# Section 9. Basic Digital Modulator Theory

*By Ken Gentile, Systems Engineer, Analog Devices, Inc.*

To understand digital modulators it may be insightful to first review some of the basic concepts of signals. With a basic understanding of signals, the concept of baseband and bandpass signals can be addressed. Which then leads to the concept of modulation in continuous time (the analog world). Once continuous time modulation is understood, the step to digital modulation is relatively simple.

<u>Signals</u>

The variety of signal types and classes is broad and a discussion of all of them is not necessary to understand the concepts of digital modulation. However, one class of signals is very important: **periodic complex exponentials**. Periodic complex exponentials have the form:

$$x(t) = \beta(t)e^{j\omega t}$$

where $\beta(t)$ is a function of time and may be either real or complex. It should be noted that $\beta(t)$ is not restricted to a function of time; it may be a constant. Also, $\omega$ is the radian frequency of the periodic signal. The natural frequency, f, is related to the radian frequency by $\omega = 2\pi f$. $\beta(t)$ is known as the **envelope** of the signal. A plot of x(t) is shown in Figure 9.1, where $\beta(t) = \sin(2\pi f_a t)$, $f_a$ = 1MHz, $\omega = 2\pi f_c t$, and $f_c$ = 10MHz. Since x(t) is complex, only the real part of x(t) is plotted (the dashed lines indicate the envelope produced by $\beta(t)$).

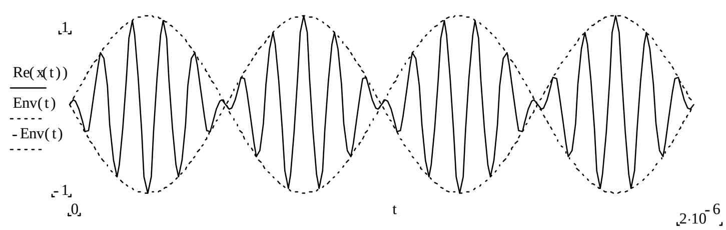


Figure 9.1. Periodic Complex Exponential

An alternative form of x(t) can be obtained by invoking Euler's identity:

$$x(t) = \beta(t)[cos(\omega t) + j sin(\omega t)]$$

This form tends to be a little more intuitive because its complex nature is clearly indicated by its real and imaginary components; a quality not so obvious in the complex exponential form.

A special subset of periodic complex exponentials are **sinusoidal signals**. A sinusoidal signal has the form:

$$x(t) = A cos(\omega t)$$

Again, using Euler's identity, x(t) can be written as:

$$x(t) = \frac{1}{2}A(e^{j\omega t} + e^{-j\omega t})$$

Note from the above equation that a sinusoidal signal contains both positive and negative frequency components (each contributing half of A). This is an important fact to remember. We are generally not accustomed to negative frequency, but it is nontheless mathematically valid. Figure 9.2 is a frequency vs. magnitude plot of a sinusoidal signal.

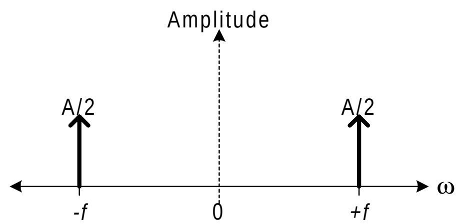


Figure 9.2. Positive and Negative Frequency

Interestingly, a sinusoidal signal can also be represented by extracting the real part of a periodic complex exponential:

$$\Re e\{Ae^{j\omega t}\} = \Re e\{A\cos(\omega t) + jA\sin(\omega t)\} = A\cos(\omega t)$$

Baseband Signals

A baseband signal is one that has a frequency spectrum which begins at 0Hz (DC) extending up to some maximum frequency. Though a baseband signal includes 0Hz, the magnitude at 0Hz may be 0 (i.e., no DC component). Furthermore, though a baseband signal usually extends up to some maximum frequency, an upper frequency limit is not a requirement. A baseband signal may extend to infinity. Most of the time, however, baseband signals are bandlimited and have some upper frequency bound, $f_{max}$. A bandlimited baseband signal can be represented graphically as a plot of signal amplitude vs. frequency. This is commonly referred to as a **spectrum**, or spectral plot. Figure 9.3 shows an example of a baseband spectrum. It's maximum magnitude, A, occurs at DC and its upper frequency bound is $f_{max}$.

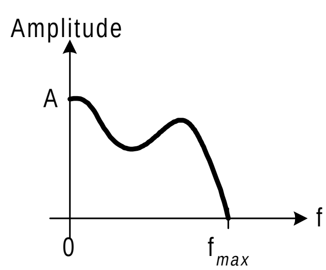


Figure 9.3. Single-Sided Bandlimited Baseband Spectrum

Note that only the positive portion of the frequency axis is shown. This type of spectrum is known as a single-sided spectrum. A more useful representation of a spectrum is one which includes the negative portion of the frequency axis is included. Such a spectrum is known as a double-sided spectrum. Figure 9.4 shows a double-sided representation of the spectrum of Figure 9.3.

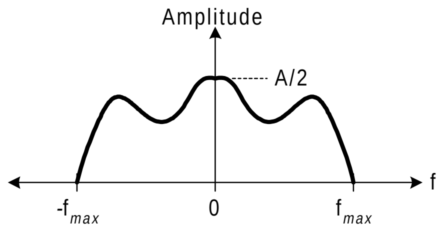


Figure 9.4.  Double-Sided Bandlimited Baseband Spectrum

Note that the signal amplitude is only 50% of the amplitude shown in the single-sided spectrum. This is because the negative frequency components are now being accounted for.  In the single-sided spectrum, the energy of the negative frequency components are simply added to the positive frequency components yielding an amplitude that is twice that of the double-sided spectrum.  Note, also, that the righthand side of the spectrum (+f) is mirrored about the 0Hz line creating the lefthand portion of the spectrum (-f).  Baseband spectra like the one above (which contain horizontal symmetry) represent **real** baseband signals.

There are also baseband signals that lack horizontal symmetry about the f = 0 line.  These are **complex** baseband signals.  An example of a complex baseband spectrum is shown below in Figure 9.5.

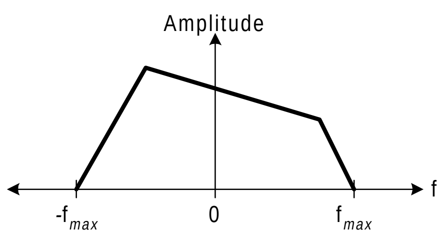


Figure 9.5.  Complex Baseband Spectrum

Note that the left side and right side of the spectrum are not mirror images.  This lack of symmetry is the earmark of a complex spectrum.  A complex baseband spectrum, since it is complex, can not be expressed as a real signal.  It can, however, be expressed as the complex sum of two real signals a(t) and b(t) as follows:

$$x(t) = a(t) + jb(t)$$

It turns out that it is not possible to propagate (transmit) a complex baseband spectrum in the real world.  Only real signals can be propagated.  However, a complex baseband signal can be transformed into a *real* bandpass signal through a process known as frequency translation or modulation (see the next section).  It is interesting to note that modulation can transform a complex baseband signal (which can not be transmitted) into a bandpass signal (which is a real and can be transmitted).  This concept is fundamental to all forms of signal transmission in use today (e.g., radio, television, etc.).

## Bandpass Signals

A bandpass signal can be thought of as a bandlimited baseband signal centered on some frequency, $f_c$ and it's negative, $-f_c$. Recall that a baseband signal is centered on f = 0. Bandpass signals, on the other hand, are centered on some non-zero frequency, $\pm f_c$, such that $|f_c| > 2f_{max}$. The value, $2f_{max}$, is the **bandwidth** (BW) of the bandpass signal. This is shown pictorially in Figure 9.6. It should be noted that there are two types of bandpass signals; those with a symmetric baseband spectrum and those with a nonsymmetric (or *quadrature*) baseband spectrum (depicted in Figure 9.6(a) and (b), respectively).

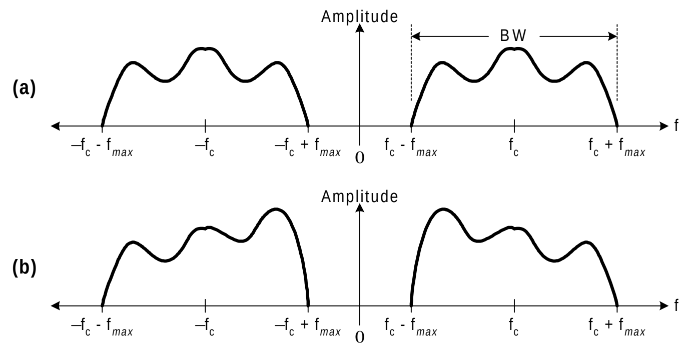


Figure 9.6. Bandpass Spectra

Mathematically, a bandpass signal can be represented in one of two forms. For case (a) we have:

$$x(t) = g(t)\mathit{cos}(\omega_c t)$$

where g(t) is the baseband signal and $\omega_c$ is radian frequency (radian frequency is related to natural frequency by the relationship $\omega = 2\pi f$). Note that multiplication of the baseband signal by $\cos(\omega_c t)$ translates the baseband signal so that it is centered on $\pm f_c$.

Alternatively, two baseband signals, $g_1(t)$ and $g_2(t)$ can be combined in quadrature fashion to produce the spectrum shown in case (b). Mathematically, this is represented by:

$$x(t) = g_1(t)\mathit{cos}(\omega_c t) + g_2(t)\mathit{sin}(\omega_c t)$$

Again, the important thing to note is that the bandpass signal is centered on $\pm f_c$. Furthermore, it should be mentioned that for quadrature bandpass signals $g_1(t)$ and $g_2(t)$ do not necessarily have to be different baseband signals. There is nothing to restrict $g_1(t)$ from being equal to $g_2(t)$. In which case, the same baseband signal is combined in quadrature fashion to create a bandpass signal.

## Modulation

The concept of bandpass signals leads directly to the concept of modulation. In fact, the translation of a spectrum from one center frequency to another is, by definition, *modulation*. The bandpass equations in the previous section indicate that multiplication of a signal, g(t), by a sinusoid (of frequency $\omega_c$) is all that is necessary to perform the modulation function. The only difference between the concept of a bandpass signal and modulation, is that with modulation it is not necessary to restrict g(t) to a baseband signal. g(t) can, in fact, be another bandpass signal. In which case, the bandpass signal, g(t), is translated in frequency by $\pm f_c$. It is this very property of modulation which enables the process of **demodulation**. Demodulation is accomplished by multiplying a bandpass signal centered on $f_c$ by *cos*($\omega_c$). This shifts the bandpass signal which was centered on $f_c$ to a baseband signal centered on 0Hz and a bandpass signal centered on $2f_c$. All that is required to complete the transformation from bandpass to baseband is to filter out the $2f_c$-centered component of the spectrum.

Figure 9.7 below shows a functional block diagram of the two basic modulation structures. Figure 9.7(a) demonstrates sinusoidal modulation, while (b) demonstrates quadrature modulation. There are, in fact, variations on these two themes which produce specialized forms of modulation. These include present- and suppressed-carrier modulation as well as double- and single-sideband modulation.

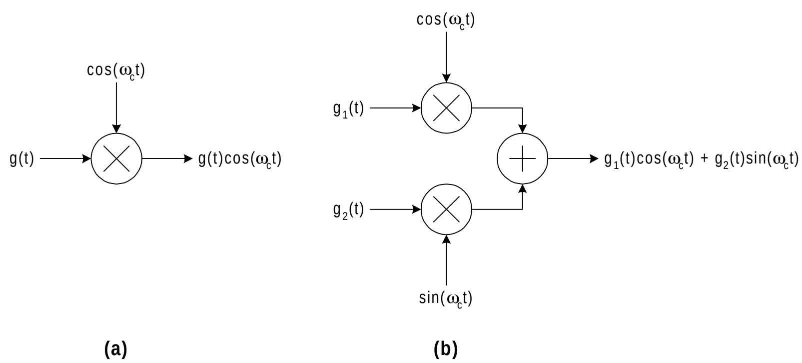


Figure 9.7. Basic Modulation Structures

The preceding sections set the stage for an understanding of **digital modulation**. The reader should be familiar with the basics of sampled systems in order to fully understand digital modulation. This chapter is written with the assumption that the reader has fundamental knowledge of sampled systems, especially with respect to the implications of the Nyquist theorem.

Digital modulation is the discrete-time counterpart to the continuous-time modulation concepts discussed above. Instead of dealing with an analog waveform, x(t), we are dealing with instantaneous samples of an analog waveform, x(n). It is implied that n, an integer index, bears a one-to-one correspondence with sampling time instants. That is, if T represents the time interval between successive samples, then nT represents the time instants at which samples are taken. The similarity between continuous- and discrete-time signals becomes obvious when their forms are written together. For example, consider a sinusoidal signal:

$x(t) = A\cos(\omega t)$ *continuous-time*
$x(n) = A\cos(\omega nT)$ *discrete-time*

The main difference in the discrete-time signal is that there are certain restrictions on $\omega$ and T as a result of the Nyquist theorem. Specifically, T must be less than $\pi/\omega$ (remember $\omega = 2\pi f$ and T is the sampling period). Since x(n) is a series of instantaneous samples of x(t), then x(n) can be represented as a series of numbers, where each number is the instantaneous value of x(t) at the instants, nT.

The importance of this idea is paramount in understanding digital modulators. In the analog world modulation is achieved by multiplying together continuous-time waveforms using specialized analog circuits. However, in the digital world it is possible to perform modulation by simply manipulating sequences of numbers. A purely numeric operation.

The modulation structures for continuous-time signals can be adapted for digital modulators. This is shown in Figure 9.8.

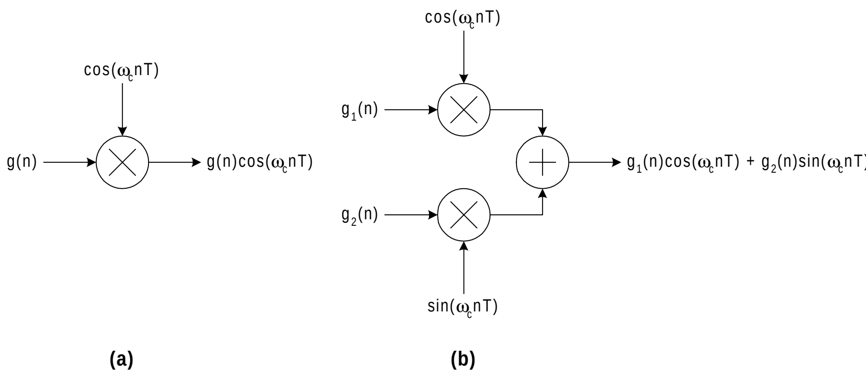


Figure 9.8. Basic Digital Modulation Structures

Here, g(n), $g_1$(n), $g_2$(n), sin($\omega_c$nT) and cos($\omega_c$nT) are sequences of numbers. The multipliers and adders are logic elements (digital multipliers and adders). Their complexity is a function of the number of bits used to represent the samples of the input waveforms. This is not much of an issue in theory. However, when implemented in hardware, the number of circuit elements can grow very quickly. For example, if the digital waveforms are represented by 8-bit numbers, then multipliers and adders capable of handling 8-bit words are required. If, on the other hand, the digital waveforms are represented by IEEE double-precision floating point numbers (64-bits), then the multipliers and adders become very large structures.

It is in the environment of digital modulators that DDS technology becomes very attractive. This is because a DDS directly generates the series of numbers that represent samples of a sine and/or cosine waveform. DDS-based digital modulator structures are shown in Figure 9.9.

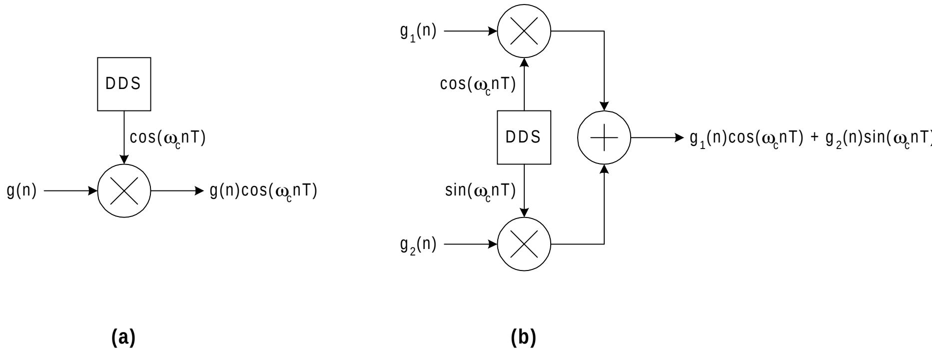


Figure 9.9. Basic DDS Modulation Structures

System Architecture and Requirements

The basic DDS modulation structures described in the previous section are very simplistic. There are a number of elements not shown that are required to actually make a digital modulator work. The most critical element is a clock source. A DDS can only generate samples if it is driven by a sample clock. Thus, without a system clock source a digital modulator is completely nonfunctional.

Also, since the digital modulator shown above amounts to nothing more than a "number cruncher", its output (a continuous stream of numbers) is of little use in a real application. As with any sampled system, there is the prevailing requirement that the digital number stream be converted back to a continuous-time waveform. This will require the second major element of a digital modulator, a digital-to-analog converter (**DAC**).

A more complete DDS modulator is shown in Figure 9.10. In order to keep things simple, only the sinusoidal modulator form is shown. The extension to a quadrature modulator structure is trivial.

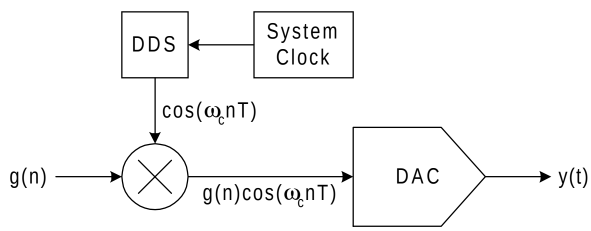


Figure 9.10. DDS Modulator

At first glance, the DDS modulator appears to be quite simple. However, there is a subtle requirement that makes digital modulation a bit more difficult to implement. The requirement is that g(n) must consist of the samples of a signal sampled at the same frequency as the DDS sample rate. Otherwise, the multiplier stage is multiplying values that have been sampled at completely different instants.

To help illustrate the point, suppose

$$g(n) = \mathit{cos}[2\pi(1\text{kHz})nT_1]$$

where $T_1$ is 0.00025 (0.25ms). Thus, g(n) can be described as a 1kHz signal sampled at 4kHz (1/ $T_1$). Suppose, also, that the DDS output is

$$\text{DDS} = \mathit{cos}[2\pi(3\text{kHz})nT_2]$$

where $T_2$ is 0.0001 (0.1ms). This means that the DDS output is a 3kHz signal sampled at 10kHz (1/ $T_2$). In the DDS modulator, the value of n (the sample index) for the multiplier is the same for both of the inputs as well as the output. So, for a specific value of n, say n=10 (i.e., the 10$^{th}$ sample) the time index for the DDS is $nT_2$, which is 0.001(1ms). Clearly, $nT_1 \neq nT_2$ (2.5ms ≠ 1ms). Thus, at time index n=10, DDS time is 1ms while g(n) time is 2.5ms. The bottom line is that the product, g(n)$cos(\omega_c nT)$, which is the output of the multiplier, is not at all what it is expected to be, because the time reference for g(n) is not the same as that of $cos(\omega_c nT)$.

The “equal sample rate” requirement is the primary design consideration in a digital modulator. If, in a DDS modulator system, the source of the g(n) signal operates at a sample rate other than the DDS clock, then steps must be taken to correct the sample rate discrepancy. The DDS modulator design then becomes an exercise in multirate digital signal processing (**DSP**). Multirate DSP requires an understanding of the techniques involved in what is known as interpolation and decimation. However, interpolation and decimation require some basic knowledge of digital filters. These topics are covered in the following sections.

## Digital Filters

Digital filters are the discrete-time counterpart to continuous-time analog filters. Some groundwork was laid on analog filters in Chapter 4. The analog filter material presented in Chapter 4 was purposely restricted to lowpass filters, because the topic was antialias filtering (which is inherently a lowpass application). However, analog filters can be designed with highpass, bandpass, and bandstop characteristics, as well.

The same is true for digital filters. Unlike an analog filter, however, a digital filter is not made up of physical components (capacitors, resistors, inductors, etc.). It is a logic device that simply performs as a number cruncher. Subsequently, a digital filter does not suffer from the component drift problems that can plague analog filters. Once the digital filter is designed, its operation is predictable and reproducible. The second feature is particularly attractive when two signal paths require identical filtering.

There are two basic classes of digital filters; one is the **FIR** (Finite Impulse Response) filter and the other is the **IIR** (Infinite Impulse Response) filter. From the point of view of filter design, the FIR is the simpler of the two to work. However, from the point of view of hardware requirements, the IIR has an advantage. It usually requires much less circuitry than an FIR with the same basic response. Unfortunately, the IIR has the potential to become unstable under certain conditions. This property often excludes it from being designed into systems for which the input signal is not well defined.

## FIR Filters

Fundamentally, an FIR is a very simply structure. It is nothing more than a chain of delay, multiply, and add stages. Each stage consists of an input and output data path and a fixed coefficient (a number which serves as one of the multiplicands in the multiplier section). A single stage is often referred to as a **tap**. Figure 9.11 below shows a simple 2-tap FIR (the input signal to the FIR filter is considered to be one of the taps).

 

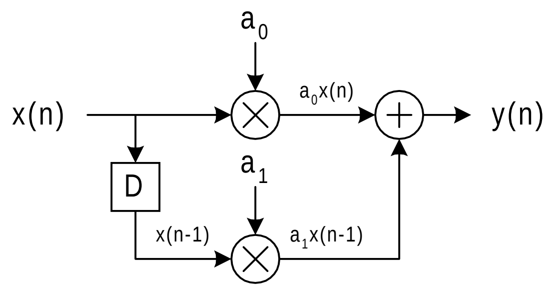


Figure 9.11. Simple FIR Filter

In this simple filter we have input, x(n), which is a number sequence that represents the sampled values of an input signal. The input signal is fed to two places. First, it is fed to a multiplier which multiplies the current sample by coefficient, $a_0$ ($a_0$ is simply a number). The product, $a_0x(n)$ is fed to one input of the output adder. The second place that the input signal is routed to is a delay stage (the "D" box). This stage simply delays x(n) by one sample. Note that the output of the delay stage is labeled, x(n-1). That is, it is the value of the previous sample. To better understand this concept consider the general expression, x(n-k), where k is an integer that represents time relative to the present. If k=0, then we have x(n), which is the value of x(n) at the present time. If k=1, then we have x(n-1), which is the the value of x(n) one sample earlier in time. If k=2, then we have x(n-2), which is the value of x(n) two samples earlier in time, etc. Returning to the figure, the output of the delay stage feeds a multiplier with coefficient $a_1$. So, the output of this multiplier is $a_1x(n-1)$, which is the value of x(n) one sample earlier in time multiplied by $a_1$. This product is fed to the other input of the output adder. Thus, the output of the FIR filter, y(n), can be represented as:

$$y(n) = a_0x(n) + a_1x(n-1)$$

This means, that at any given instant, the output of this FIR is nothing more than the sum of the current sample of x(n) and the previous sample of x(n) each multiplied by some constant value ($a_0$ and $a_1$, respectively). Does this somehow relate back to the concept of filtering a signal? Yes, it does, but in order to see the connection we must make a short jump to the world of the z-transform.

The z-transform is related to the Fourier transform in that, just as the Fourier transform allows us to exchange the time domain for the frequency domain (and vice-versa), the z-transform does exactly the same for sampled signals. In other words, the z-transform is merely a special case of the Fourier transform in that it is specifically restricted to discrete time systems (i.e., sampled systems).

By applying the z-transform to the previous equation, it can be shown that the transfer function, H(z), is given as:

$$H(z) = a_0z^0 + a_1z^{-1} = a_0 + a_1z^{-1}$$

Where $z = e^{j\omega}$, $\omega = 2\pi f/F_s$ and $F_s$ is the sample frequency. Notice that transfer function, through the use of the z-transform, results in the conversion of x(n-1) to $z^{-1}$. In fact, this can be generally extended to the case, x(n-k), which is converted to $z^{-k}$ (k is the number of unit delays).

Now let's apply some numbers to our example to make things a little more visual. Suppose we employ a sample rate of 10kHz ($F_s$=10kHz) and let $a_0=a_1=0.5$. Now, compute H(z) for different frequencies (f) and plot the magnitude of H(z) as a function of frequency:

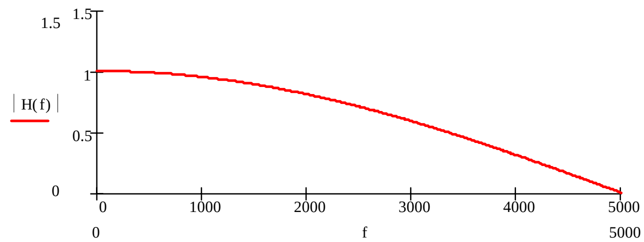


Figure 9.12. FIR Frequency Response for $a_0 = a_1 = 0.5$

Clearly, this constitutes a filter with a lowpass response. Note that the frequency axis only extends to 5kHz (½$F_s$) because in a sampled system the region from ½$F_s$ to $F_s$ is a mirror image of the region 0 to $F_s$ (a consequence of the Nyquist limit). So, from the plot of the transfer function above, we see that at an input frequency of 3kHz this particular FIR filter allows only about 60% of the input signal to pass through. Thus, if the input sample sequence {x(n)} represents the samples of a 3kHz sinewave, then the output sample sequence {y(n)} represents the samples of a 3kHz sinewave also. However, the output signal exhibits a 40% reduction in amplitude with respect to the input signal.

The simple FIR can be extended to provide any number of taps. Figure 9.13 shows an example of how to extend the simple FIR to an N-tap FIR.

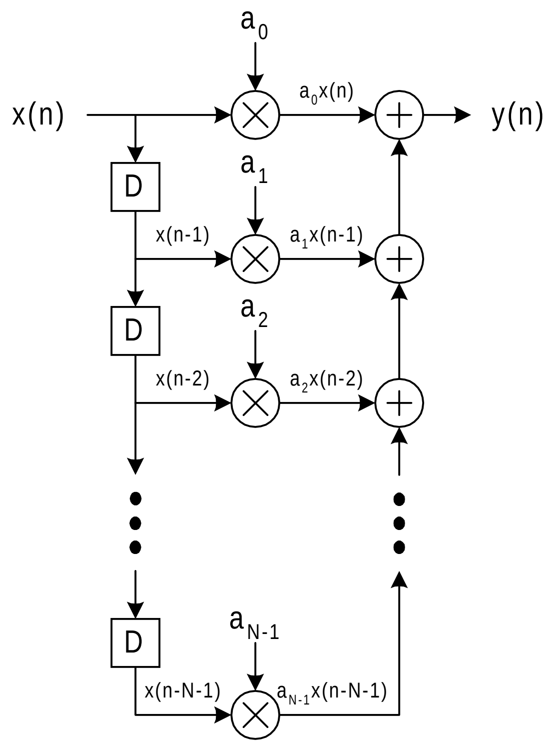


Figure 9.13. An N-tap FIR Filter

It should be mentioned that it is possible to have coefficients that are equal to 0 for a particular type of filter response. In which case, the need for the multiply and add stage for that particular coefficient is superfluous. All that is required is the unit delay stage. Also, for a coefficient of 1, a multiply is not needed. The value of the delayed sample can be passed on directly to the adder for that stage. Similarly, for a coefficient of –1, the multiply operation can be replace by a simple negation. Thus, when an FIR is implemented in hardware, coefficients that are 0, 1, or –1 help to reduce the circuit complexity by eliminating the multiplier for that stage.

An analysis of the behavior of the FIR is easy to visualize if y(n) is initially at 0 and the x(n) sequence is chosen to be 1,0,0,0,0,0… From the diagram, it should be apparent that, with each new sample instant, the single leading 1 in x(n) will propagate down the delay chain. Thus, on the first sample instant $y(n)=a_0$. On the second sample instant $y(n)=a_1$. On the third sample instant $y(n)=a_2$, etc. This process continues through the N-th sample, at which time the output of each delay stage is 0 and remains at 0 for all further sample instants. This means that y(n) becomes 0 after the N-th sample as well, and remains at 0 thereafter.

This leads to two interesting observations. First, the input of a single 1 for x(n) constitutes an impulse. So, the output of the FIR for this input constitutes the impulse response of the FIR (which is directly related to its frequency response). Notice that the impulse response only exists for N samples. Hence the name, ***Finite*** Impulse Response filter. Furthermore, the impulse response also demonstrates that an input to the FIR will require exactly N samples to propagate through the entire filter before its effect is no longer present at the output.

It should be apparent that increasing N will increase the overall delay through the FIR. This can be a problem in systems that are not delay tolerant. However, there is a distinct advantage to increasing N; it increases the sharpness of the filter response.

Examination of the above figure leads to an equation for the output, y(n), in terms of the input, x(n) for an FIR of arbitrary length, N. The result is:

$$y(n) = a_0x(n) + a_1x(n-1) + a_2x(n-2) + \ldots + a_{N-1}x(n-N-1)$$

Application of the z-transform leads to the transfer function, H(z), of an N-tap FIR filter.

$$H(z) = a_0 + a_1z^{-1} + a_2z^{-2} + \ldots + a_{N-1}z^{-(N-1)}$$

Earlier, it was mentioned that increasing the number of taps in an FIR increases the sharpness of the filter response. This is shown in Figure 9.14, which is a plot of a 10-tap FIR. All coefficients ($a_0$– $a_9$) are equal to 0.1. Notice how the rolloff of the filter is much sharper than the 2-tap simple FIR shown earlier. The first null appears at 1kHz for the 10-tap FIR, instead of at 5kHz for the 2-tap FIR.

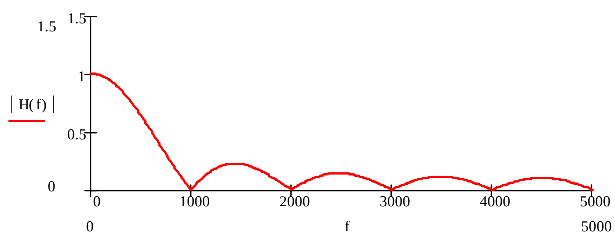


Figure 9.14. 10-Tap FIR Frequency Response for $a_{0\text{-}9} = 0.1$

It should be mentioned here, that the response of the FIR is completely determined by the coefficient values. By choosing the appropriate coefficients it is possible to design filters with just about any response; lowpass, highpass, bandpass, bandreject, allpass, and more.

<u>IIR Filters</u>

With an understanding of FIR filters, the jump to IIR's is fairly simple. The difference between an IIR and an FIR is feedback. An IIR filter has a feedback section with additional delay, multiply, and add stages that tap the output signal, y(n). A simple IIR structure is shown in Figure 9.15.

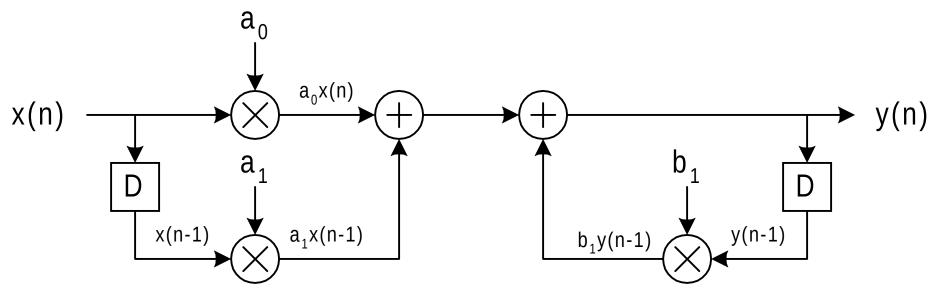


Figure 9.15. Simple IIR Filter

Notice that the left side is an exact copy of the simple 2-tap FIR. This portion of the IIR is often referred to as the **feedforward** section, because delayed samples of the input sequence are fed forward to the adder block. The right hand portion is a **feedback** section. The feedback is a delayed and scaled version of the output signal, y(n). Note that the feedback is summed with the output of the feedforward section. The existence of feedback in an IIR filter makes a dramatic difference in the behavior of the filter. As with any feedback system, stability becomes a significant issue. Improper choice of the coefficients or unanticipated behavior of the input signal can cause instability in the IIR. The result can be oscillation or severe clipping of the output signal. The stability issue may be enough to exclude the use of an IIR in certain applications.

The impulse response exercise that was done for the FIR is also a helpful tool for examining the IIR. Again, let y(n) be initially at 0 and x(n) be the sequence: 1,0,0,0…. After 2 sample instants, the left side of the IIR will have generated two impulses of height $a_0$ and $a_1$ (exactly like the simple FIR). After generating two impulses, the left side will output nothing but 0's. The right side is a different story. Every new sample instant modifies the value of y(n) by the previous value of y(n), recursively. The result is that y(n) continues to output values indefinitely, even though the input signal is no longer present. So, a single impulse at the input results in an infinitely long sequence of impulses at the output. Hence the name, <u>*Infinite*</u> Impulse Response filter.

It should be pointed out that *infinite* is an ideal concept. In a practical application, an IIR can only be implemented with a finite amount of numeric resolution. Especially, in fixed point applications, where the data path may be restricted to, say, 16-bit words. With finite numeric resolution, values very near to 0 get truncated to a value of 0. This leads to IIR performance that deviates from the ideal. This is because an ideal IIR would continue to output ever decreasing values, gradually approaching 0. However, because of the finite resolution issue, there comes a point at which small values are assigned a value of 0, eventually terminating the gradually decreasing signal to 0. This, of course, marks the end of the "infinite" impulse response.

The structure of the simple IIR leads to an expression for y(n) as:

$$y(n) = a_0x(n) + a_1x(n-1) + b_1y(n-1)$$

This shows, that at any given instant, the output of the IIR is a function of both the input signal, x(n), and the output signal, y(n). By applying the z-transform to this equation, it can be shown that the frequency response, H(z), may be expressed as:

$$H(z) = (a_0 + a_1 z^{-1}) / (1 - b_1 z^{-1})$$

Again, $z = e^{j\omega}$, $\omega = 2\pi f/F_s$ and $F_s$ is the sample frequency. As an example, let $F_s$=10kHz, $a_0 = a_1 = 0.1$, and $b_1 = 0.85$. Computing H(z) for different frequencies (f) and plotting the magnitude of H(z) as a function of frequency yields:

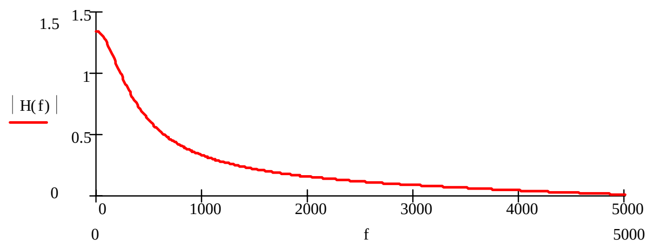


Figure 9.16. IIR Frequency Response for $a_0 = a_1 = 0.1$ and $b_1 = 0.85$

As with the FIR, it is a simple matter to expand the simple IIR to a multi-tap IIR. In this case, however, the left side of the IIR and the right side are not restricted to have the same number of delay taps. Thus, the number of "a" and "b" coefficients may not be the same. Also, as with the case for the FIR, certain filter responses may result in coefficients equal to 0, 1 or -1. Again, this leads to a circuit simplification when the IIR is implemented in hardware. Figure 9.17 shows how the simple IIR may be expanded to a multi-tap IIR. Note that the number of "a" taps (N) is not the same as the number of "b" taps (M). This accounts for the possibility a different number of coefficients in the feedforward ("a" coefficients) and feedback ("b" coefficients) sections.

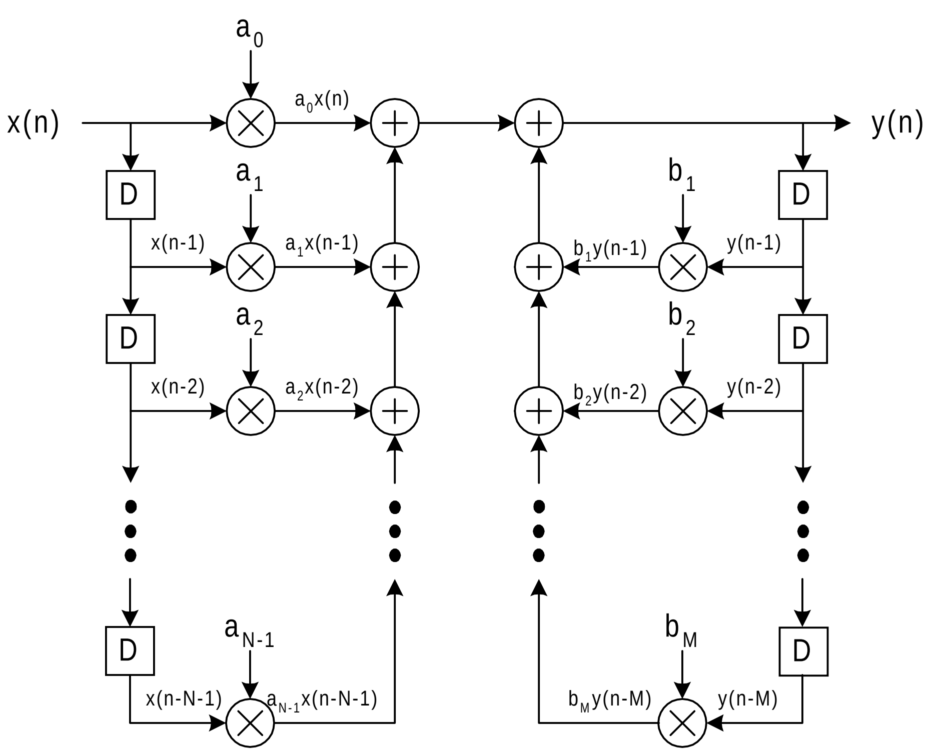


Figure 9.17. A Multi-tap IIR Filter

Examination of Figure 9.17 leads to an equation for the output, y(n), in terms of the input, x(n) for a multi-tap IIR. The result is:

$$y(n) = a_0x(n) + a_1x(n-1) + a_2x(n-2) + \dots + a_{N-1}x(n-N-1) + b_1y(n-1) + b_2y(n-2) + \dots + b_My(n-M)$$

Application of the z-transform leads to the general transfer function, H(z), of a multi-tap IIR filter with N feedforward coefficients and M feedback coefficients.

$$H(z) = (a_0 + a_1z^{-1} + a_2z^{-2} + \dots + a_{N-1}z^{-(N-1)}) / (1 - b_1z^{-1} - b_2z^{-2} - \dots - b_Mz^{-M})$$

## Multirate DSP

Multirate DSP is the process of converting data sampled at one rate ($F_{s1}$) to data sampled at another rate ($F_{s2}$). If $F_{s1} > F_{s2}$, the process is called **decimation**. If $F_{s1} < F_{s2}$, the process is called **interpolation**.

The need for multirate DSP becomes apparent if one considers the following example. Suppose that there are 1000 samples of a 1kHz sinewave stored in memory, and that these samples were acquired using a 10kHz sample rate. This implies that the time duration of the entire sample set spans 100ms (1000 samples at 10000 samples/second). If this same sample set is now clocked out of memory at a 100kHz rate, it will require only 10ms to exhaust the data. Thus, the 1000 samples of data clocked at a 100kHz rate now look like a 10kHz sinewave instead of the original 1kHz sinewave. Obviously, if the sample rate is changed but the data left unchanged, the result is undesireable. Clearly, if the original 10kHz sampled data set represents a 1kHz signal, then it would be desireable to have the 100kHz sampled output represent a 1kHz signal, as well. In order for this to happen, the original data must somehow be modified. This is the task of multirate DSP.

## Interpolation

The aforementioned problem is a case for interpolation. The function of an interpolator is to take data that was sampled at one rate and change it to new data sampled at a higher rate. The data must be modified in such a way that when it is sampled at a higher rate the original signal is preserved. A pictorial representation of the interpolation process is shown in Figure 9.18.

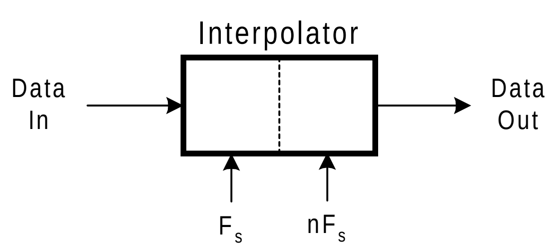


Figure 9.18. A Basic Interpolator

Notice that the interpolator has two parts. An input section that samples at a rate of $F_s$ and an output section that samples at a rate of $nF_s$, where *n* is a positve integer greater than 1 (non-integer multirate DSP is covered later). The structure of the basic interpolator indicates that for every input sample there will be *n* output samples. This begs the question: What must be done to the original data so that, when it is sampled at the higher rate, the original signal is preserved?

One might reason intuitively that, if *n*-1 zeroes are inserted between each of the input samples (*zero-stuffing*), then the output data would have the desired characteristics. After all, adding nothing (0) to something shouldn't change it. This turns out to be a pretty good start to the interpolation process, but it's not the complete picture.

The reason becomes clear if the process is examined in the frequency domain. Consider the case of interpolation by 3 ($n$ = 3), as shown in Figure 9.19.

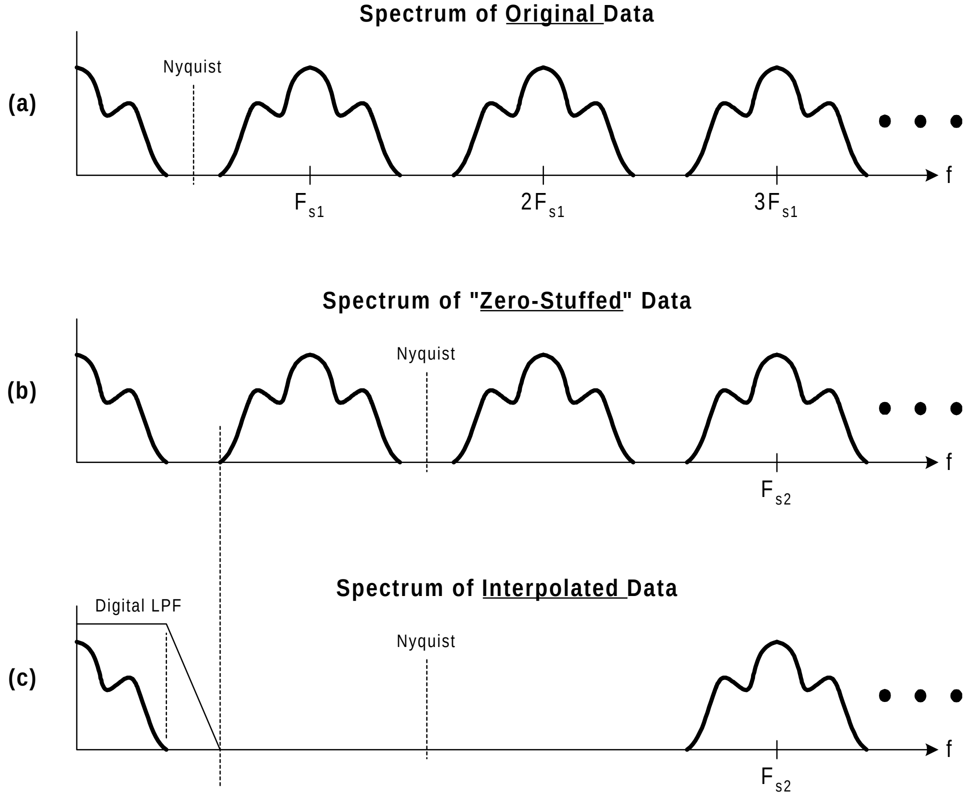


Figure 9.19. Frequency Domain View of Interpolation

The original data is sampled at a rate of $F_{s1}$. The spectrum of the original data is shown in Figure 9.19 (a). The Nyquist frequency for the original data is indicated at ½$F_{s1}$.

With an interpolation rate of 3 the output sample rate, $F_{s2}$, is equal to 3$F_{s1}$ (as is indicated by comparing Figure 9.19 (a) and (c)). The interpolation rate of 3 implies that 2 zeroes be stuffed between each input sample. The spectrum of the zero-stuffed data is shown in Figure 9.19 (b). Note that zero-stuffing retains the spectrum of the original data, but the sample rate has been increased a factor of 3 relative to the original sample rate. The new sample rate also results in a new Nyquist frequency at ½$F_{s2}$, as shown.

It seems as though the interpolation process may be considered complete after zero-stuffing, since the two spectra match. But such is not the case. Here is why. The original data was sampled at a rate of $F_{s1}$. Keep in mind, however, that Nyquist requires the bandwidth of the sampled signal to be less than ½$F_{s1}$ (which it is). Furthermore, all of the information carried by the original data resides in the baseband spectrum at the far left side of Figure 9.19 (a). The spectral images centered at integer multiples of $F_{s1}$ are a byproduct of the sampling process and carry no additional information.

Upon examination of Figure 9.19 (b) it is apparent that the Nyquist zone (the region from 0 to ½$F_{s2}$) contains more than the single-sided spectrum at the far left of Figure 9.19 (a). In fact, it contains one of the images of the original sampled spectrum. Therein lies the problem: The Nyquist zone of the 3x sampled spectrum contains a different group of signals than the Nyquist zone of the original spectrum. So, something must be done to ensure that, after interpolating, the Nyquist zone of the interpolated spectrum contains exactly the same signals as the original spectrum.

Figure 9.19 (c) shows the solution. If the zero-stuffed spectrum is passed through a lowpass filter having the response shown in Figure 9.19 (c), then the result is exactly what is required to reproduce the baseband spectrum of the original data for a 3x sample rate. Furthermore, the filter can be employed as a lowpass FIR filter. Thus, the entire interpolation process can be done in the digital domain.

Decimation

The function of a decimator is to take data that was sampled at one rate and change it to new data sampled at a lower rate. The data must be modified in such a way that when it is sampled at the lower rate the original signal is preserved. A pictorial representation of the decimation process is shown in Figure 9.20 below.

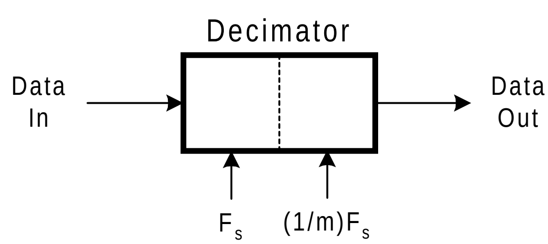


Figure 9.20. A Basic Decimator

Notice that the decimator has two parts. An input section that samples at a rate of $F_s$ and an output section that samples at a rate of $(1/m)F_s$, where *m* is a positve integer greater than 1 (non-integer multirate DSP is covered later). The structure of the basic decimator indicates that for every *m* input samples there will be 1 output sample. This begs a similar question to that for interpolation: What must be done to the original data so that, when it is sampled at the lower rate, the original signal is preserved?

One might reason intuitively that, if every *m*-th sample of the input signal is picked off (ignoring the rest), then the output data would have the desired characteristics. After all, this constitutes a sparsely sampled version of the original data sequence. So, doesn't this reflect the same information as the original data? The short answer is, "No". The reason is imbedded in the ramifications of the Nyquist criteria in regard to the original data. The complete answer becomes apparent if the process is examined in the frequency domain, which is the topic of Figure 9.20.

 

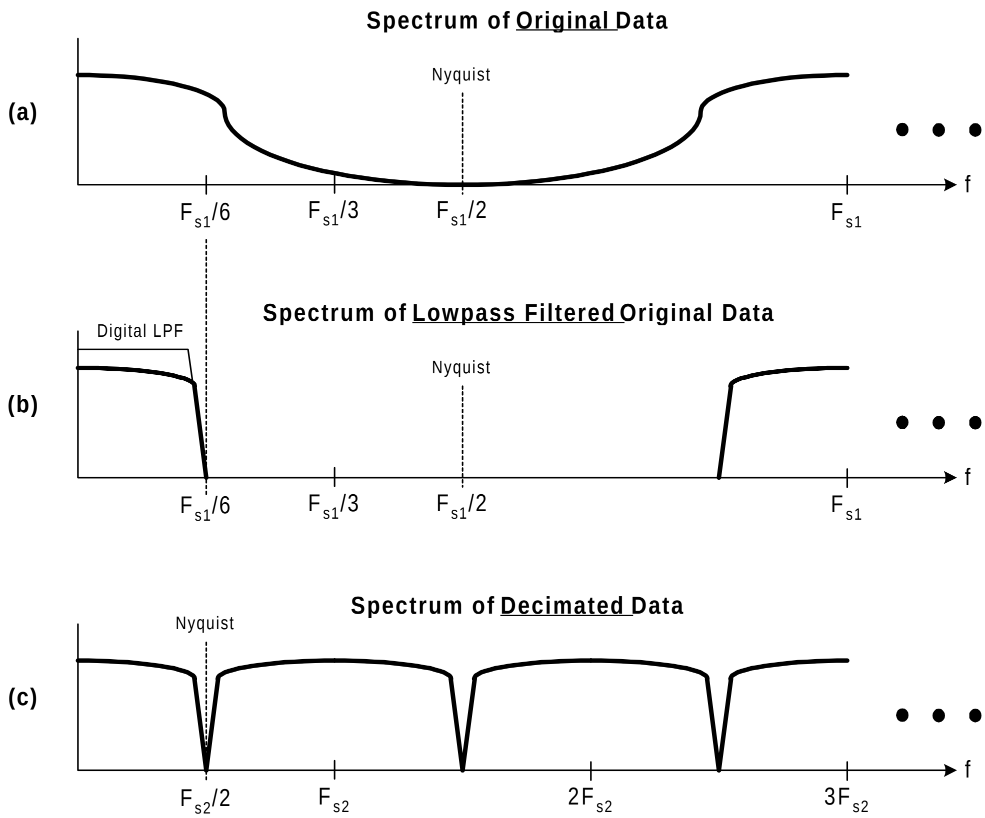


Figure 9.21. Frequency Domain View of Decimation

The matter at hand is that the decimation process is expected to translate the spectral information in Figure 9.21 (a) such that it can be properly contained in Figure 9.21 (c). The problem is that the Nyquist region of Figure 9.21 (a) is three times wider than the Nyquist region of Figure 9.21 (c). This is a direct result of the difference in sample rates. There is no way that the complete spectral content of the Nyquist region of Figure 9.21 (a) can be placed in the Nyquist region of Figure 9.21 (c). This brings up the cardinal rule of decimation: *The bandwidth of the data prior to decimation must be confined to the Nyquist bandwidth of the lower sample rate*. So, for decimation by a factor of *m*, the original data must reside in a bandwidth given by $F_s/(2m)$, where $F_s$ is the rate at which the original data was sampled. Thus, if the original data contains valid information in the portion of the spectrum beyond $F_s/(2m)$, decimation is not possible. Such would be the case in this example if the portion of the orginal data spectrum beyond $F_{s1}/6$ in Figure 9.21 (a) was part of the actual data (as opposed to noise or other interfering signals).

Assuming that the spectrum of the original data meets the decimation bandwidth requirement, then the first step in the decimation process is to lowpass filter the original data. As with the interpolation process, this filter can be a digital FIR filter. The second step is to pick off every *m*-th sample using the lower output sample rate. The result is the spectrum of Figure 9.21 (c).

## Rational n/m Rate Conversion

The interpolation and decimation processes described above allow only integer rate changes. It is often desirable to achieve a rational rate change. This is easily accomplished by cascading two (or more) interpolate/decimate stages. Using the same convention as in the previous sections, interpolation by *n* and decimation by *m*, rational *n/m* rate conversion can be accomplished as shown in Figure 9.22 below. It should be pointed out that **it is imperative that the interpolation process precede the decimation process** in a multirate converter. Otherwise, the bandwidth of the original data must be confined to $F_s/(2m)$, where $F_s$ is the sample rate of the original data.

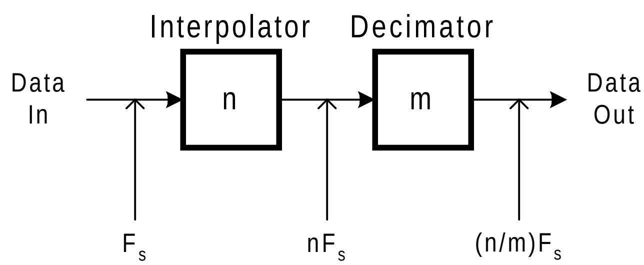


Figure 9.22. A Basic *n/m* Rate Converter

## Multirate Digital Filters

It has been shown previously that interpolators and decimators both require the use of lowpass filters. These can be readily designed using FIR design techniques. However, there are two classes of digital filters worth mentioning that are particularly suited for multirate DSP: The *polyphase FIR* filter and the *Cascaded Integrator-Comb* (**CIC**) filter.

### *Polyphase FIR*

The polyphase FIR is particularly suited to interpolation applications. Recall that the interpolation process begins with the insertion of *n*-1 zeroes between each input sample. The zero-stuffed data is then clocked out at the higher output rate. Next, the zero-stuffed data, which is now sampled at a higher rate, is passed through a lowpass FIR to complete the process of interpolation.

However, rather than performing this multi-step process, it is possible to do it all in one step by designing an FIR as follows. First, design the FIR with the appropriate frequency response, knowing that the FIR is to be clocked at the higher sample rate ($nF_s$). This will yield an FIR with a certain number of taps, T. Now add *n*-1 "delay" stages to each tap section of the FIR. A delay-only stage is the equivalent of a tap with a coefficient of 0. Thus, there are *n*-1 delay and multiply-by-0 operations interlaced between each of the original FIR taps. This arrangement maintains the desired frequency response characteristic while simultaneously making provision for zero-stuffing (a consequence of the additional *n*-1 multiply-by-0 stages). The result is an FIR with *n*T taps, but (*n*-1)T of the taps are multiply-by-0 stages.

It turns out that an FIR with this particular arrangement of coefficients can be implemented very efficiently in hardware by employing the approprate architecture. The special structure so created is a polyphase FIR. In operation, the polyphase FIR accepts each input sample *n* times because it is operating at the higher sample rate. However, *n*-1 of the samples are converted to zeros due to the additional delay-only stages. Thus, we get the zero-stuffing and filtering operation all in one package.

Cascaded Integrator-Comb (CIC)

The CIC filter is combination of a comb filter and an integrator. Before moving directly into the operation of the CIC filter, a presentation of the basic operation of a comb filter and an integrator ensues. The comb filter is a type of FIR consisting of a delay and add stages. The integrator can be thought of as a type of IIR, but without a feedforward section. A simple comb and integrator block diagram are shown in Figure 9.23.

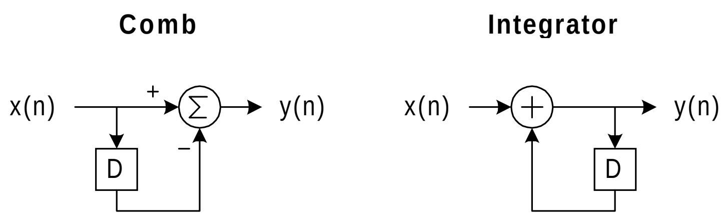


Figure 9.23. The Basic Comb and Integrator

Inspection of the comb and integrator architecture reveals an interesting feature: No multiply operations are required. In the integrator there is an implied multiply-by-one in the feedback path. In the comb, there is an implied multiply-by-negative-one in the feedforward path, but this can be implemented by simple negation, instead. The absence of multipliers offers a tremendous reduction in circuit complexity and, thereby, a significant savings in hardware compared to the standard FIR or IIR architectures. This feature is what makes a CIC filter so attractive.

In terms of frequency response, the comb filter acts as a notch filter. For the simple comb shown above, two notches appear; one at DC and the other at $F_s$ (the sample rate). However, the comb can be slightly modified by cascading delay blocks together. This changes the number of notches in the comb response. In fact, K delay blocks produces K+1 equally spaced notches. Of the K+1 notches, one occurs at DC and another at $F_s$. The remainder (if any) are equally spaced between DC and $F_s$.

The integrator, on the other hand, acts as a lowpass filter. The response characteristics of each appears in Figure 9.24 below with frequency normalized to $F_s$. The comb response shown is that for K=1 (a single delay block).

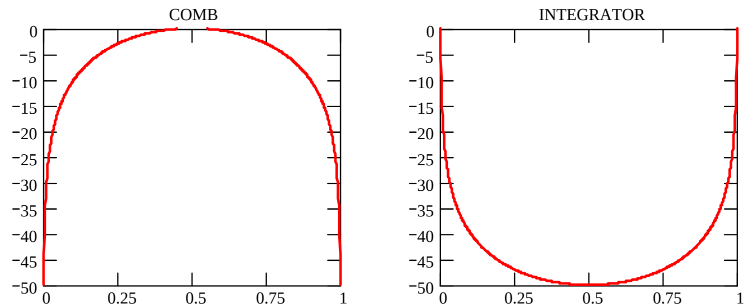


Figure 9.24. Frequency Response of the Comb and Integrator

The CIC filter is constructed by cascading the integrator and comb sections together. In order to perform integer sample rate conversion, the integrator is operated at one rate while the comb is operated at an integer multiple of the integrator rate. The beauty of the CIC filter is that it can function as either an interpolator or a decimator, depending on how the comb and integrator are connected. A simple CIC interpolator and decimator block diagram is shown in Figure 9.25. Note that the integrator and comb sections are operated at different sample rates for both the interpolator and decimator CIC filters.

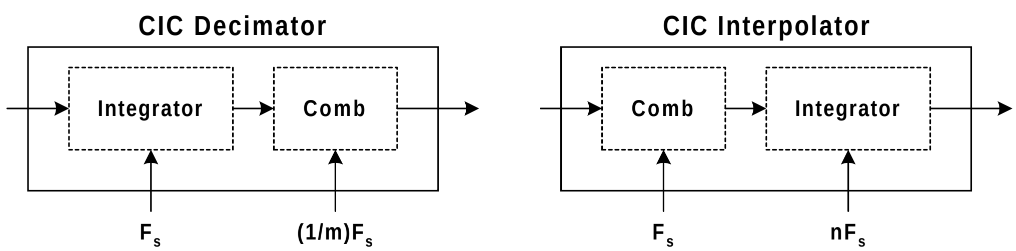


Figure 9.25. CIC Decimator and Interpolator

In the case of the interpolating version of the CIC, a slight modification to the basic architecture of its integrator stage is required which is not apparent in the simple block diagram above. Specifically, the integrator must have the ability to do zero-stuffing at its input in order to affect the increase in sample rate. For every sample provided by the comb, the integrator must insert n-1 zeroes.

The frequency response of the basic CIC filter is shown in Figure 9.26 below for an interpolation (or decimation) factor of 2; that is, n = 2 (or m = 2) as specified in the previous figure.

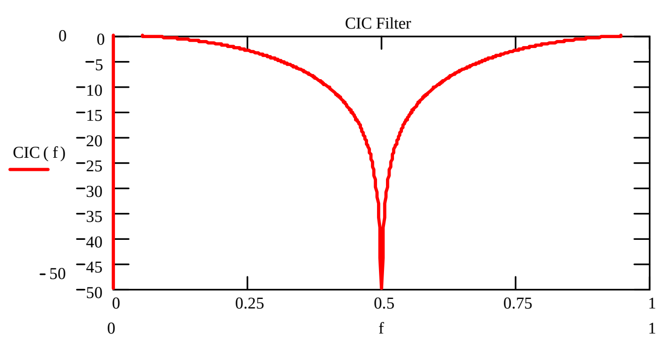


Figure 9.26. Frequency Response of the Basic CIC Filter

Inspection of the CIC frequency response bears out one of its potential problems: attenuation distortion. Recall that the plot above depicts a CIC with a data rate change factor of 2. The f=0.5 point on the horizontal access marks the frequency of the lower sample rate signal. The Nyquist frequency of the lower data rate signal is, therefore, at the f=0.25 point. Note that there is approximately 5dB of loss across what is considered the passband of the low sample rate signal. This lack of flatness in the passband can pose a serious problem in certain data communication applications and might actually preclude the use of the CIC filter for this very reason.

There are ways to overcome the attenuation problem, however. One method is to precede the CIC with an inverse filter that compensates for the attenuation of the CIC over the Nyquist bandwidth. A second method is to ensure further bandlimiting of the low sample rate signal so that its bandwidth is confined to the far lefthand portion of the CIC response where it is nearly flat.

It should be pointed out that the response curve of Figure 9.26 is for a *basic* CIC response. A CIC filter can be altered to change its frequency response characteristics. There are two methods by which this can be accomplished.

One method is to cascade multiple integrator stages together and multiple comb stages together. An example of a basic CIC interpolator modified to incorporate a triple cascade is shown in Figure 9.27.

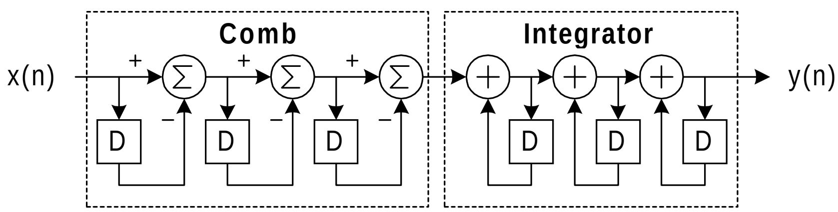


Figure 9.27. Triple Cascade CIC Interpolator

The second method is to add multiple delays in the comb section. An example of a basic CIC with one additional comb delay is shown in Figure 9.28.

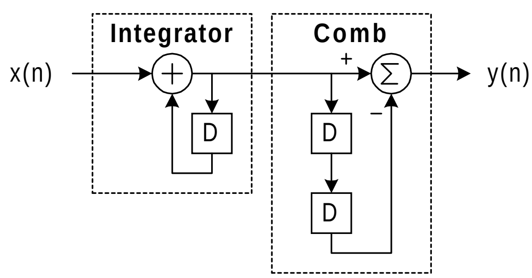


Figure 9.28. Double Delay CIC Decimator

The effect of each of these options is shown in Figure 9.29 along with the basic CIC response for comparison. In each case, a rate change factor of 2 is employed.

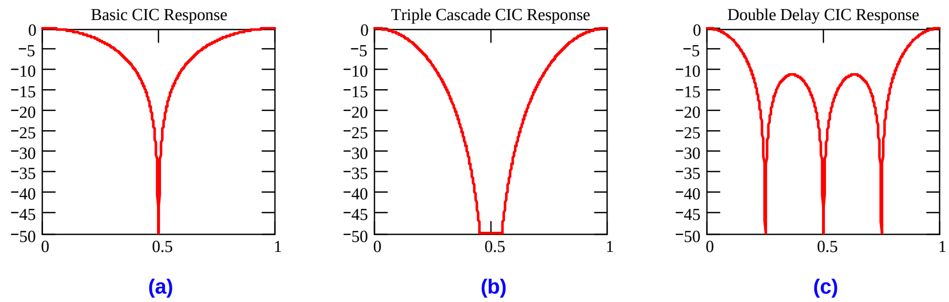


Figure 9.29. Comparison of Modified CIC Filter Responses

Note that cascading of sections (b) steepens the rate of attenuation. This results in more loss in the passband, which must be considered in applications where flatness in the passband is an issue. On the other hand, adding delays to the comb stage results in an increase in the number of zeroes in the transfer function. This is indicated in (c) by the additional null points in the frequency response, as well as increased passband attenuation. In the case of cascaded delays, consideration must be given to both the increased attenuation in the passband as well as reduced attenuation in the stopband. In addition, a combination of both methods can be employed allowing for a variety of possible responses.

## Clock and Input Data Synchronization Considerations

In digital modulator applications it is important to maintain the proper timing relationship between the data source and the modulator. Figure 9.30 below shows a simple system block diagram of a digital modulator. The primary source of timing for the modulator is the clock which drives the DDS. This establishes the sample rate of the SIN and COS carrier signals of the modulator. Any samples propagating through the data pathway to the input of the modulator must occur at the same rate at which the carrier signal is sampled. It is important that samples arriving at the modulator input do so in a one-to-one correspondence with the samples of the carrier. Otherwise, the signal processing within the modulator is not carried out properly.

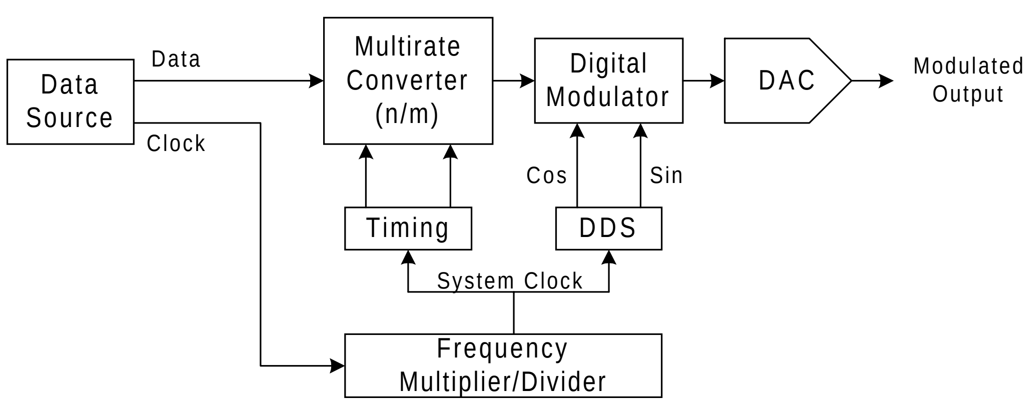


Figure 9.30. Generic Digital Modulator Block Diagram

Since the multirate converter must provide a rational rate conversion (i.e., the input and output rates of the converter must be expressible as a proper fraction), then the original data rate must be rationally related to the system clock. That is, the system clock must operate at a factor of *n/m* times the data clock (or an integer multiple of *n/m*). In simpler modulator systems, the rate converter is a direct integer interpolator or decimator. In such instances, the clock multiplier is a simple integer multiplier or divider. Thus, the data source and system clock rates are related by an integer ratio instead of by a fractional ratio.

There are basically two categories of digital modulators when it comes to timing and synchronization requirements; **burst mode** modulators and **continuous mode** modulators. A burst mode modulator transmits data in packets; that is, groups of bits are transmitted as a block. During the time interval between bursts, the transmitter is idle. A continuous mode modulator, on the other hand, transmits a constant stream of data with no breaks in transmission.

It should quickly become apparent that the timing requirements of a burst mode modulator are much less stringent than those of a continuous mode modulator. The primary reason is that a burst mode modulator is only required to be synchronized with the data source over the duration of a data burst. This can be accomplished with a gating signal that synchronizes the modulator with the beginning of the burst. During the burst interval, the system clock can regulate the timing. As long as the system clock does not drift significantly relative to the data clock during the burst interal, the system operates properly. Naturally, as the data burst length increases the timing requirements on the system clock become more demanding.

In a continuous mode modulator the system clock must be synchronized with the data source at all times. Otherwise, the system clock will eventually drift enough to miss a bit at the input. This, of course, will cause an error in the data transmission process. Thus, an absolutely robust method must be implemented in order to ensure that the system clock and data clock remain continuously synchronized. Otherwise, transmission errors are all but guaranteed. Needless to say, great consideration must be given to the system timing requirements of any digital modulator application, burst or continuous, in order to make sure that the system meets its specified bit error rate (**BER**) requirements.

## Data Encoding Methodologies and DDS Implementations

There is a wide array of methods by which data may be encoded before being modulated onto a carrier. In this section, a variety of data encoding schemes are addressed and their implementation in a DDS system. It is in this section that a sampling of Analog Devices' DDS products will be showcased to demonstrate the relative ease by which many applications may be realized. It should be understood that this section is by no means an exhaustive list of data encoding schemes. However, the concepts presented here may be extrapolated to cover a broad range of data encoding and modulation formats.

### FSK Encoding

Frequency shift keying (**FSK**) is one of the simplest forms of data encoding. Binary 1's and 0's are represented as two different frequencies, $f_0$ and $f_1$, respectively. This encoding scheme is easily implemented in a DDS. All that is required is to arrange to change the DDS frequency tuning word so that $f_0$ and $f_1$ are generated in sympathy with the pattern of 1's and 0's to be transmitted. In the case of ADI's **AD9852**, **AD9853**, and **AD9856** the process is greatly simplified. In these devices the user programs the two required tuning words into the device before transmission. Then a dedicated pin on the device is used to select the appropriate tuning word. In this way, when the dedicated pin is driven with a logic 0 level, $f_0$ is transmitted and when it is driven with a logic 1 level, $f_1$ is transmitted. A simple block diagram is shown in Figure 9.31 that demonstrates the implementation of FSK encoding. The concept is representative of the DDS implementation used in ADI's synthesizer product line.

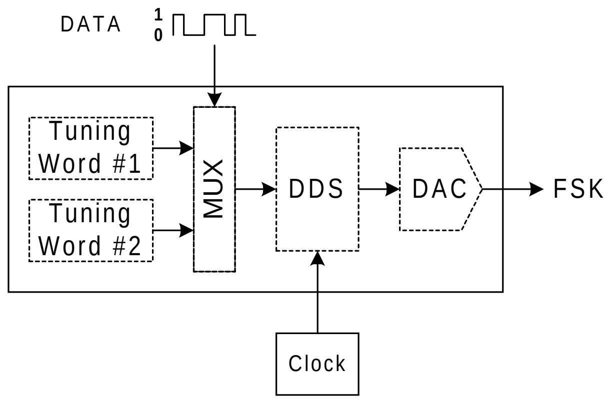


Figure 9.31. A DDS-based FSK Encoder

In some applications, the rapid transition between frequencies in FSK creates a problem. The reason is that the rapid transition from one frequency to another creates spurious components that can interfere with adjacent channels in a multi-channel environment. To alleviate the problem a method known as **ramped FSK** is employed. Rather than switching immediately between frequencies, ramped FSK offers a gradual transition from one frequency to the other. This significantly reduces the spurious signals associated with standard FSK. Ramped FSK can also be implemented using DDS techniques. In fact, the **AD9852** has a built in Ramped FSK feature which offers the user the ability to program the ramp rate. A model of a DDS Ramped FSK encoder is shown in Figure 9.32.

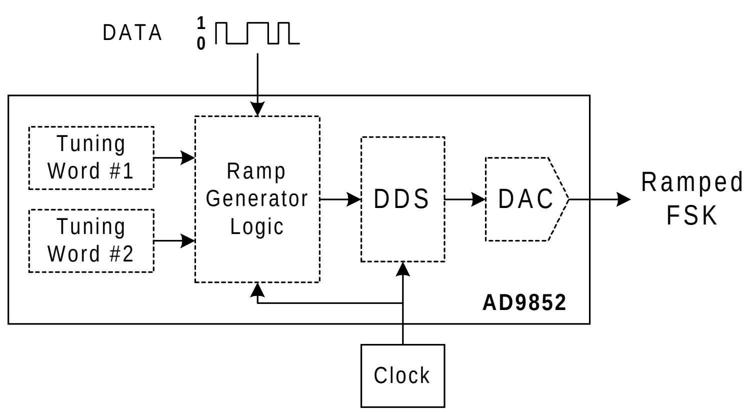


Figure 9.32. A DDS-based Ramped FSK Encoder

A variant of FSK is **MFSK** (multi-frequency FSK). In MFSK, $2^B$ frequencies are used (B>1). Furthermore, the data stream is grouped into packets of B bits. The binary number represented by any one B-bit word is mapped to one of the $2^B$ possible frequencies. For example, if B=3, then there would be $2^3$, or 8, possible combinations of values when grouping 3 bits at a time. Each combination would correspond to 1 of 8 possible output frequencies, $f_0$ through $f_7$.

## PSK Encoding

Phase shift keying (**PSK**) is another simple form of data encoding. In PSK the frequency of the carrier remains constant. However, binary 1's and 0's are used to phase shift the carrier by a certain angle. A common method for modulating phase on a carrier is to employ a quadrature modulator. When PSK is implemented with a quadrature modulator it is referred to as **QPSK** (quadrature PSK).

The most common form of PSK is **BPSK** (binary PSK). BPSK encodes 0° phase shift for a logic 1 input and a 180° phase shift for a logic 0 input. Of course, this method may be extended to encoding groups of B-bits and mapping them to $2^B$ possible angles in the range of 0° to 360° (B>1). This is similar to MFSK, but with phase as the variable instead of frequency. To distinguish between variants of PSK, the binary range is used as a prefix to the PSK label. For example, B=3 is referred to as 8PSK, while B=4 is referred to as 16PSK. These may also be referred to as 8QPSK and 16QPSK depending on the implementation.

Since PSK is encoded as a certain phase shift of the carrier, then decoding a PSK transmission requires knowledge of the absolute phase of the carrier. This is called **coherent** detection. The receiver must have access to the transmitter's carrier in order to decode the transmission. A workaround for this problem is differential PSK, or **DPSK** and can be extented to **DQPSK**. In this scheme, the change in phase of the carrier is dependent on the value of the previously sent bit (or symbol). Thus, the receiver need only identify the relative phase of the first symbol. All subsequent symbols can then be decoded based on phase changes relative to the first. This method of detection is referred to as **noncoherent** detection.

PSK encoding is easily implemented with ADI's synthesizer products. Most of the devices have a separate input register (a *phase register*) that can be loaded with a phase value. This value is directly added to the phase of the carrier without changing its frequency. Modulating the contents of this register modulates the phase of the carrier, thus generating a PSK output signal. This method is somewhat limited in terms of data rate, because of the time required to program the phase register. However, other devices in the synthesizer family, like the **AD9853**, eliminate the need to program a phase register. Instead, the user feeds a serial data stream to a dedicated input pin on the device. The device automatically parses the data into 2-bit symbols and then modulates the carrier as required to generate QPSK or DQPSK.

## QAM Encoding

Quadrature amplitude modulation (**QAM**) is an encoding scheme in which both the amplitude and phase of the carrier are used to convey information. By its name, QAM implies the use of a quadrature modulator. In QAM the input data is parsed into groups of B-bits called symbols. Each symbol has $2^B$ possible states. Each state can be represented as a combination of a specific phase value and amplitude value. The number of states is usually used as a prefix to the QAM label to identify the number of bits encoded per symbol. For example, B=4 is referred to as 16QAM. Systems are presently in operation that use B=8 for 256QAM.

The assignment of the possible amplitude and phase values in a QAM system is generally optimized in such a way to maximize the probability of accurate detection at the receiver. The tool most often used to display the amplitude and phase relationship in a QAM system is the **constellation diagram** or **I-Q diagram**. A typical 16QAM constellation is shown in Figure 9.33.

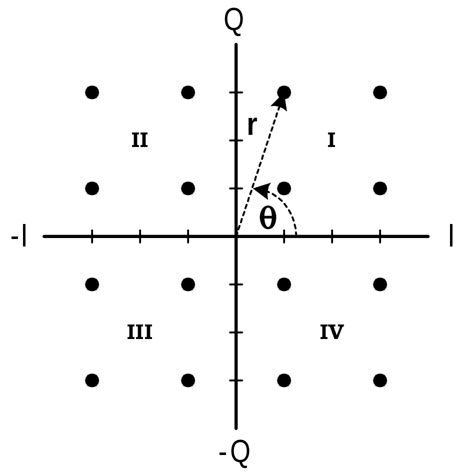


Figure 9.33. 16QAM Constellation

Each dot in Figure 9.33 represents a particular symbol (4-bits). The constellation of Figure 9.33 uses I and Q values of ±1 and ±3 to locate a dot. For example, in Quadrant I, the dots are represented by the (I,Q) pairs: (1,1), (1,3), (3,1) and (3,3). In Quadrant II, the dots are mapped as (-1,1), (-1,3), (-3,1) and (-3,3). Quadrants III and IV are mapped in similar fashion.

Note that each dot can also be represented by a vector starting at the origin and extending to a specific dot. Thus, an alternative means of interpreting a dot is by amplitude, r, and phase, θ. It is the r and θ values that define the way in which the carrier signal is modulated in a QAM signal. In this instance, the carrier can assume 1 of 3 amplitude values combined with 1 of 12 phase values.

To reiterate, each amplitude and phase combination represents one of 16 possible 4-bit symbols. Furthermore, the combination of a particular phase and amplitude value defines exactly one of the 16 dots. Additionally, each dot is associated with a specific 4-bit pattern, or symbol.

It is a simple matter to extend this concept to symbols with more bits. For example, 256QAM encodes 8-bit symbols. This yields a rather dense constellation of 256 dots. Obviously, the denser the constellation, the more resolution that is required in both phase and amplitude. This implies a greater chance that the receiver produces a decoding error. Thus, a higher QAM number means a less tolerant system to noise.

The available signal to noise ratio (SNR) of a data transmission link has a direct impact on the BER of the system. The available SNR and required BER place restrictions on the type of encoding scheme that can be used. This becomes apparent as the density of the QAM encoding scheme increases. For example, some data transmission systems have a restriction on the amount transmit power that can be employed over the link. This sets an upper bound on the SNR of the system. It can be shown that SNR and BER follow an inverse exponential relationship.  That is, BER increases exponentially as SNR decreases.  The density of the QAM encoding scheme amplifies this exponential relationship.  Thus, in a power limited data link, there comes a point at which increasing QAM density yields a BER that is not acceptable.

As with the other encoding schemes, a differential variant is possible.  This leads to differential QAM (**DQAM**).  Standard QAM requires coherent detection at the receiver, which is not always pssible.  DQAM solves this problem by encoding symbols in a manner that is dependent on the preceding symbol.  This frees the detector from requiring a reference signal that is in absolute phase with the transmitter's carrier.

The **AD9853** is capable of direct implementation of the 16QAM and D16QAM encoding.  The **AD9856**, on the other hand, can be operated in any QAM mode.  However, the **AD9856** is a general purpose quadrature modulator that accepts 12-bit 2's complement numbers as I and Q input values.  Thus, the burden is on the user to parse the input data stream and convert it to bandlimited I and Q data before passing it to the **AD9856**.

FM Modulation

Frequency modulation (**FM**) is accomplished by varying the frequency of a carrier in sympathy with another signal, the message.  In a DDS system, FM is implemented by rapidly updating the frequency tuning word in a manner prescribed by the amplitude of the message signal.  Thus, performing FM modulation with a DDS requires extra hardware to sample the message signal, compute the appropriate frequency tuning word from the sample, and update the DDS.

A broadcast quality FM transmitter using the **AD9850** DDS is described in an application note (AN-543) from Analog Devices.  The application note is included in the Appendix.

Quadrature Up-Conversion

Figure 9.30 shows an example of a generic modulator.  A quadrature up-converter is a specific subset of the generic modulator and is shown in Figure 9.34.

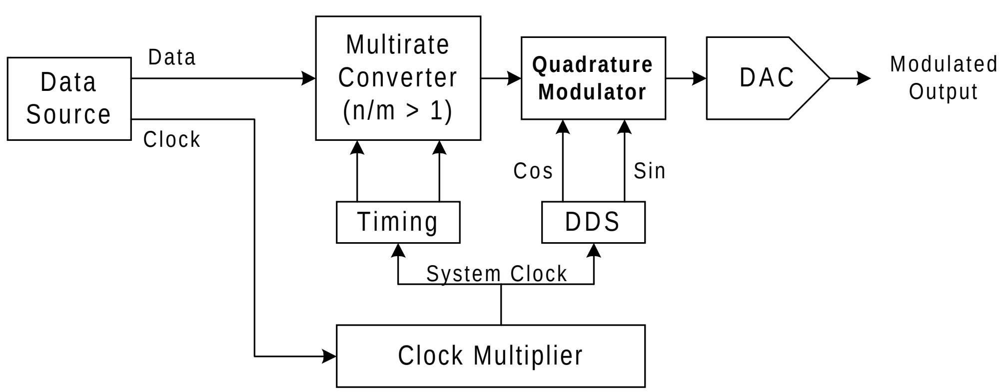


Figure 9.34.  Quadrature Up-Converter

In an up-converter, the incoming data is intended to be broadcast on a carrier which is several times higher in frequency than the data rate.  This automatically implies that the multirate converter ratio, n/m, is always greater than 1.  Similarly, the system clock is generated by a frequency multiplier since the data rate is less than the carrier frequency, which must be less than the DDS sample rate. Also, as its name implies, the modulator portion of the up-converter is of the quadrature variety.

ADI offers an integrated quadrature up-converter, the **AD9856**. The **AD9856** can be operated with a system clock rate up to 200MHz. The clock multiplier, multirate converter, DDS, digital quadrature modulator, and DAC are all integrated on a single chip. A simplified block diagram of the **AD9856** appears in Figure 9.35.

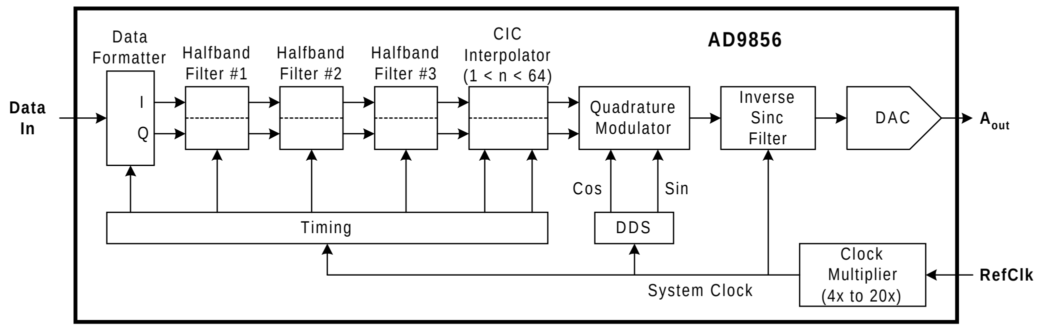


Figure 9.35. **AD9856** 200MHz Quadrature Digital Up-Converter

The halfband filters are polyphase FIR filters with an interpolation rate of 2 (n=2). Halfband Filter #3 can be bypassed, which yields a cumulative interpolation rate of either 4 or 8 via the halfband filters. The CIC interpolator offers a programmable range of 2 to 63. Thus, the overall interpolation rate can be as low as 8 or as high as 504. The input data consists of 12-bit 2's complement words that are delivered to the device as alternating I and Q values. The device also sports an Inverse Sinc Filter which can be bypassed, if so desired. The Inverse Sinc Filter is an FIR filter that provides pre-emphasis to compensate for the sinc rolloff characteristic associated with the DAC.

**References:**

[1] Gibson, J. D., 1993, Principles of Digital and Analog Communications, Prentice-Hall, Inc.

[2] Ifeachor, E. C. and Jervis, B. W., 1996, Digital Signal Processing: A Practical Approach, Addison-Wesley Publishing Co.

[3] Lathi, B. P., 1989, Modern Digital and Analog Communication Systems, Oxford University Press

[4] Oppenheim, A. V. and Willsky, A. S., 1983, Signals and Systems, Prentice-Hall, Inc.

[5] Proakis, J. G. and Manolakis, D. G., 1996, Digital Signal Processing: Principles, Algorithms, and Applications, Prentice-Hall, Inc.