Jitter separation: Where science meets art

Brig Asay - November 15, 2013

Jitter is one of the most widely used terms amongst engineers who design digital data links. Whether you design a new board or validate an old board, you'll have to confront jitter and its components. The problem is how to separate jitter into random and deterministic components. Knowing those components can uncover the sources of jitter.

Jitter is simply the "wiggle" or time variation of digital clock's edge that each transition makes. That difference in timing is called TIE (time interval error). I'll use that as the basis for this discussion.

In an ideal world, rise and fall times would be infinitely fast and every edge of a digital stream would consistently fall the exact same time away from the last. The end result would be no jitter, no wiggling, and no eyes diagrams that suffer from being closed or close to closed. Unfortunately it is not a perfect world, and what once was conceived as an infinitely fast edge, becomes slower as tradeoffs occur as data rates increase.

A signal often must travel through inexpensive PCB material. That, combined with other factors, causes a signal to lose amplitude. Plus, a signal can couple with other signals, causing it to move from a perfectly timed edge to an edge that is now moving from the clock. That's the "wiggling" that Figure 1 shows. Because of these real-world conditions, digital eye diagrams will close, making it harder for a receiver to distinguish between a logic 1 and a logic 0.

Figure 1. Jitter occurs because the time between edges in a digital data stream will vary.

Jitter separation lets you learn if the components of jitter are random or deterministic. That is, if they are caused by crosstalk, channel loss, or some other phenomenon. Separating jitter enables engineers to better understand the systematic problems of their devices and to quickly find solutions to adjust for any errors in them. The jitter-separation concept seems simple enough. In an ideal world, all jitter separation techniques would work the same and give exactly the same answers. All the tools are looking at the same jitter.
Unfortunately this is not always the case. In fact, "answers" for different jitter separations can vary widely. The problem has become prevalent enough, that compliance tools can ensure that all designers use standardized separation techniques. What seems like a simple problem that can solved fairly simply is complicated by the fact the separation tool's answers will vary by test-and-measurement vendor and instrument.

So how does an engineer know which answer is correct? This is where science meets art and the user must use the tools available to him (the science) to decide which answer best represents what he or she is debugging (the art).

**Jitter separation challenges**

One reason that answers vary from instrument to instrument is that there is often a limited amount of information available to do the separation. A reasonable analogy is that of solving a linear system of equations. It's well known you must have as many independent equations as there are unknowns. If there are too few equations, or if the equations aren't all linearly independent, then you can't get a unique answer. One way to get around this problem is to impose additional constraints or assumptions onto the system, which acts like adding more independent equations. This new system is then uniquely solvable. Solving for the unknown values of the different jitter components is a similar problem, and typically software packages must impose additional constraints, or models, in order to obtain a unique, repeatable answer.

The most fundamental separation is between components that are deterministic (deterministic jitter, or DJ), and those that are random (random jitter, or RJ). Sometimes these categories are defined as components that have bounded histograms and those that have unbounded histograms, but are still referred to as DJ and RJ. The concept of determining which jitter components are random in versus deterministic may seem simple, but the actual process is difficult. To separate jitter, an algorithm typically will need to not only separate random from deterministic jitter, but also identify classes of deterministic and random jitter. I'll limit my focus primarily to the separation of deterministic and random jitter.

I've made the assumption that the jitter-separation algorithm used here is based on TIE. Clearly, there are more ways to measure the total jitter of a signal. TIE is the variance of an edge from a clock or data signal; from a pre-determined clock.

In real-time oscilloscopes, the clock can be a software recovered clock or an explicit clock (meaning the actual system clock). The clock is set to a specific rate and being "ideal" will place its edges the same distance from each other across the entire data record. By tracking every edge in a data record and its variation from the ideal clock, the TIE builds up an entire base of records which forms the base of jitter separation. Keep in mind that in the last few years, oscilloscopes have added deep memory depth, making it possible to capture millions of histogram bins in a single acquisition. **Figure 2** shows the capture of a timing waveform with the histogram of the TIE record.
Separation of Data Dependent Jitter Before separating deterministic jitter from random jitter, it's generally easier and more accurate to remove any DDJ (data-dependent jitter) first. DDJ includes DCD (duty cycle distortion) and ISI (intersymbol interference). The reason for this is that measuring DDJ is often relatively straightforward.

Transform approaches: The spectral method

Of the various methods for separating deterministic jitter from random, or RJ/DJ separation, conceptually the spectral method is perhaps the simplest to understand. It begins by computing the FFT (fast-Fourier transform) of the TIE record (Figure 3).
Figure 3. Oscilloscope software can show a spectral view of jitter components.

The FFT shows the jitter components in the frequency domain. The spectral algorithm then chooses a threshold, which is typically an averaging of the noise floor, and the algorithm looks for peaks that are above the threshold. The peaks are considered to be the periodic components and below the threshold are the random components. The amount of RJ (random jitter) can then be quantified as the RMS value of all the random components in the spectrum (Figure 4).

Figure 4. Jitter-separation software shows the threshold chosen for jitter separation.

As the spectral method is simple in concept, it relies on some basic assumptions. It assumes that all DJ can be described as periodic in nature, and therefore everything that falls into the noise floor of the FFT must be random. Unfortunately, that's not always the case. The spectral method tends to work very well in the case where no crosstalk is present because these assumptions are typically
true. When crosstalk occurs, however, the spectral method will often confuse crosstalk components as random in nature as opposed to their true deterministic nature. The end result is that total jitter can be over reported in the spectral method.

**Histogram approaches: Deconvolution**

Deconvolution is a method that separates the clear image from the random noise and makes it possible for them to distinguish individual components in just about anything. The same idea applies in jitter separation. When looking at jitter components, each component has its own individual histogram. For instance, if you could look at them separately, the periodic jitter histogram could look like **Figure 5** and the random jitter looks like **Figure 6**.

![histogram of timing error](image)

**Figure 5.** A histogram of periodic jitter looks Gaussian in shape.
The histogram of the TIE record is the convolution of the periodic and random histograms. The challenge is separating this TIE histogram into individual components. This is what deconvolution attempts to do. When there are more than two components of jitter, the TIE histogram is the convolution of all the individual components, but we are still primarily interested in removing, or deconvolving, out the random component from the rest (that is, doing RJ/DJ separation). Assumptions must often be added to make the problem solvable, and one that is always made here is that the RJ histogram can be modeled by a Gaussian distribution. The effect of RJ can then be quantified as the standard deviation of that distribution. When there is no crosstalk present, this number should match the RMS value returned from the spectral method.

One method that is becoming increasingly popular for jitter separation is the tail-fit method. The tail-fitting idea is based on the observation that as you look further and further out on the tail of the jitter histogram, the more the shape of the TJ histogram approaches the shape of the RJ histogram. This observation holds up mathematically, where you can show that the TJ shape does asymptotically converge to the RJ shape. Because of this phenomenon, you can get an estimate of the RJ histogram by fitting a Gaussian function to the "tail portion" of the histogram. The RJ value is then computed as just the standard deviation of the fit Gaussian. The location of the Gaussian, its mean, is an estimate of the maximum value of deterministic jitter. Thus, by fitting both the left and right tails, an estimate of the peak-to-peak value of DJ is simply the distance between the two peaks of the Gaussians.

The difficulty of tail-fit is finding the right place to fit the curve to the tail and yet be limited in the amount of data that can be used. As an algorithm looks to the middle of the histogram, the data is very repeatable because these are the bins where most of the data points fall, and thus the confidence or precision of the algorithm is high; however, the accuracy is low. As an algorithm goes to the end of the tail, the precision gets lower because fewer data points fall in these bins, but the shape of the TJ histogram more closely matches the RJ histogram so accuracy improves.

The problem is that it’s very difficult to fit a curve at the end of a tail with limited data. To truly find a good fit, a tail-fit algorithm may require billions of data points, which isn’t practical in today’s test-
and-measurement equipment because it this takes time and processing power, both of which are at a premium. As a result, a tail-fit algorithm must balance the difficult tradeoff of precision/confidence versus accuracy. Again, the biggest disadvantage of the tail-fit method is that it requires large amounts of data to get the best answer. Thus, an answer at shallow memory depths may have a tail that has incorrect answers in its jitter separation.

The jitter-separation techniques presented here represent only a small portion of the many methods that separate jitter into random and deterministic components. There are polynomial, variations of tail-fit, and many others. Simply by searching for jitter separation in a web search engine, literally hundreds of techniques can be found. I’ve only mentioned a couple of separation methods, but there are many more. So which is correct? That’s where the science meets art. All techniques are subject to caveats and cases that can make them less accurate.

As someone that must evaluate jitter as part of your job, you need to know what the limitations are and what the signal that is being analyzed needs. Knowing this information, a you can properly identify the correct separation method, thus ensuring that the designs are correct.

Also see
- Jitter's faces: Random, periodic, and ISI
- The many faces of jitter
- Why does an eye diagram not correlate to a bathtub curve?
- Random and deterministic jitter
- How to measure display jitter
- Jitter and timing analysis in the presence of crosstalk
- Which jitter measurement is correct?
- The philosophy of jitter
- The basics of digital signal spectra
- Jitter: Measurement References Matter
- The scopes trial