

## Matched filtering and timing recovery in digital receivers

*A practical look at methods for signal detection and symbol synchronization.*

By Louis Litwin

The acquisition of a signal in a digital communications system requires the convergence of several signal processing algorithms before the receiver can output meaningful data. These algorithms are adaptive in nature and need to process multiple received symbols before convergence is achieved. Because of the feedback nature inherent in these algorithms, the various adaptive receiver sections are often referred to as loops. Certain receiver loops depend on other loops. Depending on the algorithm implemented, it is possible that a given loop cannot converge until one or more previous loops have sufficiently converged. The major receiver loops are listed in the order that they typically need to converge, although the order may sometimes vary depending on the implementation.

### Stage 1 – AGC

This stage scales the signal to a known power level. Automatic gain control (AGC) is typically han-

dled in the analog domain to properly scale the signal for analog-to-digital (A/D) conversion because A/D converters have a limited dynamic range. If the received signal strength is too high, the A/D conversion process will introduce a type of distortion known as clipping. If the signal strength is too low, the signal variations will toggle only a few bits at the A/D, and distortion will occur because of severe quantization.

The convergence of the AGC loop is also required for several other receiver blocks. Certain parameters and gains for various adaptive algorithms, as well as boundaries for symbol decision regions at the slicer, are based on the signal being at a known power level. In addition to the analog AGC, many receivers implement an additional AGC in the digital domain for fine signal scaling.

### Stage 2 – timing recovery

The purpose of the timing recovery loop is to obtain symbol synchronization. Two quantities must be determined by the receiver to achieve symbol synchronization. The first is the sampling frequency. Locking the sampling frequency requires estimating the symbol period so that samples can be taken at the correct rate. Although this quantity should be known (e.g., the system's symbol rate is specified to be 20 MHz), oscillator drift will introduce deviations from the stated symbol rate.

The other quantity to determine is sampling phase. Locking the sampling phase involves determining the correct time within a symbol period to take a sample. Real-world symbol pulse shapes have a peak in the center of the symbol period. Sampling the symbol at this peak results in the best signal-to-noise-ratio and will ideally eliminate interference from other symbols. This type of interference is known as intersymbol interference.

### Stage 3 – carrier recovery

An oscillator at the transmitter generates a sinusoidal carrier signal that ideally exists at some known carrier frequency. Due to oscillator drift, the actual frequency of the carrier will deviate slightly from the ideal value. This carrier is multiplied by the data to modulate the signal up to a passband center frequency. At the receiver, the passband signal is multiplied by a sinusoid generated by the local oscillator.

Preferably, the frequency of the local oscillator will exactly match the frequency of the oscillator used at the transmitter. In practice, their frequencies differ and, instead of demodulation bringing the signal to baseband, the signal will be near baseband with some frequency offset. The presence of this frequency offset will cause the received signal constellation to rotate. This “spinning” effect must be removed before accurate symbol decisions can be made. The purpose of the carrier recovery loop is to remove this frequency offset so that the signal can be processed directly at baseband.

### Stage 4 – channel equalization

Transmitting a signal through a multipath channel results in a received signal that consists

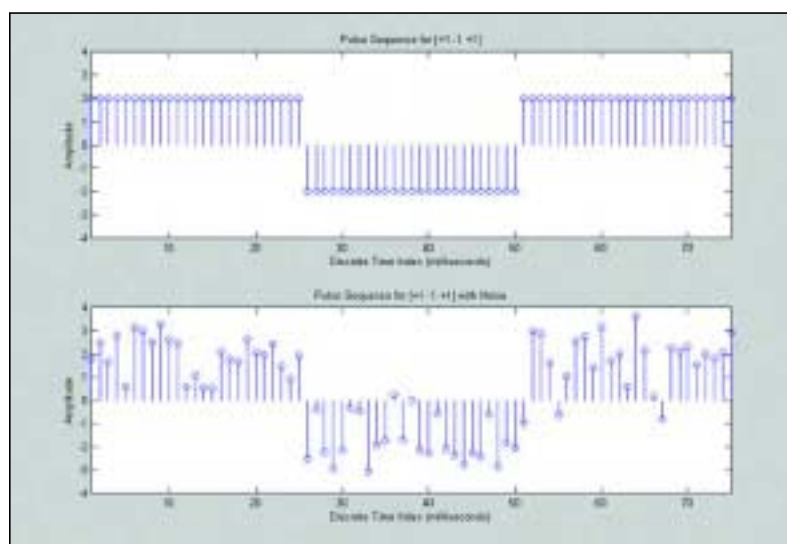


Figure 1. Rectangular pulse train shown with and without noise. The SNR for the lower plot is roughly 5 dB.

*Continued on page 36*

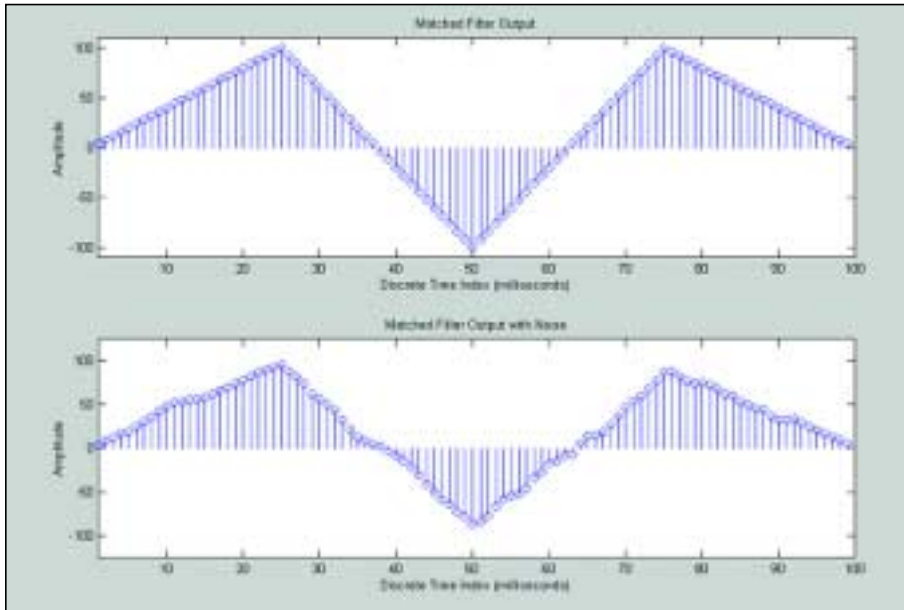


Figure 2. Matched filter outputs for signals in Figure 1.

of several delayed and scaled versions of the transmitted signal. Multiple versions of the signal occur because the receiver may pick up the signal that traveled the direct path from transmitter to receiver, as well as multiple reflected paths. The multipath channel can be viewed as a linear filter. The equalizer is an adaptive filter that attempts to remove intersymbol interference by undoing the filtering effects of the multipath channel.

Timing recovery algorithms adaptively determine the correct time to sample the symbol pulse shape. Thus, before entering into a discussion on timing recovery, some background material will be provided on the topics of matched filtering and pulse shaping.

### Signal detection

A basic problem in digital communications is the detection and estimation of a transmitted pulse in the presence of additive white Gaussian noise (AWGN). Imagine the simple case of a rectangular pulse, such as that shown in the top half of Figure 1. A data symbol of +1 is indicated by transmitting a pulse with an amplitude of +2, and similarly, a data symbol of -1 is indicated by transmitting a pulse with an amplitude of -2. The period of these pulses,  $T$ , is 25 ms. Note that the one pulse is simply a negated version of the other.

When two pulse shapes are used that have the same energy and a cross-correlation of -1, the signaling set is said to be antipodal. The estimation of the trans-

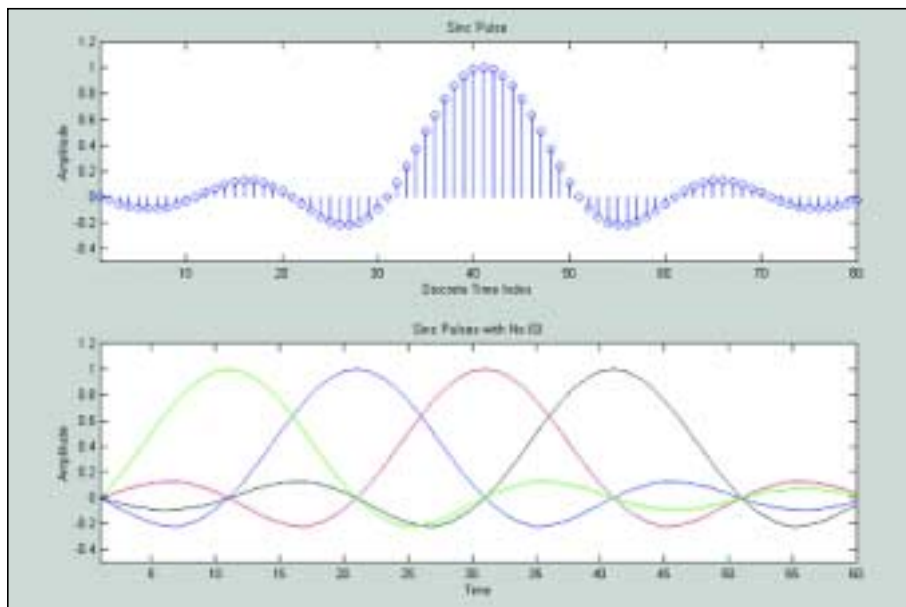


Figure 3. Sinc pulse examples.

mitted pulse shape is trivial for the case of no noise. The receiver simply takes one sample every  $T$  seconds and determines whether the sample equals +2 or -2.

Such a scheme no longer works in the presence of AWGN. White noise has infinite average power and can therefore easily drown out the received signal that is of limited power. The lower half of Figure 1 shows the same pulse sequence for the case of noise with a signal-to-noise ratio (SNR) of 5 dB. Note that the noise has severely distorted the signal, even flipping the sign of some samples. Because all practical communications systems have some non-trivial noise level, a more robust signal estimation scheme is needed.

### Matched filtering

Practical receivers estimate the transmitted signal by using a technique known as matched filtering. A receiver employing such a technique filters the received signal with a filter whose shape is “matched” to the transmitted signal’s pulse shape. The output of the filter is then sampled at time  $T$ . The matched filter’s pulse shape is a time-reversed version of the transmit pulse shape. Thus, if the transmit pulse shape  $h(t)$  is defined as:

$$h(t) \text{ for } 0 \leq t \leq T$$

then the ideal matched filter’s response  $h_m(t)$  is:

$$h_m(t) = h(T-t) \text{ for } 0 \leq t \leq T$$

Such processing has two advantages. One advantage is that typical pulse shapes have a low-pass response. By filtering the received signal with such a filter at the receiver, the frequencies containing the data signal are passed while the remaining frequencies are attenuated. This matched filtering limits the amount of the noise spectrum that is passed on to subsequent stages in the receiver. A second advantage is that a matched filter correlates the received signal with the transmit pulse shape over the symbol period  $T$ .

Recall that passing a signal  $r(t)$  through a filter  $h_m(t)$  is a convolution operation. The convolution of these two signals can be written as:

$$y(t) = \int_T^0 r(t)h_m(T-t)dt$$

where  $y(T)$  represents the output of the matched filter sampled at time  $T$ . However, the matched filter’s response

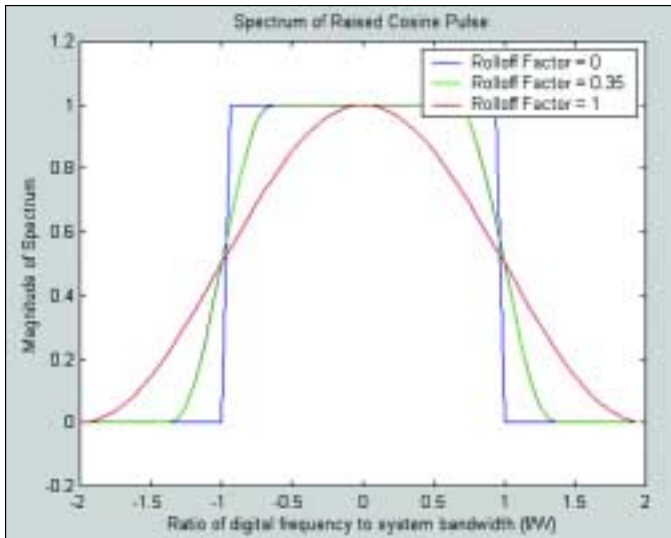


Figure 4. Spectrum of raised cosine pulse for different values of the roll-off factor.

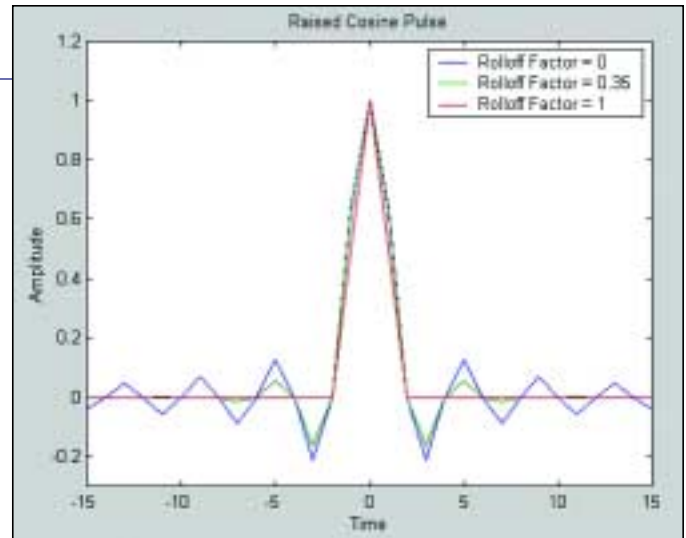


Figure 5. Raised cosine pulse for different values of the rolloff factor.

was defined as  $h_m(t) = h(T - t)$ . By substituting this definition into the above equation, the following integral is obtained:

$$y(t) = \int_0^T r(t)h(T - (T - t))dt = \int_0^T r(t)h(t)dt$$

The above equation is the cross-correlation (sampled at time  $T$ ) of  $r(t)$  with  $h(t)$  for a lag of 0. Thus, this simple derivation has illustrated how matched filtering effects the correlation of the received signal with the matched filter. Such processing results in a correlation gain by integrating the received signal energy while averaging out the zero-mean AWGN.

An example of matched filtering is shown in Figure 2. The received signals for the top and bottom halves of the figure are the signals shown in Figure 1. The matched filter used was:

$$h_m(t) = 2 \text{ for } 0 \leq t \leq T$$

Note that sampling the matched filter output at time  $T = 25$  ms provides the sample with the highest SNR. The samples from Figure 1 had an amplitude of 2, whereas the matched filter output (when sampled properly) has a value of 100. The value of 100 represents the integral, over the time period  $T$ , of the received signal pulse shape exactly lined up with the matched filter response. The value of this peak can be calculated as follows:

$$y(t) = \int_0^{25} (2 \cdot 2) dt = 25(4) - 0(4) = 100$$

This simple example illustrates how matched filtering provides the receiver

with a stronger signal to work with compared to directly sampling the received signal. The processing gain of matched filtering is especially apparent for the example with an SNR of 5 dB.

Note that the received signal is severely distorted by noise, but the matched filter's output is still close to its ideal value for the case of no noise. This result is possible because the matched filter filters out the higher frequency noise and then integrates the remaining lower frequency noise over a time period of  $T$  ms. Because AWGN is zero-mean, this integration effectively averages out the noise.

As can be seen from Figure 2, it is important to sample the matched filter's output exactly at time  $T$  to obtain the sample with the highest SNR. Sampling the matched filter's output at some time  $T + \Delta$ , (where  $\Delta$  represents a receiver timing offset) will significantly reduce the effective SNR seen by subsequent receiver blocks. This example shows the importance of keeping  $\Delta$  as close to zero as possible and thus provides motivation for the inclusion of a timing recovery loop in the receiver.

Before discussing specific timing recovery algorithms, the next sections will first illustrate the problems inherent in using this rectangular pulse shape. A more practical pulse shape known as a root-raised cosine pulse will then be introduced.

### Ideal pulse shaping

Although the use of matched filtering gives the optimum performance in the presence of AWGN, there is still a problem with using a rectangular pulse shape. Recall from Fourier theory that a rectangular pulse in the time domain is equivalent to a sinc pulse in the frequency domain. Because the tails of the

sinc pulse extend to infinity, such a pulse shape would require a system with infinite bandwidth.

The ideal pulse shape should have two properties. It should have a limited bandwidth to allow transmission on practical band-limited systems. The pulse shape should also have zero intersymbol interference if sampled at the correct time interval. That is, when a pulse train is sampled every  $T$  seconds, the value of the sample at time  $T$  should only be due to the current pulse. And there should be no interference from the other transmitted pulses.

In other words, ideally,  $h(t) = 1$  for  $t = 0$  and  $h(t) = 0$  for  $t = \pm kT$  where  $k$  is a non-zero integer. An ideal pulse shape that meets these requirements is a time-domain sinc pulse. An example of a sinc pulse for which  $T = 10$  is shown in Figure 3. Note that the pulse takes on a value of 1 at its peak and its zero-crossings occur at intervals of integer multiples of  $\pm 10$  samples away from the peak. The lower half of the figure shows a pulse train of four pulses. This example illustrates how the peak of any given pulse lines up with the zero-crossings of the remaining pulses. Therefore, there is no ISI.

### Practical pulse shaping

Although the sinc pulse represents the ideal pulse shape, it cannot be implemented in practice because the pulse extends in time for infinite duration. The infinite signal duration is due to the discontinuities in the sinc pulse's rectangular-shaped spectrum. Signals with discontinuities in their spectrum are physically unrealizable. However, practical pulse shapes can be formed by smoothing the roll-off of the spectrum and allowing it to occupy excess bandwidth beyond that which is needed for

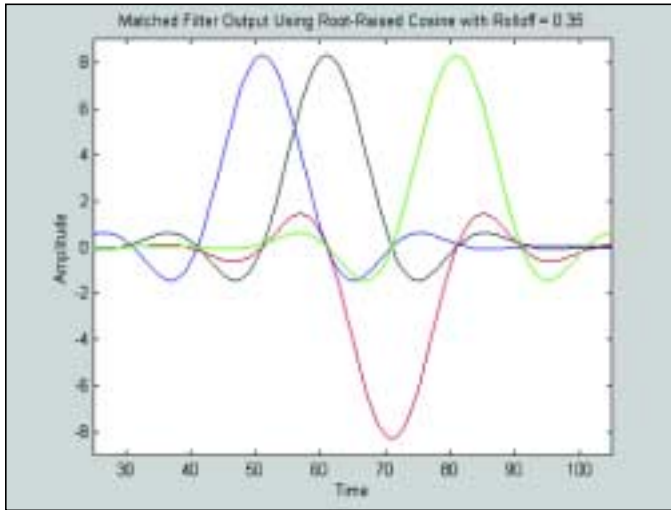


Figure 6. Zero ISI at output of matched filter using a root-raised cosine pulse.

the spectrum of the ideal sinc pulse.

One pulse shape that has properties similar to the sinc pulse, but without the frequency-domain discontinuities, is the raised cosine pulse. The raised cosine pulse has a parameter known as the rolloff factor. The value of the rolloff factor determines how rapidly the frequency-domain spectrum of the pulse rolls off. The raised cosine pulse is identical to the sinc pulse when the rolloff factor is equal to zero. As the rolloff factor is increased, the spectrum begins to decay more gradually and this increased rolloff causes the pulse to occupy more bandwidth. When the rolloff reaches its maximum value of one, the spectrum requires twice as much bandwidth as the pulse with a rolloff of zero. Practical digital communications systems often use a rolloff factor of between 0.10 and 0.35. A pulse with a rolloff factor of 0.35 occupies 35% more bandwidth than the ideal sinc pulse. Figure 4 shows the effect of the rolloff factor on the pulse's spectrum. Figure 5 contains time-domain raised cosine pulses for the same rolloff factors used in Figure 4.

The pulses in Figure 5 exhibit zero-crossings at integer multiples of the symbol period. Thus, even with non-zero roll-off factors, the raised cosine pulse maintains this desirable (from the standpoint of no ISI) property of the sinc pulse. The choice of the roll-off factor is a trade-off between required bandwidth and the duration of the time-domain pulse. Note that the tails of the time-domain pulse are reduced for higher values of the roll-off factor. The smaller tails are desirable from a

timing recovery standpoint because, in the presence of a timing offset, they will contribute less to ISI compared to the larger tails of the sinc pulse.

The most popular pulse shape used in practical communications systems is the root-raised cosine pulse. The root-raised cosine pulse is formed by taking the square root of a raised cosine pulse. This pulse shape is used to split the spectral characteristics of the raised cosine pulse equally between the transmitter and receiver.

By matched-filtering the root-raised cosine pulse and then sampling it at the symbol period, the root-raised cosine pulse is essentially squared. Thus, the output of the matched filter has a raised cosine pulse response.

An example of the matched filter output for a pulse train of root-raised cosine pulses with a rolloff factor of 0.35 is shown in Figure 6. Note that the matched filter output exhibits zero ISI because of the locations of the zero crossings for the case of perfect timing.

### Timing recovery

The previous sections have shown how intersymbol interference can be avoided by sampling the matched filter output at its peak, which occurs every  $T$  seconds. The purpose of the timing recovery loop is to alter, as necessary, the sampling frequency and sampling phase to sample the matched filter at the peaks. If the timing recovery loop is operating properly, it will provide the downstream processing blocks with symbols that were sampled at the highest SNR points available.

An example of a typical all-digital

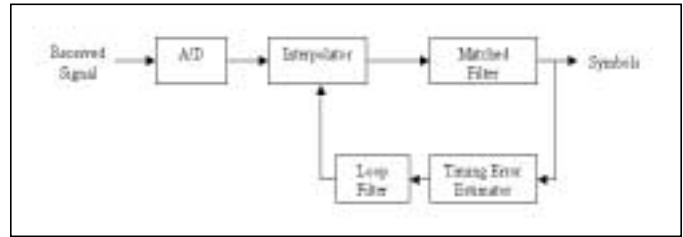


Figure 7. Example of an all-digital timing recovery loop.

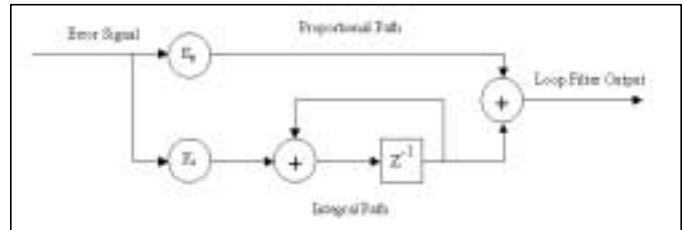


Figure 8. Structure of a typical second-order loop filter.

timing recovery loop is shown in Figure 7. After A/D conversion, the signal is passed through an interpolator. The interpolator is able to generate samples in between those actually sampled by the A/D (i.e., it interpolates). By generating these intermediate samples as needed, the interpolator can adjust the effective sampling frequency and phase.

Interpolation is accomplished by first inserting  $N-1$  zeros in between the data samples (upsampling by a factor of  $N$ ). The upsampled signal passes through a lowpass interpolation filter to remove the aliases caused by upsampling. The resulting interpolated signal is a smoothed version of the original signal and it contains  $N$  times as many samples.

Following interpolation, the output of the matched filter is sent to a timing error estimator that can use a number of different algorithms to generate a timing error. The control signal for the interpolator is formed by filtering this error signal using a standard second-order loop filter containing a proportional and an integral section. An example of a typical second-order loop filter is shown in Figure 8.

The second-order loop filter consists of two paths. The proportional path multiplies the timing error signal by a proportional gain  $K_p$ . From control theory, it is known that a proportional path can be used to track out a phase error; however, it cannot track out a frequency error. For a timing recovery loop to track out a sampling frequency error, a loop filter containing an integral path is needed. The integral path multiplies the error signal by an integral gain  $K_i$ ,

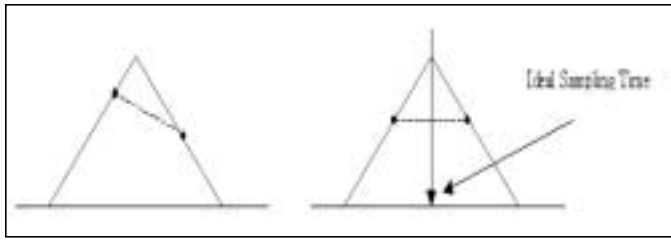


Figure 9. Method of generating error for early-late gate algorithm. The left plot shows where sampling is occurring too late. When sampling occurs at the right time, the early and late samples will be at the same amplitude.

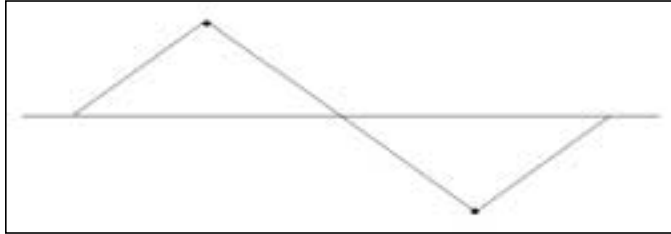


Figure 10. Correct timing:  $e_n = (-1 \cdot 1) - (-1 \cdot 1) = 0$ .

and then integrates the scaled error using an adder and a delay block. A second-order filter, such as that shown in Figure 8, can track out both a sampling phase and a sampling frequency error.

### Early-late gate algorithm

This timing recovery algorithm generates its error by using samples that are early and late compared to the ideal sampling point. The generation of the error requires at least three samples per symbol. The method of generating the error is illustrated in Figure 9. The left plot is for the case where sampling is occurring late. Note that the early and late samples are at different amplitudes. This difference in amplitude is used to derive an error for the timing recovery loop. Once the timing recovery loop converges, the early and late samples will be at equal amplitudes. The sample to be used for later processing is the sample that lies in

the middle of the early and late samples. One drawback of the early-late gate algorithm is that it requires at least three samples per symbol. Thus, it is impractical for systems with high data rates.

### Mueller and Muller Algorithm

The Mueller and Muller algorithm only requires one sample per symbol. The error term is computed using the following equation:

$$e_n = (y_n \cdot y_{n-1}) - (y_n \cdot y_{n-1})$$

where  $y_n$  is the sample from the current symbol and  $y_{n-1}$  is the sample from the previous symbol. The slicer (decision device) outputs for the current, and previous symbol are represented by  $\hat{y}_n$  and  $\hat{y}_{n-1}$ , respectively. Examples of the value for the Mueller and Muller error for the cases of different timing offsets are shown in

Figure 10, Figure 11 and Figure 12. One drawback of this algorithm is that it is sensitive to carrier offsets, and thus carrier recovery must be performed prior to the Mueller and Muller timing recovery.

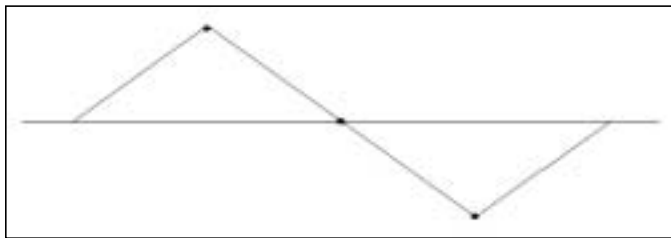


Figure 13. Correct timing:  $e_n = (-1 - 1) \cdot 0 = 0$ .

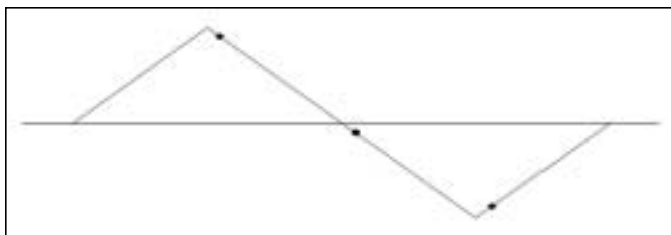


Figure 14. Timing is late:  $e_n = (-0.8 - 0.8) \cdot (-0.2) = 0.32$ .

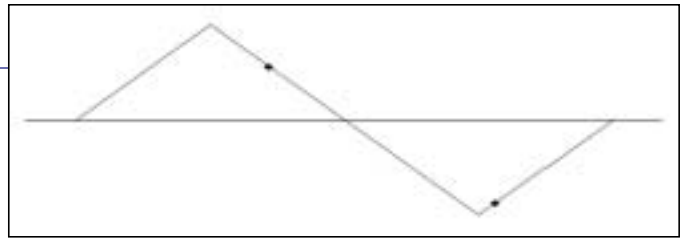


Figure 11. Timing is fast:  $e_n = (-0.8 \cdot 1) - (-1 \cdot 0.5) = -0.3$ .

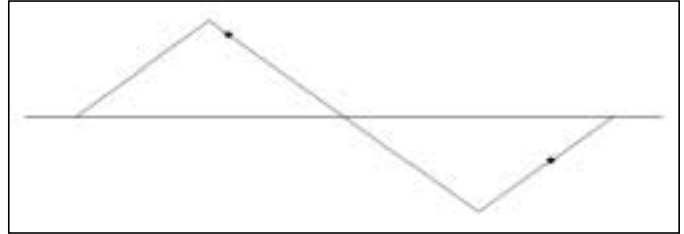


Figure 12. Timing is slow:  $e_n = (-0.5 \cdot 1) - (-1 \cdot 0.8) = 0.3$ .

### Gardner algorithm

The Gardner algorithm has seen widespread use in many practical timing recovery loop implementations. The algorithm uses two samples per symbol and has the advantage of being insensitive to carrier offsets. The timing recovery loop can lock first, therefore simplifying the task of carrier recovery. The error for the Gardner algorithm is computed using the following equation:

$$e_n = (y_n - y_{n-2}) y_{n-1}$$

where the spacing between  $y_n$  and  $y_{n-2}$  is  $T$  seconds, and the spacing between  $y_n$  and  $y_{n-1}$  is  $T/2$  seconds.

The following figures illustrate how the sign of the Gardner error can be used to determine whether the sampling is correct (Figure 13), late (Figure 14) or early (Figure 15). Note that the Gardner error is most useful on symbol transitions (when the symbol goes from positive to negative or vice-versa). The Gardner error is relatively small when the current and previous symbol have the same polarity.

A simulation was run for a timing recovery loop that used the Gardner algorithm and the results are shown in Figure 16 (page 48). The top plot shows the matched filter output samples for the in-phase component of the signal.

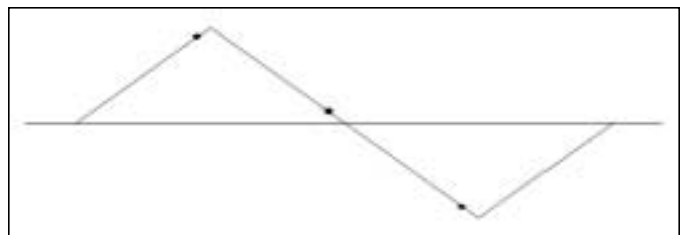


Figure 15. Timing is early:  $e_n = (-0.8 - 0.8) \cdot (0.2) = -0.32$ .

---

Notice that the timing recovery loop converges after about 600 symbols have been processed. At this point, the output of the matched filter takes on values of +1 and -1. The values are fairly constant because the matched filter output is being sampled near the ideal center point. During the first 600 symbols, when the loop is still converging, the matched filter samples take on

a wide range of amplitudes. This variance in the matched filter output is because of ISI caused by sampling the output at points other than the ideal center point. The bottom plot shows the Gardner error  $e_n$  vs. time.

#### Other methods of timing recovery

The ideal case is to have the transmitter and receiver running off of the

same clock. Although this situation is typically impossible in a wireless communications system, it can be implemented in some wired systems, such as computer networks. In such an ideal system, a timing recovery loop is not needed because synchronization is explicit.

Another alternative is to have the clock frequency transmitted along with the data. The receiver can recover this clock signal with a narrow-band bandpass filter tuned to that frequency. Although this method is used in some practical systems, it is generally inefficient because the transmission of the clock signal consumes both bandwidth and transmitter power that could have otherwise been used for sending user data. In addition, other decision-directed and non-decision-directed algorithms exist for generating an error signal for a timing recovery.

The Gardner's algorithm presented here represents a good starting point for practical implementations because of its robustness to carrier offsets, simple implementation and modest oversampling requirement of two samples per symbol. The interested reader can learn more about timing recovery algorithms by consulting the references listed at the end of this article.

#### Conclusions

This article presents the problem of detecting pulses transmitted across an AWGN channel. Merely sampling the pulses at the receiver once every symbol period is found to be ineffective because of the signal distortion due to the presence of noise with an infinite bandwidth. The concept of the matched filter receiver is introduced as a way to limit the noise at the receiver, as well as to provide a high SNR sampling point due to the correlation gain. The implementation of symbol timing synchronization is shown to be a vital process in obtaining the best SNR sampling point while also avoiding intersymbol interference.

**RF**

#### Acknowledgments

The author would like to thank Kumar Ramaswamy and Paul Knutson (both of Thomson Multimedia) for their help in proofreading this article and also for the numerous (and sometimes late-night) discussions on the topic of timing recovery.

*Continued on page 48*

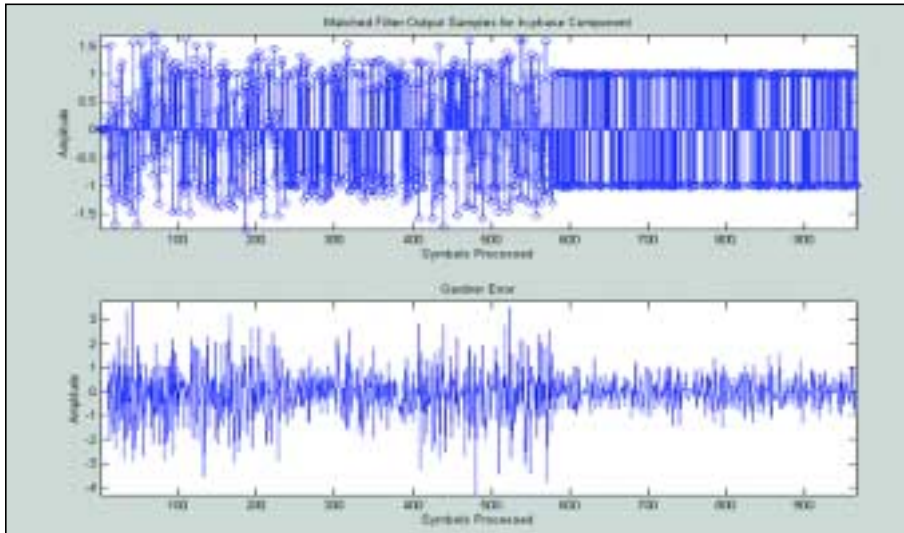


Figure 16. Simulation results using Gardner algorithm on QPSK data.

## References

[1] F. M. Gardner, "A BPSK/QPSK Timing Error Detector for Sampled Receivers," *IEEE Transactions on Communications*, vol. COM-34, pp. 423-429, May 1986.

[2] D. N. Godard, "Passband Timing Recovery in an All-digital Modem Receiver," *IEEE Transactions on Communications*, vol. COM-26, pp. 517-523, May 1978.

[3] S. Haykin, *Communication*

Systems, Wiley, NY, 1994.

[4] K. H. Mueller and M. S. Muller, "Timing Recovery in Digital Synchronous Data Receivers," *IEEE Transactions on Communications*, vol. COM-24, pp. 516-531, May 1976.

[5] J. G. Proakis, *Digital Communications*, McGraw-Hill, NY, 1995.

## About the author

Louis Litwin is a member of the technical staff with Thomson Multimedia Corporate Research where he is working on 3G CDMA technology for mobile applications. He received his M.S. degree in electrical engineering from Purdue University and his B.S. degree in electrical engineering from Drexel University.