Accelerating Jitter Tolerance Qualification for High Speed Serial Interfaces

Yongquan Fan and Zeljko Zilic
Department of ECE, McGill University
yongquan.fan@mail.mcgill.ca
zeljko.zilic@mcgill.ca

Abstract

We witness a phenomenal increase in the use of high-speed serial interfaces (HSSIs). Post-silicon validation and testing of HSSIs are critical to guarantee the design quality and the device quality. Jitter tolerance at 10^{-12} Bit Error Rate (BER) is a key parameter that is very costly to qualify due to the long test time. This paper considers an acceleration scheme to quantify post-silicon jitter tolerance. It can reduce the test time from hours to seconds in validation and to tens of milliseconds for compliance testing. Experimental results at 3 Gigabit per second (Gbps) data rate demonstrate the accuracy of our technique in pico-second range.

Keywords

Jitter, jitter tolerance, serial interface, bit error rate

1. Introduction

The aggressive scaling in deep submicron technologies has enabled the System-on-Chip (SoC) integration of a microcontroller/DSP, ADC/DAC, memory blocks, power management unit, PLL and external interfaces. The staggering complexity makes it challenging to design fault-free electronic products. As a consequence, close to 25% of all design resources at Intel are now spent on post-fabrication validation [1]. Furthermore, according to the data by Collett International, the timing, mixed-signal interfaces, clocking and crosstalk are among the prime failure reasons, each contributing 18% or more to failing the first silicon. HSSIs, which are interchangeably referred to as Serializer/Deserializer (SerDes), personify all such issues, and are hence critical to achieving the overall system quality.

It is challenging and expensive to qualify the SerDes devices, especially one key parameter - jitter tolerance. Jitter is the deviation of a signal from its ideal timing. The timing deviation may cause bit errors. Numerous HSSI standards define jitter tolerance performance at the 10^{-12} BER level, which requires running at least 10^{13} bits. This requirement fundamentally limits test speed: for instance, at 3Gbps data rate, it takes around one hour to run that many bits. With some emerging applications demanding 10^{-14} BER, direct measurement is even further from being practical.

Because of the long test time, it is usually only possible to validate the jitter performance on a bench in limited combinations of Process, Voltage and Temperature (PVT). In production, it is only assumed that jitter specifications are "guaranteed by design". Unfortunately, this assumption is not valid anymore as we keep advancing the semiconductor technology and increasing the data rate, which results in tightening the jitter budget. Devices could fail in real applications just because of their poor jitter tolerance performance. It is hence becoming imperative to perform jitter tolerance characterization thoroughly and implement jitter tolerance test in production [2][3][4]. This is the only way to ensure the design quality and device quality.

In this paper, we present a scheme to accelerate post-silicon qualification for SerDes devices. It drastically reduces jitter tolerance characterization time, and hence can improve design quality by validating the design at different PVT combinations. The scheme also makes it possible to implement jitter tolerance test in production to guarantee the quality of each device. In the remainder of the paper, we give a brief introduction of the SerDes jitter specification qualification in Section 2. In Section 3, we present the jitter injection mechanism and test signal calibration results. Section 4 introduces the jitter extrapolation algorithm and Section 5 details how we accelerate jitter tolerance qualification.

2. Background

SerDes has been widely used in communication standards such as SATA, Fiber Channel and XAUI, either standalone or as a component of a SoC. Figure 1 shows the architecture of a typical SerDes device. The transmitter (Tx) takes parallel data, converts it into serial format, and then drives the serial data to the transmission media. The PLL generates an internal high-speed serial clock for the serializer by multiplying the reference clock.

The receiver (Rx) accepts high speed serial data and restores it to the original parallel format. The Clock Data Recovery (CDR) circuitry generates a recovered clock from the received serial data, and then the recovered clock re-times the received serial data. The re-timed data is restored to parallel format by the deserializer. This transmission mechanism needs the encoding and decoding logic to manipulate the transmitted data transition density to make sure that the CDR circuit can function correctly.

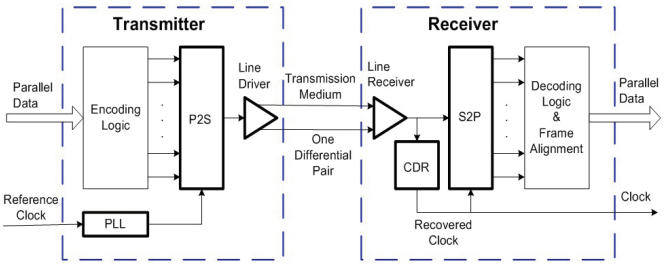

Figure 1: The architecture of a typical SerDes device

By encoding the clock into the data stream, CDR circuitry guarantees that the clock and data are in phase when the jitter frequency is in the bandwidth of the CDR. The CDR, along

with other high-speed technologies, has pushed the serial data rate above 10Gbps. However, if there is high frequency jitter in the data stream, the recovered clock may not track the data and the jitter may cause bit errors.

Jitter is composed of both deterministic and random contents. Deterministic Jitter (DJ) includes Periodic Jitter (PJ), Duty Cycle Distortion (DCD) and Inter-Symbol Interference (ISI). Random Jitter (RJ) is usually characterized statistically by a Gaussian distribution. Two parameters that can quantitatively characterize RJ are the Root-Mean-Square (RMS) value and the peak-to-peak value. The peak-to-peak value is associated with a certain level of BER. Total Jitter (TJ) is hence related to BER: the TJ is different when defined at different BER levels.

Most communication standards specify jitter in terms of DJ and TJ as separate specifications. For example, the 3Gbps SATA receiver should tolerate 0.60UI TJ at 10^{-12} BER level and should tolerate 0.42UI out-of-band DJ [5]. To guarantee the design quality, we need to validate and test whether silicon devices meet these specifications. While the details of the Tx testing on ATE are presented in [6], we concentrate on Rx jitter tolerance testing in this paper. The whole SerDes test solution is illustrated in Figure 2. Our Rx testing solution relies on an Arbitrary Waveform Generator (AWG), such as Teradyne AWG6000 [7].

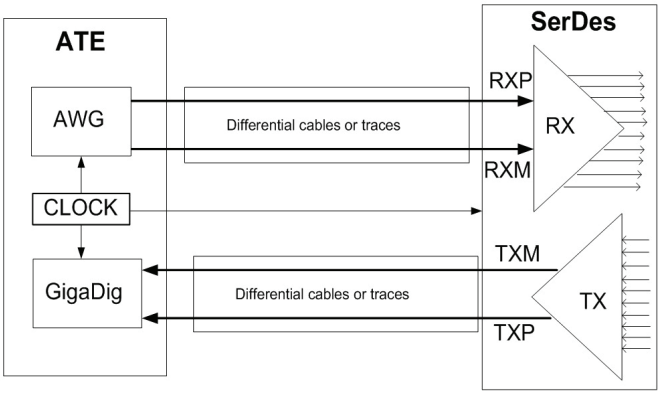

Figure 2: SerDes test solution on ATE

To perform jitter tolerance testing, we need to stress the receiver using test signals with controllable amount of jitter, to be able to test it at high BER levels and then extrapolate to the low BER level. An early investigation in [8] presents the concept of jitter tolerance extrapolation based on experiments at 1.5Gpbs data. This paper refines the extrapolation algorithm, provides analytical model and verifies its application. More importantly, we propose systematic acceleration schemes for post silicon validation and production testing, including test signal setting, jitter specification translation, and test limits determination. We verify the schemes at 3Gpbs data down to 10^{-12} BER.

3. Test signal generation – jitter injection

In our implementation, we inject jitter to the test signals by modulating ideal AWG binary signals with a user defined jitter profile. Our jitter injection involves the following four steps:

(1) Create a digitized sinusoidal signal of proper amplitude and frequency representing the PJ to inject
(2) Over-sample ideal binary data stream with sub-picosecond resolution
(3) Modulate data edges, converting jitter amplitude information to timing information by moving the data signal edge based on the jitter amplitude
(4) Filter out components above the Nyquist frequency and decimate the waveform to get desired AWG samples

On ATE, we calibrate the amount of injected jitter by connecting the AWG output to the input of a high bandwidth digitizer. The digitizer captures the AWG output and we then extract the jitter from the captured waveform [6]. To further verify our jitter injection scheme and to report a confident jitter tolerance number, we used a Wavecrest SIA-3000 to calibrate the jitter that we injected. Figure 3 shows the measured PJ and TJ values at different injected PJ levels on one tester. As can be seen, from 20ps to 200ps, the measured PJ correlates well to the injected PJ and there is a constant offset between the PJ and the measured TJ. In this case, the offset is around 80ps. When the injected PJ is below 20ps, the TJ does not change much due to the noise floor of the AWG. This does not impact us as we will show later that the test signals we need should have PJ values more than 100ps.

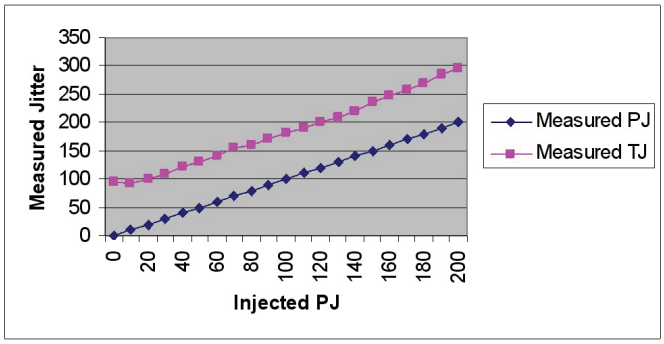

Figure 3: 3Gbps Test signal calibration

The constant offset between the injected PJ and the measured TJ is caused by the intrinsic RJ of the AWG, and DCD and ISI from the AWG and cables. The offset enables us to relate the PJ to TJ. Therefore, we can translate the TJ compliance test into PJ testing, where we can control the amount of PJ in the test signal.

4. Post-silicon jitter tolerance extrapolation

4.1 Jitter and BER

Jitter can cause bit errors in serial communication when the recovered clock re-times the data signal as shown in Figure 1. Figure 4 illustrates the relationship between jitter and BER. Ideally, the data is always sampled in the mid-bit (sampling instance *ts* = UI/2 in Figure 4(a)). This is usually true if the jitter frequency is within the bandwidth of the CDR because the sampling clock is recovered from the data signal. However, for out-of-band jitter, the sampling clock can not track the data any more and the jitter can cause bit errors.

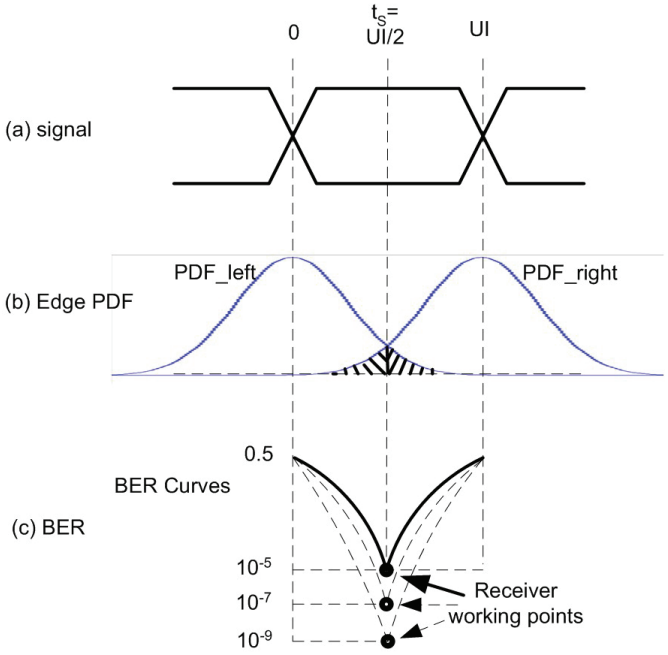

Figure 4: Jitter and BER in the receiver

Figure 4(b) shows an example of the jitter profile. The signal edge transition is disturbed by RJ. The RJ is assumed to be Gaussian. The Probability Density Function (PDF) of a zero-mean Gaussian variable is written by

$$p(x) = \frac{1}{\delta\sqrt{2\pi}} e^{-x^2/2\delta^2}$$

where δ is the standard deviation.

One important function used to characterize the Gaussian distribution is Q function, which represents the area under the tail of the Gaussian PDF. The Q function is widely used for computing the error probability in communication systems [9]. Normalized to zero mean and unit variance, $Q(x)$ is defined as

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2} dt, \quad x \geq 0$$
$$= \frac{1}{2} erfc(\frac{x}{\sqrt{2}}) \qquad (1)$$

where $erfc(x)$ denotes the complementary error function, defined as

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt \qquad (2)$$

As shown in Figure 4(b), the PDF of the left and right edges of the data bit in Figure 4(a) can be expressed by

$$p_{\_left}(x) = \frac{1}{\delta\sqrt{2\pi}} e^{-x^2/2\delta^2}$$
$$p_{\_right}(x) = \frac{1}{\delta\sqrt{2\pi}} e^{-(x-UI)^2/2\delta^2}$$

Bit errors may occur when the left edge occurs after the *ts* (illustrated by back slash) or the right edge occurs before the *ts* (illustrated by forward slash). Assuming uniform bit distribution, i.e., a 50% chance of errors in these two cases, the BER can be expressed by

$$BER(t_s) = 0.5 * (\int_{t_s}^{\infty} p_{\_left}(t)dt + \int_{-\infty}^{t_s} p_{\_right}(t)dt$$
$$= 0.5 * (\int_{t_s}^{\infty} \frac{1}{\delta\sqrt{2\pi}} e^{-\frac{t^2}{2\delta^2}} dt + \int_{-\infty}^{t_s} \frac{1}{\delta\sqrt{2\pi}} e^{-\frac{(t-UI)^2}{2\delta^2}} dt)$$
$$= \int_{t_s}^{\infty} \frac{1}{\delta\sqrt{2\pi}} e^{-\frac{t^2}{2\delta^2}} dt \qquad (3)$$

By substituting equations (1) and (2) to (3), we have:

$$BER(ts) = 0.5 * erfc(\frac{t_s}{\delta\sqrt{2}}) \qquad (4)$$

By substituting δ with the RMS value of the RJ, $RJ_{RMS.}$, equation (4) becomes

$$BER(ts) = 0.5 * erfc(\frac{t_s}{RJ_{RMS}\sqrt{2}}) \qquad (5)$$

Equation (5) directly links the jitter to BER. The BER and jitter relationship can further be transferred to BER and Q-factor. As shown in Figure 4(c), the receiver works at the crossing points of the bathtub curves. Therefore, we have

$$t_s = UI/2 \qquad (6)$$
$$UI = DJ + 2Q * RJ_{RMS} \qquad (7)$$

where Q is $Q(x)$ defined in equation (1) with $x = BER$ [10].

By substituting t_s and RJ_{RMS} in equation (5) according to equations (6) and (7), we have

$$BER = 0.5 * erfc(\frac{Q}{\sqrt{2}}) \qquad (8)$$

or

$$Q = \sqrt{2} * erfc^{-1}(2 * BER) \qquad (9)$$

Equations (8) and (9) directly link BER and Q-factor. If we know one parameter, the other can be calculated accordingly. Our jitter tolerance extrapolation would need these two equations.

In the above analysis, we ignore the DJ effects. This is reasonable since the bit errors are caused by RJ at low BER regions. Because the DJ is bounded, it only adds offsets to a bathtub curve; it does not change the shape of the lower part of the bathtub curve [10].

4.2 Jitter tolerance extrapolation algorithm

The goal of jitter tolerance extrapolation is to predict the jitter tolerance at low BER based on high BER region data. The jitter sources that stress the CDR can come from the AWG, connection cables and the CDR itself. Figure 5 illustrates these jitter sources, where we use the following symbols to represent the different jitter sources:

- $PJ_{INJECTED}$: the PJ injected in the AWG signals
- DCD_{EXT}: the DCD from the AWG and cables
- ISI_{EXT}: the ISI from the AWG and cables
- RJ_{EXT}: intrinsic RJ of the AWG
- DJ_{CDR}: intrinsic DJ of the device
- RJ_{CDR}: intrinsic RJ of the device

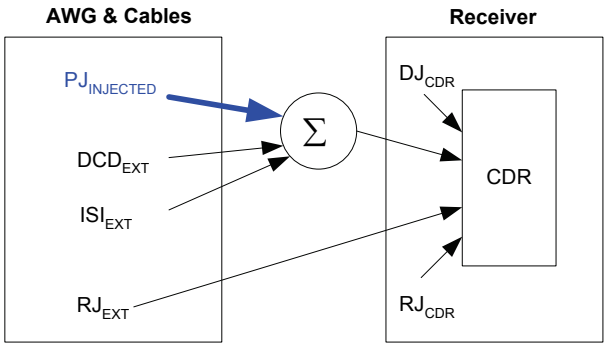

Figure 5: Jitter sources to the CDR

Under the "black box" assumption, we have no knowledge about the DJ_{CDR} and RJ_{CDR}. However, we can assume that RJ_{CDR} is a constant for the same device with the same injected jitter frequency because the RJ_{CDR} results from the CDR response to the jitter frequency. We also assume that the RJ_{EXT} is a constant when we sweep through the different amounts of PJ at a fixed frequency. This is reasonable because the RJ in the AWG comes mostly from the sampling clock, which is constant when we change the programmed samples for PJ injection. It can also be proved by the jitter calibration result shown in Figure 3, where RJ_{EXT} is included in the constant offset between the PJ and TJ. Considering RJ_{CDR} and RJ_{EXT} are independent, if we denote the total RJ seen by the CDR with RJ_{TOT}, RJ_{TOT} is a constant and we have:

$$RJ_{TOT} = \sqrt{RJ_{EXT} + RJ_{CDR}} \qquad (10)$$

The ISI and DCD are assumed to be constant when the PJ is incremented. This is also a reasonable assumption, because they are mainly determined by the group delay caused by bandwidth limitation. It is also proved by the constant offset between the PJ and TJ shown in Figure 3. Of course, how the PJ combines with the ISI depends on the relative phase relationship, which is unknown inside the DUT. As all the DJ sources are uncorrelated, the total DJ seen by the CDR, DJ_{TOT}, can be expressed as:

$$DJ_{TOT} = (PJ_{INJECTED} + ISI_{EXT} + DCD_{EXT}) + DJ_{CDR} \qquad (11)$$

Under the Q-factor model at the bathtub crossing point (filling up $1UI$), by plugging equations (10) and (11) to equation (7), we have

$$UI = PJ_{INJECTED} + DJ_{DELTA} + 2Q * RJ_{TOT} \qquad (12)$$

where DJ_{DELTA} is a constant defined as

$$DJ_{DELTA} = ISI_{EXT} + DCD_{EXT} + DJ_{CDR}$$

Rewriting equation (12) to solve Q and PJ, we have

$$Q = C \times PJ_{INJECTED} + S \qquad (13)$$

$$PJ_{INJECT} = \frac{Q - S}{C} \qquad (14)$$

where C and S are constants defined by

$$C = -\frac{1}{2RJ_{TOT}} \qquad (15)$$

$$S = \frac{UI - DJ_{DELTA}}{2RJ_{TOT}} \qquad (16)$$

Equation (13) demonstrates that the Q factor is a linear function of the injected PJ. Plugging equation (13) into equation (8), we have:

$$BER = 0.5 * erfc\left(\frac{C * PJ_{INJECTED} + S}{\sqrt{2}}\right) \qquad (17)$$

Equation (17) enables us to estimate BER according to the injected PJ in the test signal. We can extrapolate the PJ tolerance at low BER levels (such as 10^{-12}) once we know the two constant values C and S, which can be obtained using higher BER data (such as 10^{-10} and higher).

5. Accelerating jitter tolerance qualification

5.1 Jitter tolerance characterization

In design validation and production characterization, we need to get the jitter tolerance number at different PVT corners. Our scheme is to perform a jitter tolerance BER scan using test signals with different levels of injected PJ. High BER data is collected in the range of 10^{-6} to 10^{-10}. Q-factor values at different BER levels are calculated according to equation (9). Theoretically, we only need two data points to get the two constant values C and S in equation (13) according to

$$C = \frac{Q_1 - Q_2}{PJ_1 - PJ_2}$$

$$S = \frac{Q_2 * PJ_1 - Q_1 * PJ_2}{PJ_1 - PJ_2}$$

However, we need to catch a large number of bit errors when testing BER in order to get a repeatable result because of the randomness of the RJ. A better approach is to use more data points and perform linear regression fitting. Table 1 lists one example of measured BER and calculated Q values at different PJ levels. Figure 6 is a linear regression fitting of the Q factor versus PJ.

Table 1: High BER data for jitter tolerance extrapolation

PJ(ps)	216	218	220	222	224	226	228
BER	2.13 E-10	4.37 E-10	3.90 E-9	2.43 E-8	1.05 E-7	7.06 E-7	2.05 E-6
Q	6.24	6.13	5.77	5.46	5.19	4.82	4.60

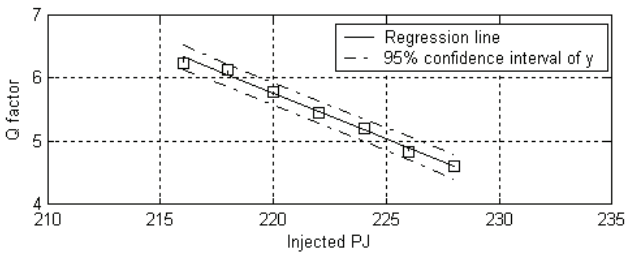

Figure 6: Q factor vs PJ

Based on the Q factor fitting result, we can now plot the BER curve as a function of the injected PJ according to equation (17), and thus predict the jitter tolerance at low BER levels. Figure 7 shows the BER curve based on the fitting result from the measurements in Table 1 (diamonds). It also shows low BER measurements for extrapolation accuracy verification (star points). As we can see, at 10^{-12} BER there is

only 1ps discrepancy between the actually measured PJ tolerance and the extrapolated PJ tolerance based on the BER data above 10^{-10}. We tried the procedure on different devices and the accuracy is within 2% while our solution can speed up the characterization >100.

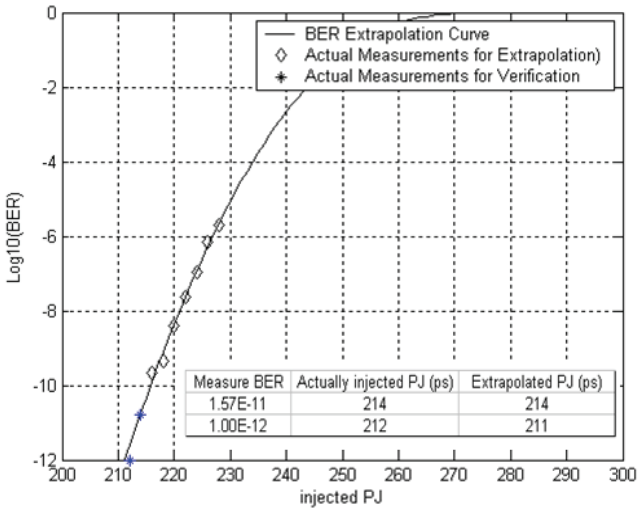

Figure 7: BER extrapolation

The jitter tolerance result presented in Figure 7 is in terms of PJ. To include the ISI+DCD and RJ, we need to link this diagram with jitter injection calibration curves shown in Figure 3. The delta between the injected PJ and the actual TJ observed by the SIA-3000 is a constant around 80ps. For this particular device under test, the jitter (TJ) tolerance at 10^{-12} BER is 292ps or 0.88UI while PJ is 212ps.

5.2. Jitter tolerance testing in production

If we directly apply the jitter tolerance characterization technique to production, the test time overhead is still a bit high because it involves BER to Q-factor translation, Q-factor fitting and BER extrapolation down to 10^{-12} level. It takes around one second, which is still too long for one parameter testing – on average an SoC device may have hundreds of parameters to test.

Considering production test is only a go/no-go judgment process, we do not need to know the exact value of the jitter tolerance for each device; we only need to know whether the jitter tolerance of a device is better than the jitter specification defined at 10^{-12} BER. Instead of performing the jitter tolerance extrapolation down to 10^{-12} BER and comparing it with the specification, we perform the jitter tolerance compliance test at a higher BER level. For example, we can do the test by qualifying 10^{-6} BER performance. We apply a test signal with a certain amount of injected PJ to the device: if the measured BER of the device is better than 10^{-6}, it passes; otherwise, it fails. For this approach, we need to solve two issues:

- Translating the total jitter tolerance specification from 10^{-12} BER level to 10^{-6} BER level
- Translating the TJ specification to injected PJ specification

The proposed jitter tolerance extrapolation algorithm can transfer jitter tolerance specifications at different BER levels. Because the Q-factor values at 10^{-12} and 10^{-6} BER levels are known (7.0374 and 4.7534 respectively [9]), according to equation (14), the PJ tolerance difference between 10^{-12} and 10^{-6} BER levels can be calculated by

$$PJ_{10^{-12}-10^{-6}} = \frac{Q(10^{-12}) - Q(10^{-6})}{C}$$

which is 25ps in the above example. According to equation (15) and equation (10), the difference is determined by the RJ in the test signal and the intrinsic RJ of the device that slightly varies from device to device. For each new design, we need to perform the jitter tolerance extrapolation to characterize the PJ difference distribution. Then we use the worst case value (the minimum value) to set test limits for production.

Next, we need to translate the SATA TJ specification into PJ specification because we can only control the amount of PJ in the test signal. The test limit we need to set in production should be based on the amount of the injected PJ. This translation is done according to the offset value between the measured TJ and the injected PJ as shown in Figure 3. To do this translation for production, we need to perform the test signal calibration at all the testers because the offset may vary from tester to tester. Figure 8 shows the offset at some testers. For these testers, it is valid to claim that the offset between the injected PJ and the actual TJ is at least 70ps.

Based on this offset, we can translate the SATA TJ specification into the PJ tolerance requirement in this case. In SATA II, the TJ specification is 200ps at 10^{-12} BER level. We can guarantee the TJ specification by checking the PJ tolerance at 130ps: if a device can tolerate 130ps PJ at 10^{-12} BER level, we can guarantee that the device meets the SATA jitter tolerance specification. Even though this might slightly overstress devices on some testers (such as Tester2 and Tester3), this is acceptable as long as it does not cause yield issues.

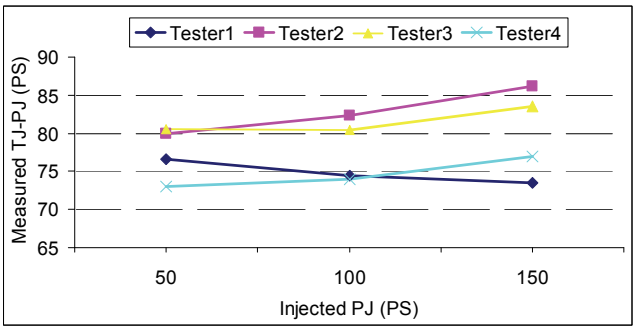

Figure 8: The offset between PJ and TJ at different testers

According to the jitter specification translation result, the PJ difference between 10^{-12} and 10^{-6} BER levels in the example is 25ps. Because at 10^{-12} BER level, the PJ tolerance requirement is 130ps, the PJ tolerance limit should be set to 155ps at 10^{-6} BER level. We can source a test signal with 155ps injected PJ to the receiver and check 10^{7} bits of recovered data. If no errors are detected, this device is classified as a good one; otherwise, it fails the jitter tolerance compliance test.

5.3. Discussions

In the proposed acceleration scheme for jitter tolerance qualification, the injected jitter calibration and the jitter tolerance extrapolation are the two key techniques we employ. When applying the solution to production testing, we need to especially pay attention to them.

We need to calibrate the test signals on all testers to make sure that the difference between injected PJ and measured TJ is bigger than the offset we used to derive the test limit, which is 70ps in the example. If it is below this, we need to tighten our test limit accordingly. In the same time, we also need to keep an eye on the possible yield loss because we overstress devices on some testers, such as on Tester2 we overstress the device by around 10ps. This should not cause issues because the design margin normally is big enough to accommodate it. Another source that provides extra margin for the test is that we classify devices with errors between 1 and 10 out of 10^7 bits as bad devices. Actually, they can be classified as good ones as they meet 10^{-6} BER performance with the injected jitter. This gives us extra guard band.

In addition, we need to do the jitter specification translation (from 10^{-12} to 10^{-6} BER levels) based on devices that can cover the product to be tested, such as devices from all process corners. Doing this from one device may not be enough. The good thing is that we only need to do this once for every new design.

Even though the experiment is conducted on Teradyne AWG6000, the jitter tolerance extrapolation technique can be used on any platform that has jitter injection capability and that can perform BER testing. The technique can rapidly report the actual jitter tolerance value or qualify a jitter tolerance specification.

6. Conclusions

We have demonstrated an innovative method to make the time-consuming jitter tolerance test run faster by at least 100 times. Experimental data collected at 10^{-12} BER demonstrates the accuracy of our technique in pico-second range. This method can drastically reduce the validation and test time. The reduced time-to-market and guaranteed performance form the foundation of a quality electronic design.

7. Acknowledgments

The authors would like to acknowledge the support from LSI Corporation. We would especially like to thank Dr. Yi Cai for providing invaluable insights to an early version of our paper. Our acknowledgement also goes to Mr. Liming Fang for exchanging knowledge and helping calibrate the test signals.

8. References

[1] P. Patra, "On the Cusp of a Validation Wall," *IEEE Design & Test*, vol. 24, n. 2, Mar.-Apr. 2007.

[2] S. Sunter, A. Roy, J. Cote, "An Automated, Complete, Structural Test Solution for SERDES", *IEEE International Test Conference*, 2004

[3] M. Hafed, D. Watkins, C. Tam and B. Pishdad, "Massively Parallel Validation of High-speed Serial Interfaces using Compact Instrument Modules", *IEEE International Test Conference*, 2006

[4] M. Ishida, T. Yamaguchi, and M. Soma, "A Method for Testing Jitter Tolerance of SerDes Receivers Using Random Jitter", *DesignCon 2007*

[5] Serial ATA International Organization: *Serial ATA Revision 2.5 Specification* ("Final Specification"), October 27, 2005

[6] Y. Fan, Y. Cai, Z. Zilic, "A High Accuracy High Throughput Jitter Test Solution on ATE for 3Gpbs and 6Gbps Serial-ATA", *IEEE International Test Conference*, 2007

[7] Teradyne, Inc. *http://www.teradyne.com*

[8] Y. Fan, Y. Cai, L. Fang, A. Verma, B. Burcanowski, Z. Zilic and S. Kumar, "An Accelerated Jitter Tolerance Test Technique on ATE fro 1.5GG/s and 3GB/s Serial-ATA", *IEEE International Test Conference ITC* 2006.

[9] J. G. Proakis, *Digital Communications*, McGraw-Hill High Education, 2001.

[10] Y. Cai, B. Laquai, and K. Luehman, "Titter Testing for Gigabit Serial Communication Transceivers", *IEEE Design & Test of Computers*, Vol. 19, Issue 1, Jan, 2002