Radar System*, volume 3, chapter Theory of operation. 1990. Publication 7255. 4, 12, 15, 16, A-1, E-2, E-3, E-4
- [2] Inc. Analog Devices. *Data Conversion Handbook*, chapter 2-2: Sampling Theory, pages 73–82. Analog Devices, Inc., 2005. 40
- [3] David K. Barton and Sergey A. Leonov. *Radar Technology Encyclopedia*, page 253. Artech House, Inc., 1998. E-3
- [4] David K. Barton and Sergey A. Leonov. *Radar Technology Encyclopedia*, page 251. Artech House, Inc., 1998. E-6
- [5] Altera Corporation. *Stratix EP1S25 DSP Development Board Datasheet*. Altera Corporation, USA, www.altera.com, ref. 1.4 edition, 2003. xii, B-7, B-8
- [6] Analog Devices. *1 GSPS Direct Digital Synthesizer (AD9858) datasheet*. Analog Devices, USA, www.analog.com/dds, rev. a edition, 2003. B-3
- [7] Analog Devices. *DDS DAC Output Evaluation Board (AD9858PCB) datasheet*. Analog Devices, USA, www.analog.com/dds, ref. 0 edition, 2004. B-3
- [8] Simon Kingsley and Shaun Quegan. *Understanding Radar Systems*, chapter 6 - Designing radar waveforms, pages 145–154. Scitech Publishing, Inc. Mendham, New Jersey, 1999. 42
- [9] Simon Kingsley and Shaun Quegan. *Understanding Radar Systems*, chapter 6.9 - Phase coding, pages 150–154. Scitech Publishing, Inc. Mendham, New Jersey, 1999. 45
- [10] Simon Kingsley and Shaun Quegan. *Understanding Radar Systems*, chapter 1 - Fundamentals, pages 1–24. Scitech Publishing, Inc. Mendham, New Jersey, 1999. E-2, E-3
- [11] Simon Kingsley and Shaun Quegan. *Understanding Radar Systems*, chapter 2 - Designing a surveillance radar, pages 25–47. Scitech Publishing, Inc. Mendham, New Jersey, 1999. E-2
- [12] Bassem R. Mahafza and Atef Z. Elsherbeni. *MATLAB Simulations for Radar Systems Design*. Chapman & Hall/CRC, 2004. E-6, E-9
- [13] Merrill I. Skolnik. *Radar Handbook*, chapter 10 - Pulse Compression Radar, pages 10.1–10.3. McGraw-Hill, Inc., 2nd edition, 1990. 42
- [14] Merrill I. Skolnik. *Radar Handbook*, chapter 10.6 - Phase-coded waveforms, pages 10.15–10.26. McGraw-Hill, Inc., 2nd edition, 1990. 45
- [15] Merrill I. Skolnik. *Introduction to Radar Systems*, chapter 1 - The Nature of Radar, pages 1–14. McGraw-Hill International Book Company, 2nd edition, 1981. International Student Edition. D-1, E-2, E-8

- [16] Merrill I. Skolnik. *Introduction to Radar Systems*, chapter 2 - The Radar Equation, pages 15–67. McGraw-Hill International Book Company, 2nd edition, 1981. International Student Edition. E–2, E–5
- [17] George W. Stimson. *Introduction to Airborne Radar*, chapter 13 - Pulse Compression, pages 163–176. Scitech Publishing, Inc., 2nd edition, 1998. 42

Appendix A

ERS Technical Information

A.1 Antenna System

The Antenna System consists of the antenna and the antenna pedestal. The particular specifications are listed below[1]:

A.1.1 Antenna Information

The antenna is constructed of four (4) horizontal planks with the following specifications:

• Transmission frequency	1,75 to 1,85GHz
• Antenna gain	> 21,5dB
• Horizontal beam width at the $-3dB$ points	$\pm 9^\circ$
• Vertical beam width at the $-3dB$ points	18°
• Antenna tilt	-3° to $+10^\circ$
• Polarization	Horizontal

A.1.2 Pedestal Information

The pedestal rotational speed can be selected to be either 10rpm or 30rpm. Azimuth information is provided by means of a north pulse (thus once every 360°) while the shaft encoder pulse is generated every $0,5^\circ$. The latter is generated by two 2° pulses shifted by 90° . North alignment is done by means of a circular dial to compensate for the difference between the north pulse and true north.

A.2 Transceiver Unit

The particular transceiver specifications are listed below[1]:

A.2.1 Transmitter Information

The transmitter specifications are listed below (note that the signal power is measured at the antenna output connector):

- Frequency 1,75 to 1,85GHz
- Peak power > 115W
- Average power > 10W
- Transmitter type Solid state
- Pulse width (at 10km range) 6, 6 μ s
- Pulse width (at 20km range) 13, 3 μ s
- PRI (at 10km range) 80 μ s
- PRI (at 20km range) 160 μ s

A.2.2 Receiver Information

- Receiving losses <5dB
- Noise figure <3dB
- Intermediate frequency None
- Frequency channels 8 selectable crystal frequencies
- Bandwidth (at 10 km range) 75kHz
- Bandwidth (at 20 km range) 150kHz
- Dynamic range of the receiver >55dB
- Video output (I and Q channels) >3V driven into a 50 Ω load

Appendix B

Upgrade Hardware Module Descriptions

B.1 9-time Frequency Multiplier Modules

The 9-times frequency multiplier module was originally designed by REUTECH RADAR SYSTEMS (RRS) for production use. These frequency multipliers were used as a COTS item in this project¹. Three (3) frequency scaled versions were constructed and calibrated. These modules were used as a system clock multiplier to provide a reference clock for the DDS modules, an exciter chain frequency multiplier and a LO chain frequency multiplier. These chain multiplier modules are used to multiply the output signals of the DDS modules back to the required frequency values since the DDS modules act as frequency dividers.

The only difference between the three modules is the centre frequencies of the internal band-pass filters. Figure (B.1) shows the design block diagram of a frequency multiplier module.

¹The complete design is RRS company property and can thus not be given in full extent.

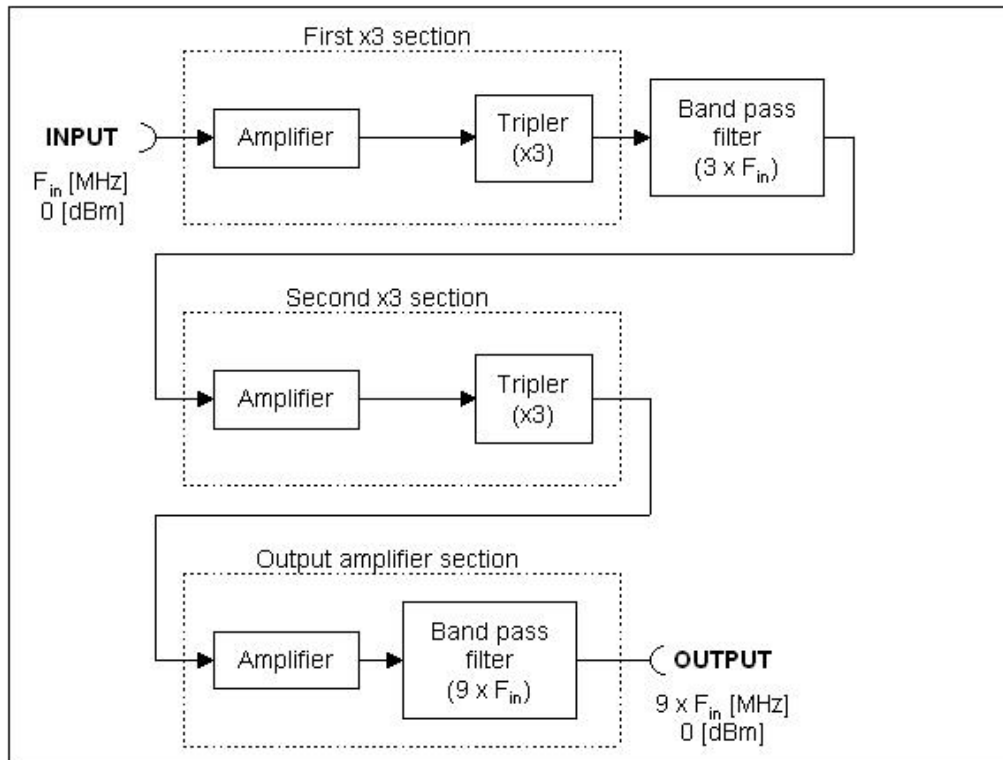


Figure B.1: Block diagram of the 9-times frequency multiplier module

The design consists of two (2) frequency tripler sections placed back to back with appropriate amplification and harmonic filtering. The required input signal power level is 0dBm. The module will produce an output signal that is nine-times higher in frequency at a output power level of 0dBm.

The multiplier hardware consists of a double sided PRINTED CIRCUIT BOARD (PCB) populated with various soldered-on components. The PCB is housed in an aluminum enclosure with dual lids which are secured with screws. The enclosure acts as RF screen for the module. The module is powered by a $+10V_{DC}$ power supply.

Figure (B.2) shows a photo of an open 9-times frequency multiplier module and figure (B.3) shows the same module but with the top lid secured.

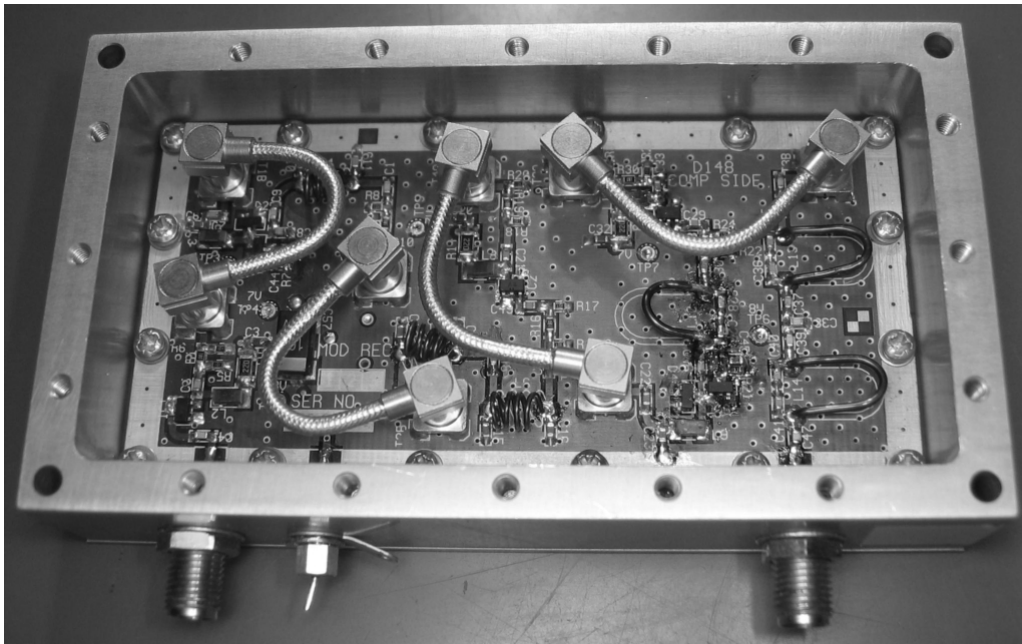


Figure B.2: Photo of an open 9-times frequency multiplier module enclosure showing the PCB.

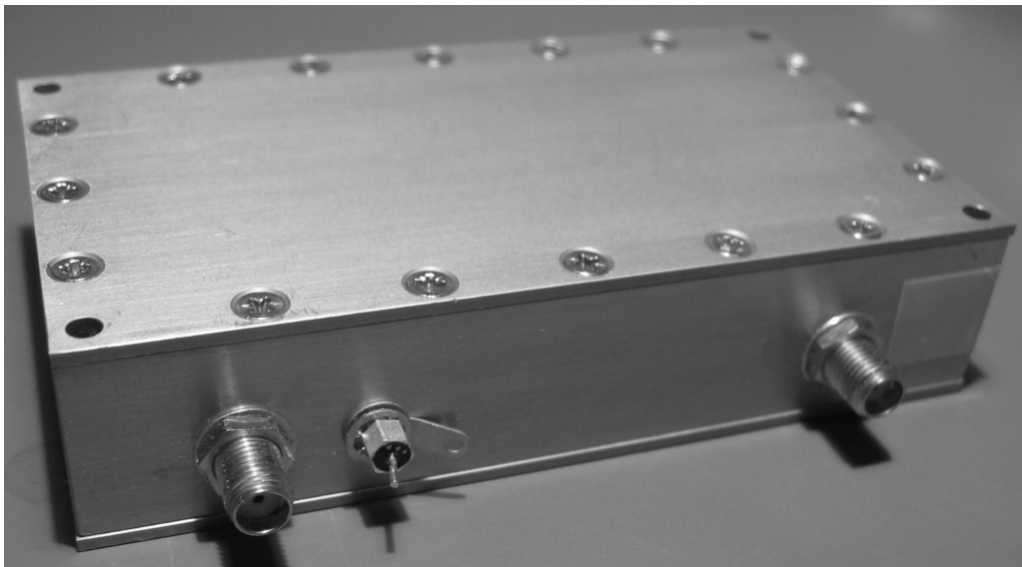


Figure B.3: Photo of the 9-times frequency multiplier module enclosure with both lids secured.

B.2 Direct Digital Synthesizers Modules

Two *Analog Devices*² *DDS DAC Output Evaluation Boards*[6, 7] (AD9858PCB) were used in the upgraded transceiver: one for the transmit chain and the other for the LO chain. These evaluation boards allow for the easy implementation and testing of the *Analog Devices 1GSPS*³ *DDS chip* (AD9858) in development and testing setups. Figure (B.4) shows a photo of the AD9858PCB evaluation board.

²<http://www.analog.com>

³Giga Samples per Second

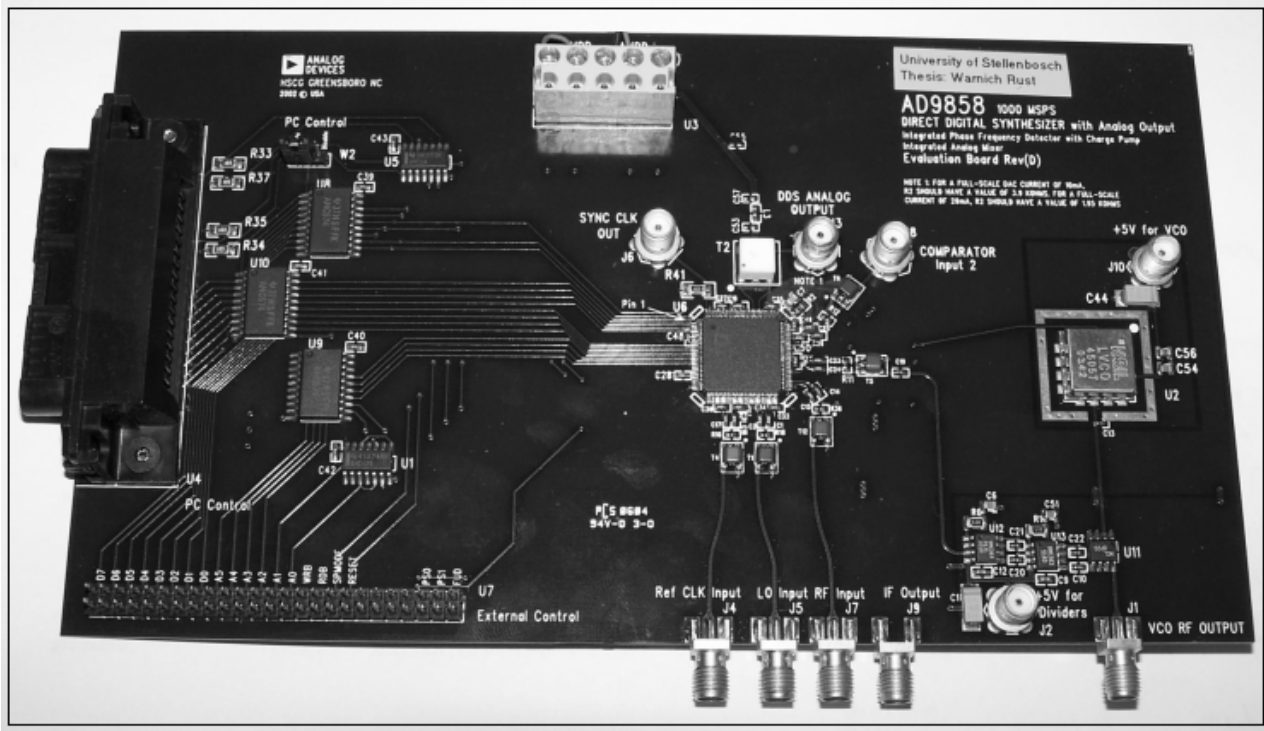


Figure B.4: A picture of the Analog Devices DDS DAC Output evaluation board (AD9858PCB)

The AD9858 DDS chip implements a high speed 10-bit DIGITAL-TO-ANALOGUE CONVERTER (DAC) that can operate at speeds up to 1GSPS. Using this technology the DDS can act as a digitally programmable frequency synthesizer capable of generating a frequency-agile wave output of up to 400MHz. Utilizing the on-chip divide-by-2 feature, the DDS can be driven from an external clock source of up to 2GHz.

The output frequency setup is done by means of programming register for any of four (4) profiles. These profiles define the FREQUENCY TUNING WORDS (FTW) and PHASE OFFSET WORDS (POW) used to produce the correct output frequencies and phase offsets. The FTW is effectively used as a scaling factor of the input reference clock (REFCLK). The relation of the output frequency (F_o) and REFCLK is given by the following equation:

$$F_o = \frac{(FTW)(REFCLK)}{2^N} \quad (B.1)$$

where $N = 32$ for the AD9858 and the FTW is a 32-bit integer value. It is possible to change between preset profiles (and thus between different (F_o) values) at a rate of $(\frac{REFCLK}{8})$. The frequency resolution is shown to be $(\frac{REFCLK}{2^{32}})$ and $(\frac{360^\circ}{2^{14}} = 0,022^\circ)$ for the phase offset resolution since the POW is a 14-bit value. Implementing frequencies and phase offsets by means of selecting profiles provides a very quick solution for change implementation since reprogramming registers would result in slower output setup times. This makes the AD9858 very useful for implementing 90° phase modulation.

An additional feature of the AD9858 is its capability to implement frequency modulation. This is done by means of setting up the DELTA-FREQUENCY TUNING WORD and the DELTA-FREQUENCY RAMP RATE WORD registers. The first parameter indicates the frequency that is added to the current output frequency and the second the rate at which these delta values are added to the current output frequency. This feature is not used in this project since phase modulation is done.

The evaluation board also provides means to access and implement the on-chip charge pump, phase frequency detector and analogue mixer of the AD9858 used for applications where both high speed synthesis, PHASE-LOCKED LOOP (PLL) and mixer functions are required. Further application areas of the AD9858 include VHF/UHF LO synthesis, tuners, instrumentation, agile clock synthesis, cellular base station hopping synthesis, radar and Sonet/SDH clock synthesis.

When multiplied through the LO chain and mixed with the received signal, an IF signal of 70,875MHz is produced. This signal is then sampled by the ADC. “(G)” is the transmit DDS module used to generate the phase modulated transmit signal. Both these modules are band-pass filtered at the output and frequency multiplied before being routed back in to the existing LO and exciter chains respectively.

Please refer to the applicable data-sheets and Appendix (C) figure (C.2) for a functional block diagram of the DDS.

B.3 RF Filter Modules

Four (4) band-pass filters were implemented in the upgraded transceiver. The function of these three (3) filters are to remove the harmonic signals of the carrier frequency. The final filter is used to filter the IF signal, containing the received signal, routed to the ADC. This is done to remove noise from the IF signal.

A generic third order Chebychev band-pass filter module (also designed by RRS) was used in the implementation of these four (4) band-pass filters needed. The filters consist of a small printed circuit board housed in a aluminum box with a screw-on lid. These four (4) filters differ in centre frequency and bandwidth. Figures (B.5) and (B.6) shows pictures of the filter module both open and with the lid secured.

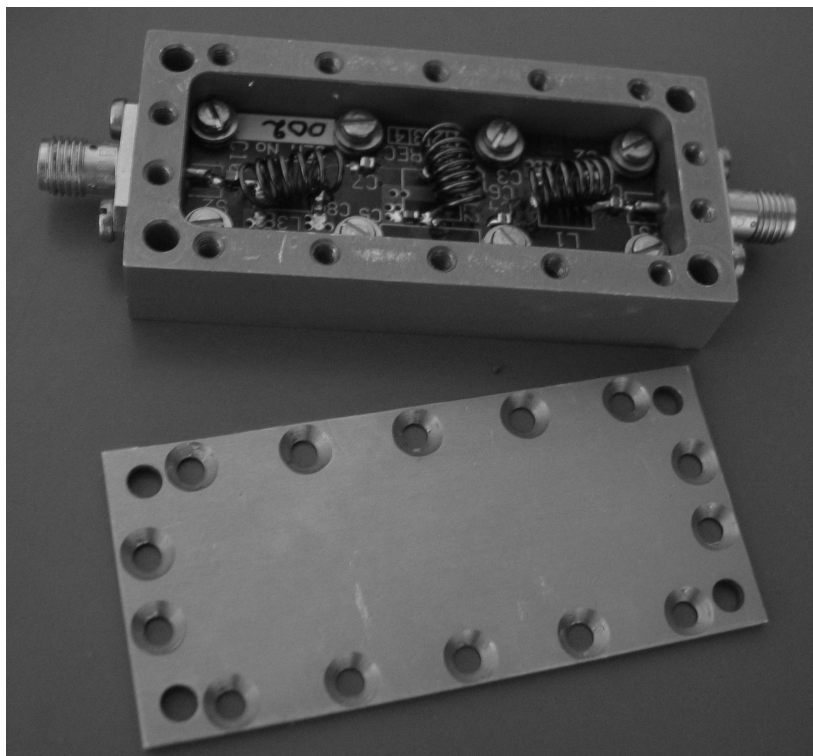


Figure B.5: Picture of an opened filter module

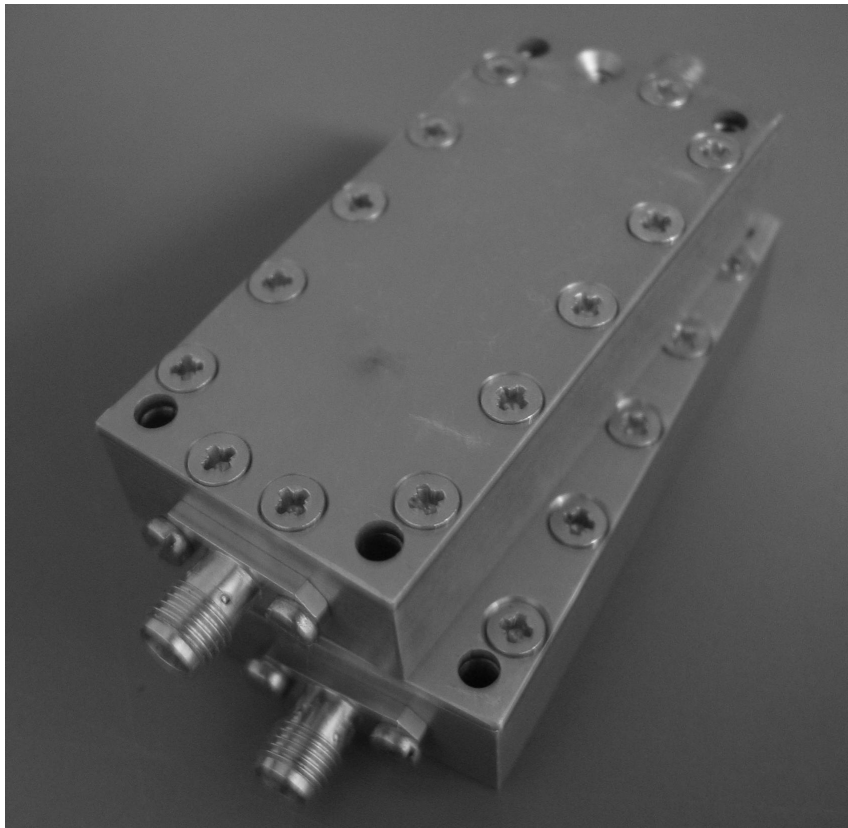


Figure B.6: Picture of closed filter modules

B.4 900MHz Amplifier Modules

The multiplied output of both the transmit and LO chain DDS modules to the rest of the respective chains must be 25dBm. Since the output of the 9-times frequency multiplier modules is only 0dBm, amplifier modules for both these chains were needed. The existing ERS 9-times frequency multiplier modules provided the solution. These modules include a 25dBm amplifier as needed.

Since these old modules are in abundance, these were easily modified by simply routing the input signal directly to the amplifier section. Figure (B.7) shows a block diagram of the modified modules.

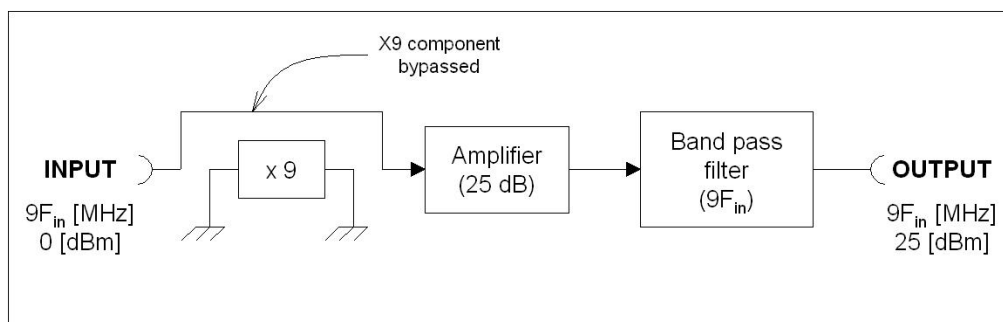


Figure B.7: Block diagram of the modified 9-times frequency multiplier module to function as an amplifier.

The amplifier modules are housed in a aluminum enclosures for screening and is powered by a +18V_{DC} supply.

B.5 RSP Clock Amplifier Module

The system oscillator is divided by means of a power divider with the one port routed to the RSP clock amplifier and the other to the exciter chain. The RSP requires a minimum signal level of 13dBm.

A *RF Micro Devices*⁴ *RF2334 general purpose amplifier evaluation board* was used to implement this amplifier. Refer to the manufacturer's website for a data-sheet of the device.

B.6 IF Amplifier Module

The existing system's receiver chain mixers mix the received signal with the LO frequency and produce a signal at base-band (since the received and LO signal frequencies are the same). This base-band signal is called the video signal and amplified via the video amplifier of which the gain is ± 40 dB. The bandwidth of the amplifier is 150kHz at the 10km range setting and 75kHz for the 20km setting.

Since IF sampling is implemented and the transmit signal has a bandwidth of 2,53125MHz, the video amplifier of the existing system was replaced by a IF amplifier operating at 70,875MHz (which is the IF frequency). The amplifier consists of three cascaded low-noise amplifier modules to achieve the needed gain. *Mini-Circuits*⁵ manufactures a range of plug-in low-noise amplifier modules operating over wide range of frequencies. The model used in the IF amplifier design (model number *MAN-1HLN*) provides a minimum gain of 10dB over a frequency band of 10-500MHz. Refer to the manufacturer for data-sheets and Appendix (C) - Figure (C.3) for more information on the amplifier module used.

Figure (B.8) shows the module block diagram.

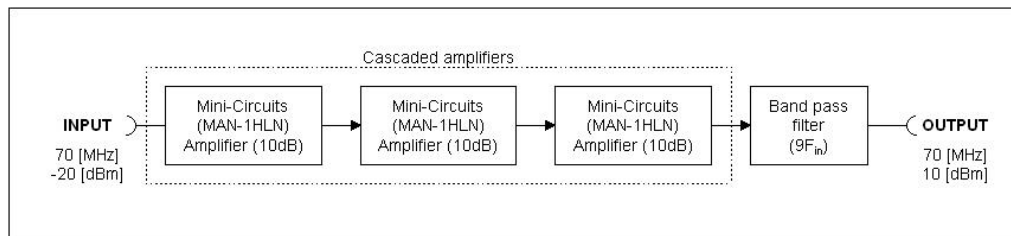


Figure B.8: Block diagram of the IF amplifier module

B.7 Altera Digital Signal Processing Development Board

The entire analogue based signal processor of the EEKHORING RADAR System (ERS) was replaced with a RADAR SIGNAL PROCESSOR (RSP) implementing DIGITAL SIGNAL PROCESSING (DSP) techniques.

The RSP was implemented on an *Altera*⁶ *Stratix EP1S25 DSP development board*[5]. This development board features the highest speed grade (-5) *Stratix EP1S25* FPGA is a powerful development platform for DSP applications. Furthermore the board has the following integrated components:

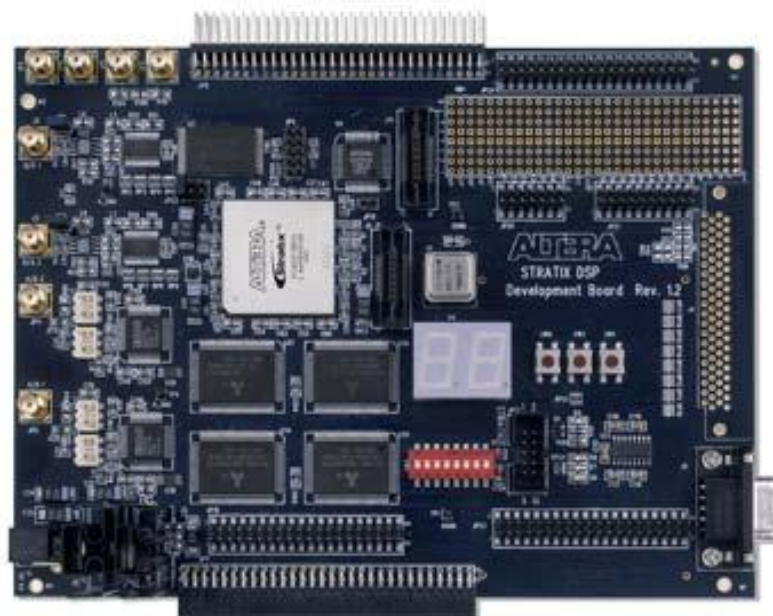
⁴<http://www.rfmd.com>

⁵<http://www.minicircuits.com>

⁶<http://www.altera.com>

- Two (2) 12-bit *Analog Devices*⁷ *AD9433* ANALOG-TO-DIGITAL CONVERTERS (ADCs) (125MSPS⁸). Only one of these ADCs was used since quadrature sampling was implemented.
- Two (2) 14-bit *Texas Instruments*⁹ *DAC940* DIGITAL-TO-ANALOG CONVERTERS (DAC) (165MSPS). The DACs were used for the debugging of sampled signals.
- Two (2) 1 Megabyte 36-bit synchronous STATIC RANDOM ACCESS MEMORY (SRAM) memory blocks. Both blocks were utilized: One for the two (2) burst maps and the second block for the time averaged filters as discussed later in this chapter. These blocks can be independently addressed.
- One (1) 32 Megabit of flash memory block.
- Various debugging components (e.g. debug headers, switches and LEDs). These were used as parameter inputs and debugging outputs to the RSP.

The *Altera Quartus II version 6.0* development suite was used to develop the RSP in VERY HIGH SPEED HARDWARE DEVELOPMENT LANGUAGE (VHDL) firmware code that defines the various RSP chains and modules. Figure (B.9) shows a picture of the development board.



Copyright © 1995-2005 Altera Corporation, 101 Innovation Drive, San Jose, California 95134, USA

Figure B.9: Picture of the *Altera Stratix EP1S25 DSP development board*[5]

⁷<http://www.analog.com>

⁸MSPS = Mega Samples Per Second

⁹<http://www.ti.com>

Appendix C

Supplementary hardware information

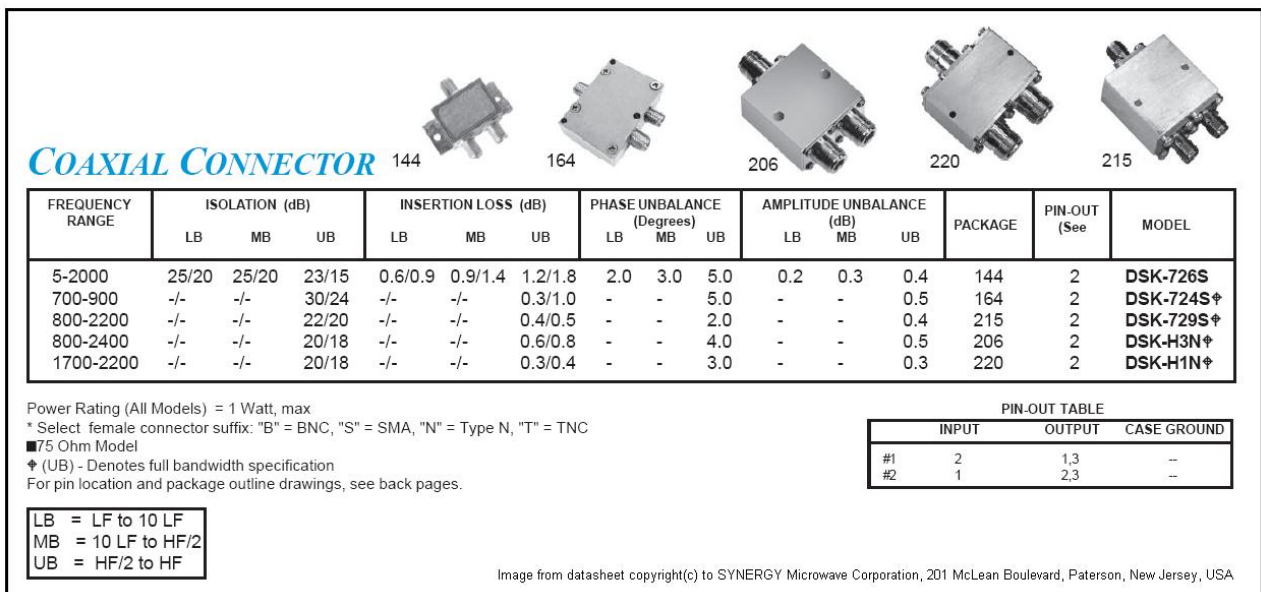


Figure C.1: Summary information of the SYNERG Microwave Corporation power dividers (DSK-726S)

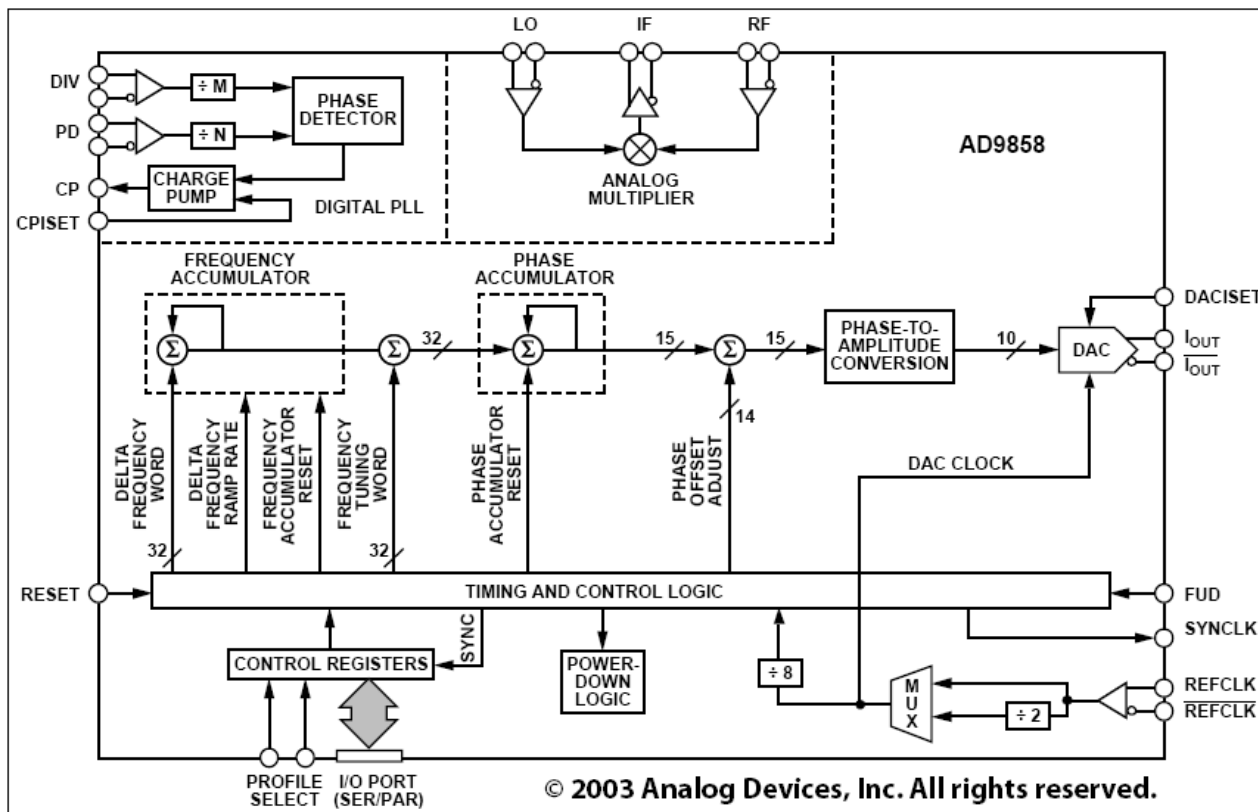


Figure C.2: Functional block diagram of the Analog Devices AD9858 DDS

MODEL NO.	FREQ. (MHz) f_l - f_u	NF (dB) Typ.	GAIN (dB)		MAXIMUM POWER (dBm)		INTERCEPT POINT (dBm) IP3 Typ.	VSWR Typ.		DC POWER		CASE STYLE Note B	CONNECTION	PRICE \$ ea. Qty. (1-9)	
			Min.	Max.	Output (1 dB Comp.)	Input (no damage)		In	Out	Volt (V)	Current (mA)				
AMP-15	5-1000	2.8	13	±0.6	±1.2	+8	+13	+22	2:1	2:1	15	29	PP120	cd	52.20
AMP-75	5-500	2.4	19	±0.4	±1.0	+12	+13	+28	2:1	2:1	15	31	PP120	cd	52.20
AMP-76	5-500	3.1	26	±0.7	±1.0	+13.5	+6	+28	2:1	2:1	15	71	PP120	cd	83.45
AMP-77	5-500	3.3	15	±0.4	±1.0	+16	+13	+32	2:1	2:1	15	56	PP120	cd	60.45
MAN-1LN**	0.5-500	3.0	28	±0.5	±1.4	+7	+15	+18	1.8:1	1.8:1	12	60	A05	cc	22.20
MAN-1HLN	10-500	3.7	10	±0.5	±0.8	+15	+15	+30	1.8:1	1.8:1	12	70	A06	cc	22.20
ZFL-500HLN	10-500	3.8	19	—	±0.4	+16	+15	+30	2:1	2:1	15	110	Y460	—	99.95
ZFL-500LN*	0.1-500	2.9	24	—	±0.5	+5	+5	+14	1.5:1	1.6:1	15	60	Y460	—	79.95
ZFL-1000LN	0.1-1000	2.9	20	—	±0.5	+3	+5	+14	1.5:1	2:1	15	60	Y460	—	89.95
NEW ZX60-1215LN	800-1000 1000-1400	0.4	14 11	— —	±1.3 ±2.5	+12.5 +12.5	+13	+26 +27.5	1.65:1 1.70:1	1.40:1 1.40:1	12	50	GA955	—	149.95
NEW ZX60-1614LN	1217-1620	0.5	11	—	±2.0	+13.5	+13	+30	1.3:1	1.3:1	12	50	GA955	—	149.95
NEW ZX60-3011	400-1000 1000-1700 1700-2400 2400-3000	1.4 1.5 1.7 1.8	12 11 9 7.5	— — — —	±.70 ±1.0 ±1.0 ±.70	+19.5 +19.5 +18.5 +18.0	+15 +15 +15 +15	+31 +31 +31 +31	1.7:1 1.7:1 1.7:1 1.7:1	1.6:1 1.6:1 1.6:1 1.6:1	12 12 12 12	120 120 120 120	GC957	—	139.95

m = mid range [$2 f_l$ to $f_u/2$]

Image from datasheet copyright(c) to Mini-Circuits, P.O. Box 350166, Brooklyn, New York, USA

Figure C.3: Summary information of the Mini-Circuits amplifier (MAN-1HLN)

Appendix D

Calculated Radar Parameters

Table (D.1) lists the calculated radar parameters calculated in the following sections.

Table D.1: Table of calculated radar parameters for the ERS

Equation	Parameter	Symbol	Value	Unit
(D.2)	Maximum unambiguous range	$R_{unambig}$	24	km
(D.3)	Receiver dead range	R_{blind}	1995	m

D.1 Range to Target

The range to the target[15] can be calculated as half the time it takes for a transmitted signal to bounce off a target and be received again, multiplied by the speed at which this signal propagates (here the speed of light):

$$R_{target} = \frac{c\tau_d}{2} [m] \quad (D.1)$$

D.2 Maximum Unambiguous Range

The maximum unambiguous range[15] for a radar system is the maximum range to a target from which a transmitted pulse can be reflected and received off before the next pulse is transmitted. For a PRF as per table (E.1), the maximum unambiguous range can be calculated from equation (D.1) to be:

$$\begin{aligned}
 R_{unambig} &= \frac{c\tau_{PRI}}{2} & (D.2) \\
 &= \frac{c}{2f_{PRF}} \\
 &= \frac{3 \times 10^8}{2(6250)} \\
 &= 24000 [m]
 \end{aligned}$$

D.3 Receiver Dead Range

The dead range refers to the period when the transmitter is active and the receiver is turned off to protect it from saturation. This period (here the transmit pulse width) translates into a range, extending from the radar outwards, when the radar cannot detect any targets. The dead range can be calculated, using (D.1), as follow:

$$\begin{aligned} R_{blind} &= \frac{c\tau_{tx}}{2} && \text{(D.3)} \\ &= \frac{(3 \times 10^8)(13,3 \times 10^{-6})}{2} \\ &= 1995 \text{ m} \end{aligned}$$

Appendix E

Mathematical Representation

In order to quantify the changes between the existing EEKHORING RADAR SYSTEM (ERS) and the upgraded version, some radar mathematical calculations are done in this chapter.

E.1 Common Radar Parameters

E.1.1 Defined Radar Parameters for the ERS

Table E.1: Table of defined radar parameters for the Eekhorng system

Component	Parameter	Symbol	Value	Unit	dB value
Transmitter	Peak transmitter power	P_t	115	W	20,607dBW
	Pulse width	τ_{tx}	13,3	μs	
	Pulse repetition interval (PRI)	τ_{PRI}	160	μs	
	Pulse repetition frequency (PRF)	f_{PRF}	6250	Hz	
	Transmit frequency	f_{tx}	1800 \pm 50	MHz	
	Wavelength	λ_{tx}	0,167	m	
	Line loss	L_{tx}			
Receiver	Noise figure	F	1,995		3
	Line loss	L_{rx}			1,50
	Available gain	G_a			50
Antenna	Sky temperature	T_{ant}	290	K	
	Horizontal beam width		9	$^\circ$	
	Rotation speed		10	rpm	
	Power gain	G_{ant}			21,5
	Loss	L_{ant}			
Target	RCS	σ	2	m^2	3,010

E.2 Signal-to-Noise Ratio Calculations

The aim is to determine the maximum detectable range [15, 16, 10, 11, 1] for a target having a $2m^2$ RADAR CROSS-SECTION (RCS) by the existing radar system and then show the SIGNAL-TO-NOISE Ratio (SNR) improvement of the upgraded system.

The SNR is given by the equation:

$$SNR = \frac{P_r}{N_o} = \frac{P_t G_t G_r \sigma \lambda^2 n_{int} L_n L_s}{(4\pi)^3 R^4 N_o} \quad (E.1)$$

These calculations are done for both the existing and upgraded systems. Table (E.2) contains the needed parameters relevant to SNR calculations.

Table E.2: Table of relevant SNR parameters

Parameter	Symbol	Value	Unit	dB value
Transmitter power	P_t	115	W	20,607dB
Transmit antenna gain	G_t			21,5
Receive antenna gain	G_r			21,5
RCS reference target	σ	2	m^2	3,010dB
Transmit wavelength	λ	$\frac{3 \times 10^8}{1,8 \times 10^9}$	m	
	λ^2			-15,563dB
Constant	$(4\pi)^3$	1984,402		32.976dB
Transmit pulse width	τ_{tx}	13,3	μs	
PRI	τ_{PRI}	160	μs	
Noise figure	F_n	1,995		3
Boltzmann's constant	k	$1,38 \times 10^{-23}$	JK^{-1}	

E.2.1 SNR of the Existing Radar System

To determine the MINIMUM DETECTABLE SIGNAL (MDS) of the ERS a further investigation of the detection chain is needed. Figure (E.1) shows the block diagram of the existing detection chain.

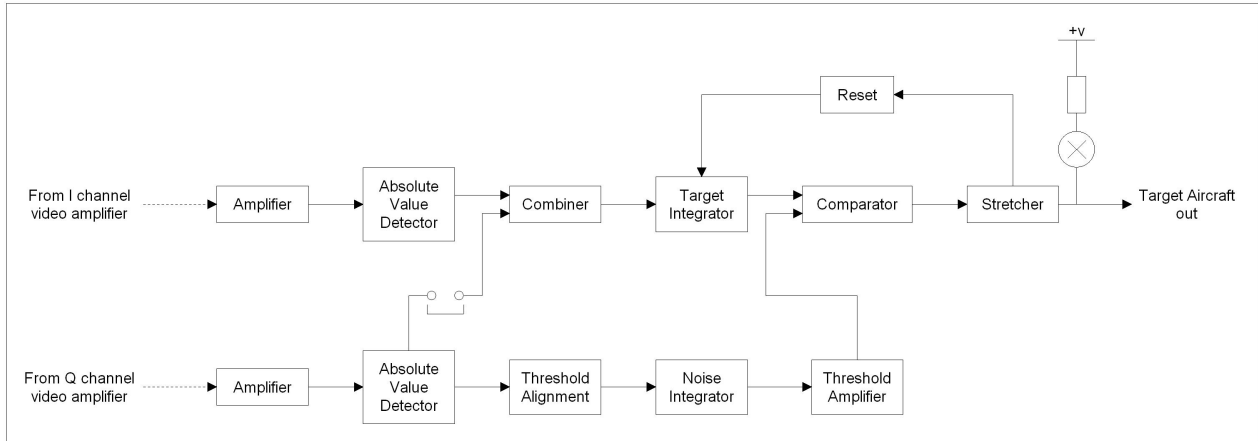


Figure E.1: ERS Aircraft Decision Circuit block diagram.

The radar utilizes sample-and-hold circuits to implement ten (10) range bins. The received signal is directly converted to base-band by means of two (2) mixers producing the IN-PHASE (I) and QUADRATURE-PHASE (Q) video channels. The filters at the output of the video amplifiers is an important aspect of the radar performance. Unfortunately the filter parameters are not specified in the documentation[1] and will thus be estimated.

Knowing that the ERS transmits a rectangular pulse of $\tau_{tx} = 13,3\mu s$ when in the 20km operational range mode, the bandwidth is calculated to be:

$$\begin{aligned}
 B &\approx \frac{1}{\tau_{tx}} \\
 &\approx \frac{1}{13,3 \times 10^{-6}} \\
 &\approx 75188 \text{ Hz}
 \end{aligned} \tag{E.2}$$

Consulting figure (L25) of the *Radar Technology Encyclopedia*[3], the optimum time-bandwidth product for the practical filter (Gaussian) is approximately 0,98. Here the filter introduces a matching loss (L_m) in the order of 0,4dB

It is thus safe to assume that the time-bandwidth product of the ERS is close to unity[10] and the filter design is almost optimal having a matching loss of 0,4dB as shown above. The matching loss must be taken into consideration during radar range calculations.

Further inspection of the ERS's detection chain shows that after the sample-and-hold circuit a so-called *PRF filter* circuit is installed, followed by the Doppler high-pass filter. Figure (E.2) shows the pass-band of these two (2) combined filters in the frequency domain with the return of a target located within the pass-band.

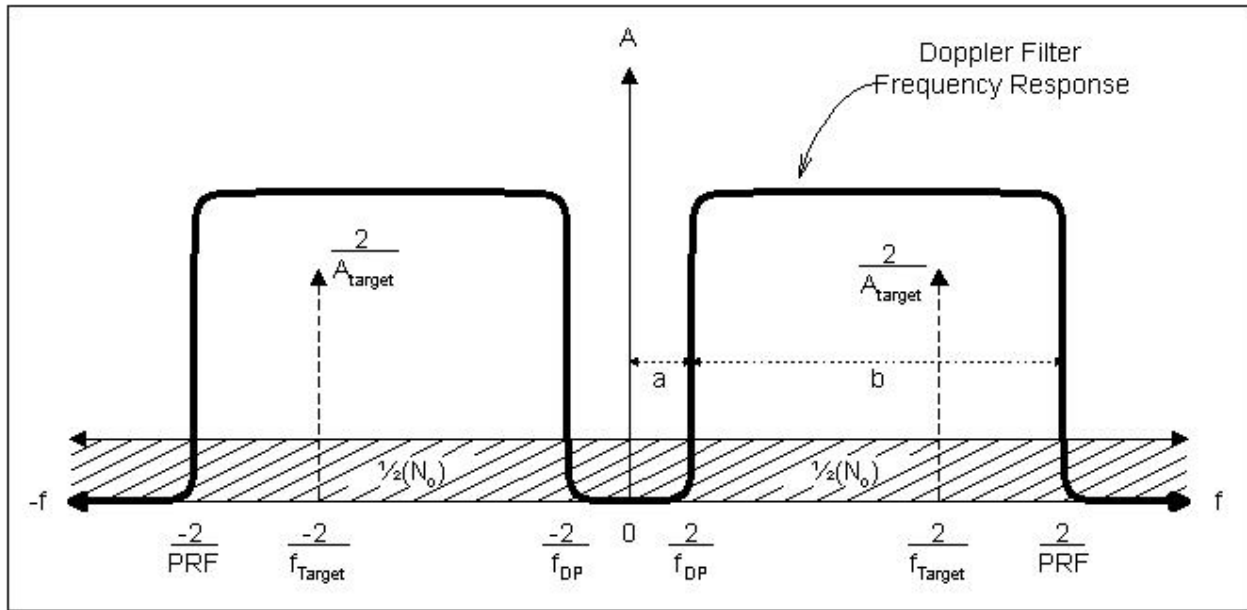


Figure E.2: Frequency domain pass-band of the Doppler filters in the SPU.

These filters introduce a small coherent gain into the system and can be calculated using the following equation:

$$\begin{aligned}
 n_{int\ coh} &= \left(\frac{a+b}{b}\right) \\
 &= \left(\frac{620+3125}{3125}\right) \\
 &= 1,198 \\
 n_{int\ coh} (dB) &= 0,786 [dB]
 \end{aligned}$$

These filters however do not represent the complete azimuthal integration. The *Aircraft Decision Circuit* [1] of figure (E.1), also shows that an *Absolute Value Detector* module is used to calculate the absolute values for both the I-channel and Q-channel data and is an approximation of $\sqrt{I^2 + Q^2}$ in the form of $(|I| + |Q|)$. This introduces a further system loss that can be calculated by averaging over all possible values of I and Q . This value will however be ignored.

The *Aircraft Decision Circuit* further shows that this approximation is fed to a *Target Integrator* module. The integrator integrates the absolute value of the return signal for a period of time after which it is cleared and a new integration period is started. Unfortunately the documentation[1] again does not specify the integration period and it must now be approximated.

The maximum integration time is the period what the target is visible in the antenna beam. This equates to the time the antenna takes to rotate by 9° :

$$\begin{aligned}
 \tau_{target\ visible} &= 9 \times \frac{60}{10 \times 360} \\
 &= 150 [ms]
 \end{aligned}$$

Assuming that the integrator can not work over the entire beam-width to detect the target at the top of the beam, it is assumed that the integration period is $\frac{1}{2}$ of the possible value, thus:

$$\begin{aligned}\tau_{int} &= \frac{1}{2}\tau_{target\ visible} \\ &= 75 [ms]\end{aligned}\tag{E.3}$$

The number of integrated pulses on target can now be calculated as:

$$\begin{aligned}n_{int} &= \tau_{int} \times f_{PRF} \\ &\simeq 468\end{aligned}$$

This value of n_{int} can now be used in the SNR calculations of the existing radar. The results are summarized in the table (E.3):

Table E.3: Summary table of ERS pulse integration parameters

Parameter	Symbol	Value	Unit	Remark
Receiver bandwidth	B	75188	Hz	
Matching Loss	L_m	0,4dB		
Coherent gain	$n_{int\ coh}$	0,78dB		
Azimuth integration type				Non-coherent
Detector Loss		0dB		assumed
Number of pulses integrated	n_{int}	468		
Integration time	τ_{int}	75	ms	

In order to calculate the SNR needed for detection, investigation into the CONSTANT FALSE ALARM RATE (CFAR) type is required. Referring to figure(E.1), target detection takes place by comparison to the integrated noise.

Here the Q-channel noise is integrated and amplified providing a detection threshold. The radar performance is determined by this circuit. The number of integrated pulses are important as this provides the so-called CFAR loss.

A false alarm rate of 1×10^{-6} is a convenient assumption here, although arbitrary. The sensitivity of this number is not very big. From *Skolnik*[16] the probability of false alarm (P_{fa}) is shown to be:

$$P_{fa} = \exp\left(\frac{-V_t^2}{2\psi_o}\right)$$

where V_t is the threshold voltage and ψ_o is the ROOT-MEAN-SQUARE (RMS) noise power (here assumed to be $\psi_o = 1$).

Decreasing from a $P_{fa} = 10^{-6}$ to a $P_{fa} = 10^{-9}$ requires an 1,7dB change in V_t . Similarly, another change of 1,2dB in V_t will decrease the probability of false alarm to $P_{fa} = 10^{-12}$. Using $P_{fa} = 10^{-6}$ will result in a false plot every 1×10^6 detections. Knowing that there are ten (10) range bins per azimuth bin and there are ($\frac{360^\circ}{9^\circ} = 40$) azimuth bins per scan, the ERS should only give a false plot every 2500 scans as calculated below:

$$\begin{aligned} n_{scans} &= \frac{1 \times 10^6}{10 \times 40} \\ &= 2500 \end{aligned}$$

This number is actually lower due to the assumption that the integration period is 75ms and not 150ms and results in the ERS giving an assumed false plot every 1250 scans. Calculated in minutes this is:

$$\begin{aligned} \tau_{false\ plot} &= 1250 \times 6 \\ &= 7500 \\ &= 125 [min] \end{aligned}$$

The integration time of the noise should be much longer than that of a target so that the presence of a target does not influence the detection process. Again, this is unknown and assumed to be ten (10) times the integration period for a target. This seems reasonable because it corresponds to a noise average taken over roughly 90° of the radar sweep coverage. This results in the use of 4687 pulses to generate the noise threshold.

Consulting the *Radar Technology Encyclopedia*[4], it is found that the effective number of reference samples, m_e can be calculated as follows:

$$\begin{aligned} m_e &= \frac{m + k}{1 + k} \\ &= \frac{4687 + 0,09}{1 + 0,09} \\ &\approx 4300 \end{aligned}$$

where k is the constant for a linear envelope detector and m the number of pulses integrated to determine the noise threshold. If the ERS works with a probability of false alarm of $P_{fs} = 10^{-9}$, the CFAR loss (L_{CFAR}) is very small and negligible.

In order to calculate the radar detection range the required SNR for detection is needed. It is traditional to specify a typical target as a *Swerling* 1 to 4 model. These *Swerling* models refer to the fluctuating characteristics of a target. *Swerling* models 1 and 2 refer to a target having a complex-body, shape such as aircraft. *Swerling* models 3 and 4 on the other hand refer to simple-body targets such as missiles. Models 1 and 3 refer to slow fluctuating targets while models 2 and 4 represents fast fluctuating targets.

For comparison purposes and convenience sake, the complexity of target modeling will not be studied here. A non-fluctuating target will be assumed. This type of target is referred to by *Barton* as a *Swerling 0* model while other authors sometimes call it a *Swerling 5* model. Using *Matlab* functions from *Matlab Simulations for Radar Systems Design*[12], the following figure (E.3) was generated indicating the probability of detection for various number of non-coherently integrated pulses.

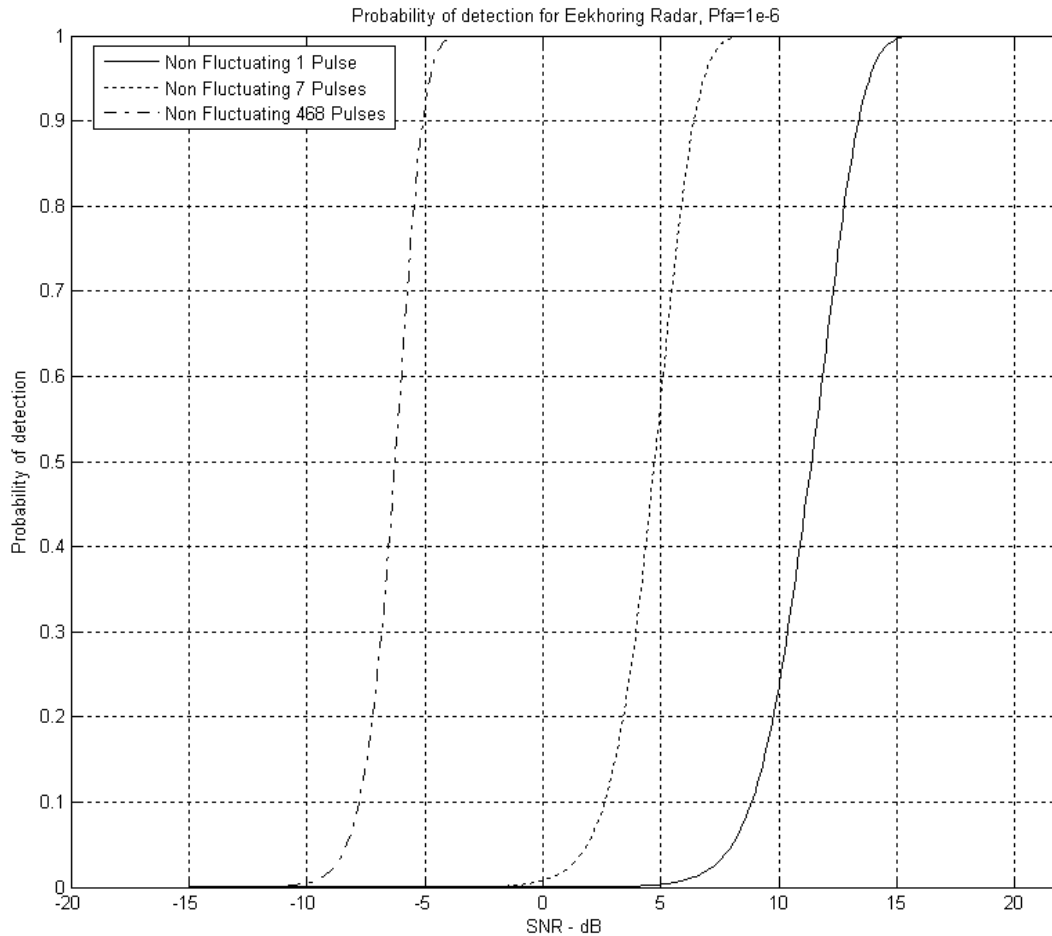


Figure E.3: Graph of the probability of detection for the ERS.

From this figure it shown that to achieve a probability of detection (P_d) of 80% using a single pulse, the required SNR is 13dB Non-coherently integrating 486 pulses, the SNR required is now -6dB to provide a P_d of 80%. This last value of P_d is used to calculate the detection range of the ERS. The reference to 7 non-fluctuating pulses are made to reflect the 64 PRIs used per burst.

The MDS of the ERS can now be calculated using the following table:

Table E.4: Table of the MDS parameters

Parameter	Symbol	Value	Unit	Remark
Thermal noise	kT_o	-174	dB	
System bandwidth	B	48,76	dB	
Noise figure	F	3		
Required SNR for detection	SNR	-6	dB	
CFAR Loss	L_{CFAR}	0	dB	
Matching Loss	L_m	0,4	dB	
Coherent integration gain	n_{int}	0,78	dB	

$$\begin{aligned}
 S_{MDS} &= kT_0 + B + F + SNR + L_{CFAR} + L_m - n_{int} \\
 &= -174 + 48,76 + 3 - 6 + 0 + 0,4 - 0,78 \\
 &= -128,62 [dBm]
 \end{aligned}$$

To determine the maximum detectable range[15] for a target having a RCS of $2m^2$ the following equation is used:

$$R_{max} = \left[\frac{P_t G_t G_r \lambda^2 \sigma}{(4\pi)^3 S_{MDS}} \right]^{\frac{1}{4}} \quad (E.4)$$

Noting that ($S_{MDS(dB)} = S_{MDS(dBm)} + 30$) and converting (E.4) to a decibel equation gives:

$$\begin{aligned}
 10 \log(R_{max}) &= \frac{1}{4} \left[P_t(dB) + G_t(db) + G_r(dB) + \lambda_{(dB)}^2 + \sigma_{(dB)} - (4\pi_{(dB)})^3 - S_{MDS(dB)} \right] \\
 R_{max(dB)} &= \frac{1}{4} [20,607 + 21,5 + 21,5 + (-15,563) + 3,010 - 32,976 - (-158,62)] \\
 &= 44,174
 \end{aligned}$$

Converting back gives:

$$\begin{aligned}
 R_{max} &= \left[10^{\frac{R_{max(dB)}}{10}} \right] \\
 R_{max} &= 26146 [m] \\
 &\sim 26 [km]
 \end{aligned}$$

This maximum detection range is for comparative calculations of the upgraded system only. It does not include system losses such as cable loss, receiver line loss, etc.¹

¹These losses were omitted for clarity reasons as these are not affected by the discussed alterations.

E.2.2 SNR of the Upgraded Radar System

The upgraded system introduces the following aspects to the detection process:

1. The receiver bandwidth is increased from 75kHz to 2,53125MHz.
2. Pulse compression with bi-phase coding is implemented.
3. Coherent integration is introduced by using Doppler filter banks over a portion of the available on target visibility time. Non-coherent integration is used for the rest.

Of the above aspects, only the coherent integration changes the detection performance of the system. Pulse compression is also not 100% efficient, so some loss can be expected.

The coherent integration takes place over 64 PRIs. The difference will be the integration loss of the non-coherent integrator (which is 7 from figure(E.3)) in the existing system. *Di Franco and Robin* (as in *Mahazda*[12]) pointed out that the integration loss can be approximated using the following equation:

$$\begin{aligned}
 L_{int} (dB) &= 10\log(\sqrt{n_{int}}) - 5,5 \\
 &= 9,0309 - 5,5 \\
 &= 3,5309 [dB]
 \end{aligned}$$

where $n_{int} = 64$ is the number of pulses integrated.

The pulse compression gain is calculated as the time-bandwidth product. Using software provided by RRS² to generate a suitable pulse compression code, figure (E.4) was generated showing two compressed pulses. One being ideal and the other filtered to approximated the compression loss (L_{comp}) found to be 0,8dB.

²Refer to chapter (5) for a discussion on Pulse Compression.

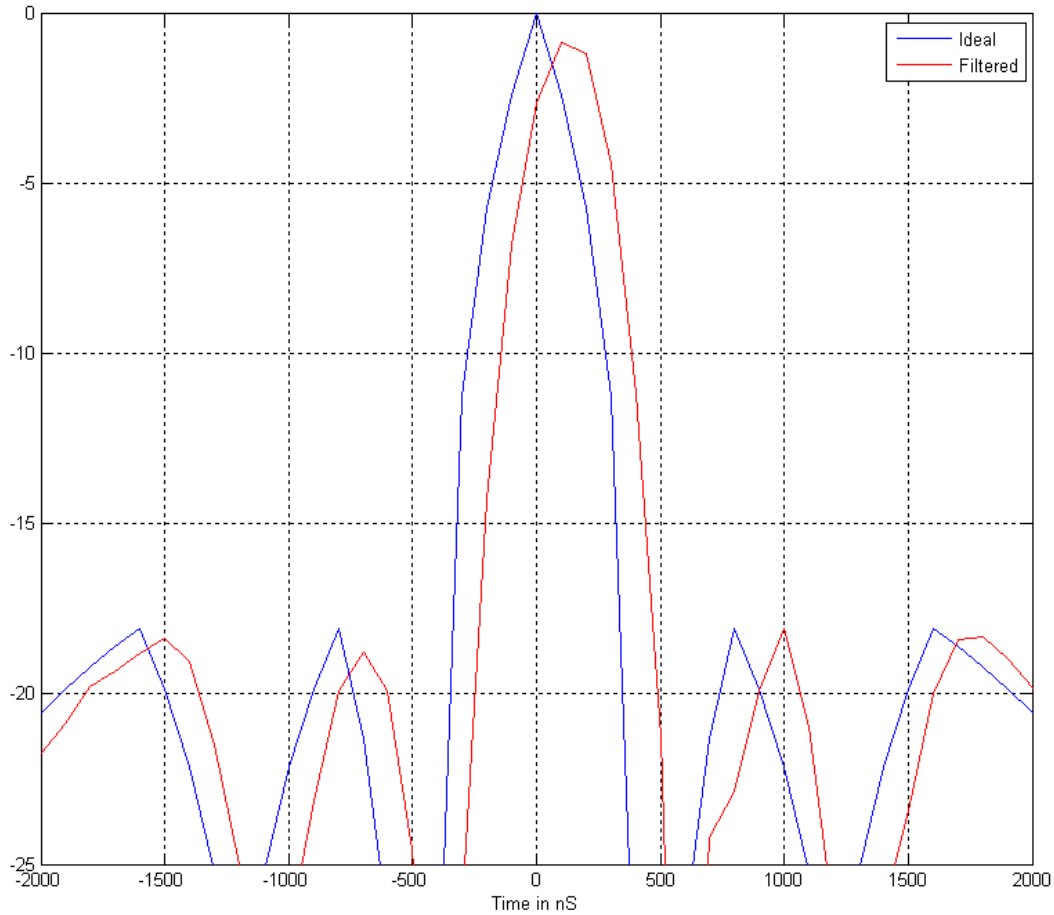


Figure E.4: Effect of the non-ideal transmit and receiver chains on the compressed pulse.

Referring to figure (E.3) showing the probability of detection for various pulses, the required SNR for detection is calculated to be the SNR to detect seven (7) non-coherent pulses minus the coherent integration gain. This results in a required SNR of -12dB.

Table E.5: Table of the MDS parameters for the upgraded system

Parameter	Symbol	Value	Unit	Remark
Thermal noise	kT_o	-174	dB	
System bandwidth	B	64,033	dB	
Noise figure	F	3		
Required SNR for detection	SNR	-12	dB	
CFAR Loss	L_{CFAR}	0	dB	
Filter matching Loss	L_m	0,4	dB	
Time-bandwidth product	TB	15,21	dB	
Compression Loss	L_{comp}	0,8	dB	
Coherent integration gain	n_{int}	18,06	dB	Included in SNR
Non-coherent integration gain	$n_{non\ coh}$	7	dB	Included in SNR

The MDS for the upgrade system is now calculated to be:

$$\begin{aligned}
 S_{MDS} &= kT_0 + B + F + SNR + L_{CFAR} + L_m - TB + L_{comp} \\
 &= -174 + 64,033 + 3 - 12 + 0 + 0,4 - 15,21 + 0,8 \\
 &= -132,977 [dBm]
 \end{aligned}$$

Thus we can now calculate the maximum detection range using equation (E.4) to be:

$$\begin{aligned}
 10\log(R_{max}) &= \frac{1}{4} \left[P_t(dB) + G_t(db) + G_r(dB) + \lambda_{(dB)}^2 + \sigma_{(dB)} - (4\pi(dB))^3 - S_{MDS(dB)} \right] \\
 R_{max(dB)} &= \frac{1}{4} [20,607 + 21,5 + 21,5 + (-15,563) + 3,010 - 32,976 - (-162,977)] \\
 &= 45,263
 \end{aligned}$$

Converting back gives:

$$\begin{aligned}
 R_{max} &= \left[10^{\frac{R_{max}(dB)}{10}} \right] \\
 R_{max} &= 33599 [m] \\
 &\sim 34 [km]
 \end{aligned}$$

Comparatively, the upgraded system has a $\sim 30\%$ performance increase in the maximum detection range. Again, this maximum detection range is for comparative calculations only. It does not include system losses such as cable loss, receiver line loss, etc.³

³These losses were omitted for clarity reasons.

Appendix F

Firmware Code Listings

F.1 Clock Edge Detector Module

```

1  -- Clock Edge Detector Module
2  -- 20 Aug 2006
3  -- JSW Rust
4  -- University of Stellenbosch
5  -----
6  -- This code must sense just the rising edge of the Sync pulse and produce a very fast replica of this signal.
7  -- This was needed because the counters were in reset state for the entire time that the sync pulse was high (13.3us).
8  -- This resulted that the DDS-LUT module did not work because there was no clock available
9
10 -- NB: The same module will be used for clearing the pri_counter when the North pulse comes along.
11 -- Comments in the code are for the sync pulse implementation.
12
13 Library ieee;
14 use ieee.std_logic_1164.all;
15
16
17 ENTITY CEDM IS
18   PORT
19     (
20       Iclk           : IN STD_LOGIC;
21       Isignal        : IN STD_LOGIC;
22
23       Opulse         : OUT STD_LOGIC
24     );
25 END CEDM;
26
27 ARCHITECTURE CEDM_arch OF CEDM IS
28
29   TYPE CEDM_type IS (trigger,sleep);
30   SIGNAL CEDM_state : CEDM_type;
31   BEGIN
32
33   PROCESS(Iclk)
34   BEGIN
35
36
37     IF (Iclk'EVENT and Iclk = '1') THEN
38
39       CASE CEDM_state IS
40
41         WHEN trigger =>
42           IF (Isignal = '1') THEN -- sit and wait until the sync pulse goes high
43             Opulse <= '1'; -- when the sync pulse is sampled (with the 81MHz clock) to be high, output that detection
44             CEDM_state <= sleep;
45           END IF;
46
47         WHEN sleep =>

```

```
48      Opulse <= '0'; -- On the next high speed clock, remove the output
49      IF (Isignal = '0') THEN -- now sleep until the sync pulse goes low, then change the state
50          CEDM_state <= trigger;
51      END IF;
52  END CASE;
53 END IF;
54 END PROCESS;
55 END CEDM_arch;
```

F.2 Clock Synchronization Module

```
1  -- Clock Sync Module
2  -- 21 Aug 2006
3  -- JSW Rust
4  -- University of Stellenbosch
5  --
6  -- Based on the CEDM.
7  -----
8
9  Library ieee;
10 use ieee.std_logic_1164.all;
11
12 ENTITY CSM IS
13 PORT
14 (PORT
15 (PORT
16   Iclk           : IN STD_LOGIC;
17   Isignal        : IN STD_LOGIC;
18
19   Opulse         : OUT STD_LOGIC
20 );
21 END CSM;
22
23 ARCHITECTURE CSM_arch OF CSM IS
24
25 TYPE CSM_type IS (high,low);
26 SIGNAL CSM_state : CSM_type;
27 BEGIN
28
29 PROCESS(Iclk)
30 BEGIN
31
32
33 IF (Iclk'EVENT and Iclk = '1') THEN
34
35   CASE CSM_state IS
36
37     WHEN high =>
38       IF (Isignal = '1') THEN
39         Opulse <= '1';
40         CSM_state <= high;
41       ELSE
42         CSM_state <= low;
43         Opulse <= '0';
44       END IF;
45
46     WHEN low =>
47       IF (Isignal = '0') THEN
48         CSM_state <= low;
49         Opulse <= '0';
50       ELSE
51         CSM_state <= high;
52         Opulse <= '1';
53       END IF;
54     END CASE;
55   END IF;
56 END PROCESS;
57 END CSM_arch;
```

F.3 DDS Setup Controller

```

1  -- DDS Controller
2  --
3  -- 2005.08.08
4  -- JSW Rust
5  -- University of Stellenbosch
6  -- wr@capesky.net
7  --
8  -- DDS controller rewrite with the help of Mike Movius from RRS - 3 Aug 2006
9  -- DDS Profile Setup:
10 -----
11 -- Frequency Tuning Word = FTW
12 -- Phase Offset Word = POW
13 -- FO = Frequency Output
14 -- N = 32
15 -- TX DDS FTW: (99.20MHz)
16 -- FTW = (FO * 2^N) / SYSCLK
17 -- FTW = (99.20exp6 * 2^32) / 892.8exp6
18 -- FTW = 477218588,44 = 477218588
19 --      = 0001 1100 0111 0001 1100 0111 0001 1100b
20 -- RX DDS FTW: (103.88MHz)
21 -- FTW = 499780637d
22 --      = 0001 1101 1100 1010 0000 1100 0001 1101b
23 --
24 -- TX DDS:
25 -----
26 -- Profile 1 : 0 MHz
27 -- Setup: FTW = 0; POW = 0d
28 --
29 -- Profile 2 : 0 MHz
30 -- Setup: FTW = 0; POW = 0d
31 --
32 -- Profile 3 : 99.20 MHz, 0deg
33 -- Setup: FTW = 477218588d; POW = 0d
34 --
35 -- Profile 4 : 99.20 MHz, 10deg
36 -- Setup: FTW = 477218588d; POW = 1000h = 4096d = 90deg
37 -- Setup: FTW = 477218588d; POW = 01C7h = 455d = 10deg (used for debugging later)
38 --
39 -- RX DDS:
40 -----
41 -- Profile 1 : 0 MHz
42 -- Setup: FTW = 0; POW = 0d
43 --
44 -- Profile 2 : 0 MHz
45 -- Setup: FTW = 0d; POW = 0d
46 --
47 -- Profile 3 : 103.089 MHz
48 -- Setup: FTW = 495975396d; POW = 0d
49 --
50 -- Profile 4 : 103.089 MHz, 0deg
51 -- Setup: FTW = 495975396d; POW = 0d (0deg)
52 --
53 LIBRARY ieee;
54 use IEEE.STD_LOGIC_SIGNED.all;
55 USE ieee.std_logic_1164.all;
56
57
58
59 ENTITY dds_controller_block IS
60   PORT
61   (
62     reset_n      : IN STD_LOGIC;
63     low_clk      : IN STD_LOGIC;
64     high_clk     : IN STD_LOGIC;
65
66
67     txdds_addr   : OUT STD_LOGIC_VECTOR(5 downto 0);
68     txdds_data   : OUT STD_LOGIC_VECTOR(7 downto 0);
69     txdds_ps     : OUT STD_LOGIC_VECTOR(1 downto 0);
70     txdds_fud    : OUT STD_LOGIC;

```

```

71 txdds_reset : OUT STD_LOGIC;
72 txdds_spmode : OUT STD_LOGIC;
73 txdds_readb : OUT STD_LOGIC;
74 txdds_writeb : OUT STD_LOGIC;
75
76 rxdds_addr : OUT STD_LOGIC_VECTOR(5 downto 0);
77 rxdds_data : OUT STD_LOGIC_VECTOR(7 downto 0);
78 rxdds_ps : OUT STD_LOGIC_VECTOR(1 downto 0);
79 rxdds_fud : OUT STD_LOGIC;
80 rxdds_reset : OUT STD_LOGIC;
81 rxdds_spmode : OUT STD_LOGIC;
82 rxdds_readb : OUT STD_LOGIC;
83 rxdds_writeb : OUT STD_LOGIC;
84
85 init_done : OUT STD_LOGIC
86 );
87
88 END dds_controller_block;
89
90
91 ARCHITECTURE dds_controller OF dds_controller_block IS
92
93
94 SIGNAL dds_reset : STD_LOGIC;
95 SIGNAL dds_address : STD_LOGIC_VECTOR(5 downto 0);
96 SIGNAL dds_writeb : STD_LOGIC;
97 SIGNAL dds_readb : STD_LOGIC;
98 SIGNAL dds_spmode : STD_LOGIC;
99 SIGNAL dds_fud : STD_LOGIC;
100 SIGNAL statecount : integer range 0 to 63;
101
102 BEGIN
103
104 -- Commonoterial
105
106 -- address both fuds at the same time
107 txdds_fud <= dds_fud;
108 rxdds_fud <= dds_fud;
109
110 -- address both address busses at the same time
111 txdds_addr <= dds_address;
112 rxdds_addr <= dds_address;
113
114 -- address both resets at the same time
115 txdds_reset <= dds_reset;
116 rxdds_reset <= dds_reset;
117
118 -- address both spmodes at the same time
119 txdds_spmode <= dds_spmode;
120 rxdds_spmode <= dds_spmode;
121
122 -- address both reads at the same time
123 txdds_readb <= dds_readb;
124 rxdds_readb <= dds_readb;
125
126 PROCESS (high_clk, low_clk, reset_n)
127
128 BEGIN
129 if (reset_n = '0') then
130 statecount <= 0;
131 dds_reset <= '1';
132 init_done <= '0';
133 elsif (high_clk'event and high_clk='0') then
134 if (low_clk = '1') then
135 if (statecount < 39) then
136 statecount <= statecount + 1;
137 init_done <= '0';
138 else
139 statecount <= 40;
140 init_done <= '1';
141 end if;
142
143 -- permanent setups

```



```

144     txdds_ps    <= "10"; -- setup TX DDS profile 2 (will later be done externally)
145 rxdds_ps     <= "10"; -- setup RX DDS profile 2 (will later be done externally)
146 dds_reset    <= '0'; -- no reset
147 dds_spmode   <= '1'; -- enable parallel mode
148 dds_readb    <= '1'; -- never allow to read
149
150
151 case statecount is
152 when 0 => dds_address <= "00" & x"0";
153     txdds_data <= x"58"; -- div 2 = off, mixer = off, PD = off
154     rxdds_data <= x"58"; -- div 2 = off, mixer = off, PD = off
155     dds_writeb <= '0'; -- enable write (active low)
156     dds_fud    <= '0'; -- now freq update yet
157
158 when 1 => dds_address <= "00" & x"0";
159     txdds_data <= x"58"; -- div 2 = off, mixer = off, PD = off
160     rxdds_data <= x"58"; -- div 2 = off, mixer = off, PD = off
161     dds_writeb <= '0'; -- enable write (active low)
162     dds_fud    <= '0'; -- now freq update yet
163
164
165 when 2 => dds_address <= "00" & x"1";
166     txdds_data <= x"00";
167     rxdds_data <= x"00";
168
169 when 3 => dds_address <= "00" & x"2";
170     txdds_data <= x"00";
171     rxdds_data <= x"00";
172
173 when 4 => dds_address <= "00" & x"3";
174     txdds_data <= x"00";
175     rxdds_data <= x"00";
176
177 when 5 => dds_address <= "00" & x"4";
178     txdds_data <= x"00"; -- delta frequency tuning word <7..0>
179     rxdds_data <= x"00";
180
181 when 6 => dds_address <= "00" & x"5";
182     txdds_data <= x"00"; -- delta frequency tuning word <15..8>
183     rxdds_data <= x"00";
184
185 when 7 => dds_address <= "00" & x"6";
186     txdds_data <= x"00"; -- delta frequency tuning word <23..16>
187     rxdds_data <= x"00";
188
189 when 8 => dds_address <= "00" & x"7";
190     txdds_data <= x"00"; -- delta frequency tuning word <31..24>
191     rxdds_data <= x"00";
192
193 when 9 => dds_address <= "00" & x"8";
194     txdds_data <= x"00"; -- delta frequency tuning rate <7..0>
195     rxdds_data <= x"00";
196
197 when 10 => dds_address <= "00" & x"9";
198     txdds_data <= x"00"; -- delta frequency tuning rate <15..8>
199     rxdds_data <= x"00";
200
201 when 11 => dds_address <= "00" & x"A";
202     txdds_data <= x"58"; -- profile 0 frequency word <7..0>
203     rxdds_data <= x"00";
204
205 when 12 => dds_address <= "00" & x"B";
206     txdds_data <= x"8e"; -- profile 0 frequency word <15..8>
207     rxdds_data <= x"00";
208
209 when 13 => dds_address <= "00" & x"C";
210     txdds_data <= x"e3"; -- profile 0 frequency word <23..16>
211     rxdds_data <= x"00";
212
213 when 14 => dds_address <= "00" & x"D";
214     txdds_data <= x"38"; -- profile 0 frequency word <31..24>
215     rxdds_data <= x"00";
216

```

```

217     when 15 => dds_address <= "00" & x"E";
218         txdds_data    <= x"00";    -- phase offset word 0 <7..0>
219         rxdds_data    <= x"00";
220
221     when 16 => dds_address <= "00" & x"F";
222         txdds_data    <= x"00";    -- phase offset word 0 <15..8>
223         rxdds_data    <= x"00";
224
225     when 17 => dds_address <= "01" & x"0";
226         txdds_data    <= x"58";    -- profile 1 frequency word <7..0>
227         rxdds_data    <= x"00";
228
229     when 18 => dds_address <= "01" & x"1";
230         txdds_data    <= x"8e";    -- profile 1 frequency word <15..8>
231         rxdds_data    <= x"00";
232
233     when 19 => dds_address <= "01" & x"2";
234         txdds_data    <= x"e3";    -- profile 1 frequency word <23..16>
235         rxdds_data    <= x"00";
236
237     when 20 => dds_address <= "01" & x"3";
238         txdds_data    <= x"38";    -- profile 1 frequency word <31..24>
239         rxdds_data    <= x"00";
240
241     when 21 => dds_address <= "01" & x"4";
242         txdds_data    <= x"00";    -- phase offset word 1 <7..0>
243         rxdds_data    <= x"00";
244
245     when 22 => dds_address <= "01" & x"5";
246         txdds_data    <= x"00";    -- phase offset word 1 <15..8>
247         rxdds_data    <= x"00";
248
249     when 23 => dds_address <= "01" & x"6";
250         txdds_data    <= x"2c";    -- must be 1C -- profile 2 frequency word <7..0>
251         --rxdds_data    <= x"E4";
252         rxdds_data    <= x"43";    -- must be 44
253
254     when 24 => dds_address <= "01" & x"7";
255         txdds_data    <= x"C7";    -- profile 2 frequency word <15..8>
256         --rxdds_data    <= x"FB";
257         rxdds_data    <= x"D7";
258
259     when 25 => dds_address <= "01" & x"8";
260         txdds_data    <= x"71";    -- profile 2 frequency word <23..16>
261         --rxdds_data    <= x"8F";
262         rxdds_data    <= x"93";
263
264     when 26 => dds_address <= "01" & x"9";
265         txdds_data    <= x"1C";    -- profile 2 frequency word <31..24>
266         --rxdds_data    <= x"1D";
267         rxdds_data    <= x"1D";
268
269     when 27 => dds_address <= "01" & x"A";
270         txdds_data    <= x"00";    -- phase offset word 2 <7..0>
271         rxdds_data    <= x"00";
272
273     when 28 => dds_address <= "01" & x"B";
274         txdds_data    <= x"00";    -- phase offset word 2 <15..8>
275         rxdds_data    <= x"00";
276
277     when 29 => dds_address <= "01" & x"C";
278         txdds_data    <= x"2c";    -- must be 1C -- profile 3 frequency word <7..0>
279         rxdds_data    <= x"1C";
280
281     when 30 => dds_address <= "01" & x"D";
282         txdds_data    <= x"C7";    -- profile 3 frequency word <15..8>
283         rxdds_data    <= x"C7";
284
285     when 31 => dds_address <= "01" & x"E";
286         txdds_data    <= x"71";    -- profile 3 frequency word <23..16>
287         rxdds_data    <= x"71";
288
289     when 32 => dds_address <= "01" & x"F";

```

```
290         txdds_data    <= x"1C";    -- profile 3 frequency word <31..24>
291         rxdds_data    <= x"1C";
292
293     when 33 => dds_address <= "10" & x"0";
294         txdds_data    <= x"c7";    -- phase offset word 3 <7..0>
295         rxdds_data    <= x"c7";
296
297     when 34 => dds_address <= "10" & x"1";
298         txdds_data    <= x"01";    -- phase offset word 3 <15..8>
299         rxdds_data    <= x"01";
300
301     when 35 => dds_writeb    <= '1';    -- complete the write (keep writeb high)
302
303     when 36 => dds_fud    <= '1';    -- activate the FUD
304
305     when 37 => dds_fud    <= '0';    -- deactivate the FUD
306
307     when others => --do nothing
308
309     end case;
310
311 end if;
312
313 end if;
314
315     txdds_writeb <= not(not(dds_writeb) and not(low_clk));
316     rxdds_writeb <= not(not(dds_writeb) and not(low_clk));
317
318 END PROCESS ;
319
320 END dds_controller;
```

F.4 DDS LUT Controller

```

1  -- DDS Look-up Table (LUT)
2  --
3  -- 2006.01.31
4  -- JSW Rust
5  -- University of Stellenbosch
6  -- wr@capestry.net
7  --
8  -- Rewrite: 6 June 06
9  --
10 --
11 -- Discription:
12 -----
13 --
14 -- The DDS LUT must switch the phase of the TX DDS when triggered by the system Sync-pulse (the TX pulse
15 -- from the Eekhorng system). The phase must be modulated with a 0 or 90deg shift which is predetermined.
16 -- The LUT must be stored in a ROM mif-file.
17
18 library ieee;
19 use ieee.std_logic_1164.all;
20 use ieee.std_logic_arith.all;
21 use ieee.std_logic_unsigned.all;
22
23 ENTITY dds_LUT_block IS
24 GENERIC( Addr_wid   : integer := 9;
25          tx_stop_bin : integer := 32; -- only enable the tx signal when modulating
26          rangebins   : integer := 404 -- 404 * 1/2.53125e6 = 160us (the PRI)
27          );
28 PORT
29 (
30   Imod_clock   : IN STD_LOGIC; -- modulation clock of 2.53125 MHz
31   Ireset       : IN STD_LOGIC; -- global reset input, low reset
32   Isync        : IN STD_LOGIC; -- radar transmit signal, the sync-pulse
33
34   Otxdds_enable : OUT STD_LOGIC; -- used to turn off the transmit signal when the transmit pulse is done.
35   OLUt_address  : OUT STD_LOGIC_VECTOR(Addr_wid-1 downto 0) -- DDS phase mod pattern output signal
36
37 );
38 END dds_LUT_block;
39
40 ARCHITECTURE dds_LUT OF dds_LUT_block IS
41
42 -----
43 -- State Machine Types:
44 -----
45
46 TYPE Tdds_LUT_type IS (idle, modulate);
47
48 -----
49 -- State Machine Signals:
50 -----
51
52 SIGNAL Sdds_LUT_state : Tdds_LUT_type;
53 SIGNAL Srange_cnt    : STD_LOGIC_VECTOR(Addr_wid-1 downto 0); -- current rangebin indicator
54 -- The amount of range counts per PRI are:
55 -- (1/PRI) / (1/mod_clock) = (1/6.25KHz) / (1/2.53125MHz) = 405
56
57
58 BEGIN
59
60 PROCESS (Ireset, Imod_clock)
61
62 BEGIN
63   IF (Ireset = '0') THEN
64     OLUt_address <= (OTHERS => '0'); -- set to address 0.
65     Srange_cnt <= (OTHERS => '0');
66     Otxdds_enable <= '0'; -- disable the transmit dds (by going to a profile which turns off the output signal)
67     Sdds_LUT_state <= idle;
68
69   ELSIF (Imod_clock'EVENT and Imod_clock = '1') THEN
70     CASE Sdds_LUT_state IS

```

```
71
72 WHEN idle =>
73   IF Isync = '1' THEN
74     Otxdds_enable <= '1'; -- enable the tx dds
75     OLUt_address <= (OTHERS => '0'); -- set to address 0.
76     Srange_cnt <= (OTHERS => '0');
77     Sdds_LUT_state <= modulate;
78   ELSE
79     Srange_cnt <= (OTHERS => '0');
80     Otxdds_enable <= '0'; -- disable the tx dds
81     Sdds_LUT_state <= idle;
82   END IF;
83
84 WHEN modulate =>
85   IF (Srange_cnt < rangebins - 1) THEN -- make sure to stop a while
86     -- before the next sync comes
87     -- along (thus the "- tx_stop_bin");
88   IF (Srange_cnt < tx_stop_bin - 1) THEN -- IF still before renage bin 32
89     Otxdds_enable <= '1'; -- enable the tx dds
90   ELSE
91     Otxdds_enable <= '0'; -- disable the tx dds
92   END IF;
93   Srange_cnt <= Srange_cnt + 1; -- inc the range counter
94   OLUt_address <= Srange_cnt + 1; -- output the next ROM address
95   Sdds_LUT_state <= modulate;
96   ELSE
97     Srange_cnt <= (OTHERS => '0'); -- reset the range counter
98     Sdds_LUT_state <= idle; -- go sit and wait for the next sync
99   END IF;
100 END CASE;
101 END IF;
102 END PROCESS;
103 END dds_LUT;
```

F.5 Down Converter Module

```

1  -- Downconverter Module
2  -----
3  -- JSW Rust
4  -- University of Stellenbosch
5  -- wr@capestry.net
6  -- 20 March 2006
7  --
8  -- This module splits the sampled signals into two channels of 20,25MHz each.
9  -- The sign of every 3rd and 4th sample is inverted to correct due the IF sampling.
10 -----
11
12 Library ieee;
13 use ieee.std_logic_1164.all;
14 use ieee.std_logic_arith.all;
15 use ieee.std_logic_unsigned.all;
16
17 ENTITY DCM_block IS
18   PORT
19     (
20       Isample_clk      : IN STD_LOGIC;
21       Ireset_n        : IN STD_LOGIC;
22       Ia2ddata        : IN signed(11 downto 0);
23
24       Iout             : OUT signed(11 downto 0);
25       Qout            : OUT signed(11 downto 0);
26     );
27 END DCM_block;
28
29
30
31 ARCHITECTURE DCM OF DCM_block IS
32
33   TYPE splittype      IS (A, B, C, D);
34   SIGNAL splitstate   : splittype;
35
36
37
38 BEGIN
39
40   PROCESS (Isample_clk,Ireset_n)
41
42     VARIABLE I_temp : signed(11 downto 0);
43
44     BEGIN
45
46       IF (Ireset_n = '0') THEN
47         splitstate <= A;
48         Iout <= "000000000000";
49         Qout <= "000000000000";
50
51       ELSIF (Isample_clk'EVENT and Isample_clk = '1') THEN
52
53         CASE splitstate IS
54           WHEN A =>
55             I_temp := Ia2ddata;
56             splitstate <= B;
57
58           WHEN B =>
59             Iout <= I_temp;
60             Qout <= Ia2ddata;
61             splitstate <= C;
62
63           WHEN C =>
64             I_temp := -Ia2ddata;
65             splitstate <= D;
66
67           WHEN D =>
68             Iout <= I_temp;
69             Qout <= -Ia2ddata;
70             splitstate <= A;

```

```
71 |  
72 |     END CASE;  
73 |     END IF;  
74 | END PROCESS;  
75 | END DCM;
```

F.6 Data Packer Module

```

1  --- Data Packer Module
2  -----
3  --- JSW Rust
4  --- University of Stellenbosch
5  --- wr@capestry.net
6  --- 20 March 2006
7  ---
8  ---
9  Library ieee;
10 use ieee.std_logic_1164.all;
11 use ieee.std_logic_arith.all;
12 use ieee.std_logic_unsigned.all;
13
14 ENTITY Data_Packer_block IS
15
16   GENERIC(pulse_stopbin : integer := 64;
17           rx_stopbin    : integer := 809;
18           zeropad_stopbin : integer := 1024
19           );
20
21   PORT
22   (
23     Iclk      : IN STD_LOGIC;
24     Ireset    : IN STD_LOGIC;
25     Isync     : IN STD_LOGIC;
26     Idata     : IN STD_LOGIC_VECTOR(23 downto 0);
27     Ipri_count : IN STD_LOGIC_VECTOR(15 downto 0);
28
29     Owr_clk   : OUT STD_LOGIC;
30     Owr_req   : OUT STD_LOGIC;
31     Odata_rdy : OUT STD_LOGIC;
32     Osig      : OUT STD_LOGIC_VECTOR(1 downto 0);
33     Odata     : OUT STD_LOGIC_VECTOR(31 downto 0);
34     Ostate    : OUT STD_LOGIC_VECTOR(3 downto 0);
35     Obincount : OUT STD_LOGIC_VECTOR(9 downto 0)
36   );
37 END Data_Packer_block;
38
39 ARCHITECTURE Data_Packer OF Data_Packer_block IS
40
41   TYPE Tpacker_type IS (idle, clk_header1, header2, clk_header2, pulse_blank,
42     clk_pulse_blank, rx, clk_rx);
43
44   SIGNAL Spacker_state : Tpacker_type;
45
46   BEGIN
47
48   PROCESS (Iclk,Ireset)
49
50     VARIABLE bin_count : Integer range 0 to 1023;
51
52     BEGIN
53
54       Obincount <= CONV_STD_LOGIC_VECTOR(bin_count, 10);
55       IF (Ireset = '0') THEN
56
57         Owr_req <= '0';
58         Owr_clk <= '0';
59         Osig <= "00";
60         bin_count := 0;
61         Odata_rdy <= '0';
62         Spacker_state <= idle;
63         -- Ostate <= "0000";
64
65       ELSIF (Iclk'EVENT and Iclk = '1') THEN
66         CASE Spacker_state IS
67
68           WHEN idle =>
69             Odata_rdy <= '0';
70             IF Isync = '1' THEN

```



```

71  --      Odata <= ("0000000000000000" & Ipri_count);
72      Odata <= Idata(23 downto 12) & ("0000") & Idata(11 downto 0) & ("0000");
73      Owr_req <= '1';
74      Owr_clk <= '0';
75      Osig <= "00";
76      bin_count := 1;
77  --      Spacker_state <= clk_header1;
78      Spacker_state <= clk_rx;
79      ELSE
80      Odata <= (OTHERS => '0');
81      END IF;
82      Ostate <= "0001";
83
84      WHEN clk_header1 =>
85      Owr_clk <= '1';
86      Spacker_state <= header2;
87      Ostate <= "0010";
88
89      WHEN header2 =>
90      Odata <= ("0000000000000000" & Ipri_count);
91      Owr_clk <= '0';
92      bin_count := bin_count + 1;
93      Spacker_state <= clk_header2;
94      Ostate <= "0011";
95
96      WHEN clk_header2 =>
97      Owr_clk <= '1';
98      Spacker_state <= pulse_blank;
99      Ostate <= "0100";
100
101      WHEN pulse_blank =>
102      Owr_clk <= '0';
103      Odata <= (OTHERS => '0');
104      bin_count := bin_count + 1;
105      Spacker_state <= clk_pulse_blank;
106      Ostate <= "0101";
107
108      WHEN clk_pulse_blank =>
109      IF (bin_count < pulse_stopbin) THEN
110      Owr_clk <= '1';
111      Spacker_state <= pulse_blank;
112      ELSE
113      Owr_clk <= '1';
114      Spacker_state <= rx;
115      END IF;
116      Ostate <= "0110";
117
118      WHEN rx =>
119      Owr_clk <= '0';
120      Odata <= Idata(23 downto 12) & ("0000") & Idata(11 downto 0) & ("0000");
121      bin_count := bin_count + 1;
122      Spacker_state <= clk_rx;
123      Ostate <= "0111";
124
125      WHEN clk_rx =>
126      IF (bin_count < rx_stopbin) THEN
127      Owr_clk <= '1';
128      Spacker_state <= rx;
129      ELSE
130      Owr_clk <= '1';
131      Odata_rdy <= '1';
132      Spacker_state <= idle;
133      END IF;
134      Ostate <= "1000";
135
136  --      WHEN zeropad =>
137  --      Owr_clk <= '0';
138  --      Odata <= (OTHERS => '0');
139  --      bin_count := bin_count + 1;
140  --      Spacker_state <= clk_zeropad;
141  --      Ostate <= "1001";
142  --
143  --      WHEN clk_zeropad =>

```

```
144 --      IF (bin_count < zeropad_stopbin - 1) THEN
145 --          Owr_clk <= '1';
146 --          Spacker_state <= zeropad;
147 --      ELSE
148 --          Owr_clk <= '1';
149 --          Spacker_state <= sig;
150 --      END IF;
151 --      Ostate <= "1010";
152
153 --      WHEN sig =>
154 --          Owr_clk <= '0';
155 --          Odata <= (OTHERS => '0');
156 --          Osig <= "11";
157 --          Spacker_state <= clk_sig;
158 --          Ostate <= "1011";
159 --
160 --      WHEN clk_sig =>
161 --          Owr_clk <= '1';
162 --          Odata_rdy <= '1';
163 --          Spacker_state <= idle;
164 --          Ostate <= "1100";
165
166 --      END CASE;
167 --      END IF;
168 --      END PROCESS;
169
170 --      END Data_Packer;
```

F.7 FIFO Status Module

```

1  -- Fifo Status Module
2  -----
3  -- JSW Rust
4  -- University of Stellenbosch
5  -- wr@capestry.net
6  -- 20 March 2006
7  --
8  --
9  Library ieee;
10 use ieee.std_logic_1164.all;
11 use ieee.std_logic_arith.all;
12 use ieee.std_logic_unsigned.all;
13
14 ENTITY Fifo_Status_block IS
15
16
17   PORT
18   (
19     IPCreq      : IN STD_LOGIC;
20     Idata_rdy   : IN STD_LOGIC;
21     Ireset      : IN STD_LOGIC;
22     Iclk        : IN STD_LOGIC;
23
24     Ogrant      : OUT STD_LOGIC;
25     Ocounterlogic : OUT Integer range 0 to 3 := 0
26
27   );
28 END Fifo_Status_block;
29
30
31 ARCHITECTURE Fifo_Status OF Fifo_Status_block IS
32   TYPE Tpacker_type IS (idle, PCready, FIFOready, grant);
33
34   SIGNAL Spacker_state : Tpacker_type;
35
36 BEGIN
37
38   PROCESS (Iclk, Ireset)
39
40     VARIABLE count : Integer range 0 to 3 := 0;
41
42     BEGIN
43       IF (Ireset = '0') THEN
44         Spacker_state <= idle;
45       ELSIF (Iclk'EVENT and Iclk = '1') THEN
46         CASE Spacker_state IS
47
48           WHEN idle =>
49             Ogrant <= '0';
50             IF (IPCreq = '1') THEN
51               Spacker_state <= PCready;
52             ELSIF (Idata_rdy = '1') THEN
53               Spacker_state <= FIFOready;
54             END IF;
55             Ocounterlogic <= 0;
56
57           WHEN PCready =>
58             IF (Idata_rdy = '1') THEN
59               Spacker_state <= grant;
60             END IF;
61             Ocounterlogic <= 1;
62
63           WHEN FIFOready =>
64             IF (IPCreq = '1') THEN
65               Spacker_state <= grant;
66             END IF;
67             Ocounterlogic <= 2;
68
69           WHEN grant =>
70             IF (IPCreq = '1') THEN

```

```
71     Ogrant <= '1';
72     ELSE
73         Spacker_state <= idle;
74         Ogrant <= '0';
75     END IF;
76     Ocounterlogic <= 3;
77
78
79     END CASE;
80 END IF;
81
82 END PROCESS;
83
84 END Fifo_Status;
```

F.8 Zero-padder Module

```

1  -- Zero-Padder Module
2  -----
3  -- JSW Rust
4  -- University of Stellenbosch
5  -- wr@capestry.net
6  -- 20 March 2006
7  --
8  --
9  Library ieee;
10 use ieee.std_logic_1164.all;
11 use ieee.std_logic_arith.all;
12 use ieee.std_logic_unsigned.all;
13
14 ENTITY Zero_Padder_block IS
15
16   GENERIC(rx_stopbin  : integer := 809;
17           zeropad_stopbin : integer := 1024
18           );
19
20   PORT
21   (
22     Iclk      : IN STD_LOGIC;
23     Ienabled  : IN STD_LOGIC;
24     Ireset    : IN STD_LOGIC;
25     Idata     : IN STD_LOGIC_VECTOR(39 downto 0);
26
27     Ofifo_req : OUT STD_LOGIC;
28     Odata     : OUT STD_LOGIC_VECTOR(39 downto 0)
29   );
30 END Zero_Padder_block;
31
32 ARCHITECTURE Zero_Padder OF Zero_Padder_block IS
33
34   TYPE Tpacker_type IS (idle, header2, header3, read_fifo, zeropad, clk_sig );
35
36   SIGNAL Spacker_state : Tpacker_type;
37
38 BEGIN
39
40   PROCESS (Iclk,Ireset)
41
42     VARIABLE bin_count : Integer range 0 to 2048;
43
44     BEGIN
45       IF (Ireset = '0') THEN
46         bin_count := 1;
47         Ofifo_req <= '0';
48         Odata <= (OTHERS => '1');
49         Spacker_state <= idle;
50
51       ELSIF (Iclk'EVENT and Iclk = '1') THEN
52         CASE Spacker_state IS
53
54           WHEN idle =>
55             Ofifo_req <= '0';
56             Odata <= (OTHERS => '0');
57             IF ( Ienabled = '1') THEN
58               Spacker_state <= header2;
59               bin_count :=1;
60             END IF;
61
62           WHEN header2 =>
63             Odata <= (OTHERS => '1');
64             Spacker_state <= header3;
65             Ofifo_req <= '1';
66
67           WHEN header3 =>
68             Odata <= Idata;
69
70

```

```
71     bin_count := bin_count + 1;
72     Spacker_state <= read_fifo;
73     Ofifo_req <= '1';
74
75     WHEN read_fifo =>
76         if bin_count < rx_stopbin THEN -- < 809
77             Odata <= Idata;
78             bin_count := bin_count + 1;
79             Spacker_state <= read_fifo;
80             Ofifo_req <= '1';
81         ELSE
82             Ofifo_req <= '0';
83             Odata <= Idata;
84             bin_count := bin_count + 1;
85             Spacker_state <= zeropad;
86         END IF;
87
88     WHEN zeropad =>
89         if bin_count < zeropad_stopbin THEN -- < 1024
90             bin_count := bin_count + 1;
91             Odata <= (OTHERS => '0');
92             Spacker_state <= zeropad;
93         ELSE
94             Odata <= ("000000")&("11")&("0000000000000000")&("0000000000000000");
95             Spacker_state <= clk_sig;
96         END IF;
97
98     WHEN clk_sig =>
99         Spacker_state <= idle;
100
101
102     END CASE;
103     END IF;
104 END PROCESS;
105
106 END Zero_Padder;
```

F.9 Memory Map Controller

```

1  -- Memory Manager Module
2  -----
3  -- JSW Rust
4  -- University of Stellenbosch
5  -- wr@capestry.net
6  -- 27 March 2006
7  --
8  -- This module handles the reading and writing of compressed and range bin data
9  -- from the PC and to the DP. The DP takes priority to the PC, provided that
10 -- at least one burst map is full (64 PRIs). Data is clocked in and out of the
11 -- module on request.
12 -- It also handles all memory controls (such as addressing, write strobing, etc).
13 -----
14
15 Library ieee;
16 use ieee.std_logic_1164.all;
17 use ieee.std_logic_arith.all;
18 use ieee.std_logic_unsigned.all;
19
20
21 ENTITY Memory_Manager_block IS
22
23   GENERIC(bursts      : integer := 64;
24           samples     : integer := 1024;
25           headerwidth : integer := 16;
26           addrwidth   : integer := 18;
27           pc_addrwidth : integer := 10;
28           datawidth    : integer := 32
29           );
30
31 PORT
32 (
33   Iclk      : IN STD_LOGIC;
34   Ireset_n  : IN STD_LOGIC;
35   OError    : OUT STD_LOGIC;
36
37
38 -- Pulse Compressor Inputs/Outputs
39   Ipc_req   : IN STD_LOGIC;
40   Ipc_we    : IN STD_LOGIC;
41   Ipc_oe    : IN STD_LOGIC;
42   Ipc_addr  : IN STD_LOGIC_VECTOR(pc_addrwidth-1 downto 0);
43   Ipc_data  : IN STD_LOGIC_VECTOR(datawidth-1 downto 0);
44   Ipc_header1 : IN STD_LOGIC_VECTOR(headerwidth-1 downto 0);
45   Ipc_header2 : IN STD_LOGIC_VECTOR(headerwidth-1 downto 0);
46 --   Ipc_bank  : IN STD_LOGIC;
47   Opc_grant : OUT STD_LOGIC;
48
49 -- Doppler Processor Inputs/Outputs
50   Idp_req   : IN STD_LOGIC;
51   Odp_FIFO_clk : OUT STD_LOGIC;
52   Odp_data   : OUT STD_LOGIC_VECTOR(datawidth-1 downto 0);
53   Odp_sig    : OUT STD_LOGIC_VECTOR(1 downto 0);
54   Odp_FIFO_en : OUT STD_LOGIC;
55
56 -- Memory Inputs/Outputs
57   IOsram1_data : INOUT STD_LOGIC_VECTOR(datawidth-1 downto 0);
58   IOsram1_addr : OUT STD_LOGIC_VECTOR(17 downto 0);
59   Osram1_adsc_n : OUT STD_LOGIC;
60   Osram1_adsp_n : OUT STD_LOGIC;
61   Osram1_adv_n  : OUT STD_LOGIC;
62   Osram1_mode_n : OUT STD_LOGIC;
63   Osram1a_oe_n  : OUT STD_LOGIC;
64   Osram1b_oe_n  : OUT STD_LOGIC;
65   Osram1a_ce_n  : OUT STD_LOGIC;
66   Osram1b_ce_n  : OUT STD_LOGIC;
67   Osram1a_wel_n : OUT STD_LOGIC;
68   Osram1b_wel_n : OUT STD_LOGIC;
69   Osram1a_weh_n : OUT STD_LOGIC;
70   Osram1b_weh_n : OUT STD_LOGIC;

```

```

71   OsrAm1a_bwe_n  : OUT STD_LOGIC;
72   OsrAm1b_bwe_n  : OUT STD_LOGIC;
73   OsrAm1_clk     : OUT STD_LOGIC;
74
75   Ovbinpos       : OUT STD_LOGIC_VECTOR(10 downto 0);
76   Odebug         : OUT STD_LOGIC_VECTOR(7 downto 0)
77
78
79
80 );
81 END Memory_Manager_block;
82
83 ARCHITECTURE Memory_Manager OF Memory_Manager_block IS
84
85 TYPE Tmanager_type IS (idle, dp_setup, dp_addr, dp_data, dp_end, pc, pc_end);
86
87 SIGNAL Smanager_state : Tmanager_type;
88 SIGNAL Saddr_blank   : STD_LOGIC_VECTOR(addrwidth-1 downto 0);
89 SIGNAL Sdp_addr      : STD_LOGIC_VECTOR(addrwidth-1 downto 0);
90 SIGNAL Spc_addr      : STD_LOGIC_VECTOR(pc_addrwidth-1 downto 0);
91 SIGNAL Sbank_addr    : STD_LOGIC_VECTOR(addrwidth-1 downto 0);
92 SIGNAL Spc_addr_zero : STD_LOGIC_VECTOR(pc_addrwidth-1 downto 0);
93 SIGNAL Sdp_bank      : STD_LOGIC;
94 SIGNAL Spc_bank      : STD_LOGIC;
95 SIGNAL Soe_n         : STD_LOGIC; -- Memory (OutputEnable)'
96 SIGNAL Sce_n         : STD_LOGIC; -- Memory (ChipEnable)'
97 SIGNAL Swe_n         : STD_LOGIC; -- Memory (writeEnable)'
98 SIGNAL Sdsp_n, Sdsc_n, Sadv_n : STD_LOGIC; -- Memory (ADSP)' and (ADSC)' and (ADV)'
99
100 BEGIN
101
102 OsrAm1a_oe_n <= Soe_n;
103 OsrAm1b_oe_n <= Soe_n;
104 OsrAm1a_ce_n <= Sce_n;
105 OsrAm1b_ce_n <= Sce_n;
106 OsrAm1a_wel_n <= Swe_n;
107 OsrAm1b_wel_n <= Swe_n;
108 OsrAm1a_weh_n <= Swe_n;
109 OsrAm1b_weh_n <= Swe_n;
110 OsrAm1a_bwe_n <= Swe_n;
111 OsrAm1b_bwe_n <= Swe_n;
112 OsrAm1_clk <= Iclk;
113 OsrAm1_adsp_n <= Sdsp_n;
114 OsrAm1_adsc_n <= Sdsc_n;
115
116 Odebug <= "00000001" WHEN (Smanager_state = idle) ELSE
117     "00000010" WHEN Smanager_state = dp_setup ELSE
118     "00000100" WHEN Smanager_state = dp_addr ELSE
119     "00001000" WHEN Smanager_state = dp_data ELSE
120     "00010000" WHEN Smanager_state = dp_data ELSE
121     "00100000" WHEN Smanager_state = dp_end ELSE
122     "01000000" WHEN Smanager_state = pc ELSE
123     "10000000" WHEN Smanager_state = pc_end;
124
125
126
127
128 PROCESS (Iclk, Ireset_n)
129
130 VARIABLE Vdp_bin_pos : Integer range 0 to samples; -- The current DP burst range bin position
131 VARIABLE Vdp_pri_pos : Integer range 0 to bursts; -- The current DP PRI position
132 VARIABLE Vpc_bin_pos : Integer range 0 to samples; -- The current PC range bin position
133 VARIABLE Vpc_burst_cnt : Integer range 0 to samples; -- The current PRI Burst count
134 VARIABLE Vpc_pri_level : Integer range 0 to 2*bursts; -- The level of the PRIs stored by the PC
135
136 BEGIN
137 IF (Ireset_n = '0') THEN
138     Vdp_bin_pos := 0;
139     Vdp_pri_pos := 0;
140     Vpc_bin_pos := 0;
141     Vpc_burst_cnt := 0;
142     Vpc_pri_level := 0;
143

```



```

144     Soe_n <= '1';
145     Sce_n <= '1';
146     Swe_n <= '1';
147     Sdsp_n <= '1';
148     Sdsc_n <= '1';
149     Sadv_n <= '1';
150
151     Soe_n <= '1';
152     Sce_n <= '1';
153     Swe_n <= '1';
154     Osr1_adv_n <= '1';
155     Osr1_mode_n <= '1';
156     OError <= '0';
157     IOSr1_addr(17 downto 0) <= (OTHERS => 'Z');
158     IOSr1_data(datawidth-1 downto 0) <= (OTHERS => 'Z');
159     Saddr_blank <= (OTHERS => '0');
160     Sbank_addr <= (OTHERS => '0');
161     Sdp_bank <= '0';
162     Spc_bank <= '0';
163     Sdp_addr <= (OTHERS => '0');
164     Spc_addr <= (OTHERS => '0');
165     Spc_addr_zero <= (OTHERS => '0');
166     Smanager_state <= idle;
167     Odp_data <= (OTHERS => '0');
168     Odp_sig <= (OTHERS => '0');
169     Odp_FIFO_clk <= '0';
170     Odp_FIFO_en <= '0';
171     Opc_grant <= '0';
172
173
174 ELSIF (Iclk'EVENT and Iclk = '1') THEN
175     CASE Smanager_state IS
176
177         WHEN idle =>
178
179             -- If more than 1 Burst Map avail, service DP requests
180             IF (Ibp_req = '1' and Vpc_pri_level > bursts-1) THEN -- Only service the DP
181                 -- requests when there is more
182                 -- that on PRI Burst ("bursts" PRIs)
183                 Odp_FIFO_en <= '0';
184                 Opc_grant <= '0';
185                 Odp_sig <= "00";
186                 Smanager_state <= dp_setup;
187
188             -- Service PC requests secondly
189             ELSIF (Ipc_req = '1') THEN
190                 IF (Spc_bank = '0') THEN
191                     Spc_addr <= (OTHERS => '0');
192                 ELSE
193                     Spc_addr <= Spc_addr_zero + bursts;
194                 END IF;
195                 Opc_grant <= '1';
196                 Odp_FIFO_en <= '0';
197                 Smanager_state <= pc;
198
199             ELSE
200                 Soe_n <= '1';
201                 Sce_n <= '1';
202                 Swe_n <= '1';
203                 Sdsp_n <= '1';
204                 Sdsc_n <= '1';
205                 Sadv_n <= '1';
206                 Odp_FIFO_en <= '0';
207                 Osr1_adv_n <= '1';
208                 Osr1_mode_n <= '1';
209                 IOSr1_addr(17 downto 0) <= (OTHERS => 'Z');
210                 IOSr1_data(datawidth-1 downto 0) <= (OTHERS => 'Z');
211
212                 Odp_data <= (OTHERS => '0');
213                 Odp_sig <= (OTHERS => '0');
214                 Odp_FIFO_clk <= '0';
215                 Odp_FIFO_en <= '0';
216                 Opc_grant <= '0';

```

```

217     Spc_addr_zero <= (OTHERS => '0');
218
219     Smanager_state <= idle;
220 END IF;
221
222 WHEN dp_setup =>
223     Vdp_pri_pos := 0;           -- Clear the current PRI position.
224     Sdp_addr <= Sdp_addr + Vdp_bin_pos; -- Setup the address to reflect the current DP burst range bin
225     Smanager_state <= dp_addr;
226
227 WHEN dp_addr =>
228     Sdsp_n <= '0';
229     Sce_n <= '0';
230     Soe_n <= '1';
231     IOSram1_addr <= Sdp_addr;
232     Odp_FIFO_en <= '0';
233     Smanager_state <= dp_data;
234
235 WHEN dp_data =>
236     IF (Vdp_pri_pos < bursts - 1) THEN -- If PRI position between 0 to 62
237         Sdsp_n <= '1';
238         Soe_n <= '0';
239         Odp_data <= IOSram1_data;
240         Odp_FIFO_en <= '1';
241         Vdp_pri_pos := Vdp_pri_pos + 1;
242         Sdp_addr <= Sdp_addr + samples;
243         Smanager_state <= dp_addr;
244     ELSE -- If PRI position 63
245         Sdsp_n <= '1';
246         Soe_n <= '0';
247         Odp_sig <= "11"; -- Set the SIG to indicate the end of the Range Burst
248         Odp_data <= IOSram1_data;
249         Odp_FIFO_en <= '1';
250         Vdp_pri_pos := 0;
251         Smanager_state <= dp_end;
252     END IF;
253
254 WHEN dp_end =>
255     Odp_sig <= "00";
256     IF (Vdp_bin_pos < samples - 1) THEN -- If the DP range bin burst pos between 0 - 1022
257         Sce_n <= '1';
258         Soe_n <= '1';
259         Odp_FIFO_en <= '0';
260         Vdp_bin_pos := Vdp_bin_pos + 1;
261         Smanager_state <= idle;
262     ELSE -- If the DP burst range bin 1023 (the last one)
263         Vdp_bin_pos := 0; -- Reset the DP burst range bin indicator
264         Sce_n <= '1';
265         Soe_n <= '1';
266         Odp_FIFO_en <= '0';
267         IF (Sdp_bank = '0') THEN
268             Sdp_addr <= Saddr_blank + samples;
269         ELSE
270             Sdp_addr <= (OTHERS => '0');
271         END IF;
272         Sdp_bank <= not(Sdp_bank); -- One Burst Bank sent to DP, switch the banks
273         Vpc_pri_level := Vpc_pri_level - bursts; -- Lower the PRI level with one Burst Bank
274         Smanager_state <= idle;
275     END IF;
276
277 WHEN pc =>
278     IF (Vpc_bin_pos = 0) THEN -- firstly write the two header infos to the first mem position
279         IOSram1_addr <= ("00000000") & (Spc_addr OR Ipc_addr);
280         IOSram1_data <= Ipc_header1 & Ipc_header2;
281         Sdsp_n <= '1';
282         Sadsc_n <= '0';
283         Swe_n <= '0';
284         Sce_n <= '0';
285         Vpc_bin_pos := Vpc_bin_pos + 1;
286         Smanager_state <= pc;
287
288     ELSIF (Vpc_bin_pos < samples - 1) THEN -- then write the data from the PC
289         IOSram1_addr <= ("00000000") & (Spc_addr OR Ipc_addr);

```

```
290     IOsram1_data <= Ipc_data;
291     Sadsp_n <= '1';
292     Sadsc_n <= '0';
293     Swe_n <= '0';
294     Sce_n <= '0';
295     Vpc_bin_pos := Vpc_bin_pos + 1;
296     Smanager_state <= pc;
297
298     ELSE
299     IOsram1_addr <= ("00000000") & Ipc_addr; -- the last one
300     IOsram1_data <= Ipc_data;
301     Sadsp_n <= '1';
302     Sadsc_n <= '0';
303     Swe_n <= '0';
304     Sce_n <= '0';
305     Smanager_state <= pc_end;
306     END IF;
307
308     WHEN pc_end => -- cleanup and go and sit in idle
309     Vpc_bin_pos := 0;
310     Vpc_pri_level := Vpc_pri_level + 1;
311     Opc_grant <= '0';
312     Sadsc_n <= '1';
313     Swe_n <= '1';
314     Sce_n <= '1';
315     Smanager_state <= idle;
316
317     IF (Vpc_burst_cnt < bursts - 1) THEN
318     Vpc_burst_cnt := Vpc_burst_cnt + 1;
319     ELSE
320     Vpc_burst_cnt := 0;
321     Spc_bank <= not(Spc_bank); -- Swot the active PC Bank
322     END IF;
323     END CASE;
324     END IF;
325     END PROCESS;
326     END Memory_Manager;
```

F.10 Alpha Filter Module

```

1  -- Alpha Filter
2  -----
3  -- JSW Rust
4  -- University of Stellenbosch
5  -- wr@capestry.net
6  -- 10 May 2006
7  --
8  -- This module implements an alpha filter used to average Doppler magnitude data
9  -- over a time period. This period is set by means of the alpha-factor.
10 -- This is implemented as a series of binary divisions. The previous result
11 -- (per range bin) is read from memory, the multiplication (or division then)
12 -- is done and the result is stored again in memory.
13 -----
14 Library ieee;
15 use ieee.std_logic_1164.all;
16 use ieee.numeric_std.all;
17 use ieee.std_logic_unsigned.all;
18
19 ENTITY alpha_filter_block IS
20
21 GENERIC( bursts      : integer := 64;
22          rbins       : integer := 1024;
23          burst_wid   : integer := 6;
24          rbin_wid    : integer := 10;
25          mag_wid     : integer := 16;
26          mem_wid     : integer := 18;
27          div_wid     : integer := 4
28          );
29
30 PORT
31 (
32   Iclk      : IN STD_LOGIC;
33   Ireset    : IN STD_LOGIC;
34   Icompute  : IN STD_LOGIC;
35
36   Iinst_mag : IN UNSIGNED(mag_wid-1 downto 0);
37   div1      : IN INTEGER range 0 to 16;
38   div2      : IN INTEGER range 0 to 16;
39   div3      : IN INTEGER range 0 to 16;
40   div4      : IN INTEGER range 0 to 16;
41
42   MemAddr   : OUT STD_LOGIC_VECTOR(mem_wid-1 downto 0);
43   MemDATA   : INOUT UNSIGNED(mag_wid-1 downto 0);
44
45   mag_valid : OUT STD_LOGIC;
46
47   -- Memory controls
48   adsc_n, adsp_n, adv_n, mode, a_oe_n, b_oe_n : OUT STD_LOGIC;
49   a_weh_n, b_weh_n, a_wel_n, b_wel_n : OUT STD_LOGIC;
50   a_ce_n, b_ce_n, a_bwe_n, b_bwe_n : OUT STD_LOGIC;
51   memclk      : OUT STD_LOGIC
52 );
53 END alpha_filter_block;
54
55 ARCHITECTURE alpha_filter OF alpha_filter_block IS
56
57 TYPE Tfilter_type IS (idle, getprev1, getprev2, partial1, multi);
58
59 SIGNAL Sfilter_state      : Tfilter_type;
60 SIGNAL Spos               : STD_LOGIC_VECTOR(mem_wid-1 downto 0);
61 SIGNAL Sinst_mag         : UNSIGNED(mag_wid-1 downto 0);
62 SIGNAL Stemp             : UNSIGNED(mag_wid-1 downto 0);
63 SIGNAL Spartial1        : UNSIGNED(mag_wid-1 downto 0);
64 SIGNAL Spartial2        : UNSIGNED(mag_wid-1 downto 0);
65 SIGNAL Spartial3        : UNSIGNED(mag_wid-1 downto 0);
66 SIGNAL Spartial4        : UNSIGNED(mag_wid-1 downto 0);
67
68 -- Memory signals
69 SIGNAL Soe_n            : STD_LOGIC; -- Memory (OutputEnable)
70 SIGNAL Sce_n            : STD_LOGIC; -- Memory (ChipEnable)

```

```

71 SIGNAL Swe_n      : STD_LOGIC; -- Memory (writeEnable)'
72 SIGNAL Sdsp_n, Sdsc_n, Sadv_n : STD_LOGIC; -- Memory (ADSP)' and (ADSC)' and (ADV)'
73
74
75 BEGIN
76
77 a_oe_n <= Soe_n;
78 b_oe_n <= Soe_n;
79 a_ce_n <= Sce_n;
80 b_ce_n <= Sce_n;
81 a_wel_n <= Swe_n;
82 b_wel_n <= Swe_n;
83 a_weh_n <= Swe_n;
84 b_weh_n <= Swe_n;
85 a_bwe_n <= Swe_n;
86 b_bwe_n <= Swe_n;
87 memclk <= Iclk;
88 adsp_n <= Sdsp_n;
89 adsc_n <= Sdsc_n;
90
91 PROCESS (Iclk,Ireset)
92
93 BEGIN
94
95 IF (Ireset = '0') THEN
96     Sfilter_state <= idle;
97     Spos <= (OTHERS => '0');
98     Sinst_mag <= TO_UNSIGNED(0,mag_wid);
99     MemAddr <= (OTHERS => '0');
100    mag_valid <= '0';
101
102    -- Setup Memory signals
103    Swe_n <= '1';
104    Sdsp_n <= '1';
105    Sdsc_n <= '1';
106    Soe_n <= '1';
107    Sce_n <= '0'; -- Enable the memory regardless
108
109 ELSIF (Iclk'EVENT and Iclk = '1') THEN
110     CASE Sfilter_state IS
111
112     WHEN idle =>
113         IF (Icompute = '1') THEN
114             -- Setup the memory to latch the memory address
115             Sdsp_n <= '0';
116             MemAddr <= Spos;
117             Sfilter_state <= getprev1;
118             Sinst_mag <= linst_mag;
119             mag_valid <= '0';
120         ELSE
121             Swe_n <= '1';
122             Sdsp_n <= '1';
123             Sdsc_n <= '1';
124             Soe_n <= '1';
125             Sce_n <= '0'; -- Enable the memory regardless
126         END IF;
127
128     WHEN getprev1 =>
129         Sdsp_n <= '1';
130         Soe_n <= '0';
131         Sfilter_state <= getprev2;
132
133     WHEN getprev2 =>
134         Stemp <= Sinst_mag - MemData;
135         Soe_n <= '1';
136         Sfilter_state <= partial1;
137
138     WHEN partial1 =>
139         Spartial1 <= Stemp srl div1;
140         Spartial2 <= Stemp srl div2;
141         Spartial3 <= Stemp srl div3;
142         Spartial4 <= Stemp srl div4;
143         Swe_n <= '0';

```

```
144     Sfilter_state <= multi1;
145
146     WHEN multi1 =>
147     MemData <= UNSIGNED(Sinst_mag) - UNSIGNED(Spartial1) - UNSIGNED(Spartial2)
148         - UNSIGNED(Spartial3) - UNSIGNED(Spartial4);
149     mag_valid <= '1';
150     Swe_n <= '1';
151
152     IF (Spos = bursts*rbins-1) THEN
153         Spos <= (OTHERS => '0');
154     ELSE
155         Spos <= Spos + 1;
156     END IF;
157     Sfilter_state <= idle;
158
159     END CASE;
160 END IF;
161 END PROCESS;
162 END alpha_filter;
```