

# Flash ADC takes the uncertainty out of high-speed data recovery

TOM NAPIER, CONSULTANT

Designing a robust bit synchronizer—a device that synchronizes to a noisy data stream and recovers the best possible binary data from it—is relatively simple when the input data is not too noisy and the data rate is fixed and accurately known. However, designing becomes much more difficult when the equipment must work over many decades in bit rate or with data whose rate or amplitude is unknown in advance.

At bit rates of 100 Mbps and higher, timing errors of a few hundred picoseconds cause significant data errors. Although you can adjust fixed-rate bit synchronizers to work at these rates, building a tunable bit synchronizer requires a fresh approach. The answer is simple: Measure both the data polarity and the input phase using the same device—a flash ADC. Then, the logic delays drop out of the equation.

The ADC samples the filtered input signal a few nanoseconds after each clock pulse, but the variation in the ADC's sampling delay is tens of picoseconds. Both phase and data samples are timed from identical clock edges. When one is correct, so is the other (see sidebar "A word on flash ADCs").

Data-recovery systems that sample the data are not new, but the published designs use a fixed sampling rate that is usually much higher than the data rate. When you are striving for extremely high data rates, the most practical system should sample as few times as possible per bit, sampling the signal at just the right point.

The current approach samples the data stream twice per bit using an ADC with a sampling rate that automatically adjusts to match the frequency and phase of the input signal. Flash ADCs are currently capable of operation at 1000M sam-

Traditional methods of recovering data and clock information from noisy baseband signals fail at data rates higher than 30 Mbps. Using the same flash ADC to sample both the input data and its phase eliminates timing errors and makes it possible to recover data at 500 Mbps.

ples/sec, so, in principle, you can recover data by this method as fast as 500M samples/sec. This technique allows a commercial product to operate at 210 Mbps using a 450M-sample/sec ADC.

Many articles describe the design and stabilization of PLLs—the circuits that synchronize an oscillator to the frequency and phase of an input signal. Some of these

articles consider the effects of input jitter and noise, but nearly all of the articles assume that the input is continuous.

The recovery of clock and data information from the noisy, baseband, NRZ data that a receiver generates is an important PLL application. In this case, the input has a (sine x)/x spectrum. This signal contains no energy at the frequency to be recovered, and a conventional PLL cannot lock to it.

## Enter the bit synchronizer

A bit synchronizer is a device that synchronizes to a noisy data stream and recovers the best possible binary data from it. All data receivers contain some form of "bit sync." Although "bit-sync" generally applies to any device that extracts data and a synchronous clock from a binary data stream, the term usually refers to a design or a device that is tunable over a range of data rates and that can recover data that is heavily contaminated by noise (Figure 1).

A true bit synchronizer thus has to generate a jitter-free clock at the same mean rate as the input data and to reject noise to provide a data output having as few bit errors as possible. A bit synchronizer locks to the input signal even if the synchronizer's bit rate deviates several percentage points from the expected value. A bit sync filters noise and makes

## DATA RECOVERY USING A FLASH ADC

the best possible estimate of the original data bits. It adjusts the input signal amplitude to a standard value and removes any dc offset.

A practical bit synchronizer operates with signals that have bit rates in a six-decade range. Thus, every part of the bit synchronizer—filters, clocks, and control loops—must work anywhere in this range (typically at 10 bps to 10 Mbps). Because the system usually specifies the desired loop bandwidth as a percentage of the data rate (typically, 0.01% to 1%), you have to adjust the operating parameters of the PLL to set up the correct loop bandwidth at the chosen rate. Performing this adjustment with an analog loop requires many switched and variable components and may even require using several tuning approaches, according to the bit rate.

### Filter the input

If the RF and IF sections of the receiver have a wide enough bandwidth, the perfect noise filter is easy to specify. This filter integrates both signal and noise for the duration of a bit. The frequency response of this filter has the same  $(\sin x)/x$  shape as the spectrum of the input data. Assuming Gaussian noise and a perfect filter, you can compute the probability that the sign of the peak of the filtered data will differ from the true data polarity for any SNR. This probability is the bit-error rate (BER).

The telemetry world measures the SNR as the energy per bit divided by the noise ( $E_b/N_0$ ) expressed in decibels. The noise in this case is the measured noise in a bandwidth equal to the reciprocal of the bit period, that is, in roughly twice the data bandwidth. Because SNR measurements generally use equal data and noise bandwidths, the  $E_b/N_0$  ratio is 3 dB worse than the SNR. That is, an  $E_b/N_0$  of  $-1$  dB corresponds to an SNR of 2 dB.

The curve of theoretical BER versus  $E_b/N_0$  sets the standard to which the designer of bit synchronizers must aspire (Figure 2). Because the bit-error performance of a bit synchro-

nizer tends to parallel this curve, it is convenient to characterize the performance as being so many “decibels worse than theory.” The right-hand curve in Figure 2 shows a bit synchronizer that is 0.5 dB worse than theory. You can view this difference as the extra transmitter power you need to achieve a desired bit-error rate.

In many cases, you cannot increase the transmitter’s power; this power may not even be under the authority of the receiving agency. In this case, there is no substitute for using the best available bit synchronizer. A 2- to 3-dB difference from theory represents poor commercial performance, a 0.5- to 1-dB difference represents good performance, and a less-than-0.25-dB difference represents the best performance. However, the law of diminishing returns quickly sets in. The price difference between a design with poor performance and one with good performance is small, but it improves the  $E_b/N_0$  ratio by 1 or 2 dB. A further 0.5-dB improvement might double or triple the cost of the equipment and would be justified only in the most critical situations.

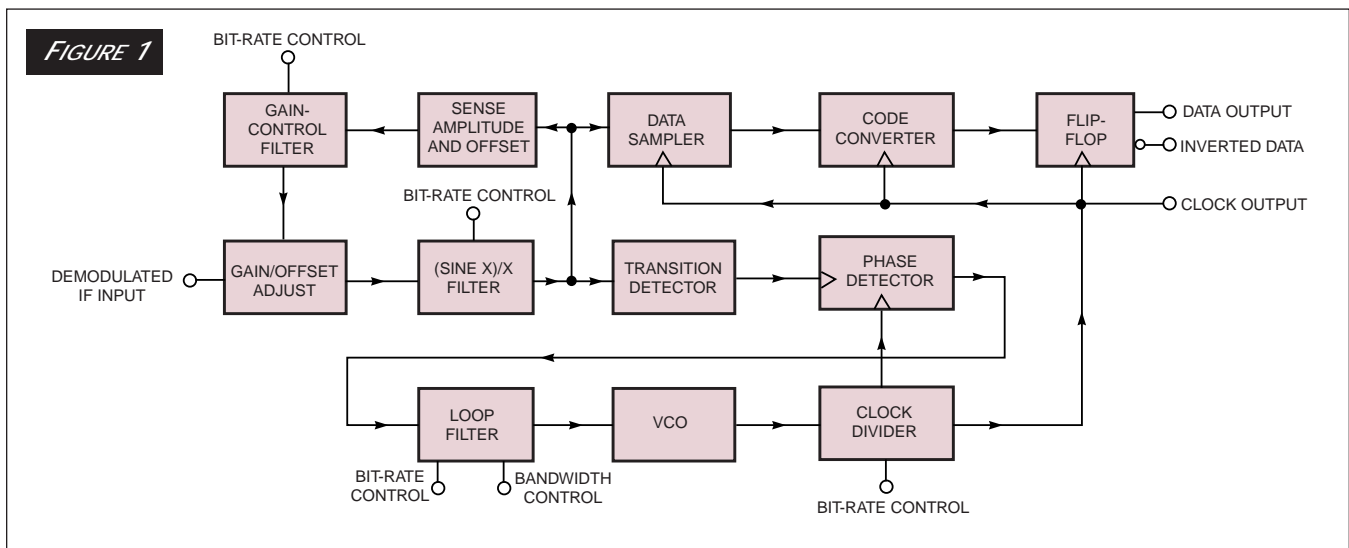
For example, as the right-hand side of the curve in Figure 2 shows, the theoretical curve is steep when the noise level is low. A 0.5-dB difference can improve the BER by a factor of five. When possible, error-correction encoding at the transmitter can also immensely improve the BER.

The perfect noise filter is mathematically easy to describe but difficult to implement even at a fixed frequency. Designing a perfect filter is hard if the bit-rate range is wide or if the data rate exceeds a few megahertz.

### The practical bit synchronizer

An ideal filter integrates both data and noise for the duration of 1 bit. The data integrates to a peak, and the noise tends to average out. Perfection in a bit synchronizer requires at least knowing exactly when each bit starts and stops so that each bit can be sampled at its peak.

The PLL in the clock-recovery circuit must know when



A bit synchronizer requires many functional blocks to generate a jitter-free clock at the same mean rate as the input data and to reject noise to reduce output-bit errors.

## DATA RECOVERY USING A FLASH ADC

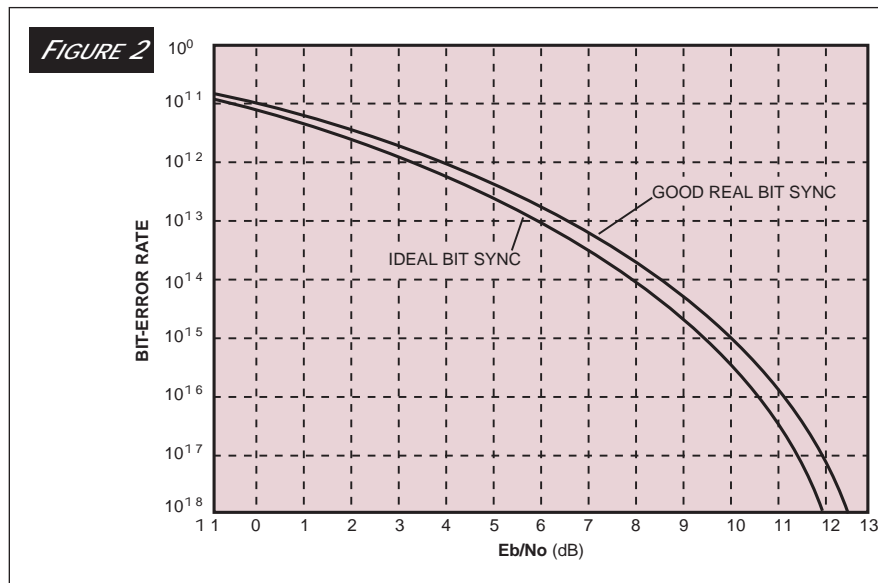
each bit stops and starts. Because the data contains no energy at the desired clock frequency, you must perform a nonlinear function on the data to determine the locations of the transitions. This function generally consists of filtering and then squaring the data or performing an absolute-value operation on the data. Clipping the result produces a pulse every time the data makes a transition. A conventional phase detector then generates an error signal that is proportional to the difference between the phase of this pulse and the locally generated clock. The phase detector drives the loop filter that controls the VCO, and the VCO generates the output clock.

Because phase information is available only when there is a transition in the data, the phase detector's gain is sensitive to the input-transition density. You can assume an average density of 50%, which is equivalent to one transition every 2 bits for random data.

However, some data patterns, such as the 01010101 pattern that often serves as the preamble to a data block and corresponds to a 100% transition density, can produce marked changes in the gain and damping of the loop. A good bit synchronizer retains lock in the absence of transitions; 512 successive one or zero bits is a common test case.

Among the many complications in a real design is the fact that the clock oscillator and the control loop must work equally well over a range of frequencies and loop bandwidths. Devising a way to switch the loop bandwidth without losing lock on the input signal is tricky.

Also, a real bit synchronizer contains more circuitry than mentioned. In most situations, a real design needs automatic-gain-control and offset-removal circuits. A bit synchronizer must also convert the data code the input stream uses to the data code the output requires. Some users require derandomizing of the data. Other users may apply convolutional encoding to the transmitted data and require a Viterbi decoder in the bit synchronizer. Most users require a signal to



The curve of theoretical BER versus  $E_b/N_0$  sets the standard to which all designers of bit synchronizers aspire. A good bit-sync design is only 0.5 dB worse than theory.

tell their downstream equipment when the synchronizer has achieved bit lock. All of these requirements add to the complexity of the product.

### Propagation delays limit performance

One flaw in the conventional bit-synchronizer design is that when the local clock phase-locks to the input, a gate driven by that local clock samples the input to determine the polarity of each data bit. Because of the propagation delay of the logic, a time difference exists between the clock edge and the ideal sampling time. Clever design can minimize but not remove this effect. For bit rates of a few kilohertz, this fixed-time delay is unimportant, but satellites now operate at data rates of tens and hundreds of megabits per second.

To put this problem in perspective, assume that the data rate is 10 Mbps, so that 1 bit is 100 nsec long. A difference of only one gate delay (for example, 5 nsec) is 5% of the bit period. On average, a timing error of this magnitude is equivalent to 0.45 dB in signal amplitude, which translates directly

## A WORD ON FLASH ADCs

A flash ADC converts the amplitude of an input signal into binary form in three stages. The ADC applies the input in parallel to many fast comparators whose thresholds are equally spaced throughout the desired input voltage range, typically 1V. At any moment, all the comparators that have thresholds below the input voltage are on, and the rest are off.

A series of latching AND gates connect between adjacent comparators so that only the gate at the boundary between the on and off comparators is active. The input clock latch-

es the AND gate outputs, and a pipelined circuit converts this one-out-of-N input to a binary output.

Flash ADCs need  $2^N$  comparators to generate an N-bit output, and thus they tend to have no more than 6 to 10 output bits. Because the input has to drive all the comparators in parallel, the input capacitance is a major limitation to the bandwidth. Luckily, the sampling bit synchronizer can often use this input capacitance as one element of the noise filter.

## DATA RECOVERY USING A FLASH ADC

into a 1-dB difference from theory and hence increases BER. At 50 Mbps, a 5-nsec gate delay represents one-fourth of the bit period, and the performance drops by 2.5 dB, which is unacceptable.

### Flash-ADC design provides the answer

A design that uses a flash ADC to sample both the data polarity and the input phase solves many of these problems by taking the logic delays out of the equation.

Luckily, an ideal noise filter converts all signal transitions into a 1-bit-period ramp whose zero-crossing occurs at the middle of the bit. Sampling this ramp near the midbit generates an output proportional to the deviation of the sampling time from the midbit position. If a flash ADC operating at twice the bit rate samples the filtered data, alternate samples will contain phase information and data. Setting the phase error to zero automatically sets the data-sample time to the data peak.

**Figure 3** shows the input NRZ signal, the filter output, and the resulting phase and data samples. The phase detector must correct the sign of the phase error, depending on the direction of the transition, and must ignore the phase sample when there is no transition. In the absence of transitions, the phase error is plus or minus full-scale. If the phase detector successfully performs these tasks, the resulting sample amplitude is proportional to the phase error (**Figure 3**).

### The phase detector

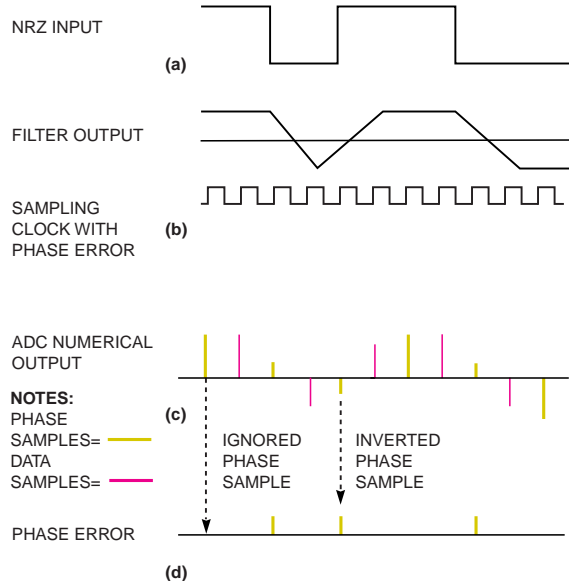
Computing the phase error requires three successive ADC outputs. Multiplying the second output by the sign of the first generates a unidirectional error. Comparing the signs of the first and third samples determines if the samples differ. If these samples differ, the phase sample is valid. However, if the first and third samples are the same, indicating that there is no transition, the phase detector generates a zero-error output. The design can process the resulting samples digitally or convert the samples into analog form and apply the result to a conventional PLL.

If you apply a half-LSB offset to the signal, multiplication of the phase sample by the sign of the data sample requires no more than complementing the phase sample with an array of exclusive-OR gates. Thus, the complete phase detector consists of the ADC, a few delay latches, one exclusive-OR gate for each bit of the sample, and another exclusive-OR gate to detect transitions (**Figure 4**). One AND gate follows for each sample bit to insert a null sample whenever there is no transition. For an offset binary output, the most significant null bit is set to one.

You can implement this system in an FPGA at data rates as high as 40 Mbps. The 210-Mbps version made use of 300k ECL logic; faster systems would require Motorola's ([www.motorola.com](http://www.motorola.com)) ECLips series parts.

The accuracy of the phase detector in **Figure 4** depends on the fact that the zero crossing during a transition coincides with the middle of the bit. It is almost impossible to achieve a filter with a perfectly linear output for each transition. However, you can build a filter that has a slightly S-shaped output. This shape has the correct zero-crossing position, and,

**FIGURE 3**



When typical noise-free BCD input data (a) drives an ideal filter, the filter's output (b) consists of a series of positive and negative ramps, each one bit-period long. The ADC samples the filtered output, producing alternating phase and data information at each rising edge of the sampling clock (c). The phase detector ignores phase errors that result from periods without transitions and inverts the necessary phase-error samples (d).

although this filter shape slightly changes the gain of the phase detector, the shape does not degrade the performance.

### Generate the clock

This phase detector actually accomplishes the functions of numerous blocks in **Figure 1**'s basic bit synchronizer, including the transition detector, the phase detector, the data sampler, and the sense-amplitude-and-offset block. The phase detector controls a VCO that runs at twice the bit rate. (At low rates, you can use a numerically controlled oscillator.) A divider circuit supplies the output bit-rate clock and specifies which of the alternate samples represents data and which represents phase.

The opposite phases of the bit-rate clock can serve as the "data" and "phase" clocks to demultiplex and process the alternate samples. Using opposite phases of the bit-rate clock halves the rate at which most of the circuit has to work. In the fastest implementation of this circuit, the discontinued Analog Devices AD9016 ADC contained both the clock divider and the data demultiplexer, greatly simplifying logic timing.

The start-up state of the clock divider is irrelevant because

