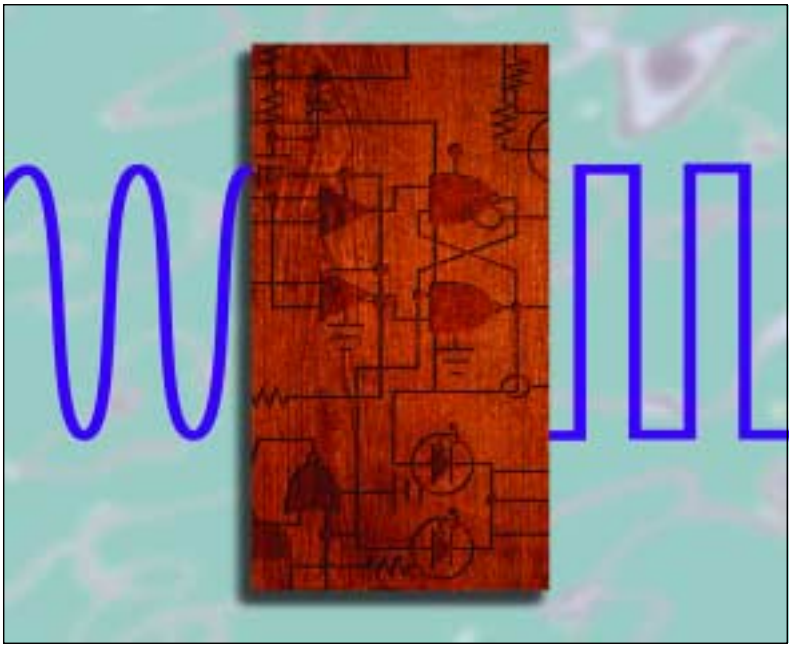
The care and feeding of digital, pulse-shaping filters

In the not-too-distant future, data will be the predominant baggage carried by wireless and other transmission systems. Pulse-shaping filters will play a critical part in maintaining signal integrity.

By Ken Gentile

As digital technology ramps up for this century, an ever-increasing number of RF applications will involve the transmission of digital data from one point to another. The general scheme is to convert the data into a suitable baseband signal that is then modulated onto an RF carrier. Some pervasive examples include cable modems, mobile phones and high-definition television (HDTV). In each of these cases, analog information is converted into digital form as an ordered set of logical 1's and 0's (bits). The task at hand is to transmit these bits between source and destination, whether by phone line, coaxial cable, optical fiber or free space.



A brief history lesson

In its simplest form, the transmission of binary information (i.e., bits) between two points is a simple task. Consider Morse code. The "dots" and "dashes" of Morse code represent a binary form of transmission that has been in use since the mid-19th century. It found application in the telegraph and ship-to-ship light signaling. In today's environment, however, digital transmission has become a much more challenging proposition.

The main reason is that the number of bits that must be sent in a given time interval (data rate) is continually increasing. Unfortunately, the data rate is constrained by the bandwidth available for a given application. Furthermore, the presence of noise in a communications system also puts a constraint on the maximum error-free data rate. The relationship between data rate, bandwidth and noise was quantified by Shannon (1948) and marked a breakthrough in communications theory.

Digital data: Peeling back the layers

In modern data transmission systems, bits or groups of bits (symbols) are typically transmitted in the form of individual pulses of energy. A rectangular pulse is probably the most fundamental. It is easy to implement in a real-world system because it can be directly compared to opening and closing a switch, which is synonymous with the concept of binary information. For example, a "1" bit might be used to turn on an energy source for the duration of one pulse interval (τ seconds), which would produce an output level, "A" (see Figure 1a). Alternately, a "0" bit would turn off the energy source, producing an output level of zero during one pulse interval.

The Fourier transform of the pulse yields its spectral characteristics, which is shown in Figure 1b. Note that a pulse of width τ has the bulk of its energy contained in the main lobe, which spans a one-sided bandwidth of $1/\tau$ Hz. This would imply that the frequency span of a data transmission channel must be at least $2/\tau$ Hz wide. More will be said about this later.

Figure 1 shows that a pulse of a given width, τ, spans a bandwidth that is inversely related to τ. If a data rate of $1/\tau$ bits per second is chosen, then each bit occupies one pulse width (namely, τ seconds). Obviously, if we wish to send bits at a faster rate, then the value of τ must be made smaller. Unfortunately, this forces the bandwidth to increase proportionally (see Figure. 1b).

Such data-rate/bandwidth relationships pose a problem for band-limited systems. This is mainly because most transmission systems have band limitations imposed by either the natural bandwidth of the transmission medium (copper wire, coaxial cable, optical fiber) or by governmental or regulatory conditions. Thus, the challenge in data

transmission systems is to obtain the highest possible data rate in the bandwidth allotted with the least number of errors (preferably none).

Pulse shaping: The details

Before delving into the details of pulse shaping, it is important to understand that pulses are sent by the transmitter and ultimately detected by the receiver in any data transmission system. At the receiver, the goal is to sample the received signal at an optimal point in the pulse interval to maximize the probability of an accurate binary

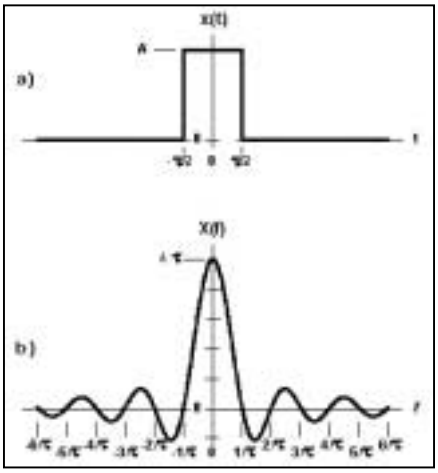


Figure 1. A single rectangular pulse and its Fourier transform.

decision. This implies that the fundamental shapes of the pulses be such that they do not interfere with one another at the optimal sampling point.

There are two criteria that ensure noninterference. Criterion one is that the pulse shape exhibits a zero crossing at the sampling point of all pulse intervals except its own. Otherwise, the residual effect of other pulses will introduce errors into the decision making process. Criterion two is that the shape of the pulses be such that the amplitude decays rapidly outside of the pulse interval.

This is important because any real system will contain timing jitter, which means that the actual sampling point of the receiver will not always be optimal for each and every pulse. So, even if the pulse shape provides a zero crossing at the optimal sampling point of other pulse intervals, timing jitter in the receiver could cause the sampling instant to move, thereby missing the zero crossing point. This, too, introduces error into the decision-making process. Thus, the quicker a pulse decays outside of its pulse interval, the less likely it is to allow timing jitter to introduce errors when sampling adjacent pulses. In addition to the noninterference criteria, there is the ever-present need to limit the pulse bandwidth, as explained earlier.

The rectangular pulse

The rectangular pulse, by definition, meets criterion number one because it is zero at all points outside of the present pulse interval. It clearly cannot cause interference during the sampling time of other pulses. The trouble with the rectangular pulse, however, is that it has significant energy over a fairly large bandwidth as indicated by its Fourier transform (see Figure 1b). In fact, because the spectrum of the pulse is given by the familiar $\sin(\pi x)/\pi x$ (sinc) response, its bandwidth actually extends to infinity. The unbounded frequency response of the rectangular pulse renders it unsuitable for modern transmission systems. This is where pulse shaping filters come into play.

If the rectangular pulse is not the best choice for band-limited data transmission, then what pulse shape will limit bandwidth, decay quickly, and provide zero crossings at the pulse sampling times? The raised cosine pulse, which is used in a wide variety of modern data transmission systems. The magnitude spectrum, $P(\omega)$, of the raised cosine pulse is given by:

$$
\begin{aligned}
&(1)\ P(\omega) = \tau \\
&\text{for } 0 \le \omega \le \frac{\pi(1-\alpha)}{\tau} \\
&(2)\ P(\omega) = \frac{\tau}{2}\left(1 - \sin\left(\left(\frac{\tau}{2\alpha}\right)\left(\omega - \frac{\pi}{\tau}\right)\right)\right) \\
&\text{for } \frac{\pi(1-\alpha)}{\tau} \le \omega \le \frac{\pi(1+\alpha)}{\tau} \\
&(3)\ P(\omega) = 0 \\
&\text{for } \omega \ge \frac{\pi(1+\alpha)}{\tau}
\end{aligned}
\qquad (1)
$$

The spectral shape of the raised cosine pulse is shown in Figure 2a. The inverse Fourier transform of $P(\omega)$ yields the time-domain response, $p(t)$, of the raised cosine pulse (see Figure 2b). This is also referred to as the impulse response and is given by:

$$
p(t) = \frac{\left(\operatorname{sinc}\frac{t}{\tau}\right)\left(\cos\frac{\alpha\pi t}{\tau}\right)}{1 - \left(\frac{2\alpha t}{\tau}\right)^2} \qquad (2)
$$

Care must be taken when (2) is used for calculation because the denominator can go to zero if $\alpha t/\tau = \pm ½$. Therefore, any program used to compute $p(t)$ must test for the occurrence of $\alpha t/\tau = \pm ½$. Because it can be shown that the limit of $p(t)$ as $\alpha t/\tau$ approaches $\pm ½$ is given by $(\pi/4)\ \operatorname{sinc}(t/\tau)$, this is the formula to use when the special case of $\alpha t/\tau = \pm ½$ is encountered.

The raised cosine pulse

Unlike the rectangular pulse, the raised cosine pulse takes on the shape of a sinc pulse, as indicated by the leftmost term of $p(t)$. Unfortunately, the name "raised cosine" is misleading. It actually refers to the pulse's frequency spectrum, $P(\omega)$, not to its time domain shape, $p(t)$. The precise shape of the raised cosine spectrum is determined by the parameter, α, where $0 \le \alpha \le 1$.

Specifically, α governs the bandwidth occupied by the pulse and the rate at which the tails of the pulse decay. A value of $\alpha = 0$ offers the narrowest bandwidth, but the slowest rate of decay in the time domain. When $\alpha = 1$,

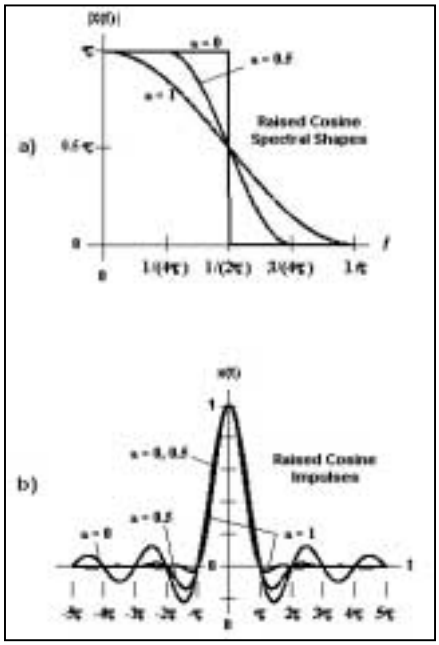


Figure 2. Spectral shape and inverse Fourier transform of the raised cosine pulse.

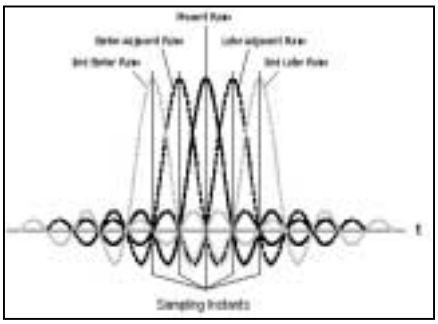


Figure 3. Interaction of raised cosine pulses when the time between pulses coincides with the data rate.

the bandwidth is $1/\tau$, but the time domain tails decay rapidly. It is interesting to note that the $\alpha = 1$ case offers a double-sided bandwidth of $2/\tau$. This exactly matches the bandwidth of the main lobe of a rectangular pulse, but with the added benefit of rapidly decaying time-domain tails. Conversely, inverse when $\alpha = 0$, the bandwidth is reduced to $1/\tau$, implying a factor-of-two increase in data rate for the same bandwidth occupied by a rectangular pulse. However, this comes at the cost of a much slower rate of decay in the tails of the pulse. Thus, the parameter α gives the system designer a trade-off between increased data rate and time-domain tail suppression. The latter is of prime importance for systems with relatively high timing jitter at the receiver.

Figure 3 shows how a train of raised cosine pulses interact when the time between pulses coincides with the data rate. Note how the zero crossings are coincident with the pulse centers (the sampling point) as desired.

It should be pointed out that the raised cosine pulse is not a cure-all. Its application is restricted to energy pulses that are real and even (i.e., symmetric about $t = 0$). A different form of pulse shaping is required for pulses that are not real and even. However, regardless of the necessary pulse shape, once it is expressible in either the time or frequency domain, the process of designing a pulse-shaping filter remains the same. In this article, only the raised cosine pulse shape will be considered. A variant of the raised cosine pulse is often used in modern systems – the root-raised cosine response. The frequency response is expressed simply as the square root of $P(\omega)$ (and square root of $p(t)$ in the time domain). This shape is used when it is desirable to share the pulse-shaping load between the transmitter and receiver.

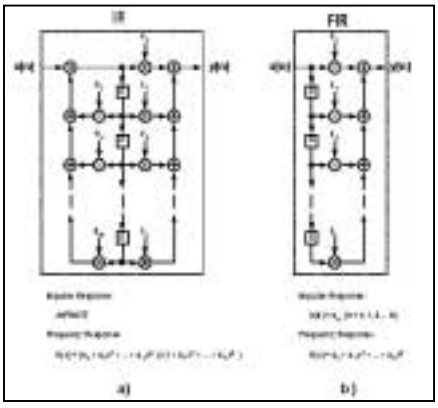


Figure 4. The functional form of FIR and IIR filters.

It's better in digital

Before the advent of digital filter design, pulse-shaping filters had to be implemented as analog filter designs. Digital filters, however, offer several advantages of analog designs. They can be integrated directly on silicon, which makes them attractive for system-on-a-chip (SoC) designs. Furthermore, the problem of component drift due to temperature and aging is eliminated. Also, their spectral characteristics are consistent and reproducible and do not suffer from component tolerance issues.

With the plethora of digital filter design tools available on the market, the designer can design a variety of digital filters with little effort.

Choices, choices, choices

Given that the pulse shape has been defined mathematically (such as the raised cosine pulse), the next task is to decide which basic category of digital filter to use: finite impulse response (FIR) or infinite impulse response (IIR).

The functional form of FIR and IIR filters is shown in Figure 4. The fundamental difference between them is the fact that the IIR contains feedback. This should be obvious from the fact that the b_i coefficient's feedback scaled and delayed samples of the output $y(n)$. Hence, the history of the output affects the future of the output. This is not true for the FIR, where $y(n)$ only depends on the history of the input samples, $x(n)$. The implication is that the response of an IIR filter to an impulse (a single non-zero sample followed by zero samples) is infinite. That is, the IIR will continue to produce non-zero output samples long after the application of an impulse. This is an undesirable consequence for data

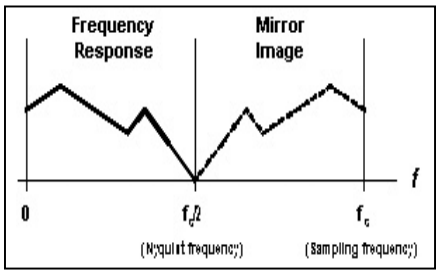


Figure 5. An arbitrary digital filter frequency response.

pulse transmission (recall the noninterference criteria).

The FIR does not suffer from this problem because its architecture does not contain any feedback elements. A single, non-zero impulse at the input will only yield output samples while the impulse propagates down the delay stages. Generally, pulse shaping filters employ FIR designs.

The basic building blocks of a digital filter are adders ($\oplus$), multipliers ($\otimes$), and unit-delays (D); all of which can be readily implemented in digital form. Adders and multipliers are composed of combinational logic while the unit delays are composed of latches (which require a clock signal). The basic filtering operation consists of a sequence of multiply/add/delay operations that occur each time the delay stages are clocked. This is effectively a convolution operation, which may be expressed as:

$$y(n) = x(n) * h(n)$$

In this expression, * is the convolution operator and should not be taken to mean simple multiplication. The Fourier transform (or z-transform in the case of digital filters) reveals that filtering is synonymous with convolution. This is the "secret" of digital filters — by using relatively simple operations (add/multiply/delay), a filtering operation can be realized. The trick, of course, is coming up with the proper

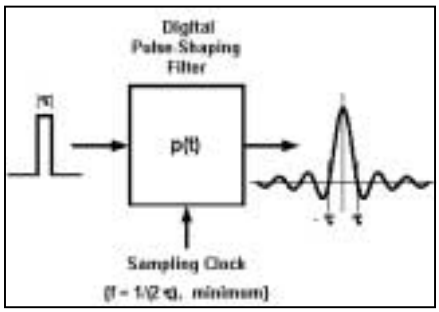


Figure 6. Converting an impulse to a raised cosine pulse by filtering.

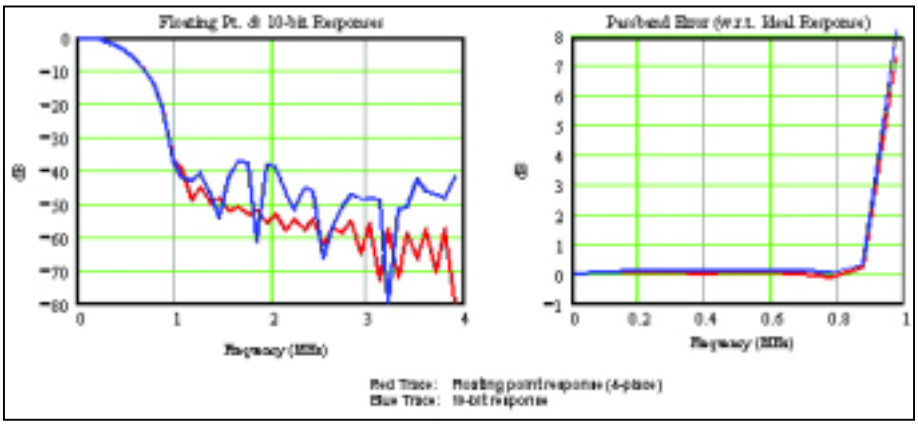


Figure 7. The frequency responses of two different versions of the raised cosine pulse-shaping filter.

$h(n)$ to produce the desired filtering operation (i.e., spectral shaping in the frequency domain). This is another topic altogether, but the many digital filter design tools that are available today make this process easier than ever before. These design tools give the designer the ability to generate the necessary filter coefficients for a desired frequency response (or vice versa).

Other variables that enter into the design process include determining the optimal number of filter coefficients and how much numeric precision (resolution) is required to get the job done.

Resolution refers to the number of bits used to represent the coefficient values, as well as the number of bits used to represent the sample values at any given point in the filter. Resolution affects the overall complexity of the design because more bits means more digital hardware.

How it comes together

Before proceeding with a design example, it should be noted from Figure 4b that $h(n)$ (the impulse response of the filter) is directly determined by the filter coefficients. This can be used to an advantage in the filter design process, because an outcome of the FIR filter design process is the impulse response, $h(n)$. The $h(n)$ values can be directly substituted for the a_i coefficients.

Step one

The first consideration in FIR filter design is the sample rate (f_s). This is the rate at which the internal delay stages are clocked. It turns out that the useful frequency response characteristic of any digital filter is limited to $\frac{1}{2}f_s$ (the Nyquist frequency), not f_s as one might assume. To demonstrate this concept, an arbitrary digital filter frequency response is shown in Figure 5. Now recall the raised cosine response (see Figure 2a), which can extend out to a frequency of $1/\tau$ (for $\alpha = 1$). If one were to operate a digital filter at the data rate ($1/\tau$), a problem would surface.

Specifically, the filter frequency response is restricted to the Nyquist rate (namely $\frac{1}{2}\tau$). The implication is that if a digital filter is used for pulse shaping, then it must operate at a sample rate of at least twice the data rate to span the frequency response characteristic of the raised cosine pulse. That is, the filter must oversample the data by at least a factor of two, preferably more.

Step two

The second consideration in FIR filter design is the number of tap coefficients (the a_i values). Typically, this is governed by two factors. The first is the amount of oversampling desired. More oversampling yields a more accurate frequency response characteristic. So a designer may elect to oversample by three, four or more. The second factor is the length of time that the filter's response is expected to span. Typically, this is determined by the number of bit (or symbol) intervals that the designer would like the filter response to occupy.

Remember, an FIR filter impulse response lasts only as long as the number of taps. If the filter oversamples by a factor of two and the desired impulse response duration is five bits (or symbols), then 10 taps are required (2 x 5 = 10). Obviously, a trade-off exists between the number of taps (circuit complexity) and the filter's response characteristic.

Why it works so well

The beauty of the pulse-shaping filter concept is that rectangular pulses can be used as the input to the filter.

Recall that the basic filtering process is synonymous with convolution in the time domain. Also recall that digital filters provide a convolution operation. For example, the filter impulse response $h(n)$ is convolved with the input samples to yield the output samples. The convolution of a rectangular pulse (more specifically, a unit impulse) with a raised cosine impulse response results in a raised cosine pulse at the output (see Figure 6). The input to the filter is a 1 or 0 (scaled to occupy the full bit width of the filter's input word size) and the output is a raised cosine pulse with all of the time and frequency domain advantages that such a pulse offers. All that is required is a digital-to-analog converter (DAC) at the output of the filter to convert the digital samples into an analog waveform.

Next, examine a detailed example of a raised cosine pulse-shaping filter design. Consider a system in which data must be transmitted at a rate of 1 Mb/s (i.e., $\tau = 1$ µs). One is also told that the timing jitter present at the

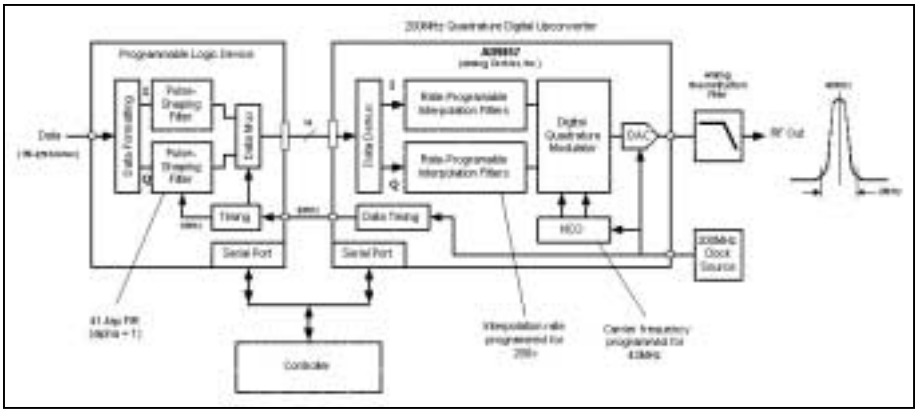


Figure 8. A block diagram of one potential RF data transmitter design using digital filters.

receiver is not known. Another design constraint is that the digital circuitry used to construct the digital filter will operate at a maximum rate of 50 MHz. Additionally, it has been given as part of the design requirement that the filter impulse response span at least five symbol periods.

In the absence of specific knowledge about timing jitter at the receiver, one is forced to assume the worst. This implies that a value of $\alpha = 1$ be used to maximize the decay of the pulse tails.

From Figure 2a, it can be seen that this corresponds to a single-sided bandwidth of $1/\tau$ (1 MHz), which means that the medium over which the data are transmitted must be able to support a 2 MHz bandwidth (the double-sided bandwidth of the raised cosine spectrum). If the medium cannot support 2 MHz, then one must consider a means of squeezing more bits into the same bandwidth. This can be done by a variety of modulation schemes (QPSK, 16-QAM, etc.). In this example, it is assumed that a bandwidth of 2 MHz is acceptable.

Because it has been determined that $\alpha = 1$, it is necessary to operate the filter at a sample rate of no less than twice the data rate (or two samples per symbol). However, to provide a more accurate spectral shape, one may choose to oversample by a factor of eight (i.e., eight samples/symbol). This means that the digital filter must operate at a rate of 8 MHz. This is well within the specified 50 MHz operating range of the digital circuitry, so the design is not in jeopardy.

It has been given that the filter impulse response be designed to span five symbols, so the filter must contain at least 40 taps (eight samples/symbol x five symbols = 40 samples). This will provide the required duration of the impulse response. However, 41 taps will be chosen to avoid the half-symbol delay associated with an even number of taps.

With α, τ, and the number of taps defined, Equation 2 can now be used to generate the filter taps. The value of t is determined at increments of 125 ns (the sampling period of the filter when operating at 8 MHz). The center of the impulse response is given the value of $t = 0$. Thus, the first value of t is 20 samples prior, or $t = -2.5$ µs.

How it all came together

The author used a PC-based version of Mathcad to compute $p(t)$ using the above information (see Appendix). Any suitable math program will do the job (MatLab, Excel, etc.). It turns out that because $p(t)$ was computed at the filter sample points, the values of $p(t)$ correspond one-to-one with $h[n]$, the impulse response of the filter. The results, rounded to four decimal places, are listed in Table 1.

Note the symmetry of the $h(n)$ values about $n = 0$. This redundancy can be used to simplify the implementation of the filter hardware. Because the filter is of the oversampling variety, further hardware simplification can be gained by using a polyphase architecture.

n	*h*(*n*)	*n*
-20	0	20
-19	-0.0022	19
-18	0.0037	18
-17	0.0031	17
-16	0	16
-15	0.0046	15
-14	0.081	14
-13	0.0072	13
-12	0	12
-11	0.0125	11
-10	0.0234	10
-9	0.0246	9
-8	0	8
-7	0.0624	7
-6	0.1698	6
-5	0.3201	5
-4	0.5	4
-3	0.686	3
-2	0.8488	2
-1	0.9603	1
0	1	0

Table 1. Impulse response values for p(t) of the filter.

If the filter is designed with floating point multipliers and adders, then the design is essentially done.

In the case of finite arithmetic, 10 bits is probably the minimum acceptable resolution to handle tap coefficients that span four decimal places. Formatted as twos-complement numbers, this will allow a range of –1.000 to +0.998046875 with a resolution of 2^{-10} (0.0009765625). This means that the multiply and add stages of the filter must be designed to handle 10-bit words. Also, the coefficients should be scaled by some fractional value to avoid overflow conditions in the hardware. A generally accepted scale factor is given by:

$$\mathbf{SF} = [\sum h(k)^2]^{-1/2}.$$

In words, it is the reciprocal of the square root of the sum of the square of each tap value. For the current example, the scale factor is: SF = 0.408249. After multiplying the $h(n)$ by SF, the resulting values are then converted to 10-bit words. Figure 7 shows the frequency responses of two versions of the raised cosine pulse-shaping filter. One is a floating-point version of the filter with the coefficients rounded to four decimal places. The other is a scaled, 10-bit, finite-math version of the filter. Both responses behave well in the passband, but the floating-point version exhibits better out-of-band attenuation. Also shown for comparison's sake is the passband error relative to the ideal response (Equation 1). Note that both responses exhibit less than 0.2 dB error over a range of about 90% of the passband.

The final frontier

With an understanding of how to create digital pulse shaping filters, the RF engineer can take on a larger role in the design of digital transmission systems. This is especially true today with semiconductor manufacturers offering highly integrated, high-speed, mixed-signal ICs. Figure 8 shows a detailed block diagram of a possible RF data transmitter design. The design is almost completely contained in two ICs (assuming a PC or other external device serves as the serial port controller).

RF

References

[1] Gibson, J. D., "Principles of Analog and Digital Communications," 2nd ed., Prentice Hall, 1993.

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

[3] The World Book Encyclopedia, Volume 13, World Book, Inc., 1994.

About the author

Ken Gentile graduated with honors from North Carolina State University where he received a B.S.E.E degree in 1996. Currently, he is a system design engineer at Analog Devices, Greensboro, NC. As a member of the Signal Synthesis Products group, he is responsible for the system level design and analysis of signal synthesis products. His specialties are analog circuit design and the application of digital signal processing techniques in communications systems. He can be reached at 336-605-4073 or by e-mail at *ken.gentile@analog.com*.

APPENDIX

Mathcad Spreadsheet for Raised Cosine Digital Filter Design

Ken Gentile Analog Devices, Inc., Greensboro, NC 27409

Functions:

Decibel: $dB(x) := if(|x| = 0, -200, 20 \cdot \log(|x|))$ Sinc: $sinc(x) := if\left(x = 0, 1, \frac{\sin(\pi \cdot x)}{\pi \cdot x}\right)$

Raised cosine spectrum: $P(\omega, \alpha, \tau) := \begin{cases} \tau & \text{if } (0 \le \omega) \cdot \left[\omega \le \frac{\pi}{\tau}(1-\alpha)\right] \\ \left[\frac{\tau}{2}\left[1 - \sin\left[\frac{\tau}{2\alpha}\left(\omega - \frac{\pi}{\tau}\right)\right]\right]\right] & \text{if } \left[\frac{\pi}{\tau}(1-\alpha) \le \omega\right] \cdot \left[\omega \le \frac{\pi}{\tau}(1+\alpha)\right] \\ 0 & \text{otherwise} \end{cases}$

Raised cosine impulse response: $p(t, \alpha, \tau) := \begin{cases} \left(\frac{\pi}{\tau} \cdot sinc\left(\frac{t}{\tau}\right)\right) & \text{if } \left|\frac{2 \cdot \alpha \cdot t}{\tau}\right| = \frac{1}{2} \\ sinc\left(\frac{t}{\tau}\right) \cdot \frac{\cos\left(\alpha \cdot \pi \cdot \frac{t}{\tau}\right)}{1 - \left(2 \cdot \alpha \cdot \frac{t}{\tau}\right)^2} & \text{otherwise} \end{cases}$

Calculate filter tap coefficients:

System sample rate: $Fs := 8 \cdot 10^6$ Sample period: $\Delta t := \frac{1}{Fs}$

Raised cosine alpha value: $alpha := 1$ Pulse period: $tau := 10^{-6}$

Number of taps: $N := 41$ Time index: $i := 0 .. N - 1$ Filter sample time values: $t_i := -2.5 \cdot 10^{-6} + i \cdot \Delta t$

Compute tap coefficients: $h := \overrightarrow{p(t, alpha, tau)}$ Taps rounded to 4 decimal places: $h4 := round(h, 4)$

Tap coefficient scale factor: $SF := \left(\sqrt{h \cdot h}\right)^{-1}$ $SF = 0.408248$ Scale taps by SF: $h_i := SF \cdot h4_i$

Quantize scaled taps to 10-bit integers: $h10_i := floor\left(2^9 \cdot h_i\right)$

Write tap values to a file: $WRITEPRN("RaisedCosineTaps") := h4$

Plot filter response:

Frequency points: $f_i := \frac{i}{N} \cdot \frac{Fs}{2}$

Normalized response of floating point filter (4 decimal places): $G1_i := dB\left(\left|gain\left(h4, \frac{f_i}{Fs}\right)\right|\right)$ $G1 := G1 - max(G1)$

Normalized response of 10-bit filter: $G2_i := dB\left(\left|gain\left(h10, \frac{f_i}{Fs}\right)\right|\right)$ $G2 := G2 - max(G2)$

Normalized response ideal raised cosine filter: $G3_i := dB\left(|P(2 \cdot \pi \cdot f_i, alpha, tau)|\right)$ $G3 := G3 - max(G3)$

Error of designed filter response relative to ideal: $Err1 := G1 - G3$ $Err2 := G2 - G3$