Increasing Accuracy of Location Determination

Exploiting Phase Change Reconstruction and Timing Measurements

LIN JI

KTH Information and Communication Technology

Master of Science Thesis
Stockholm, Sweden 2007

COS/CCS 2007-14

KUNGLIGA TEKNISKA HÖGSKOLAN
Royal Institute of Technology

Increasing Accuracy of Location Determination: Exploiting Phase Change Reconstruction and Timing Measurements

Master Thesis
Last revised: 2007-05-16

Lin Ji
Email: linusji@kth.se

Examiner
Professor Gerald Q. Maguire Jr.

Supervisor
Professor Mark T. Smith

Acknowledgment

This thesis is accomplished with kind help of a lot of people to whom I wish express my sincere thanks.

First, I wish express my deep gratitude to Professor Gerald Maguire, for reading through my thesis draft time after time, discussing with me, giving me so many valuable advices and being my examiner. I also would like to thank Professor Mark T. Smith for giving me the opportunity to do this work and his patience helping me develop the FPGA function, debugging my mistakes and suggestion about the thesis.

This work has given me a deeper view of a lot of things, such as 802.11b protocol, MATLAB programming, Software radio and FPGA design. These will be very helpful for my continuation of study or work. I've also learned how to perform a project work independently and document the results.

I will hereby say thanks to Yu Wu, Nan Zhang, Joel Anderson, the GNU radio group, my parents and all the friends who have always given me kind help and supports.

Abstract

The thesis deals with improving location determination when using time of flight of radio signals to determine the location of a radiator. The goal of this project is to enhance an existing wireless data access point to perform an accurate measurement of the time of arrival of a data signal from a transmitter, and to combine this information with information from additional wireless data access points to determine the location of the transmitter.

There have been a number of earlier efforts in indoor location determination system using different technologies. Many of which used signal strength analysis and they have low tolerance to moving obstacles such as humans, which frequently are the most usual dynamic obstacles in indoors. In this thesis, the proposed solution utilizes time stamping and sample correlation to utilize properties of the signal *waveform*, which has not previously been examined by researchers other than the examiner and advisor.

The main contribution of the project is a detailed analysis and design of a solution, as well as a comparison with other potential solutions. The main purpose of this solution is to increase the timing accuracy to below the duration of a single symbol.

The wireless device that has been analyzed implements the IEEE 802.11b protocol. Several investigations have been done to determine the best way of extracting information from the 802.11b data frame and symbol sequence; here we utilize a correlator to determine the time of arrival of a specific sequence of symbols in a data frame. The time stamping of a stream of samples has been implemented in an Altera FPGA to get a deterministic computation time.

Instead of decoding the incoming I&Q signals and mapping them to bits, the correlator is used to detect the unique sequence containing PSK encoded and Barker code spread scrambled ones , as this sequence always appears at the start of each data frame. The advantage of this approach is that using of samples of the waveform instead of bits gives a significant enhancement in timing resolution.

The design documents of this work include detailed descriptions, simulations, and plots. A number of simulations have been done to show the timing accuracy and standard deviation, as well as comparisons with several different approaches. Several potential optimizations have also been discussed in the report.

Simulation code for MATLAB and implementation code for the FPGA has been included in appendices in the end of this thesis.

Keywords: IEEE 802.11b, Correlation, Time stamping, FPGA, PSK

Inledning

Denna rapport beskriver ett examensarbete som utgår ifrån att förbättra noggrannhet av en lokaliseringsteknik som tillämpar flygtiden av radiosignal för att mäta positionen av utsändaren. Målet av detta arbete är att förstärka en befintlig trådlösdata accesspunkt för att utföra en noggrann mätning av signals anländning från sändaren, och kombinera denna information från ytterliga trådlösdata accesspunkter för att lokalisera sändaren.

Det har varit ett antal utvecklingar med olika teknologier för att ta fram en lösning för inomhus lokaliseringssystem. Många av de förslag som har kommit fram tillämpa signalstyrka och har en dålig tolerans av rörande hinder så som människor, vilka är det vanligaste fallet inomhus. Denna rapport ger ett förlag att lösa detta problem med att tillämpa tidstämpel och sampelkorrelation för att utnyttja egenskaper av signalens vågform, ett förlag som inte har blivit undersökt mycket av andra forskare.

Stor del av denna rapport består av detaljerad analys och lösningsdesign, plus en jämförelse med andra potentiella lösningar. Meningen med denna lösning är att öka noggrannhet till att felmarginal i tid ska ligga under en symboltid.

Den trådlösa enhet som har analyserats implementerar IEEE 802.11b protokollen. Flera undersökningar har utförts för att bestämma det bästa sättet att extrahera information från 802.11b dataramer och symbolsekvenser. För att göra det har vi implementerat en korrelator för att bestämma anländningstid av specifika symbolsekvenser i en dataram. En Altera FPGA har använts för att tidstämpla inkommande sampel för att ge en deterministisk beräkningstid.

Istället för att avkoda inkommande I&Q signaler och mappa dem till bitar, har vi valt att implementera korrelator så att den opererar direkt på PSK-modulerade och Barkerkod-spridda ettor, eftersom att det visar sig att denna sekvens alltid visar sig i början av varje dataram. Fördelen med denna lösning är att direktanvändning av sampel ger en signifikant ökning på tidsupplösning jämfört med bitar.

Designdokumentet av detta examensarbete består av detaljerade beskrivningar, simuleringar och grafer. Ett antal simuleringar har utförts för att visa tidsnoggrannhet och medelfel, plus jämförelser mellan olkia lösningsförslag. Olika möjliga optimeringar har också diskuterats i rapporten.

Simuleringskod för MATLAB och implementeringskod FPGA bifogas i appendix slutet av denna rapport.

Nyckelord: IEEE 802.11b, korrelation, Tidstämpel, FPGA, PSK

Table of Contents

CHAPTER 1: INTRODUCTION 1

1.1. BACKGROUND 1
1.2. EXISTING POSITIONING SYSTEMS 1
1.3. OVERVIEW 4
1.4. GOALS 5
1.5. ORGANIZATION 6

CHAPTER 2: INTRODUCTION TO IEEE 802.11B 7

2.1. IEEE 802.11B DATA FRAME STRUCTURE 7
2.2. SCRAMBLING 8
2.3. DSSS SPREADING FUNCTION 9
2.3.1. Barker code 10
2.3.2. CCK 11
2.4. DSSS MODULATION 12
2.4.1. Phase Shift Keying 12
2.4.2. Differential encoding 13

CHAPTER 3: DESIGN 14

3.1. GENERAL SYSTEM DESIGN 14
3.2. ADVANTAGES 15
3.3. DETAILED DESCRIPTION 15
3.3.1. Positioning 15
3.3.2. Time stamping 16
3.3.3. Timing measurement 16
3.3.4. System model 18
3.3.5. Test bench 19

CHAPTER 4: SIMULATION 20

4.1. IEEE 802.11B DATA FRAME ANALYSIS 20
4.2. MAC ADDRESS DECODING 20
4.3. INVESTIGATION OF THE FRAME USING SIMULATION 20
4.4. SAMPLE CORRELATION 21
4.5. SAMPLING TIME ERROR ANALYSIS 22
4.6. POSITION DETERMINATION BY ARRIVAL TIME 24

CHAPTER 5: IMPLEMENTATION 25

5.1. SINGLE USRP IMPLEMENTATION 25
5.2. PERFORMING THE CORRELATION USING THE FPGA 25
5.3. VERILOG IMPLEMENTATION 26
5.4. MULTIPLE USRP IMPLEMENTATION 27

CHAPTER 6: RESULTS 28

6.1. LOCATION DETERMINATION 28
6.2. FPGA BASED CORRELATION 29
6.3. WORK SUMMARY 31
6.3.1. Header analysis function 31
6.3.2. Mobile positioning function 32
6.3.3. USRP and recompiling the verilog code 33
6.3.4. FPGA functions 33

CHAPTER 7: CONCLUSIONS 34

7.1. CONCLUSION 34
7.2. FUTURE WORKS 34

REFERENCES 36

APPENDIX A: MATLAB CODE FOR HEADER ANALYSIS AND LOCATION DETERMINATION ... 38

A.1. HEADERANALYSIS.M 38
A.2. REALCASE.M 40
A.3. SORTARG.M 42
A.4. REALSIM.M 43

APPENDIX B: VERILOG CODE FOR TIME STAMPING AND CORRELATION 45

B.1. CORRELATION MODULE 45
B.2. TESTBENCH TO SIMULATE 46

APPENDIX C: LIST OF ACRONYMS 49

Chapter 1: Introduction

1.1. Background

Efficient and accurate location determination has long been an interesting topic. There have been several efforts to produce accurate positioning systems at different levels, local vs. global, indoors vs. outdoors. The most famous is the global positioning system (GPS). It is a satellite based navigation system. Although this system is very powerful, it is also expensive—the cost of maintaining the system is approximately US$400 million per year, including the replacement of aging satellites [16]. However, the service is considered so important for commercial tasks, such as aircraft, ship, and vehicle navigation that there are three such systems in existence: GPS, GLOSNASS, and GALILEO.

There have been a number of efforts to provide efficient and accurate indoor location determination systems and some of them are able to reach an accuracy of 95% [19]. Many of these proposals for using radio frequency measurements for indoor location determination are based on measurements of signal strength. One disadvantage of such systems is that they cannot tolerate dynamically moving obstacles such as animals and humans. As the signal will be attenuated, refracted, and reflected due to obstacles in the area, thus reducing the system's accuracy.

1.2. Existing positioning systems

Mobile positioning systems are often classified by the task they are being applied to, such as real time positioning system, fleet tracking, and traveler information services; or environment, such as indoor or outdoor system. In general, the technologies are often divided into network-based or satellite-based systems. Another classification is based on the actual device that performs the positioning solution, i.e., mobile device or the base station, leading to mobile terminal (user)-centric (such as GPS, A-GPS, E-OTD), network-centric (COO, TOA, TDOA, AOA, RSS, multipath pattern matching), or hybrid solutions. In the network-centric systems, the user's position is determined by the base station or a control center and sent to the application, while in the terminal-centric solution, the position computation is performed by the user's device.

Due to recent developments, the borders between these classes seem to be rather artificial. GPS, formerly only feasible outdoor, is now also becoming available indoor. The increasing density of indoor and outdoor WLAN networks may offer further opportunities with respect to the positioning of mobile users. In the future, a mobile user may no longer be interested in the positioning technology, but only the results; seamless switching between the different approaches should be done more or less automatically.

As the deployment of WLAN increase, WLANs are the principle means for delivering web services in limited mobility settings such as classrooms, campus areas of universities and enterprises, malls, and other indoor areas. For these types of applications WLAN positioning may be an interesting approach. Up to now, all WLAN-based position technologies use signal strength and sometimes use a propagation model. The mobile client measures the signal strengths of all surrounding access points and delivers this data to a positioning engine which in turn calculates the position by solving a maximum likelihood problem. The system is not affected by the fact that several access points transmit at the same frequency because it uses the integrated signals.

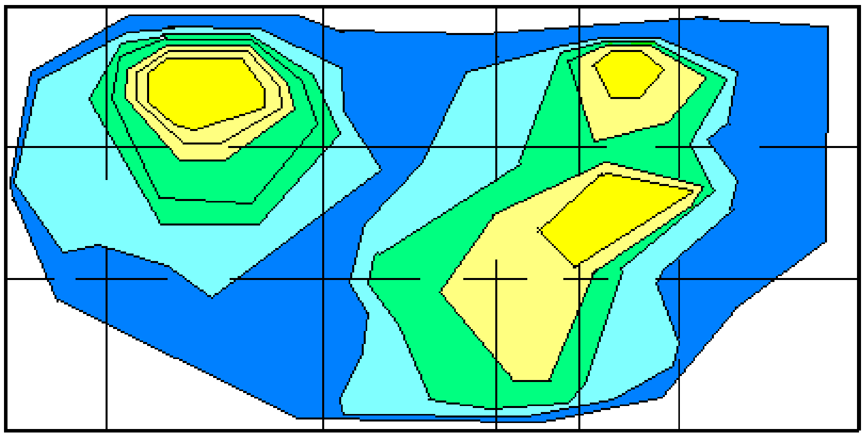

Figure 1: Indoor signal propagation (adapted from figure 2 of [13])

Other approaches utilizes modified access points in order to determine the distance to the mobile client via measuring the signal propagation time. Devices for this approach are manufactured by companies such as WhereNet (www.wherenet.com). There are is little detailed information on how these devices work, but they are expensive.

There are also Bluetooth-based positioning techniques. The idea is similar to the methods described for WLAN. Due to the shorter range a better accuracy might be expected, but the standard does not provide measurement of signal strengths. In addition, the usage of received signal strength indicator (RSSI) is hampered by poor implementations of the hardware and firmware which may vary by manufacturer. [13] Hence the actual accuracy is not better than that for WLAN based systems, in fact is likely to be worse.

See table 1 for a comparison of how today's positioning systems work. As readers may note, positioning technologies that use WLAN have about 3 meters accuracy. This is in the situation when there are no moving objects in the area. The performance degrades if there are people moving in the area, due to the attenuation affects of human body, thereby changing the the radio propagation environment such that the propagation model no longer applies.

Table 1: Criteria for selecting appropriate positioning techniques

Localization technology	Range/operational availability	Accuracy	Services and content
GSM-based approaches			
COO (Cell-of-Origin)	In principle globally	250m – 35km	Traffic information, information services
E-OTD (Enhanced Observed Time Difference)	In principle globally	100m – 500m	Location based billing, mobile yellow pages, information services
TOA (Time of Arrival)	Only after enormous capital investment globally	100m – 500m	Location based billing, mobile yellow pages, information services
OTDOA (Observed Time Difference of Arrival)	Only after enormous capital investment and with modified end devices globally	30m	Fleet management, routing functions, mobile advertisement, information services
Fingerprint	Only after enormous capital investment, feasible in urban areas	< 150m	Fleet management, routing functions, mobile advertisement, information services
GPS-based approaches			
GNSS/SBAS/DGPS	Globally	1cm – 100m	Positioning and navigation services
A-GPS (Assisted GPS)	Only after enormous capital investment globally	5m	Navigation services, security services, localization services
Indoor GPS	Indoor/special chips	20m	Localization services
WLAN/Bluetooth			
WLAN outdoor	Up to 500m around sender; cell-of-origin; distance; fingerprint	10m – 150m	Cell-of-Origin and information transfer
WLAN indoor	Up to 30m around sender; cell-of-origin; fingerprint	ca. 3m	Cell-of-Origin and information transfer
Bluetooth	Around sender/range up to 100m indoor (Class 2)	< 30m	Cell-of-Origin and information transfer
Other techniques			
Infrared beacons/ Active badges/WIPS	Around sender/range up to 10m indoor	< 10m	Cell-of-Origin and information transfer
Ultrasonic	Around sender/no hindrances	accurate	Localization services
Visual tags	Within visible range, typically indoor	Room	Localization services
Semantic positioning	In principle globally	Depending on data set	Mobile web location and information service
Relative positioning (INS, odometer etc.)	Locally, short time	meter – cm	Support technology for other localization services

1.3. Overview

This project is part of a larger effort in using propagation time of radio signals to determine the location of a radiator. The goal of the overall effort is to develop a low cost mechanism for accurately determining the location of a wireless device both indoors and outdoors. The physical phenomenon used is the time of flight of a radio signal. An intuitive description of how such a system for device location works is as follows:

Consider an ideal system consisting of a transmitting device and two receiving devices. The two receiving devices have synchronized clocks which for this example are assumed to be perfectly synchronized, and the locations in space of the two receiving devices are exactly known. When the transmitting device sends a signal the two receiving devices will report the time at which the signal arrived. By relatively simple algorithmic manipulation it can be seen that the two ideal receiving stations can position the transmitting device anywhere on an infinite band, where the width of this band is based upon the error of the time measurement caused by the sampling, see figure 2. The addition of a third receiving station, perfectly synchronization to the other clocks and located at an exactly known point in space would allow the position of a transmitting device to be known to within the area common to all three measurements, see figure 3.

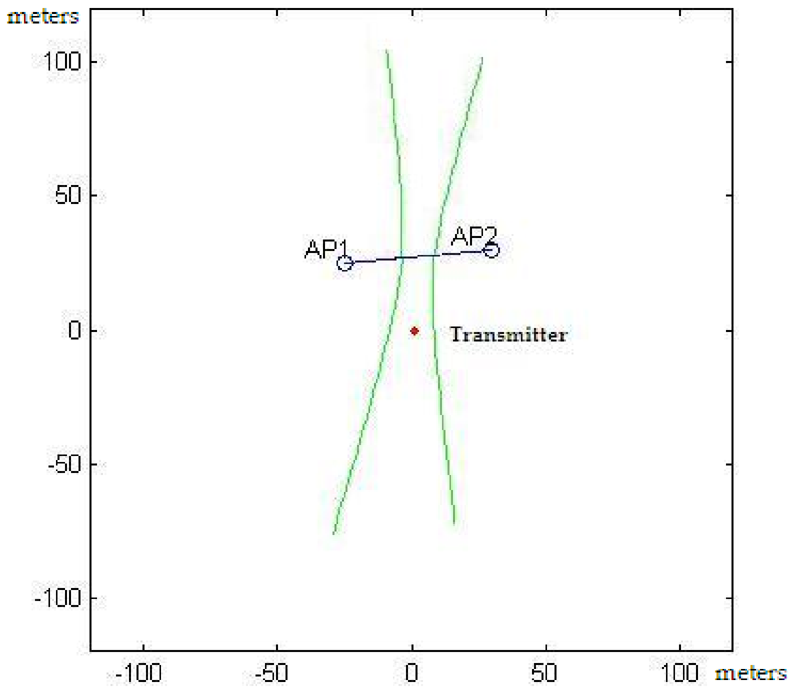

Figure 2: Two access points can define an infinite band of the mobile's possible position

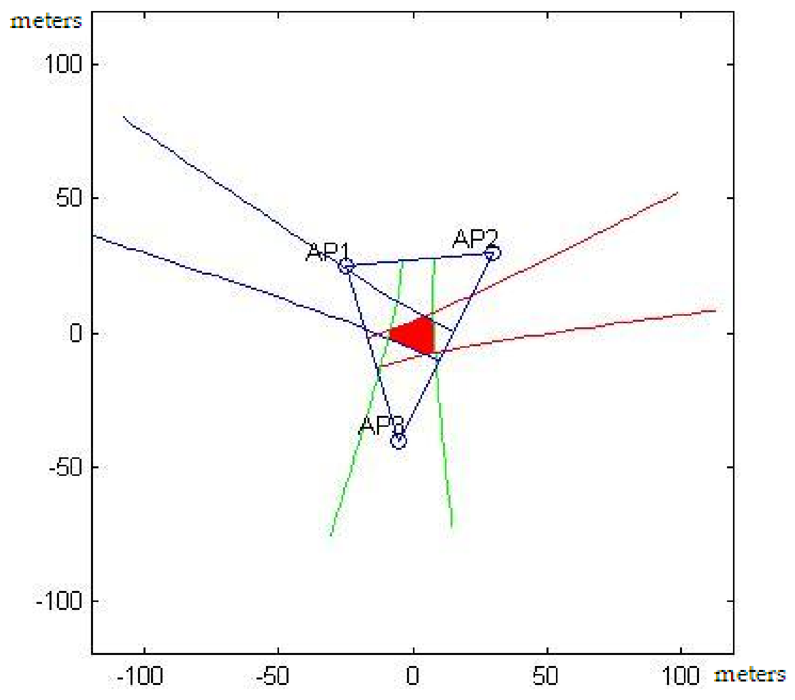

Figure 3: Three access points. The overlap of these three areas defines a finite area of the transmitter's possible position.

The advantage of such a method for determining position is that as it is not based on signal strength, the accuracy of the system is not subject to external influences that can attenuate the transmitted signal, such as position of obstacles, antenna orientation, or transmitter design. It only depends upon how accurately the receiving stations can measure both time, and the instant a signal arrives. Note that reflections, refraction, etc. will degrade the measurements, hence resulting in a larger region where the transmitter might be located, but not fundamentally change the result.

1.4. Goals

The specific deliverables for this master's project are:

1. Determine the sampling requirements for the I&Q baseband signals that can result in satisfactory signal reconstruction.

2. Determine how to reconstruct the phase change edges, or other characteristic of the received signal that can be recognized by all the access points.

3. Determine the temporal resolution requirements of the time stamping circuits.

4. Examine error sources and suggest potential ways of improving location precision, for example by analyzing the arrival times of several packets from the same transmitter to estimate an unbiased mean of the standard deviation for the timing measurement.

1.5. Organization

The remainder of this report is organized as follows:

Chapter 2 will introduce IEEE 802.11b standard, specifically the coding and modulation which have been used in the protocol, thus facilitating the reader's understanding of the remainder of this thesis.

Chapter 3 introduces all the design patterns including some initial potential solutions, and the reasons for choosing the selected approach. This chapter surveys several solutions and presents the differences between these solutions in detail, as well as their advantages and limitations.

Chapter 4 presents simulation results and its relation to the final implementation.

In Chapter 5, hardware implementations will be presented, as well as a detailed explanation of different optimizations.

The analysis of the implementation results will be presented in Chapter 6 as well as the interpretation of these results.

Finally, Chapter 7 summarizes the results, possible limitations, and future improvements.

Chapter 2: Introduction to IEEE 802.11b

This chapter introduces the relevant details of the IEEE 802.11b standard, and their relevance to the proposed location positioning system.

2.1. IEEE 802.11b Data frame structure

The IEEE 802.11b data frame structure is quite complex. It has several data fields that contain important information to ensure the data transfer. The part that is most relevant to this project is the Physical Layer Convergence Procedure PLCP protocol data unit (PPDU) header. There are 2 different types of header formats. One has a long PLCP preamble and the other is a short PLCP preamble. The long PLCP has a 128 bit preamble with scrambled ones while the short PLCP has a 64 bit preamble with scrambled zeros. The basic purpose of this scrambling will be explained later in this chapter.

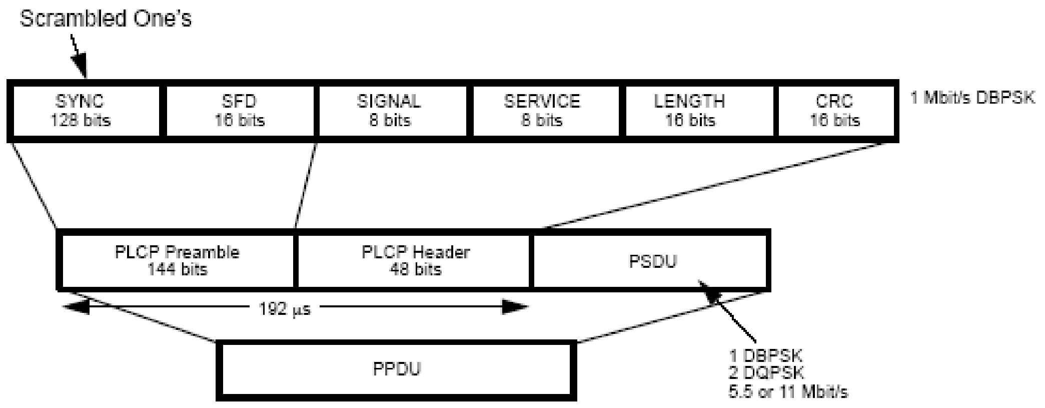

Figure 4: Long PLCP PPDU format

PLCP Frame Fields
The PLCP takes each frame that a station wishes to transmit and forms a PLCP protocol data unit (PPDU). The resulting PPDU includes the following fields in addition to the frame fields necessary for the MAC Layer. The details of the modulation and coding of these fields is described in sections 2.3 and 2.4.

Sync. This field consists of either 0s for short preamble or 1s for long preamble, alerting the receiver that a frame is about to start. The receiver synchronizes with the incoming signal. This field is important in this project to detect that a data frame will arrive.

Start Frame Delimiter. This field is always 1111001110100000 and defines the beginning of a frame. This field enables the link layer to frame the symbols, which allows the extraction of the MAC address. We will later use this MAC address to identify the particular station which we are interested in locating. This delimiter also defines when the sync pattern terminates.

Signal. This field identifies the data rate of the data portion of the frame, its binary value is equal to the data rate divided by 100Kbps. For example, the field contains the value of 00001010 for 1Mbps, 00010100 for 2Mbps, and so on. The PLCP fields, however, are always *sent at 1Mbps*. This ensures that the receiver uses the correct demodulation mechanism for the subsequent symbols, as this decoding method, changes in order to support different data rates.

Service. This field is always set to 0000 0000, and the standard reserves this field for future use.

Length. This field represents the number of microseconds that it takes to transmit the contents of the PPDU. The receiver uses this information to determine the end of the frame. While other stations can use this field to know when they should next listen to the channel (since if the MAC address indicates that the frame is not for them, then they can sleep/power down while the rest of this frame is being sent).

Frame Check Sequence. In order to detect possible errors in the physical layer header, the standard defines that this field containings a 16-bit cyclic redundancy check (CRC). The MAC Layer also performs error detection on the PPDU contents. This field is not relevant for this project because we assume that the data frames we receive have a correct header. Even if an error occurs, it will not significantly affect our system because we will utilize multiple packets, thus there is sufficient information – even if some frames are later rejected as not being relevant to our desired measurement (i.e. the MAC address might be in error and hence although we thought it matched the station we were trying to locate - if the CRC indicates an error them we should simply ignore this frame in our measurements).

PSDU. The Physical Layer Service Data Unit, is a fancy name for the payload contents of the PPDU (i.e., the actual link layer frame being sent). This field is important because it contains the sender and destination MAC addresses. As shown in figure 4, this field may be coded at different rates. This means that additional processing is needed to decode these MAC addresses. There are several proposals for how to process this field, and they will be presented in section 4.2.

2.2. Scrambling

Direct Sequence Spread Spectrum (DSSS) has been used in 802.11b to spread the energy of the signal over a broader band and hence both reduce interference with others and increase the robustness of the signal. (Details of DSSS will be covered in section 2.3). A direct sequence spread spectrum system uses a locally generated pseudo random code to encode digital data to be transmitted. The local code rate is much higher than the data rate. Data for transmission is simply logically modulo-2 added (an EXOR operation) with the faster pseudo random code. The composite pseudo random code and data can be passed through a data scrambler to randomize the output spectrum (and thereby remove discrete spectral lines) when a symbol is repeated. Here scrambling has not been used to prevent others from decoding the signal, but rather it is used to mitigate the effects of this signal on others (i.e., reduce the probability that this signal will interfere with others using the same frequency band) [17].

The scrambler and descrambler are shown in figures 5 and 6.

As stated in section 2.1, the preamble is always the same and for a long preamble it's always 1s. Therefore in this thesis there is no need to implement a scrambler or descrambler. Since the scrambled sequence can be directly matched, because since the pattern is fixed, it can be precomputed, hence a direct match is possible. This eliminates the need to implement a descrambler in the device; however, we must implement a scrambler in software so that we can computer what pattern we are to match.

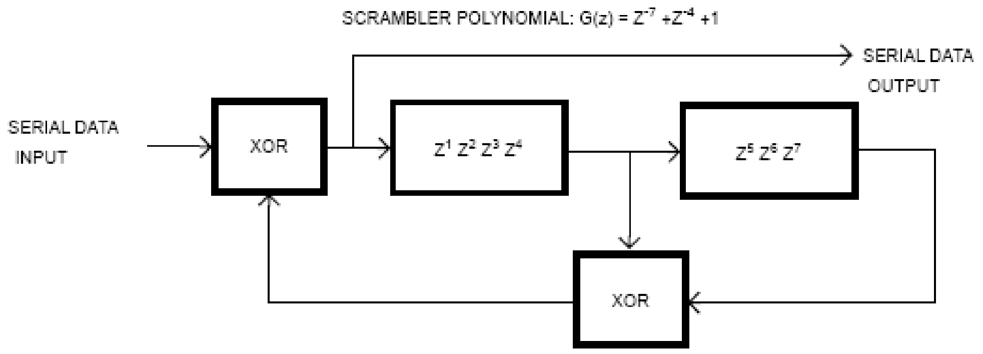

Figure 5: Data scrambler

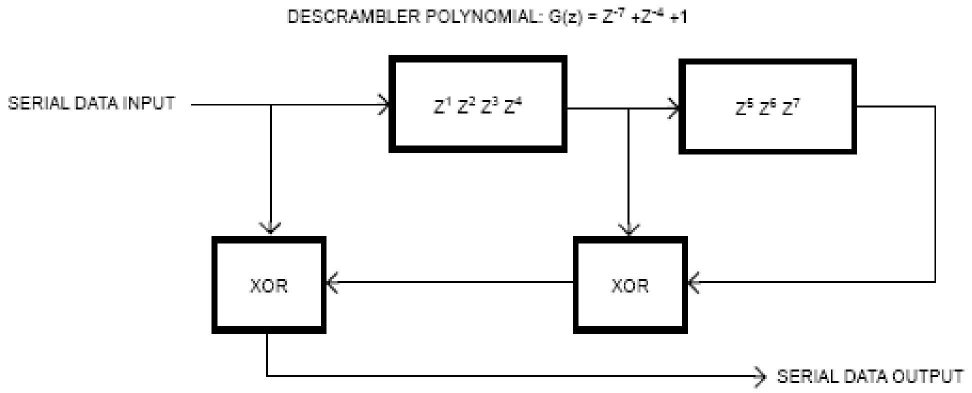

Figure 6: Data descrambler

2.3. DSSS Spreading function

Direct Sequence Spread Spectrum (DSSS) is used to disperse the signal over a relatively wide (approximately 30MHz) portion of the 2.4GHz frequency band. This results in greater immunity to narrow-band radio frequency interference as compared to narrowband signaling and reduces the impact on narrow band signals. It is this later properly that enabled the U.S. Federal Communications Commission to permit DSSS operation in the Industrial, Medical, and Instrumentation band (ISM band) as a tertiary user under part 15. This enabled license free operation of IEEE 802.11 DSSS equipment.

In order to actually spread the signal, an 802.11 transmitter combines the PPDU with a spreading sequence through the use of a binary adder. The spreading sequence is a binary code. For 1Mbps and 2Mbps operation, the spreading code is the 11-chip Barker sequence, which is 10110111000 (Note that the Barker code shown in Figure 8 is in the opposite order, but the correlation property is exactly the same). The binary adder effectively multiplies the length of the binary stream by the length of the sequence, which is 11. This increases the signaling rate and spreads the signal over a greater bandwidth.

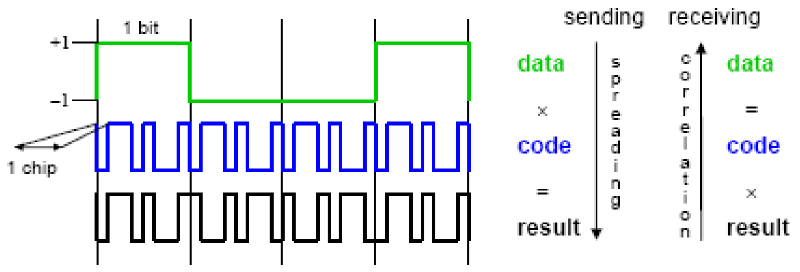

Figure 7: Data spreading

However, the 5.5Mbps and 11Mbps modes of 802.11b do not use the Barker sequence. Instead, 802.11b uses complementary code keying (CCK) to provide the spreading sequences at these higher data rates. CCK derives a different spreading code based on fairly complex functions depending on the pattern of bits being sent. The modulator simply refers to a table for the spreading sequence that corresponds to the pattern of data bits being sent. This is necessary for efficient processing of the data in order to achieve the higher data rates. Principles of Barker code and CCK are described below. Note that these higher code rates are only applicable to the data in the PSDU.

Because of the relatively wideband DSSS signal, 802.11b access points operating at high data rates must utilize specific channels to avoid spectral overlap, which can cause reductions in performance.

2.3.1. Barker code

A **Barker code** is a sequence of N values of +1 and −1,

a_j for $j = 1, \cdots, N$

such that

$$\left|\sum_{j=1}^{N-v} a_j a_{j+v}\right| \leq 1$$

for all $1 \leq v < N$.

Figure 8 shows all known Barker codes, where negations and reversals of the codes have been omitted.

Known Barker Codes

Length	Codes	
2	+1 −1	+1 +1
3	+1 +1 −1	
4	+1 −1 +1 +1	+1 −1 −1 −1
5	+1 +1 +1 −1 +1	
7	+1 +1 +1 −1 −1 +1 −1	
11	+1 +1 +1 −1 −1 −1 +1 −1 −1 +1 −1	
13	+1 +1 +1 +1 +1 −1 −1 +1 +1 −1 +1 −1 +1	

Figure 8: Barker codes

Barker codes of length 11 and 13 are used in direct-sequence spread spectrum and Pulse Compression Radar systems because of their low autocorrelation properties.

Barker codes utilize biphase modulation; that is, the change of phase in the carrier wave is 180 degrees [14].

2.3.2. CCK

Complementary Code Keying (CCK) is a modulation scheme used with wireless local area networks (WLANs) that employ the IEEE 802.11b specification and are operating at 5.5 Mbps or 11 Mbps. In 1999, CCK was adopted to to support these higher data rates.

Complementary codes, first introduced by Golay in 1961 are sets of finite sequences of equal length, such that the number of pairs of identical elements with any given separation in one sequence is equal to the number of pairs of unlike elements having the same separation in the other sequences.

The complementary codes first discussed by Golay were pairs of binary complementary codes and he noted that when the elements of a code of length N were either [-1 or 1] it followed immediately from their definition that the sum of their respective autocorrelation sequences was zero at all points except for the zero shift where it is equal to K*N. (K being the number of code words in the set).

CCK is a variation and improvement on, M-ary Orthogonal Keying and utilises 'polyphase complementary codes'. Polyphase complementary codes, first proposed by Sivaswamy--1978, are codes where each element is a complex number of unit magnitude and arbitrary phase, or more specifically for 802.11b is one of [1,-1, j,-j]. This use of these codes for WLAN was introduced by Lucent Technologies and Harris Semiconductor and was adopted by the 802.11 working group in 1998. CCK is the form of modulation utilized when 802.11b operates at either 5.5 or 11 Mbit/s. CCK was selected over competing modulation techniques as it utilized approximately the same bandwidth and could utilise the same preamble and header as existing 1 and 2 Mbit/s wireless networks thus facilitating interoperability.

The CCK modulation used by 802.11b transmits data in symbols of eight chips, where each chip is a complex QPSK bit-pair at a chip rate of 11Mchip/s. In 5.5 Mbit/s and 11 Mbit/s modes respectively 4 and 8 bits are modulated onto the eight chips of the symbol $c_0,\ldots,c_7$, where

$$\mathbf{c} = (c_0, \ldots, c_7) = \left(e^{j(\phi_1+\phi_2+\phi_3+\phi_4)}, e^{j(\phi_1+\phi_3+\phi_4)}, e^{j(\phi_1+\phi_2+\phi_4)}, -e^{j(\phi_1+\phi_4)},\right.$$
$$\left. -e^{j(\phi_1+\phi_4)}, e^{j(\phi_1+\phi_2+\phi_3)}, e^{j(\phi_1+\phi_3)}, -e^{j(\phi_1+\phi_2)}, e^{j\phi_1}\right)$$

and $\phi_1, \ldots, \phi_4$ are determined by the bits being modulated.

In other words, the phase change φ_1 is applied to every chip, φ_2 is applied to every other chip, φ_3 is applied to the first two of every four chips, and φ_4 is applied to the first four of the eight chips [15].

Wireless networks using the 802.11b specification employ CCK to operate at either 5.5 or 11 Mbit/s in the radio-frequency band ranging from 2.400 GHz to 2.4835 GHz. Networks using the IEEE 802.11g specification employ CCK when operating at 802.11b speeds. At higher speeds (up to a theoretical maximum of 54 Mbit/s), 802.11g WLANs use a more sophisticated modulation scheme called orthogonal frequency division multiplexing (OFDM). This is the same modulation method used by IEEE 802.11a WLANs in the radio-frequency band ranging from 5.725 GHz to 5.850 GHz.

2.4. DSSS Modulation

The modulator converts the spread binary signal into an analog waveform through the use of different modulation types, depending on which data rate is chosen. For example for 1Mbps operation, the standard specifies the user of differential binary phase shift keying (DBPSK). This isn't really as complex as it sounds. The modulator merely shifts the phase of the center transmit frequency to encode a binary 1 or a binary 0 within the data stream.

For 2Mbps transmission, differential quadrature phase shift keying (DQPSK) is used, this similar to DBPSK except that there are four possible phase shifts that represents pairs of data bits. This is a clever process that enables the data stream to be sent at 2Mbps while using the same amount of bandwidth as the one sent at 1Mbps. The modulator uses similar methods for the higher, 5.5Mbps and 11Mbps data rates. Principles of differential phase shift keying are described below.

Since the PLCP fields are set at 1Mbps always - it is primarily DBPSK which concerns us in this thesis. DQPSK system has been studied and simulated for future works.

2.4.1. Phase Shift Keying

Phase shift keying is a modulation method that maps the information bits into symbols in form of phase shifts.

The basic signal space function for Binary PSK is:

$$\phi(t) = \sqrt{\frac{2}{T_b}}\cos(2\pi f_c t)$$

Binary data is often conveyed with the following signals:

$$s_0(t) = \sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c t)$$ for binary "0"

$$s_1(t) = \sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c t + \pi) = -\sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c t)$$ for binary "1"

Where f_c is the frequency of the carrier--wave.

The symbols representing zeros and ones have a 180 degrees phase shift:

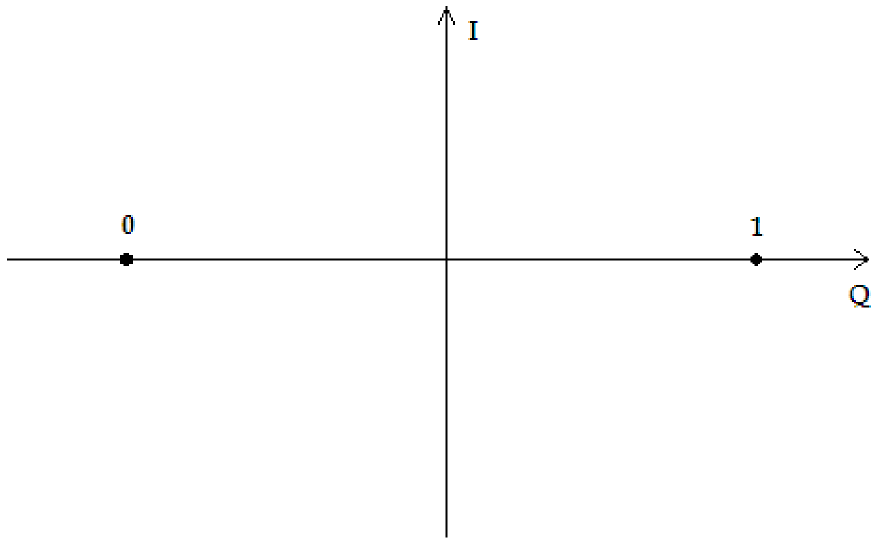

Figure 9: BPSK symbol mapping

We can easily see that for Binary phase shift keying, since the phase shift is always 180 degrees, the quadrature signal is always zero. This means that we only need to analyze the In-phase signal and this makes the whole system easier and cheaper to implement.

2.4.2. Differential encoding

In differential encoding, information is not conveyed by the absolute phase of the signal with respect to a reference, but rather by the difference between phases of successive symbols, thus eliminating the requirement for a phase reference at the receiver. In practice, this method gives greater immunity to fading.

Figure 10 illustrates how phase shifts carry the information in differential BPSK and QPSK.

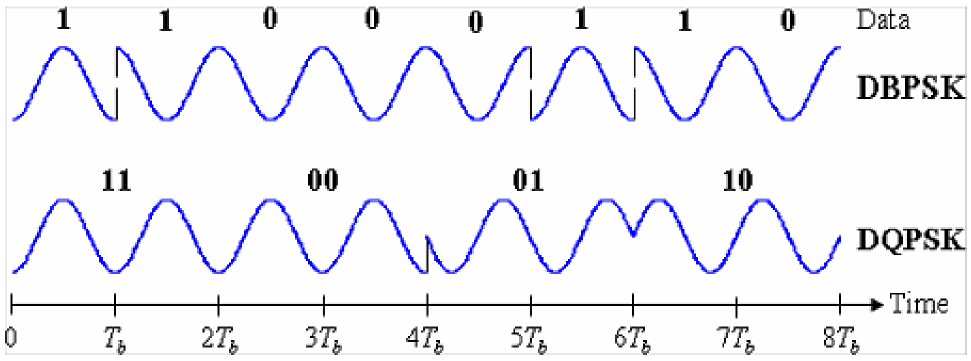

Figure 10: Differential BPSK & QPSK symbols

Chapter 3: Design

3.1. General system design

We begin by assuming that analog I and Q signals (or just the Q signal if we are not interested in the PSDU header) of the phase shift keying modulated baseband signal stream are picked off of the radio chipset and applied to a two channel, high speed analog to digital converter (ADC). The resulting digital bit stream is placed in a high speed FIFO buffer. In parallel with these bit streams, a time stamp from a hardware clock is also stored. The output of the FIFO is made available to a computer which performs digital signal processing operations to extract information from these bit streams. An outline of how the system works is as follows:

1. The arriving analog intermediate frequency signal data is sampled, digitized, and time stamped. It is then stored in a FIFO buffer that is written at the sampling rate of the ADC. The FIFO is large enough to hold enough data such that no packet's header-information will be lost due to the processing latency of the new hardware that will be co-located with the access point.

2. Demodulated and decoded packet data is provided to the DSP as shown in figure 11. Of course one could also use the MAC processing by the DSP in the AP to perform this decoding. When a packet is identified as being from the source of interest, the DSP will locate the stored digitized and time stamped data corresponding to the packet of interest. The packet identification occurs in near real time, so information in the FIFO will not be overwritten by the next packet.

3. After locating and moving the stored digitized and time stamped signal data to the DSP, it will reconstruct the waveform by extract symbols and bits. When a suitable pattern is detected, then the time stamp for that bit's arrival is sent to a central server which uses information from multiple receivers to compute the location of the mobile device.

4. The location server will obtain time stamps for the arrival of the same pattern from three or more access points. It will then perform the multilateralization calculations necessary to determine the transmitter's location.

Note that the signal processing is only concerned with reconstructing the signal's phase changes, and determining when they occurred. Because the data is time stamped deterministically in hardware this signal processing does not have to occur in real time. Another advantage is that the actual decoding of the data packet occurs normally in the existing access point hardware and software. This means that the new processing does not need to demodulate or decode the actual data packets, which means that the system can work even with encrypted data.

Note also that the design shown in figure 11 is a potential design for a future access point, but in the tests done in this thesis there is a separate computation unit for the new signal processing – since we could not reprogram the existing DSP in the access point.

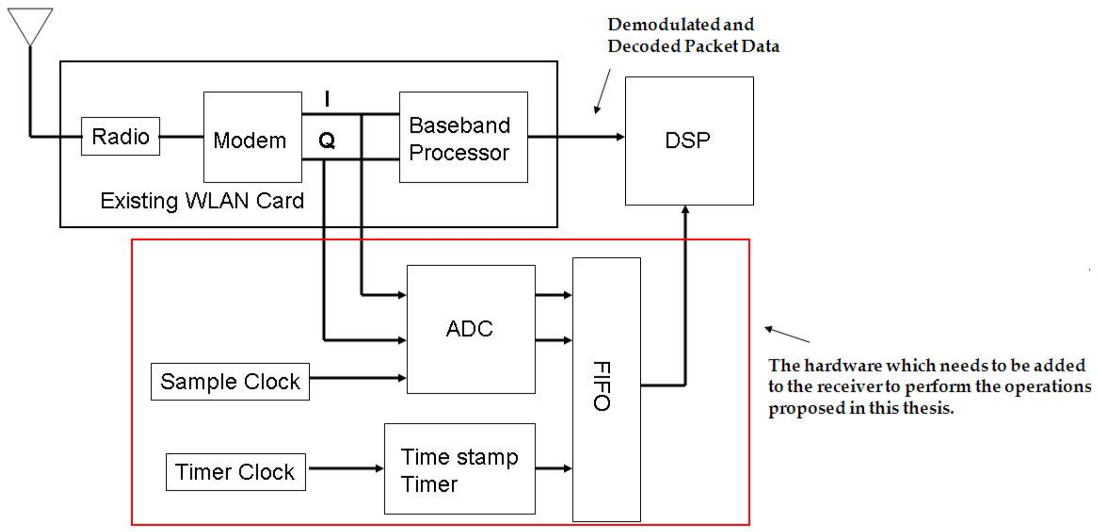
Demodulated and Decoded Packet Data
Radio
Modem
I
Q
Baseband Processor
Existing WLAN Card
DSP
ADC
Sample Clock
FIFO
Timer Clock
Time stamp Timer
The hardware which needs to be added to the receiver to perform the operations proposed in this thesis.

Figure 11: Location measurement receiving system

3.2. Advantages

There are several advantages of this design.

The first is that the signal strength is not a concern since we exploit other properties of the incoming signal. By doing this, we avoid a lot of complicated signal analyses and filters. Additionally, since we only care about the **first** incoming signal, no multipath propagation considerations are needed and therefore we need only consider an additive white Gaussian noise channel.

Another advantage is, since the transmitter will most likely send multiple frames within a short period of time, and if the user is a 'normal' person (i.e., stationary or moving only at low velocities), we can track the user even if we sometimes get unreasonable positions because of large obstacles that block the line of sight signals, hence a reflected ray arrives first leading to an incorrect estimate of the user's true position.

One important thing to remember is that we look at samples instead of bits. The reason is that the bits need first to be decoded from samples, and the bit rate in the physical layer convergence procedure has been sent at 1Mbps, which if we looked at the bits would only give a resolution of a hundred meters. That is not what we want. The samples are Barker code spreaded chips which are set at a rate of 11M chips per second. To extract this information, a sampling rate at least 22M samples per second is needed according to the Nyquist theorem. However, higher sampling rate gives us greater timing resolution - since we are not simply aiming to reconstruct the original waveform, but are in fact interested in the time it take the wave to propagate. There is a tradeoff between good sampling circuits and cheap sampling circuits. Therefore, additional analysis is needed to determine what the best solution is. This will be discussed in detail in Chapter 4.

3.3. Detailed description

3.3.1. Positioning

The position is determined by combining arrival time information from three access points and calculating the distance of mobile transmitter to each of them.

The difficulty of this approach is that since the radio signal travels at nearly the speed of light, the error in positioning will be large when using a low sampling rate. At 11 million samples per second, the distance represented by the difference between 2 samples is 27 meters. This is not sufficient to do a local positioning.

To encounter this, the first thought was that if the deviation is deterministic, we can repeat the measurment many times and get a more precise value. Or we can implement a function that for each pair of access points, gives us a set of potential positions of the user.

Since we don't have any information about how the time of arrivals will be distributed, the actual position becomes harder to determine. One way to increase the time resolution is that to increase the sampling rate. The hardware we use in this project, the Ettus USRP board [9], has a maximum sampling rate of 64 millions samples per second. This will improve our resolution from 27 meters to about 5 meters.

To improve it more, we need to utilize statistical methods. There are several proposals about how this can be performed. First we need an estimation of the timing error distribution. If the error is deterministically distributed, then what we need to do is simply subtracting this error from the signal arrive time measurements at the access points.

But error can also be stochastic. If so, then there are two possible cases: the error distribution can be estimated, or it cannot. In the first case, the solution will be to apply a statistical analysis to the distribution, and subtract the expected value of the error from the signal arrive time measurements. In second case, the situation becomes more complicated. Since we cannot tell how the error is distributed, we could simply average the error of a lot of measurements. Thus at least reducing the effect of the error. Hardware test has not been performed in this thesis.

3.3.2. Time stamping

To time stamp the samples, we use a clock that is located on the board. This clock needs to be synchronized with the other access points – in order to provide a common time base. This common time based will be derived using NTP [5]. This is straight forward to implement in the FPGA. There could be problem if the time stamping clock has a lower frequency than the sampling clock, but in our case both have the same frequency.

3.3.3. Timing measurement

This is probably one of the most important issues for this project.

As stated before, the accuracy of position determination depends on how accurate we can measure the arrival time at different APs. As radio waves propagate at the speed of light through air, a small unit of time means a large distance. It is for this reason that many earlier indoor location systems used acoustic signals, since the speed of propagation of sound is much lower than the speed of light.

Since our approach is based on signal arrival time measurements, we need to time stamp the arrival time reliably. To do this, we perform correlation (correlation, also called correlation coefficient, indicates the strength and direction of a linear relationship between two sequences. In this thesis, correlation has been utilized to detect the signal arrival). This can be done either on a digital signal processor or a FPGA (field programmable gate array). We need to compute the correlation peak and when it occurs. Since there is already a FPGA on the USRP board, we will in this thesis only consider a design based upon correlation implemented in the FPGA.

A critical issue is the sampling rate. While there are ADC circuits that have very high sampling rate, this is not true in our case as we want to create a design that can use relatively cheap hardware in order to make this system practical. The hardware we are planning to use has a sampling rate of 64M samples per second (Msps). This sampling rate is not high enough to directly give use the arrival time with the temporal resolution which we might want. We cannot be sure if the time stamps tell us the exact signal arrival time or not, because the signal arrival time can be anywhere between the time stamp Tstamp minus a half sampling time and the time stamp plus half the sampling time, {Tstamp +/– 0.5*Tsample}. Since we are not sure about where in this interval we are, we need to consider both +0.5*Tsample and -0.5*Tsample to include all possible time intervals. The sampling time at our sampling rate is 1/64M which is about 15 ns, which corresponds to about 5 meters in distance (hence 15ns * C = 4.5 meter, C = light speed). As stated before, we estimate the position of the mobile device based upon the overlapping area of the estimated distances from the three access points. By doing this measurement several times the possible position area shrinks, see figures 12 and 13.

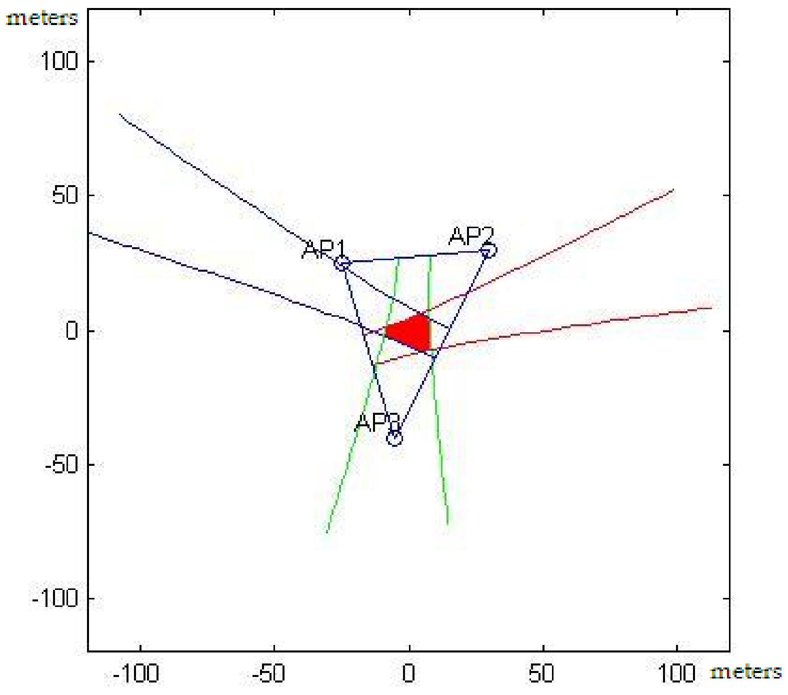

Figure 12: The overlapping area is the possible user position area

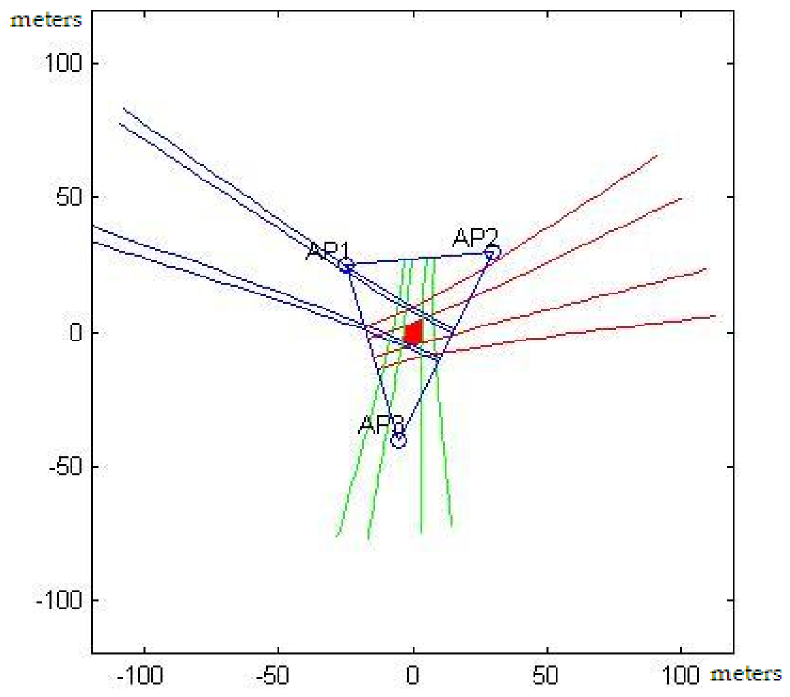

Figure 13: The overlapping area shrinks when performing the estimation several times

This computation can be done posterior, we do not need to implement it in the FPGA, but some post processing in MATLAB or with other software will be sufficient.

In IEEE 802.11b, the mobile device actually synchronizes with the access point which it is associated with, [12]. And since all access points in our system is synchronized, what may happen is that the time estimation will be good after the synchronization and getting worse and worse as time goes by, and becoming good again with next synchronization. This is good because if the error propagation is limited, we will get better estimation faster. Future researchshould examine if it is possible to predict the error.

3.3.4. System model

The model uses four WLAN cards. All have a fixed position, with three of the cards being located about the periphery and one being located interior to the hull formed by the other three. The one in the interior will be considered to be the mobile device of a user. This WLAN card sends out data packets constantly or at some rate (e.g., every 20ms such as a voice over IP client would). The other three act as APs and each one has its own USRP attached to the I&Q signals. This model simulates the realistic case of a user located in a small area. The three USRP are synchronized to the same clock. The distance to the user is fixed and known. Every time the user sends a packet, all three APs will detect it and time stamp their samples. There will be a significant time delay between the signal's arrival and the signal's detection, but as described we be able to calculate a position estimate, the goal is this estimate should not be too far from the actual user's position.

There is also another proposal for how we can simulate the user case with fewer WLAN cards and USRPs. That is, instead of using four WLAN cards and three USRPs, we use only one of each. To get measurement data for the case we described above, we do the measurement three times with different lengths of wire emulating the different distances to the user's device. This avoids signal interference (since we have a wired connection). To simulate a moving user, we shift the sending data with a suitable delay. The USRP board we use has four 64Msps 12-bits ADCs on, and since we only need to consider the In-phase signal, we should have enough inputs to emulate 4 access points. As the USRP board also has four 128Msps 14-bit digital to analog converters (DAC) it is also feasible to test the system by generating a known waveform and detecting it.

3.3.5. Test bench

The test bench utilizes the GNU software radio (http://www.gnu.org/software/gnuradio/) and USRP (universal software radio peripheral) board from Ettus Research LLC. The USRP performs the sampling and analog to digital conversion. The sampling rate is 64Msps. The FPGA that is sitting on the USRP is an Altera Cyclone EP1C12 (see section 5.1 for details). The GNU radio project uses Verilog HDL as the programming language to configure this FPGA. Here is a review article about this board: (http://spectrum.ieee.org/oct06/4654).

The GNU radio is a C++/Python hybrid system tied together by SWIG (simplified wrapper and interface generator, www.swig.org). Other libraries such as FFTW (www.fftw.org), Boost (www.boost.org) and CPP unit (cppunit.sourceforge.net/cppunit-wiki), are also necessary in the system construction. These will all be compiled using an Ubuntu Linux system.

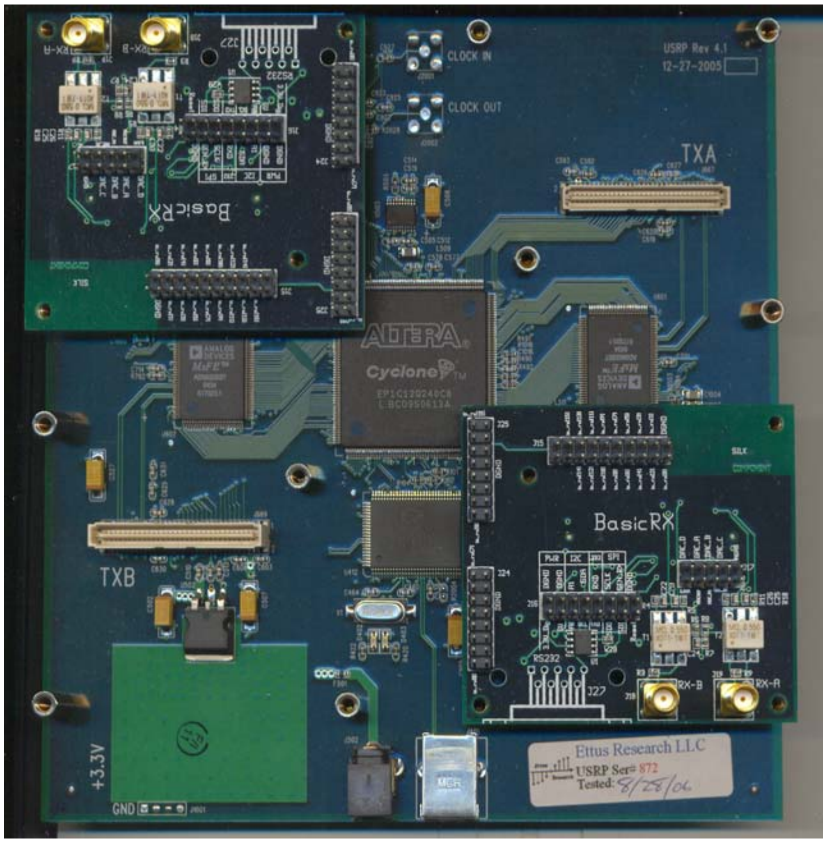

Figure 14: The Universal Software Radio Peripheral (USRP), reproduced by permission, Mark Smith 2007.

Chapter 4: Simulation

4.1. IEEE 802.11b data frame analysis

Because the PLCP preamble is always scrambled ones and always sent at 1 Mbps (that is, Barker code spreaded and DBPSK modulated), the pattern we will look for is always the same. Therefore the correlation can take place as soon as the intermediate frequency signals arrive, which gives us a significant gain in timing accuracy.

The scrambled one's will always be 0011 1111 0111 0110 0111 0100 0101 0010 0111 1100 1100 1010 1100 0110 0001 0010 1111 0101 0100 0001 0110 1011 1001 0001 1100 0000 0111 1000 1000 0110 1001 1011 in big endian order. In fact, the most unique sequence in the PLCP (Physical Layer Convergence Procedure) is the SFD (Start Frequency Delimiter) part, but this field is too short to be easily identified by the correlation. But since the scrambled one's already have good correlation property, SFD is not needed in the header correlation. However, the SFD is useful since it tells us where exactly the preamble ends, hence we can look for this sequence when later processing the header data to find MAC addresses (if needed).

The SIGNAL field indicates which transmission rate the data portion of the frame utilizes. Based upon the transmission rate we know how the frame is spreaded and which modulation has been used. This information is very useful in next step which is to decode the sender's MAC address to see if the data frame comes from the desired user's device.

Since the main purpose of this project is location determination, the decoding of MAC header for data rates higher than 2 Mbps has been skipped, although CCK decoding is quite straight forward.

4.2. MAC address decoding

To identify a specific user's device, we need to look at the sender's MAC address. As stated before, the MAC addresses which are in the PSDU field of PPDU header can be coded and sent at different data rates. This makes it more complicated for us to identify the user's device. However, since we know which device we are looking for we can precompute the pattern for each of the encodings and simply look for an instance of one of these patterns or we can let the normal MAC handling of the AP handle this.

4.3. Investigation of the frame using simulation

To analyze the IEEE 802.11b data frame, several experiments were performed using MATLAB. This leads to a number of useful insights which are described below.

The first is that fields in the IEEE 802.11b PPDU header are in big-endian (which means the first bit is the most significant bit).

The SFD field, as stated in IEEE 802.11b standard, is a unique sequence. This sequence is very useful for finding the beginning of the PLCP header and thereby extracting information from the MAC header. However, this sequence is 16 bits long, and since the PPDU header is always Barker code spreaded, the SFD sequence has 176 chips. This length was not enough to find the frame as there is insufficient correlation to detect it. Therefore, the 128 bits SYNC sequence has also been used, as this results in a 1408 chip pattern. Although a longer correlation is necessary, this does not matter since the correlation still can be computed at the rate which new samples arrive.

Additionally, correlation is needed during post processing to recognize the MAC address in order to identify the desired transmitter. Although there is only one user in our test case, there may be many other WLAN users in the test area. There is another way to do this, which is that let the WLAN card do the user identification. When the location hardware is added to a WLAN card, it is easy to use this second method. But for now, we will perform post processing to identify the specific transmitter which we are interested in.

4.4. Sample correlation

Several experiments were performed which used correlation of different length sample sequences. The results are satisfying: even if the correlation begins after the start of frame; a reliable correlation peak results. With an 11Msps sampled sequence, 1500 samples of the correlation sequence, the second highest peak, which we know does not belong to the header, has a value of about 500, while the highest peak has a correlation in about 2000. With a reduced length of the correlation sequence, this value decreases linearly. Thus at half the length, the correlation peak sinks below 500.

Another experiment tested things the reverse. That is, with a full length correlation sequence, but only part of the incoming samples, beginning after several hundred samples into the frame. This simulation shows the correlation performance even if we are not able to detect samples from the very beginning of the SYNC sequence. The result is similar to the earlier case: The correlation decreases as the number of samples missed increases. We get a correlation lower than 500 with 800 missed samples.

These results tell us that the correlator will be reliable even in the worse cases of losing more than have the sample of the synchronization sequence. Additionally, we expect to have multiple complete frames to examine, hence the loss of occasional frames will not matter significantly.

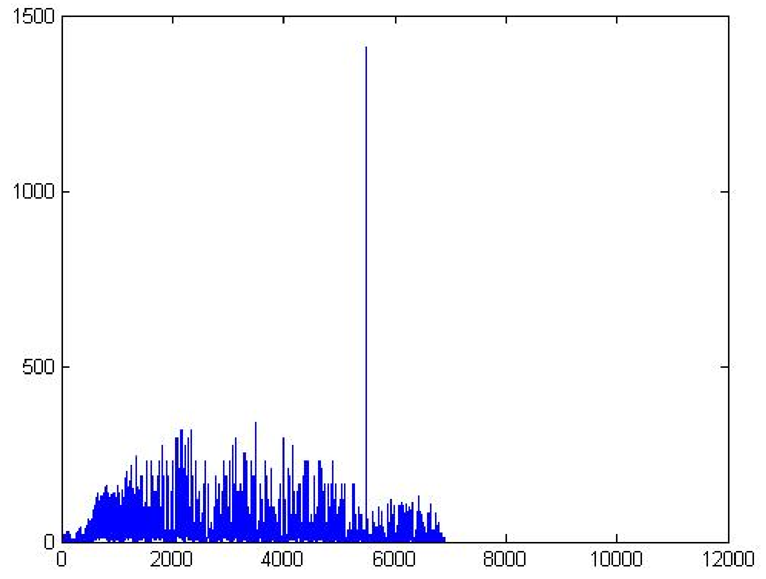

Figure 15: Sample correlation between preamble and data frame, 11Msps

In our case, the correlation will use a 64Msps sampled sequence. This expands the length of the sample sequence by a factor of 6, from 1500 to nearly 9000. With increased sampling rate, we will get a better temporal resolution, and with a longer correlation sequence, the peak of correlation becomes even more obvious. See figure 16.

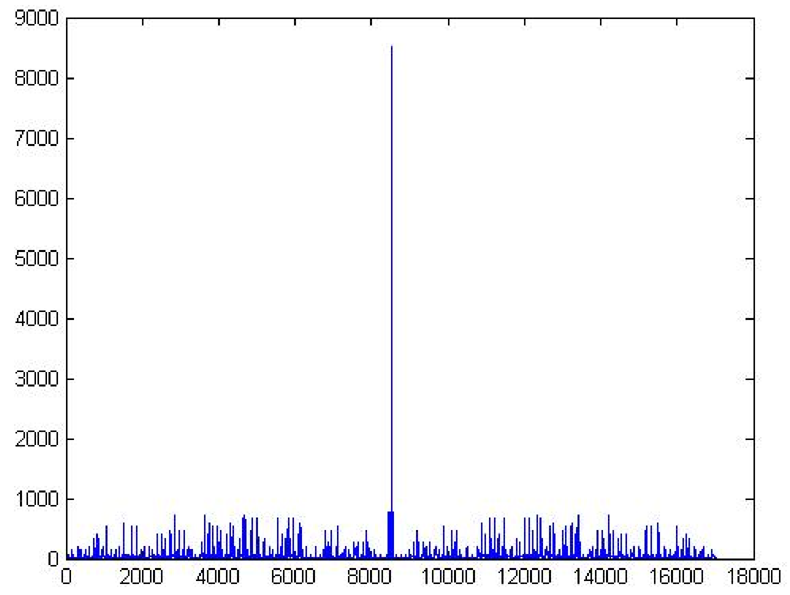

Figure 16: Sample correlation at 64 Msps

4.5. Sampling time error analysis

The sampling rate at 64Msps is relatively slow relative to the speed of light. While the correlation tells us when the sampled SYNC sequence arrives, it does so only at the temporal resolution of the sampling clock. Thus we know that the actual time of arrival could be off by up to one clock period. To illustrate the effect of this, a MATLAB simulation of the positioning error has been performed.

We ignore the radio signal processing time in the simulation because we know that this processing is deterministic. Although, this time must be accounted for in the real implementation in order to get a reliable position estimate.

Since we don’t know how the timing error is distributed, we assume in this simulation that this error is uniformly distributed. The total timing error is 1/Sampling rate which for 64Msps is 15.6 nanoseconds. According to Maximum Likelihood, this error is evenly distributed over the time stamp of the current sample, which means that the timing deviation is plus and minus a half sample time = +/- 7.8 nanoseconds. Thus the user's current position is constrained to a cylindrical area from a give AP.

Assume that the user and three access points are perfectly synchronized, and the user is not moving. The actual distance from the three access points to the user is 35 meters for AP1, 42 meters for AP2 and 40 meters for AP3. Timing deviation between -7.8 to 7.8 nanoseconds gives us a distance error between 0 and 4.68 meters. The overlapping area marked with green in figure 17 shows the area where the user is.

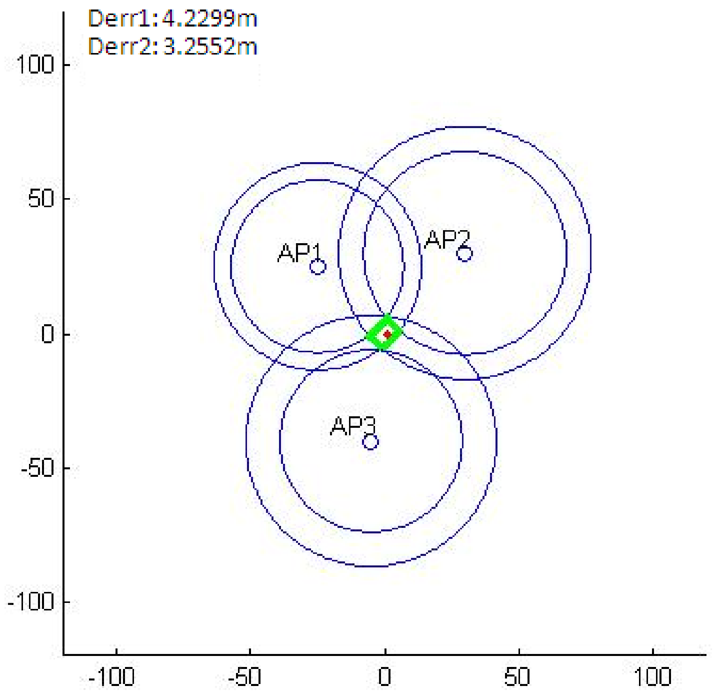

Figure 17: Location estimation with uniformly distributed timing error

To get a more precise position estimation, we perform this measurement 10 times. It results in a decreased area of the user position.

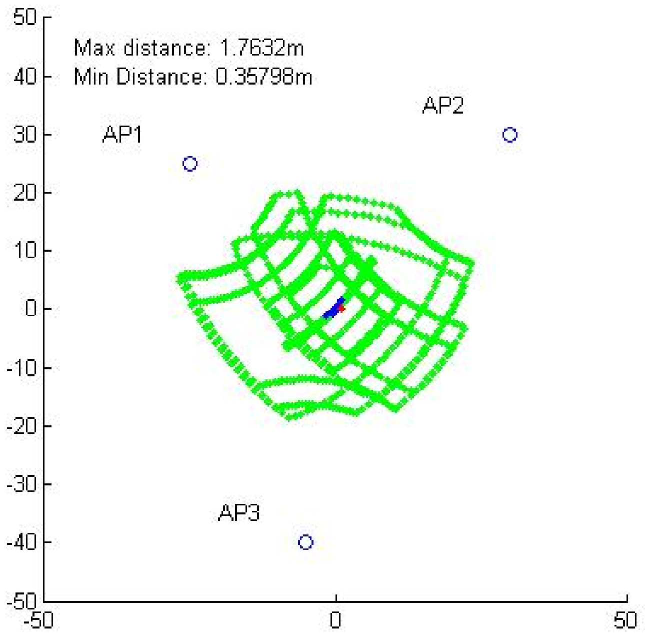

Figure 18: Overlapping of 10 possible user positions

In figure 18, the blue marked area is the overlapping area, which is the area of the possible user position. We can see that the estimation error decreased 0.36 meters. This simulation shows that the overlapping of possible user positions will give us a good estimate of the user position (since the actual user position must correspond to the measurement data from **all** the measurements).

4.6. Position determination by arrival time

In a real implementation, we will not be able to know the exact distances from the user to the access points, but only the time stamps which indicate the packet arrival time.

In this case the algorithm to determine the possible user position is:

1. Calculate the arrival time differences from three access points, which form 3 constraints.

2. Add plus and minus one half sampling time to the time differences, this results in 6 constraints.

3. Derive the possible simultaneous solutions within the area. Since we know the exact coordinates of the access points, it becomes a simple system of equations with 2 equations and 2 unknowns. For each pair of access points, we need to solve this system of equations twice considering the difference of the signal arrive time Tdiff + 0.5*Tsample and Tdiff – 0.5 *Tsample. The results are 6 sets of coordinates.

4. From these 6 sets of coordinates we form a bounded area of the possible user position.

5. Perform this calculation several times to get an improved position area estimate.

The result will be presented in section 6.1. The MATLAB code for this algorithm is included as appendix A.

Chapter 5: Implementation

The implementation has two 2 phases. The first is the single USRP implementation to investigate the difficulty of realizing the designed system. This part is the key to the complete system implementation. Once the single USRP implementation is finished, the whole positioning system would be very easy to put into practice.

5.1. Single USRP implementation

The most important parts of the single USRP implementation are the time stamping and calculation of the correlation.

The FPGA we use in this project is an Altera EP1C12 Cyclone. It is not one of Altera's most advanced products, but it is sufficient for our system design.

The EP1C12 Cyclone field programmable gate array is based on a 1.5-V, 0.13-µm, all-layer copper SRAM process, with 12,060 logic elements (LEs) and up to 234 Kbits of RAM. With features like phaselocked loops (PLLs) for clocking and a dedicated double data rate (DDR) interface to meet DDR SDRAM and fast cycle RAM (FCRAM) memory interface requirements. This Cyclone device is a cost-effective solution for data-path applications. It supports various I/O standards, including LVDS at data rates up to 640 megabits per second (Mbps), and 66- and 33-MHz, 64- and 32-bit peripheral component interconnect (PCI), for interfacing with and supporting ASSP (Application Specific Standard Product) and ASIC devices, [1].

The time stamps are derived from a 64MHz Fortiming Corp. (www.4timing.com) HC49USMD crystal clock on the USRP. In future implementations, since the APs must be synchronized, all USRPs should be synchronized to the same clock.

5.2. Performing the correlation using the FPGA

As stated before, we use the 64Msps sampled sequence to perform the correlation in the FPGA in order to determine the signal's arrival time. However, correlating a sequence of 9000 samples is unrealistic to implement in a Cyclone FPGA. The ADC samples the signal using a resolution of 12 bits. Hence 9000 samples becomes 108000 bits. Even though the correlation contains only simple logical operations, this size is still far too large for this FPGA.

Thus we need to decrease the sequence size. One theory is that, since the In-phase part of BPSK signal only contains plus and minus ones (as shown in figure 9), it is enough to perform the correlation by using the signs. This decreases the sample size from 12 bits to 1bit, and the total size to 9000 bits.

The next question is, is 9000 bits still too much for our FPGA? If so, can we decrease the requirements yet again? How many samples are needed to find a reliable correlation peak? A simulation was performed to find the answer. Figure 19 shows that with 1000 bits, we are still able to identify the correct correlation peak.

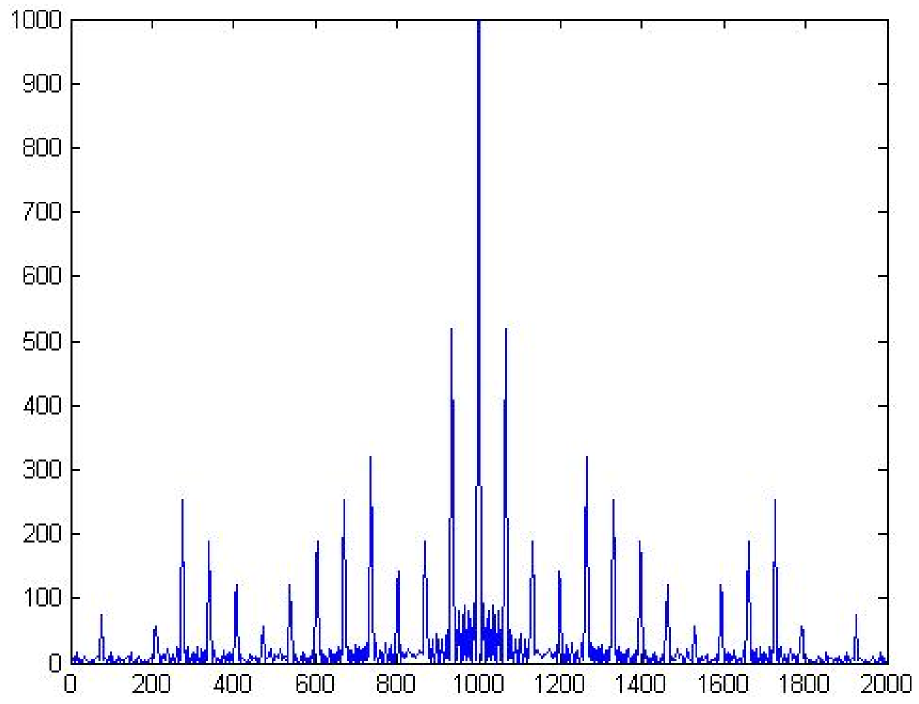

Figure 19: Cross correlation with 1000 bits

5.3. Verilog implementation

The hardware description language (HDL) used was Verilog HDL. Verilog is a C-like language that is used to model electronic systems. It supports digital design and verification, and can be used in a limited way for mixed-signal circuits. The language is case-sensitive, has a preprocessor like C, and the major control flow keywords, such as "if" and "while", are similar. The formatting mechanism in the printing routines and language operators and their precedence are also similar. However, it differs from many programming language in that the execution of statements is not strictly sequential.

A Verilog design consists of a hierarchy of modules. Modules are defined with a set of input, output, and bidirectional ports. Internally, a module contains a list of wires and registers. Concurrent and sequential statements define the behavior of the module by defining the relationships between the ports, wires, and registers. Sequential statements are placed inside a begin/end block and executed in sequential order within the block. But all concurrent statements and all begin/end blocks in the design are executed in parallel. A module can also contain one or more instances of another module to define a sub-behavior. A subset of statements in the language is synthesizable. If the modules in a design contain only synthesizable statements, software can be used to transform or synthesize the design into a netlist that describes the basic components and connections to be implemented in hardware. The netlist may then be transformed into, for example, a form describing the standard cells of an integrated circuit or a bitstream for a programmable logic device (e.g. a FPGA). [18]

The Verilog implementation needed in this project consists of two main modules: **correlator** and **timestamp**. The correlator module contains a bitpattern with length 1000, which is pre-computed by MATLAB using preamble information about the IEEE 802.11b protocol as described in section

2.1. The bit pattern is the first 1000 bits of the sampled header. As shown above, this is sufficient for us to detect the signal's arrival.

The input of the correlator module is the incoming signal samples, mapped from sample values to ones and zeros. The reason we do this is stated above. For every incoming bit, the correlator will perform a digital correlation and a hamming weight calculation to return the correlation value. This value will then be sent to the timestamp module to decide if it exceeds the threshold; if so, then the timestamp is sent to the output.

There should be no multiplications or divisions used in the implementation, as a multiplier is very resource consuming on a FPGA. The correlation operation can be realized for our case as a bit-wise inverted logical XOR. The hamming weight operation is done by summing the '1' bits of a bitarray, and the return value is the sum of the correlation output. However, to illustrate the difference between correlation using multiplication and not using it, we have performed both implementations on a testbench. The result will be presented in section 6.2.

5.4. Multiple USRP implementation

In a multiple USRP implementation, the WLAN card that sends packets is assumed to have a fixed position. This makes it easier to do the timing measurement. Since in the test configuration the distances from the user to APs are known, it is easy for us to check our position estimation against these known distances. However, due to limited time, this case will not be included in this thesis.

Chapter 6: Results

6.1. Location determination

As presented in section 4.6, the position estimation algorithm calculates the possible user positions by using differences of two arrival times at two access points. To include all potential positions, we add plus and minus a half sample time to the measured time delay. The result for one measurement is shown in figure 20.

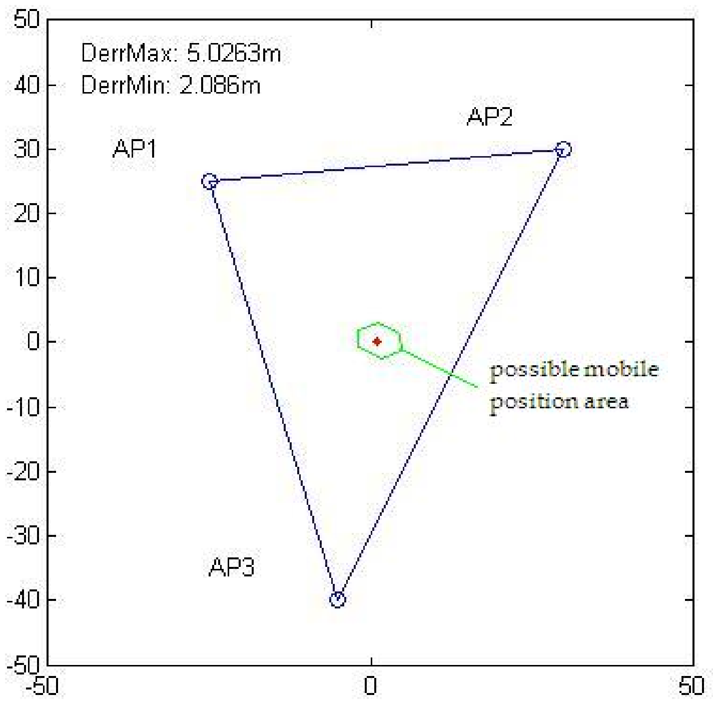

Figure 20: Result of the position estimation algorithm

The red point in figure 19 is the actual mobile position. The area of the possible mobile positions is defined by linking all six bounding points (based upon the pair-wise measurements). As we can see, the minimum estimate error is about 2 meters. Although this is a good estimation, we would like higher accuracy in our location determination.

To achieve this, we perform the measurement several times and overlap computer the **intersection** of the estimated position areas. If the mobile remains still, we do not need to include tracking (although it is a good idea to implement a tracking algorithm for the 'moving user scenario').

The result for the intersection of 10 simulated results is shown in figure 21. As we can easily see, the estimation area shrinks dramatically. The error of the position determination is between half to one meter, which is close to our requirement.

Another thing that is good to note about this implementation is that the result is similar for 10 simulations and 20 simulations. This is simply because although the estimated area shrinks to a smaller size, the calculated positions still contain random errors.

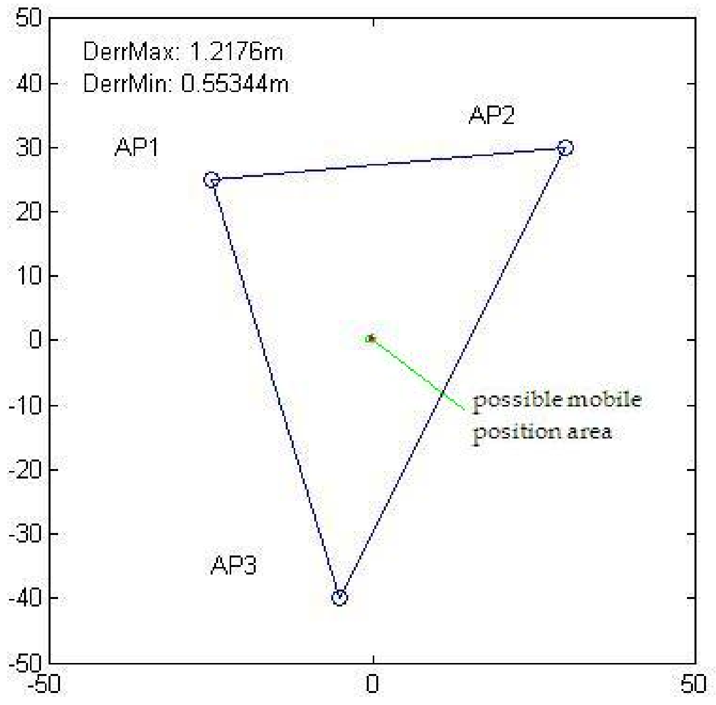

Figure 21: The error decreases to half a meter after 10 simulations with the area overlapping method

With a well synchronized measurement environment and a better algorithm, the estimated area will theoretically converge to nearly zero. This will be discussed further in Chapter 7.

6.2. FPGA based correlation

The FPGA implementation is written in Verilog and quite simple. The correlation can be performed either by using multiplications or logical XORs. The main benefit of using logical XORs is that we do not need to implement multiplications on the FPGA, which is very resource-demanding. First, FPGA on the USRP board does everything as 4 bit lookup tables (LUTs). So for example, to XOR two 2-bits numbers gives us 1 bit result, using a 4-bit LUT, while multiplying two 2-bits number gives us a 4-bits result, which needs 4 such LUTs to complete. This means that it takes 4 times the amount of power to perform two 2-bits multiplication as doing a 2-bits XOR. But since no LUT depends on the results of another LUT, the speed of these two operations is the same.

However, if we want to work with more bits, for example 3-bits number, the difference becomes more significant. XORing 3 bits still just takes one LUT to give a 1 bit result. Multiplying two 3-bit numbers will result in a 6 bit result. BUT, we can't build the multiplier from just 6 LUTs. One way to build it is as a 6 bit input device for each output bit. That means we will need three LUTs for each of the 6 output bits, and they will be connected as a binary tree for each output bit (2 LUTS giving partial results to a third LUT giving the final answer for that particular bit). In this case, the multiplier will take 18 times as many LUTs and thus 18 times the power as the XOR solution, and will run twice as slow because of the binary tree structure. The LUT driving each output bit has to wait for the partial result from the two LUTs behind it. So the more bits we use from each sample, the more benefit we get by using XORs instead of using multiplier.

But can we get as good result as we can get from multiplication by only using logical XORs? Following is a comparison between two correlations. One is done by XOR operation and the other is by multiplication.

Figure 22 shows a correlation peak at 1000 when 2 identical sequences have perfect correlation. As we can see, uncorrelated parts of the result have positive maximum at about 300 and at negative minimum at about 500. In figure 23, which shows the result from a correlation operation using logical XORs, the correlation peak is the same in a perfect correlated situation, but the uncorrelated part has a maximum at about 650. It is hard to say that the result using logical XORs is less reliable, since we get a clear peak in both implementations, but rather that it's less tolerant against noises such as signals from other access points in the area. The Verilog code is included in appendix B.

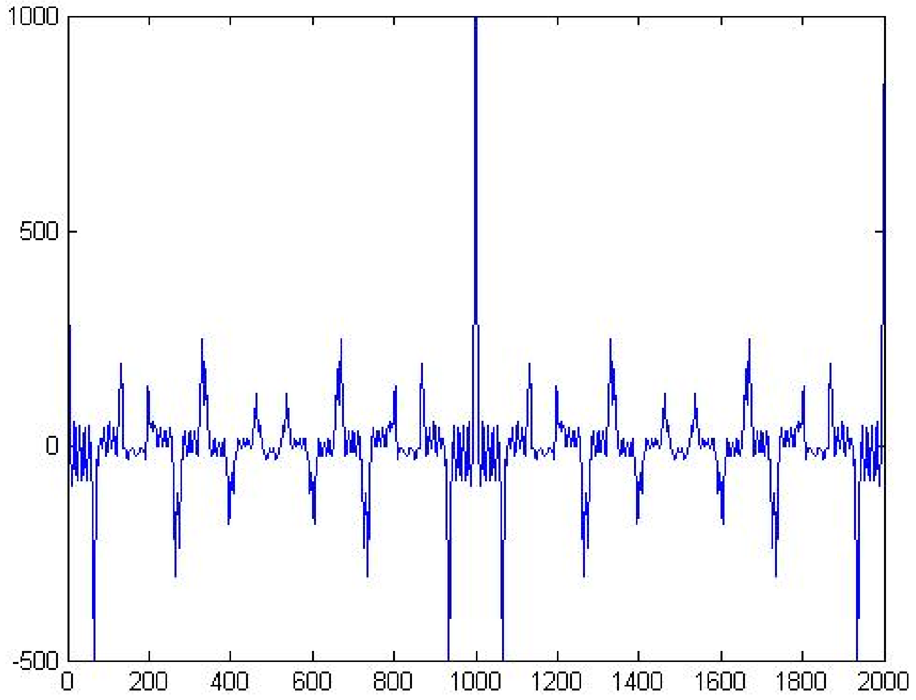

Figure 22: The bit-wise correlation using multiplication

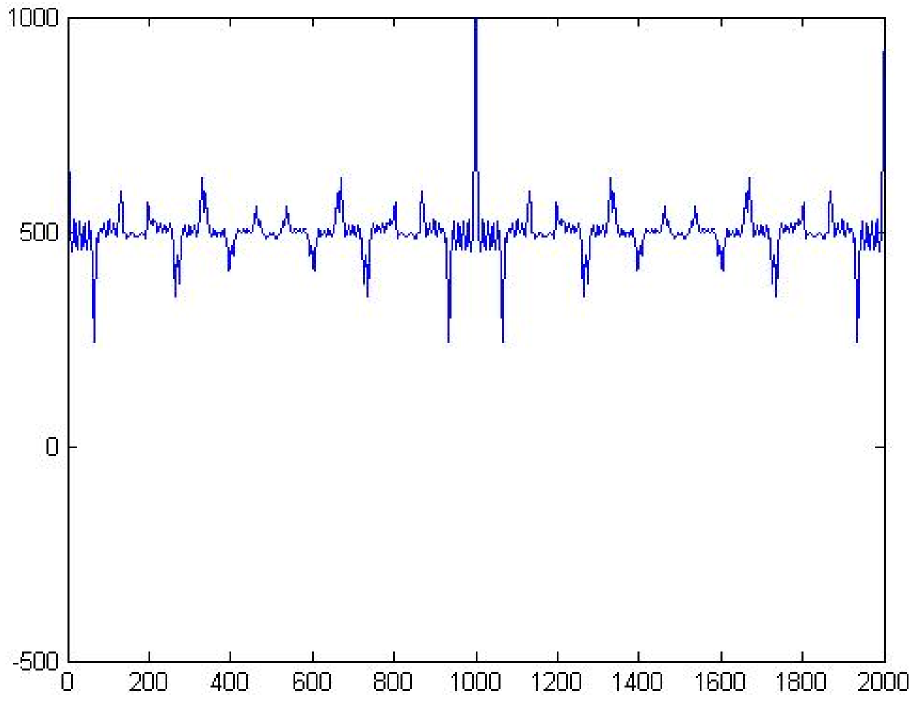

Figure 23: The bit-wise correlation using logical XOR

6.3. Work summary

Here is a summary of works that have been done. It is important to understand these aspects in order to continue this project:

1. The MATLAB functions that are developed for analyzing the IEEE802.11b header
2. The MATLAB functions that are developed for helping calculation of the mobile location.
3. The use of USRP board and recompiling of the GNU radio FPGA code.
4. FPGA functions that are developed for performing the correlation and time stamping.

6.3.1. Header analysis function

The MATLBAB simulation file for the IEEE 802.11b header analysis is *headeranalysis.m*. The file takes a data file called 'myfile2.dat' (can be changed to any name, as long as the name matches the data file), which contains coded and modulated IEEE 802.11b data frame, generated from Agilent's simulation program and extracted by G. Q. Maguire Jr., then decode/demodulate using the information according to the IEEE802.11b protocol (The Aglient's simulation tool is a very handy tool that can be used to generate IEEE 802.11 a/b/g dataframes with desired parameters.). Currently this MATLAB file only supports bit rates of 1 or 2Mbit/s, because the CCK decoding has not been implemented. The scrambler analysis can be switched on or off by changing the variable 'scrambler' to 1 or 0. Even though it cannot decode higher bit rates using CCK coding, it can tell if the bit rate is 1 Mbit/s, 2 Mbit/s, or unknown bit rate. This is done simply by checking the SIGNAL field.

One important thing to note is that this function uses the MATLAB function *dpskmod* and *dpskdemod* to perform the differential PSK modulation and demodulation. These functions are only included in latest version of MATLAB (which is version 7). When using an older version, these functions may not be available. It is not very hard to write similar functions to perform the mod/demod, and the theory is covered in previous chapters, (specifically section 2.4).

The synchronization is done by correlating the scrambled (or nonscrambled) header. This part will be replaced in the real implementation by the FPGA based correlation. Once the frame arrival is insured, we use **Start Frame Delimiter** field (referred as SCRSFD, scrambled SFD) to find the starting point of the MAC address. Note that since the SFD field is always the same, you can use it without decoding.

The Agilent's simulation program used in this thesis is called *E4438C-417 Signal studio for 802.11 WLAN* and can currently be found here:
http://www.home.agilent.com/USeng/nav/-536902336.536883181/pd.html

6.3.2. Mobile positioning function

The function is called *realcase.m* (see appendix A.2), and is uses an utility function *sortarg.m*, which sorts an array of complex values using their arguments. This is used to compute the result of the location determination function, resulting in an array of 6 complex values, and we need to sort them to plot an area by linking them.

The test bench file of this function is called *realism.m* (see appendix A.4). Note that the coordinates in the test bench file is only used to plot the position area. The time error is simulated with an evenly distributed random variable. In the later implementation, the receive time known as RecT1, RecT2, and RecT3 should be replaced by the received time form the timestamps coming out from the FPGA's output, and the random time error Terr1-3 shall be removed.

Another correction that should be made is that the signal speed is set to 10^8 meter/second now, and should be reconsidered when implementing with the hardware as light speed in air is about 0.8c, and not 1c. This will improve the system performance slightly since with reduced signal speed, the time error becomes less crucial.

As the function is written now, we estimate the mobile's location by linking the 6 output values from the location determination equations. Then we illustrate the effect of multiple measurements by updating the values and saving the closest ones. That is why the error never converges to smaller values even with increased number of calculations. In a tracking system, you do not know the user's original location, or where the user is moving. Ergo, improvements are necessary. First, the whole area defined by the solutions of the location determination equations shall be defined in coordinates /pixels. This means that we save the coordinates of all possible locations of the mobile device. Then after each calculation for the new possible position area, pixels/coordinates within the area shall be updated, and only overlapping regions should be saved. With this method, we will achieve a large improvement in location accuracy.

But, keep updating constraints of an area often is a resource demanding computation. One can ask questions like 'how many pixels/coordinates shall be defined in one quadrate meter?' or 'How often shall we update the coordinates of the points to be able to track the user?'. The more coordinates we define within an area, the shorter distance will it be between two coordinates, and thereby we get a higher positioning resolution. However, this is more close to an image processing problem rather than a communication system problem.

6.3.3. USRP and recompiling the verilog code

The Universal Software Radio Peripheral is a low-cost board designed for running GNU radio code. It is a flexible platform that can be used to implement real-time applications. The GNU radio is a powerful code library that allows you to implement signal processing for digital communications. It has very useful support for both coding and modulation for sound signals and video streaming, and most important: it is a part of GNU project, which means developers over the whole world are working to improve it.

There is an installation note available on http://comsec.com/wiki?GnuRadioWiki. Its hyperlink is also included in references to ease the future work [10]. The installation of GNU radio code is quite straight forward, although you may need to do some compiling and tests to get the code running. Once this is done, the only part that we are interested in is the FPGA.

To recompile the verilog code is tricky. There are no documentations about how the FPGA is programmed, because the USRP board was built for a single purpose: to run the software radio code. However we were lucky – there are a few people in the world who are also interested in reprogramming the FPGA. One very useful paper is from *Oussama Sekkat* at UCLA. It clearly describes the USRP construction and how to build and load new verilog code [11].

Because of the limited time, I could not perform this experiment, but the information contained in this thesis should be sufficient to do so.

6.3.4. FPGA functions

To finish the hardware implementation, we need to replace the original verilog code of the FPGA on the USRP with our own. The down sampling part shall be replaced by the pattern correlation which is described in section 6.2. The code is written using M4 macros, in order to shorten the code length. The code included in the appendix B is the wrapped version, so one needs to unwrap that. To do it, execute the following command:

```
C:\m4 filename.verilog > filename.v
```

where *filename* is the name of the verilog source file. A document about GNU M4 is [6].

After the unwrapping, the verilog code will need to be passed through a synthesis tool like Altera Quartus II. In order to successfully synthesize, the verilog code may need some changes to reflect the mapping of the logic functions onto the architecture. The USRP directly reads the compiled binary file, so no file transfer to the FPGA is needed, although you need to put the binary file in the right place. See details in Sekkat's document [11].

Chapter 7: Conclusions

7.1. Conclusion

The result presented in chapter 6 seems encouraging. We have shown that the method of location determination is effective, and successfully designed a circuit which could be realized in the FPGA to implement our solution.

By using a relatively low sampling rate and inexpensive FGPA we are able to perform an accurate location determination by analyzing signal arrival times at three different access points. By doing this, we avoid problems with methods that use signal strength and thereby have problems with determining the location of devices in an indoor area (especially one with people moving about in it). Additionally, we do not need to modify existing mobile devices' architecture, but only need to extend the APs.

The use of IEEE 802.11b protocol was a good choice. It is well described and developed, and has many advantages when attempting to detect the location of users. The preamble has good correlation properties, and the frame structure is simple to understand. On the other hand, one needs to understand the coding and digital modulation used in order to exploit this protocol. However, since the protocol is widely available and used extensively, it is just a matter of time before future developers implement solutions such as described here.

The FPGA design shows that it is possible to perform the computation on the FPGA on the USRP board. Since the rest of the verilog code which controls the GNU radio input and output is not needed for this project, the FPGA will have sufficient gate resources for óur program. Analysis of the FPGA implementation also shows that our solution is possible without using multiplications or divisions.

All the simulation and implementation files are found in Appendices A, B, and C.

7.2. Future works

Several goals that have been listed in section 1.4 have been achieved:

1. The sampling frequency at 64MHz is theoretically enough for a good timing measurement.

2. The correlation property of the samples of the Q base band signal is enough for us to detect signal arrival.

3. Time stamping frequency is the same as sampling frequency, and is sufficient.

4. Possible error sources are for example synchronization error of the access points, and noise from other mobile devices which use the same protocol.

However, what this thesis has shown is theoretical analysis and solution proposal. These need to be tested with hardware experiments.

As stated before, I could not finish some of the work because of the time. However, developers who are going to continue with this work are welcome to contact me at linusji@gmail.com with further questions.

References

[1] Altera, *Cyclone Device Handbook*, last visited 2007-01-23, Internet. URL(http://www.altera.com/literature/hb/cyc/cyc_c5v1.pdf)

[2] Altera, *Introduction to Altera Quartus II Manual*, Altera, 2006, URL(http://www.altera.com/literature/manual/intro_to_quartus2.pdf)

[3] Analog Devices, *Mixed-Signal Front-End (MxFE™) Processor for Broadband Communications AD9860/AD9862**, Analog Devices, 2002 URL(http://www.analog.com/UploadedFiles/Data_Sheets/AD9860_9862.pdf)

[4] Behrouz A. Forouzan with Sophia Chung Fegan, *TCP/IP protocol suite, Third edition,* McGraw Hill International Edition, 2003, ISBN 0-07-111583-8

[5] David L. Mills, *Network Time Protocol (Version 3): Specification*, Implementation and Analysis", RFC 1305, Internet Society, March 1992.

[6] GNU M4, *M4,* last visited 2007-01-15, Internet. URL(http://www.gnu.org/software/m4/manual/m4.pdf)

[7] IEEE , *IEEE Std 802.11b-1999(R2003) (Suppliment to ANSI/IEEE Std 802.11, 1999 edition)*, IEEE, 2003, ISBN 0-7381-1812-5

[8] Jim Geier, *802.11b Physical Layer Revealed,* 2005, Internet URL(http://www.wirelessnetworkingacademy.com/learning_center/tutorials/80211b_Physical_Layer_Revealed.htm)

[9] Matt Ettus，*Universal Software Radio Peripheral (USRP),* Ettus Research LLC, 2006 URL(http://www.ettus.com/downloads/usrp_guide.pdf)

[10] Naveen Manicka, *GnuRadio Installation Note*, 2006-05-25, Internet URL(www.eecis.udel.edu/~manicka/Research/**GnuRadio_InstallationNote**s.pdf)

[11] Oussama Sekkat, *The FPGA*, Department of Electrical Engineering, University of California, Los Angeles, Received 2006-09-11. URL(http://acert.ir.bbn.com/viewvc/gr-ucla/trunk/doc/usrp.pdf?revision=56)

[12] Pablo Brenner, *A Technical Tutorial on the IEEE 802.11 Protocol*, July 1, 1996 Internet. URL(http://www.sss-mag.com/pdf/802_11tut.pdf)

[13] R. Bill, C. Cap, M. Kofahl, and T. Mundt, *Indoor and outdoor positioning in mobile environments –a review and some investigations on WLAN-positioning*, University Rostock, Germany, 2005, Geographic Information Sciences, Vol. 10, No. 2, 2005. ISSN 1082-4006

[14] Wikipedia, *Barker code*, last visited 2006-11-15, Internet. URL(http://en.wikipedia.org/wiki/Barker_code)

[15] Wikipedia, *Complementary code keying*, last visited 2006-11-23, Internet. URL(http://en.wikipedia.org/wiki/Complementary_code_keying)

[16] Wikipedia, *Global Positioning System*, last visited 2006-10-31, Internet.
URL(http://en.wikipedia.org/wiki/GPS)

[17] Wikipedia, *Scrambler*, last visited 2006-11-25, Internet.
URL(http://en.wikipedia.org/wiki/Scrambler)

[18] Wikipedia, *Verilog*, last visited 2007-01-23, Internet.
URL(http://en.wikipedia.org/wiki/Verilog)

[19] Z. Xiang, S.Song, J.Chen, H.Wang, J.Huang, and X.Gao, "*A wireless LAN-based indoor positioning technology*", IBM Journal of Research and Development archive, Volume 48 , Issue 5/6, September/November 2004, Pages: 617 - 626, ISSN:0018-8646
URL(http://www.reserch.ibm.com/journal/rd/485/xiang.html)

Appendix A: MATLAB code for header analysis and location determination

A.1. Headeranalysis.m

```
clear;
clc;

Barker=[1 -1  1  1 -1  1  1  1 -1 -1 -1]'; % Barker sequence
SpreadingRate=length(Barker);             % Spreading rate = 11
%length=57552; %1Mbps %the total length, in the real case it should be read dynamicly
%length=32032;  %2Mbps myfile3
%length=11152;  %11Mbps myfile4
length=11000; %fix length to get 500 bits
long_preamble = 1; %1 for long preamble and 0 for short preamble
scrambler=1; %1 for on and 0 for off

if long_preamble   %Reversed, MSB first.
   SFD=[0 0 0 0 0 1 0 1 1 1 0 0 1 1 1 1];%F3A0 %SFD pattern, should be unique for DSSS PLCP
else
   SFD=[1 1 1 1 0 0 1 1 1 0 1 0 0 0 0 0];%reversed for short preamble
end

SCRSFD=[0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1];

SIGNAL=[0 1 0 1 0 0 0 0]; %MSB first. Sequence 00001010 for 1Mbps, 00010100 for 2Mbps

SCBLHD=[0 0 1 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 0 0 ...
     0 1 0 1 0 0 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 0 1 0 ...
     1 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 1 1 1 1 0 1 0 1 ...
     0 1 0 0 0 0 0 1 0 1 1 0 1 0 1 1 1 0 0 1 0 0 0 1 ...
     1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 0 0 0 0 1 1 0 ...
     1 0 0 1 1 0 1 1 0];

fid = fopen('myfile2.dat');
content = fscanf(fid,'F2I, %i\n',[2 length]); %500 bits
fclose(fid);

if max(content(1,:))>=max(content(2,:))
   PC = max(content(1,:)); %highest power. constraint 1
else
   PC = max(content(2,:));
end

I = content(1,:)/PC;
Q = content(2,:)/PC;
```

```
%Perform the downsampling
M=2;
I = I(1:M:length);
Q = Q(1:M:length);

%Make complex array
Rx = I + Q*i;

%Despreading, cause the preamble is also spreaded with barker code
Rx_symbols=Barker'*reshape(Rx(1:length/2),SpreadingRate,length/22);

%DBPSK Demodulation
Rx_Demo = dpskdemod(Rx_symbols, 2);

%Different approch of detecting Preamble&SFD
if scrambler==1
   SFD_sync=SCRSFD*2-1;
   Rx_Demo_sync=Rx_Demo(1:300)*2-1;
   [temp,Rx_sync]=max(xcorr(SFD_sync,Rx_Demo_sync));
   Rx_sync=300-Rx_sync; % 300 samples, circlic convelution
else
   SFD_sync=SFD*2-1;
   Rx_Demo_sync=Rx_Demo(1:300)*2-1;
   [temp,Rx_sync]=max(xcorr(SFD_sync,Rx_Demo_sync));
   Rx_sync=300-Rx_sync;
end

%preamble_length=129+16+8+8+16+16; %The first two bits are compensated by Rx_sync
                                  %cuz it finds the real start
preamble_length=Rx_sync+16; %This part is PLCP preamble.

ifSignal=Rx_Demo(preamble_length+1:preamble_length+9);
TR=0;
Rate=binarray2hex(ifSignal,'LSB');
if Rate(2)==0;
   disp('1 Mbps')
   TR=1;
elseif Rate(2)==1
   disp('2 Mbps')
   TR=2;
elseif Rate(2)==3
   disp('5.5 Mbps')
   TR=3;
elseif Rate(2)==6
   disp('11 Mbps')
   TR=4;
else
   disp('Unknown rate...')
end

if TR==1;
   Rx_PSDU_Demo=dpskdemod(Rx_symbols(preamble_length+49:500), 2);
   Rx_Data=binarray2hex(Rx_PSDU_Demo,'LSB');
```

```
    dec2hex(Rx_Data)
elseif TR==2;
    Rx_PSDU_Demo=dpskdemod(Rx_symbols(preamble_length+49:length/22), 4);
    Rx_Data=octarray2hex(Rx_PSDU_Demo);
    dec2hex(Rx_Data)
else
    disp('Higher transmitt rate not finished yet...')
end
```

A.2. Realcase.m

```
function crosses=realcase(SR)

%Sampling rate
SR=64*10^6;

%Time between 2 samples
TS=1/SR;

%Light speed
c=3*10^8;

%user and AP position
user=0;

%Coordinates of the APs
x1=-25;
y1=25;
x2=30;
y2=30;
x3=-5;
y3=-40;

AP1=x1+y1*i;
AP2=x2+y2*i;
AP3=x3+y3*i;

%Distances
dis1=abs(AP1);
dis2=abs(AP2);
dis3=abs(AP3);

%Real time it takes for signal to arrive at three APs
time1=dis1/c;
time2=dis2/c;
time3=dis3/c;

%Time error regarding sampling rate
```

```
err=rand;
temp=rand;
if temp>0.5
   err=err*1;
end
Terr1=TS*err/2;

err=rand;
temp=rand;
if temp>0.5
   err=err*1;
end
Terr2=TS*err/2;

err=rand;
temp=rand;
if temp>0.5
   err=err*1;
end
Terr3=TS*err/2;

%Receive time regarding time error. Here we don't care about processing
%time because it should be deterministic.
RecT1=time1+Terr1;
RecT2=time2+Terr2;
RecT3=time3+Terr3;

%Distance estimation
IntvT1=[RecT1+TS/2 RecT1-TS/2];
IntvT2=[RecT2+TS/2 RecT2-TS/2];
IntvT3=[RecT3+TS/2 RecT3-TS/2];

%Difference in arrival time + possible error
DiffT1=RecT1-RecT2+TS/2;
DiffT2=RecT1-RecT3+TS/2;
DiffT3=RecT2-RecT3+TS/2;

%Difference in arrival time - possible error
DiffT11=RecT1-RecT2-TS/2;
DiffT22=RecT1-RecT3-TS/2;
DiffT33=RecT2-RecT3-TS/2;

%Cross points
crosses=zeros(1,6);

syms x y;

fd12=sqrt((x-x1)^2+(y-y1)^2)+sqrt((x-x2)^2+(y-y2)^2)-abs(AP1-AP2);
fd23=sqrt((x-x2)^2+(y-y2)^2)+sqrt((x-x3)^2+(y-y3)^2)-abs(AP2-AP3);
fd13=sqrt((x-x1)^2+(y-y1)^2)+sqrt((x-x3)^2+(y-y3)^2)-abs(AP1-AP3);

%Solve equations to find cross points
f1=sqrt((x-x1)^2+(y-y1)^2)-sqrt((x-x3)^2+(y-y3)^2)-DiffT22*c;
```

```
f2=sqrt((x-x2)^2+(y-y2)^2)-sqrt((x-x3)^2+(y-y3)^2)-DiffT33*c;
sol=solve(f1,f2);
crosses(1)=double(sol.x)+double(sol.y)*i;

f1=sqrt((x-x1)^2+(y-y1)^2)-sqrt((x-x2)^2+(y-y2)^2)-DiffT1*c;
f2=sqrt((x-x2)^2+(y-y2)^2)-sqrt((x-x3)^2+(y-y3)^2)-DiffT33*c;
sol=solve(f1,f2);
crosses(2)=double(sol.x)+double(sol.y)*i;

f1=sqrt((x-x1)^2+(y-y1)^2)-sqrt((x-x2)^2+(y-y2)^2)-DiffT1*c;
f2=sqrt((x-x1)^2+(y-y1)^2)-sqrt((x-x3)^2+(y-y3)^2)-DiffT2*c;
sol=solve(f1,f2);
crosses(3)=double(sol.x)+double(sol.y)*i;

f1=-sqrt((x-x3)^2+(y-y3)^2)+sqrt((x-x2)^2+(y-y2)^2)-DiffT3*c;
f2=sqrt((x-x1)^2+(y-y1)^2)-sqrt((x-x3)^2+(y-y3)^2)-DiffT2*c;
sol=solve(f1,f2);
crosses(4)=double(sol.x)+double(sol.y)*i;

f1=-sqrt((x-x3)^2+(y-y3)^2)+sqrt((x-x2)^2+(y-y2)^2)-DiffT3*c;
f2=sqrt((x-x1)^2+(y-y1)^2)-sqrt((x-30)^2+(y-30)^2)-DiffT11*c;
sol=solve(f1,f2);
crosses(5)=double(sol.x)+double(sol.y)*i;

f1=sqrt((x-x1)^2+(y-y1)^2)-sqrt((x-x3)^2+(y-y3)^2)-DiffT22*c;
f2=sqrt((x-x1)^2+(y-y1)^2)-sqrt((x-x2)^2+(y-y2)^2)-DiffT11*c;
sol=solve(f1,f2);
crosses(6)=double(sol.x)+double(sol.y)*i;

crosses=sortarg(crosses);
```

A.3. Sortarg.m

```
function res=sortarg(input)
%sort an complex array by its argument
n=length(input);
args=zeros(1,n);
for i=1:n
   args(i)=angle(input(i));
end

[res,IX] = sort(args);
for j=1:n
   res(j) = input(IX(j));
end
```

A.4. Realsim.m

```
clc;
clear;

SR=64*10^6;

%user and AP position
user=0.2+0.2i;
AP1=-25+25i;
AP2=30+30i;
AP3=-5-40i;

r=10;
res=zeros(r,6);

for i=1:r
   res(i,:)=realcase(SR);
   i
end

resplot=zeros(1,6);

for i=1:6
   temp=sortabs(res(:,i));
   resplot(1,i)=temp(1);
end

%Plots
plot(user,'.r');
axis('equal');
axis([-50  50  -50  50]);
hold on;

plot([resplot resplot(1)],'g');

plot(AP1,'ob');
plot(AP2,'ob');
plot(AP3,'ob');

plot([AP1 AP2]);
plot([AP1 AP3]);
plot([AP2 AP3]);

%Text
text(real(AP1)-15,imag(AP1)+5,'AP1');
text(real(AP2)-15,imag(AP2)+5,'AP2');
text(real(AP3)-20,imag(AP3)+5,'AP3');

Merr1=num2str(max(abs(resplot)));
Merr2=num2str(min(abs(resplot)));
```

```
text(-45,45,['DerrMax: ' Merr1 'm']);
text(-45,40,['DerrMin: ' Merr2 'm']);

hold off;
```

Appendix B: Verilog code for time stamping and correlation

B.1. Correlation module

```
module correlation(mclk,bitin,reset,hw);
input mclk;
input bitin;
input reset;
output [10:0] hw;

/*  The following are macro definitions that are processed using M4.
Be sure to run this file through M4 before feeding it to Quartus!

The first line is a define, that controls if this verilog source will
be used by icarus verilog, or by the Quartus tool.  If it is for the
Quartus tool, then comment out the definition statement using slashes
and at least one # character.  M4 treats anything following a # character
as a comment, while verilog like 'C' style comments.  Use both, ie:

//##this is commented out.

The reason we need to have the definition ICARUS is to be able to handle
line continuation character differences between icarus verilog and Quartus.
Quartus wants backslashes as line continuation character.  Icarus doesn't
want any line continuation characters.  If you are going to run this source
through the icarus verilog tool, then you will need a definition for ICARUS.
If you are going to run this source through Quartus, then be sure that ICARUS
has not been defined.
###################################################################### */

define(`ICARUS')

define(`forloop',
        `pushdef(`$1', `$2')_forloop(`$1', `$2', `$3', `$4')popdef(`$1')')
define(`_forloop',
        `$4`'ifelse($1, `$3', ,
                                `define(`$1', incr($1))_forloop(`$1', `$2', `$3', `$4')')')

/* ####################################################################
End of M4 macro definitions

*/

reg [999:0] indata;
wire [999:0] pattern;
wire [999:0] corrout;
```

```
assign pattern[999]=1;
… //Here assign the pattern bits.
assign pattern[0]=1;

forloop(`i', 0, 999, `assign corrout[i] = (~(indata[i]^pattern[i]));
')


/*This register could be the correlation output.
  Use it as the data that will have the hamming weight computed. */

/* Compute the hamming weight. */

ifdef(`ICARUS',
`assign hw = corrout[0]
forloop(`i', 0, 248, `forloop(`j', 1, 4, ` + corrout[eval((i*4)+j)]')
') + corrout[997] + corrout[998] + corrout[999];',

dnl else for Quartus:

`assign hw = corrout[0] /
forloop(`i', 0, 248, `forloop(`j', 1, 4, ` + corrout[eval((i*4)+j)]') /
') + corrout[997] + corrout[998] + corrout[999];')


// Fake some serial input data

always @(posedge mclk)
begin
  if (reset) indata <= 11'b00000000000;  //sync reset
  else indata <= {indata[998:0], bitin};
end


endmodule
```

B.2. Testbench to simulate

```
module testbench();
reg             clk;
reg             [12:0] vectornum;
reg             [12:0] numerrors;
reg             [12:0] counter;
reg             [12:0] test;
reg [11:0] tmstamp;
reg             mclk;
reg             bitin;
reg             reset;
```

```
wire            [10:0] hw;

// Instantiate the device under test

correlation dut(mclk,bitin,reset,hw);

//receive data
//assign indata for simulation.
wire [1499:0]indata;

assign indata[0]=1;
...

// initialize things

initial
  begin
   vectornum = 0;
   numerrors = 0;
   counter=0;
   test=0;
   tmstamp = 0;
  end

// Generate a clk to sequence tests.  The #50 are the relative amounts of
// time the clock is high or low.  In this case, the duty cycle is 50%.

always
  begin
   clk = 0; #50; clk = 1; #50;
  end

// On each clock edge, apply next test

always @(posedge clk)
  begin
  tmstamp <= tmstamp + 1;

  if (vectornum == 0) reset = 1'b1;  //activate reset on 0th test vector
   else reset = 1'b0;

  if(hw<950) begin  //We find the peak here, this is just a test value.
  bitin=indata[counter];
  //$display("The correlation at index %d is %d :",counter,hw);
        end

  else if(counter>0)begin
  $display("The correlation peak found at index %d, with value %d",counter,hw);
  //Display a 8 bits timestamp
  //$display("Tmstamp
bit %d %d %d %d %d %d %d %d.",tmstamp[7],tmstamp[6],tmstamp[5],tmstamp[4],tmstamp[3],t
mstamp[2],tmstamp[1],tmstamp[0]);
  $display("Tmstamp is: %d",tmstamp);
```

```
//I should end the test here since I only need one peak,but don't know how. In real case this never ends.
     end
 counter=counter+1;
 mclk = 1'b1;
 end

// print out results when clk is low

always @(negedge clk)
 begin
 mclk = 1'b0;
 vectornum = vectornum + 1;
 end

endmodule
```

Appendix C: List of acronyms

ADC	Analog to Digital Converter
A-GPS	Assisted GPS
AOA	Angle Of Arrival
AP	Access Point
ASIC	Application Specific Integrated Circuit
CCK	Complementary Code Keying
COO	Cell-of-Origin
DBPSK	Differential Binary Phase Shift Keying
DGPS	Differential Global Positioning System
DQPSK	Differential Quadrature Phase Shift Keying
DSP	Digital Signal Processor
DSSS	Direct Sequence Spread Spectrum
E-OTD	Enhanced Observed Time Difference
FPGA	Field Programmable Gate Array
GLONASS	GLObal NAvigation Satellite System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HDL	Hardware Description Language
ISM band	Industrial, Medical, and Instrumentation band
LUT	Lookup table
NTP	Network Time Protocol
OFDM	Orthogonal Frequency Division Multiplexing
OTDOA	Observed Time Differ–ence of Arrival
PLCP	Physical Layer Convergence Procedure
PPDU	PLCP Protocol Data Unit

PSDU	Physical Layer Service Data Unit
RSS	Received Signal Strength
RSSI	Received Signal Strength Indicator
SBAS	Satellite Based Augmentation System
SFD	Start Frame Delimiter
TOA	Time of Arrival
TDOA	Time Difference Of Arrival
USRP	Universal Software Radio Peripheral
WLAN	Wireless Local Area Networks

COS/CCS 2007-14