mobile communications series

SOFTWARE-DEFINED RADIO for ENGINEERS

TRAVIS F. COLLINS
ROBIN GETZ
DI PU
ALEXANDER M. WYGLINSKI

Software-Defined Radio for Engineers

For a listing of recent titles in the *Artech House Mobile Communications*, turn to the back of this book.

Software-Defined Radio for Engineers

Travis F. Collins
Robin Getz
Di Pu
Alexander M. Wyglinski

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the U.S. Library of Congress.

British Library Cataloguing in Publication Data
A catalog record for this book is available from the British Library.

ISBN-13: 978-1-63081-457-1

Cover design by John Gomes

10 9 8 7 6 5 4 3 2 1

Dedication

To my wife Lauren
—Travis Collins

To my wonderful children, Matthew, Lauren, and Isaac, and my patient wife, Michelle—sorry I have been hiding in the basement working on this book. To all my fantastic colleagues at Analog Devices: Dave, Michael, Lars-Peter, Andrei, Mihai, Travis, Wyatt and many more, without whom Pluto SDR and IIO would not exist.
—Robin Getz

To my lovely son Aidi, my husband Di, and my parents Lingzhen and Xuexun
—Di Pu

To my wife Jen
—Alexander Wyglinski

Contents

CHAPTER 10

Orthogonal Frequency Division Multiplexing

Until now, we have studied several single-carrier modulation schemes where the input binary bits are modulated by a carrier signal with a center frequency f_c. However, there are other approaches where data can be communicated across a channel, including a technique referred to as *multicarrier modulation*. Instead of having one center frequency, multicarrier modulation (MCM) multiplexes serial input data into several parallel streams and transmits them over independent subcarriers. These subcarriers can be individually modulated and manipulated, allowing for their optimization with respect to the channel. The ability of MCM to tailor its transmission parameters across the different subcarriers is especially useful when combating frequency selective fading channel environments. Consequently, most high data rate communication systems, including all digital subscriber line (xDSL) modems [1–3], as well as most commercial communication standards, including Wi-Fi [4, 5], Wi-MAX [6, 7] and LTE [8, 9], use some form of MCM at the core of their implementations. In this chapter, we will explore one of the most popular forms of MCM: orthogonal frequency division multiplexing (OFDM).

10.1 Rationale for MCM: Dispersive Channel Environments

From our high school physics courses, we learned about the fundamentals of wave propagation and how they combine constructively and destructively (e.g., the ripple tank experiment). The exact same principles hold in high-speed data transmission. For example, in a wireless communication system, the transmitter emanates radiation in all directions (unless the antenna is directional, in which case the energy is focused at a particular azimuth). In an open environment, like a barren farm field, the energy would continue to propagate until some of it reaches the receiver antenna. As for the rest of the energy, it continues on until it dissipates.

In an indoor environment, as depicted in Figure 10.1(c), the situation is different. The line-of-sight component (if it exists), p_1, arrives at the receiver antenna first, just like in the open field case. However, the rest of the energy does not simply dissipate. Rather, the energy is reflected by the walls and other objects in the room. Some of these reflections, such as p_2 and p_3, will make their way to the receiver antenna, although not with the same phase or amplitude. All these received components are functions of several parameters, including their overall distance between the transmitter and receiver antennas as well as the number of reflections.

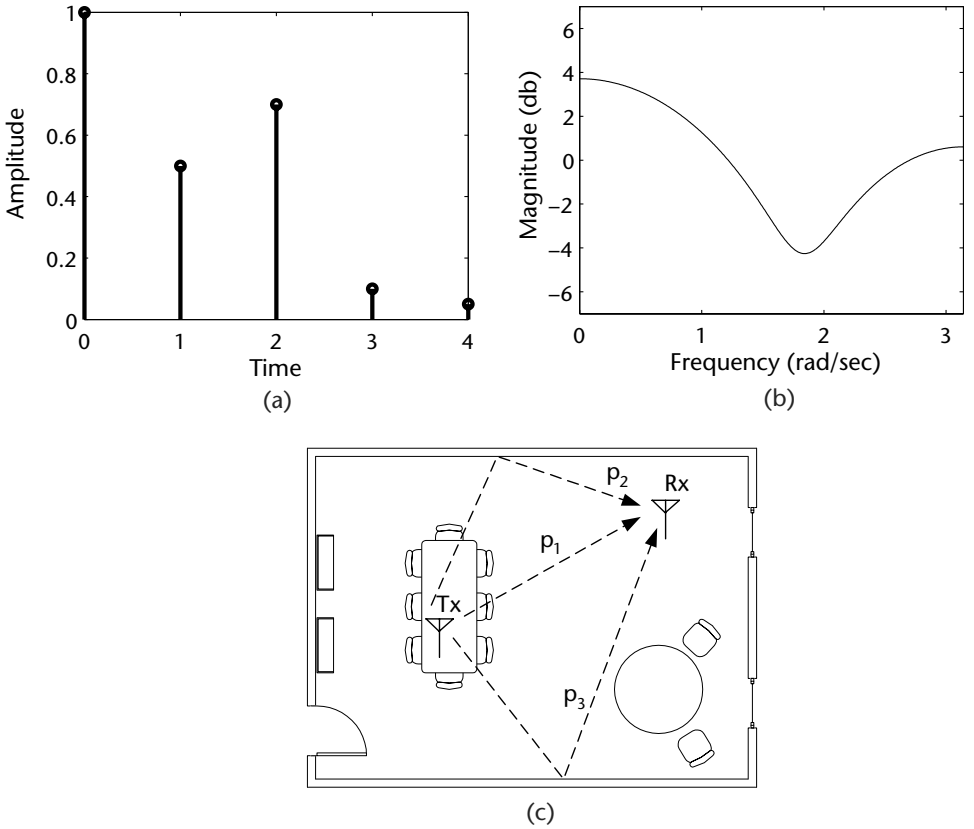

Figure 10.1 Example of a channel response due to dispersive propagation. Notice the three distinctive propagate paths, p_1, p_2, and p_3, that start at the transmitter Tx and are intercepted at the receiver Rx. (a) Impulse response, (b) frequency response, and (c) the process by which dispersive propagation arises.

At the receiver, these components are just copies of the same transmitted signal, but with different arrival times, amplitudes, and phases. Therefore, one can view the channel as an impulse response that is being convolved with the transmitted signal. In the open field case, the *channel impulse response* (CIR) would be a delta, since no other copies would be received by the receiver antenna. On the other hand, an indoor environment would have a several copies intercepted at the receiver antenna, and thus its CIR would be similar to the example in Figure 10.1(a). The corresponding frequency response of the example CIR is shown in Figure 10.1(b).

When observing Figure 10.1(b) more closely for the case of an operating environment with significant multipath characteristics, we can observe that the spectrum of the CIR appears to vary across the frequency domain. This is very problematic for any signal being transmitted across this type of channel since these signals will experience nonuniform attenuation that varies across frequency, which is relatively difficult to mitigate within the context of a single-carrier communication system since they will require the design and implementation of complex equalizer filters. On the other hand, MCM communication systems are well suited to handle this type of distortion since their multiple sub carriers are designed to divide-and-conquer the channel and treat each frequency-dependent attenuation individually, thus resulting in an implementation that has relatively low complexity.

10.2 General OFDM Model

OFDM is an efficient form of MCM, which employs the DFT and inverse DFT (IDFT) to modulate and demodulate multiple parallel data streams. As shown in Figure 10.2, the general structure of an OFDM communication system consists of a $2N$-point IDFT and a $2N$-point DFT at the transmitter and the receiver. The OFDM transceiver in Figure 10.2 operates as follows: A high-speed digital input, $d[m]$, is demultiplexed into N subcarriers using a commutator. The data on each subcarrier is then modulated into an M-QAM symbol, which maps a group of $\log_2(M)$ bits at a time. For subcarrier k, we will rearrange $a_k[\ell]$ and $b_k[\ell]$ into real and imaginary components such that the output of the modulator block is $p_k[\ell] = a_k[\ell] + jb_k[\ell]$. In order for the output of the IDFT block to be real, given N subcarriers we must use a $2N$-point IDFT, where terminals $k = 0$ and $k = N$ are don't-care inputs. For the subcarriers $1 \leq k \leq N-1$, the inputs are $p_k[\ell] = a_k[\ell] + jb_k[\ell]$, while for the subcarriers $N+1 \leq k \leq 2N-1$, the inputs are $p_k[\ell] = a_{2N-k}[\ell] + jb_{2N-k}[\ell]$.

The IDFT is then performed, yielding

$$s[2\ell N + n] = \frac{1}{2N} \sum_{k=0}^{2N-1} p_k[\ell] e^{j(2\pi nk/2N)}, \tag{10.1}$$

where this time $2N$ consecutive samples of $s[n]$ constitute an OFDM symbol, which is a sum of N different QAM symbols. This results in the data being modulated on several subchannels, which is achieved by multiplying each data stream by a $\sin(Nx)/\sin(x)$, several of which are shown in Figure 10.3.

The subcarriers are then multiplexed together using a commutator, forming the signal $s[n]$, and transmitted to the receiver. Once at the receiver, the signal is demultiplexed into $2N$ subcarriers of data, $\hat{s}[n]$, using a commutator and a $2N$-point DFT, defined as

$$\bar{p}_k[\ell] = \sum_{n=0}^{2N-1} \hat{s}[2\ell N + n] e^{-j(2\pi nk/2N)}, \tag{10.2}$$

is applied to the inputs, yielding the estimates of $p_k[\ell]$, $\bar{p}_k[\ell]$. The output of the equalizer, $\hat{p}_k[\ell]$, is then passed through a demodulator and the result multiplexed together using a commutator, yielding the reconstructed high-speed bit stream, $\hat{d}[m]$.

10.2.1 Cyclic Extensions

With the CIR be modeled as a finite impulse response filter that is convolved with a sampled version of the transmitted signal, this results in the CIR smearing past samples onto current samples, which are smeared onto future samples. The effect of this smearing causes distortion of the transmitted signal, thus increasing the aggregate BER of the system and resulting in a loss in performance.

Although equalizers can be designed to undo the effects of the channel, there is a trade-off between complexity and distortion minimization that is associated with the choice of an equalizer. In particular, the distortion due to the smearing of a previous OFDM symbol onto a successive symbol is a difficult problem. One simple solution is to put a few dummy samples between the symbols in order to capture the

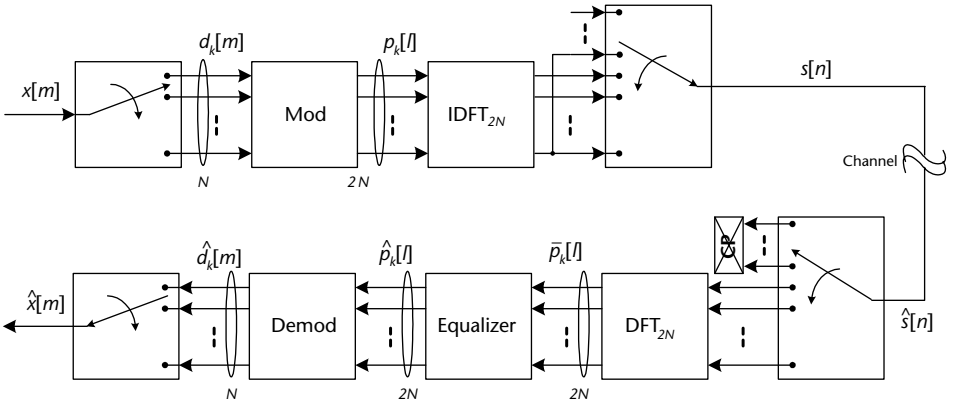

Figure 10.2 Overall schematic of an orthogonal frequency division multiplexing system, where the DFT and IDFT are employed to modulate and demodulate the data streams.

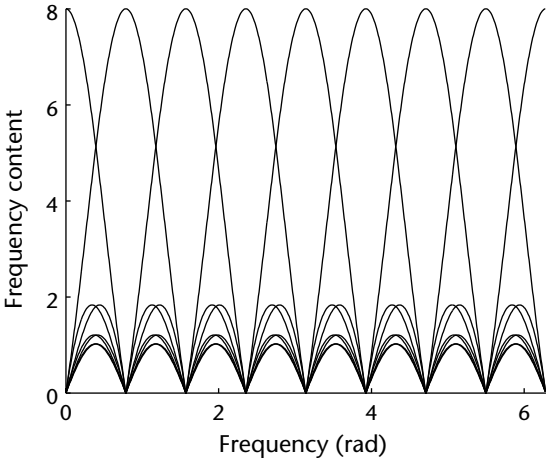

Figure 10.3 Characteristics of orthogonal frequency division multiplexing: frequency response of OFDM subcarriers.

intersymbol smearing effect. The most popular choice for these K dummy samples are the last K samples of the current OFDM symbol. The dummy samples in this case are known as a *cyclic prefix* (CP), as shown in Figure 10.4(a).

Therefore, when the OFDM symbols with cyclic prefixes are passed through the channel, the smearing from the previous symbols are captured by the cyclic prefixes, as shown in Figure 10.4(b). As a result, the symbols only experience smearing of samples from within their own symbol. At the receiver, the cyclic prefix is removed, as shown in Figure 10.4(c), and the OFDM symbols proceed with demodulation and equalization.

Despite the usefulness of the cyclic prefix, there are several disadvantages. First, the length of the cyclic prefix must be sufficient to capture the effects of the CIR. If not, the cyclic prefix fail to prevent distortion introduced from other symbols. The second disadvantage is the amount of overhead introduced by the cyclic prefix. By adding more samples to buffer the symbols, we must send more information across the channel to the receiver. This means to get the same throughput as a system without the cyclic prefix, we must transmit at a higher data rate.

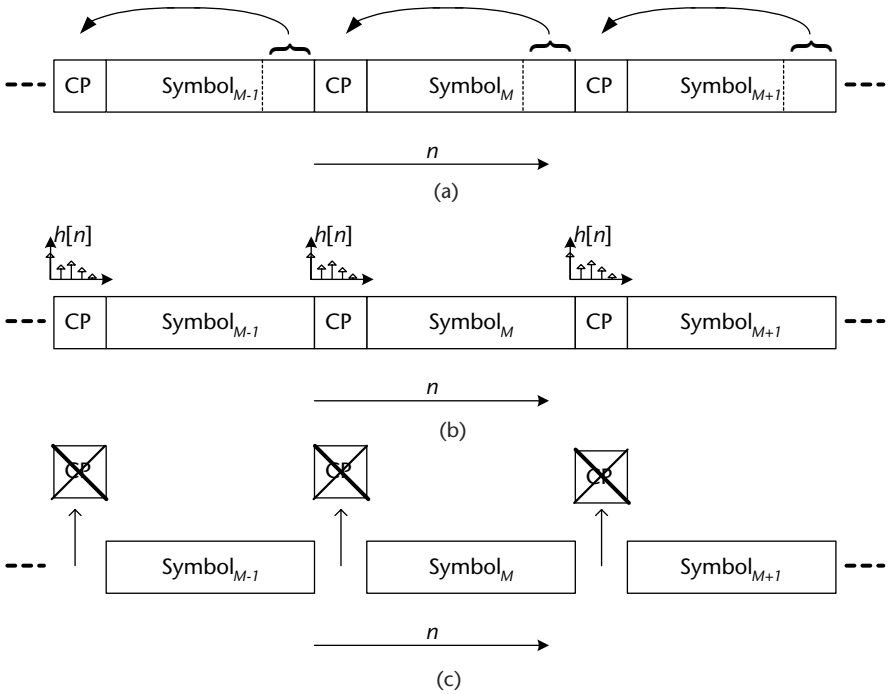

Figure 10.4 The process of adding, smearing capturing, and removal of a cyclic prefix. (a) Adding cyclic prefix to an OFDM symbol, (b) smearing by channel h(n) from previous symbol into cyclic prefix, and (c) removal of cyclic prefix.

10.3 Common OFDM Waveform Structure

In Section 10.2, we discussed a general OFDM transmission model and the CP used to limit ISI. Next we will examine a common structure within the OFDM symbol itself used among many standards such as IEEE 802.11. First, the number of subcarriers of an OFDM symbol N_s will typically be a base two number since the FFT and IFFT, which are efficient implementations of the DFT and IDFT used to modulate and demodulate the data are most efficient at these lengths. However, in a typical implementation N_g guard carriers will be utilized that occupy the outer frequency positions of the OFDM symbol. For example, in Figure 10.5, where $N_s = 64$ and $N_g = 14$, the first seven subcarriers and the last seven subcarriers are unoccupied. This is done to limit the out-of-band interference impacting the surrounding signals occupying neighboring channels. Nonetheless, the downside of using guard carriers is that it reduces the overall data rate of the transmission. Note that this is the OFDM symbol before the CP is added, which is of length N_{cp}. Furthermore, select subcarriers are chosen as training or pilot tones, which are used by equalizers at the receiver to correct for phase and frequency offsets, along with the channel effects. Similar to Figure 10.5, the selected subcarriers are often uniformly selected across the symbol in order to provide characterization across the entire OFDM symbol. These pilots tones will be known at the receiver.

Once OFDM modulation and CPs have been applied to the modulated symbols, additional structure can be added to the frame prior to analog transmission. These additions are based on the transmission type. For example, for continuously

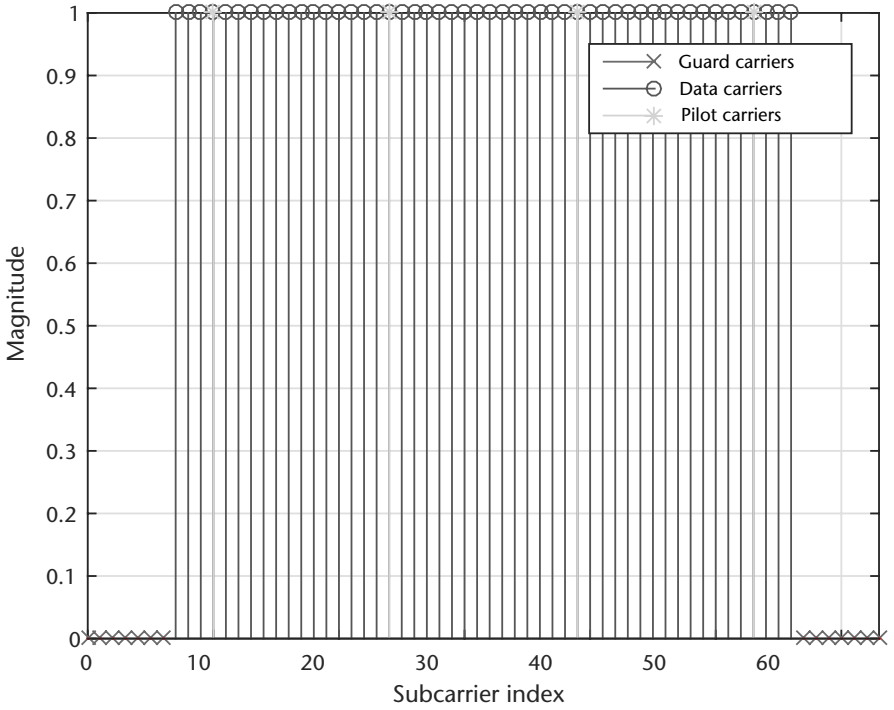

Figure 10.5 OFDM symbol subcarrier layout with data and guard carriers.

transmitting systems, such as LTE or broadcast signals, small insertions are made into the individual frames at a periodic rate in order to enable the synchronization of the receivers. These training or synchronization symbols can be minimal since the receiver can attempt to synchronize multiple times with the transmitter. However, in burst-based systems, such as wireless local area networks (WLANs), there will be large preamble structures that provide the receivers with more time to synchronize. This additional time allows recovery of frames with high probability without the need for retransmission. Retransmission can be expensive in terms of performance costs with wireless systems that have *carrier sense multiple access* (CSMA)/*collision avoidance* (CA) MAC schemes.

We will first examine the frame structure of WLAN since they are a common implementation of an OFDM system. Digital video broadcast terrestrial (DVB-T2) uses a similar structure to WLAN. However, the preamble data is modulated in a different way than WLAN [10]. We have chosen to examine OFDM transmission and reception from a standard's perspective for two reasons. First, OFDM implementations can be very unique based on their system design but generally follow two patterns, which are discussed here. Second, the standards discussed in this section are used extensively in industry and understanding how they work can be invaluable since engineers will most likely interact with WLAN or LTE at some point in their careers.

From Figure 10.6, we observe that the WLAN frame has a preamble that is made of up four full OFDM symbols and will always remain the same regardless of the mode or MCS link settings. The first section identified as the short portion of the preamble, called the Legacy Short Training Field (LSTF) in the IEEE 802.11

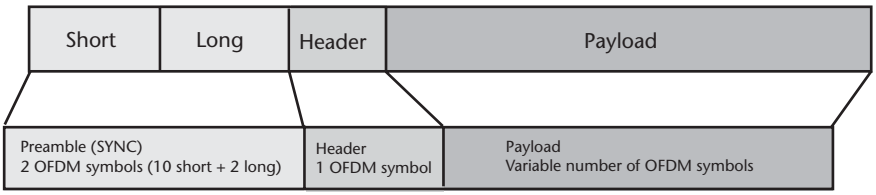

Figure 10.6 WLAN 802.11a frame outlining preamble, header, and payload sections.

standard [11], contains ten repeated copies of a short sequence. The purpose of this sequence is to allow the automatic gain control (AGC) of the receiver to stabilize over the 8 μs it is given and to perform carrier frequency offset (CFO) estimation. Generally, this is considered the packet detection phase of the receiver.

Once the packet has been detected and CFO corrected, the receiver will then utilize the long field, called the Legacy Long Training Field (LLTF) in the IEEE 802.11 standard [11], containing two repeated sequences and a CP. This portion of the preamble is designed to be used for channel estimation. Overall, an OFDM receiver will first identify the start of the packet, correct for frequency offsets, correct for channel effects, and remove residual phase distortions. This processing may be considered somewhat backward compared with the previous chapters that considered signal carrier transmissions.

Once the preambles have been utilized to correct the received frame, the header symbol will be utilized to determine the length and MCS of the remaining payload. Finally, pilot tones in the payload will be used to correct any remaining offsets and channel distortions. A similar frame can be generated from Code 10.1.

To provide an alternative perspective regarding standards that utilize OFDM, LTE is a great example to consider. Unlike WLANs, which focus all synchronization within a few symbols of the preamble, LTE uses primary synchronization signals (PSS) and secondary synchronization signals (SSS), as shown in Figure 10.7. The receiver detects via cross correlation, and utilizes both the frame boundaries along with the pilot tones sprinkled throughout the resource blocks (RBS) in order to correct at the receiver. However, since the downlink signals are constant at small intervals, if a PSS or SSS is missed it can be simply located on the next sequence provided. Once synchronized, the receiver will continually decode information from the base stations, unlike WLAN, which is more burst-based.

10.4 Packet Detection

The reasoning behind the structure of the LSTF in the preamble is based on the work by Schmidl and Cox [12]. In their paper, they outlined a symbol timing recovery stategy that relies on searching for a training symbol with two identical halves in the time domain, which will remain identical after passing through the channel, except that there will be a phase difference between them caused by the carrier frequency offset. The two halves of the training symbol are made identical (after the IFFT) by transmitting a pseudonoise (PN) sequence on the even frequencies while zeros are used on the odd frequencies. This means that at each even frequency one of the points of a source constellation (rotated BPSK for WLAN) is transmitted. Given this structure we can accomplish two tasks: first, estimate the start of the preamble

Code 10.1 Generate OFDM Packet: **genOFDMPacket.m**

```
NumFrames = 1;
%% Build OFDM Modulator
FFTLength            = 64;
NumGuardBandCarriers = [6; 5];
NumDataCarriers      = 48;
CyclicPrefixLength   = 16;
PilotCarrierIndices  = [12;26;40;54];
NumOFDMSymInPreamble = 5;
NumBitsPerCharacter  = 7;
% Convert message to bits
msgInBits = repmat(randi([0 1], NumDataCarriers, 1),10, 1);
PayloadBits = msgInBits(:);
% Calculate number of OFDM symbols per frame
NumOFDMSymbols = ceil(length(PayloadBits)/NumDataCarriers);
% Calculate number of bits padded in each frame
NumPadBits = NumDataCarriers * NumOFDMSymbols - length(PayloadBits);
% Get preamble for each frame
Preamble = double(getOFDMPreambleAndPilot('Preamble', ...
    FFTLength, NumGuardBandCarriers));
% Get pilot for each frame
Pilots = double(getOFDMPreambleAndPilot('Pilot', NumOFDMSymbols));
% BPSK modulator
BPSKMod = comm.BPSKModulator;
% OFDM modulator
DataOFDMMod = comm.OFDMModulator(...
    'FFTLength' ,           FFTLength, ...
    'NumGuardBandCarriers', NumGuardBandCarriers, ...
    'InsertDCNull',         true, ...
    'PilotInputPort',       true, ...
    'PilotCarrierIndices',  PilotCarrierIndices, ...
    'CyclicPrefixLength',   CyclicPrefixLength, ...
    'NumSymbols',           NumOFDMSymbols);

%% Modulate Data
symPostBPSK = BPSKMod.step([PayloadBits; randi([0 1], NumPadBits, 1)]);
% OFDM modulation for one frame
symPostOFDM = DataOFDMMod.step(reshape(symPostBPSK, ...
    NumDataCarriers, NumOFDMSymbols), Pilots);
% Repeat the frame
y = repmat([Preamble; symPostOFDM], NumFrames, 1);
```

using correlation in the time domain due to the known repeating sequences, and second, estimate the frequency base on the phase difference between the halves of the training symbols. There are many extensions to [12], such as Minn [13] and [14], but all maintain the principle of symmetric preambles across frequency and time.

Mathematically, the packet detector's timing metric from [12] is expressed as

$$M(k) = \frac{\sum_{m=0}^{L-1} r^*(k+m)r(k+m+L)}{\sum_{m=0}^{L-1} |r(k+m+L)|^2}, \tag{10.3}$$

where r is the received signal and L is the length of halve a training symbol. In WLAN, the length is equal t0 16 samples or 0.8 μs long. Equation (10.3) takes advantage of the the repeated sequences in time of length L present in the preamble.

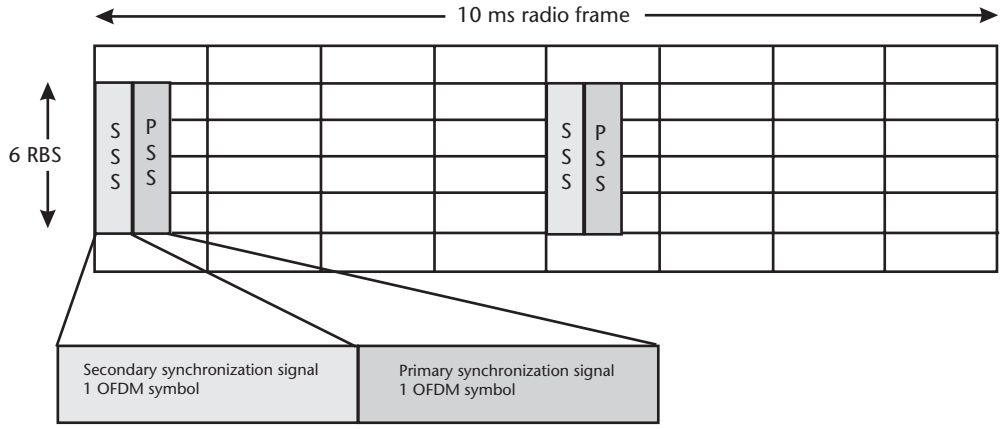

Figure 10.7 Simplified outline of LTE downlink frame with a focus on primary and secondary synchronization signals.

In Code 10.2, a simple example is provided that performs the operation shown in (10.3), which relies on our previous Code 10.1 to generate the necessary OFDM packet based on WLAN 802.11a specifications. The resulting metric is plotted in Figure 10.8, where we can observe an obvious plateau starting at the true position of the preamble start for different AWGN SNR values. At low SNR the plateau from the LSTF sequence becomes more difficult to observe, but still and be identified. Nonetheless, the purpose of the metric M is to determine in a binary degree if a packet was transmitted. The exact start of the preamble is not required, but still a rough estimate where the LSTF is located in the data is only required during this phase of the receiver.

In practice, this method becomes a threshold detection problem and we must build the receive to deal with different conditions. Equation (10.3) is an autocorrelation implementation since it utilizes only the received signal. This implementation is useful since it self-normalizes the input, trying to force the output to be between 0 and 1. This is not always possible, as observed in Figure 10.8, but the preamble does extend above the payload data for the output M even at low SNR. Therefore, it is important to select the correct value when thresholding the metric M in order to determine if a packet exists in the signal space. This parameter is actually tunable in some commercial hardware from Cisco Systems, which is defined as receiver start of packet detection Threshold (Rx SOP) [15]. Conceptually, changing this parameter will change change your wireless cell size and inversely the received packet error rate.

10.5 CFO Estimation

One problem associated with OFDM is the effect of frequency offsets. When the received symbols experience an offset greater than half the subcarrier bandwidth, the signal becomes nonorthogonal, which breaks the recovery assumptions associated with the signal. Therefore, before the OFDM signal is demodulated (i.e., processed via FFT), frequency correction must be considered. As mentioned in Section 10.4, the phase difference of sequence halves of the preamble are directly

Code 10.2 Detect Packet Start: **packetDetect.m**

```
%% Generate OFDM Waveform
genOFDMPacket;
% Add random offset
offset = randi([0 1e2]);
y = [zeros(offset,1);y];

%% Schmidl and Cox: Coarse Packet Detection
L = 16;  % Short sync field length
m = L;   % Distance between fields
N = 300; % Autocorrelation samples
M = zeros(N,1);
SNR = [0,5,10];
for SNRindx = 1:length(SNR)
    r = awgn(y,SNR(SNRindx),'measured');
    % Determine timing metric
    for k=1:N
        P = r(k:k+m)' *  r(k+L:k+m+L);
        a = abs(y(k+L:k+m+L));
        R = a'*a;
        M(k) = abs(P)^2/(R^2);
    end
    % Plot
    subplot(length(SNR),1,SNRindx);stem(M);
    hold on; stem(offset+1,M(offset+1),'r*'); hold off;
    grid on;xlabel('k');ylabel('M');
    legend('Autocorrelation','True Start');
    title(['SNR: ',num2str(SNR(SNRindx)),'dB']);
end
```

related to the frequency offset experience by receive waveform. This can be easily proven given a frequency offset Δ_f and the transmitted waveform $s(t)$:

$$r(t) = s(t)e^{j2\pi\Delta_f t/f_s} \tag{10.4}$$

Now utilizing the symbol halves in the LSTF, each of length L, we can write

$$\begin{aligned} p(k) &= \sum_{m=0}^{L-1} r^*(k+m)r(k+m+L) \\ p(k) &= \sum_{m=0}^{L-1} s(k+m)^* e^{-j2\pi\Delta_f(k+m)/f_s} s(k+m+L)e^{j2\pi\Delta_f(k+m+L)/f_s} \\ p(k) &= e^{j2\pi\Delta_f(L)/f_s} \sum_{m=0}^{L-1} s(k+m)^* s(k+m+L) \\ p(k) &= e^{j2\pi\Delta_f(L)/f_s} \sum_{m=0}^{L-1} |s(k+m)|^2. \end{aligned} \tag{10.5}$$

Therefore, we know the frequency offset is a direct result of the phase difference between the preamble halves. For a frequency offset of Δ_f, the phase difference ϕ

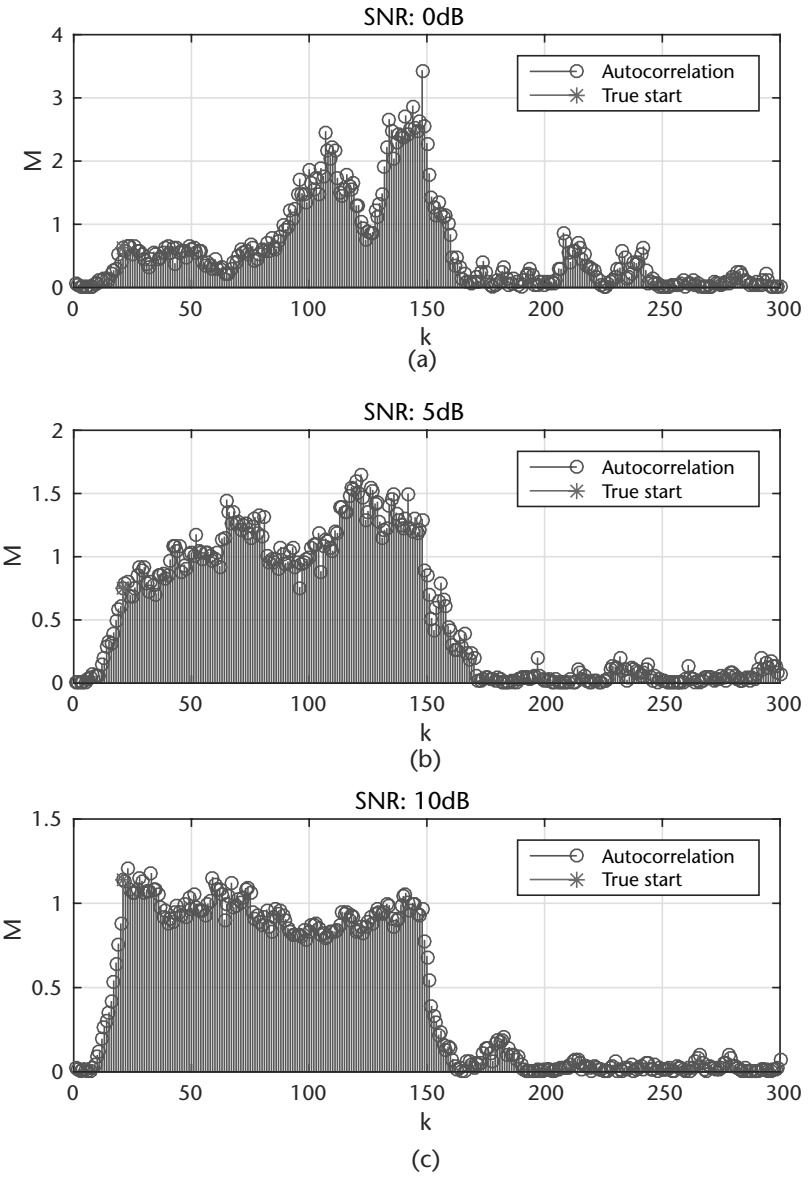

Figure 10.8 Timing metric M using in Schmidl and Cox algorithm at SNRs of 0, 5, and 10 dB. These provide rough estimates for the start of a receive frame in AWGN.

between halves of the preamble symbols can be directly related by

$$\phi = \frac{2\pi L \Delta_f}{f_s}, \tag{10.6}$$

where f_s is the sample rate. Therefore, ϕ can be estimated as

$$\hat{\phi} = tan^{-1}\left(\frac{\Im(P(k))}{\Re(P(k))}\right), \tag{10.7}$$

where

$$P(k) = \sum_{m=0}^{L-1} r^*(k+m)r(k+m+L), \tag{10.8}$$

and k in (10.7) is the within the first nine symbols of the short preamble sequence. Since ϕ can be at most π we can directly determine the largest offset:

$$\Delta_{f,max} = \frac{f_s}{2LN}. \tag{10.9}$$

In the case of WLAN 802.11a, $\Delta_{f,max} = 625$ kHz. However, in the IEEE standard a receiver must maintain a 20 PPM oscillator resulting in at most a 212-kHz difference between transmitter and receiver at 5.3 GHz [11].

Code 10.3 implements this technique from (10.5) to (10.8) over varying frequency offsets. The results are analyzed with respect to a normalized offsets. This is common since frequency offset will be dependent on the sampling rate of the system, which can be used to compensate for large offsets. Simply running a radio at a faster rate will reduce the normalized offset for the recovery algorithms.

Code 10.3 Determine CFO: **`freqEstOFDM.m`**

```
%% Generate OFDM Waveform
genOFDMPacket;
% Add random offset
offset = randi([0 1e2]); y = [zeros(offset,1);y];
SNR = 20;
%% CFO Estimation
L = 16;  % Short sync field length
m = L;   % Distance between fields
N = 300; % Autocorrelation samples
Fs = 1e6; % Sample rate
% CFO estimation over multiple frequencies
subchannelSpacing = Fs/FFTLength;
CFOs = subchannelSpacing.*(0.01:0.01:0.5);
cfoError = zeros(size(CFOs));
for cfoIndx = 1:length(CFOs)
    % Add noise
    r = awgn(y,SNR,'measured');
    % Frequency offset data
    pfOffset = comm.PhaseFrequencyOffset('SampleRate',Fs,...
        'FrequencyOffset',CFOs(cfoIndx));
    r = pfOffset(r);
    % Determine frequency offsets
    freqEst = zeros(N,1);
    for k=1:N
        P = r(k:k+m)' *  r(k+L:k+m+L);
        freqEst(k) = Fs/L*(angle(P)/(2*pi));
    end
    % Select estimate at offset estimated position
    freqEstimate = freqEst(offset);
    % Calculate CFO error
    cfoError(cfoIndx) = abs(freqEstimate-CFOs(cfoIndx))/subchannelSpacing;
end
```

Since the short section of the preamble contains ten copies of the same sequence, at most nine estimations can be made. However, it is unlikely that the early symbols of the short preamble will be detected or available due to AGC convergence. This is the reasoning behind providing additional copies of short preamble sequences. If the short preamble is recovered early, meaning a large number of the symbols were detected, then more frequency estimates can be made and eventually averaged together.

The long portion of the preamble maintains the same symmetry as the short portion, but there are only two halves to compare. Utilizing only the LLTF reduces the estimation range since L increases. Nonetheless, if the AGC is unable to lock in time the LLTF may need to be used for frequency estimation only. In the desirable case, where both the short and long preamble sequences can be used, a staged compensation design can be implemented. Similar to single carrier implementation discussed in Chapter 7, the short sequence can be used to provide a coarse offset estimate, and the long preamble can be used for fine frequency estimation. The estimation methods are segregated in this way due to the amount of data that can be used during estimation. Since the long preamble can generally use more data for a given calculation of P it can provide more reliable estimates. This analysis comes directly from [12], which estimates the variance of the estimate of Δ_f as

$$\sigma^2 = \frac{1}{L \times SNR}. \tag{10.10}$$

10.6 Symbol Timing Estimation

After packet detection and frequency correction, symbol timing can be performed. Symbol timing is similar to packet detection except that it provides symbol-level precision estimation of the preamble sequences in the time domain. To provide this additional precision a cross correlation is performed using the known transmitted preamble sequence. Using cross correlation provides better SNR of the correlation, but is not directly normalized to any range without additional scaling. This variable range of values make cross correlation not ideal for initial packet detection.

The cross-correlation technique applied in OFDM symbol timing estimation determines the boundary between the short and long training sequences with sample-level precision since at this point future usage of the LSTF is not required. An example implementation would be to utilize the first 80 samples of the LLTF sequence defined as $d_{LLTF}(t)$. The metric c of d_{LLTF} and the received signal r is

$$c(k) = \left| \sum_{m=0}^{L-1} d^*_{LLTF}(m) r(m+k) \right|^2. \tag{10.11}$$

In Code 10.4, two implementations for the calculation of c and are also presented in Figure 10.9. Both should provide identical results, but computationally the filter implementation will be more efficient.

Plotting c from either implementation in Code 10.4 should provide a plot similar to Figure 10.9, which clearly show two peaks 64 samples away from one another. These peaks represent strong correlations of the LLTF halves and will be pronounce

Code 10.4 Determine Fine Symbol Offset Metric: **`fineSymEst.m`**

```
genOFDMPacket;
% Add random offset with noise
offset = randi([0 1e2]); y = [zeros(offset,1);y];
SNR = 0; r = awgn(y,SNR,'measured');
%% Estimate fine sample offset
LSTF = Preamble(1:160);
LLTF = Preamble(161:161+160-1);
symLen = 80; % FFTLength + CPLen
known = LLTF(1:symLen,1);
% Filter
coeff = conj(flipud(known));
c_filt = abs(filter(coeff, 1, r)).^2;
% Correlation
m = abs(xcorr(r,known)).^2;
padding = length(r)-symLen+1;% Remove padding added to known sequence
c_cor = m(padding:end);
```

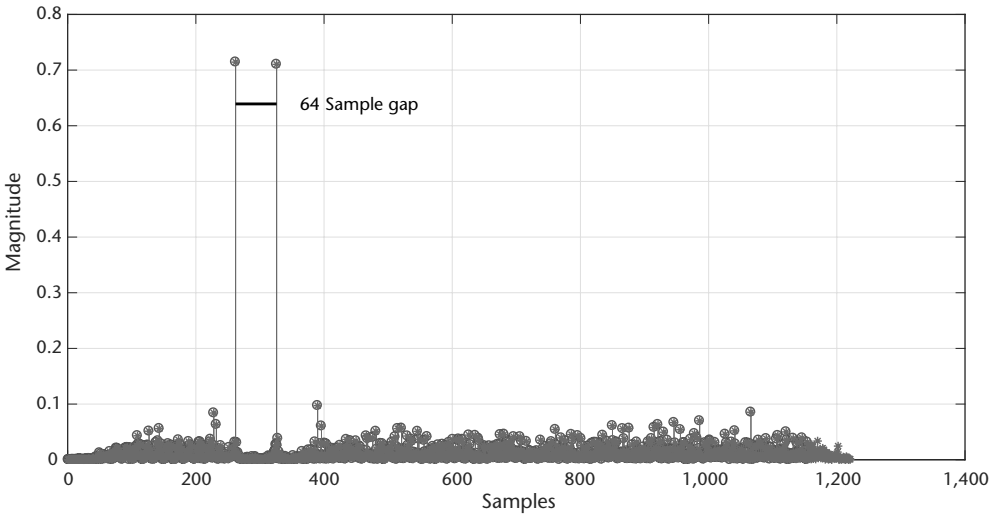

Figure 10.9 Symbol timing metric c using cross correlation for r at a SNR of 10 dB. Peaks at the positions of the LLTF sequences are clearly visible above the reset of the correlation samples.

in c even under low SNR conditions. From these sample positions OFDM symbol boundaries can be directly determined and OFDM demodulation can be performed with a FFT.

The actual offset can be calculated easily as follows in Code 10.5, since the first peak position will be shifted by half the length of the LLTF (1 OFDM symbol) due to the correlation operation.

10.7 Equalization

The last piece to the receiver is the equalizer, which is responsible for reducing channel effects and removing any residual phase or frequency offsets remaining the in the received signal. The technique discussed here is performed after OFDM

Code 10.5 Calculate Offset Position: **`fineSymEst.m`**

```
20 [v1,peak1] = max(c_cor);
21 c_cor(peak1) = 0;
22 [v2,peak2] = max(c_cor);
28 % Get numerical offset
29 if abs(peak2-peak1)==FFTLength
30     % Adjust position from start of LLTF
31     p = min([peak1 peak2]);
32     LLTF_Start_est = p-symLen;
33     LSTF_Start_est = LLTF_Start_est - length(LSTF);
34     % Display
35     disp([offset LSTF_Start_est]);
36 end
```

demodulation, although other methods do exist that can be implemented before the FFT operation.

One of the primary advantages of the cyclic prefix is that it helps transform the linear convolution between the transmitted signal $s[n]$ and the channel impulse response $h[n]$ into a symbol-by-symbol circular convolution. To clearly see this, let us take a closer look at a given OFDM symbol with cyclic prefix, the symbol starting at time $n = 0$. Denoting by $s[0], \ldots, s[2N-1]$ the $2N$ samples of output of the transmitter IDFT for the first OFDM symbol, the addition of the cyclic prefix gives rise to a new signal; namely,

$$\tilde{s}[n] = \begin{cases} s[n+2N-K] & 0 \le n \le K-1 \\ s[n-K] & K \le n \le 2N-1 \end{cases}$$

Denoted by $\tilde{r}[n]$, the result of the convolution of the signal $\tilde{s}[n]$ with the length-L channel impulse response $h[n]$ yields

$$\begin{aligned} \tilde{r}[n] &= \sum_{k=0}^{L-1} h[k]\tilde{s}[n-k] \\ &= \begin{cases} \displaystyle\sum_{k=0}^{n-K} h[k]s[n-K-k] + \sum_{k=n-K+1}^{L-1} s[n-k+2N-K] & K \le n \le K+L-1 \\ \displaystyle\sum_{k=0}^{L-1} h[k]s[n-K-k] & K+L \le n \le 2N-1 \end{cases} \end{aligned}$$

From the above equation, it is readily observed that after the removal of the cyclic prefix, the received sequence $r[n] = \tilde{r}[n+K]$ is equal to

$$r[n] = \sum_{k=0}^{2N-1} h[k]s[((n-k))_{2N}] = h[n] \textcircled{2N} s[n]. \tag{10.12}$$

Thus, the received samples resulting from the removal of the cyclic prefix are made up of the circular convolution of the sent signal (i.e., $2N$ samples per symbol) with the channel impulse response $h[n]$. Observing (10.12) in the frequency domain,

we see the following:

$$R[k] = H[k].S[k],$$

where capital letters represent $2N$-point DFTs of the corresponding sequences. Referring back to Figure 10.2, the $2N$-point DFT $R[k]$ of the received samples is already computed and is denoted by $\bar{p}_k[\ell]$.

Referring to Figures 10.3 and 10.1(b), if we consider the multiplication of the corresponding frequency samples together, we notice that each of the subcarriers experiences a different channel gain $H[k]$. Therefore, what must be done is to multiply each subcarrier with a gain that is an inverse to the channel frequency response acting on that subcarrier. This is the principle behind per tone equalization. Knowing what the channel frequency gains are at the different subcarriers, one can use them to reverse the distortion caused by the channel by dividing the subcarriers with them. For instance, if the system has 64 subcarriers centered at frequencies $\omega_k = 2\pi k/64$, $k = 0, \ldots, 63$, then one would take the CIR $h[n]$ and take its 64-point FFT, resulting with the frequency response $H[k]$, $k = 0, \ldots, 63$. Then, to reverse the effect of the channel on each subcarrier, simply take the inverse of the channel frequency response point corresponding to that subcarrier,

$$W[k] = \frac{1}{H[k]}, \tag{10.13}$$

and multiply the subcarrier with it.

Looking back at the WLAN-based frame structure, the LLTF can be best used for channel estimation since it will always been known at the receiver and has subcarriers that span the entire OFDM symbol. The LLTF has two complete OFDM symbols that are both used for channel estimation, averaging their estimates for more reliable results. To perform OFDM demodulation the `comm.OFDMDemodulator` will be utilize, which can be conveniently generated from a modulator object matching the necessary settings (see Code 10.8).

Code 10.6 OFDM Modulator/Demodulator: **`exampleMod.m`**

```
% Modulator
Mod = comm.OFDMModulator(...
        'FFTLength' ,           FFTLength,...
        'NumGuardBandCarriers', [6;5],...
        'CyclicPrefixLength',   0,...
        'NumSymbols', 2,...
        'InsertDCNull', true);
% Demodulator
Demod = comm.OFDMDemodulator(Mod);
```

Using this demodulator object the LLTF can be demodulated and the channel estimated using the least squares (zero-forcing) method described. This demodulation of the LLTF, channel estimation, and equalization of the data symbols with the LLTF is provided in Code 10.7. However, even with this additional correction frequency offsets can still exist in the data. Luckily, there are pilots still

embedded within the data symbols to correct for these problems. Code 10.7 provides equalization with these inserted pilots as well.

The pilots are utilized in a very simple way to correct for the additive phase rotation that can be experienced from symbol to symbol in the data. This rotation again is a complex gain and will be estimated in a similar way as the LLTF was used

Code 10.7 OFDM Equalization: **eqOFDM.m**

```
genOFDMPacket;
% Add random offset
offset = randi([0 1e2]); y = [zeros(offset,1);y];
%% Equalization Example
r = awgn(y(offset+1:end),15,'measured'); Fs = 1e6;
pfOffset = comm.PhaseFrequencyOffset('SampleRate',Fs,...
    'FrequencyOffset',Fs*0.001);
r = pfOffset(r);
%% Channel estimation
preambleOFDMMod = comm.OFDMModulator(...
    'FFTLength' ,           FFTLength,...
    'NumGuardBandCarriers', NumGuardBandCarriers,...
    'CyclicPrefixLength',   0,'NumSymbols', 2,'InsertDCNull', true);
od = comm.OFDMDemodulator(preambleOFDMMod);
od.PilotOutputPort = true;
% OFDM Demodulate LLTF
LLTF = Preamble(161:161+160-1); rLLTF = r(161+32:161+160-1);
[rLLTFFreq,rp] = od(rLLTF); [LLTFFreq,p] = od(LLTF(33:end));% remove CP
% Estimate channel
ls = rLLTFFreq./LLTFFreq; % Least-square estimate
chanEst = mean(ls,2); % Average over both symbols
CSI = real(chanEst.*conj(chanEst));
ls = rp./p; % Least-square estimate
chanEstPilots = mean(ls,2); % Average over both symbols
CSIPilots = real(chanEstPilots.*conj(chanEstPilots));
%% Perform Equalization
data = r(2*length(LLTF)+1:end);
odd = comm.OFDMDemodulator(DataOFDMMod);
[dataFreq,pilots] = odd(data);
% Apply LLTF's estimate to data symbols and data pilots
postLLTFEqData = bsxfun(@times, dataFreq, conj(chanEst(:))./CSI(:));
postLLTFEqPilots = ...
    bsxfun(@times, pilots, conj(chanEstPilots(:))./CSIPilots(:));
% Visualization objects
tt1 = comm.ConstellationDiagram;tt2 = comm.ConstellationDiagram;
tt2.Position = tt2.Position + [500 0 0 0];
% Estimate remaining offsets with pilots
correctedSymbols = zeros(size(postLLTFEqData));
for symbol = 1:size(postLLTFEqData,2)
    % Estimate rotation across pilots
    p = postLLTFEqPilots(:,symbol);
    e = conj(mean(p.*conj(Pilots(:,symbol))));
    % Equalize
    sig = postLLTFEqData(:,symbol).*e;
    correctedSymbols(:,symbol) = sig;
    % Visualize
    tt1(sig);tt2(postLLTFEqData(:,symbol));pause(0.1);
end
```

for equalization [16]. Assuming that the pilots in the data were also corrected by the LLTF estimates, the N_p pilots from the data OFDM $\hat{p}(k)$ symbols are evaluated against the known pilots $p(k)$ to produce a single gain:

$$e(k) = \frac{1}{N_p} \sum_{n=0}^{N_p-1} p_n(k)\hat{p}_n^*(k). \tag{10.14}$$

This complex gain $e(k)$ is then multiplied by each sample in the kth OFDM symbol. To be clear $p_n(k)$ is the nth known pilot of the kth OFDM symbol.

10.8 Bit and Power Allocation

Resource allocation is an important aspect in OFDM design, since it determines the BER performance of the OFDM system. Given a fixed bit rate and a transmitted power constraint, the BER can be minimized by properly allocating the bit and power levels over the subcarriers. In this section, we will introduce the theory and technique concerning the bit and power allocation.

Most OFDM systems use the same signal constellation across all subcarriers, as shown in Figure 10.10(a), where the commutator allocates bit groupings of the same size to each subcarrier. However, their overall error probability is dominated by the subcarriers with the worst performance. To improve performance, adaptive bit allocation can be employed, where the signal constellation size distribution across the subcarriers varies according to the measured SNR values, as shown in Figure 10.10(b), where the commutator allocates bit groupings of different sizes. In extreme situations, some subcarriers can be turned off or nulled if the subcarrier SNR values are poor.

Assuming that both the transmitter and receiver possess knowledge of the subcarrier SNR levels, one is able to determine the subcarrier BER values. Since the subcarrier BER values are dependent on the choice of modulation used for a given SNR value, we can vary the modulation scheme used in each subcarrier in order to change the subcarrier BER value. Larger signal constellations (e.g., 64-QAM) require larger SNR values in order to reach the same BER values as smaller constellations (e.g., 4-QAM) which have smaller SNR values.

We will simply allocate the number of bits b_i on subcarrier i according to

$$b_i = \log_2\left(1 + \frac{\gamma_i}{\Gamma}\right), \tag{10.15}$$

where γ_i is the SNR of subcarrier i (not in dB). Of course, (10.15) gives rise to noninteger numbers of bits around the resulting b_i's, appropriately.

Channel responses are not always linear. In particular, certain frequencies can be far more attenuated than others. It can be shown that the optimum power distribution for the subcarriers should be a constant K. Determining what this level should be is done through water pouring, or choosing the power at a frequency, $P(f)$, such that all $P(f_1) = P(f_n)$.

$P(f)$ is given by

$$P(f) = \begin{cases} \lambda - \frac{N(f)}{|H(F)|^2}, & f \epsilon F \\ 0, & f \epsilon F' \end{cases} \tag{10.16}$$

where $N(f)$ is the PDF of Gaussian noise, $H(f)$ is the transfer function representing a linear channel, and λ is the value for which

$$\int_F P(f)df = P, \tag{10.17}$$

or the area under the curve in Figure 10.11. This water filling is done so that the probability of a bit error is the same for all subcarriers.

10.9 Putting It All Together

So far, all the necessary receiver algorithms to recover OFDM symbols transmitted over the air have been provided. Obviously the overall design could be enhanced by utilizing channel code and scramblers to reduce received BER, but the basic building blocks are provided. In review a flowchart of the receiver process is provided in Figure 10.12, which provides the basic comments on how the preamble and pilots are used. The goal is the maximally utilize the preamble symbols to provide the majority of the synchronization corrections, which differs in many ways from the single carrier implementations that heavily rely on just the modulation scheme itself.

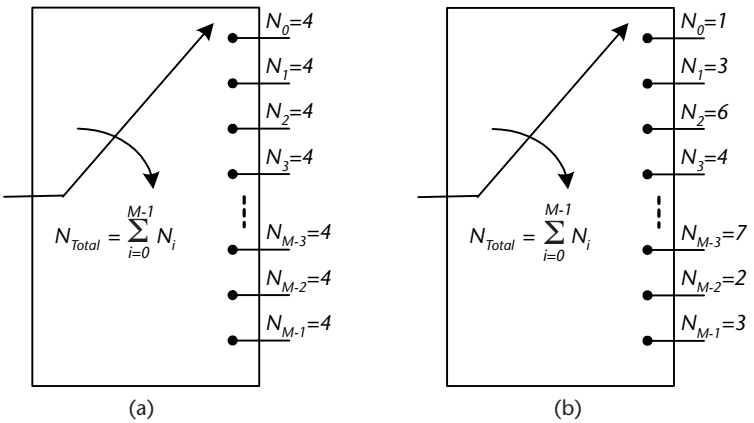

Figure 10.10 Comparison of (a) constant and (b) variable rate commutators with equivalent total rate.

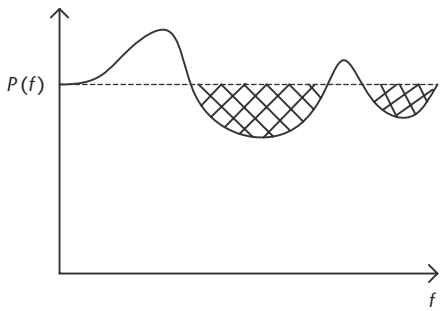

Figure 10.11 Illustration of the water-filling principle. Notice how power is allocated to each subcarrier, such that the resulting power would be a constant K.

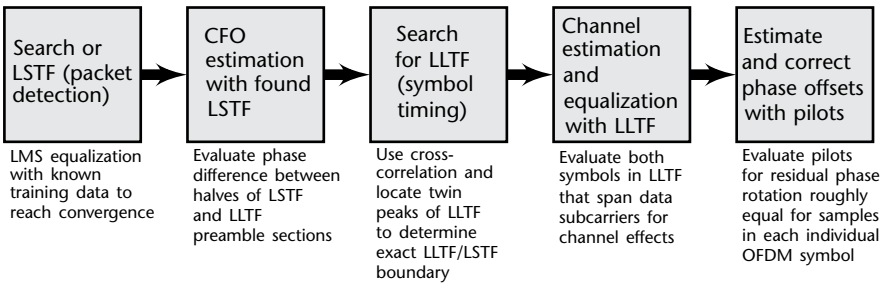

Figure 10.12 System overview for OFDM receiver for full frame recovery.

10.10 Chapter Summary

This chapter focused on the principle and implementation of multicarrier modulation, based on the 802.11a WLAN standard. Details were provided on the benefits of OFDM, including the simplistic receiver design. OFDM is the modern mechanism for high-speed transmission and reception, ranging from wired internet to even some satellite links. Although the implementation of the receiver is basically reversed from the previous succession of chapters, it can be considered much simpler than our single-carrier implementations.

References

[1] Cioffi, J. M., Chapter 34, "Asymmetric Digital Subscriber Lines," in *Communications Handbook*, CRC Press in Cooperation with IEEE Press, 1997.

[2] Golden, P., H. Dedieu, and K. Jacobsen, *Fundamentals of DSL Technology*, Boca Raton, FL: Auerbach Publications, 2004.

[3] Golden, P., and H. Dedieu, and K. Jacobsen, *Implementation and Applications of DSL Technology*, Boca Raton, FL: Auerbach Publications, 2007.

[4] Hanzo, L. L., Y. Akhtman, L. Wang, and M. Jiang, *MIMO-OFDM for LTE, WiFi and WiMAX: Coherent versus Non-coherent and Cooperative Turbo Transceivers*, Chichester, UK: Wiley-IEEE Press, 2010.

[5] Lee, B. G., and S. Choi, *Broadband Wireless Access and Local Networks: Mobile WiMaxand WiFi*, Norwood, MA: Artech House, 2008.

[6] Pareek, D., *WiMAX: Taking Wireless to the Max*, Auerbach Publications, 2006.

[7] Nuaymi, L., WiMAX: *Technology for Broadband Wireless Access*, Chichester, UK: Wiley, 2007.

[8] Sesia, S., I. Toufik, and M. Baker, *LTE—The UMTS Long Term Evolution: From Theory to Practice*, Second Edition, Chichester, UK: Wiley, 2011.

[9] Cox, C., *An Introduction to LTE: LTE, LTE-Advanced, SAE and 4G Mobile Communications*, Second Edition, Chichester, UK: Wiley, 2012.

[10] Digital Video Broadcasting (DVB), Implementation Guidelines for a Second Generation Digital Terrestrial Television Broadcasting System (DVB-T2), ETSI TS 102 831 V1.2.1, ETSI, 2012.

[11] IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems, Local and Metropolitan Area Networks–Specific Requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications–Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz, IEEE Std 802.11ac-2013 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012, IEEE Std802.11aa-2012, and IEEE Std 802.11ad-2012), December 2013, pp. 1–425.

[12] Schmidl, T. M., and D. C. Cox, "Robust Frequency and Timing Synchronization for OFDM," *IEEE Transactions on Communications*, Vol. 45, No. 12, December 1997, pp. 1613–1621.

[13] Minn, H., M. Zeng, and V. K. Bhargava, "On Timing Offset Estimation for OFDM Systems," *IEEE Communications Letters*, Vol. 4, No. 7, 2000, pp. 242–244.

[14] Kang, K. W., J. Ann, and H. S. Lee, "Decision-Directed Maximum-Likelihood Estimation of OFDM Frame Synchronisation Offset," *IEEE Electronics Letters*, Vol. 30, No. 25, December 1994, pp. 2153–2154.

[15] Cisco Inc., "Configuring Receiver Start of Packet Detection Threshold', Cisci Wireless Controller Guide, Release 8, April 2015, pp. 517–523

[16] van Zelst, A., and T. C. W. Schenk, "Implementation of a MIMO OFDM-Based Wireless LAN System," *IEEE Transactions on Signal Processing*, Vol. 52, No. 2, February 2004, pp. 483–494.