Indoor Location Detection using WLAN

ANQI LUO
and
LEI GE

KTH Information and Communication Technology

Master of Science Thesis
Stockholm, Sweden 2009

TRITA-ICT-EX-2009:209

Indoor Location Detection using WLAN

Anqi Luo and Lei Ge
anqi@kth.se
leig@kth.se

Masters thesis

11 January 2010

Examiner and Academic Supervisor
Prof. Gerald Q. Maguire Jr.

Department of Communication Systems
School of Information and Communication Technology
Royal Institute of Technology
Stockholm, Sweden

Abstract

The thesis seeks to improve the accuracy of indoor wireless local area network (WLAN) location detection. The main task of the project is the design and analysis of a solution, which utilizes the packets which are already emitted by WLAN devices. The system consists of a signal receiver and signal processing. The positioning system does not transmit, thus the detection is completely passive. The result of measurements on received frames is used to calculate the WLAN transmitter's position. It does not require any transmissions, thus the detection is completely passive. The result of the measurements can be used to calculate the WLAN transmitter's position.

Location systems are more and more viewed as a necessary element of a WLAN system. Positioning accuracy is the most important issue in location system, especial in the indoor WLAN location detection. Indoor location systems are affected by indoor environment both due to multi-path and man-made effects. To resist these problems, we introduce a way to detect the arrival of the first instance of the signal by detecting the signal header. In our experiment, we timestamp the arrival of an IEEE 802.11b header. In our prototype the logic necessary to do this is implemented in an FPGA, specifically that of the Ettus Research USRP. The additional logic is quite small and might easily be added to the receiver in an access point, thus reducing the deployment cost of a location system in a real network.

The proposed solution was experimentally verified. From our experiments, the detection works without requiring any changes to the hardware or software of the mobile device. By exploiting existing IEEE 802.11b transmissions the cost and difficulty of deployment is simplified due to the wide usage of IEEE 802.11b in mobile devices. Additionally, the preamble has good correlation properties making it is easy to detect the arrival of a IEEE 802.11frame. Our implementation is based upon open source hardware and software making it possible to implement this solution. A relatively low cost FPGA can be used as the correlation and timestamp circuit is rather simple (in terms of numbers of gates), making this solution feasible for commercial implementation. The method, implementation, testing, and analysis are presented in detail in the thesis.

Key Words: WLAN, location detection accuracy, signal processing

Sammanfattning

Avhandlingen syftar till att förbättra noggrannheten i inomhus trådlösa lokala nätverk (WLAN) placering upptäckt. Huvuduppgiften för projektet är design och analys av en lösning som utnyttjar paketen som redan avges av WLAN-enheter. Systemet består av en signal mottagare och signalbehandling. Det kräver inga transmissioner, alltså upptäckt är helt passiv. Resultatet av mätningarna kan användas för att beräkna WLAN-sändarens läge.

Målet är att förbättra noggrannheten i inomhus plats uppskattning. Läge system alltmer ses som en nödvändig del av WLAN system. Positioneringsnoggrannheten ses som den viktigaste frågan i läge system, speciellt för inomhusbruk WLAN baserade location. Läge system påverkas mer av inomhusmiljöer än utemiljön, eftersom det finns mer multi-path fading och konstgjorda effekter. Att minska dessa problem, vi införa ett sätt att känna av signalen ankomst genom att förbättra upptäckten av ankomsten av IEEE 802.11-huvudet. Detta kan bidra till att besegra multipath effekt och enkla metoden skulle kunna minska kostnaderna för placering i framtiden kopplingspunkter.

Den föreslagna lösningen har verifierats experimentellt. Från vårt experiment fungerar upptäckt utan att kräva några ändringar i hårdvara eller mjukvara för den mobila enheten. Genom att utnyttja befintliga IEEE 802.11b sändningar kostnaden och svårigheten att utbyggnaden är förenklad på grund av den breda användningen av IEEE 802.11b i mobila enheter. Dessutom "preamble" har god korrelation egenskaper som gör det lätt att upptäcka ankomsten av en IEEE 802.11-ramen. Vår genomfört bygger på öppen källkod maskin-och programvara som gör det möjligt att genomföra denna lösning. En relativt låg kostnad FPGA kan användas som korrelation och tidstämpel kretsen är ganska enkel (i termer av antalet logikelement), vilket gör denna lösning vara möjlig för kommersiell tillämpning. Metoden, implementation, testning och analys presenteras i detalj i avhandlingen.

Table of contents

List of Figures

List of Tables

Acronyms and abbreviations

ADC	Analog to digital converter
AOA	Angle of Arrival
BPSK	Binary phase shift keying
CSM	Carrier sense multiple access
CSMA/CD	Carrier sense multiple access with collision detection
CSMA/CA	Carrier sense multiple access with collision avoidance
DAC	Digital to analog converter
DDC	Digital down converters
DQPSK	Differential quadrature phase-shift keying
DSP	Digital signal processor
DSSS	Direct Sequence Spread Spectrum
DUC	Digital up converters
EIRP	Equivalent isotropically radiated power
FPGA	Field programmable gate array
LAN	Local area network
LOS	Line-of-sight
NLOS	Non-line-of-sight
PDU	Protocol data unit
PLCP	Physical layer convergence procedure
PN	Pseudo noise
PPDU	PLCP Protocol Data Unit
RSS	Received Signal Strength
SDR	Software defined radio
SNR	Signal to noise ratio
TOA	Time of arrival
TDOA	Time difference of arrival
USRP	Universal Software Radio Peripheral
WAAS	Wide Area Augmentation System
WLAN	Wireless local area network

Chapter 1 - Introduction

With the widespread adoption of cellular phones and personal mobile terminals, additional information in addition to the (possibly multimedia) content is desired. Today one of the most desired additional elements of information is the location of the transmitter, as this allows application to provide increasingly tailored content to the user. Many approaches to location detection have been developed & tested, and some have even been deployed and used. Because of the success of some of these applications, the ability to determine a terminal's location correctly and accurately has become more important than ever. The most widely used location determination systems today is the Global Positioning System (GPS). GPS is based on solving a set of simultaneous linear equations to calculate the position of a receiver based on receiving a coded signal – generally this signal is from a navigation satellite system. GPS was developed by U.S. Air Force Central Inertial Guidance Test Facility. The technique detects the time of a signal's arrival from each of several GPS satellites, then calculates the distance between the user and the known positions of these satellites. In this manner the coordinates of the receiver can be determined. This technology usually gives high accuracy, but at high cost (both for the satellite infrastructure and in power consumed by the receiver).

Another technique, first applied in 1999, is named 'Wide Area Augmentation System' (WAAS). This also utilizes the GPS signal, but can be used to reach an location accuracy of ± 1-3 meters. WAAS has been running since 1999 with almost without stop. However, WAAS requires the receiver to be able to receive positioning information data sent via geostationary satellites; hence WAAS is mostly applicable in open areas, such as at sea.[44]

Advanced GPS receivers can use carrier phase information, and other techniques (such as error correction measurements from nearby receivers located in fixed positions) to reach surveying accuracies of less than half a centimeter. However, such systems are generally limited to outdoor user or require the use of pseudolite transmitters (which act like satellites, but can be placed indoors or outdoors). A number of applications for location determination in indoor environments require an accuracy of a centimeter or several centimeters, due to the dimensions of most indoor spaces and more complicated usage scenarios. Unfortunately, reflection and refraction inside a building have a strong influence on the detected arrival time of a signal. For example, the distance that a signal traverses between the receiver and the transmitter can be doubled or even tripled when a reflection occurs, thus introducing a large error (unless there is some way to determine the time of the signal's first arrival).[2]

On the other hand, when looking into the requirements for positioning, the absolute coordinates are often not the desired result. Often a set of two or three numeric coordinates only provides useful information when these coordinates are mapped to a specific place, such as a shopping mall, a hospital, a school, the location of a specific door, etc. In this case, a large database to map numeric coordinates to civil or other useful labels is very useful. Today such databases have become quite powerful as more and more users add geo-location information to their web pages. In addition, vehicles increasingly have GPS navigation systems (often with additional sensors being used to

increase the system's accuracy and to avoid the impairments caused by trees, buildings, … blocking and/or refracting the signals from the satellites.

For indoor environments, a user may want to ask questions such as: which room am I in, how do I get to this location, how I can go to the conference room on the other corner of the floor, and so on. One approach to indoor location determination is to create maps of the signal strength for each location in the building and simply look up in this database a set of possible locations, then based upon earlier (know or estimated) locations to estimate which location is the closest to the receiver's actual location. This method is often referred to as location fingerprinting (see section 2.3). Unfortunately this approach has several problems: you need to collect the data to create such a database and the movement of people and objects may change the propagation environment significantly (rendering the database either obsolete or impractical to update sufficiently often). Alternatively you can model the building structure and use a smaller number of measurements (as is done by Ekahau Inc.). Modeling can also be combined with probabilistic methods (see section 2.3).

In addition to GPS, many other methods exist to estimate a device's location. Most of these require that the device itself use information about signals it receives to determine its location (a so-called self-localization mechanism"). Two of these algorithms, Time-Of-Arrival (TOA) and Time-Difference-Of-Arrival (TDOA), rely on the detection and calculation of a difference in a signal's arrival time, which can provide an accuracy of centimeters in some settings. Another approach is to use the received signal strength (RSS) to estimate the receiver's location. In this method the difference in strength between the sent signal and the received signal is used to calculate the distance between the device and a set of access points (or base stations). Depending upon the equipment used, the accuracy of this method can be several meters. Yet another approach, is angle of arrival – where instead of looking at the time of arrival, directional antennas are used to calculate the angle of arrival – this can be applied either with data from multiple fixed receivers or with multiple antennas on the mobile device.

Lots of research has been done to apply the above techniques to real life location determination. In a stationary environment, these mechanisms can provide a reliable estimate of the device's location. However, radio propagation within an office environment will change due to moving people, removing/adding furniture, opening a door, and so on. These changes can significantly affect the estimation of a device's location. Just as in the case of GPS, there are a number of disadvantages for these existing mechanisms when applied to estimation of location indoors. A more reliable detection system suitable for an office environment is needed. Such a solution must properly deal with multipath effects, dead spots, noise, and interference in the indoor environment.

For a purely indoor environment, there also are various other technologies for location estimation, these include infrared, ultrasound, video surveillance, and other radio signals. Comparing these techniques, computations based on properties of the propagation of a radio signal are the most effective and economical, largely because of the existence of WLAN infrastructures in buildings today and wide area wireless networks which are almost ubiquitous (at least in urban areas). While (experimental and commercial) systems have used infrared, ultrasound, and video surveillance – these approaches are generally

too expensive to utilize for an entire building – as these solutions generally require installation of additional infrastructure. Today a large number of the devices that a person uses in their daily life (such as laptop computers, mobile phones, personal digital assistants (PDAs), and other products) are equipped with a WLAN interface. Thus the user already has the necessary physical device which can be located – avoiding any need to equip these people with special tags or other devices. The WLAN and cellular infrastructures have already been installed and are maintained for other purposes (i.e., mobile communication). In addition, the specific frequencies used by WLAN ensures that the radio signal experiences less intereference than ultrasound, infrared, and low frequency radio techniques. The large volume of WLAN equipment sold has resulting in WLAN equipment becoming available at commodity prices. In conclusion, WLAN, specifically IEEE802.11b, is an effective and economic source of signals for indoor location detection.[3]

In previous research on this topic, lots of other location detection systems were proposed. One of the main factors that have been focused on is accuracy, which is often viewed as a measure of the *quality* of the system. However, the requirement for accuracy depends upon the actual use to which the location information will be put. Generally, it is possible to get more accurate location information, but at a relative high cost, which could be undesirable in some services. Alternatively wide spread availability of a low accuracy system may be sufficient for some purposes, such as locating a cellular phone that is making an emergency call. In fact, one of the main driving requirements for determining the location of people has been for emergency services, such as the E911 requirement in the United States. In this case, the U.S. Federal Communications Commission required that cellular operators be able to locate a caller to the E911 service to with in 50 to 300 meters (depending upon the technology which is used) (see [20] and [21]). One method of doing this is to use the user's cellular phone as a simple location detection device, by using information about the transmission power of both base station and cell phone and the received signal strengths. This information is already being used in the cellular system for power control; however, this approach is not discussed further in our project (as WLAN interfaces generally do not implement such power control). In many emergencies, rapid location determination is more important that high accuracy in order to enable an effective rescue, thus placing low positioning delay ahead of high positioning accuracy. In many office settings the location accuracy simply has to be sufficient to indicate the correct room, desk, or doorway – while the location delay needs to be low enough that a user will not walk past the location before the relevant location can be determined. For example, if the user is trying to return a book to the library it is important that the alert come when the user is approaching the library and not after the user has passed the library [22].

1.1 Master Thesis goal

In our project, we try to enhance the accuracy when locating the user in an indoor environment. Compare with previous projects, we aim to locate an indoor user within 3 meters. Detailed problem statement will be discussed in Chapter 2.

1.2 Master Thesis Review

In this thesis report, we will discuss this topic in the following sections:

Chapter 1. Introduction of the topic

Chapter 2. Mainly introduce the background, which method have been used before in location detection and the definition of our problem.

Chapter 3. Briefly described the tools we used in the experiment, both hardware and software.

Chapter 4. Discuss the mechanism in our thesis project

Chapter 5. Evaluation and analysis of experiment result.

Chapter 6. Based on the result we will give the conclusion of this experiment.

Chapter 7. Recommendations for future research on indoor location detection will be presented.

Chapter 2 - Background

2.1 IEEE 802.11b

Before IEEE802.11b was popularized, wireless networks, especially wireless local area network had many drawbacks, such as low transmission rate and high cost. While many applications were design for Ethernet's 10Mb/s transmission rate, initially WLANs could only provide data rates of 1 to 2 Mb/s, which limited the widespread use of WLANs. Due to the growth in both data communications and multimedia, a new standard that could provide high transmission rates was a key to making WLAN popular. The first evolutionary step was IEEE802.11b[27]. Products based on this standard have become widely used, as it supports data rates comparable the 10Mbps Ethernet which many people were used to. WLAN enables people to move unrestricted within a given coverage area, often without interruption. Providing high data rates while enabling relatively smooth handoffs between access points has required changes in both the design and application of WLAN.

2.1.1 Introduction to IEEE802.11b

IEEE802.11b supports data rates up to 11Mbps, much higher than what IEEE 802.11 supports, increasing the variety of application that were feasible as compared to the previous IEEE 802.11 protocol. A number of different rates were defined, these can be used in different situations. These rates are: 11 Mbps, 5.5Mbps, 2Mbps, and 1Mbps. The actual user data throughput which users experience is usually around 5Mbps, similar to the actual throughput of an IEEE 802.3 10Base-T wired local network.

IEEE802.11b uses the 2.4GHz Industrial, Scientific, and Medical (ISM) frequency band, which does not require a license for the user – but does require that the manufacturers meet the requirements for wireless local area network devices for their products. IEEE 802.11b WLANs can be used as a supplement to LANs or as an independent network. Today most laptop computers have a built-in IEEE 802.11b WLAN interface in addition to a IEEE 802.3 1000 base T interface. This WLAN interface helps users to avoid having to carry and connect cables. Today, most corporate and academic sites provide (legitimate) users with WLAN connectivity.

While IEEE 802.3 (Ethernet) utilizes a carrier sense multiple access (CMSA) mechanism to control when packets are sent. Specifically, IEEE 802.3 employs Carrier Sense Multiple Access with Collision Detect (CSMA/CD). With this media access protocol all stations that wish to send messages listen for when the channel is idle. As only one station can be allowed to send at a time, the others need to wait until the channel is free. If two or more stations send messages at the same time, then a collision occurs, in this case all of the messages which were being set are lost and all of the stations attempting to transmit will stop trying to send and will wait for a random amount of time before listening to see if the channel is idle – at which time they will attempt to transmit their message again. Because in a wired network it is possible to determine if someone else is attempting to transmit at the same time as you are, this method works quite well. Unfortunately, this is not easy to do in the case of a WLAN interface, thus IEEE802.11b

uses another technique CSMA with collision avoid (CSMA-CA). Collision avoidance is necessary because the radio transmitter can not detect a collision (unlike the wired LAN case). To avoid collisions each interface that wishes to transmit waits for a random time after detecting that the channel is idle before attempting to transmit. This period of time can be divided into different ranges of time in order to enable a mix of higher priority traffic and lower priority traffic (this is the subject of other protocols in the IEEE 802.11 series of standards). For details of this see [23].

2.1.2 IEEE802.11b modes

There are two modes for IEEE802.11b: infrastructure mode and *ad hoc* mode. The later mode allows direct node to node communications and does not require any infrastructure. This mode is primarily used to create *ad hoc* networks and we will not consider it further in the context of this thesis as it is rarely used in campus settings. For further details and uses of this mode see the extensive literature on mobile *ad hoc* networks (often abbreviated as MANETs).

Infrastructure mode is more complicated as it introduces a special kind of node, called an access point (AP). Each AP acts as a bridge and interconnects the AP's cell to one or more network segments. The cell consists of all of the nodes which could communicate with this AP. APs typically are equipped with a wired IEEE 802.3 or other wired network interface – thus an AP can be used to extend an existing wired LAN to enable WLAN equipped devices to communicate with any of the devices attached to the wired LAN or internetworked to the wired LAN. When the wired LAN is connected to the Internet, the addition of an AP provides a wireless node with the possibility to access the Internet. Note that APs can perform various sorts of access control – thus limiting both the set of nodes which can connect via this AP and even controlling the characteristics of their traffic (for example, bounding the maximum rate, maximum traffic in some period of time, …)[24].

As the coverage of a single access point is limited, on a corporate or academic campus a multi-access point network is quite common. Each of these access points is connected to a backbone network (typically a high speed LAN based network). As we will make use of information from multiple receivers, these receivers could be co-located with access points or perhaps even integrated into each of the access points. Because these APs are connected to the fixed LAN, this means there is not a concern about having sufficient bandwidth between our receivers.

2.1.3 IEEE 802.11b frame format

An IEEE 802.11b frame is similar to the previous IEEE 802.11 frame, with respect to data structure, enabling compatibility with the previous standard. As we can see in Figure 1, the IEEE 802.11b data frame starts with a physical layer convergence procedure (PLCP) preamble - composed of two parts: a SYNC field and a start of frame delimiter (SFD) field. This PLCP preamble is followed by a header. This header consists of 48 bits, and includes a SIGNAL field, SERVICE field, LENGTH field, and CRC field.

In IEEE 802.11b, there are two types of PLCP Protocol Data Units (PPDUs), one has a preamble with the same length as defined in IEEE 802.11. This is called a Long PLCP PPDU, and the other one with a shorter preamble is called a Short PLCP PPDU.

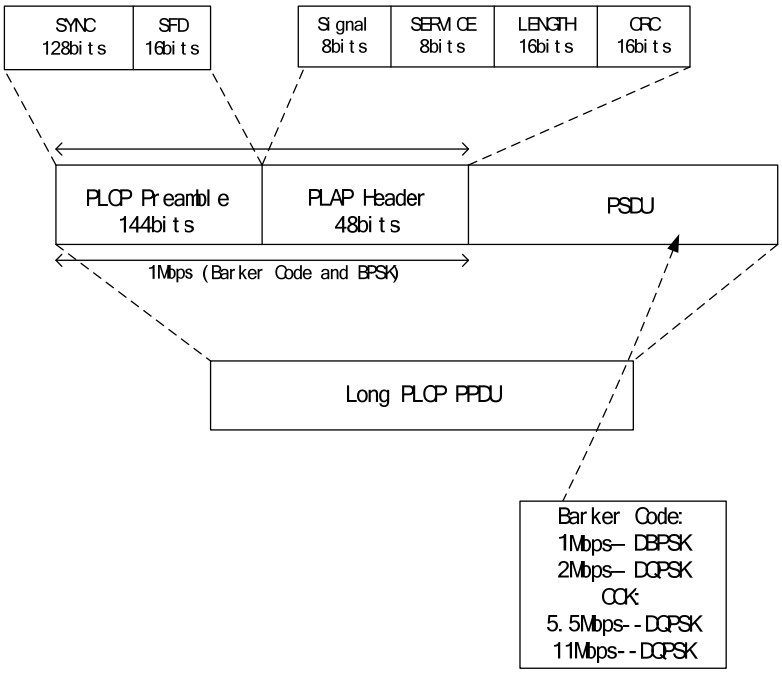


Figure 1: IEEE 802.11b Long PPDU

The preamble of a long PLCP PPDU as defined by IEEE 802.11b, shown in Figure 1, is 144 bits, which includes a 128 bits SYNC field and 16 bits SFD. The Start of Frame Delimiter (SFD) is defined as '0000 0101 1100 1111' for a long PLCP PPDU. Following this is the header (which contains Signal, Service, Length, and CRC) and the data carried. The header contains information indicating the transmission rate and frame length, protected by a CRC. This information can be used to decide how to decode the rest of the frame and how long the frame is expected to be. When a long PPDU is used, the preamble and header are the same that of IEEE 802.11. To provide backward compatibility with IEEE 802.11 the transmission rate used for this header is 1Mbps, and binary phase shift keying (BPSK) is used to modulate signal.

2.1.3.1 Short PLCP PPDU

Unlike the Long PPDU, the Short PLCP PPDU was designed to enhance throughput.

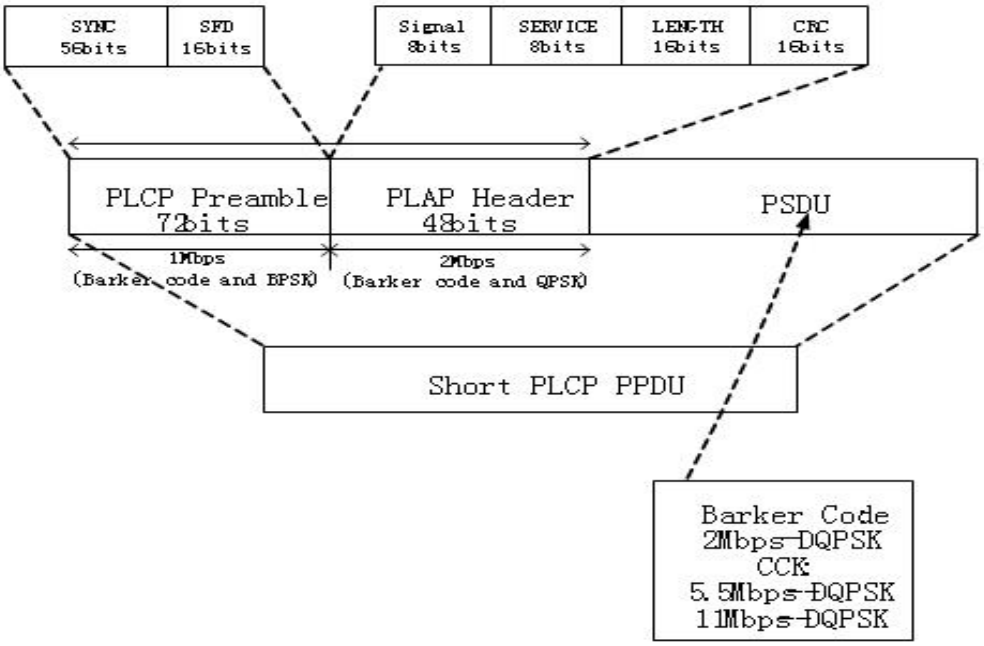


Figure 2: IEEE 802.11b Short PPDU

The difference between Long PPDU and Short PPDU are shown in Figure 2.

Comparing the two figures, it is easy to see that the length of the SYNC field is much shorter in the Short PPDU. In addition, the header is transmitted at higher rate (2Mbps) when differential quadrature phase shift keying (DQPSK) is applied. As a result, the information contained in preamble and header will be transmitted in a shorter period of time, i.e., half that of the Long PPDU. To distinguish from the Long PPDU, the SFD is defined as the binary string '1111 0011 1010 0000'.

According to the feature of Short PPDU, it is usually used in good channel condition, then all the actions, such as synchronization, channel estimation, can be finished within 56bits, as well as 2Mbps transmission rate would not cause demodulation error.

2.1.3.2 Preamble

Shown in Figure 1, the preamble is composed of a SYNC field and a SFD field.

SYNC Field — This field is primarily used to provide synchronization between a sender and receiver(s), and can be used to estimate channel condition.

SFD Field — The Start of Frame Delimiter (SFD) tells the receiver that the following data is the start of the header. After detecting the SFD, the receiver will

know that a data packet is being received.

2.1.3.3 Header

IEEE 802.11b header includes Signal, Service, Length and CRC parts, as mentioned above.

Signal Field — This field indicates what transmission rate is used for the PPDU.

Service Field — Designed to increase the flexibility of PPDU, for service future services.

Length Field — The length field of IEEE 802.11b informs the receiver of the time required to send the PPDU, but not the actual data length, which is different than the case for IEEE 802.11a.

CRC Field — The receiver can use the CRC to verify that the Signal, Service, and Length information is correct. It is used to protect the base band circuit from incorrect demodulation.

2.1.4 Applications

Generally, all the existing applications and network services based on wired LANs are now available via WLANs using the IEEE802.11b protocol. However, WLANs provide additional features. For example, in many historic sites, it is quite hard and expensive to run new wires or cables; additionally it is better to avoid cables in order to protect the historic building's appearance, construction, and so on. The most important feature of a wireless network is their flexibility. In many indoor scenarios, the network configuration must be changed frequently, which for a wired network would require an expensive (re-)deployment; where as in a wireless network there is little wiring (as only the access points are attached to the LAN) significantly reducing the installation cost. Furthermore, WLANs also support other desirable features, such as roaming. Roaming enables users to move between APs; while retaining the ability to communicate, despite attaching to a new subnet via a new AP.

2.1.5 Future of IEEE802.11b

IEEE 802.11b interfaces and access points have achieved a huge success in the market. Not only have such WLAN devices been deployed in campus settings, but they have also become widely used for applications such as production, stock control, and in retail shops. Due to a very large increase in cost performance IEEE 802.11b WLAN interfaces are increasingly incorporated into devices, as it is more cost effective to integrate this interface than add a connector to add it later. This is particularly noticeable for laptop computers, some computer peripherals, and even some wide area cellular phones (such as

Nokia's E60/E65/E70/… handsets). In addition to the application of IEEE 802.11b in the working place, WLANs are increasingly being deployed in people's homes as this avoids the need to run LAN cabling around the home and offers all the advantages of wireless connectivity (flexibility, movement, rapid installation, …).

Note that there are a number of new WLAN standards which have been introduced since 1999, these include IEEE 802.11a in the 5 GHz band, IEEE 802.11g (in the 2.4GHZ band), … most of which have been included in a single IEEE 802.11 document[26]. Today there is a proposed amendment, IEEE 802.11n (in the 5 GHz or 2.4GHz bands), to incorporate multiple-input-multiple-output (MIMO) techniques and other improvements – enabling date rates of 100Mbps or more. However, in this thesis we will not consider these alternatives and will focus strictly on IEEE 802.11b as this is the mostly widely used WLAN protocol on our campus.

2.2 Direct Sequence Spread Spectrum

Direct Sequence Spread Spectrum (DSSS) is a modulation technique [7]. DSSS differs from other modulation techniques in that the transmitted signal usually takes up much more bandwidth than the original signal. DSSS spreads the original signal with a spreading code at the transmitter. The receiver uses the same spreading code to recover the original signal. More specifically, a DSSS transmitter uses a pseudo noise (PN) code to modulate a carrier wave and at the receiving side, the receiver synchronizes with the exact phase of the sent PN code, the receiver locally reproduces this PN code (which has the same phase information as a local demodulation signal). This PN code is correlated with the received signal to reproduce the original signal.

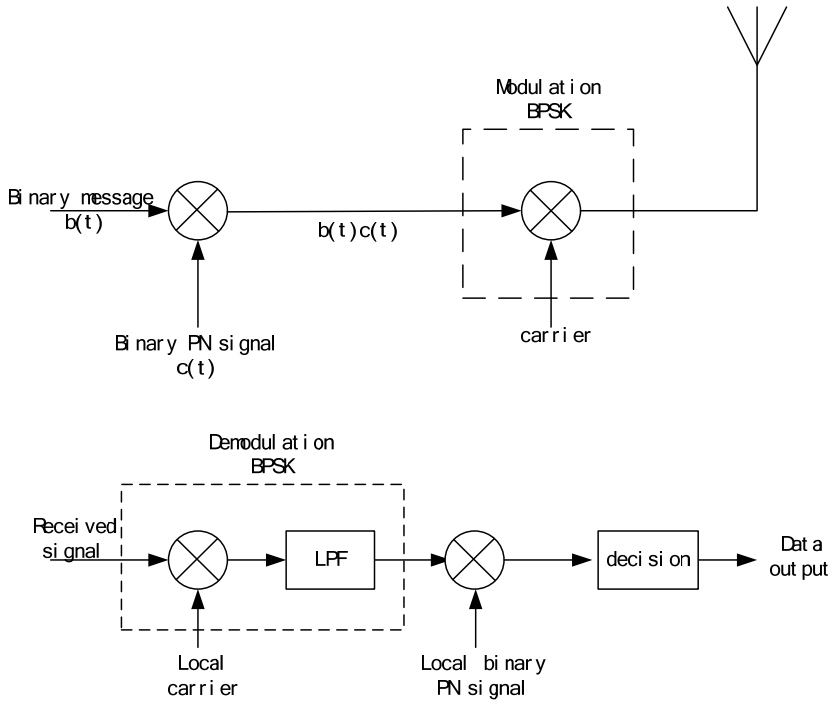


Figure 3: DSSS-BPSK system flow: sender and receiver

In the IEEE 802.11 standard, the PN code used for DSSS is called a Barker code. This is a string code with good autocorrelation. Figure 4 shows the autocorrelation of a Barker-11 code.

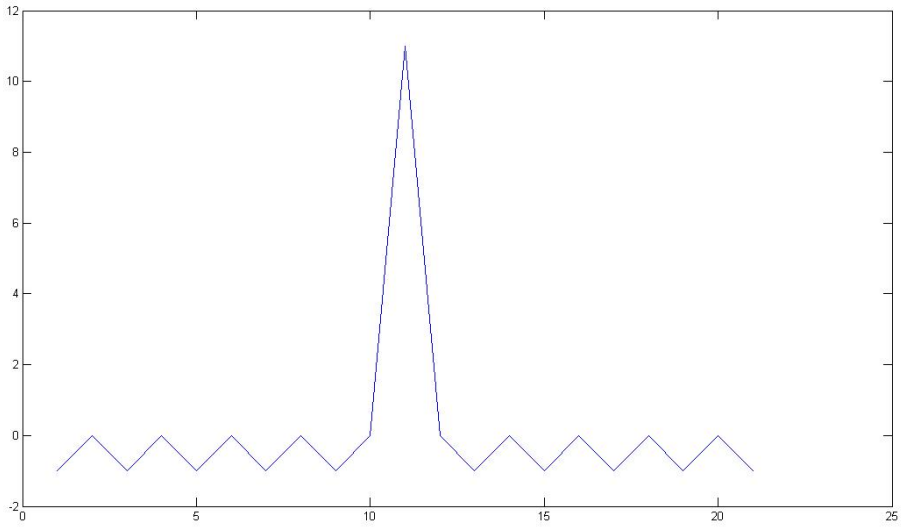


Figure 4. Autocorrelation of barker-11 code

The Barker-11 code is: {+1, -1,+1,+1,-1,+1,+1,+1,-1,-1,-1}. The autocorrelation is 11 and cross-correlation is -1. At the receiver, the same Barker code is used to correlate with the every received chip using a shift register. The result of this autocorrelation is compared against a predefined threshold to determine the original value being transmitted. The Barker code can also overcome the multipath effects, due to this good autocorrelation and cross-correlation properties.

DSSS has many advantages which have lead to it being used in many environments (particularly by the military, which exploited its resistance to jamming and with very long spreading codes it was possible to transmit with a very low probability of being detected). However, the use of DSSS in IEEE 802.11b is primarily due to a requirement by the U.S. Federal Communications Commission (FCC) that secondary use of the 902-928 MHz and 2.4 GHz ISM bands for WLAN required the use of spread spectrum techniques to avoid interference with the primary users of these bands. Thus the main property of DSSS which was of interest is that it spreads the energy of the original signal over a much wider band, thus reducing interference with the primary users. This also leads to a maximum transmitted power limitation specified as 1W equivalent isotropically radiated power (EIRP) in the US and 100mw EIRP in Europe. Additionally, the FCC mandated a listen before transmitting media access scheme (also designed to avoid interference with the primary users).

Rather than select a PN which is difficult to detect and difficult to exploit, the IEEE 802.11 working group selected a PN based upon a Gold code which facilitates the receiver detecting and synchronizing with the transmitter. Additionally, the same code is always used, unlike the situation for military communications where the PN sequence is frequently changed (often with an extremely long cycle length). We will see later that the use of a single code is a great advantage when we try to detect the arrival of a WLAN frame.

2.3 Introduction to the WLAN location detection

Since the IEEE 802.11 wireless local area network (WLAN) standard was published in 1997 (clarified in 1999 [25]), the WLAN market has grown swiftly. Today's high speed WLAN enables a user to connect to the Internet almost anywhere that people are regularly. As noted earlier, portable devices (for example, notebook computers, hand-held computers, and personal digital assistants) commonly have a WLAN interface. Given the very large numbers of devices with such a WLAN interface, there is a growing market for services and applications which can utilize location information – as provided by a location detection system in conjunction with users having a WLAN equipped device. This motivates the work undertaken in this thesis to utilize the WLAN signal *itself* to assist in accurately determining the WLAN transmitter's location.

In order to use WLAN as the basis for an indoor localization technology we would like to understand how to create a system with sufficient accuracy (in our case the target is ±1.5m) in our existing WLAN environment, without requiring a localization survey using special hardware. The following sections will examine some of the existing WLAN based positioning techniques and we will evaluate each of these in light of our requirements.

2.4 Existing WLAN location methods

Several different kinds of positioning systems exist for outdoor location determination, such as GPS, Assisted GPS, and a number of additional cellular network based systems, for example, TOA, TDOA, AOA, RSS, as well as multi-path pattern matching. Not surprisingly, many of the existing WLAN location technologies utilize these same methods. We will begin by introducing each of the techniques which might be used, then describe which are suitable for our purposes.

2.4.1 Time of Arrival (TOA)

The Time of Arrival (TOA) technique is based on measuring the difference in the arrival time, which for a direct path is the signal propagation time from a radio transmitter to one or more signal receivers. The difference in the arrival time at each receiver is directly related to the difference in path length over which the signal travels.[8] After synchronizing the transmitter and all the receivers, the difference in the propagation time (t) can be determined by recording the receive time of the signal over each path. Since the radio signal propagates at the same velocity over each of these paths (roughly $c = \sim 3 \times 10^8$ m/s) an equation in terms of the difference in propagation times can be solved to determine the location of the transmitter, as stated in equation 2-4-1. In this method the transmitter is located on a circle which is centered on the receiver's position and has a radius equal to c*t meters.

$$\sqrt{(x-x_i)^2+(y-y_i)^2} = c(t_i - t), i = a,b,c... \qquad (2\text{-}4\text{-}1)$$

If we assume that the transmission time from user to base station is *t*, then the distance between the user and the base station is *r*=*c***t*, the user should be located on the border of the circle with a center at the base station and radius of *r*. This is illustrated in Figure 7 When the signal is received by several base stations, the distance from the user to different stations can be determined, furthermore, the location of the target can be determined to be within the overlapping area of those circles. However, this is an ideal case, in which all the clocks are perfectly synchronized and there are no other influences.

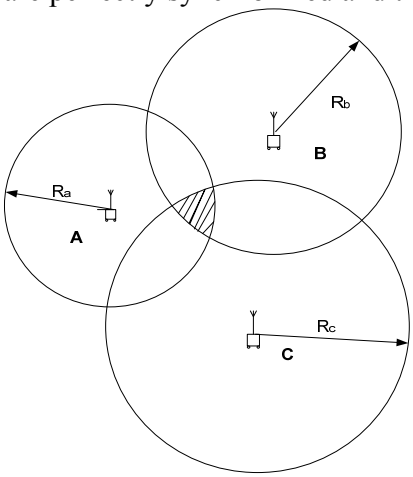


Figure 5: Ideal TOA model Three WLAN cells and the over lapping region(s) [8]

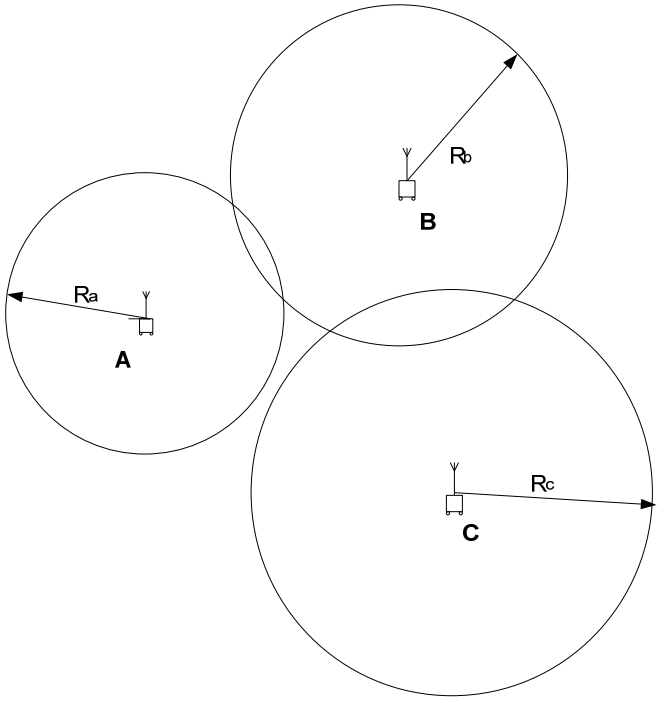


Figure 6: TOA with errors

If there are errors the result is shown in Figure 6. The difference from Figure 5 is readily apparent. For example, there might be no common areas of the circles. These errors could be due to multipath, refraction, and-or noise. Additionally, because the velocity of light is so high -- even a small time inaccuracy can lead to a large error. However, in some cases it is possible to compensate for these errors, at the cost of increased system cost and complexity. As a result, in a real scenario, TOA usually is not used alone.

When doing the calculation using data from multiple receivers, a more exact position can be determined by computing the intersection area of several circles. In practice cylinders would actually be used; as circles would represent a timing measurement with infinite resolution. In contrast, a cylinder represents the finite resolution of the timing measurement and can also include systematic errors in the measurement of the time, the error in the location of the receiver(s), etc. TOA is widely used in many location detection systems, including GPS.

Some of the advantages of this system are that it is rather straight forward to solve the set of equations to estimate the location of the device whose location is to be determined, the work is done by the receiver(s) (or by a third party who is given the received data), and the accuracy is roughly proportional to the resolution of the clock relative to the velocity of signal propagation. Note that TOA is frequently used indoors in conjunction with acoustic signals, such as ultrasonic pulses, as the velocity of propagation is much slower, hence the spatial resolution with a given frequency clock is greater than when optical or radio TOA solutions are used.

However, drawbacks exist as well. One of the distinct limitations is spatial accuracy. If the time resolution is 1μs, this translates to a ~300 meter positioning resolution. This is assuming that the transmitter and receivers are well synchronized. There are also problems with multi-path propagation of the signal – thus requiring some way to determine the first arrival of a signal, in order to reject the later arrival of reflected copies of the signal. Unfortunately, indoors there are quite often a lot of reflections from a large variety of objects and 300m resolution is not sufficient for many applications.

2.4.2 Time Difference of Arrival (TDOA)

Time difference of arrival (TDOA) is an algorithm based on TOA, which determines the position by measuring the time difference of signal arrival. This significantly decreases the requirement for time synchronization. This technique is used in a wide range of applications ranging from wireless communication to electronic warfare. Receivers are located at known fixed positions; the transmitter's position can then be determined by a hyperbolic function. The case for two receivers is illustrated in Figure 7. With additional receivers, the location can be computed using a set of hyperbolic functions, which ideally intersect in a unique point, as shown in Figure 8[10]. The exact transmit time is not need. However, in reality, the series of hyperbolic functions seldom intersect, due to various kinds of errors. Similar to TOA, a tiny error in the detected time leads to a large error in location. In this case, the positioning problem turns into an optimization problem. For a bounded space, such as an office environment or a lab scenario, an error of, for example 6 meters, which would be acceptable for outdoor

positioning system, would give a totally incorrect position, due to the limited volumes typical of an indoor space. As a result, the optimization problem is more difficult than for an open area. One the other hand, we can see that the limited area also means that the transmitter's position need only be located within a limited range, thus there is a small set of potentially valid solutions, which can simplify the optimization probelm to some extent.

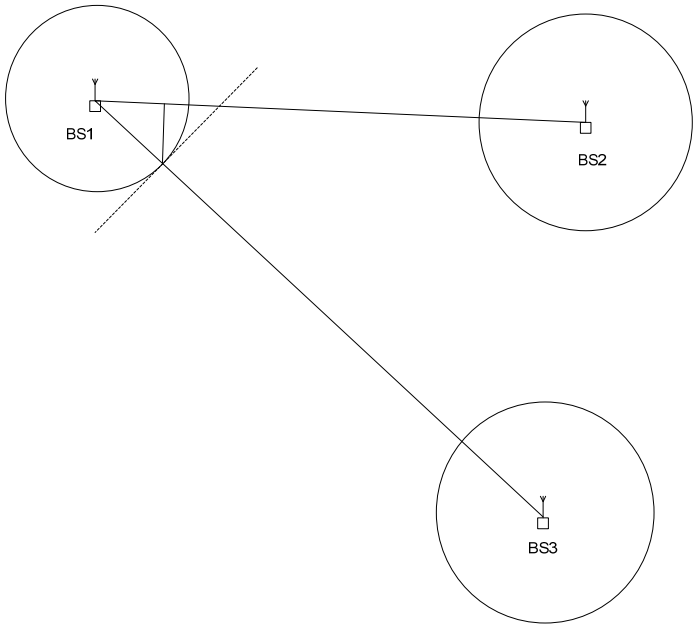


Figure 7: With two measurements

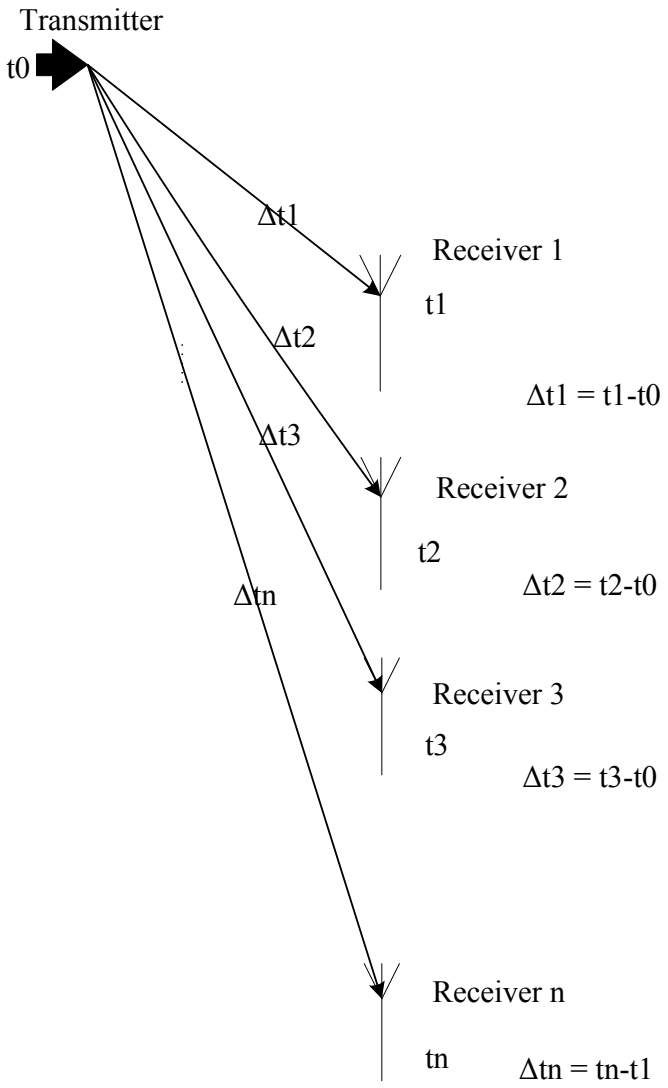


Figure 8: Mechanism of TDOA with multiple receivers

2.4.3 Angle of Arrival (AOA)

Angle of Arrival (AOA) is widely used in wide area cellular networks. AOA determines the transmitter's location by measuring the arriving angle of the detected signal. A special antenna array is needed to detect the angle of arriving signal, which indicates the direction to the target. It is possible to compute the target's position with a combination of two different antenna arrays, as showed in Figure 9. Ideally, the target user will be located where the lines cross. However, this ideal case requires a line-of-sight (LOS) transmission. Non-line-of-sight (NLOS) results in unpredictable errors, as multipath effect could contribute to the error. Addition, the accuracy is constrained by the resolution of the antennas as well [11] [33]. This mechanism is similar to TDOA, but measures the difference in angle rather than difference in time of arrival. Today the angle of arrival is calculated based on measuring the TDOA at a location using an antenna array. A set of AOA results can be used to locate the desired transmitter. [12]

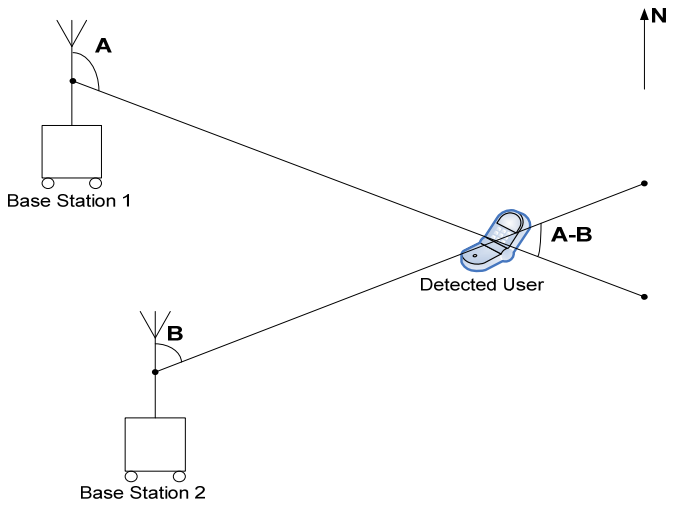


Figure 9: Mechanism of AOA [12]

AOA is a technique that requires a lot of infrastructure, thus it seems to be difficult to use this technique. Some research has been done to introduce phase shifters on each of the four feeds to resolve ambiguities and increase the beam steering accuracy to improve the AOA technique for a monopole antenna. This mechanism works in general, but there can be some error or misdetection when the signal arrives in a side-lobe. Another potential problem is that to utilize this technique widely requires extra equipment. See for example the Jordi Solsona's thesis [41], which makes use of phase array techniques and the movement of the target to provide improved accuracy.

2.4.4 Received Signal Strength

Received signal strength (RSS) methods are based upon the measurement of the magnitude of an electric field at a certain point. If the transmitter power at the transmitter's antenna is known, then by measuring the received power it is possible to estimate the distance of the receiver from the transmitting antenna. This technique requires an accurate model for the attenuation of the signal or a very large number of measurements (if there is no model). The estimated position can be improved by using estimates from multiple receivers.

In addition to problems such as reflection, the calculation is strongly dependent upon the transmitting frequency; as the attenuation at different frequencies may differ widely, even in the same environment. This complicates the computation and may also require measurements of the environment to be done with multiple radio frequencies (in the absence of a suitable model of the environment).

Despite these problems a number of researchers have used RSS signal measurements as part of their location estimation. See for example [14]. One of the advantages of RSS based systems is that a large number of WLAN receivers provide RSS measurements. Unfortunately, these measurements are not standardized – thus the measurements from different vendors (and even different models of receivers from the same vendor) may represent different received signal strengths with the same numeric value. However, it is possible to calibrate each receiver – at the cost of additional measurements and effort.

2.4.5 Location finger printing

Numerous researchers have used location finger printing to determine the location of a mobile device based upon matching the information from multiple receivers (or transmitters) with earlier measurements. Examples of these systems include Microsoft's RADAR system [45], KTH/SU DSV's SPOT system, and Cisco's Wireless Location Appliance [46]. For example, the Cisco wireless location appliance provides a real-time positioning service, that can be used to help a user to find the nearest shop (of some type) or to located the correct exit in an emergency situation.

2.5 Problem Statement

The purpose of this thesis project is to understand how to create a localization system with sufficient accuracy (in our case the target is ±1.5m indoors) exploiting the existing WLAN environment, without requiring a localization survey using special hardware. This thesis project builds upon several prior master thesis projects at the Department of Communication Systems at the Royal Institute of Technology (KTH).

The main idea underlying this project is to enhance the accuracy and the efficiency of location detection *using properties of the WLAN signal itself* rather than processing the decoded signal at the receiver. One of the reasons to process the RF or baseband WLAN signal or at least the chips themselves is that this data contains higher frequencies than the original data signal (i.e., than the decoded bits). The increase in frequency is at least proportional to the number of chips per bit (as this represents the results of the frequency expansion due to the use of DSSS).

Consider a system consisting of a transmitter and three receivers. The three receivers are assumed to be perfectly synchronized, i.e., they have synchronized clocks, and that each of the three receivers' locations is known very accurately. When the transmitter sends a signal (containing an IEEE 802.11b frame), the receivers could report the time while the signal arrived at each receiver. The location of the transmitter can be calculated by the three receivers' time stamped frame arrival times. Note that this approach avoids the need to synchronize the transmitter; in fact the receivers can do their computation based upon *any frames* which the transmitter sends. As a result it is possible to locate any device which is transmitting an IEEE 802.11 frame – *without any modification of the mobile devices themselves*. It should also be noted that the transmitter's MAC address is included in each frame which is transmitted, so that the receivers know which transmitter's frame arrival they have just time stamped. Additionally, this determination of the transmitter's MAC address can be done by the normal MAC processing of a WLAN receiver and this determination does not have a hard real-time constrain, unlike the recording of the arrival time of the frame.

This approach has been investigated in an earlier master's thesis project which used correlation of the waveform of the start of the frame (the IEEE 802.11b synchronization pattern) with the received signal [3]. This approach was first described in U.S. Patent 6,975,618 [5].

Chapter 3 - Toolkit

3.1 Software Radio

A software radio, unlike a hardware radio, utilizes software to decode a received radio signal (or to encode a signal to be transmitted). An advantage of a software radio is that potentially all the parameters of the radio can be altered by making changes in the software, rather than requiring any hardware changes. A software radio can also easily be reconfigured, thus in the limit only one universal hardware platform needs to be developed. Based on this underlying hardware and a suitable software platform, the system can be programmed to implement many different radios (perhaps even multiple radios simultaneously). An advantage of using such a single hardware platform is that a wide variety of different kinds of scenarios can easily be accommodated and the cost of designing, manufacturing, and maintaining the hardware can be reduced; as only one platform is needed, rather than a different hardware platform for each software radio type of radio.

Ideally a software radio supports a fully programmable choice of radio frequency (RF), bandwidth, channel access protocol, and channel modulation. Such an ideal software radio would consist of a computer, one or more high-speed analog to digital converters (ADCs) and digital to analog converters (DACs), and an antenna. Such a software radio places the executing code as close to the antenna(s) as possible. This converts what is generally viewed as a hardware problem into a software problem. Figure 10 shows an example of an ideal software radio receiver and Figure 11 shows the corresponding structure of a transmitter.

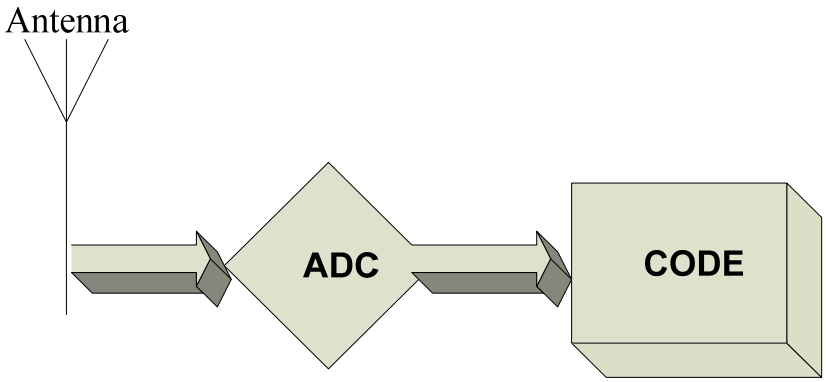


Figure 10: Ideal Software Radio block diagram (receiver)

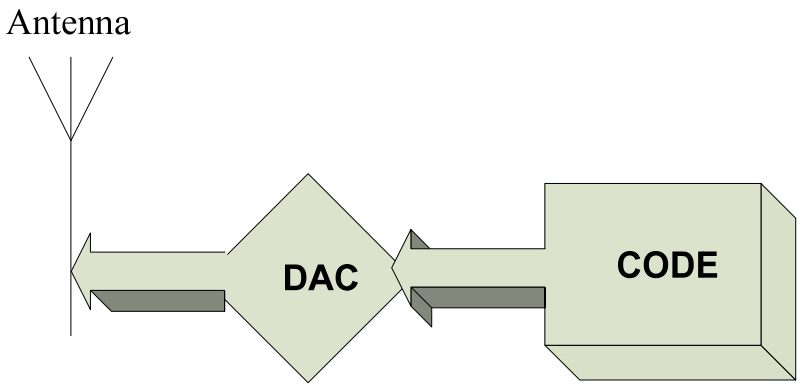


Figure 11: Ideal Software Radio block diagram (transmitter)

3.2 Software Defined Radio

However, currently there remain some barriers to realizing a fully programmable software radio. In this thesis we will make use of a software defined radio (SDR). In our case, the SDR will provide the capability to perform intermediate frequency (IF) programmable digital processing.

The basic SDR consists of a computer equipped with some low speed ADCs and DACs (such as sound card, USB microphone/headset, video camera, …) a general purpose processor which is used for performing more complex signal processing (labeled "SP in the figure below), and a special peripheral called a Universal Software Radio Peripheral (USRP). More information about the USRP will be given in section 3.4. The basic system is shown in Figure 12. Such a system can be used to receive and transmit different kinds of radio waveforms by running different software in the high speed digital signal processor (DSP), general purpose processor, or in the case of the USRP in a field programmable gate array (FPGA).

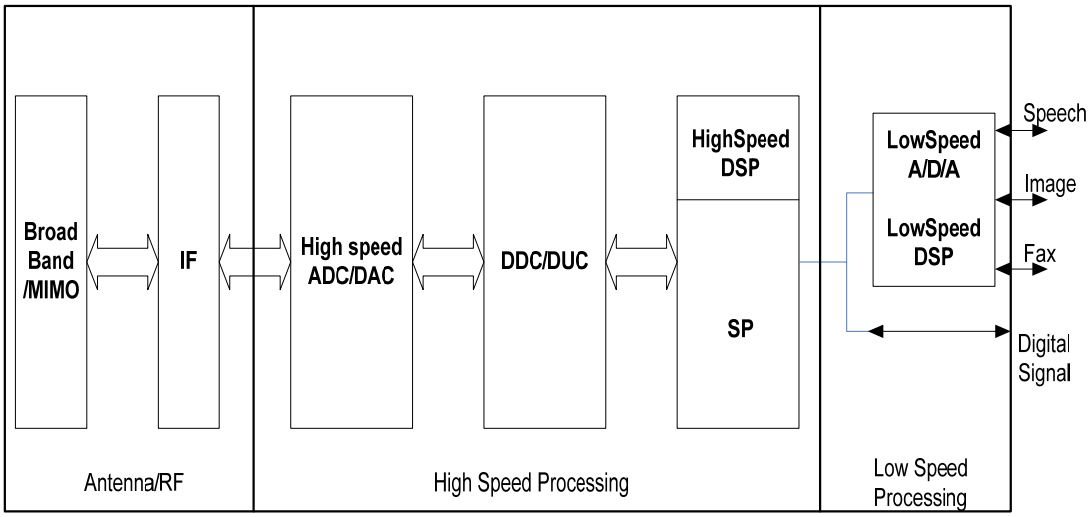


Figure 12: Basic Structure of our Software Defined Radio

Today, software radio is no longer a theoretical concept. There are numerous commercial implementations – both for civilian and military applications. This technology has a significant utility for military and cellular telephone services, as both settings need to serve a wide variety of different radio protocols in real-time.

The main advantage of software radio is that it requires only reconfiguration and possibly new software, but not hardware to support new radios. This decreases risk when developing a new radio product (or series of products), as the detailed functions can be changed at run-time – rather than being frozen in at design time. Additionally, SDR has decreased the effort needed to design and manufacture base stations for multiple standards; as Vanu, Inc. has demonstrated with their all software GSM base station [28] and in their multiple protocol base station demonstrations [29].

SDR also enabled the development of cognitive radio, which is an innovation in wireless communication aiming to allow SDRs to determine what kind of radio they should be to accomplish the goals of their user [30]. Cognitive radio is an important element in dynamic spectrum allocation - as the frequency assignment for a given device and service can be changed dynamically.

We will use an existing SDR in this thesis in order to conveniently create a radio receiver that can both receive the IEEE 802.11b WLAN frames and compute very high resolution timestamps for the arrival of these frames.

3.2.1 Analog to Digital Converter (ADC)

To understand the function of the USRP SDR, we must first examine an analog to digital converter (ADC). An ADC converts an input analog signal, such as a voltage or current, into a digital signal. Subsequently, this digital signal can be manipulated by a digital device. The use of an ADC reduces the need to do analog signal processing at the cost increasing the amount of digital data that the receiver must process.

Resolution is a crucial factor that indicates the number of discrete value into which the ADC quantizes the analog input. The resolution is usually measured in ‘bits’ and the discrete value are referred to as ‘levels’. For example, an ADC with a resolution of 8 bits can quantize the input into 256 levels, since $2^8 = 256$. Generally, an ADC with higher resolution can more accurately quantize the signal, but it has a higher cost (for a given sampling rate). Accuracy mostly depends on the quantization error and the aperture error. The former is mainly determined by the finite resolution of the ADC converter, which is unavoidable in all types of ADCs (and also digital to analog converters (DACs)). The resolution and cost can usually be traded-off against each other.

3.2.2 RF Front End

According to Nyquist, in order to avoid aliasing when sampling a signal the minimum sampling rate must be at least $2*f_{max}$, where f_{max} is the maximum frequency of the interested signal. For example, for a low pass signal whose bandwidth ranges from 0 to f_1, the sampling rate should be larger than $2* f_1$, which establishes the sampling frequency of the ADC. A similar analysis can be done concerning the output of a signal by a DAC. Unfortunately, it is expensive to have an ADC/DAC with a sampling rate of the RF signal once the frequency exceeds several hundreds of megahertz. To handle higher frequency

RF signals a RF front end is used to shift the receiving frequency band into an intermediate frequency band suitable for use with the ADC that is to be used. For example, the entire 2.4GHz ISM band of 80MHz bandwidth could be shifted to a DC to 80 MHz signal. If it is possible to sample at 160 MHz, then the entire ISM band could be digitized and processed digitally.

Figure 13 shows a baseband front end for the USRP. Figure 14 shows a 2.4GHz front end (specifically the RFX2400) for the USRP. Details of these front ends will be given later.

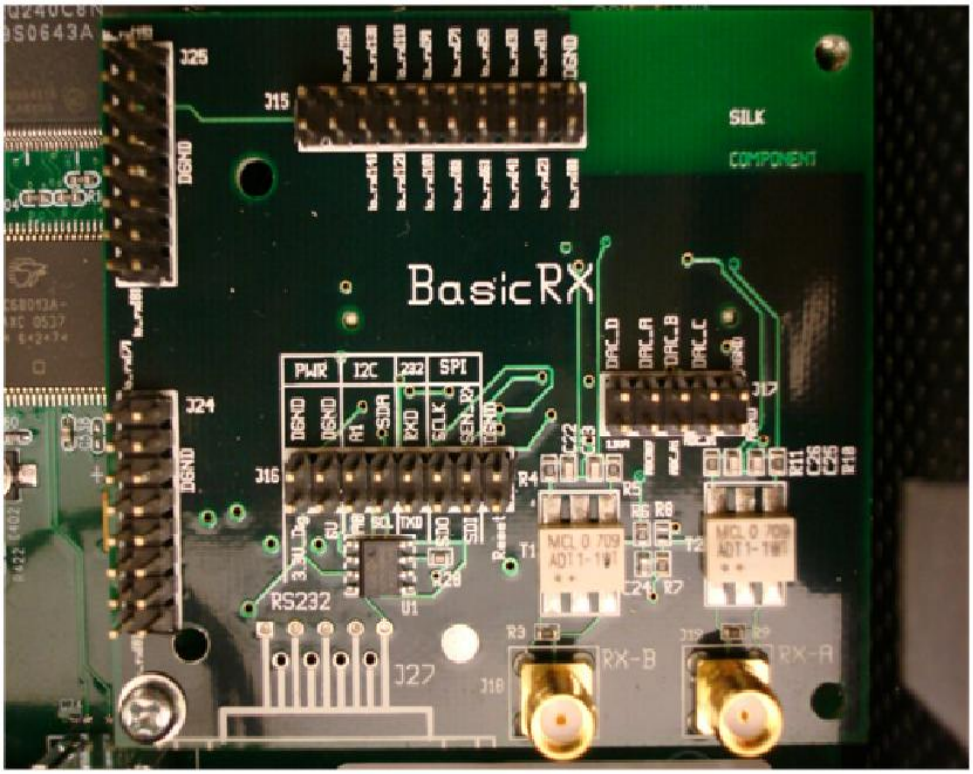


Figure 13: Picture of a basic receiver board, daughterboard, from USRP

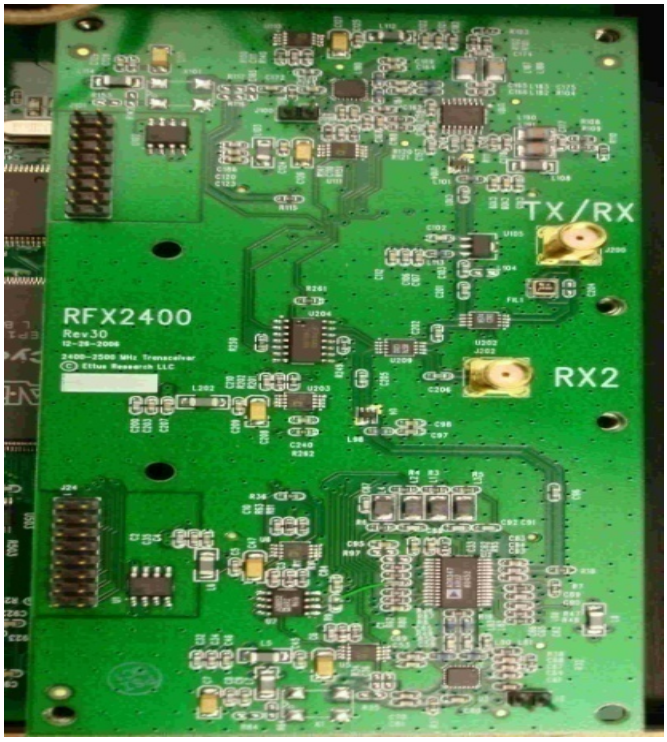


Figure 14 2.4 GHz Front End - RFX2400

3.3 GNU Radio platform

In this thesis project we have used the GNU Radio [31] platform, an open source software toolkit, as our software radio platform. Figure 15 shows a logical block diagram of our SDR receiver. From this figure, it is easy to see the main functional parts of this SDR: a receive RF front end, one or more ADCs, and code (running on a general purpose processor). The ADC transforms the analog signal into digital signals that the software can manipulate.

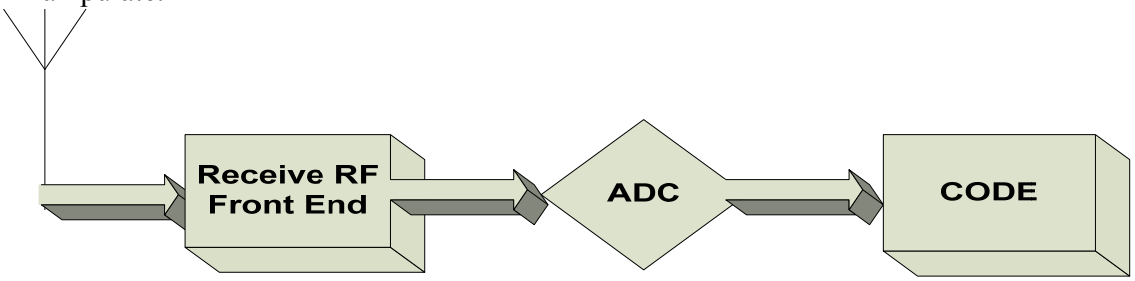


Figure 15: A logical block diagram of our SDR receiver

The GNU radio platform provides us with a large software base, reducing the amount of code which we need to develop and isolating us from some of the details of the physical SDR device. Unfortunately, this means that we need to understand how to use this platform. Details of our experiments to learn about this platform are given in section 3.4.

Signal process block is already available in GNU radio, which is very useful for most soft radio users, since users can save time and energy to write the same function blocks. On the other hand, of course, the users can always write their own function blocks. In our project, we use the default signal processing block and Linux operational system.

3.4 Universal Software Radio Peripheral (USRP)

The Universal Software Radio Peripheral (USRP)[33] has been design by Matt Ettus to provide a radio peripheral which can provide the necessary receiver (or transmitter) front end, a number of high speed ADC and DACs, and some simple digital signal processing to reduce the amount of data which has to be passed to the general purpose processor which executes the rest of the code.

The USRP provide the hardware of our SDR. It enables engineers to easily design and implement a variety of radio systems [37]. Combining the USRP with the open source GNU Radio software enables engineers to build a new radio system with minimal effort and moderate cost. A lot of developers and users have contributed code to the open source project and there are many examples of practical applications available. An important advantage of this combination is that sharing both hardware knowledge and experience significantly promotes the development of soft defined radio; bringing the design and implementation of radios to a whole new audience. Example of applications of the USRP can be found in Alaelddin Mohammed's thesis[47].

3.4.1 USRP Motherboard

The basic USRP board is designed for high speed signal processing. However, because it connects to a computer using USB 2.0, the maximum data rate to and from the board is limited to ~38 Mbytes per second[1]. Both the ADC and DAC are embedded on the USRP's motherboard, and as well as an FPGA which can be reprogrammed. On our USRP board there are four high-speed ADCs which can sample at 12 bits per sample at 64 million samples per second and four DACs that can output 14 bit samples at 128 million samples per second. These are connected by four input and output channels to the FPGA [40].

3.4.2 USRP Daughterboard

A number of different USRP daughterboards are available, see Table 1. These daughterboards cover a number of different frequency bands. On the motherboard, there are four slots that can be used to connect to daughterboards. In the simplest case this could support two baseband transmitter (BasicTX) daughterboards and two base band receiver (BasicRX) daughterboards. The BasicTX board supports a frequency range of 0.1-200 MHz, while the BasicRX board supports a range of 0.1-300 MHz. In our project, we have used the RFX2400 daughter board that covers the 2.4GHz ISM band. It has a working frequency band ranging from 2250 to 2900 MHz [33]. The daughterboards can be digitally tuned under computer control to the desired frequency of operation.

[1] Using the program – we have measured the actual transmission rate to and from the USRP using a Dell computer to be 16 Mbytes/second.

Toolkit

Table 1: USRP daughterboards [35]

Board	Supported Frequency band
BasicRX	0.1-300 MHz Receive
BasicTX	0.1-200 MHz transmit
LFRX	DC-30 MHz receive
LFTX	DC-30 MHz transmit
TVRX	50-860 MHz receive
DBSRX	800-2400 MHz Transceiver
RFX400	400-500 MHz Transceiver
RFX900	800-1000 MHz Transceiver
RFX1200	1150-1400 MHz Transceiver
RFX1800	1500-2100 MHz Transceiver
RFX2400	2250-2900 MHz Transceiver
XCVR2450	2.4-2.5 GHz and 4.9 to 5.85 GHz Transceiver

The particular version of the USRP which we will use is version 4[34]. The components on the USRP are listed in

Toolkit

Table 2. The features of this device are summarized in Table 3.

Toolkit

Table 2 USRP version 4 components

Fortiming Corp. HC49USMD	64MHz crystal clock
Analog Devices AD9862 (two)	12-/14-Bit Mixed Signal Front-End (MxFE®) Processor for Broadband Communications Each AD9862 Mixed Signal Front-End (MxFE®) Processor for Broadband Communications contains: • two 64 MS/s 12-bit analog to digital Converters • two 128 MS/s 14-bit digital to analog Converters • programmable gain amplifiers • Numerous filters
Altera Cyclone EP1C12	FPGA with 12,060 logic elements (LEs) and up to 234 Kbits of RAM [42] Additionally it supports two internal phase locked loops for general clocking with an internal f_{vco} of 500 MHz to 1 GHz (see table 4-52 on page 4-30 of [42]). This same table gives the PLL output frequency as a function of the speed grade of the chip.
Cypress FX2	High-speed USB 2.0 interface (480 Mb/s)

Table 3: Properties of the USRP version 4 board

Four 64 MS/s 12-bit analog to digital Converters
Four 128 MS/s 14-bit digital to analog Converters
Firmware for the Cyclone to implement: • Digital downconverters (DDC) with programmable decimation rates • Digital upconverters (DUC) with programmable interpolation rates
480 Mb/s USB 2.0 interface to the host processor
Support for up to four daughter cards

Chapter 4 - Mechanism

4.1 General System Design

In our project, the system consists of a mobile device with an IEEE 802.11b interface, a PC - used to analyze the received signals, and a USRP with a 2.4GHz transceiver daughtercard. An outline of how the system works is:

1. The arriving analog signal is received by the RFX2400 RF front end and down shifted to an intermediate frequency signal. This intermediate frequency signal is sampled and digitized by an ADC. The output of the ADC is stored in a FIFO at the desired sampling rate.
2. Data from this FIFO is input to a correlator which looks for an IEEE 802.11 frame header. Upon finding such a header a timestamp is sent to an output FIFO buffer for transmission via the USB interface to the attached host computer. If possible the source MAC address is decoded and also included in the USRP's output – otherwise the host computer needs to have a WLAN interface to record the source MAC addresses of the frames which are received (it can timestamp these and store them in a buffer – to later be paired with the data coming from the USRP). Logically the data is a sequence of ((timestamp, source MAC address), (timestamp, source MAC address), …).
3. In a central server, all the time stamps from multiple receivers for the same frame are processed, in order to locate the target WLAN interface.

Note that in our prototype there is **no** DSP chip. Thus any digital signal processing that is to be performed must be performed by the host computer (which does have the ability to do floating point computation) or via a digital circuit which is designed in VHDL or Verilog and downloaded to the FPGA.

Another feature of this system is that the synchronization of multiple receivers is solved by using time stamps and the known location of one or more fixed receivers. The clocks of these receivers are synchronized using the network time protocol. Importantly, only the time stamping of the detection of the start of a frame needs to be done in real time; all other processing can be done asynchronously at some later time. Also, the signal strength and effects from refraction and reflection are not a concern as the correlator only looks for the first instance of the start of a new frame. The correlator can be reset after the end of a frame.

4.2 Mechanism

The main improvement over the work of Lin Ji[3] is that a time stamp is introduced – preferably in the USRP, as described in the previous section. The timestamp will indicate the detection of the start of a frame. In IEEE 802.11b, there are two types of headers, a long physical layer convergence procedure (PLCP) protocol data unit (PPDU) header and short PPDU header. In the long header format, the synchronization is scrambled with all ones, while for the short header it is scrambled with all zeros. No matter which header is utilized, the SFD is the same for the all IEEE 802.11b frames.

The received signal is divided into two parts, an in-phase component (abbreviated as the "I" component) and quadrate component (abbreviated as the "Q" component). Both

of these are input to ADCs. To locate the target user we use the arrival time of each frame as measured by several receivers to compute the distance between the transmitter and receiver, which is basically the same concept as TDOA. We determine the arrival time by computing a timestamp for the detection of the SFD using a correlator. In Figure 16, the circuit shown in the dash block is used to compute the exact arrival time of each header, which is detected on the last bit of preamble. As shown in the figure, the ADC, FPGA, and a First In First Out (FIFO) buffer are used in this project. Note that the FIFO is implemented by the FPGA and the FX2 chip. Whenever a header is detected, a timestamp is put into the FIFO buffer. These timestamps will be sent to the host via the USB interface.

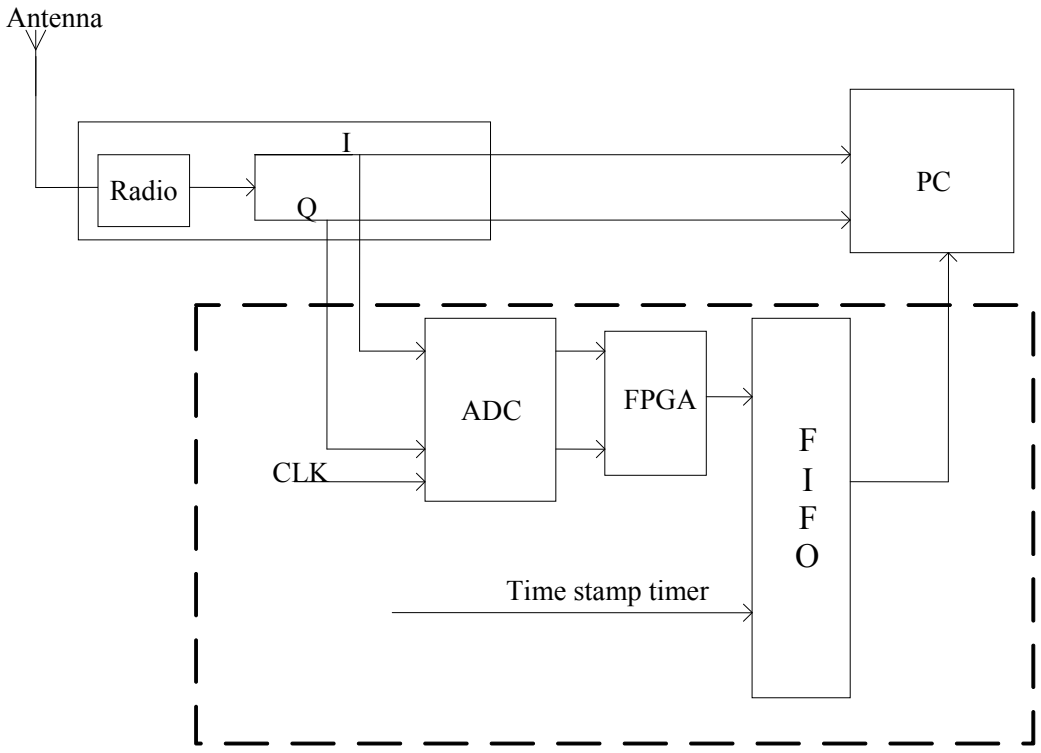


Figure 16: Location measurement system

4.2.1 Algorithm

In our project, we use three receivers to detect the coming signal, as shown in Figure 17. Three receivers are the minimum requirement for detection, from mathematic point of view. Of course more receivers can give a better resolution, but might consume more energy. In our project, we will only discuss three receiver case. From Figure 17, we can see that the user's location can be measured from three detection, however, if there are less receivers, then we can only get a probably area.

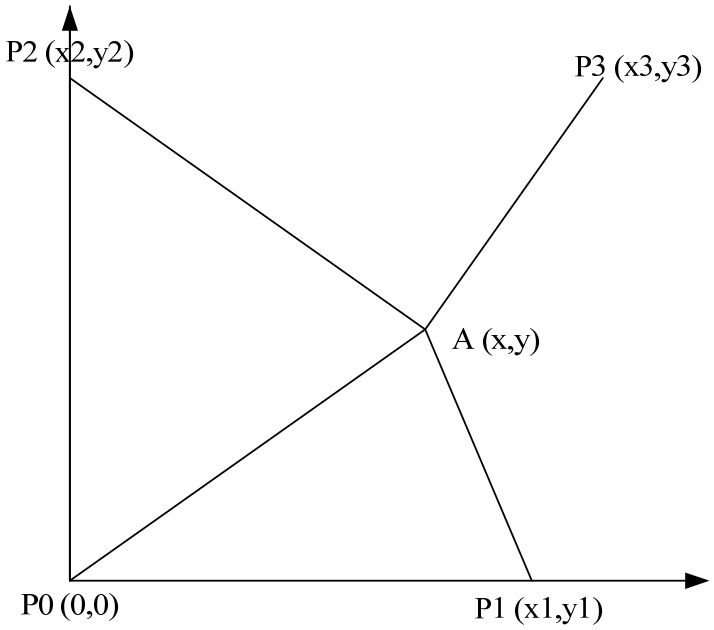


Figure 17: Location measurement system

The figure above shows how to compute the location of the A (the transmitter), given P0, P1, P2, and P3 are the known location of the receivers, *t* is time when the signal was transmitted, and *t0*, *t*1, *t*2 and *t*3 are signal arrival times, *v* is the transmission speed of the signal, which is $3*10^8$ m/s.

$$x^2 + y^2 = (to - t)v \quad (1)$$
$$(x1 - x)^2 + y^2 = (t1 - t)v \quad (2)$$
$$x^2 + (y2 - y)^2 = (t2 - t)v \quad (3)$$
$$(x3 - x)^2 + (y3 - y)^2 = (t3 - t)v \quad (4)$$

In the experiment, the locations of all the receivers are known, and we can learn the signal arrival time as well. As a result, we can calculate the transmitter's location (*x*, *y*) from the above equations. As the algorithm is quite similar to TDOA, we can expect some problems just as in TDOA. In the following section, we propose a time stamp approach to solve the synchronization problem.

4.2.2 Scrambler

The scrambler works as showed in Figure 18. The output of a scrambler is totally different from the input, thus avoiding patterns in the output. The result of the scrambler for a given input sequence depends upon the scrambling code.

Mechanism

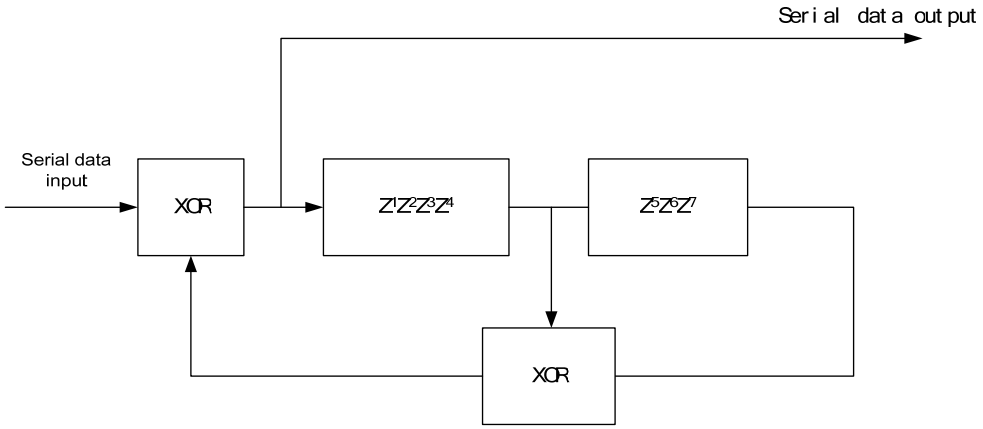


Figure 18: Data Scrambler

If the input signal is a signal with some particular pattern, then the output signal of the transmitter (without the scrambler) would also have this pattern. For example, the preamble is a 128 bit sequence of ones, this would lead to a narrow frequency spike in the power spectrum. As a result, it would interference with adjacent channels due to the cross modulation and intermodulation as well, to avoid this, a scrambler is introduced to disperse the power spectral density.

The scrambler polynomial used in our project is shown in the equation below. The scrambler is initialized with the value '1101100'.

$$G(z) = z^{-7} + z^{-4} + 1 \qquad (5)$$

4.2.3 BPSK

In the real-world, the signal along with some noise will be received by the receiver. We need to ensure that the received signal has a sufficiently high signal to noise ratio (SNR) that the signal can be reconstructed after demodulation. Phase-Shift Keying is an effective modulation technique to protect the signal against noise. In our experiment, the transmission is implemented with a binary code; hence we use DBPSK to modulate the received signal. DBPSK uses two phases which are separated by 180°. Due to the modulation mechanism, BPSK is the most robust modulation method of all PSKs, because only a very large distortion can lead to an incorrect demodulation. Thus using DBPSK decreases the probability of an error.

When the transmitted signal arrives at the receiver, the PLCP preambles arrive before header. The synchronization part of preamble has been modulated with DBPSK, and the modulated preamble is shown in Figure 20.

Mechanism

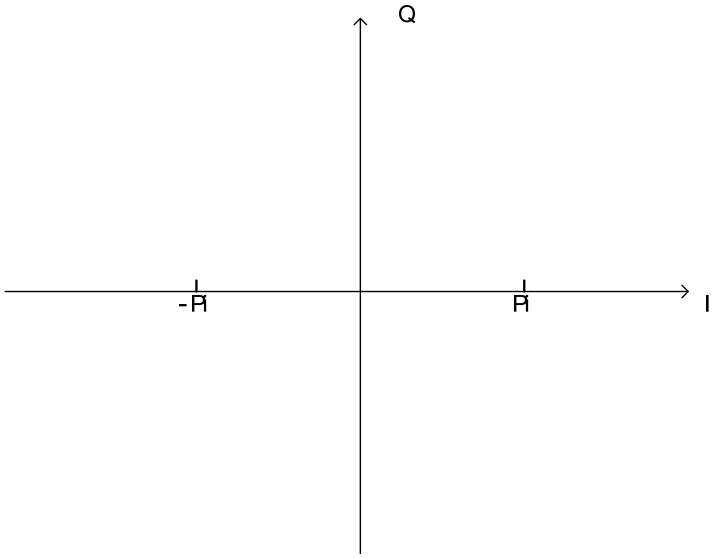


Figure 19: DBPSK constellation

After stretching the x-axis, the synchronization part of IEEE 802.11 signal after DBPSK modulation can be seen Figure 20. The constellation of DBPSK is presented in Figure 19. Pi or –pi represent the phase information of the binary code. As shown in this figure, no quadrate information is contained in DBPSK, only the in-phase component carries information. The relationship for between a binary number and the corresponding carrier wave phase are presented in Table 4

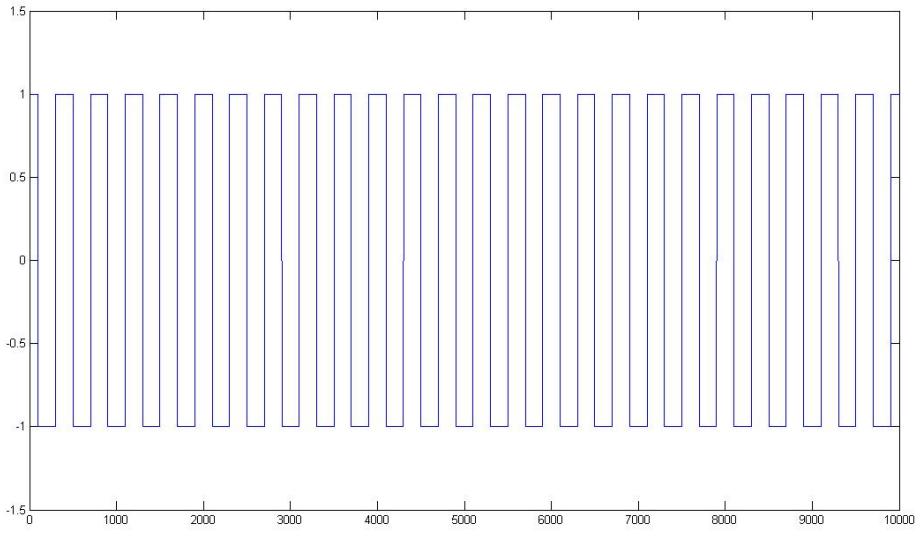


Figure 20: DBPSK Modulated IEEE 802.11 sync. after stretching

Mechanism

Table 4: Relationship of binary numbers and corresponding carrier wave phase

Binary Number		1	1	0	1	0	0	1
DBPSK Signal Phase	0	Pi	0	0	Pi	Pi	Pi	0
OR	Pi	0	Pi	Pi	0	0	0	Pi

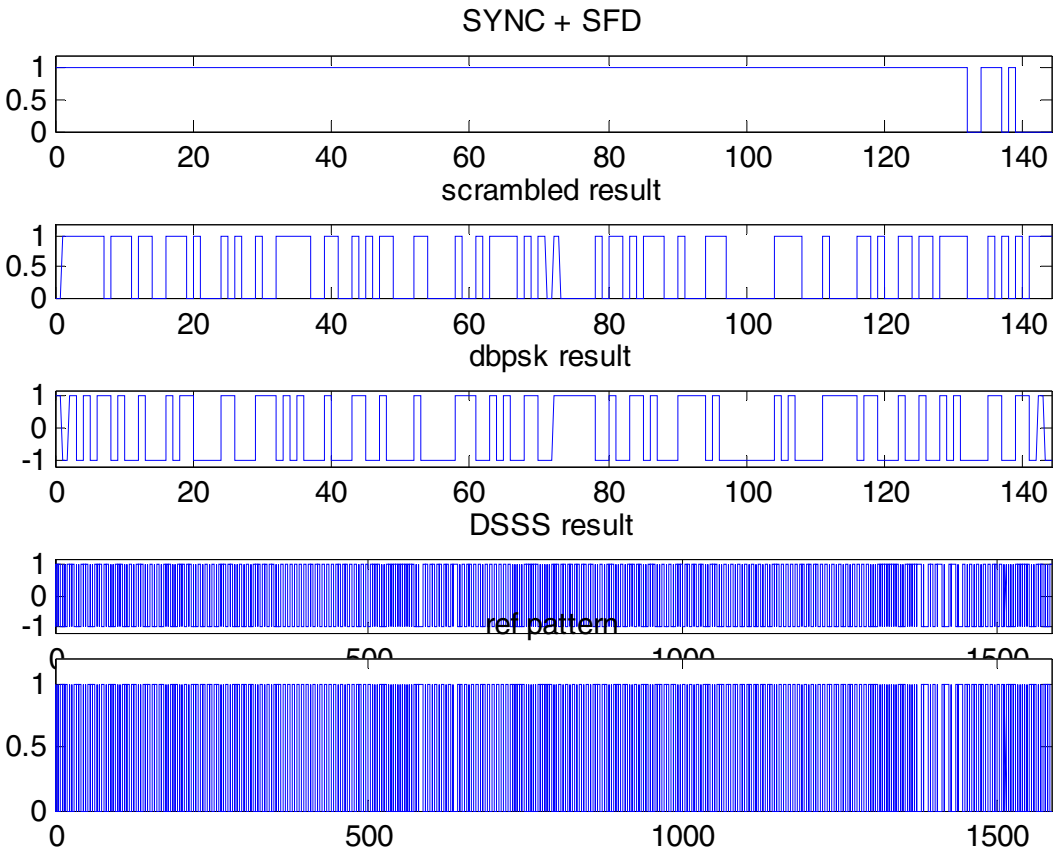


Figure 21: Signal Pattern after each block in the signal flow

The SYNC pattern is sent to scrambler first, then to DBPSK modulator. The resulting signal will be the reference pattern used by the receiver to synchronize with the transmitter. The signal flow is shown in Figure . The signals at each stage of this processing are shown in Figure 21

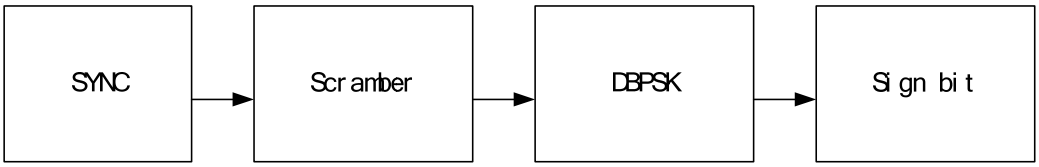


Figure 23:

4.2.4 Time Stamp

When a signal arrives at the receiver, a convolution is performed between the incoming signal and a pre-defined string. In our project we will use the detection of a peak value to know when the SFD is detected.

Due to the limited maximum data rate of the USB interface we will transfer only a time stamp (and possibility the source address of the receive frame). We will stored all of the data that we will subsequently need in a FIFO buffer, thus decoupling the real-time operation of correlation and timestamp generation from the asynchronous computation to be performed by the general purpose process (implemented on a PC).

We used MathWork's MATLAB[†] to generate a SYNC sequence, then processed it by the predefined scrambler, this turns the highly regular signal into a signal without any obvious pattern, reducing interference during transmission. After DBPSK modulation, due to the binary transmission, we map -pi to 1 and pi to 0. In this case, the output from the signal in Figure 21 is a fixed binary string, as long as the input is exactly a SYNC followed by a SDF string. This reference pattern will be stored in a shift register and will **not** be changed during the detection process. In another shift register, we store the incoming signal in a FIFO manner. To perform the cross correlation we perform an XOR calculation with the predefined reference pattern, bit by bit, as shown in Figure 23

[†] For further details about MathWorks MATLAB see http://www.mathworks.com/.

Mechanism

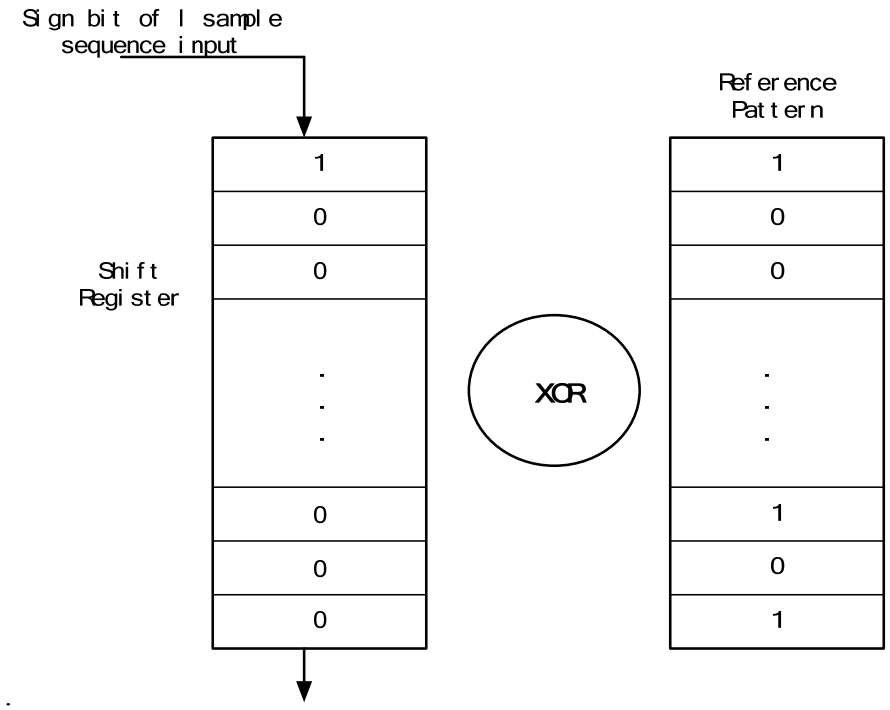


Figure 23: Correlator

When correlation exceeds some threshold, we record the time stamp and place the timestamp into an output FIFO buffer and send it to host PC. The result of this time stamp operation is shown in Figure 24.

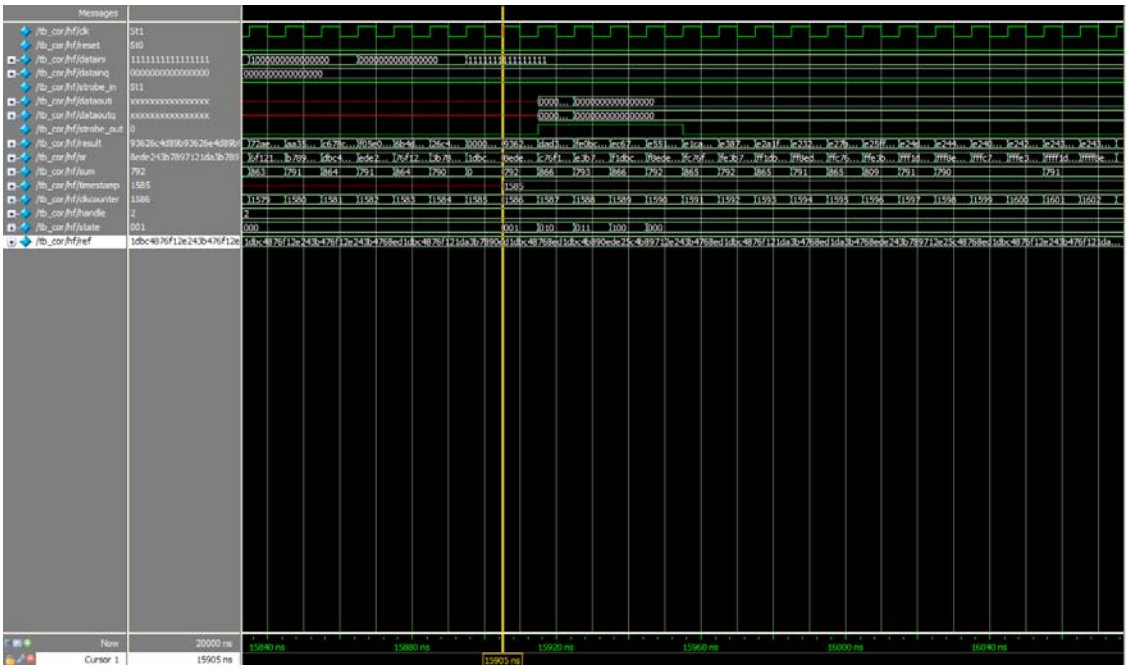


Figure 24: Result of TimeStamp operation

As we can see, the clock is running all the time. This gives use a clock value for every XOR result from the shift register. In Figure 24, the time stamp is ’1585’, which is put in the output FIFO buffer.

The results of XOR calculation are shown in Figure 25. When the predefined pattern is found (i.e., a match of the pattern occurs), we can a very large peak in the correlation calculation, indicating that the header has been found.

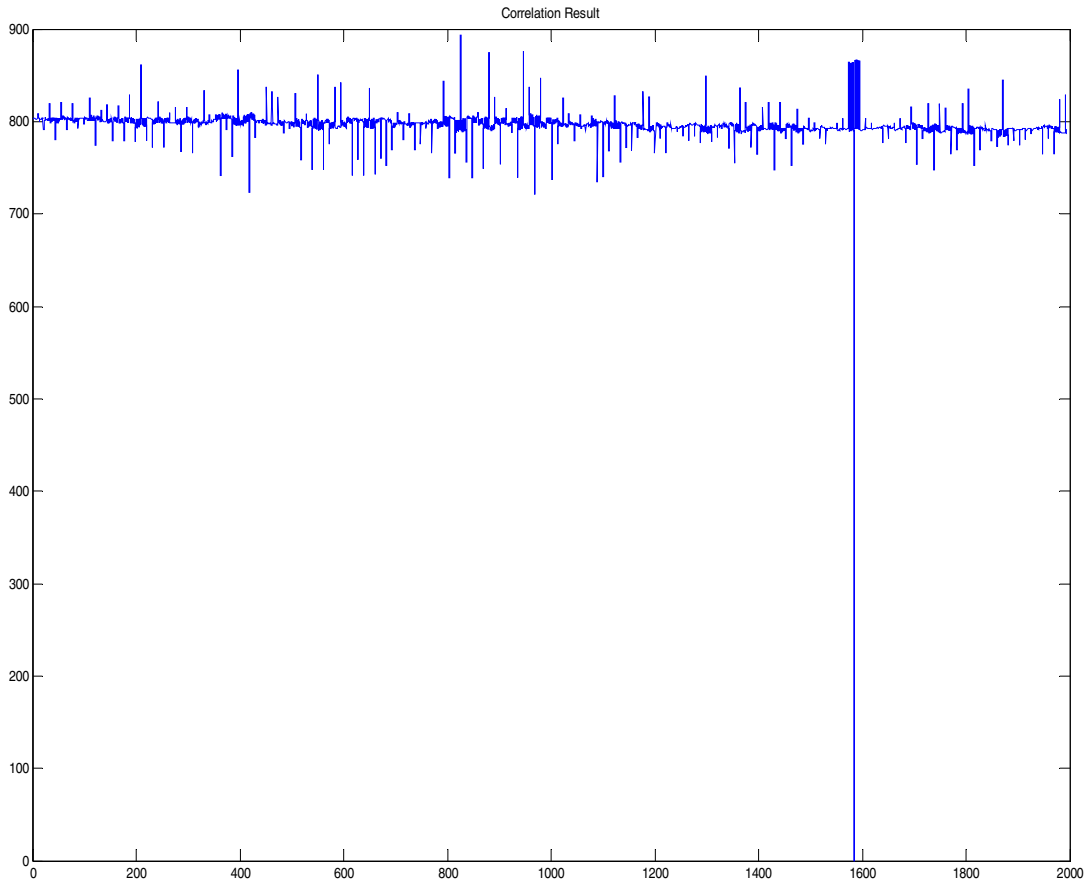


Figure 25 Correlation Result of input signal

4.2.5 USB Transmission

As we stated before, the maximum data transmission rate via USB is 32MBps. For IEEE802.11b, the transmission bandwidth is 11MHz, which requires a sampling rate of 22Msample/s to avoid aliasing according to Nyquist’s Sampling Criteria. If we assume the resolution of the ADC was 8bits, then the transmission rate required to transmit both I and Q samples would be 22Msample/s * 8bit/sample * 2 = 44MBps, which is greater

than the USB transmission rate. Fortunately, it is not necessary to transmit every sample value to the PC via USB, as we only send the time stamp information to the PC.

In our thesis project, the algorithm calculating the arrival time stamp for the newly received frame is mostly done in the FPGA and the timestamp placed in the outgoing FIFO buffer. This FIFO buffer is connected to the host via a USB 2.0 implemented by the FX2 chip on the USRP motherboard. The maximum transmission speed of USB 2.0 is 32Mbytes per second. Moreover to achieve this high data rate the data has to be sent in a large block. The smallest packet that can be sent is 512bytes, and the maximum block size is defined by specifying 'fusb_nblock' and 'fusb_block_size' [40] when the USRP is initialized. However, if the product of fusb_nblock and fusb_block_size is too large, the delay will be increased – since data produced by the USRP will only be delivered within a block transfer. In order to limit the overall real time delay, it is better to use small value for these parameters.

Assume the data packet is 512 bytes and data transmission rate of the USB is 32Mbytes per second, then if transmission is started when the data is collected – the transmission delay can be estimated as: t = 512bytes/32MBps = 16μsec. To this we have to add the buffering delay (i.e., the time required to fill the 512 byte block) – to estimate the delay between the detection of a frame and this information being passed to the PC.

Since the USB buffer is sent to the PC only when the buffer is full, then if we assume that each time stamp consists of 8 bytes, then we need to wait till we collect 512bytes/8 bytes = 2^6 = 64 timestamps before the first timestamp in the buffer will be sent to the PC. If the IEEE 802.11 link is very busy this might not be a problem (since we will rapidly get 64 frames), but if the link is not very busy then there will be quite a lot of delay before the timestamps are sent to the PC. We avoid this problem by padding the buffer with zeros to fill out the buffer every time that we get a new time stamp. This means that if the frame rate on the link exceeds ~62500 frames per second, then we can not keep up with the rate that timestamps would be produced. Fortunately this corresponds to a continuous rate of packets that is beyond what the IEEE 802.11b link can sustain, thus this problem is unlikely to happen in practice.

4.2.6 BBN's IEEE 802.11 receiver code for the USRP

The USRP has an EEPROM which stores the VendorID, productID, and a version number. At startup the host processor downloads to the FPGA on the USRP a configuration bitstream. You can regard this bitstream as object code that "programs" the operation of the FPGA. This bitsteam can be generated from a high-level hardware description language program being compiled. In the case of the USRP this is achieved using Verilog (the high level description language). The Verilog code is open source, just like the GNU Radio code, and release under the GNU-GPL license.

Researchers at BBN have implemented an IEEE 802.11 receiver for the USRP and researchers at the SPAN laboratory at the University of Utah have enhanced this code to perform the dispreading in the FPGA (thus reducing the data that must be transferred across the USB) – so that the receiver can run at full rate. These projects are: the SPAN 802.11b Receiver and BBN 802.11b Receiver. The SPAN 802.11b Receiver's goal is to

achieve a full rate of IEEE 802.11b receiver using a USRP. Details of these two projects can be found at [48] and [49].

Chapter 5 - Implementation and Evaluation

5.1 Implementation

In our project, we decided to use TDOA to calculate the location of a WLAN transmitter. However, as we have seen TDOA has a number of problems. To eliminate the problems caused by multipath we decided that we should use a method that would allow us to recognize the first instance of a signal and use the time of arrival of this signal as the arrival time, thus ignoring delay copies of the signal. We also decided to use a scheme that would allow multiple receivers to separately timestamp the signal's arrive and to post process this data to computer the device's location.

Based upon our study of the IEEE 802.11b header structure, we knew that the receiver knows how to recognize the arrival of a frame – the frame starts after it receives the SFD (after synchronizing based upon the preamble). We chose this same event as a well defined indication of the arrival of a frame. However, in our case we detect this by correlating against the expected chip sequence rather than the bit pattern for the SFD. This enables us to have a time resolution proportional to the sampling rate (up to 64 Msps), rather than the bit rate (1 or 2 Mbps) – potentially giving an increase in timing resolution of 32 to 64 times. This would correspond to 15.6 ns time resolution or a spatial resolution of approximately 4.68 m or if the distribution of spatial error is symmetric ±2.34m. This is of the order of magnitude of our desired spatial resolution.

As described in the previous chapter the USB interface from the USRP does have enough bandwidth to deliver the full baseband signal to the attached general purpose processor (in our case a PC), thus some processing has to be done in the FPGA of the USRP (such as the de-spreading processing implemented by the SPAN laboratory at the University of Utah). In our case we chose to implement recognition of the arrival of a frame and to save the value of a 64 MHz clock as a timestamp of when the frame arrived.

[1] To simply the processing that we needed to do on the FPGA we decided to implement the cross correlator in Verilog to recognize the fixed pattern of a preamble followed by an SFD. Therefore, we used Matlab to compute an IEEE 802.11b header based upon the description of the header fields and the scrambler (as described earlier in the thesis). We used Quartus as our develop tool to program the FPGA. Quartus is a good FPGA tool for beginners since it provides a powerful toolbox for developing logic circuits, with a simulator and compiler. The Verilog code that we have developed is included in

. Output from the Quartus simulator was shown earlier in Figure 24. Note that the solution takes approximately 10615 FPGA elements, approximately 88% of the total available element of FPGA, and 27% memory is used while working.

5.2 Experimental evaluation

As presented in Section 4.2.4, once we get a time stamp at the receiver that the signal arrival has been detected, we can calculate the possible location by measuring the difference in transmission time. In our experiments, we used one receiver to measure

three times of the arrival times when it is placed in different position in order to simulate three receivers situation.

As we stated in the previous description, we try to use one receiver to measure three times to simulate three receivers' situation. The first time it is placed in (0, 20)m, the second is located in (23, 0)m and the third is in (30, 32)m. We try to use TDOA to estimate the possible location. However, problem comes. How to synchronize the 'three receivers' during the measurement is the most important issue for our experiment. We need to make sure that the receiver receives the 'same' signal in three times detection. So we can not just put the receivers at three places and get the timestamp. First we tried to swap transmitter with receiver, that is, use three transmitters to send signals at the same time and use one receiver to detect them. But in realistic, this way doesn't work well since the clock is not accuracy and it is very possible that the signals will interfere each other. So we tried with another model, that is use an anchor point as a reference, to synchronize the receivers, using one receiver to place in three positions. This model can be shown as Figure 26

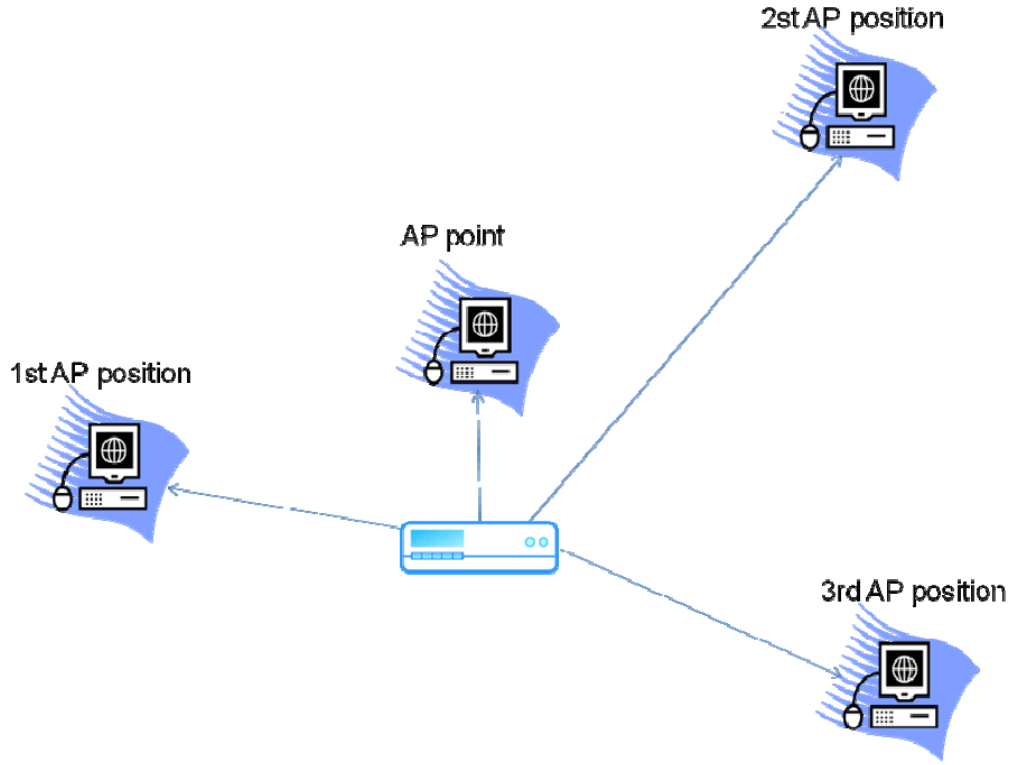


Figure 26 Our model to synchronize three transmitters

We first put the receiver at AP point (about 5 meters away from the transmitter), take this arrival time as a reference, and then keep the system running and move the receiver to the first AP position, and detect the arrival time again. Same for the other two positions' measurements. In this case, we can not get the exact arrival time at those three AP positions, but it is easy to get the time difference. And this value is what we need to plot the possible user's area with TDOA.

In Figure 27, the red spots are receivers' location and the overlap area is the possible position of transmitter. Compare these two figures, we can see that, TDOA gives a much better resolution of the possible position.

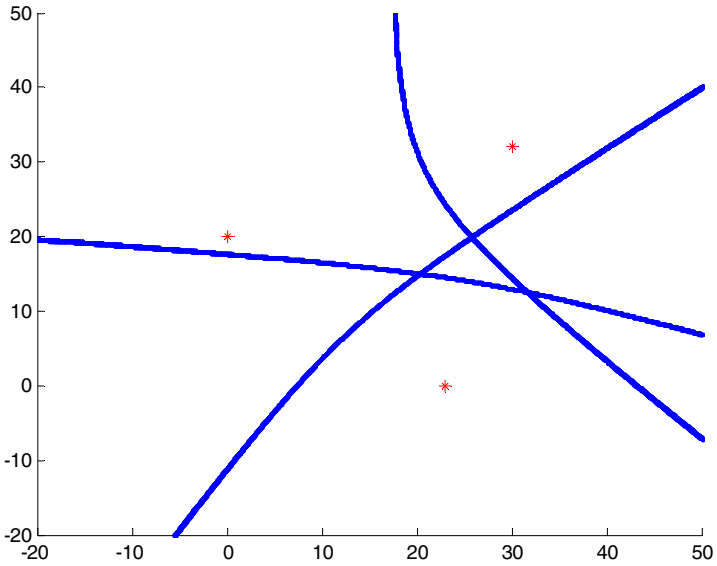


Figure 27 Single TDOA Measurement

Due to the sampling rate of our hardware and the maximum clock rate of the FPGA, the user will be located to within a radius of approximately 10 meters, which is not sufficient to meet our goals for location in an indoor environment. To enhance the accuracy, the measurements can be performed several times. Figure 28 illustrates two measurements for a stationary user, the overlapping intersection area is smaller than that in Figure 27. In the beginning of this experiment, the mechanism was designed to carry out the calculation automatically, whenever a signal is captured and the arrival time will be sent to PC to calculate the distance. Then multi-measurement is feasible to narrow down the detected area. However, during experiment, another unexpected error happened frequently, that is the FPGA cannot detect if there is an arrival signal, even though there actually is. So we have to manually take the arrival time when the header finder works, then input those data to matlab program to calculate. This stopped us from continually measuring.

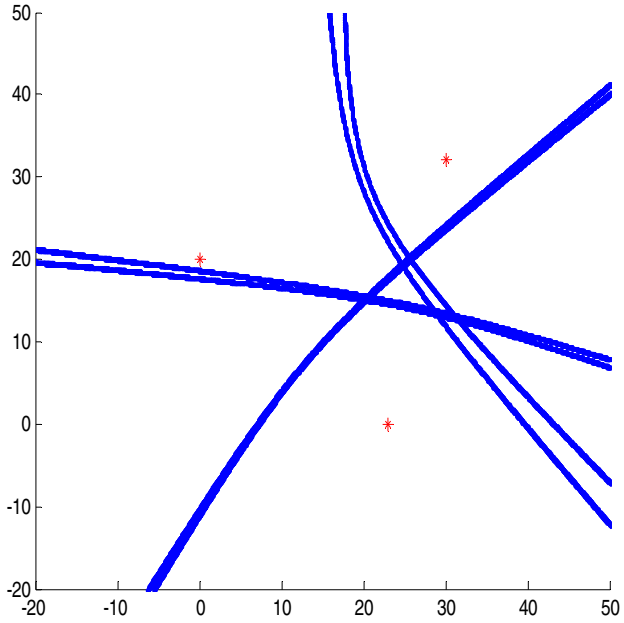


Figure 28 Multiple measurement of arrival signal from different APs

For a stationary user, the accuracy can be enhanced by making a lot of measurements. For such a station user we need to make 10 measurements to estimate the users position to within 10 m. If the user is performing a file transfer the requisite number of packets might be transmitted within xx seconds; while in the case of a VoIP user this number of packets would correspond to xx seconds (with 50 packets per second being transmitted by an active audio RTP source). Note that we do not have to be able to see what the user is transmitting nor do we care what higher level protocol they are using, the only requirement is that the user has to transmit.

However, this raises a new question: Will we have enough time to perform such multiple measurements for a moving user? Or the related question: How accurately can we track a moving user in real-time? However, we have not been able to answers either of these questions in this project and leave them to future work.

As we can see the result even for a stationary user is not as accurate as we expected in the beginning. When we consider the whole process, we can see that there are several reasons for this. First, the correlation mechanism is working, but maybe not the most efficiency one. During our testing period, one problem occurred many times. We sent a signal, but the FPGA did not detect the signal's arrival since the correlation result did not satisfy the selection requirement. A better correlation scheme should be developed. Second, the other reason for some of our problems could be our unstable test environment, as there many IEEE 802.11b packets being transmitted both from many devices and many access points. This problem has to be addressed as it will occur in the real world.

Chapter 6 - Conclusions

As presented in the previous chapters, it is encouraging to see that the solution works when trying to detect a mobile user. From our experiments, the detection works without requiring any changes to the hardware or software of the mobile device. The detection process is relatively simple and requires very little additional hardware in the receiver. In our experiment, we used an inexpensive FPGA with a relatively low rate sampling rate (in comparison to the state of the art) to located the user based upon packets sent by the user's device as measured at multiple receivers.

The computations of the device's location could be performed by a PC attached to the LAN that would receive packets from sets of modified access points. Note that if the access points already have an FPGA to implement their receiver, this might be able to be reprogrammed to add this functionality to existing APs. However, we have not examined any existing access points to see if this is true in practice.

By using IEEE 802.11b transmission the cost and difficulty of deployment is simplified due to the wide usage of IEEE 802.11b in mobile device. Additionally, the preamble has good correlation properties and it is easy to understand how one can detect the arrival of a frame. These two reasons are significant advantages for a location detection system. Our implementation on open source hardware (and software) shows that it is possible to implement this solution at relatively low cost (in terms of numbers of gates), making this solution feasible for commercial implementation.

Chapter 7 - Future work

Indoor location detection is a service that will be widely used in the future. In our project, the basic principle of operation for a time of arrival method based upon recognizing the start of frames using chips has been verified. However, our preliminary testing results show that more work on this topic is needed. Some of the ideas to explore are:

1. A higher sample rate can be used to enhance the accuracy. How much gain is possible with a device such as the USRP2 (the latest generation USRP)?
2. A better correlation mechanism should be developed. This method could take advantage of more samples – as provided by the first idea above.
3. Exploit statistics to reduce the error based upon multiple measurements for both stationary and moving devices.

It is important to note that we can exploit knowledge about the expect time variance with respect to the phase of the clock. Thus in our case we know that the time stamp advances at the rate of the sampling clock, in our case 64MHz. Hence the time stamp records the last tick of the clock, we can only tell that the signal took longer than time the clock would have shown the start of the frame being transmitted by the transmitter (if our clock were located at the transmitter) and the time that the start of the frame was detected by our receiver. Hence the timing error distribution is not symmetric, since the measured time different can not be shorter than the propagation time, but can be longer due to the finite resolution of our clock.

Due to the limited time for this project, we were unable to explore the above ideas or to make more extensive measurements. We recommend that another student should start by learning how GNU radio works before trying to implement an improved solution to the problem.

References

[2] Dong Mei and Wang Neng. Signal strength based WLAN location determination technology. Computer Applications. Vol. 24, No. 12, December 2004

[3] Lin Ji, "Increasing Accuracy of Location Determination: Exploiting Phase Change Reconstruction and Timing Measurements", Masters Thesis, Department of Communications, School of Information and Communications Technology, Royal Institute of Technology (KTH), Stockholm, Sweden, COS/CCS 2007-14, May 2007. http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/070516-Lin_Ji-Thesis-with-cover.pdf.

[4] Z.Xiang, S.Song, and J.Chen. A wireless LAN-based indoor positioning technology. IBM Research in Asia. Vol. 48, No. 5. 6, 2004.

[5] Mark T. Smith and Gerald Q. Maguire, Jr., Receiver and Correlator used to Determine Position of Wireless Device, US Patent 6,975,618, Issued December 13, 2005; Filed June 26, 2001.

[6] Xiang Jie, IEEE 802.11b, http://publish.it168.com/cword/670.shtml

[7] DSSS Introduction, http://en.wikipedia.org/wiki/DSSS

[8] Time-of-arrival location technique, Los Alamos National Laboratory, Report number 00416648, July 29, 1999, Printed in Los Alamos Science and Technology Magazine, Summer 1982, http://www.fas.org/sgp/othergov/doe/lanl/pubs/00416648.pdf

[9] TOA Model http://www.ciscosky.org/network/wireless/ShenWangLaoZhongDeMoXianDingLiJiShu.htm

[10] F. Gustafsson and F. Gunnarsson, Acoustics, Positioning using time-difference of arrival measurements, Speech, and Signal Processing, 2003. Proceedings. (ICASSP apos; 03). 2003 IEEE International Conference on Volume 6, Issue, 6-10 April 2003 Page(s): VI - 553-6 vol.6 Digital Object Identifier 10.1109/ICASSP.2003.1201741.

[11] Rong Peng and Mihail L. Sichitiu, Angle of Arrival Localization for Wireless Sensor Network,; Department of Electrical and Computer Engineering; North Carolina State University,

References

[12] Location techniques by detecting Arrival of Angle
http://www.chinaecnet.com/xsj/xsj032821.asp

[13] Eric Blossom, "Exploring GNU Radio", 29 November 2004.
http://www.gnu.org/software/gnuradio/doc/exploring-gnuradio.html

[14] Haruumi Shiode, "In-building Location Sensing Based on WLAN Signal Strength: Realizing a Presence User Agent", Masters Thesis, Department of Communication Systems, School of Information and Communication Technology, Royal Institute of Technology (KTH); Stockholm, Sweden, March 2007
http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/080314-Haruumi_Shiode-with-cover.pdf

[15] Teemu Roos, Petri Myllymäki, Henry Tirri, Pauli Misikangas, and Juha Sievänen, "A Probabilistic Approach to WLAN User Location Estimation", International Journal of Wireless Information Networks, Vol. 9, No. 3, July 2002 http://www.cs.helsinki.fi/u/ttonteri/pub/ijwin02.pdf

[16] Ali Taheri, Arvinder Singh, and Emmanuel Agu, "Location fingerprinting on infrastructure 802.11 wireless local area networks (WLANs) using Locus", 29th Annual IEEE International Conference on Local Computer Networks (LCN'04), 2004, pp.676-683.
http://doi.ieeecomputersociety.org/10.1109/LCN.2004.74
http://ieeexplore.ieee.org/iel5/9433/29935/01367305.pdf?arnumber=1367305

[17] Stuart A. Golden and Steve S. Bateman, "Sensor Measurements for Wi-Fi Location with Emphasis on Time-of-Arrival Ranging", IEEE Transactions on Mobile Computing, Vol. 6, No. 10, October 2007, pp. 1185-1198
http://doi.ieeecomputersociety.org/10.1109/TMC.2007.1002

[18] Mikko Koskela, "Location-based services in wireless local-area networks", Tampere University of Technology, 22 November 2002
http://trc.pori.tut.fi/tots/Mikko_loppuraportti_final.pdf

[19] Behdis_Zandieh, "Indoor Wireless Local Area Network (WLAN): Measurement and Modeling from a user perspective", Masters Thesis, Department of Communications, School of Information and Communications Technology, Royal Institute of Technology (KTH), Stockholm, Sweden, COS/CCS 2007-12, 28 March 2007

http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/070328-Behdis_Zandieh-with-cover.pdf

[20] U.S. Federal Communications Commission, E911 PHASE II Decisions: Introduction, FACT SHEET, U.S. Federal Communications Commission, Washington, DC, October 2001 http://www.fcc.gov/Bureaus/Wireless/News_Releases/2001/nwl0127a.pdf

[21] Public Safety and Homeland Security Bureau, Enhanced 911 - Wireless Services, U.S. Federal Communications Commission, Washington, DC, Last accessed 2008.09.22 http://www.fcc.gov/pshs/services/911-services/enhanced911/Welcome.html

[22] Sun Yu, "Context-aware applications for a Pocket PC", Masters Thesis, Department of Communications, School of Information and Communications Technology, Royal Institute of Technology (KTH), Stockholm, Sweden, COS/CCS 2007-28, 20 December 2007. http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/071220-Sun_Yu-with-cover.pdf

[23] IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999), June 12 2007 Page(s):C1 – 1184, Digital Object Identifier: 10.1109/IEEESTD.2007.373646

[24] Jia Zhou, Adding bandwidth specification to a AAA Sever, Masters Thesis, Department of Communications, School of Information and Communications Technology, Royal Institute of Technology (KTH), Stockholm, Sweden, COS/CCS 2008-19, September 2008. http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/080914-zhoujia-with-cover.pdf

[25] IEEE 802.11 Working Group, IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, IEEE, 802.111997, ISBN: 978-0-7381-3044-6, Published 1 January 1999.

[26] IEEE 802.11 Working Group, Standard for LAN/MAN - Specific requirements, Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer

(PHY) Specifications, IEEE, 802.11-2007, ISBN 978-0-7381-5655-2, Published 1 January 2007, 1228 pages.

[27] IEEE 802.11 Working Group, Part 11: Wireless LAN MAC and PHY Specifications: Higher Speed Physical Layer (PHY) Extension in the 2.4 GHz Band, IEEE, 802.11b-1999, ISBN 978-0-7381-5230-1, Published 1 January 1999.

[28] Jeffrey Steinheider, Victor Lum, and Jonathan Santos, "Field Trials of an All-software GSM Basestation", Proceedings of the Software Defined Radio Forum Technical Conference, Orlando, FL, November 2003. http://www.vanu.com/wp-content/resources/publications/Basestation_SDRFpaper.pdf

[29] Steve Muir, "Software Radio Versus Traditional Telecommunications Equipment Manufacturers (TEMS) -- Differing Philosophies Polarize Market", The Basestation Newsletter, 15 July 2008. http://www.openbasestation.org/Newsletters/July2008/vanu.htm

[30] Joseph Mitola III and G. Q. Maguire Jr., Cognitive Radio: Making Software Radios More Personal, IEEE Personal Communications, 6(4):13-18, ISSN: 1070-9916, August 1999.

[31] Eric Blossom, GNU Radio - GNU FSF Project, Free Software Foundation, 2007/03/02 03:07:48 www.gnu.org/software/gnuradio/,

[32] Stephen Cass, Hardware for Your Software Radio, Spectrum, IEEE, October 2006. http://spectrum.ieee.org/oct06/4654

[33] Matt Ettus, "Universal Software Radio Peripheral (USRP)", Ettus Research LLC http://www.ettus.com/

[34] Matt Ettus, "Universal Software Radio Peripheral: The Foundation for Complete Software Radio Systems", Ettus Research LLC, 1 November 2006, 2 pages.

[35] P. Balister and J. Reed, "USRP Hardware and Software Description", Chameleonic Radio Technical Memo No. 9,Bradley Dept. of Electrical & Computer Engineering Virginia Polytechnic Institute & State University Blacksburg, VA 24061, June 30, 2006 http://www.ece.vt.edu/swe/chamrad/crdocs/CRTM09_060727_USRP.pdf

References

[36] S.M. Shajedul Hasan and P. Balister, "Prototyping a Software Defined Radio Receiver Based on USRP and OSSIE", Chameleonic Radio Technical Memo No. 1,Bradley Dept. of Electrical & Computer Engineering Virginia Polytechnic Institute & State University Blacksburg, VA24061, December 14, 2005. http://www.ece.vt.edu/swe/chamrad/crdocs/CRTM01_051214_USRP.pdf

[37] USRP Family brochure, 2008, Brochure for the entire USRP product family

[38] USRP motherboard http://gnuradio.org/trac/wiki/UsrpFAQ/Intro/Mobo

[39] Jiang Zhou and Puxian Sun, "Radiolocation technology in cellular communication systems" , Electronic Products China, 2003.

[40] Timing Latency Questions http://gnuradio.org/trac/wiki/UsrpFAQ/Latency

[41] Jordi Solsona, Motion and location determination for RFID systems, Masters Thesis, School of Information and Communication Technology, Royal Institute of Technology (KTH), December 2008.

[42] Cyclone FPGA Family Data Sheet, Altera Corporation, May 2008 http://www.altera.com/literature/hb/cyc/cyc_c5v1_01.pdf

[43] S.M. Shajedul Hasan and P. Balister, "Prototyping a Software Defined Radio Receiver Based on USRP and OSSIE", Chameleonic Radio Technical Memo No. 1, Bradley Dept. of Electrical & Computer Engineering, Virginia Polytechnic Institute & State University Blacksburg, VA, December 14, 2005 http://www.ece.vt.edu/swe/chamrad/crdocs/CRTM01_051214_USRP.pdf

[44] How GPS works: Accuracy. http://www.kowoma.de/en/gps/accuracy.htm

[45] Paramvir Bahl and Venkata N.Padmanabhan, "RADAR: An In-Building RF-based User Location and Tracking System", Proceedings of IEEE Infocom 2000, Tel-Aviv, Israel, March 2000 http://research.microsoft.com/en-us/groups/sn-res/infocom2000.pdf

[46] Cisco Location Solution Overview, http://www.cisco.com/en/US/solutions/collateral/ns340/ns394/ns348/ns753/net_brochure0900aecd8064fe9d.pdf

[47] Alaelddin Mohammed, Studying Media Access and Control Protocols, Masters Thesis, Department of Communications, School of Information and

Communications Technology, Royal Institute of Technology (KTH), Stockholm, Sweden

[48] BBN Code Forum http://www.mail-archive.com/discuss-gnuradio@gnu.org/msg15066.html

[49] Implementation of Full-Bandwidth 802.11b Receiver, Sensing and Processing Across Networks at the University of UTAH, http://span.ece.utah.edu/pmwiki/pmwiki.php?n=Main.80211bReceiver

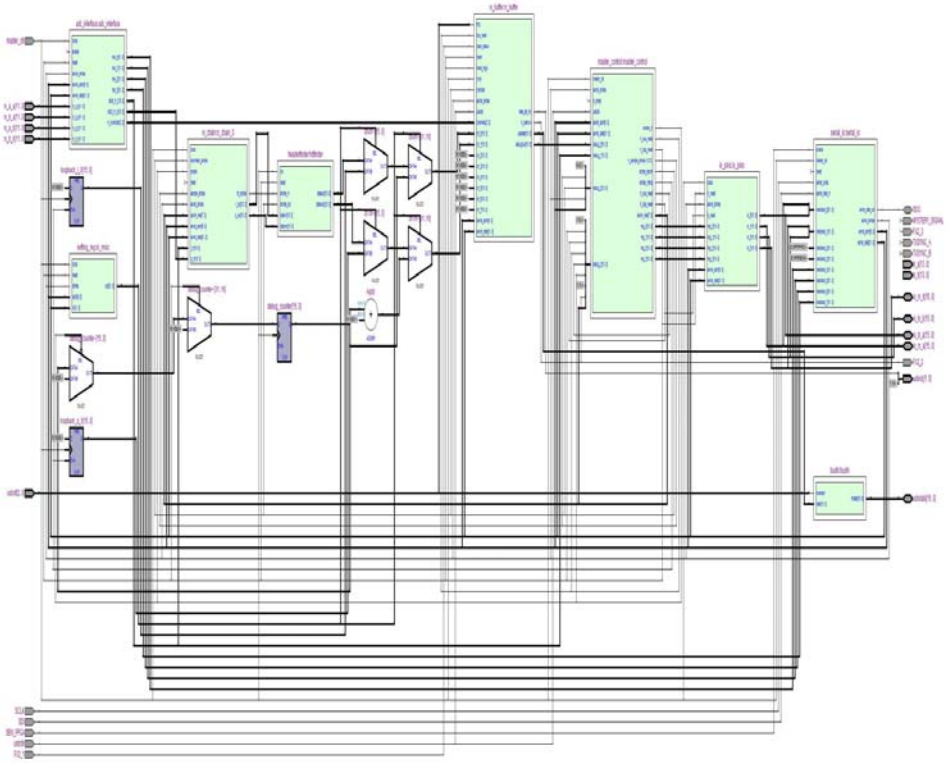
Appendices

A. Matlab work to generate verilog code

```
function gencode()
% generate part of verilog code

%% initialization
Barker = [1 -1  1  1 -1  1  1  1 -1 -1 -1]';
SYNC = ones(128,1);
SFD = [1 1 1 1 0 0 1 1 1 0 1 0 0 0 0 0]';
bits = [SYNC;SFD];
fid = fopen('cor.v', 'w');

subplot(5,1,1);
plot_bits(bits);
axis([0 Inf 0 1.2]);
title('SYNC + SFD');
%% scramble the sample
scrambled = wlan_scramble(bits);
subplot(5,1,2);
plot_bits(scrambled);
axis([0 Inf 0 1.2]);
title('scrambled result');

%% DBPSK modulattion
mod = dpskmod(scrambled,2);
rmod = real(mod);
subplot(5,1,3);
plot_bits(rmod);
axis([0 Inf -1.2 1.2]);
title('dbpsk result');

%% DSSS
dmod = reshape(Barker*rmod', [],1);
subplot(5,1,4);
```

```
plot_bits(dmod);
axis([0 Inf -1.2 1.2]);
title('DSSS result');

%% extract the sign bit
vcode = (-1*dmod + 1)/2;
subplot(4,1,4);
plot_bits(vcode);
axis([0 Inf 0 1.2]);
title('ref pattern');
%% generate the code

array_length = length(vcode) - 1;
fprintf(fid, 'wire [%d:0] result;\n', array_length);
fprintf(fid, 'reg [%d:0] sr;\n', array_length);
fprintf(fid, 'wire [15:0] sum;\n');
fprintf(fid, 'wire [%d:0] ref;\n\n', array_length);

fprintf(fid, 'assign result = sr ^ ref;\n');

fprintf(fid, 'assign sum = ');
for i = 0:array_length
    fprintf(fid, 'result[%d] + ', i);
end

fprintf(fid, 'result[%d];\n', array_length);

for i = 1:length(vcode)
    fprintf(fid, 'assign ref[%d] = %d;\n', i-1, vcode(i));
end

fclose(fid);
```

B. USRP System Flow



C. Verilog code

```
// -*- verilog -*-
module headerfinder (clk, reset, datainі, datainq, strobe_in, dataouti,
dataoutq, strobe_out);

input clk, reset;
input [15:0] datainі, datainq;
input strobe_in;
output reg [15:0] dataouti, dataoutq;
//output reg strobe_out;
input strobe_out;

wire [1583:0] result;
reg [1583:0] sr;
wire [15:0] sum;
reg [31:0] timestamp;
reg [31:0] clkcounter;
reg [15:0] regsum;

integer handle;

//typedef enum {comp, sendstamp1, sendstamp2, sendstamp3, sendstamp4}
state_type;
parameter comp = 0;
parameter header = 1;
parameter sendstamp1 = 2;
parameter sendstamp2 = 3;
parameter sendstamp3 = 4;
parameter sendstamp4 = 5;
parameter finish = 6;
parameter tail = 7;
//state_type state;
reg [2:0] state;

wire [1583:0] ref;
```

```
//initial
//begin
//  handle = $fopen("output.txt");
//end

assign result = sr ^ ref;
assign sum = result[0] + result[1] + result[2] + result[3] + result[4]
+ result[5] + result[6] + result[7] + result[8] + result[9] + result[10]
+ result[11] + result[12] + result[13] + result[14] + result[15] +
result[16] + result[17] + result[18] + result[19] + result[20] +
result[21] + result[22] + result[23] + result[24] + result[25] +
result[26] + result[27] + result[28] + result[29] + result[30] +
result[31] + result[32] + result[33] + result[34] + result[35] +
result[36] + result[37] + result[38] + result[39] + result[40] +
result[41] + result[42] + result[43] + result[44] + result[45] +
result[46] + result[47] + result[48] + result[49] + result[50] +
result[51] + result[52] + result[53] + result[54] + result[55] +
result[56] + result[57] + result[58] + result[59] + result[60] +
result[61] + result[62] + result[63] + result[64] + result[65] +
result[66] + result[67] + result[68] + result[69] + result[70] +
result[71] + result[72] + result[73] + result[74] + result[75] +
result[76] + result[77] + result[78] + result[79] + result[80] +
result[81] + result[82] + result[83] + result[84] + result[85] +
result[86] + result[87] + result[88] + result[89] + result[90] +
result[91] + result[92] + result[93] + result[94] + result[95] +
result[96] + result[97] + result[98] + result[99] + result[100] +
result[101] + result[102] + result[103] + result[104] + result[105] +
result[106] + result[107] + result[108] + result[109] + result[110] +
result[111] + result[112] + result[113] + result[114] + result[115] +
result[116] + result[117] + result[118] + result[119] + result[120] +
result[121] + result[122] + result[123] + result[124] + result[125] +
result[126] + result[127] + result[128] + result[129] + result[130] +
result[131] + result[132] + result[133] + result[134] + result[135] +
result[136] + result[137] + result[138] + result[139] + result[140] +
result[141] + result[142] + result[143] + result[144] + result[145] +
```

```
result[146] + result[147] + result[148] + result[149] + result[150] +
result[151] + result[152] + result[153] + result[154] + result[155] +
result[156] + result[157] + result[158] + result[159] + result[160] +
result[161] + result[162] + result[163] + result[164] + result[165] +
result[166] + result[167] + result[168] + result[169] + result[170] +
result[171] + result[172] + result[173] + result[174] + result[175] +
result[176] + result[177] + result[178] + result[179] + result[180] +
result[181] + result[182] + result[183] + result[184] + result[185] +
result[186] + result[187] + result[188] + result[189] + result[190] +
result[191] + result[192] + result[193] + result[194] + result[195] +
result[196] + result[197] + result[198] + result[199] + result[200] +
result[201] + result[202] + result[203] + result[204] + result[205] +
result[206] + result[207] + result[208] + result[209] + result[210] +
result[211] + result[212] + result[213] + result[214] + result[215] +
result[216] + result[217] + result[218] + result[219] + result[220] +
result[221] + result[222] + result[223] + result[224] + result[225] +
result[226] + result[227] + result[228] + result[229] + result[230] +
result[231] + result[232] + result[233] + result[234] + result[235] +
result[236] + result[237] + result[238] + result[239] + result[240] +
result[241] + result[242] + result[243] + result[244] + result[245] +
result[246] + result[247] + result[248] + result[249] + result[250] +
result[251] + result[252] + result[253] + result[254] + result[255] +
result[256] + result[257] + result[258] + result[259] + result[260] +
result[261] + result[262] + result[263] + result[264] + result[265] +
result[266] + result[267] + result[268] + result[269] + result[270] +
result[271] + result[272] + result[273] + result[274] + result[275] +
result[276] + result[277] + result[278] + result[279] + result[280] +
result[281] + result[282] + result[283] + result[284] + result[285] +
result[286] + result[287] + result[288] + result[289] + result[290] +
result[291] + result[292] + result[293] + result[294] + result[295] +
result[296] + result[297] + result[298] + result[299] + result[300] +
result[301] + result[302] + result[303] + result[304] + result[305] +
result[306] + result[307] + result[308] + result[309] + result[310] +
result[311] + result[312] + result[313] + result[314] + result[315] +
result[316] + result[317] + result[318] + result[319] + result[320] +
result[321] + result[322] + result[323] + result[324] + result[325] +
```

```
result[326] + result[327] + result[328] + result[329] + result[330] +
result[331] + result[332] + result[333] + result[334] + result[335] +
result[336] + result[337] + result[338] + result[339] + result[340] +
result[341] + result[342] + result[343] + result[344] + result[345] +
result[346] + result[347] + result[348] + result[349] + result[350] +
result[351] + result[352] + result[353] + result[354] + result[355] +
result[356] + result[357] + result[358] + result[359] + result[360] +
result[361] + result[362] + result[363] + result[364] + result[365] +
result[366] + result[367] + result[368] + result[369] + result[370] +
result[371] + result[372] + result[373] + result[374] + result[375] +
result[376] + result[377] + result[378] + result[379] + result[380] +
result[381] + result[382] + result[383] + result[384] + result[385] +
result[386] + result[387] + result[388] + result[389] + result[390] +
result[391] + result[392] + result[393] + result[394] + result[395] +
result[396] + result[397] + result[398] + result[399] + result[400] +
result[401] + result[402] + result[403] + result[404] + result[405] +
result[406] + result[407] + result[408] + result[409] + result[410] +
result[411] + result[412] + result[413] + result[414] + result[415] +
result[416] + result[417] + result[418] + result[419] + result[420] +
result[421] + result[422] + result[423] + result[424] + result[425] +
result[426] + result[427] + result[428] + result[429] + result[430] +
result[431] + result[432] + result[433] + result[434] + result[435] +
result[436] + result[437] + result[438] + result[439] + result[440] +
result[441] + result[442] + result[443] + result[444] + result[445] +
result[446] + result[447] + result[448] + result[449] + result[450] +
result[451] + result[452] + result[453] + result[454] + result[455] +
result[456] + result[457] + result[458] + result[459] + result[460] +
result[461] + result[462] + result[463] + result[464] + result[465] +
result[466] + result[467] + result[468] + result[469] + result[470] +
result[471] + result[472] + result[473] + result[474] + result[475] +
result[476] + result[477] + result[478] + result[479] + result[480] +
result[481] + result[482] + result[483] + result[484] + result[485] +
result[486] + result[487] + result[488] + result[489] + result[490] +
result[491] + result[492] + result[493] + result[494] + result[495] +
result[496] + result[497] + result[498] + result[499] + result[500] +
result[501] + result[502] + result[503] + result[504] + result[505] +
```

```
result[506] + result[507] + result[508] + result[509] + result[510] +
result[511] + result[512] + result[513] + result[514] + result[515] +
result[516] + result[517] + result[518] + result[519] + result[520] +
result[521] + result[522] + result[523] + result[524] + result[525] +
result[526] + result[527] + result[528] + result[529] + result[530] +
result[531] + result[532] + result[533] + result[534] + result[535] +
result[536] + result[537] + result[538] + result[539] + result[540] +
result[541] + result[542] + result[543] + result[544] + result[545] +
result[546] + result[547] + result[548] + result[549] + result[550] +
result[551] + result[552] + result[553] + result[554] + result[555] +
result[556] + result[557] + result[558] + result[559] + result[560] +
result[561] + result[562] + result[563] + result[564] + result[565] +
result[566] + result[567] + result[568] + result[569] + result[570] +
result[571] + result[572] + result[573] + result[574] + result[575] +
result[576] + result[577] + result[578] + result[579] + result[580] +
result[581] + result[582] + result[583] + result[584] + result[585] +
result[586] + result[587] + result[588] + result[589] + result[590] +
result[591] + result[592] + result[593] + result[594] + result[595] +
result[596] + result[597] + result[598] + result[599] + result[600] +
result[601] + result[602] + result[603] + result[604] + result[605] +
result[606] + result[607] + result[608] + result[609] + result[610] +
result[611] + result[612] + result[613] + result[614] + result[615] +
result[616] + result[617] + result[618] + result[619] + result[620] +
result[621] + result[622] + result[623] + result[624] + result[625] +
result[626] + result[627] + result[628] + result[629] + result[630] +
result[631] + result[632] + result[633] + result[634] + result[635] +
result[636] + result[637] + result[638] + result[639] + result[640] +
result[641] + result[642] + result[643] + result[644] + result[645] +
result[646] + result[647] + result[648] + result[649] + result[650] +
result[651] + result[652] + result[653] + result[654] + result[655] +
result[656] + result[657] + result[658] + result[659] + result[660] +
result[661] + result[662] + result[663] + result[664] + result[665] +
result[666] + result[667] + result[668] + result[669] + result[670] +
result[671] + result[672] + result[673] + result[674] + result[675] +
result[676] + result[677] + result[678] + result[679] + result[680] +
result[681] + result[682] + result[683] + result[684] + result[685] +
```

```
result[686] + result[687] + result[688] + result[689] + result[690] +
result[691] + result[692] + result[693] + result[694] + result[695] +
result[696] + result[697] + result[698] + result[699] + result[700] +
result[701] + result[702] + result[703] + result[704] + result[705] +
result[706] + result[707] + result[708] + result[709] + result[710] +
result[711] + result[712] + result[713] + result[714] + result[715] +
result[716] + result[717] + result[718] + result[719] + result[720] +
result[721] + result[722] + result[723] + result[724] + result[725] +
result[726] + result[727] + result[728] + result[729] + result[730] +
result[731] + result[732] + result[733] + result[734] + result[735] +
result[736] + result[737] + result[738] + result[739] + result[740] +
result[741] + result[742] + result[743] + result[744] + result[745] +
result[746] + result[747] + result[748] + result[749] + result[750] +
result[751] + result[752] + result[753] + result[754] + result[755] +
result[756] + result[757] + result[758] + result[759] + result[760] +
result[761] + result[762] + result[763] + result[764] + result[765] +
result[766] + result[767] + result[768] + result[769] + result[770] +
result[771] + result[772] + result[773] + result[774] + result[775] +
result[776] + result[777] + result[778] + result[779] + result[780] +
result[781] + result[782] + result[783] + result[784] + result[785] +
result[786] + result[787] + result[788] + result[789] + result[790] +
result[791] + result[792] + result[793] + result[794] + result[795] +
result[796] + result[797] + result[798] + result[799] + result[800] +
result[801] + result[802] + result[803] + result[804] + result[805] +
result[806] + result[807] + result[808] + result[809] + result[810] +
result[811] + result[812] + result[813] + result[814] + result[815] +
result[816] + result[817] + result[818] + result[819] + result[820] +
result[821] + result[822] + result[823] + result[824] + result[825] +
result[826] + result[827] + result[828] + result[829] + result[830] +
result[831] + result[832] + result[833] + result[834] + result[835] +
result[836] + result[837] + result[838] + result[839] + result[840] +
result[841] + result[842] + result[843] + result[844] + result[845] +
result[846] + result[847] + result[848] + result[849] + result[850] +
result[851] + result[852] + result[853] + result[854] + result[855] +
result[856] + result[857] + result[858] + result[859] + result[860] +
result[861] + result[862] + result[863] + result[864] + result[865] +
```

```
result[866] + result[867] + result[868] + result[869] + result[870] +
result[871] + result[872] + result[873] + result[874] + result[875] +
result[876] + result[877] + result[878] + result[879] + result[880] +
result[881] + result[882] + result[883] + result[884] + result[885] +
result[886] + result[887] + result[888] + result[889] + result[890] +
result[891] + result[892] + result[893] + result[894] + result[895] +
result[896] + result[897] + result[898] + result[899] + result[900] +
result[901] + result[902] + result[903] + result[904] + result[905] +
result[906] + result[907] + result[908] + result[909] + result[910] +
result[911] + result[912] + result[913] + result[914] + result[915] +
result[916] + result[917] + result[918] + result[919] + result[920] +
result[921] + result[922] + result[923] + result[924] + result[925] +
result[926] + result[927] + result[928] + result[929] + result[930] +
result[931] + result[932] + result[933] + result[934] + result[935] +
result[936] + result[937] + result[938] + result[939] + result[940] +
result[941] + result[942] + result[943] + result[944] + result[945] +
result[946] + result[947] + result[948] + result[949] + result[950] +
result[951] + result[952] + result[953] + result[954] + result[955] +
result[956] + result[957] + result[958] + result[959] + result[960] +
result[961] + result[962] + result[963] + result[964] + result[965] +
result[966] + result[967] + result[968] + result[969] + result[970] +
result[971] + result[972] + result[973] + result[974] + result[975] +
result[976] + result[977] + result[978] + result[979] + result[980] +
result[981] + result[982] + result[983] + result[984] + result[985] +
result[986] + result[987] + result[988] + result[989] + result[990] +
result[991] + result[992] + result[993] + result[994] + result[995] +
result[996] + result[997] + result[998] + result[999] + result[1000] +
result[1001] + result[1002] + result[1003] + result[1004] + result[1005]
+ result[1006] + result[1007] + result[1008] + result[1009] +
result[1010] + result[1011] + result[1012] + result[1013] + result[1014]
+ result[1015] + result[1016] + result[1017] + result[1018] +
result[1019] + result[1020] + result[1021] + result[1022] + result[1023]
+ result[1024] + result[1025] + result[1026] + result[1027] +
result[1028] + result[1029] + result[1030] + result[1031] + result[1032]
+ result[1033] + result[1034] + result[1035] + result[1036] +
result[1037] + result[1038] + result[1039] + result[1040] + result[1041]
```

```
+ result[1042] + result[1043] + result[1044] + result[1045] +
result[1046] + result[1047] + result[1048] + result[1049] + result[1050]
+ result[1051] + result[1052] + result[1053] + result[1054] +
result[1055] + result[1056] + result[1057] + result[1058] + result[1059]
+ result[1060] + result[1061] + result[1062] + result[1063] +
result[1064] + result[1065] + result[1066] + result[1067] + result[1068]
+ result[1069] + result[1070] + result[1071] + result[1072] +
result[1073] + result[1074] + result[1075] + result[1076] + result[1077]
+ result[1078] + result[1079] + result[1080] + result[1081] +
result[1082] + result[1083] + result[1084] + result[1085] + result[1086]
+ result[1087] + result[1088] + result[1089] + result[1090] +
result[1091] + result[1092] + result[1093] + result[1094] + result[1095]
+ result[1096] + result[1097] + result[1098] + result[1099] +
result[1100] + result[1101] + result[1102] + result[1103] + result[1104]
+ result[1105] + result[1106] + result[1107] + result[1108] +
result[1109] + result[1110] + result[1111] + result[1112] + result[1113]
+ result[1114] + result[1115] + result[1116] + result[1117] +
result[1118] + result[1119] + result[1120] + result[1121] + result[1122]
+ result[1123] + result[1124] + result[1125] + result[1126] +
result[1127] + result[1128] + result[1129] + result[1130] + result[1131]
+ result[1132] + result[1133] + result[1134] + result[1135] +
result[1136] + result[1137] + result[1138] + result[1139] + result[1140]
+ result[1141] + result[1142] + result[1143] + result[1144] +
result[1145] + result[1146] + result[1147] + result[1148] + result[1149]
+ result[1150] + result[1151] + result[1152] + result[1153] +
result[1154] + result[1155] + result[1156] + result[1157] + result[1158]
+ result[1159] + result[1160] + result[1161] + result[1162] +
result[1163] + result[1164] + result[1165] + result[1166] + result[1167]
+ result[1168] + result[1169] + result[1170] + result[1171] +
result[1172] + result[1173] + result[1174] + result[1175] + result[1176]
+ result[1177] + result[1178] + result[1179] + result[1180] +
result[1181] + result[1182] + result[1183] + result[1184] + result[1185]
+ result[1186] + result[1187] + result[1188] + result[1189] +
result[1190] + result[1191] + result[1192] + result[1193] + result[1194]
+ result[1195] + result[1196] + result[1197] + result[1198] +
result[1199] + result[1200] + result[1201] + result[1202] + result[1203]
```

```
+ result[1204] + result[1205] + result[1206] + result[1207] +
result[1208] + result[1209] + result[1210] + result[1211] + result[1212]
+ result[1213] + result[1214] + result[1215] + result[1216] +
result[1217] + result[1218] + result[1219] + result[1220] + result[1221]
+ result[1222] + result[1223] + result[1224] + result[1225] +
result[1226] + result[1227] + result[1228] + result[1229] + result[1230]
+ result[1231] + result[1232] + result[1233] + result[1234] +
result[1235] + result[1236] + result[1237] + result[1238] + result[1239]
+ result[1240] + result[1241] + result[1242] + result[1243] +
result[1244] + result[1245] + result[1246] + result[1247] + result[1248]
+ result[1249] + result[1250] + result[1251] + result[1252] +
result[1253] + result[1254] + result[1255] + result[1256] + result[1257]
+ result[1258] + result[1259] + result[1260] + result[1261] +
result[1262] + result[1263] + result[1264] + result[1265] + result[1266]
+ result[1267] + result[1268] + result[1269] + result[1270] +
result[1271] + result[1272] + result[1273] + result[1274] + result[1275]
+ result[1276] + result[1277] + result[1278] + result[1279] +
result[1280] + result[1281] + result[1282] + result[1283] + result[1284]
+ result[1285] + result[1286] + result[1287] + result[1288] +
result[1289] + result[1290] + result[1291] + result[1292] + result[1293]
+ result[1294] + result[1295] + result[1296] + result[1297] +
result[1298] + result[1299] + result[1300] + result[1301] + result[1302]
+ result[1303] + result[1304] + result[1305] + result[1306] +
result[1307] + result[1308] + result[1309] + result[1310] + result[1311]
+ result[1312] + result[1313] + result[1314] + result[1315] +
result[1316] + result[1317] + result[1318] + result[1319] + result[1320]
+ result[1321] + result[1322] + result[1323] + result[1324] +
result[1325] + result[1326] + result[1327] + result[1328] + result[1329]
+ result[1330] + result[1331] + result[1332] + result[1333] +
result[1334] + result[1335] + result[1336] + result[1337] + result[1338]
+ result[1339] + result[1340] + result[1341] + result[1342] +
result[1343] + result[1344] + result[1345] + result[1346] + result[1347]
+ result[1348] + result[1349] + result[1350] + result[1351] +
result[1352] + result[1353] + result[1354] + result[1355] + result[1356]
+ result[1357] + result[1358] + result[1359] + result[1360] +
result[1361] + result[1362] + result[1363] + result[1364] + result[1365]
```

```
+ result[1366] + result[1367] + result[1368] + result[1369] +
result[1370] + result[1371] + result[1372] + result[1373] + result[1374]
+ result[1375] + result[1376] + result[1377] + result[1378] +
result[1379] + result[1380] + result[1381] + result[1382] + result[1383]
+ result[1384] + result[1385] + result[1386] + result[1387] +
result[1388] + result[1389] + result[1390] + result[1391] + result[1392]
+ result[1393] + result[1394] + result[1395] + result[1396] +
result[1397] + result[1398] + result[1399] + result[1400] + result[1401]
+ result[1402] + result[1403] + result[1404] + result[1405] +
result[1406] + result[1407] + result[1408] + result[1409] + result[1410]
+ result[1411] + result[1412] + result[1413] + result[1414] +
result[1415] + result[1416] + result[1417] + result[1418] + result[1419]
+ result[1420] + result[1421] + result[1422] + result[1423] +
result[1424] + result[1425] + result[1426] + result[1427] + result[1428]
+ result[1429] + result[1430] + result[1431] + result[1432] +
result[1433] + result[1434] + result[1435] + result[1436] + result[1437]
+ result[1438] + result[1439] + result[1440] + result[1441] +
result[1442] + result[1443] + result[1444] + result[1445] + result[1446]
+ result[1447] + result[1448] + result[1449] + result[1450] +
result[1451] + result[1452] + result[1453] + result[1454] + result[1455]
+ result[1456] + result[1457] + result[1458] + result[1459] +
result[1460] + result[1461] + result[1462] + result[1463] + result[1464]
+ result[1465] + result[1466] + result[1467] + result[1468] +
result[1469] + result[1470] + result[1471] + result[1472] + result[1473]
+ result[1474] + result[1475] + result[1476] + result[1477] +
result[1478] + result[1479] + result[1480] + result[1481] + result[1482]
+ result[1483] + result[1484] + result[1485] + result[1486] +
result[1487] + result[1488] + result[1489] + result[1490] + result[1491]
+ result[1492] + result[1493] + result[1494] + result[1495] +
result[1496] + result[1497] + result[1498] + result[1499] + result[1500]
+ result[1501] + result[1502] + result[1503] + result[1504] +
result[1505] + result[1506] + result[1507] + result[1508] + result[1509]
+ result[1510] + result[1511] + result[1512] + result[1513] +
result[1514] + result[1515] + result[1516] + result[1517] + result[1518]
+ result[1519] + result[1520] + result[1521] + result[1522] +
result[1523] + result[1524] + result[1525] + result[1526] + result[1527]
```

```
+ result[1528] + result[1529] + result[1530] + result[1531] +
result[1532] + result[1533] + result[1534] + result[1535] + result[1536]
+ result[1537] + result[1538] + result[1539] + result[1540] +
result[1541] + result[1542] + result[1543] + result[1544] + result[1545]
+ result[1546] + result[1547] + result[1548] + result[1549] +
result[1550] + result[1551] + result[1552] + result[1553] + result[1554]
+ result[1555] + result[1556] + result[1557] + result[1558] +
result[1559] + result[1560] + result[1561] + result[1562] + result[1563]
+ result[1564] + result[1565] + result[1566] + result[1567] +
result[1568] + result[1569] + result[1570] + result[1571] + result[1572]
+ result[1573] + result[1574] + result[1575] + result[1576] +
result[1577] + result[1578] + result[1579] + result[1580] + result[1581]
+ result[1582] + result[1583] + result[1583];
assign ref[0] = 0;
assign ref[1] = 1;
assign ref[2] = 0;
assign ref[3] = 0;
assign ref[4] = 1;
assign ref[5] = 0;
assign ref[6] = 0;
assign ref[7] = 0;
assign ref[8] = 1;
…
…
assign ref[1580] = 1;
assign ref[1581] = 0;
assign ref[1582] = 0;
assign ref[1583] = 0;

always @(posedge reset or posedge clk)
begin
  if(reset == 1'b1)
  begin
    sr <= 0;
    clkcounter <= 0;
  end
```

```
  else
  begin
    clkcounter <= clkcounter + 1;
    if(strobe_in == 1'b1)
    begin
      sr <= {dataini[15], sr[1583:1]};
//      $fdisplay(handle,"%d\n",sum);
    end
  end

end

always@(posedge reset or posedge clk)
begin
  if(reset == 1'b1)
  begin
    state <= comp;
  end
  else
  begin
    case(state)
      comp:
      begin
        if((sum < 1600 && sum > 1000)||(sum < 400 && sum > 0))
        begin
          timestamp <= clkcounter;
          state <= header;
          regsum <= sum;
        end
        // output all '0's every 16 cycles if no header found
        // we do this to flush the usb buffer
        else if (strobe_out == 1'b1)
        begin
          dataouti <= 16'h0000;
          dataoutq <= 16'h0000;
        end
```

```
      end

        header:
        begin
            if(strobe_out == 1'b1)
            begin
              state <= sendstamp1;
              dataouti <= 16'hf0f1;
              dataoutq <= 16'hf0f2;
            end
        end

      sendstamp1:
      // 8 cycles per data, hope this can make the stream synchronize
with usb clk
      begin
        if (strobe_out == 1'b1)
        begin
          state <= sendstamp2;
          dataouti <= timestamp[15:0];
          dataoutq <= 16'ha0a1;
        end
      end

      sendstamp2:
      begin
        if (strobe_out == 1'b1)
        begin
          state <= sendstamp3;
          dataouti <= timestamp[31:16];
          dataoutq <= 16'hb0b1;
        end
      end

      sendstamp3:
      begin
```

```
        if (strobe_out == 1'b1)
        begin
          state <= sendstamp4;
          dataouti <= 16'h1f0f;
          dataoutq <= 16'hc0c1;
        end
      end

      sendstamp4:
      begin
        if (strobe_out == 1'b1)
        begin
          state <= tail;
          dataouti <=regsum;
          dataoutq <=16'hc0c1;
        end
      end

//      sendstamp5:
//      begin
//        strobe_out <= 1'b0;
//        state <=tail;
//      end

      tail:
      begin
        if (strobe_out == 1'b1)
        begin
          state <= comp;
          dataouti <= 16'haf0f;
          dataoutq <= 16'hd0d1;
        end
      end
    endcase
  end
end
```

```
endmodule
```

TRITA-ICT-EX-2009:209

www.kth.se