A High Speed Image Transmission System for Ultra-Wideband Wireless Links

by

Helen He Liang

B.S. Massachusetts Institute of Technology (2008)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2009

Author .....
Department of Electrical Engineering and Computer Science
May 8, 2009

Certified by .....
Anantha Chandrakasan
Joseph F. and Nancy P. Keithley Professor of Electrical Engineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

A High Speed Image Transmission System for Ultra-Wideband Wireless Links

by

Helen He Liang

Submitted to the Department of Electrical Engineering and Computer Science
on May 8, 2009, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Ultra-wideband (UWB) communication is an emerging technology that offers short range, high data rate wireless transmission, with low power consumption and low consumer cost. Operating in the 3.1 GHz - 10.6 GHz frequency band with bandwidth above 500 MHz, it is an overlay technology that can co-exist with other narrowband services in the same frequency range, thus alleviating the problem of over-crowded spectrum. In particular, pulse-based UWB technologies allows for building of ultra-low power, medium- to long-range transceivers, at the expense of data transmission rate.

This thesis presents a pulse-based, non-coherent UWB wireless image transmission platform. The system features a one-way wireless link. On the transmitter side, a host PC processes the images into transmittable packets in MATLAB, and sends them to the UWB radio through an interfacing FPGA module. On the receiver side, the UWB receiver radio receivers the packets, decodes the bits, and passes them back to the receiver host PC through another interfacing FPGA module. The receiver host PC collects the decoded bits and reconstructs the original image in MATLAB. The unidirectional wireless channel is complemented by a feedback path, provided through internet connection between the two host PCs. To improve usability, graphical user interfaces are setup on both host PCs. The overall system transmits 120×160 uncompressed bitmap images. It achieves a maximum payload data rate of 8 Mb/s. It is able to transmit data reliably, with above 95% packet reception rate and below 2×10^{-5} bit error rate, for distances up to 16 meters. At 16 meters, the system has a maximum transmission data rate of 2.67 Mbps.

Thesis Supervisor: Anantha Chandrakasan
Title: Joseph F. and Nancy P. Keithley Professor of Electrical Engineering

Acknowledgments

First and foremost, I'd like to thank my thesis advisor Prof. Anantha Chandrakasan for giving me a chance to work on this exciting and challenging project. I have gained a lot of useful knowledge and valuable experience in the past two years. Thank you very much for your guidance and support.

For the past two years, I have also met a lot really great people in *ananthagroup*. They are among the smartest and most hardworking people I know. Among all the people in *ananthagroup*, the person I owe most thanks to is Denis Daly. Thank you for all your advice, patience and encouragement. You have taught me a lot, and this thesis would not be possible without your help and guidance. I also would like to thank the rest of UWB team, Patrick Mercier and Manish Bhardwaj, for all the help they have given me. I learned a lot from you guys, and I truly enjoyed working and talking with you.

I'd also like to give out big hugs to all my friends here at MIT. You guys have made my years in college unforgettable. Thank you for all the fun moments, for all the amazing memories, for always being there for me, and for putting up with all my whining and complaining. In particular, I would like to thank Chun and Tiff for being such wonderful roommates. Thanks to everyone in the 'elevator suite' for all those games of stupid. Thanks to Patricia, my 'cult leader', for all the laughters and sleep-overs. Thanks to Wendi, my best friend in MIT, for all the trip we took, for all the interesting conversations we had, and for all the time we spent together in class and in lab.

Last but not least, I would like to thank my parents for all the love and support they have given me. Dad, thank you for all the trust and hope you have in me. You have more faith in me and my abilities than I do myself, and that really means a lot for me. Mom, thank you for always being there when I need you, sharing my joy and my pain. I couldn't have made it this far if it wasn't for your love, guidance, and support. Thank you for everything you've done for me in the past 23 years. You are the greatest mom, and I really couldn't have wished for anyone better.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Technology Overview

Ultra-Wideband (UWB) radio technology, traditionally known as impulse radio, traces its root back to the spark gap radio in the late 19th century [1]. Over the years, development of other wireless transmission technologies, such as narrowband radio, replaced the traditional spark gap radio, becoming the dominant force in the field of wireless communications. However, recently, researchers revisited the idea of UWB radio technology. The field quickly gained momentum, for UWB promises short-range wireless communication at high data rate, with low power consumption and low consumer cost. In 2002, the Federal Communications Commissions (FCC) approved the unlicensed use of UWB communications in the United States. The FCC specified that the UWB signals must operate below 960 MHz or within the 3.1-to-10.6 GHz frequency band, with a -10 dB bandwidth exceeds the lesser of 20% of the center frequency or 500 MHz [2]. Since this approval, companies and research institutions alike have been racing to build UWB wireless devices and systems. UWB technology can be applied to wireless communications, networking, radar, imaging, and as precise positioning systems. In particular, it brings new promise to the development of wireless personal area networks (WPAN). Devices powered by UWB technology can be built cheaply, be highly portable with long battery life, and offer high data transfer rate at a low cost.

UWB is an overlay technology. It does not interfere with existing narrowband channels in the same frequency spectrum. FCC restricts the maximum average power spectral density of the UWB signals to be below -41.3 dBm/MHz in the 3.1-to-10.6 GHz band. The FCC further reduces the average power spectral density of UWB signals in other frequency ranges, due to concerns with interference with other low signal-to-noise ratio devices, such as the global positioning system (GPS), as seen in Figure 1-1. Furthermore, the peak power of UWB signals are limited to 0dBm within a 50 MHz bandwidth window centered around the center frequency.

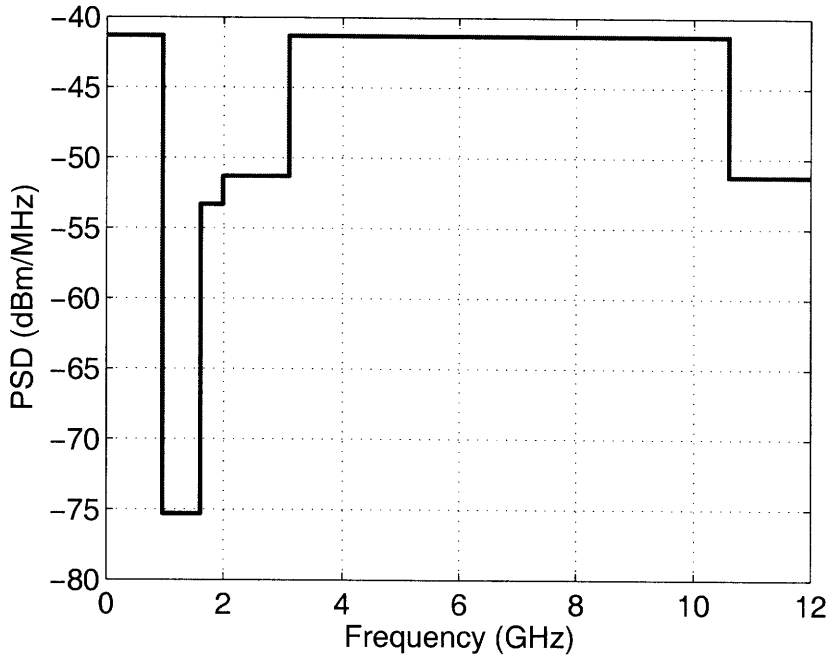


Figure 1-1: FCC Mask Restricting Average Power Spectral Densities from 0 to 12 GHz for Indoor UWB Systems

The limit on transmit power of the UWB singals allows UWB transmissions to hide among the environment noises, and avoid interfering with other narrowband transmission services sharing the same frequency spectrum, as shown in Figure 1-2. This allows UWB signals to easily share the same frequency band with other existing wireless technologies, thus alleviating the problem of scarce frequency spectrum.

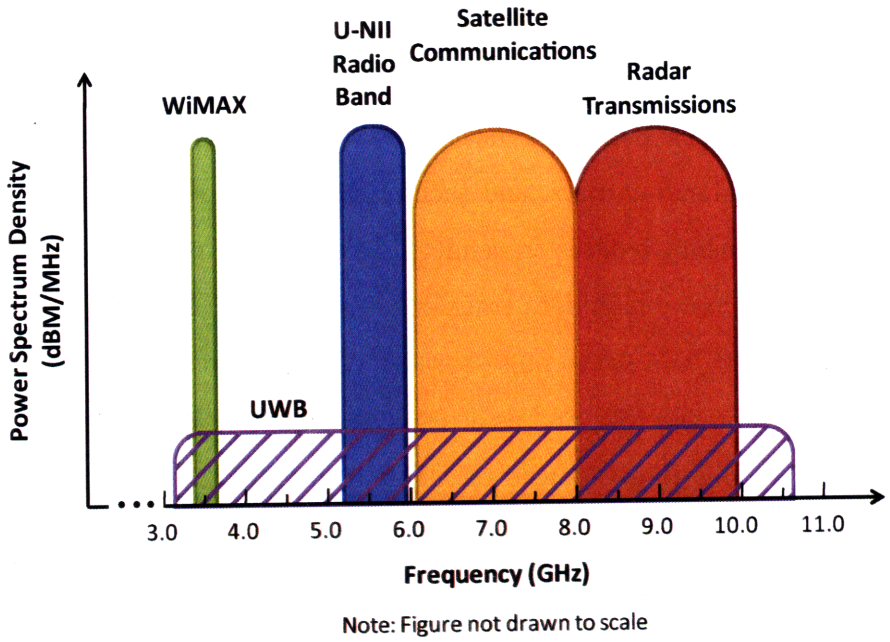


Figure 1-2: Comparison of UWB and other Wireless Technologies in the Same Frequency Range

1.2 Existing UWB Standards and Communication Schemes

After the FCC approval of UWB signals, the Institute of Electrical and Electronics Engineers (IEEE) 802.15.3a task group was formed to draw up a standardization proposal for high data rate UWB communication. The group was disbanded in 2006, after failing to reach an agreement between two distinct proposals: one uses Orthogonal Frequency Division Multiplexing (OFDM), and the other uses pulse-based signaling [3]. The two proposals sought standardization elsewhere. The OFDM-based technology is standardized by the International Standard Association (ISO), and is currently pioneered by the WiMedia Alliances, the Bluetooth Special Interest Group, Wireless USB Promoter Group, and the USB Implementers Forum. Pulsed-based UWB technologies were adopted by another IEEE task group, 802.12.4a. This technology is now a part of the IEEE 802.12.4 standard.

1.2.1 OFDM

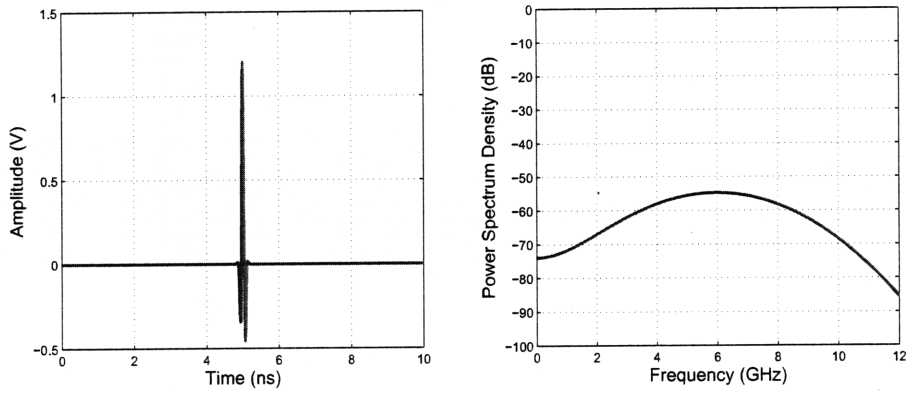
OFDM is a standard technique used in narrowband and spread spectrum wireless transmission protocols, such as IEEE 802.11a/g/n, and WiMAX. It divides a frequency band into closely spaced orthogonal sub-carriers, and each of these sub-carriers is then modulated by data and the entire frequency band is transmitted at once. The wireless USB standard uses OFDM modulation to achieve high data transfer rates up to 480 Mb/s at short ranges of up to 10 meters [4]. However, relative to pulse-based UWB, this scheme increases the complexity of the transmitter and receiver, and as a result increases the power consumption rate of the UWB system [5]. This thesis focuses on UWB transmission systems using pulse-based signaling, thus, the OFDM-based UWB technologies will not be discussed in more detail.

1.2.2 Pulse-Based UWB

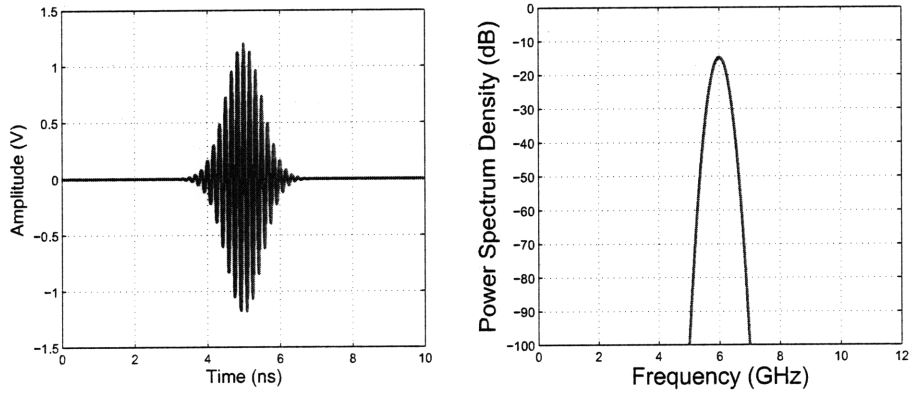
Unlike OFDM, pulse-based transmission scheme utilizes the idea of the original impulse radio, generating sequences of very narrow pulses to encode information. These pulses can be as short as fractions of nanoseconds. Figure 1-3 shows some ultra-wideband pulses, and their respective frequency responses simulated in MATLAB. As shown in Figure 1-3(a), these short pulses have corresponding frequency responses that are ultra-wideband in nature. Thus, the UWB transmitter does not require additional modulation to bring the signal to the desired frequency bands. Furthermore, multiple pulses can be sent back to back in a burst, as shown in Figure 1-3(b), to combat noise and interferences in the transmission channel [6].

Compared to OFDM, the pulse-based UWB scheme has lower maximum data throughput[3]. However, the pulse-based UWB also decreases the complexity of the transmitter and receiver, thereby reducing the power consumption of the resulting transmission system [6]. This makes pulse-based UWB scheme ideal for portable applications, where battery life is the key design constraint.

Pulse-based UWB is now a part of the IEEE 802.15.4 standard, through the 802.15.4a amendment. IEEE 802.15.4 is a standard that specifies physical layer and media access control for low rate WPAN devices. IEEE 802.15.4a splits the UWB frequency spectrum



(a)



(b)

Figure 1-3: (a) 2 ns UWB pulse in time domain and frequency domain. (b) Multiple 2 ns UWB pulse in time domain and frequency domain.

into three main groups: sub-GHz, low-band, and high-band. The sub-GHz band contains one channel, with center frequency of 499.2 GHz. The low-band contains four channels, spanning frequencies from 3 to 5 GHz. The high-band contains eleven channels, spanning frequencies from 6 to 10 GHz [7]. Figure 1-4 shows the frequency plan for the low- and high-bands of 802.15.4a. The devices operating in the low- and high-bands must support one mandatory channel, while the rest of the channels in the band are optional. The 5 to 6 GHz frequency band is avoided in the 802.15.4 standard, to avoid interferences with the 802.11a and U-NII band.

In literature, there are four common ways to encode information using pulse-based signal-

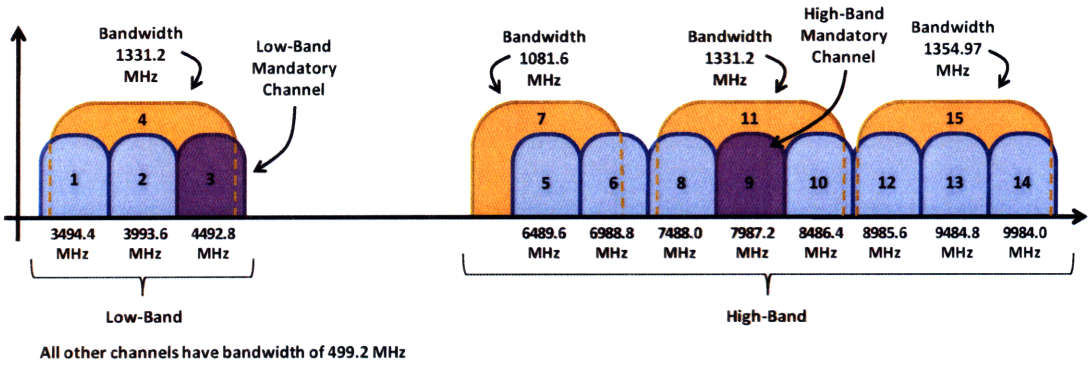


Figure 1-4: Frequency Plan for the IEEE 802.15.4a Pulse-Based UWB Technology

ing. They are On-Off Keying (OOK), Pulse Position Modulation (PPM), Pulse Amplitude Modulation (PAM), and Binary Phase Shift Keying (BPSK) [1]. These modulation schemes uses the time position, phase, or the shape of the pulses generated by the UWB transmitter to encode data. These modulation schemes are briefly described below.

OOK

OOK is a very simple modulation scheme, shown in Figure 1-5. The time domain is divided into equally spaced windows, and the presence of pulse energies in a time window corresponds to a '1', while the absence of pulses in a time window corresponds to a '0'.

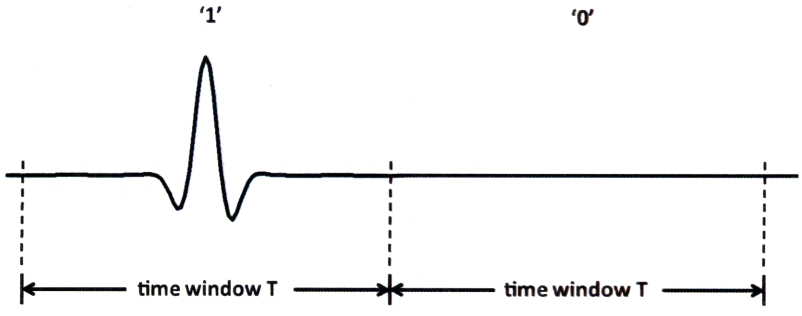


Figure 1-5: On-Off Keying Modulation Scheme

PPM

PPM uses two or more time positions to encode data. As shown in Figure 1-6, the time domain is divided into equally spaced windows T, and each window contains two slots, T_1 and T_2: the presence of pulse energies in T_1, the first slot of a time window corresponds to a '0'; the presence of pulse energies in T_2, the second slot of a time window corresponds to a '1'. Each window cans also be made to contain more than two slots so multiple bits can be encoded per sample [1]. This scheme is more immune to noise compared to the OOK scheme. Instead of comparing the pulse energy in a time slot to a certain threshold value, the PPM makes comparisons between adjacent time slots, and the bit values are determined depending on the result of the comparison. For single bit encoding, the data rate for the PPM modulation is half that of OOK modulation.

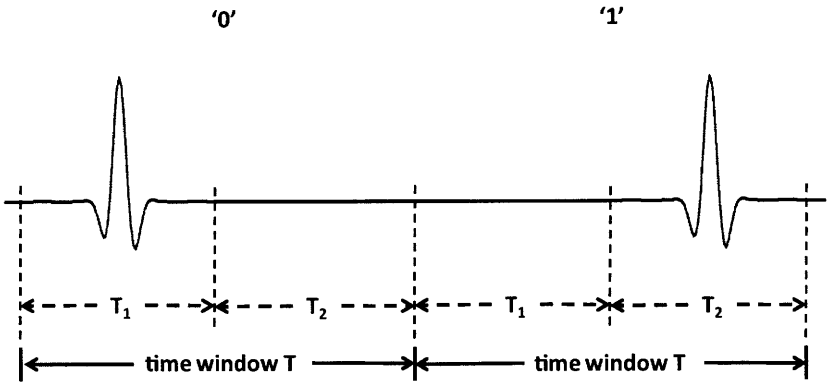


Figure 1-6: Pulse Position Modulation

PAM and BPSK

PAM uses the amplitude of the pulse sequences to encode information, as shown in Figure 1-7. The figure shows a two level modulation scheme, where each pulse provides one bit of information. However, multiple amplitude values can be used to encode more bits per pulse.

BPSK uses the polarity of the UWB pulse to encode information, as shown in Figure 1-8. One polarity indicates a '0' and the other polarity indicates a '1'. With this scheme, only one bit per pulse can be encoded.

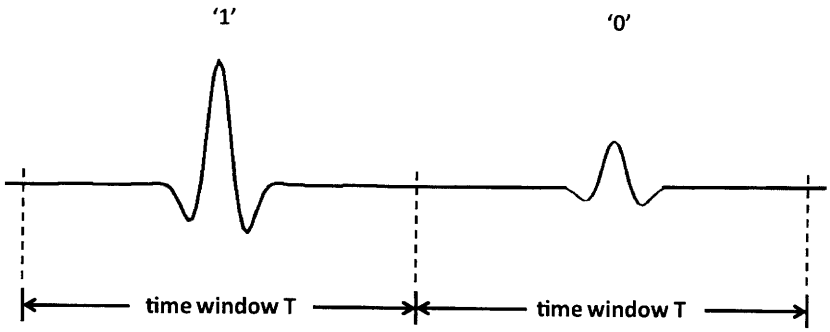


Figure 1-7: Pulse Amplitude Modulation

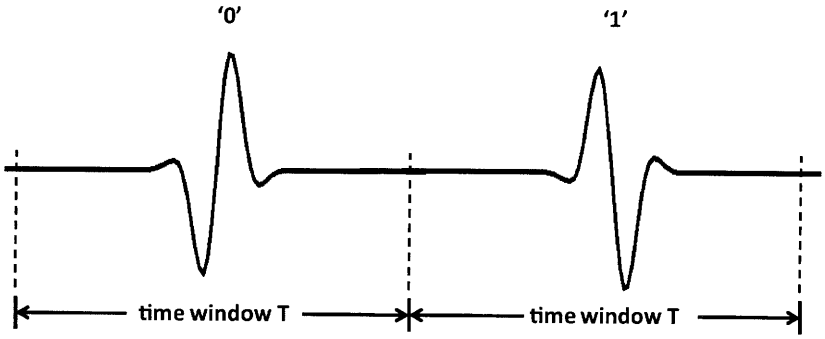


Figure 1-8: Binary Phase Shift Keying

1.2.3 Coherency

UWB receivers can be separated into coherent and non-coherent receivers, based on the demodulation techniques they employ. A coherent receiver is capable of detecting the transmitter carrier phase. It can lock to that phase with its own locally generated carrier frequency. A non-coherent receiver, on the other hand, discards the carrier phase. This greatly simplifies the complexity of the receiver architecture. However, a non-coherent receiver is limited to OOK, PPM and PAM modulation schemes. BPSK modulation encodes information in the phase of the carrier signal, and therefore is not compatible with non-coherent receivers.

In general, non-coherent receivers have a lower effective data rate for a given bit error rate. However, by discarding the phase information, non-coherent systems have relaxed frequency requirements. Therefore, transmitters with all-digital architectures are possible [3]. Essentially, the non-coherent system trades wireless transmission performance for a reduction in power consumption.

1.3 Previous Work

Since the FCC approval of unlicensed use of frequency band 3.1-10.6 GHz for UWB applications, there has been increased research interest in the field of UWB technology. Many advances have been made, both in the field of OFDM-UWB technologies, and the pulse-based UWB technologies. Compared to pulse-based UWB, OFDM-UWB can achieve a much higher data rate, up to 480 Mb/s [4]. OFDM-UWB chipsets have already be demonstrated for short-range, high data rate applications, such as streaming video from DVD players to home TV, and for wireless USB capabilities [8]. However, the disadvantages of OFDM-UWB technology are relatively high power consumption rate, and increased complexity in the transmitter and receiver circuits.

This thesis discusses the implementation of a pulse-based, non-coherent UWB wireless transmission system. As a result, the rest of this section focuses on the capabilities of some previously published pulse-based, non-coherent UWB transmitters and receivers. These systems generally operate at data rates up to tens of Mb/s, an order of magnitude less than the high data rate OFDM-UWB systems.

In 2007, David Wentzloff and Fred Lee demonstrated a 47 pJ/pulse all digital UWB transmitter and a 2.5 nJ/b non-coherent UWB receiver [9] [10]. They also presented a streaming video application built using these UWB radios [11]. This non-coherent transmission system employs PPM modulation, operating in the 3 - 5 GHz frequency range. A 30 ns time window T_{int} is used to transmit a PPM symbol. The time window is separated into time slots T_{int1} and T_{int2}. A 2 ns pulse in T_{int1} transmits a logic '1', while a 2 ns pulse in T_{int2} transmits a logic '0'. The data rate can be changed by scaling the time period between PPM symbols. In the system, 30 ns delays are introduced between adjacent PPM symbols, producing maximum data rate of 16.7 Mb/s. This demonstration platform has been shown to work at a distance of 3 meters [11].

In 2008, Dries Neirynck et. al. presented a pulse-based UWB platform [12]. Using PPM modulation techniques, this UWB system demonstrates 1 Mb/s transmission data rate at a range of 3 meters. Furthermore, they theorized that with additional gains, the system can

achieve reliable wireless communications for up to 30 meters.

In 2009, Marian Verheist et. al. demonstrated a 110pJ/pulse UWB receiver that integrates an analog front-end with a digital baseband processor, along with ranging capabilities [13]. This receiver can operate with data rates between 0.3 Mb/s to 40 Mb/s, with a transmission range over 10 meters. This system uses a coherent receiver, and operates in the sub-GHz frequency range.

1.4 Goals

This thesis aims to develop a moderate data rate, short-range wireless image transmission system. The system demonstrates a reliable data link, with a transmission range over 10 meters. It uses the ultra low power UWB wireless radios developed by the Digital Integrated Circuits and Systems Group at MIT. In particular, the UWB wireless transmission system described in this thesis uses the second generation UWB transceiver designed by Denis Daly and Patrick Mercier in 2007 as a transmitter. The system also uses the third generation UWB receiver designed by Denis Daly and Patrick Mercier in 2008, with digital baseband algorithms designed and implemented by Manish Bhardwaj and Patrick Mercier [14] [15].

The moderate data rate wireless transmission system presented in this thesis is used for image transmission. This allows visual detection of the data transmission rate and transmission errors. It provides an easy way for users to judge the quality of UWB transmission without having to examine the detailed statistics generated during the transmission process.

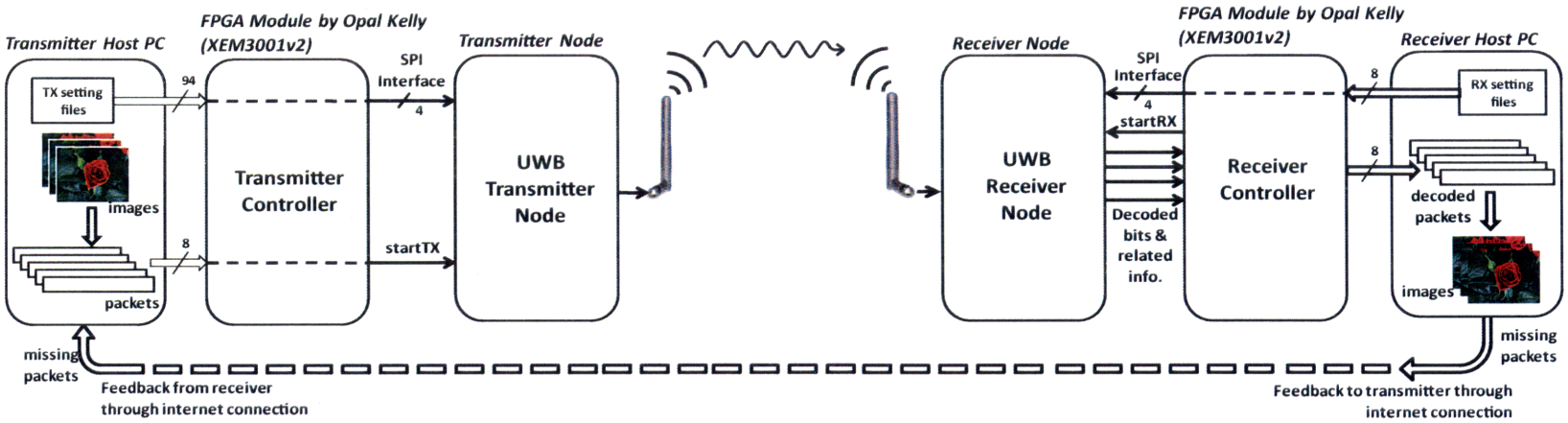


Figure 1-9: Top Level Block Diagram for the UWB Wireless Image Transmission System

A top level block diagram of the UWB image transmission system is presented in Figure 1-9. As seen in the block diagram, this image transmission system features a one-way UWB transmission channel. On the transmitter side, a host PC processes the images into transmittable packets, and sends them to the UWB radio through an FPGA module. The UWB receiver radio receives the packets, decodes the bits, and passes them back the receiver host PC through another FPGA interfacing module. The receiver host PC collects the decoded bits and reconstructs the original image. To improve the quality of the image transmission, a feedback path is provided through an internet connection between the receiver and the transmitter hosts. The feedback path is used for error correction purposes. The system also allows for dynamic adjustments of the transmitter and receiver settings to trade off the data rate with the range of transmission.

1.5 Transmission Packet Structure

A non-coherent receiver is used in the UWB image transmission system described in this thesis [14] [15]. The receiver collects signal energy in discrete time windows, and decodes transmitted bits based on these signal energies. However, the receiver has no prior knowledge of the transmitter time windows with regards to its own perceived time window. This information is essential for the correct decoding of transmitted data. Therefore, some predetermined preamble signals need to be transmitted before the actual data so the receiver can detect and synchronize its time window to the transmitter time window.

In this UWB wireless transmission system, the data to be transmitted, known as payload, is proceeded by a preamble, a start frame delimiter (SFD), and a header, as shown in Figure 1-10. The entire packet utilizes two different modulation schemes: the preamble and the SFD use OOK, and the header and the payload use PPM.

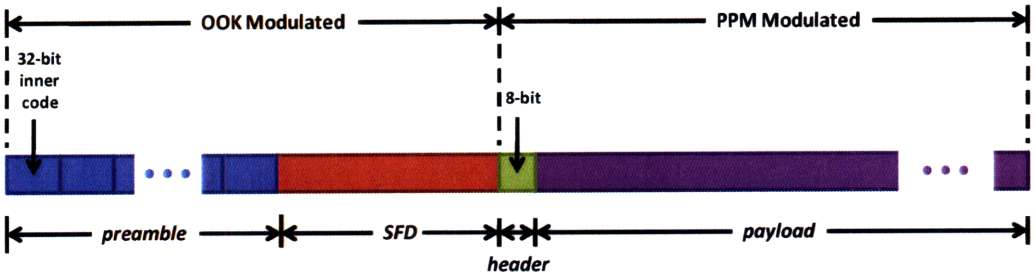


Figure 1-10: Basic Structure of a Transmitted Packet

1.5.1 Preamble

The preamble used in this system is repetitions of a pre-determined 32-bit code, known as an inner code. Each bit is transmitted using the OOK scheme, with back-to-back pulses in 31.2 ns time windows, as shown in Figure 1-11. Therefore, transmitting one repetition of the inner code takes approximately 1 μs. The receiver must successfully detect a packet and synchronize to the transmitter time windows during the length of the preamble. Therefore, the inner code needs to be repeated many times. An optimal value of the preamble length is determined based on the targeted transmission range, and can be calculated based on some programmable receiver settings. Details of this analysis and the optimal preamble length over various transmission distances can be found in Chapter 4.

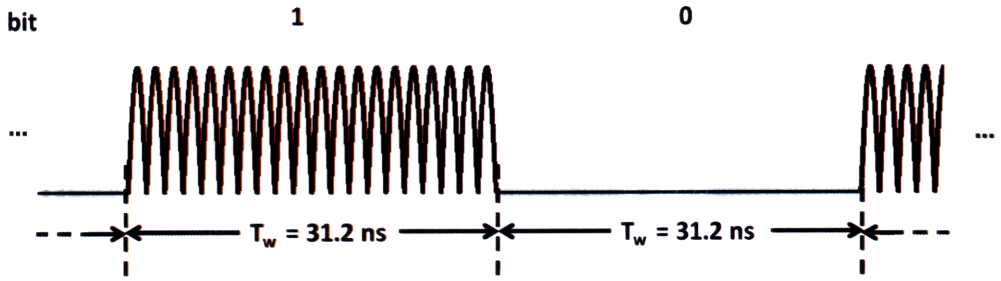


Figure 1-11: The OOK Modulation Scheme Used in Preamble and SFD Transmission

1.5.2 SFD

The SFD is used to signal the end of the preamble and the start of the header, at which point the receiver must switch to PPM to demodulate received signals into decoded bits. The SFD

is represented with a 5-bit code, known as the outer code. Each bit in the code indicates the presence or absence of a preamble inner code transmission, as shown in Figure 1-12. The SFD last approximately 5 μs long. After successful synchronization, the receiver looks for this pattern in the transmission. If found, the receiver immediately switches to demodulate PPM and decode the header and payload. Otherwise, the receiver assumes false detection and goes back to looking for the preamble.

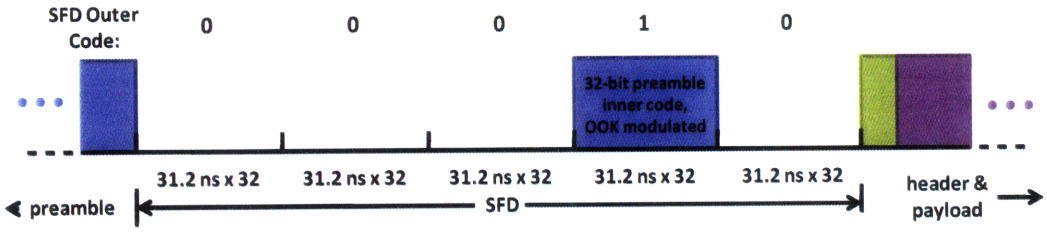


Figure 1-12: The Structure of SFD

1.5.3 Header and Payload

Unlike the preamble and the SFD, the header and the payload are encoded with PPM. The details of the PPM modulation used is shown in Figure 1-13. The receiver collects energy over two 62.4 ns time windows and determine the transmitted bit based on which time window contains greater energy. The 31.2 ns pulse burst is positioned in the middle of the 62.4 ns window to provide significant margin for synchronization.

The header is an 8-bit number that represents how many bytes there are in the payload. It is transmitted with the most significant bit first. The receiver can choose to use this header value or to ignore it. For example, the receiver can be configured to discard a packet if the header does not match a preprogrammed value. As a result, if the length of the payload is constant and known to the receiver prior to transmission, the header can serve as an error detection mechanism to improve the accuracy rate of the wireless transmission.

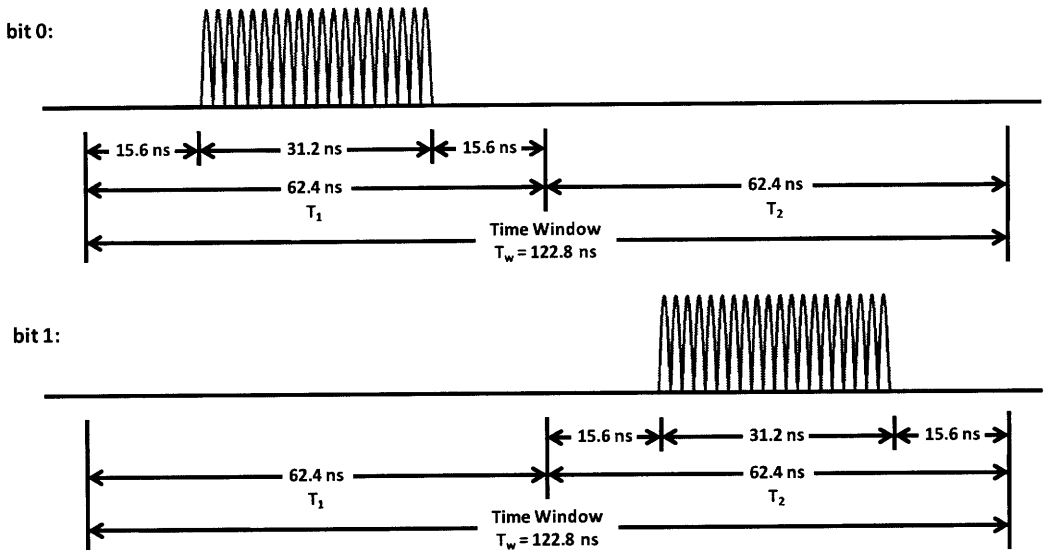


Figure 1-13: The PPM Modulation Scheme Used in Header and Payload Transmission

1.6 Thesis Structure

The rest of this thesis describes the UWB wireless image transmission system in more detail. Chapter 2 discusses the design and implementation of the transmitter system, including a high level description of the transmitter chip used, details of the transmitter controller implemented on the FPGA that interfaces the transmitter host PC and the transmitter chip, and the image processing and packet generation algorithms that are implemented on the transmitter host PC. Chapter 3 is devoted to the design and implementation of the receiver system. It includes brief descriptions of the receiver baseband algorithm and the receiver chip, some details of the receiver controller implemented on the FPGA that interfaces the receiver host PC and the receiver chip, and the payload data collection and image reconstruction algorithms that are implemented on the receiver host PC. Chapter 4 examines some techniques used to increase the range of the UWB transmission, such as increasing receiver gain settings and payload averaging. It also evaluates the performance of the overall transmission system. Finally, Chapter 5 summarizes this thesis, and offers suggested improvements that can expand this work in the future.

Chapter 2

UWB Transmitter System Hardware and Transmission Algorithms

The transmitter of the UWB image transmission system has three major components, as shown in the block diagram in Figure 2-1: a host PC, an FPGA integration module, and a transmitter node. The MATLAB scripts running on a host PC is the "brain" of the system, controlling both the configuration of the UWB radio on the transmitter node, and what is being transmitted through it. However, the commands from the host PC can only be communicated with standard protocols such as serial or USB ports. Therefore, an FPGA module with USB connectivity made by Opal Kelly Incorporated (XEM3001v2) is used to interface the host PC and the UWB transmitter node. In addition, Opal Kelly Inc. also provides an application programming interface (API), known as FrontPanel, which provides an easy way for the host PC to control and transfer data to the FPGA [16]. The three major components of the transmitter system are described in detail in the following sections.

2.1 The UWB Transmitter Node

The main component of the transmitter node is the UWB transmitter IC [17]. This non-coherent UWB transmitter has an all-digital architecture, and can be programmed to com-

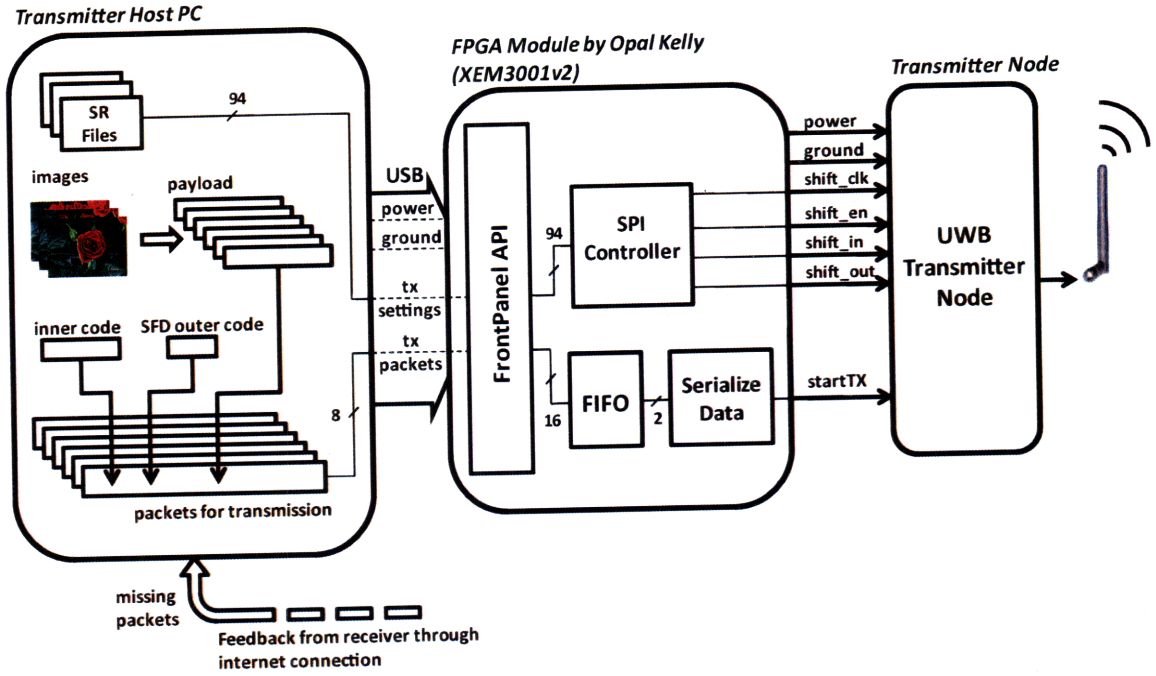


Figure 2-1: Top Level Block Diagram for the Transmitter System

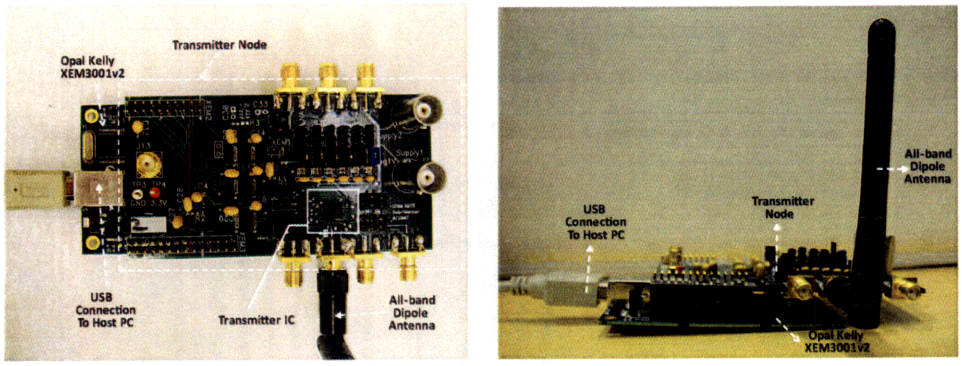
municate in one of three 499.2 MHz channels in 3-to-5 GHz frequency band, as specified by IEEE 802.15.4a. The transmitter consumes 112 to 19 pJ/pulse for data rates from 100 kb/s to 15.6 Mb/s. For the measurements of all the performance tests presented in this thesis, the transmitter is configured to transmit at a center frequency of 3.9936 GHz. However, the transmitter can be easily programmed to operate in one of the other frequency channels.

The transmitter is programmed by loading values into its shift registers through a serial peripheral interface (SPI). These shift registers contains vital information for the operation of the UWB radio, such as whether it should operate as a transmitter or a receiver, and which channel it operates in. Therefore, the shift register values must be programmed once the chip powers up.

When operating as a transmitter, back-to-back pulse bursts are generated at the rising edge of the off-chip *startTX* signal. This signal is generated by the Opal Kelly FPGA module. The number of bursts in a pulse is specified by a 5-bit *tx_pulse_num* value in shift register 14. This value can be adjusted to produce 31.2 ns of continuous pulse bursts at

each rising edge of the *startTX* signal, as specified by both the OOK and PPM modulation schemes used in this UWB image transmission system.

Figure 2-2 shows the top and the side view of the transmitter node. The transmitter chip sits on a printed circuit board (PCB) custom designed by Denis Daly and Patrick Mercier. The PCB is mounted directly on top of the XEM3001v2 module. The transmitter node uses a dipole all-band antenna, which is connected to the PCB as shown in Figure 2-2(b). The transmitter PCB also contains power management chips that allow the transmitter to receive power from the XEM3001v2 module. This eliminates the need for a separate power source, and allows for increased portability for the transmitter system in this wireless image system.



(a) (b)

Figure 2-2: Transmitter Node: (a) Top View, and (b) Side View.

2.2 Transmitter Host PC

In the UWB image transmission system, the host PC is the controlling component that is responsible for processing a series of images into packets that can be wirelessly transmitted. It is also controls the transmitter IC settings and can automatically change packet configurations based on the targeted range of the wireless transmission. A graphical user interface (GUI) is built for the UWB transmitter system in MATLAB, and a screenshot is included in Figure 2-3. This GUI is designed to accommodate users with different backgrounds in

UWB technology. Users can follow a few simple steps to set up and use this image transmitter without needing to know the details of UWB and the transmission algorithms used. However, the GUI also gives detailed information about transmitter settings and packet configurations, so users familiar with the UWB wireless transmission algorithms can easily modify these settings to suit their needs. The details on how to use this transmitter system is explained in Appendix A. The following section provides a general overview on some key features available on this GUI, and describes how these features are implemented.

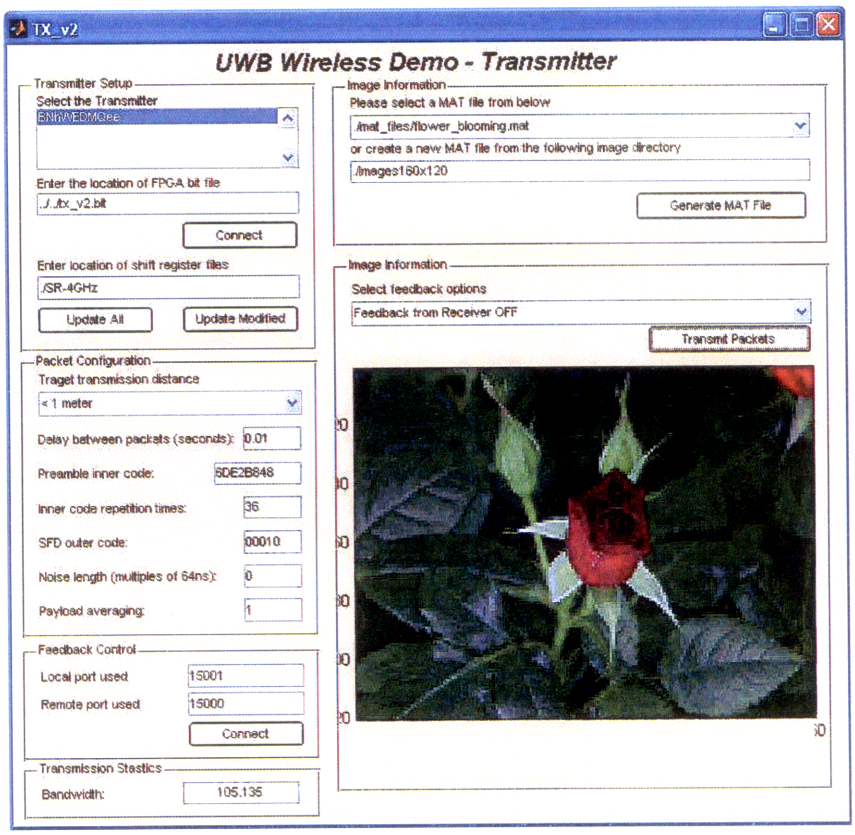


Figure 2-3: Screenshot of the Transmitter System GUI

2.2.1 Transmitter Setup

The host PC interfaces with the Opal Kelly XEM3001v2 FPGA integration module through the FrontPanel API. Upon starting the GUI, the host PC finds all connected Opal Kelly

modules. All the Opal Kelly XEM3001v2 boards are physically identical, which may confuse users. As a result, the transmitter GUI forces the user to connect only one board to the host PC. If no board or multiple boards are detected, the key features of the GUI will be disabled. This includes the ability to program the UWB transmitter IC shift registers and to generate and transmit packets.

The XEM3001v2 interface module needs to be properly configured and the FPGA needs to be initialized before it can generate proper signals to control the transmitter radio. The FPGA is initialized by loading a Xilinx bit file. A default bit file location is provided by the GUI. If the user wants to use a different bit file, s/he can simply modify the default bit file location, shown in the GUI. In addition, the host PC also ensures that the FGPA on the XEM3001v2 receives a 64 MHz clock, by configuring the phase locked loop (PLL) on the XEM3001v2 board.

The transmitter radio is configured with a set of shift registers. On the host PC side, these shift register values are written into a set of csv files, which are stored in a directory. The values in the csv files are read into MATLAB, and transferred to the XEM3001v2 module through the FrontPanel API interface. The FPGA on the XEM3001v2 module sends these values to the transmitter radio through an SPI interface. The GUI provides a default directory containing a set of these csv files; however, a user can change the default UWB transmitter settings by changing the default directory path on the GUI to a new directory containing csv files with different shift register values.

2.2.2 Packet Configuration

The packet configuration part of the GUI gives users access to the preamble inner code, the SFD outer code, the preamble length, and other related options for creating a UWB packet. These options need to be modified for different transmission distances. Detailed analysis of the optimal packet configuration for different transmission distances can be found in Chapter 4. The GUI automatically adjusts packet configurations given a target transmission range based on these findings. The default transmission range is 1 meter. Like all the previous

features on the GUI, these default packet configurations can be easily overwritten by the user.

2.2.3 Image Compression

The images being sent in this UWB wireless systems are 160 by 120 pixels. Each pixel is represented by 8-bit red, green, and blue (RGB) values. To increase the speed of transmission, the image needs to be compressed. The lower nibble of the 8-bit RGB values are masked off to 0, and therefore are not be sent. The image size halves from 56 kb to 28 kb, but there is not much reduction in the quality of the resulting image. This can be seen in Figure 2-4, which shows the original image and a compressed image side by side. This simple compression allows for doubling of the image transmission speed without much sacrifice to the image quality.

(a) (b)

Figure 2-4: Comparisons between (a) original 160 by 120 image, and (b) 160 by 120 image with lower 4 bits of RGB values masked to 0

If the payload is too long, the synchronized receiver time window may start to drift away from the transmitted time window, due to imperfect matching between the transmitter and receiver clock periods. Therefore, the payload length needs to be limited. In this wireless transmission system, each image is divided into 240 blocks, with a block size of 120 bytes. The image compression algorithm used here starts off with the two left-most pixels in the bottom

row of the image. The RGB values at these two pixels are read, and the corresponding upper nibbles are isolated. The resulting data are put into 2 entries of an array of 120 elements, as shown in Figure 2.2.3. The algorithm operates from left to right on the same row, and at the end of the current row, moves up to the left-most pixel on the next row. When the current array fills up, it is inserted into a row of a 240 by 120 matrix, and a new array is started. As a result, each array contains information for 60 pixels, which corresponds to half of a row in the image. This image compression algorithm takes each image and transforms it into a 240 by 120 matrix, which each row containing a block to be transmitted as the payload of a UWB packet.

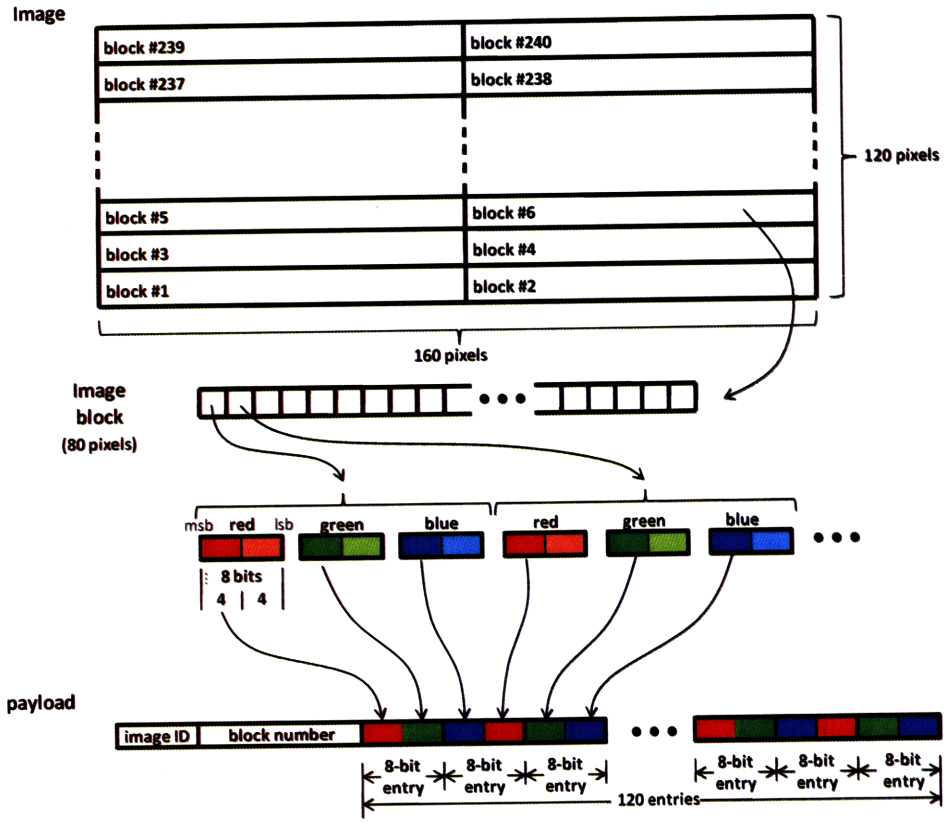


Figure 2-5: The Image Compression Algorithm

Running this image compression algorithm is very time consuming, and an image has to be processed and separated into appropriate data blocks before data packets can be generated and transmitted. Without specialized hardware support, the overall transmission

data rate can decrease dramatically if the images are processed in real time. As a result, in the current system, all the images to be transmitted are preprocessed and saved into a MAT file prior to the start of the wireless transmission. At start-up, the transmitter GUI finds all the available MAT files containing image data, and the user can select one of these files for image transmission. Alternatively, the GUI allows users to generate new MAT files from images placed in a directory. These new MAT files can be used immediately for image transmission.

Comparisons with Other Image Compression Methods

The image compression algorithm used in this system is a very basic technique. There exists other more complex compression schemes, such as MPEG and H264. MPEG and H264 are most useful when successive images are largely identical, containing only minor changes. It is commonly used to encode frames in a video file. MPEG and H264 compression schemes can dramatically decrease the amount of disk space required for each image, thus lowering the length of time required to transmit each image and improving the speed of the image transmission. However, MPEG and H264 compressions are very computational intensive, therefore are not used in the current system. In the future, specialized hardwares can be added to the transmitter system to support these image compression schemes.

2.2.4 Packet Generation

To simplify the transmitter controller on the FPGA, the host PC is responsible for translating each bit in the packet into appropriate *startTX* signal sequences for the UWB transmitter IC. To produce 31.2 ns of continuous pulse bursts, the transmitter needs a rising edge of the *startTX* signal. Table 2.1 shows the *startTX* signal sequences needed to transmit a bit in both the OOK and PPM modulation schemes. The FPGA on the XEM3001v2 module uses a 64 MHz clock, therefore, *startTX* value is updated every 15.6 ns. As a result, 2 *startTX* values are needed to encode for a bit using the OOK scheme, and 8 *startTX* values are needed to encode for a bit using the PPM scheme.

	startTX Signal Sequences (T = 15.6 ns)	
	bit 0	bit 1
OOK Modulation (31.2 ns per bit)	00	10
PPM Modulation (124.8 ns per bit)	01000000	00000100

Table 2.1: *startTX* Signal Sequences for Transmitting A Bit using OOK and PPM Modulation Schemes

On the host PC, each packet is seperated into two parts: the first portion contains the preamble and SFD; the second portion contains the header and payload. The separation is made based on the different modulation schemes. The two parts are generated independently. The preamble and the SFD are made with parameters shown in the packet configuration block of the transmitter GUI. The resulting values are then translated to *startTX* signal sequences using OOK modulation. The preamble and the SFD remain the same for all packets, therefore these values need to be generated only once and can be reused throughout the rest of the packet transmission process.

The 120-byte blocks from the image transmission comprises the main part of the payload. An image ID and a block number are attached to the front of the image block, so the receiver has enough information to reconstruct the transmitted image. The image ID is 1 byte, and the block number is 2 bytes. The resulting payload has a final length of 123 bytes. The host PC detects the length of the payload and generate appropriate header values. These values are then translated into *startTX* signal sequences using the PPM modulation scheme shown in Table 2.1.

Unlike the image compression algorithm, the packet generation algorithm runs during the image transmission process. Each packet is generated dynamically from the packet configuration settings and the image block data, and is transferred to the XEM3001v2 to be transmitted through the UWB transmitter IC.

2.2.5 Feedback Control

The image transmission system features a one way UWB wireless link, with a feedback path provided through an internet connection from the MATLAB on the receiver host PC to the

MATLAB on the transmitter host PC. The receiver host PC can use this connection to tell the transmitter host PC which packets were lost in the transmission, so the transmitter can resend these packets. This connection is established through User Datagram Protocol (UDP) in MATLAB. By default, the remote port is 5001 and the local port is 5000. These port numbers can be changed, but they have to match the remote and local port values declared on the receiver host PC. For detailed instructions on how to set up the feedback connection, please refer to Appendix A. This feedback path is used for correcting errors in the transmission, and can be disabled easily. The details of the feedback operation are explained in Chapter 3.

2.3 Opal Kelly FPGA Module (XEM3001v2)

The Opal Kelly XEM3001v2 is used as an interfacing device between the host PC and the transmitter node. It is an integration module based on a 400k gate Xilinx Spartan-3 FPGA. It has 88 pins for general I/O connectivity, and can be programmed through a USB cable connection. The FPGA is programmed with state machines that provide the control signals for the transmitter radio. These state machines are described later in this section.

The XEM3001v2 board also has power management peripherals such that the FPGA can be entirely powered by the 5-V supply from the USB connector. Furthermore, it can supply power to external devices connected to the module. The transmitter node utilizes this functionality. The XEM3001v2 board also has a PLL that can provide up to 5 clocks, with frequencies up to 150 MHz. The PLL and the Spartan-3 FPGA can both be configured through the FrontPanel API. In addition to configuration of hardware components, the FrontPanel API also allows easy data transfer between the host PC and the XEM3001v2 module.

2.3.1 Data Transfer through FrontPanel API

The FrontPanel API, developed by Opal Kelly Incorporated, offers an easy way to communicate between a host PC and the XEM3001v2 modules through the USB connection. The API provides three ways to move data between the host PC and the FPGA integration modules, as summarized below.

- *Wires*: Wires are asynchronous connections between the PC and the Opal Kelly XEM3001v2. They are slow, and can be updated at most 1000 times per second [16]. They are usually used for asynchronous control signals, such as reset, and to convey state of internal signals from the FPGA.

- *Triggers*: Triggers are synchronous connections between the PC and the XEM3001v2 module. Like wires, they are slow connections, and can be updated at most 1600 times per second [16]. As an input to the XEM3001v2 module, when asserted, the trigger creates a signal that is high for one clock cycle. As an output from the XEM3001v2, the FrontPanel API controller sets the trigger bit when it detects a rising edge on a signal. Once set, the trigger will remain set until the FrontPanel polls the XEM3001v2 module. Therefore, multiple triggers between polls can not be detected. The triggers are useful for initiating single events, such as when the PC wants to start a state machine on the XEM3001v2 module, or when the module notifies the PC that it has completed a certain task.

- *Pipes*: Pipes are also synchronous connections between the PC and the XEM3001v2 module. Unlike triggers, they continuously transfer data bytes in series, and are ideal for downloading and uploading large memory contents. Once a pipe transfer starts, it will continue to completion, therefore, a first-in-first-out (FIFO) memory storage should be placed in the FPGA so continuous data flow can be guaranteed. The pipe transfers are ideal for moving large quantities of data, with bandwidth of 353 kB/s to 38.2 MB/s depending on the size of the transfer block [16]. The pipe data are transferred as 8-bit words on the PC side, but converted to 16-bit words on the XEM3001v2

FPGA side. The first 8-bit word transferred on the PC side becomes the lower byte on the FPGA side, and the second 8-bit word becomes the higher byte.

2.3.2 Transmitter Controller

The FPGA on the XEM3001v2 module is responsible for generating appropriate signals to configure and control the transmitter IC. The XEM3001v2 module receives the packet from the host PC via pipe transfer, and translate it into *startTX* signals to the transmitter radio so appropriate UWB pulses can be generated. A simple state machine is needed to serialize received 16-bit word to single bit *startTX* signal. Figure 2-6 shows the overall block diagram of this system.

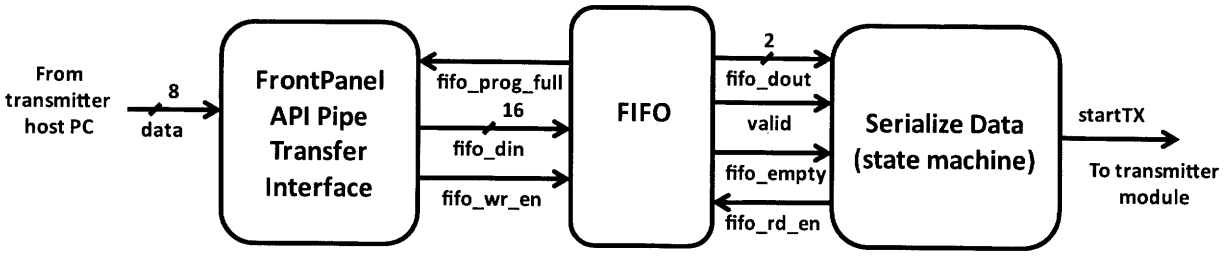


Figure 2-6: Block Diagram for the Transmitter Controller

The pipe transfer interface feeds directly into a FIFO. The *FIFO_prog_full* signal is configured so that it pulls high when the FIFO no longer has enough room to store another packet. This signal is used to enable the pipe transfer interface, so that the host PC can only start a transfer when the FIFO has enough room to store the entire packet. To simplify the *serialize_data* state machine, the FIFO is designed to be asymmetric, with 16-bit word input width and 2-bit word output width. This FIFO can be implemented using the Xilinx CORE generator. As a result, the *serialize_data* is a very simple state machine. The state machine operates such that as long as there is data residing in the FIFO (signaled by *FIFO_empty*), it will assert *FIFO_rd_en* to read 2-bit data from the FIFO, and send each bit to the transmitter radio as the *startTX* signal.

As mentioned before, the radio is configured by programming a series of shift registers through a SPI interface. The shift register values are passed in parallel into the XEM3001v2

module using FrontPanel wires, and the shift register programming is initiated with a FrontPanel trigger. The trigger starts the state machine, which enables the SPI interface, generates the proper clock signal (*shift_clk*), serializes the shift register values, and passes the resulting bits onto the *shift_in* line. This state machine is designed by Denis Daly and Patrick Mercier, therefore, this thesis will not discuss it in more detail.

2.4 Summary

This chapter provided a description of the structure and implementations of the transmitter system. The host PC in the transmitter system processes a series of images into UWB packets. It sends these packets to the UWB transmitter radio through an Opal Kelly FPGA integration module. In addition, a GUI is provided on the host PC so users can easily use the transmitter system to suit their need. The receiver system structure and implementation are discussed in the next chapter.

Chapter 3

UWB Receiver System Hardware and Algorithms

This chapter focuses on the design and implementation of the receiver system in the UWB image transmission platform. A top-level block diagram is provided in Figure 3-1. Like the transmitter system, the receiver system is broken down into three major blocks: the host PC, the XEM3001v2 interfacing module (this module will be referred to henceforth as the receiver controller), and the receiver node. In this system, the receiver node detects and decodes incoming packets, and then it sends the decoded bits, along with other related information, to the receiver controller. The receiver controller gathers the decoded bits into data packets, and forwards them to the host PC. The host PC then reconstructs the transmitted images from these data packets. The host PC is also responsible for the configuration of the receiver node at start-up.

The UWB receiver node is based on a UWB receiver chip designed by Denis Daly, Patrick Mercier and Manish Bhardwaj in 2008 (from now on referred to as the 2008 UWB receiver). This receiver integrates the baseband processor on-chip with the UWB RF front-end. This chapter first describes the receiver baseband processing algorithm. It then discusses the receiver node. The receiver controller implemented on the Opal Kelly XEM3001v2 FPGA integration module is examined next. Finally, this chapter discusses the MATLAB GUI

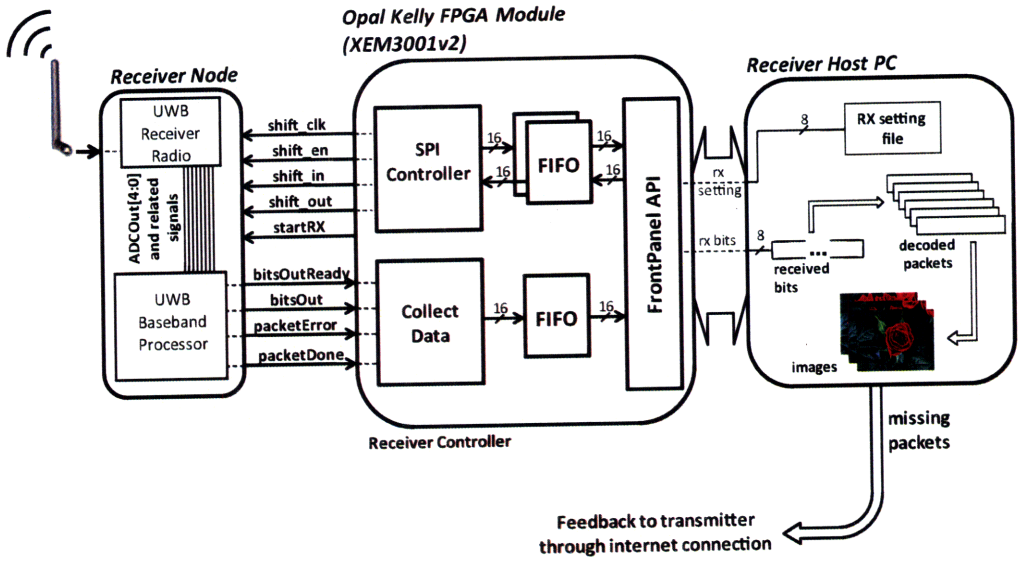


Figure 3-1: Top Level Block Diagram for the Receiver System

that is implemented on the host PC, and details how the original transmitted image is reconstructed from received packets.

3.1 Receiver Baseband Algorithm

This section describes the basic algorithm that the UWB receiver uses to detect, synchronize, and decode transmitted data. The algorithm is incorporated into the 2008 UWB receiver chip, and is designed and implemented by Manish Bhardwaj and Patrick Mercier [14].

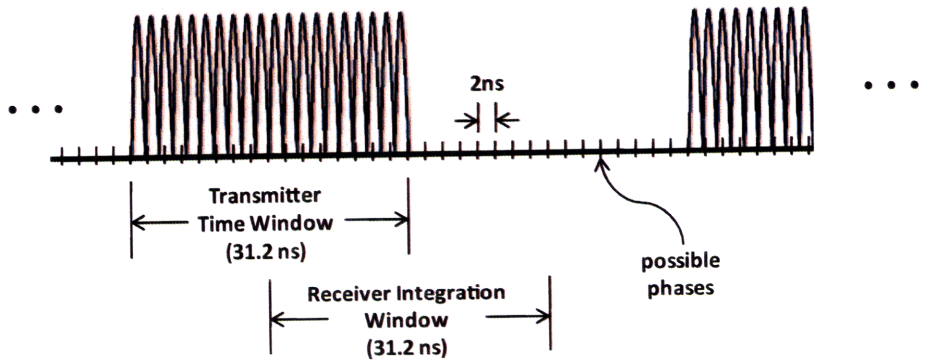


Figure 3-2: Phase of the Receiver Integration Window

The wireless image transmission system features a non-coherent receiver. To receive a packet, the receiver baseband processor needs to detect a transmitted signal and then synchronize the receiver integration window to the received signal before it can successfully decode transmitted data. The baseband algorithm achieves synchronization by using many parallel correlators. The UWB receiver radio collects total received energy over a 31.2 ns time window, and it converts the result into a 5-bit code using an analog-to-digital converter (ADC). The ADC samples can arrive at 16 different starting spaces, or phases, with respect to the true transmitted time window, as shown in Figure 3-2. The phases are separated from each other by 2 ns. The ADC samples are then compared with a preprogrammed codebook. The codebook has 32×16 entries, corresponding to the expected values of ADC readings upon reception of the 32-bit inner code for all 16 phases. Detection and synchronization are achieved by correlating the received ADC samples with the codebook. A packet is detected if the correlation is over a certain threshold. Synchronization is determined by selecting the phase with the highest correlation coefficient. The baseband processor adjusts the start of the receiver integration time window according to the selected phase in order to match the observed transmitter time window.

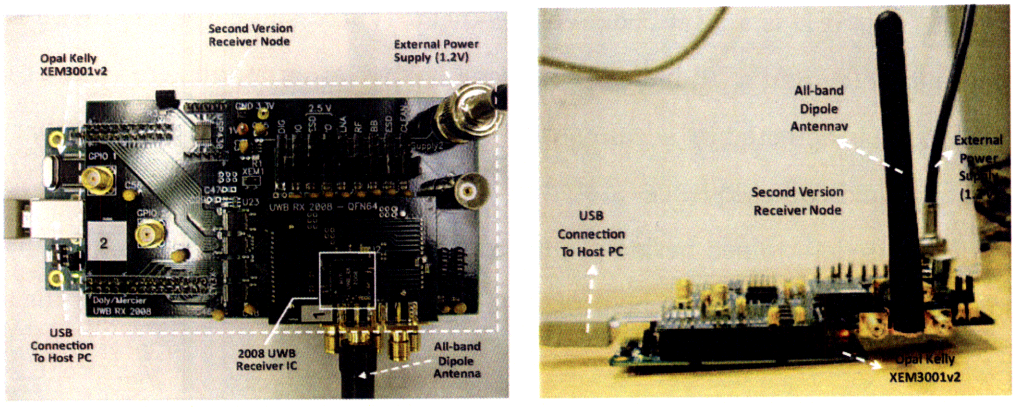
After successful synchronization, the baseband processor starts to look for the SFD. The presence of the SFD indicates the start of the header, followed by the actual data transmission. The SFD is a 5 bit code. A '1' indicates successful correlation with the 32-bit preamble inner code, and a '0' indicates successful correlation an all-zero code of the same length. The SFD is detected by accumulating these correlation results, and comparing them with predetermined code sequences stored in the SFD codebook. The receiver looks for the SFD code for a predetermined period of time after synchronization is achieved. This timeout value, known as *SFDtimeout*, can be programmed into the baseband processor through shift registers.

Finally, after the SFD code is found, the baseband processor demodulates the PPM encoded received signals. The ADC values collected over two adjacent 62.4 ns time windows are compared, and a bit value is declared depending on which time window has higher ADC

value (or pulse energy). The first 8 bits decoded are the header bits. The value of the header indicates how many bytes of data are expected in the payload. The baseband processor demodulates the said amount of bytes, and then returns to the detection mode to wait for the next incoming packet. In addition, the receiver can enter sleep mode for a programmable amount of time after successful reception of a packet. This allows power reductions in low data rate applications, where significant delays are expected between successive packets.

3.2 The Receiver Node

The receiver node is implemented with the 2008 UWB receiver [14][15]. This chip integrates the UWB receiver radio with a baseband processor. It contains many shift registers, all of which can be programmed through an SPI interface. These shift registers contain the configuration parameters for the receiver node, such as receiver signal amplifications, detection threshold, *nAveDet*, *nAveSynch*, and *SFDtimeout*. There are also read-only shift registers used to convey relevant information from the receiver IC to the outside world. One read-only shift register contains the first 64 bytes of a decoded packet. The other read-only shift registers record some important internal signal values, offering insights into the inner workings of the radio and the baseband processor. They are useful for debugging purposes.



(a) (b)

Figure 3-3: The Receiver Node: (a) Top View, and (b) Side View.

Figure 3-3 shows the top and side view of the receiver node. The PCB board used here is

a test board for the 2008 UWB receiver, designed by Denis Daly. The receiver node mounts on top of the Opal Kelly XEM3001v2 module, and uses a dipole all-band antenna. The receiver node can be powered directly from the Opal Kelly XEM3001v2 module, however, for maximum performance, an external 1.2 V power supply is preferred for the digital I/O, thereby limiting the mobility of the receiver system.

In addition to the read-only packet shift register, the decoded bits can be read in real time through the *bitsOut* and *bitsOutReady* output pins of the 2008 UWB receiver chip. This is a faster interface compared to reading the shift register. Also, the shift register has limited size, thereby limiting the length of the payload to less than 64 bytes. The *bitsOut* and *bitsOutReady* pins have no such limitations. As a result, the receiver controller module uses these two signals, along with signals *packetError* and *packetDone*, to collect decoded bits for the wireless image transmission system.

3.3 Receiver Controller

The receiver controller is implemented on the XEM3001v2 FPGA integration module made by Opal Kelly Corporation. It provides an interface that allows the host PC to communicate with the receiver node. The receiver controller takes advantage of the features of the FrontPanel API to make transferring data easy between the host PC and the XEM3001v2 module. Details of the FrontPanel API and XEM3001v2 features can be found in the previous chapter.

There are two main parts to the receiver controller: the SPI controller and the data collector. The SPI controller passes the shift register values from the host PC to the receiver node. The values in each shift register are moved to the receiver controller via pipe transfer, and stored in a FIFO. A trigger signal is used to start the state machine that operates the SPI interface. Likewise, when shift registers are read, their values are stored into a FIFO in the receiver controller. They can then be passed back to the host PC through pipe transfer. The SPI controller was designed by Denis Daly, and thus, this thesis will not describe it in more detail.

The second part of the receiver controller is the data collector, which reads the demod-

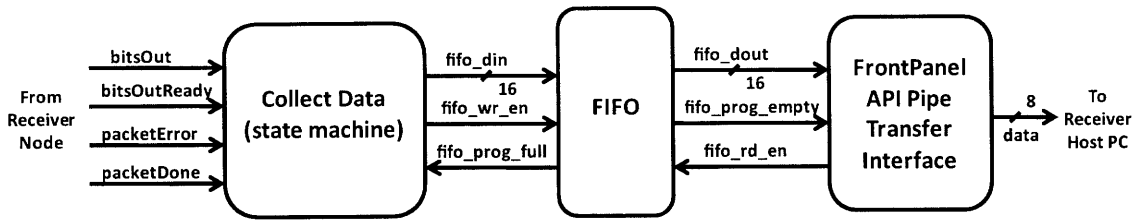


Figure 3-4: Block Diagram for the Data Collector

ulated bits from the receiver node, stores them into a FIFO, and moves them to the host PC via pipe transfers. Figure 3-4 shows a block diagram of the data collector. The *bitsOut*, *bitsOutReady*, *packetError*, and *packetDone* signals are outputs from the receiver node. *BitsOutReady* is a pulsed signal. When pulled high, it indicates that a demodulated bit is available on the *bitsOut* pin. Both the header and the payload values are outputted on the *bitsOut* pin. Signals *packetError* and *packetDone* are pulsed signals that convey the status of a received packet. A pulse on *packetError* indicates a header error. This occurs when the received header fails to match the expected header value that is programmed in the receiver shift register. After a header error, the receiver node no longer attempts to decode the remaining payload. The *packetError* signal is pulsed on the clock cycle after the first bit in the payload is decoded, even though in this situation the payload is ignored.

A pulse on *packetDone* indicates that the receiver node has successfully decoded all of the payload information for a given packet. It can be used to separate the decoded bits into data packets. *packetDone* is also important in distinguishing the header bits from the payload bits. This signal is available three clock cycles after the last bit in the payload is decoded. Figure 3-5 shows a simple timing diagram of these four signals.

3.3.1 Implementation of the Data Collector

The data collector portion of the receiver controller is implemented by the *collectData* state machine. This state machine gathers decoded bits from the four previously mentioned input signals. It parallelizes the bits into 16-bit words, and stores them in a FIFO. Once the FIFO collects over 1500 bytes of data (around 20 packets), it asserts the *fifo_prog_empty* signal.

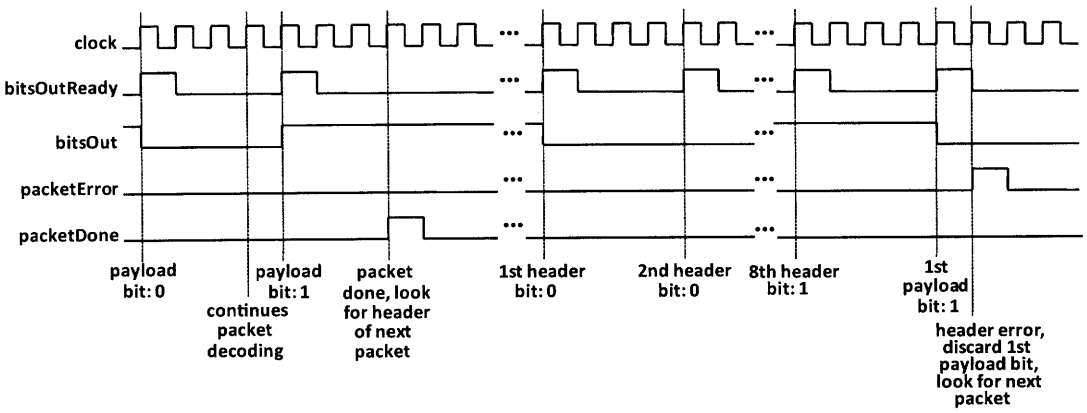


Figure 3-5: Timing Diagram of Signals *bitsOut*, *bitsOutReady*, *packetError*, and *packetDone*.

This allows the FrontPanel API to read all the data stored in the FIFO and pass them to the host PC. Each packet includes an 8-bit header, followed by the entire payload. If a header error occurs, only the 8-bit header is stored into the FIFO. Figure 3-6 shows the state transition diagram of the state machine.

In this state machine, internal registers are used as counters. *Header_count* keeps tracks of the number of header bits decoded thus far. *Bit_pos* is used to account for the next index of the 16-bit *fifo_din* that will store the incoming decoded bit. It is also used to show when the *fifo_din* is ready to be written into the FIFO.

Once the state machine initializes, it waits in *IDLE* state until it receives a valid data bit. The state machine loads the bit into *fifo_din* and updates the appropriate counters. When it arrives at the *HEADER_WAIT* state, it waits for the next demodulated bit. The state machine keeps track of the total number of bits decoded, and it checks for header errors upon the reception of this 9th bit. If a header error occurs, it discards the 9th bit, and keeps the first 8 header bits in *fifo_din*. Otherwise, the 9th bit is recorded as the 1st bit of the payload.

A pulse in *packetDone* signals the end of the current packet. This moves the state machines back to the *HEADER_WAIT* state, waiting for the next demodulated bit, which is the 1st header bit for the next packet. Also, *fifo_prog_full* signal is asserted if there is not enough room in the FIFO to store an additional packet. The state machine moves back into

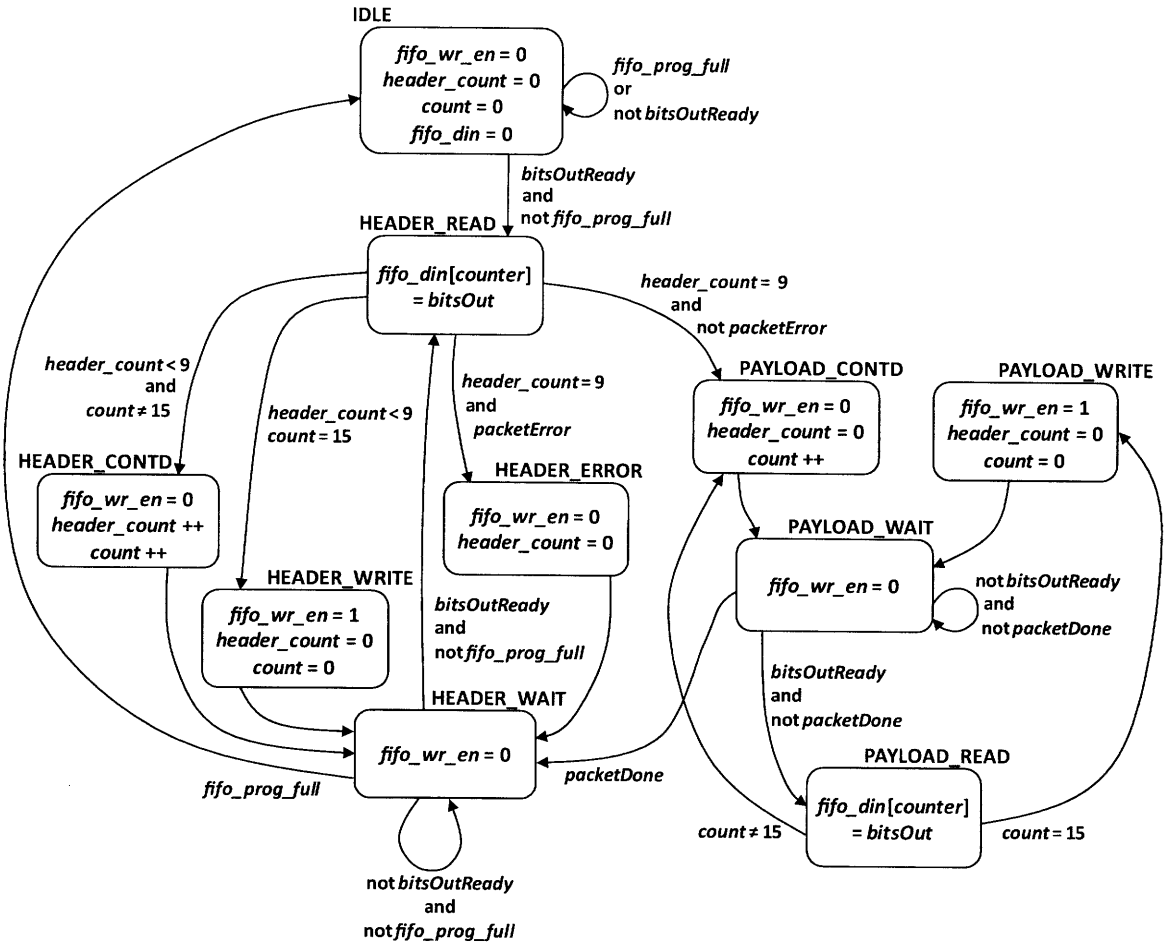


Figure 3-6: Station Transition Diagrams for the *collectData* State Machine on the Receiver Controller

the *IDLE* state until the backlog resolves. This guarantees that all the packets recorded and transferred to the host PC are complete with no missing information.

3.4 Receiver Host PC

Like the host PC in the transmitter system, the host PC in the receiver system acts like the "brain," controlling configuration for the UWB receiver radio and the baseband processor. A GUI is set up on the receiver host PC, with a screenshot shown in Figure 3-7. Like the transmitter GUI, the receiver GUI is designed to be easy to use for people without prior knowledge of the UWB transmission technology. At the same time, it provides more

experienced users easy access to some important receiver settings and parameters. The instructions on how to set up and use the receiver GUI can be found in Appendix A.

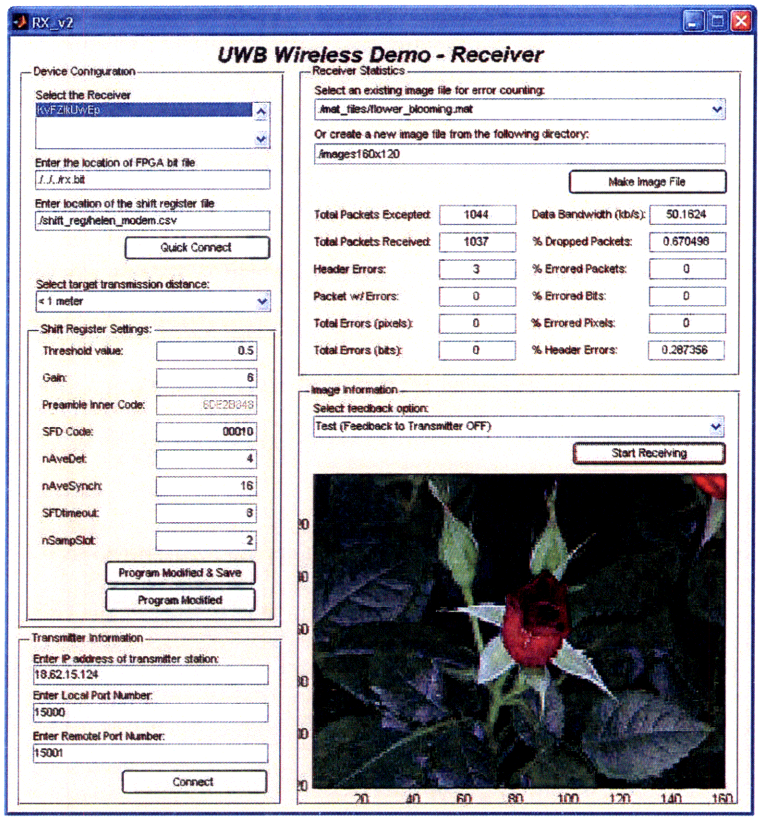


Figure 3-7: Screenshot of Receiver System GUI

The receiver host PC's operation is largely separated into four parts: receiver setup, image reconstruction, transmitter feedback, and transmission statistics. The following sections provide details of implementation and a general overview of the functionalities of each part.

3.4.1 Receiver Setup

The receiver host PC communicates with the receiver node through the receiver controller, which is implemented on the Opal Kelly XEM3001v2 integration module. Like the transmitter GUI, the receiver GUI allows only one XEM3001v2 module to be connected to the host PC to avoid confusion. Upon startup, the receiver GUI looks for all the XEM3001v2 modules that are connected to the host PC. If no board is found, or multiple boards are

connected, then some key features in the GUI are disabled. These include the ability for the receiver host PC to initialize the receiver controller, to program the shift registers on the receiver node, and to poll the receiver controller for received data and starting image reconstruction. When this happens, a user needs to make sure that one XEM3001v2 module is connected to the host PC. After doing so, s/he should restart the GUI.

Next, the Opal Kelly XEM3001v2 needs to be appropriately configured. The receiver controller is implemented on the FPGA of the XEM3001v2 module and configured by loading a bit file. The receiver GUI provides a default bit file, which implements the receiver controller described in the previous section. However, the GUI also allows users to provide a different bit file that implement other versions of the receiver controller. The host PC also configures the PLL on the XEM3001v2 module to make sure the receiver controller receives a 32 MHz clock signal.

Finally, the host PC configures the shift register settings on the receiver node. The values to be written into the shift registers are available in a csv file. The receiver GUI provides a default location for such a csv file. However, users can provide alternative csv files and program the receiver node with different sets of shift register values. The GUI also displays some key receiver settings from the csv file, such as *nAveDet*, *nAveSynch*, *SFDtimeout*, and *nSampSlot*, so the users can easily track the values that programmed into the receiver node. More information about these settings can be found in Table 3.1.

Depending on the targeted transmission range, some of the receiver node parameters from the csv file may need to be changed to ensure optimal performance. Detailed analysis of the best parameter settings for different transmission range can be found in Chapter 4. The receiver GUI automatically adjusts these parameters for a selected transmission range. The default transmission range is 1 meter. The GUI also allows users to disable this feature and restore the affected shift registers to their original values as specified by the csv file.

Receiver Settings	Description	Requirements
Gain	Controls how much the received UWB signal is amplified before it is converted to digital values for baseband processing.	The minimum gain is 1, the maximum gain is 6.
Detection Threshold	The baseband processor correlates the received signals with values in a pre-programmed codebook. A packet is detected if the correlation is greater the threshold value specified by this parameter.	Low threshold values increases false detections, high threshold values cause the baseband processor to miss many packets altogether. A default threshold value of 0.4 is used for this system.
Preamble Inner Code	Repetitions of the preamble inner code proceeds real payload data. The codebook that the receiver baseband processor uses for detection and synchronization is generated with the inner code values.	This value must match the inner code value used by the transmitter.
SFD Outer Code	The SFD outer code is used to generate the SFD codebook that the receiver baseband processor uses to find the location of SFD in an incoming UWB packet.	The SFD outer code must match the value used by the transmitter.
nAveDet	Represents the amount of time that the baseband processor collects and averages the received ADC samples, before correlating them with the codebook for packet detection.	*nAveDet* must be power of 2, with a maximum value of 32.
nAveSynch	Represents the amount of time that the baseband processor collects and averages the received ADC samples, before correlating them with the codebook for receiver synchronization.	*nAveSynch* must be power of 2, with a maximum value of 32.
SFDtimeout	Specifies the amount of time the baseband processor waits for the detection of SFD after successful synchronization. If the SFD is not detected within this period of time, the receiver returns to preamble detection.	The maximum value of *SFDtimeout* is 32.
nSampSlot	Controls the amount of payload averaging by the receiver. More information about payload averaging can be found later in this chapter.	This value must match the payload bit repetition value used by the transmitter.

Table 3.1: Summary of Key Receiver Parameters Available on GUI

In addition, the receiver GUI allows users to vary the displayed shift register settings without modifying them in the csv file. This makes it easier for experienced users to fine-tune the receiver settings for a given transmission environment. When the user is satisfied with the result, s/he has the option to save the modified shift register settings into a new csv file, allowing the same settings to be used later.

3.4.2 Image Reconstruction

The receiver host PC continuously polls the receiver controller for received packets. In contrast, the receiver controller waits until it has gathered at least 1500 bytes of data before it grants the host PC access. Thus, the host PC receives around 20 decoded packets at a time.

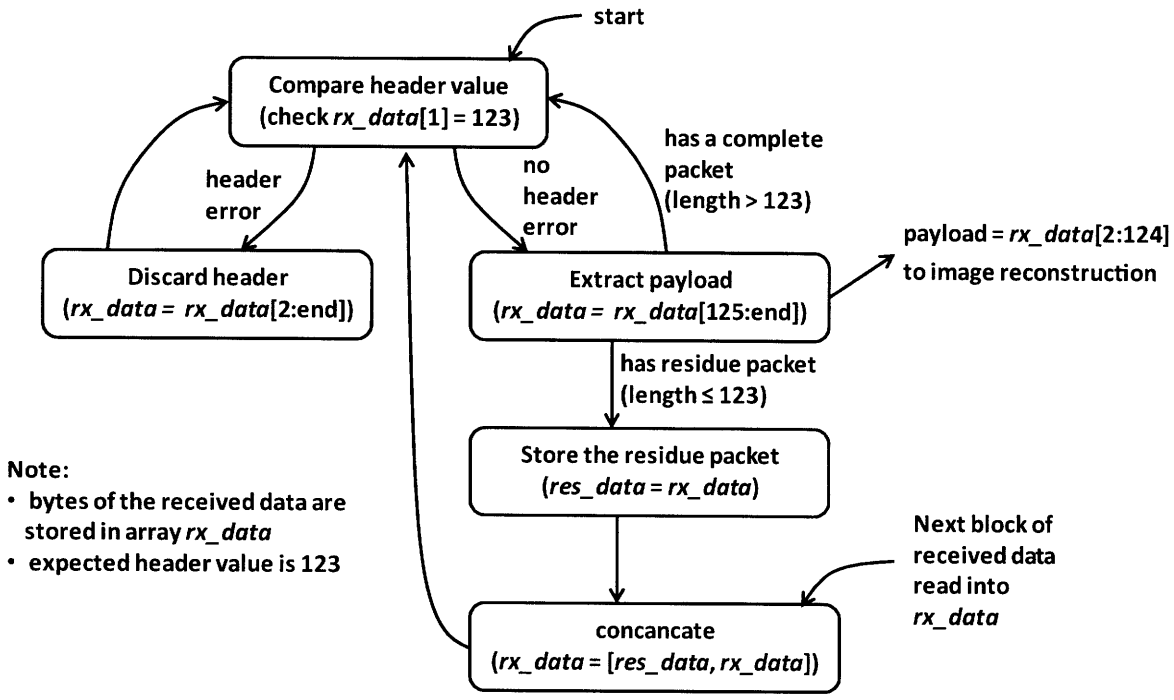


Figure 3-8: Program Flow of the Receiver Payload Extraction Process

Figure 3-8 shows the process that the host PC uses to read and group received packets. Recall that the receiver controller records only the value of the 8-bit header when there is a header error, and the header followed by the entire payload when there if no header error.

Therefore, when the host PC first receives a block of data, it knows the 8-bit value is the header. It checks for header errors by comparing the value with the expected size of the payload (in this case, 123 bytes). If the two numbers do not match, then a header error has occurred. The value is discarded and the host PC checks the next byte of the received data for header error. When there is no header error, the host PC extracts the payload, which is the next 123 bytes of data. After the payload is processed, the host PC continues with header check on the remainder of the received data. The receiver host PC cycles through the received data in this manner until there is not enough data left for a complete packet. This happens when the receiver host PC polls the receiver controller while the receiver node is decoding a packet. Thus, a portion of the packet reaches the receiver host PC, while the remainder of the packet has yet to be decoded. The missing section of the packet becomes available when the next block of received data arrives at the receiver host PC. When an incomplete packet is detected, the host PC stores the residue packet and its header into a buffer. When the next block of received data becomes available, the receiver host PC concatenates the data in the buffer with the new received data, and it repeats the process starting with header checks.

Each extracted payload is processed to retrieve the pixel data. The decoded pixels are inserted into the appropriate locations to reconstruct the transmitted image. The first byte in the payload contains the image ID, the next two bytes contains the block number, and the rest of the payload is a data block containing RGB values for 60 pixels, which corresponds to half of a row in the reconstructed image. The image ID is useful for indicating which image a particular pixel block belongs to. This is especially important when feedback control is enabled, as discussed in the next section. The block number marks where the pixel should be inserted in an image. Specific pixel information is recovered by extracting 4-bit values from the data block. These 4-bit values are the upper nibbles of the RGB values. The lower nibble of the RGB values are set to 0. Figure 3-9 is an illustration of this process.

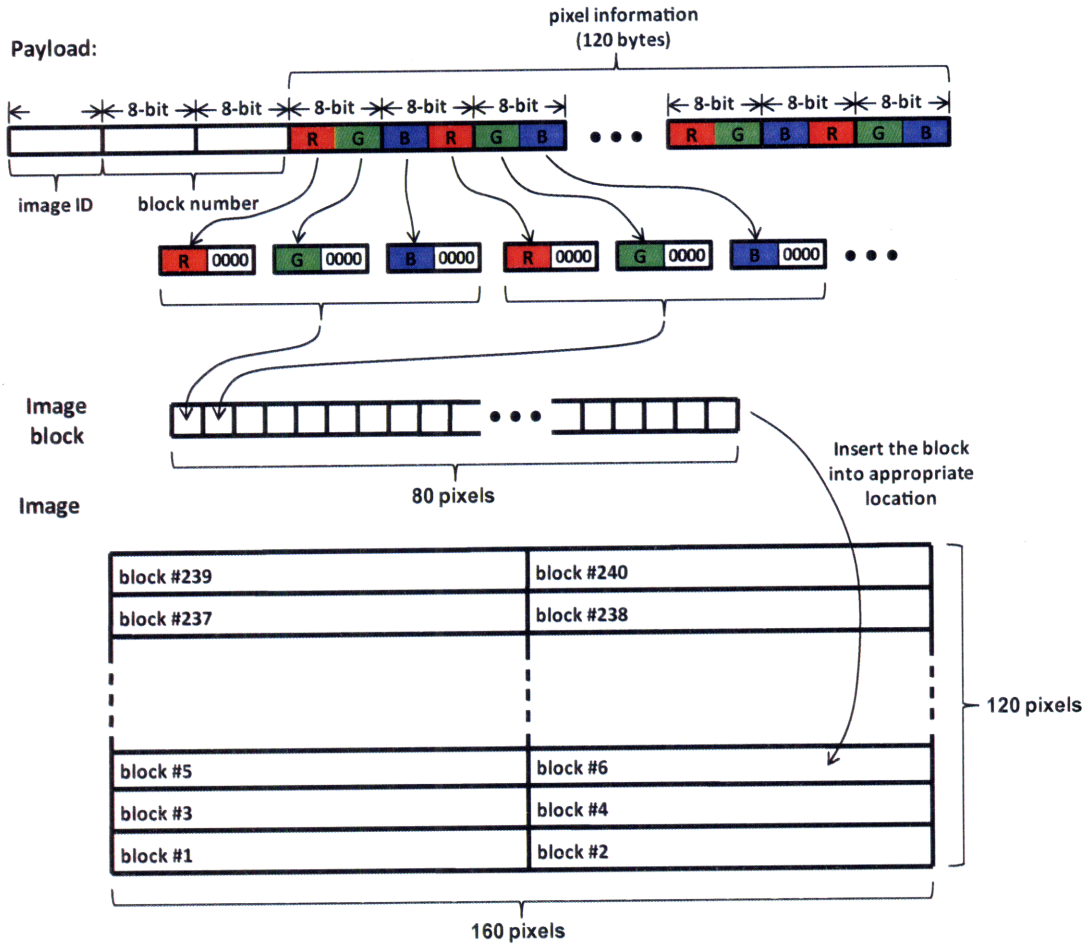


Figure 3-9: The Image Reconstruction Algorithm

3.4.3 Feedback to the Transmitter

Commercial wireless transmission systems usually feature two-way radios, so the receiver can provide the transmitter with feedbacks on the transmission quality, and signals the transmitter to resend dropped packets if necessary. While the UWB wireless transmission link featured in this thesis is a one-way link, a feedback path is provided through internet connections between the transmitter host PC MATLAB and the receiver host PC MATLAB. To use this feedback path, the user has to enter the IP address of the transmitter host PC and define the local and remote ports. By default, the local port is 5001 and the remote port is 5000. These can be changed, but they have to match the ports on the transmitter

host PC in order to properly initialize the feedback connection. The feedback connection allows the receiver host PC to inform the transmitter host PC which packets are lost in transmission and should be re-send. The receiver GUI gives users the option to disable this feedback feature.

When feedback mode is enabled, the receiver host PC keeps track of both the current image and the next image through the use of image IDs. Two arrays are created, *current_image_status* and *next_image_status*. The indices of these arrays correspond to the block numbers. Each entry stores a status flag that marks whether the block has been received. The two arrays are updated after the payload information from each packet is processed. The receiver host PC sends *current_image_status* to the transmitter host PC through the UDP connection when it finds packets from the next image. The receiver host PC also checks for missing packets in the current image. If missing packets exist, the receiver host PC waits for the transmitter to re-send these packets. Otherwise, the current image is successfully reconstructed and the receiver host PC moves on to reconstruct the next image. The current image is replaced with the next image, and *current_image_status* is replaced with *next_image_status*. The receiver host PC also resets the next image and the array *next_image_status*.

The transmitter host PC checks for the UDP message queue after transmitting all the packets from the current image and 20 packets from the next image. This is done so the receiver controller can gather enough extra data to push all the packets in the current image to the receiver host PC. Upon receiving *current_image_status*, the transmitter host PC determines which packets are dropped and resends them. After resending the dropped packets, the transmitter host PC continues with packet transmissions of the next image until it receives another *current_image_status* update from the receiver system through the UDP interface, at which point it checks for dropped packets from the current image and re-sends them. This process is repeated until the transmitter host PC finds that all packets from the current image have been received by the receiver host PC. After this, the transmitter host can continue transmitting the next image without interruptions.

3.4.4 Transmission Statistics

The image transmission system allows users to visually detect transmission errors and transmission data rate by watching the image reconstruction process. In addition, the receiver GUI provides quantitative transmission statistics so that users can have a more accurate way to judge the transmission performance. All the performance measurements are updated on the GUI after the host PC polls the receiver controller.

Packet Error Checking

The receiver GUI allows users to compare received packets with the original transmitted data to detect any errors that may have occurred during the transmission process. Each of the original images used for transmission is broken down into a 120 by 240 matrix, with each row in the matrix corresponding to an image block. These matrices are saved in mat files. At start-up, the receiver GUI finds all available mat files, and the user can select the file that matches the transmitted data. If there is no matching mat file, the GUI can create a new mat file from a directory containing the transmitted image files. The new mat file can be used immediately for error checking. Alternatively, the user can also select to not check for transmission errors, and the transmission statistics related to packet errors will not be calculated.

Performance Measurements Parameters

All the performance measurements are listed below. Note that only data rate calculations are available when in transmitter feedback mode. The statistics regarding packet errors are only available when a mat file containing transmitted image information is provided to the receiver.

- *Data Rate*: Data rate is a measure of the transmission data rate. It is the ratio of the total amount received data and the time elapsed since the first byte of data was received. The speed of the wireless transmission system is largely limited by processing power of the transmitter and the receiver PCs.

- *Total Received Packets*: This measurement keeps track of the total number of successfully decoded packets. It includes the packets decoded with non-header payload errors. It is not available when using the feedback mode.

- *Total Expected Packets*: Since the image packets are transmitted in order, the number of transmitted packets can be calculated based on the image ID and the block number of the last successfully decoded packet. This is an estimate of the total number of packets that the receiver system expects to see. This measurement is not available when using the feedback mode.

- *%Dropped Packets*: This measurement is calculated by taking the ratio of the number of dropped packets to the number of total expected packets. The total number of dropped packets is obtained by taking the difference between the number of received packets and expected packets. This measurement is not available when using the feedback mode.

- *Total Header Errors*: A header error occurs when the receiver host PC detects a header that does not match the expected length of the payload. This measurement keeps track of the total number of header errors that has occurred.

- *%Header Errors*: The *%header errors* is the ratio of the total header errors and the total number of expected packets. This measures the rate of false detections made by the receiver node. It is not available when feedback mode is used.

- *Total Packet with Errors*: Sometimes, errors are introduced to non-header payload data during the transmission process, and transmitted bits are corrupted. *Total Packet with Errors* measures the total number of packets with such corrupted data. It is not available when using feedback, and it is not available when the users select not to check for transmission errors.

- *%Packet With Errors*: The *%packet with errors* is the ratio of the corrupted packets to the total number of received packets. This measures the rate of packet corruption.

This number is not available when using feedback, nor when the user selects to not check for transmission errors.

- *Total Pixels with Errors*: This keeps track of the total number of corrupted pixels that the receiver host PC received. This measurement is not available when using feedback, nor when the user selects to not check for transmission errors.

- *%Pixels with Errors*: This is the ratio of the corrupted pixels and the total number of pixels received so far. It measures the amount of corruption that can be detected with visual senses. This measurement is not available when using feedback, nor when the user selects to not check for transmission errors.

- *Total Bits with Errors*: This keeps track of the total number of corrupted bits received by the receiver host PC. It is not available when using feedback, nor when the user selects to not check for transmission errors.

- *%Bits with Errors*: This is the ratio of the corrupted bits to the total number of the bits received so far. It measures the rate of data corruption of the wireless transmission system. This measurement is not available when using feedback, nor when the user selects to not check for transmission errors.

3.5 Summary

This chapter describes the structure and implementation of the receiver system. The receiver host PC reconstructs the transmitted images from data demodulated by the receiver node. In addition, the receiver host PC provides an easy-to-use GUI. A feedback path is available between the receiver system and the transmitter system, improving overall transmission reliability. The high speed UWB wireless image transmission system is constructed with the receiver and the transmitter systems as described in this chapter and Chapter 2. The next chapter examines the performance of this wireless image transmission system for different transmission distances.

Chapter 4

Performance of the UWB Wireless Image Transmission System

This chapter evaluates the performance and reliability of the UWB wireless image transmission system. The system achieves a 95% packet reception rate and a 2×10^{-5} bit error rate for transmission distances up to 12 meters. With the addition of a reflective surface, which creates constructive multi-path and increases the signal-to-noise ratio, the system can operate at distances up to 16 meters. The UWB link has maximum data rating ranging from 2.67 Mb/s to 8.01 Mb/s, depending on the transmission distance.

The chapter first examines the effects of wireless channel attenuation and receiver gains. Next, the chapter describes various techniques used to improve the packet reception rate and the bit error rate for long distance transmissions. This includes techniques that reduce payload errors and increase receiver synchronization accuracies. The chapter then describes some limitations of the current system that prevent further increases in the transmission range. Then the maximum achievable data rates for the wireless transmission system are discussed. The chapter then summarizes the transmitter settings used and examines whether the transmitted signals violate spectral mask as specified by the FCC. Finally, different sets of receiver parameters used by the transmission system at various transmission distances are summarized, and overall system performances are presented.

4.1 Channel Attenuation and Gain

For wireless transmission systems, the transmitted signals are greatly attenuated while traveling through air. The amount of attenuation depends largely on the target transmission distance, and the transmission environment. Under ideal conditions, when signals are transmitted in unobstructed free space and there is no multi-path, the signal attenuation can be predicted by Friis formula, shown in equation 4.1 [18].

$$\frac{P_r}{P_t} = G_t G_r (\frac{\lambda}{4\pi R})^2 \tag{4.1}$$

The Friis formula cannot accurately predict the signal attenuation for the UWB wireless channel described in this thesis, since there are significant amount of signal reflections and multi-path effects. However, it shows insight into the relationship between path loss and the transmission distance: doubling the transmission distance quadruples the signal attenuation in the channel. Figure 4-1 shows the measured signal attenuation for the UWB wireless link for transmission distances up to 13 meters. The measurements are made with 4.0 GHz signals, using all-band dipole antennas. The expected signal attenuation ranges from 30 dB for short distance transmission to greater than 63 dB for distances greater than 13 meters. To accommodate the dramatic changes in signal strength, the UWB image transmission system described in this thesis is equipped with multiple sets of transmitter and receiver parameters, each optimized for a different transmission path loss.

4.1.1 Receiver Gain

The receiver RF front end can be programmed to one of six gain settings, scaling gain by approximately 40 dB. This programmability can be used to compensate for the path loss in the transmission channel. For long distance transmission, where large path loss is expected, the receiver gain needs to be high; however, for short range UWB transmission, a high receiver gain results in signals that saturate the receiver ADC. This hinders the ability of the baseband processor to accurately detect and synchronize to UWB packets. As a result,

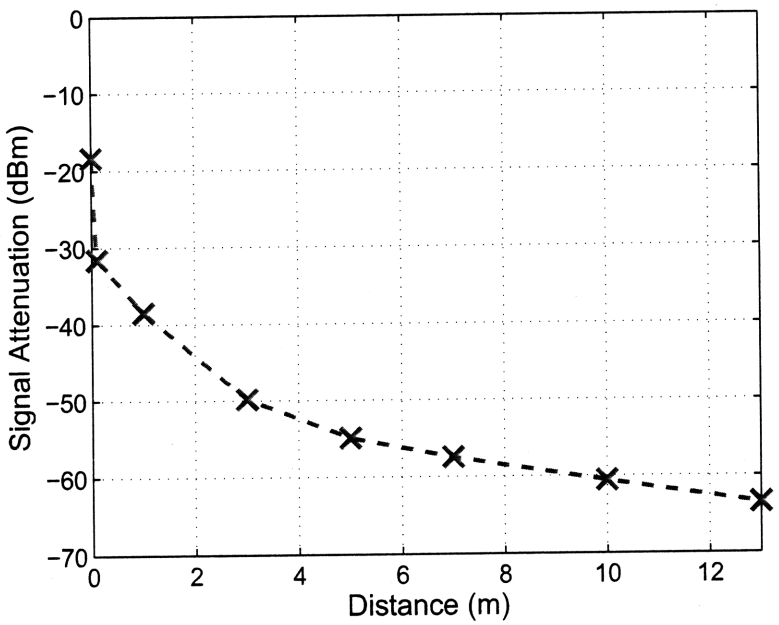


Figure 4-1: The Amount of Attenuation Expected from the UWB Transmission Channel using All-Band Dipole Antennas

Transmission Distance	Receiver Gain Settings
less than *0.1 m*	3
0.1 m to *0.75 m*	4
0.75 m to *1.5 m*	5
1.5 m to *16 m*	6

Table 4.1: Receiver Gain Settings for Different Target Transmission Ranges

different receiver gain settings are automatically selected by the GUI when the users specifies different transmission ranges, as shown in Table 4.1.

Path loss is not the only concern in this wireless system. The performance and reliability of the system is affected more by multi-path and noise. In a noise-free environment, when the transmitter output in connected to the receiver input via cable and attenuators, the packet reception rate is above 97% for attenuations up to 99 dB, as shown in Figure 4-2. A wired attenuation of 99 dB corresponds to path loss well beyond 13 meters, as shown by the signal attenuation measurements in Figure 4-1. However, the current system does not operate well beyond 12 meters, as the addition of noise dramatically degrades the performance of the

transmission system. A large receiver gain scales both the signal and noise, thus the signal-to-noise ratio seen by the receiver remains unchanged. In fact, the system can only achieve a transmission distance of 5 meters by increasing the receiver gain alone. Other receiver baseband parameters need to be modified to improve the performance and reliability of the UWB wireless image system.

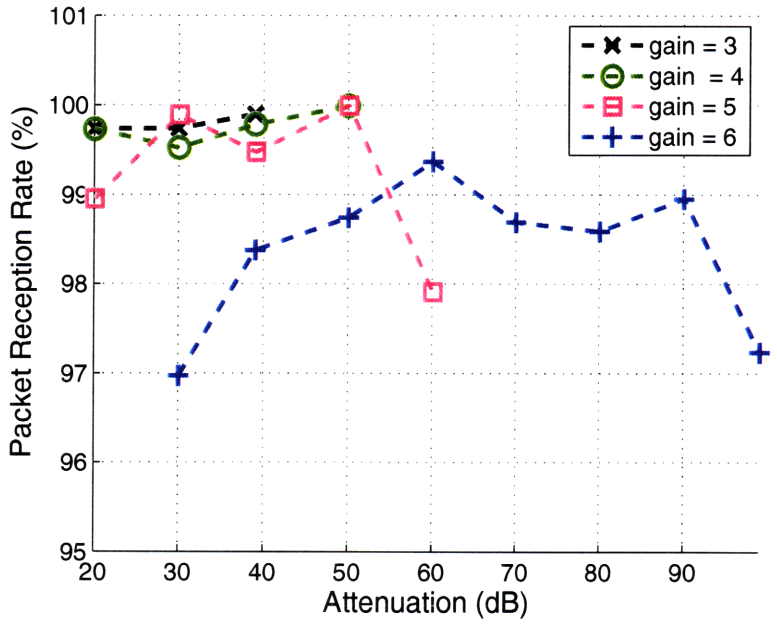
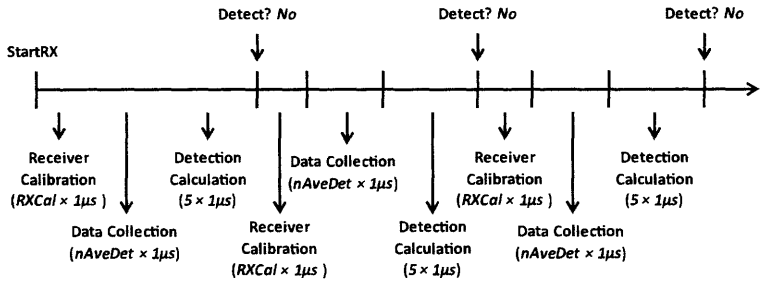


Figure 4-2: Packet Reception Rate in a Noise-Free Environment (Output of Transmitter Node Connected to Input of Receiver Node Via Cable)

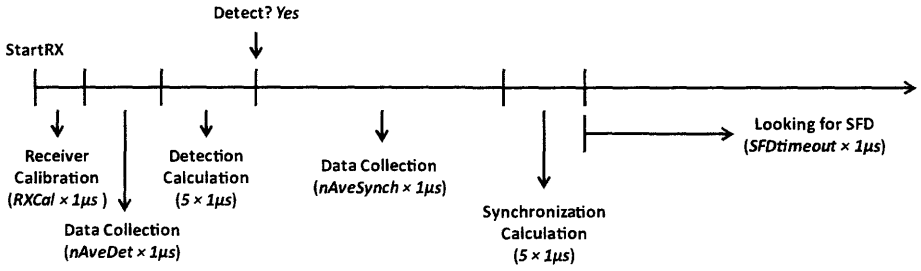
4.2 Reduction of Synchronization Errors

As transmission distances and path loss increase, the signal-to-noise ratio decreases, making it more difficult for the receiver to correctly detect and synchronize to an incoming packet. The receiver baseband processor correlates the received signal to values in a predetermined codebook for packet detection and synchronization, as shown in Figure 4-3. The receiver is triggered by a pulsed *startRX* signal. When triggered, the receiver first calibrates its RF front-end for a few micro-seconds, and then it collects data for $nAveDet \times 1\ \mu s$. Using the

collected data, the baseband processor calculates the received signal's correlations with the codebook, and determines whether a packet is present. This calculation takes approximately 5 μs. If a packet is not detected, the receiver begin the process again. If a packet is detected, the receiver spends $nAveSynch \times 1$ μs collecting more data for synchronization. After the data is collected, the receiver baseband processor spends another 5 μs calculating received signal's correlations with the codebook to find the correct phase. After the phase is determined, the receiver looks for the SFD for $SFDtimeout \times 1$ μs.



(a)

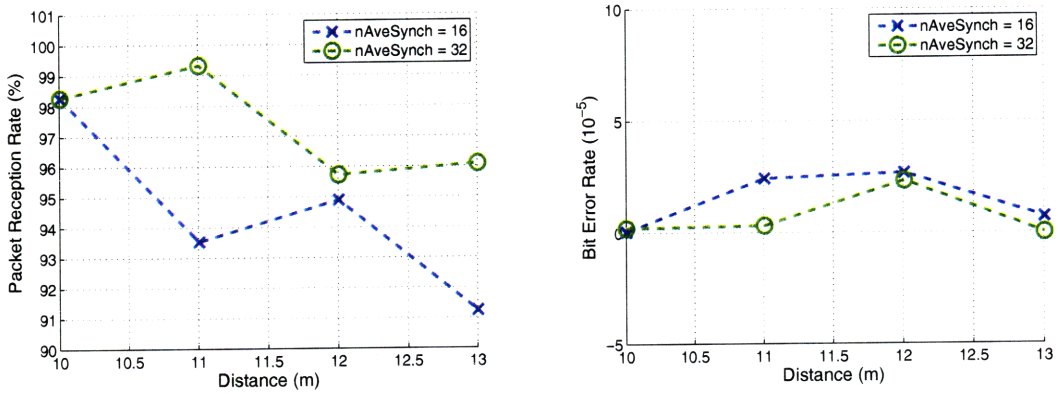


(b)

Figure 4-3: The Transition of Receiver States When (a): No Packet Detection Occurs, and (b): Packet Detection Occurs

The *nAveDet* and *nAveSynch* values are adjustable via shift registers, and they play a key role in the packet detection and synchronization process. As signal-to-noise ratio decreases, larger *nAveDet* and *nAveSynch* values allow the receiver to collect and average receiver signals over longer period of time. This helps to smooth out effect of noise, and allows the baseband processor to better detect a packet and synchronize to the transmitter

time window. The default *nAveDet* and *nAveSynch* values are 4 and 16, respectively. For transmission distances greater than 10 meters, the *nAveSynch* value is increased to 32. The increase in the synchronization data collection period improves both the packet reception rate and the bit error rate of the wireless transmission system, as seen in Figure 4-4.



(a) (b)

Figure 4-4: Impact of Increasing *nAveSynch* on (a): the Packet Reception Rate, and (b): the Bit Error Rate

4.2.1 Relating *nAveDet* and *nAveSynch* to Preamble Length

Apart from playing key roles in the detection and synchronization phases in the receiver baseband processor, *nAveDet* and *nAveSynch* also have a direct impact on the minimum required length of the preamble of the UWB packets. The preamble needs to be long enough for the receiver to finish synchronization calculations. A transmitted packet could arrive at the receiver anytime during the detection process. One example of this is shown in Figure 4-5. A packet arriving in the middle of the data collection period may not be detected, since the signal strength may not be high enough after averaging. Therefore, the receiver may fail to detect the packet in the first detection cycle. As a result, the preamble length needs to be long enough so that when the packet is detected in the second detection cycle, the receiver can still synchronize and find the SFD. Taking this into account, the optimal duration of the

preamble can be obtained by Equation 4.2.

$$preamble\_duration = nAveDet + 5 + RXCal + nAveDet + 5 + nAveSynch + 5 \quad (4.2)$$

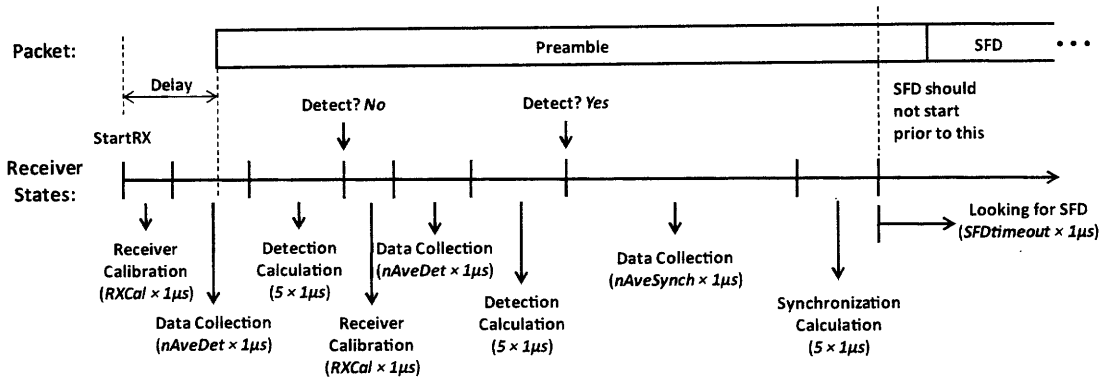


Figure 4-5: An Example of Packet Arriving at the Receiver During the Detection Data Collection State

It takes 1 μs to send one repetition of the 32-bit preamble inner code, therefore, the calculated *preamble_duration* is the number of times that the preamble inner code should be repeated. An optimal *SFDtimeout* value can be empirically determined once the optimal inner code repetition value is known. For transmission distances less than 10 meters, a *nAveDet* value of 4 and a *nAveSynch* value of 16 are used. For transmission distances greater than 10 meters, the *nAveSynch* value is increased to 32. Table 4.2 shows the optimal preamble length and *SFDtimeout* values for these settings.

Parameters	Short Transmission Range	Long Transmission Range
nAveDet	4	16
nAveSynch	4	32
SFDtimeOut	16	24
preamble_length	40	56

Table 4.2: Optimal Preamble Length and *SFDtimeout* Values for Selected *nAveDet* and *nAveSynch* Pairings

The relationship among *nAveDet*, *nAveSynch*, *SFDtimeout*, and *preamble_length* are verified experimentally to ensure that the receiver node can correctly detect, synchronize, and

decode all transmitted packets regardless of what state the receiver is in. As shown in Figure 4-6, the experiment is conducted under noise-free conditions, with a cable connecting the output of the transmitter to the input of the receiver. In addition, the transmitter generates a pulsed *startRX* signal before transmitting each packet. This *startRX* signal is sent to the receiver via a cable. The transmitter controls the amount of delays between the start of the pulsed *startRX* signal and the start of the transmitted packets, which regulates when a particular transmitted packet will arrive in the receiver detection process (see Figure 4-5). The effects of the packet arrival time on the receiver detection and synchronization algorithms can be studied using this setup.

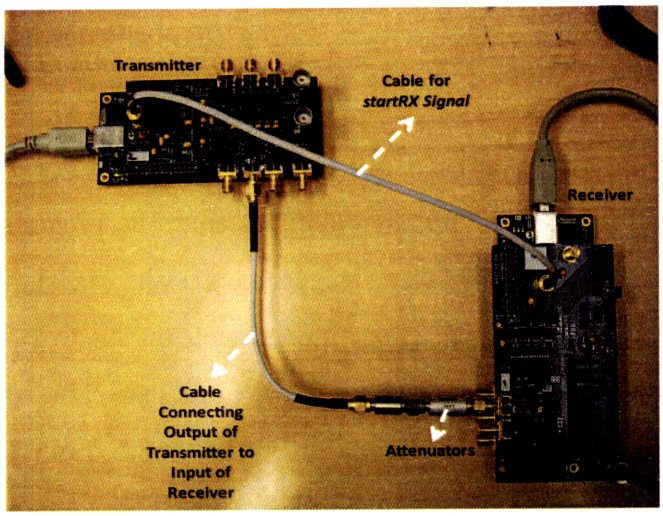
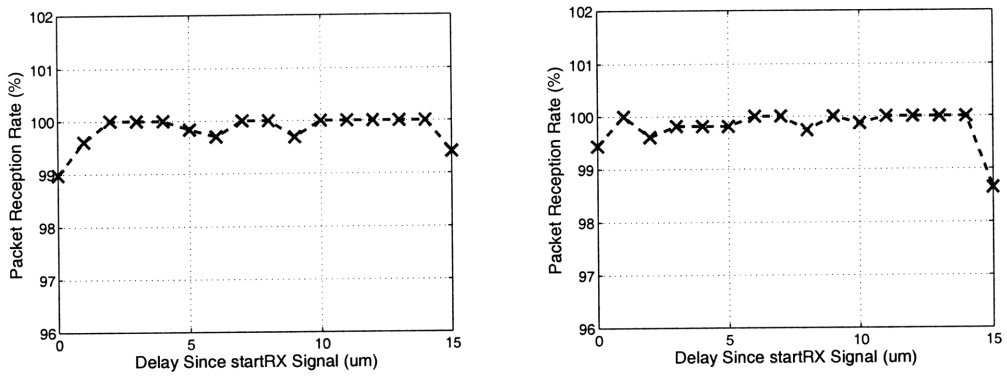


Figure 4-6: Experimental Setup for Transmitter Trigger Receiver in a Noise-Free Transmission Environment

Figure 4-7 shows the packet reception rates for delays up to 15 μs, using values presented in Table 4.2. For both cases, the packet reception rate remains above 98%, which suggests that in a noise-free environment, operating with the values given, the receiver node can correctly detect, synchronize, and decode all packets regardless of what state the receiver is in.



(a) $nAveDet = 4$, $nAveSynch = 16$, $SFDtime\text{-}out = 20$, *Preamble Length* $= 40$

(b) $nAveDet = 4$, $nAveSynch = 32$, $SFDtime\text{-}out = 24$, *Preamble Length* $= 56$

Figure 4-7: Packet Reception Rate vs. Time Delays in Packet Arrival Since *startRX* Signal

4.3 Reduction of Payload Errors

The presence of noise in the wireless channel degrades the performance of the transmission systems. The UWB system can transmit data with bit error rate of 0 in a noise-free environment, created by connecting the output of the transmitter to the input of the receiver via cables and attenuators. When the system goes wireless, the addition of noise can cause the receiver to incorrectly decode data. As transmission distances increase, the signal-to-noise ratio at the receiver is reduced, resulting in an increase in bit error rate and packet error rate. A bit error in the header generates a header error, which forces the receiver to discard the entire packet. Payload averaging is a technique that reduces bit errors in the presence of noise, thus improving the performance of the UWB wireless system.

Payload averaging affects only PPM data and is controlled by the *nSampSlot* parameter in the receiver baseband settings. In the original PPM timing, a 124.8 ns time window is divided into two 62.4 ns time slots, T_1 and T_2. The received bit value is determined based on which time slot contains greater energy. With payload averaging, each time slot is repeated *nSampSlot* times, as shown in Figure 4-8. This allows receiver to collect and average the received UWB signals over a longer period of time, thus reducing the effects of sudden spikes in noise and increasing decoding accuracy.

In the UWB wireless image transmission system, payload averaging is necessary for trans-

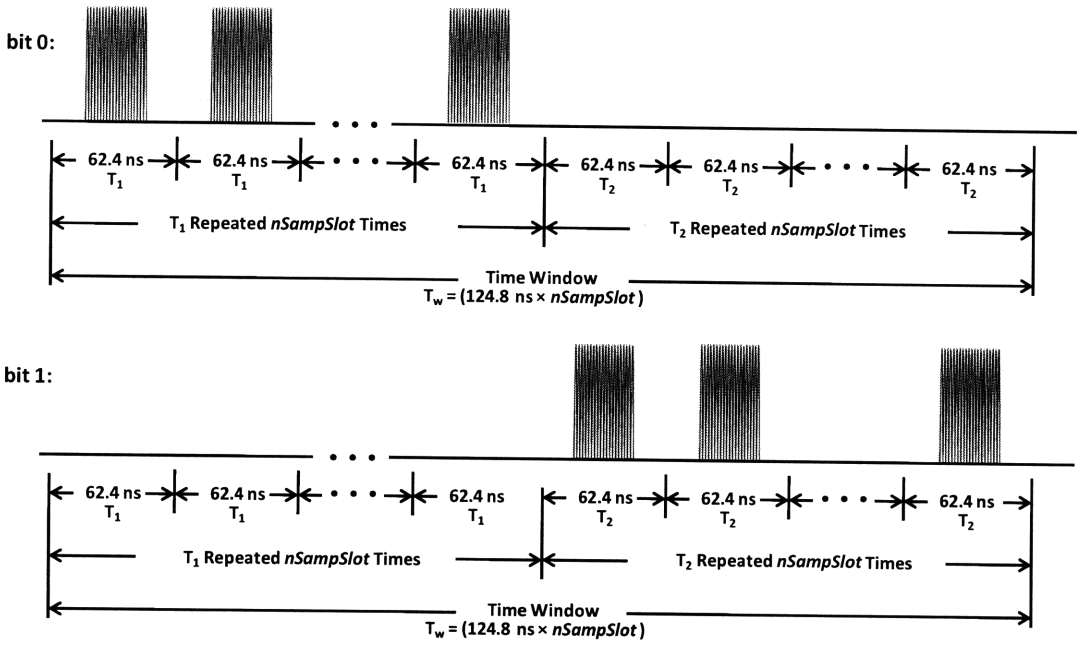


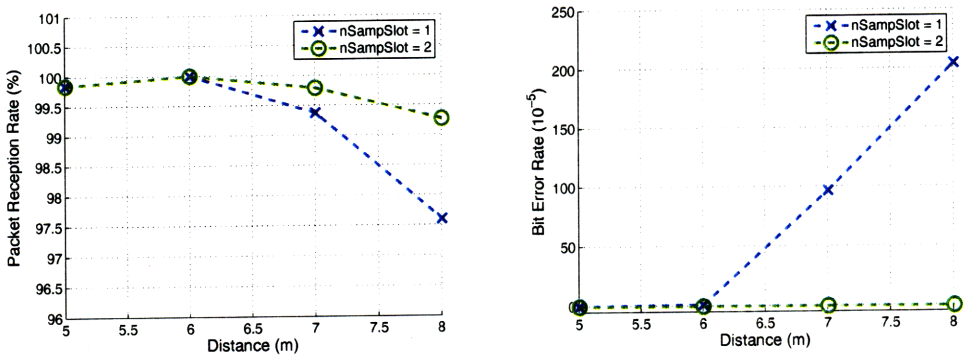
Figure 4-8: PPM Data with Payload Averaging

mission distances above 6 meters. Figure 4-9 shows impact of payload averaging on the bit error rate and packet reception rate. As seen in the figure, the bit error rate and packet reception rate can be dramatically improved with *nSampSlot* of 2, especially at transmission distance of 7 and 8 meters. At transmission distance of 11 meters, the *nSampSlot* value is increased again to 3. Table 4.3 shows the *nSampSlot* values used for different transmission ranges.

Transmission Distance	***nSampSlot***
less than *5 m*	1
5 m to *11 m*	2
11 m to *16 m*	3

Table 4.3: *nSampSlot* Values for Different Target Transmission Ranges

There is a trade-off between the maximum *nSampSlot* allowed and the length of the payload, due to clock drift. Large *nSampSlot* values increase the length of the packet. While receiving a packet, the synchronized receiver integration window begins to drift away from the transmitter time window, due to the imperfect matching between the transmitter and receiver clock periods. Thus the bits at the end of the payload cannot be correctly decoded,



(a) (b)

Figure 4-9: Impact of Payload Averaging on (a): the Packet Reception Rate, and (b): the Bit Error Rate

Figure 4-10: Received Image with Payload Length of 123 bytes and *nSampSlot* of 5

as shown in Figure 4-10. With payload length of 123 bytes, the *nSampSlot* is limited to 4. This limit can be increased if the payload length is reduced. However, such reduction decreases the ratio of payload length to the preamble length, which reduces the effective transmission data rate of the system.

4.4 Limitations to the System

As transmission distance increases, the signal-to-noise ratio decreases, and it becomes harder for the receiver baseband to detect incoming packets. The receiver baseband processor detects a packet by collecting received signal for $nAveDet \times 1\ \mu s$, averages the signal, and

correlates it with value in a predetermined codebook. A packet is detected if the correlation is over a certain threshold. For transmission distances beyond 12 meters, the signal-to-noise ratio becomes so low that the receiver baseband can no longer detect any incoming packets.

At this transmission range, some UWB packets can still be detected if the detection threshold is lowered significantly to values below 0. The low detection threshold forces the baseband processor to constantly detect and achieve synchronization, even when there is no UWB packet present. This allows the receiver to recover the transmitted packets whose arrivals coincide with the start of receiver detection phase. Otherwise, the receiver baseband processor misses the packets, as it tries to synchronize and decode noise signals. Thus, the packet reception rate is extremely low (below 10%), and the receiver power consumption increases dramatically, as it is constantly trying to synchronize to received signals.

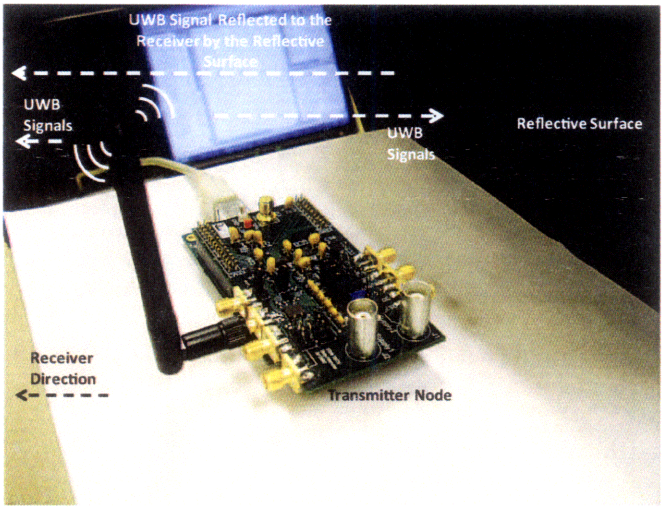


Figure 4-11: Transmitter with the Addition of a Reflective Surface

Currently, to increase the transmission distance beyond 12 meters, a reflective surface is placed behind the transmitter, as shown in Figure 4-11. The surface reflects some of the UWB signals emitted by the antenna toward the direction of the receiver. The presence of the reflective surface creates constructive multi-path in the UWB channel, and increases the amount of UWB signal seen by the receiver, thus increasing the signal-to-noise ratio and allowing the receiver to correctly detect incoming packets with the default detection

threshold of 0.4. The addition of the reflective surface pushes the transmission distance of the current system to 16 meters.

4.5 Transmission Data Rate

Without payload averaging, the PPM time window is 124.8 ns, which results in a peak data rate of 8.01 Mb/s. With *nSampSlot* of 3, a single bit spans 374.4 ns, which results in a peak data rate of 2.67 Mb/s. The average transmission data rates for the UWB transmission link are lower than the peak data rates, due to presence of auxiliary information in the packet structure, such as the preamble, the SFD, and the header. These components are necessary for detection and synchronization of the wireless packet; however, they do not contain actual data. Factoring in these, the UWB link can achieve maximum average transmission rates ranging from 2.28 Mb/s to 5.83 Mb/s, depending on the target transmission range, as shown in Table 4.4,

Transmission Distance	**Settings**		**Transmission Time Per Packet**	**Average Data Rate**
	Preamble Length	*Payload Averaging*		
< 5m	40	none	123.8 μs	5.83 Mb/s
5m - 10m	40	2	292.5 μs	3.36 Mb/s
10m - 11m	56	2	308.5 μs	3.19 Mb/s
11m - 16m	56	3	432.3 μs	2.28 Mb/s

Table 4.4: Average UWB Wireless Link Data Rate for Different Transmission Range

4.6 FCC Compliance

FCC places limits on both the peak and average radiated power for the UWB signals [2]. In the 3.1-to-10.6 GHz band, the average power spectral density must be below -41.3 dBm/MHz. The peak power is limited to 0 dBm, measured within 50 MHz resolution bandwidth (RBW) window around the UWB signal's center frequency. Both the average and peak power measurements can be obtained using a spectrum analyzer. It is important to measure the power

emitted by the transmitter in the current system, and compare the measurements with those specified by the FCC.

Most spectrum analyzers are not equipped with a 50 MHz frequency filter, therefore, the peak power measurement is usually performed using at a lower RBW. The peak power limit is then converted to $P_{pk} \leq 20log_{10}\frac{RBW}{50MHz}$. This is a very conservative conversion. The peak power measurements shown in this section are made with RBW of 8 MHz, thus the peak power limit is set to -15.9 dBm.

Four different transmitter settings are used in this UWB wireless transmission system. These settings and their respective peak power measurements are shown in Table 4.5. It can be seen from the table that the peak power emissions of the current transmitter exceeds the peak power limit of -15.9 dBm. However, as mentioned before, the conversion equation used is very conservative. Taking this into account, the peak power emitted by the transmitter in the current system is very close to the peak power limit set by the FCC.

System Range	**Settings**		Peak Power
	Preamble Repetition	*Payload Averaging*	
< 5m	40	none	-11.732 dBm
5m to 10m	40	2	-12.020 dBm
10m to 11m	40	3	-11.836 dBm
11m to 16m	56	3	-12.010 dBm

Table 4.5: Transmitter Settings and their Respective Peak Power Emission Measurements

Figure 4-12 shows the average power spectral density of the transmitted UWB signals. The UWB transmitter is compliant with the FCC average power requirements when the transmitted signal has preamble length of 40 without payload averaging. As the transmission distance increases beyond 5 meters, the transmitter parameters are modified to ensure reliable performance. This increases the average power emitted by the transmitter, resulting in violations with the FCC mask in the 0.96-to-1.61 GHz frequency band.

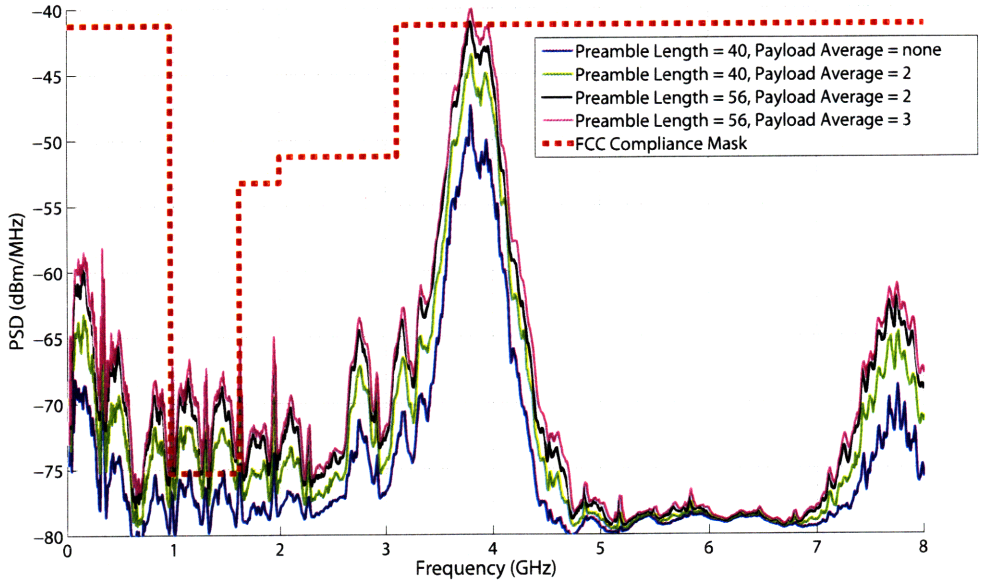


Figure 4-12: Average Power Spectrum Density of the Transmitted UWB Signals

4.7 Summary

To summarize, this chapter describes various techniques used to improve the packet reception rate and the bit error rate of the UWB wireless image transmission system. Parameters such as receiver gain, detection threshold, payload averaging, and receiver synchronization data collection period (*nAveSynch*) are modified based on target transmission distance. Changes in these parameters also affects the UWB packet structure, which are specified in the transmitter system. Table 4.6 shows the finalized settings used for the transmitter, while Table 4.7 shows the finalized settings used for the receiver. With these settings, the UWB wireless system achieves packet reception rate above 95%, and bit error rate below 2×10^{-5} for transmission distances up to 12 meters. Beyond 12 meters, the addition of a reflective surface to the back of the transmitter node can further boost the transmitted signal strength, and increase the transmission distance to 16 meters. Figure 4-13 and 4-14 show the details of these system performance measurements.

System Range	Settings	
	Preamble Repetition	*Payload Averaging*
< 5m	40	none
5m to 10m	40	2
10m to 11m	56	2
11m to 16m	56	3

Table 4.6: Transmitter Settings for Transmission System Range Up To 16m

System Range	Settings			
	Gain	*nAveSynch*	*SFDtimeout*	*Payload Averaging*
< 0.1m	3	16	20	none
0.1m to 0.75m	4	16	20	none
0.75m to 1.75m	5	16	20	none
1.75m to 5m	6	16	20	none
5m to 10m	6	16	20	2
10m to 11m	6	32	24	2
11m to 16m	6	32	24	3

Table 4.7: Receiver Settings for Transmission System Range Up To 16m

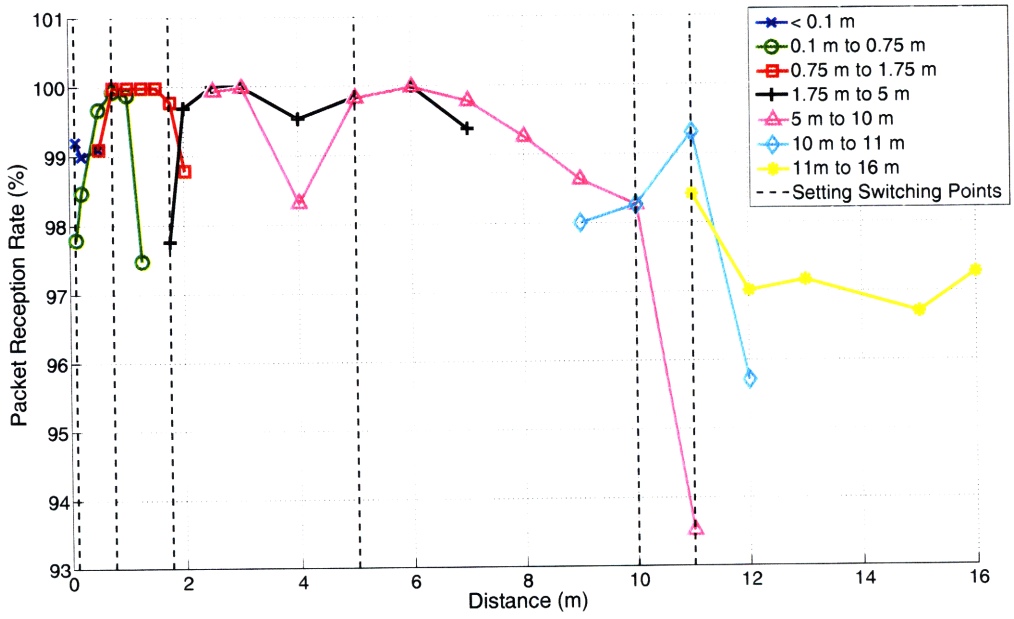


Figure 4-13: Packer Reception Rate for the UWB Wireless Image Transmission System Using Settings Shown in Tables 4.7 and 4.6.

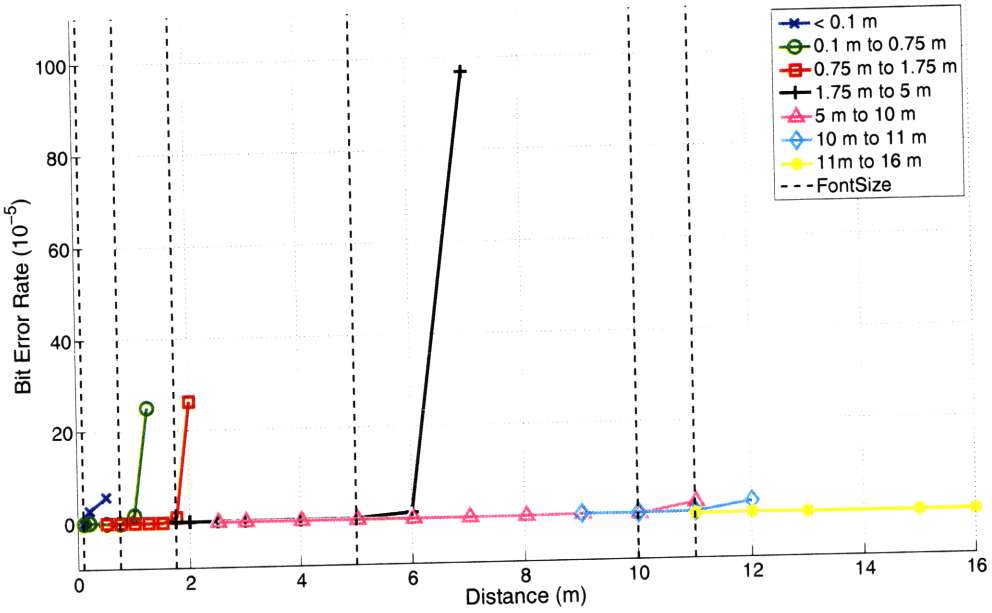


Figure 4-14: Bit Error Rate for the UWB Wireless Image Transmission System Using Settings Shown in Tables 4.7 and 4.6.

Chapter 5

Conclusion

5.1 Summary of Thesis

Ultra-wideband technologies offer the promise of wireless transmission with low power consumption, and allow for re-use of frequency spectrum that are already assigned to other wireless transmissions, thereby alleviating the problem of overcrowding the spectrum. In particular, pulse-based UWB technology allows for building of ultra-low power, medium- to long-range transceivers, at the expense of data rate. Now a part of IEEE 802.15.4 standard, pulse-based UWB technology has great potential for use in WPAN devices, freeing people from wires, and enabling transmission of video, audio, and data among many portable devices.

This thesis demonstrates a wireless UWB image transmission system. The system features a unidirectional UWB link, built using pulse-based, non-coherent UWB transmitter and receiver chips [17] [14] [15]. Both the transmitter and the receiver chips are controlled by their respective host PCs, through Opal Kelly XEM3001v2 FPGA integration modules. The pictures transmitted in this system are 120 $\times$ 160 pixel bitmap or PNG images, with 12 bit RGB color depth. Each image is separated into 240 blocks of 120 bytes. Each block is proceeded by an image ID, and a block number, resulting in payload that is 123 bytes long. Predetermined preamble sequences, along with a SFD and a header, are sent before the

payload, so the receiver system can detect and synchronize to the incoming packet. Once the payload is demodulated on the receiver station, the image ID and the block number are stripped and read, and the image block is inserted into the reconstructed image at the specified location.

The unidirectional UWB link is supplemented with a feedback path. This feedback path is established through an internet connection between the transmitter and receiver host PCs, and is used for error correction purposes. The receiver host PC provides the transmitter host PC with a list of packets that were not received, so the transmitter system can resend these packets, thus improving overall transmission performance.

The UWB image transmission system achieves a peak transmission data rate of 8.01 Mb/s. By increasing the receiver gain settings and receiver synchronization data collection period, and employing techniques such as payload averaging, the system can transmit reliably with packet reception rate above 95% and bit error rate below 2×10^{-5}, for distances up to 12 meters. The system can operate up to 16 meters, with the addition of a reflect surface to the transmitter to boost the signal strength. For transmission distances greater than 10 meters, the presence of the reflective surface almost always improve the system performance. At 16 meters, the system has a peak transmission data rate of 2.67 Mbps, and a maximum average data rate of 2.28 Mbps.

5.2 Future Work

There are three main ways that the current wireless UWB image transmission system can be improved. The current system only handles 120×160 images. The receiver system has prior knowledge to the size of the image sent, and uses this information for image reconstruction. In the future, the system can be improved to handle images of various sizes. The receiver system should have no prior knowledge to the dimension of the images. This information should be among the data transmitted to the receiver system. Also, the image compression scheme used in the current system is very simple: it masks off the lower 4 bits of 8-bit RGB values. To further reduce the size of each image, more complex compression algorithms, such

as MPEG and H264, can be implemented. These image compression algorithms are very computation intensive, and the compression process can be very time consuming. Therefore, specialized hardware support should be added to the transmission system if such image compression schemes are used.

In the current system, the average transmission data rate is largely limited by the processing speeds of the transmitter and receiver host PCs, as well as the data transfer rate between the host PCs and their respective Opal Kelly XEM3001v2 modules. This two factors hinders the realization of the maximum 8 Mb/s data rate that the UWB wireless link is capable of achieving. To increase effective data rate, time intensive computations can be moved from the host PCs to the FPGA on the XEM3001v2 modules, so they can be parallelized and executed in real time. For example, the transmitter host PC generates *startTX* signal sequences, and then transfers them to the XEM3001v2 module, which relays the information to the transmitter chip. This process can be parallelized if the *startTX* signals are generated on the XEM3001v2 FPGA and relayed to the transmitter chip at the same time. Similarly, in the receiver system, the image reconstruction process can be implemented on the XEM3001v2 FPGA, so demodulated bits can be placed in an image as they becomes available. Moving these operations to the FPGA allows for parallel processing, and should eliminate the performance bottleneck in the current system.

The current system can also be expanded by developing a Media Access Controller (MAC) to fully support the 802.15.4 standard. The MAC controller can be written in MATLAB, and need to be allow time sharing among multiple transmitter and receiver nodes. In addition, the current system lacks ranging capabilities, since the UWB link is unidirectional. In the future, a bidirectional UWB link can be implemented, and the system can be expanded to support ranging.

Appendix A

Instructions for Using the UWB Wireless Image Transmission System

The files needed for implementing both the transmitter and the receiver systems can be found on a CD. There are two file folders: *tx_2008* which implements the transmitter, and *verilog_ok_rx2008* which implements the receiver. The directory structures are described below. Opal Kelly FrontPanel software and drivers need to be installed on both the transmitter and receiver host PC. The software can be found online at http://www.opalkelly.com.

A.1 Directory Structures

A.1.1 Transmitter

The *tx_2008* directory contains all the files needed to set up the transmitter system. Various Verilog files, including pre-compiled bit file that implements the transmitter controller on the XEM3001v2 FPGA, can be found in the top level of the directory. The MATLAB files that implement the transmitter GUI can be reached by navigating to *./matlab/video_tx_2008*. This is the home directory for the transmitter GUI. From here, the transmitter GUI can be started by typing "TX_v2" into MATLAB command line. The directories containing relevant transmitter shift register files can also be found here.

A.1.2 Receiver

The receiver directory structure is similar to that of the transmitter. The *verilog_ok_rx2008* directory contains all the files needed to set up the receiver system. Various Verilog files, including pre-compiled bit file that implements the receiver controller on the XEM3001v2 FPGA, can be found in the top level of the directory. The MATLAB files that implement the receiver GUI can be reached by navigating to *./matlab/rx_gui_2008*. This is the home directory for the receiver GUI. From here, the receiver GUI can be started by typing "RX_v2" into MATLAB command line. Some relevant receiver shift register files can be found in the *./shift_reg* directory.

A.2 Transmitter Setup

1. Attach the transmitter PCB board on top of the Opal Kelly XEM3001v2 FPGA integration module as shown in Figure 2-2 in Chapter 2.

2. Connect the Opal Kelly XEM3001v2 FPGA module to the transmitter host PC via a USB cable. Make sure only one module is connected to the PC.

3. On the transmitter host PC, open MATLAB and load the transmitter GUI. A screenshot of the GUI at start up is shown in Figure A-1, with key features labeled.

4. If a Opal Kelly XEM3001v2 FPGA module is connected to the PC, its serial number should be displayed the textbox under "Select the Transmitter", located on the top left corner of the GUI, in the "Transmitter Setup" panel. If the host PC finds no or multiple Opal Kelly modules, the text in the box will change to reflect this, and key features of the GUI, such as the "Transmit Packets" button, will be grayed out. When this occurs, please close the transmitter GUI, make sure only one Opal Kelly module is connected to the host PC, and then start the GUI again.

5. Enter the location of a desired bit file that implements the transmitter controller on the Opal Kelly XEM3001v2 FPGA module, or use the default bit file location provided

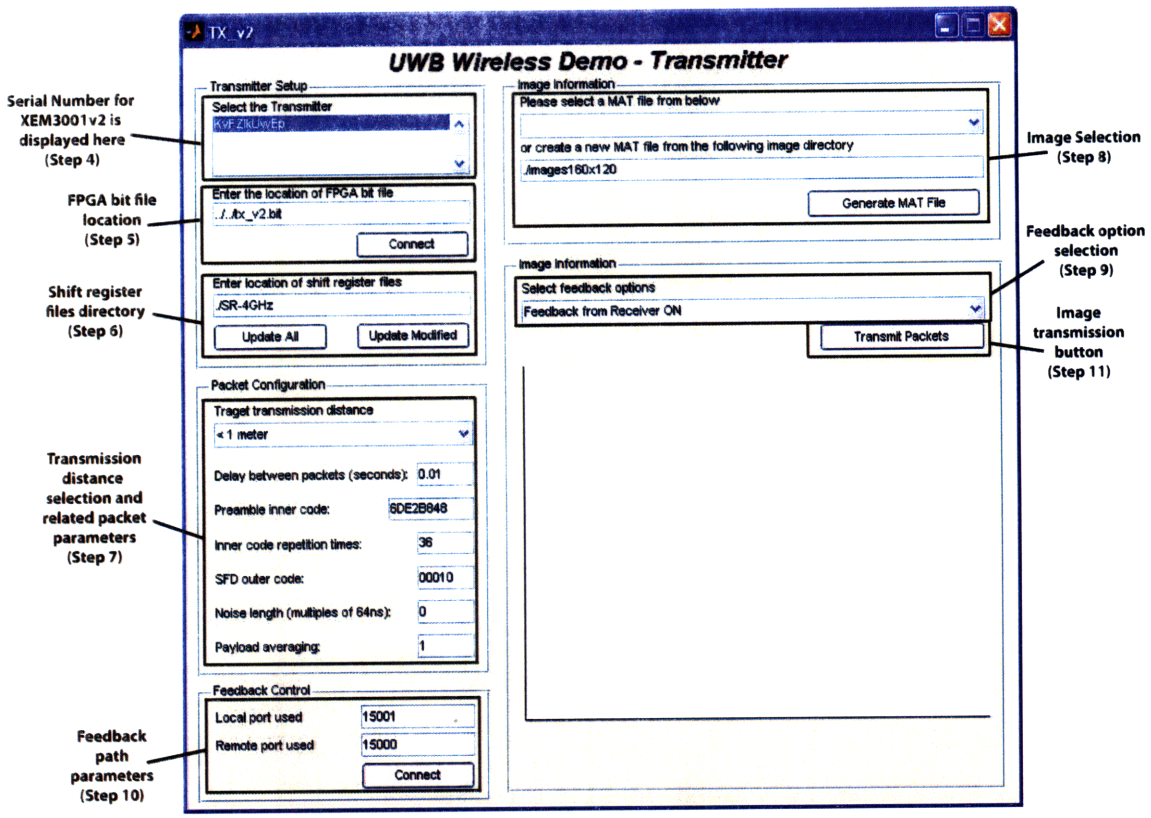


Figure A-1: Screenshot of the Transmitter GUI on Startup

by the GUI. Press the "Connect" button.

6. Enter the directory that contains a set of csv files with values to program the shift registers on the transmitter node, or use the default directory provided by the GUI. Press the "Update All" button.

7. In the drop-list located under text "Target transmission distance" in the "Packet Configuration" panel, select a desired target range for the image transmission system. There are three options, as shown in Figure A-2. The selection changes the values of the packet parameters used in the transmission system. These parameters are also displayed below the drop-list. A user can also change these values at will to improve the performance of the wireless link for a given transmission environment. The *preamble inner code* and the *SFD outer code* must match those on the receiver GUI, and *payload*

averaging must match the *nSampSlot* parameter on the receiver GUI.

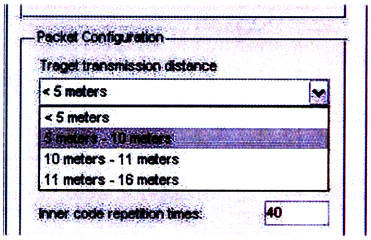


Figure A-2: Target Transmission Range Options on the Transmitter GUI

8. On the top right section of the transmitter GUI, in the "Image Information" panel, select a desired mat file from the drop-list. These mat files contain sets of compressed images that can be sent via the UWB wireless link. New mat files can also be created from raw images. This can be done by entering the directory that contains the desired 120 $\times$ 160 images into the textbox, and clicking the "Generate MAT File" button. The transmitter GUI places the generated mat file in the './mat_files' directory. The new mat file can now be selected from the droplist.

9. The feedback option establishes a UDP connection between the receiver host PC and the transmitter host PC. The receiver host PC uses the feedback path to inform the transmitter host PC which packets are dropped in the wireless transmission and should be send again. The feedback option can be selected from the drop-list located under "Select feedback options".

10. If feedback path is not desired, skip this step. Otherwise, enter two distinct numbers as the local and remote ports in the UDP connection. The numbers can be entered into textboxes located on the lower left side of the transmitter GUI, in the "Feedback Control" panel. By default, the local port is 15001, and the remote port is 15000. Click the "Connect" button to establish the UDP connection.

11. Wait until the receiver GUI is properly set up. Once the receiver system is in the receiving mode, click "Transmit Packets" button on the transmitter GUI to start UWB transmission. The image that is currently being transmitted should appear, and the

rate of data transmission is displayed on the lower left corner, as shown in Figure 2-3 in Chapter 2.

A.3 Receiver Setup

1. Attach the receiver PCB board on top of the Opal Kelly XEM3001v2 FPGA integration module, and a 1.2 V power supply to the receiver PCB, as shown in Figure 3-3 in Chapter 3.

2. Connect the Opal Kelly XEM3001v2 FPGA module to the receiver host PC via a USB cable. Make sure only one module is connected to the PC.

3. On the receiver host PC, open MATLAB and load the receiver GUI. A screenshot of the GUI upon startup is shown in Figure A-3, with key features labeled.

4. If a Opal Kelly XEM3001v2 FPGA module is connected to the PC, its serial number should be displayed the textbox under text "Select the Receiver", located on the top left corner of the GUI, in the "Device Configuration" panel. If the host PC finds no or multiple Opal Kelly modules, the textbox will change its text to warn the users, and key features of the GUI, such as the "Start Receiving" button, will be grayed out. When this occurs, please close the receiver GUI, make sure only one Opal Kelly module is connected to the host PC, and then start the GUI again.

5. Enter the location of a desired bit file that implements the receiver controller on the Opal Kelly XEM3001v2 FPGA module, or use the default bit file location provided by the GUI.

6. Enter the location of a csv file with values to program the shift registers on the receiver node, or use the default location provided by the GUI. Click the "Quick Connect" button to load the bit file from the previous step to the FPGA module, and program the shift registers on the receiver node.

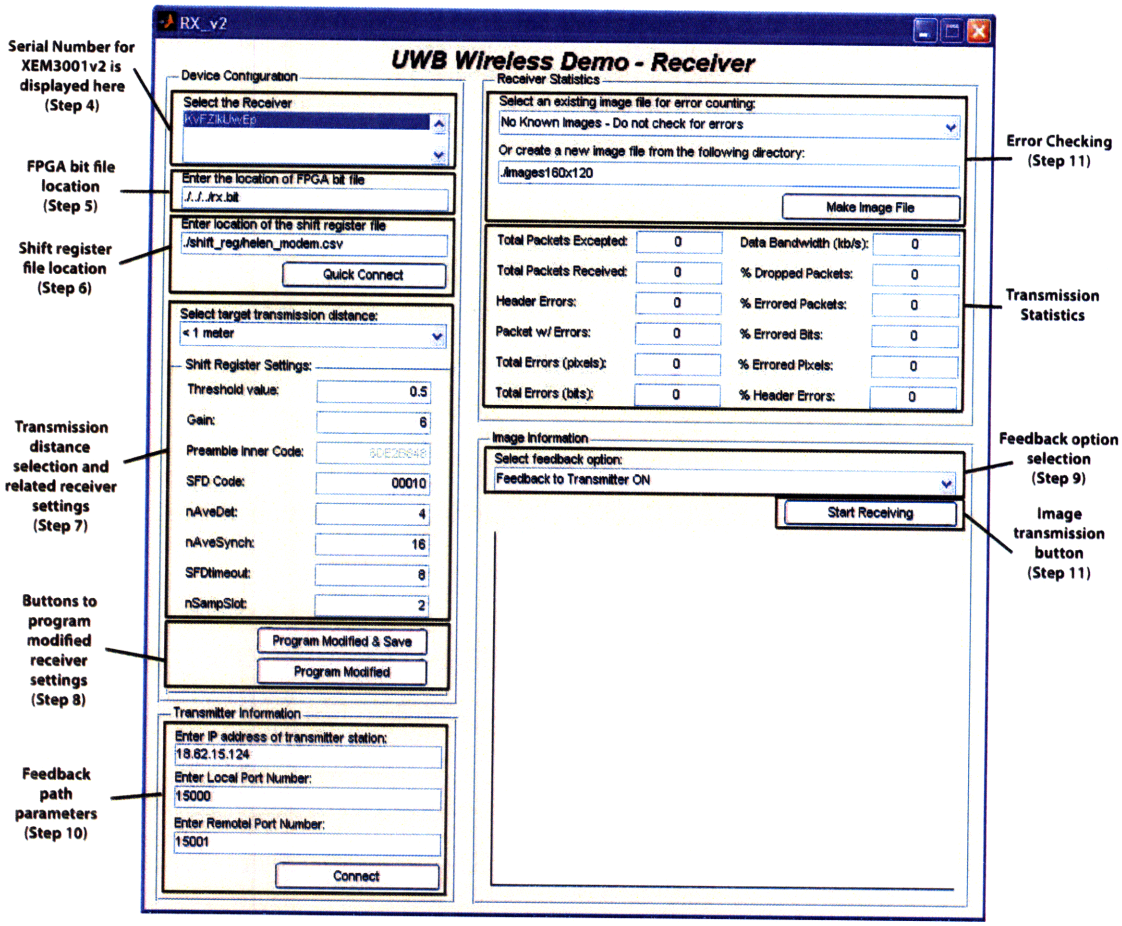


Figure A-3: Screenshot of the Receiver GUI

7. In the drop-list located under "Select target transmission distance", select a desired target range for the image transmission system. There are six options, as shown in Figure A-4. This selection changes the values of the receiver parameters used. These values are also shown beneath the droplist. If the option 'none' is selected, the receiver settings are programmed by values from the shift register file instead. A user can also change these parameters at will to improve the performance of the wireless link for a given transmission environment. However, the *preamble inner code* and the *SFD outer code* must match those on the transmitter GUI, and the *nSampSlot* must match the *payload averaging* parameter on the transmitter GUI.

8. Click the "Program Modified" button to load the modified receiver parameters into

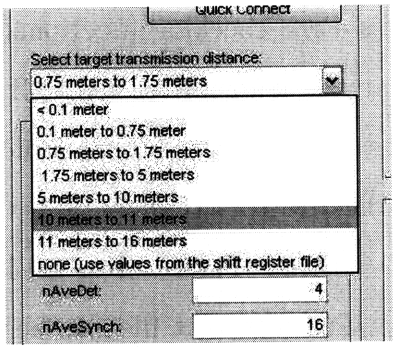


Figure A-4: Target Transmission Range Options on the Receiver GUI

the shift registers on the receiver node. If a user wishes to save the modified receiver parameters into another csv file, s/he could click the "Program Modified & Save" button instead.

9. The feedback option establishes a UDP connection between the receiver host PC and the transmitter host PC. The receiver host PC uses this feedback path to inform the transmitter host PC which packets are dropped in the wireless transmission process and should be send again. The feedback option can be selected from the drop-list located under text "Select feedback options". If this feedback path is not desired, then skip the next step.

10. If the feedback path is desired, enter the IP address of the transmitter host PC, and two distinct numbers as the local and remote port in the UDP connection. These can be entered into textboxes located on the lower left side of the receiver GUI, in the "Transmitter Information" panel. By default, the local port is 15000, and the remote port is 15001. The numbers can be changed, but the local port number here needs to match the remote port number in the transmitter GUI, and the remote port number here needs to match the local port number in the transmitter GUI. Click the "Connect" button below the textboxes to establish the UDP connection. Skip the next step.

11. If the feedback path is not desired, the user has the option of check the received image information against the original images, to detect any errors in data transmission. Select a mat file from the droplist in the "Receiver Statistics" panel on the upper right

side of the GUI to use this error checking functionality. These mat files contain sets of compressed images that the transmitter sent. If the transmitted images are not represented by one of these mat files, a new mat file can be created. In the textbox below the drop-list, enter the directory that contains the transmitted images, then click the "Generate MAT File" button. The receiver GUI places the generated mat file in the './mat_files' directory. The new mat file can be selected from the droplist. This error checking functionality can be disabled by selecting "No Error Checking" option from the droplist.

12. Click the "Start Receiving" button to place the receiver system in receiving mode. On the transmitter system, start sending UWB packets. Wait for the packets to arrive at the receiver system. Reconstructed images, along with the associated transmission statistics, should be displayed on the receiver GUI, as shown in Figure 3-7 in Chapter 3.

Appendix B

List of Acronyms

UWB Ultra-Wideband

SPI Serial Peripheral Interface

FPGA Field Programmable Gate Array

ACC Analog to Digital Converters

FIFO First-In-First-Out

PPM Pulse Position Modulation

API Application Programming Interface

GUI Graphic User Interface

OOK On-Off Keying

PCB Printed Circuit Board

PLL Phase Locked Loop

SFD Start of Frame Delimiter

PAM Pulse Amplitude Modulation

BPSK Binary Phase Shift Keying

FCC Federal Communications Commission

WPAN Wireless Personal Area Networks

CMOS Complimentary Metal Oxide Semiconductor

OFDM Orthogonal Frequency Division Multiplexing

RGB Red Green Blue

UDP User Datagram Protocol

JTAG Joint Task Action Group

DCM Digital Clock Manager

PROM Programmable Read-Only Memory

IEEE International Electrical and Electronic Engineers

MAC Media Access Controller

Bibliography

[1] G. R. Aiello and G. D. Rogerson, "Ultra-Wideband Wireless Systems," *IEEE Microwave Magazine*, vol. 4, no. 2, pp. 36–47, Jun. 2003.

[2] "First Report and Order," FCC 02-48, Federal Communications Commission, ET Docket 98-153, Feb. 2002.

[3] P. P. Mercier, "An All-Digital Transmitter for Pulsed Ultra-Wideband Communication," Master's project, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2008.

[4] "How It Works: UWB, WPAN, and WiMedia Radio Space," White Paper by WiMedia Alliances, Aug. 2008, available online: http://www.wimedia.org/en/resources/whitepapers.asp?id=res.

[5] F. S. Lee, "Energy Efficient Ultra-Wideband Radio Transceiver Architectures and Receiver Circuits," PhD Dissertation, Massaschusetts Institute of Technology, Department of Electrical Engineering and Computer Science, Jun. 2007.

[6] M.-G. D. Benedetoo, T. Kaiser, A. F. Molisch, I. Oppermann, C. Politano, and D. Porcino, *UWB Communication Systems: A Comprehensive Overview.* New York, NY: Hindawi Publishing Corporation, 2006.

[7] "IEEE Standard for PART 15.4: Wireless MAC and PHY Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs): Amendment 1: Add Alternate PHY," 2007, avaiable online: http://standards.ieee.org/getieee802/download/802.15.4a-2007.pdf.

[8] D. D. Wentzloff, "Pulse-Based Ultra-Wideband Transmitters for Digital Communication," PhD Dissertation, Massaschusetts Institute of Technology, Department of Electrical Engineering and Computer Science, Jun. 2007.

[9] D. D. Wentzloff and A. P. Chandrakasan, "A 47pJ/pulse 3.1-to-5GHz All-Digital UWB Transmitter in 90nm CMOS," in *IEEE International Solid-State Circuits Conference*, Feb. 2007, pp. 118–119.

[10] F. S. Lee and A. P. Chandrakasan, "A 2.5nJ/b 0.65V 3-to-5GHz Subbanded UWB Receiver in 90nm CMOS," in *IEEE International Solid-State Circuits Conference*, Feb. 2007, pp. 116–117.

[11] D. D. Wentzloff, F. S. Lee, D. C. Daly, M. Bhardwaj, P. P. Mercier, and A. P. Chandrakasan, "Energy Efficient Pulsed-UWB CMOS Circuits and Systems," in *IEEE International Conference on Ultra-Wideband*, Sep. 2007.

[12] D. Neirynck, L. Huang, G. Dolmans, O. Rousseaux, and B. Gyselinckx, "Ultra-Low Power Impulse Ultra-Wideband Demonstration Platform," in *IEEE Internal Conference on Ultra-Wideband*, Sep. 2008, pp. 146–148.

[13] M. Verhelst, N. Van Helleputte, G. Gielen, and W. Dehaene, "A Reconfigurable, 0.13μm CMOS 110pJ/pulse, Fully Integrated IR-UWB Receiver for Communication and Sub-cm Ranging," in *IEEE International Solid-State Circuits Conference*, Feb. 2009, pp. 250–251.

[14] P. P. Mercier, M. Bhardwaj, D. C. Daly, and A. P. Chandrakasan, "A 0.55V 16Mb/s 1.6mW Non-Coherent IR-UWB Digital Baseband with $\pm$1ns Synchronization Accuracy," in *IEEE International Solid-State Circuits Conference*, Feb. 2009, pp. 252–253.

[15] D. C. Daly, P. P. Mercier, M. Bhardwaj, A. L. Stone, J. Voldam, R. B. Levine, J. G. Hildebrand, and A. P. Chandrakasan, "A Pulsed UWB Receiver SoC for Insect Motion Control," in *IEEE International Solid-State Circuits Conference*, Feb. 2009, pp. 200–201.

[16] *FrontPanel - User's Manual (v3.0)*, Opal Kelly Coporation, 2005.

[17] P. P. Mercier, D. C. Daly, and A. P. Chandrakasan, "A 19pJ/pulse UWB Transmitter with Dual Capacitively-Coupled Digital Power Amplifiers," in *IEEE Radio Frequency Integrated Circuits Symposium*, Jun. 2008, pp. 47–50.

[18] J. D. Kraus, *Electromagnetics*, 4th ed. New York: McGraw-Hill, 1992.