APPLICATION NOTE

Total Jitter Measurement at Low Probability Levels, Using Optimized BERT Scan Method

KEYSIGHT

Introduction

Jitter, in the context of high-speed digital data transmission, is usually defined as the deviation of the decision threshold crossing time of a digital signal from its ideal value. Jitter describes a timing uncertainty, and therefore has to be considered when we look at the timing budget of a design. In one sense, jitter is just another component that makes part of the bit period unusable for sampling, just like setup–and hold times. However, unlike setup- and hold times that are usually thoroughly specified for logic families and can be taken from data sheets, jitter is a function of the design and has to be measured.

Jitter is caused by a great variety of processes, for example, crosstalk, power supply noise, bandwidth limitations, etc. Therefore, there are many different categories of jitter. Depending on whom you ask, jitter is categorized as bounded and unbounded, correlated and uncorrelated, data-dependent and non-data-dependent, random and deterministic, periodic, and non-periodic, to just name the most common ones. What is undisputed however is that all the different kinds of jitter add up to a quantity that is called total jitter (*TJ*). This is the quantity that you have to account for in your design, and this paper describes one way to measure it quickly and accurately.

Our approach uses a bit error ratio tester (BERT), the only instrument available today that can directly measure *TJ* peak-to-peak values. Total jitter measurement methods using BERT scans have been available for a long while, however, the long measurement times required for a full scan limited the use to characterization applications where direct measurements with good accuracy are required. By careful use of statistics and probability theory, we were able to reduce measurement times by more than one order of magnitude.

This paper is organized into four sections: in the first and second sections, we recall the basics of jitter and bit error ratio analysis and introduce the necessary probability theory. Section three shows how full bathtub measurements are measured conventionally, and one common optimization. In the last section, we present the bracketing approach to total jitter measurement and show two example implementations.

Jitter

Analog and digital definitions of jitter

There are two definitions of jitter, an analog and a digital one. In the analog world, jitter is also known as phase noise, and is defined as a phase offset that continually changes the timing of a signal:

$S(t) = P(t + \phi(t))$

where $S(t)$ is the jittered signal waveform, $P(t)$ is the undistorted waveform, and $\phi(t)$ is the phase offset or phase noise. This definition is most useful in the analysis of analog waveforms like clock signals and is frequently used to express the quality of oscillators.

In the digital world, we're looking only at the 1/0 and 0/1 transitions of the signal, and jitter is therefore only defined when such a transition occurs. The jittered digital signal can be written as

$t_n = T_n + \phi_n$

where t_n is the time when the n^{th} transition occurred, is the ideal timing value for the n^{th} transition, and ϕ_n is the time offset of this transition, also known as the timing jitter.

Note that there are many possible choices for T_n: physical quantities such as threshold crossings of a reference clock or a recovered clock, or arithmetic quantities, like multiples of the nominal bit duration at the given data rate. This is something to always keep in mind when making jitter measurements since results can vary dramatically with the choice of the reference. A drastic example is spread spectrum clocking (*SSC*), where low-frequency jitter is deliberately introduced to keep emissions in regulated frequency bands below the allowed maximum; a jitter measurement that uses a clean, non-SSC clock as the reference will show the desired SSC as jitter.

Jitter categories

Every high-speed digital link in a design is subject to many jitter sources, each with different root causes, characteristics, and possible design solutions. Examples are:

- **Inter symbol interference (ISI),** which is caused by attenuation and bandwidth limitations of a transmission structure. ISI is a function of the data rate, board layout and material, and the data pattern sent over the link. Most multi-gigabit designs today use transmitter pre-emphasis or receiver equalization to deal with this and also limit the maximum run-length of continuous 1s or 0s by the use of 8b/10b coding or the like.
- **Switching power supply crosstalk**, which is caused by an improperly decoupled power distribution on a PCB or inside of a chip package. The resulting jitter is periodic with a frequency that is typically many orders of magnitude lower than the data rate.
- **Noise**, either thermal noise in the transmitter and receiver chips, or other noise coupled into the transmission structure. Jitter caused by noise usually has a very wide bandwidth.

One widely accepted classification system divides total jitter (*TJ*) into random jitter (*RJ*) and deterministic jitter (*DJ*); *DJ* is then further divided into correlated data-dependent jitter (*DDJ*) and uncorrelated periodic jitter (*PJ*). Detailed descriptions of jitter categories and separation techniques can be found in Figures 1, 2, and 3. For most of the analysis in this paper, we will use a *TJ* mixture that consists of a random part and a periodic part only. The major difficulty of *TJ* measurement stems from the unbounded nature of *RJ*, and we wouldn't gain any insight if we added a correlated term to the deterministic jitter.

Jitter as a time waveform

The analog jitter definition as a continuous function of time is a vivid one, so we're going to use it in some places in this paper, even though we're interested in digital jitter analysis mostly. We don't lose anything by that since we can simply sample it at regular intervals later. The total jitter continuous-time waveform is the sum of all independent jitter component time waveforms:

$$J(t) = PJ(t) + RJ(t) + DDJ(t) + ...$$

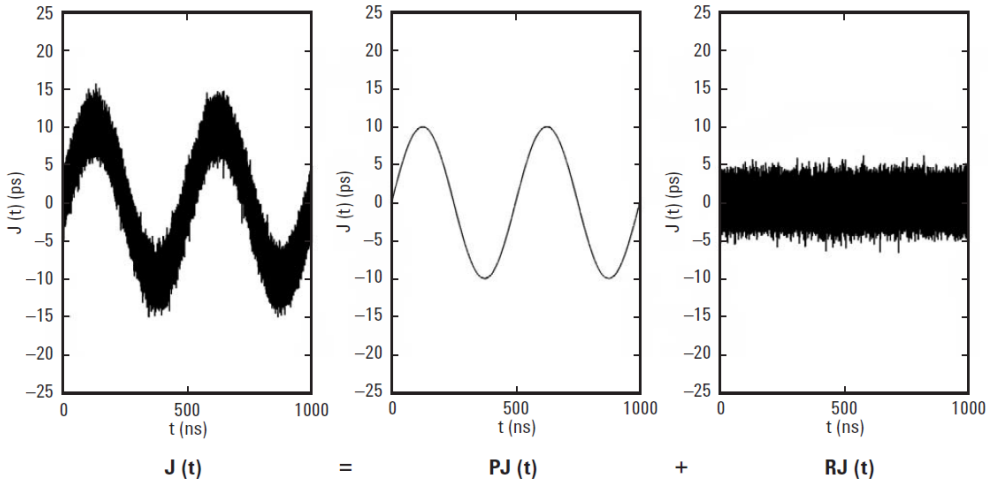

Figure 1. The total jitter time waveform is the sum of the individual components

Figure 1 shows an example for a 10.0 ps sinusoidal *PJ* with 2.0 MHz and a 1.5 ps RMS *RJ*, over an observation period of 1 µs. Since no instrument exists today that can directly measure the jitter time waveform, we're using simulated data: a pure sine wave for *PJ(t)*, and normally distributed random numbers for *RJ(t)*.

In order to assemble the timing budget for a design, we need total jitter as a single number in the dimension of time(s). This is usually a peak-to-peak value, that is, the maximum value minus the minimum value:

$TJ_{PkPk} = \max(J(t)) - \min(J(t))$

The total jitter peak-to-peak for the example in Figure 1 turned out to be about 31 ps. But unfortunately, this result is not a useful TJ_{PkPk} value, because the *RJ* term describes an unbounded random process. This means that the observed min and max values and thus the TJ_{PkPk} value get larger as we measure for a longer period of time. In the limit, the minimum is minus infinity, and the maximum plus infinity, and TJ_{PkPk} thus infinity (twice).

Probability density functions

The usual way to deal with such a problem is to make use of the fact that the individual terms are independent.

Thus, we can build histograms or calculate the probability density function (PDF) for the individual jitter components, and use a convolution operation to calculate the total jitter PDF:

$J(x) = PJ(x) * RJ(x) * DDJ(x) *...$

The *TJ* peak-to-peak value is then the maximum non-zero probability PDF value minus the minimum non-zero PDF value. Figure 2 shows the PDFs for the example above; TJ_{PkPk} is 31 ps, exactly the same value that we got from the time waveform.

The PDF has two advantages over the time waveform: first, it can be measured directly on many different types of test equipment, for example, sampling oscilloscopes, real-time oscilloscopes, and time interval analyzers. Second, the PDF of a Gaussian process is well known.

Thus, we can calculate the total jitter PDF, if we know the *RJ* RMS value and the PDFs of all other jitter components. The resulting *TJ* PDF however is still non-zero over the whole definition range, leading to the same infinite and thus meaningless TJ_{PkPk} reading that we got earlier. However, since we're dealing with probabilities anyway, it's easier to define the TJ_{PkPk} values as a function of some sort of probability level.

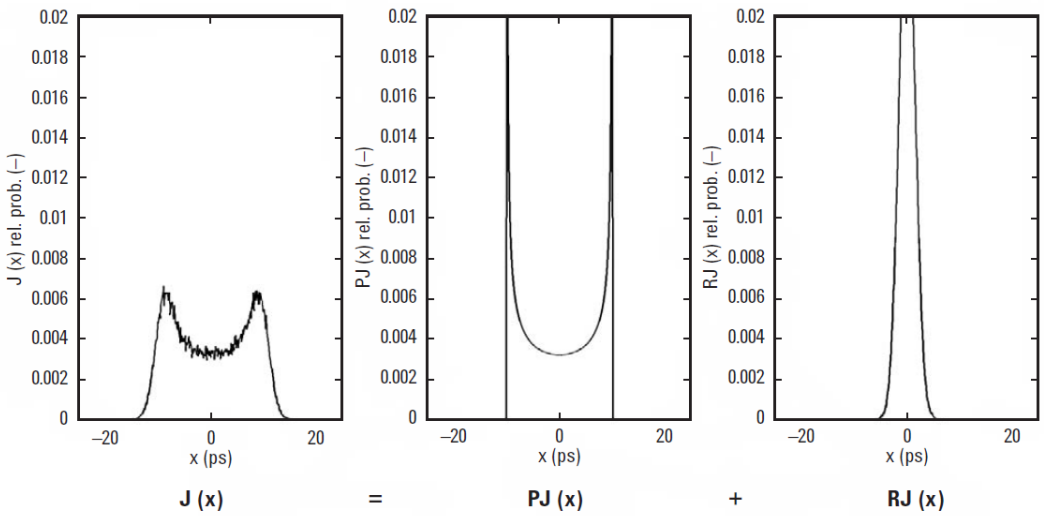

Figure 2. The total jitter PDF is the convolution of the individual component's PDFs

Cumulative probability density functions

Expressing *TJ* peak-to-peak as a function of a probability level can be done easily once we construct a cumulative probability density function (CDF), by integrating the PDF:

$$CDF(t) = \int_{-\infty}^{t} PDF(x)dx$$

The CDF tells us for each time value the probability that the transition happened earlier. *TJ* peak-to-peak for a probability level of y is then the time value where CDF = 1 – y/2, minus the time value where CDF = y/2. Figure 3 shows the *TJ* CDF for the example above. The *TJ* peak-to-peak value that includes all but $1e^{-3}$ of the population is 28.51 ps, while TJ_{PkPk} for $1e^{-4}$ is 30.13.

One important thing to note from Figure 3 is the fact that we don't have any CDF values lower than $1e^{-5}$. This is because the plots were generated using 100,000 random floating-point values on a computer, and the lowest possible non-zero PDF and thus CDF value, in this case, is $1e^{-5}$. From this observation immediately follows that we need at least 2/y samples if we want to directly calculate TJ_{PkPk} at probability level y from a measured PDF. For example, at the probability level of $1e^{-12}$ that is required by many standards, at minimum $2e^{12}$ samples need to be acquired.

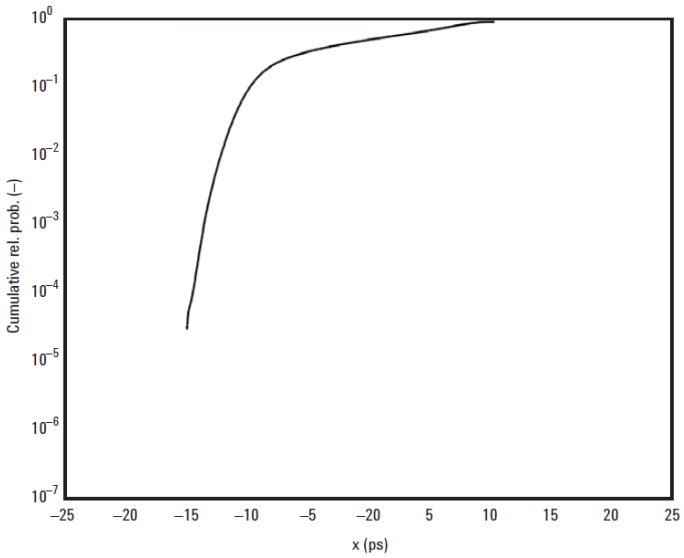

Figure 3. The total jitter CDF, using a log scale for probability axis

Total jitter calculation from measured PDFs

As we've shown in the last section, many samples are needed in order to directly calculate the TJ_{PkPk} value at low probability levels. Unfortunately, all test equipment existing today that can assemble PDFs from direct measurements or samples suffers from low sampling rates, and real-time oscilloscopes that have high sampling rates need many sampling passes because of memory limitations. At a sampling rate of 100 kHz, the acquisition of $2e^{12}$ samples takes more than 230 days, so even an improvement in sampling rate of a factor of 100 would still make the direct measurement impractical. Because of this limitation, *TJ* readings on oscilloscopes and time interval analyzers are usually extrapolated from a PDF that was measured using a much lower number of samples. Many assumptions are made in the extrapolations and, while they give estimates of *TJ* in seconds, the different techniques frequently give wildly inconsistent results. When there is no substitute for a genuine measurement without any assumptions it's useful to remember that *TJ* can only be measured on a BERT.

Bit Error Ratio

Definition

The quality of a digital transmission system can be expressed most naturally in terms of how many bits out of a transmitted sequence were received in error. This is usually done on a Bit Error Ratio Tester (BERT), a piece of test equipment that consists of a reference quality receiver, expected data generation, a digital compare mechanism, and counters for received bits and errors. During the test, received bits are compared to the respective expected bits; each compares operation increments the number of compared bits counter, and the error counter is incremented for every failed compare.

The main result of a test is the bit error ratio (*BER*), which is defined as

$$\mathrm{BER} = \frac{N_{Err}}{N_{Bits}}$$

where N_{Err} is the number of errors and N_{Bits} the number of bits. This equation is used both for measured and actual *BER* values; the measured value approaches the actual *BER* in the limit as $N_{Bits} \rightarrow \infty$.

Bit error ratio measurement as a binomial process

The bit error ratio measurement is a perfect example of a binomial process: for each bit that is received in the BERT's error detector and compared against the expected data, there are exactly two possible outcomes: either the bit was received in error, or not. If we assume that the errors observed during a *BER* measurement are independent of each other, and if the conditions don't change over time, we can model a *BER* measurement using the binomial distribution:

$$P_{Binomial}(N_{Err}, N_{Bits}, BER) = \frac{N_{Bits}!}{(N_{Bits} - N_{Err})!} \cdot BER^{N_{Bits}} \cdot (1 - BER)^{N_{Bits} - N_{Err}}$$

In most practical cases, we're dealing with low bit error ratios and high numbers of compared bits. For $BER < 1e^{-4}$ and $N_{Bits} > 100{,}000$, the Poisson distribution approximates the binomial distribution within double precision numerical accuracy. It is considerably easier to evaluate and has only one governing parameter μ, which is the average number of errors we expect to observe for a given *BER* and N_{Bits}:

$\mu = BER \cdot N_{Bits}$

The PDF of a Poisson distribution for a *BER* measurement experiment is then

$$P_{Poisson}(N_{Err}, \mu) = e^{-\mu} \cdot \frac{1}{N_{Err}!} \cdot \mu^{N_{Err}}$$

where N_{Err} must be an integer, while μ can be any non-negative real number.

Accuracy of bit error ratio measurements

Knowledge of the probability density function that describes *BER* measurements allows us to come up with accuracy estimates. As an example, we compare three *BER* measurements on a system with an actual *BER* of $1e^{-12}$, and vary only the number of compared bits:

- $N_{Bits} = 1e^{12}$ (μ = 1). The probability of getting exactly one error in the test (which is equivalent to measuring the exact *BER* of $1e^{-12}$) is 0.3679
- $N_{Bits} = 1e^{13}$ (μ = 10). The probability of getting 10 errors (*BER* = $1e^{-12}$) is only 0.1215.
- $N_{Bits} = 1e^{14}$ (μ = 100). The probability of getting 100 errors (*BER* = $1e^{-12}$) is even less, namely 0.0399.

Does this mean that the results get better if fewer bits are compared? Exactly the contrary is the case. Figure 4 shows the discrete PDFs for the μ = 1 and μ = 10 cases. The absolute probability values are indeed higher for μ = 1, but only because there are fewer possible outcomes.

Note how, for μ = 1, the probability of observing zero errors (*BER* = 0) is exactly the same as for observing one error (*BER* = $1e^{-12}$). For μ = 10 however, the probability of no error is almost zero ($4.54e^{-5}$).

Likewise, the probability of observing two errors for μ = 1 (*BER* = $2e^{-12}$, double the actual value) is 0.1839, but the probability of observing 20 errors in the case of μ = 10 (which is the same bit error ratio) is only 0.0019.

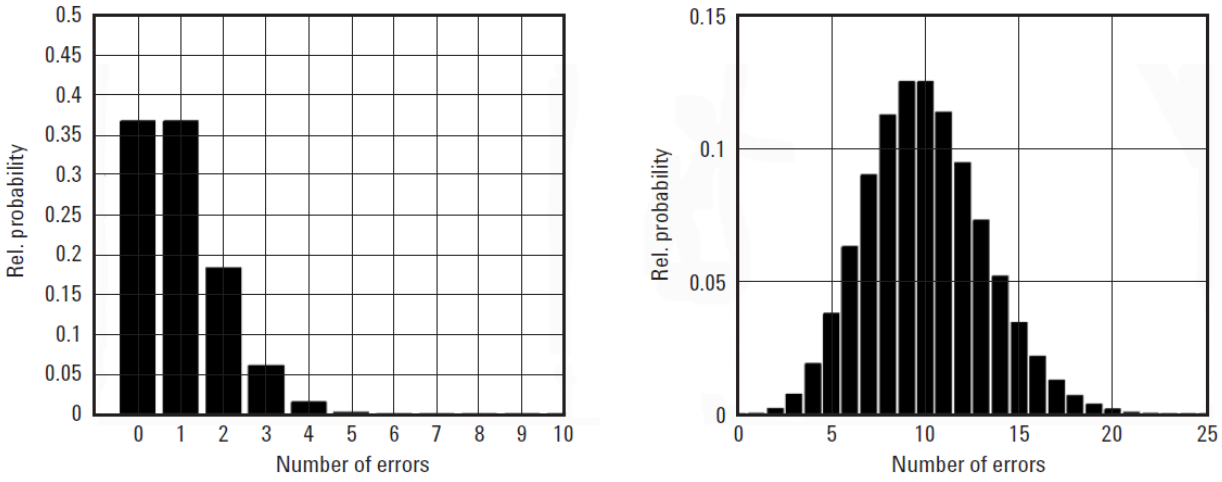

Figure 4. Probability density functions for a Poisson distribution with = 1 (left) and = 10 (right)

If we repeat the same measurement over and over again, the observed N_{Err} values are distributed with a standard deviation of $\sqrt{\mu}$. And while this value increases with μ, the spread in terms of *BER* decreases (remember that *BER* equals μ divided by N_{Err}). In Figure 5, we plotted the PDFs for three values of μ (1, 10, 100), using the *BER* value rather than N_{Err} on the x-axis, and normalized the probability values to unity. The distribution of the measured bit error ratio gets narrower if we increase μ, which is equivalent to increasing the number of compared bits if the actual *BER* is constant.

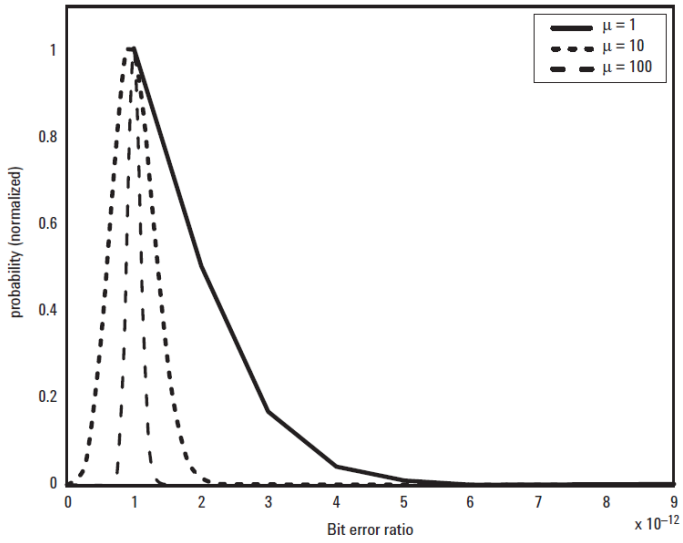

Figure 5: Normalized probability density functions for Poisson distributions with μ = 1, μ = 10, and μ = 100, in terms of bit error ratio rather than N_{Err}

So far, we assumed that the actual *BER* is known, and derived accuracy estimates based on this knowledge. In a real-life situation, however, the actual *BER* is obviously unknown. So how can we get accuracy estimates after a measurement, when only N_{Bits} and N_{Err} are known? Fortunately, we can simply use N_{Err} as an estimator for μ and derive the standard deviation for the measurement from there.

Confidence levels on bit error ratio measurements

Quite often, we don't need to measure the exact *BER* but can stop the measurement if we are certain that the *BER* is above or below a limit. In the jitter tolerance test, for example, we need to assert that the device under test operates with a *BER* better than for example $1e^{-12}$; whether the true *BER* is $1.1e^{-13}$ or $2.7e^{-15}$ is irrelevant.

Our confidence in such an assertion can be expressed in terms of a confidence level. A confidence level sets a limit on the maximum or minimum of the true value of a quantity, based on a measurement. If we compare $3.0e^{12}$ bits without getting errors, we can say that the *BER* is below $1e^{12}$ at the 95% confidence level. This example demonstrates the power of this approach: we measured a *BER* of zero but using the number of compared bits and some sensible assumptions, we can be reasonably sure that the *BER* is lower than $1e^{-12}$. How can those confidence levels be derived?

Let's make an example: if we compare $5e^{12}$ bits and get a single error, how confident can we be that the *BER* is < $1e^{-12}$? The measured *BER*, in this case, is $0.2e^{-12}$, which indicates that the *BER* is indeed lower than $1e^{-12}$, but the measurement was made with a large uncertainty. Using the Poisson distribution, we can calculate the probabilities of observing zero or one error in $5e^{12}$ bits, assuming the *BER* is exactly $1e^{12}$. We evaluate $P(0,5) = 0.0067$ and $P(1,5) = 0.0337$, so the probability of observing zero or one error in $5e^{12}$ bits if the *BER* is $1e^{-12}$ equals 4.04%. Our confidence that the *BER* is below $1e^{-12}$ is then 95.96%, and we've thus set an upper limit on the bit error ratio.

Table 1 shows statistics for upper and lower limits on *BER* at a confidence level of 95%. In order to set an upper limit, we need to transmit at least *y* bits with no more than *x* errors. In order to set a lower limit, we need to detect at least *x* errors in no more than *y* transmitted bits. The numbers for the upper limits were derived in analogy to the example above, by solving

$$\sum_{0}^{N_{Err}} P(N_{Err},\mu) = (1 - 0.95)$$

for μ; the number of bits required for a given confidence level of 95% is then μ divided by the target *BER*. Similarly, the numbers for lower limits can be derived by solving

$$\sum_{0}^{N_{Err}} P(N_{Err},\mu) = 0.95$$

Unfortunately, a closed solution for these equations doesn't exist, but they can be solved numerically. A detailed description of an alternative method on how to compute confidence levels using a Bayesian technique can be found in Figure 4.

Table 1. Statistics for lower and upper limits on *BER* of 10^{-12}, on the 95% confidence level. To convert to *BER* of $1e^{N}$, just replace the exponent "12" with N.

95% confidence level lower limits, *BER* > 10^{-12}		95% confidence level upper limits, *BER* < 10^{-12}	
Min number of errors	**Max number of compared bits (x1e12)**	**Max number of errors**	**Min number of compared bits (x1e12)**
1	0.05129	0	2.996
2	0.3554	1	4.744
3	0.8117	2	6.296
4	1.366	3	7.754
5	1.970	4	9.154
6	2.613	5	10.51
7	3.285	6	11.84

Using the upper and lower limits given in Table 1, we can for each measurement check whether the *BER* is below or above $1e^{-12}$ at the 95% confidence interval. The minimum and maximum numbers of bits for low numbers of errors are shown graphically in Figure 6. Note that there is a wide gap where the *BER* is so close to $1e^{-12}$ that we can't really decide. If we compared $3e^{12}$ bits for example and got 2 errors (a measured bit error ratio of $0.667e^{-12}$), we are in the "undecided" white area on the graph.

In such a case, we need to transmit more bits until the number of bits either reaches the upper limit ($4.744e^{12}$), or until we see more errors. If the actual *BER* is very close to $1e^{-12}$ however, we are unable to put a lower or upper limit on the *BER*, no matter how many bits we transmit. Whether such a test fails or passes entirely depends on the application.

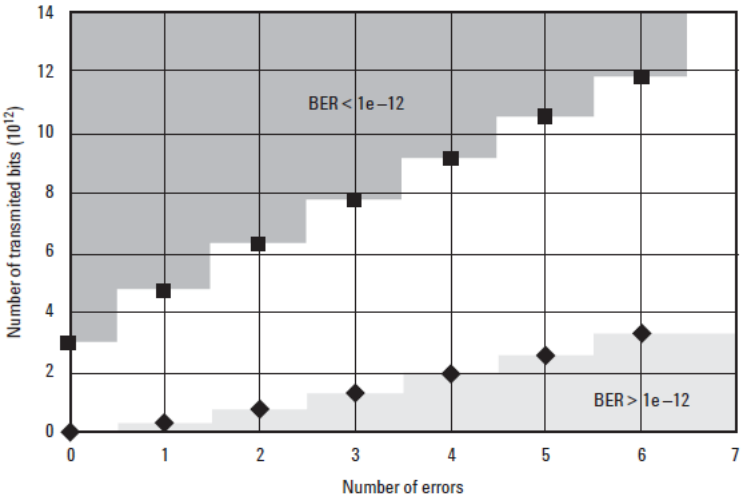

Figure 6. The 95% confidence level boundaries for upper (dark grey) and lower (light grey) limits on a *BER* of $1e^{-12}$

Sample point set up on a bit error ratio tester

Almost every BERT existing today has the ability to set its reference receiver to arbitrary decision thresholds and sample delays. Figure 7 shows an eye diagram acquired on a sampling oscilloscope with the definitions of the sampling delay offset and threshold. Time values are often shown in unit intervals, which is just the reciprocal of the bit rate. For example, at 10 Gbit/s, a unit interval equals 100 ps. By definition, the optimum sampling point has a time offset of zero. Modern BERTs are able to find the optimum sample delay offset and threshold automatically, and all sample delay offsets are relative to this sample point.

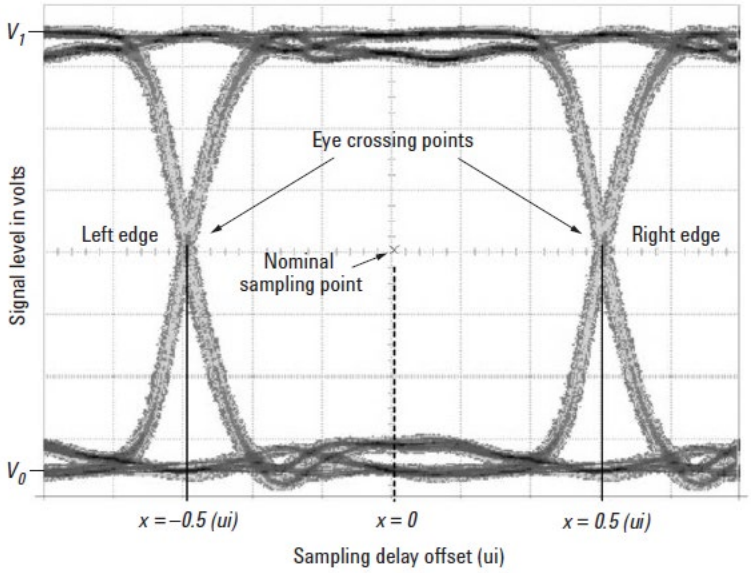

Figure 7. Eye diagram measured on a sampling oscilloscope, with BERT sampling setup definitions. The nominal or optimum sample point is located in the middle of the eye diagram

Bit error ratio and jitter

Bit errors can be caused by either logic errors in the transmitter itself, or by amplitude noise and jitter seen by the receiver. Unfortunately, amplitude noise is indistinguishable from jitter, which is why all jitter measurement analyses assume that amplitude noise is negligible. Same for logic errors, if there is a source of not randomly distributed errors in a system, jitter analysis cannot work.

Jitter, in a standard *BER* test at the optimum sample point, causes bit errors if the jitter peak-to-peak value exceeds 1 UI, so that the BERT receiver “sees” either the previous or the next bit. We can also cause error rates by moving the sample point toward the edge of the signal. If the sampling point is located exactly at the left edge (at –0.5 UI), only the jitter value and the value of the neighboring bit determine whether we see an error or not. The same is true at the right edge. Since we only observe an error if the neighboring bit is different from the current one, the *BER* at this sampling point will be equal to half the transition density.

BERT scan plots

The total jitter PDF is accumulated over many edge transitions, thus we can in turn place a *TJ* histogram on every edge. The *BER* vs. sampling delay offset is then the integral over the *TJ* PDF from the optimum sampling point to the left and to the right. Figure 8 shows an example, using the same jitter values that we used earlier, however this time with a rectangular *PJ* rather than a sinusoidal one. Note that the maximum *BER* in this example is 0.5, since we did the simulation for a random data pattern. Since the probability of two identical consecutive bits (the transition density) on a random data sequence is one-half, we had to scale the CDF integral by 0.5.

Since the PDF is placed on the edge, *BER* is usually measured beyond the /–0.5 UI offset; common values are ±0.75 UI. Such a curve is commonly termed a “bathtub curve”, because of its characteristic shape.

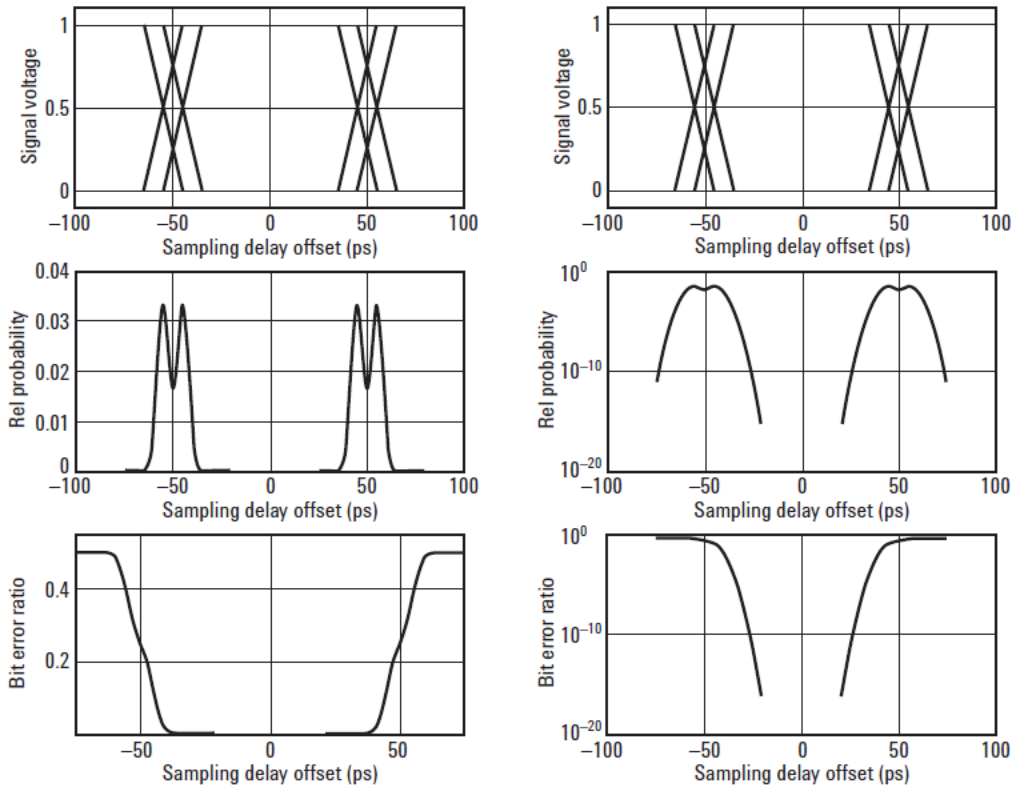

Figure 8. Schematic eye diagram (top), jitter histogram (middle), and *BER* vs. sampling delay offset (“bathtub curve”, bottom) in linear scale (left) and logarithmic scale (right), for a 10 Gbit/s signal with 10 ps *PJ* and 3 ps RJ_{rms}

Calculating total jitter from bathtub curves

Since the BERT scan curve is related to the total jitter cumulative probability density function (CDF), we can derive TJ_{PkPk} values from it. One possibility would be to simply take the right-hand side of the curve, and derive the peak-to-peak value from there, just as we did earlier with the CDF from measured histograms. But there is a much more intuitive possibility: we calculate the intersection of the left and right branch of the BERT scan curve with a *BER* threshold and get the eye-opening or phase margin at this

particular *BER* level as the difference between the two. Then, the TJ_{PkPk} value equals the system period minus the phase margin. The beauty of this derivation is that we can immediately relate it to a timing budget: the *TJ* peak-to-peak value is the portion of the unit interval that is not available for sampling if we need a *BER* performance better than the *BER* threshold used in the calculation.

Figure 9 shows the bathtub curve for the example above, in logarithmic scale and with a *BER* level of $1e^{-12}$. The intersections are at ±24.48 ps, so the phase margin is 48.96 ps. With the system period of 100 ps at 10 Gbit/s, the total jitter peak-to-peak value equals 51.04 ps.

Obviously, the lower the *BER* we select for the *TJ* calculation, the higher the *TJ* peak-to-peak value will be. This is consistent with the results we got from our earlier *TJ* peak-to-peak calculations where we used the measured CDFs. One important difference between the results however is that the *BER* scan method automatically takes the transition density into account: if consecutive bits in a data sequence are identical, then jitter will not cause an error, leading to a lower *TJ* reading. This is a good thing in virtually all applications since it is consistent with expectations from a timing budget standpoint.

Use of bathtub curves beyond total jitter calculation

Most jitter packages based on oscilloscopes or time interval analyzers available today plot bathtub curves, by integrating the extrapolated total jitter PDF. Those bathtub curves cannot possibly capture real bit errors; in fact, they will not even detect if you accidentally crossed data and data bar. Quite often, if chip timing is marginal, or if a rare digital logic error occurs in a design, bathtub curves can exhibit a *BER* floor, meaning that the *BER* is not zero in the middle of the eye. This indicates a severe design problem, and can only be observed on a *BERT*. If your *BERT* has capture memory or capture-around error capabilities, you can even debug your design on the digital level, just as you would do with a logic analyzer.

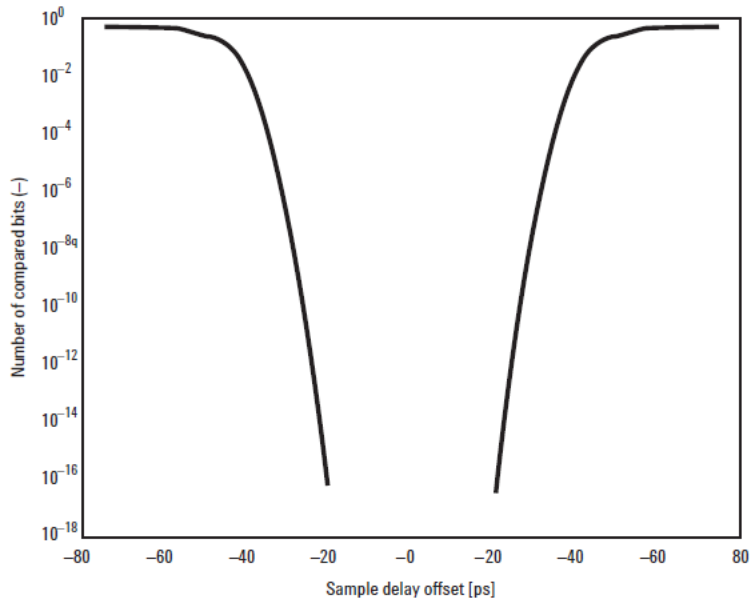

Figure 9. BERT scan or bathtub curve in logarithmic scale

Measuring the Bathtub Curve

Brute force

The easiest way to measure a bathtub curve is to sample an identical number of bits at equidistant sampling locations. In order to get minimum statistical confidence, we need to measure at least ten times the reciprocal target bit error ratio. For example, in order to measure a bathtub curve down to $1e^{-12}$ we need to compare $1e^{13}$ bits at each sampling point. A bathtub curve at 10 Gbit/s with a 1 ps resolution then takes 41.67 hours.

Number of errors optimization

The accuracy of a bit error ratio measurement increases with the number of errors that we observe. This means that we've measured the high *BER* portion of the bathtub, which we don't actually need for the *TJ* measurement, with high accuracy. One common optimization for bathtub measurements is therefore to limit the acquisition to a certain number of errors, in addition to the number of bits. Then, all *BER* values where the number of errors is reached before the number of bits are measured with the same accuracy. Figure 10 shows the numbers of compared bits for the same bathtub curve as in Figure 8, using $1e^{13}$ compared bits and a 1 ps delay resolution, for three different settings of the number of errors optimization.

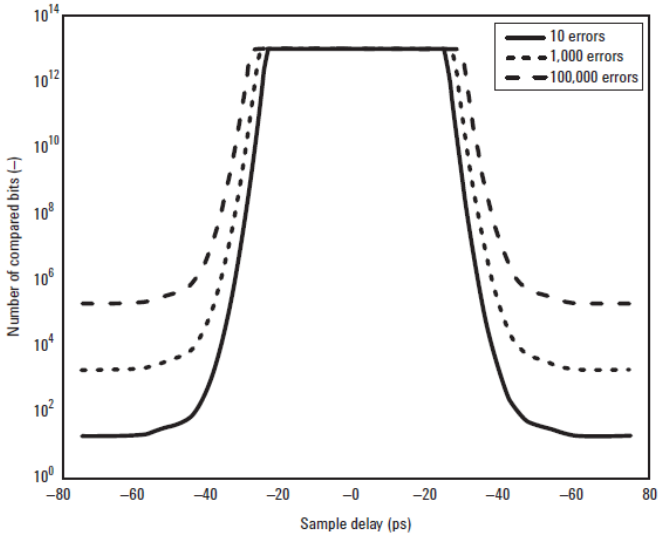

Figure 10. Number of compared bits vs. sample delay, for a 1 ps delay resolution at 10 ps DJ_{PkPk} and 3 ps RJ_{rms}. The lower the number of errors setting, the fewer bits need to be compared

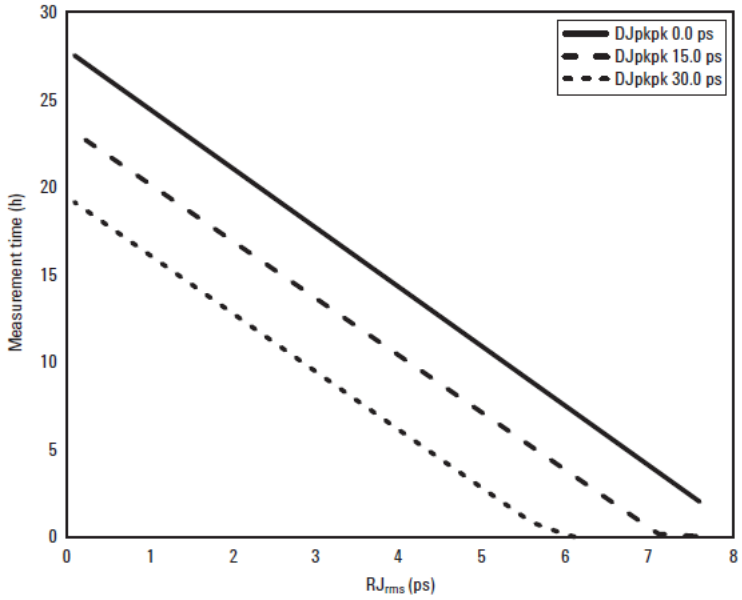

Figure 11. Measurement time vs. RJ_{rms}, for a 1ps delay resolution and three different DJ_{PkPk} values. The higher the total jitter, the faster the measurement

The number of error optimizations cuts down measurement time in the area of the bathtub curve where error rates are high. This means that the total measurement time depends on how much jitter is present on the signal. Figure 11 shows how long a bathtub measurement with $1e^{13}$ compared bits and 1000 errors will take, as a function of RJ_{rms} and for three different values of DJ_{PkPk}. Measurement times decrease almost linearly with both RJ_{rms} and DJ_{PkPk}, and approach zero as the *TJ* goes to 1 UI.

Note that the durations given are average values, and will vary slightly due to the statistical nature of *BER* measurements. Also, we didn't include the time required for moving the delay on a BERT and the minimum gate time. These numbers are usually very low however (in the millisecond range), so they don't have much influence on measurement times for all practical cases.

The fast total jitter algorithm

The major drawback of the number of error optimizations for the purpose of *TJ* evaluation is that most of the measurement time is spent in the middle of the bathtub, where the measured *BER* is zero. Since we take the *TJ* information from the intersections of the bathtub slopes with a *BER* threshold, this is unnecessary unless we want to verify that there is no (or a very low) *BER* floor. If we are purely interested in the *TJ* result, it is sufficient to find the sample delay offsets on the left and right slope of the bathtub curve where the *BER* is exactly $1e^{-12}$. We call these points $x_L(1e^{-12})$ and $x_R(1e^{-12})$. *TJ* peak-to-peak is then simply the bit interval, minus the difference between x_R and x_L. Unfortunately, since the BERT has a finite delay resolution, it is virtually impossible to set the sampling delay to exactly these points. And even if it was possible, we would need to observe an infinite number of bits to prove that the *BER* is really exactly $1e^{-12}$.

Bracketing approach

Since we are unable to find a single point on the slope where the *BER* is exactly $1e^{-12}$, we relax our search goal to an interval that brackets it, in the sense that we are reasonably sure that the point where *BER* is equal to $1e^{-12}$ lies within the interval. In Figure 12, we have shown this for the left slope: we search for an interval [*x*–, *x*+] that brackets the x_L point, where *x*+ and *x*– are separated by no more than the desired delay step accuracy of the *TJ* measurement, Δx.

We don't need to know the exact *BER* values at *x*+ and *x*–, it is sufficient to assert that *BER* (*x*–) > $1e^{-12}$ and *BER* (*x*+) < $1e^{-12}$ at a desired confidence level. If we choose 95%, we have determined that ($1e^{-12}$) is within the interval [*x*–, *x*+] with a confidence level of at worst 90%. For lack of better knowledge, we assume that is in the middle of the bracketing interval. Since the distance between *x*– and *x*+ is Δx, x_L is accurate to ±0.5 Δx. Repeating the procedure for the right slope of the bathtub curve yields x_R with the same accuracy, and we can calculate *TJ* peak-to-peak the same way we did before, with an accuracy of $\pm\sqrt{2\Delta x}$.

Search strategies

There are many possible algorithms to search for the interval [*x*–, *x*+]. Things to keep in mind during the implementation of a fast total jitter measurement using the bracketing approach are:

- Measurement times increase with decreasing *BER*, so that the search should be performed from left to right for the left edge, and from right to left on the right edge. The main goal of the search algorithm is to minimize the number of failed attempts to find *x*+. At 10 Gbit/s it takes about 5 minutes to compare $3e^{12}$ bits — the biggest investment in the process
- Once *x*– has been determined, *x*+ is often within Δx because of the steep slopes of the bathtub curve at low *BER* values
- The search lends itself well to an iterative process. We continue to refine the search steps until we've reduced the interval to the desired accuracy. The measurement times can be traded in a well-defined way against measurement accuracy
- In order to get better initial values for the search, it is a good idea to perform a relatively fast complete bathtub scan. From the data, we can get reasonable first guesses for *x*– and *x*+, either directly or by fitting an inverse error function to the data
- If the device under test has a *BER* floor, the search can be stuck because the *BER* never gets below $1e^{-12}$. A robust implementation has to account for this.

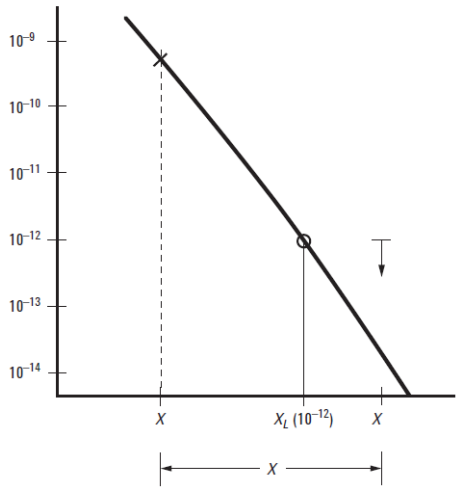

Figure 12. Definitions for the bracketing approach, on the left slope lower *BER* region

Example implementation one: linear search

In this section, we present a simple example implementation of the fast *TJ* measurement algorithm, using a linear search with constant step size.

1. Choose the desired uncertainty of the *TJ* measurement, *∂TJ*, and determine the delay step accuracy required for this:

   $$\Delta x = \frac{\delta TJ_{\beta}}{\sqrt{2}}$$

2. Starting at the optimum sample point, move the BERT analyzer sample delay to –0.75 UI.

3. Compare data until you observe at least one error, or until the number of compared bits exceeds $2.996e^{12}$.

4. If you stopped because of an error, check the number of bits compared so far:

   - if $N_{bits} < 0.05129e^{12}$, the *BER* is > $1e^{-12}$ at the 95% level, and we set *x*– to the current delay. Increase the sample delay by Δ*x*, and continue with step 3.
   - else, we are in the "undecided region". Increase the sample delay by Δx, and continue with step 3. If you end up here again, the bathtub either has a *BER* floor, or the slope of the bathtub is so gentle that we are unable to find *x*– and *x*+ that satisfy our accuracy requirement.

5. If you stopped because you reached the $2.996e^{12}$ limit without a single error, the *BER* is lower than $1e^{-12}$ at the 95% confidence level, and we set *x*+ to the current delay. Calculate $x_L(1e^{-12})$. We are done with the left slope, and continue with step 6.

6. Continue through steps 2 and 5, this time with an initial delay of +0.75, and with decreasing delay. This gives the *x*+ and *x*– values for the right slope of the bathtub curve. Calculate $x_R(1e^{-12})$.

7. Calculate *TJ*, measurement finished.

Average measurement times for this implementation are given in Figure 13. Note that measurement times with the bracketing approach are not strictly repeatable, since at lower *BER* values the time until the first error is observed is randomly distributed.

We've done the simulations using the average time between errors; variations will be averaged out in most real measurements.

For low *RJ* values, the measurement is complete after about 10 minutes, independent of *DJ*. Because of the very steep slopes in low *RJ* situations, only one measurement at low *BER* needs to be done per slope. The measurement time is then dominated by the time required to compare $2.996e^{12}$ bits (5 minutes), once per slope. This is independent of the delay resolution used: the minimum test time at 10 Gbit/s is always 10 minutes, no matter how coarse the resolution is.

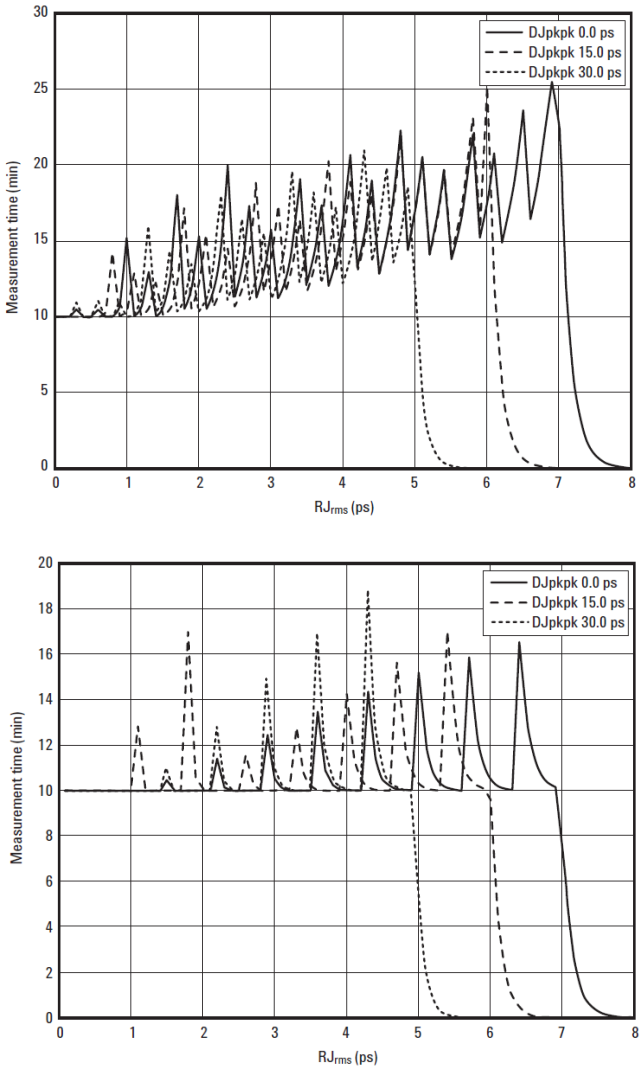

Figure 13. Measurement time vs. RJ_{rms}, for 1 ps (top) and 5 ps (bottom) delay resolution and three different DJ_{PkPk} values

For increasing *RJ* values, measurement time goes up because more points are located on the slope of the bathtub curve. The saw tooth shape in this region is really an indication of the random variability of the measurement time: it entirely depends on how many points are located on the slope, and where. The lower resolution setting hits fewer points on the slope, so the measurement completes earlier with decreasing resolution.

For high *RJ* values, test time quickly drops to almost zero, depending on the *DJ* value. If the total jitter exceeds 1 UI, the bathtub is closed, and the algorithm fails because no points with *BER* < $1e^{-12}$ can be found. But this is detected fairly quickly, unlike in a conventional bathtub measurement.

From Figure 13, we get an average measurement time of about 15 to 20 minutes, at 10 Gbit/s and with a 1 ps delay step resolution. By comparison with Figure 11, we find that the bracketing approach reduces measurement times by about a factor of 40 – 100, depending on *RJ* and *DJ* values, and a good portion of luck.

Example implementation two: binary search

Our second example implementation is more sophisticated: a full bathtub curve with a low number of bits that completes in a minute is used to get initial estimated as to where the bracket points might be. The actual brackets are then found by using binary search, dynamically adjusting the step size until the desired accuracy is reached.

This example is given mostly to show what's possible; the benefit in terms of measurement time saved over the simple implementation depends entirely on the shape of the bathtub curve.

1. Choose the desired uncertainty of the *TJ* measurement, ∂TJ, and determine the delay step accuracy required for this:

$$\Delta x = \frac{\delta TJ_{\beta}}{\sqrt{2}}$$

2. Measure a complete bathtub curve with a step size of Δx, using $1e^9$ bits and $1e^3$ errors

3. Make your first guess for $x\_$: to determine which of the delay settings can be used as your first guess for $x\_$ use Table 1. For example, if at the last time- delay setting of the fast bathtub two errors were observed, then Table 1 says that for two errors the maximum number of transmitted bits consistent with *BER* > $1e^{-12}$ at the 95% confidence level is $0.3554e^{12}$; since $1e^9$ bits were transmitted we have *BER* > $1e^{-12}$ with much better than 95% confidence. Define $x_0 = x$ and set $x\_ = x$, the first guess for the left bracket.

Once x_0 is defined, then data will be acquired at time-delay settings $x_n = x_0 + n_x$, be prepared to keep track of the number of errors detected and bits transmitted at each of these delay settings, (N_{Err}, N_{Bits}).

4. Try to get your first candidate for $x+$: Extrapolate the left slope of the bathtub down to $1e^{-12}$ by fitting a complimentary error function in the usual way (this is the standard bathtub plot technique for estimating *TJ* with a BERT). Determine the number of steps, n, from x_0 that give x_n closest to x'. That is, set n to the closest integer to $(x' - x_0)/\Delta x$ (things should go a little faster if you err on the side of higher *BER*, that is, round down). Set the time delay to $x_n = x_0 + n\Delta x$.

5. With the delay at x_n, transmit bits until either an error is received or 2.996 x 10^{12} bits are transmitted without an error.

6. If an error is detected in step 5: use Table 1 to determine if you have a new lower limit, $BER > 1e^{-12}$. If you have a new lower limit, then set $x\_ = x_n$ and set $n := n+1$ and go back to step 5. If you can't get a new lower limit, then continue to step 7.

7. Tweener situation: If an error is detected in step 5 but the total number of errors for the number of transmitted bits at the delay x_n is too small to give the lower limit, $BER > 1e^{-12}$, then we find ourselves in the gap between the shaded regions of Figure 6. This is the annoying "tweener situation" that almost never occurs but any decent algorithm must account for it. While annoying, it's not too bad a place to be because x_n is probably very close to $x(1e^{-12})$. Continue to transmit bits until you either get another error or have transmitted a total of $3.3e^{12}$ bits at x_n. If you get an error go back to step 6, if you don't, then you have between one and six errors, $1 \le N_{err} \le 6$, at x_n and are firmly set in the tweener region and x_n is the tweener-delay. The idea at this point is to make sure that x_n is really consistent with $x(1e_{-12})$. If so, then we'll use it, if not, we'll use something like it, but flag the result with a larger uncertainty and glean some useful information:

   a) If $x\_ = x_n - 1$ and $x+\_ = x_n + 1$, then set $x_L(10_{-12}) = x^n$ and go to step 9.

   b) If $x\_ \neq x_n - 1$, then decrement n: = $n - 1$ and go to step 4.

   c) If $x+\_ \neq x_n + 1$ or there is still no candidate for $x+$, then increment n: = $n+1$ and go back to step 5.

   d) It is extremely unlikely that you'll end up here, with more than one tweener point. This is the most interesting case of all because it tells us that there is a very gentle slope of BER near $BER = 1e^{-12}$. This indicates that something very strange, some low probability recurring deterministic event is going on. If you get here, then you really do need to perform the full bathtub curve, even if it takes a weekend, to figure out what is going on.

8. If $3e^{12}$ bits were transmitted without an error in step 4, then set $x_+ = x_n$.

9. If $x\_$ is farther from x_+ than Δx, that is, if $x_+ - x\_ > \Delta x$, then set $x = x_+ - \Delta x$, i.e., set $n = (x_+ - x_0)/\Delta x - 1$, and iterate the process by going back to step 5. But, if $x_+ - x\_ \le \Delta x$, then continue to step 10.

10. You've finished the left slope: set $x_L(1e^{-12}) = 1/2\ (x_+ + x\_)$ and repeat steps 1 through 9 for the right slope to obtain $x_R(1e^{-12})$.

11. Having obtained $x_L(1e^{-12})$ and $x_R(1e^{-12})$, you have $TJ(1e^{-12})$ with an accuracy of $\pm\sqrt{2}\Delta x$.

Conclusion

In this paper, we've shown that total jitter peak-to-peak can be measured on a bit error ratio tester and introduced a new method that allows us to trade test time versus accuracy.

We provided two different example implementations of our algorithm and showed that an improvement in measurement time of more than a factor of 40 compared to a conservative bathtub measurement can be achieved. Typical test times are approximately 20 minutes at 10 Gbit/s, and little more than one hour at 2.5 Gbit/s, for a total jitter measurement that was done at the $1e^{-12}$ *BER* level with a confidence level of better than 90%.

Due to the direct measurement approach, accuracy of the results is independent of the total jitter PDF. This is a significant advantage over other methods based on oscilloscopes or time interval analyzers, which fail miserably if the jitter distribution doesn't fit the extrapolation model.

References

[1] Ransom Stephens, "*Analyzing Jitter at High Data Rates*," IEEE Optical Communications, February 2004.

[2] Ransom Stephens, "*The Rules of Jitter Analysis*," Analog Zone, September 2004.

[3] "*Precision jitter analysis using the 86100C DCA-J*", Keysight Technologies.

[4] Lee Barford, "*Sequential Bayesian Bit Error Rate Measurement*", IEEE Transactions of Instrumentation and Measurement, Vol. 53, No. 4, August 2004