The Meaning of Total Jitter

Ransom Stephens, Ph.D.

Abstract:

Total Jitter is an increasingly important quantity in the development and specification of serial data links but, while it is well defined, it is not well understood. Total Jitter is like peak-to-peak jitter but referenced to a given Bit Error Ratio; it is the amount of eye-closure at a given BER. This paper clarifies the meaning of TJ and shows how measurements of jitter subcomponents, like Random Jitter (RJ) and Deterministic Jitter (DJ), can be used to provide fast accurate estimates of TJ(BER).

In analyzing jitter we have two separate but related concerns. First, we need a standard way of predicting how well two network elements, say a transmitter and a receiver, will work together and maintain an acceptable Bit Error Ratio (BER) – these are interoperability requirements. It is to this end that Total Jitter at a Bit Error Ratio was defined. We'll use the notation, TJ(BER), to keep things clear because the term "Total Jitter" is often used to describe quantities and distributions that are less well defined. Depending on the context, "Total Jitter" may refer to a measurement of the jitter Probability Density Function (PDF) which gives the probability of a logic transition occurring at a given time. A measurement of the jitter PDF can be performed on an oscilloscope, as shown in Figure 1, by making a histogram of the crossing point. "Total Jitter" is also used to distinguish the jitter of the whole system from jitter that is caused by its parts. In this paper, TJ(BER) is the total jitter defined at a bit error ratio.

Which brings us to our second concern in jitter analysis: diagnostics. If you have a problem with jitter – and that can only mean one thing, the BER is too high – just knowing the value of the BER, the peak-to-peak jitter, or even TJ(BER), doesn't help you solve the problem. For diagnostics, you need to break the jitter down into separate categories that, hopefully, point in the direction of the causes of jitter.

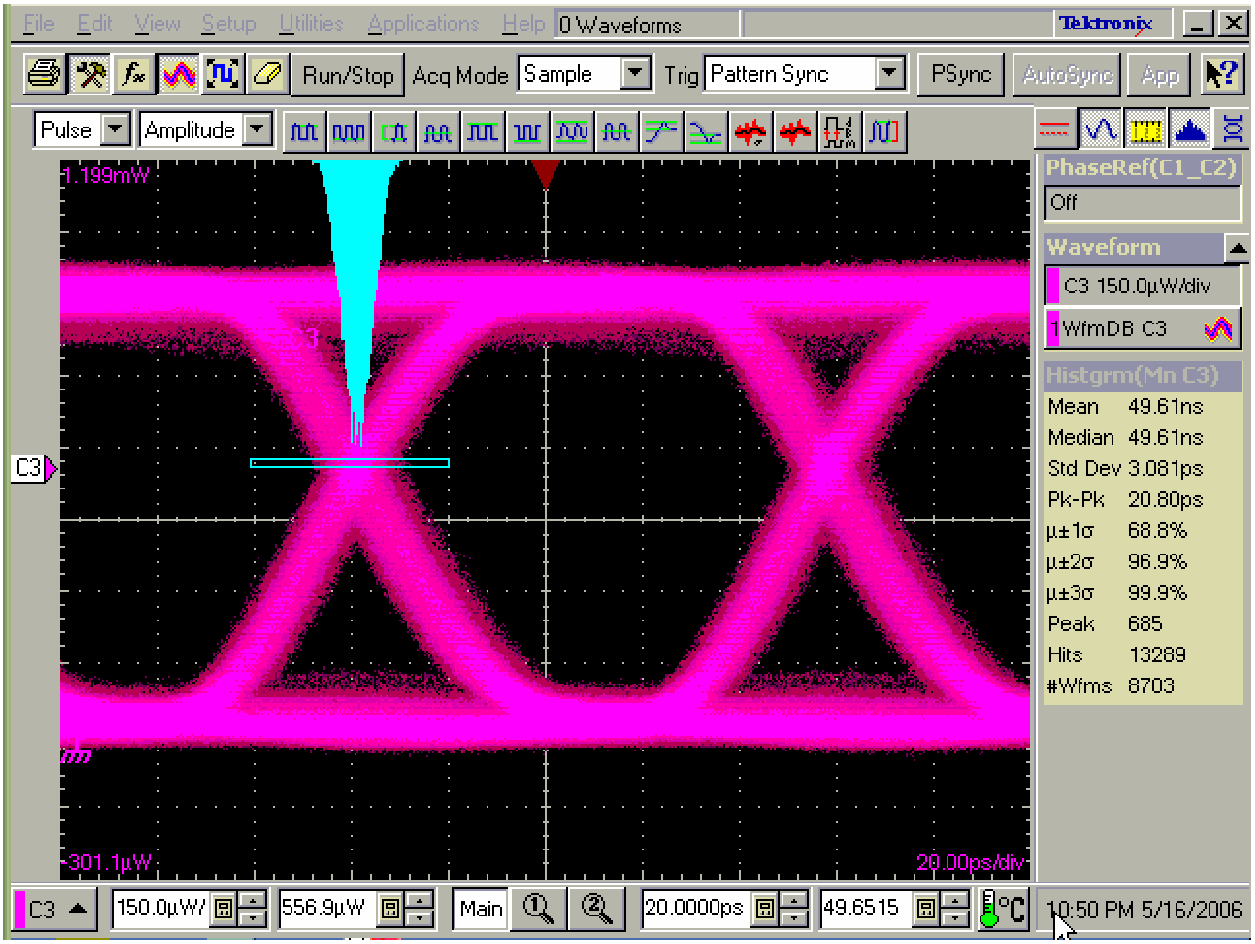


Figure 1: An eye diagram showing a measurement of the jitter probability density function as a histogram projected from the crossing point.

Assuring interoperability and performing diagnostics are not independent. Measurements of different types of jitter can be used to make accurate estimates of TJ(BER) – *estimates* that, in many cases, are more accurate than *measurements*! In this paper, we concentrate on TJ(BER) – what it is, why it is useful, and how it is measured and estimated. We'll cover diagnostics in another paper.

One point that you should keep in mind is that jitter analysis, by its very nature, is a simplification. Analyzing jitter independent of voltage noise can be useful, but no thorough signal integrity analysis neglects the relationship of the two.

In Search of Peak-to-Peak Jitter

We need a quantity that manufacturers of different network elements can use to judge whether or not their parts can operate in a system to a specified BER. The naïve solution is simply to measure the peak-to-peak jitter. The problem with peak-to-peak jitter is that the value you measure depends on how long you make the measurement. It's even worse than that, if we all agree to measure peak-to-peak jitter for, say, 100 seconds, we'll still get different results. It is in this sense that peak-to-peak jitter is ill-defined.

To find a quantity that both links jitter to the BER and is well defined, it's useful to distinguish random jitter from deterministic jitter. Deterministic Jitter (DJ, from here on) is caused by things like impedance mismatches and electromagnetic interference. If we knew everything about the circuit, the effect on the signal from such causes could be determined. DJ is also limited in its effect. That is, DJ can only shift the timing of a logic transition by some maximum amount. It is in this sense that DJ is bounded and, therefore, has a well defined peak-to-peak spread which we'll call DJ(p-p).

Random Jitter (RJ), as the name implies, is caused by things like thermal oscillations, flicker, and shot noise that result in levels of jitter that cannot be predicted on an edge-by-edge basis. The thing to understand about RJ is that it is caused by a huge number of very small effects. The variation in the width of the traces on a printed circuit board, fluctuations of conductivity in a conductor caused by impurities, variations in resistance caused by random fluctuations in the local voltage that individual electrons feel, and literally zillions of other tiny effects combine in a way that is dictated by the fundamental theorem of Statistical Physics: The Central Limit Theorem, which makes the whole thing very simple.

The Central Limit Theorem says that a large number of small independent processes combine in a way that follows a Gaussian distribution. This means that the variation in the timing of a specific logic transition from the ideal time is centered at zero, can occur arbitrarily early or late, but nearly always occurs within a few units of sigma from the ideal. Sigma, or σ, is the width of the Gaussian distribution. It's not the full-width-at-half-max, but it's the same idea and it's the only thing we need to know about the distribution; σ is also the rms value of the RJ distribution and gives a complete description of RJ. If we define x as the time-delay relative to the ideal time of a logic transition, then the RJ probability density function is given by

$$G(x) = N \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{1}$$

Don't worry about the constant, *N*, in front of the exponential in Eq. (1), it's just there to force the integral of the distribution to one. As you can see in Figure 2, *G*(*x*) is unbounded. The probability of RJ shifting a logic transition by an amount *x*, is proportional to *G*(*x*), the height of the curve at *x*.

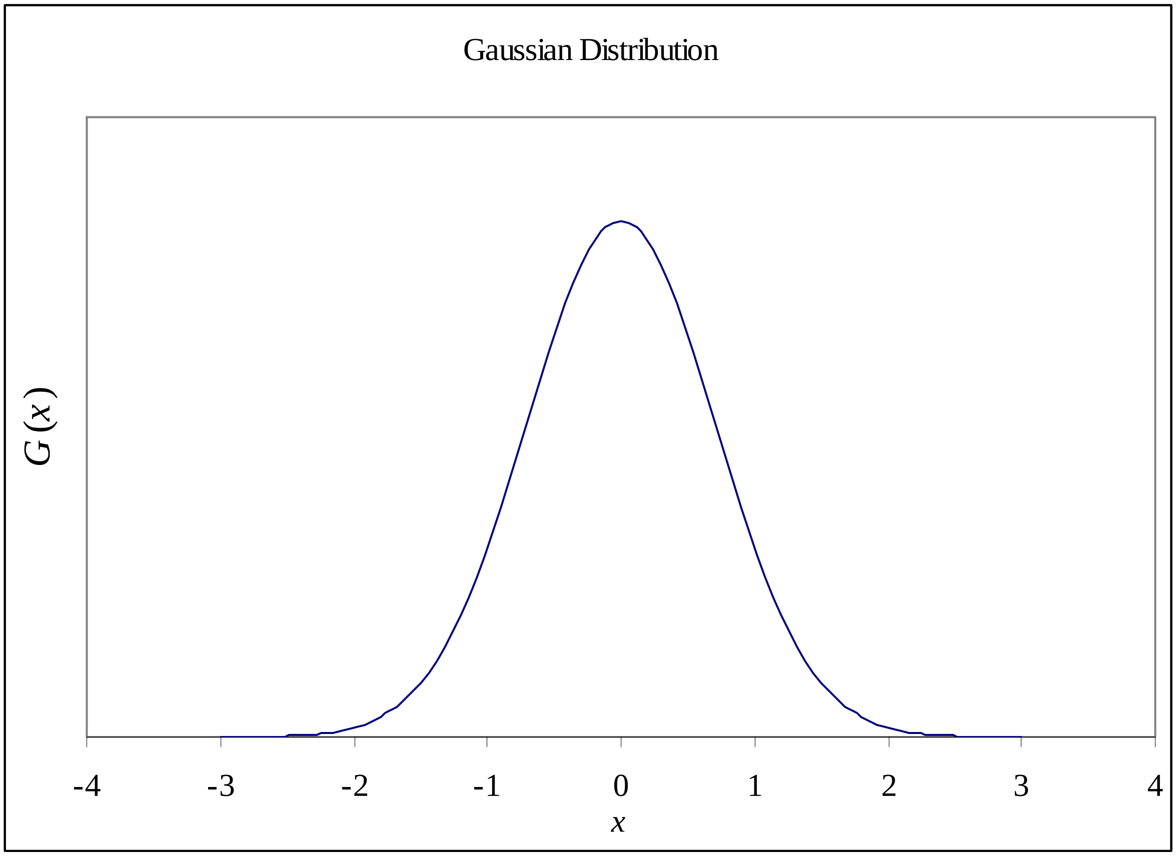


Figure 2: A Gaussian, or normal, distribution. The probability of RJ causing a logic transition to be jittered in time by an amount *x* is proportional to the height of the curve at *x*, *G*(*x*).

TJ(BER) – Total Jitter at a Bit Error Ratio

Notice that, since G(*x*) never actually makes it to zero, it is possible for RJ to cause a logic transition to happen ten years from when it's supposed to. In the same sense, the laws of thermodynamics allow for the possibility that all of the air molecules in the room where you are sitting can spontaneously condense in a corner, leaving you in a vacuum where your eyes can pop out – it's possible, but so unlikely as to be irrelevant. But this possibility alludes to the reason that peak-to-peak jitter measurements get larger the longer that they are measured.

Because RJ is unbounded, peak-to-peak jitter is ill-defined.

But even if peak-to-peak jitter were well defined, it wouldn't do us much good. Remember, the only reason we care about jitter is if it causes errors. What we really need is something like a peak-to-peak jitter measurement that tells us about the amount of eye-closure. If we used an oscilloscope to measure peak-to-peak jitter and waited long enough, eventually the eye would close, that is, the peak-to-peak jitter would be the full bit period.

Think of a signal whose BER, when measured on an ideal receiver, is 10^{-12}. The peak-to-peak jitter measurement would be the bit period, indicating total eye closure, after something like 10^{12} bits were transmitted. It's not completely well defined because random fluctuations would still vary the amount of eye closure even after 10^{12} bits, but now at least the peak-to-peak jitter is explicitly correlated to the BER. It is in this sense that Total Jitter defined at a Bit Error Ratio, TJ(BER), is like a peak-to-peak jitter measurement.

TJ(BER) relates the level of jitter to the bit error ratio of interest: it is the amount of eye closure at a given BER. In the example above, the simple peak-to-peak jitter measurement gave total eye closure after 10^{12} bits. A measurement of TJ(10^{-12}) would reliably give TJ(10^{-12}) = T_B. But what if the actual BER were 10^{-18} and the system were specified to 10^{-12}? In this case, after 10^{12} bits were transmitted the eye would still be open, TJ(10^{-12}) would give us the amount of eye closure at a BER of 10^{-12} for an eye that has a BER of 10^{-18}.

Figure 3a shows an eye diagram and Figure 3b shows the "bathtub plot" that corresponds to it. A bathtub plot is a measurement of the BER as a function of the time-delay position of the sampling point – BER(x). It can be measured by stepping the sampling point horizontally across an eye diagram and measuring the BER at each point. At the edges, BER is very high and, as the sampling point is stepped to the eye-center, the BER drops precipitously until the sampling point approaches the other edge, when the BER increases again.

The *eye opening* at a BER is the distance between the two sloping curves in Figure 3b at the desired BER. TJ(BER) is the *eye closure* at the desired BER: the nominal bit period, T_B, less the eye opening at that BER.

$$\mathrm{TJ(BER)} = T_B - \mathrm{eye\,opening(BER)} \tag{2}$$

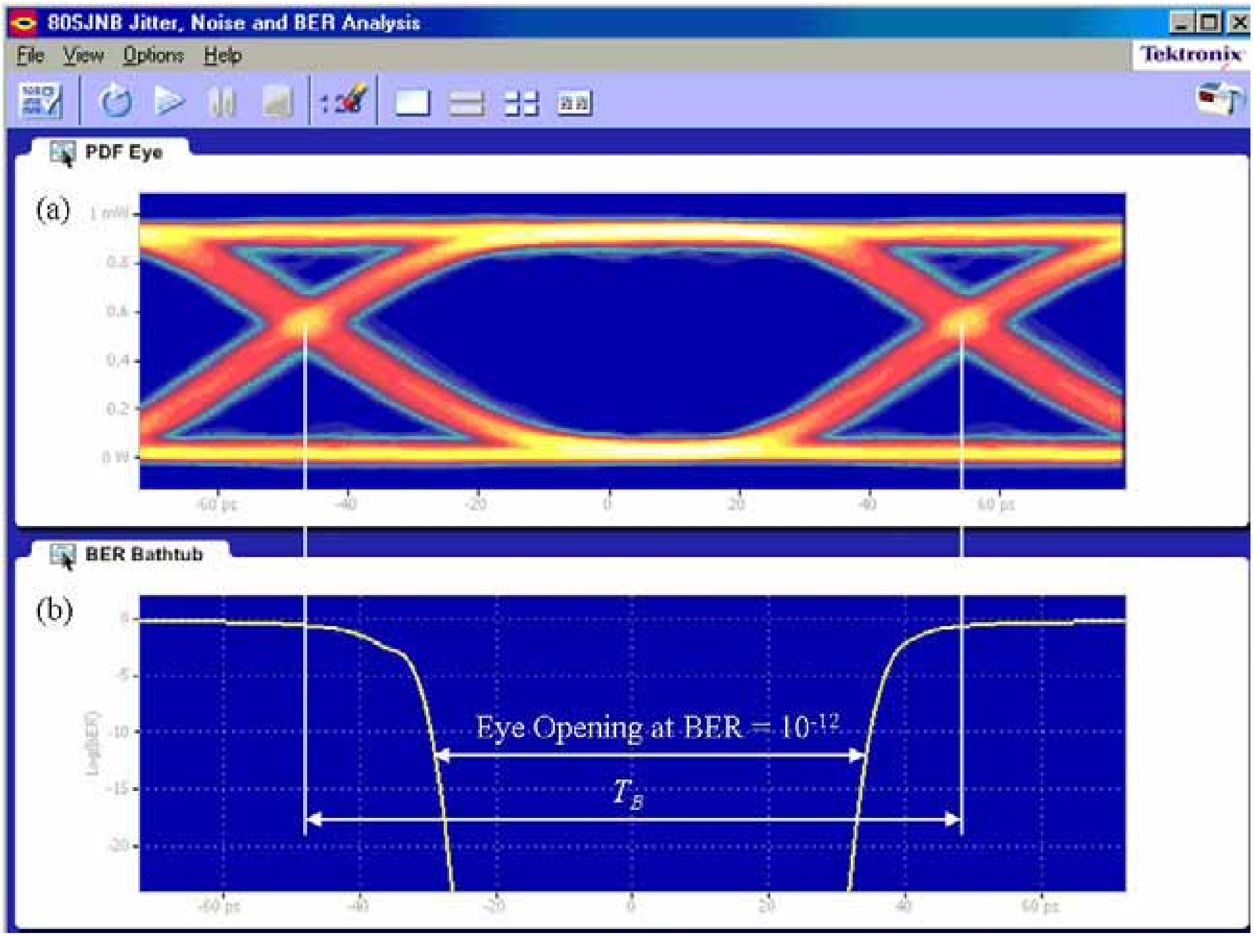


Figure 3: (a) An eye diagram and, (b), the corresponding "bathtub-plot": The bit error ratio as a function of the time-position of the sampling point with respect to the ideal logic transition time, BER(*x*).

Now consider Figure 4. The BER of this system is exactly 10^{-12}. You can tell because the eye is completely closed where the two slopes meet. This also means that TJ(10^{-12}) = T_B, and, for that matter, TJ(10^{-13}) = TJ(10^{-30}) = T_B, because the eye can only close completely.

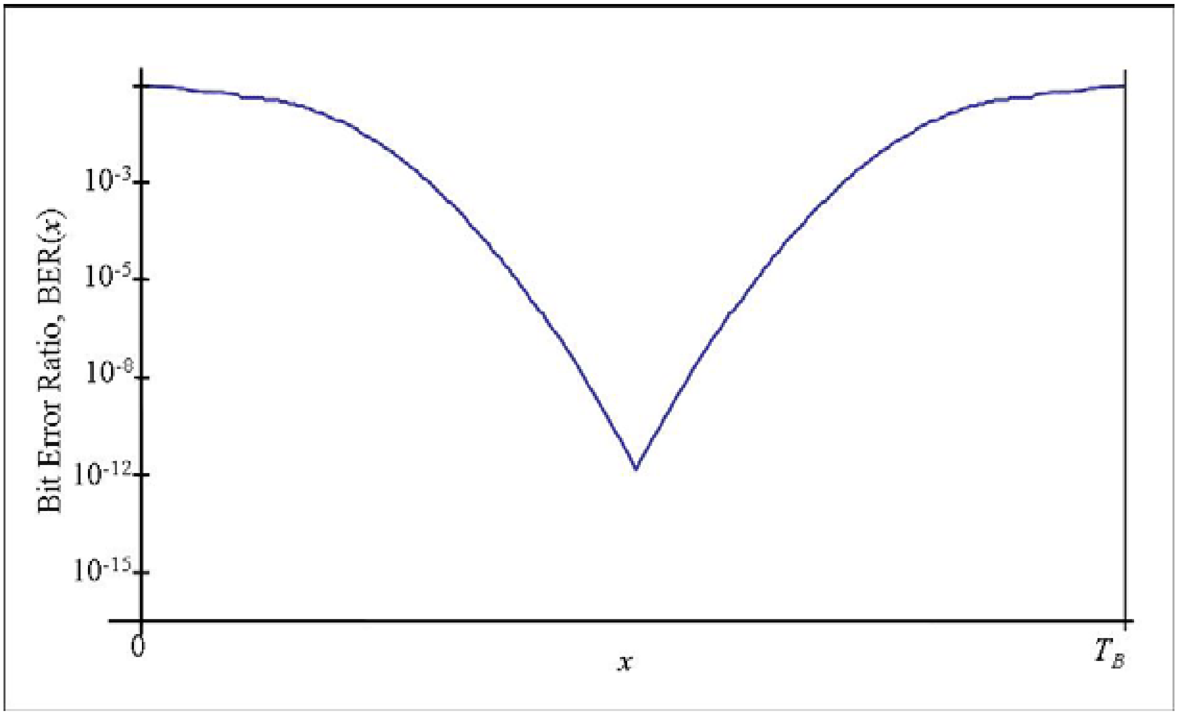


Figure 4: The bathtub plot of an eye that is closed at BER = 10^{-12}.

Strictly speaking the only practical way to *measure* TJ(BER) at BERs lower than about 10^{-8} is with a Bit Error Ratio Tester. A BERT can measure TJ(BER) in two ways. It can do the brute force measurement, as in Figure 3 and Figure 4, of BER(*x*) to get TJ(BER), but measuring BER(*x*), to low BERs like 10^{-12} takes a long time. It's the sort of measurement that you might start in the evening and expect to be finished the next morning. TJ(BER) can also be measured on a BERT by using a bracketing technique which cuts the measurement time down to about half an hour for a 5 Gb/s signal.

The reason that TJ(BER) measurements take so long is simple. In principle one cannot measure the rate of occurrence of an event without probing the system with a statistical sample sufficient to observe the event. Thus, measurements of TJ(BER) require the detection of at least 6/BER bits (6×10^{12} bits for BER = 10^{-12} at 5 Gb/s would take 20 minutes) bits for the bracketing technique and about 100/BER bits (10^{14} bits for BER = 10^{-12} at 5 Gb/s would take five and a half hours) for an accurate brute force measurement of BER(*x*) to get sufficient statistics to reference the eye closure to the BER. In an industry where Moore's Law rules, it's easy to understand why such a great effort has gone into finding techniques that accurately *estimate* TJ(BER) from a much smaller data sample.

The largest uncertainty in a BERT measurement of TJ(BER) is introduced by the error detector sensitivity. Without going into a lot of BERT detail, "the error detector sensitivity" is the smallest difference between logic levels a BERT can distinguish, typically around 50 mV – much larger than the voltage resolution and accuracy of either a real-time or equivalent-time oscilloscope. The time-base of BERTs manufactured in the last few years are quite accurate and contribute less uncertainty than does the sensitivity; for BERTs manufactured more than five years ago, timing nonlinearities and time-base resolution make TJ(BER) measurements very difficult.

The point is that TJ(BER) can only be *measured* on a BERT, but if it could be measured on an oscilloscope, the measurement would be more accurate.

RJ and DJ

There are two ways to get TJ(BER) without having to wait as long as is required to accumulate 6/BER bits. First, if each component of the jitter distribution can be separately measured, including RJ and all types of DJ, then they can be convolved into the jitter PDF from which BER(x) can be calculated for any x. Given BER(x), it's easy to extract TJ(BER) as in Figure 3b and Eq. (2). A simple measurement of the Jitter PDF, like the histogram in Figure 1, doesn't have enough data to build a PDF that is accurate enough to extract BER(x). Further, in complicated cases, even with the best jitter analysis techniques, it's not possible to distinguish every DJ source and measure the correlations between the different DJ sources as is necessary to build the complete PDF.

The other way to get TJ(BER) is to use the dual-Dirac model. The dual-Dirac model provides an easy way to estimate TJ(BER) by measuring the RJ σ and a quantity much like the peak-to-peak spread of DJ, DJ(p-p).

$$\mathrm{TJ(BER)} = 2Q_{BER} \times \mathrm{RJ} + \mathrm{DJ}(\delta\delta) \tag{3}$$

Here, Q_{BER} is a factor that relates the BER to the distance from the center of the RJ Gaussian, listed in Table 1 for different BERs, and DJ($\delta\delta$) is a model dependent parameter that is easy to define and can be measured in many different ways.

BER	Q_{BER}
10^{-10}	6.35
10^{-11}	6.70

10^{-12}	7.05
10^{-13}	7.35
10^{-14}	7.65

Table 1: Values of Q_{BER} for different BERs.

We will cover the details of the dual-Dirac model in another paper. In a nutshell, the dual-Dirac model uses the fact that, for a signal with only RJ, BER(x) can be calculated from Eq. (1) and assumes that the tails of almost all jitter distributions follow the RJ Gaussian distribution, Figure 2, shifted toward the center of the eye.

Whether we calculate TJ(BER) by measuring RJ, all sources of DJ, convolving them together and integrating the resulting PDF, or by measuring RJ and DJ($\delta\delta$) and using Eq. (3) we are making the same fundamental assumption: that there are no rare processes whose observation would require a huge statistical sample. Integral to this assumption is the idea that the RJ Gaussian accurately represents the tails of the jitter PDF. If it doesn't, then the estimates of TJ(BER) will be biased. At a deeper level, there is also some question about whether or not what we consider RJ is actually random. The Central Limit Theorem indicates that if there are enough small DJ effects, their distribution will mimic a Gaussian – a truncated or "bounded" Gaussian. In practice, these are not problems that test engineers need to worry about; they are problems that justifiably prevent the designers of technology standard specifications from sleeping at night. We can never be absolutely be certain that the tails of the jitter PDF follow the RJ Gaussian without acquiring that 100/BER statistical sample, but, in the vast majority of cases, careful eye-diagram analysis, thorough study of jitter diagnostic information, and experience with your design provide clues of when your TJ(BER) estimates are being corrupted.

In most cases TJ(BER) can be estimated very quickly and accurately on high-fidelity, low noise oscilloscopes, such as the Tektronix CSA8200 equivalent-time sampling oscilloscope with 80SJNB jitter and noise analysis software and the Tektronix TDS6000 series real-time oscilloscope with TDSJIT3 jitter analysis software, that also provide a great deal of diagnostic information.

Conclusion

The main points of this paper are:

1. TJ(BER) is an abstract form of peak-to-peak jitter that is referenced to a specific bit error ratio.
2. TJ(BER) is useful for assuring interoperability, but not so useful for diagnosing jitter problems.

3. While TJ(BER) can only be *measured* on a bit error ratio tester, TJ(BER) calculated from measurements of the jitter probability density function or the dual-Dirac model, Eq. (3), are frequently more accurate.

It is important that you also be aware of a few things that were *not* discussed in this paper. First, look at Eq. (2), the bit period, T_B, in a serial data system is much more complicated than just the reciprocal of the data rate. In jitter analysis all we care about is the jitter that can cause errors, so the clock that supplies "T_B" in any of the analyses discussed here, can have a variety of different frequency characteristics.

We also didn't go into any detail about the different sources of jitter, where they come from, how they are correlated with and can effect the signal-to-noise ratio, and how they need to be analyzed – we didn't even mention that crosstalk can destroy any of these measurements. Nor did we discuss the details of the dual-Dirac model and why it's so useful in compliance specifications. Each of these topics is the subject of another short paper.

What the Dual-Dirac Model is and What it is Not

Ransom Stephens

October, 2006

Abstract:

The dual-Dirac model is a simple tool for estimating total jitter defined at a bit error ratio, TJ(BER), for serial data components and systems. By virtue of the fact that it can be used to combine the RJ and DJ of different elements to predict the TJ(BER) for a system, the dual-Dirac model is a key tool in specifications for serial data links. In this paper, we present the dual-Dirac model, how and why it is used in specifications, how it is used to estimate the Total Jitter of a system, the assumptions it makes and where they fail.

The dual-Dirac model was introduced several years ago as a tool for quickly estimating the total jitter defined at a bit error ratio, TJ(BER) [for the definition of TJ(BER), see Part 1 in this series, *The Meaning of Total Jitter*]. Since its introduction, the merits of the dual-Dirac model have been controversial. The controversy is almost always caused by misunderstandings in the meaning of the model parameters. The dual-Dirac model uses two parameters that, in most of the literature, are called RJ and DJ but, unless the model assumptions are strictly valid, these parameters are model dependent. To cut out that confusion, let's distinguish the model dependent parameters by calling them RJ($\delta\delta$) and DJ($\delta\delta$). As we'll see, if these model-dependent parameters are properly measured, then the relationship

$$\mathrm{TJ(BER)} = 2Q_{BER}\times\mathrm{RJ}(\delta\delta) + \mathrm{DJ}(\delta\delta) \qquad (1)$$

is model-*independent*. That is, TJ(BER) calculated from Eq. (1) is as accurate as the measurements of RJ($\delta\delta$) and DJ($\delta\delta$); any inaccuracy in TJ(BER) is due to the measurement, not the model. But I'm getting ahead of myself, let's go back and start at the beginning.

The dual-Dirac model

Since the following is a discussion of a mathematical model, the formalism is somewhat deep. The main reason we present it here is to show where Eq. (1) comes from and provide a rigorous foundation for the conclusions drawn in the subsequent section. If you're not intrigued by the formalism, then skim through this section, pay attention to the graphics, and you'll be well prepared for the following section where the important points of how to use the dual-Dirac model are made.

First, the Dirac-delta function, $\delta(x - x_0)$, is conveniently defined so that it is zero everywhere but at $x = x_0$, where it's infinite, but infinite in such a way that its integral is one:

$$\delta(x - x_0) \equiv \begin{cases} 0, & x \neq x_0 \\ \to \infty, & x = x_0 \end{cases} \quad \text{with} \quad \int \delta(x - x_0)dx = 1. \tag{2}$$

That is, $\delta(x - x_0)$ is a spike centered at $x = x_0$.

Second, as we know, the Probability Density Function (PDF) for RJ is a Gaussian,

$$\mathrm{PDF}_{RJ}(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{x^2}{2\sigma^2}\right]. \tag{3}$$

Third, the distributions of different components of jitter that are independent, like RJ and DJ, combine through convolution,

$$\begin{aligned} \mathrm{PDF}(x) &= \mathrm{PDF}_{DJ}(x) * \mathrm{PDF}_{RJ}(x) \\ &= \int \mathrm{PDF}_{DJ}(u) * \mathrm{PDF}_{RJ}(x-u)du \end{aligned} \tag{4}$$

Figure 1 shows the pieces of the dual-Dirac model. The dual-Dirac DJ PDF is simply the sum of two Dirac delta functions, one centered at μ_L and one at μ_R,

$$\mathrm{PDF}_{dual\text{-}Direc\ DJ}(x) = \delta(x - \mu_L) + \delta(x - \mu_R). \tag{5}$$

The peak-to-peak DJ of this distribution is simply the separation of the two Dirac deltas, $\mu_R - \mu_L$. Now introduce Gaussian RJ, Eq. (3) and combine it with the dual-Dirac DJ, Eq. (5), using the convolution, Eq. (4), to get the complete dual-Dirac PDF,

$$
\begin{aligned}
\mathrm{PDF}_{dual-Dirac}(x) = \mathrm{RJ} * \mathrm{DJ} &= \frac{1}{\sqrt{2\pi}\sigma}\int\left[\delta(x'-\mu_L)+\delta(x'-\mu_R)\right]\exp\left(-\frac{(x-x')^2}{2\sigma^2}\right)dx' \\
&= \frac{1}{\sqrt{2\pi}\sigma}\left[\exp\left(-\frac{(x'-\mu_L)^2}{2\sigma^2}\right)+\exp\left(-\frac{(x'-\mu_R)^2}{2\sigma^2}\right)\right]
\end{aligned}
\tag{6}
$$

two Gaussians, each displaced from the origin.

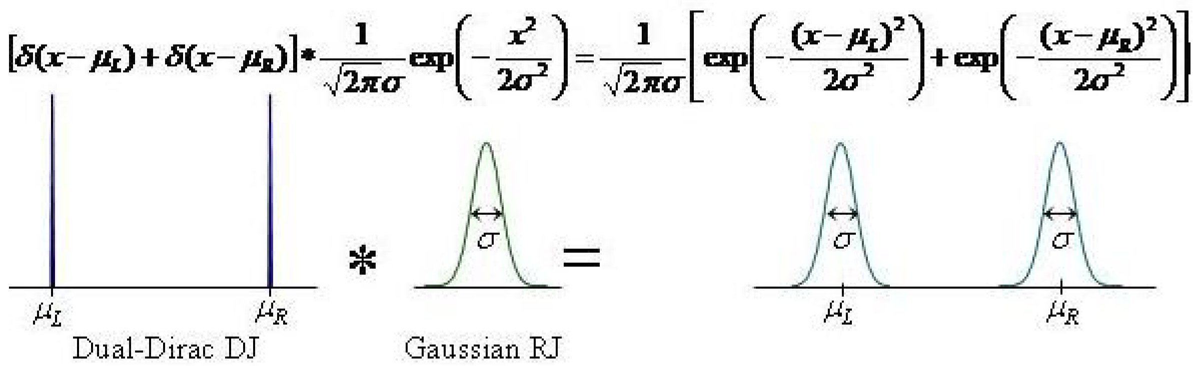


Figure 1: The dual-Dirac jitter distribution. In (a) the DJ and RJ distributions and, in (b), their convolution.

Another way to think of the dual-Dirac model that may be more intuitive is to see how it evolves in an eye diagram. Start with no jitter in Figure 2a and introduce only the dual-Dirac DJ in Figure 2b. Notice that there are two distinct logic-transition trajectories, one has its crossing point at μ_L and the other at μ_R. When RJ is introduces in Figure 2c, the two trajectories are smeared according to the Gaussian. The dual-Dirac jitter distribution is shown by the histogram in the upper left corner of Figure 2c. In practice, the dual-Dirac DJ distribution can be realized by square-wave phase modulation – hardly a realistic scenario.

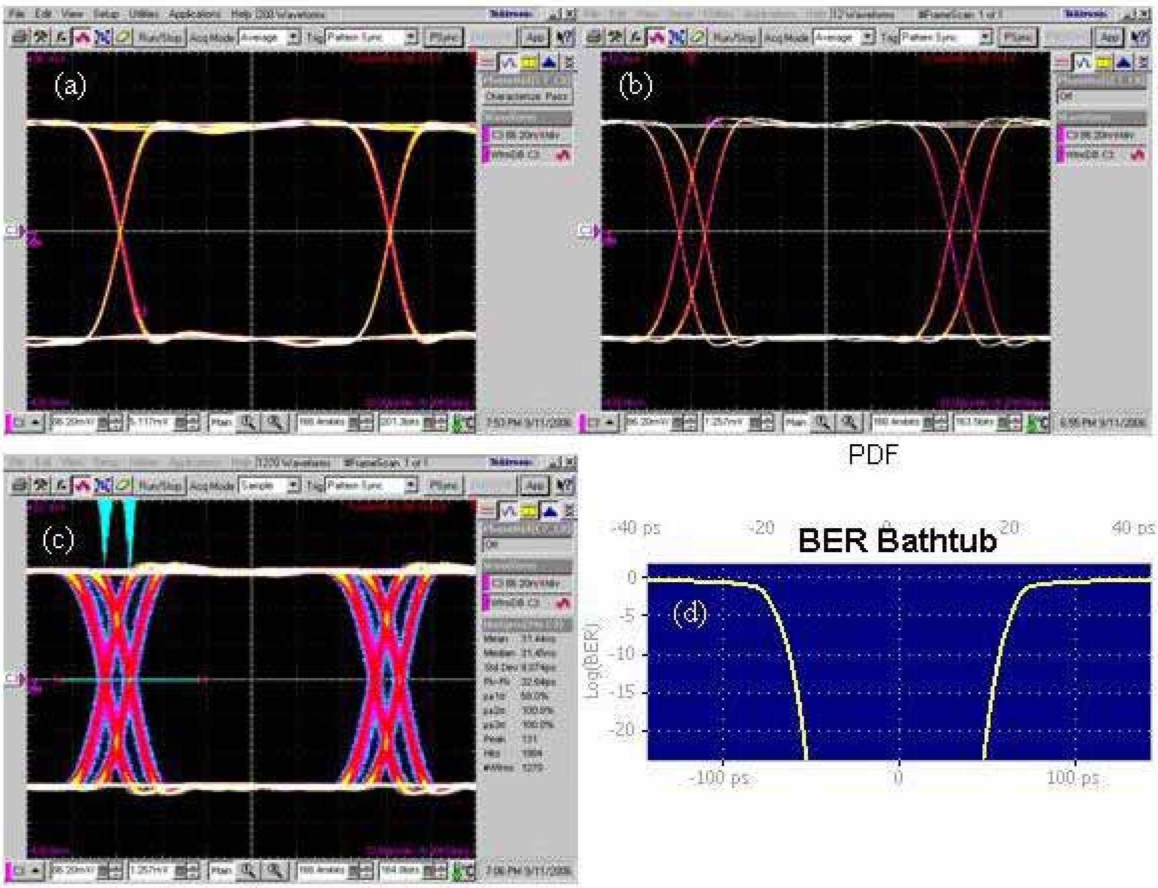


Figure 2: An eye diagram with, (a) no jitter, (b) dual-Dirac DJ, (c) RJ and dual-Dirac DJ, and (d) bathtub plot, BER(x).

Once we have the jitter PDF, the bathtub plot shown in Figure 2d can be calculated. BER(x) is the probability that an error will occur if the sampling point is positioned at the time-delay position, x. In general, BER(x) is given by

$$\text{BER}(x) = \rho_T \int_x^{\infty} \text{PDF}(x')dx' + \rho_T \int_{-\infty}^{x} \text{PDF}(x'-T)dx' \qquad (7)$$

where ρ_T is the logic transition density (i.e., the ratio of the number of transitions to the number of bits). The first term on the right of Eq. (7) accounts for fluctuations across the sampling point from left to right and the second term accounts for fluctuations across the sampling point from right to left.

Plug the dual-Dirac model, Eq. (6) into Eq. (7), and get

$$\mathrm{BER}_{\delta\delta}(x) = \rho_T \left[\mathrm{erfc}\left(\frac{x-\mu_L}{\sqrt{2}\sigma} \right) + \mathrm{erfc}\left(\frac{(x-T)-\mu_R}{\sqrt{2}\sigma} \right) \right] \tag{8}$$

where erfc(x) is the "complementary error function." Evaluate the complementary error functions, do a bunch of algebra, and get Eq. (1),

$$\mathrm{TJ(BER)} = 2Q_{BER}\times\mathrm{RJ}(\delta\delta) + \mathrm{DJ}(\delta\delta) \tag{1}$$

where Q_{BER} is a constant that can be calculated from the complementary error function and is tabulated for a few values of BER in Table 1. RJ is given by σ and DJ is given by the separation of the two Dirac-delta functions, $\mu_R - \mu_L$. That is,

$$\mathrm{RJ}(\delta\delta) = \sigma \text{ and } \mathrm{DJ}(\delta\delta) = \mu_R - \mu_L. \tag{9}$$

Equation (1) with Eq. (9) is a nice result, but what good does it do with a real-life jitter distribution like the one in Figure 3?

BER	Q_{BER}
10^{-10}	6.3
10^{-11}	6.7
10^{-12}	7.0
10^{-13}	7.4
10^{-14}	7.7

Table 1: Values of Q_{BER} for different BERs.

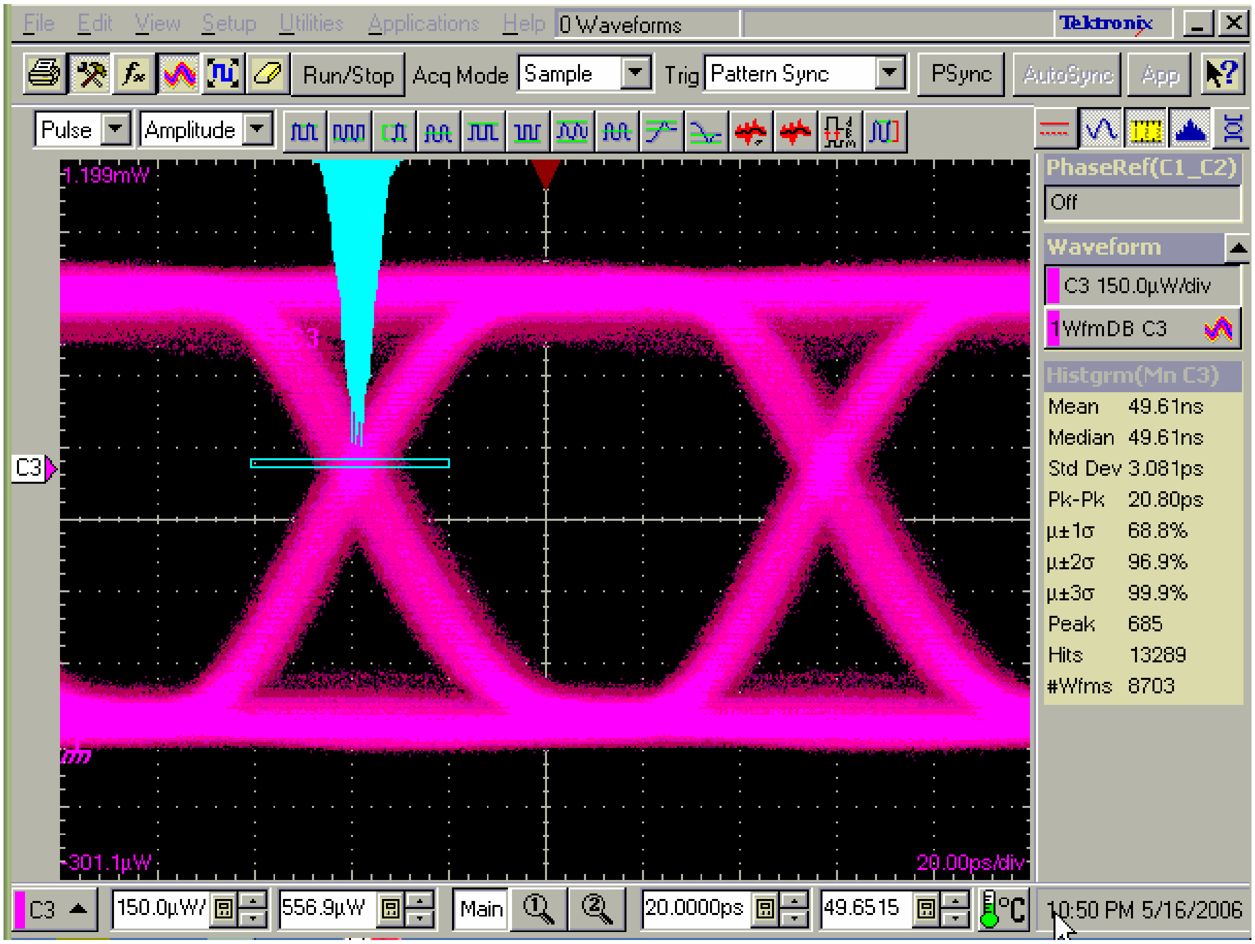


Figure 3: A more realistic eye-diagram.

What the dual-Dirac model is and what it is not

Since the calculation of TJ(BER) depends only on the tails of the jitter distribution, i.e., from "x to ∞" and "–∞ to x" as in Eq. (7), we can calculate TJ(BER) using Eq. (1) for *any* jitter distribution as long as:

(1) we can neglect amplitude noise, and

(2) the tails of the jitter distribution are dominated by RJ.

These are the two key assumptions of the dual-Dirac model.

Here's another way of saying it:

As long as the tails of the distribution follow the RJ Gaussian at the BER we care about, TJ(BER) is given by Eq. (1).

The dual-Dirac model is just a convenient way of lining the Gaussian RJ distribution up with the data. That's all it is.

The trick to using the dual-Dirac model, is remembering that its two parameters, RJ($\delta\delta$) and DJ($\delta\delta$), are in general model-dependent and have to be measured from the data accordingly. Fortunately, RJ($\delta\delta$) and DJ($\delta\delta$) can be measured in many different ways for any jitter distribution for which the two assumptions above are valid.

Let's start with RJ($\delta\delta$) because it's much less model-dependent than DJ($\delta\delta$). As long as the rising and falling slew rates are the same, the dual-Dirac RJ is exactly the model-independent RJ given by the width, or rms, of the RJ Gaussian, σ. On the other hand, if the edges are not symmetric, then RJ($\delta\delta$) is defined as the average of the Gaussian tails of the left and right widths:

Symmetric edges, RJ is model-independent: RJ($\delta\delta$) = RJ $\equiv \sigma$ (10)

Asymmetric edges, RJ is model-dependent: RJ($\delta\delta$) $\equiv$ ½ ($\sigma_L + \sigma_R$) (11)

In the vast majority of systems the difference between Eq. (10) and (11) is smaller than the uncertainty in the measurements of σ_L and σ_R and we can safely assume RJ($\delta\delta$) = RJ = σ.

By its nature the RJ of one element is statistically independent of the RJ of another element. That is, the RJ of one element doesn't interfere with the RJ of another. Independence means that the RJ distribution of two network elements is given by the convolution of the two RJ distributions. The convolution of two Gaussians is a Gaussian whose width is the square-root of the sum of the squares of the individual widths. Thus, the RJ for a system of N elements, is

$$\sigma_{system} = \sqrt{\sigma_1^2 + \sigma_2^2 + \ldots + \sigma_N^2}\,. \tag{12}$$

DJ($\delta\delta$) is more complicated. Where the model dependence of RJ is almost always negligible, the model dependence of DJ($\delta\delta$) is rarely negligible.

The only case where the actual peak-to-peak DJ, lets call it DJ(p-p), is the same as the dual-Dirac DJ, DJ($\delta\delta$), is when the DJ distribution really is given by the sum of two Dirac-delta functions, for example, when the phase is modulated by a square-wave. Figure 4 shows how DJ(p-p) differs from DJ($\delta\delta$). By their bounded nature, DJ distributions tend to have sharp edges and, by its Gaussian nature, RJ has long smooth tails. The process of convolving DJ with Gaussian RJ is a process of smoothing the edges of the DJ distribution. As the edges of the DJ distribution are sanded down, they appear to be pulled inward, closer to the nominal crossing point. The result is a general statement about the model-dependence of DJ,

$$\mathrm{DJ}(\delta\delta) \le \mathrm{DJ}(\text{p-p}). \tag{13}$$

The result, Eq. (13), is the reason that the dual-Dirac model suffers so much unwarranted criticism. It's a model. It's okay for a model to have model-dependent parameters as long as the model-dependence is included when they're measured. In other words, to use Eq. (1), we need to measure DJ($\delta\delta$), not DJ(p-p) – which in some ways is good news. It's easier to measure DJ($\delta\delta$) than it is to measure DJ(p-p). An accurate measurement of DJ(p-p) requires a complete understanding of *all* sources of DJ, including their relative phases, which is difficult to measure accurately on most equipment. Accurate measurement of DJ($\delta\delta$) only requires that the data sample is large enough to have sampled all DJ processes.

The main point is that, from the compliance point of view, DJ($\delta\delta$) is more useful than DJ(p-p). The DJ($\delta\delta$) of different components can be combined to estimate the DJ($\delta\delta$) of a system, but the DJ(p-p) cannot.

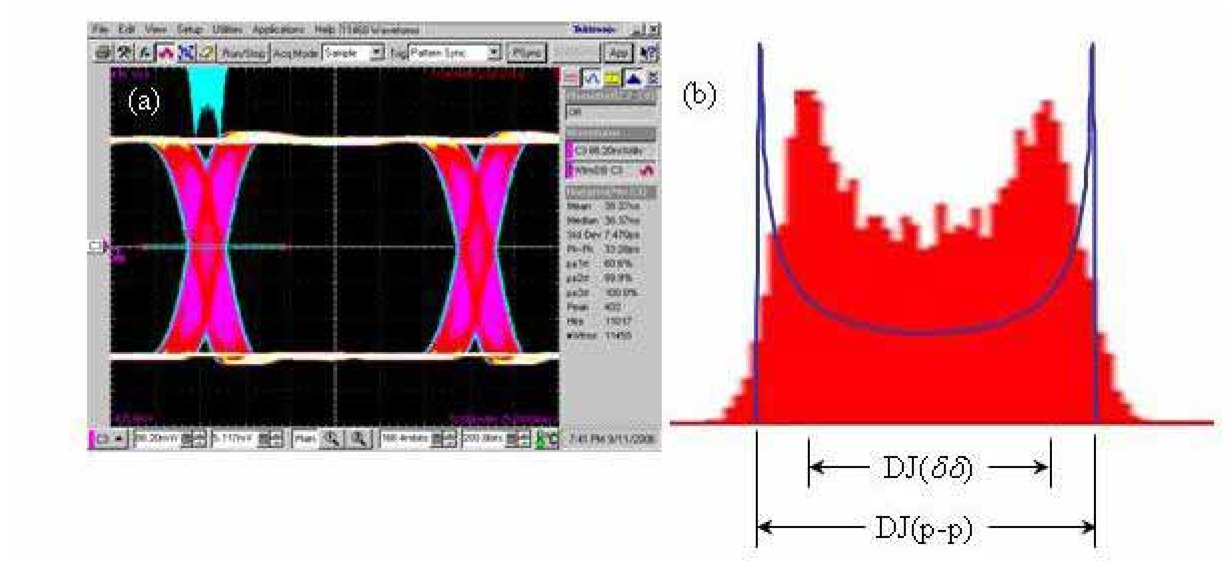


Figure 4: (a) An eye diagram and jitter histogram for a signal with sinusoidal DJ and RJ, (b), an exploded view of the jitter histogram with the underlying sinusoidal DJ PDF overlaid. Notice how the sharp edges of the DJ distribution are pulled inward when RJ is introduced. In both cases, the actual peak-to-peak DJ, DJ(p-p), and the model-dependent dual-Dirac DJ, DJ($\delta\delta$), are shown, demonstrating Eq. (13).

If the PDF of one jitter source changes when the PDF of another source is changed, then those two sources are *dependent* or *correlated*. The peak-to-peak value of the convolution of two *independent* distributions is the sum of the peak-to-peak values of each distribution. On the other hand, the peak-to-peak value of the convolution of two *correlated* distributions is less than or equal to the sum of the peak-to-peak values of each distribution. Generalizing to a system of N components, we can say

$$DJ_{system}(\text{p-p}) \leq DJ_1(\text{p-p}) + DJ_2(\text{p-p}) + \ldots + DJ_N(\text{p-p}). \quad (14)$$

The problem is that the DJ of different system elements is almost always correlated. For example, the ISI of a transmitted signal affects the ISI introduced by the transmission channel. If we used the sum of the component's DJ(p-p) to estimate the DJ of the system we'd have two problems. First, we'd always overestimate the actual DJ(p-p) and, second, DJ(p-p) isn't the right parameter to use in Eq. (1) for estimating TJ(BER), we need DJ($\delta\delta$) for that.

We need a simple expression for combining the DJ($\delta\delta$) of different components for use, along with the combination of component RJ of Eq. (12), in Eq. (1). The obvious candidate is

$$DJ_{system}(\delta\delta) \approx DJ_1(\delta\delta) + DJ_2(\delta\delta) + \ldots + DJ_N(\delta\delta). \tag{15}$$

Equation (15) is an approximation, and should be treated with appropriate suspicion, though it's usually a pretty good approximation. Think of it like this, as more DJ sources are introduced, the resulting DJ distribution gets smoother around the edges. The smoother the distribution, the greater the discrepancy between DJ($\delta\delta$) and DJ(p-p). Now, since the maximum possible DJ(p-p) is given by the sum of the component DJ(p-p) values, Eq. (14), and since DJ($\delta\delta$) is generally smaller than DJ(p-p), it's reasonable to expect that $DJ_{system}(\delta\delta)$ should be smaller than that given by the sum, Eq. (15). We might then expect that Eq. (15) is a conservative estimate for the system DJ. That's the argument, anyway.

It's possible to get an idea of the accuracy of Eq. (15) by analyzing the different DJ components, a diagnostic technique that's the subject of another paper in this series, "All About the Acronyms: RJ, DJ, DDJ, ISI, DCD, PJ, SJ,…"

Equations (12) and (15) along with (1), rewritten here,

$$\sigma_{system} = \sqrt{\sigma_1^2 + \sigma_2^2 + \ldots + \sigma_N^2}\,. \tag{12}$$

$$DJ_{system}(\delta\delta) \approx DJ_1(\delta\delta) + DJ_2(\delta\delta) + \ldots + DJ_N(\delta\delta). \tag{15}$$

$$TJ(BER) = 2Q_{BER}\times\sigma + DJ(\delta\delta) \tag{1}$$

provide the tools we need to estimate the TJ(BER) of a system from the RJ and DJ($\delta\delta$) of its components. It is for this reason that when DJ is quoted in the standards specification for a given technology like FibreChannel, PCI-Express, FBD, SATA, et cetera, it is DJ($\delta\delta$), not DJ(p-p) that is relevant.

Conclusion

The dual-Dirac model is useful both for estimating TJ(BER) and combining the RJ and DJ of separate components to estimate the TJ(BER) of a system – provided the model dependent parameters, DJ($\delta\delta$) and RJ($\delta\delta$) are used. The difference between the model-dependent RJ, RJ($\delta\delta$), and the true RJ, σ, is almost always negligible and it's perfectly safe to assume RJ = σ. On the other hand, the model-dependent DJ, DJ($\delta\delta$), is different than the true peak-to-peak DJ, DJ(p-p). At least in the context of standards DJ($\delta\delta$) is more useful than DJ(p-p).

But it's still a model. It still rests on assumptions that can be debated. For example, most techniques for measuring RJ run into serious problems in environments that include crosstalk and other forms of "bounded uncorrelated jitter." Another potential problem is that If RJ doesn't follow a Gaussian distribution then the very foundation of the model collapses or if the DJ distribution mimics the RJ distribution than many measurement techniques will overestimate RJ. In fact, a large number of small deterministic effects can result in a distribution that is indistinguishable from a Gaussian down to a very low BER. These are called "truncated Gaussians." If the Gaussian is truncated at a BER of 10^{-8} and a data sample of 10^{7} logic transitions is used to measure it, then the RJ is upwardly biased and Eq. (1) will yield a higher TJ(BER) than is the truth. Amplitude noise can also seriously complicate the situation; jitter analysis only considers one dimension of a two-dimensional problem.

All About the Acronyms: RJ, DJ, DDJ, ISI, DCD, PJ, SJ, …

Ransom Stephens, Ph.D.

Abstract:

Jitter analysis is yet another field of engineering that is pock-marked with acronyms. Each category and type of jitter has its own acronym and every one of them gives insight into problems that limit the bit error ratio of a system. In this paper, we define the categories and types of jitter, their origins and interrelationships, and how they can be used to diagnose and debug system hardware.

Acronyms seem like convenient abbreviations once we know them, but prior to that, they're a closed language that can feel like no more than a bunch of obfuscating terms when used by others. Of course, when we use them, acronyms are simply shorthand that "everyone knows." In this paper we err in the opposite direction. Rather than annoy you by using too many acronyms, I'm going to annoy you by spelling out every acronym almost every time I use it.

Understanding a system usually consists of separating it into simple bite-sized pieces which combine to form an arbitrarily complex system. The same is true of jitter analysis. If we're given a system whose Total Jitter at a Bit Error Ratio of 10^{-12}, i.e., TJ(BER) = TJ(10^{-12}), is larger than allowed by a product requirement or standard specification, it doesn't tell us anything about how to fix the problem [For a review of TJ(BER), see the first installment in this series, *Part I: The Meaning of Total Jitter*]. We need to re-categorize, analyze, and sort the system into successively more simple pieces, hence the acronyms.

The jitter "Family Tree," Figure 1, shows one way to categorize different types of jitter. The first branch separates Random (RJ) and Deterministic Jitter (DJ). The second branch splinters Deterministic Jitter (DJ) into Data-Dependent Jitter (DDJ), Periodic Jitter (PJ), and Bounded Uncorrelated Jitter (BUJ). The primary source of Data-Dependent Jitter (DDJ) is Inter-Symbol Interference (ISI), but the amount of ISI is affected by the level of Duty-Cycle Distortion (DCD) – hence the dotted line connecting DCD and ISI. Some people put Duty Cycle Distortion (DCD) under Data-Dependent Jitter (DDJ) because of the affect DCD has on Inter-Symbol Interference (ISI). I chose to put Duty-Cycle Distortion (DCD) under Periodic Jitter (PJ) – after all, DCD is an asymmetry in the clocking of logic transitions. Periodic Jitter (PJ) is on the uncorrelated side of the diagram. The terms "correlated" and "uncorrelated" in the diagram indicate whether the amplitude of jitter changes with different transmitted data signals or data rates, that is, whether or not the jitter amplitude is “correlated to the data.” Generally Periodic Jitter (PJ) is uncorrelated, but Duty-Cycle Distortion (DCD) is a type of PJ that is correlated, so it’s on that side of the diagram. Sinusoidal Jitter (SJ) is the simplest type of Periodic Jitter (PJ) and is rarely correlated to the data.

There are many different ways that jitter can be categorized in a diagram like that in Figure 1. The important thing is not how we draw such a figure, but that we recognize the components, understand their causes, and appreciate whether or not one component can affect another.

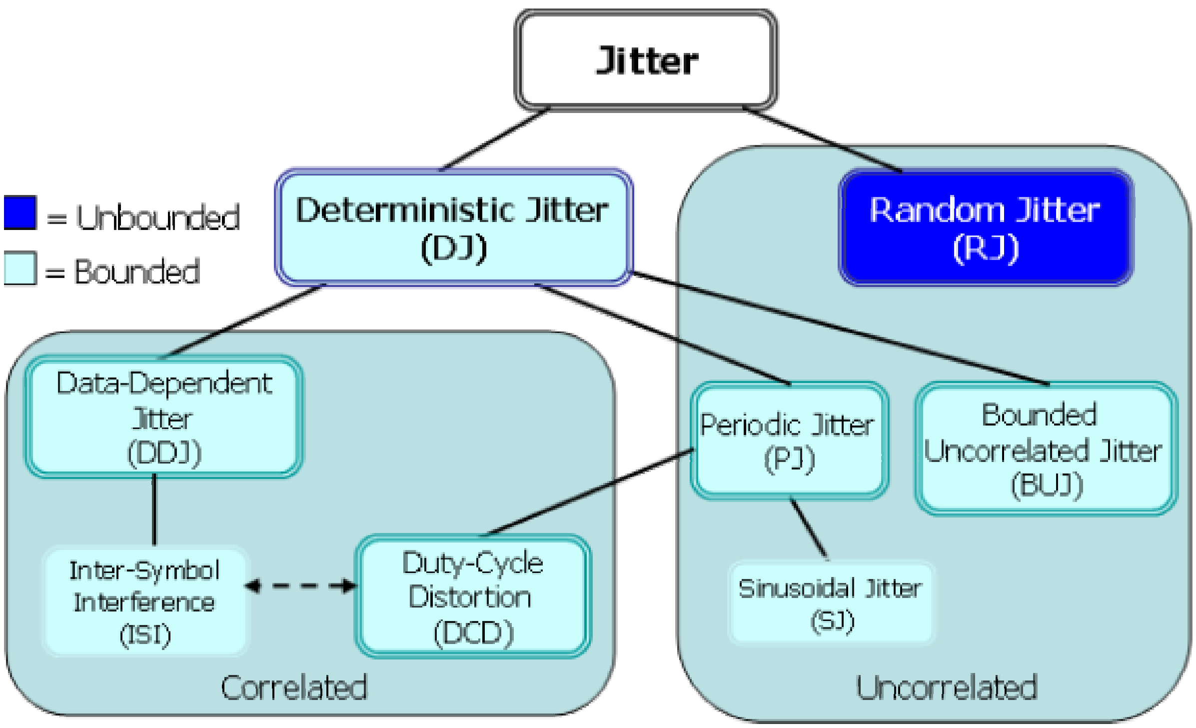


Figure 1: The jitter “Family Tree.”

In the following sections we’ll go through each type of jitter and sort out some of the different ways that jitter is categorized.

Random Jitter – RJ

As described in the first installment of this series, *The Meaning of Total Jitter*, Random Jitter (RJ) is caused by the combination of a huge number of sources, each of very small magnitude. According to the Central Limit Theorem, RJ should follow a Gaussian distribution, Figure 2.

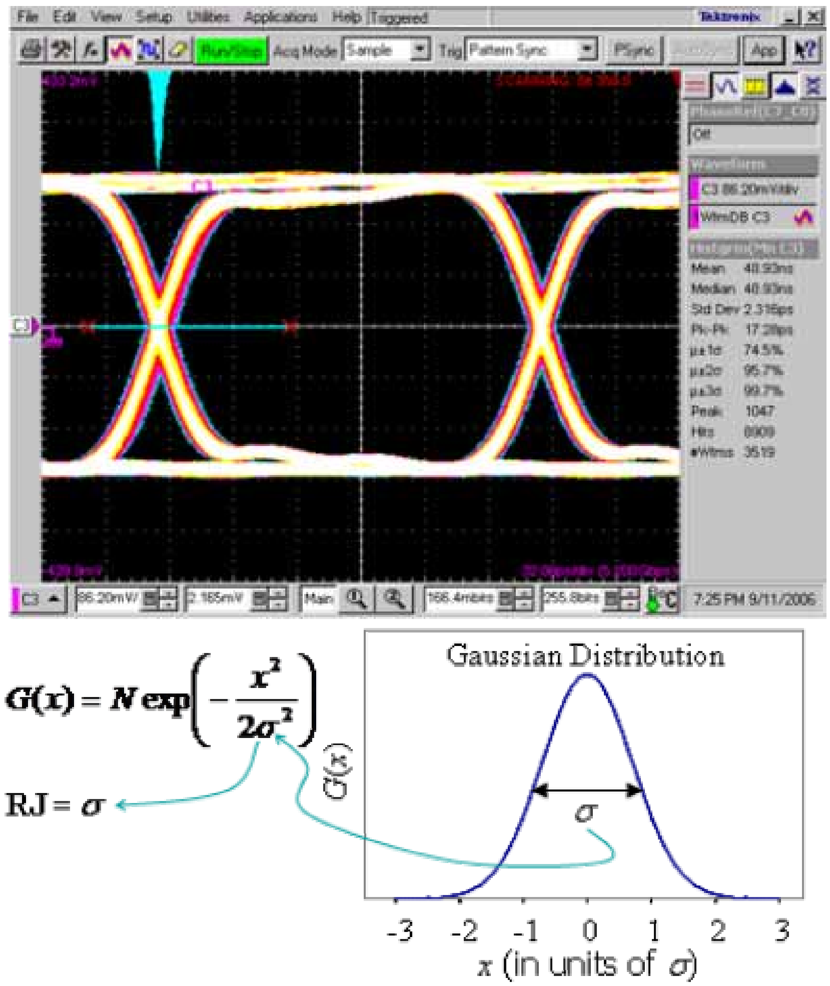


Figure 2: Random Jitter (RJ) follows a Gaussian distribution.

RJ is primarily caused by thermal processes, microscopic variations in the resistance and impedance of circuit traces which can be caused by the inevitable small variations of trace width, dielectric properties such as asymmetries in the weave of FR-4, and many other microscopic effects that are statistically impossible to isolate.

Since RJ follows an unbounded distribution it is quantified by the width, or standard deviation, σ, of its distribution.

- RJ is unbounded – there is a finite probability that random effects could cause a logic transition to appear *anywhere*, though, of course, the probability of an extremely large amount of RJ on a given transition is increasingly small. For example, the probability of RJ causing jitter greater than seven times the standard deviation of its distribution is one in a trillion.

- RJ is uncorrelated to the data – the amount of RJ on a given transition is not related to the transmitted data signal or data rate.
- RJ is aperiodic – RJ is random in nature and doesn't occur with any predictable regularity.
- RJ is independent of the other sources of jitter in the sense that changing RJ has no effect on the magnitudes of other types of jitter.

Deterministic Jitter – DJ

Deterministic Jitter (DJ) is the jitter that remains after Random Jitter (RJ) has been removed. In principle, though almost never in practice, DJ can be calculated from a complete understanding of the circuit and its environment. Since DJ can be composed of all the other types of jitter, it doesn't follow a given distribution function the way that Random Jitter (RJ) follows a Gaussian. On the other hand, since DJ is composed of a finite number of deterministic processes its distribution is bounded.

We usually characterize DJ by either its peak-to-peak value, DJ(p-p), or a model dependent version of the peak-to-peak value that is derived from the remarkably convenient dual-Dirac model, DJ($\delta\delta$) – which is described in the second part in this series, *The Dual-Dirac Model, What it is and What it is Not*.

Problems caused by DJ can be diagnosed by further resolving it into its constituents.

- DJ is bounded but doesn't follow a general distribution function.
- DJ may include both periodic and aperiodic components that may or may not depend on the transmitted data signal. That is, the DJ of a given logic transition may or may not affect the jitter of another transition.

Duty-Cycle Distortion – DCD

Duty-Cycle Distortion (DCD) is a measure of the asymmetry in the duty cycle of the transmitter. In Figure 3, the one in the sequence 00001000 is of a different width than the zero in 11110111 – the duty-cycle is distorted. Another manifestation of DCD is the difference in the rise and fall times of a signal resulting in a fixed time displacement of the rising and falling edges. In either case, in the absence of Inter-Symbol Interference (ISI), DCD follows a simple bimodal distribution. Equivalently, the amplitude of DCD is given by the difference of the average positions of rising and falling edges.

DCD is usually caused by an asymmetry in either the clock signal driving the transmitter or in a limiting amplifier within the transmitter.

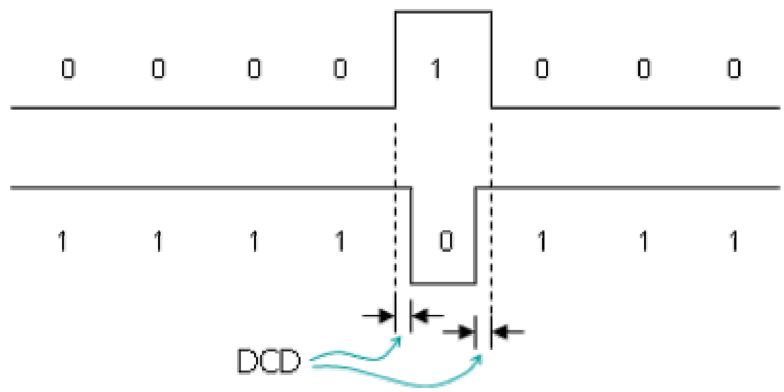


Figure 3: Simple example of Inter-symbol interference.

DCD is correlated with Inter-Symbol Interference (ISI) in the sense that changing ISI can result in a change in DCD, and vice versa. Another way to think of the correlation is in the sense of interference; DCD and ISI interfere with each other. Because of this correlation or interference, DCD is sometimes categorized with Inter-Symbol Interference (ISI) under Data-Dependent Jitter (DDJ). In a system with no Inter-Symbol Interference (ISI), the level of DCD is explicitly independent of the transmitted data signal and so is not really "data-dependent."

- DCD is bounded
- DCD follows a simple bimodal distribution
- DCD is usually caused by a clock asymmetry or limiting amplifier imperfection and so is periodic at the data-rate
- DCD is correlated with Inter-Symbol Interference (ISI) – a change in DCD causes a change in ISI and vice versa

Data-Dependent Jitter – DDJ

Data-Dependent Jitter (DDJ) encompasses all jitter whose magnitude is affected by the transmitted data signal. For example, as illustrated in Figure 4, when the jitter of a 0 → 1 transition that follows a sequence of alternating bits, e.g., 01010101, differs from a 0 → 1 transition that follows a long string of identical bits, e.g., 00000001, that jitter is called data-dependent.

DDJ is caused primarily by a combination of macroscopic impedance mismatches, the resistance and frequency response of the transmission path, and asymmetries in the rising and falling edges of the transmitted signal. DDJ is usually associated with Inter-Symbol Interference (ISI) but is affected by Duty-Cycle Distortion (DCD). The result, as shown in Figure 5, is a waveform that differs significantly from the ideal.

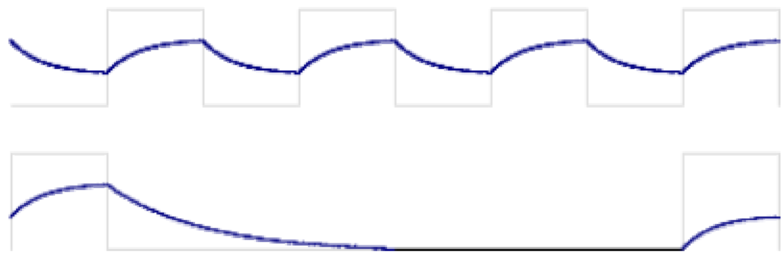


Figure 4: Example waveform of simple Data-Dependent Jitter (DDJ).

Generally two random variables are *correlated* if changing one of the variables causes the other to change. Correlation is an important concept which appears in two separate contexts in jitter analysis. First, as already described, jitter is said to be correlated to the data if the amplitude of jitter is affected by the transmitted data signal or the data rate. Second, the primary source of DDJ, Inter-Symbol Interference (ISI) is correlated to, or interferes with, Duty-Cycle Distortion (DCD) – a change in the ISI of a signal changes the DCD of that signal and vice-versa.

DDJ is a type of “correlated jitter” by virtue of its dependence on the transmitted data signal.

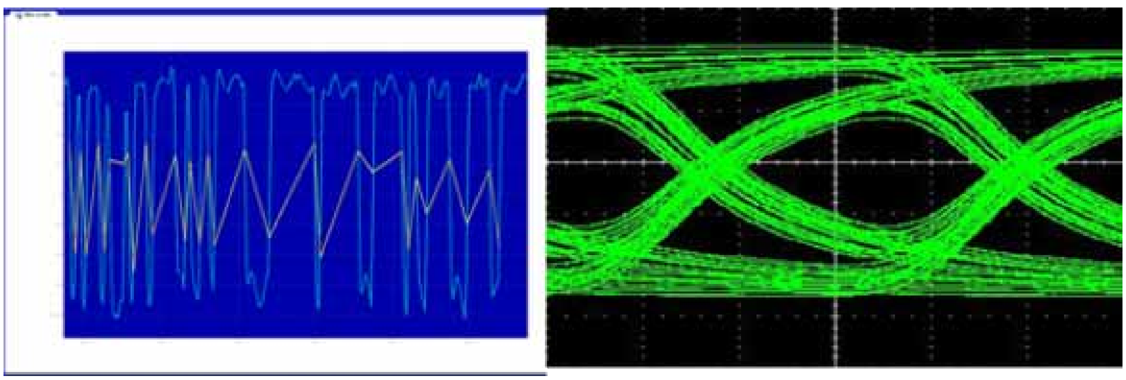


Figure 5: Data-dependent jitter and eye diagram.

- DDJ follows a distribution that can’t be described by any single global distribution function
- DDJ is also known as “correlated jitter” because data-dependent is equivalent to “correlated to the transmitted data signal”

Inter-Symbol Interference – ISI

Inter-Symbol Interference (ISI) is the primary cause of Data-Dependent Jitter (DDJ). The situation is complicated by the correlation of ISI and Duty-Cycle Distortion (DCD).

ISI is caused by a combination of the design of the trace and circuit geometry, the media composing both the conductor and dielectric of the circuit, and the waveform of the transmitted signal. In much of the literature, the design of the transmitter itself, including package design, is neglected as a source of ISI. It is important to keep in mind that the ISI of the transmitted waveform can interfere with the ISI introduced by the transmission path. A lot of confusion erupts from comparison of ISI measurements on a given cable but with different transmitters.

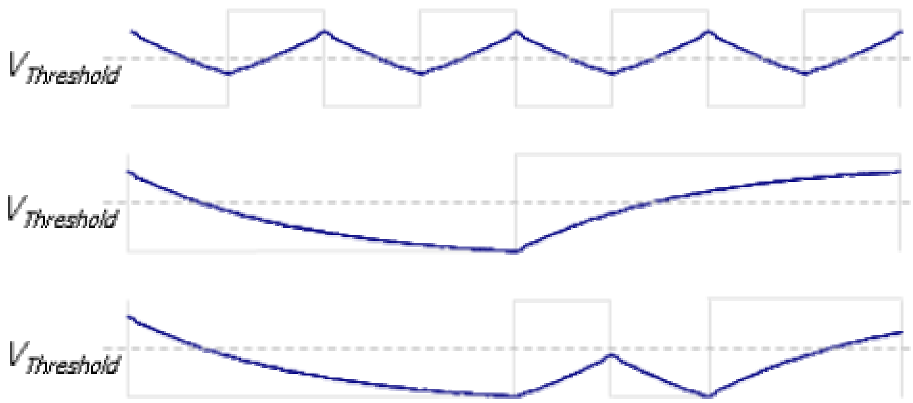


Figure 6: Simple example of Inter-Symbol Interference (ISI). $V_{Threshold}$ is the logic-decision threshold, if the observed voltage is greater than $V_{Threshold}$ then the bit is identified as a 1, if less than $V_{Threshold}$, a 0.

A good example of ISI is the modification of the pulse-shape of different bits in a signal as they traverse a transmission line. Transmission lines at data rates above about 100 MHz are better thought of as complicated waveguides in a dielectric medium. The resistance of the conducting traces cause signal attenuation and the frequency-dependence of the dielectric medium causes non-uniform frequency response. The non-uniform frequency response subjects the signal to a filtering effect. The dominant frequency component of a given bit is determined by the identity of the bits that surround it. Consider the data sequences shown in Figure 6 where a simple Resistor-Capacitor (RC) time constant is used to illustrate ISI. In Figure 6a, the data signal, 01010101, is a clock signal at half the data rate. The response of the circuit to the transmitted data signal is sufficient for each bit to cross the logic-decision voltage threshold and be accurately identified. In Figure 6b, the data signal, 00001111 is a clock signal at one-eighth the data rate. Over the string of Consecutive Identical Bits (CIB or CID) the time constant is sufficiently short for the signal to reach the voltage rail but, in so doing, does not permit enough time for the signal to cross the voltage threshold during the first logic 1 following the string of 0s. In fact, the 00001111 string would be identified as 10000111. A mixed example is shown in Figure 6c; here the 00001011 signal would be identified as 10000001.

Now combine the unavoidable non-uniform frequency response of the dielectric, like Flame Retardant Type-4 (FR-4), with the interference of signals from discrete impedance mismatches that cause multiple reflections and multiple paths from input to output and you can see how ISI can cause complex problems, Figure 7. The result is that the waveform of any bit in a data signal can differ depending on the number of Consecutive Identical Bits (CIB or CID) preceding it. Data coding, for example 8b10b coding, is used in many technologies, e.g., PCI Express, to prevent the occurrence of long strings of Consecutive Identical Bits (CIB or CID). An important feature of ISI is that it affects both the amplitude and timing of the signal. In other words, it is a source of amplitude noise as well as jitter.

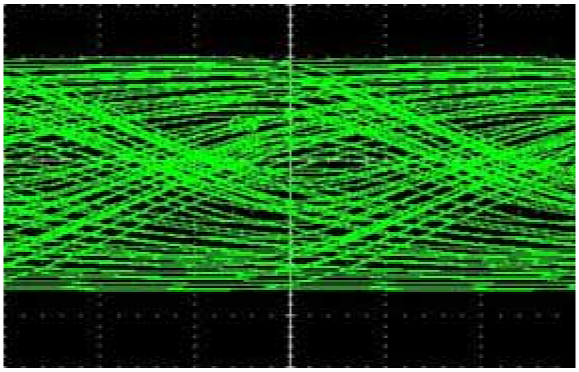


Figure 7: A signal with a great deal of Inter-Symbol Interference (ISI).

It is difficult to combine the ISI introduced by different circuit elements. For example if the transmitted signal has a given level of ISI, then the ISI introduced by the transmission path will be different than if the transmitted signal had no ISI. In other words, the ISI of consecutive circuit elements interferes with one another – it is *correlated*.

ISI provides a good example of what it means to be "deterministic." Time Domain Reflectometry (TDR) can be used to measure the frequency and attenuation characteristics of a circuit resulting in the impulse response as represented, for example, in the Scattering parameters (*S*-parameters), from which the circuit's transfer function can be calculated. The transfer function can then be applied to any data sequence to yield the ISI waveform. That is, the transfer function is the complete understanding of the circuit and its environment and, with it, we can predict the jitter of any given logic transition – exactly what is meant by *deterministic* jitter. Due to its predictable nature, it is possible to correct ISI at the receiver using equalization techniques.

There are two types of diagnostic techniques that can be used to reduce ISI. First, before a circuit is even built, the ISI of different designs can be compared by calculating the Scattering parameters (*S*-

parameters) in a simulation. Second, Time Domain Reflectometry (TDR) can be used to locate discrete impedance mismatches like those often found at vias and connectors.

- ISI is bounded
- ISI is only periodic if the signal is a repeating pattern
- ISI is caused by the geometry and media of the conductor and dielectric
- ISI is also caused by discrete impedance mismatches like those found at vias and connectors which result in multiple reflections
- ISI can be introduced by the transmitter
- ISI of different circuit elements are correlated to each other – they interfere
- ISI can be predicted from the impulse response, such as can be derived from the Scattering parameters (*S*-parameters) as measured through Time Domain Reflectometry (TDR)

Periodic Jitter – PJ

Periodic Jitter (PJ) includes any jitter at a fixed frequency or period. PJ is ultimately an example of periodic phase modulation and may be the most useful category of jitter. It's easy to measure accurately and appears in the jitter-frequency spectrum as distinct peaks. The jitter-frequency is the offset frequency of the jitter with respect to the data rate. When a PJ peak is identified it can usually be associated with a noisy circuit element operating at that same frequency. The classic example is power supply feed-through, but PJ can also be caused by crosstalk from neighboring data lines or any other type of Electromagnetic Interference (EMI).

PJ is always bounded and may have components that are correlated to the data, such as Duty-Cycle Distortion (DCD).

- PJ can have a variety of wave shapes (e.g., square-wave phase modulation is PJ that results in a dual-Dirac distribution) with corresponding jitter-frequency spectra
- PJ is bounded and follows a distribution that can be calculated if the amplitudes, frequencies, and relative phases of all harmonics and PJ sources are measured
- PJ is easy to measure accurately
- PJ is useful in diagnosing jitter problems

Sinusoidal Jitter – SJ

Sinusoidal Jitter (SJ) is Periodic Jitter (PJ) at just one frequency. A tremendous amount of work has been done in SONET/SDH jitter analyses to precisely calibrate jitter analyzers by applying SJ of known frequency and amplitude and using the classic Bessel null technique.

In the time domain, SJ follows an easily identifiable "suspension bridge" structure as shown in Figure 8.

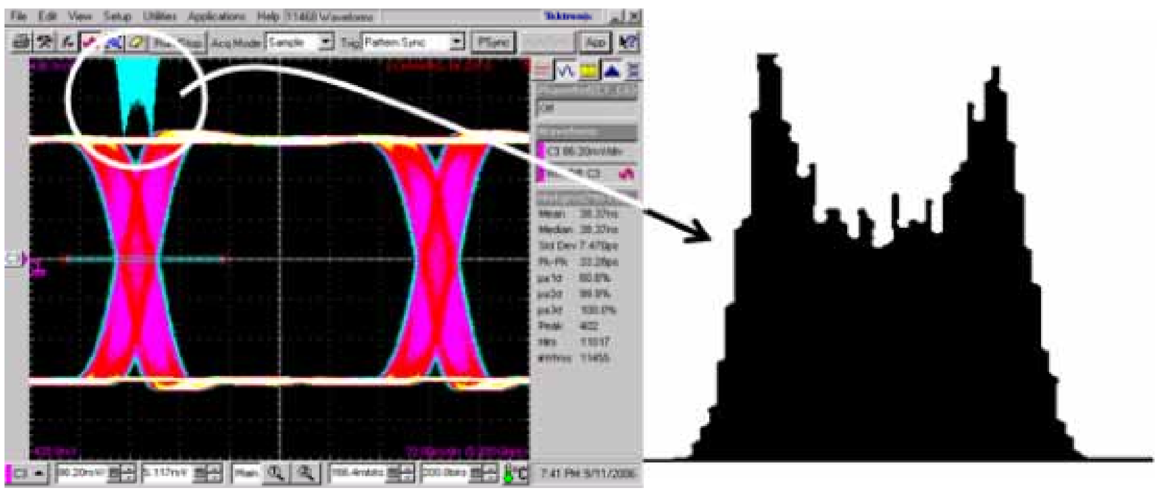


Figure 8: Sinusoidal jitter.

- SJ is Periodic Jitter (PJ) at just one frequency
- SJ is bounded and uncorrelated to the data
- SJ is easy to measure accurately
- SJ can be applied to a signal for use in calibrating test equipment

Bounded Uncorrelated Jitter – BUJ

Bounded Uncorrelated Jitter (BUJ) is a category used by most of the industry for organizing ignorance. That is, most of the literature uses BUJ to represent all the types of jitter that we don't know how to measure. We assume that the only unbounded jitter is Random Jitter (RJ), by implication BUJ is deterministic.

The two most debated flavors of BUJ are non-stationary jitter and the jitter-effects of crosstalk. Non-stationary jitter is sporadic – neither periodic nor data-dependent – over the possible measurement time scales. A good example is the state-dependent jitter of complex gate arrays where certain data patterns present a repeatable, but rare, switching transient. Crosstalk presents a more challenging problem for jitter analysis, one where the distinction between amplitude noise and timing jitter can't be neglected.

Strictly speaking the most easily measured type of jitter, Periodic Jitter (PJ), is both bounded and uncorrelated and so could accurately be categorized as BUJ. One manufacturer of test equipment uses the acronym BUJ for Periodic Jitter (PJ). It is semantically accurate, but inconsistent with the vast majority of jitter literature, an unfortunate choice that has caused a lot of confusion.

- BUJ bounded and uncorrelated to the data
- BUJ is usually used as a receptacle for the jitter we can't measure
- The two most commonly discussed sources of BUJ are non-stationary jitter and the jitter effects of crosstalk

Conclusion

There are many ways to categorize jitter. The most common criteria are:

- Random Jitter (RJ) vs Deterministic Jitter (DJ) which is equivalent to unbounded vs bounded
- correlated vs uncorrelated
- data-dependent vs data-independent
- periodic vs aperiodic

There is room to argue about whether some types of jitter, like Duty-Cycle Distortion (DCD), should be categorized the way they I did here, but these arguments distract from the important issues. Namely, how measurements of the different types of jitter can be used to reduce the Bit Error Ratio (BER) of a system.

	acronym	bounded/ unbounded	correlated/ uncorrelated	periodic/ aperiodic	Example cause
Random Jitter	RJ	Unbounded	Uncorrelated	Aperiodic	Thermal noise
Deterministic Jitter	DJ	Bounded	Either	Either	Inter-Symbol Interference
Periodic Jitter	PJ	Bounded	Either	Periodic	Power supply feed-through
Sinusoidal Jitter	SJ	Bounded	Uncorrelated	Periodic	Electromagnetic interference
Data-Dependent Jitter	DDJ	Bounded	Correlated	Aperiodic	Impedance mismatch
Duty-Cycle Distortion	DCD	Bounded	Correlated	Periodic	Clock asymmetry
Inter-Symbol Interference	ISI	Bounded	Correlated	Aperiodic	Non-uniform frequency response of a transmission line
Bounded Uncorrelated Jitter	BUJ	Bounded	Uncorrelated	Aperiodic	Crosstalk

Table 1: Summary of jitter acronyms.

Jitter Analysis in Systems with Crosstalk

Ransom Stephens, Ph.D.

Abstract:

In many cases jitter can be analyzed independently of voltage noise, but not in the presence of crosstalk. Crosstalk is a source of bounded uncorrelated jitter that is notoriously difficult to identify and tends to confound jitter analysis algorithms. In this paper we introduce crosstalk as an example of a jitter source that doesn't fall conveniently under one of the standard jitter acronyms and presents a challenge that can be addressed by expanding from jitter analysis to the full two dimensions of phase noise and amplitude noise.

Noise is noise. Consider an eye diagram like that in Figure 1 – the plot of overlapping logic signals. Noise is evident on the signal in both the vertical (voltage) and horizontal (time-delay) directions. Since jitter is usually the source of Bit Error Ratio (BER) problems it's easy to fall into the trap of thinking that jitter is only caused by timing noise, but voltage noise also causes jitter. In many cases it is important to remember what I've mentioned several times over the course of this series:

Jitter analysis only considers one dimension of the two-dimensional noise problem.

Inter-Symbol Interference (ISI) is a good example of a noise source that obviously affects both the horizontal and vertical components of an eye-diagram.

In Part 3 of this series, *All About the Acronyms: RJ, DJ, DDJ, ISI, DCD, PJ, SJ,…* I referred to Bounded Uncorrelated Jitter (BUJ) as "all the types of jitter that we don't know how to measure." Crosstalk is perhaps the best example of BUJ.

We begin by showing how jitter and voltage noise are each caused by either or both of phase noise and amplitude noise. We then introduce crosstalk, describe how it affects signals and how differential signaling reduces the signal degradation. With that out of the way we discuss a few ways to tell when crosstalk is confusing your jitter analysis techniques and how crosstalk can be identified by simultaneously analyzing the jitter and voltage noise of a signal.

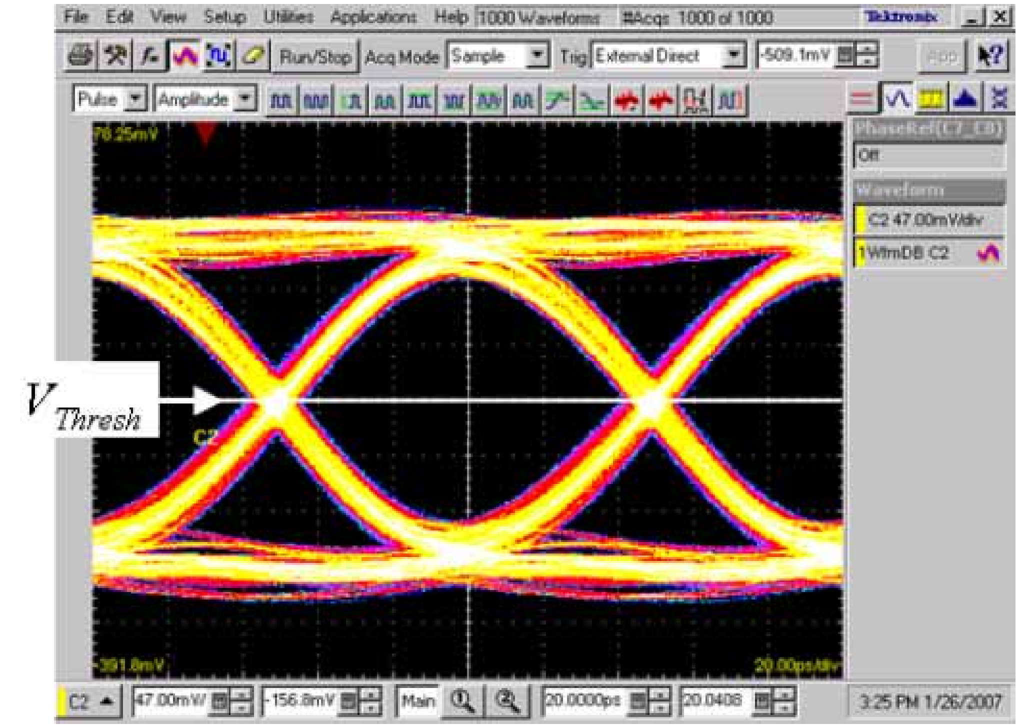


Figure 1: An eye diagram with a variety of causes of jitter and voltage noise.

Phase Noise vs. Amplitude Noise – Jitter vs. Voltage Noise

Jitter is the variation in the timing of the significant instants of a digital signal. In other words, if we choose a voltage threshold in the eye-diagram, at say V_{Thresh} in Figure 1, then the time-delays at which each signal crosses that threshold are the "significant instants" and their variation over different logic transitions is what we think of as jitter.

Phase noise is the variation in the phase of a signal; we can think of it as noise that shifts the signal back and forth horizontally. Similarly, amplitude noise is the noise that shifts the signal vertically up and down. Jitter can be caused by either and, in general, is caused by both. In the limit of an ideal digital signal with zero rise and fall times (infinite slew rates), jitter would only be caused by phase-noise. In a real signal, logic transitions have continuous trajectories with finite slew rates and non-zero rise/fall times. Consider Figure 2a. The logic transition drawn with a solid line occurs at time t_0. If that same transition is shifted vertically downward, then the transition occurs at t_1. The vertical fluctuation causes jitter of $t_0 - t_1$. The slower the rise-time, the greater the impact of amplitude noise on jitter. Phase noise has the same effect on voltage noise only rotated by 90°, Figure 2b.

Figure 2: The effect of (a) amplitude noise on jitter and, (b), phase noise on voltage noise.

If the slope, or slew rate, of the signal is *dV/dt* and the voltage shift is δV, then the effective peak-to-peak jitter caused by the voltage shift is,

$$J_{PP} \approx \frac{\delta V}{dV/dt}. \qquad (1)$$

In analyzing voltage noise we can apply a bunch of acronyms just as we did for jitter (see Part 3 of this series, *All About the Acronyms: RJ, DJ, DDJ, ISI, DCD, PJ, SJ,…*):

- RJ → RN (random noise)
- DJ → DN (deterministic noise)
- DDJ → DDN (data-dependent noise)
- PJ → PN (periodic noise)

Further, we can discriminate between the horizontal (phase-noise) and vertical (amplitude noise) contributions to each type of jitter or noise. For example, we can distinguish RJ(h) and RJ(v). We'll see that expanding the analysis in the vertical direction provides useful tools for determining how much of a problem is caused by timing noise and how much by amplitude noise.

Crosstalk

Crosstalk occurs when one signal is affected by a neighboring signal. It is usually a capacitive coupling between nearest neighbors and can be reduced by shielding, increasing the space between signal-carrying conductors, limiting the slew rates of signals, and the use of differential signaling.

At high data rates a signal propagates more like a guided wave than a simple DC current. The wave is guided by the conducting trace but radiates through the dielectric medium. When more than one signal is present, every conducting trace on the board includes artifacts of the signals on every other trace. The common jargon is to say that an *aggressor* signal causes crosstalk on a *victim* signal. Crosstalk occurs when the signal of an aggressor is picked up by the conductor guiding the victim signal. Unavoidable discontinuities in circuit layout, like connectors and vias, where capacitive coupling is greatest, are critical points that act like antennas in generating crosstalk. Ultimately, crosstalk is a form of electromagnetic interference (EMI). Since electromagnetic fields are linear, the magnitude of the aggressor signal at the victim trace simply adds to the victim signal.

Maxwell's equations show that electromagnetic radiation is caused by the acceleration of electric charge. If we think of digital signals as DC but with sharp transitions between logic levels, the greatest acceleration of charge is during the transitions. Thus the greatest amount of electromagnetic radiation is emitted during logic transitions. The amplitude of the radiation is proportional to the charge acceleration, given by the rate of change of the current, di/dt, which, by Ohm's law, is proportional to the slew rate of the aggressor, dV_{agg}/dt.

Think of a pulse of noise radiating through the circuit board. At every logic transition of the aggressor signal, crosstalk appears on the victim as a sharp jolt of amplitude noise. Since the amount of noise is proportional to the slew rate, crosstalk can be reduced by limiting the slew rate of signal transitions or, equivalently, setting a lower bound on rise and fall times.

Crosstalk is usually quantified as the ratio of the voltage of the largest crosstalk component induced by the aggressor to the victim voltage in decibels.

It is useful to keep in mind that crosstalk is highly directional. Crosstalk noise that propagates in the same direction as the victim signal is called Far End crosstalk (FEXT) and crosstalk that propagates in the opposite direction of the victim signal is called Near End crosstalk (NEXT). FEXT is usually attenuated by the expanse of dielectric between aggressor transmitter and victim receiver, making NEXT the more aggravating of the two.

The most common technique for reducing crosstalk in serial data systems is the use of differential signaling. A signal is transmitted by two very close conducting traces, one with a signal and the other with the inverse of the signal. Since the sum of the two is zero, no net radiation should propagate through the dielectric medium of the board. At the receiver, the voltage of one trace is subtracted from the other giving twice the voltage spread of the signal on either trace. Ideally no crosstalk noise can propagate in a differential system, but at high data rates the realities of nonzero skew between the two traces and the necessary non-zero separation of the traces allow some radiation to leak out.

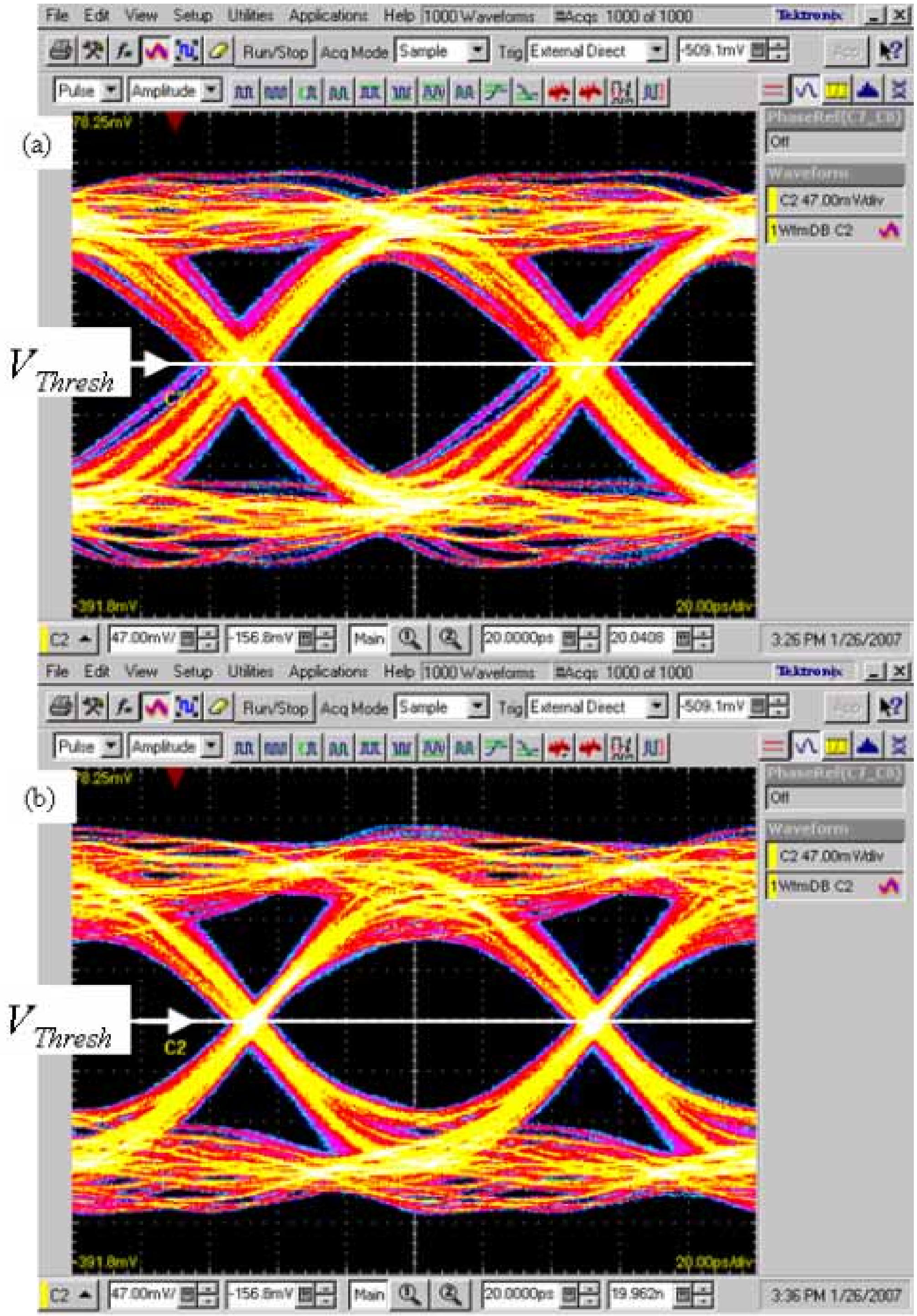


Figure 3: Crosstalk with a frequency-locked aggressor. In (a) pulses of crosstalk noise coincide with logic transitions and, in (b), pulses of crosstalk noise close the center of the eye.

At higher data rates, trying to reduce crosstalk by limiting transition slew rates is impractical. Differential signaling is still effective at reducing crosstalk, but the crosstalk problem itself is so much larger that differential signaling isn't enough.

If the crosstalk aggressor and the victim signal are governed by the same reference clock then they are probably both frequency and phase locked – that is, they are transmitted at precisely the same underlying phase and have a fixed phase relationship. When the aggressor and victim are frequency and phase locked, the crosstalk signature appears in an eye diagram as a localized region where the trajectory splinters into separate trajectories over a short period, as shown in Figure 3. An upward pulse occurs when the aggressor has a 0 → 1 transition, there's no effect when the aggressor does not undergo a transition and there is a downward pulse when the aggressor experiences a 1 → 0 transition. Design engineers are usually conscious of this possibility and, when possible, design a relative skew of the victim and aggressor traces so that the signature occurs during the victim's logic transitions as in Figure 3a. This way the effect is concentrated at the crossing point, as far from the eye-center as possible so that the effect on the receiver decision circuit is minimized. In other words, systems are designed to minimize the voltage noise at the expense of maximizing the jitter. Figure 3b shows the effect at the eye-center.

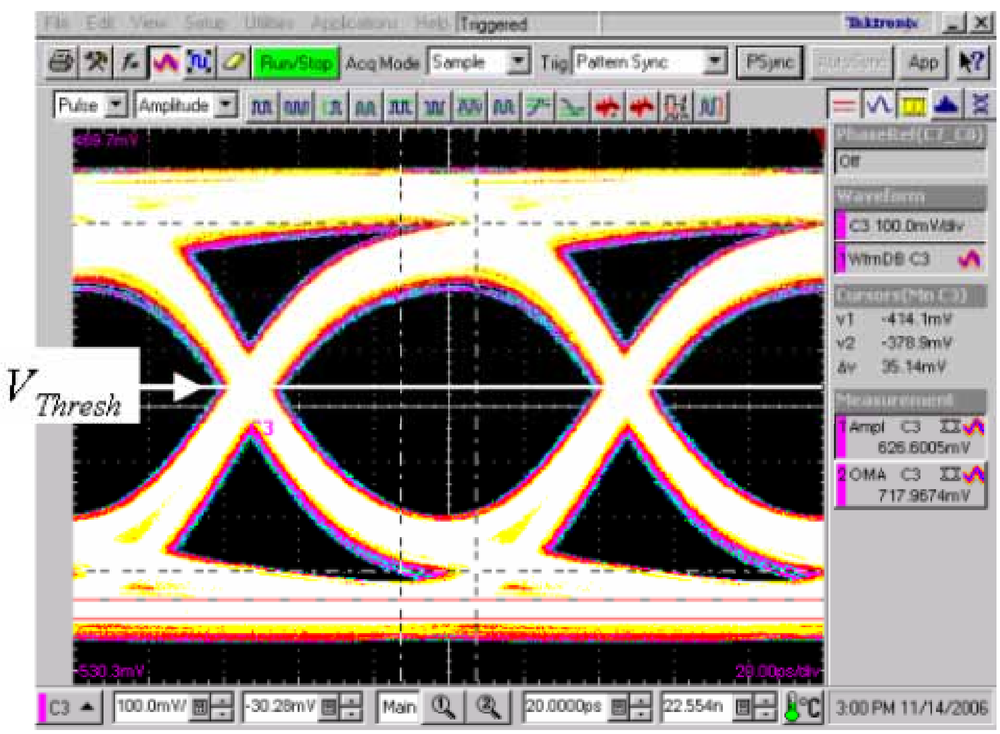


Figure 4: Crosstalk without a frequency-locked aggressor.

When the aggressor and victim are not governed by the same reference clock or, for whatever reason, are not frequency and phase locked, then, in an eye-diagram, crosstalk looks like an excess of voltage noise, Figure 4.

In the frequency domain, if the victim and aggressor are frequency and phase locked crosstalk appears as a set of spikes at subharmonics of the data rate, Figure 5a. Spectrally-based analysis techniques appropriately identify crosstalk as both Periodic Jitter (PJ) and Periodic Noise (PN). Tail-fitting techniques, whether applied to bathtub plots or jitter histograms, would mistake the crosstalk for RJ, dramatically increasing their extrapolated Total Jitter defined at a Bit Error Ratio, TJ(BER), estimates.

On the other hand, if the victim and aggressor are not frequency and phase locked, then the situation is more complicated. On real-time sampling equipment, where the frequency components are not aliased, the jitter and voltage noise spectra have subharmonic peaks that, rather than appearing as sharp lines, are smeared into broad resonance shapes. On under-sampling equipment, like an equivalent-time sampling oscilloscope, where the spectrum is aliased, crosstalk appears as continuous noise, Figure 5b. In both cases spectrally-based jitter analysis techniques that measure RJ by integrating the jitter-spectrum continuum, mistake at least some crosstalk for RJ and RN. In this case too, tail-fitting techniques mistake crosstalk for RJ and, in both cases TJ(BER) is overestimated.

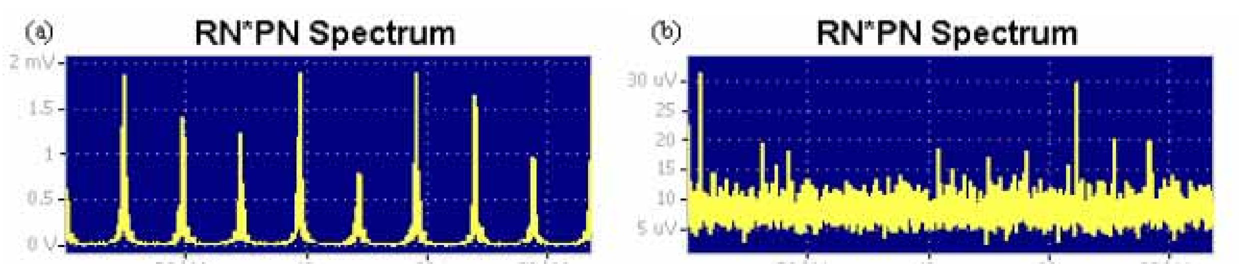


Figure 5: Frequency domain view of (a) frequency and phase locked crosstalk, and, (b) neither frequency nor phase locked crosstalk.

Signal Analysis with Crosstalk

The best way to deal with crosstalk problems, of course, is to anticipate the problem and design around it. If the differential *S*-parameters between aggressor and victim lanes are evaluated in a simulation before the board is built, then aggressor voltages at the signal trace can be calculated and the design revised until the crosstalk problem is resolved. Similarly, on an existing circuit, if the differential *S*-parameters can be measured (for example by Time Domain Reflectometry) then the crosstalk problem can be calculated. Tektronix offers a package for network parameter measurements, IConnect, that automatically calculates up to eight aggressors. The next best approach is to apply a signal to neighboring lines and measure the response on the victim line using a setup like that in Figure 6. The ratio of the crosstalk voltage at the victim trace to the victim's signal voltage, usually expressed in decibels, is the crosstalk signal.

Once you have the maximum aggressor voltage at the signal trace, you can estimate the maximum jitter caused by crosstalk with Eq. (1) by using the smallest slew-rate, *dV/dt*, of the signal trajectories.

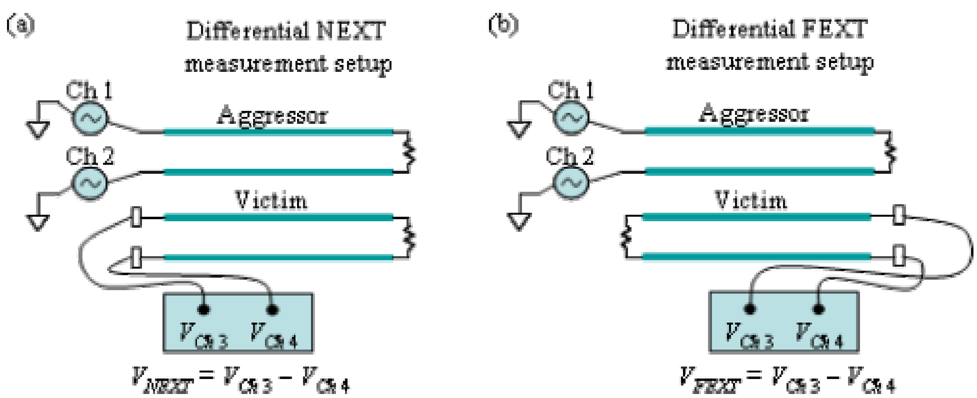


Figure 6: Setup for measuring crosstalk in differential systems. (a) For Near End Crosstalk (NEXT) and (b) for Far End Crosstalk (FEXT).

Let's turn to the more annoying situation: We're given a system whose Bit Error Ratio is too high and we don't know why – the unfortunate but usual case for those of us in the signal integrity analysis business.

In performing a simple one-dimensional jitter analysis there are few ways to tell if crosstalk is the problem. If the data on neighboring lines is governed by the same reference clock so that the signals are frequency and phase locked, then an obvious crosstalk signature is the combination of a comb-like structure of subharmonics in the jitter spectrum, as in Figure 5a, and the eye diagram splintering at the crossing point as in Figure 3.

If potential aggressors don't share a reference clock with the victim then the situation is more difficult. The first clue is if the jitter analyzer reports an inordinately large RJ measurement. It is rare that thermal effects, the ultimate cause of RJ, manage to conspire to greater than 3 ps RMS. If the RJ reported is larger than 3 ps then it's likely that crosstalk is causing problems.

Thorough characterization of a signal in the presence of crosstalk requires simultaneous voltage-noise and jitter analysis. Tektronix offers a software option, JNB, that runs on their equivalent-time sampling oscilloscopes. JNB performs the full two-dimensional noise analysis by, essentially, applying the same algorithms along the voltage axis to get RN, DN, et cetera that are used along the time-delay axis to get RJ, DJ, et cetera (see Pavel Zivny's white paper at the Tektronix web site for the details) and then extends the analysis to its logical conclusion, generating the two-dimensional version of a bathtub plot – a BER contour diagram.

Generally, when the rms random voltage noise, RN, in an eye-diagram approaches the rms jitter, RJ, it indicates that the problem is dominated by amplitude noise, rather than phase noise. Since RN is measured in Volts and RJ is measured in seconds, we can compare the relative contributions to RJ and RN through ratios of the contributions from horizontal (phase) and vertical (amplitude) noise. If

$$RN(v)/RN(h) >> RJ(h)/RJ(v) \tag{2}$$

then the problem is caused by amplitude noise. Since crosstalk appears to the analysis techniques as random noise when the victim and aggressors have separate reference clocks, when Eq. (2) holds it is a strong indicator of crosstalk.

Other tricks to identifying crosstalk require more control over the aggressor channel than you might have. For example, if it's possible to turn off the suspected aggressor signal, then you can compare the RJ measurement with and without a signal on the aggressor. If RJ-with aggressor > RJ-without then the problem is crosstalk. A work-around in this case is to use the measurement of RJ with the aggressor off and the measurement of dual-Dirac DJ with the aggressor on in the dual-Dirac model to estimate the Total Jitter of the system at the BER of interest, TJ(BER) (see Part 1 and 2 of this series for background on TJ(BER) and the dual-Dirac model respectively).

Since crosstalk is BUJ, it follows a bounded distribution. The bounded nature of the distribution is obscured by the complexity of the data pattern. The seemingly random distribution of 1s and 0s causes different amounts of voltage noise to be transmitted on each aggressor-signal transition. When crosstalk is combined with other types of jitter, particularly Data-Dependent Jitter (DDJ), pretty soon the central limit theorem comes into play and the jitter Probability Density Function (PDF, e.g., as represented by the crossing point histogram) appears to follow a Gaussian distribution.

If it is possible to control the data pattern of the aggressor, transmitting an alternating signal, 1010, would make it easier to identify crosstalk. The regularity of the pulses of crosstalk noise on the victim would sit at just the one frequency. In the frequency and phase-locked case the crosstalk would jump out as PJ and PN. In the unlocked case, it would be somewhat trickier. If the analysis were preformed on data-rate sampling equipment, like a real-time oscilloscope, then the smeared frequency peak should be observable; if the spectrum were aliased, it's harder to say. In either case, though, the distributions of uncorrelated jitter and noise, RJ∗PJ and RN∗PN, respectively, would not follow Gaussian distributions. The combination of a non-Gaussian distribution and no obvious PJ or PN peaks would indicate something funny is going on and odds are good that funny thing is crosstalk.

Conclusion

There are many approaches to the problem of jitter analysis on signals with crosstalk. None of them provide compartmentalized one-button-push results the way we can expect from, for example, measurements of DDJ and PJ. The main thing to keep in mind is that crosstalk is a type of amplitude noise, not phase noise. It creeps into signal analysis as jitter by virtue of the finite slew rate of real digital signals. Signals with crosstalk should be analyzed in the full two-dimensional space of phase noise and amplitude noise or jitter and voltage noise.

Use of a combined jitter and noise analysis technique makes it possible to identify crosstalk from analysis of a victim signal alone.

In general, even if it's not known whether or not crosstalk is on a signal, the TJ(BER) estimated from a combined jitter and noise technique is much more accurate than one can reasonably expect from a one-dimensional jitter analysis. It's one of the many ugly "features" of BUJ.

Clock Recovery in Serial-Data Systems

Ransom Stephens, Ph.D.

Abstract:

The definition of a bit period, or unit interval, is much more complicated than it looks. If it were just the reciprocal of the data rate we'd be in worse trouble with jitter than we already are. In this installment of Jitter 360, we investigate the true identity of the unit interval and how serial-data systems use a recovered clock instead of an independent reference clock. The investigation will reveal the key features of clock recovery that affect the bit error ratio, namely bandwidth and peaking – including which part of the jitter-spectrum matters most.

It's convenient to think of a receiver, as shown in Figure 1, as a device that uses a clock to position the sampling point in time so that a comparator can decide whether the signal voltage at that time is greater or less than a decision threshold value. If it's greater, the receiver identifies a logic 1, and if it's less, a logic 0. Setting the voltage decision threshold is simple – the threshold is almost always zero for differential systems – but the time-position of the sampling point is much trickier. This is where clock recovery comes in.

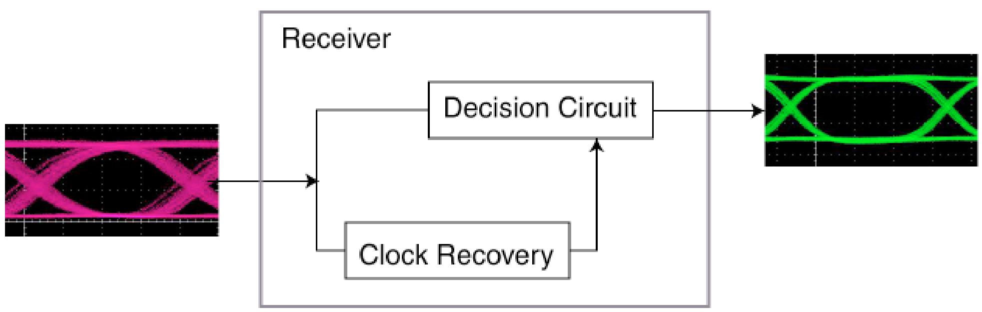


Figure 1: Simple diagram of a serial-data receiver.

Consider a simple system with an absolute external reference clock, Figure 2a. If we tune the relative phase of the clock so that it's in phase with the incoming data transitions and then have the receiver sample on falling clock edges, the sampling point should be the center of every bit, Figure 2b. The unit interval in this system is exactly the reciprocal of the nominal data rate. Usually when we speak of the timing of bits, this is what we have in mind. It's easy to think about, but it has some problems.

The first problem is that providing an absolute external clock to both the transmitter and receiver requires an extra data line and an expensive clock low jitter clock. Oops, did I say expensive?

The biggest problem, though, is that having the perfect external clock *increases* the Bit Error Ratio (BER)!

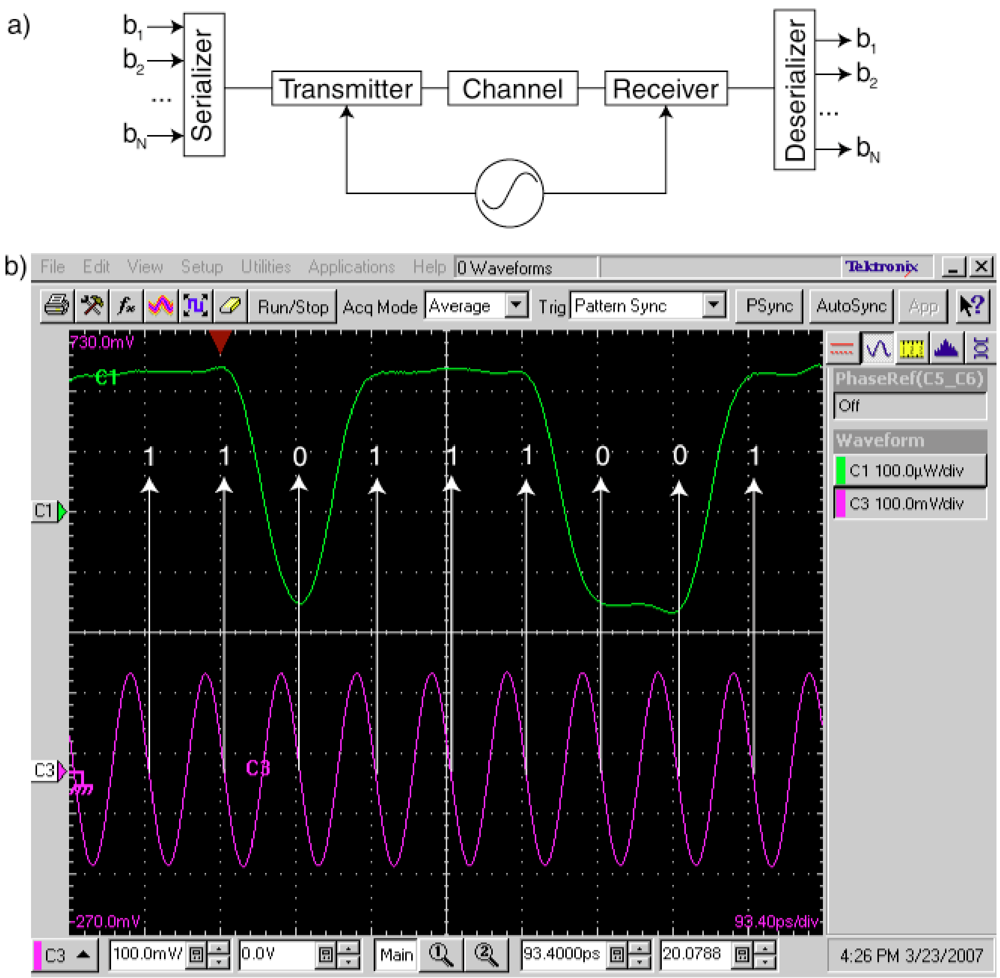


Figure 2: (a) System with an external clock and (b) setting the time-position of the sampling point.

What would happen if we could set the sampling point half a bit after the time at which logic transitions *actually* occur instead of half a bit after the time at which they *should* occur? Jitter would never cause an error!

In this ideal world, we could trigger on a logic transition and sample half a bit period later. The sampling point would have exactly the same jitter as the data and the signal would never fluctuate across the sampling point. The only price we would pay is a more complicated definition of a unit interval.

We can get close to such a utopian world by recovering the clock from the data itself. An infinite bandwidth clock recovery system would trigger the clock signal on a data transition and the sampling point timing would have the same jitter as the data. If the data and clock have the same jitter, then they dance in harmony and bits are identified not at ideal times but at the best times – the jitter on the clock *tracks* the jitter on the data and the BER isn't affected by jitter.

In real life, with a finite bandwidth clock recovery circuit, the low frequency jitter tracks the data. Only jitter at frequencies above the clock recovery bandwidth causes errors. It gets better, not only does clock reconstruction reduce the BER, but it allows the use of clocks that have lots of jitter on them (i.e., cheaper clocks) and it doesn't require a trace or cable to get a clock signal from the transmitter to the receiver (i.e., cheaper design).

Clock Recovery

There are two basic types of clock recovery, those that are more analog in nature, like Phase-Locked Loops (PLL), and those that are more digital in nature. Digital, in this context means that the clock is reconstructed from multiple discrete samples rather than the continuous analog data signal. The Phase Interpolator (PI) is a well known example, though there are many proprietary techniques. The biggest practical difference between PLLs and PIs is cost and the biggest theoretical difference is ease of parameterization and modeling. PIs usually have a faster lock time than PLLs, use less power, and require less surface area resulting in less expensive designs. Like any circuit element, a well designed PI can do its job just fine, certainly as well as a well designed PLL. The problem with PIs is that their nonlinear behavior is difficult to anticipate making it much harder to debug a troubled PI-based clock recovery system than one that is PLL-based.

Since transfer functions for whole families of PLLs can be written down in terms of a few simple parameters, where PIs can have different parameterizations for nearly every design, technology standards committees specify clock recovery in terms of PLLs. One approach is to use a simple second-order PLL with transfer function, $H(s)$

$$H(s) = \frac{2s\zeta\omega_n + \omega_n^2}{s^2 + 2s\zeta\omega_n + \omega_n^2}\,. \tag{1}$$

There are two key parameters associated with this PLL, the peaking, given by ζ, and the natural frequency ω_n. The bandwidth is given by

$$\omega_{3\text{dB}} = \omega_n\sqrt{1 + 2\zeta^2 + \sqrt{(1+2\zeta^2)^2 + 1}}\,. \tag{2}$$

The two parameters determine what range of jitter frequencies common to both the clock and data signals. Jitter at frequencies *below* the bandwidth cutoff doesn't cause errors unless the peaking, ζ, is too high. PLL peaking causes jitter frequencies near the cutoff to be *amplified* and show up in the recovered clock at greater amplitudes than in the data signal.

First order PLLs are a simple alternative which don't have any peaking. From the perspective of a standards body there is a tradeoff in which way to model clock recovery. The ideal would be to use an arbitrarily complex model so that every parameter could be specified – but no one would understand the specification. The second order PLL modeled by Eq. (1) provides a compromise allowing both the bandwidth and peaking to be constrained by the specification.

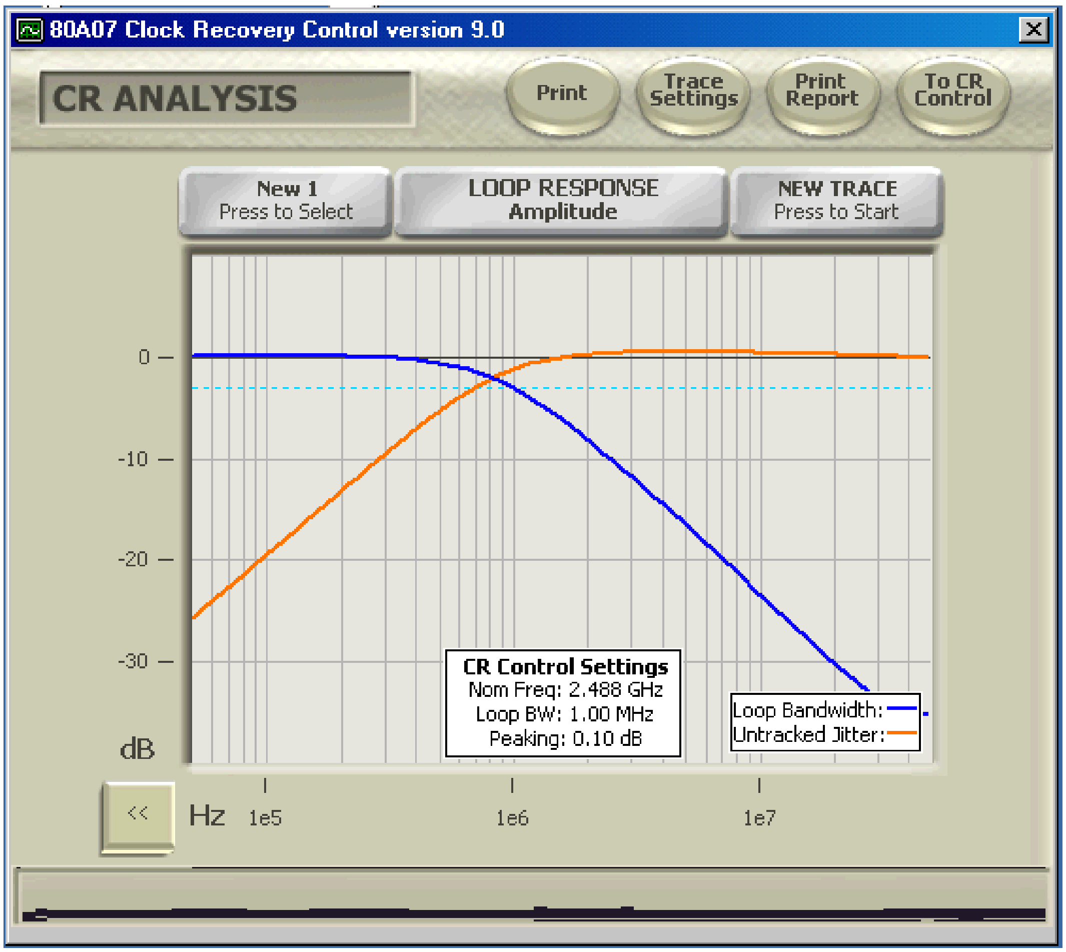


Figure 3: The frequency response of a 2nd order PLL.

Figure 3 shows the frequency response of a CDR based on a second order PLL. By reconstructing the clock from the data, jitter in the frequency band below the cutoff is common to both. When the clock signal and data signal jitter together in harmony, that jitter doesn't cause errors. The wider the CDR bandwidth, the smaller is the BER-relevant jitter-frequency band. Given the transfer function for a CDR, it's straightforward to implement in on a real time oscilloscope – a nice way to separate the relevant jitter and analyze the signal as it would be seen by a receiver's decision circuit.

Figure 4 shows two eye diagrams. The data signal in each is identical, the only difference is the bandwidth of the recovered clock used to trigger the oscilloscope. A very narrow bandwidth clock recovery shows a great deal of jitter. As the bandwidth of the recovered clock increases, the clock tracks jitter on the signal and, in Figure 4b there is almost no effective jitter.

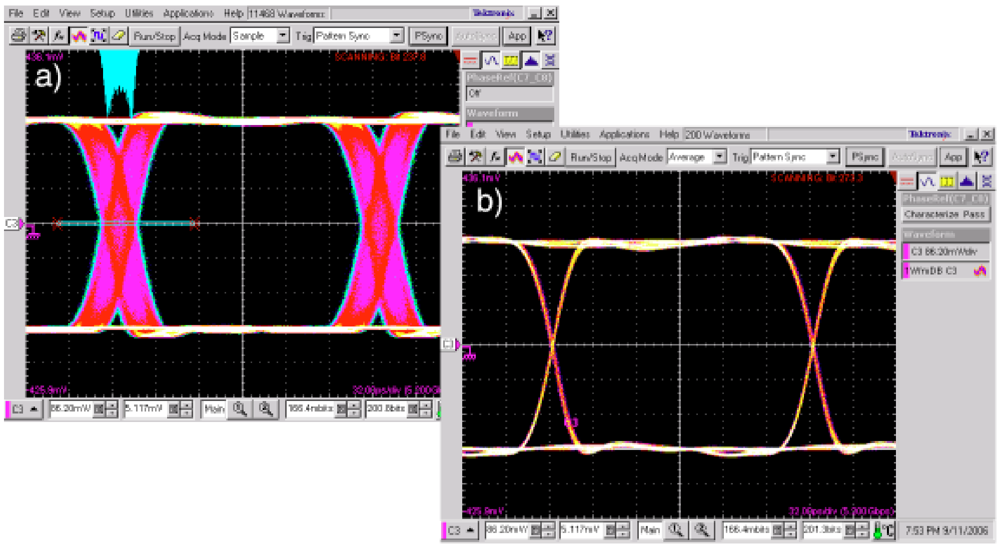


Figure 4: Eye diagrams of the same signal with (a) very small clock recovery bandwidths and (b) very wide clock recovery bandwidth.

Figure 5 is a block diagram of a PLL-based Clock Data Recovery (CDR) circuit. The data signal is split into two paths. One path guides the logic signal to the decision circuit (DC), and the other path goes to a PLL clock recovery circuit. The PLL is composed of a phase-frequency detector, a low-pass filter, and a voltage-controlled oscillator (VCO). The VCO locks to a phase and frequency in such a way as to minimize the difference in its phase and the phase of data-transitions. The recovered clock signal is then used as the time reference in the decision circuit.

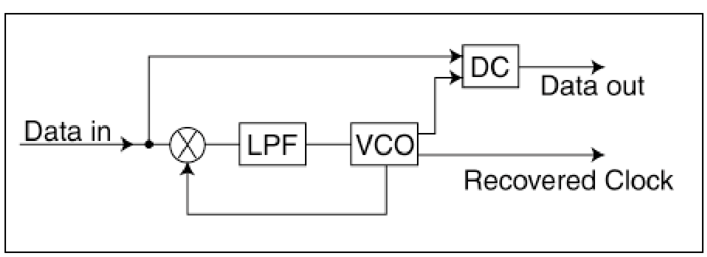


Figure 5: Phase-locked loop based Clock Data Recovery (CDR) circuit.

There are a couple of fairly obvious requirements of data signals in these embedded clock systems. In order to extract a clock signal of the appropriate frequency, the data signal cannot have extremely long runs of consecutive identical bits. Rather, the data must have plenty of transitions to drive the phase detector and tune the VCO. Second, the overall signal must be DC-balanced. If there aren't enough transitions, the VCO may not be able to lock on the data signal and, even if it locks, it will drift. To ensure enough transitions, data signals must be encoded. For example, "8B/10B encoding" is a common technique that encodes each 8 bits of data into a 10 bit symbol. The low 5 bits of data are encoded into 6 bits and the following 3 bits are encoded into 4 bits so that the transmitted signal is DC-balanced (same number of zeros as ones) and has no more than 4 consecutive identical bits.

Figure 6 is a block diagram of a Phase Interpolator (PI). PIs sample a data rate reference clock and compare it to the incoming data to fix the relative phase of the sampling point. Figure 6 is a simple configuration. A distributed 100 MHz reference clock is multiplied up to the data rate by a simple PLL. The data rate clock is sampled to determine that pair of reference phases that surround logic transitions in the data signal. The sampling point of the decision circuit is then determined by interpolating between those two reference phases. Since PIs require a data-rate reference clock, most designs include a distributed system reference clock.

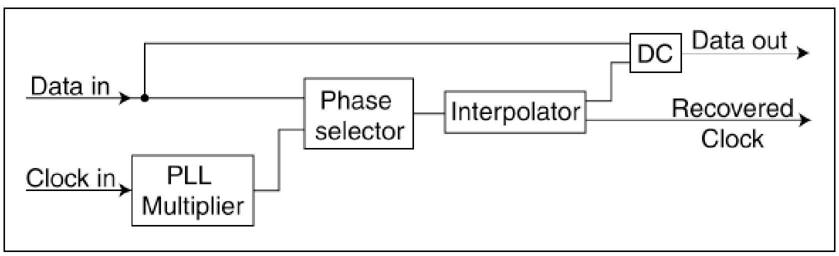


Figure 6: Phase Interpolator block diagram.

Delay

To reap the greatest rewards from the use of clock recovery, it is important that the relative delay of the data signal and the recovered clock be small. Remember, if the clock and data both have the same jitter at the decision circuit, then that jitter doesn't cause errors. But if the jitter on the clock differs from the jitter on the data, then we just end up with more jitter and a larger BER. If there is a delay between the reconstructed clock and the data, then the transitions don't share the same jitter. They're like two dancers listening to different music. If there's appreciable delay, rather than have the clock and data signals dance in harmony, the clock steps on the data's feet.

Actually, it's not quite as bad as I've made it out. Figure 3 shows that the low-frequency end of the jitter spectrum is rendered irrelevant to the BER in a CDR-based system. A delay of a few bit periods between the data and the clock may not have much effect on the BER since the tracked jitter is at a frequency much lower than the data rate. Still, we must be aware that delay can cause problems… especially in systems that use spread-spectrum clocks.

Spread Spectrum Clocking

Spread Spectrum Clocking (SSC) is a technique used to smear radiated energy into larger frequency bands through low frequency clock modulation. Smearing the energy reduces the peak energy at any given frequency, making it easier to comply with regulations that limit emitted power in narrow frequency bands. Typically, the clock is modulated at 33 kHz from zero to -0.5% of the nominal data rate in either a triangle wave or a "Hershey's Kiss" waveform.

At the receiver, the CDR bandwidth should generously cover the modulation bandwidth and the delay must be small enough that any SSC deviations that leak through are tracked. The problem introduced by delay is exacerbated at the transition from positive to negative going triangle FM. The peculiarities of SSC, how it's implemented, how it's affected by peaking and how it is best analyzed to reduce BER is a subject worthy of a paper all its own. In the next paper of this series, *Part 6: Reference Clock Jitter and Data Jitter*. we'll discuss more aspects of SSC.

Conclusion

There are many reasons for using clock recovery in serial-data systems and reducing the BER caused by jitter is a very good one.

In most existing specifications, and nearly every foreseeable specification, for technology over about 3 Gb/s, clock recovery elements are required to have a bandwidth above a specified minimum – to encourage data-jitter/clock-jitter tracking – and peaking below a specified maximum – to limit jitter amplification. For example, FibreChannel 4X, at 4.25 Gb/s requires a loop bandwidth of 2.55 MHz with maximum peaking of 0.3 dB. PCI Express Generation II (at 5 Gb/s) requires a loop bandwidth between 5 and 16 MHz with peaking no larger than 1 dB or loop bandwidth between 8 and 16 MHz with maximum peaking of 3 dB. In the latter case, the wider loop bandwidth affords greater peaking; separate specifications are provided for greater design flexibility.

In the part six of this series, we'll return to the topic of clock recovery as we discuss the role of the reference clock itself. Stay tuned…

Reference Clock Jitter and Data Jitter

Ransom Stephens, Ph.D.

Abstract:

Reference clock jitter sets the ultimate limit on the bit error ratio of a serial data system. The traditional techniques for quantifying jitter on clock signals don't help system designers estimate the clock's contribution to the total jitter defined at a bit error ratio. New techniques are being introduced to model how transmitters and receivers use reference clocks so that the system RJ and DJ attributable to the reference clock can be measured and maximums specified. This installment of Jitter 360 shows how reference clock jitter affects data jitter at each point in the system and describes techniques for its analysis.

We've seen that the different types of jitter are predominantly generated in specific components of the network. For example, Data Dependent Jitter (DDJ) is generated in the transmission path; Periodic Jitter (PJ) is caused by electromagnetic interference of one sort or another; and Random Jitter (RJ) is caused by "the combination of a huge number of sources, each of very small magnitude" (lifted from Part 3 of this series, *All About the Acronyms*). Most of those very small RJ sources originate in the oscillator that drives the reference clock of the transmitter.

As data rates get higher, jitter budgets get tighter, and we can no longer afford to lump jitter caused by the reference clock under "transmitter jitter." As rates surpass 2 Gb/s more and more standards bodies are providing explicit specifications for reference clock jitter and the new specifications are not easily related to the numbers recorded on most data sheets.

Figure 1 is a block diagram of a serial-data system. The *four* primary components are the transmitter, transmission path, receiver, and reference clock. The role of the reference clock is to define the timing of logic transitions at the transmitter and to provide a reference for setting the time-delay of the sampling point at the receiver. There are two types of clock systems. In embedded clock systems the only clock information available to the receiver is embedded in the data; in this case the dashed line in Figure 1 is not connected. In distributed clock systems, the dashed line in Figure 1 connects the reference clock to the receiver.

In this edition of Jitter 360, we'll see how RJ is generated in oscillators, how reference clock jitter propagates onto data jitter, and we'll have a look at how reference clocks can be evaluated in the emerging high rate serial-data standards.

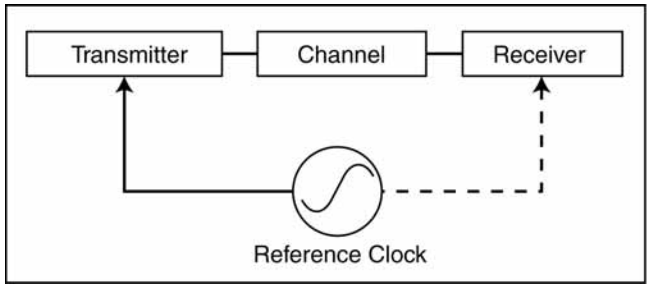
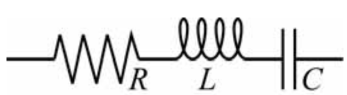


Figure 1: Serial-data straw diagram emphasizing the role of the reference clock.

Oscillators

The oscillator, whether based on an inexpensive LC circuit or a crystal, provides the periodic structure on which a digital system is based. The parameters used to describe oscillators are easily defined by using the example of an LRC circuit shown, Figure 2.



Resonant frequency: $\omega_R = \sqrt{\frac{1}{LC} - \left(R/2L\right)^2} \approx \frac{1}{\sqrt{LC}}$

Damping factor: $\zeta = R/2L$

Bandwidth: $\Delta f = \frac{\zeta}{\pi} = \frac{R}{2\pi L}$

Quality: $Q = \frac{f_R}{\Delta f} = \frac{1}{R}\sqrt{\frac{L}{C}}$

Figure 2: A simple oscillator and typical parameters.

The highest quality oscillators are based on crystals. Crystal oscillators, Figure 3, are composed of media, like quartz, which have easily excited piezoelectric properties. Stable oscillations are generated through a simple sequence based on the fact that any excitation will have its greatest response at the crystal's resonant frequency. First, random noise is applied to the crystal. The response is amplified and a feedback loop generates subsequent crystal response that builds at the resonant frequency. The output amplitude quickly stabilizes by virtue of either its own self-limiting nonlinearities or a reduction in amplifier gain. Figure 4 shows the spectrum of the settling behavior of an oscillator; notice the excitation of harmonic spurs.

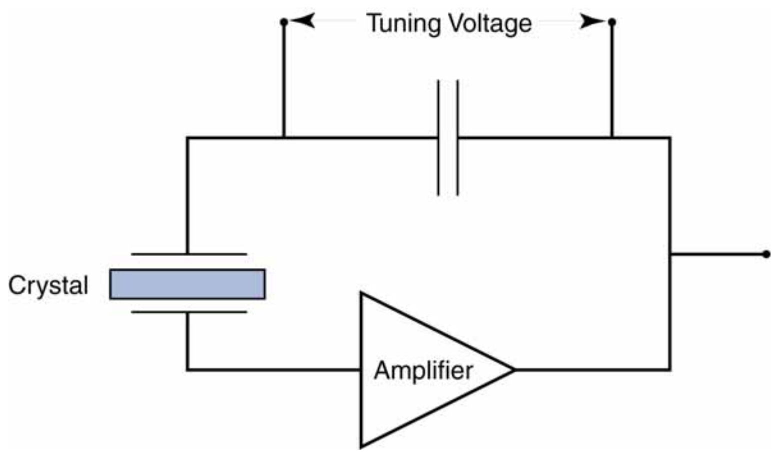


Figure 3: Crystal oscillator [circuit diagram]

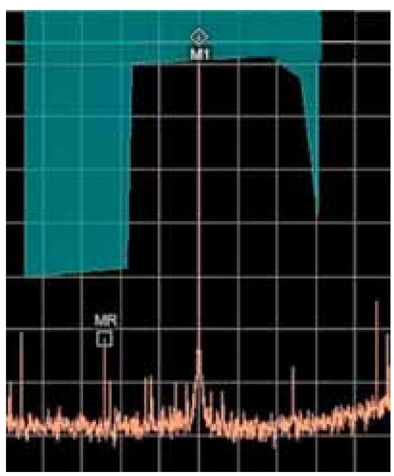


Figure 4: Spectrum of an oscillator as it settles.

The crystal's resonant frequency is determined primarily by its geometry – size, shape and density – which varies slightly with temperature, pressure, humidity, and applied voltage. Figure 5a shows an ideal oscillator spectrum and Figure 5b a spectrum including white noise. The nonzero width of the resonance is the dominant source of near-carrier noise. Tens of kHz above resonance, crystal vibrations can cause spurs at integer multiples of the difference between the vibration and carrier frequencies resulting in both periodic noise and jitter, Figure 5c. Far from the carrier the dominant noise effects are caused by inadequacies in the oscillator feedback loop such as impedance mismatches and power-supply feed-through that sits atop the white noise background.

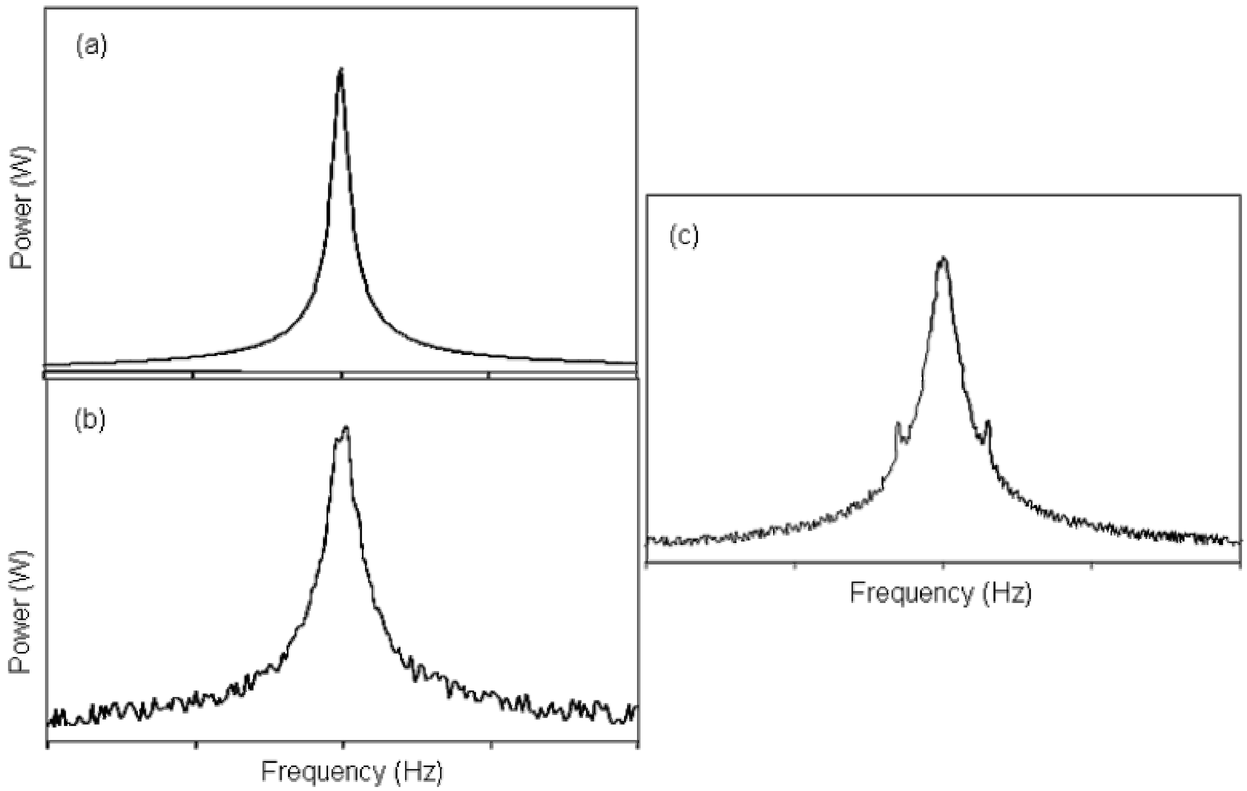


Figure 5: Oscillator or clock signal in the frequency domain, (a) an ideal Lorentzian resonance, (b) with white noise, (c) with spurs from crystal vibrations.

An ideal oscillator signal is a sinusoid with unvarying amplitude, A, and frequency, f:

$$\psi(t) = A \sin(2\pi ft)$$

a real oscillator has both amplitude noise, $\delta A(t)$ and phase noise, $\varphi(t)$

$$\psi(t) = (A + \delta A(t)) \sin(2\pi ft + \varphi(t)).$$

Random jitter from the oscillator is ultimately caused by a combination of phase noise and amplitude noise. Up to a factor of 2π, the phase noise, $\varphi(t)$ is precisely timing noise, the horizontal fluctuation of the signal. Amplitude noise, $\delta A(t)$, is the vertical fluctuation of the signal. Recall from Part 4: *Jitter Analysis in*

Systems With Crosstalk, that amplitude noise, $\delta A(t)$, introduces jitter as well as voltage noise as a consequence of the signal's nonzero rise/fall times.

The phase noise component of RJ originates in the continuum background of the frequency spectrum (a result exploited by the best jitter analysis algorithms which equate the rms jitter-frequency noise with the width of the RJ distribution, σ).

Of the different types of jitter, RJ is the most intractable from the design standpoint. Still, RJ can be reduced by understanding its primary sources. Excessively broad resonance structures can be caused by vibration, mechanical shock, or temperature variations which result in small random variations in the resonant frequency.

The Lorentzian shape of the resonance curve is caused by "flicker" – a process that appears in phenomena ranging from the behavior of stock prices to the behavior of stars – is related to the way that oscillations propagate through different regions of the crystal.

If the shape of the peak is asymmetric, flat-topped, or otherwise non-Lorentzian, then something fundamental is probably wrong with the crystal itself.

Noisy electronics in the oscillator circuit, usually from the amplifier, can cause the resonant curve to widen at its base.

White noise is a flat continuous background that is caused primarily by thermal noise in resistors, inductors, diodes and so forth.

Unfortunately, RJ cannot be eliminated by use of a limiting amplifier. Nor can RJ that is close to the carrier be reduced by filtering. Conversely, jitter that is far from the carrier, primarily caused by white noise, can be filtered.

At the Transmitter

To define the timing of logic transitions, the transmitter requires a data-rate clock signal. To convert the reference clock to a data-rate clock, transmitters typically use frequency multiplying Phase Locked Loops (PLLs). The important thing to remember is that frequency multiplication amplifies the phase noise by the *square* of the multiplication factor. That is, the phase noise increases by 20 dB for every 10 dB of multiplication.

The 20 dB increase in phase noise for each 10 dB of multiplication is the primary reason that high data rate standards address the jitter caused by reference clocks independent of other sources. There are two options: use higher quality oscillators or be very careful in how reference clocks are specified.

The frequency multiplying PLL, Figure 6a, introduces additional RJ, primarily from its Voltage Controlled Oscillator (VCO). Nonlinearities in the PLL are a principle cause of Duty Cycle Distortion (DCD).

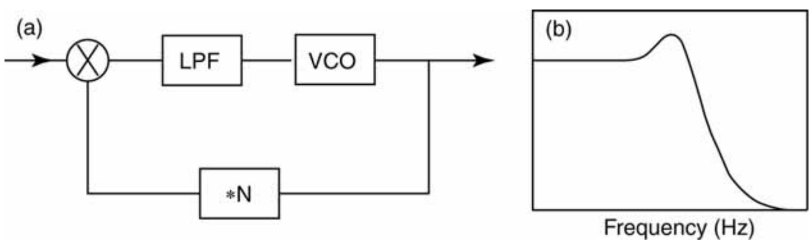


Figure 6: PLL Multipler (a) block diagram, (b) frequency response.

On the other hand, the PLL also acts as a filter. Figure 6b shows a typical PLL frequency response, just as in Part 5, *Clock Recovery in Serial-Data Systems*, the question begs: What part of the jitter frequency spectrum contributes to the bit error ratio? Remember, the only reason we care about jitter is that it causes errors.

At the Receiver

The reference clock plays different roles at the receiver. All imperfections on the reference clock that make it past the filtering effect of the transmitter PLL-multiplier are on the data recovered at the receiver.

As discussed at length in Part 5, *Clock Recovery in Serial-Data Systems*, the data-rate clock is recovered at the receiver either with or without the assistance of a distributed reference clock. The primary issue, from the BER standpoint, is whether or not the jitter on the recovered clock tracks the jitter on the data. Just as the bandwidth of the receiver's clock recovery circuit determines the frequencies of jitter that are tracked, the bandwidth of the transmitter's PLL-multiplier determines the reference clock jitter that makes it through to the data. The ideal situation is to have a narrow bandwidth multiplier at the transmitter and a wide bandwidth clock recovery at the receiver. This way, little jitter makes it from the clock to the data and the jitter that does make it is also on the recovered clock so that the sampling point jitters in phase with the data.

There is an added complication when the reference clock is distributed to the receiver. For example, in the Phase Interpolator (PI) based clock recovery circuit, Figure 7, there is another PLL multiplier. The frequency response of the receiver's multiplier is likely to have a different frequency response than the one at the transmitter. Mismatched multipliers have the potential to introduce small amounts of jitter to the recovered clock that are not on the data.

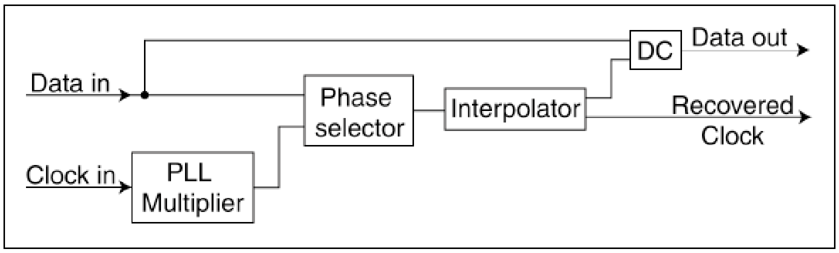


Figure 7: Phase interpolator based receiver.

Analyzing Clock Jitter

It is difficult to translate the quantities that traditionally appear on clock data sheets – peak-to-peak phase jitter, period jitter, and cycle-to-cycle jitter – to the only relevant question: What impact does the clock have on the system Bit Error Ratio?

The most useful traditional clock evaluation is the phase noise spectrum. If the bandwidth of the transmitter clock multiplier and the bandwidth of the receiver clock recovery circuit are both known, then at least we can determine whether the bulk of the phase noise can affect the BER. This is the purpose of phase noise mask tests.

The increased use of RJ and DJ to estimate TJ(BER) encourage clock vendors to quote RJ and DJ under the extreme transmitter bandwidths of both the transmitter-multiplier and receiver clock-recovery specified for different technologies. This way, as shown in Part 1, *The Meaning of Total Jitter*, the system TJ(BER) budget can be estimated for different transmitter/channel/receiver/reference clock combinations.

Real time oscilloscopes and spectrum analyzers can be used to assemble data from a clock signal to evaluate the relevant RJ and DJ by applying the transfer functions of the worst-case transmitter-multiplier/receiver-clock-recovery combinations described by the standard.

The trace of a real time oscilloscope is used to determine a data set of the timing of logic transitions, $\{t_n\}$. Numerical techniques from Digital Signal Processing (DSP) are then applied to the data to simulate the transmitter-multiplier and receiver clock-recovery responses. Similarly, the frequency spectrum from a real time spectrum analyzer can be processed by applying the same techniques in the frequency domain. In either case, the usual jitter analysis techniques can then be applied to the simulation to determine the effective worst-case RJ and DJ that the clock may contribute to a system. With the RJ and DJ in hand, the dual-Dirac model (see Part 2, *What the Dual-Dirac Model is and What it is Not*) can be used to estimate the reference clock contribution to the overall system BER,

$$\mathrm{TJ(BER)} = 2Q_{BER}\times\mathrm{RJ} + \mathrm{DJ}.$$

Conclusion

Reference clock jitter plays such a key role in the ultimate BER of serial-data systems that the worst case RJ/DJ for each application should be reported on clock data sheets. This would allow design engineers to assemble systems with predictable TJ(BER).

The reference clock itself is a primary source of RJ. The shape of the oscillator resonance and the phase noise spectrum can be used for diagnostics that can reduce RJ. The reference clock can also contribute PJ and DCD.

The transmitter usually multiplies a low rate reference clock up to the data rate. The PLL multiplier includes a VCO that contributes additional jitter and the multiplication process itself amplifies the jitter by the square of the multiplication factor.

The combination of a narrow bandwidth multiplier at the transmitter and wide bandwidth clock recovery at the receiver produces a system with the lowest BER.

Reference clocks can be evaluated using DSP techniques on data provided by real-time oscilloscopes or spectrum analyzers to simulate the effect of different transmitter PLL-multiplier/receiver-clock-recovery schemes resulting in RJ and DJ estimates that can be used to estimate the overall system TJ(BER).