

Chapter 17 Multi-Carrier Modulation

Contents

- Slide 17-1 Brief History of Multi-Carrier Modulation
- Slide 17-3 A Simple OFDM Transmitter
- Slide 17-5 Interleaving
- Slide 17-6 Mapping Interleaver Blocks to Carriers
- Slide 17-9 Generating the Transmit Signal by an IFFT
- Slide 17-11 The Cyclic Prefix
- Slide 17-15 A Basic OFDM Receiver
- Slide 17-17 Demodulation and Equalization
- Slide 17-18 Optimum Equalizer Coefficients
- Slide 17-21 Clock Acquisition and Tracking
- Slide 17-22 Effect of Sample Clock Offset
- Slide 17-25 Estimating Sample Clock Period
- Slide 17-28 Tracking the Sample Clock
- Slide 17-32 Tracking with a 2nd Order PLL
- Slide 17-35 Symbol Alignment
- Slide 17-39 OFDM Experiments
- Slide 17-41 64-Point IFFT and FFT
- Slide 17-43 Simulating a Data Bit Stream

Slide 17-43 Constellation Mappers and Inverses
Slide 17-46 The Impulse Response Duration
Slide 17-48 Completing the Transmitter
Slide 17-54 Making the Receiver
Slide 17-64 Optional Additional Experiments

Chapter 17

Multi-Carrier Modulation

This chapter explores the fundamental concepts of multi-carrier modulation. It is used extensively in current wireless systems. These systems are very complex and this chapter will only look at the fundamental concepts used for modulation and demodulation by implementing a simple transmitter and receiver. The systems also include sophisticated error correcting codes, interleavers, and network protocols.

A Brief Overview of the History and Implementation of Multi-Carrier Modulation

Multi-carrier modulation is often selected for situations where the channel frequency response and noise interference vary significantly with frequency and/or time. Applications include:

- Broadband wireline digital subscriber lines. This application is essentially obsolete since much higher speed cable and fiber optic internet systems have become common. It has been called *discrete multi-tone modulation* (DMT) in the wireline community.
- Fading wireless channels where it is called *orthogonal frequency division multiplexing* (OFDM) in the wireless community. These include:
 - HF digital radio communication over long distances like from ship to shore
 - Wi-Fi networks using the IEEE standards 802.11a, 802.11g, and 802.11n. See the complete Chapter 17 for more details.
 - The 4G LTE cell phone system. Here it is called OFDMA for orthogonal frequency division multiple access.
 - The 5G NR cell phone system.

Multi-path Fading in Wireless Systems

Transmitted wireless radio signals usually take multiple paths from the transmitter to the receiver as a result of reflections from buildings and other objects. This causes signal fading at the receiver as the reflected signals add or subtract from each other. The fading is usually slowly time varying as the transmitter, receiver, and/or reflectors move. At any fixed time, the amplitude and phase responses of the channel can vary significantly over the channel bandwidth.

Description of a Simple OFDM Transmitter

In multi-carrier modulation the allotted frequency band is partitioned into multiple non-overlapping narrow-band subchannels. The frequency response over a subchannel is assumed to be essentially constant and equal to some complex number giving the amplitude and phase shift of the subchannel. A sinusoidal carrier is transmitted at the center of each subchannel.

The block diagram of a simple OFDM transmitter is shown in Figure 1. Successive blocks of many data bits are collected from one or more data sources.

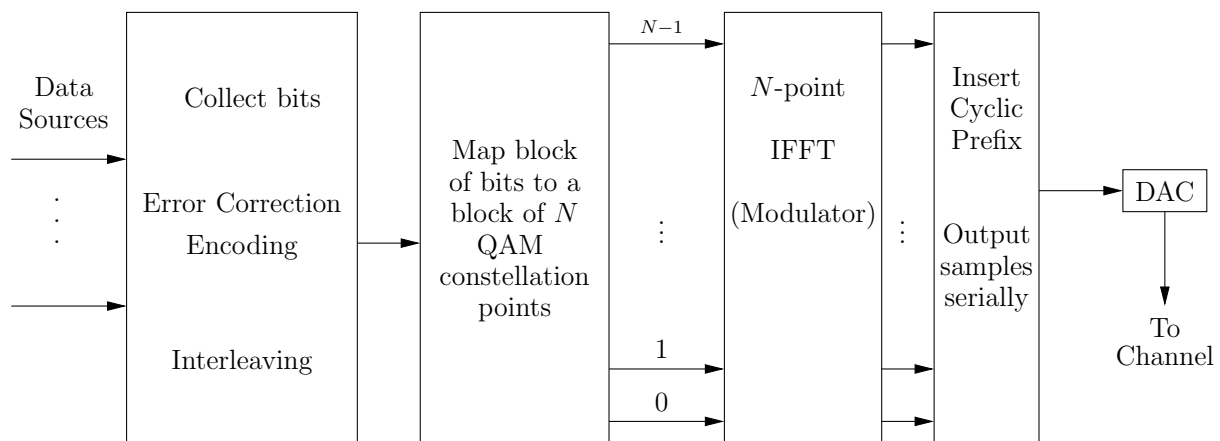


Figure 1: Basic OFDM Modulator

Error Correction Encoding

Additional redundant bits are added to each block by error correction encoding. A variety of error correcting codes are used in various standards. Turbo and LDPC codes are used in cell phone systems.

Interleaving

The time duration of a data block is typically long enough so that the channel can go through one or more fades during the input bit collection. A deep fade would cause a burst of errors at the output of the receiver for any type of modulation used to transmit the data block. The effects of fading are overcome by *interleaving*.

The error correcting codes used in most current wireless systems accept a block of input bits and append some check bits to form a code word of, say, M bits. The codes used are designed to correct a certain number of bit errors randomly distributed in the codeword and are not decoded well when the errors are clumped together in a burst. They also are not decoded correctly if the number of bit errors exceeds the design limit.

An interleaver separates the bits in each codeword in time by shuffling multiple codewords together. There are a number of types of interleavers. The simplest takes in L codewords and puts each codeword in an $L \times M$ rectangular array. L is called the interleaving depth. Then bits are read out of the array to the modulator successively going down the columns. A burst of B bit errors causes about B/L individual codeword errors after de-interleaving at the receiver.

Note: The complete Multi-Carrier chapter describes the convolutional interleaver used in the DSL wireline system.

Mapping Interleaver Blocks to Carriers

Blocks of bits are collected from the interleaver. Each block is partitioned into $N/2$ sub-blocks where N is a power of 2. Suppose each sub-block contains n_i bits for $i = 0, 1, \dots, N/2 - 1$ with $\sum_{i=0}^{N/2-1} n_i = N$.

Each sub-block selects a complex number

$$Z_i = x_i + jy_i = A_i e^{j\theta_i}$$

From a set of 2^{n_i} points called a *constellation*.

Figure 2 shows two simple constellations often used. They are sometimes called 4-QAM and 16-QAM. The 4-point constellation is also called QPSK for *quadrature phase shift keying*. The number at a constellation point is the decimal value of the binary n_i -tuple selecting that point.

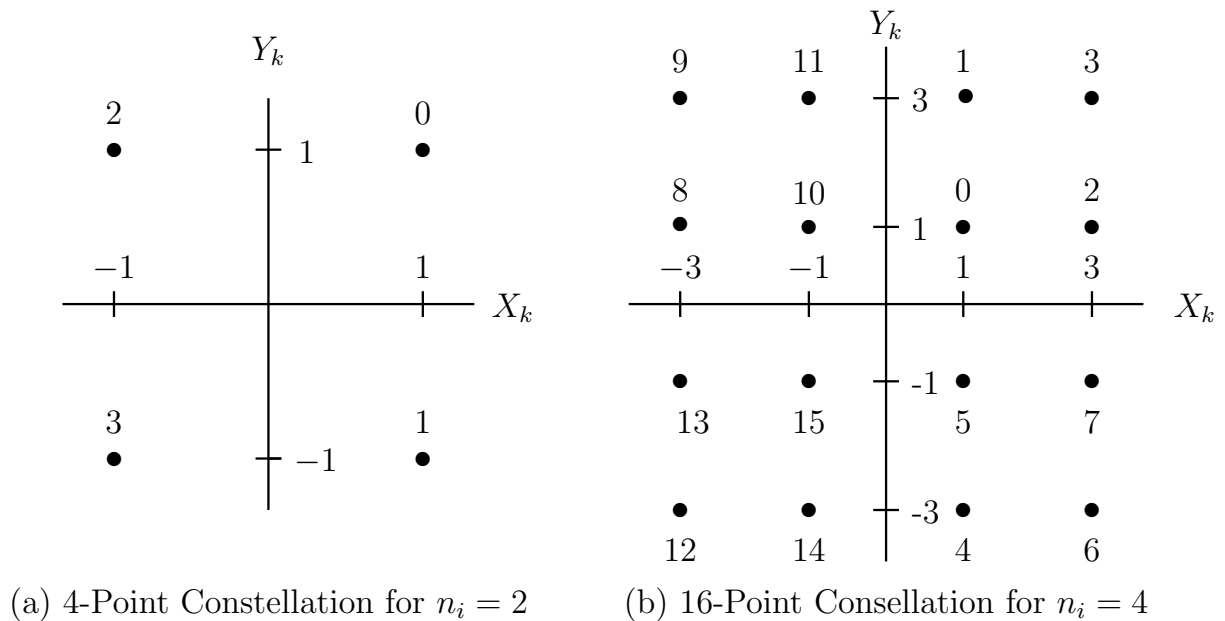


Figure 2: Constellations with 4 and 16 Points

The constellation point for sub-block i is used to scale the amplitude of a sinusoidal carrier at frequency ω_i by amplitude A_i and shift its phase by θ_i . This is called *quadrature amplitude modulation* (QAM). The carrier frequencies are chosen so that the carriers are orthogonal over their duration.

The number of bits n_i assigned to each carrier may differ and is chosen based on channel measurements such as signal-to-noise ratio at the carrier frequency.

Suppose the bit rate from the interleaver is $R_b = 1/T_b$ bits/sec and that M bits are collected for a block. Then the block and transmitted symbol duration is $T = MT_b$ and the symbol rate is $R_s = 1/T = 1/(MT_b)$ which is generally much smaller than the input bit rate.

Generating the Transmit Signal by an IDFT

The transmitted time signal is generated by an N -point inverse discrete Fourier transform (IDFT) which is efficiently computed by the inverse fast Fourier transform (IFFT). First the complex conjugates of the $N/2$ constellation points are appended to the constellation point vector to form an N -point vector as follows:

$$Z_0 = Z_{N/2} = 0 \quad (1)$$

$$Z_{N-k} = \overline{Z_k} \text{ for } k = 1, \dots, N/2 - 1 \quad (2)$$

This makes the N -point time sequence $z_n = \text{IDFT}\{Z_k\}$ real.

Z_0 is the complex amplitude of the zero frequency carrier which cannot carry a complex number.

$Z_{N/2}$ is the the complex amplitude of a carrier at half the sampling rate which alternates between 1 and -1 and cannot carry a complex number.

Therefore they are not used.

The time sequence computed by the IFFT, is

$$\begin{aligned}
 z_n &= \frac{1}{N} \sum_{k=0}^{N-1} Z_k e^{j \frac{2\pi}{N} k n} \\
 &= \frac{1}{N} \sum_{k=0}^{N-1} Z_k e^{j 2\pi \left(k \frac{f_1}{N}\right) n T_1}; n = 0, \dots, N-1 \quad (3)
 \end{aligned}$$

where $f_1 = 1/T_1$ is the sampling frequency.

Using the complex conjugate symmetry property of the Z_k sequence and it follows that

$$\begin{aligned}
 z_n &= \frac{1}{N} \sum_{k=1}^{\frac{N}{2}-1} \left[Z_k e^{j \frac{2\pi}{N} k n} + Z_{N-k} e^{j \frac{2\pi}{N} (N-k) n} \right] \\
 &= \frac{1}{N} \sum_{k=1}^{\frac{N}{2}-1} A_k \left[e^{j \left(\frac{2\pi}{N} k n + \theta_k \right)} + e^{-j \left(\frac{2\pi}{N} k n + \theta_k \right)} \right] \\
 &= \frac{2}{N} \sum_{k=1}^{\frac{N}{2}-1} A_k \cos \left(\frac{2\pi}{N} k n + \theta_k \right) \\
 &= \frac{2}{N} \sum_{k=1}^{\frac{N}{2}-1} A_k \cos \left[2\pi \left(k \frac{f_1}{N} \right) n T_1 + \theta_k \right] \quad (4)
 \end{aligned}$$

Therefore, the time sequence z_n for an OFDM symbol duration is the sum of cosines at the frequencies $f^{(k)} = kf_1/N$ for $k = 1, \dots, N/2 - 1$ sampled with period T_1 .

The Cyclic Prefix

The impulse response of a wireline channel lasts for many samples. It can also last for many samples for a wireless channel as a result of multi-path propagation. A wireless channel can change relatively quickly with time as a result of motion of the transmitter, receiver, and reflecting objects. However, OFDM modulation and linear time-invariant methods can be used if the channel changes insignificantly over the duration of a symbol and channel estimates are continually updated. The channel impulse response causes intersymbol interference (ISI) between adjacent OFDM symbols.

To characterize the intersymbol interference, suppose the samples for a symbol starting at time 0 and ending at time M are $s_0, s_1, \dots, s_M$.

Suppose the impulse response, $h(n)$, can be nonzero only for $n = 0, 1, \dots, L$ and that the symbol duration is longer than the impulse response duration, that is, $M > L$.

The ISI from this first symbol into the next is

$$I(n) = \sum_{k=n-L}^M s_k h(n-k) \text{ for } n = M+1, \dots, M+L \quad (5)$$

as $h(n-k)$ slides off the right side of the the symbol sample sequence.

The ISI problem has been solved by increasing the symbol duration to $N' = N + \nu$ samples by prepending a cleverly chosen sequence of ν samples which is longer than the impulse response duration. This provides a guard time between symbols for the ISI to decay to zero during the beginning of the second symbol.

The prepended samples are the last ν samples of the symbol, that is, $(z_{N-\nu}, \dots, z_{N-2}, z_{N-1})$, so the augmented symbol sequence is

$$(s_{-\nu}, \dots, s_{-1}, s_0, \dots, s_{N-1}) = (z_{N-\nu}, \dots, z_{N-2}, z_{N-1}, z_0, z_1, \dots, z_{N-1}).$$

Let the time origin $n=0$ be chosen at $s_0 = z_0$.

Notice that

$$\text{mod}(n, N) = \begin{cases} N + n & \text{for } n = -\nu, \dots, -1 \\ n & \text{for } n = 0, \dots, N - 1 \end{cases}$$

Thus the augmented symbol sequence can be written as

$$s_n = z_{\text{mod}(n, N)} \text{ for } n = -\nu, \dots, N - 1$$

Then the channel output for $0 \leq n \leq N - 1$, ignoring noise, is

$$r(n) = \sum_{k=n-L}^n s_k h(n-k) = \sum_{k=n-L}^n z_{\text{mod}(k, N)} h(n-k) \quad (6)$$

This convolution can also be expressed as

$$r(n) = \sum_{k=0}^N z_k h_{\text{mod}(n-k, N)} \quad (7)$$

where

$$h_n = \begin{cases} h(n) & \text{for } 0 \leq n \leq L \\ 0 & \text{for } L + 1 \leq n \leq N - 1 \end{cases}$$

This formula can be thought of as the convolution of $h(n)$ with a periodically repeated version of the N symbol samples or the *circular* or *cyclic* convolution of the N -point sequences z_n and h_n .

The DFT of the cyclic convolution of two N -point signals is the product of their DFT's, that is

$$R_k = \text{DFT}\{r_n\} = \text{DFT}\{z_n\} \text{DFT}\{h_n\} = Z_k H_k \quad (8)$$

The $N' = N + \nu$ points of the augmented sequence are sent to the DAC starting with the cyclic prefix.

A Basic OFDM Receiver

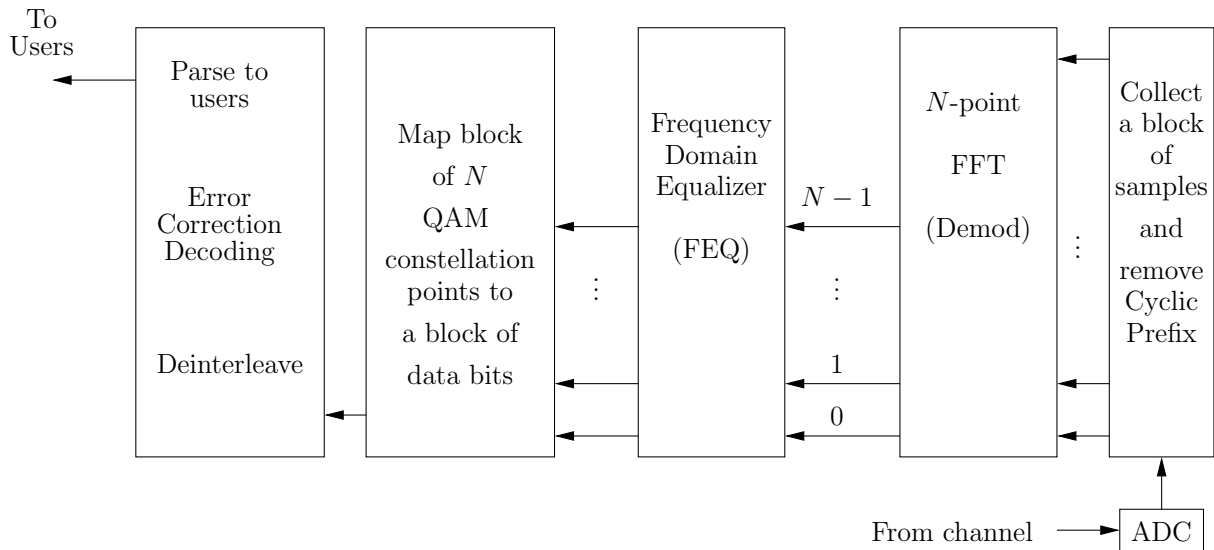


Figure 3: Basic OFDM Demodulator

The block diagram for a basic OFDM receiver is shown in Figure 3. Samples are collected from the receiver's front end and one of the first steps is to synchronize to the OFDM symbol timing. Then for each block of symbol samples the cyclic prefix is deleted leaving N samples. An N -point FFT is computed and the result is passed to the Frequency Domain Equalizer (FEQ).

As the transmitted symbol passes through the channel, the QAM constellations for the subcarriers are rotated and changed in amplitude by the complex numbers corresponding to the channel frequency responses at the subcarrier frequencies. The constellations are corrected by multiplying essentially by the reciprocals of the subchannel response complex numbers. This assumes that these numbers are known and OFDM systems periodically probe the channel to measure its frequency response.

The equalized constellations for each carrier are quantized to the nearest ideal constellation points which are then mapped back to the corresponding bits. The resulting bit stream is deinterleaved and passed to the error correction decoder. The decoded bits are finally parsed and passed to the appropriate users.

Demodulation and Frequency Domain Equalization

The samples for an OFDM symbol are collected and a sequence of N consecutive samples are extracted by removing the cyclic prefix. Let the resulting sequence for symbol i be $r_{n,i}$ for $n = 0, \dots, N - 1$.

The FFT demodulator output is

$$\begin{aligned} R_{k,i} &= \sum_{n=0}^{N-1} r_{n,i} e^{-j \frac{2\pi}{N} nk} \\ &= Z_{k,i} H_k + V_{k,i}; \quad k = 0, \dots, N - 1 \end{aligned} \quad (9)$$

where H_k is the channel frequency response at subcarrier k , $Z_{k,i}$ is the transmitted constellation point, and $V_{k,i}$ is a zero-mean noise sample with variance $\sigma_k^2 = E\{|V_{k,i}|^2\}$. The constellation power is $S_k = E\{|Z_{k,i}|^2\}$

The effect of H_k is to scale and rotate the received constellation. Only the lower half of the points are needed because of the complex conjugate symmetry of the transform of a real sequence.

Channel Estimation

The channel frequency response can be estimated during initialization when a known sequence with a spectrum that covers the signal band is transmitted for many symbols. One method for estimating H_k is to average $R_{k,i}/Z_{k,i}$ over a number of symbols. This assumes that the sample clock has already been adequately acquired and drifts negligibly between symbols.

The initial channel frequency response estimates can be adaptively updated during regular data transmission to refine the initial estimates, track slow changes in the channel, and compensate for small sample clock drifts.

Optimum Equalizer Values The LMS algorithm with a one-tap equalizer can be used to find the best equalizer coefficient for each subcarrier. Let the equalizer coefficient for subcarrier k be W_k , the equalized received point be $\tilde{Z}_{k,i} = W_k R_{k,i}$, and the estimation error be $\Delta_{k,i} = Z_{k,i} - W_k R_{k,i}$.

Under noisy conditions, W_k can be chosen to minimize

$$\Lambda_k = \text{E} \left\{ |\Delta_{k,i}|^2 \right\} = \text{E} \left\{ (Z_{k,i} - W_k R_{k,i}) \overline{(Z_{k,i} - W_k R_{k,i})} \right\} \quad (10)$$

If H_k is known, the closed form solution is

$$\hat{W}_k = \frac{S_k}{S_k |H_k|^2 + \sigma_k^2} \overline{H}_k \quad (11)$$

and the corresponding mean-squared error is

$$\hat{\Lambda}_k = \sigma_k^2 \frac{S_k}{S_k |H_k|^2 + \sigma_k^2} \quad (12)$$

Notice that if there is no noise so that $\sigma_k^2 = 0$, $\hat{W}_k = 1/H_k$

Iterative Coefficient Computation

Since H_k is not actually known, the iterative LMS algorithm can be used. Let A_k and B_k be the real and imaginary parts of W_k so that

$W_k = A_k + jB_k$. Taking derivatives with respect to A_k and B_k gives

$$\frac{\partial \Lambda_k}{\partial A_k} = -2 \text{E} \{ \Re \{ \Delta_{k,i} \overline{R}_{k,i} \} \} \quad (13)$$

and

$$\frac{\partial \Lambda_k}{\partial B_k} = -2 \text{E}\{\Im m\{\Delta_{k,i} \bar{R}_{k,i}\}\} \quad (14)$$

so

$$\frac{\partial \Lambda_k}{\partial W_k} \triangleq \frac{\partial \Lambda_k}{\partial A_k} + j \frac{\partial \Lambda_k}{\partial B_k} = -2 \text{E}\{\Delta_{k,i} \bar{R}_{k,i}\} \quad (15)$$

This suggests the approximate update formula

$$\begin{aligned} W_k(i+1) &= W_k(i) + \mu \Delta_{k,i} \bar{R}_{k,i} \\ &= W_k(i) + \mu [Z_{k,i} - W_k(i) R_{k,i}] \bar{R}_{k,i} \end{aligned} \quad (16)$$

where μ is a small positive constant.

It is known that in order for the mean-squared error to converge, μ must satisfy the stability constraint

$$0 < \mu < 2/\text{E}\{|R_{k,i}|^2\} = 2/(S_k |H_k|^2 + \sigma_k^2) \quad (17)$$

Since the product $\Delta_{k,i} \bar{R}_{k,i}$ is a rough estimate of its expected value, the coefficient estimate converges to the correct mean value with excess fluctuations. Making μ small reduces the excess variance but slows the convergence time.

Decision Directed Adaptation

The true value of $Z_{k,i}$ is not known at the receiver. However, after initial channel estimation, the equalizer output $\tilde{Z}_{k,i} = W_k R_{k,i}$ will be close to the true value. Then an accurate estimate of $Z_{k,i}$ can be made by quantizing $\tilde{Z}_{k,i}$ to the closest ideal constellation point. The quantization procedure results in very accurate decisions during initialization symbol sequences when the 4-point QPSK constellation is used. This “decision directed” approach works very well in practice even for constellations of large size after the equalizer coefficient has converged and μ is made small.

Sample Clock Acquisition and Tracking

The sample clocks in a transmitter and remote receiver will have some frequency offset. This offset will be small because the standards place tight tolerances on the clock frequency. Nevertheless, a small frequency error will

accumulate to a large phase error over time leading to significant degradation of performance.

The receiver must lock its clock to the transmitter clock based on timing information derived from its received signal. The offset can be estimated in an open loop manner during initialization when known signals are transmitted, and updated in a closed loop tracking mode during normal data transmission.

Effect of Sample Clock Offset on the Output of the Receiver's FFT

Let $T_1 = 1/f_1$ be the sampling period of the transmitter. Consider the symbol with samples, z_n , given by (3) starting at time $n = 0$ with a ν sample cyclic prefix longer than the channel impulse response prepended for time $-\nu, -\nu + 1, \dots, -1$. When this $N + \nu$ point sequence is applied to the DAC, the resulting

continuous-time signal is

$$z(t) = \frac{1}{N} \sum_{\ell=0}^{N-1} Z_{\ell} e^{j2\pi\ell \frac{f_1}{N} t} \quad (18)$$

Let the sampling period at the receiver be $T_2 = (1 + \epsilon)T_1$ where ϵ is a small constant. Then, the samples observed at the receiver are

$$\begin{aligned} r_{n,0} &= \frac{1}{N} \sum_{\ell=0}^{N-1} H_{\ell} Z_{\ell} e^{j2\pi\ell \frac{f_1}{N} n(1+\epsilon)T_1} \\ &= \frac{1}{N} \sum_{\ell=0}^{N-1} H_{\ell} Z_{\ell} e^{j \frac{2\pi}{N} (1+\epsilon)n\ell} \end{aligned} \quad (19)$$

where H_k is the channel frequency response at subcarrier k . The receiver computes the DFT of this sequence to get

$$\begin{aligned} R_{k,0} &= \sum_{n=0}^{N-1} r_{n,0} e^{-j \frac{2\pi}{N} nk} \\ &= \sum_{n=0}^{N-1} \frac{1}{N} \sum_{\ell=0}^{N-1} H_{\ell} Z_{\ell} e^{j \frac{2\pi}{N} (1+\epsilon)n\ell} e^{-j \frac{2\pi}{N} nk} \end{aligned}$$

$$\begin{aligned}
&= \sum_{\ell=0}^{N-1} H_{\ell} Z_{\ell} \frac{1}{N} \sum_{n=0}^{N-1} e^{j \frac{2\pi}{N} n[(1+\epsilon)\ell-k]} \\
&= \sum_{\ell=0}^{N-1} H_{\ell} Z_{\ell} Q(k, \ell, \epsilon)
\end{aligned} \tag{20}$$

where

$$\begin{aligned}
Q(k, \ell, \epsilon) &= \frac{1}{N} \sum_{n=0}^{N-1} e^{j \frac{2\pi}{N} n[(1+\epsilon)\ell-k]} \\
&= \frac{1}{N} \frac{e^{j2\pi[(1+\epsilon)\ell-k]} - 1}{e^{j \frac{2\pi}{N} [(1+\epsilon)\ell-k]} - 1}
\end{aligned} \tag{21}$$

By factoring out e to half the exponent in the numerator and denominator, this can be put into the following more informative form:

$$Q(k, \ell, \epsilon) = e^{j \frac{2\pi}{N} [(1+\epsilon)\ell-k] \frac{N-1}{2}} \frac{\sin \pi[(1+\epsilon)\ell-k]}{N \sin \frac{\pi}{N} [(1+\epsilon)\ell-k]} \tag{22}$$

Equation (20) shows that an effect of timing offset is to introduce crosstalk from one subchannel into another.

The component proportional to Z_k in $R_{k,0}$ is

$$H_k Z_k Q(k, k, \epsilon) = H_k Z_k e^{j \frac{2\pi}{N} k \epsilon \frac{N-1}{2}} \frac{\sin \pi k \epsilon}{N \sin \frac{\pi}{N} k \epsilon} \quad (23)$$

The timing frequency offset causes a phase rotation of $H_k Z_k$ by the angle $\beta_k = k \frac{2\pi}{N} \epsilon \frac{N-1}{2}$ that increases linearly with frequency index and scales its amplitude. The amplitude scaling is negligible for small $|\epsilon|$ and computation shows that the crosstalk components are small when the timing offset is within the tolerances required by the standards. Therefore,

$$R_{k,0} \approx H_k Z_k e^{j \frac{2\pi}{N} k \epsilon \frac{N-1}{2}} \quad \text{for } |\epsilon| \ll 1 \quad (24)$$

Open Loop Estimation of the Sample Clock Period Error

Now suppose the symbol including the cyclic prefix is repeated starting at time N . The transmitted signal is

$$z_1(t) = z[t - (N + \nu)T_1] \quad \text{for } NT_1 \leq t \leq (2N + \nu)T_1 \quad (25)$$

The cyclic prefix for this repeated symbol extends over $NT_1 \leq t \leq (N + \nu)T_1$. Skipping over what it thinks is the received cyclic prefix, the receiver collects the N -point sequence

$$\begin{aligned}
r_{n,1} &= z_1[(n + N + \nu)T_2] \\
&= z_1[(n + N + \nu)(1 + \epsilon)T_1] \\
&= z[n(1 + \epsilon)T_1 + (N + \nu)\epsilon T_1] \\
&= \frac{1}{N} \sum_{\ell=0}^{N-1} H_\ell Z_\ell e^{j \frac{2\pi}{N} \ell \epsilon (N + \nu)} e^{j \frac{2\pi}{N} (1 + \epsilon) n \ell} \\
&\quad \text{for } n = 0, \dots, N - 1
\end{aligned} \tag{26}$$

and computes the DFT

$$R_{k,1} = \sum_{\ell=0}^{N-1} H_\ell Z_\ell e^{j \frac{2\pi}{N} \ell \epsilon (N + \nu)} Q(k, \ell, \epsilon) \tag{27}$$

Again, the subcarrier crosstalk is small if $|\epsilon| \ll 1$ and

$$R_{k,1} \approx H_k Z_k e^{j \frac{2\pi}{N} k \epsilon \frac{N-1}{2}} e^{j \frac{2\pi}{N} k \epsilon (N + \nu)} \tag{28}$$

At this point in the initialization procedure, the channel frequency response H_k may not be known yet. The phase effects of this response and the constellation point Z_k can be removed by forming the product

$$\Gamma_{k,1} = R_{k,1} \overline{R}_{k,0} = |H_k Z_k|^2 e^{j \frac{2\pi}{N} k \epsilon (N + \nu)} \quad (29)$$

which has the angle

$$\varphi_{k,1} = \arg \Gamma_{k,1} = \frac{2\pi}{N} k \epsilon (N + \nu) \quad (30)$$

The preceding derivations have neglected the effects of noise. An accurate estimate of ϵ can be obtained when noise is present by averaging $\Gamma_{k,i} = R_{k,i} \overline{R}_{k,i-1}$ over several symbols for a range of k , computing phase estimates $\hat{\varphi}_k$ from the averages, and then choosing ϵ to form a least-squares fit of a straight line to the phase estimates over the range of k .

Suppose the range is $k_1 \leq k \leq k_2$. Then the least-squares fit estimate is

$$\hat{\epsilon} = \frac{N}{2\pi(N + \nu)} \left(\sum_{k=k_1}^{k_2} k^2 \right)^{-1} \sum_{k=k_1}^{k_2} k \hat{\varphi}_k \quad (31)$$

The sum of squares can be computed by using the formula

$$\sum_{n=1}^N k^2 = \frac{N(N + 1)(2N + 1)}{6} \quad (32)$$

Tracking the Sample Clock Period

Once training is finished, random OFDM symbols generated by customer data are transmitted and the open loop method of estimating the timing period error just presented cannot be used. At the end of training, the frequency domain equalizer (FEQ) should be well adjusted and a good estimate of ϵ should have been made and inserted into the clock generator.

The constellations observed for each subcarrier at the output of the receiver's FFT should be very close to the ideal transmitted constellations. If there is a small residual clock period error, the constellations will appear to slowly rotate if the FEQ is kept constant as shown by (30).

These observations suggests that a decision-directed method can be used for tracking the sample clock. Let

$$\hat{Z}_{k,i} = |\hat{Z}_{k,i}|e^{j\alpha_{k,i}}$$

be the ideal constellation point closest to the equalized point

$$\tilde{Z}_{k,i} = |\tilde{Z}_{k,i}|e^{j\psi_{k,i}}$$

for the symbol at time i . The angle between the equalized and estimated ideal constellation points can be determined as

$$\begin{aligned}
\zeta_{k,i} &= \arg(\tilde{Z}_{k,i} \overline{\hat{Z}_{k,i}}) \\
&= \arg\left(|\tilde{Z}_{k,i}| |\hat{Z}_{k,i}| e^{j(\psi_{k,i} - \alpha_{k,i})}\right) \\
&= \psi_{k,i} - \alpha_{k,i}
\end{aligned} \tag{33}$$

In rectangular form

$$\tilde{Z}_{k,i} \overline{\hat{Z}_{k,i}} = |\tilde{Z}_{k,i}| |\hat{Z}_{k,i}| \cos \zeta_{k,i} + j |\tilde{Z}_{k,i}| |\hat{Z}_{k,i}| \sin \zeta_{k,i} \tag{34}$$

The phase rotation from one symbol to the next will be small when the sample clock is being tracked well. Using the approximation $\sin x \approx x$ for $|x| \ll 1$ and the fact that $|\tilde{Z}_{k,i}| \approx |\hat{Z}_{k,i}|$ when the FEQ is well adjusted, gives

$$\begin{aligned}
\Im\{\tilde{Z}_{k,i} \overline{\hat{Z}_{k,i}}\} &= |\tilde{Z}_{k,i}| |\hat{Z}_{k,i}| \sin \zeta_{k,i} \\
&\approx |\tilde{Z}_{k,i}| |\hat{Z}_{k,i}| \zeta_{k,i} \approx |\hat{Z}_{k,i}|^2 \zeta_{k,i}
\end{aligned} \tag{35}$$

which is proportional to the phase rotation.

Computing $\Im m\{\tilde{Z}_{k,i}\overline{\hat{Z}_{k,i}}\}$ from the rectangular forms requires the difference of two real products and is an efficient method for implementation with DSP's.

According to (30) $\zeta_{k,i}$ should be proportional to the frequency index k and ϵ if noise is ignored. A straight line can be fit to the phase rotations at different subcarriers to give the following formula similar to (31) for estimating the residual timing offset to within a constant:

$$\zeta_i = C \sum_{k \text{ used}} k \zeta_{k,i} \quad (36)$$

where C is an appropriate scale factor.

The phase rotation estimate can be used to iteratively update the sampling period.

- If $\zeta_i > 0$, the constellation has rotated counter clockwise, and the sampling period is too large and should be reduced.
- If $\zeta_i < 0$ the sampling period is too small and should be increased.

Suppose the FEQ is perfectly adjusted at symbol time i and is held constant after that. At symbol time i there will be no constellation rotation.

According to (30), the constellation at symbol time $i + 1$ will be rotated by $\varphi_{k,1} = \frac{2\pi}{N}k\epsilon(N + \nu)$. This can be corrected by rotating the equalized point in the opposite direction by $\varphi_{k,1}$.

Mathematically, the counter-rotation can be accomplished by multiplying the equalized point $\tilde{Z}_{k,i+1}$ by $e^{-j\varphi_{k,1}}$. Each successive symbol the equalized point has to be counter-rotated by a larger angle. That is, at symbol $i + q$ the equalized point would have to be rotated by angle $-q\varphi_{k,1}$.

Tracking with a Second-Order Phase-Locked Loop

A second-order PLL can be implemented to generate the required rotation. Let the PLL output at symbol time i be τ_i which is an estimate of the fractional timing error ϵ .

The corresponding constellation point should be rotated by the angle $-\frac{2\pi}{N}k\tau_i(N + \nu)$.

First, the phase errors are accumulated using the equation

$$\delta_i = \delta_{i-1} + \beta\zeta_i \quad (37)$$

Then, the PLL output is computed as

$$\tau_i = \tau_{i-1} + \delta_i \quad (38)$$

where β is a small constant.

The idea is that when the phase rotation estimate ζ_i has a constant component, its accumulated value will converge in the closed loop to the constant required to increment τ_i each iteration to force the observed rotation error to zero. Adding a term $\alpha\zeta_i$ with α a small positive number to the equation for τ_i would allow the loop to eliminate a constant constellation tilt. The FEQ already corrects for any constant tilt, so it is not necessary to add this term. For this arrangement to work correctly, the FEQ must change its coefficients much more slowly than the timing PLL tracks.

Sample Stuffing and Robbing

The PLL output τ_i may grow linearly without bound unless some corrective action is taken. If no action is taken, the symbol timing can slip into the cyclic prefix or into an adjacent symbol. An approach is to keep $|\tau_i| \leq 0.5$ by stuffing or robbing a sample between symbols.

- When τ_i exceeds 0.5, this indicates that the equalized constellations are rotated in a counter-clockwise direction and the sampling period is too large. This can be corrected by advancing the timing by one sample. The advance can be implemented by “stuffing” an extra sample into the cyclic prefix to move the N points collected for the symbol forward in the sample buffer by one sample. Then τ_i should be replaced by $\tau_i - 1$.
- When τ_i becomes less than -0.5 , this indicates the sampling is occurring too early and a one sample delay should be introduced. This can be accomplished by deleting or

“robbing” a sample from the cyclic prefix to move the symbol collection time one sample later relative to the sample buffer and then τ_i should be replaced by $\tau_i + 1$.

In this scheme, τ_i will vary between -0.5 and 0.5 .

Symbol Alignment Acquisition and Tracking

Special training signals are sent during initialization to allow the receiver to set its AGC level, measure the channel frequency response, and acquire sample and symbol timing.

The receiver must determine where a symbol starts so it can delete the cyclic prefix and select a sequence of N ISI free samples for the FFT demodulator.

The receiver can search symbols for the repetition of the cyclic prefix N samples by some type of correlation process. For example, the receiver

could compute the periodicity metric

$$\begin{aligned}
p_{n,i} &= \sum_{\ell=n-d}^n (r_{\ell,i} - r_{\ell-N,i})^2 \\
&= \sum_{\ell=n-d}^n r_{\ell,i}^2 + \sum_{\ell=n-d}^n r_{\ell-N,i}^2 - 2 \sum_{\ell=n-d}^n r_{\ell,i} r_{\ell-N,i}
\end{aligned} \tag{39}$$

where d is somewhat less than the cyclic prefix length ν .

This could also be averaged over several symbols. The metric will be close to zero when n is at the end of the symbol and the two samples subtracted are theoretically identical and gets large elsewhere. The minimum of the metric indicates the location of the end of a symbol.

A test for the location of the minimum is to check when for some appropriately chosen small positive

constant γ

$$\begin{aligned} \sum_{\ell=n-d}^n r_{\ell,i}^2 + \sum_{\ell=n-d}^n r_{\ell-N,i}^2 - 2 \sum_{\ell=n-d}^n r_{\ell,i} r_{\ell-N,i} \\ < \gamma \left(\sum_{\ell=n-d}^n r_{\ell,i}^2 + \sum_{\ell=n-d}^n r_{\ell-N,i}^2 \right) \end{aligned} \quad (40)$$

or

$$\sum_{\ell=n-d}^n r_{\ell,i} r_{\ell-N,i} > \frac{1-\gamma}{2} \left(\sum_{\ell=n-d}^n r_{\ell,i}^2 + \sum_{\ell=n-d}^n r_{\ell-N,i}^2 \right) \quad (41)$$

In this last form, the search is for correlation peaks rather than periodicity metric nulls.

The computation could be simplified by replacing the right-hand side by a fixed threshold. However, the sum on the right-hand side is an estimate of the average power and provides an automatic threshold normalization.

Remaining Blocks

The remaining blocks in the receiver do the reverse of the corresponding blocks in the transmitter.

- The constellation points at the output of the FEQ are quantized and mapped back into a bit sequence.
- This sequence is deinterleaved, processed by the Reed-Solomon decoder.
- The decoder output stream is descrambled.
- The resulting output bits are parsed and sent to the users.

Making an OFDM Transmitter and Receiver

In this experiment you will make a simple OFDM transmitter and receiver.

Set the TMS320C6713 DSK sampling rate to $f_1 = 16$ kHz.

The transmitter should use an $N = 64$ point IFFT and the receiver a 64-point FFT. The subcarriers are at frequencies that are multiples of $16000/64 = 250$ Hz. The subcarriers for $k = 0$ and 32 should be set to zero and not used. The subcarriers for $k = 1, \dots, 31$ will be used to send QPSK (4-QAM) or 16-QAM constellations corresponding to 2 and 4 bits per constellation point, respectively.

The subcarrier points for $k > 32$ are determined by the constellation points for $k < 32$ by the complex conjugate symmetry rule

$$Z_{64-k} = \overline{Z_k} \quad \text{for } k = 1, \dots, 31 \quad (42)$$

which will force $z_n = \text{IFFT}\{Z_k\}$ to be real.

Use a cyclic prefix of $\nu = 4$ samples. You may have to lengthen this after measuring the channel impulse response. Then the length of regular data symbols will be $N' = 68$ samples.

The number of symbols transmitted per second will be $16000/68 = 235.294$. The minimum number of bits transmitted per symbol is $2 \times 31 = 62$ when QPSK is used for all the used subcarriers giving a data rate of $62 \times 16000/68 = 14588.235$ bits per second if no sync symbols are used. The maximum number of bits per symbol is $4 \times 31 = 124$ when 16-QAM is used for all the used subcarriers giving a data rate of $124 \times 16000/68 = 29176.47$ bits per second if no sync symbols are used. A sync symbol will be inserted after every 68 data symbols, so the data rates are scaled by $68/69$.

Carry out the steps in the following slides to make and test your transmitter and receiver.

Making a 64-Point IFFT and a 64-Point FFT

You will need a 64-point IFFT for your transmitter and a 64-point FFT for your receiver.

Modify the decimation-in-time FFT function from Chapter 4 to be a 64-point IFFT. This function should accept the 64-point constellation array in natural order, shuffle it into bit-reversed order, and then perform the equivalent of the “decimation-in-time” FFT algorithm to get the output time sequence in natural order.

Read required cosine and sine values for the “twiddle factors” from a pre-computed table rather than compute them recursively like in the FFT function from Chapter 4. Only a single sine table is needed, not both a sine and cosine table, because $\cos \theta = \sin(\theta + \pi/2)$.

You can achieve some computational savings in the last stage by using the fact that the output must be real and not computing the imaginary part.

Test your IFFT by putting in some sequences that have known outputs. For example, Let m be an integer with $0 \leq m \leq 31$. Then

$Z_k = \cos(2\pi mk/64)$ has the IFFT,

$$z_n = 0.5\delta[n - m] + 0.5\delta[n - (64 - m)].$$

Modify the decimation-in-time FFT function in Chapter 4 to be a 64-point FFT. Again, read the sine/cosine twiddle factors from a table. Some computational savings can be achieved by using the fact that the input is real in the first stage. Test your function by using some inputs with known FFT's.

Simulating a Data Bit Stream

Implement a 23-stage self synchronizing scrambler as described in Chapter 9 with input $x(n) = 1$ for all n to generate simulated customer random data. Use the connection polynomial $h(D) = 1 + D^{18} + D^{23}$. Let the input to the scrambler be $x(n)$. Then, the scrambler output is

$$y(n) = x(n) + y(n - 18) + y(n - 23) \quad (43)$$

where “+” is modulo two addition. The input can be computed from the output, that is, descrambled by computing

$$x(n) = y(n) + y(n - 18) + y(n - 23) \quad (44)$$

Making Constellation Point Mappers and Inverses

Initially you will test your transmitter with QPSK constellations for all of the 31 used subcarriers. In this mode, you should generate blocks of 62 bits with the scrambler.

Later you will use 16-QAM for all 31 used subcarriers which will require blocks of 124 bits.

Make a function that maps an array of 62 input bits into a 64 element complex array Z with $Z[0] = Z[32] = 0$, $Z[1], \dots, Z[31]$ QPSK constellation points $X_k + jY_k$ corresponding to successive pairs of input bits using the rule stated on slide 17-7 and illustrated in Figure 2, and $Z_{64-k} = \overline{Z_k}$ for $k = 1, \dots, 31$.

Make another function for 16-QAM that maps an array of 124 input bits into a 64 element complex array Z with $Z[0] = Z[32] = 0$, $Z[1], \dots, Z[31]$ 16-QAM constellation points $X_k + jY_k$ corresponding to successive groups of four input bits using the rule stated on page 17-7 and illustrated in Figure 2, and $Z_{64-k} = \overline{Z_k}$ for $k = 1, \dots, 31$.

Make the inverse functions. That is, make functions to convert a 64 element complex array back into an array of 62 bits for QPSK and 124 bits for 16-QAM. These functions should quantize a complex point $R[k]$ corresponding to an element of the output of the FFT in the receiver to the ideal constellation point shown in Figure 2 that is nearest to it in Euclidean distance, and put the two or four data bits corresponding to the quantized point into a 62 or 124 element array for $k = 1, \dots, 31$.

Implement a descrambler using (44). Test your inverse mappers and descrambler separately. Then test the cascade of the scrambler, mapper, inverse mapper, and descrambler for a string of transmitted OFDM symbols. Do this internally in a program for one DSP without worrying about getting data to and from the codec. The output of the descrambler should be all 1's except, possibly, for a short initial transient if the scrambler and descrambler initial states differ.

Once the cascade is working, insert the transmitter IFFT and receiver FFT. First leave the IFFT output as a floating-point array. Use this array as the FFT input. Do this internally in the single DSP program without worrying about getting data to and from the codec. Next convert the IFFT output array to an integer format scaled appropriately for the right codec channel and use it internally in the program as the receiver FFT input and check that the cascade works. You will have to structure the inverse mapper to account for the scaling.

Measuring the Channel Impulse Response Duration

The duration of the channel impulse response must be known to select the length of the cyclic prefix. Send the OFDM samples generated in the DSP for the transmitter to the right Line Out channel. Loop the Line Out back to the Line In with a cable.

Later in this chapter, you will be asked to connect the Line Out of one station to a Line In of another station. The channels for both of these cases should be very similar.

Write a program to first send 128 zeros to the right channel to clear any initial startup transients. Then periodically send pulses separated in time by more than what you expect the duration of the channel impulse to be to the right Line Out channel. For example, send

$$s[n] = \sum_{k=0}^{\infty} 20000 \delta[n - 64k] \quad (45)$$

The scale factor, 20000, was somewhat arbitrarily chosen to use a significant portion of the range of the ADC. This signal is a periodic train of unit pulses occurring every 64 samples with zeros in between.

Collect 256 samples from the looped back Line In in an array starting when you send the first pulse.

Examine the array using a watch window or more visually informatively by using the graphing capabilities of Code Composer and determine the duration of the channel impulse response. Choose a value, ν , for the cyclic prefix length that is somewhat larger than the duration of the channel impulse response.

Completing the Transmitter

Now it is time to put together the transmitter blocks along with an initialization sequence and operate in real-time.

Use ping-pong buffers to send output samples to the right channel Line Out when transmitting regular data symbols. Define two integer arrays, `ping[]` and `pong[]`, each of which can hold the output samples for one OFDM symbol. Samples should be sent to the output from one of the arrays using an interrupt service routine triggered by interrupts from the McBSP1 XRDY1 flag.

While the samples are going out of one array, your program should generate the samples for the next OFDM symbol and put them into the other array and then wait for all the samples to be transmitted from the first array. Then the functions of the two arrays should be swapped with samples transmitted from the array just filled and samples for a new OFDM symbol generated and put into the other array.

As an initial test of your ping-pong buffer scheme, select a subcarrier with index k_0 in the set $\{1, 2, \dots, 31\}$ and a constant C_{k_0} to give a significant DAC output, and form the complex 64-dimensional array of subcarrier constellation points

$$Z_k = \begin{cases} 0 & \text{for } k = 0 \text{ and } 32 \\ C_{k_0} & \text{for } k = k_0 \\ \overline{C}_{k_0} & \text{for } k = 64 - k_0 \\ 0 & \text{otherwise} \end{cases} \quad (46)$$

Send the 64-point time sequence z_n repeatedly and without a cyclic prefix. Perform the IFFT for each symbol, even though the results will be the same each time, to make sure it is integrated into your ping-pong scheme correctly. The transmitted signal should be a cosine wave at the subcarrier frequency. Check that this is what you see on the oscilloscope. Try this for several subcarrier frequencies.

Once your ping-pong buffer scheme is working, create a program to make the transmitter send the following sequence of signals:

1. When your transmitter program first starts, send about one second of silence. That is, send samples that are all 0 volts for about one second. This will allow any initial transients that might occur when your program is loaded and started to disappear. It will also allow the receiver to recognize that the transmitter is not yet sending a signal and get into a waiting state.

2. Next, send the following symbol without a cyclic prefix 1024 times. This is called the R-REVERB signal in the ADSL standard.

The data bits for R-REVERB are generated by a 6-stage PN sequence generator according to the rule:

$$\begin{aligned} d_n &= 1 & \text{for } n &= 1, \dots, 6 \\ d_n &= d_{n-5} + d_{n-6} & \text{for } n &= 7, \dots, 64 \end{aligned} \tag{47}$$

Thus the initial state of the shift register $(d_6, d_5, d_4, d_3, d_2, d_1)$ is initialized to all 1's for each R-REVERB symbol, so they all use the same data sequence. The data bits for the OFDM symbols are taken from the right side of the shift register as it is clocked.

This sequence has period 63 so $d_{64} = d_1$. Subcarriers 0 and 32 are set to 0 and bits d_1 and d_2 are not used. Bit pairs (d_{2k+1}, d_{2k+2}) are used to QPSK modulate subcarriers $k = 1, \dots, 31$ as shown up to a scale factor in Table 1.

Table 1: Mapping of Two Data Bits to QPSK Constellation Points

d_{2k+1}	d_{2k+2}	Decimal Label	X_k	Y_k
0	0	0	+1	+1
0	1	1	+1	-1
1	0	2	-1	+1
1	1	3	-1	-1

Bits d_{33} and d_{34} which modulate subcarrier 16 are overwritten by bits (0,0) to generate the constellation point $(X_{16}, Y_{16}) = (+1, +1)$ to form a *pilot carrier*.

This will allow the receiver to detect the presence of a signal, estimate symbol clock timing frequency offset, and initially train its equalizer. You should compute the real 64-point R-REVERB time sequence just once, scale it to use a significant range of the DAC, convert it to the integer format for the right

DAC channel, and store the resulting 64-point sequence in an integer array for repeated transmission. Since you have already implemented ping-pong buffers, a good approach would be to store the 64-point sequence in both `ping[]` and `pong[]` and let your ping-pong scheme alternate between sending the two buffers.

3. After all the R-REVERB symbols have been sent, send the 180 degree phase reversal of the R-REVERB symbol ten times. It is called the R-SEGUE symbol in the ADSL standard. The phase reversal provides a timing mark that tells the receiver that actual customer data signal will be transmitted next.
4. Next, go into an endless loop and transmit data symbols including a cyclic prefix. The data blocks for the symbols should be generated by the 23-stage scrambler with all 1's as its input. Use QPSK for all the subcarriers.

5. Send a sync symbol after every 68 data symbols. The sync symbol is identical to the R-REVERB symbol except with the cyclic prefix added.
6. Once you get your receiver to work with QPSK, you can use 16-QAM for the subcarriers. You might want to send QPSK for some known time period initially, and then switch to 16-QAM. Send a sync symbol after every 68 data symbols.

Making the Receiver

Receivers are usually much more difficult to make than transmitters. They require functions to

- detect signal presence
- do automatic gain control
- acquire and track sample and symbol clocks
- perform adaptive equalization
- and implement sophisticated decoders for error correction

In this experiment, you will implement the receiver in a different DSK than the transmitter so sample clock acquisition and tracking, and symbol location issues must be addressed.

Connect the Line Out of your transmitter DSK to the Line In of the receiver DSK. Perform the tasks listed below to make your ADSL receiver.

1. Make a function to detect when an OFDM signal is present at the receiver input. One approach is to detect signal energy by squaring the input samples and averaging the squares with a first-order exponential averager. Let $d(n)$ be the averager output and $r(n)$ the received signal. Then the power detection system should compute

$$d(n) = \beta d(n - 1) + (1 - \beta)r^2(n) \quad (48)$$

where β is a positive constant slightly less than 1 that determines the averaging time constant. The closer β is to 1, the slower the averager will respond but the smoother its output will be.

When $d(n)$ rises above a threshold for several samples, decide an OFDM signal is present.

The threshold determines the false detection probability when the channel is noisy.

- When the threshold is low, an OFDM signal will be detected quickly but false alarms will also be frequent.
- When the threshold is high, false alarms will be rare but it will take longer to detect the signal. Of course, if the threshold is too high there will be no false alarms but the signal will never be detected.

Experimentally select values for β and the threshold so the start of a received OFDM signal is reliably detected by the end of one or two 64-sample symbols.

Before a signal is detected, keep the receiver functions reset and waiting. When a signal is detected, start a counter to keep track of how many R-REVERB symbols have been received.

2. The receiver must track sample clock frequency offset and OFDM symbol alignment. Also, the FFT you implemented does an in-place computation and overwrites the input array with the output array. These considerations make it inconvenient to use the ping-pong buffer approach in the receiver.

An approach you can use is to set up a circular buffer of floats for received samples whose length is two symbols long, that is, $2N' = 2(64 + \nu)$ elements. The received samples should be converted to floats when stored in the circular buffer. A variable pointing to the start of a symbol can be defined. Sample clock adjustments and symbol alignment can be made by modifying this pointer. The 64 samples for a symbol, excluding the cyclic prefix, can be copied to a separate input array for the FFT.

3. Begin collecting samples in the circular buffer once a signal is detected. Declare a 64-element complex array for the FFT input. Copy successive blocks of 64 samples to the real part of this array and zero the imaginary part. Form the FFT's of the blocks. During the first 200 R-REVERB symbols, estimate the sample clock fractional period error, ϵ , using the “open loop” method described on slide 17-28 using (31).
4. For the next several symbols, compute an estimate of the reciprocal of the channel frequency response, $1/H_k$, for each subcarrier. These will be used as the initial frequency domain equalizer (FEQ) coefficients. You already know the FFT values for the R-REVERB symbol from the definition of this signal. Let $R_{k,i}$ be the receiver's FFT output at index k for symbol i , and Z_k be the transmitted FFT value. The estimate for one

received symbol is

$$W_k(i) = \frac{Z_k}{R_{k,i}} = \frac{Z_k \overline{R_{k,i}}}{|R_{k,i}|^2} \quad (49)$$

The arithmetic average of these estimates over several symbols can be computed to reduce noise. You can do the divisions using the standard C floating-point division operator. Compute these estimates for $k = 1, \dots, 31$.

5. Initialize the FEQ coefficients to the values computed in the previous task. Then begin adaptively updating the equalizer coefficients using the LMS algorithm given by (16) in slide 17-3. Begin running the FEQ to compute the equalized values $\tilde{Z}_{k,i} = W_k(i)R_{k,i}$. Use an update constant μ large enough to track phase rotations caused by sample clock frequency offset but small enough to have relatively small coefficient jitter. You will have to turn on the slicer that quantizes FEQ outputs to the nearest ideal constellation points.

Do not start the sample clock tracking algorithm yet. Run in this mode for 256 symbols. The constellations you observe at the equalizer output should be the QPSK input constellations. You might ask why this works with no cyclic prefix? The answer is that the R-REVERB symbol is being repeated. The end of one symbol acts as the cyclic prefix for the next.

6. A good diagnostic tool and informative way of observing the receiver operation is a *constellation display*. This display shows the real versus imaginary parts of the FEQ outputs for the equalized symbol stream for a particular subcarrier in an X-Y display on an oscilloscope. When the channel noise is small and the equalization is good, well defined constellation points will be observed. If additive noise with approximately Gaussian characteristics is present, the ideal constellation points will smear into circular

clusters. Poor equalization will also cause clusters. When sample clock frequency offset is present and the equalizer updates are very slow, the observed constellation will slowly rotate.

To make a constellation display, send the real part of the equalizer output for a subcarrier to the left channel of the Line-Out and the imaginary part to the right channel. The equalized subcarrier outputs are only computed once per OFDM symbol. Repeat sending the equalized output for the chosen subcarrier for each of the $64 + \nu$ output samples during a symbol. Display the two channels as an X-Y plot on the oscilloscope. Freeze the equalizer coefficients by making μ zero and repeat R-REVERB symbols continuously. Look at a constellation display to see if there is constellation rotation caused by sample clock frequency offset.

7. After the 256 symbols of LMS equalizer updating (456 symbols after the start of signal detection) set the equalizer update scale factor to a small value and turn on the sample clock algorithm specified by (37) and (38) and perform the constellation counter-rotations. Initialize δ_i according to the open loop value computed during the first 200 received symbols. The clock PLL should track significantly faster than the equalizer. Adjust the timing phase when $|\tau_i|$ exceeds its threshold by shifting the 64-sample record copied from the circular buffer forward or backward by one sample and correcting τ_i as discussed on slide 17-33.
8. Continue running the FFT, FEQ, clock tracker, and symbol counter. When the counter gets close to 1024, begin looking for the 180 degree phase shift that occurs when R-SEGUE begins. Use your ingenuity to come up with an algorithm for detecting the phase shift. Your algorithm should operate on the

received sample sequence and detect the phase shift within 4 samples. Count for 10 symbols from this timing mark to determine when regular data transmission begins with symbols including a cyclic prefix. The timing mark should allow you to determine with reasonable accuracy when the cyclic prefix of the first data symbol starts.

9. Turn on the cyclic prefix correlation algorithm presented starting on slide 17-35 for determining symbol alignment when the reception of regular data symbols begins. You should already have a ball park estimate of when a symbol begins. Search around this point in the circular buffer for the phasing that gives a null in the periodicity metric or a peak in the cyclic prefix correlation. For each received symbol, skip over the cyclic prefix and copy 64 good symbol samples to the FFT input array. Perform the FFT, FEQ, sample clock tracking, slicing, inverse map to a bit sequence, and descrambling. If everything is

working properly, the descrambler output should be all 1's.

Additional optional tasks you can perform are:

1. Add Gaussian noise to the transmitted samples and make a plot of the bit-error rate as a function of signal-to-noise ratio.
2. Implement the interleaver in the transmitter and deinterleaver in the receiver.
3. Pass the transmitter output samples through a filter simulating a worse channel before sending them to the DAC in the transmitter and observe system performance.
4. Come up with a scheme for estimating the signal-to-noise ratio at the receiver for each subcarrier when there is noise on the channel and the amplitude response varies significantly. Do some research and implement an algorithm for optimally assigning the number of bits to each

subcarrier to achieve a desired error probability.

5. Implement the ADSL Reed-Solomon encoder in the transmitter and decoder in the receiver. This is a major task and is worthy of being counted as an entire experiment.
6. Browse through the various standards for systems that use OFDM and implement additional features they contain.