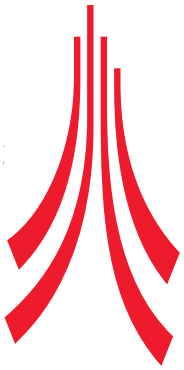


Real-Time Tracking for Airborne Broadband Ultrasound

Mohammed Alloulah

June 2011

School of Computing and Communications

Lancaster University

Submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

To mama,
May you rest in peace – “Allah yerhamek.”

Abstract

Context refers to a collection of elements with which people associate situations. The location and state of objects in an environment constitute an important subset of context, which has been thoroughly researched and established. By tracking the movement of people and objects within an environment, context-aware applications may be realized, enabling a host of interaction schemes that are intuitive, utilitarian, and fun. Many sensing technologies have been shown to supply tracking services to context-aware systems. These sensing technologies are complementary, and none possesses all desirable tracking attributes for all situations. The preference for the use of a particular tracking technology is often much dependent on the application at hand. Ad hoc, mobile applications are particularly hard to satisfy, given their dynamic nature and the centralized, infrastructure-reliant arrangement of most of the accurate tracking systems available.

The first part of this dissertation describes methods for embedded, real-time airborne broadband ultrasonic tracking. The tracker has been built around the assumption of a mobile operation that is deployed ad hoc and upon demand. In order for this to happen, the embedded, real-time operation of sensor nodes has been emphasized. The efficient signalling designs that make way for multiuser, ad hoc tracking deployment have been thoroughly characterized, and shown to perform close to their infrastructure-reliant counterparts. System-level parameterization of tracking is also possible subject to application needs.

The remainder of this work shows for the first time in literature that real-time Doppler processing in the airborne broadband ultrasonic modality is possible, whereby velocity inference of mobile nodes is facilitated. Building on advancements from underwater acoustics research, a complex Doppler receiver has been derived and characterized. Its implications on real-time realizations have been studied and characterized utilizing a high-level synthesis architectural exploration methodology. This has revealed that it is feasible to implement real-time Doppler tracking for airborne broadband ultrasound using modern reconfigurable fabrics (i.e. FPGAs). The dissertation concludes by examining the applicability of findings on even more complex forms of processing such as multiuser direction-of-arrival estimation by means of beamforming, and binary Doppler-tolerant reception.

Preface

This dissertation has not been submitted in support of an application for another degree at this or any other university. It is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated.

Mohammed Alloulah

Acknowledgment

I would like to thank the members of the Embedded Interactive Systems (EIS) Group; past and present. The wide range of research being conducted at EIS is inspiring, and serves to ground one's research in reality.

My sincerest gratitude goes to my boss Dr. Mike Hazas. His uncompromising commitment to pursuing brand new ideas has been the unspoken force behind this research, making me literally look underwater and synthesize at a high level! Had it not been for his patience, this PhD would have simply not been possible. His enthusiasm and support on research-y and administrative issues are greatly appreciated.

I am most grateful to my examiners Dr. Dinesh Pamunuwa and Prof. Milica Stojanovic for the stimulating and informative discussion.

I would also like to thank Prof. Hans Gellersen for graciously funding the initial part of this research.

Carl Fischer deserves special thanks for his help in setting up the Linux VM for high-level synthesis and server administration.

Throughout the duration of my research, coffees with Dr. Andrew Scott have been most instrumental at maintaining my well-being. Andrew was all ears whenever I felt the need to ramble about all sorts of issues; technical, political, and otherwise.

Although they came in late in the game, late-night "sessions" with Dr. Fahim Kawsar have opened my eyes on the logistics of research. In another life, I would most certainly heed his mentoring advice.

Matthew Oppenheim converted my schematics into shiny green PCBs. One day, I will try to learn the arcane art. Until then, Matt continues to be the object of both my admiration and my envy.

At some point, Bashar Altakrouri came from my part of the world and landed in my office. His authentic Middle Eastern recipes are not to go by uncommended.

Members of the Sensors and Devices (SenDev) at Microsoft Research Cambridge are cordially acknowledged for taking time off their busy schedule to review this work.

Specifically, I would like to thank Steve Hodges, James Scott, Shahram Izadi, and the then-visiting professor Joe Paradiso for their input on this research.

Chris Rottner of the then-AutoESL coordinated an academic license deal for AutoPilot. Also Dirk Seynhaeve patiently provided orientation with miscellaneous issues surrounding high-level synthesis using AutoPilot.

Finally, I would like to express my deepest thanks to my family and Rachel Burrows for their continued support throughout.

This research has been supported by a Microsoft Research PhD Scholarship, an EPSRC grant “ANTHORN”, and an EC FP6 grant “RELATE”. The Xilinx university program is also appreciated. The faculty of Science and Technology at Lancaster University has provided additional funds for travel and equipment.

The complete absolutely free man, definitively and completely satisfied by what he is, the man who is perfected and completed in and by this satisfaction, will be the Slave who has "overcome" his Slavery. If idle Mastery is an impasse, laborious Slavery, in contrast, is the source of all human, social, historical progress. History is the history of the working Slave.

—*Alexandre Kojève*, Introduction to the Reading of Hegel

It was the slave's continuing desire for recognition that was the motor which propelled history forward, not the idle complacency and unchanging self-identity of the master.

—*Francis Fukuyama*, The End of History and the Last Man

And say thou: work on! Allah beholdeth your work and so do His apostle and the believers, and anon ye will be brought back to the Knower of the hidden and the manifest; He will then declare unto you that which ye have been working. (Surah Al-Tawba 9:105)

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Recent years have seen a sharp rise in interactive, user-centric applications of a distinctive feature; that pertaining to space inhabited by people. Numerous examples can be found, from gaming consoles to mobile phones implementing gesture recognition functions. Fueled by increasing processing power and decreasing size and cost of hardware, next generation mobile and wearable computing will be about the coupling of people's activities and information through spaces [134]. This is to be achieved through the continuous sensing of the movement of people and their devices through everyday living physical spaces.

Information gathered from physical spaces is collectively referred to as *context*. *Context-aware* computing is about discerning those attributes of the physical environment most intuitive to human cognition, such as light conditions and temperature. Of these physical attributes, the position, bearing, and movement of people and objects have received much attention in both academia and industry, as they make possible location-based services. Such physical sensing is designated in this dissertation as *tracking*.

The need for real-time tracking in computing systems spans areas from leisure (such as gaming and social networking) to business applications (such as asset tracking or optimizing the workflows of organizations). A commonality between these areas is the dynamic nature of interaction. This interactivity puts pressure on technologists to attempt to approximate the performance of high-end tracking systems operating in a predefined manner (such as motion capture for medical instrumentations) while catering for needs anew posed by future dynamic environments.

1.1 Research Motivation

Animals such as bats and dolphins rely on ultrasonic sensing and communication in order to cope with their physical environments; especially, in scenarios involving spatial interaction and navigation. The same inaudible acoustics have been widely used for the provision of tracking services in a variety of human scenarios. Despite the associated installation overhead, ultrasonic trackers have previously demonstrated responsiveness and accuracy suitable for realizing complex interactive scenarios.

Existing ultrasonic tracking systems require extensive setup and careful management schemes [118]. For this reason, centralized control has been at the heart of real-world implementations [88] in order to successfully execute functionality, mitigate against collisions, simplify processing requirements, and enhance the quality of service.

However, with the emergence of parallel computational devices, a new spatial computing paradigm is now afforded, wherein the execution of complex functionality occurs in space by a dedicated logic circuit rather than in time by a sequential von Neumann model [34]. This new trend will prompt further sustained boost in hardware capabilities. It has been and continues to be a major enabler behind increased sophistication across all application domains; high-end and low-end alike.

A seminal prior work has demonstrated accurate and robust indoor localization using a wide bandwidth (50 kHz) of in-air ultrasound [61]. This broadband ultrasonic wireless channel could potentially afford new capabilities. Yet little was done to capitalize on this bandwidth because of the associated processing requirements and the non-trivial Doppler distortions encountered.

1.2 Research Statement

In a nutshell, limitations of previous approaches can be traced to a tight centralism and the desire to keep processing workloads minimal at the periphery of the system. The goal of this doctoral research is to take the converse approach, whereby considerable processing is performed by peripheral devices (i.e. tags) in a decentralized fashion. As such, improved utilization (both in the wireless channel's bandwidth and for computations) will be facilitated in order to explore new processing methods for the novel modality of airborne broadband ultrasound. Equally, the implications of such approach on hardware processing requirements need to be assessed thoroughly. It is posited that such a system would allow vast improvements in ad hoc, mobile tracking.

1.3 Dissertation Outline

Chapter 2 establishes the environment within which this research is carried out. It surveys existing tracking technologies, and concludes by highlighting the enhancements brought about by what airborne broadband ultrasonic (ABU) tracking has to offer.

The dissertation is then composed of three parts, each of which represents a major contribution of this PhD research.

Part I addresses embedded, real-time transceiver designs for the ABU modality. Chapter 3 gives a brief introduction to reconfigurable fabrics. Chapters 4 and 5 describe design, implementation, and evaluation of the essential transceiver components required by an ad hoc ABU tracking system.

Part II embarks on a novel development of phase-coherent ABU. Chapter 6 first derives a DS CDMA adaptive, Doppler-tolerant tracker, inspired by decades of advancements in underwater acoustic (UWA) communications research. The tracker is then analyzed and an empirical evaluation of performance is reported.

Part III then considers the feasibility of a comprehensive real-time, embedded adaptive ABU tracker. Chapter 7 introduces the high-level synthesis tools needed to implement such complex algorithms. Chapter 8 examines the implications of the algorithmic findings in chapter 6 on the real-time architecture. It compares various tracking configurations from an implementation standpoint and gauges the suitability of the proposed Doppler-tolerant tracker.

Finally chapter 9 concludes this work, and identifies important areas to expand on this research.

CHAPTER 2

Indoor Tracking

This chapter discusses the the state-of-the-art in indoor tracking. It then demonstrates that the novel airborne broadband ultrasonic modality is a viable candidate for the provision of indoor tracking; especially, targeting ad hoc, mobile applications.

Section 2.2 reviews research domains requiring low-rate tracking. Section 2.3 surveys existing technologies for high-rate tracking. Section 2.4 exposes limitations of current enabling technologies and identifies a set of desirable properties for indoor trackers. Section 2.5 justifies airborne broadband ultrasound as an important addition to existing trackers, and proposes avenues for enhancing the state-of-the-art.

2.1 Introduction

The tracking of people and objects in indoor environments is a problem of great significance to a number of domains. Tracking encompasses initial positioning (or localization) followed by continual monitoring of movement thereafter. Tracking can be classified as either low-rate or high-rate. In low-rate tracking, the emphasis is more on monitoring the location of people and objects with the possibility of deriving the rate of change of position; speed, albeit with decreased fidelity of measurements. Methods for high-rate tracking, on the other hand, achieve improved estimation fidelity facilitating the inference of motion pattern with much greater accuracy.

The following discussion will first focus on user-centric applications for the emerging “smart” environments as examples of low-rate tracking which are being increasingly sought after. The domains of Ubiquitous Computing, Mobile Computing, location-based services, and sensor networks will be examined in order to establish the connection to what can be collectively dubbed as smart environments. Later, high-rate tracking will be

visited in the context of more critical, yet narrower in scope applications such as motion capture for medical instrumentation. Various contending technologies in this narrower application space will be surveyed.

2.2 Low-rate Tracking – Relevance

Low-rate tracking is relevant to a number of areas. The following will define and discuss how these areas rely on tracking information.

2.2.1 Ubiquitous/Pervasive Computing

Essentially, the ubiquitous vision [144, 143] is tightly coupled with the physical world it wishes to transform. The physical environment is the active medium through which human perception/actuation natively operates. Ubiquitous systems seek to render the space of the intricate real-life services *self-configurable*; effectively, providing users with transparent functions that facilitate the day-to-day tasks without explicit intervention. Thus ubiquitous systems bear an intimate relationship with what is coined as context-aware computing [116] or sentient computing [68].

Context-aware computing emulates human perception by reasoning about the surrounding environment in lexicon intelligible to the human mind. This lexicon includes all aspects of innate human cognition such as temperature, ambient light, proximity to objects, etc. Systems in context-aware computing detect events in the environment arising from changing conditions, cast these events in terms of human intelligible lexicon, and react accordingly. An integral subset of context is those physical properties which enable the estimation of the spatial conditions of people and objects alike, be they displacement, orientation, or mobility.

2.2.2 Location-awareness for Ubiquitous and Mobile Computing[1]

Location-aware applications are seeing increased usage, particularly in outdoor and urban environments, where existing wireless communication infrastructures can be exploited to derive a user's *coarse* position [38, 112]. These systems typically have accuracies in the tens of meters, satisfying the requirements of many applications (such as navigation or location-based yellow page searches) in these relatively large-scale environments.

Indoor applications bring a different set of challenges as they tend to require *fine-grained* location information. This is particularly true for certain classes of user-centric applications wherein the demand is on real-time, reliable, accurate, and responsive measurements with possibly orientation information. Examples include "follow-me" computing [18]; augmented navigation [95]; environmental service discovery [98], impromptu

[1] This subsection features some material coauthored with Dr. Mike Hazas from:
"Embedded Broadband Ultrasonic Sensing for Robust and Scalable Positioning," EPSRC grant.

communication and data sharing between co-located users; and creating and viewing digital annotations of physical objects in the environment [49, 114].

Pre-existing infrastructures are difficult to rely upon indoors while meeting the localization requirements of the majority of location-aware applications. For instance, the classic Global Positioning System (GPS) has limited usage indoors [112]. Additionally, location sensing built on top of communication systems such as Wi-Fi and Global System for Mobile communications (GSM) is problematic; especially, in medium to deep indoors, electrically noisy indoor scenarios, subterranean places (e.g. parking), and others [27]. Simplistically speaking, these systems haven't been designed and optimized to provide such services. Moreover, the irregularities (multipath and fading) of the indoor radio wireless channel degrade their accuracy yet further. Nonetheless, coarse-grained positioning can be readily achieved using techniques such as WiFi fingerprinting [26].

Infrastructure-reliant systems have traditionally supplied such "quality" location information. For high-end applications e.g. flight simulation, optimum performance is usually strived for. Various technologies are utilized—sometimes in a hybrid manner—in order to heavily instrument a confined operational space [145]. For more scalable scenarios, utilizing an abundance of localizing nodes coordinated by a centralized service, technologies such as ultrasound and ultra-wideband (UWB) radio are capable of meeting the stringent requirements of all applications [7, 54, 14, 51]. However, this comes at a high price tag, installation, and calibration costs ruling out their cost-effective deployment for all but the most specialized groups of users e.g. researchers, motion capture for television studios, etc.

The calibration argument

To elaborate further on the shortcomings of infrastructure-reliant systems, [118] demonstrates the extensive calibration and surveying often needed by such systems. These procedures can be quite laborious even when taking advantage of more precise, faster custom devices instead of hand measurements.[2] The caveat, however, is that these devices are expensive, difficult to operate as they require certain level of expertise from the user. The surveyed volume consisted of two small rooms (each with about twelve fixed units) and one large room (with twenty-six fixed units). All units were surveyed using both a crate and a theodolite. Since it is a proprietary device developed at AT&T Labs Cambridge, we will stick to the official definition of the crate presented by [118] as "a purpose-built device comprising a rigid frame with three spring-loaded reels of cable mounted on it; each reel can electronically report the length of its extended cable." By touching the end of each cable to each survey point, trilateration software can determine the points' locations. A theodolite is the usual instrument used by surveyors to measure the horizontal and vertical angles between itself and the object under observation by means of a reflector. The accuracy of these two instruments were compared against hand measurements using a tape measure taken between forty fixed unit pairs. Table 2.1

[2] According to [118], a typical room with fifteen fixed units would take approximately forty-five minutes to survey.

shows comparison of the absolute error of Euclidean 3D distances comparison for each device. Also the time taken by the two researchers to survey the large and small rooms is provided.

Table 2.1: Comparison of device-assisted conventional surveying techniques[a]

	Absolute error (mm)		Surveying time (min)	
	Mean	Std. deviation	Small room	Large room
Crate	3.9	3.5	40	50
Theodolite	3.7	2.9	47	65

[a] Note that these results, though not included in the final publication manuscript in [118], were compiled by Dr. Hazas in preparation.

2.2.3 Location-based services

Expansively speaking, location-based services (LBS) fall under a broader concept: geographic information systems (GIS) which is concerned with referencing and tying information systems in whole to Earth—It makes sense for people to classify things spatially since space constitutes existence.[3] LBS, in turn, are key enabling components of the emerging m-businesses. M-businesses are analogous to e-businesses with the specialty of being conducted in a *mobile* fashion.

It is customary that the concept of quality of service (QoS) is introduced whenever a system has a commercial scope. In fact, practitioners wasted no time in analyzing the implications of location provision as an imminent driving force across the space of m-businesses. The following will survey up-to-date papers which focus on the requirements of location readings set by LBS.

When assessing the business opportunities in LBS the authors of [112] state that in niche consumer applications "services could be extremely rich and narrow in their focus." They also predict that in industrial and corporate applications "extensive range of enabling technologies will make this happen in conjunction with existing network infrastructure." This is concurred in [105] with "precise LBS will probably rely on a combination of technologies." On the competitiveness in LBS, [89] observes that "They concluded that superior customer experiences, distinctive, secure, high-quality service and branding will be vital for LBS providers to gain an advantage." With reference to QoS, [73] articulates that "empirical evidence that the quality of LBS content is an issue to be explicitly addressed in both practice and research."

Now that the need for quality location provision is established, quantitative metrics for gauging this quality have to be devised.

The infrastructure for LBS has three constituent elements associated with: communication, positioning, and mapping of application [75] as illustrated in figure 2.1. The most

[3] This statement is only valid under the materialistic school of thought.

comprehensive analysis of LBS QoS has to take into consideration all three layers upon which the service is realized. However, for the sake of simplicity, we will only comment on contributions from the positional component and, where necessary, mention others to facilitate the discussion.

Figure 2.1: LBS Infrastructure

Among other variants of QoS figures of merit, the most frequently and unanimously referred to metrics are accuracy and response time [73, 33, 27, 104]. Accuracy is the precision to which the location reading is reported, and response time is the rate at which the application is updated with a new location. The latter is being formally dubbed as position reporting frequency (PRF) [33]. Though not immediately visible, PRF is of particular importance for it is [33] "one of the means that define the overall perceived quality of the LBS for the end user, not just a means to determine user's exact position." There are two QoS metrics, namely positioning availability and continuity of service, that are influenced indirectly by PRF. Firstly, when perceived from the application side, PRF can have an adverse effect if a satisfactory level is not maintained. Secondly, PRF can strain the communication layer and cause it to fail altogether if set to a high value since it places unnecessary data traffic on the channel e.g. highly interactive gaming applications. [27] adds another parameter to QoS: availability, and defines it as the percentage of time and space in which LBS is operative. Finally, [104] concludes that many QoS issues remain to be fully addressed such as: accountability for accuracy, availability of location information, prioritization, PRF, the user's freedom to opt-in and opt-out of services, the transparency of transactions, and duration of location information storage.

2.2.4 Location-awareness for wireless sensor networks (WSNs)

The distinction between localization for ubiquitous computing and that for WSNs is essentially a blurry one. It might at first appear that WSNs place stricter emphasis on form factor and power consumption than ubiquitous computing applications. However, the fact of the matter is that these "artificial barriers" (form factor and power consumption) are mostly application-dependent rather than absolute limits on the usability in WSNs. In other words, there is nothing that various technological options can not remedy (miniaturize and cut down on its power consumption) subject to a strong drive by suitable applications. As an example, consider the case of the widely used optical mouse. The optical mouse employs a basic chip that performs image processing to determine the displacement the mouse undergoes when moving. It would appear that this "extra" processing overhead, when compared to traditional mechanical mice (based on rotary sensors), is unjustifiable. Nevertheless, due to mass-production, optical mice were able to compete in the market and in fact are almost the mainstream choice nowadays. This is a very

faithful illustration of a niche user-centric application that was able to take off and enjoy huge success.

Having stated the above, the trend for WSNs localization is to use the same communication resources to maintain as much compact integration as possible. Pushing toward this, UWB Radio is being extensively investigated as it holds the promise for short range communication with fairly accurate positioning capabilities (typically $\sim$20 to $\sim$30cm accuracy) [123, 56, 102]. However, the realities surrounding UWB are different due to many outstanding issues in this relatively immature technology. [117, 82] argue that realistic indoor environments typical of office and industrial premises remain a challenge for UWB due to multipath fading, limiting the technology's theoretical performance. In addition, to simplify the complexity of receivers and interference[4] (both ISI and MAI) [111], current UWB systems require a means for synchronization either by a physical wire [50] in the case of a centralized location system, or by time division multiplexing (TDM) [56] in the case of wireless embedded nodes (round-trip measurements). This in turn affects PRF, an important QoS metric, when the number of tags/nodes increases in the environment. Moreover, despite regulations UWB has compatibility issues with various current RF standards [35, 90] giving rise to complex mitigation techniques. Lastly, [65] states that

> Many open questions exist also in the areas of system scalability (large number of UWB devices operating in a given area), mutual interference between similar and dissimilar devices, required and achievable level of QoS, to name a few. Concerning localization, it will be necessary to determine the required level of accuracy of any given application and whether this level of quality can be maintained under varying channel and network load conditions and dynamics.

Just recently, some vendors have started offering physical transceiver chips with location engine for WSNs applications [70]. This indicates that, since location is such a fundamental element of context, positioning capabilities are better realized in hardware to relieve the nodes' software, which results in more efficient use of the nodes' computational resources focusing on the functionality in hand.

2.2.5 Summary

Low-rate indoor tracking with coarse granularity is readily available using WiFi fingerprinting. Fine-grained positioning can be achieved with a variety of methods. These methods possess differing performance trade-offs which are application-dependent. Therefore, low-rate indoor tracking continues to receive considerable attention [84]; the classic issue of balancing accuracy, update rate, and deployment/calibration cost remains at large. As a consequence, many indoor applications have not yet flourished into fulfilling their application requirements compared to their outdoor and urban counterparts. This leaves an intriguing avenue for future research to propose a low-rate, fine-grained indoor tracker

[4] This is still a very active research topic.

that can be more readily adapted to various application needs in an increasingly diverse usage space.

2.3 High-rate Tracking – A Survey

Historically, the need for high-rate tracking have been driven by computer graphics systems. Computer graphics systems are built either for serious applications such as flight simulations and medical instrumentations, or for less mission-critical tasks such as video games. In this context, high-rate tracking is often referred to as motion tracking. The wide array of available technologies for motion tracking has received exhaustive treatment in prior literature. It will be shown that the examination presented earlier for low-rate tracking remains applicable for the high-rate case too.

In what follows, a quick review of various enabling modalities will be given. For more information on motion tracking state-of-the-art, the keen reader is referred to previous excellent surveys for a lengthy exposition [52, 28].

2.3.1 Mechanical

Somewhat outdated, mechanical sensing relies on direct contact with the trackable object in order to infer position and movement. Elaborate arrangement of mechanically inter-connected pieces is eventually linked with electromechanical transducers for data conversion. Then the knowledge of the mechanical device is incorporated to interpret the measurements and arrive at the object's motion estimate. Examples of commercial products are MetaMotion's Gypsy [8] and Fakespace's Boom [4].

Also classified as mechanical are contact-based systems that react to exerted force, such as pressure sensors under floors to track pedestrians [17]. Pattern detection techniques can then be applied to work out motion.

Not surprisingly, mechanical tracking is robust given the direct contact. It, however, is of limited coverage and can be cumbersome.

2.3.2 Inertial

Modern inertial sensing utilizes two units of three orthogonal gyroscopes and three orthogonal accelerometers in order to estimate orientation and position, respectively. The main strength is that measuring devices are compact and self-contained. Inertial systems handle navigation well but still require a supplementary modality for initial positioning and the inevitable drift subsequently. Examples of commercial systems using inertial sensing are the Wireless Motion Tracker from Xsens [15] and the NavShoe from [7].

2.3.3 Magnetic

Magnetic sensing is composed of a transmitter operating in either AC or pulsed DC fashion and a receiver unit. The receiver unit includes three orthogonal coils which enable

the estimation of orientation and location using the field strength. The technology has a number of general advantages. Accuracy could reach sub-centimeter and sub-degree in position and orientation, respectively. Typical applications are in computer graphics, virtual reality, and medical instrumentation. A variety of commercial products are available from Polhemus [11] and Ascension [1].

Magnetic tracking has powerful features when used for human-centric tracking. First, tags can be quite compact. Second, the ability of fields to travel through the human body translates into no line-of-sight requirement. Third, true multiple access can be achieved by exciting a number of receiver tags concurrently.

Shortcomings of the technology include the following: First, the range of coverage is short due to the cubic attenuation of fields. Second, performance is susceptible to degradation resulting from interference of conductive objects, especially in unprepared, non-open indoor environments. Third, procurement is quite costly and installation is tedious.

2.3.4 Optical

Optical sensing is further divided into two categories: computer vision-based and active LED-based.

Computer vision-based systems employ cameras and image processing to track fiducial markers and determine their position and orientation. Fiducial markers could also be either passive or active. In restricted spaces, measurements could be quite accurate and responsive at the expense of extensive setup. The tracking of generic objects using feature detection remains a challenge due to the dynamic variability of medium conditions e.g. light levels and the associated high processing workload of vision kernels. The OptiTrack from NaturalPoint, Inc is an example of active motion tracking system for professional studios [10].

LED-based systems consist of units emitting beams of infrared or laser that are worn by the user. Receiver panels are installed thoroughly in the environment in order to provide sufficient coverage for the narrow optical beams. Under guaranteed line-of-sight and heavily instrumented space conditions, such systems can be quite accurate. However, these conditions can not be always guaranteed. An example from the literature is the HiBall infrared-based system [146].

2.3.5 Radio Frequency

Indoor Radio Frequency sensing has converged in the past few years to signal strength-based and UWB-based. Coarse tracking in buildings can be realized by fingerprinting techniques using signal strength. UWB signalling can on the other hand supply fine-grained tracking capability to within 15cm accuracy by estimating the time of flight. An example of an UWA commercial system is Ubisense [14]. UWB can only begin to estimate velocity when derived from high update rate position estimates (40 kHz).

Among the disadvantages of the technology are the requisite infrastructure, co-existence

between various standards sharing the spectrum, and increasing RF congestion in typical indoor environments. These limitations have been discussed thoroughly for UWB in section 2.2.4.

2.3.6 Acoustic

Acoustic systems work by estimating the time-of-flight of a signal. Sound-based systems are both intrusive and susceptible to human activities. Narrowband ultrasonic systems are widely used for the provision of low-cost, fine-grained tracking [88]. The Cricket system has been extensively used in the Ubicomp and sensor networks communities [125, 108].

Narrowband Ultrasound

In narrowband ultrasonic systems, ultrasonic pulses are sent between devices (or *nodes*). Receiver nodes record the *times-of-arrival* (TOAs) of incoming pulses. In a system where there are multiple receiver nodes with known location, the TOAs can be used to estimate the position of a transmitting node. The accuracy of the positioning system relies upon the receiver nodes' ability to reliably estimate pulse times-of-arrival.

Recently, there has been an emphasis on ad hoc, infrastructureless systems due to their utility for uninstrumented and unprepared environments. For ad hoc ultrasonic localization [62], it is common for each transmitter node to emit an RF signal to trigger nearby receiver nodes, prior to sending its ultrasound pulse. This allows receivers to directly calculate TOAs (and thus a transmitter-to-receiver range) from the difference of the arrival times of the RF signal and the ultrasonic pulse. However as the number of transmitter nodes in such a system increases, they must negotiate to share the RF channel. For the simple CSMA radio communication schemes (such as 802.15.4) in use in these very low power, embedded networks, channel sharing becomes non-trivial as the number of simultaneous contenders increases [152].

2.3.7 Summary

Perhaps best captured by Welch et al. in "Motion Tracking: No Silver Bullet but a Respectable Arsenal" [145], various enabling technologies often excel at a particular usage scenario while being less adaptable to different requirements: mechanical sensing necessitates direct contact, inertial sensing suffers from drift, magnetic sensing is liable to interference from metallic objects and is of decreased range, optical sensing is vulnerable to changing light conditions and in need of extensive setups, RF has open issues to do with sharing the spectrum, and acoustic sensing calls for centralized coordination.

The jury is still out on a superior system capable of fulfilling all the diverse requirements in an increasingly complex application space. Therefore, further work on motion tracking is clearly justified—In the least, it may result in yet another "piece of ammunition" for the motion tracking "arsenal". At best, a disruptive technology might materialize and prove more amenable to many different adaptations across the application space.

2.4 Tracking Properties

2.4.1 Limitations of Existing Systems

The limitations of current tracking approaches as exposed by the discussion on low-rate and high-rate indoor systems are pinpointed as follows.

Deployment-difficulty Traditional high fidelity tracking has been primarily enabled by centralized, infrastructure-heavy systems e.g. vision, optical, magnetic, UWB, or ultrasonic. Installation and calibration overheads (involving extra time, special equipment, expert knowledge, and cost) do not lend these systems to situations wherein the rapid deployment of service is of paramount importance.

Narrowband Ultrasound Despite being the most widely used modality for practical fine-grained indoor positioning [88], conventional narrowband ultrasound scales poorly with node-to-node range and the number of participating users. This is because the narrowband pulses must be time multiplexed to reach destination without collisions, before a new time-slot in a frame can begin. Some degree of distributedness can be accomplished though by shifting control to RF with the downside of incurring increased traffic which might not be favourable in an already RF-congested environment. Additionally, narrowband ultrasound remains

- inherently single access
- difficult to encode unique identification in ranging signals
- vulnerable to ultrasonic noise

2.4.2 Desirable Properties

Having surveyed available tracking technologies, a tracking system has to possess certain properties in order to lower the cost and deployment barriers of indoor tracking applications. This is especially true when attempting to support demanding applications involving spontaneous interaction and real-time navigation. Thus, with the aforementioned treatment in mind, a set of *combined* properties are believed to improve the state-of-the-art in indoor tracking by virtue of synergy. These are:

High-fidelity The concept of high-fidelity tracking encompasses in turn the following:

- *Accurate tracking information*: It is this property that allows tracking accuracy to be maintained while moving throughout the operation volume with fewer nodes deployed.
- *High update rate*: It enables high measurements refresh rate taken at all nodes in the environment to supply correspondingly high aggregate tracking frequency to

applications in support of even the most demanding indoor location-aware applications.

- *Richer location models*: Extending the capabilities of the tracking nodes to supply velocity information (and optionally orientation), in addition to location, can increase the robustness of tracking algorithms and can enable enhanced, seamless interaction paradigms between users and applications.

- *Service provision which maintains performance competitive with current systems even as the number of users changes*: The user has the freedom to opt-in and opt-out of the service with no implication on the service itself that must scale potentially to over a hundred users. Also preferably there should be no need for prioritization. For example, subscribing more users to the service does not affect update rate nor does it necessitate a reconfiguration of the service—which could incur latency due to extra communication.

- *Error reporting estimate*: A measure of the "goodness" of the tracking estimates should be provided to the application. Plausible means to accomplishing this may include physical-layer algorithmic primitives (e.g. channel estimate status) in addition to high-level integrity of solution fit (w.r.t the particular nodal distribution i.e. geometric dilution of precision GDOP). These should be developed taking into account the capabilities of the underlying hardware.

Minimal deployment It is the desire to offer applications high-fidelity tracking service similar to that of infrastructure-reliant systems, yet minimizing the associated effort such requirements entail. Therefore, the system should be rapidly deployable without tedious installation.

2.5 Airborne Broadband Ultrasound

Airborne broadband ultrasound (ABU) covers a wide stretch of inaudible acoustic frequencies which have been used to provide *scalable* (i.e. multiuser) and highly robust ranging and localization of static tags in indoor environments [61]. However because ABU signals are wideband and propagate relatively slowly, they present two primary challenges which thus far have limited the development of practical ABU systems for indoor tracking. These challenges are:

1. *Real-time, embedded processing*: Efficient real-time architectures for ABU do not exist in literature. To draw analogies with the established RF field, ad hoc tracking over an ABU channel calls for a miniature asynchronous basestation capability in every device of a batch of co-located users participating in tracking. Such capabilities will allow for a peer-to-peer operation whereby:

- only a single broadcast-style RF trigger would be needed to indicate the start of a ranging time interval for a group of co-located nodes. No additional RF traffic would be incurred by adding nodes to the system.
- simultaneous true multiple access in ABU

2. *Mobile Doppler tracking*: continuous ABU sensing enables the inference of relative velocity by means of phase-coherent methods.

The new notion to be advocated for is involved processing for the novel ABU modality. The aim is to support a new class of emerging, infrastructureless systems that place particular emphasis on uninstrumented and unprepared environments.

2.5.1 ABU Principles

At the transducer level, inexpensive Polyvinylidene Fluoride (PVDF) films [137] are utilized for creating transmitter and receiver devices. PVDF transducers have been shown to be effective for accurate, highly robust, and scalable indoor localization [60]. The combined transmitter-receiver ABU channel extends to around 50 kHz of usable bandwidth.

The general principles of positioning over the ABU band remains similar to narrowband ultrasound. However, with the ability to have overlapping ranging messages, looser coordination between devices suffices. Both centralized and distributed systems can take advantage of this. The gains are however more pronounced in peer-to-peer sensing, wherein a back channel (i.e. RF) is typically used for frame control. In the ABU case, only one broadcast RF message is required to maintain sychronicity between co-located users.

2.5.2 Ad Hoc, Mobile ABU

The chief advantage afforded by ABU is the ability to have overlapped messaging through the use of CDMA [154, 64]. This singular fact is argued to be more in line with the ad hoc applications the novel modality seeks to service. In addition, the acoustic modality's bandwidth and wavelength allow for intensive processing that can still be performed in real-time under form factor, power, and/or cost constraints. Unlike RF, ABU can natively sense motion resulting from human-scale activities through the Doppler effect. This affords physical-layer primitives that are correlated with movement. By acting on these primitives, not only derivation of range and speed, but also switching between system tracking modes (i.e. stationary or Doppler-tolerant) can be accommodated. This will empower mobile computing applications with additional physical sensing capabilities. It will also open a rich, untapped resource for future research on PHY, MAC, and application design utilizing ABU's Doppler effect. Further, from a computational embeddedness standpoint, modern reconfigurable fabrics provide an ideal vehicle for the coming of age of ABU devices whose functionalities adapt to users' needs, in ways currently unachievable and intractable to high-speed RF signalling.

2.5.3 Application Domains and Considerations

The primary beneficiary of ABU would be mobile, ad hoc applications. Examples include inexpensive motion capture for gaming, support for spontaneous interaction in pervasive environments, computer-assisted living, or robotics. On-the-move users might be able to opt in or opt out unannounced without interruption to the provision of service. No extensive prior instrumentation of spaces should be necessary, nor would end-users be expected to perform specialized tasks during setup and/or operation.

Before delving into the details of the proposed endeavour, it is important to step back a bit and present a semi-technical discussion informed by knowledge derived from the application domain. To this end, the following will link the approach taken in developing airborne broadband ultrasound to the target indoor ad hoc environment.

Propagation The airborne acoustic channel for ultrasonic frequencies is characterised by a different decay profile to that of RF. Together, absorption and the inverse square law of spherical propagation result in nearly 8 dB attenuation (at 50 kHz) in a signal's sound pressure level (SPL) as its distance from the source doubles. Thus, multipath arrivals from far away are not expected. However, relatively short-delay multipath can occur, due to hard, smooth surfaces such as windows. Also, transmitter-receiver phase variations due to node mobility affect the order-comparable carrier and code frequencies (50 kHz and 20 kHz in our system). Therefore, standard RF spread spectrum receiver designs would necessarily fail if one were to apply them directly to airborne ultrasound.

Signalling The inter-connectedness of design space parameters governing the ABU operation in the target environment is captured in figure 2.2. With the above application domains (wearable battery-powered tags; easy-to-install infrastructure) in mind, form factor and portability are two requirements aggressively sought after. Therefore the use of high coding gain is mandatory in order to transmit at reduced power levels while simultaneously maximizing range and compactness. A by-product of this is long distance coverage per code period.[5] Henceforth, the system's maximum node separation is assumed to be dictated by the spreading code length, and acquiring timing readily derives range in sub-chip resolution depending on the chip oversampling rate employed.[6] This can be thought of as a "standing" wave extending between a transmitter-receiver pair. Provided that the receiver begins code acquisition at the same time of transmission, the code epoch as seen by the receiver will denote the distance offset. The assumption simplifies the system design and would be important for the discussion in this thesis.

Detection modes There are two detection modes that an airborne broadband ultrasound receiver needs to support:

[5] This can be seen by converting code duration into distance using the speed of propagation i.e. $c = d/t$.

[6] In DS CDMA systems, a chip refers to a bit within a spreading code.

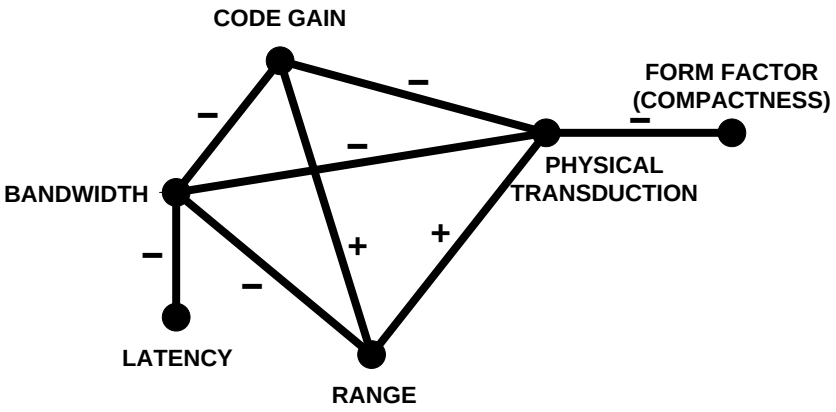


Figure 2.2: Graphical portrayal of ABU's signalling design space adapted from [19].

- *Static*: In this case, nodes are stationary most of the time and rely on TOA inference after a broadcast-style RF trigger has been sent to initiate the pseudoranging[7] interval for a group of co-located users.

- *Doppler-tolerant*: In case of severe Doppler distortion resulting from motion (e.g. caused by someone walking while wearing a node), a more sophisticated receiver mode must be devised. In order to keep the multiuser problem tractable, and depending on the tracking interval, a number of unmodulated codes are sent that would train the receiver adaptively in a mode of operation designated as *joint range-speed tracking*. Here, acquiring timing readily derives range in sub-chip resolution depending on the chip oversampling rate employed. Adaptive training also produces chip-rate phase variations as a by-product of coherent processing which can be used to estimate node velocity.
  This mode is analogous to the rationale of Sutherland's pioneering head-mounted display system [135]. Sutherland proposed a continuous wave source transmission (i.e. standing wave) whereby the relative phase is tracked in order to estimate the distance at the receiver. Sutherland's proposal had two shortcomings. First, the relative distance is measured only within a cycle. Meaning, absolute distance is worked out by keeping track of initial distance and the number of accumulated cycles. Second, multipath receptions of the continuous wave will inevitably interfere with the direct path and degrade the measurement setup. Both issues are overcome in the ABU method proposed in this thesis. First, by making a "standing" spreading code equate to the maximum expected distance, code timing readily achieves distance disambiguation, of course provided that transmission and acquisition commence at the same time—The RF broadcast-style trigger message will guarantee this. Second, in direct sequence spread spectrum (DSSS) signalling, multipath arrivals at the receiver are resolvable, alleviating the problem in narrowband waves.

[7] This term is borrowed from the GPS world and is intended to reflect the uncertainty of the measurement process.

However, in order for ABU to advance the state-of-the-art in indoor spatial sensing, a number of fundamental design requirements needs to be addressed; It is these requirements that the methods developed in this thesis aim to satisfy.

Asynchronous, multi-user The property refers to the need to have no constant synchronization between a transmitter-receiver pair operating in the indoor environment. It mostly originates from the application domains. The successful realization of an asynchronous, multi-user operation greatly simplifies deployment and enables richer interaction schemes. At the same time, when compared to classic narrowband systems, immediate enhancements to performance are obtained in the form of

- vast scalability
- robust estimation by virtue of coding gain

Minimize out-of-band signalling For DSSS ultrasonic ranging operating in an indoor environment, out-of-band signalling refers to any form of packet header which is to prefix the original ranging message and be used as a means of obtaining coarse synchronization. Examples of out-of-band signalling techniques are the widely used ambiguity function (see [121] and references therein), or short spreading sequences designed to provide coding gain in the presence of certain expected Doppler levels. There are two reasons that render these techniques inappropriate for the indoor airborne broadband ultrasonic environment. First, decreasing coding gain will in turn dictate transmitting at higher power which runs counter to the portability and mobility aims, especially given the low transduction efficiency of PVDF piezo films in the ABU frequencies. Second, the multi-user operation will doubly increase in complexity; both in RF and ABU—It will necessitate the heavy use of the back RF channel for the arbitration (time slotting) of Doppler packets. It will also require using a bank of matched filters to scan (albeit coarsely) for Doppler shifts in order to obtain initial synchronization. Consequently, if we define the *ad hocness* of a system as the degree to which centralized control is needed for the successful execution of the system's functionalities, it follows that out-of-band signalling compromises ad hocness and nullifies some of the advantages of ABU; namely, avoidance of extensive RF control messaging.

Motion-tolerant Motion-induced Doppler distortion for airborne *broadband* acoustics has hitherto not been treated in the context of wireless communications. Narrowband Doppler techniques for ultrasound are in practice [94]. The time-dilation or compression that airborne acoustics undergo is of the same order (up to a tenth of the center frequency) often encountered in RF inter-satellite communications. As discussed earlier, long ranging signals are a necessity in our ABU band. When subjected to the Doppler effect, these ranging signals undergo aggressive time-varying expansion or compression. That is, the concept of Doppler coherence time in relation to the symbol (i.e. code) does

not hold. To put it differently, the time-dilation or compression affects parts of the ranging message non-uniformly. It is due to this fact that Doppler compensation can only be conducted adaptively at a much finer resolution to that of the ranging message.

Channel-based For long spreading codes, it was established earlier that the channel has a one-to-one correspondence with a very important physical quantity: range. Devising an algorithm which natively relies on this quantity has immediate advantages for aiding the derivation and monitoring of range e.g. when multipath is encountered, it can be passed on to upper network layers fusing spatial information. Another example is reporting channel power as a means of inferring a confidence measure for tracking. Other non-immediate benefits include the coarse time synchronization afforded by the channel which can be used to selectively switch to nonlinear equalization in order to aid the initial convergence of adaptation.

2.5.4 Contributions of this PhD research

Given the examination of the state-of-the-art presented thus far, this thesis will pursue further development for the novel ABU modality on two categoric frontiers of equal importance: amenability to embedded real-time realization, and adaptive ABU detection for velocity estimation. The former will span two parts: DS CDMA PHY Transceiver Groundwork (Part I chapters 4 and 5) and High-Level Synthesis Exploration (Part III chapter 8). The latter will be dedicated a self-contained, comprehensive treatment in Coherent ABU (Part II chapter 6).

Embedded, Real-time Realization

As discussed earlier, the myriad tracking technologies already out there possess varying performance tradeoffs. The adoption of the ABU modality as a viable candidate to supplement mature established technologies is contingent on successfully demonstrating the feasibility of embedded, real-time ABU operation. For the static mode of operation, this demonstration will be carried out in Part I chapters 4 & 5, where:

1. an all-digital ABU transmitter is designed and implemented. The transmitter supports parametrization of system-level decisions such as chip rate and roll-off factor. Unlike previous work, it is shown that ranging messages can be halved in duration for increased update rate.

2. an efficient multiuser despreader kernel is proposed, implemented, and empirically verified. The despreader enables (1) embedded ad hoc ranging for devices of minimal resources (2) complex functionalities such as multiuser beamfroming when many despreader cores are replicated.

For the Doppler-tolerant mode of operation, Part III chapter 8:

1. explores the architectural implications of Coherent ABU in order to gauge the potential of an embedded, real-time, and Doppler-tolerant tracker. These architectural findings should also be incorporated into the thinking for driving further algorithmic research.

Coherent ABU

A multiuser, adaptive phase-coherent ABU receiver represents a novel motion tracking technology. Part II Chapter 6:

1. derives an ABU purpose-built acquisition and tracking algorithm by building on previous Underwater Acoustic (UWA) communications research.
2. establishes the theory of motion tracking in the novel ABU band
3. studies by means of empirical data motion tracking in ABU and considers the permutations of various phase tracking components and their interaction.

PART I

DS CDMA PHY TRANSCEIVER GROUNDWORK

CHAPTER 3

Preliminaries

This chapter briefly goes through the hardware design practices utilized in the designs presented in this part of the dissertation. An overview of FPGA devices is first given. Then the FPGA-based signal processing design flow is discussed.

3.1 Field Programmable Gate Arrays

Modern Field Programmable Gate Arrays (FPGAs) are reconfigurable integrated circuits which consist of a large grid of interconnected components. The Xilinx Virtex family will be used to drive this introduction. Equivalent devices from other vendors share the same characteristics, but with different terminology. The following will enumerate the components of the Virtex family. Where applicable, brief commentary on their utility will be provided.

- Configurable logic blocks (CLBs): CLBs are the main *fine-grained*, *generic* logic resources within an FPGA. CLBs are further divided into *slices* which in turn contain look-up tables (LUTs) and flip-flops (FFs). Multiplexers and carry logic are also included within a slice to facilitate the construction of algebraic and arithmetic expressions. LUTs and FFs are utilized for building distributed RAMs and shift registers.
- Routing interconnect: A special multi-layered network of switching logic is available to provide connectivity within the chip, both locally and globally.
- Block RAMs (BRAMs): Dual-port BRAMs are configurable in depth and word width. They are arranged in columns inside the device for storage and computationally-intensive applications requiring cache to maintain throughput.

- DSP slices: For scalable performance, dedicated blocks for realizing the widely recurring multiply-accumulate (MACC) operation are available in quantity. The DSP slices can also be configured at run-time by changing the OP mode in order to realize a host of arithmetic and logic functions. DSP slices can be cascaded into a variety of filtering structures with guaranteed performance.
- Clocking resources: Blocks responsible for digital clock management along with their distribution trees are present in FPGA devices.

Fine-grained logic fabric is available in abundance. Comparatively, other custom blocks (e.g. BRAMs) are less in quantity given their much larger silicon area. The terms *soft* and *hard* are often used to distinguish between the fine-grained fabric (LUTs and FFs) and the coarse-grained blocks (BRAMs and DSP slices), respectively.

3.1.1 Design flow

The design of complex end-to-end DSP applications utilizing FPGAs is regarded as a challenging task. The use of systematic design methodologies is mandatory in order to overcome complexity and ensure the timely completion of design tasks. Figure 3.1 shows the FPGA-based signal processing design flow adhered to in the first part of this dissertation (chapters 4 & 5).

Three methodologies are used interchangeably for realizing various system components. This is because there exists no one simple approach that can accomplish all design objectives simultaneously. The following will briefly touch on these three design approaches.

HDL

Hardware description languages (HDLs) have traditionally been the medium of choice for describing the behaviour of digital logic at the register transfer level (RTL). Regardless of the methodology employed, modern tools translate higher abstraction levels of a design into HDL RTL. This is done in order to capitalize on decades-worth of development that resulted in mature robust technologies. It is therefore imperative that designers be able to work at the RTL level at least for integration purposes and/or low-level control and configurations. The HDL flavour utilized for this doctoral work is VHDL.

System Generator

The Xilinx ISE tool suite includes a DSP-oriented tool that allows the building of designs graphically using MATLAB's Simulink environment. The Xilinx blockset includes hardware macros like memories, multiplexers, DSP slices, etc. It also contains higher DSP constructs for larger blocks such as FFTs and filters. All the powerful features of Simulink remain accessible from within System Generator. The tool can also compile designs targeted for co-simulation. Co-simulation allows for the evaluation of designs running on real hardware. In the so-called "hardware-in-the-loop", data is communicated to/from FPGA

devices in order to compute DSP statistical metrics for a given design that would have taken long time to produce in simulation e.g. bit error rate (BER). This procedure is also considered as a formal verification of functionality.

AccelDSP

AccelDSP is also a DSP-targeted tool from Xilinx. The main advantage of AccelDSP is the automatic generation of optimized fixed-point designs from their floating-point counterparts. However, this only applies to stationary DSP processing with predictable, deterministic error models. A rudimentary set of directives is available for limited architectural exploration such as unrolling filtering loops.

The overall design flow is depicted in figure 3.1.

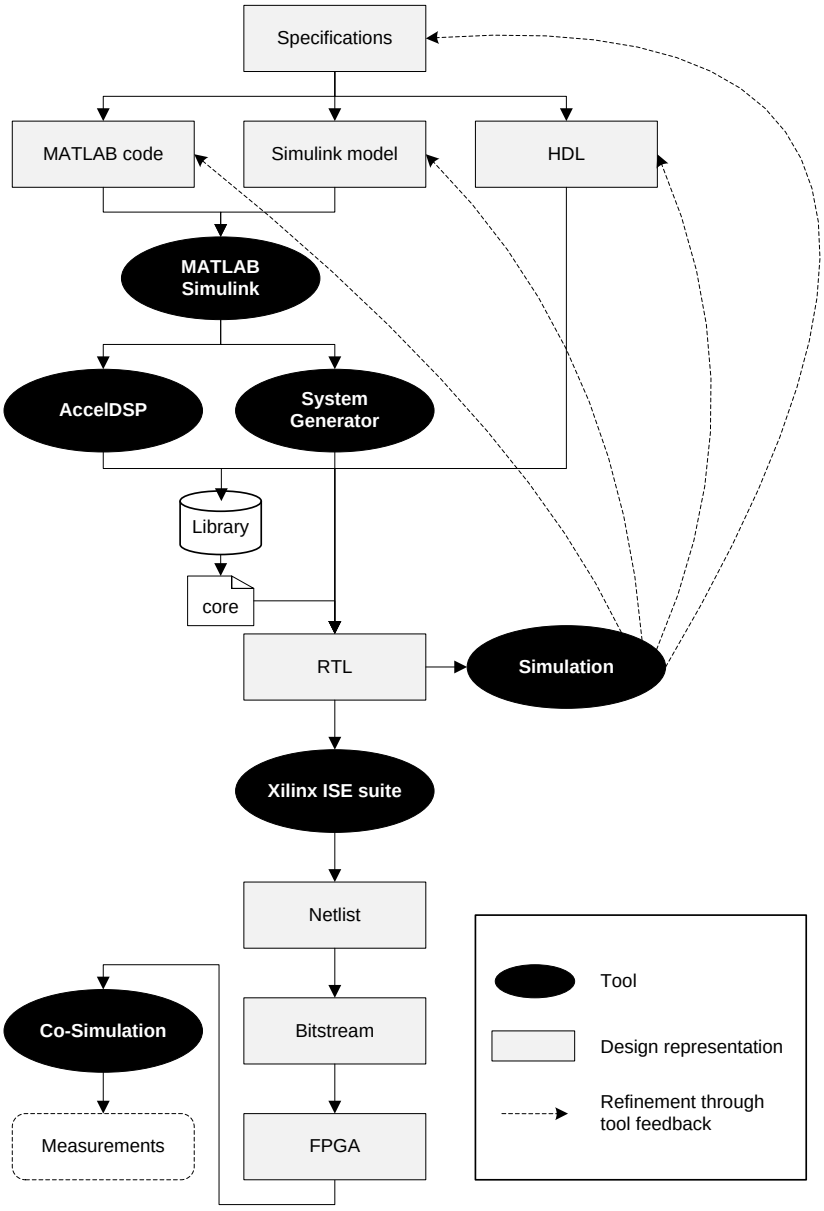


Figure 3.1: DSP Design flow using Xilinx's ISE tool suite

3.2 Summary

This chapter provides background material on FPGAs and their associated design flow from a signal processing perspective. In the following two chapters, this design flow should be assumed. That is, beyond this point, no specific reference will be made with respect to the practical methodologies implicitly featuring in the remainder of this part.

CHAPTER 4

All-Digital Airborne Broadband Ultrasonic Transmitter

This chapter details the design and implementation of an all-digital airborne broadband ultrasonic DSSS modulator intended for use in handheld or tethered transmitter nodes. Section 4.1 details the requirements that had to be considered in the design process. Section 4.2 presents the ABU transmitter architecture. By means of a case study of real measurements, section 4.3 illustrates how the transmitter architecture meets design objectives in relation to ABU signalling. The area metrics are also provided. Section 4.4 comments briefly on the utilized design flow. Finally, section 4.5 concludes with closing remarks on ABU system-level considerations.

The transmitter is compact in terms of area and flexible in terms of parameterization. A major aspect of the design is in the careful economy exercised. This is carried out as to allow the power efficient ASIC-like, but sparse FLASH FPGAs to comfortably host the required logic. The transmitter is shown to have signalling performance commensurate with the sensor noise floor of piezo films in ABU transmitters and receivers.

4.1 Requirements

As alluded to in the introduction, the transmitter architecture is meant to fulfill a set of system requirements. Specifically, these are:

- Flexibility: Due to the fact that airborne broadband ultrasound for ranging applications is still pretty much a research endeavour, it was necessary to keep this early investigation as flexible as possible, enabling a platform for further future fine-tuning

and real-world, deployment-driven experimentations. In more established modalities such as GPS, mature and stable off-the-shelf components (including custom silicon) that are tailored to the task at hand exist. For example, designed-to-purpose analogue reconstruction filters can be found with optimal application-dependent responses (e.g. pulse shaping). These parts are oftentimes designed with a host of other system parameters in mind. The route, on the other hand, taken in the ABU design presented here is not perfectly optimized, allowing more flexibility. To this end, the DAC exciting the transmitting transducer is oversampled, and the reconstruction stage which follows is relaxed as not to affect the applied digital processing.

- Programmability: The ability to sweep system parameters in a programmable fashion also has implications on the choice of signal processing realizations for various subcomponents. This is because certain filter structures are more restrictive and difficult to adapt and leverage for functionally-equivalent operations. For instance, a perfect linear-phase bandpass filter can be easily realized at a low oversampling rate, but less so when the oversampling rate goes up. Consider the need for tuning the raised-cosine roll-off factor according to the cumulative distribution function (CDF) of the ranging error—in the area of RF communications, similar pulse shape tuning is often carried out with the objective of minimizing the bit error rate (BER)—For some permutations of the roll-off factor and chip rate, there could be significant spectral spillage in the audible range necessitating bandpass filtering, which may or may not be realizable as a linear-phase FIR depending on the oversampling rate. Therefore, there is an ever present trade-off between parameter-coupled optimizations and functional generality.

- Efficiency: While catering for the above two requirements, the design still has to exhibit a good degree of efficiency as to remain amenable to low-cost implementations, and relevant to custom silicon designs in future work.

- Reuse: Design sub-blocks are to be leveraged for other system functionalities. Channeling into the same argument from the programmability requirement, the need for an efficient bandpass filtering at higher oversampling rates is perhaps best justified when considering the receiver. Initial oversampling at the receiver enhances SNR with no implications on the remaining processing workload since the sampling rate can be reduced through decimating later [85, p. 505–507]. Such strategy could be employed for broadband FIR beamforming wherein signals need to be sampled well above the Nyquist rate [58, p. 1211]. Thus, it is desirable to design sub-blocks that could be reused in a number of places within the overall system.

The rationale behind the transmitter architecture in this chapter attempts to simultaneously address the above requirements, with good balance of trade-offs.

4.2 Transmitter architecture

A simplified block diagram of the all-digital transmitter is depicted in figure 4.1.

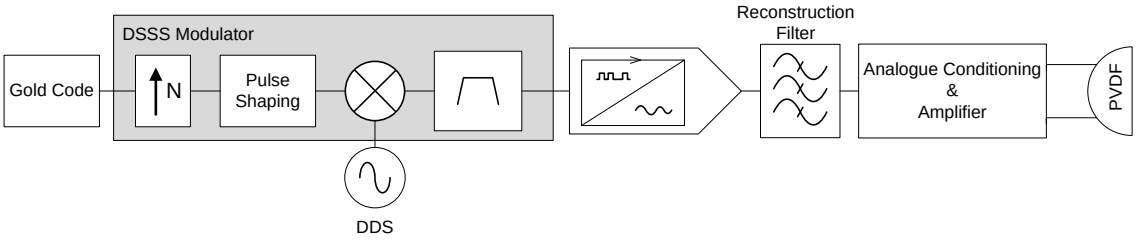


Figure 4.1: Transmitter simplified block diagram

The digital transmitter consists of a Gold code generation unit, a numerically-controlled oscillator (NCO), and a DSSS modulator. A mixed-signal DAC then follows whose output is in turn smoothed with an analogue reconstruction filter. Various signal scaling, conditioning, and amplification are applied before finally driving the piezo film.

A free-running NCO at the digital system clock (100 MHz) was instantiated. For flexibility, enables were then used to sample the NCO, subject to the oversampling rate employed. The only downside to this prototype approach is the high memory utilization. This can be easily rectified by parametrized instantiation in order to drastically reduce the memory required to store a quarter of a carrier wavelength at the oversampling rate. Lastly, the parameters used to instantiate the oscillator are listed: spurious free dynamic range of 96 dB (16-bit), frequency resolution of 0.4 Hz, and for flexibility a phase generator with sine-cosine LUT-based architecture was chosen. The remaining parameters specific to the transmitter will be stated in subsequent sections upon discussing their relevance to the overall design.

4.2.1 DSSS modulator

A sequence of chips that constitutes a packet—which could be either a positioning burst of one code, or a tracking packet with a number of back-to-back codes—is modulated as follows. The Gold code signal is upsampled by a certain oversampling factor N, by inserting zeros to the baseband stream. The oversampled code is then raised cosine-filtered. The shaped chips are mixed up to the carrier frequency and further bandpass filtered. The fractional two's complement signal is then made bipolar in preparation for a DAC write. A final piece of logic checks a chip counter and forces a zero write when the packet has been sent. This is necessary because a floating voltage level at the output may damage the piezo film.

Raised cosine filter

A raised cosine filter spanning six chip durations is used. The roll-off factor depends on other system-level parameters. In addition to band-limiting the spectral content of the DSSS acoustic signal, the filter also aids in mitigating against ICI in the ranging mode

of operation. In the tracking mode, band-limiting the chips has a “soothing” effect on the adaptive equalization by way of rejecting energy from spectral components outside the designated band. This mildly enhances tracking in the presence of the Doppler effect.

Since the signal at baseband is binary, the filter taps are only required to be signed binary, i.e. two bits of resolution. This makes the filter tap delayline quite efficient and similarly simplifies arithmetic operations within the filter. Automated fixed-point analysis with a directive applied on the needed output word length (16.15) resulted in the following internal precisions:

- input: 2.0
- coefficients: 13.12
- accumulator: 21.12

The filter is fully rolled for maximum resource sharing, bearing in mind the relatively low rate of the modality. Moreover, the interpolating nature of the filter means that zero-valued taps need not be evaluated resulting in further computational savings and better latency.

Bandpass filter

The usual approach for linear-phase filtering often seen in communication transceivers relies on multi-rate signal processing techniques. This is performed in order to change the normalized frequency and in turn yield efficient FIR realizations in the band of interest. On the other hand, signal processing for acoustics (e.g. speech) sacrifices linear-phase in favour of efficiency by often employing IIR filters for a variety of tasks. This is quite logical since phase is of lesser concern in speech. Underwater acoustic modems, by contrast, tend to be software-based using buffer-and-process approach applied to packets. None of these established modalities offer a readily viable strategy for airborne broadband ultrasound, given its different band of operation when compared to RF, different nature of processing in comparison to speech, and different real-time, form factor and power requirements when compared to underwater acoustics.

For the airborne broadband ultrasound transmitter, an elliptic IIR filter with very good phase and group delay properties around the DSSS system carrier was chosen. The band-stops are relaxed to combat any phase non-linearities in the band of interest while providing additional attenuation of -20 dB for out-of-band frequncies (e.g. audible) and 0.5 dB bandpass ripple as will be demonstrated in the following subsection by the PSD of the acoustic DSSS signal. The design parameters give around 1.5 samples of group delay at the sampling frequency (200 kHz) in the entire 30-70 kHz band, and roughly 0.6 samples in the band within which the greatest amount of transmission power lies (35-65 kHz). The final filter is Direct-Form II (DF2) of order 10, and 5 second-order sections (SOS, also called biquads). The round-off noise power spectrum for the quantized *parallel* realization is better than -90dB/Hz in the band of interest. Contrary to a perfect linear-phase FIR filter, the behaviour of the chosen elliptic IIR filter remains consistent as we increase the sampling frequency. This allows the filter to be repurposed for other system

functionalities, such as providing improved SNR at the receiver. For the realization of bandpass filter, a fully rolled core for cascaded biquads was designed in RTL description. Figure 4.2 illustrates the core.

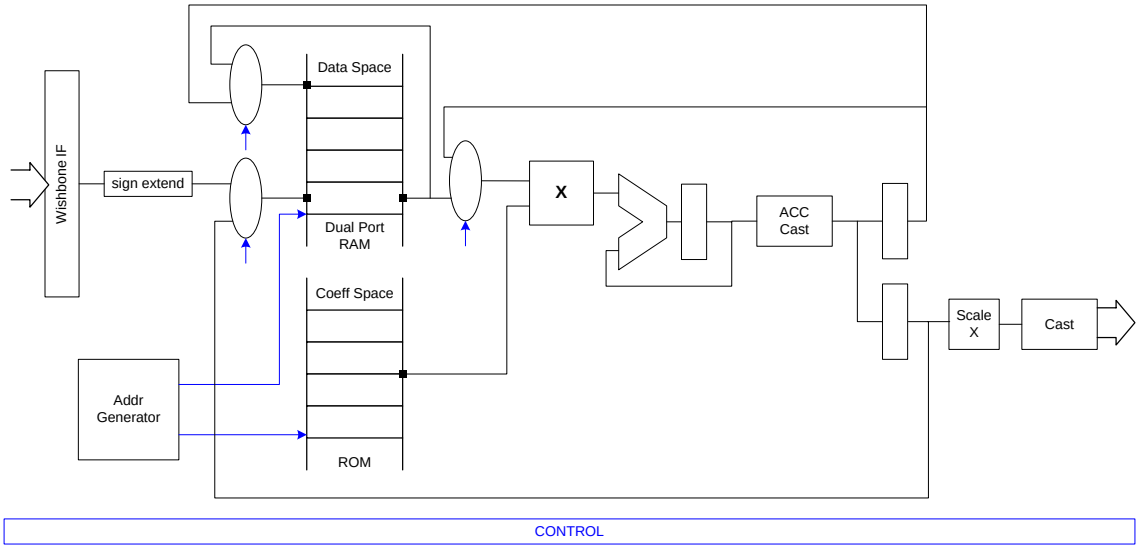


Figure 4.2: Rolled DF2 SOS core simplified diagram

A dual port RAM is designated as a data space for contiguously storing the input and states of each second-order section. Similarly, the scaling constants and biquad coefficients are held in a ROM. A control state machine realizes the sequence of operations for computing a biquad output. The output is then either fed forward to the next stage or communicated to the external logic after scaling, if applicable. The automated fixed-point filter design was manually modified for resource sharing with the following precisions:

- input: 16.15
- coefficients: 17.15
- internal states: 25.15
- accumulator: 48.30
- rounding in casts: convergent

The design of the multiuser Gold code generation will be briefly discussed in the next chapter

4.3 Case study

In order to demonstrate the flexibility being afforded by the proposed transmitter architecture, this section will present a test case of varying few system parameters. The point to be made is to show how an appropriate range of system parameters can be accommodated utilizing the all-digital transmitter architecture.

The case study will analyze the numerical performance of the ABU channel, using two chip rates with different pulse shaping filters designed to limit the DSSS signal to a designated band. For this particular test case, the band-pass IIR filter discussed above will be redundant and as such will not be included in the transmitter. Nonetheless, the rolled elliptic filter will be analyzed with respect to the aggregate transmitter-receiver ABU channel.

4.3.1 Doubling the chip rate

In contrast to RF, airborne broadband ultrasound is in its infancy. Driven by enormous scope for applications across a multitude of domains—e.g. military, consumer, etc—decades of research in RF from physical transduction to high level algorithms have resulted in a mature, stable, and usable technology. Consequently, RF algorithm developers often find themselves in a position where they do not need to compromise on low level limiting factors such as usable bandwidth. For example, GPS chip rate is free to occupy the maximum bandwidth around the carrier. Only recently, work on characterizing the relationship between pulse shaping and GNSS indoor positioning accuracy has appeared in the literature [124]. Counter to RF, the scarcity of usable bandwidth in high-rate coherent underwater acoustics entails a more careful design around the limited capabilities of the wireless channel. Airborne broadband ultrasound is similar to underwater acoustics in this respect and similar design techniques need to be incorporated.

It was established earlier that transducer excitation in the airborne broadband ultrasonic frequencies is of limited efficiency. It then follows that long spreading codes are needed in order to boost signal-to-noise levels. Further, there are situations where increased chip rate beyond that which the channel can straightforwardly sustain is desirable. A strong case for such desire is made in coherent tracking where higher chip rate could aid Doppler equalization and may prove more favourable than finer chip oversampling at lower chip rate. The criteria for system-level preference could be a number of things:

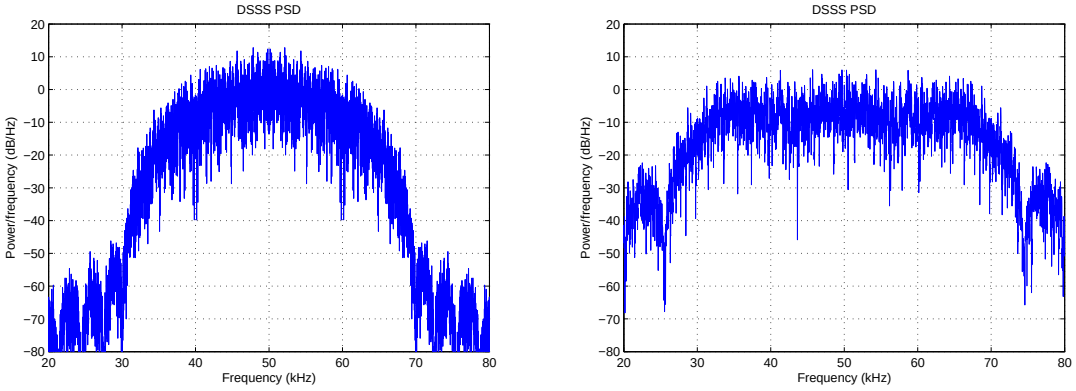
- receiver-related: e.g. required silicon area and power consumption.
- transmitter-related: e.g. power consumption or range as determined by transducer efficiency at the chosen parameter set.
- application-related: e.g. required accuracy, range, duration of ranging signal, or lifetime of service (which is in turn influenced by the transmitter's power consumption in the mobile case).

This is a faithful example of how intertwined the application design space could potentially be.

Relying on the natural frequency response of the transducer is not appropriate for two reasons. First, the crude convolution between the transducer impulse response and the ranging message will introduce spectral artifacts that are parameter-dependent (e.g. chip rate and transmission power), impacting system-level design characterization (i.e.

CDF of ranging error). This can be lessened to a certain extent by the proposed signal processing. Second, limiting the spectral content of pulses to the known frequency response capabilities of the transducer results in improved power consumption. Hazas shows that there is a dependency between the power drawn by the transmitter and the spectral content of the excitation signal [60, p. 67].

To this end, the 20 kHz (or kChips/s) chip rate was doubled and a raised-cosine filter with a roll-off factor of 0.15 was used to limit the bandpass bandwidth in order to operate within the characteristics of the combined transmitter-receiver ABU wireless channel. The remaining system parameters were: 200 kHz sampling frequency for both the transmitter and the receiver, and 10x & 5x oversampling for 20 kHz & 40 kHz chip rates, respectively. Under this setup, the PSDs of the transmitter's excitation signals as measured by the averaged periodograms method [46, p. 57] are illustrated in figure 4.3. It is clear that the doubling of the chip rate spreads the energy of the DSSS ABU signal more around the carrier (50 kHz).



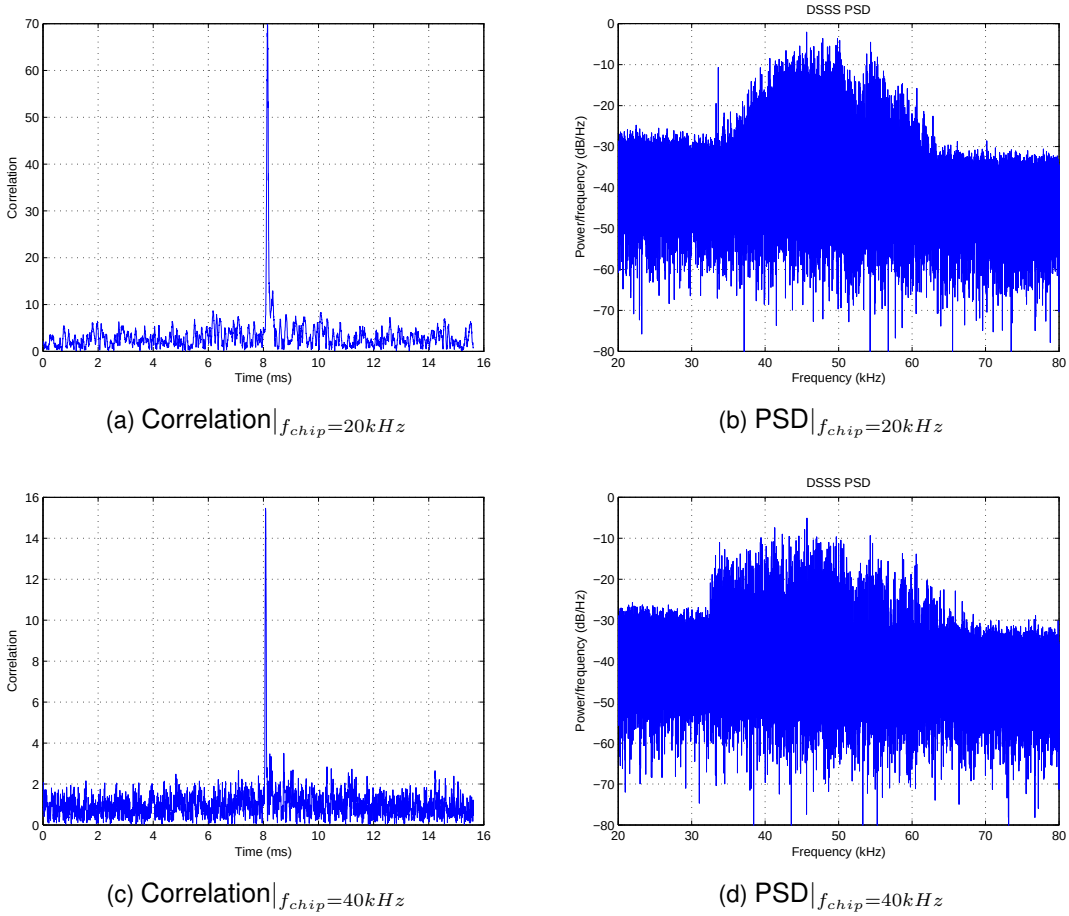
(a) Tx PSD$|_{f_{chip}=20kHz}$ (b) Tx PSD$|_{f_{chip}=40kHz}$

Figure 4.3: Tx excitation PSDs at two chip rates: 20 kHz & 40 kHz

A transmitter-receiver pair is positioned on-axis at a distance of 2.741 m[1]. The time-of-flight of a code-length burst is measured under the two chip rates. A one-second continuous transmission is sampled at the receiver for the evaluation of the PSD for both chip rates as well. Figure 4.4 depicts the correlation and PSD obtained from the two configurations.

After removing various systematic errors in the setup—e.g. the software latency of the trigger—the calculated distances from the correlation peaks of the 20 kHz and 40 kHz chip rates are 2.753 m and 2.729 m, respectively. The ranging errors for both cases are on the order of 1 cm. Inspecting the corresponding correlations of figures 4.4a & 4.4c reveals that the double rate correlation gain is reduced with respect to the single rate. This reduction is around 6.5 dB. However, so does the noise of the correlation—the faster rate seems to have "dithered" the noise better. As expected from Parseval's duality principle, the 40 kHz PSD in figure 4.4d is also reduced similarly with respect to the 20 kHz PSD of figure 4.4b. It also can be seen that the double rate PSD is more spread around the

[1] as measured by a laser range finder



(a) Correlation$|_{f_{chip}=20kHz}$ (b) PSD$|_{f_{chip}=20kHz}$

(c) Correlation$|_{f_{chip}=40kHz}$ (d) PSD$|_{f_{chip}=40kHz}$

Figure 4.4: Parseval view of the ABU DSSS system at two chip rates: 20 kHz & 40 kHz

50 kHz carrier but remains buried in the noise floor of the receiver outside the intended range due to stricter roll-off factor control.

4.3.2 Numeric system-level analysis

The transmitter architecture is modelled in both floating-point and fixed-point arithmetic. The objective is to demonstrate that the fidelity of the proposed embedded realization is commensurate with what the airborne broadband ultrasonic channel has to ofter.

Transmitter – An RMS view

Classically, in the context of fixed-point filter design of limited precision, computing the SNRs of a filter's input and output is used for the sake of improving quantization and dynamic range of input, output, and various intermediate types inside the filter. Thus the comparison is quite useful. The objective is to preserve and match the SNR of the output to the SNR of the input. This is done as follows: Let $\mathbf{v}_{in}^{float}$ and $\mathbf{v}_{out}^{float}$ be the input and output vectors of the aggregate floating-point transmitter filtering whose lengths are N. Similarly, $\mathbf{v}_{in}^{fix}$ and $\mathbf{v}_{out}^{fix}$ be the input and output vectors of the aggregate fixed-point

transmitter filtering. Then:

$$RMS_{in/out}^{float/fix} = \frac{\left|\left|\mathbf{v}_{in/out}^{float/fix}\right|\right|}{\sqrt{N}} \tag{4.1}$$

The two numeric representations of the input and output vectors are then differenced respectively.

$$\Delta\mathbf{v}_{in/out} = \mathbf{v}_{in/out}^{float} - \mathbf{v}_{in/out}^{fix} \tag{4.2}$$

The RMS values of the difference vectors are then computed as a measure of the numeric artifacts injected and the degradation the signal underwent traversing the precision-limited datapath.

$$RMS_{in/out}^{\Delta} = \frac{\left|\left|\Delta\mathbf{v}_{in/out}\right|\right|}{\sqrt{N}} \tag{4.3}$$

Note that a Δ notation was used as a superscript for the RMS of the difference vectors in order not to convey a false notion of linearity. Finally, the input/output SNRs are obtained by comparing the respective input/output difference RMSs against the ideal floating-point reference.

$$SNR_{in/out} = 20\log\frac{RMS_{in/out}^{float}}{RMS_{in/out}^{\Delta}} \tag{4.4}$$

The stimulus is baseband digital Gold code signal with infinite quantization precision. Plugging the fixed-point model of the transmitter into the above equations results in the SNRs reported in table 4.1.

Table 4.1: Transmitter SNR

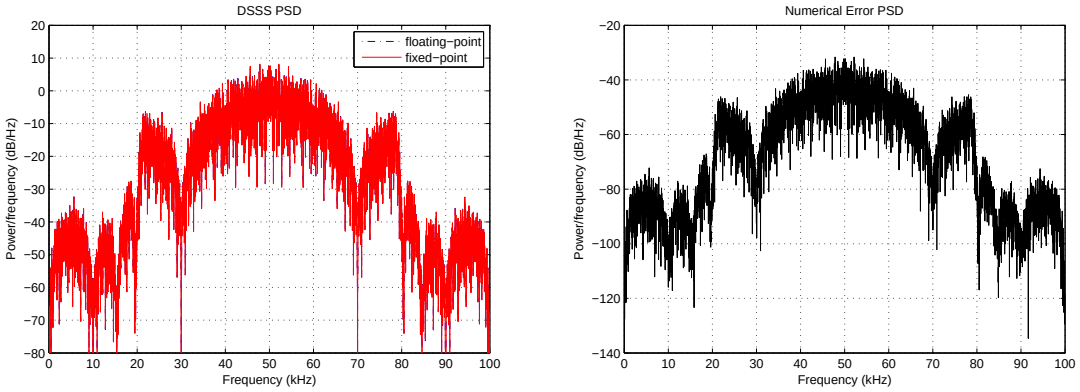
Functionality	Chip Rate (kChips/s)	Roll-off (β)	Input SNR (dB)	Output SNR (dB)
Tx	20	1.0	∞	67.64
	40	0.15	∞	63.47

Varying the chip rate has a mild effect on the SNR of the transmitter's excitation signal. Roughly a drop of 4.2 dB accompanies the doubling of the chip rate. However, the two SNRs are significantly higher that that of the aggregate ABU channel which stands at approximately 24 dB, at one meter range [60, p. 76].

Elliptic IIR – A PSD-RMS view

As mentioned earlier, the rolled DF2 SOS architecture inevitably introduces numerous rounding errors in order to share the computation datapath. In order to assess the core's fidelity, analysis with respect to the transmitter's DSSS signal is conducted. The chip rate is set to 20 kHz. The raised cosine filter is removed. The Gold code is oversampled. The up-mixed signal is bandpass filtered under the parameters enumerated earlier on

page 32. Figure 4.5a shows the PSDs of both floating-point and fixed-point representations overlaid. It is evident that the fixed-point transmitter datapath very well approximates the ideal floating point computations, to the point where error is hard to see at scale.



(a) overlaid (b) error

Figure 4.5: PSDs of the Elliptic IIR filtering in floating-point and fixed-point representations

The error between the two numerical representations is plotted. Inspecting the PSD of the error reveals that it is well below -20dB. This is again commensurate with the capabilities of the combined Tx-Rx channel of the piezo film transducers [61, sect. 5.3]. Similar conclusions could be arrived at from an RMS view as is provided in table 4.2.

Table 4.2: Elliptic IIR rolled DF2 biquads SNR

Block	Chip Rate	Input SNR (dB)	Output SNR (dB)
Rolled DF2 SOS IIR	20 kHz	∞	39.85

The core's fidelity is dereased with respect to the raised cosine FIR filter. However, the use of the core is an example of a design which seems compromised at first glance, but can only be appreciated in view of the limited transducer and acoustic channel capabilities.

4.3.3 Area

The resource occupancy of the rolled DF2 SOS core as obtained from place-and-route is listed in table 4.3. The core is compact and is independent of the filter's order due to rolling.

The silicon area of the transmitter prototype for both parameter sets analyzed earlier is given table 4.4. In situations where both the raised cosine and elliptic bandpass filters are needed, the corresponding resources in tables 4.3 & 4.4 should be added for the area estimate of the final transmitter.

Table 4.3: FPGA resource requirements for DF2 IIR core[a]

Block	Resources		
	DSP48s	BRAMs	Slices
Rolled DF2 SOS IIR	3	2	176

[a] Elliptic 10th order IIR, 5 SOSs
[b] Device: Xilinx Virtex-4 XC4FX12

Table 4.4: Tx FPGA area requirements[a]

Resource	Used	Utilization %
DSP48s	1	3
BRAMs	5	13
Slices	308	5

[a] Device: Xilinx Virtex-4 XC4FX12

4.4 Comments on methodology

In relation to methodology, three environments are utilized in the hardware design flow: System Generator, and AccelDSP, and VHDL. The transmitter DSP simulations are performed in MATLAB and Simulink. Functional-level simulations are carried out in MATLAB. Numerical analysis is also conducted in MATLAB using the Fixed-Point Toolbox. DSP-oriented simulations of sub-blocks are carried out in System Generator. Simulink's Stateflow is used to generate Wishbone-compliant [100] hardware handshake signals, feed stimuli to cores, and hand over results back to MATLAB. The automated fixed-point design for the raised cosine filter is performed in AccelDSP utilizing directives. The DF2 SOS core is designed in RTL. VHDL is used to realize the overall transmitter design and code various low-level components concerning timing, control (e.g. for interpolation), and arithmetic operations. A VHDL testbench is also used for final verification whereby the excitation signal is logged to files and compared against the fixed-point MATLAB model.

4.5 Summary

Previously unreported in prior art, this chapter has shown that it is possible to create a flexible, low-power, and low cost ABU transmitter suitable for embedded and mobile application scenarios. It is demonstrated that the proposed architecture can accommodate variability in system-level parameters. Later, in due time, further linkage will be made in the context of the transmitter signalling and its influence on Doppler-tolerant tracking. The trasmitter's signal fidelity and configurablity will in turn impact the receiver's signal processing and its accompanying architecture. Ultimately, a joint cost-benefit analysis across all layers has to be conducted in order to arrive at an acceptable performance and cost trade-off for the application at hand.

CHAPTER 5

Efficient Multiuser Despreader

This chapter describes a novel and efficient core (or kernel) that makes acoustic simultaneous multiple access possible. Section 5.1 discusses the relevance of the kernel. Related work is reviewed in section 5.2. Section 5.3 exposes the peculiarities in ABU that call for a departure from mainstream RF designs. Section 5.4 sets the scene by quickly reviewing the background surrounding the utility of the core. Section 5.5 describes the proposed despreader architecture. Implementation and ranging evaluation then follow in sections 5.6 & 5.7 respectively. Section 5.8 comments briefly on the utilized design flow. Finally, a conclusion is drawn in section 5.9.

5.1 Overview

An efficient core that makes acoustic simultaneous multiple access possible is proposed. To this end, architectures from the spread spectrum and radar literature are adapted to create an efficient and accurate despreader for direct sequence spread spectrum (DSSS), code division multiple access (CDMA) broadband ultrasonic signals. The use of the well-known finger-based RAKE architecture is demonstrated in conjunction with efficient batch Gold code generation in order to realize an acoustic multiuser despreader which is amenable to the portable, low cost and low power realization required in sensor networks and ad hoc indoor positioning systems.

Specifically, the proposed CDMA despreader core is designed from the ground up with lower-rate acoustic signals (centre frequencies and bandwidths in the tens of kilohertz) in mind to allow for increased time-multiplexing (i.e. resource sharing). Thus fingers in the acoustic despreader architecture detect multiple users (rather than multiple signal paths) in an interleaved fashion in order to maximize resource sharing. Consequently,

the number of co-located users that can be accommodated. In addition, the despreading codes for all users can be generated economically and on-the-fly. Finally, to validate the architecture, the design has been implemented in reconfigurable fabric (an FPGA). The accuracy of the kernel when utilized for ranging, using a data set gathered from a deployment of broadband ultrasonic transmitters and receivers, is reported.

5.2 Related Work

Since the early implementations of broadband ultrasound for indoor localization [63], few systems have appeared in the literature describing results in the same band of operation or further exploring signalling aspects. Work by Prieto et al. evaluates the ranging performance of CDMA acoustic signals in both still air and moving air conditions [107]. They report sub-centimetre error using relatively short (32 bit) Golay codes. However, they utilized more powerful (and bulky) transducers operating in the primarily audible range of 5–25 kHz. This would have allowed for more favourable signal-to-noise ratios than are possible with compact piezofilms operating in the ultrasonic range (above 30 kHz), and it is unclear how much this SNR improvement contributed to their accuracy.

Another system described by Ureña et al. [138] replicates previous signalling [61], except that it substitutes Kasami codes for Gold ones. The system relies on the differential time-of-arrival of signals from five transmitters to estimate one receiver's location. Sporadic measurements are shown but no serious characterization of accuracy within the test volume is conducted. The paper also discusses issues of embedded design, considering sequential and parallel implementations of the Kasami code correlator. The authors report the implementation of single-user, parallel correlator with eight-bit resolution which utilized 2,328 slices on a Spartan-IIE architecture. How the receiver embedded processing logic scales as the number of transmitters increases is not discussed.

Álvarez et al. recommend a limit of about 10 ms of Doppler coherence time to govern code length, in order to specifically address the air turbulence effects they observed outdoors [21]. For indoor environments turbulence is less aggressive, yet still present due to air conditioning units and building draughts. In the proposed solution for tracking in the presence of Doppler, chapter 6 advocates that equalization per-chip mitigates against the symbol coherence time problem. The advantage of this method is that there are no hard limits on code length, transmission power, or transducer efficiency. More flexibility in these parameters allows the design of devices with small form factor and portability—crucial for small, battery-powered sensor nodes and wearable tags.

5.3 Implications of the ABU modality on system design

There are two major differences between the proposed ultrasonic DSSS detector, and those used for RF:

Multiuser fingers

Firstly, conventional RF RAKE receivers are at most used for limited multi-code reception when few parallel codes (two to three) are needed for increased data rate [66]. This is because they are concerned with combating multipath through diversity in order to yield improved channel capacity [81]. In the proposed broadband ultrasound multi-user receiver, fingers are aimed at parallel multiuser detection. This is because the acoustic spread spectrum system is more radar-like, wherein fingers are assigned different user codes as opposed to offsets of the same code.

Chip-rate correlations

Secondly, fingers in RF RAKE receivers operate on symbols and are clocked at the chip rate (after timing has been acquired), resulting in code-rate correlation. RF code acquisition is dealt with through a vast body of literature wherein detectors oftentimes conduct multidimensional searches and employ sophisticated techniques (e.g. active variable sequential dwell time) in order to render the multiuser problem tractable. In contrast to the DSSS CDMA signals in GPS and radio communications, a user's broadband ultrasound ranging signal consists of a single PN sequence, and transmitter/receiver synchronization cannot be assumed. Thus, the multiuser correlation must be carried out at the chip rate.

5.4 Background

This section illustrates the usage scenarios of the proposed kernel. Specifically, it will be shown how the design can be leveraged as a building block (kernel) for three ABU functionalities, namely: ranging, beamforming, and Doppler-tolerant tracking.

5.4.1 Ranging

The optimum DSSS TOA estimator has been reported in spread spectrum literature [67, sect. 6.9.2], in the context of noncoherent, passive acquisition detectors. The operation is performed on signals whose frequency and pseudorandom code are known. In the BPSK case, the single user signal is given by

$$s^u(t) = \sqrt{2\mathcal{P}} \times PN^u(t-T)\cos[\omega_0(t-T) + \theta^u(t-T) + \varphi^u] \tag{5.1}$$

where ^{u} is a superscript denoting a user, $\mathcal{P}$ is the power of the shaped pulse, PN is the pseudorandom sequence of chips, T is the duration of a chip, ω_0 is the carrier frequency, θ^u is the chip-dependent phase modulation function, and φ^u is the unknown phase in the received signal and is assumed to remain constant over the processing time (i.e. the transmitter node is stationary relative to the receiver node).[1] At the receiver, one or more

[1] Chapter 6 deals with the mobile case where this assumption does not hold.

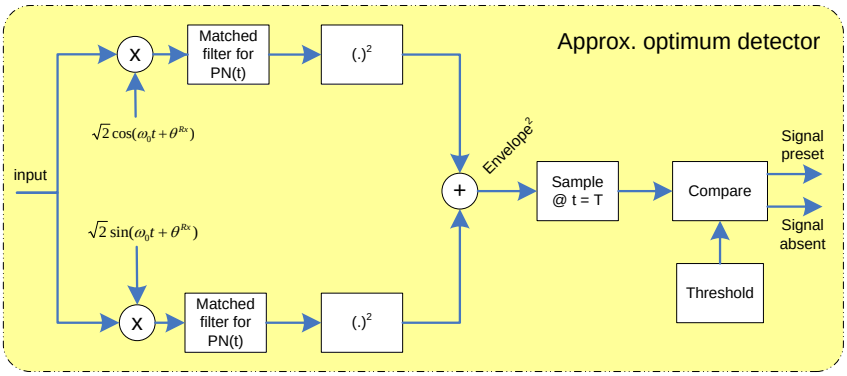


Figure 5.1: Approx. optimum detector for TOA estimation

user signals can arrive concurrently, and can be expressed as

$$y(t) = \sum_{u=1}^{U} s^u(t - \tau_u) + n(t) \tag{5.2}$$

where $n(t)$ is the AWGN of the receiver, U is the number of users present, and τ_u is a user's position-dependent time offset.

This signal is heterodyned down to its in-phase and quadrature components at baseband. Matched filtering is then carried out on both components and the envelope is extracted.[2] It can be shown [67] that the optimum detector requires the knowledge of two parameters k and $I_0()$ where $k = 2/(N_0\sqrt{R})$, $R = E_T/N_0$, E_T is the energy (Watts-seconds) in the code burst, N_0 is the one-sided noise spectral density (Watts/Hz), and $I_0(x)$ is the modified Bessel function of the first kind and zeroth order.

The optimum CDMA detector is therefore difficult to implement (especially as a small, handheld device or wearable tag) owing to the need to estimate the signal energy, noise level, and the function $I_0(x)$ of the sampler on-the-fly. Observing the monotonic nature of the signal's envelope, the above optimum structure can be approximated [43] to create a feasible multiuser realization. Neither the signal-to-noise ratio nor $I_0(x)$ is required in this case. The approximate detector is shown in figure 5.1.

Thus multiuser ranging calls for the ability to correlate against all possible user codes in parallel and as economically as possible.

5.4.2 Beamforming

Another usage scenario for the proposed kernel is *broadband* beamforming. This can be easily conceptualized by inspecting the equation of the classic broadband FIR beamformer [139, 58]

$$y(k) = \sum_{p=1}^{P} \sum_{m=0}^{M-1} w_{p,m}^{*}\, x_p(k - m) \tag{5.3}$$

[2] Filtering here is matched to the PN sequence only.

where P is the number of sensors, M is the number of delays in each of the sensor channels, and $w_{p,m}$ is the complex coefficient specific to a sensor-delay pair. Further, in the multiuser case and similar to the ranging application, x_p is given by

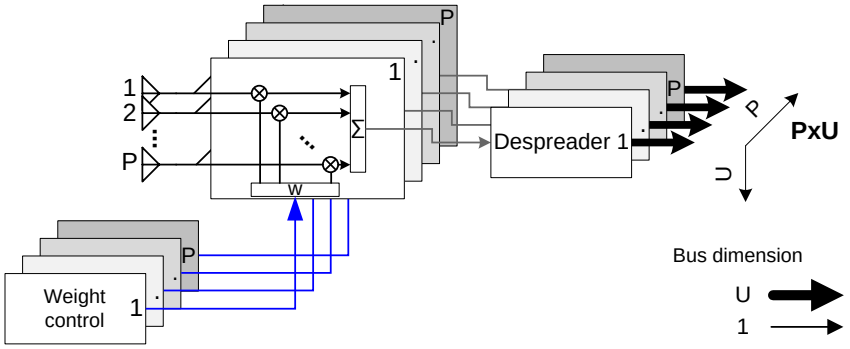
$$x_p(t) = \sum_{u=1}^{U} x_p^u(t - \tau_u) + n(t) \tag{5.4}$$

with

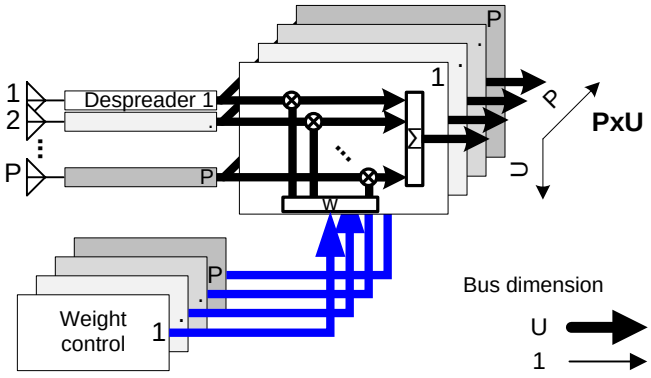
$$x_p^u(k) = \sum_{n=0}^{L-1} q^u(n)\ g(t - nT_c) \tag{5.5}$$

where q^u is the spreading code of user ^{u}, L is the coding gain, T_c is the chip duration, and g is the the chip pulse shaping waveform.

Observing the commutativity of the linear summations in the above equations, specifically that of equations (5.3) & (5.5), two beamformer structures can be realized [136, 42] as shown in figure 5.2. It can be shown that the so-called symbol-based (SB) configuration has better performance properties over that of the chip-base (CB) configuration [141]. This, however, comes at the expense of increased hardware complexity.



(a) Chip-based



(b) Symbol-based

Figure 5.2: DS CDMA delay-and-sum beamforming. Symbol-based (b) tends to have better performance than chip-based (a), at increased complexity cost.

Both configurations illustrate that despeading is an integral part of beamforming.

Therefore, an efficient despreader kernel that can be instantiated in logic many times is required in a multiuser beamformer irrespective of the employed configuration.

5.4.3 Doppler tracking

The last usage for the proposed kernel is in the Doppler-tolerant acquisition and tracking for tags on the move, as will be developed in chapter 6. Specifically, two steps during adaptation would make use of the kernel: (1) oversampled at times in equation (6.25), and (2) as a vector dot product with no oversampling in equation (6.26). The reader is referred to chapter 6 for the full derivation. With some specialization, both equations can be implemented utilizing the architecture of the proposed kernel.

5.5 Proposed Architecture

The proposed core consists of the following units: a complex circular buffer, user code generation unit, a matched-filter bank, and a control state machine.[3] The design is parameterizable to account for various system variables: system clock, chip rate, code length, oversampling rate, number of scan codes, data word width, and number of guard bits. A functional diagram of the core is presented in figure 5.3, and the significant components are described in detail below.

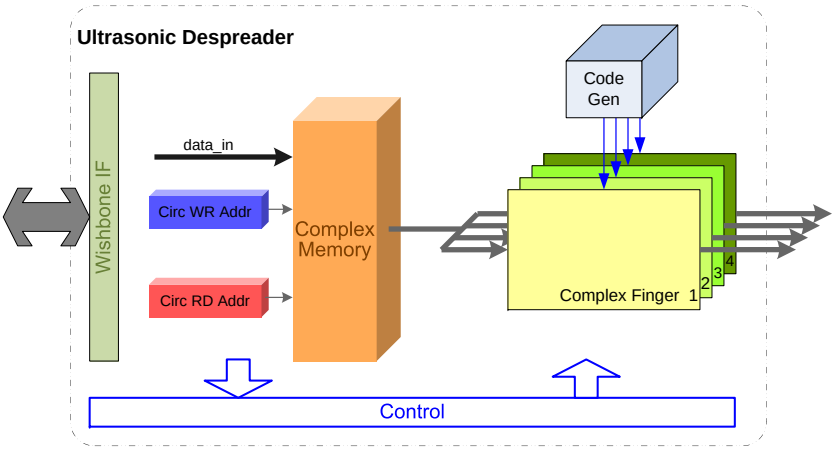


Figure 5.3: Multiuser despreader core for ultrasonic ranging

Complex memory

The complex memory is a circular buffer for storing the complex baseband signal. It is addressed using two counters for write and read operations. In order to allow for on-the-fly batch code generation instead of storing the entire bit sequences for all users, a reversed order memory access is utilized observing the commutative property of convolution. The

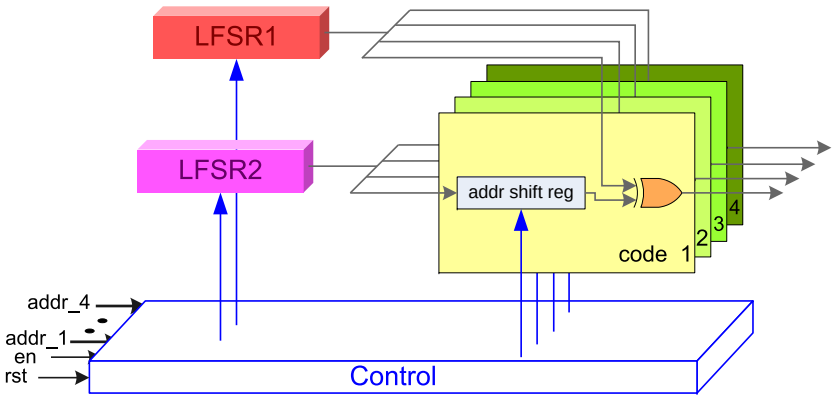
[3] Here, the word complex refers to resource duplication necessary for the real and imaginary parts of the baseband signal.

depth of the buffer is the product of two system parameters: the length of the spreading code L and the chip oversampling rate N_s.

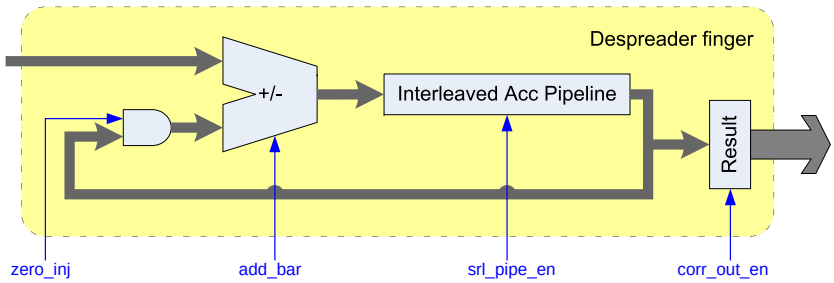
Matched-filter bank

The matched-filter bank has a number of complex fingers. As discussed in section 5.3, the complex fingers are aimed at parallel multiuser reception as opposed to multipath. Each finger is assigned a number of codes (interleaved) which is a function of system clock, chip rate, code length, and oversampling rate according to the formulas: $O = f_{\text{clk}}/f_{\text{chip}}$, $U = \text{floor}(O/(L \times N_s))$, where O is the number of operations that can be accommodated before the arrival of next chip, f_{clk} is the digital system clock, and f_{chip} is the chip rate, U is the number of users that can be correlated against in a complex finger. Thus one has to use as many complex fingers as necessary to scan for a certain number of users, for a given system clock rate and PN code length.

The complex finger is at the essence of the despreader. In addition to N_s and U, two other generics are passed to the block: G_{bits} and D_{wdth}. These are used to determine the accumulation pipeline width and defined as follows: G_{bits} is the width of the integer portion of the accumulator (determined by expected SNR), and D_{wdth} is the fixed point precision of the incoming fractional samples.



(a) Code generator



(b) Complex despreader finger

Figure 5.4: Detail of core blocks

The operation of the finger can be best understood by examining figure 5.4b. Samples are read out firstly from memory and either added to or subtracted from the accumulator subject to the boolean code. The accumulation pipleline stores the users' partial integra-

tions as interleaved groups of oversampled chips. The control unit determines whether to feed back the accumulations by way of an inject zero signal that acts effectively as enable. Once the correlation is complete, results are passed out through enabling a capture register.

User code generation unit

As mentioned earlier, it is more efficient to generate the Gold code on-the-fly and step it through the operation, rather than storing the long sequences in memory. This is particularly important when a node scans for say thirty-two users in parallel. The spreading code used is a 511-bit Gold code [44]. It is generated by XORing two ninth-order maximal length sequences. These in turn can be obtained by specifying the feedback terms of a linear feedback shift register (LFSR). By shifting one LFSR output with respect to the other, another orthogonal code is obtained.

The current implementation utilizes an extremely efficient shift register macro whose output is addressable. That is, one can delay the second Gold code LFSR though the shift register and be able select the appropriate code by varying the address. This is quite scalable and results in extremely compact realization. Nevertheless, for the sake of flexibility, each parallel code output within an octal Gold batch can access eight shifts irrespective of other outputs. This was designed bearing in mind that the current fabric used for prototyping has almost as many flip-flops as addressable shift register macros. Once the unit is initialized, the code is stepped forward by means of an enable signal while constantly having a moving window of eight shifts as to enable concurrent code accesses. For example, in an implementation requiring eight codes and two interleaved users per finger, four parallel shift register macros are initially addressed by even pointers whose least significant bits are toggled by control in order to access the remaining four odd codes. The code unit is fully illustrated in figure 5.4a.

De-interleaver logic

It is the responsibility of the external logic to de-interleave integrations to properly reconstruct the multicode filter output subject to application considerations. The unit consists of a shorter delayline akin to the accumulation pipeline with the exception that it operates only on the part of the integration likely to contain correlation spikes as set by the detection threshold (SNR).

5.6 Implementation

The despreader core is intended to be embedded in an ultrasound-capable sensor node, mobile device, or wearable tag. In the ranging mode of operation, the core will be activated only upon an RF trigger for a search duration corresponding to the maximum expected TOA. Due to this duty cycling and the core's compact logic requirements, one

can envision a realization on the highly power efficient, low-cost FLASH reconfigurable fabrics, or even SRAM devices with hibernation capabilities.

To demonstrate the design, an eight-user ultrasonic despreader was implemented on a Xilinx Virtex-4 XC4FX12 device using RTL description. This was carried out using Xilinx's 10.3 ISE tool suite. The core operates on a 100 MHz system clock with data quantised to 16.15 signed fractional representation, and 8 guard bits. Signalling parameters utilised are a 20 kHz chipping rate, 511 bit Gold code (which leads to about 8.7 m maximum node separation in *Doppler-tolerant* mode), and four-times chip oversampling rate. Under these parameters, four fingers are required, since a maximum of two interleaved users can be accommodated for in a single complex finger. Table 5.1 shows the final place-and-route metrics of the four-finger, eight-user core.

Table 5.1: FPGA resource requirements[a]

Resource	Used	Utilization %
Slices	632	11
LUTs	699	6
BRAMs	4	11
Max frequency	103.67 MHz	

[a] Device: Xilinx Virtex-4 XC4FX12

Two comments on implementation efficiency are worth making at this point. First, a long carry chain is implemented, allowing a relatively low clock rate. This is done owing to the desire to have lower power consumption for the mobile device. Alternatively, one might prefer to choose higher clock rates in order to conserve resources by packing more interleaved users, and utilise pipelining to break the carry-dominated critical path. Pipelining would be required, for example, in the recent low-power, but sparser FLASH-based FPGAs. Second, CDMA systems effectively share power. When received signal powers are near equal, the *soft capacity* is maximised, allowing improved maximum range, detection rate, and number of simultaneous users. In location systems where acoustic power control of multiple transmitter nodes is feasible, further optimisation can be achieved in the de-interleaver and capture register. The concept of fusing thresholding and filtering loops, and the resource conservation merits of doing so have been previously discussed [77].

5.7 Empirical Ranging Evaluation

The despreader core was co-simulated on the Xilinx XC4FX12 hardware, using captured sensor data. This verifies the core's functionality and allows for the evaluating of the accuracy of real-time ranging realized on top of the kernel. Data utilized was gathered using a broadband ultrasonic localization system [61, sect. 5.1]. The experimental conditions of the measurement setup are summarized here. Gold codes of length 511 were used to BPSK modulate a 50 kHz carrier, at a chip rate of 20 kHz. The signal from each re-

ceiver's analogue stage was sampled at just over 200 kHz, with twelve bits of resolution per sample.

A primary transmitter node was used to emit one hundred ranging messages, across sixteen horizontal positions (shown as diamonds in Figure 5.5), at each of three different heights (10, 50 and 100 cm above the floor). To emulate the presence of co-located users, four other transmitter nodes (triangles in Figure 5.5), emitted their ranging messages simultaneous to the node placed at the test positions. All transmitters emit ranging signals at the same acoustic power. Four receiver nodes (stars in Figure 5.5) were mounted on the ceiling, about 220 cm above the floor.

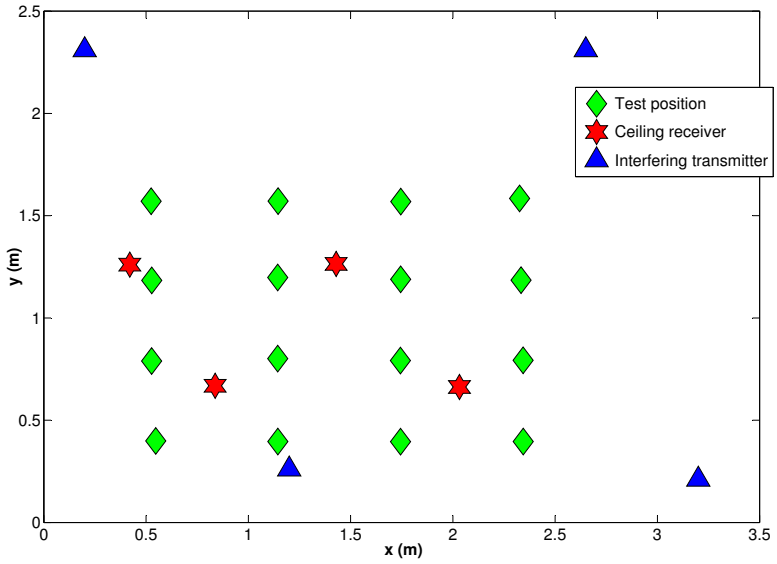


Figure 5.5: Top view of deployment area

Figure 5.6 illustrates the multiuser performance of the co-simulated dataset. For each of the four ceiling receivers, the one hundred ranging events were processed at the forty-eight transmitter locations—a total of 19,200 measurements. The core's overall detection rate for ranging signals from the primary transmitter node is 65.4%.[4] The undetected ranging signal occurrences are commensurate with two effects previously observed with this deployment: (1) the directivity of the transducers [61, figs. 7 & 11]; and (2) transmitter near-far effects [61, sec. 4.3]. When the despreader core successfully detects events, its ranging error is better than three centimetres (2.92 cm) at the ninety-fifth percentile confidence level.

5.8 Comments on methodology

In this chapter, the hardware design methodology utilized three environments: MATLAB, System Generator, and VHDL. The despreader core is designed in RTL. Functional-level simulations are performed in MATLAB. DSP-oriented software simulations are carried

[4] Ranging success rates higher than the reported 65% would be easily achievable using dynamic thresholding and enhanced peak detection methods [60, p. 94,109, & 152].

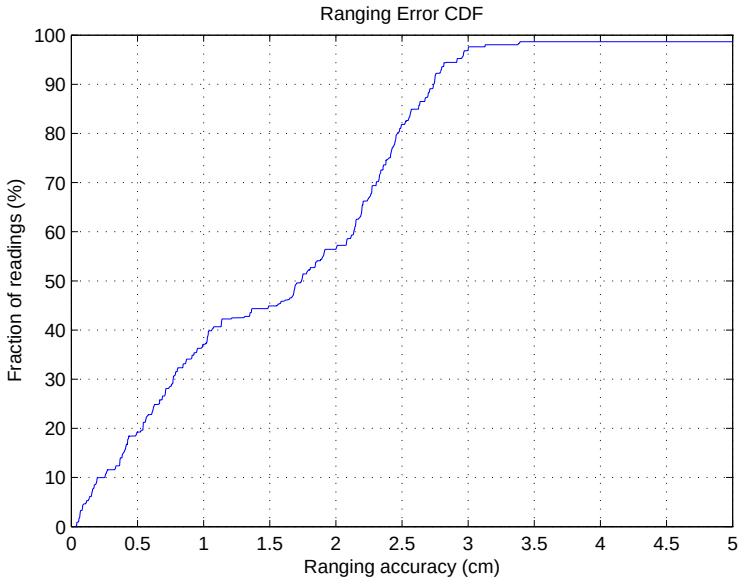


Figure 5.6: Cumulative distribution of ranging errors

out in System Generator. Simulink's Stateflow is used for: hardware handshake signals, feeding stimuli to the despeader, and handing over results back to MATLAB. After initial software simulations, the design is then co-simulated for the generation of the ranging CDF reported in section 5.7. This is achieved as follows. First, the despreader is wrapped around input and output buffering stages with control logic. The buffering stages are implemented using shared memories which can be accessed by software in MATLAB. Second, shared registers are utilized in order to implement memory access mutual exclusion (MUTEX) between hardware on the FPGA and software in MATLAB. A simple 4-phase asynchronous protocol is employed for MUTEX whereby each respective side owns a shared register, and polls on the other shared register before it can toggle the value of its register. Third, a MATLAB script realizes the final hardware-in-the-loop co-simulation. Some details of the co-simulation system can be found in appendix D.

5.9 Conclusion

An efficient DS CDMA core capable of multiuser acoustic detection has been detailed and characterized; it employs a finger-based design, where each complex finger scans for two interleaved user spreading codes. The core makes it feasible for devices with minimal computation and power resources (such as those found in mobile computing and sensor networks) to take advantage of the noise robustness, increased range, and higher update rates afforded by spread spectrum, simultaneous multiple access acoustic ranging. The performance of such real-time broadband ranging was assessed qualitatively on a dataset gathered from a real deployment and was shown to be favourable. With its efficiency and low resource requirements, the core can be used as a despreading kernel to realize higher-level functionalities such as adaptive acquisition of Doppler-shifted signals (chapter 6) and multiuser beamforming.

PART II

COHERENT AIRBORNE BROADBAND ULTRASOUND

CHAPTER 6

Adaptive ABU Detection

6.1 Introduction

This chapter develops novel coherent acoustic tracking for the ABU modality. The aim is to investigate the viability of the technology for motion tracking in indoor environments. Previously uncharted, the principles governing the operation of ABU motion tracking is thoroughly studied and characterized.

The receiver is first derived in section 6.2. It draws on advances in coherent spread spectrum underwater wireless communication, most notably that of Stojanovic et al. for a single user [131] and for the multiuser case [132]. Building on earlier intuition [129], the chapter then departs to introduce a *modified* implicit decision-feedback equalizer (DFE), intended as a bespoke equalizer for airborne broadband ultrasound. This structure is then further expanded and equipped with linear interpolation capability, in similar vein to other coherent underwater systems [121, 120, 122]. It is worth noting that the final algorithm is computationally tractable, and as such, is highly amenable to embedded realization as will be shown in chapter 8.

The second part of the chapter reports on the principles governing the operation of phase tracking in the ABU band. In section 6.3, this presentation proceeds in a step-by-step manner to build hybrid Doppler trackers. This is important owing to the absence of prior experience on phase tracking in the novel ABU band, hence the need to consider all possible methods and their combinations. Section 6.4 gives a brief overview of the experimental setup and data analysis methodology. Following this, section 6.5 empirically characterizes ABU phase tracking, applying the presented methods on concrete data captured using ABU sensors. Finally, a summary is drawn in section 6.6.

6.2 Receiver Derivation

This section derives a comprehensive Doppler tracker for ABU signals. Starting with the DSSS model, various components are progressively introduced and incorporated until the final receiver structure is arrived at.

6.2.1 DSSS System Model

The complex baseband signal at the receiver is modelled as

$$v(t) = \sum_m q(m)h(t - mT_c)\,.\,e^{j\theta(t)} + w(t) \tag{6.1}$$

where q is the spreading code, $h(t)$ is the channel impulse response which accounts for transmitter and receiver filtering, T_c is the chip duration, $\theta(t)$ is a phase function modeling the aggregate offset and/or distortion resulting from transmitter-receiver oscillator mismatch and/or mobility respectively, and $w(t)$ is an additive noise which is assumed to be uncorrelated with the signal i.e. $\mathbf{E}\{q(t)w(t) = 0\}$.

At the receiver, the signal is sampled at the chip oversampling rate N_s and arranged in a column vector spanning the spreading code length L and whose topmost entry is the most recent input sample. The received signal vector expressed as the spreading code convolved with the channel can now be rewritten in vector format as

$$\mathbf{v}(k) = \sum_m \mathbf{h}[m]q[k - m] \cdot e^{j\boldsymbol{\theta}(k)} + \mathbf{w}(k) \tag{6.2}$$

where

$$\mathbf{v}(k) = \begin{bmatrix} \underline{\mathbf{v}}_{L-1}(k) \\ \vdots \\ \underline{\mathbf{v}}_0(k) \end{bmatrix} \tag{6.3}$$

is an $LN_s \times 1$ vector whose entries $\underline{\mathbf{v}}_l(k)$ are partial input vectors containing the l^{th} chip-worth of samples of the spreading code (depending on the chip oversampling rate N_s).

In similar vein

$$\mathbf{h}(k) = \begin{bmatrix} \underline{\mathbf{h}}_{L-1}(k) \\ \vdots \\ \underline{\mathbf{h}}_0(k) \end{bmatrix} = \mathbf{h}[0] \tag{6.4}$$

Further, noting the causality of the system, $\mathbf{h}[m]$ in the convolution kernel is defined as $\mathbf{h}(k)$ shifted down by $m \times N_s$ samples; otherwise, $\mathbf{h}(k)$ remains to denote the channel at time kT_c.

$$\mathbf{h}[m] = \begin{bmatrix} \left.\begin{matrix} \underline{\mathbf{0}} \\ \vdots \end{matrix}\right\} m \\ \underline{\mathbf{h}}_{L-1}(k) \\ \vdots \\ \underline{\mathbf{h}}_{m}(k) \end{bmatrix} \tag{6.5}$$

where $\underline{\mathbf{0}}$ is N_s zeros.

Similarly, $q[\,\cdot\,]$ denotes modulo-L addressing of the spreading sequence. Also,

$$\boldsymbol{\theta}(k) = \begin{bmatrix} \theta(k+L-1) \\ \vdots \\ \theta(k) \end{bmatrix} \tag{6.6}$$

is a vector contains L-chip phase deviations that are multiplied element-wise with their respective L $\underline{\mathbf{v}}_l(k)$ as denoted with the dot operator $\cdot$ in the equation where we assumed that e raised to a vector is a vector of e's raised to the individual entries. Lastly, $\mathbf{w}(k)$ is an $LN_s \times 1$ vector of additive noise.

Alternative system interpretation

For the sake of visualizing adaptation in later sections, a more insightful view of the channel can be given as

$$\mathbf{H}(k) = \Big[\mathbf{h}[0] \;\; \mathbf{h}[1] \ldots \mathbf{h}[L-2] \;\; \mathbf{h}[L-1]\Big] \tag{6.7}$$

where $\mathbf{H}(k)$ is an $LNs \times L$ matrix of L channel shifts. Similarly, a sliding window of the speading code at any given point in time kT_c can be represented by the $L \times 1$ vector

$$\mathbf{q}(k) = \begin{bmatrix} q[k] \\ \vdots \\ q[k-L+1] \end{bmatrix} \tag{6.8}$$

Then, the system can be alternatively represented as

$$\mathbf{v}(k) = \mathbf{H}(k)\mathbf{q}(k) \cdot e^{j\boldsymbol{\theta}(k)} + \mathbf{w}(k) \tag{6.9}$$

6.2.2 Core DS CDMA Adaptive Engine

This subsection along with subsection 6.2.4 follow closely the derivation of Stajanovic et al. in [131]. Later subsections will depart—building in places on the intuition from [129] and [121]—to introduce the feedforward filter and linear interpolator stages developed for airborne broadband ultrasound.

Channel-based formulation

The acquisition algorithm begins with using the fact that noise is uncorrelated with the spreading sequence

$$\mathbf{h}(k) = \mathbf{E}\Big\{\mathbf{v}(k) \cdot e^{-j\boldsymbol{\theta}(k)} q^*(k)\Big\} \tag{6.10}$$

leading to a simple stochastic approximation of the form:

$$\hat{\mathbf{h}}(k) = (1 - \lambda_{ch}) \sum_{m=0}^{k} \lambda_{ch}^{k-m} \mathbf{v}(m) \cdot e^{-j\boldsymbol{\theta}(m)} q^*(m) \tag{6.11}$$

where λ_{ch} is an exponential forgetting factor. The expression is written recursively as:

$$\hat{\mathbf{h}}(k) = \lambda_{ch} \hat{\mathbf{h}}(k-1) + (1 - \lambda_{ch}) \mathbf{v}(k) \cdot e^{-j\boldsymbol{\theta}(k)} q^*(k) \tag{6.12}$$

For the i.i.d. gold code sequences, optimal tap selection through truncation in magnitude is readily achieved [129].

Recursive interference-free signal reconstruction

The channel vector is fractional. Under inter-chip-interference (ICI)—be it caused by the time-varying Doppler distortion w.r.t chips within an entire code, or due to multipath—equation (6.2) can be expanded as:

$$\mathbf{v}(k) = \left[\mathbf{h}[0]q[k] + \sum_{m \neq 0} \mathbf{h}[m]q[k-m]\right] \cdot e^{j\boldsymbol{\theta}(k)} + \mathbf{w}(k) \tag{6.13}$$

where $\mathbf{h}(k) = \mathbf{h}[0]$ the channel vector of shift zero at time kT_c.

The ICI signal is defined as

$$\mathbf{v}_b(k) \triangleq \sum_{m \neq 0} \mathbf{h}[m]q[k-m] \cdot e^{j\boldsymbol{\theta}(k)} \tag{6.14}$$

and the ICI-free signal as

$$\begin{aligned} \mathbf{v}_f(k) &\triangleq \mathbf{h}[0]q[k] \cdot e^{j\boldsymbol{\theta}(k)} + \mathbf{w}(k) \\ &= \mathbf{v}_0(k) \end{aligned} \tag{6.15}$$

and the signal-plus-ICI signal

$$\begin{aligned} \bar{\mathbf{v}}(k) &\triangleq \sum_{m=0}^{L-1} \mathbf{h}[m]q[k-m] \cdot e^{j\boldsymbol{\theta}(k)} \\ &= \mathbf{v}_b(k) + \mathbf{h}[0]q[k] \cdot e^{j\boldsymbol{\theta}(k)} \end{aligned} \tag{6.16}$$

Adding and subtracting a $\mathbf{h}[0]q[k] \cdot e^{j\boldsymbol{\theta}(k)}$ term to the RHS of equation (6.13) gives:

$$\begin{aligned} \mathbf{v}(k) &= \mathbf{v}_f(k) + \mathbf{v}_b(k) \\ &= \mathbf{v}_f(k) + \bar{\mathbf{v}}(k) - \mathbf{h}(k)q[k] \cdot e^{j\boldsymbol{\theta}(k)} \end{aligned} \tag{6.17}$$

In addition, $\bar{\mathbf{v}}(k)$ obeys a shifting law such that

$$\bar{\mathbf{v}}(k) = \begin{bmatrix} \underline{\bar{\mathbf{v}}}_{L-1}(k) \\ \downarrow^{N_s} \quad \bar{\mathbf{v}}(k-1) \end{bmatrix} \tag{6.18}$$

where

$$\begin{aligned} \underline{\bar{\mathbf{v}}}_{L-1}(k) &= \sum_{m=0}^{L-1} \underline{\mathbf{h}}_{L-1-m}(k)q[k-1-m] \\ &= \sum_{m=0}^{L-1} q[k+m]\underline{\mathbf{h}}_m(k) \end{aligned} \tag{6.19}$$

and $\downarrow^{N_s} \bar{\mathbf{v}}(k-1)$ symbolizes down-shifting previous signal-plus-ICI by N_s samples.

Replacing the channel vector with its estimate in everything we derived so far, we arrive at a channel adaptation procedure that can be performed in two coupled steps. Firstly, equation (6.17) is rearranged in an expression as to allow the *reconstruction* of an estimate of the ICI-free signal:

$$\hat{\mathbf{v}}_f(k) = \mathbf{v}(k) - \hat{\bar{\mathbf{v}}}(k) + \hat{\mathbf{h}}(k)q[k] \cdot e^{j\hat{\boldsymbol{\theta}}(k)} \tag{6.20}$$

Secondly, using the ICI-free signal in (6.12) for better channel estimation yields

$$\hat{\mathbf{h}}(k) = \lambda_{ch}\hat{\mathbf{h}}(k-1) + (1-\lambda_{ch})\hat{\mathbf{v}}_f(k)q^*[k] \tag{6.21}$$

Once the channel estimate and the ICI-free signal have been computed, the chip estimate is defined by the filtering operation

$$\hat{q}(k) = \frac{1}{E_{\hat{\mathbf{h}}}(k)}\hat{\mathbf{h}}'(k)\hat{\mathbf{v}}_f(k) \tag{6.22}$$

where

$$E_{\hat{\mathbf{h}}}(k) = \hat{\mathbf{h}}'(k)\hat{\mathbf{h}}(k) \tag{6.23}$$

Note that during situations of rapid channel strength deterioration—e.g. resulting from aggressive Doppler levels—an algorithmic check is applied to gauge whether the channel energy drops below a predefined threshold. At such a condition, the reciprocal of the channel energy is assigned a unity value. This is done in order to prevent ill-conditioned divisions in equation (6.22). It was also observed that these techniques could help the algorithm recover without total loss of code timing. The empirical characterization presented later in the chapter will illustrate this point further.

This model allows us to monitor the state of self-interference explicitly. It allows the

algorithm to pass *multipath* information to better constrain and inform the localization problem. It can be shown [132] that this formulation can be expanded in the multiuser case to account for multiple access interference (MAI), but at the expense of increased computation complexity and memory requirements. Specifically, with a total of U users present in the system, the recursive reconstruction of the average signal vector becomes defined as

$$\underline{\bar{\mathbf{v}}}_{L-1}(k) = \sum_{u=1}^{U} \underline{\bar{\mathbf{v}}}^{u}_{L-1}(k) \tag{6.24}$$

where

$$\underline{\bar{\mathbf{v}}}^{u}_{L-1}(k) = \sum_{m=0}^{L-1} q^{u}[k+m]\underline{\mathbf{h}}^{u}_{m}(k) \tag{6.25}$$

with the superscript ^{u} denoting a user's reconstructed signal, spreading code, and channel vector.

Acquisition test

Monitoring the MSE of a running despreader provides a measure of signal-to-noise-plus-interference ratio (SNIR) at the output:

$$\begin{aligned}
\hat{d}(k) &= \mathbf{E}\{\hat{q}(k)q^{*}(k)\} \\
&= \frac{1}{L}\sum_{m=k-L+1}^{k} \hat{q}(m)q^{*}(m) && (6.26)\\
e_{\hat{d}}(k) &= 1 - \hat{d}(k) && (6.27)\\
SNIR_{out} &\sim \mathbf{E}\,|e_{\hat{d}}(k)|^{2} \\
&= -10\log_{10}\left(\frac{1}{N_d}\sum_{m=k-N_d+1}^{k} |e_{\hat{d}}(m)|^{2}\right) && (6.28)
\end{aligned}$$

where N_d is the length of despreader observation.

Therefore, acquisition can be declared when $SNIR_{out}$ stabilizes, in accordance with what is expected for the indoor acoustic environment. Alternatively, once the MSE drops below a pre-specified threshold, this can be used as an indication for successful acquisition.

6.2.3 Implicit DFE Extension

Applying a decision feedback equalizer (DFE) to $\mathbf{v}(k)$ at the chip-rate, as if no spreading is taking place, a chip estimate can be expressed as

$$\hat{q}(k) = \mathbf{a}'\mathbf{v}(k) - \mathbf{b}'\mathbf{q}(k-1) \tag{6.29}$$

where **a** and **b** are the feedforward and feedback coefficient vectors respectively, and $\mathbf{q}(k-1)$ is the vector of the previously searched chips (during acquisition) whose length is in accordance with distortion span in chips M.

$$\mathbf{q}(k-1) = \begin{bmatrix} q[k-1] \\ \vdots \\ q[k-M] \end{bmatrix} \tag{6.30}$$

If we plug our signal model (6.2) into (6.29) and forget about the noise for now, we obtain

$$\hat{q}(k) = \left[\mathbf{a}'\hat{\mathbf{h}}[0]q[k] + \mathbf{a}' \sum_{m>0} \hat{\mathbf{h}}[m]q[k-m] \right] \cdot e^{j\hat{\boldsymbol{\theta}}(k)} - \mathbf{b}'\mathbf{q}(k-1) \tag{6.31}$$

Inspecting equation (6.31), it is therefore required that

$$\mathbf{b}' = \mathbf{a}' \sum_{m>0} \hat{\mathbf{h}}[m] \cdot e^{j\hat{\boldsymbol{\theta}}(k)} \tag{6.32}$$

so that the ICI contribution to the chip estimate be eliminated. In other words, the task of backward filtering is readily taking place through shifts of the channel estimate, where we assumed that the channel is zero outside the *multipath span*, i.e.

$$\hat{h}_n(k) = 0, \;\; n > M \tag{6.33}$$

Equally, we can plug (6.17) into (6.29)

$$\begin{aligned} \hat{q}(k) &= \mathbf{a}'\hat{\mathbf{v}}_f(k) + \mathbf{a}'\hat{\mathbf{v}}_b(k) - \mathbf{b}'\mathbf{q}(k-1) \\ &= \mathbf{a}'\hat{\mathbf{v}}_f(k) + \mathbf{a}'\hat{\bar{\mathbf{v}}}(k) - \mathbf{a}'\hat{\mathbf{h}}(k)q[k] \cdot e^{j\boldsymbol{\theta}(k)} - \mathbf{b}'\mathbf{q}(k-1) \end{aligned} \tag{6.34}$$

Comparing (6.34), (6.31), and (6.20), one notices that it is possible to further filter the incoming signal $\mathbf{v}(k)$ with a linear adaptive equalizer $\mathbf{a}'(k)$ *independent of* $\hat{\mathbf{h}}(k)$ in order to obtain an improved chip estimate at time kT_c. The clean chip estimate is defined by the term containing the reconstructed ICI-free feedforward signal—which is *dependent upon* the goodness of the channel estimate

$$\hat{q}(k) = \mathbf{a}'(k)\hat{\mathbf{v}}_f(k) \tag{6.35}$$

The combined operation effectively realizes an *implicit* DFE wherein the separation of the incoming signal into forward and backward components is denoted by the subscripts $_f$ and $_b$ respectively.

Equating (6.35) and (6.22) is interesting:

$$\mathbf{a}'(k) = \frac{1}{E_{\hat{\mathbf{h}}}(k)}\hat{\mathbf{h}}'(k) \tag{6.36}$$

In the absence of *multipath*, the sole role of $\mathbf{a}'(k)$ would be to combat *phase* distortion in the fractional chip fed into the spread spectrum algorithm. Equation (6.36) tells us that if the channel alone fails at equalizing the chip estimate perfectly, the feedforward filter of the implicit DFE would help in accomplishing the task by eliminating residual distortion. This is especially the case for a highly-Doppler-susceptible, non-stationary indoor acoustic environment.

To this end and with the intuition of equation (6.36) in mind, the *modified* fractional feedforward filter is made to operate on incoming chips at the oversampling rate N_s, although its adaptation remains at the chip rate such that

$$v(t) = \sum_{m}^{L_{ff}} a(mT_{c/.})s(t - mT_{c/.}) \tag{6.37}$$

where $T_{c/.} = T_c/N_s$, and $s(t)$ is instead the raw complex baseband signal at the receiver now. In vector format, we define

$$\underline{\mathbf{c}}(k) \triangleq \begin{bmatrix} c_{N_s}(k) \\ \vdots \\ c_1(k) \end{bmatrix} \tag{6.38}$$

to be an oversampled chip at time kT_c , and

$$\underline{\mathbf{v}}(k) \triangleq \left[\mathbf{a}'(k) \begin{bmatrix} c_{N_s}(k) & \cdots & c_1(k) \\ \vdots & \iddots & \downarrow^1 \mathbf{s}(k-1) \\ c_1(k) & \iddots & \\ \downarrow^{N_s} \mathbf{s}(k-1) & & \end{bmatrix}\right]^T \tag{6.39}$$

to be the modified fractional feedforward filtering at time kT_c. The length of the filter is designed to span a certain number of oversampled chips L_{ff}, and is devoted to combating phase distortion by means of matching the filter response to the incoming signal. Inevitably, it also introduces ICI which will be leaked away by the sparse channel-based backward filtering and futher combated within the interference cancellation readily occurring in equation (6.34).

Adaptation Criteria

Regardless of the recursion utilized, filtering is always performed according to equation (6.39). The recursions of the adaptive filters that follow account for the most general case of the DFE-PLL mode. To discount the PLL from the joint adaptation, the complex exponential of the PLL is simply replaced by unity. Thus, if we apply the three most widely used adaptations on the system at hand, the recursions for the feedforward filter are as

follows.

LMS The LMS-adapted FF filter is given by

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \mu_{LMS}\; e^*_{\hat{q}}(k)\; e^{-j\hat{\theta}(k)}\; \mathbf{s}(k) \tag{6.40}$$

ϵ-NLMS The ϵ-NLMS-adapted FF filter is given by

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \frac{\mu_{NLMS}}{\epsilon + ||\mathbf{s}(k)||^2}\; e^*_{\hat{q}}(k)\; e^{-j\hat{\theta}(k)}\; \mathbf{s}(k) \tag{6.41}$$

RLS The RLS-adapted FF filter is given by

$$\begin{aligned}
\mathbf{u}(k) &= \mathbf{s}(k)\; e^{-j\hat{\theta}(k)} \\
\mathbf{P}(k) &= \lambda^{-1}\left[\mathbf{P}(k-1) - \frac{\lambda^{-1}\mathbf{P}(k-1)\mathbf{u}'(k)\mathbf{u}(k)\mathbf{P}(k-1)}{1+\lambda^{-1}\mathbf{u}(k)\mathbf{P}(k-1)\mathbf{u}'(k)}\right] \\
\mathbf{w}(k) &= \mathbf{w}(k-1) + \mathbf{P}(k)\mathbf{u}'(k)\left[q(k) - \mathbf{u}(k)\mathbf{w}(k-1)\right] \\
\mathbf{a}(k+1) &= \mathbf{w}'(k)
\end{aligned} \tag{6.42}$$

6.2.4 Integrated Second-order PLL

The signal baseband model (6.2) includes an explicit term to account for transmitter-receiver phase variations at the chip rate. While these variations are due to mismatch and/or node mobility, they also affect the order-comparable carrier and code frequencies. This phase term is estimated using a second-order, stochastic gradient phase-locked loop (PLL) according to the MMSE criterion of the chip estimate as follows.

The chip estimate error is first generated

$$e(k) = q[k] - \hat{q}(k) \tag{6.43}$$

Using (6.22) and (6.17), the expression of $\hat{q}(k)$ is phase-corrected with a complex conjugate phase vector through a similar element-wise multiplication:

$$\hat{q}(k) = \frac{1}{E_{\mathbf{h}}(k)}\mathbf{h}'(k)\left[\mathbf{v}(k) - \mathbf{v}_b(k)\right] \cdot e^{-j\boldsymbol{\theta}(k)} \tag{6.44}$$

The gradient is then shown to be [130]

$$\begin{aligned}
\frac{\partial|e^2(k)|}{\partial\hat{\theta}} &= 2\;\mathrm{Re}\left\{\frac{\partial e(k)}{\partial\hat{\theta}}e^*(k)\right\} \\
&= -2\;\mathrm{Im}\left\{\frac{1}{E_{\hat{\mathbf{h}}}(k)}\hat{\mathbf{h}}'(k)\left[\dot{\mathbf{v}}(k)\cdot e^{-j\hat{\boldsymbol{\theta}}(k)}\right]e^*(k)\right\}
\end{aligned} \tag{6.45}$$

where $\dot{\mathbf{v}}(k)$ is the chip-by-chip, adaptively forwarded vector. The PLL is then implemented as

$$\begin{aligned}\psi(k) &= \mathrm{Im}\left\{\frac{1}{E_{\hat{\mathbf{h}}}(k)}\hat{\mathbf{h}}'(k)\left[\dot{\mathbf{v}}(k)\cdot e^{-j\hat{\boldsymbol{\theta}}(k)}\right]e^{*}(k)\right\}\\ \hat{\theta}(k+1) &= \hat{\theta}(k) + K_1\psi(k) + K_2\sum_{m\leq k}\psi(k)\end{aligned} \tag{6.46}$$

where K_1 and K_2 are the loop constants and often chosen such that $K_2 = K_1/10$.

A shorthand notation for a boxcar moving average implementation of the summation in equation (6.46) will be denoted by

$$\psi_{acc}(k) = \sum_{m\leq k}\psi(k) \tag{6.47}$$

for later referral in subsequent sections.

6.2.5 Linear Interpolation Stage

No Doppler-tolerant wideband receiver is complete without an interpolator capability. Under the narrowband assumption, the Doppler effect can be sufficiently modelled as a frequency shift. Contrary to this, in wideband signals the ideal model is better represented by an accurate rate change operation [121]

$$r(nT_s) = s\big((1+\delta)nT_s\big) \tag{6.48}$$

The removal of the non-uniform Doppler shift across the spectral content of the wideband signal is then performed as a sampling frequency rescaling operator

$$f_s = (1+\delta)f_s^{Doppler} \tag{6.49}$$

where $f_s^{Doppler}$ is the original sampling frequency of the receiver at which the Doppler effect is manifest.

The ultra high velocities in the ABU band would strain even a combined DFE-PLL equalizer being updated at the chip rate. Among the symptoms of such strain are excessive equalizer tap rotation and possible phase tracking loop instability [120]. Therefore, a successful state-of-the-art Doppler-tolerant receiver should be equipped also with an interpolator. This is particularly pertinent to the airborne acoustic medium, whose speed of propagation is less than one fourth that of the underwater environment.

The framework afforded by the DS CDMA adaptive engine derived earlier facilitates a front-end interplator stage, also operating at the chip rate, that can seamlessly plug into the global cooperative MSE minimization criterion.

The unique challenge in ABU phase tracking is primarily due to the ever present high-order airborne motion moments which accompany indoor-scale human movement. When

weighing the carrier frequency by the speed of acoustic propagation in-air, one notices that unlike RF for instance, movement indoors will not be “dampened” by a small factor, consequently giving rise to rapid variations in response to instantaneous motion. This can be best visualized by examining the Doppler shift formula and noting the linear slope that translates increased velocity into accentuated Doppler shift.

$$f_d \uparrow = \frac{f_{carr}}{c}\, v \uparrow \tag{6.50}$$

Thus it is likely that sophisticated workarounds have to devised, whereby cooperative phase tracking mechanisms are deployed. This will be elaborated on further in later sections when we develop the concept of *soft phase handover* in the context of a resilient operation taking place in a realistic environment with no prior assumptions on higher-order motion moments.

To this end, we define a *time-varying* stage that linearly interpolates the incoming complex baseband signal in accordance with a cost function driven by the core DS CDMA receiver. This arrangement is shown in figure 6.1.

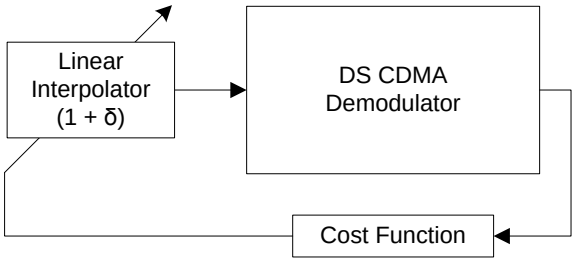
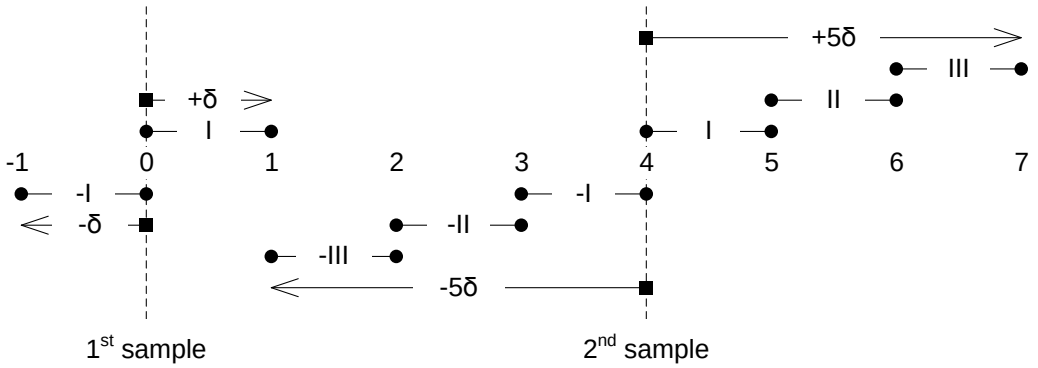


Figure 6.1: Adaptive resampling stage

In order to minimize the distortion introduced by linear interpolation (LI), the complex baseband signal is well oversampled and decimation occurs tacitly, i.e. all oversamples participate in interpolation as depicted in the eight-to-two example case of figure 6.2.



[a] Six pairs of samples are available to linearly interpolate the second sample by five times the current value of the interpolation index delta.
[b] Note that the implementation is intentionally kept symmetrical around the positive and negative directions despite having two extra pairs in the negative direction.

Figure 6.2: A diagram to illustrate a combined 8-to-2 interpolation and decimation[a]

In a decision-directed mode, the maximum likelihood (ML) carrier phase estimator for the signal $s(t;\phi)$ with a known information sequence $\{I_n\}$ is given by [110]

$$\hat{\phi}_{ML} = -\tan^{-1}\left[\mathrm{Im}\left(\sum_{n=0}^{K-1} I_n^* y_n\right) \Big/ \mathrm{Re}\left(\sum_{n=0}^{K-1} I_n^* y_n\right)\right] \tag{6.51}$$

where y_n is the output of the matched filter in the n^{th} signal interval T, and K is a positive integer defining the observation interval $T_0 = KT$. Therefore, in a similar fashion [122] the interpolation factor is adapted per step as

$$I(k+1) = I(k) + K_p\, \phi(k) \tag{6.52}$$

where K_p is a phase tracking constant, and $\phi(k)$ is the ML phase estimate. Within our code acquisition DS CDMA formulation, $\phi(k)$ is updated at the chip rate using the chip phase as opposed to the symbol phase

$$\phi(k) = \angle\hat{q}(k) \tag{6.53}$$

6.2.6 Putting It Together

With all the aforementioned components, the final receiver architecture is one that is user-oriented, decoupled, and extensible. A block abstraction of the overall receiver is presented in figure 6.3.

6.2.7 Initialization Mode

It is crucial for the convergence of the algorithm that the channel be allowed to stabilize before attempting to phase-correct, feedforward, or interpolate. This is because the SNR per chip is usually low. All these algorithmic operations are therefore halted by $2L$ chip intervals from the beginning of acquisiton. Since the maximum separation between two nodes corresponds to a code length, a delay of two codes allows at least L constructive in-sync chip additions giving good channel gain to bootstrap proceeding operations. Furthermore, at one chip preceding the instance at which subsequent processing kicks in, the channel is truncated in magnitude leaving only those coefficients whose strengths are comparable to the dominant impulse according to a pre-specified ratio. This ratio controls strength of noise rejection, and is typically given a value of 0.5.

6.2.8 Tracking Mode

Tracking is accomplished through sparsing the channel estimate. For ABU, multipath arrivals from far away are not expected. In addition, the focus of this work is Doppler compensation, cast as an equalization problem. Nevertheless, very short multipath—say off ultrasonic reflective surfaces such as windows or computer displays—can be accommodated for with the inevitable tradeoff of weakening the Doppler tracking capability of

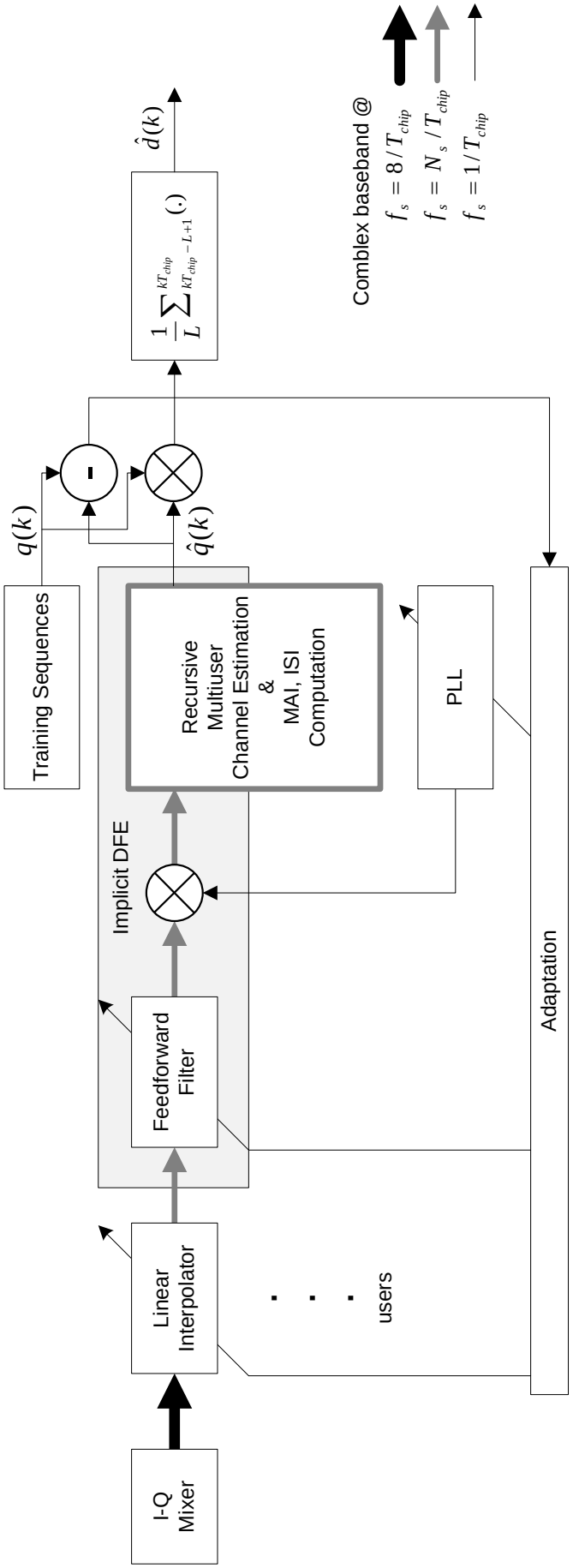


Figure 6.3: The novel receiver for airborne broadband ultrasound

the algorithm. This is because the adaptation energy will have to be split to work against two distinct physical dynamics.

Therefore similar to previous UWA work [129], we zoom into the vicinity of the strongest channel coefficient so that only $M = M_2 + M_1 + 1$ taps are retained; max, M_2 above, and M_1 below. All vector operations involving the sparse channel need not now be evaluated outside the M boundaries. Sparsed vectors are now denoted by the superscript ^{s}, and all affected equations in acquisition mode have to be reworked accordingly as follows:

$$
\begin{aligned}
\hat{\tilde{\mathbf{v}}}(k) &= \begin{bmatrix} \hat{\underline{\mathbf{v}}}^s_{L-1}(k) \\ \downarrow^{N_s} \quad \hat{\tilde{\mathbf{v}}}(k-1) \end{bmatrix} \\
\hat{\underline{\mathbf{v}}}^s_{L-1}(k) &= \sum_{m=M_1}^{M_2} q[k+m]\hat{\underline{\mathbf{h}}}_m(k) \\
\hat{\mathbf{h}}^s(k) &= \hat{\mathbf{h}}(k)\Big|_{M_2:M_1} \\
\hat{\mathbf{v}}_f(k) &= \hat{\mathbf{v}}_0(k) = \mathbf{v}(k) - \hat{\tilde{\mathbf{v}}}(k) + \hat{\mathbf{h}}^s(k) q[k] \cdot e^{j\hat{\boldsymbol{\theta}}(k)} \\
E_{\hat{\mathbf{h}}}(k) &= \hat{\mathbf{h}}^{s\prime}(k)\hat{\mathbf{h}}^s(k) \\
\hat{q}(k) &= \frac{1}{E^s_{\hat{\mathbf{h}}}(k)} \hat{\mathbf{h}}^{s\prime}(k)\hat{\mathbf{v}}^s_f(k),\ \hat{\mathbf{v}}^s_f(k) = \hat{\mathbf{v}}_f(k)\Big|_{M_2:M_1} \\
\hat{\mathbf{h}}(k) &= \lambda_{ch}\hat{\mathbf{h}}^s(k-1) + (1-\lambda_{ch})\hat{\mathbf{v}}_f(k) q^*[k] \\
\psi(k) &= \mathrm{Im}\left\{ \frac{1}{E_{\hat{\mathbf{h}}}(k)} \hat{\mathbf{h}}^{s\prime}(k) \left[\mathbf{v}^s(k) \cdot e^{-j\hat{\boldsymbol{\theta}}(k)} \right] e^*(k) \right\}
\end{aligned}
\tag{6.54}
$$

6.3 Motion Tracking in ABU – Theory of Operation

Here we establish all relevant concepts surrounding the operation of motion tracking in ABU. The proposed receiver structure incorporates a number of components that jointly track phase variations in the minimum mean square error (MMSE) sense. The rationale behind such a comprehensive receiver structure is to keep this early investigation within the context of the novel ABU modality as generic and extensible as possible. Despite heavily drawing on established techniques from the underwater acoustics (UWA) coherent communications literature, it is only through extending the methods and verifying them using deployment-driven characterization that a robust ABU theory of operation in the presence of noise, interference, and higher-order airborne motion moments can be realized. Thus, a combination of phase tracking methods will be jointly required in order to achieve motion resilience for various usage scenarios across the application space.

6.3.1 Foundational Concepts

Before we embark on discussing the theory of motion tracking through the use of coherent ABU processing, a few foundational comments are worth making first.

Instantaneous Frequency

The instantaneous frequency (IF) of a nonstationary signal is a parameter that characterizes the time-variedness the signal undergoes [29] throughout a given observation interval. It has a particular importance since it can often be correlated with a physical phenomenon being observed by means of that signal. Estimators for IF tend to differ depending on the nature of the signal at hand [30] (e.g. narrowband or wideband) which is further coupled with the end application itself (e.g. conditions surrounding measurements and utilized processing steps).

For ABU tracking, IF is the parameter that will enable us to infer the *velocity* of a mobile transducer (transmitter or receiver) w.r.t the a stationary one, or the relative velocity of two mobile transducers. The conversion is straightforwardly illustrated by the well-known Doppler formula

$$v_i(k) = \frac{c}{f_{carr}}\, f_i(k) \tag{6.55}$$

where $v_i(k)$ and $f_i(k)$ are the instantaneous velocity and frequency at time kT_c respectively, f_{carr} is the carrier frequency, and $c \approx 345$ m/s (the acoustic speed of propagation in air).

There are numerous IF estimators reported in traditional signal processing literature (see [30] and references therein). We define the continuous-time IF in the context of our chip-rate adaptive receiver as

$$f_i(t) = \frac{1}{2\pi}\frac{d}{dt}\phi_{chip}(t) \tag{6.56}$$

where $\phi_{chip}(t)$ is the *unwrapped* phase of the chip estimate. It is the single most sought after quantity that demodulation aims to recover. In discrete-time, simple differencing can be substituted for differentiation, provided that the discrete $\phi_{chip}(k)$ is reliable enough. Thus discrete-time IF at time kT_c becomes defined by

$$f_i(k) = \frac{1}{2\pi}\big(\phi_{chip}(k) - \phi_{chip}(k-1)\big) \tag{6.57}$$

$$\phi_{chip}(k) = \arctan\frac{\mathrm{Im}\,\{\hat{q}(k)\}}{\mathrm{Re}\,\{\hat{q}(k)\}} \tag{6.58}$$

The problem of noise influence on $\phi_{chip}(k)$ estimation will be dealt with in subsequent sections.

Instantaneous Bandwidth

It is further intuitive to conceptualize how IF might give rise to an Instantaneous Bandwidth (IB) analogous to the traditional frequency-bandwidth relationship, where IB represents a measure of IF's spread (i.e. standard deviation) at any given time.

The practical implications of IB translate to the need to relax the stop bands of the system's bandpass and lowpass filtering stages as to accommodate for the instantaneous

stretch (or shrinkage) of the ABU signal's spectral content whilst under motion stresses. That is, additional slack is added to the bandpass frequency for both the BPF and LPF.

Instantaneous Acceleration

The adaptive estimation of instantaneous acceleration (IA) is useful for two reasons. Firstly, passing acceleration measurements to the location algorithm contributes towards improved positioning estimate robustness, and to some extent helps mitigate against numerical artifacts such as those resulting from the use of nonlinear trigonometric functions like the arctangent that could be ill-posed at certain values. Secondly, and more importantly, online mechanisms for the inference of additional physical properties of motion (e.g. acceleration) could add to the richness of the sensory model and allow for sophisticated workarounds in the presence of higher motion moments in the ABU band. For instance, IA could be used to drive a control loop that would prevent LI from overcompensation of phase during instances of rapid acceleration. This concept will be further explained and developed later in this dissertation in subsection 6.5.4.

The IA estimator for coherent ABU is developed here in analogy with a monocomponent FM signal. Specifically, we note that the running despreader $\hat{d}(k)$ in equation (6.26) can be thought of as a *phasor* at any given time kT_c. This phasor represents a measure of how much, after demodulation, the receiver code is in-lock with the undistorted transmitted code. The rate of change in the demodulated despreader phase is now associated with the second motion moment, acceleration. This is because the complex depreader is already a first-order statistical moment of the chip estimates, which in turn can be seen as a complex-valued random variable (note the expectation operator in equation (6.26)). Dismissing the noise component for the sake of clarity, the despreader phasor can therefore be modelled in the continuous-time domain as

$$\begin{aligned} z_{\hat{d}}(t) &= a_{\hat{d}}(t)\ e^{j\phi_{\hat{d}}(t)} \\ &= x_{\hat{d}}(t) + j\ y_{\hat{d}}(t) \end{aligned} \tag{6.59}$$

with

$$\phi_{\hat{d}}(t) = 2\pi \int_{-\infty}^{t} f_i'(\tau)\ d\tau \tag{6.60}$$

and $f_i'(t)$ denoting the time-varying, second-order instantaneous frequency.

Differentiating the despreader phase, using the classic continuous-time formula for FM discriminators [59], yields the discrete-time estimator

$$\hat{f}_i'(k) = \frac{1}{2\pi} \frac{x_{\hat{d}}(k) y_{\hat{d}}'(k) - x_{\hat{d}}'(k) y_{\hat{d}}(k)}{x_{\hat{d}}^2(k) + y_{\hat{d}}^2(k)} \tag{6.61}$$

For a convergent $\hat{d}(k)$ i.e. after acquisition has been declared and tracking has begun, a number of approximations hold:

1. if $\mathrm{Re}\{\hat{d}(k)\}$ is stable
   - $x_{\hat{d}}(k) \approx 1$
   - $x'_{\hat{d}}(k) \approx 0$
   - $x^2_{\hat{d}}(k) \approx 1$
2. if phase lock is maintained, $y^2_{\hat{d}} \approx 0$

leading to the following second-order IF estimator in our ABU system

$$\hat{f}'_i(k) \propto \frac{1}{2\pi}\, y'_{\hat{d}}(k) \tag{6.62}$$

The second-order IF component is proportional to the rate of change (first derivative) of the imaginary part of the running despreader. In equation (6.62), equality is substituted for proportionality bearing in mind the limited code-length average which is provided by the running despreader. That is, additional processing[1] will be required if we were to convert this unitless entity into a meter-per-second-squared instantaneous acceleration estimator.

The final IA estimator is then obtained by scaling the second-order IF component according to

$$a_i(k) = \frac{c}{f_{carr}}\, \hat{f}'_i(k) \tag{6.63}$$

Doppler Vector Formulation

In the absence of extensive instrumentation allowing a highly controlled experimental setup, imprecise transmitter-receiver axis alignments will introduce slight measurement deviations to that obtained from groundtruth. This is due to the Doppler effect being dependent only on the *projected* transmitter velocity components onto the line connecting the transmitter and receiver [126], which under the far field assumption reduces to

$$\underline{\mathbf{v}}_{tr} \approx \frac{\langle \underline{\mathbf{v}}_t, \underline{\mathbf{r}}_r \rangle}{||\underline{\mathbf{r}}_r||^2} \underline{\mathbf{r}}_r = P_{\underline{\mathbf{r}}_r}(\underline{\mathbf{v}}_t) \tag{6.64}$$

with the projection operator P being defined as

$$P_{\underline{\mathbf{y}}}(\underline{\mathbf{x}}) \triangleq \frac{\underline{\mathbf{x}}^T \underline{\mathbf{y}}}{\underline{\mathbf{y}}^T \underline{\mathbf{y}}} \underline{\mathbf{y}} \tag{6.65}$$

where

[1] Such as simple scaling or exponential statistical weighting.

- $\underline{\mathbf{r}}_r = [r_x^r \; r_y^r \; r_z^r]^T$ is the 3D range vector of the receiver,
- $\underline{\mathbf{v}}_t = [v_x^t \; v_y^t \; v_y^t]^T$ is the 3D velocity vector of the transmitter, and
- $\underline{\mathbf{v}}_{tr} = [v_x^{tr} \; v_y^{tr} \; v_z^{tr}]^T$ is the resultant 3D velocity vector of the transmitter-receiver pair.

Spreading Code

Dithering has found applications in a variety of processing schemes pertaining to digital acoustics [140]. We note that the system-level design choice of using long spreading sequences is doubly advantageous: it allows the channel to have a one-to-one correspondence with the physical range. Additionally, the Gaussian-like distribution property of the spreading code has the beneficial by-product of producing a dithering effect that would combat bias and noise accumulation during adaptation, and consequently boost performance and reliability.

Leaky Adaptation

In our system, the known information sequence is used only as a means to probe the conditions of the wireless acoustic channel, be it displacement or mobility. Consequently, there is no keen interest in data-bearing communication. This allows for efficient realization of classic leaky adaptive techniques [57] as to aid stability, tracking, and combat drift.

Specifically, the proposed solution is designed with a number of ameliorative techniques in mind:

- *Circular-leaky LMS*: This variant of the classic LMS algorithm is developed to combat drift (i.e. unbounded weights growth) and ensure unbiased estimates. In the original formulation, leakage is introduced to a single tap, per update step, only if that tap magnitude exceeded a threshold [115, p. 303] [96].
- *Independent regressors*: When the regressors are i.i.d., the design of the LMS step size is considerably relaxed in the context of its influence on convergence, learning, and stability. This is because i.i.d. regressors tend to approximate small step-sizes performance on a variety of adaptation metrics [115, p. 384-387]. For our ABU system at hand, this translates to more leeway in experimenting with various LMS step sizes.

Building on the above rationale, we introduce the following *ad hoc* perturbation to the standard LMS adaptation criterion. We dub this arrangement Doubly Reinforced Code Epoch (DRCE) and define it as follows: Once every code, the last chip is skipped and replaced by the first incoming chip which is consequently used to generate the adaptation error. The next iteration proceeds normally at the first chip again. This effectively doubles the duration allocated for the adaptation of the first chip, and injects data-dependent perturbation to weights. The modification introduced to the circular Gold sequence at time kT_c is simply

$$\mathbf{q}(k) = \begin{bmatrix} q[k] \\ \vdots \\ q[k-L+2] \\ q[k-L] \end{bmatrix} \tag{6.66}$$

where $q[k] = q[k-L]$.

From empirical grounding, it was found that this *ad hoc* technique has a perceivable effect on Doppler tracking for the PLL and DFE-PLL configurations, ranging from mild for the former to critical for the latter.

For the sake of completeness, a last remark ought to be made here. A less formal inspiration for DRCE leaky adaptation can be found in the domain of pedestrian tracking using inertial sensors. According to the so-called zero-velocity updates (ZUPT) criterion, at the end of each estimation cycle, the velocity is set to zero upon detection of a stance phase in order to prevent excessive error growth in the navigation model [53]. This is vaguely related to DRCE—observing the periodicity of back-to-back code transmissions, DRCE injects a cyclic perturbation to weights on a code-by-code basis. However, in DRCE, the perturbation is derived from the incoming data too.

6.3.2 Phase Tracking Methods

The various phase-tracking mechanisms that the novel system incorporates are presented next. For the most part, the theory of operation is consistent throughout the various phase-tracking stages of the algorithm. A special interesting case will be discussed, that somewhat deviates from the theoretical norm. The following will enumerate all cases and their permutations. Empirical data will be provided in section 6.5 in order to support the theory presented here.

Fractional Channel

In regards to the impact of the channel on the coherent algorithm, few points are worth bringing to the fore:

- Truncating the fractional channel in magnitude is of utmost importance for adaptation. It is the single step at the bottom of the hierarchy that ensures proceeding functions act on the cursor chip at any given time kT_c while rejecting excessive noise levels due to the relatively low sensitivity of the piezo film transducer. The cursor channel taps can also be viewed as accumulators that provide coding gain against noise.

- A fractional complex channel also performs limited equalization whose strength increases with the rise of the chip oversampling rate. That said, the dynamic filtering performance of the channel tends to drop later on, depending on a combination of

factors: tap migration [83], PLL performance determined by its tracking constants in the PLL-only mode, and higher order motion moments.

Monitoring the unwrapped phase of the cursor channel coefficients reveals that the channel has indeed little contribution to phase tracking, once various other mechanisms have been deployed.

PLL

In isolation, various PLL configurations have been analyzed in the RF literature. In the context of GPS tracking, the unconditionally stable second-order PLL was found to be sensitive to acceleration stress [142]. A second-order, decision directed PLL has been successfully embedded within the classic DFE kernel for improved and aided phase-coherent UWA acoustic communications [130].

Due to the highly nonstationary nature of Doppler distortion in the ABU band, a third order PLL, though capable of improved tracking in the presence of acceleration, was ruled out from the beginning. A typical usage scenario indoors in the ABU band would involve prevalent high order motion moments (e.g. jerk stress), which could cause stability problems for a third order PLL. Therefore, the same PLL structure proposed by Stojanovic et al. was deemed to be the PLL design of choice.

The second order PLL has been experimented with extensively. This was carried out in isolation from other system phase-tracking components, with the exception of the fractional channel of the core DS CDMA engine. Using empirical measurement as a basis, a number of properties governing PLL's Doppler tracking attributes in the ABU band has been observed. These are:

- *Leaking*: This concept will be revisited when discussing the feedforward (FF) filter component of the implicit DFE. Despite having been developed primarily for LMS FF, DRCE leaky adaptation was found to influence the operation of the second-order PLL in the ABU band. This influence comes in the form of a stabilizing effect under certain operational conditions.

- *Channel truncation*: The truncation of the channel magnitude plays an additional role in maintaining phase-tracking stability. When PLL's adaptation introduces oscillatory transients as determined by the tracking constants, truncation has a smoothing effect on the PLL's phase-trackability.

- *Sensitivity to higher-order motion moments*: Unlike in UWA coherent communication systems, the PLL does not constitute a reliable means for inferring the motion pattern. It should rather be viewed as an aiding mechanism in the joint adaptive sense. Moreover, the ever present higher-order motion moments in ABU will degrade the quality of PLL's phase-tracking effort.

- *Limits on trackability*: Classically in UWA, a PLL compensates for carrier motion-induced distortion [128] as opposed to symbol distortion [97]. Residual Doppler is

dealt with by the feedforward filter (FF) part in a DFE, whose effect on adaptation is demonstrated by tap translation and rotation in FF [109]. And since the Doppler effect in the ABU band (which has large fractional bandwidth) affects both of the order-comparable carrier and code frequences, there is a fundamental limit on the PLL's ability to correct for phase variations arising from motion. However, fast chip-rate update can to some extent boost the responsiveness of the PLL which could be of particular merit in certain situations.

- *Relationship to chip oversampling* : Empirical evidence for the ABU operation in the presence of Doppler suggests that the PLL's ability to extract phase variations from the incoming chips is enhanced by increasing the chip oversampling rate.

- *Tacit lowpass filtering via downsampling*: In PLL-only mode, the monotonic unwrapped phase of chip estimates is affected by jitter. This means that direct differencing in equation (6.57) will give rise to erroneous IF estimation. Somewhat similar to [55], one solution could be to low-pass filter the unwrapped phase estimate first. A computationally more favourable alternative is to simply decimate the unwrapped phase before taking the derivative. This effectively discards all instantaneous jumps in the unwrapped phase. To this end, in the PLL-mode, discrete IF computation is reworked as

$$f_i(k_\downarrow) = \frac{1}{2\pi}\big(\phi_{chip}(k/L_\downarrow) - \phi_{chip}(k/L_\downarrow - L_\downarrow)\big) \tag{6.67}$$

  which is only defined at integer indices. $L_\downarrow$ is chosen commensurate with human-rate measurements, e.g. 30Hz. This also has advantages from signal chain point of view. That is, communicating measurements to upper network layers is conducted at a much reduced rate compared to that of transceiver processing.

- *Trackability*: The trackability of the PLL in the presence of Doppler shift is an exercise in tradeoff between responsiveness and stability. This tradeoff is determined by the tracking constants employed, namely K_1 and K_2 in equation (6.46). Fine-tuning these constants in the ABU band is difficult. This is because achieving the small loop noise bandwidth for systems with large fractional bandwidth is very hard [142, p. 183] [122]. This is also exasperated by the rather large dynamic range of IF resulting from typical human-scale movements in the ABU indoor environment. Meaning, it is hard to calibrate tracking constants for the wide range of velocities that are likely to arise.

DFE

In similar vein to the PLL discussion, observations from the empirical operation of the DFE in the ABU band are given next:

- *Limits on trackability*: The rapid equalizing ability of the fractional DFE when operated at the chip rate allows for modest Doppler trackability even in highly Doppler-susceptible ABU band. When under Doppler stress, the FF or the implicit DFE

exhibits a specific evolutionary pattern, in reaction to the incoming Doppler-shifted chips. This pattern can be observed in two ways:

1. The magnitude peak within the filter coefficient vector will continuously move in order to maintain synchronization lock. Henceforth, this will be referred to as the *translation* effect.
2. The phase of the complex taps will rotate in a way corresponding to the Doppler levels. Hereafter, this effect will be abbreviated as tap *rotation*.

However, the DFE alone will only perceive the incoming *total* (carrier and code) Doppler distortion as chip time-dilation/compression (or symbol for non-spread systems). Thus unlike the PLL, the DFE alone was not able to demonstrate an ability to measure the Doppler-induced velocity to levels comparable to those of the groundtruth.

- *Operational conditions*: When analyzed in isolation, and building on previous results from UWA coherent communications [97], there are three properties that the DFE ought to adhere to in order to successfully track Doppler

  1. High update rate for maximum tracking performance, which is readily addressed by the fast chip rate.
  2. Low adaptive step size to minimize misadjustment. This is confirmed by the need to scale down both RLS and NLMS adaptation criteria by roughly a code length for them to result in similar equalization effect on the running despreader when compared to simple LMS. However, both RLS and NLMS could not successfully yield a phase pattern correlated with motion. This fact could be linked to individual chips not being reliable enough to steer adaptation adamantly; rather, their combined fluctuating effect is what drives the update. In this, the ABU DFE operation is closer to that of a control loop—while the chip error will hit zero sometimes, it constantly fluctuates at other times assuming $\{+1,-1\}$ chip values so that overall, it keeps the despreader well-behaved. This also explains the “third” chip at zero in the performance scatter plots of the algorithm (e.g. see appendix B).
  3. Short feedforward filter span keeps the convergence time acceptable. This is also readily fulfilled by the chip-rate structure of the derived receiver.

- *Leaking*: DRCE *leaky* adaptation is of major influence on the proper functioning of the DFE in our ABU system.

DFE-PLL

In the joint DFE-PLL mode, there is an apparent phase tracking redundancy. In UWA systems, PLL’s tracking constants are made substantially higher than the FF’s learning

rate—which is characterized by its length[2] and percentage of error feedback—such that the PLL dominates the FF equalizer [72].

Some observations can once again be made from the empirical operation of the DFE-PLL in the ABU band:

- *Competition*: A special case arises when a competition between the feedforward complex filter and the second-order PLL is allowed. When both attempt to compensate for Doppler-induced phase distortion in near-equal strengths, it was observed empirically that the unwrapped phase of the chip estimates is no longer a monotonically increasing function. Rather, it possesses some notable attributes that are correlated with motion.

  1. The unwrapped phase tends to inflect upon reversal of motion.
  2. The unwrapped phase drifts during motion pauses with a certain slope.
  3. Provided that the responsiveness of the DFE-PLL configuration is tuned accordingly, the range covered by the unwrapped phase, in a time interval of continuous motion, is very close to the limit of the speed in that motion segment. This can be ascertained by converting the unwrapped phase into a unitless turns-per-second entity according to the direct weighing

     $$\Omega_i(k) = \frac{c}{f_{carr}} \frac{1}{2\pi} \phi_{chip}(k) \tag{6.68}$$

     where Ω_i is the weighted instantaneous phase, measuring the turns per second that the phase makes throughout the course of the periodic motion stimulus.
  4. The shape of the unwrapped phase is also a function of the responsiveness of this competition.

  This seems to suggest that the unusual competition between the PLL and DFE in tracking the phase results readily in a *measure* of the IF.

- *Leaking*: The DRCE *leaky* adaptation remains of pronounced efficacy for certain setups (carefully tuned competition) in the joint DFE-PLL configuration in our ABU system.

LI

As discussed earlier, LI is more inclusive in correcting for wideband Doppler-shifted signals, in that it will address the different distortion rates in all present frequencies. In our system this means that the carrier and code frequencies can in principle be both dealt with. The issues around LI governing operation are:

- *Trackability*: The adaptive responsiveness of LI is dictated by the choice of K_p in equation (6.52). Higher values though may inject excessive noise into the system

[2] Note that in the absence of multipath in ABU, the FF length provides added phase trackability.

and/or degrade performance under rapid Doppler dynamics (i.e. high-order motion moments). Once again, the optimal choice is a compromise between responsiveness and robustness. Ideally, faithful modelling of ABU's usability scenarios indoors would afford a better insight for this design process. Sharif et al. [122] utilize an SNR-driven characterization of various system conditions (e.g. velocity, acceleration, transient behaviour, etc.) in order to arrive at optimal values for the tracking constant. In other words, better-performing models might be designed using knowledge about movement characteristics for a particular application scenario. The endeavor in this disseration is, however, aimed at investigating the viability of phase coherent ABU in general and no such characterization will be attempted at this stage.

- *Leaking*: Contrary to the above findings—namely that of the PLL, DFE, and DFE-PLL—leaky adaptation is not required in the case of LI. This is attributed to LI's ability to adjust the pace of sampling in conjunction with incoming Doppler levels. Effectively, this means removing much of the phase tracking strain to which leaking was originally devised.

- *Sensitivity to higher-order motion moments*: The inspection of figure 6.2 reveals an obvious weakness of LI. Under expeditious motion intervals caused by higher-order moments, overcompensation—as determined by K_p—can occur whereby the interpolated value of the second sample falls outside the resolvable window of oversamples either in the positive or negative direction. This will vastly degrade the quality of interpolation.

LI-DFE

Despite LI's agility in tracking Doppler, distortion introduced by linear interpolation needs to be dealt with. The matched filtering properties of the FF readily provides an equalizing effect for a variety of distortion sources. Therefore, the LI-DFE combination represents a very viable phase tracking mechanism.

- *Trackability*:

  1. The interpolation factor's polarity determines the direction of movement: backward or and forward.
  2. Under balanced conditions, minimal tap translation and rotation per motion segment in FF takes place since LI will dominate phase tracking. By "balanced", it is meant that the interpolation factor will return to rest condition (zero) at the end of any motion segment.

- *Leaking*: Similar to LI, the LI-DFE configuration does not require leaky adaptation.

- *Sensitivity to higher-order motion moments*: Acceleration and/or jerk can tip the balance of LI such that the interpolation factor will not return to rest. In this case, the

FF will intervene briefly to adjust the phase resulting in ambiguity in the measured rate of change from the slope of the unwrapped phase of the chip estimate.

Interpolation-DFE-PLL

Hinted at previously [55], there is a brief mention in the UWA literature of a tri-combinatory interpolation plus DFE with embedded second-order PLL. This configuration relies on the phase tracked by the PLL to drive the resampling operation. It, however, was not pursued for the following reasons:

- The work is unpublished.[3]
- As described, the interpolation was based on a filter-bank approach. This means that there could be a large memory storage penalty and/or computational overhead to the method. This is especially disadvantageous for a real-time, resource-constrained realization.
- In the ABU band, as elaborated on above, there is empirical evidence that suggests that enhanced PLL operation requires a higher oversampling rate in order to extract phase variations. This is undesirable since it will impact all aspects of operation; sampling, complexity (i.e. silicon footprint), and power consumption.

6.4 Experiment, Data Capture, and Analysis Methodology

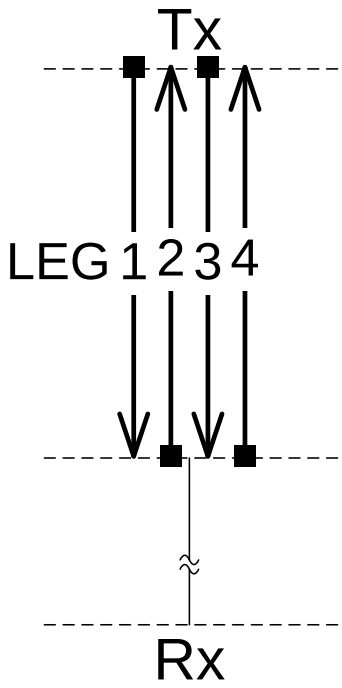
This section touches on the utilized experimental setup and, data capture arrangement, and analysis methodology. This is done in order to give the reader a better grasp on the non-algorithmic issues surrounding the development of ABU motion tracking. The setup for the data collection experiment is detailed first. Then an overview on the employed strategy for data capture, processing, and analysis follows.

6.4.1 Experimental Setup

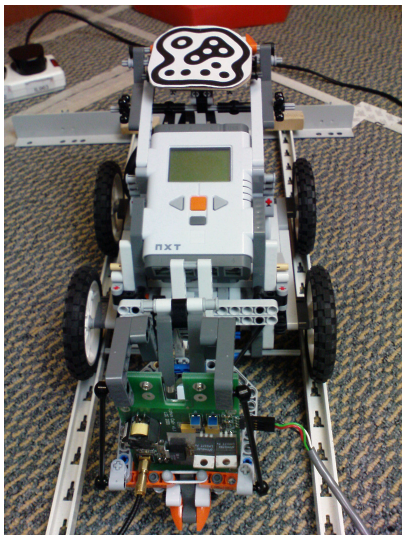
Throughout the empirical study of section 6.5, one dataset will be used for the characterization of various algorithmic configurations. This dataset has typical maximum speed (about 0.8 m/sec) and contains high-order motion moments. In order to facilitate the characterization of these various adaptive algorithmic combinations, we utilize a vision-based tracking system for the generation of real-time position and speed groundtruth. A transmitter is mounted on a Lego Mindstorms robot moving forward and backward on a track while facing a stationary ABU receiver. The robot is tagged with a fiducial marker [113]. A PC-based application written in .NET C# monitors the movement of the robot using the computer vision-based tracking. Upon motion, the multi-threaded application triggers an FPGA-based ABU transmitter and simultaneously commences data acquisition. The synchronization of vision tracking and data acquisition is therefore achieved implicitly since

[3] This is confirmed by the authors.

they are executed in the same multi-threaded application. The all-digital transmitter has an independent oscillator to the PC clock. The experimental setup used for data collection is shown in figure 6.4. The structure of the C# program utilized for the Doppler experimental setup is given in appendix E.



(a) Transmitter path moving forward and backward while facing a stationary receiver.

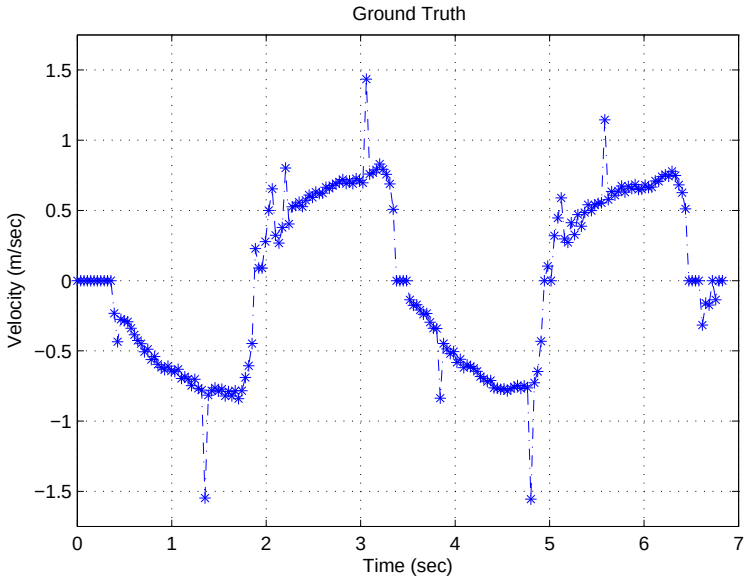


(b) Lego Mindstroms robot with mounted transmitter.

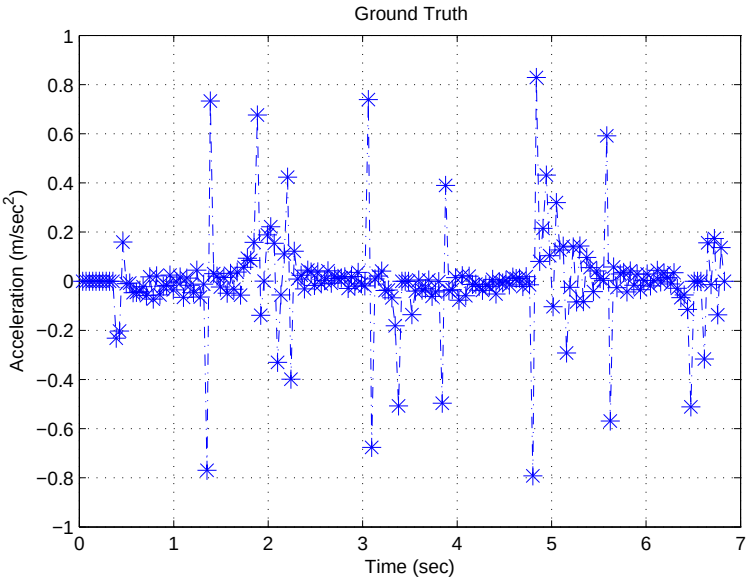
Figure 6.4: Experimental setup

The vision-based groundtruth is depicted in figure 6.5.[4] An initial motion stress was programmed into the robot in order to allow vision-tracking to register an "about-to-move" event necessary for triggering the FPGA transmitter and synchronizing acquisition and tracking. Before proceeding with the evaluation section, comments on the fidelity of the experimental setup are now due in order to set the scene for analysis. It is likely that the imperfect experimental setup gives rise to asymmetrical high-order motion moments in the forward and backward directions. This is also in line with the rear-wheel drive layout of the robot, and the front-mounted transmitter. Without access to a precision experimental setup with better controlled conditions, further analysis of the empirical operation in regard to high-order motion moments is difficult. Moreover, the vision tracking camera used for groundtruth generation is of limited shutter speed capabilities (around 27 Hz) which limits the time resolution (instantaneous accuracy). That said, and despite these limitations, the experimental setup is sufficient for the purposes of this dissertation. The primal focus is on measuring velocity, which can be achieved to good accuracies under this setup.

[4]Note that groundtruth is reported "as is" i.e. unsmoothed and unfiltered.



(a) velocity



(b) acceleration

Figure 6.5: Vision tracking groundtruth

6.4.2 Data Capture and Analysis

On the vision-tracking application side, the robot 2D coordinates along with their time stamps are recorded throughout an experiment run. Simultaneously, data from the stationary receiver is buffered in memory. After the specified time interval has elapsed, both records (stamped coordinated and ABU receiver data) are logged into files. These files are them imported into MATLAB for subsequent processing and analysis. Processing in MATLAB is ruled out due to the adaptive nature of operation and the unreasonably long time it would take MATLAB to go through say a 7-second dataset.

MATLAB formats the data accordingly and prepares a file for the actual processing. The algorithm is written in templatized C++ and compiled using the the GCC GNU compiler [5]. In addition to computations, a C++ class containing ping-pong data structures records at run-time the evolution of all algorithmic variables. Overlapping this, the class will also be logging processing results into files. Upon completion, these files are imported back into MATLAB for analysis and subsequent visualization.

6.5 Operational Parametric Interplay – Empirical Study

In this section, we present performance plots on all ABU theory of operation developed thus far. When appropriate, additional commentary and expanded discussion will be supplied. The various algorithmic parameters under which performance plots where generated will be thoroughly listed.

The evaluation is structured according to the three methods which demonstrated some ability to track Doppler-induced phased variations in ABU: PLL, DFE-PLL, and DFE-LI. For each method, velocity inference and the evolution of phase tracking will be examined. Following this, stress analysis for the DFE-PLL and DFE-LI will be performed using the IA estimator discussed on page 70.

6.5.1 PLL

Unless otherwise stated, PLL tracking is deployed after the duration of two code lengths has elapsed.[5]

As discussed earlier, the PLL's ability to track Doppler-induced phase variations alone is limited. An empirical characterization of PLL's performance as obtained from the real dataset is presented next. This is done in order to understand the behaviour of PLL ABU tracking in isolation of other system components.

The influence of the permutations of various algorithmic parameters was investigated. Table 6.1 summarizes the obtained results. The four algorithmic parameters shown are: channel rescue threshold $\mathcal{E}_{E_h}$, chip oversampling N_s, tracking constants $K_{1,2}$, and leaky adaptation $DRCE$. The figures of merit against which performance is qualitatively assessed are tracking stability and the relative spread of the chip estimates. Chip spread is

[5] This is controlled from within the algorithm by the constant `PHASE_KICKIN`.

related to stability in the sense that the less spread the chip estimates exhibit, the more likely PLL-only tracking will remain stable.

Table 6.1: Summary of PLL performance

RESULTS		ALG. PARAMS.			
stable	spread	$\mathcal{E}_{E_h}$	N_s	$K_{1,2}$	$DRCE$
YES	MODERATE	0	2	1×10^{-5}	FALSE
YES	LARGE	0	2	1×10^{-5}	TRUE
NO	-	0	2	5×10^{-5}	FALSE
NO	-	0	2	5×10^{-5}	TRUE
NO	-	0	4	1×10^{-5}	FALSE
NO	-	0	4	1×10^{-5}	TRUE
YES	MODERATE	0	4	5×10^{-5}	FALSE
NO	-	0	4	5×10^{-5}	TRUE
NO	-	1×10^{-6}	2	1×10^{-5}	FALSE
NO	-	1×10^{-6}	2	1×10^{-5}	TRUE
NO	-	1×10^{-6}	2	5×10^{-5}	FALSE
NO	-	1×10^{-6}	2	5×10^{-5}	TRUE
NO	-	1×10^{-6}	4	1×10^{-5}	FALSE
NO	-	1×10^{-6}	4	1×10^{-5}	TRUE
NO	-	1×10^{-6}	4	5×10^{-5}	FALSE
YES	BEST	1×10^{-6}	4	5×10^{-5}	TRUE
YES	MODERATE	1×10^{-7}	2	1×10^{-5}	FALSE
YES	LARGE	1×10^{-7}	2	1×10^{-5}	TRUE
YES	LARGE	1×10^{-7}	2	5×10^{-5}	FALSE
NO	-	1×10^{-7}	2	5×10^{-5}	TRUE
NO	-	1×10^{-7}	4	1×10^{-5}	FALSE
NO	-	1×10^{-7}	4	1×10^{-5}	TRUE
YES	MODERATE	1×10^{-7}	4	5×10^{-5}	FALSE
NO	-	1×10^{-7}	4	5×10^{-5}	TRUE

To comment on these results, we note that the stability of the algorithm is determined by the choice of the tracking constants and the chip oversampling rate. Tracking constants are also linked with the Doppler-shift levels present in that dataset. Chip oversampling is a system-level parameter.

Two mechanisms for aiding tracking and stability were characterized. The enabling of DRCE leaking in the PLL-only mode seems to have a mild yet interesting effect on performance. Specifically, the combination of $N_s = 4$ and DRCE $=$ TRUE was the only stable permutation when channel rescue mechanism at $\mathcal{E}_{E_h} = 1 \times 10^{-6}$ was deployed. Simultaneously, this was the most well-behaved case in terms of chip estimate spread. The critical value of the channel power rescue at $\mathcal{E}_{E_h} = 1 \times 10^{-6}$ seems to have helped

stabilize the DRCE mode when the PLL was overdriven at $K_1 = 5 \times 10^{-5}$. Although other stable cases could be obtained when neither mechanism was deployed, the major difference, was a noticeable increase in the chip scatter spread. This can be ascertained by examining the comprehensive performance plots provided in appendix A.

For the sake of completeness, a final noteworthy comment on the second-order PLL needs to be made. The equivalent of the boxcar accumulator in equation (6.47) is known sometimes in the RF literature by its functional name: velocity accumulator [142, p. 182]. Although correlated with motion sometimes, $\psi_{acc}(k)$ in our ABU system can not be viewed as a reliable indicator of the movement. This is again linked to the excessive levels of high order motion moments in ABU to which the second-order PLL is susceptible. These motion moments will deform and offset $\psi_{acc}(k)$ continuously in time.

Velocity Inference

The unwrapped phase of the chip estimate clearly illustrates the time-dilation/compression in the form of convex/concave segments jointed by inflection points. This can be seen in figure 6.6. It is also interesting to note that even in cases when timing is lost, phase somewhat faithful to the Doppler pattern is still obtained. This is mostly due to the spreading code, and back-to-back continuous transmission. That is, the rate at which individual chips adjust their phases remains correlated with motion.

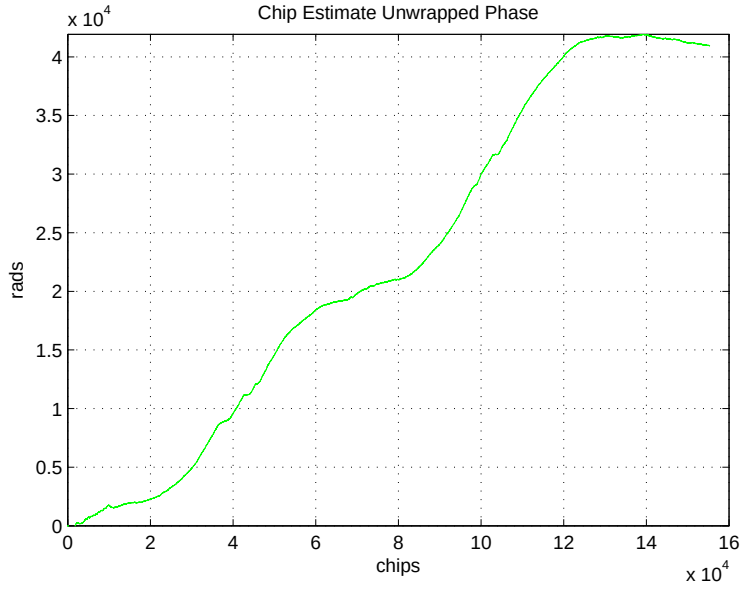


Figure 6.6: Chip estimate phase pattern under the Doppler effect of the test case

Using equations (6.67), we can therefore obtain a crude estimate of the IF, in the PLL-only mode. The instantaneous velocity is then computed from the weighted IF according to equation (6.55). Utilizing 30 Hz as an observation rate for decimation, 340 m/s speed of sound in air, 20 kHz chip rate, and a 50 kHz carrier frequency we get the measurements in figures 6.7 and 6.8.

Although generally the PLL-only configuration can at best yield crude approximation of the motion for relatively low speed, there is one interesting finding worth putting in

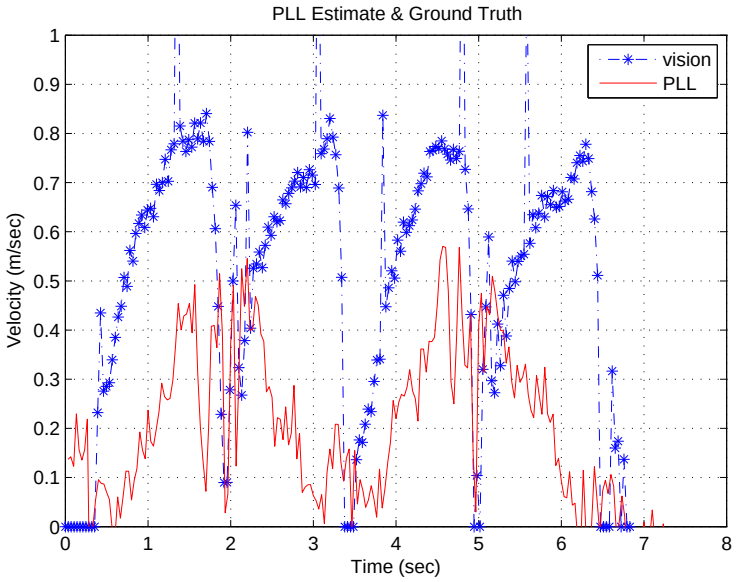


Figure 6.7: Absolute-value vision tracking groundtruth and 2x PLL velocities

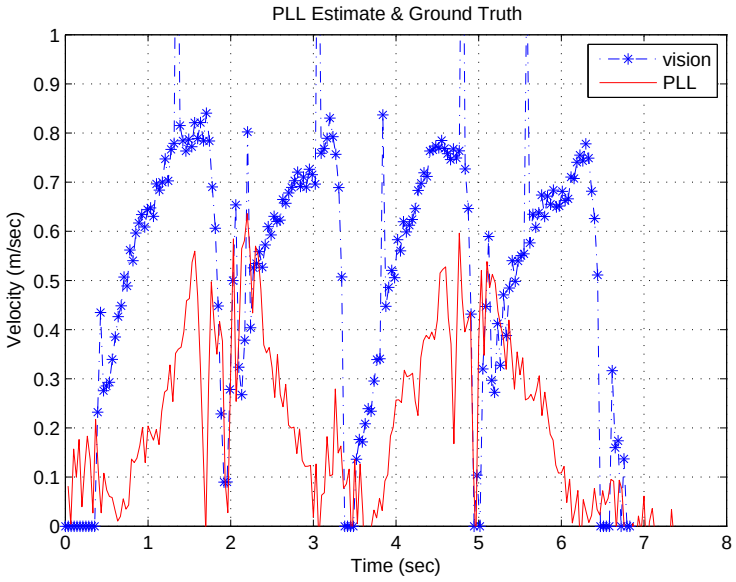


Figure 6.8: Absolute-value vision tracking groundtruth and 4x PLL velocities

words at this stage. With $N_s = 2$ the velocity range is slightly decreased when compared to that obtained from $N_s = 4$. At the same time, K_1 has to be higher for $N_s = 4$ (compared with $N_s = 2$) in order for tracking to be stable. This effect can be observed by inspecting figures 6.7 and 6.8.

The last entry of the second batch in table 6.1 may suggest that higher PLL tracking constants yield better velocity range in the PLL-only mode. The two proposed methods, namely channel rescue and DRCE, could potentially help to stabilize the operation of the second order PLL at higher $K_{1,2}$ values.

Phase Evolution

The dynamic behaviour of the second-order PLL under the Doppler effect is next demonstrated. The two stable entries with moderate spread from the first batch in table 6.1 are chosen for illustration. Examining figures 6.9 & 6.10, the sensitivity of the PLL to the instances of high order motion moments is evident. As can be seen, this is most pronounced in $\psi(k)$ and $\psi_{acc}(k)$.

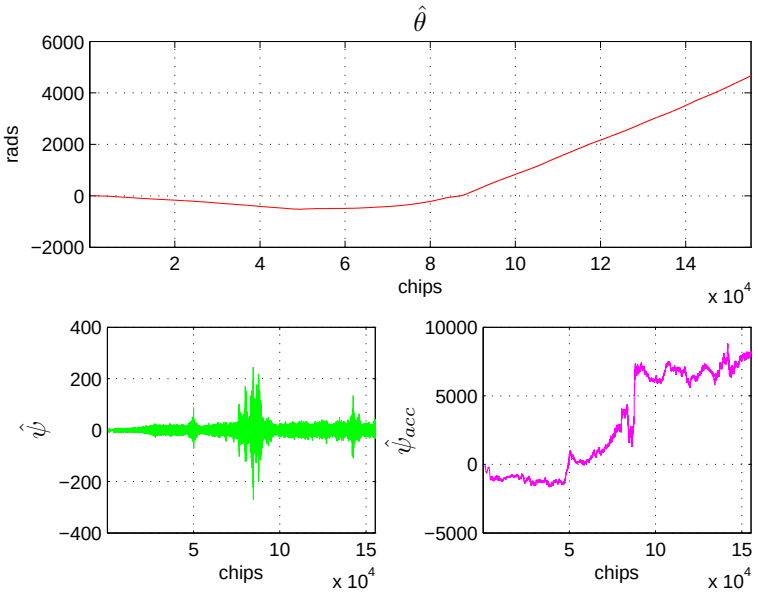


Figure 6.9: The parametric evolution of the 2x PLL under the Doppler effect

6.5.2 DFE-PLL

The empirical observation of the special case arising when the DFE and PLL are competing in ABU phase tracking is intriguing. Table 6.2 summarizes the conditions under which a phase pattern somewhat resembling the motion groundtruth was obtained. This phenomenon could only be observed with 4x chip oversampling rate.

In line with the earlier discussion, three examples of the empirical operation for the DFE-PLL mode will be shown. No solid conclusions can be drawn from this mode. Rather, the intention is to demonstrate the correlation between the unwrapped phase of the chip estimate, and the motion pattern exerted. That is, in this mode, under certain

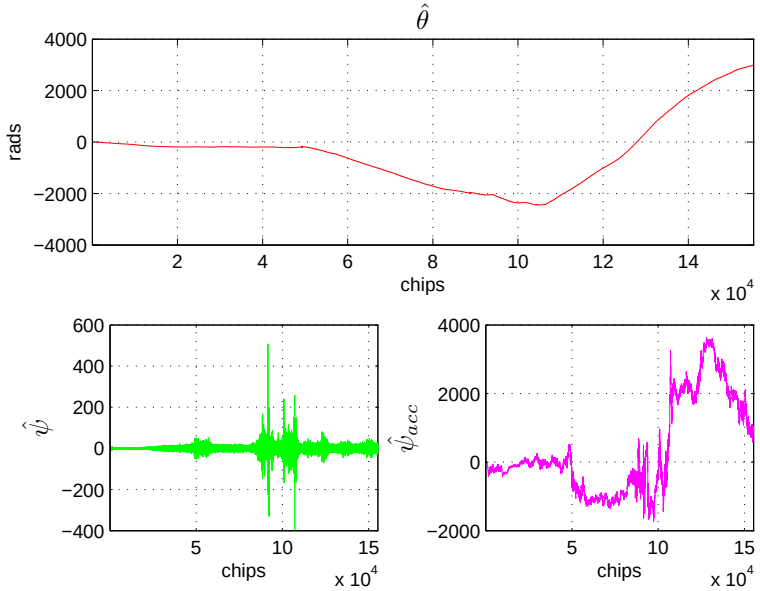


Figure 6.10: The parametric evolution of the 4x PLL under the Doppler effect

Table 6.2: Summary of DFE-PLL performance

RESULTS	ALG. PARAMS.				
motion correlated	N_s	μ_{LMS}	L_{ff}	$K_{1,2}$	$DRCE$
YES	4	1×10^{-4}	2	1×10^{-4}	FALSE
YES	4	1×10^{-4}	2	1×10^{-4}	TRUE
NO	4	5×10^{-4}	8	5×10^{-4}	FALSE
YES	4	5×10^{-4}	8	5×10^{-4}	TRUE
NO	4	5×10^{-4}	10	5×10^{-4}	FALSE
YES	4	5×10^{-4}	10	5×10^{-4}	TRUE

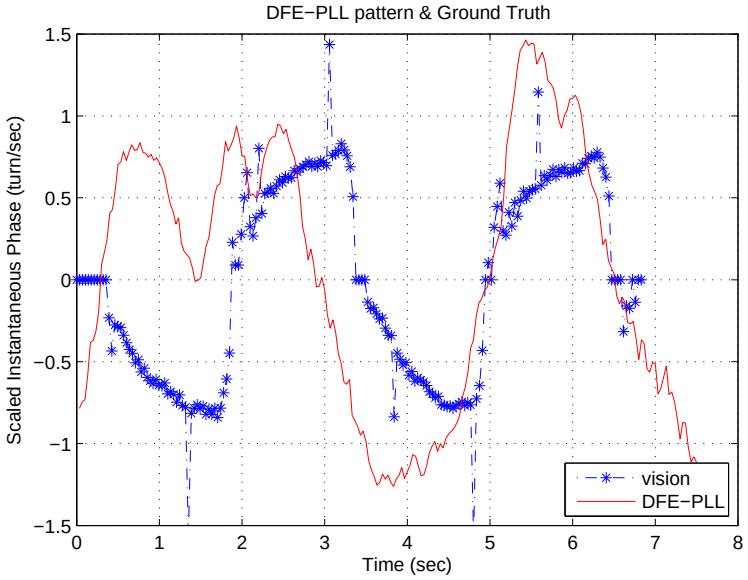
conditions, the adaptive unwrapped phase could potentially be a direct estimator of the instantaneous frequency without applying any further processing. The full plots of the DFE-PLL combinations reported in table 6.2 are given in appendix B.

Instantaneous Doppler Pattern

For all three visited cases, the unwrapped phase of the chip estimate is scaled according to equation (6.55) and the DC component is removed. The resulting entity refers to the turns the unwrapped phase covers over the course of motion. In the absence of solid theoretical formalism, this will be characterized using the unit of “turn-per-second” bearing in mind the periodicity of the forward-and-backward motion stimulus.

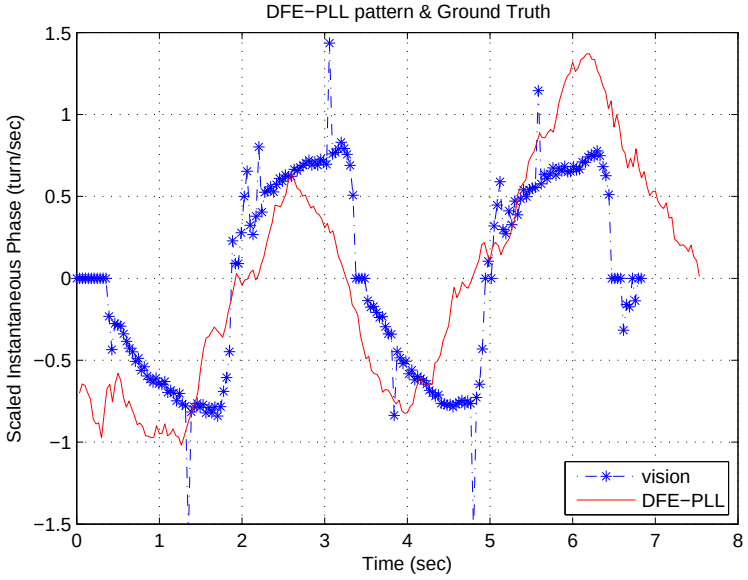
Using the first, fourth, and sixth entries of table 6.2, the corresponding graphs of the scaled unwrapped phase of the chip estimates are shown in figures 6.11, 6.12, & 6.13 respectively.

An interesting finding surfaces here. Note that as we increase the strength of DFE-



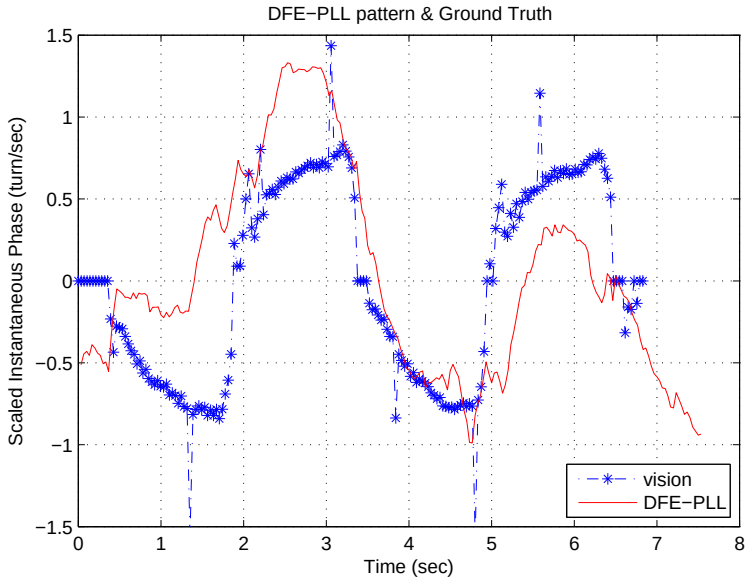
[a] parameters: $N_s = 4$, $\mu_{_{LMS}} = 1 \times 10^{-4}$, $L_{ff} = 2$, $K_1 = 1 \times 10^{-4}$, and $DRCE =$ `FALSE`

Figure 6.11: Vision tracking groundtruth and case 1[a] DFE-PLL velocities



[a] parameters: $N_s = 4$, $\mu_{_{LMS}} = 5 \times 10^{-4}$, $L_{ff} = 8$, $K_1 = 5 \times 10^{-4}$, and $DRCE =$ `TRUE`

Figure 6.12: Vision tracking groundtruth and case 2[a] DFE-PLL velocities



[a] parameters: $N_s = 4$, $\mu_{_{LMS}} = 5 \times 10^{-4}$, $L_{ff} = 10$, $K_1 = 5 \times 10^{-4}$, and $DRCE =$ TRUE

Figure 6.13: Vision tracking groundtruth and case 3[a] DFE-PLL velocities

PLL competition—as parameterized by LMS step size, filter length, and tracking constants—leaking becomes important for the preservation of the instantaneous Doppler pattern embedded in phase information. Simultaneously—as will be discussed in the stress analysis section—the phase range becomes more faithful to the motion stimulus while the quality of the IA estimator is enhanced.

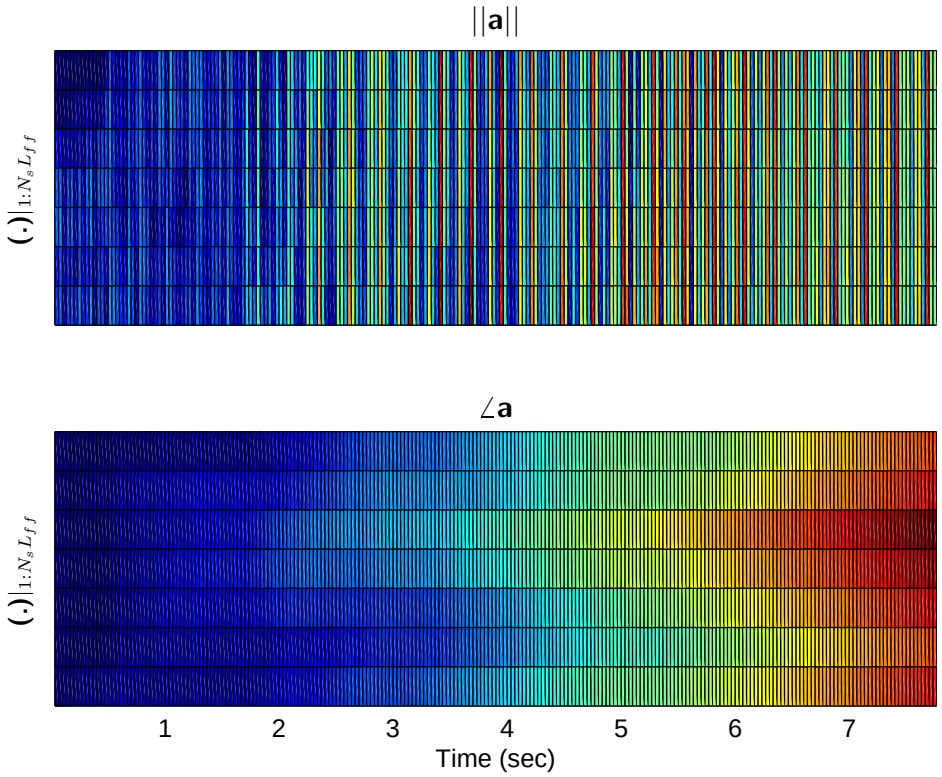
Phase Evolution

Unless otherwise stated, the instance at which the FF of the implicit DFE is deployed always equals to two codes.[6]

As discussed in the theory of operation section, the FF filter of the implicit DFE exhibits rapid tap translation and rotation in response to incoming Doppler levels. Thus, inspecting the evolution of the FF magnitude and phase provides considerable insights into the inner workings of its equalizing effect on Doppler-distorted chips. We, therefore, illustrate graphically the magnitude and phase evolution of the three aforementioned cases. Unless otherwise stated, the adaptive algorithm of the FF filter will be LMS for now. Additionally, the joint PLL performance will be contrasted against in order to better reason about the combined DFE-PLL adaptive mode.

In the first case of figure 6.14, both tap translation and rotation are very noticeable in the magnitude and phase evolution of all FF coefficients, respectively. Despite the low resolution of the observation rate, careful examination of the magnitude evolution reveals vertical colour gradient which is associated with the moving peak within FF's coefficient vector. As was discussed in the theory section, this indeed corresponds to the translation

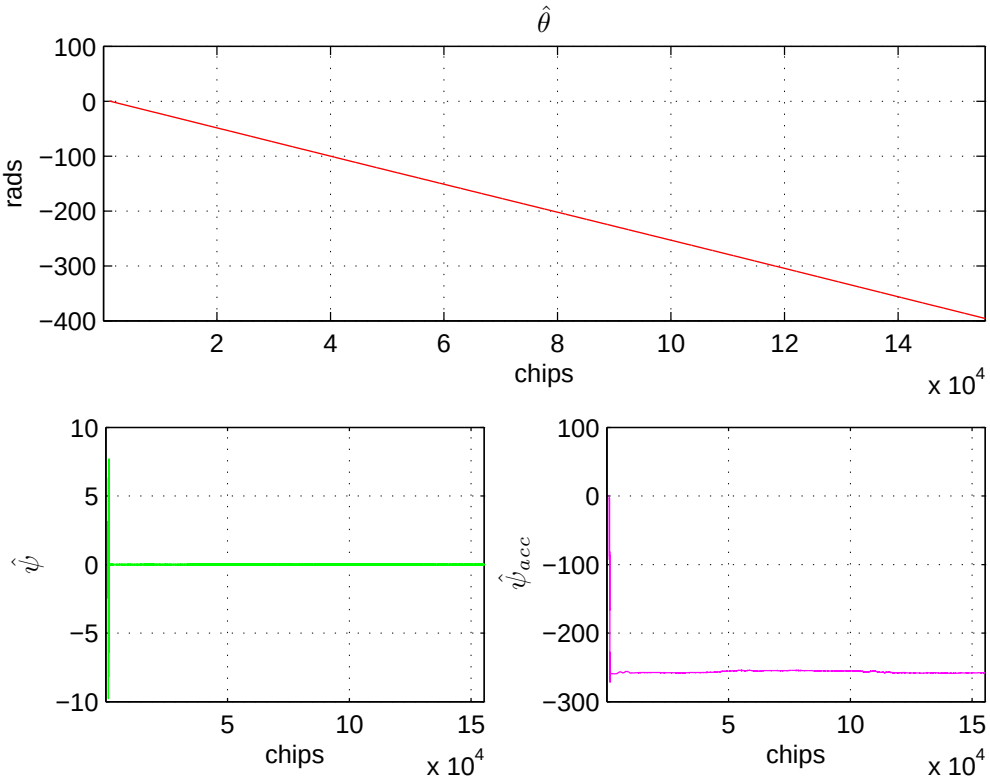
[6] This is controlled from within the algorithm by the constant FF_KICKIN.



[a] parameters: $N_s = 4$, $\mu_{LMS} = 1 \times 10^{-4}$, $L_{ff} = 2$, $K_1 = 1 \times 10^{-4}$, and $DRCE =$ FALSE
[b] generated with reduced observation rate of 30 Hz by decimating the full-rate data

Figure 6.14: Evolution of FF filter during DFE-PLL tracking, magnitude & phase, case 1[a]

effect. In regard to the phase, the horizontal colour gradient in the argument of FF's coefficient vector illustrates the rotation effect. It is clearer to link this with the tracking of motion in time, which is mostly dominated by FF in this case. This is further reinforced by the visibly idle ψ_{acc} of the joint PLL evolution in figure 6.15.



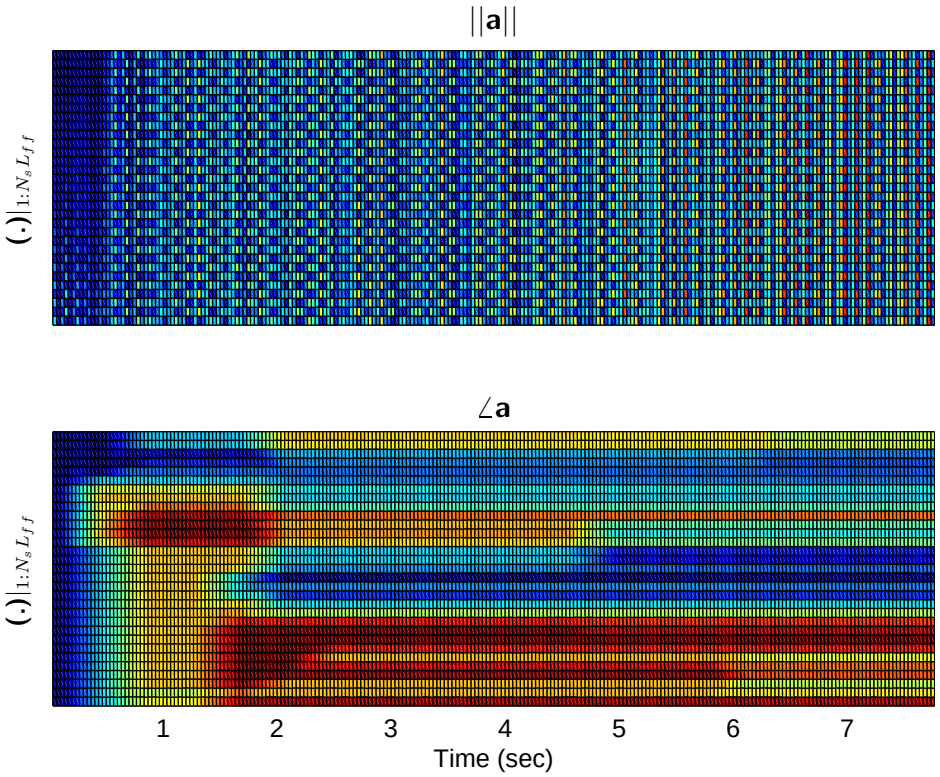
[a] parameters: $N_s = 4$, $\mu_{_{LMS}} = 1 \times 10^{-4}$, $L_{ff} = 2$, $K_1 = 1 \times 10^{-4}$, and $DRCE =$ FALSE

Figure 6.15: Evolution of PLL during DFE-PLL tracking, case 1[a]

In the remaining two cases, there is an increased distortion introduced by longer FF lengths. This affects FF phase trackability and allows the PLL to become more active. The notable finding in these two cases is that leaky adaptation (i.e. DRCE) is required in order for the instantaneous phase plots to be correlated with motion. The competitive DFE-PLL mode of operation under these conditions seems to remain stable as indicated by figures 6.12 & 6.13. The longer FF filters also remain well-behaved and bounded in magnitude throughout motion tracking.

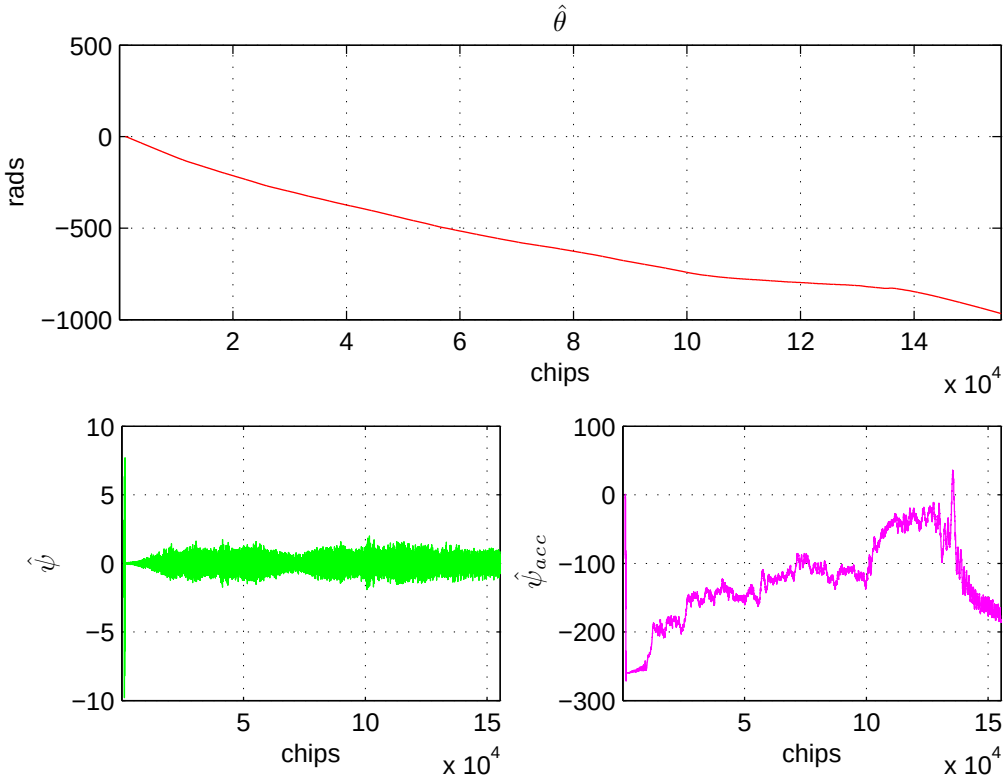
In 6.16, both tap translation and rotation are less visible, despite being active to varying extents—the translation effect is somewhat more evident than tap rotation. This can be explained by inspecting the joint PLL evolution in 6.17, and noting how the PLL is shouldering a considerable phase tracking role.

In the third case (figure 6.18), the two extra taps (multiplied by four for the fractional length) give the FF a slight advantage in phase trackability over the PLL when compared to the previous case. This can be seen in the decreased range of ψ_{acc} in figure 6.19. Consequently, tap translation and rotation are back to comparable levels to the first case (figure 6.14). Nonetheless, the PLL now plays an increased role when compared to the



[a] parameters: $N_s = 4$, $\mu_{_{LMS}} = 5 \times 10^{-4}$, $L_{ff} = 8$, $K_1 = 5 \times 10^{-4}$, and $DRCE = \texttt{TRUE}$
[b] generated with reduced observation rate of 30 Hz by decimating the full-rate data

Figure 6.16: Evolution of FF filter during DFE-PLL tracking, magnitude & phase, case 2[a]



[a] parameters: $N_s = 4$, $\mu_{LMS} = 5 \times 10^{-4}$, $L_{ff} = 8$, $K_1 = 5 \times 10^{-4}$, and $DRCE =$ TRUE

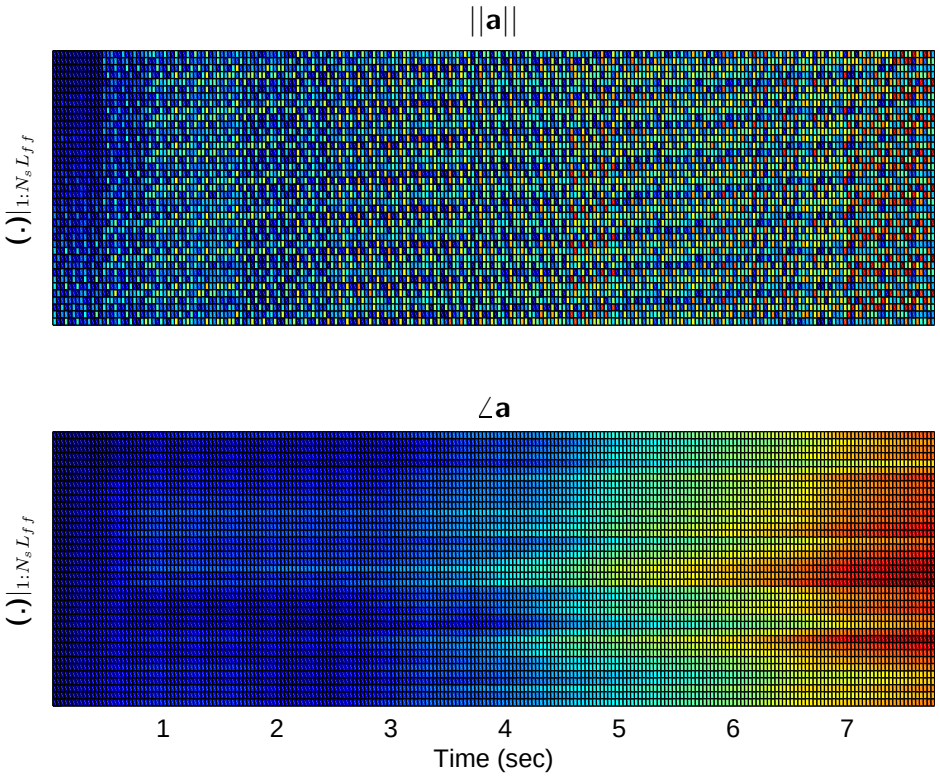
Figure 6.17: Evolution of PLL during DFE-PLL tracking, case 2[a]

first case, which is attributed to the weakened longer FF trackability. The scatter plot of the equalized chips corroborate this finding (see appendex B, figure B.6c for the chip estimate scatter of this case).

6.5.3 DFE-LI

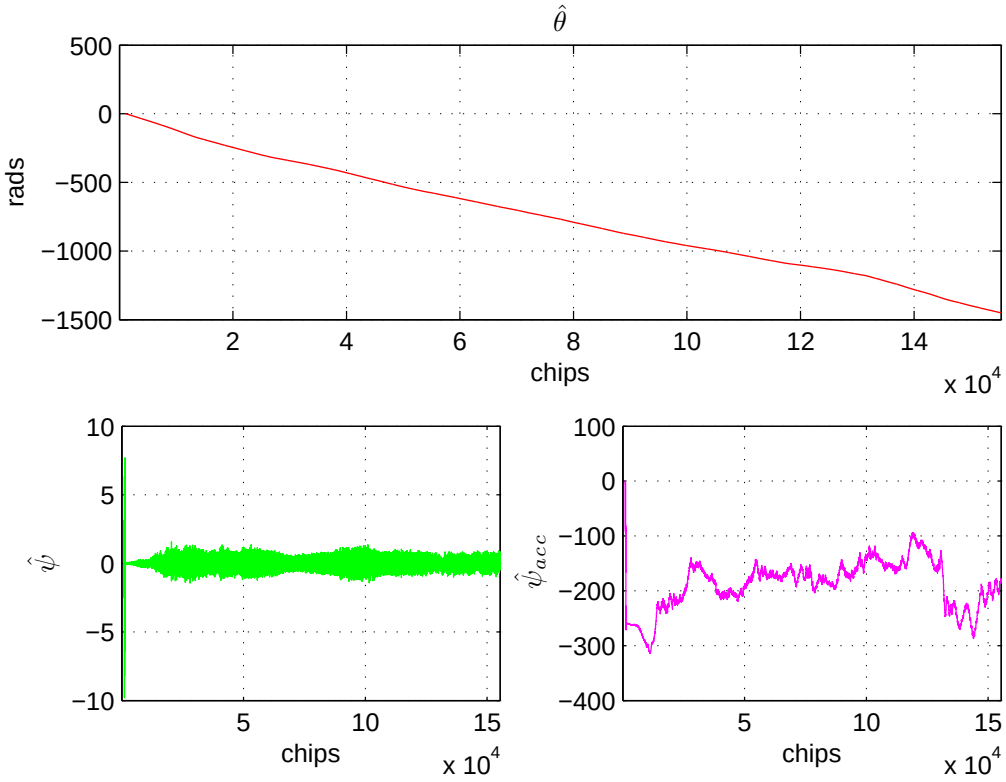
As previously established, the DFE-LI combination is a very viable candidate for the tracking of heavily Doppler-distorted ABU signals. That said, DFE-LI tracking is not without performance limitations. It will be shown that in the ABU band, the presence of severe high order motion moments can interfere with tracking and ultimately result in erroneous velocity measurements even though code timing itself is maintained. Therefore, the DFE-LI configuration is one that needs dynamic activation and de-activation, subject to tight high-order motion moment control.

In order to harness the increased phase trackability of the DFE-LI configuration, algorithmic primitives for the inference of high order motion moments have to be abstracted. These primitives may then be used for the dynamic switching on and off of the DFE-LI structure. For example, the instantaneous acceleration estimator that was derived earlier could be used to dynamically regulate the DFE-LI mode of operation. As such, IA would detect instances of acceleration and prevent overcompensation in phase, ensuring that the linear interpolation index returns to rest condition at the end of each motion leg. This will be discussed further, below.



[a] parameters: $N_s = 4$, $\mu_{LMS} = 5 \times 10^{-4}$, $L_{ff} = 10$, $K_1 = 5 \times 10^{-4}$, and $DRCE =$ TRUE
[b] generated with reduced observation rate of 30 Hz by decimating the full-rate data

Figure 6.18: Evolution of FF filter during DFE-PLL tracking, magnitude & phase, case 3[a]



[a] parameters: $N_s = 4$, $\mu_{_{LMS}} = 5 \times 10^{-4}$, $L_{ff} = 10$, $K_1 = 5 \times 10^{-4}$, and $DRCE =$ TRUE

Figure 6.19: Evolution of PLL during DFE-PLL tracking, case 3[a]

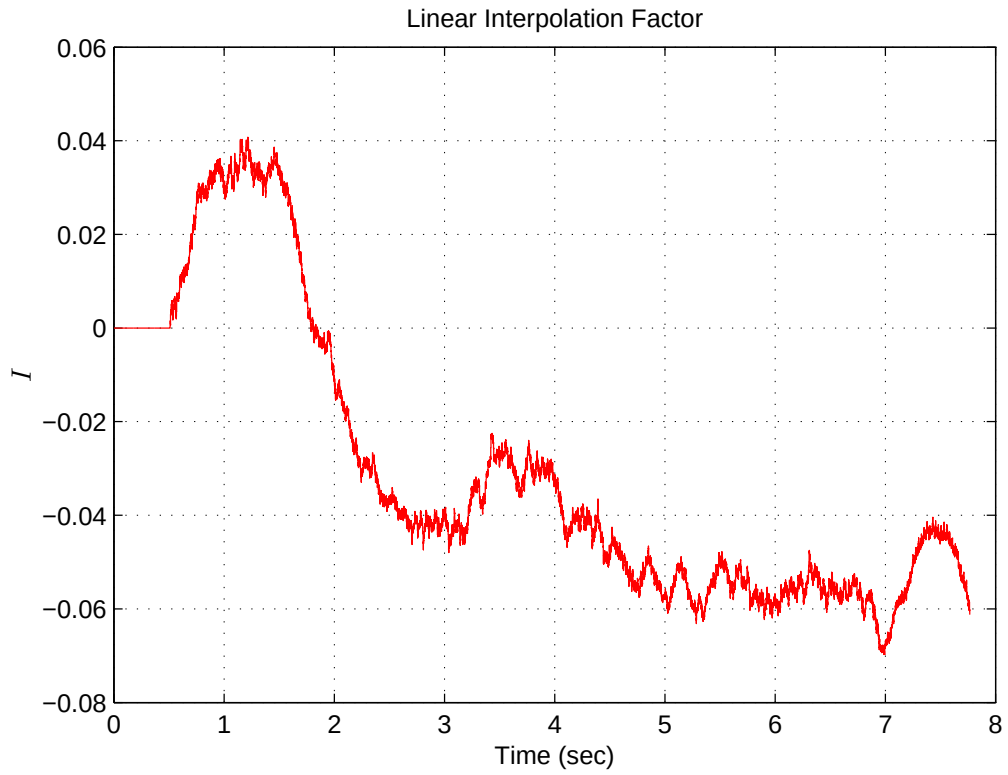
Test Setup

To expand on the subject of possible algorithmic clues to govern the run-time operation in the DFE-LI mode, a hard-coded example is presented to demonstrate the idea. Using a priori knowledge of the pattern of the Doppler stimulus, the initial acceleration and jerk of the robot with the mounted transmitter is avoided by delaying the instance at which the LI goes on-line. Thus, noting figure 6.5, an activation time[7] of $20L$ is conveniently chosen. This value corresponds to half a second. The resulting evolution of the linear interpolation index is shown in figure 6.20.

The Doppler stimulus of the test case has four forward and backward motion legs (figure 6.4a). The analysis of the performance of the DFE-LI mode as obtained from the setup explained above is given next.

- Initially: The linear interpolation index remains at zero until activated.
- Leg 1: During the first leg of motion, the linear interpolation index rises, commensurate with the increased Doppler levels. Upon deceleration, it then falls back, reaching zero when the robot is completely stationary.
- Leg 2: During the second leg of motion, reversal takes place and a negative swing interval commences accordingly. However, following the decelerating motion mo-

[7] This is controlled from within the algorithm by the constant INTERP_KICKIN.

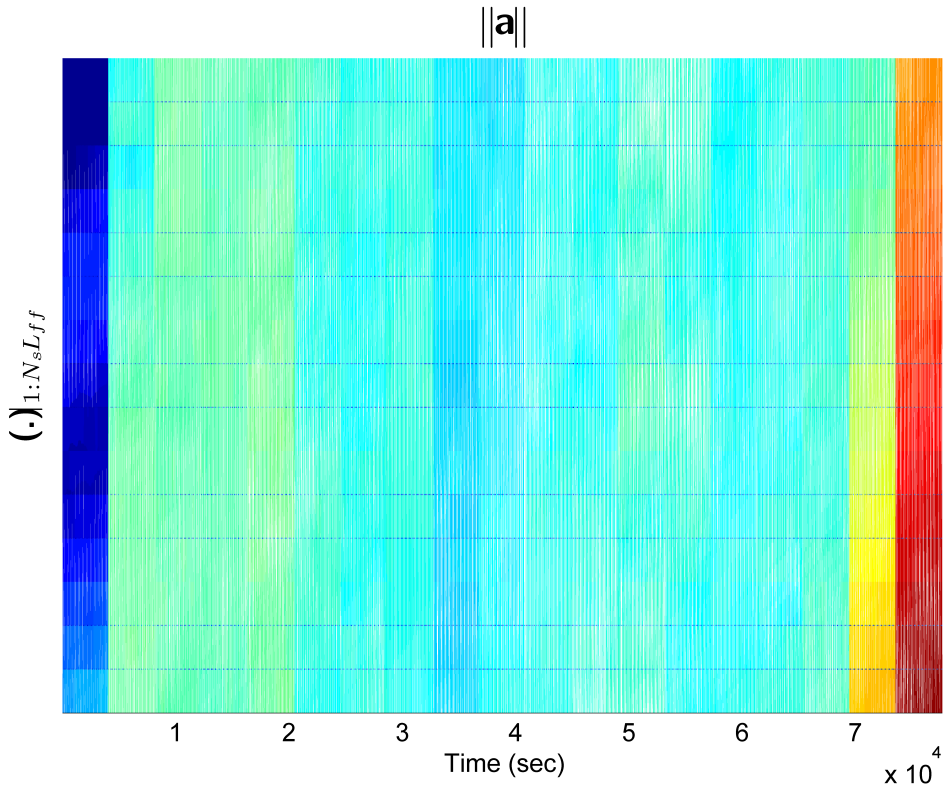


[b] parameters: $N_s = 2$, $\mu_{_{LMS}} = 1 \times 10^{-4}$, $L_{ff} = 8$, $K_p = 4 \times 10^{-5}$

Figure 6.20: Evolution of linear interpolation index

ments in the second leg, the linear interpolation index fails to return to rest condition at zero.

Both the positive and negative cycles are maintained. It is interesting to see that the transition between leg 1 and leg 2 went smoothly, in spite of the higher motion moments. It would seem that deceleration in the forward direction and acceleration in the backward direction contain less aggressive high order motion moments. This is however hard to ascertain with confidence, given the limited instantaneous accuracy of the vision-tracking utilized in the experimental setup. The rear-wheel drive layout of the robot with the mounted transmitter in the front may be one way to explain this observation—meaning, the weight of the motor will cause increased jerk when the robot breaks abruptly in the backward direction for instance. Following deceleration in the backward direction in leg 2, the linear interpolation index diverges away from rest condition. This is attributed to the FF intervening aggressively in the phase compensation, and consequently tipping the LI off balance. The evidence of this momentary amplification in the magnitude of FF can only be detected with high observation resolution. As shown in figure 6.21, a vertical stripe of sharper contrast occurs across all coefficients at the same time the linear interpolation index begins meandering away in the negative swing direction.



[b] parameters: $N_s = 2$, $\mu_{LMS} = 1 \times 10^{-4}$, $L_{ff} = 8$, $K_p = 4 \times 10^{-5}$
[a] rendered with OpenGL at the chip rate and 1200 dpi resolution

Figure 6.21: Evolution of FF magnitude[a]

Velocity Inference

In the DFE-LI mode, the linear estimation of velocity per motion leg is computed from the slope of the unwrapped phase of the chip estimate according to equations (6.55) & (6.57). The sign of the linear interpolation index per motion segment is obtained as

$$I_{sgn}(k) = \mathrm{sgn}\big(I(k)\big) \tag{6.69}$$

I_{sgn} is then used to delimit the motion segment and the velocity estimation is performed as

$$v_i(k) = \frac{c}{f_{carr}} \, \frac{1}{2\pi} \, \frac{\phi^u_{chip} - \phi^l_{chip}}{t^u - t^l} \tag{6.70}$$

where the superscripts ^{u} and ^{l} denote respectively the upper and lower bounds of the motion segment as determined by I_{sgn}.

The velocity estimate for the first motion segment of the case presented in figure 6.20 was measured to be 0.8047 m/s. This is the most accurate of all phase tracking mechanisms discussed so far. That said, it is the trickiest to tame within the context of a resilient ABU indoor operation having no prior assumptions or restrictions on high-order motion moments.

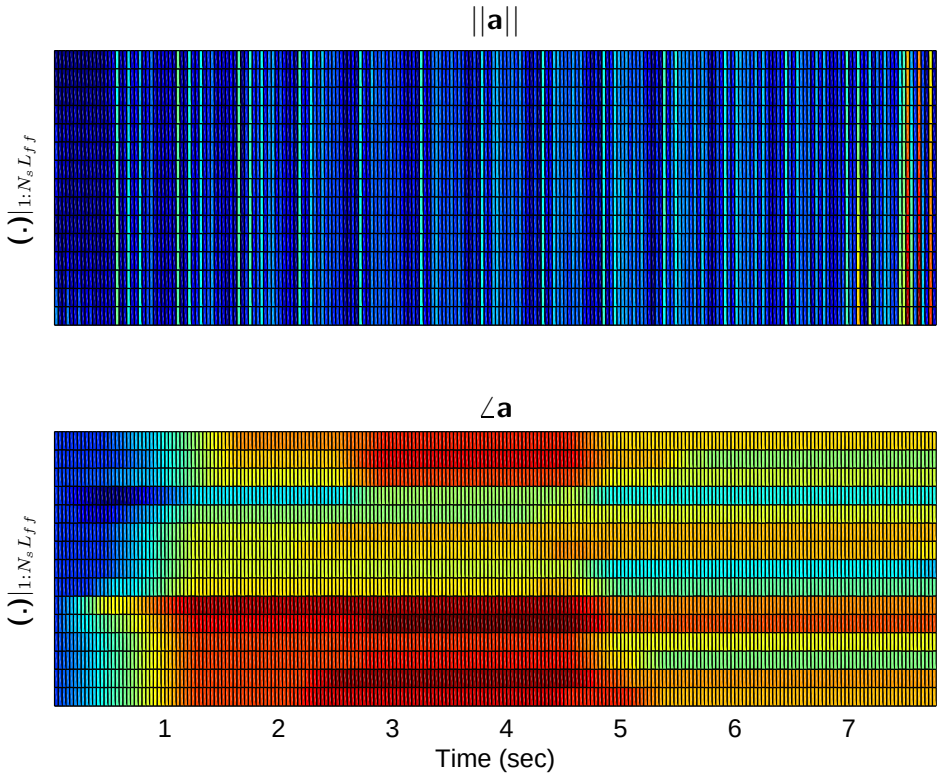
Phase Evolution

Figure 6.22 illustrates Doppler tracking influence on FF's evolution in the DFE-LI mode for the test case discussed above. In contrast to the DFE-PLL mode, the translation effect in the peak of FF's magnitude coefficient vector is very minimal. Meaning, no serious Doppler tracking is being performed by the FF, rather residual distortion is what FF seems to be reacting to. Similarly, inspecting the unwrapped phase of the FF's coefficient vector reveals that while minimal tap rotation does take place at certain instances during tracking, the phase henceforth is being held constant until further disturbance is encountered again.

6.5.4 Stress Analysis

It is very informative to attempt to formulate an algorithmic primitive for the inference of dynamic stress conditions that ABU signals undergo in human-scale indoor movement. Not only could these primitives potentially enrich the sensing model, but also they may be utilized to achieve tighter run-time algorithmic control, boosting robustness and allowing for a wider array of usage scenarios.

Following the earlier derivation and equations (6.62) & (6.63), the imaginary part of the despreader, alongside its derivative are presented next. The cases which resulted



[a] parameters: $N_s = 2$, $\mu_{_{LMS}} = 1 \times 10^{-4}$, $L_{ff} = 8$, $K_p = 4 \times 10^{-5}$
[b] generated with reduced observation rate of 30 Hz by decimating the full-rate data

Figure 6.22: Evolution of FF filter during DFE-LI tracking

in motion-correlated chip estimate phase will be analyzed, namely for the DFE-PLL and DFE-LI modes.

DFE-PLL

As illustrated earlier, there are three sets of configurations in the competitive DFE-PLL mode with instantaneous phase pattern exhibiting visible motion-induced variations.

The first case (figure 6.11) is qualitatively the least representative of motion. We reason that the relatively short FF length and the weak PLL drive are at the boundary of phase tracking, as can be further inferred from the relatively unfaithful range of the scaled phase. It therefore comes as no surprise that the corresponding IA estimator as derived from the imaginary part of the running despreader would be equally unfavourable. As can be seen in figure 6.23, the IA estimator shows very little evidence of second-order motion moments. However, few spikes indicative of the initial motion disturbance are present.

The second case with increased FF length (figure 6.12) is better representative of the Doppler stimulus, and clear correlation with motion can be seen. In conjunction with instantaneous chip estimate phase, the IA estimator (figure 6.24) seems to be better behaved—it contains clear peaks around the acceleration and deceleration instances during the course of the robot movement. However, it seems to oscillate significantly at the end, flagging a borderline despreader phase stability problem. This can be further seen by inspecting figure B.4a and noticing the MSE of the running despreader.

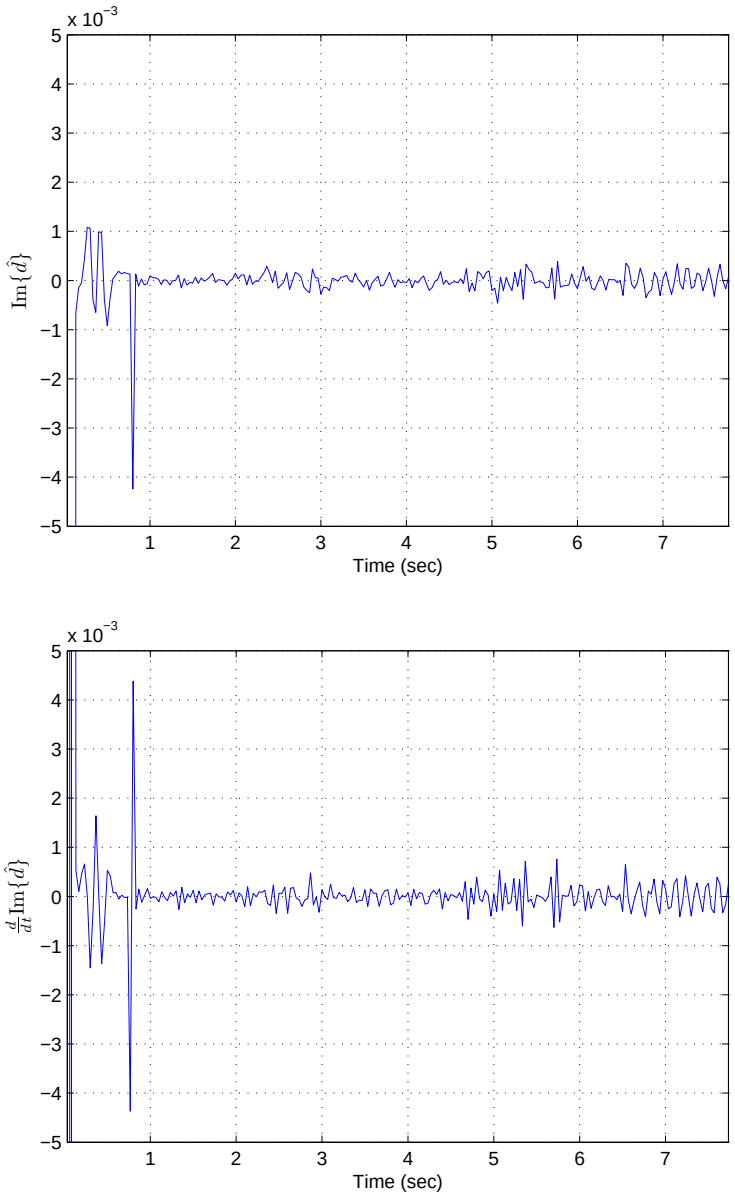
The third case (figure 6.13) with the further increased FF length is best in terms of similarity with both velocity and acceleration groundtruths. The instantaneous phase exhibits strong correlation with motion. Additionally, the IA estimator of figure 6.25 emphasizes all time instances at which second-order motion moments are expected. It also seems to cope better with the end of the motion stimulus.

DFE-LI

In the DFE-LI mode, the IA estimator is given in figure 6.26. Despite having reduced range to that of the DFE-PLL mode, the IA estimator is still strongly correlated with second-order motion moments. Moreover, it is better behaved than the IA in the DFE-PLL mode in terms of the end of stimulus response, as can be inferred from the absence of significant tail oscillations.

IA Overview

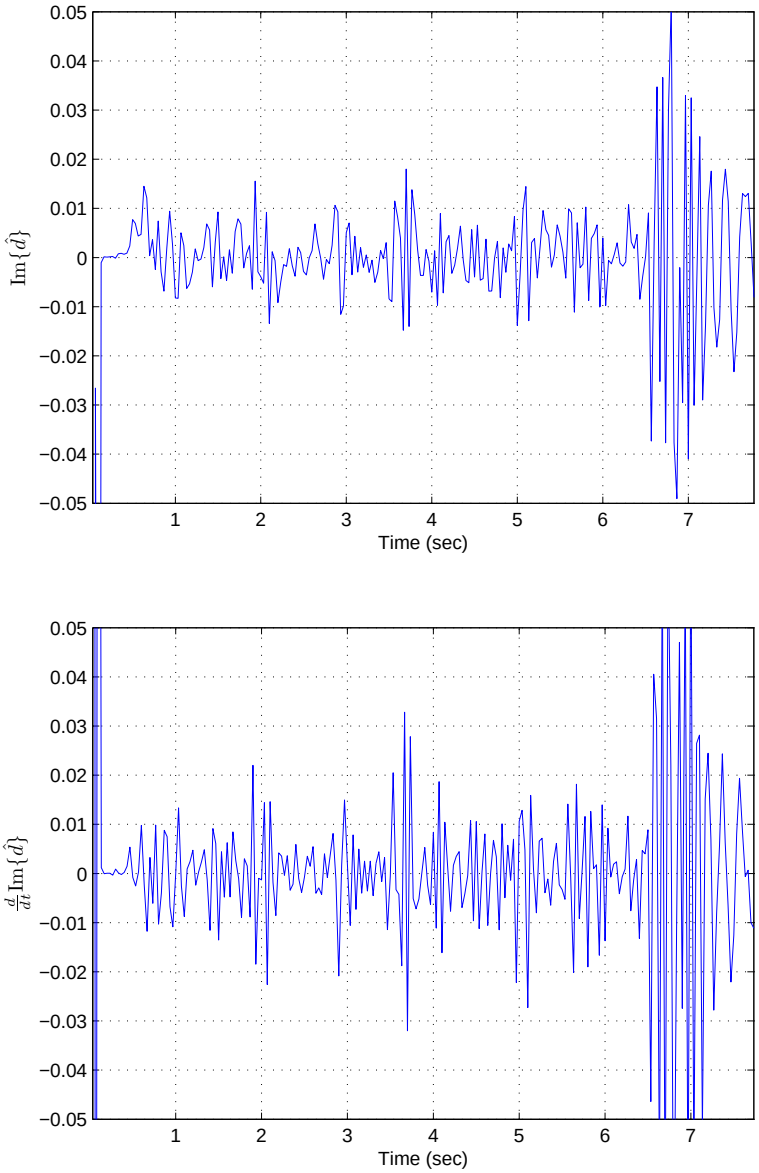
For the last two estimators—case 3 DFE-PLL & DFE-LI—inspecting the derivative of the imaginary part of the despreader is instructive; it tends to spike significantly in and around the instances in time when motion commences or ceases. This suggests that the imaginary part of the despreader is indeed correlated with high order motion stresses. As discussed in the theory section, the code-length averaging provided by the running despreader means also that additional processing (e.g. statistical weighting) may be needed



[a] parameters: $N_s = 4$, $\mu_{LMS} = 1 \times 10^{-4}$, $L_{ff} = 2$, $K_1 = 1 \times 10^{-4}$, and $DRCE =$ FALSE

[b] generated with reduced observation rate of 30 Hz by decimating the full-rate data

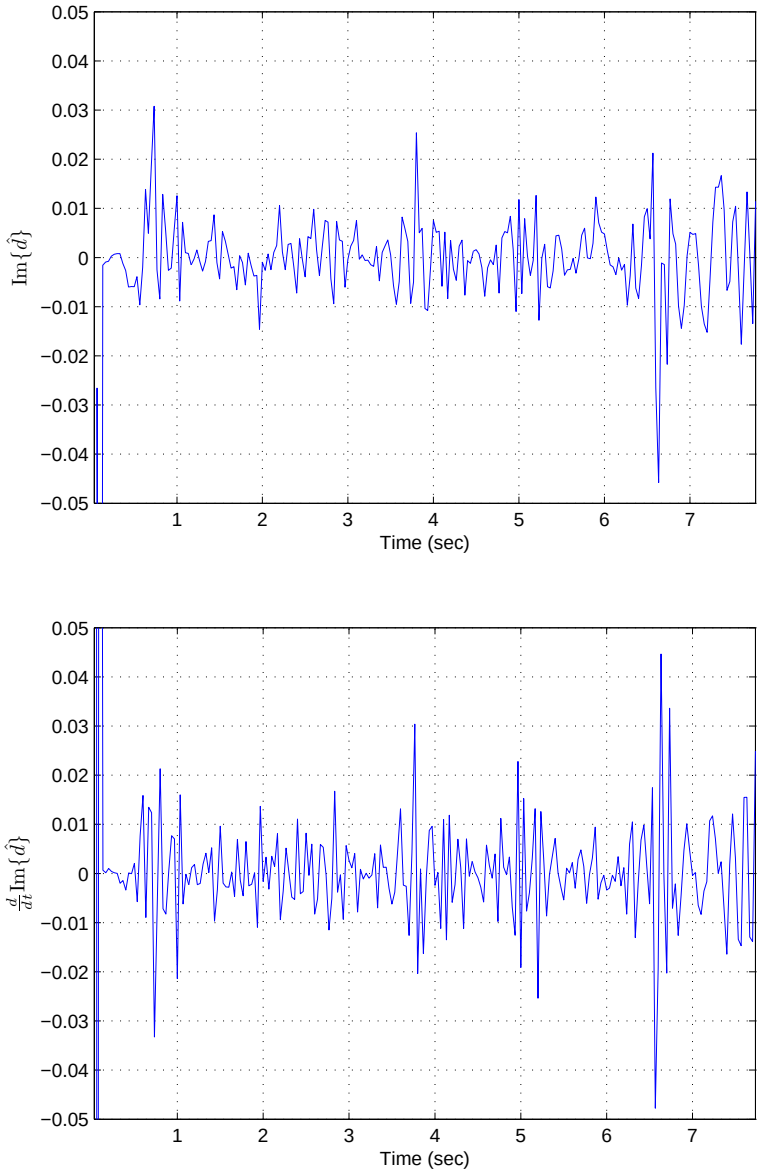
Figure 6.23: Instantaneous Acceleration estimator during case 1[a] DFE-PLL tracking



[a] parameters: $N_s = 4$, $\mu_{LMS} = 5 \times 10^{-4}$, $L_{ff} = 8$, $K_1 = 5 \times 10^{-4}$, and $DRCE =$ TRUE

[b] generated with reduced observation rate of 30 Hz by decimating the full-rate data

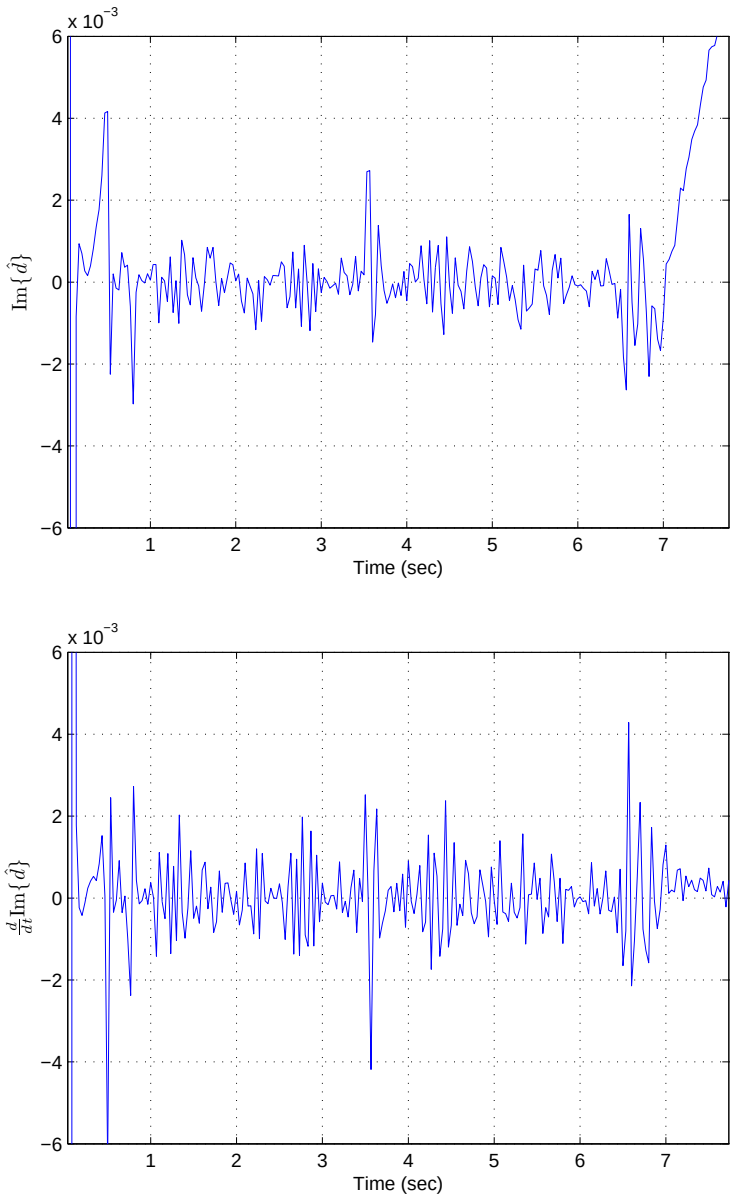
Figure 6.24: Instantaneous Acceleration estimator during case 2[a] DFE-PLL tracking



[a] parameters: $N_s = 4$, $\mu_{LMS} = 5 \times 10^{-4}$, $L_{ff} = 10$, $K_1 = 5 \times 10^{-4}$, and $DRCE =$ TRUE

[b] generated with reduced observation rate of 30 Hz by decimating the full-rate data

Figure 6.25: Instantaneous Acceleration estimator during case 3[a] DFE-PLL tracking



[a] parameters: $N_s = 2$, $\mu_{_{LMS}} = 1 \times 10^{-4}$, $L_{ff} = 8$, $K_p = 4 \times 10^{-5}$
[b] generated with reduced observation rate of 30 Hz by decimating the full-rate data

Figure 6.26: Instantaneous Acceleration clue during DFE-LI tracking[a]

in order to convert this algorithmic entity into an instantaneous acceleration (m/s^2) estimator. Otherwise, for the sake of algorithmic control only, this entity can be used "as is" (unitless) provided that a comprehensive understanding of its behaviour whilst under high-order motion stresses exists.

In specific terms, distinct sustained spikes can be immediately spotted in three places: at the beginning of the Doppler stimulus around 0.5 sec; around the momentary pause of the robot at roughly 3.5 sec between leg 2 & leg 3; and at the end of the test case slightly before 7 sec.

This empirical evidence of a run-time IA estimator presents a promising lead towards the realization of motion resilience in ABU, irrespective of high-order motion moments. This estimator can be used to realize a form of phase "handover", possibly utilizing control theory, as to lessen the impact of high-order motion moments on phase tracking.

6.6 Summary

In this chapter, a novel acquisition and tracking algorithm has been derived. Special purpose-built adaptation techniques are proposed, most notably the DRCE mechanism which was shown to have a direct impact on performance. The theory of motion tracking in the novel ABU band is thoroughly developed. Additionally, empirical evaluation of the proposed adaptive algorithm is carried out. The empirical study of ABU operation is concluded by proposing the dynamic use of a high-order motion moments estimator. This estimator is aimed at achieving motion inference resilience in the ABU band with no prior assumption on operational conditions.

PART III

HIGH-LEVEL SYNTHESIS EXPLORATION

CHAPTER 7

Preliminaries

This chapter briefly goes through the hardware design practices utilized in the designs presented in this part of the thesis. An overview and justification of high-level synthesis (HLS) is first given. Then the HLS signal processing design flow is discussed.

7.1 Background

In scope-specific embedded applications (e.g. audio, protocol processing, etc.), general-purpose processors' limited performance precludes their use to meet the multidimensional requirements of these systems [99]. Hypothetically, there exists a super fast general-purpose microprocessor that can cope with the data-intensive tasks of these applications, at the expense of power, form factor, and cost. Instead, developers have traditionally utilized RTL designs to "exploit the intrinsic parallelism of many data-intensive problems...to achieve tens of hundreds of times the performance achieved by a general purpose processor." [99, p. 152–154] The caveat associated with classic RTL, however, is the inherent inflexibility. Hence there is a need for a general hardware/software codesign methodology to strike a balance between generality and implementation efficiency. Historically, this has been accomplished by capitalizing on RTL's *performance* while using software to *flexibly* integrate the overall system. For example, after hardware correlators have detected valid peaks, software can combine the time stamps to produce an estimate of the angle-of-arrival.

It is foreseen that all variants of current SoC technologies and methodologies be reduced to a single hardware/software co-design problem utilizing future platform FPGAs [71] [74, p. 7]. The term platform refers to a heterogeneous fabric architecture that contains many special-purpose blocks such as MACC engines, processor cores, serial transceivers. In

the broader sense, it is not clear yet what the enabling process technology would be (e.g. SRAM, FLASH, hybrid). However, the target technology would be a function of many factors such as production volume, level of flexibility, performance (measured in clock rate), power consumption, cost, etc.

Sun Microsystems terms the methodology by which a design is executed throughout the entire flow by one team as “construct by correction” [74, p. 15-16]. By doing so the UltraSPARC team was able to see the impact of their architectural decisions on area, power, and performance. The UltraSPARC development project was one of the most successful in Sun Microsystem’s history. The same concept of “construct by correction” can be shifted one layer up to overlay algorithms and architectures (as opposed to architecture and floorplanning in Sun’s approach) as to allow exploring the cost/performance tradeoffs for various combinations (algorithms and architectures) by way of judgment and experience. Austin et al. [24] call this process “joint optimizations” and adds

> To build practical mobile supercomputers, system architects need to jointly optimize across algorithms, architectures, and circuits. We don’t have all the answers today about how to solve all the problems inherent in mobile supercomputing, but we believe that we have identified some useful approaches. We can control tradeoffs in vertically integrated manner:
>
> - microarchitectures that can take advantage of advanced circuit features,
> - programs that automatically exploit application-specific architectures, and
> - software to glue the layers together and allow existing off-the-shelf applications to use the system efficiently [24, p. 83].

7.2 State-of-the-art

The mature field of hardware/software codesign [147] has recently converged to the so-called embedded system-level (ESL) design [32]. ESL is defined as [87, p. 20] “the utilization of appropriate abstractions in order to increase comprehension about a system, and to enchance the probability of a successful implementation of functionality in a cost-effective manner, while meeting necessary constraints.” Today, EDA tools exist to facilitate the exploration and optimization of a given design (e.g. written in a sequential software programming language) with judicious intervention from the informed user in order to better guide the automated process. The incorporation of domain-specific expertise [39] into the mapping of functionality and vice versa is the best known method for obtaining optimal results in the increasingly complex application space. A related subbranch to this grander endeavour is Multiprocessor System-on-Chip (MPSoC) [86] wherein intensive computations must also meet real-time, low-power, low-cost constraints [148]. Platform-oriented ESL design flow methodologies can address many of the diverse requirements seen in emerging computationally-intensive applications [151]. Many of these design flows have come to rely on the widely-embraced high-level languages (HLLs) such as C language variants in order to raise the abstraction level and enhance productivity [103].

The process by which HLLs are utilized for hardware description is termed high-level synthesis (HLS).

In [23], Arvind et al. make the case for HLS. By means of HLS, an automated evaluation of different architectural options (a.k.a exploration) for a given design while working at a higher abstraction level is performed. The need for HLS is traced to three trends in the stat-of-the-art [23].

1. The increasingly complex application space is giving rise to correspondingly complex ASICs, which are expensive, difficult, and risky to produce.

2. Architectural exploration is hard under traditional methodologies. This is because gauging the impact of alternative system-level decisions is impractical. For example, while adding/removing datapaths and memories is relatively straightforward, the associated redesign and reverification of control is not.

3. The pressure from increasing complexity calls for a “correctness by construction” methodology wherein: (1) the exploring of a large architectural design space is performed automatically, (2) the resultant quality is comparable to hand-crafted designs.

A case study of such microarchitectural exploration is provided in [41]. Dave et al. supply us with a set of remarks in order to motivate the wide adoption of an HLS design flow.

1. There is an ever present need for maximum energy efficiency ruling out the possibility of a purely software-based implementation given the impractically high clock rate required.

2. Two contending practices are parallelization and folding. While parallelization will lower power consumption by virtue of decreased required clock frequency, it will also duplicate hardware and increase area. Folding, on the other hand, will allow for module reuse but will also require higher operating frequency.

3. There is no a priori knowledge of what is required in order to realize a system with a certain set of objectives. As such, architectural investigation of the entire design space is needed to arrive at a compromise between area, power, and cost, particular to the design’s set of objectives.

4. If non-trivial parametrization were to be conducted, the architectural exploration of the design at hand becomes significantly more tedious without the use of a high-level methodology.

5. Under such a methodology, insights into the the cost-area-power tradeoffs of a design are afforded *early* on with respect to the design process. This will present *empirical* evidence on the feasibility of the design, rather than relying on mere intuition.

On a more in-depth take on the use of high-level methodologies, Buyukkurt et al. [34] single out streaming applications, such as signal processing, as a class of domains on which high-level approaches are particularly effective. At the crux of this design flow are extensive compile-time transformations (a.k.a directives) which would be deemed too complex for a human user to implement in a timely manner.

Papakonstantinou et al. makes a comprehensive case for HLS being a viable emerging tool to replace traditional RTL, especially for design space exploration [103]. Additionally, references therein [103] utilize the methodology for limited exploration to crudely estimate execution cycles and area. This is because slow synthesis and place-and-route (PAR) often inhibit more thorough characterization.

7.3 The point

When faced with the problem of investigating the feasibility of a computationally-intensive algorithm with further application-mandated constraints—real-time, area, and power—traditional methodologies fall short of being effective given a tight completion timeline. Additionally, the very nature of the uncertainty surrounding the investigation poses extra barriers on the likelihood of converging on an acceptable solution in a vast unexplored design space. The automation of this process coupled with acute understanding of the algorithmic workload are indispensable prerequisites for tackling a problem of such specific nature. Emerging ESL design flows hold the promise of agile algorithmic architectural co-exploration, with guaranteed quality of results. These design methodologies are likely to increase in importance as we witness an era of boosted levels of sophistication in various application spaces.

7.4 HLS Design flow

There are various commercial HLS tools already available. SystemC [13] is one tool popular for a unified representation of hardware and software, but is viewed as lacking for the next generation hop in raising the level of abstraction of complex systems [149]. This is because SystemC does not hide enough details from the programmer in comparison to say untimed C. The mainstream adoption of HLS is contingent on the ability to express designs at the highest possible abstraction level, such that clear advantages over current methodologies are obtained. Despite being designed to support parallel hardware constructs, non-C languages have not gained traction in the HSL market, such as Bluespec by Bluespec Inc [2]. The industry seems to be more in favour of C/C++ dialects because they offer clear advantages in terms of: (1) the promise of the highest level of algorithmic exploration though advanced parallelizing compilations, and (2) being able to leverage stable open source tools for simulation. Many C-based tools are available from a number of vendors [3, 12]. Some tools require special coding practices such as Mitrion-C [9] and ImpulseC [6]. For the design exploration in this disseration, the C-based HLS tool AutoPi-

lot[1] from Xilinx is chosen because: (1) its ability to piggyback onto the Xilinx downstream ISE tool suite which is already available under the Xilinx university program (XUP), and (2) being an industry leader with good testimonial track record of quality of results.[2]

AutoPilot is a platform-based tool which incorporate device-specific architecture characterization into the high-level compilation process. The HLS design flow using AutoPilot and Xilinx ISE tool suite for back-end synthesis and place-and-route is shown in figure 7.1.

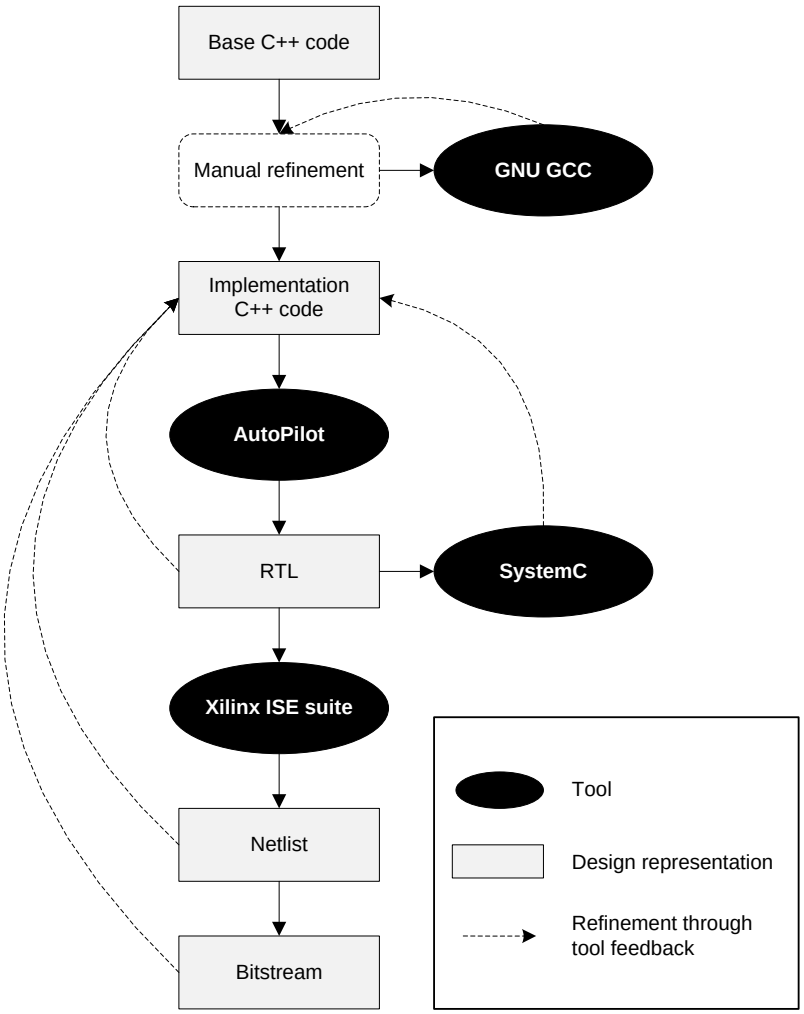


Figure 7.1: Design flow using AutoPilot HLS with Xilinx's ISE tool suite

The base floating-point C++ code is refined using the supplied arbitrary precision API. This process is manual and the GNU GCC compiler is utilized to bring the refined code to a first pass without having to go all the way down in the hardware tool chain. Once the implementation code has been arrived at, AutoPilot compilations are used for C-to-HDL translation. A second pass of refinements is required at this stage to remedy any undetected sequential constructs. The HDL code generated by AutoPilot is then synthesized using the ISE tool suite and further iterations occur. At this point, design exploration begins with the aid of the powerful high-level directives set available. Sporadically, place-and-route (PAR) is conducted to sanity-check the intermediate results. PAR is a very

[1] Formerly from AutoESL.

[2] For instance, see `http://www.deepchip.com/`.

lengthy process and its impractical to perform it on all design exploration branches. It is, however, important to PAR a design after initial satisfaction by synthesis results is reached. As will be vindicated by certain findings from chapter 8, this ensures that any unfavourable design decisions that have slipped past synthesis undetected are eventually flagged. Iterations continue until the final design space exploration concludes once satisfactory results have been arrived at, meeting all design objectives.

CHAPTER 8

FPGA-Based Real-Time Receiver

This chapter conducts a thorough architectural exploration on the algorithmic finding of chapter 6 in order to assess the feasibility of an embedded, real-time, motion tracker in the ABU band. Section 8.1 discusses the rationale behind the work and the approach taken. Section 8.2 describes the general concepts surrounding the design in the context of an HLS methodology. Section 8.3 outlines the hierarchical structure of the design components. Section 8.4 goes through the fine details of the receiver's algorithmic operations, and illustrates how the design rationale presented earlier in section 8.1 is implemented. Section 8.5 analyzes the obtained results in terms of four metrics: area, power, throughput, and order of execution. The block-floating point numerical format is then introduced in section 8.6, and its implications on the system operation are examined. Section 8.7 comments on the representativeness of the obtained results in relation to the ABU tracking feasibility. Finally, results are summarized and a conclusion is drawn in section 8.8.

8.1 Introduction

This chapter leverages a state-of-the-art ESL design methodology in order to progressively arrive at an optimized real-time hardware realization for the proposed adaptive algorithm. The aim is to demonstrate the immediate feasibility of such a receiver for modern DSP-capable reconfigurable fabrics i.e. FPGAs. Fixed-point computations are crucial for a battery-powered handheld operation necessary for mobile computing applications. Due to the non-stationarity of the Doppler phenomenon and the adopted non-linear equalization, manual optimization closure has to be iteratively pursued which is formally dubbed *simulation driven analysis* in [31, p. 201]. In a typical approach for *time-varying* and *recursive* systems, comprehensive testing is conducted in order to arrive at bounds for the

peak values of signals. These bounds are then multiplied by a “safety factor” to guarantee no occurrence any overflow in signal datatypes. Hence the design is ultimately dependent on exhaustive testing whilst under the range of velocity-induced Doppler stresses likely to arise in real-world scenarios.

The design in its base floating-point and fixed-point versions is written all in templatized C++. A C++ HLS description of the design allows for platform-independent portability and ease of maintenance. Such approach not only comes with a powerful set of automated architectural optimizations through high-level directives, but also enables rapid joint algorithmic-architectural evaluation with unprecedented agility [151]. The challenge, however, is to—besides meeting adaptation deadlines and required precision—manually develop a custom spatial algorithmic form with scalable memory and processing as to account for various system parameters. This involves refining the templatized C++ code against the base sequential algorithm while accommodating its natural data propagation until memory access bottlenecks and latency constraints have been overcome and satisfied.

The chapter is also aimed at dispelling the common belief held by Ubicomp and Pervasive Computing communities that complex signal processing is inaccessible, and as such one need not strive to realize sophisticated sensing devices.

8.2 FPGA-Based Adaptive Receiver

This section describes the experience of utilizing an HLS tool for C++ to RTL generation. First, an introductory rationalization of the approach taken in the design will be given. In what follows, an overview of hardware design issues specific to the algorithmic workload is presented.

8.2.1 Chip Oversampling

In summing up the spirit of the proposed adaptive algorithm, spreading code vectors' worth of oversampled chips are fed to the custom computational datapath. These entities can be an oversampled chip, a reconstructed oversampled chip, or a fractional channel coefficient. The algorithm consists largely of vector arithmetics at the DSSS length and convolution loops at the code length applied to fractional entities. With 20 kHz chip rate and a nominal digital system clock of 100 MHz, the maximum clock budget per adaptation step is 5000 cycles. While this is quite long in digital terms, the number of loops that have to be performed on the oversampled code length vectors can easily exceed a chip duration. This is because a large proportion of adaptation is purely sequential in nature entailing vector data dependencies. Nonetheless, the one commonality among all computations is that operations are replicated on chip oversamples. Thus a form of data-level parallelism underlying processing is readily achievable by the concurrent manipulation of a stream of chip oversamples. To this end, as will be further explained in subsequent sections, all loops are partially unrolled by an oversampling factor while memories are

partitioned cyclically by the same factor as to be able to service the chip-wide computational datapath. Such an arrangement is abstracted for a generic MACC operation in figure 8.1.

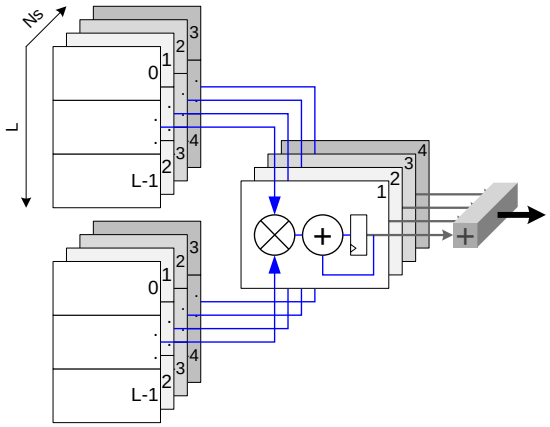


Figure 8.1: Abstract 3D (space-time) MACC using HLS memory partitioning and loop unrolling

8.2.2 Parallelism

The algorithm exhibits a strong form of data-level parallelism determined by the chip oversampling rate. However, in order to tap into this underlying algorithmic uniformity, a few other considerations mandated by the processing workload have to be met, such as expression-balancing some timing-critical operations. This rules out the possibility of using a generic SIMD DSP for realizing a computationally-intensive algorithm of such specific nature, while having equally stringent application requirements on form factor and power consumption. This is because an SIMD realization is likely to require faster clock rate in order to make up for any associated software latencies. At the same time, generic SIMDs will have increased power consumption. Chip area, power consumption, and form factor are requirements particularly important for mobile scenarios, which historically have been met using custom silicon. The advent of low-power FPGAs provides a much low-cost, low-risk alternative as discussed earlier.

Both task parallelism and instruction parallelism are automatically handled by HLS. Since the early introduction of HLS as a viable design methodology, more credible automation can now extract functional and instruction parallelisms seamlessly, albeit to a certain degrees. Nonetheless, judicious intervention from an expert user allows for added parallelism subject to broader design space investigation for various trade-offs (such as area, speed, etc) by means of which the final solution is steered.

For the real-time adaptive application at hand, boosted levels of instruction parallelism have been further extracted by pipelining the numerous loops that constitute the algorithm. In addition, the original software code underwent a major refinement with respect to instruction parallelism. This involved visualizing a classic pipeline in mind and rewriting the untimed, sequential C++ code as to reflect a pipelined operation with prefetch, execute, writeback if applicable, and update. This is important so that the pipeline initiation

interval (II) remains at one even though the pipeline depth itself could be as high as 30 in certain loops, hence resulting in no pipeline stalls. In such a case, after a small initial ramp-up latency (e.g. 30 clock cycles), the throughput of a loop would be maximized at one operation per clock cycle.

Another remark pertaining to parallelism is the following. Global parameters were loaded in variables local to loops or functions prior to proceeding with computations. This is necessary for maximum throughput in these rather long loops. Failure to do so results in long pipeline stalls since the tool effectively translates every occurrence of a global parameter into a function call, which greatly degrades operation scheduling.

Also linked to optimal pipelining is expression balancing in loops. This is a factor that impacts area greatly too; especially, in loops containing complex multiplications which are further partially unrolled. One observation that might be counter-intuitive here is the following. Attempting to conserve hard MACC slices using less partial unrolling at the expense of increased unrolling of MACC-free loops will degrade timing and consequently compromise achievable adaptation rate. MACC slices are optimized for performance whereas utilizing generic fabric in serious computations will worsen the critical path of the synthesized logic.

8.2.3 Memory

Partitioning

Exploiting the underlying data-level parallelism necessitates a multiport memory architecture as to increase data throughput. In order to read out a complete oversampled chip from the signal vectors in one clock cycle, memories were cyclically partitioned by the chip oversampling rate. This is done without introducing any modifications to the code by applying high-level directives. Cyclic partitioning of a signal vector results in a number of equivalent smaller vectors whose entries are the parent's entries interleaved across the now increased memory ports by the required factor. This uniform partitioning is applied to all DSSS vectors; circular and linear. There are: the signal vector $\mathbf{v}(k)$, the average signal vector $\hat{\bar{\mathbf{v}}}(k)$, the interference-free vector $\hat{\mathbf{v}}_0(k)$, and the channel vector $\hat{\mathbf{h}}(k)$. Furthermore, both the Gold code sequence q and its estimate $\hat{q}$ are similarly partitioned. This is done for a different reason as will be discussed in the following section. Finally, since the feedforward filter is targeted towards chips, it is much shorter and as such does not require partitioning.

Consistent addressing

It is required that the memory access pattern for various vectors be consistent throughout the algorithm. This entailed refining the original code such that addressing these memories becomes unified as either high-to-low or low-to-high. This aids streaming of data from one function/loop to another and is of particular importance when vector data dependencies occur. Abiding by this consistent access pattern ensures that a datum entry can be immediately streamed to where it is next required as opposed to having to wait

on the whole vector operation. The use of the indices `high_ndx` & `low_ndx` on page 121 illustrates how this is accomplished in C++.

Contention

access stall Various functions combine to make a heavy use of the two circular signal vectors; namely, $\mathbf{v}(k)$ and $\hat{\mathbf{v}}(k)$. Aggravated by vector data dependencies at times, this gives rise to access bottlenecks due to multiplexing and the limited memory ports available coupled with operation rescheduling and/or retiming, which the tool also automatically performs part of a global optimization problem. These access bottlenecks resulted in pipeline initiation interval (II) of value 2 for two loops in the adaptive algorithm; effectively, doubling their latencies. Initially, this was mitigated against by fully partitioning the two circular signal vectors into registers. In the fully partitioned architecture, the circular vectors have as many ports as entries. This is obviously of tremendous generic fabric cost not to mention the very long decode logic delays on these rather deep DSSS-length vectors. Synthesis had proceeded successfully initially. It was not until placing-and-routing the design that failure to meet timing objectives under the fully partitioning high-level directive surfaced. Therefore, memory access bottlenecks and subsequently pipeline stalls on the two circular signal vectors had to be tolerated for these two loops and were deemed unavoidable as opposed to going fully partitioned. This is only applicable to the 4x oversampling case.

In order to compensate for these bottlenecks, parallel gain has to be extracted from somewhere else. Bearing in mind the extremely low hardware cost of the last loop `despreading_loop`—i.e. binary coefficients to put it in signal processing terms—it was decided to partially unroll the loop by the same factor that occurs throughout all loop unrolling directives; the chip oversampling rate. Additional speed-up remains possible if need be as to ensure meeting the final adaptation deadline.

writeback dependencies The sequence of computations involving the channel estimate vector amounts to four loops; namely, updating, finding maximum coefficient, truncating if applicable, and calculating power. In order to achieve maximum throughput on this subset of algorithmic operations, the following techniques were utilized resulting in modifications to the original code. Firstly, a temporary channel estimate vector was allocated and passed to a first of two channel-related functions which manually merges updating and searching for the maximum coefficient. The updated channel estimate was written to the temporary vector as to mitigate against writeback dependencies that would have otherwise arisen in the pipelined loop. Expression balancing is also applied on the update loop. Simultaneously on the fly, a fast complex magnitude estimator [20] is used for the max search. Listing 8.1 demonstrates the expression-balanced, pipelined loop. The second loop then takes on the tasks of truncating the temporary channel vector into the original channel vector while calculating the channel power at the same time. Ordinarily, having two loops in a producer-consumer fashion in succession facilitates increased task parallelism by the application of high-level dataflow directives for the inference of

possible execution overlap between the loops.[1] Due to the intricate[2] and intertwined nature of four operations and after numerous experimentations, the best performance was achieved through this manual code rewriting.

Listing 8.1 illustrates the various concepts discussed thus far. First, the channel and interference-free signal are fetched from memory into pipleline registers. Second, multiplications are performed and stored in intermediate registers. Third, the channel is updated and the result is written back to the temporary channel vector in order to avoid writeback dependencies. Fourth, at the same time, the magnitude of the current channel coefficient is computed and the result is written back to the channel magnitude vector. Finally, the current magnitude is tested against the maximum coefficient, which is updated correspondingly if applicable.

```
1   update_n_magx_loop : for (int m = 0; m < cdma.SIZE(); m++)
2   {
3     //! fetch
4     Yi1 = cdma.ds.h_hat_vect[m];
5     Yi2 = cdma.ds.v0_hat_vect[m];
6
7     //! multiply
8     h = this->cmplx_mult(Yi1, Xr1);
9     v0 = this->cmplx_mult(Yi2, Xr2);
10
11    hm = h + ( q_kTc ? -v0 : v0 );
12
13    //! writeback
14    h_hat_vect_temp[m] = hm;
15
16    hMag = fxMath.template
17              cmplx_mag_eqrppl<
18                  h_hat_mag_type,
19                  typename D::ch_type,
20                  ap_ufixed<18,1>
21                              >(hm);
22    acqParams.h_hat_mag_vect[m] = hMag;
23    if ( hMag > magxCh )
24    {
25      magxCh = hMag;
26      chDelay = m;
27    }
28  }
```

Listing 8.1: Merged channel update and maximum coefficient search with writeback dependency avoidance

feedforward filter adaptation The complex feedforward filter is implemented as an FIR filter with a RAM-based circular buffer. This is more economical than a register file realization especially given the real and imaginary parts of the signal and coefficient

[1] In AutoPilot, such optimization is available through the `set_directive_dataflow` directive.

[2] There is a runtime condition for truncation that may be hampering the efforts of the HLS tool to reach the optimal latency.

buffers; which means quadrupling the generic fabric utilization for such a realization. As a result of the relatively short delayline of the oversampled chip-order filter, adaptation was implemented efficiently incurring only twice the filter's length in clock cycles latency as follows. The two delaylines are loaded into local arrays reversing the signal vector. Indexing the delaylines has to be done in a consistent manner with the actual filtering subroutine as elaborated on above. The local arrays are then used with the adaptation factor to update the original coefficient vector while always pipelining and expression balancing for maximum throughput.

8.2.4 Loops

Various loop-related arrangements have already been alluded to in prior discussion. Nonetheless, for the sake of a cohesive treatment of hardware concepts, the transformations pertaining to loops will be enumerated again under this heading. First, channel-related loops are manually merged due to the reasons discussed above. Apart from this manual step, subsequent operations are performed automatically through the use of high-level directives. Second, unrolling is applied on virtually all loops occurring in the algorithm. Nested loops are fully unrolled. The remaining loops in which intensive processing takes place are partially unrolled by the oversampling rate factor. Third, pipelining directives are utilized on the partially unrolled loops for maximal parallelism.

8.3 Implementation

This section covers the implementation details of the proposed algorithm. This is necessary primarily because of to the relative obscurity of the state-of-the-art HLS design methodology and the use of untimed C++ for hardware synthesis. The templatized C++ remains parameterizable in that it mirrors the generic feature in more traditional HDL design flows.

8.3.1 Code Level

The fixed-point algorithm is organized as shown in listing 8.2.

Five classes make up the final algorithmic wrapper class. These are now discussed in detail.

DSSS

Class `dsss_type` encapsulates all DSSS data types, entities, and access methods. As can be seen, nested templatization is used in order to logically separate the entities and access methods. The various DSSS entities are: code sequence `q`, channel vector `h_hat_vect`, interference-free signal vector `v0_hat_vect`, signal vector `v`, and average signal vector `v_bar_hat`. A common structure is used for the circular buffers `v` and `v_bar_hat`. For the sake of flexibility, this structure has two indices `high_ndx` & `low_ndx`

```
1   //! Algorithm instance
2   static algo_fixed_type<
3     dsss_type< ds_base<
4         sig_type,
5         avrg_sig_type,
6         ch_type,
7         ds_addr_type,
8         params::L, params::Ns> >,
9     fir_type< fir_base<
10        sig_type,
11        ff_coeff_type,
12        ff_addr_type,
13        params::L_ff, params::Ns> >,
14    cordic_type< cordic_base<
15        cordic_int_type,
16        cordic_fract_type> >,
17    fix_math_type,
18    bfp_type<sig_type,int,params::L,params::Ns>
19                                                > algo;
```

Listing 8.2: Algorithm instantiation

to point to the high and low position of vectors at any given time kTc. In block-floating point (BFP) mode, all vectors share a common exponent `expnt`. The access methods are: `push_chip`, `slice_h_hat_vect`, and `push_estimate`. Note that this templatized approach means also that a function will only be instantiated upon demand, i.e. when called.

FIR

Similar to DSSS, class `fir_type` contains all FIR data types, entities, and access methods. The entities are: the signal circular buffer `sig` and the coefficients vector `coeffs`. One access method is provided for pushing data into the delayline, namely `push`.

CORDIC

Trigonometric functions in the algorithm are implemented with a COordinate Rotation DIgital Computer core [79]. CORDIC is versatile and is ideally suited to hardware realizations. The low-rate of ultrasound makes evaluating trigonometric functions iteratively over a small number of clock cycles very favourable, in terms of area and throughput.

The core has two modes of operation: rotation and vectoring. The rotation mode is used to evaluate the complex exponential function occurring throughout the algorithm. The vectoring mode is used to obtain the arctangent of a complex value needed in the linear interpolator.

The classic CORDIC equations [22] are

$$
\begin{aligned}
x_{i+1} &= x_i - y_i \cdot d_i \cdot 2^{-i} \\
y_{i+1} &= y_i + x_i \cdot d_i \cdot 2^{-i} \\
z_{i+1} &= z_i - d_i \arctan(2^{-i})
\end{aligned} \tag{8.1}
$$

where depending on the mode (rotation or vectoring), d_i is given by

$$
d_i = \begin{cases} -1 & \text{if } z_i < 0, +1 \text{ otherwise; rotation} \\ +1 & \text{if } y_i < 0, -1 \text{ otherwise; vectoring} \end{cases} \tag{8.2}
$$

In HLS, the hardware directive `inline` provides a powerful architectural exploration mechanism. When inlining a function, a dedicated instance of the function is synthesized in the callee region. Since the overall algorithm has a relatively large layout, it would be infeasible to attempt to conserve resources and instantiate one CORDIC core to have all trigonometric functions routed to it. Any resource gains by such an arrangement will easily be diminished by the long caller logic. Another remark pertaining to the CORDIC operation is the potential use of the hardware directive `instantiate`, which allows for a call-specific realization of a given hardware function. This could in certain cases have an optimizing effect by removing additional unwanted control logic around the function. In the core `rotate` method in class `cordic_type`, it was deemed that such an optimization have a very little effect on the resultant logic as it would only save a boolean check inside the rotation loop.

The CORDIC operation is defined only over the interval $|\varphi| < \frac{\pi}{2}$. Other unaccounted quadrants have to be corrected for. The final algorithmic class contains a function that performs range checks to flag the right quadrant before calling the CORDIC core. Upon return, the proper sign is then reconstructed.

Fixed-point math

`fix_math_type` is a template class which encapsulate a variety of further templatized mathematical utility functions that are designed to aid the fixed-point implementation of the algorithm while retaining maximal code modularity.

To demonstrate the functionality of the class, two examples will be given. Firstly, during channel truncation, the complex magnitude of the channel coefficient needs to be estimated inside a loop, and on the fly. In order to achieve this, a fast magnitude estimation method with equiripple-error characteristics is utilized [47].

Secondly, three fully templatized routines for complex multiplication are included in class `fix_math_type` for the cases of two complex multiplicands, and complex and real ones. At the point of instantiation, the return type and the two operands can all be specialized. This allows for maximum flexibility during programming. A typical usage case is shown in listing 8.3.

```
1  ACC0 += fxMath.template
2            cmplx2_mult<
3              cmultPlus1_type,
4              typename D::ch_type,
5              typename D::avrg_sig_type >(Y0, X0);
```

Listing 8.3: Flexible complex multiplication

Block Floating-Point

Class `bfp_type` handles the task of formatting the incoming samples into a block floating-point representation (BFP) [69]. It employes a ping-pong buffering scheme with the appropriate exponents and flags to accomplish formatting. Max and min searches are carried out on-the-fly inside the `push` access method for maximum throughput. A templatized method is available as an external interface for the passing of the current exponent and the formatted chip as listing 8.4 illustrates.

```
1  //! @brief Buffering Stage
2  typename D::sig_type buffed_chip[params::Ns];
3  typename B::exponent_type ex;
4
5  bool outcome;
6  bfpFrmttr.template buffer<bool>(outcome, ex, buffed_chip, chip);
7
8  if ( outcome )
9  {
10 .
11 .
12 .
```

Listing 8.4: BFP operation

It is worth pointing out that the method has a latency of one spreading code duration. That is, strictly speaking BFP is not real-time. However, a latency of one code duration (around 25 ms) is likely to be imperceivable from an application standpoint. Further afield, the issues surrounding the BFP operation will be aggregated in one subsection.

The Algorithm

The wrapper class `algo_fixed_type` ties all aforementioned classes and applies the various algorithmic operations in order to realize the final adaptation.

8.4 Algorithmic Operations Implementation

Here the various operations that make up the proposed algorithm, along with the high level directives applied and other synthesis considerations, are thoroughly detailed.

Before discussing the algorithmic operations, a general note on the buffers that make up the data plane of the combined adaptation is due. Three types of high-level directives are applied as shown in the snippet in listing 8.5.

```
1  set_directive_array_partition -type cyclic -factor 4 -dim 1 dsss_type<B>::
       push_estimate ds.v_bar_hat.vect
2  set_directive_array_partition -type cyclic -factor 4 -dim 1 dsss_type<B>::
       slice_h_hat_vect ds.h_hat_vect
3  set_directive_unroll dsss_type<B>::push_chip/push_chip_loop
4  set_directive_unroll dsss_type<B>::push_estimate/push_chipEst_loop
5  set_directive_unroll dsss_type<B>::slice_h_hat_vect/slice_fract_loop
6  set_directive_array_stream algo_fixed_type<D,F,C,FM>::exe cdma.ds.h_hat_vect
7  set_directive_array_stream algo_fixed_type<D,F,C,FM>::exe cdma.ds.v0_hat_vect
8  set_directive_array_stream algo_fixed_type<D,F,C,FM>::exe acqParams.
       h_hat_mag_vect
```

Listing 8.5: General directives

Cyclic partitioning by the chip oversampling factor is applied in access methods of class `dsss_type`. When in an iterative processing loop, throughput is maintained by unrolling the loops that feature in access methods. The data buffers with static boundaries (i.e. not circular) are streamed in order to enhance timing. This is because when needed streaming enables functions to commence processing before the entire data vector has been received.

Linear Interpolate

Linear interpolation requires only light processing and there is no need for architectural directives. However, due to the limited dynamic range of fixed-point math, a synthesis command for the inclusion of a floating-point unit in the module is applied as shown in the snippet in listing 8.6.

```
1  add_library xilinx/virtex5/virtex5_fpv5
```

Listing 8.6: LI command

Feedforward Filter

For complex FIR filtering, the kernel is pipelined for maximum throughput, but no additional architectural directives are utilized. That is, the nested operation is left purely sequential owing to the relative short length of the chip delayline.

The two-dimentional buffers in figure 8.2 signify a sequential operation whose outcome is feedforward filtered chip $\underline{c}$.

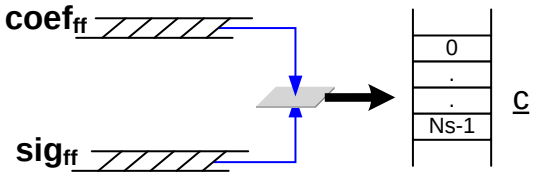


Figure 8.2: Feedforward filter

```
1  set_directive_pipeline algo_fixed_type<D,F,C,FM>::fir_exe_kernel/fir_kernel
```

Listing 8.7: FF directive

Carrier-synchronize the chip

The phase of the incoming forward equalized chip is corrected by the complex exponential $e^{-j\theta}$ driven by the PLL. Similar to the FF, and as can be inferred from figure 8.3, the complex multiplication is left rolled. The now forwarded and synchronized chip is denoted by $\underline{c}'$.

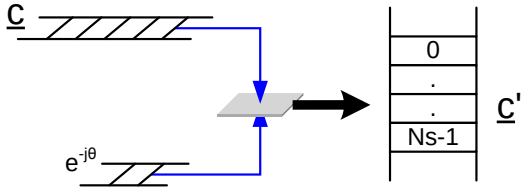


Figure 8.3: Carrier synchronization

Push chip onto DSSS signal vector

The access method `push_chip` from class `dsss_type` is invoked. The chip oversampling loop is fully unrolled for maximum throughput.

Form the average signal vector

Two chip oversampling rate–deep arrays are allocated for the average signal and the channel. These arrays are partitioned cyclically by the oversampling rate i.e. fully[3]. The initialization loop is fully unrolled. Inside the code-length loop, two nested accumulation loops resulting from polling on a chip are fully unrolled as well. The final loop is pipelined for maximum throughput.

Figure 8.5 illustrates the forming of the average signal vector. The three-dimensional operation indicates concurrency in the chip oversampling rate dimension, and temporal processing in the code length dimension. The outcome is a single entity $\underline{\overline{v}}$ of N_s entries, which is later pushed onto the circular average signal vector.

[3]partitioning an array by its length breaks the array into registers

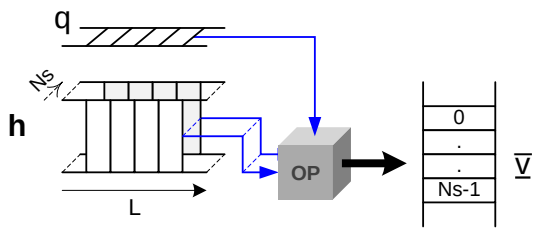


Figure 8.4: Form average signal vector

```
1  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM>::estimate_v_bar_hat_vect v_bar_hat_line
2  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM>::estimate_v_bar_hat_vect hm_hat_line
3  set_directive_unroll algo_fixed_type<D,F,C,FM>::estimate_v_bar_hat_vect/
       initi_line_loop
4  set_directive_unroll algo_fixed_type<D,F,C,FM>::estimate_v_bar_hat_vect/
       v_bar_conv_chip_loop_true
5  set_directive_unroll algo_fixed_type<D,F,C,FM>::estimate_v_bar_hat_vect/
       v_bar_conv_chip_loop_false
6  set_directive_pipeline algo_fixed_type<D,F,C,FM>::estimate_v_bar_hat_vect/
       v_bar_code_loop
```

Listing 8.8: Average signal directives

Form the estimate of the interference-free signal vector

Two vectors from class `dsss_type` feature in this function: the channel and the interference free signal. It should be clear by now that the cyclic partitioning by the oversampling rate is a repetitive step that ensures increased throughput in each function. The DSSS-length (i.e. LN_s) loop is unrolled by the oversampling rate too. The loop is then pipelined for maximum throughput. Additionally, owing to having two operands being assigned per iteration, an additional high-level directive for expression balancing is needed. This directive instructs the synthesis tool to use additional resources in a balanced tree fashion in order to allow for single clock execution of the operation.

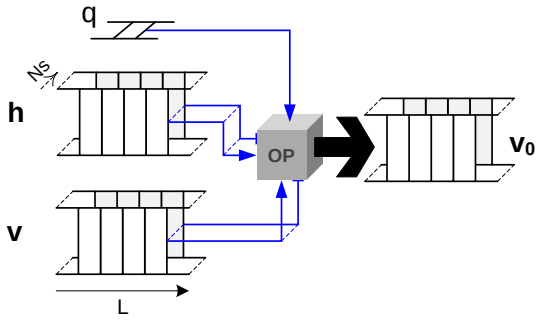


Figure 8.5: Interference-free signal vector

Figure 8.5 depicts the operation in 3D, wherein boosted parallelism is extracted from the chip oversampling rate dimension while temporal folding takes place in the code length dimension. The outcome is a three dimensional vector.

```
1  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM>::estimate_v0_hat_vect cdma.ds.h_hat_vect
2  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM>::estimate_v0_hat_vect cdma.ds.v0_hat_vect
3  set_directive_unroll -factor 4 algo_fixed_type<D,F,C,FM>::
       estimate_v0_hat_vect/v0_size_loop
4  set_directive_pipeline algo_fixed_type<D,F,C,FM>::estimate_v0_hat_vect/
       v0_size_loop
5  set_directive_expression_balance algo_fixed_type<D,F,C,FM>::
       estimate_v0_hat_vect/v0_size_loop
```

Listing 8.9: Interference-free signal directives

Form the chip estimate

The function makes use of the three major high-level directives that characterize the various algorithmic stages sor far. It employs: cyclic partitioning of memories by the oversampling factor, loop unrolling by the oversampling factor, and loop pipelining for maximum throughput.

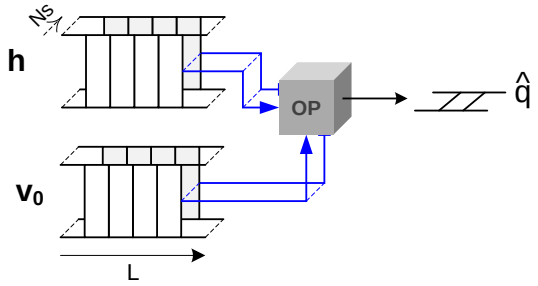


Figure 8.6: Chip estimate

```
1  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM>::estimate_chip cdma.ds.h_hat_vect
2  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM>::estimate_chip cdma.ds.v0_hat_vect
3  set_directive_unroll -factor 4 algo_fixed_type<D,F,C,FM>::estimate_chip/
       q_hat_loop
4  set_directive_pipeline algo_fixed_type<D,F,C,FM>::estimate_chip/q_hat_loop
```

Listing 8.10: Chip estimation directives

As can be seen from figure 8.6, a 3D multiply-accumulate operation using the channel and the interference-free signal is carried out to produce a single complex estimate of the current chip.

Update the channel estimate vector and find maximum magnitude

As described earlier, the channel operations have been manually fused and customized. There are four operations in total: update, find maximum coefficient, truncate, and compute the energy. The four operations have been merged into two functions, each simultaneously handles two operations. Channel estimate update and finding maximum coefficient are performed in this function. The remaining two operations are left to another function which will be discussed below.

Figure 8.7 shows how four vectors are cyclically partitioned by the oversampling rate. The channel and interference-free vectors are from class `dsss_type`. The additional two vectors are a temporary channel[4] and the channel magnitude vector. The loop is unrolled by the oversampling factor and pipelined. Expression balancing is also applied in order to have a sustained throughput of one clock cycle after the pipeline initiation interval. The channel coefficient of maximum magnitude is searched for on-the-fly. In order for this to happen, the function responsible for estimation from class `fix_math_type` is expression-balanced as well. Expression balancing is also applied on the overall region of the update loop for maximal throughput.

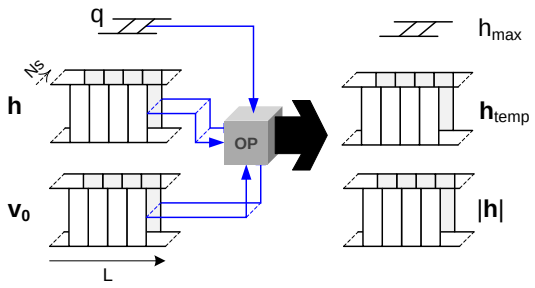


Figure 8.7: Update channel - step 1

The 3D, concurrent operation of figure 8.7 takes as input: a the current boolean chip, the partitioned channel, and the partitioned interference-free signal. The outcome is: the magnitude of the maximum coefficient, an updated channel estimate held in a temporary partitioned vector, and a partitioned channel magnitude vector.

Truncate channel estimate vector if applicable

The second channel-related function deals with truncating the channel and computing its energy on-the-fly. Similar cyclic partitioning, loop unrolling, and pipelining are applied too.

As shown in figure 8.8, a 3D operation is applied on the partitioned temporary channel update and channel magnitude vectors using the maximum channel magnitude for thresholding. The outcome is then written back to the channel in the data plane. At the same time, the energy corresponding to non-zero coefficients is accumulated and then reciprocated for later use. That is, this function contains one unsigned division.

[4] see p. 119

```
1  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM>::update_n_magx_ch cdma.ds.h_hat_vect
2  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM>::update_n_magx_ch cdma.ds.v0_hat_vect
3  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM>::update_n_magx_ch h_hat_vect_temp
4  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM>::update_n_magx_ch acqParams.h_hat_mag_vect
5  set_directive_unroll -factor 4 algo_fixed_type<D,F,C,FM>::update_n_magx_ch/
       update_n_magx_loop
6  set_directive_pipeline algo_fixed_type<D,F,C,FM>::update_n_magx_ch/
       update_n_magx_loop
7  set_directive_expression_balance fix_math_type::cmplx_mag_eqrppl/
       cmag_eqrppl_region
8  set_directive_expression_balance algo_fixed_type<D,F,C,FM>::update_n_magx_ch/
       update_n_magx_loop
```

Listing 8.11: Channel update step 1 directives

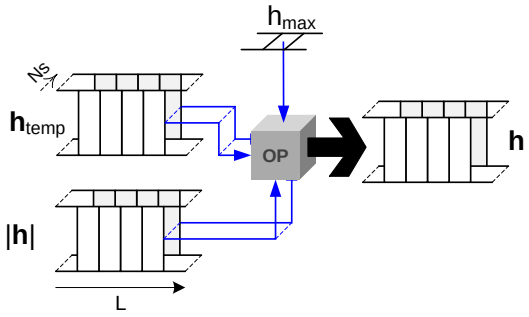


Figure 8.8: Update channel - step 2

```
1  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM>::trunc_n_pow_ch h_hat_vect_temp
2  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM>::trunc_n_pow_ch acqParams.h_hat_mag_vect
3  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM>::trunc_n_pow_ch cdma.ds.h_hat_vect
4  set_directive_unroll -factor 4 algo_fixed_type<D,F,C,FM>::trunc_n_pow_ch/
       trunc_n_pow_loop
5  set_directive_pipeline algo_fixed_type<D,F,C,FM>::trunc_n_pow_ch/
       trunc_n_pow_loop
```

Listing 8.12: Channel update step 2 directives

Generate chip estimation error

In this function, a simple differencing to obtain the chip error is carried out.

Update decision directed PLL

The channel and signal vectors from class `dsss_type` are used in the PLL update procedure. Both are partitioned as is becoming the norm so far. The update loop is unrolled, piplelined, and expression-balanced.

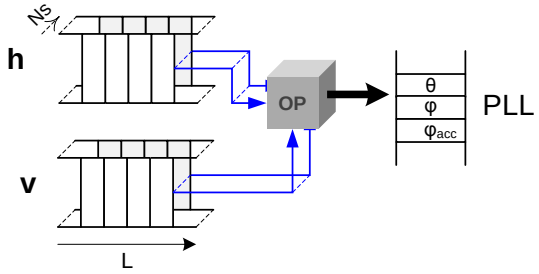


Figure 8.9: Update decision directed PLL

The operation shown in figure 8.9 corresponds to a 3D MACC filtering using the partitioned channel and signal vectors. The result is then used with the energy reciprocal from above to compute ψ, ψ_{acc}, and θ. The overhead of the 3D filtering leaves plenty of time for the division from the second channel update function to evaluate.

```
1  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM>::update_pll cdma.ds.h_hat_vect
2  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM>::update_pll cdma.ds.v.vect
3  set_directive_unroll -factor 4 algo_fixed_type<D,F,C,FM>::update_pll/
       phase_loop
4  set_directive_pipeline algo_fixed_type<D,F,C,FM>::update_pll/phase_loop
5  set_directive_expression_balance algo_fixed_type<D,F,C,FM>::update_pll/
       phase_loop
```

Listing 8.13: PLL update directives

Perform despreading

For the despreading loop, the code sequence and the code estimate sequence are partitioned cyclically by an oversampling rate factor. The loop is then unrolled by the same factor and pipelining is applied.

A 3D integration loop is performed in figure 8.10. The parallel gain is extracted from the oversampling rate dimension. The outcome is a complex value corresponding to the running despreader within a code window.

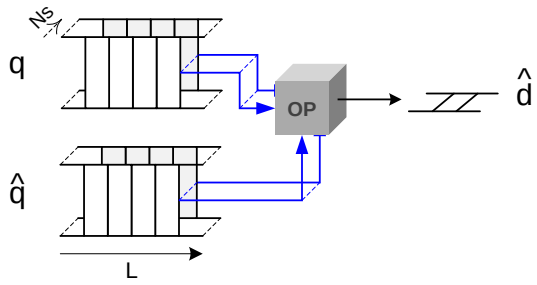


Figure 8.10: Despread

```
1  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM,B>::despread cdma.ds.q
2  set_directive_array_partition -type cyclic -factor 4 -dim 1 algo_fixed_type<D
       ,F,C,FM,B>::despread acqParams.q_hat
3  set_directive_unroll -factor 4 algo_fixed_type<D,F,C,FM,B>::despread/
       despreading_loop
4  set_directive_pipeline algo_fixed_type<D,F,C,FM,B>::despread/despreading_loop
```

Listing 8.14: despreader directives

Test for [cont.] acquisition

This function tests for initial acquisition. The squared despreader error is compared against a threshold. Upon falling below the threshold value, testing is declared valid. Upon a number of successive valid tests, acquisition is declared. Code acquisition is reset if any of the already mentioned conditions is violated. This function is recursive and requires little hardware resources. As such, no directives are applied.

Update feedforward filter LMS-adapting

The LMS FF adaptation was discussed in page 120. In order to realize the update, three directives are needed as shown in listing 8.15. Both the local load loop and the update loop are pipelined for maximum throughput. The latter is also expression-balanced to allow for on-the-fly complex multiplication and complex addition.

```
1  set_directive_pipeline algo_fixed_type<D,F,C,FM>::
       lms_adapt_feedforward_filter/lms_update0
2  set_directive_pipeline algo_fixed_type<D,F,C,FM>::
       lms_adapt_feedforward_filter/lms_update1
3  set_directive_expression_balance algo_fixed_type<D,F,C,FM>::
       lms_adapt_feedforward_filter/lms_adapt_region
```

Listing 8.15: FF update directives

Update linear interpolator

No directives are applied for the LI update. The arctangent implemented with the CORDIC core is left fully rolled. One constant real MACC is utilized for the interpolation index update.

8.5 Analysis

Having described the algorithm implementation in detail, the corresponding analysis will be supplied next. The analysis is based on HLS synthesis by Xilinx's AutoPilot[5] version 2010.a.2. The device targeted is the Xilinx Virtex5 XC5VSX94T, package FF1136, speed grade -1. The basic synthesis commands are clock period of 10 ns with 1.25 ns uncertainty. The nominal clock period corresponds to a budget of 5000 cycles per adaptation step at the system's 20 kHz chip rate. The value provided by the clock uncertainty influences early architectural decisions during the process of *scheduling & binding* [25]. This is done as to ensure that back-end processes (such as synthesis and place-and-route) have enough slack to successfully target a specific device technology. The back-end tools are the Xilinx ISE 12.2 suite.

The IO used in the RTL implementation settings are an input complex chip of type RAM and an output flag of type WIRE connected in a slave bus setup. The flag is tied to the boolean value indicating successful acquisition, which is updated every adaptation step.

The characterization is conducted with the following nominal fixed-point precisions throughout the algorithm:

1. complex incoming samples of 16-bit word length
2. complex channel coefficients of 18-bit word length
3. arithmetic operations using the maximum precision that the DSP48E slice has to offer [150]
4. miscellaneous other custom word lengths, which were arrived at by inspecting the equivalent floating-point operation

In what follows, the characterization of area, power, and throughput will be reported from synthesis results. PAR will be used to sanity-check stages of architectural exploration in order to reject unfavourable conclusions that might have slipped past synthesis unflagged. Nonetheless, PAR results will not be reported. Besides the long completion overhead, this is also because the downstream tools will prune and optimize away all unnecessary logic that is not tied to the final output assignment. The output assignment is currently a single boolean flag indicating [continual] acquisition. A production-level implementation has to communicate out more algorithmic primitives such as chip estimates for

[5] formerly of AutoESL

estimating velocity, timing for range[6], channel status for fidelity, etc. Note that when requiring more IO assignments, no additional latency will be incurred since external communication can be overlapped with computations [40] using non-blocking reads/writes [48]. It is also worth nothing that in the context of *automated* architectural exploration, prior work in literature oftentimes reports on execution cycles and area estimates from HLS only in avoiding the lengthy overhead of synthesis and PAR [153, 127].

As noted earlier in section 8.1, the Doppler effect is non-stationary. Therefore, a manual fixed-point precision optimization has to be pursued iteratively. This is formally referred to as *simulation driven analysis* in [31]. Thus, engineering the necessary fixed-point precision is laborious and is outside the scope of the study conducted in this chapter. The main two objectives of this architectural exploration are: (1) to demonstrate the feasibility of the proposed Doppler-tolerant algorithm; (2) gauge the impact of the previous algorithmic findings of chapter 6 on the proposed architecture.

8.5.1 Area

For the unified scalable architecture described earlier, the area implications of the different algorithmic tracking configurations are analyzed here. In what follows, the y-axis on the left-hand-side corresponds to the number of occupied resources for *hard* modules (i.e. BRAMs and DSP slices) and *soft* fine-grained generic fabric (i.e. LUTs and FFs). For graphical illustration purposes, the soft fabric is scaled down by a thousand. The y-axis on the right-hand-side also shows the maximum throughput—and thus adaptation rate—that can be achieved. Throughout the subsection, the analysis is based around two systemic parameters, namely code length and chip oversampling rate, across the various components of the Doppler algorithmic operations previously derived.

Core DS CDMA Engine

An examination of the requisite area for the Core DS CDMA Adaptive Engine is provided here. The channel-based formulation is relatively expensive in terms of BRAM modules when compared to a simple DFE. However, the need for this formulation is best rationalized by considering the multiuser operation which is a critical requirement in a real-world deployed system. In a multiuser operation, effective interference cancellation involves keeping track of the entirety of the long spreading code in order to subtract contributions from other user codes at potentially *different* ranges from the receiver. Such an arrangement has been touched upon previously in UWA research [132].

The growth of area per DSSS length is illustrated in figure 8.11. It is instructive to note the following observations on the *chip-oriented architecture*.

Halving the code length does not result in substantial savings in the hard modules, namely BRAMs and DSP slices, whereas halving the chip oversampling rate does have a substantive conservation effect on hard modules, and soft generic fabric too. Minor savings, however, in soft generic fabric are obtained when halving the code length. The

[6] at least once initially

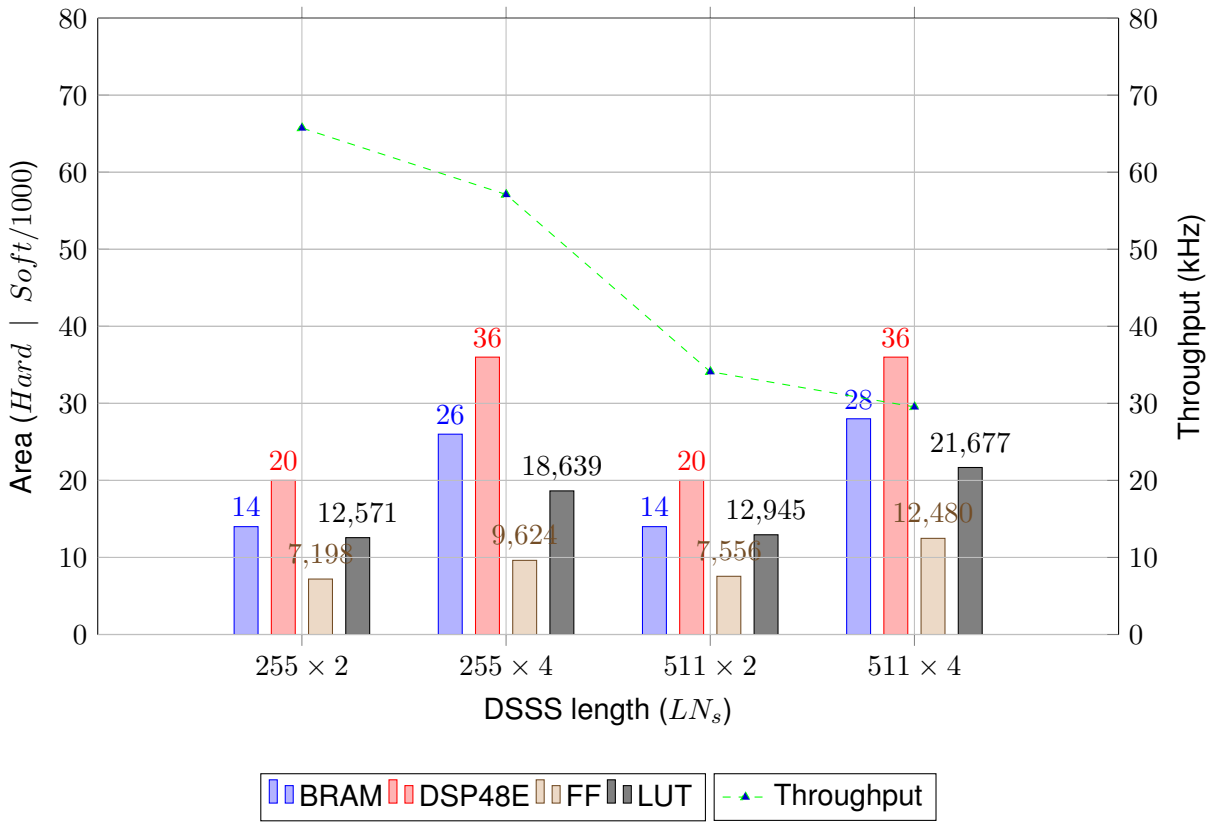


Figure 8.11: HLS area metrics for Core CDMA Engine

non-effect of code length on DSP slices is expected since computations are dependent on the algorithmic workload rather than the length of the buffered signals. Nonetheless, it is interesting to note that no significant BRAM reduction results from decreasing code length. This is due to the sequential nature of most of the operations in the algorithm. In the proposed architecture, data is streamed from the global data plane to where it is needed. This streaming is achieved by high throughput FIFOs which are implemented using BRAMs. The number of required FIFOs is algorithm-dependent as opposed to code length-dependent. As expected, the major effect of halving the code length on the chip-oriented architecture is seen in the doubling of the computational throughput, as illustrated by the dashed line in figure 8.11. Halving the chip oversampling rate results in mild throughput enhancement. In order to reap area savings as we decrease code length, modifications to the high-level directives of the chip-oriented architecture have to be made: cyclic memory partitioning and loop unrolling can be relaxed by a factor of two, in order to balance attainable throughput with the requisite area.

PLL

The area implications of the PLL-only tracking configuration is examined in figure 8.12. The varying of code length and chip oversampling rate give rise to identical observations to those just discussed for the Core DS CDMA Engine. It is also noted that a sharp increase in the needed DSP slices over that of the bare Core DS CDMA Engine can be seen. This is due to the extra processing needed to drive the PLL update. Throughput is

almost linear as we move across code length and chip oversampling permutations.

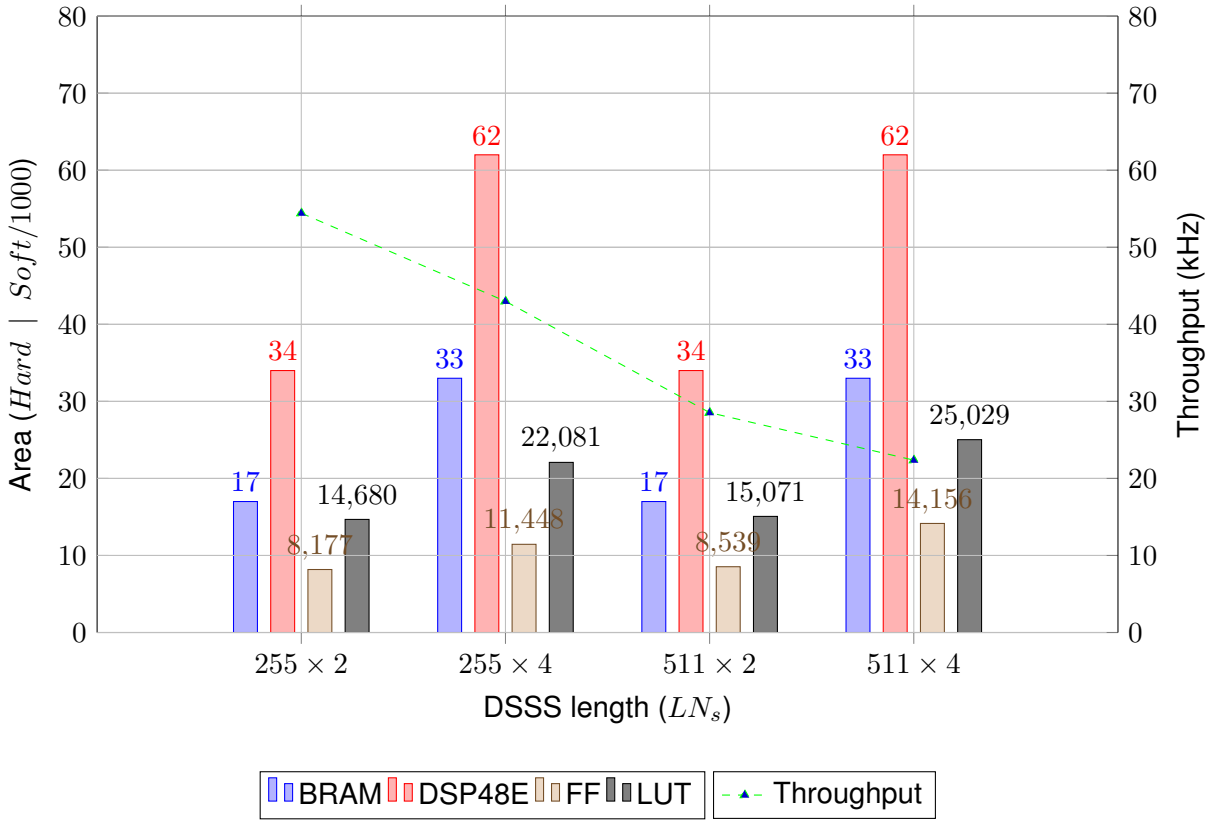


Figure 8.12: HLS area metrics for Core CDMA Engine + PLL

DFE-PLL

The area requirements of the DFE-PLL tracking configuration is shown in figure 8.13. The observations pertaining to the effect of varying code length and chip oversampling on area still hold. The area features a moderate growth over that of the PLL-only configuration across all resources, but is more pronounced in the hard modules. This is due to the inclusion of the FF filtering stage at the beginning of the underlying Core DS CDMA Engine, which is processing-intensive as anticipated. Hence, more BRAMs and DSP slices are needed for the complex equalizer filtering and LMS adaptation. Soft generic fabric acts effectively as glue for arithmetic expressions, which explains the gentle increase above the corresponding PLL-only case. Also similar to the PLL configuration, throughput exhibits linear rise in code-chip rate permutations.

DFE-LI

The area footprint of the DFE-LI tracking configuration is shown in figure 8.14. As derived in chapter 6, the DFE-LI configuration is defined over a one-chip oversampling rate only. Halving the code length has little effect on area, though a commensurate rise in throughput does take place. The resource occupancy of the configuration is somewhat comparable to that of the respective DFE-PLL configuration. However, at the reduced

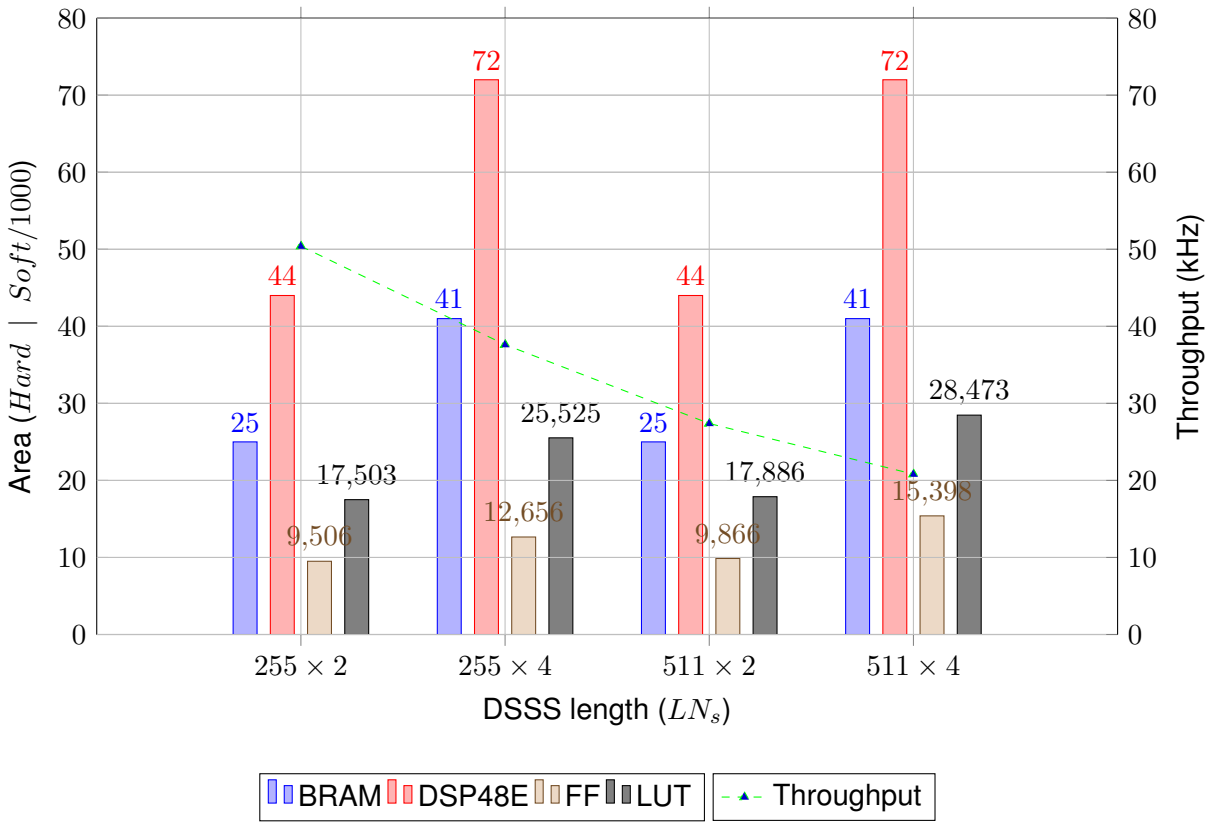


Figure 8.13: HLS area metrics for DFE-PLL

chip oversampling rate, the equivalent DFE-PLL case is not a valid motion tracker as was discussed in chapter 6. From a tracking functionality standpoint, the comparison should be made with the DFE-PLL configuration at the doubled chip rate. In such a case, DFE-LI offers clear area savings, across all resources. Note that the inclusion of a floating-point unit in the interpolator has contributed to the rise in the number of DSP slices.

Summary

The area impact of the different modes of ABU motion tracking which demonstrated some ability to track the Doppler effect is presented next. This is analyzed for the case of 511-bit Gold code in figure 8.15 and the case of 255-bit Gold code in figure 8.16.

In order to remove any implementation-specific bias (e.g. device-specific), the area and throughput are normalized by the values of the DFE-PLL mode, which corresponds to the largest resource and latency requirements. This is illustrated in figure 8.17 for the 511-bit case and in figure 8.18 for the 255-bit case.

8.5.2 Power

The power consumption of previously derived tracking configurations will be characterized, based on the HLS power estimates in the final synthesis reports. This is meant to provide a rough understanding of the power trade-offs of various configurations.

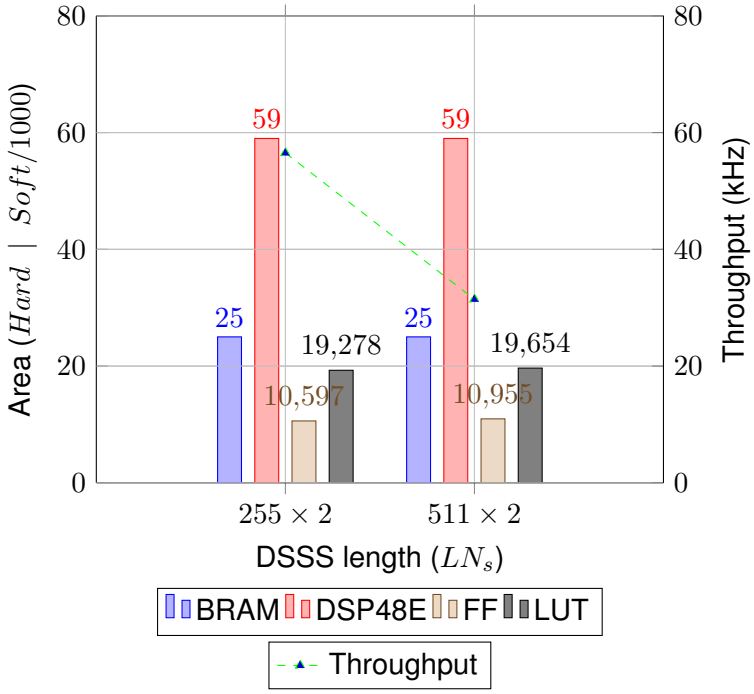


Figure 8.14: HLS area metrics for DFE-LI

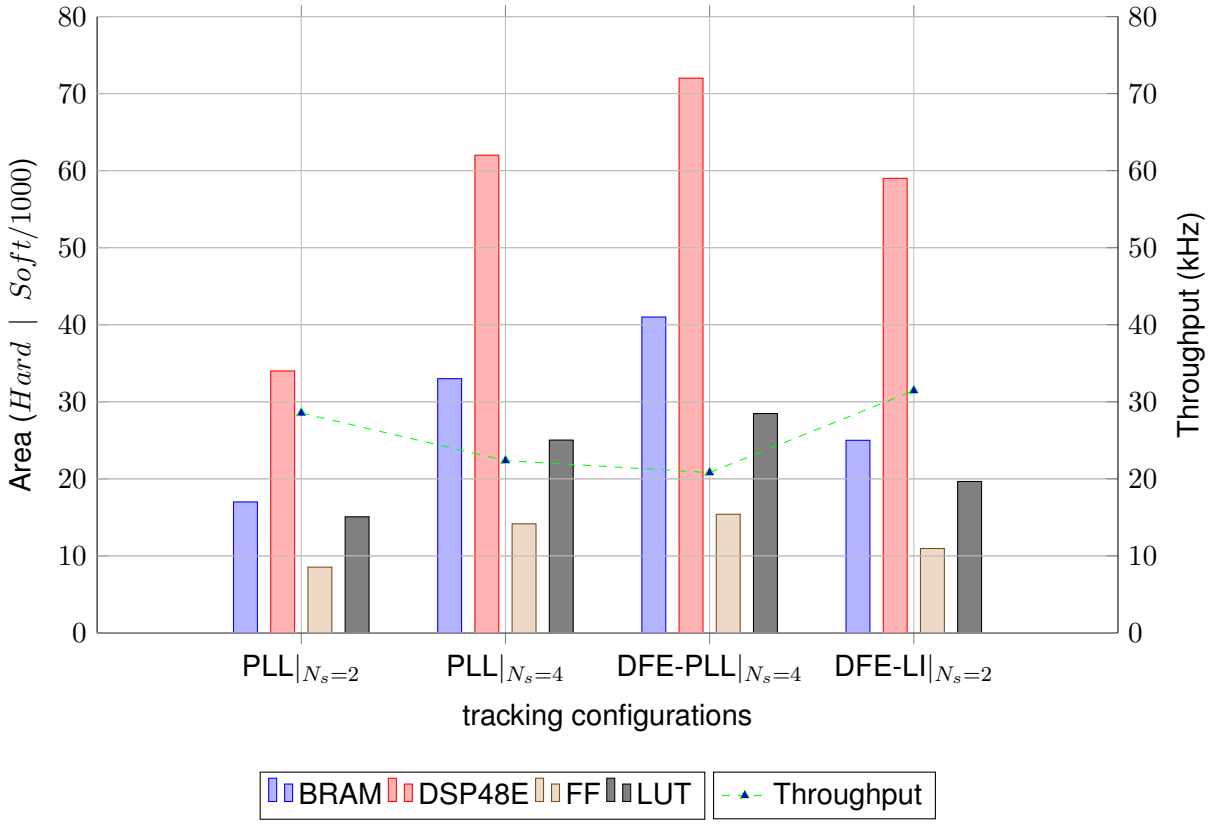


Figure 8.15: 511-bit code HLS metrics comparison between PLL, DFE-PLL, and DFE-LI

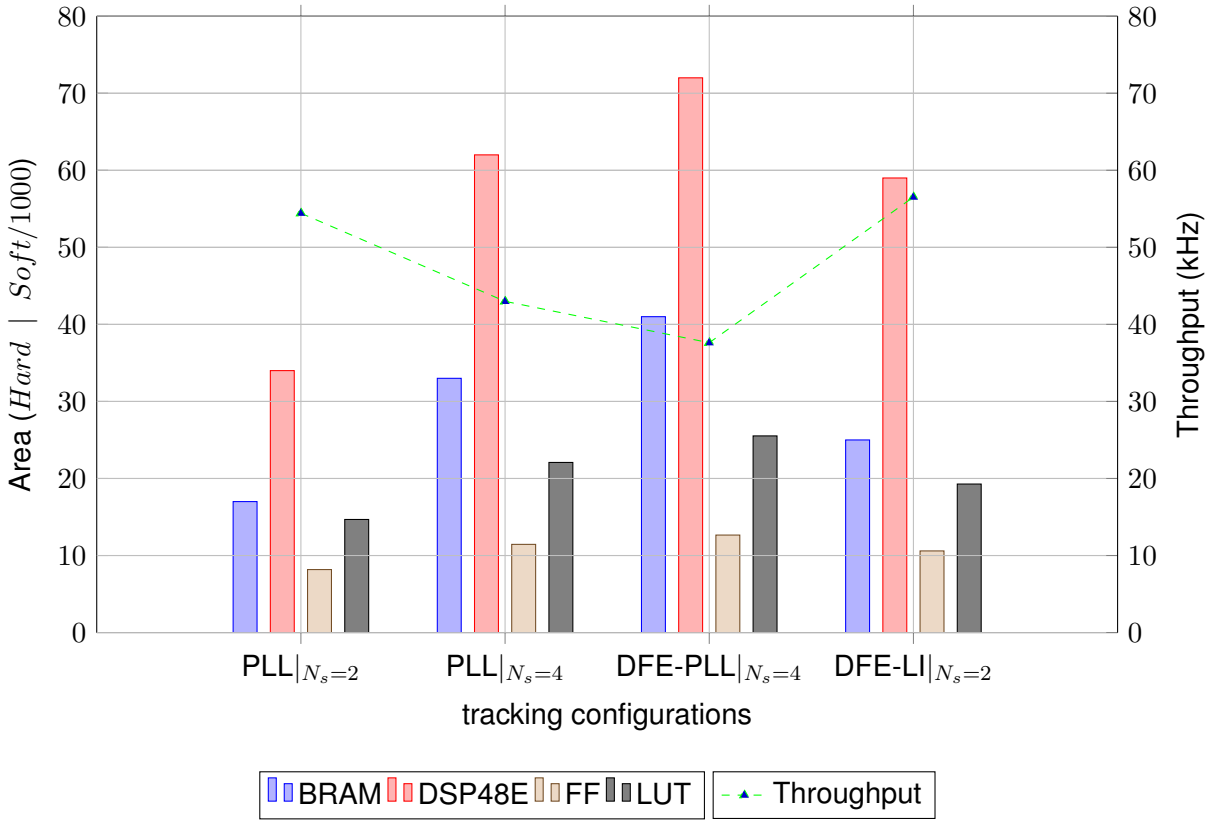


Figure 8.16: 255-bit code HLS metrics comparison between PLL, DFE-PLL, and DFE-LI

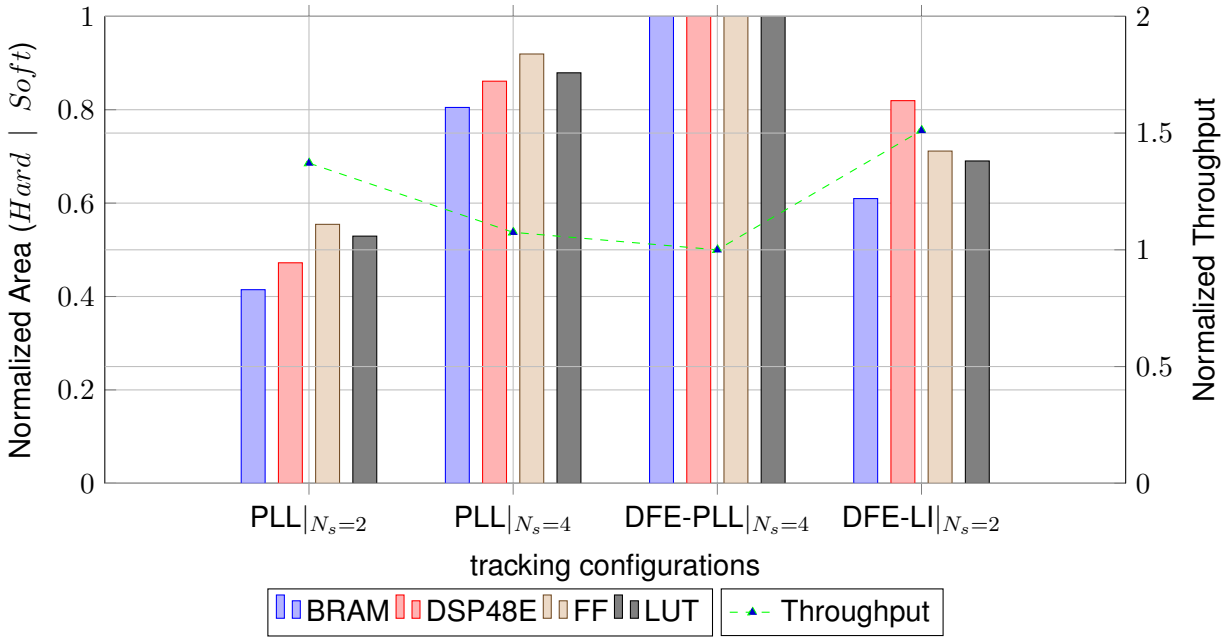


Figure 8.17: 511-bit code normalized HLS metrics comparison between PLL, DFE-PLL, and DFE-LI

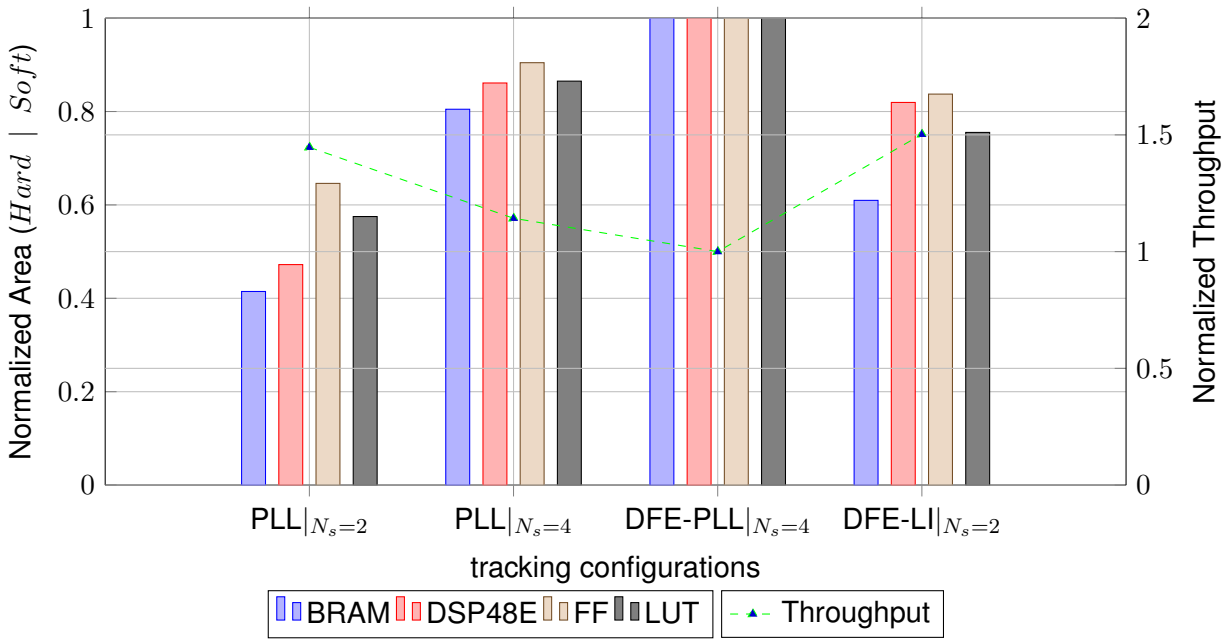


Figure 8.18: 255-bit code normalized HLS metrics comparison between PLL, DFE-PLL, and DFE-LI

The estimate figures will be given when loops traverse the full length of data vectors. This is true for initial acquisition, which typically lasts for few code lengths (e.g. four). The power consumption drops significantly during tracking owing to *sparsing* the channel. As alluded to in chapter 6, only non-zero coefficients need processing in a sparse channel. Hence all vector computations implemented as loops involving the sparse channel will have their boundaries adjusted accordingly. However, it is hard to investigate this effect on the dynamic power consumption profile under the current setup of the HLS tool. So the analysis below focuses on acquisition power consumption only.

Core CDMA Engine

It has been established through earlier discussion how the chip-oriented architecture is decoupled from the code length. This is corroborated again in figure 8.19 where the power profile of the Core CDMA Engine is shown. Only the doubling of the chip oversampling rate results in significant increase in the estimated power. For an oversampling rate of two, the power consumption remains almost the same despite the doubling of code length. At an increased oversampling rate of four, more component power is consumed, because of additional generic fabric to tie the underlying quadruply parallel operation as we double the code length. This is perhaps accredited to being able to accommodate the former case ($N_s = 2$) better given the inherent dual-port nature of the hard modules (BRAMs & DSP slices) whose power consumption is more favourable compared to crude fine-grained fabric. In the latter case, additional components such as intermediate pipelining registers will have adverse effect on power cost.

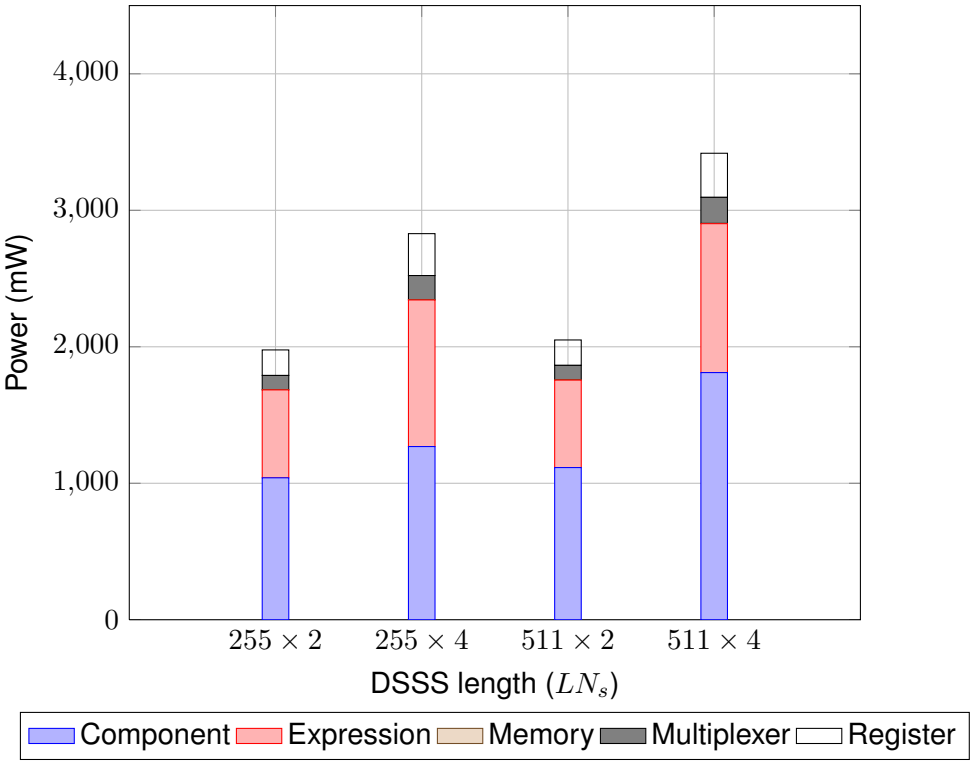


Figure 8.19: HLS acquision power estimate for Core CDMA Engine

PLL

In relation to the power impact of the PLL tracker of figure 8.20, the same observations from the Core CDMA Engine are upheld. This concept is further reinforced noting the nonlinear rise in power across the chip oversampling rate with the inclusion of the PLL's logic—whereas it results in 300 mW increase in the $N_s = 2$ case, it causes roughly 500 mW rise in the $N_s = 4$ case.

DFE-PLL

The power consumption in the DFE-PLL tracker case is illustrated in figure 8.21. Once again the above observations are upheld—at $N_s = 2$ the rising power profile changes very little with the doubling of the code length, but at $N_s = 4$ the same nonlinear effect is manifest. The slope of the total power between the two code lengths at $N_s = 4$ remains consistent throughout the three configurations of Core CDMA Engine, PLL, and DFE-PLL.

DFE-LI

In the DFE-LI configuration of figure 8.22, the power profile is evidently increased when compared to the previous tracker configurations at the same oversampling rate of two. In particular, the power contributed by processing expressions, has risen. This is primarily due to the floating-point unit required by the interpolator. Still, the power profile behaviour at the two code lengths remains consistent to the above findings.

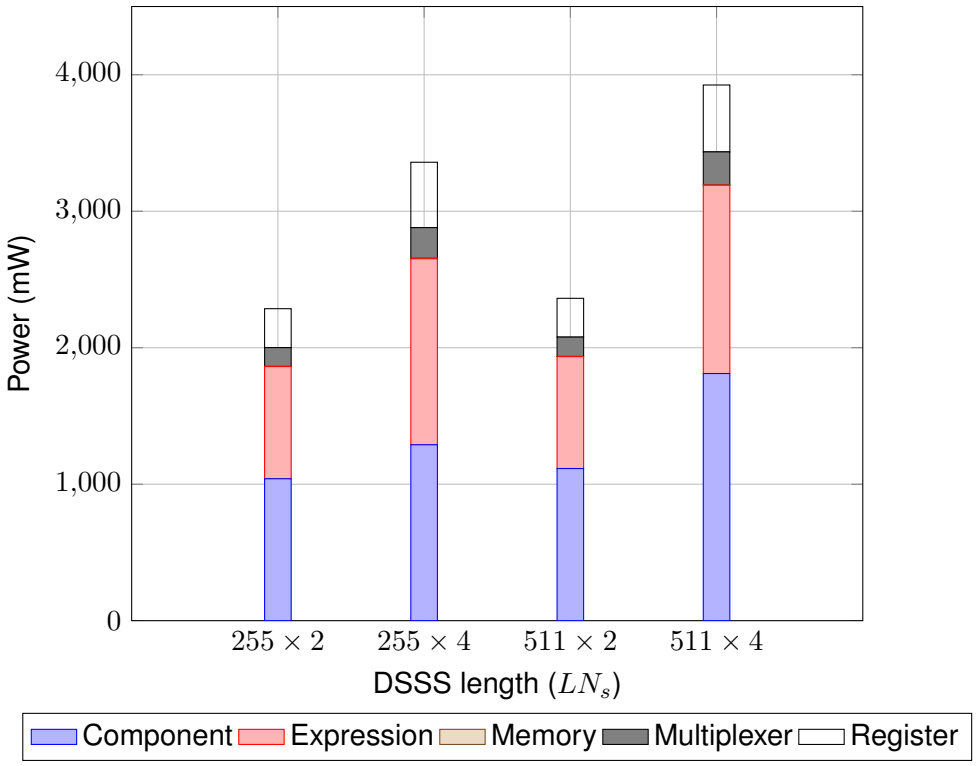


Figure 8.20: HLS acquision power estimate for PLL tracker

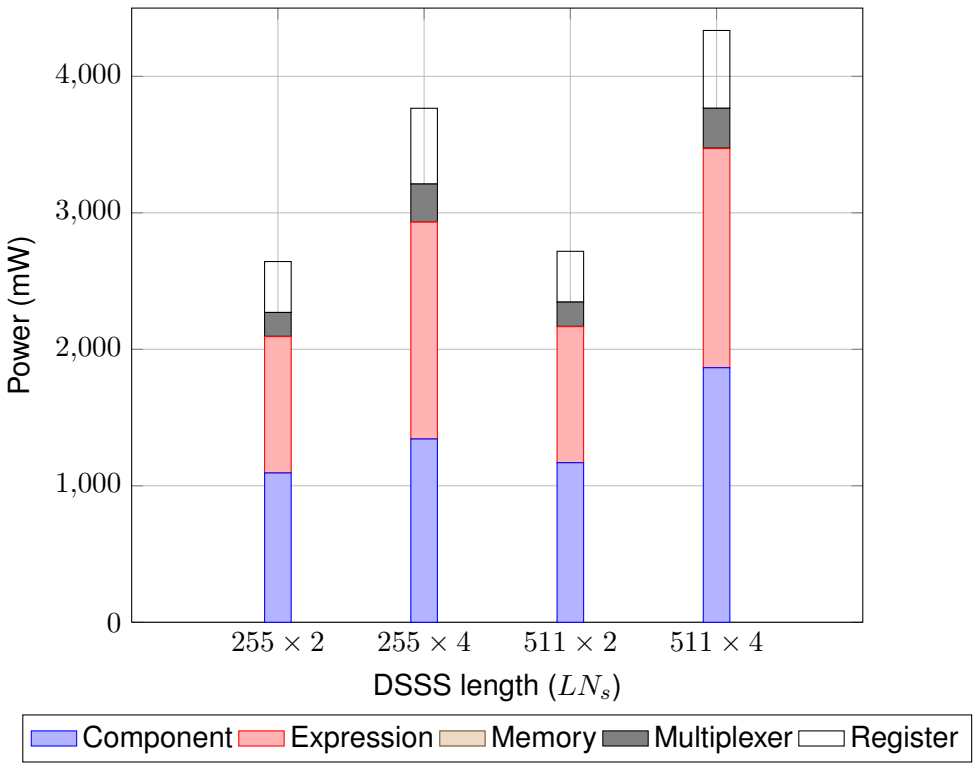


Figure 8.21: HLS acquision power estimate for DFE-PLL tracker

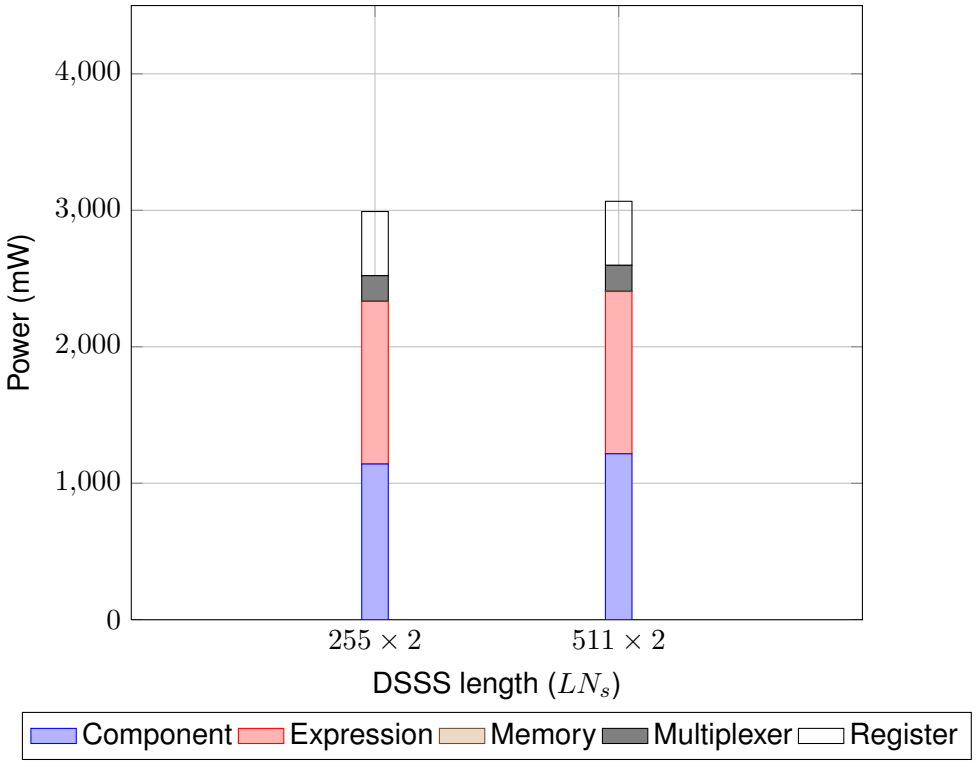


Figure 8.22: HLS acquision power estimate for DFE-LI tracker

Summary

The approximate power profile of the different modes of ABU motion tracking which demonstrated some ability to track the Doppler effect are give for the cases: 511-bit Gold code in figure 8.23, and the case of 255-bit Gold code in figure 8.24. Note that the normalization has revealed the presence of a very small contribution from memory. In the normalized plots, the vertical axes are equally spaced across all constituent power contributors. This arrangement displays the variations these power contributors undergo while moving between the different tracking configurations investigated.

In order to avoid any implementation-specific bias, the power estimates are normalized by the values of the DFE-PLL mode, which corresponds to the largest consumption of the three configurations. This is illustrated in figure 8.25 for the 511-bit case and in figure 8.26 for the 255-bit case.

8.5.3 DFG

A data flow graph (DFG) is an effective tool for the modelling and analysis of signal processing tasks [78, 80]. Its powerful features pose no restrictions on granularity and allow for expressing concurrency explicitly. DFG is also used in fine-grained logic *retiming* through *cut set* theory. DFG can additionally be used in dynamic applications with independent subtasks [106].

In the context of HLS, the data dependencies of a workload are extracted after initial code analysis leading to a so-called control DFG (CDFG) model. The model is comprised of interconnected nodes which translate to the tasks in the code at hand. The arrange-

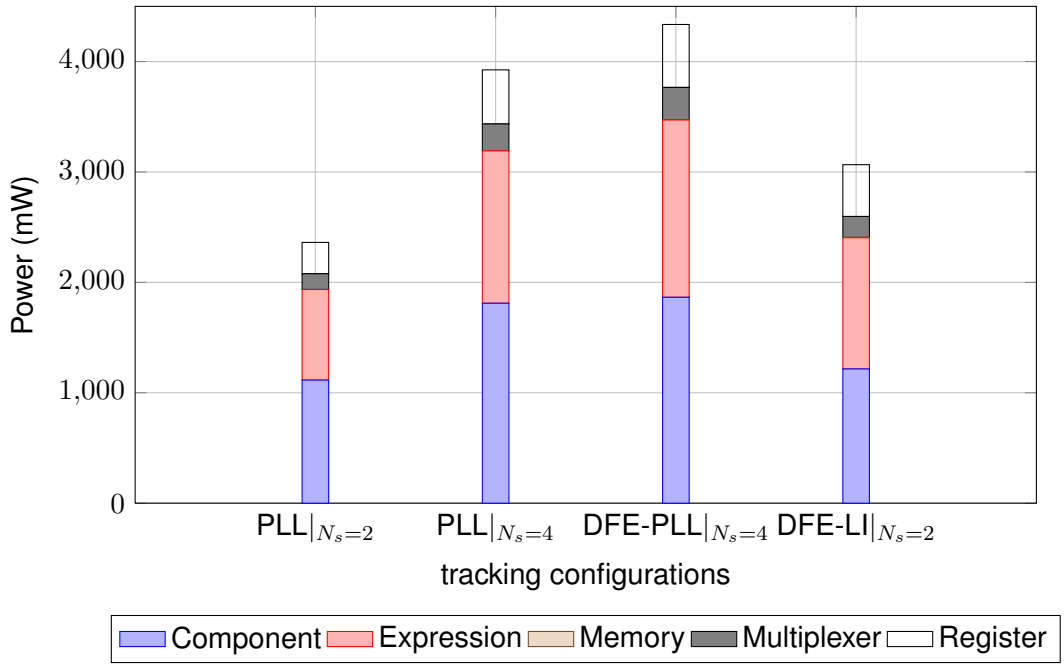


Figure 8.23: 511-bit code HLS power estimate comparison between PLL, DFE-PLL, and DFE-LI

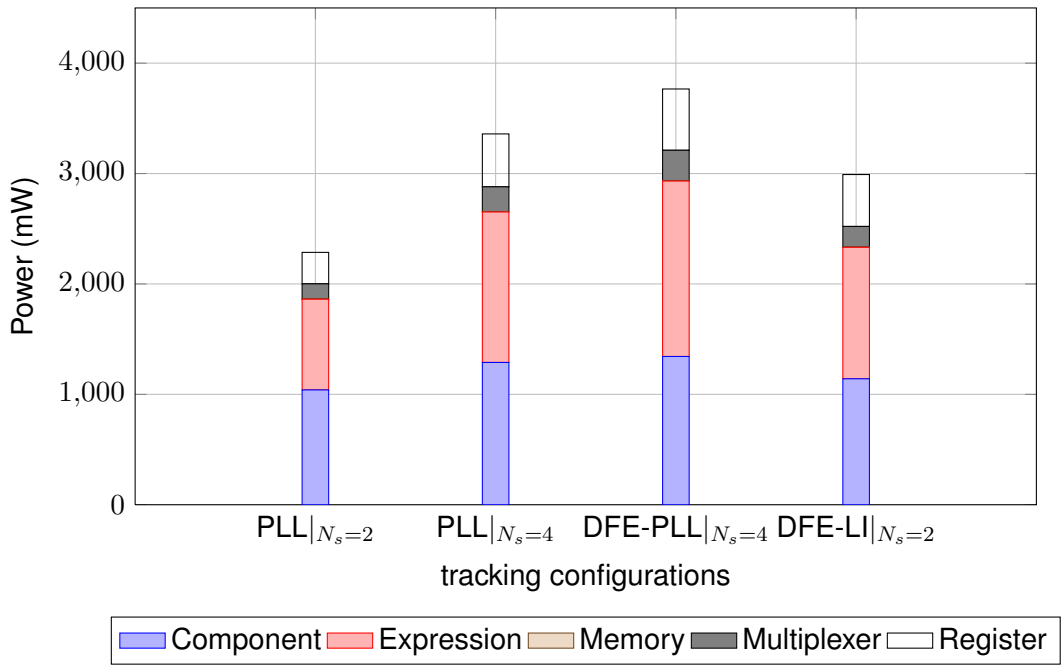


Figure 8.24: 255-bit code HLS power estimate comparison between PLL, DFE-PLL, and DFE-LI

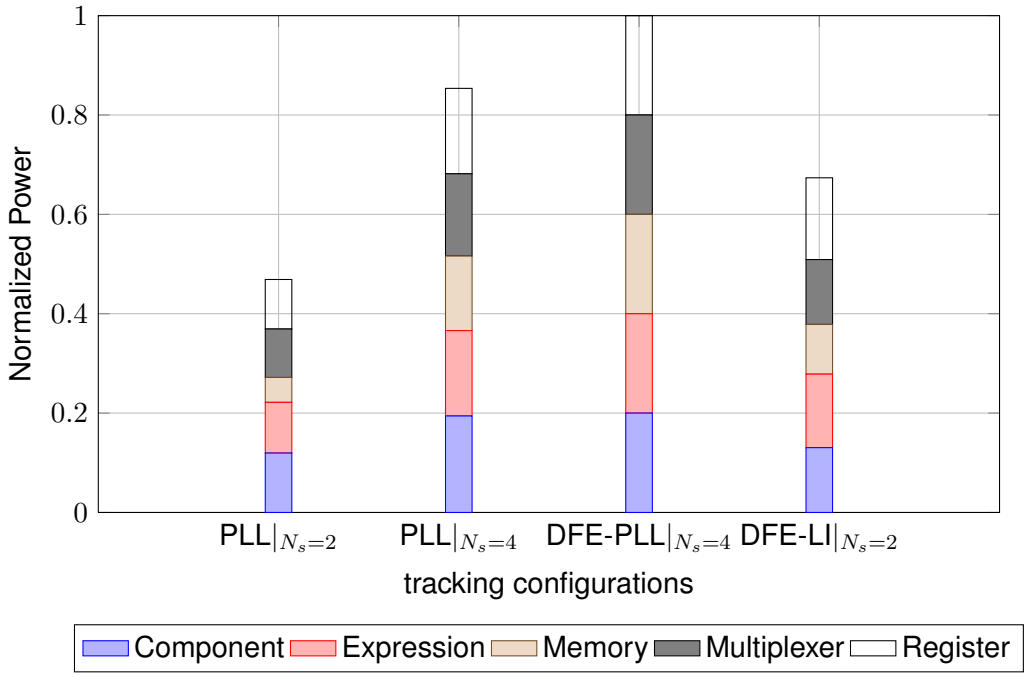


Figure 8.25: 511-bit code HLS power estimate comparison between PLL, DFE-PLL, and DFE-LI

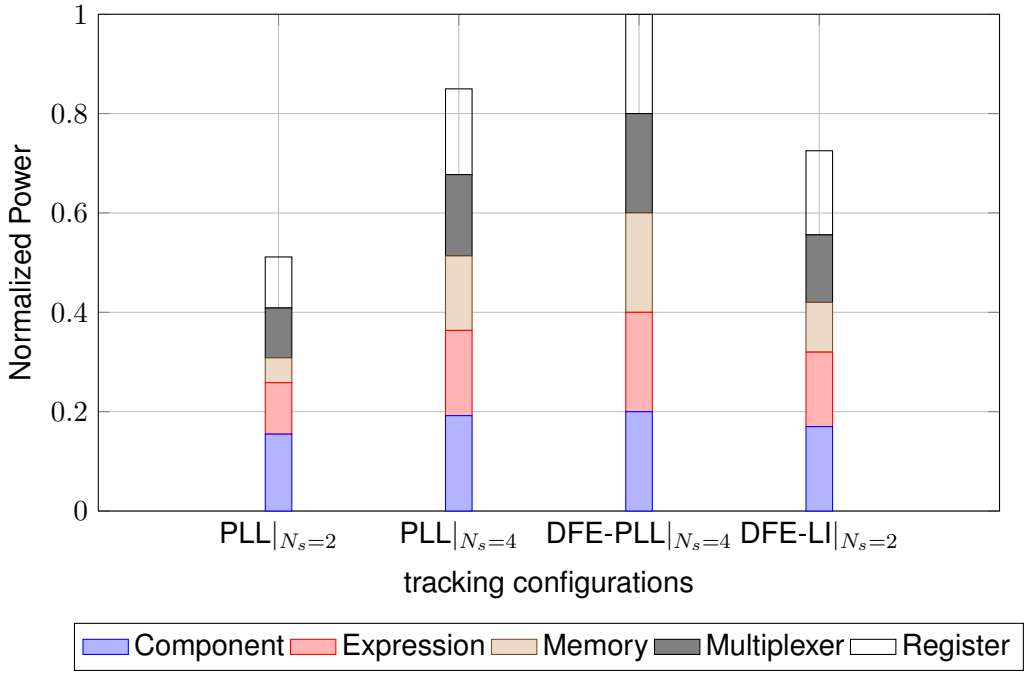
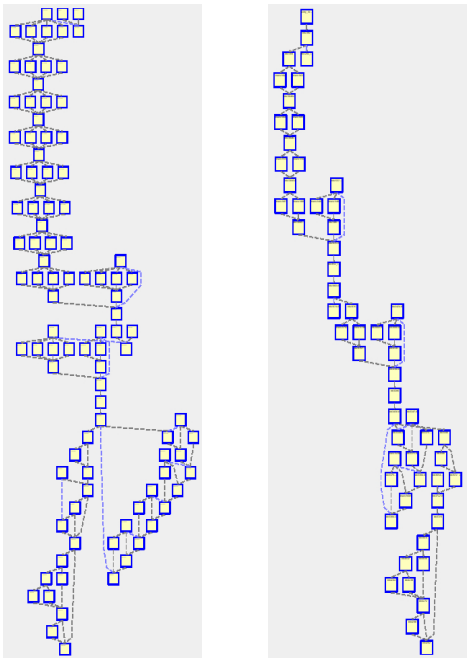


Figure 8.26: 255-bit code HLS power estimate comparison between PLL, DFE-PLL, and DFE-LI

ment of the nodes and their connections reflect the order of execution within the final compiled code [45]. Naturally, an HLS tool will strive to yield the best possible CDFG model of the code subject to various constraints supplied by the informed user. Thus inspecting the final coarse-grained CDFG model of an algorithm gives significant insights into the constraints the high-level directives have on compilation. Therefore, similar to above, the proposed architecture is incrementally analyzed with respect to the chip oversampling rate. The area characterization has already shown that code length has little effect on area under the proposed architecture. The incremental analysis is meant to build understanding of the behavior of the chosen chip-oriented architecture as we progressively insert the processing functions which constitute the tracker configurations derived earlier.

Core CDMA Engine



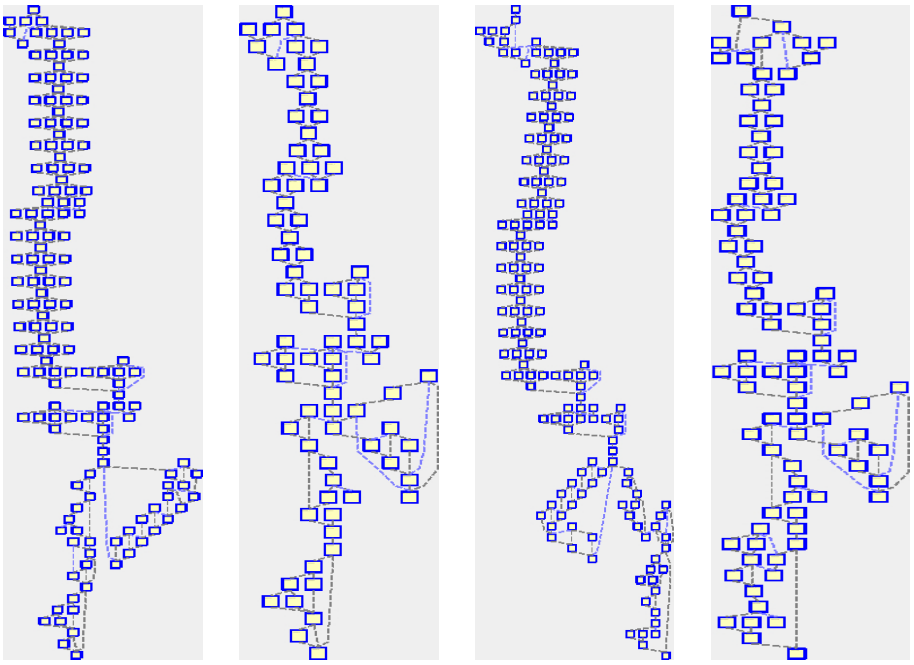
(a) $N_s = 4$ (b) $N_s = 2$

Figure 8.27: CDFG for Core CDMA Engine

The CDFGs of the Core CDMA Engine are examined in figure 8.27 for the two chip oversampling rate cases. As touched upon previously, a large portion of the channel-based, recursive interference cancellation is purely sequential, with vector data dependencies. When applying cyclic memory partitioning and loop unrolling by the chip oversampling rate, the order of execution portrayed by the CDFGs is spread horizontally, indicating faster execution—In figure 8.27b, this is evident by the sequential pairs of tasks which feature in the upper part of the graph. In figure 8.27a, this is illustrated by the horizontal, concurrent (i.e. independent) quadruple tasks. These tasks converge to a subsequent node which in turn dispatches yet another concurrent set of four nodes later. The bottom part of both SDFGs contains naturally-occurring parallelism as can be inferred from the branch. This is attributed to the latter part of the algorithm where after

estimating the current chip, channel update and despreading can be readily overlapped.

PLL



(a) PLL$|_{N_s=4}$ (b) PLL$|_{N_s=2}$ (c) DFE-PLL$|_{N_s=4}$ (d) DFE-PLL$|_{N_s=2}$

Figure 8.28: CDFG for PLL, DFE-PLL trackers

The CDFGs of the PLL-only tracker configuration are given in the left-hand-side of figure 8.28. The PLL has understandably lengthened the graph adding more latency to the final adaptive operation. The reliance of the PLL on the circular signal vector results in added multiplexing logic which is not as straightforward to schedule when compared say to the LI. The LI plugs into the front-end of the architecture seamlessly with lesser implications on other logic. Figures 8.30b & 8.30a do not differ markedly, apart from a factor of two reduction in the scope of concurrency extracted from cyclic partitioning and unrolling.

DFE-PLL

With the inclusion of the FF filter, the CDFGs of the DFE-PLL tracker configuration are shown on the right-hand-side of figure 8.28. When compared to that of the PLL-only configuration, the graphs for both chip oversampling rate cases have few additional tasks at the beginning for filtering. The tails of the graphs are also appended with tasks responsible for driving the LMS update tasks. Note that the lower part of the CDFG of the DFE-PLL tracker at four oversampling rate is mirrored when compared to the respective PLL tracker graph. This is not an anomaly and the two graphs remain functionally equivalent in the context of spatial execution. That is, the reproducibility of the chip-oriented architecture is upheld in both graphs.

DFE-LI

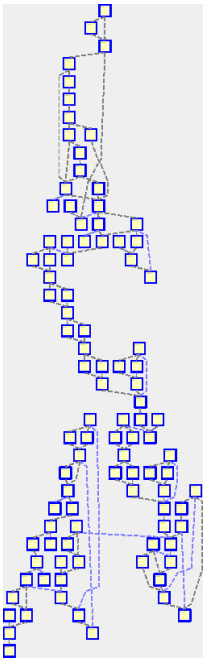


Figure 8.29: CDFG for DFE-LI tracker

Here, the PLL is removed and replaced by the LI stage. As discussed earlier, the DFE-LI tracker configuration was defined for $N_s = 2$ only. The resultant CDFG is provided in figure 8.29. The DFE-LI tracker has a very favourable data dependencies. As such, it approaches the throughput of the bare Core CDMA Engine. This is primarily due to the very strong parallelism it exhibits in the latter part of adaptation, where two distinct execution branches can be seen with minimal data coupling. The DFE-LI tracker is therefore not only the most accurate in terms of Doppler correction, but also the most amenable to a concurrent form of realization. However, from a numerical precision standpoint, the performance of a hybrid floating-point interpolation and BFP CDMA adaptation remains to be investigated as will be discussed in the BFP case study provided.

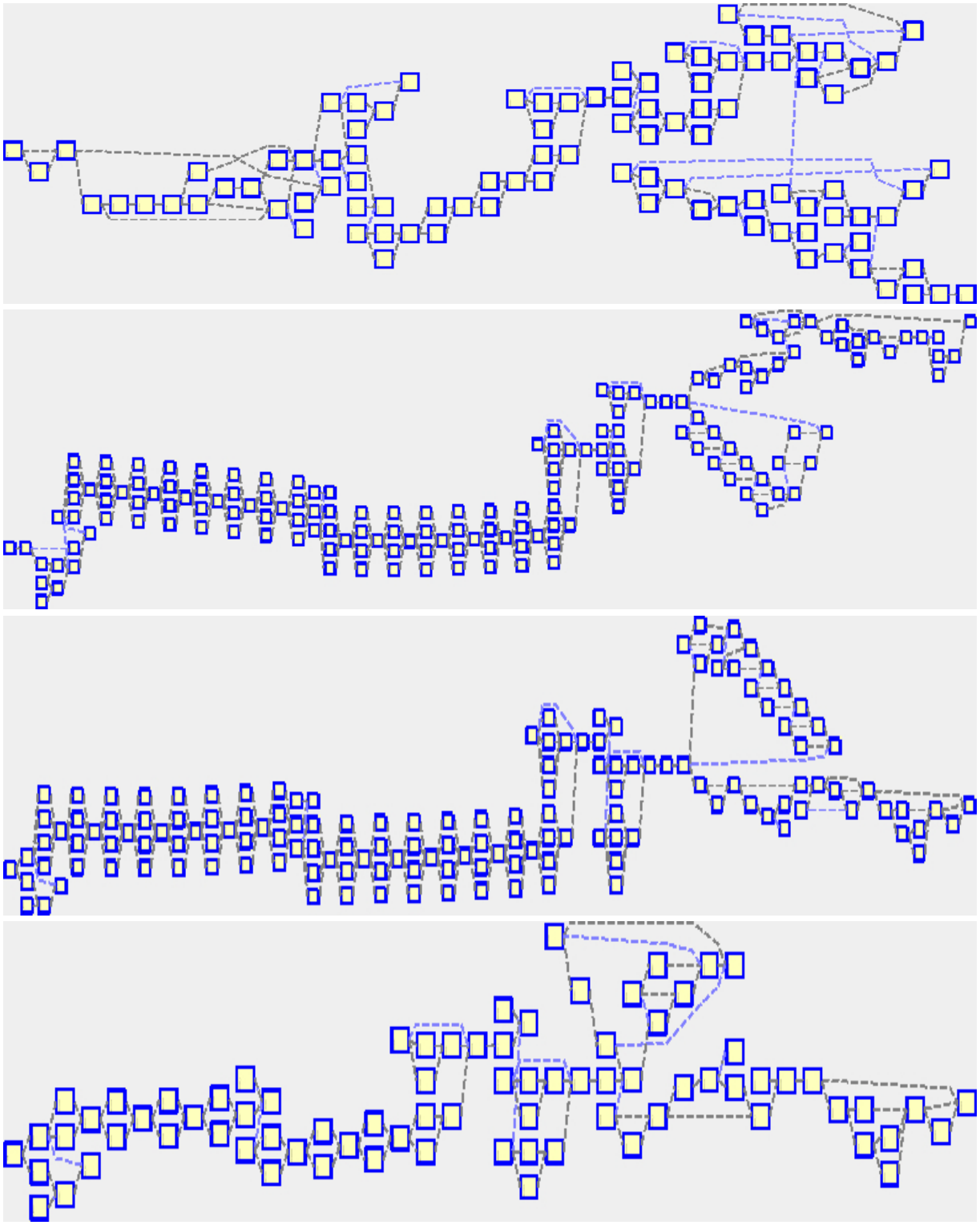
Summary

For the sake of visual comparison, the CDFGs of all tracker configurations are cascaded side-by-side in 8.30.

8.6 BFP

The Block-floating point (BFP) data representation [69] is a hybrid format intended to strike a balance between the wide dynamic range of floating-point computations, and the efficiency of fixed-point processing.

Under the BFP data format, incoming data is partitioned into nonoverlapping blocks of L samples. Each block has a common exponent derived from the sample of greatest magnitude.



(a) PLL|$_{N_s=2}$ (b) PLL|$_{N_s=4}$ (c) DFE-PLL|$_{N_s=4}$ (d) DFE-LI|$_{N_s=2}$

Figure 8.30: CDFG of tracking configurations

$$\begin{aligned} \mathbf{x} &= [x_1, \ldots, x_L] \\ \mathbf{x} &= [x_1^{\leftrightarrow}, \ldots, x_L^{\leftrightarrow}] \cdot 2^{\gamma} \end{aligned} \tag{8.3}$$

where

- $x_l^{\leftrightarrow} = x_l \cdot 2^{-\gamma}$ is the scaled sample i.e. mantissa
- $\gamma = \lfloor \log_2 x_{max} \rfloor + 1 + S$ is the block exponent
- $x_{max} = max(||x_1||, \ldots, ||x_L||)$ the sample whose magnitude is maximum in any given block
- S is a scaling factor for the prevention of overflow during subsequent operations

The scaling factor reduces the magnitude of any mantissa within a block to the range $0 \leq ||x_1|| < 2^{-S}$. This factor should be designed with the subsequent computations in mind in order to safely maximize the dynamic range of fixed-point arithmetics.

Noting the vectorized nature of the proposed algorithm, the single exponent of the incoming data will propagate throughout adaptation. That is, all DSSS vectors will be computationally derived from the incoming data, namely: the channel, the interference-free signal, the input signal, and the average signal. This affords a large scope of area and power reduction compared to the case when data is represented in floating-point format.

It can be shown that with proper care taken, the added difficulty imposed by this formulation can be overcome even for the more complex case of adaptive filters. This is because in adaptive filters, the evolution of coefficients in time mutually couples filtering with weight updating [92]. Indeed, recent works on BFP have demonstrated its viability in the domain of adaptive filtering [36, 91] and the corresponding realization [37, 119, 93].

The characterization of area, power, and throughput was conducted earlier using nominal fixed-point precisions acting as interim “placeholders”. As touched upon earlier, thorough analysis under dynamic adaptation is needed in order to realize a numerically working system. This iterative *simulation driven* [31] process is involved and laborious. However, for the sake of completion, commentary on the feasibility of this process under the proposed approach will be supplied next.

8.6.1 BFP Issues

1. *Memory*: At the expense of using additional ping-pong buffers, non-blocking reads and writes [48] can be used to dynamically scale data in conjunction with the incoming sample precision as to maximize the fixed-point dynamic range. This rescaling will have to be performed on the input signal vector and the average signal vector. The rescaling of the channel can be further fused with the update loop using a complex barrel shifter.

2. *Hybrid Arithmetics*: One powerful feature of HLS design methodology is the ability to mix numeric types, switching between floating-point and fixed-point on demand. With some area and power penalties, this could be further capitalized on to ease and simplify the realization of the proposed algorithm's update. The main point, however, is to keep the Core CDMA Engine operating in BFP, due to its intensive nature.

3. *Lag*: No additional latency will be incurred by rescaling since it can be overlapped with computations [40]. Such operation would happen only once every code length when SNR increases or decreases with range. Nevertheless, a short lag does take place from a real-time velocity inference point of view. This is because a full buffer will have to be searched for the sample with maximum magnitude before processing can commence. This lag will precisely equal a code length at the system's chip rate. For all practical purposes, a lag value of 25 ms should not hamper the application-level user experience. Further techniques such as extrapolating the measured velocity [28] could be used in order to mitigate against this lag if so desired.

8.6.2 Case Study – PLL

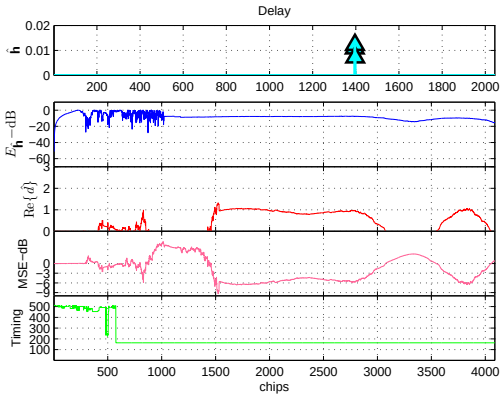
In order to validate the proposed concept, a case study is provided for a PLL operation utilizing BFP. For simplicity, the stimulus fed to the algorithm is a static dataset collected from a stationary transmitter-receiver pair. The objective is to show that BFP can be applied in the ABU band.

Figure 8.31 illustrates the PLL algorithmic configuration of the static dataset for both the block and the base floating-point implementations. Eight codes are processed. As discussed in chapter 6, the instance at which the PLL becomes active is delayed by two codes in order to allow for at least a code-length channel gain. The PLL in both cases is driven identically with the tracking constants $K_1 = 5 \times 10^{-4}$ and $K_2 = K_1/10$.

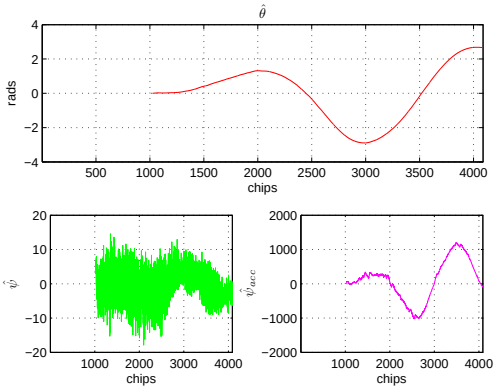
Inspecting figure 8.31a, it can be seen that the BFP despreader begins to stabilize starting at the fourth code. The energy in the BFP despreader later begins to oscillate between being in-lock and $\frac{\pi}{2}$ out-of-phase (in the not-shown imaginary part). On the other hand, the floating-point despreader of figure 8.31c takes roughly an additional code to converge but remains stable thereafter. The deviation in operation between the two numerical formats is starker when comparing the actual PLL primitives in figures 8.31b & 8.31d. In conclusion, the BFP behaviour of the PLL is quite different from the equivalent floating-point at the same drive. Thus, although a BFP operation is feasible in principle, updated characterization of performance, and dynamic parametric interplay have to be first undertaken. The arbitrary precision data types in HLS readily have a wide support of rounding and saturation logic at the hardware level.

8.7 Applicability of Results

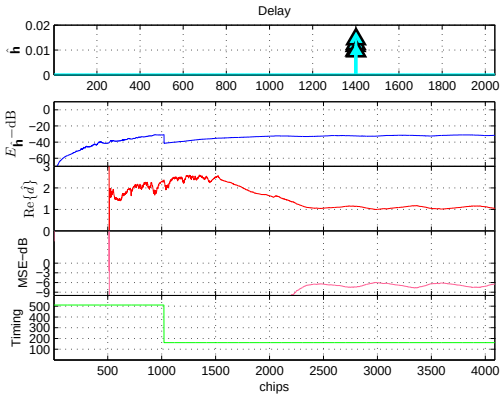
Commentary on the representativeness of this chapter's results is supplied here.



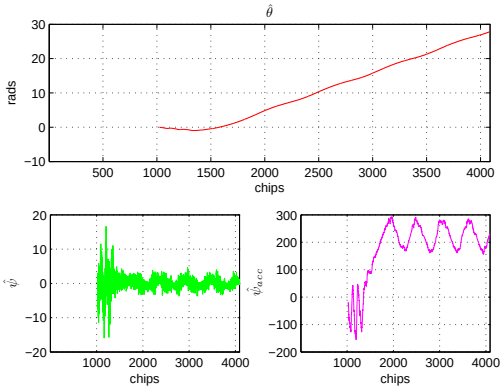
(a) BFP Timing



(b) BFP PLL



(c) Floating-point Timing



(d) Floating-point PLL

Figure 8.31: BFP PLL performance

In regard to area and power HLS estimates (sections 8.5.1 & 8.5.2), it is stressed that some variations would be expected in a well-engineered, production-level implementation. To shed more light on this point, consider the fidelity of magnitude estimation during the channel update procedure. Under dynamic stress analysis, it might be concluded that cruder approximation could be tolerated by means of employing a stricter truncation threshold. Comparing the error characteristics and the implementation efficiency of binary [16] and equiripple [47] magnitude estimators demonstrates the point. And observing the expression-balanced nature of the on-the-fly operation, a decent reduction in resource requirements could be attained. Another example can be seen in the average signal formation procedure where a conditional complex accumulation loop is unrolled. It may be feasible to reduce the accumulation bit-width on the presumption that the ABU channel is always sparse. As such, very limited accumulation would actually take place. A third example is the precision of the CORDIC core and its effect on the dynamic phase tracking behaviour of the PLL.

Therefore, the point to make with respect to HLS area and power estimates is that these metrics are likely to undergo some changes if further work were to be pursued in this direction. With this mind, the normalized differential results comparing tracker configurations provide better quantitative measures that are perhaps more indicative of

area and power requirements.

In regard to the BFP operation (section 8.6), rescaling will be required for DSSS and FIR entities as discussed earlier, which will increase area and power slightly.

The CDFG analysis (section 8.5.3) remains perfectly valid under any further modifications to be introduced. While fixed-point optimization and scaling will influence HLS metrics, the order of execution of algorithmic operations is decoupled from alteration to the mathematical mechanics of the code. The CDFGs of tracker configurations are to be viewed as guidelines on the real-time amenability of the proposed algorithm and its chip-oriented architecture.

8.8 Summary

This chapter studies the architectural feasibility of an ABU motion tracker, which is a prerequisite to ad hoc tracking. A thorough architectural exploration of various tracker configurations is conducted using a high-level synthesis (HLS) methodology. The study demonstrates that ABU motion tracking is well within the reach of modern FPGA devices. A chip-oriented architecture is proposed and exhaustively characterized. A block floating-point (BFP) numerical format is also proposed, and its implications on system-level issues are reviewed. The area, power, and throughput metrics of the proposed chip-oriented architecture at 511-bit code length are summarized in table 8.1, for both the absolute and normalized values.

Apart from algorithm-related findings, the tool's reproducibility of results is noted. It is the author's belief that HLS indeed provides a very viable non-tedious alternative to architectural exploration for complex designs for the agile gauging of algorithmic implications on area, power, and throughput.

Table 8.1: HLS metrics comparison between PLL, DFE-PLL, and DFE-LI[a]

Metric	TRACKING CONFIGURATION			
	PLL$\|_{Ns=2}$	PLL$\|_{Ns=4}$	DFE-PLL$\|_{Ns=4}$	DFE-LI$\|_{Ns=2}$
Absolute				
BRAM	17	33	41	25
DSP48E	34	62	72	59
FF	8,539	14,156	15,398	10,955
LUT	15,071	25,029	28,473	19,654
Power	2362	3925	4336	3066
Throughput	28.54	22.37	20.81	31.45
Normalized[a]				
BRAM	0.415	0.805	1.00	0.610
DSP48E	0.472	0.861	1.00	0.819
FF	0.555	0.919	1.00	0.712
LUT	0.529	0.879	1.00	0.690
Power	0.545	0.905	1.00	0.707
Throughput	1.371	1.075	1.00	1.511

[a] 511-bit code length
[b] w.r.t DFE-PLL
[c] Xilinx Virtex-5 XC5VSX94T
[d] 100 MHz clock, 1.25 ns clock period uncertainty

CHAPTER 9

Conclusion

This dissertation has developed the novel modality of airborne broadband ultrasound (ABU) in support of ad hoc, mobile tracking applications. This chapter summarizes the research results and findings, and discusses avenues of future research on the ABU modality.

9.1 Summary

Development of indoor trackers has been extensively undertaken for systems requiring low-rate tracking, and those in need of high-rate tracking. Chapter 2 examines past trends in-depth along with their enabling technologies. The emerging emphasis on ad hoc, mobile, and reliable tracking is shown to be difficult to achieve under current low-rate tracker methods, and high-performance, high-rate trackers that are infrastructure-reliant and expensive to deploy and calibrate. Thus, tracking fidelity and rapid deployment can not be simultaneously attained. Despite its potential for accuracy, robustness, and scalability, previous work on ABU has not addressed the challenges posed by (1) embedded, real-time realization and (2) susceptibility to the motion-induced Doppler effect.

Chapter 3 describes an all-digital ABU transmitter prototype architecture. The architecture allows for mobile transmitter nodes with limited resources. The architecture also supports system-level parameterization so that it may be tailored to the signalling needs of a particular application. For instance, the system's chip rate can be doubled for decreased ranging message duration and increased Doppler trackability.

An efficient despreader kernel is designed and implemented in chapter 4. The kernel enables distributed nodes to simultaneously detect ranging messages from other co-located users. This true multiple access arrangement is evaluated through co-simulation

using real dataset gathered using ABU transmitters and receivers. The multiple access performance was found to be accurate to just under 3 cm with a return rate of around 65%, which is in accordance with previously characterized effects under this deployment using offline workstation-class processing. Additionally, the efficient core can be replicated many times on one reconfigurable fabric to perform complex functionalities such as multiuser beamforming in real-time under current hardware capabilities.

Chapter 6 considers the problem of estimating velocity from Doppler-distorted ABU signals. Computationally-intensive adaptive processing is proposed, drawing on methods from high-rate underwater acoustic communications. The operation of such an adaptive receiver is analyzed with respect to three phase tracking strategies (PLL, FF, and LI) and their combinations. Grounded in an empirical analysis, understanding of the ABU adaptive operation has been researched for the first time, with promising results from two tracker configurations, namely: DFE-PLL and DFE-LI. This serves to highlight that ABU can supply accurate measurements indicative of users' mobility indoors. The use of an algorithmic primitive for the inference of acceleration under dynamic stress conditions is proposed. This is envisioned to be key for realizing a resilient ABU motion tracker with no prior assumptions on high-order motion moments. This is important because of the range of accelerations possible in human movement indoors.

Having demonstrated motion inference by means of ABU Doppler processing in chapter 6, the feasibility of such a tracker is studied in chapter 8, from a computational architecture standpoint. This is of primary concern to ad hoc tracking. Using a high-level synthesis (HLS) methodology, comprehensive architectural exploration of various tracker configurations is conducted. The study illustrates that ABU motion tracking is well within the reach of modern DSP-equipped FPGAs. A chip-oriented architecture is proposed and thoroughly scrutinized. A block floating-point (BFP) numerical format is also proposed, and its implications on system-level issues are reviewed. The DFE-LI configuration is found to be the most favourable in terms of area and velocity accuracy. Comparatively, the inclusion of the PLL functionality can be accommodated, but less straightforwardly. However, as pointed out in chapter 6, the DFE-LI operation is influenced by high-order motion moments which are prevalent in the ABU band under human-scale movements. The DFE-PLL seems to be more able to cope with high-order motion moments.

9.2 Future Work

Perhaps most immediate of all, work for creating prototype ABU nodes to operate in the static mode can now commence, utilizing results from chapters 4 and 5. A number of FPGA technologies currently exists to facilitate prototyping, ranging from power efficient FLASH-based architectures for unobtrusive, small tags to high-resource, SRAM-based for handheld units with more battery capacity.

The receiver prototype is not ideal [61]. The current design makes use of a large-valued resistor in a current-to-voltage configuration mode. The noise figure of a signal chain is vastly determined by that of the input stage. Excessively large resistor values

translate into large thermal noise. An improved setup has already been implemented for a *chip-spaced* piezo film array using a T-shaped resistor arrangement and is shown to produce cleaner signals [101]. A more complex approach would be to redesign the receiver using direct current conversion at the ADC instead of voltage conversion, which is becoming more widely available. The influence of the receiver's noise was briefly touched on in chapter 6. It would be interesting to see whether the improved receiver circuitry could have any influence on adaptation.

Chapter 6 discussed the possibility of tighter dynamic control for the interpolation index utilizing the IA estimator. This hypothesis should be investigated in future work. Methods from control theory may provide valuable input to the problem. Although this is somewhat close in concept to a tri-combinatory interpolation-DFE-PLL configuration [55], the hypothesis has potentially considerable silicon area advantage. The reader is reminded that one finding from chapter 7 is the higher resource cost associated with the PLL's operation. Additionally, there would be high memory storage requirement in an interpolation-DFE-PLL configuration, owing to the need for a bank of filters for the interpolator. Nonetheless, it may be necessary to consider such a tri-combinatory configuration in future research in order to gauge the practical implications on the fidelity of tracking in the ABU band. If clear advantages are obtained, such functionality could be reserved for a fixed base-station implementation with lesser resource restrictions.

Chapter 8 proposed an efficient implementation based around block floating-point (BFP). As discussed earlier, it is therefore necessary to investigate the performance of adaptation using BFP under dynamic conditions. This is likely to be a laborious operation and should be deferred until a genuine commitment for building a real-time ABU motion tracker is reached. Note that the numerical design optimization in NASA's Apollo mission consumed around 30% of total development resources [76].

The support of non-blocking memory accesses in the AutoPilot HLS tool has only been added recently to release version v2010.a. The inner workings of the application programming interface (API) for streaming memories are complex and not documented. Access to the support team of the tool is very restricted under academic licenses. Therefore, using the tool to the fullest extent mandates industrial-class support by the vendor. For instance, full support would be necessary in order to arrive at the added memory requirements of BFP rescaling and data output which should overlap computations.

On the algorithmic front, there are three items for future research. Two of these items are related to array signal processing for ABU. Figure 9.1 shows a uniform linear array (ULA) with receiver elements spaced by half-a-chip, assuming a system chip rate of 20 kHz. The array is designed for complex baseband beamforming which would help boost the poor transduction efficiency of piezo films.

To keep processing amenable to real-time operation, there are two functionalities for which the array is to be utilized. First, the envisioned direction-of-arrival (DOA) method to be applied on the chip-spaced array is basic, low-resolution broadband FIR beamforming. It is worth stressing that, from a system-level perspective, the method has to fulfill two main requirements: (1) real-time realization (2) unhindered multiuser capability. The first

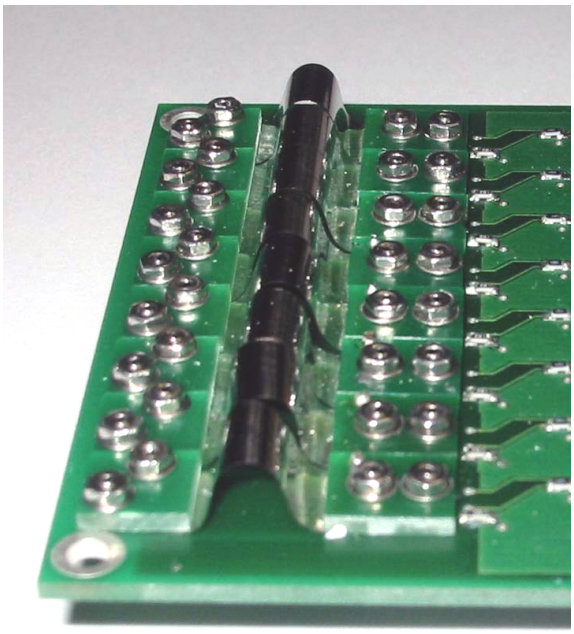


Figure 9.1: PVDF transducer elements with half-chip spacing.

requirement is readily supported by the proposed multiuser despreader kernel of chapter 5. For the second requirement, the DOA functionality must *scale irrespective* of the array's number of elements. Many super-resolution methods—in addition to being very computationally expensive for real-time multiuser monitoring—suffer from performance degradation when the number of impinging sources exceeds the dimensionality of the array. It is also postulated that from a usability viewpoint, dividing the interactive environment into a number of spatial sectors is likely to be sufficient for most of the usage scenarios sought after. Second, as is customary in UWA systems, an adaptive spatial combiner stage would boost the performance and reliability of Doppler tracking as a direct consequence of signal-to-noise (SNR) ratio enhancement. Further, and since in this ABU system element-spacings have spatial significance, monitoring the state of the adaptive combiner could potentially allow us to derive a measure of users' orientations, relative to a tethered basestation node.

Finally, the remaining item on the agenda for future research is *binary* Doppler-tolerant reception. Outside the scope of ad hoc operation, and in scenarios where ABU infrastructural installations can be assumed, wired-wall or ceiling-mounted nodes may be employed with continuous multiuser transmissions. The continuous stream would resemble that of GPS in that it consists of *data* frames with acquisition preambles. Consequently, the need for RF synchronization would be removed altogether. Such topology is infinitely scalable. However, this comes at the expense of equipping receiver tags in this topology with *binary* Doppler-tolerant reception capabilities. Such receiver effectively doubles the computational complexity i.e. silicon area. It has been reported in the UWA literature. Dubbed the hypothesis feedback equalizer [133], it entails having two parallel adaptation branches all the time scanning for two [binary] hypotheses ahead of the end

of any given symbol in order to mitigate against data transitions.

APPENDIX A

PLL Performance Plots

Table A.1: Summary of PLL performance for data set 1

RESULTS		ALG. PARAMS.			
stable	spread	$\mathcal{E}_{E_h}$	N_s	$K_{1,2}$[a]	$DRCE$
YES	MODERATE	0	2	1×10^{-5}	FALSE
YES	LARGE	0	2	1×10^{-5}	TRUE
NO	-	0	2	5×10^{-5}	FALSE
NO	-	0	2	5×10^{-5}	TRUE
NO	-	0	4	1×10^{-5}	FALSE
NO	-	0	4	1×10^{-5}	TRUE
YES	MODERATE	0	4	5×10^{-5}	FALSE
NO	-	0	4	5×10^{-5}	TRUE
NO	-	1×10^{-6}	2	1×10^{-5}	FALSE
NO	-	1×10^{-6}	2	1×10^{-5}	TRUE
NO	-	1×10^{-6}	2	5×10^{-5}	FALSE
NO	-	1×10^{-6}	2	5×10^{-5}	TRUE
NO	-	1×10^{-6}	4	1×10^{-5}	FALSE
NO	-	1×10^{-6}	4	1×10^{-5}	TRUE
NO	-	1×10^{-6}	4	5×10^{-5}	FALSE
YES	BEST	1×10^{-6}	4	5×10^{-5}	TRUE
YES	MODERATE	1×10^{-7}	2	1×10^{-5}	FALSE
YES	LARGE	1×10^{-7}	2	1×10^{-5}	TRUE
YES	LARGE	1×10^{-7}	2	5×10^{-5}	FALSE
NO	-	1×10^{-7}	2	5×10^{-5}	TRUE
NO	-	1×10^{-7}	4	1×10^{-5}	FALSE
NO	-	1×10^{-7}	4	1×10^{-5}	TRUE
YES	MODERATE	1×10^{-7}	4	5×10^{-5}	FALSE
NO	-	1×10^{-7}	4	5×10^{-5}	TRUE

[a] $K_2 = K_1/10$


HI-RES PLOTS SNIPPED

APPENDIX B

DFE-PLL Performance Plots

Table B.1: Summary of DFE-PLL performance for data set 1[a]

RESULTS	ALG. PARAMS.				
motion correlated	N_s	μ_{LMS}	L_{ff}	$K_{1,2}$[a]	$DRCE$
YES	4	1×10^{-4}	2	1×10^{-4}	FALSE
YES	4	1×10^{-4}	2	1×10^{-4}	TRUE
NO	4	5×10^{-4}	8	5×10^{-4}	FALSE
YES	4	5×10^{-4}	8	5×10^{-4}	TRUE
NO	4	5×10^{-4}	10	5×10^{-4}	FALSE
YES	4	5×10^{-4}	10	5×10^{-4}	TRUE

[a] $K_2 = K_1/10$


HI-RES PLOTS SNIPPED

APPENDIX C

DFE-LI Performance Plots

Table C.1: Summary of DFE-LI performance for data set 1

RESULTS	ALG. PARAMS.			
symmetry score	N_s	μ_{LMS}	L_{ff}	K_p
5	2	1×10^{-4}	8	1×10^{-5}
4	2	1×10^{-4}	8	2×10^{-5}
2	2	1×10^{-4}	8	4×10^{-5}
3	2	1×10^{-4}	8	8×10^{-5}
1	2	1×10^{-4}	8	16×10^{-5}
6	2	1×10^{-4}	8	32×10^{-5}


HI-RES PLOTS SNIPPED

APPENDIX D

Co-simulation

This appendix gives a flavour on how the co-simulation system employed in chapter 5 is built.

Listing D.1: Matlab function for executing a co-simulation session, excerpt of class CoSim

```
% execute a co-simulation session
function exec_session(obj)
   cd(obj.COSIM_PATH);
   for m = 0:obj.nCycles-1
       % write stimulus to memory address 0 to 1023 of input memory
       obj.foo{1}(0:obj.BuffLen-1) = ...
           obj.stimulus_re(1+m*obj.BuffLen:m*obj.BuffLen+obj.BuffLen);
       obj.foo{2}(0:obj.BuffLen-1) = ...
           obj.stimulus_im(1+m*obj.BuffLen:m*obj.BuffLen+obj.BuffLen);

       obj.mutex_sw(0) = 255;  % PHASE 1

       % PHASE 2
       num = 00;
       while (num ~= 255)
           num = obj.mutex_hw(0);
       end

       obj.mutex_sw(0) = 00;   % PHASE 3

       % PHASE 4
       num = 255;
       while (num ~= 00)
           num = obj.mutex_hw(0);
       end

       % read values at memory address 0 to 1023 of output memory
       for n = 1:obj.pOut_NUM
           obj.result_re(1+m*obj.BuffLen:m*obj.BuffLen+obj.BuffLen,n) = ...
               obj.bar{n,1}(0:obj.BuffLen-1);
           obj.result_im(1+m*obj.BuffLen:m*obj.BuffLen+obj.BuffLen,n) = ...
               obj.bar{n,2}(0:obj.BuffLen-1);
       end
   end
   cd(obj.OrigDir);
end % exec_session
```

Listing D.2: Matlab script excerpt for hardware-in-the-loop

```
.
.
.
%% CO-SIMULATE

% co-simulation sequence
if ( initConfig == 0)
    cosim.init_session(1);
    initConfig = 1;
else
    cosim.init_session(0);
end
cosim.load_costim(stim_re, stim_im);
cosim.format_indata();
cosim.exec_session();
cosim.format_outdata();
res_re = cosim.result_re;
res_im = cosim.result_im;
cosim.terminate_session();
.
.
.
```

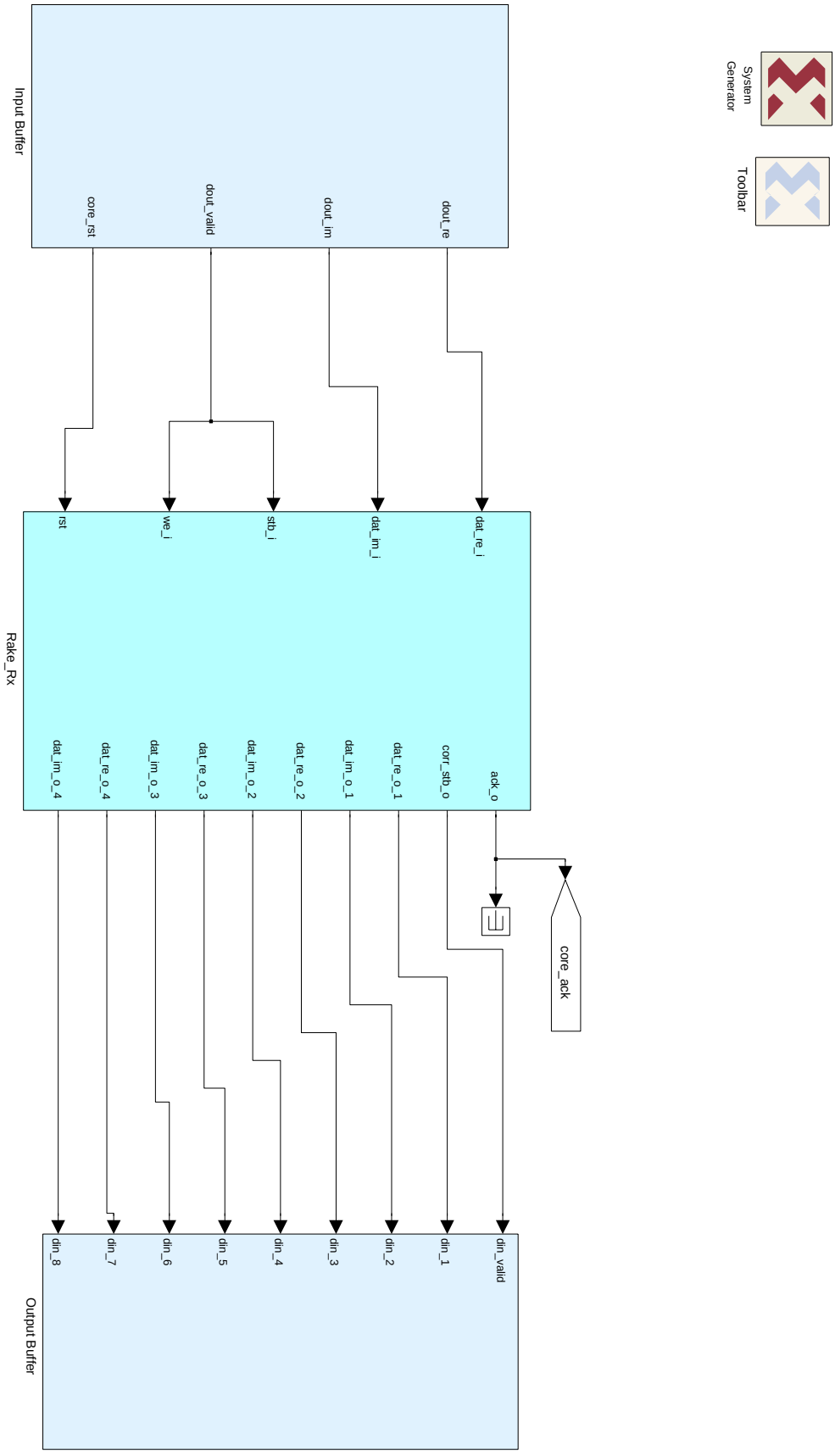


Figure D.1: Co-simulation system top level

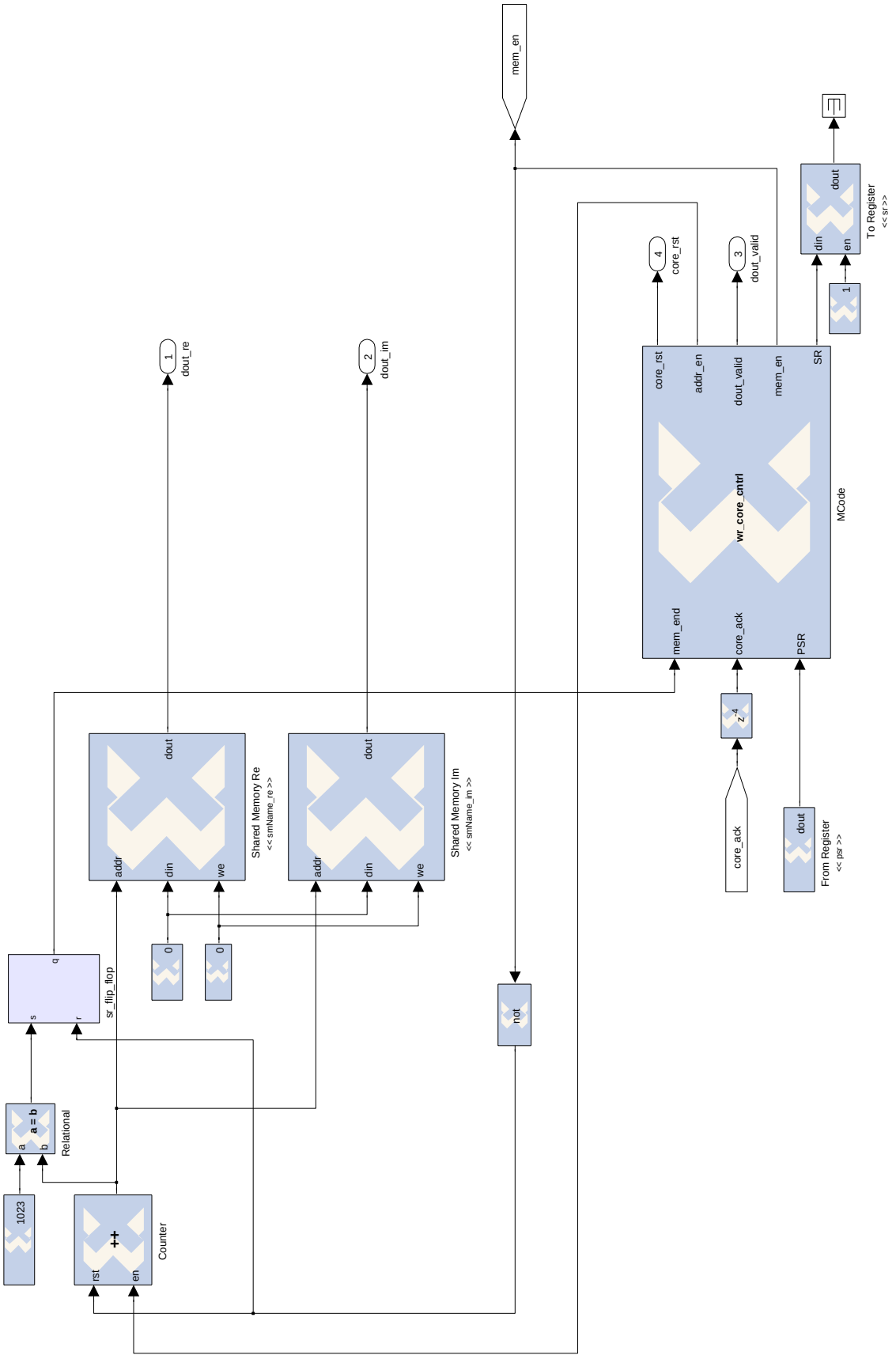


Figure D.2: Co-simulation system input stage

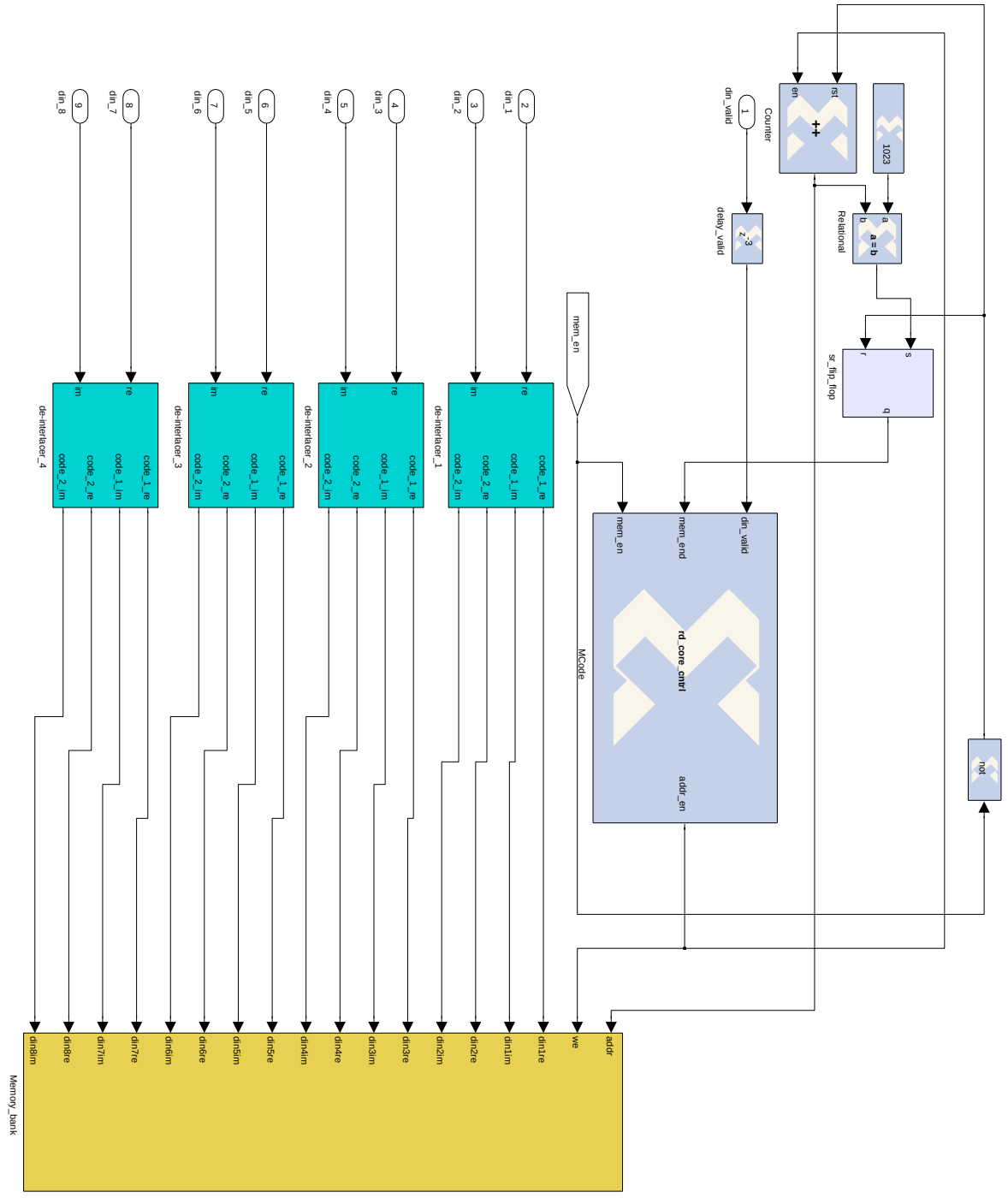


Figure D.3: Co-simulation system output stage

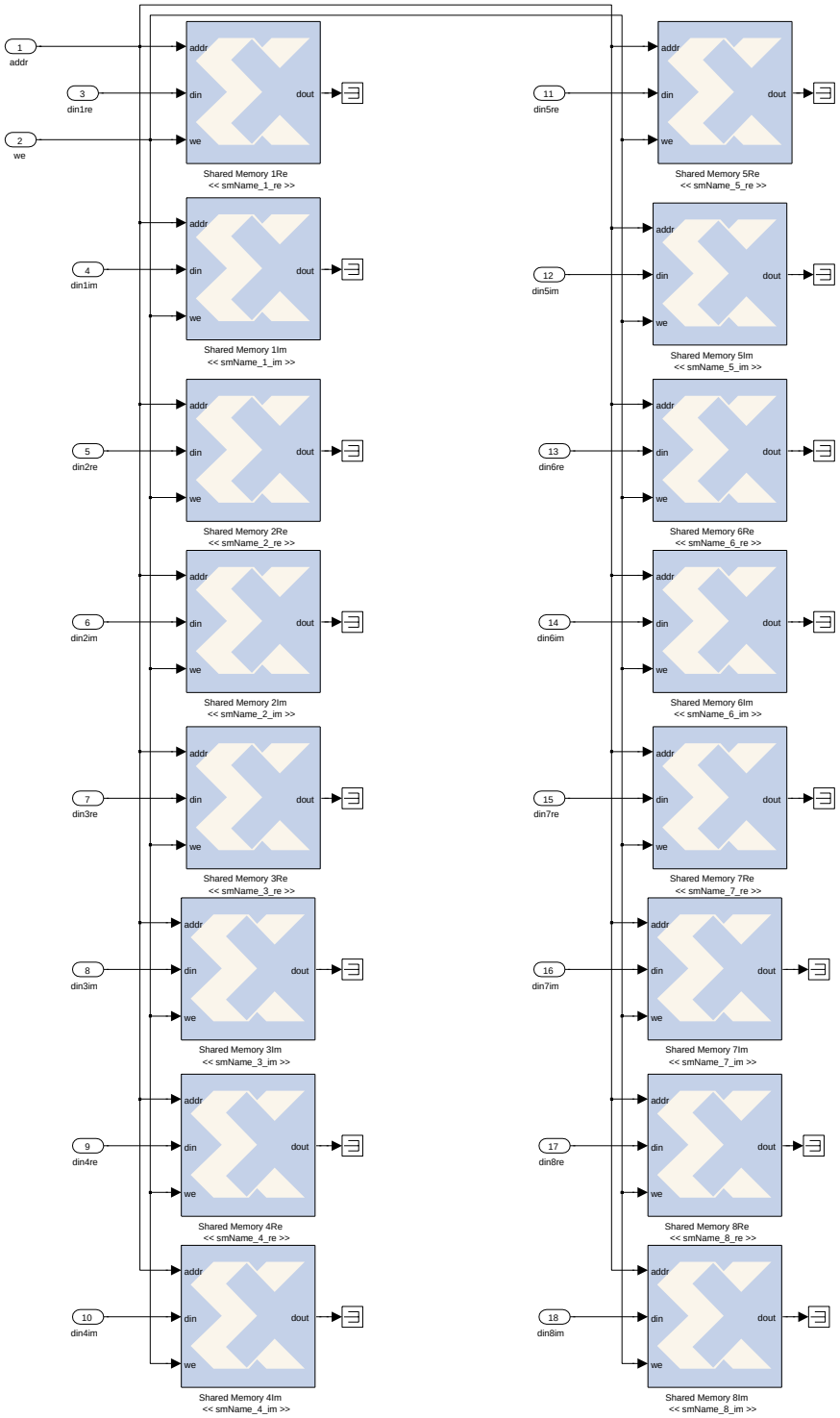


Figure D.4: Co-simulation system output buffer

APPENDIX E

C# Doppler Data Acquisition Program

This appendix illustrate the overall structure of the C# program built for the Doppler experiment. As listing E.1 shows, two classes `reacTIVisonEx` and `AcqNI` are utilized for vision-tracking and data acquisition, respectively. A simple synchronization mechanism is used in the multi-threaded application.

```
1   namespace DopplerExperiment
2   {
3     public partial class reacTIVisonEx : Form, TuioListener{...}
4
5     public class AcqNI{...}
6
7     public struct Fiducial{...}
8
9     public delegate void FiduMovingEventHandler(object sender,
          FiduMovingEventArgs e);
10
11    public class FiduMovingEventArgs : EventArgs{...}
12
13    class HiPre{...}
14  }
```

Listing E.1: Structure of C# Doppler experiment

APPENDIX F

Prototype and Experimental Setup

This appendix shows photos of the ABU prototype system and equipment used in the Doppler experimental setup and data collection.

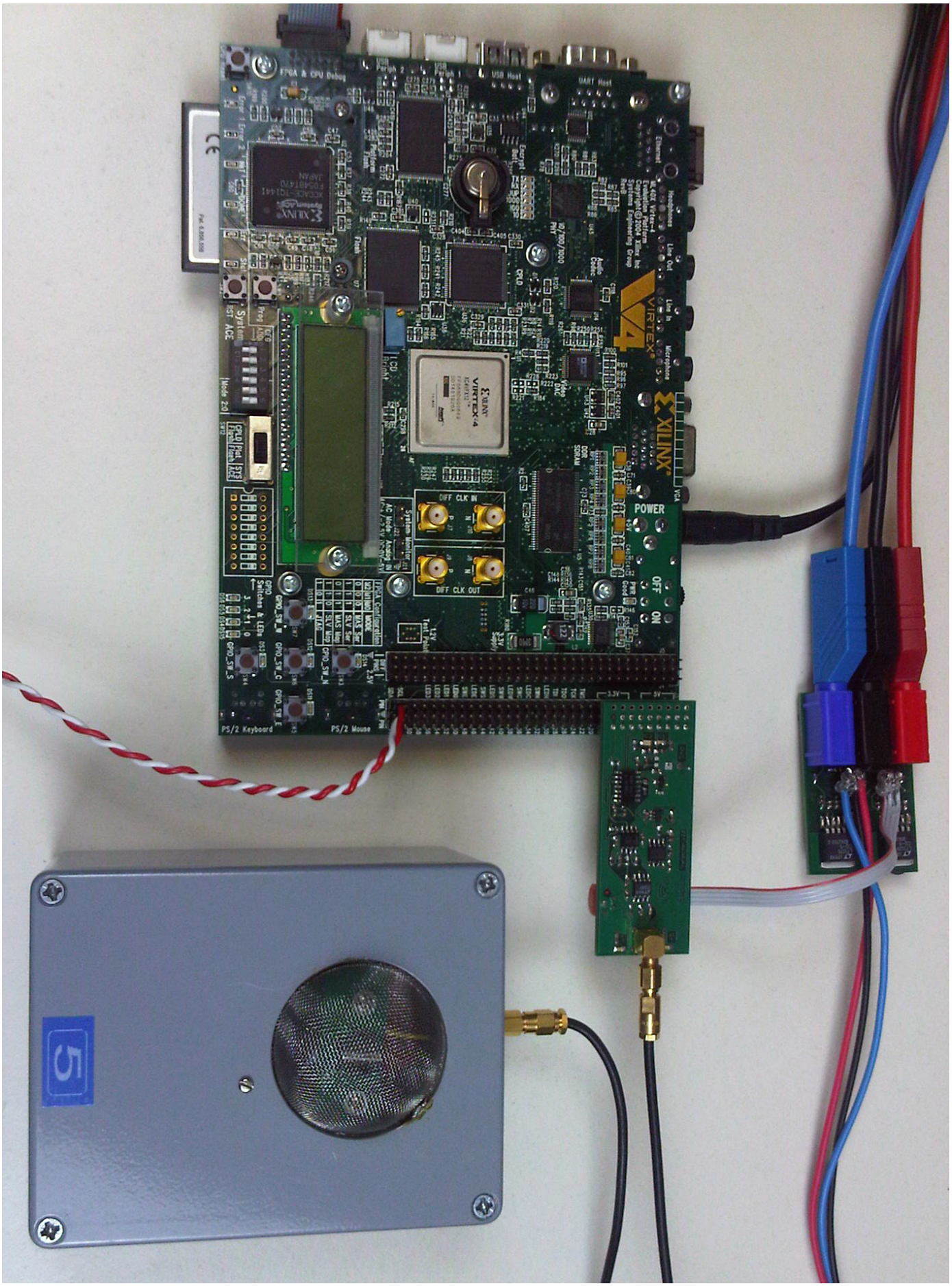


Figure F.1: System prototype

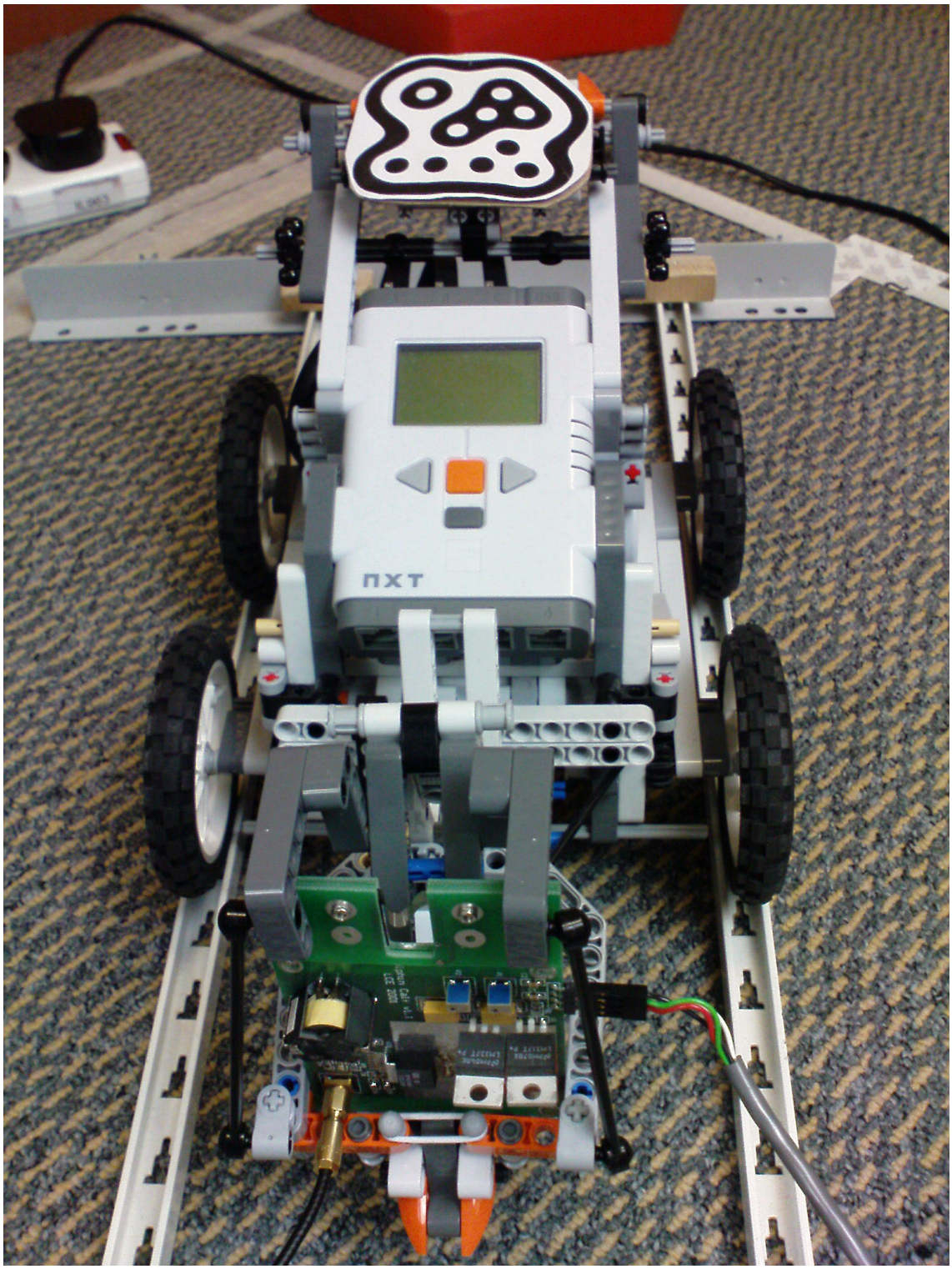


Figure F.2: Robot with mounted transmitter

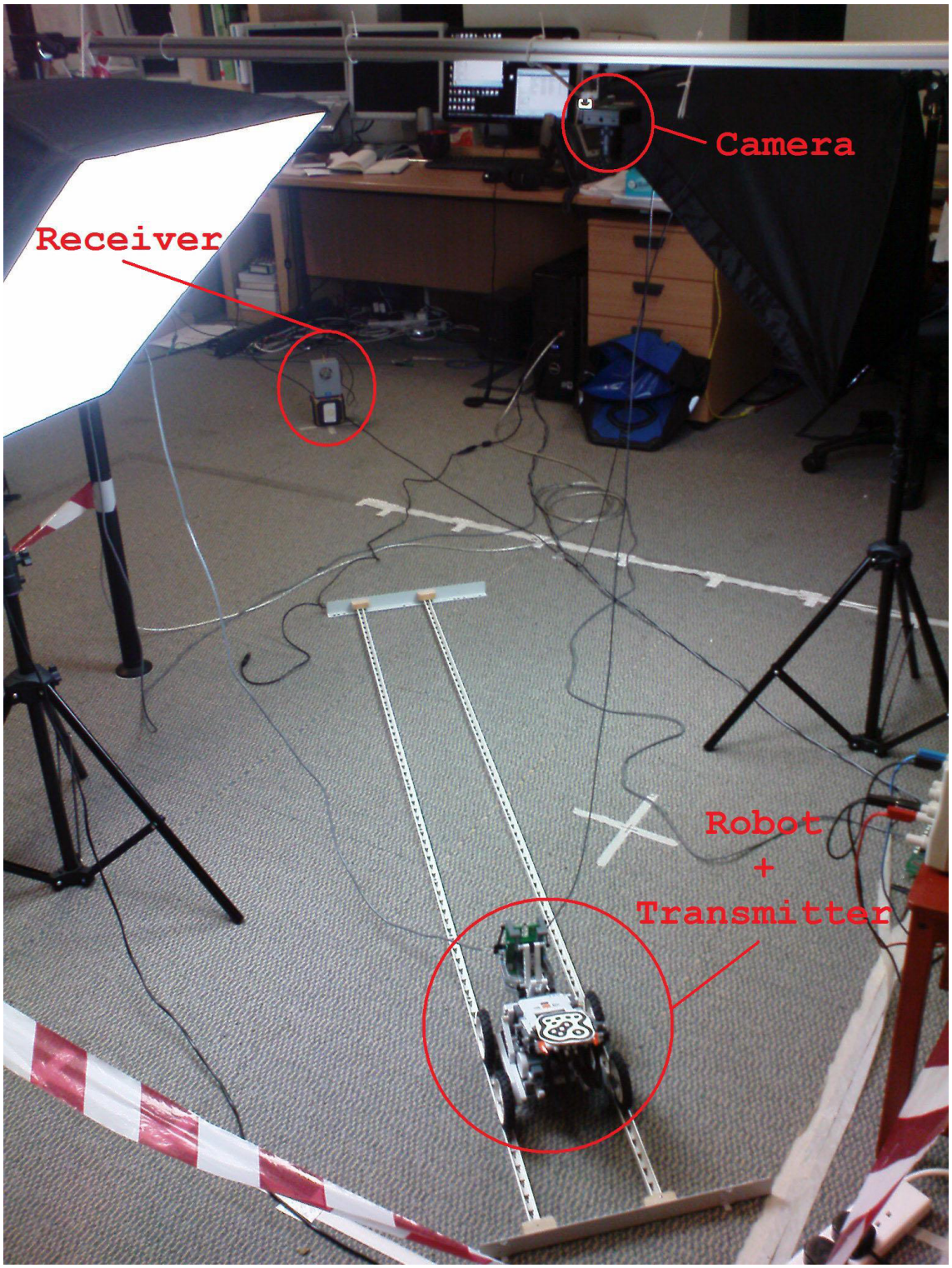


Figure F.3: Doppler data collection setup

APPENDIX G

List of Acronyms

ABU airborne broadband ultrasound

ADC analog-to-digital converter

ASIC application-specific integrated circuit

AWGN additive white Gaussian noise

BER bit error rate

BFP block floating-point

CDF cumulative distribution function

CDMA code division multiple access

CDFG control data flow graph

CORDIC coordinate rotation digital computer

DAC digital-to-analog converter

DF2 direct form II

DFE decision-feedback equalizer

DFG data flow graph

DGC dynamic gain control

DOA direction of arrival

DSP digital signal processor

DSSS direct-sequence spread spectrum

EDA electronic design automation

ESL electronic system level

FDMA frequency division multiple access

FIR finite impulse response

FPGAs Field Programmable Gate Arrays

GDOP geometric dilution of precision

GIS geographic information systems

GPS Global Positioning System

GSM Global System for Mobile communications

HLS high-level synthesis

ICI inter-chip interference

II initiation interval

IIR infinite impulse response

IO input/output

LBS location-based services

LFSR linear feedback shift register

LMS least mean squares

MAC medium access control

MACC multiply-accumulate

ML maximum-likelihood

MMSE minimum mean square error

MSE mean-square error

MUTEX mutual exclusion

NCO numerically-controlled oscillator

PAR place and route

PLL phase-locked loop

PN pseudorandom noise

PRF position reporting frequency

PSD power spectrum density

PVDF Polyvinylidene Fluoride

QoS quality of service

RF radio frequency

RMS root mean square

RTL register transfer level

SIMD single instruction multiple data

SNIR signal-to-noise-plus-interference ratio

SNR signal-to-noise ratio

SoC system-on-chip

SOS second-order sections

TDM time division multiplexing

TDMA time division multiple access

TOA Time of Arrival

TOF time of flight

UWA underwater acoustics

UWB ultra-wideband

WSNs wireless sensor networks

VLIW very long instruction word

Bibliography

[1] Ascension Technology Corporation. http://www.ascension-tech.com/, 06 June 2011.

[2] Bluespec, Inc. http://www.bluespec.com/, 06 June 2011.

[3] Catapult C Synthesis. http://www.mentor.com/esl/catapult/overview, 06 June 2011.

[4] Fakespace Labs, Inc. http://www.fakespacelabs.com/, 06 June 2011.

[5] GCC, the GNU Compiler Collection. http://gcc.gnu.org/, 06 June 2011.

[6] Impulse Accelerated Technologies. http://www.impulseaccelerated.com/, 06 June 2011.

[7] InterSense, Inc. http://www.intersense.com/, 06 June 2011.

[8] Meta Motion. http://www.metamotion.com/, 06 June 2011.

[9] Mitrionics. http://www.mitrionics.com/, 06 June 2011.

[10] NaturalPoint, Inc. http://www.naturalpoint.com/, 06 June 2011.

[11] Polhemus, Inc. http://www.polhemus.com/, 06 June 2011.

[12] Synphony C Compiler. http://www.synopsys.com/Systems/BlockDesign/HLS/Pages/SynphonyC-Compiler.aspx, 06 June 2011.

[13] SystemC. http://www.systemc.org/, 06 June 2011.

[14] Ubisense Limited. http://www.ubisense.net/, 06 June 2011.

[15] Xsens. http://www.xsens.com/, 06 June 2011.

[16] W.T. Adans and J. Brady. Magnitude approximations for microprocessor implementation. *Micro, IEEE*, 3(5):27–31, oct. 1983.

[17] M.D. Addlesee, A. Jones, F. Livesey, and F. Samaria. The ORL active floor [sensor system]. *Personal Communications, IEEE*, 4:35–41, 1997.

[18] Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward, and Andy Hopper. Implementing a sentient computing system. *IEEE Computer*, 34(8):50–56, August 2001.

[19] Christopher Alexander. *Notes on the Synthesis of Form*. Harvard University Press, October 24 1964.

[20] Mark Allie and Richard Lyons. High-Speed Square Root Algorithms. In Richard Lyons, editor, *Streamlining Digital Signal Processing*, chapter 16, pages 165–172. John Wiley & Sons, Inc., 2007.

[21] F. J. Álvarez, J. Ureña, M. Mazo, A. Hernández, J. J. García, and C. de Marziani. High reliability outdoor sonar prototype based on efficient signal coding. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 53(10):1862 –1872, October 2006.

[22] Ray Andraka. A survey of CORDIC algorithms for FPGA based computers. In *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, FPGA '98, pages 191–200, New York, NY, USA, 1998. ACM.

[23] Arvind, R.S. Nikhil, D.L. Rosenband, and N. Dave. High-level synthesis: an essential ingredient for designing complex ASICs. In *Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on*, pages 775–782, nov. 2004.

[24] T. Austin, D. Blaauw, S. Mahlke, T. Mudge, C. Chakrabarti, and W. Wolf. Mobile supercomputers. *Computer*, 37(5):81–83, May 2004.

[25] AutoESL Design Technologies, INC. *AutoPilot User Guide*, v2010.a edition, June 2010.

[26] P. Bahl and V.N. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775 –784, 2000.

[27] F. Barceló, F. Evennou, L. de Nardis, and P. Tomé. Advances in indoor Location. In *LIAISON - ISHTAR Workshop*, 2006.

[28] G. Bishop B.D. Allen and G. Welch. Tracking: Beyond 15 minutes of thought: Siggraph 2001 course 11. In *Course Notes, Ann. Conf. Computer Graphics and Interactive Techniques (Siggraph 2001)*, New York, 2001. ACM Press.

[29] Boualem Boashash. Estimating and Interpreting the Instantaneous Frequency of a Signal-Part I: Fundamentals. *Proceedings of the IEEE*, 80(4):519–538, April 1992.

[30] Boualem Boashash. Estimating and Interpreting the Instantaneous Frequency of a Signal-Part II: Algorithms and Applications. *Proceedings of the IEEE*, 80(4):539–569, April 1992.

[31] C. Bouganis and G.A. Constantinides. Synthesis of DSP Algorithms from Infinite Precision Specifications. In P. Coussy and A. Moriavec (Eds.), editors, *High-Level Synthesis From Algorithm to Digital Circuit*, chapter 11. Springer-Verlag, 2008.

[32] Grant Martin Brian Bailey and Andrew Piziali. *ESL Design and Verification: A Prescription for Electronic System Level Methodology*. Elsevier Morgan Kaufmann, 2007.

[33] Lidija Busic and Renato Filjar. The role of position reporting frequency in LBS QoS establishment. *Software in Telecommunications and Computer Networks, 2006. SoftCOM 2006. International Conference on*, pages 209–213, Sept. 29 2006-Oct. 1 2006.

[34] Betul Buyukkurt, John Cortes, Jason Villarreal, and Walid A. Najjar. Impact of high-level transformations within the ROCCC framework. *ACM Trans. Archit. Code Optim.*, 7:17:1–17:36, December 2010.

[35] D. Cassioli, R. Giuliano, and F. Mazzenga. Analysis of UWB system capacity in a realistic multipath environment with coexistence constraints. *Communications, IET*, 1(3):391–397, June 2007.

[36] M. Chakraborty and A. Mitra. A block floating-point realization of the gradient adaptive lattice filter. *Signal Processing Letters, IEEE*, 12(4):265–268, april 2005.

[37] M. Chakraborty, R. Shaik, and Moon Ho Lee. A block-floating-point-based realization of the block LMS algorithm. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 53(9):812–816, sept. 2006.

[38] Yu-Chung Cheng, Yatin Chawathe, Anthony LaMarca, and John Krumm. Accuracy characterization for metropolitan-scale Wi-Fi localization. In *Proceedings of the Third International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 233–245, Seattle, USA, June 2005.

[39] J. Cong, G. Reinman, A. Bui, and V. Sarkar. Customizable domain-specific computing. *Design Test of Computers, IEEE*, 28(2):6–15, march-april 2011.

[40] Jason Cong and Yi Zou. FPGA-Based Hardware Acceleration of Lithographic Aerial Image Simulation. *ACM Trans. Reconfigurable Technol. Syst.*, 2(17):17:1–17:29, Sept. 2009.

[41] N. Dave, M. Pellauer, S. Gerding, and Arvind. 802.11a transmitter: a case study in microarchitectural exploration. In *Formal Methods and Models for Co-Design, 2006. MEMOCODE '06. Proceedings. Fourth ACM and IEEE International Conference on*, pages 59–68, july 2006.

[42] M. Dell'Anna and A.H. Aghvami. Performance of optimum and suboptimum combining at the antenna array of a W-CDMA system. *Selected Areas in Communications, IEEE Journal on*, 17(12):2123–2137, dec 1999.

[43] J. V. DiFranco and W. L. Rubin. *Radar Detection*. Prentice-Hall, Englewood Cliffs, N.J., 1968.

[44] E.H. Dinan and B. Jabbari. Spreading codes for direct sequence cdma and wideband cdma cellular networks. *Communications Magazine, IEEE*, 36(9):48–54, Sep 1998.

[45] John P. Elliot. *Understanding Behavioral Synthesis*. Kluwer Academic Publishers, 1999.

[46] Shlomo Engelberg. *Digital Signal Processing: An Experimental Approach*. Springer-Verlag, London, 2008.

[47] A. Filip. Linear approximations to sqrt(x^2+y^2) having equiripple error characteristics. *Audio and Electroacoustics, IEEE Transactions on*, 21(6):554–556, dec 1973.

[48] Michael Fingeroff. *High-Level Synthesis Blue Book*. Xlibris Corporation, 21 May 2010.

[49] George W. Fitzmaurice. Situated information spaces and spatially aware palmtop computers. *Communications of the ACM*, 36(3):39–49, July 1993.

[50] R.J. Fontana. Recent system applications of short-pulse ultra-wideband (UWB) technology. *Microwave Theory and Techniques, IEEE Transactions on*, 52(9):2087–2104, Sept. 2004.

[51] R.J. Fontana, E. Richley, and J. Barney. Commercialization of an ultra wideband precision asset location system. *Ultra Wideband Systems and Technologies, 2003 IEEE Conference on*, pages 369–373, 16-19 Nov. 2003.

[52] E. Foxlin. Motion tracking technologies and requirements. In Kay Stanney, editor, *Handbook of Virtual Environment Technologies*, chapter 8, pages 163–210. Lawrence Erlbaum Publishers, Hillsdale, N.J., 2002.

[53] E. Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *Computer Graphics and Applications, IEEE*, 25(6):38–46, nov.-dec. 2005.

[54] Eric Foxlin, Michael Harrington, and George Pfeifer. Constellation: A wide-range wireless motion-tracking system for augmented reality and virtual set application. In Michael Cohen, editor, *Proceedings of SIGGRAPH 98*, pages 371–378. Addison Wesley, 1998.

[55] L. Freitag, M. Johnson, and D. Frye. High-rate acoustic communications for ocean observatories-performance testing over a 3000 m vertical path. In *OCEANS 2000 MTS/IEEE Conference and Exhibition*, volume 2, pages 1443 –1448, September 2000.

[56] Sinan Gezici, Zhi Tian, Georgios Giannakis, Hisashi Kobayashi, Andreas Molisch, Vincent Poor, and Zafer Sahinoglu. Localization via Ultra-Wideband Radios: a look at positioning aspects for future sensor networks. *IEEE Signal Processing Magazine*, 22(4):70–84, July 2005.

[57] R. Gitlin, H. Meadors, and S. Weinstein. An algorithm for the stable operation of a digitally-implemented fractionally-spaced adaptive equalizer. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '82.*, volume 7, pages 1379–1383, May 1982.

[58] L.C. Godara. Application of antenna arrays to mobile communications. II. beam-forming and direction-of-arrival considerations. *Proceedings of the IEEE*, 85(8):1195–1245, Aug 1997.

[59] F. J. Harris. Exact FM detection of complex time series. In *ISSPA '87*, pages 70–73, Brisbane, Australia, 1987.

[60] Mike Hazas. *Indoor Location Systems*. PhD in Location Sensing, Laboratory for Communication Engineering – University of Cambridge, Digital Technology Group, William Gates Building, 15 JJ Thomson Avenue, Cambridge, CB3 0FD, 2002.

[61] Mike Hazas and Andy Hopper. Broadband ultrasonic location systems for improved indoor positioning. *IEEE Transactions on Mobile Computing*, 5(5):536–547, May 2006.

[62] Mike Hazas, Christian Kray, Hans Gellersen, Henoc Agbota, Gerd Kortuem, and Albert Krohn. A relative positioning system for co-located mobile devices. In *Proc. of MobiSys*, pages 177–190, 2005.

[63] Mike Hazas and Andy Ward. A novel broadband ultrasonic location system. In *Proc. of UbiComp*, pages 264–280. Springer-Verlag, 2002.

[64] Christian Lders Henrik Schulze. *Theory and Applications of OFDM and CDMA: Wideband Wireless Communications*. John Wiley & Sons, Ltd, 3 JAN 2006.

[65] Walter Hirt and Domenico Porcino. Pervasive ultra-wideband low spectral energy radio systems (PULSERS). White Paper Version 1.6, WWRF/WG4/UWB-Subgroup, 18 November 2002. Pulsers Deliverable.

[66] Harri Holma and Antti Toskala, editors. *WCDMA for UMTS*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[67] Jack K. Holmes. *Spread Spectrum Systems for GNSS and Wireless Communications*. Artech House, Inc., Norwood, MA, USA, 2007.

[68] Andy Hopper. The clifford paterson lecture 1999: Sentient computing. *Philosophical Transactions of The Royal Society: Mathematical, Physical and Engineering Sciences*, 358:2349–2358, 2000.

[69] J. H.Wilkinson. *Rounding Errors in Algebraic Processes*. Prentice-Hall, Englewood Cliffs, NJ, 1963.

[70] Texas Instruments. System-on-Chip (SoC) Solution for ZigBee/IEEE 802.15.4 Wireless Sensor Network. `http://focus.ti.com/docs/prod/folders/print/cc2431.html`, 27 November 2007.

[71] A.A. Jerraya and W. Wolf. Hardware/software interface codesign for embedded systems. *Computer*, 38(2):63–69, Feb. 2005.

[72] M. Johnson, L. Freitag, and M. Stojanovic. Improved doppler tracking and correction for underwater acoustic communications. In *Acoustics, Speech, and Signal Processing, (ICASSP), IEEE Int'l Conf. on*, volume 1, pages 575–578, Apr. 1997.

[73] A. Katasonov and M. Sakkinen. Content quality in location-based services: a case study. *Pervasive Services, 2005. ICPS '05. Proceedings. International Conference on*, pages 461–464, 11-14 July 2005.

[74] Michael Keating and Pierre Bricaud. *Reuse methodology manual: for system-on-a-chip designs*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[75] K. Kolodziej and J. Danado. In-building positioning: modeling location for indoor world. *Database and Expert Systems Applications, 2004. Proceedings. 15th International Workshop on*, pages 830–834, 30 Aug.-3 Sept. 2004.

[76] H. Kreide and D.W. Lambert. `http://history.nasa.gov/computers/Ch4-2.html`. H. Kreide and D.W. Lambert, "Computation: Aerospace Computers in Aircraft, Missiles and Spacecraft," Space/Aeronaut., 42, 78 (1964); see also N.H. Herman and U.S. Lingon, "Mariner 4 Timing and Sequencing," Astronaut. Aeronaut., 43 (October 1965).

[77] Dhananjay Kulkarni, Walid A. Najjar, Robert Rinker, and Fadi J. Kurdahi. Compile-time area estimation for LUT-based FPGAs. *ACM Trans. Des. Autom. Electron. Syst.*, 11(1):104–122, 2006.

[78] Sun-Yuan Kung. On supercomputing with systolic/wavefront array processors. *Proceedings of the IEEE*, 72(7):867–884, july 1984.

[79] B. Lakshmi and A. S. Dhar. Cordic architectures: A survey. *VLSI Design*, page 19, 2010.

[80] E.A. Lee and D.G. Messerschmitt. Synchronous data flow. *Proceedings of the IEEE*, 75(9):1235–1245, sept. 1987.

[81] Jhong Sam Lee and Leonard E. Miller. *CDMA Systems Engineering Handbook*. Artech House, Inc., Norwood, MA, USA, 1998.

[82] Joon-Yong Lee and R.A. Scholtz. Ranging in a dense multipath environment using an UWB radio link. *Selected Areas in Communications, IEEE Journal on*, 20(9):1677–1683, Dec 2002.

[83] Weichang Li and J. C. Preisig. Estimation of rapidly time-varying sparse channels. *IEEE Journal of Oceanic Engineering*, 32(4):927–939, October 2007.

[84] H. Liu, H. Darabi, P. Banerjee, and J. Liu. Survey of wireless indoor positioning techniques and systems. *Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(6):1067–1080, Nov. 2007.

[85] Richard G. Lyons. *Understanding Digital Signal Processing*. Prentice Hall, 2nd ed. edition, 2004.

[86] Grant Martin. Multi-processor soc-based design methodologies using configurable and extensible processors. *Journal of Signal Processing Systems*, 53:113–127, 2008. 10.1007/s11265-007-0153-7.

[87] Grant Martin. Practical System-Level Design Methodologies for Processor-Centric SoC and Embedded Systems. HiPEAC's ACACES 2009 Summer School Course Notes, July 12–18 2009.

[88] Michael McCarthy, Paul Duff, Henk L. Muller, and Cliff Randell. Accessible Ultrasonic Positioning. *IEEE Pervasive Computing*, 5(4):86–93, 2006.

[89] Chris McMahon, Matthew; Steketee. Investigation of proposed applications for lbs enabled mobile handsets. *Mobile Business, 2006. ICMB '06. International Conference on*, pages 26–26, June 2006.

[90] R.W.; Brink S.T.; Mahadevappa R. Mishra, S.M.; Brodersen. Detect and avoid: an ultra-wideband/WiMAX coexistence mechanism [topics in radio communications]. *Communications Magazine, IEEE*, 45(6):68–75, June 2007.

[91] A. Mitra and M. Chakraborty. The NLMS algorithm in block floating-point format. *Signal Processing Letters, IEEE*, 11(3):301–304, march 2004.

[92] A. Mitra, M. Chakraborty, and H. Sakai. A block floating-point treatment to the LMS algorithm: efficient realization and a roundoff error analysis. *IEEE Transactions on Signal Processing*, 53(12):4536 – 4544, dec. 2005.

[93] Rafiahamed Shaik Mrityunjoy Chakraborty and Moon Ho Lee. An efficient implementation of the sign LMS algorithm using block floating point format. *EURASIP Journal on Advances in Signal Processing*, vol. 2007, 2007.

[94] S. Nakamura, T. Sato, M. Sugimoto, and H. Hashizume. An accurate technique for simultaneous measurement of 3D position and velocity of a moving object using a single ultrasonic receiver unit. In *Indoor Positioning and Indoor Navigation (IPIN), Int'l Conf. on*, pages 1 –7, Sept. 2010.

[95] Wolfgang Narzt, Gustav Pomberger, Alois Ferscha, Dieter Kolb, Reiner Müller, Jan Wieghardt, Horst Hörtner, and Christopher Lindinger. Pervasive information acquisition for mobile AR-navigation systems. In *Proceedings of the Fifth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 13–20, Monterey, USA, October 2003.

[96] V.H. Nascimento and A.H. Sayed. Unbiased and stable leakage-based adaptive filters. *Signal Processing, IEEE Transactions on*, 47(12):3261–3276, Dec 1999.

[97] J.A. Neasham, D. Thompson, A.D. Tweedy, M.A. Lawlor, O.R. Hinton, A.E. Adams, and B.S. Sharif. Combined equalisation and beamforming to achieve 20 kbits/s acoustic telemetry for rovs. In *OCEANS '96. MTS/IEEE. 'Prospects for the 21st Century'. Conference Proceedings*, volume 2, pages 988–993, Sep 1996.

[98] Mark W. Newman, Jana Z. Sedivy, Christine M. Neuwirth, W. Keith Edwards, Jason I. Hong, Shahram Izadi, Karen Marcelo, and Trevor F. Smith. Designing for serendipity: Supporting end-user configuration of ubiquitous computing environments. In *Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, pages 147–156, London, England, June 2002.

[99] Jari Nurmi, editor. *Processor Design: System-on-Chip Computing for ASICs and FPGAs*. Kluwer Academic Publishers / Springer Publishers, P.O. Box 17, 3300 AA Dordrecht, The Netherlands., 2007.

[100] OpenCores. WISHBONE System-on-Chip (SoC) Interconnection Architecture-for Portable IP Cores. `http://opencores.org/opencores,wishbone`, 2010. Accessed 6 June 2011.

[101] M Oppenheim. PVDF phased-array analog front end. *Circuit Cellar*, issue 244:20–25, November 2010.

[102] I. Oppermann, L. Stoica, A. Rabbachin, Z. Shelby, and J. Haapola. UWB wireless sensor networks: UWEN - a practical example. *Communications Magazine, IEEE*, 42(12):S27–S32, Dec. 2004.

[103] Alexandros Papakonstantinou, Yun Liang, John A. Stratton, Karthik Gururaj, Deming Chen, Wen-Mei W. Hwu, and Jason Cong. Multilevel granularity parallelism synthesis on FPGAs. In *Field-Programmable Custom Computing Machines (FCCM), 2011 IEEE 19th Annual International Symposium on*, pages 178–185, may 2011.

[104] K. Perusco, L.; Michael. Control, trust, privacy, and security: evaluating location-based services. *Technology and Society Magazine, IEEE*, 26(1):4–16, Spring 2007.

[105] L. Perusco and K. Michael. Humancentric applications of precise location based services. *e-Business Engineering, 2005. ICEBE 2005. IEEE International Conference on*, pages 409–418, 18-21 Oct. 2005.

[106] P. Poplavko, T. Basten, M. Bekooij, J. van Meerbergen, and B. Mesman. Task-level timing models for guaranteed performance in multiprocessor networks-on-chip. In

Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems, CASES '03, pages 63–72, New York, NY, USA, 2003. ACM.

[107] J.C. Prieto, A.R. Jiménez, J. Guevara, J.L. Ealo, F. Seco, J.O. Roa, and F. Ramos. Performance evaluation of 3D-LOCUS advanced acoustic LPS. *Instrumentation and Measurement, IEEE Transactions on*, 58(8):2385 –2395, 2009.

[108] N.B. Priyantha, H. Balakrishnan, E.D. Demaine, and S. Teller. Mobile-assisted localization in wireless sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, pages 172–183, march 2005.

[109] J.G. Proakis. Adaptive equalization techniques for acoustic telemetry channels. *Oceanic Engineering, IEEE Journal of*, 16(1):21–31, Jan 1991.

[110] John Proakis and Massoud Salehi. *Digital Communications*. McGraw-Hill, 5th edition, November 6 2007.

[111] R.C. Qiu, H. Liu, and X. Shen. Ultra-wideband for multiple access communications. *IEEE Communications Magazine*, 43(2):80–87, Feb 2005.

[112] Bharat Rao and Louis Minakakis. Evolution of mobile location-based services. *Communications of the ACM*, 46(12):61–65, 2003.

[113] reacTIVision 1.4. `http://reactivision.sourceforge.net/`.

[114] Jun Rekimoto and Katashi Nagao. The world through the computer: Computer augmented interaction with real world environments. In *User Interface Software and Technology (UIST)*, pages 29–36, Pittsburgh, USA, November 1995. ACM.

[115] Ali H. Sayed. *Adaptive Filters*. Wiley-IEEE Press, hardcover edition, May 2008.

[116] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85 –90, dec. 1994.

[117] V. Schwarz, A. Huber, and M. Tuchler. Accuracy of a commercial UWB 3D location/tracking system and its impact on LT application scenarios. *Ultra-Wideband, 2005. ICU 2005. 2005 IEEE International Conference on*, pages 599–603, 5-8 Sept. 2005.

[118] James Scott and Mike Hazas. User-Friendly Surveying Techniques for Location-Aware Systems. In *Proceedings of UbiComp 2003: Fifth International Conference on Ubiquitous Computing*, volume 2864, pages 45–54, Seattle, USA, October 2003.

[119] R. Shaik and Mrityunjoy Chakraborty. An efficient realization of the decision feedback equalizer using block floating point arithmetic. In *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, pages 1047 –1050, dec. 2006.

[120] B.S. Sharif, J. Neasham, O.R. Hinton, and A.E. Adams. Closed loop doppler tracking and compensation for non-stationary underwater platforms. In *OCEANS 2000 MTS/IEEE Conference and Exhibition*, volume 1, pages 371–375, Sept 2000.

[121] B.S. Sharif, J. Neasham, O.R. Hinton, and A.E. Adams. A computationally efficient doppler compensation system for underwater acoustic communications. *IEEE Journal of Oceanic Engineering*, 25(1):52–61, jan 2000.

[122] B.S. Sharif, J. Neasham, O.R. Hinton, A.E. Adams, and J. Davies. Adaptive doppler compensation for coherent acoustic communication. *Radar, Sonar and Navigation, IEE Proceedings* -, 147(5):239 –246, Oct. 2000.

[123] Qicai Shi, S. Kyperountas, N.S. Correal, and Feng Niu. Performance analysis of relative location estimation for multihop wireless sensor networks. *Selected Areas in Communications, IEEE Journal on*, 23(4):830–838, April 2005.

[124] D. Skournetou and E. Lohan. Pulse shaping investigation for the applicability of future GNSS signals in indoor environments. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, pages 1–7, sept. 2010.

[125] Adam Smith, Hari Balakrishnan, Michel Goraczko, and Nissanka Priyantha. Tracking moving devices with the cricket location system. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, MobiSys '04, pages 190–202, New York, NY, USA, 2004. ACM.

[126] Julius O. Smith. *Physical Audio Signal Processing for Virtual Musical Instruments and Digital Audio Effects*. W3K, November 6 2010.

[127] B. So, P.C. Diniz, and M.W. Hall. Using estimates from behavioral synthesis tools in compiler-directed design space exploration. In *Design Automation Conference, 2003. Proceedings*, pages 514–519, june 2003.

[128] M. Stojanovic. Recent advances in high-speed underwater acoustic communications. *Oceanic Engineering, IEEE Journal of*, 21(2):125 –136, Apr 1996.

[129] M. Stojanovic. Efficient processing of acoustic signals for high-rate information transmission over sparse underwater channels. *Physical Communication*, 1(2):146 – 161, 2008.

[130] M. Stojanovic, J.A. Catipovic, and J.G. Proakis. Phase-coherent digital communications for underwater acoustic channels. *IEEE Journal of Oceanic Engineering*, 19(1):100–111, Jan 1994.

[131] M. Stojanovic and L. Freitag. Acquisition of direct sequence spread spectrum acoustic communication signals. In *OCEANS 2003. Proceedings*, volume 1, pages 279–286, 2003.

[132] M. Stojanovic and L. Freitag. Multiuser code acquisition in multipath channels. In *Oceans 2005 - Europe*, volume 1, pages 74–79, June 2005.

[133] Milica Stojanovic and Lee Freitag. Multichannel Detection for Wideband Underwater Acoustic CDMA Communications. *IEEE Journal of Oceanic Engineering*, 31(3):685–695, July 2006.

[134] Norbert Streitz and Paddy Nixon. Introduction. *Communications of the ACM - The disappearing computer*, 48:32–35, March 2005.

[135] Ivan E. Sutherland. A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, AFIPS '68 (Fall, part I), pages 757–764, New York, NY, USA, 1968. ACM.

[136] S. Tanaka, A. Harada, M. Sawahashi, and F. Adachi. Experiments on coherent adaptive antenna array diversity for wideband DS-CDMA mobile radio. *Selected Areas in Communications, IEEE Journal on*, 18(8):1495–1504, aug 2000.

[137] M. Toda and S. Tosima. Theory of curved, clamped, piezoelectric film, air-borne transducers. *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, 47(6):1421–1431, nov 2000.

[138] J. Ureña, A. Hernández, A. Jiménez, J. M. Villadangos, M. Mazo, J. C. García, J. J. García, F. J. Álvarez, C. de Marziani, M. C. Pérez, J. A. Jiménez, A. R. Jiménez, and F. Seco. Advanced sensorial system for an acoustic LPS. *Microprocessors and Microsystems*, 31(6):393 – 401, 2007.

[139] B.D. Van Veen and K.M. Buckley. Beamforming: a versatile approach to spatial filtering. *ASSP Magazine, IEEE*, 5(2):4 – 24, april 1988.

[140] Stanley P. Vanderkooy, John; Lipshitz. Dither in digital audio. *J. Audio Eng. Soc*, 35(12):966–975, 1987.

[141] Nanyan Wang, P. Agathoklis, and A. Antoniou. Analysis of beamformer configurations for DS-CDMA systems. *Signal Processing, IEEE Transactions on*, 53(3):945–956, march 2005.

[142] Philip W. Ward, John W. Betz, and Christopher J. Hegarty. Satellite Signal Acquisitions, Tracking, and Data Demodulation. In Elliott Kaplan and Christopher Hegarty, editors, *Understanding GPS: Principles and Applications*, chapter 5. Artech House, Inc., 2006.

[143] Mark Weiser. Some computer science issues in ubiquitous computing. *Communication ACM*, 36(7):75–84, 1993.

[144] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11, 1999.

[145] G. Welch and E. Foxlin. Motion tracking: No silver bullet, but a respectable arsenal. *Computer Graphics and Applications, IEEE*, 22(6):24 –38, nov.-dec. 2002.

[146] Greg Welch, Gary Bishop, Leandra Vicci, Stephen Brumback, Kurtis Keller, and D'nardo Colucci. High-performance wide-area optical tracking: The hiball tracking system. *Presence: Teleoper. Virtual Environ.*, 10:1–21, February 2001.

[147] W. Wolf. A decade of hardware/software codesign. *Computer*, 36(4):38–43, April 2003.

[148] W. Wolf, A.A. Jerraya, and G. Martin. Multiprocessor System-on-Chip (MPSoC) Technology. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27:1701–1713, 2008.

[149] Nancy Wu and Gary Smith. ESL synthesis: tips for implementing a viable ESL-synthesis flow. *EDN*, July 15 2010.

[150] Xilinx, Inc. *Virtex-5 FPGA XtremeDSP Design Considerations User Guide*, ug193 (v3.4) edition, June 2010.

[151] Zhiru Zhang, Yiping Fan, Wei Jiang, Guoling Han, Changqi Yang, and Jason Cong. AutoPilot: A Platform-Based ESL Synthesis System. In P. Coussy and A. Moriavec (Eds.), editors, *High-Level Synthesis From Algorithm to Digital Circuit*, chapter 6. Springer-Verlag, 2008.

[152] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proc. of SenSys*, pages 1–13, Los Angeles, November 2003.

[153] Heidi Ziegler and Mary Hall. Evaluating heuristics in automatically mapping multi-loop applications to FPGAs. In *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*, FPGA '05, pages 184–195, New York, NY, USA, 2005. ACM.

[154] Kamil Sh. Zigangirov. *Theory of Code Division Multiple Access Communication*. IEEE Press, 445 Hoes Lane, Piscataway, NJ 08854., 2004.