TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

**MIKA TAKALA**

# HIGH-LEVEL SYNTHESIS FOR TRAVEL-TIME TOMOGRAPHY

Master of Science thesis

Examiners: Prof. Timo D. Hämäläinen,
Assistant Prof. Sampsa Pursiainen
Examiners and topic approved by the
Faculty Council of the Faculty of
Computing and Electrical Engineering
on 2nd August 2015.

# ABSTRACT

**MIKA TAKALA**: High-level synthesis for travel-time tomography
Tampere University of Technology
Master of Science thesis, 66 pages
June 2016
Master's Degree Programme in Signal Processing and Communications Engineering
Major: Embedded systems
Examiners: Prof. Timo D. Hämäläinen & Asst. Prof. Sampsa Pursiainen
Keywords: Travel-time tomography, high-level synthesis, Catapult C, FPGA, Matlab

Tomography is used to acquire an image of inner contents of an object or material. Typically measurements are based on penetrating waves. Measurements require signal generating and recording hardware, data preprocessing and finally computationally heavy mathematical inversion to recover the unknown parameters. The goal is to decrease the data transfer requirement on, for example, tomography mission to a Near-Earth Asteroid. This could enable small, inexpensive spacecraft to collect tomography data. Travel-time tomography, which uses signal travel times, is one of the methods that can be used to achieve this goal. Tomography algorithms are still under heavy development, for which reason hardware prototyping cycle should be very short. High-level synthesis is used to generate hardware by using high level programming language. It helps the designer to implement hardware design changes quickly, especially when the requirements change.

In this work, a setup to collect acoustic tomography data was developed. Data preprocessing hardware for travel-time tomography was implemented with Mentor Graphics Catapult high-level synthesis tool. The calculation of travel-time values was implemented first in Matlab and the scripts were then transformed to C code. Catapult was used to implement hardware on the FPGA from these C codes. Evaluation of the workflow was performed and interfacing options for the module to a PC running Matlab were studied.

Travel-time tomography was shown to be a feasible method to recover target objects. Determinining the time period to use in measuring a travel-time is an issue. Simulation of signal noise sensitivity on an asteroid mission was accomplished by reducing the accuracy of preprocessor calculations. A method where signal power is integrated over a time period was evaluated and it proved to be surprisingly stable in recovering targets from the test area even with noisy signals. Tomography algorithms changed over the course of the project, and high-level synthesis enabled to implement the designs.

# TIIVISTELMÄ

**MIKA TAKALA**: Korkean tason synteesin soveltaminen matka-aikatomografiaan
Tampereen teknillinen yliopisto
Diplomityö, 66 sivua
Kesäkuu 2016
Signaalinkäsittelyn ja tietoliikenteen koulutusohjelma
Pääaine: Sulautetut järjestelmät
Tarkastajat: Professori Timo D. Hämäläinen ja apulaisprofessori Sampsa Pursiainen
Avainsanat: Matka-aikatomografia, korkean tason synteesi, Catapult C, FPGA, Matlab

Tomografiaa käytetään materiaalin tai esineen sisäisen rakenteen tutkimiseen. Yleensä mittaukset perustuvat materiaalin läpäiseviin aaltoihin. Mittauksiin tarvitaan signaalin generoivaa ja nauhoittavaa laitteistoa, datan esikäsittelyä ja lopuksi laskennallisesti raskasta matemaattista inversiota, jotta tuntemattomat parametrit saadaan selville. Tavoitteena on vähentää siirrettävän datan määrää esimerkiksi Maan lähiasteroidille suuntautuvalla avaruuslennolla. Se mahdollistaisi pienten ja halpojen avaruusalusten kerätä tomografiadataa. Matka-aikatomografia, joka käyttää signaalin kulkuaikoja, on yksi keino millä tämä tavoite voidaan saavuttaa. Tomografia-algoritmeja kehitetään aktiivisesti, jonka vuoksi laitteistoprototyyppisyklin tulisi olla lyhytkestoinen. Korkean tason synteesiä käytetään laitteiston generoimiseen korkeamman tason ohjelmointikielellä. Se auttaa suunnittelijaa laitteiston muutosten toteuttamisessa nopeasti, kun vaatimukset muuttuvat.

Tässä työssä kokeellisen datan keräämiseksi rakennettiin testiympäristö. Matka-aikatomografiadatalle toteutettiin esikäsittelylaitteisto Mentor Graphicsin Catapult synteesityökalulla. Matka-aika-arvojen laskenta toteutettiin ensin Matlabissa ja skriptit muunnettiin C-koodeiksi, jonka pohjalta Catapulttia käytettiin laitteiston generoimiseen FPGA-piirille. Työvuota arvioitiin ja laskentamoduulien kytkemistä PC:llä toimivaan Matlab-ohjelmaan tutkittiin.

Matka-aikatomografia osoittautui mahdollistavan esineiden löytämisen testiympäristöstä. Matka-ajan laskemisessa tarvittavan ajanjakson määrittämiseen liittyy avoimia kysymyksiä. Signaalin kohinaherkkyyttä simuloitiin pienentämällä matka-aikojen laskentatarkkuutta. Sellaista mittaustapaa testattiin, jossa signaalin tehoa integroidaan tietyn ajanjakson ajan. Se näytti löytävän esineet yllättävän hyvin myös kohinaisella datalla. Tomografia-algoritmit muuttuivat projektin aikana ja korkean tason synteesi mahdollisti muutosten käytännön toteuttamisen.

# PREFACE

After financial realities prevented me from designing the Aalto-1 cubesat's computer system as thesis project in 2012 my studies were on hold for a while. In November 2014, Tampere University of Technology organized a TUT Forum event at Tampere Hall. There Professor Sampsa Pursiainen met a team of students from Castor, the space club of Tampere University of technology. That meeting resulted in Prof. Pursiainen getting in touch with me in regards to a possibility of a Master's thesis work in the framework of mathematics and asteroid mining. Our first meeting brought up fear of being overwhelmed by the mathematics involved, but I didn't let the interesting subject run away this time.

I took a giant leap into the unknown regarding the mathematical side of things, having studied only the mandatory engineering mathematics courses years before. I thank Prof. Pursiainen very much for his support and for the help to understand the underlying math over countless amount of meetings during his workdays for the duration of the project. He also had vision in helping me to formulate the research problem to suit his current research needs and my major in embedded systems engineering. I thank Prof. Timo D. Hämäläinen, who guided me towards trying high-level synthesis in the work so that the rather out of this world application could be accepted as my thesis. The idea of exactly how to connect these fields of science and engineering was decided some time later.

I want to give very warm thanks to the members of Castor (Juha Koljonen, Ilari Graf, Cliona Shakespeare, Esa Niemi and Niki Suominen) for bringing the very heavy vacuum chamber for display at the TUT forum event that day in 2014. I wouldn't have been contacted regarding this thesis work possibility if it wasn't for these wonderful people. I also thank my friends, former roommates, countless amount of exercise work partners and of course my parents and family for extraordinary support during my years of studying at TUT.

Tampere, 17.5.2016

Mika Takala

# TABLE OF CONTENTS

1. Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

2. Applications and constraits of on-site waveform tomography . . . . . . . . . . 5

2.1 Subsurface imaging of small planetary objects (SPOs) . . . . . . . . . . . 5

2.2 Waveform tomotraphy in civil engineering . . . . . . . . . . . . . . . . . . 8

2.3 Modern biomedical engineering applications . . . . . . . . . . . . . . . . 8

3. Mathematical methodology . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

3.1 Forward modeling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

3.2 Forward simulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

3.3 Inversion approach . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

3.4 Path integrals and ray tracing . . . . . . . . . . . . . . . . . . . . . . . 16

3.5 Travel-time calculation . . . . . . . . . . . . . . . . . . . . . . . . . . 16

3.6 Signal noise . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

4. Experimental data gathering hardware . . . . . . . . . . . . . . . . . . . . . 20

4.1 Test setup and scenario . . . . . . . . . . . . . . . . . . . . . . . . . . 20

4.2 Analysis of the test setup's error sources . . . . . . . . . . . . . . . . . . 23

4.3 General basis for the hardware to simulation interface . . . . . . . . . . . 27

5. Hardware design, implementation, optimization and benchmarks . . . . . . . . 29

5.1 Design aspects . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29

5.2 Overall design concept . . . . . . . . . . . . . . . . . . . . . . . . . . 30

5.3 Interface protocol and hardware . . . . . . . . . . . . . . . . . . . . . . 33

5.3.1 Protocol . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

5.3.2 Matlab code . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36

5.3.3 Hardware interface on the FPGA . . . . . . . . . . . . . . . . . . 37

5.4 Computation unit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38

5.4.1 Matlab to C workflow options . . . . . . . . . . . . . . . . . . . . 38

5.4.2 Catapult C workflow and code . . . . . . . . . . . . . . . . . . . . 39
5.4.3 Problems encountered . . . . . . . . . . . . . . . . . . . . . . . . 41
5.5 Control unit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
6. Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
6.1 Inversion results . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
6.2 High-level synthesis results . . . . . . . . . . . . . . . . . . . . . . 48
7. Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
7.1 Tomography . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
7.2 Hardware . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
8. Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
Bibliography . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59

# LIST OF FIGURES

1.1 Spacecraft concept drawing . . . . . . . . . . . . . . . . . . . . . . . . . 2

1.2 Sensor-to-software interface . . . . . . . . . . . . . . . . . . . . . . . . 3

2.1 Typical NEA orbits . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

2.2 Asteroid Belt . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

2.3 Magnetic resonance imaging and ultrasonic imaging hardware compared . 9

3.1 Orbiter-to-orbiter and lander-to-orbiter scenario . . . . . . . . . . . . . . 12

3.2 Visualization of measured and simulated travel-time calculation . . . . . 19

4.1 Block diagram of hardware used in test setup . . . . . . . . . . . . . . . 21

4.2 Test setup with a speaker and two microphones . . . . . . . . . . . . . . 21

4.3 Top down diagram of the test equipment . . . . . . . . . . . . . . . . . . 22

4.4 Transmitted and recorded pulse . . . . . . . . . . . . . . . . . . . . . . . 25

4.5 Matlab script flow chart . . . . . . . . . . . . . . . . . . . . . . . . . . 27

5.1 Embedded hardware project overview . . . . . . . . . . . . . . . . . . . 31

6.1 Integrating/integrating travel time results . . . . . . . . . . . . . . . . . . 45

6.2 Thresholded/integrating travel time results . . . . . . . . . . . . . . . . . 46

6.3 Thresholded/thresholded travel time results . . . . . . . . . . . . . . . . 47

7.1 Asteroid with a surface layer . . . . . . . . . . . . . . . . . . . . . . . . 52

7.2 Future development for data gathering hardware . . . . . . . . . . . . . . . 55

# LIST OF TABLES

1.1 Summary of the goals of the thesis project . . . . . . . . . . . . . . . . . . 3

4.1 Hardware used in the test . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

4.2 Size and center coordinates on the yoga mat . . . . . . . . . . . . . . . . . 23

5.1 Data transmission protocol . . . . . . . . . . . . . . . . . . . . . . . . . . 34

5.2 Parameters and their defaults. . . . . . . . . . . . . . . . . . . . . . . . . 35

6.1 Synthesis results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49

7.1 Acoustic tomography compared to others . . . . . . . . . . . . . . . . . . 51

8.1 Summary of the results . . . . . . . . . . . . . . . . . . . . . . . . . . . 57

# LIST OF ABBREVIATIONS AND SYMBOLS

| | |
|---|---|
| AU | Astronomical Unit, the average distance from the Earth to the Sun |
| CONSERT | Comet Nucleus Sounding Experiment, a radio tomography instrument on ESA's Rosetta mission |
| DSI | Deep Space Industries, a commercial company from the United States aiming to mine asteroids |
| FDTD | Finite-Difference Time Domain, a simulation method for wave equations |
| FPGA | Field Programmable Gate Array, a type of programmable integrated circuit |
| ESA | European Space Agency |
| HLS | High-Level Synthesis, a method of producing synthesizable digital integrated circuit hardware descriptions by using higher-level programming language such as C |
| LEO | Low-Earth Orbit, altitudes from 150 to 1000 km for example. |
| NASA | National Aeronautics and Space Administration, the space agency of the United States of America |
| NEA | Near-Earth Asteroid |
| Rosetta | ESA's ongoing mission to study comet 67P |
| TUT | Tampere University of Technology |

| | |
|---|---|
| $u$ | electric potential field |
| $f$ | frequency |
| $\lambda$ | wave length |
| $\varepsilon_r$ | relative permittivity |
| $\sigma$ | conductivity |
| $\Omega$ | (bounded) domain such as a test area |

# 1. INTRODUCTION

Tomography is a non-destructive method of acquiring information about the unknown inner contents of an object or material. Usually tomography is achieved by dividing the test subject into sections. Then a signal is passed through the test subject and the signals that have passed through are received in another location and recorded. The received signals must be further processed and combined with data from other sections to reconstruct the unknown parameters.

This work concentrates on waveform tomography in which either a transverse electromagnetic or longitudinal acoustic wave travels through a target domain and the task is to recover some parameter distribution(s) within the domain. For example, velocity of the wave in different parts of the domain or the absorbtion parameter can be useful if recovered. Typical tomography systems use acoustic, ultrasonic or electromagnetic waves. Other signal modalities include, e.g., direct or alternating current measurement that can be used to recover impedance distributions. Tomographic imaging based on electromagnetic or acoustic wave propagation requires computationally heavy mathematical inversion of the data to retrieve the relevant result set, such as an image or a three-dimensional model of the test subject. Tomography in general has wide range of applications, including medical imaging and process quality control.

The current focus of the research, such as [66, 67] is, in particular, to develop tomography methods suitable for cases in which the signals are sparse. One example of such case is tomography of asteroids where only very limited number of orbiting probes or landers can be used. This extreme case is presented in this work as the potential final target application of the research. High cost of space missions is the ultimate limiting factor. An example of such a project is the COmet Nucleus Sounding Experiment by Radiowave Transmission (CONSERT) experiment on the European Space Agency's Rosetta mission, which uses one comet orbiting spacecraft and a single comet lander [42]. The total cost of the mission is 1.4 billion Euros [26]. The signal modality used in the CONSERT experiment results in an incomplete coverage of the recorded signals, i.e,. signal sparsity. CONSERT

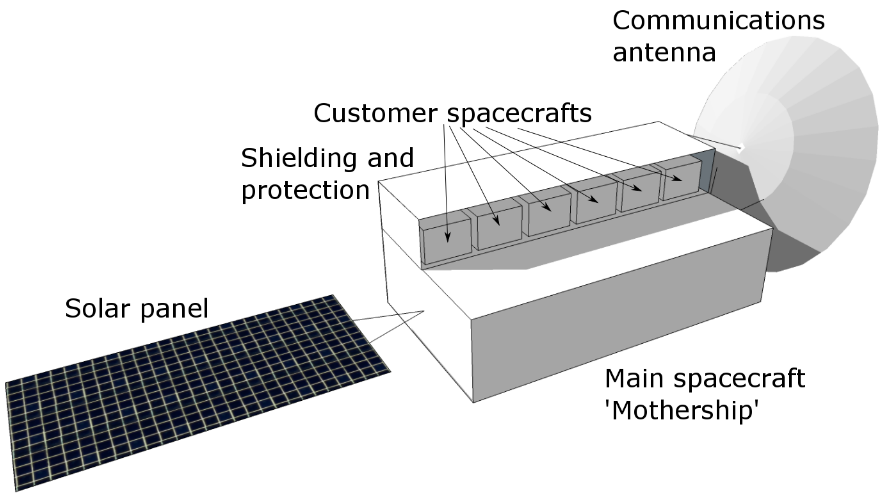


***Figure 1.1*** *Spacecraft concept with undeployed customer spacecrafts. Adapted from [23].*

utilizes a lander-to-orbiter measurement scenario where the signal is transmitted by the Philae lander and recorded on the Rosetta spacecraft. There are also other than financial limitations in such missions. Namely, the orbiters or landers are embedded computer systems operating in challenging space environment [1, 30]. They might be short-lived, their memory capacity is limited, and the amount of data that can be transmitted to scientists on Earth is also limited. The short less than 3-day lifetime of the Philae lander on the Rosetta mission is a concrete example of such a device and durability [25].

Some commercial organizations, such as Deep Space Industries (DSI) [22] and Planetary Resources [62] are developing spacecraft technologies to enable future asteroid mining missions. DSI is selling passenger slots for small satellites on near-earth asteroid prospecting missions [23]. A concept similar to one shown by DSI is shown in figure 1.1. The artistic figure highlights the important services of protection and communication for the customer spacecrafts for the journey to the target asteroid, before the customer spacecrafts are released to begin their own missions at the target. The bandwidth to Earth is shared with all other passenger (client) spacecrafts via the main antenna.

This thesis work concentrates on the aspects of a sensor-to-software interface for waveform tomography. Sensor-to-software interface here means the preprocessing and data transfer from a physical sensor to a tomography algorithm implemented on computation software such as Matlab. The general scenario is illustrated in figure 1.2. There are multiple goals and results in this thesis, and all of them are summarized in table 1.1. The ultimate goal is to help coupling the existing mathematical methodology with real-life ap-

***Table 1.1*** *Summary of the goals of the thesis project*

| Goal | Short description |
|---|---|
| Evaluate high-level synthesis workflow | Implement a part of the tomography algorithm, namely the travel-time calculation scripts using Catapult C and evaluate the design process and the resulting hardware |
| Design a basis for sensor-to-software interface | Design a method for data transfer between physical sensors, implemented calculation modules and Matlab calculation software |
| Experimental data for travel-time tomography | Gather experimental data to test the calculation system and evaluate travel-time tomography method's performance in this application |
| Evaluate hardware implementation aspects | Evaluate bit depth and other hardware effects and compare the results with Matlab and with real missions if possible |

plications under on-site restrictions. Current tomography algorithms are mathematically heavy, but at least some parts of the processing can be performed on-site. In particular, travel-time tomography is considered as a potential solution to reduce the amount of data that needs to be transmitted between the hardware sensors (antennas, receivers, transmitters and microphones) and the computation software. This is essential in applications with a restricted data transfer capability, such as a planetary space mission. In this tomography scenario the data is preprocessed before being transmitted to Earth. One of the goals of

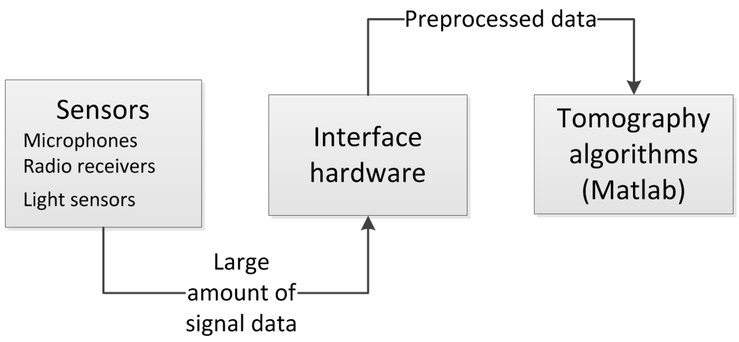


***Figure 1.2*** *Block diagram of data coming from sensors, processed by hardware and delivered to tomography algorithm for further processing.*

this thesis work is to enable these small, inexpensive spacecraft to collect tomography data and process it in a way that makes it easier to store and transmit to Earth.

Data processing performed on embedded hardware has its limitations, and an important scientific goal is to get a first look at how the current mathematical model and related data processing works under these limitations. Because of on-site limitations in this planetary space mission scenario the tomography algorithms should not be run on-site. In other scenarios, such as on portable ultrasound scanner device, all tomography algorithms could be implemented on the embedded device as the limitations are not as severe.

To perform these travel-time preprocessing operations a field programmable gate array (FPGA) chip on a development board is used as a demonstration platform for the design. Hardware on the FPGA is implemented in a novel way by transforming the original Matlab scripts to C code. Mentor Graphics Catapult hardware synthesis tool [57] is used to generate the hardware from the C code. The development of integrated circuits via this kind of high-level synthesis (HLS) method reduces development times and is potentially more suitable for non-hardware oriented engineers. The interfacing methods between the computation module and a PC running Matlab software are also considered. It has to be noted that Mentor Graphics Inc. acquired Calypto Design Systems Inc., the developer of Catapult software, during the course of the project [56], so the literature referenced in this work often refers to Calypto Catapult.

This work is organized as follows. The next chapter introduces the reader potential contexts of waveform tomography and the physical limitations and challenges. Some different types of tomography and their typical applications are explained. Fine details and mathematics are omitted for simplicity. This serves as an introduction to the following chapter. Chapter 3 explains the mathematical methodology used in modeling and propagating an acoustic or electromagnetic wave. The travel-time tomography application for the FPGA demonstration is also introduced. In chapter 4 the data gathering setup consisting of acoustic speaker and microphones is documented. The relevance of the acoustic setup is also explained. The requirements for the FPGA system is refined based on real data acquired from the experimental setup. Chapter 5 contains details of the design and implementation of the embedded system on the FPGA development board and its interfaces to running Matlab software on a PC. The selected Matlab to hardware workflow is detailed. Chapter 6 shows the overall results of the work and chapter 7 discusses the results, reflects on the relevance of work to other tomography applications and discusses possibilities for future development. Conclusions are given in chapter 8.

# 2. APPLICATIONS AND CONSTRAITS OF ON-SITE WAVEFORM TOMOGRAPHY

The mathematics and methods underlying waveform tomography in different applications are very similar to each other. Usually the on-site limitations and possibilities differ in addition to the unknown parameters that have different sizes and different characteristics. In this chapter, a few examples of tomography applications and current research are given with particular emphasis on the asteroid tomography scenario, which is used as one of the targets of the research done in this project. The three application fields presented in this chapter are only examples and they do not represent all the fields where tomography is used. From the Finnish point of view, significant research of tomography of the ionosphere has been done at the Finnish Meteorological Institute, such as [61, 81].

## 2.1 Subsurface imaging of small planetary objects (SPOs)

Typical near-earth asteroids (NEA:s) orbit around the sun in such way that their furthest point from the Sun is 1.3 astronomical units (AU), where 1 AU is the average distance of the Earth from the Sun, averaging 149 597 871 kilometres. Other orbital parameters of NEA:s divide them to different classes [14, 58]. In short, NEA:s orbit the Sun in similar distance from the Sun as the Earth, but in order to get there, the mission might be required to fly closer or further to the Sun from the initial launch point (Earth). Typical NEO asteroid orbits in respect to the Sun and Earth's orbit are illustrated in figure 2.1.

The spacecraft is protected from solar radiation only during launch and early part of the mission. In that timeframe, the satellite is flying inside the Earth's magnetic field, which protects the Earth and spacecraft from most of the solar particle radiation. Also during the same time, infrared radiation from Earth warms up the spacecraft. For the most part of the mission, Earth's infrared radiation won't be warming up the spacecraft and it is not protected from solar radiation by the Earth's magnetic field as it flies far away from the Earth [1]. This means that the requirements for such spaceraft's electronics and other functions

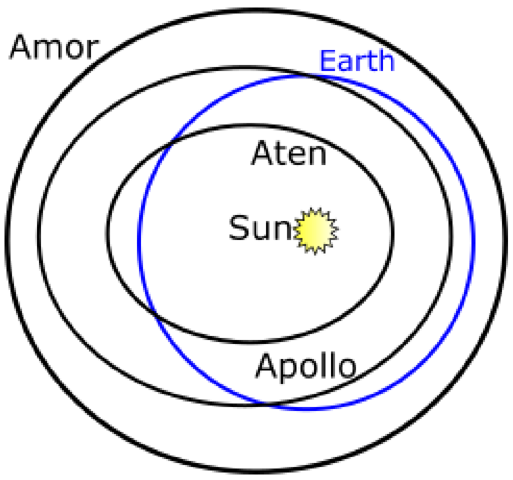


***Figure 2.1*** *Illustration of the orbits for the most common classes of Near Earth Asteroids in general relation to the Earth's orbit around the Sun. Figure is not to scale [58].*

are more demanding than a typical low-cost cubesat in Low Earth Orbit (LEO). Cubesats are typical low-cost demonstration satellites that have, in the last ten years, become very popular space technology demonstration platforms for universities and commercial companies [19]. Most of them have flown only as secondary payloads on other space missions and they have, so far, only been flying in LEO [45]. As shown in DSI's concept spacecraft (figure 1.1), the small customer spacecrafts are protected from the space environment and supported with electricity by the mothership. In order to keep cost low, it is assumed that the commercial customers will be conducting their research missions with a low budget. This translates directly to limitations on the embedded computer hardware among other things such as weight and complexicity of systems. The budget for scientific instruments on these missions could be smaller than those on governmental missions, such as the instruments on Rosetta mission. Radiation hardened electronics might not be needed if the required lifetime is short [1]. Power generation in free flight might not be sufficient to keep up the level of charge in the batteries unless there are prolonged recharging periods, which then makes the spacecraft more exposed to radiation due to accumulating radiation dose. Power limitations also translate directly to heat management [30]. Sensitive components might require heating to keep them from freezing, and that requires electrical energy. Commercial companies are unlikely to have Radioisotope Heating Units (RHU), which are based on the radioactive decay heat of plutonium, as such substances are strictly controlled and available in very limited quantities and their production is expensive [17].

There are ongoing or upcoming high-budget national space missions to NEAs. Japanese Hayabusa 2 mission, launched in 2014 has a standalone MASCOT lander, which uses

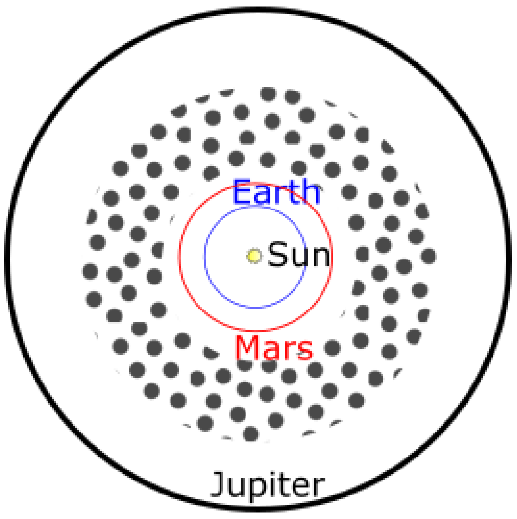


***Figure 2.2*** *Illustration of the location of the asteroid belt between the orbits of planets Mars and Jupiter. Figure is not to scale [21].*

radiometry to determine properties of the asteroid's surface composition [44]. A sounding (tomography) instrument was proposed for the flight but it was not included in the flown spacecraft [34]. National Aeronautics and Space Agency (NASA) is currently flying its Dawn mission, launched in 2007, to dwarf planets Vesta and Ceres in the asteroid belt [60]. NASA is also launching OSIRIS-REx mission to a NEA in 2016 [59, 78]. Neither of these missions have a tomography instrument onboard.

All of the mentioned conditions are even more demanding, if the flights are not flown to a NEA, like the Dawn mission. A large quantity of asteroids reside in the asteroid belt, which is in the area between the orbits of planets Mars and Jupiter. This zone is illustrated in the figure 2.2. Distance from the Sun is in between 2-4 AU:s. Available sunlight intensity is inversely proportional to the square of the distance from the Sun, so the intensity at 4 AU is 1/16th of the intensity at the Earth distance. This available electric power directly also constrains the available data bandwidth to Earth. The distance to Earth is larger so the power needed to transmit the same amount of data in the same time period is also larger [30]. New communication methods, such as laser-based space communications, are being developed to alleviate the communication problem [13] and might be available for use in future.

The current view of the science community is to prefer a large number of inexpensive asteroid prospecting missions to gather some data from the huge number of possible asteroids [24]. This scientific need for data can be contrasted with the small amount of national missions, such as the mentioned Hayabusa 2, Dawn and OSIRIS-REx. The inner contents of asteroids could vary even within the same class of asteroids so any data is wanted [14]. DSI's mothership concept enables these kind of missions. The first target for such a mission is one of the NEAs. Plans for future flights to the asteroid belt have not been detailed. One of the goals of this thesis work is to enable small, inexpensive spacecraft to collect waveform tomography data and process it in a way that makes it easier to store and transmit to Earth.

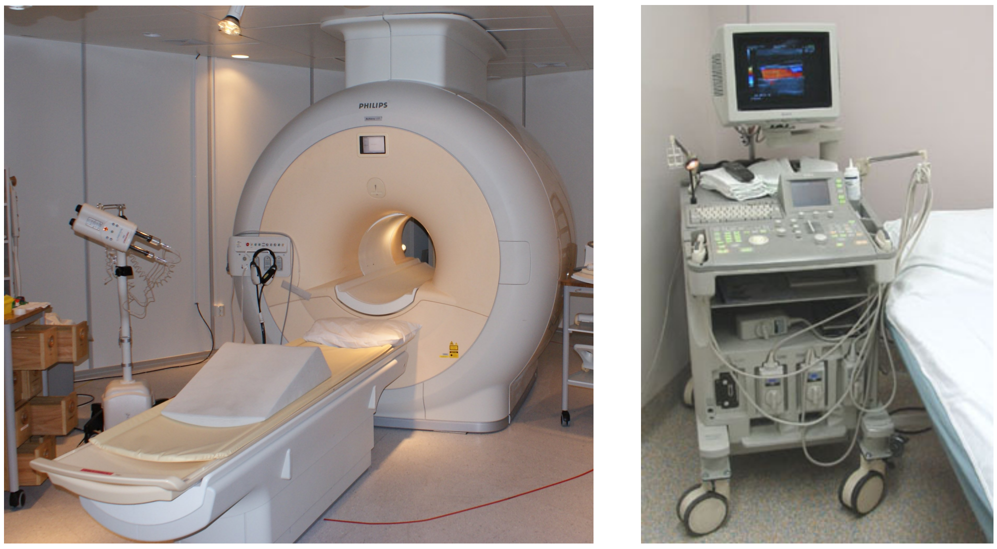
## 2.2 Waveform tomotraphy in civil engineering

The oldest travel-time tomography application is in the field of seismic tomography. Calculation of an earthquake location by measuring the travel times of the seismic waves produced by the earthquake was first envisioned in the 1760s [28]. Developments for this application in recent years has concentrated on reconstructing more accurate local or remote 3-dimensional location and geological data from the received signals [85].

More localized ultrasonic or seismic tomography can be used in many applications. These include, for example, mapping of oil fields and reconstructing the underground structure for underground tunneling purposes [32, 86]. Oil industry uses hardware deep underground that can be inspected in place using ultrasonic travel-time tomography [47]. Measurement instruments for concrete structures are being studied and the current research is directed towards non-desctructive evaluation methods for bigger structures by using waveform tomography [20]. These applications differ in some ways from asteroid tomography as the measurement devices can be left in place for longer time to gather data. Similarly to asteroid tomography scenarios, the measurements can only be taken from certain accessible directions, resulting in signal sparsity.

## 2.3 Modern biomedical engineering applications

In general, usage of tomographic imaging in biomedical field has been popular. Due to medical benefits, imaging methods without using x-rays or other radioactive substances have gained popularity. Examples of these methods include magnetic resonance imaging (MRI) and ultrasonic tomography [75]. MRI uses powerful magnets to polarize hydrogen atoms. This polarization then produces a signal that can be used to produce, in the case of traditional MRI, a 2D image or a sliced view of the subject. A traditional ultrasonic scanner uses the echo of the ultrasound signal to display a similar 2D image.



(a) MRI imaging hardware (b) Ultrasound imaging hardware

***Figure 2.3*** *Comparison of biomedical magnetic resonance imaging [2] and ultrasonic imaging hardware [70]. Images licensed under permission.*

One of the research directions is medical 3D tomography using an ultrasound scanner, which is different to the aforementioned traditional ultrasonic imaging which typically produces a 2D image. For example, screening for, and imaging breast cancer, is typically done with mammography. It uses x-rays to produce a 2D image of the breast. Possible abnormalities can be seen in such an image, but the accuracy is limited into two dimensions unless many images are taken. The patient also receives a radiation dose due to the use of the x-rays. Using an ultrasonic scanner and processing the data using the researched methods, an accurate 3D image of the breast can be recovered without using radiation [12]. Research such as this is popular for other biomedical research applications. The additional benefit of using ultrasound is, in general, the smaller size and cheaper prize of the 3D-capable ultrasonic imaging devices compared to equivalent MRI imaging hardware. MRI hardware is also costly to maintain because of the liquid helium that needs to be replenished to keep the instruments at their operating temperature. Figure 2.3 shows general difference in size of the hardware.

The mathematical methods to model the utrasonic scanner system, its signals and wave propagation being used in modern biomedical engineering are very similar to those being used for asteroid tomography research. The medical field is constantly developing better devices and new methods for imaging. One of the Tampere University of Technology's research groups has, for example, developed neonatal electroencephalography

(EEG). EEG is a method which measures the electrical activity in brain using only a few electrodes attached to the head of the test subject. The electrodes cannot be installed in every direction in regards to the brain, so the available signals to measure the activity are sparse. The research [69] has focused on the improvement of the electrode model used in this application.

# 3. MATHEMATICAL METHODOLOGY

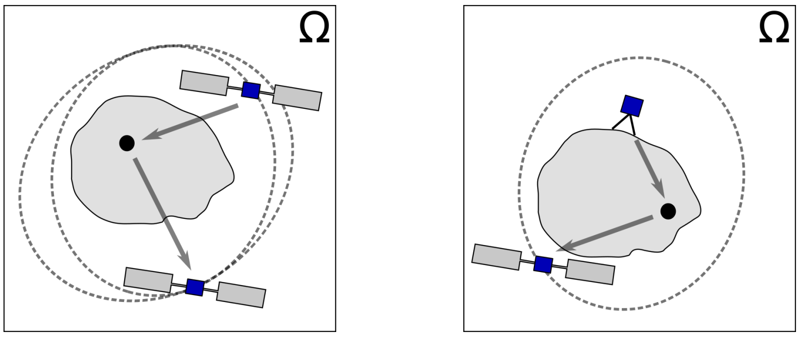
This chapter explains the principles of mathematical methodology needed in tomography imaging via waveform data. Formulas are abbreviated and explained so that the reader should understand them with a basic engineering mathematics background. The mathematical background for travel-time tomography is explained. The chapter briefly refers the recent related research on waveform tomography [67]. This background is important as the same methodology and modeling scheme is used in the context of this work.

The mathematical background of waveform tomography can be divided into a few important aspects. Firstly, the acoustic or electromagnetic waves are modeled via equation system in the section 3.1. The model is developed in such a way that it can be more easily simulated as a system of first-order differential equations. This is then further formulated into a system of only weak form formulas, which have a solution. Then a simulation methodology is selected in section 3.2. The system is formulated into a matrix form so that a finite element mesh can be used to simulate the system. The simulation is performed using Finite Difference Time Domain (FDTD) method [71]. Finally the inversion strategy is selected, but as this work does not relate to the actual inversion of the data, the detailed explanations for the inversion strategy are not included.

This chapter includes the the basics of modeling the full wave system and its data which forms a mathematical basis for using travel-time data in tomography inversion algorithms. For simplicity, the further adjustments needed to use the travel-time data are only briefly referred to and full explanations are in the literature.

## 3.1 Forward modeling

The waveform signal is modeled as a scalar field $u$ presenting a wave, that can be electromagnetic or acoustic. Modeling of the system does not depend on the type of wave used, but in this text, the wave is assumed to be electromagnetic. The computational domain $\Omega$ is assumed to contain the target object of the tomography together with its close surroundings. It is assumed that an electromagnetic pulse is then transmitted through the area. There are different possibilities in this scenario. During the measurements, the transmitters and receivers can be either fixed or moving, touch the surface or be located within a distance from it. In tomography of asteroids the cases of fixed and moving sensors can be associated with lander-to-orbiter and orbiter-to-orbiter cases, respectively. Figure 3.1 illustrates thesel scenarios. The selected mathematical model of the system works for both of these cases.



(a) Orbiter to orbiter (b) Lander to orbiter

***Figure 3.1*** *Asteroid tomography with only orbiters (a) and with a lander and orbiter (b).*

The scalar electric potential field $u$ transmitted by the antenna is assumed to obey the following wave equation system:

$$\varepsilon_r \frac{\partial^2 u}{\partial t^2} + \sigma \frac{\partial u}{\partial t} - \Delta_{\vec{x}} u = \frac{df}{dt} \sum_{k=1}^{S} \delta(\vec{x} - \vec{x}^{(k)}) \tag{3.1}$$

$$u(0, \vec{x}) = \frac{\partial u}{\partial t}(0, \vec{x}) = 0 \text{ for all } \vec{x} \in \Omega \tag{3.2}$$

where $\varepsilon_r$ is the relative permittivity and $\sigma$ is the conductivity distribution of the $\Omega$. The permittivity and conductivity of the $\Omega$ will affect the transmitted signal. When the original signal $\Delta_{\vec{x}} u$ is substracted from the sum of received signals, only the changed signals remain. On the right-hand side of the equation 3.1, the sum represents a set of point sources that can be thought as points that transmit a pulse that will be different to the original signal. For mathematical rigour, it must be assumed that only the set of points belonging to $\Omega$ are sending signals and that there are no other signal sources present.

Both sides of 3.1 are integrated with respect to a time interval [0, T]. New variables are defined as follows: $\vec{g}(t, \vec{x}) = \nabla u(\tau, \vec{x})$. The resulting system is of the following first order differential equation form:

$$\varepsilon \frac{\partial u}{\partial t} - \sigma u - \nabla \cdot \vec{g} = f \sum_{k=1}^{S} \delta(\vec{x} - \vec{x}^k) \tag{3.3}$$

$$\frac{\partial \vec{g}}{\partial t} - \nabla u = 0. \tag{3.4}$$

Assuming that both the electric field $u$ and its gradient $g$ are zero at the beginning ($t = 0$), and multiplicating the functions with arbitary test functions $v$ and $w$ and then integrating 3.3 and 3.4 by parts results in the weak form

$$\frac{\partial}{\partial t} \int_\Omega \vec{g} \cdot \vec{w}\, d\Omega - \int_\Omega \vec{w} \cdot \nabla u\, d\Omega = 0, \tag{3.5}$$

$$\frac{\partial}{\partial t} \int_\Omega \varepsilon\, u\, v\, d\Omega + \int_\Omega \sigma\, u\, v\, d\Omega + \int_\Omega \vec{g} \cdot \nabla v\, d\Omega = \begin{cases} f, \text{ if } & \vec{x} = \vec{x}^k \\ 0, \text{ otherwise.} \end{cases} \tag{3.6}$$

If the domain $\Omega$, the signals at the initial conditions and the model properties are regular enough, then the weak form can be shown to have a unique solution for electric field $u$ [27]. The present approach is unitless and can be scaled for propagation of many types of waves. For example, it has been shown in [67] how to obtain SI-unit values correspondig to $t, \vec{x}\, \varepsilon_r$, $\sigma$ and $c = \varepsilon_r^{-1/2}$ (signal velocity).

## 3.2 Forward simulation

To simulate the system defined by the the weak-form equation 3.5, the domain $\Omega$ is discretized through a finite element mesh $\mathcal{T} = T_1, T_2, \ldots, T_n$. The mesh is equipped with normalized finite element basis functions $\varphi_1, \varphi_2, \ldots, \varphi_n \in H^1(\Omega)$. The discretized solution is assumed to be of the form $u = \sum_{j=1}^n p_j\, \varphi_j$ and $\vec{g} = \sum_{k=1}^d g^{(k)} \vec{e}_k$ . Here $g^{(k)} = \sum_{i=1}^m q_i^{(k)} \chi_i$ is the sum of the element indicator functions $\chi_1, \chi_2, \ldots, \chi_m \in L_2(\Omega)$. Indicator function is a function that outputs either 0 or 1 depending on the input values. Parameter $d$ is the number of spatial dimensions; usually it is 2 or 3.

Restricting the test functions for $v$ and $\vec{g}$ results to a Ritz-Galerkin -type matrix [15] version of the weak form formula 3.5:

$$\frac{\partial}{\partial t}\mathbf{A}\mathbf{q}^{(k)} - \mathbf{B}^{(k)}\mathbf{p} + \mathbf{T}^{(k)}\mathbf{q}^{(k)} = 0, \quad \text{for} \;\; k = 1, \ldots, d, \tag{3.7}$$

$$\frac{\partial}{\partial t}\mathbf{C}\mathbf{p} + \mathbf{R}\mathbf{p} + \mathbf{S}\mathbf{p} + \sum_{k=1}^{d} \mathbf{B}^{(k)^T}\mathbf{q}^{(k)} = \mathbf{f}, \tag{3.8}$$

where $\mathbf{p} = (p_1, p_2, \ldots, p_m)$, $\mathbf{q}^{(k)} = (q_1^{(k)}, q_2^{(k)}, \ldots, q_n^{(k)})$ and

$$A_{i,i} = \int_{T_i} d\Omega, \qquad A_{i,j} = 0 \quad \text{if} \quad i \neq j, \tag{3.9}$$

$$B_{i,j}^{(k)} = \int_{T_i} \vec{e}_k \cdot \nabla \varphi_j \, d\Omega, \quad \text{for} \;\; k = 1, \ldots, d \tag{3.10}$$

$$C_{i,j} = \int_{\Omega} \varepsilon \, \varphi_i \varphi_j \, d\Omega, \quad R_{i,j} = \int_{\Omega} \sigma \, \varphi_i \varphi_j \, d\Omega, \quad S_{i,j} = \int_{\Omega} \xi \, \varphi_i \varphi_j \, d\Omega, \tag{3.11}$$

$$T_{i,j}^{(k)} = \int_{\Omega} \zeta^{(k)} \, \varphi_i \varphi_j \, d\Omega, \quad f_i = \int_{\Omega} f \, \varphi_i \, d\Omega. \tag{3.12}$$

Partial derivatives over time for equations 3.7 and 3.8 can be approximated with finite difference method with $\Delta t$ time steps with N points of time. This results in leap-frog time integration formulas as follows:

$$\mathbf{q}^{(k)}{}_{\ell+1/2} = \mathbf{q}^{(k)}_{\ell-1/2} + \Delta t \mathbf{A}^{-1}\left(\mathbf{B}^{(k)}\mathbf{p}_\ell - \mathbf{T}^{(k)}\mathbf{q}^{(k)}_{\ell-1/2}\right), \quad \text{for} \;\; k = 1, \ldots, d, \tag{3.13}$$

$$\mathbf{p}_{\ell+1} = \mathbf{p}_\ell + \Delta t \, \mathbf{C}^{-1}\left(\mathbf{f} - \mathbf{R}\mathbf{p}_\ell - \mathbf{S}\mathbf{p}_\ell - \sum_{k=1}^{d} \mathbf{B}^{(k)^T}\mathbf{q}^{(k)}{}_{\ell+1/2}\right). \tag{3.14}$$

Thus, using leap-frog integration method [84], a signal propagating in $\Omega$ domain can be simulated when modeled in the manners described in this chapter. This leap-frog method is formally known as the Finite Difference Time Domain (FDTD) method, which is used to propagate a wave equation in simulation [71]. In short, FDTD generally works by first solving electric field $u$ at given time step. After that the magnetic field is solved at the same space or volume at the next time step. The step-by-step process is repeated until the wave has propagated completely or the system has otherwise achieved a steady state.

In the case of electromagnetic waves, the interesting property of the $\Omega$ is permittivity which affects the speed of the wave. Permittivity is handled as a sum of a fixed background permittivity $\varepsilon_{bg}$ distribution and variable perturbation $\varepsilon_r^{(p)} = \sum_{j=1}^{n} c_j \chi'_j$ which were composed by indicator functions $\chi'_j$ over the $\Omega_0 \cup \Omega_1$. The simulation is linearized around background permittivity through a Jacobian matrix $J$. Jacobian matrix is a list of all first order partial derivatives of a vector-valued function. It is formed by differentiating formulas 3.13 and 3.14 over suitable time steps as follows:

$$\frac{\partial \mathbf{q}^{(k)}{}_{\ell+1/2}}{\partial c_j} = \frac{\partial \mathbf{q}^{(k)}_{\ell-1/2}}{\partial c_j} + \Delta t \mathbf{A}^{-1} \left( \mathbf{B}^{(k)} \frac{\partial \mathbf{p}}{\partial c_j} - \mathbf{T}^{(k)} \frac{\mathbf{q}^{(k)}_{\ell-1/2}}{\partial c_j} \right) \quad \text{for } k = 1, \ldots, d, \tag{3.15}$$

$$\begin{aligned} \frac{\partial \mathbf{p}_{\ell+1}}{\partial c_j} = \frac{\partial \mathbf{p}_{\ell}}{\partial c_j} - \Delta t \mathbf{C}^{-1} \left( \mathbf{R} \frac{\partial \mathbf{p}_{\ell}}{\partial c_j} + \mathbf{S} \frac{\partial \mathbf{p}_{\ell}}{\partial c_j} + \sum_{k=1}^{d} \mathbf{B}^{k^T} \frac{\partial \mathbf{q}^{(k)}_{\ell+1/2}}{\partial c_j} \right) \\ -\Delta t \frac{\partial \mathbf{C}^{-1}}{\partial c_j} \left( \mathbf{R} \mathbf{p}_{\ell} + \mathbf{S} \mathbf{p}_{\ell} + \sum_{k=1}^{d} \mathbf{B}^{k^T} \mathbf{q}^{(k)}_{\ell+1/2} \right) \end{aligned} \tag{3.16}$$

## 3.3 Inversion approach

According to [40], if potential data $u$ is measured at a set of measurement points, the following formula 3.17 enables estimation of an unknown permittivity coordinate vector $x$ via inversion methodology.

$$Lx = u + Lx_0 \tag{3.17}$$

The current research is using classical regularization technique, which aims at minimising the regularized objective funcion $F(x) = ||Lx_0 - Lx||_2 + ||\alpha D x||_1$ via the iterative recursion procedure $x_k = \left( L^T L + \alpha D \Gamma_k D \right)^{-1} L^T u$ where $\Gamma_k$ is a diagonal weighting matrix defined as $\Gamma_k = \text{diag}\,(||Dx_k||)^{-1}$. $D$ is a regularization matrix and its entries satisfy

$$D_{i,j} = \beta \delta_{i,j} + \frac{\int_{T_i \cap T_j} \left( 2\delta_{i,j} - 1 \right) ds}{max_{i,j} \left( \int_{T_i \cap T_j} ds \right)} \tag{3.18}$$

where $\delta_{i,j} = 1$ if $i = j$, and 0 otherwise. This simple inversion approach usually converges sufficiently in, for example, 5 iteration steps. The value of parameter $\beta$ determines the balance between the regularization matrices. A small value for $\beta$ leads to inverse estimates with low total variation and larger values can be expected to result in well-localized estimates [66, 67, 68]. In addition to classical reqularization, inversion computations can rely on a statistical approach, which can often increase the reliability and robustness of inverse estimates compared to regularization methods [40].

## 3.4 Path integrals and ray tracing

In addition to finite element methods and FDTD computations briefly explained in chapter 3.2, a forward simulation for the travel time can be obtained via integrals of the form $T = c \int_p \sqrt{\epsilon_r} dl$ where $c$ is a constant and $P$ is a signal path [11]. In this approach, prediction of the signal paths has a central role. This can be done based on Snell's law of reflection and reftraction [66]. Ray tracing is a good alternative for the FDTD method as it requires only a fraction of the computational work that is needed to propagate the actual wave. However, the FDTD approach can be expected to be better, as the linearized leap-frog formulas also include information on how the signal paths change under the perturbation of the permittivity distribitution.

## 3.5 Travel-time calculation

The relative permittivity distribution $\varepsilon_r$ can be recovered using complete wave as modelled in the previous sections, or by using its travel time [11]. In addition to these, for example, the total received signal power or even some other types of schemes can be used. However, the relationship between the permittivity and the power is ambiguous, since, in addition to permittivity fluctuations, also the conductivity distribution $\sigma$ absorbing the wave energy affects the received signal power [11]. Based on the complete wave $u$ the travel time can be calculated as follows [68]. The travel time $\tau_i$ of a signal travelling within a time interval $[T_0, T_1]$ is given by formula

$$\tau_i = \frac{\int_{T_0}^{T_1} t u(t, x_i)^2 dt}{\int_{T_0}^{T_1} u(t, x_i)^2 dt} \tag{3.19}$$

which differentiated with regard to $x_i$ yields a following form:

$$\frac{\partial T}{\partial x_i} = \frac{2\int_0^\infty (t - T)\, u \frac{\partial u}{\partial x_i} dt}{\int_0^\infty u^2 dt} \tag{3.20}$$

This means that the travel time can be linearized using the actual wave data and its linearization. The formula 3.19 equals to summing squares of the received signal values multiplied by their time difference divided by the sum of all the squared signal values. These values can be calculated in discrete time steps for each $x_i$. The electric potential distribution $u$ is the same distribution that is simulated using the wave equation system formulas 3.1 and 3.2. These values for each discrete location $x_i$ can then be used in forward simulation if the system is linearized in such way that the travel time values can be used. That linearized form is shown in [68].

To interface the mathematic model and real signals, the some selections have to be made. These include how to decide the time period $[T_0, T_1]$ and whether or not to use the above integrating travel time formula 3.19 or to use a simpler travel time (delay). During the course of this thesis work, various methods for determining this time period were evaluated, experimented and implemented, which would not have been feasible if the development time of the hardware had been prohibitively long for each version. Of all different types of travel time calculations, these two types of methods for measuring travel time and signal detection methods were focused on:

1. Detect the time period from where the signal reaches a certain percentage of the maximum power level. In this case, the fully integrated travel time value 3.19 is calculated and used. This method is referred to as *integrating* method in further chapters.

2. Detect the time values from where the signal reaches a pre-selected threshold value. This threshold value is set beforehand and the same threshold value is used for all measurement locations $x_i$. Results may be variable. This method is referred to as *thresholding* in the further chapters.

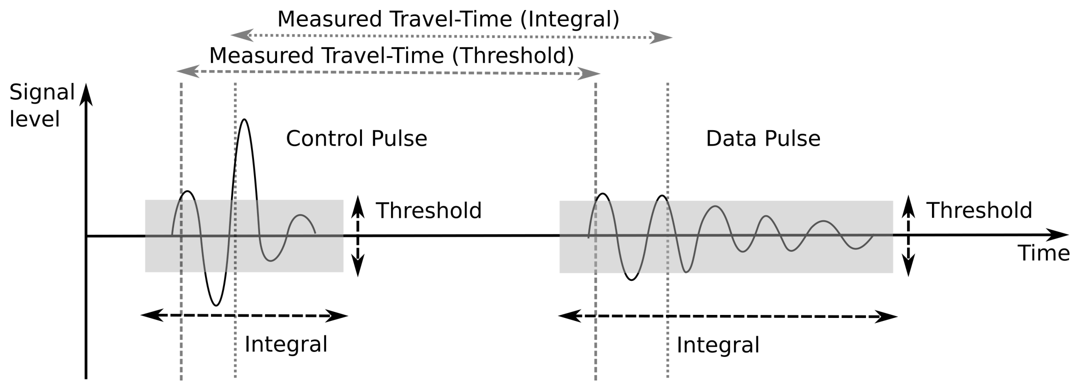
Important aspect for finding a threshold value or determining the power level is the signal level itself. Both of the methods need a certain level which to compare signal values to. Both the control signal and measured signal can be normalized. The normalization can be based on their combined maximum value or a on separate maximum value for each signal. A separate value calculated with prior knowledge of the measurement system can also be used. The effect of different normalization methods isn't extensively studied on this work, but parameters and methods used are documented with the results.

## 3.6 Signal noise

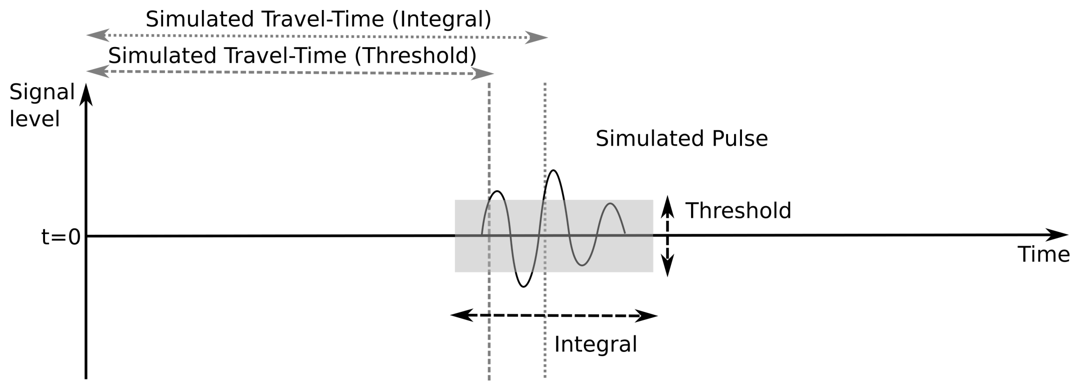
In next chapter, a test setup to gather some realistic acoustic data is introduced. In real world, signals have noise resulting from the digital measurement hardware or by other sources such as echoes or propagation of the signal via paths that are not defined in $\Omega$. Digital noise an be thought as white noise and the other types are systematic noise sources. Mathematically it is not good to have noise in the signal, as it makes the reconstruction of the data more difficult. Real asteroid and comet space missions have been flown, and signals have included noise. For example the first results of the Rosetta mission's CONSERT instrument indicate that the instrument's measurement had 12 dB of noise, and that the achieved signal to noise ratio was 20 dB [43]. This error can be simulated by artificially degrading the experimental data to lesser accuracy, such as reducing its bit depth, or by reducing the accuracy in the calculation of the travel-time value itself. In this work, both of these methods are tried.

In calculating the travel time value, additional length of signal extending the selected time period $[T_0, T_1]$ can be included in reverse direction (before) the start of the received signal detection point. Also the total time period can be shortened so that the noise signals towards the end of the signal are reduced. Figures 3.2(a) and 3.2(b) show these properties of the signal. Figure 3.2(a) also shows how a travel time is measured only if a starting point (control pulse) is also measured, i. e. syncronization in the measurement system has to be achieved. The same or different methods for the detection of control and measurement signals can be used, but experimental data is needed to see how it affects the measurement and the overall inversion results. The measurement need for also the control pulse can be compared to a simulation 3.2(b) where the starting point (t = 0 s) is known by default.

By using any of these methods with adjustments described in this chapter, there is exactly one calculated travel time $\tau_i$ value for each location of received signal $x_i$ for the given signal transmitter position. The original (unaffected) signal also arrives to each location $x_i$ as in formula 3.1 and it also has a travel time. The original signal and its travel time can be simulated and calculated using FDTD simulation method [71] or by simpler ray tracing, and the inversion of the data as shown in [68] can be done by using these travel-time values.



(a) Measured travel-time calculation. A signal containing two pulses has been recorded at an arbitary point in time. The travel-time is the time difference of these pulses.


Simulated Travel-Time (Integral)
Simulated Travel-Time (Threshold)
Signal level
Simulated Pulse
Threshold
t=0
Time
Integral

(b) Simulated travel-time calculation. A pulse has been propagated by simulation methods and it arrives at the receiver at the simulated time. That time is directly the travel-time of the simulated pulse.

***Figure* 3.2** *Visualization of travel time calculation types. Greyed area represent one example of time period included in the integrating method. In the (a) image the measured travel time is the difference between the two signals. In simulation the travel time is known by measuring only one pulse (image (b)). The (b) image also highlights how the methods (integral and threshold) can also be used in simulation.*

# 4. EXPERIMENTAL DATA GATHERING HARDWARE

In this chapter the test setup for data gathering and experimenting with acoustic travel-time tomography is described. The strengths and weaknesses of the aspects of the setup in relation to tomography are discussed. Suitable values to be used in travel time calculation's different parameters are determined based on the physical characteristics of the hardware.

## 4.1 Test setup and scenario

To obtain real experimental data cheaply and quickly, a number of test setups were considered. Test method options were not limited to using acoustic waves. Ultrasonic range finders, signal generators and microphone setups were considered, such as Proceq Pundit ultrasonic measurement products [65]. Different types of test scenarios were studied. Previous works such as [46] had modeled electromagnetic wave travel-time tomography which was then applied in gathering data of a water reservour in [47]. A master's degree work had implemented a whole ultrasonic signal transmitter-recorder laboratory setup and its data gathering system [38]. Implementation of a similar ultrasonic system was deemed too costly for this thesis.

An acoustic setup with one speaker and two microphones was finally selected. A block diagram of the system is in figure 4.1. The recording device is a Mac computer equipped with an external sound card interface running Audacity recording software [8] and Matlab is used to generate a suitable pulse signal to the speaker. Selected audio recording format is 32-bit floating point audio using 48 000 Hz sample rate. This is the best possible accuracy for the hardware. To eliminate the effects of possible signal propagation delays from the Matlab to the speaker and of the distortion of the sound pulse due to the speaker itself, one of the microphones is positioned right in front of the speaker and its signal is saved in the right channel of the recording. This signal is referenced in this chapter as the transmitter. The other microphone is on the perimeter of target area in a known position and its signal is saved in the left channel. This is referenced as the receiver. The time

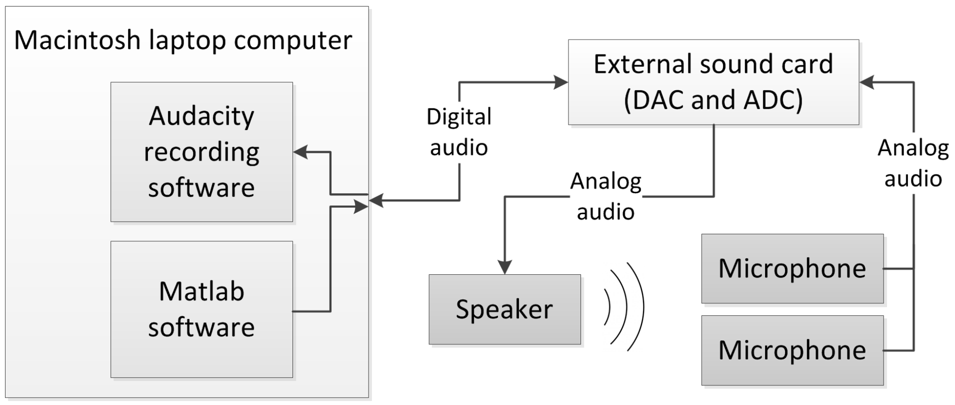

Macintosh laptop computer
Audacity recording software
Matlab software
Digital audio
External sound card (DAC and ADC)
Analog audio
Analog audio
Speaker
Microphone
Microphone

***Figure 4.1*** *Block diagram of hardware used in test setup. Physical interfaces and Digital-to-Analog and Analog-to-Digital converters (DAC and ADC) of the external sound card were used.*

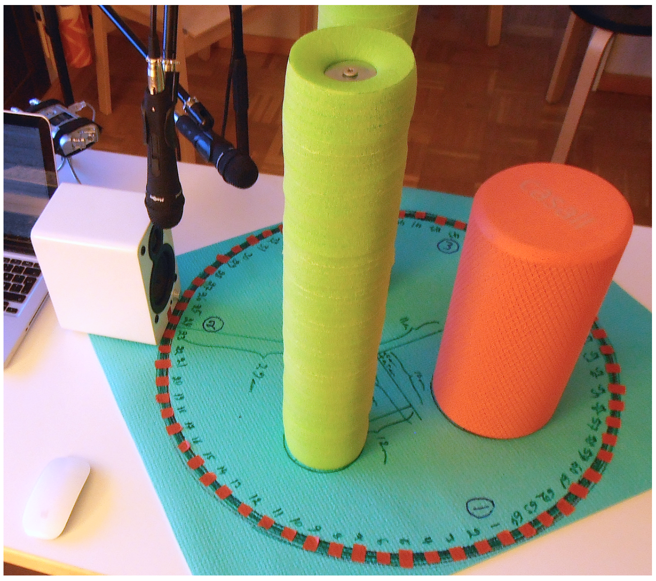


***Figure 4.2*** *Test setup with a speaker and two microphones.*

difference between these signals is used to calculate the travel time as explained in chapter 3.5 and with these two signals the synchronization of the measurement is achieved. Picture 4.2 shows the test setup with both microphones near each other. Technical data of the hardware used in the setup is listed in table 4.1.

The target area has a commercial yoga mat as soft floor covering. The targets themselves are three yoga pillows of different sizes standing in the upright position. The material of these is polyvinyl chloride (PVC) plastic. Mathematically and geometrically easy scenario, where the yoga pillows are apart from each other is chosen as the baseline test

***Table 4.1*** *Hardware used in the test setup*

| **Hardware item** | **Details** |
|---|---|
| Exibel BX-320 Speaker | Maximum power: 20 Watts<br>20 Hz - 18 000 Hz frequency response |
| MadBoy TUBE-022 Microphone | Unidirectional dynamic microphone<br>60 Hz - 13 000 Hz frequency response<br>-72 dB sensitivity |

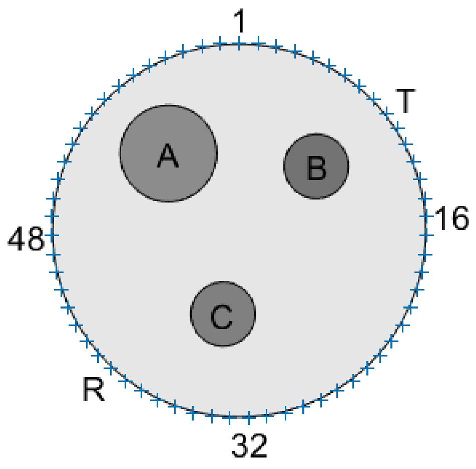


***Figure 4.3*** *Top down diagram of the equipment. Four numbered positions (1, 16, 32 and 48) are numbered to show clockwise ordering. T denotes transmitter, R receiver and A-C the objects. Small crosses on the perimeter are the possible locations of the transmitter and receiver, and an example for possible positions are shown by T (position 10) and R (position 40). The T and R should be on the perimeter, but in this picture, they are moved outside for clarity.*

scenario. Picture 4.3 shows a top down view of the scenario. The perimeter of the area is divided to 64 positions with clockwise ordering. The positions are shown in the picture as light crosses. In table 4.2 the positions (x, y) and sizes of the objects in the scenario is listed.

The process of acquiring the test data is as follows. To record data, speaker and the microphones are connected to the Mac computer via an external sound card and relevant sofwares were launched. Then the speaker&microphone combination (transmitter) is placed in an initial position and the other microphone (receiver) on top of the perimeter ring. For a transmitter position, the sound from the receiver is recorded from each of the positions on the perimeter by repeating the step multiple times and moving the receiver to a different position each time. For the initial dataset, all perimeter recordings by using

*Table 4.2 Size and center coordinates of the targets on the yoga mat. Origo's size is the size of the test area.*

| Target | x (cm) | y (cm) | Diameter (cm) |
|---|---|---|---|
| A | -11.0 | 12.0 | 15.0 |
| B | 12.0 | 10.0 | 10.0 |
| C | -2.5 | -13.0 | 10.0 |
| Origo | 0.0 | 0.0 | 58.0 |

three arbitary transmitter positions 1, 24 and 43 were recorded. Measurements for positions where the speaker and microphone were close to each other was not obtained for the data set, as the receiver signal didn't appear to be any different to the recording of the transmitted signal.

The relation of this test setup in regards to an asteroid tomography mission is the following. Two or more orbiters are orbiting an asteroid with unknown inner structure. One of the orbiters sends a signal pulse at a known position which is then recorded by another orbiter(s) in known position(s), and the resulting signal recordings are then send via low power communications links to a mothership for further processing. This further processing will consist of compressing or other processing, such as calculating travel-time values. In the next chapter, an FPGA implementation for calculation of travel-time data for the data obtained using this test setup is designed.

## 4.2 Analysis of the test setup's error sources

A challenge of using acoustic signals in the test setup is that the signals are not as accurate as those to the electromagnetic or even ultrasonic signals. The signal recordings were made in a typical livingroom conditions using standard commercial hardware. Recordings could be made better in many ways, but using this imperfect dataset puts the mathematical methodology and models into a better test. Sources of errors and imperfections are, in no particular order, listed as follows:

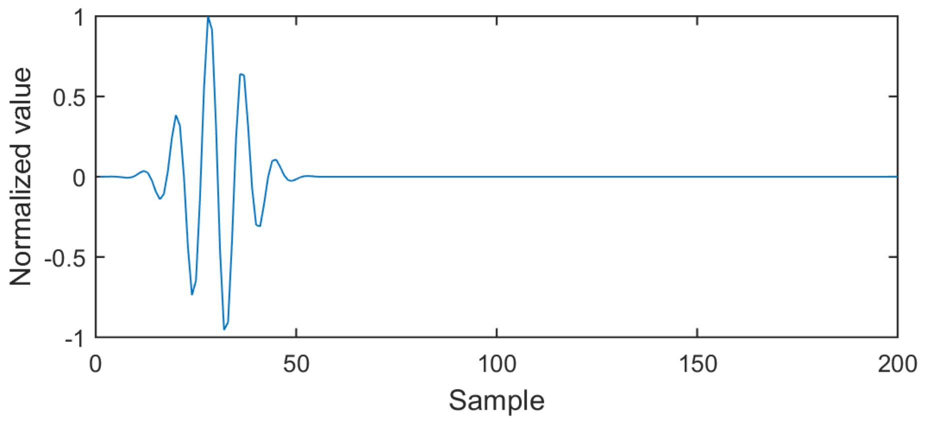
- Reflections of the acoustic signal coming from the mat floor, i.e. the signal does not encounter the objects A-C at all, or gets reflected to the mat afterwards. This exists in all recorded data. Mathematically this is the main area not defined in test area $\Omega$.

- Reflections of the acoustic signal from different parts of the room. The room was sufficiently large compared to the test setup, so these weak echoes arrive very late compared to the real signal, so the induced errors are assumed to be nonexistant.
- Background noise in the room. The background noise is weak and constant, so the induced error is assumed to be insignificant.
- The limited frequency response and resolution of the speaker. This induces a large error.
- Directionality of the sound wave produced by the speaker. The speaker case has a shape which directs sound, so the source of the sound wave is directional.
- Placement accuracy of the speaker&microphone and the receiver microphone. The speakers and microphones were positioned as well as possible with ruler. More accurate placement methods and measurements could be implemented, but induced error is not assumed to be very big.
- Errors from digital reproduction and recording of the played and recorded sound by the computer. These errors are considered to be insignificant compared to other error sources.

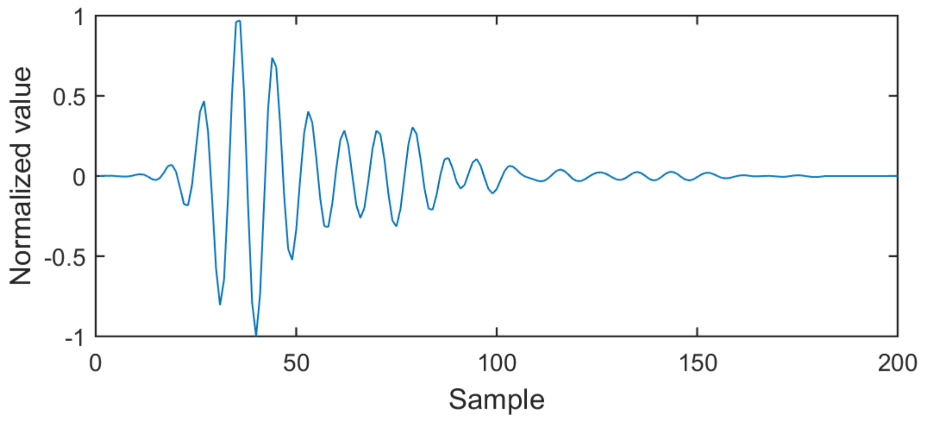
Of these, the limited frequency response of the speaker is assumed to be most significant, as a speaker cannot reproduce a pulse of arbitary sharpness because of the limited frequency response. In addition, the speaker diaphragm will continue to resonate and produce sound after the transmission. The transmitted pulse was engineered to be as sharp as possible for the speaker. A sharp pulse was generated with Matlab with following commands:

```
t = 0:pi/4.2:14*pi;       % sets signal frequency and length
a = sin(t);               % creates signal
f = exp(-1/(2*7^2).*([1:59]- 30).^2); % volume pulse curve
a = a.*f;                 % modulation
p = audioplayer(a,48000); % audio playback at 48000 Hz sample rate
```

This generates around 5.8 kHz pulse modulated with another signal so that the speaker diaphragm has a chance to reproduce it. The speaker used in the test is a BX-320 active speaker. It is a two-element speaker with a low frequency (bass) and a tweeter element. No data is available for the actual crossover frequency between these two elements. For higher quality speaker hardware this would be known or even adjustable. For this test, it was estimated that a 5.8 kHz signal would most likely only come from the tweeter element based on typical audio crossover frequencies obtained from various sources and prior



(a) Transmitted pulse



(b) Actual transmission as recorded from speaker

***Figure 4.4*** *Transmitted pulse and the same pulse recorded from the speaker. The ringing effect in (b) can be seen.*

knowledge. Figure 4.4 shows the transmitted and recorded pulse signal. The diaphragm movement after the actual pulse transmission can be clearly seen in the figure 4.4(b). Using this kind of speaker, the diaphragm ringing induces an error signal (ringing) after the actual pulse is transmitted. Judging by the figure 4.4(b), the noise of about 0.4 times the peak signal value for a duration of around 50 samples. Using the integrating method in travel time calculation (see chapter 3.5), at least some of the error could be reduced.

The second worst error source is the directionality of the sound wave produced by the speaker. The mathematical model as written in chapter 3 assumes that the sound wave starts from a point source and propagates equally in every direction. As the speaker diaphragm is mounted in a certain orientation and installed in a speaker enclosure, the sound wave won't travel at the same intensity in all directions. One of the tasks of a speaker enclosure is to direct the sound towards audience.

The third worst error source is the placement accuracy of the receiver and the transmitter. This is due to some difficulties in measuring the locations consistantly. The speaker is bulky and the mechanical arms (see figure 4.2) holding the microphones are difficult to place with consistant accuracy. On a real asteroid prospecting mission with multiple orbiting spacecraft or landers, the measurement accuracy for the positions of the devices might not be known, so this inaccuracy is a relevant topic.

The placement method of the speaker and microphone has also an effect when calculating the actual data inversion, which assumes that the pulse originates from exactly the perimeter of the target area. With this test setup, the microphone that records the transmitted pulse cannot physically be at the same place as parts of the speaker. As in figure 4.2, it is located right in front of the speaker, with unknown placement accuracy and variation. If the distance from the speaker is, for example, 3 cm, then sound travelling in air at the speed of 343 m/s takes around 0.03 m / 343 m/s $\approx$ 0.00009 s $\approx$ 0.09 ms to travel this distance (based on time = distance/speed, $t = d/v$). At the 48 000 Hz sampling rate which results in a sample time of 1/48 000 s $\approx$ 0.02 ms, this equals to an error of 4-5 samples. This value varies based on the exact speed of sound at the time of measurement and the distance of the speaker. In calculation of a travel-time value for a measurement, this has to be taken into account.

At the beginning of the measurements, it was not sure if the recorded sound signals could be used in reconstructing the target area with the existing mathematical methods. The initial data set with 3 different transmitter positions was recorded and then the test setup was put into storage. This data set represents a minimal data set, and more could have been included. Combination of the error sources result in the recorded data being less than ideal which results in the test case being a difficult inverse problem. Adjusting the simulation model to take some of these error sources into account in the forward simulation is not in the scope of this thesis project. Nevertheless, during the project the thesis supervisor had time to adjust a previously tried Matlab travel-time inversion routine to take some aspects of this test setup in account. Those aspects were, namely, the directionality of the sound produced by the speaker and the speaker case. The result of that work is that it is benefical to model some aspects of the speaker and microphone themselves in the simulation. Another approach would have been to measure the directionality of the sound produced by the speaker with many recordings with just a speaker and a microphone. An equalizer model, parametrized with the distance and angle from the speaker, could have been produced based on those recordings and the actual tomography recordings could have been processed using it.

## 4.3 General basis for the hardware to simulation interface

Calculating the travel time value for a recorded signal is based on the mathematical background explained in chapter 3.5. For the thresholded travel time case, the calculation can be implemented with a very simple algorithm. For the integrating case, the algorithm is only slightly more complex.

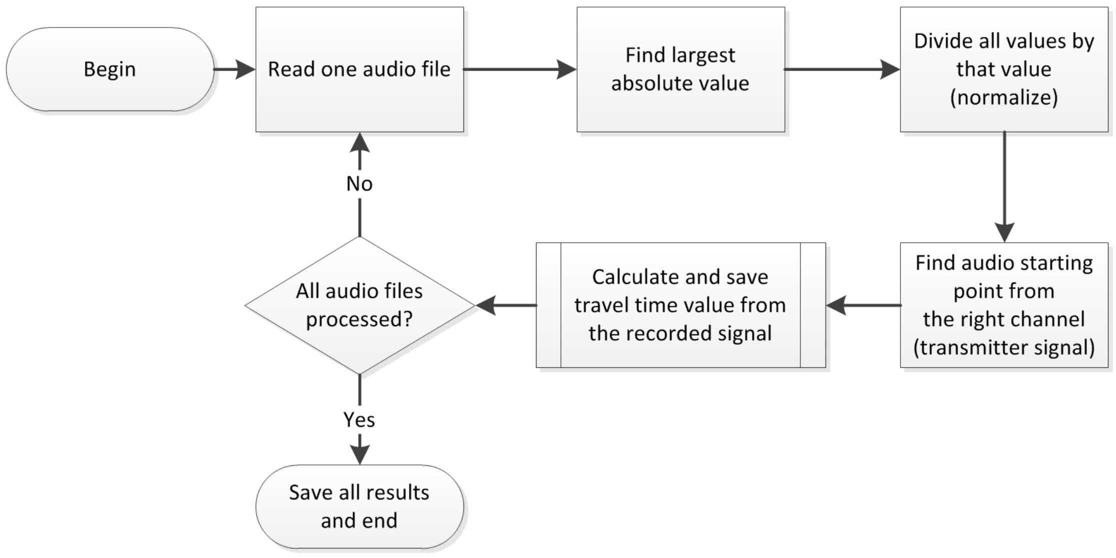

Begin
Read one audio file
Find largest absolute value
Divide all values by that value (normalize)
Find audio starting point from the right channel (transmitter signal)
Calculate and save travel time value from the recorded signal
All audio files processed?
No
Yes
Save all results and end

***Figure 4.5*** *A simplified high level flow chart of the travel-time calculation algorithm in Matlab.*

Initially the travel-time calculation was developed in Matlab. The script read all audio files that were pre-recorded using the test setup, and calculated all travel-time values for all of them. A flow chart of this operation is presented in figure 4.5. The script read the file, normalized both the transmitted signal (control pulse) and the received signal at $x_i$ and then calculated the travel-time value. The high level requirement is to implement the same or similar system using real hardware. There are no hard real-time requirements. It is assumed that as long as the calculated travel-time values are essentially the same as those originally calculated with Matlab, the result is good enough.

There are a few parameters on the algorithm that have been omitted from the figure 4.5. Among those are the threshold value used used to find a starting point in the signal, and the amount of samples before and after that to include in the calculation of the travel time value. In the original Matlab script, some parameters were in a separate parameter files and some were manually changed in the script themselves. The two types of calculations shown in chapter 3.5 are to be implemented. These are the plain travel time value (delay) thresholding that can be easily measured, and the full integrating version that is more complex as it adds the squares of the sample values.

# 5. HARDWARE DESIGN, IMPLEMENTATION, OPTIMIZATION AND BENCHMARKS

In this chapter, the design of the system is described in detail. Overall design process was done in steps over many months. The application of an high-level synthesis to generate travel-time calculator hardware based on Matlab scripts was selected as one of the main goals of the thesis near the beginning of the project. Terasic's Altera DE2 development board [77] was readily available for development. It is rather old, but it offers a stable development platform for evaluation, so it was selected as the board to base the work on. The FPGA chip on the development board was also supported in Catapult. Evaluation of Catapult high-level synthesis tool and its workflow in this application was suggested. At the middle part of the thesis project, some additional future development directions were introduced, and those also affected the work content and design. At that point the features to be implemented were locked in, communication methods and protocols were compared and selected, and the actual calculation modules were implemented.

## 5.1 Design aspects

In general, Matlab scripts abstract a lot of matrix operations. In Matlab, it is possible to use multi-dimensional matrices and perform elementwise operations on them. In hardware solutions, these results in loops of varying complexicity. Loops can duplicate data, which has to be minimized on FPGA to minimize the amount of logic units used. Focusing on these aspects willl also keep the chip area and power consumption low. This mandates careful study of the code in its C language form. The Matlab travel-time calculation script reads all audio files and processes them in a loop. When using C and developing hardware in general, a simpler way to test if things work is to calculate the travel time one audio file at the time. In this work, the calculation is done that way. It could be expanded in a future development of the system if a target application necessitates it.

Matlab calculates using floating point precision by default. The experimental data was recorded using 32-bit floating point precision, which was the highest accurary the hardware was capable of recording. Floating point arithmetics produce larger hardware structures than fixed point calculations. Using audio data converted (in Matlab) to 16-bit, 8-bit or even 4-bit precision, it is possible to simulate limitations in hardware accuracy and the noise sensitivity, which is one of the goals of this thesis work. Reduced accuracy was studied by implementing calculation using only integer arithmetics and by using signal data converted to signed 16-bit and 8-bit accuracy. Studying the effect of these limitations is relevant scientifically, as referenced in chapter 3.6.

Using high-level synthesis to implement this type of calculation module does not save the hardware developer from using at least some traditional methods for the other parts of the design. Traditional methods in this context are designing state machines and other hardware structures in VHDL either via manual coding or by using graphical design tools. These parts can include design of communication methods to and from the FPGA and a state machine to control the process on the FPGA. Those could be developed with high-level synthesis, but in this thesis work, their implementation method does not affect the overall goals and results.

## 5.2 Overall design concept

The interface between the data gathering hardware and a PC running the Matlab software is a possible final result of the design. An overall picture of the embedded hardware project of this thesis work is in figure 5.1. As the experimental acoustic data was gathered separately, there is no requirement for real-time data capturing. In this work, only data preprocessing was implemented. In addition to preprocessor to calculate travel-time values, the target tomography preprocessing type could change in the future. Other types of data preprocessing, such as compressing the data with filtering or calculating a frequency spectrum are potentially viable future research topics. Future development of the system could include an interface to an audio codec chip on the FPGA development board [77] or to another type of data gathering system, such as to components with ultrasonic capabilities, such as the Avisoft's UltraSoundGate bioacoustic recording system [9].

An interface to Matlab is required because the inversion algorithms to recover the unknown parameters from the data are running on a PC and Matlab. In future, those algorithms could also be transferred to run on the FPGA. This could be the case if the system was a commercial standalone product. The main difficulty in the inversion algorithms and Matlab code is the complexicity of the algorithms and the large amount of

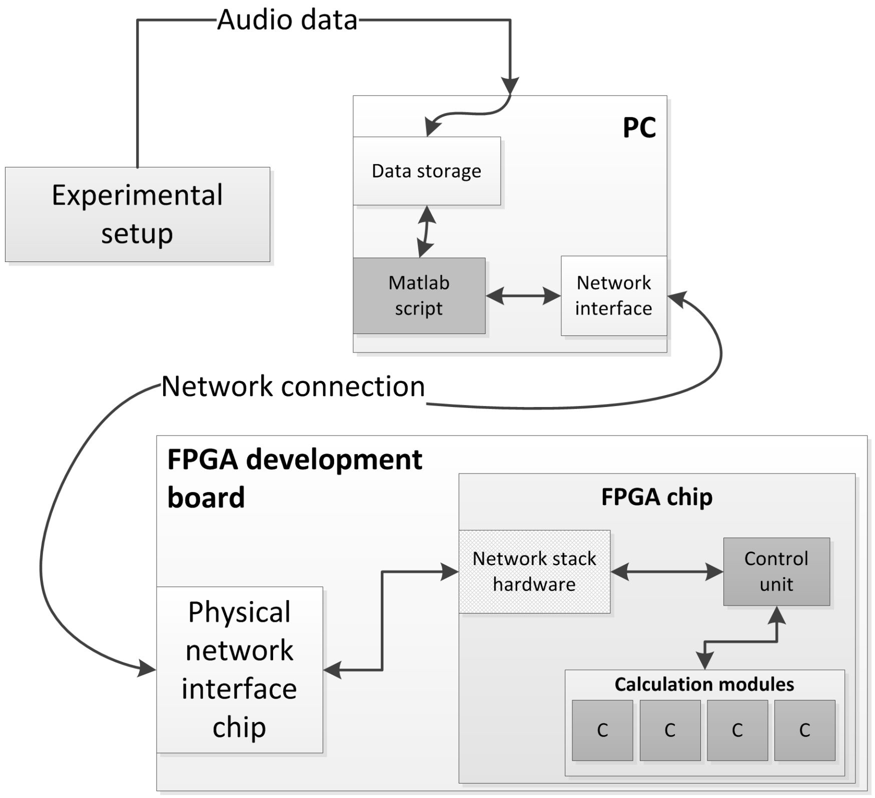


***Figure 5.1** Overview of the embedded hardware project.*

data needed at runtime. Those algorithms include computationally heavy simulation of wave propagation with finite-difference time domain methods. Inversion algorithm can also be computationally heavy, if the tomography problem is three-dimensional and it uses a large data set. In the 2D travel-time tomography case of this work, the computation demands are reduced. Memory consumption of Matlab (gigabytes) when running the inversion routines is still much larger than available memory (a few megabytes) on the DE2 platform. The chip area would be large, as the computations should be performed with as high accuracy as possible by using floating point arithmetics. Specifying suitable bit widths for significand and exponent for these calculations and intermediate results would need a careful analysis. The task of preprocessing the data using hardware generated by high-level synthesis was selected as the first step in this hardware development process.

Memory consumption for a stereo audio file recorded with 32-bit floating point precision with unneeded silence cut out from the beginning and the end can be calculated. Sound traveling at 340 m/s takes around 0.0017 seconds to travel 0.58 m, which is the diameter of the test setup area. At 48 000 Hz sampling rate this equals to length of 82 audio samples. Having extra signal at the at the start and especially in the end is beneficial. By analyzing the audio recordings from the test setup, it was deemed that 500 samples should be enough for every case. The resulting stereo audio data is around 4 000 bytes at 32-bit precision. The data is not needed to be stored in any external memory such as SRAM (static random access memory), but that could be implemented as needed. One reason is that the FPGA can't contain enough of the data because of limited amount of internal SRAM. Other reason is the future development of this design is to record the data itself rather than getting it from Matlab. The data set collected with the experimental test setup contains around 180 audio files, 60 receiver locations $x_i$ for each of 3 different transmitter (speaker) locations which results in total size of around 720 kB for a complete data set. Overall the data set recorded as wave files by the test setup is much larger, as the recordings include seconds of silence at the beginning and the end. The Altera DE2 evaluation board has 8 MB + 512 kB of external SRAM memory [77] and FPGA chip (part number EP2C35F672C6N) has 483 840 bits = 60480 bytes of internal memory [4]. This indicates that the data for a calculation of one travel-time value should fit even on the FPGA chip's internal memory.

Doing calculations with floating point precision results in more complex chip being produced by synthesis tools [29]. Other, not as accurate data representation formats used in standard wave files are 16-bit PCM and 8-bit PCM [39]. PCM is pulse code modulation, in which the samples are stored with at a known sampling frequency, quantisized to equal differences and rounded to nearest digital value. As the data is read from the files by Matlab, the signedness is kept also for 8-bit data calculation. Implementing the travel time calculation module in this way gives an idea how many logic units each implementation consumes on the FPGA. The more inaccurate formats could affect the final inversion results. Using the 8th bit for sign for the 8-bit PCM wastes one bit so the accuracy is the same as 7-bit unsigned.

The only actual requirement is that the calculation of the travel time values works and that the system outputs the same or similar values as scripts in Matlab. There is also flexibility in the data formats. As an example, Matlab can be used to interpret the result values. For example, an integer value 1234 can be interpreted as thousands of a millisecond, giving it a meaning of 1.234 ms. This enables some simplifications in data transfer and calculation. The performance and optimizations of any parts of the design are not vital for mathematical point of view, but investigating how different types of optimizations affect the high-level synthesis of the travel time calculation module give a view of how well the process works.

## 5.3 Interface protocol and hardware

Interfacing the travel-time calculator modules with Matlab requires a few design selections. First a data format has to be selected. One of the options studied is reading raw wave (.wav) files on the FPGA. The standard includes different variations for also other things than those implied from different bit depth of the sample data [39]. Universally working hardware-based wave file reader is not needed for this project. Other possible format is Matlab's own .mat file format, which can be written from and read to in Matlab. There are at least four file format versions in common use for these data files depending on the Matlab version [51]. Calculation results could be saved in .mat file too, and in the wave file case in to a simple text file.

For data transmission, there are many options, such as networking, Universal Serial Bus (USB) data storage and serial communication. USB-based approach would need an implementation or intellectual property usage of the layered USB stack [79] including file system level read/write access. Using a system that uses that method would involve, for example, manual swapping of USB storage media between the system and the PC running Matlab. Communication via a RS-232 serial port [10] would be easiest to implement on the FPGA, as the development boards including the Altera DE2 has a serial port [77]. Not all desktop computers have a serial port and on laptop computers they are even less common. Transfer speeds using a RS-232 serial port speed could also become a limitation for future applications of the system. There are standard interfaces concerning test and measurement instruments, such as Virtual Instrument Software Architecture (VISA), which includes software interfaces and physical hardware specifications for connecting instruments to computers over network [37]. Matlab Instrument Control Toolbox has support for this standard over a variety of physical interfaces as well [53].

In this design work, the selected transfer method is User Datagram Protocol (UDP) [64] over a dedicated local area network interface on the PC. With this type of live interface, the system would be more usable as there is no need for external usb memory sticks. Also it would be more extendable because there would be no speed issues. Local area network is fast enough for even large data sets. By using a dedicated network interface, it can be assumed that no UDP packets are lost because of network issues such as congestion. This was also tested by running two Matlab instances on a local network to see if any packets are actually lost. Matlab scripts used functions from Matlab Instrument Control Toolbox [54]. The result of that brief test was that no packets were lost at nominal home network loads. The media access control (MAC) [36] addresses for the PC and the development

***Table 5.1*** *Data transmission protocol.*

| Command | Purpose | Format |
|---|---|---|
| Reset | Reset system to initial state | R |
| Send calculation mode | Set calculation mode | M[s] where [s] is either i or t (integrating/thresholded) mode. |
| Send normalization mode | Set data normalization mode | N[x] where [x] is either 0 (normalize in regards to this file) or 1 (normalize the file by using a maximum value provided by a parameter). |
| Send parameter | Set a parameter | P[x]_[v] where [x] is parameter name and [v] its value. |
| Send format | Sets input data format | F[x] where [x] is a number 0-1 meaning 0 = 8-bit PCM and 1 = 16-bit PCM. Other formats can be added if needed. |
| Set length | Set number of samples in audio data | W[n] where [n] is the number of samples. |
| Send right data | Send audio samples for the right channel (one each in following packets) | R |
| Send left data | Send audio samples for the left channel (one each in following packets) | L |
| Start calculation | Calculate travel time using inputted data and parameters | S |
| Receive result | Receive result value from the system | =[value] where [value] is the result in thousands of a millisecond |
| Receive error | Receive error from the system | E |

board are known, so there is no requirement to implement address resolution protocol (ARP) [63] at the FPGA side. Those can be preset at the hardware synthesis time. The ARP table can be manipulated manually on the PC.

### 5.3.1 Protocol

Data transmission protocol for the system is a custom protocol, which enables setting of the calculation mode, the number of samples (length of the audio data), a few parametres

***Table 5.2*** *Parameters and their default values. Refer to figure 3.2 for illustration.*

| Parameter | Purpose | Default |
|---|---|---|
| length | Number of samples of audio in the file | 500 |
| n1 | Number of samples used to correct the difference between the control signal recording point and actual perimeter for test area | 5 |
| n2 | Number of samples to step forwards from measured signal's detection point | 250 |
| n3 | Number of samples to step back from measured signal's detection point | 0 |
| th | Thresholding level in percentage of normalized signal value for the signal detection | 70 |
| p | Normalization target percentage | 100 |

for calculation, and the actual transmission of data itself. The protocol is described in table 5.1 and parameters and their default values in table 5.2. This simple protocol is designed to be extendable and comparatively easy to implement on a hardware-based state machine or a small microprocessor such as NIOS II synthesized on the FPGA. Performance of the data transfer is not very optimal with this kind of protocol because of large overheads from the UDP packets. The VISA architecture could perform better.

The commands and payload data is transmitted as follows. The calculation mode, parametres and length can be sent in any order. Any repeated commands overwrite an older setting, and if no values are sent, the default value is used. Audio data can be sent either left or right channel first. Data must be sent in the format set by the mode command before the calculation can be done. After sending the start command to the system, the result will be received in Matlab. All commands and numeric values are sent and received as ASCII strings with a possible minus sign (-) included. For example, setting a number of samples to 250 would result in these characters being transferred:

```
Hexadecimal values          ASCII
0x4C, 0x32, 0x35, 0x30      L, 2, 5, 0
```

Possible error situations are handled as follows: If audio data has not been send at all for both audio channels in the format defined by the format command, the error command is received in Matlab. If more data than specified by set length command is received, the extra samples are ignored. If less data is sent, the missing values from the end are set as zeroes. Any unknown commands, parameters or invalid parameter values are responded to by sending an error to Matlab on PC. Also the system does not take an empty signal into an account and the results using that kind of values are not defined. This means that in regards to error handling, the design of the system is not very complete. Integrated system testing would be needed to see which error situations are encountered in general use. Empty or too silent signals were not encountered by using the acoustic test data, but they could be possible when using other types of wave data. Error handling could be improved by sending a status back to Matlab after reception of each command, but this would complicate the Matlab script.

### 5.3.2 Matlab code

Matlab script that sends and receives data uses network communication functions served by Matlab Instrument Control Toolbox [54]. A basic script script reads an audio file into array. Then it asks user for the data processing mode, audio data format and parameters. The final user input is for pressing enter to start. After that, a reset command is sent, followed by parameters, length, format and mode commands. Audio data is send for both channels, followed by the start command. After this, Matlab immediately starts to wait for the reception of the result command, or an error command. The error command is the first one in the reception buffer if there was an error command sent any time in the process.

After testing the concept on two running Matlabs on local network, the script was expanded to read a set of audio files into an array. Then the parameters were asked like in the one file case, and the calculations performed on the FPGA. The result values were saved into another array with the same indexing scheme as the array containing the audio files, and the array was saved into Matlab data (.mat) file. For ease of identification of the result files, the name for the files was set to include the method for calculation (integrating or thresholded travel time), audio data bit depth and all the parameters used. An example file name for this file naming scheme:
`results_integ_8bit_th90_n1_5_n2_200_n3_300.mat`.

### 5.3.3 Hardware interface on the FPGA

The FPGA development boards have a physical network interface chip (often called just PHY), which functions as the electrical interface between the network and the FPGA chip itself. The DE2 board has a 10/100 megabits per second (Mbps) capable PHY, which is connected with to the FPGA with a general processor interface [76, p. 43]. Other boards can have different types of PHY interfaces and for a custom printed circuit board designs the options include the most simple solutions that employ the I2C serial bus, such as the W5500, which also has integrated TCP/IP and UDP layer [83].

At the time the architecture selection for the system was made, there was knowledge of an available interface implementation for a FPGA development board. It was assumed that it would be available for the DE2 development board. This affected the design. The existing interface was revealed to be an implementation for another type of FPGA development board. One other GNU General Public License (GPL) licensed core was found at OpenCores, but it was also for other type of FPGA development board [3]. Solution to achieve the desired functionality has been developed for a slightly newer version of DE2 development board by a team of students at Cornell University[73]. Documentation of that work shows that the development team consisted of two persons, the project took 5 months, and that there were a lot of problems that remained unsolved at the end of the project. Suitable commercial intellectual property blocks (IP blocks) such as Altera's own IP blocks are available to achieve the connectivity for a commercial product, if needed [7]. As such, it was desided to defer the development of the ethernet interface between the physical network controller on the development board and the state machine controlling the calculation modules to a later project.

Comparing different types of solutions for a network interface there seems to be two different types of architectures available. Some of them are designs that offer all the needed communication layers implemented in hardware, either on the FPGA or on a separate PHY chip. The student project [73] connects the PHY to a bus on the FPGA and at least some of the layers are implemented on a NIOS II processor also connected to the bus. As this solution is open-source, it would be possible to extend it directly to include data transfer to the calculation units.

## 5.4 Computation unit

This section discusses the workflow from Matlab to hardware. Solutions are briefly compared and then the selected method is documented in more detail. Some aspects of Catapult tools in this application and some general problems that were encountered during the project are presented. The detailed description of the Catapult workflow is not documented here. Good tutorials, such as [48] and [82] are available online, and the High-Level Synthesis Bluebook [29] can be used for reference.

### 5.4.1 Matlab to C workflow options

In general, there is more than one method for producing hardware description (HDL) from Matlab. In this work the selected method is the direct implementation of the same functionality in synthesizable algorithmic C language. Other promising method would have been Matlab's HDL Coder workflow [52], which would have propably produced synthesizable hardware description directly from Matlab. Based on the workflow tutorial available from Matlab website [50], the HDL Coder workflow gives the designer some limited insight on what type of design is actually produced and how it performs. Catapult's reporting system gives the same information, such as a visualization of the critical path in a better form. The HDL Coder tutorial also hints that it is possible to simulate the design directly with FPGA in the loop. The details of this functionality were not studied in this thesis.

The original travel-time calculation script implemented in Matlab has a lot of things that have to take into acount. Firstly, Matlab variables are used without any data types, although Matlab uses commonly used datatypes in the background [55]. Converting the Matlab script to use directly the correct types of variables does not save time here as almost the same effort can be used to code the same functionality in Catapult C, and at that point, the data types would have to be redefined again. Matlab has a tool called Codegen, which can be used to output a function implemented in Matlab directly in C [49]. Experiments with that functionality to speed up development of the code in C was performed with a simple data normalization script, which had an array of values and two loops to find the biggest value and to normalize it. The resulting C language code was functional and readable. Testing it further by, for example, replacing a `for` loop implemented for finding the largest value in the matrix with Matlab's own `max()` function made it barely readable. Extending it with a few variables made it too difficult to read. Matching of loop iterators and variable names to Matlab code functionality were the main issues. This is important because the Matlab scripts include large amounts of functions that abstract loops. The functionality of the program was still the same as the Matlab script, as expected. The original travel time calculation scripts in Matlab also read all the audio files for an entire data set into arrays all at once, which makes even the Matlab scripts and their indexing constructs somewhat difficult to read at times.

Based on these points, the fastest method to convert these types of calculation modules for a single travel-time measurement using high-level synthesis was deemed to be to just implement them directly in algorithmic C language. This means that the implementation was done side-by-side, looking at Matlab code and implementing the same functionality in Catapult. In this way the loops and code structure could be controlled and that way the high-level synthesis implementation result can be taken into account.

### 5.4.2 Catapult C workflow and code

The side-by-side Matlab to Catapult C workflow decision has a big downside. The Catapult High-Level Synthesis Bluebook, which is the how-to manual for the tool and the algorithmic C language, strongly suggests that a C testbench must be first written for the code [29]. In this type of implementation method, the original C-code does not exist and the all the testbenches for different bit width and calculation mode implementations are missing. Testbenches were implemented, but rather than using a golden C/C++ design to compare the result, input values and a golden result value was calculated in Matlab and hard-coded into the testbench. Initially some simple simulations were also made in Modelsim simulation software without an actual testbench. Catapult also has an option in the workflow to generate Matlab executables to be used within Matlab from the developed, synthesizable algorithmic C code. These could then be used on Matlab as testing method. This functionality was not tested in the scope of this project as the results were verified via other means.

The basic version of the C code was implemented once, and then adapted for each of the different bit depths and calculation modes that the work was aiming to test. The resulting C files were then further developed each as their own Catapult C project. This enabled isolating the different designs from each other, although Catapult's hierarchial design allows the designer to do this for each code file separately. The selected method is not optimal. Each of new things learned about Catapult toolchain and suitable coding style had to be transferred to many files and each of the changes made in the original

Matlab scripts quite late during the project had to be repeated in equally many files. Some of these issues wouldn't have been encountered if the files had been in the same Catapult project.

Two types of interface designs were tried for a calculation module. First a complete table of input data for each channel was tried as the parameter of the function like in the following definition, where only one parameter is included for clarity:

```
ac_int<9, false> calculate(ac_int<8, true> data[500])
```

The 500+500 sample target resulted in this being mapped to memory interface by utilizing the FPGA chip specific libraries. The setting for the threshold of this memory interface generation could be set up in the Catapult tool settings and also the input and output mapping of function could be manually adjusted. Accessing data with pointer enables data being retrieved with a for-loop to local registers where it can be manipulated. This results in a very large chip area, which warrants a study of which method is better for the given design. Example of this type of function definition:

```
void calculate(ac_int<9, false> *result, ac_int<8, true> *data_ptr)
```

Catapult was then able to combine the loop needed to retrieve the samples through the pointer with other loops in the design, so there was no significant overhead as such from this operation. Also Catapult was able to determine if pointers were used for inputs or outputs, so the result could also be given via a pointer. Short comparison with these different interfacing styles indicated that both were easy to use. Initially the pointer method and retrieving the data was selected, but it became evident that it would duplicate data into registers or memory. This would consume chip area or double the amount of memory. These duplications should be avoided. The table as parameter format was selected along with other function parameters and return values set as pointers. The table would be automatically mapped to memory if it was sufficiently large as in case of the sample data in this work. Ultimately the SRAM is meant for storing data, either on the FPGA chip or on external memory chip.

The Matlab scripts transferred to C had many loops. They were mainly inferred from Matlab functions like `find()`, `abs()` and `sum()` when they were called for arrays of audio samples. Catapult C gives the designer three options for optimization of loops. These are loop unrolling, loop pipelining and loop sharing (referenced in the previous paragraph)

[29]. Almost all loops in the travel time calculation scripts implemented in C code go through all the 500 data samples. Loop unrolling, which generates own hardware for a set amount of loop iterations is not efficient to use in this application. The memory interface became slower when unrolling was tried. If the overall throughput rate was a factor for the designer, then the unrolling would have been beneficial. Optimizing memory reads to be sequental would be essential. Loop pipelining is feasible to enable for some loops. It decreased the throughput time and thus overall module worked faster, but for nested loops it had to carefully studied how often to start a new outer loop iteration in the pipeline. It is interesting to note that the loop combining was enabled by default in Catapult. This resulted in confusion at the beginning as some loops seemed to just disappear at Register Transfer Level (RTL) generation.

### 5.4.3 Problems encountered

Getting to know the Catapult C language details and the Catapult toolchain brought up a few general issues that were encountered more often than issues stemming from the process of learning the toolchain's possibilities. These issues were unoptimally synthesizable loops copied over from Matlab optimizations, determination of realistic bit depths for some intermediate variables, dividing a result with a large value, efficient division algorithms and memory consumption of the Catapult tools themselves. Catapult software version used in the project was Catapult University Version 8.0/226676.

For high-level synthesis, all the loops must have clearly defined starting and ending points [29]. For example, a `for` loop with iterator `(int i=0; ; i++)` would result in undefined number of maximum iterations and the synthesis tool does not know how many loop iterations to take into account. This type of problem wasn't encountered in the project, as the maximum number of iterations was almost always 500. The issues with loop iterators had to be taken into account. The Matlab scripts had some optimizations that only worked in Matlab. An example of this is a case where a pulse was first found in the control signal, the iterator index was saved, and the search of a pulse from measured signal (on the other audio channel) was started at this index. Catapult didn't know how many iterations the loop was to perform at most. This could be set manually, but the better way was to remove the Matlab optimization and just start to search from the beginning. The control pulse always preceded the measured signal pulse in the tomography signals, which allowed to remove this optimization. There is still room for further optimization.

Catapult tools do not offer a way to calculate a maximum bit width for an arbitary integer number. Example of this is the divisor in equation 3.19 from chapter 3.5, which has a sum of squares of numbers for a part of the signal. Knowing the bit width of the element (signed 8 or 16 bits) and thus the maximum value ($2^7 - 1$ or $2^{15} - 1$), the width of the square could be calculated. The difficulty comes from summing the values, where a realistic upper limit of the number of squared values cannot be calculated. The worse case calculation of the case where the signal is constant at a maximum value was used. Because of this there is room for optimization in here. More evaluations with the tomography measurement data would be required to optimize this value.

Many operations in the Matlab scripts, such as normalization, require divisions. Depending on the bit width of the data, the divisor's bit width could be quite large. For the integrating travel-time calculation, there is a huge divisor that first sums the squares of values and then divides a sum with that value. This also originates from equation 3.19. Catapult C seems to generate all divisions as combinatorial logic if written directly as in typical C code. This gives a false sense of flexibility and the designer has to know what is being generated. Catapult uses accelerated vendor libraries which enables the utilization of hardware multipliers on the FPGA. There is no such possibility for the division operator, so in general, division by a variable will result in a slow combinatorial block. Catapult C has a math library `mgc_ac_math.h` which has synthesizable, more algorithmic division operators for integers. The existance of these more efficient algorithms for division was known. The library is not clearly mentioned in the High-Level Synthesis Bluebook [29], but it is found in the Catapult C library folders. Changing all divisions to use division operators offered by this library improved the results, and for most division operations resulted in a working design where combinatorial dividers did not. The division algorithms use `for` loops, which also requires some loop optimization work when the functions are used. These `for` loops make pipelining the loops difficult. The division loops, such as this division loop and outer signal value normalization loop, become nested loops, and the iteration count for the inner for loop is unknown. Pipelining the outer loop didn't have a noticeable effect to the performance because of this, as the worst-case division delay had to be accounted for.

Trying things such as loop unrolling for loops with large number of iterations, such as 500 also took a large amount of time. If too many loops were unrolled in the design, the software eventually crashed often. Thus, memory consumption of the tool is an issue, but not a significant factor in this project. Catapult was tried on a few different Windows 7 computer installations with either 4 GB or 8 GB of memory. Crashes were not hard to produce with the trial and error knowledge of when they might happen. Crashes or software freezes also happened usually after 4-5 solutions when developing the code and testing out different parameters in Catapult. A solution here refers to a step-by-step completion of a workflow after a parameter or a code change that affects the design. The possibility of these crashes had be taken into account and development had to be divided into sessions of suitable length. Crashes also lead to loss of the reports and generated files for the solutions that was worked on. That didn't help the designer in determining if the solution and its settings where better or worse than before. In a Catapult project, is seems that saving only the code files was not enough. User also had to manually save the project file, which was often forgotten. Longer running tasks couldn't be reliably left running on this version of Catapult, because the achieved results were not saved in case of a crash.

## 5.5 Control unit

The system depicted in figure 5.1 has a control unit inside the FPGA. During this project, a complete control unit was not designed or implemented. This is because of the control unit's tasks are heavily dependent on the design and implementation of the network interface hardware on the FPGA and those tasks were not performed. Nevertheless, some parts of control unit were tried, implemented on both Catapult and traditionally via a graphical VHDL state machine design tool. This control unit is the part of the design that has to be very much changed if the system is further developed to record the data using the audio codec chip or another type of device. This extensibility requirement means that it shouldn't be optimized too much for this thesis work.

The main task of the control unit is to serialize/deserialize and save the data coming from and transmitted to Matlab via the network interface using the protocol designed in chapter 5.3.1. The main reason, however, to try to implement some type of control unit was to learn how to interface the calculation modules generated by Catapult with other hardware. The control unit prototype tested on Catapult didn't have methods of sending nor receiving the data from Matlab. The prototype was built for storing every needed type of data into memory and passing it to the right calculation module, brought to the same Catapult project environment. This taught about possibilities in hierarchial design in Catapult tools and it also gave an idea how many logic elements the control unit implementation using high-level synthesis might consume.

# 6. RESULTS

In this chapter, the results of the project are presented. These include pictures produced by travel-time inversion algorithm scripts using different types of data preprocessing parameters and a summary of the developed travel-time calculation modules.

## 6.1 Inversion results

Figures 6.1- 6.3 show test area recovery (inversion) results. These figures consist of image pairs. Outlines of the three yoga pillows are superimposed on top of the images. For each image the left image is the actual result image. From that image, an area with strongest values is collected so that it has the same size (area) as the yoga pillow's size in that general location. The right image shows only those areas colored with solid black colour. The Relative Overlapping Area (ROA) is the ratio of this area overlapping the actual area of the yoga pillow and $\mathrm{ROA}_{\mathrm{min}}$ is the worst result of the three pillows.

The figures were produced by using a prototype Matlab routine which was adapted from previous work by the thesis supervisor. The script uses ray tracing to propagate the simulated original signal from the speaker and FDTD methods with 3 iteration steps in inversion to recover the image. The regularization parameters $\alpha$ and $\beta$ (refer to section 3.3) used in the inversion are both 0.01. The effect of these parameters was not extensively studied. The inversion script was updated to take some aspects of the experimental data gathering setup 4.1 into account, as described in chapter 4.2. To evaluate data processing artifacts, bit depths of signed 16 and signed 8 bits were selected to be evaluated to see how noise in the data affect the image recovery result. Calculation of data using floating point arithmetics would show the best possible recovery results without any data processing effects. After a few experiments it was concluded that the results from a 16-bit signed integer travel-time calculation module shows the same recovery results (both as pictures and Relative Overlapping Area (ROA) percentages) as the original calculations with floating point arithmetics on Matlab. For that reason, the floating point result column is omitted here, and data is presented for cases with integer arithmetics.

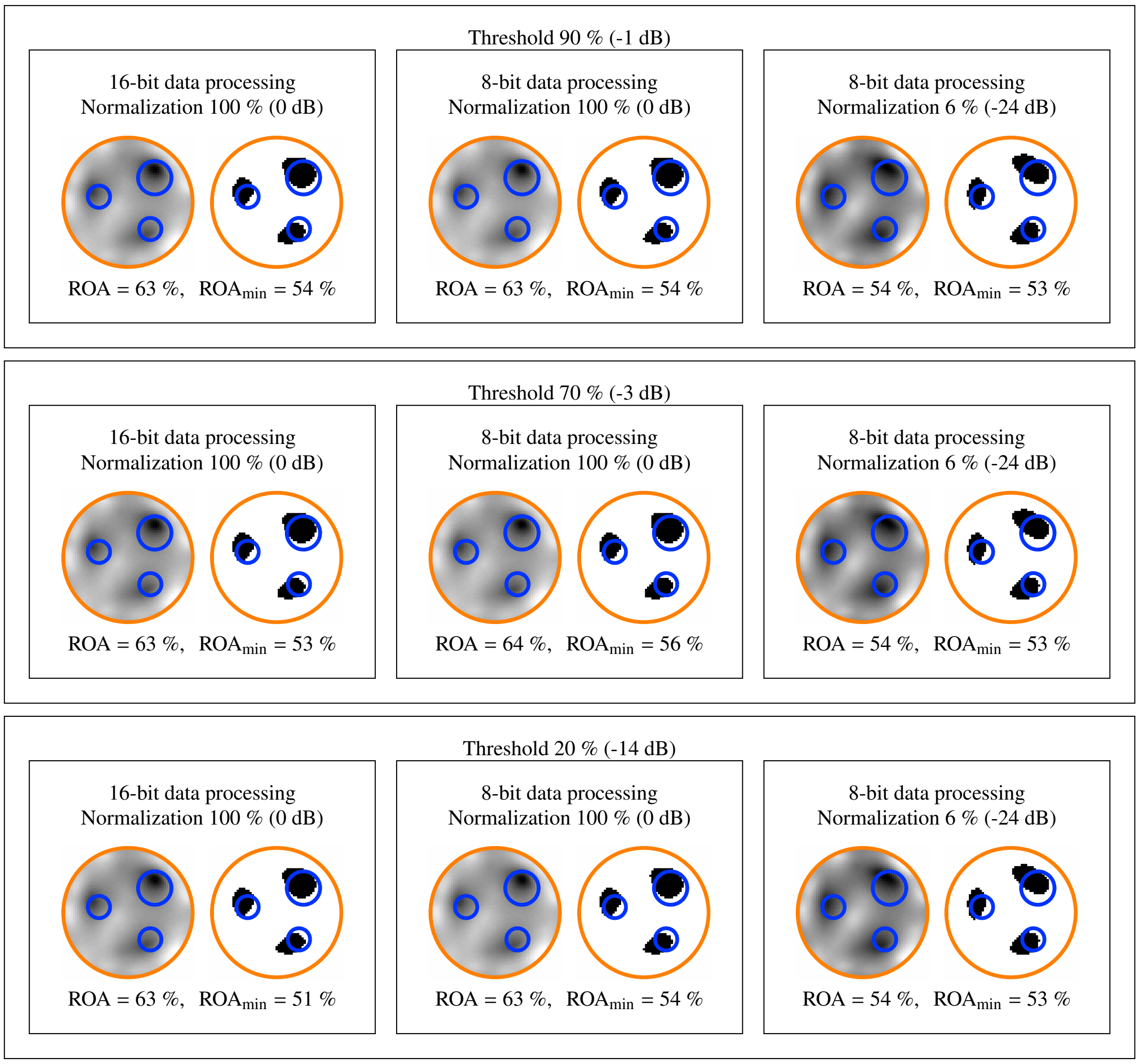

Threshold 90 % (-1 dB)
16-bit data processing
Normalization 100 % (0 dB)
ROA = 63 %, ROA$_{min}$ = 54 %
8-bit data processing
Normalization 100 % (0 dB)
ROA = 63 %, ROA$_{min}$ = 54 %
8-bit data processing
Normalization 6 % (-24 dB)
ROA = 54 %, ROA$_{min}$ = 53 %
Threshold 70 % (-3 dB)
16-bit data processing
Normalization 100 % (0 dB)
ROA = 63 %, ROA$_{min}$ = 53 %
8-bit data processing
Normalization 100 % (0 dB)
ROA = 64 %, ROA$_{min}$ = 56 %
8-bit data processing
Normalization 6 % (-24 dB)
ROA = 54 %, ROA$_{min}$ = 53 %
Threshold 20 % (-14 dB)
16-bit data processing
Normalization 100 % (0 dB)
ROA = 63 %, ROA$_{min}$ = 51 %
8-bit data processing
Normalization 100 % (0 dB)
ROA = 63 %, ROA$_{min}$ = 54 %
8-bit data processing
Normalization 6 % (-24 dB)
ROA = 54 %, ROA$_{min}$ = 53 %

***Figure 6.1*** *Results with integrating travel time used for both the simulated signal and measured signal.*

Signals used in the inversion routine to recover images of the test area were normalized. Measured signals were normalized to maximum measured signal level of the whole data set and the control signal was normalized to its maximum value for each individual control signal recording. 16-bit preprocessing was evaluated at 100 % normalization of the measured signal and 8-bit at 100 % and 6 %. The 6 % represents an extreme case where the signal noise is large, which simulates a weak reception of transmitted signals or an insensitive receiver. For these calculation versions (columns in the pictures), different threshold levels for initial signal detection were evaluated. These were 90 %, 70 % and 20 % levels, corresponding to around -1 dB, -3 dB and -14 dB of the maximum value of a normalized signal. These are presented as rows in the pictures.

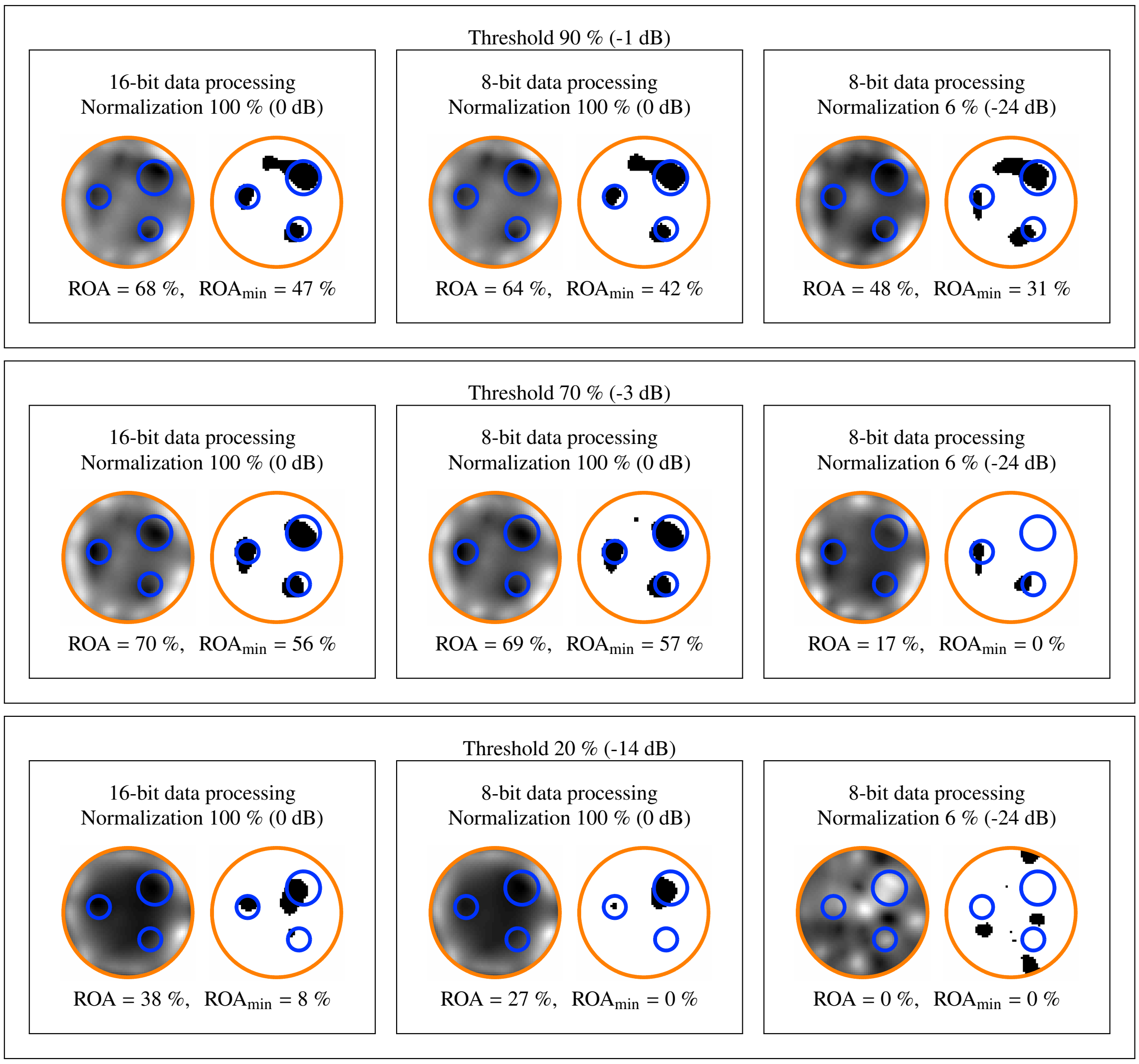


***Figure 6.2*** *Results with integrating travel time used for the measured signal only. The simulated signal was only thresholded.*

Parameters affecting the determination of the integrating time interval $[T_0, T_1]$ of a signal were as follows. Compensation for the control signal recording position (parameter *n1*, see table 5.2), which was slightly inside the perimeter of the test setup's target area was 5 samples. From a threshold detection point, 250 samples were included in the forwards direction (towards the end of the recording) and maximum of 50 samples in the backwards direction (towards the start of the recording). These are the parameters *n2* and *n3*. Generally workable starting values for these parameters were selected and the effect of these values were not studied extensively.

Figure 6.1 shows a case where both the simulated original signal and the measured signal

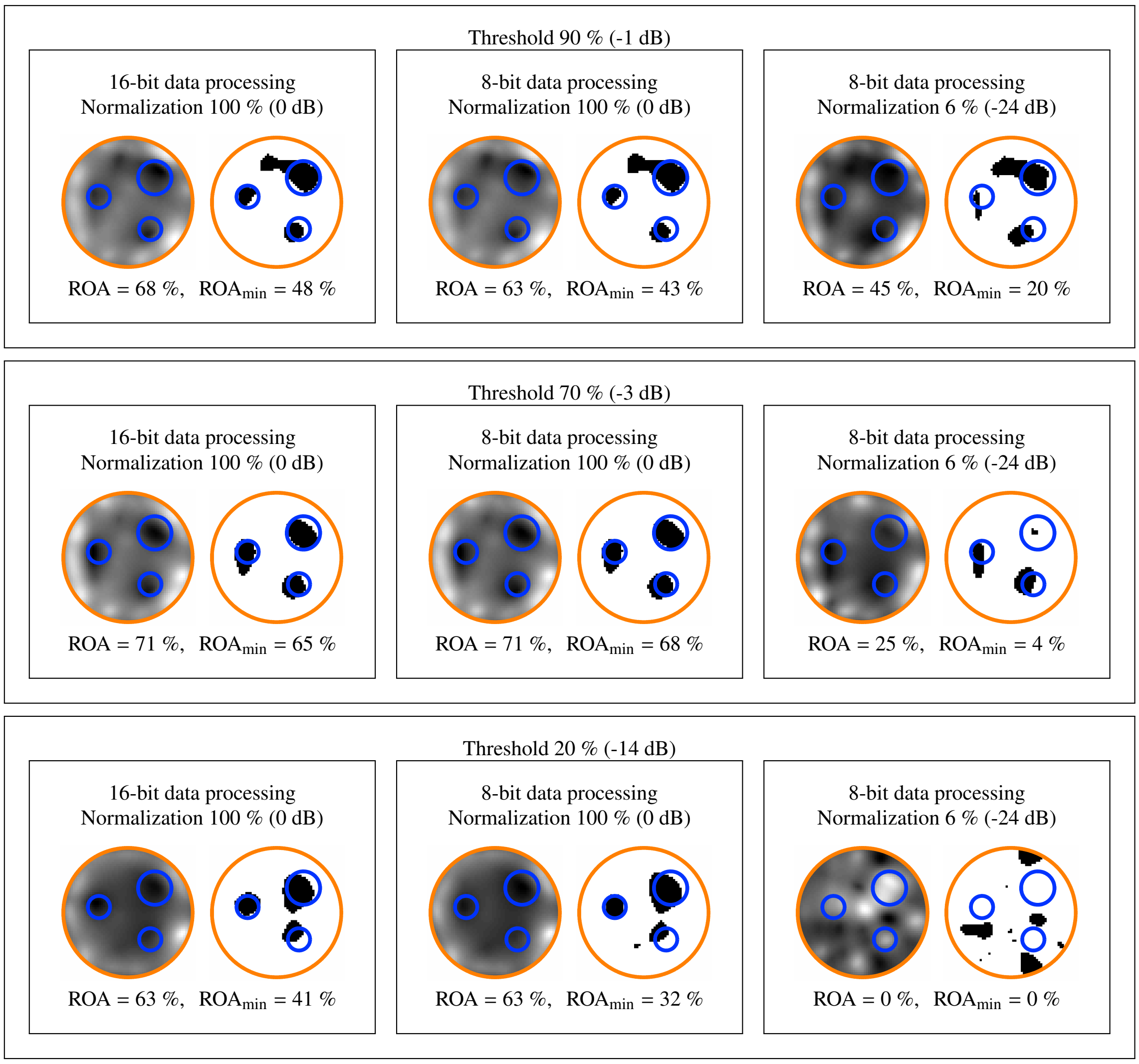

Threshold 90 % (-1 dB)
16-bit data processing
Normalization 100 % (0 dB)
ROA = 68 %, ROAmin = 48 %
8-bit data processing
Normalization 100 % (0 dB)
ROA = 63 %, ROAmin = 43 %
8-bit data processing
Normalization 6 % (-24 dB)
ROA = 45 %, ROAmin = 20 %
Threshold 70 % (-3 dB)
16-bit data processing
Normalization 100 % (0 dB)
ROA = 71 %, ROAmin = 65 %
8-bit data processing
Normalization 100 % (0 dB)
ROA = 71 %, ROAmin = 68 %
8-bit data processing
Normalization 6 % (-24 dB)
ROA = 25 %, ROAmin = 4 %
Threshold 20 % (-14 dB)
16-bit data processing
Normalization 100 % (0 dB)
ROA = 63 %, ROAmin = 41 %
8-bit data processing
Normalization 100 % (0 dB)
ROA = 63 %, ROAmin = 32 %
8-bit data processing
Normalization 6 % (-24 dB)
ROA = 0 %, ROAmin = 0 %

***Figure 6.3*** *Results with simple thresholded travel time used for both the simulated signal and measured signal.*

used integrating travel-time calculation as explained in chapter 3.5. It can be seen that the ROA results are relatively stable for processing with both bit depths and also with much degraded data. There is only a 12 % difference between the the best average recovery value and the worst value. The worst recovery result is found at 100 % normalization, 20 % threshold value and 16-bit travel-time calculation.

Figure 6.2 shows the same results when using integrating travel time value only for the measured signal. The sensitivity to the threshold value can be seen as the ROA percentages start to drop with 70 % and 20 % threshold values. At the worst, the ROA is 0 %, i.e. no recovery was possible. Figure 6.3 for a case where only thresholding was used in the travel-time detection. It shows that using only thresholding produced the best results. ROA was 71 % for both 16-bit and 8-bit processing when using 70 % threshold, but the results are worse when the bit depth and data quality decreases. Overall, the intermediate version (figure 6.2) produced the worst results, which indicates that the same method should be used in simulation and in measurements.

Overal, the recovery of the test object locations on the target area by using the inversion methods described in chapter 3 seem to work. In this chapter, the results are presented in such way that the difference between integrating travel-time and thresholded travel-time methods is evident. From these results, it can be deduced that the integrating travel time method is more stable than thresholding method if the signal quality decreases, especially if the signal is normalized to a low value simulating a weak signal. This can been seen from the stability of the relative overlapping area (ROA) percentages (53 - 64 %) for each of results in figure 6.1 compared to figure 6.3. Although the results cannot be directly compared to other works, these percentages are comparable to other results achieved by the research group at TUT. In [67], the best ROA percentage was 71 % by using and inverting the full wave data. With travel-time tomography and using the thresholding method the best recovery result of exactly the same 71% was achieved. The integrating travel-time method was only only 7 % less accurate. The most important result is that the integrating method is more stable when data preprocessing was performed using less accurate data especially when simulating a weak signal.

## 6.2 High-level synthesis results

In table 6.1 the final results of the synthesis for the calculation modules is presented. The results are the values given by Altera's Quartus II tool after compiling the designs, placing and routing them and after running timing analysis [6]. In general, the results represent designs that were optimized in the same way regarding loop unrolling and pipelining, based on experience of different loop behaviors. The designs used memory interfaces from the vendor libraries provided by Catapult. This had an effect when loops were unrolled, making the designs slower and larger in chip size, so loop unrolling wasn't used. Optimizations enabled were loop pipelining for the unnested division operation loops and loop combining for channel normalizations. The main optimization mode in Catapult's hierarchial constraints settings was selected to be the chip area.

The results include the number of logic elements and the maximum clock frequency. The number of logic elements is also given as the percentage of available logic elements on

***Table 6.1*** *Synthesis results for the travel-time calculation modules implemented with Catapult C. The percentages given for area is the percentage of available logic elements used on the FPGA chip (EP2C35F672C6N) of the DE2 board.*

| **Travel-time calculation mode** | **Bit depth** | **Speed (Mhz)** | **Latency (cycles)** | **Area (Logic Elements)** |
|---|---|---|---|---|
| Thresholding | 8 | 106.0 | 13 594 | 821 (2.5 %) |
| Thresholding | 16 | 99.4 | 13 558 | 1 075 (3.2 %) |
| Integrating | 8 | 74.8 | 13 565 | 891 (2.7 %) |
| Integrating | 16 | 82.6 | 13 581 | 1 253 (3.8 %) |
| Combined system | | 51.9 | 54 235 | 4 176 (12.6 %) |

the DE2 development board's Cyclone II FPGA chip. The latency (in clock cycles) is also shown. It should be noted that the 8-bit and 16-bit SRAM usages are combined when instantiating the system into one hardware synthesis project. This combined system is also presented in the table.

The designs presented in the table are by no means completely optimized. The designs are the result of first round of learning process in regards to high-level synthesis using Catapult. The designs work and produce correct results and they have some, but not all error situations taken into consideration. Impossible normalization percentages, signal threshold levels and signal levels completely below the threshold detection level are detected as errors. The next round of design optimization would involve design partioning using Catapult C's `AC_Channel` data flows. By partitioning the designs into smaller units they could get faster and loop control logic could be further optimized [29]. From the table it can be seen that the current method of instantiating the designs into one project is not optimal and makes the design very slow. Design partitioning would move the workflow away from the direct implementation of the Matlab script, but this work shows that small designs can be implemented in this direct way. Catapult might not have crashed as often with more partitioned structure of the design.

# 7. DISCUSSION

This chapter is divided in two sections. First the mathematical results, such as tomography results and further development paths in terms of mathematical modeling and research are discussed. The second part discusses the results of hardware aspects of the project.

## 7.1 Tomography

Tomography applications differ from each other by the speed of the wave used. The relevancy of the acoustic travel-time tomography in this work can be compared to other applications by comparing the wave propagation speeds and feature diameters. The speeds depend on the type of the wave (electromagnetic, acoustic, ultrasonic etc.) and the material it travels in. The feature size that can be detected depends on ratio of the total size of the target area $\Omega$ and the wave length $\lambda$. The area in this regard can be the diameter of a concrete beam or height of the ionosphere, for example. The relevancy of the acoustic travel-time tomography results, such as the the stability of integrating method used in calculation, can be estimated in other tomography applications by comparing these values.

Table 7.1 is a summary of these type of comparisons. This paragraph includes the source data and reasoning behind the values in the table. In ultrasonic material testing for concrete the velocity of the wave is 3500 m/s [41]. On typical ultrasound frequency of 100 kHz wavelength of the signal is $\lambda = v/f = 0.035$ m and the diameter of the concrete beam being tested could be, for example, 0.3 m. A fault or crack inside the beam could be 0.03 m, for example. That results in a size/wavelength ratio of around 0.86. For electromagnetic waves, a relevant term to compare could be the relative permittivity $\varepsilon_r$. An asteroid can be approximated as consisting of sand. A possible dielectric constant for sand is $\varepsilon_r = 4$ [35]. Speed of electromagnetic wave propagation through material can be approximated roughly as $v = c_0/\sqrt{\varepsilon_r}$ where $c_0$ is the speed of light in vacuum. That gives approximation of the speed of the wave in sand: $1.499 \cdot 10^8$ m/s. The Rosetta mission used 10 MHz sampling rate in its CONSERT instrument measurements [42], which equals to $\lambda$ of 15 m. If the diameter of a feature or anomaly inside the asteroid is 15 m, then the ratio

***Table 7.1*** *Relevancy of the acoustic tomography results based on the parameters of tomography applications.*

| **Tomography** | **Signal velocity** $\boldsymbol{v}$ **(m/s)** | **Signal frequency** $\boldsymbol{f}$ **(Hz)** | **Wave length** $\boldsymbol{\lambda}$ **(m)** | **Feature diameter** $\boldsymbol{d}$ **(m)** | $\boldsymbol{d/\lambda}$ | **Relevant** |
|---|---|---|---|---|---|---|
| **Acoustic test** | **343** | $\mathbf{5.8 \cdot 10^3}$ | **0.059** | **0.10** | **~ 1.7** | |
| Asteroid tomography | $1.499 \cdot 10^8$ | $10 \cdot 10^6$ | 15 | 15 | 1 | Yes |
| Material testing (concrete) | 3 500 | $100 \cdot 10^3$ | 0.035 | 0.3 | 0.86 | Yes |
| Medical ultrasound | 1 500 | $1 \cdot 10^6$ | 0.0015 | 0.015 | 10 | Yes |
| Ionospheric tomography | $300 \cdot 10^6$ | $200 \cdot 10^6$ | 1.5 | $100 \cdot 10^3$ | ~ 670 | No? |

of these is 1. Medical ultrasound imaging uses frequencies in the range of 1 - 20 MHz [18]. Using the 1 MHz value and the speed of 1500 m/s for ultrasound propagation in human tissue [16], the wavelength is 0.0015 m. Breast tumours are classified by their size. A T1 class tumour can have a maximum diameter of 0.02 cm [74]. Using a size of 0.015 m here, the ratio of the wavelength and size is 10. Here it has to be noted, that for this tumour case, the feature size is the tumour rather than the size of the breast. In tomography of the ionosphere, the speed of the electromagnetic wave is around the speed of light, $300 \cdot 10^6$ m/s and a typical frequency used in tomography is between 120 Mhz and 400 Mhz. Using a value of 200 Mhz the wavelength is 1.5 m. The size of the ionosphere is up to 1000 km from the surface of the Earth and a typical vertical feature to be recovered can be 100 km, for example [80]. Repeating the comparison for this tomography scenario, the ratio of wavelength and size of the $\Omega$ can be 670. This suggests that tomography of the ionosphere might be too different in characteristics to acoustic tomography.

The current mathematical research has multiple directions for study and implementation to continue research in. Mathematical method development could include usage of a combination of travel-time and full waveform data in the inversion. That could improve the accuracy of the detection of, for example, dense surface layers on asteroids as depicted in figure 7.1. The signals reflect back very strongly from those layers, and using travel-time the reflection can be detected more accurately. Calculation of travel-time values

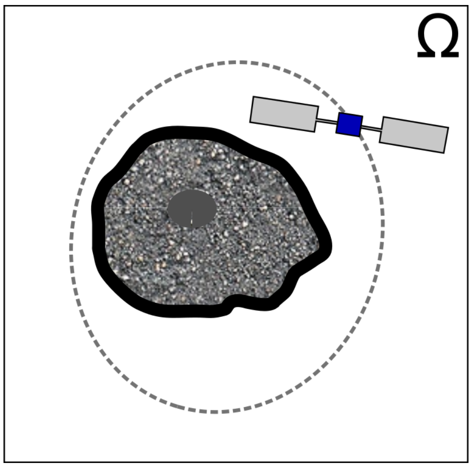


***Figure 7.1*** *Asteroid with a strong surface layer.*

is only one data preprocessing method that can be used, and mathematical methods to accomodate other types of filtering, such as compressing the signals in frequency domain could be developed. In addition, the inversion method development is still ongoing.

On asteroid missions, the spatial measurement and time synchronization (temporal) accuracy of the spacecraft locations has not been analyzed. Fixed signal sources (landers) can be imaged provided that the orbiters or the mothership [23] have a camera. For orbiters some other means of determining their positions (orbits) might be needed. The locations of the orbiters have to be known at the time of every measurement. This strongly suggests that a separate position measurement system for asteroid orbiting spacecrafts could be needed. Based on the difficulties in determining positions manually in the test setup in this work, at least the mathematical model's sensitivity to position errors should be analyzed. That research could give answers on how accurate positioning is needed on an asteroid scouting mission. Rosetta's CONSERT experiment used a simple time syncronization method where the time synchronization was encoded in the signal between the Philae lander and the Rosetta probe [11]. The same type of system for time synchronization could be used for the cases of multiple orbiters, but the overall arrangement would be more complex.

Mathematical research of asteroid tomography is in need of realistic asteroid models. In a real asteroid, the surface layer topography has a relation to the wave length used in tomography. There are many types of asteroids. Inner contents could resemble ruble piles or solid rock. Algorithmic generation of these realistic models (surface and inner contents) based on available asteroid data would benefit the tomography research.

## 7.2 Hardware

High-level synthesis was used to generate hardware. Initially at the project start, a brief look at the available high-level synthesis tools and languages was performed. Usage of Catapult C was expected to be easier than some of the other tools due to the similarity of the algorithmic C language to regular C syntax. This expectation proved to be correct.

Using Catapult to directly implement the Matlab script functionality directly produced results fast, which was expected. Small changes in the mathematical algorithms and Matlab scripts during the project were implemented into the actual hardware designs usually within minutes. Larger changes required more time to analyze the effect of the changes. The changes would have been impossible to implement as quickly by using traditional tools. Catapult tool does not directly offer comparative analytical help in quicky analyzing different optimizations without actually having the tool calculate the results. There is a useful table in the software, which shows the results of different workflow runs (solutions) of a project. The user has to dig into various report files to see if the detailed results changed into better or worse. Having the testing and verification integrated into the workflow does save time. The difficulties in the project included traditional issues in generating suitable test inputs and outputs for the different modules. These were generated from Matlab by manually reducing the bit depth of the values. The testbenches done in this project were not by any means complete. Even the Matlab script didn't take into account empty inputs or other conditions, which hardware designers should take into consideration for the systems to always return defined outputs.

The implemented travel-time calculation modules have lots of room for optimization, because all features of the tool were not extensively studied. Retrospectively it would have been nice to get deeper into the workings of high-level synthesis in Catapult C. For example, the better utilization of data busses with Catapult C:s AC Channel datatypes [29] might have benefitted the design. Catapult also has support for SystemC:s `bigint` variable types [57], which could be useful in determining the bit depths or abstracting the task away from the designer. This could help in saving a big sum of squares result into a temporary variable.

Analyzing the results presented in table 6.1, it can be seen that the integrating travel time method is slower but only slightly larger in chip size. As the speed in general wasn't the target of optimization in this work, it can be said that by implementing the integrating method was worth the additional chip area, as evidenced by the mathematical results obtained via the travel-time values calculated with that method. It can also be seen that the designs do fit on the Cyclone II FPGA chip on the DE2 development board. Memory on the chip is an issue. For further continuation of the work towards an integrated commercial product, an FPGA chip with larger onboard memory and demonstration board for it would be required. Careful analysis to find a fitting chip should be performed for a commercial product. The amount of logic elements used in total (4 176) can be compared to the smallest Cyclone II chip version, which has 4 608 programmable logic elements [4]. The modules wouldn't leave much area for any other hardware on the smallest Cyclone II chip. In general, the complete design including the control systems and interfacing hardware would fit onto a Cyclone II FPGA chip, but it isn't recommended for new designs by Altera. The Quartus II tool, which is used to program the FPGA chip with the designed hardware, has now dropped the support for the old Cyclone II platform [5]. This gives additional motivation for exploration of suitable development platform for the future.

The project ended with lots of open development paths. One of them is to change some parts of the design into a system that collects all the data and does preprocssing, such as computing the travel-time values, and transfers everything to Matlab, including the full wave data recording. In addition to the travel-time values, other types of data processing, such as spectral analyses or filters, could be implemented. Figure 7.2 shows this kind of system built to connect to an FPGA or Digital Signal Processor (DSP) development board. It includes servo motors and servo motor drivers, power supply and some other needed hardware. The receiver is rotated with another stepper motor around the rotating table by an arm under the table. Servo motors could skip steps, but careful selection of hardware can mitigate that. Assuming no slipping of the mechanical interfaces, the transmitter and receiver can be placed with a known accuracy. This setup could be extended to include limited movements in the Z axis as well, making it somewhat limited 3D tomography test setup.

The transmitter and receiver could be based on ultrasonic or electromagnetic devices. Based on this system, a full commercial laboratory instrument could be built. Based on further brief tests with more difficult shapes for target objects, the acoustic travel-time tomography did not produce as good results as the circular objects tested in this work. Because of that, the design of this future development path should focus at least on the ultrasonic instruments. The research group already has some limited second-hand experience with ultrasound systems [68] and aspects of air-coupled ultrasonic systems have been studied elswhere, for example in [33]. Commercial application of such as system, if developed, could be non-touching material sample testing and the samples in this case could be anything ranging from concrete to human tissue or body parts.

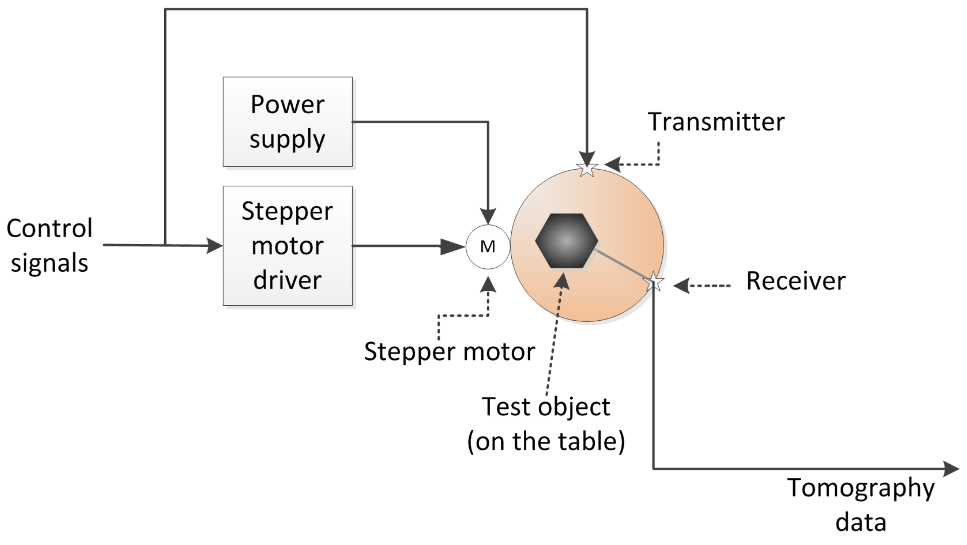


***Figure 7.2*** *Elements of a possible future development of the data gathering hardware including a rotating table, stepper motors and control signals.*

High-level synthesis such as Catapult could be used to built a synthesizable network interface stack to the PC running Matlab when implementing this type of device. That would be a good further learning experience of the toolchain and its possibilities. Making sure it or at least a minimum implementation is synthesizable on the University Version of Catapult and releasing with a suitable open source license would benefit many FPGA developers. Generic design might be difficult to achieve due to differences in the PHY hardware on various FPGA evaluation boards from different manufacturers. The inversion routines used in this work currently exist only in Matlab. Implementing them in algorithmic C would be a large project. Synthesizable hardware could be generated if the project was to be started. Careful analysis of required computation accuracy and the resulting hardware size would have to be performed to estimate if the hardware would fit on an FPGA. Both the ultrasonic and electromagnetic versions of the test setup would need much larger sampling rates resulting in increased memory consumption, and the inversion algorithms would have more data to work on. That would also result in additional memory increase. As an example, the inversion routines to generate the result pictures from the acoustic 2D test setup consumed around 2 gigabytes of RAM at runtime.

In regards to asteroid missions flown into NEA:s, the governmental missions of NASA and European Space Agency (ESA) are likely to have sufficiently large budgets to implement deterministically generated hardware, in which the designs can be formally verified to minimize risk of failure. For the commercial scouting missions, which are not likely to be as complex and strongly funded, high-level synthesis introduces a way to generate FPGA hardware more productively and cost-effectively. Failure tolerancy could be implemented via other ways, such as by redundant hardware or by preferring to launch sufficiently large number of low cost spacecrafts. High-level synthesis has also been studied in generating redundant hardware [72]. It has been used to automatically generate triple modular redundancy, which has triple area overhead. The research has focused on methods of giving the control of the amount of tolerancy to the designer to reduce this large overhead.

A timescale of a space mission to a NEA or a comet is long. From the hardware feature specification freeze it could take anywhere from 1 to 10 years before a mission is launched into space. The space flight travel time to the target, such as a NEA, could take another 1 to 10 years, depending on the target and the launch time. As an example, the Rosetta mission was initially approved in 1993, launched in 2004 and arrived at its target in 2014 [31]. If the mathematical algorithms and possibilities are developing as fast as they have during this thesis project, it remains to be analyzed if implementing the signal preprocessing with hardware designs fixed completely at the time of spacecraft's launch is the right thing to do. The hardware project undertaken during this thesis work showed that the mathematical methodologies must be solidly tested and the hardware has to be flexible. This is because the mathematical details and Matlab scripts changed many times, and the hardware had to follow the changes.

# 8. CONCLUSIONS

In this work, the primary goals of evaluation of the high-level synthesis for travel-time calculation and evaluation of the hardware aspects in relation to travel-time tomography were accomplished. List of all of the project goals and results along with a self-evaluation of the success are listed in table 8.1.

***Table 8.1*** *Summary of the results of the project and self-evaluation of completion of targets.*

| **Goal** | **Result** | **Success** |
|---|---|---|
| Evaluate high-level synthesis workflow | Synthesis workflow was evaluated and some common issues were found. | **80** % |
| Design a basis for sensor-to-software interface | Aspects of the interface were researched and designed. Final push for integrated working system was not completed. | **50** % |
| Experimental data for travel-time tomography | Experimental data was collected successfully with a cost-effective setup. | **100** % |
| Evaluate hardware implementation aspects | Processing with lesser bit depth resulted in more inaccurate recovery result, but recovery with the integrating method's travel-time values proved to be surprisingly good, also in terms of hardware cost. | **150** % |

Mathematically the acoustic travel-time tomography produced better results than expected. The integrating method of calculating travel time produced a surprising result. The method kept stability when the travel-time calculation results were obtained with more inaccurate calculation via integer arithmetics and more inaccurate acoustic data. This was predicted before the project started, but the actual results were much better than expected. It is likely that these results will be further analyzed and tested in the future within the inverse problems research group at the Deparment of Mathematics, TUT. This testing could involve ultrasonic tomography.

Experimental data collection for acoustic tomography was accomplished with an inexpensive test setup consisting of yoga mat and pillows, commercial microphones and a commercial speaker. Inaccuracies of setup were documented and their relation to asteroid tomography scenario was analyzed. Using the test setup, one complete data set including three transmitter positions were recorded and the data was used in this work.

Travel-time calculation modules were converted from Matlab scripts to synthesizable algorithmic C code manually. Catapult C high-level synthesis tool was then used to evaluate and further develop the different modules. Traditional testbenching methods integrated into Catapult workflow were used for testing and verification. Synthesis workflow options in terms of interfacing with Matlab were briefly studied. Different workflow options and possibilities left a lot to study further, especially in terms of Matlab verification and integration. Algorithmic C with Catapult was studied and some interesting issues, such as loop behaviors and pitfalls in division implementation were documented. Stability of the version of the Catapult software used in this work caused minor issues in some situations. The direct implementation workflow left a lot of possibilities regarding high-level synthesis to study further, and the resulting designs in this project are not optimal.

Interface from Matlab running on PC to a preprocessor module on an FPGA demonstration board was studied. This goal was not completely reached due to differences in various FPGA development boards and project time constraints. Some aspects, such as the protocol and the Matlab side of the system were designed. This proved to be a good decision. While the implemented travel-time calculation modules would have fitted on the FGPA chip, the selected Altera DE2 development board is old and the Cyclone II FGPGA shouldn't be used on any commercial products.

Project ended with many development paths in terms of mathematics and hardware development. Better experimental data could be collected with automatic ultrasonic data collection setup and the mathematics could be developed to work with different types of data preprocessing, exceeding the limits of travel-time tomography. Development paths, if undertaken, might eventually result in a commercial tomography product. Asteroid mining, which was used as a motivator in this thesis project, might benefit from the mathematical results and perhaps also from more productive FPGA hardware development involving high-level synthesis.

# BIBLIOGRAPHY

[1] V. Agrawal, *Satellite Technology: Principles and Applications*. Wiley, 2014, 846 p.

[2] J. Ainali, “Philips MRI in Sahlgrenska Universitetsjukhuset, Gothenburg, Sweden,” 2008, used under Creative Commons Attribution 3.0 Unported license. Available: https://en.wikipedia.org/wiki/Magnetic_resonance_imaging#/media/File:MRI-Philips.JPG. Accessed 9.2.2016.

[3] N. Alachiotis, “UDP/IP Core,” 2010, Available: http://opencores.org/project,udp_ip__core. Accessed 11.12.2015.

[4] Altera Inc., “Cyclone II Device Handbook, Volume 1,” 2008, 470 p. Available: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/cyc2/cyc2_cii5v1.pdf. Accessed 1.5.2016.

[5] ——, “Quartus II Software and Device Support Release Notes Version 13.1,” 2013, 20 p. Available: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/rn/archives/rn_qts_131_dev_support.pdf. Accessed 13.5.2016.

[6] ——, “Introduction to the Quartus® II Software Version 10.0,” 2014, 136 p. Available: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/manual/intro_to_quartus2.pdf. Accessed 13.5.2016.

[7] ——, “Interface protocols,” 2015, Available: https://www.altera.com/products/intellectual-property/all-ip/interface-protocols.html. Accessed 15.12.2015.

[8] Audacity Project, “Audacity® project website,” 2016, Available: http://audacityteam.org. Accessed 1.2.2016.

[9] Avisoft Bioacoustics e.K., “Avisoft Bioacoustics website,” 2016, Available: http://www.avisoft.com/. Accessed 10.2.2016.

[10] J. Axelson, *Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems*, 2nd ed., ser. Complete Guides Series. Lakeview Research, 2007, 380 p.

[11] Y. Barbin, W. Kofman, E. Nielsen, T. Hagfors, R. Seu, G. Picardi, and H. Svedhem, “A two dimensional simulation of the CONSERT experiment (radio tomography of comet Wirtanen),” *Advances in Space Research*, vol. 24, pp. 1127–1138, Jan. 1999.

[12] W. A. Berg, A. I. Bandos, E. B. Mendelson, D. Lehrer, R. A. Jong, and E. D. Pisano, “Ultrasound as the primary screening test for breast cancer: Analysis from ACRIN 6666,” *Journal of the National Cancer Institute*, vol. 108, no. 4, 2016.

[13] D. M. Boroson, B. S. Robinson, D. V. Murphy, D. A. Burianek, F. Khatri, J. M. Kovalik, Z. Sodnik, and D. M. Cornwell, “Overview and results of the lunar laser communication demonstration,” *Proceedings of SPIE*, vol. 8971, pp. 89 710S–89 710S–11, 2014.

[14] W. Bottke, *Asteroids III*, ser. University of Arizona space science series. University of Arizona Press, 2002, Tucson, Arizona. 785 pp.

[15] D. Braess, *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*, 3rd ed. Cambridge University Press, 2007, Cambridge, UK. 384 p.

[16] J. Cameron, *Physical Properties of Tissue. A Comprehensive Reference Book*, F. A. Duck, Ed. Academic Press, UK, 1991, 336 p.

[17] A. Caponiti, “Space and defense power systems program information briefing,” Office of Space and Defense Power Systems (NE-75), Office of Nuclear Energy, 2015, Available: http://www.lpi.usra.edu/opag/feb2015/presentations/15_Caponiti%20OPAG%20charts%202-20-2015.pdf. Accessed 10.7.2015.

[18] V. Chan and A. Perlas, *Atlas of Ultrasound-Guided Procedures in Interventional Pain Management*. New York, NY: Springer New York, 2011, ch. Basics of Ultrasound Imaging, pp. 13–19.

[19] A. Chin, R. Coelho, R. Nugent, R. Munakata, and J. Puig-Suari, “The CubeSat: The Picosatellite Standard for Research and Education,” 2008, AIAA SPACE 2008 Conference & Exposition, 9-11 September 2008, San Diego, California.

[20] H. Choi and J. S. Popovics, “NDE Application of ultrasonic tomography to a full-scale concrete structure,” *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 62, no. 6, pp. 1076–1085, 6 2015.

[21] C. Crow, “The main asteroid belt,” 2009, Available: http://dawn.jpl.nasa.gov/science/pdfs/Asterbelt.pdf. Accessed 15.10.2015.

[22] Deep Space Industries, “Deep Space Industries website,” 2015, Available: http://deepspaceindustries.com. Accessed 15.6.2015.

[23] ——, “Mothership - Affordable missions beyond low Earth orbit,” 2015, Available: http://deepspaceindustries.com/wp-content/uploads/2015/08/DSI-Mothership.pdf. Accessed 15.6.2015.

[24] M. Elvis, “Prospecting asteroid resources,” in *Asteroids: Prospective Energy and Material Resources*, V. Badescu, Ed. Springer-Verlag GmbH, 2013, pp. 81-129.

[25] European Space Agency, “Pioneering Philae completes main mission before hibernation,” 2014, Available: http://www.esa.int/Our_Activities/Space_Science/Rosetta/Pioneering_Philae_completes_main_mission_before_hibernation. Accessed 5.1.2016.

[26] ——, “Rosetta’s frequently asked questions,” 2015, Available: http://www.esa.int/Our_Activities/Space_Science/Rosetta/Frequently_asked_questions. Accessed 15.2.2016.

[27] L. Evans, *Partial Differential Equations*, ser. Graduate studies in mathematics. American Mathematical Society, Providence, Rhode Island, 1998, 749 pp.

[28] A. Fichtner, *Full Seismic Waveform Modelling and Inversion.* Springer-Verlag Berlin Heidelberg, Berlin, 2011, 343 p.

[29] M. Fingeroff, *High-Level Synthesis Blue Book.* Mentor Graphics Corporation, 2010, 272 p.

[30] P. Fortescue, J. Stark, and G. Swinerd, *Spacecraft Systems Engineering*, 3rd ed. Wiley & Sons, Chichester, England, 2003, 678 p.

[31] German Aerospace Center (DLR) Institute of Planetary Research, Asteroids and Comets, “Rosetta at a glance - technical data and timeline,” 2016, Available: http://www.dlr.de/dlr/en/desktopdefault.aspx/tabid-10728/584_read-386/. Accessed 15.4.2016.

[32] J. Guo and C. Luo, “Application of tunnel seismic image approach to the advanced geological prediction for tunnel,” *Journal of Multimedia*, vol. 9, no. 7, 2014, pp. 879-886.

[33] K. S. Hall, “Air-coupled ultrasonic tomographic imaging of concrete elements,” Ph.D. dissertation, University of Illinois, 2011, 106 p.

[34] A. Herique, W. W. Kofman, A. Barucci, P. Beck, J. Biele, S. M. Clifford, J. Goutail, E. Heggy, T. Ho, A. Kumamoto, J. Lasue, A. Levasseur-Regourd, P. Michel, E. Nielsen, T. Ono, P. Pujet, D. Plettemeier, S. Ulamec, and S. Zine, "ASSERT for Mascot / Hayabusa 2 mission: A radar tomography of 1999 JU3," *AGU Fall Meeting Abstracts*, Dec. 2010.

[35] S. S. Hubbard, J. J. E. Peterson, E. L. Majer, P. T. Zawislanski, K. H. Williams, J. Roberts, and F. Wobber, "Estimation of permeable pathways and water content using tomographic radar data," in *The Leading Edge*. Springer-Verlag GmbH, 1997, 5 pp.

[36] IEEE Computer Society, "IEEE 802®: Overview And Architecture," 2014, 74 p. Available: http://standards.ieee.org/getieee802/download/802-2014.pdf. Accessed 6.2.2016.

[37] IVI Foundation, "IVI Specifications," 2011, Available :http://www.ivifoundation.org/specifications/default.aspx. Accessed 14.4.2016.

[38] W. B. Johnson, "Design and testing of a laboratory ultrasonic data acquisition system for tomography," Master's thesis, Virginia Polytechnic Institute and State University, 2004, 108 p. Available: http://scholar.lib.vt.edu/theses/available/etd-01142005-133202/unrestricted/wjohnson_thesis.pdf. Accessed 12.7.2015.

[39] P. Kabal, "Audio file format specifications," 2015, Available :http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/WAVE.html. Accessed 3.2.2016.

[40] J. P. Kaipio and E. Somersalo, *Statistical and Computational Methods for Inverse Problems*. Springer-Verlag GmbH, New York, 2004, 340 p.

[41] Kanwarjot Singh, "Speed of sound in some common solids," 2011, available: http://www.engineeringtoolbox.com/sound-speed-solids-d_713.html. Accessed 15.2.2016.

[42] W. Kofman, Y. Barbin, J. Klinger, A.-C. Levasseur-Regourd, J.-P. Barriot, A. Herique, T. Hagfors, E. Nielsen, E. Grün, P. Edenhofer, H. Kochan, G. Picardi, R. Seu, J. van Zyl, C. Elachi, J. Melosh, J. Veverka, P. Weissman, L. Svedhem, S. Hamran, and I. Williams, "Comet nucleus sounding experiment by radiowave transmission," *Advances in Space Research*, vol. 21, no. 11, pp. 1589 – 1598, 1998.

[43] W. Kofman, A. Herique, Y. Barbin, J.-P. Barriot, V. C. S. Clifford, P. Edenhofer, C. Elachi, C. E. J.-P. Goutail, E. Heggy, L. Jorda, J. Lasue, A.-C. Levasseur-Regourd, E. Nielsen, P. Pasquero, F. P. P. Puget, D. Plettemeier, Y. Rogez, H. Sierks, C. S. H. Svedhem, I. Williams, S. Zine, and J. V. Zyln, “Supplementary Materials for Properties of the 67P/Churyumov-Gerasimenko interior revealed by CONSERT radar,” *Science Magazine*, vol. 349, no. 6247, 2015, 22 p. Available: http://www.sciencemag.org/content/349/6247/aab0639/suppl/DC1. Accessed 9.11.2015.

[44] H. J. Kramer, “Hayabusa-2, japan’s second asteroid sample return mission,” 2016, Available: https://directory.eoportal.org/web/eoportal/satellite-missions/h/hayabusa-2. Accessed 9.2.2016.

[45] E. Kulu, “Nanosatellite Database by Erik,” 2014, Available: http://www.nanosats.eu/. Accessed 9.2.2016.

[46] T. J. Lee, J. H. Suh, H. J. Kim, Y. Song, and K. H. Lee, “Electromagnetic traveltime tomography using an approximate wavefield transform,” *Society of Exploration Geophysicists Geophysics*, vol. 67, no. 1, pp. 68–76, 2002.

[47] T. J. Lee and T. Uchida, “Electromagnetic traveltime tomography: Application for reservoir characterization in the Lost Hills oil field, California,” *Society of Exploration Geophysicists Geophysics*, vol. 70, no. 3, pp. 51–58, 2005, Available: http://library.seg.org/doi/abs/10.1190/1.1925743?journalCode=gpysa7. Accessed 1.8.2015.

[48] V. Madisetti, M. Sinnokrot, D. Lee, and A. Gerstlauer, “Introduction to Catapult C,” 2010, 19 p. Available: http://users.ece.utexas.edu/~gerstl/ee382v_s11/soc/catapult/Catapult_Tutorial.pdf. Accessed 11.10.2015.

[49] MathWorks, Inc., “C Code Generation from MATLAB,” 2016, Available: http://se.mathworks.com/help/comm/ug/code-generation-from-matlab.html. Accessed 15.2.2016.

[50] ——, “Getting Started with MATLAB to HDL Workflow,” 2016, Available: http://se.mathworks.com/help/hdlcoder/examples/getting-started-with-matlab-to-hdl-workflow.html. Accessed 20.3.2016.

[51] ——, “MAT-File Versions,” 2016, Available: http://se.mathworks.com/help/matlab/import_export/mat-file-versions.html. Accessed 5.2.2016.

[52] ——, “Matlab HDL Coder Overview,” 2016, Available: http://se.mathworks.com/products/hdl-coder/. Accessed 20.3.2016.

[53] ——, “Matlab Instrument Control Toolbox: Key Features,” 2016, Available: http://se.mathworks.com/products/instrument/features.html. Accessed 15.4.2016.

[54] ——, “Matlab Instrument Control Toolbox: UDP documentation,” 2016, Available: http://se.mathworks.com/help/instrument/udp.html. Accessed 5.2.2016.

[55] ——, “Numeric types,” 2016, Available: http://se.mathworks.com/help/matlab/numeric-types.html. Accessed 5.2.2016.

[56] Mentor Graphics Inc., “Mentor Graphics Acquires Calypto Design Systems,” 2015, Available: https://www.mentor.com/company/news/mentor-acquires-calypto-design-systems. Accessed 20.9.2015.

[57] Mentor Graphics, Inc., “High-Level Synthesis and RTL Low-Power,” 2016, Available: https://www.mentor.com/hls-lp/catapult-high-level-synthesis/. Accessed 12.3.2016.

[58] National Aeronautisc and Space Administration and Jet Propulsion Laboratory, “Near Earth Orbit Program: Neo Groups,” 2015, Available: http://neo.jpl.nasa.gov/neo/groups.html. Accessed 13.10.2015.

[59] ——, “About OSIRIS-REx,” 2016, Available: https://www.nasa.gov/mission_pages/osiris-rex/index.html. Accessed 17.3.2016.

[60] ——, “Dawn mission website,” 2016, Available: http://dawn.jpl.nasa.gov/. Accessed 17.3.2016.

[61] J. Norberg, L. Roininen, J. Vierinen, O. Amm, D. McKay-Bukowski, and M. Lehtinen, “Ionospheric tomography in bayesian framework with gaussian markov random field priors,” *Radio Science*, vol. 50, no. 2, pp. 138–152, 2015.

[62] Planetary Resources, “Planetary Resources website,” 2015, Available: http://www.planetaryresources.com. Accessed 10.6.2015.

[63] D. C. Plummer, “RFC 826: An Ethernet Address Resolution Protocol,” 1982, 10 p. Available: https://tools.ietf.org/pdf/rfc826.pdf. Accessed 1.4.2016.

[64] J. Postel, “RFC 768: User Datagram Protocol,” 1980, 3 p. Available: https://tools.ietf.org/pdf/rfc768.pdf. Accessed 1.4.2016.

[65] Proceq SA, "Ultrasonic pulse velocity and pulse echo testing of concrete," 2015, Available: http://www.proceq.com/nondestructivetestequipment/concrete-testing/ultrasonic-pulse-velocity.html. Accessed 12.8.2015.

[66] S. Pursiainen and M. Kaasalainen, "Iterative alternating sequential (IAS) method for radio tomography of asteroids in 3D," *Planetary and Space Science*, vol. 82, pp. 84–98, 2013.

[67] ——, "Detection of anomalies in radio tomography of asteroids: Source count and forward errors," *Planetary and Space Science*, vol. 99, pp. 36 – 47, 2014.

[68] ——, "Electromagnetic 3D subsurface imaging with source sparsity for a synthetic object," *Inverse Problems*, vol. 31, no. 12, p. 17, 2015.

[69] S. Pursiainen, M. Bauer, J. Vorwerk, H. Köstler, and C. Wolters, "Comparison Study for Whitney (Raviart-Thomas) Type Source Models in Finite Element Method Based EEG Forward Modeling," *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 11, pp. 2648–2656, 2015.

[70] D. Rickey, "Medical ultrasound scanner," 2006, used under Creative Commons Attribution-ShareAlike 2.5 Generic license. Available: https://en.wikipedia.org/wiki/Medical_ultrasound#/media/File:AlokaPhoto2006a.jpg. Accessed 10.2.2016.

[71] J. B. Schneider, *Understanding the FDTD Method.* John B. Schneider, 2012, 403 p. Available: http://www.eecs.wsu.edu/~schneidj/ufdtd/. Accessed 5.6.2015.

[72] A. Shastri, G. Stitt, and E. Riccio, "A scheduling and binding heuristic for high-level synthesis of fault-tolerant FPGA applications," in *2015 IEEE 26th International Conference on Application-specific Systems, Architectures and Processors*. IEEE Computer Society, 2015, pp. 202–209.

[73] M. Spanier and A. Gorenstein, "Ethernet Communication Interface for the FPGA," 2011, Student project work. Available: http://people.ece.cornell.edu/land/courses/ece5760/FinalProjects/f2011/mis47_ayg6/mis47_ayg6/. Accessed 5.2.2016.

[74] J. A. Sparano, "TNM Classification for Breast Cancer," 2015, Available: http://emedicine.medscape.com/article/2007112-overview. Accessed 28.4.2016.

[75] P. Suetens, *Fundamentals of Medical Imaging*, 2nd ed. Cambridge University Press, Cambridge, UK, 2009, 253 p.

[76] Terasic Inc., “DE2 Development and Education Board User Manual,” 2006, 72 p. Available: ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf. Accessed 20.4.2016.

[77] ——, “Altera DE2 Board,” 2013, Available: http://www.terasic.com.tw/cgi-bin/page/archive.pl?No=30. Accessed 2.4.2016.

[78] University of Arizona, “Osiris-REx asteroid sample return mission instrumentation,” 2016, Available: http://www.asteroidmission.org/instrumentation/. Accessed 20.3.2016.

[79] USB Implementers Forum, Inc., “Universal serial bus revision 2.0 specification,” 2000, 622 p. Available: http://www.usb.org/developers/docs/usb20_docs/usb_20_032516.zip. Accessed 11.4.2016.

[80] M. M. J. L. van de Kamp, “Medium-scale 4-D ionospheric tomography using a dense GPS network,” *Annales Geophysicae*, vol. 31, no. 1, pp. 75–89, 2013.

[81] J. Vierinen, J. Norberg, M. S. Lehtinen, O. Amm, L. Roininen, A. Väänänen, P. J. Erickson, and D. McKay-Bukowski, “Beacon satellite receiver for ionospheric tomography,” *Radio Science*, vol. 49, no. 12, pp. 1141–1152, 2014.

[82] G. Wang, “Catapult C Synthesis Work Flow Tutorial,” 2010, 20 p. Available: http://www.ece.rice.edu/~gw2/pdf/Technical_notes_Catapult_work_flow_tutorial_for_ELEC522.pdf. Accessed 11.10.2015.

[83] WIZnet Co., Ltd, “W5500 Overview,” 2016, Available :http://www.wiznet.co.kr/product-item/w5500/. Accessed 19.4.2016.

[84] K. S. Yee, “Numerical solution of initial boundary value problems involving maxwells equations in isotropic media,” *IEEE Trans. Antennas and Propagation*, pp. 302–307, 1966.

[85] D. Zhao, *Multiscale Seismic Tomography*. Springer Geophysics, Japan, 2015, 304 p. Available: http://link.springer.com/book/10.1007%2F978-4-431-55360-1. Accessed 1.3.2016.

[86] X.-f. Zhong, Y.-x. Wu, D.-h. Li, Q. Li, and X.-g. Wang, “Ultrasonic tomography and its applications in oilfield,” *Journal of Zhejiang University Science A*, vol. 6, pp. 1420–1423, December 2005, Available: http://link.springer.com/article/10.1007/BF02888926. Accessed 1.3.2016.