Signals, Samples and Stuff: A DSP Tutorial (Part 1)

"DSP" is a buzzword of the '90s. Have you wondered what it's all about? This article begins an ongoing QEX *walk through the forest of DSP.*

By Doug Smith, KF6DX/7

What *is* all this DSP business, anyway? How does it really work? Certain crucial concepts are used in digital radio design. In this first article of a series, I'll describe these concepts in some detail. All of them are important to understand in the execution of a DSP transceiver. In the second article, we'll investigate an actual design. We'll look at *architectural* issues that impact decisions made during development, and we'll review the final configuration. In the third article, we'll survey some advanced DSP techniques such as adaptive filtering and special demodulation methods.

The Mathematics of Complex Signals

DSP implementations of radio transceiver functions compel designers to reexamine the mathematics that describe them. Computers and microprocessors are good at crunching numbers, but one thing stands out about them: they do exactly what they are told! Therefore, if we expect a DSP system to generate an SSB signal, we'd better know those calculations to perform, and those to avoid.

PO Box 4074
Sedona, AZ 86340
e-mail **dsmith@sedona.net**

Real and Complex Signals

Let's start with the job of taking a real input signal, say the audio from a microphone, and converting it to an SSB signal that can be transmitted over the air. We have to translate its frequency upward by the carrier frequency's value, and in so doing, preserve the spectral content. If we wish to produce an upper sideband (USB) signal, we want the carrier and lower sideband to be suppressed by as much as possible. We'll explore how the mathematics of complex signals achieve this using the so-called "phasing method" of SSB generation.

Of course, we generate SSB signals in other ways, such as the filter method. Because DSP makes it easy to build broadband phase shifters, and because the use of complex signals gives rise to a great deal of flexibility and precision, the phasing method has dominated DSP SSB generation to date.

Complex signals are not generally well understood, and

they're an obstacle in the way of those who wish to grasp the concepts. The idea of negative frequency is especially troublesome. A real signal, such as a cosine wave, is normally thought of as a positive frequency. It can be seen on an oscilloscope or spectrum analyzer in the positive-frequency domain. It can be transmitted and detected normally. We shall see, however, that such a signal actually consists of positive *and* negative frequencies when examined in the complex domain.

Our real cosine wave embodies the relation:

$$X_t = \cos \omega t \qquad \text{(Eq 1)}$$

where $\omega = 2\pi f$, and t is time. In the complex domain, the cosine wave is really the sum of two complex signals:

$$X_t = \frac{1}{2}\left[(\cos \omega t + j \sin \omega t) + (\cos \omega t - j \sin \omega t)\right] \qquad \text{(Eq 2)}$$

This signal has both positive- and negative-frequency components! The left-hand term is positive, the right-hand negative—the imaginary terms cancel and the real terms reinforce to make the equation true. In the complex plane, where the real part is one axis and the imaginary part the other, this signal can be represented as two vectors rotating in opposite directions. See Fig 1.

While this depiction is beautiful and elegant to a mathematician, what does it really mean to you and me? Well, it means that signals represented in complex form can have a one-sided spectrum, ie, having only positive or only negative frequencies. This is useful as we mix our signal upward to its final RF position.

Frequency Translation and Complex Mixing

We see that if we were able to translate the spectrum of our cosine wave—with its symmetrical positive and negative parts—upward in frequency far enough, we'd have two positive frequencies separated by twice the original signal frequency. For a real signal, this is exactly what happens when it's applied to an analog mixer! Both the sum and difference frequencies are generated, and the amplitude of each is half the original amplitude. So it's no coincidence that a mixer's conversion loss is about 6 dB—precisely what physics predicts.

We now invoke an identity discovered by Euler, which states that:

$$e^{jwt} = \cos \omega t + j \sin \omega t \qquad \text{(Eq 3)}$$

This is a more convenient notation for complex sinusoidal signals. So now, our real cosine wave takes the form:

$$\cos \omega t = \frac{1}{2}\left(e^{j\omega t} + e^{-j\omega t}\right) \qquad \text{(Eq 4)}$$

When we mix this with a real carrier, say

$$y_t = \cos \omega_0 t \qquad \text{(Eq 5)}$$

we get the product of the two inputs:

$$(\cos \omega_0 t)(\cos \omega t) = \left[\frac{\left(e^{j\omega_0 t} + e^{-j\omega_0 t}\right)}{2}\right]\left[\frac{\left(e^{j\omega t} + e^{-j\omega t}\right)}{2}\right] \qquad \text{(Eq 6)}$$

$$= \frac{\left[e^{j(\omega_0+\omega)t} + e^{-j(\omega_0+\omega)t}\right] + \left[e^{j(\omega_0-\omega)t} + e^{-j(\omega_0-\omega)t}\right]}{4} \qquad \text{(Eq 7)}$$

$$= \frac{1}{2}\left[\cos(\omega_0 + \omega)t + \cos(\omega_0 - \omega)t\right] \qquad \text{(Eq 8)}$$

The multiplication of the two cosine waves results in a frequency translation of their two-sided spectra. Now, let's consider what happens when we mix or multiply two complex, one-sided signals—as opposed to real, two-sided signals—together.

SSB Generation

We now present a new function, I_t, to represent the amplitude of the microphone audio versus time. This is a real signal with a two-sided spectrum. It's possible to convert this real signal to a complex signal—with only positive-frequency components—by generating a quadrature signal, Q_t, wherein all frequencies are phase-shifted by 90° from I_t, and treating I_t and Q_t as a complex, or *analytic,* pair. The signal $I_t + jQ_t$ contains only positive frequencies. The negative frequencies cancel each other, while the positive frequencies reinforce. The function that phase shifts all the frequency components by 90° is called a *Hilbert transform.* This would be difficult to achieve in the analog world, but is easy in DSP.

We now multiply this analytic signal by a complex oscillator:

$$Y_t = e^{j\omega_0 t} \qquad \text{(Eq 9)}$$

and the translated signal takes the form:

$$e^{j\omega_0 t}(I_t + jQ_t) = (\cos \omega_0 t + j \sin \omega_0 t)(I_t + jQ_t) \qquad \text{(Eq 10)}$$

$$= (I_t \cos \omega_0 t - Q_t \sin \omega_0 t) + j(I_t \sin \omega_0 t + Q_t \cos \omega_0 t) \qquad \text{(Eq 11)}$$

Since we're interested in transmitting this SSB signal via a single path over the air, we only have to compute the real part. This "half-complex" mixer is implemented as shown in Fig 2. It produces a real USB signal. This is, in fact, the good old phasing method.

Properties of SSB Signals

Since the amplitude of the carrier is constant, the amplitude of the SSB signal can be specified as some function of the modulating signal. If we think of the converted microphone audio signal $I_t + jQ_t$ as a vector, it follows that its length is equal to the instantaneous amplitude:

$$A_t = \left(I_t^2 + Q_t^2\right)^{\frac{1}{2}} \qquad \text{(Eq 12)}$$

The phase of the signal is the instantaneous angle of this rotating vector:

$$\phi_t = \tan^{-1}\left(\frac{Q_t}{I_t}\right) \qquad \text{(Eq 13)}$$

Now we can rewrite the real part of Eq 11 as:

$$\Re\left[e^{j\omega_0 t}(I_t + jQ_t)\right] = A_t \cos(\omega_0 t + \phi_t) \qquad \text{(Eq 14)}$$

This shows that an SSB signal is a hybrid of both amplitude and phase modulation. Also, note that having defined the amplitude and phase of the baseband signal in Eq 12 and 13 above, we can write:

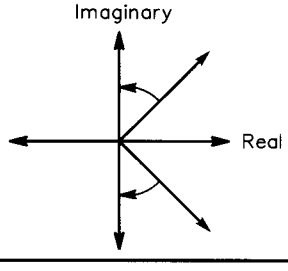

Fig 1—Vector representation of a real cosine wave.

$$(I_t + jQ_t) = A_t e^{j\phi_t} \quad \text{(Eq 15)}$$

directly relating the envelope and phase to the analytic baseband signal. The amplitude and phase of this analytic signal are identical to those of the SSB wave of Eq 11.

DSP Receiver / Exciter Architectures for SSB

With all this under our belts, we're ready to examine how these concepts apply to actual digital SSB receivers and exciters. While we won't explore all the ways DSP transceivers can be configured, and won't yet dive into all the minute detail, we will look at several designs to illustrate the principles.

Fig 3 is the block diagram of a digital exciter. The audio is normally low-pass filtered before sampling to remove components above half the sampling frequency of the analog-to-digital converter (ADC). A sampling frequency that is one quarter of the output sampling frequency is convenient. The audio passes through two band-pass filters, one of which incorporates a 90° phase shift. This converts the real signal to a complex signal $I_t + jQ_t$ having only positive frequencies. The filters are identical in frequency response and differ only in their phase responses.

The construction details of these filters will be treated later, under "Digital Filters." For now, it's sufficient to say that it's easy to build a frequency-independent DSP phase shifter—a fantasy in the analog world! The key concept here is that the real and imaginary parts of the analytic signal $I_t + jQ_t$ are handled separately in DSP.

The analytic signal is translated to the output frequency

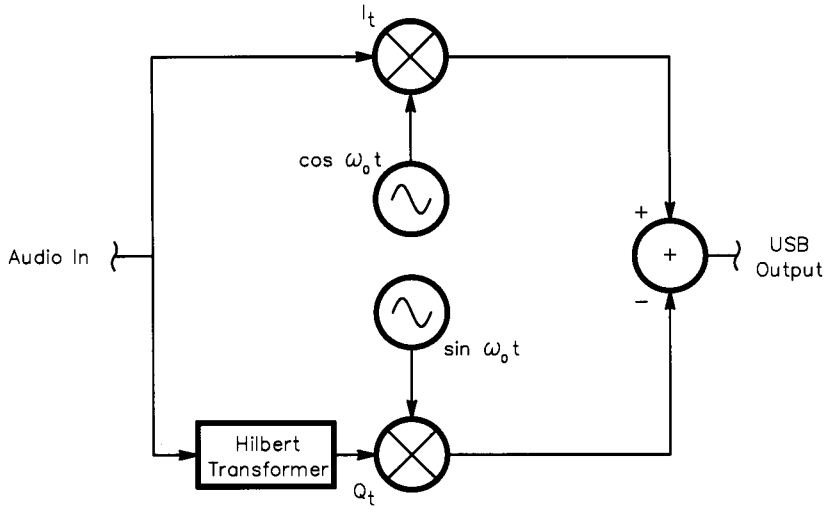

Fig 2—Block diagram of a half-complex mixer.

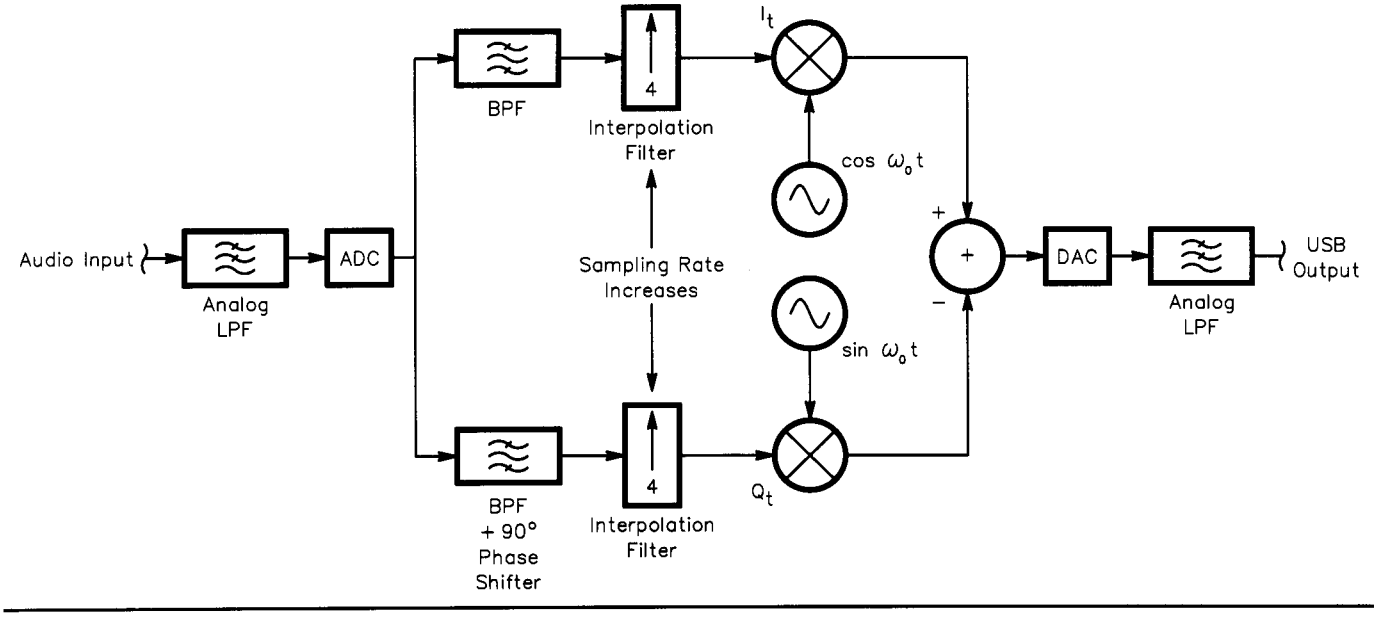

Fig 3—Block diagram of a digital SSB exciter.

through multiplication by a complex carrier:

$$e^{j\omega_0 t} = \cos \omega_0 t + j \sin \omega_0 t \qquad \text{(Eq 16)}$$

and the result is just as it was for Eq 11. Only the real part is computed, and that is at a sampling rate four times the input sample rate. For every sample from the filters, we use four samples of the complex oscillator.

This is beneficial, because for each full output cycle, the cosine oscillator produces values 1, 0, –1, and 0; the sine oscillator produces values 0, 1, 0, and –1. No actual multiplications take place, which saves time and accuracy. The sampling rate of the filter outputs must be artificially increased to make this procedure work—it's called *interpolation*.

The sampling process causes the output spectrum to repeat at harmonics of the sampling frequency. To remove these *aliases*, an analog *anti-alias* filter is required after the digital-to-analog converter (DAC). A digital interpolation filter eases the requirements of this anti-alias filter. It operates at the higher output sampling rate, taking the additional input samples to be zero. The filter shape is designed to attenuate the harmonic spectra in the original signal.

In the example, a USB signal is produced. Had the sum of the real and imaginary parts been taken instead of the difference, an LSB signal would have emerged.

An Independent Sideband (ISB) Exciter

We saw how easy it was to change sidebands in the above example by either subtracting or adding the real and imaginary parts. This makes it easy to create an ISB exciter that transmits separate information on each sideband.

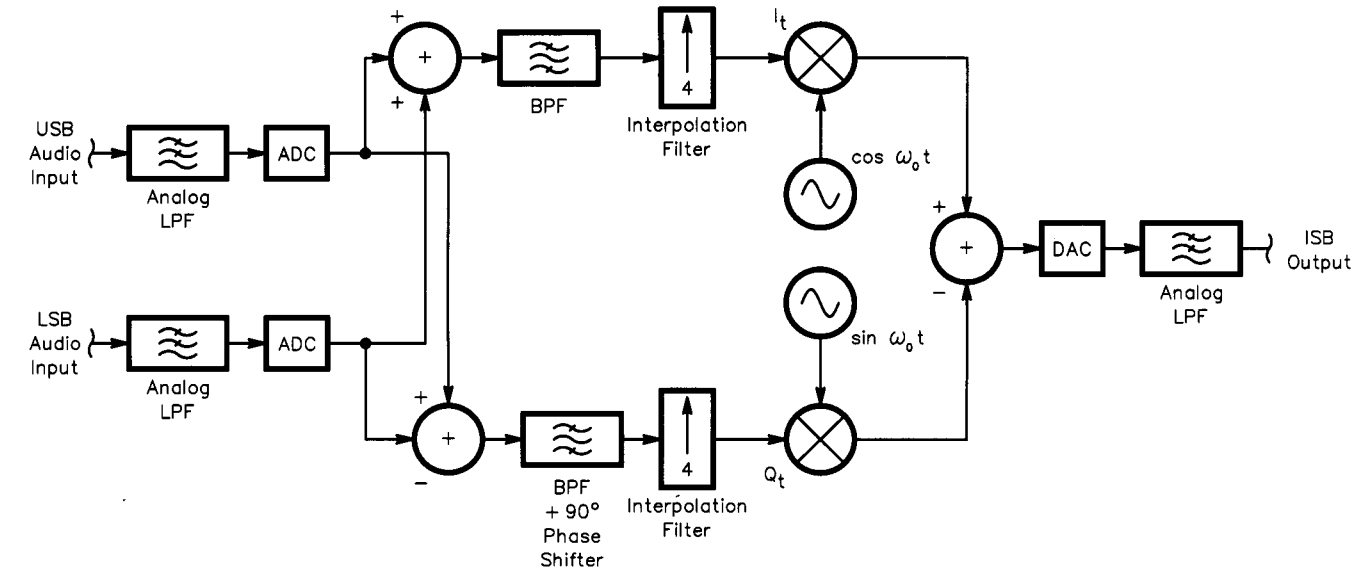

Fig 4—Block diagram of a digital ISB exciter.

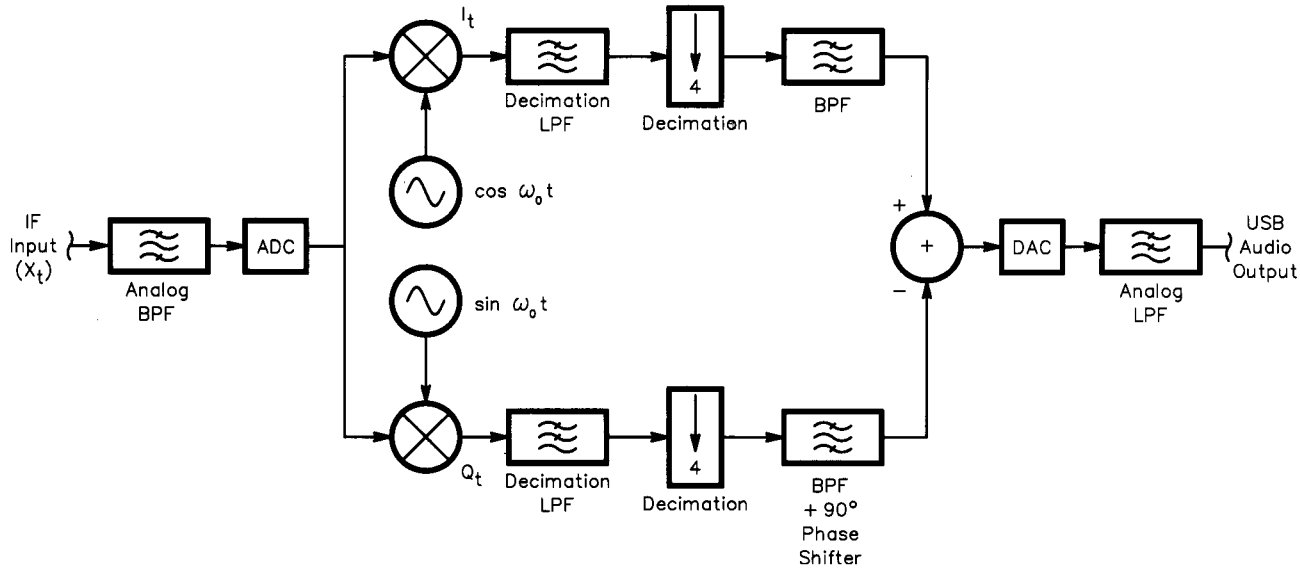

Fig 5—Block diagram of a digital SSB receiver.

The block diagram of such an exciter is shown in Fig 4. Working through the math for each sideband—and taking the input for the other sideband to be zero—shows that the system works as advertised.

While there are other ways to design sideband exciters in DSP, these examples illustrate the flexibility afforded by using the phasing method. The basic structures in the exciter architecture are directly applicable to a sideband receiver, too. Let's consider what happens on the receiving end of an SSB radio link, and see how the math works in reverse.

A Digital SSB Receiver

As in digital exciters, phasing methods prevail in receivers because it is easy to build frequency-independent phase shifters and because amplitude accuracy is preserved throughout. The first job is to convert the analog IF signal to digital form. Because of dynamic range considerations, the frequency at which the sampling is performed is limited to just above the audio range. Current technology dictates that a 16-bit ADC is used at a low-frequency IF.

As expertise in ADCs progresses, the digitization point will move closer to the antenna. For now, we must stick with traditional analog front ends, using very high-frequency first IFs to avoid image and spurious problems, and low-frequency second IFs to satisfy the ADCs. The sampled IF signal must be sharply band pass limited because of the restrictions imposed by current DSP processor performance.

Fig 5 presents a block diagram for a digital SSB receiver. After the IF signal is digitized, we wish to reduce the sampling rate—and the filtered bandwidth—as soon as possible. This is so because we need as much time as possible between input samples for the intense computations we must perform. Reduced sampling rates also ease the design of the digital filters that provide the final selectivity.

A technique known as *harmonic sampling* allows us to reduce the initial sampling rate to twice the input signal bandwidth. Let's say this bandwidth is about 15 kHz, enough for FM and other modulation formats, yet not so wide as to let in lots of QRM.

Therefore, the sampling rate must be at least 30 kHz, and we select something slightly higher to ease the analog IF filtering requirements. The digitized signal is then translated to baseband using the complex mixing algorithms outlined previously. Since the input signal, X_t is real, only two multiplications are necessary:

$$X_t e^{j\omega_0 t} = X_t \cos \omega_0 t + jX_t \sin \omega_0 t \qquad \text{(Eq 17)}$$

Now we have an analytic signal as before, and the value of the oscillator, ω_0, is chosen to beat the carrier frequency to zero. Again, the sampling rate is converted by choosing an IF that is four times the sample rate after translation, so that the oscillator values are only 1, 0, –1 or 0. This time though, we're *reducing* the sampling rate, and this is called *decimation*.

The sampling process again produces spectra at harmonics of the sampling frequency. To avoid mixing these into the passband at the reduced sampling rate, a decimation filter is required. This filter operates at the higher sampling rate, and limits the bandwidth to half the lower sampling rate.

Since the carrier frequency is at zero, the spectrum of our analytic signal contains both negative and positive components. The negative frequencies represent the lower sideband, and the positive frequencies the upper sideband. The I_t and Q_t signals are passed through two band-pass filters, one of which has a built-in 90° phase shift. These filters provide the final receiver selectivity, and are again identical in frequency response. The outputs of these filters are either subtracted or added to demodulate the USB or LSB audio. The digital audio signal is converted back to analog by the DAC, and the output is low-pass filtered to remove the sampling-frequency-caused harmonic spectra.

Obviously, we could both add and subtract the terms to produce an ISB receiver. Very little additional processing overhead would be involved, but we would need another DAC and low-pass filter.

We see that this receiver processes signals in a way that is the reverse of the digital exciter above. The mathematical relationships we described above define the transforms between real audio signals and real SSB signals, and we've seen how they work in both directions. Analytic signals can

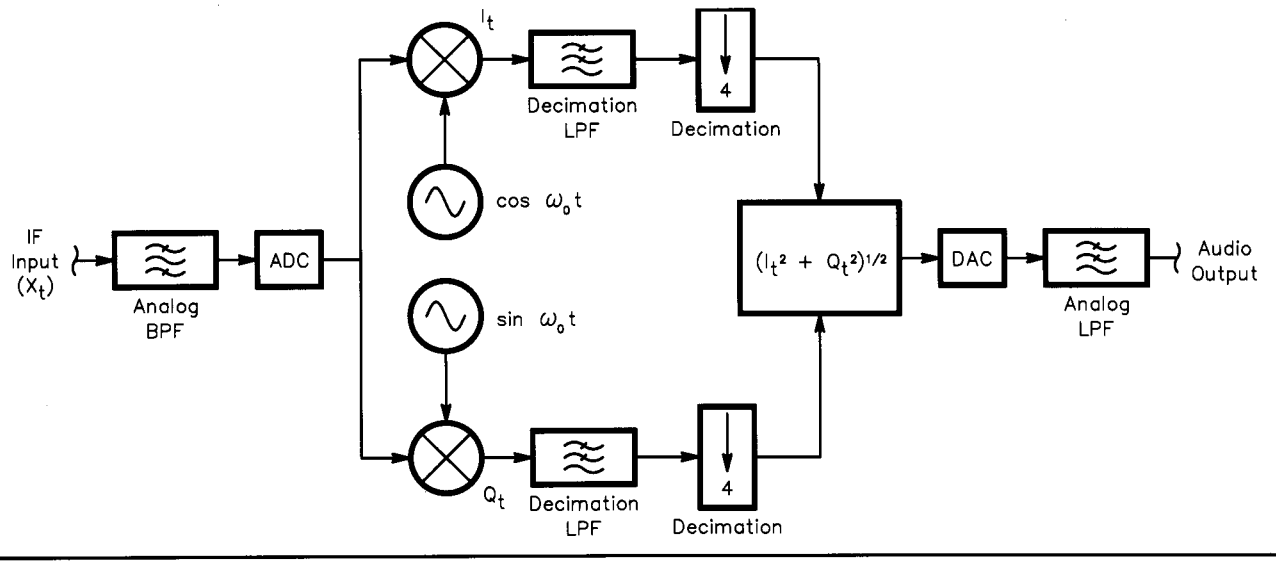

Fig 6—Block diagram of a digital AM receiver.

also be used to demodulate other types of signals, such as AM and FM. Let's look briefly at how these are handled in DSP.

AM Demodulation

In conventional AM, the envelope of the received signal is equal to the amplitude of the baseband audio. So to implement an AM detector, we can compute the magnitude of the analytic signal $I_t + jQ_t$ as in Eq 12. The resulting audio signal is free of the distortion encountered in detectors using rectification methods. Note that in the AM demodulator shown in Fig 6, the second pair of filters isn't necessary; and the decimation filters provide the final selectivity.

Now that we must use Eq12, we're stuck with computing the square root of a number. We could use the relation

$$X^{\frac{1}{2}} = \log^{-1}\left(\frac{\log X}{2}\right) \qquad \text{(Eq 18)}$$

but this involves computing a logarithm, or looking it up from a very large table. Fortunately, Sir Isaac Newton comes to the rescue with his root-finder algorithm!

In the 17th century these calculations were quite a burden, and anything that sped them up was a major blessing. Logarithms had only just been invented, and large tables of them were both expensive and scarce. Newton found a quick, simple iterative method for finding the roots of certain equations. For square roots, it goes like this: Take a crude guess at the square root of the number in question. Divide the number by the crude guess. Add the crude guess to this result, and divide it all by 2. Then, use this result as the new crude guess and repeat the process to obtain the desired accuracy.

$$let\ GUESS_{new} = \left(\frac{\frac{Number}{GUESS_{old}} + GUESS_{old}}{2}\right) \qquad \text{(Eq 19)}$$

$$let\ GUESS_{old} = GUESS_{new}$$

REPEAT

In practice, the accuracy of the result reaches the limit of 16-bit representation in five or six iterations. This method is much quicker and more accurate, for assembly-language implementations, than any alternative.

FM Demodulation

In FM, the instantaneous frequency of the carrier is equal to the amplitude of the baseband signal. We discovered how to compute the phase of our analytic signal above, so we can build a PM demodulator right away using Eq 13. The arctangents can be looked up from a table, or we can compute them using power series.

Once the phase at each sample time is found, the frequency is calculated as the rate of change of the phase:

$$f_t = \frac{d\phi_t}{dt} \qquad \text{(Eq 20)}$$

Differentiating the string of phase samples is accomplished using the technique of *first differencing*. We simply take the difference between the adjacent samples:

$$f_n = \phi_n - \phi_{n-1} \qquad \text{(Eq 21)}$$

This is the FM demodulator output.

Alternatively, FM can be directly calculated by evaluating the vector relationships and using:

$$f_t = \frac{Q_t\left(\frac{dI_t}{dt}\right) - I_t\left(\frac{dQ_t}{dt}\right)}{I^2 + Q^2} \qquad \text{(Eq 22)}$$

then low-pass filtering.

Finally, a digital discriminator can be used to translate the FM to AM, which is then demodulated as described above.

While the mathematics of the various modulation formats can be complicated, they dictate exactly what the computational block diagrams must look like. In a DSP implementation, the equations hold true and the results are precisely as predicted by theory. Next, we'll develop the theories of sampling and data-rate conversion to see what limitations they impose on us, and what advantages apply to digital radio design.

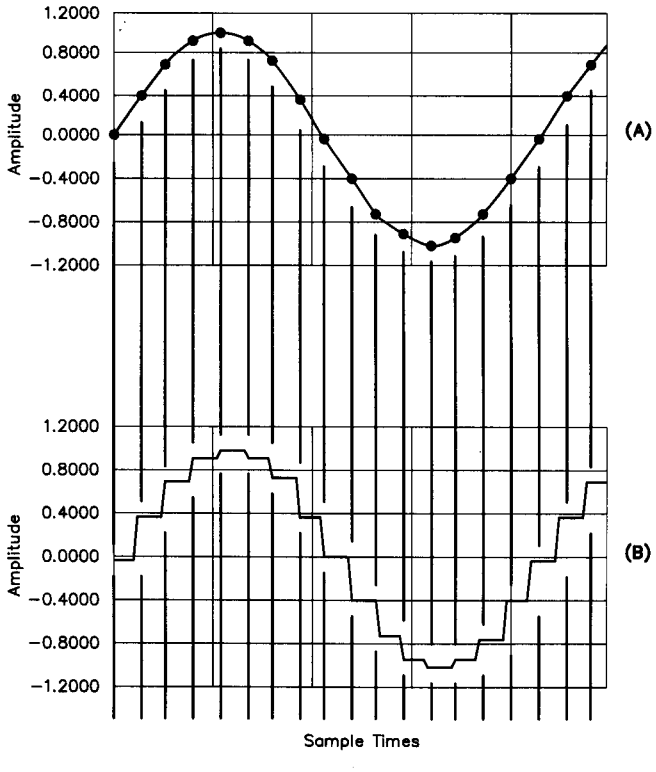

Fig 7a—Sine wave of frequency much less than the sampling frequency.
Fig 7b—Sampled sine wave.

Sampling Theory and Multirate Processing

In the field of radio communication, we deal with analog signals. In order to perform our DSP magic on them, therefore, the first and last steps in the processing chain involve conversion to and from digital form. Once a signal is digitized, we fight to preserve its integrity through the numerical precision of our calculations. At the interfaces to the analog world, the enemies are manifold: noise, distortion and lack of dynamic range and resolution. We'll explore the characteristics of sampled signals, and the reasons for degradation in conversion techniques.

Sampling Theory: Digitizing the Analog World

Sampling is the periodic measurement of the signal voltage and the conversion of this voltage to a number. If we take f_s measurements per second, we call f_s the sampling

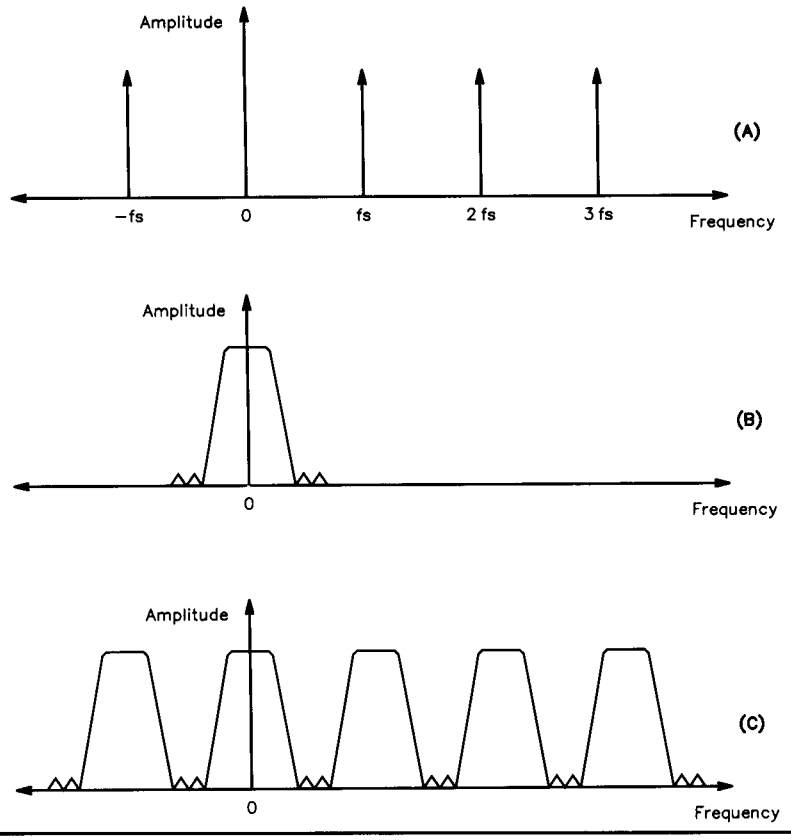

Fig 8a—Spectrum of sampling impulses.
Fig 8b—Spectrum of a band of real input signals.
Fig 8c—Spectrum of the sampled band of signals.

frequency. We can represent the sampled signal as a string of amplitude measurements over time, as shown for the case of a sine wave in Fig 7.

To completely understand the nature of the sampled signal, first consider a function δ_t, consisting of unit impulses spaced at intervals of the sampling time, f_s^{-1}. This is the time-domain representation of the sampling function. The sampled signal can be thought of as the sampling function multiplied by the continuous input signal, x_t:

$$y_t = x_t \delta_t \qquad \text{(Eq 23)}$$

The result is the *convolution* of the two signals—equivalent to their mixing in the analog world. A look at the spectrum of this result shows that it repeats at intervals of the sampling frequency, as shown in Fig 8 for a band of signals.

These repetitions are called *aliases*, and are as real as the fundamental in the sampled signal. They each contain all the information sufficient to describe the original signal. In our example, the sampling frequency is much higher than the signal frequency, so the output still roughly resembles the original sine wave.

Sine Wave, Alias Sine Wave: Harmonic Sampling

Imagine that the input frequency is greater than the sampling frequency. (See Fig 9.) Now, the output no longer matches the input. Note that the sampled signal is still in the shape of a sine wave, though, and that its frequency is lower than that of the input. Ordinarily, this wouldn't be a happy situation.

Nevertheless, a downward frequency translation is useful in the design of an IF-DSP receiver. In addition, lower sam-

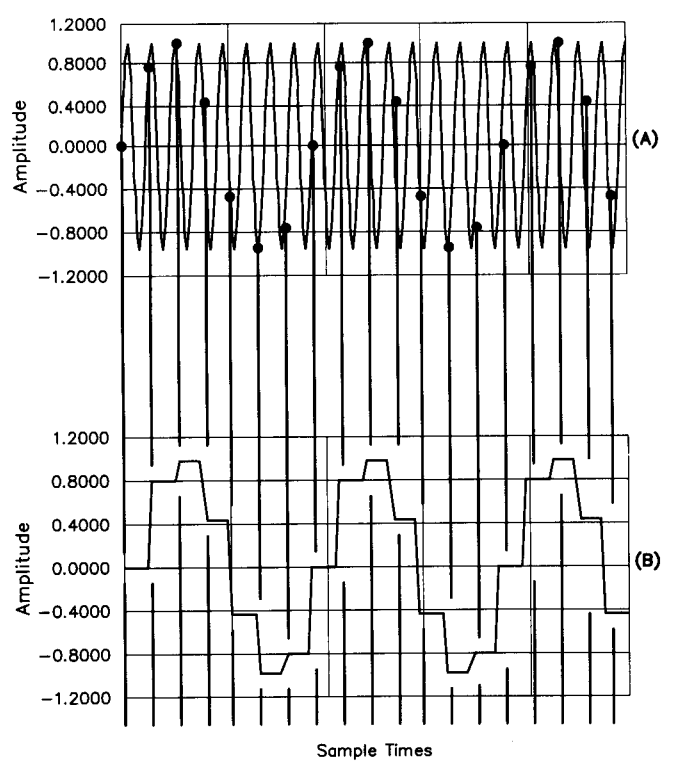

Fig 9a—Sine wave of frequency greater than sampling frequency.
Fig 9b—Harmonically sampled sine wave.

pling frequencies are desirable because they provide more time between samples for signal modification algorithms. Caution is required, though. An input signal near twice the frequency of that in Fig 9 would produce the same output. To use this technique, therefore, we must first band pass limit the input. This trick is known as *harmonic sampling.*

The input signals must fall between the fundamental, or some harmonic, of the sampling frequency and the point halfway to the next harmonic. A frequency translation will take place, but no information about the shape of the input signal will be lost because of this sampling technique. A frequency-domain representation of harmonic sampling is illustrated in Fig 10.

Information *can* be lost, however, because of inaccuracies and noise introduced by the ADC. After digital processing, additional distortion can be introduced by the DAC as we convert back to analog. Let's look at these errors.

Noise and Distortion in Signal Conversion

An ADC is a device that measures an analog signal voltage and outputs a proportional number. A DAC is another device that performs the reverse operation, outputting some analog voltage proportional to the numerical input. Both of these devices inject noise and distortion, at levels that are predicted by sampling theory. Their performance is so critical to any DSP transceiver that it's worth our effort to examine causes of degradation and learn how to combat them.

Aside from stray noise picked up in the physical circuits, the deleterious effects occurring in signal converters can be grouped as follows, and discussed:

- Nonlinearities in quantization step sizes
- Quantization noise
- Aperture jitter
- Noise figure and distortion in analog stages
- Zeroth-order sample-and-hold distortion

Nonlinearities

Nonlinearity means distortion, and—in this case—noise. The quantization steps of any real converter are not perfectly spaced, and conversion results are contaminated by the inaccuracy. In general, two types of nonlinearities can be characterized: differential nonlinearity (DNL), and integral nonlinearity (INL).

DNL is the measure of the output nonuniformity from one input step to the next. It is expressed as the maximum error in the output between adjacent input steps, measured over the entire input range of the device. In an 8-bit converter, for example, the worst errors typically occur when the output changes from 01111111 to 10000000, using base-2 notation.

Since we're talking about the accuracy of the smallest steps the converter can resolve, noisy low-order distortion products caused by this effect limit the dynamic range of the device. Current technology uses correction systems to compensate for temperature variations that would otherwise degrade performance beyond acceptable bounds.

A converter is considered *monotonic* if a steady increase

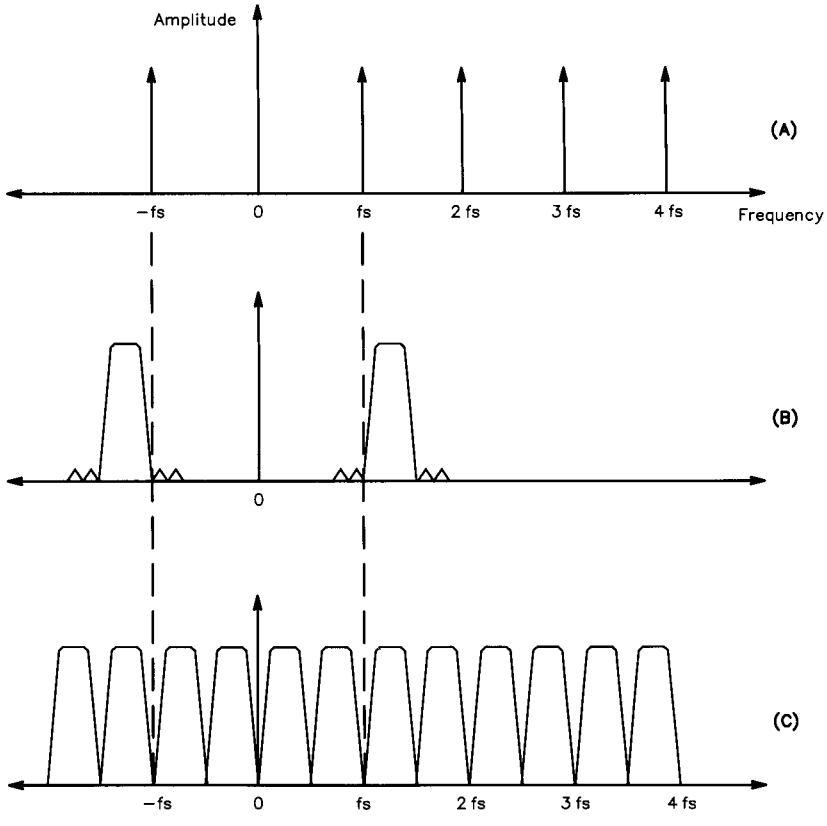

Fig 10a—Spectrum of sampling impulses.
Fig 10b—Spectrum of a band of real signals.
Fig 10c—Spectrum of harmonically sampled band of signals.

in the input signal always results in an increase in the output. Device manufacturers attempt to hold DNL to ±0.5 bits so that monotonicity is maintained.

The second measure of nonlinearity, INL, is a measure of the device's large-signal-handling ability. If we inject a signal of amplitude A and measure the output, then inject a signal of amplitude 100 A, we expect the output to increase in exact proportion. The INL is a measure of the maximum error in the output between any two input levels. Another way of depicting this measurement is to plot the input against the output, and see how straight the line is.

INL produces harmonic distortion and IMD that are obviously undesirable. Typical values are from ±1 to ±2 bits over the entire range.

Quantization Noise

Quantization noise, caused by the inability to resolve signals near the amplitude of the smallest quantization step, is basic to all converters and spread uniformly over the entire input bandwidth of $f_s/2$. The noise power is:

$$P_{qn} = \frac{V_{peak}^2}{3R2^{2b}} \qquad \text{(Eq 24)}$$

in watts, and so the noise density is:

$$ND_{qn} = \frac{P_{qn}}{\left(\frac{f_s}{2}\right)} = \frac{2V_{peak}^2}{3\,f_s R 2^{2b}} \qquad \text{(Eq 25)}$$

in watts/Hertz, where V_{peak} is half the maximum peak-to-peak input signal, R is the ADC input resistance, and b is the number of bits of resolution. Note that the quantization noise density decreases by 6 dB for every bit of resolution in the converter and by 3 dB every time we double the sampling frequency.

Since the maximum sine wave the converter can handle produces a power of:

$$P_{sine} = \frac{\left(\frac{V_{peak}}{\sqrt{2}}\right)^2}{R} = \frac{V_{peak}^2}{2R} \qquad \text{(Eq 26)}$$

the maximum signal-to-noise ratio (SNR) is:

$$\frac{P_{sine}}{P_{qn}} = \frac{3\left(2^{2b}\right)}{2} = (6.02b + 1.76)dB \qquad \text{(Eq 27)}$$

For a 16-bit converter, this is about 98 dB. If we could increase the sampling rate by some factor N, then digitally filter the output back down to the lower rate, we could

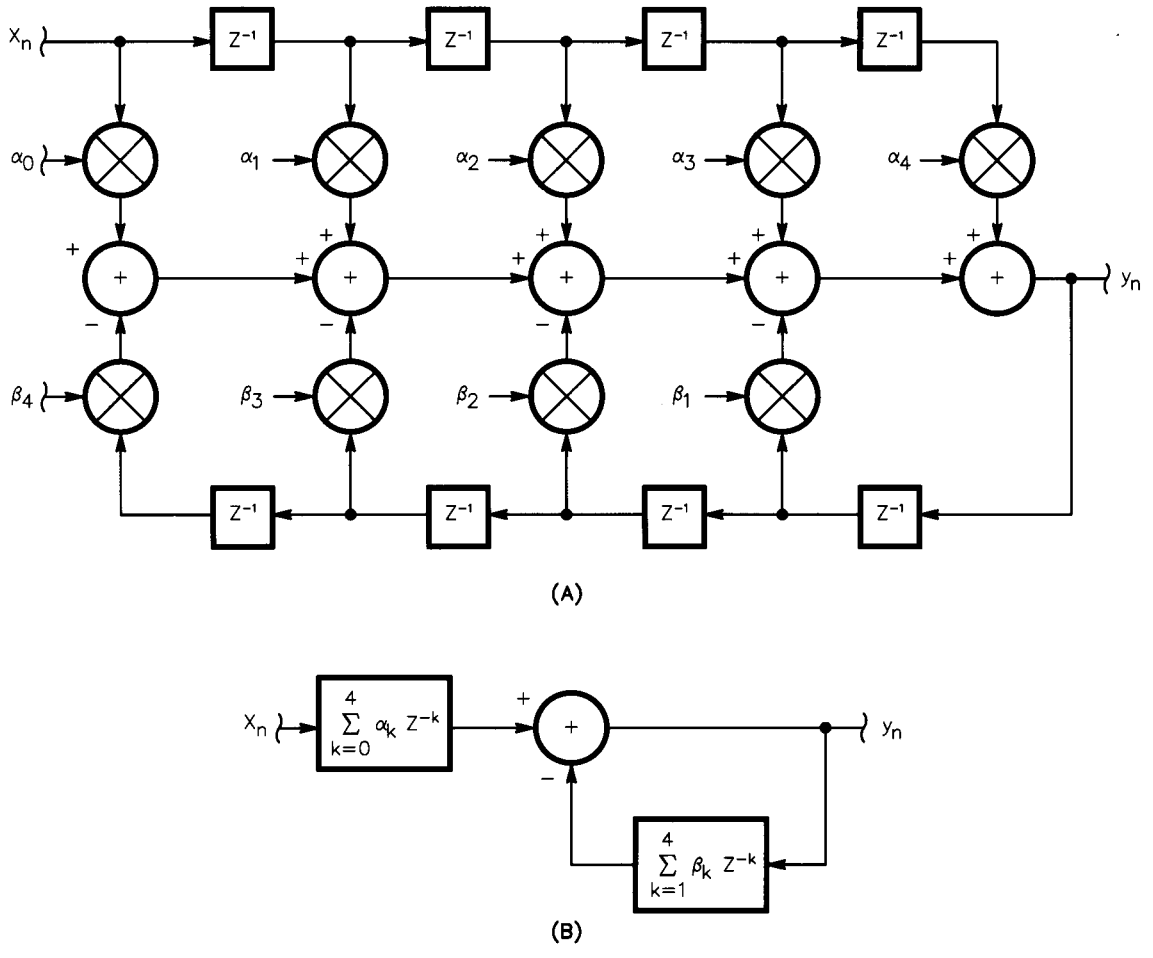

Fig 11a—Block diagram of an IIR filter for L = 5.
Fig 11b—Equivalent block

increase the SNR by almost the factor N. The quantization noise would be spread over a much larger bandwidth, and we'd use a decimation filter to eliminate the higher-frequency noise in bringing the sampling rate back down. This technique is known as *over-sampling*.

So-called "sigma-delta" ADCs use this method to achieve the best possible dynamic range and noise performance. They represent the state of the art in ADC technology.

Aperture Jitter

In addition to the above sources of noise and distortion, noise is introduced by slight variations in the exact times of sampling. Phase noise in the clock source used to derive the sample times, as well as other inaccuracies in the sampling mechanisms, produce undesired phase modulation of the sampled signal. If we assume this phase noise is not correlated with the input signal, a sine wave input generates a white noise component in the output having a power density relative to the input power:

$$\frac{N_{aj}}{P_{sine}} = \frac{8\pi^2 f^2 \sigma_a^2}{f_s} \qquad \text{(Eq 28)}$$

where f is the signal frequency, and σ_a is the RMS time jitter in the sampling rate. Note that the result is expressed in dB/Hz, and is proportional to the squares of both the signal frequency (f) and the time jitter (σ_a).

Noise Figure Issues

Quite often, the smallest signal the converter can discern is so small that noise generated in the analog stages of the converter becomes a problem. Consider a converter with 16 bits of resolution and a peak-to-peak input limit of 5 V. Say its input impedance is 10 kΩ. The noise power contributed by quantization is:

$$P_{qn} = \frac{2.5^2}{(3)(10{,}000)2^{(2)(16)}} \approx 48.5 \times 10^{-15}\,W \approx -103\,dBm \qquad \text{(Eq 29)}$$

It's likely, if the ADC comparators have a wide input bandwidth and a noise figure of more than 6 dB, that their noise contribution exceeds this amount. It's imperative, therefore, to evaluate converters based on all their properties before deciding to incorporate them in a DSP transceiver.

Zeroth-Order Sample-and-Hold Distortion

Typical converters are sample-and-hold devices. That is, they continue to output the last sampled value throughout the sample period. This effect acts as a filter having a frequency response:

$$H_\omega = \frac{\sin\left(\frac{\pi\omega}{\omega_s}\right)}{\left(\frac{\pi\omega}{\omega_s}\right)} \qquad \text{(Eq 30)}$$

This results in a high-frequency rolloff that is quite undesirable in many circumstances. For example, if the output frequency is one quarter of the sample frequency, an attenuation of about 1 dB will occur. This is mainly a problem in the final DAC stage, where signals are converted back to analog. We see that increasing the output sampling frequency reduces the attenuation, and that may call for interpolation of the output.

I've referred to changes in sampling rate called decimation and interpolation and alluded to the advantages in performing these rate changes. Now, let's look at them in more detail. This will lead us naturally into a discussion of digital filtering.

Sampling Rate Reduction: Decimation

Sampling at higher rates can be quite beneficial because it eases the design of the analog filters we must use to avoid the aliasing phenomenon. It also helps reduce noise introduced by quantization and aperture jitter.

At some stage, however, we want to reduce the sampling rate in order to provide as much time as possible in between samples for other calculations. When it's time to digitally filter some signals, making the filter bandwidth a large fraction of the sampling frequency makes it easier to build a sharp-skirted filter. Reduction of the sampling rate is usually referred to as decimation.

Decimation is normally done by integer values—although it doesn't have to be—and is the same as resampling the signal at the lower rate. The resampled signal has a spectrum repeating at intervals of the lower sampling frequency, and so we have to reduce the bandwidth to less than half this value to avoid the aliasing that would destroy information.

The decimation filter, operating at the higher sampling rate f_{in}, eliminates components above $f_{out}/2$ so that aliasing won't occur after the rate reduction. The output signal can be processed at the f_{out} rate, which requires less time. We see, though, that in sampling our signal at the lower rate, we're going to have to either average or discard some of the input samples.

When we filter the signal at the higher rate, then resample at the lower, it turns out it's legitimate to just discard the unneeded samples during decimation. No information about the input signal will be lost since aliasing is avoided. So, why compute these output samples when we're only going to throw them away? We'll calculate only the ones we're going to keep, and this is equivalent to running the decimation filter at the lower sampling rate. This *boxcar* technique is typical of those used by DSP designers to save time and effort.

Increasing the Sampling Rate: Interpolation

While a low sampling rate is pleasant for the reasons outlined above, it may cause difficulties when it's time to convert signals back to analog. The alias products aren't far above the desired signals, and are therefore difficult to filter out. The solution is to increase the sampling rate, and hence the frequencies of the alias products, using the process of interpolation.

We usually do this by an integer factor, although again, we don't have to. While there may be advantages to changing the sampling rate by other rational factors, such as 3/2, we're after an arrangement that solves the aliasing problem with a minimum of processing. In most cases, a doubling or tripling of the rate is sufficient to allow a reasonable anti-aliasing filter to be constructed.

To double the sampling rate, we'll insert additional samples with a value of zero between the existing samples. An interpolation filter is required, using the zero-inserted data as its input. It is a low-pass, operating at the higher sampling rate, which removes the alias components due to the lower sampling rate.

We've seen some properties of sampled signals and learned how the sampling theorem can be used to our advantage in designing a DSP transceiver. We've also seen the potential trade-offs that selection and conversion of sampling rates present. Next, on our path toward acquaintance with the important DSP methods, a thorough understanding of digital filtering is in order.

Digital Filters

The ability to construct high-performance filters is probably the most important reason for using DSP in radio transceivers. An expensive crystal or mechanical filter with a single bandwidth can be replaced by a set of superior digital filters, which offer as many bandwidths as the associated memory can support.

We can build digital filters that have linear phase response, which is very difficult in the analog world. This can be important for certain data modulation modes. Once the filter is designed, each unit is identical to the next—no alignment is necessary! Finally, filter shapes and responses that are impractical in the analog world can be easily implemented in DSP because there are no production variations.

Infinite-Impulse-Response (IIR) Filters

IIR filters are notable for the presence of feedback, which finite impulse response (FIR) filters do not have. For this reason, IIR filters are usually designed by converting traditional analog filter responses, such as Chebyshev and elliptical. IIR filters can have much sharper transition regions than FIR filters (for the same number of multiplications), but they bring with them the nonlinear phase responses of their analog brethren. They are much more susceptible to overflow problems, and are not necessarily unconditionally stable. They are also prone to *limit cycles*, low-level oscillations sustained by inaccuracies in numerical representation.

Designers can attempt to compensate these unwanted traits, but they may find that the resulting computational load isn't worth it. Nevertheless, we shall describe the synthesis and use of IIRs, as they have their places in modern radio development.

The transfer function of an analog Chebyshev low-pass filter can be written as the ratio of a constant to an *n*th-order polynomial:

$$H_s = \frac{K}{s^n + a_1 s^{n-1} + a_1 s^{n-2} + \ldots + a_n} \qquad \text{(Eq 31)}$$

Tables in the literature, such as Zverev, list the values of the coefficients, a, which are related to the cutoff frequency and used to derive actual component values for the filter. The low-pass design can be transformed to band-pass or bandstop response. Two popular methods exist for deriving the digital transfer function from the analog transfer function. These are known as the *impulse-invariant* method and the *bilinear transform* method.

The impulse-invariant method assures that the digital filter will have an impulse response equivalent to its analog counterpart, and therefore the same phase response. Problems arise, though, if the bands of interest are near half the sampling frequency. The digital filter's response can develop serious errors in this case. Because of this, the impulse-invariant method isn't as good as the bilinear transform method.

The bilinear transform method makes a convenient substitution for *s* in Eq 31 above, and the filter output comes out looking like:

$$y_n = \sum_{k=0}^{L-1} \alpha_k x_{n-k} - \sum_{k=1}^{L-1} \beta_k y_{n-k} \qquad \text{(Eq 32)}$$

This filter has L zeros and L – 1 poles.

The block diagram of such a filter for L = 5 is shown in Fig 11. Each box marked "Z^{-1}" is a one-sample-time delay. Feedback is evident in the diagram: The paths involving coefficients labeled ß loop back and are added to the signal path.

This *direct form* equation can be factored into 2-pole sections, and implemented in cascaded form. For each section:

$$y'_n = \left(\sum_{k=0}^{2} \alpha'_k x'_{n-k} - \sum_{k=1}^{2} \beta'_k y'_{n-k} \right) \qquad \text{(Eq 33)}$$

The output of each section serves as the input to the next. This configuration requires a few more multiplications than the direct form, but is less prone to instability and limit-cycle problems when proper pole-zero pairing is used. Additional information about IIR filters can be found in the literature.

Finite-Impulse-Response (FIR) Filters

This digital filter is by far the most popular for SSB use because of its linear phase response. The transfer function of an FIR filter has only zeros, so to implement one, we eliminate the poles and compute only the left half of Eq 32. The output then takes the form:

$$y_n = \sum_{k=0}^{L-1} h_k x_{n-k} \qquad \text{(Eq 34)}$$

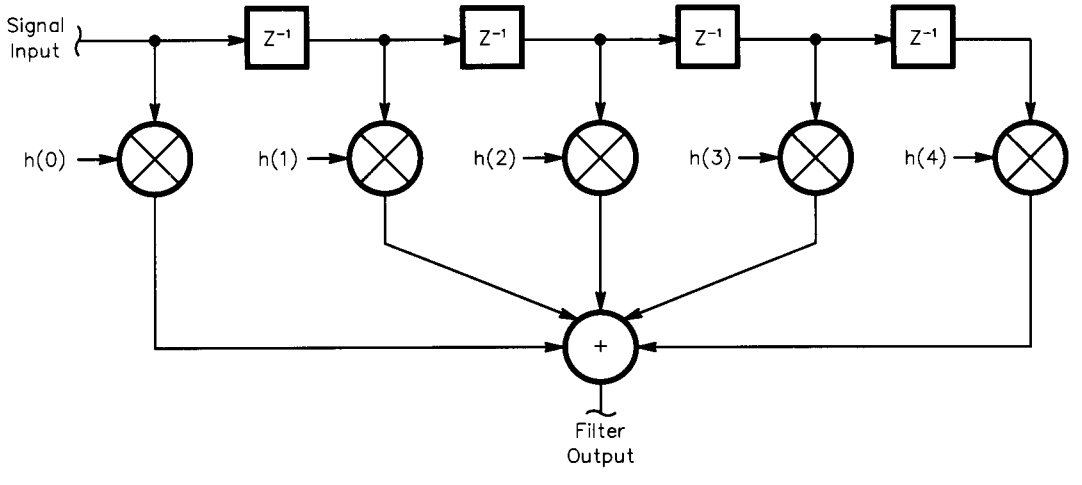

Fig 12—Block diagram of an FIR filter for L = 5.

where h is the set of L coefficients. The coefficients represent the impulse response of the filter, and the filter is said to have *length* L.

The block diagram of such a filter for L = 5 is shown as Fig 12. The set of input registers holding delayed samples of x is just a tapped digital delay line, and for this reason, this filter can be said to have 5 *taps*. Since the output depends only on past input values, the filter can be said to be a *causal process*.

Normally, the impulse response has a symmetry about center, and it turns out this is enough to ensure that no differential or group delay distortion is produced. Since there is no feedback, the filter is unconditionally stable. Almost any frequency response can be generated if enough taps can be used. Many excellent CAD programs are available to design digital filters, relieving us of the burden of generating coefficient sets.

Numerical Accuracy Effects in FIR Filters

In general, adding more taps sharpens the transition regions in the frequency response. In addition, accurate frequency and phase responses require a large number of taps. Because it is truncated at both ends, the finite coefficient set is only an approximation of that required for the exact response we want. Some error in the result must be tolerated.

After Rabiner and Gold, the number of taps required can be estimated using:

$$L = 1 - \frac{10\log(\delta_1\delta_2) - 15}{14\left(\frac{f_T}{f_s}\right)} \qquad \text{(Eq 35)}$$

where δ_1 is the allowable passband ripple, δ_2 is the stopband attenuation, f_T is the transition bandwidth, and f_s the sampling frequency. This assumes, of course, enough bits of resolution are used to achieve the required accuracy. In actual practice, filters of over 100 taps are used to realize shape factors of less than 1.14.

When computers are used to design FIR filters, coefficients can be represented to the full accuracy of the program—usually in floating-point format with 12 or more decimal significant figures in the mantissa. Real implementations ordinarily don't achieve this accuracy, as we're typically limited to 16 bits in an embedded DSP design. The truncation of coefficients and data degrades the response and, of course, sets the dynamic range.

We know that the product of two 16-bit numbers is a 32-bit number, so we need at least that many bits in the final accumulator to avoid losing accuracy. Another point of interest: While the delay through the filter isn't dependent on frequency, the absolute value of that delay can be quite large. In fact, it is equal to:

$$T_{FIR} = \frac{L \bullet t_s}{2} \qquad \text{(Eq 36)}$$

When we get around to using these filters in our design, we'll want to consider the effects of these delays.

Fixed-Point Mathematics and Scaling Problems

The typical DSP microprocessor is a 16-bit machine using fixed-point math. This means numbers are represented internally as signed fractions between –1 and 1 in *2s complement* format. The most significant bit (MSB) represents the sign of the number, and the 15 least significant bits (LSBs) the magnitude—or the complement of the magnitude for a negative number. In hexadecimal format, $7fff is the largest positive number and represents $1–2^{-15}$, while $8000 is the most-negative number, and represents (–1). This is a convenient format, since the product of two fractions is always another fraction! When we start adding the fractions together, however, (as in our FIR filter above) we may generate a number whose magnitude is greater than unity—a result known as *overflow*.

Most DSPs using 16-bit data and coefficients have final accumulators with at least 32 bits. A trade-off exists between the possibility of overflow—which is catastrophic—and the loss of accuracy in the LSBs, via truncation during operations. Especially in FIR filters with sharp transition regions, and under certain input conditions, the output can exceed ±1. The worst-case output can grow as large as the sum of the absolute value of all the coefficients:

$$y_{max} = \sum_{k=0}^{L-1} |h_k| \qquad \text{(Eq 37)}$$

We must scale either the data or the coefficients by the reciprocal of this factor to ensure against overflow.

At the small-signal end of things, the bit-resolution of the system determines the dynamic range because of the presence of quantization noise, just as in the case of ADCs or DACs. It is computed almost the same way. Its normalized amplitude is:

$$V_{qn} = \frac{2^{-(b+1)}}{\sqrt{3}} \qquad \text{(Eq 38)}$$

where b is the number of bits used, internally, to represent numbers.

Truncation of numerical results is another form of quantization noise, also computed in the same way. If each of the L multiplications is truncated, then the noise amplitude from this source is:

$$V_{trunc} = \frac{2^{-(b+1)}L}{\sqrt{3}} \qquad \text{(Eq 39)}$$

It's interesting to note that while truncation of the coefficients affects the response characteristics of the filter, it doesn't contribute to the noise in the output.

Hilbert Transforms

We know that to build a DSP radio, it's convenient to use phasing methods as discussed previously. We need a way to shift the phase of signals by 90°. For a single frequency, this is easy: We just insert a delay of one-quarter cycle. Over a range of frequencies, though, an FIR structure is required to obtain a frequency-independent phase shifter. We'll call this a Hilbert transformer.

Coefficients for a Hilbert transformer can be generated

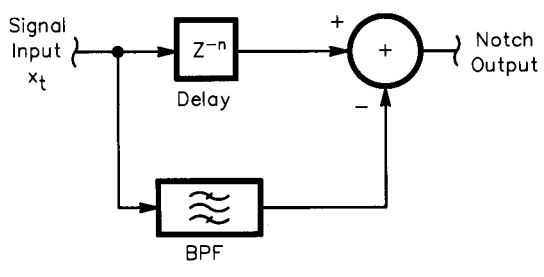

Fig 13—Block diagram of digital notch filter.

by CAD programs, and optimized for the bandwidth of interest. As noted above, the coefficient set will be symmetrical about center. The phase response is, of course, a straight line. A nice property of Hilbert transform impulse responses is that if the total number of taps is odd, the even-numbered coefficients are zero. This cuts our number of calculations in half.

Also, to minimize the computational load, let's see if we can somehow avoid having a separate Hilbert transformer in our phasing-method implementation. We'll attempt to build a pair of band-pass filters, with the frequency response we need and whose phase responses are 90° different from each other. These can then be used directly in our receiver/exciter to create and operate on analytic signals.

Analytic Filter-Pair Synthesis

The frequency translation theorems we explored above can be used to advantage in creating our pair of filters. If we start with a low-pass filter having impulse response, h_t, and frequency response, H_ω, multiplying the impulse response by a complex sinusoid

$$e^{j\omega_0 t} = \cos\omega_0 t + j\sin\omega_0 t \qquad \text{(Eq 40)}$$

results in two sets of coefficients, one for the real part, and one for the imaginary part:

$$h_{I_t} = h_t \cos\omega_0 t$$
$$h_{Q_t} = h_t \sin\omega_0 t \qquad \text{(Eq 41)}$$

The frequency response of either one of these filters is given by:

$$H_\omega = \frac{H_{(\omega-\omega_0)} + H_{(\omega+\omega_0)}}{2} \qquad \text{(Eq 42)}$$

which is a band-pass filter centered at ω_0. The first filter in Eq 41 has a phase response 90° different from the second. The frequency translation theorem works on the responses of filters just as well as it does on real signals!

To perform this transformation on the L coefficients of the initial low-pass filter, we calculate new coefficients:

For $0 \le k \le L-1$,

$$h_{I_k} = h_k \cos\omega_0\left(k - \frac{L}{2} + \frac{1}{2}\right)t_s \text{ OR}$$
$$h_{Q_k} = h_k \sin\omega_0\left(k - \frac{L}{2} + \frac{1}{2}\right)t_s \qquad \text{(Eq 43)}$$

We can implement an IF shift in our receiver design simply by altering the value of ω_0, and computing the new coefficients. We can alter the transmitter's frequency response by convolving the impulse response of our analytic filter pair with that of a filter having the desired characteristic. It's evident that FIR filters yield flexibility beyond that of any analog technique.

Digital Notch Filters

The other type of filter of interest to us is the notch, designed to remove a single frequency. Note that such a filter can be constructed by subtracting the output of a narrow band-pass filter from the broadband input, as shown in Fig 13. We include a delay of

$$Z^{-n} = \frac{L \bullet t_s}{2} \qquad \text{(Eq 44)}$$

in the broadband input to compensate for the delay through the band-pass filter, whose length is L.

An unusual type of notch filter has been described by

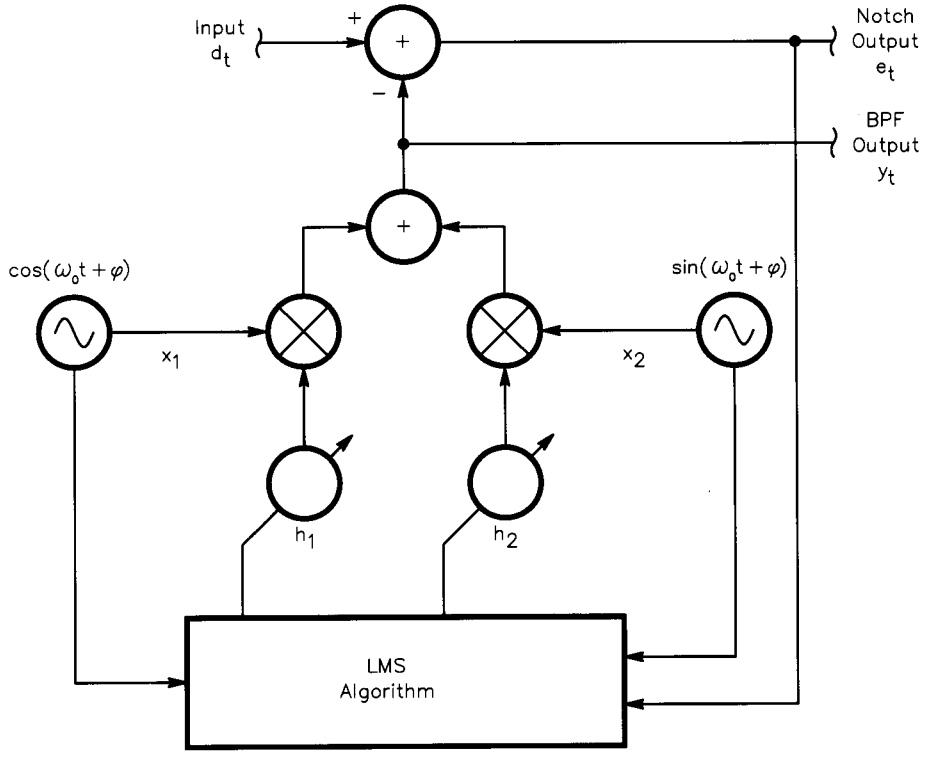

Fig 14—Block diagram of adaptive, manually tuned notch filter.

Widrow and Stearns, in which the number of taps in the band-pass filter is minimized. In fact, they were able to show that only two taps were needed for each frequency to be notched! DSP designers love this, since the amount of computation is almost nil. In describing this notch filter, we'll introduce the concept of the *adaptive interference canceler*, and we'll touch on some of the theory involved in adaptive signal processing.

The Adaptive, Manually Tuned Notch Filter

The situation is this: We want to copy a broadband signal, such as an SSB voice signal, and suddenly, a dreadful carrier appears in the passband! Our notch filter will remove it, and we'll have complete control over the notch width, along with a depth limited only by the bit resolution of our system.

Dr. Widrow discovered that one can build a filtering system to minimize repetitive signal energy by altering the filter coefficients "on the fly" using a certain algorithm. Known as the *least-mean-squares* (LMS) method, it describes a way to adjust FIR filter coefficients over time to remove an undesired tone in the input! A reference signal is used, which is of the exact frequency of the interfering tone. The algorithm then forms a band-pass filter that is subtracted from the broadband input to create the notch.

The block diagram of this system is shown in Fig 14. The broadband input is called d_t, and the reference input is a pure cosine wave:

$$x_t = A\cos(\omega_0 t + \phi) \qquad \text{(Eq 45)}$$

The cosine wave is sampled and fed to the input of one multiplier. It is also phase-shifted by 90° to produce a sine wave, which is fed to the second multiplier. The multiplier outputs are then added, as in a regular FIR filter, to form the band pass output. This output y_t is then subtracted from the broadband input signal to produce the notch output, e_t. Note that the band pass output is also available at no additional overhead.

While the initial values of the coefficients h_1 and h_2 are unimportant, the procedure for updating them is defined by the LMS algorithm as:

$$h_{1(t+1)} = h_{1t} + 2\mu e_t x_{1t}$$
$$h_{2(t+1)} = h_{2t} + 2\mu e_t x_{2t} \qquad \text{(Eq 46)}$$

where $0 < \mu < 1$, and the sampled reference inputs are:

$$x_{1t} = A\cos(\omega_0 t + \phi)$$
$$x_{2t} = A\sin(\omega_0 t + \phi) \qquad \text{(Eq 47)}$$

In the final analysis, it can be shown that as the reference inputs are sinusoidal, the system is linear and time-invariant for the output e_t. Several interesting points follow about the characteristics of this notch.

Adaptive Notch Filter Properties

First, the 3 dB bandwidth of the notch can be shown to be:

$$BW = \frac{2\mu A^2}{t_s} \qquad \text{(Eq 48)}$$

radians/second. The Q of the filter is simply the center frequency divided by the bandwidth:

$$Q = \frac{\omega_0}{2\mu A^2}(t_s) \qquad \text{(Eq 49)}$$

Therefore, we have control over the bandwidth by varying the factor, μ, and the amplitude of the reference signal, A, in the equation. The depth of the null is, in general, superior to that of a fixed filter because the algorithm will maintain the correct phase relationship for ideal cancellation, even if the reference frequency is changing slowly.

Each additional tone to be notched creates the need for two more taps in the adaptive filter. Any noise in the input causes us to add more taps to achieve sufficient accuracy. More detail of adaptive processing will be provided in future articles, and still more details may be found in the references listed.

Summary

DSP microprocessors are optimized for the multiply-and-accumulate (MAC) operation that forms the backbone of digital filtering and other DSP algorithms. We've discovered that FIR filter structures are definitely the way to go when designing DSP subsystems, and that quite a few nuances of theory make life easier.

In Part 2 of our review of DSP techniques, we'll look at an actual transceiver scheme, and examine how the strategies we've learned result in an efficient, high-performance design.

REFERENCES

1. Oppenheim, A. V. and Schafer, R. W., *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1975.
2. Sabin, W .E. and Schoenike, E. O., editors, *Single Sideband Systems and Circuits*, McGraw-Hill, New York, NY, 1987.
3. Alkin, O., *PC-DSP*, Prentice-Hall, Englewood Cliffs, NJ, 1990.
4. Widrow, B. and Stearns, S. D., *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1985.
5. Frerking, M. E., *Digital Signal Processing in Communications Systems*, Van Nostrand-Reinhold, New York, NY, 1993.
6. Rabiner, L. R. and Gold, B., *Theory and Application of Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1975.
7. Zverev, A. I., *Handbook of Filter Synthesis*, John Wiley & Sons, New York, NY, 1967.
8. Panter, P. F., *Modulation, Noise, and Spectral Analysis*, McGraw-Hill, New York, NY, 1965.

Doug Smith, KF6DX/7, is an electrical engineer with 17 years experience designing HF transceivers, control systems, DSP hardware and software. He joined the amateur ranks in 1982, and placed on the air one of the first HF scanning bulletin-board systems. At Kachina Communications, Inc. in central Arizona, he is currently exploring the state of the art in digital transceiver design. ☐☐