



Cape Peninsula
University of Technology

Nano-satellite GPS Receiver Design and Implementation: A Software-to-Firmware Approach

by

Nganyang Paul Bayendang

Thesis submitted in partial fulfilment of the requirements for the degree

Master of Technology: Electrical Engineering

in the Faculty of Engineering

at the Cape Peninsula University of Technology

Supervisor: **Prof Elmarie Biermann**

Co-supervisors: **Prof Robert Van Zyl and Mr Arno Barnard**

Bellville Campus

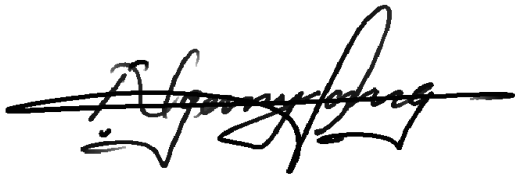
Date submitted: **22 January 2015**

CPUT copyright information

This thesis may not be published either in part (in scholarly, scientific or technical journals), or as a whole (as a monograph), unless permission has been obtained from the University.

DECLARATION

I, Nganyang Paul Bayendang, declare that the contents of this thesis represent my own unaided work, and that this thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.



Signed

22 January 2015

Date

ABSTRACT

Space-borne GPS receivers designed for nano-satellites are faced with various challenges. This research is undertaken to address the problems of inefficiency and high-costs associated with space-borne GPS receivers. The problem of inefficiency relates to poor performances of the GPS receiver in terms of the algorithmic models, execution speed, memory usage and errors proneness. The problem of high-costs relates to the spacegrade hardware cost, implementation complexity, development time, as well as the manufacturing, production and the testing processes involved.

The research objectives are to i) establish an efficient high-dynamics software-defined GPS receiver, ii) demonstrate a firmware approach and then iii) postulate a low-cost hardware implementation roadmap. The research methodology employed to address the problems and to attain the objectives is based-on using Matlab computing platform to i) implement a software-defined GPS receiver using free open-source GPS receiver algorithms, ii) further develop the software GPS receiver and lastly iii) convert the improved GPS receiver algorithms to firmware. The GPS receiver was successfully implemented in Matlab floating-point algorithms with a $\pm 100\text{kHz}$ Doppler search bins and was used to post-process a pre-captured real GPS L1 C/A signal dataset. The pre-captured GPS signal was acquired, tracked, decoded and post-processed to extract the navigation message; use to compute the GPS receiver position, UTC date and time.

Attempt to convert the entire Matlab floating-point GPS receiver algorithms to equivalent VHDL implementations failed; however, three of the Matlab floating-point algorithms (`check_t.m`, `deg2dms.m` and `findUtmZone.m`), were successfully converted to equivalent fixed-point formats in Matlab, Simulink and finally VHDL. These three algorithms, now created and optimised to fixed-point formats (efficient and enable implementation unto a low-cost microcontroller), set the basis for the firmware implementation. They were simulated and verified in Matlab, Simulink and VHDL using the Matlab HDL Coder workflow. Altera Quartus II software was then used to compile (synthesise, place & route and generate programming files) the three converted generic VHDL algorithms to embedded firmware, suitable for a FPGA programming.

The Matlab HDL Coder workflow used in this research is feasible and can be used to accurately design and implement an improved GPS receiver and furthermore achieve it in three equivalent algorithms. This conclusion was drawn and the proposed recommendations are to address the conversion issues in the other Matlab floating-point GPS receiver algorithms that failed in the conversion process and to further develop and implement the GPS receiver as a fully functional unit, based-on the Xilinx space-grade, radiation hardened and low-cost Virtex 5QV FPGA.

ACKNOWLEDGEMENTS

I wish to thank:

- CPUT and F'SATI for establishing the CubeSat space programme and research facilities.
- French South African Institute of Technology (F'SATI) for the sponsorship.
- Prof Elmarie Biermann of INFOBAHN for her supervision – manuscript proof-reading.
- Mr Arno Barnard of University of Stellenbosch for his co-supervision – technical advice.
- Prof Robert Van Zyl and Mr Francois Visser of F'SATI for coordinating the research.
- Staffs and students of F'SATI / CPUT for their lectures and support where applicable.
- IEEE for establishing and providing GPS information on the IEEEExplorer website.
- MathWorks for creating Matlab and Simulink with associated tools and resources.
- Creators of GNSS technologies and authors who have published materials on GNSS.
- Danish GPS Centre, specifically for making their GPS resources truly open-source.
- All the search engines, online information databases and websites on GNSS.

The financial assistance of F'SATI towards this research is personally acknowledged. Opinions expressed in this thesis and the conclusions arrived at, are those of the author, and are not necessarily to be attributed to F'SATI.

DEDICATION

This research work is dedicated to the past, present and future members of Bayendang's family (based in Mamfe Town from Tali II village) and the indigenous scholar sons and daughters of the great Manyu Division in South West Province – known for being highly intelligent in Cameroon.

GLOSSARY

1U CubeSat	Nano-satellite of form-factor; 10 x 10 x 10 cm ³ , weighs < 1.3 kg and uses < 1W
3U CubeSat	Nano-satellite of dimension; 10 x 10 x 30 cm ³ , weighs < 4 kg and uses < 5W
ACS	Attitude Control Sub-system
ADC	Analog to Digital Converter
ADCS	Altitude Determination & Control Systems
ASIC	Application Specific Integrated Circuit
BASS	Block Adjustment of Synchronising Signals
BOC	Binary Offset Carrier
BPSK	Binary (Bi) Phase Shift Keying
C/A	Coarse / Acquisition
CDMA	Code Division Multiple Access
CDMF	Conventional Digital Matched Filter
COCOM	Coordinating Committee for Multilateral Export Controls
COTS	Commercial Off The Shelf
CPU	Central Processing Unit
CPUT	Cape Peninsula University of Technology
CS	Communication Sub-systems
CSC	Combination of Squared Correlators
CSOC	Consolidated Space Operations Center
DCM	Direction Cosine Matrix
DDMF	Differential Digital Matched Filter
DFT	Direct Fourier Transform
DLL	Delay Lock Loop
DOP	Dilution Of Precision
DSP	Digital Signal Processing
DSSS	Direct Sequence Spread Spectrum
ECEF	Earth-Centered Earth-Fixed
eCOS	embedded Configurable Operating System
EGNOS	European Geostationary Navigation Overlay Service
EIRP	Equivalent Isotropic Radiated Power
EPL	Early, Prompt and Late
EKF	Extended Kalman Filter
ERTOS	Embedded Real-Time Operating Systems
F'SATI	French South African Institute of Technology
FFT	Fast Fourier Transform
FLL	Frequency Lock Loop
FPGA	Field Programmable Gate Array
GDOP	Geometric Dilution Of Precision
GENSO	Global Educational Network for Satellite Operation
GNSS	Global Navigation Satellite Systems – (GPS, GLONASS, Galileo and Compass)
GPL	General Public License
GPS	Global Positioning System (USA GNSS)
GSOC	German Space Operation Centre

<u>G</u> UI	Graphical User Interface
HDOP	Horizontal Dilution Of Precision
<u>H</u> OW	How-Over Word
I	In-phase
IC	Integrated Circuit
IEEE	Institutes of Electrical and Electronic Engineers
IF	Intermediate Frequency
IFT	Inverse Fourier Transform
IFFT	Inverse Fast Fourier Transform
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IODE	Issue Of Date Ephemeris
IRNSS	Indian Regional Navigation Satellite System
<u>I</u> TAR	International Traffic in Arms Regulations
L1/L2/L5	Link 1 / Link 2 / Link 5
LEO	Low Earth Orbit (at an altitude above Earth, between ~100 km and ~1000 km)
LFSR	Linear Feedback Shift Registers
LHCP	Left Hand Circular Polarisation
LNA	Low Noise Amplifier
LOS	Line Of Sight
<u>L</u> SB	Least Significant Bit
MCB	Massive Correlator Banks
MEO	Medium Earth Orbit (at an altitude above Earth from ~10000 km to ~20000 km)
MLS	Maximum Length Sequence
MSAS	MTSAT Satellite Augmentation System
MSB	Most Significant Bit
<u>M</u> TSAT	Multifunctional Transport Satellite
Nanosatellite	A very small satellite of mass between 1 and 10 kg
NASA	National Aeronautics Space Administration
NCO	Numerically Controlled Oscillator
<u>N</u> ORAD	North American Aerospace Defense Command
OBC	Onboard Computer
<u>O</u> S	Operating System
P-POD	Poly-Picosatellite Orbital Deployer
PDOP	Position Dilution Of Precision
PLL	Phase Lock Loop
ppm	parts per million
PPS	Precise Positioning Service
PRN	Pseudo Random Noise
PS	Power Sub-system
PVT	Position, Velocity and Time
<u>P</u> (Y)-code	Precision code
Q	Quadrature

<u>Q</u> ZSS	Quasi-Zenith Satellite System
RAAN	Right Ascension of the Ascending Node
RAD	Rapid Applications Development
RAM	Random Access Memory
RF	Radio Frequency
RHCP	Right Hand Circular Polarisation
RICA	Reconfigurable Instruction Cell Array
RMS	Root Mean Square
ROM	Read Only Memory
RTOS	Real-Time Operating System
SA	Selective Availability
SAA	South Atlantic Anomaly
SBAS	Satellite Based Augmentation Systems
SCB	Supplementary Correlator Banks
SDLC	Software / System Development Life Cycle
SNR	Signal to Noise Ratio
SOC	System On Chip
SPDMF	Segment Processing Digital Matched Filter
SPS	Standard Positioning Service
SS	Structural Sub-system
SSTL	Surrey Satellite Technologies Ltd
STL	Scalar Tracking Loop
SUN	University of Stellenbosch
<u>S</u> V	Space Vehicle (commonly known as Satellite)
TCS	Thermal Control Sub-system
TDOP	Time Dilution Of Precision
TEC	Total Electrons Content
TLE	Two-Line Elements
TLM	Telemetry
TOW	Time Of Week
TT&C	Tracking, Telemetry & Control
<u>T</u> TFF	Time To First Fixed
USERE	User Equivalent Range Error
U	Unit
UTC	Universal Coordinated Time
<u>U</u> TM	Universal Time Mercator
VDOP	Vertical Dilution Of Precision
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
<u>V</u> TL	Vector Tracking Loop
WAAS	Wide Area Augmentation System
<u>W</u> LS	Weighted Least Squares
XMC	Extended Multiple Correlator

Table of Contents

DECLARATION.....	II
ABSTRACT.....	III
ACKNOWLEDGEMENTS	IV
DEDICATION	V
GLOSSARY	VI
CHAPTER 1.....	17
1 INTRODUCTION	17
1.1 Thesis Overview.....	17
1.2 Nano-satellites Overview.....	17
1.2.1 CubeSats Background.....	18
1.2.2 CubeSats Specifications.....	19
1.2.3 CubeSats Objectives	20
1.2.4 CubeSats Sub-systems.....	20
1.2.5 CubeSats Applications	21
1.2.6 CubeSats Past Missions	22
1.2.7 CubeSats Constellations	22
1.3 Research Problem Statement	23
1.4 Research Problem Background	23
1.5 Research Literature Review	25
1.5.1 Hardware-defined Approach by Using COTS Devices.....	27
1.5.2 Hardware-defined Approach by Modifying Commercial GPS Receiver	27
1.5.3 Software-defined / Firmware-defined Approaches	28
1.5.4 GPS Receivers Acquisition and Tracking Algorithms	30
1.6 Research Project Objectives.....	32
1.6.1 Secondary Objective: Develop Software Receiver / Address First Research Problem	32

1.6.2	Tertiary Objective: Firmware GPS Receiver Design and Implementation.....	32
1.7	Research Project Design Methodology	33
1.8	Research Project Delineations	38
1.9	Research Project Outcomes	38
1.9.1	GPS Receiver Expected Results	38
1.9.2	Research Project Significance	39
1.9.3	Research Project Scientific Contributions.....	39
1.10	Thesis Structure.....	41
1.11	Summary.....	41
CHAPTER 2.....	42
2	GPS BACKGROUND	42
2.1	Introduction	42
2.2	Global Navigation Satellite System (GNSS) Technologies	42
2.3	GPS Ground Stations.....	43
2.4	GPS Satellites.....	45
2.5	GPS Signals	47
2.5.1	GPS Signal Carrier Frequency.....	47
2.5.1.1	GPS Signal Carrier Frequencies Selection Criteria.....	47
2.5.1.2	Types of GPS Receiver Carrier Frequencies	48
2.5.1.2.1	GPS L1 Signal Carrier Frequency.....	48
2.5.1.2.2	GPS L2 Signal Carrier Frequency.....	48
2.5.2	GPS Signal Pseudo-Random Noise (Gold) Codes.....	49
2.5.2.1	GPS C/A - Code.....	49
2.5.2.2	GPS P(Y) - Code	49
2.5.2.3	GPS C/A - Code Generation	50
2.5.2.4	GPS P(Y) - Code Generation.....	52
2.5.3	GPS Signal Navigation Data Message	53
2.5.3.1	GPS Signal Navigation Data Message Technical Parameters Summary.....	54
2.5.4	GPS L1 Signal Modulation Schemes	56

2.5.4.1	Direct Sequence Spread Spectrum (DSSS)	56
2.5.4.2	Code Division Multiple Access (CDMA)	57
2.5.4.3	Binary Phase Shift Keying (BPSK)	57
2.6	Summary	59
 CHAPTER 3.....		60
3	GPS RECEIVER	60
3.1	Introduction	60
3.2	Hardware-defined GPS Receivers	61
3.2.1	Using Discrete GPS Receiver COTS Components and Devices	62
3.2.2	Modifying COTS GPS Receiver Unit	62
3.2.3	Porting or Emulating the Internal Architecture of the GPS Receiver Hardware	62
3.2.3.1	Emulation	62
3.2.3.2	RTOS.....	62
3.3	Software-defined GPS Receivers.....	63
3.3.1	Software-defined GPS Receivers Based-on Matlab Only.....	63
3.3.2	Software-defined GPS Receivers Based-on Simulink Only.....	63
3.3.3	Software-defined GPS Receivers Based-on Both Matlab and Simulink.....	63
3.4	Firmware-defined GPS Receivers	64
3.4.1	Conversion of Software to Firmware in a Similar Language	64
3.4.2	Conversion of Software to Firmware in a Different Language.....	64
3.5	GPS Receiver Functional Blocks	65
3.5.1	GPS RF Section (GPS Front-end).....	66
3.5.2	GPS DSP Section (GPS Correlator).....	68
3.5.2.1	Correlation.....	68
3.5.2.1.1	Cross-correlation	69
3.5.2.1.2	Auto-correlation	69
3.5.2.2	Acquisition.....	71
3.5.2.3	Tracking	72
3.5.3	GPS Embedded Section (GPS Baseband Processor)	74
3.5.3.1	Decoding.....	74
3.5.3.1.1	Bit Synchronisation.....	74
3.5.3.1.2	Locating Preamble.....	74

3.5.3.1.3	Extracting Navigation Message	75
3.5.3.2	Processing.....	76
3.5.3.2.1	GPS Satellites Position	76
3.5.3.2.2	GPS Satellites to Receiver Pseudo-range	79
3.5.3.2.3	GPS Receiver Position	80
3.5.3.2.4	GPS Receiver Velocity	81
3.5.3.2.5	GPS Receiver Time.....	81
3.6	GPS Errors Sources.....	82
3.7	GPS Errors Sources Mitigation	83
3.7.1	GPS Ground Station Associated Errors / Sources	83
3.7.1.1	Selective Availability.....	83
3.7.1.2	Human Mistakes	83
3.7.1.3	Equipments / Devices Related Errors	83
3.7.2	GPS Satellites Associated Errors / Sources	83
3.7.2.1	Number of Visible or Available GPS Satellites.....	83
3.7.2.2	Ephemeris Parameters Errors	84
3.7.2.3	GPS Satellite Geometry (DOP) Errors	84
3.7.2.4	GPS Satellites Clock Errors	84
3.7.3	GPS Signal Associated Errors / Sources	85
3.7.3.1	Ionospheric Propagation or Scintillation Effect	85
3.7.3.2	Tropospheric Propagation Effect	86
3.7.3.3	Multipath Effect.....	87
3.7.3.4	Doppler’s Frequency Shift Effect.....	88
3.7.3.5	Jamming / Spoofing Signals Effect	88
3.7.4	GPS Receiver Associated Errors / Sources	89
3.7.4.1	GPS Receiver Clock Errors.....	89
3.7.4.2	GPS Receiver’s Noise	89
3.7.4.3	Software Errors	89
3.7.4.4	Antenna Phase Center Offset / Variation	89
3.7.5	User Equivalent Range Error (UERE) Budget.....	90
3.8	Space-borne GPS Receiver Design Considerations	91
3.8.1	Design Considerations – Physical Challenges.....	91
3.8.1.1	Radiation	91
3.8.1.2	Extreme Temperatures.....	91
3.8.1.3	Vacuum	91
3.8.2	Design Considerations – Non-physical Challenges	92

3.8.2.1	Doppler Frequency Shift.....	92
3.8.2.2	GPS Orbital Geometry	93
3.8.2.3	Multipath Effect.....	93
3.8.2.4	GPS Receiver Antenna Orientation	93
3.8.2.5	GPS Signal Strength.....	93
3.8.2.6	Atmospheric Effect.....	93
3.9	Summary	94
 CHAPTER 4.....		95
4	GPS RECEIVER DESIGN AND IMPLEMENTATION	95
4.1	Introduction	95
4.2	GPS Receivers Design Specifications	95
4.2.1	The Software-defined GPS Receiver Design Specifications.....	95
4.2.2	The Firmware-defined GPS Receiver Design Specifications.....	96
4.3	GPS Receiver Algorithms Design and Implementation	96
4.3.1	Software-defined GPS Receiver: Phase 1 Based-on Matlab	96
4.3.1.1	GPS Receiver Software Path Initialisation	97
4.3.1.2	GPS Receiver Software Settings Initialisation	97
4.3.1.3	GPS L1 C/A Signal Acquisition Flowchart.....	98
4.3.1.4	GPS L1 C/A Signal Tracking Flowchart	99
4.3.1.5	GPS L1 C/A Signal Positioning Flowchart.....	100
4.3.1.6	GPS L1 C/A Receiver Position Computation Summary Flowchart	103
4.3.1.7	GPS L1 C/A Receiver Velocity Computation Flowchart	104
4.3.1.8	GPS L1 C/A Receiver UTC Date and Time Computations Flowchart.....	104
4.3.2	Software-to-Firmware Conversion: Phase 2 Based-on Matlab / Simulink and VHDL.....	107
4.3.2.1	Software-to-Firmware (Matlab to VHDL Algorithms) Conversion Procedure.....	109
4.3.2.1.1	Step 1: Ensure all GPS Matlab Functions, Scripts and Test Vectors are in a Local Folder	109
4.3.2.1.2	Step 2: Simulate / Verify the Floating-point Matlab Software GPS Receiver Algorithms.....	109
4.3.2.1.3	Step 3: Create a New GPS VHDL Coder Project	110
4.3.2.1.4	Step 4: Convert the GPS Matlab Codes from Floating-point to Fixed-point.....	111
4.3.2.1.5	Step 5: Generate GPS HDL Codes (Convert Fixed-point Matlab Algorithms to VHDL).....	116
4.3.3	Firmware-defined GPS Receiver: Phase 3 Based-on VHDL	122
4.3.3.1	Simulation using Matlab Test-bench Workflow.....	122
4.3.3.2	Simulation using Matlab Algorithms Workflow	123

4.3.3.3	Simulation using Simulink Co-simulation Workflow	124
4.3.3.4	Verification with FPGA-in-the-loop Wizard	126
4.3.3.5	Simulating and Verifying the Generated VHDL GPS Receiver Algorithms	126
4.3.4	Low-cost Space-borne GPS Receiver Hardware Implementation Roadmap.....	127
4.4	GPS Receiver Algorithms Test and Validation Procedure	129
4.5	Summary	129
 CHAPTER 5.....		130
5	GPS RECEIVER / ALGORITHMS RESULTS AND INTEPRETATIONS	130
5.1	Introduction	130
5.2	Researched GPS Receiver and the Discrete Generated Algorithms Results	130
5.2.1	Software-defined GPS Receiver Results Based-on Matlab Floating-point Algorithms	130
5.2.2	Software-defined GPS Receiver Results Based-on 5 Matlab Fixed-point Algorithms.....	137
5.2.2.1	calcLoopCoef.m Equivalent Matlab Fixed-point Algorithm Result.....	138
5.2.2.2	check_t.m Equivalent Matlab Fixed-point Algorithm Result	138
5.2.2.3	clsin.m Equivalent Matlab Fixed-point Algorithm Result	140
5.2.2.4	deg2dms.m Equivalent Matlab Fixed-point Algorithm Result	142
5.2.2.5	findUtmZone.m Equivalent Matlab Fixed-point Algorithm Result.....	144
5.2.3	Software-defined GPS Receiver Results Based-on 5 Discrete Simulink Models.....	146
5.2.3.1	calcLoopCoef.m Equivalent Simulink Models Results Comparison	146
5.2.3.2	check_t.m Equivalent Simulink Models Results Comparison	146
5.2.3.3	clsin.m Equivalent Simulink Models Results Comparison.....	147
5.2.3.4	deg2dms.m Equivalent Simulink Models Results Comparison	148
5.2.3.5	findUtmZone.m Equivalent Simulink Models Results Comparison	148
5.2.4	Firmware-defined GPS Receiver Results Based-on 5 Discrete VHDL Algorithms	149
5.2.4.1	calcLoopCoef.m Equivalent VHDL Algorithm Result.....	149
5.2.4.2	check_t.m Equivalent VHDL Algorithm Result	149
5.2.4.3	deg2dms.m Equivalent VHDL Algorithm Result	151
5.2.4.4	findUtmZone.m Equivalent VHDL Algorithm Result.....	153
5.2.4.5	clsin.m Equivalent VHDL Algorithm Result	155
5.2.4.6	Summary of the GPS Receiver Algorithms Results Comparison and Error Analysis	155
5.2.4.7	Firmware Compilation Results of check_t, deg2dms and findUtmZone VHDL Algorithms	156
5.3	The GPS Receiver and Discrete Generated Algorithms Results Interpretations.....	158

5.3.1	Software-defined Matlab Floating-point GPS Receiver Results Interpretations.....	158
5.3.2	The GPS Receiver Generated Algorithms Results Interpretations	160
5.4	Summary	160
CHAPTER 6.....		161
6	CONCLUSIONS AND RECOMMENDATIONS	161
6.1	Conclusions.....	161
6.2	Recommendations.....	162
6.2.1	Recommendations to issues encountered	162
6.2.2	Recommendations for future research	163
APPENDIX A		164
7	GPS SIGNAL L1 C/A ACQUISITION AND TRACKING TECHNIQUES	164
7.1	Introduction	164
7.2	GPS L1 C/A Signal Acquisition Techniques	164
7.2.1	Mitel GP2000 Acquisition and Tracking Algorithms.....	164
7.2.2	Tong or Serial Search (Time Domain Correlation) Acquisition Technique	165
7.2.3	Parallel Frequency Space Search (Delay & Multiply FFT) Acquisition Technique.....	166
7.2.4	Parallel Code Phase Search (Circular Cross-correlation) Acquisition Technique.....	167
7.2.5	Block (Non-coherent Integration) Acquisition Technique	168
7.2.6	Acousto-optic Acquisition Technique.....	169
7.2.7	Multiple Correlators Bank Acquisition Technique.....	170
7.2.8	Extended Multiple Correlators Acquisition Technique.....	170
7.2.9	Matched Filter Correlator Acquisition Technique	171
7.2.10	Reconfigurable Instruction Cell Array Acquisition Technique	171
7.2.11	Averaging Correlator Acquisition Technique.....	172
7.2.12	Bayesian Acquisition Technique	173
7.2.13	Wavelet De-noising Acquisition Technique.....	173
7.2.14	Lifting Wavelet Acquisition Technique	173
7.2.15	Wavelet Transform Acquisition Technique	173
7.3	GPS L1 Signal Tracking Techniques.....	174

7.3.1	PLL and Costas Loop Carrier Tracking Techniques	175
7.3.2	Basic DLL Early-Late Code Tracking Technique	176
7.3.3	Robust DLL Early-Late Code Tracking Technique	177
7.3.4	Combination of Carrier and Robust DLL Code Tracking Techniques	178
7.3.5	Block Adjustment of Synchronising Signals Tracking Technique.....	179
7.3.6	Combination of Squared Correlators Tracking Technique.....	179
7.3.7	Nonlinear Code Tracking Filter Technique.....	180
7.3.8	Kalman Filter Tracking Loop Technique.....	180
7.3.9	Extended Kalman Filter Tracking Technique.....	181
7.3.10	Inertial Navigation System Tracking Technique	182
7.4	Summary	182
APPENDIX B		183
8 GPS APPLICABLE MATHEMATICAL ANALYSIS		183
8.1	Introduction	183
8.2	GPS L1 C/A Signal Analysis.....	183
8.2.1	GPS L1 Carrier Frequency Analysis.....	184
8.2.2	GPS C/A – Code Analysis.....	184
8.2.3	GPS Navigation Data Message Analysis.....	186
8.2.4	GPS L1 C/A Signal Acquisition Analysis.....	186
8.2.4.1	Parallel Code Phase Search (Circular Cross-Correlation) Acquisition Technique Analysis	187
8.2.5	GPS L1 C/A Signal Tracking Analysis	188
8.2.5.1	PLL Analysis.....	188
8.2.5.2	Carrier Tracking Analysis	190
8.2.5.3	Code Tracking Analysis	191
8.2.6	GPS Navigation Data Message Decoding Analysis.....	192
8.2.7	GPS Navigation Data Message Processing Analysis.....	193
8.2.7.1	GPS Satellite Position Computation Analysis	193
8.2.7.2	Transformation of GPS Satellites Position Orbital Frame to ECEF Coordinate System.....	194
8.2.7.3	Pseudo-range Estimation Analysis	195
8.2.7.4	GPS Time Correction Analysis.....	195
8.2.7.5	GPS Receiver Position Computation Analysis	197
8.2.7.5.1	Linearisation of the Pseudo-range Observations	198
8.2.7.5.2	Application of Least-squares Method to the Linearised Observations.....	199
8.2.7.6	GPS Receiver Velocity Computation Analysis	201

8.2.7.7	GPS Receiver Position Transformation from XYZ (ECEF) to Geodetic Coordinate.....	203
8.2.7.7.1	Newton’s Method for Cartesian (XYZ) to Geodetic ($\varphi \lambda h$) Transformation	203
8.2.7.8	Dilution of Precision (DOP) Analysis.....	206
8.2.7.9	Maximum Doppler Shift Computation Analysis	207
8.3	Summary	208
APPENDIX C		209
9	GPS EXTRA MISCELLANEOUS RESULTS.....	209
9.1	Floating-point Matlab GPS Receiver Acquisition Results.....	209
9.2	Floating-point Matlab GPS Receiver Tracking Results	211
9.3	First 4 GPS Satellites Orbital Parameters Preview.....	216
9.4	Researched GPS Receiver Algorithms with Problems during the Conversion Process	217
9.4.1	acquisition.m: performs cold start searching for available GPS satellites signals	217
9.4.2	calculatePseudoranges.m: computes the range error between satellites and receiver	217
9.4.3	cart2geo.m: transforms rectangular (ECEF) coordinates to geographic.....	218
9.4.4	cart2utm.m: converts Cartesian (XYZ) coordinates to ENU in UTM	218
9.4.5	checkPhase.m: checks the phase of the decoded navigation data message	219
9.4.6	clkSin.m: provides real and imaginary output for use by cart2utm.m.....	219
9.4.7	dms2mat.m: splits deg2dms.m output to hours, minutes, seconds and milli-seconds.....	220
9.4.8	e_r_corr.m: corrects errors in GPS satellites orbital positions	220
9.4.9	ephemeris.m: decodes ephemeris data and time of week from the navigation data	221
9.4.10	findPreambles.m: deduces first sub-frame or the start of the navigation message.....	221
9.4.11	generateCAcode.m: creates the 1023 C/A code phases for any 32 GPS PRN code	222
9.4.12	gps_time.m: extracts mean GPS week number and second from ephemeris data	223
9.4.13	gps2utc.m: computes UTC date and time from the extracted GPS time	223
9.4.14	init_tracking.m: prepares the acquired GPS data for tracking	224
9.4.15	invert.m: inverts applicable navigation bits for use by checkPhase.m	224
9.4.16	leastSquarePos.m: computes GPS receiver position using least-squares approach.....	225
9.4.17	makeCaTable.m: pre-computes / creates a lookup table for all 32 GPS PRN codes.....	225
9.4.18	navPartyChk.m: checks the parity of the decoded navigation data sub-frames	226
9.4.19	postNavigation.m: performs post-processing of the decoded GPS data	227
9.4.20	satpos.m: compute GPS satellites position (crashes Matlab R2013a)	227
9.4.21	showChannelStatus.m: display the acquisition and tracking results in a table	228

9.4.22	togeod.m: calculates geodetic coordinates from Cartesian coordinates.....	228
9.4.23	topocent.m: transforms changing position coordinates to topocentric coordinates	229
9.4.24	tracking.m: tracks the acquired GPS data to extract navigation bit streams	230
9.4.25	tropo.m: corrects tropospheric errors in pseudo-ranges and carrier phases	230
9.4.26	twosComp2dec.m: converts a two-complement binary number.....	231

REFERENCES.....232

List of Figures

Figure 1.1: Model of a CubeSat (left) and various types of designed CubeSats (right).....	18
Figure 1.2: Illustration of three 1U CubeSats inside a P-POD.....	19
Figure 1.3: CubeSat architecture showing basic sub-systems.....	20
Figure 1.4: Polar (left) and Walker (right) orbits constellations	22
Figure 1.5: A single orbital plane overview representation of GPS technology (not to scale)	26
Figure 1.6: Software-defined GPS receiver architecture	28
Figure 1.7: GPS signal acquisition serial search technique	31
Figure 1.8: Concise summary of the research project objectives.....	32
Figure 1.9: RAD SDCL methodology.....	33
Figure 1.10: Internal functional blocks and fundamental tasks of the GPS receiver under research	34
Figure 1.11: Various development analyses and simulation tools	35
Figure 1.12: Summary of the software-to-firmware GPS receiver design methodology.....	37
Figure 1.13: Thesis organisation.....	40
Figure 2.1: GPS ground control segment	44
Figure 2.2: GPS ground control sites	44
Figure 2.3: GPS satellites constellation.....	45
Figure 2.4: GPS DOP (left), Good DOP (middle) and Poor DOP (right).....	46
Figure 2.5: GPS ground track as noted by the red dotted line	46
Figure 2.6: Generation of PRN C/A Gold codes.....	50
Figure 2.7: Illustration of a typical GPS navigation data message composition – a super-frame.....	53
Figure 2.8: GPS signal navigation message – frame compositions in 3D	55
Figure 2.9: GPS signal navigation message – TLM and HOW	55
Figure 2.10: GPS C/A and P(Y) codes – spreading the GPS L1 signal.....	56
Figure 2.11: Concise summary of GPS signal theory composition.....	58
Figure 2.12: Simplified summary of GPS L1 signal structure.....	59
Figure 2.13: Detailed summary of GPS L1 and L2 signals generation.....	59
Figure 3.1: Hardware and software trends of GPS receiver.....	60
Figure 3.2: Summary of a typical software GPS receiver depicting functional blocks	65
Figure 3.3: GPS front-end basic functional blocks	66
Figure 3.4: GPS receiver front-end IC architecture.....	67
Figure 3.5: Illustration of correlation concept using two 32 bits PRN code patterns.....	68
Figure 3.6: Conceptual received GPS L1 C/A signal transmitted by SV 21 showing first 50 bits	70
Figure 3.7: GPS receiver generated replica of GPS L1 C/A signal of SV 21 showing first 50 bits	70
Figure 3.8: Auto (left) and cross (right) correlation results.....	70
Figure 3.9: Illustration of an acquired GPS signal from satellite 17.....	72
Figure 3.10: GPS time, Z-count and truncated Z-count	75

Figure 3.11: Kepler’s orbital parameters	76
Figure 3.12: GPS clock bias and pseudo-range illustration	79
Figure 3.13: Pseudo-ranges, SV, GPS receiver position and clock bias illustration	80
Figure 3.14: GPS receiver velocity calculation.....	81
Figure 3.15: Global TEC maps and corresponding ionospheric range errors	85
Figure 3.16: Atmosphere with focus on tropospheric effect	86
Figure 3.17: Depiction of multipath and its effects on received GPS signal.....	87
Figure 3.18: Illustration of Doppler frequency caused by satellite motion.....	88
Figure 3.19: Various GPS L1 C/A UERE budget analyses	90
Figure 3.20: RADCAL satellite Doppler shift.....	92
Figure 4.1: Illustration of initialising the GPS receiver software directory.....	97
Figure 4.2: Preview of the GPS software receiver settings initialisation.....	97
Figure 4.3: Illustration of the acquisition algorithm execution process	98
Figure 4.4: Illustration of the tracking algorithm execution process.....	99
Figure 4.5: Illustrative summary of the positioning process	100
Figure 4.6: Illustration of the GPS satellites position and pseudo-ranges algorithms execution	101
Figure 4.7: Illustration of the GPS receiver least-squares algorithm execution process	102
Figure 4.8: GPS receiver position computation summary.....	103
Figure 4.9: GPS receiver velocity calculation	104
Figure 4.10: Illustration of the GPS receiver UTC date and time calculations.....	104
Figure 4.11: Grand and simplified execution summary of the software-defined GPS receiver operation	105
Figure 4.12: The GPS receiver software-to-firmware conversion methodology.....	108
Figure 4.13: Illustration of Matlab functions and corresponding test-benches in project folder	110
Figure 4.14: Illustration of Step 4; sub-steps 1 to 3 – Define input types and simulate files	112
Figure 4.15: Illustration of Step 4; sub-steps 3 and 4 – Modify proposed fixed-point types and validate.....	113
Figure 4.16: Illustration of Step 4; sub-steps 4 and 5 – Error analysis and test numerics.....	114
Figure 4.17: Matlab floating-point to fixed-point conversion benefits and structure.....	115
Figure 4.18: Illustration of Step 5 – HDL code generation settings	117
Figure 4.19: Illustration of Step 5 – Generated equivalent Simulink model	118
Figure 4.20: Illustration of Step 5 – Generated VHDL codes for calcLoopCoef.m	119
Figure 4.21: Illustration of Step 5 – Generated VHDL codes for calcLoopCoef.m continues	120
Figure 4.22: Illustration of Step 5 – End of generated VHDL codes for calcLoopCoef.m	121
Figure 4.23: Matlab test bench stimulating VHDL simulation.....	123
Figure 4.24: MATLAB algorithm used in VHDL simulation	124
Figure 4.25: Bi-directional Simulink model and HDL simulation	125
Figure 4.26: Firmware GPS algorithm simulation and verification with HDL test-bench.....	126
Figure 4.27: Research methodology implementation summary.....	127
Figure 4.28: GPS receivers’ test and validation setup procedure.....	128

Figure 5.1: Three different plots of the recorded GPS signal (Multipath.bin dataset)	130
Figure 5.2: 3D plot of all the acquired GPS L1 C/A signals (with ± 5 kHz Doppler search bins).....	131
Figure 5.3: 3D plot of all the acquired GPS L1 C/A signals (with ± 10 kHz Doppler search bins).....	131
Figure 5.4: 3D plot of all the acquired GPS L1 C/A signals (with ± 50 kHz Doppler search bins).....	132
Figure 5.5: 3D plot of all the acquired GPS L1 C/A signals (with ± 100 kHz Doppler search bins).....	132
Figure 5.6: GPS satellites PRN (above threshold of 2) present in the acquired data at ± 100 kHz Doppler.....	133
Figure 5.7: Illustration of the GPS satellites acquisition process during processing.....	134
Figure 5.8: Summary of acquisition, tracking and navigation DOP results.....	135
Figure 5.9: UTC date and time, UTM, GPS receiver position and orbiting satellites sky plot.....	136
Figure 5.10: check_t.m variables definition and Matlab floating to fixed-point conversion results	138
Figure 5.11: check_t.m Matlab fixed-point algorithm verification and error analysis results	139
Figure 5.12: clsin.m variables definition and Matlab floating to fixed-point simulation results.....	140
Figure 5.13: clsin.m Matlab fixed-point algorithm verification and error analysis results.....	141
Figure 5.14: deg2dms.m variables definition and Matlab floating to fixed-point conversion results	142
Figure 5.15: deg2dms.m Matlab fixed-point algorithm verification and error analysis results	143
Figure 5.16: findUtmZone.m variables definition and Matlab floating to fixed-point conversion results.....	144
Figure 5.17: findUtmZone.m Matlab fixed-point algorithm verification and error analysis results	145
Figure 5.18: check_t.m Simulink floating and fixed-point models results comparison	146
Figure 5.19: clsin.m Simulink floating and fixed-point models results comparison	147
Figure 5.20: deg2dms.m Simulink floating and fixed-point models result comparison.....	148
Figure 5.21: findUtmZone.m Simulink floating and fixed-point models results comparison.....	148
Figure 5.22: check_t.m conversion to VHDL and the algorithm verification processes.....	149
Figure 5.23: Preview of check_t.m algorithm – converted to VHDL.....	150
Figure 5.24: deg2dms.m conversion to VHDL and the algorithm verification processes	151
Figure 5.25: Preview of deg2dms.m algorithm – converted to VHDL.....	152
Figure 5.26: findUtmZone.m conversion to VHDL and the algorithm verification processes	153
Figure 5.27: Preview of findUtmZone.m algorithm – converted to VHDL	154
Figure 5.28: clsin.m generation process to VHDL and verification failure	155
Figure 5.29: check_t firmware compilation result – ready now to be programmed into a FPGA.....	156
Figure 5.30: deg2dms firmware compilation result – ready now to be programmed into a FPGA.....	157
Figure 5.31: findUtmZone firmware compilation result – ready now to be programmed into a FPGA.....	157
Figure 5.32: Illustration of the used pre-captured GPS signal dataset (Multipath.bin) – date and time.....	159
Figure 7.1: GPS signal acquisition serial search technique	165
Figure 7.2: Depiction of parallel frequency space search technique.....	166
Figure 7.3: Portray of parallel code phase search algorithm	167
Figure 7.4: Space integration acousto-optic correlator principle.....	169
Figure 7.5: Acousto-optic GPS receiver acquisition system	169
Figure 7.6: Multiple correlators bank acquisition technique	170

Figure 7.7: Single arm of the XMC acquisition technique.....	170
Figure 7.8: Portray of matched filter acquisition technique.....	171
Figure 7.9: Illustration of the acquisition process.....	172
Figure 7.10: Illustration of a basic GPS receiver tracking loop.....	174
Figure 7.11: Illustration of a Costas PLL.....	175
Figure 7.12: Simple DLL_E-P-L code tracking principle.....	176
Figure 7.13: DLL_E-P-L code tracking process.....	176
Figure 7.14: Robust DLL_E-P-L code tracking principle.....	177
Figure 7.15: Combined carrier and robust DLL code tracking approach 1.....	178
Figure 7.16: Combined carrier and robust DLL code tracking approach 2.....	178
Figure 7.17: Basic illustration of a Kalman filter tracking loop concept.....	180
Figure 7.18: Illustration of a VTL assisting tracking loop.....	181
Figure 7.19: INS aided loop architecture illustration.....	182
Figure 8.1: Tracking block output – depicting recovered GPS navigation message.....	192
Figure 8.2: GPS receiver position computation using pseudo-ranges, SV position and clock bias.....	198
Figure 8.3: Transformation between ECEF to ENU coordinates.....	203
Figure 9.1: Configuration settings of the software GPS receiver before processing.....	209
Figure 9.2: Abridged acquisition results of the Matlab floating-point software GPS receiver.....	210
Figure 9.3: Channel 1 illustration of tracked GPS satellite #2, showing fully decoded navigation message.....	211
Figure 9.4: Channel 2 illustration of tracked GPS satellite #24, showing fully decoded navigation message.....	211
Figure 9.5: Channel 3 illustration of tracked GPS satellite #10, showing fully decoded navigation message.....	212
Figure 9.6: Channel 4 illustration of tracked GPS satellite #30, showing fully decoded navigation message.....	212
Figure 9.7: Channel 5 illustration of tracked GPS satellite #6, showing fully decoded navigation message.....	213
Figure 9.8: Channel 6 illustration of tracked GPS satellite #7, showing fully decoded navigation message.....	213
Figure 9.9: Channel 7 illustration of tracked GPS satellite #32, showing poor decoded navigation data.....	214
Figure 9.10: Channel 8 illustration of tracked GPS satellite #14, showing poor decoded navigation data.....	214
Figure 9.11: Channel 9 illustration of tracked GPS satellite #1, showing poor decoded navigation data.....	215
Figure 9.12: Channel 10 illustration of un-tracked GPS satellite #? : Navigation message decoding failed.....	215
Figure 9.13: Preview of the first four acquired GPS satellites orbital parameters.....	216
Figure 9.14: Illustration of some conversion issues in the acquisition.m algorithm.....	217
Figure 9.15: Illustration of some conversion issues in the calculatePseudoranges.m algorithm.....	217
Figure 9.16: Illustration of some conversion issues in the cart2geo.m algorithm.....	218
Figure 9.17: Illustration of some conversion issues in the cart2utm.m algorithm.....	218
Figure 9.18: Illustration of some conversion issues in the checkPhase.m algorithm.....	219
Figure 9.19: Illustration of some conversion issues in the clksin.m algorithm.....	219
Figure 9.20: Illustration of some conversion issues in the dms2mat.m algorithm.....	220
Figure 9.21: Illustration of some conversion issues in the e_r_corr.m algorithm.....	220
Figure 9.22: Illustration of some conversion issues in the ephemeris.m algorithm.....	221

Figure 9.23: Illustration of some conversion issues in the findPreambles.m algorithm	221
Figure 9.24: Illustration of some conversion issues in the generateCAcode.m algorithm.....	222
Figure 9.25: Illustration of some conversion issues in the gps_time.m algorithm	223
Figure 9.26: Illustration of some conversion issues in the gps2utc.m algorithm	223
Figure 9.27: Illustration of some conversion issues in the init_tracking.m algorithm	224
Figure 9.28: Illustration of some conversion issues in the invert.m algorithm	224
Figure 9.29: Illustration of some conversion issues in the leastSquarePos.m algorithm.....	225
Figure 9.30: Illustration of some conversion issues in the makeCaTable.m algorithm.....	225
Figure 9.31: Illustration of some conversion issues in the navPartyChk.m algorithm.....	226
Figure 9.32: Illustration of some conversion issues in the postNavigation.m algorithm	227
Figure 9.33: Illustration of some conversion issues in the satpos.m algorithm	227
Figure 9.34: Illustration of some conversion issues in the showChannelStatus.m algorithm.....	228
Figure 9.35: Illustration of some conversion issues in the togeod.m algorithm	228
Figure 9.36: Illustration of some conversion issues in the topocent.m algorithm	229
Figure 9.37: Illustration of some conversion issues in the tracking.m algorithm.....	230
Figure 9.38: Illustration of some conversion issues in the tropo.m algorithm.....	230
Figure 9.39: Illustration of some conversion issues in the twosComp2dec.m algorithm	231

List of Tables

Table 1.1: Summary of specifications and costs of space-borne GPS receivers.....	24
Table 1.2: Hardware GPS devices utilised for space-borne GPS receivers.....	29
Table 1.3: Various GPS baseband processors.....	29
Table 1.4: Summary of low and high-dynamics GPS signal acquisition techniques.....	30
Table 1.5: Summary of low and high-dynamics GPS signal tracking techniques.....	31
Table 2.1: Comparison and summary of GNSS technologies	43
Table 2.2: GPS C/A Gold codes phase designations.....	52
Table 2.3: Modulo-2 addition and equivalent BPSK truth table comparison.....	58
Table 3.1: Summary of suitable hardware GPS receiver front-end ICs.....	67
Table 3.2: Summary of potential software GPS receiver front-end devices	67
Table 3.3: GPS signal acquisition and tracking states	72
Table 3.4: GPS L1 C/A Acquisition Techniques	73
Table 3.5: GPS L1 C/A Tracking Techniques	73
Table 3.6: Miscellaneous Research on GPS L1 C/A Acquisition and Tracking Techniques	73
Table 3.7: GPS satellites ephemeris parameters definition	77
Table 3.8: GPS satellites ephemeris parameters value.....	77
Table 3.9: GPS orbital elements parameters definition.....	78
Table 3.10: GPS orbital elements parameters definition continued	78
Table 4.1: Summary of all the applicable Matlab floating-point GPS receiver algorithms	106
Table 5.1: Matlab floating-point GPS receiver software-to-firmware conversion results summary.....	137
Table 5.2: Summary of the GPS receiver algorithms output results comparison and error analysis	155
Table 8.1: Performance comparison of three common acquisition techniques	186
Table 8.2: Various types of DLL discriminators	191

CHAPTER 1

INTRODUCTION

1.1 Thesis Overview

The goal of the Global Navigation Satellite System (GNSS) technology, is to accurately assign every point ($<1\text{m}^2$) on Earth a unique address (coordinates – latitude, longitude and altitude), as well as to provide accurate timing information (El-Rabbany, 2002: 1). Advances in GNSS research, enabled the development of space-borne Global Positioning System (GPS) receivers, for use in nano-satellites in Low Earth Orbit (LEO). Nano-satellites (specifically CubeSats), are also designed and developed by staff and students at the French South African Institute of Technology (F'SATI¹) – a space technology research unit at the Cape Peninsula University of Technology (CPUT). CubeSats measuring $10\times 10\times 10\text{cm}^3$ and weighing $\sim 1\text{kg}$ are a popular affordable platform used by universities and industries worldwide. They boost short development cycles for satellites systems engineering, in-situ space research and modest resolution imaging.

A GPS receiver onboard a CubeSat serves various purposes, such as autonomous real-time orbit determination and in-situ ionospheric research. Contrary to terrestrial GPS receivers that are affordable, readily available and subjected to less challenging environments, a space-borne GPS receiver for use in CubeSats, must be designed to cope with challenging operational conditions, such as hazardous space weather and extreme dynamic Doppler's shifts in excess of $\pm 100\text{ kHz}$ (Gleason and Gebre-Egziabher, 2009: 330; Avansi and Tortora, 2010; Lofgren, n.d.). Faced with these challenges and high unit cost of $\sim \text{R}100\ 000$ of available space-borne GPS receiver units (Montenbruck 2008: 3 & 4), this research demonstrates the design and implementation of an efficient GPS receiver as well as postulates the implementation of a low-cost space-grade model.

1.2 Nano-satellites Overview

Nano-satellites are any miniature artificial satellites (irrespective of their shapes or form-factors) of mass between 1 and 10 kg. Nano-satellites are not linked to nanotechnology but are simply a descriptive attribute to the miniaturisation of satellite technology. The nano-satellite of interest to this research is the CubeSat and is therefore discussed in detailed in the subsequent sections.

¹ <http://www.cput.ac.za/fsati>

1.2.1 CubeSats Background

CubeSats are miniature nano-satellites (or spacecrafts) introduced in 1999 by Prof. Jordi Puig-Suari at California Polytechnic State University in San Luis Obispo and Prof. Bob Twiggs at Stanford University in Palo Alto (Paluszek *et al.*, 2010: 3; CubeSat, 2014: 5). The primary aim has been to enable higher institutions around the world to study and implement the features and functionalities of a larger satellite at a micro level, with minimum costs and shorter development time. As a result, commercial-off-the-shelf (COTS) components are mostly utilised in the developments of CubeSats to reduce costs at the expense of life expectancy. CubeSats design and launch cost varies and it is developed globally at more than 100 institutions (universities, colleges and industries) worldwide. In AMSAT-UK (2014) and CubeSat (2014), more than 100 CubeSats have been launched² with the first in Russia in 2003. Figure 1.1 depicts CubeSats.

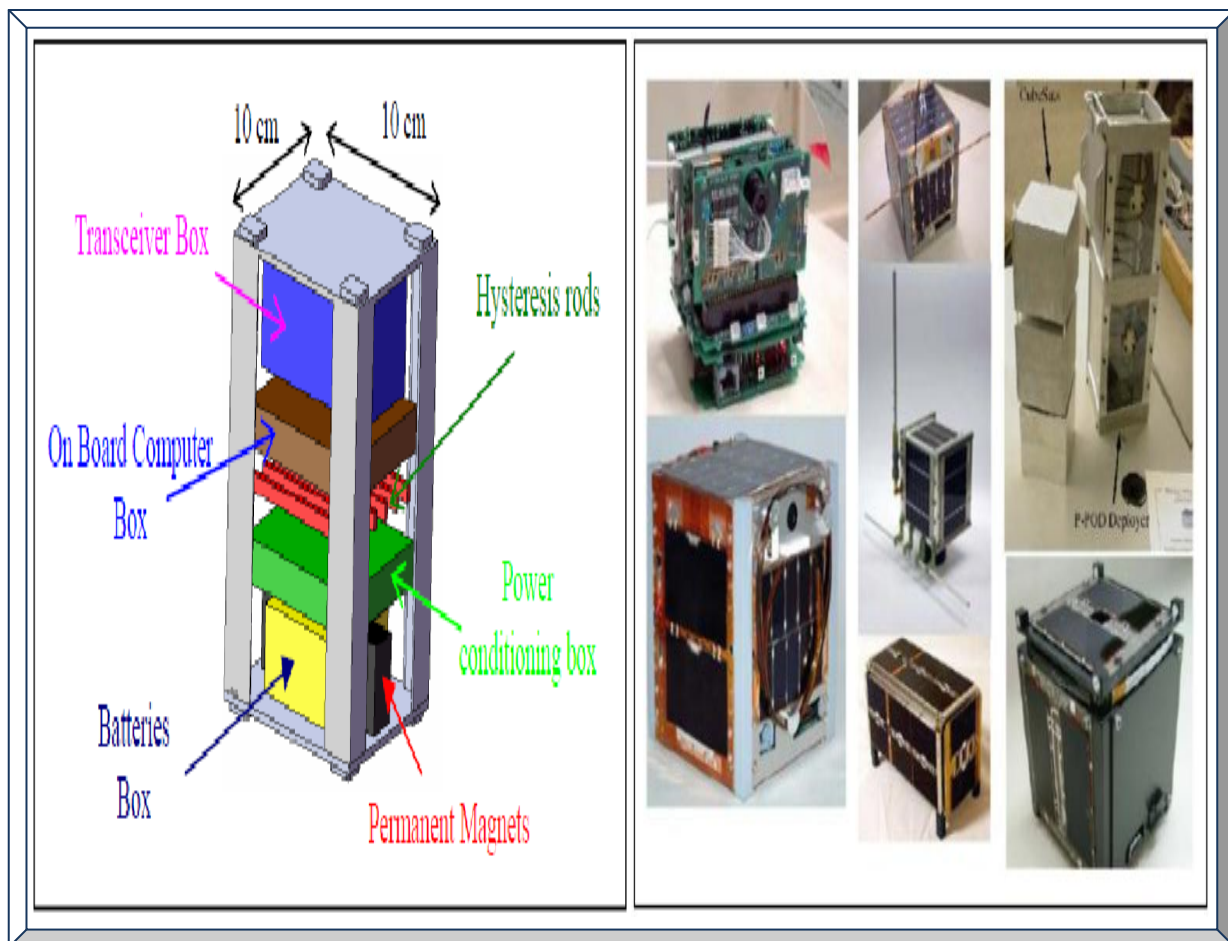


Figure 1.1: Model of a CubeSat (left) and various types of designed CubeSats (right)

(Adapted from Arif, 2010: 300; Sweeting, 2010: 56)

² <https://directory.eoportal.org/web/eoportal/satellite-missions/space-agencies>

1.2.2 CubeSats Specifications

CubeSats are cubical nano-satellites of mass between 1 and 10 kg. Their design and construction must adhere to strict standard dimensions and there are commonly three CubeSats standard form-factors, namely 1U, 2U and 3U – where U = Unit = 10cm (NASA, n.d.). It should be noted that, the width and height of all CubeSats are the same (i.e. 10cm by 10cm) and what differentiate them are their lengths and masses. Therefore, a 1U CubeSat has a dimension of 10cm x 10cm x 11cm and weighs approximately <1.33 kg (NASA, 2014.); whereas, a 3U CubeSat has a dimension of 10cm x 10cm x 30cm and weighs approximately < 4 kg (Nelson, 2010: 4 - 7).

The 1U and 3U form-factors of CubeSats are the most common and widely used (use also by F'SATI); as such, it will be used throughout in this thesis as a reference to illustrate CubeSats concept where applicable. These dimensions are standard and were established to enable a CubeSat to fit in its launch platform known as the Poly-Picosatellite Orbital Deployer (P-POD).

Adhering to this standard means that, multiple CubeSats can be easily and safely placed in a P-POD and cost-effectively launched as a piggyback (i.e. the P-POD is mounted on-top of a bigger spacecraft during launch and both combined units are launched as a single entity by the launch vehicle). Figure 1.2 illustrates a descriptive sketch of the 1U, 3U and P-POD form-factors as described in Bein (2008: 1 - 5) and California Polytechnic State University (2009: 5).

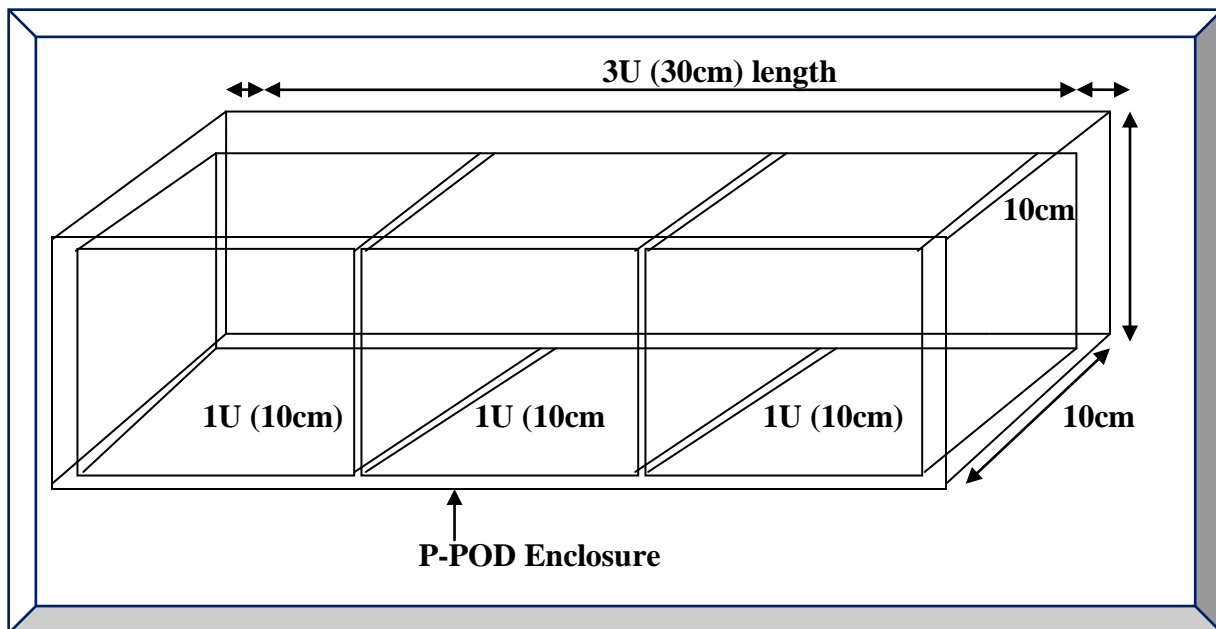


Figure 1.2: Illustration of three 1U CubeSats inside a P-POD

1.2.3 CubeSats Objectives

CubeSats were developed to assist universities and higher learning institutes worldwide to perform space science and exploration, by using CubeSat low-cost platform as an opportunity for students to learn and practice satellite engineering at a simpler micro level (Smalarz, 2011: 1 - 3).

Furthermore, human capacity development skills achieved from CubeSats trainings will enable graduate students to take-part in larger space missions at national and/or international levels.

Finally, CubeSats programme provides quick access to space by using small payloads; thus, it allows a research facility to quickly carry out an urgent space experiment (Bein, 2008: 1).

1.2.4 CubeSats Sub-systems

Arif (2010: 297 - 305) indicates that, a typical CubeSat consists of two sections: a payload and a platform – the payload defines the CubeSat mission and constitutes the sensor(s) while the platform sustains the mission and constitutes the sub-systems. Figure 1.3 illustrates a simplified architecture of a typical CubeSat system and is briefly explained in the following paragraphs.

A payload is the most vital part of a CubeSat, as it defines the mission objective (e.g. payload containing sensor for remote monitoring of Earth) for launching a satellite into space. Due to CubeSats constraints, a payload needs to be selected such that its power, size and mass parameters meet the CubeSat specifications. In some cases, a CubeSat contains a single payload with no redundancy and as a result, it has to be very reliable and fault proof (Munteanu, 2009: 9).

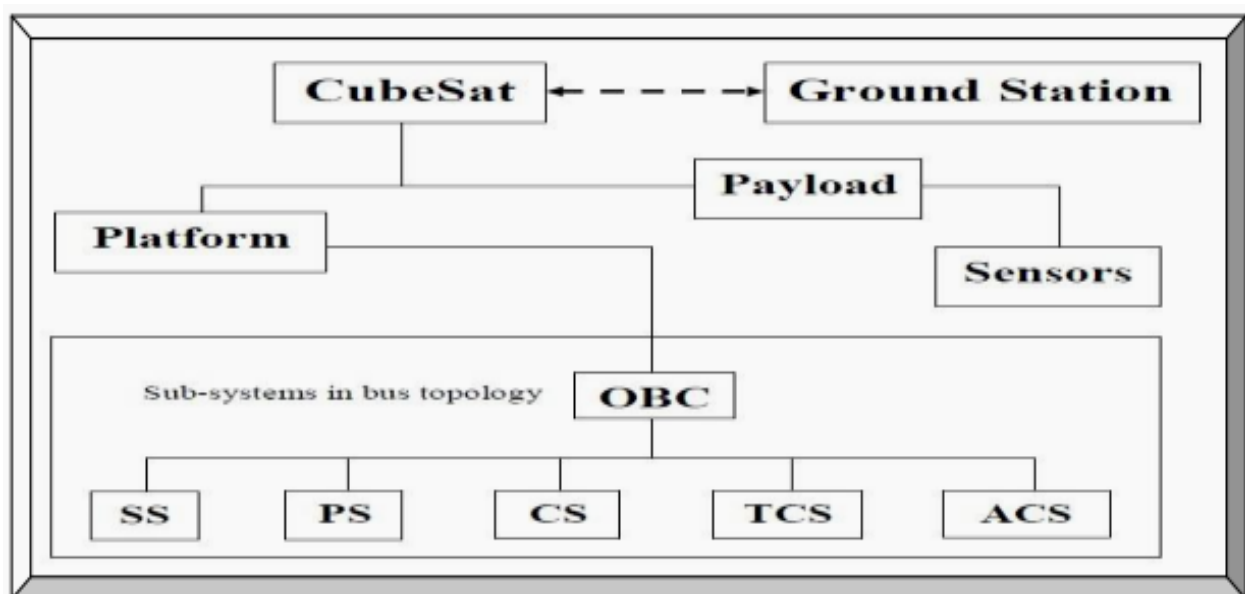


Figure 1.3: CubeSat architecture showing basic sub-systems

A platform offers structural supports for the payload, sustains the mission and encompasses the sub-systems – which includes the followings as highlighted in (Arif, 2010: 298 - 316).

- Structural Sub-system (SS): Provides mechanical support during launch and protects CubeSat components from space adverse conditions such as radiations and extreme temperatures.
- Power Sub-system (PS): Power is derived from the Sun and is harnessed by solar cells and stored in Lithium-ion batteries, further regulated and distributed to the various sub-systems.
- Communication Sub-system (CS): Provides 2-way communications between the ground stations and orbiting CubeSats. CS often constitutes the antenna, transponder and transceiver.
- Thermal Control Sub-system (TCS): Maintains the internal temperature of the CubeSat within functional limits, using active and passive thermal control means (e.g. via radiation process).
- Attitude Control Sub-system (ACS): Maintains orbiting CubeSats orientation with respect to Earth and Sun, for communication and power respectively. Spin stabilisations is often used.
- On-board Computer Sub-system (OBC): It is the central processing unit and it executes the flight software that is responsible for telemetry and tele-commands with the ground-station, as well as delegation of tasks and coordination of processes to and from the various sub-systems.

The mentioned sub-systems are the most fundamental found in CubeSats and they vary depending on mission aims. These sub-systems constitute modules and or discrete devices. Other less common sub-systems include propulsion or thrusters and in NASA Goddard (2014), it can be used to advance CubeSat to deep space missions by controlling the speed, orbit and altitude.

CubeSats require ground stations for communications and sharing of data, such as commands and payload data. Communications are usually done via private ground stations using the AX.25 protocol with the amateur radio services or via the global distributed network of ground stations such as GENSO and INTELSAT (Munteanu, 2009: 10 - 54; Smalarz, 2011: 26; Lee, n.d.: 1).

1.2.5 CubeSats Applications

In NASA Goddard (2013), CubeSats can be used in the following applications: in-situ space science research, environmental remote sensing of Sun, Earth observation (e.g. fires) or imaging, technology demonstration / verification, communication if constellated, testing components in real space conditions, helio-physics, astro-physics, geosciences, satellite servicing, space station inspection, space debris removal, cosmology, link between lightning and terrestrial gamma ray flashes, water detection in moon using infrared spectrometry and potential deep space missions. Koontz *et al.* (2014) states CubeSats can as well be used to measure Near-Earth Objects (NEOs).

1.2.6 CubeSats Past Missions

Since CubeSats inception, they have been some launch successes to space, as well as failures during launch (Bein, 2008: 4 & 5; Munteanu, 2009: 6 & 7). Amongst the success stories was the launch of CPUAT's and F'SATI first CubeSat – ZACUBE-01³, launched on November 21, 2013.

1.2.7 CubeSats Constellations

CubeSats constellations is an interconnection network of two or more orbiting CubeSats in space – configured on a specific orbital plane of Earth and inclined at a certain angle and altitude. Constellation improves CubeSat mission performance, coverage, continuous data exchange and mitigates mission failure if a CubeSat in a constellation fails. CubeSats are mostly constellated in Sun synchronous orbits, typically at 98° inclination angle from ~ 400 – 1000 km altitudes above Earth. Communications between constellated CubeSats is via RF and Laser links. In Sandau *et al.* (2010: 113 - 121), a constellation of small satellites is suitable for measuring near-Earth space conditions. CubeSats constellations can be simulated using STK tool (Munteanu, 2009: 60 - 75).

There are various theoretical types of CubeSats constellations and the two most common (see Figure 1.4) are the Polar (requires CubeSats to be longitudinally and inclined at 90°) and Walker (requires all CubeSats orbits to have the same criss-cross inclination at 98°) orbits constellations. These two were considered by researchers at University of Washington in their Rapid Terrestrial Imaging application, in which a minimum of LEO CubeSats were constellated to minimise focal length and telescope diameter, for adequate image resolution (Bernhardt *et al.*, 2009: 4 - 14).

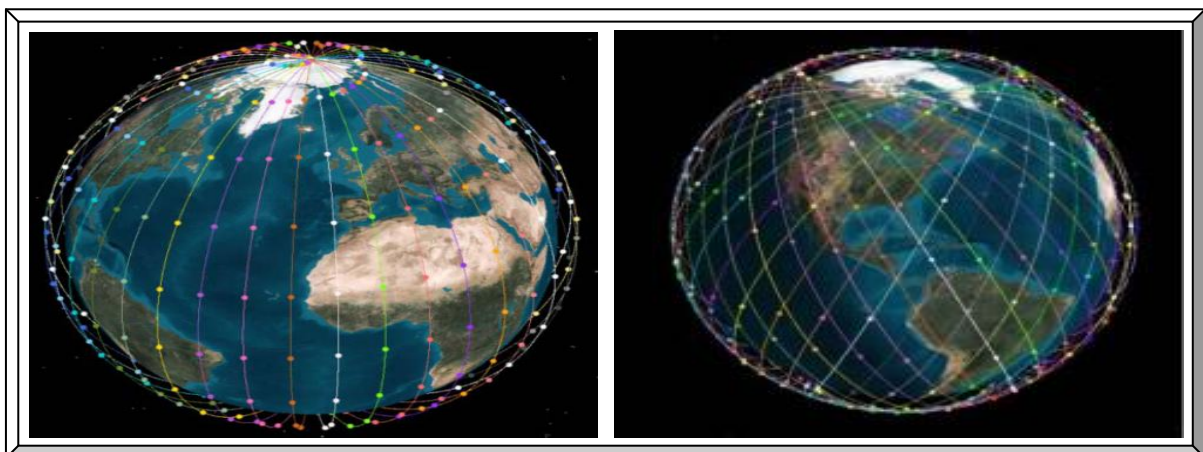


Figure 1.4: Polar (left) and Walker (right) orbits constellations
(Adapted from Bernhardt *et al.*, 2009: 4 & 5)

³ <https://directory.eoportal.org/web/eoportal/satellite-missions/v-w-x-y-z/zacube-1>

1.3 Research Problem Statement

This research is undertaken to address two challenges in Global Navigation Satellite Systems (GNSS) technologies. These are i) inefficient Global Positioning System (GPS) Link 1 (L1) Coarse/Acquisition (C/A) processing (acquisition, tracking and decoding) algorithms for use by GPS L1 C/A receivers on-board an orbiting CubeSat in Low Earth Orbit (LEO) and ii) the high-costs associated with space-borne or high-dynamics GPS receiver design and implementation.

1.4 Research Problem Background

F'SATI is a space research unit at CPUT that focuses on the design and development of LEO CubeSats. F'SATI's first CubeSat (ZACUBE-01 or TshepisoSat or ZA-003) was built and developed by staff and students and was successfully launched on November, 21 2013. The design and construction of its first 3U CubeSat (ZACUBE-02) is in progress. ZACUBE-01 relies on the North American Aerospace Defense Command (NORAD) Two-Line Elements (TLEs) data for position and velocity determinations. As mentioned in Kahr *et al.* (2010), TLEs is a set of orbital parameters, used to provide navigation to any satellites or objects orbiting the Earth.

Although the TLEs approach is free (Celestrak, 2013), its usage is however not autonomous and requires the ground station(s) to continuously access the NORAD or Space-track website to update the orbiting satellite(s) with its new TLEs parameters during over-pass communication with the ground station(s). A real-time on-board GPS receiver will be a viable autonomous alternative or supplement to the TLEs approach. However, most GPS receivers that are commercially available cannot be publicly upgraded and therefore be used in space due to:

- Their inability to handle high-dynamics Doppler frequency shifts – which hampers GPS signal acquisition, tracking, lock, position, velocity and time accuracies (see Section 1.4.4). The inability in the GPS algorithms can be attributed to inefficiency due to (i) wrong mathematical models used and (ii) the computational inability of the implemented algorithmic models. The mathematical models refer to the different mathematical techniques or concepts that an algorithm is based-on. The computational inability refers to the slowness in execution speed and high memory usage of the implemented algorithms.
- GPS ITAR / COCOM limits embedded in commercial GPS receivers – ITAR stands for International Traffic in Arms Regulations. COCOM – discontinued, was a Coordinating Committee for Multilateral Export Controls. The GPS ITAR or previously COCOM limits, restrict operations of commercial GPS receivers to altitudes of <18km and velocities of <515m/s, (Lu, 2003: 63; Nortier, 2003: 20; CCII, n.d.: 2) by ensuring automatic shut-down or faulty readings if one or both limits are exceeded. This is to prevent free use of commercial GPS receivers in ballistic missiles applications. Therefore, only a handful of vendors (refer to Table 1.1), do design and commercialise space-borne GPS receiver units.

Table 1.1: Summary of specifications and costs of space-borne GPS receivers

Vendors/Dealers	SSTL (SGR-10)	SSTL (SGR-05U)	CubeSat Shop	NovAtel OEM615-D2S
Unit Price ®	R1,055,700	R150,000	R155,000	R95,000
Dimensions (cm³; g)	19.5x16.2x4.8; 950	7x4.5x1; 20	5x2x0.5; 30	6x3x2; 25
Power Rating (W; V)	5.5; 18 to 38	1; 5	1; 5	<1; +4.5 to 18
Operating temp (°C)	-20 to +50	0 to +50	-10 to +50	-40 to +85
Distinct feature	24 channels	12 channels	SDR	GPS / Glonass
Position accuracy	10m	10m	< 10m	10m
Velocity accuracy	15cm/s	15cm/s	< 25cm/s	20cm/s
Radiation dose	10kRad (Si)	10kRad (Si)	Tolerant	> 10kRad (Si)
Interface	CAN, RS422	RS232	I2C, RS422/485	CAN b, Serial
Lead time	6 months	3 months	3 months	3 months

Montenbruck (2008: 3 & 4), points out that, GPS receivers for space applications are subjected to challenges with respect to (i) space design considerations and (ii) space design cost factors:

i) Space design considerations include:

- Legal issues (ITAR and export laws).
- Signal high-dynamics (excessive Doppler's frequency shift).
- Extreme high and low temperatures.
- Vacuum.
 - Outgassing / leakage.
 - Decreased heat transfer.
- Vibration (only during launch).
- Radiation.
- Total ionization dose.
- Single event effects.

ii) Space design cost factors include:

- Small market segment.
 - ~ 300 satellites in LEO.
 - ~ 5 years life expectancy.
 - 5 – 10 geodetic annual grade missions.
- Space-grade components.
 - ~ R1800k regular receiver cost in addition to numerous R1800k non-regular cost.
- Acceptance tests and qualification.
 - R90 – 900k facility usage and R270 – 450k monthly labour cost.
- Software.
 - Reliability and performance requirements.
 - Software engineering standards.
 - Testing.

Hence, these challenges foster the research of F'SATI's space-borne GPS L1 C/A receiver.

To address these challenges, literature surveys on space-borne GPS receivers and GPS receiver's algorithms summarised in Sections 1.5.1 to 1.5.3 and elaborated in Appendix A were conducted.

1.5 Research Literature Review

The literature survey spans a detailed analyses of GPS satellites signal, correlator and baseband processing algorithms; orbital mechanics (coordinate systems and Keplerian orbit); GPS Digital Signal Processing (DSP); GPS errors sources such as multipath, ionospheric and Doppler's effect; as well as suitable GPS receiver development platforms such as Matlab and Simulink.

Presented below are brief descriptions of aspects of GPS technology relevant to this research.

GPS satellites refer to the space segment of GPS technology. Although the exact number varies, in Maini and Agrawal (2011: 516 & 517), it is indicated that, it consists of a constellation of 24 Medium Earth Orbit (MEO) satellites with four spares. The 24 satellites comprise six orbital planes with each plane constituting of four satellites. All planes are equally spaced 60° apart and all four satellites in each plane are unequally spaced apart as shown in Figures 1.5 and 2.3. Each orbital plane is inclined at an angle of 55° from the equatorial plane and each satellite in its plane is at an altitude of $\sim 20,200$ km above Earth with an orbital period of ~ 12 hours; repeating the same ground track twice every 1 sidereal day, at a velocity of ~ 3.888 km/s (approximately half LEO nano-satellite velocity – see Section 1.4.4). This GPS constellation ensures that, at any point in time, at least 5 to 8 satellites are visible from anywhere on Earth and a minimum of four are needed for accurate positioning (Tsui, 2005: 31), especially if the altitude is varying.

GPS signal refers to the invisible part and the transmission bands include L1, L2 and L5. The L2 band is reserved for USA military use and operates at a P(Y)-code signal frequency of 1.2276 GHz. This research will focus on the L1 band which is reserved for civil applications and has a C/A carrier signal frequency of 1.57542 GHz with a clock or data rate of 1023 kilo chips/s. The L1 band is divided into 25 frames that are further sub-divided into five sub-frames consisting of 300 chips (bits) per sub-frame and 1500 chips/per frame (Navstar IS-GPS-200H, 2013: 75 - 86). Hence, with a navigation data rate of 50 bps; the transmission rate will be 20ms per bit, 6s per sub-frame and 30s/frame. These GPS signals are transmitted as Pseudo Random Noise (PRN) Gold codes, spread by Direct Sequence Spread Spectrum (DSSS) and Code Division Multiple Access (CDMA) techniques. Modulation is via Binary (Bi) Phase Shift Keying (BPSK) scheme. PRN codes reach Earth as very weak signals embedded in noise and correlation is the only way of retrieval. Trilateration is commonly performed to compute pseudo-range (Gleason & Gebre-Egziabher, 2009: 56) from which; Position, Velocity and Time (PVT) estimates are deduced.

GPS receivers refer to the portable segment and are covered in detailed in Sections 1.5.1 to 1.5.3.

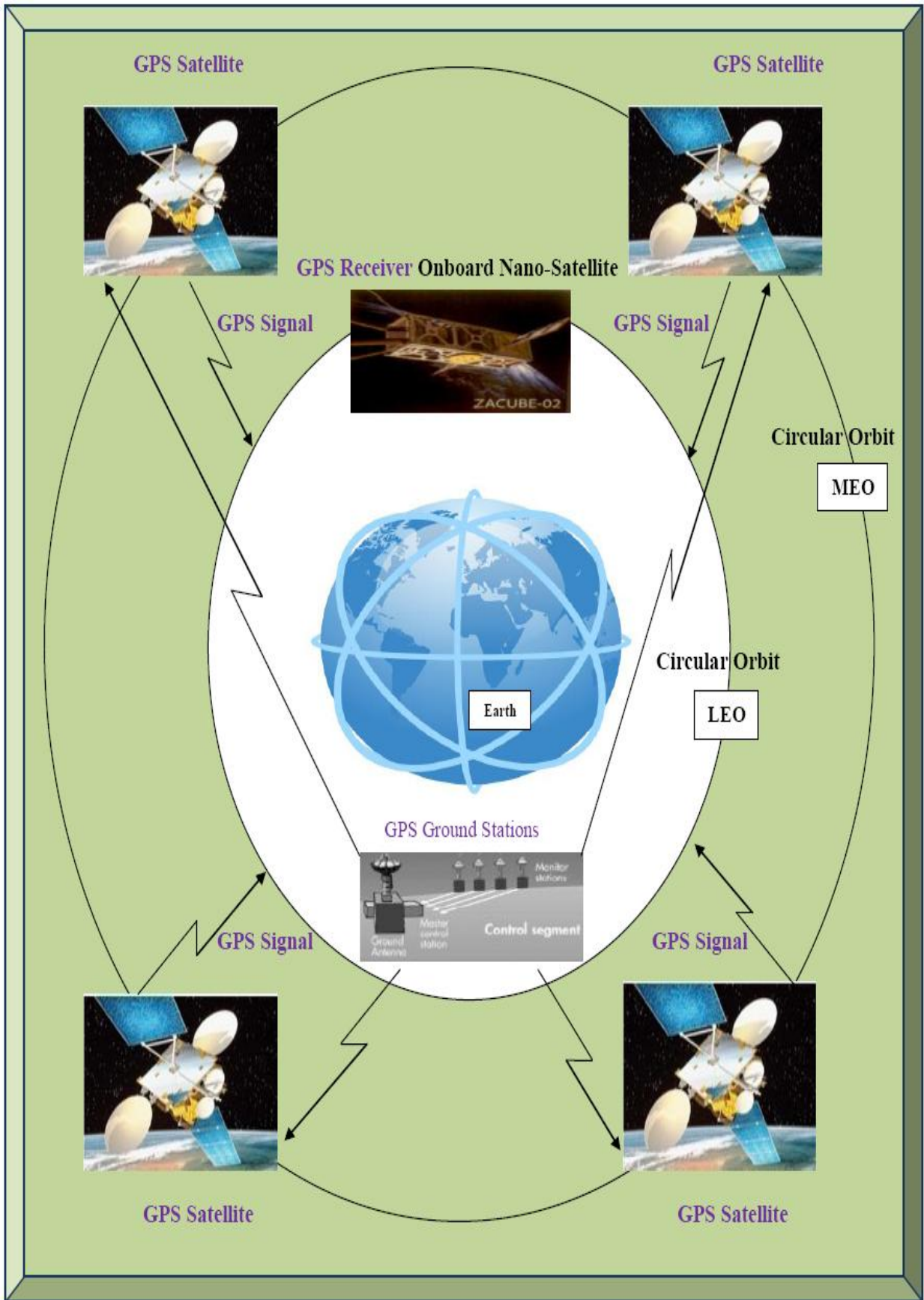


Figure 1.5: A single orbital plane overview representation of GPS technology (not to scale)

Research projects focusing on the design and implementation of space-borne receivers, some of which are highlighted in Table 1.1, have been the focus of a number of institutions (refer to Table 1.2). Three approaches used were examined and discussed as follows:

- Hardware-defined approach by using Commercial Off The Shelf (COTS) devices.
- Hardware-defined approach by modifying potential commercial GPS receiver units.
- Software-defined / Firmware-defined approaches (designing from first principles using DSP).

1.5.1 **Hardware-defined Approach by Using COTS Devices**

With this approach, discrete COTS components and open-source codes are utilised to design a space-borne GPS receiver, in which, a combination of Application Specific Integrated Circuit (ASIC) and DSP devices are used. These ASIC devices include Radio Frequency (RF) components such as GPS front-end and GPS correlator. The DSP devices could be any suitable programmable hardware (GPS baseband processor) that performs GPS data/maths computations.

Table 1.2 summarises satellite missions where space-borne GPS receivers were utilised and also lists the type of components that were used and the missions aim. From the data in Table 1.2, it can be seen that most of the GPS front-end and correlator devices in the space-borne GPS receivers, were implemented using Zarlink GP2015 and GP2021. GPS correlators design and baseband processing are the core of a GPS receiver; thus, features of the devices presented in Tables 1.2 and 1.3 were examined. The GP4020 or GP2000 seems suitable but have been discontinued and no longer supported by Zarlink, though they can still be sourced but unreliable.

1.5.2 **Hardware-defined Approach by Modifying Commercial GPS Receiver**

In Nortier (2003: 54), this approach used to be the popular and quickest way to design a space-borne GPS receiver. However, it carries some legal implications if done without the GPS receiver vendor's approval. Vendors such as NovAtel, Orion, Sigtec Navigation and Microsemi (acquired Zarlink⁴ formerly Mitel⁵) of commercial GPS receiver products, allow their GPS receivers to be modified for space applications, provided the client is willing to pay an extra fee. In the modification process, appropriate hardware and software changes including elevation mask are made and the lines of source-codes that implement the GPS ITAR or COCOM velocity (speed) and or altitude (height) restrictions (pre-programmed in the GPS baseband processor) are removed (Lu, 2003: 63 & 64). The amended source-codes are debugged and upgraded on the baseband processor and the implemented changes executed (Nortier, 2003: 76 & 77).

⁴ <http://ulp.zarlink.com/zarlink/hs/about.htm>

⁵ <http://en.wikipedia.org/wiki/Zarlink>

1.5.3 Software-defined / Firmware-defined Approaches

A software-defined GPS receiver unit refers to a unit that is implemented on a computer. This approach provides a versatile/re-programmable design and development platform (Borre *et al.*, 2007). With this method, the main GPS receiver functional blocks, namely the GPS correlator and baseband processor, are implemented based-on DSP or from emulating the internal circuitry of a known GPS receiver (Greenberg, 2005). Once the software implementation is realised, it is further fully debugged and optimised (Tsui, 2005: 3 - 5). Firmware is software embedded into hardware, such as a Field Programmable Gate Array (FPGA). Figure 1.6 depicts this approach.

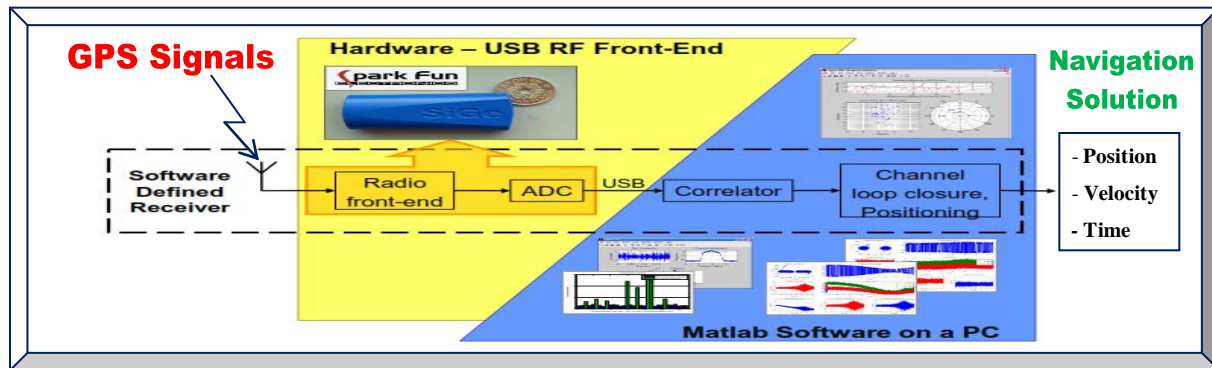


Figure 1.6: Software-defined GPS receiver architecture
(Adapted from Plausinaitis, 2009c: 17)

The first functional block of any GPS receiver is the GPS front-end and its purpose is to down-convert the received GPS signal. Transmitted GPS signals are Right Hand Circularly Polarised (RHCP) and thus require a RHCP antenna to receive the signal. Although a GPS front-end can be built using discrete components, it is readily available as ASIC with in-built Low Noise Amplifier (LNA), Analog to Digital Converters (ADC), filters and mixers (Grewal *et al.*, 2001: 81 & 82). Thus, just an antenna, oscillator, saw-filters and few passive components are required.

The second functional block is the GPS correlator that is used to acquire and track the down-converted GPS signal – by correlating incoming GPS signal with known replicas. As depicted in Table 1.2, GPS correlators are available as ASIC and DSP devices, consisting of 12 channels. The more GPS receiver correlator channels, the better the performance (McNamara, 2004: 54).

The third and final functional block is the GPS baseband processor and its purpose is to decode and process the GPS signals from the GPS correlator. From Tables 1.2 and 1.3, GPS baseband processors designs vary amongst vendors. Some GPS baseband processors have built-in GPS front-end and correlators housed in one package, others contain only GPS correlators, while most have just the GPS baseband processor for executing the GPS signal processing algorithms.

Table 1.2: Hardware GPS devices utilised for space-borne GPS receivers

Institutions	Mission	GPS Front-end	GPS Correlator	GPS Baseband	Research Payload or aim	Year
Surrey Space	Surrey Satellites	Mitel GP2000 (GP2015, GP2021, ARM60B processor)			Orbit Determination	-
Various institutes in Korea	-	-	12 Channel correlator	DSP processor	Tracking and Attitude Control	-
Standford / NASA	Space/Earth Science	Zarlink GP2010	Zarlink 2021	Mitel GPS chip	Spacecraft Formation-flying	1999
Tsinghua / Surrey	Tsinghua-1 MicroSat	Mitel GP2010	Mitel GP2021	ARM60	Position and Velocity	1999
GSOC	CHAMP, BIRD	Mitel GP2000 has built-in: (GP2015, GP2021, ARM60B processor)			Sounding Rockets and Sat	2001
Virginia Polytech	Ionosphere Research	Zarlink GP2015	Zarlink P2021	Mitel GPS Builder	GPS Signal Sensor	2003
New South Wales, Sydney	Colony II CubeSat	Zarlink GP2015	Zarlink P2021	Cyclone II FPGA	Proof of Concept	2004
GSOC/SUN	Sunsat 2004	Phoenix (SigTec MG5001)			Orbit Determination	2004
New South Wales	-	Zarlink GP2015	GP2021 FPGA	FPGA processor	Open-source GNSS Receiver	2004
Portland State Uni	Unmanned Satellites	GP2010	GP4020		Open-source GPS	2005
GSOC	X-sat, PRISMA	GP2015	Phoenix-XNS/HD Rx (Zarlink GP4020)		Orbits Determination	2006
New Brunswick	CASSIOPE	NovAtel OEM4-G2L GPS receiver			Multi-purpose	2007
National Cheng Kung	CKUTEX	Zarlink GP2015	Zarlink GP2021	ARM920T processor	Experimental Payload for LEO Mission	2010

Summarised in Table 1.3 are GPS receiver’s baseband processors with their respective vendors.

Table 1.3: Various GPS baseband processors

GPS baseband processors	Zarlink	Mitel	Atmel	Sony	Prolific	uNav	Nemerix
	GP4020	GP2000	ATR0622P	CXD2932	PL-6313	uN8130	NJ2020

Each of the three mentioned methods of designing a space-borne GPS receiver, has advantages and disadvantages. The hardware approach is more suitable in high-dynamics applications (in spacecraft and satellites) because of its continuous GPS satellites tracking architectures. The mostly used GPS baseband processors (GP2000 / GP4020) are now obsolete and non-upgradeable, since production has stopped (Zarlink Semiconductor, n.d.). If sourced and used in new projects, the GP2000 / GP4020 cannot be supported or improved to suit advances in GNSS technology, contrary to software-to-firmware approach which is thus chosen as the research path.

1.5.4 GPS Receivers Acquisition and Tracking Algorithms

To utilise an in-coming GPS signal from any orbiting GPS satellite, a GPS receiver must first acquire and then track it. Acquisition is a highly computational demanding task, as it involved searching within a three dimensional space of Doppler frequency shifts, GPS satellites PRN or Gold codes and unknown time delays. Once acquisition is complete, a fine closed-loop synchronisation known as tracking takes over and tries to keep the signal locked. In LEO, these processes becomes more challenging as they involved GPS satellites and CubeSats in respective MEO and LEO, at corresponding velocities of ~ 3.888 km/s and ~ 7.888 km/s (Nortier, 2003: 23).

As stated in Nortier (2003: 22 & 23), Gleason and Gebre-Egziabher (2009: 330) and Avanzi and Tortora (2010); Doppler shifts results to tremendous difficulty in GPS signal acquisition, tracking instability / cycle slips (GPS receiver loses lock) and finally PVT computation inaccuracies due to high relative motion and high-dynamics which i) enormously limit contact time between satellites in MEO and LEO – CubeSats will be rapidly changing hemispheres relative to respective GPS satellites and ii) increased Doppler shift rates and mean Doppler shifts as high as ± 55 kHz (~ 5 to ~ 11 times that of terrestrial GPS receivers). The quality of a GPS receiver algorithm is determined by its ability to acquire GPS signals as quickly as possible (especially in unfavourable conditions, such as high Doppler shift, interference and multipath), to track and stay locked (without cycle slips) to the acquired GPS signal and to compute PVT solutions as quickly and accurately as possible (Glennon *et al.*, 2011). In light of this, various low and high-dynamics GPS receiver algorithms have been proposed and implemented as possible solutions. Tables 1.4 and 1.5 respectively outline the acquisition and tracking techniques that are examined.

Table 1.4: Summary of low and high-dynamics GPS signal acquisition techniques

Acquisition Algorithms	Source
Acousto-optic Technique	Bazzi <i>et al.</i> (1993)
Mitel GP2000	Mitel GP2000 (1998: 14 & 15)
Averaging Correlator Acquisition	Alaqeeli <i>et al.</i> (2003)
Parallel Code Phase Search (Delay & Multiply FFT)	Tsui (2005: 139); Borre <i>et al.</i> (2007: 82)
Block (Non-coherent Integration) Acquisition	Tsui (2005: 144); Shi <i>et al.</i> (2009)
Multiple Correlators Bank Acquisition	Akopian <i>et al.</i> (2006)
Lifting Wavelet Acquisition	Djebouri <i>et al.</i> (2006)
Serial or Sequential Search Acquisition	Borre <i>et al.</i> (2007: 75 – 78)
Parallel Frequency Space Search Acquisition	Borre <i>et al.</i> (2007: 78 – 81)
Wavelet De-noising Acquisition	Tian & Yang (2008)
Reconfigurable Instruction Cell Array Acquisition	El-Rayis <i>et al.</i> (2010)
Wavelet Transform Acquisition	Jianguo <i>et al.</i> (2010)
Extended Multiple Correlators Acquisition	Xiaowen <i>et al.</i> (2010)
Bayesian Acquisition Technique	O'Mahony <i>et al.</i> (2012)
Matched Filter Correlator Acquisition	Ta <i>et al.</i> (2012)

Table 1.5: Summary of low and high-dynamics GPS signal tracking techniques

Tracking Algorithms	Source
Block Adjustment of Synchronising Signals	Tsui (2005: 170 – 175)
Combined Carrier and Robust DLL Code Tracking	Tsui (2005: 168); Borre <i>et al.</i> (2007: 106)
PLL and Costas Loop Carrier Tracking	Borre <i>et al.</i> (2007: 93 – 96)
Basic DLL Early-Late Code Tracking	Borre <i>et al.</i> (2007: 96 – 100)
Robust DLL Early-Late Code Tracking	Borre <i>et al.</i> (2007: 98)
Nonlinear Code Tracking Filter Technique	Gustafon <i>et al.</i> (2009)
Inertial Navigation System Tracking Technique	Rogers (2007); Xiaoyong & Baiqi (2011)
Combination of Squared Correlators Tracking	Falletti <i>et al.</i> (2012)
Extended Kalman Filter Tracking Technique	Peng <i>et al.</i> (2012)
Kalman Filter Tracking Loop Technique	Song <i>et al.</i> (2012)

In Gerein and Brown (2001), Tsui (2005) and Borre and Strang (2012), a software-defined GPS receiver approach presents a suitable platform to develop GPS receivers’ algorithms, as well as to devise new and advanced techniques, as existing GPS receiver algorithms have pros and cons.

For instance, in Borre *et al.* (2007: 75 - 78), a serial search technique is a very popular “method of signal acquisition in CDMA systems” – to which GPS belongs; however, it is time consuming as it involves the algorithms to search through all 1023 PRN code spaces and 41 Doppler’s frequency shift bins. In high-dynamics situations, this algorithm can be problematic, as the GPS satellites and GPS receiver will surely lose communication before processing is even completed.

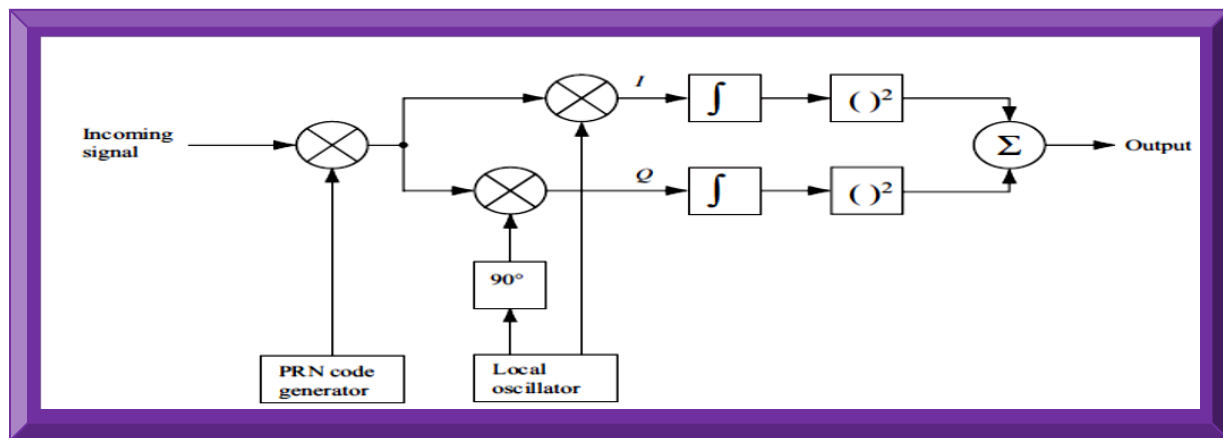


Figure 1.7: GPS signal acquisition serial search technique
(Adapted from Borre *et al.*, 2007: 76)

Figure 1.7 illustrates GPS signal serial search acquisition technique. It basically involves first generating respective GPS PRN codes, followed by an In-phase (I) and Quadrature (Q) 90° phase shifted carrier frequencies; which are then integrated, squared and finally summed together. The output is passed to the tracking algorithm section, to ensure the acquired signals are maintained.

From the research problem background and literature review, the following objectives were set.

1.6 Research Project Objectives

The main objective is to design and implement a software-defined, high-dynamics GPS receiver to acquire, track, decode and process GPS L1 C/A signals, with main focus on GPS receiver Position (latitude, longitude and altitude) calculation, as well as the Velocity and Time (PVT).

From this primary objective, the secondary and tertiary sub-objectives are derived as follows:

1.6.1 Secondary Objective: Develop Software Receiver / Address First Research Problem

The secondary objectives are to improve the proposed software-defined GPS receiver and to use it as a space-borne GPS receiver development platform / tool; to i) address the first research problem of inefficient GPS receiver algorithms and to ii) realise an improved GPS receiver algorithm(s) to acquire, track and decode incoming high-dynamics GPS L1 C/A signal in LEO.

1.6.2 Tertiary Objective: Firmware GPS Receiver Design and Implementation

The tertiary objectives are i) to convert (change to a high level embedded language e.g. VHDL) the improved software GPS receiver algorithm(s) to a firmware GPS receiver algorithm(s), which can readily be programmed into a suitable FPGA microcontroller and ii) to finally address the second research problem of high-cost associated with space-borne GPS receivers units.

Illustrated in Figure 1.8 is a brief flowchart and summary of the research project objectives.

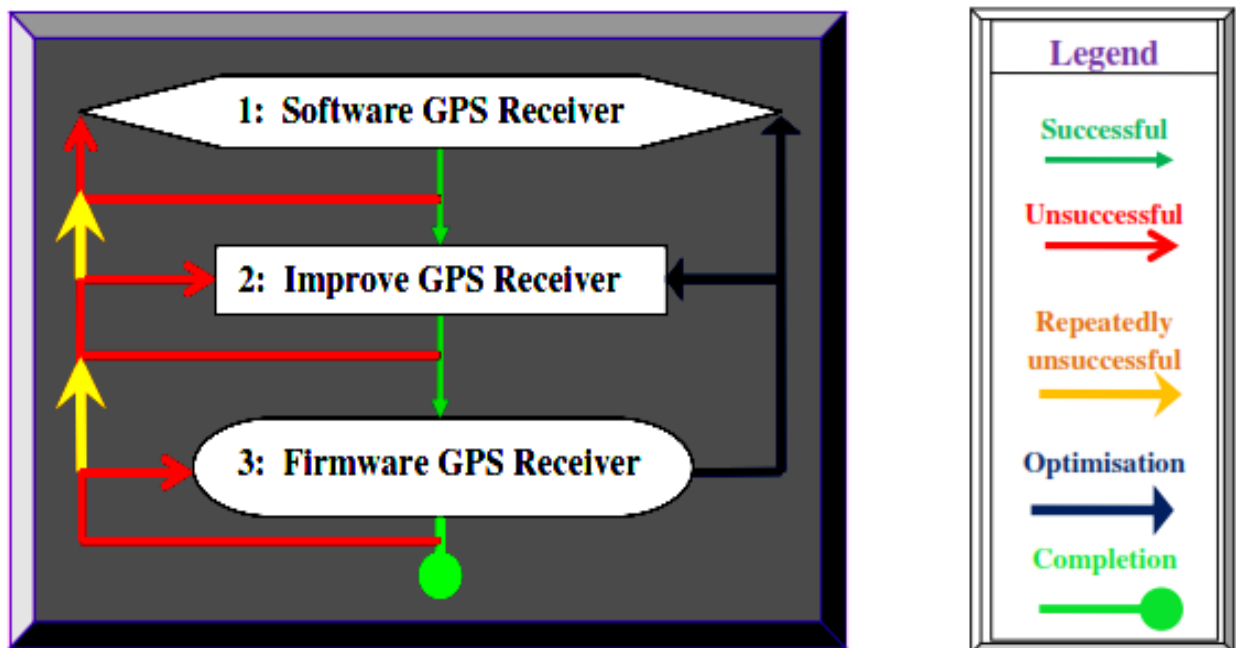


Figure 1.8: Concise summary of the research project objectives

1.7 Research Project Design Methodology

The literature review outlined in Section 1.5 set the basis for the design methodology. From the relevant literature assessment, the software approach is chosen as the preferred design and implementation path. A software approach is more flexible and enables rapid developments and devise of new algorithms. The algorithms summarised in Tables 1.4 and 1.5 were investigated to select the most suitable in-line with the research objectives, as well as a preferred software path.

Being a complex project, Software or System Development Life Cycle (SDLC) methodologies were investigated to select a suitable model to efficiently design and implement the project. From investigating various SDLC models, the Rapid Applications Development (RAD) model in Figure 1.9 was chosen as the preferred approach. In this model, the project requirements are set, the design to implementation phases split into sections and time-framed (see Figure 1.9) for discrete incremental executions, with the requirements revisited until the outcome is satisfactory.

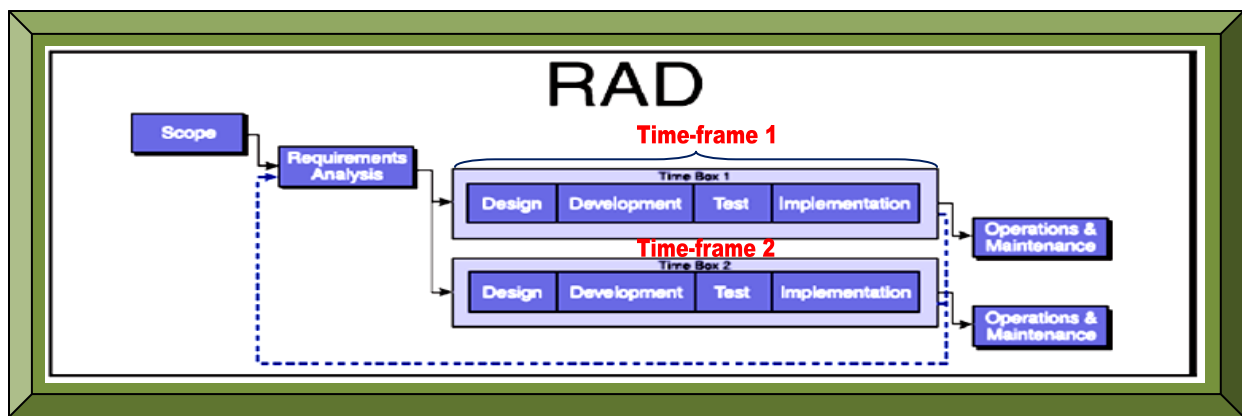


Figure 1.9: RAD SDCL methodology

(Adapted from PM Solutions Technology, 2003: 5)

Scope:

The research project objectives in Section 1.6 define the scope of the research project.

Requirements Analysis:

The software-defined GPS receiver has no formal requirements since it will mostly serve as the development platform. However, the improved software-defined GPS receiver algorithms, as well as the firmware-defined GPS receiver algorithms should have the features stated in Chapter 4. The specifications were identified from the research objectives and are defined as follows:

- Process high-dynamics GPS C/A L1 signals with acceptable accuracy within 10m.
- Low-cost model(s).

Design:

Here, the requirements are theoretically examined, mathematically modelled and systematically executed. Doing so eliminates errors, isolates unknowns and eases the design to debug processes.

Development:

The GPS receiver architecture in Figure 1.10 is the model under research that will be designed, developed and implemented. Hamsa *et al.* (2009) mentioned that the design, development and implementation of the GPS receiver software/firmware using Matlab/Simulink, makes the DSP coding in each part of the GPS receiver architecture very clear, easier to understand, modify and debug. In Dubey (2013), developing algorithms with MathWorks tools improves the efficiency.

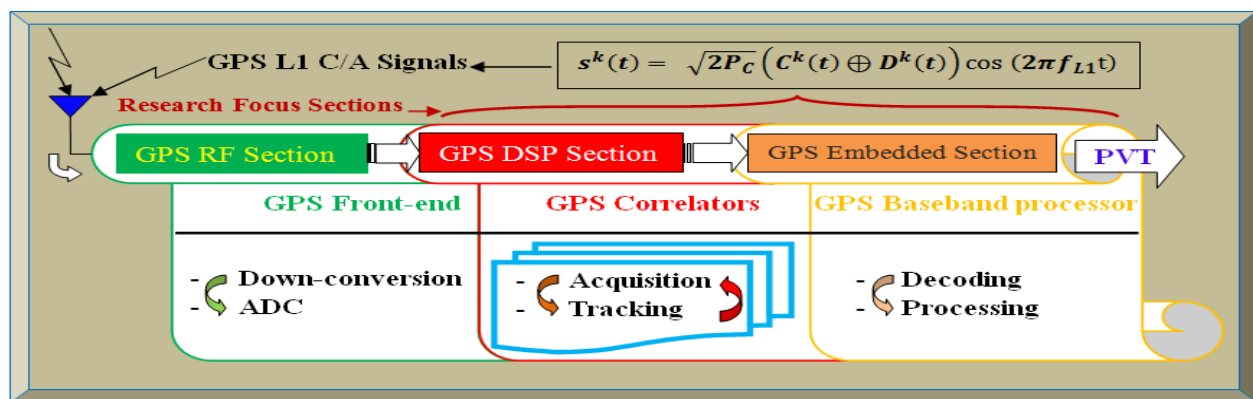


Figure 1.10: Internal functional blocks and fundamental tasks of the GPS receiver under research

This research project constitutes three main sections: the RF, DSP and software / embedded sections. The RF section will not be researched in the project, only the DSP and software / firmware (embedded) sections are researched and developed. The computing platform used is Matlab. The main processes involved in the software GPS receiver development are as follows:

- i) Implement a software-defined GPS receiver using open-source Matlab GPS algorithms.
- ii) Improve these GPS receiver algorithms using Matlab and associated toolboxes.
- iii) Automatically convert the improved Matlab software GPS algorithms to firmware.

The main steps involved in the GPS receiver algorithms development process are as follows:

- Step 1: Create and copy all GPS Matlab functions, scripts and test vectors to a local folder.
- Step 2: Simulate and verify each floating-point Matlab GPS algorithm using its test-bench.
- Step 3: Create a new GPS VHDL coder project to associates all the design files together.
- Step 4: Convert each of the GPS Matlab algorithm(s) from floating-point to fixed-point.
- Step 5: Generate GPS HDL codes (convert fixed-point Matlab GPS algorithms to VHDL).

Test:

Basically, two main series of simulated tests are performed in the research and these are i) first a low-dynamics test (terrestrial tests to check functionalities on Earth) and ii) a high-dynamics test (space-borne tests to check functionalities in LEO). Before integration, both tests types are done at unit level and then a suitable test path is examined to avoid unnecessary redundant tests. Figure 1.11 represents a pathway of various possible analyses tests tools that can be utilised. As stated in DSP-FPGA (2006), the Xilinx test path has been demonstrated before and is feasible.

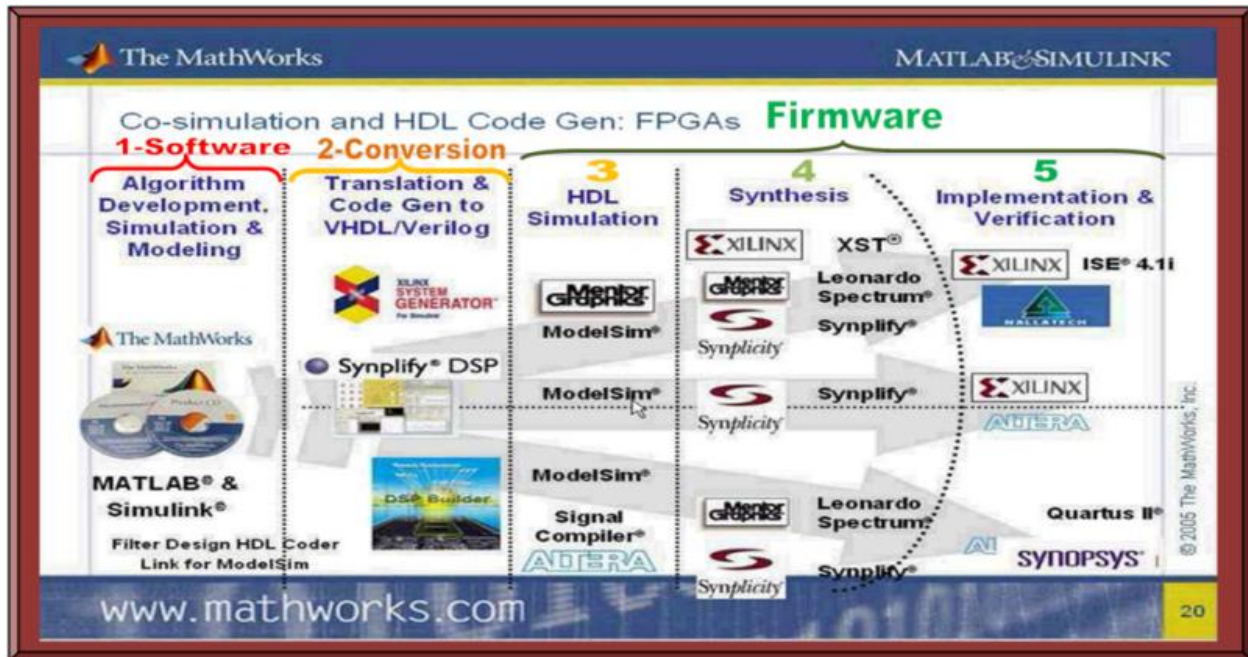


Figure 1.11: Various development analyses and simulation tools
(Adapted from MathWorks, 2005)

Once the GPS receiver design and implementation have been completed, a quick terrestrial functional test to check whether the implemented algorithms are properly operational, will be to use a physical GPS front-end device (see Figure 1.6) to capture real GPS L1 C/A signals to a computer and then post-process the captured data. This processed data is further compared with that acquired by a quality commercial GPS receiver unit operating under the same environmental conditions. The algorithm is considered valid provided the processed GPS data output from both the test and control GPS receivers are approximately the same. Once this basic terrestrial functionality test is successful; high-dynamics Doppler, altitude, velocity and space condition tests can be performed using a high-dynamics GPS simulator (Tsai *et al.*, 2010: 4). Space-borne GPS receiver testing is explained in Montenbruck and Holt (2002) and Gold and Brown (2004). Based-on the test results, further optimisation can be performed until the outcome is acceptable.

Implementation:

As shown in Figure 1.12, a real incoming GPS L1 C/A signal is captured and the GPS signal replica is as well generated. These will serve as received GPS signals datasets, to be used as inputs for initial testing of the GPS receiver functional units; in which, these GPS signal datasets are acquired, tracked and post-processed in off-line (non real-time) mode. Doing so mitigates timing challenges and unknown parameters which can bottleneck the first implementation stage.

The first step in the implementation phase will be to execute a preferred acquisition algorithms listed in Table 1.4. This algorithm of choice should be able to acquire the GPS signal datasets. If need be, this algorithm will then be modified where applicable, to meet the design specifications.

Similarly, a suitable GPS receiver tracking algorithm listed in Table 1.5 will as well be implemented to track and lock to the acquired GPS L1 C/A signal. The tracking algorithm should be able to extract the navigation bits stream. The algorithm will also be tailored if necessary.

Once tracking is successful, the navigation data from the tracking section are decoded and processed as per the GPS signal specifications datasheet (Navstar IS-GPS-200H, 2013). The GPS baseband processing algorithms should be able to compute GPS satellites position and pseudo-ranges, from which the GPS receiver position is computed as well as the velocity and time.

The next step will be to subject the GPS receiver to real-time GPS signal feeds, captured by a USB GPS front-end connected to a computer. This shall be used to test the Matlab/Simulink software-based GPS receiver algorithms in real-time. From the test results, more improvements shall be carried out until the outcome is reasonable, in comparison with a control GPS receiver.

To address the first research problem of inefficient GPS receiver correlator and baseband processing algorithms, simulated space-borne tests shall be performed and if successful, the improved algorithms are converted to firmware as well as subjected through the same tests.

Finally, to address the second research problem (the space-borne GPS receiver high-costs issue), the proposed approach in Winternitz *et al.* (2004: 9), is to focus on COTS FPGAs that are space-grade (e.g. Xilinx Virtex-5QV – Eecatalog.com, 2011) for use in the firmware stages 3 to 5; as depicted in Figure 1.11. The challenges shall be to (i) convert to VHDL (ii) fit / embed the firmware into a single suitable/available space qualified FPGA / DSP device of tiny form factor with onboard resources to meet the design specifications and (iii) repeatedly optimise and test the firmware codes while at the same time trading less and maintaining the receiver functionalities.

Presented in Figure 1.12 is the software/firmware-defined GPS receiver's development process.

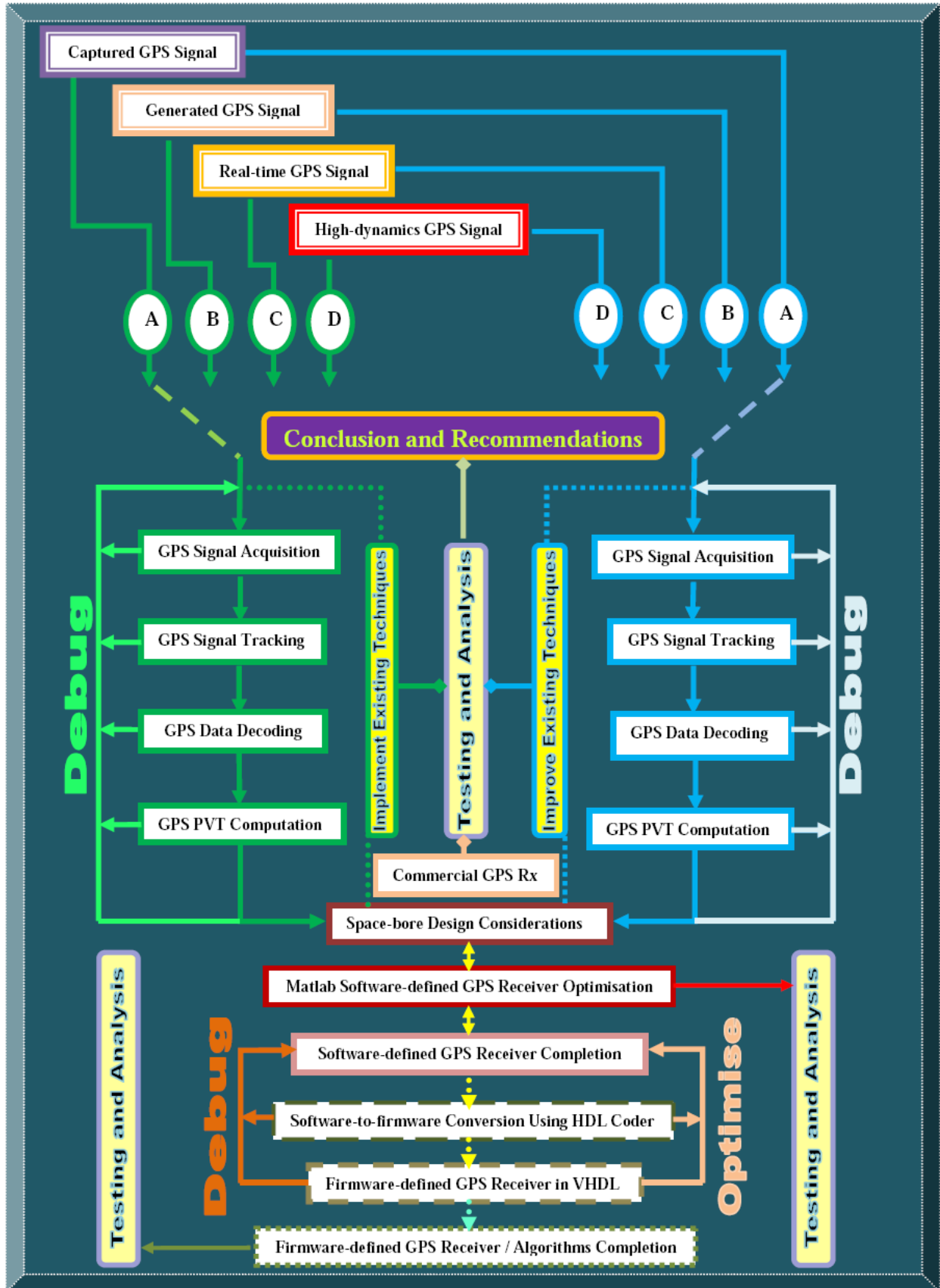


Figure 1.12: Summary of the software-to-firmware GPS receiver design methodology

1.8 Research Project Delineations

This research project has the following demarcations:

As per the primary and secondary objectives, this research will focus mainly on designing and implementing a space-borne software-defined GPS receiver algorithm(s) – to acquire, track, decode and process GPS L1 C/A signal for positioning and as well to obtain velocity and time.

- Embedding the firmware onto a suitable physical microcontroller (FPGA) and designing a hardware low-cost space-borne GPS receiver, do not form part of the present research.
- A real-time GPS L1 C/A signal processing will not be performed.
- Only basic simulated high-dynamics test might be conducted but no space performance test.

1.9 Research Project Outcomes

This research is projected to have the following possible outcomes:

1.9.1 GPS Receiver Expected Results

The GPS receiver is expected to yield the following primary parameters:

- GPS Receiver Position
 - Latitudes
 - Longitudes
 - Altitudes
- Velocity
- UTC Time
- UTC Date

Other secondary resultant parameters that could as well emanate as by-products include:

- Dilution Of Precision (DOP)
 - Position DOP
 - Geometric DOP
 - Vertical DOP
 - Horizontal DOP
 - Time DOP
- Universal Time Mercator (UTM) Zone

1.9.2 Research Project Significance

This research project will serve as the foundation and development platform for the design, implementation and potential commercialisation of F'SATI's low-cost space-borne GPS receiver and possibly other space-borne GNSS receivers for use in nano-satellites.

The research will enable future F'SATI's nano-satellites to automatically and autonomously (i) determine the position(s) of their ground tracking station(s) for tele-commands and telemetry (ii) provides real-time onboard orbit determination / navigation and (iii) to power-off vital electronic devices when around the South Atlantic Anomaly (SAA) region – the SAA region is a high risk radiation area in space (physically above Brazil) that poses ionisation hazard to space objects.

The secondary parameters of Dilution of Precision (DOP: defines GPS satellites geometry in space) and Universal Time Mercator (UTM) zone shall emanate as the processing by-products. DOP presents how orbiting GPS satellites are arranged in space with reference to a GPS receiver. GPS satellites DOPs affect GPS receiver accuracy at the time of position calculation. DOP estimates as well enable a GPS receiver to predict GPS satellites trajectory. UTM zone can be used to provide the time zone of the geographical location of where the GPS signal was acquired.

From the research outcomes, various vital applications to nano-satellites and space science which can also be investigated are respectively CubeSats attitude determination and in-situ space weather ionospheric or scintillation research using GPS. In Yunck and Melbourne (1995), details of some potential applications of space-borne GPS receivers are explained. Additional benefits of software-defined space-borne GPS receivers can as well be found in Gold and Brown (2004).

1.9.3 Research Project Scientific Contributions

There three main research components involved are:

- An improved software-defined GPS receiver.
- A firmware-defined GPS receiver algorithms'.
- A low-cost space-borne GPS receiver implementation approach.

The research resultant scientific contributions to emanate include:

- Discrete Matlab fixed-point GPS receiver algorithms.
- Equivalent Simulink models of the Matlab fixed-point GPS receiver algorithms.
- Equivalent VHDL codes of the Matlab fixed-point GPS receiver algorithms.
- Discrete test benches in Matlab and VHDL for the researched GPS receiver algorithms.
- Improved Matlab floating-point GPS receiver with comprehensive supporting documentation.

The future goal is to have a fully functional space qualified low-cost GPS receiver for CubeSats.

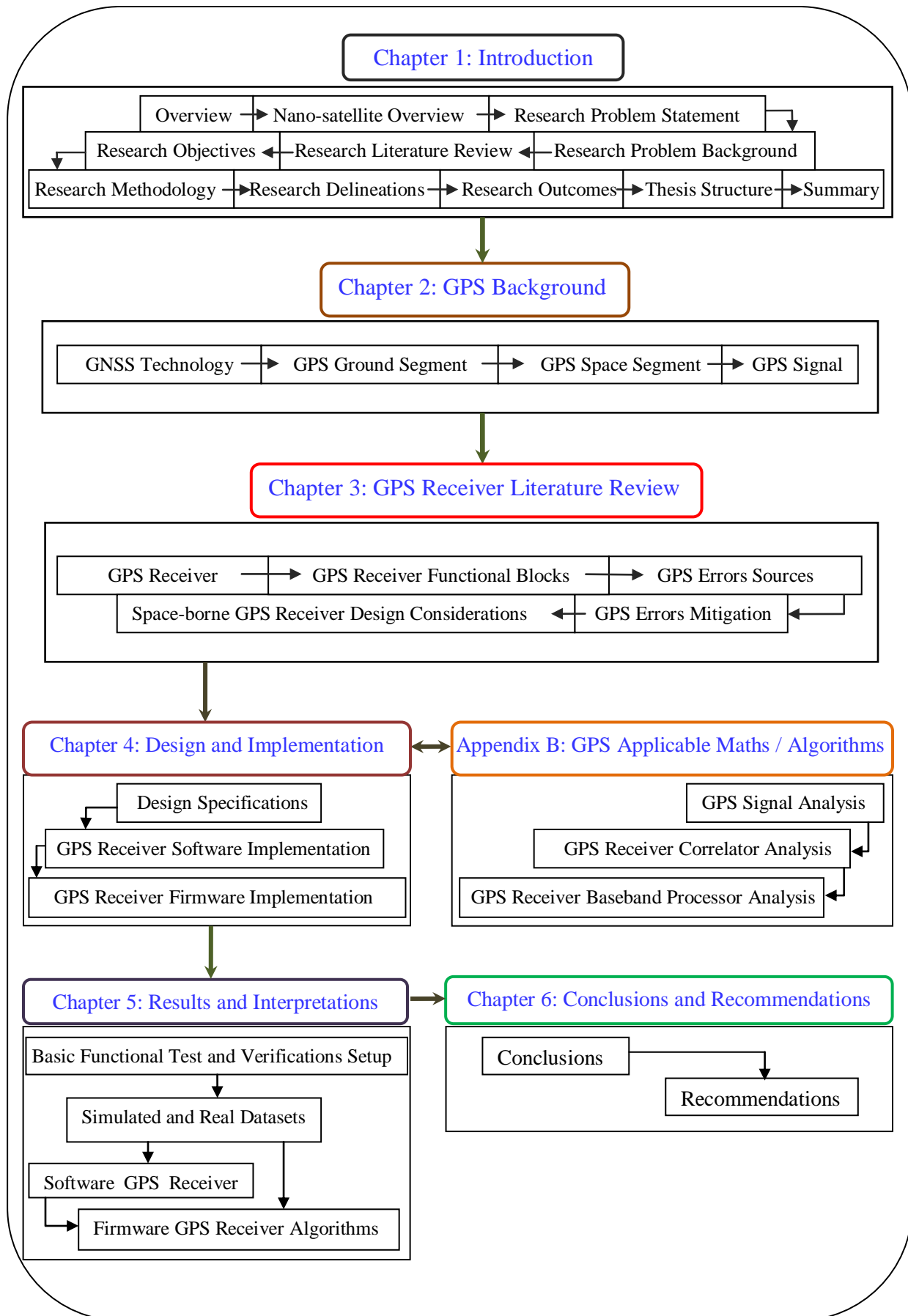


Figure 1.13: Thesis organisation

1.10 Thesis Structure

Figure 1.13 depicts the thesis organisation. Chapter 1 discussed the research proposal and nano-satellite technology with focus on CubeSats. In Chapter 2 is a detailed theoretical background research on GPS with emphasis on GPS signals structure and GPS receiver types. Covered in Chapter 3 and Appendix A, are detailed literature reviews on GPS receiver architecture focusing on GPS correlator algorithms. Appendix B examines the mathematics and algorithms in GPS technology relevant to this research – from which a detailed software-to-firmware design and implementation methodology for the nano-satellite GPS receiver is postulated and executed in Chapter 4. The implemented GPS receiver functional test results (using a simulated real datasets) as well as interpretations are covered in Chapter 5. Concluding remarks and recommendations, as well as improvements to the researched GPS receiver, with focus on the research scientific contributions to the GPS receiver design and implementation are finally drawn in Chapter 6.

1.11 Summary

Chapter 1 started with the review of nano-satellites with attention on CubeSats. The research proposal is the main aspect; in which, high-costs in available space-borne GPS receivers and algorithms efficiency for GPS receivers in high-dynamics environments are briefly discussed. A thorough literature survey on GPS receiver acquisition and tracking algorithms was conducted to set the basis for a suitable algorithm of choice, in-line with the research objectives. The chosen GPS algorithms will be implemented in software based-on Matlab / Simulink platforms. The established schemes will serve as development platforms for the space-borne or high-dynamics GPS L1 C/A correlator and baseband processing algorithm(s) for PVT computations. The designed algorithm(s) will be subjected to simulated low and possibly high-dynamics functional tests. The completed space-borne GPS receiver is expected to be used in future to autonomously and automatically determine orbiting LEO nano-satellites spacecraft position (latitude, longitude and altitude) by acquiring, tracking, decoding and processing transmitted GPS L1 C/A signals.

CHAPTER 2

GPS BACKGROUND

2.1 Introduction

This chapter examines GPS receiver research and other studies related to GNSS and GPS in particular. Various GPS receiver information sources such as textbooks, ebooks, dissertations, theses, research articles, online publications, seminars, workshops, videos, datasheets, journals and conference proceedings are consulted to study the GPS receiver design and implementation.

Chapter 2 describes GPS technology background by examining (i) GNSS technologies (ii) GPS ground stations (iii) GPS satellites and (iv) GPS signal. The first three are discussed in brief whereas the GPS signal is thoroughly covered. Chapter 3 is dedicated for GPS receivers only.

2.2 Global Navigation Satellite System (GNSS) Technologies

According to Gleason and Gebre-Egziabher (2009: 21), GNSS is a constellation of satellites designed to provide global and regional positioning, as well as timing information for users on Earth. However, GNSS technologies such as GPS has been utilised for space-based applications.

GNSS technologies currently consist of full and partial global navigation systems, as well as regional navigation systems. The full GNSS technologies are GPS (USA) and GLONASS (Russia) while the partial ones are the GALILEO (Europe) and COMPASS or BeiDou2 (China).

The regional satellites navigation system comprises of the several worldwide regional Satellite Based Augmentation Systems (SBAS) – which includes Wide Area Augmentation System (WAAS) in the United States, the European Geostationary Navigation Overlay Service (EGNOS) in Europe, the Multifunctional Transport Satellite (MTSAT)-Satellite Augmentation System (MSAS) in Japan; Quasi-Zenith Satellite System (QZSS) in Japan and the Indian Regional Navigation Satellite System (IRNSS) (Hegarty, 2012: 1 - 5). All GNSS technologies operate in the L-band (between 1 and 2 GHz) frequency range and although they have evolved differently (competes and cooperates) over the years (Borre & Strang, 2012: 11 - 14), they share some unique features. For instance, GPS L1 C/A and Galileo E1 carriers both operate on 1.57542 GHz but respectively use Binary Phase Shift Keying (BPSK) and Binary Offset Carrier (BOC) modulation techniques. GPS L1 C/A and GLONASS L1 signals both use BPSK but utilise different carrier frequencies and access techniques. Table 2.1 summaries GNSS technologies.

Table 2.1: Comparison and summary of GNSS technologies

Comparison Parameter	GPS	GLONASS	GALILEO	COMPASS
Owner	USA	Russia	Europe	China
First Launched	1978	1982	2005	2000
Fully Operational	1993 / 1994	1991 then 2011	2019	2020
Main Frequency (GHz)	1.57542	1.602	1.57542	1.57542
Common Signal Name	GPS L1 C/A	GLONASS L1	GALILEO E1	COMPASS B1
Modulation Type	BPSK	BPSK	CBOC	MBOC
Access Techniques	CDMA	FDMA / CDMA	CDMA	CDMA
Chipping Rate (MHz)	1.023	0.511	1.023	1.023
Code length (chips)	1023	511	4092	1023
Satellites in Constellation	24 - 32	21 - 26	27 - 30	27 - 35
Orbital Altitude (km)	~20 200	~19 100	~23 616	~20 - ~36000
Orbital Period	~12 hours	~11 hrs and 15mins	~14 hours	~12 - 24 hours

As a result, most modern navigation devices use a combination of different GNSS technologies to complement each other. GNSS technologies are rapidly evolving with the most successful and widely used worldwide being the GPS and it was therefore selected as the focus of this research.

2.3 GPS Ground Stations

This is the Earth control segment of GPS and consists of four un-manned monitor stations and one manned master control ground station distributed around Earth (see Figures 2.1 and 2.2). They have précised positions and are equipped with superior quality GPS receivers and Cesium oscillators for constant monitoring and control of GPS satellites. Information from these four monitor stations is remotely sent to a master control station – the Consolidated Space Operations Center (CSOC) based at Falcon Air Force Base in Colorado, Springs. The data is processed to obtain the timing bias and ephemeris of each GPS satellite, their positions as a function of time, the clock parameters, the atmospheric data, as well as the almanac. The master control station coordinates all commands and constellation control activities, which includes: to i) monitor GPS satellites performance standards ii) to generate and upload new navigation data needed to uphold performance standards of the satellites iii) real-time detection and response to satellite health that is feedback into the navigation message (Tsui, 2005: 31). All data is remotely sent back from CSOC via three of the four GPS ground monitor stations once and to every GPS satellite daily.

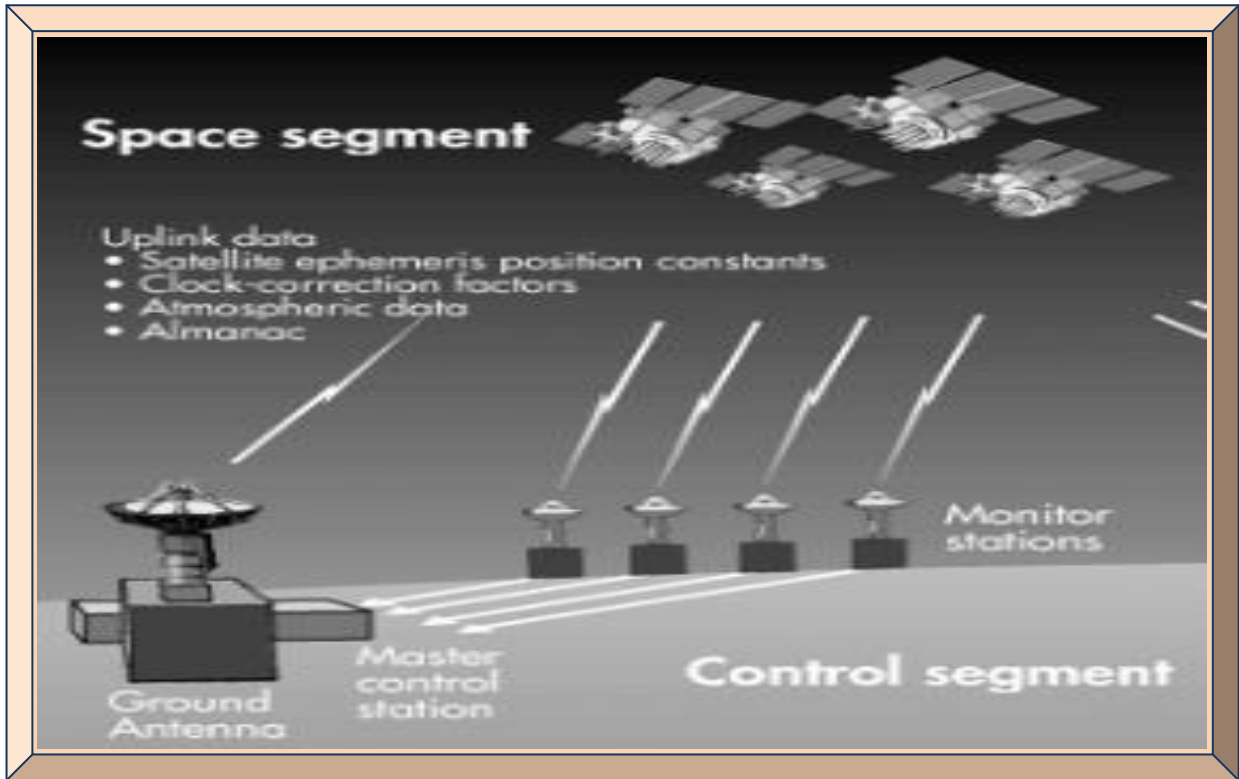


Figure 2.1: GPS ground control segment
(Adapted from McNamara, 2004: 51)

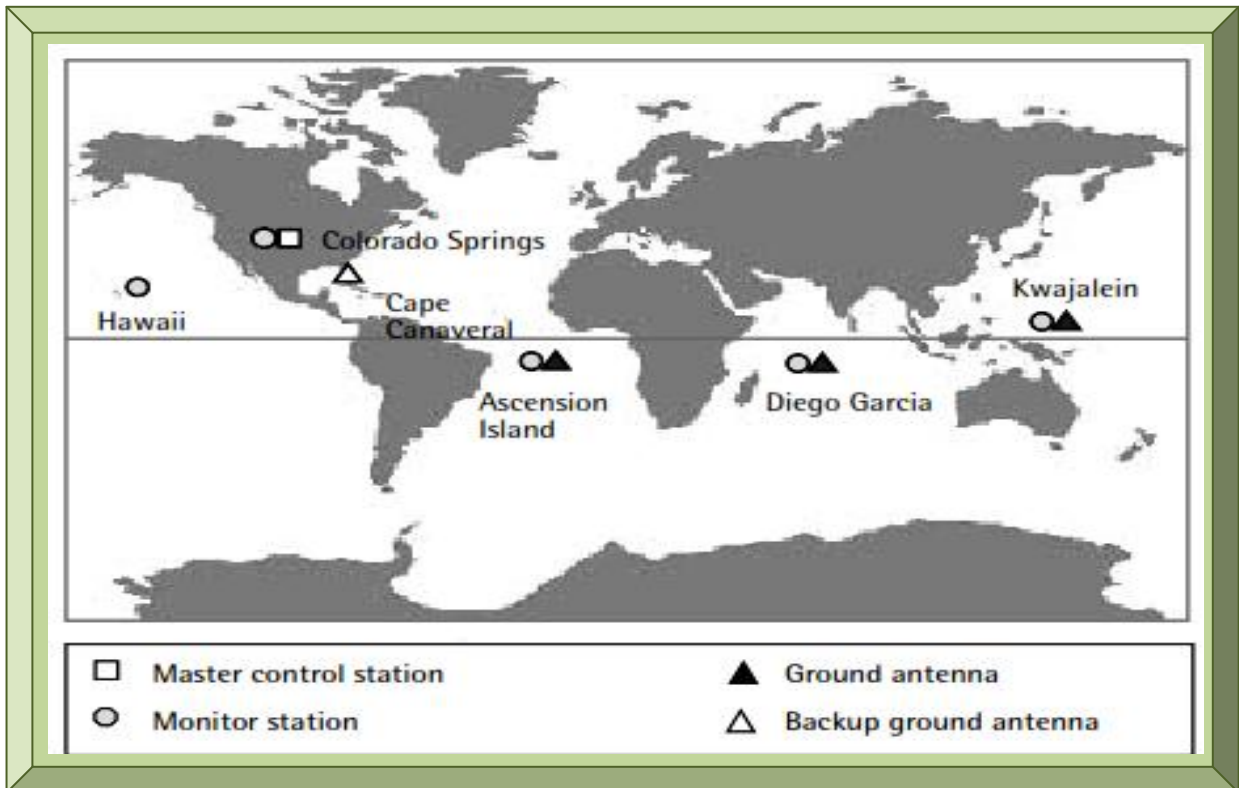


Figure 2.2: GPS ground control sites
(Adapted from El-Rabbany, 2002: 7)

2.4 GPS Satellites

This is the space segment of GPS technology and it is basically a transceiver, as it transmits GPS signals to GPS receivers and transmits and receives signals to and from GPS ground control stations. Signals transmitted by GPS satellites are Right Hand Circularly Polarised (RHCP) and as a result, a GPS receiver's antenna also has to be RHCP to receive incident GPS signals. Multipath signals are reflected signals and thus induce errors if received. If a multipath RHCP GPS signal becomes Left Hand Circularly Polarised (LHCP), it will be rejected by a RHCP GPS antenna. However, if a multipath RHCP GPS signal becomes RHCP, it will not be rejected.

The exact number of GPS satellites – sometimes called Space Vehicles (SV) presently varies. According to Borre and Strang (2012: 11), a constellation of 32 GPS satellites are in operation. These 32 satellites constitute six orbital planes with each plane consisting of four satellites. All planes are equally spaced 60 degrees apart and all four satellites per plane are unequally spaced apart. Each orbital plane is inclined at an angle of 55° from the equatorial plane and each satellite in its plane is at an altitude of $\sim 20,200$ km above the Earth with an orbital period of ~ 12 hrs and repeats the same ground track twice, every sidereal day. Each GPS satellite carries its own high precision atomic clock to maintain very accurate timing. The GPS constellation ensures that, at any point in time, at least 5 to 8 satellites are visible from anywhere on Earth and a minimum of three are needed for location determination while a minimum of four for positioning (location plus altitude) especially if the elevation is varying (Tsui, 2005: 31) as in spacecrafts. Apart from navigation and timing, GPS satellites are also equipped with NUDET sensors to globally detect and monitor nuclear activities as they orbit Earth. Figure 2.3 depicts GPS satellites constellation.

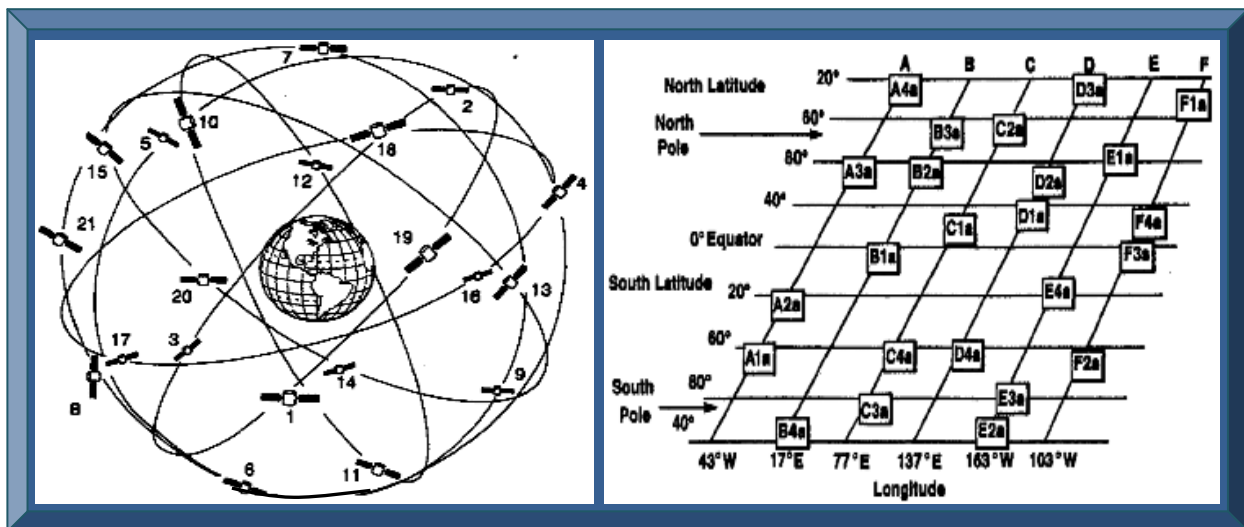


Figure 2.3: GPS satellites constellation
(Adapted from Kemt, n.d.b: 1)

Dilution Of Precision (DOP) is a very vital aspect of orbiting GPS satellites, as it affects GPS receiver acquisition and accuracy. DOP is simply how orbiting GPS satellites are arranged or spaced with reference to a GPS receiver. It is used to describe the geometric strength of GPS satellites configuration of GPS accuracy, at the time of position calculation. DOP is inversely proportional to the volume of GPS satellite tetrahedron formed (see Figure 2.4); that is, higher satellites tetrahedron volume implies lower DOP. A DOP of 1 implies best GPS configuration. There are five types of DOP, namely; Vertical DOP (VDOP), Horizontal DOP (HDOP), Position DOP (PDOP), Time DOP (TDOP) and Geometric DOP (GDOP). Figure 2.4 illustrates the DOP.

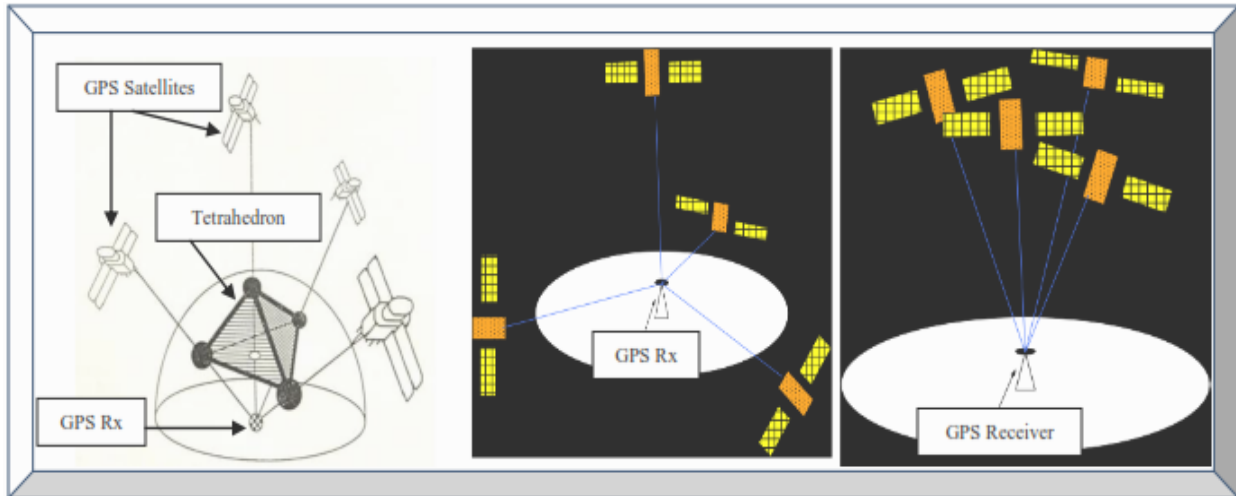


Figure 2.4: GPS DOP (left), Good DOP (middle) and Poor DOP (right)
(Adapted from Lohan, n.d.)

Another important aspect in GPS technology is *GPS satellites ground track* – which is the trace of GPS satellites defined course around Earth. Figure 2.5 red dotted line depicts this parameter.

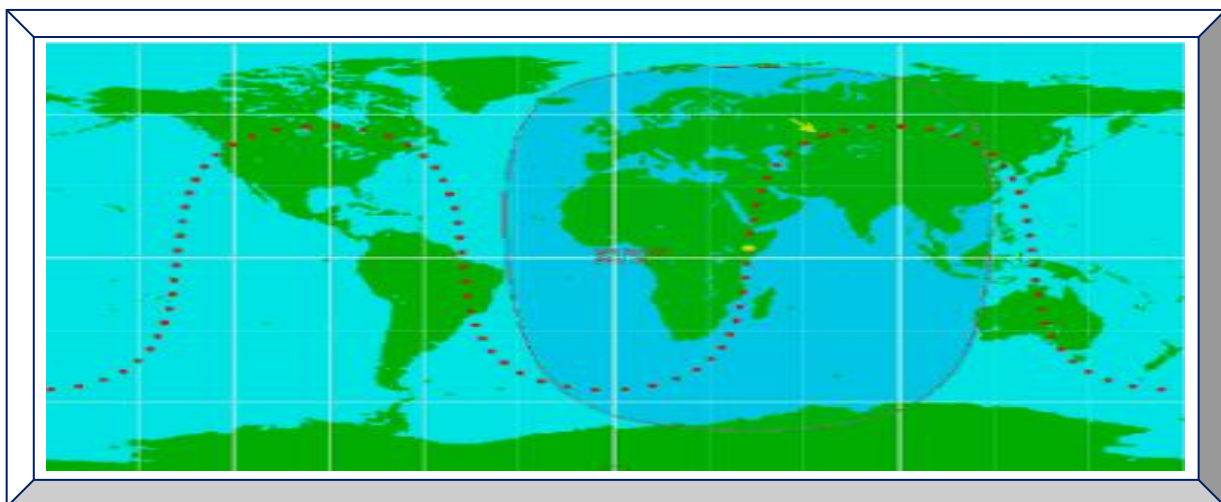


Figure 2.5: GPS ground track as noted by the red dotted line
(Adapted from GPS.Caltech, 2009: 7)

2.5 GPS Signals

A GPS signal is the invisible part of GPS technology and its properties have a major role in the principle and accuracy of Position, Velocity and Time (PVT) determination, navigation data transmission, simultaneous reception of numerous GPS satellites signals, multipath effects, insusceptibility against interferences and provisions of corrections for ionospheric delay of signals (Kowoma.de, 2009). In light of this, the GPS signal structure composes of three signal components; namely the Carrier Frequency, the Pseudo-Random Noise (PRN or Gold) code and the Navigation Data Message. These three signals are blended together to form a composite signal, using BPSK modulation and spread by DSSS and CDMA techniques (Tsui, 2005: 71).

This composite GPS signal structure is broken down and discussed in detailed as follows:

2.5.1 GPS Signal Carrier Frequency

The GPS signal carrier frequency ensures GPS navigation data is propagated through free space to Earth. The following section highlights the selection criteria that were used to choose the different GPS receiver carrier frequencies when GPS technology was established by its creators.

2.5.1.1 GPS Signal Carrier Frequencies Selection Criteria

Transmission of GPS signals require suitable carrier frequencies and the following five aspects were considered when GPS signal carrier frequencies choice was selected (Kowoma.de, 2009):

- The carrier frequencies should lie within 100 MHz and 10 GHz because delays caused by the ionospheres are enormous for frequencies out of this range.
- The carrier frequency must be below 2 GHz, as reception of frequencies greater than 2 GHz might require a beam antenna, which might not be suitable for portable applications.
- The carrier frequency must be above 1 GHz, because (i) higher frequency supports higher data rates and (ii) the electromagnetic waves speed of propagation in media such as air, directly deviates with respect to frequency from the speed of light in media such as vacuum.
- For BPSK modulation with a carrier, the pseudo-random noise codes need a frequency with high bandwidth; therefore, high bandwidth frequencies were chosen to accommodate this.
- The frequencies chosen should be within a range, where the propagation of GPS signals are not negatively affected / influenced by bad weather phenomena such as snow, rain and clouds.

Thus, in Kowoma.de (2009), GPS creators chose carrier frequencies to be in the L-band (carrier frequencies between 1 - 2 GHz) based on these considerations. These carriers are discussed next.

2.5.1.2 Types of GPS Receiver Carrier Frequencies

GPS signals are often transmitted in three different L-band carrier frequencies, namely: L1, L2 and L5 – until recently in 2009; with respective carrier frequencies of 1.57542 GHz, 1.22760 GHz and 1.17645 GHz. These L1, L2 and L5 carrier frequencies are simply selected harmonics of the fundamental frequency ($f_0 = 10.23$ MHz) generated by the GPS satellites atomic clocks. The carrier frequencies up-translation process can simply be deduced as follows (Xu, 2007: 2):

$$\text{L1 carrier frequency} = f_0 \times 154 = 10.23 \text{ MHz} \times 154 = 1575.42 \text{ MHz} = 1.57542 \text{ GHz}$$

$$\text{L2 carrier frequency} = f_0 \times 120 = 10.23 \text{ MHz} \times 120 = 1227.60 \text{ MHz} = 1.22760 \text{ GHz}$$

$$\text{L5 carrier frequency} = f_0 \times 115 = 10.23 \text{ MHz} \times 115 = 1176.45 \text{ MHz} = 1.17645 \text{ GHz}$$

However, there exist L1C, L2C (i.e. variants of L1 and L2), L3 and L4 carrier frequencies but only the GPS L1 and L2 carriers are mostly known and used and will thus, be explained further.

2.5.1.2.1 GPS L1 Signal Carrier Frequency

GPS L1 carrier is the most popular and widely used frequency, with the following properties:

- Carrier frequency is 1.57542 GHz.
- Wavelength is ~ 19.04 cm ($\lambda = c/f$); where c is the speed of light and f the carrier frequency.
- Constitutes Coarse Acquisition (C/A) and Precision (P)-codes – both phase shifted by 90° .
- Transmitted L1 signal has an EIRP level of ~ 27 dBW towards Earth.
- L1 signal modulated by C/A-code has a received signal level of approximately -130 dBm.
- L1 signal modulated by P(Y)-code has a received signal level of approximately -132 dBm.
- L1 bandwidth is ~ 20 MHz (i.e. 20.46 MHz to be précised) to accommodate the P(Y)-code.

2.5.1.2.2 GPS L2 Signal Carrier Frequency

GPS L2 carrier frequency is usually reserved for USA military and has the following features:

- Carrier frequency is 1.22760 GHz.
- Wavelength is ~ 24.44 cm ($\lambda = c/f$).
- Constitutes Precision P(Y)-code only.
- L2 signal modulated by P(Y)-code has a signal level of approximately -135 dBm on Earth.
- L2 bandwidth is ~ 20 MHz (20.46 MHz), since P(Y)-code chipping frequency is 10.23 MHz.

2.5.2 GPS Signal Pseudo-Random Noise (Gold) Codes

GPS Pseudo-Random Noise (PRN) is a Gold code, randomly generated by Maximum Length Sequence (MLS) or Linear Feedback Shift Registers (LFSR) using a deterministic sequence. Due to the varying unique patterns of the random series of 1s and 0s, they are often referred to as noise (Altera, 2003: 1; Borre & Strang, 2012: 16 - 19). GPS PRN codes are principally of two types, namely the C/A and P(Y) codes. These two PRN codes types are orthogonal with distinct properties and generation algorithms. The following sections covered both C/A and P(Y) codes.

2.5.2.1 GPS C/A - Code

GPS C/A-code is a less precise and less complex code, commonly referred to as the civilian code since it is a Standard Positioning Service (SPS) which carries free unprotected navigation data, readily available to the public for civil use. C/A code has the following technical attributes:

- Chipping frequency is 1.023 MHz (i.e. $0.1 \times f_0 = 0.1 \times 10.23 \text{ MHz}$).
- Chip length is 977.5 ns (i.e. $1 / 1.023 \text{ MHz}$) or wavelength of $\sim 293.255 \text{ m}$ (from $\lambda = c / f$).
- 1023 bits (chips) long as generated by its PRN polynomial: $f(x) = 1 + x^3 + x^{10}$.
- Modulates L1 signal only and at a clock or chipping rate of 1.023 Mega chips/s.
- Repeats every 1ms or has a 1ms duration (i.e. $1023 \text{ chips} / 1.023 \text{ MHz} = 1 \text{ ms per chip}$).
- Bandwidth is 2.046 MHz (i.e. $2 \times L1_{\text{caf}} = 2 \times 1.023 \text{ MHz}$) – $L1_{\text{caf}}$ is C/A chipping frequency.
- Has 37 unique Gold codes (practically 36, since Gold codes 34 and 37 are the same).

2.5.2.2 GPS P(Y) - Code

GPS L2 signal constitutes only the P-code, which is more accurate than the C/A code. P-code is a complex Precise Positioning Service (PPS) reserved only for US military and authorised users. Anti-spoofing activation in 1994 led to P-code encryption by a W-code and has been transmitted as a Y-code and is now often denoted as P(Y)-code. It has the following technical characteristics:

- Clock or chipping frequency is $f_0 = L1_{\text{pyf}} = 10.23 \text{ MHz}$.
- Chip length is 97.75 ns (i.e. $1 / 10.23 \text{ MHz}$) or wavelength of $\sim 29.3255 \text{ m}$ (from $\lambda = c / f$).
- P(Y) code is $235.4695928 \times 10^{12}$ bits long (i.e. $15,345,037 \times 15,345,000$ chips).
- Modulates both L1 and L2 signals at a clock or chipping rate of 10.23 Mega chips/s.
- Has a 38 weeks long code ($235.4695928 \times 10^{12} / 10.23 \text{ MHz} = 266 \text{ days} = \sim 38 \text{ weeks}$).
- 1 week segment per satellite – reset/repeats weekly (usually every Saturday breaking Sunday).
- Bandwidth is 20.46 MHz (i.e. $2 \times L1_{\text{pyf}} = 2 \times 10.23 \text{ MHz}$) – $L1_{\text{pyf}}$ is P(Y) chipping frequency.
- Has 37 unique codes segment – with each satellite having a unique 1 week long segment.

2.5.2.3 GPS C/A - Code Generation

The GPS C/A-code has a relatively short length that is randomly generated from 1023 chips, (known as chip instead of bit because it is just a PRN code containing no data) consisting ideally of a pseudo-random sequence of 512 ones (1's) and 511 zeros (0's). The principle is as follows:

In Johanssen *et al.* (1998: 6), GPS PRN C/A belongs to the family of Gold codes and are used because of their distinct cross and auto-correlation properties. Gold codes generators constitute two shift registers known as G1 and G2, with each having 10 cells consisting of 10 chips. Setting or resetting G1 and G2 with all 0's is an invalid state. G1 and G2 must therefore be initialised or re-initialise to all 1's before and after shifting the last chip. By using the formula: $2^n - 1 = 2^{10} - 1$, two sets of 1023 chips long sequence of codes can be generated and modulo-2 added multiple times to give a unique 37 PRN Gold codes. The process is then re-iterated (Rao, 2009: 31 - 34).

The shift register G1 (G_1 generator) PRN code polynomial: $f(x) = 1 + x^3 + x^{10}$ is tapped at positions 3 and 10, then modulo-2 added and feedback to its input. That is, if G_1 generator is reset to all 1's, G1 output after 10 clock pulses will be 1000111000. Likewise, G_2 PRN generator polynomial: $f(x) = 1 + x^2 + x^3 + x^6 + x^8 + x^9 + x^{10}$ is tapped at positions 2, 3, 6, 8, 9 and 10; modulo-2 summed together and feedback to its input. Figure 2.6 and Table 2.2 respectively depict GPS C/A PRN Gold codes generation principle and PRN Gold codes phase designations.

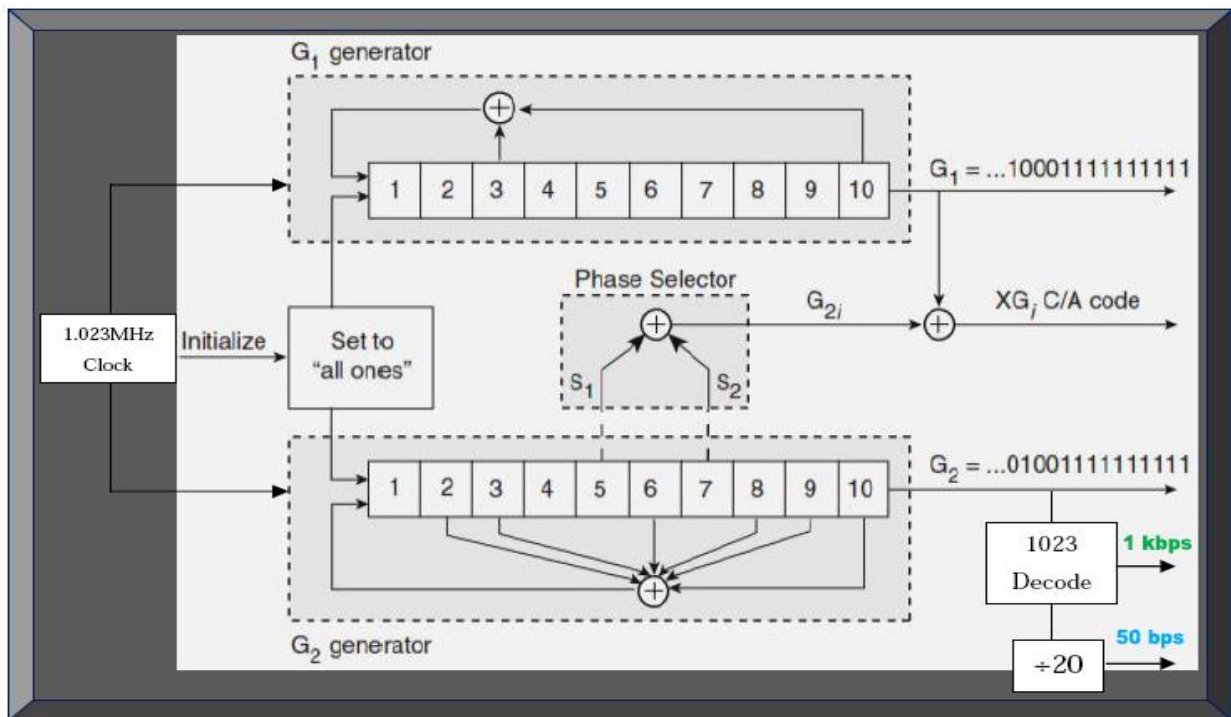


Figure 2.6: Generation of PRN C/A Gold codes
(Adapted from Misra & Enge, 2006)

As shown in Figure 2.6; G1 MLS shift register has one output whereas G2 MLS shift register has two outputs (i.e. G2 and G2i). G2i is the phase select output which is the sum of a modulo-2 adder, produced by tapping at any two positions (e.g. 5 & 7 in the case of Figure 2.6) of G2 shift register to generate a delayed version of output G2. This G2i output is finally added with the G1 output by an ultimate modulo-2 adder to generate the final pseudo-random 37 unique GPS C/A Gold codes (Borre *et al.*, 2007: 21 - 25). The tapped position determines the delay time and thus, the different 36 (ideally 37 but practically 36 since code 34=37) ID codes for each GPS satellites. As per Gold code algorithm, there will be 37 unique codes of the 1023 code length, from which 32 of the 37 have been currently utilised as C/A codes to designate each of the 32 GPS satellites. Hence, GPS receivers can identify respective orbiting GPS satellites from the received C/A code.

For example, in order to receive GPS satellite 25 C/A Gold code; tabs 5 and 7 positions of shift register G2 must be selected (see Figure 2.6 and Table 2.2). With this selection, the G2i output sequence is delayed 513 chips from the G2 output giving a corresponding octal value of 1743 (i.e. first 10 chips is 1111100011 if expressed in binary form). For good correlation to occur, the replica code generated for GPS satellite 25 must exactly match C/A code 25 (Tsui, 2005: 77).

Table 2.2 summarises C/A Gold codes phase designations and is briefly explained as follows: Column 1 represents GPS satellites ID numbers 1 to 32 corresponding to GPS C/A Gold codes 1 to 32 on Column 2. C/A Gold codes 33 to 37 are reserved for other uses – ground transmitters. Column 3 is a code phase selector and its output G2i is a modulo-2 sum of any two G2 generator (shift register) tapped positions. Column 4 presents the code delay (difference between outputs G2i and G2) measured in chips. It simply illustrates an alternative way of expressing column 3 because the delay is automatically determined once the code phase selections are chosen. Finally, column 5 is the octal number format representation of the first 10 chips (number of chips will be 10 if converted to binary) of the C/A code generated for each satellite and is used to validate the generated PRN C/A Gold code (Tsui, 2005: 73 - 78; Navstar IS-GPS-200H, 2013: 2 - 13).

The other four (instead of five GPS C/A codes, since Gold codes 34 and 37 are the same) from 33 to 37 in Table 2.2, are reserved for other uses (e.g. ground transmitters). However, since only 24 and a maximum of 32 GPS satellites are officially known to be in orbit; any of these 32 C/A Gold codes is usually re-cycled once its respective satellite malfunctioned in orbit. The replaced satellite C/A Gold code is then re-assigned to its replacement satellite (Borre & Strand, 2012: 19).

Table 2.2: GPS C/A Gold codes phase designations

(Adapted from Tsui, 2005: 77)

Satellite ID Number	GPS PRN Signal Number	Code Phase Selection	Code Delay Chips	First 10 Chips C/A Octal
1	1	2 ⊕ 6	5	1440
2	2	3 ⊕ 7	6	1620
3	3	4 ⊕ 8	7	1710
4	4	5 ⊕ 9	8	1744
5	5	1 ⊕ 9	17	1133
6	6	2 ⊕ 10	18	1455
7	7	1 ⊕ 8	139	1131
8	8	2 ⊕ 9	140	1454
9	9	3 ⊕ 10	141	1626
10	10	2 ⊕ 3	251	1504
11	11	3 ⊕ 4	252	1642
12	12	5 ⊕ 6	254	1750
13	13	6 ⊕ 7	255	1764
14	14	7 ⊕ 8	256	1772
15	15	8 ⊕ 9	257	1775
16	16	9 ⊕ 10	258	1776
17	17	1 ⊕ 4	469	1156
18	18	2 ⊕ 5	470	1467
19	19	3 ⊕ 6	471	1633
20	20	4 ⊕ 7	472	1715
21	21	5 ⊕ 8	473	1746
22	22	6 ⊕ 9	474	1763
23	23	1 ⊕ 3	509	1063
24	24	4 ⊕ 6	512	1706
25	25	5 ⊕ 7	513	1743
26	26	6 ⊕ 8	514	1761
27	27	7 ⊕ 9	515	1770
28	28	8 ⊕ 10	516	1774
29	29	1 ⊕ 6	859	1127
30	30	2 ⊕ 7	860	1453
31	31	3 ⊕ 8	861	1625
32	32	4 ⊕ 9	862	1712
**	33	5 ⊕ 10	863	1745
**	34*	4 ⊕ 10	950	1713
**	35	1 ⊕ 7	947	1134
**	36	2 ⊕ 8	948	1456
**	37*	4 ⊕ 10	950	1713

*34 and 37 have the same C/A code.
 **GPS satellites do not transmit these codes; they are reserved for other uses.

2.5.2.4 GPS P(Y) - Code Generation

Generating the P(Y)-code is similar to the C/A code generation principle, but differs as four shift registers with 12 cells are rather used. Two registers are combined to produce an X1 output of length 15,345,000 chips that repeats every 1.5s (15,345,000 chips / 10.23 MHz) and the other two registers are also combined giving an X2 output of 15,345,037 chips (37chips longer). Akin to the C/A-code, the X1 and X2 codes can be combined with 37 distinct delays tapped on the X2 shift register to generate 37 different one-week segments of the P(Y)-code, with each of the first 32 segments assigned a corresponding satellite PRN ID and the rest reserved (Rao, 2009: 34 - 38). For instance, GPS satellites with PRN IDs of 10 and 15; respectively refer to GPS satellites assigned to the 10th and 15th week segments of GPS PRN (Gold) P-code. In GPS L1 signal, acquiring P(Y)-code often first requires C/A-code acquisition that is now mitigated in GPS L5.

2.5.3 GPS Signal Navigation Data Message

A GPS signal navigation message constitutes time-tagged data bits marking each sub-frame time of transmission from the respective GPS satellites. The main composition is abridged in Figure 2.7 and Section 2.5.3.1. In Grewal *et al.* (2001: 33), the following are needed to compute PVT:

- *Satellite Almanac Data:* Each GPS satellite transmits the same almanac orbital data, which enables a GPS receiver at any given time to compute the estimated location of every satellite in the GPS constellation. However, almanac data is inaccurate to determine the exact GPS receiver position. It is nonetheless stored in the receiver where it remains valid for ~3 months. It is mostly utilised to (i) obtain the approximate projected Doppler shift signal, to assist in quick acquisition of the satellites signals and Time To First Fixed (TTFF) and (ii) rapidly determine and store visible satellites at a specified location, to enable the search for those satellites when the GPS receiver is later powered-on in a state known as the *warm-start mode*.
- *Satellite Ephemeris Data:* Akin to almanac data, ephemeris data is used to compute a GPS satellite position with more accuracy. This is needed to transform delays in signal propagation for computing a GPS receiver precise position and velocity. Contrary to almanac data that is coarse, GPS satellites ephemeris data is fine and always up to date and unique to that satellite (broadcasted only by it). The data is usually valid for 4 to 6 hours, after which new ephemeris data must be acquired. Issue Of Date Ephemeris (IODE) keeps track of successive updates.
- *Signal Timing Data:* The 50 Hz data stream contains time-stamp, which is utilised to institute the time of transmission of specific points about the GPS signal. This information is vital to ascertain a GPS satellite-to-receiver propagation delay, used for pseudo-range calculations.
- *Ionospheric Delay Data:* Ranging bias caused by ionospheric effects can be partly cancelled, using the approximations of the ionospheric delay (often implemented in the algorithm of position calculations) that are always broadcasted in the GPS navigation data message.

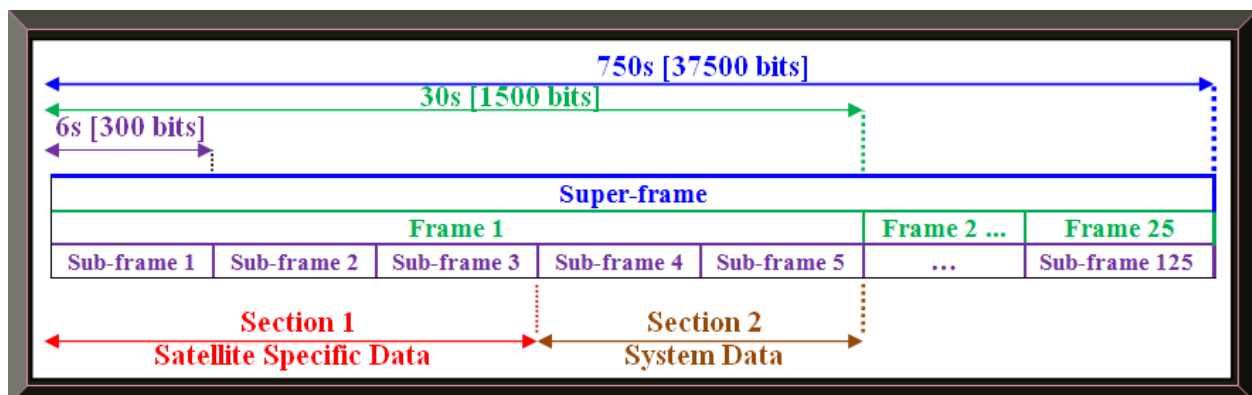


Figure 2.7: Illustration of a typical GPS navigation data message composition – a super-frame

2.5.3.1 GPS Signal Navigation Data Message Technical Parameters Summary

Summarised below are the basic theory needed to understand GPS navigation data message; however, the decoding and processing of GPS messages require an in depth knowledge of the parameters and orbital mechanics, which are covered in detailed in the Navstar IS-GPS-200H.

A GPS signal navigation data message has the following properties (see Figures 2.7 to 2.9):

- Very low data rate of 50 bits/s.
- Bit length is 20 ms (i.e. 1/50 Hz).
- A complete GPS navigation data message consist of a super-frame made of 125 sub-frames.
- The transmission duration for this super-frame (constituting of 25 frames) is 750s = 12.5mins.
- Each of the 25 frames (each frame) is divided into 5 sub-frames totaling to 125 sub-frames.
- Each frame consists of 1500 bits while each sub-frame consists of 300 bits.
- Each sub-frame contains 10 words starting with a unique preamble (10001011 or 01110100).
- Each word has 30 bits including parity bits (i.e. Hamming code for each word).
- Each sub-frame begins with a Telemetry (TLM), followed by a Hand-Over Word (HOW).
- TLM is the first word transmitted and contains an 8 bits preamble (10001011 or 01110100).
- HOW is the second word and has 17 bits Time Of Week (TOW) and its sub-frame ID.
- TOW is synchronised to starting of each next sub-frame.
- TOW count is the 19 Least Significant Bit (LSB) of the Z-count.
- Z-count (29 bits) generate HOW data and count epoch (e.g. number of P-code X1 epochs).
- Each frame is transmitted every 30s while each sub-frame is transmitted every 6s.
- Each bit in a frame is transmitted every 20 ms (i.e. 1/50 Hz) with MSB transmitted first.
- Each of the 5 sub-frames in a frame is further partitioned into two sections (see Figure 2.7).
- Section 1 is referred to as Satellite Specific Data while Section 2 is known as System Data.
- Section 1: Satellite Specific Data refers to Ephemeris and Clock corrections parameters.
- Section 2: System Data refers to Almanac and Ionospheric parameters.
- Section 1 consists of sub-frames 1 to 3 while section 2 consists of sub-frames 4 and 5.
- Section 1 repeats after every 30s whereas section 2 repeats every 750s (12.5 minutes).
- Section 1 sub-frame 1 contains satellite clock correction terms and GPS week number.
- Section 1 sub-frames 2 and 3 contain precise ephemeris data (SVs orbital parameters).
- Section 2 sub-frame 4 has ionospheric and UTC data and satellites 25 - 32 almanac data.
- Section 2 sub-frame 5 has almanac data for satellites 1 - 24 and almanac reference time.

It is also worth mentioning that, the navigation data in sub-frames 1 to 3 are regular and iterate with each frame after every 30s besides during occasional updates; whereas, sub-frames 4 and 5 repeats every 750s (at the rate of a super-frame) except when in occasional maintenance / update by ground stations. Figures 2.8 and 2.9 portray GPS navigation data structure in more detailed.

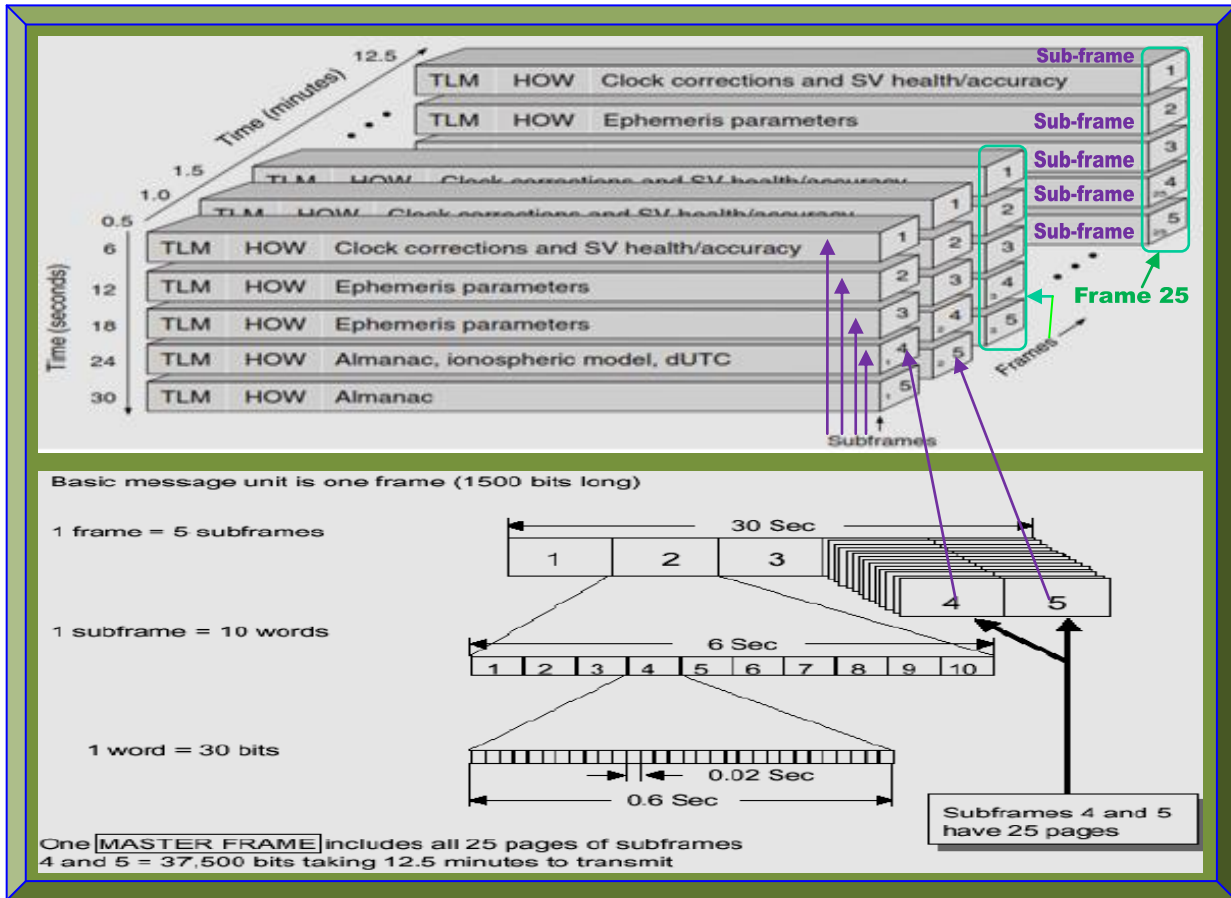


Figure 2.8: GPS signal navigation message – frame compositions in 3D
(Adapted from Borre *et al.*, 29: 2007)

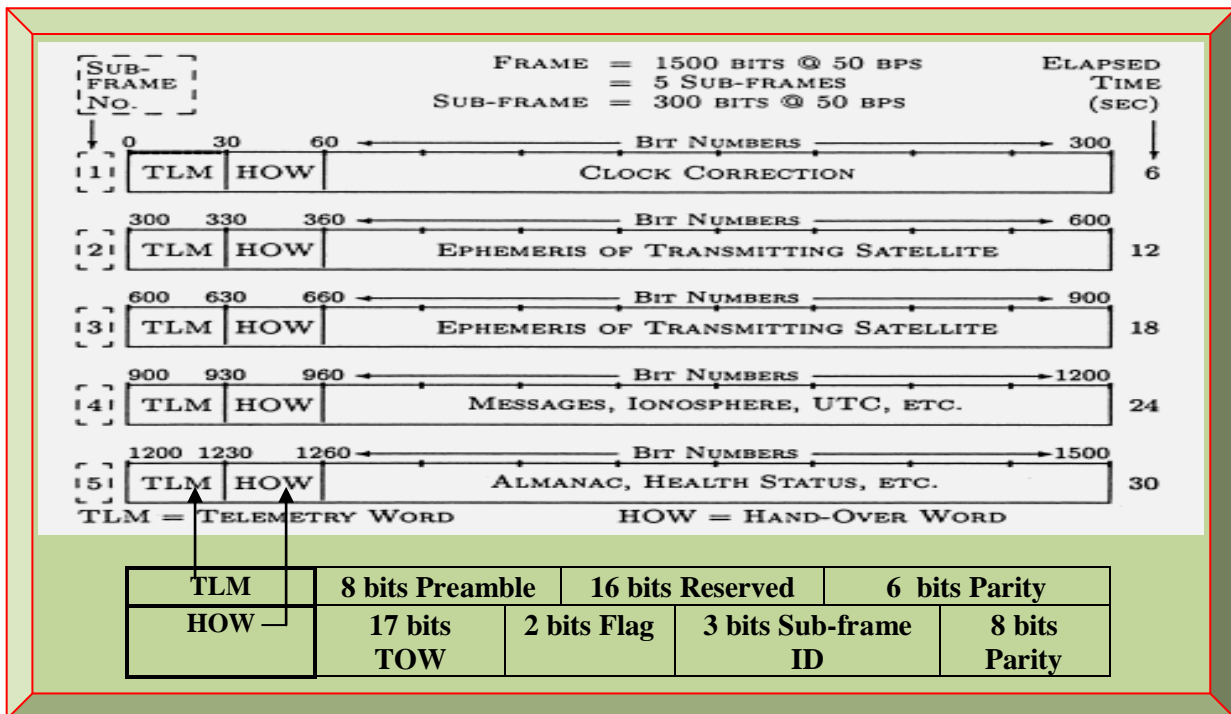


Figure 2.9: GPS signal navigation message – TLM and HOW
(Adapted from Grewal *et al.*, 2001: 34)

2.5.4 GPS L1 Signal Modulation Schemes

This refers to modulation schemes and multiple access techniques implemented in GPS signal transmission. A GPS signal constitutes a Carrier Frequency, PRN Gold codes and Navigation Data, which are blended together by BPSK and transmitted as RHCP (Manandhar *et al.*, 2006).

Modulation improves signal to noise ratio and ensures data is efficiently/reliably transmitted, received and decoded. In GPS, the modulating signals are the Navigation Message, PRN C/A and P(Y) Gold Codes while the modulated signals are the L1, L2 and L5 carrier frequencies.

The modulation and access techniques that ensure GPS signals is propagated and received are:

- Direct Sequence Spread Spectrum (DSSS).
- Code Division Multiple Access (CDMA).
- Binary Phase Shift Keying (BPSK).

2.5.4.1 Direct Sequence Spread Spectrum (DSSS)

Direct Sequence (DS) is a spread spectrum technique that increases the transmitted signal bandwidth by combining it with a pseudo-random noise-like signal of higher bits or data rate. With this approach, just a single carrier frequency is used by all transmitters and the frequency bandwidth is randomly spread using PRN codes. The PRN codes used in GPS signals are C/A and P(Y) Gold codes. These Gold codes spread GPS L1 and L2 signals such that, the harmful effects of unwanted signals are suppressed and the magnitude of the transmitted signal power level is reduced to essentially hide the GPS signal in background noise as shown in Figure 2.10.

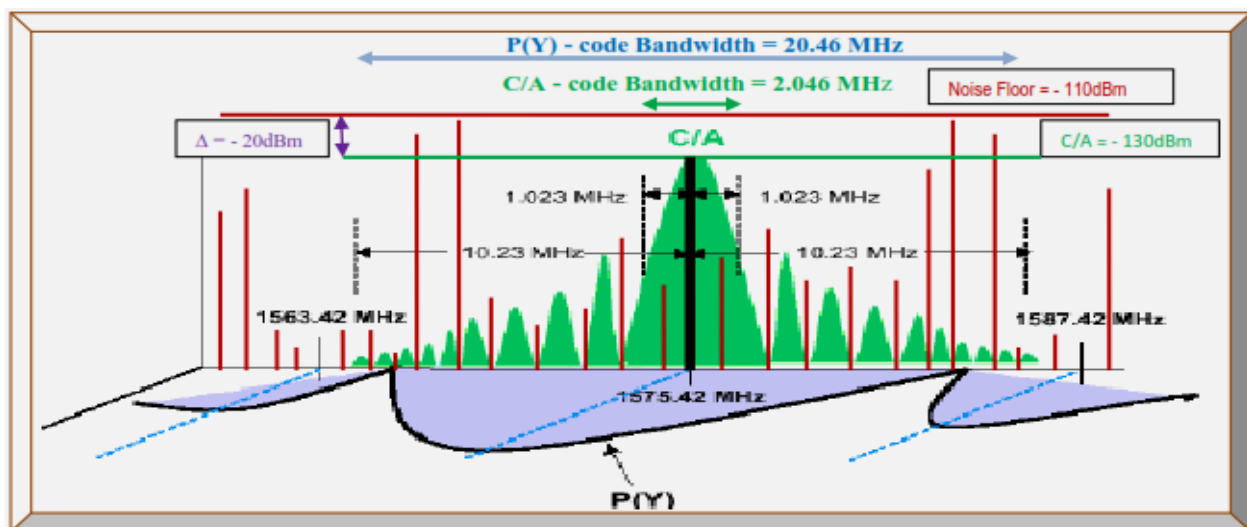


Figure 2.10: GPS C/A and P(Y) codes – spreading the GPS L1 signal
(Adapted from Kemt, n.d.a: 22)

2.5.4.2 Code Division Multiple Access (CDMA)

CDMA is a multiple access technique which enables multiples GPS satellites and receivers to utilise the same carrier frequencies with separate PRN codes distributed by DSSS technique. CDMA nature of GPS signal ensures the following benefits (Altera, 2003: 1 - 2; Tsui, 2005: 71).

- Each satellite is designated a unique PRN code as its ID, thus all can use the same carriers.
- Avoidance of accidental synchronism with other interfering signals.
- Makes it more difficult for GPS signals to be intentionally jammed.
- Very little dynamic range is required from the signal when sampled by the GPS receiver.
- Serial search acquisition, where generated PRN codes and carrier signals are wiped-off.
- All the satellites signal power is overlaid and lowered at the resulting IF; therefore, the spread spectrum bandwidth shows the summation of all visible GPS signals power. See Figure 2.10.

2.5.4.3 Binary Phase Shift Keying (BPSK)

BPSK is a shift keying modulation technique that modulates PRN codes onto the carrier; whereby, a change in the PRN codes states will cause a 180° phase shift in the carrier frequency.

BPSK technique in GPS signal modulation is analogous to modulus-2 addition; thus, in GPS DSP, the former is preferred; in which, GPS PRN chips designations of 0's and 1's are respectively designated as 1's and -1's; whereby, in-terms of BPSK modulation; "0" means a "no shift" whereas "1" means a "shift". During practical implementation, multiplying by "1" un-affects the carrier (no shift) while multiplying by "-1" inverts the carrier (180 phase-shift). Therefore, BPSK modulation is equivalent to Modulus-2 addition. Modulus-2 addition is analogous to an X-OR logic gate; whereby, the output is a high logic (1) provided both inputs are of opposite logic states and a low logic (0), if both inputs logic states are the same. Likewise, BPSK output is simply multiplication of input PRN chips; in which, the output will be a high logic (-1) if both input logic states are opposite and a low logic (1) if both input logic states are the same, as adapted and illustrated in Table 2.3 (Borre *et al.*, 2007: 19; Tsui, 2005: 88).

In a GPS L1 signal blended by BPSK and DS-CDMA techniques, the in-phase (I) component is modulated by a P(Y)-code and the quadrature (Q) component is modulated by a C/A-code and the two Gold codes [C/A & P(Y)] are separated by a 90° phase shift (see Figures 2.12 and 2.13).

Figures 2.11 to 2.13 summarily illustrate the theoretical details and inter-relations in GPS L1 and L2 signals components and chain of events involved from GPS signal generation to transmission.

Table 2.3: Modulo-2 addition and equivalent BPSK truth table comparison

(Adapted from Tsui, 2005: 88)

Phase-shift keying Modulation 1		Modulo-2 Adder Output \oplus	Phase-shift Keying Modulation 2		BPSK Output \otimes
Input 1	Input 2		Input 1	Input 2	
0	0	0	1	1	1
0	1	1	1	-1	-1
1	0	1	-1	1	-1
1	1	0	-1	-1	1

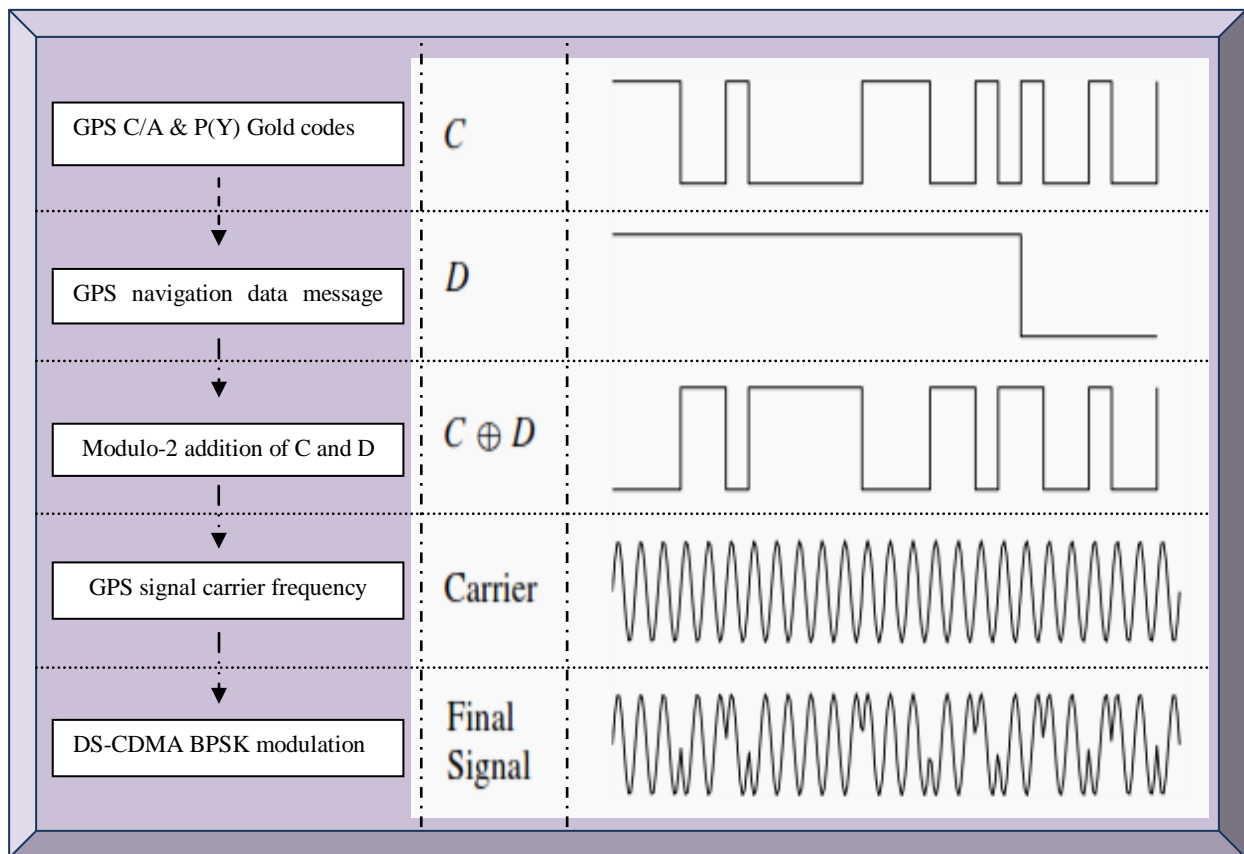


Figure 2.11: Concise summary of GPS signal theory composition

(Adapted from Borre *et al.*, 2007: 21)

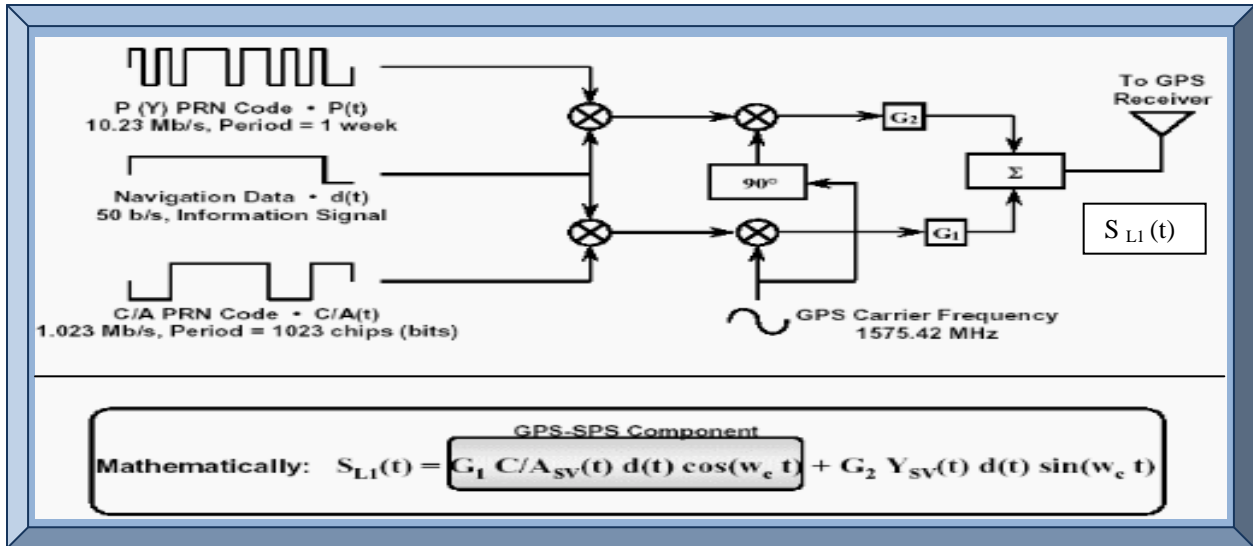


Figure 2.12: Simplified summary of GPS L1 signal structure
(Adapted from Kemt, n.d.a: 20)

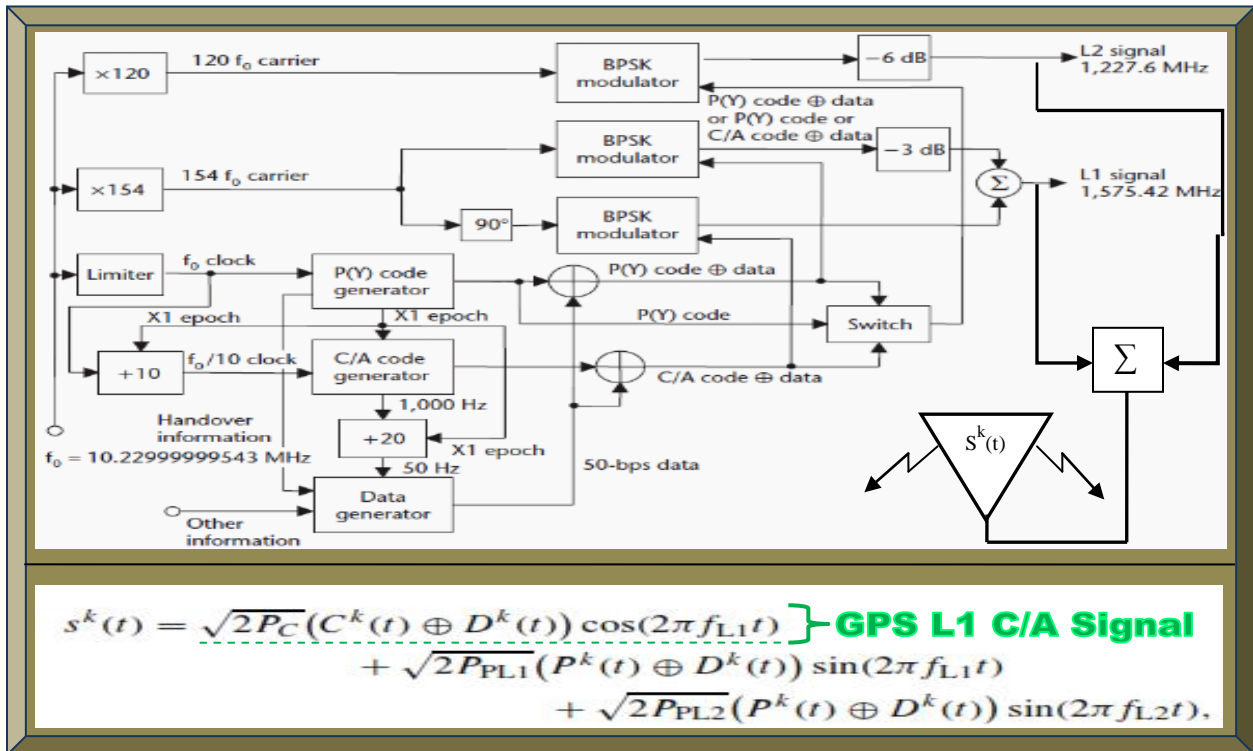


Figure 2.13: Detailed summary of GPS L1 and L2 signals generation
(Adapted from Rao, 2009: 16)

2.6 Summary

This chapter introduced GNSS technologies with the focus on GPS. GPS satellites and ground stations were discussed in brief, whereas GPS signal composition was covered in-depth. Understanding the content of GPS signal structure on how the signal is generated, transmitted, received, decoded and processed; are central to the GPS receiver design and implementation.

CHAPTER 3

GPS RECEIVER

3.1 Introduction

A GPS receiver is the portable user segment of GPS technology and depending on the application; its use varies on land (e.g. surveying), sea (e.g. ships navigation), underground (e.g. mining depth), air (e.g. aircraft formation flying) and space (e.g. satellites tracking). GPS receivers are the focus of this research and specifically the space-borne type for use in nano-satellites. However, irrespective of the end applications, the basic architecture is similar and requires a GPS front-end (for signal reception and down-conversion), GPS correlator (acquisition and tracking) and GPS baseband processor (decoding and processing). The fundamental outcomes of most GPS receivers are position, velocity and time determination (Braasch & Van Dierendonck, 1999). Position determination is the main parameter of interest.

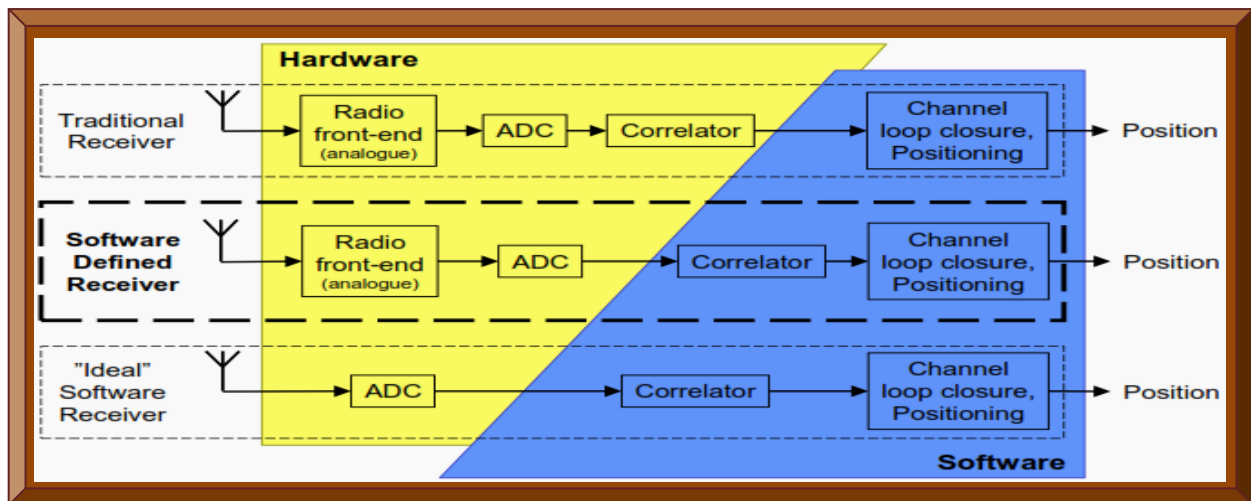


Figure 3.1: Hardware and software trends of GPS receiver

(Adapted from Plausinaitis, 2009c: 16)

Figure 3.1 depicts that, a GPS receiver constitutes both hardware and software. The level of hardware and software involvement defines a GPS receiver type and are classified and discussed under the following three categories. The goal is to choose the most suitable GPS receiver type.

- Hardware-defined GPS receivers.
- Software-defined GPS receivers.
- Firmware-defined GPS receivers.

3.2 Hardware-defined GPS Receivers

By hardware-defined, it means the GPS receiver is designed and implemented using dedicated hardware components. This include amongst others, a dedicated chips or Integrated Circuit (IC) for the GPS front-end, the GPS correlator and the GPS baseband processor (Doberstein, 2012).

Hardware GPS receiver design is often considered as traditional because the internal architecture or circuitry is fixed. However, the performance and functionalities can be modified as follows:

- Replacing the hardware component(s) with newer or better versions, if footprint(s) is equal.
- Upgrading the firmware of some of the component(s) to fix bugs and to add new features.
- Issuing commands to enable or disable and to fine-tune certain functionalities (e.g. power).
- Re-programming or writing new processing source-codes into the GPS baseband processor.

Besides some of the above modification possibilities, Ledvina *et al.* (2003: 2) mentioned some disadvantages and the following drawbacks can be attributed to hardware-defined GPS receivers:

- Their ASIC architecture hinders modification to implement latest trends in GNSS technology.
- Hardware components could become obsolete and hinders replacement and re-production.
- The complete unit could be bulky if many discrete hardware components are used.
- Power consumption could be high as the number and type of hardware components increase.
- Could take longer to prototype / develop if various discrete hardware components are used.
- Could be costly to produce as a result of different types of hardware components used.
- Discrete devices do have diverse operating abilities; thus, unreliable (Renaudie *et al.*, 2007).
- Use of many different hardware components requires specialised knowledge of each involved.

A hardware-defined GPS receiver is suitable in high-dynamics applications, since a dedicated hardware device (a GPS receiver ASIC correlator) is faster (Yao *et al.*, n.d.: 2). This provides continuous and reliable GPS satellites tracking with lock architecture technology (Diaz, 2012).

Doberstein (2012) discussed the design of a hardware-defined GPS receiver in detailed. Since the focus of this research is to design and implement a GPS receiver for space applications (i.e. for use in Cubesats), various ways of a GPS receiver design and implementation were investigated. From the investigation (refer to Tables 1.2 and 1.3), the following most common and feasible approaches for a hardware-defined GPS receiver in this regards, were identified as follows:

1. Using discrete GPS receiver hardware COTS components and devices.
2. Modifying a COTS hardware GPS receiver unit.
3. Porting or emulating the internal architecture or circuitry of the GPS receiver hardware.

3.2.1 Using Discrete GPS Receiver COTS Components and Devices

This is covered in detailed in Section 1.5.1, Tables 1.2 and 1.3. It was the most popular approach as most of the devices were based-on ASIC COTS components. However, as a result of unavailability of the popular open-source Zarlink GP4020 GPS baseband processor from the manufacturer and reliable suppliers, this approach is now less popular and not recommended for new designs, due to lack of genuine future replacements, no future technical support and reliability concerns – especially in space applications. However, there still exists lots of useful reference design information for understanding GPS receiver architecture (Parkinson, 2006: 71).

3.2.2 Modifying COTS GPS Receiver Unit

This is partly covered in Section 1.5.2. In an effort to establish an open-source GPS development platform, Kelley *et al.* (n.d.), alternatively succeeded to capture GPS signals from a Zarlink GP2021 correlator IC and modify it on a PC ISA I/O card, instead of a GPS baseband processor.

3.2.3 Porting or Emulating the Internal Architecture of the GPS Receiver Hardware

The internal architecture(s) of most of the discrete GPS hardware devices are fixed but with this approach, the GPS correlator and baseband processor internal circuitry are migrated and made re-configurable either by (a) emulation or (b) Real-Time Operating System (RTOS). Thus, the GPS hardware IC functionalities can only be modified by re-programming or firmware upgrade. These two porting approaches have evolved differently and have both adopted an open-source framework to enable future unrestricted developments. Two typical instances are presented:

3.2.3.1 Emulation

According to Engel *et al.* (2004), porting by emulation enables modification of the GPS receiver to suit advances in GNSS technologies. Their idea was to strip down and copy the Zarlink GP2021/GP2015 devices GPS architecture to an Altera Nios II FPGA processor and to further create a generic open-source GNSS platform. This approach is feasible but technically involved.

3.2.3.2 RTOS

Contrary to the emulation approach, Greenberg (2005), established an open source General Public License (GPL)-GPS development platform based-on porting an eCOS RTOS to the built-in Zarlink GP4020 ARM7TDMI baseband processor. This programmably interact transparently with the internal GPS correlator. Although there are newer porting plans, this approach requires multiple softwares and is limited to obsolete products, based-on the discontinued GP4020 chip.

3.3 Software-defined GPS Receivers

This was briefly introduced in Section 1.5.3. In Gerein and Brown (2001), a software-based receiver has become the most popular approach in designing and implementing a GPS receiver, as it enables the use of generic hardware that can be tailored with software to suit the end applications (spaceborne in this regards). According to Tsui (2005: 3 - 4), this method evolved from software radio concept; whereby, the received signal is digitised as close to the antenna as possible, by employing DSP techniques. A software-defined approach offers a flexible research development platform, which is ideal for experimenting novel algorithms (Borre *et al.*, 2007: ix).

3.3.1 Software-defined GPS Receivers Based-on Matlab Only

In Borre *et al.* (2007), Borre and Strang (2011); software-defined GPS receivers based-on Matlab have gained tremendous popularity over the years and can be regarded as the de facto approach. Thus, majority of the GPS research (see Reference list) published on Institute of Electrical and Electronic Engineers (IEEE) websites, were researched using Matlab software from MathWorks. This approach basically requires Matlab software running on a suitable PC and real-time functionalities can be tested using a connected GPS front-end with ADC. Real GPS signals are then acquired and process continuously. GPS data can as well be captured and stored for post processing, as well as a simulated or pseudo GPS L1 C/A signal can be generated and utilised.

3.3.2 Software-defined GPS Receivers Based-on Simulink Only

In Hamza *et al.* (2009), Simulink – a graphical computing software tool in Matlab, was used as a software development platform to design and implement a GPS receiver unit. Similar to Matlab approach, GPS signal data is acquired by a GPS front-end with ADC and processed on a PC.

3.3.3 Software-defined GPS Receivers Based-on Both Matlab and Simulink

This approach utilises a combination of both Matlab and Simulink software's from MathWorks to design and implement a GPS receiver. This approach seems suitable, especially as Matlab and Simulink are integrated together as a single package and supports designs inter-conversion. With this method, both software's complement each other in scenarios where GPS receiver algorithms cannot efficiently and / or easily be designed or implemented using either Matlab or Simulink.

In Gleason and Gebre-Egziabher (2009); C/C++, Octave and Python PC languages can be used. In all, a GPS front-end and any suitable PC software with DSP and maths libraries can be used.

3.4 Firmware-defined GPS Receivers

Contrary to software-defined GPS receivers that are designed to run on a PC by utilising the PC's Central Processing Unit (CPU), Random Access Memory (RAM), Read Only Memory (ROM), Operating System (OS) and other resources; a firmware-defined GPS receiver is designed to run on standalone System-On-Chip (SOC) devices such as microcontrollers. Akin to software GPS receivers, they are easily modifiable although with embedded proprietary toolkits.

The design and implementation often follows a similar procedure as that of software-defined GPS receivers; however, the final software is transformed to firmware (small tokenised embedded software for a specific hardware) to run on a particular microcontroller (e.g. Virtex-5QV) in one of two ways: (i) direct adaptation to firmware for a suitable microcontroller or (ii) first converted to software in another language and finally to firmware in an embedded language.

3.4.1 Conversion of Software to Firmware in a Similar Language

This approach is applicable in instances where the software development platform and GPS receiver software have been written in a language that can easily be tailored for direct programming into a suitable microcontroller, without any conversion from one language to another. In other words, the same programming language is utilised within the software development platform and the microcontroller. For instance, a software GPS receiver can be developed in ANSI-C or HDL which is then easily adapted to firmware for direct programming to a suitable C or VHDL based ARM or FPGA microcontroller (Ramakrishnan *et al.*, 2008).

3.4.2 Conversion of Software to Firmware in a Different Language

This approach lends itself to scenarios where the software GPS receiver has been written in a language that cannot directly be used to program a microcontroller but has to be converted to a suitable embedded language using multiple software languages and toolkits (e.g. Matlab with Altera or Xilinx)⁶. This can be performed in two stages as follows: (i) the final GPS receiver software (e.g. developed in Matlab / Simulink) is converted to firmware in another language (e.g. to tokenise code in VHDL or C or C++) using a suitable workflow (e.g. Matlab HDL coder) and (ii) it is further developed and programmed into a suitable microcontroller (e.g. Altera FPGA). Contrary to software which is directly modifiable, a firmware (software embedded in hardware) cannot be directly modified and can only be upgraded and as a whole. The research objectives / available resources, necessitate the software and firmware approaches to be the preferred choice.

⁶ http://www.mathworks.com/products/hdl-coder/?s_tid=hp_fp_list

3.5 GPS Receiver Functional Blocks

This refers to the main operational units or internal building blocks of a GPS receiver. Although the internal architecture of GPS receivers looks similar, literature review from various sources often categorise these internal blocks differently. As briefly mentioned in Section 1.5.3, GPS receivers consists of three main sections namely the RF, DSP and Embedded. Outlined below and illustrated in Figure 3.2 are the three sections which are further split into six sub-sections. This research focuses only on the DSP and Embedded sections designs and implementation. The RF section design and implementation is not covered but is briefly discussed for completeness.

1. GPS RF Section (GPS Front-end)
 - Down-conversion
 - ADC
2. GPS DSP Section (GPS Correlator)
 - Acquisition
 - Tracking
3. GPS Embedded Section (GPS Baseband Processor)
 - Decoding
 - Processing

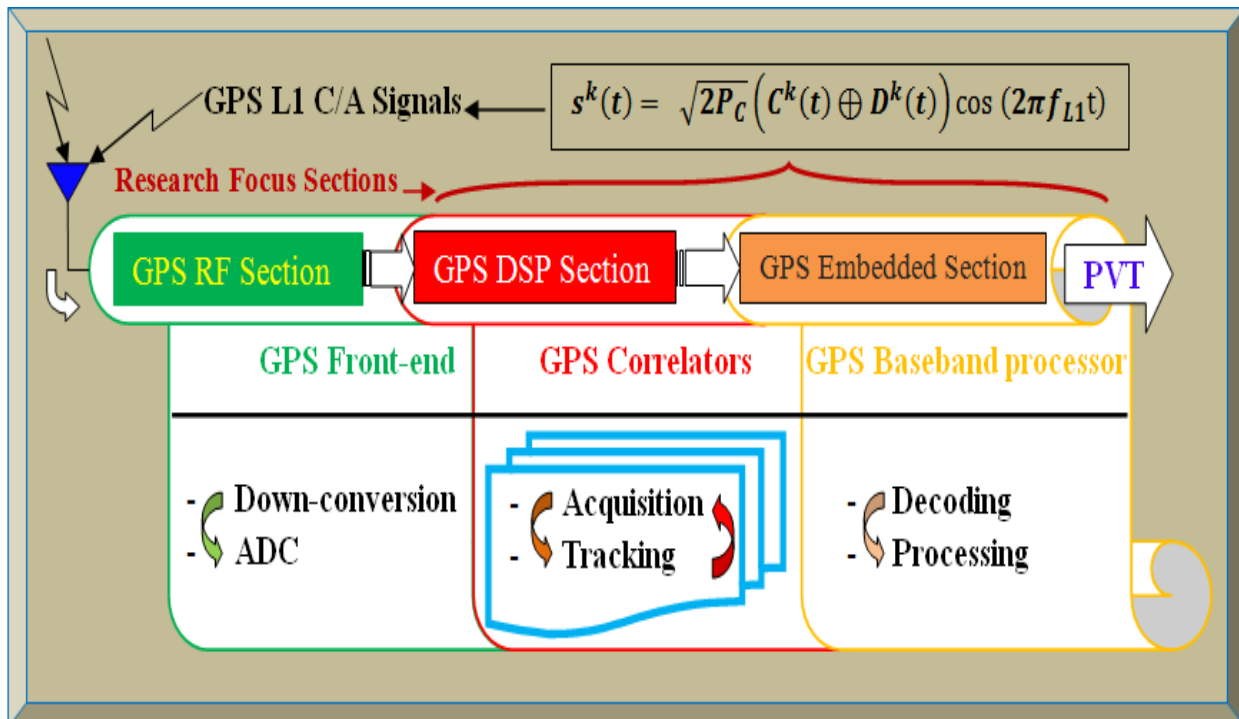


Figure 3.2: Summary of a typical software GPS receiver depicting functional blocks

3.5.1 GPS RF Section (GPS Front-end)

This is the first internal functional block of a GPS receiver and is responsible for converting weak / noisy high frequency GPS signals to a low Intermediate Frequency (IF) suitable for DSP. It is readily available from various vendors as a dedicated chip / ASIC of excellent performance. Since the RF section falls outside the scope of this thesis, the reader is referred to Plausinaitis (2008); Samper *et al.* (2009); Qi *et al.* (2012) for further technical readings. Apart from the antenna and some passive components (e.g. resistors, capacitors and inductors), a typical GPS receiver front-end often consists of the following components either discrete or in-built in a chip:

- *Filters*: extract /filter the desired frequency (1.57542 GHz in this regard) from the aerial input.
- *LNAs*: amplify with little distortion the desired GPS signal. It is placed after the antenna filter.
- *Mixers*: up or down converts the received frequencies at various stages in the GPS front-end.
- *Frequency Synthesizers/Oscillators*: generate reference signals of single/ multiple frequencies.
- *ADCs*: convert the down-converted analog signal (IF) to digital bits for DSP in the correlator.

The filters, LNAs, mixers, frequency synthesizers and/or oscillators usually interact to compose the down-converter and together with the ADC, basically constitute a GPS receiver front-end.

At the input, the antenna is usually the first component of any GNSS receiver and it could be discrete (external active RHCP antenna) and /or built-in (internal passive RHCP antenna). Its function is to transform electromagnetic waves (GPS signals) to electrical or vice versa and provides some frequency discrimination, as well as boosts the GPS signal for the GPS front-end. At the GPS front-end ADC output, the most vital parameters are the Sign (I) and Mag (Q) data.

Figure 3.3 depicts a GPS front-end basic internal functional blocks and Figure 3.4 shows an IC.

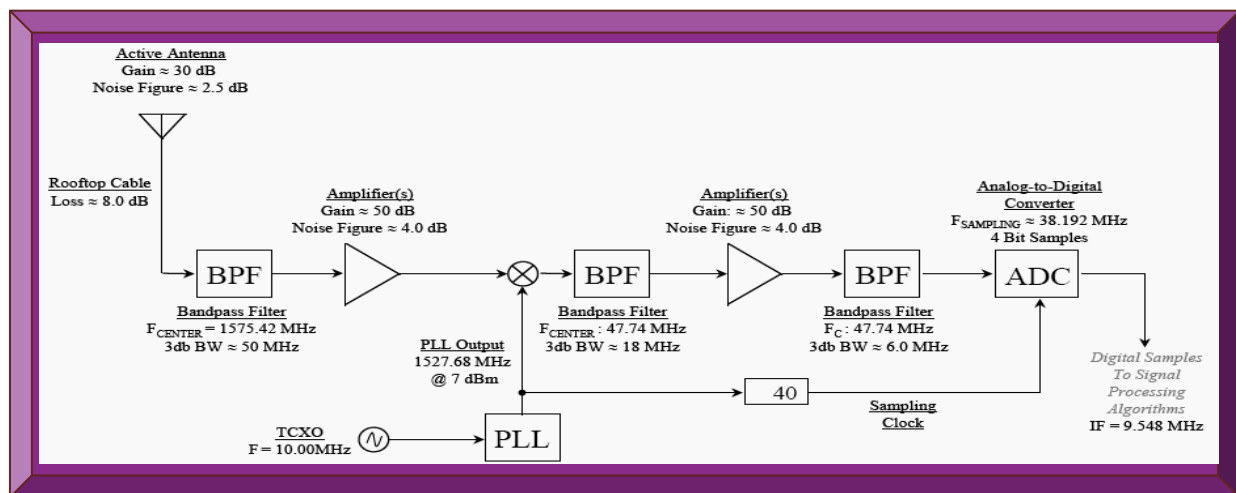
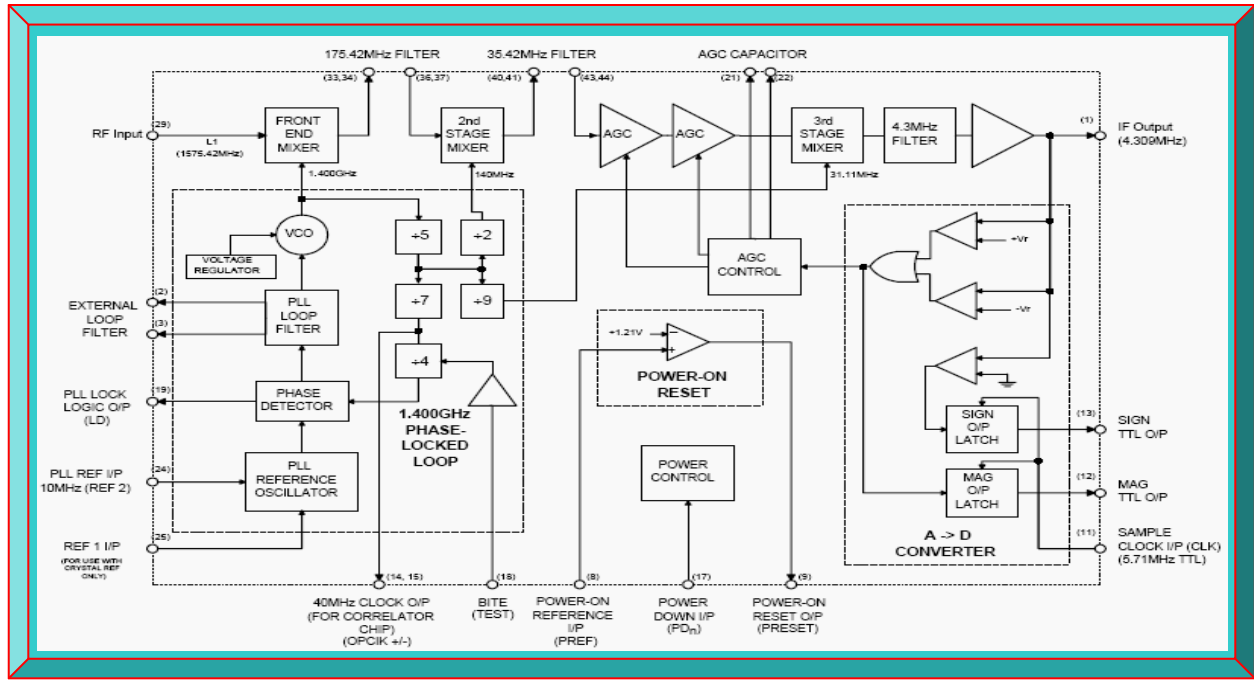


Figure 3.3: GPS front-end basic functional blocks
(Adapted from Plausinaitis, 2008: 56)



**Figure 3.4: GPS receiver front-end IC architecture
(Adapted from Zarlink GP2010 / GP2015 chips)**

Possible GPS front-end devices were examined to select the most suitable device. The summary is presented in Tables 3.1 and 3.2. Table 3.2 devices are ideal for software implementation and the SiGe GN3S Sampler is the most suitable, since it has been used by various research facilities.

Table 3.1: Summary of suitable hardware GPS receiver front-end ICs

Manufacturer	Part number	Dimension	V_Supply	Temperature
Atmel	ATR0601, ATR0603	4x4 mm ²	2v7 to 3v3	-40 to +85°C
Freescale	MRFIC1505	7x7 mm ²	2.7 to 3.3V	-40 to +85°C
Infineon	BGM781N11	2.5x2.5x0.7 mm ³	1v5 to 3v6	-40 to +85°C
MAXIM	MAX2769B, MAX2741/5	7x7 mm ²	2v4 to 3v6	-40 to +85°C
Nemerix	NJ1006A	5x5 mm ²	2.2 to 3.6V	-40 to +85°C
PHILIPS	UAA1570HL	7x7x1.4 mm ³	2.7 to 5V	-40 to +85°C
SiGe	SE4120L	4x4 mm ²	2v7 to 3v6	-40 to +85°C
SONY	CXA1951AQ	9x7x1.5 mm ³	2.7 to 5.5V	-40 to +85°C
ST	STA5630, STB5610	5x5x1 mm ³	1v8, 3v3	-40 to +85°C
uNAV	un8021C / uN1005	4x4x0.9 mm ³	2.7 to 3.3V	-40 to +85°C
Zarlink	GP2010, GP2015	10/7x10/7x2/1.4 mm ³	3 to 5V	-40 to +150°C

Table 3.2: Summary of potential software GPS receiver front-end devices

Software GPS Front-end	SiGe GN3S Sampler v3 USB Front-end	NordNav R30 Sampling Receiver
------------------------	------------------------------------	-------------------------------

3.5.2 GPS DSP Section (GPS Correlator)

Principally, the GPS DSP section comprises the GPS correlator that performs correlation in the forms of acquisition and then tracking, which according to Shi *et al.* (2009), these two processes are key in software-defined GPS receivers. As constrained by GPS signal structure, correlation is the most reliable and efficient technique that can be used to recover GPS signal buried in noise (Zheng, 2008: 6; El-Rayis *et al.*, 2010; Jin, 2012: 5; Ta *et al.*, 2012: 1). The theoretical concept of correlation and its application in GPS signal acquisition and tracking is briefly covered next. For details on GPS acquisition and tracking algorithms, refer to Tables 3.4 - 3.6 and Appendix A.

3.5.2.1 Correlation

Theoretically, the principle of correlation is based-on comparing and quantifying the degree of time, phase, code and frequency alignments in two signals. This could be via cross-correlation (between signals of different kind) and / or auto-correlation (between signals of the same kind).

Correlation threshold is a set value that determines if the result from correlating two signals can be considered as good (accepted) or bad (rejected). This threshold value can be fixed or made to vary autonomously depending on the environmental conditions or the application. However, setting a high threshold value lowers good acquisition probability but decreases false acquisition. Attaining or exceeding a high threshold value relates to good correlation, which often requires that the received signal be powerful with insignificant interferences. This is most likely only when the GPS satellites and receiver have line of sight and GPS sources of errors are minimal. Conversely, setting a low threshold value makes acquisition quicker; however, the acquisition result is unreliable as a low threshold value is susceptible to false triggering (Tsui, 2005: 229).

Correlation value is the number of bits between two GPS PRN codes that have the same value. In Figure 3.5, PRN code patterns of a received and generated GPS signals are shown. The correlation value in this example is 13 (the sum of occurrence of the bits highlighted in orange).



Figure 3.5: Illustration of correlation concept using two 32 bits PRN code patterns

Correlation result is the outcome of correlating two signals. It is considered accepted if it exceeds the set threshold value and rejected if it is below the set threshold value. For example, the correlation result of Figure 3.5 is -6, which is the difference between the correlation and un-correlation value (i.e. 13 - 19). Thus, with a threshold > -6 , the correlation result will be rejected.

3.5.2.1.1 Cross-correlation

Cross-correlation is the correspondence between a PRN code and a phase shifted version of another PRN code of the same length. Therefore, the correlating process between two or more PRN codes transmitted by different GPS Space Vehicles (SV) is known as cross-correlation. It is also the first step in GPS signal acquisition during which the GPS correlator performs a coarse search of available satellites and compares the correlation result against the cross-correlation threshold (Ta *et al.*, 2012). If the correlation result exceeds the correlation threshold, acquisition is completed and that particular GPS signal is then auto-correlated in a process known as tracking (El-Rayis *et al.*, 2010). During tracking, the GPS correlator performs a fine search and tries to maintain lock onto the acquired satellite(s). The acquired satellite being tracked can become unavailable due to reasons such as its poor health status (often notified by the ground station in the decoded navigation message) and interferences of the satellites signals by sources of errors (ionosphere, troposphere, excessive Doppler effects, extreme temperatures) especially in orbiting CubeSats, upon which the GPS correlator could lose lock (Markgraf *et al.*, n.d.: 6).

3.5.2.1.2 Auto-correlation

Correlating two identical signals is known as auto-correlation and in GPS technology; it is a one to one correspondence between a PRN signal against a phase shifted version (replica) of itself. For example, SV 21 does transmits Gold code 21 and if this transmitted signal is received and correlated against itself by the GPS receiver, the process is known as auto-correlation and it is the principle a GPS correlator employs when refining / locking to an acquired GPS PRN signal.

The incoming GPS signal (PRN 21) and the identical locally generated replica (rPRN 21) are always practically not aligned in phase and frequency; as a result of clock bias errors, atmospheric delays and Doppler shift encountered by the transmitted GPS PRN 21 signal. The locally generated replica by the GPS receiver correlator has to be continuously shifted until both signals are aligned (Borre *et al.*, 2007: 70). Depending on the correlation result, the degree of alignment can be considered accepted or rejected. Once the correlation threshold is attained or exceeded, the correlated result will give a significant rise in amplitude (see Figure 3.8) and this effect is termed acquisition or the GPS SV 21 signal is acquired. To maintain this acquisition, the GPS signal coarse code phase and frequency values have to be continuously refined and locked unto (Borre *et al.*, 2007: 71 & 72). From research, various techniques for acquiring and tracking GPS signals have been proposed and implemented. These are briefly discussed in Appendix A.

The concept of an ideal auto-correlation process is illustrated as follows in Figures 3.6 and 3.7.

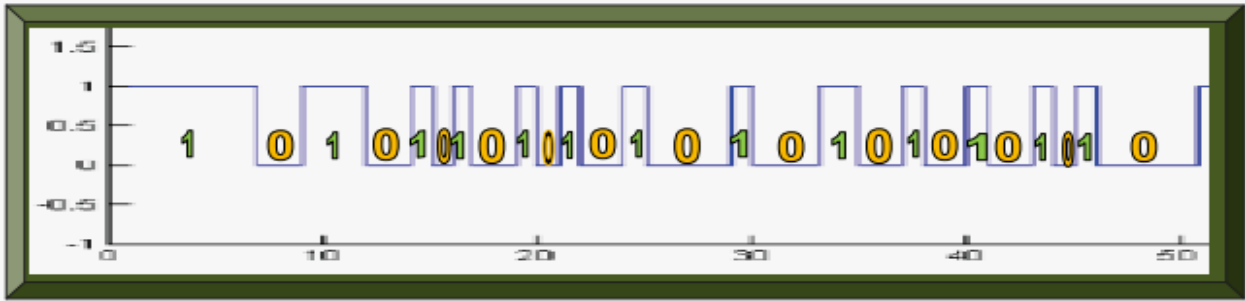


Figure 3.6: Conceptual received GPS L1 C/A signal transmitted by SV 21 showing first 50 bits

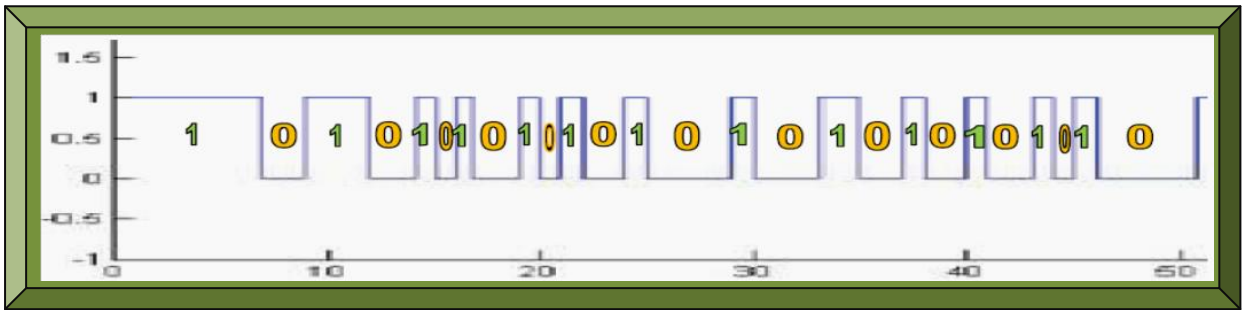


Figure 3.7: GPS receiver generated replica of GPS L1 C/A signal of SV 21 showing first 50 bits

Since Figures 3.6 and 3.7 each has 50 bits (1023 max) represented by illustrating only 26 bits of 1's and 0's; the correlation threshold can be set to a value $> +25$. In this instance, it can be set to $+45$; however, since this is an ideal example because both signals are designated to be properly aligned for illustration sake, the correlation value is 50; hence, the magnitude or peak of the correlation result is 50 as well, signifying an ideal acquisition result in this illustrative scenario.

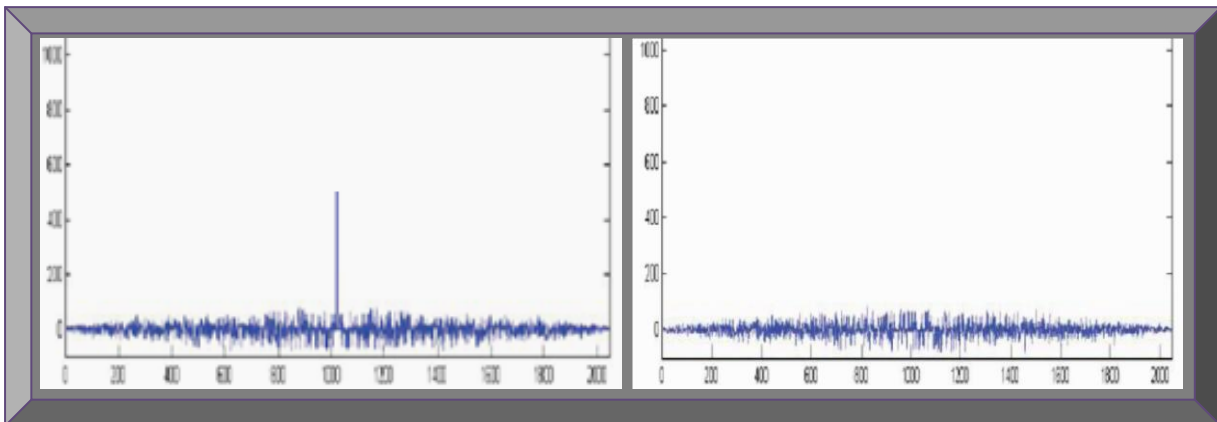


Figure 3.8: Auto (left) and cross (right) correlation results
(Adapted from Borre *et al.*, 2007: 26)

Figure 3.8 shows the signal spectrums resulting from auto and cross-correlating two GPS signals. Thus, the ability of GPS correlator (s) to quickly acquire GPS signals and to track and maintain lock unto the GPS signals in adverse conditions, determines and defines a GPS receiver quality.

3.5.2.2 Acquisition

Acquisition is the first of the two correlator tasks performed by a GPS correlator. Practically there is always more than 1 correlator or channel in a GPS receiver – usually 12, with each correlator tracking a specific GPS satellite (Greenberg, 2005: 7). For a GPS L1 C/A signal acquisition to occur, the GPS receiver must perform a rough search for a minimum of four available satellites. This search is usually termed two dimensional as it involves both the code phase and carrier frequency of the transmitted GPS L1 C/A signal (Greenberg, 2005: 8). The CDMA nature of GPS L1 C/A signal enables a GPS receiver to recognise and separate multiple GPS satellites on the same carrier frequency based on the unique PRN code assigned to each GPS satellite. As a result, the PRN codes only correspond or correlate, if they are perfectly aligned. Hence, each GPS satellite transmits a unique PRN code, which at lag 0; cross-correlates well only with a different GPS satellite's PRN code, as well as, auto-correlates well with only the same GPS satellite's PRN codes. In other words, different GPS PRN codes are highly mutually orthogonal with each other, making them very unlikely to be misinterpreted as another. Each of these unique GPS satellite's PRN codes, range from 1 to 37 and are often refer to as Gold codes (see Table 2.2, though there are a total of 1025 each of length 1023 bits); from which, only 1 to 32 are assigned to uniquely represent each orbiting GPS satellite (Borre & Strang, 2012: 17 - 21). Thus, acquisition finds the start of the PRN codes, the carrier frequencies and phase transitions.

In Borre *et al.* (2007: 69 - 71), the GPS receiver during cold start acquisition process, normally has outdated or no almanac data in memory. It then enters a coarse search mode and generates PRN code phases and carrier frequencies replicas to match the incoming code phase and carrier frequency of the transmitted GPS signal. An offset is computed by the MLS shift registers to generate the best correlation and during the offset computation, the 1023 bits of the satellite PRN code are first compared with the generated or stored PRN code. If correlation is not attained, the 1023 bits of the GPS correlator PRN codes are shifted by one bit relative to the GPS satellite's PRN codes and the signals are iteratively compared until correlation is achieved or all 1023 possible cases have been tried. If no correlation, the process is iterated via the frequency search bins by offsetting the generated carrier frequency to the next value until acquisition is obtained.

As a result, a GPS receiver's correlators must continuously shift (slide) the code phases and carrier frequencies of the generated replicas until it matches or aligns with that of the received GPS satellites signals. Once correlated, the GPS signal amplitude of the correlation result will rise significantly, implying acquisition and control is passed to the GPS correlator tracking loop.

3.5.2.3 Tracking

Tracking is the second of the two correlator tasks performed by a GPS correlator and it refines the acquisition process by ensuring a constant lock of the received and generated GPS signals. Once coarse acquisition is completed, the GPS receiver correlators then perform fine acquisition to track and lock all available acquired GPS satellites. Once locked, the tracking loops select or determine the GPS satellites to listen to, for navigation data decoding (Johanssen *et al.*, 1998: 8).

Obtaining and maintaining GPS lock often requires a clear line of sight between GPS receiver and satellites antennas; however, the received GPS signals code phases and carrier frequencies are usually affected by GPS sources of errors (atmospheric delays, Doppler and multipath). Therefore, as stated in Borre *et al.* (2007: 69 - 72), the incoming GPS signals and the generated replicas will always be misaligned or different in PRN code phases and carrier frequencies and thus, will not correlate due to these misalignments. As a result, the GPS receiver correlator temporally loses lock and has to re-track the acquired GPS signal. Furthermore, should the GPS signal impairment condition between GPS satellites and receiver exceeds 4 to 6 hours; the ephemeris data will become outdated and the GPS correlator will revert back to acquisition mode and performs new GPS satellites search – starting from the supposed warm or hot start states. Table 3.3 depicts a GPS receiver acquisition states and Figure 3.9 portrays a typical acquisition.

Table 3.3: GPS signal acquisition and tracking states

	State	Technical meaning	Real life analogy
1	Cold	GPS receiver has outdated almanac data	Used and powered-off for months
2	Warm	GPS receiver has recent almanac data	Used and powered-off for days / weeks
3	Hot	GPS receiver has both almanac/ephemeris data	Briefly powered-off or connection lost

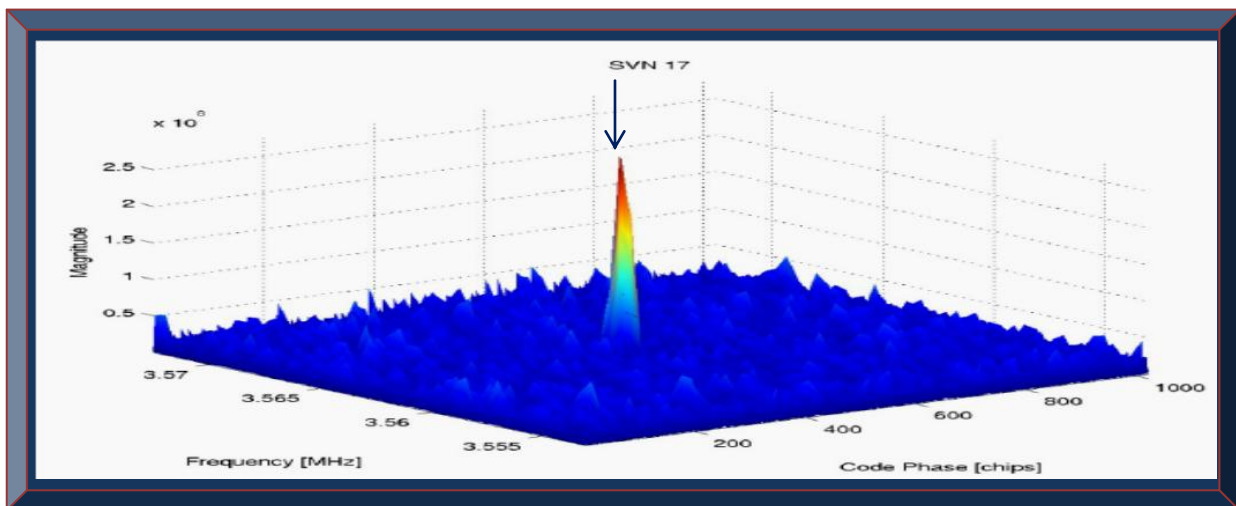


Figure 3.9: Illustration of an acquired GPS signal from satellite 17
(Adapted from Plausinaitis, 2009c: 6)

Table 3.4: GPS L1 C/A Acquisition Techniques

Acquisition Algorithms	Source
Acousto-optic Technique	Bazzi <i>et al.</i> (1993)
Mitel GP2000	Mitel GP2000 (1998: 14 & 15)
Averaging Correlator Acquisition	Alaqeeli <i>et al.</i> (2003)
Parallel Code Phase Search (Delay & Multiply FFT)	Tsui (2005: 139); Borre <i>et al.</i> (2007: 82)
Block (Non-coherent Integration) Acquisition	Tsui (2005: 144); Shi <i>et al.</i> (2009)
Multiple Correlators Bank Acquisition	Akopian <i>et al.</i> (2006)
Lifting Wavelet Acquisition	Djebouri <i>et al.</i> (2006)
Serial or Sequential Search Acquisition	Borre <i>et al.</i> (2007: 75 – 78)
Parallel Frequency Space Search Acquisition	Borre <i>et al.</i> (2007: 78 – 81)
Wavelet De-noising Acquisition	Tian and Yang (2008)
Reconfigurable Instruction Cell Array Acquisition	El-Rayis <i>et al.</i> (2010)
Wavelet Transform Acquisition	Jianguo <i>et al.</i> (2010)
Extended Multiple Correlators Acquisition	Xiaowen <i>et al.</i> (2010)
Bayesian Acquisition Technique	O'Mahony <i>et al.</i> (2012)
Matched Filter Correlator Acquisition	Ta <i>et al.</i> (2012)
Fast Satellite Selection Based-on Newtons Identities	Meng <i>et al.</i> (2013)
Sub-Sampled Fast Fourier Transform	Rao and Ratnam (2013)
Improve Fast Fourier Transform	Feng <i>et al.</i> (2013)

Table 3.5: GPS L1 C/A Tracking Techniques

Tracking Algorithms	Source
Block Adjustment of Synchronising Signals	Tsui (2005: 170 – 175)
Combined Carrier and Robust DLL Code Tracking	Tsui (2005: 168); Borre <i>et al.</i> (2007: 106)
PLL and Costas Loop Carrier Tracking	Borre <i>et al.</i> (2007: 93 – 96)
Basic DLL Early-Late Code Tracking	Borre <i>et al.</i> (2007: 96 – 100)
Robust DLL Early-Late Code Tracking	Borre <i>et al.</i> (2007: 98)
Nonlinear Code Tracking Filter Technique	Gustafon <i>et al.</i> (2009)
Inertial Navigation System Tracking Technique	Rogers (2007); Xiaoyong and Baiqi (2011)
Combination of Squared Correlators Tracking	Falletti <i>et al.</i> (2012)
Extended Kalman Filter Tracking Technique	Peng <i>et al.</i> (2012)
Kalman Filter Tracking Loop Technique	Song <i>et al.</i> (2012)
Wavelet Packet Denoising	Eddine and Mohamed (2013)
Carrier Phase Measurement	Lu <i>et al.</i> (2013)

Table 3.6: Miscellaneous Research on GPS L1 C/A Acquisition and Tracking Techniques

Acquisition Source	Tracking Source
Ma <i>et al.</i> (2004)	Kalyanaraman <i>et al.</i> (2004)
Gunawardena and Graas (2006)	Lin <i>et al.</i> (2006)
Lashley (2006)	Sagiraju <i>et al.</i> (2006)
Mao <i>et al.</i> (2006)	Hsu <i>et al.</i> (2008)
Li <i>et al.</i> (2008)	Zhe <i>et al.</i> (2008)
Yao <i>et al.</i> (2009)	Lashley and Bevly (2009)
Diaz <i>et al.</i> (2012)	Miao <i>et al.</i> (2009)
Tang <i>et al.</i> (2012)	Yang-bo <i>et al.</i> (2009)
Borio <i>et al.</i> (2013)	Borio <i>et al.</i> (2010)
Kumar and Paidimarry (2014)	Li <i>et al.</i> (2013)

3.5.3 GPS Embedded Section (GPS Baseband Processor)

This is the last functional block of a GPS receiver. It principally serves two purposes: i) to decode the tracked GPS signals and ii) process the GPS navigation data. Some of the relevant technical details regarding GPS baseband processing are partly covered in Section 2.5.3. As described in Tsui (2005: 186 - 223); Borre *et al.* (2007: 109 - 135); Navstar IS-GPS-200H (2013) specification, the decoding and processing follow a standard procedure. The next subsections briefly highlights the theoretical aspects of the GPS Baseband Processor. Appendix B thoroughly covers the mathematical analysis (mainly the PVT calculations) applicable to this research.

3.5.3.1 Decoding

In Borre *et al.* (2007: 109 - 125), decoding means to reliably recover the GPS navigation data message. For that to successfully happen, the following processes have to be logically followed.

3.5.3.1.1 Bit Synchronisation

According to Borre *et al.* (2007: 109), this refers to finding the bit transition and bit value, which basically involves detecting a zero crossing – where the output alternates from 1 to -1 or vice versa. Locating this zero crossing automatically determines the bit transition time and makes it possible to find all the other bit transitions, that are usually 20ms (duration of a GPS navigation data message = 1/50 Hz) from each other. Since the tracking block output sampling rate is often 1000sps (samples per second) which corresponds to a value of 1ms, the output signal must be converted from 1000sps to 50bps (bits per second). A bit must be represented by 20 samples since a GPS navigation data message bit duration is 20ms. This conversion procedure is termed bit synchronisation and is the most crucial step in GPS navigation data message decoding.

3.5.3.1.2 Locating Preamble

After bit synchronisation, a GPS navigation data is identified and the next step is to find the start. The preamble signifies the start of a GPS navigation message sub-frame and its length is 8-bits. The transmitted preamble pattern is often 10001011 but it can also appear inverted as 01110100, due to the Costas loop's ability to also track a 180° phase shifted signal (Borre *et al.*, 2007: 110). Usually, these two 8-bits patterns can transpire anywhere in the received data (and in alternating versions) so a further check must be performed to validate this preamble. The authentication procedure is often correlation, which checks if the same preamble is iterated every 6s, corresponding to the transmissions time between two successive sub-frames (see Section 2.5.3; Figures 2.7-2.9). A correlation result of 8 or -8 signifies a valid preamble, as well as a valid data.

3.5.3.1.3 Extracting Navigation Message

Realising valid preambles identifications mark the beginning of a navigation data sub-frame. The preamble is located in the TLM (the first word), followed by the HOW (see Figures 2.8 and 2.9 for details). Parity bits are used to check whether the received transmitted GPS navigation data has been decoded correctly. The GPS L1 C/A signal parity check procedure often involves a series of modulus-2 additions of all 30 data bits in a sub-frame. Navstar IS-GPS-200H (2013: 136 - 138) specification, explains in detailed, the parity encoding and decoding procedure. Transmission time refers to when the current received GPS sub-frame was transmitted and this is calculated from the Time Of Week (TOW). The TOW truncated version is termed the Z-count. Figure 3.10 depicts the TOW and Z-count – the Z-count is the elapsed number of seconds since the last GPS week rollover in units of 1.5s. The maximum value of the Z-count is 403199 – since 604800s makes a week and dividing this by 1.5 gives 403200. The Z-count value in the HOW is an abridged version having only the 17 Most Significant Bits (MSB). This reduction makes the Z-count increment in 6s steps, corresponding to the time between transmissions of two successive navigation sub-frames. The truncated Z-count value in the HOW, corresponds to the transmission time of the subsequent navigation data sub-frame. To obtain the transmission time of the current sub-frame, the truncated Z-count should be multiplied by 6 and 6s should be deducted from the result (Borre *et al.*, 2007: 113 & 114). Figure 3.10 illustrates the GPS time.

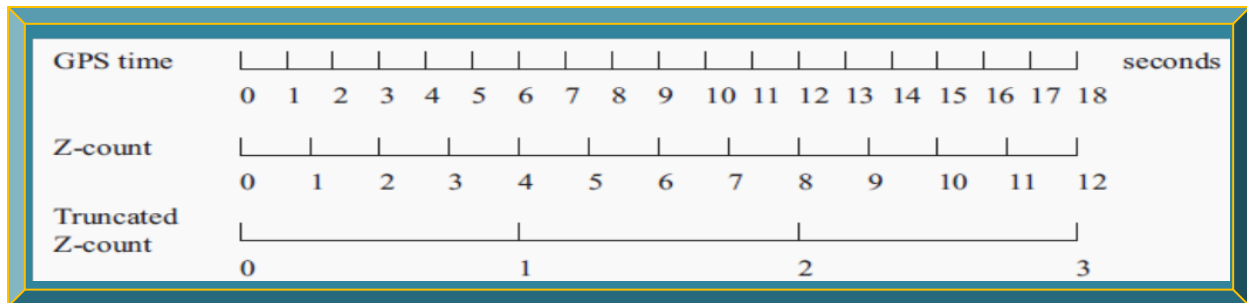


Figure 3.10: GPS time, Z-count and truncated Z-count
 (Adapted from Borre *et al.*, 2007: 113)

The ephemeris, almanac, ionosphere and UTC parameters are as well extracted according to the procedures described in ARINC IS-GPS-200D (2004) and / or Navstar IS-GPS-200H (2013).

Summarily, sub-frame 1 is first decoded and the most vital parameter to be extracted is the clock correction, followed by sub-frames 2 and 3 ephemeris parameters (use to calculate SVs position) and next is sub-frame 4 UTC/ ionosphere parameters and finally sub-frame 5 almanac parameter.

3.5.3.2 Processing

Processing is the interpretation and use of the decoded GPS navigation data. The ultimate use of the recovered GPS navigation data is to compute the receiver PVT (Borre *et al.*, 2007: 114-135).

Contrary to decoding which follows a strict sequential procedure governed by the GPS navigation message specifications, GPS data processing depends on the PVT solution used. Usually, GPS satellites position is first determined, followed by pseudo-range computations and finally GPS receiver position calculations; from which velocity and time can be further found.

3.5.3.2.1 GPS Satellites Position

GPS satellites positions in space are described by Keplerian parameters and must be transformed to Earth-Centered Earth-Fixed (ECEF) coordinates on Earth. Determining GPS satellites position entails using the ephemeris data in sub-frames 2 and 3 recovered from the GPS navigation data message. Ephemeris data is specific to a particular SV and is valid between 4 to 6 hours after decoding. Ephemeris data as well constitutes updated history of a GPS satellite orbit (it describes the orbit a GPS satellite is moving in). This together with Keplerian parameters, are used to accurately determine or estimate the current position of any GPS satellite. Figure 3.11 depicts the six Kepler's orbital elements and for details pertaining to their usage in SV position calculation, see Tsui (2005: 51 - 67); Borre *et al.* (2007: 114 - 119); Navstar IS-GPS-200H (2013: 101 - 105).

In Figure 3.11, center of Earth is denoted as C , the ascending node K and the true/mean anomaly as f/μ . Tables 3.7-3.10 show the standard ephemerides (ephemeris plural) and orbital parameters.

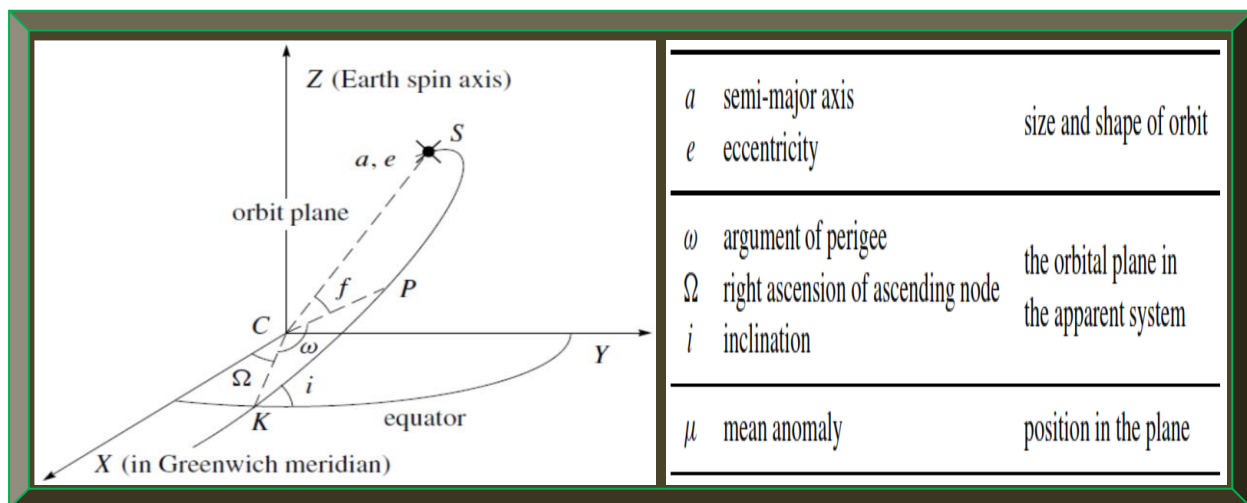


Figure 3.11: Kepler's orbital parameters
(Adapted from Strang & Borre, 1997: 483 & 485)

Table 3.7: GPS satellites ephemeris parameters definition

(Adapted from Navstar IS-GPS-200H, 2013: 102)

Ephemeris Data Definitions	
M_0	Mean Anomaly at Reference Time
Δn	Mean Motion Difference From Computed Value
e	Eccentricity
\sqrt{A}	Square Root of the Semi-Major Axis
Ω_0	Longitude of Ascending Node of Orbit Plane at Weekly Epoch
i_0	Inclination Angle at Reference Time
ω	Argument of Perigee
$\dot{\Omega}$	Rate of Right Ascension
IDOT	Rate of Inclination Angle
C_{uc}	Amplitude of the Cosine Harmonic Correction Term to the Argument of Latitude
C_{us}	Amplitude of the Sine Harmonic Correction Term to the Argument of Latitude
C_{rc}	Amplitude of the Cosine Harmonic Correction Term to the Orbit Radius
C_{rs}	Amplitude of the Sine Harmonic Correction Term to the Orbit Radius
C_{ic}	Amplitude of the Cosine Harmonic Correction Term to the Angle of Inclination
C_{is}	Amplitude of the Sine Harmonic Correction Term to the Angle of Inclination
t_{oe}	Reference Time Ephemeris (reference paragraph 20.3.4.5)
IODE	Issue of Data (Ephemeris)

Table 3.8: GPS satellites ephemeris parameters value

(Adapted from Navstar IS-GPS-200H, 2013: 103)

Ephemeris Parameters				
Parameter	No. of Bits**	Scale Factor (LSB)	Effective Range***	Units
IODE	8			
C_{rs}	16*	2^{-5}		meters
Δn	16*	2^{-43}		semi-circles/sec
M_0	32*	2^{-31}		semi-circles
C_{uc}	16*	2^{-29}		radians
e	32	2^{-33}	0.03	dimensionless
C_{us}	16*	2^{-29}		radians
\sqrt{A}	32	2^{-19}		$\sqrt{\text{meters}}$
t_{oe}	16	2^4	604,784	seconds
C_{ic}	16*	2^{-29}		radians
Ω_0	32*	2^{-31}		semi-circles
C_{is}	16*	2^{-29}		radians
i_0	32*	2^{-31}		semi-circles
C_{rc}	16*	2^{-5}		meters
ω	32*	2^{-31}		semi-circles
$\dot{\Omega}$	24*	2^{-43}		semi-circles/sec
IDOT	14*	2^{-43}		semi-circles/sec

* Parameters so indicated shall be two's complement, with the sign bit (+ or -) occupying the MSB;
 ** See citation for complete bit allocation in subframe;
 *** Unless otherwise indicated in this column, effective range is the maximum range attainable with indicated bit allocation and scale factor.

Table 3.9: GPS orbital elements parameters definition
(Adapted from Navstar IS-GPS-200H, 2013: 104)

Elements of Coordinate Systems (sheet 1 of 2)	
$\mu = 3.986005 \times 10^{14} \text{ meters}^3/\text{sec}^2$	WGS 84 value of the earth's gravitational constant for GPS user
$\dot{\Omega}_e = 7.2921151467 \times 10^{-5} \text{ rad/sec}$	WGS 84 value of the earth's rotation rate
$A = (\sqrt{A})^2$	Semi-major axis
$n_0 = \sqrt{\frac{\mu}{A^3}}$	Computed mean motion (rad/sec)
$t_k = t - t_{oe}^*$	Time from ephemeris reference epoch
$n = n_0 + \Delta n$	Corrected mean motion
$M_k = M_0 + nt_k$	Mean anomaly
$M_k = E_k - e \sin E_k$	Kepler's Equation for Eccentric Anomaly (may be solved by iteration) (radians)
$v_k = \tan^{-1} \left\{ \frac{\sin v_k}{\cos v_k} \right\}$	True Anomaly
$= \tan^{-1} \left\{ \frac{\sqrt{1-e^2} \sin E_k / (1-e \cos E_k)}{(\cos E_k - e) / (1-e \cos E_k)} \right\}$	
* t is GPS system time at time of transmission, i.e., GPS time corrected for transit time (range/speed of light). Furthermore, t_k shall be the actual total time difference between the time t and the epoch time t_{oe} , and must account for beginning or end of week crossovers. That is, if t_k is greater than 302,400 seconds, subtract 604,800 seconds from t_k . If t_k is less than -302,400 seconds, add 604,800 seconds to t_k .	

Table 3.10: GPS orbital elements parameters definition continued
(Adapted from Navstar IS-GPS-200H, 2013: 105)

Elements of Coordinate Systems (sheet 2 of 2)	
$E_k = \cos^{-1} \left\{ \frac{e + \cos v_k}{1 + e \cos v_k} \right\}$	Eccentric Anomaly
$\Phi_k = v_k + \omega$	Argument of Latitude
$\delta u_k = c_{us} \sin 2\Phi_k + c_{uc} \cos 2\Phi_k$	} Second Harmonic Perturbations
$\delta r_k = c_{rs} \sin 2\Phi_k + c_{rc} \cos 2\Phi_k$	
$\delta i_k = c_{is} \sin 2\Phi_k + c_{ic} \cos 2\Phi_k$	
$u_k = \Phi_k + \delta u_k$	Corrected Argument of Latitude
$r_k = A(1 - e \cos E_k) + \delta r_k$	Corrected Radius
$i_k = i_0 + \delta i_k + (\text{IDOT}) t_k$	Corrected Inclination
$\left. \begin{aligned} x_k' &= r_k \cos u_k \\ y_k' &= r_k \sin u_k \end{aligned} \right\}$	Positions in orbital plane.
$\Omega_k = \Omega_0 + (\dot{\Omega} - \dot{\Omega}_e) t_k - \dot{\Omega}_e t_{oe}$	Corrected longitude of ascending node.
$\left. \begin{aligned} x_k &= x_k' \cos \Omega_k - y_k' \cos i_k \sin \Omega_k \\ y_k &= x_k' \sin \Omega_k + y_k' \cos i_k \cos \Omega_k \\ z_k &= y_k' \sin i_k \end{aligned} \right\}$	Earth-fixed coordinates.

3.5.3.2.2 GPS Satellites to Receiver Pseudo-range

Pseudo-range is the ballpark distance between orbiting GPS satellites and a GPS receiver. The received GPS signal time will differ from the time it was transmitted, due to error sources. This difference in time is known as time bias and accounts for all pseudo-range measurements. However, if there is no time difference (bias error = 0), the pseudo-range equals the range (actual distance). In Borre and Strang (2012: 9), 1ms timing bias translates to ~300km error in distance.

The pseudo-ranges of each GPS satellite are obtained by multiplying the speed of light with the time taken for each GPS satellite transmitted signal to reach a GPS receiver. In Borre *et al.* (2007: 119 - 121), computing pseudo-range is best achieved in two steps: (i) finding the initial set of pseudo-ranges and (ii) use the initial set to estimate or keep track of future pseudo-ranges. Figure 3.12 depicts pseudo-range concept involving GPS sub-frames received from 4 GPS SVs.

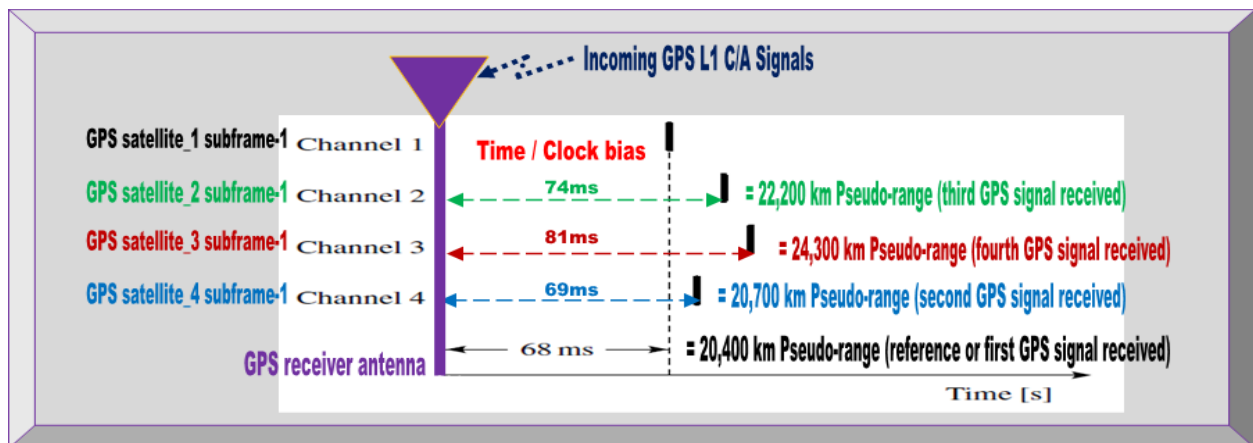


Figure 3.12: GPS clock bias and pseudo-range illustration
(Adapted from Borre *et al.*, 2007: 120)

GPS signals travel time from orbiting GPS satellites to Earth varies between 65 and 83ms. It is used to set the initial pseudo-range. GPS SVs closest to Earth normally have the earliest arriving sub-frames. From Figure 3.12, GPS satellite 1 (Channel 1) has the earliest travel time of 68ms and it is used as the reference to compute the travel times of the rest (Borre *et al.*, 2007: 119-121).

The GPS receiver position can be determined from the initial pseudo-ranges and the result is the GPS receiver position (X, Y, Z) with a clock offset of dt . The clock offset is used to fine-tune the time of travel of the reference satellite (i.e. 68ms of GPS satellite 1) and with this procedure, the GPS receiver can determine the actual pseudo-ranges after two calculations by: i) taking note of the difference in ms between the start of the sub-frame with respect to the reference GPS satellite and ii) the start of the C/A code that gives the exact pseudo-range for the orbiting GPS satellites.

3.5.3.2.3 GPS Receiver Position

From SVs positions and pseudo-range measurements, the GPS receiver position is computed as a set of XYZ coordinates, which are respectively transformed to latitude, longitude and altitude. Further optimisation is done to reduce error sources. See Sections 3.6, 3.7 and also Appendix B.

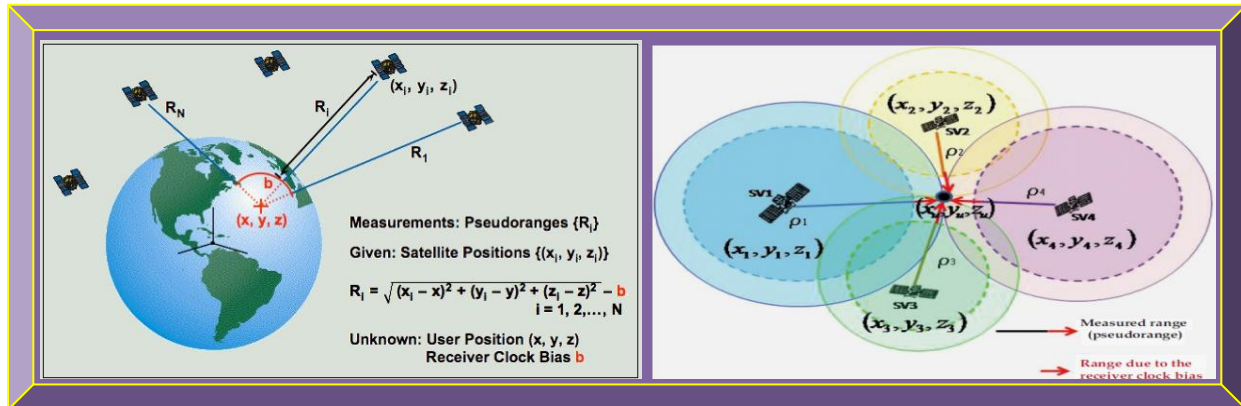


Figure 3.13: Pseudo-ranges, SV, GPS receiver position and clock bias illustration
(Adapted from Misra & Enge, 2006; Jin, 2012)

Figure 3.13 portrays the relationship between pseudo-ranges, SV position, GPS receiver position and clock (time) bias. See Jin (2012: 107 - 125) for details. Two main methods are used for calculating GPS receiver position, namely i) least squares and ii) the Kalman filter techniques.

Least Squares:

In Strang and Borre (1997: 448, 458), this approach outputs a basic single point positioning, whereby a set of linearised equations are solved to get approximate distances using trilateration and not triangulation (yields angles not distance). The approximated residuals are defined as the difference between the real observations and the new estimated model for the observations. The least-squares solution (covered in Appendix B, Section 8.2.7.5.2) is then obtained by changing the initial position estimate value, until the sum of squares of the approximated residuals are minimised. If some observations are considered less accurate than others, then the Weighted Least Squares (WLS) approach is used to address the in-accuracies. Refer to Grewal *et al.* (2001: 26 & 27); Tsui (2005: 63 - 67); Gleason and Gebre-Egziabher (2009: 56 - 64) for more details.

Kalman Filter:

It is an advanced method, employed in positioning. It uses a recursive algorithm to give optimum predictions of GPS receiver PVT, based-on current measurements and noise statistics. The filter algorithm contains dynamic model of the GPS receiver motion and uses it to output a set of GPS receiver PVT state estimates and related error variances (Gleason & Gebre-Egziabher, 2009: 67).

3.5.3.2.4 GPS Receiver Velocity

Similar to position, GPS receiver velocity can be estimated from various approaches. In Kaplan and Hegarty (2006: 58 - 61), these approaches include (i) approximate differentiation of position (ii) use of least-squares on Doppler's shift data or frequency from the carrier loop (iii) use of carrier phase measurements or cycles to determine how many cycles had occurred since signal reception and (iv) Kalman filter state vector. Refer to Grewal *et al.* (2001: 28); Xu (2007: 212); Gleason and Gebre-Egziabher (2009: 64 & 65). GPS receiver velocity calculation using method (ii) is analysed in Appendix B, Section 8.2.7.6. Figure 3.14 previews the processes involved.

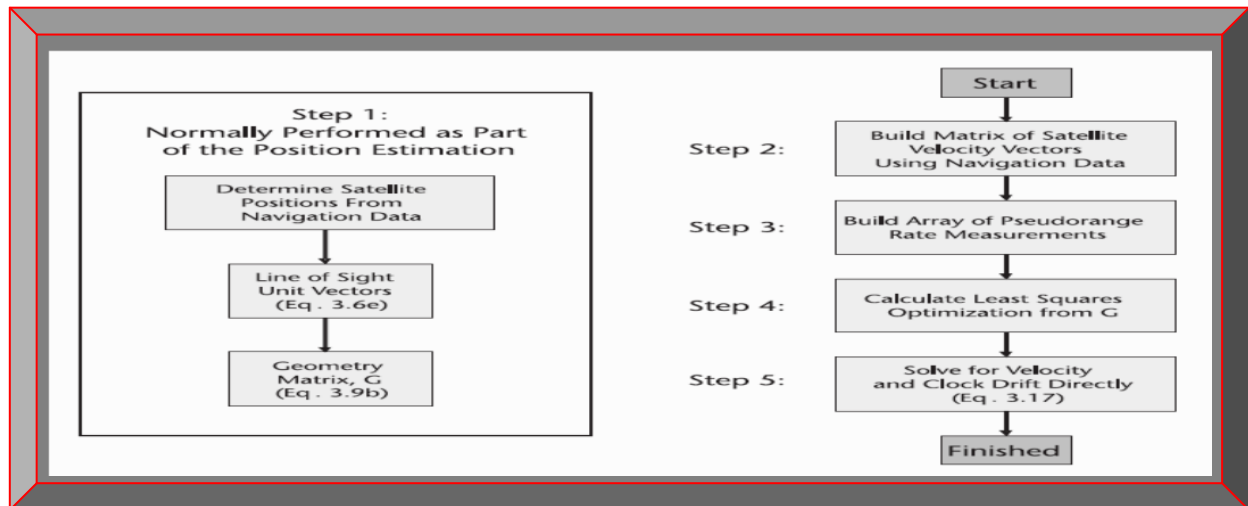


Figure 3.14: GPS receiver velocity calculation
(Adapted from Gleason & Gebre-Egziabher, 2009: 65)

3.5.3.2.5 GPS Receiver Time

With GPS receiver clock bias already deduced from computing the position, what is required is to determine the absolute time to which this bias value is referenced. This is achieved using data in the recovered GPS satellite's navigation message – from which there is a GPS timestamp for each transmitted sub-frame. This information is primarily used by the GPS receiver to make pseudo-range measurements, as well as to coarsely set GPS receiver reference clock. The precise GPS time when the GPS receiver latched the pseudo-range and Doppler measurements can be expressed as a function of signal reception time and clock bias. GPS time is expressed in the form of GPS week and by definition, it is exactly 604800s/week, and rolls over to 0 once per week (Saturday breaking Sunday). Using suitable formulas, GPS time can be converted to UTC (GMT) or any local time. GPS time differs from UTC by integer number of seconds which often drifts with time since 1980 (Borre & Strang, 2012: 260 - 262). Refer to Appendix B for details.

3.6 GPS Errors Sources

GPS technology is prone to a variety of errors sources which degrades the performance of various aspects of its operation – manifested at the receiver. These error sources have been the focus of researchers and several solutions have been proposed and implemented within the last number of years. This section focuses on GPS sources of errors known to have or currently affects GPS technology, the details of the error sources, their effects, as well as corresponding possible mitigation techniques. Due to the size of the study, some of these mitigation techniques, where applicable, are described in Section 3.7 and mathematically analysed in Appendix B.

The errors sources have been classified according to the GPS technology aspect they affect:

GPS Ground Station Associated Errors / Sources

- Selective Availability (SA) – is permanently disabled since May 2000.
- Human mistakes.
- Equipment and device related.

GPS Satellites Associated Errors / Sources

- GPS satellites unavailability.
- GPS satellites ephemeris parameter errors.
- GPS satellite geometry: Dilution of Precision (DOP) errors.
- GPS satellites clock errors.

GPS Signal Associated Errors / Sources

- Ionospheric propagation or scintillation effects.
- Tropospheric propagation effect.
- Multipath effect.
- Doppler's frequency shift effect.
- Jamming and spoofing of signals.

GPS Receiver Associated Errors / Sources

- GPS receiver clock errors.
- GPS receiver's noise.
- Software errors.
- Antenna phase centre offset and variation.

3.7 GPS Errors Sources Mitigation

This section briefly discussed the GPS error sources in Section 3.6 and proposed likely solutions. Dana (2000); Grewal *et al.* (2001: 103 - 130); El-Rabbany (2002: 27 - 44); Kalyanaraman *et al.* (2004); Kaplan and Hegarty (2006: 301 - 321); Xu (2007: 43 - 85); GPS.Caltech (2009: 16 - 50); Gleason and Gebre-Egziabher (2009: 55 - 115); Jin (2012: 357 - 422), examined all the details.

3.7.1 GPS Ground Station Associated Errors / Sources

These errors originate at the GPS ground control segments and could affect the entire GPS.

3.7.1.1 Selective Availability

The intentional degrading of GPS satellites clocks and ephemeris data could be performed by the US Department of Defence for military reasons. This in-turn degrades GPS signals which are noticeable by the GPS receiver poor performance (e.g. high in-accuracies in PVT calculations). Although selective availability was permanently stopped in May 2000 and no longer a GPS error source, several methods including that in Xiong *et al.* (2000), were researched to mitigate it.

3.7.1.2 Human Mistakes

In Gleason and Gebre-Egziabher (2009: 105 &106), un-intentional blunders such as wrong data feeds or updates, are caused by staff working at GPS ground control stations. However, current measures make the probability of such errors happening and going viral or un-noticed minimal.

3.7.1.3 Equipments / Devices Related Errors

This is as a result of machine(s) malfunctions and could either incorrectly performs its functions or performs no function at all. Contrary to human errors, the possibility of these errors happening is certain; however, control and backup systems often minimise the propagation of these errors.

Conclusively, errors that might originate from GPS ground control segments are minimal, though the effect of such errors should they occur, could be huge as they could affect the entire GPS.

3.7.2 GPS Satellites Associated Errors / Sources

These are errors that affect or originate from GPS satellites. Such errors could be intentional or un-intentional and could directly or in-directly affect GPS signals and therefore GPS receivers.

3.7.2.1 Number of Visible or Available GPS Satellites

The more GPS SVs a GPS receiver can see, the faster the acquisition and better PVT accuracies.

3.7.2.2 Ephemeris Parameters Errors

In El-Rabbany (2002: 28 & 29), these are in-accuracies in the ephemeris data from GPS satellites and as a result, the calculated GPS satellite's position based on such data might be in-accurate to <10m. Usually, the four monitoring stations of the ground control segment, estimates the orbital parameters of all GPS satellites in space and the result is then processed by the GPS master control station of the GPS ground control segment to determine new ephemeris data. The newly processed ephemeris parameters, together with other navigation data, are sent back to three of the four monitoring stations for upload to respective orbiting GPS satellites. Ephemeris errors from each GPS satellites are transmitted in the navigation message and upon message recovery by the GPS receiver; these errors are used to correct GPS satellites positions during PVT computations.

3.7.2.3 GPS Satellite Geometry (DOP) Errors

As illustrated in Section 2.4, GPS satellites-receiver geometry does have a considerable impact on the accuracy of obtained PVT estimates. In other words, some GPS satellites-to-receiver geometries will result to better PVT solutions than others. Therefore, it is vital to have a way of comparing different GPS satellites-to-receiver geometries and the metric normally used for measuring this impact is DOP. It simply represents the degree to which GPS satellites-to-receiver geometry dilutes the accuracy of the PVT solutions at the time of computation. Computed DOP values are usually by-products of PVT calculations and their provisions can be used to improve the computed PVT solutions, if they are extremely in-accurate (Kaplan & Hegarty, 2006: 302).

3.7.2.4 GPS Satellites Clock Errors

In El-Rabbany (2002: 31 & 32), each GPS satellite is equipped with four on-board ultra-high two Cesium and two Rubidium precision atomic clocks. These clocks directly control the timing of GPS signals transmission. Besides their high accuracy, precision, reliability and robustness; they are still prone to errors (slight timing drifts) which must be corrected to enable easy synchronisation of the time in all orbiting GPS SVs. GPS SVs clocks are allowed to relatively drift to some extent, so that they can be estimated by ground control stations to process clock corrections data coefficients for respective GPS satellites. These clock errors are sent back to the respective GPS SVs via subsequent navigation data uploads, which are in turn transmitted back by GPS SVs in the navigation data messages and upon recovery by GPS receivers, the clock corrections are used when computing PVT. These corrections are good but not perfect and will result in small errors in the computed PVT. Thus, to improve this GPS SVs clock error estimates, sophisticated technique like differencing, can be used to mitigate inaccuracies in the PVT result.

3.7.3 GPS Signal Associated Errors / Sources

In the context of this document, these are any sources of errors or errors that directly or indirectly negatively affect the propagation of GPS signals from orbiting GPS satellites to GPS receivers.

3.7.3.1 Ionospheric Propagation or Scintillation Effect

The ionosphere is the region of Earth atmosphere stretching from 50 to 1000 km above sea level. The electron density or Total Electrons Content (TEC) in this region of the atmosphere is unevenly distributed, due to fluctuations depending on the time of the day and space weather conditions. This random intermittent variation in TEC is known as scintillation effect. TEC concentration affects the propagation of GPS L1 C/A RF signals via the ionosphere in two ways: i) group delay which is manifested as a longer pseudo-range and ii) phase advance which is manifested as a shorter carrier phase. Ionospheric effects are currently one of the main sources of errors to GPS signals. The amount of GPS signal delays caused by the ionosphere is transmitted in the GPS navigation message and after navigation message recovery, it is used to correct timing delays when computing GPS receiver PVT. Figure 3.15 shows maps of worldwide TEC distribution comparison at night and day and subsequent associated pseudo-range errors.

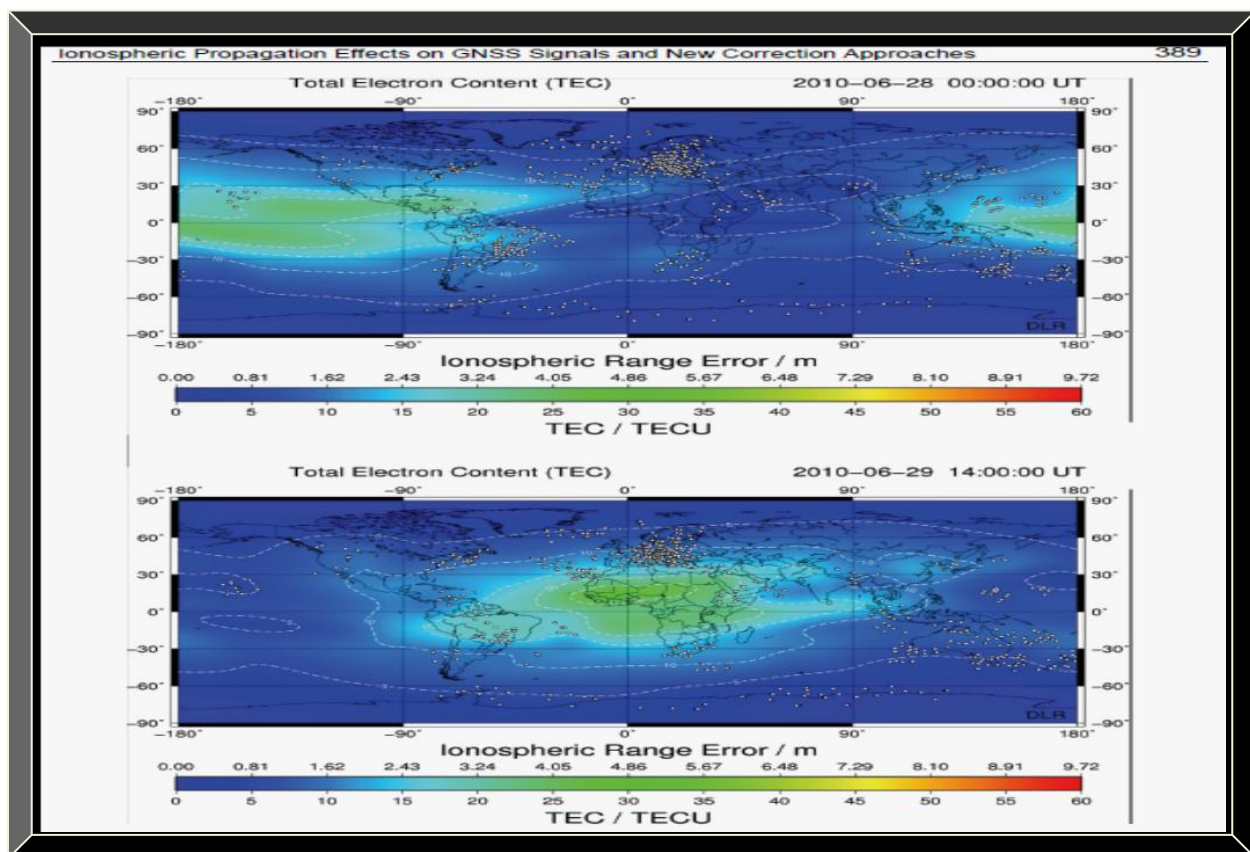


Figure 3.15: Global TEC maps and corresponding ionospheric range errors
(Adapted from Jin, 2012: 389)

3.7.3.2 Tropospheric Propagation Effect

The troposphere is the section of the atmosphere below the ionosphere and it consists of a hydrostatic (dry) and wet portions accounting for water vapour. The dry portion accounts for 90% of tropospheric refraction, while the wet portion accounts for 10%. The models for the wet troposphere are less accurate than those for the dry troposphere and as a result, the wet troposphere errors, though minimal, could have a greater effect on the pseudo-range bias than those of the dry troposphere. Tropospheric delay errors are proportional to temperature, pressure and humidity and can be mitigated by assigning an elevation mask for the GPS receiver; whereby, GPS L1 C/A signals from low elevation GPS satellites could be avoided. With a 15° elevation mask, 5 to 8 GPS satellites can be simultaneously observed from any location on Earth at any point in time. Figure 3.16 illustrates the entire Earth atmosphere and the effective sections.

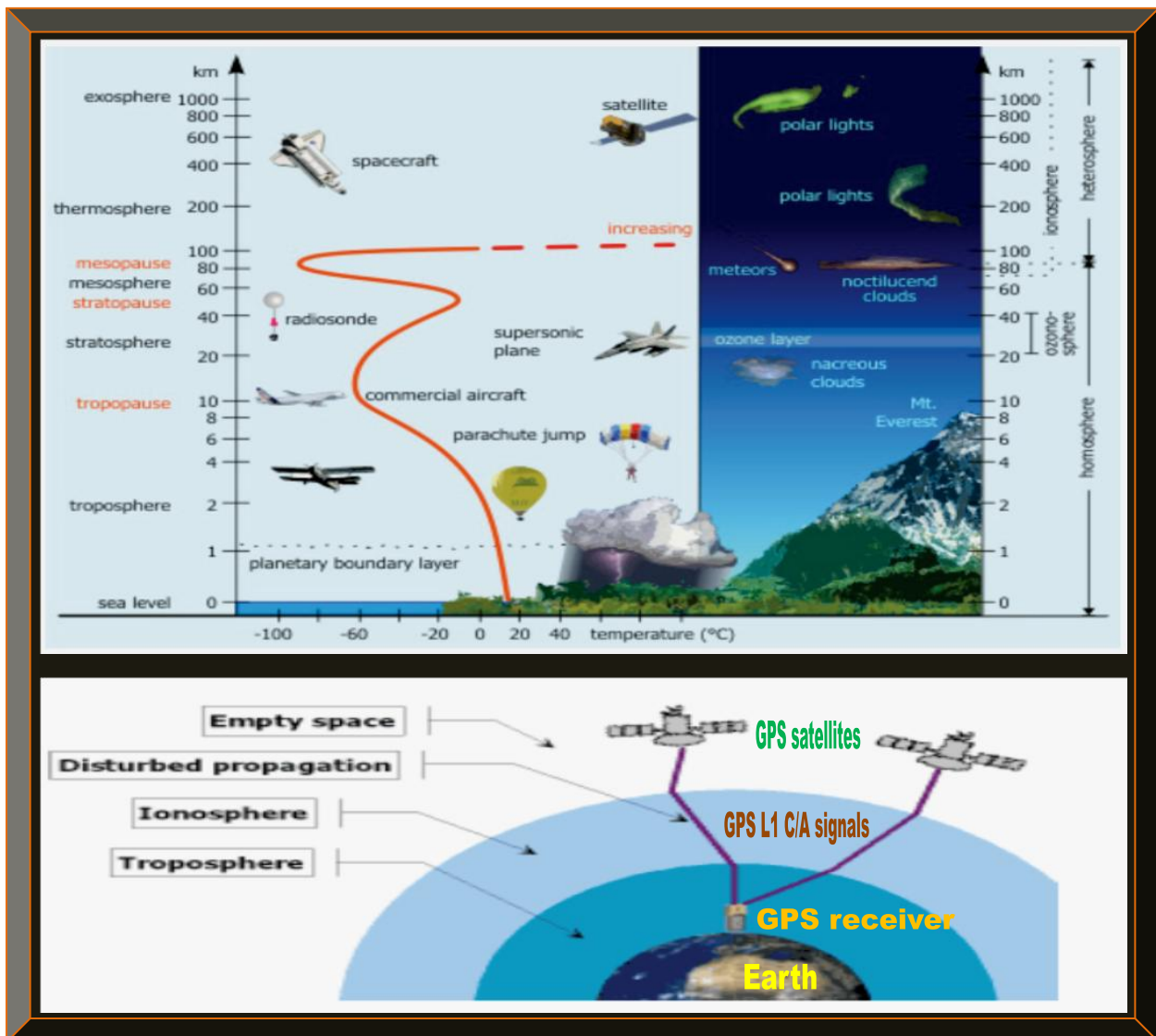


Figure 3.16: Atmosphere with focus on tropospheric effect
(Adapted from GPS.Caltech, 2009: 20 & 21)

3.7.3.3 Multipath Effect

Multipath effect is when there is a change in the direction of propagation of a signal as a result of obstruction in the signal path. The reflected signal(s) from different paths are then mixed with the original direct signal creating a resultant signal – distorted in phase / amplitude. Multipath error manifest when this distorted resultant GPS signal, then enters the GPS receiver processing blocks via the GPS receiver antenna. Multipath is a chief error source for both the carrier-phase and especially pseudo-range measurements. Multipath errors can be quite significant in severe multipath surroundings (e.g. in urban areas) especially in static application such as surveying.

Multipath errors are grouped into two types, namely static and dynamic. The static kind is more serious and could last for minutes since the GPS receiver will be stationery or moving slowly. Multipath associated with high-dynamics GPS receivers onboard a CubeSat could be less severe, since such errors could last only fractions of a second, especially in the case of a CubeSat with small form-factor orbiting in free space at extreme speed. In Grewal *et al.* (2001: 115 - 125), various multipath errors mitigation techniques, especially for static GPS receivers have been researched but this research focuses on high-dynamics GPS receiver and such techniques will not be discussed. Multipath errors in high-dynamics GPS receivers can be alleviated by GPS receiver antenna design; whereby, in the case of GPS signals, a RHCP antenna is normally used, since the transmitted signal is RHCP. If the incoming GPS signal is reflected once or thrice, it becomes LHCP and is filtered out by a RHCP antenna but if it is reflected twice or in even successions (2, 4, 6, 8... times) it reverts to RHCP signal and is filtered through by a RHCP antenna (Manandhar *et al.*, 2006). As a result of this weakness, a RHCP antenna approach is complemented by other multipath mitigation techniques during acquisition and tracking. Figure 3.17 portrays multipath.

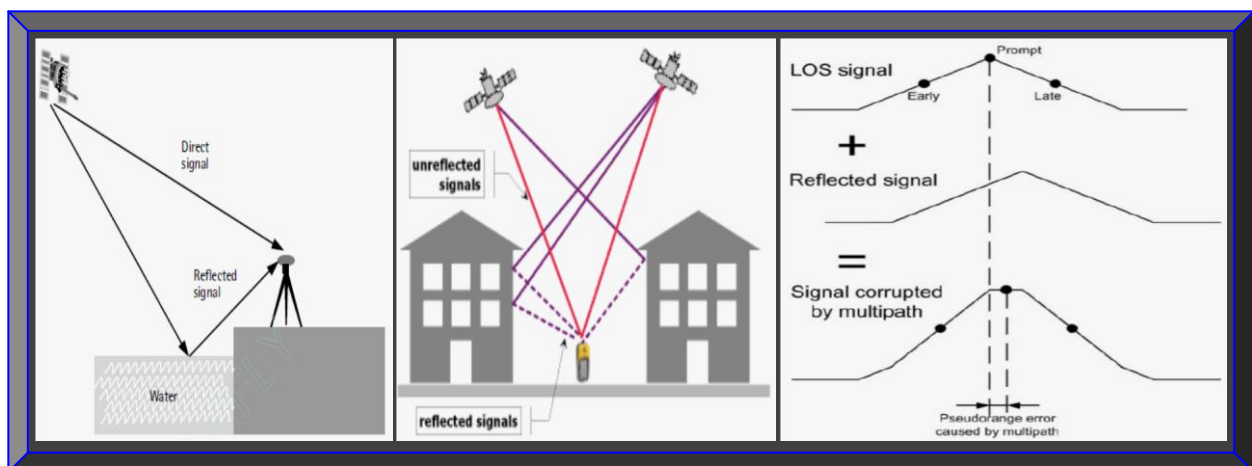


Figure 3.17: Depiction of multipath and its effects on received GPS signal
(Adapted from El-Rabbany, 2002: 33; Gleason & Gebre-Egziabher, 2009: 298; GPS.Caltech, 2009: 31)

3.7.3.4 Doppler's Frequency Shift Effect

Doppler's effect or simply Doppler, in satellite telecommunication and in this regard GPS, is the change in carrier frequency and or PRN code phase shift of the GPS signal, as a result of relative motion between GPS satellites and GPS receiver(s). According to Tsui (2005: 37), Doppler frequency shift that is caused by orbiting GPS satellites motion, on both the coarse acquisition C/A code and on the GPS carrier frequency, varies from $\sim\pm 5$ kHz in a stationary /low speed GPS receiver, to typically $\sim\pm 10$ kHz for a high-speed GPS receiver on-board an aircraft or sports car. In Gleason and Gebre-Egziabher (2009: 330), GPS Doppler shift could reach up to ± 40 kHz for a GPS receiver on-board a LEO satellite. Doppler's effect affects the acquisition time and lock state of a GPS receiver; however, it is often mitigated by performing a frequency sweep across the Doppler value range at a defined rate depending on the value. For instance, for a ± 50 kHz Doppler, a carrier frequency sweep from -50 kHz to $+50$ kHz at a rate or step of usually between 100 Hz to 500 Hz is used during GPS signal acquisition process. For details, see Tsui (2005: 34 - 40); Borre *et al.* (2007: 77); Borio *et al.* (2010). Figure 3.18 depicts Doppler's effect in GPS.

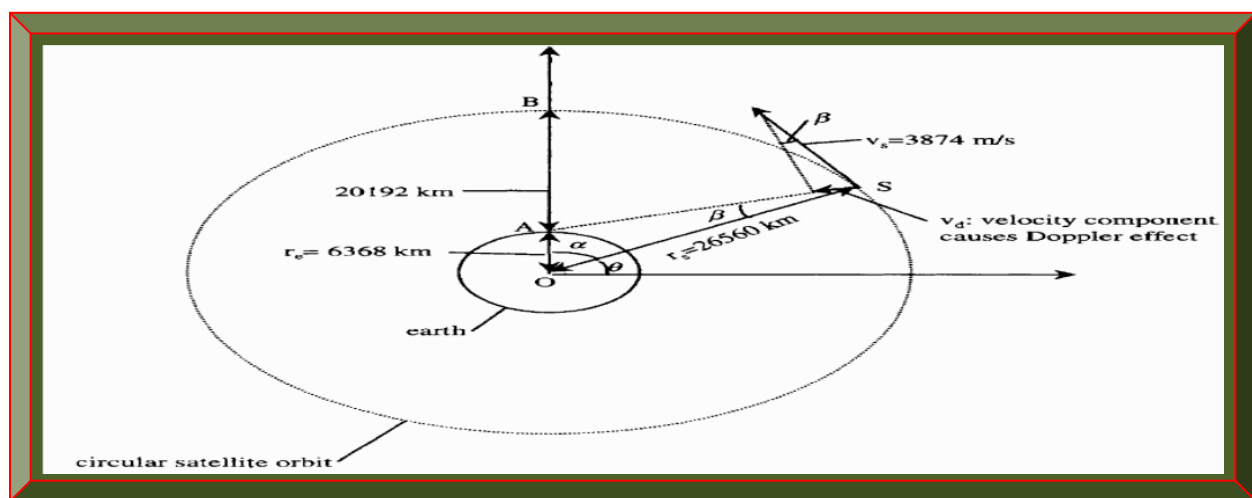


Figure 3.18: Illustration of Doppler frequency caused by satellite motion
(Adapted from Tsui, 2005: 35)

3.7.3.5 Jamming / Spoofing Signals Effect

According to Xu (2007: 82), a jamming or spoofing signal – from a jammer or spoofer (contrary to a pseudolite – a false signal from a device use to generate GPS-like signals similar to those transmitted from actual orbiting GPS satellites), is one that interferes or mimics real GPS signals, to prevent GPS receivers from acquiring and tracking actual incoming GPS signals. A jamming signal is more powerful than actual GPS signals and usually come from low elevation angles. This can be allayed by noting GPS signal strengths and acquiring it from elevation angles $>15^\circ$.

3.7.4 GPS Receiver Associated Errors / Sources

These are error sources which internally or externally affect the performance of a GPS receiver.

3.7.4.1 GPS Receiver Clock Errors

In El-Rabbany (2002: 31), GPS receivers' built-in clock is not as accurate as the atomic clocks used in GPS satellites. GPS receivers clocks are usually made of inexpensive quartz crystal with in-accuracies of <10ppm (ppm: parts per million) which often drift with time and temperature – causing slight timing errors. As a result, GPS receivers clock errors are much more than those of GPS SVs. Similar to GPS SVs clock errors, milli-seconds timing in-accuracies often translates to hundreds of km range errors. GPS receiver clock errors are mitigated by including or treating it as an extra error (unknown) parameter, when computing the navigation PVT solution; whereby, it is subtracted with GPS SVs clock error. A better mitigation technique is to formulate a suitable mathematical model of the crystal clock error and incorporate it in a Kalman filter state vector.

3.7.4.2 GPS Receiver's Noise

Errors from GPS receiver noise, often arise in the RF front-end and DSP stages but may also be as a result of poor design. GPS receiver noise errors are easily addressed by refining the design.

3.7.4.3 Software Errors

These occur due to programming bugs in the acquisition, tracking or PVT algorithms, as well as usage of a wrong reference coordinate system or datum. Debugging usually mitigate such errors.

3.7.4.4 Antenna Phase Center Offset / Variation

As stated in El-Rabbany (2002: 34), the antenna phase center point is where the GPS signal is received. The geometric distance between the satellite at time of signal transmission and the receiver at time of signal reception, is the distance of the phase centers of both the GPS SVs and GPS receiver antennas. Usually, the antenna phase center does not matches with the physical or geometrical center of the antenna – as it changes relative to the azimuth and elevation of GPS satellites, as well as the strength of the incoming GPS signal. GPS receiver antenna phase center error size varies depending on the antenna type and is typically a few centimeters which are usually negligible in most practical applications that do not need centimeter position accuracy. Proper orientation and placement of GPS receiver antenna can alleviate phase center variations. For mathematical details and modeling of antenna phase centre offset, see Xu (2007: 82 - 86).

3.7.5 User Equivalent Range Error (UERE) Budget

In Grewal *et al.* (2001: 129), it is preferable to analyse the effect of each of GPS errors, by converting it into an equivalent range error experienced by a GPS receiver. Usually, errors from different sources do not have the same statistical models; however, a total rough error estimate can be obtained or determined by forming a root-sum-square of the UERE from all error sources.

Figure 3.19 compares a summary of different error budgets from various sources. The dominant error sources are ionospheric and multipath; however, the UERE for a space-borne GPS receiver will be different, since some of these errors are negligible or non-applicable. See Section 3.8.

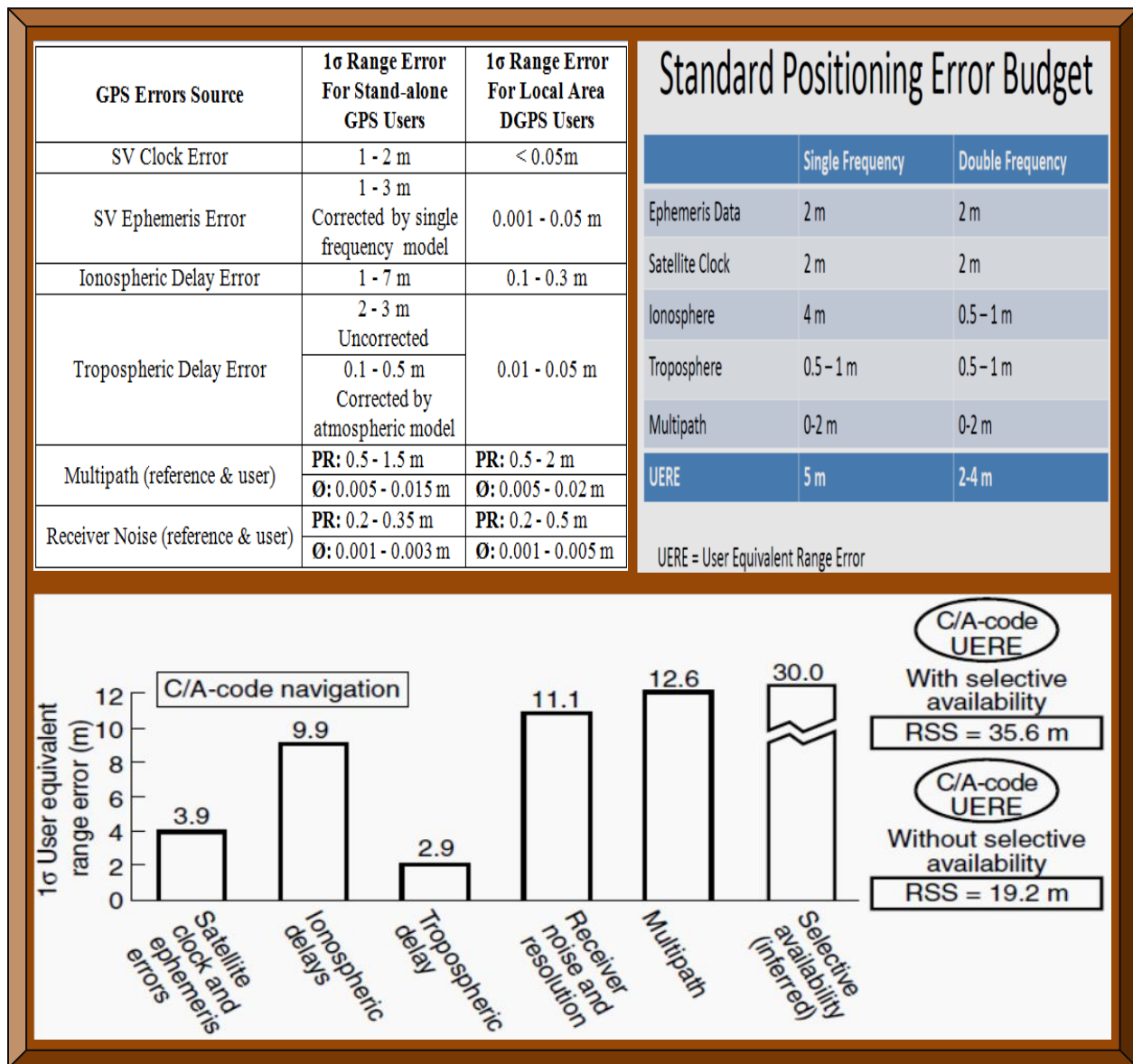


Figure 3.19: Various GPS L1 C/A UERE budget analyses

(Adapted from Grewal *et al.*, 2001: 129; Gleason & Gebre-Egziabher, 2009: 89; GPS.Caltech, 2009: 32)

3.8 Space-borne GPS Receiver Design Considerations

This section discusses various challenges faced by a GPS receiver on-board an orbiting CubeSat. A space-borne GPS receiver design considerations can be divided into two types, namely i) physical and ii) non-physical. Both of these are briefly addressed in the following sub-sections.

3.8.1 Design Considerations – Physical Challenges

Physical challenges in design considerations, in the context of this document refer to those factors or space environmental conditions with a damaging effect on a GPS receiver hardware (such as electronic devices / materials) and consequently leads to a malfunction or degradation in the performance of the GPS receiver in space. These factors include space radiation or ionisation, extreme temperature (very hot and very cold) and vacuum. Physical challenges are concisely discussed, since this research focus is on software and firmware-defined GPS receivers. For details regarding physical challenges faced by a space-borne GPS receiver, see Montenbruck *et al.* (2001); Montenbruck *et al.* (2002); Langley *et al.* (2004); Underwood *et al.* (2004).

3.8.1.1 Radiation

This poses the most threat, which includes ionisation, single event effects, upsets and latch-ups. Appropriate hardware radiation shielding and hardening are usually the mitigation methods used. Two types of checks are usually performed: i) total ionisation dose testing and ii) single effect event testing, which involves subjecting the GPS receiver hardware to a confined radiation test chamber and checking the GPS receiver components performances (Underwood *et al.*, 2004).

3.8.1.2 Extreme Temperatures

GPS receivers operating in space are subjected to extreme temperatures of high sun intensity and extreme coldness (during non-sunlight periods). There is also very sharp thermal cycling from extreme heat to cold and vice versa. Thus, operating in such conditions could damage or cause intermittent hardware devices malfunctions. As a result, unpowered and powered temperatures cycling are performed in a controlled environment (test chamber) at both extreme temperatures and the GPS receiver operation is monitored. In Glennon *et al.* (2011), extreme heat condition is mitigated by radiation heat transfer (space most ideal heat transfer means), heat-sinks and reflective structural surfaces. Extreme cold conditions can be alleviated by using micro heaters.

3.8.1.3 Vacuum

In vacuum, plastic devices usually out-gassed and the mitigation approach is conformal coating.

3.8.2 Design Considerations – Non-physical Challenges

This pertains to scenarios in space that do not cause physical damage to a GPS receiver but do negatively affect the operational performance (intermittent functioning, inefficiency and unreliability). This basically constitutes some of the sources of error already discussed in Section 3.7. The mitigation techniques are based-on using robust algorithms in the correlator (acquisition and tracking) and baseband processing (PVT computations) which are covered in Appendix A.

3.8.2.1 Doppler Frequency Shift

Coping with very high Doppler's shift is one of the most non-physical challenges encountered by space-borne GPS receivers. In Avansi and Tortora (2010); Lofgren (n.d), Doppler's shift could respectively extend in excess of $\pm 50\text{kHz}$ and $\pm 100\text{kHz}$. This high Doppler's shift tremendously affects GPS signal acquisition, tracking and PVT computations – since a GPS receiver on-board a LEO satellite (CubeSat) has very limited time to be in contact with a GPS satellite, due to the high relative motion involved between both satellites. Figure 3.20 illustrates a $\pm 40\text{kHz}$ Doppler on the RADCAL satellite in LEO. Mitigating Doppler's shifts requires implementing a robust acquisition, tracking and PVT algorithms – which are thoroughly discussed in Appendix A, as well as mathematically analysed in Appendix B and finally the implementation in Chapter 4.

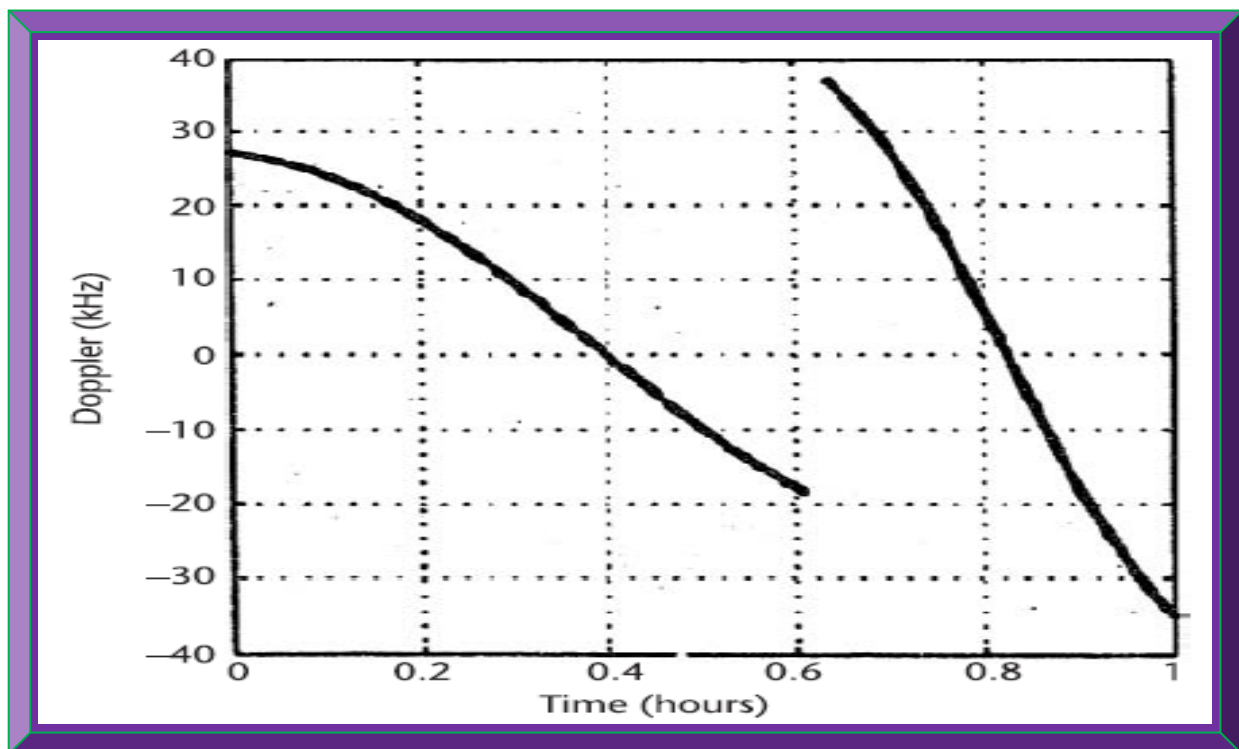


Figure 3.20: RADCAL satellite Doppler shift
(Adapted from Gleason & Gebre-Egziabher, 2009: 330)

3.8.2.2 GPS Orbital Geometry

The higher the altitude of a GPS receiver (on-board a CubeSat), the poorer its DOP and position accuracy. This is due to narrow field of view and unavailability of visible orbiting GPS satellites. Efficient acquisition, tracking and PVT estimates algorithms usually compensates for poor DOP.

3.8.2.3 Multipath Effect

Multipath could have little or no effect on a GPS receiver on-board a CubeSat, due to i) the relative speed involved between GPS satellites and receiver ii) the size and shape of a CubeSat. However, space debris and other satellites could reflect GPS signals causing multipath; thus, the possibility of its occurrence and appropriate mitigation approaches have to be as well considered.

3.8.2.4 GPS Receiver Antenna Orientation

Most CubeSat are not spin stabilised and for reliable communication between GPS satellites and a GPS receiver, their antennas must have direct Line Of Sight (LOS). A clumsy but effective mitigation approach is equipping each side of the CubeSat with a micro GPS antenna and linking them up internally via suitable RF impedance matching network connected to the GPS front-end.

3.8.2.5 GPS Signal Strength

Although the separation distance between GPS satellites and a GPS receiver on-board a CubeSat in space is less relative to GPS receivers on Earth, the direct LOS signal strength level of fluctuations in both cases is similar. Therefore, it is not always imperative to adjust the gain pattern of a space-borne GPS receiver antenna, since LEO altitude is within the region known as the terrestrial service volume (Gleason & Gebre-Egziabher, 2009: 333). Although the antenna gain pattern can be automatically controlled, adjusting it above the normal GPS signal power level might degrade the GPS receiver operation, if it has anti-jamming /anti-spoofing protections.

3.8.2.6 Atmospheric Effect

A GPS receiver on-board a CubeSat in LEO is subjected to ionospheric sources of error and little caused by the troposphere. The troposphere is <50km and has little effect on an orbiting CubeSat (>400 km). Therefore, only the ionosphere error model is relevant in the PVT estimates solution.

For more information on Section 3.8, see Yunck and Melbourne (1995); Willms (1999); Unwin and Oldfiel (2000); Langley *et al.* (2004); Montenbruck *et al.* (2005); Gleason and Gebre-Egziabher (2009: 329-335); Avanzi and Tortora (2010); Tsai *et al.* (2010); Glennon *et al.* (2011).

3.9 Summary

Chapter 3 presented the theoretical details of GPS receivers and how they can be adapted for space use. Past studies as highlighted in Section 3.3, revealed that, a software GPS receiver is the most viable option, due its flexibility and furthermore, most GNSS research and implementations published on IEEE websites, are based-on Matlab software. As a result, the GPS receiver being research will be software-defined based-on Matlab and Simulink, as well as the associated applicable toolboxes, such as HDL coder. A complete analysis of the signal processing chain in GPS receivers functional blocks, with emphasis on the acquisition and tracking algorithms (explained in-depth in Appendix A), as well as PVT computations were investigated and detailed in Appendix B. In Appendix A, the parallel code phase acquisition technique was shortlisted as the preferred choice, whereas the Costa PLL and DLL tracking algorithms were chosen as well, since their implementations have already been justified using Matlab software and are publicly available. Hence, developing a software-defined GPS receiver for space use could be facilitated using Matlab. Finally, associated GPS error sources, with Doppler's shift being the most crucial, as well as possible mitigation techniques were thoroughly examined to conclude Chapter 3.

CHAPTER 4

GPS RECEIVER DESIGN AND IMPLEMENTATION

4.1 Introduction

This chapter focuses on the design and implementation of the space-borne GPS receiver. The design specifications are put forward and form the basis of the proposed design. The algorithms utilised are a result of the technical analyses done in Chapters 3, as well as Appendices A and B. The algorithms to be implemented are mostly existing open-source GPS codes and will be modified where applicable (see Table 4.1) to meet the design specifications. The algorithm(s) will be implemented in three phases: initially as a software-defined GPS receiver based-on Matlab in Phase 1, as well as corresponding Simulink models, followed by translation to VHDL in Phase 2, using HDL Coder and Verifier and synthesis to firmware in Phase 3, using Altera Quartus II. Pre-testing and verifications of the different implemented algorithms models will be executed at each phase with relevant test vectors or test benches and the results will be analysed.

4.2 GPS Receivers Design Specifications

The design specifications are based-on the requirement analyses derived from the research objectives. The design specifications for the software GPS receivers and the firmware algorithms are different in terms of the implementations and are discussed in the following sections. It should be noted that, most of the software requirements are non-functional and cannot be tested.

4.2.1 The Software-defined GPS Receiver Design Specifications

The software design specification is based-on the primary objective, in-line with high-dynamics GPS algorithms. The implemented GPS receiver unit should conform to the following qualities:

- *Reliability*: it should always work correctly for problems defined within the research project.
- *Accuracy*: it should produce accurate results as guaranteed by the problem and input datasets.
- *Robustness*: it should work for hard problems and should provide information when it fails.
- *Efficiency*: it should execute in less time and consume less system resources such as memory.
- *Maintainability*: it should be comprehensive and modifiable by anyone with adequate skills.
- *Portability*: it should be adaptable with little or no change to new computing environments.
- *Usability*: it should have a convenient and user friendly interface with some helpful notes.
- *Applicability*: it should be able to address the research problems as well as similar ones.
- *Modularity*: it should consist of simple independent modules which can easily be integrated.

4.2.2 The Firmware-defined GPS Receiver Design Specifications

The firmware-defined GPS receiver design specifications are based-on the research problem statement and defined in-line with the secondary and tertiary objectives of the research project. The three main firmware-defined GPS receiver algorithms requirements are defined as follows:

- Should conform and validate the software-defined GPS receiver algorithms where possible.
- Process high-dynamics GPS C/A L1 signals with acceptable accuracy within 10m.
- Low-cost model(s).

4.3 GPS Receiver Algorithms Design and Implementation

The algorithms utilised are based-on the GPS mathematical equations that are discussed and analysed in Appendix B. These algorithms are also based-on open-source GPS source-codes, resources hosted by Danish GPS Centre and GPS algorithms analysed in the following books; Tsui (2005), Borre *et al.* (2007), Gleason and Gebre-Egziabher (2009); Borre and Strang (2012).

Due to the size, the codes length and number of algorithms involved (~40 m-files); mostly the applicable equivalent Simulink models, flowcharts and screenshots are included in this report. The complete GPS algorithms and the test-benches can be found in the accompanying disc. Table 4.1 summarises the GPS algorithms, as well as relevant references to the source-codes.

The design and implementation of the GPS receiver (some algorithms only in phase 2 and 3) occurs in three phases as per the objectives of this research. These phases are outlined as follows:

- Software-defined GPS receiver: Phase 1 based-on Matlab only.
- Software-to-Firmware-defined GPS receiver: Phase 2 based-on Matlab/Simulink and VHDL.
- Firmware-defined GPS receiver algorithm(s): Phase 3 based-on VHDL only.

4.3.1 Software-defined GPS Receiver: Phase 1 Based-on Matlab

This section examines how the GPS receiver algorithms will be implemented in software utilising a personal computer. The implementation follows the same order as stated in Chapter 3, commencing with the acquisition section, followed by the tracking loop and lastly the navigation message decoding and processing to compute PVT estimates. Test vectors (test benches for each GPS algorithms) and input dataset (the pre-captured GPS L1 signal dataset) are also addressed.

Figures 4.1 to 4.10 portray the various steps in the GPS software signal processing chain, illustrating the PVT computations. Each of these sections is briefly explained. For more information, see the relevant reference books listed in Section 4.3 and the applicable algorithms.

4.3.1.1 GPS Receiver Software Path Initialisation

The directory containing all the GPS software files and pre-captured GPS L1 C/A signal datasets are placed here. Ensure all the files are bugs or errors free before running *init.m* file. Processing starts if 1 is entered and the *init.m* file is executed. Figure 4.1 depicts the initialisation process.

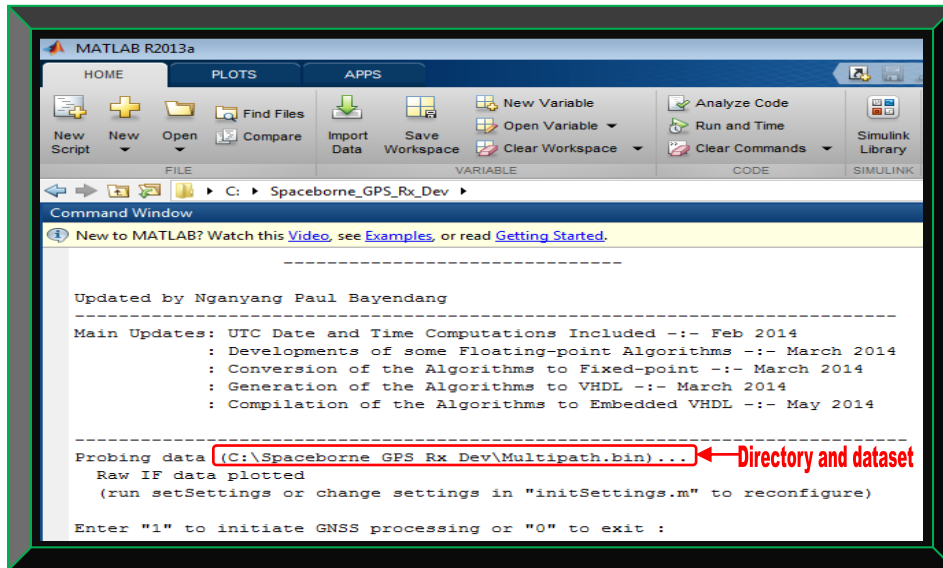


Figure 4.1: Illustration of initialising the GPS receiver software directory

4.3.1.2 GPS Receiver Software Settings Initialisation

The settings of all the parameters to be used in the GPS signal processing is defined here. First ensure *settings* is correct prior to executing the *init.m* file. The *initSettings.m* file (containing the initialisation settings values) is executed once the appropriate settings in Figure 4.2 are defined.

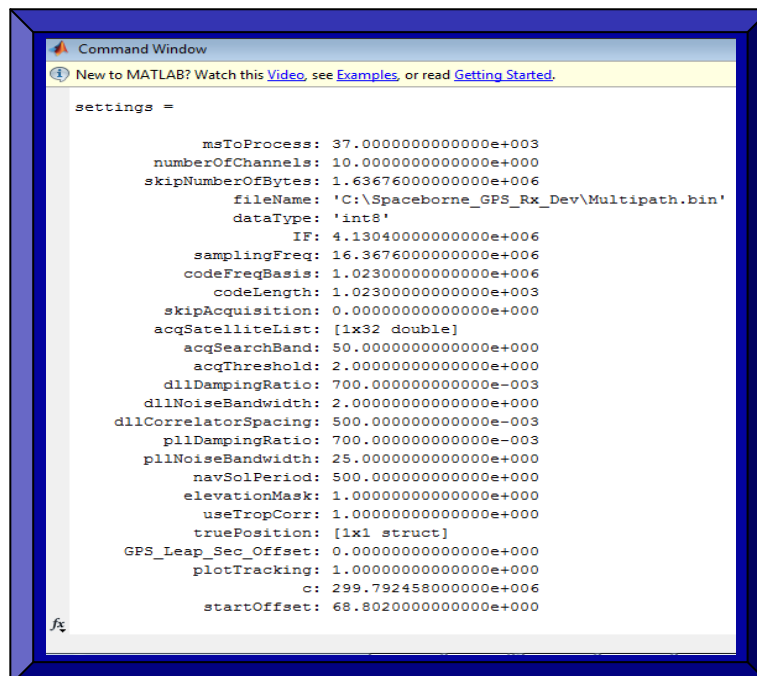


Figure 4.2: Preview of the GPS software receiver settings initialisation

4.3.1.3 GPS L1 C/A Signal Acquisition Flowchart

Figure 4.3 depicts the acquisition process; whereby, the Matlab file *acquisition.m* is debugged and saved in the directory illustrated in Section 4.3.1.1. This *acquisition.m* file is based on parallel code phase search and executes the processes discussed in Sections 3.5.2 and 8.2.4. The *acquisition.m* file is invoked (and calls other files) once the *initSettings.m* file is executed.

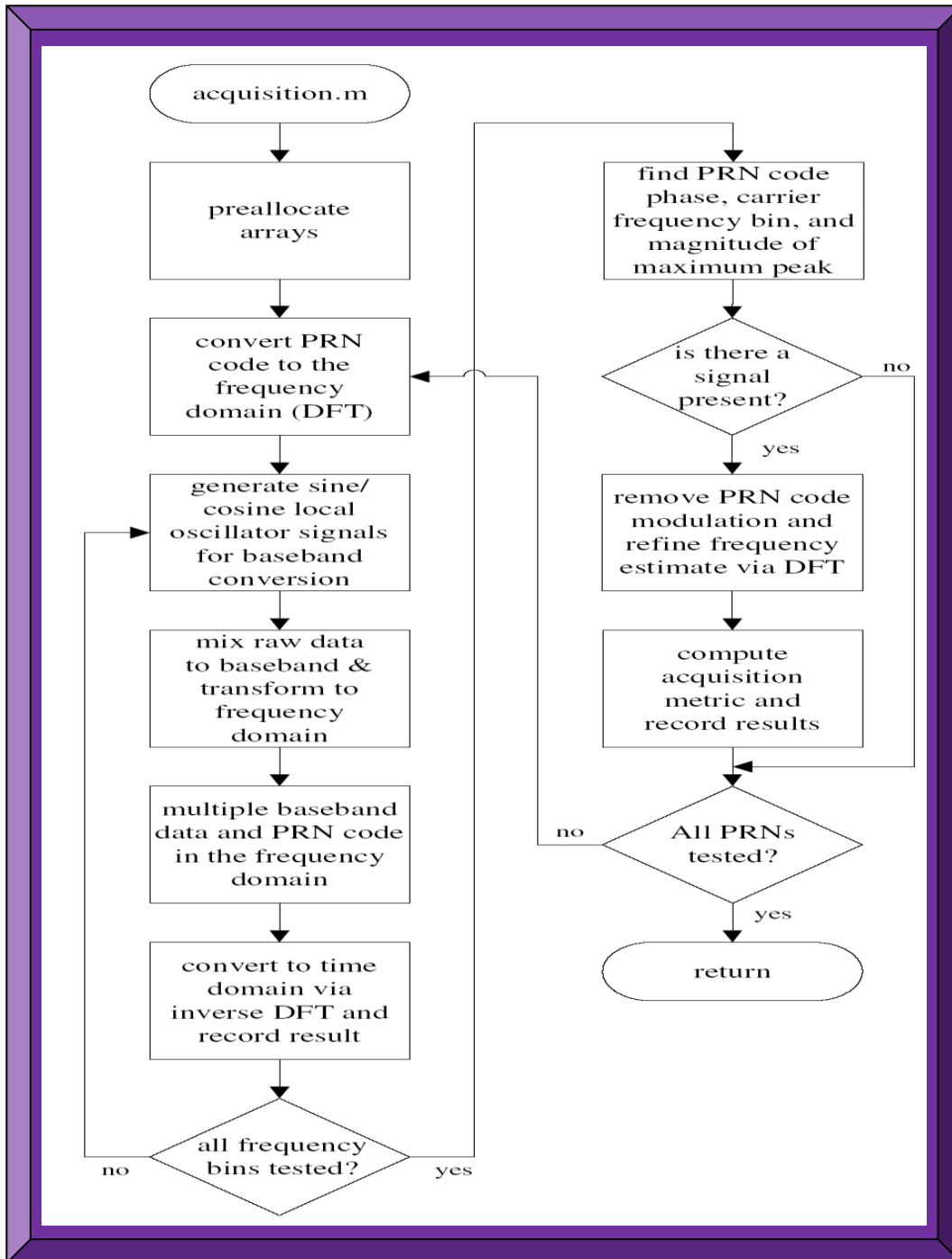


Figure 4.3: Illustration of the acquisition algorithm execution process

(Adapted from Borre *et al.*, 2007: 146)

4.3.1.4 GPS L1 C/A Signal Tracking Flowchart

Figure 4.4 depicts the tracking process; whereby, the Matlab file *tracking.m* is debugged and saved in the directory illustrated in Section 4.3.1.1. This *tracking.m* file is based-on Costa PLL and DLL tracking techniques and executes the processes discussed in Sections 3.5.2 and 8.2.5. The *tracking.m* file is invoked (with other files) once the *acquisition.m* file completes execution.

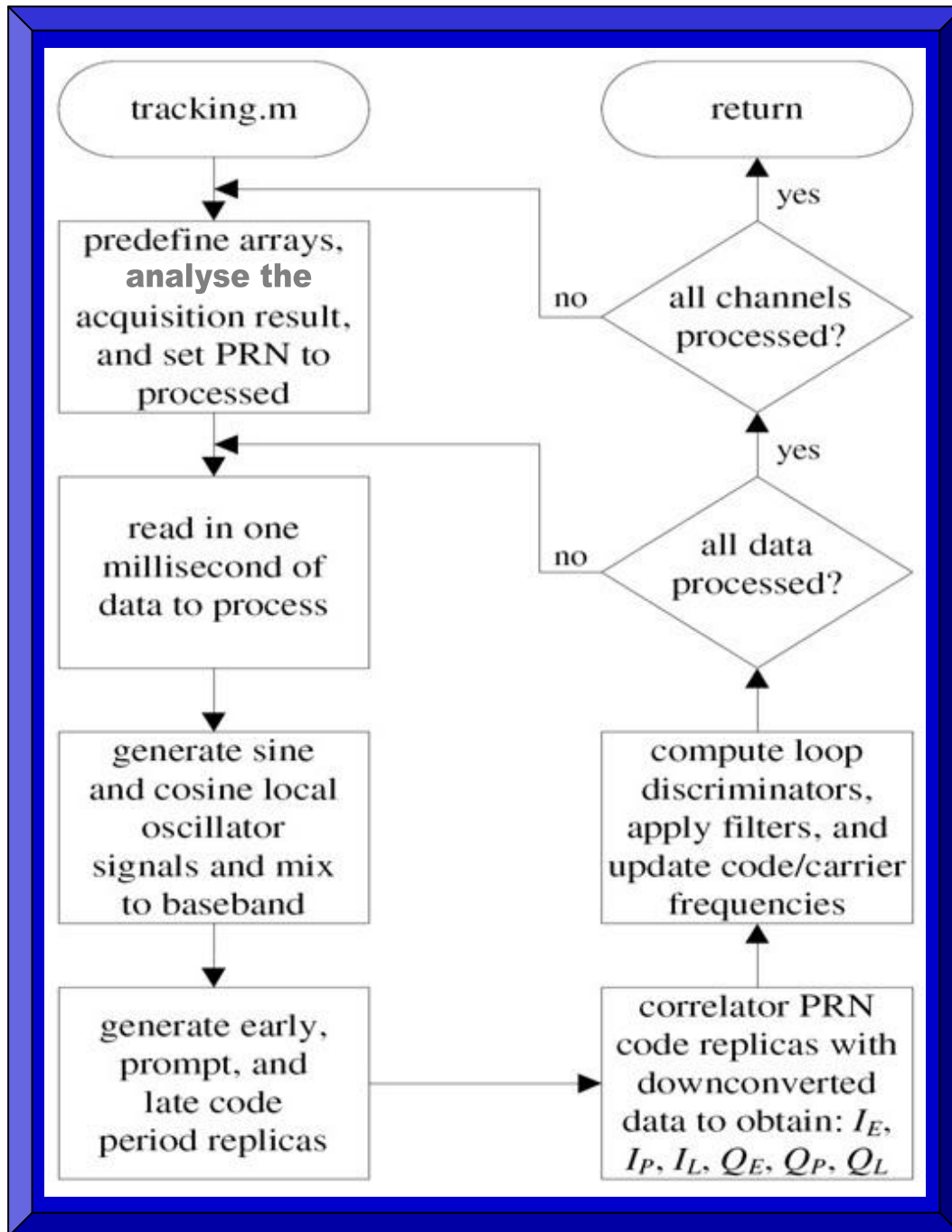


Figure 4.4: Illustration of the tracking algorithm execution process

(Adapted from Borre *et al.*, 2007: 147)

4.3.1.5 GPS L1 C/A Signal Positioning Flowchart

Figure 4.5 depicts the positioning process; whereby, the Matlab file *postNavigation.m* is debugged and saved in the directory illustrated in Section 4.3.1.1. This *postNavigation.m* file is based-on least-squares method and executes the processes explained in Sections 3.5.3 and 8.2.6. The *postNavigation.m* file is invoked (and calls other files) once *tracking.m* is run successfully.

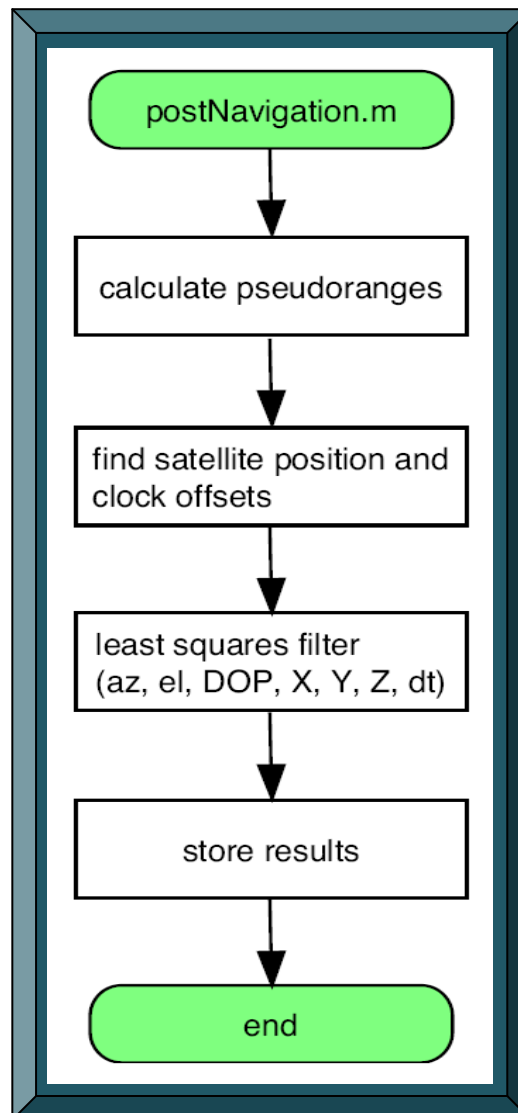


Figure 4.5: Illustrative summary of the positioning process
(Adapted from Borre & Akos, 2005: 5)

Figure 4.5 summarises the positioning process. The details involved are illustrated in Figures 4.6 and 4.7. Once the *postNavigation.m* file has successfully executed, control is passed to the *calculatePseudoranges.m* file, to compute the acquired satellites positions and pseudo-ranges. The outcome is passed to the *leastSquarePos.m* file which calculates the GPS receiver position.

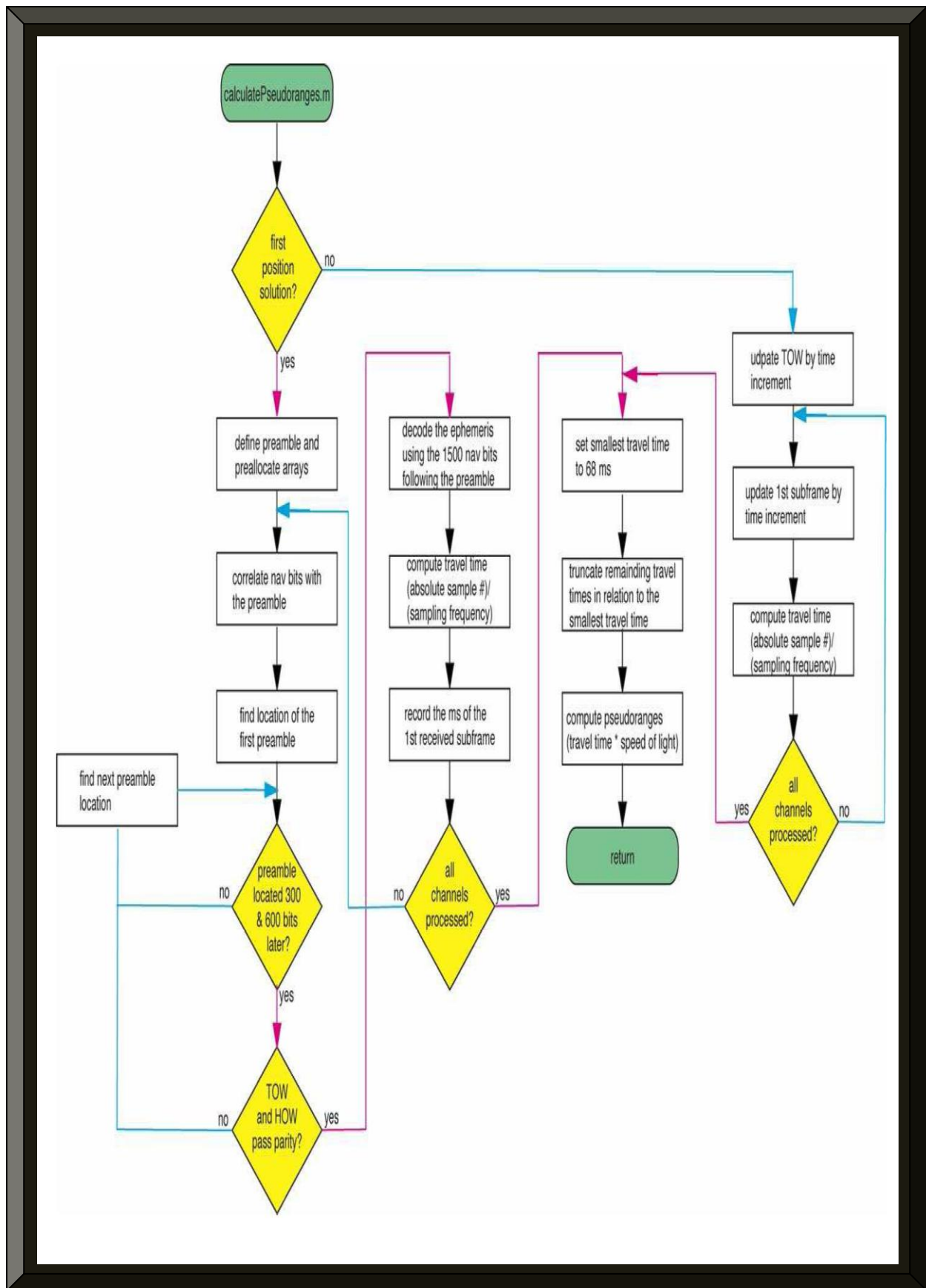


Figure 4.6: Illustration of the GPS satellites position and pseudo-ranges algorithms execution
(Adapted from Borre & Akos, 2005: 4)

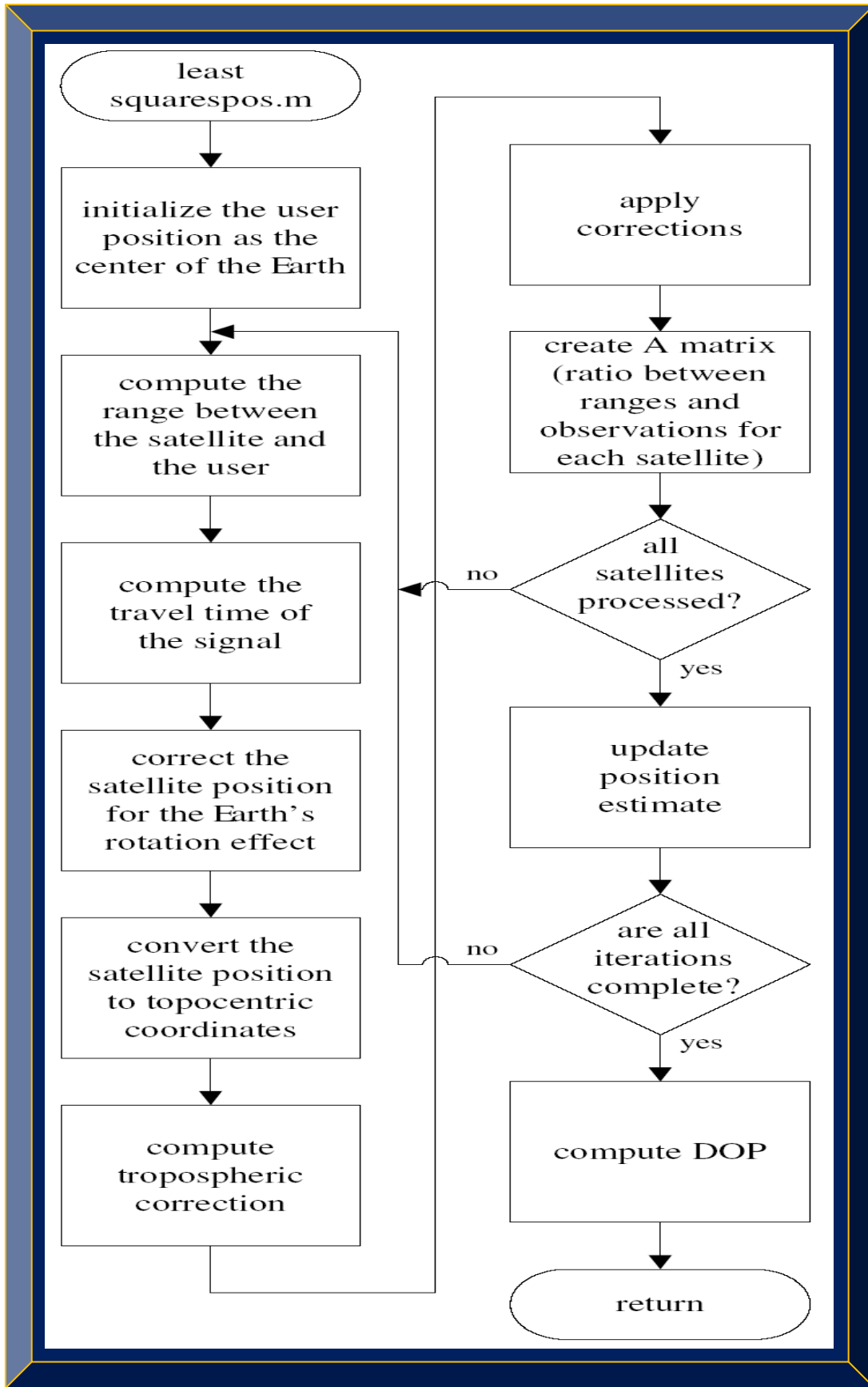


Figure 4.7: Illustration of the GPS receiver least-squares algorithm execution process
 (Adapted from Borre *et al.*, 2007: 151)

4.3.1.6 GPS L1 C/A Receiver Position Computation Summary Flowchart

Figure 4.8 summarises the GPS receiver position calculation processes. The *postProcessing.m* file is the main file and sequentially executes the *acquisition.m*, *tracking.m* and *postNavigation* files. These three files in turn call other applicable sub-files (see Figure 4.11 for the complete GPS receiver execution process) also placed in the GPS software directory. Some of these files plot the outcome of successive stages of the processing, while others serves as input or output at successive and intermediate stages in the processing chain. Some of the files are modified to suit the research. For details on all the files involved including the modifications made, see Table 4.1.

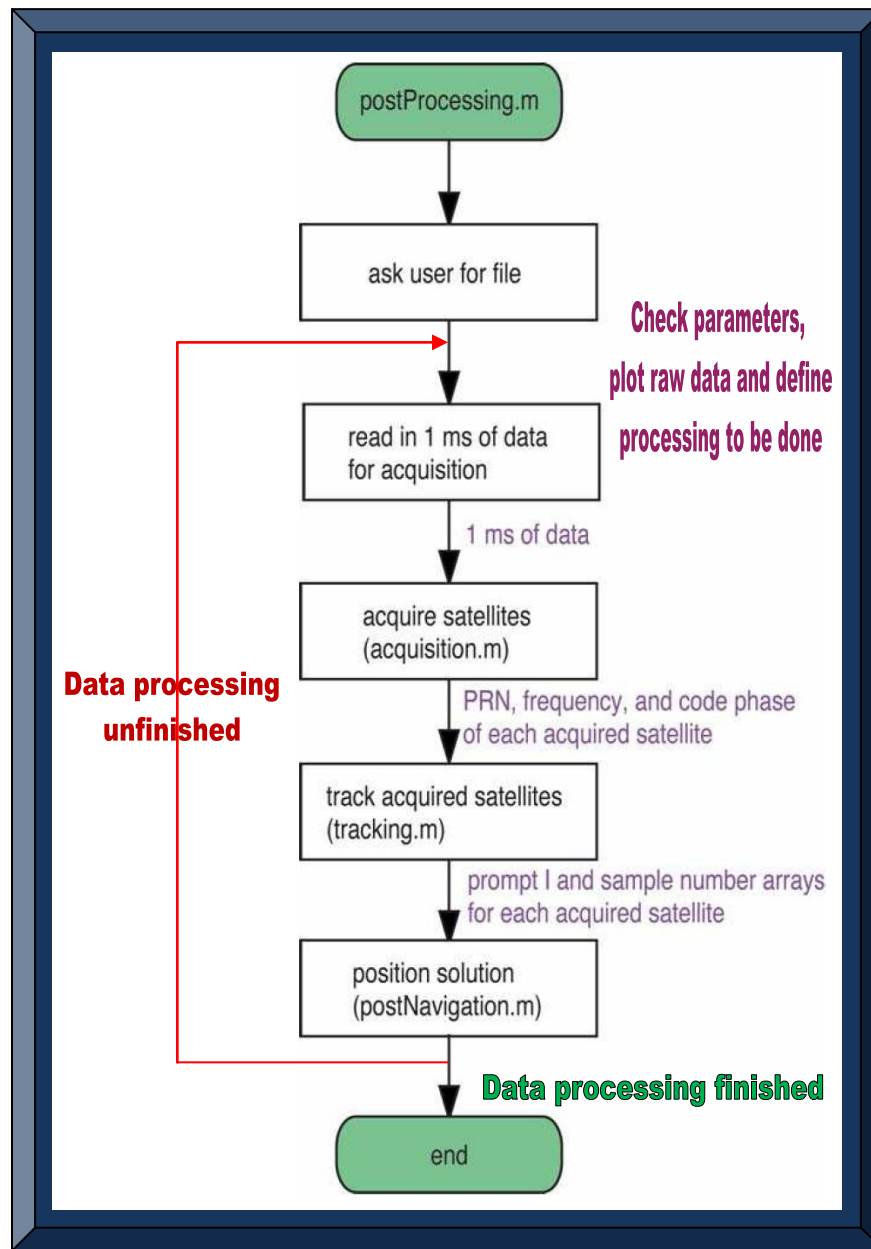


Figure 4.8: GPS receiver position computation summary

(Adapted from Borre & Akos, 2005: 3)

4.3.1.7 GPS L1 C/A Receiver Velocity Computation Flowchart

Figure 4.9 illustrates the process involved in estimating the GPS receiver velocity from the position estimates. The Matlab algorithm was not implemented, for reasons stated in Chapter 6.

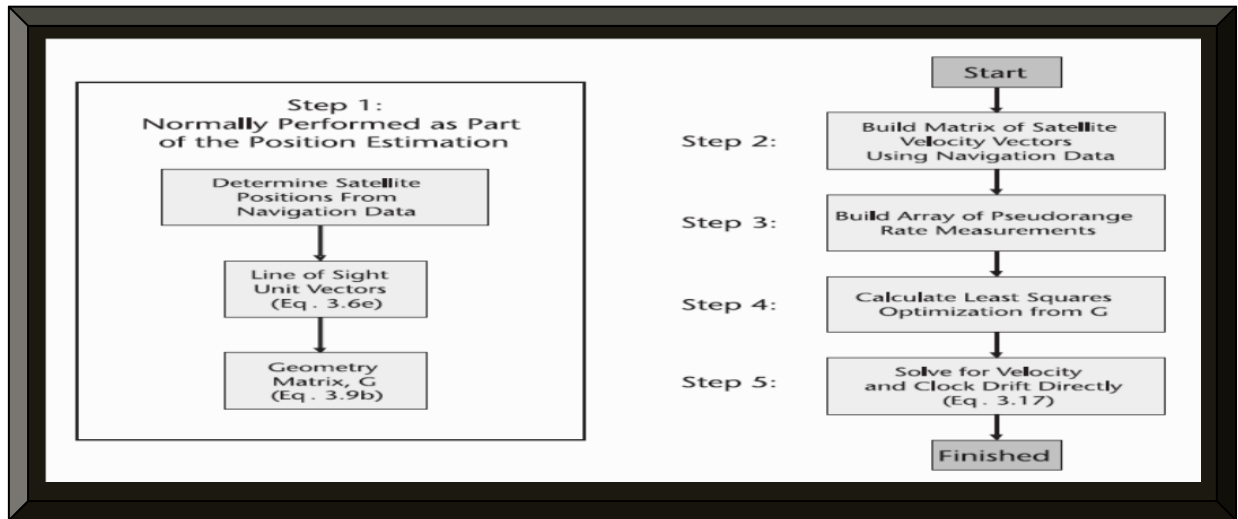


Figure 4.9: GPS receiver velocity calculation
(Adapted from Gleason & Gebre-Egziabher, 2009: 65)

4.3.1.8 GPS L1 C/A Receiver UTC Date and Time Computations Flowchart

Figure 4.10 illustrates the process involved in computing the GPS receiver's date and time in UTC format. The file *gps2utc.m* does the conversion using GPS week number and seconds as inputs, to give the output date (year, month and day) and time (hour, minute and second) in UTC.

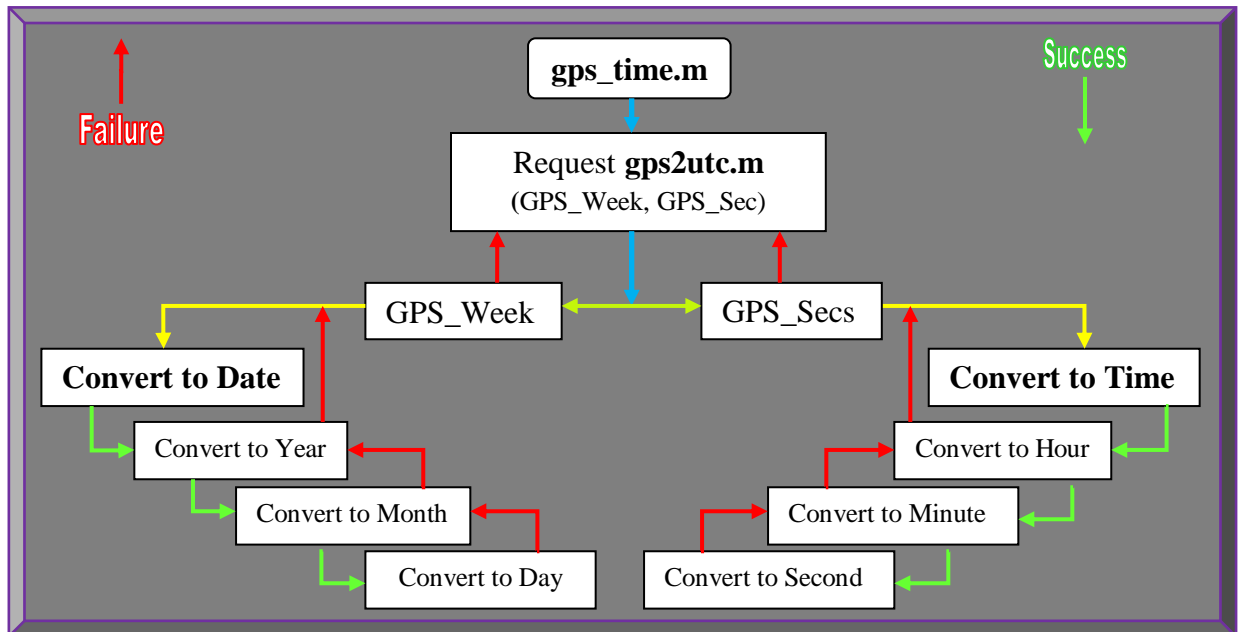


Figure 4.10: Illustration of the GPS receiver UTC date and time calculations

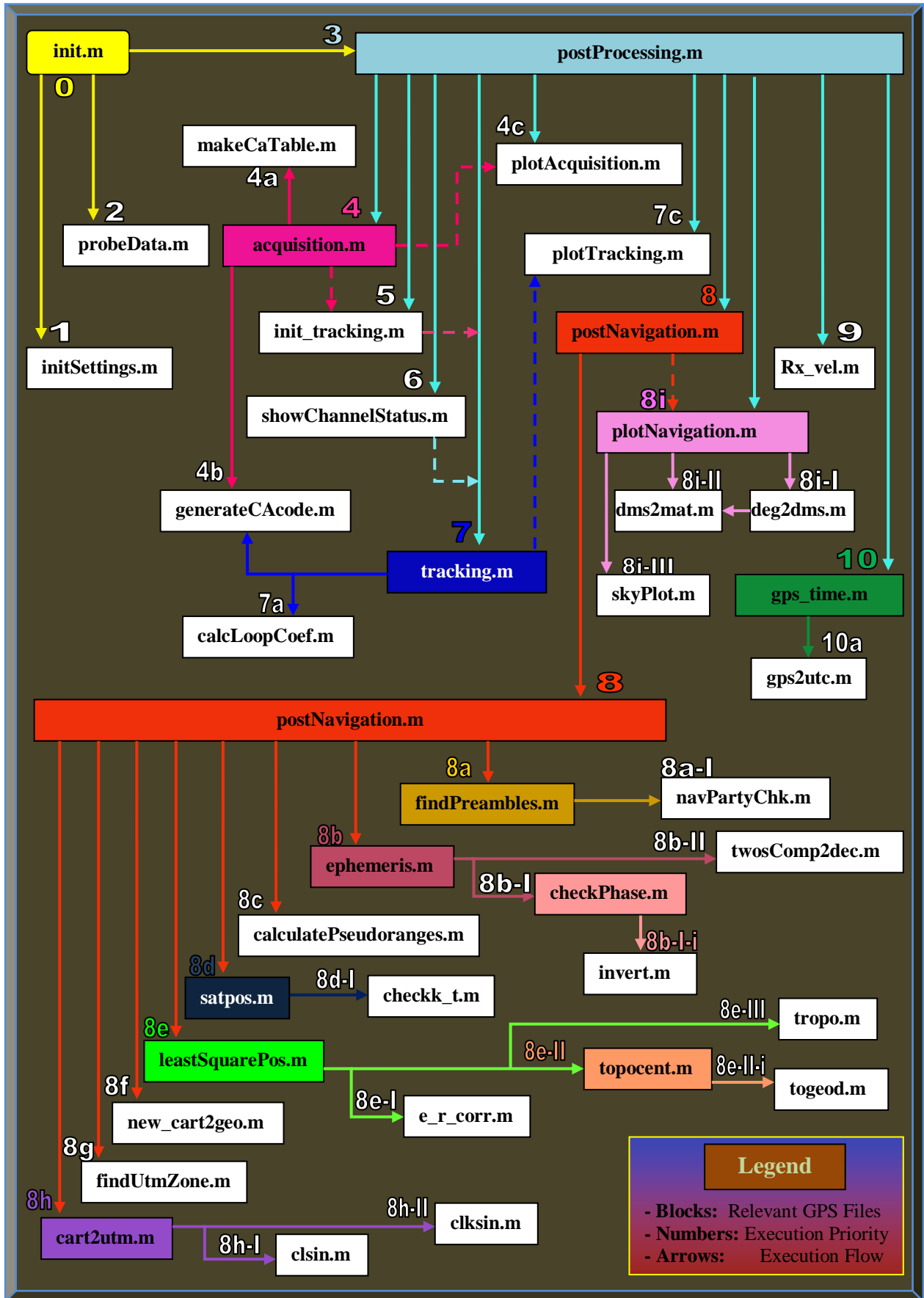


Figure 4.11: Grand and simplified execution summary of the software-defined GPS receiver operation

Summarised in Table 4.1 are the Matlab GPS receiver floating-point files (functions and scripts).

Table 4.1: Summary of all the applicable Matlab floating-point GPS receiver algorithms

Matlab GPS Receiver Floating-point Files	Main Modifications / Contributions Made		GPS Receiver Files Remarks / Purpose	Source-code References (Authors / Contributors)
1) acquisition.m	Option to choose best FFT	HDL code generation / TB	Searches GPS satellites	D. Plausinaitis & DM. Akos
2) c2gm.m	None: Ideal and stable for higher altitudes $h > 100\text{km}$		Replaces cart2geo.m	Borre & Strang; P. Meissl
3) calcLoopCoef.m	Wn returned in output	HDL code generation / TB	Calc PLL / DLL coefs	D. Plausinaitis & DM. Akos
4) calculatePseudoranges.m	None	HDL code generation / TB	Calc all pseudoranges	D. Plausinaitis <i>et al.</i>
5) cart2geo.m	None	HDL code generation / TB	Converts cart to geo	Kai Borre
6) cart2utm.m	None	HDL code generation / TB	Converts cart to utm	Kai Borre
7) check_t.m	None	HDL code generation / TB	Checks GPS TOW	Kai Borre
8) checkPhase.m	None	HDL code generation / TB	Checks parity	D. Plausinaitis & DM. Akos
9) clksin.m	None	HDL code generation / TB	Sums $i+j$ arg of sinus	Kai Borre
10) clsin.m	None	HDL code generation / TB	Sums sinus of arg	Kai Borre
11) deg2dms.m	None	HDL code generation / TB	Converts degs to d/m/s	Kai Borre & D. Plausinaitis
12) dms2mat.m	None	HDL code generation / TB	Splits dms to mat	Kai Borre & D. Plausinaitis
13) e_r_corr.m	None	HDL code generation / TB	Gives Earth rot ECEF	Kai Borre
14) ephemeris.m	None	HDL code generation / TB	Decodes ephemerides	D. Plausinaitis & K. Larson
15) findPreambles.m	None	HDL code generation / TB	Finds start of 1 st frame	D. Plausinaitis <i>et al.</i>
16) findUtmZone.m	Display UTM zone	HDL code generation / TB	Locates UTM zone	Darius Plausinaitis
17) generateCAcode.m	None	HDL code generation / TB	Generate satellites PRN	D. Plausinaitis <i>et al.</i>
18) gps2utc.m	Entire code re-structured	HDL code generation / TB	Converts GPST to UTC	M. Evans & J. LaMance
19) gps_time.m	Added to SoftGNSS v3.0	HDL code generation / TB	Extracts mean GPST	Paul Bayendang
20) init.m	Number format changed to longEng, display settings		Initialises SoftGNSS	D. Plausinaitis & DM. Akos
21) init_tracking.m	Display channels info	HDL code generation / TB	Initialises tracking	D. Plausinaitis <i>et al.</i>
22) initSettings.m	Define GPS rx parameters, Elapsed GPS sec field added		Initialises settings	Darius Plausinaitis
23) invert.m	None	HDL code generation / TB	Inverts binary bits	D. Plausinaitis <i>et al.</i>
24) leastSquarePos.m	Disp all DOPs mean values	HDL code generation / TB	Computes Rx position	Kai Borre <i>et al.</i>
25) makeCaTable.m	None	HDL code generation / TB	Creates all 32 sats code	D. Plausinaitis <i>et al.</i>
26) navPartyChk.m	None	HDL code generation / TB	Computes parity status	D. Plausinaitis & K. Larson
27) new_cart2geo.m	None: Ideal for both Earth and Space LEO altitudes		Replaces cart2geo.m	Borre & Strang; Areg H.
28) plotAcquisition.m	Plot all acquired GPS satellites in 3D on the same graph		3D plot of acq result	Darius Plausinaitis
29) plotNavigation.m	Labels / texts renamed	Plot 3D GPS receiver position, UTM variations		Darius Plausinaitis
30) plotTracking.m	None		Plot tracking results	Darius Plausinaitis
31) postNavigation.m	Displayed texts typos fixed	HDL code generation / TB	Calc rx nav solution	D. Plausinaitis & K. Larson
32) postProcessing.m	UTC date/time conv. added, date/time stamps, text disp		Post-processes dataset	D. Plausinaitis & DM. Akos
33) probeData.m	None		Plots raw data info	D. Plausinaitis & DM. Akos
34) satpos.m	None	HDL code generation / TB	Calc sats XYZ position	Kai Borre <i>et al.</i>
35) setSettings.m	None		Creates new settings	SoftGNSS
36) showChannelStatus.m	None	HDL code generation / TB	Disp all channels status	P. Rinder <i>et al.</i>
37) skyPlot.m	None		Plot satellites skyview	D. Plausinaitis & K. Larson
38) togeod.m	None	HDL code generation / TB	Calc geodetic given ref	Kai Borre <i>et al.</i>
39) topocent.m	None	HDL code generation / TB	Transforms vector dx	Kai Borre
40) tracking.m	None	HDL code generation / TB	Code / carrier tracking	Dennis M. Akos <i>et al.</i>
41) tropo.m	None	HDL code generation / TB	Calc tropospheric corr	Kai Borre <i>et al.</i>
42) twosComp2dec.m	None	HDL code generation / TB	Converts 2-comp bin#	Darius Plausinaitis
Matlab test-benches (TB): x 31	31 scripts written to test each GPS functions in Table 5.1		Functions test-bench	Paul Bayendang

4.3.2 Software-to-Firmware Conversion: Phase 2 Based-on Matlab / Simulink and VHDL

This section explains how the GPS receiver algorithms implemented in floating-point Matlab (Figure 4.11); are converted to equivalent Simulink models and finally to VHDL. That is, the implemented GPS receiver algorithms are translated from software based-on floating-point Matlab to fixed-point Matlab and to firmware based-on VHDL, with equivalent Simulink models of the same algorithms as by-products. The software-to-firmware conversion process utilises the Matlab HDL coder - a toolbox in Matlab that converts Matlab / Simulink algorithms to a Hardware Design Language (HDL). The algorithms converted to HDL, in this regards to VHDL, are independent of the Matlab files / Simulink models and vendor's non-specific FPGA. This means, the converted VHDL codes are standalone and do not need Matlab / Simulink to function and can be ported, tailored and embedded as firmware into any FPGA. Discrete Matlab test benches are written for all the applicable GPS functions. They also serve as automatic VHDL test-benches during the conversion process as well as test vectors during the verification process.

Previously (before Matlab R2012a), directly converting Matlab codes to HDL was impossible and could only be achieved by first converting, in this regard, the software GPS Matlab files to Simulink using the Matlab Function Block (formerly called Embedded Matlab Function Block before Matlab R2011a) – the Matlab Function Block enables porting of Matlab codes to Simulink and facilitates usage of Matlab codes together with Simulink models in Simulink. The Simulink models of the Matlab GPS software are not only used to validate the Matlab implementations but can also serves as improved alternative algorithm models (either discretely or collectively), for each of the functions of the floating-point Matlab software GPS receiver.

Although MathWorks HDL Coder toolbox was released in Matlab version R2008, the literature review for converting Matlab codes (especially on GPS) to other forms (C/C++, Simulink and HDL) have been done using variations of the procedures and tools described in Hamsa (2009). This was based-on Matlab/Simulink, Real Time Workshop, Target Support Package (TC6) and Texas Instrument Code Composer Studio using C6713 DSP starter kit. The Danish GPS receiver research Group 09gr944 (2009), used Matlab/Simulink and Xilinx System Generator ISE Tool based-on ML507 EK – 5VFX70TFF1136-3 chip. The work in this thesis utilises *Matlab R2013a* and integrated tools and toolboxes that include Simulink, HDL Coder, HDL Verifier and the free Altera Quartus II and ModelSim-Altera software packages. Since the generated VHDL codes are generic, the targeted FPGA in the end is the Xilinx space-grade Virtex-5QV (XQR5VFX130).

During the conversion process, Simulink is used to test and verify the Simulink models of the equivalent Matlab floating and fixed-point codes. The HDL Coder uses the Matlab fixed-point toolbox to convert Matlab floating-point codes to equivalent fixed-point algorithms to (i) Matlab (ii) Simulink models (iii) HDL (i.e. VHDL). The HDL Verifier with ModelSim-Altera simulates the generated VHDL codes and verifies the results with the original Matlab codes. The Altera Quartus II design suite can be invoked directly within the Matlab HDL conversion workflow to compile (synthesis, place & route and generate programming files) the generated VHDL codes to firmware. Alternatively, the Altera Quartus II software can be indirectly use to post compile the generated VHDL code, without necessarily being invoked from Matlab. By firmware, it means the synthesised VHDL codes can readily be embedded into a microcontroller. The processes and tools involved are described in Olivieri *et al.* (2011: 12); Kumar (2012: 13 & 36). Figure 4.12 depicts the entire process used in this thesis and each is elaborated in their respective sections.

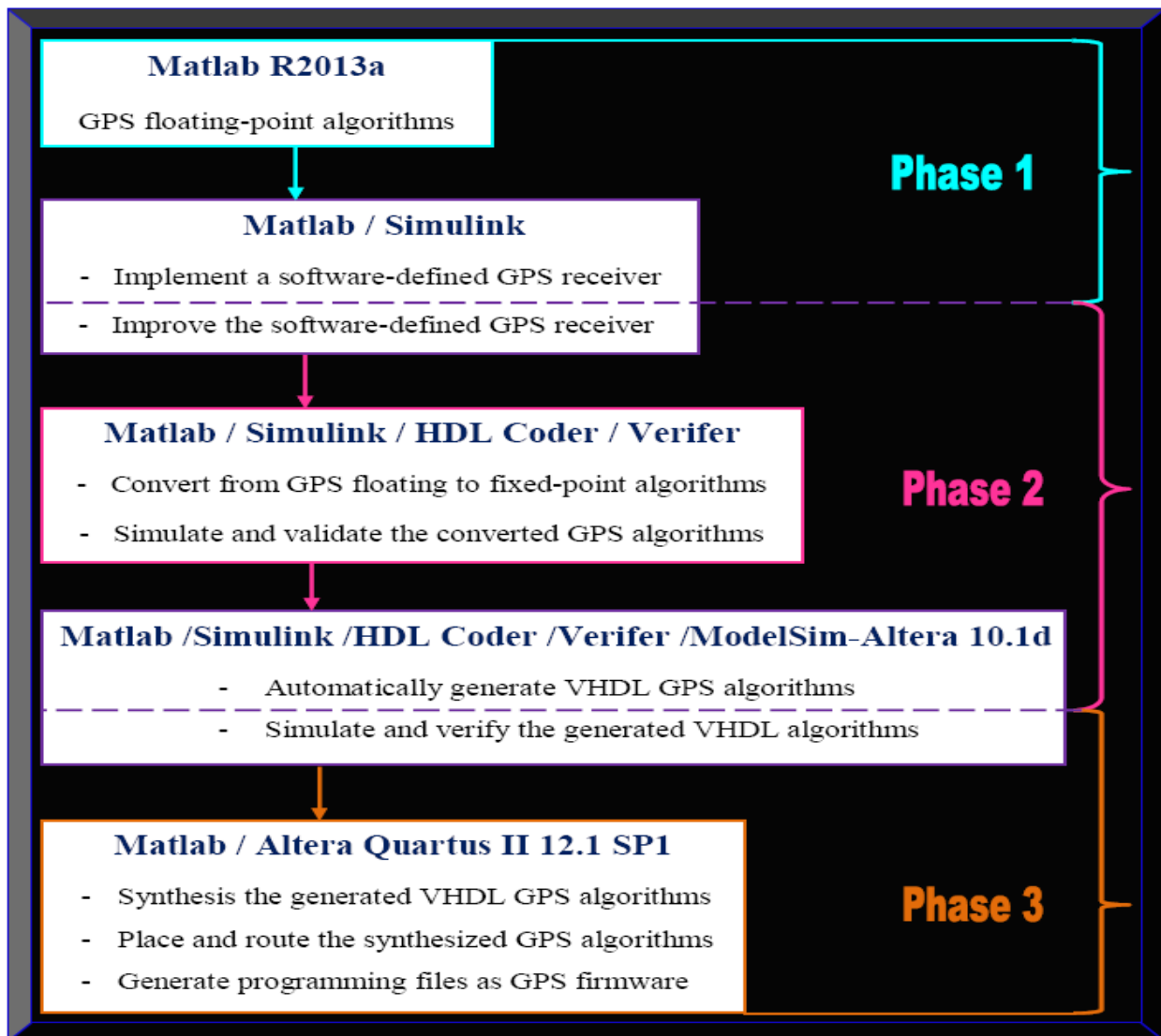


Figure 4.12: The GPS receiver software-to-firmware conversion methodology

4.3.2.1 Software-to-Firmware (Matlab to VHDL Algorithms) Conversion Procedure

VHDL code can be generated either from Matlab command line and / or from the Graphical User Interface (GUI). The following procedure discussed, makes use of both methods where relevant.

4.3.2.1.1 Step 1: Ensure all GPS Matlab Functions, Scripts and Test Vectors are in a Local Folder

The first step to directly convert Matlab floating-point codes to finally VHDL, usually requires a sequence of steps in Matlab command line to create a *design_name* (the name of a Matlab function) and a *test-bench_name* (either a Matlab script or function with no input), followed by copying the *design_name(s)* and *test-bench_name(s)* to a local temporary directory using Matlab commands. The simple alternative approach used in this thesis is to create or copy all the Matlab files (functions, scripts and test vectors) to a local folder different from the same root directory as the Matlab installation. All the files are placed in the folder named *Spaceborne_GPS_Rx_Dev*.

4.3.2.1.2 Step 2: Simulate / Verify the Floating-point Matlab Software GPS Receiver Algorithms

This step simply validates the Matlab codes. In other words, it ensures all the GPS Matlab functions, scripts and test vectors execute without runtime errors. The outcome of this step should exactly be the same as the results from Phase 1, when the Software-defined GPS receiver was executed. That is, the GPS signal processing chain should follow the same order as in Chapters 3 and 4; where the acquisition section is first processed, followed by the tracking loop and finally navigation message decoding and processing to compute PVT estimates. After step 1, the simulation process is initiated by simply typing the name of the test-bench *init.m* in Matlab command line. The results obtained are exactly the same as the results of Phase 1, as required.

Next is to verify the floating-point design compliance for fixed-point code generation. If the verification fails, the coding style is re-structured and syntax supporting fixed-point code generation is used. These include using System Objects instead of Functions, ensuring the test bench files are Matlab scripts or functions without arguments, the designed GPS files are all Matlab functions, at least one call is made to the designed GPS Matlab functions from the test bench (as all call sites contributes when determining the proposed fixed-point types), the *#codegen* pragma is included in every function, in-lining in functions is controlled for modular codes generation in multiple files and persistent variables are used instead of global variables. The algorithm is again re-simulated until successful completion before proceeding to step 3.

Additional information and limitations can be obtained from MathWorks website or Matlab help.

4.3.2.1.3 Step 3: Create a New GPS VHDL Coder Project

Following step 2 is the creation of a new project in VHDL. This new project is created either from (i) executing the following Matlab command `coder -hdlcoder -new GPS_Receiver`; where `GPS_Receiver` is the name of the new VHDL project or (ii) by using the Matlab GUI and navigate to HDL Coder. In the Matlab HDL Coder Project dialog window, the project is named as `GPS_Receiver` and maintained as the default location path. In addition, a new file called `GPS_Receiver.prj` is created in the current folder `GPS_Receiver`. This project file holds all the project selections made in the GUI whereas the project folder simply contains all the Matlab GPS files. Converting all the GPS files at once is impossible and should definitely fail. The approach used is to note all the actual GPS function or design files needing conversion, from the files that simply plot the processed results or are in the form of Matlab scripts. The GPS functions or design files are then scaled where possible and discrete corresponding Matlab test benches are created for each. The Matlab test-benches in their simplest form provide the test vector or basic inputs to their respective functions and as well invoke the Matlab functions. The Matlab test-benches are further used by Matlab R2013a HDL Coder workflow advisor wizard, to automatically define the type of the input variable(s) in terms of its class, size and complexity. Figure 4.13 shows some applicable GPS Matlab function files with their respective test-benches.

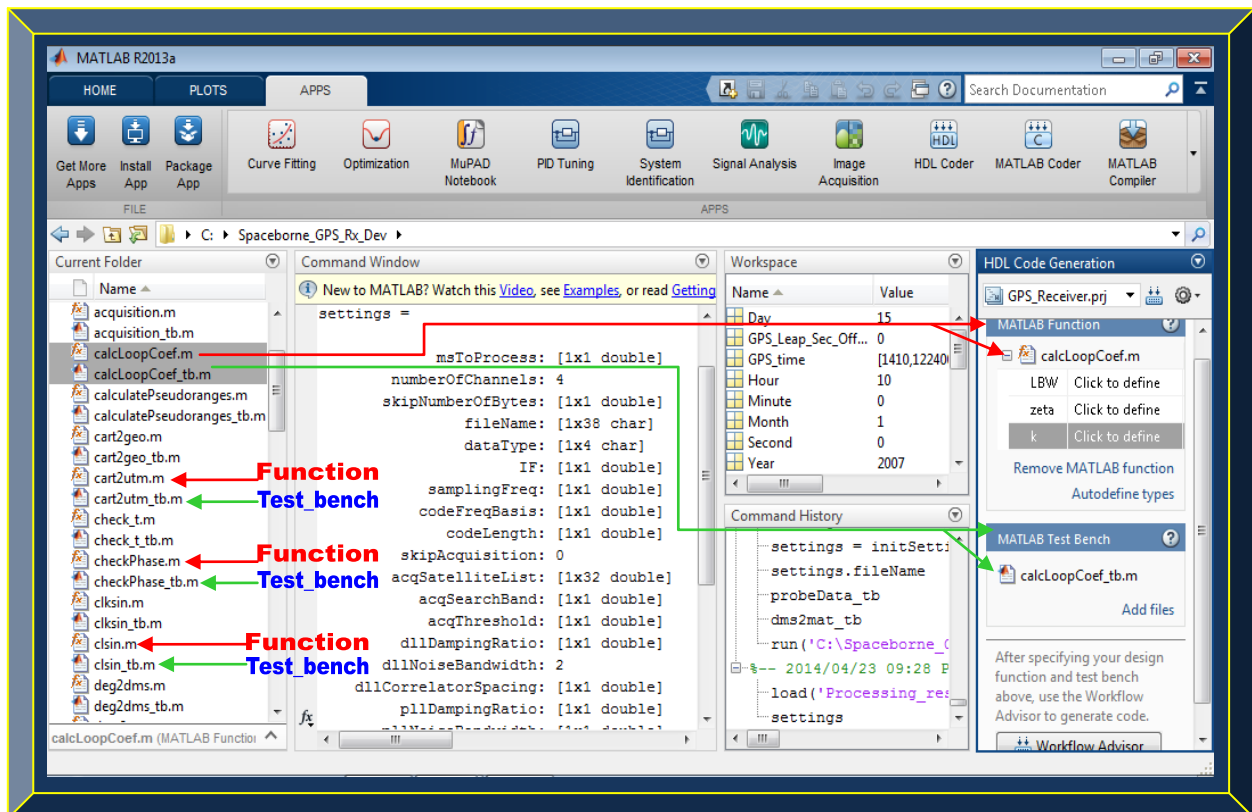


Figure 4.13: Illustration of Matlab functions and corresponding test-benches in project folder

4.3.2.1.4 Step 4: Convert the GPS Matlab Codes from Floating-point to Fixed-point

According to MathWorks, applications in signal processing for reconfigurable platforms need algorithms that are usually specified based-on floating-point operations but because of power, cost and performance reasons; they are often implemented based-on fixed-point operations either as special-purpose hardware in FPGAs or as firmware / software for DSP cores. However, fixed-point conversion can be very cumbersome and time-consuming, which normally demands most of the total design and implementation time. As a result, automated tools in Matlab versions R2012a and above; were developed to simplify, accelerate and optimise the conversion process.

The goal in software implementations is usually to define an optimised fixed-point specification that minimises the time of execution and the code size for a specified constraint in computational accuracy. During the GPS software-to-firmware conversion process, the algorithm optimisation is accomplished through the alteration of the binary point location (for scalability) and choosing a data word length, according to the various kinds of data that the target processor will support.

Hardware implementation dictates that, the complete architecture be optimised such that an efficient implementation will minimise both the power consumed and the area used. As a result, the aim of the conversion process is often focused around minimising the operator word length.

The tool in Matlab that facilitates step 4 conversion process is the Floating-point to Fixed-point Workflow that is integrated in HDL Coder Workflow Advisor in Matlab version R2012a and above. This tool is the key (determines the quality) and the most time consuming step in the software-to-firmware conversion process. It is applicable to all the design files (Matlab GPS functions) and their test-benches (Matlab GPS scripts) and involves the following sub-steps:

- Sub-step 1: Launch HDL Coder Workflow Advisor from Matlab R2013a by clicking APPS, and then HDL Coder. Ensure the floating-point design is set for code generation.
- Sub-step 2: Add the design file & test-bench to their respective sections. Click the auto-define types to compute fixed-point types by simulating the test-bench and word length.
- Sub-step 3: Run the simulation to generate readable and traceable fixed-point Matlab codes, as well as manually adjust (if necessary) and apply the suggested fix-point types.
- Sub-step 4: Validate and verify the generated fixed-point type or design.
- Sub-step 5: Lastly, test the numerics by comparing the numerical accuracy of the generated Matlab fixed-point algorithm with the original Matlab floating-point algorithm.
- Sub-step 6: If sub-step 5 results are unsatisfactory, refine and re-iterate from sub-step 2.

Figures 4.13 - 4.16 depicts the floating to fix-point conversion processes using *calcLoopCoef.m*.

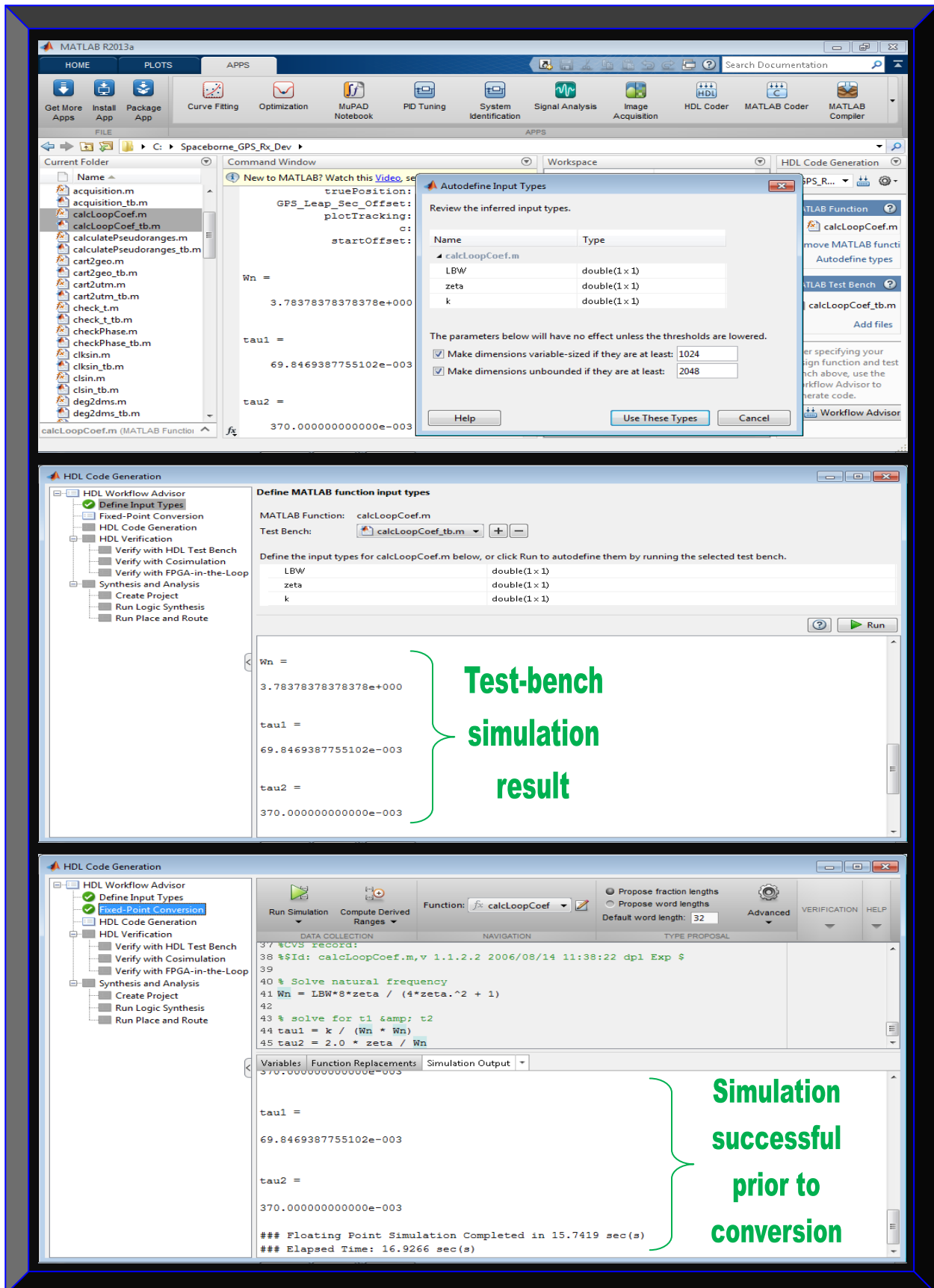


Figure 4.14: Illustration of Step 4; sub-steps 1 to 3 – Define input types and simulate files

Fixed-point conversion successful

```

29 %GNU General Public License for more details.
30 %
31 %You should have received a copy of the GNU General Public License
32 %along with this program; if not, write to the Free Software
33 %Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301,
34 %USA.
35 %-----
36
37 %CVS record:
38 %%Id: calcLoopCoef.m,v 1.1.2.2 2006/08/14 11:38:22 dpl Exp $
39
40 % Solve natural frequency
41 Wn = LBW*8*zeta / (4*zeta.^2 + 1)
42
43 % solve for t1 & t2
44 tau1 = k / (Wn * Wn)
45 tau2 = 2.0 * zeta / Wn

```

Variables | Function Replacements | Type Validation Output

```

### Generating Fixed Point MATLAB Code calcLoopCoef\_FixPt using Proposed Types
### Generating Fixed Point MATLAB Design Wrapper calcLoopCoef\_wrapper\_FixPt

### Generating Mex file for ' calcLoopCoef_wrapper_FixPt '
Code generation successful: View report

```

Test-bench Input data type validation

```

32 %along with this program; if not, write to the Free Software
33 %Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301,
34 %USA.
35 %-----
36
37 %CVS record:
38 %%Id: calcLoopCoef.m,v 1.1.2.2 2006/08/14 11:38:22 dpl Exp $
39
40 % Solve natural frequency
41 Wn = LBW*8*zeta / (4*zeta.^2 + 1)
42
43 % solve for t1 & t2
44 tau1 = k / (Wn * Wn)
45 tau2 = 2.0 * zeta / Wn

```

Validation succeeded

Variable	Type	Sim Min	Sim Max	Static Min	Static Max	Whole Nu...	Proposed Type
k	double	1	1			Yes	numerictype(0, 1, 0)
zeta	double	0.7	0.7			No	numerictype(0, 32, 32)
▲ Output							
tau1	double	0.07	0.07			No	numerictype(0, 32, 35)
tau2	double	0.37	0.37			No	numerictype(0, 32, 33)
▲ Local							
Wn	double	3.78	3.78			No	numerictype(0, 32, 30)

Figure 4.15: Illustration of Step 4; sub-steps 3 and 4 – Modify proposed fixed-point types and validate

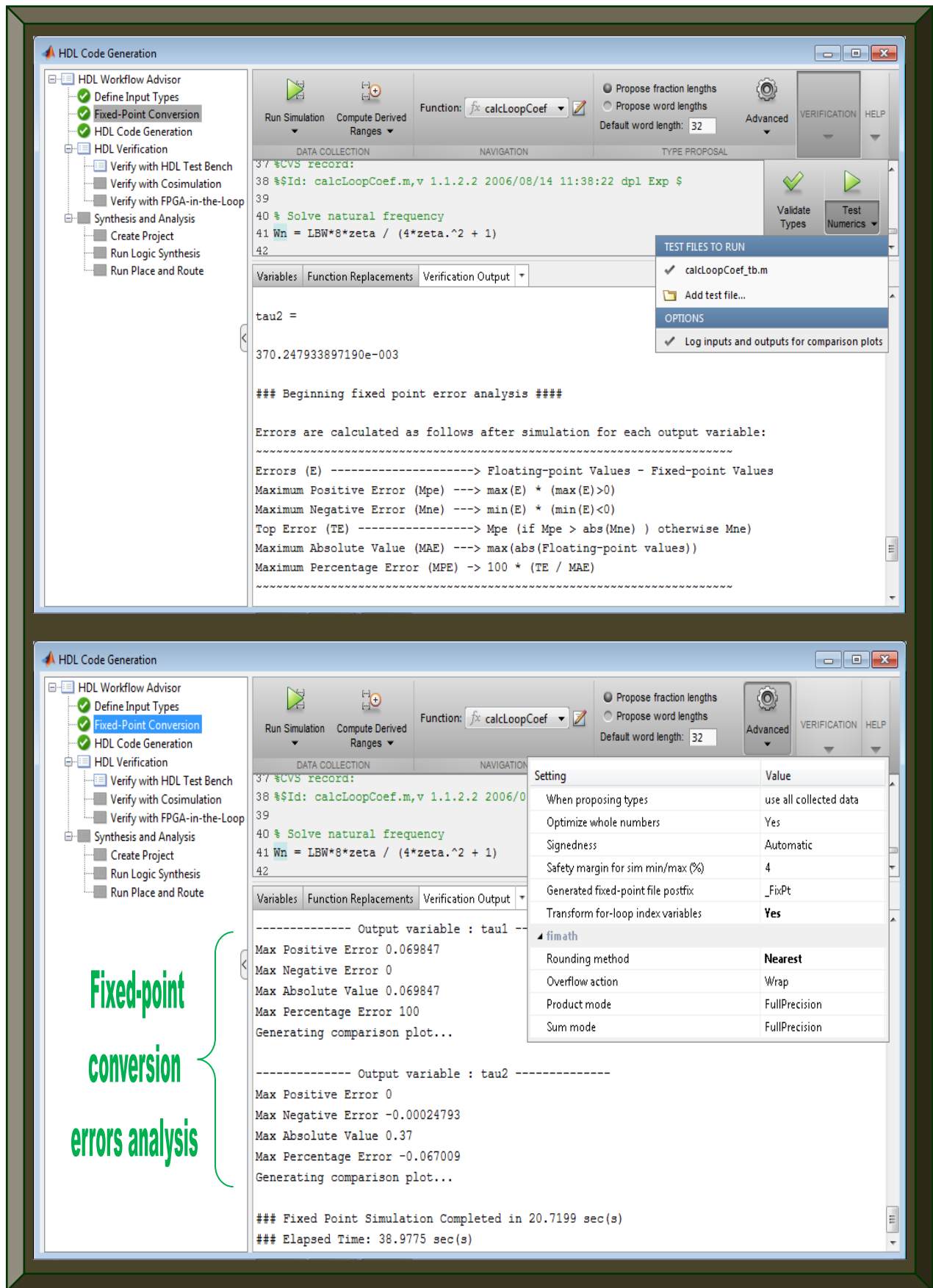


Figure 4.16: Illustration of Step 4; sub-steps 4 and 5 – Error analysis and test numerics

Figure 4.17 depicts the benefits and structure of Matlab floating-point to fixed-point conversion.

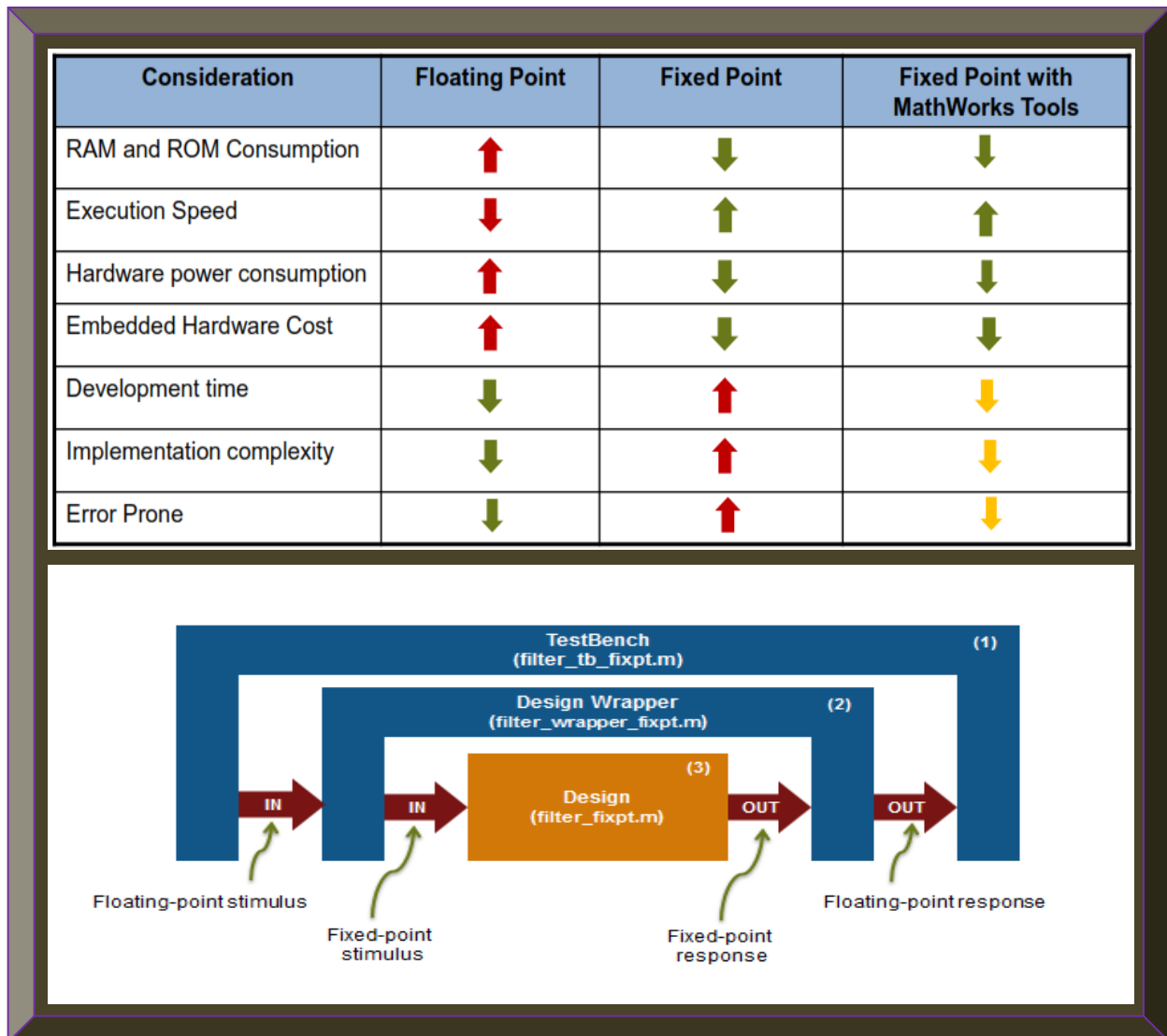


Figure 4.17: Matlab floating-point to fixed-point conversion benefits and structure

(Adapted from Dubey, 2013: 7; MathWorks, 2014)

The following summarises the steps performed during the fixed-point type validation process:

- The design files are converted to fixed-point to generate fixed-point Matlab algorithms.
- All user-written functions (NB: only one function can be converted at a time) called in the floating-point design are transformed to fixed-point and added in the generated design file.
- A new design wrapper file is created to convert the floating-point data values provided by the test-bench to the fixed-point types determined for the design inputs during conversion.
- The fixed-point values are then input into the converted fixed-point design.
- All the generated fixed-point files are subsequently stored in an output directory.
- The generated fixed-point designs are subsequently utilised for VHDL code generation.

4.3.2.1.5 Step 5: Generate GPS HDL Codes (Convert Fixed-point Matlab Algorithms to VHDL)

After successful completion of Step 4, the next step is to convert the fixed-point GPS Matlab codes to VHDL. During this process, the following options in the HDL code generation menu should be selected or ticked. NB: options shown will vary depending on the Matlab version used.

- In Matlab R2013a, select HDL Code Generation / Target tab / Target Selection:
 - Select VHDL language.
 - Tick the Check HDL conformance, generate HDL and generate EDA scripts check boxes.
- In the Coding Style tab:
 - Select the preserve the Matlab code comments check box.
 - Tick the include Matlab source code as comment and the generate report check boxes.
 - Under HDL / Coding Standards, select Industry.
 - Under Generated Files, ensure the default VHDL file extension is .vhd.
- View the Clocks & Ports tab to ensure settings are relevant for hardware implementation.
- In the Optimisation tab: Tick map persistent array values to RAM. Tick the distribute pipeline registers, enter 1 for the input and output pipelining options. Select stream loops.
- In the Advanced tab:
 - Select the RAM with clock enable option under the RAM architecture.
 - Tick the generate instantiable code for functions check box.
 - Under Simulink integration section, tick the generate Matlab function block check box.
- In the Script Option tab, simply review the default settings or skip it.
- Finally, navigate to Verify with HDL Test Bench:

In the Output Settings, tick generate HDL test bench and also Simulate with generated HDL test bench; with ModelSim as the simulation tool if installed. Tick all in Test Bench options.

Once finished with all the above settings, click the Run button to generate the VHDL codes. The execution time of the code generation process depends on the number of Matlab files involved and the size of respective files. Upon successful completion, click on the links to browse the generated GPS Matlab fixed-point and GPS VHDL algorithms. Otherwise, click and review any error or warning messages, apply the necessary corrections where applicable and re-run the step.

Next is to examine the generated untitled Matlab function block model(s) from the HDL code generation process. Open it to view the fixed-point Matlab code from the original floating Matlab design. Rename the model(s) with their corresponding GPS design names. The generated Matlab function block(s) can be simulated in Simulink. Optionally, HDL code can be generated from the Matlab function block by executing the Matlab command *makehdl* ('corresponding design file name'). Figures 4.18 - 4.22 show Step 5 results using the same file *calcLoopCoef.m*.

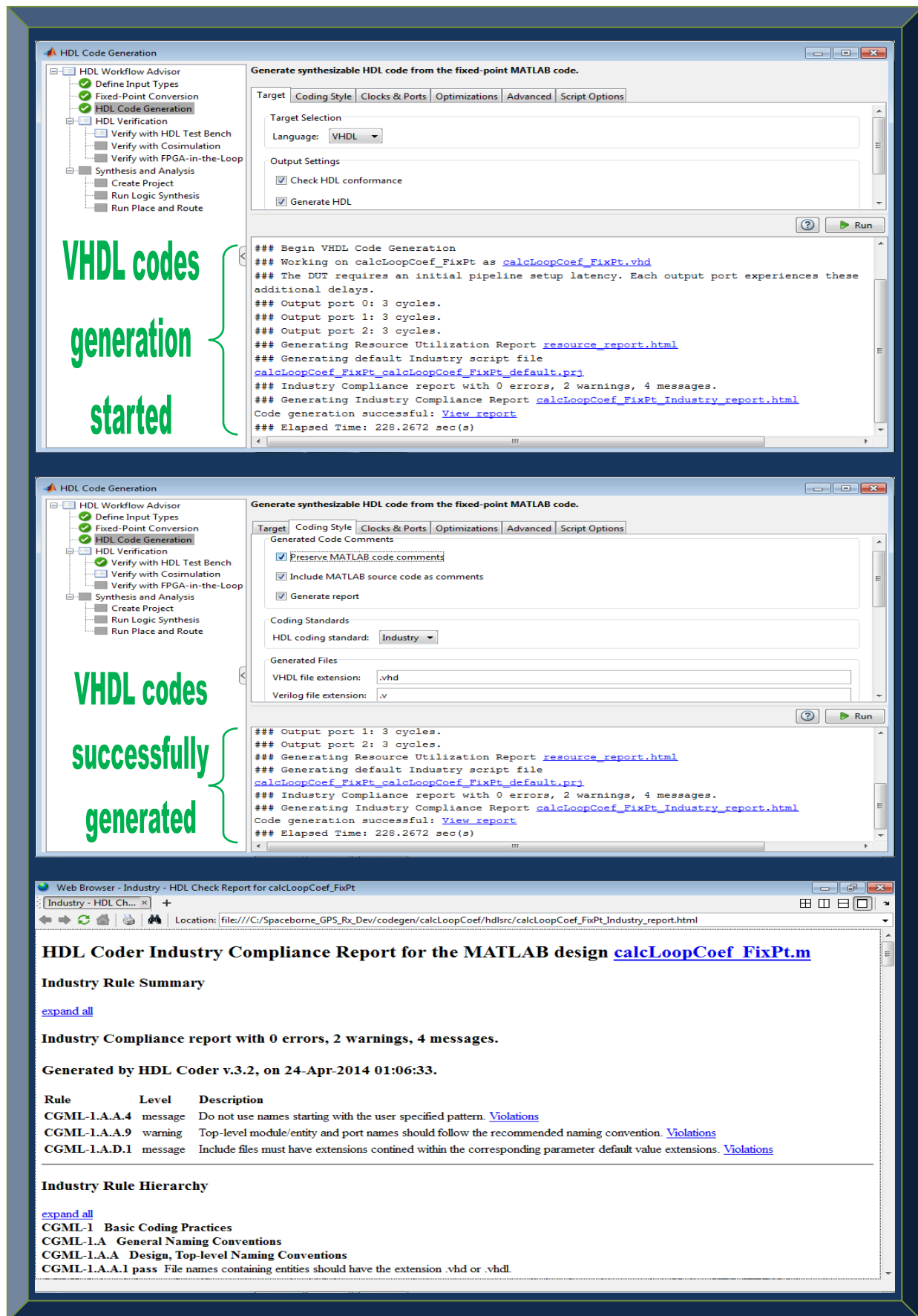


Figure 4.18: Illustration of Step 5 – HDL code generation settings

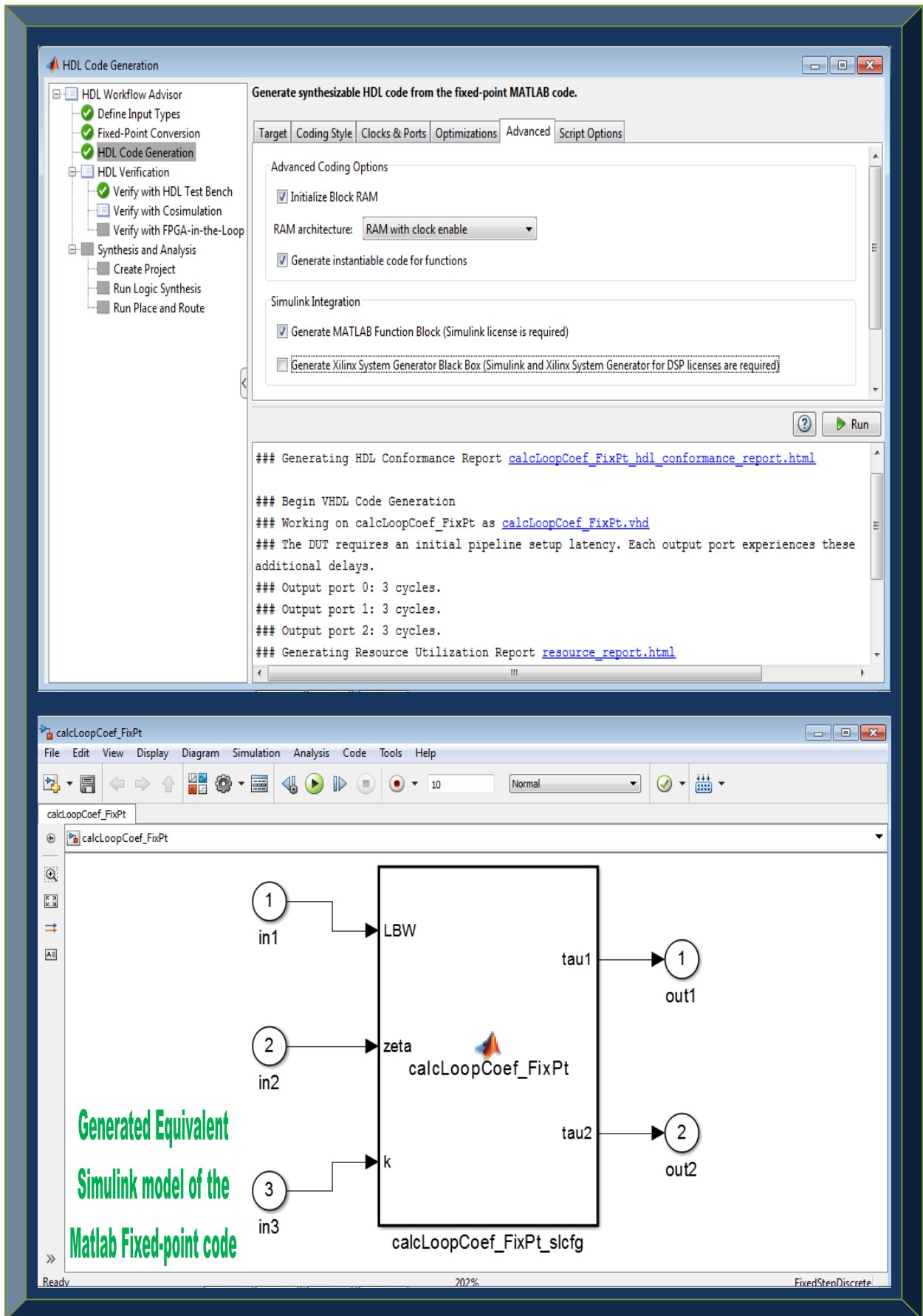


Figure 4.19: Illustration of Step 5 – Generated equivalent Simulink model

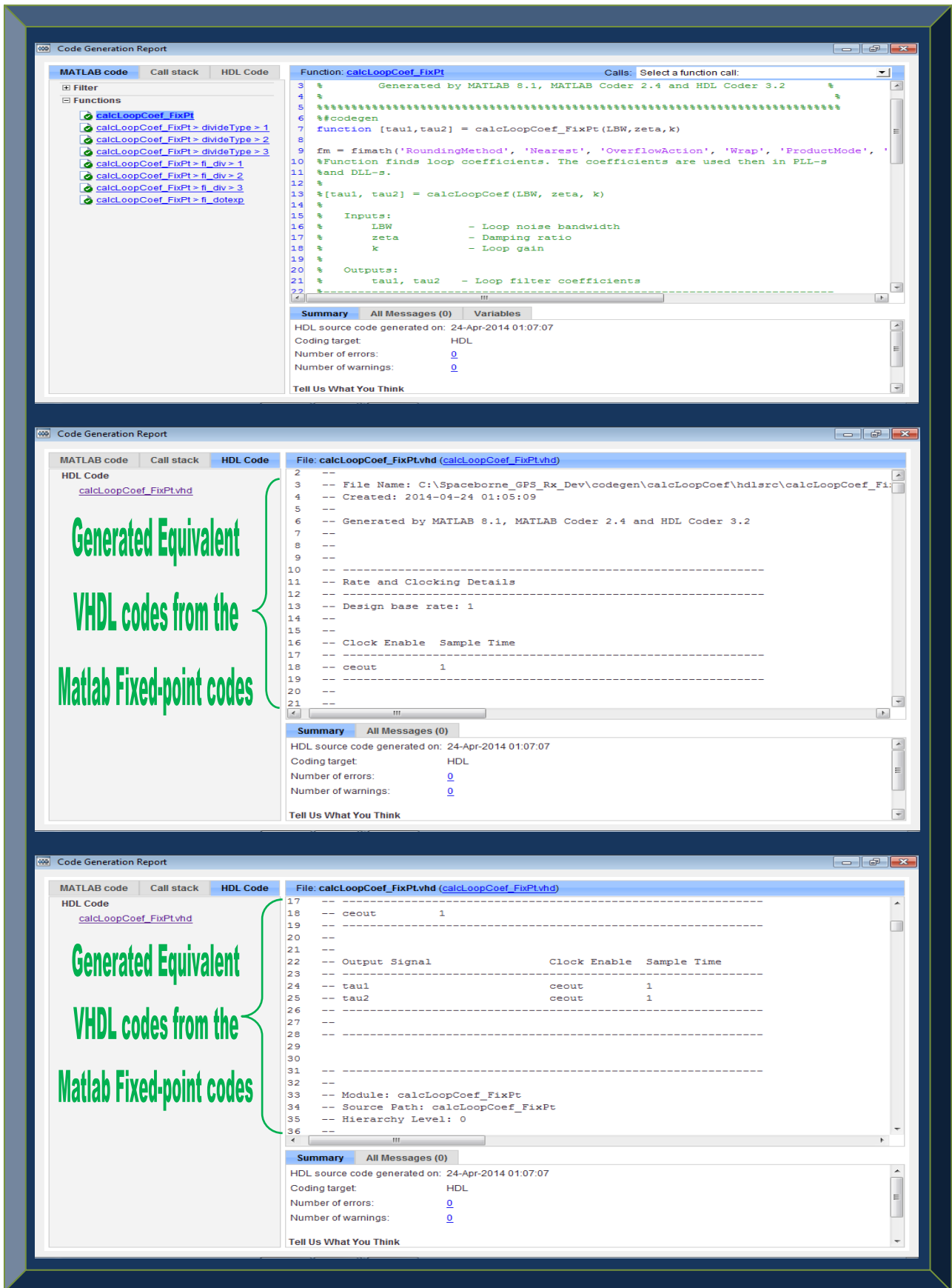


Figure 4.20: Illustration of Step 5 – Generated VHDL codes for `calcLoopCoef.m`

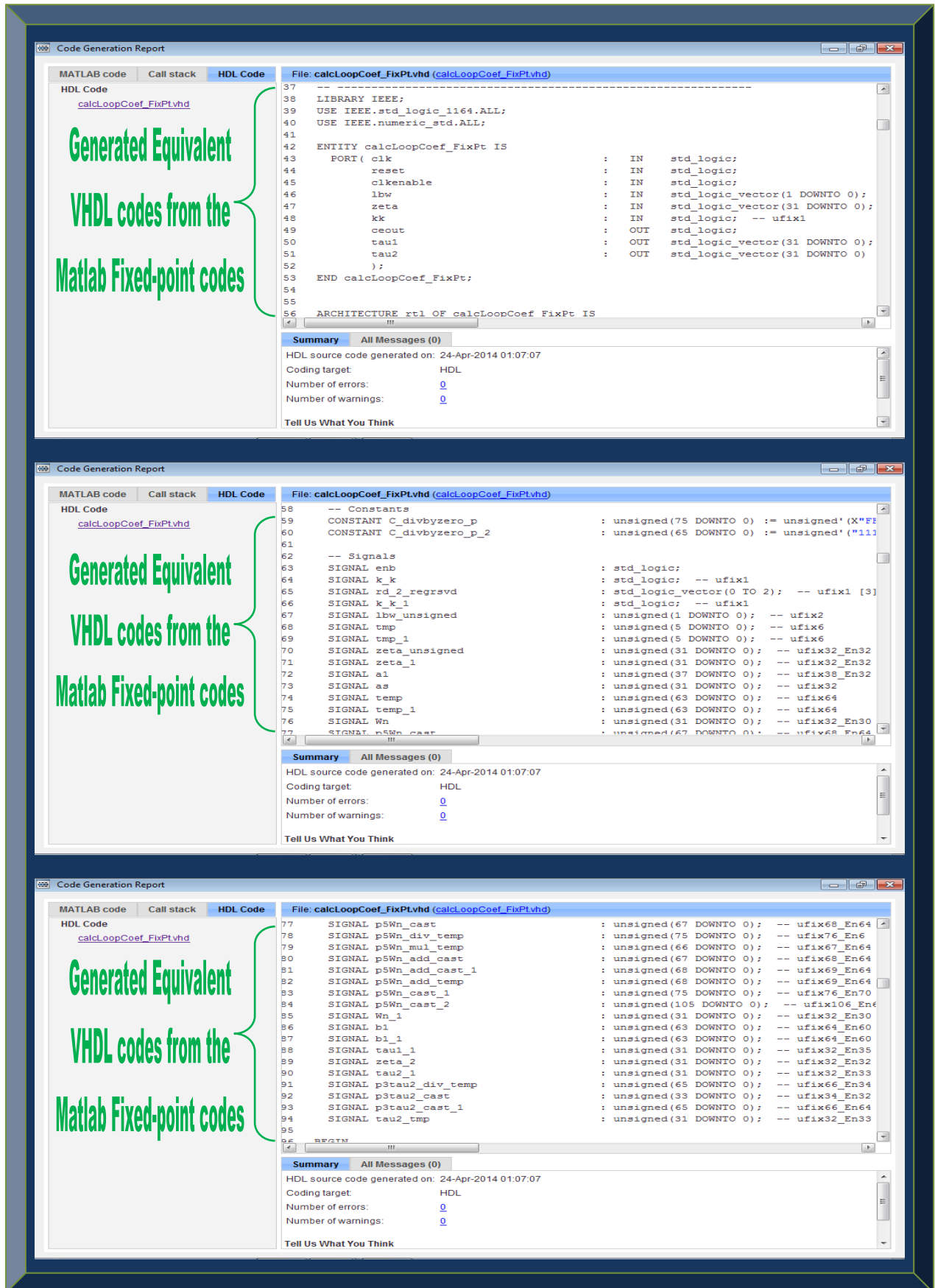


Figure 4.21: Illustration of Step 5 – Generated VHDL codes for calcLoopCoef.m continues

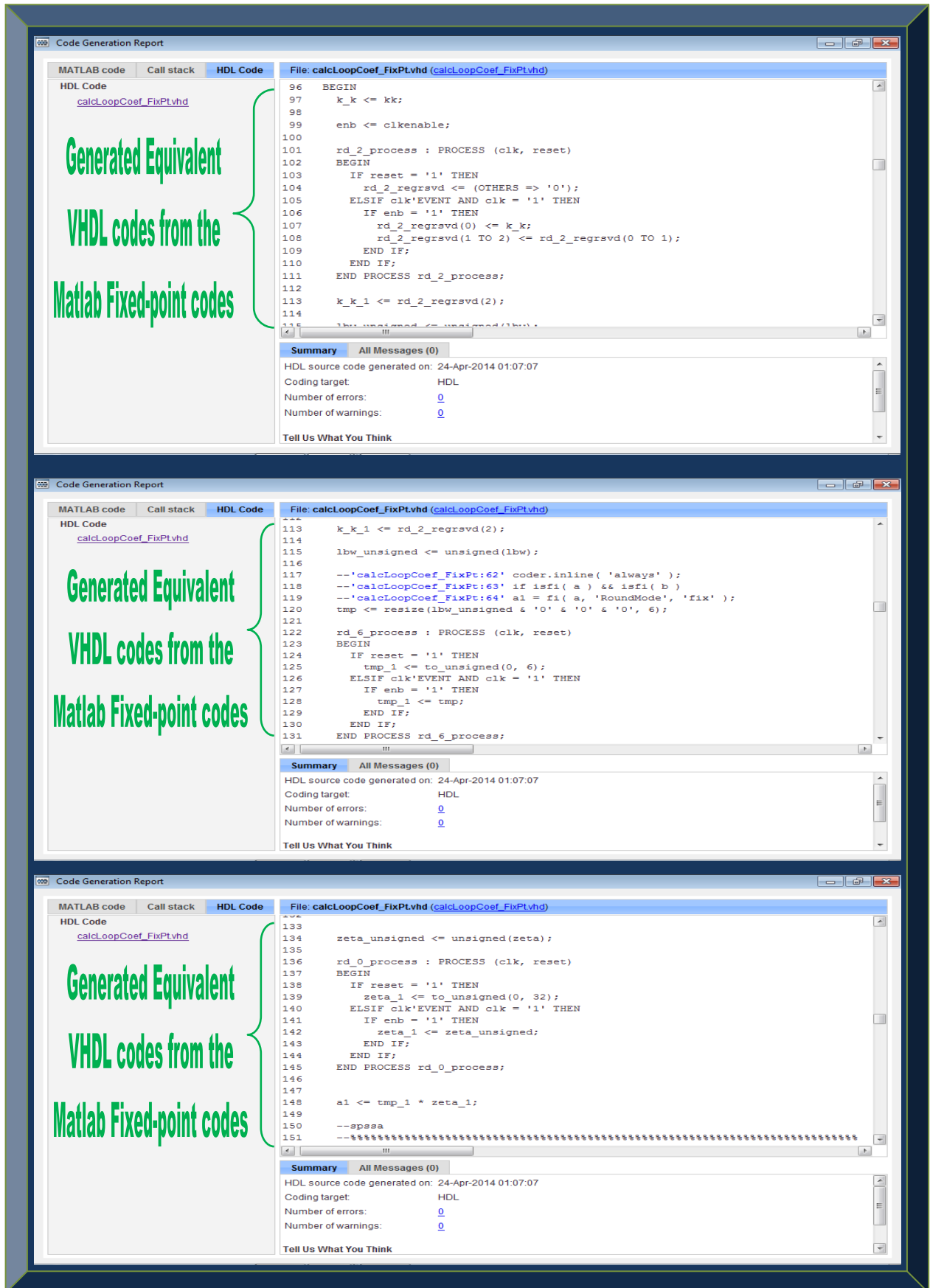


Figure 4.22: Illustration of Step 5 – End of generated VHDL codes for calcLoopCoef.m

4.3.3 Firmware-defined GPS Receiver: Phase 3 Based-on VHDL

This section examines how the converted GPS algorithms in VHDL are simulated and verified. It also provides a roadmap for synthesis and implementation on a FPGA. Co-simulation can be a challenging task, especially with automatically generated codes, as sync must be retained across various aspects of the source model. This includes sample rates, feed-forward and feed-through systems and other various parameters, as well as settings used during code generation while setting up the HDL Verifier block and the target EDA Simulator Link (e.g. ModelSim-Altera).

The automated co-simulation model generation takes the guess-work out of the HDL co-simulation block and simulator setup, by deciphering all the compiled model and code generation information. In addition, all the automated settings are documented in the generated scripts. The end result is a co-simulation model ready to verify the generated VHDL code in software.

EDA Simulator Link is a co-simulation interface which presents a bi-directional link between Matlab, Simulink and HDL simulators from Mentor Graphics, Cadence and Synopsys; thereby enabling verification of VHDL, Verilog and mixed-language implementations. It enables Matlab codes or Simulink models to be used as test-benches to generate stimulus for an HDL simulation and then analyses the response of the simulation. It also enables replacement of several HDL components with Matlab codes or Simulink models, enabling simulation of the whole system before all the HDL design elements are available. EDA Simulator Link also facilitates interactive and batch-mode co-simulation on a single computer, across heterogeneous platforms and networks. The three workflows that support Matlab codes and HDL simulations are:

- Matlab test-bench.
- Matlab algorithm or design (function or component).
- Simulink co-simulation.

4.3.3.1 Simulation using Matlab Test-bench Workflow

With this approach, Matlab is the server while the HDL simulator is the client. However, timing is done by the HDL simulator since Matlab algorithms are un-timed. Multiple Matlab test-benches or designs running on the same machines or different remote servers can be connected to the HDL simulator. A Matlab server can as well interface to several HDL simulators entities. Furthermore, the capability of Matlab test-bench permits reuse of the Matlab test-bench that was created to verify the Matlab algorithm(s) in Step 2, to be used as well to verify the generated VHDL implementation of the Matlab algorithms. The algorithms can be compared with a golden reference for the algorithm(s) in the co-simulation. Figure 4.23 depicts this workflow structure.

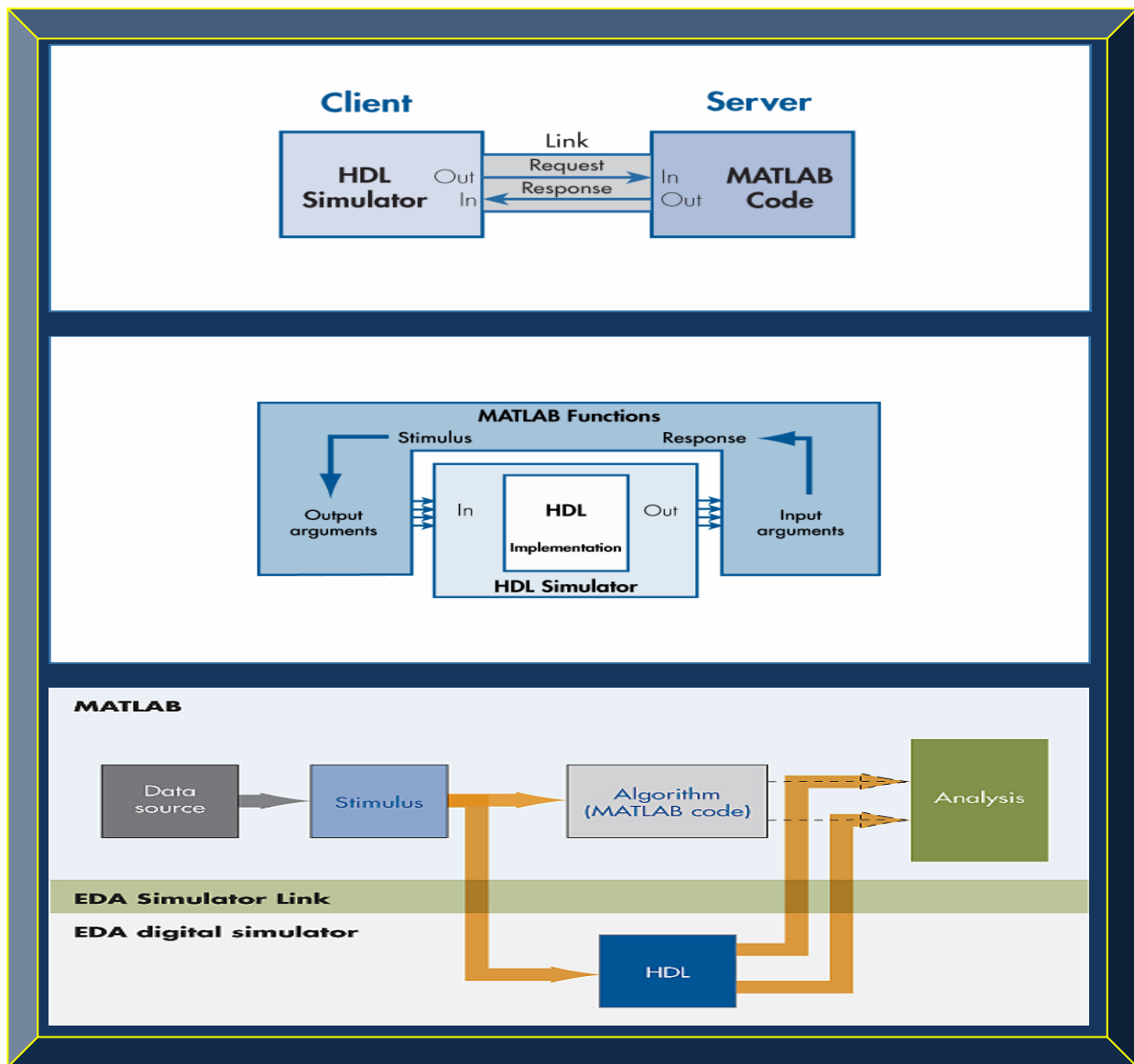


Figure 4.23: Matlab test bench stimulating VHDL simulation
 (Adapted from MathWorks, n.d.: 2 - 4)

From Figure 4.23, the Matlab test-bench code provides on-the-fly stimulus to drive and analyse the implementation, while the HDL implementation block runs in the HDL simulator.

4.3.3.2 Simulation using Matlab Algorithms Workflow

With the Matlab algorithm or component capability simulation approach, Matlab is the server whereas the HDL simulator is the client. The Matlab algorithm or function, substitutes the HDL algorithm or component in an HDL simulation. That is, the Matlab algorithm is used in place of entities not yet coded or converted in HDL, enabling simulation of the entire system before all the HDL design elements are available. The Matlab algorithm or design under test, replaces a golden reference (either a Matlab function code or a Simulink model that is still unavailable or being developed) for the algorithm in the co-simulation. Figure 4.24 illustrates this approach.

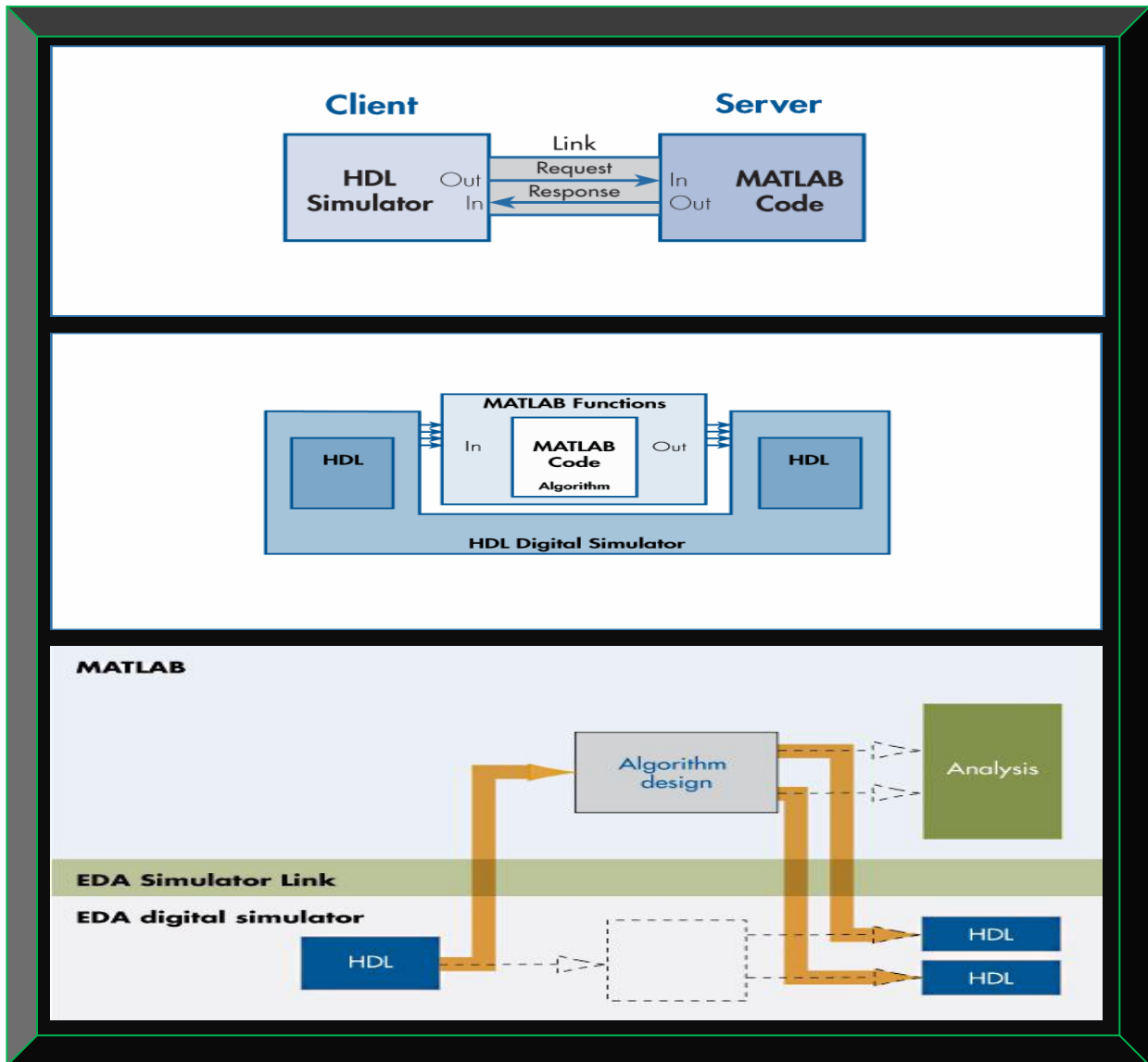


Figure 4.24: MATLAB algorithm used in VHDL simulation

(Adapted from MathWorks, n.d.: 2 - 4)

4.3.3.3 Simulation using Simulink Co-simulation Workflow

With Simulink co-simulation capability, a HDL simulator can be connected to a Simulink co-simulation block, whereby the Simulink model is the client and the HDL Simulator is the server. In this setup, Simulink is synchronised with the HDL simulator by scaling Simulink time to HDL simulator time. Furthermore, Simulink model can be connected with one or more co-simulation blocks, to one or more HDL simulators. System level depictions of communications, signal processing and other systems can be created with Simulink and related blocksets in this manner. In this thesis, the Simulink models are Matlab Function Blocks and can be used to create GPS software test-benches for VHDL implementations or substituted as algorithms in place of VHDL blocks or subsystems for VHDL debugging and verifications. Figure 4.25 depicts this approach.

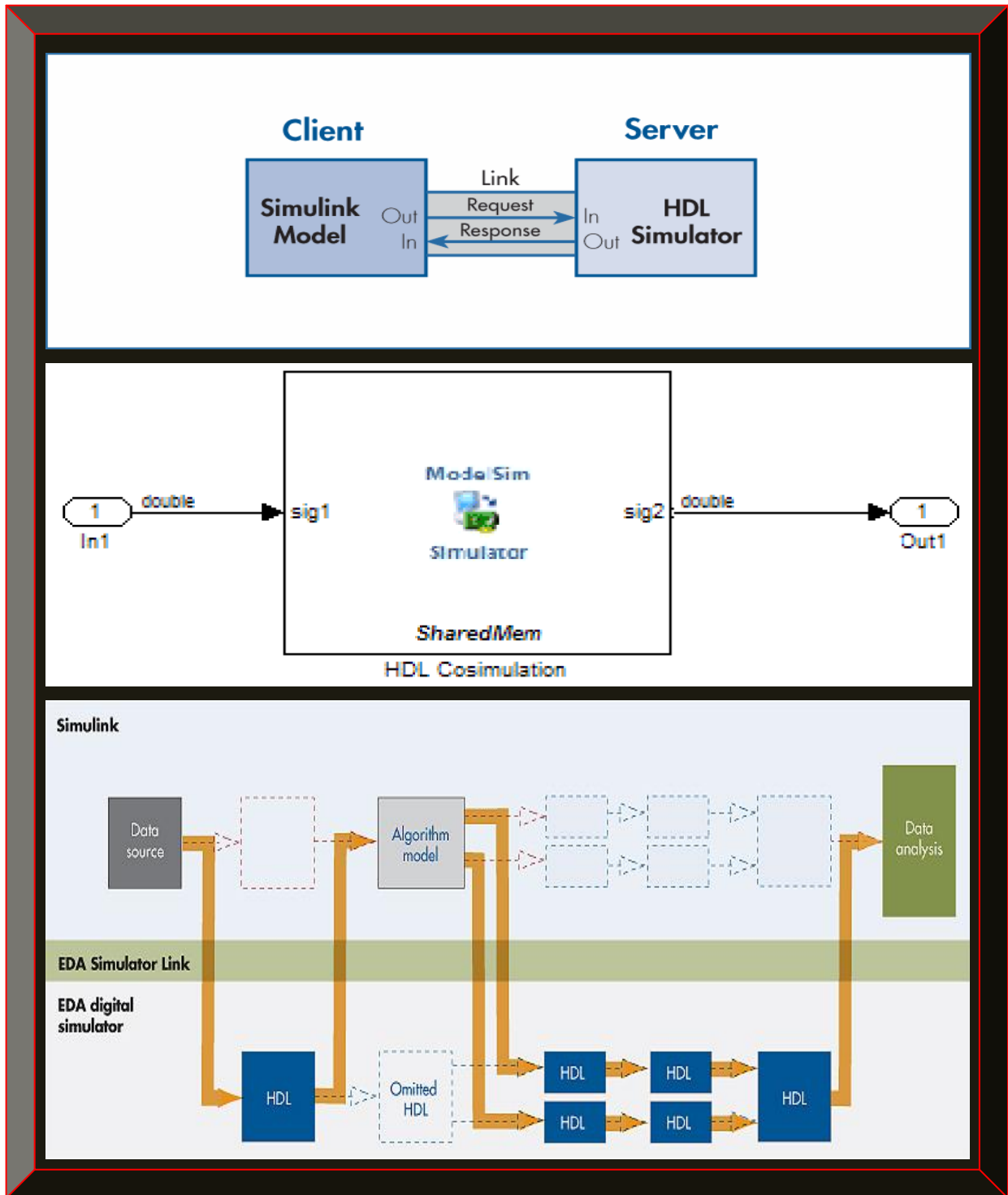


Figure 4.25: Bi-directional Simulink model and HDL simulation

(Adapted from MathWorks, n.d.: 2 & 3)

From Figure 4.25, standalone Simulink co-simulation blocks can be used to instantiate available HDL implementations into a Simulink model, to create a mixed Simulink system model with a HDL simulation as a bidirectional test-bench and component. This approach requires a license.

4.3.3.4 Verification with FPGA-in-the-loop Wizard

Sections 4.3.3.1 to 4.3.3.3 presented three possible methods for simulating, as well as verifying the generated GPS VHDL codes in software. In addition to the software-to-firmware implementation and verification workflows described in Sections 4.3.3.1 to 4.3.3.3 and depending on the Matlab version used, one or a combination of these methods can be achieved easily with HDL Verification Workflow Wizard, as well as FPGA-in-the-loop (FIL) verification, if supported. FIL in contrary to the three simulation and verification methods, enables simulating and verifying the generated GPS VHDL codes as software in hardware, using either a supported Xilinx or Altera FPGA development kit, to implement a HDL firmware-defined GPS algorithm.

4.3.3.5 Simulating and Verifying the Generated VHDL GPS Receiver Algorithms

The simulation / verification approach utilised in this thesis is the method described in Section 4.3.3.1. The VHDL test-bench used is generated from the original Matlab test-bench. The other approaches required a paid license software or hardware. The free software used to simulate and verify these generated VHDL GPS algorithms is the ModelSim-Altera, illustrated in Figure 4.26.

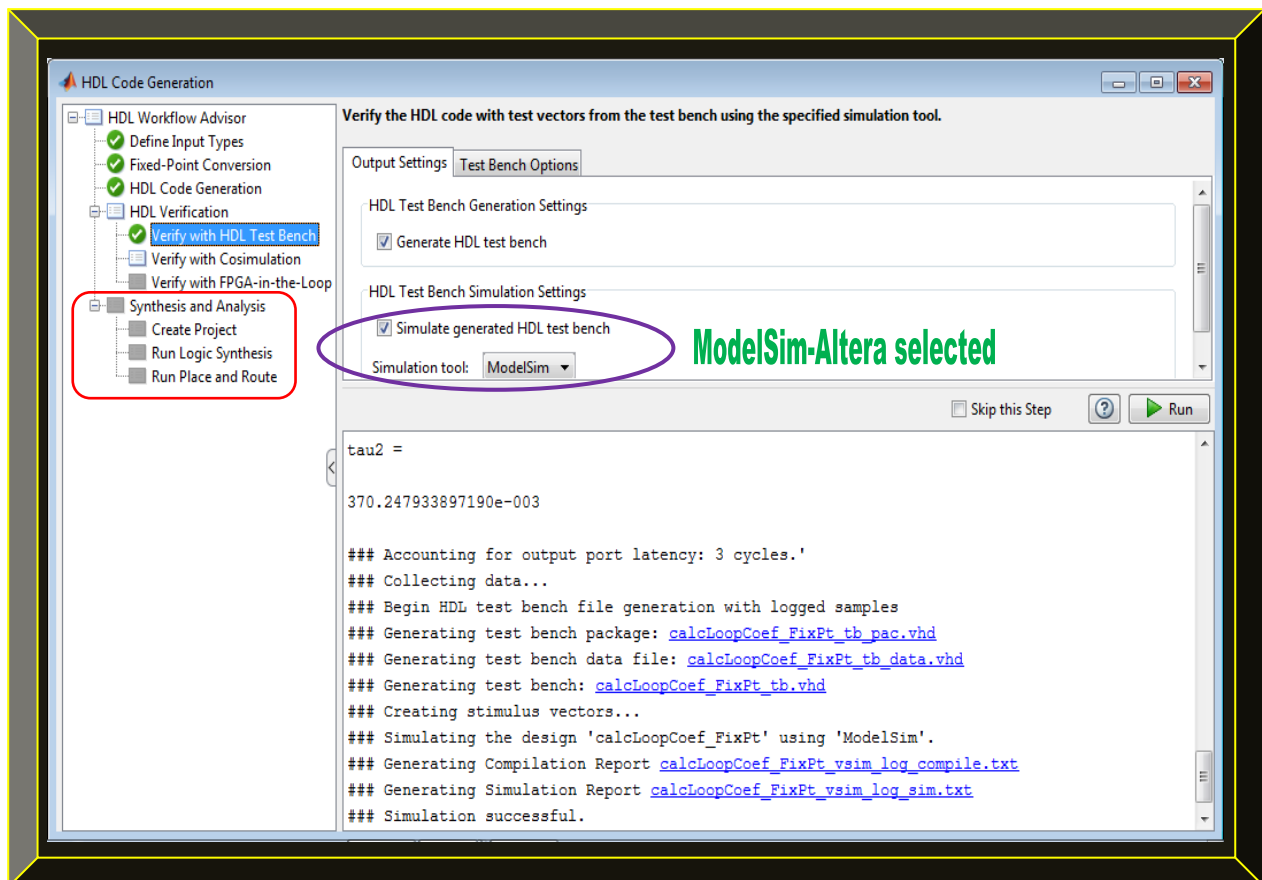


Figure 4.26: Firmware GPS algorithm simulation and verification with HDL test-bench

4.3.4 Low-cost Space-borne GPS Receiver Hardware Implementation Roadmap

This section postulates a low-cost hardware implementation of the space-borne GPS receiver. Once the firmware GPS receiver VHDL algorithm(s) are simulated and verified as outlined in Section 4.3.3.4, the next step (which is supported by Matlab version R2013a and shown in red in Figure 4.26), is to synthesis, place & route and finally generate a programming file that can be downloaded into the targeted FPGA. However, this hardware implementation is not practically covered in this thesis but a roadmap is proposed. In a Danish GPS receiver research (Group 09gr944, 2009), a GPS receiver algorithm based-on Matlab / Simulink, were ported to Xilinx Virtex 5 FPGA – 5VFX70TFF1136-3 chip, using Xilinx System Generator ISE Tool and ML507 EK. Similarly, with further development and optimisation, porting this thesis firmware GPS receiver VHDL algorithm(s) to the Xilinx low-cost space-grade Virtex-5QV (XQR5VFX130) FPGA is realistic, based-on the presented workflow in Section 4.3 and abridged in Figure 4.27.

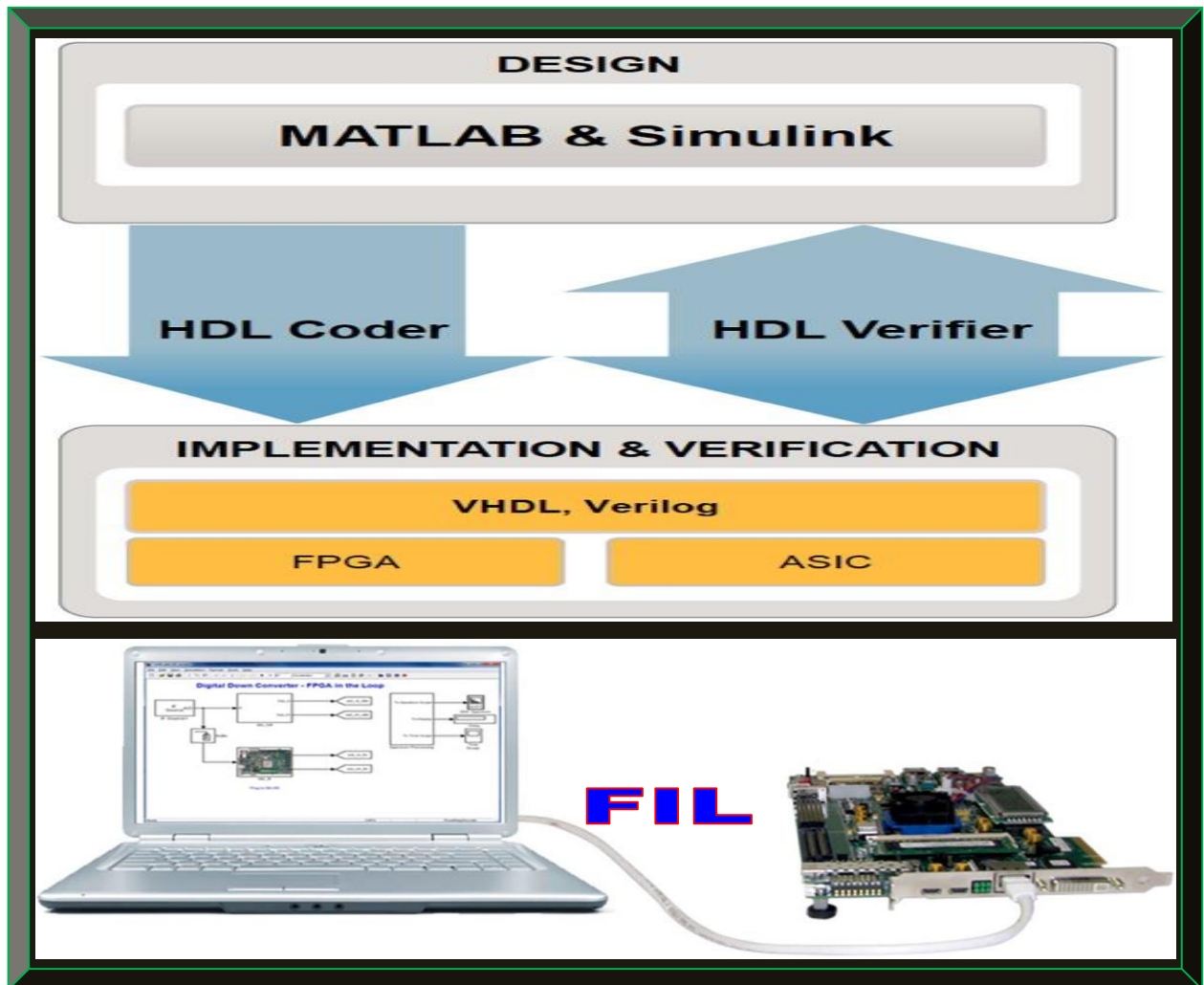


Figure 4.27: Research methodology implementation summary

(Adapted from Kumar, 2012: 13 & 36)

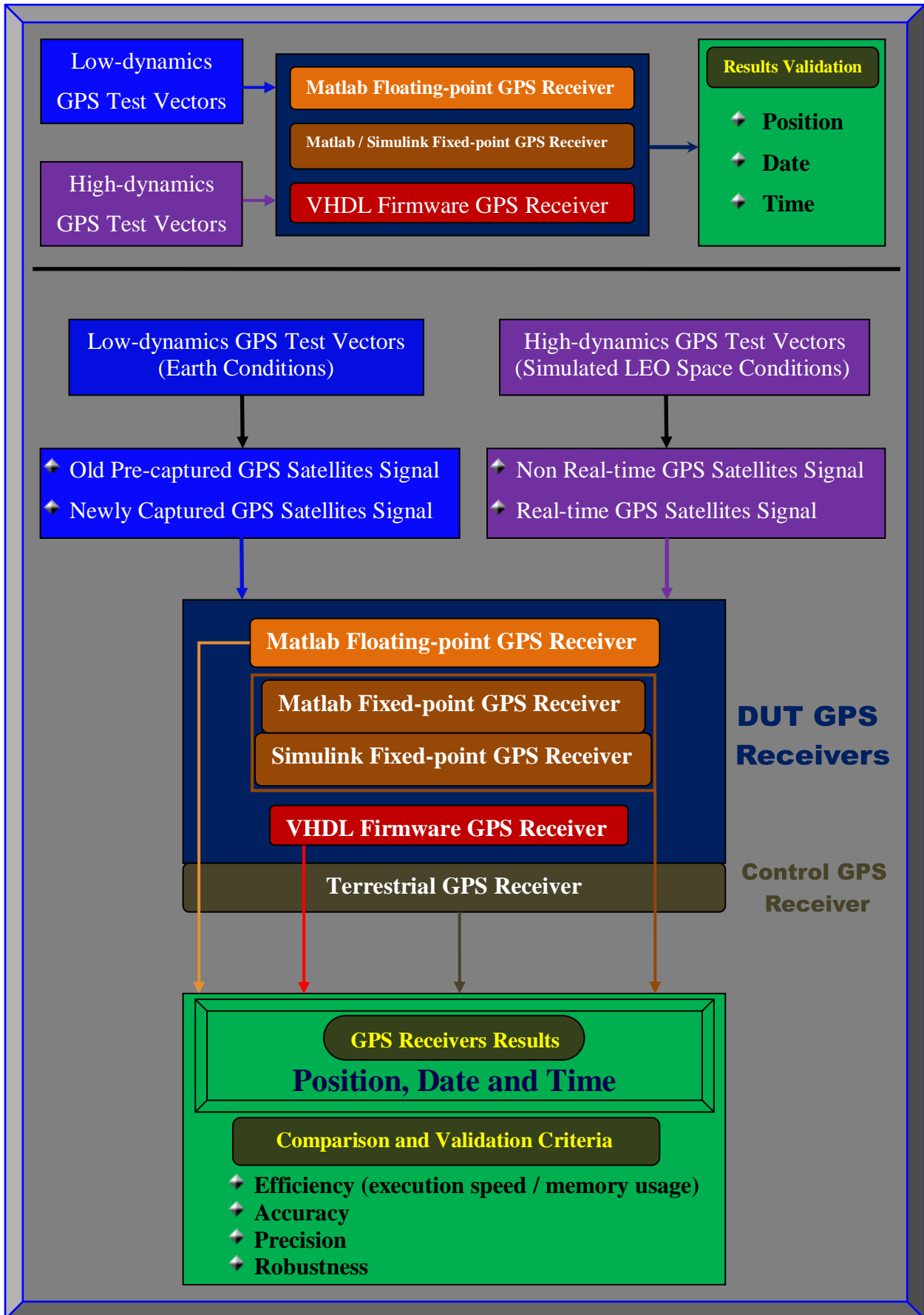


Figure 4.28: GPS receivers' test and validation setup procedure

4.4 GPS Receiver Algorithms Test and Validation Procedure

This section presents a harmonised functional test setup for the researched GPS receivers. The test and validation procedure starts by using a low-dynamics pre-captured real GPS signal (of filename *Multipath.bin*, obtained from Danish GPS Centre website) as a test vector to stimulate (either discretely or as an integrated unit if possible) the researched GPS receiver algorithms (i.e. software-defined floating-point GPS receiver in Matlab, software-defined fixed-point GPS receiver codes in Matlab/Simulink and the firmware-defined GPS receiver algorithms in VHDL).

The expected outcome with focus on efficiency and accuracy is to have the same or a result that correlates and validates the various types of GPS receiver algorithms discretely, as well as corresponding correlating results of the GPS receivers' algorithms, if tested as an integrated unit.

Following the low-dynamics test and validation using Danish GPS Centre dataset (which serves mostly as a control test and validation setup), is to use a real-time software GPS receiver front-end (see Table 3.2) to capture a live orbiting GPS satellites signal and use it as a stimulus to the various GPS receivers algorithms – discretely and as an integrated unit. The final navigation solution outcome, with focus on efficiency and accuracy, should be examined and compared with a navigation solution from a commercial GPS receiver (use as a control test and validator).

The two low-dynamics tests and validation procedures described thus far, simply serve to qualify the performance of the implemented GPS algorithms, with focus on their functional correctness.

Finally, the next test and validations procedures are to subject the various GPS receiver algorithms, with focus on the firmware-defined GPS receiver VHDL algorithms (using the other algorithms types as control), to a LEO simulated high-dynamics (space-borne) GPS receiver environment, using a non-real-time and then a real-time test vector (datasets) with extreme Doppler shifts, LEO altitudes, interferences and ionospheric conditions. This is to test and validate the navigation solutions (latitude, longitude, altitude, time, date and velocity) of the researched GPS algorithms in terms of their efficiency, accuracy, precision and robustness. Figure 4.28 depicts the full harmonised tests and validation setup for the GPS receivers as a unit.

4.5 Summary

This chapter demonstrated the approach for creating four equivalent types of GPS receivers' algorithms from a single design. The implementations were based-on the Matlab HDL workflow, with extensive use of the HDL Coder and the HDL Verifier toolboxes. Emphasis was placed on software-to-firmware conversion of Matlab GPS codes and the simulations and validation setup.

CHAPTER 5

GPS RECEIVER / ALGORITHMS RESULTS AND INTEPRETATIONS

5.1 Introduction

Chapter 5 examines the researched GPS receiver results and interpretations. That is, the results and the interpretations of the software GPS receiver algorithms in floating-point Matlab, their corresponding equivalent fixed-point Matlab and Simulink models, as well as the final VHDL codes. Since only the floating-point Matlab algorithms work as an integrated GPS receiver unit, the generated equivalents are evaluated and compared discretely. Section 5.2 only presents and highlights the various results, whereas Section 5.3 interprets and explains the results in detailed.

5.2 Researched GPS Receiver and the Discrete Generated Algorithms Results

This section only depicts the software-defined GPS receiver results and where possible, the three generated equivalents GPS algorithms types (i.e. to fixed-point in Matlab, Simulink and VHDL).

5.2.1 Software-defined GPS Receiver Results Based-on Matlab Floating-point Algorithms

The results of each of the processing stages (i.e. acquisition, tracking and navigation) are shown.

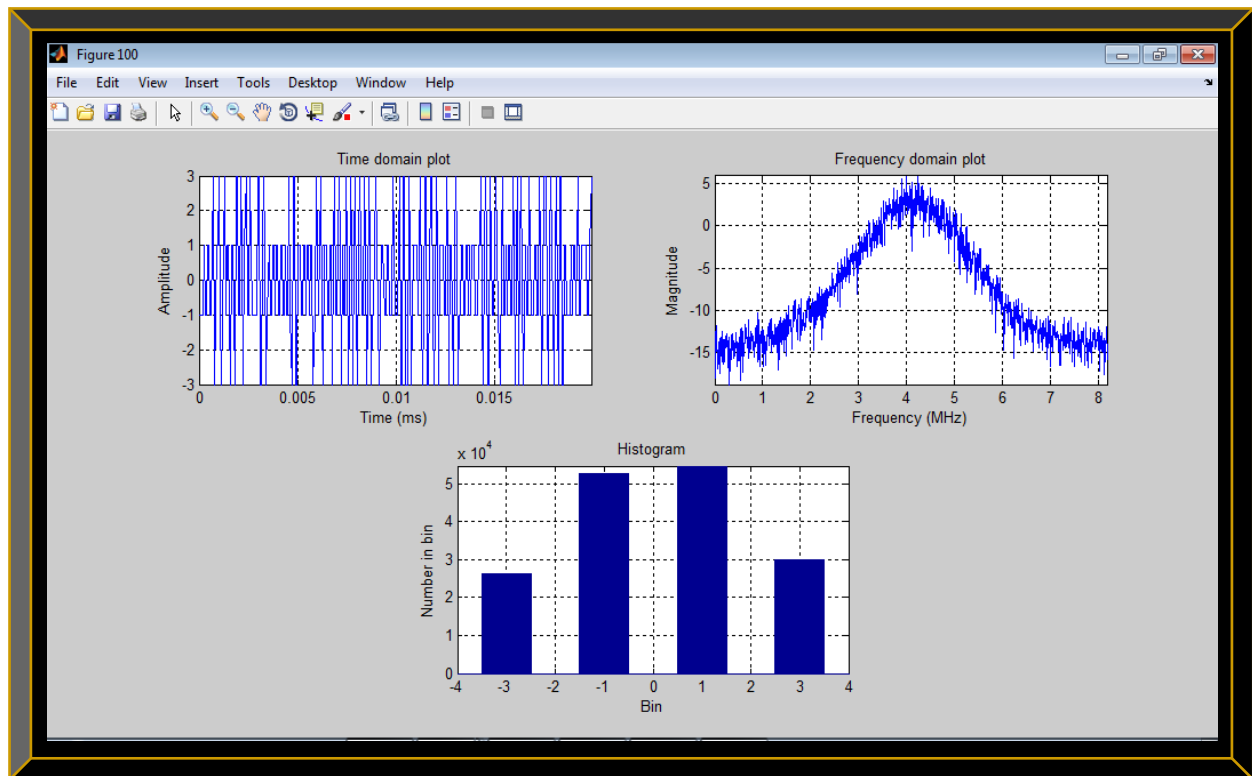


Figure 5.1: Three different plots of the recorded GPS signal (Multipath.bin dataset)

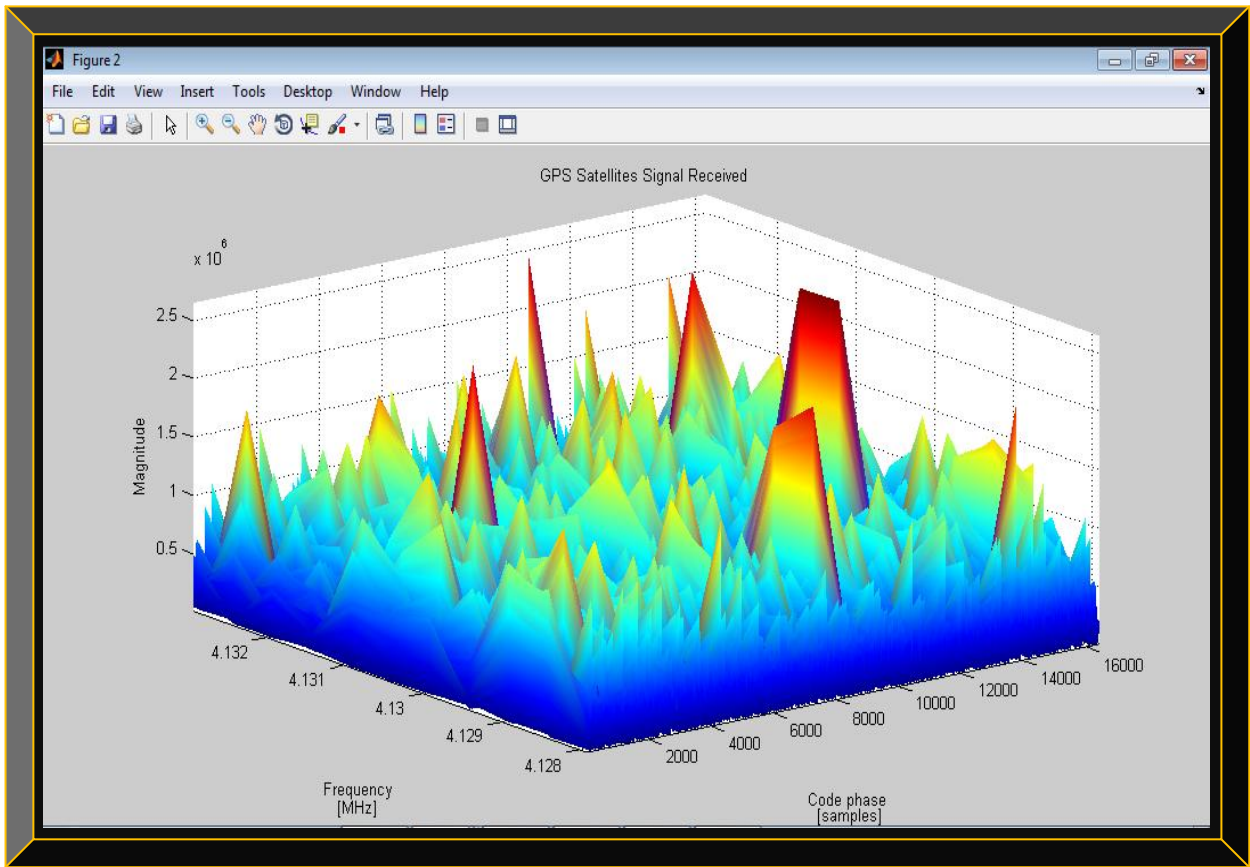


Figure 5.2: 3D plot of all the acquired GPS L1 C/A signals (with ± 5 kHz Doppler search bins)

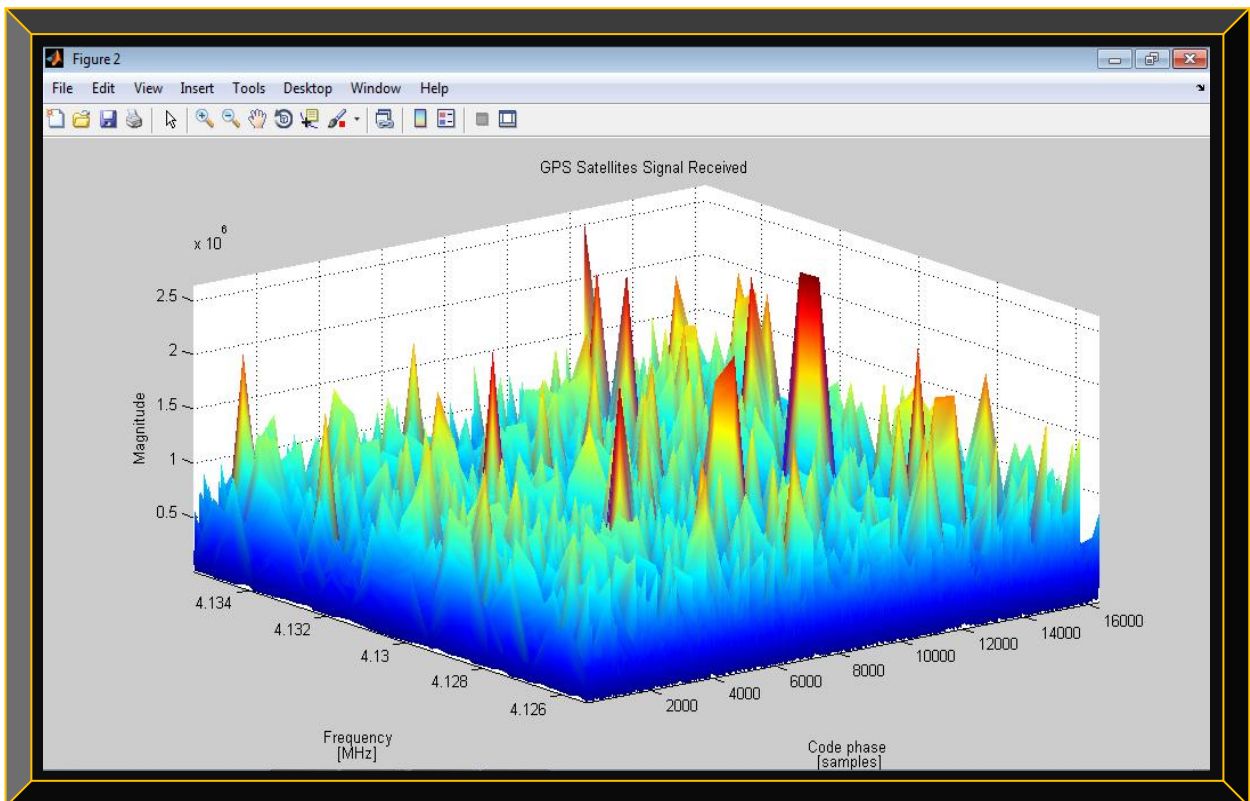


Figure 5.3: 3D plot of all the acquired GPS L1 C/A signals (with ± 10 kHz Doppler search bins)

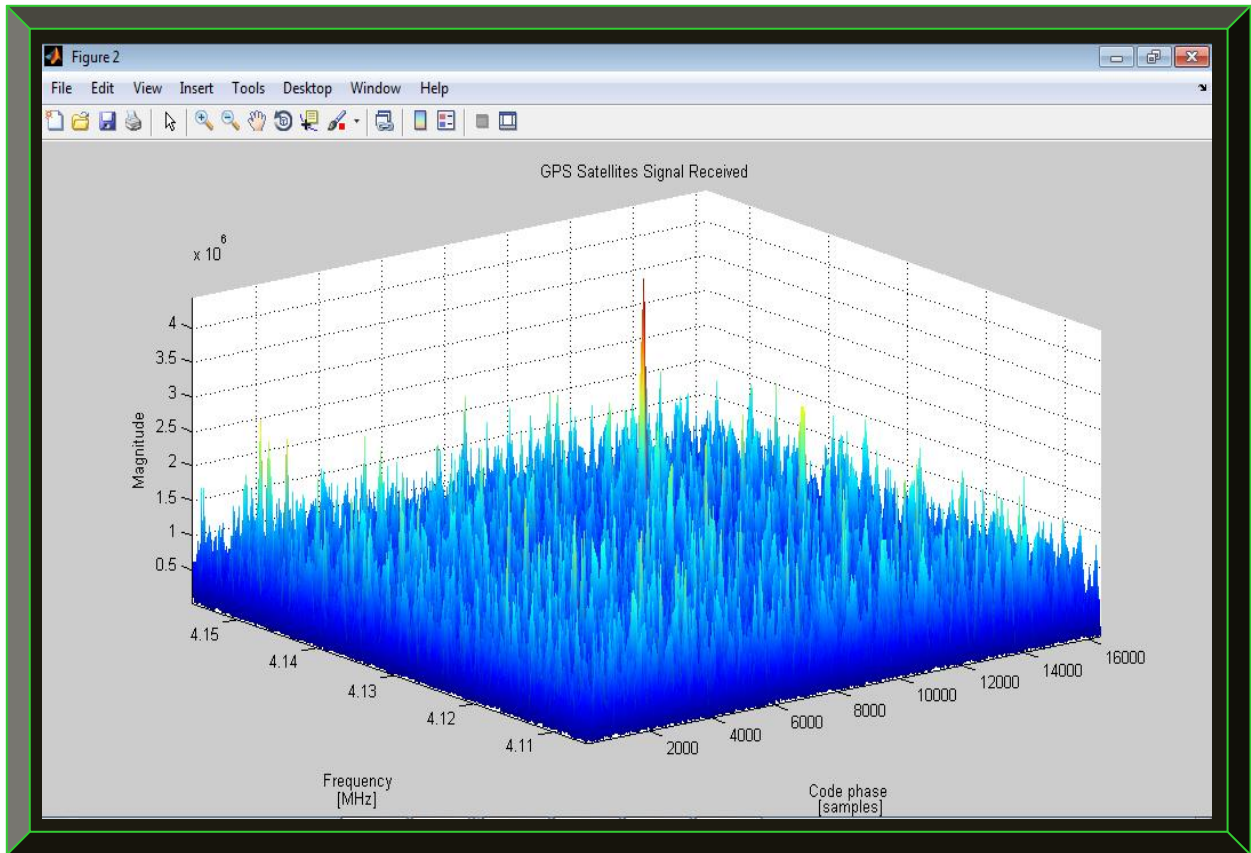


Figure 5.4: 3D plot of all the acquired GPS L1 C/A signals (with $\pm 50\text{kHz}$ Doppler search bins)

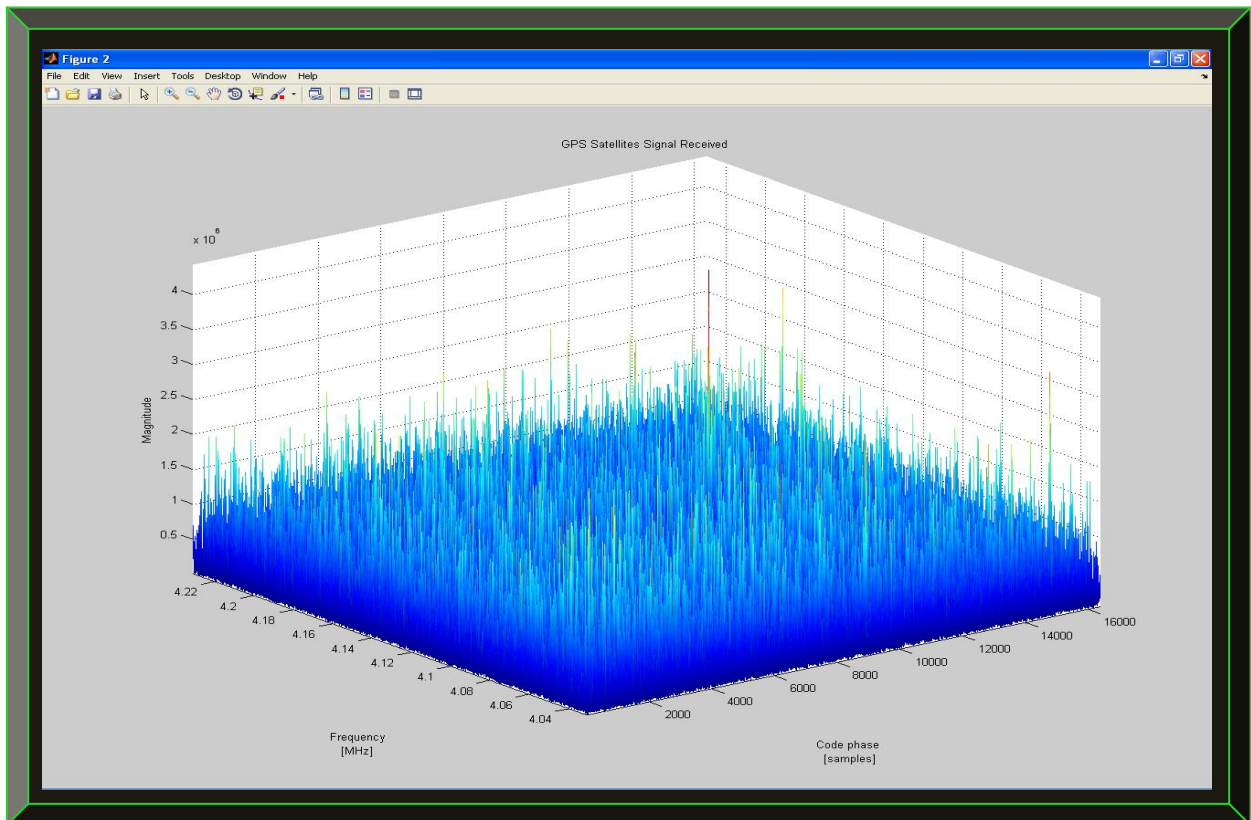


Figure 5.5: 3D plot of all the acquired GPS L1 C/A signals (with $\pm 100\text{kHz}$ Doppler search bins)

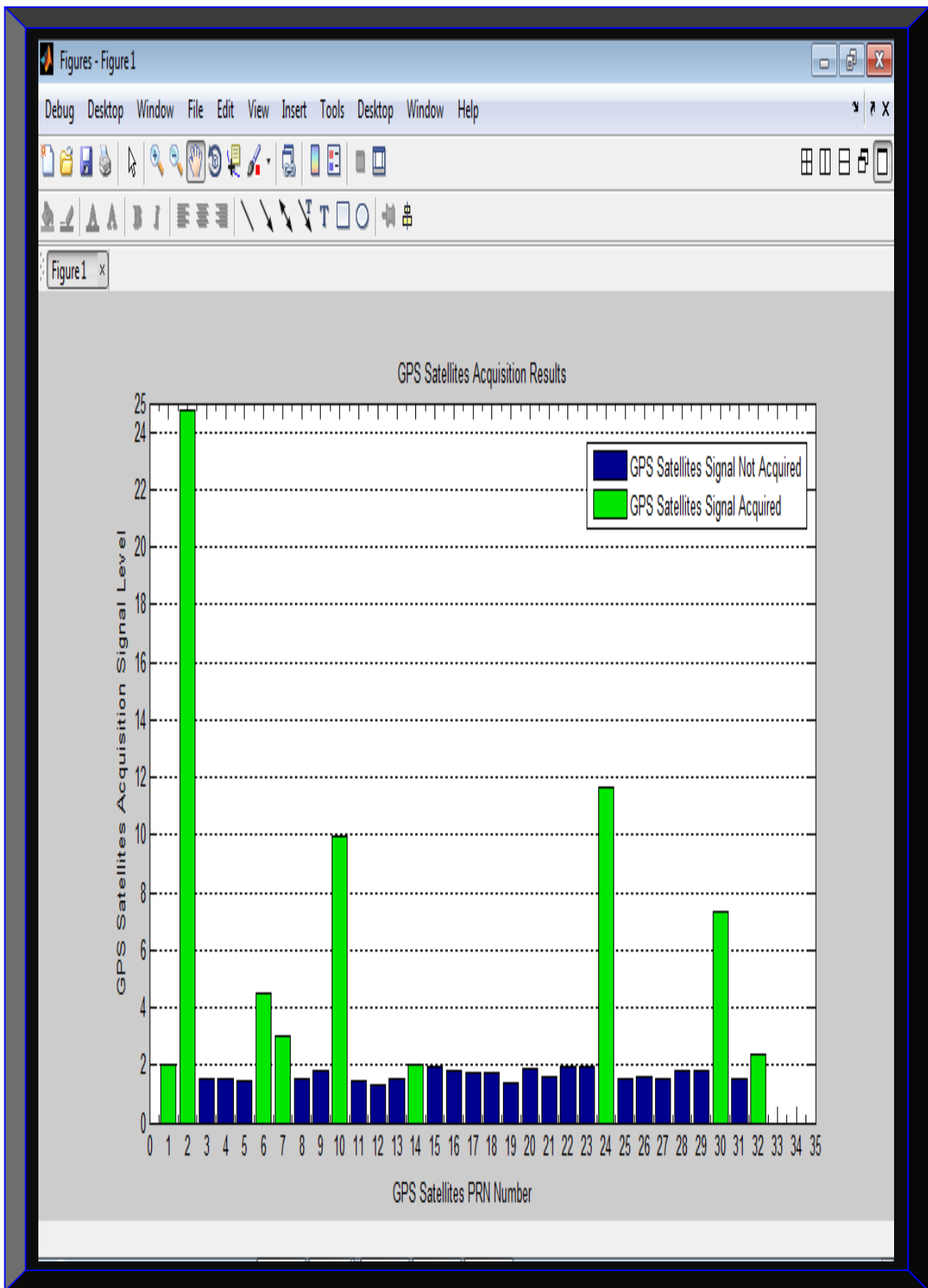


Figure 5.6: GPS satellites PRN (above threshold of 2) present in the acquired data at $\pm 100\text{kHz}$ Doppler

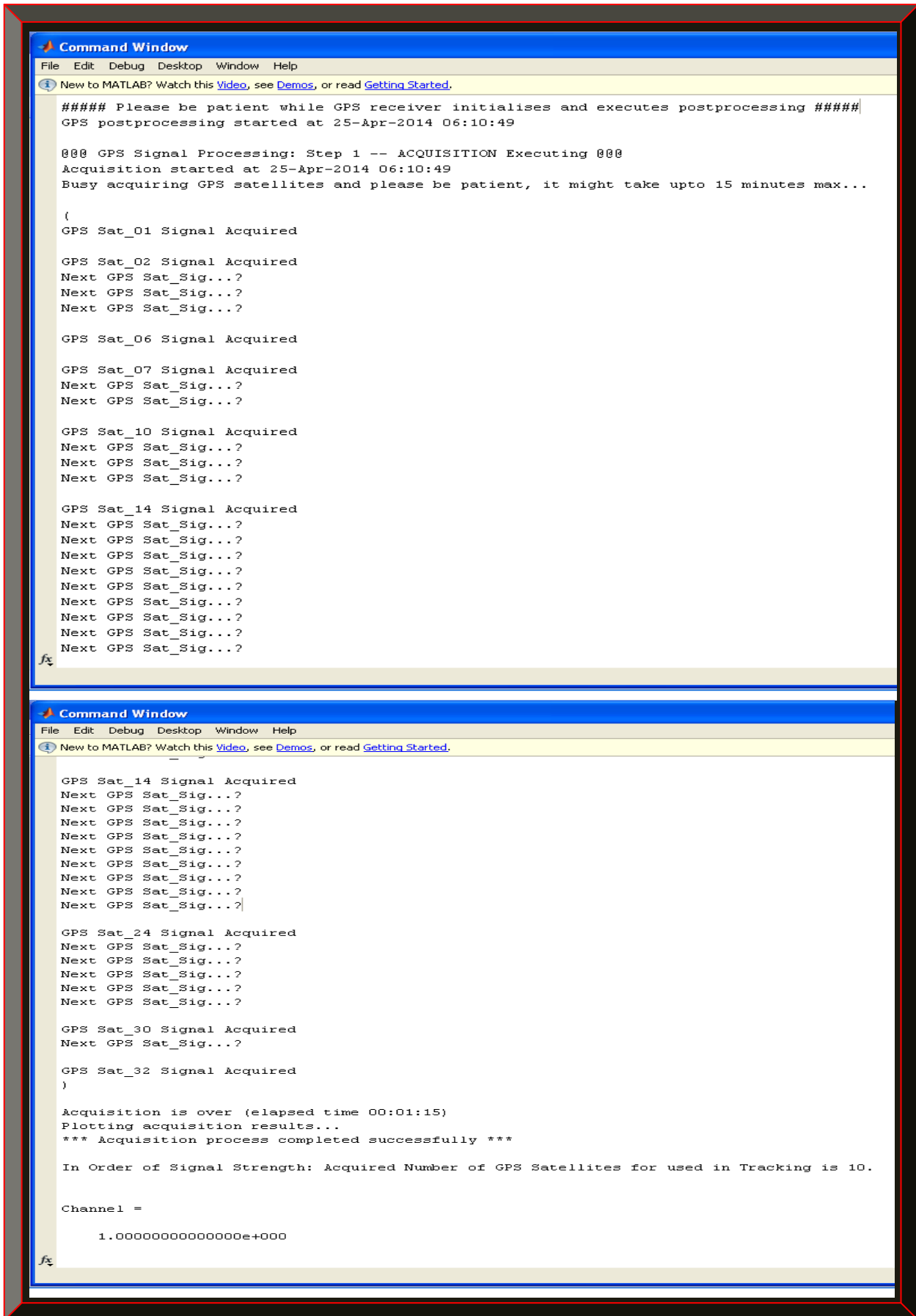


Figure 5.7: Illustration of the GPS satellites acquisition process during processing

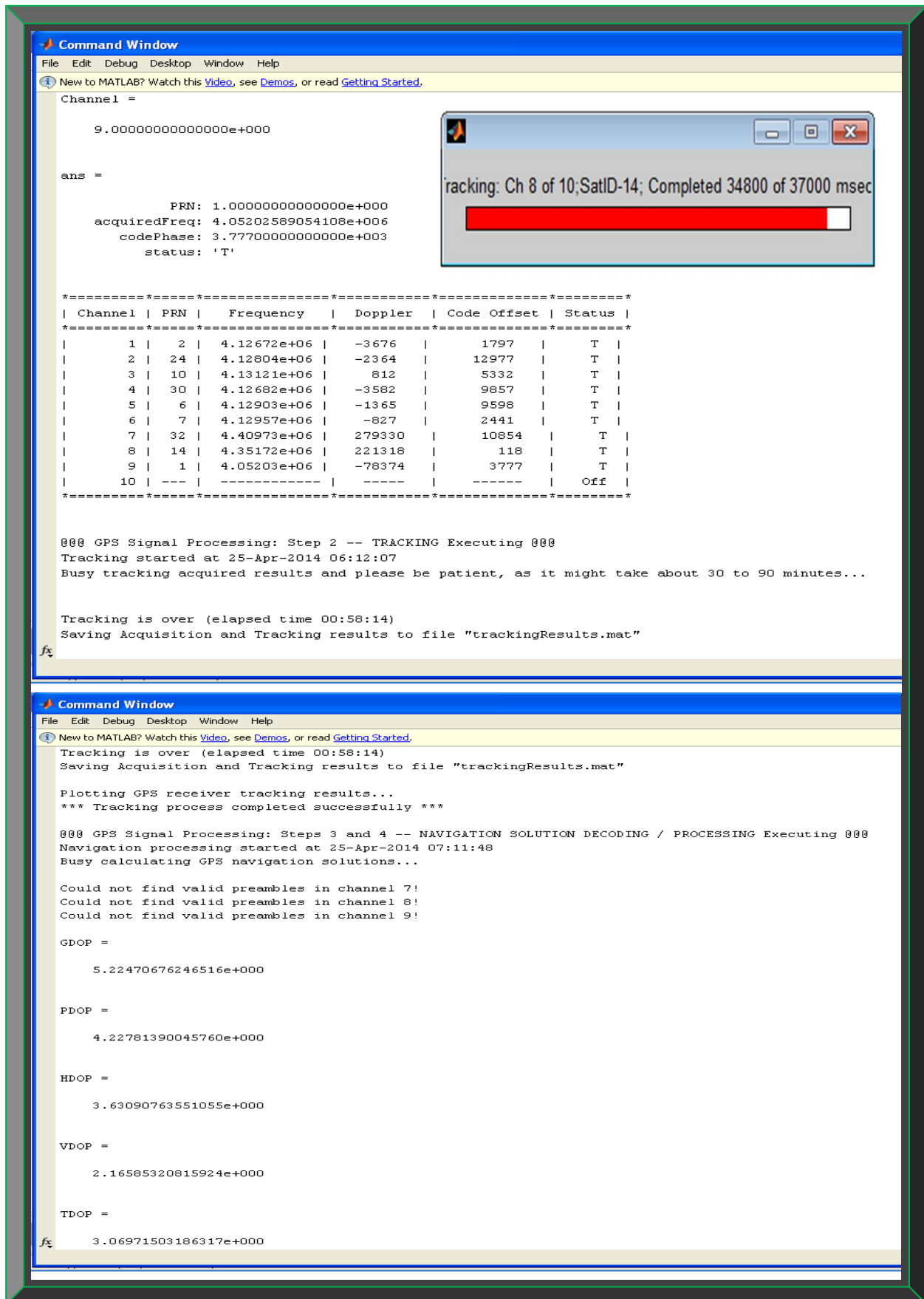


Figure 5.8: Summary of acquisition, tracking and navigation DOP results

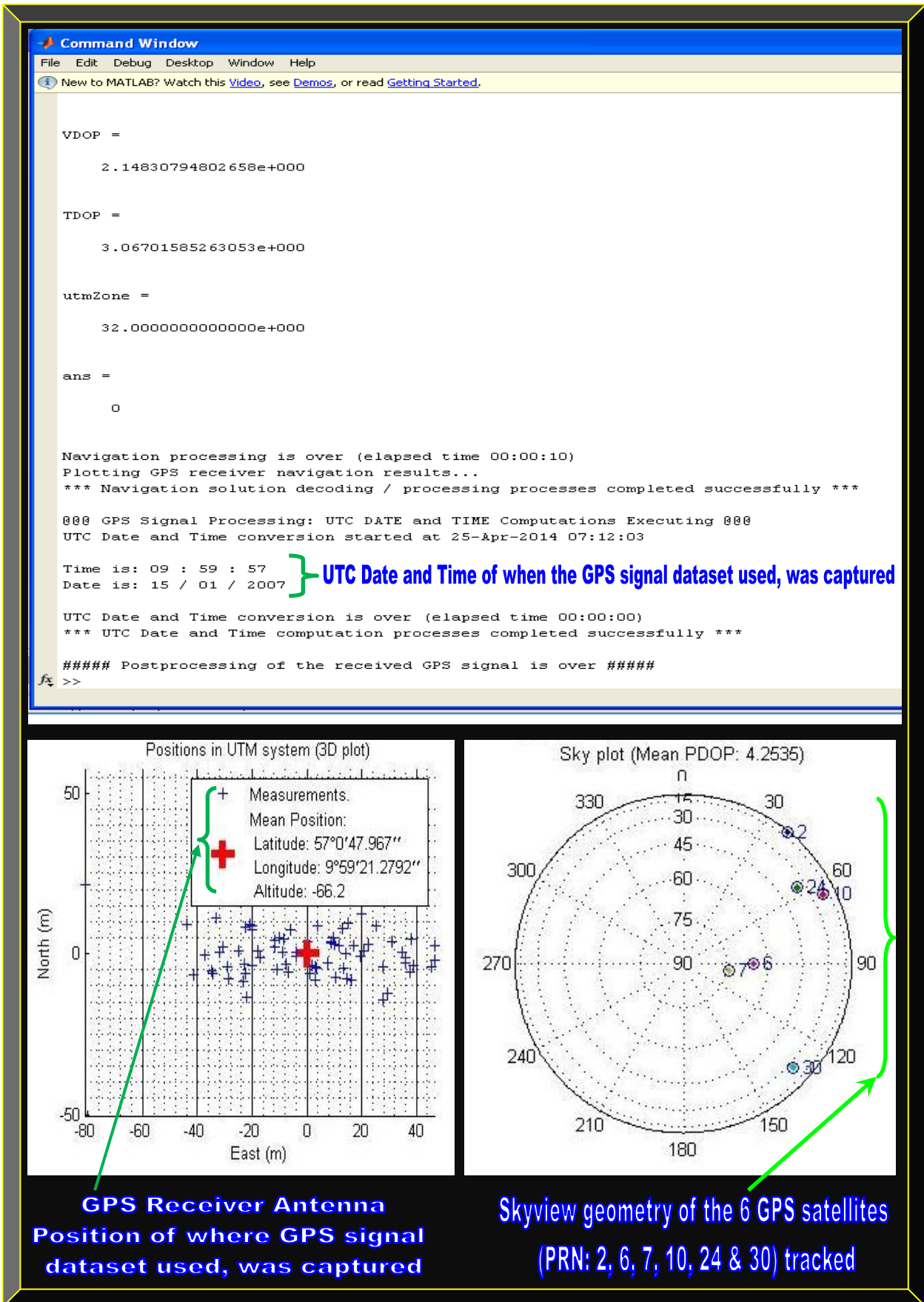


Figure 5.9: UTC date and time, UTM, GPS receiver position and orbiting satellites sky plot

5.2.2 Software-defined GPS Receiver Results Based-on 5 Matlab Fixed-point Algorithms

The results presented in this section, do not follow the flow shown in Section 5.2.1 because some of the functions in the acquisition, tracking and navigation processes; failed one or more of the floating to fixed-point conversion stages as summarised in Table 5.1. As a result; the acquisition, tracking and navigation processes could not be integrated as a continuous fixed-point Matlab software-defined GPS receiver unit. However, the approach employed was to discretely convert and test each function of the GPS software. Thus, five of the successful fixed-point algorithms results and a summary of all the results are presented. See Section 9.4 for details on the failures.

Table 5.1: Matlab floating-point GPS receiver software-to-firmware conversion results summary

GPS Receiver Floating-point Matlab Design Files	Matlab Floating-point GPS Receiver Software-to-Firmware Conversion Stages Results			
	Define Input Types	Fixed-point Conversion	VHDL Code Generation	VHDL Code Verification
1) acquisition.m	Passed	Fail	Fail	Fail
2) calcLoopCoef.m	Passed	Passed	Passed	Passed
3) calculatePseudoranges.m	Passed	Fail	Fail	Fail
4) cart2utm.m	Passed	Fail	Fail	Fail
5) check_t.m	Passed	Passed	Passed	Passed
6) checkPhase.m	Passed	Fail	Fail	Fail
7) clkSin.m	Passed	Fail	Fail	Fail
8) clsin.m	Passed	Passed	Passed	Fail
9) deg2dms.m	Passed	Passed	Passed	Passed
10) dms2mat.m	Passed	Fail	Fail	Fail
11) e_r_corr.m	Fail	Fail	Fail	Fail
12) ephemeris.m	Passed	Fail	Fail	Fail
13) findPreambles.m	Passed	Fail	Fail	Fail
14) findUtmZone.m	Passed	Passed	Passed	Passed
15) generateCAcode.m	Passed	Passed	Fail	Fail
16) gps2utc.m	Passed	Fail	Fail	Fail
17) gps_time.m	Passed	Fail	Fail	Fail
18) init_tracking.m	Passed	Fail	Fail	Fail
19) invert.m	Passed	Fail	Fail	Fail
20) leastSquarePos.m	Fail	Fail	Fail	Fail
21) makeCaTable.m	Passed	Fail	Fail	Fail
22) navPartyChk.m	Passed	Fail	Fail	Fail
23) new_cart2geo.m	Passed	Fail	Fail	Fail
24) postNavigation.m	Fail	Fail	Fail	Fail
25) satpos.m	Fail	Fail	Fail	Fail
26) showChannelStatus.m	Passed	Fail	Fail	Fail
27) togeod.m	Passed	Fail	Fail	Fail
28) topocent.m	Passed	Fail	Fail	Fail
29) tracking.m	Passed	Fail	Fail	Fail
30) tropo.m	Passed	Fail	Fail	Fail
31) twosComp2dec.m	Passed	Passed	Fail	Fail

5.2.2.1 calcLoopCoef.m Equivalent Matlab Fixed-point Algorithm Result

This is used as an example in Chapter 4 (Section 4.3.2.1.4) to describe the Matlab floating-point GPS receiver software-to-firmware workflow. For illustrative details, consult Figures 4.14 - 4.16.

5.2.2.2 check_t.m Equivalent Matlab Fixed-point Algorithm Result

Figures 5.10 and 5.11 are *check_t.m* floating to fixed-point algorithms results and error analysis.

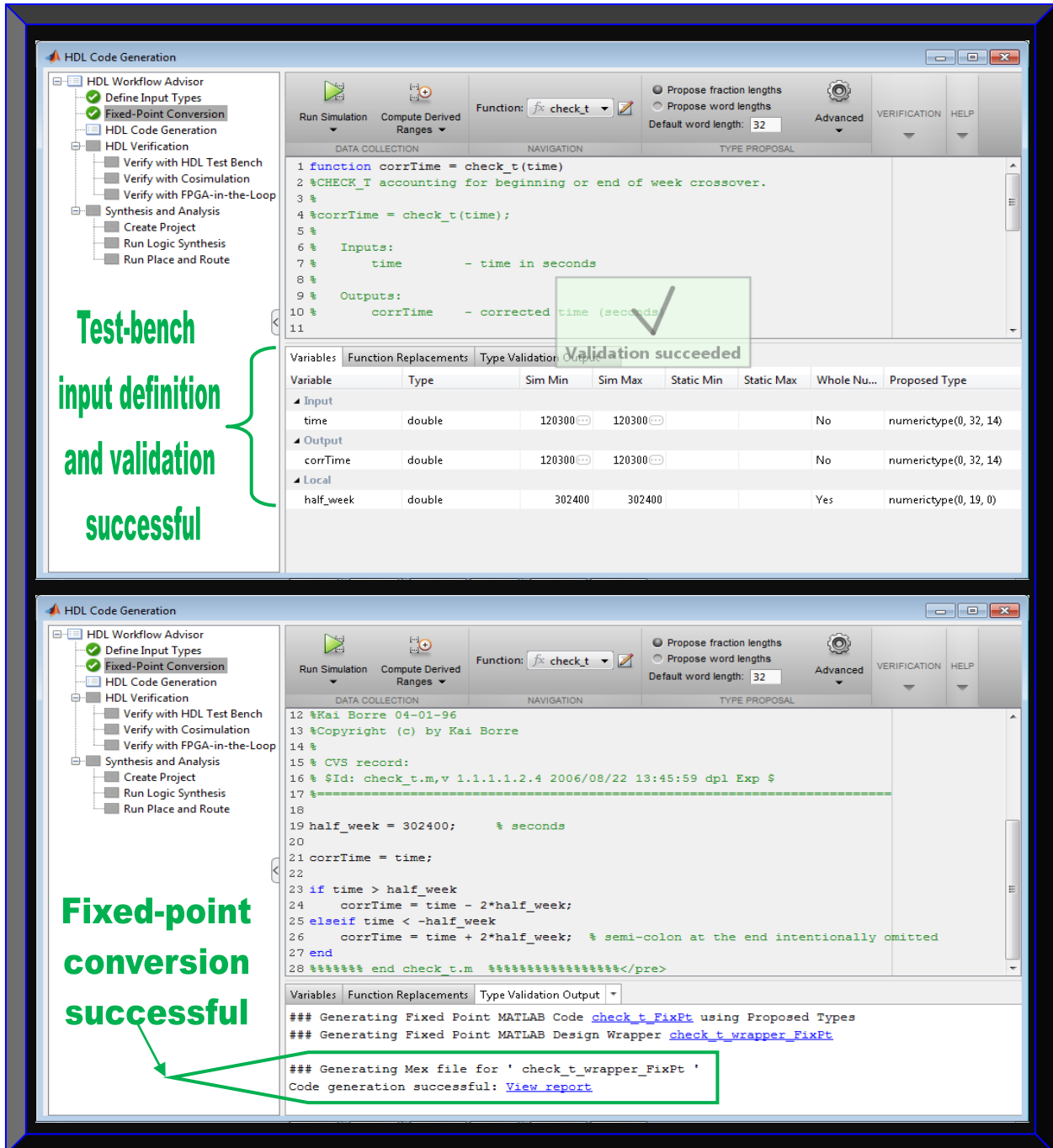


Figure 5.10: *check_t.m* variables definition and Matlab floating to fixed-point conversion results

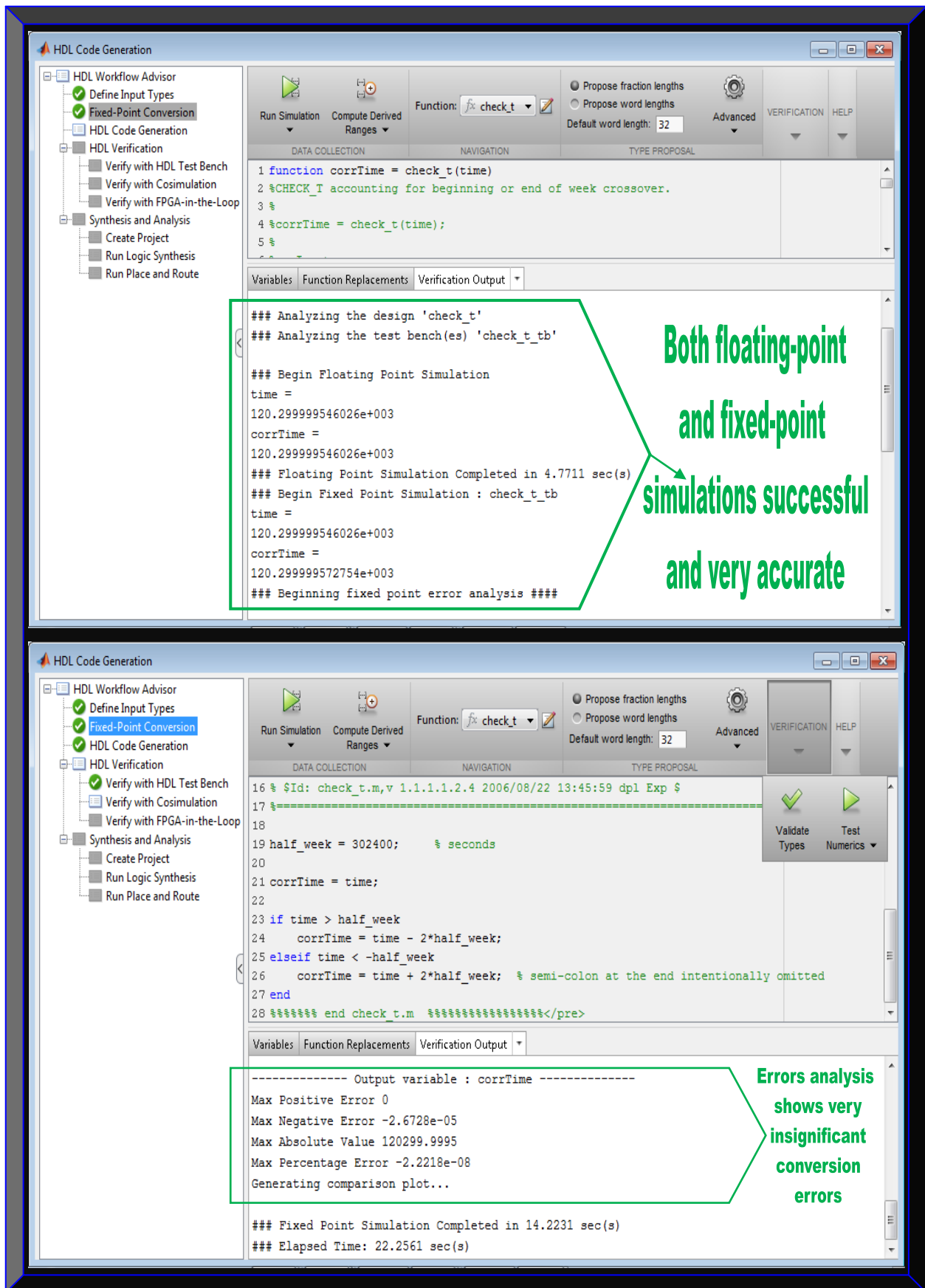


Figure 5.11: check_t.m Matlab fixed-point algorithm verification and error analysis results

5.2.2.3 *clsin.m* Equivalent Matlab Fixed-point Algorithm Result

Figures 5.12 and 5.13 are the *clsin.m* floating to fixed-point algorithms results and error analysis.

The screenshot displays the HDL Code Generation interface. On the left, the HDL Workflow Advisor shows 'Fixed-Point Conversion' as a completed step. The main window shows the MATLAB function code for *clsin.m* and a table of variable definitions.

Test-bench input definition and validation successful

Variable	Type	Sim Min	Sim Max	Static Min	Static Max	Whole Nu...	Proposed Type
Input							
ar	4x1 double	0	0			No	numerictype(1, 32, 39)
argument	double	1.99	1.99			No	numerictype(0, 32, 30)
degree	double	4	4			Yes	numerictype(0, 3, 0)
Output							
result	double	0	0			No	numerictype(1, 32, 39)
Local							
cos_arg	double	-0.82	-0.82			No	numerictype(1, 32, 31)
hr	double	0	0			No	numerictype(1, 32, 39)
hr1	double	0	0			No	numerictype(1, 32, 48)
hr2	double	0	0			No	numerictype(1, 32, 57)
t	double	1	4			Yes	numerictype(0, 3, 0)

The bottom screenshot shows the simulation results for the floating-point algorithm. A 'TEST FILES TO RUN' dialog box is open, listing *clsin_tb.m*. The simulation output shows the following values:

```

### Analyzing the design 'clsin'
### Analyzing the test bench(es) 'clsin_tb'

### Begin Floating Point Simulation
ar =
-3.37077907000000e-003
4.73444769000000e-006
-8.29914570000000e-009
15.8785330000000e-012
degree =
4.00000000000000e+000
argument =
1.99080000000000e+000
result =
-3.08133898371910e-003
### Floating Point Simulation Completed in 7.3454 sec(s)
    
```

Floating-point simulation successful

Figure 5.12: *clsin.m* variables definition and Matlab floating to fixed-point simulation results

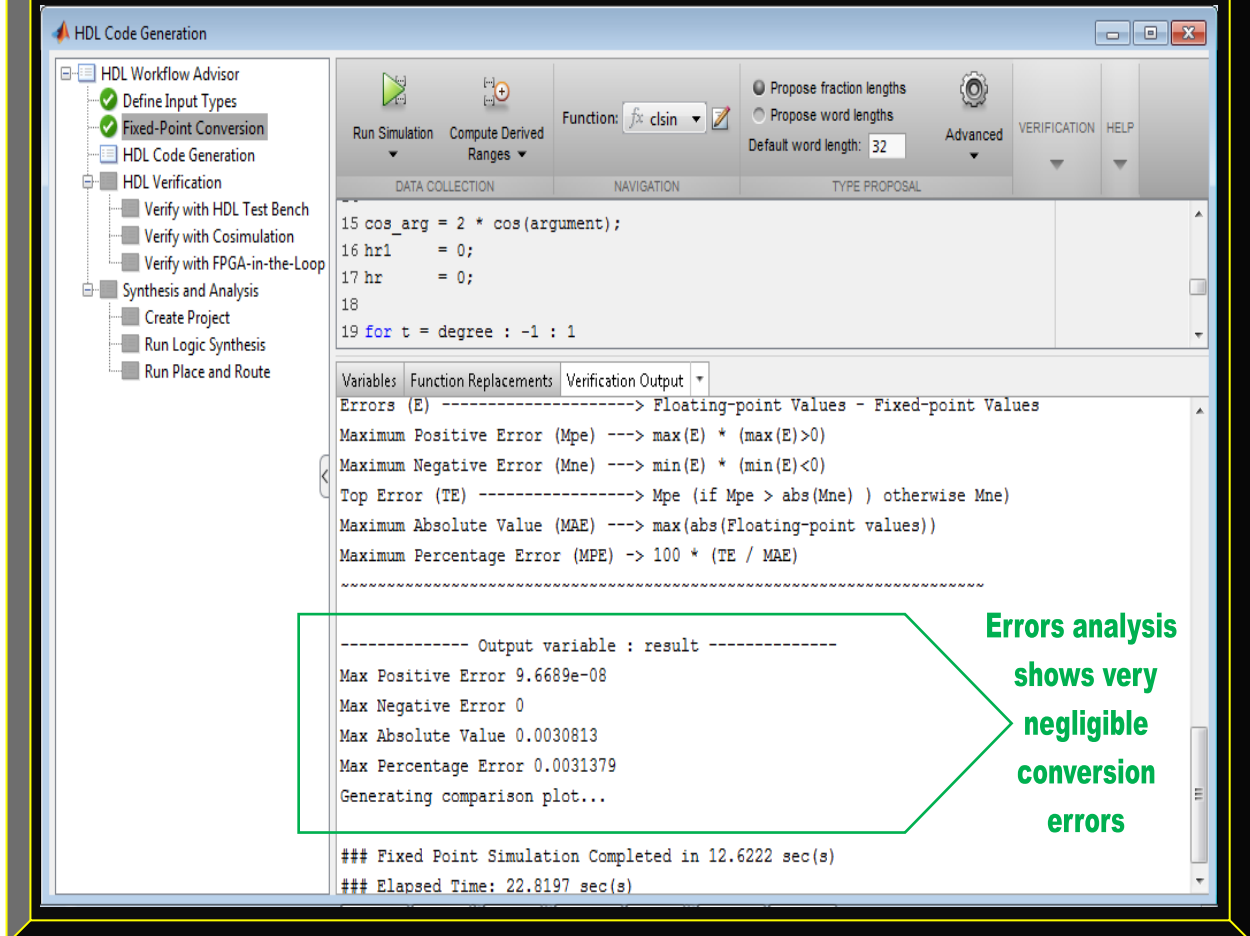
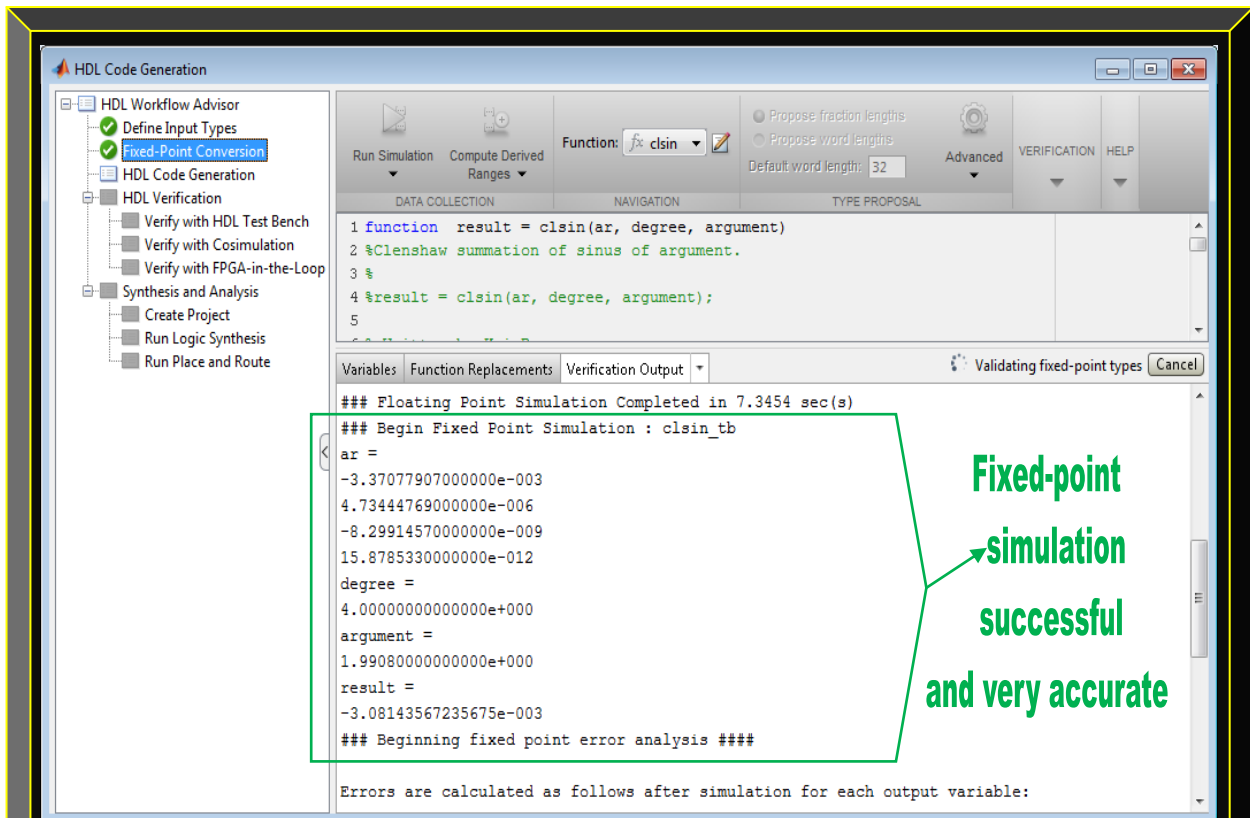


Figure 5.13: clsin.m Matlab fixed-point algorithm verification and error analysis results

5.2.2.4 deg2dms.m Equivalent Matlab Fixed-point Algorithm Result

Figures 5.14 and 5.15 are *deg2dms.m* floating to fixed-point algorithms result and error analysis.

The figure consists of two screenshots of the HDL Code Generation tool interface. The top screenshot shows the 'Type Validation Output' table, and the bottom screenshot shows the 'Type Validation Output' text log.

Test-bench input definition and validation successful

Variable	Type	Sim Min	Sim Max	Static Min	Static Max	Whole Nu...	Proposed Type
Input							
deg	double	57.03...	57.03...			No	numerictype(0, 32, 26)
Output							
dmsOutput	double	5701.45...	5701.45...			No	numerictype(0, 32, 19)
Local							
decimal	double	0.03...	0.03...			No	numerictype(0, 32, 36)
int_deg	double	57	57			Yes	numerictype(0, 6, 0)
min	double	1	1			Yes	numerictype(0, 1, 0)
min_part	double	1.76...	1.76...			No	numerictype(0, 32, 31)
neg_arg	logical	0	0			Yes	numerictype(0, 1, 0)
sec	double	45.46...	45.46...			No	numerictype(0, 32, 26)
sec_part	double	0.76...	0.76...			No	numerictype(0, 32, 32)

Fixed-point conversion successful with few warnings

```

48 function y = fi_uminus(a)
49 coder.inline( 'always' );
50 if isfi( a )
51     nt = numerictype( a );
52     fm = fimath( a );
53     new_nt = numerictype( 1, nt.WordLength + 1, nt.FractionLength );
54     y = -fi( a, new_nt, fm );
55 else
56     y = -a;
57 end
58 end

### Generating Fixed Point MATLAB Code deg2dms_FixPt using Proposed Types
### Generating Fixed Point MATLAB Design Wrapper deg2dms_wrapper_FixPt

### Generating Mex file for ' deg2dms_wrapper_FixPt '
Warning: Overflow or saturation occurred during constant folding. Value of this expression
cannot be exactly represented in its type.

Warning in ==> deg2dms_FixPt Line: 34 Column: 4
Code generation successful (with warnings): View report

```

Figure 5.14: *deg2dms.m* variables definition and Matlab floating to fixed-point conversion results

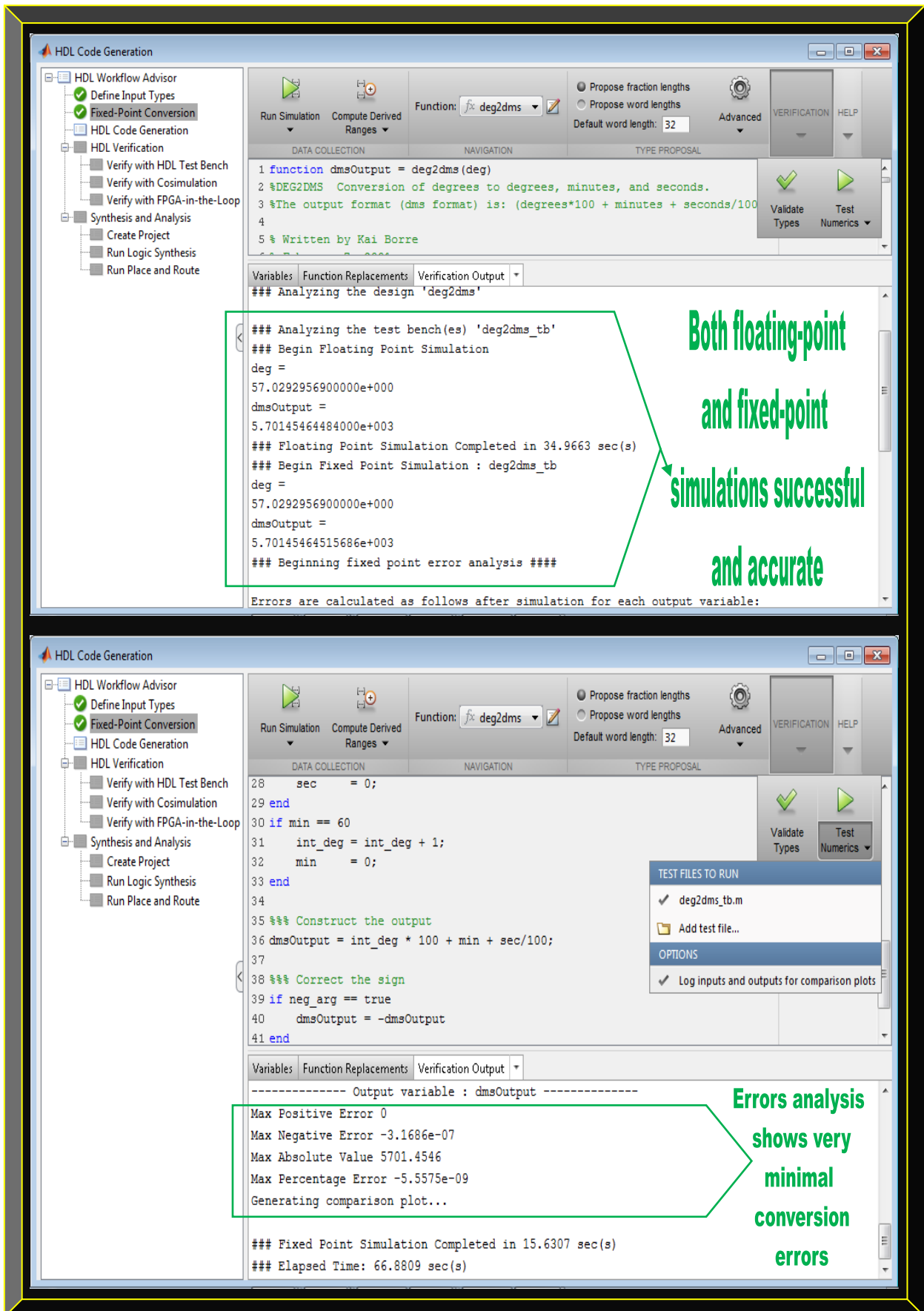


Figure 5.15: `deg2dms.m` Matlab fixed-point algorithm verification and error analysis results

5.2.2.5 findUtmZone.m Equivalent Matlab Fixed-point Algorithm Result

Figures 5.16 and 5.17 are *findUtmZone.m* floating to fixed-point codes results and error analysis.

Test-bench input definition and validation successful

Function: `findUtmZone_FixPt`

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %
3 %       Generated by MATLAB 8.1, MATLAB Coder 2.4 and HDL Coder 3.2
4 %
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 %#codegen
7 function utmZone = findUtmZone_FixPt(latitude,longitude)
8
9 fm = fimath('RoundingMethod','Nearest','OverflowAction','Wrap','ProductMode','FullPre
10 %Function finds the UTM zone number for given longitude and latitude.
11 %The longitude value must be between -180 (180 degree West) and 180 (180
12 %degree East) degree. The latitude must be within -80 (80 degree South) and
13 %84 (84 degree North).
14 %
15 %utmZone = findUtmZone(latitude, longitude);

```

Variable	Type	Sim Min	Sim Max	Static Min	Static Max	Whole Nu...	Proposed Type
Input							
latitude	double	57.03	57.03			No	numeric(0, 32, 26)
longitude	double	9.95	9.95			No	numeric(0, 32, 28)
Output							
utmZone	double	32	32			Yes	numeric(0, 6, 0)

Fixed-point conversion successful with few warnings

```

3 %       Generated by MATLAB 8.1, MATLAB Coder 2.4 and HDL Coder 3.2
4 %
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 %#codegen
7 function utmZone = findUtmZone_FixPt(latitude,longitude)
8
9 fm = fimath('RoundingMethod','Nearest','OverflowAction','Wrap','ProductMode','FullPre
10 %Function finds the UTM zone number for given longitude and latitude.
11 %The longitude value must be between -180 (180 degree West) and 180 (180
12 %degree East) degree. The latitude must be within -80 (80 degree South) and
13 %84 (84 degree North).

```

```

### Generating Fixed Point MATLAB Code findUtmZone_FixPt using Proposed Types
### Generating Fixed Point MATLAB Design Wrapper findUtmZone_wrapper_FixPt

### Generating Mex file for ' findUtmZone_wrapper_FixPt '
Warning: Overflow or saturation occurred during constant folding. Value of this expression
cannot be exactly represented in its type.

Warning in ==> findUtmZone_FixPt Line: 43 Column: 6
Code generation successful (with warnings): View report

```

Figure 5.16: *findUtmZone.m* variables definition and Matlab floating to fixed-point conversion results

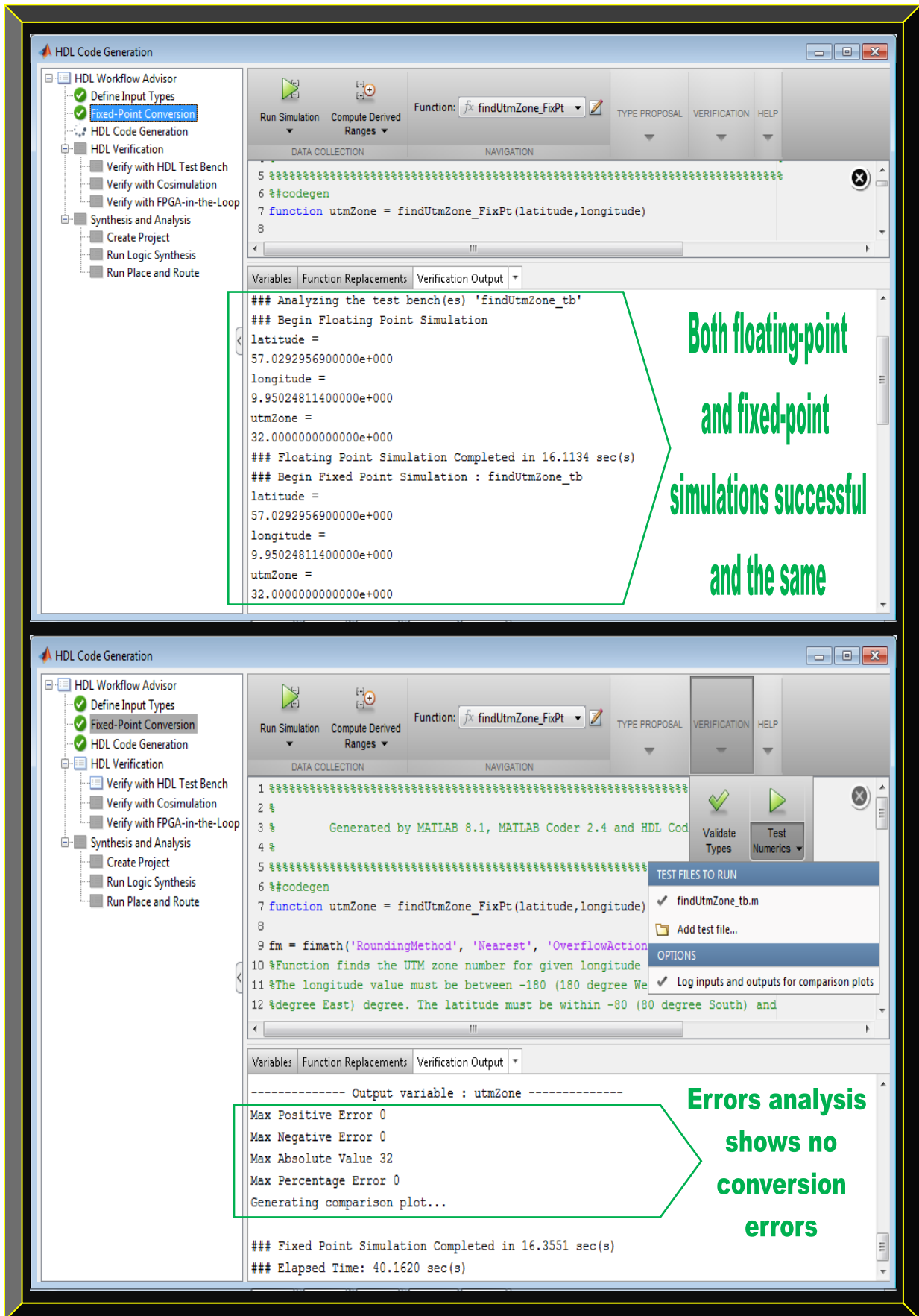


Figure 5.17: `findUtmZone.m` Matlab fixed-point algorithm verification and error analysis results

5.2.3 Software-defined GPS Receiver Results Based-on 5 Discrete Simulink Models

As mentioned in Section 5.2.2, only the corresponding GPS receiver Simulink models results, generated from their respective fixed-point algorithms are presented in this section. Furthermore, Simulink models based-on the floating-point Matlab GPS algorithms were manually created and the results are compared alongside with their respective generated fixed-point Simulink models.

5.2.3.1 calcLoopCoef.m Equivalent Simulink Models Results Comparison

This is used as an example in Chapter 4 (Section 4.3.2.1.5) to explain the Matlab floating-point GPS receiver software-to-firmware workflow. Refer to Figure 4.19 for applicable details.

5.2.3.2 check_t.m Equivalent Simulink Models Results Comparison

Figure 5.18 shows comparison results of *check_t.m* as floating and fixed-point Simulink models.

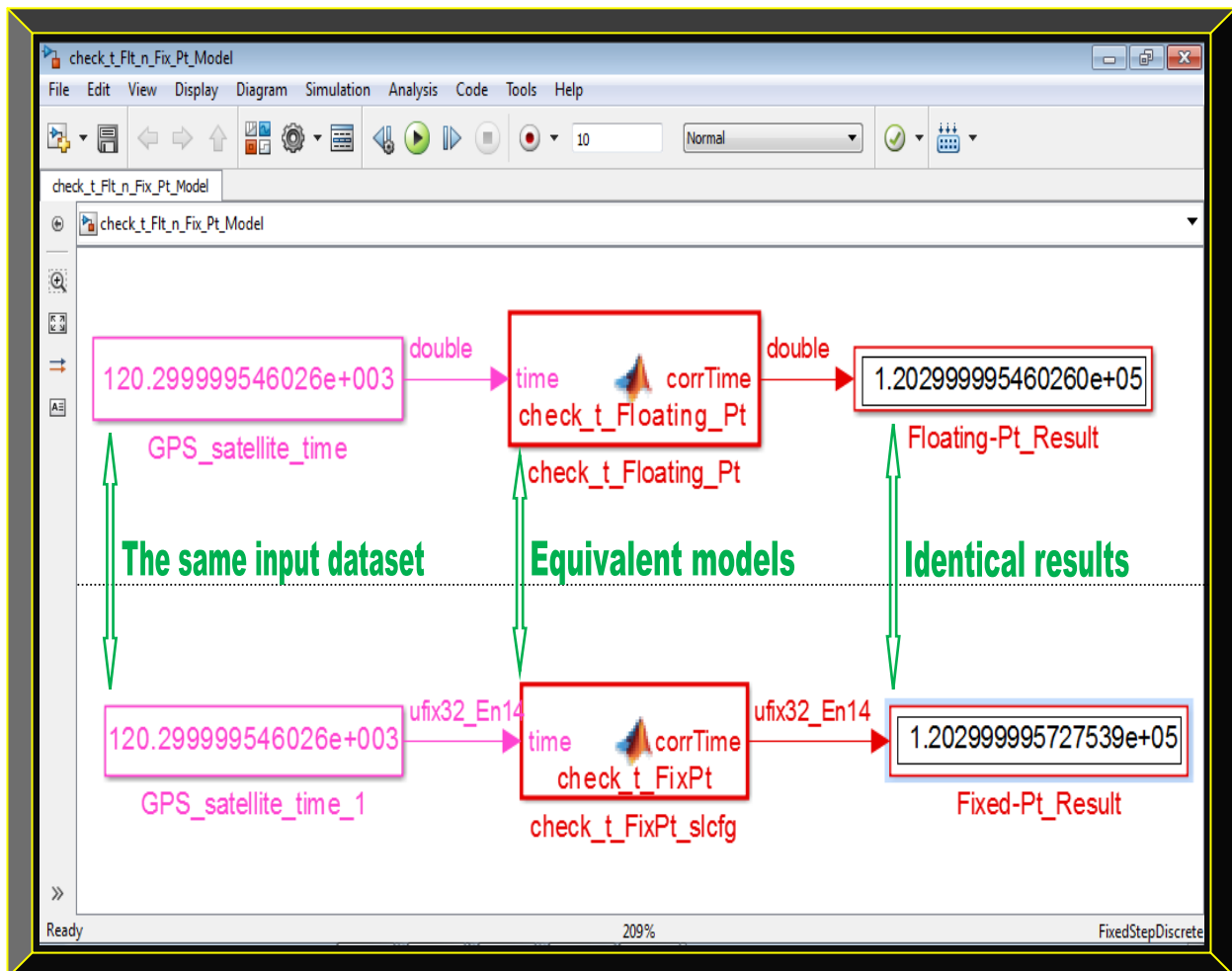


Figure 5.18: *check_t.m* Simulink floating and fixed-point models results comparison

5.2.3.3 clsin.m Equivalent Simulink Models Results Comparison

Figure 5.19 shows comparison results of *clsin.m* as Simulink floating and fixed-point models

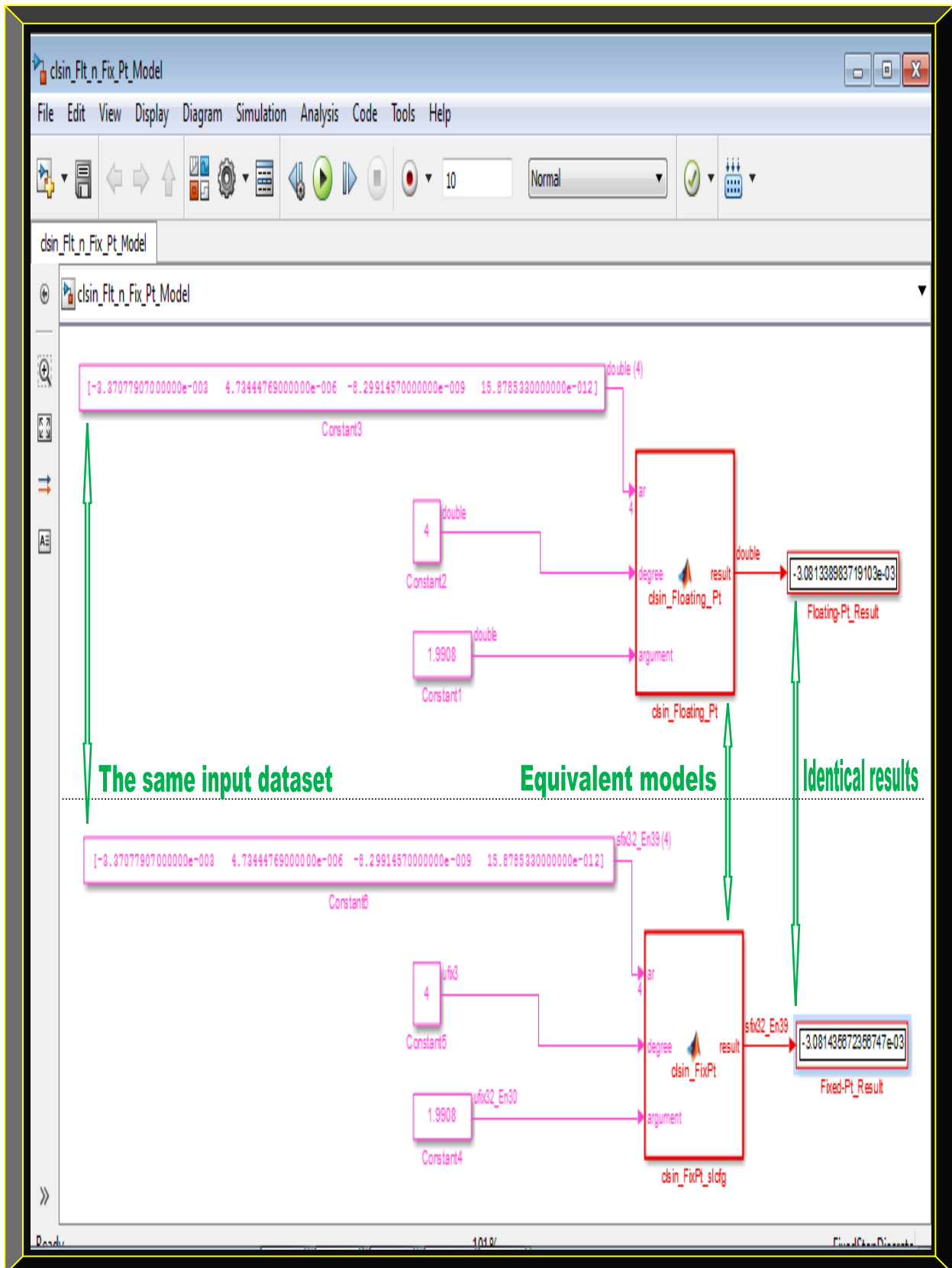


Figure 5.19: *clsin.m* Simulink floating and fixed-point models results comparison

5.2.3.4 deg2dms.m Equivalent Simulink Models Results Comparison

Figure 5.20 shows *deg2dms.m* Simulink floating and fixed-point models comparison results.

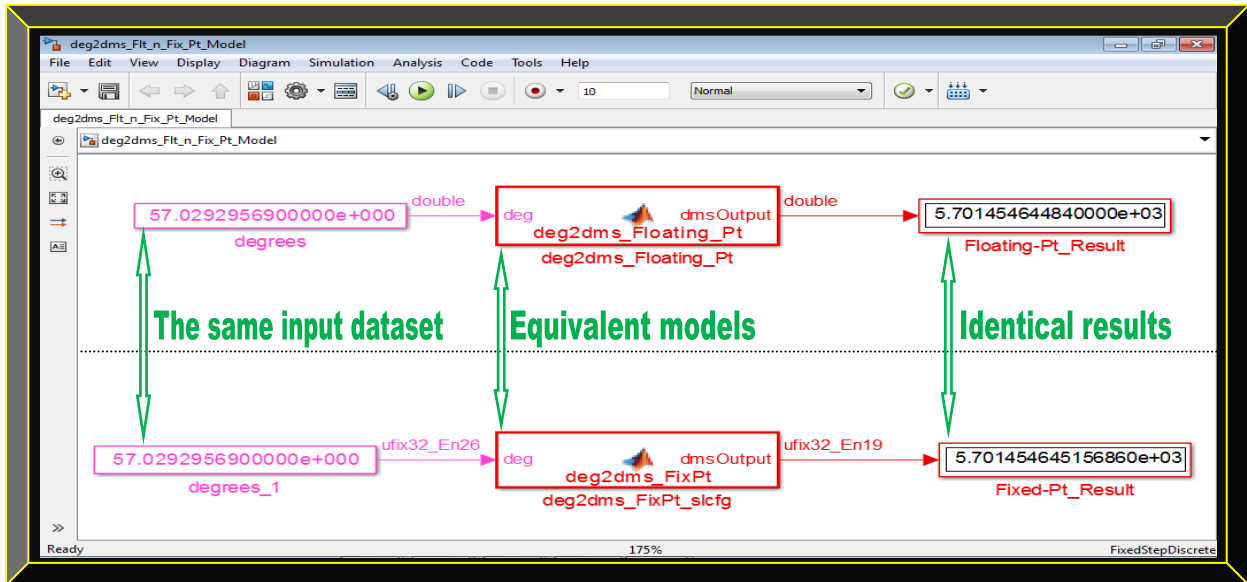


Figure 5.20: *deg2dms.m* Simulink floating and fixed-point models result comparison

5.2.3.5 findUtmZone.m Equivalent Simulink Models Results Comparison

Figure 5.21 shows *findUtmZone.m* Simulink floating and fixed-point models comparison result.

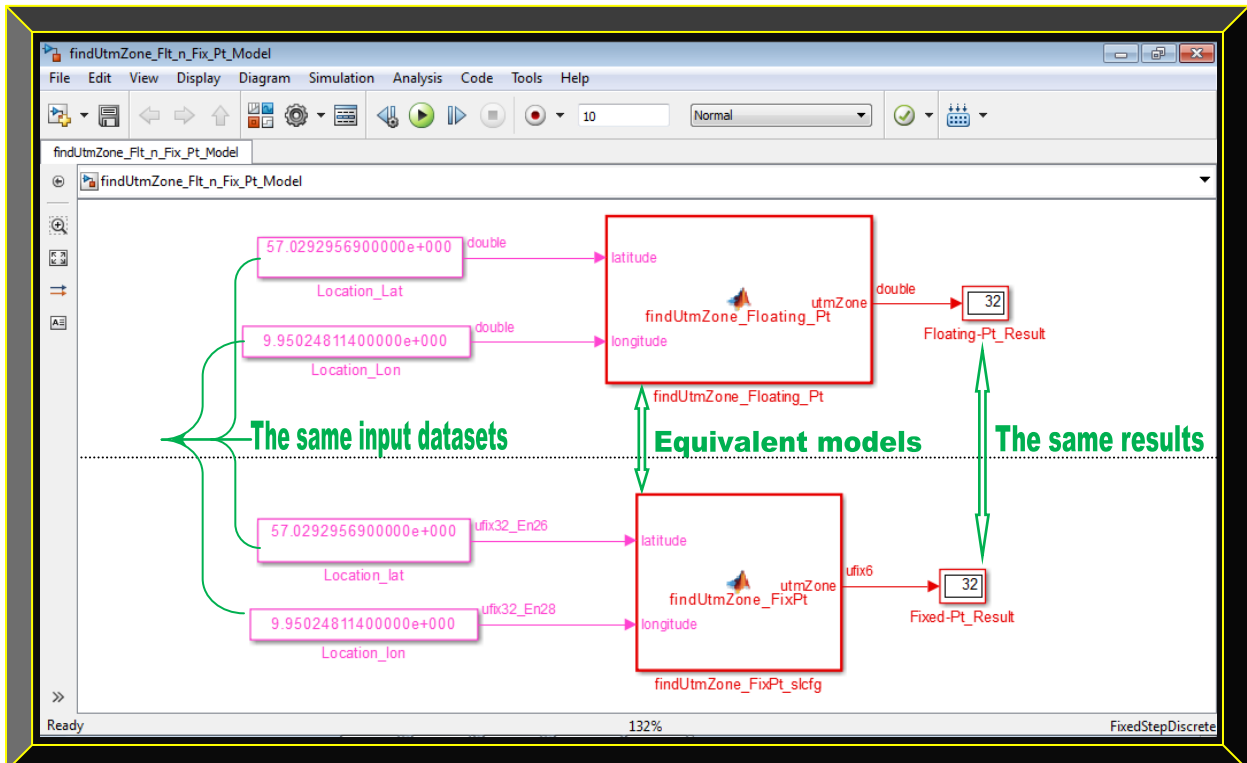


Figure 5.21: *findUtmZone.m* Simulink floating and fixed-point models results comparison

5.2.4 Firmware-defined GPS Receiver Results Based-on 5 Discrete VHDL Algorithms

For reasons already indicated in Section 5.2.2, only the corresponding GPS receiver VHDL algorithms results, generated from their respective fixed-point algorithms are presented here.

5.2.4.1 calcLoopCoef.m Equivalent VHDL Algorithm Result

This is used as an example in Chapter 4 (Section 4.3.2.1.5) to describe the Matlab floating-point GPS receiver software-to-firmware workflow. For descriptive details, see Figures 4.20 - 4.22.

5.2.4.2 check_t.m Equivalent VHDL Algorithm Result

Figures 5.22 and 5.23 show the equivalent generated VHDL algorithm and the verification result.

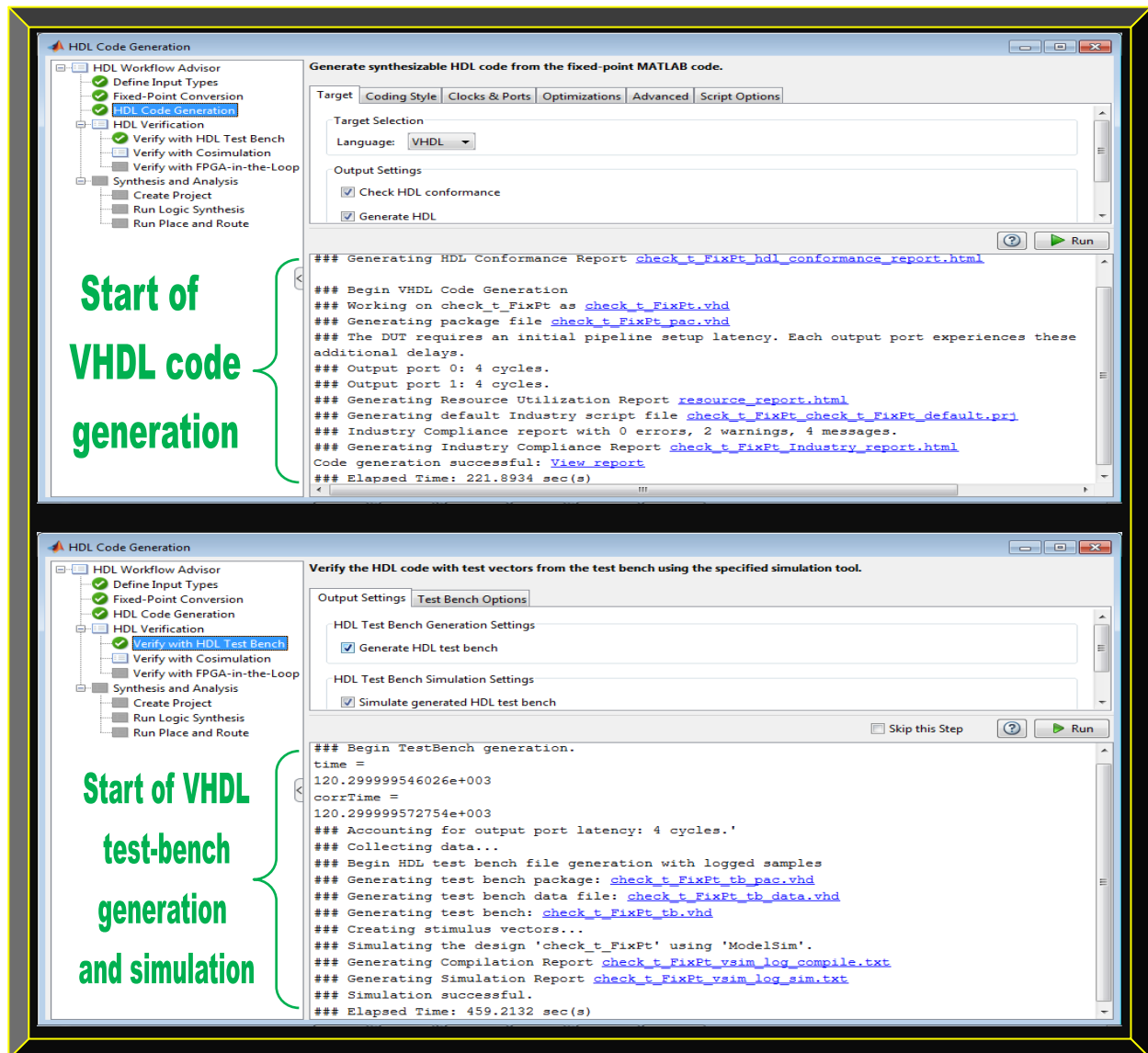


Figure 5.22: check_t.m conversion to VHDL and the algorithm verification processes

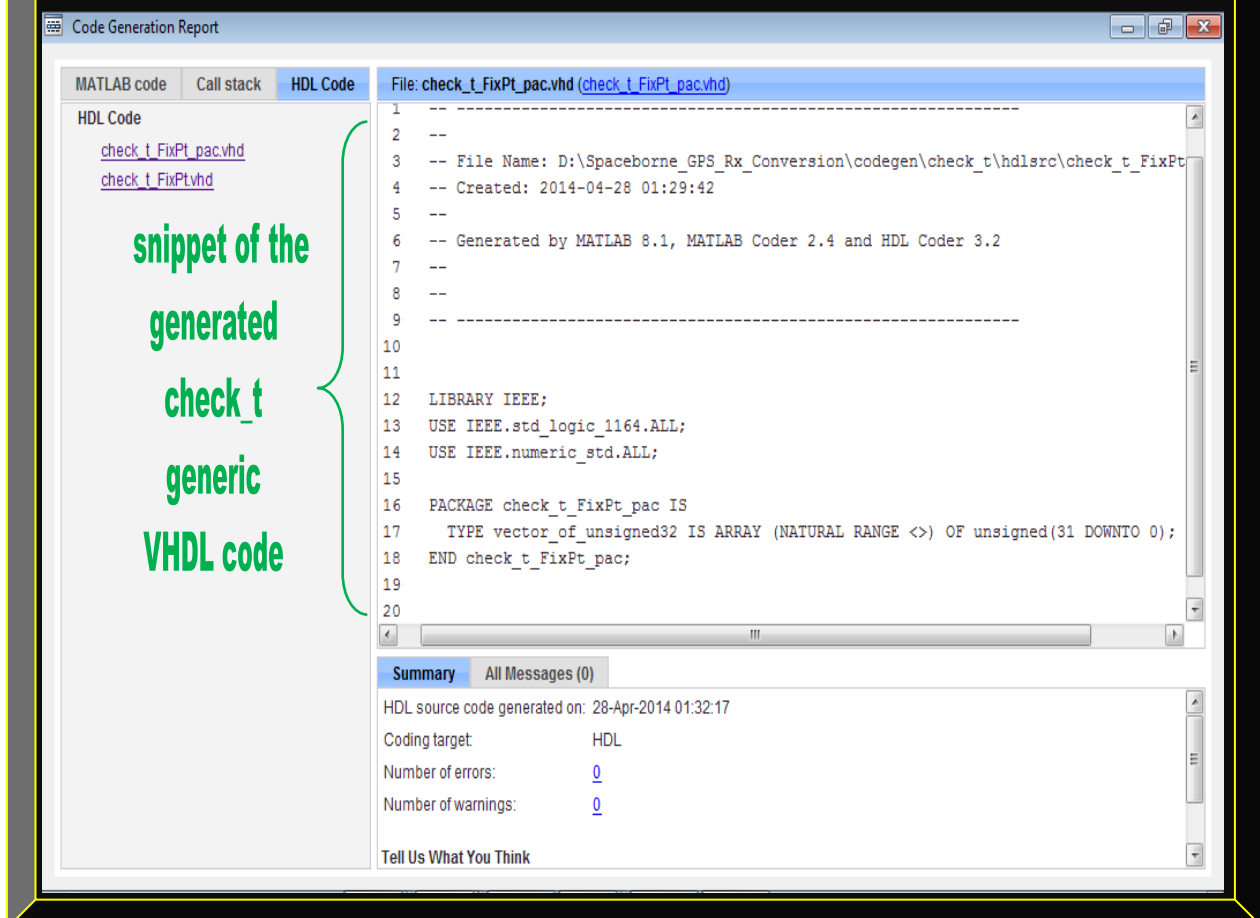
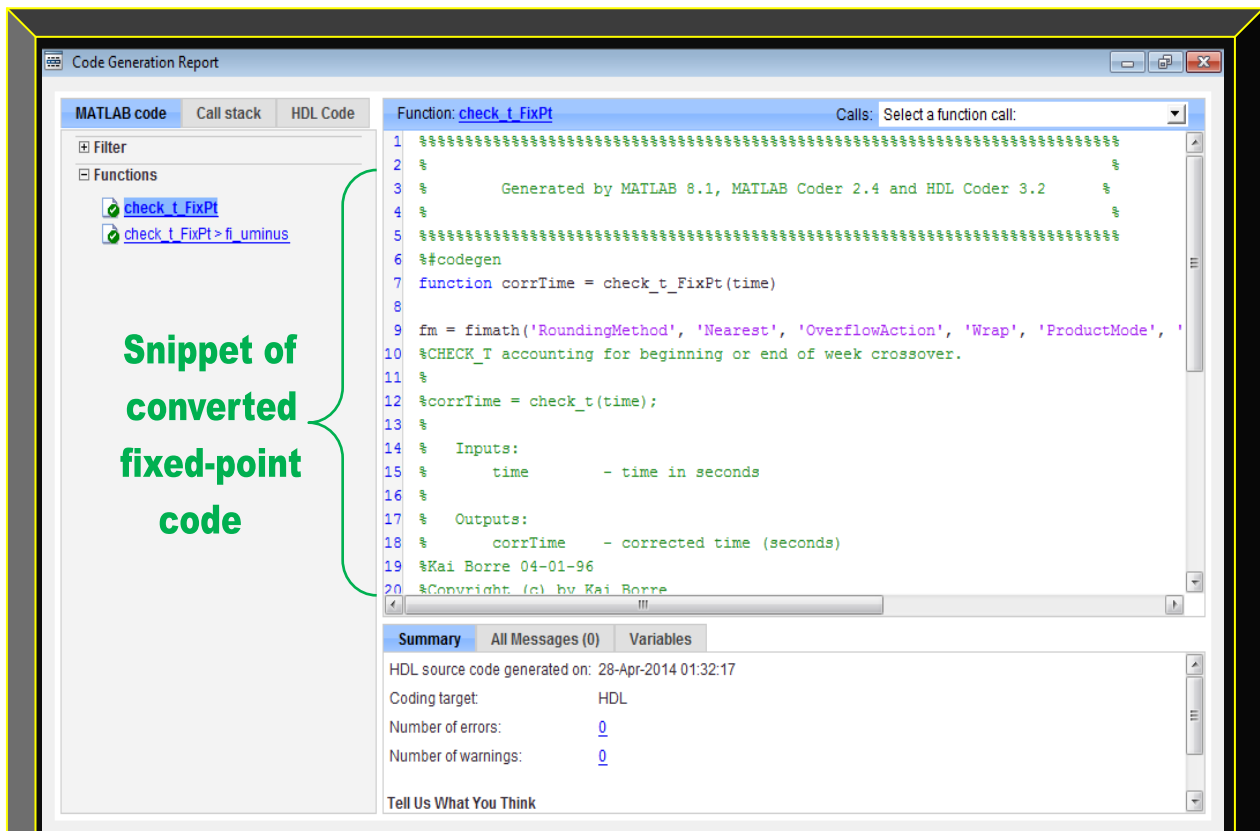


Figure 5.23: Preview of check_t.m algorithm – converted to VHDL

5.2.4.3 deg2dms.m Equivalent VHDL Algorithm Result

Figures 5.24 and 5.25 show the equivalent generated VHDL algorithm and the verification result.



Figure 5.24: deg2dms.m conversion to VHDL and the algorithm verification processes

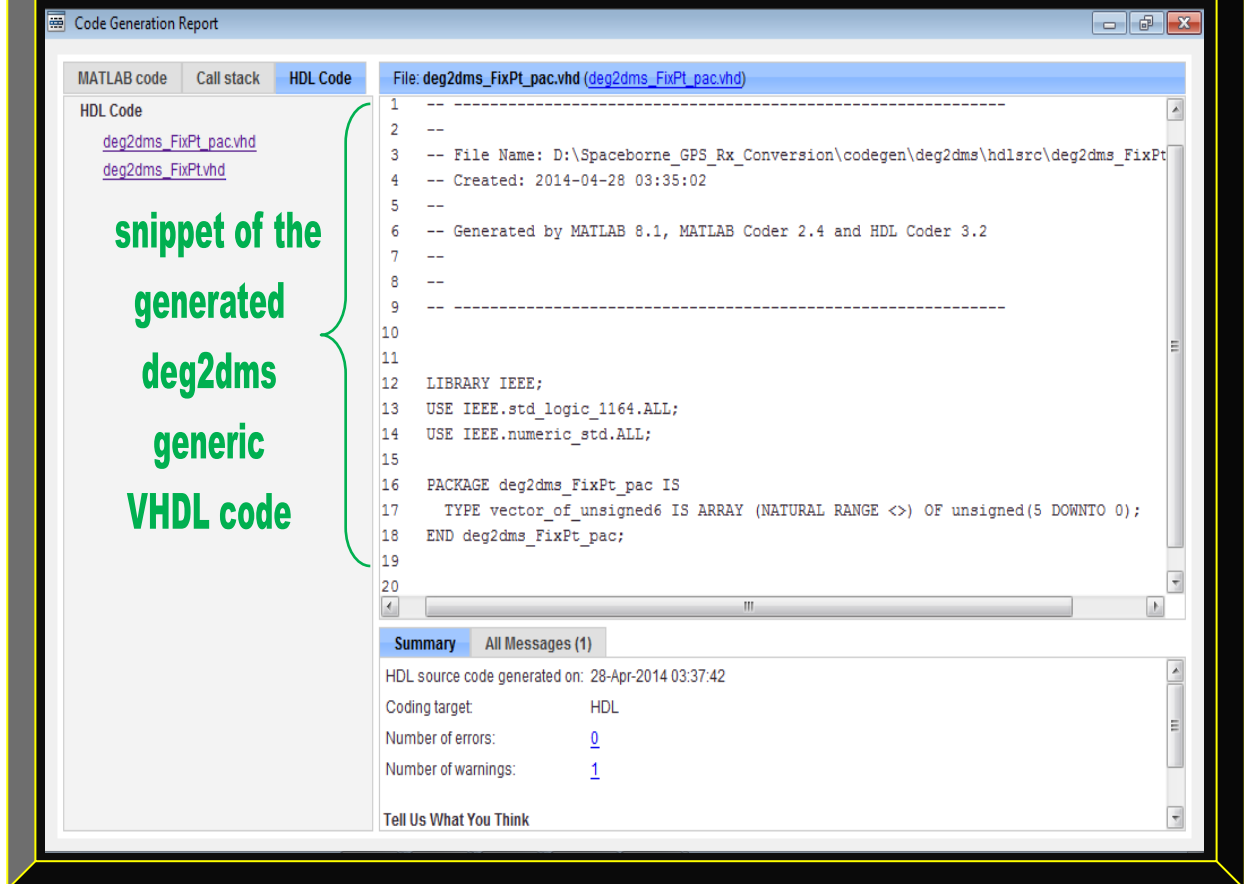
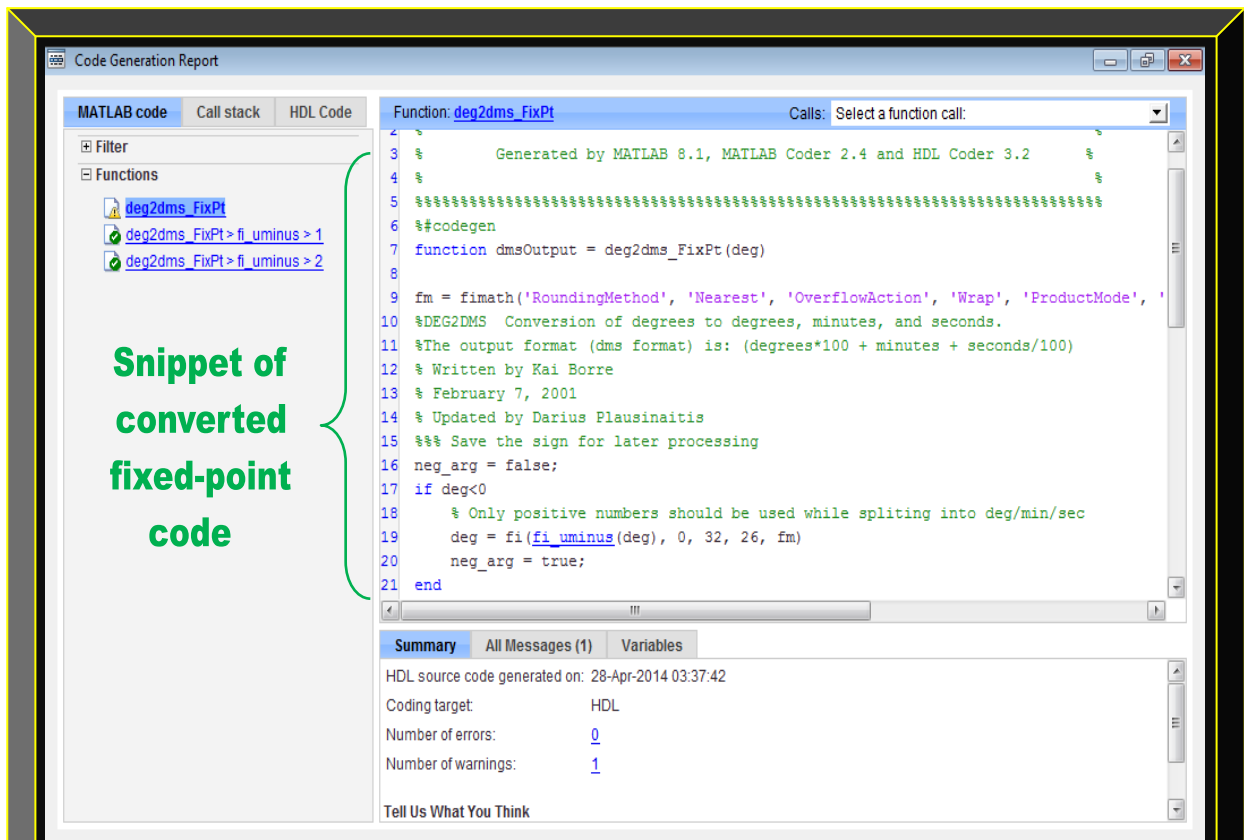


Figure 5.25: Preview of deg2dms.m algorithm – converted to VHDL

5.2.4.4 findUtmZone.m Equivalent VHDL Algorithm Result

Figures 5.26 and 5.27 show the equivalent generated VHDL algorithm and the verification result.

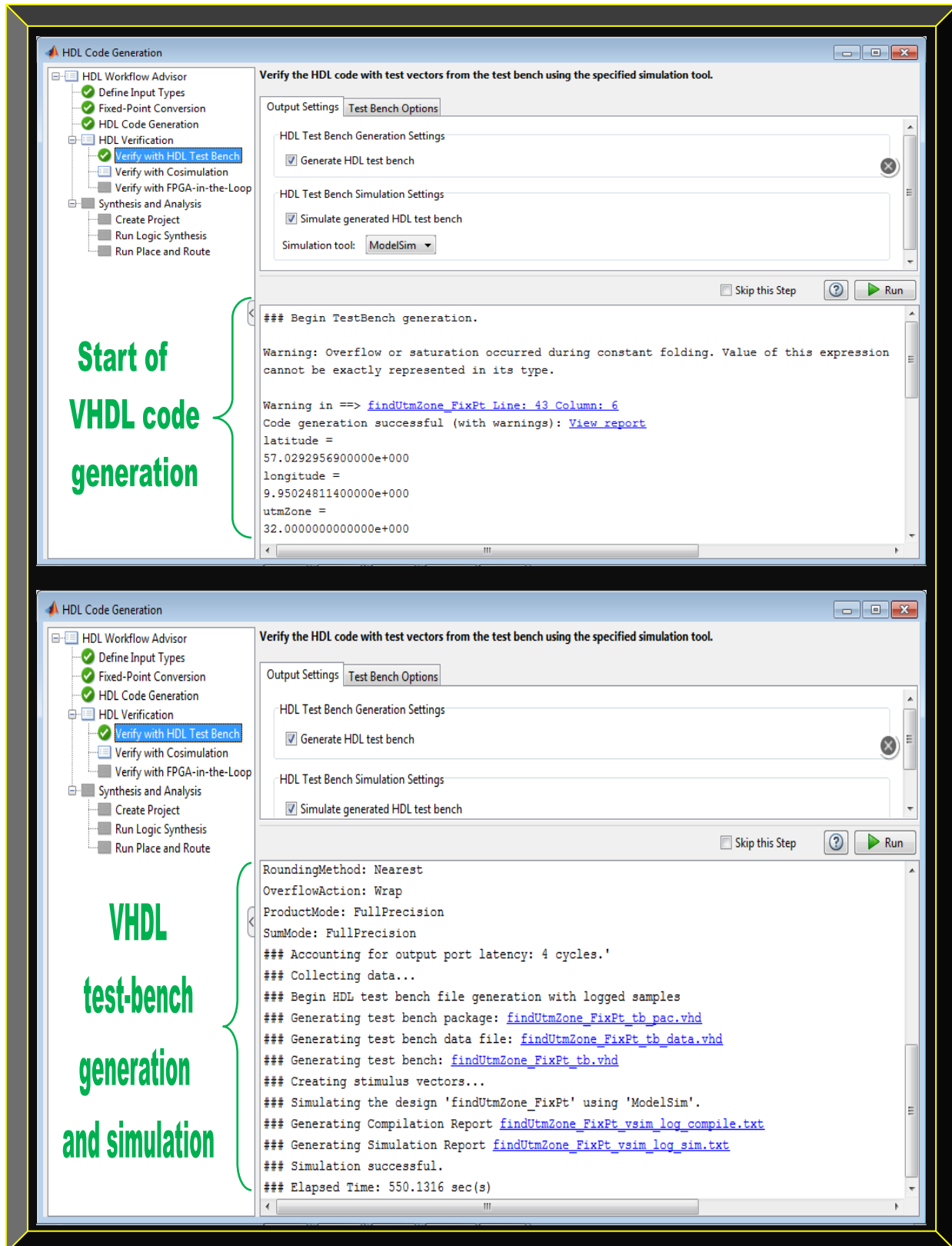


Figure 5.26: findUtmZone.m conversion to VHDL and the algorithm verification processes

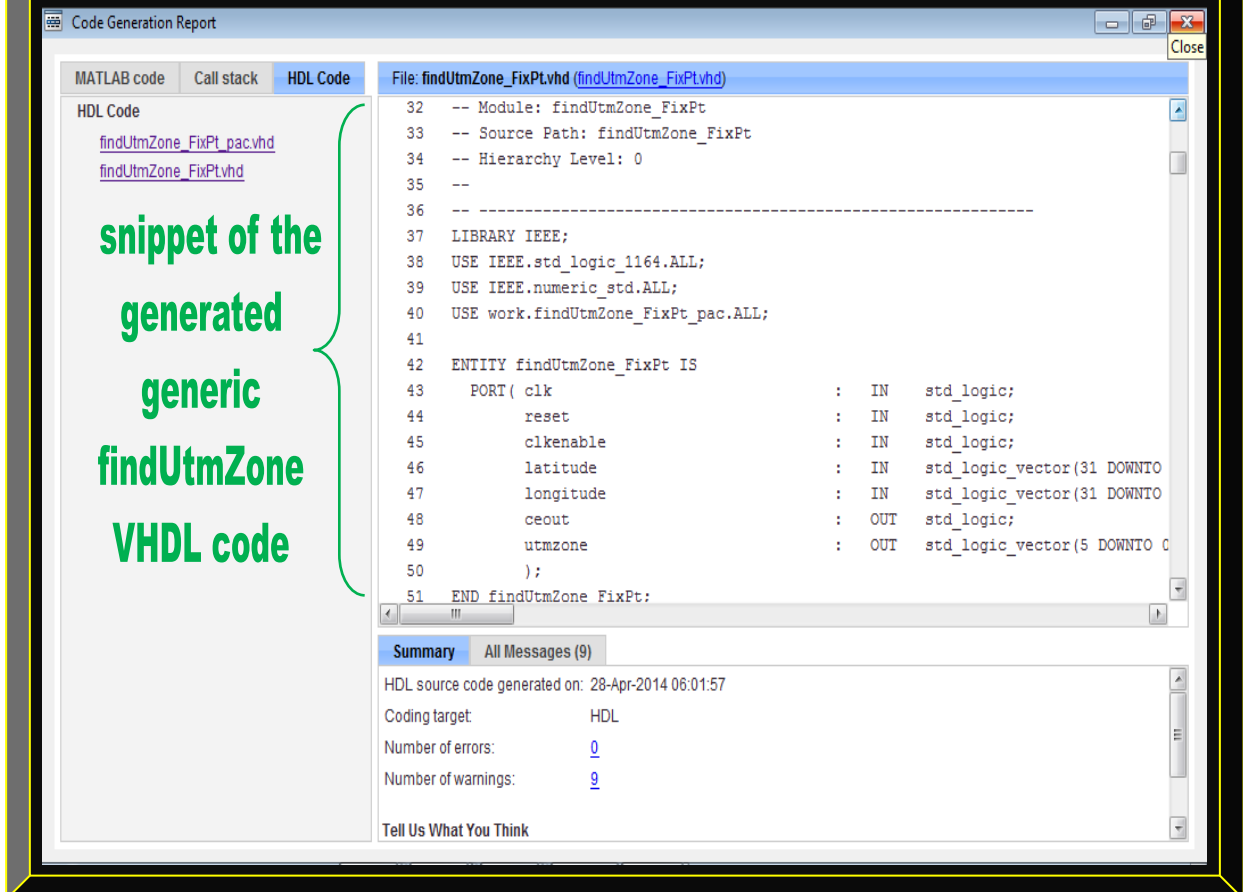
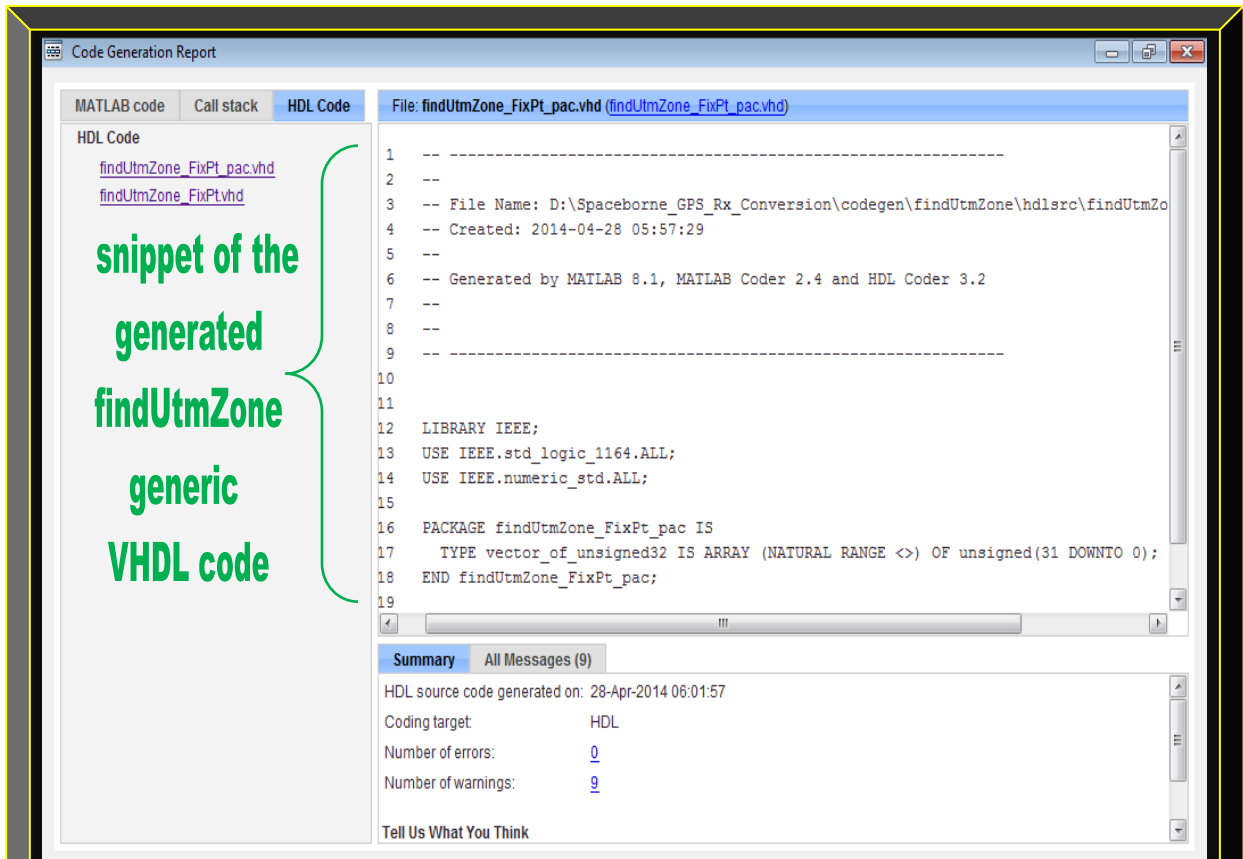


Figure 5.27: Preview of findUtmZone.m algorithm – converted to VHDL

5.2.4.5 clsin.m Equivalent VHDL Algorithm Result

Figure 5.28 depicts *clsin.m* conversion to VHDL but the verification process was unsuccessful.

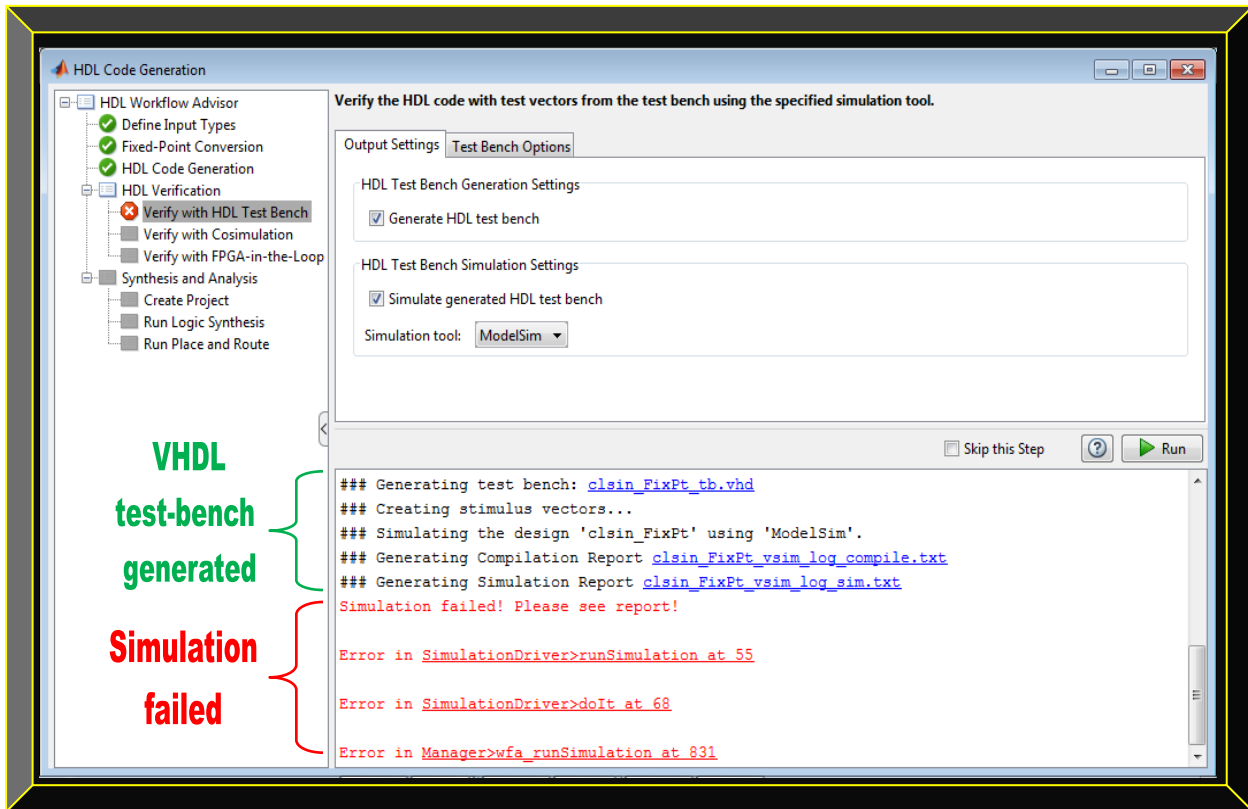


Figure 5.28: *clsin.m* generation process to VHDL and verification failure

5.2.4.6 Summary of the GPS Receiver Algorithms Results Comparison and Error Analysis

Table 5.2 summarises the output results of each of the equivalent four types of GPS receiver algorithms (Matlab floating-point – now as reference, Matlab fixed-point, Simulink fixed-point and VHDL). The same test-bench was applied to all the four equivalent algorithms. The errors in each GPS algorithm is then analysed to justify the accuracy of the entire conversion process.

Table 5.2: Summary of the GPS receiver algorithms output results comparison and error analysis

Successful GPS Receiver Algorithms Design Files	Four Equivalent GPS Receiver Algorithms (from each design file) Output Results and Error Analysis				
	Matlab Floating-Point	Matlab Fixed-point	Simulink Fixed-point	VHDL	Max % Error
2) <i>calcLoopCoef</i>	tau2 = 0.3700000000	3.70247933897190e-1	3.70247933897189e-1	3.70247933897190e-1	-0.067009
5) <i>check_t</i>	1.202999995460260e5	1.202999995727540e5	1.202999995727539e5	1.202999995727540e5	-2.2218e-8
8) <i>clsin</i>	-3.081338983719103e-3	-3.081435672356750e-3	-3.081435672356747e-3	Fail	0.0031379
9) <i>deg2dms</i>	5.70145464484000e3	5.70145464515686e3	5.70145464515686e3	5.70145464515686e3	-5.5575e-9
14) <i>findUtmZone</i>	32	32	32	32	0

5.2.4.7 Firmware Compilation Results of check_t, deg2dms and findUtmZone VHDL Algorithms

This section presents three of the generated VHDL algorithms (*check_t*, *deg2dms* and *findUtmZone*) compiled results using Altera Quartus II software. The version of Matlab used (Matlab version R2013a) had problems detecting Altera or Xilinx synthesis tools; therefore, the synthesis and analysis step in the Matlab HDL Coder workflow could not be done. However, to complete the research methodology workflow, the workaround employed was to directly execute this step using preferably Altera Quartus II Design Suite (ease of use and easily downloadable with less license hassles and limitations). The three generated VHDL algorithms were analysed & synthesised, placed & routed and programming files generated, as well as timing analysis. This enables these files to be now embedded or programmed as firmware into a suitable FPGA. Figures 5.29 - 5.31 depict the compilation results and for that reason, conclude the workflow.

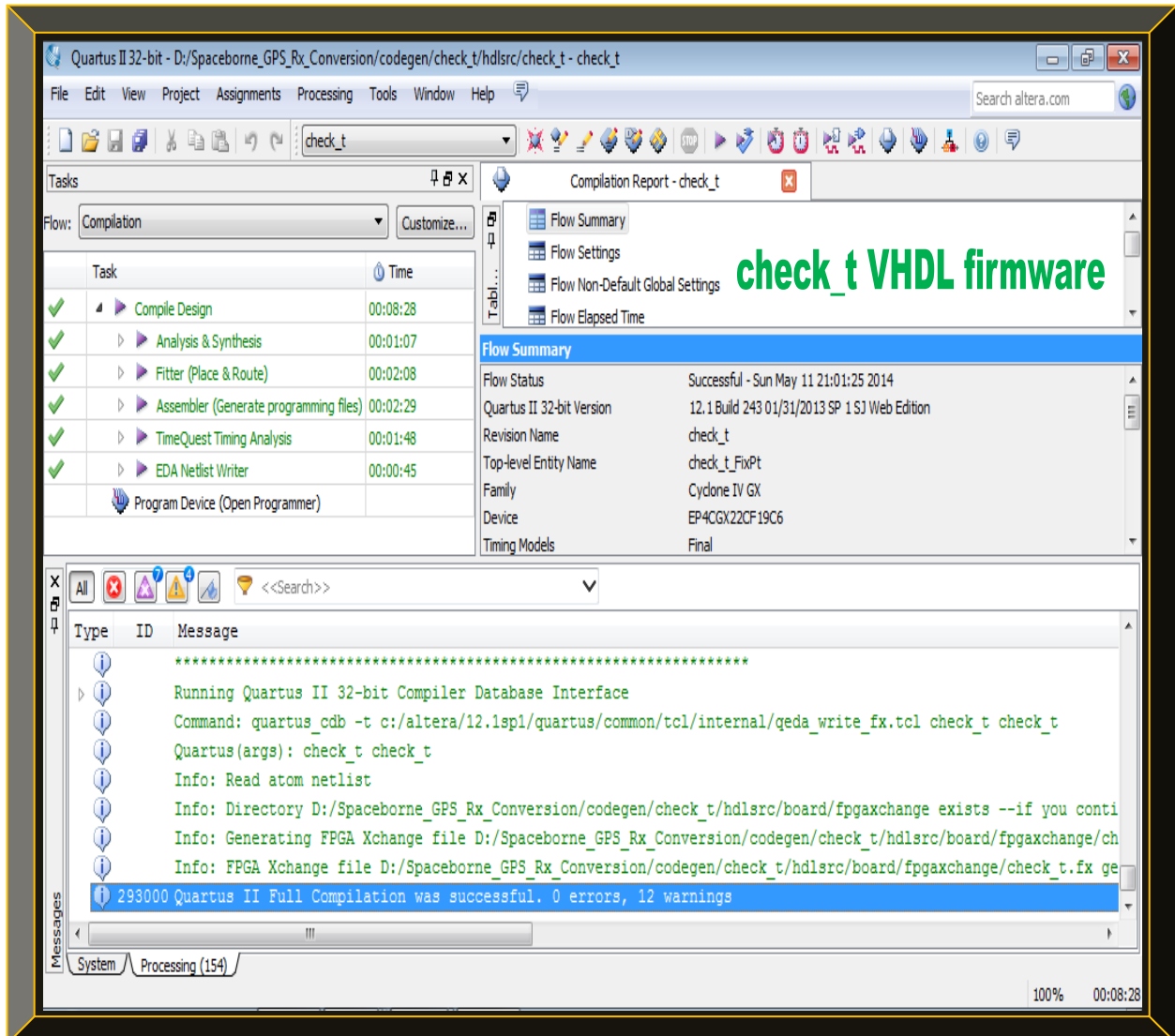


Figure 5.29: check_t firmware compilation result – ready now to be programmed into a FPGA

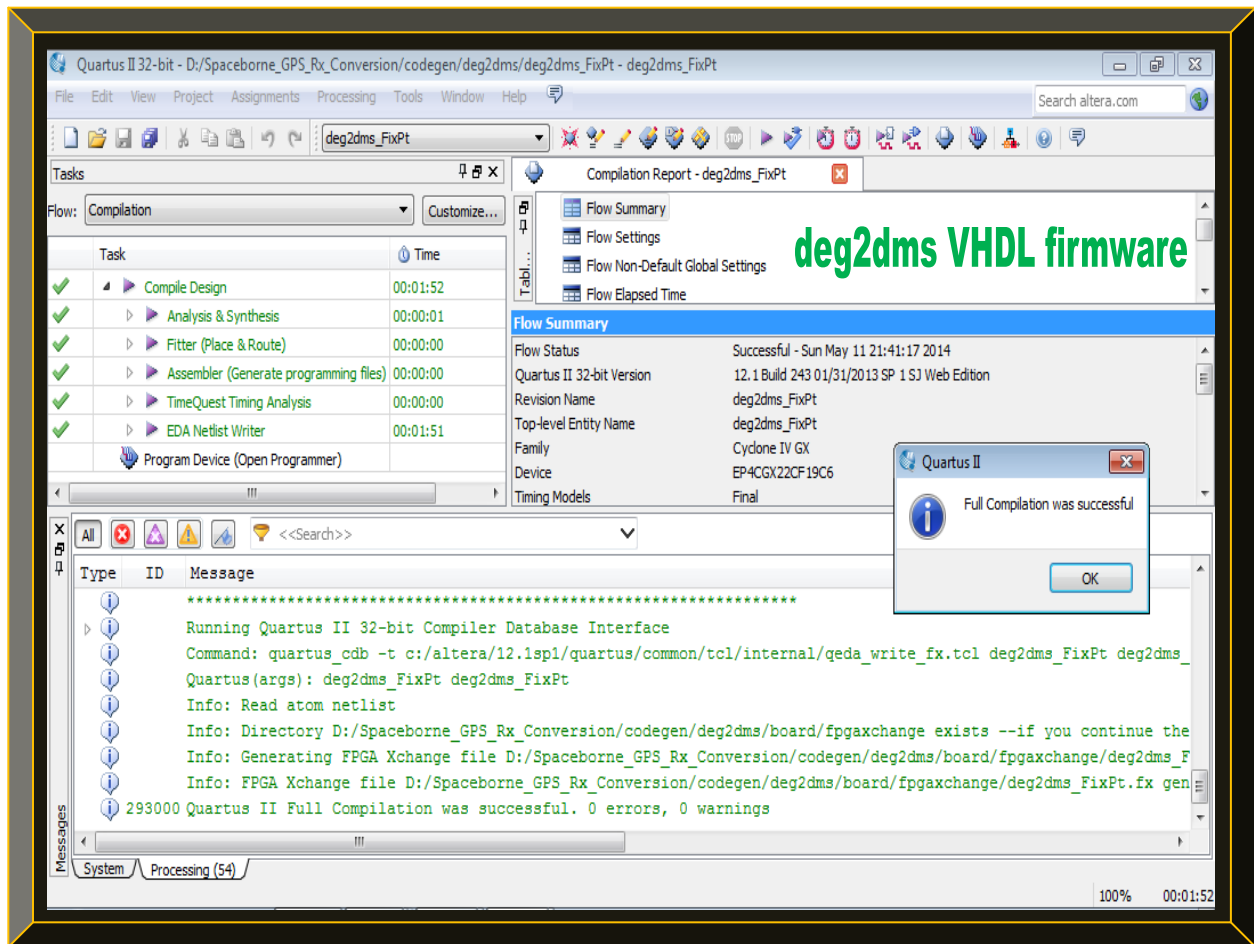


Figure 5.30: deg2dms firmware compilation result – ready now to be programmed into a FPGA

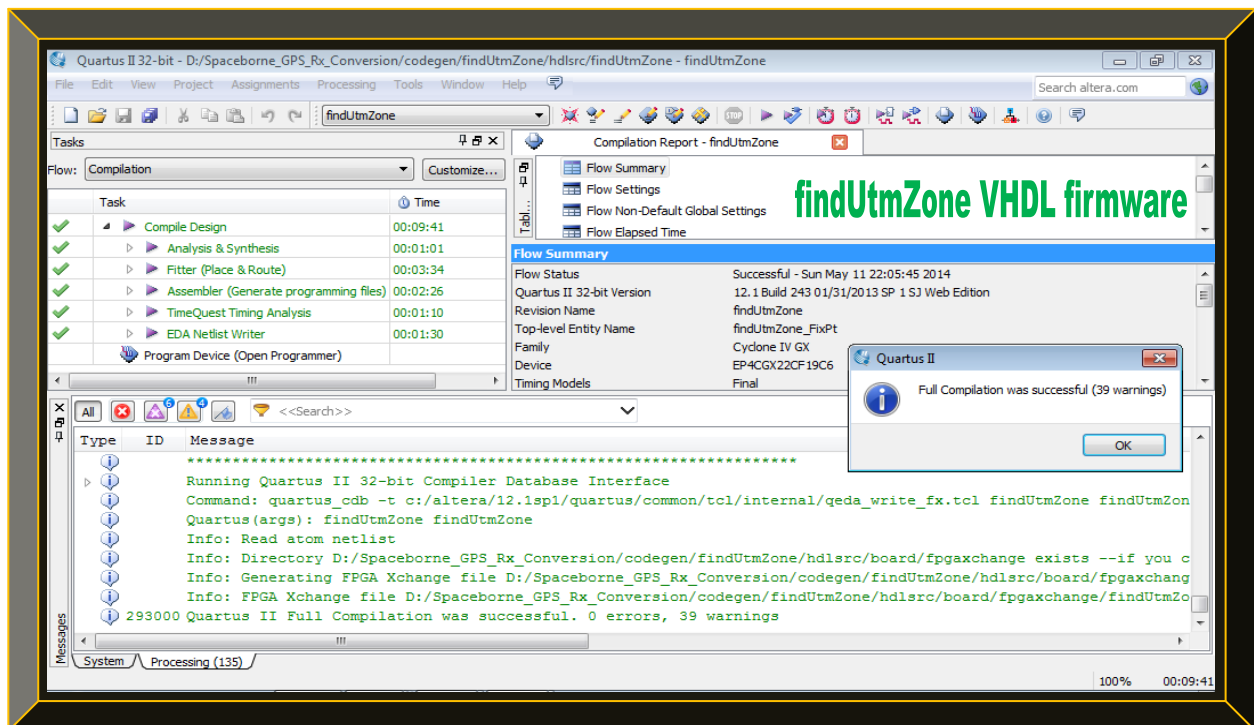


Figure 5.31: findUtmZone firmware compilation result – ready now to be programmed into a FPGA

5.3 The GPS Receiver and Discrete Generated Algorithms Results Interpretations

This section interprets the results of the researched GPS receiver, that is, the results of the software-defined Matlab floating-point GPS receiver as a unit, as well as the various discrete GPS receiver algorithms of the three generated implementations types are explained in-depth.

5.3.1 Software-defined Matlab Floating-point GPS Receiver Results Interpretations

The results of the software-defined Matlab floating-point GPS receiver are explained as follows:

Figure 5.1 simply confirms that the pre-captured real GPS signal is valid and can be processed. Figures 5.2 - 5.5 demonstrate the effects of Doppler frequency shift to GPS signals. Figure 5.2 is a plot of the captured GPS L1 C/A signal, subjected to a ± 5 kHz (-5kHz to + 5kHz on either side of the pre-captured GPS L1 C/A IF) Doppler search bins based-on parallel code phase acquisition algorithm implemented in the researched GPS receiver. Figures 5.3 to 5.5 depict the same scenerio but now, respectively at ± 10 kHz, ± 50 kHz and ± 100 kHz Doppler search bins. It is observed that, at ± 5 and ± 10 kHz Dopplers – which simulate a low-dynamics situation on Earth, the acquisition process is the fastest and the captured GPS SV signals are more visible, as shown in Figures 5.2 and 5.3. At ± 50 and ± 100 kHz Dopplers – which simulates a high-dynamics search situation in space (not a typical space-borne case, since the GPS receiver and the pre-captured GPS signal dataset are static), the acquisition process is the slowest and the GPS SV signals are less visible, as portrayed in Figures 5.4 and 5.5. Furthermore, at ± 50 and ± 100 kHz Doppler search bins, 9 of the 10 GPS satellites (max of ten channels were specified in the settings and nine satellites acquisition threshold was >2) that were searched in the pre-captured dataset (*Multipath.bin*), were acquired as shown in Figures 5.6 to 5.8, as well as in Figure 9.2 of Appendix C. These results, thus validate a functional high-dynamics Doppler search algorithm.

The tracking result is also successful and it is observed that, the more the number of channels to be tracked or used, the longer the tracking process. The benefits however, is a more accurate navigation solution, as justified by the DOP values in Figures 5.8 and 5.9. Furthermore, the navigation bits preambles of GPS satellites 32, 14 and 1 of channels 7, 8 and 9 respectively (refer to Figure 5.8 as well as Figures 9.9 to 9.11 of Section 9.2 in Appendix C); could not be deduced due to the excessive Doppler frequency shift, as shown in Figure 5.8. Thus, only the top six GPS satellites of PRN 2, 6, 7, 10, 24 and 32 were tracked in that order; because of (i) their strongest signal levels (see Figure 5.6), (ii) less Doppler associated to the GPS L1 C/A IF and code phases (see Figure 5.8) and (iii) the presence of a valid navigation bit preamble (see Figures 9.3 to 9.8).

Figure 5.9 bottom right, illustrates a skyview geometry of the tracked top six GPS satellites and as shown, the GPS satellites DOP is poor, although the mean DOP values of GDOP, VDOP, HDOP, PDOP and TDOP as shown in Figure 5.8 bottom, were within acceptable limits. GPS satellite of PRN 2, has the strongest signal level or strength but also the lowest elevation angle. Using an elevation mask of 10 or lower, this SV signal can be discarded, if it is considered as a pseudolite – signal from a device (e.g. a ground transmitter / jammer) that mimics GPS satellites.

The navigation solutions from the tracked navigation bits were computed and the outcome is the GPS receiver position (more correctly the position of the GPS front-end antenna) of where the processed dataset was captured. As shown in Figure 5.9 bottom left, the mean GPS receiver position is found in UTM coordinate and the UTM zone of 32 is deduced (see Figure 5.9 top). Channel 10 as shown in Figures 5.8 top and 9.12, was not acquired because its signal level was below the acquisition threshold of 2 as defined in the settings. Thus, it was un-tracked (has the status OFF) and as a result, no navigation data message was decoded and hence emerges blank.

Figure 5.9 top, depicts the computed UTC date and time of when the processed dataset (*Multipath.bin*) was acquired. The date (01/15/2007) is correctly computed, as it is the same as the date of when the *Multipath.bin* file (Figure 5.32) was named. However, the UTC time (09:59:57) is incorrect by ~ 25 minutes late; which could be as a result of the renaming of the file *Multipath.bin*, ~ 25 minutes later. Figure 9.13 illustrates some relevant GPS orbital parameters.

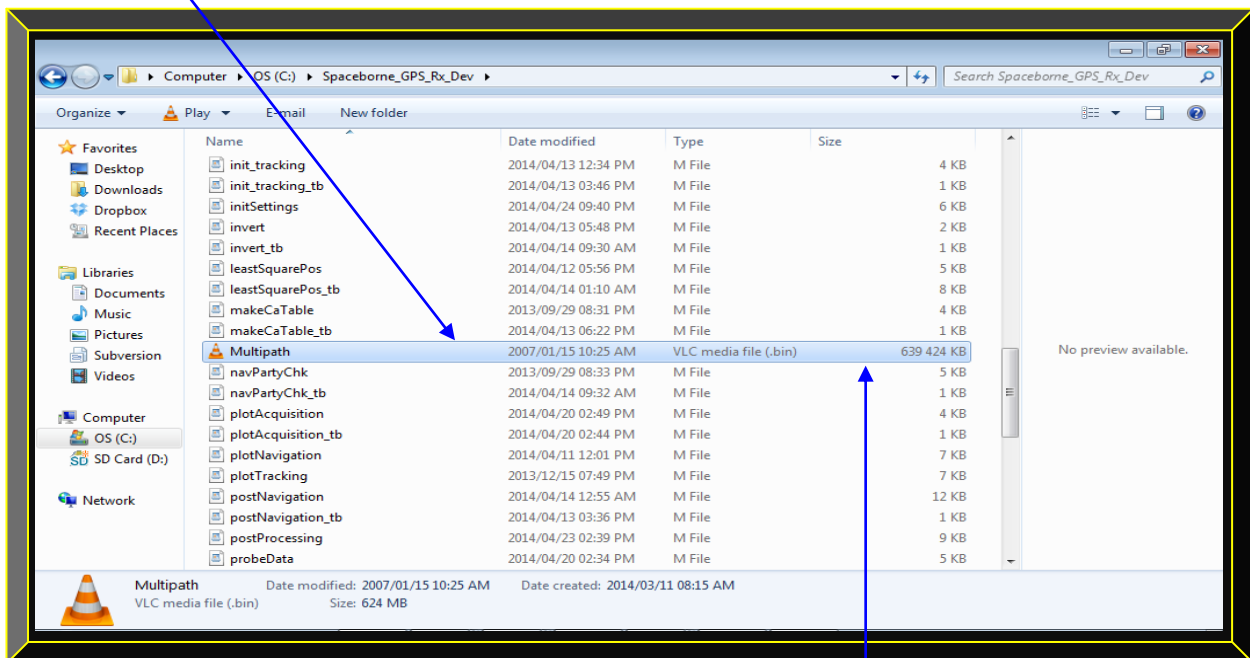


Figure 5.32: Illustration of the used pre-captured GPS signal dataset (*Multipath.bin*) – date and time

5.3.2 The GPS Receiver Generated Algorithms Results Interpretations

As discussed in Section 5.2, only five of the Matlab floating-point software-defined GPS receiver algorithms (*calcLoopCoef.m*, *check_t.m*, *clsin.m*, *deg2dms.m* and *findUtmZone.m*) were each successfully converted to equivalent (i) Matlab fixed-point algorithms (ii) Simulink fixed-point models and finally (iii) VHDL algorithms. Therefore, only the results of these mentioned algorithms are analysed, validated and explained. Table 5.2 clearly depicts the comparison and analysis of the conversion errors involved. It is observed that, each of the algorithms generated to Matlab fixed-point, Simulink fixed-point and VHDL; produced the same outputs as their corresponding Matlab-floating point algorithms, with insignificant errors. As a result, all the three types of generated GPS algorithms (besides the *clsin.m* that failed VHDL verification) are functionally equivalent, since they all yield the same results as their corresponding Matlab floating-point algorithms (see Table 5.2). Figures 5.29 to 5.31 exemplify three of the generated VHDL algorithms, successfully compiled in Altera Quartus II, with some few timing warnings.

The reasons for the failures in the other GPS algorithms, as detailed in Section 9.4 include:

- Some of the Matlab constructs (e.g. while loop) and built-in functions (e.g. *asin*, *atan*, *rem* etc) in Matlab R2013a, do not currently support Matlab floating to fixed-point conversions. Some do support floating to fixed-point conversion but not to VHDL.
- Some of the Matlab floating-point algorithms seems poorly written, though they do work.
- The Matlab R2013a package used was at times unstable during the conversion process.

5.4 Summary

This chapter discussed in detailed, the researched GPS receiver results, as well as how the results were interpreted and validated. Only the Matlab floating-point software-defined GPS receiver yielded the expected results as a unit (i.e. acquired, tracked, decoded and processed the pre-captured GPS L1 C/A signal dataset to compute position, UTC date and time). In the researched workflow, attempts were made to convert the Matlab floating-point software-defined GPS receiver algorithms to VHDL; from which only 5 out of 31, were successfully converted and four of the five were verified in VHDL and three compiled in Altera Quartus II. Equivalent Matlab fixed-point and Simulink models of the five algorithms were also created and test-benches for all 31 GPS algorithms were written to define and simulate each Matlab floating-point GPS receiver algorithms prior to conversion. The equivalent GPS algorithms results are exact with few errors.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

Chapter 6 concludes the research and recommends future work. The conclusions reached, relate to the research problem, objectives, literature review, methodology, results and the contributions.

6.1 Conclusions

The following sections of the research are evaluated to draw the conclusions in what follows:

- The research was undertaken to address the problems of inefficient space-borne GPS receiver algorithms and the high-cost associated with the design and implementation. The problems were addressed by developing open-source GPS algorithms, using Matlab HDL coder workflow and Altera software, with focus on the Xilinx low-cost space-grade FPGA.
- The primary or main objective was to implement a software-defined GPS receiver. This was achieved using Matlab. The receiver was used to post-process a pre-captured, real GPS L1 C/A signal to acquire, track, decode, process and compute the position, UTC time and date.
- The secondary objective was to improve the primary objective. This was reasonably attained using the Matlab R2013a HDL Coder workflow. Five of the GPS receiver algorithms (*calcLoopCoef.m*, *check_t.m*, *clsin.m*, *deg2dms.m* and *findUtmZone.m*) were improved from Matlab floating to fixed-point implementations; making them more compact, efficient and implementable into a low-cost microcontroller. These five algorithms with the exception of *clsin.m*, were further converted to industry standard VHDL, simulated and as well verified.
- The tertiary objective was to implement a firmware and to propose a low-cost space-borne GPS receiver roadmap. Three of the five generated VHDL algorithms were compiled to programmable firmware using Altera Quartus II. Algorithms in fixed-point do enable low-cost design; thus, the proposed device is the Xilinx low-cost space-grade Virtex-5QV FPGA.
- The literature review was conducted to analyse and select the most suitable acquisition and tracking techniques; in terms of efficiency, robustness and the ease to implement and develop, in-line with the research objectives. From the analyses, the parallel code phase search and the combined carrier and robust code tracking techniques were implemented to give a high-dynamics Doppler search bin exceeding $\pm 100\text{kHz}$, demonstrating LEO viability.
- The methodology employed was to utilise Matlab R2013a HDL coder workflow and free development tools from Altera. The rationale was to use the chosen development platform to improve open-source GPS receiver algorithms and use in high-dynamics LEO GPS receiver.

- The results substantiated that, if the software-implementation post-processed a real GPS pre-captured dataset to correctly compute position, time and date; and if some of these algorithms were as well generated in VHDL and yielded accurate results as their Matlab floating-point equivalents, then a valid FPGA firmware implementation, in-line with the research tertiary objective is feasible, as justified by the results in Sections 5.2.3 and 5.2.4. However, the failures in the conversion processes revealed limitations in the methodology. Furthermore, the GPS receiver velocity computation was not viable, because a pre-captured static GPS signal was used. That is, the GPS receiver (i.e. the antenna) has to be dynamic. Velocity can only be computed (and accurately), provided the receiver's antenna is moving.
- The research contributions entail a workflow that utilises, tailors and improves free open-source GPS algorithms, in terms of efficiency for space application, using free and low-cost tools and devices. The GPS receiver algorithms can be converted to three equivalent types.

6.2 Recommendations

In what follows are recommendations i) to issues encountered and ii) for future research.

6.2.1 Recommendations to issues encountered

Problems were encountered during implementation and the following solutions are suggested:

- Always use the latest and valid software tools and toolboxes to mitigate bugs and incompatibility issues. Additional benefits include more features and improved support.
- Ensure Matlab R2013a constructs and functions to be used support HDL code generation. A list of supported constructs and functions is found in Matlab help. Re-structure the Matlab algorithms listed in Section 9.4 to coding styles suited for HDL code generations.
- For Matlab built-in functions that do not support HDL code generation and their use is imperative; examine the Matlab code in great details and implements it as a black-box approach and try re-coding it in VHDL. The presented Matlab to HDL workflow supports legacy implementations of VHDL algorithms, alongside those automatically generated.
- If an algorithm cannot be converted due to its size, rescale it to smaller modules and keep the coding in each as simple as possible. Test each discretely and later integrate. Avoid using constructs such as while loops and if statements. Preferably use for loops and case structures respectively. Employ the use of Matlab system objects (instead of Matlab functions) if possible. Algorithms implemented using Matlab system objects are more efficient in the coding structure and as a result, executes faster and occupies less memory.

- If issues persist, consider re-starting Matlab and perhaps re-booting the computer. The HDL coder of Matlab version R2013a (in my case) had issues of not updating (the added or new design file fails to replace the previous one and was giving conversion issues). The work-around was to re-launch it (close and re-open it) prior to any new conversions.

6.2.2 Recommendations for future research

The research accomplished in this thesis, sets the foundation for the following research topics:

- Implement an equivalent Matlab fixed-point software-defined GPS receiver unit.
- Implement an equivalent Simulink fixed-point software-defined GPS receiver unit.
- Implement an equivalent VHDL and or Verilog firmware-defined GPS receiver unit.
- Implement a complete functional space-borne GPS receiver unit – post processing.
- Implement a complete functional space-borne GPS receiver unit – in real-time
- Implement and design a physical space-borne hardware-defined GPS receiver.
- Implement and design a physical low-cost space-borne hardware-defined GPS receiver.

Other areas of research (software/firmware/hardware) which can emanate include the following:

- A complete space-borne GPS receiver unit that do both orbit and attitude determinations: that is, it should be able to calculate position, velocity and as well determines orientation.
- In-situ ionospheric research: that is, the GPS receiver in real-time, should be able to use the ionospheric data found in the decoded navigation message to classify the ionosphere.
- An onboard computer which implement the space-borne GPS receiver; either as an onboard hardware module or as an in-built real-time operating system task, which can be remotely upgraded (bug fixes /new functionalities) by the ground-station during overpass.
- A low-cost onboard computer in-cooperating all the above GPS features or research.

Complete in this regard means, the GPS receiver should has the GPS front-end and antenna integrated as well (as a discrete hardware or software-defined) with the correlator and base-band.

Low-cost in this regard means the GPS receiver is made affordable for research purpose.

APPENDIX A

GPS SIGNAL L1 C/A ACQUISITION AND TRACKING TECHNIQUES

7.1 Introduction

Acquisition and tracking are the two fundamental tasks performed by any GPS receiver correlators (be it hardware or software-defined types). Several GPS signal acquisition and tracking techniques have been researched and implemented over the years, due to challenges (GPS sources of errors) encountered by GPS receivers and the various fields of applications involved – especially in high-dynamics applications, geodesy and survey. The techniques make use of one or more mathematical properties of the GPS signal. Depending on the GPS signal parameters exploited, a technique can be considered as simple or sophisticated. Each of the GPS signals acquisition and tracking techniques have advantages and disadvantages that influence the quality of a GPS receiver. In light of this, GPS receiver correlators acquisition and tracking algorithms were examined and the theoretical aspects are briefly discussed in what follows. The first and second sections respectively cover the acquisition and tracking techniques that were analysed to choose the most viable based-on ease of implementation and performance efficiency.

7.2 GPS L1 C/A Signal Acquisition Techniques

These techniques are used to perform GPS L1 C/A signal acquisition and include the followings.

7.2.1 Mitel GP2000 Acquisition and Tracking Algorithms

The obsolete Mitel GPS correlator chipset was one of the most widely used in most space-borne GPS receivers as its popularity availed itself to open-source GPS receiver platforms. It employed the basic traditional technique of acquiring and tracking GPS L1 C/A signal, which is inefficient in terms of execution speed and memory usage. Its GPS correlator architect performs a search in both the code phase and carrier frequency planes across all 1023 code phases at a search rate of 0.25 chips/ms to a series of frequency bins in 500Hz steps (for a ± 10 kHz Doppler) until the GPS signal is detected. Signal detection occurrence then closes the code and carrier tracking loops. Code tracking is achieved using a Phase Locked Loop (PLL) since PLLs are good at maintaining lock, due to their very narrow loop bandwidths. However, these same properties make them poor at acquiring lock without help from Frequency Locked Loop (FLL). Carrier tracking is achieved using FLLs. FLL offers superior dynamic performance, robustness and insensitivity to interference over PLLs; thus, the carrier tracking loop aids the code tracking loop to yield carrier cycles and phase measurements, which smoothes the code pseudo-ranges (Mitel, 1998: 14 & 15).

7.2.2 Tong or Serial Search (Time Domain Correlation) Acquisition Technique

This section covers parts of the sequential search technique not addressed earlier. Kaplan and Hegarty (2006: 223-226) refer to this technique as a Tong search detector, which is based on sequential variable dwell time detection. In Borre *et al.* (2007: 76 - 78), this method is popularly called serial search. It is common in signal acquisition in CDMA systems to which GPS belongs.

With this approach, the incoming GPS signal is first multiplied by locally generated GPS PRN code sequences, corresponding to the respective GPS satellites of interest (practically, it is often advisable to generate replica signals for all 32 GPS satellites, since there is no prior knowledge of which GPS satellite is operational, except only after acquisition), followed by multiplying the outcome with a locally generated GPS carrier signal. These processes are respectively called code and carrier wipe-offs and what remains of the GPS signal is just the GPS navigation message. Since the GPS In-phase (I) signal generated at the satellites will not necessarily correspond to the demodulated (I) component (because the phase of the received signal will be unknown due to GPS error sources), the basis is to have an (I) and Quadrature (Q) components of the received GPS signal. These two are respectively achieved by multiplying with a locally generated GPS carrier signal and a 90° phase-shifted version of the locally generated carrier.

The (I) and (Q) signals are then integrated over 1ms, corresponding to the length of a C/A code (i.e. 1023 chips / 1023 kHz = 1ms), and finally squared and added. Ideally, the GPS signal power should be located in the (I)-arm of the signal, as the C/A code is only modulated onto that. However, in practice, it is at times located in the (Q) signal part of the demodulated GPS signal and to ascertain GPS signal detection, it is imperative to investigate both the (I) and (Q) signals.

In summary, GPS signal acquisition using serial search technique involves first generating respective GPS PRN codes followed by carrier frequency generation, then integration, squaring and finally summation of all the (I) and (Q)-arms results. Figure 7.1 summaries this technique.

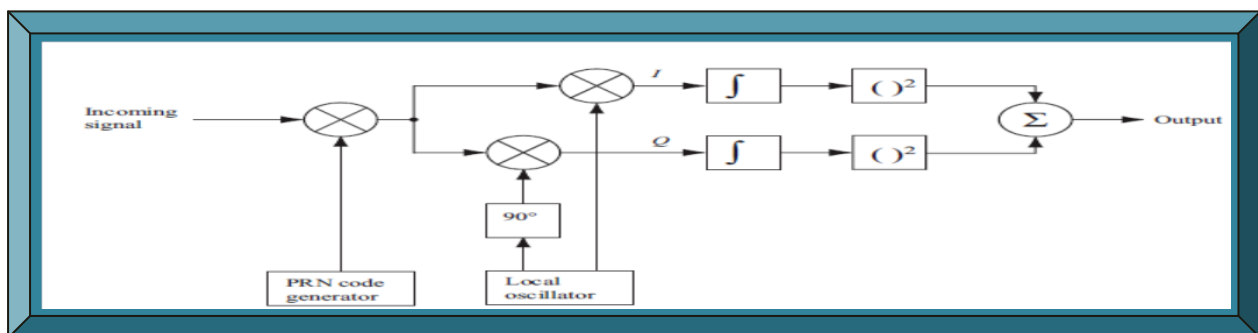


Figure 7.1: GPS signal acquisition serial search technique
(Adapted from Borre *et al.*, 2007: 76)

7.2.3 Parallel Frequency Space Search (Delay & Multiply FFT) Acquisition Technique

Accomplishing a complete GPS signal acquisition using serial search algorithm do consumes time and memory – as it requires sequentially searching through all possible values of both code phases and carrier frequencies (i.e. $1023 \times 41 = 41943$ combinations) of all 32 GPS SV. Therefore, if one of the two parameters (code phase or carrier frequency) could be removed from the search procedure or if possibly, implemented concurrently, the procedure performance would increase considerably (Borre *et al.*, 2007: 78 - 81). Hence the name, parallel frequency space search acquisition technique. This method is briefly discussed next and abridged in Figure 7.2.

As the name suggests, the search of the frequency parameter is parallelised by applying Fourier transforms to perform a transformation from time domain into frequency domain. Akin to the serial search technique, a locally generated GPS PRN sequence (with a PRN code corresponding to each specific GPS satellite and a code phase between 1023 chips i.e. 0 to 1022 chips) is first multiplied by the incoming GPS signal. The resulting signal is then transformed into frequency domain by either a Discrete Fourier Transform (DFT) or a Fast Fourier Transform (FFT). FFT is the faster of the two but requires a radix-2 length input sequence (i.e. 2^n , where n is a positive integer value). Conversely, if DFT is utilised instead, the accuracy of the determined frequency depends on its length and this corresponds to the number of samples in the data analysed. Contrary to the serial search technique, the parallel frequency space search only steps through the 1023 different code phases, at the expense of a frequency domain transformation with each code phase. Therefore, by parallelising the frequency space search, it should be viable to do a faster GPS signal acquisition technique implementation, compared to the serial search technique.

By aligning the incoming GPS signal and the locally generated PRN codes, the FFT output will have a significant peak at the IF plus Doppler offset frequency. The possible peak frequency of all the components is finally found by computing the absolute values, as illustrated in Figure 7.2.

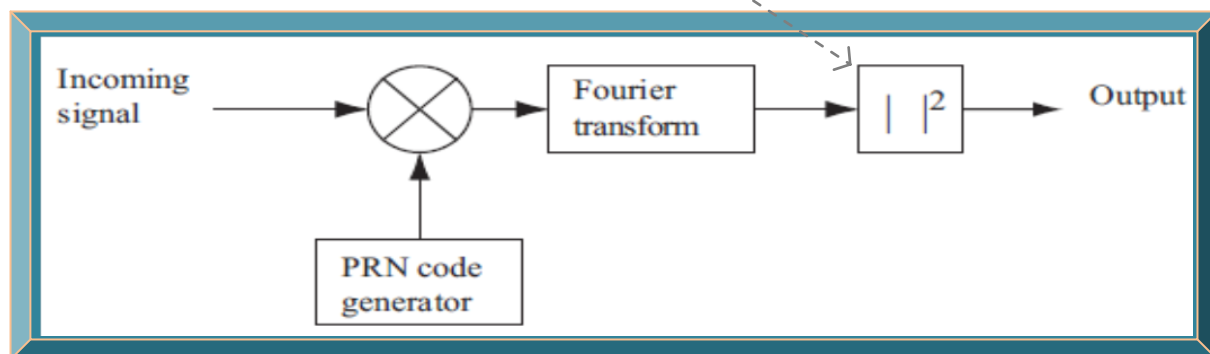


Figure 7.2: Depiction of parallel frequency space search technique
(Adapted from Borre & Strang, 2012: xxix)

7.2.4 Parallel Code Phase Search (Circular Cross-correlation) Acquisition Technique

While the serial search technique involves correlating both GPS PRN code phases and carrier frequencies (1023×41), the parallel frequency space search technique entails just the GPS PRN code phase (1023), as it eliminates the necessity to search through all 41 possible frequencies. Therefore, if the acquisition is parallelised in the code phase dimension, only 41 steps will be performed compared to the 1023 in the parallel frequency space search algorithm and 41943 in the serial search algorithm. Hence, this technique is usually referred to as parallel code phase technique. Instead of multiplying the input signal with a PRN code with 1023 different code phases as in the other two acquisition techniques, it is more effective to perform a correlation with the incoming GPS signal and a PRN code, without shifting the code phases. This type of correlating process is termed circular cross-correlation (Tsui, 2005: 139; Borre *et al.*, 2007: 82).

As summarised in Figure 7.3, the incoming GPS signal is first multiplied by a locally generated carrier signal and a 90° phase shifted version; to respectively yield the (I) and (Q) components, that are Fourier transformed to a complex input signal. Next, a generated PRN code is Fourier transformed into frequency domain and the result is complex conjugated. The Fourier transforms of the input signal and that of the complex conjugate of the generated PRN code, are then multiplied and the result of the multiplication is then transformed into time domain by an Inverse Fourier Transform (IFT). Finally, an absolute value is computed (because DFT / FFT / IFT give complex outcomes) and the output is the correlation between the input signal and the PRN code. If a peak is present in the correlation, the index of this peak marks the PRN code phase of the incoming GPS signal. The Fourier transform of the generated PRN code must only be performed once for each acquisition – one Fourier transform and one IFT for each of the 41 frequencies. This technique computational efficiency relies on the implementation of these functions, since the PRN code phase is more accurate compared to the other techniques, as it gives a correlation value for each sampled code phase (i.e. for a 10MHz sampler, a sampled PRN code has 10000 samples, which translates to a code phase accuracy of 10000 distinct values rather than 1023).

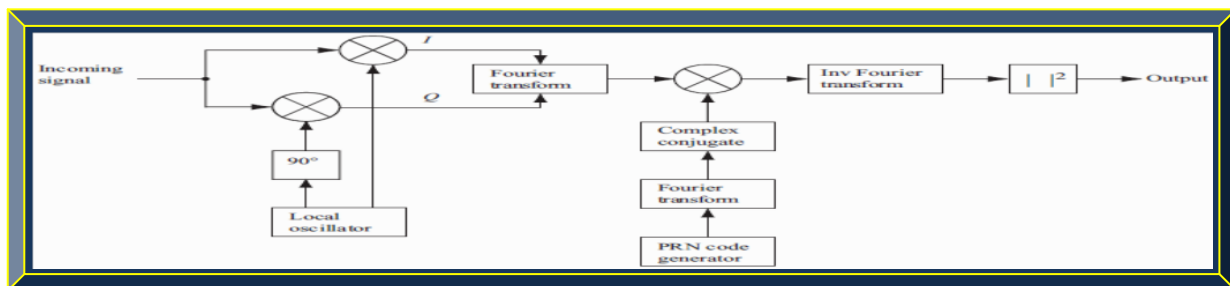


Figure 7.3: Portray of parallel code phase search algorithm
(Adapted from Borre *et al.*, 2007: 83)

7.2.5 Block (Non-coherent Integration) Acquisition Technique

The acquisition techniques presented thus far are considered to use coherent integration, which means the acquisition is performed continuously within a specified duration, usually 1ms period.

Since circular cross-correlation performs acquisition in just 41 steps, it is evident that, it is the fastest; however, it is based-on acquiring 1ms of data, which at times cannot detect a weak signal. Hence, longer data records are needed to acquire a weak signal and one way to process more data is through non-coherent integration. For example, if 5ms of data are used, the data can be divided into five 1ms blocks. Each 1ms of data are in turn processed separately and the results are added together in the end; hence, the name blocks acquisition. The cost of using this technique is simply the extra number of operations involved, apart from the last stage addition. Non-coherent integration offers improvement compared to the coherent method, which however uses fewer operations compared to the non-coherent method (Tsui, 2005: 144; Shi *et al.*, 2009).

Additional advantage of this technique is the improvement in signal to noise ratio and its ability to keep performing acquisition on successive 1ms of data and summing the results. The final result is compared with a set threshold and whenever the final result exceeds the threshold, the incoming GPS signal is detected. A maximum data length can be chosen and if an incoming GPS signal cannot be detected within this data length, the acquisition process will automatically stop.

By using this approach, strong incoming GPS signals can be detected in successive 1ms of received GPS data, but weak signals will take a longer data length. Furthermore, if phase shift or bit transition occurs in GPS navigation data during acquisition, only the 1ms of data with the phase shift will be affected and the other half or successive 1ms of data will not be affected. As a result, this technique mitigates bit transitions in the middle of a navigation message acquisition.

Bit transition normally signifies the start of a new GPS navigation message and should this occur in the course of a 1ms navigation data message acquisition, the already acquired data will be corrupted (since it will be incomplete) and will require re-acquisition; otherwise, the navigation solution based-on this corrupted data will be wrong. The benefits of block acquisition technique often render its deployment in conventional hardware-defined GPS receivers (Tsui, 2005: 144).

Besides being utilised in GPS signal acquisition, the non-coherent integration technique is also used in GPS signals tracking. However, prior to applying non-coherent integration, coherent integration must first be implemented and is discussed under tracking techniques in Section 7.3.

7.2.6 Acousto-optic Acquisition Technique

In Bazzi *et al.* (1993), a GPS correlator, based-on the acousto-optic technique was proposed and implemented. The technique utilises a laser wave as a spatial carrier for the GPS signals to be processed and both GPS signals are confined in an acoustic (bulk or surface) delay-line. The main challenge being fitting the 1ms GPS PRN code period into a $25\mu\text{s}$ aperture acoustic Bragg cell. This issue was addressed by developing and implementing a technique based-on time compression. Benefits of this approach include using a single Doppler-frequency search loop that yields a very rapid (speed of light) GPS signal acquisition without any code phase search needed, since the time of GPS signal arrival is obtained quasi-instantaneously once Doppler is resolved. Refer to Figure 7.4 for details regarding this method and Figure 7.5 for the acquisition system.

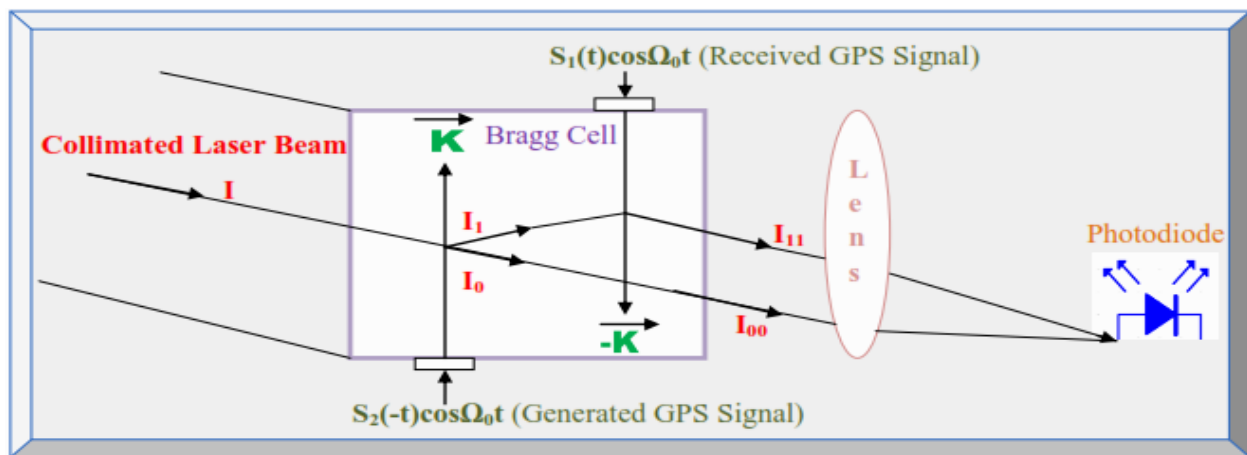


Figure 7.4: Space integration acousto-optic correlator principle
(Adapted from Bazzi *et al.*, 1993)

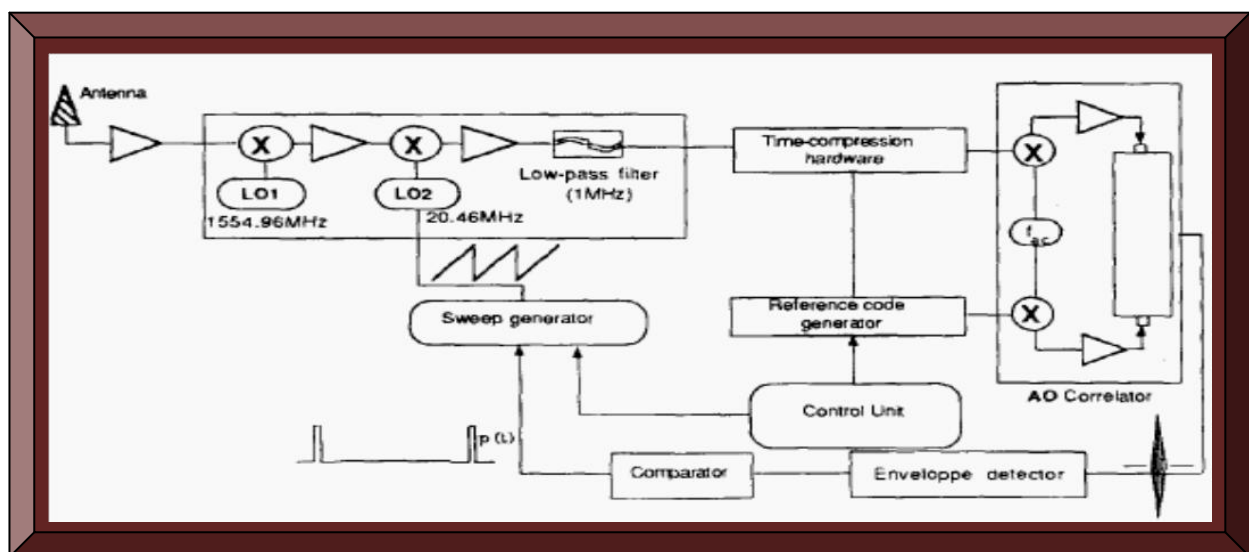


Figure 7.5: Acousto-optic GPS receiver acquisition system
(Adapted from Bazzi *et al.*, 1993)

7.2.7 Multiple Correlators Bank Acquisition Technique

According to Akopian *et al.* (2006), a more efficient way for acquiring GPS signals is to use a bank of correlators consisting of several stages of coherent and non-coherent integrations blocks with detection decisions. In this approach, they developed massive correlator structures with different distributions of initial search and validation tasks. This is depicted in Figure 7.6 and consists of two sections, namely Massive Correlator Banks (MCB) and Supplementary Correlator Banks (SCB). This approach requires a conventional massive correlator to check all possible options, leading to performing unnecessarily computations. However, the rationale to improve the technique was to exclude unnecessary fractions of search options after short integration lengths. Once a decision is made for most of the search options, this enables the MCB architecture to stop at the processing stage and transfers the task to check the remaining options to a smaller correlator bank (SCB). The MCB is now available to check other sets of feasible (code phase and frequency) pairs. Furthermore, MCB and SCB may utilise other algorithms with different integration methods and lengths, due to structural separations (Akopian *et al.*, 2006).

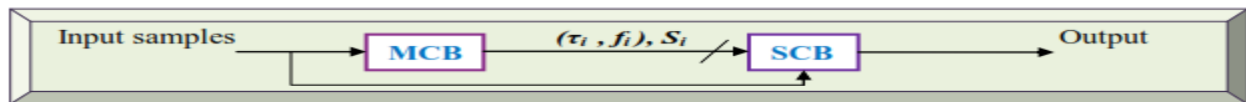


Figure 7.6: Multiple correlators bank acquisition technique
(Adapted from Akopian *et al.*, 2006)

7.2.8 Extended Multiple Correlators Acquisition Technique

Acquisition can be the most time-consuming operation if using a serial search technique. Although this is mitigated using faster techniques such as multiple correlators, matched filters and FFT; their operational complexity tends to consume too much power. To address the problem of acquisition time and complexity, the Extended Multiple Correlator (XMC) was postulated by Xiaowen *et al.* (2010). It simply consists of a combination of three variations of the locally generated PRN codes to correlate the incoming GPS signal, as illustrated in Figure 7.7.

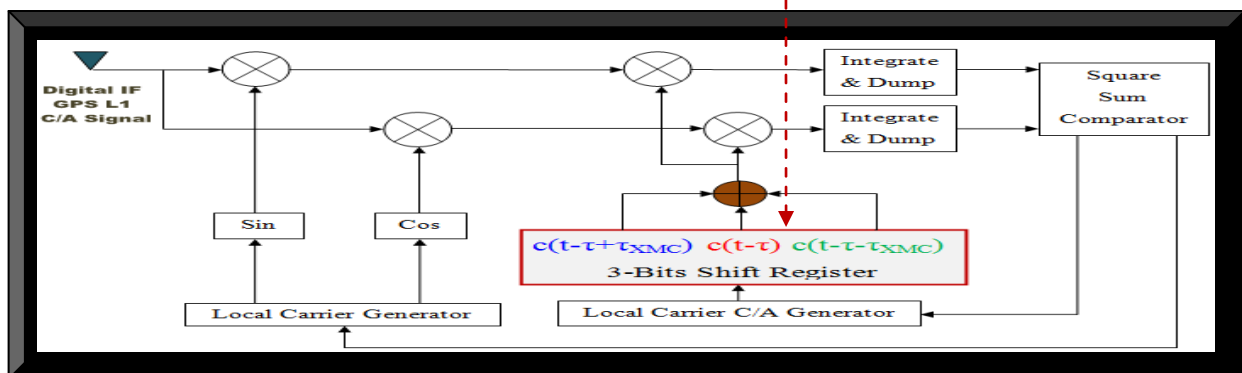


Figure 7.7: Single arm of the XMC acquisition technique
(Adapted from Xiaowen *et al.*, 2010)

7.2.9 Matched Filter Correlator Acquisition Technique

The matched filter technique was proposed to improve the GPS signal acquisition time. The algorithm calculates code phase (I) correlation when the buffer becomes full, from accumulating successive incoming GPS signal samples in buffers or shift registers. The correlation of (I) + 1 is calculated by shifting into the buffer new data samples that are received while at the same time, the oldest stored samples are shifted-out. The correlation outcome of subsequent code phases can be rapidly determined, requiring 977ns for each code phase, since the majority of the incoming data samples are saved in memory. However, to save the incoming GPS signal samples, it requires an additional 1023-register buffer and the buffers must be initially filled before any correlation results can be determined, so a further 1ms should be added to the total acquisition period. Since it requires the same number of additions and multiplications to complete the PRN 1023-phase acquisition, its computational complexity is equivalent to the serial search technique (El-Rayis *et al.*, 2010; Ta *et al.*, 2012). Figure 7.8 portrays the matched filter acquisition method.

The match filter technique presented thus far is known as Conventional Digital Matched Filter (CDMF) and other variances with improved acquisition time include the Differential Digital Matched Filter (DDMF) and the Segment Processing Digital Matched Filter (SPDMF).

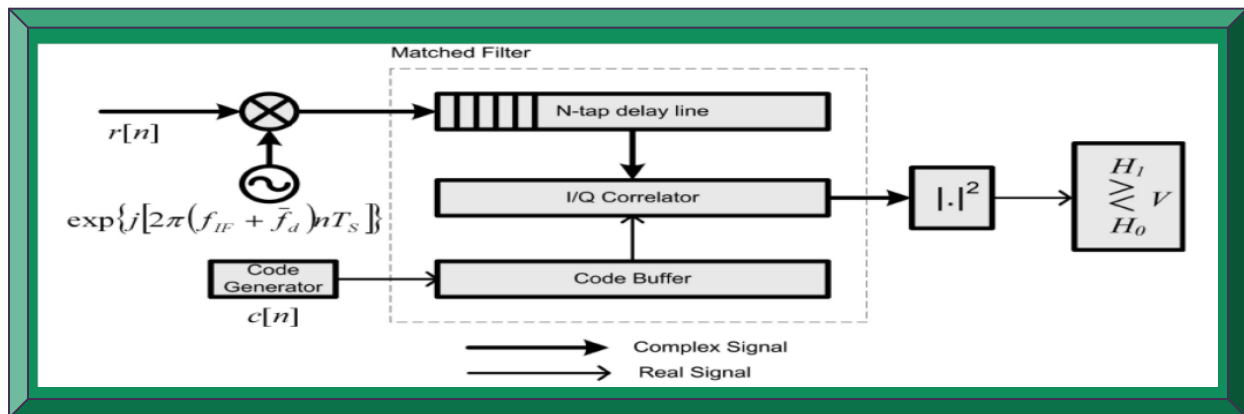


Figure 7.8: Portray of matched filter acquisition technique
(Adapted from Ta *et al.*, 2012)

7.2.10 Reconfigurable Instruction Cell Array Acquisition Technique

El-Rayis *et al.* (2010), introduced a new correlation engine targeting critical GPS positioning performance. The correlator processor is based-on the Reconfigurable Instruction Cell Array (RICA) paradigm. The correlator engine inherits various capabilities and advantages that are characteristic of the architecture of digital match filter correlators. Being a DSP-like correlator, it offers high level of programmability, bundled with a reconfigurable data-path and parallelism.

7.2.11 Averaging Correlator Acquisition Technique

The averaging method computes the mean of the incoming 5000 GPS samples to 1023 averaged samples. The 1023 new samples may signify the original chips of the C/A code, if the initial point of the averaging operation is selected in the right position. A full C/A code is represented by 5,000 samples and either four or five samples each are utilised to represent its chips. Therefore, the correct averaging initialisation point is one of five successive samples, since the C/A code is circular. As a result, this will generate 5 of the 1023 averaged-samples codes and one of the generated sequences is regarded as a good estimation of the original C/A code (or the best recovery). This best recovered averaged sequence will constitute the strongest peak amongst the other four and approximate the code phase in chips (i.e. 1/1023 ms). The resultant five correlation functions are a good ballpark figure of the 5000-points correlation function and as a result, can be used with a triangle fitting to refine the code phase (Alaqaeli *et al.*, 2003).

With this technique, the acquisition time should theoretically be reduced, as calculating five 1023-points FFTs and IFFTs requires less time (in software and in hardware) than the 5,000-point FFTs and IFFTs. However, 1023-points FFT (or IFFT) is not a power of two, which as a result, makes its implementation difficult. Instead of mitigating this issue with zero-padding-based solution which is complex, Alaqaeli *et al.* (2003) approach is to change the down-sampling rate from 1023 to 1024, which consequently down-sampled (or averaged) the 5000 samples to 1024 points. Similarly, the local PRN code is up-sampled to 5000 and then down-sampled to 1024 points. Therefore, the averaging correlator uses 1024 averaged samples and 1024-points modified C/A codes. This modified C/A code is as well considered a unique code relating to the chosen C/A code, since it cannot be generated from a different C/A code. According to Alaqaeli *et al.* (2003), the outcome of the technique is a rapid and easy way to implement the acquisition algorithm, which as well has latent real-time acceptable accuracy. Figure 7.9 depicts this method.

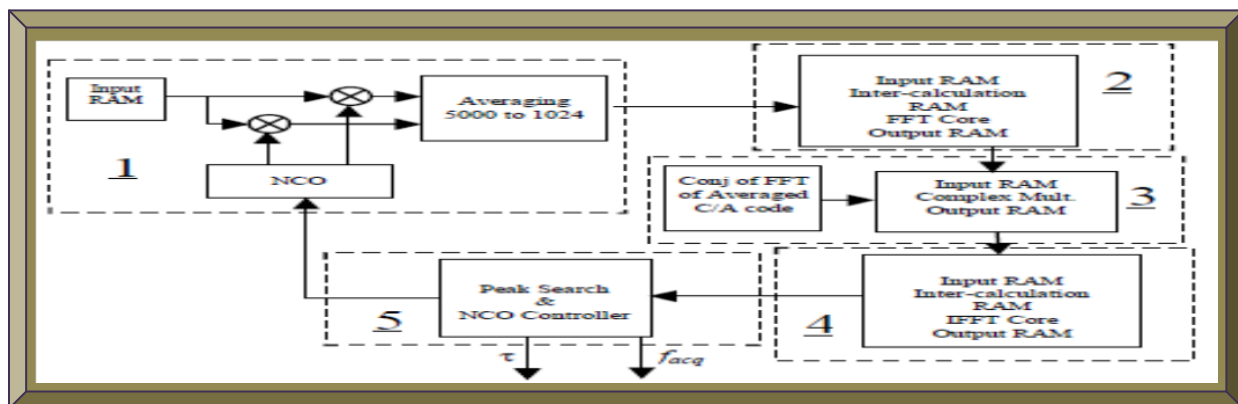


Figure 7.9: Illustration of the acquisition process
(Adapted from Alaqaeli *et al.*, 2003)

7.2.12 Bayesian Acquisition Technique

With this approach, O'Mahony *et al.* (2012) explore Bayes' probability theorem to predict the Doppler frequency in weak incoming GPS signals, by evaluating the noise distribution at given correlator outputs at sample points throughout the search space. Tests conducted by O'Mahony *et al.* (2012) indicate faster performance compared to other conventional acquisition techniques.

7.2.13 Wavelet De-noising Acquisition Technique

Similar to the Bayesian technique that focused on analysing the noise content of weak incoming GPS L1 C/A signals and predicting the Doppler frequency, Tian and Yang (2008), applied wavelets de-noising to conduct statistical analysis and to estimate the derivation of noise. The algorithm first made use of correlation and differential correlation methods to acquire very weak GPS L1 signals, followed by noise derivation estimate and noise model establishment and finally applying the wavelet de-noising process. This approach has the advantage of increasing the sensitivity of weak GPS signals (decrease background noise) without increasing integration time.

7.2.14 Lifting Wavelet Acquisition Technique

Faced with the challenges of acquiring weak GPS signals, Djebouri *et al.* (2006), proposed a substitute correlation algorithm, by applying lifting wavelet decomposition to develop a robust GPS signals acquisition system that improved acquisition performance and maintain continuity of service over noisy transmission channel. Their goal was to realise a robust, accurate and less complex GPS L1 signals acquisition system and to facilitate its implementation. This technique outperforms both FFT search and other signal decimation schemes in hostile/noisy environments.

7.2.15 Wavelet Transform Acquisition Technique

According to Jianguo *et al.* (2010), incoming GPS L1 C/A signals are occasionally very weak and susceptible to various sources of interference. Multipath and jamming are examples of interferences which change incoming GPS L1 C/A signals transiently and make it difficult to correlate. With wavelet transform being an ideal transient analysis tool, Jianguo *et al.* (2010) employed it to analyse the correlation power of GPS signals. Using a single reflection multipath model, wavelet transform is used to analyse the influence of multipath signals on the correlation function of PRN C/A code and can efficiently extract multipath from received GPS L1 signals.

7.3 GPS L1 Signal Tracking Techniques

Once a GPS receiver's acquisition stage has aligned the received and locally generated codes within less than half a chip period, a fine closed-loop synchronisation known as tracking takes over and keeps the two codes aligned and locked. If for some reasons the tracked GPS signal loses lock, the phenomenon is termed cycle slips and the phase measurement has to re-start. Generally, the tracking system in a conventional GPS receiver consists of a PLL or FLL for carrier phase tracking and a DLL for code tracking. The PLL and or FLL, generate carrier phase measurements, which are utilised solely or to smooth the raw pseudo-range measurements from the DLL. Similar to acquisition, tracking techniques also perform carrier and code wipe-offs.

FLL and PLL are respectively non-coherent and coherent systems (Gleason & Gebre-Egziabher, 2009: 34). A non-coherent FLL delivers a more robust tracking performance in high-dynamics environments and in very weak signal conditions. However, in order to enable precise pseudo-range rate, integrated Doppler measurements and carrier phase positioning; a GPS receiver must as well track the phase of the incoming carrier. Hence, a coherent PLL system is also required. Common practice is for the PRN code sequence to be acquired and tracked using a DLL. This can be achieved when implementing either non-coherent FLL or coherent PLL carrier tracking. However, different correlations and discriminators are required for each system. The dot-product discriminator is a common mechanism suitable for tracking with both coherent and non-coherent systems (Gleason & Gebre-Egziabher, 2009: 37). Figure 7.10 depicts tracking of a GPS signal.

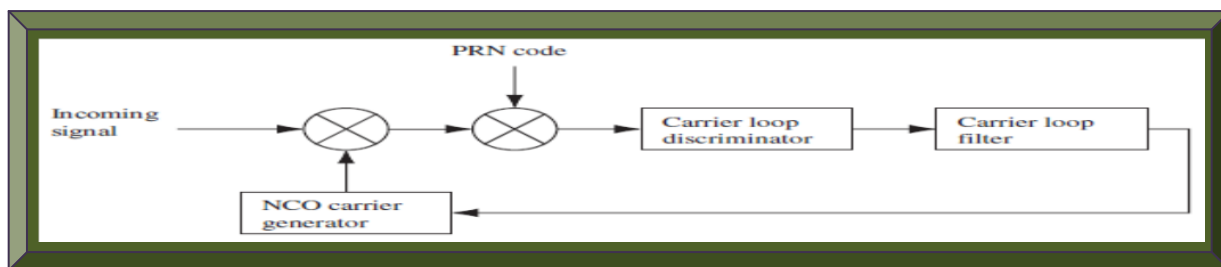


Figure 7.10: Illustration of a basic GPS receiver tracking loop
(Adapted from Borre *et al.*, 2007: 93)

The damping ratio and noise bandwidth affects a GPS receiver correlator tracking performance. A large noise bandwidth implies that the tracking loop can quickly lock to the real frequency but has a fairly vast frequency noise in the locked state and can lose lock easily, especially in high-dynamics. With a smaller noise bandwidth, it takes some time before the tracking loop is locked to the frequency but once locked, the frequency remains stable. A trade-off is therefore required.

Tracking techniques are used to perform GPS L1 C/A signal tracking and include the followings.

7.3.1 PLL and Costas Loop Carrier Tracking Techniques

As mentioned, a GPS signal is a bi-phase PRN coded signal, as a result, changes in carrier and phase caused by Doppler (relative motion of GPS satellites and receiver), multipath and the atmosphere; have to be tracked. The tracked outputs are coupled together and are discussed next.

As shown in Figure 7.10, the carrier frequency loop receives an (I) signal modulated only by the navigation data as the PRN C/A code is wiped-off or stripped from the input signal. To track this carrier frequency, a PLL ensures an exact carrier frequency replica is generated to successfully demodulate this navigation data, which is the essential goal. The PLL discriminator block finds the phase error in the locally generated carrier replica. The output of this discriminator, which is the phase error or a function of it, is then filtered and fed-back to the Numerically Controlled Oscillator (NCO), to adjust the locally generated carrier frequency. In this way, the local carrier frequency could be an almost exact replica of the carrier frequency in the incoming GPS signal.

In Kaplan and Hegarty (2006: 164 - 175), the drawback of ordinary PLL is its sensitivity to 180° phase shifts, that is affected by GPS navigation bit transitions. Therefore, for a PLL to be used in GPS receiver correlator, it has to be immune to 180° phase shifts. This limitation is mitigated with a modified PLL, known as Costas loop, shown in Figure 7.11. It consists of an (I) and a (Q)-arm. The outputs from these arms are correlated with the received GPS signals which are then filtered and their phases compared against each other through an arctangent comparator and feedback to the NCO via a carrier loop filter. A Costas PLL is unaffected by a 180° phase shift, as well as navigation bit or phase transitions; thus, it is the preferred PLL for use in GPS receiver correlator. The phase error is minimised when the correlation in the (Q)-arm is zero and the correlation value in the (I)-arm is maximum. That is, the discriminator outputs are zero when the real phase error is 0 and $\pm 180^\circ$; as a result, the Costas loop is insensitive to 180° phase shifts. Of the Costas discriminators, the arctan is the most precise, though the most time consuming. Costas loop purpose is to conserve if not all, then most of the received GPS signal power in the (I)-arm.

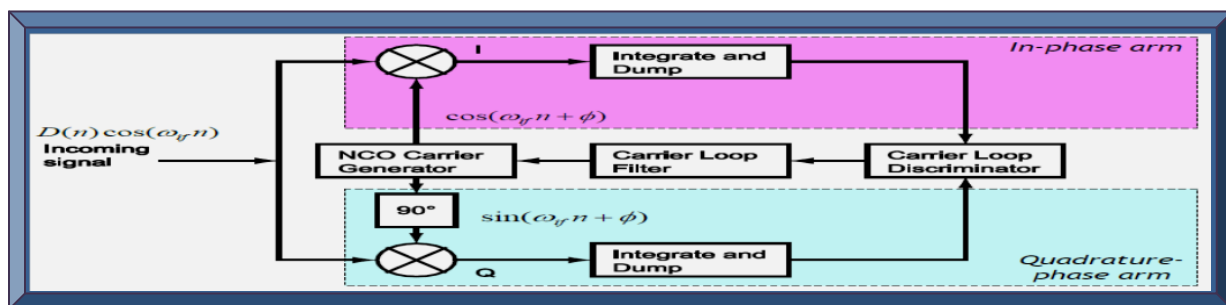


Figure 7.11: Illustration of a Costas PLL
(Adapted from Plausinaitis, 2009a: 17)

7.3.2 Basic DLL Early-Late Code Tracking Technique

The purpose of the code tracking loop is to keep track of the code phase of a specific PRN code in the incoming GPS signal. The code tracking loop output is a perfectly aligned replica of the PRN code. The code tracking loop in GPS receiver is a DLL, known as an Early-minus-Late tracking loop. The rationale behind the DLL is to correlate the received GPS signal with three replicas of the PRN code; designated as the *Early (E)*, the *Prompt (P)* and the *Late (L)* codes. According to Borre *et al.* (2007: 96 - 100), the DLL can be modeled as a linear PLL and thus, the performance of the loop can be envisaged based-on this model. In other words, the carrier and code loop filters design could be similar and just the parameter values are configured differently.

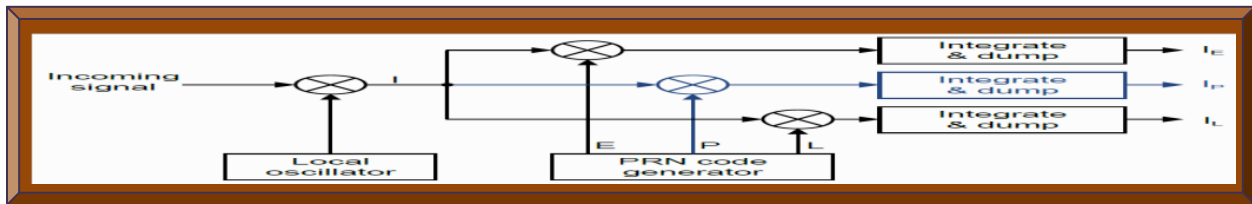


Figure 7.12: Simple DLL_E-P-L code tracking principle
(Adapted from Plausinaitis, 2009b: 7)

GPS PRN code tracking starts by converting the received C/A code to baseband; whereby, the incoming GPS signal is multiplied with a perfectly aligned local replica of the carrier frequency, followed by multiplication with the three (E), (P) and (L) code replicas. These three replicas are nominally generated with a spacing of $< \pm 1/2$ chip. After the second multiplication, the three code replicas outputs are integrated and dumped. The outcome of this process is a numerical value signifying how much a specific code replica correlates with the code in the incoming GPS signal. The three correlation outputs: $E_{(t)}$, $P_{(t)}$ and $L_{(t)}$ are then compared to determine the one that yields the highest correlation. Figure 7.12 depicts the (E), (P) and (L) concept. Figure 7.13 illustrates the process. In (a) $P_{(t)}$ is half a chip early (unaligned), so the PRN code phase must be decreased (i.e. the code sequence must be delayed). In (b) $P_{(t)}$ is now a quarter chip unaligned, so the PRN code is decreased again. In (c) $P_{(t)}$ is perfectly aligned and thus gives peak correlation indicating the PRN code phase is properly tracked. In (d) $P_{(t)}$ is a quarter chip late - unaligned.

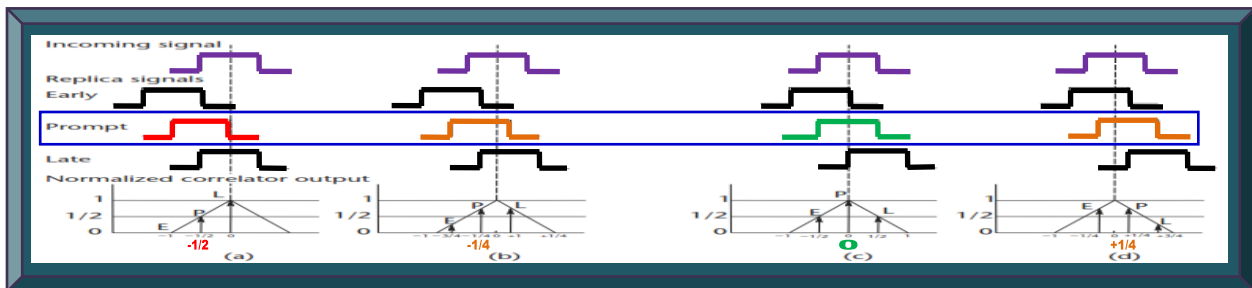


Figure 7.13: DLL_E-P-L code tracking process
(Adapted from Kaplan & Hegarty, 2006: 177)

7.3.3 Robust DLL Early-Late Code Tracking Technique

In Figure 7.12, the DLL with just the three in-phase E-P-L arms is optimal when the incoming GPS signal is locked in phase and frequency; however, when there is a phase error on the local carrier frequency, the signal will be noisier, making it more difficult for the DLL to stay locked unto the PRN code. Hence, a robust GPS receiver correlator DLL shown in Figure 7.14 is used.

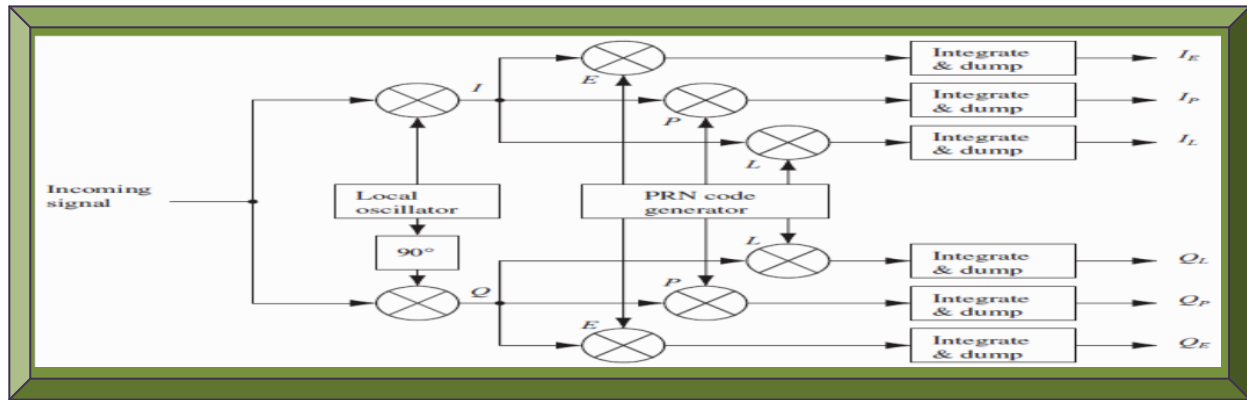


Figure 7.14: Robust DLL_E-P-L code tracking principle
(Adapted from Borre & Strang, 2012: xxxiii)

In this robust approach, the PRN code is tracked on both the (I) and (Q) components of the incoming GPS signal. In the initial state of the carrier tracking loop, the carrier phase of the tracked GPS signal is unknown. As a result, the energy from the correlations is likely to be situated in the Q-arm of the correlator. Once the carrier tracking loop successfully tracks the carrier phase, most of the energy will be located in the I-arm. The benefits are (i) PRN code tracking is faster as it provides better conditions to the carrier tracking loop and (ii) it does not depend on the phase in the local carrier. If the incoming GPS signal is in-phase with the local carrier, all the energy will be in the I-arm but if the local carrier and incoming phase are drifting, the signal energy will alternate between the I-arm and Q-arm (Borre & Strang: 2012: xxxv).

Lastly, the outputs from the six correlators (x3 I-arm and x3 Q-arm) are combined via a tracking code loop discriminator and feedback to the PRN code generator, to properly adjust the code phase of the generated PRN code. The normalised early-minus-late power discriminator is suitable because it makes the code tracking loop independent of the carrier tracking loop, as it estimates correlation in both the (I) and (Q) branches of the code tracking loop (Shi *et al.*, 2009).

It is essential to note that the space between the E-P-L codes determines the noise bandwidth in the DLL. If the discriminator spacing is $>1/2$ chip, the DLL can handle wider dynamics and be more noise robust / vice versa. This is autonomously done in advanced GPS receiver correlator.

7.3.4 Combination of Carrier and Robust DLL Code Tracking Techniques

Sections 7.3.1 to 7.3.3 depict the details of the carrier and code tracking loops separately. However, the carrier and code tracking loops are usually combined to reduce computational load. They are often coupled as one, since the PRN code replica used to wipe-off the PRN code in the carrier tracking loop, comes from the code tracking loop. Similarly, the two local carrier replicas used to wipe-off the carrier frequency in the code tracking loop, comes from the carrier tracking loop. Figures 7.15 and 7.16 depict two different approaches for implementing the GPS signals carrier and code tracking methods. PLL maths analysis in correlation is covered in Appendix B.

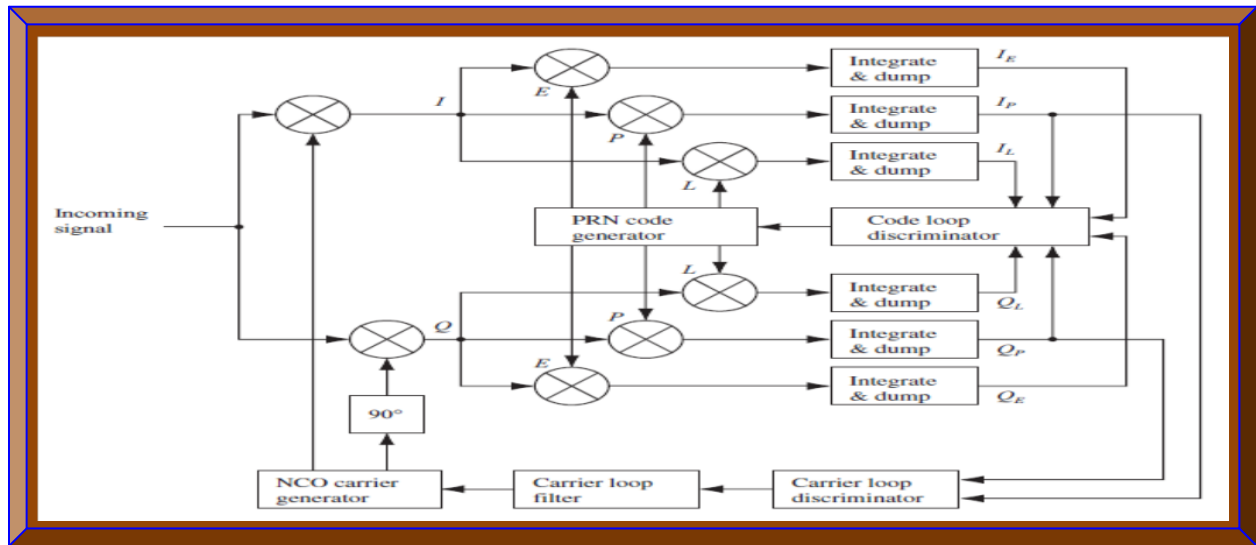


Figure 7.15: Combined carrier and robust DLL code tracking approach 1
(Adapted from Borre & Akos, 2005: 4)

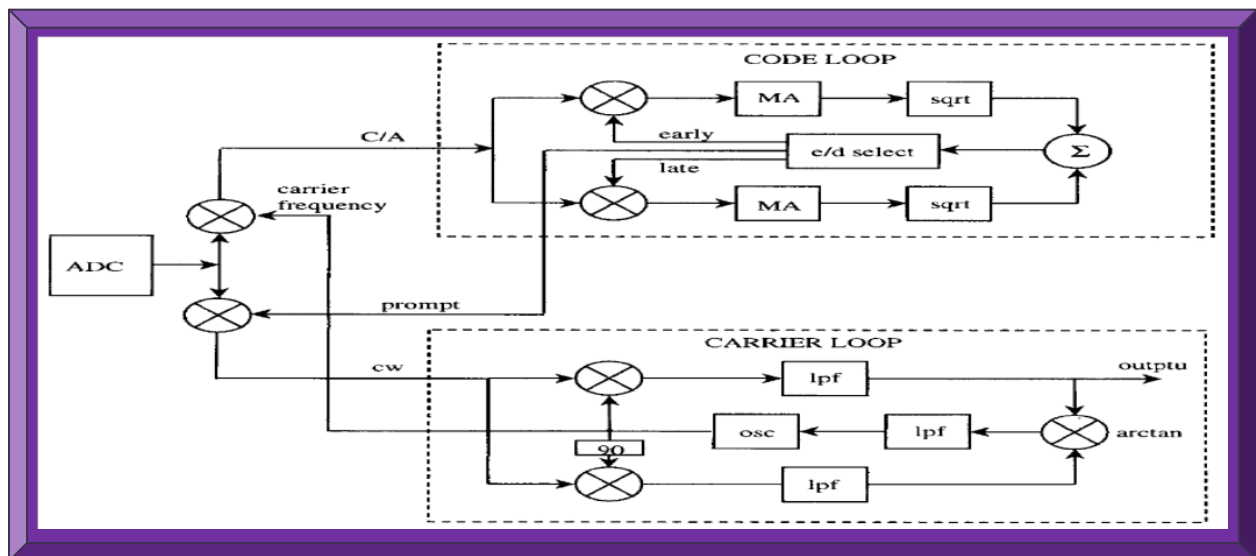


Figure 7.16: Combined carrier and robust DLL code tracking approach 2
(Adapted from Tsui, 2005: 168)

7.3.5 Block Adjustment of Synchronising Signals Tracking Technique

The tracking techniques discussed so far are conventional and requires adjusting certain parameters (phase, frequency and code) of the GPS signal. In the Block Adjustment of Synchronising Signals (BASS), the locally generated carrier frequency, is updated at most every 10ms, because the carrier frequency changes very slowly for a stationary receiver, contrary to the update rate of a dynamic GPS receiver, such as that on a CubeSat. The three outputs from the BASS technique are (i) the phase angle, obtained from the prompt code and is computed every 1ms (ii) the C/A code, which is used all the time once generated and (iii) the fine time resolution, that is obtained from the early and the late outputs of the code loop (Tsui, 2005: 170 - 175, 181).

The BASS technique uses the three tracking outputs of phase angle, the beginning of the C/A code and the fine time resolution to calculate the GPS receiver's position. The C/A code shift beginning can be checked with an update rate of 10ms, during which, the input data must be shifted one data point, either to the left or to the right to align the locally generated code, if the misalignment between the input data and the locally generated C/A code is in excess of 100ns. The actual starting points, which are referenced for every 10ms to the input data points of the received GPS C/A code, must be maintained, as these are used to find the beginning of the sub-frames when decoding the recovered GPS navigation data message (Tsui, 2005: 180 - 182).

The BASS technique discussed, is based-on tracking 1ms of C/A code and is recommended for tracking strong signals. To track weak GPS signal using BASS, more than 1ms of data must be tracked to improve sensitivity and the maximum data length that can be tracked coherently without using complicated processing, is 20ms, since GPS navigation data is 20ms (1/50 Hz) in length. Tracking of data > 20ms might encounter phase transitions, which will likely corrupt the already tracked data. This drawback can be mitigated by using multiple 20ms data and finding successive phase transitions in the process but at the expense of complexity (Tsui, 2005: 183).

7.3.6 Combination of Squared Correlators Tracking Technique

According to Falletti *et al.* (2012), a Combination of Squared Correlators (CSC) outputs, can be used as a DLL discrimination function for multiband GNSS receivers, to alleviate multipath effects in GPS signals tracking. Their approach can be defined in a coherent and non-coherent formulations and it consists of four correlators per channel plus the prompt correlator. Their technique can be compared with the double-delta architecture and the StrobeTM discriminator.

7.3.7 Nonlinear Code Tracking Filter Technique

To alleviate unintentional and intentional jammings, as a result of relatively low signal power at the GPS receiver antenna, due to sub-optimal code tracking loop designs which do not account for measurement non-linearities near loss-of-lock; Gustafon *et al.* (2009), developed a non-linear code tracking filter. Their architecture is based-on a thorough minimum-variance solution of the navigation problem, as opposed to using pre-specified tracking loop architectures. Their filter implementation can be viewed in terms of the classical notions of error detector functions, which depends on Signal-to-Noise Ratio (SNR) and Root Mean Square (RMS) code tracking error. Detector functions are defined for both code tracking error and code tracking error variance and due to a measurement-dependent term in the covariance calculations, their filter responds to rapidly varying jammer power more quickly than current designs. Extended-range tracking is utilised to produce linear state vector and error detection functions, with optimal response within the maximum limit of the correlator range, which as well lessens the need for re-acquisitions.

Pertaining to GPS code tracking, this technique utilised Gaussian moment approximation, in which, algorithms for approximating estimation of the conditional mean and covariance matrix of the state vector can be deduced if the measurement history is given. The ultimate goal is to estimate near loss of code lock and the outcome from implementing the technique. Gustafon *et al.* (2009), also predicted significant anti-jamming improvements relative to designs from high-fidelity simulations and hardware demonstrations. Furthermore, computational requirements in their technique are comparable to extended Kalman filter and vector tracking loop techniques.

7.3.8 Kalman Filter Tracking Loop Technique

Song *et al.* (2012), assessed the design of second-order, third-order, second-order IMU-assisted-based and Kalman tracking loop filters and deduced from the simulated results that, the Kalman tracking loop filter, is the best to accurately track dynamic signals as well as remove most of the interference in the incoming GPS signal. The GPS signal tracking loops can as well be optimised for corresponding improvement in noise. In Kalman filter, the discriminator output is essentially the filter observation and the tracking error is the filter state variables. Figure 7.17 illustrates this.

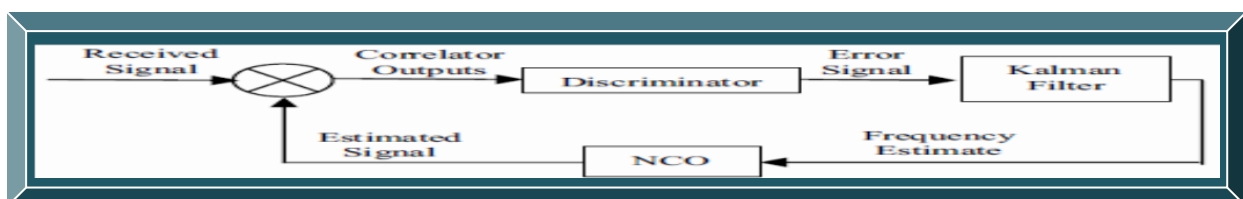


Figure 7.17: Basic illustration of a Kalman filter tracking loop concept

(Adapted from Song *et al.*, 2012)

7.3.9 Extended Kalman Filter Tracking Technique

In an effort to mitigate short-comings in conventional DLL, PLL and FLL tracking loops based-on the Scalar Tracking Loop (STL) principle, Peng *et al.* (2012), implemented a Vector Tracking Loop (VTL) based-on Extended Kalman Filter (EKF) with adaptive covariance matrices. The uniqueness in this approach is that, both the STL and VTL are implemented to utilise the results from the VTL to assist the STL, once an error in the scalar loop is detected during tracking.

In traditional STL, GPS signals from each satellite are process separately and the different channels tracking results are then merged to estimate the navigation solutions. The performance of STLs are limited by the filter coefficients, such as filter orders and noise bandwidths. Furthermore, STL ignores the inherent relationship between the navigation solutions and the tracking loop status. In this regard, a STL is more like an open loop system and yields poor performance when GPS signal outages occur; due to scintillation, interferences and Doppler. Conversely, a VTL provides a deep level of integration between signal tracking and navigation solutions in a GPS receiver; with the most prominent benefits being the increased interference immunity, robust dynamic performance, the ability to operate at low signal power and short signal outages. As shown in Figure 7.18, the PVT states of the receiver are used as the state vector for the Kalman filter. The code phase can be guessed by the distance between the SV and the receiver, while the error in the pseudorange measurement can be modeled as a Gaussian distribution and the Doppler can be approximated by the GPS receiver and SVs relative motion. To validate the result, the performance of VTL is compared with the traditional STL on three different data sets: (i) GPS data with strong scintillation impacts gathered in the previous solar maximum (ii) raw GPS RF data with short signal outages and (iii) high-dynamics GPS data with long interval signal outages from a GPS simulator. The results validated the performance improvement of the VTL over scintillation impacts and showed that, with the availability of GPS satellites ephemerides, the VTL can maintain signal lock during prolong intervals of GPS signal unavailabilities and the pseudo-range approximation can be within one PRN code chip accuracy.

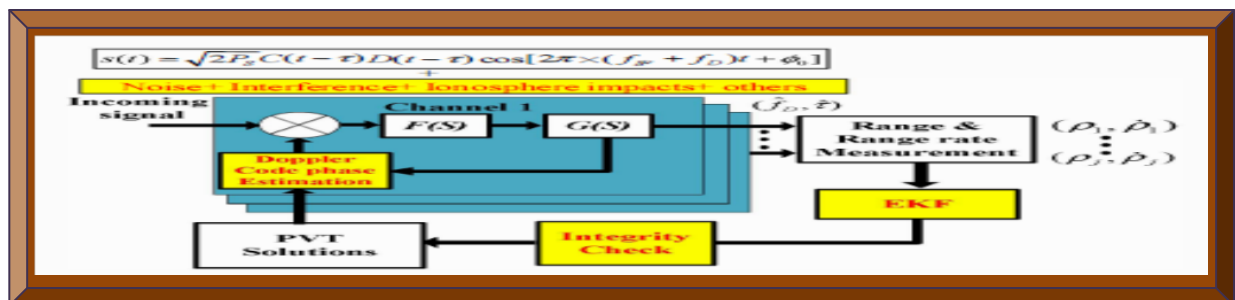


Figure 7.18: Illustration of a VTL assisting tracking loop
(Adapted from Peng *et al.*, 2012)

7.3.10 Inertial Navigation System Tracking Technique

As an alternative approach to the problem of the GPS receiver tracking loop easily losing lock in high-dynamics environment, Xiaoyong and Baiqi (2011), demonstrated an Inertial Navigation System (INS) tracking loop. In this, the velocity of the GPS receiver is used to advance the tracking loop, improve anti-jamming capability and to elevate performance of the GPS receiver.

This scheme is shown in Figure 7.19. It is similar to the conventional GPS code tracking technique in that, a DLL is used to track the C/A code and a Costas PLL to track the carrier phase. PLL is used to aid and reinforce the DLL. As a result, INS velocity aiding will complement and/or rescue the Costas PLL, if it loses lock due to extreme Doppler shifts in high-dynamics condition.

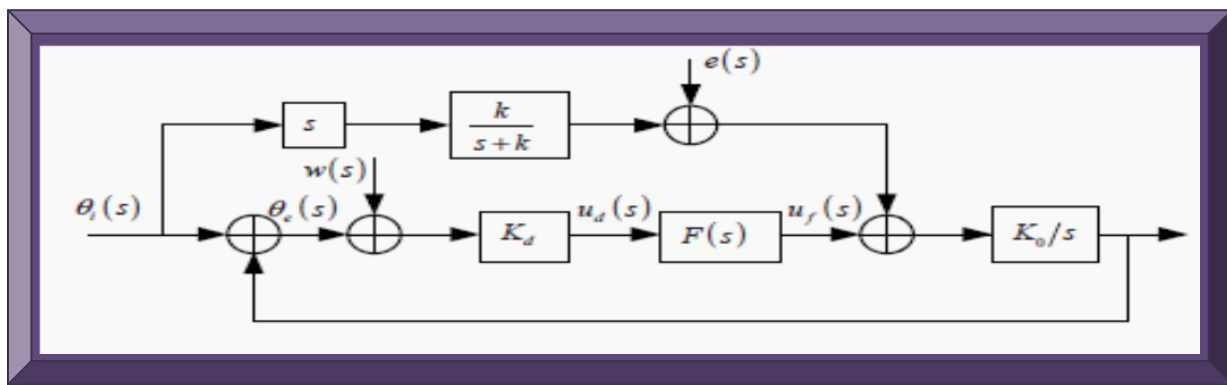


Figure 7.19: INS aided loop architecture illustration
(Adapted from Xiaoyong & Baiqi, 2011)

7.4 Summary

This section briefly examined various techniques on how to design and implement a GPS L1 C/A correlator. The purpose is to analyse and select the most suitable acquisition and tracking techniques; in terms of efficiency, robustness and the ease to implement and develop, in-line with the research objectives. From the analysis, the acquisition techniques chosen are the parallel code phase search and the block or non-coherent integration, while the tracking techniques selected, are the combined carrier and robust code tracking, as well as the BASS techniques.

To conclude, the GPS signal acquisition and tracking techniques presented and listed in Tables 3.4 - 3.6, are most of the various known distinct ways of designing a reliable GPS receiver correlator. Additionally, there are other factors such as data size, execution time, performance estimation, damping ratio and noise bandwidth that must be traded-off or taken into account.

APPENDIX B

GPS APPLICABLE MATHEMATICAL ANALYSIS

8.1 Introduction

Appendix B focuses on GPS mathematical aspects relevant to the research, as well as the theory to support the reasoning. It systematically starts with the mathematical analysis of the transmitted GPS L1 C/A signal, followed by acquisition, then tracking and finally the processing of PVT.

8.2 GPS L1 C/A Signal Analysis

As detailed in Section 2.5, a GPS signal is composed of a carrier frequency, PRN codes and a navigation data message which are blended together by DSSS - CDMA - BPSK schemes. In Borre and Strang (2012: xxvii), the transmitted GPS signal is mathematically represented as:

$$\begin{aligned}
 s^k(t) &= \sqrt{2P_C} \left(C^k(t) \oplus D^k(t) \right) \cos(2\pi f_{L1}t) && \text{GPS L1 C/A Signal} \\
 &+ \sqrt{2P_{PL1}} \left(P^k(t) \oplus D^k(t) \right) \sin(2\pi f_{L1}t) && \text{GPS L1 P(Y) Signal} \\
 &+ \sqrt{2P_{PL2}} \left(P^k(t) \oplus D^k(t) \right) \sin(2\pi f_{L2}t) && \text{GPS L2 P(Y) Signal}
 \end{aligned}
 \quad \left. \vphantom{\begin{aligned} s^k(t) \\ + \\ + \end{aligned}} \right\} \text{Equation 8.1}$$

where:

$s^k(t)$ is the composite signal from a single GPS satellite k ($k = 1$ to 32 different GPS satellites)

P_C, P_{PL1} and P_{PL2} are the respective amplitudes or powers of signals with C/A or P(Y) codes

C^k and P^k are respectively the C/A and P(Y) code sequences assigned to satellite number k

D^k is the navigation data sequence associated with any of the 32 GPS satellites

f_{L1} and f_{L2} are the corresponding carrier frequencies of GPS L1 and L2 signals

t is instant of time

\cos and \sin are the in-phase and quadrature components associated with respective GPS signals

\oplus denotes modulus two addition (equivalent to X-OR operation if bits are of type 1 and -1)

Refer to Table 2.3 for details on modulo-2 addition and its application to BPSK in GPS.

8.2.1 GPS L1 Carrier Frequency Analysis

This research focuses on the GPS L1 carrier frequency. It has the following properties:

- Carrier frequency is 1.57542 GHz (i.e. $L1 = f_0 \times 154 = 10.23 \text{ MHz} \times 154 = 1575.42 \text{ MHz}$).
- Wavelength is $\sim 19.04 \text{ cm}$ ($\lambda = c/f$); where f is the carrier frequency and c the speed of light.
- Constitute Coarse Acquisition (C/A) and Precision (P) Codes – both phase shifted by 90° .
- The transmitted GPS L1 signal has an EIRP level of $\sim 27 \text{ dBW}$ towards Earth.
- L1 signal modulated by C/A-code has a received signal level of approximately -130 dBm .
- L1 signal modulated by P(Y)-code has a received signal level of approximately -132 dBm .
- L1 bandwidth is $\sim 20 \text{ MHz}$ (20.46 MHz to be exact) to accommodate the P(Y)-code.

8.2.2 GPS C/A – Code Analysis

This research focuses on the GPS C/A code. It has the following technical attributes:

- Chipping frequency is 1.023 MHz (i.e. $0.1f_0 = 0.1 \times 10.23 \text{ MHz}$).
- Chip length is 977.5 ns (i.e. $1/1.023 \text{ MHz}$) or wavelength of $\sim 293.255 \text{ m}$ (from $\lambda = c/f$).
- 1023 bits (chips) in length as generated by its PRN polynomial.
- Modulates L1 signal only and at a clock or chipping rate of $1.023 \text{ Mega chips/s}$.
- Repeats every 1 ms or has a 1 ms duration (i.e. $1023 \text{ chips}/1.023 \text{ MHz} = 1 \text{ ms}$).
- Bandwidth is 2.046 MHz (i.e. $2 \times L1_{\text{caf}} = 2 \times 1.023 \text{ MHz}$); $L1_{\text{caf}}$ is C/A chipping frequency.
- Has 37 unique Gold codes (practically 36, since Gold codes 34 and 37 are the same).

Theoretical details on GPS C/A operation and generation are discussed in Section 2.5.2.3.

GPS C/A 1023 codes lengths are generated based on two 10 bits LFSR polynomials G1 and G2.

$$\left. \begin{aligned} G1(X) &= 1 + X^3 + X^{10} \\ G2(X) &= 1 + X^2 + X^3 + X^6 + X^8 + X^9 + X^{10} \end{aligned} \right\} \text{Equation 8.2}$$

As stated in Borre *et al.* (2007: 19), the auto-correlation function for the C/A code is given as:

$$r_p(\tau) = \frac{1}{NT_c} \int_0^{NT_c} p(t)p(t+\tau)dt \quad \text{Equation 8.3}$$

An alternative representation of the auto-correlation function in Johanssen *et al.* (1998: 6) is:

$$CA_{ii}(\tau) = \int_{-\infty}^{\infty} CA_i(t)CA_i(t+\tau)dt \quad \text{Equation 8.4}$$

where $r_p(\tau) = CA_{ii}(\tau)$ are the auto-correlation functions, $p = CA_i$ is the C/A code for the i^{th} satellite and τ is the phase of the time shift (t).

In a similar reasoning, the cross-correlation function is given as:

$$CA_{ij}(\tau) = \int_{-\infty}^{\infty} CA_i(t)CA_j(t + \tau)dt \quad \text{Equation 8.5}$$

where CA_i and CA_j are the respective C/A codes for the i^{th} and j^{th} satellites.

Generally, GPS C/A codes are often referred to as Gold codes and are chosen for GPS signals because of their unique auto and cross-correlation properties, which are summarised as follows:

- There is nearly no cross-correlation between the same GPS C/A codes (e.g. CA_i and CA_i).

Thus, there is always cross-correlation between different GPS C/A codes (e.g. CA_i and CA_j).

$$CA_{ij}(\tau) = \sum_{t=0}^{1022} CA_i(t)CA_j(t + \tau) \approx 0 \quad \text{for all } \tau \quad \text{Equation 8.6}$$

- There is nearly no auto-correlation (except for zero lag) between different GPS C/A codes (e.g. CA_i and CA_j). Meaning, there is always auto-correlation between the same GPS C/A codes (e.g. CA_i and CA_i). It should be noted that, any two different GPS C/A codes are nearly orthogonal.

$$CA_{ii}(\tau) = \sum_{t=0}^{1022} CA_i(t)CA_i(t + \tau) \approx 0 \quad \text{for all } |\tau| \geq 1 \quad \text{Equation 8.7}$$

These two properties are very useful during acquiring, tracking and decoding of GPS signals.

Furthermore, the auto-correlation functions for $n = 10$; will have a peak of magnitude 1023:

$$CA_{ii,peak} = 2^n - 1 = 1023 \quad \text{Equation 8.8}$$

Similarly, the auto-correlation functions for $n = 10$; will have a magnitude of ≤ 65 .

$$|CA_{ii}| \leq 2^{0.5(n+2)} + 1 \leq 65 \quad \text{Equation 8.9}$$

where n is the maximum number of states in each shift register (G1 and G2 of Equation 8.2).

The cross-correlation function also satisfies Equation 8.9.

As indicated in Johanssen *et al.* (1998: 7), the cross-correlation properties of C/A codes are not ideal and may under certain conditions, cause false acquisition, especially in scenarios where multiple GPS satellites signals are received with different delays and very high Doppler offsets.

8.2.3 GPS Navigation Data Message Analysis

This is briefly discussed in detailed in Section 2.5.3. The complete technical and mathematical details on decoding and processing of GPS L1 C/A signal, is beyond the scope of this thesis. For further reading, refer to the Navstar IS-GPS-200H (2013) specification.

8.2.4 GPS L1 C/A Signal Acquisition Analysis

Various acquisition techniques were investigated and summarised in Tables 3.4 and 3.5. In serial search acquisition method, the maximum number of search combinations is deduced as follows:

$$\text{Code phase} \times \left(2 \times \left\lceil \frac{\text{Doppler}}{\text{Search Step}} \right\rceil + 1 \right) \quad \text{Equation 8.10}$$

$$1023 \times \left(2 \times \left\lceil \frac{10000}{500} \right\rceil + 1 \right) = 1023 \times 41 = 41\,943 \text{ search combinations or bins}$$

Johanssen *et al.* (1998: 9), states that, the 500 Hz search step is a trade-off in speed and accuracy. In parallel frequency space search, only the 1023 different code phases are searched, whereas in parallel code phase search, only the number of carrier frequencies is searched. From these three acquisition techniques, the parallel code phase search (circular cross-correlation) is the most cumbersome but based-on its execution time and number of search bins, it will be used as the preferred acquisition method for the space-borne GPS receiver. Depending on the Doppler shift, this technique searches for the number of carrier frequencies in parallel with the 1023 different code phases. Table 8.1 compares these three techniques according to Borre *et al.* (2007: 85).

Table 8.1: Performance comparison of three common acquisition techniques
(Adapted from Borre *et al.*, 2007: 85)

Algorithm	Execution time	Repetitions	Complexity
Serial search	87	41,943	Low
Parallel frequency space search	10	1023	Medium
Parallel code phase search	1	41	High

These 41 number of repetitions within the parallel code phase search is based-on a ± 10 kHz Doppler shift, which is usually the maximum consideration for a terrestrial GPS receiver. For a space-borne GPS receiver, the Doppler frequency can be in excess of ± 25 , ± 50 and ± 100 kHz; which respectively translates to repetitions (search bins or combinations) of 101, 201 and 401.

That is, for a ± 25 , ± 50 and ± 100 kHz Doppler, the number of search bins is: $2 \left\lceil \frac{\text{Doppler}}{500} \right\rceil + 1$.

8.2.4.1 Parallel Code Phase Search (Circular Cross-Correlation) Acquisition Technique Analysis

This section examines the mathematical details. In Tsui (2005: 137 - 141), the discrete Fourier transforms of two finite length sequences $x(n)$ and $y(n)$, both with length N are computed as:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi kn}{N}} \quad \text{and} \quad Y(k) = \sum_{n=0}^{N-1} y(n) e^{-\frac{j2\pi kn}{N}} \quad \text{Equation 8.11}$$

From which the circular cross-correlation (not convolution) sequence between these two finite length sequences $x(n)$ and $y(n)$ both with length N and with periodic repetition, is computed as:

$$z(n) = \frac{1}{N} \sum_{m=0}^{N-1} x(m) y(m+n) = \frac{1}{N} \sum_{m=0}^{N-1} x(-m) y(m-n) \quad \text{Equation 8.12}$$

The discrete N -point Fourier transform of $z(n)$ with N^l omitted, can therefore be expressed as:

$$Z(k) = \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} x(-m) y(m-n) e^{-\frac{j2\pi kn}{N}} \quad \text{Equation 8.13}$$

$$Z(k) = \sum_{m=0}^{N-1} x(m) e^{\frac{j2\pi km}{N}} \sum_{n=0}^{N-1} y(m+n) e^{-\frac{j2\pi k(m+n)}{N}} \quad \text{Equation 8.14}$$

$$Z(k) = X^*(k)Y(k) \quad \text{Equation 8.15}$$

where $X^*(k)$ is the complex conjugate of $X(k)$

To summarise, the procedure for searching the number of carrier frequencies is as follows:

- 1) The incoming signal is multiplied with a cosine or sine signal to give the in-phase or quadrature signal components.
- 2) The in-phase and quadrature signal components are combined to produce a complex input.
- 3) The produced complex input signal is transformed by FFT.
- 4) Local replica of the C/A code for each respective satellite is then generated.
- 5) FFT is applied to these codes, complex conjugated and multiplied with step 3 output.
- 6) IFFT is then performed and the absolute values of all the signal components are computed.
- 7) Maximum value if satellite K is present will be found at code phase τ and frequency $f - \Delta f$.
- 8) Otherwise, if no distinct maximum value is found, the search procedure is iterated.

The parallel code phase search technique is chosen as the preferred algorithm in researching the GPS receiver. See the source-code in the Matlab file, *acquisition.m*, for the full implementation.

8.2.5 GPS L1 C/A Signal Tracking Analysis

The basic goal of acquisition is to de-spread the incoming signal and pass the carrier frequency and code phase (usually both have Doppler shifts or offsets) to the tracking section. The tracking section then continuously tracks both the carrier frequencies and code phases to achieve and to maintain lock. The carrier frequency is tracked using FLL (which tracks the frequency of the carrier) and or PLL (which tracks the phase of the carrier). The code phase is tracked using DLL.

The theory behind these techniques is covered in Appendix A and the mathematics is as follows:

8.2.5.1 PLL Analysis

The starting point in this thesis is from a second-order PLL that often contains a first-order filter and a Voltage Controlled Oscillator (VCO), with transfer functions respectively expressed as:

$$F(s) = \frac{1}{s} \frac{\tau_2 s + 1}{\tau_1} \quad \text{Equation 8.16}$$

$$N(s) = \frac{K_0}{s} \quad \text{Equation 8.17}$$

$$H(s) = \frac{k_d F(s) N(s)}{1 + k_d F(s) N(s)} \quad \text{Equation 8.18}$$

where $F(s)$ and $N(s)$ are the respective transfer functions of the filter and VCO, K_0 the VCO gain, τ_1 and τ_2 the loop filter coefficients, $H(s)$ the transfer function of a linearised analog PLL and k_d the phase discriminator gain. Substituting Equations 8.16 and 8.17 into Equation 8.18 yields:

$$H(s) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + s\zeta\omega_n s + \omega_n^2} \quad \text{Equation 8.19}$$

where the natural frequency (ω_n): $\omega_n = \sqrt{(k_0 k_d) / \tau_1}$ Equation 8.20

and the damping ratio (ζ): $\zeta = 0.5\tau_2\omega_n$ Equation 8.21

According to Chung *et al.* (1993: 3 & 4), the above transfer functions are analog versions and to convert these transfer functions to digital forms, the bilinear transformation is used on Equation 8.19 to yield the following equivalent digital transfer function $H_1(z)$, for the PLL model as:

$$H_1(z) = \frac{(4\zeta\omega_n T + (\omega_n T)^2) + 2(\omega_n T)^2 z^{-1} + ((\omega_n T)^2 - 4\zeta\omega_n T) z^{-2}}{(4 + 4\zeta\omega_n T + (\omega_n T)^2) + (2(\omega_n T)^2 - 8) z^{-1} + (4 - 4\zeta\omega_n T + (\omega_n T)^2) z^{-2}} \quad \text{Equation 8.22}$$

In a similar reasoning in Chung *et al.* (1993: 3 & 4), the linearised digital second-order PLL transfer functions are deduced as:

$$F(z) = c_1 + \frac{c_2}{1-z^{-1}} \quad \text{Equation 8.23}$$

$$N(z) = \frac{k_0 z^{-1}}{1-z^{-1}} \quad \text{Equation 8.24}$$

where K_0 is the discriminator gain, $F(z)$ is the transfer function of the filter and $N(z)$ is the transfer function of now the Numerically Control Oscillator (NCO), contrary to the analog VCO.

The rationale is to obtain the coefficients C_1 and C_2 in the second-order PLL. This is deduced by comparing the transfer function for the digital and analog PLLs. The transfer function for the digital version can be found using:

$$H(z) = \frac{\theta_o(z)}{\theta_i(z)} = \frac{K_d F(z) N(z)}{1 + K_d F(z) N(z)} \quad \text{Equation 8.25}$$

Substituting Equations 8.23 and 8.24 into Equation 8.25, gives the new transfer function:

$$H_2(z) = \frac{K_0 K_d (C_1 + C_2) z^{-1} - K_0 K_d C_1 z^{-2}}{1 + [K_0 K_d (C_1 + C_2) - 2] z^{-1} + (1 - K_0 K_d C_1) z^{-2}} \quad \text{Equation 8.26}$$

As a result, the equations for the two coefficients C_1 and C_2 are established by comparing Equations 8.22 and 8.26 to give:

$$C_1 = \frac{1}{K_0 K_d} \frac{8\zeta\omega_n T}{4 + 4\zeta\omega_n T + (\omega_n T)^2} \quad \text{Equation 8.27}$$

$$C_2 = \frac{1}{K_0 K_d} \frac{4(\omega_n T)^2}{4 + 4\zeta\omega_n T + (\omega_n T)^2} \quad \text{Equation 8.28}$$

where $K_0 K_d$ is the loop gain, ζ is the damping ratio (controls how fast the filter reaches its settling point and the degree of overshoot allowed), ω_n is the natural frequency and T the sampling time.

Lastly, the natural frequency (ω_n) is deduced from:

$$\omega_n = \frac{8\zeta B_L}{4\zeta^2 + 1} \quad \text{Equation 8.29}$$

where B_L is the noise bandwidth (controls the amount of noise allowed) in the filter loop.

8.2.5.2 Carrier Tracking Analysis

Usually, the phase error or a function of the phase error is the output of the discriminator. This is filtered and utilised as a feedback to the NCO, to adjust the frequency of the local carrier wave. In this way, the local carrier frequency could be an almost precise copy of the incoming carrier.

In Kaplan and Hegarty (2006: 164 - 170), an ordinary PLL is sensitive to 180° phase shift which makes it susceptible to navigation data bits transition after every 50 Hz or 20ms cycle. A robust PLL, known as Costas loop is therefore used. Its 180° phase shift in-sensitivity, mitigates the GPS data navigation phase transitions. The goal of Costa loop is to conserve all the energy in the in-phase (I)-arm of the carrier frequency. Borre and Strang (2012: xxxii) analyse PLL as follows:

If the code replica is perfectly aligned, the multiplication in the I-arm yields the following:

$$D^k(n) \cos(\omega_{IF}n) \cos(\omega_{IF}n + \varphi) = 0.5 D^k(n)[\cos(\varphi) + \cos(2\omega_{IF}n + \varphi)] \quad \text{Equation 8.30}$$

where φ is the phase difference between the phase of the input signal and the phase of the local replica of the carrier phase. The multiplication in the quadrature (Q)-arm yields the following:

$$D^k(n) \cos(\omega_{IF}n) \sin(\omega_{IF}n + \varphi) = 0.5 D^k(n)[\sin(\varphi) + \sin(2\omega_{IF}n + \varphi)] \quad \text{Equation 8.31}$$

Low pass filtering the resultant two signals remove the double IF terms to give the followings:

$$I^k = 0.5D^k(n) \cos(\varphi) \quad \text{Equation 8.32}$$

$$Q^k = 0.5D^k(n) \sin(\varphi) \quad \text{Equation 8.33}$$

The phase error of the local carrier phase replica is then computed by dividing Equation 8.33 with Equation 8.32 to give Equation 8.34. The result is feedback to the carrier phase oscillator.

$$\frac{Q^k}{I^k} = \frac{0.5D^k(n) \sin(\varphi)}{0.5D^k(n) \cos(\varphi)} = \tan(\varphi) \quad \text{Equation 8.34}$$

$$\varphi = \arctan\left(\frac{Q^k}{I^k}\right) \quad \text{Equation 8.35}$$

From Equation 8.35, the phase error (φ) is minimised when the correlation value in the Q-phase arm is zero and the correlation value in the I-phase arm is maximum. Although this arctan discriminator is very precise, it is very time consuming and often requires an improved version.

8.2.5.3 Code Tracking Analysis

Code tracking loop objective is to keep track of the code phase of a specific code in the signal. The output is usually a perfectly aligned replica of the code. The code tracking loop in the GPS receiver is a DLL, called an early-late tracking loop and a robust implementation is based-on the non-coherent discriminator highlighted in red in Table 8.2. The rationale behind the DLL is to correlate the input signal with three replicas of the code. Table 8.2 summarises the various types.

Table 8.2: Various types of DLL discriminators
(Adapted from Borre & Strang, 2012: xxxvi)

Type	Discriminator D	Characteristics
Coherent	$I_E - I_L$	Simplest of all discriminators. Does not require the Q branch but requires a good carrier tracking loop for optimal functionality.
	$(I_E^2 + Q_E^2) - (I_L^2 + Q_L^2)$	Early minus late power. The discriminator response is nearly the same as the coherent discriminator inside $\pm \frac{1}{2}$ chip.
Noncoherent	$\frac{(I_E^2 + Q_E^2) - (I_L^2 + Q_L^2)}{(I_E^2 + Q_E^2) + (I_L^2 + Q_L^2)}$	Normalized early minus late power. The discriminator has a great property when the chip error is larger than a $\frac{1}{2}$ chip; this will help the DLL to keep track in noisy signals.
	$I_P(I_E - I_L) + Q_P(Q_E - Q_L)$	Dot product. This is the only DLL discriminator that uses all six correlator outputs.

According to Tsui (2005: 169), the procedure applied to both the carrier and code loops to track an acquired GPS L1 C/A signal can be abridged as follows:

- Define the gain of the carrier and code loops, as well as the bandwidths. The loop gain comprises the gains of the NCO and the phase detector. The bandwidth of the carrier loop is often wider than the code loop because it tracks the locked GPS signal for a longer duration.
- Select the carrier loop to be 20 Hz and the noise bandwidth of the code loop to be 1 Hz. The choice is subjective and takes a set of potentials values suitable for the tracking algorithm.
- Choose the damping factor $\zeta = 0.707$. This value of ζ is usually viewed as close to optimal.
- Calculate the natural frequency (ω_n) from Equation 8.29.
- The carrier loop gain is chosen to be $4\pi \times 100$ and the code loop gain (K_0K_1) to be 50. The choice of these values depends on a set of several viable selections. Equations 8.27 and 8.28 are used to compute the filter constants C_1 and C_2 . Once all the values are known, the carrier and code phase loops can be automatically adjusted to follow the input signals every 1ms.

8.2.6 GPS Navigation Data Message Decoding Analysis

In Borre *et al.* (2007: 88), the down-converted and filtered signal $S^k(n)$ of a satellite k (where ω_{IF} is the intermediate angular frequency), from a GPS receiver front-end can be represented as:

$$s^k(t) = \sqrt{2P_C}(C^k(t)D^k(t) \cos(\omega_{IF}t) + \sqrt{2P_{PL1}}(P^k(t)D^k(t) \sin(\omega_{IF}t) \quad \text{Equation 8.36}$$

Similarly, this GPS signal from satellite k after the ADC stage can as well be represented as:

$$s^k(n) = C^k(n)D^k(n) \cos(\omega_{IF}n) + e(n) \quad \text{Equation 8.37}$$

where n is a discrete time signal in unit of $1/f_s$ samples and $e(n)$ the distorted P(Y) code or noise.

This down-converted signal $S^k(n)$, is acquired and tracked to give the navigation signal $D^k(n)$. Once the tracking loop is locked and aligned, the baseband 50Hz (bps) or 20ms navigation message can then be recovered and decoded. Usually, the outcome from the tracking loop is the value of the in-phase arm of the tracking block reduced to the values 1 and -1. Therefore, once the GPS navigation message data bits transition times are located from the tracking block output, they can be sampled at 1000sps (corresponding to a value every 1ms). This tracked GPS signal must then be converted from 1000bps to 50bps before the navigation data can be decoded. That is, 20 consecutive sampled values must be replaced by 1 only, as a GPS navigation data is 20ms. This conversion procedure is termed bit synchronisation. See Section 3.5.3 and Navstar IS-GPS-200H (2013) for details. Figure 8.1 represents a typical recovered navigation data message, with the arrows marking the bit transitions 20ms apart. This navigation data signal is very strong, since a weaker signal will have the dots closer to zero and clustered together without separations.

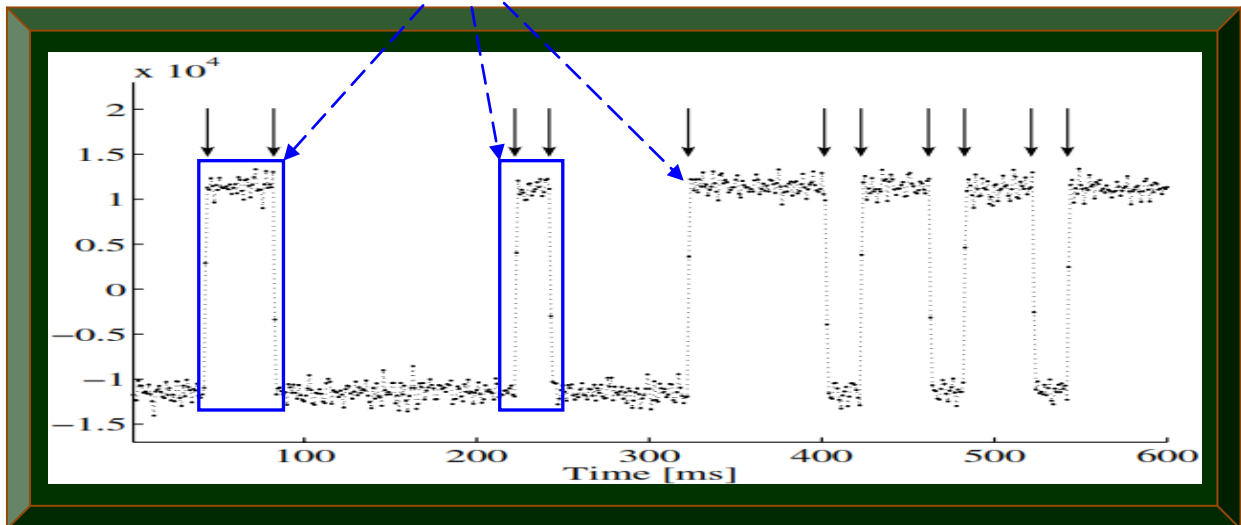


Figure 8.1: Tracking block output – depicting recovered GPS navigation message
(Adapted from Borre *et al.*, 2007: 110)

8.2.7 GPS Navigation Data Message Processing Analysis

From successful decoding, the navigation data is ready for processing. Refer to Section 3.5.3.2.

8.2.7.1 GPS Satellite Position Computation Analysis

Once the GPS navigation message is decoded and the satellites ephemeris orbital data extracted, the first step is to compute the satellites positions. The equation describing the position vector r , of the satellites with respect to the centre of the Earth in Borre and Strang (2012: 267 & 268) is:

$$r = \begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix} = \begin{bmatrix} a(\cos E - e) \\ a\sqrt{1 - e^2} \sin E \\ 0 \end{bmatrix} \quad \text{Equation 8.38}$$

a = semi major axis; e = the eccentricity; E = the eccentric anomaly

ξ = represents the axis pointing to the perigee

η = represents the axis towards the descending node

ζ = represents the axis perpendicular to the orbital plane

The norm of the position vector which is the geometric distance between the orbiting GPS satellites and the centre of the Earth can be calculated from simple geometry to be:

$$\|r\| = a(1 - e \cos E) \quad \text{Equation 8.39}$$

The mean motion or movement n , which is the mean angular satellite velocity, is computed as:

$$n = \frac{2\pi}{T} = \sqrt{\frac{\mu}{a^3}} = \sqrt{\frac{GM}{a^3}} + \Delta n \quad \text{Equation 8.40}$$

where T is the period of one revolution of the satellite, Δn is the mean motion difference obtained from sub-frame 2 in bits 91 to 106 and the product GM or μ is the Earth's universal gravitational constant with value $3.986005 \times 10^{14} \text{ m}^3/\text{s}^2$. A more accurate value of μ is obtained from the decoded broadcasted ephemeris data and is used in the satellites position calculations.

The true anomaly f , the eccentric anomaly E and the mean anomaly μ are related by the equation:

$$f = \arctan \frac{\eta}{\xi} = \arctan \frac{\sqrt{1-e^2} \sin E}{\cos E - e} \quad \text{where } E = \mu + e \sin E \quad \text{Equation 8.41}$$

Equation 8.41 is fundamental in performing calculations for the orbiting GPS satellites positions.

In GPS orbital plane, the rectangular or Cartesian coordinates of GPS satellites position is:

$$\begin{bmatrix} r_j^k \cos f_j^k \\ r_j^k \sin f_j^k \\ 0 \end{bmatrix} \quad \text{Equation 8.42}$$

where $r_j^k = \|r(t_j)\|$ are derived from Equation 8.39 with a , e and E evaluated for $t = t_j$.

8.2.7.2 Transformation of GPS Satellites Position Orbital Frame to ECEF Coordinate System

GPS satellites orbit is referenced to the centre of the Earth but in order to find a user position, the received GPS navigation data must be related to a certain point on or above the surface of the Earth. From using Direction Cosine Matrix (DCM), the GPS satellites position vector (r_j^k) is then rotated into the XYZ (ECEF) coordinate system by the following sequence of 3D rotations:

$$R_3(-\Omega_j^k) R_1(-i_j^k) R_3(-\omega_j^k) \quad \text{Equation 8.43}$$

For a rotation about the Z-axis:

$$R_3(\varphi) = \begin{bmatrix} \cos \varphi & \sin \varphi & 0 \\ -\sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Equation 8.44}$$

Similarly for a rotation about the X-axis

$$R_1(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & \sin \varphi \\ 0 & -\sin \varphi & \cos \varphi \end{bmatrix} \quad \text{Equation 8.45}$$

Finally, the geocentric coordinate $[X^k(t_j), Y^k(t_j), Z^k(t_j)]$ of satellite k at time t_j is computed as:

$$\begin{bmatrix} X^k(t_j) \\ Y^k(t_j) \\ Z^k(t_j) \end{bmatrix} = R_3(-\Omega_j^k) R_1(-i_j^k) R_3(-\omega_j^k) \begin{bmatrix} r_j^k \cos f_j^k \\ r_j^k \sin f_j^k \\ 0 \end{bmatrix} \quad \text{Equation 8.46}$$

In Strang and Borre (1997: 484), GPS satellites do not adhere to the orbital mechanics justified thus far, as more accurate time-dependent orbital values that are broadcasted in the ephemerides of the decoded navigation message should be used. Given GPS satellite signal transmit time t (in GPS time), the below parameters are used in Equation 8.46 to compute GPS satellites positions.

Time elapsed since t_{oe} : $t_j = t - t_{oe}$

Mean anomaly at time t_j : $\mu_j = \mu_0 + \left(\sqrt{\frac{GM}{a^3}} + \Delta n \right) t_j$

Iterative solution for E_j : $E_j = \mu_j + e \sin E_j$

True anomaly (f_j): $f_j = \arctan \frac{\sqrt{1-e^2} \sin E_j}{\cos E_j - e}$; ω_e is the mean Earth rotation

Longitude of ascending node (Ω_j): $\Omega_j = \Omega_0 + (\dot{\Omega} - \omega_e) t_j - \omega_e t_{oe}$; $\omega_e = 7.2921151467e^{-5} \text{rad/s}$

Argument of perigee (ω_j): $\omega_j = \omega + f_j + C_{\omega c} \cos 2(\omega + f_j) + C_{\omega s} \sin 2(\omega + f_j)$

Radial distance (r_j): $r_j = a(1 - e \cos E_j) + C_{rc} \cos 2(\omega + f_j) + C_{rs} \sin 2(\omega + f_j)$

Inclination (i_j): $i_j = i_0 + it_j + C_{ic} \cos 2(\omega + f_j) + C_{is} \sin 2(\omega + f_j)$

8.2.7.3 Pseudo-range Estimation Analysis

The measured distances between orbiting GPS satellites and the receiver is not the true range due to a common bias called pseudo-ranges. According to Borre *et al.* (2007: 119), estimating pseudo-ranges can be accomplished in two computational steps. Firstly, the initial set of pseudo-ranges is first computed followed by keeping track of all the pseudo-ranges in the second step. Refer to Section 3.5.3.2.2 and Navstar IS-GPS-200H (2013) specification for applicable details.

In its simplest form, the equation describing pseudo-range is represented as:

$$p_i^k = p_i^k + c\Delta t_i + \text{noise} \quad \text{Equation 8.47}$$

where p_i^k is the pseudo-range of satellite k and $c\Delta t_i$ is the bias (error) causing the pseudo-range.

Expanding on Equation 8.47, the pseudo-range on L1 including the known sources of errors is:

$$p_i^k = R_i^k + c(\Delta t_i - \Delta t^k) + T_{p_i^k} + I_{p_i^k} + M_{p_i^k} + \varepsilon_{p_i^k} \quad \text{Equation 8.48}$$

- p_i^k = pseudo-range measured on L1 frequency between satellite k and receiver i
- R_i^k = geometric range measured on L1 frequency between satellite k and receiver i
- c = speed of light
- Δt_i = user or receiver clock error
- Δt^k = satellite clock error
- $T_{p_i^k}$ = tropospheric delay, $I_{p_i^k}$ = ionospheric delay, $M_{p_i^k}$ = multipath delay
- $\varepsilon_{p_i^k}$ = combined miscellaneous sources of errors

8.2.7.4 GPS Time Correction Analysis

Tsui (2005: 62) indicates that, all GPS satellites transmit signals at the same time but the clock errors in each satellite and receiver and the associated delays creates pseudo-ranges between time of transmission t^k and arrival time t_u at the receiver. The equation describing this relationship is:

$$p_i^k = c(t_u - t^k) \quad \text{Equation 8.49}$$

To correct t^k , the value of t^k must first be known which makes the correction process difficult. To facilitate the process, t_c (obtained from TOW in the decoded navigation message) is considered to represent the coarse GPS system time at time of transmission, corrected by the transit time. The actual total time difference between the time t_c and the epoch time t_{oe} , will be the time t^k and should account for the end or beginning of the week crossovers (i.e. Saturday breaking Sunday).

$$\begin{aligned} \text{If } t^k = t_c - t_{oe} > 302400 \quad \text{then } t^k = t^k - 604800 \quad \text{or } t_c = t_c - 604800 \\ \text{If } t^k = t_c - t_{oe} < -302400 \quad \text{then } t^k = t^k + 604800 \quad \text{or } t_c = t_c + 604800 \end{aligned}$$

t_{oe} is usually obtained from ephemeris data found in the decoded navigation message, 604800 is the time of a week in seconds (i.e. $7 \times 24 \times 3600$) and 302400 is the time of half a week in seconds.

According to Tsui (2005: 62 & 63), the following process can be used to correct GPS time t^k :

Firstly, the mean motion or movement n is computed as:

$$n = \sqrt{\frac{\mu}{a_s^3}} + \Delta n \quad \text{Equation 8.50}$$

$\sqrt{a_s}$ and Δn are found in the decoded ephemeris data and $\mu = GM = 3.986005 \times 10^{14} \text{ m}^3/\text{s}^2$

Followed by the mean anomaly M , which is calculated from n or Equation 8.50 as:

$$M = M_0 + n(t_c - t_{oe}) \quad \text{Equation 8.51}$$

where M_0 is in the ephemeris data found in the decoded navigation message.

In Equation 8.51, it is advisable to rather use t_c as the initial GPS time since t^k is not yet obtained.

From M , the eccentric anomaly E is iteratively found from:

$$E = M + e_s \sin E \quad \text{Equation 8.52}$$

where e_s is the eccentricity of the satellite orbit obtained from the decoded ephemeris data.

Next, a constant F is defined as:

$$F = \frac{-2\sqrt{\mu}}{c^2} = 4.442807633 \times 10^{-10} \text{ sec} / \text{m}^{1/2} \quad \text{Equation 8.53}$$

From which the relativistic correction term is deduced as:

$$\Delta t_r = F e_s \sqrt{a_s} \sin E \quad \text{Equation 8.54}$$

As a result, the overall time correction term (satellites clock offset Δt^k) is computed as:

$$\Delta t^k = a_{f0} + a_{f1}(t_c - t_{oc}) + a_{f2}(t_c - t_{oc})^2 + \Delta t_r + T_{GD} \quad \text{Equation 8.55}$$

where $a_{f0}, a_{f1}, a_{f2}, t_{oc}$ are the individual clock corrections terms and T_{GD} is to account for the effect of satellite group delay differentials. All these parameters are found in the ephemeris data of the decoded GPS navigation data. For GPS to UTC time, see Kaplan and Hegarty (2006: 63).

Finally, the GPS time of transmission t^k , corrected for transit time can be obtained from:

$$t^k = t_c - \Delta t^k \quad \text{Equation 8.56}$$

Equation 8.56 is the applicable time correction for software GPS receivers, contrary to Equation 8.49 for hardware receivers. t_c defines the time, common to all SV pseudo-ranges observations.

It is very crucial to ensure that the corrected GPS time is used in the final position calculations.

8.2.7.5 GPS Receiver Position Computation Analysis

With the knowledge of GPS satellites position, pseudo-ranges, coordinate system and GPS time correction; the GPS receiver position can now be calculated. Borre and Strang (2012: xxxvii), postulated that, the most generally used algorithm for position computation from pseudo-ranges is based-on the least-squares method, which is employed when the unknowns exceed the observations. In what follows, is a mathematical analysis for using least-squares to compute GPS receiver position from pseudo-ranges p_i^k (containing clock error) to four or more GPS satellites.

From Equation 8.48, the basic observational equation for the pseudo-range p_i^k is therefore:

$$p_i^k = R_i^k + c(\Delta t_i - \Delta t^k) + T_{p_i^k} + I_{p_i^k} + M_{p_i^k} + \varepsilon_{p_i^k}$$

The geometrical (exact) range R_i^k between orbiting GPS satellites and receiver is computed as:

$$R_i^k = \sqrt{(X^k - X_i)^2 + (Y^k - Y_i)^2 + (Z^k - Z_i)^2} \quad \text{Equation 8.57}$$

Replacing R_i^k in Equation 8.48 with Equation 8.57 gives ≥ 4 equations for X_i, Y_i, Z_i and $c\Delta t_i$ as:

$$p_i^k = \sqrt{(X^k - X_i)^2 + (Y^k - Y_i)^2 + (Z^k - Z_i)^2} + c(\Delta t_i - \Delta t^k) + T_{p_i^k} + I_{p_i^k} + M_{p_i^k} + \varepsilon_{p_i^k} \quad \text{Equation 8.58}$$

From analysing Equation 8.58, the satellite clock offset (error) Δt^k is obtained from the ephemeris data found in the decoded navigation message, from which the GPS time is corrected and used to compute the position (X^k, Y^k, Z^k) of satellite k . The constant c is the speed of light. The sources of errors which include the tropospheric delay $T_{p_i^k}$, the multipath effect $M_{p_i^k}$ and the ionospheric delay $I_{p_i^k}$ are estimated from a priori model with the $I_{p_i^k}$ coefficients obtained from the broadcasted ephemeris in the decoded navigation message. The error term $(\varepsilon_{p_i^k})$ is minimised using the least-square method. Therefore, in Equation 8.58, the unknown parameters to be computed are the GPS receiver position (X_i, Y_i, Z_i) and the GPS receiver clock offset (Δt_i) . The other parameters are known. Generally, any other significant errors or sources of errors discussed in Sections 3.6 to 3.8 can be modelled separately if possible and added as term(s) to Equation 8.58. Any other errors or sources of errors which are insignificant can as well be treated as $(\varepsilon_{p_i^k})$.

Computing the GPS receiver position requires at least four pseudo-ranges (see Figure 8.2) and Equation 8.58 is nonlinear with respect to the receiver position (X_i, Y_i, Z_i) . Least-squares method thus, first requires linearisation of $\sqrt{(X^k - X_i)^2 + (Y^k - Y_i)^2 + (Z^k - Z_i)^2}$ in Equation 8.58.

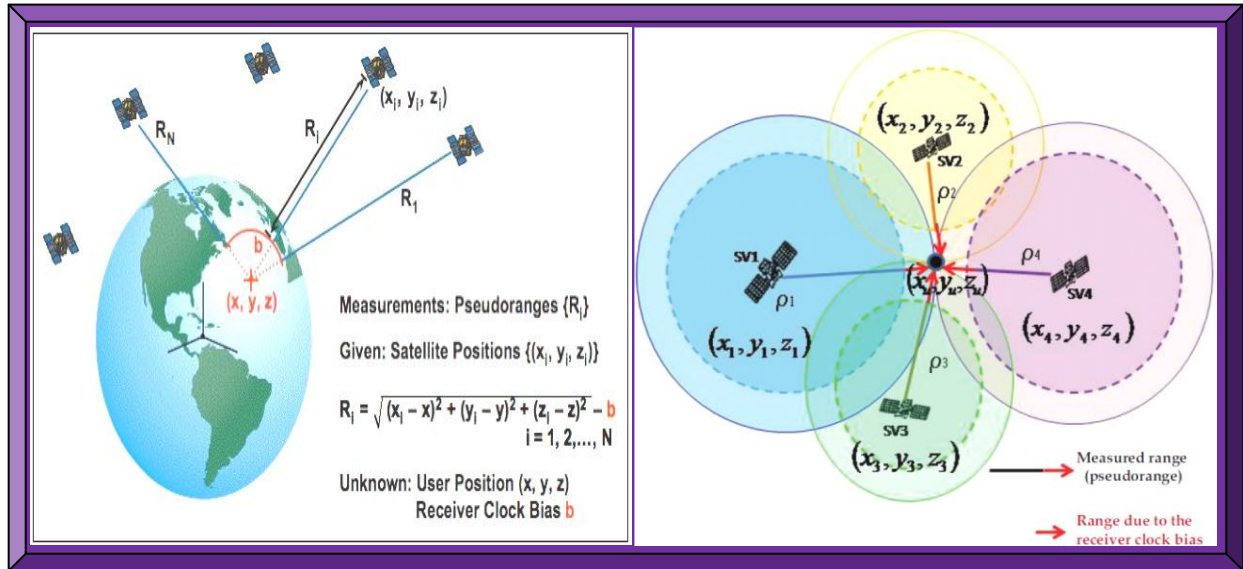


Figure 8.2: GPS receiver position computation using pseudo-ranges, SV position and clock bias
 (Adapted from Misra & Enge, 2006; Jin, 2012)

8.2.7.5.1 Linearisation of the Pseudo-range Observations

From Equation 8.58, the non-linear term is modeled and analysed as:

$$f(X_i, Y_i, Z_i) = \sqrt{(X^k - X_i)^2 + (Y^k - Y_i)^2 + (Z^k - Z_i)^2} \quad \text{Equation 8.59}$$

In Borre and Strang (2012: xxxix), linearisation begins by first finding an initial position for the GPS receiver $(X_{i=0}, Y_{i=0}, Z_{i=0})$, which for an Earth based receiver, is often chosen as Earth's centre $(0, 0, 0)$. The process is then updated with the increment $(\Delta X, \Delta Y, \Delta Z)$. This is written as:

$$\left. \begin{aligned} X_{i=1} &= X_{i=0} + \Delta X \\ Y_{i=1} &= Y_{i=0} + \Delta Y \\ Z_{i=1} &= Z_{i=0} + \Delta Z \end{aligned} \right\} \quad \text{Equation 8.60}$$

The approximate GPS receiver position is updated by this increments and the Taylor expansion of $f(X_{i=0} + \Delta X, Y_{i=0} + \Delta Y, Z_{i=0} + \Delta Z)$ is applied to the linear terms. This is represented as:

$$\left. \begin{aligned} f(X_{i=1}, Y_{i=1}, Z_{i=1}) &= f(X_{i=0}, Y_{i=0}, Z_{i=0}) + \frac{\delta f(X_{i=0}, Y_{i=0}, Z_{i=0})}{\delta X_{i=0}} \Delta X \\ &+ \frac{\delta f(X_{i=0}, Y_{i=0}, Z_{i=0})}{\delta Y_{i=0}} \Delta Y + \frac{\delta f(X_{i=0}, Y_{i=0}, Z_{i=0})}{\delta Z_{i=0}} \Delta Z \end{aligned} \right\} \quad \text{Equation 8.61}$$

Equation 8.61 only incorporates first order terms; hence, the updated function f estimates the GPS satellites position. The partial derivatives are as a result of the square root in Equation 8.59.

In Kaplan and Hegarty (2006: 56), the partial derivatives terms of Equation 8.61 is expressed as:

$$\left. \begin{aligned} \frac{\delta f(X_{i=0}, Y_{i=0}, Z_{i=0})}{\delta X_{i=0}} &= - \frac{X^k - X_{i=0}}{R_i^k} \\ \frac{\delta f(X_{i=0}, Y_{i=0}, Z_{i=0})}{\delta Y_{i=0}} &= - \frac{Y^k - Y_{i=0}}{R_i^k} \\ \frac{\delta f(X_{i=0}, Y_{i=0}, Z_{i=0})}{\delta Z_{i=0}} &= - \frac{Z^k - Z_{i=0}}{R_i^k} \end{aligned} \right\} \text{Equation 8.62}$$

Assuming $R_{i=0}^k$ to be the geometric range found from the approximate receiver position implies:

$$R_{i=0}^k = \sqrt{(X^k - X_{i=0})^2 + (Y^k - Y_{i=0})^2 + (Z^k - Z_{i=0})^2} \quad \text{Equation 8.63}$$

As a result, p_i^k for the four first-order linearised observation equations can be put acrossed as:

$$\left. \begin{aligned} p_i^k &= R_{i=0}^k - \frac{X^k - X_{i=0}}{R_i^k} \Delta X - \frac{Y^k - Y_{i=0}}{R_i^k} \Delta Y - \frac{Z^k - Z_{i=0}}{R_i^k} \Delta Z \\ &+ c(\Delta t_i - \Delta t^k) + T_{p_i^k} + I_{p_i^k} + M_{p_i^k} + \varepsilon_{p_i^k} \end{aligned} \right\} \text{Equation 8.64}$$

8.2.7.5.2 Application of Least-squares Method to the Linearised Observations

In Borre and Strang (2012: xxxix), a least-squares problem with no exact solution is stated as:

$$\mathbf{Ax} = \mathbf{b} \quad \text{Equation 8.65}$$

The matrix A has m rows and n columns, where $m > n$. There are therefore less free parameters x_1, \dots, x_m than more observations b_1, \dots, b_m . The length of the error vector (Equation 8.66) is minimised by the best choice $\hat{\mathbf{x}}$. If the standard way is used to measure this length, such that Equation 8.67 is the sum of squares of the m separate errors, then minimising this quadratic gives the standard equation for $\hat{\mathbf{x}}$ as conveyed in Equation 8.68. NB: the bold face letters mean vectors.

$$\hat{\mathbf{e}} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}} \quad \text{Equation 8.66}$$

$$\|\mathbf{e}^2\| = (\mathbf{b} - \mathbf{Ax})^T (\mathbf{b} - \mathbf{Ax}) \quad \text{Equation 8.67}$$

$$\mathbf{A}^T \mathbf{A} \hat{\mathbf{x}} = \mathbf{A}^T \mathbf{b} \quad \text{or} \quad \hat{\mathbf{x}} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} \quad \text{Equation 8.68}$$

The covariance matrix for the parameter $\hat{\mathbf{x}}$ can be written as:

$$\left. \begin{aligned} \Sigma_{\hat{\mathbf{x}}} &= \hat{\sigma}_0^2 (A^T A)^{-1} \\ \text{where } \hat{\sigma}_0^2 &= \frac{\hat{\mathbf{e}}^T \hat{\mathbf{e}}}{m-n} \end{aligned} \right\} \quad \text{Equation 8.69}$$

The linearised pseudo-range (Equation 8.64) can then be expressed in vector form as:

$$\left. \begin{aligned} p_i^k &= R_{i=0}^k - \left[\frac{X^k - X_{i=0}}{R_i^k} + \frac{Y^k - Y_{i=0}}{R_i^k} + \frac{Z^k - Z_{i=0}}{R_i^k} - 1 \right] [\Delta X \ \Delta Y \ \Delta Z \ c\Delta t_i]^T \\ &\quad - c\Delta t^k + T_{p_i}^k + I_{p_i}^k + M_{p_i}^k + \varepsilon_{p_i}^k \end{aligned} \right\} \quad \text{Equation 8.70}$$

Re-arranging Equation 8.70 into a least-squares problem in the matrix form of $A \mathbf{x} = \mathbf{b}$, yields:

$$\underbrace{\begin{bmatrix} -\frac{X^k - X_{i=0}}{R_i^k} - \frac{Y^k - Y_{i=0}}{R_i^k} - \frac{Z^k - Z_{i=0}}{R_i^k} + 1 \end{bmatrix}}_A \underbrace{\begin{bmatrix} \Delta X \\ \Delta Y \\ \Delta Z \\ c\Delta t_i \end{bmatrix}}_{\mathbf{x}} = \underbrace{p_i^k - R_{i=0}^k + c\Delta t^k - T_{p_i}^k - I_{p_i}^k - M_{p_i}^k - \varepsilon_{p_i}^k}_{\mathbf{b}} \quad \text{Equation 8.71}$$

A unique least-squares solution is found provided there are $m \geq 4$ equations. From Equation 8.71, \mathbf{b} can be written as in Equation 8.72, where the unknowns are the updates to X_i, Y_i, Z_i and Δt_i .

$$\mathbf{b}_i^k = p_i^k - R_{i=0}^k + c\Delta t^k - T_{p_i}^k - I_{p_i}^k - M_{p_i}^k - \varepsilon_{p_i}^k \quad \text{Equation 8.72}$$

Finally, by manipulating Equation 8.71, the GPS receiver solution equation can be expressed as:

$$A \mathbf{x} = \begin{bmatrix} -\frac{X^1 - X_{i=0}}{R_0^1} - \frac{Y^1 - Y_{i=0}}{R_0^1} - \frac{Z^1 - Z_{i=0}}{R_0^1} + 1 \\ -\frac{X^2 - X_{i=0}}{R_0^2} - \frac{Y^2 - Y_{i=0}}{R_0^2} - \frac{Z^2 - Z_{i=0}}{R_0^2} + 1 \\ -\frac{X^3 - X_{i=0}}{R_0^3} - \frac{Y^3 - Y_{i=0}}{R_0^3} - \frac{Z^3 - Z_{i=0}}{R_0^3} + 1 \\ \vdots \\ -\frac{X^m - X_{i=0}}{R_0^m} - \frac{Y^m - Y_{i=0}}{R_0^m} - \frac{Z^m - Z_{i=0}}{R_0^m} + 1 \end{bmatrix} \begin{bmatrix} \Delta X_{i=1} \\ \Delta Y_{i=1} \\ \Delta Z_{i=1} \\ c\Delta t_{i=1} \end{bmatrix} = \mathbf{b} - \mathbf{e} \quad \text{Equation 8.73}$$

For $m \geq 4$, there is a unique solution $\Delta X_{i=1}, \Delta Y_{i=1}, \Delta Z_{i=1}$. This unique solution is then added to the approximate GPS receiver position to get the next approximate receiver's position as follows:

$$X_{i=1} = X_{i=0} + \Delta X_{i=1} \quad Y_{i=1} = Y_{i=0} + \Delta Y_{i=1} \quad Z_{i=1} = Z_{i=0} + \Delta Z_{i=1} \quad \text{Equation 8.74}$$

The next iteration restarts from Equations 8.71 to 8.74 from $i = 0$, incremented to $i = 1$, next $i = 2$ etc (often max of $i = 3$) until the solution $\Delta X_{i=1}, \Delta Y_{i=1}, \Delta Z_{i=1}$ converges to meter level accuracy.

8.2.7.6 GPS Receiver Velocity Computation Analysis

With the GPS receiver position computed, the velocity estimates can be deduced using a similar least-squares approach to get the first derivative of the position – which is basically velocity.

In Gleason and Gebre-Egziabher (2009: 64), the pseudo-range (Equation 8.48) is usually the starting point for deriving the velocity equation. Therefore, using a similar rationale to that used in developing the position solution, this equation can be written in terms of GPS satellite-to-receiver relative velocities, line of sight unit vector and associated errors which are expressed as:

$$\dot{p}_i^k = [v^k - v_i]L_{unit}^k + \dot{b} + \dot{T}_{p_i^k} + \dot{I}_{p_i^k} + \dot{M}_{p_i^k} + \dot{\epsilon}_{p_i^k} \quad \text{Equation 8.75}$$

where:

- \dot{p}_i^k : is the pseudo-range rate for satellite k measured in m/s (known)
- v^k : is the velocity vector for satellite k (often calculated from navigation data) in m/s.
- v_i : is the velocity vector for receiver i measured in m/s (unknown)
- L_{unit}^k : is the receiver-to-satellite line-of-sight unit vector (derived from Taylor expansion)
- \dot{b} : is the rate of change of the receiver clock or clock drift in m/s (unknown)
- $\dot{T}_{p_i^k}$: is the rate of change of the tropospheric delay
- $\dot{I}_{p_i^k}$: is the rate of change of the ionospheric delay
- $\dot{M}_{p_i^k}$: is the rate of change of the multipath delay
- $\dot{\epsilon}_{p_i^k}$: is an accumulation of the unknown pseudo-range rate error terms in m/s

Assuming a simplistic model with the errors terms ignored, the GPS receiver velocity in Grewal *et al.* (2001: 28), can be estimated and written as:

$$\dot{p}_i^k = [(X^k - X_{i=0})(\dot{X}^k - \dot{X}_{i=0}) + (Y^k - Y_{i=0})(\dot{Y}^k - \dot{Y}_{i=0}) + (Z^k - Z_{i=0})(\dot{Z}^k - \dot{Z}_{i=0})]/p_i^k \quad \text{Equation 8.76}$$

where:

- \dot{p}_i^k = range rate (known)
- p_i^k = range (known)
- (X^k, Y^k, Z^k) = satellite positions (known)
- $(\dot{X}^k, \dot{Y}^k, \dot{Z}^k)$ = satellite position rate (known)
- $(X_{i=0}, Y_{i=0}, Z_{i=0})$ = receiver position (known from position calculations)
- $(\dot{X}_{i=0}, \dot{Y}_{i=0}, \dot{Z}_{i=0})$ = receiver velocity (unknown and can therefore be computed)

Expanding and rearranging Equation 8.76 yields:

$$\begin{aligned} \dot{p}_i^k + \frac{1}{p_i^k} [\dot{X}^k(X^k - X_{i=0}) + \dot{Y}^k(Y^k - Y_{i=0}) + \dot{Z}^k(Z^k - Z_{i=0})] & \quad \text{Equation 8.77} \\ = \frac{1}{p_i^k} [(X^k - X_{i=0}) \dot{X}_{i=0} + (Y^k - Y_{i=0}) \dot{Y}_{i=0} + (Z^k - Z_{i=0}) \dot{Z}_{i=0}] & \end{aligned}$$

For $m \geq 4$ equations (i.e. when the number of orbiting satellites ≥ 4), Equation 8.77 translates to:

$$\left[\begin{array}{c} \dot{p}_i^1 + \frac{1}{p_i^1} [\dot{X}^1(X^1 - X_{i=0}) + \dot{Y}^1(Y^1 - Y_{i=0}) + \dot{Z}^1(Z^1 - Z_{i=0})] \\ \dot{p}_i^2 + \frac{1}{p_i^2} [\dot{X}^2(X^2 - X_{i=0}) + \dot{Y}^2(Y^2 - Y_{i=0}) + \dot{Z}^2(Z^2 - Z_{i=0})] \\ \dot{p}_i^3 + \frac{1}{p_i^3} [\dot{X}^3(X^3 - X_{i=0}) + \dot{Y}^3(Y^3 - Y_{i=0}) + \dot{Z}^3(Z^3 - Z_{i=0})] \\ \vdots \\ \dot{p}_i^m + \frac{1}{p_i^m} [\dot{X}^m(X^m - X_{i=0}) + \dot{Y}^m(Y^m - Y_{i=0}) + \dot{Z}^m(Z^m - Z_{i=0})] \end{array} \right] = \left[\begin{array}{ccc} \frac{(X^1 - X_{i=0})}{p_i^1} & \frac{Y^1 - Y_{i=0}}{p_i^1} & \frac{(Z^1 - Z_{i=0})}{p_i^1} \\ \frac{(X^2 - X_{i=0})}{p_i^2} & \frac{Y^2 - Y_{i=0}}{p_i^2} & \frac{(Z^2 - Z_{i=0})}{p_i^2} \\ \frac{(X^3 - X_{i=0})}{p_i^3} & \frac{Y^3 - Y_{i=0}}{p_i^3} & \frac{(Z^3 - Z_{i=0})}{p_i^3} \\ \vdots & \vdots & \vdots \\ \frac{(X^m - X_{i=0})}{p_i^m} & \frac{Y^m - Y_{i=0}}{p_i^m} & \frac{(Z^m - Z_{i=0})}{p_i^m} \end{array} \right] \left[\begin{array}{c} \dot{X}_{i=0} \\ \dot{Y}_{i=0} \\ \dot{Z}_{i=0} \end{array} \right] \quad \text{Equation 8.78}$$

where Equation 8.78 is in the standard form $\mathbf{b} = \mathbf{A} \dot{\mathbf{x}}$.

If $m \geq 4$, there is a unique solution $[\dot{X}_{i=0}, \dot{Y}_{i=0}, \dot{Z}_{i=0}]$. This unique solution is then added to the approximate GPS receiver velocity to get the next approximate receiver's velocity and this iterative optimisation process continues until the GPS receiver velocity solution converges.

8.2.7.7 GPS Receiver Position Transformation from XYZ (ECEF) to Geodetic Coordinate

Once the GPS receiver position is determined (usually in XYZ rectangular coordinate referenced to Earth's centre), it must be transformed to a useful coordinate system (e.g. ENU coordinate) which suits the intended application, usually to a point at the surface of Earth or above Earth (See Figure 8.3). Coordinate transformations are means for converting a vector characterised in one coordinate system to a suitable characterisation in a different coordinate system. In Borre and Strang (2012: 39), the choice of the preferred coordinate system goes in line with the chosen Earth model (as a sphere or ellipsoid or geoid). The geoid is the most accurate (mm level) model because of the considerations of flattening / perturbations corrections but also the most complex. Geodetic simply refers to geographic plus inclusion of altitude. Borre and Strang (2012: 39 - 49), covers various methods for changing a GPS receiver's position on Earth. Newton's ellipsoidal model is chosen, as it is applicable to Earth and space-borne GPS receivers and is thus analysed.

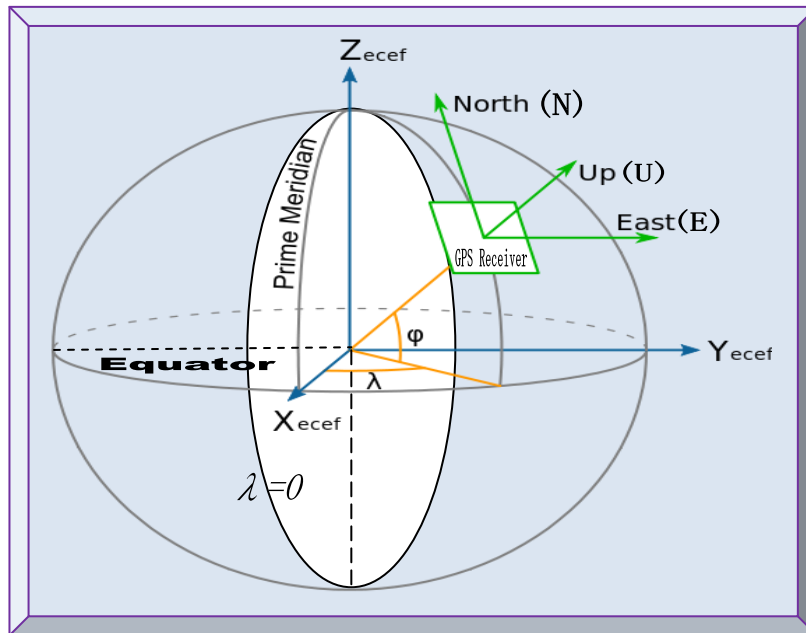


Figure 8.3: Transformation between ECEF to ENU coordinates
(Adapted from Borre and Strang, 2012: 52)

8.2.7.7.1 Newton's Method for Cartesian (XYZ) to Geodetic (φ λ h) Transformation

Newton's method is an optimisation technique best use to approximate non-linear programming problems of quadratic nature. In Borre and Strang (2012: 45), this method is applied as follows:

$$\varphi = \arctan\left(\frac{Z}{\sqrt{X^2 + Y^2}}\right) \quad \lambda = \arctan\left(\frac{Y}{X}\right) \quad h = \sqrt{X^2 + Y^2 + Z^2} - a \quad \text{Equation 8.79}$$

where φ , λ , h will be the corresponding latitude, longitude and altitude from the XYZ position.

Initial values for $\varphi_0, \lambda_0, h_0$ are chosen for the spherical case; $f = 0$, f is the flattening of the Earth. From Equations 8.80 and 8.81, λ is calculated explicitly from Equation 8.82; therefore ($\lambda = \lambda_0$).

$$X = (N + h) \cos\varphi \cos\lambda \quad \text{Equation 8.80}$$

$$Y = (N + h) \cos\varphi \sin\lambda \quad \text{Equation 8.81}$$

$$\lambda = \tan^{-1} \left(\frac{Y}{X} \right) \quad \text{Equation 8.82}$$

where N is the distance along the line which passes through the Z-axis and it is represented as:

$$N = a / \sqrt{1 - f(2 - f)\sin^2\varphi} \quad \text{Equation 8.83}$$

Since λ is clearly known from Equation 8.82, Newton's method is applied to find φ and h only:

$$\sqrt{X^2 + Y^2} = (N + h) \cos\varphi \quad \text{Equation 8.84}$$

$$Z = [(1 - f)^2 N + h] \sin\varphi \quad \text{Equation 8.85}$$

Let Equations 8.86 and 8.87 respectively define the functions $g_1(\varphi, h)$ and $g_2(\varphi, h)$ as:

$$g_1(\varphi, h) \equiv (N + h)\cos\varphi - \sqrt{X^2 + Y^2} = 0 \quad \text{Equation 8.86}$$

$$g_2(\varphi, h) \equiv [(1 - f)^2 N + h]\sin\varphi - Z = 0 \quad \text{Equation 8.87}$$

From Equation 8.83, the first partial derivative of N with respect to φ becomes:

$$\begin{aligned} \frac{\partial N}{\partial \varphi} &= a(-0.5)(1 - f(2 - f)\sin^2\varphi)^{-1.5}(-2f(2 - f)\sin\varphi\cos\varphi) \\ &= \frac{N^3}{a^2} f(2 - f)\sin\varphi \cos\varphi \end{aligned} \quad \text{Equation 8.88}$$

Similarly, the first partial derivatives of $g_1(\varphi, h)$ and $g_2(\varphi, h)$ with respect to φ and h are required as per Newton's method and these derivatives are simply the Jacobian of Equations 8.86 and 8.87, defined as J :

$$J = \begin{bmatrix} \frac{\partial g_1}{\partial \varphi} & \frac{\partial g_1}{\partial h} \\ \frac{\partial g_2}{\partial \varphi} & \frac{\partial g_2}{\partial h} \end{bmatrix}$$

The four elements of the Jacobian J are then calculated as follows:

$$\begin{aligned} \frac{\partial g_1}{\partial \varphi} &= \frac{\partial N}{\partial \varphi} \cos\varphi - (N + h) \sin\varphi = \frac{N^3}{a^2} f(2 - f) \sin\varphi \cos^2\varphi - N \sin\varphi - h \sin\varphi \\ &= \frac{N^3}{a^2} \sin\varphi \left[f(2 - f) \cos^2\varphi - \frac{a^2}{N^2} \right] - h \sin\varphi \\ &= \frac{N^3}{a^2} \sin\varphi [f(2 - f) \cos^2\varphi - 1 + f(2 - f) \sin^2\varphi] - h \sin\varphi \\ &= \frac{N^3}{a^2} \sin\varphi (-1 + 2f - f^2) - h \sin\varphi = - \left[\frac{N^3}{a^2} (1 - f)^2 + h \right] \sin\varphi \end{aligned}$$

$$\begin{aligned}
\frac{\partial g_2}{\partial \varphi} &= (1-f)^2 \frac{\partial N}{\partial \varphi} \sin \varphi - [(1-f)^2 N + h] \cos \varphi \\
&= (1-f)^2 \frac{N^3}{a^2} f(2-f) \sin^2 \varphi \cos \varphi + (1-f)^2 N \cos \varphi + h \cos \varphi \\
&= (1-f)^2 \frac{N^3}{a^2} \cos \varphi \left[f(2-f) \sin^2 \varphi + \frac{a^2}{N} \right] + h \cos \varphi \\
&= (1-f)^2 \frac{N^3}{a^2} \cos \varphi [f(2-f) \sin^2 \varphi + 1 - f(2-f) \sin^2 \varphi] + h \cos \varphi \\
&= (1-f)^2 \frac{N^3}{a^2} \cos \varphi + h \cos \varphi = \left[\frac{N^3}{a^2} (1-f)^2 + h \right] \cos \varphi
\end{aligned}$$

The other two partial derivatives are simply:

$$\frac{\partial g_1}{\partial h} = \cos \varphi$$

$$\frac{\partial g_2}{\partial h} = \sin \varphi$$

Substituting the Jacobian elements with their respective derivatives and simplifying reduces to:

$$J = \begin{bmatrix} -\left[\frac{N^3}{a^2} (1-f)^2 + h \right] \sin \varphi & \cos \varphi \\ \left[\frac{N^3}{a^2} (1-f)^2 + h \right] \cos \varphi & \sin \varphi \end{bmatrix}$$

Replacing the term $\left[\frac{N^3}{a^2} (1-f)^2 + h \right]$ with D gives:

$$J = \begin{bmatrix} -D \sin \varphi & \cos \varphi \\ D \cos \varphi & \sin \varphi \end{bmatrix}$$

Furthermore, if the determinant is $-D$, the inverse is:

$$J^{-1} = \frac{1}{\det(J)} \begin{bmatrix} \sin \varphi & -\cos \varphi \\ -D \cos \varphi & -D \sin \varphi \end{bmatrix} = \begin{bmatrix} -\frac{1}{D} \sin \varphi & \frac{1}{D} \cos \varphi \\ \cos \varphi & \sin \varphi \end{bmatrix}$$

Finally, the solution (φ, h) using Newton's method is deduced as follows:

- 1) Firstly, an initial guess as the current solution $(\varphi, h)_{current}$ is started.
- 2) Secondly, for each iteration, a new solution and current values of g_1 and g_2 are computed and updated as follows:

$$\begin{bmatrix} \varphi \\ h \end{bmatrix}_{new} = \begin{bmatrix} \varphi \\ h \end{bmatrix}_{current} - \begin{bmatrix} -\frac{1}{D} \sin \varphi & \frac{1}{D} \cos \varphi \\ \cos \varphi & \sin \varphi \end{bmatrix}_{current} \begin{bmatrix} g_1(\varphi, h) \\ g_2(\varphi, h) \end{bmatrix}_{current} \quad \text{Equation 8.89}$$

After approximately four iterations, the optimised values of φ and h should converge to zero.

8.2.7.8 Dilution of Precision (DOP) Analysis

DOP provides a measure of how orbiting GPS satellites are spaced with respect to a GPS receiver. DOP values are useful when planning observational periods. There are five types of DOPs, namely GDOP, PDOP, HDOP, VDOP and TDOP. DOPs values are computed as follows:

$$\text{Geometric DOP: } \text{GDOP} = \sqrt{(\sigma_X^2 + \sigma_Y^2 + \sigma_Z^2 + \sigma_{\text{cdt}}^2)}/\sigma_{\text{UERE}} = \sqrt{(\sigma_E^2 + \sigma_N^2 + \sigma_U^2 + \sigma_{\text{cdt}}^2)}/\sigma_{\text{UERE}}$$

$$\text{Position DOP: } \text{PDOP} = \sqrt{(\sigma_X^2 + \sigma_Y^2 + \sigma_Z^2)}/\sigma_{\text{UERE}} = \sqrt{(\sigma_E^2 + \sigma_N^2 + \sigma_U^2)}/\sigma_{\text{UERE}}$$

$$\text{Horizontal DOP: } \text{HDOP} = \sqrt{(\sigma_X^2 + \sigma_Y^2)}/\sigma_{\text{UERE}} = \sqrt{(\sigma_E^2 + \sigma_N^2)}/\sigma_{\text{UERE}}$$

$$\text{Vertical DOP: } \text{VDOP} = \sigma_Z/\sigma_{\text{UERE}} = \sigma_U/\sigma_{\text{UERE}}$$

$$\text{Time DOP: } \text{TDOP} = \sigma_{\text{cdt}}/\sigma_{\text{UERE}} \quad \text{where } c\sigma_{\text{dt}} = \sigma_{\text{cdt}}$$

$$\text{GDOP} = (\text{PDOP}^2 + \text{TDOP}^2)^{1/2} = (\text{HDOP}^2 + \text{VDOP}^2 + \text{TDOP}^2)^{1/2} \quad \text{Equation 8.90}$$

Strang and Borre (1997: 462 & 463) states, all DOPs value are dimensionless and if multiplied with range errors, they give approximate position errors. Very good observations are achieved, if PDOP is < 5 and when at least ≥ 4 orbiting GPS satellites are utilised in the PVT measurements.

From the GPS receiver solution analysed in Section 8.2.7.5, the covariance matrix Σ_{ECEF} for (X_i, Y_i, Z_i, cdt_i) contains information regarding the geometric quality of the GPS receiver position determination. When the GPS satellites are well spaced in orbit, DOP is smaller and \hat{x} more accurate. The Σ_{ECEF} trace harmonises this DOP information into a single number as the sum of the four variances $\sigma_X^2 + \sigma_Y^2 + \sigma_Z^2 + \sigma_{\text{cdt}}^2$ (independent of the coordinate system). Furthermore to the GDOP information, these variances involve the accuracy of the observations.

Instead of relying on up to date (to first acquire GPS satellites signal) ephemerides to compute satellites position, DOP values enable GPS receivers to use low accuracy almanac coarse values to predict GPS satellites position with sufficient accuracy. Since the quality of GPS satellites constellation to receiver geometry varies from time to time, the knowledge of DOPs is essential.

$$\text{Therefore, from covariance (dx)} = \Sigma_{ECEF} = \begin{bmatrix} \sigma_X^2 & \sigma_{XY} & \sigma_{XZ} & \sigma_{X,\text{cdt}} \\ \sigma_{YX} & \sigma_Y^2 & \sigma_{YZ} & \sigma_{Y,\text{cdt}} \\ \sigma_{ZX} & \sigma_{ZY} & \sigma_Z^2 & \sigma_{Z,\text{cdt}} \\ \text{---} & \text{---} & \text{---} & \text{---} \\ \sigma_{\text{cdt},X} & \sigma_{\text{cdt},Y} & \sigma_{\text{cdt},Z} & \sigma_{\text{cdt}}^2 \end{bmatrix} \quad \text{Equation 8.91}$$

The propagation law transforms Equation 8.91 to a covariance matrix expressed in ENU frame.

Letting the top left sub-matrix in Equation 8.91 be represented by S , this sub-matrix S , transforms to an ENU frame (see Equation 8.93) after applying the DCM F^T (see Equation 8.92).

As indicated in Strang and Borre (1997: 462), the matrix F links the ECEF coordinate system with the Cartesian coordinate differences in the local frame at φ (latitude) and λ (longitude) as F^T

$$\begin{aligned}
 F^T &= R_{3(\pi)} R_{2(\varphi-0.5\pi)} R_{3(\lambda-\pi)} \\
 &= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sin\varphi & 0 & \cos\varphi \\ 0 & 1 & 0 \\ -\cos\varphi & 0 & \sin\varphi \end{bmatrix} \begin{bmatrix} -\cos\lambda & \sin\lambda & 0 \\ \sin\lambda & -\cos\lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\sin\varphi \cos\lambda & -\sin\varphi \sin\lambda & \cos\varphi \\ \cos\varphi \cos\lambda & \cos\varphi \sin\lambda & \sin\varphi \end{bmatrix} \quad \text{Equation 8.92}
 \end{aligned}$$

$$\Sigma_{\text{ENU}} = F^T S F = \begin{bmatrix} \sigma_E^2 & \sigma_{\text{EN}} & \sigma_{\text{EU}} \\ \sigma_{\text{NE}} & \sigma_N^2 & \sigma_{\text{NU}} \\ \sigma_{\text{UE}} & \sigma_{\text{UN}} & \sigma_U^2 \end{bmatrix} \quad \text{Equation 8.93}$$

For detailed mathematical analysis on general coordinate transformations, consult Grewal *et al.* (2001: 324 - 369) and for converting GPS coordinates to navigation frame, refer to Drake (2002).

8.2.7.9 Maximum Doppler Shift Computation Analysis

For completeness, this section analyses how maximum Doppler shift is inferred. As emphasised in Sections 3.7.3.4 and 8.2.4, excessive Doppler shift accounts for most of the challenges in GPS receiver acquisition and tracking, which are very crucial for a space-borne GPS receiver in LEO.

In Nortier (2003: 22 & 23), Doppler frequency shift in space is in the region of $\sim \pm 60$ kHz and can be calculated as follows:

$$f' = f_0 \sqrt{\frac{1 + v/c}{1 - v/c}} = f_0 + f'_d \quad \text{Equation 8.94}$$

where:

- f' = New GPS L1 carrier frequency
- f_0 = GPS L1 carrier frequency
- f'_d = Doppler frequency shift
- v = GPS receiver relative velocity
- c = speed of light (299.792458×10^6 m/s)

From Equation 8.94, f'_d is computed as:

$$f'_d = f_0 \left[\sqrt{\frac{1 + vc^{-1}}{1 - vc^{-1}}} - 1 \right] \quad \text{Equation 8.95}$$

In orbital mechanics, the velocity of a satellite in circular orbit is:

$$v = v_{\text{sat}} = \sqrt{\frac{\mu}{r}} \quad \text{Equation 8.96}$$

where:

μ = GM which is Earth's universal gravitational constant

r = Earth's orbital radius

In typical LEO, a CubeSat and GPS satellites at respective altitudes of ~700 and ~20200 km, have corresponding velocities in the order of ~ 8 and ~ 4 km/s. However, the resultant relative velocity will never be reached ± 12 km/s because of the altitude differences and therefore with a value of ± 10 km/s as a rough maximum relative velocity (Nortier, 2003: 22 & 23), the maximum Doppler shift in the carrier frequency will be ~ ± 55 kHz. Doppler shift also affects the C/A code frequency (1023 kHz) but at a much lesser degree. In addition to Doppler shift, Doppler shift rate is also problematic and could cause tracking issues, especially in narrow band PLLs.

For an in-depth and alternative way for calculating Doppler frequency shift and Doppler rates, as well as related GPS signal Doppler parameters, consult Zheng (2005: 15 - 18).

8.3 Summary

Appendix B systematically analysed the applicable mathematics involved at each stage of GPS L1 C/A signal processing; from the point of signal transmission by orbiting GPS satellites to acquisition, tracking and final processing of the decoded navigation data for PVT computations, which is the ultimate goal. Sources of GPS errors were not analysed discretely as laid down in the theoretical section in Chapter 3, since they were already considered as an integral part of the transmitted GPS signal. Effort was made to portray the complete mathematical design of a GPS receiver, with emphasis on applications in space where feasible. As seen, a GPS receiver is a sophisticated maths processor, involving various facets of applied mathematics. As a result, analysing relevant GPS mathematical aspects and subsequent modelling is imperative, as it sets the path for better algorithms designs, proper implementations and development in software.

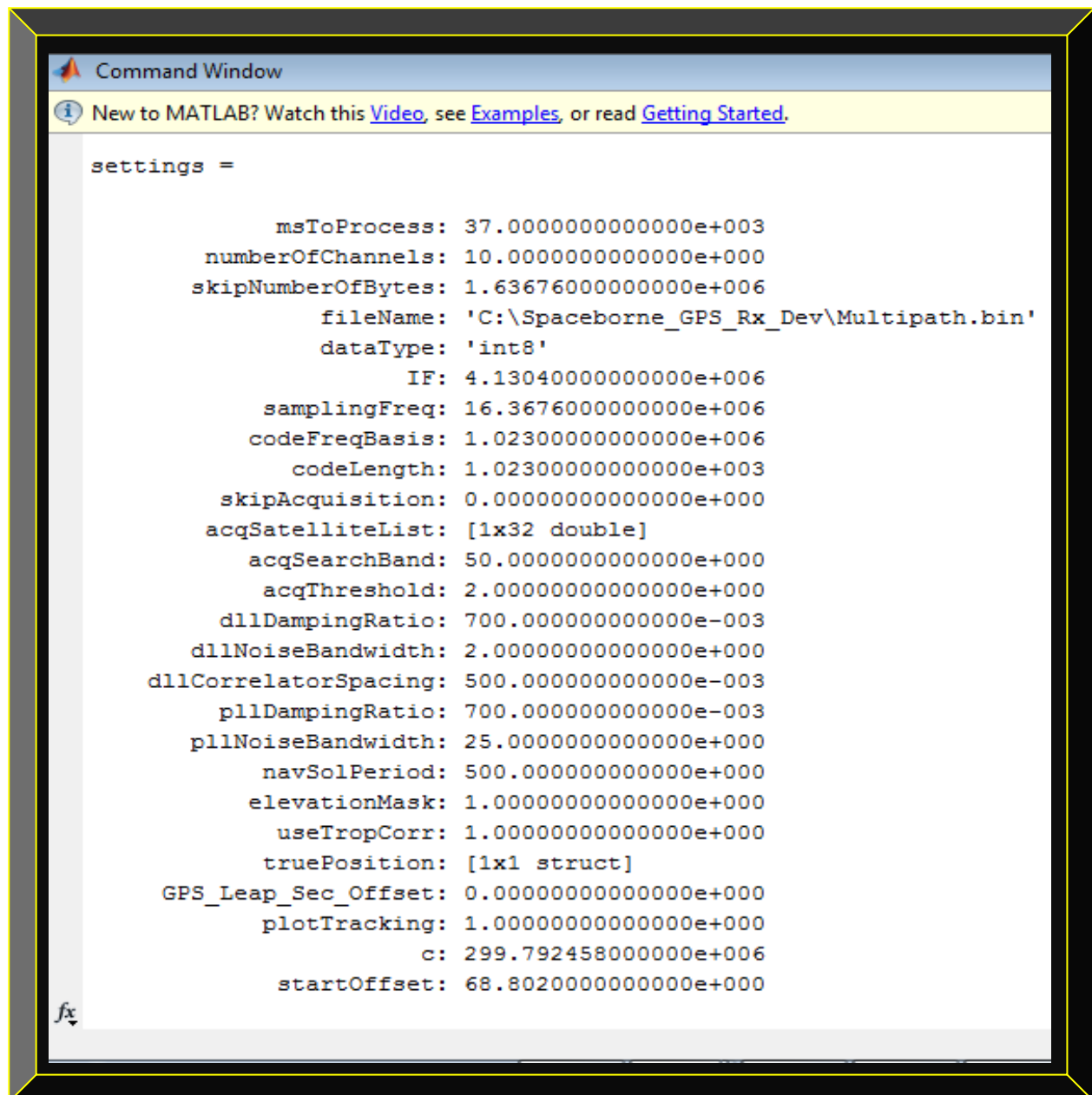
APPENDIX C

GPS EXTRA MISCELLANEOUS RESULTS

Appendix C contains detailed results of the Matlab floating-point GPS receiver. It includes the acquired and tracked channels acquisition and tracking results, plus some few orbital parameters.

9.1 Floating-point Matlab GPS Receiver Acquisition Results

Figure 9.2 shows the acquired results of the GPS receiver first 10 channels, defined in Figure 9.1.



```
Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.

settings =

    msToProcess: 37.00000000000000e+003
  numberOfChannels: 10.00000000000000e+000
 skipNumberOfBytes: 1.63676000000000e+006
      fileName: 'C:\Spaceborne_GPS_Rx_Dev\Multipath.bin'
      dataType: 'int8'
           IF: 4.13040000000000e+006
   samplingFreq: 16.36760000000000e+006
  codeFreqBasis: 1.02300000000000e+006
   codeLength: 1.02300000000000e+003
 skipAcquisition: 0.00000000000000e+000
acqSatelliteList: [1x32 double]
   acqSearchBand: 50.00000000000000e+000
   acqThreshold: 2.00000000000000e+000
  dllDampingRatio: 700.000000000000e-003
 dllNoiseBandwidth: 2.00000000000000e+000
dllCorrelatorSpacing: 500.000000000000e-003
  pllDampingRatio: 700.000000000000e-003
  pllNoiseBandwidth: 25.00000000000000e+000
   navSolPeriod: 500.000000000000e+000
   elevationMask: 1.00000000000000e+000
   useTropCorr: 1.00000000000000e+000
   truePosition: [1x1 struct]
GPS_Leap_Sec_Offset: 0.00000000000000e+000
   plotTracking: 1.00000000000000e+000
           c: 299.792458000000e+006
   startOffset: 68.80200000000000e+000
```

Figure 9.1: Configuration settings of the software GPS receiver before processing

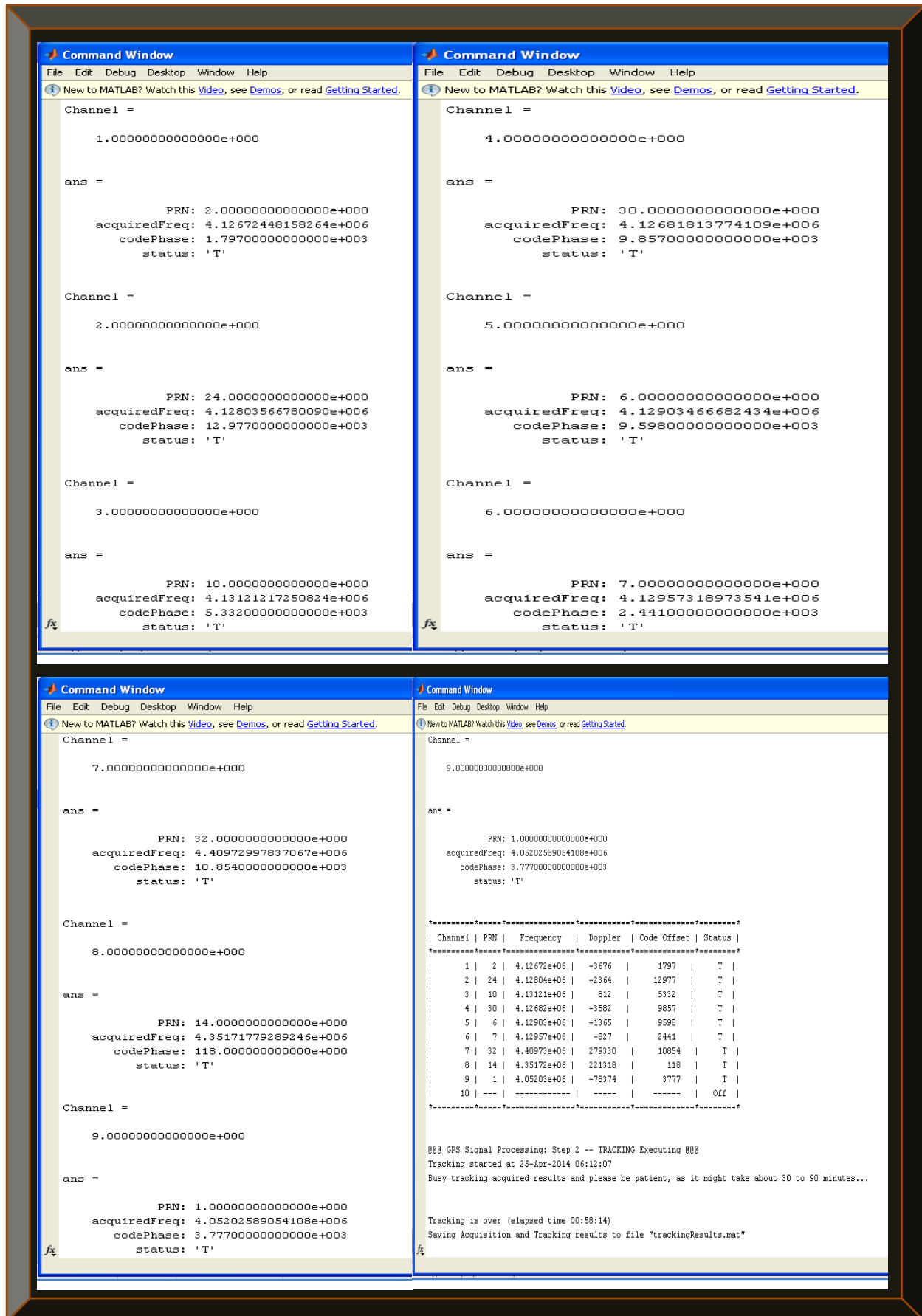


Figure 9.2: Abridged acquisition results of the Matlab floating-point software GPS receiver

9.2 Floating-point Matlab GPS Receiver Tracking Results

This section depicts the very detailed tracking results of each tracked GPS satellite or channel.

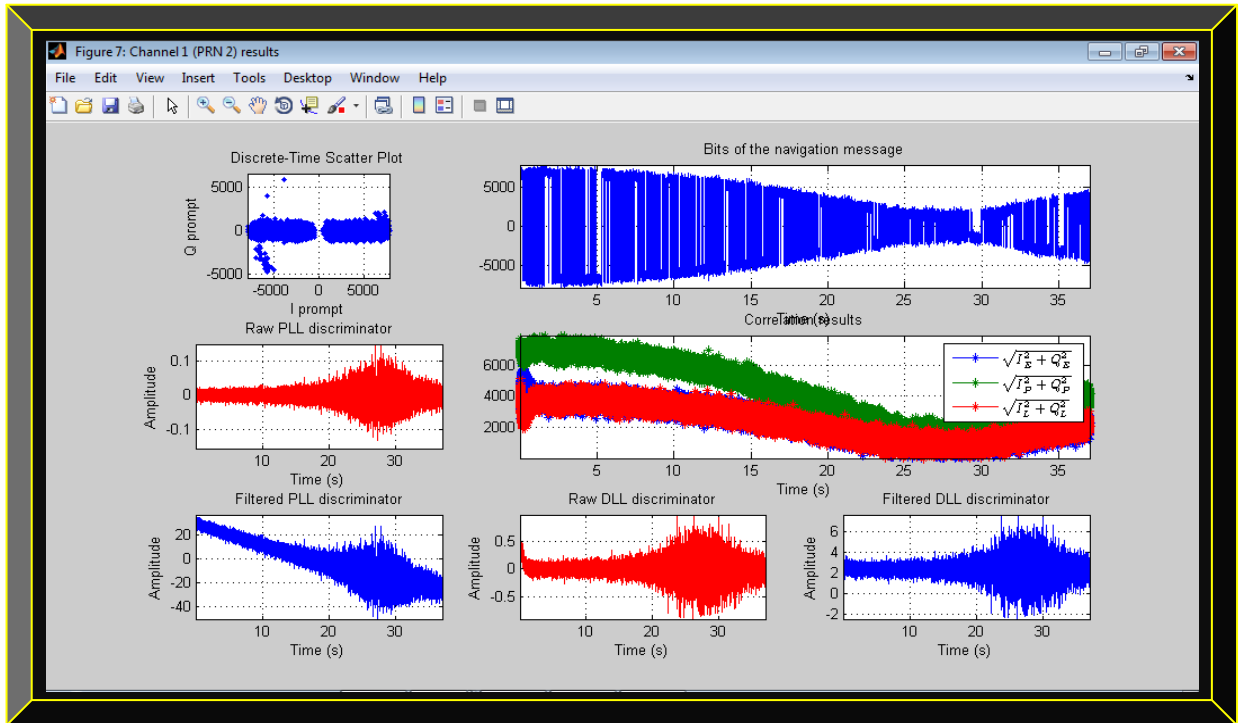


Figure 9.3: Channel 1 illustration of tracked GPS satellite #2, showing fully decoded navigation message

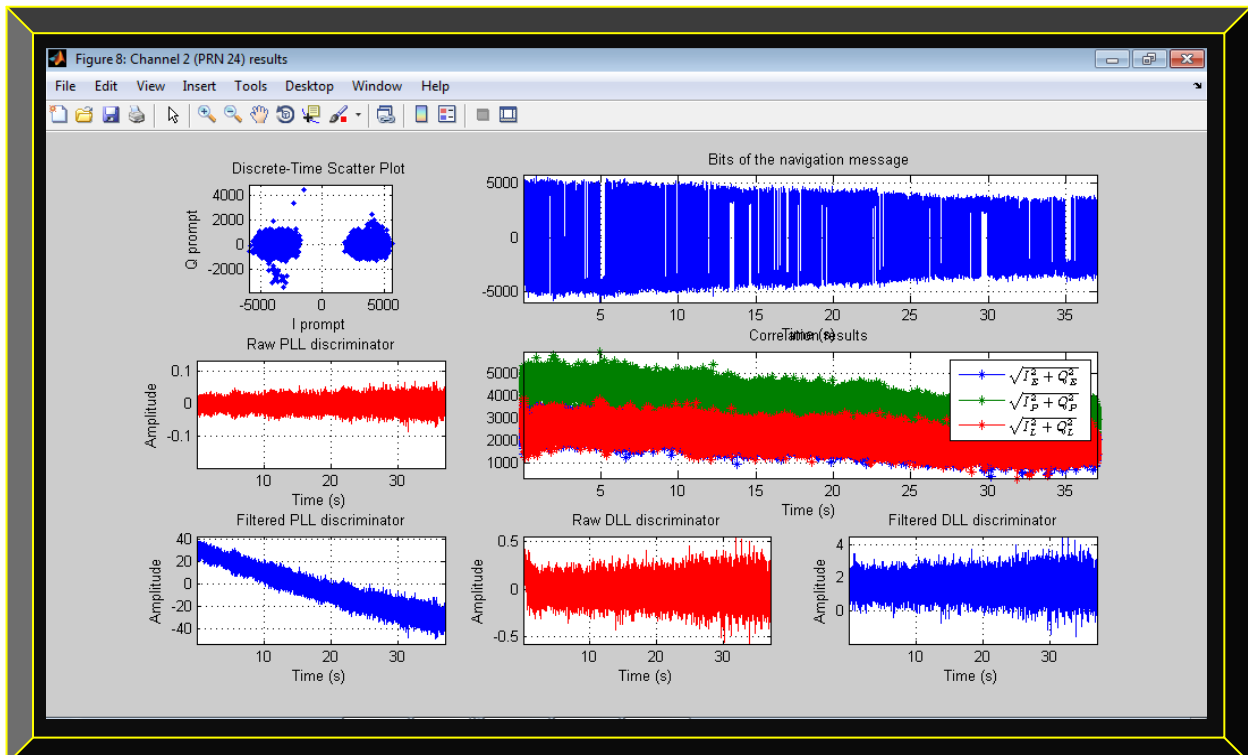


Figure 9.4: Channel 2 illustration of tracked GPS satellite #24, showing fully decoded navigation message

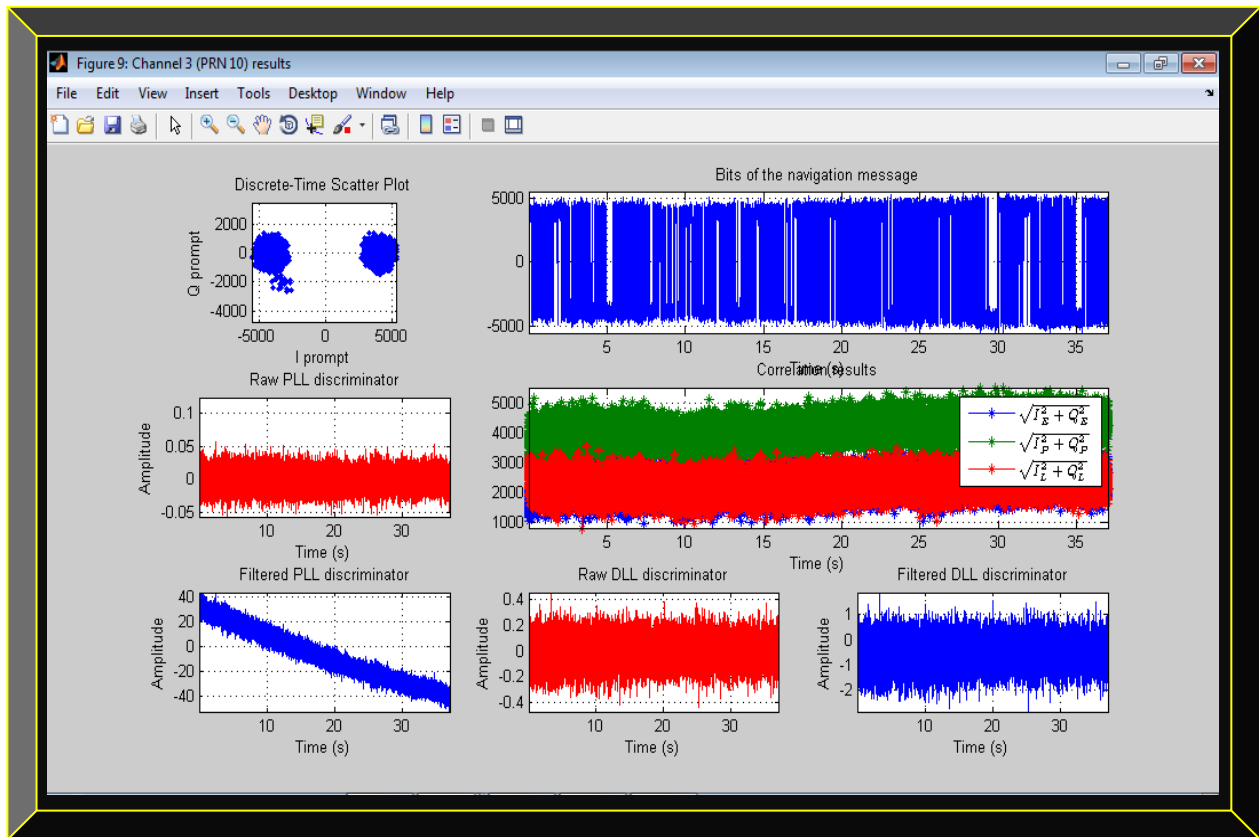


Figure 9.5: Channel 3 illustration of tracked GPS satellite #10, showing fully decoded navigation message

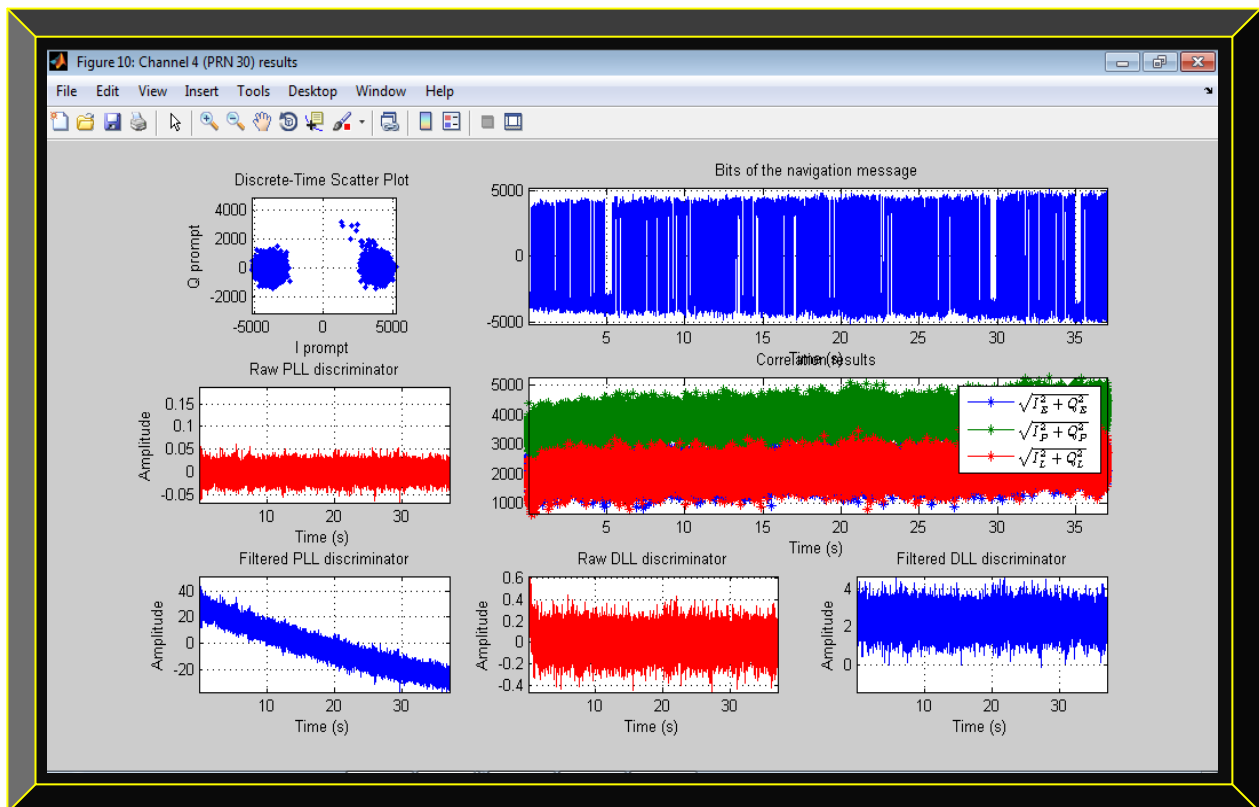


Figure 9.6: Channel 4 illustration of tracked GPS satellite #30, showing fully decoded navigation message

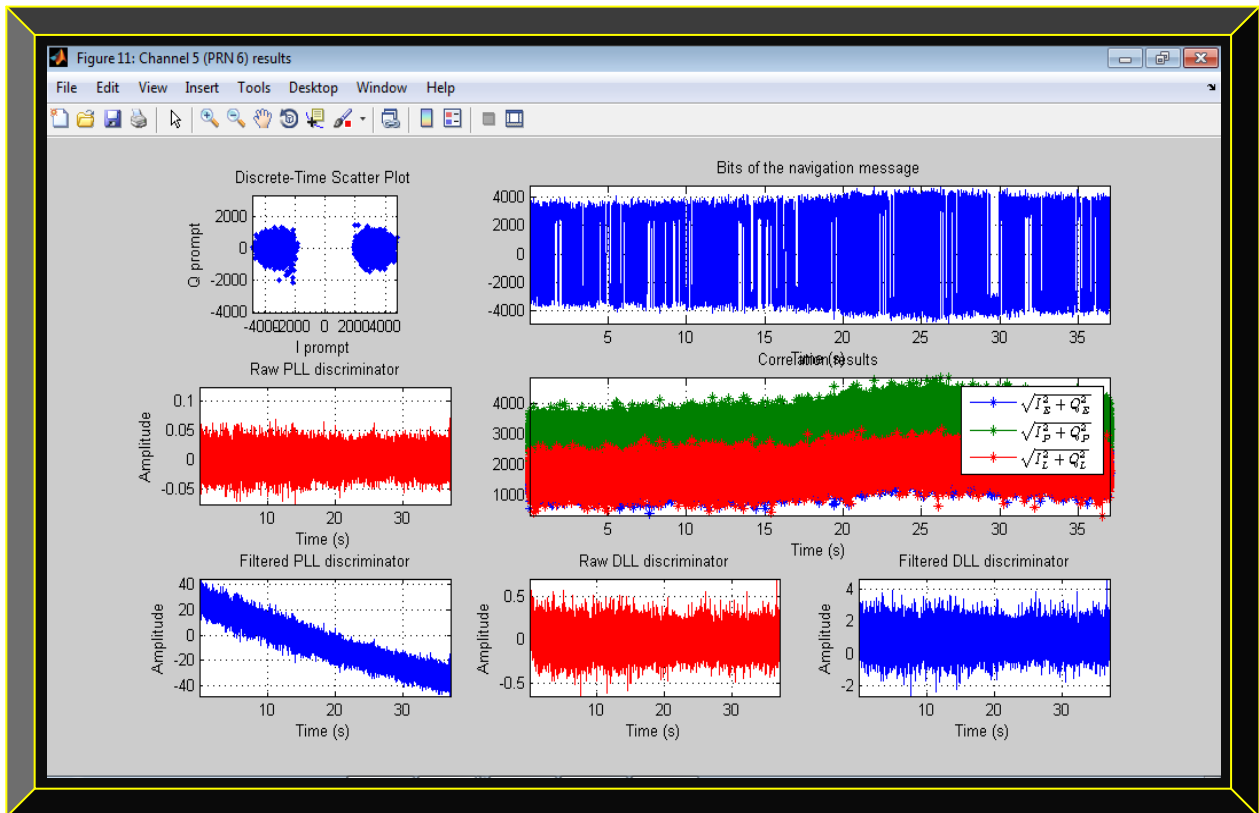


Figure 9.7: Channel 5 illustration of tracked GPS satellite #6, showing fully decoded navigation message

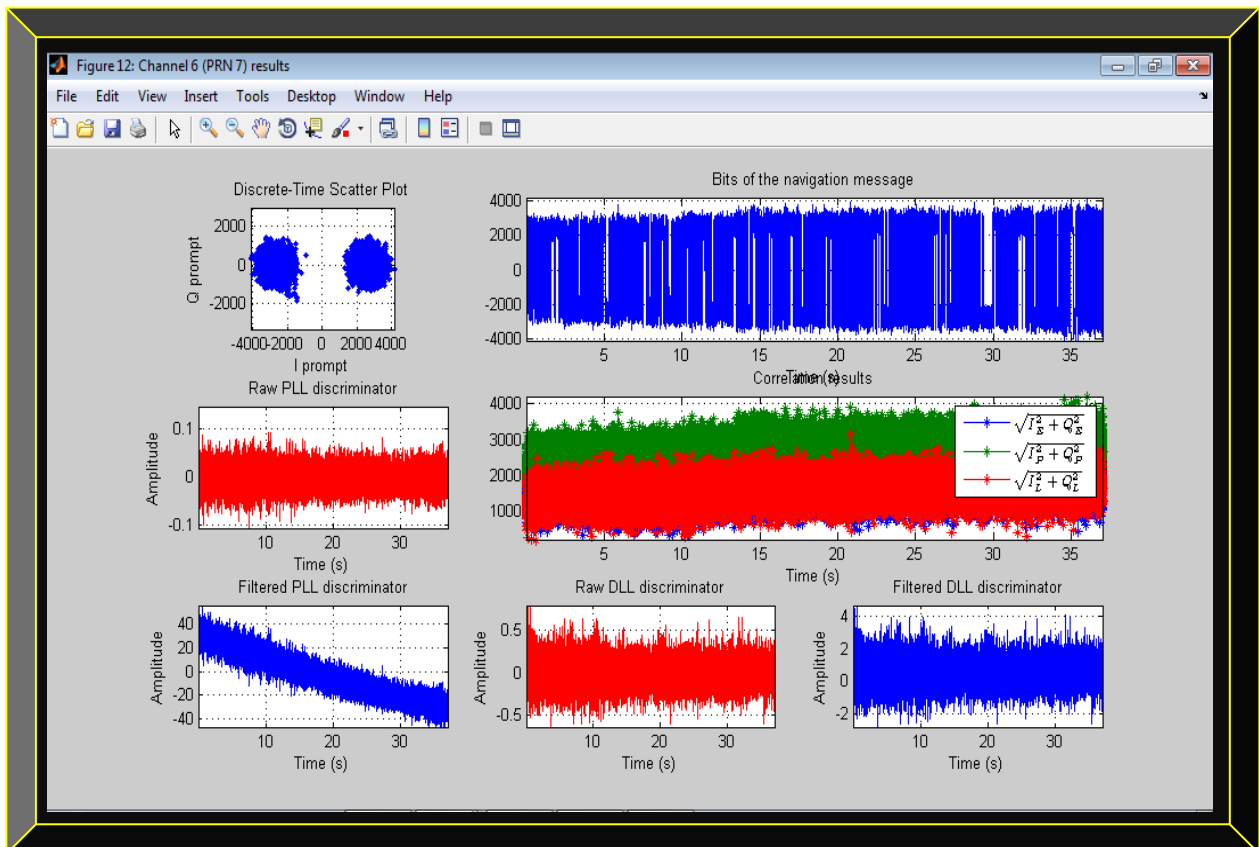


Figure 9.8: Channel 6 illustration of tracked GPS satellite #7, showing fully decoded navigation message

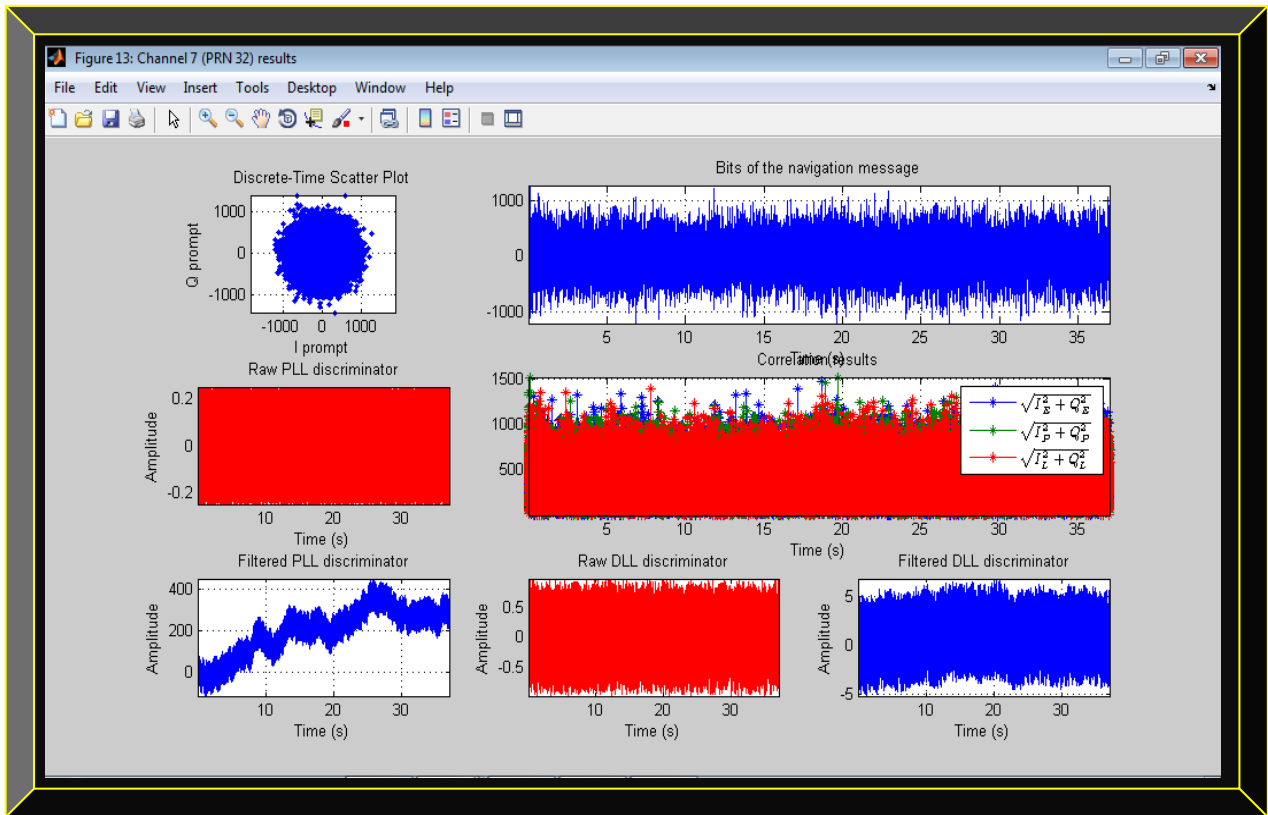


Figure 9.9: Channel 7 illustration of tracked GPS satellite #32, showing poor decoded navigation data

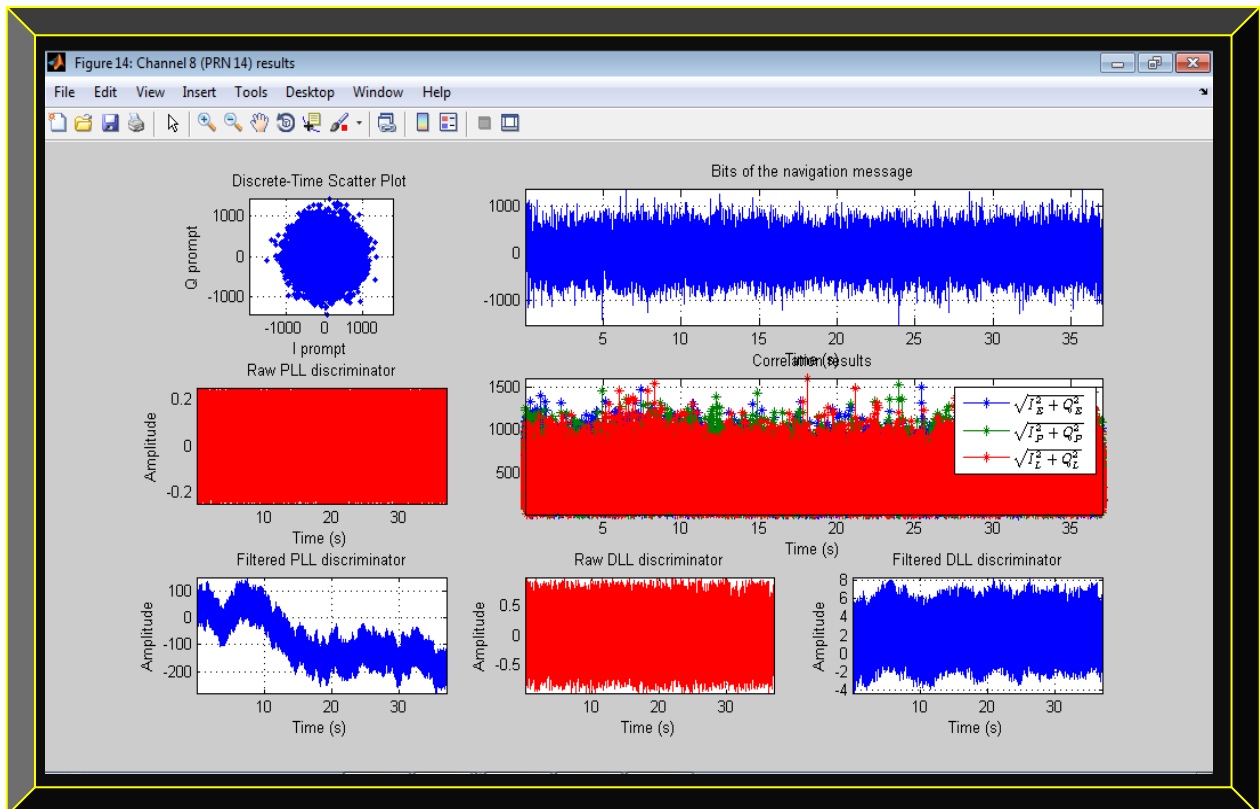


Figure 9.10: Channel 8 illustration of tracked GPS satellite #14, showing poor decoded navigation data

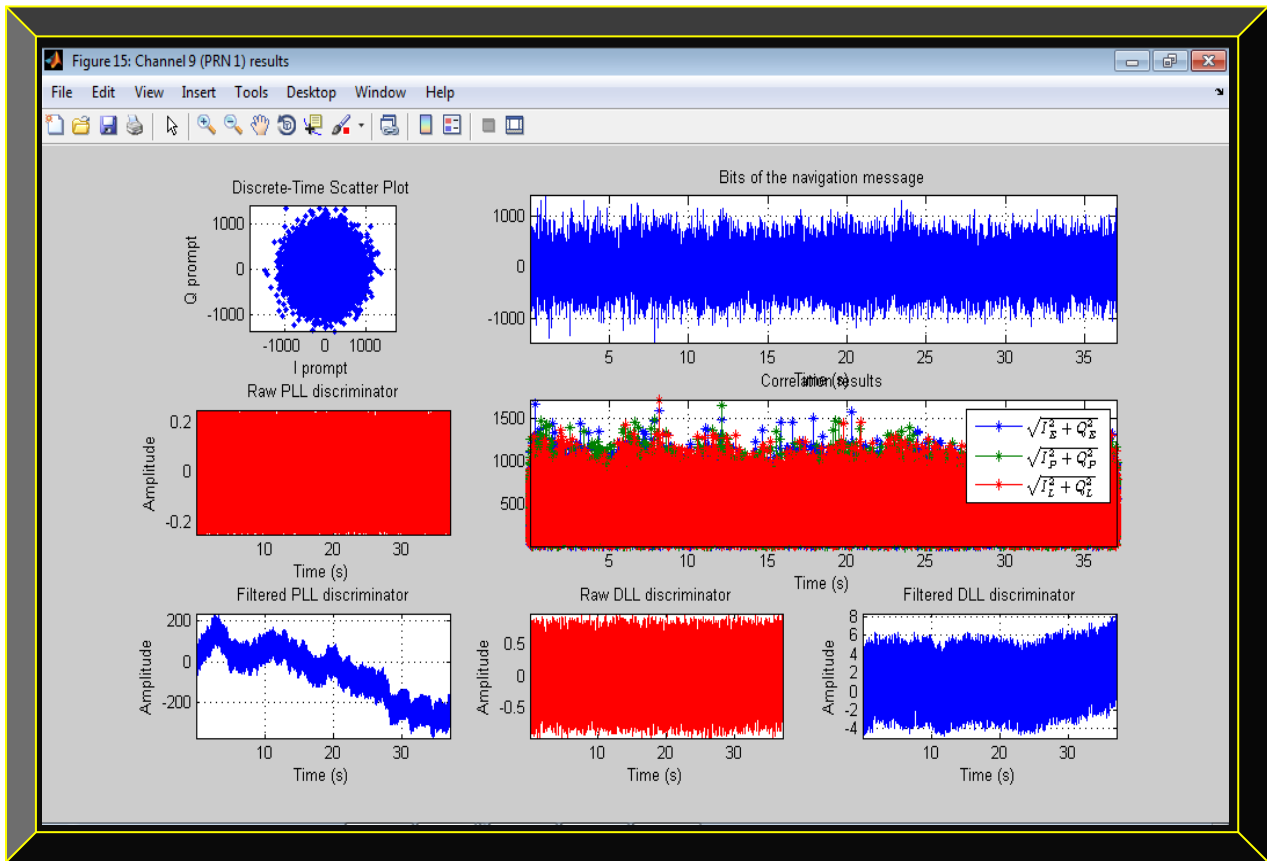


Figure 9.11: Channel 9 illustration of tracked GPS satellite #1, showing poor decoded navigation data

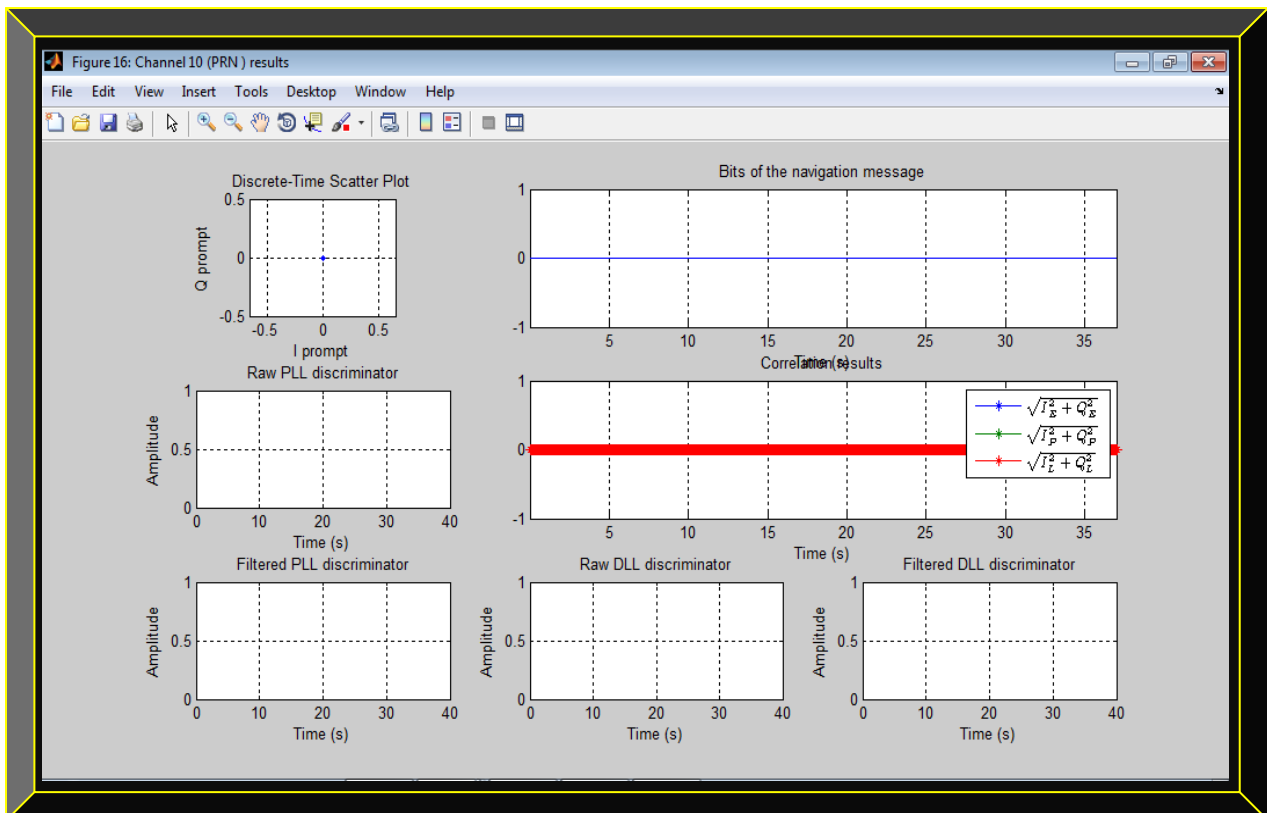


Figure 9.12: Channel 10 illustration of un-tracked GPS satellite #: Navigation message decoding failed

9.3 First 4 GPS Satellites Orbital Parameters Preview

This section simply shows some vital details of the computed GPS satellites orbital parameters; such as the transmit time of the satellites signals, their travel times, pseudo-ranges, satellites positions, satellites clocks corrections, the GPS receiver position, the time bias, DOPs and UTM.

```

Processing_results_2 - Notepad
File Edit Format View Help
utmZone =
32
ans =
0
transmitTime =
120310.5
travelTime =
Columns 1 through 3
77.6735175953075      69.3478822091878      71.8592458455517
Column 4
71.1655141739978
pseudoranges =
Columns 1 through 3
23285934.7614035      20789972.0645869      21542859.9420642
Column 4
21334884.4190567
time =
120310.499926527
satPositions =
Columns 1 through 3
-14237224.9134546      0      0
13300911.2689854      0      0
17704414.6805625      0      0
Column 4
0
0
0
satClkCorr =
Columns 1 through 3
7.34738495015329e-05      0      0
Column 4
0
time =
120310.499931201
satPositions =
Columns 1 through 3
-14237224.9134546      -550965.175708401      0
13300911.2689854      15383286.3109627      0
17704414.6805625      21862661.4555584      0
Column 4
0

Processing_results_2 - Notepad
File Edit Format View Help
satClkCorr =
Columns 1 through 3
7.34738603842355e-05      6.87845936912457e-05      9.59229878409886e-05
Column 4
0
time =
120335.999977401
satPositions =
Columns 1 through 3
-14296175.7552491      -620497.402176107      -3031370.72951761
13298136.9795797      15382519.414943      19006009.4638696
17658904.3120931      21862085.6166128      18055833.00712
Column 4
18027983.2353065
18456513.2937491
6242584.72900386
satClkCorr =
Columns 1 through 3
7.34738603842355e-05      6.87845936912457e-05      9.59229878409886e-05
Column 4
2.25771476968877e-05
pos =
Columns 1 through 3
3427958.42376531      603727.275334636      5326678.83400409
Column 4
-1608693.55445933
GDOP =
8.400223475997
PDOP =
7.13911661296331
HDOP =
6.52597634467637
VDOP =
2.89458438505653
TDOP =
4.42682374092333
utmZone =
32
ans =
0
transmitTime =

```

Figure 9.13: Preview of the first four acquired GPS satellites orbital parameters

9.4 Researched GPS Receiver Algorithms with Problems during the Conversion Process

This section presents the failures encountered in the software-to-firmware conversion processes.

9.4.1 acquisition.m: performs cold start searching for available GPS satellites signals

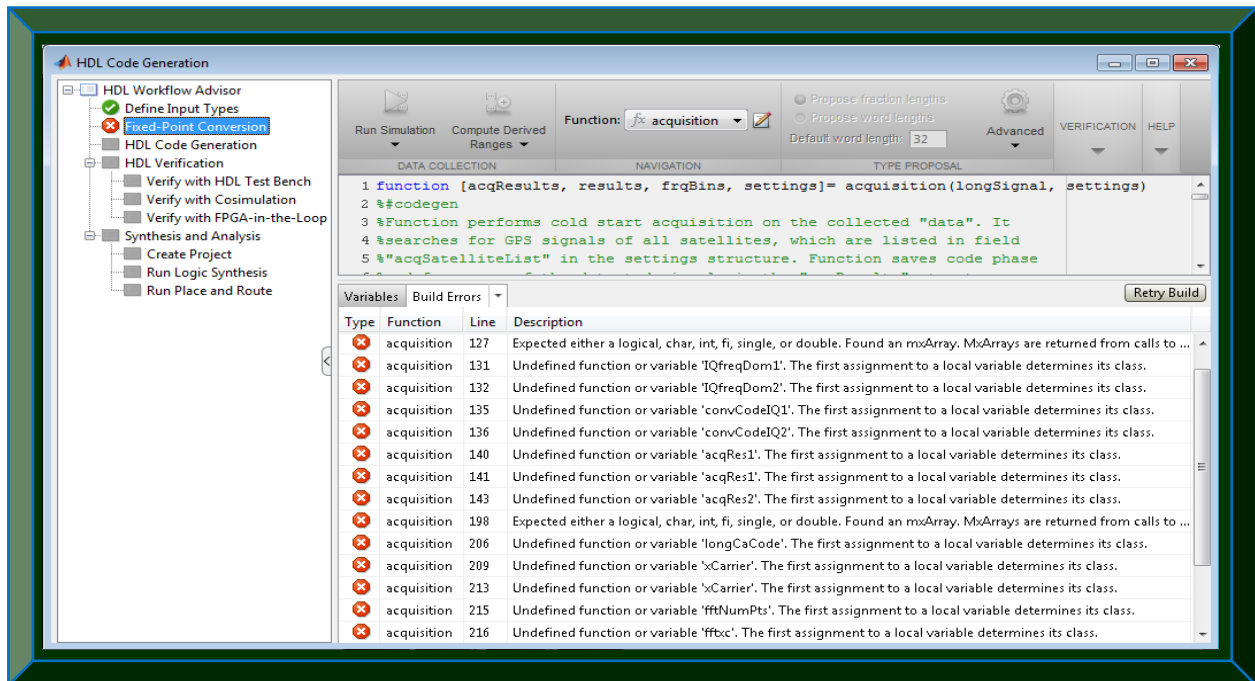


Figure 9.14: Illustration of some conversion issues in the acquisition.m algorithm

9.4.2 calculatePseudoranges.m: computes the range error between satellites and receiver

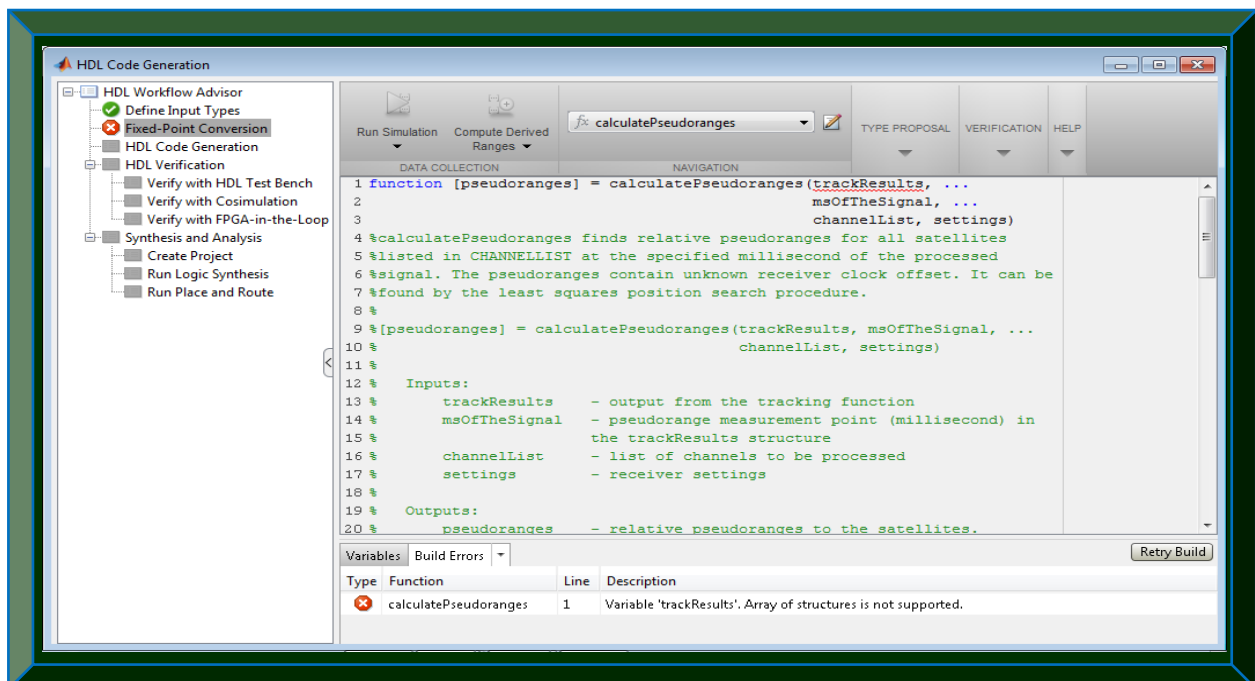


Figure 9.15: Illustration of some conversion issues in the calculatePseudoranges.m algorithm

9.4.3 cart2geo.m: transforms rectangular (ECEF) coordinates to geographic

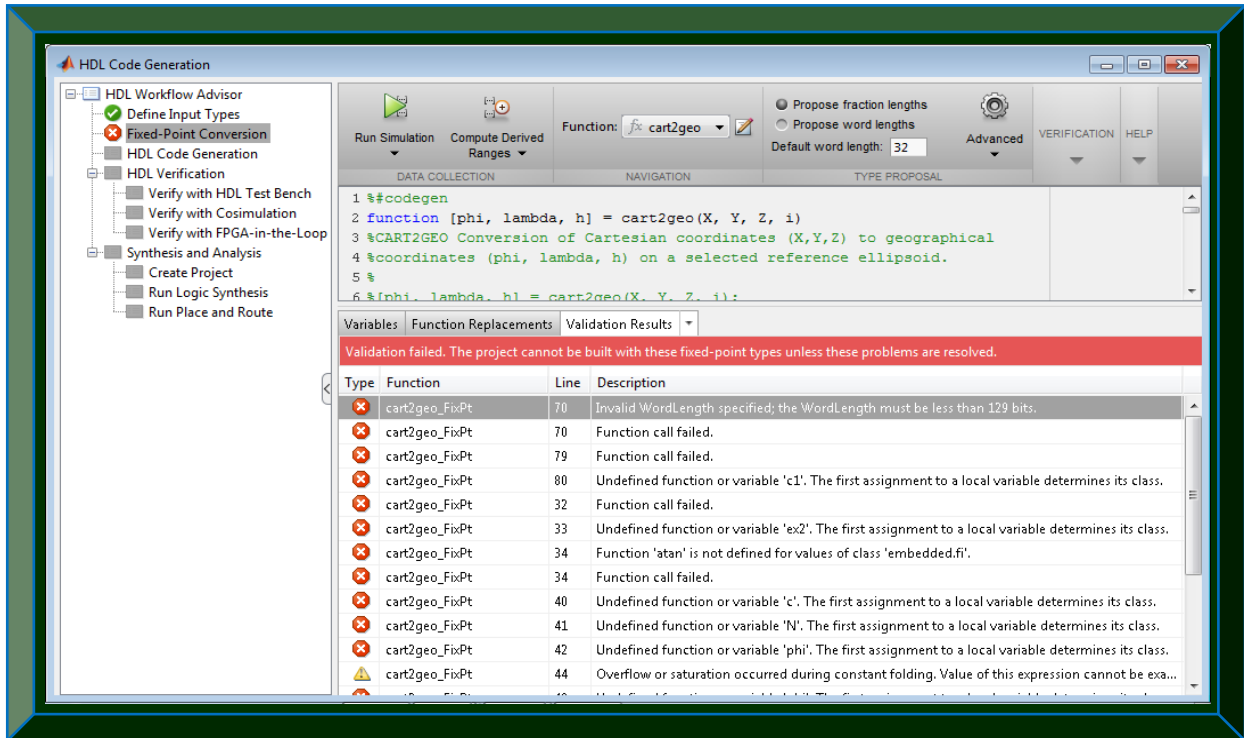


Figure 9.16: Illustration of some conversion issues in the cart2geo.m algorithm

9.4.4 cart2utm.m: converts Cartesian (XYZ) coordinates to ENU in UTM

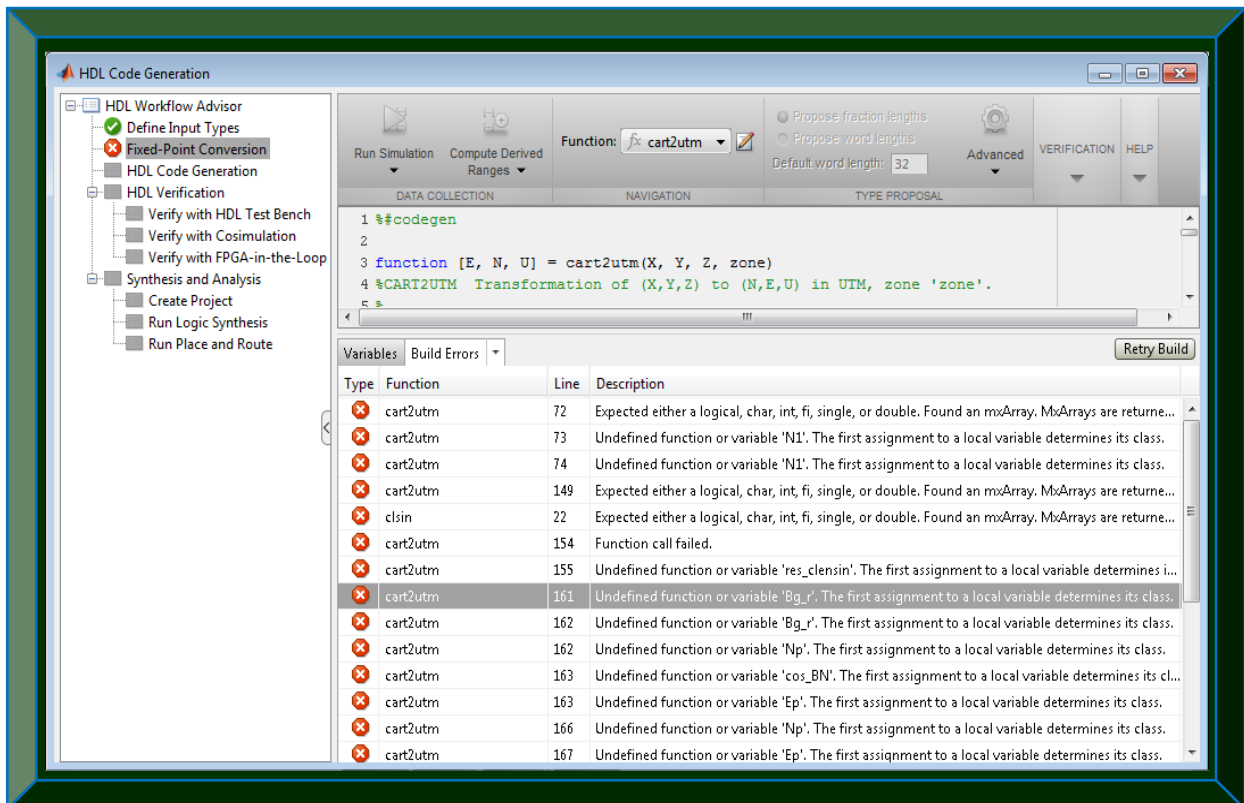


Figure 9.17: Illustration of some conversion issues in the cart2utm.m algorithm

9.4.5 checkPhase.m: checks the phase of the decoded navigation data message

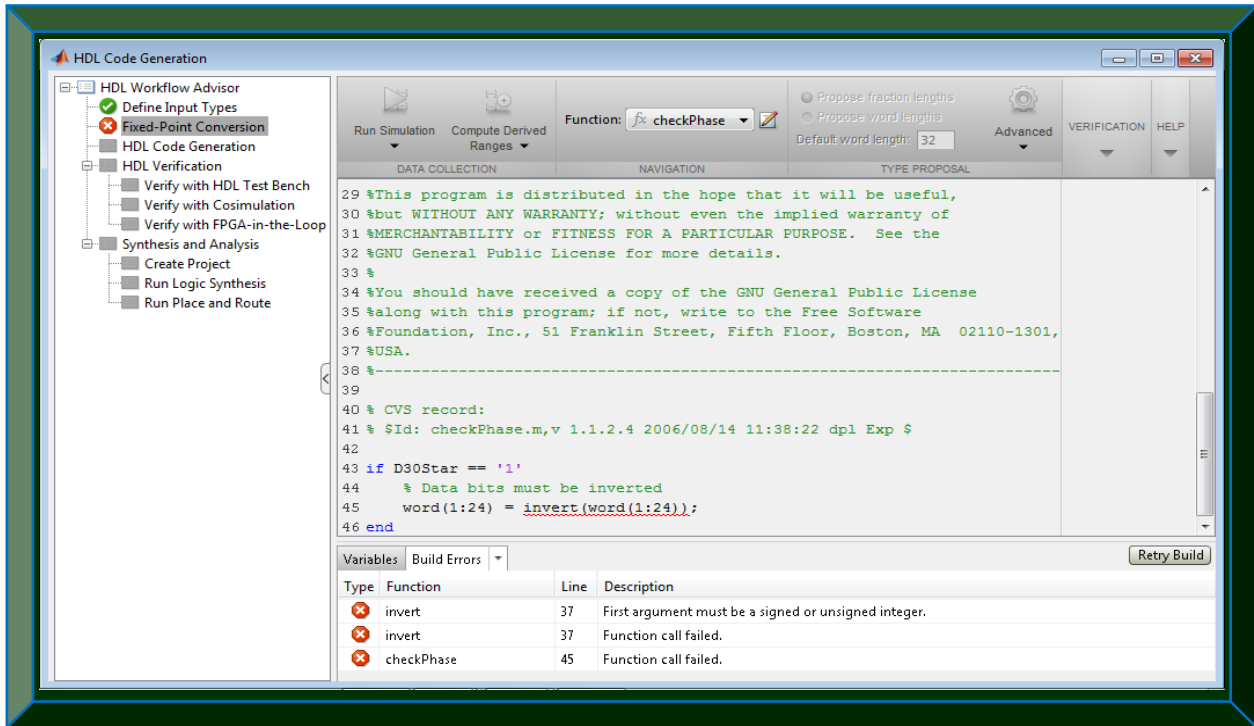


Figure 9.18: Illustration of some conversion issues in the checkPhase.m algorithm

9.4.6 clkSin.m: provides real and imaginary output for use by cart2utm.m

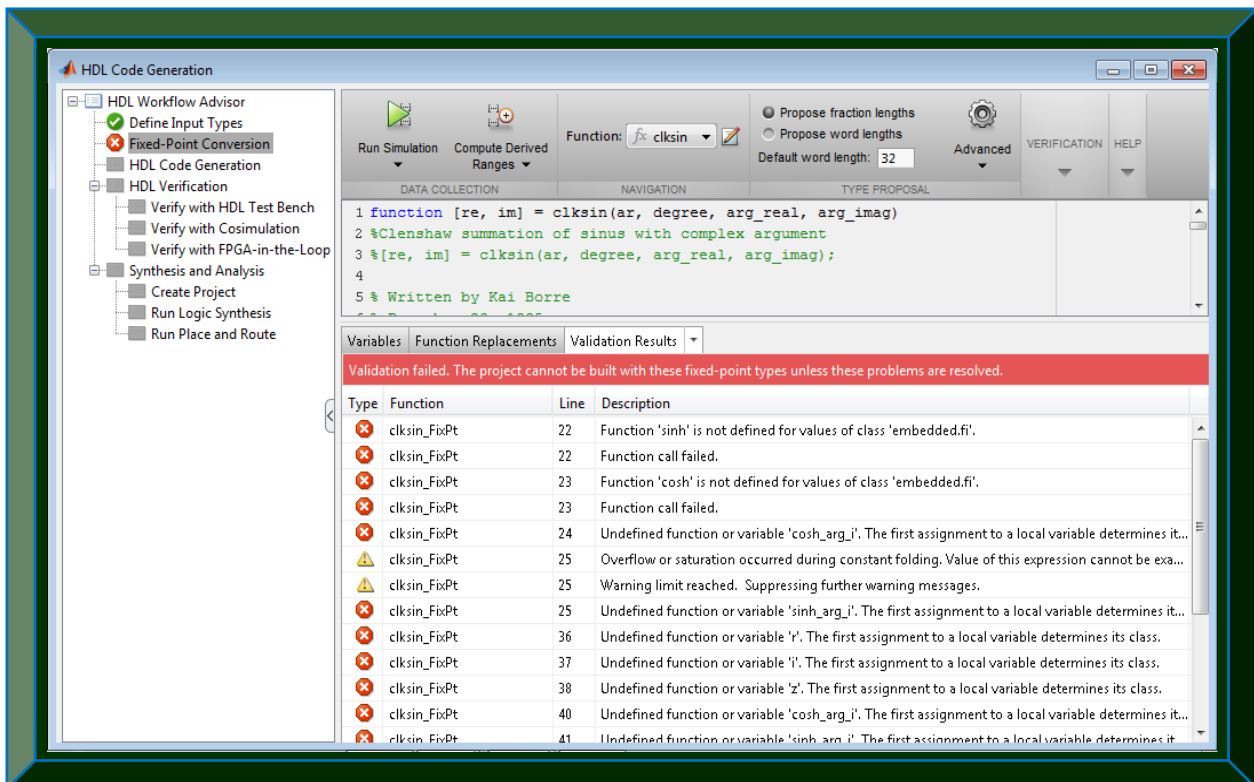


Figure 9.19: Illustration of some conversion issues in the clkSin.m algorithm

9.4.7 dms2mat.m: splits deg2dms.m output to hours, minutes, seconds and milli-seconds

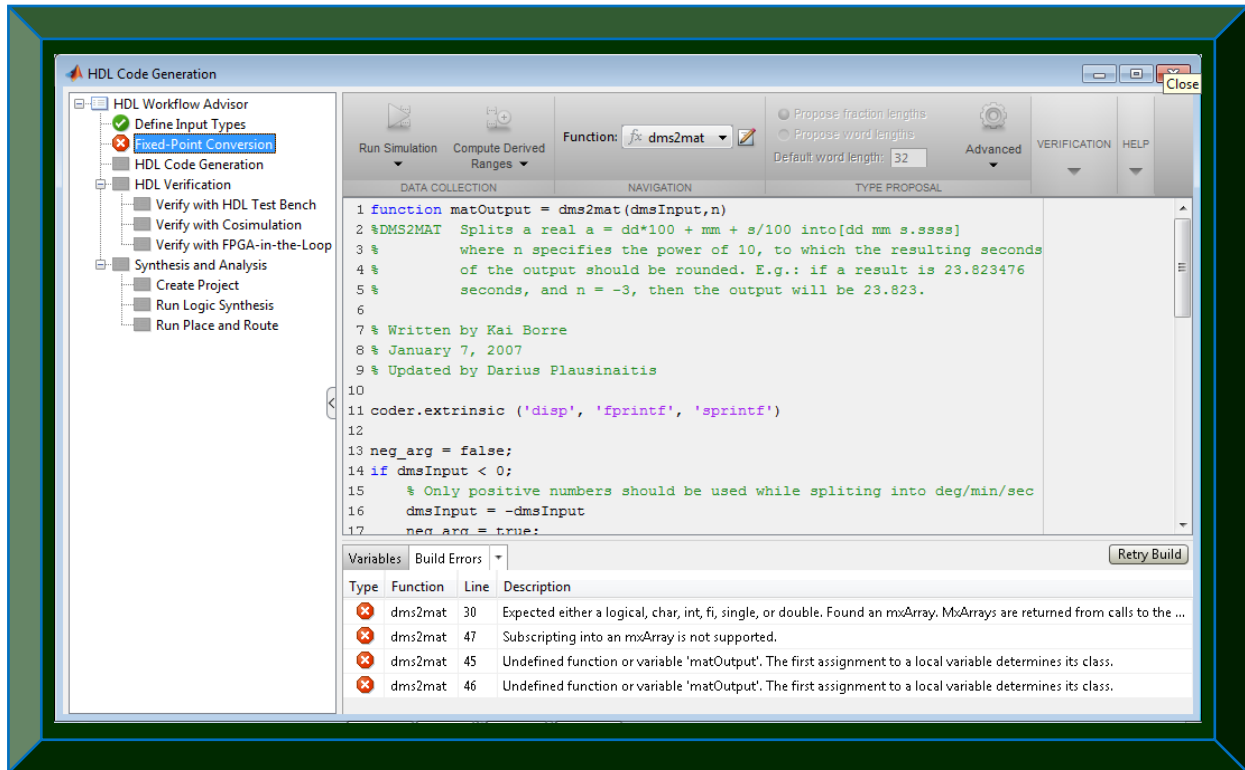


Figure 9.20: Illustration of some conversion issues in the dms2mat.m algorithm

9.4.8 e_r_corr.m: corrects errors in GPS satellites orbital positions

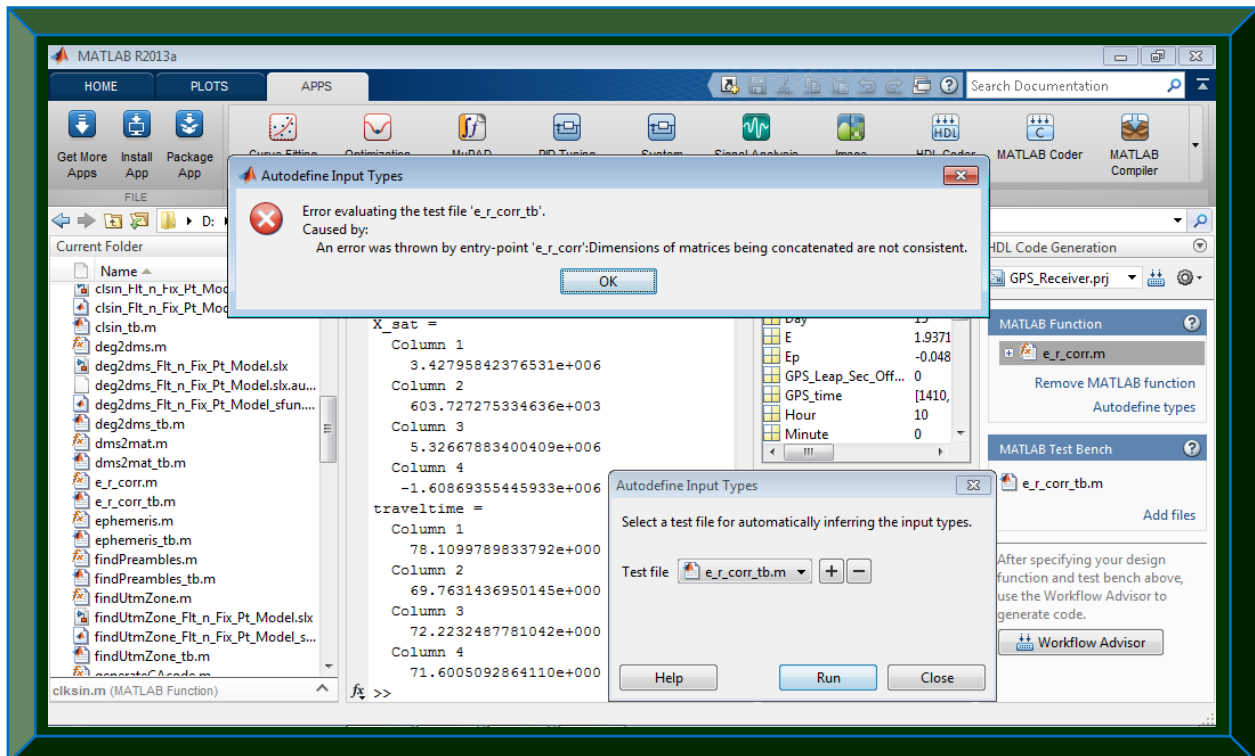


Figure 9.21: Illustration of some conversion issues in the e_r_corr.m algorithm

9.4.9 ephemeris.m: decodes ephemeris data and time of week from the navigation data

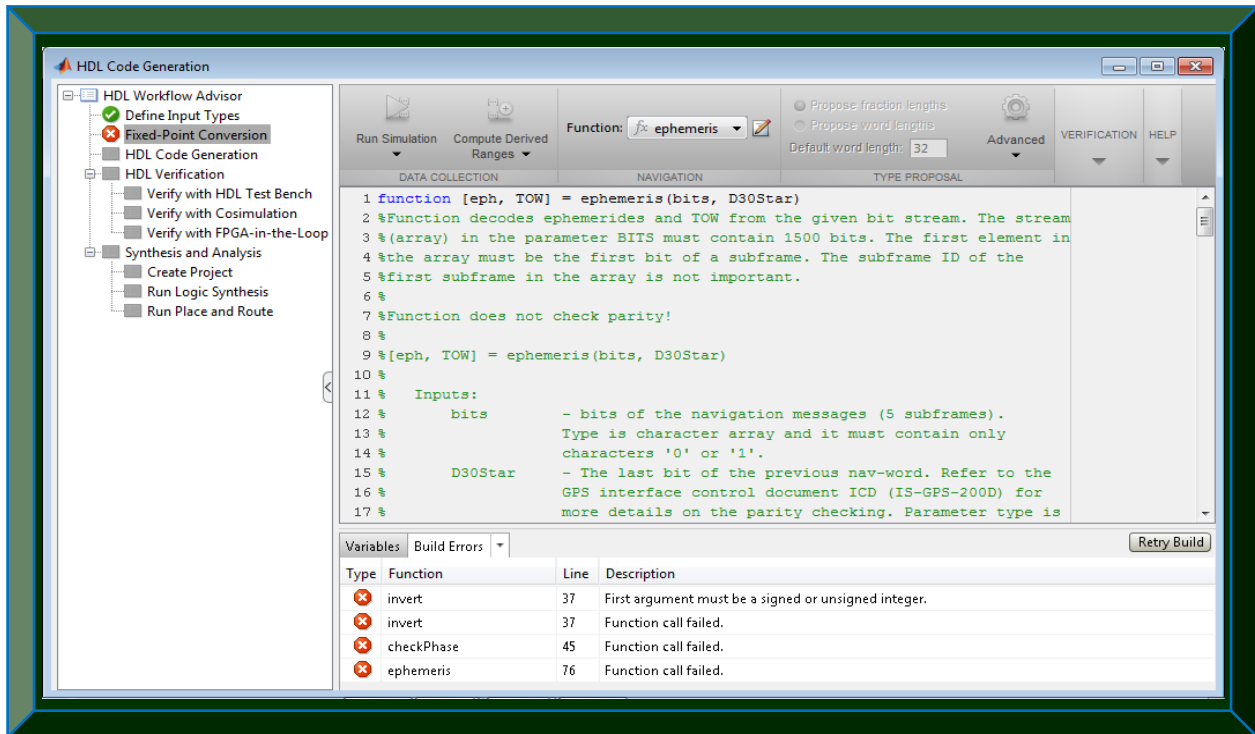


Figure 9.22: Illustration of some conversion issues in the ephemeris.m algorithm

9.4.10 findPreambles.m: deduces first sub-frame or the start of the navigation message

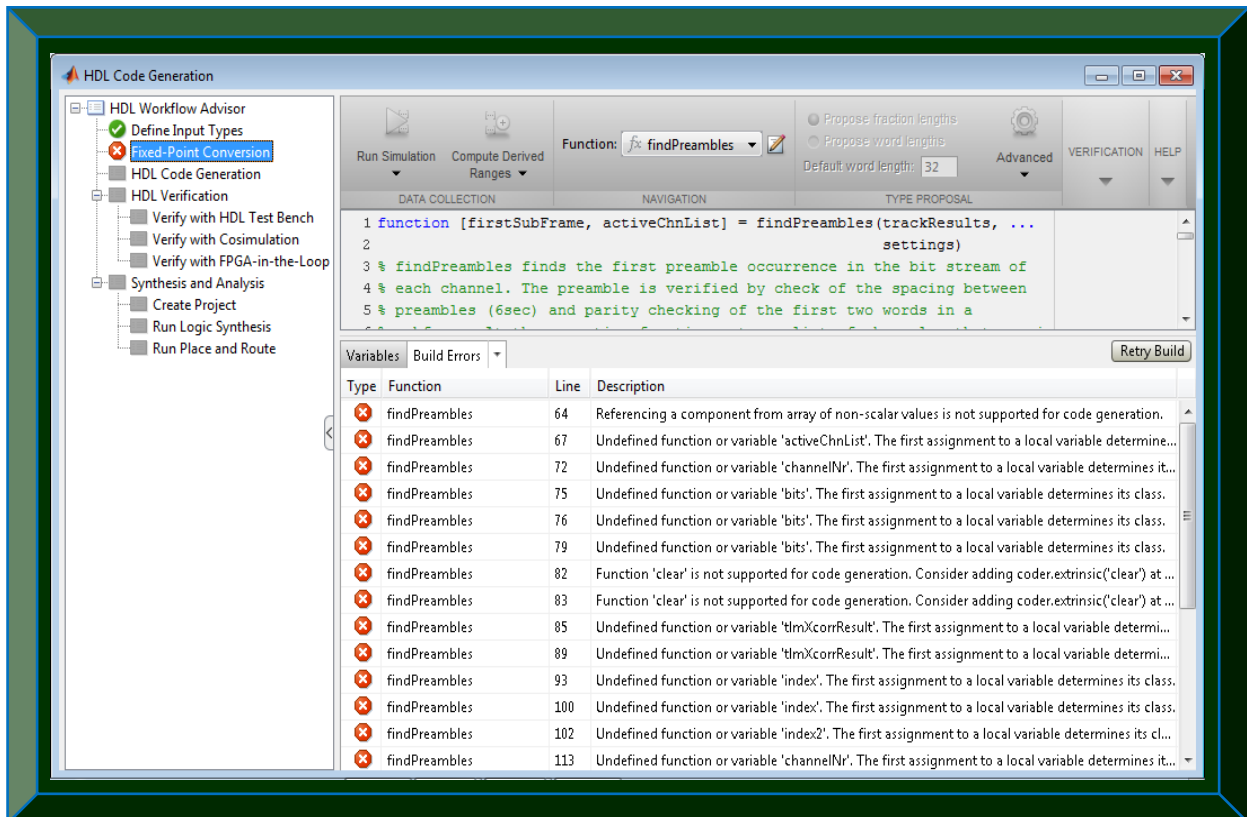


Figure 9.23: Illustration of some conversion issues in the findPreambles.m algorithm

9.4.11 generateCAcode.m: creates the 1023 C/A code phases for any 32 GPS PRN code

The figure consists of three vertically stacked screenshots from the HDL Code Generation tool.

The top screenshot shows the MATLAB function `generateCAcode` being executed. The code defines a function that takes a PRN number as input and returns a vector of C/A code chips. The output shows simulation results for PRN 1022 and 1023, but ends with a red error message: "Error using generateCAcode_FixPt (line 61) Index exceeds matrix dimensions. Index value 0 exceeds valid range [1-1023] of array g1."

The middle screenshot shows the "Generate synthesizable HDL code from the fixed-point MATLAB code" dialog box. The "Language" is set to VHDL, and options for "Check HDL conformance", "Generate HDL", and "Generate EDA scripts" are checked. A "Run" button is visible at the bottom right.

The bottom screenshot is a web browser displaying the "HDL Code Generation Check Report for 'generateCAcode_FixPt'". The report is dated 2014-04-28 23:21:28 and contains a table of errors:

Function Location	Level	Description
generateCAcode_FixPt:83		'g2': Error: variable-size matrix type is not supported for HDL code generation.
generateCAcode_FixPt:81		'varargin_1': Error: variable-size matrix type is not supported for HDL code generation.
generateCAcode_FixPt:81		'varargin_1': Error: variable-size matrix type is not supported for HDL code generation.
generateCAcode_FixPt:81		'varargin_1': Error: variable-size matrix type is not supported for HDL code generation.
generateCAcode_FixPt:81		'varargin_1': Error: variable-size matrix type is not supported for HDL code generation.
generateCAcode_FixPt:81		'var1': Error: variable-size matrix type is not supported for HDL code generation.
generateCAcode_FixPt:81		'g2': Error: variable-size matrix type is not supported for HDL code generation.
generateCAcode_FixPt:81		'var1': Error: variable-size matrix type is not supported for HDL code generation.
generateCAcode_FixPt:83		'b0': Error: variable-size matrix type is not supported for HDL code generation.
generateCAcode_FixPt:83		'b0': Error: variable-size matrix type is not supported for HDL code generation.
generateCAcode_FixPt:83		'b': Error: variable-size matrix type is not supported for HDL code generation.

Figure 9.24: Illustration of some conversion issues in the generateCAcode.m algorithm

9.4.12 `gps_time.m`: extracts mean GPS week number and second from ephemeris data

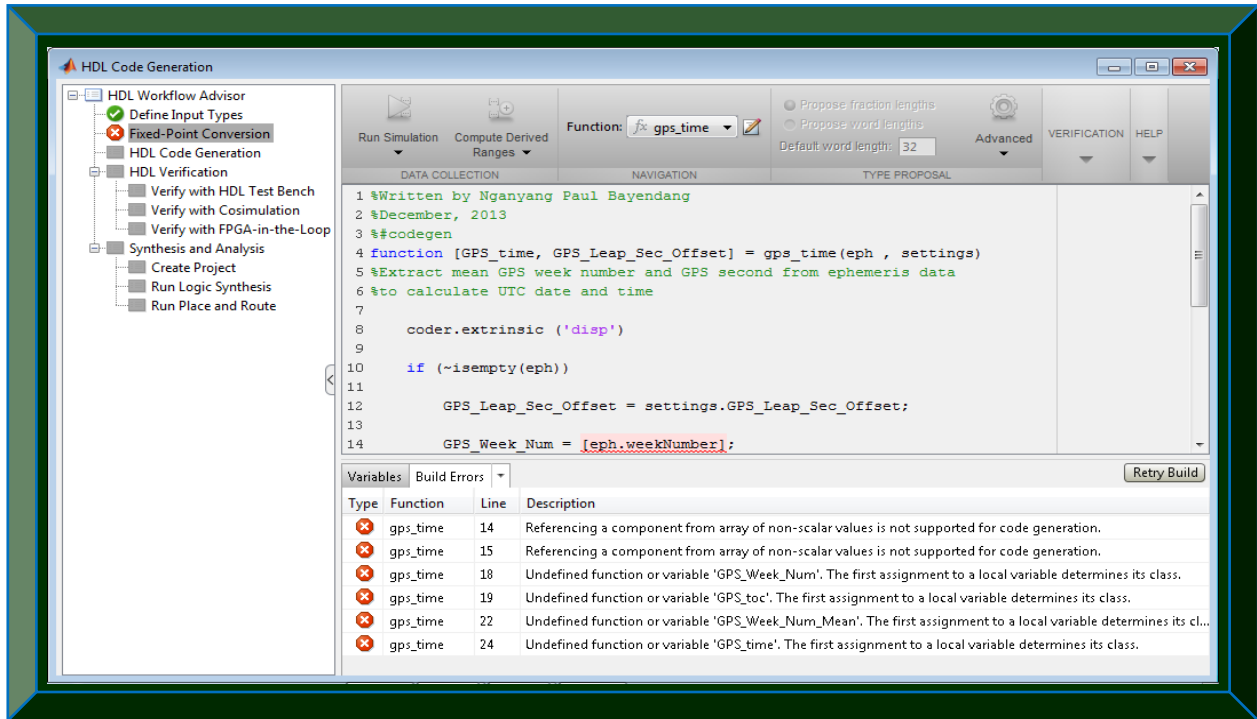


Figure 9.25: Illustration of some conversion issues in the `gps_time.m` algorithm

9.4.13 `gps2utc.m`: computes UTC date and time from the extracted GPS time

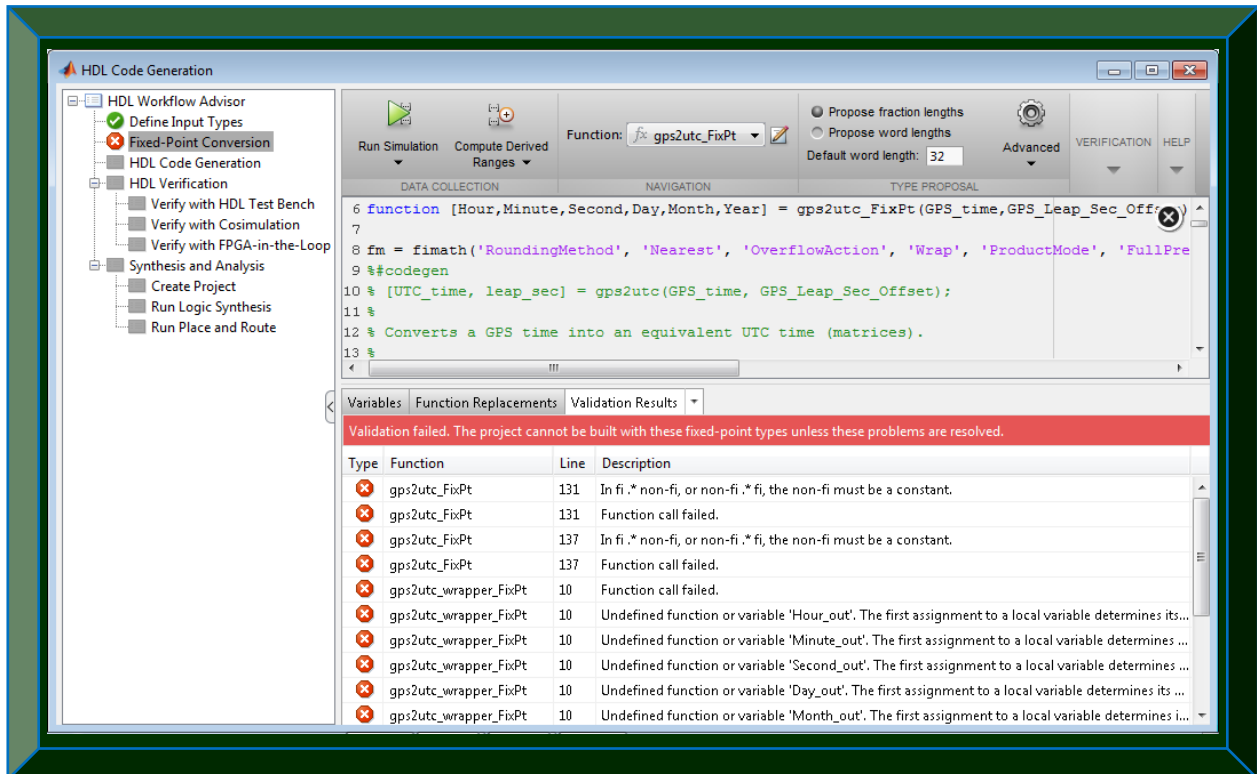


Figure 9.26: Illustration of some conversion issues in the `gps2utc.m` algorithm

9.4.14 `init_tracking.m`: prepares the acquired GPS data for tracking

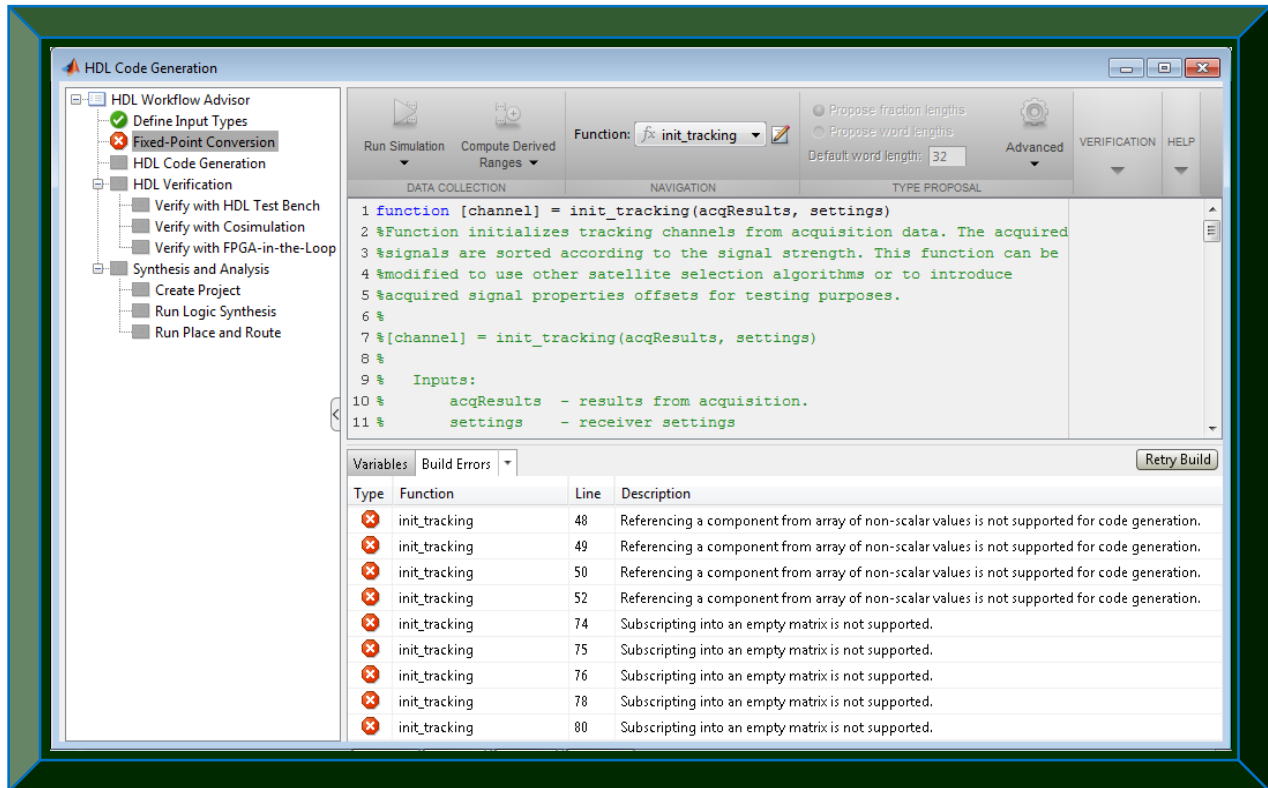


Figure 9.27: Illustration of some conversion issues in the `init_tracking.m` algorithm

9.4.15 `invert.m`: inverts applicable navigation bits for use by `checkPhase.m`

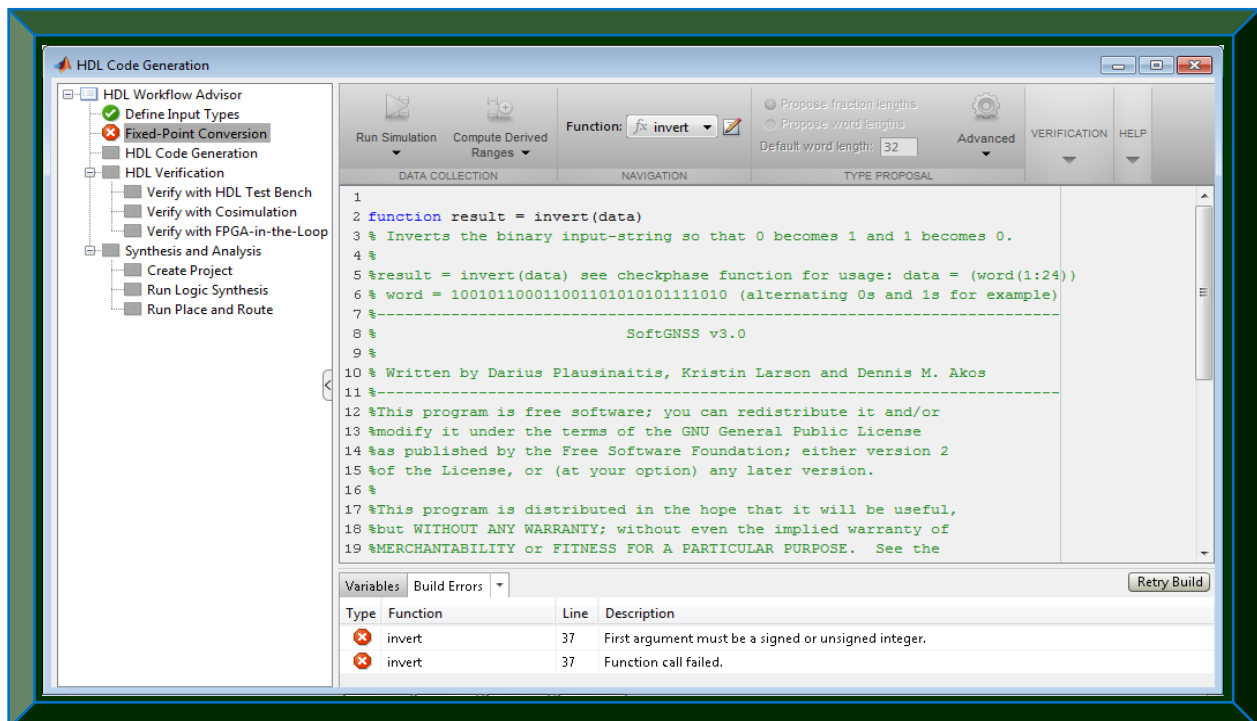


Figure 9.28: Illustration of some conversion issues in the `invert.m` algorithm

9.4.16 leastSquarePos.m: computes GPS receiver position using least-squares approach

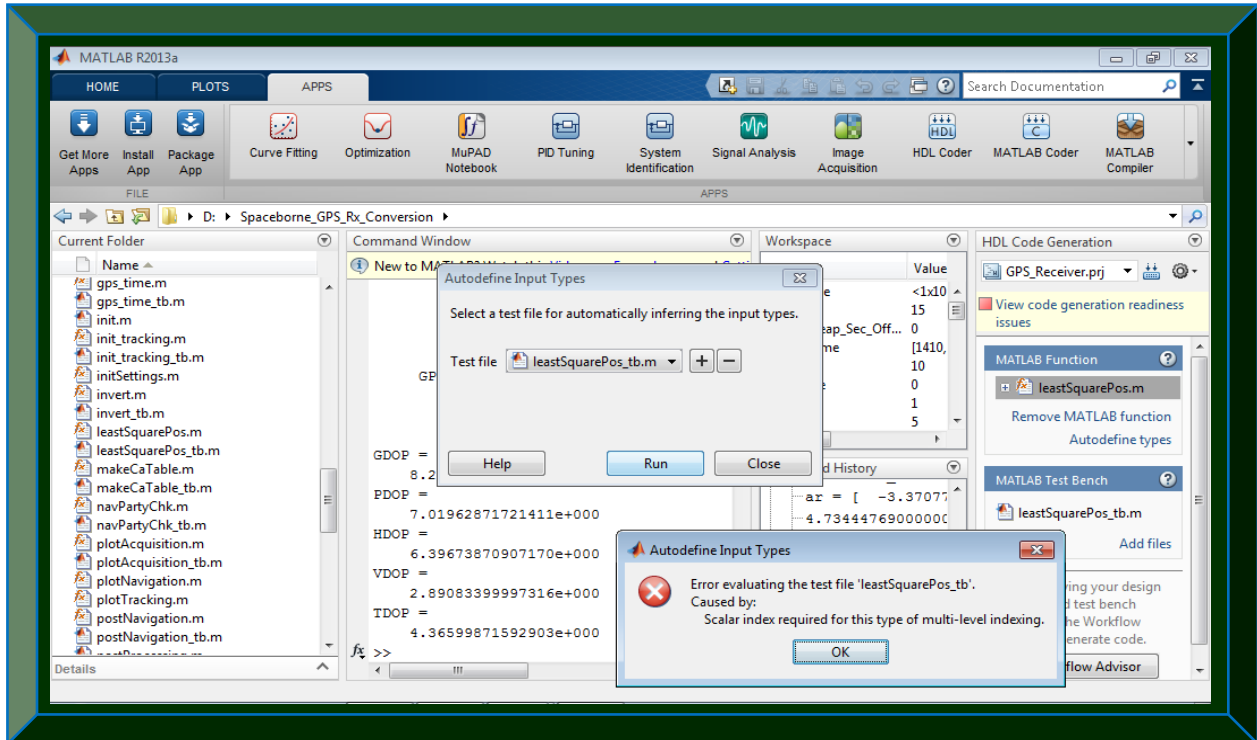


Figure 9.29: Illustration of some conversion issues in the leastSquarePos.m algorithm

9.4.17 makeCaTable.m: pre-computes / creates a lookup table for all 32 GPS PRN codes

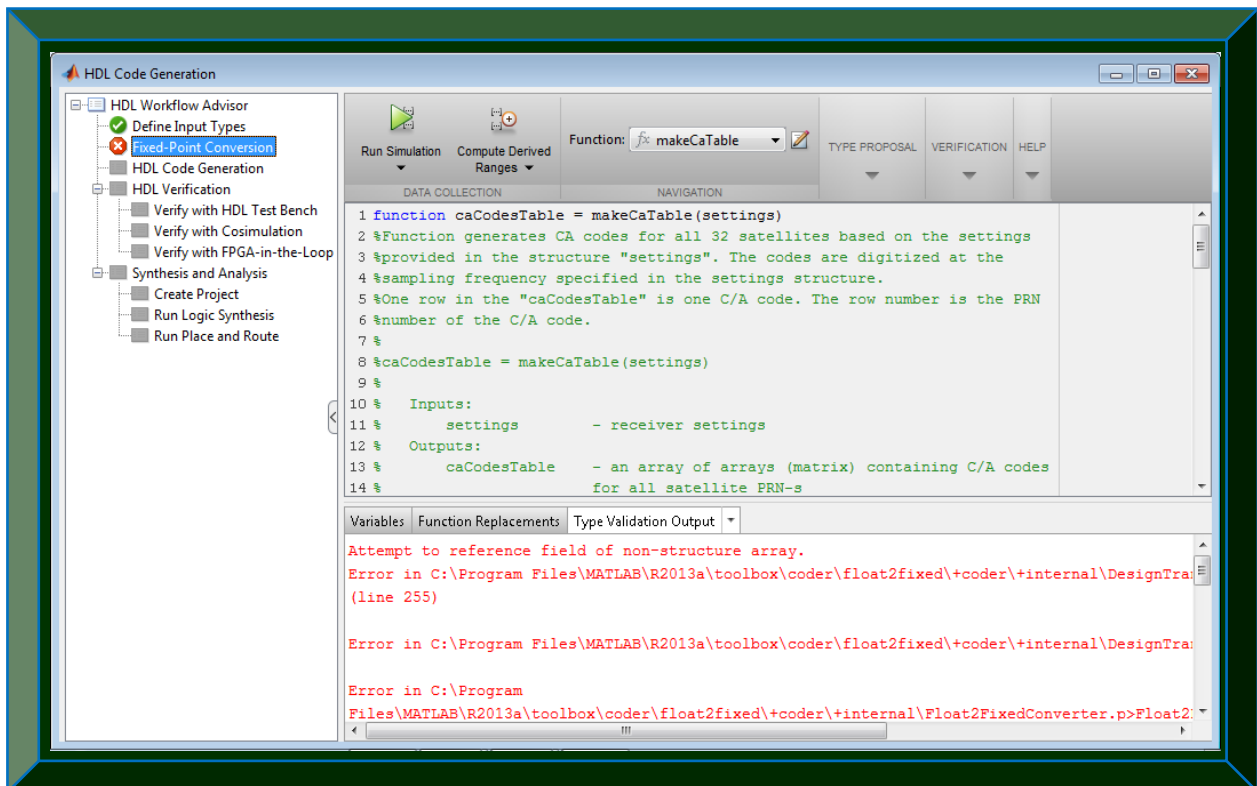


Figure 9.30: Illustration of some conversion issues in the makeCaTable.m algorithm

9.4.18 navPartyChk.m: checks the parity of the decoded navigation data sub-frames

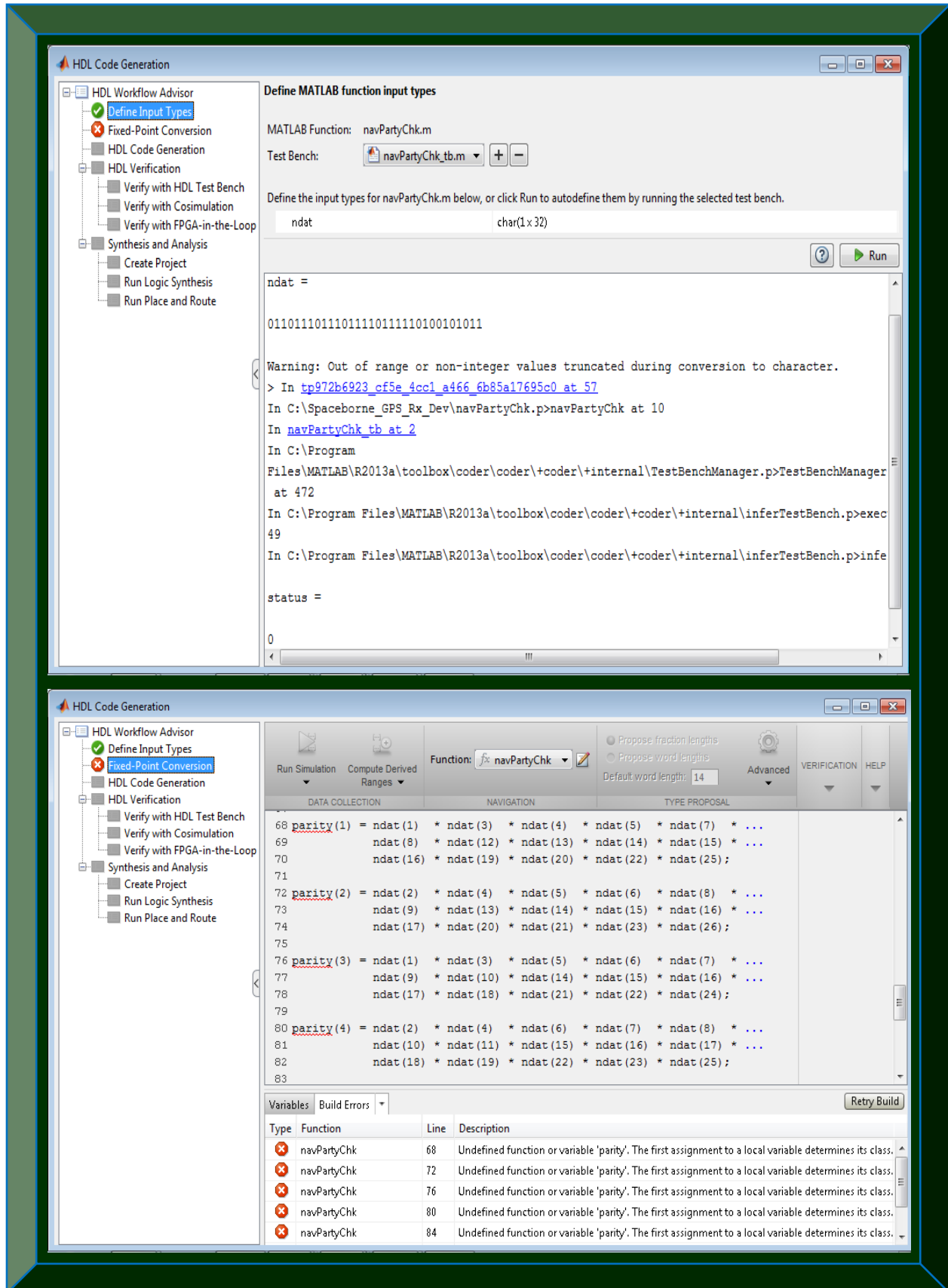


Figure 9.31: Illustration of some conversion issues in the `navPartyChk.m` algorithm

9.4.19 postNavigation.m: performs post-processing of the decoded GPS data

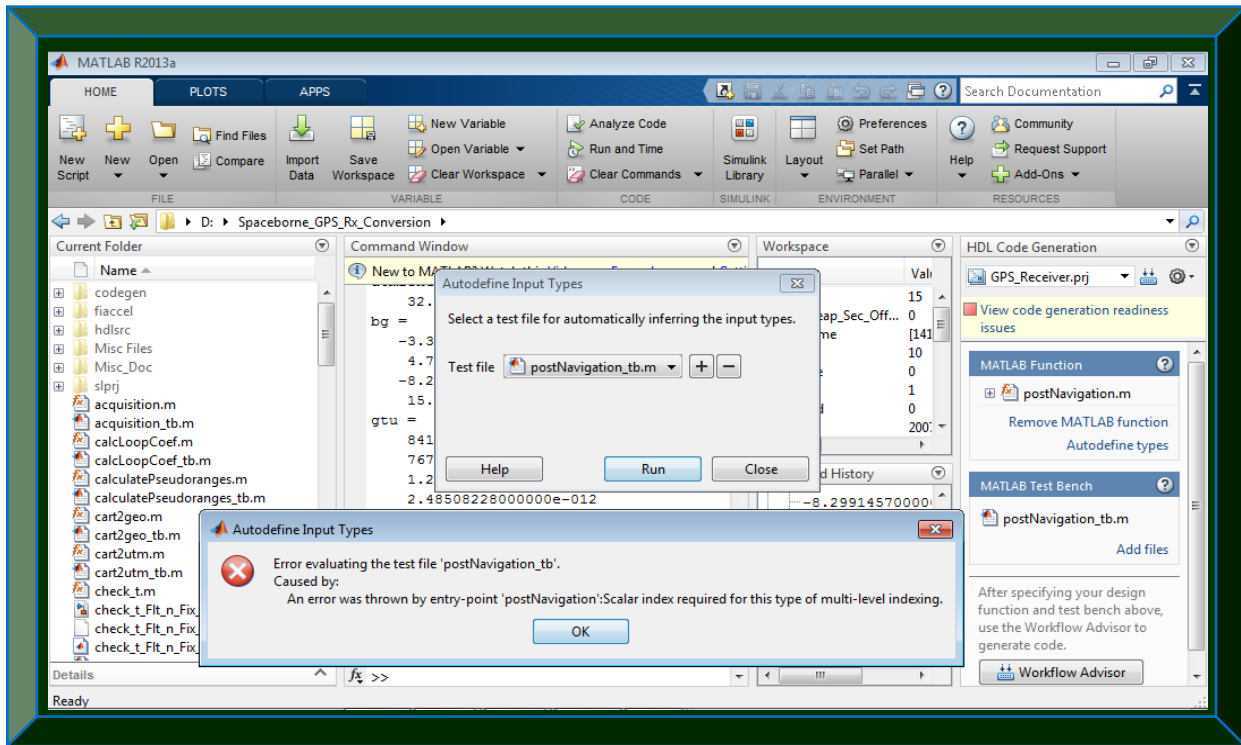


Figure 9.32: Illustration of some conversion issues in the postNavigation.m algorithm

9.4.20 satpos.m: compute GPS satellites position (crashes Matlab R2013a)

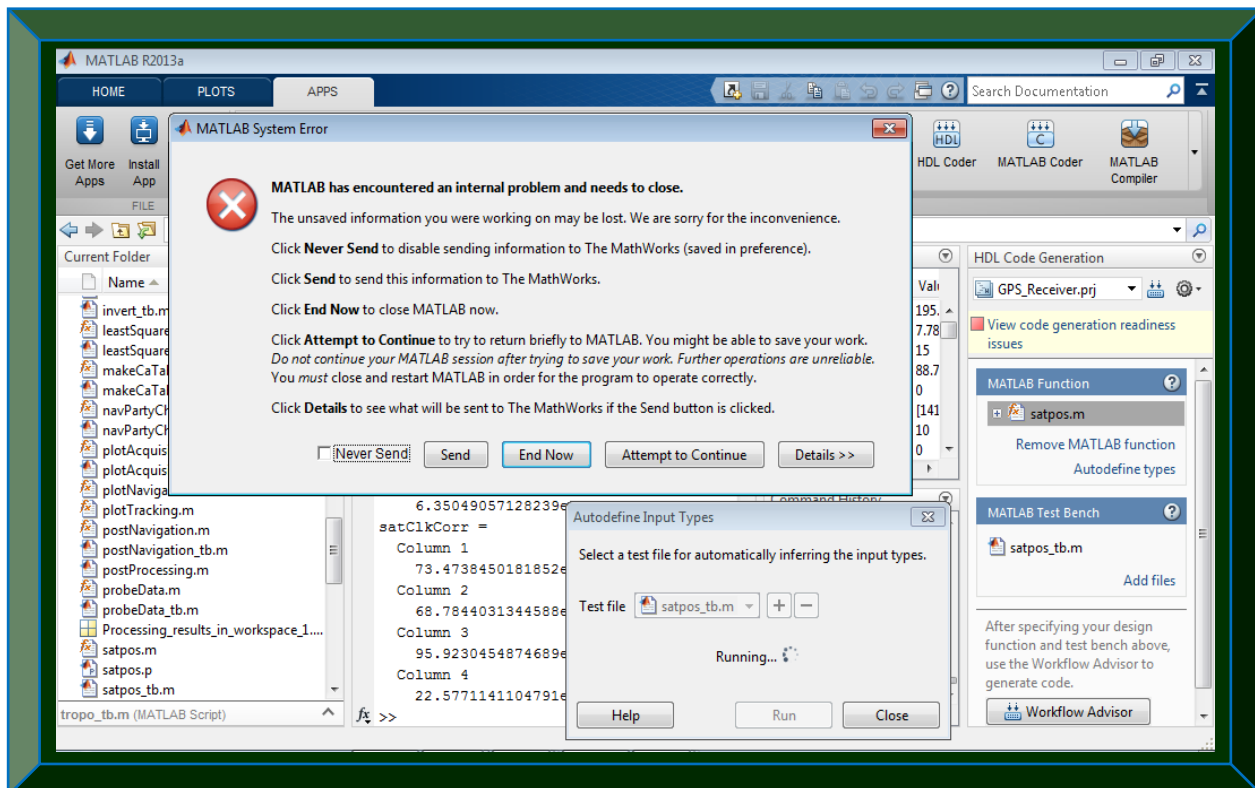


Figure 9.33: Illustration of some conversion issues in the satpos.m algorithm

9.4.21 showChannelStatus.m: display the acquisition and tracking results in a table

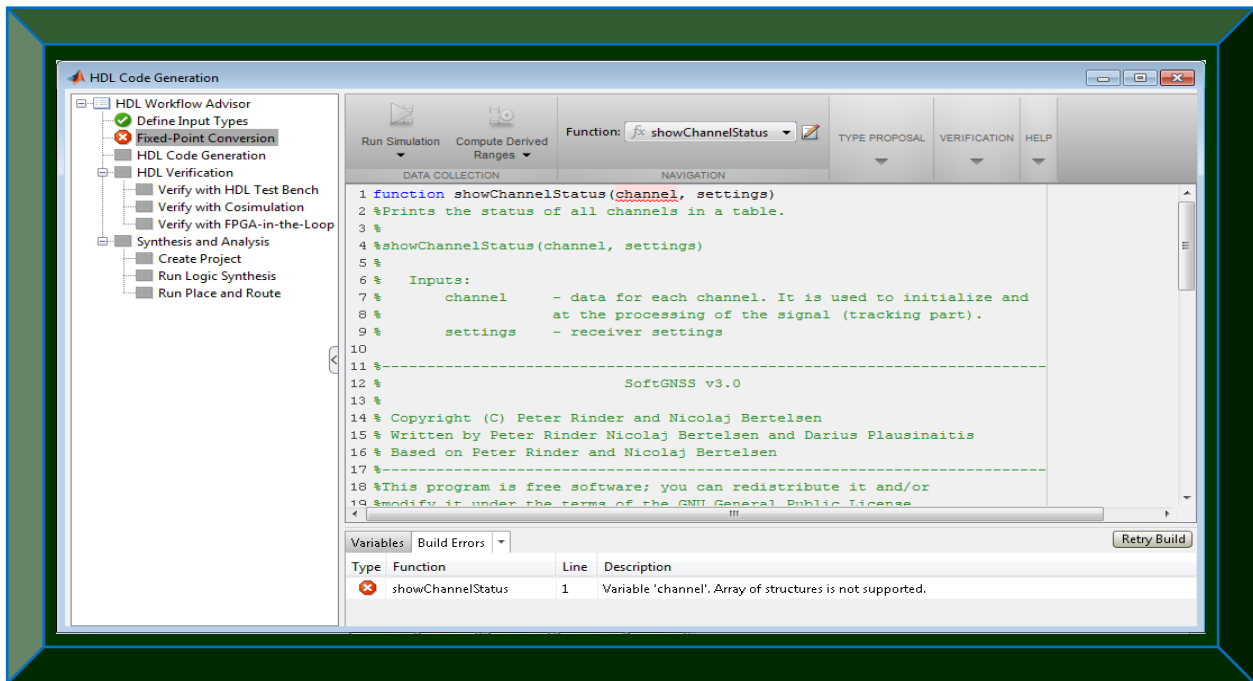


Figure 9.34: Illustration of some conversion issues in the showChannelStatus.m algorithm

9.4.22 togeod.m: calculates geodetic coordinates from Cartesian coordinates

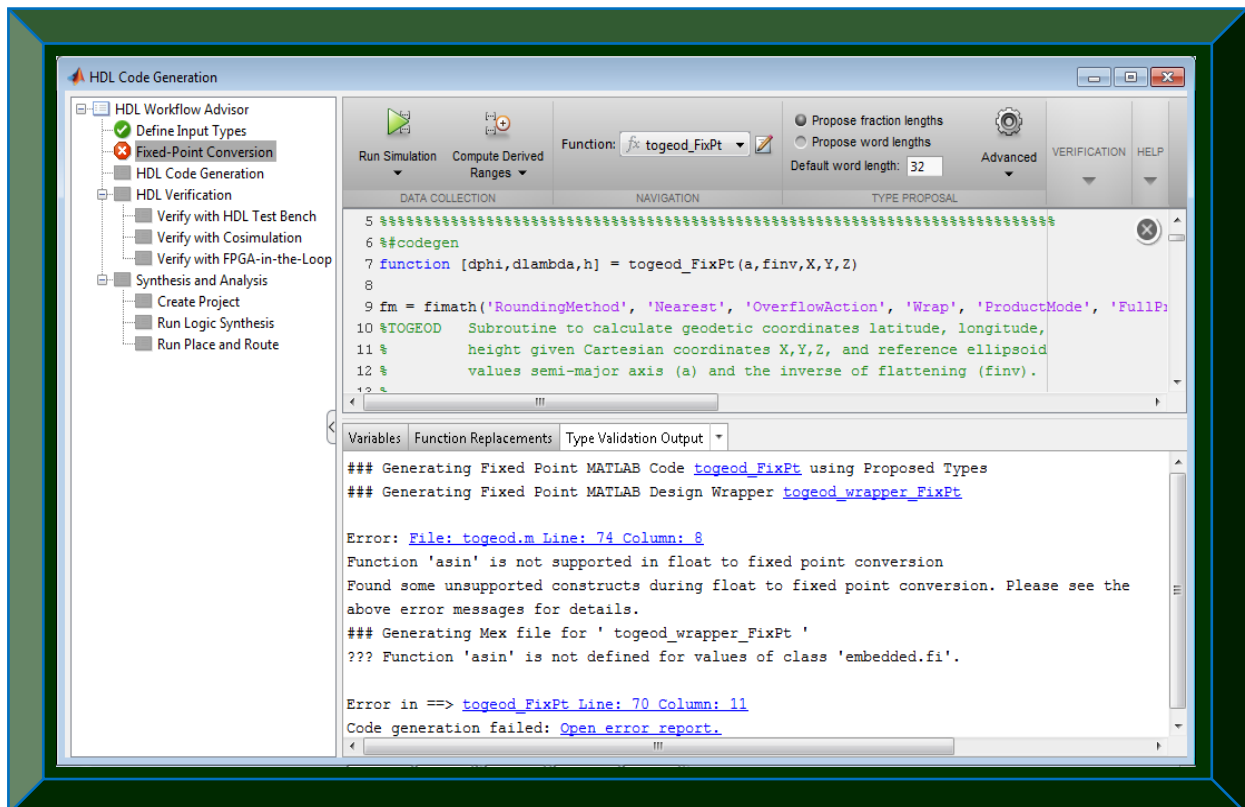


Figure 9.35: Illustration of some conversion issues in the togeod.m algorithm

9.4.23 topocent.m: transforms changing position coordinates to topocentric coordinates

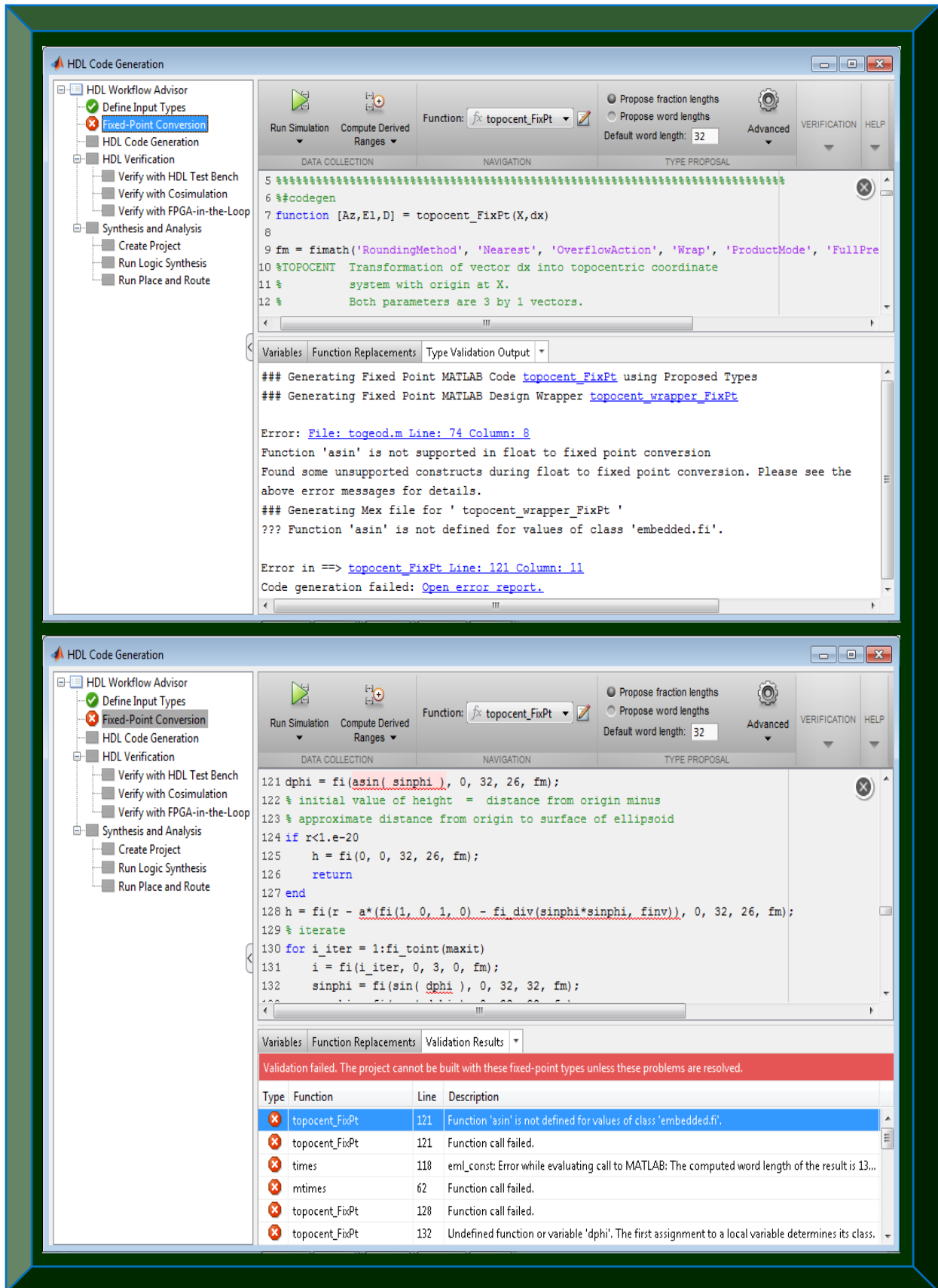


Figure 9.36: Illustration of some conversion issues in the topocent.m algorithm

9.4.24 tracking.m: tracks the acquired GPS data to extract navigation bit streams

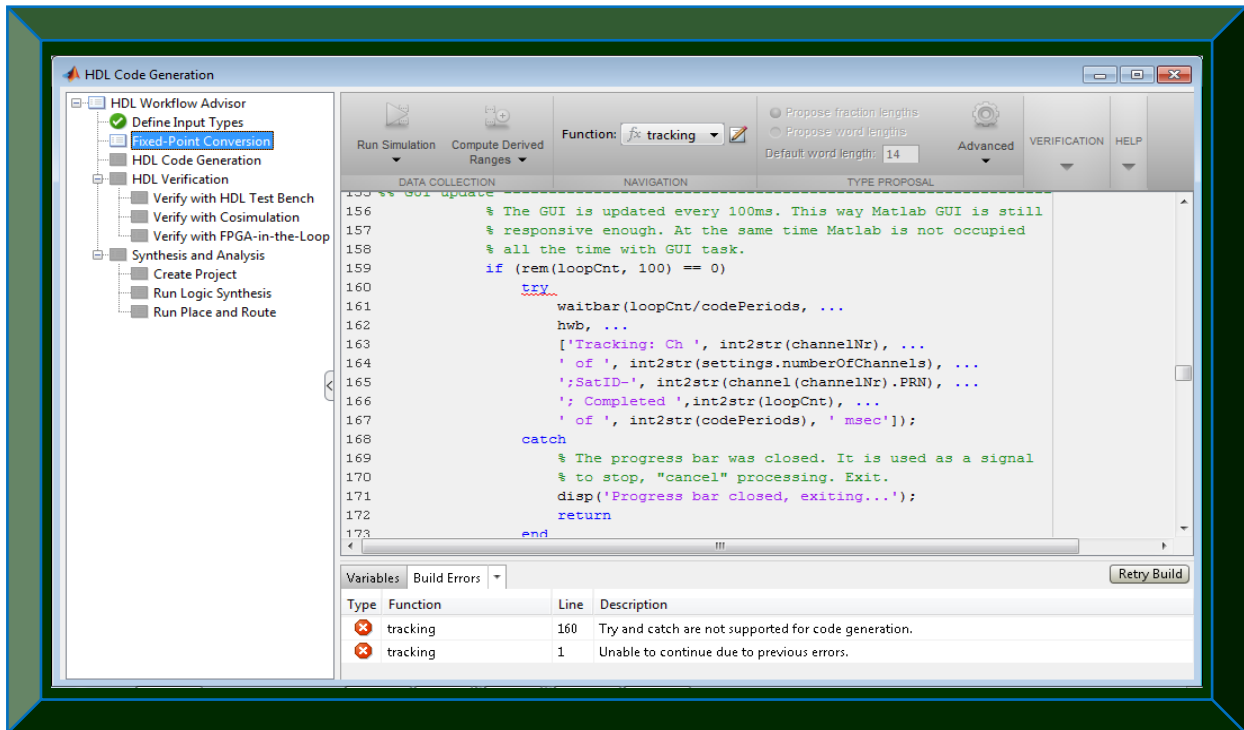


Figure 9.37: Illustration of some conversion issues in the tracking.m algorithm

9.4.25 tropo.m: corrects tropospheric errors in pseudo-ranges and carrier phases

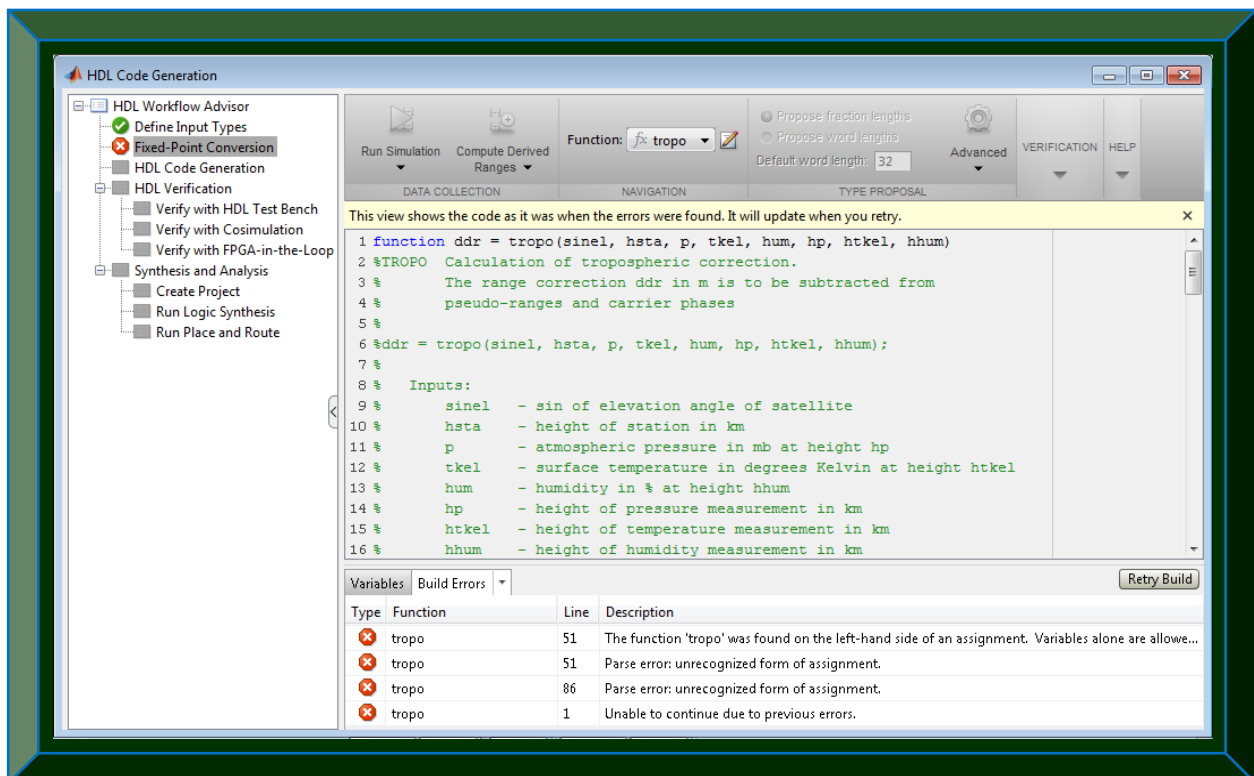


Figure 9.38: Illustration of some conversion issues in the tropo.m algorithm

9.4.26 twosComp2dec.m: converts a two-complement binary number

The figure consists of two screenshots of the HDL Code Generation tool interface, showing the workflow and the results of code generation for the `twosComp2dec.m` algorithm.

Top Screenshot: The HDL Workflow Advisor on the left shows the 'Fixed-Point Conversion' step selected. The main window displays the MATLAB function code for `twosComp2dec`. Below the code, the 'Type Validation Output' tab shows a warning: 'Warning: Type not found for 'binaryNumber''. Below that, it indicates the generation of fixed-point MATLAB code and a design wrapper. An error is reported: 'Error: File: twosComp2dec.m Line: 43 Column: 30 Function 'mpower' is not supported in float to fixed point conversion Found some unsupported constructs during float to fixed point conversion. Please see the above error messages for details.'

Bottom Screenshot: The HDL Workflow Advisor on the left shows the 'Fixed-Point Conversion' step selected. The main window displays the MATLAB function code for `twosComp2dec`. Below the code, the 'Verification Output' tab shows the results of a floating point simulation and a fixed point simulation. The floating point simulation shows the input `binaryNumber = 11010101101011010101` and the output `intNumber = -2.773547000000000e+006`. The fixed point simulation shows the same input and output, followed by the message 'Beginning fixed point error analysis'.

Figure 9.39: Illustration of some conversion issues in the `twosComp2dec.m` algorithm

REFERENCES

- Akopian, D., Sagiraju, P.K., & Turunen, S. 2006. Performance of two-stage massive correlator architecture for fast acquisition of GPS signals. *2006 IEEE Region 5 Conference*: 137 – 140, April 7 – 9. San Antonio, Texas, USA.
- Alaqeeli, A., J Starzyk, J. & Graas, F.V. 2003. Real-time acquisition and tracking for GPS receivers. *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03*, 4: 500 – 503, May 25 – 28. Bangkok, Thailand.
- Altera. 2003. Gold code generator reference design. <http://www.altera.co.jp/literature/an/an295.pdf>
[Date accessed: 27 April 2013]
- AMSAT-UK. 2014. CubeSat information. <http://amsat-uk.org/satellites/satellite-missions-database/>
[Date accessed: 28 June 2014]
- Arif, T.T. (ed.). 2010. *Aerospace Technologies Advancement*. Vukovar, Croatia: Intech.
- ARINC Engineering Services LLC. 2004. Navstar Global Positioning System: Interface Specification (IS-GPS-200D). <http://www.gps.gov/technical/icwg/IS-GPS-200D.pdf>.
[Date accessed: 20 May 2013]
- Avanzi, A. & Tortora, P. 2010. Design and implementation of a new spaceborne FPGA-based dual frequency GPS and Galileo software defined receiver. *2010 5th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC)*, 1 – 7, December 8 – 10. Noordwijk, The Netherlands.
- Bauer, F. H., Hartman, K., How, J. P., Bristow, J., Weidow, D., & Busse, F. 1999. Enabling spacecraft formation Flying through Spaceborne GPS and Enhanced Automation Technologies. *ION-GPS Conference*, 15 September 1999. Nashville, Tennessee, USA.
- Bazzi, O., Torguet, R., Bruneel, C. & Kastellk, J. C. 1993. A compact acousto-optic correlator for rapid GPS signal processing. *Proceedings of IEEE Ultrasonic's Symposium*, 1: 491 - 493, 31 October – 3 November 1993. Baltimore, Maryland, USA.
- Bein, A. L. 2008. NPS-SCAT: The construction of NPS' first prototype cubesat. Master's thesis, Naval Postgraduate School, California. <http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA488676>
[Date accessed: 18 April 2012]
- Bernhardt, M., Borth, A., Brennan, R., Galgana, E., Gangar, P., Kemis, A., Lutzenhiser, N., Moravec, K., Mzali, S., Nazari, Z., Ross, J., & Shin-White, E. 2009. Rapid terrestrial imaging cubesat constellation. University of Washington. http://www.agi.com/downloads/partners/edu/UW_PDR_2009_paper.pdf
[Date accessed: 18 April 2012]
- Borio, D., O'Driscoll, C. & Fortuny, J. 2013. Jammer impact on Galileo and GPS receivers. *2013 International Conference on Localisation and GNSS (ICL-GNSS)*: 1 – 6, June 25–27. Turin, Italy.
- Borio, D., Sokolova, N. & Lachapelle, G. 2010. Doppler measurement accuracy in standard and high sensitivity global navigation satellite system receivers. *2011 IET Radar, Sonar and Navigation*, 5(6), 657 – 665.

- Borre, K., & Akos, D. 2005. A software-defined GPS and Galileo receiver: single-frequency approach. *Proceedings of the 18th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2005)*: 1632 – 1637, September 13 – 16. Long Beach, California, USA.
- Borre, K., Akos, D.M., Bertelsen, N., Rinder, P. & Jensen, S.H. 2007. *A Software-defined GPS and Galileo receiver: a single-frequency approach*. New York: Birkhauser.
- Borre, K. & Strang, G. 2012. *Algorithms for global positioning*. Massachusetts: Wellesley-Cambridge Press.
- Braasch, M.S. & Van Dierendonck, A. J. 1999. GPS Receiver Architectures and measurements. *Proceedings of the IEEE*, 87(1), 48 – 64, January 1999.
- California Polytechnic State University. 2009. CubeSat design specification revision 12. http://www.srl.utu.fi/AuxDOC/tke/radmon/cubesat_standard.pdf [Date accessed: 20 July 2014]
- California Polytechnic State University. n.d. CubeSat Past Launches. <http://cubesat.atl.calpoly.edu/index.php/missions/past-launches> [Date accessed 19 April 2012]
- CCII. n.d. Remote monitor and controller. <http://www.cci.co.za/products/pdf/rmc.pdf> [Date accessed: 07 July 2013]
- Celestrak. 2013. Supplemental two-line element sets. <http://celestrak.com/NORAD/elements/supplemental> [Date accessed: 5 April 2013].
- Choi, E., Yoon, J., Lee, B., Park, S. & Choi, K. n.d. Development Of spaceborne GPS receiver with real-time orbit determination using unscented Kalman filter. *IAC-09.B2.5.5*.
- Chung, B., Chien, C., Samueli, H. & Jain, R. 1993. Performance analysis of an all-digital BPSK direct-sequence spread-spectrum IF receiver architecture. *IEEE Journal on Selected Areas in Communications*, 11(7): 1096 – 1107, September.
- CubeSat. 2014. CubeSat design specification revision 13. http://www.cubesat.org/images/developers/cds_rev13_final.pdf [Date accessed: 25 June 2014].
- Díaz, J.G., García, J.G. & Roncagliolo, P.A. 2012. An FPGA implementation of a data-bit asynchronous GPS/GLONASS correlator. *2012 Argentine School of Micro-Nanoelectronics, Technology and Applications (EAMTA)*: 27 – 33, August 9 and 10. Cordoba, Argentina.
- Djebouri, D., Djebbari, A. & Djebbouri, M. 2006. Robust GPS satellite signal acquisition using lifting wavelet transform. *Radio Engineering*, 15(1), April. Czech.
- Doberstein, D. 2012. *Fundamentals of GPS receivers: A hardware approach*. New York: Springer.
- Drake, S.P. 2002. Converting GPS Coordinates to Navigation Coordinates. DSTO Electronics and Surveillance Research Laboratory: Edinburgh, Australia. <http://www.dsto.defence.gov.au/publications/2443/DSTO-TN-0432.pdf> [Date accessed: 31 December 2013]
- DSP-FPGA. 2006. The Design and implementation of a GPS receiver channel: An FPGA plus DSP creates a powerful combination. <http://dsp-fpga.com/article-id/?79> [Accessed: 10 April 2013]

- Dubey, G. 2013. Fixed-point design in Matlab and Simulink. <http://www.matlabexpo.com/in/2013/proceedings/fixed-point-design-in-matlab-and-simulink.pdf> [Date accessed: 17 January 2014]
- Eddine, D.D., & Mohamed, D. 2013. An optimal method using wavelet packet de-noising for a GPS of an observation satellite. *2013 Saudi International Electronics, Communications and Photonics Conference (SIEPCPC 2013)*. April 27 – 30. Riyadh, Saudi Arabia.
- Eecatalog.com. 2011. Xilinx space-grade Virtex-5QV FPGA in production. <http://eecatalog.com/military/2011/07/21/xilinx-space-grade-virtex-5qv-fpga-in-production/> [Date accessed: 17 September 2012]
- Engel, F., Mumford, P., Parkinson, K., Rizos, C. & Heiser, G. 2004. An open GNSS receiver platform architecture. *The 2004 International Symposium on GNSS / GPS*, December 6 – 8. Sydney, Australia
- El-Rabbany, A. 2002. *Introduction to GPS: the global positioning system*. Norwood, Massachusetts: Artech House.
- El-Rayis, A.O., Arslan, T. & Erdogan, A.T. 2010. A processing engine for GPS correlation. *IEEE Eighth Symposium on Application Specific Processors (SASP)*, June 13 – 14. Anaheim, California, USA.
- Falletti, E. & Motella B. 2012. Combination of squared correlators for multipath mitigation in safety of life global navigation satellite systems receivers. *IET Radar Sonar Navigation*, 6(7): 611 – 619. Italy.
- Feng, Z.W., Gai-yun, W. & Lei, C. 2013. Research on acquisition algorithms of GPS receiver. *2013 Third international Conference on Intelligent System Design and Engineering Applications (ISDEA)*: 1148 – 1151, January 16 – 18. Hong-Kong, China.
- França Jr., J. A. & Morgado, J. A. 2010. Real-time implementation of a low-cost INS/GPS system using xPC target. *Journal of Aerospace Engineering, Sciences and Applications*, 2(3), September - December. Rio de Janeiro, Brazil.
- Gerein, N., & Brown, A. 2001. Modular GPS software radio architecture. *Proceedings of the 14th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GPS 2001)*, 2048 – 2058, September 11 – 14. Salt Lake City, Utah, USA.
- Gerner, J.L., Issler, J.L., Laurichesse, D., Mehlen, C. & Wilhelm, N. 2000. TOPSTAR 3000 – An enhanced GPS receiver for space applications. ESA bulletin 104 — November. <http://www.esa.int/esapub/bulletin/bullet104/gerner104.pdf> [Date accessed: 17 March 2013]
- Gill, E., Montenbruck, O., Arichandran, K., Tan, S.H. & Bretschneider, T. 2004. High-precision on-board orbit determination for small satellites – The GPS-based XNS on X-Sat. *Sixth Symposium on Small satellites Systems and Services*, September 20 – 24. La Rochelle, France.
- Gleason, S. & Gebre-Egziabher, D. 2009. *GNSS: applications and methods*. Norwood, Massachusetts: Artech House.
- Glennon, É., Parkinson, K., Mumford, P., Shivaramaiah, N., Li, Y., Li, R., & Jiao, Y. 2011. A GPS receiver designed for cubesat operations. *2011 Australian Space Science Conference*, September 26 – 29. Canberra, Australia.

- Gold, K. & Brown, A. 2004. Architecture and Performance Testing of a Software GPS Receiver for Space-based Applications. *Proceedings of ION National Technical Meeting*: 624 – 635, January 26 – 28. San Diego, California, USA.
- GPS World. 2012. Receiver survey. <http://gpsworld.com/resources/gps-world-receiver-survey/> [Date accessed: 25 April 2014]
- GPS.Caltech. 2009. Global Positioning System: what it is and how we use it for measuring the Earth's movement. http://web.gps.caltech.edu/classes/ge167/file/GPS_Lecture_1.pdf [Date accessed: 22 June 2012]
- Greenberg, A., 2005. Open-source software for commercial off-the-shelf GPS receivers. Master's thesis. Portland State University, Oregon, USA.
- Grewal, M.S., Weill, L.R. & Andrews, A.P. 2001. *Global positioning systems, inertial navigation, and Integration*. New York: Wiley
- Group 09gr944. 2009. An investigation for FPGA implementation. Applied Signal Processing and Implementation. <http://vbn.aau.dk/files/19026857/09gr944.pdf> [Date accessed: 7 January 2014]
- Gunawardena, S. & Graas, F.V. 2006. Split-sum correlator simplifies range computations in GPS receiver. *Electronics Letters*, 42 (25): 1469 – 1471, December 7.
- Gustafson, D.E., Dowdle, J.R., Elwell, J.M. & Flueckiger, K.W. 2009. A nonlinear code tracking filter for GPS-based navigation. *IEEE Journal of Selected Topics in Signals Processing*, 3(4): 627 – 638, August.
- Hamsa, G.G., Zekry, A.A. & Moustafa, M.N. 2009. Implementation of a complete GPS receiver on the C6713 DSP through Simulink: *2009 Journal of Global Positioning Systems*, 8(1): 76 – 86.
- Hegarty, C. J. 2012. GNSS signals – An overview. *IEEE International Frequency Control Symposium (FCS)*, May 21 – 24, Baltimore, Maryland. USA.
- Hsu, L.T., Sun, C.C. & Jan, S.S. 2008. Comparison of acquisition and tracking methods for software receiver. *2008 Proceedings of the 4th Asian Space Conference*, October, 2008. Taipei, Taiwan.
- Jianguo, M., Kejing, C., Bao, L. & Yinbing, Z. 2010. An extraction method for multipath signals of GPS based on correlation function. *Second International Conference on Computer Modeling and Simulation 2010, ICCMS'10*: 129 – 132, January 22 – 24. Sanya, Hainan, China.
- Jin, S. 2012. *Global navigation satellite systems – signal, theory and applications*. Rijeka, Croatia: InTech.
- Johanssen, F., Mollaei, R., Thor, J. & Uusitalo, J. 1998. GPS Satellite Signal Acquisition and Tracking. <http://www.sm.luth.se/csee/courses/sms/019/1998/navstar/navstar.pdf> [Date accessed: 23 September 2013]
- Kahr, E., Skone, S. & O'Keefe, K. 2010. Orbit determination for the Canx-2 nano-satellite using intermittent GPS data. *ION GNSS 2010, Session D4b*, September 21 – 24. Portland, Oregon, USA.
- Kalyanaraman, S.K., Kelly, J.M., Braasch, M.S. & Kacirek, J. 2004. Influence of GPS code tracking on carrier-phase multipath performance. *IEEE Aerospace Conference Proceedings*, 3, March 6 – 13.

- Kaplan, E. D. & Hegarty, C. J. (eds). 2006. *Understanding GPS, principles and applications*. Second Edition. Boston, Massachusetts: Artech House.
- Kelley, C., Barnes, J. & Cheng, J. n.d. Open-source GPS: Open-source software for learning about GPS. <http://www.gmat.unsw.edu.au/snap/publications/opensourcegps.pdf> [Accessed: 23 April 2013]
- Kemt. n.d.a. GPS tutorial 2: Signals and messages. http://www.kemt-old.fei.tuke.sk/predmety/KEMT559_SK/GPS/GPS_Tutorial_2.pdf [Date accessed: 28 June 2014]
- Kemt. n.d.b. GPS supplement: GPS satellites constellation. http://www.kemt-old.fei.tuke.sk/predmety/KEMT559_SK/GPS/Nov25GPSSupplement.pdf [Date accessed: 28 June 2014]
- Kondoh, Y., Ishijima, Y., Kawano, I., Iwata, T. & Suzuki, H. 2009. Development of a new generation space-borne GPS receiver. *2009 IEEE Aerospace Conference*: 1 – 8, March 7 – 14. Big Sky, Montana, USA.
- Koontz, S.L., Bevilacqua, R. & Swenson, C. 2014. CubeSat technology adaptation for in-situ characterisation of NEOs. https://icubesat.files.wordpress.com/2014/05/icubesat-org_2014_b-2-5-neos_koontz_201405281603.pdf [Date accessed: 30 June 2014]
- Kowoma.de. 2009. The GPS system: Transmitted GPS signals. <http://archive.today/eC7C> [Date accessed: 25 April 2014]
- Kumar, P. 2012. Generating, optimising and verifying HDL code with Matlab and Simulink. <http://www.matlabexpo.com/in/2013/proceedings/generating-optimizing-and-verifying-hdl-code-with-matlab-and-simulink.pdf> [Date accessed: 15 February 2014]
- Kumar, B.P. & Paidimarry, C.S. 2014. Development and analysis of C/A code generation of GPS receivers in FPGA and DSP. *2014 Recent Advances in Engineering and Computational Sciences (RAECS)*: 1 – 5, March 6 – 8. Chandigarh, India.
- Langley, R.B., Montenbruck, O., Markgraf, M. & Kim, D. 2004. Qualification of a commercial dual-frequency GPS receiver for the e-POP platform onboard the Canadian CASSIOPE spacecraft. *2nd ESA Workshop on Satellite Navigation User Equipment Technologies, NAVITEC2004*. December 8 – 10. Noordwijk, The Netherlands.
- Lashley, M. 2006. Kalman filter based tracking algorithms for software GPS receivers. Master's thesis. Auburn University, Alabama, USA. http://gavlab.auburn.edu/uploads/Lashley_thesis.pdf [Date accessed: 12 September 2012]
- Lashley, M. & Bevly, D.M. 2009. Vector delay/frequency lock loop implementation and analysis. *ION 2009 International Technical Meeting*, January 26 – 28. Anaheim, California, USA.
- Ledvina, B.M., Powell, S.P., Kintner, P.M. & Psiaki, M.L. 2003. A 12 channel real-time GPS L1 software receiver. *Proceedings of the 2003 National Technical Meeting of the Institute of Navigation*: 767 – 782, January 22 – 24. Anaheim, California, USA.

- Lee, R., n.d. Northern communication, GPS-based science nano-satellite constellation mission. [http://www.spacemic.net/abstracts/\[Finalist\]%20Northern%20Communication%20and%20GPS-based%20science%20Nanosatellite%20constellation%20mission.pdf](http://www.spacemic.net/abstracts/[Finalist]%20Northern%20Communication%20and%20GPS-based%20science%20Nanosatellite%20constellation%20mission.pdf) [Accessed 18 April 2012]
- Li, H., Gao, H., Pei, W. & Shi, S. 2013. Implementation of a software-based GPS receiver. *2013 IEEE 4th International Conference on Electronics Information and Emergency Communication (ICEIEC)*: 140 – 143, November 15 – 17. Beijing, China.
- Li, J., Xu, T., Zhuang, Z. & Guan, Y. 2008. The analysis of a novel adaptive PN-code acquisition for GPS DSSS receivers. *Proceedings of the 2008 IEEE International Conference on Information and Automation*, June 20 – 23. Zhangjiajie, China.
- Lin, W. H. Mao, W. L. Tsao, H. W. Chang, F. R. & Huang, W. H. 2006. Acquisition of GPS software receiver using split-radix FFT. *2006 IEEE International Conference on Systems, Man and Cybernetics*, October 8 – 11. Taipei, Taiwan.
- Lofgren, J.U. n.d. (Spaceborne) Global Positioning Systems. http://munin.irf.se/frames/tecnology_main_gps2.html [Date accessed: 25 April 2014]
- Lohan, S. n.d. Lecture 12: TLT 5606. Spread spectrum techniques: Satellite-based positioning (III). http://www.cs.tut.fi/kurssit/TLT-5606/Lec12_TLT5606_S.pdf [Date accessed: 23 October 2012]
- Lu, D. 2003. Development and testing of a space-borne GPS signal strength Sensor. Master's thesis, Virginia Polytechnic Institute and State University. <http://scholar.lib.vt.edu/theses/available/etd-10082003-121827/unrestricted/ThesisReport.pdf> [Date accessed: 2 May 2012].
- Lu, X., Zhang, Y. & Zheng, Z. 2013. Carrier tracking loop improvement in a new indoor positioning system. *ICCIS'13 2013 International Conference on Computational and Information Sciences*: 1388 – 1391, June 21. Washington DC, USA.
- Ma, C., Lachapelle, G. & Cannon, M.E. 2004. Implementation of a software GPS receiver. *Proceedings of ION GNSS 2004 (Session A3)*, September 21 – 24. Long Beach, California, USA.
- Maini, A. K. & Agrawal, V. 2011. *Satellite Technology: Principle and Applications*. Second Edition. West Sussex, UK: WILEY
- Manandhar, D., Shibasaki, R. & Torimoto, H. 2006. GPS reflected signal analysis using software receiver. *Journal of Global Positioning Systems*, 5(1 - 2): 29 – 34. Tokyo, Japan.
- Mao, W. L., Chen, A. B., Tseng, Y. F., Chang, F. R., Tsao, H. W. & Huang, W. S. 2006. Design of peak-finding algorithm on acquisition of weak GPS signals. *IEEE International Conference on Systems, Man and Cybernetics*, October 8 – 11. Taipei, Taiwan.
- Markgraf, M. & Montenbruck, O. n.d.. Phoenix-HD – A miniature GPS tracking system for scientific and commercial rocket launches. http://www.dlr.de/rb/Portaldata/38/Resources/dokumente/gsoc_dokumente/rt/ISLT_05.pdf [Date accessed: 16 November 2013]
- MathWorks. 2005. Implementing a GPS receiver with DSP and FPGA hardware using Simulink and related tools. <https://www.mathworks.com/company/events/webinars/wbnr30463.html> [Date accessed: 16 August 2012].

- MathWorks. 2014. Working with generated fixed-point files.
<http://www.mathworks.com/help/hdlcoder/examples/working-with-generated-fixed-point-files.html> [Date accessed: 1 February 2014].
- MathWorks. n.d. EDA simulator link 3.3. Verify HDL and Verilog using HDL simulators and FPGAs.
<http://www.mathworks.com/products/datasheets/pdf/eda-simulator-link.pdf>
 [Date accessed: 7 August 2013]
- McNamara, J. 2004. *GPS for dummies*, New Jersey: Wiley.
- Meng, F., Zhu, B. & Wang, S. 2013. A new fast satellite selection algorithm for BDS-GPS receivers. *2013 IEEE Workshop on Signal Processing Systems (SiPS)*: 371 – 376, October 16 – 18. Taipei City, Taiwan.
- Miao, J., Sun, Y., Liu, J. & Chen, W. 2009. A Kalman filter based tracking loop in weak GPS signal processing. *Sixth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD'09*: 438 – 442, August 14 – 16. Tianjin, China.
- Misra, P. & Enge, P. 2006. *Global positioning system: signals, measurements and performance*. Second Edition. Lincoln, Massachusetts: Ganga-Jamuna Press.
- Mitel Semiconductor. 1998. GP2000: GPS Chipset – Designer’s Guide.
<http://www.datasheetarchive.com/dlmain/Datasheets-41/DSA-803003.pdf>
 [Date accessed: 23 November 2012]
- Montana, J. 2003. Global Positioning System. IT 488. George Mason University.
<http://www.softwaresearch.net/fileadmin/src/docs/teaching/SS04/OS/GPS.ppt>
 [Date accessed: 25 August 2012]
- Montenbruck, O. 2008. GNSS Receivers for Space Applications. ACES and Future GNSS-Based Earth Observation and Navigation.
http://www.iapg.bv.tum.de/mediadb/14973/14974/04_ACES_WS_08_SGNSS.pdf
 [Date accessed: 5 June 2013]
- Montenbruck, O., Garcia-Fernandez, M., & Williams, J., 2006. Performance comparison of semi-codeless GPS receivers for LEO satellites. *GPS Solutions-Springer*, 10(4): 249 – 261, March 22.
- Montenbruck, O. & Holt, G. 2002. Spaceborne GPS receiver performance testing. Space Flight Technology, German Space Operations Center (GSOC).
http://www.dlr.de/rb/en/Portaldata/38/Resources/dokumente/gsoc_dokumente/rt/TN_0204.pdf
 [Accessed: 19 July 2012].
- Montenbruck, O., Markgraf, M. Leung, S. & Gill, E. 2001. A GPS receiver for space applications. *ION-GPS-2001*, September 12 – 14. Salt Lake City, Utah, USA.
- Montenbruck, O., Nortier, B. & Mostert, S. 2004. A miniature GPS receiver for precise orbit determination of the Sunsat 2004 micro-satellite. *ION NTM (National Technical Meeting)*, January 26 – 28. San Diego, California. USA.
- Munteanu, A.C. 2009. Nanosat /Cubesat constellation concept. Master thesis, Cranfield University, UK.
<http://epubl.ltu.se/1653-0187/2009/092/LTU-PB-EX-09092-SE.pdf> [Accessed: 18 April 2012]

- NASA. n.d. Cubesat facts sheet.
http://www.nasa.gov/pdf/627973main_513455main_CubeSatFactsheet_Final.pdf
[Date accessed: 19 April 2012]
- NASA. 2009. Cloud cubesat. <http://nasa.olin.edu/projects/2009/cloudsat/cubesat.htm> [Date accessed: 19 April 2012]
- NASA. 2014. CubeSats initiative. http://www.nasa.gov/directorates/heo/home/CubeSats_initiative.html
[Date accessed: 27 June 2014]
- NASA Goddard. 2013. Tech-transfer news: CubeSat development at NASA Goddard.
<http://techtransfer.gsfc.nasa.gov/newsletter/TechTransferNews-Spring13.pdf>
[Accessed: 28 June 2014]
- Navstar IS-GPS-200H. 2013. Navstar GPS space segment / navigation user segment: interfaces specification (IS-GPS-200H). http://www.navcen.uscg.gov/pdf/gps/IS_GPS_200H.pdf
[Date accessed: 19 June 2014]
- Nelson, J. M. 2010. Persistent military satellite communications coverage using a cubesat constellation in low Earth orbit. Master's thesis, University of Central Florida, Orlando, USA.
http://www.agi.com/downloads/partners/edu/Nelson_Thesis.pdf [Date accessed: 18 April 2012]
- Nortier, B. J. 2003. A space-borne GPS receiver. Master's thesis, University of Stellenbosch, Cape Town.
- O'Mahony, N. & Cruz, F. P. 2012. A novel sequential Bayesian approach to GPS acquisition. *2012 Third International Workshop on Cognitive Information Processing (CIP)*: 1 – 6, May 28 – 30. Baiona, Pontevedra, Spain.
- Olivieri, S., Bagni, D. & Visintini, M. 2011. Implementing Matlab and Simulink algorithms on FPGAs.
http://www.t3lab.it/wp-content/uploads/download/20110915/MathWorks_FPGA%20Seminar%20with%20Xilinx.pdf
[Date accessed: 29 November 2013]
- Paluszek, M., Castro, E. & Hyland, D. 2010. The cubesat book.
<http://www.psatellite.com/CubeSat/documents/CubeSatBook.pdf> [Accessed: 18 April 2012]
- Parkinson, K.J., Dempster, A.G., Mumford, P. & Rizos, C. 2006. FPGA based GPS receiver design consideration: *Journal of Global Positioning Systems (2006)*, 5(1 – 2): 70 – 75. Australia.
- Peng, S., Morton, Y.J. & Di, R. 2012. A multiple-frequency GPS software receiver design based on a vector tracking loop. *2012 IEEE/ION Position Location and Navigation Symposium (PLANS)*: 495 – 505, April 23 – 26. Myrtle Beach, South Carolina, USA.
- Plausinaitis, D. 2008. GNSS receiver front-ends II: components: GPS receiver technology MM8. Danish GPS Center. http://kom.aau.dk/~dpl/courses/mm08_slides.pdf [Date accessed: 7 October 2012].
- Plausinaitis, D. 2009a. Carrier tracking Loop; loop filter: GPS signals and receiver technology MM12. Danish GPS Center. http://kom.aau.dk/~dpl/courses/mm12_slides.pdf
[Date accessed: 7 October 2012].
- Plausinaitis, D. 2009b. Code tracking; multipath: GPS signals and receiver technology MM13. Danish GPS Center. http://kom.aau.dk/~dpl/courses/mm13_slides.pdf [Date accessed: 7 October 2012].

- Plausinaitis, D. 2009c. Receiver overview; code generation and carrier generation: GPS signals and receiver technology MM10. Danish GPS Center. http://kom.aau.dk/~dpl/courses/mm10_slides.pdf [Date accessed: 7 October 2012].
- PM Solutions Technology. 2003. White paper series: selecting a software development life cycle methodology. www.pmsolutions.com [Date accessed: 25 March 2013].
- Qi, N., Xu, Y., Chi, B., Xu, Y., Yu, X., Zhang, X., Xu, N., Chiang, P., Rhee, W. & Wang, Z. 2012. A dual channel Compass/GPS/GLONASS/Galileo reconfigurable GNSS receiver in 65nm CMOS with on-Chip I/Q calibration. *IEEE Transactions on Circuits and Systems—I: Regular Papers*, 59(8), 1720 – 1732. August.
- Ramakrishnan, S., Gao, G.X., Lorenzo, D.D., Walter, T., Enge, P. & Akos, D. 2008. Design and analysis of reconfigurable embedded GNSS receivers using model-based design tools. *Proceedings of the 21st International Technical Meeting of the Satellite Division (ION GNSS 2008)*: 2293 – 2303, September 16 – 19. Savannah, Georgia, USA.
- Rao, M.V.G., & Ratnam, V.G. 2013. Faster GPS / IRNSS acquisition via sub sampled FFT and thresholding. *2013 Annual IEEE Indian Conference*: 1 – 4, December 13 – 15. Mumbai, India.
- Rao, S.B., 2009. Design and implementation of a GPS receiver channel and multipath delay estimation using Teager-Kaiser operator. Master's thesis, Indian Institute of Science, Bangalore, India. http://www.serc.iisc.ernet.in/graduation-theses/babu_09.pdf [Date accessed: 17 July 2012]
- Renaudie, C., Markgraf, M., Montenbruck, O. & Garcia, M. 2007. Radiation testing of commercial-off-the shelf GPS technology for use on low Earth orbit satellites. *9th European Conference on Radiation and its Effects on Components and Systems, 2007. RADECS 2007*: 1 – 8, September 10 – 14. Deauville, France.
- Rogers, R.M. 2007. *Applied mathematics in integrated navigation Systems*, Third Edition. Virginia: American Institute of Aeronautics and Astronautics (AIAA) Education Series.
- Sagiraju, P. K., Agaian, S. & Akopian, D. 2006. Reduced complexity acquisition of GPS signals for software embedded applications. *IEE (IET) Proceedings - Radar, Sonar and Navigation*, 153(1): 69 – 78, February.
- Samper, J.M., Pérez, R.B. & Lagunilla, J.M. 2009. *GPS & Galileo: dual RF front-end receiver and design, fabrication and test*. New York: McGraw-Hill Companies.
- Sandau, R., Roeser, H. & Valenzuela, A. (eds). 2010. *Small satellite missions for Earth observation*. Berlin: Springer Berlin Heidelberg.
- Shi, L., Chuanrun, Z., Xingqun, Z. & Boxiong, W. 2009. Implement of real-time GPS LI software receiver. *Fourth International Conference on Computer Science & Education, 2009 (ICCSE'09)*, 1132 – 1137, July 25 – 28. Nanning, China.
- Smalarz, B. R. 2011. Cubesat constellation analysis for data relaying. Master's thesis, California Polytechnic State University, San Luis Obispo, USA. <http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1691&context=theses> [Date accessed: 18 April 2012]

- Song, G., Wang, C., Xi, F. & Zhang, A. 2012. The performance of tracking loop filter for software GPS receiver simulation analysis. *Proceedings of the 2012 International Conference on Computer Science and Electronics Engineering (ICCSEE'12)*, (3): 703 – 706. Washington DC, USA.
- Spirent Communications. 2014. <http://www.spirent.com/Products/GSS6700> [Accessed: 20 April 2014]
- Strang, G. & Borre, K. 1997. *Linear algebra, geodesy, and GPS*. Massachusetts: Wellesley-Cambridge Press.
- Sweeting, M. 2010. *Microsatellite; moving from research to constellation; meeting real operational missions*. http://park.its.u-tokyo.ac.jp/nsat/NS1/files/11th.PM/Session_5/Presentation_Martin-Sweeting.pdf [Date accessed: 16 April 2012]
- Ta, T.H., Shivaramaiah, N.C., Dempster, A.G. & Presti, L.L. 2012. Significance of cell-correlation phenomenon in GNSS matched filter acquisition engines. *IEEE Transactions on Aerospace and Electronic Systems*, 48(2): 1264 – 1286, April.
- Tang, B.Z., Longfield, S., Bhave, A.S. & Manohar, R. 2012. A low power asynchronous GPS baseband processor. *2012 18th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*: 33 – 40, May 7 – 9. Lyngby, Denmark.
- Tian, J. & Yang, L. 2008. A novel GNSS weak signal acquisition using wavelet de-noising method. *Proceedings of the 2008 National Technical Meeting of the Institute of Navigation*: 303 – 309, January 28 – 30. San Diego, California, USA.
- Tsai, Y., Chang, C., Juang, J. & Lin, C. 2010. Implementation and test of a space-borne GPS receiver payload of micro-satellite. *2010 International Symposium on GPS/GNSS*, October 26 – 28. Taipei, Taiwan.
- Tsui, J.B. 2005. *Fundamentals of global positioning system receivers: A software approach*. Second Edition. New Jersey: WILEY.
- Underwood, C., Unwin, M., Sorensen, R.H., Frydland, A. & Jameson, P. 2004. Radiation testing campaign for a new miniaturised space GPS receiver. *2004 IEEE Radiation Effects Data Workshop*: 120 – 124, July 22. Atlanta, USA.
- Unwin, M.J. & Oldfield, M.K. 2000. The design and operation of a COTS space GPS receiver. <http://www.researchgate.net/publication/228854740> The design and operation of a COTS space GPS receiver. [Date accessed: 24 July 2013]
- Willms, B. 1999. Space integrated GPS/Ins (SIGI) navigation system for space shuttle. *Proceedings of the 18th Digital Avionics Systems Conference, 1999*, 2: 7.A.4-1 – 7.A.4-8, October 24 – 29, St Louis, Missouri, USA.
- Winternitz, L., Moreau, M., Gregory, J., Boegner, G.J. & Sirotzky, S. 2004. Navigator GPS receiver for fast acquisition and weak signal in space applications. *Proceedings of the 17th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2004)*: 1013 – 1026, September 21 – 24. Long Beach, California, USA.

- Xiaowen, S., Qing H., Shufang, Z., Jingbo, Z. & Vi, J. 2010. Fast acquisition of GPS signal using extended multiple correlator based on FPGA. *2010 International Conference on Computer Application and System Modeling (ICCASM)*, 5: 208 – 211, October 22 – 24. Taiyuan, China.
- Xiaoyong, M. & Baiqi, L. 2011. Design of an INS aided high dynamic GPS receiver. *2011 International Conference on Electronics, Communications and Control (ICECC)*: 1404 – 1407, September 9 – 11. Ningbo, China.
- Xiong, Y., Huang, D., Ding, X. & Chen, Y. 2000. Wavelet transform based Kalman filtering algorithm for anti-selective availability effect. *Geographic Information Sciences*, 6(2): 165 – 169, December. China.
- Xu, G. 2007. *GPS Theory, Algorithms and Applications*. Second Edition. Heidelberg, Berlin: Springer.
- Yang-bo, H., Xiao-mei, T. & Gang, O. 2009. Differential correlator for tracking GPS signals. *2009 International Conference on Information Engineering and Computer Science (ICIECS 2009)*: 1– 41, December 9 – 20. Wuhan, China.
- Yao, X., Qin, X. & Fang, J. 2009. A novel fast acquisition algorithm for GPS C/A signals. *2009 17th International Conference on Geo-informatics*: 1– 6, August 12 – 14. Fairfax, Virginia, USA.
- Yunck, T. P. & Melbourne, W.G. 1995. Spaceborne GPS for Earth science. *International Union of Geodesy and Geophysics (IUGG) Meeting, Symposium 15*: 113 – 122, July 3 – 4. Boulder, Colorado, USA.
- Zarlink Semiconductor. n.d. GP2010 / GP2015. GPS receiver RF front-ends. <http://ulp.zarlink.com/zarlink/gp2015-datasheet-sept2007.pdf> [Date accessed: 13 October 2012].
- Zarlink Semiconductor. n.d. GP2021/GP4020. http://www.zarlink.com/zarlink/hs/82_GP4020.htm [Accessed: 14 October 2012].
- Zhang, H. & Unwin, M. 1999. The use of GPS receiver on Tsinghua-1 microsatellite. http://www.researchgate.net/publication/228742963_The_Use_of_GPS_Receiver_on_Tsinghua-1_Microsatellite [Date accessed: 27 June 2014].
- Zhe, H., Chuanrun, Z. & Xingqun, Z. 2008. Signal quantization effects on acquisition process of GPS receiver: the analysis and simulation. *Congress on Image and Signal Processing, 2008 (CISP'08)*, 1: 197 – 200, May 27 – 30. Sanya, Hainan, China.
- Zheng, S.Y. 2005. Signal acquisition and tracking for a software GPS receiver. Master's thesis, Blacksburg, Virginia, USA. http://scholar.lib.vt.edu/theses/available/etd-02202005-223653/unrestricted/zheng_thesis2.pdf [Date accessed: 15 September 2013]
- Zheng, Y. 2008. Adaptive antenna array processing for GPS receivers. Master's thesis, Adelaide, Australia. <https://ebooks.adelaide.edu.au/dspace/bitstream/2440/49670/2/01front.pdf>. [Date accessed: 5 October 2013]