

University of Windsor

Scholarship at UWindor

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

2010

An FPGA-based 77 GHzs RADAR signal processing system for automotive collision avoidance

Sundeep Lal
University of Windsor

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>

Recommended Citation

Lal, Sundeep, "An FPGA-based 77 GHzs RADAR signal processing system for automotive collision avoidance" (2010). *Electronic Theses and Dissertations*. 7979.
<https://scholar.uwindsor.ca/etd/7979>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

**An FPGA-based 77 GHz
RADAR
Signal Processing System for
Automotive Collision Avoidance**

By
Sundeep Lal

A Thesis
Submitted to the Faculty of Graduate Studies
Through Electrical and Computer Engineering
In Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science
at the University of Windsor

Windsor, Ontario, Canada

2010

© Sundeep Lal



Library and Archives
Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-62749-5
Our file Notre référence
ISBN: 978-0-494-62749-5

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.


Canada

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

An FPGA implementable Verilog HDL based signal processing algorithm has been developed to detect the range and velocity of target vehicles using a MEMS based 77 GHz LFM CW long range automotive radar. The algorithm generates a tuning voltage to control a GaAs based VCO to produce a triangular chirp signal, controls the operation of MEMS components, and finally processes the IF signal to determine the range and velocity of the detected targets. The Verilog HDL code has been developed targeting the Xilinx Virtex-5 SX50T FPGA. The developed algorithm enables the MEMS radar to detect 24 targets in an optimum timespan of 6.42 ms in the range of 0.4 to 200 m with a range resolution of 0.19 m and a maximum range error 0.25 m. A maximum relative velocity of ± 300 km/h can be determined with a velocity resolution in HDL of 0.95 m/s and a maximum velocity error of 0.83 m/s with a sweep duration of 1 ms.

A Sincere Dedication

To mom, dad, amma, thatthha, Mannu and Sunali with love...

It is your ever-encouraging faith in me that keeps me going.

Om Sai Ram

Acknowledgement

Before all I submit my prayers to Almighty God who has always kept His Guiding Hand upon me and whose bounteous Blessings reveal themselves in the form of all the lovely people who have made this work possible.

With utmost sincerity I express my gratitude and respect to my advisor Dr. Sazzadur Chowdhury, who has always inspired me to work with honesty, integrity and discipline. His timely guidance and reassuring aura have been indispensable boons contributing to the completion of this thesis.

I am thankful to Mohan Thangarajah, Matt Murawski and Tugrul Zure for their helpful comments, and extend my note of thanks to Dr. Mosaddequr Rehman for his encouraging words.

This note would be incomplete without thanking Andria Ballo for always being there for every engineering soul in distress, her ever-readiness to help, valuable guidance and the uncanny ability to remember the name of every single student.

Lastly I would like to mention the names of my best friends Jitender and Rishi who have always been around with their comic and witty remarks that worked well in lowering my stress level throughout the course of this research.

Table of Contents

Author's Declaration of Originality.....	iii
Abstract	iv
A Sincere Dedication	v
Acknowledgement.....	vi
List of Figures.....	ix
List of Tables	xi
List of Abbreviations.....	xii
Nomenclature.....	xiv
CHAPTER 1: INTRODUCTION	1
1.1 Problem Statement	1
1.2 Hypothesis	6
1.3 Motivation	6
1.4 Research Methodology	7
1.5 Principal Results	8
1.6 Thesis Organization	9
CHAPTER 2: LITERATURE SURVEY.....	10
2.1 Literature Review	10
2.1.1 Selecting the Type of Radar.....	13
2.1.2 Beamforming with Phased Array Antennæ.....	15
2.1.3 Direction of Arrival Estimation using Phased Array Antennæ.....	20
2.1.4 Frequency Generation, Tuning and Linearity.....	21
2.1.5 Selecting the Development Platform	22
2.1.6 State-of-the-Art in Automotive Radar.....	24
2.1.7 Recent Work Done in FPGA-based LFMCW Digital Signal Processing.....	28
CHAPTER 3: REQUIREMENTS FOR THE TARGET FMCW SYSTEM.....	29
3.1 System Requirements Identification.....	29
3.2 Selecting the Required FMCW Waveform.....	31
3.3 Linear Frequency Modulated Continuous Wave Radar	32
3.3.1 Derivation of Range and Velocity for LFMCW	34
3.3.2 LFMCW Radar Signal Generation using VCO	38
3.3.3 Received Echo Signal Conditioning for LFMCW.....	39
3.4 Digital Signal Processing Tools.....	41
3.4.1 Time-domain Window	41
3.4.2 The Fast Fourier Transform	48
3.4.3 Constant False Alarm Rate (CFAR) Processor.....	49
3.4.4 Miscellaneous Topics.....	52
CHAPTER 4: RADAR CONTROL AND SIGNAL PROCESSING ALGORITHM	59
4.1 Radar Transmitter Control and MEMS RF SP3T Switch Control.....	61

4.2	Radar Receiver Flow Control and Signal Processing.....	63
4.3	Selecting the Radar Sweep Bandwidth.....	65
4.4	Configuration of System Components	68
4.4.1	ADC	68
4.4.2	FFT	68
4.4.3	CFAR.....	68
4.4.4	Peak Pairing	69
4.5	Developed Algorithm Summary	69
CHAPTER 5: SOFTWARE IMPLEMENTATION AND SIMULATION		71
5.1	Software Implementation of the Radar Signal Processing Algorithm.....	71
5.2	Testing Stage 1: Windowing versus No Windowing.....	74
5.2.1	Results without Windowing	75
5.2.2	Results with Windowing	76
5.3	Testing Stage 2: 3-Lane Highway Scenario with Narrow Beam.....	77
5.4	Testing Stage 3: Scenario with 7 Targets Detected in a Single Wide Beam	86
5.5	Observations from Software Simulation Results.....	91
CHAPTER 6: HARDWARE IMPLEMENTATION AND VALIDATION		92
6.1	Hardware Implementation of the Radar Signal Processing Algorithm	92
6.1.1	Radar Signal Processing Algorithm on FPGA	94
6.2	Validation of the HDL Implementation of the Signal Processing Algorithm	115
6.2.1	Test 1: 3-Lane Highway Scenario with Narrow Beam	116
6.2.2	Test 2: Scenario with 7 Targets Detected in a Single Wide Beam	124
6.3	Hardware Synthesis Results for the Developed Algorithm	131
6.4	Observations from HDL Implementation of the Developed Algorithm	133
CHAPTER 7: CONCLUSIONS.....		135
7.1	Discussions and Conclusions	135
7.2	Future Work	136
REFERENCES		138
APPENDIX		142
A1.	MATLAB listing for Radar Signal Processing Algorithm testing	143
A2.	MATLAB listing for error calculation from 10-bit rounding of Window functions.....	150
A3.	HDL listing for TLC.....	151
A4.	HDL listing for SAMPLER.....	158
A5.	HDL wrapper for Xilinx FFT v7.0 core	162
A6.	HDL listing for FDR.....	163
A7.	HDL listing for PSD	168
A8.	HDL listing for CFAR.....	170
A9.	HDL listing for PPM.....	174
VITA AUCTORIS		182

List of Figures

Figure 1.1: Automotive radar system conceptual diagram	4
Figure 2.1: Pulsed Doppler radar waveform	12
Figure 2.2: Transmit signal frequency for FSK-CW and triangular FMCW.	13
Figure 2.3: Six patch array antenna of radiating elements	16
Figure 2.4: Radiation pattern for 3 patch array and 6 patch array	17
Figure 2.5: Analog beamformer	18
Figure 2.6: Schematic of the intrinsic beamforming capability of the Rotman lens.....	19
Figure 2.7: Non-linear frequency response of a typical RF VCO.	22
Figure 2.8: Radar applications in the automotive industry	25
Figure 2.9: Distronic Plus by Mercedes-Benz	26
Figure 3.1: FMCW waveforms left to right: Sine, Saw-tooth and Triangular	31
Figure 3.2: LFM CW Transmit, Receive and Beat frequency.	33
Figure 3.3: FPGA based tuning voltage generation for VCO to produce LFM CW chirps.....	38
Figure 3.4: Time-domain RF signal showing up and down frequency chirps.....	39
Figure 3.5: Conceptual diagram of an RF mixer.	40
Figure 3.6: Spectral leakage due to rectangular windowing	42
Figure 3.7: Time and Frequency domain representations of various window functions.....	43
Figure 3.8: CA-CFAR processor architecture	51
Figure 3.9: Safe distance between two vehicles	58
Figure 4.1: Flowchart for the operation of modulation and transmitter control unit.	62
Figure 4.2: Radar signal processing algorithm	64
Figure 4.3: Variation of range resolution with LFM CW sweep bandwidth.	66
Figure 5.1: Flowchart for MATLAB simulation of the radar signal processing algorithm.	73
Figure 5.2: Test case highway scenario	78
Figure 5.3: Time-domain signals for the up and down sweep of Beam 1	80
Figure 5.4: Frequency analysis of return signals in Beams 1, 2 and 3	83
Figure 5.5: Hypothetical scenario with a single wide-angle antenna beam	86
Figure 5.6: Frequency analysis for the wide-angle beam scan.....	88
Figure 6.1: Xilinx Virtex-5 SX50T mounted on Development Board ML506	93
Figure 6.2: HDL blocks for the radar signal processing algorithm.....	94
Figure 6.3: Black box view of radar control and signal processing algorithm	95
Figure 6.4: TLC in Xilinx ISE RTL viewer.....	96
Figure 6.5: Xilinx ISE RTL view of SAMPLER unit with sub-modules WINDOW and TDR.....	100
Figure 6.6: Timing diagram for SAMPLER module.....	101
Figure 6.7: Xilinx ISE RTL view of FFT v7.0 core.....	103
Figure 6.8: Timing diagram for Xilinx FFT core v7.0	104
Figure 6.9: Xilinx ISE RTL schematic view of two sub-modules forming the FDR unit	106

Figure 6.10: Timing diagram for FDR.....	107
Figure 6.11: Peak intensity calculation unit	108
Figure 6.12: Four PSD units work in parallel	109
Figure 6.13: RTL view of CFAR module.....	110
Figure 6.14: Timing diagram for CFAR module.....	111
Figure 6.15: RTL view of PPM	112
Figure 6.16: Timing diagram for PPM showing 4 detected targets from CFAR.....	113
Figure 6.17: Test case highway scenario	116
Figure 6.18: Test Case 1: 3-Lane simulation waveform results from Xilinx ISE Simulator	118
Figure 6.19: HDL simulation results for Test Case 1	122
Figure 6.20: Test Case 2: Hypothetical scenario with a single wide-angle antenna beam	125
Figure 6.21: HDL simulation results for Test Case 2	128
Figure 6.22: LFM CW sweep timing diagram for the realized HDL system.	134
Figure 7.1: Typical angle and range coverage for SRR, MRR and LRR	137

List of Tables

Table 1.1: Fatality count around the globe.....	3
Table 2.1: Speed Comparison of a typical FPGA.....	23
Table 2.2: Commercially available new generation of automotive radar systems.....	27
Table 2.3: Previous generation of automotive radar systems.....	27
Table 3.1: The next generation of Long Range Radar	30
Table 3.2: Comparison of common Window functions.....	47
Table 3.3: Atmospheric attenuation at 70-80 GHz.....	56
Table 4.1: Initially provided System Specifications	60
Table 4.2: Final parameters for the devised signal processing algorithm.....	70
Table 5.1: Practical Test Case Highway Scenario – Target Description.....	78
Table 5.2: Results from MATLAB Simulation of 3-Lane Narrow Beam Scenario.....	84
Table 5.3: Errors from MATLAB Simulations of 3-Lane Narrow Beam Scenario	85
Table 5.4: Hypothetical Test Case – Target Description.....	87
Table 5.5: Results from MATLAB Simulations of 3-Lane Single Wide Beam Scenario	90
Table 5.6: Errors from MATLAB Simulations of 3-Lane Single Wide Beam Scenario	90
Table 6.1: Xilinx Virtex-5 SX50T features.....	93
Table 6.2: Port description for TLC	97
Table 6.3: Port description for SAMPLER	101
Table 6.4: Xilinx FFT IP core parameterization	102
Table 6.5: Port description for FFT	103
Table 6.6: Port description for FDR	106
Table 6.7: Port description for PSD	108
Table 6.8: Port description for CFAR	110
Table 6.9: Sensitivity Adjustment for CA-CFAR Processor.....	112
Table 6.10: Port description for PPM	113
Table 6.11: Results from HDL Simulation of 3-Lane Narrow Beam Scenario.....	123
Table 6.12: Errors from HDL Simulations of 3-Lane Narrow Beam Scenario	124
Table 6.13: Results from HDL Simulations of 3-Lane Single Wide Beam Scenario.....	129
Table 6.14: Errors from HDL Simulations of 3-Lane Single Wide Beam Scenario	129
Table 6.15: Comparison of MATLAB and HDL range results for wide beam scenario	130
Table 6.16: Comparison of MATLAB and HDL velocity results for wide beam scenario	131
Table 6.17: Resource Usage for the Radar Signal Processing Algorithm on Virtex-5 SX50T	132
Table 6.18: Timing Achievements of HDL Implementation.....	132
Table 6.19: Achieved Timing Details for Developed LFM CW Radar System	134

List of Abbreviations

MEMS – Microelectromechanical Systems

Radar – Radio Detection and Ranging

RF – Radio Frequency

SP3T – Single Pole Triple Throw

PRF – Pulse Repetition Frequency

DSP – Digital Signal Processor(-ing)

FPGA – Field Programmable Gate Array

DAC – Digital to Analog Converter

ADC – Analog to Digital Converter

FSK – Frequency Shift Keying

LFMCW – Linear Frequency Modulated Continuous Wave

HDL – Hardware Description Language

ECCM – Electronic Counter-Countermeasures

FFT – Fast Fourier Transform

DFT – Discrete Fourier Transform

DIT – Decimation In Time

DIF – Decimation In Frequency

CA(OS)-CFAR – Constant False(Ordered Statistics) Constant False Alarm Rate

RTL – Register Transfer Level

TLC – Top Level Control

TDR – Time-domain Data RAM

FDR – Frequency-domain DATA RAM

PSD – Power Spectral Density

PPM – Peak Pairing Module

RCS – Radar Cross-Section

CPI – Coherent Processing Interval

VCO – Voltage Controlled Oscillator

LRR – Long Range Radar

MRR – Medium Range Radar

SRR – Short Range Radar

IF – Intermediate Frequency

MSPS – Mega-Samples Per Second

LUT – Look-Up Table

FF – Flip-Flop

BUFG – Global Buffer

BUFGCTRL – Global Clock Buffer

RAM – Random Access Memory

ROM – Read-Only Memory

DSP48E – Xilinx Digital Signal Processing Slice (5th Generation)

DOA – Direction Of Arrival

ISE – Integrated System Environment

LPF – Low Pass Filter

AWGN – Additive White Gaussian Noise

EM – Electro-Magnetic

MMIC – Monolithic Microwave Integrated Circuits

UWB – Ultra-Wide Band

SMD – Surface Mount Device

Nomenclature

r = target range

c = speed of RF waves through air

$T_{\text{two-way}}$ = two-way travel time for RF wave from radar sensor to target and back

v_{rel} = relative velocity

v_{target} = target velocity

v_{host} = host vehicle velocity

f_d = Doppler frequency shift

λ = radio wave wavelength

f_b = beat frequency or instantaneous intermediate frequency

f_t = transmit signal frequency

f_r = received signal frequency

τ_0 = travel time for RF wave from radar sensor to target

B = LFMCW sweep bandwidth

T = LFMCW sweep duration

k = rate of change of frequency in LFMCW sweep = B / T

X_k = frequency domain sample

x_n = time domain sample

P_{fa} = probability of false alarm

T_c = CFAR dynamic threshold

σ = Radar Cross-Section of a target

N_{Th} = thermal noise

T_A = absolute atmospheric temperature

SNR_Q = quantization signal-to-noise ratio

f_s = sampling frequency

f_{res} = frequency resolution of FFT

CHAPTER 1: INTRODUCTION

This chapter starts with a clear definition of the issue this research work addresses, explaining the importance of the work and its outcomes. Facts about road safety and accident records around the globe are presented and automotive radar applications are identified as an effective means of enhancing vehicular safety features. The potential benefits of automotive radar systems in road safety are highlighted, and the radar being developed at the University of Windsor is presented along with a concise operating principle. Finally, the principal results of this research work are listed.

1.1 Problem Statement

The objective and goal of this research is to develop an FPGA-implementable signal processing algorithm for use in a Microelectromechanical Systems (MEMS) based linear frequency modulated continuous wave (LFMCW) long range automotive radar to determine the range and velocity of targets in the vicinity of a host vehicle.

Loss of lives and property damage due to automotive collisions can be minimized if it is possible to detect the proximity of other vehicles, pedestrians, and obstacles in real-time using advanced microelectromechanical systems (MEMS) based sensor technology. The current technology for short and long range proximity detection, such as: stand-alone ultrasonic sensor or sensor arrays, electromagnetic radar units (present in high-end vehicles only), lasers, and cameras mounted on side view mirrors fall short of establishing a real-time dynamic safety shell around a vehicle due to their high latency time associated with microelectronic signal processing and need for mechanical scanning of the target area in case of radars. Moreover, due to high cost of stand-alone manufacturing, automakers are reluctant to incorporate these solutions in low-end

vehicles. As a result the overall road safety situation remains almost the same even if some of the vehicles are equipped with advanced collision or pre-crash warning systems. To put the problem in perspective, less than 1% of vehicles running in Canadian highways are equipped with advanced radar sensors.

Market research firm Strategy Analytics predicts that over the period 2006 to 2011, the use of long-range distance warning systems in cars could increase by more than 65 percent annually, with demand reaching 3 mn units in 2011, with 2.3 mn of them using radar sensors. By 2014, 7 percent of all new cars will include a distance warning system, primarily in Europe and in Japan [18].

Global auto industries and governments are extensively pursuing radar based proximity detection systems for (1) ACC support with Stop&Go functionality, (2) collision warning, (3) pre-crash warning, (4) blind spot monitoring, (5) parking aid (forward and reverse), (6) lane change assistant and (7) rear crash collision warning. The European Commission (EC) has set an ambitious target to reduce road deaths by 50% by the end of 2010. In North America alone the rate of fatalities related to road accidents has been stagnant at approximately 43,000 per year, which sums to a huge annual loss of life and property [15]. It has been concluded that the use of Forward Collision Warning long range radar and Lane Departure Warning camera-based sensor among other security features will become very effective to reduce road fatality rates. In [15], it has been mentioned that with the proposed crash prevention technologies equipped in vehicles, the number of crashes can be reduced by 3.8 mn in North America, and the number of human lives saved from that amounts close to 17,000 per year. This warrants the use of long range radar as an indispensable feature to improve highway safety and minimize loss of lives and property damage.

Table 1.1: Fatality count around the globe [15]

	Fatality count in 2005	Fatality rate per 100 million vehicle miles
North America	43,443	1.5
European Union	41,600	1.3
Japan	6,871	1.4

Pulse-Doppler vs FMCW Radar

Some of the earlier automotive radar applications relied on a high-power Pulsed Doppler radar technique, but the suitability of the technique came under criticism after the televised failure of the Mercedes-Benz pulsed radar assisted Distronic cruise control system on Stern TV in 2005 [17]. This has instigated the industry to study and use the FMCW radar technique for modern radar systems. FMCW radar in automotive applications is still a developing field of study, with on-going research at all system levels including signal processing and RF hardware design.

The MEMS Radar

The application of an automotive radar system is classified according to the range it covers. Long range radar (LRR) and medium range radar (MRR) are used in cruise control and collision avoidance, and short range radar (SRR) is used in collision avoidance, crash-prevention and parking-assist systems.

Having established that automotive radar can be very helpful in reducing the number of fatal accidents, it is essential that low cost and reliable radar systems be made to improve road safety globally. Lower cost (compared to \$2000-\$3000 approx. for current systems) will enable even lower-end vehicles to be equipped with safety options, boosting road safety.

MEMS technology offers the advantage of realizing low cost batch fabricatable high performance RF components like Rotman lens, RF switches that can be used to realize a compact high performance lightweight radar in a small form factor. Such a MEMS based radar system has been developed at the University of Windsor, Ontario, Canada. A block diagram for the MEMS based radar has been developed as part of this thesis and is shown in Figure 1.1.

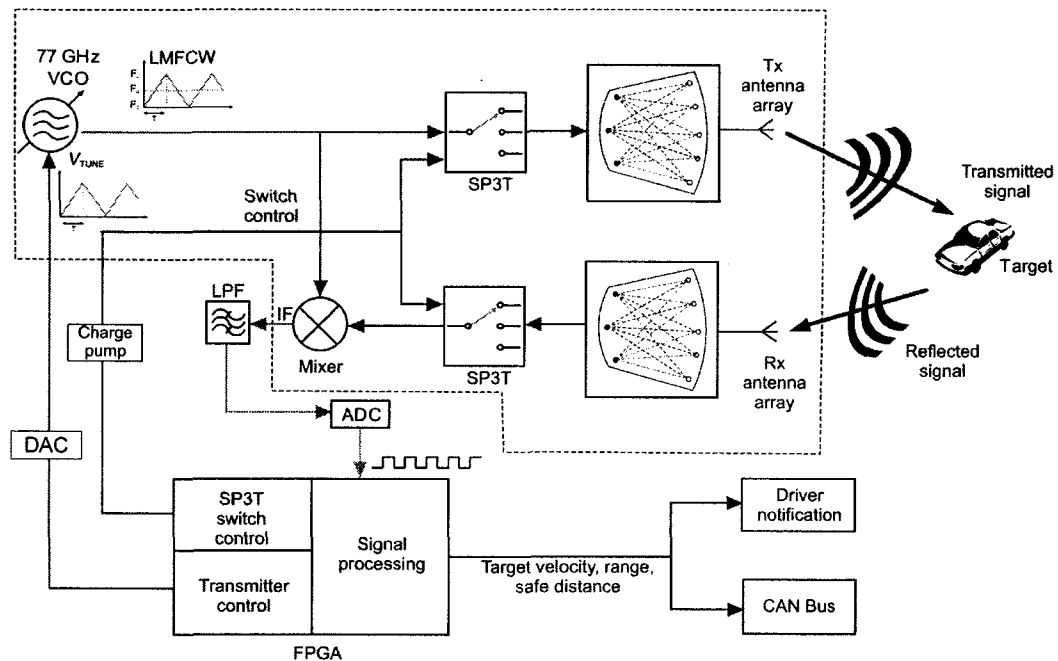


Figure 1.1: Automotive radar system conceptual diagram showing Rotman lens and SP3T switches. Only major components are shown.

MEMS Radar Operating Principle:

1. An FPGA implemented control circuit generates a triangular signal (V_{tune}) to modulate a voltage controlled oscillator (VCO) to generate a linear frequency modulated continuous wave (LFMCW) signal having a frequency sweep range of 0-400 MHz centered at 77 GHz.
2. The signal is fed to a MEMS SP3T switch.

3. An FPGA implemented control algorithm controls the SP3T switch to sequentially switch the LFM CW signal among the three beam ports of a MEMS implemented Rotman lens.
4. As the LFM CW signal arrives at the array ports of the Rotman lens after traveling through the Rotman lens cavity, the time-delayed in-phase signals are fed to a microstrip antenna array that radiates the signal in a specific direction.
5. The sequential switching of the input signal among the beamports of the Rotman lens enables the beam to be steered across the target area in steps by a pre-specific angle.
6. On the receiving side, a receiver antenna array receives the signal reflected off a vehicle or an obstacle and feeds the signal to another SP3T switch through another Rotman lens.
7. An FPGA based control circuit controls the operation of the receiver SP3T switch so that the signal output at a specific beamport of the receiver Rotman lens can be mixed with the corresponding transmit signal.
8. The output of the receiver SP3T switch is passed through a mixer to generate an IF signal in the range of 0-200 KHz.
9. An Analog-to-digital converter (ADC) samples the received IF signal and converts it to a digital signal.
10. Finally, an FPGA implemented algorithm processes the digital signal from the ADC to determine the range and velocity of the detected target.

The goal of this thesis is to develop the FPGA implementable algorithm to realize the functionality of the MEMS Radar system as described above to detect the distance and velocity of target vehicle(s) in a pre-specified range to meet the requirements of a long range automotive radar.

1.2 Hypothesis

Owing to the passive nature of the MEMS Rotman lens, a relatively enhanced cycle time can be achievable as compared to current state-of-the-art systems. The FPGA based control and signal processing algorithm can be implemented as a low cost ASIC. Together with the miniature MEMS components, and appropriate off-the-shelf radar frontend, the target system would offer a highly compact higher performance small form factor radar solution for automotive applications.

The efficiency of the FPGA control and signal processing implementation will be gauged by resource usage, speed and its accuracy compared to floating-point MATLAB simulations.

1.3 Motivation

The automotive scenario is fast-paced, with every millisecond being precious in time-critical applications such as collision avoidance and collision mitigation systems. Existing automotive radar sensors are critical components of the overall safety system of a vehicle, and their cycle time or refresh time (these terms are used interchangeably through this thesis) – the time over which the entire field of view is covered – should be considerably short. At a speed of 200 km/h a vehicle travels 2.78 meters in 50ms, the refresh time of a typical existing system such as Bosch LRR3. Such latency in the safety mechanism of the vehicle in response to a potential threat increases the possibility of an accident.

This thesis aims at exploiting the intrinsic beamforming capability of the Rotman lens, the fast signal processing and parallel computing on FPGAs, and the reliability of the LFM CW method in target detection to provide digital signal processing and control of a lightweight state-of-the-art compact radar sensor for automotive safety systems.

1.4 Research Methodology

The course of developing an FPGA-based LFM CW radar signal processing algorithm for the 77 GHz MEMS radar sensor involves the following steps:

1. Study the initial system specifications provided by the project supervisor based on the MEMS based radar sensor presented in [1].
2. Survey of literature on radar systems, radar signal processing and target tracking, radio frequency attenuation, and acceptable parameters for automotive collision avoidance systems.
3. Development of a robust and fast radar signal processing algorithm and development of a mathematical model of the same.
4. Decision on system peripherals such as data converters and interfaces according to target system parameters.
5. Simulation of the algorithm in MATLAB for a typical highway traffic test scenario.
6. Development of HDL code for implementation on FPGA.
7. Verification of the developed HDL code using the same test scenario as in (3) for a comparison of accuracy between fixed-point HDL signal processing and floating-point MATLAB processing.
8. Fine-tuning and optimization of the HDL code for implementation on target FPGA.

1.5 Principal Results

1. A reusable and parameterizable ready-to-implement LFMCW radar signal processing algorithm for FPGA/ASIC with minimal latency of 212 μs and a competitive radar cycle time of 6.78 ms has been created. Major achieved performance specifications of the developed system are listed below:
 - Operating frequency – 77 GHz (within regional radio frequency allocation)
 - Bandwidth – 800 MHz (within regional bandwidth limits)
 - Maximum (Minimum) distance – 200 (0.4) meters
 - Range resolution (in HDL) – 0.19 meters
 - Maximum target range error – 0.25 meters
 - Worst-case range accuracy – 99.75% (beyond 100 meters)
 - Maximum relative velocity - ± 300 km/h (receding and approaching target)
 - Velocity resolution (in HDL) – 0.95 m/s
 - Maximum target velocity error – 0.83 m/s
 - Worst-case velocity accuracy – 99.17% (beyond 60 km/h)
 - Beam steerability - $\pm 4.5^\circ$ (beam width 9°) [1]
 - Maximum target count for 3-beam Rotman lens radar – 24
2. A superior signal processing time compared to recent FPGA-based implementations as presented in [28].

1.6 Thesis Organization

Developing from the introduction in Chapter 1, Chapter 2 concisely summarizes the available literature of radar technology and studies state-of-the-art standards in automotive radar sensors, and their applications, in order to produce a list of target specifications for the MEMS radar sensor developed at the University of Windsor.

Chapter 3 of this thesis propounds a thorough background and mathematical conceptualization of radar topics, focusing on LFM CW radar theory. The underlying concept of radio detection and ranging systems is presented considering different issues affecting performance, such as noise, attenuation and non-linearity all with reference to the design of an automotive radar sensor. Essential signal conditioning and processing approaches are discussed with focus on frequency analysis of the radar signal.

Chapter 4 builds on the foundations laid in Chapter 2, and presents the developed radar signal processing algorithm. The different components in the algorithm are discussed in further detail.

Chapter 5 shows a MATLAB implementation and simulation of the radar signal processing algorithm. Effects of different signal processing methods such as time-domain windowing and Fourier transform on a noisy signal are studied. Simulation results are presented to validate the accuracy of the developed algorithm.

Hardware implementation of the conceived algorithm is laid out in the form of FPGA modules in Chapter 6. Realization of the modules is carried out in Verilog HDL (Verilog 2005 – IEEE Standard 1364-2005) using Xilinx development software, where fixed-point and resource usage considerations for the signal processing, sampling and control algorithm are presented. Code validation is done using Xilinx ISE ISim simulator with the same real-valued time-domain data samples as used in MATLAB code verification. Chapter 7 furnishes the concluding remarks on the research work, shedding light on achieved system specifications, future amendments and possible expansions to the work presented herein.

CHAPTER 2:

LITERATURE SURVEY

This chapter covers a review of the existing literature on radar systems, identifying the types of radars available. The advantages of frequency modulated continuous wave (FMCW) radar over pulsed and frequency shifting radars are recognized, based on which the decision of using FMCW radar is selected as the right match for the target automotive radar. Important radar concepts are described, especially beamforming and beam steering for solid-state phased array antenna radars. The Rotman lens' role in this radar system is described, and a platform for the radar signal processing algorithm is selected. The latter part of this chapter presents state-of-the-art automotive radar systems, highlighting the Bosch LRR3 as a guideline for the specifications of the system developed in this thesis.

2.1 Literature Review

Radar technology has long been used in military, aerospace, marine, geographical, weather monitoring and global positioning applications [9]. The first conceptualization of RF radar was made in 1920 by Bells Labs and in 1922 by Guglielmo Marconi [10]. It has recently found increasing popularity in the automotive arena with automobile manufacturers incorporating radars for adaptive cruise control (ACC), parking aid, pre-crash warning, and collision avoidance systems.

Radar systems can be classified by two major types: Pulsed and Continuous Wave [2]. Both implementations have distinct operating principle, transmit signal generation, receive signal conditioning and processing, control and synchronization issues, and power requirements.

Pulsed Radar: Pulsed radars send short-duration (in the range of a few hundred nanoseconds) high-power (typically in kilowatts range) pulses which illuminate a target in the line-of-sight. A pulse is essentially a sinusoid (carrier wave) at the chosen operating frequency: the Doppler shift in the carrier wave frequency within the pulse corresponds to the relative velocity of the target, and the time taken for the radar to detect a return of the pulse determines the range of the target. The pulse repetition frequency (PRF) between two consecutive pulses is a critical factor in Pulsed radar design. Pulsed radar is a mature technology. The waveform for Pulsed radar is shown in Figure 2.1.

In Pulsed radar the range and relative velocity of the target are determined as follows:

$$\text{Range, } r = \frac{c \times T_{\text{two-way}}}{2} \quad (2.1)$$

$$\text{Relative velocity, } v_{\text{rel}} = \frac{-f_d \times \lambda_0}{2} \quad (2.2)$$

Here, c is the speed of electromagnetic radiation in air, $T_{\text{two-way}}$ is the two-way travel time for a pulse reflected from the target to return to the source, f_d is the Doppler shift and λ_0 is the operating wavelength.

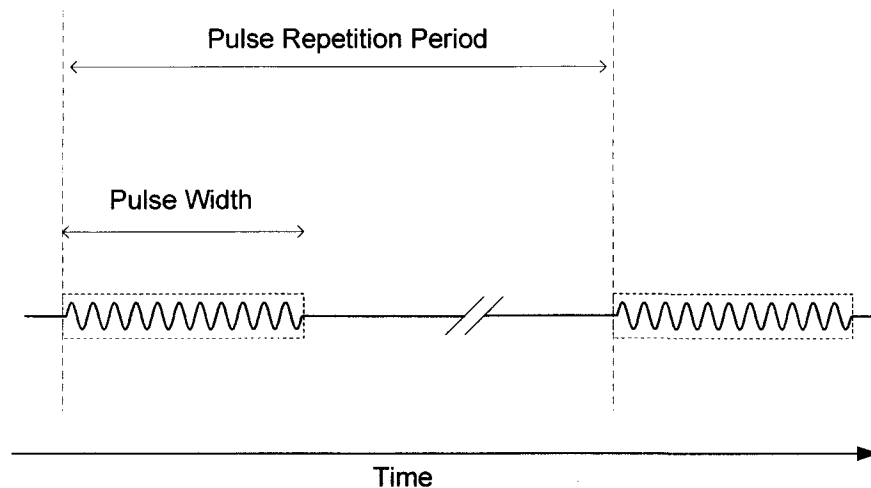


Figure 2.1: Pulsed Doppler radar waveform – short pulses with high peak power are broadcast in the direction of the target. A pulse contains a few hundred oscillations of the RF signal. The return of a pulse is timed and analysed for Doppler shift. [11]

Continuous Wave Radar: Continuous Wave radars continuously transmit the RF wave at a lower power level (typically less than 50mW) and a selected frequency. The CW radar systems continuously observe the return from a target over a period of time, commonly called the Coherent Processing Interval (CPI). During the CPI, the instantaneous transmit and receive signals are mixed, and the resultant intermediate frequency (IF) signal is assessed over the CPI for valid targets. The CW radar technology is still under constant refinement with new strategies related to both hardware and signal processing algorithms being developed. There are two prime implementations of CW radar: FH- (Frequency Hopping) or FSK-CW (Frequency Shift Keying) radar and FMCW (Frequency Modulated) radar. In FSK-CW the RF jumps between multiple frequencies over a CPI, whereas FMCW makes use of a frequency chirp in a sine, saw-tooth or triangular fashion [12]. The transmit waveforms for both CW radar types are shown in Figure 2.2.

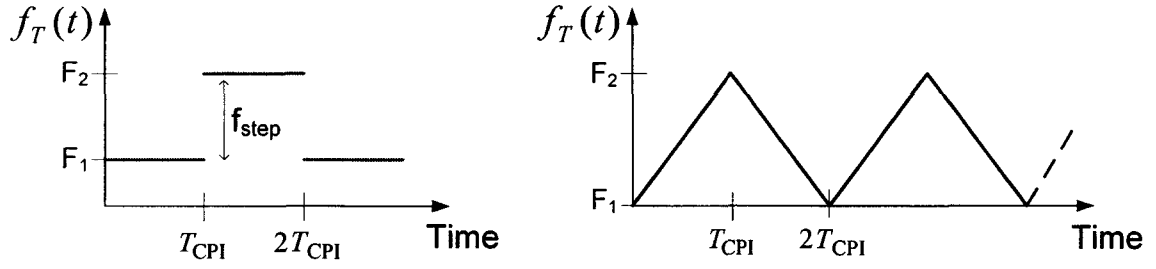


Figure 2.2: Transmit signal frequency for FSK-CW (left) radar – frequency hopping – and triangular FMCW (right) radar – linear frequency up and down sweeps (or chirps).

Range for FSK-CW radar,

$$r = \frac{c\Delta\Phi}{4\pi(F_2 - F_1)} \quad (2.3)$$

Relative Velocity for FSK-CW radar,

$$v_{rel} = \frac{-f_d \times \lambda_0}{2} \quad (2.4)$$

Here, c is the speed of electromagnetic radiation in air, $\Delta\Phi$ is the difference in phase shift at the two frequencies F_1 and F_2 , f_d is the Doppler shift and λ_0 is the operating wavelength.

2.1.1 Selecting the Type of Radar

Pulsed Doppler, FSK-CW and LFM CW radars are distinguished by the type of waveform, the operating power, computational cost, hardware requirements and application. Where Pulsed radar suffers lower atmospheric attenuation, CW radar is well suited to short-range applications with low transmit power. Keeping in mind the automotive scenario, which is the central theme around this thesis, the following disadvantages are visible in these radar types.

Pulsed Doppler disadvantages:

- Velocity measurement limited by *blind speed* when f_d is a multiple of the PRF.
- Maximum measurable Doppler shift has to be less than PRF to avoid ISI among different pulses and target returns.
- To reduce the above velocity ambiguity the PRF can be increased, however increasing the PRF creates range ambiguity.
- Relatively high power requirements in the automotive scenario.
- Greater risk of jamming or confusion due to high-power pulses from other Pulsed radars.

FSK-CW disadvantages:

- Invisible targets in the direct path of the radar.
- Target range is computed based on the difference in phase shift for two consecutive frequency hops. This makes the system subject to phase noise.
- The CPI needs to be large enough to avoid range ambiguity.

The disadvantages posed by both Pulsed Doppler and FSK-CW radars mandate a type of radar which does not suffer the same, and is apropos in the automotive scenario. LFM CW radar overcomes these disadvantages with:

- No theoretical limit to range resolution and better short range detection.
- Reduced effects of clutter and atmospheric noise.
- Lower power rating than Pulsed radar.
- Less effects of phase noise.
- More resistance to interference from other similar radars in the vicinity.

- No theoretical blind spots.
- Resistance to jamming (frequency modulation is a common tool in ECCM – Electronic Counter-Countermeasures – to overcome jamming effects)

This qualitative comparison warrants the use of LFM CW for the MEMS radar sensor under development, especially for long range radar (LRR) application.

Apart from the distinction in operating principles of different radar types, there are design issues common to all types in general. These are:

- Beamforming technique
- Frequency generation, tuning and linearity
- Platform for implementation of radar signal processing algorithm

2.1.2 Beamforming with Phased Array Antennæ

2.1.2.1 Microelectronic Beamforming

The primitive approach in communications to rotate a scanning beam over an azimuthal angle was to physically rotate a directional antenna mounted on a gyrating platform. To reduce the delay and power usage inherent to this mechanically rotating part, solid-state antennæ with microelectronic beamforming were developed. Beamforming is an aspect of wireless systems where directional signal transmission and/or reception are desired. In other words, beamforming can be referred to as a form of spatial filtering [7]. It is a technique applied in both transmission and reception, depending on the application. In communications, high directivity is desired in the direction of the signal source for a low-noise high-fidelity link to be established. In radar

systems, beamforming allows a means of electronic steering of a narrow scanning beam to detect targets with higher angular resolution.

Essentially, beamforming with phased array antennæ – which is the type of antenna used in the radar system under development – is the ability to simulate a large directional radiation pattern using a set of smaller non-directional radiating antennæ [4]. A beamformer does this by adjusting the amplitude and phase of the radiation at every radiating element and forming a pattern of constructive interference in the desired direction and destructive interference elsewhere.

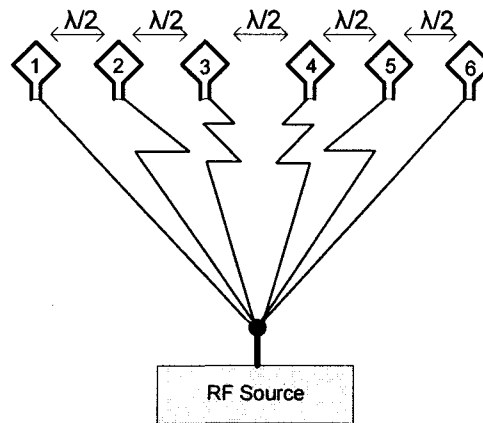


Figure 2.3: Six patch array antenna of radiating elements.

Figure 2.3 illustrates the concept of beamforming using an array of 6 radiating elements (or patches). Each element is separated by a distance of $\lambda/2$, where λ is the wavelength of the waves being radiated. The RF source passes an identical signal down the 6 different paths leading to the radiating patches. The RF signal travels different distances to reach the radiating patch, which essentially creates a different path delay for the signal. This delay manifests itself as a phase shift in the original signal. These phase shifted RF signals are radiated and produce an interference pattern which adds up to a main lobe and possibly some sidelobes, with nulls occurring in intermittently.

Figure 2.4 shows the radiation pattern of a 3 patch array antenna and a 6 patch array antenna. As a design rule for linear patch array antennæ, a higher number of patches produce a more directional and sharper beam.

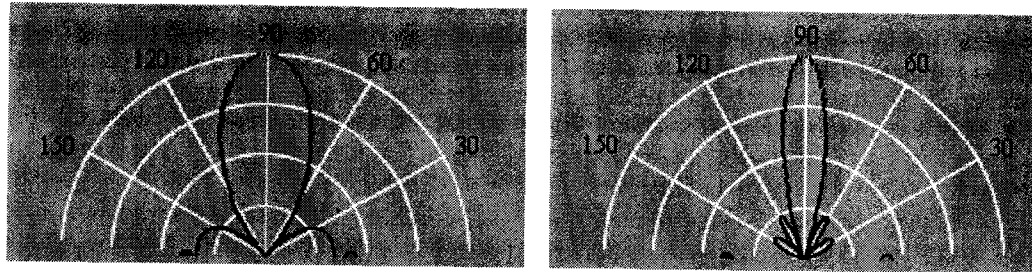


Figure 2.4: Radiation pattern for 3 patch array (left) and 6 patch array (right). (The figures are extracts from graphs generated using Java applets distributed with *Fundamentals of Applied Electromagnetics 6th Edition* by Ulaby, Michielsson, Ravaioli.)

Beamforming involves both the generation of a directional pattern as well as steering of the main lobe over the azimuth and also the elevation angles. Microelectronic beamforming can be categorized into two main types:

- Analog Beamforming
- Digital Beamforming

2.1.2.1.1 Analog Beamforming

Figure 2.5 illustrates the general layout of an analog beamformer that can be implemented using analog RF circuit components. After generation, an RF signal is fed to the radiating elements after altering the phase using electronically tuned phase shifting elements and constant weights to form a directional beam. An analog triangle or sine wave generator can be used to continuously vary the phase shifting elements, which effectively causes the beam to be steered [4]. Bosch LRR2 automotive radar has been developed to operate using this analog beamforming concept.

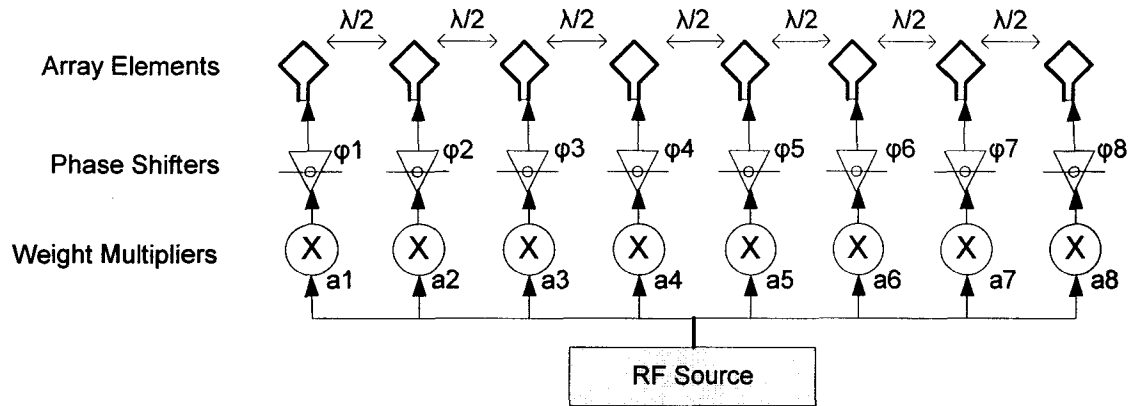


Figure2.5: Analog beamformer with power and phase adjustment to rotate the beam.

2.1.2.1.2 Digital Beamforming

Instead of using analog circuits to control the phase and power of the signal fed at every antenna patch, digital control offers the following advantages [5-6]. Denso bistatic 77 GHz LRR and Toyota CRDL 77GHz LRR radar both operate on a digital beamforming principle.

- Improved beamformer control: The phase at individual patch or sub-array level can be accurately controlled. The beam shape and size can be controlled electronically to any degree resulting in a more selective beamforming.
- Switching between multiple beams: Switching between beams of different widths by enabling or disabling array elements or generating distinct beams using separate sub-arrays.
- High precision control of phase shift and power: DSPs or FPGAs are powerful tools for high-resolution high-speed precise digital control of antenna components. These digital circuits can be used to drive high power antenna circuits with improved control and precision as compared to conventional analog implementations.

Digital beamformers require memory blocks, adders and multipliers as system building blocks. These digital components are available in high-speed on-chip resources in FPGAs which typically operate at clock frequencies of 550 MHz (e.g. Virtex 6 FPGA by Xilinx). This makes digital beamforming techniques more feasible and efficient. Digital beamforming does require more signal conditioning prior to digital processing. If the signal frequency is too high (greater than 100 MHz, say) direct sampling is not possible. To overcome this issue, the signal needs to be down-converted to an intermediate frequency (IF) using an RF mixer which can be sampled. Various beamformer architectures are available in [3-4].

2.1.2.2 Rotman Lens Beamformer

A Rotman lens [1] is a passive device that can enable a beamforming and beam steering capability with out any microelectronic signal processing as needed by analog or digital beamformers. During operation, the electromagnetic property of a dielectric cavity is exploited to realize a directional in-phase signal.

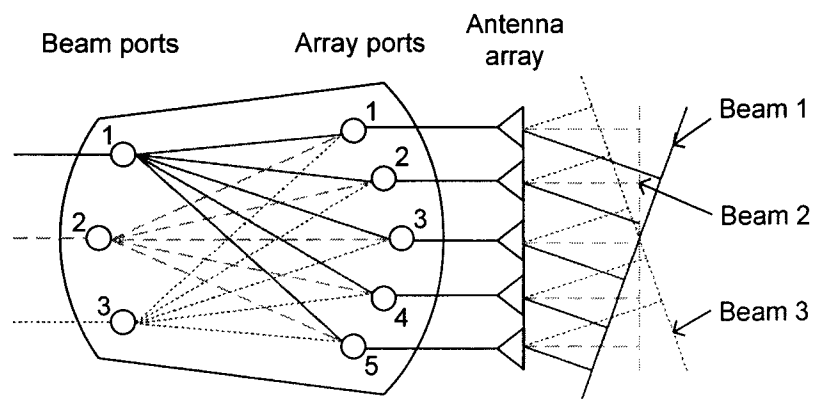


Figure 2.6: Schematic of the intrinsic beamforming capability of the Rotman lens [1].

The body of the Rotman lens has beam ports on one side and array ports on the opposite side. The central beam (beam port 2 in Figure 2.6) guides the input signal through channels of equal length to the array elements, creating a forward-facing beam. On beam ports 1 and 3 the input signal travels through different path lengths to the antenna patches, thus undergoing phase shift leading to the beam being steered as shown [8]. Typical Rotman lenses are large and are realized using microstrip substrates like Duroid 5880 or dielectric material filled waveguides. Figure 2.6 illustrates the schematic representation of a Rotman lens. Recently a novel MEMS based air-filled waveguide type Rotman lens has been reported [1].

2.1.3 Direction of Arrival Estimation using Phased Array Antennæ

Direction of Arrival estimation or DOA using classical approach required a gyrating radar antenna that would pin-point the exact angle of a target. However, with solid-state antennæ and beamforming, DOA estimation requires digital processing. With higher clock speeds and parallel processing capability of FPGAs and multi-core DSPs, this digital processing does not pose any limitations. Two techniques have been compared in literature [30]: DOA estimation by spatial frequency and DOA estimation by phase difference.

DOA by the spatial frequency: this method is limited by the number of array antenna elements. A larger number of array elements are required for better accuracy and precision. It is shown in [30] that with 10 elements the DOA estimation can be unreliable using this method. For reliable and accurate measurement of target angle a 128 element array is then used, which in real-life applications is impractical and would increase hardware.

DOA by phase difference: this method is proposed as a superior method to the spatial frequency method, and requires fewer antenna elements for good precision DOA measurement. The technique is described as follows:

- Let there be n patch array elements in the antenna. Sample each array element individually at the same time and process the samples through 1-D FFT to obtain the spectral power distribution for detected targets.
- Let there be m peaks in the FFT spectrum of each of the n element corresponding to m targets. Compute the phase of each complex peak and produce a matrix $[\Phi_{i,j}]$ for $i = 1, 2, 3 \dots m$ and $j = 1, 2, 3 \dots n$.
- Compute the phase change for every row of $[\Phi_{i,j}]$, taking $\Phi_{i,1}$ as the primary phase for the i 'th target, and obtain a new phase difference matrix $[\Psi_{i,j}]$ with the same definitions for indices i and j .
- Obtain the average of each row pertaining to a single target from $[\Psi_{i,j}]$, thus obtaining an array of averages $[\bar{\Psi}_i]$. Use the average computed, along with the observed wavelength λ_i for the particular target (obtained from the peak frequency resolution in the previous steps) and the known distance between individual array elements d , to compute the angle of arrival using the equation:

$$\bar{\Psi}_i = 2\pi \frac{d}{\lambda_i} \sin \theta_i \quad (2.5)$$

Where θ_i is the angle of the i 'th target.

2.1.4 Frequency Generation, Tuning and Linearity

Generation of the RF radar signal is typically accomplished by means of a voltage controlled oscillator (VCO). In FSK-CW or simple Pulsed Doppler radar a constant frequency is broadcast over a CPI or pulse respectively, however for LFM CW a frequency chirp is realized by tuning the VCO using a triangular modulating signal. This gives rise to linearity considerations in the transmitter, which arises due to a non-linear rate of change of output frequency per unit change in tuning voltage. Linearity of a VCO is defined as follows [13].

$$\text{Linearity, } \delta = \frac{|f_e(t)|_{\max}}{B} \quad (2.6)$$

Here, $|f_e(t)|_{\max}$ is the maximum absolute value of $|f_e(t)|$, which is the error or difference between the ideally expected output frequency $|f_{\text{ideal}}(t)|$ of the VCO and the actual output frequency $|f_{\text{actual}}(t)|$ of the VCO, and B is the bandwidth over which the VCO is being tuned.

$$f_e(t) = f_{\text{ideal}}(t) - f_{\text{actual}}(t) \quad (2.7)$$

Due to material imperfections, stray capacitance and inductance in high frequency RF circuits, VCOs tend to have a non-linear frequency vs. voltage curve as in Figure 2.7. These drifts in the output frequency gradient cause phase errors in an LFW-CW radar among others [2].

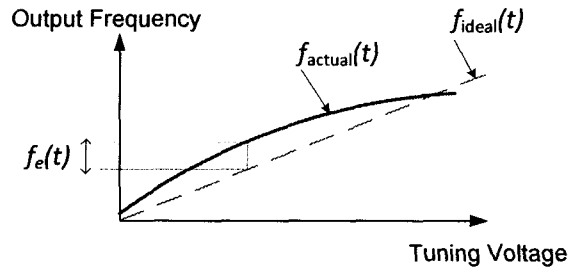


Figure 2.7: Non-linear frequency response of a typical RF VCO.

2.1.5 Selecting the Development Platform for the Radar Signal Processing System

The transmitter incorporates radar signal generation, tuning and linearity control. These aspects become critical in LFM CW radar due to the requirement of highly linear frequency sweeps. In LFM CW radar the signal generation and sweep modulation can be accomplished using analog or digital modulation. Analog PLLs or Phase Locked

Loops containing a VCO were used in early CW systems, however were overtaken by digital systems with better frequency response, excellent linearity, easier design and improved performance in noise [2].

In digital implementation of a radar transmitter the control and modulation algorithm can be based on a Digital Signal Processor (DSP) or a Field Programmable Gate Array (FPGA). Due to their highly parallel nature, ability to run several tasks simultaneously without stalling other tasks, and on-chip resources (such as RAM blocks, LUTs, fast DSP multipliers) FPGAs are the preferred solution for digital signal processing. The use of FPGAs for DSP has been boosted by the wide availability of fully customizable IP cores from various providers spanning many application areas such as DSP, automotive, communications, computer networking and bus interfaces among others [14]. According to benchmark results presented in [21], [22] and [28], the latencies for a 2048-point FFT on a 32-bit Intel Core 2 Duo @ 3 GHz, an Analog Devices ADSP-BF53x and a Texas Instruments TMS320C67xx are tabulated in Table 2.1. Comparison of these with an FPGA at a much lower clock frequency demonstrates the power of FPGAs as modern-day high-bandwidth DSP solutions.

Table 2.1: Speed Comparison of a typical FPGA versus a general purpose Dual Core Processor and a Digital Signal Processor

Manufacturer	Part Name	Clock Frequency (MHz)	2048-point FFT Latency (μs)	Number of Clock Cycles
Intel	32-bit Core 2 Duo	3000	37.55	112650
Analog Devices	ADSP-BF53x	600	32.40	19440
Texas Instruments	TMS320C67xx	600	34.20	20520
Xilinx	Virtex-5 FFT Core	200	39.60	7920

Even with a low clock frequency of 200 MHz the FPGA has comparable speed performance compared to the other processors at higher clock rates. Power consumption of a digital circuit is proportional to the total gate-level switching required to compute a particular result: the higher the clock frequency and required clock cycles, the greater the amount of switching, and thus the higher the power consumption. Given the automotive scenario, FPGAs offer a desirable combination of speed and power efficiency.

Furthermore, to deal with possible VCO non-linearity FPGAs can be used to implement a DDS or Direct Digital Synthesis algorithm. DDS is a method of creating arbitrary yet repetitive waveforms using a RAM or LUT, a counter, and a DAC, components that are readily available on FPGA platforms. DDS promises optimal linearity in frequency sweeps, precise frequency tuning, and excellent phase error recovery [2].

Based on the analyses presented here, the development platform of choice for this thesis is FPGA technology. A successful implementation of a radar sensor transmitter and receiver based on FPGA technology is the Radar Digital Unit (RDU) of South African Synthetic Aperture Radar II (SASAR II) in May 2004, by the University of Cape Town [22].

2.1.6 State-of-the-Art in Automotive Radar

Research on automotive radar began as early as the 1950s, although commercialization only became possible in the late 1990s with the launch of various manufacturers introducing the early versions of collision warning, parking assist and adaptive cruise control radars [23]. Daimler-Chrysler launched their first “autonomous cruise control” radar in 1999 with Mercedes S-class models, marketed as “Distronic”. Further developments of 77 GHz LRR and 24 GHz UWB SRR were launched as a combination of cruise control, parking assist and collision warning systems, marketed in

2003 as “Distronic” and a second version marketed as “Distronic Plus” [24]. Figure 2.8 shows the Daimler-Chrysler automotive radar application portfolio, which has set an industry-wide standard on radar systems. The Distronic Plus system, which includes 1 LRR at 77 GHz and 4 SRRs at 24 GHz, is shown in Figure 2.9.

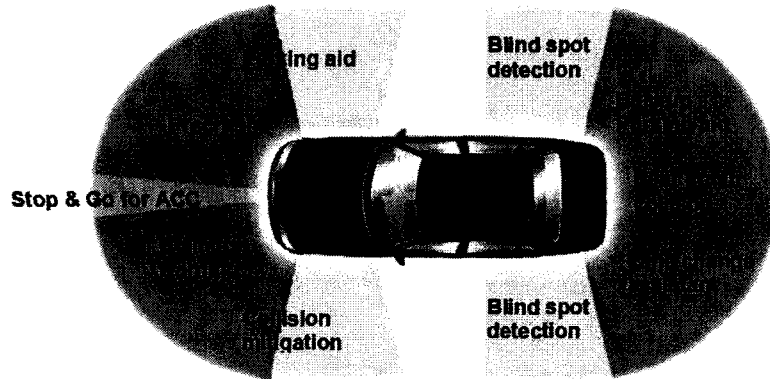


Figure 2.8: Radar applications in the automotive industry © Daimler-Chrysler 2005.

One of the promising development initiatives was the German government funded Daimler-Chrysler research project named KOKON [25]. The main outcomes of this research were development of cost-effective 76 – 81 GHz automotive radar systems, vehicular integration conceptualization, and standardization of the 76 – 81 GHz radio frequency band for automotive applications. The KOKON project is a successor to the RoCC project, which is a joint-venture of Daimler-Chrysler, BMW, Bosch, Continental and Infineon [25].

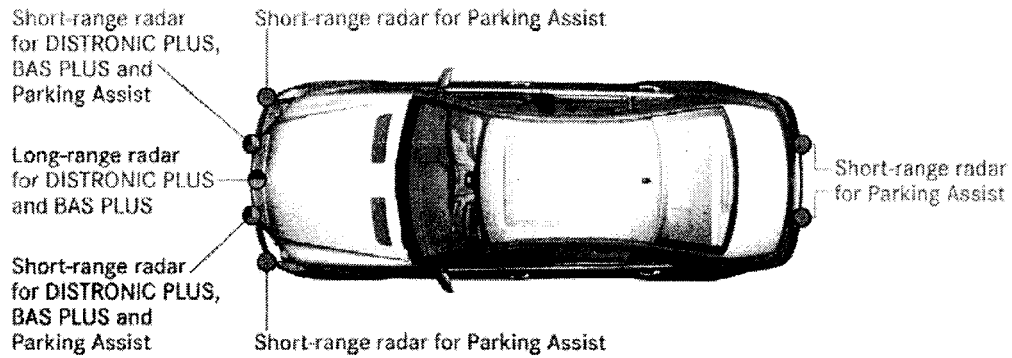


Figure 2.9: Distronic Plus by Mercedes-Benz © Daimler-Chrysler 2005.

The RoCC project essays a study of automotive radar vehicular integration and live testing, investigation of complete sensor packaging including DSP unit(s), evaluation of automotive radar beyond 100 GHz, SMD packaging of RF MMICs, feasibility study for 500 GHz UWB automotive radar based on LFM CW technique, improvement of energy efficiency and multi-mode multi-range self-calibrating sensors. The lattermost objective is currently one of the most pursued topics in automotive radar; recent self-calibrating dual-band MMICs such as those presented in literatures [26] and [27] propose the capability of switching between 24 GHz and 77 GHz SRR, MRR and LRR using the same MMIC RF radar frontend.

The MEMS Rotman lens and MEMS RF switch combination central to this thesis can be used in conjunction with a reconfigurable patch array antenna in order to accomplish SRR, MRR and LRR beamforming using the same hardware. The control of such a system would be easily realizable digitally by means of the FPGA control algorithm.

Table 2.2 lists some of the commercially available automotive radar systems by different developers and their operating specifications. The AC3 by TRW Automotive is a third-generation adaptive cruise control radar operating at 77 GHz, capable of scanning targets up to 250 meters distant [20]. Table 2.3 shows a list of the previous generation

of radar systems and their capabilities as listed by a report from Fujitsu presented in reference [16].



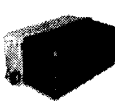




Table 2.2: Commercially available new generation of automotive radar systems [23]

Developer	Operation Frequency	Radar Type	Range (m)	Relative Velocity (km/h) ¹	Field of View	Refresh Time (ms) ²
TRW Automotive	77 GHz	Pulsed Doppler	1 - 250	±220	±8°	50
Delphi	76 GHz	Pulsed Doppler	1 - 174	-360 to +90	±10°	50
Denso	77 GHz	FMCW	2 - 150	±200	±20°	50
Bosch	77 GHz	FMCW	0.5 - 250	-500 to +250	±30°	50

¹ Negative sign means velocity of approaching target; positive sign means velocity of receding target.

² Processing times are not included.

Table 2.3: Previous generation of automotive radar systems – listing by Fujitsu [16]

Manufacturer	Our company	ADC	Delphi	Bosch	Honda elesys	Denso	Hitachi
Appearance							
External Dimensions (mm)	89×107×86	136×133×68	137×67×100	91×124×79	123×98×79	77×107×53	80×108×64
Modulation Method	FM-CW	FM Pulse	FM-CW	—	FM-CW	FM-CW	2-frequency CW
Detection Range	4m to 120m or greater	Approx. 1m to 150m	Approx. 1m to 150m	2m to 120m or greater	4m to 100m or greater	Approx. 2m to 150m	Approx. 1m to 150m
Horizontal Detection Angle	±8°	Approx. ±5°	Approx. ±5°	±4°	±8°	±10°	±8°
Angle Detection Method	Mechanical Scan	Beam Conversion	Mechanical Scan	Beam Conversion	Beam Conversion	Phased Array	Monopulse
EHF Device	MMIC	GUNN	GUNN	GUNN	MMIC	MMIC	MMIC

One of the most recent systems from Table 2.2 is the Bosch LRR3 (as marketed) which was launched in September 2009 on the Porsche Panamera 2010 model. One of the claims of Bosch LRR3 is being the world's smallest radar sensor package at 74mm x 77mm x 58mm. The MEMS radar system being developed at the University of Windsor has close to half the dimensions at 30mm x 40mm x 10mm owing to the compact MEMS Rotman lens beamformer and antenna design.

These state-of-the-art automotive radar systems provide a target for this thesis and help set the aims for the speed and efficiency of the radar signal processing algorithm presented in this thesis.

2.1.7 Recent Work Done in FPGA-based LFM CW Digital Signal Processing

A recent study, in 2009, on FPGA-based LFM CW radar signal processing algorithm has been presented in [28], where a Xilinx Virtex-II Pro FPGA at 50 MHz has been employed. For a radar cycle (or refresh) time of 60ms the developers have used a sampling time of 1240 μ s and a processing time of 1250 μ s per frequency sweep. The spectral analysis is first done using an FFT core, after which the software processing for peak detection and range-velocity computations has been done using a soft-processor MicroBlaze core by Xilinx. The developers quote a usage of 4100 DSP48 slices and 35% of on-chip Block RAM usage, and several Xilinx IP cores to optimize timing requirements.

This work is given due consideration in light of the aims of this project, and a faster signal processing algorithm would be a key outcome of this thesis.

CHAPTER 3:


REQUIREMENTS FOR THE TARGET FMCW SYSTEM

This chapter reviews the relevant mathematical models associated with FMCW radar to process the reflected radar signal to determine the range and the velocity of the targets. The range and velocity equations are reviewed for the automotive radar algorithm for both relatively stationary and moving targets. Necessary mathematical process blocks have been identified and their characteristics are studied to determine the operating parameters. Several other issues such as atmospheric attenuation, effects of temperature, false alarm rate, removal of clutter, types of radar targets, and have also been reviewed. The gathered knowledge has been used in the next chapter to develop a robust highly accurate control and signal processing algorithm for the MEMS Rotman lens based radar.

3.1 System Requirements Identification

In [19], the requirements for state-of-the-art automotive long range radar have been identified in Table 3.1. Daimler-Chrysler has specified the operating parameters of the next generation of long range radar for automotive applications. The parameters key to the work presented in this thesis are range coverage, range accuracy, relative velocity coverage, velocity accuracy, and cycle time.

Table 3.1: The next generation of Long Range Radar (from Daimler-Chrysler)

			DAIMLERCHRYSLER		
Specification Next Generation LRR					
	Unit	DC-Spec		Unit	DC-Spec
Range <i>!</i>	m	1 ... 200	Power	W	< 6
Range Accuracy	m	$\pm 0,25$	Transmit Power	mW	< 10
Velocity Range <i>!</i>	km/h	-100 ... 260	Sensor Size (WxHxD)	mm	100*100*50
Velocity Accuracy	km/h	$\pm 0,5$	Sensor Weight	g	< 500
Opening Angle Horizontal	deg	20	Operation Temperature	°C	-40 ... 85
Angle Resolution Horizontal	deg	not def.	Storage Temperature	°C	-40 ... 105
Alignment Offset Horizontal	deg	± 3 additiv	Mounting Position Offset Horizontal	cm	± 80
Opening Angle Vertical	deg	4,5	Mounting Position Offset Vertical	cm	> 50
Angle Resolution Vertical	deg	not def.	Misalignment Detection <i>!</i>	deg	< 0,1
Alignment Offset Vertical	deg	± 2	Automatic Adjustment Horizontal <i>!</i>	-	yes
Cycle Time	ms	< 50	Automatic Adjustment Vertical <i>!</i>	-	yes
Interface	-	CAN	Blockage Detection Time <i>!</i>	sec	< 1
			77 GHz Interference Safety	-	yes

16

Based on the next-generation specifications in Table 3.1 [19], the target radar signal processing algorithm need to meet at least the following performance specifications:

1. Range: 200 meters
2. Range accruay: 0.25 meters
3. Relative velocity: -100 to 250 km/h
4. Velocity accuracy: ± 0.5 km/h
5. Cycle time: < 50ms

3.2 Selecting the Required FMCW Waveform

FMCW is the type of radar for which the algorithm presented in this thesis has been designed. The use of FMCW as the radar technique of choice has been justified in Chapter 2. FMCW waveforms – note that LFM CW is a special case of FMCW where the modulating waveform is linear – exist in various standard implementations: sine wave, saw-tooth and triangular. Figure 3.1 illustrates these three types.

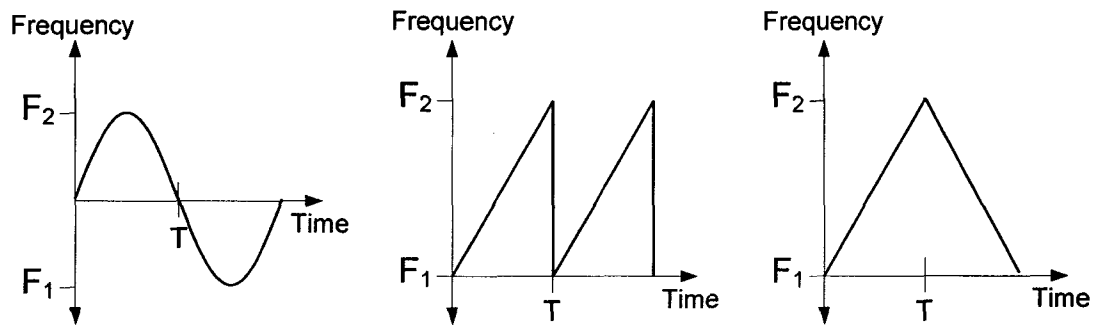


Figure 3.1: FMCW waveforms left to right: Sine, Saw-tooth and Triangular. (The period T is equivalent to CPI in Chapter 1)

Sine wave modulation is seldom used in contemporary FMCW systems due to the extra latency added in computing and adjusting sine wave coefficients. Also, sine wave modulation has less tolerance for VCO non-linearity as compared to the linear variants of FMCW waveforms. However, at lower operating frequencies (few hundred MHz) sine wave modulation is realizable and offers easy analog modulation without the need for digital waveform generation.

The saw-tooth waveform only has a positive frequency sweep, and is thus easier to control and tune electronically. However, the computation of range and velocity relies on phase calculation of the beat frequency over a minimum of 2 sweeps, and thus

requires more processing when compared to the triangular waveform. Range and velocity may not be determined simultaneously.

The favoured FMCW waveform is the triangular waveform due to the ability to determine both range and velocity. The difference in up sweep and down sweep frequencies is equivalent to twice the Doppler shift of the target, thus allowing simultaneous range and velocity computation. Another benefit of the triangular waveform is that the different sweep directions make the system more resistant to stationary clutter and jamming signals by having a more dynamic instantaneous frequency.

3.3 Linear Frequency Modulated Continuous Wave Radar

The LFM CW technique relies on a linear frequency sweep (or chirp) over a carefully selected bandwidth and measures the received beat frequency f_b from all targets (and false targets or clutter) that fall in the field of view of the radar beam. As discussed, triangular modulation is chosen for this thesis. The beat frequency is defined as the instantaneous difference in the frequencies of the transmitted and received radar signal:

$$f_b(t) = f_t(t) - f_r(t) \quad (3.1)$$

The bandwidth and chirp period (termed *CPI* in Chapter 2 and *T* hereon) are critical parameters in determining the refresh rate, range resolution and velocity resolution of the targets. A larger sweep bandwidth improves range resolution, which is a desirable effect. However, the limiting factor to higher bandwidth is the linearity of the VCO that is used to generate the radar signal. Figure 3.2 shows the LFM CW transmitted and received signals illustrating the beat frequency obtained in the up (positive) and down (negative) frequency sweeps.

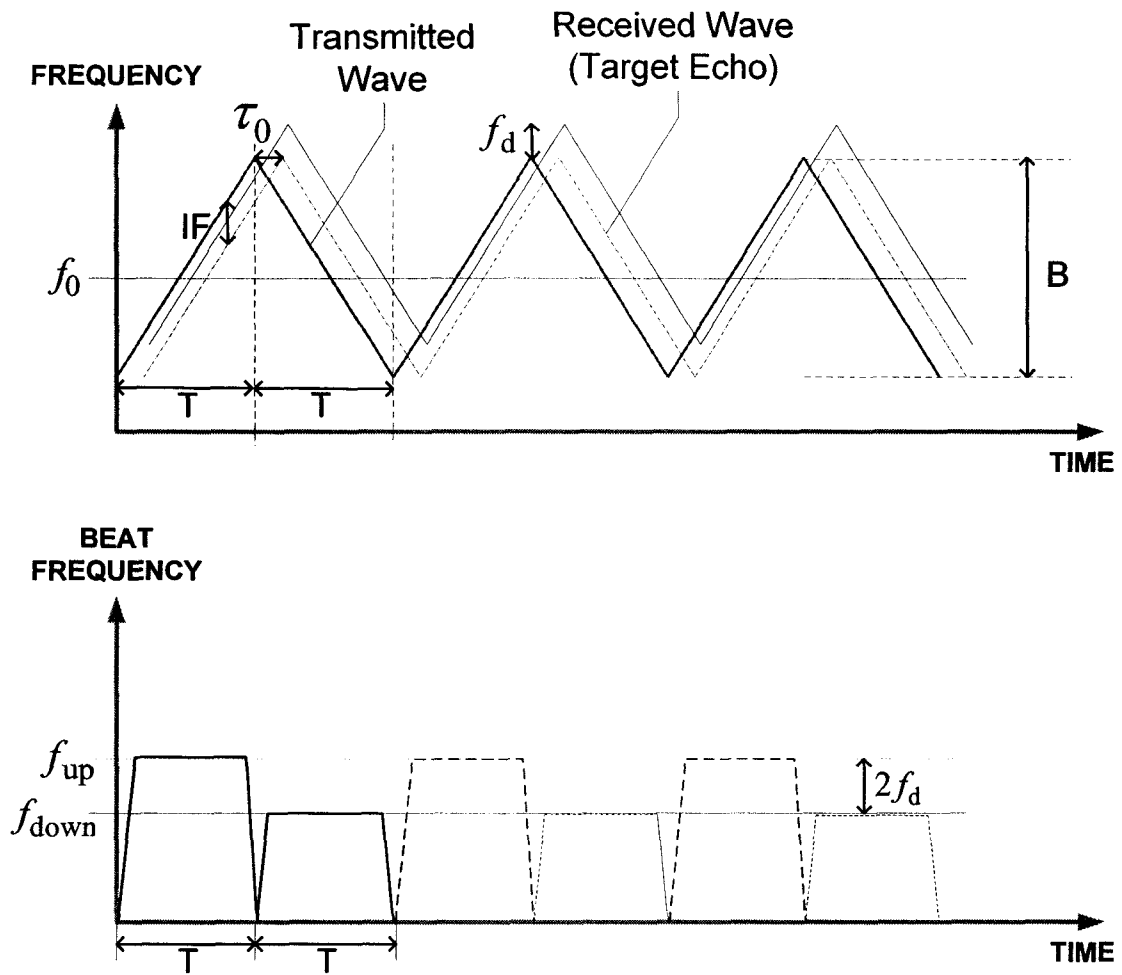


Figure 3.2: LFM CW Transmit, Receive and Beat frequency.

Here,

- τ_0 = round trip delay time for the signal to be received from the target
- f_d = Doppler shift due to relative target velocity
- f_0 = starting frequency for operation bandwidth
- B = operation bandwidth
- T = sweep duration (same for both up and down sweeps in this thesis)
- f_b = beat frequency or intermediate frequency
- f_{up} = up sweep beat frequency
- f_{down} = down sweep beat frequency

3.3.1 Derivation of Range and Velocity for LFM CW

The following is a concise step-wise derivation of the range and velocity equations for LFM CW radar:

Let $f_t(t)$ = transmitted radar signal

$f_r(t)$ = received target echo signal

$k = \frac{B}{T}$ = rate of change of frequency over a single sweep

3.3.1.1 First case: Relatively Stationary Target

A relatively stationary target is a target with zero relative velocity compared to the radar sensor or host vehicle, and as such does not contribute to any Doppler shift of the received echo signal. The transmitted radar signal can be defined as a complex sinusoid with a base frequency of f_0 modulated over a bandwidth of B Hz [29].

$$f_{t1}(t) = \exp\left(j2\pi\left(f_0 t + \frac{1}{2}kt^2\right)\right) \quad (3.2)$$

The modulation of the transmit signal is evident from the frequency term in equation (3.2) above. The term $\frac{1}{2}kt^2$ adds a fraction of the total sweep bandwidth depending on the instantaneous time t .

The received echo signal can be defined as a complex sinusoid delayed by a round trip delay time τ_0 .

$$f_{r1}(t) = \exp\left(j2\pi\left(f_0(t - \tau_0) + \frac{1}{2}k(t - \tau_0)^2\right)\right) \quad (3.3)$$

Multiplying in time (or mixing) the transmitted and received signals, and ignoring the high frequency component in the mixer output, produces the beat or intermediate frequency of interest. In the case of a relatively stationary target, beat frequencies for both up and down sweep are identical and can be expressed as:

$$f_{b1}(t) = f_{t1}(t) \otimes f_{r1}(t) = \exp\left(j2\pi\left(f_0\tau_0 + kt\tau_0 - \frac{1}{2}k\tau_0^2\right)\right) \quad (3.4)$$

Differentiating the phase of the beat signal in (3.4) w.r.t. time t gives the instantaneous beat frequency that is directly proportional to the range of the target.

$$f_{up1} = \frac{d\left(f_0\tau_0 + kt\tau_0 - \frac{1}{2}k\tau_0^2\right)}{dt} = k\tau_0 \quad (3.5)$$

Therefore, both up and down sweep beat frequencies are defined for a stationary target.

$$f_{up1} = f_{down1} = k\tau_0 = k \frac{2r}{c} \quad (3.6)$$

Here, r is the range of the target and c is the speed of EM waves in air. Thus for a relatively stationary target the range is computed by taking the average of the up and down sweep instantaneous beat frequencies as follows [29]:

$$r = \left(\frac{f_{up1} + f_{down1}}{2}\right) \times \frac{c}{2k} \quad (3.7)$$

3.3.1.2 Second case: Moving Target

Consider a moving target with velocity v_r relative to the radar sensor or host vehicle. This velocity introduces an additional term in the transmitted and received signals due to the Doppler shift. This Doppler shift is approximated by $f_0 v_r / c$ [29]. The following transmitted signal is generated for the up sweep.

$$f_{t2}(t) = \exp\left(j2\pi\left(f_0 t + \frac{1}{2} k t^2\right)\right) \quad (3.8)$$

The received signal for the up sweep is affected by twice the amount of Doppler shift due to two-way travel of the radar wave, as well as round trip delay as in the case of the stationary target.

$$f_{r2}(t) = \exp\left(j2\pi\left(f_0(t - \tau_0) + \frac{1}{2} k(t - \tau_0)^2 + 2f_0 \frac{v_r}{c}(t - \tau_0)\right)\right) \quad (3.9)$$

Multiplying the transmitted and received signals in time we obtain the beat frequency for the up sweep as given in equation (3.9).

$$f_{b_up}(t) = \exp\left(j2\pi\left(f_0 \tau_0 + \left(k\tau_0 + 2f_0 \frac{v_r}{c} - 2k\tau_0 \frac{v_r}{c}\right)t - \frac{1}{2} k\tau_0^2 + 2\frac{k}{c}\left(v_r - \frac{v_r^2}{c}\right)t^2\right)\right) \quad (3.10)$$

The constant and second order terms in the above equation can be ignored for a stable computation of the instantaneous up sweep frequency by differentiating w.r.t time t .

$$f_{up2} = \frac{d\left(\left(k\tau_0 + 2f_0 \frac{v_r}{c} - 2k\tau_0 \frac{v_r}{c}\right)t\right)}{dt} = k\tau_0 + 2f_0 \frac{v_r}{c} - 2k\tau_0 \frac{v_r}{c} \approx k\tau_0 + f_d \quad (3.11)$$

The above approximation is possible as $2k\tau_0 \frac{v_r}{c} = 2k \frac{2r}{c} \frac{v_r}{c} = 4kr \frac{v_r}{c^2} \ll 1$ for bandwidths under 1 GHz. Larger bandwidths in tens of GHz also produce negligible frequency values for this term, and thus this term can be safely neglected.

During the down sweep, the Doppler shift manifests as a negative entity due to the negative slope of the modulating wave. Note that $f_d < B$. This gives rise to the following beat frequency signal at the receiver of the radar sensor:

$$f_{b_down}(t) = \exp \left(j2\pi \left(f_0\tau_0 + \left(k\tau_0 - 2f_0 \frac{v_r}{c} - 2k\tau_0 \frac{v_r}{c} \right) t - \frac{1}{2} k\tau_0^2 \right) + 2 \frac{k}{c} \left(v_r - \frac{v_r^2}{c} \right) t^2 \right) \quad (3.12)$$

Differentiating (3.12) w.r.t. time t we get the down sweep frequency for a moving target with relative velocity v_r .

$$f_{down2} = \frac{d \left(\left(k\tau_0 - 2f_0 \frac{v_r}{c} - 2k\tau_0 \frac{v_r}{c} \right) t \right)}{dt} = k\tau_0 - 2f_0 \frac{v_r}{c} - 2k\tau_0 \frac{v_r}{c} \approx k\tau_0 - f_d \quad (3.13)$$

From this analysis, the range and velocity of any target for the LFM CW technique can be determined. Adding (3.11) and (3.13) we get

$$f_{up2} + f_{down2} = k\tau_0 + f_d + k\tau_0 - f_d = 2k\tau_0 = 2k \frac{(2r)}{c}$$

$$\text{Hence, range } r = \frac{(f_{up2} + f_{down2})}{2} \times \frac{c}{2k} \quad (3.14)$$

This is similar to the range expression derived earlier for a stationary target.

The relative velocity of the target can be derived by subtracting (3.13) from (3.11) to extract the Doppler shift caused by the target.

$$f_{up2} - f_{down2} = k\tau_0 + f_d - (k\tau_0 - f_d) = 2f_d = 4f_0 \frac{v_r}{c}$$

$$\text{Hence, relative velocity, } v_r = \frac{(f_{up2} - f_{down2})}{4} \times \frac{c}{f_0} \quad (3.15)$$

Given equation (3.15), the actual target velocity can be computed based on knowledge about the host vehicle velocity.

$$\text{Actual target velocity, } v_{target} = v_{host} - v_r \quad (3.16)$$

3.3.2 LFM CW Radar Signal Generation using VCO

A core component in contemporary radar systems is the VCO or voltage controlled oscillator. As the name implies, a VCO is supplied an input analog tuning voltage which translates to a change in internal capacitances leading to a change in generated output frequency. For the LFM CW radar under development the output frequency has been chosen as a triangular chirp, with a positive sweep in frequency following by a negative sweep. This requires a triangular modulating signal, which can be generated using an FPGA with relative ease.

The modulating unit requires an up/down counter that will feed a DAC which will output the tuning voltage to the VCO. The digital counter will count up for the up sweep, and count down back to zero for the down sweep. The refresh rate and resolution of the DAC are important parameters affecting the linearity of the LFM CW frequency chirps. Figure 3.3 shows the radar signal generation method employed in the algorithm presented in this thesis, based on a digital counter implemented in an FPGA.

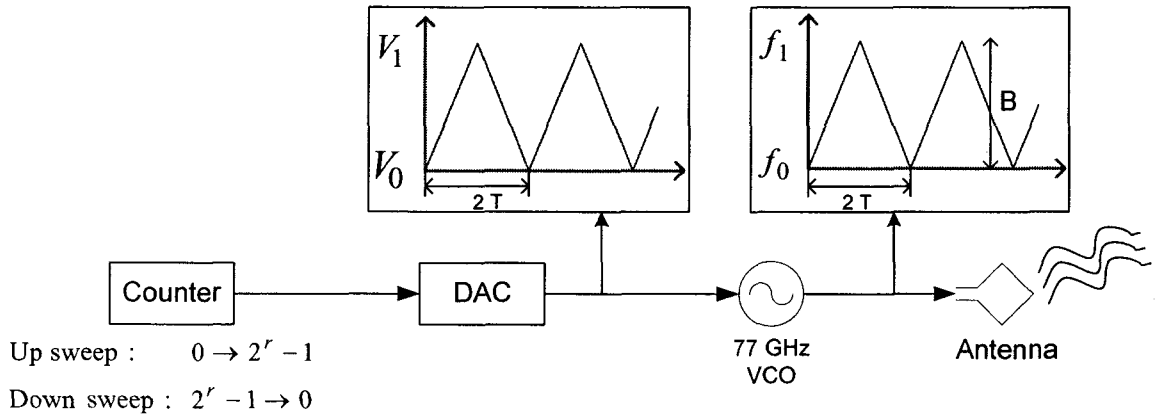


Figure 3.3: FPGA based tuning voltage generation for VCO to produce LFM CW chirps

The modulation results in a time-domain chirp signal resembling the conceptual waveform in Figure 3.4. The up frequency sweep is followed by a down sweep over time.

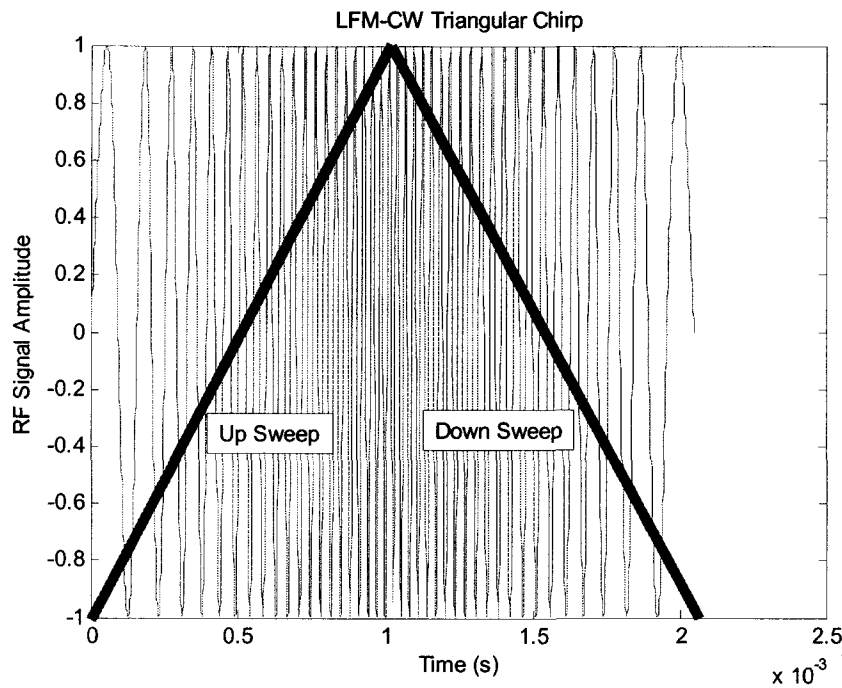


Figure 3.4: Time-domain RF signal showing up (red) and down (purple) frequency chirps for LFM-CW radar.

3.3.3 Received Echo Signal Conditioning for LFM-CW

Prior to digital signal processing of a received target echo, conditioning of the RF signal is required. Conditioning is typically accomplished using analog processing and involves the following components:

1. Low Noise Amplifier: boost the received echo signal using a low noise amplifier to counter atmospheric and hardware attenuation.

2. Mixer: time-domain multiplication (frequency-domain convolution) of the instantaneous received echo signal with the instantaneous radar signal being transmitted. Let $a_r \sin(w_r t)$ and $a_t \sin(w_t t)$ be the received and transmitted signals at any time, then the output of the mixer is the difference and sum of these frequencies. Figure 3.5 shows the conceptual diagram of a mixer.

$$a_r \sin(w_r t) \otimes a_t \sin(w_t t) = \frac{a_r a_t}{2} [\sin((w_r + w_t)t) + \sin(w_r - w_t)t] \quad (3.17)$$

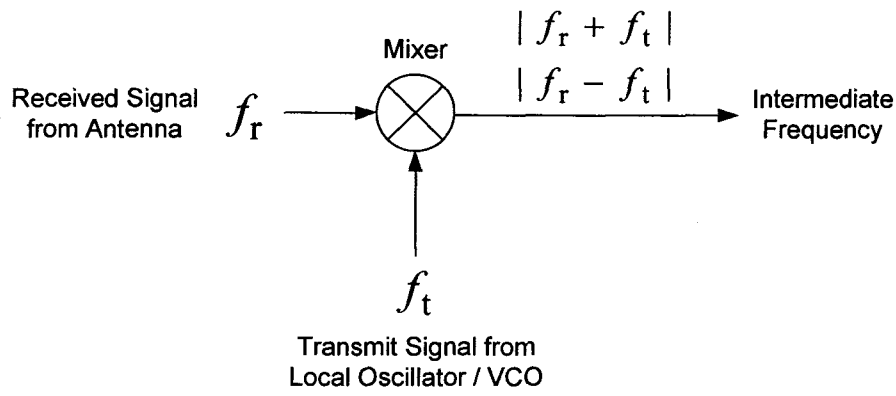


Figure 3.5: Conceptual diagram of an RF mixer.

3. Low Pass Filter: filter out the high frequency component from the output of the mixer and extract the beat frequency of interest, $(w_r - w_t)$.
4. Analog to Digital Converter: sample the IF or the beat frequency slightly above Nyquist rate to avoid aliasing. The ADC is a critical component in determining the efficiency and accuracy of the entire radar signal processing algorithm. The output resolution of the ADC commands the memory usage, speed and precision of range and velocity computation: higher resolution provides lower quantization noise and improved precision at the cost of time and required memory. The sampling rate of the ADC is proportional to the bandwidth the radar system operates at.

3.4 Digital Signal Processing Tools

The following is a list of the major signal processing steps required in a radar system:

1. Time-domain windowing
2. Spectral analysis using the Fast Fourier Transform
3. Constant False Alarm Rate processing

3.4.1 Time-domain Window

After signal conditioning, the data is digitized and available through the ADC, which samples the time-domain beat frequency or intermediate frequency over a restricted length of time t seconds, say. Spectral analysis is done on the time-domain data using the FFT, which assumes that the data consists of an integral number of wavelengths of the signal. However, samples from the ADC seldom contain an exact integral number of wavelengths, and the intermediate frequency in itself is distorted by noise and microwave interference. Sampling by an ADC is equivalent to multiplying a time-domain signal by a rectangular window function. This leads to the formation of spectral noise in the form of leakage [31].

Spectral leakage is caused by the sudden *slicing* of a time-domain signal. For there to be no spectral leakage the signal would have to be sampled over an infinite length of time, which is not feasible. Time-limiting a signal means multiplying it by a rectangular window function, which causes the signal to be non-band-limited, giving rise to power leakage into neighbouring frequencies from the actual frequency of interest. Figure 3.6 illustrates the effect.

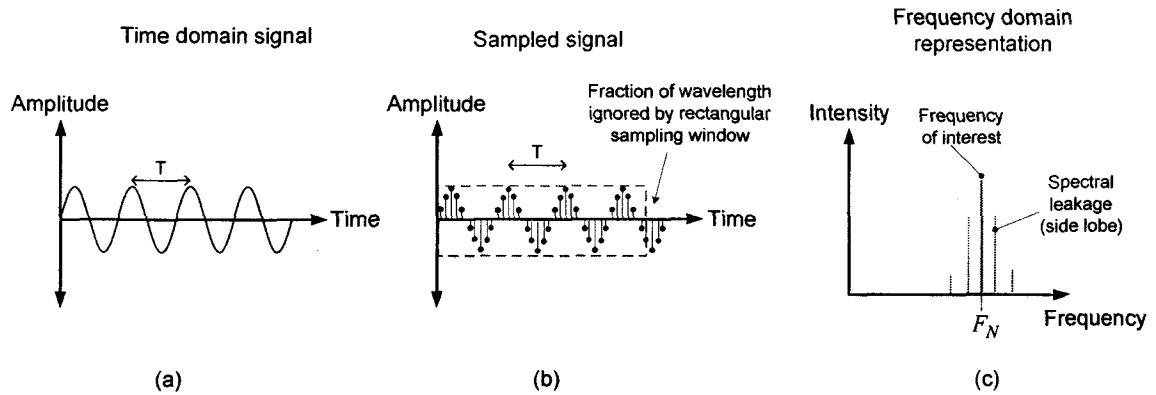


Figure 3.6: (a) Time-domain continuous wave with period T ; (b) Sampled time-domain signal multiplied by a rectangular window through ADC; (c) Spectral leakage due to rectangular windowing where $F_N = 1/T$ is the frequency of interest.

In order to reduce the effects of spectral leakage, different windowing functions have been investigated [31]. An ideal window function is a time-domain function whose energy is band-limited. When multiplied by a time-domain signal, an ideal window function helps focus the energy of the signal and reduce spectral leakage. Although ideal window functions are practically unrealizable, there exist windows that can greatly reduce the sidelobe spectral leakage as well as attenuate frequencies other than the frequency of interest, similar to the action of a filter. Figures 3.7(a), 3.7(b), 3.7(c) and 3.7(d) offer a comparison of some window functions, namely Rectangular, Triangular, Hann and Hamming. The equations for each window are given, where $w(n)$ represents the set of all time-domain coefficients of the window. The n th coefficient is multiplied by the n th time-domain sample.

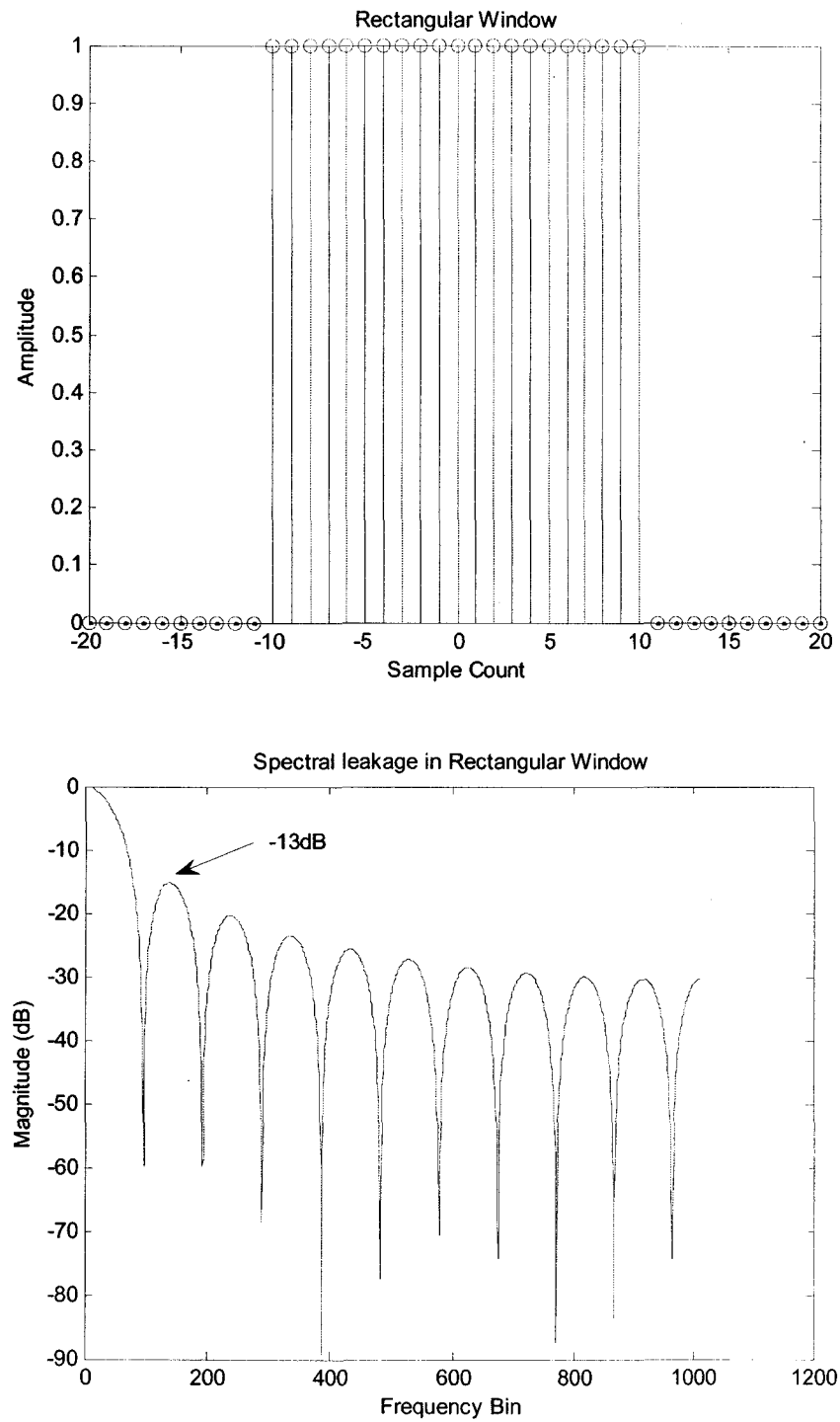


Figure 3.7(a): Time and Frequency domain representations of Rectangular window with 21 points with 2048-point FFT.

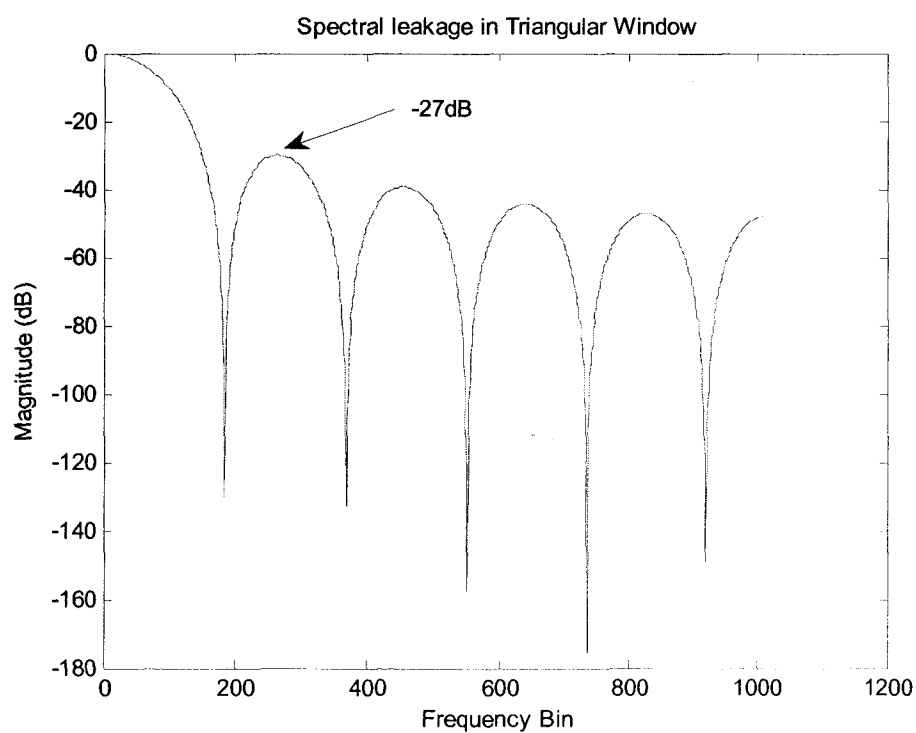
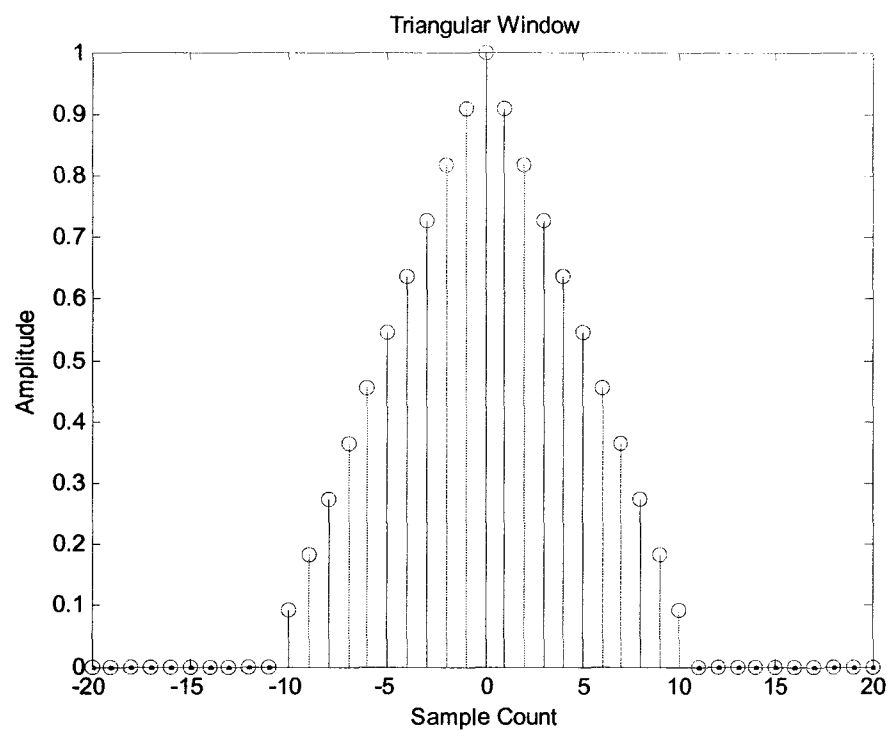


Figure 3.7(b): Time and Frequency domain representations of Triangular window with 21 points and 2048-point FFT.

$$w(n) = 0.5 \left(1 - \cos \left(\frac{2\pi n}{N-1} \right) \right) \quad (3.18)$$

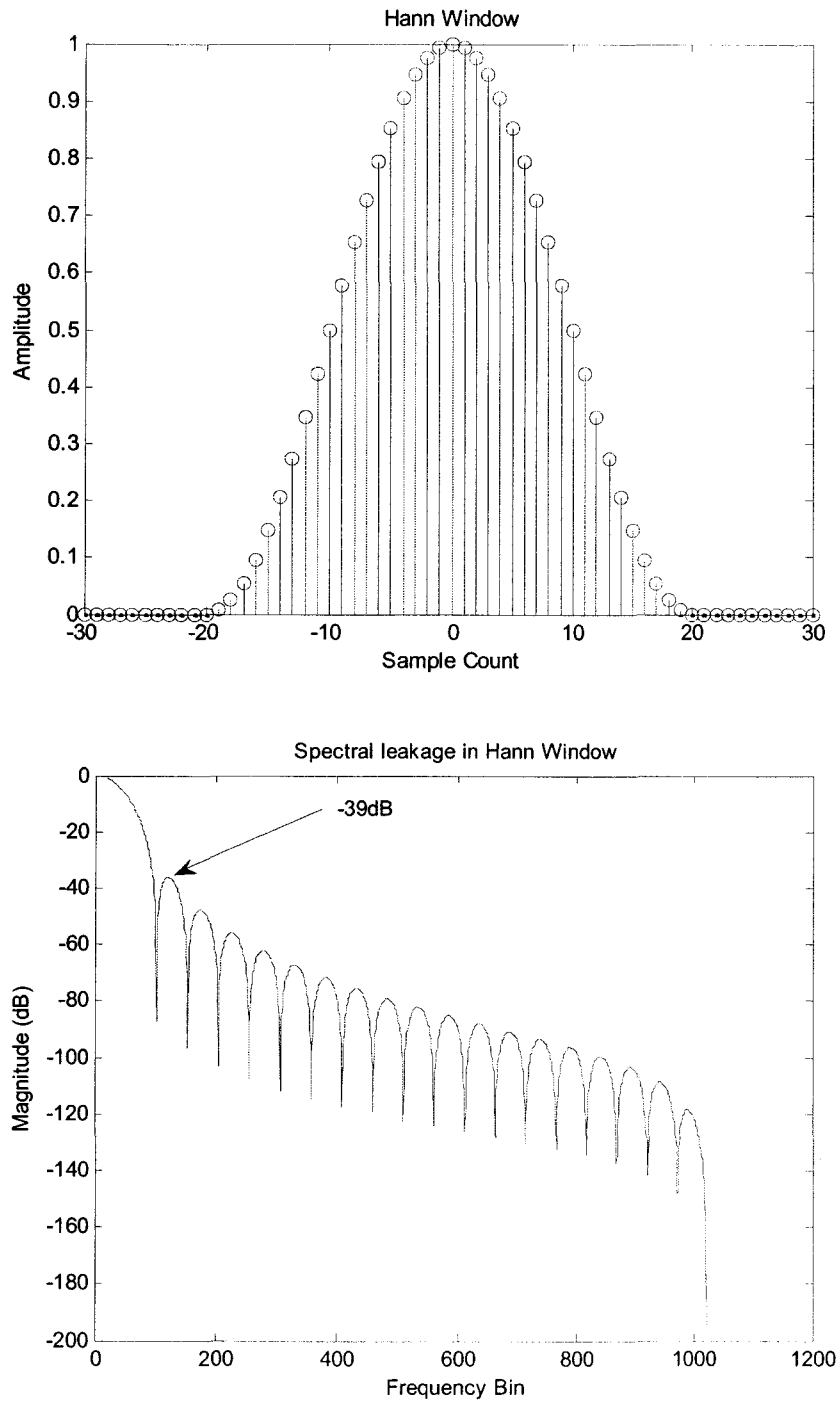


Figure 3.7(c): Time and Frequency domain representations of Hann window with 41 points and 2048-point FFT. In the equation, N = number of time-domain points.

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) \quad (3.19)$$

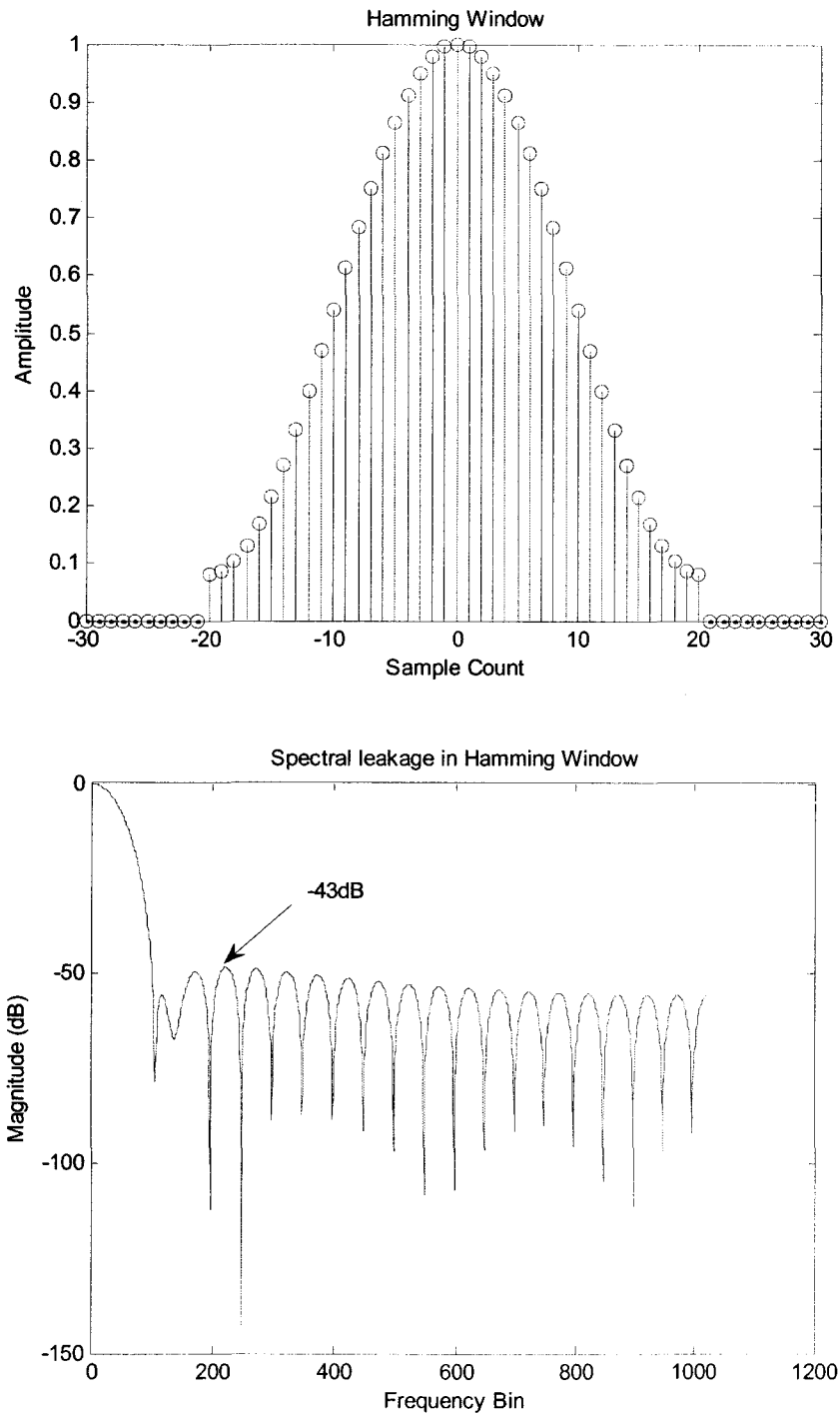


Figure 3.7(d): Time and Frequency domain representations of Hamming window with 41 points and 2048-point FFT. In the equation, N = number of time-domain points.

Table 3.2: Comparison of common Window functions [31]

Window	Main-lobe width (-3dB) (no. of frequency bins)	Highest side-lobe level (dB)	Roll-off rate (dB/octave)
Rectangular	0.89	-13	-6
Triangular	1.28	-27	-12
Hamming	1.36	-43	-6
Hann	1.64	-39	-18
Blackman	1.68	-58	-18

Table 3.2 lists some well-known window functions compared to the default rectangular window. An ideal window function would have a unit main-lobe width, very low side-lobe level and steep roll-off. Looking at the table, the best side-lobe attenuation and roll-off are for the Blackman window; however the main-lobe width is large. This means that the energy of the main lobe is spread across 1.68 frequency bins, and this may be inferred in some systems as spectral leakage as well. The Hamming window is commonly employed in communication systems, although the roll-off is smaller than the rest.

For this project, a Hamming window is chosen. The reasons for this choice are:

1. Excellent side-lobe attenuation.
2. Good accuracy even after truncation to 5 decimal places precision in fixed-point multiplications.
3. Optimal main-lobe width; the poor roll-off can be easily dealt with using CFAR processing (discussed later).

3.4.2 The Fast Fourier Transform

Perhaps the most widely used signal processing routine is the famous FFT algorithm developed by James W. Cooley and John W. Tukey [32]. The algorithm is a re-definition of the Discrete Fourier Transform in which an arbitrary N-point DFT is broken down into smaller DFTs recursively until computationally simple DFTs are possible. This forms the well-known butterfly architecture.

The simplest form of the FFT developed by Cooley and Tukey is the Radix-2 Decimation-in-Time algorithm. The DFT is defined by the following formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j \frac{2\pi}{N} nk} \quad (3.20)$$

Here, k is an integer from 0 to $N-1$, $j = \sqrt{-1}$, N is the total number of time-domain samples, and n is an index. The Radix-2 DIT FFT partitions the DFT into odd and even indices, thus dividing an N-point DFT into 2 DFTs of size $N/2$.

More generally, the Cooley-Tukey FFT algorithm divides an N-point FFT into N_1 FFTs of size N_2 , i.e. $N = N_1 N_2$. First, N_1 DFTs of size N_2 are performed. Secondly, the outputs of the first step are multiplied by weights called twiddle factors. Finally, N_2 DFTs of size N_1 are performed on the result of step 2. If $N_1 < N_2$ the algorithm is called a Radix- N_1 Decimation-in-Time FFT, otherwise if $N_2 < N_1$ the algorithm is called a Radix- N_2 Decimation-in-Frequency FFT.

The Radix of an FFT algorithm affects the speed and complexity of the FFT. The two common algorithms used are Radix-2 DIT and Radix-4 DIT. The Radix-4 DIT algorithm is computationally quicker than the Radix-4 DIT algorithm [33].

$$\text{Number of complex multiplications for Radix-2} = \frac{N}{2} \log_2 N$$

$$\text{Number of complex multiplications for Radix-4} = \frac{3}{8} N \log_2 N = 75\% \text{ of Radix-2}$$

Number of complex additions for Radix-2 = $N \log_2 N$

Number of complex additions for Radix-4 = $N \log_2 N$ = same as Radix-2

Radix-4 thus requires 25% less complex multiplications than Radix-2 DIT algorithm, making it a faster FFT. In this project a Radix-4 FFT is used, the details of which are mentioned in Chapter 4 of this thesis.

3.4.3 Constant False Alarm Rate (CFAR) Processor

The CFAR unit makes it possible for radar systems to operate despite contamination of received signal with noise, interference, clutter and effects of attenuation. The CFAR unit runs an adaptive algorithm responsible for filtering out all spurious spectral peaks in the FFT output and extracts only those peaks that have a high probability of being real targets. The adaptive nature of CFAR processors enables them to identify real target returns in the presence of changing noise and clutter from surrounding false targets. In contrast, non-adaptive detection systems, called *clairvoyant detectors* in [34], use a static threshold to detect valid targets.

After the spectral intensity of a signal is received from the FFT unit, the CFAR unit detects valid targets. For non-adaptive detectors a constant threshold, T_c , is used. Each frequency bin (frequency-domain sample from the FFT) is compared in absolute value to T_c . If $|X[n]| > T_c$ then there exists a valid target at frequency $n \times \frac{f_s}{2}$, where $X[n]$ is the discrete frequency domain representation of the received radar IF, n is an integer between 0 to $(N-1)/2$, N is the FFT size, and f_s is the rate at which the IF is sampled. However, since noise and interference are stochastic and random processes this static threshold can produce high number of false alarms.

CFAR algorithms overcome the short-coming of non-adaptive systems by dynamically changing the threshold T_c according to the amount of noise and clutter

present in the surrounding frequency bins of that target. There are various CFAR algorithms constantly being developed and refined, however two methods have seen widespread application in radar systems: OS-CFAR (Ordered Statistic CFAR) and CA-CFAR (Cell Averaging CFAR) [35-36]. There have been several variations to these basic two CFAR types; however, the details are beyond the scope of this thesis.

A typical CA-CFAR architecture is shown in Figure 3.8 [37-38]. This is the CFAR architecture employed for the system developed in this thesis. The principle of operation of the CA-CFAR unit can be summarized in the following steps:

1. Square law detector removes any possible negative values from the FFT output, in essence computing the absolute value or intensity of each frequency bin.
2. G number of guard bands are left on either side of the CUT (cell-under-test), which help overcome spectral leakage effects.
3. $M/2$ number of cells (or frequency bins) are averaged on either side of the guard bands. Let $avgL$ be the average of the left hand side $M/2$ cells, and $avgR$ be the average of the right hand side $M/2$ cells. The index k ranges from 1 to $M/2$.
4. The average of $avgL$ and $avgR$ is computed and multiplied by a predetermined constant K to obtain the dynamic threshold T_c . The value of the CFAR parameter K is determined by the following equation:

$$K = P_{fa}^{-\frac{1}{M}} - 1 \quad (3.21)$$

Here, P_{fa} is the acceptable preselected probability of false alarm and M is the depth of the CFAR averaging [40].

5. The CUT is compared with T_c obtained from step 4. If $CUT > T_c$ then a valid target detection is declared [39].

The CA-CFAR processor runs through the entire FFT output $X[n]$ considering each cell as the CUT. Given the parallel nature of this CA-CFAR architecture, FPGAs can immensely speed up detection owing to their parallel processing capabilities [39].

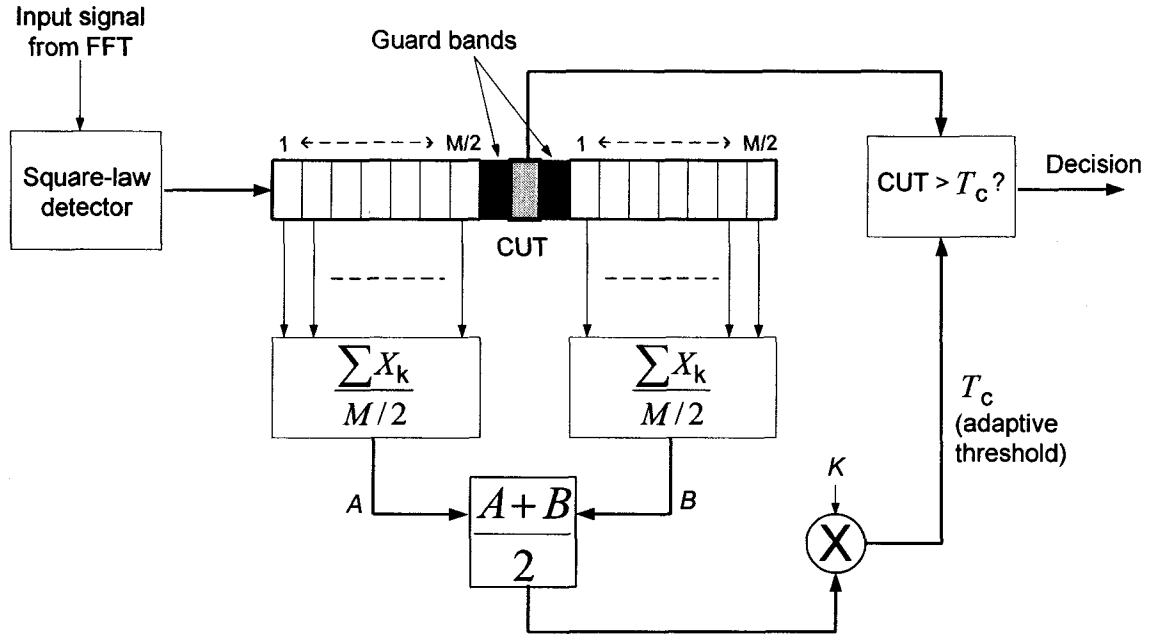


Figure 3.8: CA-CFAR processor architecture as implemented in this thesis.

The CA-CFAR has two slight variants from the implementation shown in Figure 3.8. Instead of computing the average of A and B , the GO (greatest of) –CFAR makes use of the greater value between A and B , while the LO (least of) –CFAR makes use of the smaller value between A and B to be multiplied by K . So in GO-CFAR, $T_c = A \times K$ where $(A > B)$ and in LO-CFAR, $T_c = B \times K$ where $(A < B)$ [41].

3.4.4 Miscellaneous Topics

3.4.4.1 Radar Targets

In the 1950s, Peter Swerling of RAND Corporation developed mathematical models to classify radar targets into 5 types based on the RCS or radar cross-section they display [43]. These are known as the Swerling I, II, III, IV and V models for radar targets, and present a mathematical model to determine the RCS of a radar target [42]. The classification is modeled using the *chi-squared distribution*, which is beyond the scope of this thesis. In simple terms, the parameters that affect the type of Swerling model are:

- Shape of the target.
- Degree of freedom for the target.
- Maximum and average RCS viewable from the target.
- Variation pattern in the RCS of the target with time and space.

The RCS of a target is determined by the following relation:

$$\sigma = \lim_{r \rightarrow \infty} 4 \pi r^2 \frac{|E_s|^2}{|E_i|^2} \quad (3.22)$$

Here, E_s = Scattered field intensity at distance r

E_i = Incident EM intensity on object

The radar cross-section of a vehicle is one of the factors which determine the maximum unambiguous range the radar can cover.

Swerling I targets:

- Consist of 5 or more scattering surfaces equally contributing to the overall RCS.
- Have a constant RCS throughout a CPI or scanning interval, but independently varying RCS in different radar beam scans.

- The distribution of RCS is described by the following relation [43]:

$$p(\sigma) = \frac{1}{\sigma_{\text{avg}}} e^{-\frac{\sigma}{\sigma_{\text{avg}}}} \quad (3.23)$$

Where σ is the RCS of the target and σ_{avg} is the mean value of RCS for the target.

Swerling II targets:

- Classification is similar to that of Swerling I, however the RCS varies during a single frequency sweep or CPI instead of staying constant. This represents more dynamic targets.

Swerling III targets:

- Consist of 1 main scattering entity and may possess several less significant smaller scattering surfaces.
- The RCS p.d.f. tends to remain constant through a single LFM CW sweep scan.
- The p.d.f. is characterized by equation (3.24) as follows:

$$p(\sigma) = \frac{4\sigma}{\sigma_{\text{avg}}^2} e^{-\frac{2\sigma}{\sigma_{\text{avg}}}} \quad (3.24)$$

Swerling IV targets:

- Similar to Swerling III targets, however the RCS scattering varies within a single scan and thus represents a more dynamic case of Swerling III targets.

Swerling V targets:

- Characterized by a constant RCS independent of time. These targets are easiest to detect as there is ideally no spectral deviation during or over consecutive frequency sweeps.

Typically, Swerling II and IV targets are harder to track due to variation in RCS, and hence reflected power, over a single CPI or sampling interval.

3.4.4.2 Noise

Contemporary radar systems are affected by various types of noise sources. Noise may originate from the signal conditioning analog components, the RF circuitry and antennæ, and the digital processing of the signal. Major sources of noise are listed:

1. Background noise – cosmic radiation, atmospheric absorption of EM radiation and noise temperature of the Earth contribute to background noise which manifests as white noise in all communication systems. This noise gets amplified throughout the system and can be accounted for by adequate signal processing.
2. Thermal noise – generated due to thermal motion of semiconductor charge carriers contributing to increased resistance in electronic and RF circuit components [5].

$$\text{Thermal noise, } N_{\text{Th}} = kT_{\text{A}}B \quad (3.25)$$

Where k is Boltzmann's constant, T_{A} is the average absolute temperature around the circuit components and B is the system bandwidth.

3. 1/f noise – *pink noise* power is inversely proportional to frequency. High frequency systems such as radars suffer less effects of 1/f noise [44].

4. Quantization noise – when sampling the intermediate frequency of the radar return using an ADC all continuous samples are rounded to the nearest quantization level available. For instance, for a 10-bit ADC with an input range of 1V-5V, an input of 0.22V would mean $1024 * (0.22V / 4V) = 56.32$ levels. However, since the number of levels in the ADC is an integer from 1 to 1024, this voltage would be quantized to level 56 corresponding to 0.21875V, hence an error of 0.125% is induced.

The SNR_Q or signal-to-quantization-noise ratio for an ADC is defined as follows [45]:

$$SNR_Q (dB) = 6.02N + 4.77 + 20\log_{10}(L_F) \quad (3.26)$$

Here, L_F is the RMS input voltage divided by the maximum acceptable voltage for the ADC.

3.4.4.3 Attenuation

Atmospheric attenuation is a necessary evil in radar systems. Attenuation varies with weather and the moisture level in the air. Table 3.3 lists attenuation under different weather conditions [46]. For this research work, an attenuation of 0.8 dB/km has been considered, which falls between light rain and medium rain conditions, resulting in an SNR of 4.73 dB.

Table 3.3: Atmospheric attenuation at 70-80 GHz

Condition	Precipitation Rate (mm/hr)	Attenuation (dB/km)
Clear, dry air	0.00	0.1
Drizzle	0.25	0.2
Light rain	1.25	0.5
Medium rain	12.50	1.5
Heavy rain or snow	25.00	9.0

Although severe weather conditions can completely mask a target, within operable conditions attenuation is beneficial for a radar system. One of the most important aspects of LFM CW radars is *peak pairing*. Peak pairing is the technique by which a peak detected in the up sweep is paired with a peak detected in the down sweep as belonging to the same target. Every target manifests as a peak in each of the sweeps, therefore if reliable peak pairing is not accomplished, the target information would be grossly incorrect. One of the most logical criteria for peak pairing is power level comparison: a target at distance 10 m will have larger frequency-domain peak magnitude than a target at 30 m.

3.4.4.4 Clutter

Radar clutter is defined as the unwanted back-scatter reflection to the radar sensor from objects of no interest or invalid targets. In the automotive scenario, clutter is contributed by trees, water, buildings, sign posts, road surface, barriers or dividers, and even the host vehicle's bumper, among other sources. All these objects are not real targets of interest such as cars or trucks in the path of the vehicle; however clutter does contribute to the received radar signal at the antenna. Most sources of clutter are stationary sources and thus remain fixed to a particular frequency bin over a scan sweep. The other property of clutter is the low power and constant RCS, and can be

classified as Sterling V type targets. Except ground clutter, most low-intensity spurious *spikes* in the output of a frequency analyzer can be effectively removed by means of CFAR processing, owing to the fact that all clutter sources exhibit very little or no Doppler shift over LFMCW frequency chirps. Ground clutter infests the lower frequencies due to its close proximity to the host vehicle, and can thus be handled by filtering out those frequencies. In digital signal processing, ground clutter is removed by ignoring high-power returns in the lower frequency bins of the FFT output, and is a valid method assuming that the probability of a target existing within 30 cm of the radar sensor is very low.

Albeit the general attempt at removal of clutter from the target return spectrum, a recent literature in [47] illustrates the idea of making use of clutter as valuable information in mapping the surrounding scenario. Literature [47] propounds the estimation of road curvature and detection of road dividers and partitions based on common clutter received in automotive radar applications. Such information can prove useful in determining advanced security aspects of the trajectory of the host vehicle, and act as a smarter adaptive cruise control system.

3.4.4.5 Radar Jamming

Jamming occurs when high-power microwave signals occupy the entire bandwidth of operation of a radar sensor and render it incapable of distinguishing between false and true targets. Although typically jamming has been an intentional ploy by security agencies [48], in the automotive radar scenario jamming may occur due to interference from nearby radar systems operating at the same instant frequency at the very same time, or from broadband pulsed Doppler radars that generate high-power pulses.

Frequency hopping is a well-known ECCM or Electronic Counter Countermeasures solution. This allows FSK radars with several frequency hops better resistance to

jamming, although complete resistance is not guaranteed. Likewise, LFMCW radars suffer less effects of jamming due to the constant frequency chirps.

3.4.4.6 Safe Distance Determination

A concise formula for safe distance calculation has been presented in literature [55]. Consider the scenario in Figure 3.9, where the host vehicle with the radar sensor is moving at velocity v_2 following a vehicle at velocity v_1 .

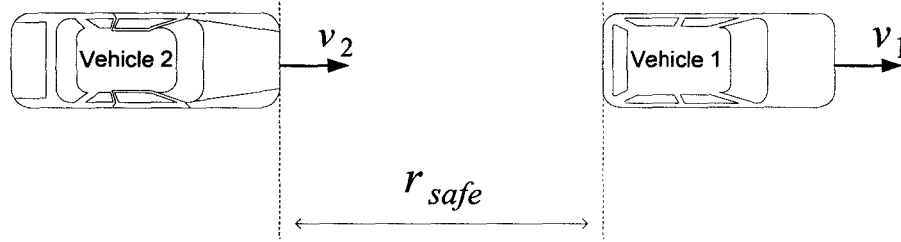


Figure 3.9: Safe distance between two vehicles.

Let the deceleration rate of the host vehicle be a_1 and the deceleration rate of the radar target vehicle be a_2 . Finally, let T_r be the reaction time of the driver of the host vehicle. Then, the safe distance that should be maintained by the host vehicle from the leading vehicle is given by

$$r_{\text{safe}} = \frac{1}{2b_2}v_2^2 - \frac{1}{2b_1}v_1^2 + v_2T_r \quad (3.27)$$

The value of b_1 and b_2 is dependent on the braking performance of the vehicles in different road conditions. On a dry road, $b_1 \approx 6.5 \text{ m/s}^2$ and $b_2 \approx 6.0 \text{ m/s}^2$ assuming $T_r = 1.0 \text{ s}$. On a surface covered with ice, $b_1 \approx 2.6 \text{ m/s}^2$ and $b_2 \approx 2.0 \text{ m/s}^2$ [55].

CHAPTER 4:

RADAR CONTROL AND SIGNAL PROCESSING ALGORITHM

This chapter presents the developed algorithm and where it fits into the whole automotive radar system. The long range automotive radar system being developed at the University of Windsor has three primary requirements: target range measurement, target velocity measurement and target angle measurement. This thesis develops a system to measure target range and velocity based on the LFM CW approach using a MEMS Rotman lens, MEMS RF switches and phased array antennæ for transmission and reception. The signal processing algorithm controls the modulation of the linear frequency chirps in the transmission side and also processes the received echo signal after it has been conditioned. Signal conditioning and common noise and attenuation issues faced by radar developers, have been detailed in Chapter 3.

This chapter lists the decisions made while designing the radar signal processing algorithm, and describes the operation of individual blocks with reference to the initial system specifications described in Table 4.1.

Table 4.1: Initially provided System Specifications

Parameter	Value
Radar type	LFMCW
Operating frequency	77 GHz
VCO used	TLC77xs ¹
Target model(s) considered	Reliability guaranteed with Swerling I, III and V type targets
Beamformer	Rotman lens
Number of beams	3 beams ²
Processing duration per beam	2 ms
Beam width	$\pm 4.5^\circ$
Antenna type	Phased array antenna
Radar processing unit (RPU) platform	FPGA

¹ 76.5 GHz MMIC VCO by TLC Precision Wafer Technology

² Reference [1]

Figure 1.1 shows the conceptual diagram of the entire radar system, showing the major components of the MEMS based radar system including the MEMS Rotman lens, MEMS RF switches, and the FPGA for signal processing. As shown, the tuning voltage is obtained from the DAC, and as described in Chapter 3 this translates to the triangular frequency chirp which is broadcast through the SP3T switch and Rotman lens combination into the phased array antenna.

4.1 Radar Transmitter Control and MEMS RF SP3T Switch Control

Responsibilities of the algorithm in the transmission part of the radar system:

1. Generate the radar frequency chirp by tuning the VCO with a voltage sweep through a DAC.
2. Synchronize chirp generation with receiver side signal processing, giving appropriate delay when the sampler is busy.
3. At the end of every down sweep, modify the MEMS switch control bits to switch to the next beam port, thus changing beam direction.
4. Switch between MEMS Rotman lens beam ports; beam port 1 to beam port 2, beam port 2 to beam port 3, beam port 3 back to beam port 1.

Figure 4.1 illustrates the transmitter side operation flowchart for the algorithm. On system reset, the sensor begins with beam port 1 of the Rotman lens, and by default would be designed to start with the up sweep or positive frequency chirp. The DAC is configured to output a voltage range from $V_{\text{tune-min}}$ to $V_{\text{tune-max}}$, which is the range required to tune the VCO over the desired sweep bandwidth of the system.

For the target sweep duration of 1 ms, a 10-bit DAC with a 900 ns refresh period would be a suitable choice based on current market availability of fast DACs.

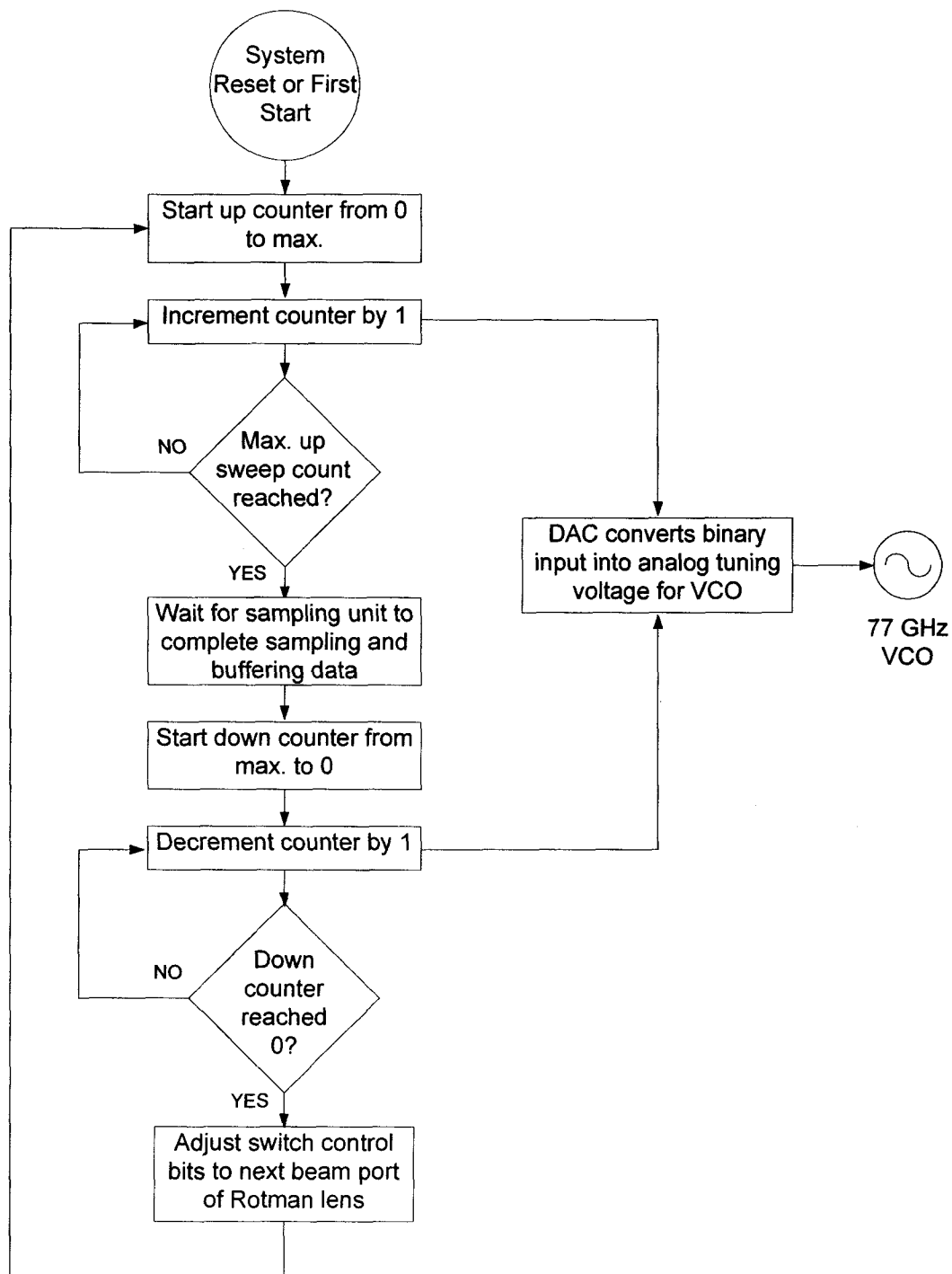


Figure 4.1: Flowchart for the operation of the developed radar algorithm's modulation and transmitter control unit.

4.2 Radar Receiver Flow Control and Signal Processing

The main part of the radar algorithm is its signal processing routine, the input to which are the time-domain ADC samples and the output from which is target information. The signal processing algorithm is responsible for the following internal tasks:

1. Apply the Hamming window to the time-domain samples acquired from the ADC.
2. Fast Fourier Transform of the windowed time-domain samples.
3. Peak intensity calculation for every frequency bin of the FFT output.
4. Run a CFAR algorithm and detect valid target peaks, neglecting noise and clutter, for both up and down sweeps.
5. Once both up and down sweeps have been processed by the CFAR unit, carry out peak pairing to calculate the target information.

The developed signal processing algorithm discussed above is defined in Figure 4.2. The superimposed graphs are generated from MATLAB and depict the time-domain samples as it passes through the radar signal processing system.

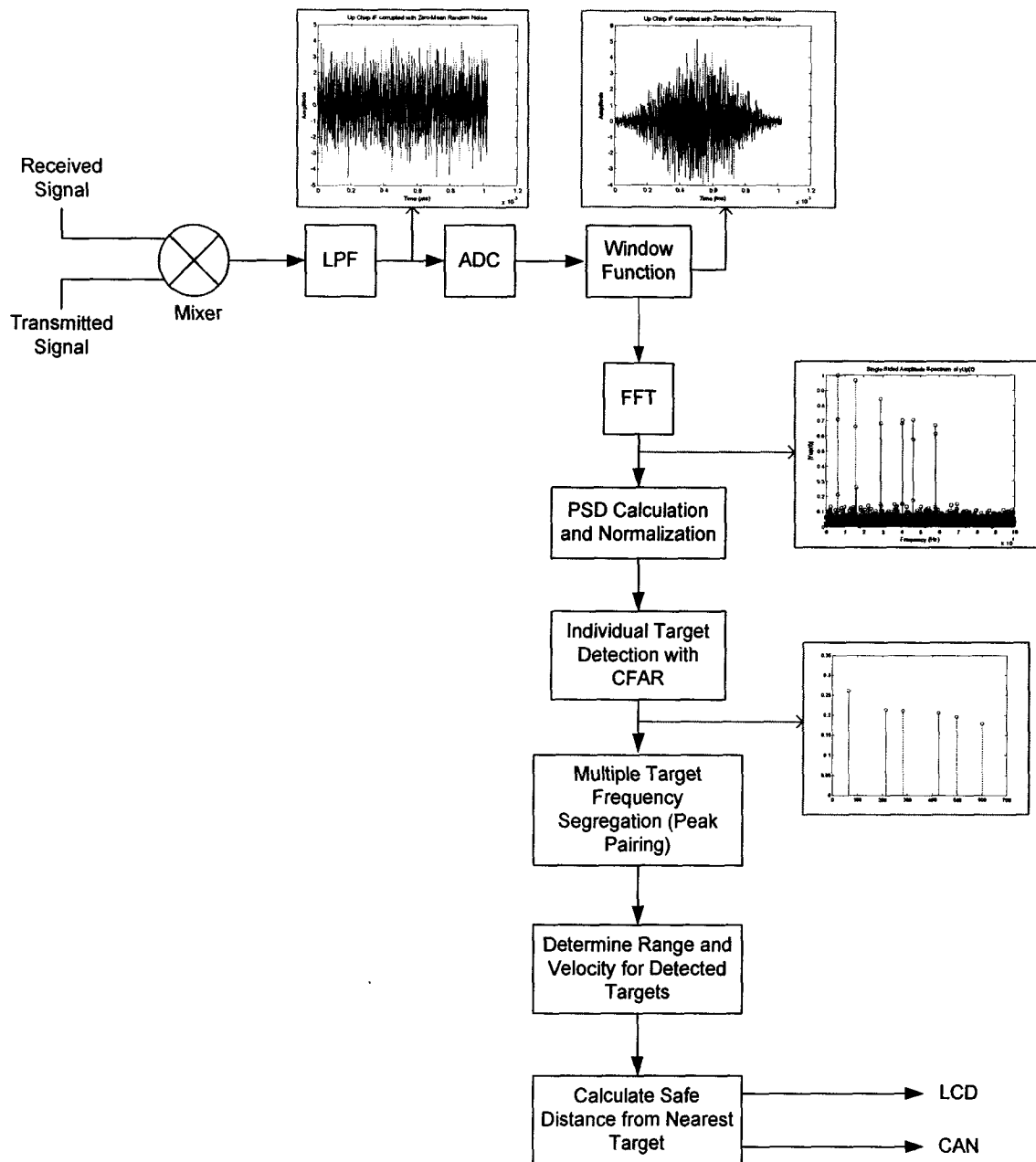


Figure 4.2: Radar signal processing algorithm developed as part of the radar control unit for this thesis. The first two superimposed graphs represent the time-domain sampled signal; the graphs post FFT processing represent frequency-domain processing stages. Signal conditioning steps are also shown – Mixer, LPF and ADC.

4.3 Selecting the Radar Sweep Bandwidth

The choice of components for the system is vital in determining the efficiency of the algorithm. One of the system parameters affecting the system components is the bandwidth of the system over which the frequency sweeps are made. The bandwidth selection involves a major trade-off: a higher bandwidth improves range resolution for the radar system (refer to equation 4.1), but also suffers non-linearity effects of the VCO. Following this trade-off, sweep bandwidths of 200 MHz, 400 MHz, 600 MHz, 800 MHz and 1 GHz were tested in MATLAB for the developed algorithm. Due to frequency spectrum allocation policies bandwidths are currently restricted to 77.5 GHz.

$$\text{Range resolution for LFM CW radar, } \Delta R = \frac{c}{2B} \quad (4.1)$$

$$\text{Velocity resolution for LFM CW radar, } \Delta v_r = \frac{\lambda}{2T} \quad (4.2)$$

Here, c is the speed of the EM radar wave in air, B is the LFM CW sweep bandwidth, λ is the wavelength of the radar wave, and T is the up or down sweep duration [49].

The graph in Figure 4.3 shows the results for maximum intermediate frequency and range resolution from the tests on different bandwidths. The target radar specifications for maximum range and maximum relative velocity were selected as 200 meters and ± 300 km/h in line with the state-of-the-art Bosch LRR3 radar presented in Chapter 1.

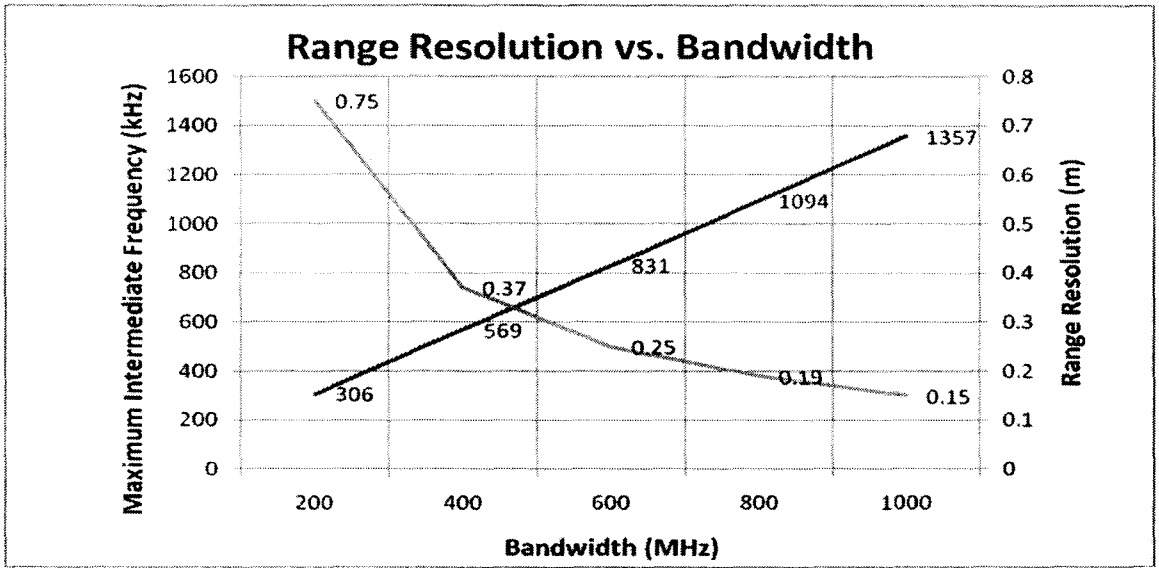


Figure 4.3: Variation of range resolution and maximum intermediate frequency with LFM sweep bandwidth.

As illustrated in Figure 4.3, an increase in bandwidth does improve range resolution for the radar system, but also increases the intermediate frequency. An increase in intermediate frequency means an increase in the sampling frequency, following Nyquist's sampling theorem. Another trade-off must be made at this point. Consider equations (4.2) and (4.3).

$$\text{Frequency resolution of FFT, } f_{\text{res}} = \frac{f_s}{N} \quad (4.3)$$

Here, f_s is the sampling frequency and N is the point size of the FFT which is equal to the number of time-domain samples collected.

The resolution of the FFT affects the minimum range gap between two frequency bins in the output of the FFT, which is nothing but the range resolution of the radar system. According to equation (4.3) frequency resolution can be improved by either lowering sampling frequency or by increasing the sampling duration or both. Increasing the sampling duration also improves velocity resolution by equation (4.2), however the cycle time of the radar system increases, which is an undesirable effect.

Now, consider first a bandwidth of 200 MHz. The maximum expected intermediate frequency of a target at 200 meters distance going at a relative velocity of +300 km/h is close to 306 kHz. The minimum required sampling frequency is twice this frequency and is equal to 612 kHz. Restricting the point size of the FFT algorithm to 1024, say, for quick computation, we have $N = 1024$. This gives an FFT resolution of 597.66 Hz/bin, which translates to an ideal-case minimum target separation of 0.45 meters (which is within the theoretic range resolution of the radar sensor for this bandwidth – refer to Figure 4.3).

Secondly, consider a bandwidth of 1000 MHz. The maximum expected intermediate frequency of the same target now becomes 1357 kHz. The required sampling frequency for this bandwidth would be at least 2714 kHz. Restricting FFT size to 1024, the frequency resolution is equal to 2650.39 Hz/bin, corresponding to a minimum target separation of 0.40 meters. This range resolution is better than achieved with a bandwidth of 200 MHz using the same FFT point size.

Through this discussion it seems desirable to have a higher bandwidth however the limiting factor of VCO linearity is to be taken into account. The non-linearity of any MMIC VCO is a key issue, especially at higher bandwidths. In order to operate in a linear part of the VCO transfer function, as discussed in Chapter 2, the sweep bandwidth should not be set too high. Most modern MMIC VCOs promise a linearity of 0.5% over 1-2 GHz range (TLC 77xs VCO datasheet from TLC Precision Wafer Technology).

Following this discussion, and keeping under consideration the frequency resolution and timing constraint of 2 ms per beam or 1 ms per sweep, the bandwidth of 800 MHz is chosen. This choice necessitates an ADC with a sampling frequency of 2.2 MHz (2.2 MSPS or Mega Samples per Second), which over 1 ms would collect close to 2048 samples. A power of 2 is preferred for the sample count N so that a Radix-2 DIT FFT algorithm can be used, allowing faster and more hardware-efficient implementation on FPGA.

4.4 Configuration of System Components

4.4.1 ADC

As discussed above, with a bandwidth of 800 MHz a respectable range resolution is achievable. For this bandwidth, an ADC with sampling frequency of 2.2 MHz is required. Or alternately, a sampling frequency of 2 MHz can be used over 1.024 ms for an exact total of 2048 samples.

4.4.2 FFT

Over 1 ms (or 1.024ms), close to 2048 samples will be collected using the chosen ADC rate. This would be the size of the FFT required for the signal processing algorithm.

4.4.3 CFAR

The CA-CFAR was chosen as the CFAR processor architecture for this thesis. Results have been presented in Chapter 4 of this thesis supporting the validity of this choice. The CA-CFAR processes a total of 1024 frequency-domain peaks (only half of the FFT output is considered as the FFT is a symmetric algorithm) to identify valid targets from clutter and noise. The probability of false alarm P_{fa} is selected as 10^{-6} for the algorithm, with an averaging depth M of 4 cells on either side of the CUT and 2 guard bands on either side of the CUT. This generates the following value of scaling constant K :

$$K = P_{fa}^{-\frac{1}{2M}} - 1 = (10^{-6})^{-\frac{1}{2 \times 8}} - 1 \approx 1.3714$$

Using 2 guard bands on either side of the CUT allows for enhanced noise handling capability and increased immunity to spectral leakage as a secondary line of defense after the Hamming window.

4.4.4 Peak Pairing

Two criteria for peak pairing are used in the proposed radar signal processing algorithm [48], assuming Swerling I, III and V targets. These are:

1. Spectral proximity: With the chosen system bandwidth, a relative velocity of 300 km/h corresponds to a maximum frequency bin shift of 84 bins between up sweep and down sweep peaks belonging to the same target. This frequency shift is due to Doppler shift. Therefore, a peak detected in the up sweep will only be paired with a detected peak in the down sweep if they are within 84 frequency bins of each other.
2. Power level: The peak intensity of the FFT output is indicative of the power level in the return of a given target. A distant target would produce a larger beat frequency but at lower power compared to a nearer target. This relation has high probability of occurrence and can therefore be used as a pairing criterion, by which a peak in the up sweep would be paired with a peak in the down sweep if the difference in their power levels is small.

4.5 Developed Algorithm Summary

The decisions presented in this chapter set the ground for the software (MATLAB) and hardware (HDL on FPGA) testing of the devised radar signal processing algorithm. The subsequent chapters show simulation results and a comparison in performance of floating-point software (MATLAB) and fixed-point (HDL) systems. Table 4.2 lists the final parameters for this thesis.

Table 4.2: Final parameters for the devised signal processing algorithm

Parameters	Value
LFMCW sweep bandwidth	800 MHz
FFT size	2048
FFT type	Mixed Radix-2 and Radix-4 DIT
Up/down sweep duration	1 ms
ADC resolution / sampling rate	11 bits / 2.2 MSPS
DAC resolution / refresh period	10 bits / 900 ns
Target range	0.40 m – 200 m
Target relative velocity	± 300 km/h
CFAR Algorithm	CA-CFAR
CFAR Parameters	One-side cell-averaging depth = 4 One-side guard band count = 2

CHAPTER 5:

SOFTWARE IMPLEMENTATION AND SIMULATION

MATLAB simulations of the developed radar signal processing algorithm are carried out in this chapter, and the results presented. Based on the target specifications, the mathematical theory, and the component configurations the signal processing algorithm is tested for two different cases. The first case simulates targets detected in a 3-beam MEMS radar, and the second case assumes a large wide-angle beam with a large number of targets. The results from the MATLAB simulation validate the developed algorithm and form the basis for the HDL implementation of the same.

5.1 Software Implementation of the Radar Signal Processing Algorithm

Following the Research Methodology stated in Chapter 1, after the development of the algorithm and decision on peripherals' configurations, the next step is a detailed MATLAB simulation of the proposed algorithm. MATLAB R2006b Version 7.3 has been used to develop the code for and verify the radar signal processing algorithm.

There are three stages to testing the algorithm in MATLAB:

1. Test the algorithm with and without a window function to validate the need for the extra processing produced by windowing.
2. Test the algorithm with a test case for a practical 3-lane highway scenario with 3 narrow beams between 3° - 6° width each.
3. Test the algorithm for a hypothetical scenario with a large number of targets in a single wide-angle beam between 15° - 30° in order to see the effect of saturating the radar sensor.

Before the results are analyzed, we must revisit the chosen algorithm parameters as presented in Chapter 4. With these system settings, one can proceed testing the algorithm. The flowchart in Figure 5.1 shows a flowchart of the sequential MATLAB program used to simulate the radar signal processing algorithm (see Appendix A1 for complete code listing).

Frequency sweep bandwidth = 800 MHz

Sampling frequency = 2 MHz

Sampling duration (up/down sweep duration) = 1.024 ms

Number of time-domain samples = 2048

FFT size = 2048

FFT frequency resolution = $2 \text{ MHz} / 2048 = 976.5625 \text{ Hz/bin}$

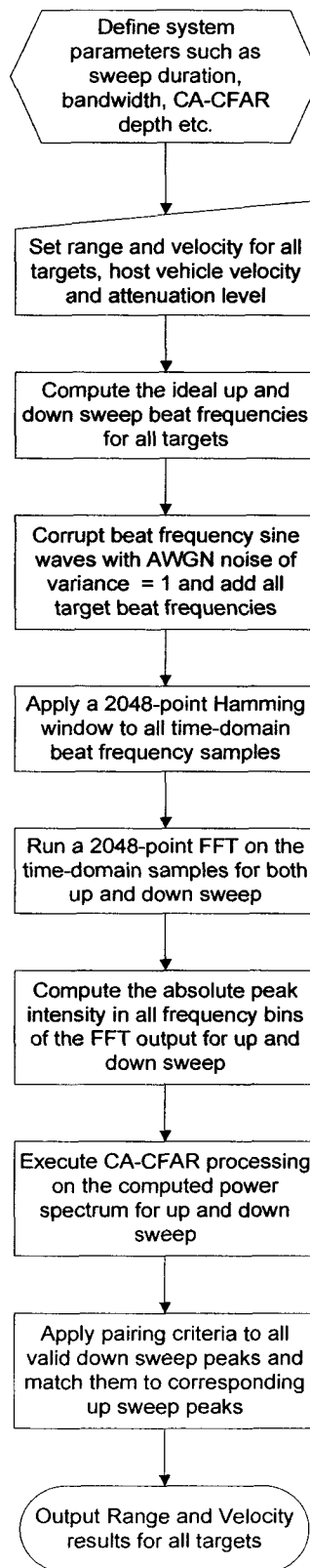


Figure 5.1: Flowchart for MATLAB simulation of the radar signal processing algorithm.

5.2 Testing Stage 1: Windowing versus No Windowing

Test scenario:

- 1 target at distance 142 meters.
- Target velocity is 165 km/h.
- Host vehicle velocity is 70 km/h.
- Therefore relative velocity is $(70 - 165) \text{ km/h} = -95 \text{ km/h}$, due to a negative Doppler shift caused by a receding target.

For the described target, the up sweep frequency would be a sum of the beat frequency component due to the distance of 142 meters, f_R , and the Doppler shift due to relative velocity -95 km/h, f_D , obtained from equations (3.11) and (3.13):

$$f_{\text{up_calculated}} = f_R + f_D = \frac{2kr}{c} + \frac{2f_0 v_r}{c}$$
$$= \frac{2 \left(\frac{800 \text{ MHz}}{1.024 \text{ ms}} \right) 142 \text{ m}}{2.973 \times 10^8 \text{ m/s}} + \frac{2 \times 76.7 \text{ GHz} \times \left(-\frac{95}{3.6} \text{ m/s} \right)}{2.973 \times 10^8 \text{ m/s}} = 732683.97 \text{ Hz}$$

Similarly, the down sweep frequency for the target amounts to the difference between f_R and f_D :

$$f_{\text{down_calculated}} = f_R - f_D = \frac{2kr}{c} - \frac{2f_0 v_r}{c}$$
$$= \frac{2 \left(\frac{800 \text{ MHz}}{1.024 \text{ ms}} \right) 142 \text{ m}}{2.973 \times 10^8 \text{ m/s}} - \frac{2 \times 76.7 \text{ GHz} \times \left(-\frac{95}{3.6} \text{ m/s} \right)}{2.973 \times 10^8 \text{ m/s}} = 759774.08 \text{ Hz}$$

5.2.1 Results without Windowing

The results obtained from the MATLAB simulation without any windowing stage in the algorithm are as follows:

Up sweep frequency bin number obtained from CFAR = 752 bins

∴ Up sweep frequency f_{up} obtained through algorithm

= 752 bins x frequency resolution

= 752 bins x 976.5625 Hz/bin

= 734375.00 Hz

Down sweep frequency bin number obtained from CFAR = 780 bins

∴ Down sweep frequency f_{down} obtained through algorithm

= 780 bins x 976.5625 Hz/bin

= 761718.75 Hz

Now, by equation (2.14) the target range from the simulation result is computed as follows:

$$\begin{aligned} r &= \frac{(f_{up} + f_{down})}{2} \times \frac{c}{2k} \\ &= \frac{(734375 + 761718.75) \text{ Hz}}{2} \times \frac{2.973 \times 10^8 \text{ m/s}}{2 \times \frac{800 \text{ MHz}}{1.024 \text{ ms}}} = 142.33 \text{ m} \end{aligned}$$

And, by equation (3.15) the target velocity from the simulation result is computed as

$$\begin{aligned}
v_r &= \frac{(f_{\text{up}} - f_{\text{down}})}{4} \times \frac{c}{f_0} \\
&= \frac{(734375 - 761718.75) \text{ Hz}}{4} \times \frac{2.973 \times 10^8 \text{ m/s}}{76.7 \times 10^9 \text{ Hz}} \\
&= -26.497 \text{ m/s} \\
&= -95.39 \text{ km/h}
\end{aligned}$$

5.2.2 Results with Windowing

The results obtained from the MATLAB simulation without any windowing stage in the algorithm are as follows:

Up sweep frequency bin number obtained from CFAR = 751 bins

\therefore Up sweep frequency f_{up} obtained through algorithm

= 751 bins x frequency resolution

= 751 bins x 976.5625 Hz/bin

= 733398.44 Hz

Down sweep frequency bin number obtained from CFAR = 779 bins

\therefore Down sweep frequency f_{down} obtained through algorithm

= 779 bins x 976.5625 Hz/bin

= 760742.11 Hz

Now, by equation (3.14) the target range from the simulation result is computed as

$$\begin{aligned}
r &= \frac{(f_{\text{up}} + f_{\text{down}})}{2} \times \frac{c}{2k} \\
&= \frac{(733398.44 + 760742.11)\text{Hz}}{2} \times \frac{2.973 \times 10^8 \text{ m/s}}{2 \times \frac{800\text{MHz}}{1.024\text{ms}}} = 142.15 \text{ m}
\end{aligned}$$

And, by equation (3.15) the target velocity from the simulation result is computed as

$$\begin{aligned}
v_r &= \frac{(f_{\text{up}} - f_{\text{down}})}{4} \times \frac{c}{f_0} \\
&= \frac{(733398.44 - 760742.11)\text{Hz}}{4} \times \frac{2.973 \times 10^8 \text{ m/s}}{76.7 \times 10^9 \text{ Hz}} \\
&= -26.497 \text{ m/s} \\
&= -95.39 \text{ km/h}
\end{aligned}$$

From this simulation result, the velocity of the target was obtained as the same with and without window. However, there is an observed improvement in range measurement by $(142.33 - 142.15) \text{ m} = 18 \text{ cm}$. Without windowing, the error for the measured range is $(142.33 - 142)/142 \times 100 = 0.23\%$. With the Hamming window, this error is reduced to $(142.15 - 142)/142 \times 100 = 0.11\%$, thus using an extra signal processing step to apply the Hamming window function to the time-domain samples offers considerable improvement in range measurement.

5.3 Testing Stage 2: 3-Lane Highway Scenario with Narrow Beam

Figure 5.2 and Table 5.1 illustrate the highway scenario being tested in this case. A 3-beam Rotman lens radar sensor has been considered, as described in Chapter 4. The host vehicle is taken to be travelling at 70 km/h.

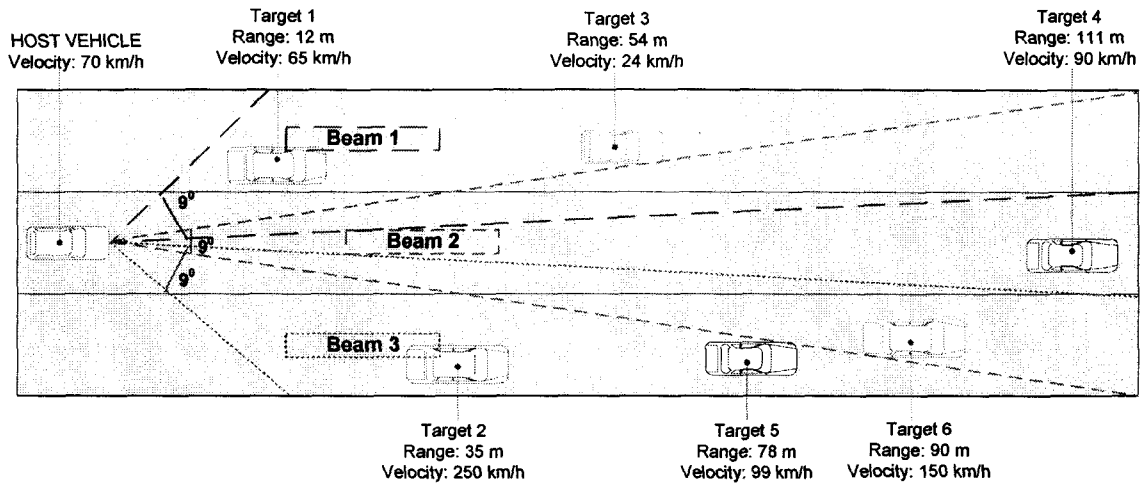


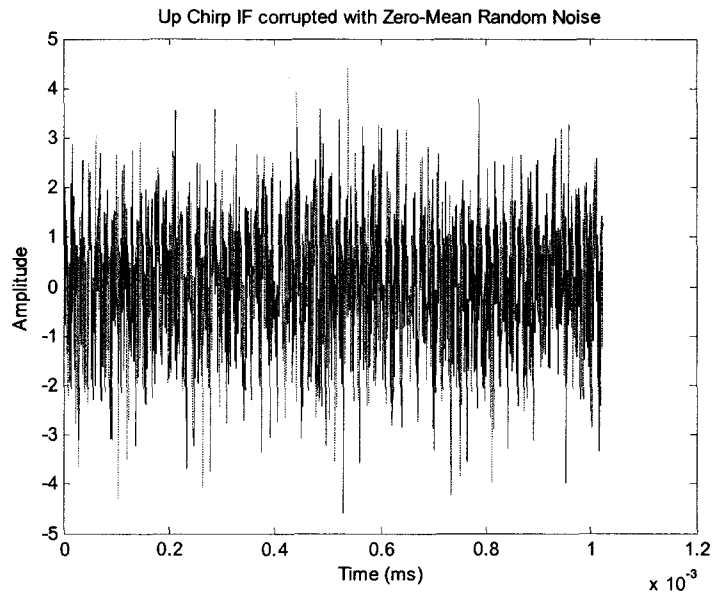
Figure 5.2: Test case highway scenario. Beam 1 shines 2 targets, Beam 2 covers 2 targets, and Beam 3 covers 3 of the targets. Beam width for the antenna is assumed to be 9°, with 4.5° Rotman lens beam steering.

Table 5.1: Practical Test Case Highway Scenario – Target Description

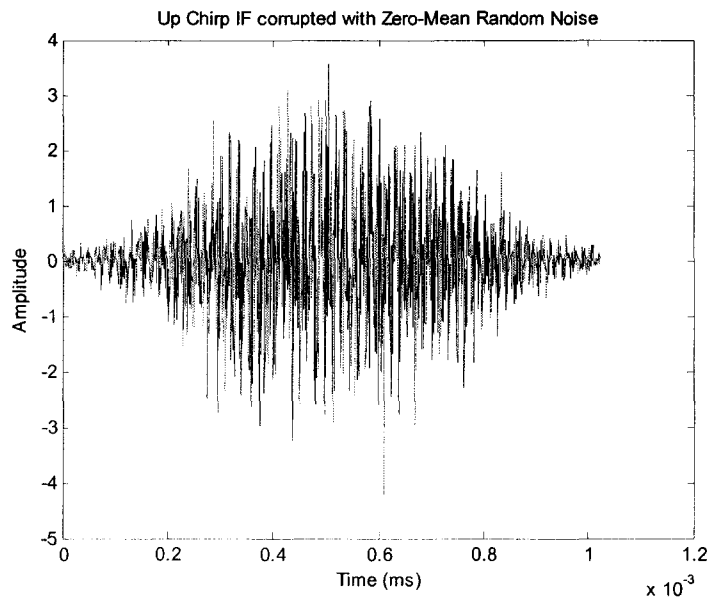
Beam Port Number	Target ID	Range (m)	Velocity (km/h)	Theoretical Up Sweep IF (Hz)	Theoretical Down Sweep IF (Hz)
1	1	12	65	63784	62358
	3	54	24	290397	277280
2	4	111	90	580509	586212
	6	90	150	461541	484354
3	2	35	250	158148	209477
	5	78	99	405783	414053
	6	90	150	461541	484354

All targets are assumed to be Swerling I or III type, and it is tacitly assumed that the return from each target sums up at the receiving phased array antenna of the MEMS radar sensor. This gives rise to the time-domain signals for Beam 1 up and down

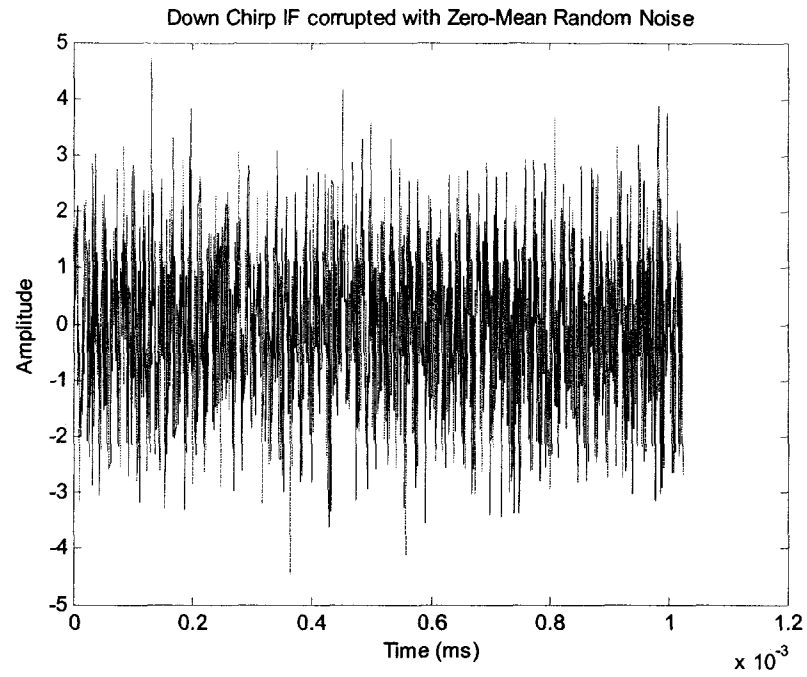
frequency sweeps shown in Figure 5.3, before and after being multiplied by the window function. The simulated time-domain signals for Beam 2 and 3 are similar to those illustrated in Figure 5.3. The signal has been corrupted with AWGN (Additive White Gaussian Noise) with unit variance. The simulated signal-to-noise ratio is 4.73 dB.



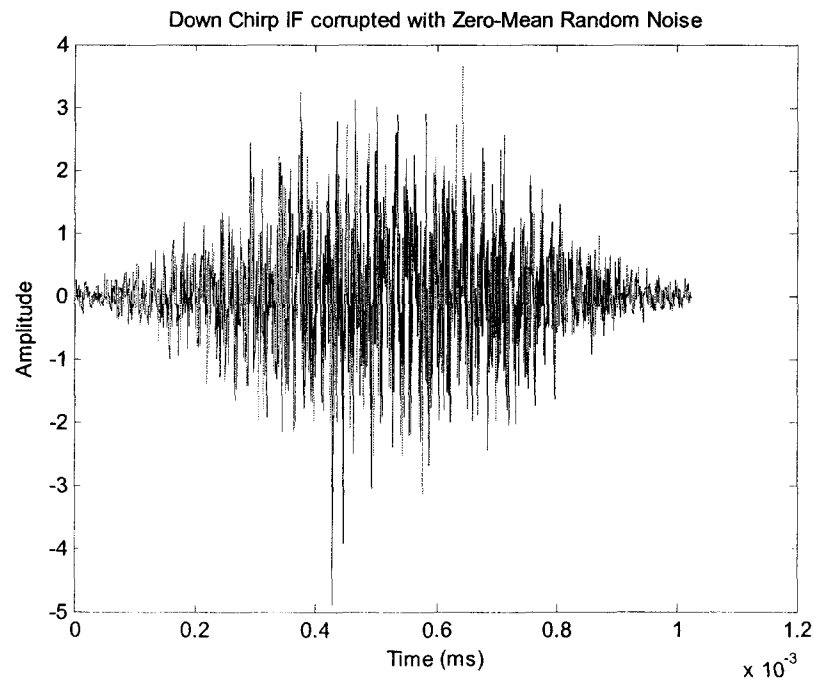
5.3(a) Received up sweep IF before windowing.



5.3(b) Up sweep IF signal after Hamming window.



5.3(c) Received down sweep IF before windowing.



5.3(d) Down sweep IF signal after Hamming window.

Figure 5.3: Time-domain signals for the up and down sweep of Beam 1 of the Rotman lens presented in the test scenario, before and after multiplication with the Hamming window.

Figure 5.4 shows the frequency analysis output from the FFT for Beams 1, 2 and 3 after windowing. The respective targets have been marked.

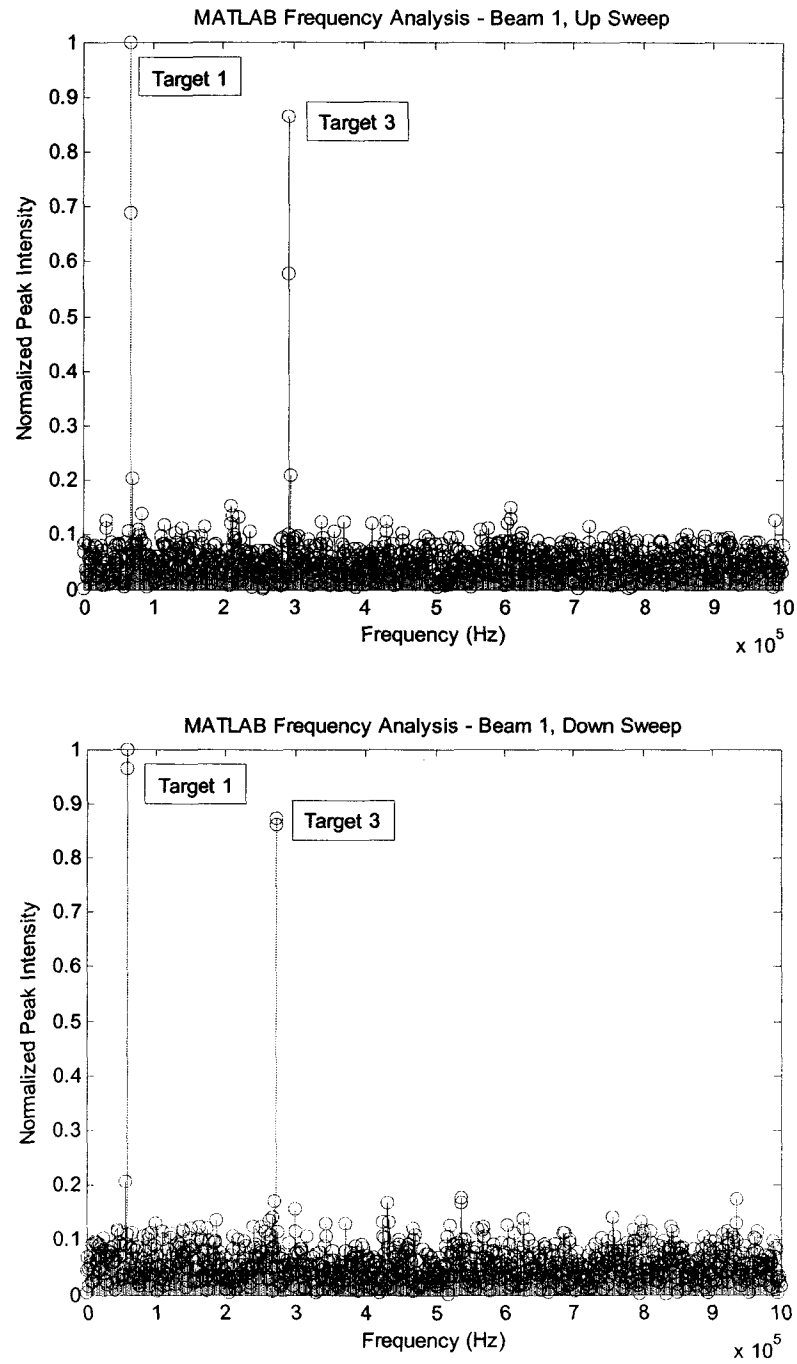


Figure 5.4(a): Spectral analysis of beam 1 targets in the up and down sweeps.

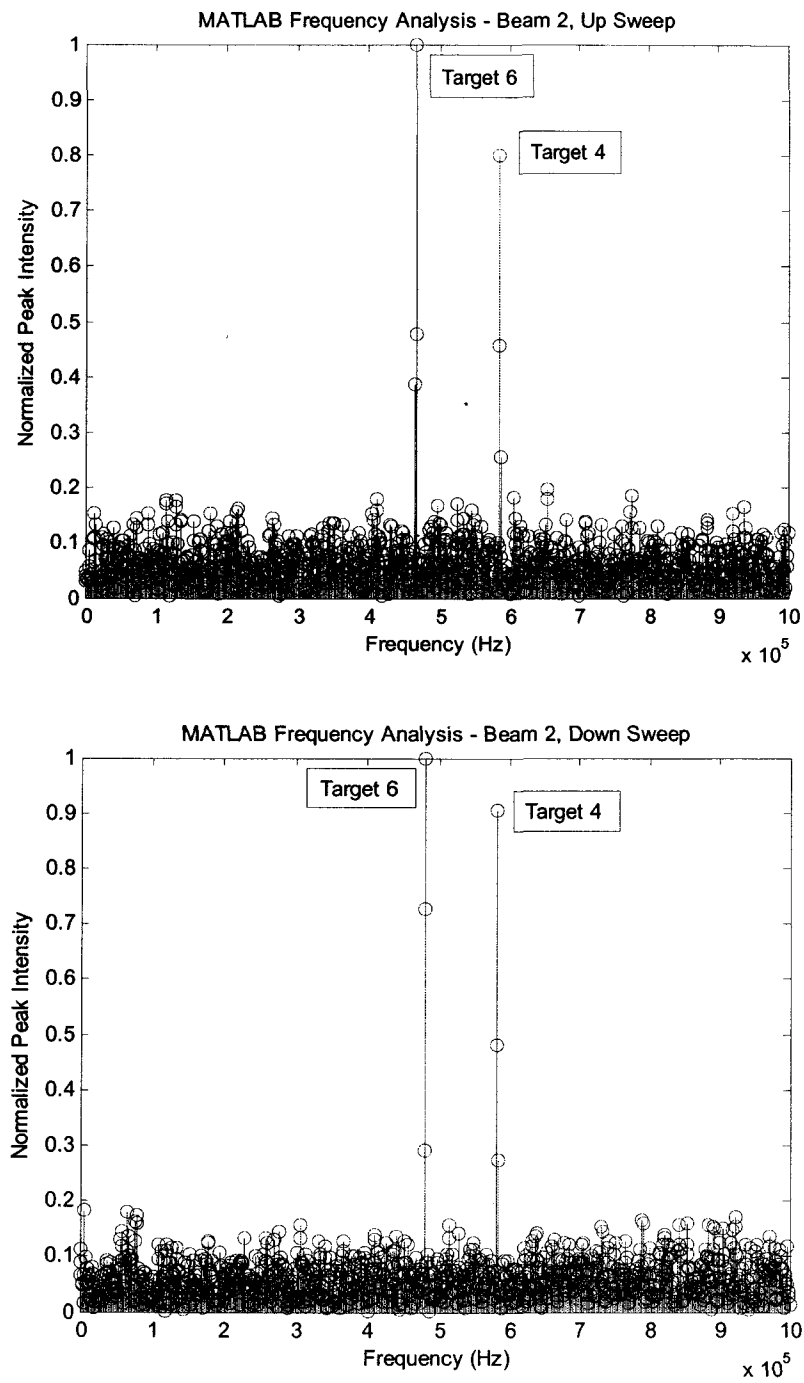


Figure 5.4(b): Spectral analysis of beam 2 targets in the up and down sweeps.

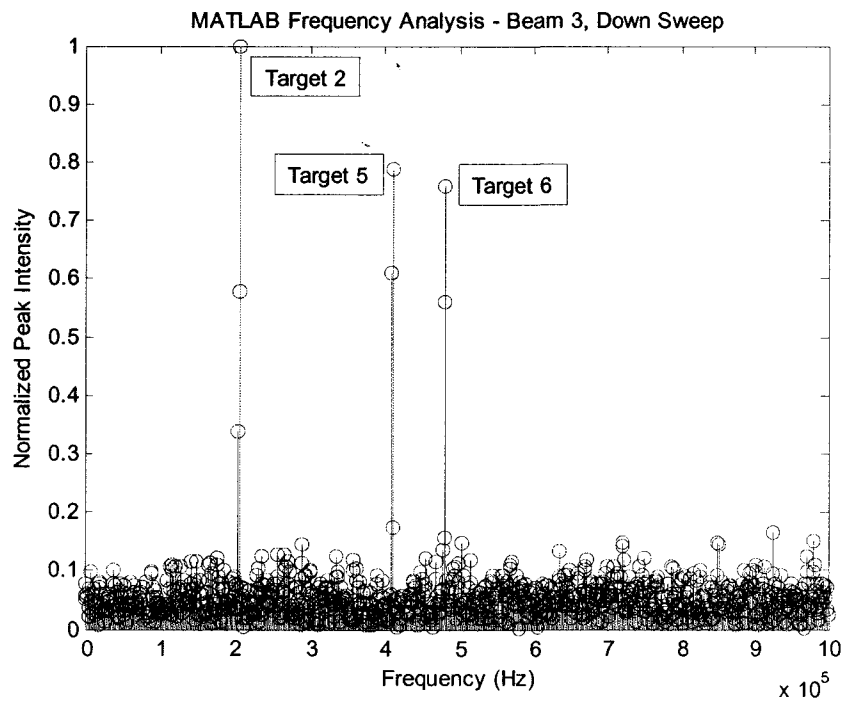
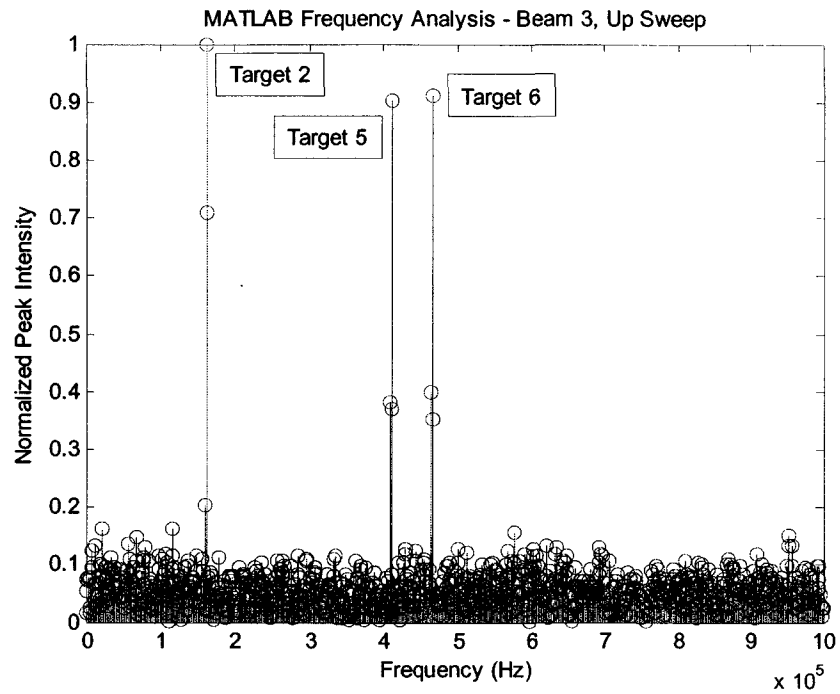


Figure 5.4(c): Spectral analysis of beam 3 targets in the up and down sweeps.

Figure 5.4: Frequency analysis of return signals in Beams 1, 2 and 3 shows the presence of targets.

The error induced in the floating-point MATLAB based radar signal processing algorithm is primarily due to the added AWGN added in the code (see Appendix for complete MATLAB listing), which is visible in the spectral plots in Figure 5.4. Table 5.2 shows the results obtained from the MATLAB simulations of the algorithm. The results presented are after successful pairing of the up sweep and down sweep peaks. Table 5.3 displays the errors from the simulation results.

Table 5.2: Results from MATLAB Simulation of the Developed Algorithm for 3-Lane Narrow Beam Scenario

Beam Port Number	Target ID	Measured Up Sweep IF (frequency bins) ¹	Measured Down Sweep IF (frequency bins) ¹	Measured Range (m)	Measured Velocity (km/h) ²
1	1	67	66	12.36	66.59
	3	299	286	54.35	25.71
2	4	596	602	111.30	90.44
	6	474	497	90.21	148.36
3	2	164	216	35.30	247.15
	5	417	426	78.32	100.66
	6	474	498	90.30	151.76

¹ Frequency resolution for 2048-point FFT = 976.5625 Hz/bin

² Target velocity has been calculated using equation (3.16)

Table 5.3: Errors for the Developed Algorithm from MATLAB Simulations for 3-Lane Narrow Beam Scenario (SNR = 4.73dB)

Beam Port Number	Target ID	Error in Range Measurement (m)	Error in Velocity Measurement (km/h)
1	1	0.36	1.59
	3	0.35	1.71
2	4	0.30	0.44
	6	0.21	1.64
3	2	0.30	2.85
	5	0.32	1.66
	6	0.30	1.76

Maximum error in range measurement for the developed algorithm: 0.36 m

Maximum error in velocity measurement: 2.85 km/h

5.4 Testing Stage 3: Hypothetical Scenario with 7 Targets Detected in a Single Wide Beam

The test scenario is presented in Figure 5.5. Only one wide-angle beam is considered for this simulation. The host vehicle velocity is set at 100 km/h, and it has direct line-of-sight detection of 7 simulated targets.

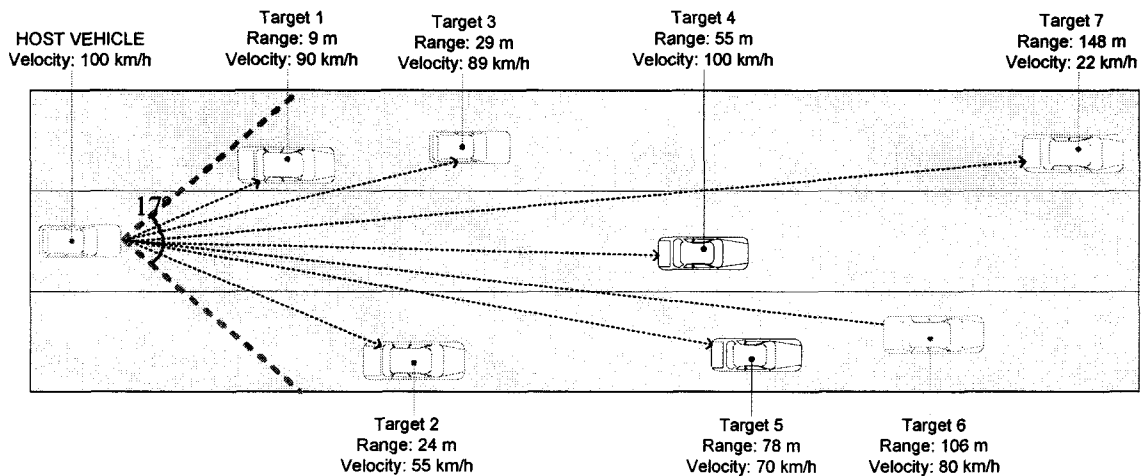


Figure 5.5: Hypothetical scenario with a single wide-angle antenna beam using only one beam port of the Rotman lens, i.e. no beam steering required to cover 3 central highway lanes.

To ensure fair and reliable testing, different target descriptions were used from Testing Stage 2. These target descriptions are tabulated in Table 5.4, and the results obtained from the MATLAB simulation are presented in Table 5.5. Figure 5.6 looks at the frequency analysis of the wide-angle beam, clearly labeling the 7 simulated targets. The CFAR processing results are shown, where all 7 target peaks have been correctly identified and extracted. This validates the accuracy of the employed CA-CFAR algorithm.

Table 5.4: Hypothetical Test Case – Target Description

Target ID	Range (m)	Velocity (km/h)	Theoretical Up Sweep IF (Hz)	Theoretical Down Sweep IF (Hz)
1	9	123	44004	50563
2	24	55	132585	119753
3	29	89	153990	150853
4	55	100	289060	289060
5	78	70	414239	405684
6	106	80	559964	554261
7	148	22	789013	766771

These targets have been selected randomly, and the test results are displayed after 6 complete iterations of the system for the same targets. This is one of the approaches to ensuring fair and reliable test results.

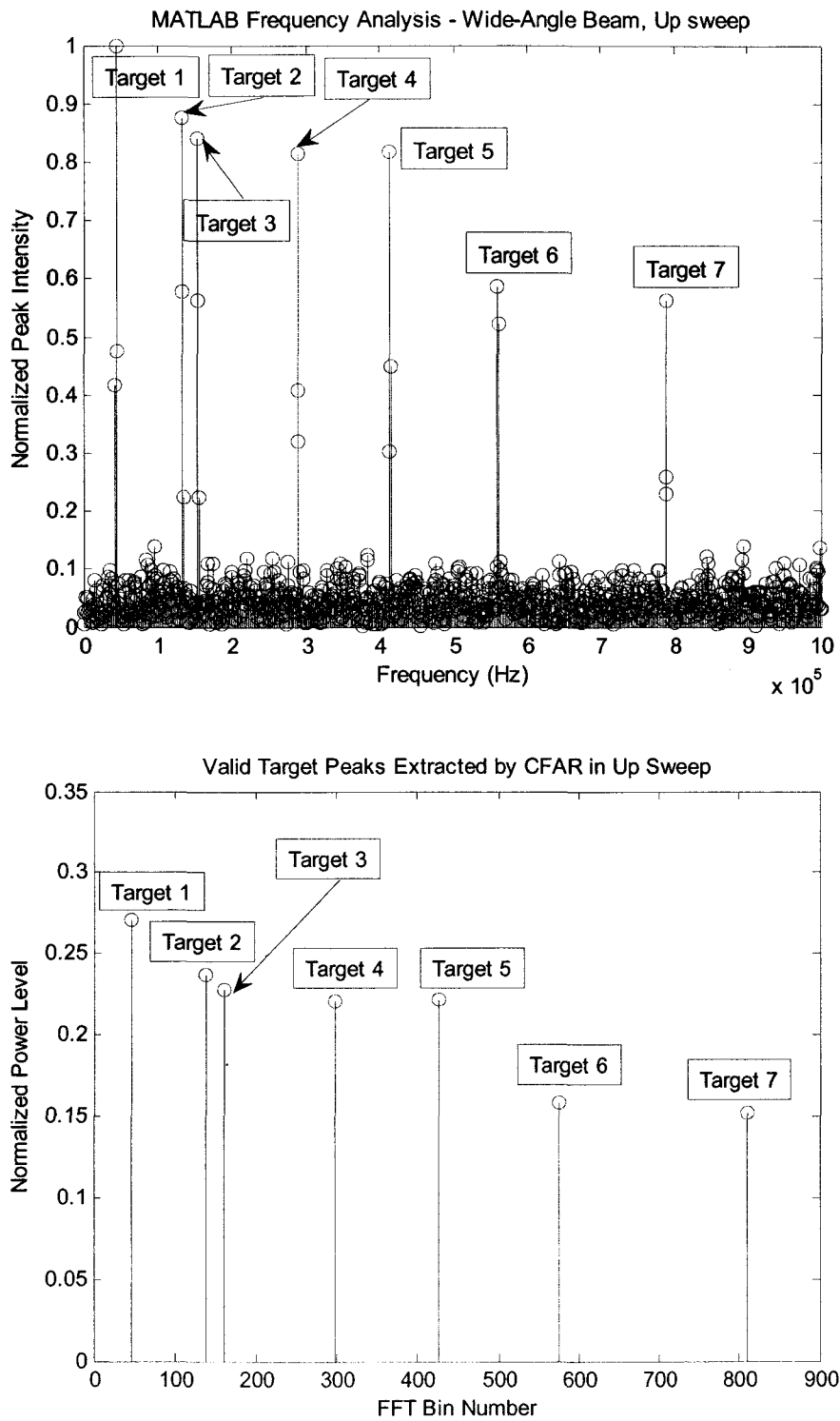


Figure 5.6(a): Frequency analysis of up frequency sweep for the wide-angle beam scan. The valid targets are shown as detected by the CFAR unit.

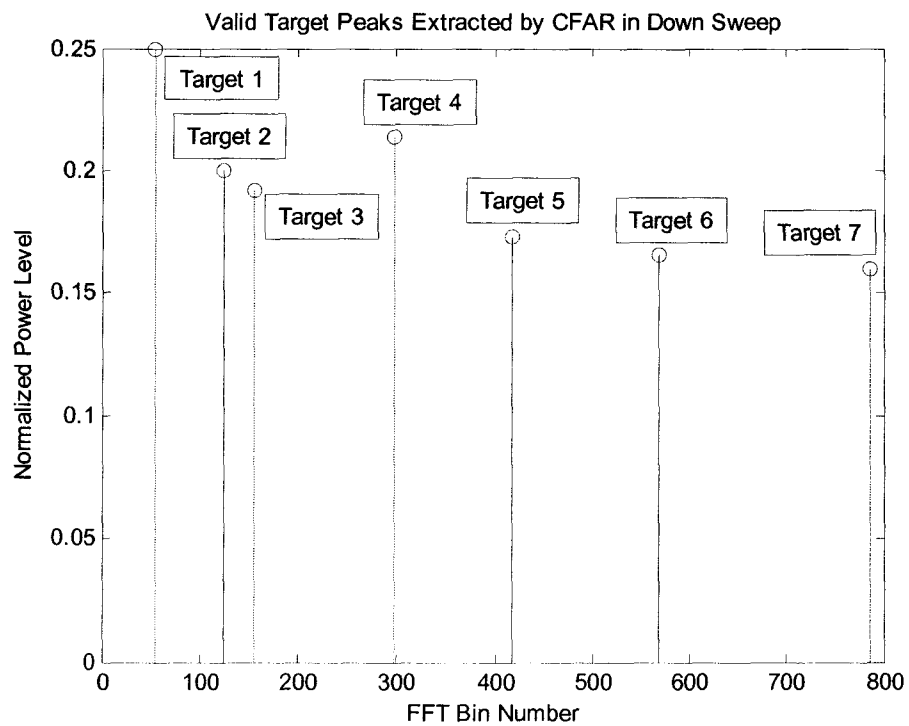
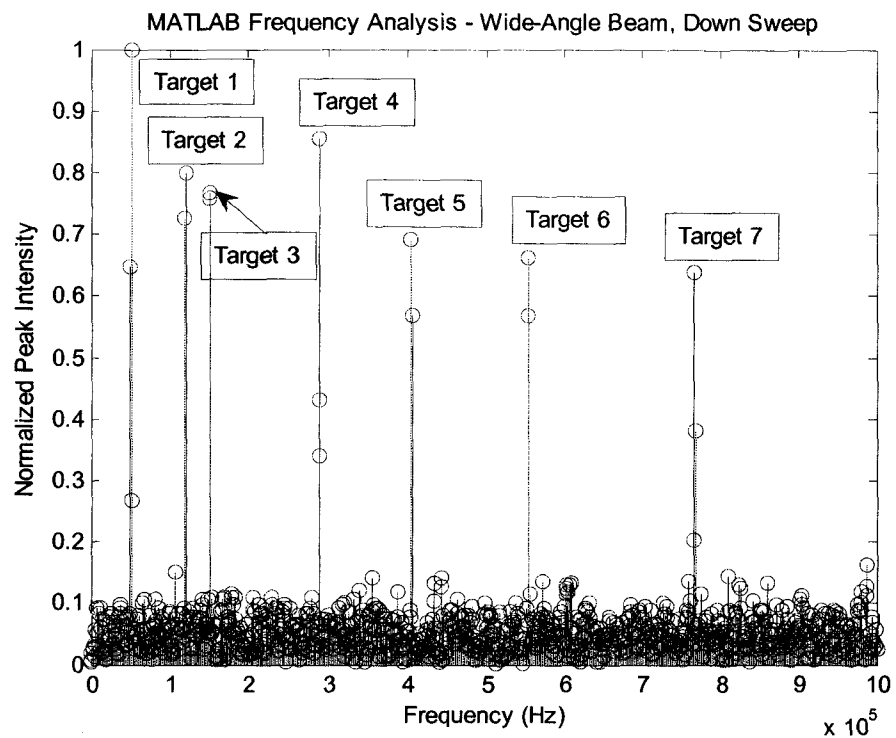


Figure 5.6(b): Frequency analysis of down frequency sweep for the wide-angle beam scan. The valid targets are shown as detected by the CFAR unit.

Table 5.5: Results from MATLAB Simulations of the Developed Algorithm for 3-Lane Single Wide Beam Scenario

Target ID	Measured Up Sweep IF (frequency bins) ¹	Measured Down Sweep IF (frequency bins) ¹	Measured Range (m)	Measured Velocity (km/h) ²
1	47	54	9.38	123.85
2	138	124	24.34	52.31
3	159	156	29.27	89.78
4	298	298	55.37	100.00
5	426	417	78.32	69.34
6	575	569	106.28	79.56
7	810	787	148.37	21.64

¹ Frequency resolution for 2048-point FFT = 976.5625 Hz/bin

² Target velocity has been calculated using equation (3.16)

Table 5.6: Errors for the Developed Algorithm from MATLAB Simulations for 3-Lane Single Wide Beam Scenario (SNR = 4.73dB)

Target ID	Error in Range Measurement (m)	Error in Velocity Measurement (km/h)
1	0.38	0.85
2	0.34	2.69
3	0.27	0.78
4	0.37	0.00
5	0.32	0.66
6	0.28	0.44
7	0.37	0.36

The error of the obtained target range and velocity measurements from the MATLAB Simulation of the radar signal processing algorithm are shown in Table 5.6.

Maximum error in range measurement for the developed algorithm: 0.38 m

Maximum error in velocity measurement: 2.69 km/h

5.5 Observations from Software Simulation Results

The simulations results confirm the validity of the developed algorithm and chosen system parameters such as bandwidth of 800 MHz and up/down sweep duration of 1.024ms. The chosen ADC sample rate of 2.0 MHz is appropriate for capturing exactly 2048 time-domain samples of the intermediate frequency or beat frequency signal.

The CA-CFAR algorithm has been tested and its operation validated through accurate extraction of valid targets from a background of noise and clutter with an SNR of 4.73 dB, which is a good performance with reference to literature [50] in which the author has described better SNR under normal conditions at mm-wavelengths.

The maximum error observed in the range determination of any target is 38 cm, while the maximum error in target velocity measurement is 2.85 km/h or 0.79 m/s. These errors are within tolerable limits compared to state-of-the-art automotive radars studied in Chapter 2.

CHAPTER 6:

HARDWARE IMPLEMENTATION AND VALIDATION

The signal processing algorithm is coded in Verilog HDL and the modular design has been shown in this chapter. The data flow through individual modules is described, and an overview of the entire HDL implementation is produced. A few alterations and fine-tuning of the FFT and CFAR modules have been done to improve noise tolerance and accommodate short range, medium range and long range target return attenuation and power variation. The coded system is simulated using Xilinx ISim and the waveforms have been illustrated. The results are promising and show lower error than the MATLAB simulations, primarily due to the fixed-point rounding of data as it propagates through the digital logic.

6.1 Hardware Implementation of the Radar Signal Processing Algorithm

The advantages of modern FPGAs over DSPs in running signal processing tasks have been highlighted in Chapter 2. The state-of-the-art Bosch LRR3 has a cycle time of 50 ms. An FPGA implementation presented in [28] displays a signal processing latency of 1250 μ s for a single LFM CW sweep using with a 1024-point FFT using a Xilinx Virtex-II Pro FPGA clocked at 50 MHz. To achieve a smaller computation latency per sweep, and hence a smaller cycle time for the MEMS based automotive radar, the target FPGA for this thesis is selected as Virtex-5 SX50T.

Figure 6.1 shows an annotated snapshot of the Virtex-5 development board and Table 6.1 highlights the main aspects of this FPGA. One of the advantages of using the Virtex-5 FPGA from Xilinx is the high integration capacity of the design, and the higher operating clock frequency, owing to the improved gate-level performance with 65 nm

technology. A faster clock frequency enables quicker computation of signal processing routines thus reducing overall cycle time for the MEMS radar further. It should be noted here that the MEMS Rotman lens and MEMS SP3T switches devised for the radar system are capable of handling switching times well below 1 ms.

Table 6.1: Xilinx Virtex-5 SX50T features [51]

Feature	Value
DSP48E Slices	288
Block / Distributed RAM	4,752 kb / 780 kb
Total LUT Bits	> 13 million
Maximum Clock Frequency	550 MHz
Gate Technology	65 nm
I/O Voltage / Core Voltage	1.2 V – 3.3 V / 1.0 V

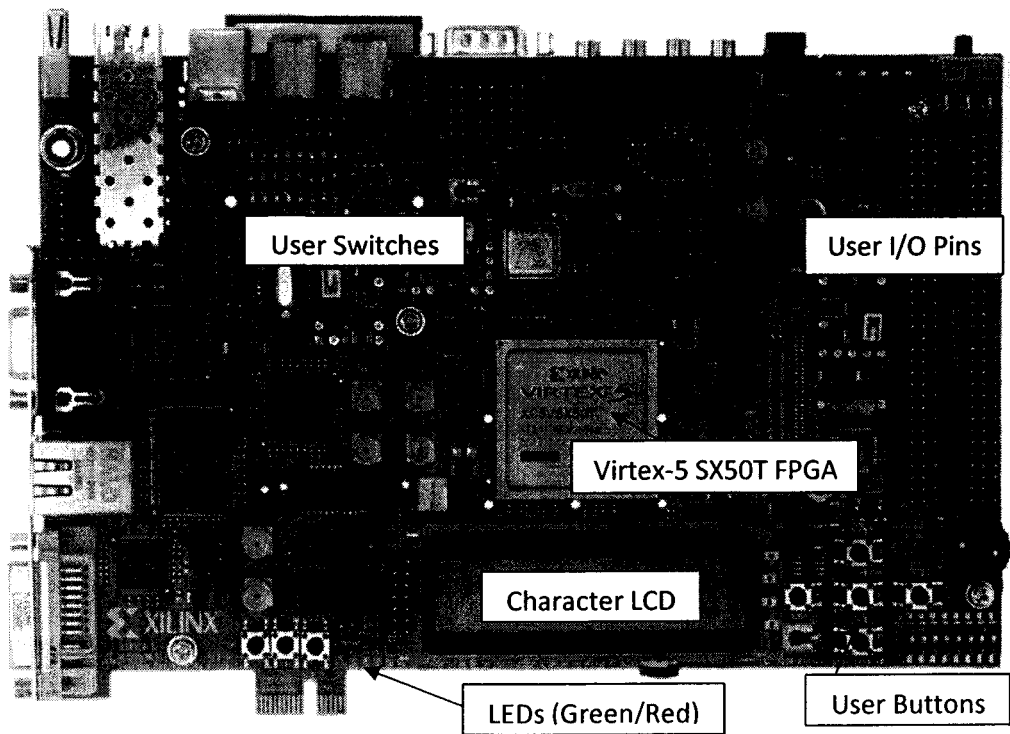


Figure 6.1: Xilinx Virtex-5 SX50T mounted on Development Board ML506 (annotated).

The resources offered by Xilinx Virtex-5 suffice for the developed signal processing algorithm and future expansions of the MEMS automotive radar project, while offering optimal speed. The on-chip system monitor has core temperature and power consumption sensors that can be used to ensure the system is always in working capacity.

6.1.1 Radar Signal Processing Algorithm on FPGA

The block diagram for the HDL implementation on FPGA is presented in Figure 6.2. The language used for the FPGA implementation is Verilog HDL (Verilog 2005 – IEEE Standard 1364-2005). The coding and simulation has been done using Xilinx ISE Design Suite 11.5. The HDL blocks are synonymous to the signal processing stages of the algorithm presented and tested in Chapter 5, and have been developed for a bandwidth of 800 MHz and a frequency sweep of 1.024 ms.

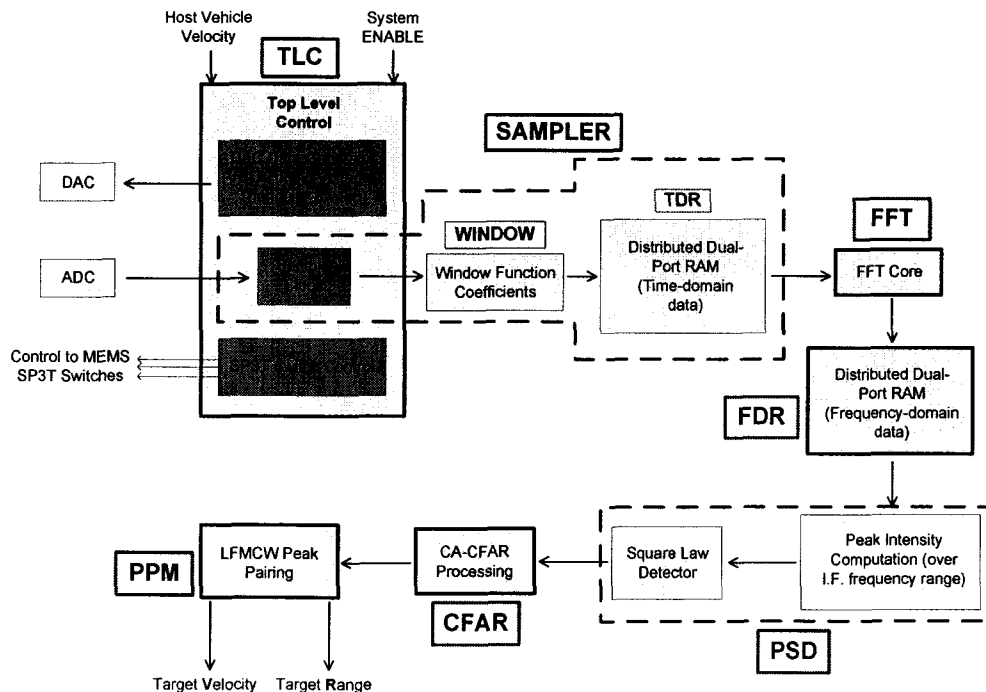


Figure 6.2: HDL blocks for the radar signal processing algorithm.

6.1.1.1 Top Level Control (TLC)

The TLC is the interface of the radar control and signal processing algorithm to the real world and MEMS radar RF components. The control part of the algorithm synchronizes the radar transmission and signal processing, and provides the sampling clock to the ADC and captures real-valued time-domain samples from the intermediate frequency of target echoes through the *sampler* unit. The TLC also provides a clock to the DAC, along with data bits to generate the tuning voltage for the VCO as discussed in previous chapters. The operation of this top level module is described by the flowchart in Figure 4.1.

Figure 6.3 below illustrates entire radar control and signal processing algorithm as a black box as seen from *outside* the FPGA.

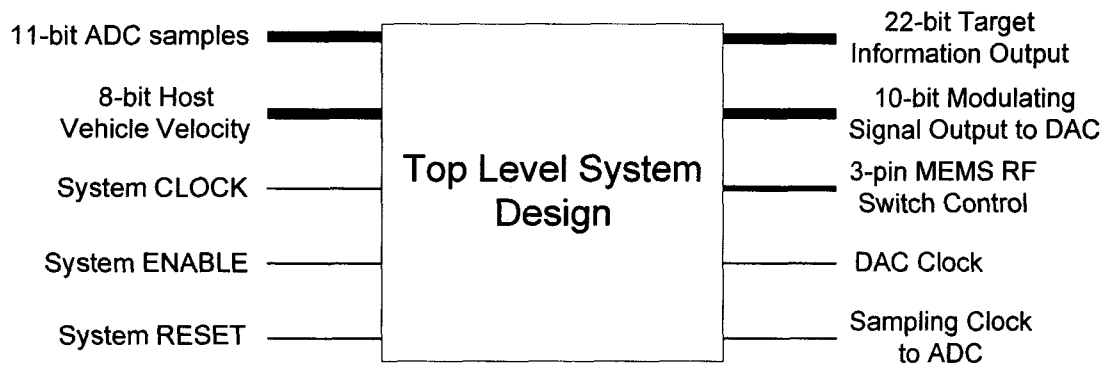


Figure 6.3: Black box view of radar control and signal processing algorithm. The thicker lines represent data buses. The left side represents inputs and the right side shows the outputs.

The 22-bit target information output from the unit has the following format:

[10-bit target velocity] [10-bit target range] [2-bit beam port number]

6.1.1.1.1 Velocity Precision

The velocity is presented in a [9.1] format in km/h, meaning 9 bits for the integer part and 1 bit for the fractional part. This means velocity measurement is restricted to a precision of 0.5 km/h. However, internally there is 5-bit precision for velocity calculation which gives a precision of 0.03125 km/h. The 5-bit precision has been curtailed to 1-bit fractional precision in order to restrict the length of the output target information.

6.1.1.1.2 Range Precision

The range is output in a [8.2] format, thus a precision of 0.25 meters is imposed on the HDL implementation. However, as in the case of velocity calculation, this fractional precision can be extended up to 11-bit precision or $0.00048828125 \text{ m} \approx 488 \mu\text{m}$. Since such precision is not required in automotive radar applications, the 11-bit internal precision is replaced with 2-bit fractional precision to shorten the output word length.

The beam port number appended at the end of target information represents the beam number the target was detected in, which is indicative of the estimated direction of the target.

Figure 6.4 below shows the top level module as seen in the Xilinx ISE Design Suite. Table 6.2 describes the input and output signals.

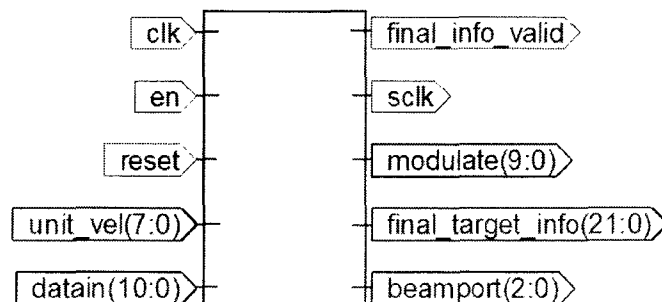


Figure 6.4: TLC in Xilinx ISE RTL viewer.

Table 6.2: Port description for TLC

HDL Port Name	Direction	Description
clk	Input	System clock at 550 MHz from ML506 development board on-board clock generator
en	Input	System enable signal
reset	Input	Global synchronous system reset
unit_vel	Input	Host vehicle velocity
datain	Input	Real-valued time-domain ADC samples
final_info_valid	Output	Signal is logic '1' or HIGH if a new target range and velocity information are being output
sclk	Output	Sampling clock from TLC to ADC
modulate	Output	10-bit data to DAC generated from an up/down counter in TLC – used to generate the tuning voltage to modulate the VCO output ¹
final_target_info	Output	This contains the target range and velocity measurement along with 2 bits describing the beam direction in which the target was detected
beamport	Output	Control pins for the MEMS SP3T switches to control the direction of the radar beam by controlling the beam port of the Rotman lens being fed

¹ For the TLC77xs VCO being used for this thesis, tuning voltage range of 2.5V to 6.5V generates output frequency range 76.5 ± 1 GHz. Therefore, for 800 MHz bandwidth centered at 76.9 GHz, the tuning voltage is 4.5 V to 6.1 V is required. A value of 0 on the *modulate* port will be output from the DAC as 4.5 V, and a value of (11 1111 1111)₂ or 1023 will result in 6.1 V.

The 3-pin MEMS RF switch control signal contains a bit each for the 3 MEMS switches that are controlled through charge pumps connected to the FPGA pins. These MEMS SP3T switches are responsible for routing the RF signal generated by the VCO to the appropriate beam port of the Rotman lens thus steering the beam. Isolation

between the supply voltage of the MEMS switches and the RF signal travelling through them is done using a bias-tee for each switch.

The system clock input for Virtex-5 is 550 MHz, obtained from the ML506 development kit from Xilinx. This clock signal is divided internally to an operating clock of 100 MHz, which is the target operating frequency for the radar algorithm.

Time-domain samples from the ADC are obtained in 11-bit format as per the decided resolution of the ADC (Chapter 3).

6.1.1.2 Sampling Unit (SAMPLER): sub-module Window Function (WINDOW)

This is a sub-module of the *sampler* module and contains a ROM storing 1024 coefficients of a 2048-point Hamming window. Since the Hamming window is symmetric, storing the first 1024 values is memory efficient. This sub-module contains a simple 10-bit up/down counter that extracts the coefficient depending on the index of the time-domain sample.

The Hamming window coefficients are floating point numbers, thus representing them in digital hardware requires rounding off. The precision of the coefficients is chosen to be 10 bits, with the maximum of $(11\ 1111\ 1111)_2$ representing the maximum coefficient value of 1. The rounded off Hamming coefficients are thus stored as integers ranging from (0.08×1023) to (1×1023) . The coefficients are obtained from MATLAB code which carries out the following steps (refer to Appendix for MATLAB listing):

1. Create a Hamming window of size 2048.
2. Multiply the window coefficients by 1023 to scale them to a 10-bit range.
3. Round off the scaled coefficients to the nearest integer.
4. Save the first 1024 coefficients in *sampler* ROM.

Although this rounding does introduce a secondary quantization error after the ADC, the results from simulation of the algorithm in HDL show desired accuracy and precision. The percentage error produced by from rounding the Hamming coefficients is 0.084%, which has negligible effects on the signal processing. A similar approach has been presented and validated by Hampson in [56].

The scaling an *x-bit* time-domain sample by multiplication with a window coefficient returns a scaled *x-bit* number; there is no change in the word length of the samples. This is done by retrieving only the most significant *x* bits from the result of the multiplication. Thus:

x-bit time-domain sample → WINDOW → *x-bit scaled time-domain sample*

This method of scaling has a maximum error of 0.1% per sample which is negligible. The preservation of word length proves efficient later on in the signal processing by limiting the memory sizes and reducing processing speed while retaining adequate accuracy.

6.1.1.3 Sampling Unit (SAMPLER): sub-module Time-Domain Data RAM (TDR)

This is a sub-model of the *sampler* module which is a dual-port Block RAM. This sub-module stores the windowed time-domain samples collected from the ADC. The width of the data RAM is 12 bits, and the depth is 2048 – 2 MHz ADC sampling over 1.024 ms. An important note to make is that although the ADC output is 11 bits long, the TDR module stores 12-bit samples. The extra bit is merely a ‘0’ added to the front of every sample. This is done as the FFT core used in this project works with 2’s complement input and output data, so appending a ‘0’ at the beginning of every time-domain sample converts all samples to positive values. The only effect of this method is a high DC component being detected in the first frequency bin of the FFT, which is safely ignored as it represents a negligible target range of 0.186 m or 18.6 cm. The first few

range gates of the FFT are ignored to avoid nearby clutter return from the host vehicle's bumper, the immediate ground level, and internal reflections in the radar sensor.

The TDR module is also responsible for feeding the sampled data to the FFT core. The TDR monitors the sample index being displayed from the FFT core and outputs the sample at that index. In this case, the index from the FFT core is used as the address to access the RAM in TDR. Upon sending all 2048 samples to the FFT core, the TDR sends a "start calculation" active-high signal to the FFT core. Figure 6.5 shows the overall *sampler* module as seen from Xilinx ISE. The timing diagram for the *sampler* unit is shown in Figure 6.6.

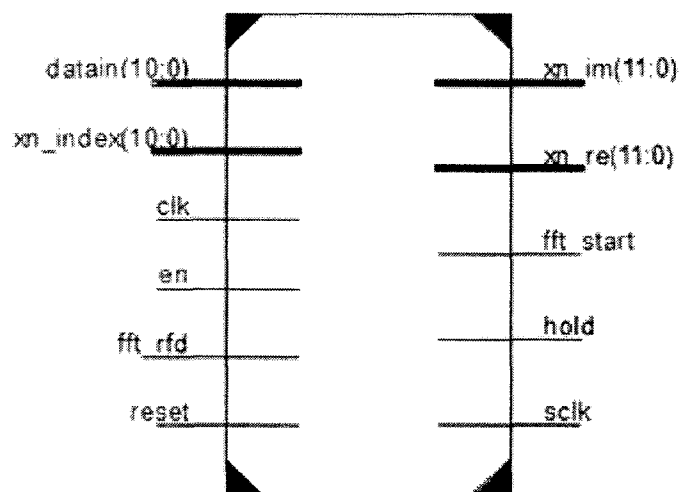


Figure 6.5: Xilinx ISE RTL view of *sampler* unit with sub-modules WINDOW and TDR.

Table 6.3: Port description for SAMPLER

HDL Port Name	Direction	Description
datain	Input from TLC	11-bit ADC sample
xn_index	Input from FFT	Index of the sample being passed to the FFT core
clk	Input from TLC	Operating clock of 100 MHz
en	Input	Enable signal
fft_rfd	Input from FFT	Control signal from FFT core indicating it is ready to accept new batch of data for processing
reset	Input	Global synchronous reset
xn_im	Output to FFT	Imaginary part of time-domain sample – this port is permanently grounded to 0
xn_re	Output to FFT	Real part of time-domain sample – windowed time-domain samples from dual-port RAM
fft_start	Output to FFT	Active-high start signal for FFT – initiates FFT computation
hold	Output to TLC	Active-high signal to TLC – a level ‘1’ on this wire makes the TLC halt modulation and sampling while all data is fed from RAM to the FFT core
sclk	Output to TLC	Sampling clock to ADC generated by the <i>sampler</i> unit

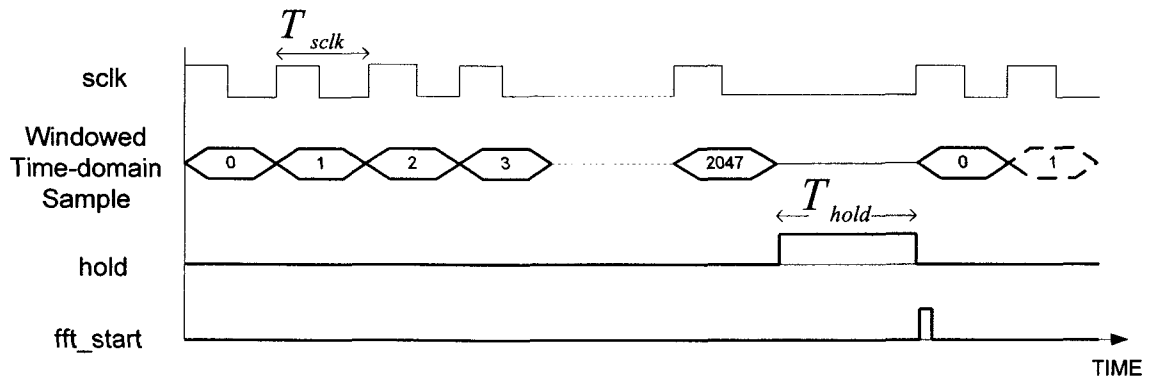


Figure 6.6: Timing diagram for SAMPLER module. When $hold = 1$ all windowed time-domain samples are fed to the FFT core. Values T_{sclk} and T_{hold} are presented in Table 6.16. The pulse widths are not drawn to scale.

6.1.1.4 Fast Fourier Transform Core (FFT)

This module contains a 2048-point FFT core generated using Xilinx Core Generator, which is part of the Xilinx ISE Design Suite 11.5 package. Xilinx FFT v7.0 (version 7.0) has been used in this thesis. Core Generator offers fully customizable, high-performance, parameterized signal processing IP cores from Xilinx. The parameters used for the FFT core implemented in this thesis are displayed in Table 6.4. The Xilinx ISE block for the FFT is shown in Figure 6.7.

Table 6.4: Xilinx FFT IP core parameterization

Parameter	Value
FFT size	2048
Architecture type	Burst I/O ¹
Radix	Mixed 2/4
Input word length	12 bits
Output word length	12 bits (scaled)
Scaling type	Rounding
I/O data type	2's complement
Internal phase factor length	16 bits ²

¹ refer to FFT datasheet from Xilinx [52]. The two available architectures are Burst I/O and Streaming I/O. Burst I/O architecture has been chosen due to its lower resource consumption.

² this parameter affects the precision of the FFT calculation. 16 bits was chosen for the phase factor word length as a trade-off between accuracy and resource usage.

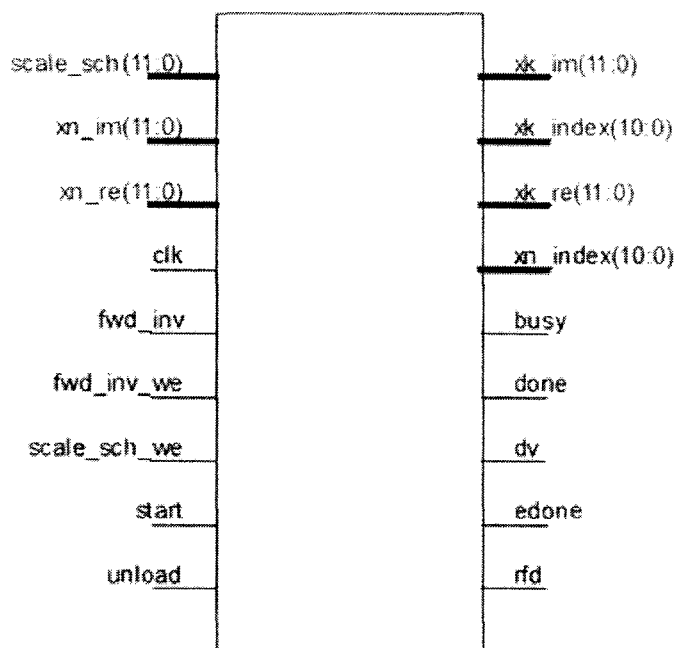


Figure 6.7: Xilinx ISE RTL view of FFT v7.0 core.

Table 6.5: Port description for FFT

HDL port name	Direction	Description
<code>scale_sch</code> ²	Input from TLC	Scaling schedule for all stages of the FFT – a default value of (0110 1010 1010) ₂ has been used ¹
<code>xn_im</code>	Input from TDR	Imaginary part of the time-domain sample
<code>xn_re</code>	Input from TDR	Real part of the time-domain sample
<code>clk</code>	Input from TLC	Operating clock at 100 MHz
<code>fwd_inv</code> ²	Input from TLC	'1' for FFT, '0' for IFFT (inverse FFT)
<code>fwd_inv_we</code> ²	Input from TLC	Write enable for <code>fwd_inv</code>
<code>scale_sch_we</code> ²	Input from TLC	Write enable for <code>scale_sch</code>
<code>start</code>	Input from TDR	Start signal initiates FFT computation
<code>unload</code>	Input from FDR	Signal to start unloading result from FFT
<code>xk_im</code>	Output to FDR	Imaginary part of frequency-domain FFT result
<code>xk_index</code>	Output to FDR	Index of frequency-domain sample being unloaded

xk_re	Output to FDR	Real part of frequency-domain FFT result
xn_index	Output to TDR	Index of time-domain sample being loaded
busy	Output (unconnected)	Active-high busy signal
done	Output to FDR	Active-high completion signal
dv	Output to FDR	Active-high data valid pin – logic '1' while unloading FFT results
edone	Output (unconnected)	Early completion signal – goes to logic '1' one clock cycle before <i>done</i>
rfd	Output to TDR	Ready For Data – logic '1' when FFT core is ready to accept new batch of time-domain data for processing

¹ refer to Xilinx FFT datasheet [52]. The scaling schedule specifies the number of bits to be scaled at the end of each internal FFT stage. This scaling ensures the same output word length as the input, in this case 12 bits.

² these signals offer run-time configurability to the FFT core.

The timing diagram for the Xilinx FFT core is shown in Figure 6.8 below.

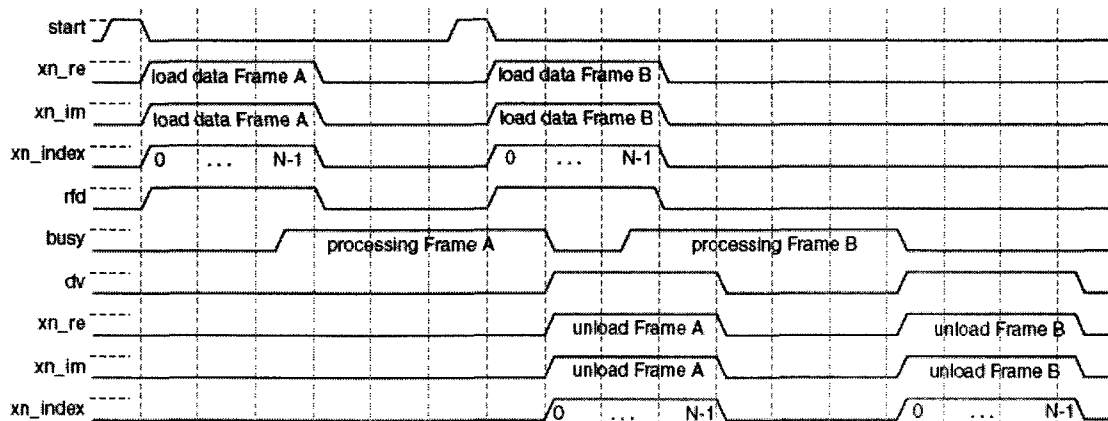


Figure 6.8: Timing diagram for Xilinx FFT core v7.0 (refer to datasheet in reference [52]).

6.1.1.5 Frequency-Domain Data RAM (FDR)

This unit is made of two sub-modules. The first sub-module monitors the *done* signal from the FFT core to be asserted, upon which it requests the FFT core to start unloading the result of the FFT by asserting the *unload* signal of the FFT core. The sub-module then accepts the frequency-domain samples from the FFT once the *DV (data valid)* signal from the FFT core is asserted, converts the 2's complement samples into positive values. This gives the *absolute* value of each real and imaginary frequency sample, setting up the next stage of the signal processing which deals with peak intensity calculation for each complex frequency sample.

The second sub-module contains two Block RAMs, one each for real and imaginary samples from the FFT. Only the latter half of the FFT results is stored due to the observation that the first half of the Xilinx FFT core has more noise and inaccuracy than the latter half. The fact that the FFT of a real-valued signal is symmetric about the central frequency bin allows the first half of the frequency-domain data to be ignored. Each stored sample is 12 bits in length, therefore the total RAM used is:

$$2 \times 12 \times 1024 \text{ bits} = 24 \text{ kb}$$

Once all 1024 frequency-domain samples have been retrieved and stored, the second sub-module of FDR begins the peak intensity calculation procedure by squaring the real and imaginary parts, summing them up and passing them to the PSD module. Figure 6.9 shows the RTL view of the two sub-modules forming the FDR unit and Table 6.6 lists the port descriptions. Figure 6.10 illustrates the timing of events related to the FDR module.

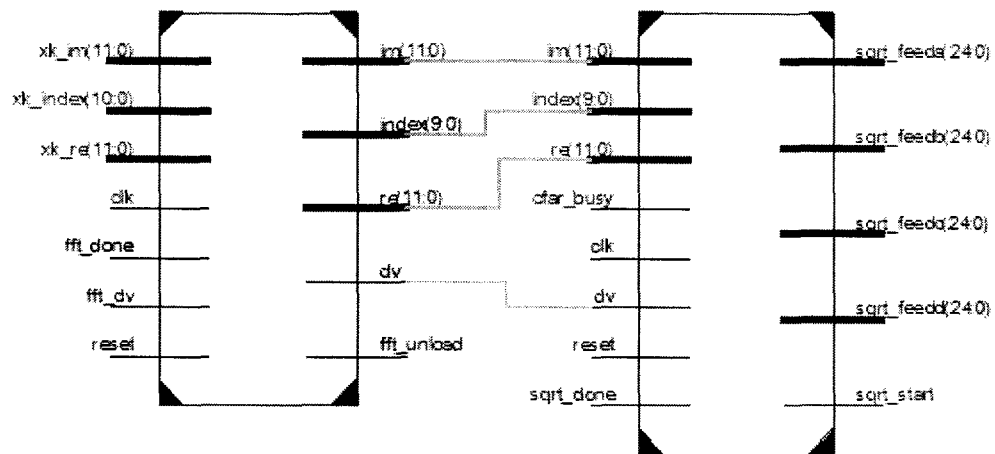


Figure 6.9: Xilinx ISE RTL schematic view of two sub-modules forming the FDR unit.

Table 6.6: Port description for FDR

HDL Port Name	Direction	Description
xk_im	Input from FFT	2's complement imaginary part of complex frequency-domain sample from FFT core
xk_index	Input from FFT	Index of frequency-domain sample being unloaded from the FFT core
xk_re	Input from FFT	2's complement real part of complex frequency-domain sample from FFT core
clk	Input from TLC	Operating clock of 100 MHz
fft_done	Input from FFT	FFT completion signal from FFT core
fft_dv	Input from FFT	Signal is logic '1' when valid output data is being unloaded from the FFT core
reset	Input	Global synchronous reset
cfar_busy	Input from CFAR	Busy signal from the CFAR unit – logic '1' causes FDR and PSD units to halt
sqrt_done	Input from PSD	Completion signal from PSD module
sqrt_feeda/b/c/d	Output to PSD	Four complex values sent per clock cycle to PSD unit
sqrt_start	Output to PSD	Signal asserted to instruct PSD module to commence peak intensity computation

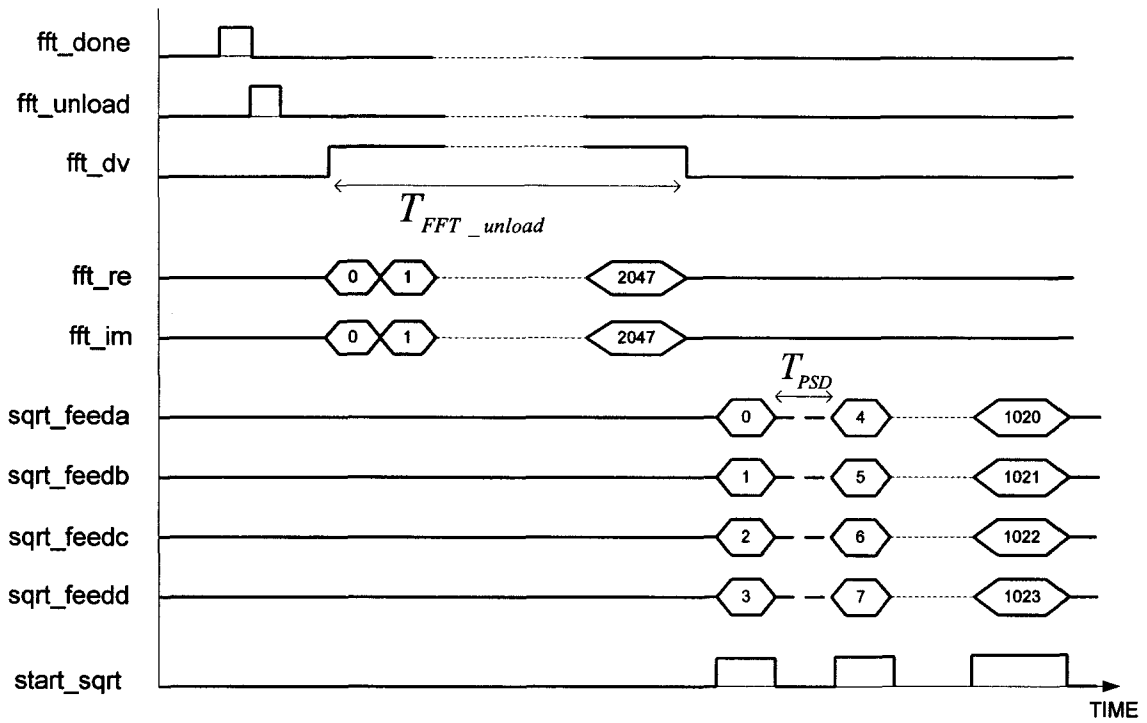


Figure 6.10: Timing diagram for FDR. [T_{FFT_unload} = 1024 clock cycles] and T_{PSD} is defined in Table 6.16.

6.1.1.6 Peak Intensity Calculator (PSD)

The PSD module computes the peak intensities of all the 1024 captured FFT output samples. It processes one sample at a time upon assertion of the *sqrt_start* signal. The signal processing algorithm contains 4 of these modules operating in parallel, allowing faster processing of all 1024 frequency-domain samples. Buses *sqrt_feeda/b/c/d* from the FDR are each inputs to one of these PSD modules. Once the peak intensity is computed, it is passed through a square-law detector unit which essentially ensures that no peak intensity value is negative before being passed to the CFAR processor. The positive-valued, frequency-domain peak intensity is sent to the CFAR processing module in groups of 4.

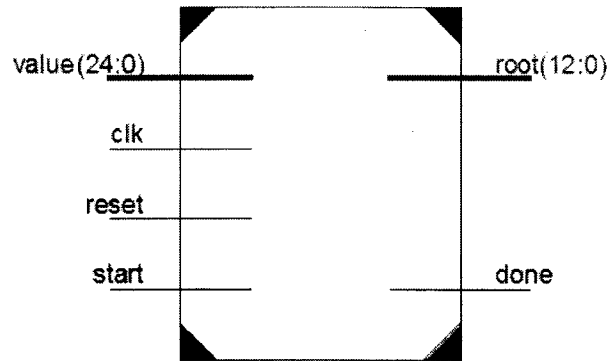


Figure 6.11: Peak intensity calculation unit.

Table 6.7: Port description for PSD

HDL Port Name	Direction	Description
value	Input from PSD units	Mapped to sqrt_feeda/b/c/d from FDR
clk	Input from TLC	Operating clock at 100 MHz
reset	Input	Global synchronous reset
start	Input from FDR	Start peak intensity computation signal from FDR
root	Output to CFAR	Peak intensity computation result to CFAR unit
done	Output to FDR	Completion of peak intensity calculation

6.1.1.7 Constant False Alarm Rate Processor (CFAR)

The CA-CFAR algorithm has been detailed in previous chapters of this thesis. The HDL implementation of the CA-CFAR algorithm is a vital component of the radar signal processing algorithm. It is solely responsible for removal of unwanted clutter and noise while detecting valid targets from an unknown attenuation pattern arising from different weather conditions.

The CFAR processor receives frequency-domain peak intensity values in batches of 4 from the 4 PSD units working in parallel, as shown in Figure 6.12. These 4 values are stored in a Block RAM in the following order:

Result of *sqrt_feeda* stored in index 0 of Block RAM.

Result of *sqrt_feedb* stored in index 1 of Block RAM.

Result of *sqrt_feedc* stored in index 2 of Block RAM.

Result of *sqrt_feedd* stored in index 3 of Block RAM.

In the similar order, the next 4 received peak intensity values from the 4 PSD units are stored in index 4, 5, 6 and 7 of the RAM. The RTL block diagram for the CFAR processor is shown in Figure 6.13, and the port description is provided in Table 6.8. The timing diagram depicting the operation of the CFAR unit is shown in Figure 6.14.

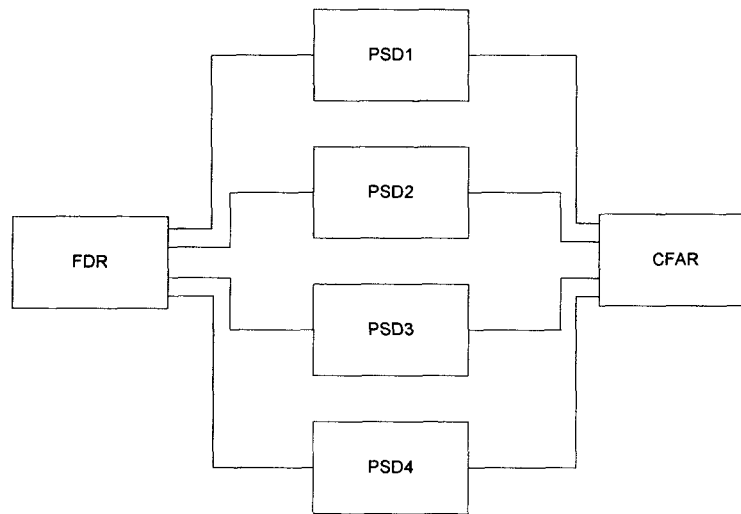


Figure 6.12: Four PSD units work in parallel to speed up peak intensity computation.

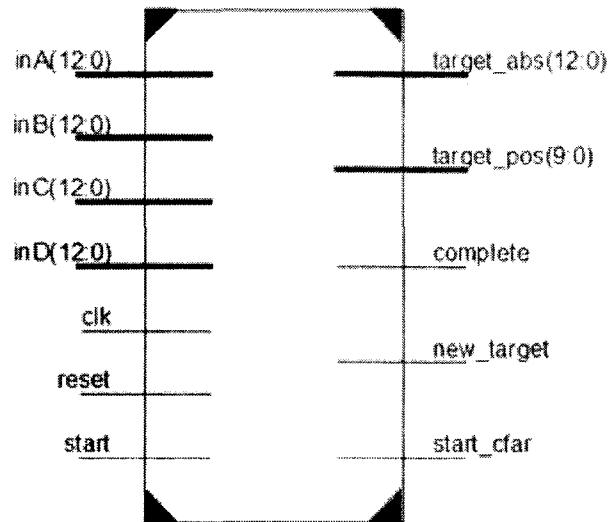


Figure 6.13: RTL view of CFAR module.

Table 6.8: Port description for CFAR

HDL Port Name	Direction	Description
inA/B/C/D	Input from PSD	Peak intensity values from 4 parallel PSD units
clk	Input from TLC	Operating clock at 100 MHz
reset	Input	Global synchronous reset
start	Input from PSD	Active-high signal that is logic '1' when new peak intensity values are available to be read from the PSD units
target_abs	Output to PPM	Peak intensity of detected target output to Peak Pairing module
target_pos	Output to PPM	Spectral position (FFT bin number) of detected target output to Peak Pairing module
complete	Output to PPM	CFAR completion signal for all 1024 values
new_target	Output to PPM	Active-high signal that is logic '1' to alert the Peak Pairing module when a new valid target is detected
start_cfar	Output to FDR	Mapped to <i>cfar_busy</i> signal to FDR indicating CFAR unit is busy

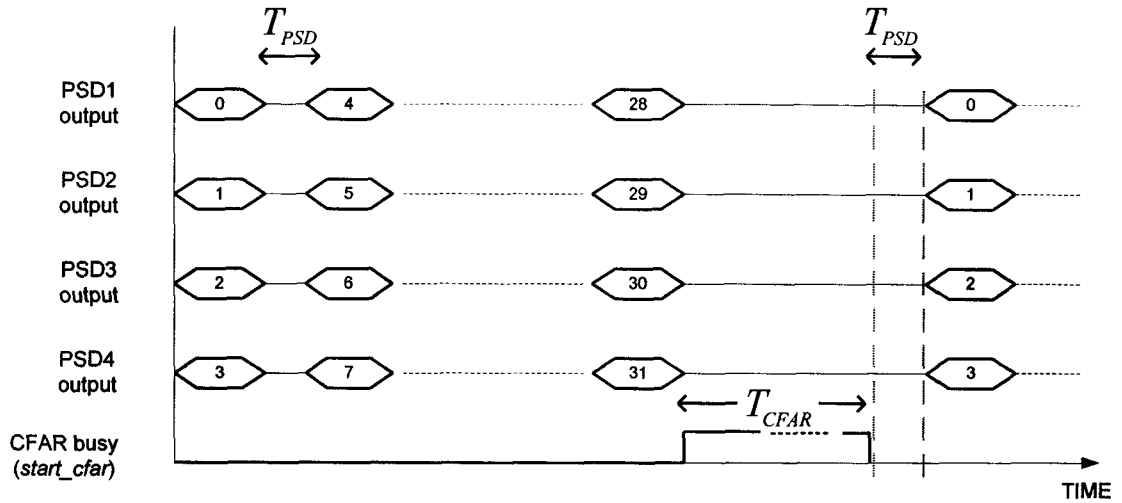


Figure 6.14: Timing diagram for CFAR module: 32 peak intensity values are collected by the CFAR unit from 4 PSD units working in parallel. For processing delays T_{PSD} and T_{CFAR} refer to Table 6.16.

Reasons for processing 32 frequency-domain values at a time:

1. Lower memory requirements for the CFAR module.
2. Reduce complexity and improve speed in CFAR module.

6.1.1.7.1 Important modification to the CA-CFAR processor

Due to atmospheric attenuation targets far away appear with smaller peak intensities. Low power peaks were observed in the CFAR when modeling far away targets, and in some cases this led to their exclusion by the CA-CFAR process. In order to overcome this problem, the sensitivity of the CFAR processor was increased for medium-range and long-range targets by reducing the P_{fa} used to compute the constant K . The adjustments are presented in Table 6.9. This approach increased the detection rate for medium- and long-range targets.

Table 6.9: Sensitivity Adjustment for CA-CFAR Processor

Radar range	FFT bin range	Corresponding range (m)	P_{fa}	Constant K^1
Short	1 – 512	0.186 – 95.136	10^{-7}	6.499
Medium	513 – 852	95.322 – 158.312	10^{-6}	4.623
Long	853 – 1024	158.498 – 200.000	10^{-5}	3.217

¹ As mentioned in Chapter 4, cell-averaging depth is 4 on either side of CUT i.e. $M = 8$. These values of K have been rounded off in the fixed-point HDL implementation.

6.1.1.8 Peak Pairing Module (PPM)

The Peak Pairing unit was implemented *as is* from the MATLAB model of the radar signal processing algorithm. The criteria of peak pairing used are Spectral Proximity and Power Level comparison as described in Chapter 4. Figure 6.15 and Figure 6.16 display the Xilinx RTL view and the timing diagram for the PPM, respectively. Table 6.10 provides port descriptions for the module. The output of the PPM is the target range and velocity information already described in the TLC section of this chapter.

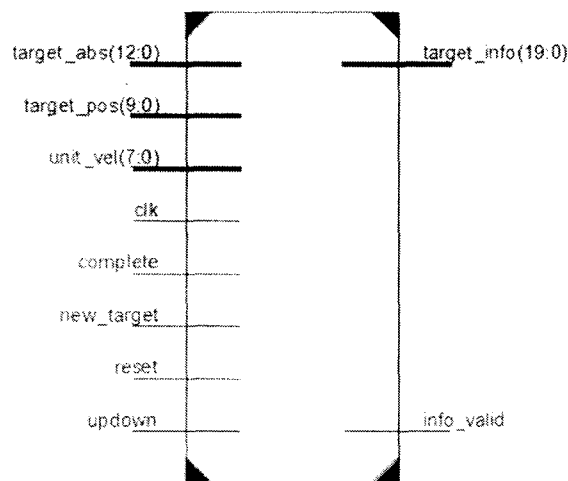


Figure 6.15: RTL view of PPM.

Table 6.10: Port description for PPM

HDL Port Name	Direction	Description
target_abs	Input from CFAR	Peak intensity of a detected target by CFAR
target_pos	Input from CFAR	Spectral position of a detected target by CFAR
unit_vel	Input from TLC	Velocity of host vehicle in km/h
clk	Input from TLC	Operating clock at 100 MHz
complete	Input from CFAR	Completion signal for CFAR processing of all 1024 frequency-domain samples
new_target	Input from CFAR	Active-high signal that is logic '1' when a new valid target is detected by CFAR
reset	Input	Global synchronous reset
updown	Input from TLC	Is equal to logic '1' during a positive frequency chirp and logic '0' during a negative frequency chirp
target_info	Output to TLC	Bus containing computed target information with most significant 10 bits for target velocity, next 10 bits for target range, and final 2 bits for beam number in which the target was detected
info_valid	Output to TLC	Active-high signal that is at logic '1' when new target information is available to the TLC

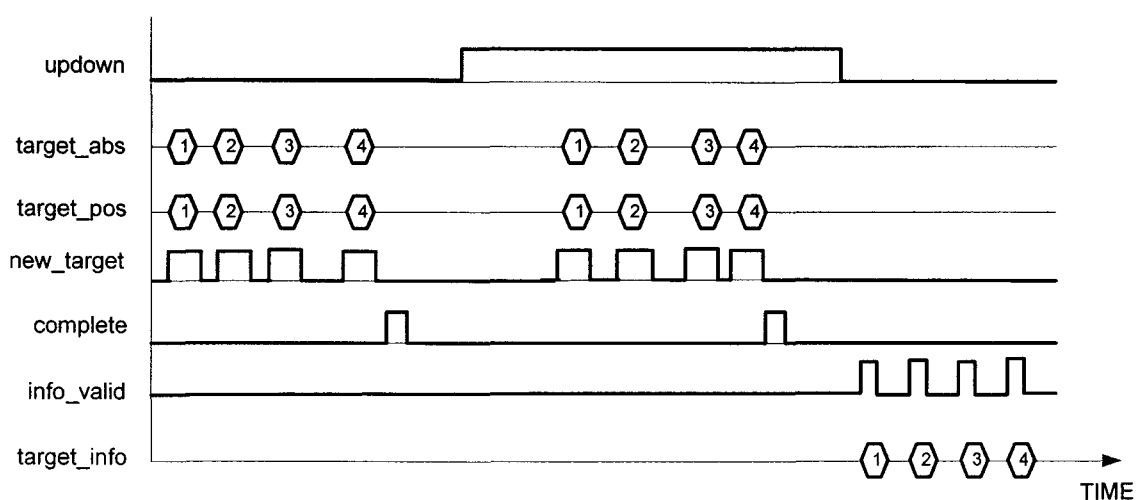


Figure 6.16: Timing diagram for PPM showing 4 detected targets from CFAR.

As illustrated in Figure 6.16, the Peak Pairing module collects peaks from the CFAR processor for both the up and down frequency sweeps. Once all target peaks and spectral positions have been received from the CFAR, the LFM CW equations for range and velocity are applied to retrieve the target information and output it over the *target_info* bus.

Let us re-state the range and velocity equations from Chapter 3:

$$\text{Range, } r = \frac{(f_{\text{up}} + f_{\text{down}})}{2} \times \frac{c}{2k}$$

$$\text{Velocity, } v_r = \frac{(f_{\text{up}} - f_{\text{down}})}{4} \times \frac{c}{f_0}$$

Doing the multiplications and divisions to calculate range and velocity would be hardware in-efficient and consume more clock cycles. An easier method is to pre-calculate the factors for range and velocity so that a direct multiplication with the sum and difference of the target spectral positions (or frequency bin numbers) would generate the target range and velocity, respectively.

The range factor for the implemented system parameters is:

$$rf = \frac{1}{2} \times \frac{c}{2k} \times F_{\text{res}} = \frac{2.973 \times 10^8}{4 \times \frac{800 \times 10^6}{1.024 \times 10^{-3}}} \times \frac{2 \times 10^6}{2048} \quad (6.1)$$

$$= 0.09290625$$

Here, F_{res} is the frequency resolution of the FFT core.

This value has been approximated as an 11-bit number equal to $(00010111110)_2$, where all bits represent the fractional part. This sequence thus corresponds to a decimal value of 0.0927734375.

Similarly, the velocity factor is:

$$vf = \frac{1}{4} \times \frac{c}{f_0} \times F_{\text{res}} \times 3.6 = \frac{1}{4} \times \frac{2.973 \times 10^8}{76.9 \times 10^9} \times \frac{2 \times 10^6}{2048} \times 3.6 \quad (6.2)$$

$$= 3.39790414$$

Here, the value of **3.6** has been multiplied here to convert the calculated velocity from m/s into km/h. The central frequency for the LFM CW chirps has been set to 76.9 GHz, as the TLC VCO permits a sweep range of 76.5 GHz – 77.3 GHz to form a bandwidth of 800 MHz.

This value has been approximated by a 7-bit binary number equal to (1101101)₂ where the first 2 bits represent the integer part and the last 5 bits represent the fractional part, corresponding to a decimal value of 3.40625.

6.2 Simulation and Validation of the HDL Implementation of the Signal Processing Algorithm

To accomplish simulation and validation of the entire HDL implementation and ensure readiness of the Verilog HDL code for downloading to the Virtex-5 FPGA, the following steps were followed:

1. All individual modules are assembled to form the top level control module TLC.
2. A Verilog test-bench is coded to run tests on the TLC module.
3. Time-domain samples of the intermediate frequency generated from the traffic scenarios presented in Chapter 5 are extracted in hexadecimal format from MATLAB. A total of 2048 samples are extracted.
4. The time-domain samples are passed to the TLC through the test-bench, thus imitating the external ADC at 2 MSPS sampling rate.

5. The simulation test-bench is run in Xilinx ISE Simulator and the resultant waveforms are observed for the output of the TLC.
6. The results are compared to the actual parameters of the simulated targets.

The test on the HDL modules involved the same scenarios used to verify the signal processing algorithm in Chapter 5.

6.2.1 Test 1: 3-Lane Highway Scenario with Narrow Beam

Recall the test scenario presented in Figure 6.17. The HDL design was clocked at 100 MHz and tested for timing compliance with the desired 1 ms up or down sweep time as part of the target MEMS radar specifications.

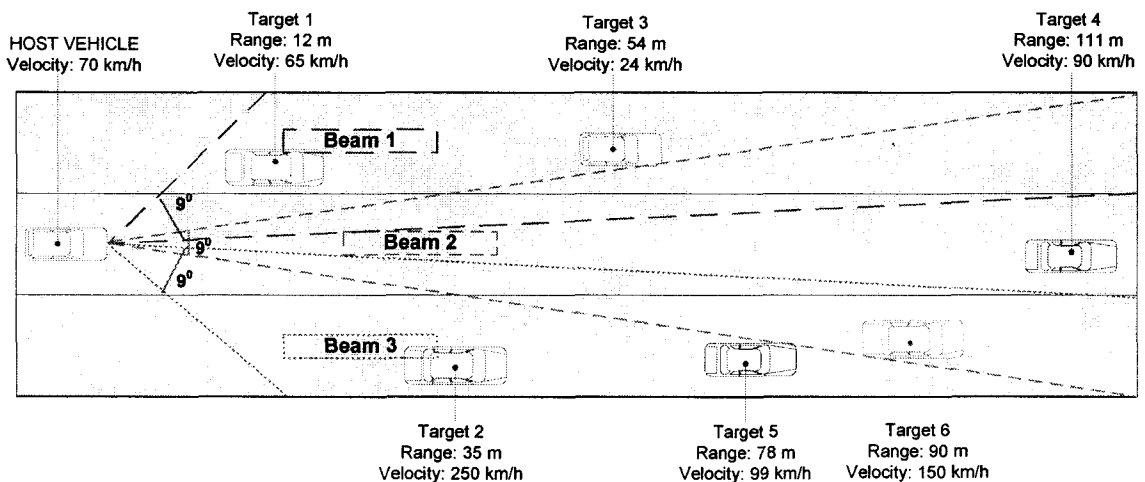


Figure 6.17: Test case highway scenario. Beam 1 shines 2 targets, Beam 2 covers 2 targets, and Beam 3 covers 3 of the targets. Beam width for the antenna is assumed to be 9°, with 4.5° Rotman lens beam steering.

The test-bench is coded to display the timing of each major event. The following is the output from the Xilinx ISim Simulation:

```
up sampling start: 110
up sampling done: 1023890
down sampling start: 1044610
down sampling done: 2068150
beam 1 first target info out: 2259300

up sampling start: 2259300
up sampling done: 3303390
down sampling start: 3303400
down sampling done: 4347650
beam 2 first target info out: 4347820
...
up sampling start: 4347820
up sampling done: 5391910
down sampling start: 5391920
down sampling done: 6436170
beam 3 first target info out: 6436380
```

The numerical values in the output are the exact time in nanoseconds at which the labeled event occurred. Therefore, sampling 1024 time-domain values took 1023780 ns or 0.1024 ms approximately, which is the expected sampling duration.

Additionally, this timing information gives the total time taken for 1 beam to be scanned and all target information to be output from the PPM. Start of sampling the up frequency sweep for beam 1 is at 110 ns, and the first target information for beam 1 is output at 2259300 ns, thus a total time of 2259190 ns or 2.26 ms approximately. This confirms that a total processing latency of less than 0.25 ms per beam has been achieved.

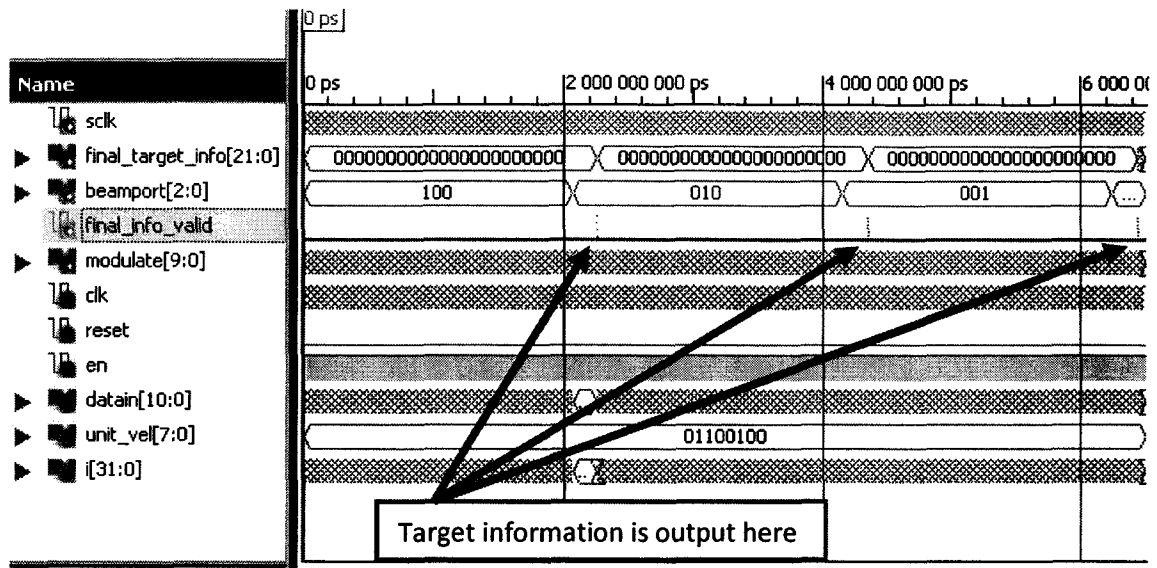


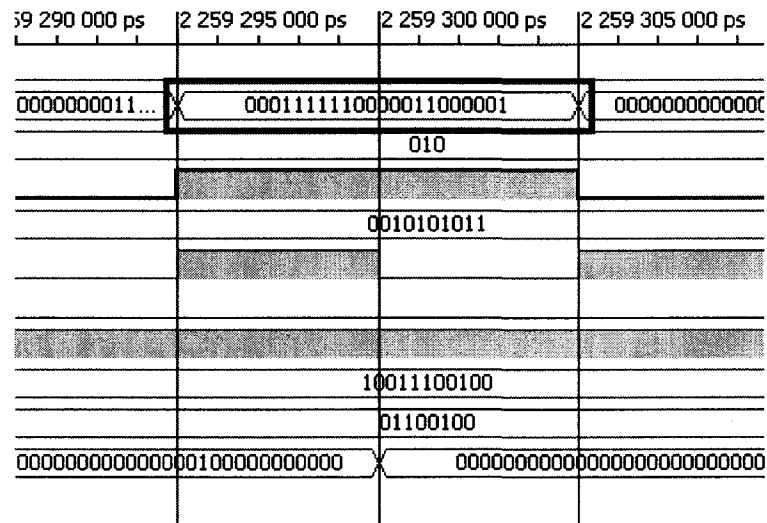
Figure 6.18: 3-Lane simulation waveform results from Xilinx ISE Simulator. The spikes on the *final_info_valid* signal show the positions where new target information is available.

Figure 6.18 shows target detection, and also shows the change in the 3-pin beam port control bus responsible for controlling the MEMS SP3T switches. The control signals are accurate and occur at the correct time. On the left hand side of the figure the list of displayed variables is as follows:

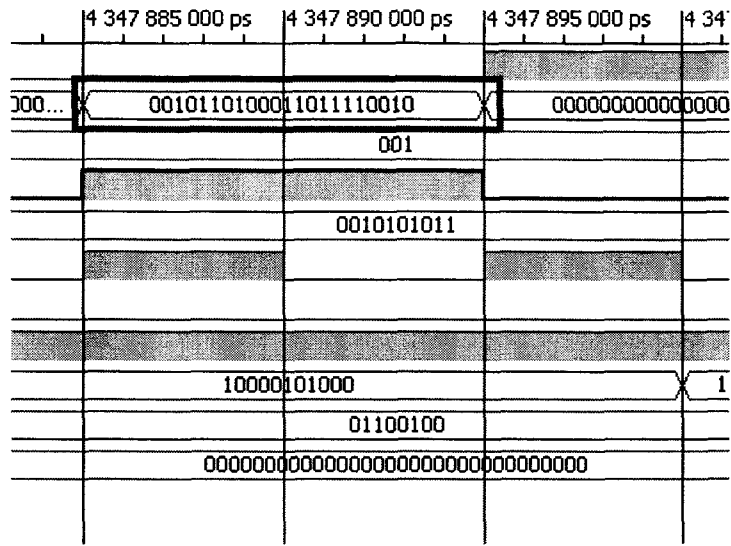
1. *sclk*: Sampling clock
2. *final_target_info*: 22-bit target information
3. *beamport*: 3-bit control bus for MEMS SP3T switches to control beam direction through the MEMS Rotman lens
4. *final_info_valid*: Signal goes to logic '1' when new target information is output
5. *modulate*: the 10-bit counter output to the DAC which forms the up and down sweeps for the VCO tuning voltage
6. *clk*: Operating clock of 100 MHz
7. *reset*: Global synchronous reset

8. *en*: System enable signal
9. *datain*: MATLAB samples are input via this port to the TLC, imitating time-domain ADC samples
10. *unit_vel*: This is the host vehicle velocity, which has been set to $(0110\ 0100)_2$ or 100 km/h
11. *i*: An index variable used in the Verilog test-bench code

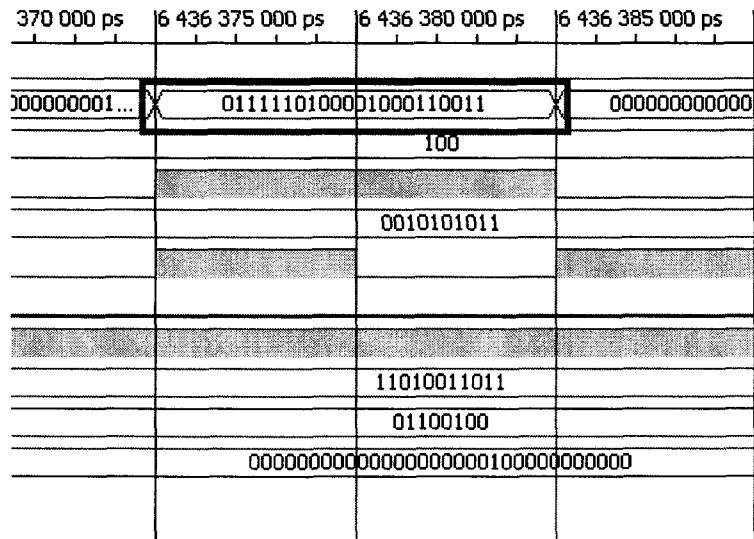
The results for the range and velocity measurements obtained from HDL simulation are illustrated in binary format in Figure 6.19, and tabulated in Table 6.11.



6.19(a): Beam 1, Target 1



6.19(d): Beam 2, Target 4



6.19(e): Beam 3, Target 2

The 22-bit target information is understood as follows:

$$(0111110100\ 0010001100\ 11)_2$$

$$\begin{aligned}\text{Most significant 10 bits} &= \text{velocity of Target 2 in 9-integer-1-fractional bit format} \\ &= (011111010)_2.(0)_2 \\ &= 250.0\text{ km/h}\end{aligned}$$

$$\begin{aligned}\text{Next 10 significant bits} &= \text{range of Target 2 in 8-integer-2-fractional bit format} \\ &= (00100011)_2.(00)_2 \\ &= 35.00\text{ m}\end{aligned}$$

In a similar fashion, all detected target ranges and velocities can be computed. These have been listed in Table 6.11.

Table 6.11: Results from HDL Simulation of the Developed Algorithm for 3-Lane Narrow Beam Scenario

Beam Port Number	Target ID	Measured Up Sweep IF (frequency bins) ¹	Measured Down Sweep IF (frequency bins) ¹	Measured Range (m)	Measured Velocity (km/h)
1	1	71	60	12.00	63.0
	3	303	280	54.00	22.0
2	4	478	493	111.00	90.0
	6	600	597	90.00	151.0
3	2	167	211	35.00	250.0
	5	421	421	78.00	100.0
	6	478	497	90.00	151.5

¹ Frequency resolution for 2048-point FFT = 976.5625 Hz/bin

Table 6.12: Errors for the Developed Algorithm from HDL Simulations of 3-Lane Narrow Beam Scenario (SNR = 4.73dB)

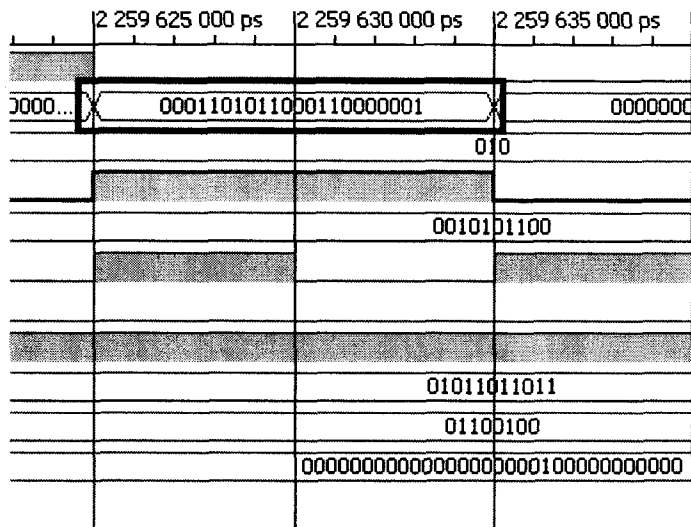
Beam Port Number	Target ID	Error in Range Measurement (m)	Error in Velocity Measurement (km/h)
1	1	0.00	2.0
	3	0.00	2.0
2	4	0.00	0.0
	6	0.00	1.0
3	2	0.00	0.0
	5	0.00	1.0
	6	0.00	1.5

Maximum error in range measurement for the developed algorithm: 0.00 m

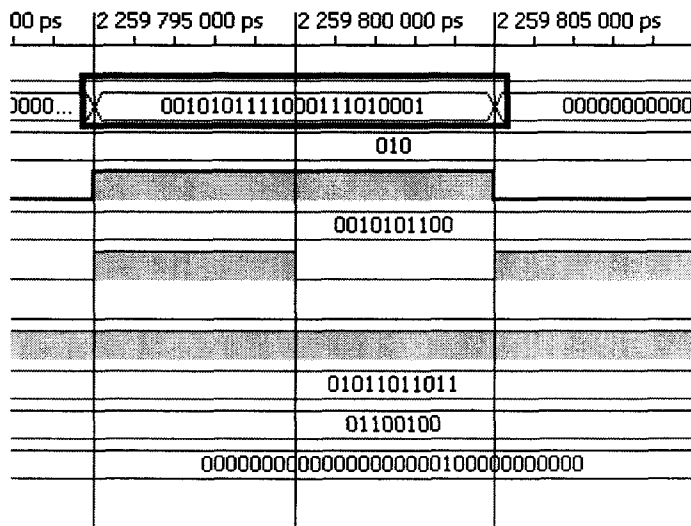
Maximum error in velocity measurement: 1.50 km/h

6.2.2 Test 2: Hypothetical Scenario with 7 Targets Detected in a Single Wide Beam

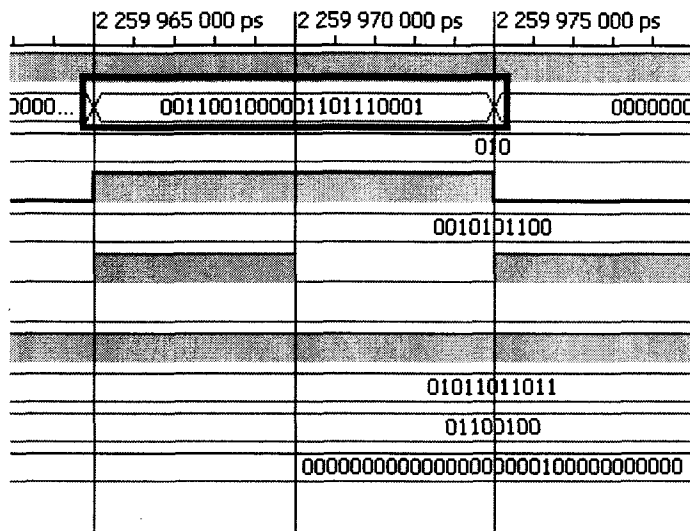
Figure 6.20 shows the scenario in consideration. It is a replica of the test carried out on the MATLAB model of the radar signal processing unit. The target ranges and velocities have been selected randomly to ensure fair testing. Through the verification process several target configurations were tested using randomly generated targets spread over the allowable range for the developed system, and the results presented in this chapter have been obtained after 6 iterations for each scenario. This is applicable for both Test 1 and Test 2 cases.



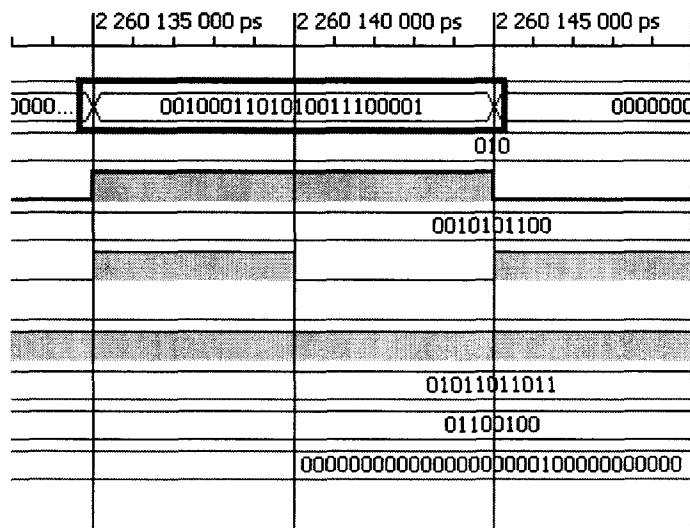
6.21(b): Target 2



6.21(c): Target 3



6.21(d): Target 4



6.21(e): Target 5

Table 6.13: Results from HDL Simulations of the Developed Algorithm for 3-Lane Single Wide Beam Scenario

Target ID	Measured Up Sweep IF (frequency bins) ¹	Measured Down Sweep IF (frequency bins) ¹	Measured Range (m)	Measured Velocity (km/h) ²
1	46	53	9.00	123.5
2	137	123	24.00	53.5
3	159	155	29.00	87.5
4	297	297	55.00	100.0
5	425	416	78.00	70.5
6	574	569	106.00	83.0
7	809	786	147.75	22.0

¹ Frequency resolution for 2048-point FFT = 976.5625 Hz/bin

² Target velocity has been calculated using equation (3.16)

Table 6.14: Errors for the Developed Algorithm from HDL Simulations for 3-Lane Single Wide Beam Scenario (SNR = 4.73dB)

Target ID	Error in Range Measurement (m)	Error in Velocity Measurement (km/h)
1	0.00	0.5
2	0.00	1.5
3	0.00	1.5
4	0.00	0.0
5	0.00	0.5
6	0.00	3.0
7	0.25	0.0

Maximum error in range measurement for the developed algorithm: 0.25 m

Maximum error in velocity measurement: 3.00 km/h

At this point a comparison can be made between the MATLAB simulation results and the HDL simulation results for the developed radar signal processing algorithm. HDL results are seen to be in accordance with software simulation results, and this proves the mathematical accuracy of the developed hardware system on FPGA. Table 6.15 and Table 6.16 show the difference between MATLAB and HDL results for range and velocity, respectively, for the wide beam scenario presented in Figure 6.20.

Table 6.15: Comparison of MATLAB and HDL range results for wide beam scenario

Target ID	Target Distance from Host Vehicle (m)	MATLAB calculated value (m)	HDL determined value (m)	Δ MATLA-Actual (m)	Δ HDL-Actual (m)	Δ MATLAB-HDL (m)
1	9.00	9.38	9.00	0.38	0.00	0.38
2	24.00	24.34	24.00	0.34	0.00	0.34
3	29.00	29.27	29.00	0.27	0.00	0.27
4	55.00	55.37	55.00	0.37	0.00	0.37
5	78.00	78.32	78.00	0.32	0.00	0.32
6	106.00	106.28	106.00	0.28	0.00	0.28
7	148.00	148.37	147.75	0.37	0.25	0.62

Table 6.16: Comparison of MATLAB and HDL velocity results for wide beam scenario

Target ID	Target Velocity relative to Host Vehicle (km/h)	MATLAB calculated value (km/h)	HDL determined value (km/h)	Δ MATLA-Actual (km/h)	Δ HDL-Actual (km/h)	Δ MATLAB-HDL (km/h)
1	123	123.85	123.5	0.85	0.5	0.35
2	55	52.31	53.5	2.69	1.5	1.19
3	89	89.78	87.5	0.78	1.5	2.28
4	100	100.00	100.0	0.00	0.0	0.00
5	70	69.34	70.5	0.66	0.5	1.16
6	80	79.56	83.0	0.44	3.0	3.44
7	22	21.64	22.0	0.36	0.0	0.36

From Table 6.15 and Table 6.16 it can be concluded that the HDL results are in good accordance with the MATLAB results, and have higher accuracy compared to MATLAB results. This is due to the quantization involved in fixed-point HDL. The maximum measured range discrepancy between MATLAB and HDL is 62 cm, and the maximum measured velocity difference is 3.44 km/h or 0.95 m/s.

6.3 Hardware Synthesis Results for the Developed Algorithm

Table 6.15 lists the resource usage for the developed HDL design of the signal processing algorithm. The target device has been selected as the Virtex-5 SX50T FPGA. Table 6.16 lists the timing achievements of the HDL implementation.

Table 6.17: Resource Usage for the Radar Signal Processing Algorithm on Virtex-5 SX50T

Resource	Used	Available	Percentage Usage
Slice registers	1357	32640	4%
Slice LUTs	7445	32640	23%
DSP48E slices	17	288	6%
Fully used LUT-FF pairs	705	8097	9%
BUFG/BUFGCTRLs	1	32	3%
FPGA fabric area ratio	21	100	21%

Table 6.18: Timing Achievements of HDL Implementation

Operation	Effective Clock Cycles per Beam	Latency per Beam with Operating Clock at 100 MHz (ms)
Up sweep sampling ($T_{\text{sclk}} = 0.5 \mu\text{s}$)	204756	2.047560
Window and feed time- domain samples to FFT core (T_{hold})	2072	0.020720
FFT calculation	3960	0.039600
Peak intensity calculation with 4 PSD units in parallel (T_{PSD})	10743	0.107430
CFAR processing and Peak Pairing (T_{CFAR})	4388	0.060460
Total Signal Processing Latency	21163	0.211630
Overall Latency	225928	2.259280

6.4 Observations from HDL Implementation of the Developed Algorithm

The following noteworthy observations have been made about the HDL implementation of the radar signal processing algorithm:

1. The worst case range measurement error is seen to be 0.25 m. This can be further reduced by increasing the word length of the range output, which is currently restricted to 10 bits.
2. The worst case velocity measurement error is noted to be 3 km/h, which corresponds to 0.83 m/s. This error is within tolerance limits of the automotive radar arena, however can be improved further by making use of more bits for the output result.
3. Proper synchronization of the modules has been achieved.
4. The HDL design can operate at a maximum of 160 MHz, although a 100 MHz operating frequency is selected for ease of clock generation.
5. Generation of the modulating waveform data to the DAC operates as required.
6. The sampling clock is tuned at 2 MHz and the TLC unit samples over 1.024 ms to gather a total of 2048 time-domain samples.
7. The HDL design operates within the time frame of 1.024 ms, and gives a result for a single beam scan in less than 0.22 ms as shown in Figure 6.22.
8. The HDL results are within acceptable error limits compared to the MATLAB results, thus validating the HDL implementation of the algorithm. Due to truncation and rounding used in the fixed-point HDL implementation, the HDL code appears to generate better results compared to the floating-point MATLAB model. This was seen to be true over 6 iterations of running the system on the same time-domain data, however may or may not always hold true.

Table 6.19: Achieved Timing Details for Developed LFM CW Radar System

Parameter	Value
Up sweep duration	1.024 ms
Down sweep duration	1.024 ms
Maximum Design Operating Frequency	160 MHz (65-nm FPGA technology)
Processing Time per Beam (@ 100 MHz)	2.04756 ms sampling + 0.21163 ms processing = 2.25928 ms
Processing Time for 3 Beam RADAR	$2.25928 \text{ ms} \times 3 = 6.77784 \text{ ms}^1$ => 147 MHz refresh rate

¹ This value is assuming that the sweep generation is stalled during processing, which is not the case. In actual implementation, processing of the previous beam is done during the next sweep as shown in Figure 6.22. The actual time is $(2.048 + 0.020720) \times 3 + 0.211630 = 6.41779 \text{ ms}$.

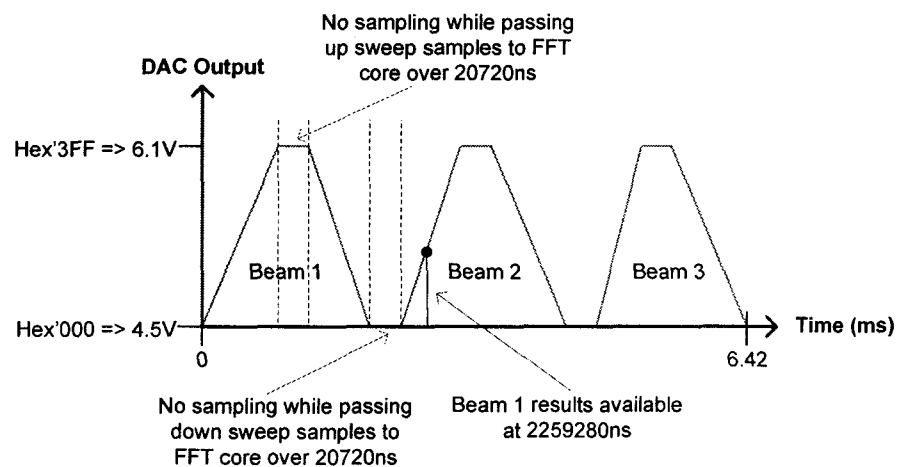


Figure 6.22: LFM CW sweep timing diagram for the realized HDL system.

7.1 Discussions and Conclusions

A Xilinx Virtex-5 SX50T FPGA platform targeted Verilog HDL based signal processing algorithm has been developed to process the drive, control and decision making signal processing tasks associated with a MEMS implemented Rotman lens based LFM CW long range radar to detect the velocity and range of target vehicles in typical highway conditions. Necessary building blocks of the complete system have been developed and implemented to realize a fast radar control and signal processing algorithm in hardware. Excellent agreement between the MATLAB implemented mathematical models and Verilog HDL code generated results verify the accuracy of the HDL modules. The developed Verilog HDL codes can be used to fabricate an ASIC that can be incorporated in a 3-D integrated complete radar system to realize a small form-factor low-cost automotive radar. A hardware latency time as low as 211.63 μ s clocked at 100 MHz has been achieved which is superior to state-of-the-art commercially reported radar systems. This is almost 3 times faster than a recent FPGA implementation presented in [28], where an LFM CW signal processing system has been implemented on a Xilinx Virtex-II Pro FPGA with a latency of 1250 μ s clocked at 50 MHz. The results for range and velocity calculations are promising and accurate with 100% detection in a tested SNR of 4.73 dB under an atmospheric attenuation of 0.8 dB/km corresponding to light or medium rain conditions. Swerling I, III and V type targets have been simulated. The maximum error in range measurement is 25 cm, and the maximum error in velocity measurement is 3 km/h or 0.83 m/s. The bandwidth of the LFM CW radar waveform is set to 800 MHz, and the radar algorithm is capable of covering a range of 200 meters with a maximum relative target velocity of ± 300 km/h (receding and approaching targets).

The excellent speed performance of the algorithm validates the use of FPGAs in radar signal processing and allows the MEMS radar sensor to operate with a cycle time of 6.78 ms for a 3-beam sensor, which is at least 7 times faster than the Bosch LRR3 [23]. Beam direction control by means of MEMS SP3T RF switches and a MEMS Rotman lens has been implemented in the radar algorithm and found to operate in coherence with the radar system specifications.

7.2 Future Work

This thesis opens the path to many additional features that can be added to the MEMS radar sensor system. The following are some of the exciting possible future developments to the field of automotive radar systems with regard to this thesis:

1. Accurate target angle measurement using an FPGA-based implementation of Direction-of-Arrival or DOA algorithms, such as Phase-Difference DOA estimation using double 1-D FFT [30], MUSIC [53], or ESPRIT [54].
2. Higher resolution of ADC input and target information output to improve range precision from 25 cm down to 5 cm and velocity precision from 0.5 km/h down to 0.125 km/h provided the sweep bandwidth is increased to 2 GHz and the sweep duration is increased to at least 6 ms.
3. Inculcate the ability to gather road clutter and create a virtual map of the road by smartly using clutter information to detect side fences and dividers along with vehicles, as presented in literature [47].
4. Use alternating frequency bands and bandwidths to increase chances of target detection and improve detection accuracy by comparing results from both bands.
5. Decrease the sweep duration to 0.5 ms and study the effect on signal processing accuracy and precision.

6. Implement an OS-CFAR module parallel to the CA-CFAR module developed herein in order to increase system fidelity by dynamic comparison of the results of both modules.
7. Estimate the RCS of a detected target in close proximity or *threat zone* of the host vehicle and compute the mass and impact force in case of collision.
8. Implementation of a multi-mode automotive radar system consisting of an SRR, MRR and LRR, as in Figure 7.1, running on the same processing unit and hardware. Such a system would be realizable by means of a reconfigurable antenna that can be controlled using the FPGA algorithm.
9. Implementation of a combined FSK-monopulse and LFM CW radar using the same hardware to improve the functional dimensions to realize a compact small form-factor cost-effective automotive radar.

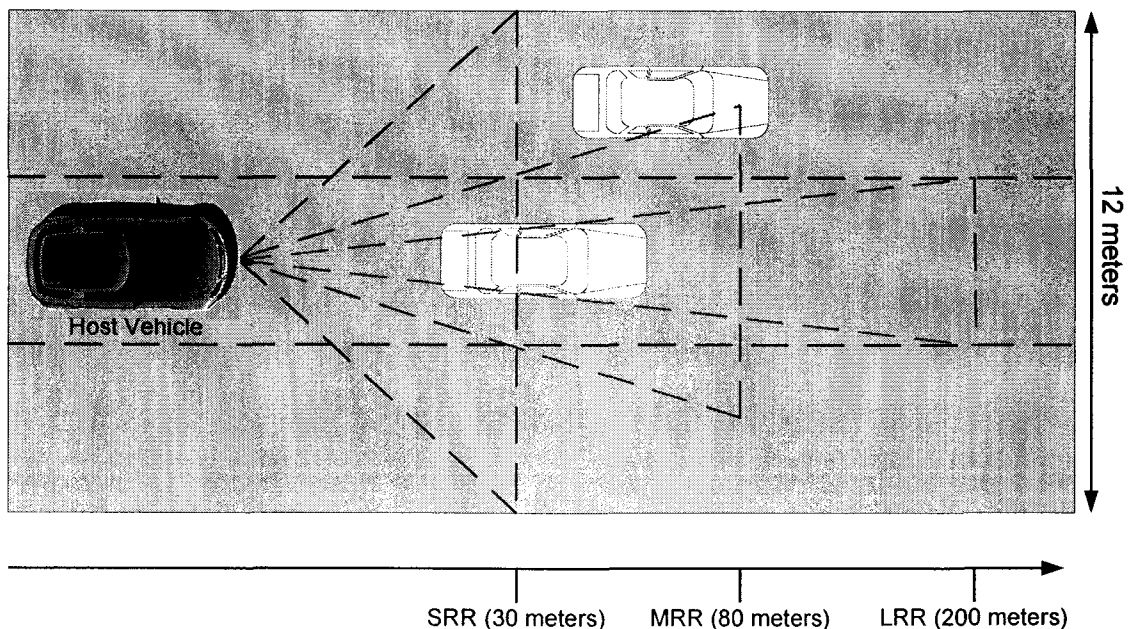


Figure 7.1: Typical angle and range coverage for forward-looking collision avoidance SRR, MRR and LRR over a 3-lane road.

REFERENCES

1. A. Sinjari, S. Chowdhury, "MEMS Automotive Collision Avoidance Radar Beamformer," in *Proc. IEEE ISCAS2008*, Seattle, WA, 2008, pp. 2086-2089.
2. Y. K. Chan, S. Y. Lim, "Synthetic Aperture Radar (SAR) Signal Generation," *Progress in Electromagnetics Research B*, Vol. 1, pp. 269-290, 2008.
3. R. A. Mucci, "A Comparison of Efficient Beamforming Algorithms," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. ASSP-32, No. 3, pp. 548-558, Jun. 1984.
4. T. Haynes. (1998, March 26). *A Primer on Digital Beamforming*, Spectrum Signal Processing. White paper [Online]. Available: http://www.spectrumsignal.com/publications/beamform_primer.pdf
5. C. Wolff. (2008). *Digital Beamforming* [Online]. Available: <http://www.radartutorial.eu/06.antennas/an51.en.html>
6. J. Bass, E. Rodriguez, J. Finnigan, C. McPheeters. (2005, July). *Beamforming Basics* [Online]. Available: <http://cnx.org/content/m12563/latest/>
7. S. Chowdhury, M. Ahmadi, G. A. Jullien, W. C. Miller, "A Surface Mountable MEMS Beamforming Microphone Array and Associated MEMS Socket Structure," in *Symp. Microelectronics Research and Development in Canada (MR&DCAN 2001)*, Ottawa, ON, 2001.
8. L. Hall, H. Hansen, D. Abbott, "Rotman Lens for mm-wavelengths," in *Proc. IEEE Smart Structures, Devices and Systems (SPIE2002)*, Vol. 4935, pp. 215-221, 2002.
9. M. I. Skolnik, *Radar Handbook*, 2nd ed., New York: McGraw-Hill, 1970.
10. BBC h2g2. (2003, July). *The History of Radar* [Online]. Available: <http://www.bbc.co.uk/dna/h2g2/A591545>
11. "Fundamentals of Radar", Encyclopædia Britannica Online, [2010 April 29]. Available: <http://www.britannica.com/EBchecked/topic/488278/radar>
12. H. Rohling, *Some Radar Topics: Waveform Design, Range CFAR and Target Recognition, Advances in Sensing with Security Applications*. [NATO Security Through Science Book Series, Vol. 2]. Netherlands: Springer Netherlands, 2006. (ISBN: 978-1-4020-4284-3)
13. S. Hu, X. Wang, Q. Si, "Effects of FM Linearity of Linear FM Signals on Pulse-compression Performance," in *Proc. IEEE Conf. Radar*, Verona, NY, 2006.
14. T. Curtis, M. Curtis (2004). *High Performance Digital Signal Processing*, Curtis Technology (UK) Ltd. [Online]. Available: <http://www.curtistech.co.uk/>
15. D. Hoetzer, D. Freundt, "Driver Assist & Crash Avoidance Technologies: Radar and Video Systems", Robert Bosch GmbH, presented at Telematics Detroit, Detroit, MI, 2008.

16. S. Yamano *et al.*, "76GHz Millimeter Wave Automobile Radar using Single Chip MMIC", Fujitsu Ten Technical Journal No. 23, Fujitsu Ten Corp., Plymouth, MI, 2004.
17. A. Anderson. (November 2005). *Mercedes-Benz Sudden Acceleration Incidents* [Online]. Available: <http://www.antony-anderson.com/Cruise/9.5%20SA%20Links/Mercedes%20Benz.html>
18. R. Lachner, "Development Status of Next Generation Automotive Radar in EU – Infineon Radar Technology," presented at ITS Forum 2009, Tokyo (Japan), 2009.
19. J. Wenger, "RF-Applications in Vehicles – Today and Tomorrow – DaimlerChrysler," presented at DaimlerChrysler AG, Ulm (Germany), 2006.
20. TRW Automotive (2009). *Adaptive Cruise Control (ACC)* [Online]. Available: http://www.trw.com/sites/default/files/DAS_ACC_Eng09_0.pdf
21. Xilinx Corporation (2010). *ISE Design Suite: Intellectual Property*. [Online]. Available: <http://www.xilinx.com/ipcenter/>
22. J. M. Webster, "The Development of a Radar Digital Unit for the SASAR II Project," M.S. Thesis, Dept. Elect. Eng., Univ. Cape Town, Cape Town, South Africa, 2004.
23. M. Schneider, "Automotive Radar – Status and Trends," in *German Microwave Conf. (GeMic2005)*, Ulm (Germany), 2005, pp. 144-147.
24. J. Wenger, "Automotive Radar – Status and Perspectives," in *IEEE CSIC 2005 Dig.*, 2005, Sitges (Spain), 2005, pp. 21-24.
25. H.-L. Bloecher, A. Sailer, G. Rollmann, J. Dickmann, "79 GHz UWB Automotive Short Range Radar – Spectrum Allocation and Technology Trends," *Adv. Radio Sc.*, Vol. 7, 2009, pp. 61-65. Available: www.adv-radio-sci.net/7/61/2009/ars-7-61-2009.html
26. V. Jain, B. Javid, P. Heydari, "A 24/77GHz Dual-Band BiCMOS Frequency Synthesizer," in *IEEE 2008 Conf. Custom Integrated Circuits Conference (CICC)*, San Jose, CA, 2008, pp. 487-490.
27. V. Jain, B. Javid, P. Heydari, "A BiCMOS Dual-Band Millimeter-Wave Frequency Synthesizer for Automotive Radars," *IEEE J. Solid-State Circuits*, Vol. 44, No. 8, pp. 2100-2113, Aug. 2009.
28. J. Saad, A. Baghdadi, "FPGA-based Radar Signal Processing for Automotive Driver Assistance System," in *IEEE/IFIP Intl. Symp. Rapid System Prototyping*, Fairfax, VA, 2009, pp. 196-199.
29. D. E. Barrick, "FM/CW Radar Signals and Digital Processing," NOAA Technical Report ERL 283-WPL 26, U.S. Department of Commerce, Boulder (Colo.), Jul. 1973.
30. X. Bo, L. G.-lin, L. C.-hua, "DOA Estimation Based on Phase-difference," in *Proc. IEEE ICSP2006*, Beijing (China), 2006, pp. 1-4.

31. F. J. Harris, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *Proc. IEEE*, Vol. 66, No. 1, pp. 51-83, Jan. 1978.
32. J. W. Cooley, J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Math. Comput.*, Vol. 19, No. 90, pp. 297-301, 1965.
33. H. V. Sorensen, D. L. Jones, M. T. Heideman, C. Sidney Burrus, "Real-Valued Fast Fourier Transform Algorithms," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. ASSP-35, No. 6, pp. 849-863, Jun. 1987.
34. F. Gini, M. Greco, A. Farina, "Clairvoyant and Adaptive Signal Detection in Non-Gaussian Clutter : A Data-dependent Threshold Interpretation," *IEEE Trans. Signal Processing*, Vol. 47, No. 6, pp. 1522-1531, Jun. 1999.
35. L. Abdou, F. Soltani, "OS-CFAR and CMLD Threshold Optimization in Distributed Systems Using Evolutionary Strategies," *J. Signal, Image and Video Processing (SIVip)*, 2008, No. 2, pp. 155-167, Dec. 2008.
36. V. Kyovtorov *et al.*, "Parallel FPGA Design of CA CFAR Algorithm," Comp. Eng. Lab, TU-Delft, October 15, 2009.
37. R. S. Raghavan, "Analysis of CA-CFAR Processors for Linear-Law Detection," *IEEE Trans. Aerospace and Electronic Systems*, Vol. 28, No. 3, pp. 661-665, Jul. 1992.
38. T. R. Saed, J. K. Ali, Z. T. Yassen, "An FPGA Based Implementation of CA-CFAR Processor," *Asian Journal of Information Technology*, Vol. 6, No. 4, pp. 511-514, 2007.
39. R. Cumplido, C. Torres, S. López, "On the Implementation of an Efficient FPGA-based CFAR Processor for Target Detection," in *Proc. IEEE ICEEE and CEI, Acapulco (Mexico)*, 2004, pp. 214-218.
40. G. Galati (Ed.), *Advanced Radar Techniques and Systems*, IEEE Radar, Sonar, Navigation and Avionics Series 4, Exeter (UK): Short Run Press Ltd, 1993.
41. L. L. Scharf, *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*, New York: Addison Wesley, 1991. (ISBN: 0-201-19038-9).
42. M. I. Skolnik, *Introduction to Radar Systems*, 3rd ed., New York: McGraw-Hill, 2001.
43. P. Swerling, "Probability of Detection for Fluctuating Targets," The RAND Corp., Santa Monica, CA, Mar. 17, 1954.
44. P. Bak, C. Tang, K. Wiesenfeld, "Self-Organized Criticality: An Explanation of 1/f Noise," *Physical Review Letters*, Vol. 59, pp. 381-384, 1987.
45. R. Lyons, R. Yates. (2005, June). *Reducing ADC Quantization Noise* [Online]. Available: <http://www.mwrf.com/Articles/Index.cfm?Ad=1&Ad=1&ArticleID=10586>
46. P. W. Gorham, "RF Atmosphere Absorption/Ducting," Antarctic Impulsive Transient Antenna Project (ANITA), Univ. Hawaii (Manoa), April 21, 2003.

47. C. Lundquist, U. Orguner, T. B. Schön, "Tracking Stationary Extended Objects for Road Mapping using Radar Measurements," *IEEE Symp. Intelligent Vehicles IV'09*, Xi'an (China), 2009.
48. BBC h2g2. (2003, July). *Radar Countermeasures: Range Gate Pull-off* [Online]. Available: <http://www.bbc.co.uk/dna/h2g2/A637535>
49. D. Kok, J. S. Fu, "Signal Processing for Automotive Radar," in *IEEE Radar Conf.(EURAD2005)*, Arlington, VA, June 2005, pp. 842-846.
50. K. Estmer, "Noise Model of an FMCW Radar System," M.S. Thesis, Dept. Elect. Eng., Royal Inst. of Tech., Stockholm (Sweden), May 2002.
51. Xilinx Corporation (2010). *Virtex-5 Family/Virtex-5/ML506 Development Board Datasheets* [Online]. Available: <http://www.xilinx.com/support/documentation/ml506.htm>
52. Xilinx Corporation (2010). *Xilinx FFT v7.0 Datasheet* [Online]. Available: http://www.xilinx.com/support/documentation/ip_documentation/xfft_ds260.pdf
53. J. Kumar, B. Mazhari, "Spectral Estimation Using a MUSIC Algorithm - Altera Nios II Embedded Processor Design Contest," Altera Corporation, San Jose, CA, 2005. Available: http://www.altera.com/literature/dc/1.6-2005_India_3rd_IITKanpur-web.pdf
54. R. Roy, T. Kailath, "ESPRIT – Estimation of Signal Parameters Via Rotational Invariance Technique", *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 37, No. 7, pp. 984-995, Jul. 1989.
55. M. Huang, *Vehicle Crash Dynamics*, Michigan: CRC Press, 2002. (ISBN: 0-8493-0104-1)
56. G. Hampson, "Implementation Results of a Windowed FFT", Sys. Eng. Div., Ohio State Univ., Columbus, OH, July 12, 2002. Available: <http://esl.eng.ohio-state.edu/~rsttheory/iip/window.pdf>

APPENDIX

Contents:

1. MATLAB listing for Radar Echo Signal Generation and Radar Signal Processing Algorithm testing
2. MATLAB listing for percentage error calculation from 10-bit rounding of Window functions
3. HDL listing for TLC
4. HDL listing for SAMPLER
5. HDL wrapper for Xilinx FFT v7.0 core
6. HDL listing for FDR
7. HDL listing for PSD
8. HDL listing for CFAR
9. HDL listing for PPM

A1. MATLAB listing for Radar Echo Signal Generation and Radar Signal Processing Algorithm testing

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                                ABOUT THIS CODE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% The code generates a set of intermediate frequencies for a long range
% radar, for both the up and down sweeps. The cell-averaging cfar algorithm
% is then employed, followed by removal of spectral copies, and a final
% loop to remove any left-over noise components from the target map. This
% leaves a final cfar matrix with all valid targets, which are plotted.
% -----
% The target echo power is attenuated by 0.4dB/km as the factor of
% attenuation of RF radiation in clear air. This factor can be changed
% once a more appropriate/practical value is obtained.
%
% The algorithm eliminates any targets which are within +-1 frequency bin
% of another target. This puts an upper limit to the number of targets the
% system can detect:
% Maximum number of simultaneously detectable targets = (NFFT/2)/3
% where NFFT is the length of the Fourier transform.
% Due to leakage and noise effects, this number can be practically as low
% as (NFFT/2)/5. The noise and leakage effects persist to an extent despite
% windowing.
% -----
% The original ca-cfar algorithm has poorer performance with higher number
% of targets. To overcome this problem, a duplicate or ghost target removal
% scheme is employed, followed by a secondary threshold. This enables
% operation at a deteriorated probability of false alarm. Originally using
% Pfa = 10^-9, and finally using Pfa = 10^-6. This allows multiple targets
% to be detected with a resolution of 2.7 metres at same velocities.
%
% The Pfa can be lowered further, which results in more false targets but
% at low power. These can be removed by using a tertiary threshold scheme.
%
% Increasing the sweep bandwidth from 200MHz to 500MHz, and sampling rate
% from 1MSps to 3MSps can help improve the resolution to a certain extent,
% such that the range resolution drops to 1 metre.
% -----
% The FMCW LRR simulated here can only detect the maximum relative velocity
% of 300KMPH reliably at a minimum distance of 10 meters.
% -----
% Windowing is NOT included in this code.
%
% Finally, the code uses the frequency information from the up sweep and
% the down sweep to compute the range and velocity of each detected target.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                DEVELOPER: SUNDEEP LAL (MEMS LAB)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
clc

Tsweep = 1.024*10^-3; % Chirp duration in seconds
FswEEP = 800 * 10^6; % Chirp bandwidth in Hz
% Largely affects the range resolution of the system
% A larger sweep bandwidth increases the spectral

```

```

% gap between targets, giving better cfar detection.
c = 2.973 * 10^8; % Speed of EM waves in m/s
Ft = 76.9 * 10^9; % Central transmission frequency

% Frequency sweep rate in s^-2
k = Fsweep/Tsweep;

% Target ranges in m
rangesUp = [9 24 29 55 78 106 148]; % hypothetical scenario
%rangesUp = [12 54]; % practical road scenario beam1
%rangesUp = [111 90]; % practical road scenario beam2
%rangesUp = [35 78 90]; % practical road scenario beam3

% Target velocities in km/h
% (all targets assumed to travel in same direction)
% (all targets assumed to have zero acceleration during frequency chirp)
velocities = [123 55 89 100 70 80 22]; % hypothetical scenario
%velocities = [65 24]; % practical road scenario beam1
%velocities = [90 150]; % practical road scenario beam2
%velocities = [250 99 150]; % practical road scenario beam3

% Host vehicle velocity in km/h
velocity = 100;

% Target echo received power factors assuming worst case scenario of
% 0.8dB/km attenuation in light rain
for i=1:length(rangesUp)
    loss = -2*0.8*rangesUp(i)/1000; % Two-way atmospheric absorption loss (dB)
                                % of 77GHz
    atten(i) = 10^loss; % Attenuation factor
end

% Relative velocities in m/s
for i=1:length(rangesUp)
    relativeVelocity(i) = (velocity - velocities(i))/3.6;
end

% Change in ranges after up sweep
for i=1:length(rangesUp)
    rangesDown(i) = rangesUp(i) + relativeVelocity(i)*Tsweep;
end

% Up and Down sweep frequencies in Hz
for i=1:length(rangesUp)
    upIF(i) = k*2*rangesUp(i)/c + 2*Ft*relativeVelocity(i)/c;
    downIF(i) = k*2*rangesDown(i)/c - 2*Ft*relativeVelocity(i)/c;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Up chirp IF %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Fs = 2*10^6; % Sampling frequency
T = 1/Fs; % Sample time
L = Fs * Tsweep; % Length of signal
t = (0:L-1)*T; % Time vector for up chirp

xUp = 0;
% Sum of all target frequencies in the up chirp
for i=1:length(rangesUp)
    xUp = xUp + atten(i)*sin(2*pi*upIF(i)*t);
end

```

```

yUp = xUp + randn(size(t)); % Sinusoids plus system noise

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Down chirp IF %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xDown = 0;
% Sum of all target frequencies in the down chirp
for i=1:length(rangesUp)
    xDown = xDown + atten(i)*sin(2*pi*downIF(i)*t);
end
yDown = xDown + randn(size(t)); % Sinusoids plus system noise

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Make hex time-domain data for HDL simulation %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
uu = yUp;
uu = uu - min(uu);
uu = uu./max(uu);
uu = uu.*2047;
uu = round(uu);
uuhex = dec2hex(uu);
dd = yDown;
dd = dd - min(dd);
dd = dd./max(dd);
dd = dd.*2047;
dd = round(dd);
ddhex = dec2hex(dd);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Apply Window to time-domain samples %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
window = hamming(2048);
for i=1:2048
    yUp(i) = yUp(i) * window(i);
    yDown(i) = yDown(i) * window(i);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot time-domain received IF %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(1)
subplot(2,1,1)
plot(t(1:L),yUp(1:L))
title('Up Chirp IF corrupted with Zero-Mean Random Noise')
xlabel('Time (ms)')
ylabel('Amplitude')
figure(2)
subplot(2,1,1)
plot(t(1:L),yDown(1:L))
title('Down Chirp IF corrupted with Zero-Mean Random Noise')
xlabel('Time (ms)')
ylabel('Amplitude')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot frequency-domain received IF %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
NFFT = 2^nextpow2(L); % Next power of 2 from length of y
Yup = fft(yUp,NFFT)/L;
Ydown = fft(yDown,NFFT)/L;
f = Fs/2* linspace(0,1,NFFT/2+1);

```



```

% Plot single-sided amplitude spectrum
figure(3)
%subplot(2,1,2)
stem(f,abs(Yup(1:NFFT/2+1))./max(abs(Yup(1:NFFT/2+1))))
title('Single-Sided Amplitude Spectrum of yUp(t)')
xlabel('Frequency (Hz)')
ylabel('|Yup(f)|')
figure(4)
%subplot(2,1,2)
stem(f,abs(Ydown(1:NFFT/2+1))./max(abs(Ydown(1:NFFT/2+1))))
title('Single-Sided Amplitude Spectrum of yDown(t)')
xlabel('Frequency (Hz)')
ylabel('|Ydown(f)|')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CA-CFAR detection
%  $K = Pfa^{(-1/(2*M))} - 1$ 
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% |-----|
% | UP SWEEP |
% |-----|
Pfa = 10^-6;
M = 4; % Depth of cell averaging on one side of CUT
GB = 2; % Number of guard bands around Cell-Under-Test
K = Pfa^(-1/(2*M)) - 1; % Cell averaging factor
tmpcfar = [0 0 0 0]'; % Initiate the cfar matrix
countup = 1;
countupfinal = 0;

for CUT=2:NFFT/2 % Start from index 2 to avoid DC component caused by
                % system and channel noise. Stop at (NFFT/2-30) to
                % limit maximum target range, relative velocity at
                % 150m,300kmph
    avgL = 0; % Average on left side of Cell-Under-Test
    avgR = 0; % Average on right side of Cell-Under-Test

    % Compute the averages
    if(CUT<=M+GB)
        for i=1:M
            avgR = avgR + abs(Yup(CUT+i+GB));
        end
        avgR = avgR/M;
    elseif(CUT>=NFFT/2-M-GB)
        for i=1:M
            avgL = avgL + abs(Yup(CUT-i-GB));
        end
        avgL = avgL/M;
    else
        for i=1:M
            avgL = avgL + abs(Yup(CUT-i-GB));
            avgR = avgR + abs(Yup(CUT+i+GB));
        end
        avgR = avgR/M;
        avgL = avgL/M;
    end
end

```

```

% Compute threshold
T = (avgR+avgL)/2 * K;
% Decision
if(abs(Yup(CUT))>T)
    countup = countup + 1;
    tmpcfar(1,countup) = abs(Yup(CUT));
    tmpcfar(2,countup) = CUT;
end
end
tmpcfar(1,countup+1) = 0;
tmpcfar(2,countup+1) = 0;

% REMOVE ALL SPECTRAL COPIES HERE
j = 1;
for i=2:length(tmpcfar(1,:))-1
    if((tmpcfar(2,i)~=tmpcfar(2,i+1)-1)&&(tmpcfar(2,i)~=tmpcfar(2,i+1)))
        if((tmpcfar(2,i)==tmpcfar(2,i-1)+1)|| (tmpcfar(2,i)==tmpcfar(2,i-1)))
            tmpplcfar(1,j) = max(tmpcfar(1,i-1),tmpcfar(1,i));
            tmpplcfar(2,j) = tmpcfar(2,i);
            j = j + 1;
        else
            tmpplcfar(1,j) = tmpcfar(1,i);
            tmpplcfar(2,j) = tmpcfar(2,i);
            j = j + 1;
        end
    end
end
end

% Eliminate any residual false alarms
ST = 0.6 * mean(tmpplcfar(1,:)); % Secondary Threshold computed from
                                % mean of all detected target powers
j = 1;
for i=1:length(tmpplcfar(1,:))
    if(tmpplcfar(1,i)>ST)
        cfar(1,j) = tmpplcfar(1,i);
        cfar(2,j) = tmpplcfar(2,i); % * Fs/NFFT;
        j = j + 1;
        countupfinal = countupfinal + 1;
    end
end

% Plot detected targets
figure(5)
subplot(2,1,1)
title('CFAR-detected targets for yUp(t)')
xlabel('Frequency (Hz)')
ylabel('|Yup(f)|')
stem(cfar(2,:),cfar(1,:));

% |-----|
% | DOWN SWEEP |
% |-----|
countdown = 1;
countdownfinal = 0;

for CUT=2:NFFT/2 % Start from index 2 to avoid DC component caused by
                  % system and channel noise. Stop at (NFFT/2-30) to
                  % limit maximum target range, relative velocity at
                  % 150m,300kmph
    avgL = 0; % Average on left side of Cell-Under-Test
    avgR = 0; % Average on right side of Cell-Under-Test

```

```

% Compute the averages
if(CUT<=M+GB)
    for i=1:M
        avgR = avgR + abs(Ydown(CUT+i+GB));
    end
    avgR = avgR/M;
elseif(CUT>=NFFT/2-M-GB)
    for i=1:M
        avgL = avgL + abs(Ydown(CUT-i-GB));
    end
    avgL = avgL/M;
else
    for i=1:M
        avgL = avgL + abs(Ydown(CUT-i-GB));
        avgR = avgR + abs(Ydown(CUT+i+GB));
    end
    avgR = avgR/M;
    avgL = avgL/M;
end

% Compute threshold
T = (avgR+avgL)/2 * K;
% Decision
if(abs(Ydown(CUT))>T)
    countdown = countdown + 1;
    tmpcfar(3,countdown) = abs(Ydown(CUT));
    tmpcfar(4,countdown) = CUT;
end

end
tmpcfar(3,countdown+1) = 0;
tmpcfar(4,countdown+1) = 0;

% REMOVE ALL SPECTRAL COPIES HERE
j = 1;
for i=2:length(tmpcfar(1,:))-1
    if((tmpcfar(4,i)~=tmpcfar(4,i+1)-1)&&(tmpcfar(4,i)~=tmpcfar(4,i+1)))
        if((tmpcfar(4,i)==tmpcfar(4,i-1)+1)|| (tmpcfar(4,i)==tmpcfar(4,i-1)))
            tmpplcfar(3,j) = max(tmpcfar(3,i-1),tmpcfar(3,i));
            tmpplcfar(4,j) = tmpcfar(4,i);
            j = j + 1;
        else
            tmpplcfar(3,j) = tmpcfar(3,i);
            tmpplcfar(4,j) = tmpcfar(4,i);
            j = j + 1;
        end
    end
end

end

% Eliminate any residual false alarms
ST = 0.6 * mean(tmpplcfar(3,:)); % Secondary Threshold computed from
                                % mean of all detected target powers
j = 1;
for i=1:length(tmpplcfar(3,:))
    if(tmpplcfar(3,i)>ST)
        cfar(3,j) = tmpplcfar(3,i);
        cfar(4,j) = tmpplcfar(4,i); % * Fs/NFFT;
        j = j + 1;
        countdownfinal = countdownfinal + 1;
    end
end
end

```

```

% Plot detected targets
figure(6);
%subplot(2,1,2)
title('CFAR-detected targets for yDown(t)')
xlabel('Frequency (Hz)')
ylabel('|Ydown(f)|')
stem(cfar(4,:),cfar(3,:));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot target IF phase %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:length(cfar(1,:))
    phaseup(i) = 180 * atan( imag( Yup(cfar(2,i)) ) / real( Yup(cfar(2,i)) ) );
    phasedown(i) = 180 * atan( imag( Ydown(cfar(4,i)) ) / real(
Ydown(cfar(4,i)) ) );
end

i=1:length(cfar(1,:));
figure(7);
subplot(2,1,1);
stem(i,(phaseup));
subplot(2,1,2);
stem(i,(phasedown));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%
% Pairing %
%%%%%%%%%%%%

% PAIRING IS DONE BASED ON TWO STAGES:
% 1) the up sweep and down sweep intermediate frequencies of the same target
% will be within 22 cells of each other.
% 2) if there are multiple down sweep intermediate frequencies that fall in
% the criteria in (1) for a given frequency in the up sweep, then peak power
% comparison is done.

```

A2. MATLAB listing for percentage error calculation from 10-bit rounding of Window functions

```
clear all
clc

window = hamming(2048); % get the coefficients of 2048-point Hamming
                        window
window = window.*1023; % scale the coefficients to 10-bit range
rounded_window = round(window); % round off window coefficients to
                                nearest integer

% compute the percentage error from rounding
for i=1:2048
    perr(i) = abs( window(i)-rounded_window(i) )/window(i) * 100;
end

mean(perr) % display the average percentage error from rounding
```

A3. HDL listing for TLC

```
`timescale 1ns / 1ps

/////////////////////////////////////////////////////////////////
// This is the full radar system, including the controller and the digital signal
// processing modules. The input are 2048 11-bit time domain samples from the ADC,
// the outputs are a modulation signal for the VCO tuning voltage via DAC, and the
// detected target information.
//
// - SUNDEEP LAL -
/////////////////////////////////////////////////////////////////

module toplevel(
    clk,
        reset,
        en,
        datain,
        unit_vel,
        sclk,
        final_target_info,
        beamport,
        final_info_valid,
        modulate );

// Inputs
input clk, reset, en;
input [10:0] datain; // ADC -> ADC_CAPTURE
input [7:0] unit_vel; // vehicle velocity

// Outputs
output sclk; // SCLK is sampling clock to ADC
output [21:0] final_target_info; // 9.1bits velocity, 8.2bits range, 2bits beamport
output [2:0] beamport; // 100 - beamport1, 010 beamport2, 001 - beamport3
output final_info_valid; // data valid output from PAIRING module
output [9:0] modulate; // 10-bit DAC output for tuning voltage

// Output registers
reg [21:0] final_target_info;
reg [2:0] beamport;
reg final_info_valid;
reg [9:0] modulate;

// Internal registers
reg updown;
reg modclock; // modulation clock at 1MHz to update DAC value
reg [5:0] modtimer; // counter for clock division: 100MHz -> 1MHz
reg moddone; // flag to mark update of VCO tuning voltage
reg dirchange; // flag to mark change of sweep direction
(* KEEP = "TRUE"*) reg [18:0] velmulres; // used in adjusting target velocity for beamports 1/3 angle
reg [8:0] velmulfac; // multiplication factor 1/cos(10) for target at +-10 degrees
reg st; // internal flag

// Internal connections
wire hold; // ADC_CAPTURE -> TOPLEVEL (busy signal, do not change sweep direction)
```

```

wire fwd_inv_we, fwd_inv;
    assign fwd_inv_we = 1;
    assign fwd_inv = 1; // forward FFT
wire scale_sch_we;
    assign scale_sch_we = 0;
wire [11:0] scale_sch;
    assign scale_sch = 12'd0; // use default scaling schedule for FFT stages
wire fft_start; // ADC_CAPTURE -> FFT
wire [11:0] xn_re, xn_im; // ADC_CAPTURE -> FFT
wire [10:0] xn_index; // FFT -> ADC_CAPTURE
wire [10:0] xk_index; // FFT -> UNLOAD_FFT
wire [11:0] xk_re, xk_im; // FFT -> UNLOAD_FFT
wire fft_busy, fft_edone; // FFT -> ?toplevel?
wire fft_rfd, fft_dv, fft_done; // FFT -> UNLOAD_FFT
wire [9:0] index; // UNLOAD_FFT -> FFT_CAPTURE
wire [11:0] re, im; // UNLOAD_FFT -> FFT_CAPTURE
wire dv; // UNLOAD_FFT -> FFT_CAPTURE
wire sqrt_done; // TOPLEVEL -> FFT_CAPTURE
wire donea, doneb, donec, doned; // SQRT -> TOPLEVEL
    assign sqrt_done = donea & doneb & donec & doned;
wire [24:0] sqrt_feeda, sqrt_feedb, sqrt_feedc, sqrt_feedd; // FFT_CAPTURE -> SQRT
wire sqrt_start; // FFT_CAPTURE -> SQRT
wire [12:0] roota, rootb, rootc, rootd; // SQRT -> CFAR
wire [12:0] target_abs; // CFAR -> PAIRING
wire [9:0] target_pos; // CFAR -> PAIRING
wire new_target, complete; // CFAR -> PAIRING
wire start_cfar; // CFAR -> FFT_CAPTURE
wire [19:0] target_info; // PAIRING -> TOPLEVEL
wire info_valid; // PAIRING -> TOPLEVEL

```

```

//-----
// 1Mhz clock generator from 100MHz system clock
// - used to update VCO tuning voltage via DAC
//-----
always @ ( posedge clk )
begin
    if( reset == 1 ) // synchronous reset
    begin
        modtimer <= 6'd0;
        modclock <= 1'b0;
    end

    else if( modtimer == 49 )
    begin
        modclock <= ~modclock; // invert modulation clock
        modtimer <= 6'd0; // clear counter
    end

    else
        modtimer <= modtimer + 1;
end

```

```

//-----
// Beamport, tuning voltage and sweep control
//-----

```

```

always @ ( posedge clk )
begin
    if( reset == 1 ) // synchronous reset
    begin
        beamport <= 3'b100; // start with beamport1
        modulate <= 10'd0;
        updown <= 1'b1; // start in up sweep
        moddone <= 1'b0;
        dirchange <= 1'b0;
    end

    // if FFT computation has begun
    else if( fft_start == 1 && dirchange == 0 )
    begin
        dirchange <= 1'b1; // flag: mark change of sweep direction

        if( updown == 1 ) // if current sweep direction is up
        begin
            modulate <= modulate; // set up modulation signal for down sweep
            updown <= 1'b0; // switch to down sweep
        end

        else // if current sweep direction is down
        begin
            modulate <= 10'd0; // set up modulation signal for up sweep
            updown <= 1'b1; // switch to up sweep

            if( beamport == 3'b100 ) // if currently using beamport1
                beamport <= 3'b010; // switch to beamport2
            else if( beamport == 3'b010 ) // if currently using beamport2
                beamport <= 3'b001; // switch to beamport3
            else // if currently using beamport3
                beamport <= 3'b100; // switch back to beamport1
            end
        end
    end

    else if( hold == 1 )
    begin
        modulate <= modulate; // hold at max. while ADC_CAPTURE is busy feeding FFT
        dirchange <= 1'b0; // clear flag
    end

    else if( modclock == 1 && moddone == 0 )
    begin
        if( updown == 1 ) // up sweep
        begin
            if( modulate < 1023 )
                modulate <= modulate + 1; // increase tuning voltage
            else
                modulate <= modulate; // hold at max.
        end

        else if( updown == 0 )
        begin
            if( modulate > 0 )
                modulate <= modulate - 1; // decrease tuning voltage
            else

```



```

        modulate <= modulate; // hold at min.

    end

    moddone <= 1'b1; // flag: tuning voltage has been updated

end

else if( modclock == 0 )
    moddone <= 1'b0; // clear flag: ready for next 'modclock' pulse to update..
    //..tuning voltage via 'modulate'
end

//-----
// Adjust target velocity according to beam angle w.r.t. vehicle
//-----
always @ ( posedge clk )
begin
    if( reset == 1 )
    begin
        final_target_info <= 22'd0;
        final_info_valid <= 1'b0;
        velmulres <= 19'd0;
        velmulfac <= 9'b100000001; /* - 1.8bits number, decimal equivalent is 1.00390625
        - multiplication factor for target at 5 degrees
        angle to the vehicle is 1/cos(5) = 1.00382 */

        st <= 1'b0;

    end

    else
    begin
        if( info_valid == 1 || st == 1 )
        begin
            if( st == 0 )
            begin
                // if current beamport1/2, then previous beamport is 3/1
                // i.e. target is at an angle of +-10 degrees beam
                if( beamport == 3'b100 || beamport == 3'b010 )
                begin
                    velmulres <= target_info[19:10] * velmulfac;
                    // extract range (unaffected by angle), append beamport#
                    if( beamport == 3'b100 )
                        final_target_info[11:0] <= {target_info[9:0],2'd3};
                    else if( beamport == 3'b010 )
                        final_target_info[11:0] <= {target_info[9:0],2'd1};
                    st <= 1'b1; // mark flag to add adjusted velocity
                end
                // else if previous beamport is 2, target velocity remains unchanged
                else
                begin
                    final_target_info[21:0] <= {target_info,2'd2}; // append beamport#
                    final_info_valid <= 1'b1;
                end
            end

            end

        else if( st == 1 && velmulres[17:9] <= 300 ) // add adjusted velocity to output..
        begin
            //..and allow max. 300kmph

```

```

        final_target_info[21:12] <= velmulres[17:8];
        final_info_valid <= 1'b1;
        st <= 1'b0; // clear flag
    end

    end

end

if( final_info_valid == 1 )
begin
    final_info_valid <= 1'b0; // clear flag
    final_target_info <= 22'd0;
end

end

//-----
// Module instantiation
//-----

adc_capture adc_capture_1(
    .clk(clk),
    .reset(reset),
    .en(en),
    .fft_rfd(fft_rfd),
    .datain(datain),
    .xn_index(xn_index),
    .xn_re(xn_re),
    .xn_im(xn_im),
    .fft_start(fft_start),
    .hold(hold),
    .sclk(sclk) );

fft_2048 fft_2048_1(
    .fwd_inv_we_i(fwd_inv_we),
    .rfd_i(fft_rfd),
    .start_i(fft_start),
    .fwd_inv_i(fwd_inv),
    .dv_i(fft_dv),
    .unload_i(fft_unload),
    .scale_sch_we_i(scale_sch_we),
    .done_i(fft_done),
    .clk_i(clk),
    .busy_i(fft_busy),
    .edone_i(fft_edone),
    .scale_sch_i(scale_sch),
    .xn_re_i(xn_re),
    .xk_im_i(xk_im),
    .xn_index_i(xn_index),
    .xk_re_i(xk_re),
    .xn_im_i(xn_im),
    .xk_index_i(xk_index) );

unload_fft unload_fft_1(
    .clk(clk), // global clock
    .reset(reset), // global synchronous reset

```

```

.fft_done(fft_done), // completion signal from FFT core
.fft_dv(fft_dv), // data valid signal from FFT core
.xk_index(xk_index), // data index from FFT core
.xk_re(xk_re), // real output from FFT
.xk_im(xk_im), // imaginary output from FFT
.fft_unload(fft_unload), // unload transform results from FFT core
.index(index), // 1023 -> 0 index to FFT_CAPTURE
.re(re), // real output to FFT_CAPTURE
.im(im), // imaginary output to FFT_CAPTURE
.dv(dv) ); // data valid signal to FFT_CAPTURE

fft_capture fft_capture_1(
.clk(clk),
.reset(reset),
.index(index), // sample index from 1023 down to 0 from UNLOAD_FFT
.re(re), // real FFT output data from UNLOAD_FFT
.im(im), // imaginary data from UNLOAD_FFT
.dv(dv), // data valid signal from UNLOAD_FFT
.cfar_busy(start_cfar), // busy signal from CFAR unit, halt feeding SQRT while high
.sqrt_done(sqrt_done), // completion signal from SQRT units
.sqrt_feeda(sqrt_feeda), // output to SQRT units
.sqrt_feedb(sqrt_feedb),
.sqrt_feedc(sqrt_feedc),
.sqrt_feedd(sqrt_feedd),
.sqrt_start(sqrt_start) ); // start signal to all SQRT units

sqrt sqrt1(
.clk(clk),
.reset(reset),
.value(sqrt_feeda), // 25-bit input sum of real^2 + imag^2
.start(sqrt_start), // start signal from FFT_CAPTURE
.root(roota), // square root of input
.done(donea) ); // completion signal

sqrt sqrt2(
.clk(clk),
.reset(reset),
.value(sqrt_feedb),
.start(sqrt_start),
.root(rootb),
.done(doneb) );

sqrt sqrt3(
.clk(clk),
.reset(reset),
.value(sqrt_feedc),
.start(sqrt_start),
.root(rootc),
.done(donec) );

sqrt sqrt4(
.clk(clk),
.reset(reset),
.value(sqrt_feedd),
.start(sqrt_start),

```

```

        .root(rootd),
        .done(doned) );

cacfar_32 cacfar_32_1(
    .clk(clk),
        .reset(reset),
        .inA(roota), // inA,inB, inC, inD are obtained from 4 different sqrt modules
        .inB(rootb),
        .inC(rootc),
        .inD(rootd),
        .start(sqrt_done), // start receiving values from Sqrt modules..
        .target_abs(target_abs), // new target peak intensity
        .target_pos(target_pos), // new target frequency bin number
        .new_target(new_target), // new target detected signal
        .start_cfar(start_cfar), // high when busy, mapped to cfar_busy in FFT_CAPTURE
        .complete(complete) ); // completion of CFAR processing for current data batch

pairing pairing_1(
    .clk(clk),
        .reset(reset),
        .new_target(new_target), // new target detected signal from CACFAR_32 module
        .target_abs(target_abs), // new target peak intensity
        .target_pos(target_pos), // new target frequency bin number
        .complete(complete), // CFAR completion signal
        .updown(updown), // updown = 1(0) during up(down) sweep sampling i.e. down(up) sweep processing
        .unit_vel(unit_vel), // vehicle velocity
        .target_info(target_info), // MSB -> 10 bits velocity, 10 bits range <- LSB
        .info_valid(info_valid) ); // target information valid signal to display unit

endmodule

```

A4. HDL listing for SAMPLER

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////
// This module is responsible for capturing data from the ADC, buffering it, and
// transferring it to the FFT module for frequency analysis. There is a clock
// divider that divides the system clock down to sampling clock.
//
// - SUNDEEP LAL -
////////////////////////////////////////////////////////////////

module adc_capture(
    clk,
        reset,
        en,
        fft_rfd,
        datain,
        xn_index,
        xn_re,
        xn_im,
        fft_start,
        hold,
        sclk );

// Inputs
input clk; // global clock
input reset; // global reset
input en; // enable
input fft_rfd; // FFT core ready-for-data signal
input [10:0] datain; // input sample from ADC
input [10:0] xn_index; // 2048 samples

// Outputs
output [11:0] xn_re; // real part of sample data to FFT core
output [11:0] xn_im; // imaginary part of sample data to FFT core
output fft_start; // start FFT calculation
output hold; // hold while passing data to FFT core
output sclk; // sampling clock at 2MHz to drive ADC

// Internal registers
reg [11:0] xn_re;
reg [11:0] xn_im;
reg fft_start;
reg hold;
reg sclk;

reg [4:0] sclk_cnt; // clock divider counter
reg sample_read; // internal flag
reg [10:0] data_buf [2047:0];
reg [10:0] data_cnt;
reg feedfft; // internal flag
reg feeddone; // internal flag
(* KEEP = "TRUE" *) reg [21:0] mult_res; // window multiplication result register
```

```

reg [9:0] window [1023:0]; // window function

//-----
// generate sampling clock @ 2MHz from 100MHz supply
//-----
always @ ( posedge clk )
begin

    if( reset == 1 )
    begin
        sclk <= 1'b0;
        sclk_cnt <= 5'd0;
    end

    else if( reset == 0 && en == 1 && hold == 1 )
    begin
        sclk <= 1'b0;
        sclk_cnt <= 5'd0;
    end

    else if( reset == 0 && en == 1 && hold == 0 )
    begin
        if( sclk_cnt == 24 ) // count 24 -> 100MHz, 19 -> 80MHz, 14 -> 60MHz
        begin
            sclk <= ~sclk;
            sclk_cnt <= 5'd0;
        end

        else
        begin
            sclk_cnt <= sclk_cnt + 1;
        end
    end
end

end

//-----
// Capture data from adc
//-----
always @ ( posedge clk )
begin

    if( reset == 1 )
    begin
        data_cnt <= 11'd0;
        feedfft <= 1'b0;
        sample_read <= 1'b0;
    end

    else if( reset == 0 && en == 1 && sclk == 1 && feedfft == 0 )
    begin
        if( sample_read == 0 )
        begin
            data_buf [data_cnt] <= datain; // store data in buffer
            data_cnt <= data_cnt + 1; // increment buffer index
            sample_read <= 1'b1;
        end
    end
end

```

```

        end

        if( data_cnt == 2047 && sample_read == 0 )
        begin
            feedfft <= 1'b1; // hold is asserted after 2 clk cycles
        end

    end

    else if( reset == 0 && sclk == 0 )
        sample_read <= 1'b0;

    if( reset == 0 && feaddone == 1 )
        feedfft <= 1'b0; // clear flag
    end

end

//-----
// Send captured data to FFT core
//-----
always @ ( posedge clk )
begin

    if( reset == 1 )
    begin
        xn_re <= 12'd0;
        xn_im <= 12'd0;
        hold <= 1'b0;
        fft_start <= 1'b0;
        mult_res <= 22'd0;
        feaddone <= 1'b0;
    end

    else if( reset == 0 && feaddone == 1 )
        feaddone <= 1'b0; // clear flag

    else if( reset == 0 && en == 1 && feedfft == 1 )
    begin
        if( fft_start == 0 && hold == 0 )
        begin
            //$display("FFT feed start: %d", $time);
            fft_start <= 1'b1; // start FFT core
            hold <= 1'b1; // halt sampling while passing data to FFT
            mult_res <= {1'b0, data_buf [xn_index]} * window [xn_index];
        end

        else if( fft_start == 1 && xn_index == 0 && fft_rfd == 1 )
        begin
            fft_start <= 1'b0;
            xn_re <= mult_res [21:10]; // truncate and send
            mult_res <= {1'b0, data_buf [xn_index + 1]} * window [xn_index + 1];
        end

        else if( xn_index > 0 && xn_index < 1023 && fft_rfd == 1 )
        begin
            xn_re <= mult_res [21:10]; // truncate and send
            mult_res <= {1'b0, data_buf [xn_index + 1]} * window [xn_index + 1];
        end
    end
end

```

```

        end

        else if( xn_index > 1022 && xn_index < 2047 && fft_rfd == 1 )
        begin
            xn_re <= mult_res [21:10]; // truncate and send
            mult_res <= {1'b0,data_buf [xn_index + 1]} * window [2047 - xn_index - 1];
        end

        else if( xn_index == 2047 && fft_rfd == 1 )
        begin
            xn_re <= mult_res [21:10]; // truncate and send
            hold <= 1'b0; // resume sampling next sweep
            feeddone <= 1'b1; // feedfft is deasserted after 2 clk cycles
        end
    end

end

//-----
// set window function coefficients
//-----
always @ ( posedge clk )
begin
    if( reset == 1)
    begin
        window[0] <= 0;
        window[1] <= 1;
        window[2] <= 2;
        ..
        .. // define window coefficients here
        ..
        window[2047] <= 1023;
    end
end

end
endmodule

```


A5. HDL wrapper for Xilinx FFT v7.0 core

```
`timescale 1ns / 1ps

module fft_2048 (
    fwd_inv_we_i, rfd_i, start_i, fwd_inv_i, dv_i, unload_i, scale_sch_we_i, done_i, clk_i, busy_i, edone_i, scale_sch_i,
    xn_re_i, xk_im_i, xn_index_i, xk_re_i, xn_im_i, xk_index_i
);
    input fwd_inv_we_i;
    output rfd_i;
    input start_i;
    input fwd_inv_i;
    output dv_i;
    input unload_i;
    input scale_sch_we_i;
    output done_i;
    input clk_i;
    output busy_i;
    output edone_i;
    input [11 : 0] scale_sch_i;
    input [11 : 0] xn_re_i;
    output [11 : 0] xk_im_i;
    output [10 : 0] xn_index_i;
    output [11 : 0] xk_re_i;
    input [11 : 0] xn_im_i;
    output [10 : 0] xk_index_i;

    xfft_v6_0 fft (
        .fwd_inv_we(fwd_inv_we_i),
        .rfd(rfd_i),
        .start(start_i),
        .fwd_inv(fwd_inv_i),
        .dv(dv_i),
        .unload(unload_i),
        .scale_sch_we(scale_sch_we_i),
        .done(done_i),
        .clk(clk_i),
        .busy(busy_i),
        .edone(edone_i),
        .scale_sch(scale_sch_i),
        .xn_re(xn_re_i),
        .xk_im(xk_im_i),
        .xn_index(xn_index_i),
        .xk_re(xk_re_i),
        .xn_im(xn_im_i),
        .xk_index(xk_index_i)
    );
endmodule
```

A6. HDL listing for FDR

[Unit to unload data from FFT]

```
`timescale 1ns / 1ps

/////////////////////////////////////////////////////////////////
// This module is needed to compute the 2's complement of the FFT output in order
// to compute the absolute value accurately. The output of this module is unsigned
// data to the FFT_CAPTURE module.
//
// - SUNDEEP LAL -
/////////////////////////////////////////////////////////////////

module unload_fft(
    clk,
        reset,
        fft_done, // completion signal from FFT core
        fft_dv, // data valid signal from FFT core
        xk_index, // data index from FFT core
        xk_re, // real output from FFT
        xk_im, // imaginary output from FFT
        fft_unload, // unload transform results from FFT core
        index, // 1023 -> 0 index to FFT_CAPTURE
        re, // real output to FFT_CAPTURE
        im, // imaginary output to FFT_CAPTURE
        dv ); // data valid signal to FFT_CAPTURE

// Inputs
input clk, reset, fft_done, fft_dv;
input [10:0] xk_index;
input [11:0] xk_re, xk_im;

// Outputs
output fft_unload;
output [9:0] index;
output [11:0] re, im;
output dv; // data valid to FFT_CAPTURE

// Registers
reg fft_unload;
reg [9:0] index;
reg [11:0] re, im;
reg dv;

// Main process
always @ ( posedge clk )
begin

    if( reset == 1 ) // synchronous reset
    begin
        fft_unload <= 1'b0;
        index <= 10'd0;
        re <= 12'd0;
    end
end
```

```

        im <= 12'd0;
        dv <= 1'b0;
    end

    else
    begin
        if( fft_done == 1 )
            fft_unload <= 1'b1; // pulse fft_unload to start receiving FFT output
        else
            fft_unload <= 1'b0;

        if( fft_dv == 1 )
        begin
            if( xk_index > 1023 ) // only capture lower half of the FFT output
            begin
                if( xk_re[11] == 1 ) // if negative number output from FFT
                    re <= ~xk_re + 1'b1;
                else
                    re <= xk_re;

                if( xk_im[11] == 1 ) // if negative number output from FFT
                    im <= ~xk_im + 1'b1;
                else
                    im <= xk_im;

                index <= index - 1; // decrement index (first state 0 to 1023)..
                dv <= 1'b1; //..this enables reverse order storage of FFT..
                                //..output in FFT_CAPTURE
            end
        end

        else // clear outputs while not receiving from FFT
        begin
            index <= 10'd0;
            re <= 12'd0;
            im <= 12'd0;
            dv <= 1'b0;
        end
    end

end
endmodule

```

[Unit to store FFT output]

```
`timescale 1ns / 1ps

/////////////////////////////////////////////////////////////////
// This code accepts FFT output from the UNLOAD_FFT module in unsigned form. This
// module then squares the real and imaginary parts, adds them together and feeds
// the sum to 4 SQRT units running in parallel, thus computing the absolute peak
// intensity for each frequency bin of the FFT.
//
// - SUNDEEP LAL -
/////////////////////////////////////////////////////////////////

module fft_capture(
    clk,
        reset,
        index, // sample index from 1023 down to 0 from UNLOAD_FFT
        re,
        im,
        dv,
        cfar_busy,
        sqrt_done,
        sqrt_feeda,
        sqrt_feedb,
        sqrt_feedc,
        sqrt_feedd,
        sqrt_start );

// Inputs
input clk; // global clock
input reset; // global reset
input [9:0] index; // FFT output index in reverse order from UNLOAD_FFT
input [11:0] re; // real output from UNLOAD_FFT
input [11:0] im; // imaginary output from UNLOAD_FFT
input dv; // data valid signal from UNLOAD_FFT
input cfar_busy; // signal from CFAR module, mapped to output start_cfar
input sqrt_done; // completion signal from module absval

// Outputs
output [24:0] sqrt_feeda; // output to module absval for calculation
output [24:0] sqrt_feedb;
output [24:0] sqrt_feedc;
output [24:0] sqrt_feedd;
output sqrt_start; // initiate module sqrt for new calculation

// Registers
reg [24:0] sqrt_feeda; // input to module sqrt
reg [24:0] sqrt_feedb; // input to module sqrt
reg [24:0] sqrt_feedc; // input to module sqrt
reg [24:0] sqrt_feedd; // input to module sqrt
reg sqrt_start;

reg start_abs; // internal flag to start calculation of absolute values
reg abs_done;
reg [11:0] re_buf [1023:0]; // memory for real FFT output
reg [11:0] im_buf [1023:0]; // memory for imaginary FFT output
```

```

reg sta; // internal flag
reg [23:0] sq_rea; // square of real part, for absolute value calculation
reg [23:0] sq_ima; // square of imaginary part
reg [23:0] sq_reb; // square of real part, for absolute value calculation
reg [23:0] sq_imb; // square of imaginary part
reg [23:0] sq_rec; // square of real part, for absolute value calculation
reg [23:0] sq_imc; // square of imaginary part
reg [23:0] sq_red; // square of real part, for absolute value calculation
reg [23:0] sq_imd; // square of imaginary part
reg [9:0] indexi;

//-----
// Capture data from FFT core
//-----
always @ ( posedge clk )
begin

    if( reset == 1 )
    begin
        start_abs <= 1'b0;
    end

    // if previous set of FFT data has been processed
    else if( abs_done == 1 )
    begin
        start_abs <= 1'b0; // clear flag
    end

    else if( dv == 1 && start_abs == 0 )
    begin
        re_buf [index] <= re; // index is 1023 -> 0, storing values in reverse
        im_buf [index] <= im;

        if( index == 0 )
            start_abs <= 1'b1; // start absolute value calculation
    end

end

//-----
// Compute absolute value (send to sqrt units)
//-----
always @ ( posedge clk )
begin

    if( reset == 1 )
    begin
        abs_done <= 1'b0;
        sta <= 1'b0;
        sq_rea <= 24'd0; sq_ima <= 24'd0;
        sq_reb <= 24'd0; sq_imb <= 24'd0;
        sq_rec <= 24'd0; sq_imc <= 24'd0;
        sq_red <= 24'd0; sq_imd <= 24'd0;
        sqrt_feeda <= 25'd0;
        sqrt_feedb <= 25'd0;
    end
end

```

```

        sqrt_feedc <= 25'd0;
        sqrt_feedd <= 25'd0;
        sqrt_start <= 1'b0;
        indexi <= 10'd0; /* counter to count up to 1024 values, since only the latter half of the FFT output
                           is considered for CFAR */

    end

    // clear flags
    else if( abs_done == 1 )
    begin
        abs_done <= 1'b0;
        indexi <= 10'd0;
    end

    // only pass new values to sqrt units if CFAR unit is not busy
    else if( reset == 0 && start_abs == 1 && cfar_busy == 0 )
    begin

        // square real and imaginary components
        if( sta == 0 )
        begin
            sq_rea <= re_buf[indexi] * re_buf[indexi]; // re^2
            sq_ima <= im_buf[indexi] * im_buf[indexi]; // im^2
            sq_reb <= re_buf[indexi+1] * re_buf[indexi+1]; // re^2
            sq_imb <= im_buf[indexi+1] * im_buf[indexi+1]; // im^2
            sq_rec <= re_buf[indexi+2] * re_buf[indexi+2]; // re^2
            sq_imc <= im_buf[indexi+2] * im_buf[indexi+2]; // im^2
            sq_red <= re_buf[indexi+3] * re_buf[indexi+3]; // re^2
            sq_imd <= im_buf[indexi+3] * im_buf[indexi+3]; // im^2
            sta <= 1'b1;
        end

        // sum multiplication results from previous cycle
        else if( sta == 1 && sqrt_start == 0 )
        begin
            sqrt_feeda <= sq_rea + sq_ima; // (re^2 + im^2) pass value to first sqrt
            sqrt_feedb <= sq_reb + sq_imb; // (re^2 + im^2) pass value to second sqrt
            sqrt_feedc <= sq_rec + sq_imc; // (re^2 + im^2) pass value to third sqrt
            sqrt_feedd <= sq_red + sq_imd; // (re^2 + im^2) pass value to fourth sqrt
            sqrt_start <= 1'b1; // initiate module sqrt - sqrt( re^2 + im^2 )
        end

        if( sqrt_done == 1 && sqrt_start == 1 )
        begin
            sqrt_start <= 1'b0; // halt sqrt calculation
            sta <= 1'b0; // clear flag

            if( indexi == 1020 )
            begin
                abs_done <= 1'b1; // mark completion of absval calculation
                indexi <= 10'd1023;
            end
            else
                indexi <= indexi + 4; // increment index to previous complex value
            end

        end
    end
end endmodule

```

A7. HDL listing for PSD

```
`timescale 1ns / 1ps

module sqrt(
    clk,
        reset,
        value,
        start,
        root,
        done);

// Inputs
input clk;
input reset; // global synchronous reset
input [24:0] value; // input value to be processed
input start; // start signal

// Outputs
output [12:0] root; // square root of input value
output done; // completion signal

// Internal registers
reg [12:0] root;
reg done;
reg edone;
reg [12:0] error;
reg [25:0] root_square;
reg [3:0] count; // down counter to index individual bits in the root
reg sta;

//-----
// Calculate the square root
//-----
always @ ( posedge clk )
begin

    if( reset == 1 )
    begin
        root <= 13'b1000000000000;
        root_square <= 26'd0;
        error <= 13'd0;
        edone <= 1'b0;
        done <= 1'b0;
        count <= 4'd12;
        sta <= 1'b0;
    end

    // refresh internal variables for new value
    else if( done == 1 )
    begin
        done <= 1'b0;
        count <= 4'd12;
        root <= 13'b1000000000000;
        root_square <= 26'd0;
        error <= 13'd0;
    end
end
```

```

end

else if( edone == 1 )
begin
    done <= 1'b1;
    edone <= 1'b0;
    if( error < root )
        root <= root + 1; // round off result if required
    end
end

// start calculating square root
else if( start == 1 )
begin

    // stage A: square the root
    if( sta == 0 )
    begin
        root_square <= root * root;
        sta <= 1'b1; // set flag for next stage
    end

    // stage B: compare root_square and change the root
    else if( sta == 1 )
    begin
        if( root_square > value ) // if root^2 is greater than value
        begin
            root [count] <= 1'b0; // clear current bit
            if( count > 0 )
                root [count-1] <= 1'b1; // assert next bit
            end
        else if( root_square < value ) // if root^2 is less than value
        begin
            root [count-1] <= 1'b1; // assert next bit
        end

        // adjust down counter
        if( count > 0 )
        begin
            count <= count - 1'b1; // decrement count
        end
        else if( count == 0 ) // if the last bit has been assessed
        begin
            edone <= 1'b1; // signal completion of calculation
            if( root_square > value )
                error <= root_square - value; // compute error
            else if( root_square < value )
                error <= value - root_square; // compute error
            end

            sta <= 1'b0; // reset flag
        end
    end

    end
end

endmodule

```


A8. HDL listing for CFAR

```
`timescale 1ns / 1ps

/////////////////////////////////////////////////////////////////
// This module implements the CA-CFAR algorithm to identify valid targets from
// discrete frequency samples with noise and clutter. These samples are obtained
// by computing the peak intensity for every frequency bin as output from the FFT.
//
// - SUNDEEP LAL -
/////////////////////////////////////////////////////////////////

module cacfar_32(
    clk,
        reset,
        inA, // inA,inB, inC, inD are obtained from 4 different sqrt modules
        inB,
        inC,
        inD,
        start,
        target_abs,
        target_pos,
        new_target,
        start_cfar,
        complete);

// Inputs
input clk;
input reset;
input [12:0] inA, inB, inC, inD;
input start; // start recieving values from sqrt modules
                // mapped to output 'done' on module sqrt

// Outputs
output [12:0] target_abs;
output [9:0] target_pos;
output new_target;
output start_cfar; // signal to module fft_capture to halt during CFAR calculation
output complete; // all 1024 values completed

// Internal registers
reg [12:0] target_abs;
reg [9:0] target_pos;
reg new_target;
reg start_cfar;
reg complete;

reg [12:0] buffer [31:0]; // store 32 cells for CFAR processing
reg [9:0] indexa; // used in buffering data
reg [4:0] indexb; // used in buffering data
reg [4:0] indexc; // for CFAR routine
(* KEEP = "TRUE"*) reg [14:0] avgL; // cell averaging to left of CUT
(* KEEP = "TRUE"*) reg [14:0] avgR; // cell averaging to right of CUT
reg [12:0] avg; // threshold average
reg cfar_done;
reg [1:0] st; // internal flag to sort CFAR stages
```

```

(* KEEP = "TRUE"*) reg [17:0] T; // dynamic threshold result from CFAR processing
reg [4:0] K; // 5-bit decimal constant for CFAR
reg [12:0] CUT;

//-----
// Accept data from module sqrt
//-----
always @ ( posedge clk )
begin
    if( reset == 1 )
    begin
        indexa <= 10'd0;
        indexb <= 5'd0;
        start_cfar <= 1'b0;
    end

    else if( complete == 1 ) // if all 1024 values have been processed
    begin
        indexa <= 10'd0;
        indexb <= 5'd0;
        start_cfar <= 1'b0;
    end

    else if( start == 0 && start_cfar == 1 ) // if CFAR processing is active
    begin
        if( cfar_done == 1 )
        begin
            start_cfar <= 1'b0; // clear signal, proceed with buffering
            indexb <= 5'd0; // reset for next 32 values
        end
        else
        begin
            start_cfar <= 1'b1;
            indexb <= 5'd31; // to avoid truncation by Xilinx ISE
        end
    end

    else if( start == 1 && start_cfar == 0 ) // if CFAR processing is not active
    begin
        buffer[indexb] <= inA;
        buffer[indexb+1] <= inB;
        buffer[indexb+2] <= inC;
        buffer[indexb+3] <= inD;

        if( indexa == 1020 ) // 1024 counter
            indexa <= 10'd1023; // avoid truncation and mark completion of all samples
        else
            indexa <= indexa + 4;

        if( indexb == 28 )
        begin
            indexb <= 5'd0; // 32 counter
            start_cfar <= 1'b1; // start CFAR routine
        end
        else
            indexb <= indexb + 4;
    end
end

```

```

end
end

//-----
// CFAR process
//-----
always @ ( posedge clk )
begin
    if( reset == 1 )
    begin
        new_target <= 1'b0;
        target_abs <= 13'd0;
        target_pos <= 10'd0;
        avg <= 13'd0;
        avgR <= 15'd0;
        avgL <= 15'd0;
        indexc <= 5'd0;
        cfar_done <= 1'b0;
        st <= 2'b00;
        K <= 5'b01011; // setting K = (11111) to avoid truncation
                        //  $K = Pfa^{(-1/(2*M))} - 1$  ; e.g.  $Pfa=10^{-7}$ ,  $M=4$ ,
                        // therefore  $K=6.49 \sim (11010)$ 
                        // K has 3 integer bits, 2 fraction bits

        T <= 18'd0;
        CUT <= 13'd0;
        complete <= 1'b0;
    end

    else if( complete == 1 )
        complete <= 1'b0;

    // After every 32 values or valid target detection
    else if( cfar_done == 1 || new_target == 1 )
    begin
        cfar_done <= 1'b0; // reset flag, ready for next batch of 32 cells
        target_abs <= 13'd0;
        target_pos <= 10'd0;
    end

    // Get the averages for M=4
    else if( start_cfar == 1 && cfar_done == 0 && st == 2'b00 )
    begin
        new_target <= 1'b0; // reset new valid target output signal

        if( indexa >= 10'd0 && indexa <= 10'd511 )
            K <= 5'd20; //  $Pfa = 10^{-7}$ , min.  $K = 5.00$ 
        else if( indexa >= 10'd512 && indexa <= 10'd851 )
            K <= 5'd17; // Reduced  $K = 4.25$  for attenuated medium range targets
        else if( indexa >= 10'd852 )
            K <= 5'd16; // Reduced  $K = 4.00$  for attenuated long range targets

        if( indexc < 6 )
        begin
            avgR <= buffer[indexc+3] + buffer[indexc+4] + buffer[indexc+5]
                    + buffer[indexc+6];
            avgL <= buffer[indexc+3] + buffer[indexc+4] + buffer[indexc+5]

```

```

                                + buffer[indexc+6];
    end
    else if( indexc > 25 )
    begin
        avgR <= buffer[indexc-3] + buffer[indexc-4] + buffer[indexc-5]
            + buffer[indexc-6];
        avgL <= buffer[indexc-3] + buffer[indexc-4] + buffer[indexc-5]
            + buffer[indexc-6];
    end
    else
    begin
        avgR <= buffer[indexc+3] + buffer[indexc+4] + buffer[indexc+5]
            + buffer[indexc+6];
        avgL <= buffer[indexc-3] + buffer[indexc-4] + buffer[indexc-5]
            + buffer[indexc-6];
    end
    st <= 2'b01; // move to next CFAR stage
end

// Add the averages
else if( start_cfar == 1 && cfar_done == 0 && st == 2'b01 )
begin
    avg <= avgR[14:3] + avgL[14:3] + 1; // (avgR/4 + avgL/4)/2 + 1 (to avoid zero)
    st <= 2'b10;
end

// Compute the dynamic threshold
else if( start_cfar == 1 && cfar_done == 0 && st == 2'b10 )
begin
    T <= avg * K; // threshold value for current CFAR cells
    CUT <= buffer[indexc]; // CUT has equal word length as integer part of T
    st <= 2'b11;
end

// Decision to extract valid target from clutter
else if( start_cfar == 1 && cfar_done == 0 && st == 2'b11 )
begin // $display("%d %d", CUT, indexa+indexc-32);
    if( CUT > T[14:2] && CUT > 13'd7 ) // compare integer part and exclude FFT noise
    begin
        new_target <= 1'b1; // assert new valid target signal to pairing module
        target_abs <= CUT; // output target peak intensity
        target_pos <= indexa + indexc - 30; // output target FFT bin number
        K <= 5'b00000; // temporary clear to avoid truncation
    end
    if( indexc == 31 ) // mark completion of CFAR processing on current 32 cells
        cfar_done <= 1'b1;
    if( indexc == 31 && indexa == 1023 ) // if all 1024 samples done
        complete <= 1'b1; // send completion signal to pairing module
    indexc <= indexc + 1; // move to next cell for CFAR processing
    st <= 2'b00;
end

if( new_target == 1 )
    new_target <= 1'b0; // reset new valid target signal

end
endmodule

```

A9. HDL listing for PPM

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////
// This module is responsible for pairing the peaks detected by the CFAR unit
// and producing the target ranges and velocities for all detected targets.
//
// - SUNDEEP LAL -
////////////////////////////////////////////////////////////////

module pairing(
    clk,
        reset,
        new_target,
        target_abs,
        target_pos,
        complete,
        updown,
        unit_vel,
        target_info,
        info_valid
    );

// Inputs to module
input clk; // system/global clock
input reset; // synchronous reset
input new_target; // new valid target from CFAR module
input [12:0] target_abs; // target peak intensity
input [9:0] target_pos; // target frequency bin number
input complete; // CFAR completion signal from CFAR module
input updown; /* sweep direction, 1 for up, 0 for down
                this signal is used inverted (0 for up, 1 for down) because..
                ..during down sweep sampling, up sweep processing is done and..
                .. vice versa */
input [7:0] unit_vel; // radar unit's velocity / car's velocity

// Outputs from module
output [19:0] target_info; // 10 bits target velocity, 10 bits target distance
output info_valid; // signal to display module

// Internal registers
reg [19:0] target_info;
reg info_valid;

reg [12:0] abs_bufup [7:0]; // maximum 8 targets in up sweep
reg [9:0] pos_bufup [7:0];
reg upfill; // flag to mark fully filled up sweep buffers
reg [12:0] abs_bufdown [7:0]; // maximum 8 targets in down sweep
reg [9:0] pos_bufdown [7:0];
reg downfill; // flag to mark fully filled down sweep buffers
reg [2:0] count; // index for up sweep and down sweep buffers
reg [2:0] paircount; // final count of records accepted for pairing from CFAR
reg start_pairing; // flag to commence pairing and output process
reg pairing_done; // flag to mark completion of pairing process
reg [2:0] indexup; // counter to count through up sweep records while pairing
```

```

reg [2:0] indexdown; // counter to count through down sweep records while pairing
reg [2:0] tmpindex; // used to store the final matching pair index
reg [6:0] vel_fac; // multiplication constant for velocity calculation
(* KEEP = "TRUE"*) reg [17:0] velocity; // computed velocity - (13bits).(6bits)
reg [10:0] range_fac; // multiplication constant for range calculation
(* KEEP = "TRUE"*) reg [21:0] range; // computed range - (11bits).(11bits)
reg [1:0] st; // internal flag
reg stb; // internal flag
reg [9:0] posa, posb; // used to analyse spectral closeness during pairing
reg [13:0] absa, absb, absc; // used to analyse peak intensity closeness during pairing
reg [10:0] sum_pos, diff_pos; // sum for range, diff for velocity
reg faster; // 0 if target is slower, 1 is target is faster
reg updone; // mark up sweep processing done

//-----
// Accept data from CFAR module
// - spectral copies are ignored by this module
//-----
always @ ( posedge clk )
begin

    if( reset == 1 )
    begin
        count <= 3'd0;
        paircount <= 3'd0;
        abs_bufup[0] <= 13'd0;
        pos_bufup[0] <= 10'd0;
        abs_bufdown[0] <= 13'd0;
        pos_bufdown[0] <= 10'd0;
        upfill <= 1'b0;
        downfill <= 1'b0;
        start_pairing <= 1'b0;
        updone <= 1'b0;
    end

    // clear pairing process flags
    else if( reset == 0 && pairing_done == 1 )
    begin
        start_pairing <= 1'b0;
        paircount <= 3'd0;
        updone <= 1'b0;
    end

    // if CFAR processing for current sweep direction is complete
    else if( reset == 0 && complete == 1 )
    begin
        if( updown == 0 ) // if up sweep is done
        begin
            paircount <= count; // store the total number of targets for later use
            updone <= 1'b1;
        end

        count <= 3'd0; // reset counter to 0
        upfill <= 1'b0; // clear flags
        downfill <= 1'b0;
    end
end

```

```

        if( updown == 1 && updone == 1 ) // if the down sweep has been completely obtained
        begin
            // $display("%d %d %d %d %d %d %d %d %d %d",pos_bufup[0],pos_bufup[1],pos_bufup[2],pos_bufup[3],pos_bufup[4],pos_bufup[5],pos_bufup[6],pos_bufup[7],
            paircount);
            // $display("%d %d %d %d %d %d %d %d %d %d",pos_bufdown[0],pos_bufdown[1],pos_bufdown[2],pos_bufdown[3],pos_bufdown[4],pos_bufdown[5],pos_bufdown[6],pos_bufdown[7],count);
            start_pairing <= 1'b1;
        end
    end

//-----
// UP SWEEP
//-----
else if( reset == 0 && updown == 0 && new_target == 1 && upfill == 0 )
begin // $display("up %d %d",target_abs,target_pos);

    // first valid target detection stored without 'spectral copy' checking
    if( count == 0 && target_pos > 4 ) // ignore DC values
    begin
        abs_bufup[count] <= target_abs;
        pos_bufup[count] <= target_pos;
        count <= count + 1;
    end

    // 'spectral copy' checking
    else if( count >= 1 )
    begin
        // if new CFAR detection is a 'spectral copy' of previous target
        if( target_pos == pos_bufup[count-1] + 1 )
        begin
            if( target_abs > abs_bufup[count-1] ) // store larger peak intensity
            begin
                abs_bufup[count-1] <= target_abs; // update previous target record
                pos_bufup[count-1] <= target_pos;
            end
        end

        else
        begin
            abs_bufup[count] <= target_abs; // add new target record
            pos_bufup[count] <= target_pos;
            count <= count + 1; // increment counter
            if( count == 7 )
                upfill <= 1'b1; // mark up sweep buffer filled
            end
        end
    end

end

//-----
// DOWN SWEEP
//-----
else if( reset == 0 && updown == 1 && new_target == 1 && downfill == 0 )

```

```

begin //$display("down %d %d",target_abs,target_pos);

    // first valid target detection stored without 'spectral copy' checking
    if( count == 0 && target_pos > 4 ) // ignore DC values
    begin
        abs_bufdown[count] <= target_abs;
        pos_bufdown[count] <= target_pos;
        count <= count + 1;
    end

    // 'spectral copy' checking
    else if( count > 0 )
    begin
        // if new CFAR detection is a 'spectral copy' of previous target
        if( target_pos == pos_bufdown[count-1] + 1 )
        begin
            if( target_abs > abs_bufdown[count-1] ) // store larger peak intensity
            begin
                abs_bufdown[count-1] <= target_abs; // update previous target
                pos_bufdown[count-1] <= target_pos;
            end
        end

        else
        begin
            abs_bufdown[count] <= target_abs; // add new target record
            pos_bufdown[count] <= target_pos;
            count <= count + 1; // increment counter
            if( count == 7 )
                downfill <= 1'b1; // mark up sweep buffer filled
        end
    end

    end

    // clear the record from down buffer when a pair has been matched successfully
    if( st == 2'b10 && start_pairing == 1 )
    begin
        abs_bufdown[tmpindex] <= 13'd0;
        pos_bufdown[tmpindex] <= 10'd0;
    end

end

//-----
// Peak Pairing
// Criteria:
//          (1) +-84 frequency bins
//          (2) compare peak intensity
//-----
always @ ( posedge clk )
begin

    if( reset == 1 )
    begin

```



```

        target_info <= 20'd0;
        info_valid <= 1'b0;
        pairing_done <= 1'b0;
        indexup <= 3'd0;
        indexdown <= 3'd0;
        tmpindex <= 3'd0;
        vel_fac <= 7'b1101101; // (11.01101)binary = (3.40625)decimal
        range_fac <= 11'b00010111110; // (0.00010111110)binary = (0.0927734375)decimal
        /* these factors have been obtained by converting the equations into
           constants, saving hardware and making computation quicker:
            $Fr = 4 * F_{sweep} / T_{sweep} * range / c$  ,  $Fd = 2 * Ft * relative\_velocity / c$  */
        st <= 2'b00;
        stb <= 1'b0;
        posa <= 10'd0;
        posb <= 10'd0;
        absa <= 13'd0;
        absb <= 13'd0;
        absc <= 13'd0;
        sum_pos <= 11'd0;
        diff_pos <= 11'd0;
        faster <= 1'b0;
        velocity <= 18'd0;
        range <= 22'd0;
    end

    // if pairing is complete
    else if( reset == 0 && pairing_done == 1 )
    begin
        target_info <= 20'd0;
        info_valid <= 1'b0;
        pairing_done <= 1'b0;
        indexup <= 3'd0;
        indexdown <= 3'd0;
        tmpindex <= 3'd0;
        st <= 2'b00;
        stb <= 1'b0;
        posa <= 10'd0;
        posb <= 10'd0;
        absa <= 13'd0;
        absb <= 13'd0;
        absc <= 13'd0;
        sum_pos <= 11'd0;
        diff_pos <= 11'd0;
        faster <= 1'b0;
        velocity <= 18'd0;
        range <= 22'd0;
    end

    // pair target peaks from up and down sweeps
    else if( reset == 0 && start_pairing == 1 && indexdown <= paircount-1 )
    begin
        target_info <= 20'd0;
        info_valid <= 1'b0;

        if( st == 2'b00 )
        begin
            // lower limit for criteria (1)

```

```

        if( pos_bufup[indexup] > pos_bufdown[indexdown] )
            posa <= pos_bufup[indexup] - pos_bufdown[indexdown]; // limit to +-84 i.e.
                                                                    300kmph
        else
            posa <= pos_bufdown[indexdown] - pos_bufup[indexup];

        /* calculate peak intensity difference between current up sweep value
           and current down sweep value */
        if( abs_bufup[indexup] > abs_bufdown[indexdown] )
            absa <= abs_bufup[indexup] - abs_bufdown[indexdown];
        else
            absa <= abs_bufdown[indexdown] - abs_bufup[indexup];

        /* calculate peak intensity difference between current up sweep value
           and previously stored best match value */
        if( abs_bufup[indexup] > abs_bufdown[tmpindex] )
            absb <= abs_bufup[indexup] - abs_bufdown[tmpindex];
        else
            absb <= abs_bufdown[tmpindex] - abs_bufup[indexup];

        /* calculate peak intensity difference between next up sweep value
           and previously stored best match value for the current target */
        if( indexup < paircount - 1 ) begin
            if( abs_bufup[indexup+1] > abs_bufdown[tmpindex] )
                absc <= abs_bufup[indexup+1] - abs_bufdown[tmpindex];
            else
                absc <= abs_bufdown[tmpindex] - abs_bufup[indexup+1]; end
        else
            absc <= 13'd8191;

        // ensure next up sweep sample is within +-84 range of previous best match
        if( indexup < paircount - 1 ) begin
            if( pos_bufup[indexup+1] > pos_bufdown[indexdown] )
                posb <= pos_bufup[indexup+1] - pos_bufdown[indexdown];
            else
                posb <= pos_bufdown[indexdown] - pos_bufup[indexup+1]; end
        else
            posb <= 10'd1023;

        st <= 2'b01; // next stage
    end

////////// update best match according to criteria (1,2)
    else if( st == 2'b01 )
        begin
            // if the peak in the down sweep is spectrally close to peak in up sweep
            if( posa < 84 && posa <= posb )
                begin
                    // if current down sweep peak is closer in intensity
                    if( absa <= absb && absa <= absc )
                        tmpindex <= indexdown; // update best match index
                end
            end

            if( indexdown == paircount-1 ) // if all down sweep peaks have been assessed
                st <= 2'b10; // next stage
        else
            begin

```

```

                                indexdown <= indexdown + 1; // move to next down sweep peak
                                st <= 2'b00; // return to re-compute new parameters
                                end
                                end

////////// obtain sum and difference of matched frequency bin indices
                                else if( st == 2'b10 )
                                begin
                                        indexdown <= 3'd0; // clear index to restart from first record in down sweep
                                        sum_pos <= pos_bufup[indexup] + pos_bufdown[tmpindex]; // for target range

                                        if( pos_bufdown[tmpindex] > 0 ) begin
                                                // for target relative velocity
                                                if( pos_bufup[indexup] > pos_bufdown[tmpindex] ) // slower target
                                                begin
                                                        diff_pos <= pos_bufup[indexup] - pos_bufdown[tmpindex];
                                                        faster <= 1'b0;
                                                end
                                                else // faster target
                                                begin
                                                        diff_pos <= pos_bufdown[tmpindex] - pos_bufup[indexup];
                                                        faster <= 1'b1;
                                                end
                                                end

                                        st <= 2'b11; // next stage
                                        end

                                else begin
                                        if( indexup < paircount - 1 )
                                        begin
                                                indexup <= indexup + 1;
                                                st <= 2'b00;
                                        end
                                        else
                                                pairing_done <= 1'b1;
                                        end
                                end

                                end

////////// compute the velocity and range and output as single bus
                                else if( st == 2'b11 )
                                begin
                                        if( stb == 0 ) // stage to compute velocity and range
                                        begin
                                                if( faster == 0 ) // if the target is not faster than own vehicle
                                                        velocity <= vel_fac * diff_pos;
                                                else // if the target is faster than own vehicle
                                                        velocity <= vel_fac * diff_pos;

                                                range <= range_fac * sum_pos;
                                                stb <= 1'b1;
                                        end

                                        else // final step: output target_info, update indexup
                                        begin
                                                if( faster == 0 ) // extract (9bits).(0bit) velocity
                                                        target_info[19:11] <= unit_vel - velocity[13:5];

```

```

else
    target_info[19:11] <= unit_vel + velocity[13:5];

    target_info[10] <= velocity[4]; // attach the fraction bit

    target_info[9:0] <= range[18:9]; // extract (8bits).(2bits) range
    info_valid <= 1'b1; // alert display unit of valid target information
    tmpindex <= 3'd0;
    posa <= 10'd0;
    posb <= 10'd0;
    absa <= 13'd0;
    absb <= 13'd0;
    stb <= 1'b0;
    st <= 2'b00; // reset to first state
    indexup <= indexup + 1; // move to next record in up sweep buffer

    if( indexup == paircount ) // if all records have been assessed
        pairing_done <= 1'b1;
    end
end
end
end
endmodule

```

VITA AUCTORIS

Sundeeep Lal was born in 1984 in New Delhi, India. He completed his first degree in engineering titled M. Eng. (Hons.) in Electronic and Computer Engineering from the University of Nottingham (UK) in 2007, during which he underwent industrial training at Philips Semiconductors SDN. BHD. (Petaling Jaya, Malaysia). His research interests include Processor Architecture, Digital Signal Processing on FPGAs, Neurofuzzy Control and Optimization Algorithms, and Microelectromechanical Systems (MEMS). At the time of writing this thesis Sundeeep is a member of the MEMS Lab, and a candidate for the degree of M. A. Sc. in Electrical and Computer Engineering, at the University of Windsor (Ontario, Canada).