

Multirate Digital Signal Processing, Oversampling of Analog-to-Digital Conversion, and Undersampling of Bandpass Signals

12

CHAPTER OUTLINE

12.1 Multirate Digital Signal Processing Basics.....	555
12.1.1 Sampling Rate Reduction by an Integer Factor	556
12.1.2 Sampling Rate Increase by an Integer Factor	562
12.1.3 Changing the Sampling Rate by a Noninteger Factor L/M	567
12.1.4 Application: CD Audio Player	571
12.1.5 Multistage Decimation	574
12.2 Polyphase Filter Structure and Implementation.....	578
12.3 Oversampling of Analog-to-Digital Conversion	585
12.3.1 Oversampling and Analog-to-Digital Conversion Resolution	586
12.3.2 Sigma-Delta Modulation Analog-to-Digital Conversion	592
12.4 Application Example: CD Player.....	601
12.5 Undersampling of Bandpass Signals	603
Simulation Example	608
12.6 Sampling Rate Conversion Using the TMS320C6713 DSK	608
12.7 Summary	613

OBJECTIVES:

This chapter investigates basics of multirate digital signal processing, illustrates how to change a sampling rate for speech and audio signals, and describes the polyphase implementation for the decimation filter and interpolation filter. Next, the chapter introduces the advanced analog-to-digital conversion system with the oversampling technique and sigma-delta modulation. Finally, the chapter explores the principles of undersampling of bandpass signals.

12.1 MULTIRATE DIGITAL SIGNAL PROCESSING BASICS

In many areas of digital signal processing (DSP) applications—such as communications, speech, and audio processing—rising or lowering of a sampling rate is required. The principles relating to changing the sampling rate belong essentially within the topic of *multirate signal processing* (Ifeachor

and Jervis, 2002; Proakis and Manolakis, 1996; Porat, 1997; Sorensen and Chen, 1997). As an introduction, we will focus on the sampling rate conversion; that is, sampling rate reduction or increase.

12.1.1 Sampling Rate Reduction by an Integer Factor

The process of reducing the sampling rate by an integer factor is referred to as *downsampling* of a data sequence. We also refer downsampling as “decimation” (not taking one of ten). The term “decimation” has been accepted and used in many textbooks and fields. To downsample a data sequence $x(n)$ by an integer factor of M , we use the following notation:

$$y(m) = x(mM) \quad (12.1)$$

where $y(m)$ is the downsampled sequence, obtained by taking a sample from the data sequence $x(n)$ for every M samples (discarding $M-1$ samples for every M samples). As an example, if the original sequence with a sampling period $T = 0.1$ second (sampling rate = 10 samples per second) is given by

$$x(n) : 8 \quad 7 \quad 4 \quad 8 \quad 9 \quad 6 \quad 4 \quad 2 \quad -2 \quad -5 \quad -7 \quad -7 \quad -6 \quad -4 \dots$$

and we downsample the data sequence by a factor of 3, we obtain the downsampled sequence as

$$y(m) : 8 \quad 8 \quad 4 \quad -5 \quad -6 \dots$$

with the resultant sampling period $T = 3 \times 0.1 = 0.3$ second (the sampling rate now is 3.33 samples per second). Although the example is straightforward, there is a requirement to avoid aliasing noise. We will illustrate this next.

From the Nyquist sampling theorem, it is known that aliasing can occur in the downsampled signal due to the reduced sampling rate. After downsampling by a factor of M , the new sampling period becomes MT , and therefore the new sampling frequency is

$$f_{sM} = \frac{1}{MT} = \frac{f_s}{M} \quad (12.2)$$

where f_s is the original sampling rate.

Hence, the folding frequency after downsampling becomes

$$f_{sM}/2 = \frac{f_s}{2M} \quad (12.3)$$

This tells us that after downsampling by a factor of M , the new folding frequency will be decreased M times. If the signal to be downsampled has frequency components larger than the new folding frequency, $f > f_s/(2M)$, aliasing noise will be introduced into the downsampled data.

To overcome this problem, it is required that the original signal $x(n)$ be processed by a lowpass filter $H(z)$ before downsampling, which should have a stop frequency edge at $f_s/(2M)$ (Hz). The corresponding normalized stop frequency edge is then converted to

$$\Omega_{stop} = 2\pi \frac{f_s}{2M} T = \frac{\pi}{M} \text{ radians} \quad (12.4)$$

In this way, before downsampling, we can guarantee the maximum frequency of the filtered signal satisfies

$$f_{\max} < \frac{f_s}{2M} \quad (12.5)$$

such that no aliasing noise is introduced after downsampling. A general block diagram of decimation is given in Figure 12.1, where the filtered output in terms of the z-transform can be written as

$$W(z) = H(z)X(z) \quad (12.6)$$

where $X(z)$ is the z-transform of the sequence to be decimated, $x(n)$, and $H(z)$ is the lowpass filter transfer function. After anti-aliasing filtering, the downsampled signal $y(m)$ takes its value from the filter output as

$$y(m) = w(mM) \quad (12.7)$$

The process of reducing the sampling rate by a factor of 3 is shown in Figure 12.1. The corresponding spectral plots for $x(n)$, $w(n)$, and $y(m)$ in general are shown in Figure 12.2.

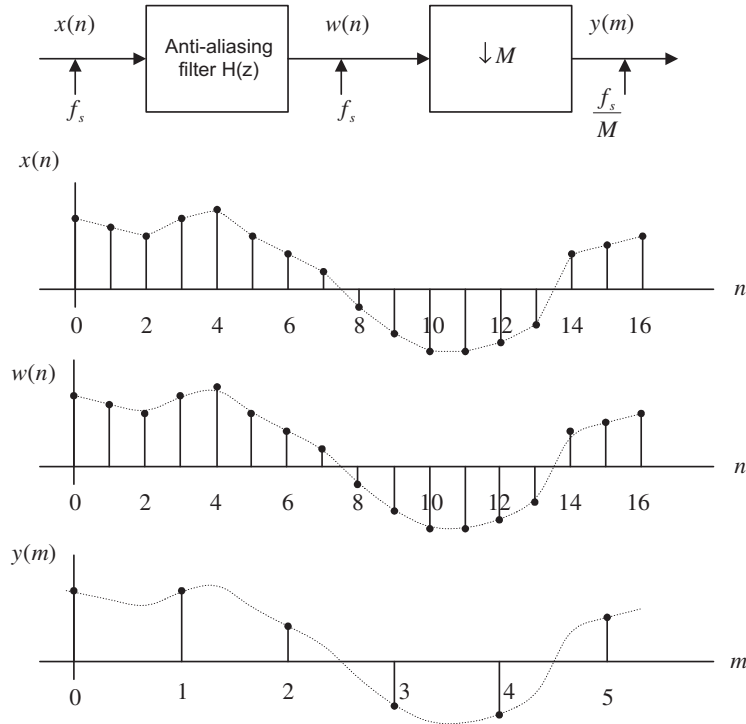
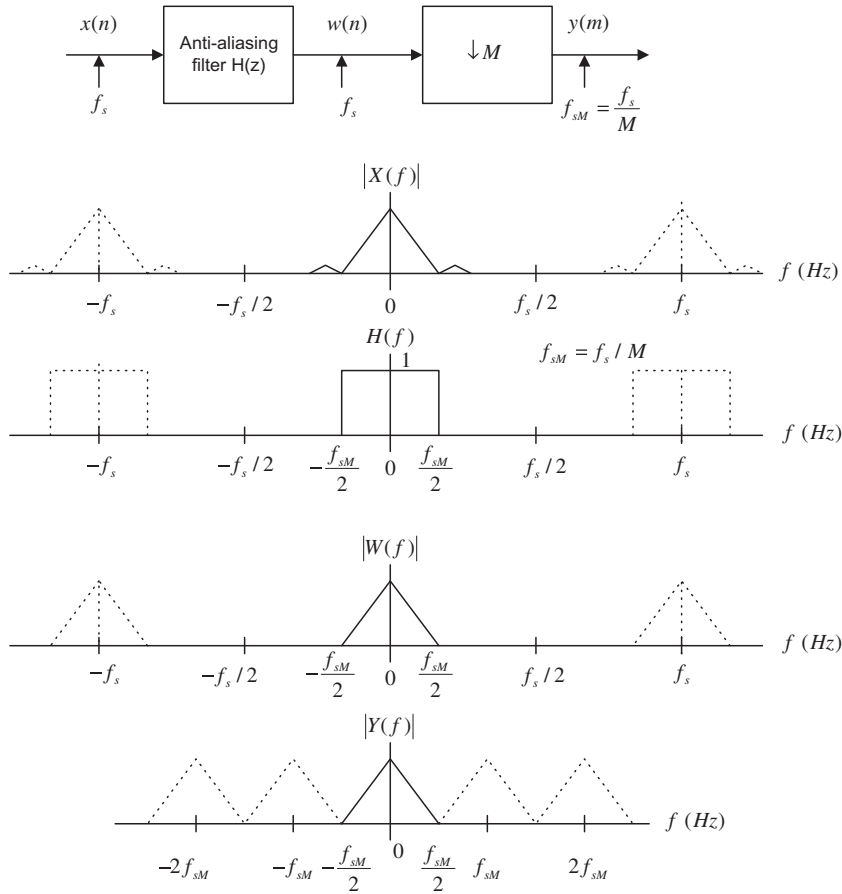


FIGURE 12.1

Block diagram of the downsampling process with $M = 3$.

**FIGURE 12.2**

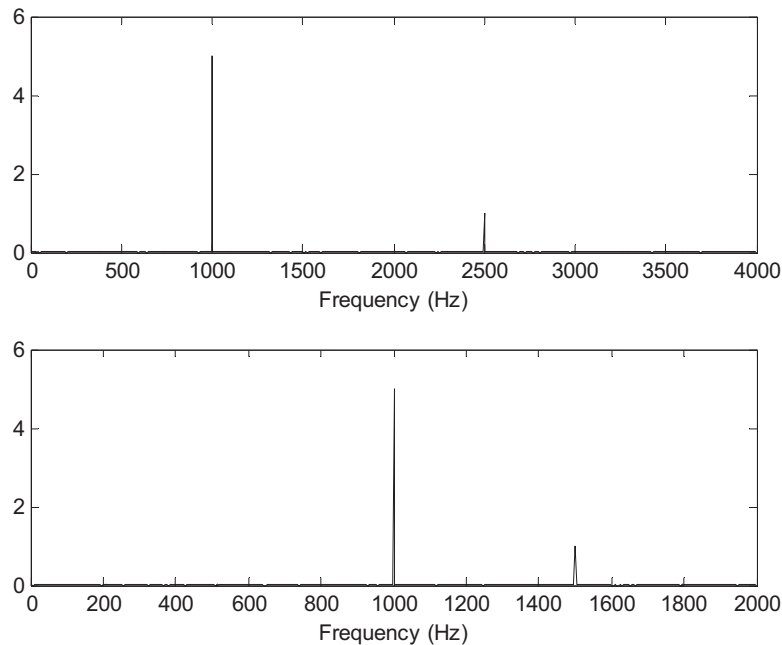
Spectrum after downsampling.

To verify this principle, let us consider a signal $x(n)$ generated by the following:

$$x(n) = 5 \sin\left(\frac{2\pi \times 1,000n}{8,000}\right) + \cos\left(\frac{2\pi \times 2,500}{8,000}\right) = 5 \sin\left(\frac{n\pi}{4}\right) + \cos\left(\frac{5n\pi}{8}\right) \quad (12.8)$$

With a sampling rate of $f_s = 8,000$ Hz, the spectrum of $x(n)$ is plotted in the first graph in [Figure 12.3A](#), where we observe that the signal has components at frequencies of 1,000 Hz and 2,500 Hz. Now we downsample $x(n)$ by a factor of 2, that is, $M = 2$. According to Equation (12.3), we know that the new folding frequency is $4,000/2 = 2,000$ Hz. Hence, without using the anti-aliasing lowpass filter, the spectrum would contain an aliasing frequency of $4 \text{ kHz} - 2.5 \text{ kHz} = 1.5 \text{ kHz}$ introduced by 2.5 kHz, plotted in the second graph in [Figure 12.3A](#).

Now we apply a finite impulse response (FIR) lowpass filter designed with a filter length of $N = 27$ and a cutoff frequency of 1.5 kHz to remove the 2.5 kHz signal before downsampling to avoid

**FIGURE 12.3A**

Spectrum before downsampling and spectrum after downsampling without using the anti-aliasing filter.

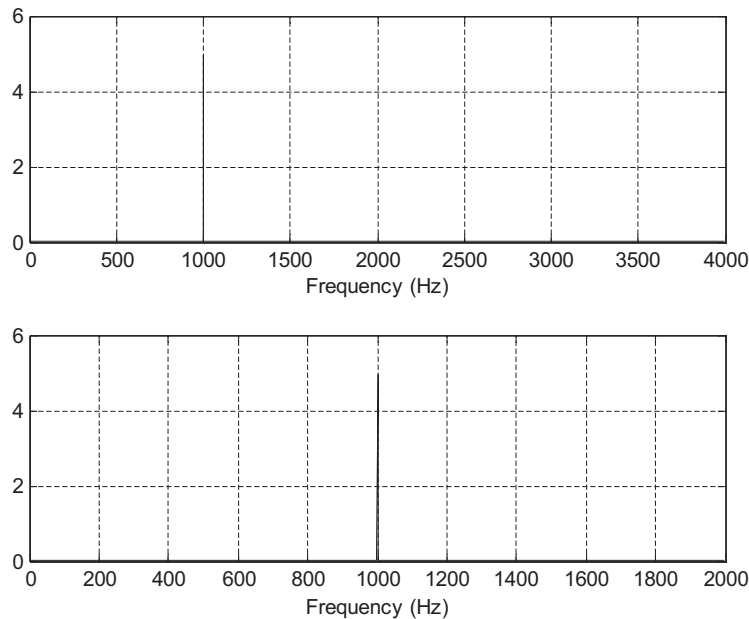
aliasing. How to obtain such specifications will be discussed in the later example. The normalized cutoff frequency used for design is given by

$$\Omega_c = 2\pi \times 1,500 \times (1/8,000) = 0.375\pi$$

Thus, the aliasing noise is avoided. The spectral plots are given in [Figure 12.3B](#), where the first plot shows the spectrum of $w(n)$ after anti-aliasing filtering, while the second plot describes the spectrum of $y(m)$ after downsampling. Clearly, we prevent aliasing noise in the downsampled data by sacrificing the original 2.5-kHz signal. Program 12.1 gives the details of the MATLAB implementation.

Program 12.1. MATLAB program for decimation.

```
close all; clear all;
% Downsampling filter (see Chapter 7 for FIR filter design)
B=[0.00074961181416 0.00247663033476 0.00146938649416 -0.00440446121505 ...
-0.00910635730662 0.00000000000000 0.02035676831506 0.02233710562885...
-0.01712963672810 -0.06376620649567 -0.03590670035210 0.10660980550088...
0.29014909103794 0.37500000000000 0.29014909103794 0.10660980550088...
-0.03590670035210 -0.06376620649567 -0.01712963672810 0.02233710562885...
0.02035676831506 0.00000000000000 -0.00910635730662 -0.00440446121505...
0.00146938649416 0.00247663033476 0.00074961181416];
```

**FIGURE 12.3B**

Spectrum before downsampling and spectrum after downsampling using the anti-aliasing filter.

```
% Generate 2048 samples
fs=8000;           % Sampling rate
N=2048;           % Number of samples
M=2;              % Downsample factor
n=0:1:N-1;
x=5*sin(n*pi/4)+cos(5*n*pi/8);
% Compute single-sided amplitude spectrum
% AC component will be doubled, and DC component will be kept at the same value
X=2*abs(fft(x,N))/N;X(1)=X(1)/2;
% Map the frequency index up to the folding frequency in Hz
f=[0:1:N/2-1]*fs/N;
% Downsampling
y=x(1:M:N);
NM=length(y);      % Length of the downsampled data
% Compute the single-sided amplitude spectrum for the downsampled signal
Y=2*abs(fft(y,NM))/length(y);Y(1)=Y(1)/2;
% Map the frequency index to the frequency in Hz
fsM=[0:1:NM/2-1]*(fs/M)/NM;
subplot(2,1,1);plot(f,X(1:1:N/2));grid; xlabel('Frequency (Hz)');
subplot(2,1,2);plot(fsM,Y(1:1:NM/2));grid; xlabel('Frequency (Hz)');
figure
```

```

w=filter(B,1,x);          % Anti-aliasing filtering
% Compute the single-sided amplitude spectrum for the filtered signal
W=2*abs(fft(w,N))/N;W(1)=W(1)/2;
% Downsampling
y=w(1:M:N);
NM=length(y);
% Compute the single-sided amplitude spectrum for the downsampled signal
Y=2*abs(fft(y,NM))/NM;Y(1)=Y(1)/2;
% Plot spectra
subplot(2,1,1);plot(f,W(1:1:N/2));grid; xlabel('Frequency (Hz)');
subplot(2,1,2);plot(f,M,Y(1:1:NM/2));grid; xlabel('Frequency (Hz)');

```

Now we focus on how to design an anti-aliasing FIR filter, or decimation filter. We will discuss this topic via the following example.

EXAMPLE 12.1

Consider a DSP downsampling system with the following specifications:

Sampling rate = 6,000 Hz

Input audio frequency range = 0–800 Hz

Passband ripple = 0.02 dB

Stopband attenuation = 50 dB

Downsample factor $M = 3$

Determine the FIR filter length, cutoff frequency, and window type if the window method is used.

Solution:

The specifications are reorganized as

Anti-aliasing filter operating at the sampling rate = 6,000 Hz

Passband frequency range = 0–800 Hz

Stopband frequency range = 1–3 kHz

Passband ripple = 0.02 dB

Stopband attenuation = 50 dB

Filter type = FIR

The block diagram and specifications are depicted in Figure 12.4.

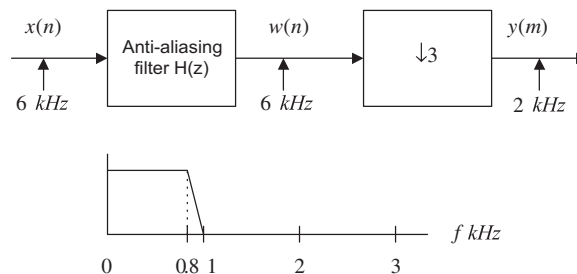


FIGURE 12.4

Filter specifications for Example 12.1.

The Hamming window is selected, since it provides 0.019 dB ripple and 53 dB stopband attenuation. The normalized transition band is given by

$$\Delta f = \frac{f_{stop} - f_{pass}}{f_s} = \frac{1,000 - 800}{6,000} = 0.033$$

The length of the filter and the cutoff frequency can be determined by

$$N = \frac{3.3}{\Delta f} = \frac{3.3}{0.033} = 100$$

We choose an odd number; that is, $N = 101$, and

$$f_c = \frac{f_{pass} + f_{stop}}{2} = \frac{800 + 1,000}{2} = 900 \text{ Hz}$$

12.1.2 Sampling Rate Increase by an Integer Factor

Increasing a sampling rate is a process of upsampling by an integer factor of L . This process is described as follows:

$$y(m) = \begin{cases} x\left(\frac{m}{L}\right) & m = nL \\ 0 & \text{otherwise} \end{cases} \quad (12.9)$$

where $n = 0, 1, 2, \dots$, $x(n)$ is the sequence to be upsampled by a factor of L , and $y(m)$ is the upsampled sequence. As an example, suppose that the data sequence is given as follows:

$$x(n): 8 \ 8 \ 4 \ -5 \ -6 \dots$$

After upsampling the data sequence $x(n)$ by a factor of 3 (adding $L - 1$ zeros for each sample), we have the upsampled data sequence $w(m)$ as

$$w(m): 8 \ 0 \ 0 \ 8 \ 0 \ 0 \ 4 \ 0 \ 0 \ -5 \ 0 \ 0 \ -6 \ 0 \ 0 \dots$$

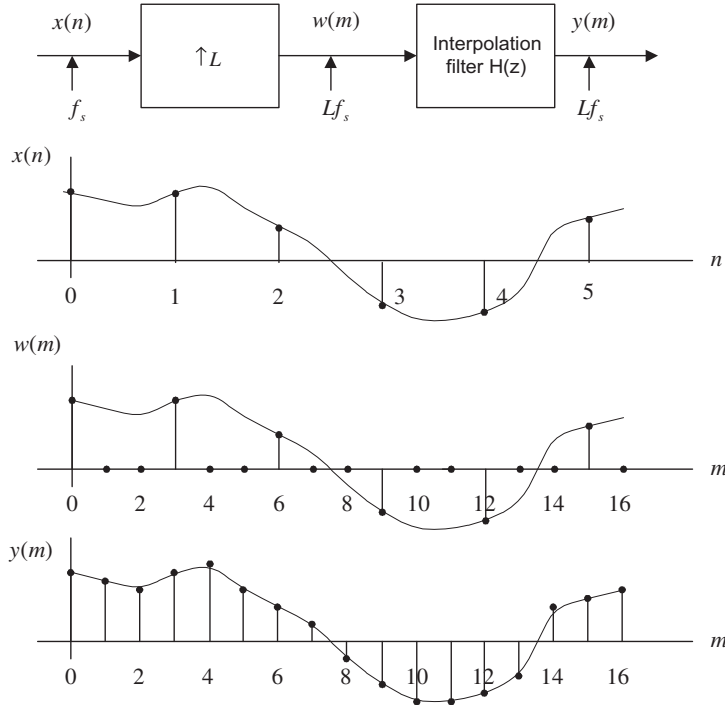
The next step is to smooth the upsampled data sequence via an interpolation filter. The process is illustrated in [Figure 12.5A](#).

Similar to the downsampling case, assuming that the data sequence has the current sampling period of T , the Nyquist frequency is given by $f_{\max} = f_s/2$. After upsampling by a factor of L , the new sampling period becomes T/L , thus the new sampling frequency is changed to be

$$f_{sL} = Lf_s \quad (12.10)$$

This indicates that after upsampling, the spectral replicas originally centered at $\pm f_s, \pm 2f_s, \dots$ are included in the frequency range from 0 Hz to the new Nyquist limit $Lf_s/2$ Hz, as shown in [Figure 12.5B](#). To remove those included spectral replicas, an interpolation filter with a stop frequency edge of $f_s/2$ in Hz must be attached, and the normalized stop frequency edge is given by

$$\Omega_{stop} = 2\pi \left(\frac{f_s}{2} \right) \times \left(\frac{T}{L} \right) = \frac{\pi}{L} \text{ radians} \quad (12.11)$$


FIGURE 12.5A

Block diagram for the upsampling process with $L = 3$.

After filtering via the interpolation filter, we will achieve the desired spectrum for $y(n)$, as shown in [Figure 12.5B](#). Note that since the interpolation is to remove the high-frequency images that are aliased by the upsampling operation, it is essentially an anti-aliasing lowpass filter.

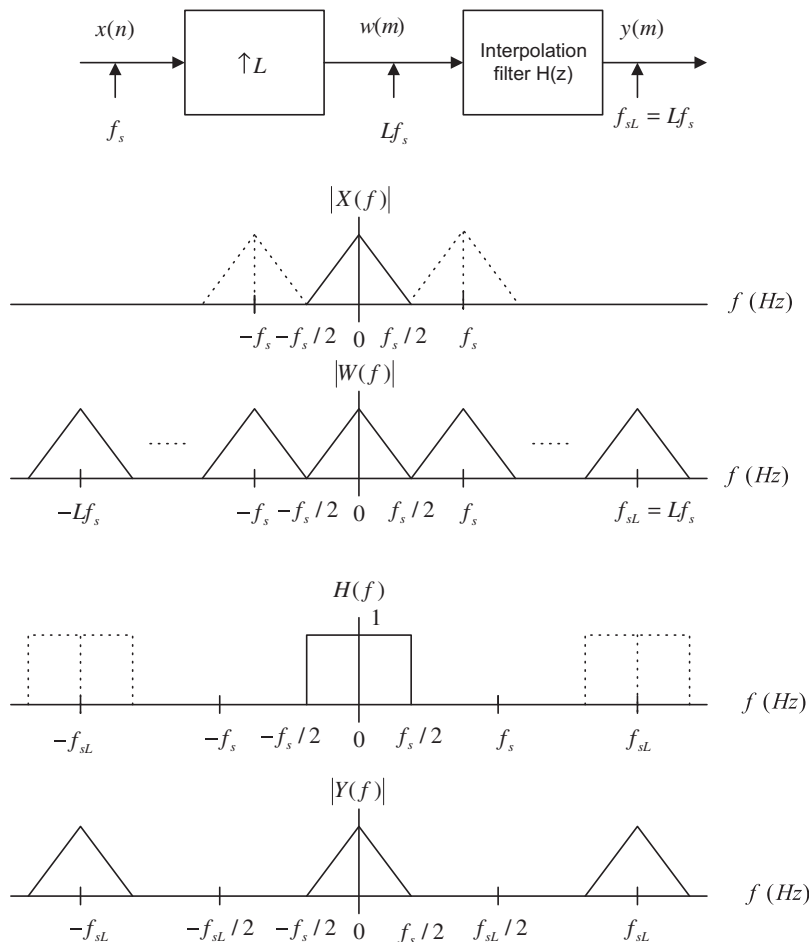
To verify the upsampling principle, we generate the signal $x(n)$ with 1 kHz and 2.5 kHz as follows:

$$x(n) = 5 \sin \left(\frac{2\pi \times 1,000n}{8,000} \right) + \cos \left(\frac{2\pi \times 2,500n}{8,000} \right)$$

with a sampling rate of $f_s = 8,000$ Hz. The spectrum of $x(n)$ is plotted in [Figure 12.6](#). Now we upsample $x(n)$ by a factor of 3, that is, $L = 3$. We know that the sampling rate is increased to $3 \times 8,000 = 24,000$ Hz. Hence, without using the interpolation filter, the spectrum would contain the image frequencies originally centered at the multiple frequencies of 8 kHz. The top plot in [Figure 12.6](#) shows the spectrum for the sequence after upsampling and before applying the interpolation filter.

Now we apply an FIR lowpass filter designed with a length of 53, a cutoff frequency of 3,250 Hz, and a new sampling rate of 24,000 Hz as the interpolation filter, whose normalized frequency should be

$$\Omega_c = 2\pi \times 3,250 \times \left(\frac{1}{24,000} \right) = 0.2708\pi$$

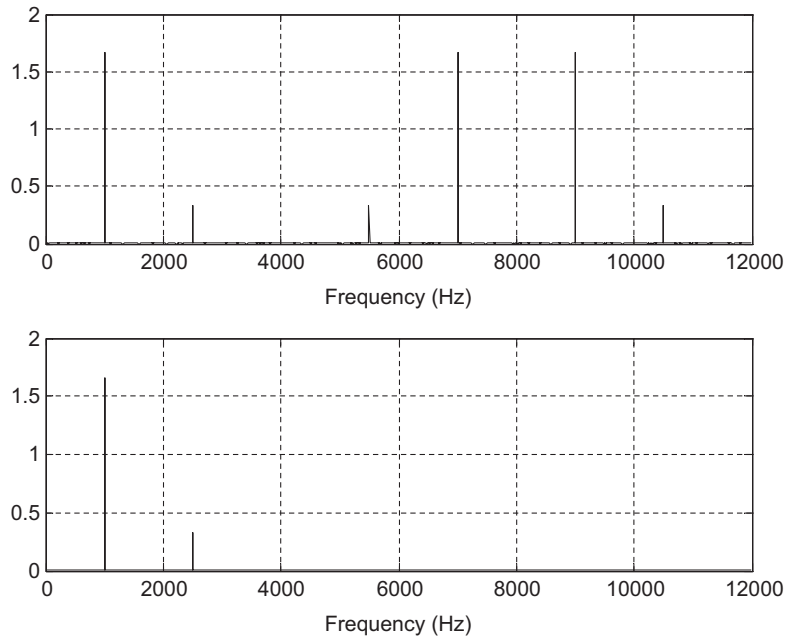
**FIGURE 12.5B**

Spectra before and after upsampling

The bottom plot in Figure 12.6 shows the spectrum for $y(m)$ after applying the interpolation filter, where only the original signals with frequencies of 1 kHz and 2.5 kHz are presented. Program 12.2 shows the implementation details in MATLAB.

Program 12.2. MATLAB program for interpolation.

```
close all; clear all
% Upsampling filter (see Chapter 7 for FIR filter design)
B = [-0.00012783931504 0.00069976044649 0.00123831516738 0.00100277549136...
     -0.00025059018468 -0.00203448515158 -0.00300830295487 -0.00174101657599...
     0.00188598835011 0.00578414933758 0.00649330625041 0.00177982369523...]
```

**FIGURE 12.6**

(Top) The spectrum after upsampling and before applying the interpolation filter; (bottom) spectrum after applying the interpolation filter.

```
-0.00670672686935 -0.01319379342716 -0.01116855281442 0.00123034314117...
0.01775600060894 0.02614700427364 0.01594155162392 -0.01235169936557...
-0.04334322148505 -0.05244745563466 -0.01951094855292 0.05718573279009...
0.15568416401644 0.23851539047347 0.27083333333333 0.23851539047347...
0.15568416401644 0.05718573279009 -0.01951094855292 -0.05244745563466...
-0.04334322148505 -0.01235169936557 0.01594155162392 0.02614700427364...
0.01775600060894 0.00123034314117 -0.01116855281442 -0.01319379342716...
-0.00670672686935 0.00177982369523 0.00649330625041 0.00578414933758...
0.00188598835011 -0.00174101657599 -0.00300830295487 -0.00203448515158...
-0.00025059018468 0.00100277549136 0.00123831516738 0.00069976044649...
-0.00012783931504];
% Generate the 2048 samples with fs = 8000 Hz
fs=8000;           % Sampling rate
N=2048;            % Number of samples
L = 3;             % Upsampling factor
n=0:1:N-1;
x=5*sin(n*pi/4)+cos(5*n*pi/8);
% Upsampling by a factor of L
w=zeros(1,L*N);
```

```

for n=0:1:N-1
    w(L*n+1)=x(n+1);
end
NL = length(w);           % Length of the upsampled data
W=2*abs(fft(w,NL))/NL;W(1)=W(1)/2; % Compute one-sided amplitude spectrum
f=[0:1:NL/2-1]*fs*L/NL; % Map the frequency index to the frequency (Hz)
% Interpolation
y=filter(B,1,w);          % Apply interpolation filter
Y=2*abs(fft(y,NL))/NL;Y(1)=Y(1)/2; % Compute the one-sided amplitude spectrum
fsL=[0:1:NL/2-1]*fs*L/NL; % Map the frequency index to the frequency (Hz)
subplot(2,1,1);plot(f,W(1:1:NL/2));grid; xlabel('Frequency (Hz)');
subplot(2,1,2);plot(fsL,Y(1:1:NL/2));grid; xlabel('Frequency (Hz)');

```

Now let us study how to design an interpolation filter via Example 12.2.

EXAMPLE 12.2

Consider a DSP upsampling system with the following specifications:

Sampling rate = 6,000 Hz

Input audio frequency range = 0–800 Hz

Passband ripple = 0.02 dB

Stopband attenuation = 50 dB

Upsample factor $L = 3$

Determine the FIR filter length, cutoff frequency, and window type if the window design method is used.

Solution:

The specifications are reorganized as follows:

Interpolation filter operating at the sampling rate = 18,000 Hz

Passband frequency range = 0–800 Hz

Stopband frequency range = 3–9 kHz

Passband ripple = 0.02 dB

Stopband attenuation = 50 dB

Filter type: FIR filter

The block diagram and filter frequency specifications are given in Figure 12.7.

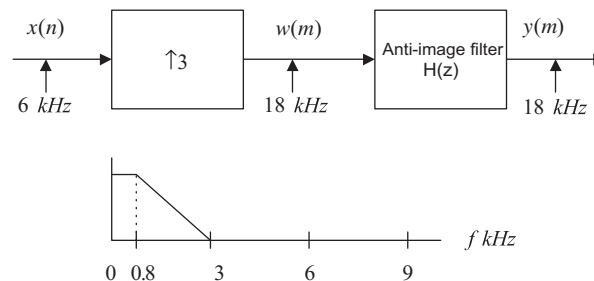


FIGURE 12.7

Filter frequency specifications for Example 12.2.

We choose the Hamming window for this application. The normalized transition band is

$$\Delta f = \frac{f_{stop} - f_{pass}}{f_{sL}} = \frac{3,000 - 800}{18,000} = 0.1222$$

The length of the filter and the cutoff frequency can be determined by

$$N = \frac{3.3}{\Delta f} = \frac{3.3}{0.1222} = 27$$

and the cutoff frequency is given by

$$f_c = \frac{f_{pass} + f_{stop}}{2} = \frac{3,000 + 800}{2} = 1,900 \text{ Hz.}$$

12.1.3 Changing the Sampling Rate by a Noninteger Factor L/M

With an understanding of the downsampling and upsampling processes, we now study sampling rate conversion by a noninteger L/M . This can be viewed as two sampling conversion processes. In step 1, we perform the upsampling process by a factor of integer L following application of an interpolation filter $H_1(z)$; in step 2, we continue filtering the output from the interpolation filter via an anti-aliasing filter $H_2(z)$, and finally execute downsampling. The entire process is illustrated in Figure 12.8.

Since the interpolation and anti-aliasing filters are in a cascaded form and operate at the same rate, we can select one of them. We choose the one with the lower stop frequency edge and choose the most demanding requirements for passband gain and stopband attenuation for the filter design. A lot of computational savings can be achieved by using one lowpass filter. We illustrate the procedure via the following simulation. Let us generate the signal $x(n)$ by

$$x(n) = 5 \sin\left(\frac{2\pi \times 1,000n}{8,000}\right) + \cos\left(\frac{2\pi \times 2,500n}{8,000}\right)$$

with a sampling rate of $f_s = 8,000$ Hz and frequencies of 1 kHz and 2.5 kHz. Now we resample $x(n)$ to 3,000 Hz by a noninteger factor of 0.375, that is,

$$\left(\frac{L}{M}\right) = 0.375 = \frac{3}{8}$$

Upsampling is at a factor of $L = 3$ and the upsampled sequence is filtered by an FIR lowpass filter designed with a filter length $N = 53$ and a cutoff frequency of 3,250 Hz at a sampling rate of $3 \times 8,000 = 24,000$ Hz. The spectrum for the upsampled sequence and the spectrum after application of the interpolation filter are plotted in Figure 12.9A.

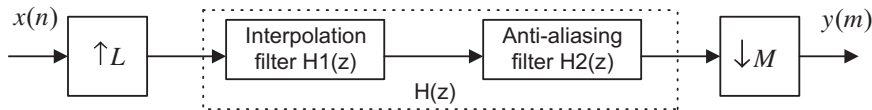
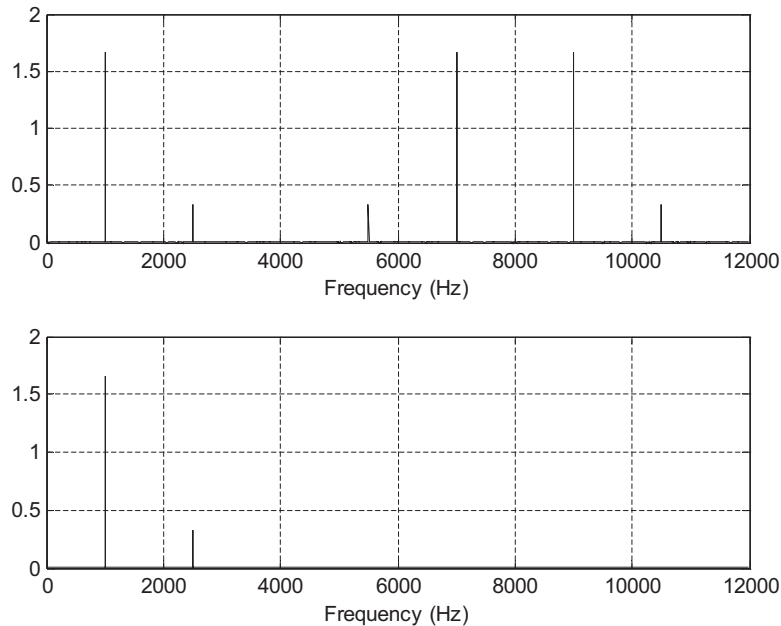


FIGURE 12.8

Block diagram for sampling rate conversion.

**FIGURE 12.9A**

(Top) Spectrum after upsampling, and (bottom) spectrum after interpolation filtering.

The sequence from step 1 can be filtered via another FIR lowpass filter with filter length $N = 159$ and a cutoff frequency of 1,250 Hz, followed by downsampling by a factor of $M = 8$. The spectrum after the anti-aliasing filter and the spectrum for the final output $y(m)$ are plotted in Figure 12.9B. Note that the anti-aliasing filter removes the frequency component of 2.5 kHz to avoid aliasing. This is because after downsampling, the Nyquist limit is 1.5 kHz. As we discussed previously, we can select one filter for implementation. We choose a FIR lowpass filter with $N = 159$ and a cutoff frequency of 1,250 Hz because its bandwidth is smaller than that of the interpolation filter. The MATLAB implementation is listed in Program 12.3.

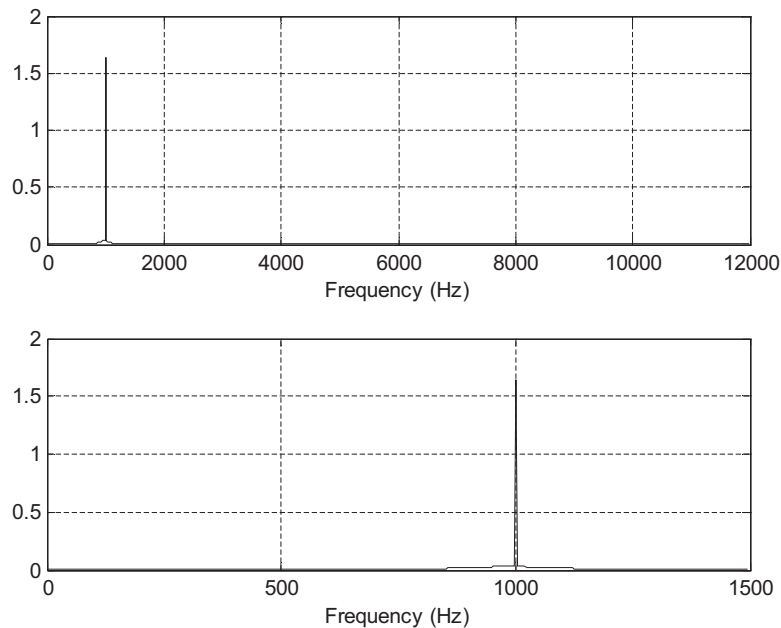
Program 12.3 MATLAB program for changing sampling rate with a noninteger factor.

```
close all; clear all; clc;
% Downsampling filter
Bdown=firwd(159,1,2*pi*1250/24000,0.4);
% Generate 2048 samples with fs=8000 Hz
fs=8000; % Original sampling rate
N = 2048; % The number of samples
L=3; % Upsampling factor
M=8; % Downsampling factor
n=0:1:N-1; % Generate the time index
x=5*sin(n*pi/4)+cos(5*n*pi/8); % Generate the test signal
% Upsampling by a factor of L
w1=zeros(1,L*N);
```

```

for n=0:1:N-1
    w1(L*n+1)=x(n+1);
end
NL= length(w1);           % Length of upsampled data
W1=2*abs(fft(w1,NL))/NL;W1(1)=W1(1)/2; % Compute the one-sided
                                % amplitude spectrum
f=[0:1:NL/2-1]*fs*L/NL;      % Map frequency index to its frequency in Hz
subplot(3,1,1);plot(f,W1(1:1:NL/2));grid
xlabel('Frequency (Hz)');
w2=filter(Bdown,1,w1);      % Perform the combined anti-aliasing filter
W2=2*abs(fft(w2,NL))/NL;W2(1)=W2(1)/2; % Compute the one-sided
                                % amplitude spectrum
y2=w2(1:M:NL);
NM=length(y2);             % Length of the downsampled data
Y2=2*abs(fft(y2,NM))/NM;Y2(1)=Y2(1)/2;% Compute the one-sided
                                %amplitude spectrum
% Map frequency index to its frequency in Hz before downsampling
fbar=[0:1:NL/2-1]*24000/NL;
% Map frequency index to its frequency in Hz
fsM=[0:1:NM/2-1]*(fs*L/M)/NM;
subplot(3,1,2);plot(f,W2(1:1:NL/2));grid; xlabel('Frequency (Hz)');
subplot(3,1,3);plot(fsM,Y2(1:1:NM/2));grid; xlabel('Frequency (Hz)');

```

**FIGURE 12.9B**

(Top) Spectrum after anti-aliasing filtering, and (bottom) spectrum after downsampling.

Therefore, three steps are required to accomplish the process:

1. Upsampling by a factor of $L = 3$;
2. Filtering the upsampled sequence by an FIR lowpass filter designed with filter length $N = 159$ and a cutoff frequency of 1,250 Hz at a sampling rate of $3 \times 8,000 = 24,000$ Hz;
3. Downsampling by a factor of $M = 8$.

EXAMPLE 12.3

Consider a sampling conversion DSP system (Figure 12.10A) with the following specifications:

Audio input $x(n)$ is sampled at the rate of 6,000 Hz.

Audio output $y(m)$ is operated at the rate of 9,000 Hz.

Determine the filter length and cutoff frequency for the combined anti-aliasing filter $H(z)$, and the window types, respectively, if the window design method is used.

Solution:

The filter frequency specifications and corresponding block diagram are developed in Figure 12.10B.

Specifications for the interpolation filter $H_1(z)$:

Passband frequency range = 0–2,500 Hz

Passband ripples for $H_1(z) = 0.04$ dB

Stopband frequency range = 3,000–9,000 Hz

Stopband attenuation = 42 dB

Specifications for the anti-aliasing filter $H_2(z)$:

Passband frequency range = 0–2,500 Hz

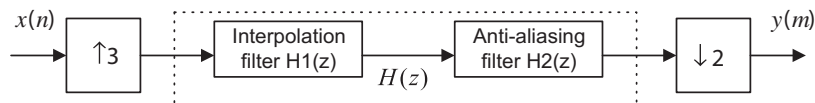


FIGURE 12.10A

Sampling conversion in Example 12.3.

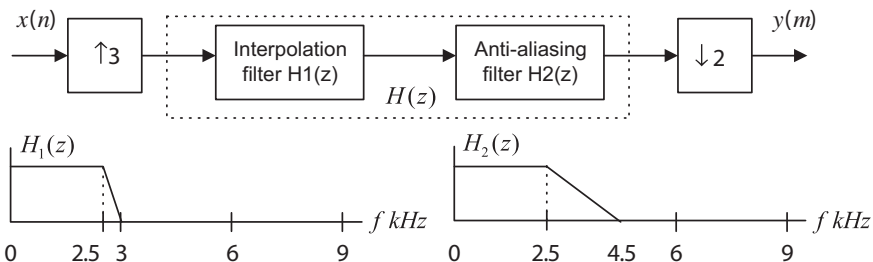


FIGURE 12.10B

Filter frequency specifications for Example 12.3.

Passband ripples for $H_2(z) = 0.02$ dB
 Stopband frequency range = 4,500–9,000 Hz
 Stopband attenuation = 46 dB
 Combined specifications $H(z)$:
 Passband frequency range = 0–2,500 Hz
 Passband ripples for $H(z) = 0.02$ dB
 Stopband frequency range = 3,000–9,000 Hz
 Stopband attenuation = 46 dB

We use an FIR filter with a Hamming window. Since

$$\Delta f = \frac{f_{stop} - f_{pass}}{f_{sL}} = \frac{3,000 - 2,500}{18,000} = 0.0278$$

the length of the filter and the cutoff frequency can be determined by

$$N = \frac{3.3}{\Delta f} = \frac{3.3}{0.0278} = 118.8$$

We choose $N = 119$, and

$$f_c = \frac{f_{pass} + f_{stop}}{2} = \frac{3,000 + 2,500}{2} = 2,750 \text{ Hz}$$

12.1.4 Application: CD Audio Player

In this application example, we will discuss principles of the upsampling and interpolation-filter processes used in CD audio systems to help with reconstruction filter design.

Each raw digital sample recorded on a CD audio system contains 16 bits and is sampled at the rate of 44.1 kHz. Figure 12.11 describes a portion of one channel of the CD player in terms of a simplified block diagram.

Let us consider the situation without upsampling and application of a digital interpolation filter. We know that the audio signal has a bandwidth of 22.05 kHz, that is, the Nyquist frequency, and digital-to-analog conversion (DAC) produces the sample-and-hold signals that contain the desired audio band and images thereof. To achieve the audio band signal, we need to apply a *reconstruction filter* (also called a smooth filter or anti-image filter) to remove all image frequencies beyond the Nyquist frequency of 22.05 kHz. Due to the requirement of the sharp transition band, a higher-order analog filter design becomes a requirement.

The design of the higher-order analog filter is complex and expensive to implement. As shown in Figure 12.11, in order to relieve such design constraints, we can add the upsampling process

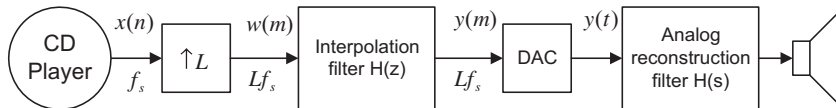


FIGURE 12.11

Sample rate conversion in the CD audio player system.

before DAC, followed by application of the digital interpolation filter (assume $L = 4$). Certainly, the interpolation filter design must satisfy the specifications studied in the previous section on increasing the sampling rate by an integer factor. Again, after digital interpolation, the audio band is kept the same, while the sampling frequency is increased by fourfold ($L = 4$), that is, $44.1 \times 4 = 176.4$ kHz.

Since the audio band of 22.05 kHz is now relatively low compared with the new folding frequency ($176.4/2 = 88.2$ kHz), the use of a simple first-order or second-order analog anti-image filter may be sufficient. Let us look the following simulation.

A test audio signal with a frequency of 16 kHz and a sampling rate of 44.1 kHz is generated using the formula

$$x(n) = \sin\left(\frac{2\pi \times 16,000n}{441,000}\right)$$

If we use an upsampling factor of 4, then the bandwidth would increase to 88.2 kHz. Based on the audio frequency of 16 kHz, the original Nyquist frequency of 22.05 kHz, and the new sampling rate of 176.4 kHz, we can determine the filter length as

$$\Delta f = \frac{22.05 - 16}{176.4} = 0.0343$$

Using the Hamming window for FIR filter design leads to

$$N = \frac{3.3}{\Delta f} = 96.2$$

We choose $N = 97$. The cutoff frequency therefore is

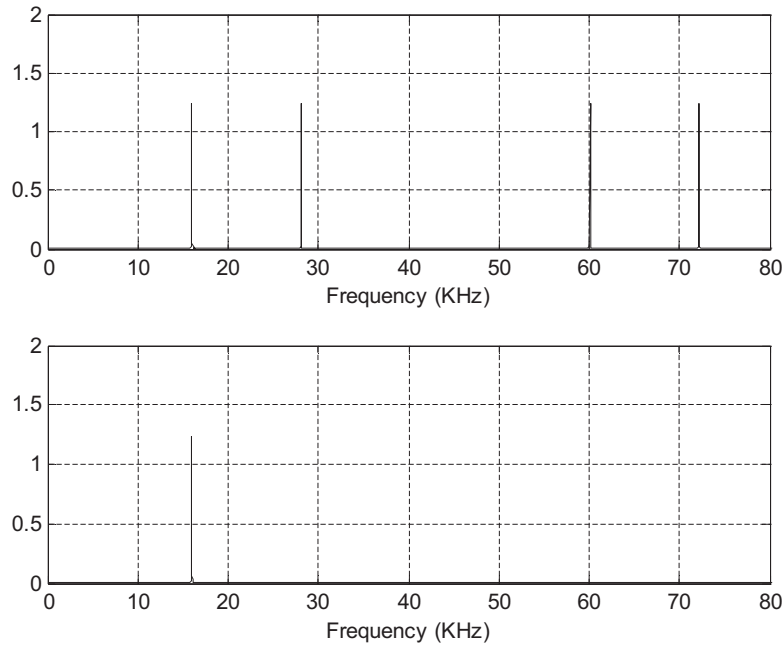
$$f_c = \frac{16 + 22.05}{2} = 19.025 \text{ kHz}$$

The spectrum of the interpolated audio test signal is shown in [Figure 12.12](#), where the top plot illustrates that after the upsampling, the audio test signal has a frequency of 16 kHz, along with image frequencies coming from $44.1 - 16 = 28.1$ kHz, $44.1 + 16 = 60.1$ kHz, $88.2 - 16 = 72.2$ kHz, and so on. The bottom graph describes the spectrum after the interpolation filter. From lowpass FIR filtering, an interpolated audio signal with a frequency of 16 kHz is observed.

Let us examine the corresponding process in the time domain, as shown in [Figure 12.13](#). The upper left plot shows the original samples. The upper right plot describes the upsampled signals. The lower left plot shows the signals after the upsampling process and digital interpolation filter. Finally, the lower right plot shows the sample-and-hold signals after DAC. Clearly, we can easily design a reconstruction filter to smooth the sample-and-hold signals and obtain the original audio test signal. The advantage of reducing hardware is illustrated. The MATLAB implementation can be seen in Program 12.4.

Program 12.4. MATLAB program for CD player example.

```
close all; clear all; clc
% Generate the 2048 samples with fs = 44100 Hz
fs= 44100;           % Original sampling rate
T=1/fs;              % Sampling period
```

**FIGURE 12.12**

(Top) The spectrum after upsampling, and (bottom) the spectrum after applying the interpolation filter.

```

N=2048;                % Number of samples
L=4;
fsL=fs*L;              % Upsampling rate
%Upsampling filter (see Chapter 7 for FIR filter design)
Bup=firwd(97,1,2*19025*pi/fsL,0.4);
n=0:1:N-1;            % Generate the time indices
x=5*sin(2*pi*16000*n*T); % Generate the test signal
% Upsampling by a factor of L
w=zeros(1,L*N);
for n=0:1:N-1
    w(L*n+1)=x(n+1);
end
NL=length(w);          % Number of the upsampled data
W=2*abs(fft(w,NL))/NL; W(1)=W(1)/2; % Compute the one-sided
                                % amplitude spectrum
f=[0:1:NL/2-1]*fs*L/NL; % Map the frequency index to its frequency in Hz
f=f/1000;              % Convert to KHz
% Interpolation
y=filter(Bup,1,w);      % Perform the interpolation filter
Y=2*abs(fft(y,NL))/NL; Y(1)=Y(1)/2; % Compute the one-sided
                                % amplitude spectrum

```

```

subplot(2,1,1);plot(f,W(1:1:NL/2));grid;
xlabel('Frequency (kHz)'); axis([0 f(length(f)) 0 2]);
subplot(2,1,2);plot(f,Y(1:1:NL/2));grid;
xlabel('Frequency (kHz)');axis([0 f(length(f)) 0 2]);
figure
subplot(2,2,1);stem(x(21:30));grid
xlabel('Number of Samples');ylabel('x(n)');
subplot(2,2,2);stem(w(81:120));grid
xlabel('Number of Samples'); ylabel('w(n)');
subplot(2,2,3);stem(y(81:120));grid
xlabel('Number of Samples'); ylabel('y(n)')
subplot(2,2,4);stairs([80:1:119]*1000*T,y(81:120));grid
xlabel('Time (ms)'); ylabel('y(t)')

```

12.1.5 Multistage Decimation

The multistage approach for downsampling rate conversion can be used to dramatically reduce the anti-aliasing filter length. Figure 12.14 describes a two-stage decimator.

As shown in Figure 12.14, a total decimation factor is $M = M_1 \times M_2$. Here, even though we develop a procedure for a two-stage case, a similar principle can be applied to general multistage cases.

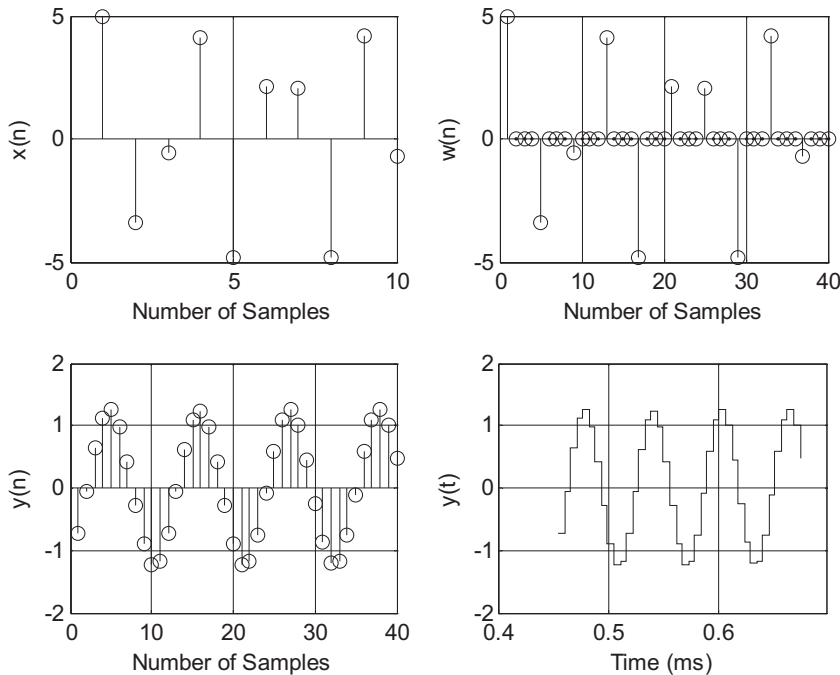


FIGURE 12.13

Plots of signals at each stage according to the block diagram in Figure 12.11.

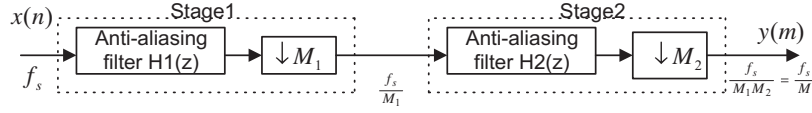


FIGURE 12.14

Multistage decimation.

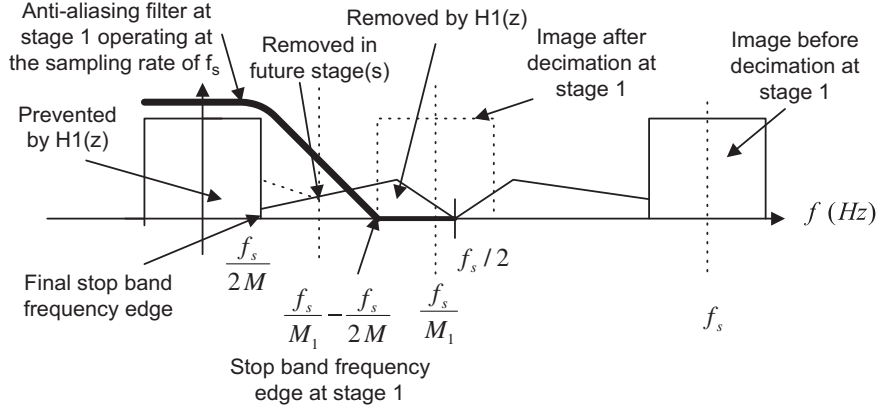


FIGURE 12.15

Stopband frequency edge for the anti-aliasing filter at stage 1 for two-stage decimation.

Using the two-stage decimation in Figure 12.15, the final Nyquist limit is $\frac{f_s}{2M}$ after final down-sampling. So our useful information bandwidth should stop at the frequency edge of $\frac{f_s}{2M}$. Next, we need to determine the stop frequency edge for the anti-aliasing lowpass filter at stage 1 before the first decimation process begins. This stop frequency edge is actually the lower frequency edge of the first image replica centered at the sampling frequency of $\frac{f_s}{M_1}$ after the stage 1 decimation. This lower frequency edge of the first image replica is then determined by

$$\frac{f_s}{M_1} - \frac{f_s}{2M}$$

After downsampling, we expect that some frequency components from $\frac{f_s}{2M_1}$ to $\frac{f_s}{M_1} - \frac{f_s}{2M}$ to be folded over to the frequency band between $\frac{f_s}{2M}$ and $\frac{f_s}{2M_1}$. However, these aliased frequency components do not affect the final useful band between 0 Hz to $\frac{f_s}{2M}$ and will be removed by the anti-aliasing filter(s) in the future stage(s). As illustrated in Figure 12.15, any frequency components beyond the edge $\frac{f_s}{M_1} - \frac{f_s}{2M}$

can fold over into the final useful information band to create aliasing distortion. Therefore, we can use this frequency as the lower stop frequency edge of the anti-aliasing filter to prevent the aliasing distortion at the final stage. The upper stopband edge (Nyquist limit) for the anti-image filter at stage 1 is clearly $\frac{f_s}{2}$, since the filter operates at f_s samples per second. So the stopband frequency range at stage 1 is therefore from $\frac{f_s}{M_1} - \frac{f_s}{2M}$ to $\frac{f_s}{2}$. The aliasing distortion, introduced into the frequency band from $\frac{f_s}{2M}$ to $\frac{f_s}{2M_1}$, will be filtered out after future decimation stage(s).

Similarly, for stage 2, the lower frequency edge of the first image developed after stage 2 down-sampling is

$$\frac{f_s}{M_1 M_2} - \frac{f_s}{2M} = \frac{f_s}{2M}$$

As is evident in our two-stage scheme, the stopband frequency range for the second anti-aliasing filter at stage 2 should be from $\frac{f_s}{2M}$ to $\frac{f_s}{2M_1}$.

We summarize the specifications for the two-stage decimation as follows:

Filter requirements for stage 1:

- Passband frequency range = 0 to f_p
- Stopband frequency range = $\frac{f_s}{M_1} - \frac{f_s}{2M}$ to $\frac{f_s}{2}$
- Passband ripple = $\delta_p/2$, where δ_p is the combined absolute ripple on the passband
- Stopband attenuation = δ_s

Filter requirements for stage 2:

- Passband frequency range = 0 to f_p
- Stopband frequency range = $\frac{f_s}{M_1 \times M_2} - \frac{f_s}{2M}$ to $\frac{f_s}{2M_1}$
- Passband ripple = $\delta_p/2$, where δ_p is the combined absolute ripple on the passband
- Stopband attenuation = δ_s

Example 12.4 illustrates the two-stage decimator design.

EXAMPLE 12.4

Determine the anti-aliasing FIR filter lengths and cutoff frequencies for the two-stage decimator with the following specifications and the block diagram in Figure 12.16A:

Original sampling rate: $f_s = 240$ kHz

Audio frequency range: 0–3,400 Hz

Passband ripple: $\delta_p = 0.05$ (absolute)

Stopband attenuation: $\delta_s = 0.005$ (absolute)

FIR filter design using the window method

New sampling rate: $f_{sM} = 8$ kHz

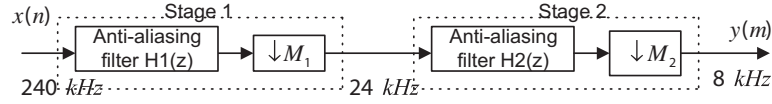


FIGURE 12.16A

Multistage decimation in Example 12.4.

Solution:

$$M = \frac{240 \text{ kHz}}{8 \text{ kHz}} = 30 = 10 \times 3$$

We choose $M_1 = 10$ and $M_2 = 3$; there could be other choices. Figure 12.16B shows the block diagram and filter frequency specifications.

Filter specifications for $H_1(z)$:

Passband frequency range: 0–3,400 Hz

Passband ripples: $0.05/2 = 0.025$ ($\delta_p \text{ dB} = 20 \log_{10}(1 + \delta_p) = 0.212 \text{ dB}$)

Stopband frequency range: 20,000–120,000 Hz

Stopband attenuation: 0.005 , $\delta_s \text{ dB} = -20 \times \log_{10}(\delta_s) = 46 \text{ dB}$

Filter type: FIR, Hamming window

Note that the lower stopband edge can be determined as

$$f_{\text{stop}} = \frac{f_s}{M_1} - \frac{f_s}{2 \times M} = \frac{240,000}{10} - \frac{240,000}{2 \times 30} = 20,000 \text{ Hz}$$

$$\Delta f = \frac{f_{\text{stop}} - f_{\text{pass}}}{f_s} = \frac{20,000 - 3,400}{240,000} = 0.06917$$

The length of the filter and the cutoff frequency can be determined by

$$N = \frac{3.3}{\Delta f} = 47.7$$

We choose $N = 49$, and

$$f_c = \frac{f_{\text{pass}} + f_{\text{stop}}}{2} = \frac{20,000 + 3,400}{2} = 11,700 \text{ Hz}$$

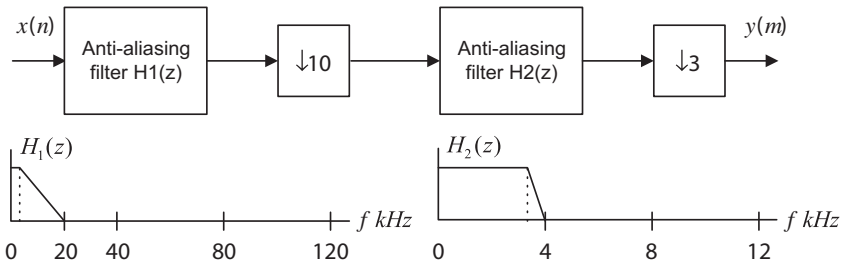


FIGURE 12.16B

Filter frequency specifications for Example 12.4.

Filter specifications for $H_2(z)$:

Passband frequency range: 0–3,400 Hz
 Passband ripples: $0.05/2 = 0.025$ (0.212 dB)
 Stopband frequency range: 4,000–12,000 Hz
 Stopband attenuation: 0.005 , $\delta_s \text{ dB} = 46$ dB
 Filter type: FIR, Hamming window

Note that

$$\Delta f = \frac{f_{\text{stop}} - f_{\text{pass}}}{f_{sM1}} = \frac{4,000 - 3,400}{24,000} = 0.025.$$

The length of the filter and the cutoff frequency can be determined by

$$N = \frac{3.3}{\Delta f} = 132$$

We choose $N = 133$, and

$$f_c = \frac{f_{\text{pass}} + f_{\text{stop}}}{2} = \frac{4,000 + 3,400}{2} = 3,700 \text{ Hz}$$

The reader can verify this case by using only one stage with a decimation factor of $M = 30$. Using the Hamming window for the FIR filter, the resulting number of taps is 1,321, and the cutoff frequency is 3,700 Hz. Thus, such a filter requires a huge number of computations and causes a large delay during implementation compared with the two-stage case.

The multistage scheme is very helpful for sampling rate conversion between audio systems. For example, to convert CD audio at a sampling rate of 44.1 kHz to MP3 or Digital Audio Tape (DAT), in which the sampling rate of 48 kHz is used, the conversion factor $L/M = 48/44.1 = 160/147$ is required. Using the single stage scheme may cause impractical FIR filter sizes for interpolation and downsampling. However, since $L/M = 160/147 = (4/3)(8/7)(5/7)$, we may design an efficient three-stage system, in which stages 1, 2, and 3 use the conversion factors $L/M = 8/7$, $L/M = 5/7$, and $L/M = 4/3$, respectively.

12.2 POLYPHASE FILTER STRUCTURE AND IMPLEMENTATION

Due to the nature of the decimation and interpolation processes, polyphase filter structures can be developed to efficiently implement the decimation and interpolation filters (using fewer multiplications and additions). As we will explain, these filters are all-pass filters with different phase shifts (Proakis and Manolakis, 1996), thus we call them *polyphase filters*.

Here, we skip their derivations and illustrate implementations of decimation and interpolation using simple examples. Consider the interpolation process shown in Figure 12.17, where $L = 2$.

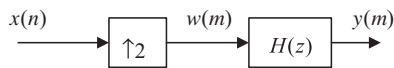


FIGURE 12.17

Upsampling by a factor of 2 and a four-tap interpolation filter.

Table 12.1 Results of the Direct Interpolation Process in Figure 12.17 (8 multiplications and 6 additions for processing each input sample $x(n)$)

n	$x(n)$	m	$w(m)$	$y(m)$
$n = 0$	$x(0)$	$m = 0$	$w(0) = x(0)$	$y(0) = h(0)x(0)$
		$m = 1$	$w(1) = 0$	$y(1) = h(1)x(0)$
$n = 1$	$x(1)$	$m = 2$	$w(2) = x(1)$	$y(2) = h(0)x(1) + h(2)x(0)$
		$m = 3$	$w(3) = 0$	$y(3) = h(1)x(1) + h(3)x(0)$
$n = 2$	$x(2)$	$m = 4$	$w(4) = x(2)$	$y(4) = h(0)x(2) + h(2)x(1)$
		$m = 5$	$w(5) = 0$	$y(5) = h(1)x(2) + h(3)x(1)$
...

We assume that the FIR interpolation filter has four taps, shown as

$$H(z) = h(0) + h(1)z^{-1} + h(2)z^{-2} + h(3)z^{-3}$$

and the filter output is

$$y(m) = h(0)w(m) + h(1)w(m-1) + h(2)w(m-2) + h(3)w(m-3)$$

For the purpose of comparison, the direct interpolation process shown in Figure 12.17 is summarized in Table 12.1, where $w(m)$ is the upsampled signal and $y(m)$ the interpolated output. Processing each input sample $x(n)$ requires applying the difference equation twice to obtain $y(0)$ and $y(1)$. Hence, for this example, we need eight multiplications and six additions.

The output results in Table 12.1 can be easily obtained by using the polyphase filters shown in Figure 12.18.

In general, there are L polyphase filters. With a designed interpolation filter $H(z)$ of N taps, we can determine each bank of filter coefficients as follows:

$$\rho_k(n) = h(k + nL) \quad \text{for } k = 0, 1, \dots, L-1 \quad \text{and } n = 0, 1, \dots, \frac{N}{L} - 1 \quad (12.12)$$

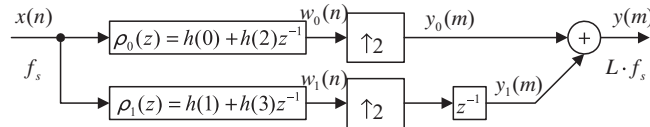
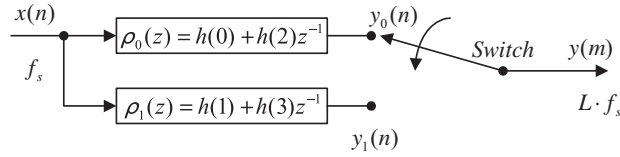


FIGURE 12.18

Polyphase filter implementation for the interpolation in Figure 12.17 (4 multiplications and 3 additions for processing each input sample $x(n)$).

**FIGURE 12.19**

Commutative model for the polyphase interpolation filter.

For our example, $L = 2$ and $N = 4$, we have $L - 1 = 1$, and $N/L - 1 = 1$, respectively. Hence, there are two filter banks, $\rho_0(z)$ and $\rho_1(z)$, each having a length of 2, as illustrated in Figure 12.18. When $k = 0$ and $n = 1$, the upper limit of time index required for $h(k + nL)$ is $k + nL = 0 + 1 \times 2 = 2$. When $k = 1$ and $n = 1$, the upper limit of the time index for $h(k + nL)$ is 3. Hence, the first filter $\rho_0(z)$ has the coefficients $h(0)$ and $h(2)$. Similarly, the second filter $\rho_1(z)$ has coefficients $h(1)$ and $h(3)$. In fact, the filter coefficients of $\rho_0(z)$ are a decimated version of $h(n)$ starting at $k = 0$, while the filter coefficients of $\rho_1(z)$ are a decimated version of $h(n)$ starting at $k = 1$, and so on.

As shown in Figure 12.18, we can reduce the computational complexity from eight multiplications and six additions down to four multiplications and three additions for processing each input sample $x(n)$. Generally, the computation can be reduced by a factor of L as compared with the direct process.

The commutative model for the polyphase interpolation filter is given in Figure 12.19.

EXAMPLE 12.5

Verify $y(1)$ in Table 12.1 using the polyphase filter implementation in Figures 12.18 and 12.19, respectively.

Solution:

Applying the polyphase interpolation filter as shown in Figure 12.18 leads to

$$\begin{aligned} w_0(n) &= h(0)x(n) + h(2)x(n-1) \\ w_1(n) &= h(1)x(n) + h(3)x(n-1) \end{aligned}$$

When $n = 0$,

$$\begin{aligned} w_0(0) &= h(0)x(0) \\ w_1(0) &= h(1)x(0) \end{aligned}$$

After interpolation, we have

$$y_0(m) : w_0(0) \quad 0 \quad \dots$$

and

$$y_1(m) : 0 \quad w_1(0) \quad 0 \quad \dots$$

Note: there is a unit delay for the second filter bank. Hence

$$\begin{aligned} m = 0, \quad y_0(0) &= h(0)x(0), \quad y_1(0) = 0 \\ m = 1, \quad y_0(1) &= 0, \quad y_1(1) = h(1)x(0) \end{aligned}$$

Combining two channels, we finally get

$$m = 0, y(0) = y_0(0) + y_1(0) = h(0)x(0)$$

$$m = 1, y(1) = y_0(1) + y_1(1) = h(1)x(0)$$

Therefore, $y(1)$ matches that in the direct interpolation process given in Table 12.1.

Applying the polyphase interpolation filter using the commutative model in Figure 12.19, we have

$$y_0(n) = h(0)x(n) + h(2)x(n-1)$$

$$y_1(n) = h(1)x(n) + h(3)x(n-1)$$

When $n = 0$,

$$m = 0, y(0) = y_0(0) = h(0)x(0) + h(2)x(-1) = h(0)x(0)$$

$$m = 1, y(1) = y_1(0) = h(1)x(0) + h(3)x(-1) = h(1)x(0)$$

Clearly, $y(1) = h(1)x(0)$ matches the $y(1)$ result in Table 12.1.

Next, we will briefly explain the properties of polyphase filters (that is, they have all-pass gain and possible different phases). Each polyphase filter $\rho_k(n)$ operating at the original sampling rate f_s (assuming 8 kHz) is a downsampled version of the interpolation filter $h(n)$ operating at the upsampling rate Lf_s (32 kHz assuming an interpolation factor of $L = 4$). Considering that the designed interpolation FIR filter coefficients $h(n)$ are the impulse response sequence with a flat frequency spectrum up to a bandwidth of $f_s/2$ (assuming a bandwidth of 4 kHz with a perfect flat frequency magnitude response, theoretically) at a sampling rate of Lf_s (32 kHz), we then downsample $h(n)$ to obtain polyphase filters by a factor of $L = 4$ and operate them at a sampling rate of f_s (8 kHz).

The Nyquist frequency after downsampling should be $(Lf_s/2)/L = f_s/2$ (4 kHz); at the same time, each downsampled sequence $\rho_k(n)$ operating at f_s (8 kHz) has a flat spectrum up to $f_s/2$ (4 kHz) due to the $f_s/2$ (4 kHz) bandlimited sequence of $h(n)$ at the sampling rate of Lf_s (32 kHz). Hence, all of the polyphase filters are all-pass filters. Since each polyphase $\rho_k(n)$ filter has different coefficients, each may have a different phase. Therefore, these polyphase filters are the all-pass filters with possible different phases, theoretically.

Next, consider the decimation process in Figure 12.20. Assuming a three-tap decimation filter, we have

$$H(z) = h(0) + h(1)z^{-1} + h(2)z^{-2}$$

$$w(n) = h(0)x(n) + h(1)x(n-1) + h(2)x(n-2)$$

The direct decimation process is shown in Table 12.2 for the purpose of comparison. Obtaining each output $y(m)$ requires processing filter difference equations twice, resulting in six multiplications and four additions for this particular example.

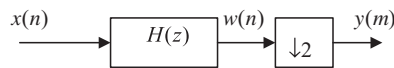
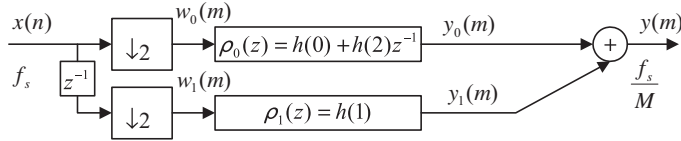


FIGURE 12.20

Decimation by a factor of 2 and a three-tap anti-aliasing filter.

Table 12.2 Results of the Direct Decimation Process in Figure 12.20 (6 multiplications and 4 additions for obtaining each output $y(m)$)

n	$x(n)$	$w(n)$	m	$y(m)$
$n = 0$	$x(0)$	$w(0) = h(0)x(0)$	$m = 0$	$y(0) = h(0)x(0)$
$n = 1$	$x(1)$	$w(1) = h(0)x(1) + h(1)x(0)$ discard		
$n = 2$	$x(2)$	$w(2) = h(0)x(2) + h(1)x(1) + h(2)x(0)$	$m = 1$	$y(1) = h(0)x(2) + h(1)x(1) + h(2)x(0)$
$n = 3$	$x(3)$	$w(3) = h(0)x(3) + h(1)x(2) + h(2)x(1)$ discard		
$n = 4$	$x(5)$	$w(4) = h(0)x(4) + h(1)x(3) + h(2)x(2)$	$m = 2$	$y(2) = h(0)x(4) + h(1)x(3) + h(2)x(2)$
$n = 5$	$x(6)$	$w(5) = h(0)x(5) + h(1)x(4) + h(2)x(3)$ discard		
...

**FIGURE 12.21**

Polyphase filter implementation for the decimation in Figure 12.20. (3 multiplications and 1 addition for obtaining each output $y(m)$).

The efficient way to implement a polyphase filter is given in Figure 12.21.

Similarly, there are M polyphase filters. With the designed decimation filter $H(z)$ of N taps, we can obtain filter bank coefficients by

$$\rho_k(n) = h(k + nM) \quad \text{for } k = 0, 1, \dots, M-1 \quad \text{and} \quad n = 0, 1, \dots, \frac{N}{M} - 1 \quad (12.13)$$

For our example, we see that $M-1 = 1$ and $N/M - 1 = 1$ (rounded up). Thus, we have two filter banks. Since $k = 0$ and $n = 1$, $k + nM = 0 + 1 \times 2 = 2$. The time index upper limit required for $h(k + nM)$ is 2 for the first filter bank $\rho_0(z)$. Hence $\rho_0(z)$ has filter coefficients $h(0)$ and $h(2)$.

However, when $k = 1$ and $n = 1$, $k + nM = 1 + 1 \times 2 = 3$, the time index upper limit required for $h(k + nM)$ is 3 for the second filter bank, and the corresponding filter coefficients are required to be $h(1)$ and $h(3)$. Since our direct interpolation filter $h(n)$ does not contain the coefficient $h(3)$, we set $h(3) = 0$ to get the second filter bank with one tap only, as shown in Figure 12.21. Also as shown in that figure, achieving each $y(m)$ requires three multiplications and one addition. In general, the number of multiplications can be reduced by a factor of M .

The commutative model for the polyphase decimator is shown in Figure 12.22.

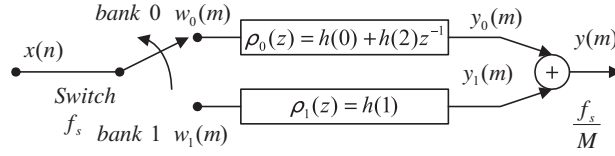


FIGURE 12.22

Commutative model for the polyphase decimation filter.

EXAMPLE 12.6

Verify $y(1)$ in Table 12.2 using the polyphase decimation filter implementation in Figure 12.21.

Solution:

Using Figure 12.21, we write the difference equations as

$$y_0(m) = h(0)w_0(m) + h(2)w_0(m-1)$$

$$y_1(m) = h(1)w_1(m)$$

Assuming $n = 0$, $n = 1$, $n = 2$, and $n = 3$, we have the inputs as $x(0)$, $x(1)$, $x(2)$ and $x(3)$, and

$$w_0(m) : x(0) \quad x(2) \quad \dots$$

Delaying $x(n)$ by one sample and decimating it by a factor of 2 leads to

$$w_1(m) : 0 \quad x(1) \quad x(3) \quad \dots$$

Hence, applying the filter banks yields the following: For $m = 0$, we have inputs for each filter as

$$w_0(0) = x(0) \text{ and } w_1(0) = 0$$

Then

$$y_0(0) = h(0)w_0(0) + h(2)w_0(-1) = h(0)x(0)$$

$$y_1(0) = h(1)w_1(0) = h(1) \times 0 = 0$$

Combining two channels, we obtain

$$y(1) = y_0(1) + y_1(1) = h(0)x(0) + 0 = h(0)x(0)$$

For $m = 1$, we get inputs for each filter as

$$w_0(1) = x(2) \quad \text{and} \quad w_1(1) = x(1)$$

Then

$$y_0(1) = h(0)w_0(1) + h(2)w_0(0) = h(0)x(2) + h(2)x(0)$$

$$y_1(1) = h(1)w_1(1) = h(1)x(1)$$

Combining two channels leads to

$$y(1) = y_0(1) + y_1(1) = h(0)x(2) + h(2)x(0) + h(1)x(1)$$

We note that $y(1)$ is the same as that shown in Table 12.2. Similar analysis can be done for the commutative model shown in Figure 12.22.

Program 12.5 demonstrates the polyphase implementation of decimation. The program is modified based on Program 12.1.

Program 12.5. Decimation using polyphase implementation.

```
close all; clear all;
% Downsampling filter (see Chapter 7 for FIR filter design)
B =[0.00074961181416 0.00247663033476 0.00146938649416 -0.00440446121505...
-0.00910635730662 0.00000000000000 0.02035676831506 0.02233710562885...
-0.01712963672810 -0.06376620649567 -0.03590670035210 0.10660980550088...
0.29014909103794 0.37500000000000 0.29014909103794 0.10660980550088...
-0.03590670035210 -0.06376620649567 -0.01712963672810 0.02233710562885...
0.02035676831506 0.00000000000000 -0.00910635730662 -0.00440446121505...
0.00146938649416 0.00247663033476 0.00074961181416];
% Generate 2048 samples
fs=8000; % Sampling rate
N=2048; % Number of samples
M=2; % Downsample factor
n=0:1:N-1;
x=5*sin(n*pi/4)+cos(5*n*pi/8);
% Compute the single-sided amplitude spectrum
% AC component will be doubled, and DC component will be kept the
% same value
X=2*abs(fft(x,N))/N;X(1)=X(1)/2;
% Map the frequency index up to the folding frequency in Hz
f=[0:1:N/2-1]*fs/N;
% Decimation
w0=x(1:M:N); p0=B(1:2:length(B)); % Downsampling
w1=filter([0 1],1,x); % Delay one sample
w1=w1(1:M:N); p1=B(2:M:length(B)) % Downsampling
y=filter(p0,1,w0)+filter(p1,1,w1);
NM=length(y); % Length of the downsampled data
% Compute the single-sided amplitude spectrum for the downsampled
% signal
Y=2*abs(fft(y,NM))/NM;Y(1)=Y(1)/2;
% Map the frequency index to the frequency in Hz
fsM=[0:1:NM/2-1]*(fs/M)/NM;
% Plot spectra
subplot(2,1,1);plot(f,X(1:1:N/2));grid; xlabel('Frequency (Hz)');
subplot(2,1,2);plot(fsM,Y(1:1:NM/2));grid; xlabel('Frequency (Hz)');
```

Program 12.6 demonstrates polyphase implementation of interpolation using the information in Program 12.2.

Program 12.6. Interpolation using polyphase implementation.

```
close all; clear all
% Upsampling filter (see Chapter 7 for FIR filter design)
B =[ -0.00012783931504 0.00069976044649 0.00123831516738 0.00100277549136...
-0.00025059018468 -0.00203448515158 -0.00300830295487 -0.00174101657599...
```

```

0.00188598835011 0.00578414933758 0.00649330625041 0.00177982369523...
-0.00670672686935 -0.01319379342716 -0.01116855281442 0.00123034314117...
0.01775600060894 0.02614700427364 0.01594155162392 -0.01235169936557...
-0.04334322148505 -0.05244745563466 -0.01951094855292 0.05718573279009...
0.15568416401644 0.23851539047347 0.27083333333333 0.23851539047347...
0.15568416401644 0.05718573279009 -0.01951094855292 -0.05244745563466...
-0.04334322148505 -0.01235169936557 0.01594155162392 0.02614700427364...
0.01775600060894 0.00123034314117 -0.01116855281442 -0.01319379342716...
-0.00670672686935 0.00177982369523 0.00649330625041 0.00578414933758...
0.00188598835011 -0.00174101657599 -0.00300830295487 -0.00203448515158...
-0.00025059018468 0.00100277549136 0.00123831516738 0.00069976044649...
-0.00012783931504];
% Generate 2048 samples with fs=8000 Hz
fs=8000; % Sampling rate
N=2048; % Number of samples
L = 3; % Upsampling factor
n=0:1:N-1;
x=5*sin(n*pi/4)+cos(5*n*pi/8);
p0=B(1:L:length(B)); p1=B(2:L:length(B)); p2=B(3:L:length(B));
% Interpolation
w0=filter(p0,1,x);
w1=filter(p1,1,x);
w2=filter(p2,1,x);
y0=zeros(1,L*N);y0(1:L:length(y0))=w0;
y1=zeros(1,L*N);y1(1:L:length(y1))=w1;
y1=filter([0 1],1,y1);
y2=zeros(1,L*N);y2(1:L:length(y2))=w2;
y2=filter([0 0 1],1,y2);
y=y0+y1+y2; % Interpolated signal
NL = length(y); % Length of the upsampled data
X=2*abs(fft(x,N))/N;X(1)=X(1)/2; % Compute the one-sided amplitude
% spectrum
f=[0:1:N/2-1]*fs/N; % Map the frequency index to the frequency (Hz)
Y=2*abs(fft(y,NL))/NL;Y(1)=Y(1)/2; % Compute the one-sided amplitude
% spectrum
fsL=[0:1:NL/2-1]*fs*L/NL; % Map the frequency index to the frequency (Hz)
subplot(2,1,1);plot(f,X(1:1:N/2));grid; xlabel('Frequency (Hz)');
subplot(2,1,2);plot(fsL,Y(1:1:NL/2));grid; xlabel('Frequency (Hz)');

```

Note that wavelet transform and subband coding are also topics in the area of multirate signal processing. We will pursue these subjects in Chapter 13.

12.3 OVERSAMPLING OF ANALOG-TO-DIGITAL CONVERSION

Oversampling of the analog signal has become more popular in the DSP industry to improve the resolution of analog-to-digital conversion (ADC). Oversampling uses a sampling rate that is much higher than the Nyquist rate. We can define an oversampling ratio as

$$\frac{f_s}{2f_{\max}} \gg 1 \quad (12.14)$$

The benefits of an oversampling ADC include

1. helping to design a simple analog anti-aliasing filter before ADC, and
2. reducing the ADC noise floor with possible noise shaping so that a low-resolution ADC can be used.

12.3.1 Oversampling and Analog-to-Digital Conversion Resolution

To begin developing the relation between oversampling and ADC resolution, we first summarize the regular ADC and some useful definitions discussed in Chapter 2:

$$\text{Quantization noise power} = \sigma_q^2 = \frac{\Delta^2}{12} \quad (12.15)$$

$$\text{Quantization step} = \Delta = \frac{A}{2^n} \quad (12.16)$$

where A = full range of the analog signal to be digitized and n = number of bits per sample (ADC resolution).

Substituting Equation (12.16) into Equation (12.15), we have

$$\text{Quantization noise power} = \sigma_q^2 = \frac{A^2}{12} \times 2^{-2n} \quad (12.17)$$

The power spectral density of the quantization noise with an assumption of uniform probability distribution is shown in Figure 12.23. Note that this assumption is true for quantizing a uniformly

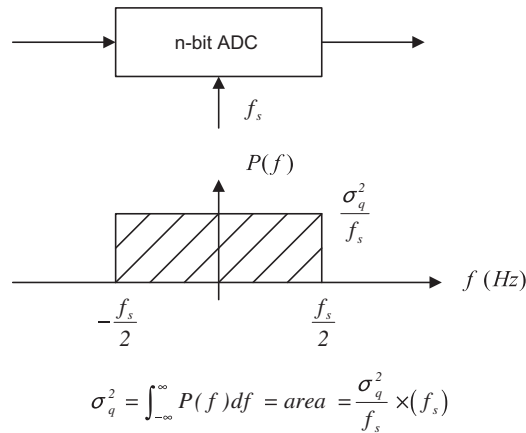


FIGURE 12.23

Regular ADC system.

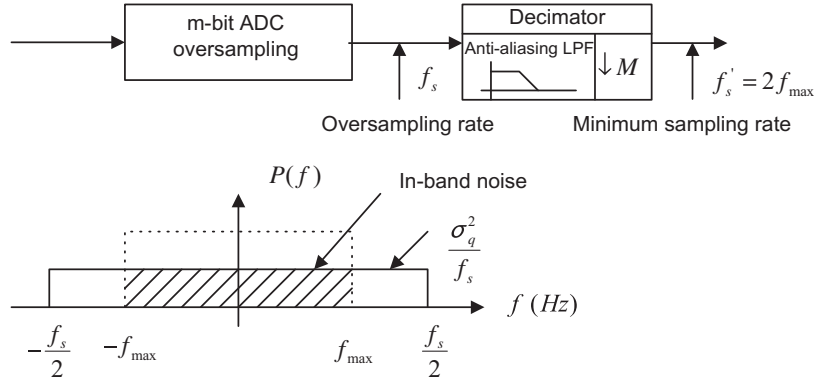


FIGURE 12.24

Oversampling ADC system.

distributed signal in a full range with a sufficiently long duration. It is not generally true in practice. See research papers authored by Lipshitz et al. (1992) and Maher (1992). However, using the assumption will guide us to some useful results for oversampling systems.

The quantization noise power is the area obtained from integrating the power spectral density function in the range of $-f_s/2$ to $f_s/2$. Now let us examine the oversampling ADC, where the sampling rate is much larger than that of the regular ADC; that is $f_s \gg 2f_{\max}$. The scheme is shown in Figure 12.24.

As we can see, oversampling can reduce the level of noise power spectral density. After the decimation process with the decimation filter, only a portion of quantization noise power in the range from $-f_{\max}$ and f_{\max} is kept in the DSP system. We call this an *in-band frequency range*.

In Figure 12.24, the shaded area, which is the quantization noise power, is given by

$$\text{Quantization noise power} = \int_{-\infty}^{\infty} P(f) df = \frac{2f_{\max}}{f_s} \times \sigma_q^2 = \frac{2f_{\max}}{f_s} \cdot \frac{A^2}{12} \times 2^{-2m} \quad (12.18)$$

Assuming that the regular ADC shown in Figure 12.23 and the oversampling ADC shown in Figure 12.24 are equivalent, we set their quantization noise powers to be the same to obtain

$$\frac{A^2}{12} \cdot 2^{-2n} = \frac{2f_{\max}}{f_s} \cdot \frac{A^2}{12} \times 2^{-2m} \quad (12.19)$$

Equation (12.19) leads to two useful equations for applications:

$$n = m + 0.5 \times \log_2 \left(\frac{f_s}{2f_{\max}} \right) \text{ and} \quad (12.20)$$

$$f_s = 2f_{\max} \times 2^{2(n-m)} \quad (12.21)$$

where

f_s = sampling rate in the oversampling DSP system

f_{\max} = maximum frequency of the analog signal

m = number of bits per sample in the oversampling DSP system

n = number of bits per sample in the regular DSP system using the minimum sampling rate

From Equation (12.20) and given the number of bits (m) used in the oversampling scheme, we can determine the number of bits per sample equivalent to the regular ADC. On the other hand, given the number of bits in the oversampling ADC, we can determine the required oversampling rate so that the oversampling ADC is equivalent to the regular ADC with the larger number of bits per sample (n). Let us look at the following examples.

EXAMPLE 12.7

Given an oversampling audio DSP system with maximum audio input frequency of 20 kHz and ADC resolution of 14 bits, determine the oversampling rate to improve the ADC resolution to 16-bit resolution.

Solution:

Based on the specifications, we have

$$f_{\max} = 20 \text{ kHz}, m = 14 \text{ bits} \quad \text{and} \quad n = 16 \text{ bits}$$

Using Equation (12.21) leads to

$$f_s = 2f_{\max} \times 2^{2(n-m)} = 2 \times 20 \times 2^{2(16-14)} = 640 \text{ kHz}$$

Since $f_s/(2f_{\max}) = 2^4$, we see that each doubling of the minimum sampling rate ($2f_{\max} = 40 \text{ kHz}$) will increase the resolution by a half bit.

EXAMPLE 12.8

Given an oversampling audio DSP system with a maximum audio input frequency of 4 kHz, and ADC resolution of 8 bits, an a sampling rate of 80 MHz, determine the equivalent ADC resolution.

Solution:

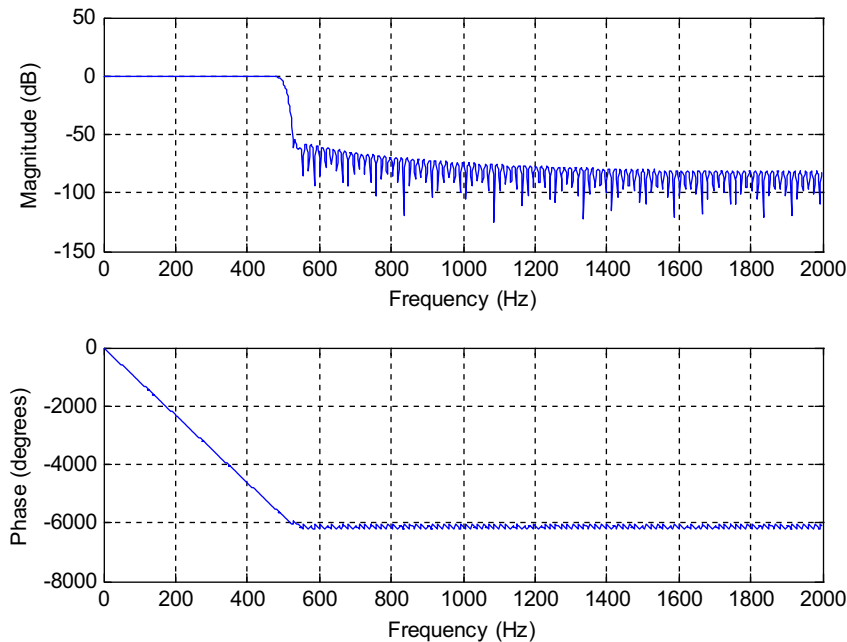
Since $f_{\max} = 4 \text{ kHz}$, $f_s = 80 \text{ kHz}$, and $m = 8 \text{ bits}$, applying Equation (12.20) yields

$$n = m + 0.5 \times \log_2 \left(\frac{f_s}{2f_{\max}} \right) = 8 + 0.5 \times \log_2 \left(\frac{80,000 \text{ kHz}}{2 \times 4 \text{ kHz}} \right) \approx 15 \text{ bits}$$

The MATLAB program shown in Program 12.7 validates the oversampling technique. We consider the following signal,

$$x(t) = 1.5 \sin(2\pi \times 150t) + 0.9 \sin(2\pi \times 175t + \pi/6) + 0.6 \sin(2\pi \times 200t + \pi/4) \quad (12.22)$$

with a regular sampling rate of 1 kHz. The oversampling rate is 4 kHz and each sample is quantized using a 3-bit code. The anti-aliasing lowpass filter is designed with a cutoff frequency of $\Omega = 2\pi f_{\max} T = 2\pi \times 500/4,000 = 0.25\pi$ radians. Figure 12.25 shows the frequency responses of

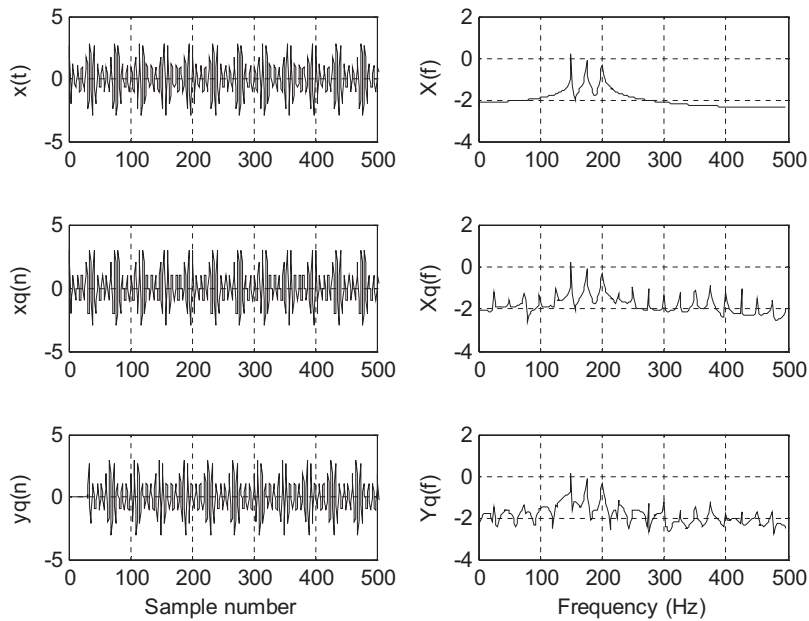
**FIGURE 12.25**

Frequency responses of the designed filter.

the designed filter while Figure 12.26 compares the signals in the time and frequency domains, respectively, where $x(t)$ denotes the continuous version, $x_q(n)$ is the quantized version using a regular sampling rate of 1 kHz, and $y_q(n)$ is the enhanced version using the oversampling system with $L = 4$. The detailed amplitude comparisons are given in Figure 12.27. The measured signal-to-noise ratios (SNRs) are 14.3 dB using the regular sampling system and 21.0 dB using the oversampling system. Since $L = 4$, the achieved signal is expected to have 4-bit quality ($0.5 \times \log_2 4 = 1$ bit improvement). From simulation, we achieve an SNR improvement of approximately 6dB. The improvement will stop when L increases due to the fact that when the sampling increases the quantization error may have correlation with the sinusoidal signal. The degradation performance can be cured using the dithering technique (Tan and Wang, 2011), which is beyond our scope.

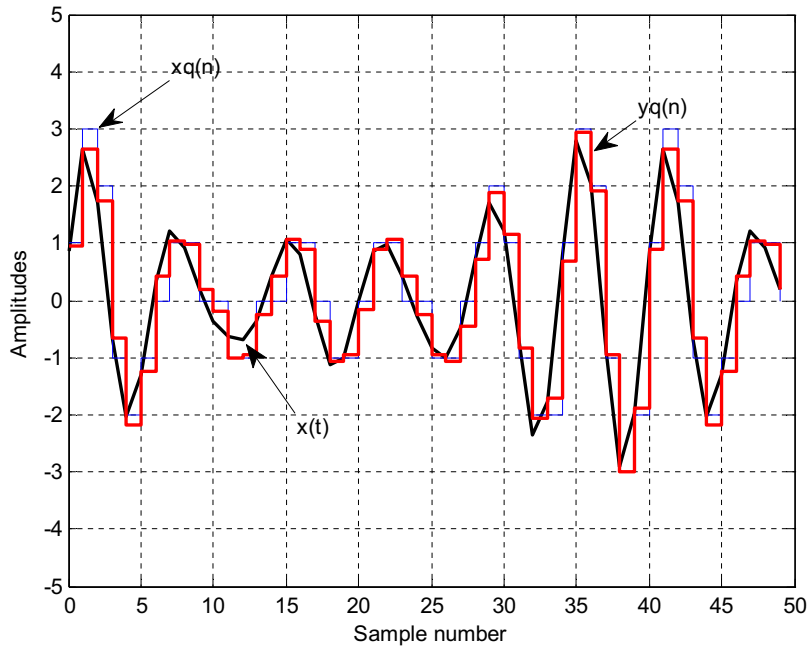
Program 12.7. Oversampling implementation.

```
clear all; close all; clc
ntotal=512;
n=0:ntotal; % Number of samples
L=4; % Oversampling factor
nL=0:ntotal*L; % Number of samples for oversampling
numb=3; % Number of bits
A=2^(numb-1)-1; % Peak value
f1=150; C1=0.5*A; f2=175; C2=A*0.3; f3=200; C3=A*0.2; % Frequencies and amplitudes
```

**FIGURE 12.26**

Signal comparisons in both the time and frequency domains.

```
fmax=500;fs=1000;T=1/fs; % Maximum frequency, sampling rate, sampling period
fsL=L*fs;TL=1/fsL;% Oversampling rate and oversampling period
% Sampling at fs=1000 Hz
x=C1*sin(2*pi*f1*n*T)+C2*sin(2*pi*f2*n*pi/6)+C3*sin(2*pi*f3*T*n+pi/4);
xq=round(x); % Quantized signal at the minimum sampling rate
NN=length(n);
f=[0:ntotal-1]*fs/NN;
M=32*L;nd=M/L; % Delay in terms of samples due to anti-aliasing filtering
B=firwd(2*M+1,1,2*pi*fmax/fsL,0,4); % Anti-aliasing filter design (5% transition
% bandwidth)
figure(1);
freqz(B,1,1000,fsL)
% Oversampling
xx=C1*sin(2*pi*f1*nL*TL)+C2*sin(2*pi*f2*nL*TL+pi/6)+C3*sin(2*pi*f3*nL*TL+pi/4);
xxq=round(xx); % Quantized signal
% Down sampling
y=filter(B,1,xxq);% Anti-aliasing filtering
yd=y(1:L:length(y));% Downsample
figure (2)
subplot(3,2,1);plot(n,x,'k');grid;axis([0 500 -5 5]);ylabel('x(t)')
Ak=2*abs(fft(x))/NN; Ak(1)=Ak(1)/2;
```

**FIGURE 12.27**

Comparisons of continuous, regular sampled, and oversampled signal amplitudes.

```
subplot(3,2,2);plot(f(1:NN/2),log10(Ak(1:NN/2)),'k');grid;ylabel('X(f)'); axis([0 500
-4 2])
subplot(3,2,3);plot(n,xq,'k');grid;axis([0 500 -5 5]);ylabel('xq(n)');
Ak=2*abs(fft(xq))/NN; Ak(1)=Ak(1)/2;
subplot(3,2,4);plot(f(1:NN/2),log10(Ak(1:NN/2)),'k');grid;ylabel('Xq(f)'); axis([0 500
-4 2])
subplot(3,2,5);plot(n,yd,'k');grid;axis([0 500 -5 5]);ylabel('yq(n)');
xlabel('Sample number');
Ak=2*abs(fft(yd))/NN; Ak(1)=Ak(1)/2;
subplot(3,2,6);plot(f(1:NN/2),log10(Ak(1:NN/2)),'k');grid;ylabel('Yq(f)'); axis([0 500
-4 2])
xlabel('Frequency (Hz)');
figure (3)
plot(n(1:50),x(1:50),'k','LineWidth',2); hold % Plot of first 50 samples
stairs(n(1:50),xq(1:50),'b');
stairs(n(1:50),yd(1+nd:50+nd),'r','LineWidth',2);grid
axis([0 50 -5 5]);xlabel('Sample number');ylabel('Amplitudes')
snr(x,xq);
snr(x(1:ntotal-nd),yd(1+nd:ntotal));
```

12.3.2 Sigma-Delta Modulation Analog-to-Digital Conversion

To further improve ADC resolution, *sigma-delta modulation* (SDM) ADC is used. The principles of the first-order SDM are described in Figure 12.28.

First, the analog signal is sampled to obtain the discrete-time signal $x(n)$. This discrete-time signal is subtracted by the analog output from the m -bit DAC, converting the m -bit oversampled digital signal $y(n)$. Then the difference is sent to the discrete-time analog integrator, which is implemented by the switched-capacitor technique, for example. The output from the discrete-time analog integrator is converted using an m -bit ADC to produce the oversampled digital signal. Finally, the decimation filter removes outband quantization noise. Further decimation processes can change the oversampling rate back to the desired sampling rate for the output digital signal $w(m)$.

To examine the SDM, we need to develop a DSP model for the discrete-time analog filter described in Figure 12.29.

As shown in Figure 12.29, the input signal $c(n)$ designates the amplitude at time instant n , while the output $d(n)$ is the area under the curve at time instant n , which can be expressed as a sum of the area under the curve at time instant $n - 1$ and an area increment:

$$d(n) = d(n - 1) + \text{area increment}$$

Using the extrapolation method, we have

$$d(n) = d(n - 1) + 1 \times c(n) \quad (12.23)$$

Applying the z-transform to Equation (12.23) leads to a transfer function of the discrete-time analog filter as

$$H(z) = \frac{D(z)}{C(z)} = \frac{1}{1 - z^{-1}} \quad (12.24)$$

Again, considering that the m -bit quantization requires one sample delay, we get the DSP model for the first-order SDM depicted in Figure 12.30, where $y(n)$ is the oversampling data encoded by m bits each, and $e(n)$ represents quantization error.

The SDM DSP model represents a feedback control system. Applying the z-transform leads to

$$Y(z) = \frac{1}{1 - z^{-1}} (X(z) - z^{-1}Y(z)) + E(z) \quad (12.25)$$

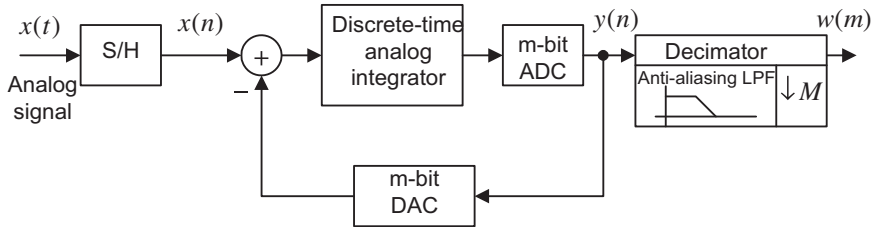


FIGURE 12.28

Block diagram of SDM ADC.

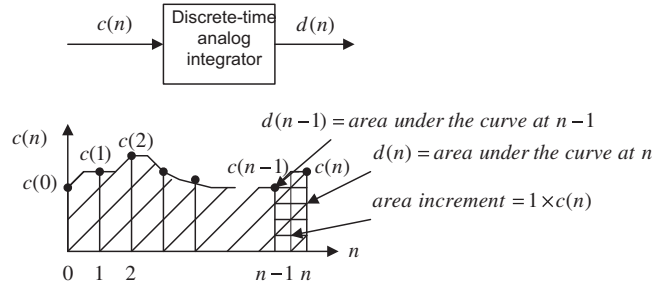


FIGURE 12.29

Illustration of discrete-time analog integrator.

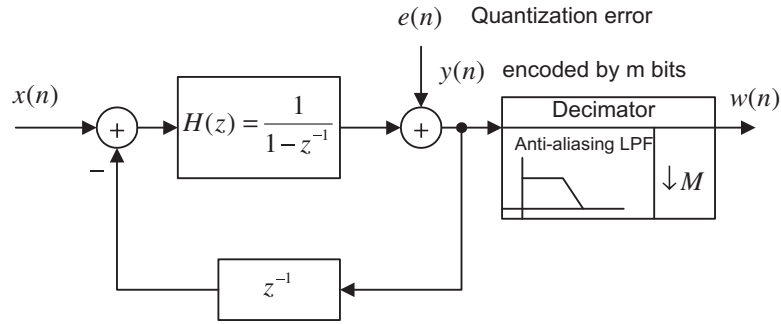


FIGURE 12.30

DSP model for first-order SDM ADC.

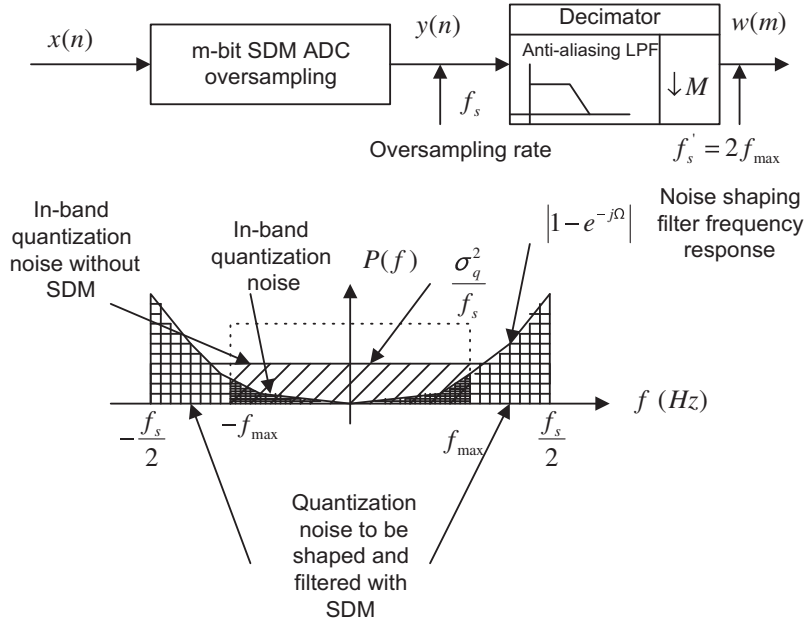
After simple algebra, we have

$$Y(z) = \underbrace{X(z)}_{\substack{\text{Original} \\ \text{digital signal} \\ \text{transform}}} + \underbrace{(1 - z^{-1})}_{\substack{\text{Highpass} \\ \text{filter}}} \cdot \underbrace{E(z)}_{\substack{\text{Quantization} \\ \text{error} \\ \text{transform}}} \quad (12.26)$$

In Equation (12.26), the indicated highpass filter pushes quantization noise to the high-frequency range, where later the quantization noise can be removed by the decimation filter. Thus we call this highpass filter $(1 - z^{-1})$ the *noise shaping filter* (illustrated in Figure 12.31).

Shaped-in-band noise power after use of decimation filter can be estimated by the solid area under the curve. We have

$$\text{Shaped-in-band noise power} = \int_{-\Omega_{\max}}^{\Omega_{\max}} \frac{\sigma_q^2}{2\pi} |1 - e^{-j\Omega}|^2 d\Omega \quad (12.27)$$

**FIGURE 12.31**

Noise shaping of quantization noise for SDM ADC.

Using the Maclaurin series expansion and neglecting the higher-order terms due to the small value of Ω_{\max} , we yield

$$1 - e^{-j\Omega} = 1 - \left(1 + \frac{(-j\Omega)}{1!} + \frac{(-j\Omega)^2}{2!} + \dots \right) \approx j\Omega$$

Applying this approximation to Equation (12.27) leads to

$$\text{Shaped-in-band noise power} \approx \int_{-\Omega_{\max}}^{\Omega_{\max}} \frac{\sigma_q^2}{2\pi} |j\Omega|^2 d\Omega = \frac{\sigma_q^2}{3\pi} \Omega_{\max}^3 \quad (12.28)$$

After simple algebra, we have

$$\text{Shaped-in-band noise power} \approx \frac{\pi^2 \sigma_q^2}{3} \left(\frac{2f_{\max}}{f_s} \right)^3 = \frac{\pi^2}{3} \cdot \frac{A^2 2^{-2m}}{12} \left(\frac{2f_{\max}}{f_s} \right)^3 \quad (12.29)$$

If we let the shaped-in-band noise power equal the quantization noise power from the regular ADC using a minimum sampling rate, we have

$$\frac{\pi^2}{3} \cdot \frac{A^2 2^{-2m}}{12} \left(\frac{2f_{\max}}{f_s} \right)^3 = \frac{A^2}{12} \cdot 2^{-2n} \quad (12.30)$$

We modify Equation (12.30) into the following useful formats for applications:

$$n = m + 1.5 \times \log_2 \left(\frac{f_s}{2f_{\max}} \right) - 0.86 \quad (12.31)$$

$$\left(\frac{f_s}{2f_{\max}} \right)^3 = \frac{\pi^2}{3} \times 2^{2(n-m)} \quad (12.32)$$

EXAMPLE 12.9

Given the following DSP system specifications, determine the equivalent ADC resolution.

Oversampling rate system

First-order SDM with 2-bit ADC

Sampling rate = 4 MHz

Maximum audio input frequency = 4 kHz

Solution:

Since $m = 2$ bits, and

$$\frac{f_s}{2f_{\max}} = \frac{4,000 \text{ kHz}}{2 \times 4 \text{ kHz}} = 500$$

we calculate

$$n = m + 1.5 \times \log_2 \left(\frac{f_s}{2f_{\max}} \right) - 0.86 = 2 + 1.5 \times \log_2 (500) - 0.86 \approx 15 \text{ bits}$$

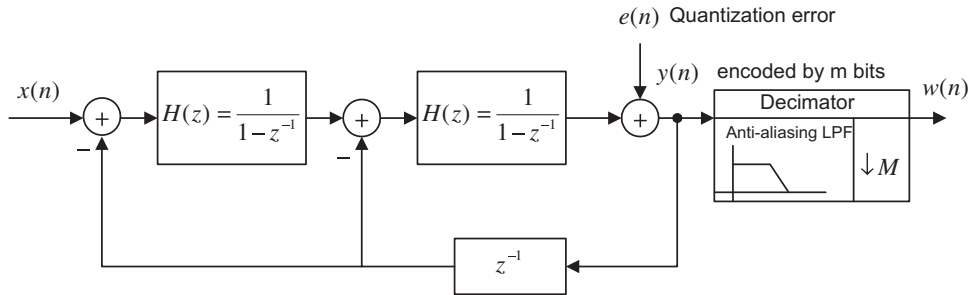
We can also extend the first-order SDM DSP model to the second-order SDM DSP model by cascading one section of the first-order discrete-time analog filter as depicted in Figure 12.32.

Similarly to the first-order SDM DSP model, applying the z-transform leads to the following relationship:

$$Y(z) = \underbrace{X(z)}_{\substack{\text{Original} \\ \text{digital signal} \\ \text{transform}}} + \underbrace{(1 - z^{-1})^2}_{\substack{\text{Highpass} \\ \text{noise shaping} \\ \text{filter}}} \cdot \underbrace{E(z)}_{\substack{\text{Quantization} \\ \text{error} \\ \text{transform}}} \quad (12.33)$$

Notice that the noise shape filter becomes a second-order highpass filter; hence, the more quantization noise is pushed to the high frequency range, the better ADC resolution is expected to be. In a similar analysis to the first-order SDM, we get the following useful formulas:

$$n = m + 2.5 \times \log_2 \left(\frac{f_s}{2f_{\max}} \right) - 2.14 \quad (12.34)$$

**FIGURE 12.32**

DSP model for the second-order SDM ADC.

$$\left(\frac{f_s}{2f_{\max}}\right)^5 = \frac{\pi^4}{5} \times 2^{2(n-m)} \quad (12.35)$$

In general, the K th-order SDM DSP model and ADC resolution formulas are given as

$$Y(z) = \underbrace{X(z)}_{\substack{\text{Original} \\ \text{digital signal} \\ \text{transform}}} + \underbrace{(1 - z^{-1})^K}_{\substack{\text{Highpass} \\ \text{noise shaping} \\ \text{filter}}} \times \underbrace{E(z)}_{\substack{\text{Quantization} \\ \text{error} \\ \text{transform}}} \quad (12.36)$$

$$n = m + 0.5 \cdot (2K + 1) \times \log_2 \left(\frac{f_s}{2f_{\max}} \right) - 0.5 \times \log_2 \left(\frac{\pi^{2K}}{2K + 1} \right) \quad (12.37)$$

$$\left(\frac{f_s}{2f_{\max}}\right)^{2K+1} = \frac{\pi^{2K}}{2K + 1} \times 2^{2(n-m)} \quad (12.38)$$

EXAMPLE 12.10

Given an oversampling rate DSP system with the following specifications, determine the effective ADC resolution:

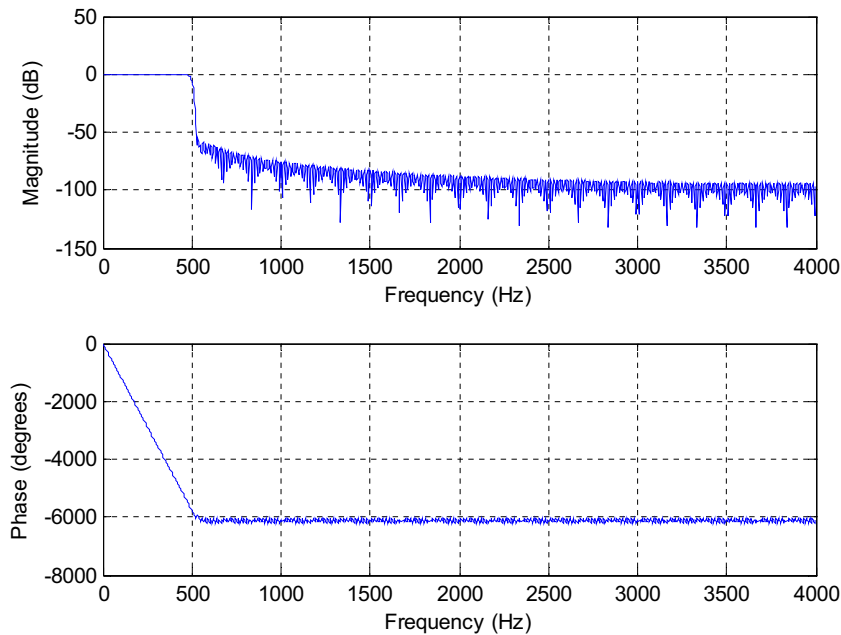
Second-order SDM = 1-bit ADC

Sampling rate = 1 MHz

Maximum audio input frequency = 4 kHz

Solution:

$$n = 1 + 2.5 \times \log_2 \left(\frac{1,000 \text{ kHz}}{2 \times 4 \text{ kHz}} \right) - 2.14 \approx 16 \text{ bits}$$

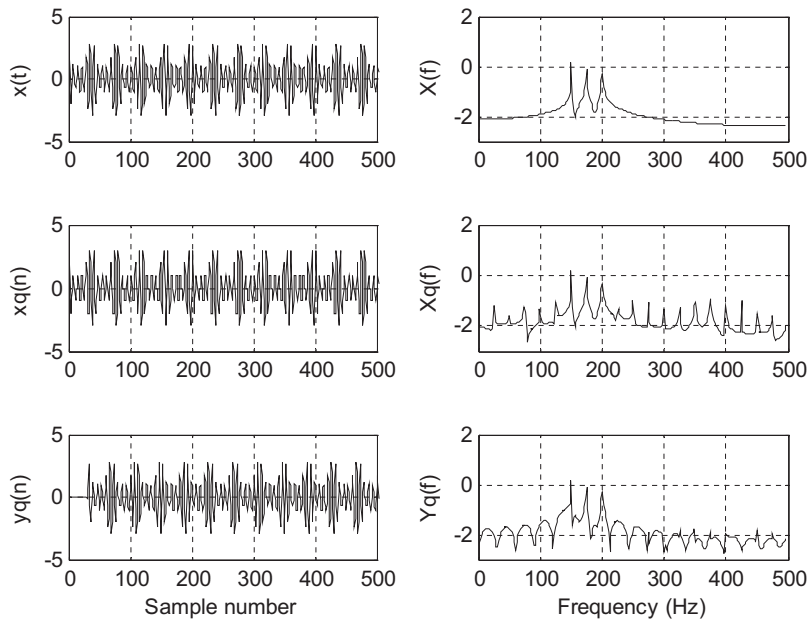
**FIGURE 12.33**

Frequency responses of the designed filter.

We implement the first-order SDM system using the same continuous signal in Equation (12.22). The continuous signal is originally sampled at 1 kHz and each sample is encoded using 3 bits. The SDM system uses an oversampling rate of 8 kHz ($L = 8$) and each sample is quantized using a 3-bit code. The anti-aliasing lowpass filter is designed with a cutoff frequency of $\Omega = 2\pi f_{\max}T = 2\pi \times 500/8,000 = \pi/8$ radians. Figure 12.33 shows the frequency responses of the designed filter while Figure 12.34 compares the time and frequency domain signals, where $x(t)$ designates the continuous version, $x_q(n)$ denotes the quantized version using a regular sampling rate ($L = 1$), and $y_q(n)$ is the enhanced version using $L = 8$. The detailed amplitude comparisons are given in Figure 12.35. The measured SNRs are 14.3 dB in the regular sampling system and 33.83 dB in the oversampling SDM system. We observe a significant SNR improvement of 19.5 dB. The detailed implementation using MATLAB is given in Program 12.8.

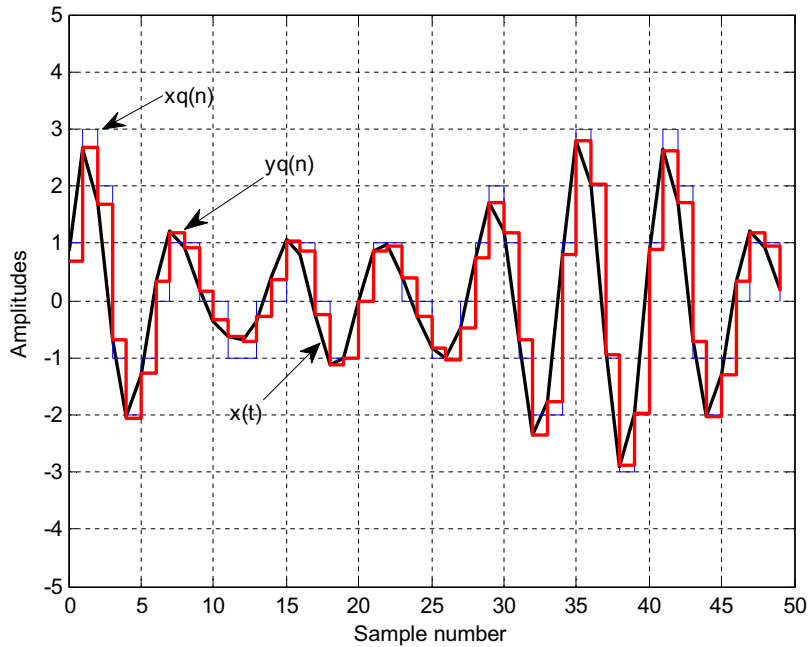
Program 12.8. First-order SDM oversampling implementation.

```
clear all; close all; clc
ntotal=512; % Number of samples
n=0:ntotal;
L=8; % Oversampling factor
nl=0:ntotal*L; numb=3; A=2^(numb-1)-1; % Peak value
f1=150; C1=0.5*A; f2=175; C2=A*0.3; f3=200; C3=A*0.2; % Frequencies and amplitudes
fmax=500; fs=1000; T=1/fs; % Sampling rate and sampling period
fsL=L*fs; TL=1/fsL; % Oversampling rate and oversampling period
```

**FIGURE 12.34**

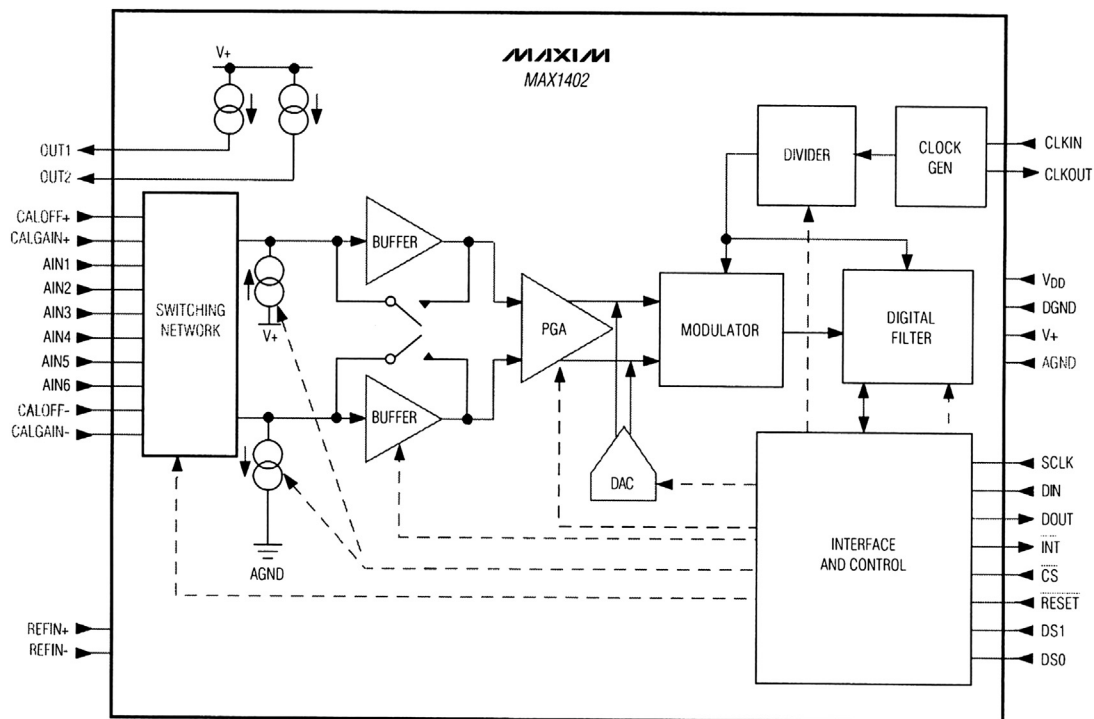
Signal comparisons in both the time and frequency domains.

```
% Sampling at fs = 1000 Hz
x=C1*sin(2*pi*f1*n*T)+C2*sin(2*pi*f2*T*n+pi/6)+C3*sin(2*pi*f3*T*n+pi/4);
xq=round(x); % Quantization
NN=length(n);
M=32*L;nd=M/L; % Delay in terms of samples for anti-aliasing filtering
B=firwd(2*M+1,1,2*pi*fmax/fsL,0,4); % Design of an anti-aliasing filter
figure(1)
freqz(B,1,1000,fsL);
% Oversampling
xx=C1*sin(2*pi*f1*nL*TL)+C2*sin(2*pi*f2*nL*TL+pi/6)+C3*sin(2*pi*f3*nL*TL+pi/4);
% The first-order SDM processing
yq=zeros(1,ntotal*L+1+1); % Initializing the buffer
y=yq;
for i=1:ntotal*L
    y(i+1)=(xx(i+1)-yq(i))+y(i);
    yq(i+1)=round(y(i+1));
end
xxq=yq(1:ntotal*L+1); % Signal quantization
% Downsampling
y=filter(B,1,xxq);
yd=y(1:L:length(y));
f=[0:ntotal-1]*fs/NN;
```

**FIGURE 12.35**

Comparison of continuous, regular sampled, and oversampled signal amplitudes.

```
figure (2)
subplot(3,2,1);plot(n,x,'k');grid;axis([0 500 -5 5]);ylabel('x(t)');
Ak=2*abs(fft(x))/NN; Ak(1)=Ak(1)/2;
subplot(3,2,2);plot(f(1:NN/2),log10(Ak(1:NN/2)),'k');grid;
axis([0 500 -3 2]);ylabel('X(f)');
subplot(3,2,3);plot(n,xq,'k');grid;axis([0 500 -5 5]);ylabel('xq(n)');
Ak=2*abs(fft(xq))/NN; Ak(1)=Ak(1)/2;
subplot(3,2,4);plot(f(1:NN/2),log10(Ak(1:NN/2)),'k');grid;
axis([0 500 -3 2]);ylabel('Xq(f)');
subplot(3,2,5);plot(n,yd,'k');grid;axis([0 500 -5 5]);ylabel('yq(n)');
xlabel('Sample number');
Ak=2*abs(fft(yd))/NN; Ak(1)=Ak(1)/2;
subplot(3,2,6);plot(f(1:NN/2),log10(Ak(1:NN/2)),'k');grid;
axis([0 500 -3 2]);ylabel('Yq(f)');xlabel('Frequency (Hz)');
figure (3)
plot(n(1:50),x(1:50),'k','LineWidth',2); hold
stairs(n(1:50),xq(1:50),'b');
stairs(n(1:50),yd(1+nd:50+nd),'r','LineWidth',2);
axis([0 50 -5 5]);grid;xlabel('Sample number');ylabel('Amplitudes');
snr(x,xq);
snr(x(1:ntotal-nd),yd(1+nd:ntotal));
```

**FIGURE 12.36**

Functional diagram for the sigma-delta ADC.

Next, we review application of oversampling ADC in industry. Figure 12.36 illustrates a function diagram for the MAX1402 low-power, multichannel oversampling sigma-delta analog-to-digital converter used in industry. It applies a sigma-delta modulator with a digital decimation filter to achieve 16-bit accuracy. The device offers three fully differential input channels, which can be independently programmed. It can also be configured as five pseudo-differential input channels. It comprises two chopper buffer amplifiers and a programmable gain amplifier, a DAC unit with predicted input subtracted from the analog input to acquire the differential signal, and a second-order switched-capacitor sigma-delta modulator.

The chip produces a 1-bit data stream, which will be filtered by the integrated digital filter to complete ADC. The digital filter's user-selectable decimation factor offers flexibility as conversion resolution can be reduced in exchange for a higher data rate or *vice versa*. The integrated digital lowpass filter is a first-order or third-order Sinc infinite impulse response filter. Such a filter offers notches corresponding to its output data rate and its frequency harmonics, so it can effectively reduce the developed image noises in the frequency domain (the Sinc filter is beyond the scope of our discussion). The MAX1402 can provide 16-bit accuracy at 480 samples per second and 12-bit accuracy at 4,800 samples per second. The chip finds wide application in sensors and instrumentation. The MAX1402 data sheet contains detailed figure information (Maxim Integrated Products, 2007).

12.4 APPLICATION EXAMPLE: CD PLAYER

Figure 12.37 illustrates a CD playback system, also described earlier in this chapter. A laser optically scans the tracks on a CD to produce a digital signal. The digital signal is then demodulated, and parity bits are used to detect bit errors due to manufacturing defects, dust, and so on, and to correct them. The demodulated signal is again oversampled by a factor of 4 and hence the sampling rate is increased to 176.4 kHz for each channel. Each digital sample then passes through a 14-bit DAC, which produces the sample-and-hold voltage signals that pass the anti-image lowpass filter. The output from each analog filter is fed to its corresponding loudspeaker. Oversampling relaxes the design requirements of the analog anti-image lowpass filter, which is used to smooth out the voltage steps.

The earliest system used a third-order Bessel filter with a 3-dB gain attenuation at 30 kHz. Notice that first-order sigma delta modulation (first-order SDM) is added to the 14-bit DAC unit to further improve the 14-bit DAC to 16-bit DAC.

Let us examine the single-channel DSP portion as shown in Figure 12.38.

The spectral plots for the oversampled and interpolated signal $\bar{x}(n)$, the 14-bit SDM output $y(n)$, and the final analog output audio signal are given in Figure 12.39. As we can see in plot (a) in the figure, the quantization noise is uniformly distributed, and only in-band quantization noise (0 to 22.05 kHz) is expected. Again, 14 bits for each sample are kept after oversampling. Without using the first-order SDM, we expect the effective ADC resolution due to oversampling to be

$$n = 14 + 0.5 \times \log_2 \left(\frac{176.4}{44.1} \right) = 15 \text{ bits}$$

which is fewer than 16 bits. To improve quality further, the first-order SDM is used. The in-band quantization noise is then shaped. The first-order SDM pushes quantization noise to the high-frequency range, as illustrated in plot (b) in Figure 12.39. The effective ADC resolution now becomes

$$n = 14 + 1.5 \times \log_2 \left(\frac{176.4}{44.1} \right) - 0.86 \approx 16 \text{ bits}$$

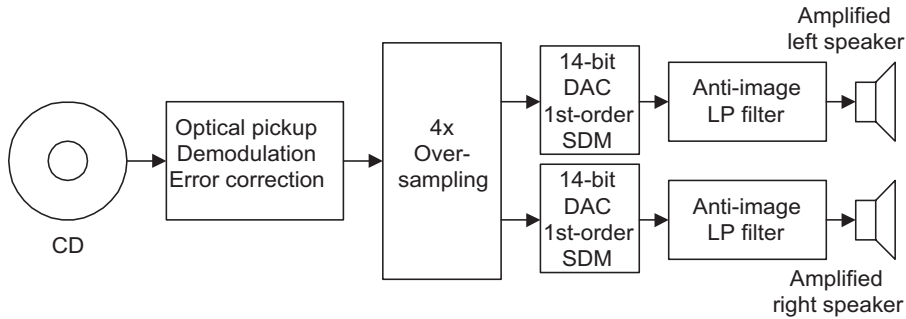


FIGURE 12.37

Simplified decoder of a CD recording system.

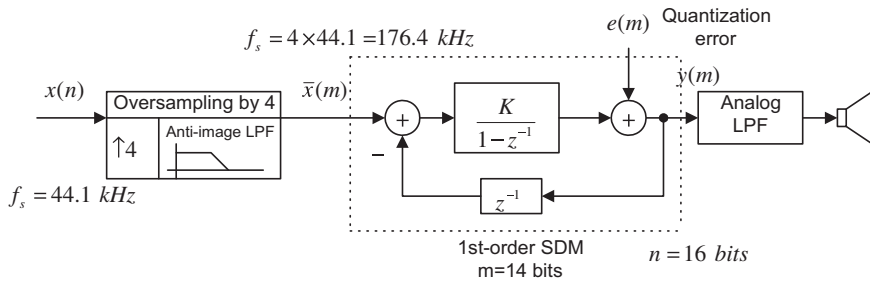
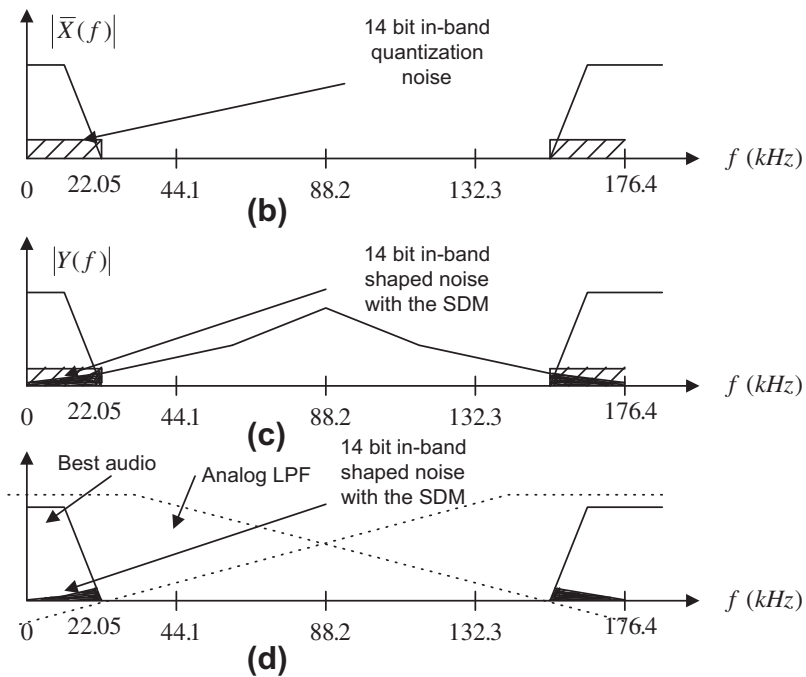
**FIGURE 12.38**

Illustration of oversampling and SDM ADC used in the decoder of a CD recording system.

**FIGURE 12.39**

Spectral illustrations for oversampling and SDM ADC used in the decoder of a CD recording system.

Hence, 16-bit ADC audio quality is preserved. On the other hand, from plot (c) in Figure 12.39, the audio occupies a frequency range up to 22.05 kHz, while the DSP Nyquist limit is 88.2 kHz, so the low-order analog anti-image filter can satisfy the design requirement.

12.5 UNDERSAMPLING OF BANDPASS SIGNALS

As we discussed in Chapter 2, the sampling theorem requires that the sampling rate be twice as large as the highest frequency of the analog signal to be sampled. The sampling theorem ensures the complete reconstruction of the analog signal without aliasing distortion. In some applications, such as the modulated signals in communications systems, the signal exists in only a small portion of the bandwidth. Figure 12.40 shows an amplitude modulated (AM) signal in both the time domain and frequency domain. Assuming that the message signal has a bandwidth of 4 kHz and a carrier frequency of 96 kHz, the upper frequency edge of the AM signal is therefore 100 kHz ($f_c + B$). Then the traditional sampling process requires that the sampling rate be larger than 200 kHz ($2(f_c + B)$), resulting in at a high processing cost. Note that sampling the baseband signal of 4 kHz only requires a sampling rate of 8 kHz ($2B$).

If a certain condition is satisfied at the undersampling stage, we are able to make use of the aliasing signal to recover the message signal, since the aliasing signal contains the folded original message information (which we used to consider distortion). The reader is referred to the undersampling technique discussed in Ifeachor and Jervis (2002) and Porat (1997). Let the message to be recovered have a bandwidth of B , the theoretical minimum sampling rate be $f_s = 2B$, and the carrier frequency of the modulated signal be f_c . We discuss the following cases.

Case 1

If $f_c = \text{even integer} \times B$ and $f_c = 2B$, the sampled spectrum with all the replicas will be as shown in Figure 12.41(a).

As an illustrative example in the time domain for Case 1, suppose we have a bandpass signal with a carrier frequency of 20 Hz; that is,

$$x(t) = \cos(2\pi \times 20t)m(t) \quad (12.39)$$

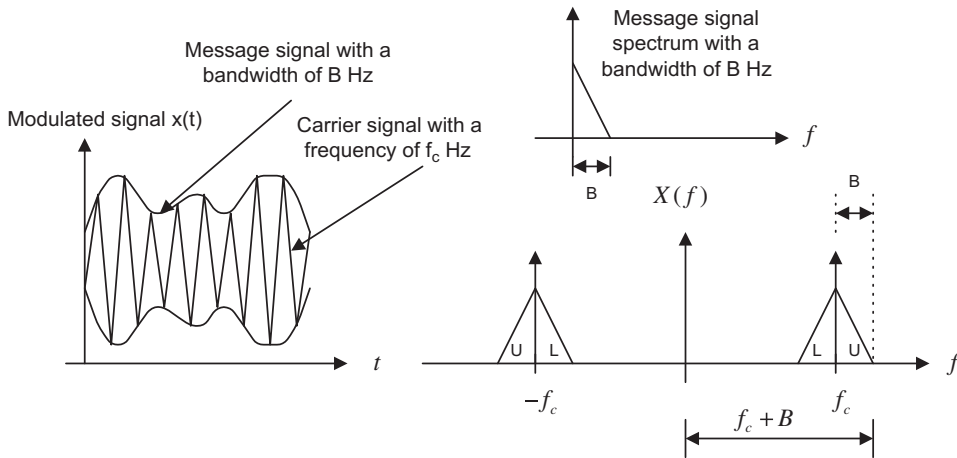
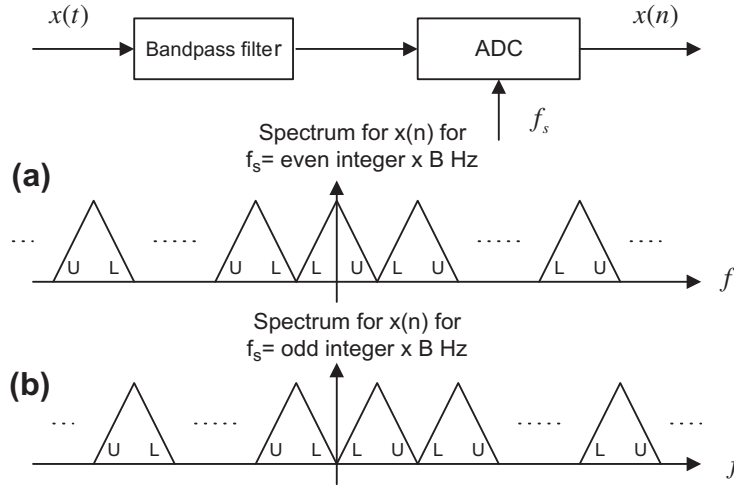


FIGURE 12.40

Message signal, modulated signal, and their spectra.

**FIGURE 12.41**

Spectrum of the undersampled signal.

where $m(t)$ is the message signal with a bandwidth of 2 Hz. Using a sampling rate of 4 Hz by substituting $t = nT$, where $T = 1/f_s$ into Equation (12.39), we get the sampled signal as

$$x(nT) = \cos(2\pi \times 20t)m(t)|_{t=nT} = \cos(2\pi \times 20n/4)m(nT) \quad (12.40)$$

Since $10n\pi = 5n(2\pi)$ is a multiple of 2π ,

$$\cos(2\pi \times 20n/4) = \cos(10\pi n) = 1 \quad (12.41)$$

we obtain the undersampled signal as

$$x(nT) = \cos(2\pi \times 20n/4)m(nT) = m(nT) \quad (12.42)$$

which is a perfect digital message signal. Figure 12.42 shows the bandpass signal and its sampled signals when the message signal is 1 Hz, given as

$$m(t) = \cos(2\pi t) \quad (12.43)$$

Case 2

If $f_c = \text{odd integer} \times B$ and $f_c = 2B$, the sampled spectrum with all the replicas will be as shown in Figure 12.41(b), where the spectral portions L and U are reversed. Hence, frequency reversal will occur. Then a further digital modulation in which the signal is multiplied by the digital oscillator with a frequency of B Hz can be used to adjust the spectrum to be the same as that in Case 1.

As another illustrative example for Case 2, let us sample the following the bandpass signal with a carrier frequency of 22 Hz, given by

$$x(t) = \cos(2\pi \times 22t)m(t) \quad (12.44)$$

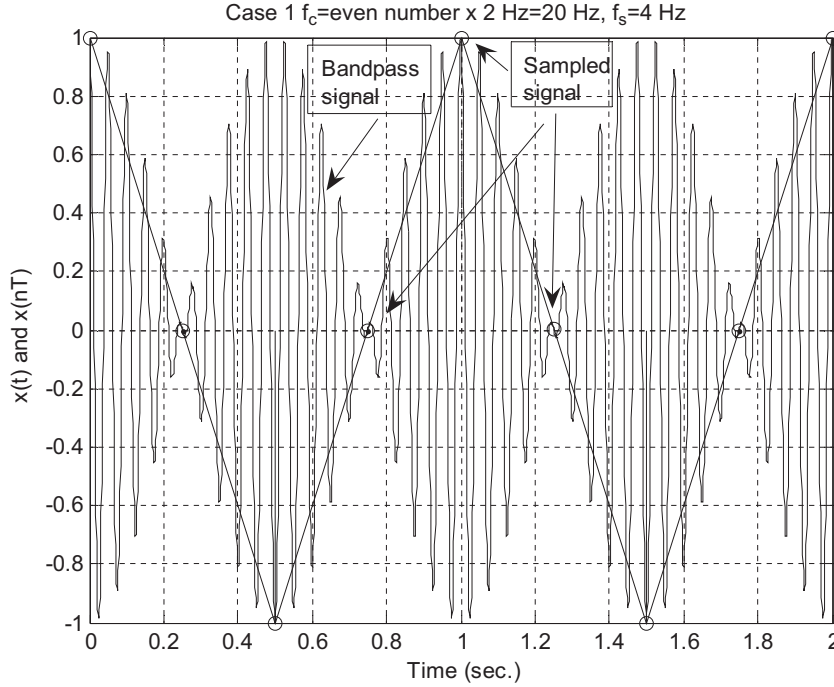


FIGURE 12.42

Plots of the bandpass signal and sampled signal for Case 1.

Applying undersampling using a sampling rate of 4 Hz, it follows that

$$x(nT) = \cos(2\pi \times 22n/4)m(nT) = \cos(11n\pi)m(nT) \quad (12.45)$$

Since $11n\pi$ can be either an odd or an even integer multiple of π , we have

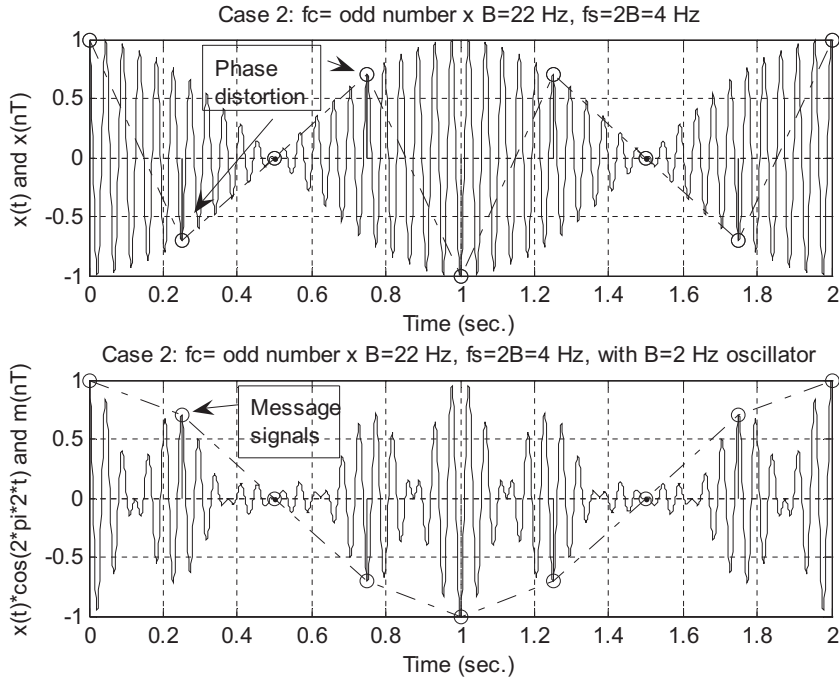
$$\cos(11n\pi) = \begin{cases} -1 & n = \text{odd} \\ 1 & n = \text{even} \end{cases} \quad (12.46)$$

We see that Equation (12.46) causes the message samples to change sign alternatively with a carrier frequency of 22 Hz, which is the odd integer multiple of the message bandwidth of 2 Hz. This in fact will reverse the baseband message spectrum. To correct the spectrum reversal, we multiply an oscillator with a frequency of $B = 2$ Hz by the bandpass signal, that is

$$x(t)\cos(2\pi \times 2t) = \cos(2\pi \times 22t)m(t)\cos(2\pi \times 2t) \quad (12.47)$$

Then the undersampled signal is given by

$$\begin{aligned} x(nT)\cos(2\pi \times 2n/4) &= \cos(2\pi \times 22n/4)m(nT)\cos(2\pi \times 2n/4) \\ &= \cos(11n\pi)m(nT)\cos(n\pi) \end{aligned} \quad (12.48)$$

**FIGURE 12.43**

Plots of the bandpass signals and sampled signals for Case 2.

Since

$$\cos(11\pi n)\cos(\pi n) = 1 \quad (12.49)$$

it follows that

$$x(nT)\cos(2\pi \times 2n/4) = \cos(\pi \times 11n)m(nT)\cos(\pi \times n) = m(nT) \quad (12.50)$$

which is the recovered message signal. Figure 12.43 shows the sampled bandpass signals with the reversed message spectrum and the corrected message spectrum, respectively, for a message signal having a frequency of 0.5 Hz; that is,

$$m(t) = \cos(2\pi \times 0.5t) \quad (12.51)$$

Case 3

If $f_c = \text{noninteger} \times B$, we can extend the bandwidth B to \bar{B} such that

$$f_c = \text{integer} \times \bar{B} \quad \text{and} \quad f_s = 2\bar{B} \quad (12.52)$$

Then we can apply Case 1 or Case 2. An illustration of Case 3 is included in the following example.

EXAMPLE 12.11

Given a bandpass signal with the spectrum and carry frequency f_c shown in Figures 12.44A, 12.44B, and 12.44C, respectively, and assuming the baseband bandwidth $B = 4$ kHz, select the sampling rate and sketch the sampled spectrum ranging from 0 Hz to the carrier frequency for each of the following carrier frequencies:

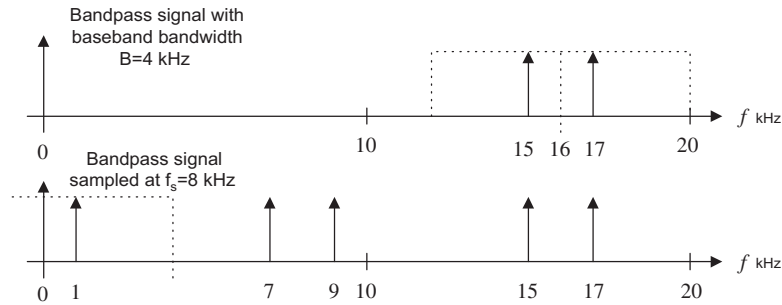


FIGURE 12.44A

Sampled signal spectrum for $f_c = 16$ kHz.

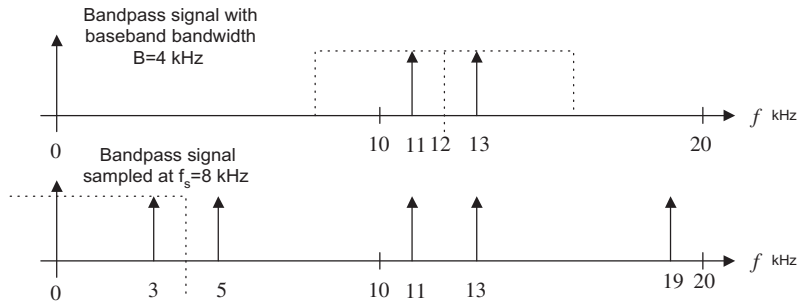


FIGURE 12.44B

Sampled signal spectrum for $f_c = 12$ kHz.

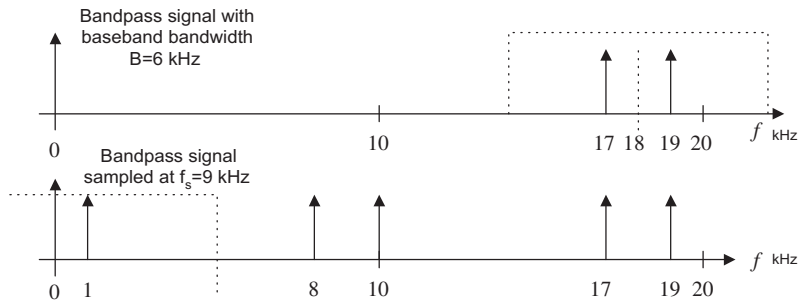


FIGURE 12.44C

Sampled signal spectrum for $f_c = 18$ kHz.

- a. $f_c = 16$ kHz
- b. $f_c = 12$ kHz
- c. $f_c = 18$ kHz

Solution:

- a. Since $f_c/B = 4$ is an even number, which is Case 1, we select $f_s = 8$ kHz and sketch the sampled spectrum shown in Figure 12.44A.
 - b. Since $f_c/B = 3$ is an odd number, we select $f_s = 8$ kHz and sketch the sampled spectrum shown in Figure 12.44B.
 - c. Now, $f_c/B = 4.5$, which is a noninteger. We extend the bandwidth $\bar{B} = 4.5$ kHz, so $f_c/\bar{B} = 4$ and $f_s = 2\bar{B} = 9$ kHz. The sketched spectrum is shown in Figure 12.44C.
-

Simulation Example

An AM with a 1-kHz message signal is given as

$$x(t) = [1 + 0.8 \times \sin(2\pi \times 1,000t)]\cos(2\pi \times f_c t) \quad (12.53)$$

Assuming a message bandwidth of 4 kHz, determine the sampling rate, use MATLAB to sample the AM signal, and sketch the sampled spectrum up to the sampling frequency for each of the following carrier frequencies:

- a. $f_c = 96$ kHz
 - b. $f_c = 100$ kHz
 - c. $f_c = 99$ kHz
- a. For this case, $f_c/B = 24$ is an even number. We select $f_s = 8$ kHz. Figure 12.45A describes the simulation, where the upper left plot is the AM signal, the upper right plot is the spectrum of the AM signal, the lower left plot is the undersampled signal, and the lower right plot is the spectrum of the undersampled signal displayed from 0 to 8 kHz.
 - b. $f_c/B = 25$ is an odd number, so we choose $f_s = 8$ kHz, and a further process is needed. We can multiply the undersampled signal by a digital oscillator with a frequency of $B = 4$ kHz to achieve the 1-kHz baseband signal. The plots of the AM signal spectrum, undersampled signal spectrum, and the oscillator mixed signal and its spectrum are shown in Figure 12.45B.
 - c. For $f_c = 99$ kHz, $f_c/B = 24.75$. We extend the bandwidth to $\bar{B} = 4.125$ so that $f_c/\bar{B} = 24$. Hence, $f_s = 8.25$ kHz is used as the undersampling rate. Figure 12.45C shows the plots for the AM signal, the AM signal spectrum, the undersampled signal based on the extended baseband width, and the sampled signal spectrum ranging from 0 to 8.25 kHz, respectively.

This example verifies the principles of undersampling of bandpass signals.

12.6 SAMPLING RATE CONVERSION USING THE TMS320C6713 DSK

Downsampling by an integer factor of M using the TMS320C6713 is depicted in Figure 12.46. The idea is that we set up the DSK running at the original sampling rate and update the DAC channel once for M samples. The program (Tan and Jiang, 2008) is shown in Program 12.9.

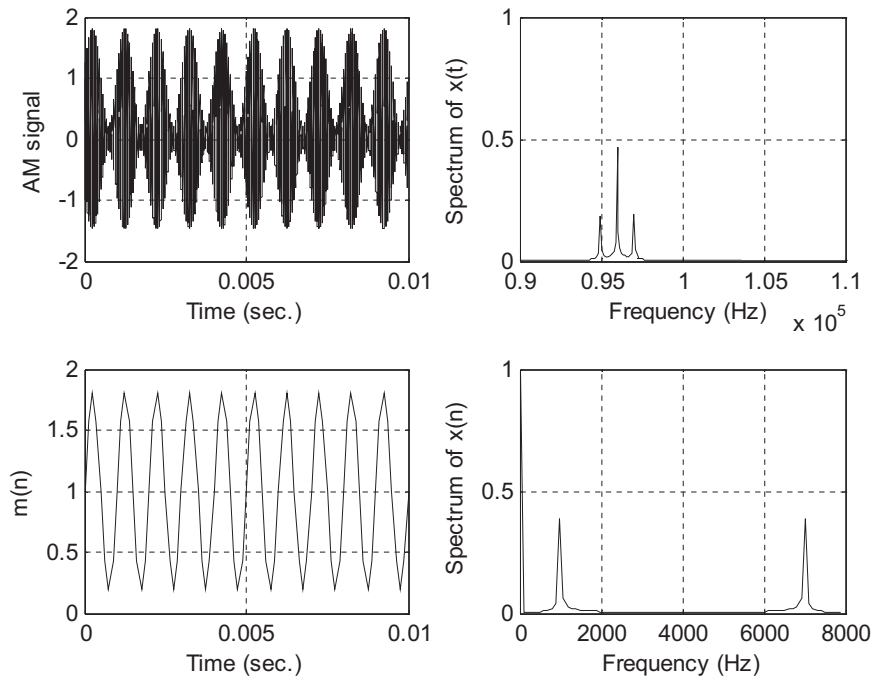


FIGURE 12.45A

Sampled AM signal and spectrum for $f_c = 96$ kHz.

Program 12.9. Downsampling implementation (anti-aliasing filter has 67 coefficients that are stored in an array `b[67]`).

```
int M=2;
int Mcount=0;
float x[67];
float y[1]={0.0};
interrupt void c_int11()
{
    float lc; /*Left channel input */
    float rc; /*Right channel input */
    float lcnew; /*Left channel output */
    float rcnew; /*Right channel output */
    int i,j;
    float sum;
    // Left channel and right channel inputs
    AIC23_data.combo=input_sample();
    lc=(float) (AIC23_data.channel[LEFT]);
    rc= (float) (AIC23_data.channel[RIGHT]);
```

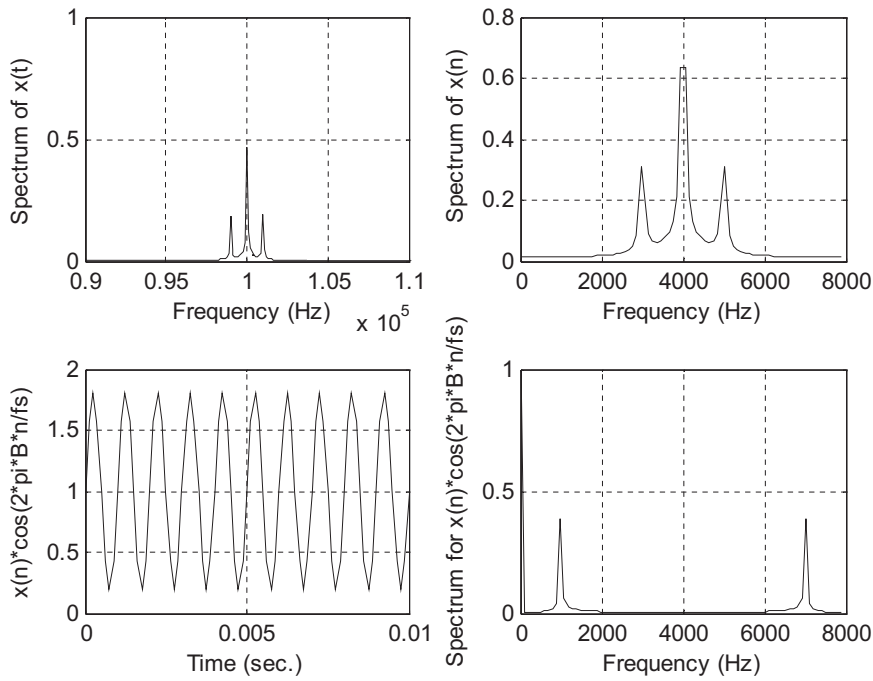


FIGURE 12.45B

Sampled AM signal and spectrum for $f_c = 100$ kHz.

```
// Insert DSP algorithm below
Mcount++;
for(i=66; i>0; i--) // Update input buffer
{ x[i]=x[i-1]; }
x[0]=lc; // Load new sample
sum=0.0;
for(i=0; i<67; i++) // FIR filtering
{ sum=sum+x[i]*b[i]; }
if (Mcount==M)
{ Mcount=0;
  y[0]=sum; } // Update DAC with processed sample (decimation)
// End of the DSP algorithm
lcnew=y[0]; /* Send to DAC */
rcnew=y[0];
AIC23_data.channel[LEFT]=(short) lcnew;
AIC23_data.channel[RIGHT]=(short) rcnew;
output_sample(AIC23_data.combo);
}
```

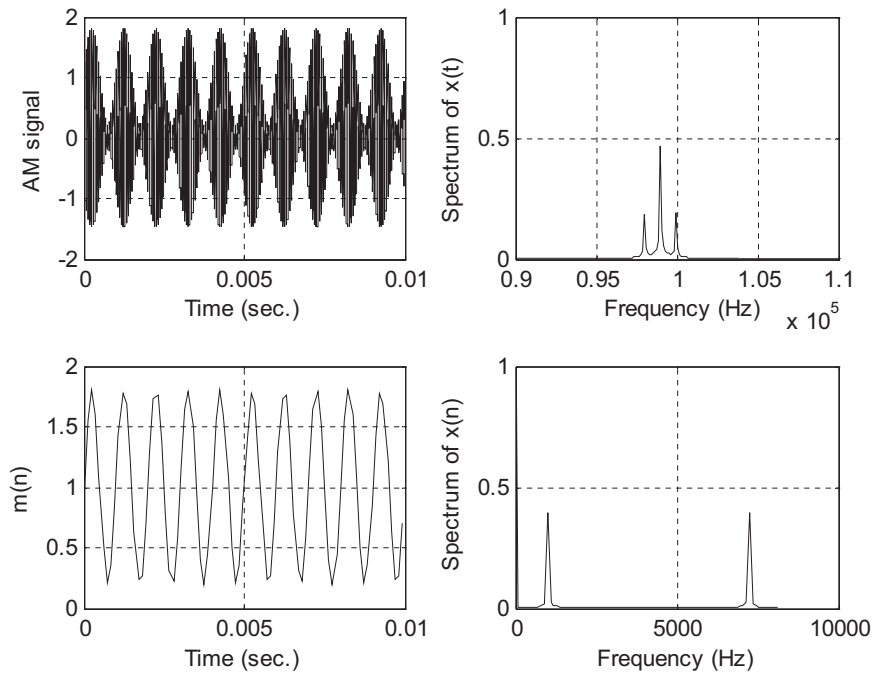
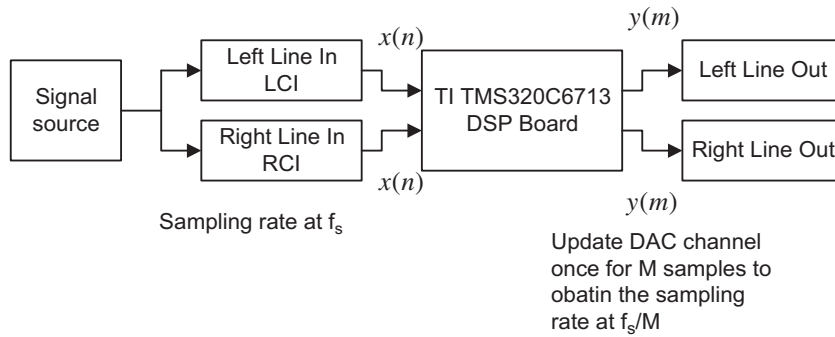



FIGURE 12.45C

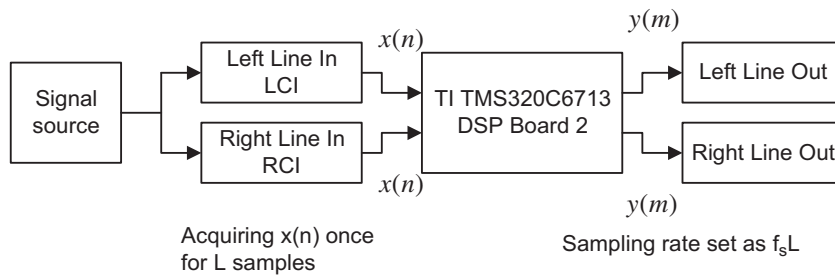
Sampled AM signal and spectrum for $f_c = 99$ kHz.

Upsampling by an integer factor of L using the TMS320C6713 is depicted in Figure 12.47. Again, we set up the DSK to run at the upsampling rate f_{sL} and acquire a sample from the ADC channel once for L samples. The program (Tan and Jiang, 2008) is shown in Program 12.10. Program 12.10. Upsampling implementation (anti-image filter has 67 coefficients that are stored in an array `b[67]`).

```
int L=2;
int Lcount=0;
float x[67];
float y[1]={0.0};
interrupt void c_int11()
{
    float lc; /*Left channel input */
    float rc; /*Right channel input */
    float lcnew; /*Left channel output */
    float rcnew; /*Right channel output */
    int i,j;
    // Left channel and right channel inputs
    AIC23_data.combo=input_sample();
```


FIGURE 12.46

Downsampling using the TMS320C6713.


FIGURE 12.47

Upsampling using the TMS320C6713.

```

lc=(float) (AIC23_data.channel[LEFT]);
rc= (float) (AIC23_data.channel[RIGHT]);
// Insert DSP algorithm below
Lcount++;
for(i=66; i>0; i--) // Update input buffer with zeros
{ x[i]=x[i-1]; }
x[0]=0;
if (Lcount==L)
{
    x[0]=lc; // Load new sample for every L samples
    Lcount =0;
}
y[0]=0.0;
for(i=0; i<67; i++) // FIR filtering
{ y[0]=y[0]+x[i]*b[i]; }
y[0]=(float) L*y[0]

```

```
// End of the DSP algorithm
lcnew=y[0]; /* Send to DAC */
rcnew=y[0];
AIC23_data.channel[LEFT]=(short) lcnew;
AIC23_data.channel[RIGHT]=(short) rcnew;
output_sample(AIC23_data.combo);
}
```

12.7 SUMMARY

1. Downsampling (decimation) by an integer factor of M means taking one sample from the data sequence $x(n)$ for every M samples and discarding the last $M - 1$ samples.
2. Upsampling (interpolation) by an integer factor of L means inserting $L - 1$ zeros for every sample in the data sequence $x(n)$.
3. Downsampling requires a decimation (anti-aliasing) filter to avoid frequency aliasing before downsampling.
4. Upsampling requires an interpolation (anti-image) filter to remove the images after interpolation.
5. Changing the sampling rate by a noninteger factor of L/M requires two stages: an interpolation stage and a downsampling stage.
6. Two-stage decimation can dramatically reduce the anti-aliasing filter length.
7. Polyphase implementations of the decimation filter and interpolation filter can reduce the complexity of the filter operations, that is, fewer multiplications and additions.
8. Using oversampling can improve the regular ADC resolution. Sigma-delta modulation ADC can achieve even higher ADC resolution, using the noise shaping effect for further reduction of quantization noise.
9. The audio CD player uses multirate signal processing and oversampling.
10. Undersampling can be used to sample the bandpass signal, leading to applications in communications.

12.8 PROBLEMS

- 12.1. Consider a single-stage decimator with the following specifications:
 - Original sampling rate = 1 kHz
 - Decimation factor $M = 2$
 - Frequency of interest = 0–100 Hz
 - Passband ripple = 0.015 dB
 - Stopband attenuation = 40 dB
 - a. Draw the block diagram for the decimator.
 - b. Determine the window type, filter length, and cutoff frequency if the window method is used for the anti-aliasing FIR filter design.
- 12.2. Consider a single-stage interpolator with the following specifications:
 - Original sampling rate = 1 kHz

Interpolation factor $L = 2$
 Frequency of interest = 0–150 Hz
 Passband ripple = 0.02 dB
 Stopband attenuation = 45 dB

- a. Draw the block diagram for the interpolator.
- b. Determine the window type, filter length, and cutoff frequency if the window method is used for the anti-image FIR filter design.

12.3. Consider a single stage decimator with the following specifications:

Original sampling rate = 8 kHz
 Decimation factor $M = 4$
 Frequency of interest = 0–800 Hz
 Passband ripple = 0.02 dB
 Stopband attenuation = 46 dB,

- a. Draw the block diagram for the decimator.
- b. Determine the window type, filter length, and cutoff frequency if the window method is used for the anti-aliasing FIR filter design.

12.4. Consider a single-stage interpolator with the following specifications:

Original sampling rate = 8 kHz
 Interpolation factor $L = 3$
 Frequency of interest = 0–3,400 Hz
 Passband ripple = 0.02 dB
 Stopband attenuation = 46 dB

- a. Draw the block diagram for the interpolator.
- b. Determine the window type, filter length, and cutoff frequency if the window method is used for the anti-image FIR filter design.

12.5. Consider the sampling conversion from 4 kHz to 3 kHz with the following specifications:

Original sampling rate = 4 kHz
 Interpolation factor $L = 3$
 Decimation factor $M = 2$
 Frequency of interest = 0–400 Hz
 Passband ripple = 0.02 dB
 Stopband attenuation = 46 dB

- a. Draw the block diagram for the interpolator.
- b. Determine the window type, filter length, and cutoff frequency if the window method is used for the combined FIR filter $H(z)$.

12.6. Consider the design of a two-stage decimator with the following specifications:

Original sampling rate = 32 kHz
 Frequency of interest = 0–250 Hz
 Passband ripple = 0.05 (absolute)

Stopband attenuation = 0.005 (absolute)

Final sampling rate = 1,000 Hz

- a. Draw the decimation block diagram.
- b. Specify the sampling rate for each stage.
- c. Determine the window type, filter length, and cutoff frequency for the first stage if the window method is used for anti-aliasing FIR filter design ($H_1(z)$).
- d. Determine the window type, filter length, and cutoff frequency for the second stage if the window method is used for the anti-aliasing FIR filter design ($H_2(z)$).

12.7. Consider the sampling conversion from 6 kHz to 8 kHz with the following specifications:

Original sampling rate = 6 kHz

Interpolation factor $L = 4$

Decimation factor $M = 3$

Frequency of interest = 0–2,400 Hz

Passband ripple = 0.02 dB

Stopband attenuation = 46 dB

- a. Draw the block diagram for the processor.
- b. Determine the window type, filter length, and cutoff frequency if the window method is used for the combined FIR filter $H(z)$.

12.8. Consider the design of a two-stage decimator with the following specifications:

Original sampling rate = 320 kHz

Frequency of interest = 0–3,400 Hz

Passband ripple = 0.05 (absolute)

Stopband attenuation = 0.005 (absolute)

Final sampling rate = 8,000 Hz

- a. Draw the decimation block diagram.
- b. Specify the sampling rate for each stage.
- c. Determine the window type, filter length, and cutoff frequency for the first stage if the window method is used for anti-aliasing FIR filter design ($H_1(z)$).
- d. Determine the window type, filter length, and cutoff frequency for the second stage if the window method is used for anti-aliasing FIR filter design ($H_2(z)$).

12.9. a. Given an interpolator filter

$$H(z) = 0.25 + 0.4z^{-1} + 0.5z^{-2}$$

draw the block diagram for interpolation polyphase filter implementation for the case of $L = 2$.

b. Given a decimation filter

$$H(z) = 0.25 + 0.4z^{-1} + 0.5z^{-2} + 0.6z^{-3}$$

draw the block diagram for decimation polyphase filter implementation for the case of $M = 2$.

- 12.10.** Using the commutative models for the polyphase interpolation and decimation filters,
- draw the block diagram for interpolation polyphase filter implementation for the case of $L = 2$, and $H(z) = 0.25 + 0.4z^{-1} + 0.5z^{-2}$;
 - draw the block diagram for decimation polyphase filter implementation for the case of $M = 2$, and $H(z) = 0.25 + 0.4z^{-1} + 0.5z^{-2} + 0.6z^{-3}$.

- 12.11.** a. Given an interpolator filter

$$H(z) = 0.25 + 0.4z^{-1} + 0.5z^{-2} + 0.6z^{-3} + 0.7z^{-4} + 0.6z^{-5},$$

draw the block diagram for interpolation polyphase filter implementation for the case of $L = 4$.

- b. Given a decimation filter

$$H(z) = 0.25 + 0.4z^{-1} + 0.5z^{-2} + 0.6z^{-3} + 0.5z^{-3} + 0.4z^{-4}$$

draw the block diagram for decimation polyphase filter implementation for the case of $M = 4$.

- 12.12.** Using the commutative models for the polyphase interpolation and decimation filters,
- draw the block diagram for interpolation polyphase filter implementation for the case of $L = 4$, and $H(z) = 0.25 + 0.4z^{-1} + 0.5z^{-2} + 0.6z^{-3} + 0.7z^{-4} + 0.6z^{-5}$;
 - draw the block diagram for decimation polyphase filter implementation for the case of $M = 4$, and $H(z) = 0.25 + 0.4z^{-1} + 0.5z^{-2} + 0.6z^{-3} + 0.5z^{-3} + 0.4z^{-4}$.

- 12.13.** Consider a speech system with the following specifications:

Speech input frequency range: 0–4 kHz

ADC resolution = 16 bits

Current sampling rate = 8 kHz

- Determine the oversampling rate if a 12-bit ADC chip is used to replace the speech system.

- Draw the block diagram.

- 12.14.** Consider a speech system with the following specifications:

Speech input frequency range: 0–4 kHz

ADC resolution = 6 bits

Oversampling rate = 4 MHz

- Draw the block diagram.

- Determine the actual effective ADC resolution (number of bits per sample).

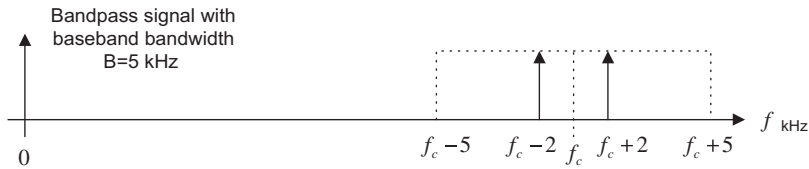
- 12.15.** Consider an audio system with the following specifications:

Audio input frequency range: 0–15 kHz

ADC resolution = 16 bits

Current sampling rate = 30 kHz

- a. Determine the oversampling rate if a 12-bit ADC chip is used to replace the audio system.
 - b. Draw the block diagram.
- 12.16. Consider an audio system with the following specifications:
 Audio input frequency range: 0–15 kHz
 ADC resolution = 6 bits
 Oversampling rate = 45 MHz
 - a. Draw the block diagram.
 - b. Determine the actual effective ADC resolution (number of bits per sample).
- 12.17. Consider the following specifications of an oversampling DSP system:
 Audio input frequency range: 0–4 kHz
 First-order SDM with a sampling rate of 128 kHz
 ADC resolution in SDM = 1 bit
 - a. Draw the block diagram using the DSP model.
 - b. Determine the equivalent (effective) ADC resolution.
- 12.18. Consider the following specifications of an oversampling DSP system:
 Audio input frequency range: 0–20 kHz
 Second-order SDM with a sampling rate of 160 kHz
 ADC resolution in SDM = 10 bits
 - a. Draw the block diagram using the DSP model.
 - b. Determine the equivalent (effective) ADC resolution.
- 12.19. Consider the following specifications of an oversampling DSP system:
 Signal input frequency range: 0–500 Hz
 First-order SDM with a sampling rate of 128 kHz
 ADC resolution in SDM = 1 bit
 - a. Draw the block diagram using the DSP model.
 - b. Determine the equivalent (effective) ADC resolution.
- 12.20. Consider the following specifications of an oversampling DSP system:
 Signal input frequency range: 0–500 Hz
 Second-order SDM with a sampling rate of 16 kHz
 ADC resolution in SDM = 8 bits
 - a. Draw the block diagram using the DSP model.
 - b. Determine the equivalent (effective) ADC resolution.
- 12.21. Given a bandpass signal with a spectrum shown in [Figure 12.48](#), and assuming the bandwidth $B = 5$ kHz, select the sampling rate and sketch the sampled spectrum ranging from 0 Hz to the carrier frequency for each of the following carrier frequencies:
 - a. $f_c = 30$ kHz

**FIGURE 12.48**

Spectrum of the bandpass signal in Problem 12.21.

- b. $f_c = 25$ kHz
 - c. $f_c = 33$ kHz
- 12.22.** Given a bandpass signal with a spectrum shown in Figure 12.48, and assuming $f_s = 10$ kHz, select the sampling rate and sketch the sampled spectrum ranging from 0 Hz to the carrier frequency for each of the following carrier frequencies:
- a. $f_c = 15$ kHz
 - b. $f_c = 20$ kHz
- 12.23.** Given a bandpass signal with a spectrum shown in Figure 12.48, and assuming $B = 5$ kHz, select the sampling rate and sketch the sampled spectrum ranging from 0 Hz to the carrier frequency for each of the following carrier frequencies:
- a. $f_c = 35$ kHz
 - b. $f_c = 40$ kHz
 - c. $f_c = 22$ kHz

12.8.1 MATLAB Problems

Use MATLAB to solve Problems 12.24 to 12.30.

- 12.24.** Generate a sinusoid with a frequency of 1,000 Hz for 0.05 second using a sampling rate of 8 kHz.
- a. Design a decimator to change the sampling rate to 4 kHz with the specifications below:
Signal frequency range: 0–1,800 Hz
Hamming window required for FIR filter design
 - b. Write a MATLAB program to implement the downsampling scheme, and plot the original signal and the downsampled signal versus the sample number, respectively.
- 12.25.** Generate a sinusoid with a frequency of 1,000 Hz for 0.05 second using a sampling rate of 8 kHz.
- a. Design an interpolator to change the sampling rate to 16 kHz with the following specifications:
Signal frequency range: 0–3,600 Hz
Hamming window required for FIR filter design

- b. Write a MATLAB program to implement the upsampling scheme, and plot the original signal and the upsampled signal versus the sample number, respectively.
- 12.26.** Generate a sinusoid with a frequency of 500 Hz for 0.1 second using a sampling rate of 8 kHz.
- a. Design an interpolation and decimation processing algorithm with the following specifications to change the sampling rate to 22 kHz:
Signal frequency range: 0–3,400 Hz
Hamming window required for FIR filter design
- b. Write a MATLAB program to implement the scheme, and plot the original signal and the sampled signal at the rate of 22 kHz versus the sample number, respectively.
- 12.27.** Repeat Problem 12.24 using the polyphase form for the decimator.
- 12.28.** Repeat Problem 12.25 using the polyphase form for the interpolator.
- 12.29. a.** Use MATLAB to create a 1-second sinusoidal signal using a sampling rate of 1 kHz,
- $$x(t) = 1.8 \cos(2\pi \times 100t) + 1.0 \sin(2\pi \times 150t + \pi/4)$$
- where each sample $x(t)$ can be rounded off using a 3-bit signed integer (directly round off the calculated $x(t)$). Evaluate the signal-to-quantization-noise ratio (SQNR).
- b. Use MATLAB to design an oversampling system including an anti-aliasing filter with a selectable integer factor L using the same equation for the input $x(t)$.
- c. Recover the signal using the quantized 3-bit signal and measure the SQNRs for the following integer factors: $L = 2$, $L = 4$, $L = 8$, $L = 16$, and $L = 32$. From the results, explain which one offers better quality for the recovered signals.
- 12.30. a.** Use MATLAB to create a 1-second sinusoidal signal using a sampling rate of 1 kHz,
- $$x(t) = 1.8 \cos(2\pi \times 100t) + 1.0 \sin(2\pi \times 150t + \pi/4)$$
- where each sample $x(t)$ can be rounded off using a 3-bit signed integer (directly round off the calculated $x(t)$). Evaluate the signal-to-quantization-noise ratio (SQNR).
- b. Use MATLAB to implement the first-order SDM system including an anti-aliasing filter with an oversampling factor of 16. Measure the SQNR (signal to noise ratio due to quantization).
- c. Use MATLAB to implement the second-order SDM system including an anti-aliasing filter with an oversampling factor of 16. Measure the SQNR. Compare the SQNR with the one obtained in (b).

12.8.2 MATLAB Project

12.31. Audio rate conversion system:

Given a 16-bit stereo audio file (“No9seg.wav”) with a sampling rate of 44.1 kHz, design a multistage conversion system and implement the designed system to convert the audio file from 44.1 to 48 kHz. Listen and compare the quality of the original audio with the converted audio.