Reconfigurable OFDM Receiver for Next Generation Wireless Mesh

by

Aveek Dutta

B.Tech. Kalyani University - India, 2002

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirements for the degree of
Master of Science
Department of Electrical and Computer Engineering
2008
This thesis entitled:
Reconfigurable OFDM Receiver for Next Generation Wireless Mesh
written by Aveek Dutta
has been approved for the Department of Electrical and Computer Engineering

Dirk Grunwald

Douglas Sicker

Eugene Liu

Date ________________

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.
Performance of current mesh networks suffer due to the lack of an intelligent physical layer. Research and development in higher layer protocols alone cannot reduce the latency and performance of next generation wireless networks. If such a physical layer was designed so that it minimizes the involvement of higher layer protocols when it comes to routing, then latency in wireless networks can be reduced significantly. The transceiver for such a physical layer should have the ability to selectively switch the incoming packets “on-the-fly” to a different part of the spectrum so that a full-duplex transmit-receive chain is possible. This is also facilitated by the use of orthogonal modulation techniques either using orthogonal codes (OCDM) or orthogonal frequencies (OFDM) for transmission. An intelligent PHY like the one mentioned above will also play an integral part when it comes to cooperative transport of packets using MIMO systems and exploiting the spatial diversity of multiple antenna systems.

In this thesis a convincing solution to the above problem is presented which will open new avenues in the search for ultra low-latency mesh networks. A design and implementation of a 802.11a/g compatible physical layer transceiver has been proposed which also supports OFDMA and physical layer routing or Wormhole Routing by using decode-amplify-forward relaying. The thesis is comprised of three parts, (i) Design and implementation of 802.11a/g OFDM receiver in FPGA, (ii) Optimizing the receiver chain to include adaptation to frequency agile transmitters and (iii) Incorporating additional design features that enables the existing transceiver to work as intermediate relay with spectrum switching in a multi-hop wireless mesh network.
Dedication:

To Dola and My Parents
Acknowledgements

I would like to thank my adviser Prof. Dirk Grunwald and Prof. Douglas Sicker, for their support and guidance during my graduate studies at University of Colorado. I would also like to thank Prof. Youjian (Eugene) Liu for his insightful comments. Lastly, I thank all my colleagues, who directly or indirectly encouraged and motivated me.
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Related Work</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Theoretical Background</td>
<td>9</td>
</tr>
<tr>
<td>3.1</td>
<td>OFDM as a Multicarrier Communication Method</td>
<td>9</td>
</tr>
<tr>
<td>3.2</td>
<td>Multichannel Communication</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>OFDM Receiver Design</td>
<td>13</td>
</tr>
<tr>
<td>4.1</td>
<td>System Parameters</td>
<td>13</td>
</tr>
<tr>
<td>4.2</td>
<td>The Radio Components</td>
<td>13</td>
</tr>
<tr>
<td>4.3</td>
<td>Packet Detection and synchronization</td>
<td>14</td>
</tr>
<tr>
<td>4.4</td>
<td>Carrier Frequency Offset Correction</td>
<td>16</td>
</tr>
<tr>
<td>4.5</td>
<td>Long Correlation and Packet Timing</td>
<td>18</td>
</tr>
<tr>
<td>4.6</td>
<td>De-prefix and FFT</td>
<td>21</td>
</tr>
<tr>
<td>4.7</td>
<td>Equalization</td>
<td>22</td>
</tr>
<tr>
<td>4.8</td>
<td>Demodulation</td>
<td>28</td>
</tr>
<tr>
<td>4.9</td>
<td>De-interleaver, De-puncture, Viterbi Decoder and De-scrambler</td>
<td>30</td>
</tr>
<tr>
<td>4.9.1</td>
<td>De-interleaver</td>
<td>30</td>
</tr>
<tr>
<td>4.9.2</td>
<td>De-puncture</td>
<td>31</td>
</tr>
</tbody>
</table>
4.9.3 Viterbi Decoder ................................................. 32
4.9.4 De-scrambler .................................................. 33
4.10 Signal Symbol Decoder ........................................ 33

5 OFDM Receiver - Physical Layer Router 36
5.1 Frequency Agile Receiver ......................................... 36
5.1.1 Design and Implementation ................................... 38
5.2 Frequency Domain Switch/Router ............................... 39
5.2.1 Design and Implementation ................................. 41

6 Performance Evaluation and Analysis 44
6.1 Packet Detection and timing ..................................... 44
6.2 Equalizer and Channel Estimation ............................ 46
6.2.1 Equalizer Performance ....................................... 46
6.2.2 Channel Estimation ........................................... 46
6.3 Hardware Utilization ............................................ 48
6.4 Receiver Latency ................................................ 49

7 Conclusion 52

8 Future Work 54

Bibliography 55
# Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>OFDM Building Blocks</td>
<td>10</td>
</tr>
<tr>
<td>3.2</td>
<td>Multihop Relaying using OFDMA</td>
<td>11</td>
</tr>
<tr>
<td>3.3</td>
<td>Co-operative Relaying using OFDMA</td>
<td>12</td>
</tr>
<tr>
<td>4.1</td>
<td>Cascaded Integrator Comb Filter</td>
<td>15</td>
</tr>
<tr>
<td>4.2</td>
<td>Packet Detector using CIC filters</td>
<td>15</td>
</tr>
<tr>
<td>4.3</td>
<td>Packet Detector Output</td>
<td>16</td>
</tr>
<tr>
<td>4.4</td>
<td>Carrier Frequency Offset Block</td>
<td>17</td>
</tr>
<tr>
<td>4.5</td>
<td>Carrier Frequency Offset Estimation</td>
<td>18</td>
</tr>
<tr>
<td>4.6</td>
<td>Various Long Correlation Blocks</td>
<td>20</td>
</tr>
<tr>
<td>4.7</td>
<td>Long Correlator Output</td>
<td>21</td>
</tr>
<tr>
<td>4.8</td>
<td>The FFT Unit</td>
<td>22</td>
</tr>
<tr>
<td>4.9</td>
<td>Pilot Arrangement in OFDM</td>
<td>23</td>
</tr>
<tr>
<td>4.10</td>
<td>The Equalizer</td>
<td>26</td>
</tr>
<tr>
<td>4.11</td>
<td>The Linear Interpolator</td>
<td>26</td>
</tr>
<tr>
<td>4.12</td>
<td>Interpolator Output</td>
<td>27</td>
</tr>
<tr>
<td>4.13</td>
<td>Decision Boundaries</td>
<td>29</td>
</tr>
<tr>
<td>4.14</td>
<td>Maximum Likelihood Demodulator</td>
<td>29</td>
</tr>
<tr>
<td>4.15</td>
<td>Puncturing and De-puncturing of Convolution Codes</td>
<td>31</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>4.16 De-interleaver, De-puncture and Viterbi Decoder</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>4.17 De-scrambler block</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>4.18 802.11a/g Signal Symbol</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>4.19 Signal Symbol Decoder</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>5.1 Frequency Agile Receiver</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>5.2 Subchannel Interpolator</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>5.3 Frequency Domain Switch</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>5.4 Frequency Domain Switch using System Generator</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>5.5 Input and Output Spectrum for Switch</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>6.1 Packet detection in presence of channel noise</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>6.2 Packet timing in presence of channel noise</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>6.3 Equalizer Performance</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>6.4 Behavior of Pilot Subcarriers</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>6.5 Total hardware Utilization on a Virtex-IV</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>6.6 Receiver hardware Utilization</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>6.7 Receiver hardware Utilization Table</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Transceivers in radio communication are the first interface to the transmission medium and hence are largely responsible for the fidelity, functionality and versatility of the communication systems. As defined by the OSI layer the physical layer forms the lowest layer which interacts with the underlying communication medium, either wired or wireless. Today’s ever demanding data and voice communication requires low latency, reliable and high data rate networks. Traditional network architectures has been emphasizing on the improving the the MAC and its higher layer with limited additions to the physical layer architectures and design. With the advent of medium and high data systems like DVB, ADSL, WiFi, focus has slowly started to shift towards design of hardware that are intelligent and can outperform legacy systems with little or no help from the higher layers.

In the quest of a new evolved hardware, which is reconfigurable on the fly and robust at the same time, FPGA based hardware are proving to be the choice of ongoing research. FPGA are becoming faster and high gate count makes it suitable for RF and baseband signal processing. Specially design pipelined cores are available from different FPGA vendors (Xilinx, Altera, etc) giving the designers the liberty to design systems with tighter timing constraints and complex logics. Algorithms like CORDIC(Coordinate Rotation Digital Computer) [1] offer capabilities like division, arctangent and angle rotation that are often used in radio receiver subsystems like
equalizers and channel estimation. Furthermore readily available cores to synthesize DSP algorithm in the form of digital filters also offer great flexibility in designing DSP hardware with FPGA.

While much of the focus is drawn towards FPGAs when it comes to reconfigurable hardware, software equivalents are also proving to be a close competition. With the advent of faster processors and interfaces coupled with an abundance of user capabilities and programmability software defined radios are no more a futuristic idea. Platforms like GNU Radio [2] and VANU software radio [3] platforms offer the user to make subsystems more reconfigurable on the fly. Often a hybrid design has proved to be beneficial. This takes the benefits from both worlds and merges them to form a fast, low latency, reconfigurable and with complex signal processing elements. Faster interfaces with the hardware in the form of PCI and PCI Express coupled with the processing power of FPGAs makes it a formidable platform for developing next generation transceivers.

Complexity of the hardware increases with data rate, fidelity and operating frequency of the communication system. 802.11a/g has been adopted as the standard for wireless data communication and is a part of our day to day activity. Although the maximum data rate supported by WiFi is theoretically set to 54 Mbps it is hardly achieved in reality due to significant radio interference in the 2.4GHz ISM band. Furthermore since wireless networks are inherently multi-hop networks, routing delays which involves receive-store-process-queue-forward-contend-transmit as the processing chain at every node cripples the network to such extent that the effective throughput over a link is in few Mbps rather than tens of Mbps. While much work is being done on developing better routing and transport layer protocols it is not until recently that researchers have started to focus on developing hardware and related protocols to bring the latency down by making an intelligent PHY [4], [5] rather than an intelligent transport, network or MAC layer. Motivated by the need of hardware that will help us to build a low latency MANET as envisioned in [5], a hardware has been designed that not
only acts as a conventional 802.11a/g transceiver but also intelligently uses the spectrum by using frequency domain switching and adaptive modulation techniques. The transceiver, apart from relaying signals, also has the ability to route packets over multiple hops making packet-detect-equalize-demodulate-modulate-switch-frequency-transmit as the processing pipeline which eliminates delay oriented methods like store-process-queue-forward-contend. Although, reduced contention does not follow directly from the above design, use of preassigned assigned channels and scheduling mechanism and resource allocation similar to a TDMA network will certainly bring out the true essence of such a system. Clearly this requires an access control layer that has to be a part of the physical layer and should be able to make decision on path access with little or no intervention of the MAC layer. Also the proposed hardware should support multiple access techniques like OFDMA to further reduce latency. Use of orthogonal multi-carrier modulation techniques like OFDM along with a frequency agile transceiver enables every node in the network to make full use of the spectrum. Multi-carrier communication also introduces the idea of sub-channels where a pre-assigned-defined number of sub-carriers are grouped to form a subchannel. Thus every transceiver can be thought of as a switch/relay that can receive in one subchannel and transmit it in another set of subchannel based on the channel conditions. This kind of frequency agility requires a frequency agile transmitter as well as a frequency agile receiver. A software defined OFDM transmitter has been designed by researchers [6] at University of Colorado at Boulder which has successfully shown the evolution and performance of a frequency agile transmitter. In this thesis the evolution of an OFDM receiver capable of receiving and decoding information on different subchannels, and also capable of switching the incoming signals to a different subchannel on the fly, will be discussed. Such a hardware will also play an integral part in MIMO systems and collaborative communication using spatial diversity because different subchannels can be used to transmit the same packet and then combined at the receiver by exploiting spatial diversity. This improves the
reliability of wireless networks.

The hardware design itself poses its own challenges from an algorithmic point of view and easy portability to reconfigurable hardware platforms. Managing complexity and delivering a standard performance is always a challenge for hardware implementation of different algorithms. Packet detection and equalization are the two most important functions of the proposed baseband hardware platform. The complexities of the analog front-end responsible for downconverting and gain control has not been discussed in this thesis. However it is to be recognized and acknowledged that the performance of the baseband receiver largely depends on the fidelity and performance of the front-end since OFDM is inherently susceptible to frequency offsets at the downconverter and sampling clock errors. Gain control also plays an important role since improper gain control might result in saturating the ADC of the FPGA platform. Certain trade-offs were also required during equalization. While theoretically a lot of algorithms have been shown to provide excellent results, few of them have been actually implemented in hardware due to involvement of complex mathematical operations such as matrix inversion and complex divisions which introduces long processing chains and stalls the receiver pipeline. Therefore, linear interpolation techniques with pilot aided channel estimation has been chosen as the algorithm for our implementation. The Viterbi decoder is also a critical element in the datapath which introduces long processing chains and buffering due to the trellis aided decoding algorithms. An efficient open source implementation of the decoder has been used in this work. As a complete design platform Xilinx System Generator has been used along with other Xilinx tools like ISE.

The rest of the thesis is organized as follows. Chapter 2 describes related work. In chapter 3 we discuss the theoretical aspects of OFDM and how these properties are exploited to form the different building blocks of the receiver. Chapter 4 deals with the detailed description of the designs and algorithms for the proposed OFDM transceiver/switch/relay. Chapter 5 shows the implementation of the relay and switch
modes of operation. In chapter 6 we present the results and benchmark the performance of the receiver. Chapter 7 concludes the work and finally chapter 8 proposes some future directions that utilizes the OFDM transceiver-relay.
Chapter 2

Related Work

This chapter deals with related works in FPGA implementation of OFDM receivers and its use in reducing latency in wireless networks.

We start by citing previous works in implementing OFDM transceivers in FPGAs. Most of the instances of hardware implementation employ a hybrid design using the processing power of either the host PC or the embedded power PC core in the FPGA. Basic OFDM modem implementation has been around for while. Dick and Harris [7] describe algorithms and designs for packet detection, timing and equalization that can be implemented with relatively low complexity in FPGAs. Speeth et al. [8], [9] is one of the earliest to formally lay out the design aspects and algorithms for an OFDM receiver.

Troya et al. [10] describes an ASIC implementation of an OFDM receiver. The WARP project at Rice University [11] is another FPGA based development platform and have implemented a complete OFDM transmit and receive chain in the FPGA. The WARP platform is also programmed using Simulink and System Generator. Fifield et al. [6] has already implemented a frequency agile OFDM transmitter that is capable of efficiently utilization the available spectrum by selectively transmitting in some of the subcarriers rather than all subcarriers as done in 802.11a/g [12]. Wouters et al. [13] proposed a novel OFDM wireless modem with adaptive loading and their design is also optimized into an ASIC which supports HYPERLAN and IEEE 802.11a.

Coleri et al. [14] deals with the different equalization techniques for OFDM using
pilots to extract the channel characteristics and use the channel estimates to equalize the incoming signals. We utilize the block-type pilot arrangements of 802.11a/g and use linear interpolation techniques to equalize the signal. The KUAR platform from Kansas University [15], [16] is also a complete implementation of OFDM physical layer. They also implement a NC-OFDM where certain subchannels can be selectively and dynamically disabled.

Subchannel allocation mechanism in multicarrier communication using OFDM has been studied for some time now and is a critical part in 802.16a based systems. Zhu et. al [17] describes an adaptive subchannel allocation methods in a OFDM communication system. Although channel allocation does not form a part of this work, it is worth mentioning because the receiver proposed here has the capability to adapt to any kind of dynamic channel allocation from a control unit, usually a part of the path access control abstraction. As envisioned by Ramanathan [5] a receiver capable of switching waveforms from one subchannel to other on the fly will substantially reduce the latency in existing wireless networks. Simultaneous transmit and receive can be done only if the receiving subchannels and the transmitting subchannel do not interfere. This is particularly difficult in existing wireless networks because the packet detection and synchronization still depends on the physical layer preamble which is transmitted using all the subcarriers available. One apparent solution to this problem is to use the preambles to periodically synchronize the transmitter and the receiver and use TDMA slots to communicate. Although this is a feasible solution for one hop communication, it introduce a lot of overheads and delay as the number of hops increases. So synchronization and packet detection still forms an integral part of the receiver which needs further work and certainly calls for changes in existing protocols. Therefore, in future physical layers will have to handle three primitives - receive, discard or relay/re-broadcast after switching the subchannels instead of two, either keep it or discard it because currently the routing decisions are not a part of the physical layer. Finally our implementation of
this OFDM receiver with capability of spectrum switching and real time duplex relaying is believed to be novel and we know of no other publication that utilizes hardware transceivers for reducing latency in wireless networks.
Chapter 3

Theoretical Background

This chapter presents the theoretical background required to implement an OFDM receiver and briefly describes the concepts involved in a orthogonal multicarrier communications. The orthogonality property is used to allow multiple users to use the same center frequency within each others interference zone. This also leads to better utilization of the spectrum by allowing more number of users to access the channel.

3.1 OFDM as a Multicarrier Communication Method

OFDM is a multicarrier communication method that relies on the orthogonality between the modulating frequencies. OFDM signal is generated by inverse Fourier transform and decoded by computing Fourier transform of the incoming signal. FFT algorithm is chosen as a very efficient way of implementing the multicarrier modulation/demodulation technique. The basic block diagram of an OFDM communication system is given in fig.3.1.

A multicarrier communication system provides for a higher data rate with an underlying lower data-rate baseband modulation. The overall data rate is directly dependent upon the size of the IFFT/FFT engine. The maximum data rate that can be achieved by using a 64 QAM with 2/3 rate convolution coding and a 64pt FFT is 54Mbps.
3.2 Multichannel Communication

Since the subcarriers in OFDM are orthogonal, they do not interfere amongst themselves. Therefore each of the subcarriers or a set of subcarriers can be used as separate data-carriers. This is where, multicarrier modulation techniques has its advantage over single carrier systems. Each of the subchannel is transmitted after upconversion using the same carrier frequency and thus more number of users can communicate in a wireless mesh network. Carrier frequency reuse within the interference region of nodes is certainly an advantage over current Wi-Fi enabled mesh network technology. But this is dependent of the availability of number of subchannels. Although there is an upper limit to throughput it can still be a lot higher than existing modulation techniques. Splitting the available spectrum into different subchannel is called Orthogonal Frequency Division Multiple Access or OFDMA.

In order to be able to utilize the benefits of OFDMA we need a transceiver which is able to transmit and receive as well in any subchannel as directed. The real challenge in an OFDMA network is not the downlink but in the uplink, when multiple nodes transmits data to a single destination. Misalignment of the FFT window will lead to erroneous decoding. Therefore it requires tightly synchronized clock in all nodes, which
is hard to achieve. But with chipscale atomic clocks in the making [18] the prospects of a tightly time synchronous network is not very far.

Fig.3.2 shows how multichannel communication can be used to relay packets from across the network. In this architecture the key concept is the use of the same carrier frequency $f_c$ for simultaneous reception and transmission. The available spectrum is divided into into 4 orthogonal subchannels. Node A, which is the source uses subchannel #0 and subchannel #2 to transmit different data to Node B and C on carrier frequency $f_c$. With prior knowledge of the subchannels Node B and C decode the information bits only on subchannels #0 and #2 respectively. Utilizing the orthogonality between the subchannels Node B and Node C switch the incoming information bits from subchannel #0 and #2 to subchannels #1 and #3 respectively towards the destination Node D and Node E. This can be transmitted using the same carrier frequency $f_c$ without interfering with the incoming reception since the subchannels are orthogonal to each other. This is one of the biggest advantages of OFDM.

![Figure 3.2: Multihop Relaying using OFDMA](image)

Another application of OFDMA and orthogonal subchannels is in co-operative
relaying. Fig. 3.3 shows the source, Node A can send the same information with space-time coding on different subchannels, which after relaying using frequency domain switching can be diversity combined at node D which is the final destination.

In both the above application, it is to be noted that channels in every hop are different and there is a need to equalize the signal after every hop. Therefore, if pilot aided equalization is used then every subchannel requires a minimum number of pilots so that the receiver is able to equalize the signal before relaying. Implementation of frequency domain switching and relaying has been described in sections 5.1 and 5.2.

Figure 3.3: Co-operative Relaying using OFDMA
Chapter 4

OFDM Receiver Design

This chapter presents the actual design and implementation of an OFDM receiver in FPGA. It also describes the various algorithms that has been implemented in the hardware to successfully decode 802.11a/g packets.

4.1 System Parameters

For this implementation we chose the 802.11a/g [12] physical layer specification with some changes to the specification. While keeping the subcarrier spacing constant, the sampling frequency has been increased 4 times to 80 MHz and the FFT/IFFT size to 256 instead of 20MHz and 64pt FFT/IFFT respectively. This allows us to accommodate more number of subcarriers with the same bandwidth, providing greater throughput but at the expense of a higher clock frequency for the baseband receiver. For this implementation we have chosen to use only 64 of the subcarriers and the receiver has been designed around this assumption. Since the design operates at a higher clock frequency, the number of samples processed per unit time is four times as compared to conventional 802.11a/g receivers.

4.2 The Radio Components

The OFDM transceiver components consist of the following:

(1) Radio Frontend from Fidelity Comtech, Inc - This radio is responsible for
up/down conversion to/from the 2.4GHz ISM band. Gain control is also a part of this unit which can be controlled by software on the host computer.

(2) Xilinx ExtremeDSP development kit - IV manufactured by Nallatech

The ExtremeDSP board includes a Virtex IV equipped with a PCI/USB interface, two sets of A/D and D/A converters.

4.3 Packet Detection and synchronization

The packet detection of an OFDM utilizes the periodic nature of the short preamble. Every 802.11a/g packet has a short preamble of 10 symbols each 64 samples long. Schmidl and Cox [19] explains a packet detection algorithm which has been used in many implementation. The short preamble symbols are periodic with a period of 64 samples. The upper branch calculates the autocorrelation energy between r(n) and r(n+64) which is given by,

\[ C(d) = \sum_{i=0}^{D-1} (r^*_d + i \cdot r_{d+i+D}) \]  \hspace{1cm} (4.1)

where \( D = 64 \). The lower branch computes the signal energy during that autocorrelation window, given by

\[ R(d) = \sum_{i=0}^{D-1} |r_{d+i+D}|^2 \]  \hspace{1cm} (4.2)

The decision metric is obtained as

\[ M(d) = \frac{C(d)}{R(d)} \]  \hspace{1cm} (4.3)

Both of these correlation can be implemented using a recursive summing unit or a single stage CIC (Cascaded Integrator Comb) Filter expressed by the iterative eq. (4.4) and the block diagram in figure 4.1.

\[ y(n) = y(n) + x(n) - x(n-D) \]  \hspace{1cm} (4.4)

In hardware the CIC is realized using a subtractor and an accumulator after the com-
puting the correlation energy. Therefore three CIC units are shown in figure 4.2 along with the two branches of the packet detector as described above. The division in computing the decision metric as given in eq. (4.3) is avoided by scaling \( R(d) \) and comparing with \( C(d) \). Fig. 4.3 shows the output of the packet detector. The plateau shape of the output results from the periodic nature of the short preamble. The autocorrelation energy starts to build up and remains constant after 2 symbol periods. The second trace showing the raw signal energy during the autocorrelation phase also builds up and goes constant from the second symbol as all the short preamble symbols have the same energy. In the presence of noise, the plateaus will not be flat but will have inclination with jagged edges. The packet detect module has been shown to detect packets in the presence of noise. The scaling factor for the signal energy needs to be dynamically adjusted to adapt to the changing signal energy or else this might lead to false detection or no detection at all.
4.4 Carrier Frequency Offset Correction

Since the transmitter and the receiver clocks are powered by different crystal oscillator there will always be an offset between the two. At the receiver during down-conversion the oscillator at the receiver will try to tune to the desired center frequency but due to oscillator drifts, let there be an offset of $\delta f$. The transmitted signal can be expressed as

$$y_n = s_n e^{-j2\pi f_{tx} n T_s}$$  \hspace{1cm} (4.5)

after downconversion we get

$$r_n = s_n e^{-j2\pi \delta f n T_s} \quad \text{where} \quad \delta f = f_{rx} \sim f_{tx}$$  \hspace{1cm} (4.6)

Since the preambles are periodic with periodicity $D = 64$ sample for short preambles and $D = 256$, we can use this property to extract the frequency offset $\delta f$ as follows

$$Z = \sum_{n=1}^{L-1} r_n \cdot r_{n+D}^* = e^{-j2\pi \delta f D T_s} \sum_{n=1}^{L-1} |s_n|^2$$  \hspace{1cm} (4.7)
L and D are the integration length periodicity which are different for short and long preambles. The longer the integration time the better is the frequency estimate. Therefore for short preamble $L = 128$ and $D = 64$, where as for long preamble $L = 256$ and $D = 256$. The frequency offset can directly computed using the autocorrelation energy in eq. (4.7) as follows,

$$
\delta f = \frac{1}{2\pi DT_s} \arctan(Z)
$$

Translating the algorithm in the hardware involves the following design block/steps:

- Detect the short preambles to trigger the change of integration time (from $L = 128$ to $L = 256$)
- Calculate the frequency offset by calculating the correlation energy and performing the arctan operation.
- A phase accumulator to latch on to the correct phase after at the end of the coarse and fine estimates of the phase.
- Finally a frequency synthesizer to generate the desired frequency equivalent to the calculated phase offset.

![Figure 4.4: Carrier Frequency Offset Block](image-url)
Fig. 4.4 shows the hardware logic design. Arctan is implemented using the Xilinx “Cordic Arctan” IP. The short preamble detection is performed by correlating the input signal with a local copy of the short preamble stored at the receiver. The detailed implementation is explained in the next subsection called “Long Correlation”. Fig. 4.5

![Image](image_url)

**Figure 4.5: Carrier Frequency Offset Estimation**

shows a typical output of the phase accumulator. The correlation energy is latched during the short and long preambles. The coarse estimate is updated twice during the short preamble and then the estimate is made finer twice at the end of the long the two preambles. The phase estimate is finally latched and fed to the Direct Digital Synthesizer (DDS) to produce the offset frequency which is then multiplied with the incoming signal using an I/Q complex multiplier.

### 4.5 Long Correlation and Packet Timing

Once a valid packet is detected by the packet detect block and carrier frequency offset has been corrected it is the job of the long correlator block to extract the correct timing of start of the packet which is defined by the start of the signal symbol. This
trigger is extremely important for the performance of the receiver blocks because this triggers the FFT block to start performing the FFT operation to convert the input signal into the frequency domain. The long preamble consists of two symbols each 256 samples long preceded by a 128 sample long cyclic prefix. Correlation is performed with a local copy of the long preamble. Correlation involves 256 sample long multiply-add and one shift operation every clock pulse. Therefore, multiplication is to be done for all 256 samples which is the correlation window and then all the results are added to form the correlation energy for that clock. Multiplication are all floating point operation and is to be done both for I and Q channels. Also to perform 256 additions the correlator needs to run at 256 times the input sample rate. This is certainly not implementable due to a lack of FPGA resources. To eliminate unnecessary hardware and degenerate the correlator to simple logical operations we used the following approach:

1. Use the “sign” bit of the I and Q samples instead of the actual values to eliminate any floating point operation.

2. The local copy of the long preamble also consists of the sign of the time domain signal.

3. Since the objective of correlation is to search for an exact set of samples as stored in the receiver, we use XNOR as the, the truth table shown in fig.4.6(a). Whenever the sign of the input sample matches that of the local copy the output is a '1' else '0'.

Fig. 4.6 shows various processing elements involved in the correlator. The slicer and concat blocks in fig. 4.6(a) iteratively accumulates the sign of every incoming samples which is passed on to the XNOR and addition tree. The accumulated sign form a 256bit number which is further sliced using the slice blocks as shown in fig. 4.6(c). The slicers form a binary tree which divides the input bits (256) into 256 individual bit operation of xnor with the corresponding sign of sample which is stored as a local copy.
The output of the XNOR is again added using a binary conquering tree to form the correlation energy for that correlation window. The basic processing element for the XNOR and add logic is shown in fig. 4.6(d). The total correlation can be maximum of $N^2$ where $N$ is the number of positive samples (sign bit = 1) in the time domain version of the long preamble.

The output of the long correlator is shown in fig. 4.7. The two peaks mark the end of the two long preamble symbols. The second peak is detected and marks the end of the packet preamble and start of the “signal” symbol. This signal is used to trigger the FFT block as shown in section 4.6.
4.6 De-prefix and FFT

Every OFDM symbol carries a 64 sample long cyclic prefix. This cyclic prefix carries no information and is used to compensate for any multipath effect. Therefore, these 64 samples needs to be stripped of for every OFDM symbol. The FFT trigger input from the long correlator is used to load data into the FFT engine. The FFT engine used here is the IP block from Xilinx that is a part of the Xilinx System Generator development platform. The FFT engine along with the cyclic prefix removal is shown in fig. 4.8. The counter keeps track of the sample count to be skipped, which is 64 in this case and then triggers the FFT to ensure that the FFT window has the correct samples at the input. The effect of deviations in this timing has been discussed in [7], [9], [8], [10]. Therefore, this timing is of paramount importance when it comes to successfully decoding an OFDM symbol. The FFT engine is operated in “pipelined streaming mode” [20] and has been optimized for speed. However the latency in the FFT engine is pretty significant compared to the processing delay of the entire receiver.
pipeline. Detailed analysis of timing issues and latency has been done in the chapter 6. The output of the FFT produces the frequency domain representation of the OFDM symbol. The subcarriers are indexed from -128 to 127 for a 256 point FFT, where the order in which the subcarriers are latched at the output of the FFT engine is 

\[0, 1, 2, 3, \ldots, 127, -128, -127, \ldots, -3, -2, -1]\.

This is important for the receiver sub-systems that follow the FFT as discussed in section 4.7.

4.7 Equalization

The equalizer is an integral part of the receiver. Many equalization algorithms have been discussed in [7], [9], [8], [10], [14], [13], and [21]. The equalizer is responsible to correct any phase and magnitude errors in the signal which is introduced by the time-varying, frequency selective wireless channel. A wireless channel is often modeled as a sum of a number of delay paths. If \(h(t)\) be the time-domain representation of the channel for M different delay paths can be written as,

\[g(t) = \sum_{m=1}^{M} \alpha_m \delta(t - \tau T_s)\]  \hspace{1cm} (4.9)

where \(\alpha_m\) is a complex valued coefficient.

At the receiver a N point DFT is used to transform the data back to frequency domain. Let \(X = [X_k]^T\) and \(Y = [Y_k]^T\) for \((k = 0, \ldots, N-1)\) denote the input data of the IDFT at the transmitter and the output data of the DFT at the receiver respectively. Let
$\bar{g} = [g_k]^T$ and $\bar{n} = [n_n]^T$ be the channel impulse response and AWGN respectively. Define the input matrix as $\bar{Y} = \text{diag}(\bar{X})$ and the DFT matrix as

$$
\tilde{F} = \begin{bmatrix}
W_N^{00} & \ldots & W_N^{0(N-1)} \\
\vdots & \ddots & \vdots \\
W_N^{(N-1)0} & \ldots & W_N^{(N-1)(N-1)}
\end{bmatrix}
$$

where $W_N^{ik} = \left(\frac{1}{\sqrt{N}}\right)^{-j2\pi \left(\frac{ik}{N}\right)}$. Also define $\tilde{H} = DFT_N(\bar{g}) = \tilde{F}\bar{g}$ and $\tilde{N} = DFT_N(\bar{n}) = \tilde{F}\bar{n}$. Under the assumption that noise and channel are uncorrelated and in absence of ISI we can write the following,

$$
\bar{Y} = DFT_N(IDFT_N(\bar{X}) \ast \bar{g} + \bar{n}) = \bar{X}\tilde{H} + \tilde{N}
$$

This equation demonstrates that OFDM systems can be viewed as a 2D lattice in time and frequency plane. Pilot subcarriers are inserted according to a pre-determined order which is used to estimate the channel. The channel estimate for the intermediate subcarriers can be obtained by interpolation between the pilots. Pilot arrangement are of two type, [a] Block Type and [b] Comb Type as shown in fig. 4.9.

Figure 4.9: Pilot Arrangement in OFDM

Pilots are BPSK modulated symbols placed at regular intervals in time-frequency plane. In block type transmission all the N subcarriers are sent as pilots and this pilot OFDM symbols are sent at periodic intervals in time. Whereas in comb type arrangement each OFDM symbol has pilot symbols at periodic frequency bins. The block type arrangement is useful in slow fading channel whereas the comb type is useful in capturing
channel characteristics in a fast fading channel. In 802.11a/g comb type pilot arrangements are used and therefore in this work we have implemented the equalization of OFDM symbols involving comb type pilots. Equalization using pilots is largely dependent on the number of pilots in an OFDM symbol. For 802.11a/g four pilot tones are inserted in subcarriers \([-21 -7 7 21]\) and are used to estimates the channel. Let there be \(N_p\) number of pilot in one OFDM symbol and are uniformly inserted in with \(S\) subcarriers apart, where \(S = N/N_p = 14\). The receiver knows the pilot locations \(\tilde{P} = [P_k]^T\), \((k = 0,N_P - 1)\), the pilot values \(\tilde{X}^P = [P_k]^T\) \((k = 0,N_P - 1)\) and the received signal \(\tilde{Y}\). Given these information a Least Squares estimate can be made at the pilot location given by,

\[
\hat{H}_{LS}^p = \begin{bmatrix} Y(P_0) \\ X_0^p \\ \vdots \\ Y(P_{N_p-1}) \\ X_{N_p-1}^p \end{bmatrix}^T
\]

The task here is to estimate the channel condition at the data subcarriers given the LS estimates at the pilot subcarriers, \(\hat{H}_{LS}^p\) and the received signal \(\tilde{Y}\). 1D interpolation techniques can be used to estimate the channel at the data subcarriers. 1D interpolation are of following types,

[a] Linear Interpolation: This is a piece-wise constant interpolation where the channel estimate between two pilots is given by :

\[
\hat{H}(kS + t) = \hat{H}_{LS}^p(k) + \left[\hat{H}_{LS}^p(k + 1) - \hat{H}_{LS}^p(k)\right] \cdot \frac{t}{S}, \quad 0 \leq t \leq S
\]

Linear interpolation is easy to implement in hardware as it requires 1 MAC unit to perform the task.

[b] Second order interpolation: This technique uses three pilots instead of two as in the linear interpolation case. It is a second order polynomial curve fitting technique used to approximate the channel estimates at the data subcarriers given the pilots.
Linear combinations of three pilot estimates gives better interpolation results.

\[ H_e(k) = H_e(mL + l) \quad 0 \leq l \leq L \]  
\[ = c_1 H_p(m - 1) + c_0 H_p(m) + c_{-1} H_p(m + 1) \]  
\[ \text{where,} \begin{cases} 
  c_1 = \frac{\alpha(\alpha - 1)}{2} \\
  c_0 = -(\alpha - 1)(\alpha + 1) \\
  c_{-1} = \frac{\alpha(\alpha + 1)}{2} 
\end{cases}, \quad \alpha = \frac{l}{N} \]  

It is to be noted that since 52 subcarriers are used instead of the total available 64, this technique cannot be estimated for subcarriers \([0:-7]\) and \([21:26]\) because during these intervals we do not have three pilots to do a complete non-linear estimation. Resorting to linear interpolation can be a solution in this case. Also because of this discontinuity of data carriers the channel estimate of two adjacent OFDM symbol are independent and as a result we cannot use the last pilot of the previous symbol to do the interpolation of subcarriers \([0:-7]\). Using 256 subcarrier will allow us to interpolate between two consecutive OFDM symbol and also allow us to use non-linear interpolations as well.

[c] Spline Interpolation: Spline interpolation uses higher order polynomial to obtain the best fit given the pilot carriers. This technique finds the solution of a system of linear equation of the form \( Y = AX \) where the matrices have orders in multiples of \( N \) (DFT size). As a result this algorithm is hard to implement in hardware due to immense complexity involved in computing the inverse of a matrix. Therefore, software simulation is used in form of the spline function in MATLAB to analyze the effect of a spline interpolation.

For hardware implementation, a linear interpolation equalizer have been designed and incorporated in FPGA as shown in fig. 4.10 The interpolation block is shown in fig.4.11. This block is responsible for computing the channel estimates of the data subcarriers using linear interpolation. The constant multiplier multiplies the difference between two pilots by the \( 1/S = 1/14 \) and the accumulator accumulates the results
Figure 4.10: The Equalizer

for every $t \in [1 : S]$ and adds to the previous pilot. The output provides the channel estimates for all the data subcarriers. Once the channel estimates of the subcarriers are obtained at the output of the interpolator the received subcarriers $\tilde{Y}(k)$ are corrected as follows,

$$\text{angle} \left[ \hat{X}(k) \right] = \text{angle} \left[ Y(k) \right] - \text{angle} \left[ H_e(k) \right]$$  \hspace{1cm} (4.12)$$

$$\text{mag} \left[ \hat{X}(k) \right] = \frac{\text{mag} \left[ Y(k) \right]}{\text{angle} \left[ H_e(k) \right]}$$  \hspace{1cm} (4.13)$$

Figure 4.11: The Linear Interpolator
Where $H_e(k)$ are the channel estimates obtained from the linear interpolator block. The I and Q signals are computed from the magnitude and angle as follows,

$$I_{corrected} = mag\left[\hat{X}(k)\right] \cdot \cos(\text{angle}[\hat{X}(k)])$$

(4.14)

$$Q_{corrected} = mag\left[\hat{X}(k)\right] \cdot \sin(\text{angle}[\hat{X}(k)])$$

(4.15)

Figure 4.12: Interpolator Output

The linear interpolation plots in 4.12 show how the pilots are used to estimate the channel transfer function. Separate interpolation has been done for phase and magnitude. Fig. 4.12 shows the magnitude and phase interpolation of a BPSK modulated OFDM symbol. The red colored samples shows the pilots which has been changed by the time varying frequency selective channel. With the prior knowledge of the pilots we know that the magnitude of the pilots as transmitted is '1' for all pilots and phase of the pilots are $[0, 0, 0, \pi]$. The objective of the equalizer at the receiver is to restore the magnitude and phase of the pilots and assume that the channel in between the pilots is linear, so that the subcarriers can be equalized by linear interpolation. Fig.
4.12 also show that the magnitude and phase of the pilots are restored to their original transmitted values. The uneven magnitude is largely due to the non-linearities involved in the CORDIC arctan IP and also due to limited precision in a fixed point design. The performance of the equalizer is shown in chapter 6 where we present the constellation plots for actual 802.11g Wi-Fi packets. It is noticed that the equalizer degrades for tighter constellation suggesting that we might need to use non-linear interpolation techniques and also use inter symbol interpolation. But although there are some outliers for QAM constellations we are still able to decode the information bits correctly due to the underlying FEC, interleaving and scrambling.

4.8 Demodulation

The demodulator follows the equalizer. The equalizer de-rotates the constellation and restores the constellation to its original configuration. the demodulator is a maximum likelihood baseband demodulator which performs threshold test as per the symbol energy specified in 802.11a/g specification [12]. Fig. 4.13(a) shows the constellation map for 16QAM modulation. The symbol energy has been normalized using a normalization factor as given in Fig. 4.13(b). Decision boundaries are given by the perpendicular bisector of the line joining the two symbols. This gives the optimum decoder as it minimizes the Euclidean distance between received signal and the nearest constellation point.

Also it is to be noted that the right half of the constellation is the mirror image of the constellation to the left, except for the sign bit $b_0$. Therefore instead of performing the threshold for 16 constellation points we can demodulate using two threshold separately on the I and Q channel as shown in fig. 4.14. The output of the demodulator is always a 6bit which is the maximum no of bits per subcarrier for a 64QAM modulation. The proper number information bits are extracted in the de-interleaver as discussed in 4.9, thus ensuring that only the information bits are extracted at the de-interleaver.
A key point to this demodulator is a feedback loop from the receiver block after the demodulator viz. de-interleaver and the signal symbol decoder. The signal symbol con-
tains the information about the modulation and coding rate of the entire payload that follows the signal symbol. The parameter called $N_{bpsc}$ has the information of the number of bits per subcarrier, which also specifies the baseband modulation type. $N_{bpsc}$ is used as a control signal to a set of multiplexers that selects the appropriate decision boundaries and ensures correct demodulation of the information bits. Similar techniques have been employed for higher constellations like 64QAM as well, only with the addition that the number of threshold test have been doubled when compared to that of a 16QAM. As of the current design the demodulator is capable of decoding BPSK, QPSK, 16QAM and 64QAM signals.

4.9 De-interleaver, De-puncture, Viterbi Decoder and De-scrambler

4.9.1 De-interleaver

The interleaver is a block interleaver with a block size corresponding to the number of information bits in a single OFDM symbol denoted by $N_{CBPS}$. The interleaver is defined by a two-step permutation. The first permutation ensures that adjacent coded bits are mapped onto nonadjacent subcarriers. The second ensures that adjacent coded bits are mapped alternately onto less and more significant bits of the constellation and, thereby, long runs of low reliability (LSB) bits are avoided. Like the transmitter the de-interleaving at the receiver is also governed by two permutation. At the receiver let $j$ denote the index of the received signal and $i$ denotes the index after the first interleaving and $k$ denotes the index after the second permutation then the permutation equations are defined as

The first permutation,

\[
i = s \times \text{floor} \left[ \frac{j}{s} \right] + \left[ j + \text{floor} \left( 16 \times \frac{j}{N_{CBPS}} \right) \right] \mod(s) \quad j = 0,1,\ldots,N_{BPSC} - 1
\]

(4.16)

where \( s = \max \left[ \frac{N_{BPSC}}{2}, 1 \right] \)

(4.17)
This permutation is just the inverse of the second permutation at the transmitter. The second permutation is defined as

\[ k = 16 \times i - \left[ N_{CBPS} - 1 \right] \times \left\lfloor \frac{i}{N_{CBPS}} \right\rfloor \times \frac{i}{N_{CBPS}} i = 0, 2, \ldots, N_{CBPS} - 1 \]

This permutation is the inverse of second permutation at the transmitter.

The implementation is done using M-Code which is a combination of slicers that rearranges the ordering of the bits based on eq.(4.16) and eq.(4.9.1) and then concatenated after rearranging. The indexing map can be predetermined and the the can be hardcoded into the receiver and only needs to changed when the interleaver at the transmitter changes.

### 4.9.2 De-puncture

De-puncturing is the process to insert the bits that were stolen at the transmitter to increase the coding rate. There are two types of puncturing in 802.11a/g. The specification manual [12], page 18 shows the detailed process of puncturing and de-puncturing the data bits, also shown in fig.4.15. Fig.4.15(a) shows the puncturing and de-puncturing of the bits for 3/4 rate convolution codes and fig.4.15(b) shows the puncturing and de-puncturing for rate 2/3 convolution codes.

![Figure A](image1)

![Figure B](image2)

Figure 4.15: Puncturing and De-puncturing of Convolution Codes
The de-puncture block is also implemented using M-Code which inserts the dummy bits at appropriate indices.

### 4.9.3 Viterbi Decoder

The Viterbi Decoder uses trellis based decoding. The convolution encoder at the transmitter uses \([133, 171]\) as the polynomial to produce a \(1/2\) rate convolutional code with constraint length 7. The receiver employs a trellis based maximum likelihood Viterbi decoder which decodes the input bits to obtain the basic information bits. The trellis length is chosen to be 5 times the constraint length.

For implementation, we have used an open source verilog implementation of the trellis based viterbi decoder [22]. The block has been optimized for memory utilization and speed. However, additional hardware is required for this block to work in terms of additional buffering and control signals. Fig. 4.16 shows the complete implementation of the de-interleaver, de-puncture and Viterbi decoder using system generator blocks along with the control signals.

![Figure 4.16: De-interleaver, De-puncture and Viterbi Decoder](image-url)
4.9.4 De-scrambler

The de-scrambler is just the inverse of the scrambling operation as mentioned in the specification [12]. The same scrambler unit is used to scramble and de-scramble the data. Scramblers are essentially a chain of shift registers with appropriate taps XOR-ed together to give a pseudo-random code of a fixed length. The scrambler fig.4.17 shows the scrambler unit.

![Figure 4.17: De-scrambler block](image)

It is to be noted that the “signal” symbol is not scrambled whereas all subsequent symbols have to go through the de-scrambler. Thus a control signal is required called “signal-done” to multiplex between the unscrambled “signal” symbol bits and the de-scrambled bits of the subsequent symbols.

4.10 Signal Symbol Decoder

The signal symbol is the first symbol in the physical layer payload. The signal symbol contains all the information required to decode the information bits. The signal symbol is composed of 24 bits and contains the following fields as specified in [12]:

- Rate - bits 0 - 3
- Reserved bit - bit 4
• Length - bits 5 - 16 with bit 16 being the MSB. The length field contains the number of octets of information in the packet.

• Parity bit - bit 17. The parity bit contains even or odd parity for bits 0 to 16.

• Signal Tail - bits 18 - 23. The tail bits are used to initialize the de-scrambler at the receiver.

The “signal” symbol is shown in fig. 4.18

![Figure 4.18: 802.11a/g Signal Symbol](image)

The “signal” symbol decoder is implemented by a 24 stage shift register. The outputs corresponding to the individual fields mentioned above are latched using a suitable control signal. Fig.4.19 shows the implementation. The decoder is a set of 24 register units connected in a chain which is controlled by a clock that counts from 0 to 23. At the end of the 23rd count the outputs are latched as shown by the output ports.
Figure 4.19: Signal Symbol Decoder
5.1 Frequency Agile Receiver

In this section, the OFDM receiver has been customized to be able to decode the information bits from a certain set of subcarriers, called subchannels. For this implementation the following points are to be taken into consideration:

- The signal symbol is the key to decode on subchannels, as it contains the information about modulation type and coding rate of the packet and its length. Since the signal symbol is still transmitted on all subcarriers it should be decoded as any other standard OFDM symbol and then use that information to change the receiver pipeline to tune to a particular subchannel.

- The equalizer needs to be changed to be able to equalize on one subchannel using a less number of pilots. Using linear interpolation using fewer number of pilots will lead to a lower precision in the equalized signal.

- The transmitter also needs to be frequency agile and should be able to transmit data from different users into different subchannels. To do this 52 subcarriers are split into two subchannels, subchannel #1 has subcarriers −26 to −1 and subchannel #2 has subcarriers 1 to 26. The information bits from two users are processed separately for encoding, interleaving and puncturing and then placed on two subchannels.
• For simplicity the encoding and length of the data from two users are kept same. This structure lets us use the same signal symbol configuration without adding additional fields/symbols for two separate users. The signal symbol can be modified to accommodate the encoding information of two separate users.

• At the receiver, since we are decoding on any one of the two subchannels and because the de-interleaver expects data bits on all subcarrier, as that’s how it was encoded at the transmitter, two successive OFDM symbols with data bits on either subchannel #1 or subchannel #2 are to be merged to form 52 subcarriers and then feed it to the de-interleaver. Therefore the de-interleaver and the Viterbi decoder remain unchanged for a full channel decoding or a subchannel decoding. This helps in reducing the requirement of control signal for all the receiver blocks.

• But this design calls for additional buffering and a multirate design as shown in fig.5.1. The data going out of the data buffer is half that of the data rate coming in, because two OFDM symbols, with data on half the subcarriers will have to be merged to form on full spectrum OFDM symbol.

• Another way of doing this would be to change the transmitter to interleave and encode on only the set of subcarriers allotted to a particular user rather than performing the encoding on all subcarriers. This will require additional control signal at the transmitter as well as the receiver which involves additional hardware.

• Another important aspect of the implementation process is to compensate for the order in which the subcarriers are interpreted by the FFT block. At the transmitter the encoding and interleaving is done by arranging the subcarriers as \([-128, -127, \ldots, -1, 0, 1, 2, \ldots, 126, 127]\), with pilots in subcarriers
While the FFT block requires its input and output ordering to be $[0, 1, 2, \ldots, 127, -128, -127, \ldots, -2, -1]$. Thus it can visualized that the subchannel receiver not only requires to chop of a section of the spectrum but also rearrange them in a proper order so that the de-interleaver and viterbi decoder is transparent to the fact that its a subchannel decoder. This introduces additional latency in the system on top of the additional hardware requirement for buffering. But this eliminates the change of the entire transmit and receive pipeline.

- Control signal to tune receiver to a particular subchannel can be done in two ways. One way is to embed that information in the signal symbol itself which is to be decoded first anyway to extract information required to decode the packet. Otherwise it can be controlled by the MAC layer in the form of a software control, if we can think of some kind of a out of band signaling involved before the actual transmission begins which sets up the receiver pipeline accordingly.

### 5.1.1 Design and Implementation

Fig. 5.1: Frequency Agile Receiver

Fig. 5.2 shows the linear interpolation equalizer output for subchannel #2. Same subchannel interpolation is applied to the phase of the incoming signal as well. For
subchannel #1 pilots $[-21, -7]$ are used and for subchannel #2 pilots $[7, 21]$ are used.

The control signal “sc_sel” shown in fig. 5.1 control which subcarrier to equalize and decode.

![Subchannel Interpolator](image)

Figure 5.2: Subchannel Interpolator

### 5.2 Frequency Domain Switch/Router

In this section we present an addition in the existing OFDM receiver so that it can work as a frequency domain switch and switch between incoming and outgoing subchannel on-the-fly. This makes the receiver to work as a full-duplex transceiver rather than a conventional half-duplex transceiver. Physical layer routing of packets has been discussed at length in [23]. As claimed by [23] we can use the physical layer to route packets. But in order to do this we need a hardware that supports the subchannel switching with control signals either embedded in the received packet or using using software control from the host controller. Fig. 5.3 shows the schematic for a frequency domain switch. This type of full duplex transceiver requires the use of two sets of radio front-end. With advances in fabrication technology cost of front-ends will certainly not be a hindrance to this architecture. Using the receiver and the transmitter as a pipeline without any feedback loop and negligible turnaround time between the receive and
transmit mode, latency in multihop mesh network will reduce greatly as also mentioned in [5]. With this configuration of the receiver, path access mechanism can be designed in higher layer that will be responsible in setting up the path and allocating subchannels and then the intermediate nodes can act as a pipeline to switch incoming packets on-the-fly onto another subchannel without mutual interference while using a full duplex mode of communication.

![Figure 5.3: Frequency Domain Switch](image)

Designing such a frequency switch will has its own issues and challenges which are explained as follows:

- The preamble is still used for packet detection and CFO estimation. The preamble is transmitted using the full spectrum. Similarly the signal symbol also uses all the full spectrum. This imposes a restriction in using the receiver as a full duplex relay because the relayed signal going out of the transmitter block cannot use the full spectrum to transmit the preamble and the signal symbol required to detect and decode the relayed packet in the next hop, simply because being a full duplex relay, the receiver pipeline is still receiving data on at least one of the subchannels during the transmission. Therefore, the transmitting and the receiving signal would interfere with each other.

- A possible and convincing solution to this situation is to change the preamble
and the signal symbol to transmit only in the subchannel that is transmitting and not all the subcarriers. To do this either the sequence of the long and short preamble symbols needs to be changed or it has to be spread onto multiple symbols on the transmitting subchannels only.

- Another plausible solution is to use some sort of a label so that receiver is able to detect and decode the packet. This type of feature can be found in Multiprotocol Label Switching.

- Lastly if we can use the transceiver in a TDMA environment, then it would eliminate the packet detect and synchronization procedure. Not only would it eliminate all the additional hardware required for packet detection and packet timing, it would also allow the relayed signals to be transmitted without interference and also decoded at the next hop is required. But this method would require prior handshaking between the transmitter and the receiver to align their respective clocks. Also clock drifts play a crucial role in TDMA networks. Since OFDM is a FFT based modulation scheme, exact alignment of the FFT window at the receiver is very important. Therefore performance of TDMA networks and eliminating the packet detect and synchronization procedure is something to be investigated and then implemented using this transceiver.

5.2.1 Design and Implementation

Since switching subcarriers involves rearrangement of the subcarriers, buffering of samples is an integral part of the design. For this implementation we keep the same transmitter design as described in section 5.1 with two subchannels. The selection of the subchannel is controlled by the host controller. The host controller also has control signals to either receive the packet or relay it towards the destination. Also decoding can be performed on either subchannels as shown in section 5.1 or on all subcarriers as
done in conventional 802.11a/g compliant OFDM receiver.

The equalizer performs equalization only on the subchannel selected as mentioned in section 5.1. The equalization required to restore the modulation levels of the subcarriers. Otherwise destination node will not be able to equalize the the cumulative effect of multiple frequency selective channel over which the signal has been relayed through multiple intermediate nodes. Fig.5.4 shows the implementation of the switch. A set of registers “switch_reg” control the dataflow to the “FIFO_switch” which performs the buffering and switches the order in which data is fed to the IFFT Block. For IFFT, the IFFT unit from the transmitter is used. This calls for additional control signal for the transmitter as well to distinguish between the data coming in from a originating node or a relayed signal. This is required due to the transmitter design which had been explained in [6].

The spectrum of the of the incoming signal and the operation of the switch is shown in fig.5.5. Fig.5.5(a) shows the incoming signal having data on all subcarriers. Fig.5.5(b) shows subchannel #2 has been chopped off at the equalizer and then equalized as shown in fig.5.2. This signal with data on one of the subchannel is fed to the frequency switch which is then switched over to subchannel #1 as shown in fig.5.5(c) and then transmitted using the transmitter side front-end.
Figure 5.4: Frequency Domain Switch using System Generator

Figure 5.5: Input and Output Spectrum for Switch
Chapter 6

Performance Evaluation and Analysis

This chapter shows the performance of different receiver sub-systems. To show this we have used actual 802.11a/g Wi-Fi packets using a suitable front-end radio which downconverts the RF signal to baseband and then the analog signal is converted to baseband digital samples by the ADC of the FPGA. The receiver has been tested with different types of modulation and in different type of environment. In this chapter some of the key results have been highlighted.

6.1 Packet Detection and timing

In section 4.3 and section 4.5 the functioning of the packet detect and timing has been detailed. Fig. 4.3 and fig. 4.7 shows the output of the packet detect and timing in absence of noise. Fig.6.1 shows the output of the packet detection and timing blocks in presence of noise. Although the energy detector shown in fig.6.1(c) shows multiple detection, we actually don’t care about those because its the fist rising edge that detects the packets and latched for resetting the entire receiver pipeline. It is clear that the shape of the short preambles symbols are far from ideal. But since the packet detection uses the periodicity of the preambles, autocorrelation still exists which is seen in figure fig.6.1(b). Therefore its relative energy that is important in packet detection and not the absolute energy in the signal.

Figure 6.2 shows the correlation of the long preamble for the same noisy signal.
Figure 6.1: Packet detection in presence of channel noise

The correlation peaks are no longer same in height as in figure 4.7 because noise affects the two long preambles independently.

Figure 6.2: Packet timing in presence of channel noise
6.2 Equalizer and Channel Estimation

6.2.1 Equalizer Performance

In this section the constellations for the received and equalized signals are discussed. Fig. 6.3 shows the constellations for BPSK, QPSK and 16QAM modulations. It is seen that BPSK and QPSK are decoded with 100 percent accuracy due to their large noise margin only if they had the required signal to noise ratio and enough signal power to pass the threshold test of the packet detect and timing blocks. As the constellations gets tighter the margin of error reduces and decoding becomes more and more difficult. But the intrleaver, FEC and scrambler still allows us to decode information bits under poor channel condition. For equalization a linear interpolation techniques has been employed. The piece-wise linear interpolation can be smoothened by passing the piecewise interpolation through a low pass filter, which is believed to provife better results. Furthermore with the use of 256 subcarriers we will be able to use non-linear interpolation techniques. This work is to be considered as one of the future additions to the current design.

6.2.2 Channel Estimation

In order to provide a better picture of the channel used for the experiments we plot the channel estimates as given by the pilots. Fig. 6.4 shows the channel estimates for a BPSK modulated pilot subcarriers for 11 OFDM symbols = 44µs. the variance for four different pilot is certainly not the same and also their range are different. It is of interest to note that the channel changes not only over multiple symbol but also within one symbol.

This exercise has been carried out to assess the need for the pilot symbols in every OFDM symbol. Since pilot subcarriers occupy certain part of the spectrum it would be helpful if we can still decode the information bits without the pilots. This is helpful when
we split the spectrum into smaller subchannels then with the current setup a minimum ratio is to be maintained between subcarriers and pilots. Another method that has been investigated in [21] is to use the long sequence of known subcarriers at the beginning of the signal symbol. This known sequence can be used to estimate the channel over
all subcarriers. But as shown in fig.6.4(b) and fig.6.4(d) the pilots vary differently over time and the channel is independent from one symbol to other. Therefore we cannot use the channel estimate from one symbol to equalize a different symbol.

6.3 Hardware Utilization

The overall hardware utilization is shown in fig.6.5 for the FPGA platform chosen for this implementation. The following functional blocks have been included in the transceiver:
(1) The OFDM Transmitter

(2) The OFDM Receiver

(3) Frequency Switch

(4) Subchannel Receiver

(5) PCI Interface and DMA transfer Logic

(6) ZBT Memory Module required for the transmitter [6]

![Figure 6.5: Total hardware Utilization on a Virtex-IV](image)

The receiver block can be further broken down to smaller subsystems and hardware utilization of individual subsystem has been shown in fig.6.6. The equalizer consumes the maximum hardware because of the use of cordic arctan, division and sin/cos IPs. Also there are two separate equalizers in it, one for full spectrum equalization and one for subchannel equalization. The hardware utilization for the receiver module only is also given in fig.6.7. The utilization is calculated after the MAP process and are collected using the resource estimator tool in system generator.

### 6.4 Receiver Latency

The latency involved in the receiver is discussed in this section. This is required to determine the turnaround time for the receiver, which also affects the latency in
the network. Identifying latency in different blocks helps to identify bottlenecks in the pipeline and motivate design changes as well. It seems that the FFT core has the maximum latency. Usually for any practical transceiver we are interested in the minimum time that is required for the MAC/PHY to receive the last symbol of a frame at the air interface process the frame and respond with the first symbol on the air interface of the response frame. This includes receiver side PHY layer processing delay + MAC processing delay + Transmitter side processing delay + PCI transfer delay for both Rx and Tx + Front-end radio hardware delay. If we disregard the MAC processing
delay and the PCI transfer delay then we can summarize the following:

(1) Receiver side:
Difference between the last symbol received at the air interface to last bit transferred to host = 14.83 µsec.

(2) Transmitter side:
Difference between the FIFO read signal to the first analog sample out from the DAC = 11.68 µsec.

(3) Time to relay a signal on the fly after switching subchannels = 18.8 µsec. This includes the time for FFT and then time required to perform the IFFT at the transmitter side.

(4) Key note: The FFT/IFFT module consumes bulk of the latency = 7.4 µsec. x 2 (for Tx and Rx) = 14.8 µsec.

It is observed that most of the time is consumed by the FFT/IFFT unit and other than that the latency is attributed largely to various buffering elements required for proper functioning of the pipeline. In order to further reduce latency we need to use better pipelined cores with faster cycle times.
Chapter 7

Conclusion

In this thesis a complete design and implementation of an OFDM receiver in FPGA has been described. Performance analysis shows that the hardware works as good as any commercial signal analyzer with OFDM decoder. However, it has also been investigated that using a fewer number of subcarriers compared to the FFT size and using pilot based linear interpolation equalizer does not provide the best results as the channel starts to degrade and the constellation gets tighter. It can also be concluded that there is scope for optimizing the hardware utilization if all 256 subcarriers are used at the transmitter. The real challenge in multichannel cut through switching over multiple hops is the synchronization and signal detection not to mention the need of equalization. The synchronization and packet detection can be substituted with an out of band handshaking with very accurate clocks at both ends. However the narrow margin of error in alignment of FFT window between the transmitter and the receiver does pose a challenge in actual deployment of such system. This is something worth investigating in future. Equalization has proved to be another important operation in the receiver or the relay chain. As mentioned in section 6.2.2 that even for static nodes channel characteristics cannot be assumed to be constant over the duration of the packet transmission. Channel estimation by using a known sequence to capture sufficient channel information and then use that estimate to equalize the OFDM symbols that follow will not provide the optimum solution. Therefore, equalization still needs
to be done on a symbol basis to ensure correct decoding of the information bits. To
conclude we also understand that in order for the transceiver to work as a physical layer
router we need an equally intelligent path access control mechanism and interference
aware channel assignment algorithms to route packets on-the-fly without involvement
of higher layers.
Chapter 8

Future Work

The transceiver described in this thesis has the potential to open new avenues in future research in the field of wireless mesh networks. A relay oriented physical layer hardware allows to route packets without little or no involvement of the higher layers. In order to accomplish this an interference aware channel assignment mechanism and channel adaptation over multiple hops are very important. Also, scalability of the existing design is very important towards an actual deployment of a low latency mesh network. Therefore, we can lay out the following as possible addition to the existing design:

(1) Customize the receiver to decode on all 256 subcarriers. For this we need to define pilots for FFT size greater than 64.

(2) Embed sub-channel information in the signal field or define new protocols. Also define preambles for each sub-channel for synchronization without interfering with relayed signals.

(3) Define channel assignment algorithms and path access mechanisms.

(4) A very promising but challenging alternative would be to use TDMA to synchronize the transmitter and the receiver.

(5) Another useful addition would be to handle mobility in the receiver in terms of Doppler correction along long data packets.
Bibliography


