PCI Express™ Architecture

PCI Express™ Jitter Modeling
Revision 1.0RD

July 14, 2004

PCI EXPRESS®

REVISION	REVISION HISTORY	DATE
1.0RD	PCI SIG member review	07/14/04

PCI-SIG disclaims all warranties and liability for the use of this document and the information contained herein and assumes no responsibility for any errors that may appear in this document, nor does PCI-SIG make a commitment to update the information contained herein.

Contact the PCI-SIG office to obtain the latest revision of this document.

Questions regarding this document or membership in PCI-SIG may be forwarded to:

Membership Services
www.pcisig.com
E-mail: administration@pcisig.com
Phone: 503-291-2569
Fax: 503-297-1090

Technical Support
techsupp@pcisig.com

DISCLAIMER

This document is provided "as is" with no warranties whatsoever, including any warranty of merchantability, noninfringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. PCI-SIG disclaims all liability for infringement of proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

PCI Express is a trademark of PCI-SIG.

*All other product names are trademarks, registered trademarks, or servicemarks of their respective owners.

Contents

Figures

Preface

About This Document

The goals of this document are to increase the knowledge base of the PCI SIG members regarding the relationships between the reference clock jitter, the transmitter and receiver for different types of receiver architectures. We develop the definitions, theory of operation and mathematical tools used for predicting the effect of jitter in the PCI Express architecture.

This document assumes that the reader is familiar with the electrical specifications as defined by the *PCI Express Base Specification* [1].

Technical questions and comments regarding this document may be sent to the chairs of the PCI Express jitter working group:

<div style="margin-left: 2em;">

Andy Martwick
Intel Corporation
andy.martwick@intel.com

Dr. Mike Li
Wavecrest Corporation
mli@wavecrest.com

</div>

Acknowledgements

The following companies have contributed content to this document:

- Agilent

- Advanced Micro Devices

- Integrated Circuit Systems, Inc.

- Intel

- LSI Logic

- Texas Instruments

- Wavecrest

- National Instruments

- nVIDIA

Each company retains the copyright to the material contributed. The material in this document may not be reproduced without the content owners consent. Questions regarding reprint and content ownership may be addressed to the chairs of the jitter work group.

1 Introduction

Serial communication is not new to the PC architecture; examples of existing technologies include RS-232, Ethernet, USB, USB2.0, and serial ATA. However, as the data rates enter the multiple gigabit/second rate and the bit unit intervals are reduced to less than ½ of a nanosecond, new challenges arise in the modeling and understanding of jitter and the impact of jitter on the system. Previous second order effects that were once unimportant can no longer be ignored.

This paper develops a system model for reference clock jitter in a PCI Express system that can be used to predict the impact of the reference clock jitter on the system timing budgets. The model is developed specifically for the jitter introduced by the reference clock as defined in the *PCI Express Card Electromechanical Specification, Rev. 1.0a* [2].

1.1 Limitations

This paper does not discuss the other sources of jitter in the system, or the nature of random jitter (Rj) vs. deterministic jitter (Dj). These issues will be developed in a future paper.

This paper does not account for the data transition density effects of random data patterns. The data channel is modeled assuming a repeating 10101 pattern, equal to a clock. This simplification provides an unambiguous phase definition for both the clock and the data.

For convenience, the models developed are in the S domain, whereas the actual operation of the sampled system is a discrete time response that is more accurately modeled in the Z domain. Studies were briefly done to ensure that the error between these two methods was minimal.

Only two types of receiver architectures are considered, PLL based and Digital based. Simplifications are made about the frequency response of these types of receivers; other receiver architectures may be possible that require different models.

The PCI Express specification defines the timing budgets to be met at a BER of 10^{-12}. Similarly, discussions of eye closure or timing budgets in this paper are to be assumed at the same BER of 10^{12}. The examples in this paper were developed on data sets that extend to a BER of 10^{-6}. The mathematics for inclusion of the BER at 10^{-12} in the system model will be developed in a future paper.

1.2 Document Outline

In Section 2, we develop the detailed definition of jitter, and how jitter can be represented using discrete math. In Section 3, we describe some of the possible receiver architectures used in PCI Express. Section 4 describes the transfer functions and frequency response of these types of architectures. This section also fully develops the system model for the reference clock to "receiver eye closure" at the receiver. Section 5 develops a derivation of the frequency response of the compliance measurement and provides the "measurement eye closure." Section 6 discusses the results and suggests next steps.

2 Definitions

2.1 Bits

A PCI Express receiver is a differential receiver that measures the difference between the two signals to determine the binary data value of a 0 or a 1 at a particular time. These two signals are called D+ and D-. When the two signals are equal, this is called a threshold crossing. An example of two threshold crossings is shown in Figure 1.
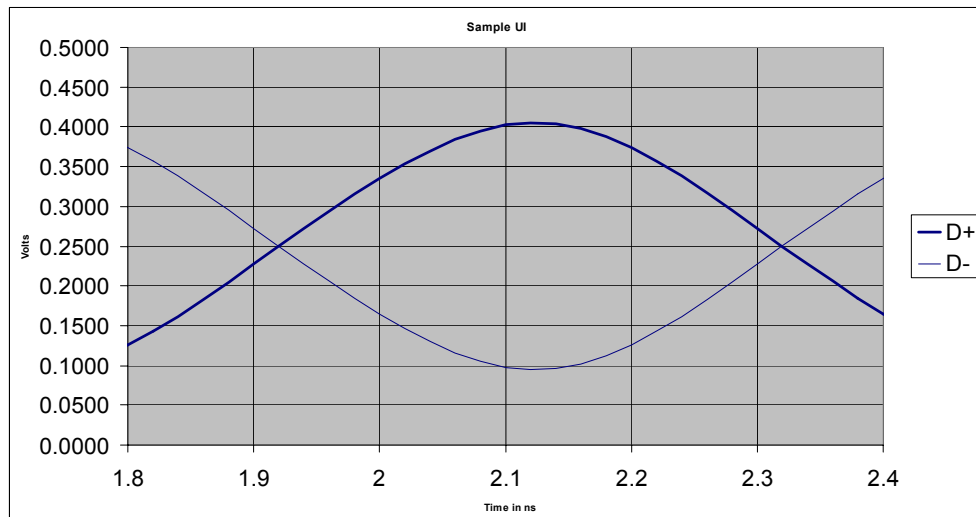
Figure 1: Threshold Crossings

In this example, the threshold crossings occur at 0.25 V. The time difference between the crossings of a repeating 101010 data pattern defines the period of the bit (a Unit Interval or UI).

The bit amplitude is defined as the absolute value of the difference between D+ and D-. It is generally highest near the middle of the bit and goes to 0 at the threshold crossings.

2.2 Bit Errors

The goal of the receiver is to sample the data at the point where the voltage and timing margins are as large as possible for all bit patterns in order to minimize the bit error rate.

Figure 2 shows an idealized symmetric bit where the maximum voltage and timing is in the bit center.
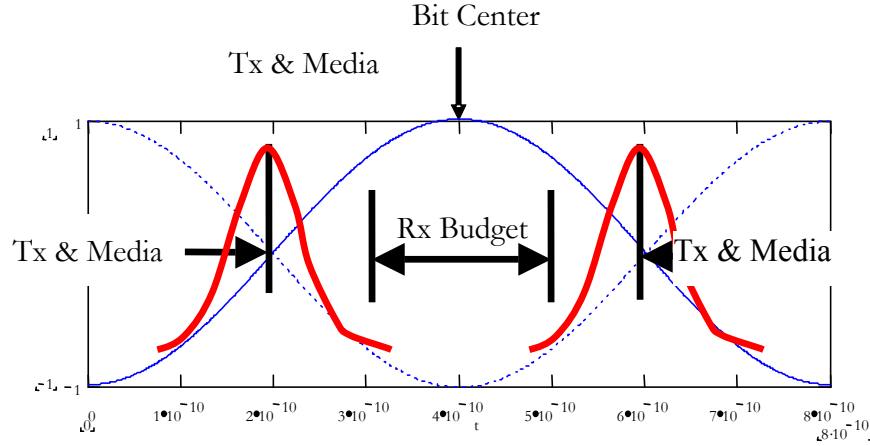
Figure 2: Ideal Sample Location is at the Bit Center

As the sample location deviates from the ideal sample location, the bit error rate increases. This can be seen qualitatively in the BER contour distribution of Figure 3. The coordinates of sampling time and voltage determine the sample point location. As shown by the arrow in Figure 3, the BER decreases along the line perpendicular to the contours the direction pointing to the center of the bit. For an optimal BER performance, the sampling point should be placed at a region where BER contour is smaller than 10^{-12} as defined in the *PCI Express Base Specification* [1].

Sampling Time (ts)
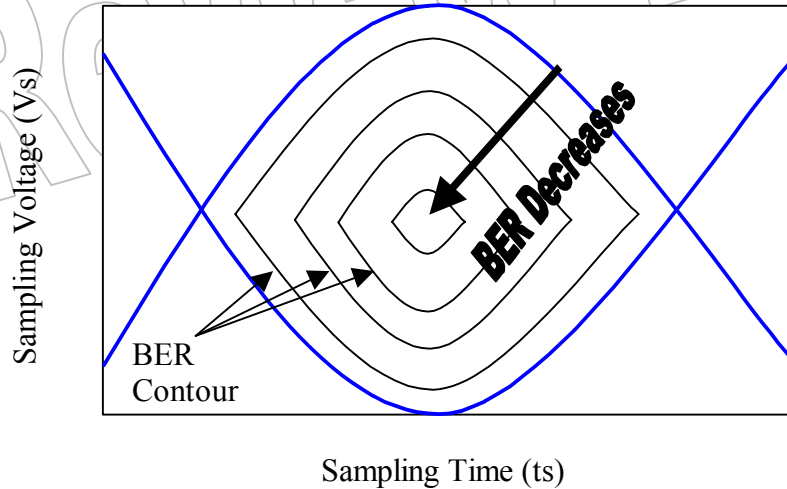
Figure 3: Bit Error Rate Contours

2.3 Timestamps of Threshold Crossings

Assuming the data consists of a repeating 101010 pattern, the measurement of each consecutive threshold crossing can be recorded and stored as an array. This is the timestamp array of the threshold crossings.

The timestamp array of the threshold crossings are equivalent to the accumulated phase of the data UI, also known as the absolute phase. In the case of the 400 ps UI, every 400 ps then represents one complete "cycle" or "revolution" and is equal to 2 pi radians. The absolute phase, Θ, starts at 0 and proceeds to grow unbounded, at the rate of

10

$$\Theta_n \propto nT, \quad n = 0, 1, \ldots, \infty .$$

For the ideal case where there is no phase jitter, the first UI starts at 0 radians and ends at 2 pi radians. The second UI starts at 2 pi radians and ends at 4 pi radians. The third UI starts at 4 pi radians and ends at 6 pi, and so on. Thus every UI can be thought of as one complete cycle or a complete "revolution" of the clock. T can be replaced by 2 π to get the equivalent "radians" from a UI period as in

$$\Theta_n = 2n\pi, \quad n = 0, 1, \ldots, \infty .$$

This is shown in Figure 4, where the straight line is a measurement where the threshold crossings are exactly 400 ps apart. The stem lines have a sinusoidal error term added to them. The Y axis is the absolute phase of the signals and is represented in ns and parenthetically in radians. The X axis is the ideal clock for each measurement and is given in ns.
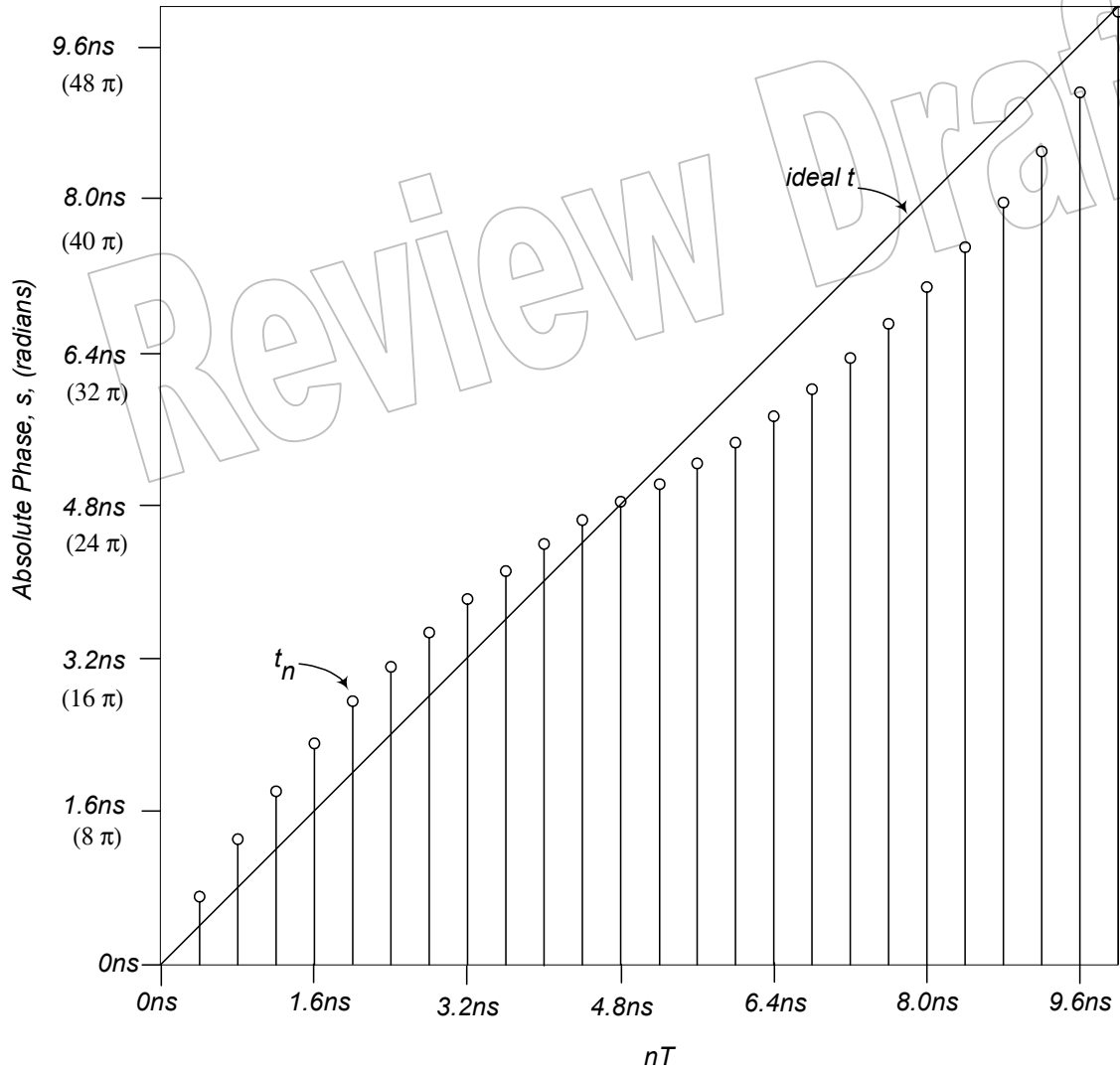
Figure 4: Absolute Phase vs. nT

2.4 Bit Period (T) and Unit Interval (UI)

The Unit Interval is defined in (2.1) as the difference in a measurement and the previous measurement.

$$(2.1) \qquad UI_m = t_m - t_{m-1}, \quad m = 0, 1, \ldots, \infty$$

The ideal bit period, T, is a mathematical convenience for developing the understanding of what jitter is and how to derive it. In practice, the bit period is extracted from the data itself. This extraction process produces the "recovered clock" that has the "recovered period," and is actually used in the calculations.

2.5 Phase Jitter (Φ)

The phase jitter is defined as the difference between the measured time and the ideal bit period T. Phase Jitter is an accumulation of the time error from the ideal time of n*T.

$$(2.2) \qquad \Phi_n = t_n - nT, \quad n = 1, 2, \ldots, \infty$$

Figure 5 shows an example of sinusoidal phase jitter at 100 MHz with an arbitrary magnitude of +- 450 ps shown on the Y axis.
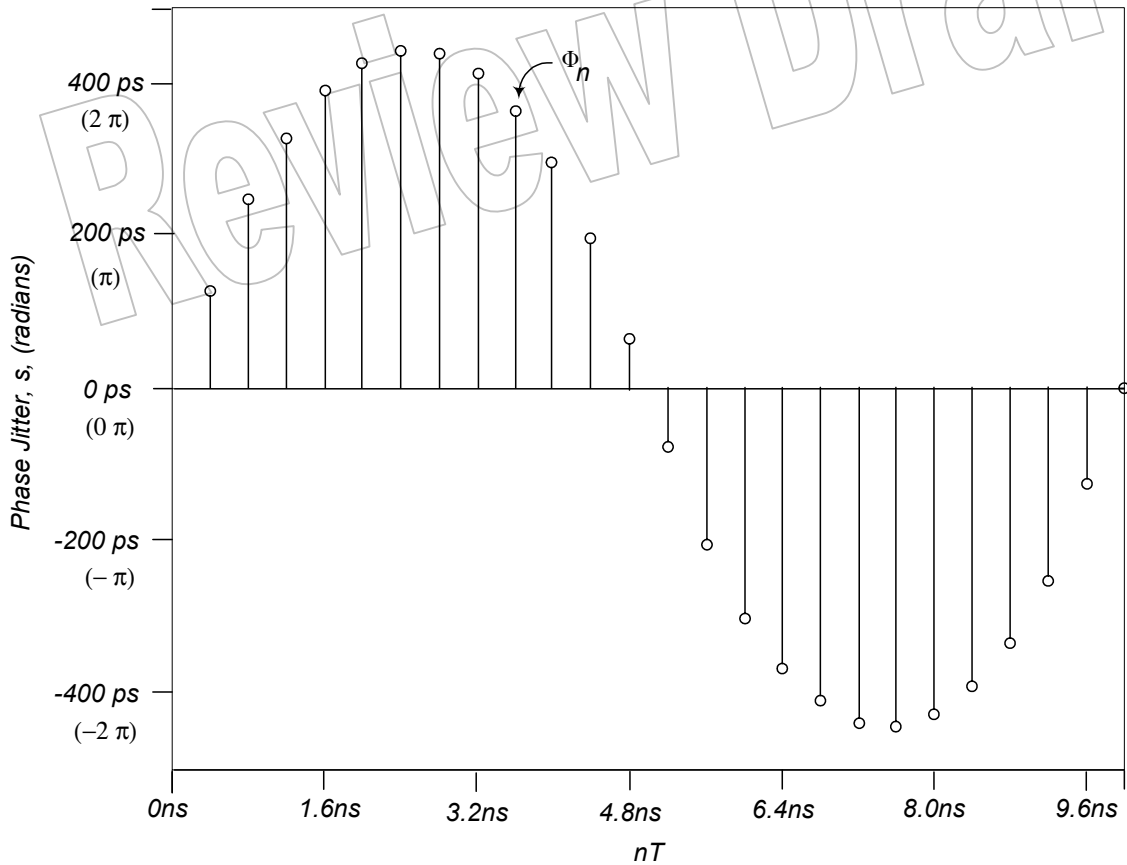
Figure 5: Phase Jitter vs. nT

12

2.6 Period Jitter (Φ')

The period Jitter (Φ') is the difference between the measured period and the ideal period and is defined as:

$$(2.3) \qquad \Phi'_n = (t_n - t_{n-1}) - T, \quad n = 1, 2, \ldots, N$$

Combining equations 2.2 and 2.3, it can be shown[1] that the period jitter, Φ', is also

$$(2.4) \qquad \Phi'_n = \Phi_n - \Phi_{n-1}.$$

This is the first difference function of the phase jitter, Φ.

Period jitter, Φ', is shown in Figure 6, where T is the ideal 400 ps. The Y axis shows the magnitude of the period jitter in ps and, parenthetically, in radians.
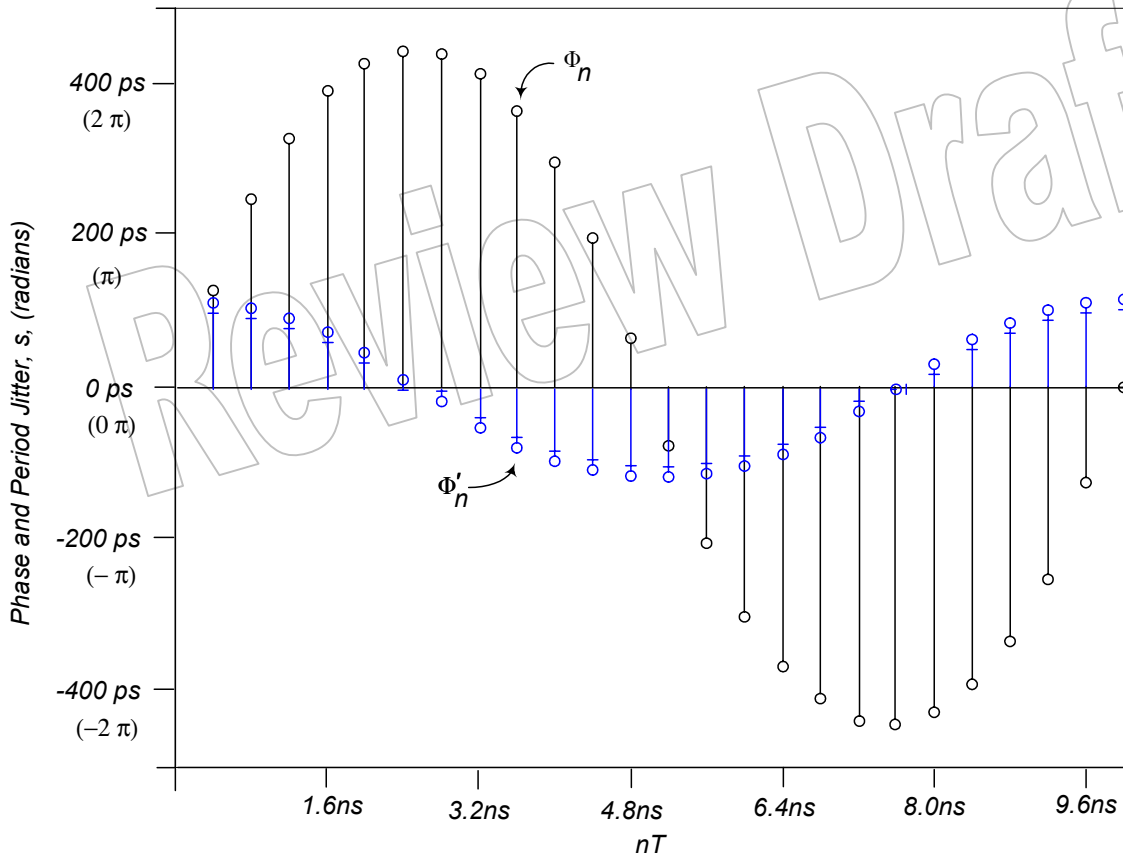
Figure 6: Phase and Period Jitter vs. nT

---

[1] For instance, see reference [5].

2.7 Cycle to Cycle Jitter (*Φ''*)

The cycle to cycle jitter is the difference between consecutive bit periods and is defined as:

$$(2.5) \qquad \Phi_n^{''} = (t_n - t_{n-1}) - (t_{n-1} - t_{n-2}), \quad n = 1, 2, \ldots, N$$

Combining equations (2.4) and (2.5), it can be shown that this is also

$$(2.6) \qquad \Phi''_n = \Phi'_n - \Phi'_{n-1}, \qquad n = 1, 2, \ldots, N$$

and is the first difference function of **Φ'**, or the second difference function of **Φ**. This is shown in Figure 7. The Y axis shows the magnitude of the cycle to cycle jitter in ps and parenthetically in radians.
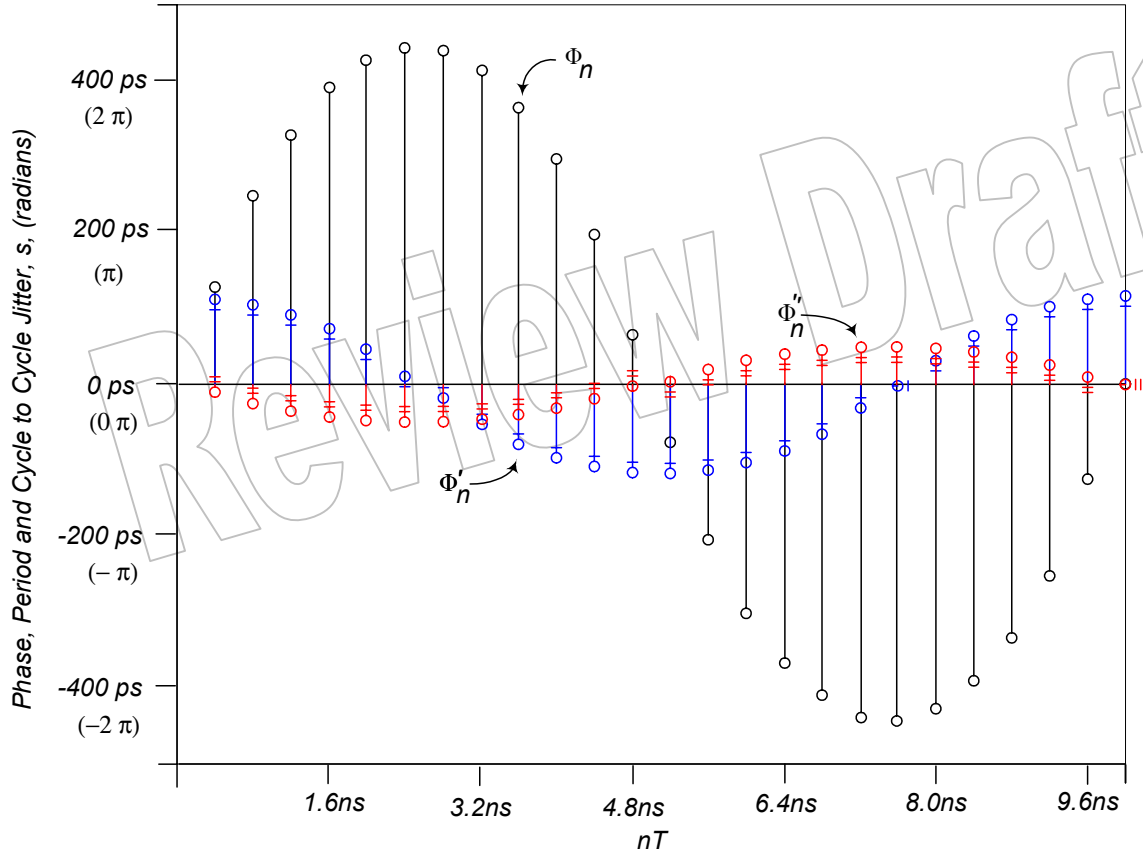
Figure 7: Cycle to Cycle Jitter and Difference Function of Period Jitter vs. Sample Number

2.8 Jitter Relationships

All three jitter types are shown in Figure 7 for a phase jitter magnitude of 450 ps (2.25 π radians) at a frequency of 100 MHz. The first and second difference function from **Φ** to **Φ'** to **Φ''**, respectively, can be seen in Figure 7.

It follows that all three designations of jitter, phase jitter, $\mathbf{\Phi}$, period jitter, $\mathbf{\Phi'}$, and cycle-to-cycle jitter, $\mathbf{\Phi''}$, are different ways to represent the same physical behavior of the clock signal. Given a continuous record in time of jitter in any one of the forms, the other two can be derived.

Without a continuous record in time, conversion between the different representations of jitter is not possible. For example, if only a peak-to-peak value for the period jitter is known, then determining the peak-to-peak value of phase jitter or the peak-to-peak value of the cycle-to-cycle jitter is not possible.

2.9 Qualitative Discussion of Jitter

For simplicity, assume that the data transmitted is a repeating 101010 pattern. This discussion also applies to an arbitrary data pattern by taking into account the missing transitions as appropriate.

In the definition of phase jitter, we subtracted where the threshold crossing was measured from where the threshold crossing should have been. In this sense, the phase jitter is a relative "distance" from an ideal location, where distance is measured in radians.[2]

An analogy can be drawn with rotational motion:

❑ Phase jitter can be considered the relative distance that the actual (measured) phase has moved from the absolute phase of the ideal clock. In other words, it is the number of radians that the phase of the clock is vs. where it should be.

❑ Period jitter is the speed at which the phase is changing.

❑ Cycle to cycle jitter is the acceleration of the phase from or to the ideal phase.

In all receiver architectures, the goal of the clock recovery is to align the sampling clock to the incoming data stream. Certain data recovery architectures, such as phase interpolating type, rely on the phase of the reference clock to grossly align the sample clocks, and then have other mechanisms to provide fine adjustment of the reference clock phase to the incoming data phase as discussed in [3]. For these types, the presence of phase jitter and the response of the components to phase jitter is critical for the proper operation of the system.

Important Point: It is crucial that the previous section is well understood. Jitter considerations for serial buses are different from the jitter considerations for parallel buses. Parallel buses are mainly concerned with cycle-to-cycle and period jitters, since these affect the bus setup and hold timings. Phase jitter has little effect on parallel buses. As we shall see in the next sections, serial links are primarily concerned with the phase jitter; cycle-to-cycle and period jitter are secondary.

---

[2] In this document, the phase jitter magnitude will be expressed in units of either radians or seconds and will be obvious from the context.

3 Data Recovery Circuits

3.1 Clock Data Recovery Circuit (CDR)

We will consider two types of clock/data recovery circuits (CDRs) for the PCI Express Link, the digital based and PLL based. The data recovery operation requires looking at the incoming data and recovering the phase and frequency of the incoming data stream, after accounting for missing threshold crossings. A clock is locally generated that matches the data phase and frequency and is used to sample the data at the receiver.

3.1.1 Digital Based

The Phase Interpolator and Oversampling CDRs use a digital mechanism to achieve phase alignment of the clock with the data. They belong to the class of digital CDRs and are not easily modeled. They also have limited ability to track changes in frequency, and, in general, rely on the systems common reference clock system and internal reference PLLs to recover the frequency information. The ability to track data is limited by the edge density and the feedback loop of the system. In general, these devices will be modeled by a relatively low bandwidth, on the order of 1-2 MHz.

3.1.2 PLL Based

The PLL based CDR does not use the common reference clock to recover the phase or the frequency. It looks exclusively at the incoming data and adjusts phase and frequency accordingly, while accounting for missing bit transitions. It is has a well-known transfer function and is easily modeled.

3.2 Receiver Eye Closure

Figure 8 shows the relationship of the reference jitter to the eye closure seen at the receiver. This will be called Rx eye closure. This is the diagram for a Phase Interpolator type clock recovery. A description of this type of clock recovery can be found in [3].
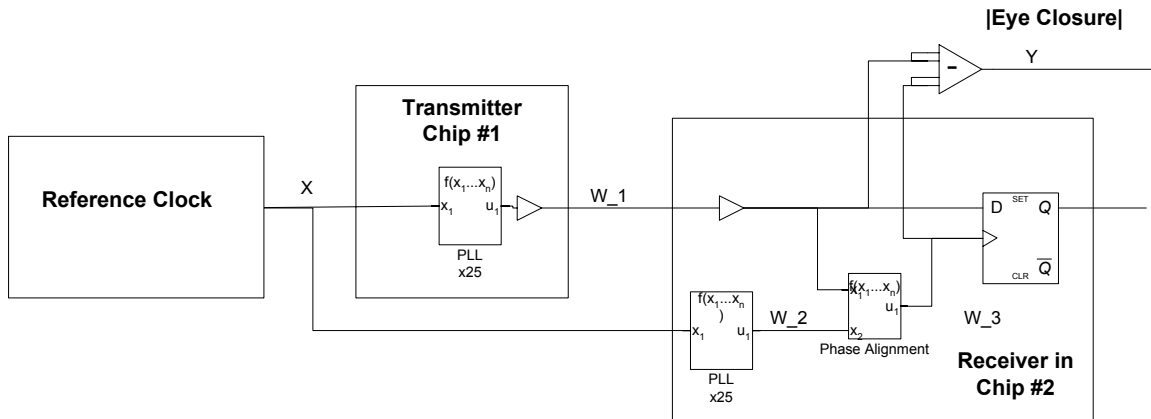
Figure 8: System Model for Reference Clock Input X to Eye Closure Y

For this architecture, the receiver sees phase jitter not as the absolute value of W_1 but as the relative difference between W_1 and W_3. The closure is given as Y and consists of all the sources of jitter.

This paper addresses only the jitter that is caused by the mismatch in the PLLs in response to jitter content of X and the bandwidths of the PLLs. There are many other sources of jitter that consume the timing budget at the receiver; these will be explored in some detail in a future paper.

4 System Transfer Functions

In this section, we will define the model and system transfer function for the PLLs. A review of system transfer functions can be found in reference [4], from which we adopt the notation conventions used in this advisory.

More information on PLL transfer functions and modeling PLLs in the s-domain can be found in reference [6].

4.1 PLL Transfer Function

The input signal to a PLL is

$$(4.1) \qquad V_{in}(t) = A_{in} \sin(\omega_{in} t + P_{in}(t))$$

and the output signal of the PLL is

$$(4.2) \qquad V_{out}(t) = A_{out} \sin(\omega_{out} t + P_{out}(t)).$$

Here ω_{in} is the input carrier frequency and ω_{out} is the multiplied output frequency, both given in radians per second. The terms $P_{in}(t)$ and $P_{out}(t)$, expressed in radians, represent the absolute phase in time of the input and output signals and are sometimes referred to as "excess phase" or, in this discussion, phase jitter. In a properly designed PLL, the input and output frequencies and amplitudes do not change with time. The phase signals $P_{in}(t)$ and $P_{out}(t)$ are a function of time, so the PLL has a phase transfer function. In the s domain, the phase transfer function of the PLL $H(s)$ is given by

$$(4.3) \qquad H(s) = \frac{P_{out}(s)}{P_{in}(s)}.$$

$P_{out}(s)$ and $P_{in}(s)$ are the Laplace transforms of $P_{out}(t)$ and $P_{in}(t)$. The PLL is therefore a control system that transfers the phase modulation at its input to its output. Knowing the transfer function and the input phase, the output phase can be calculated.

Actual PLLs used in typical CMOS process are often third-order or higher functions of s and exhibit other slight nonlinearities. However, they can all be approximated as a second-order transfer function. This discussion exclusively uses the simpler second-order transfer function as an approximation.

4.2 Second Order PLL Transfer Functions

The second order model is based on the active proportional integration control loop with the transfer function given by:

$$(4.4) \qquad H1(s) = \frac{2s\zeta\omega_n + \omega_n^2}{s^2 + 2s\zeta\omega_n + \omega_n^2}$$

In equation (4.4), ζ is the damping factor, and ω_n is the natural frequency. This function is not meant as a requirement for an implementation. It is used as a bounding function for modeling purposes to establish the lower limit for the f_3 dB frequency and the maximum peaking. In practice, the actual transfer functions are likely third order and higher.

The translation between natural frequency ω_n and the 3 dB frequency is given by

$$\textbf{(4.5)} \qquad \omega_{3dB} = \omega_n \sqrt{1 + 2\zeta^2 + \sqrt{(1 + 2\zeta^2)^2 + 1}}\ .$$

An upper allowed bound of 3 dB of peaking was chosen for analysis in order to limit the resultant worst case gain resulting in ζ of 0.54 minimum. This results in the transfer functions shown in Figure 9 for the example where the 3 dB frequency (f_3 dB) is 15 MHz.
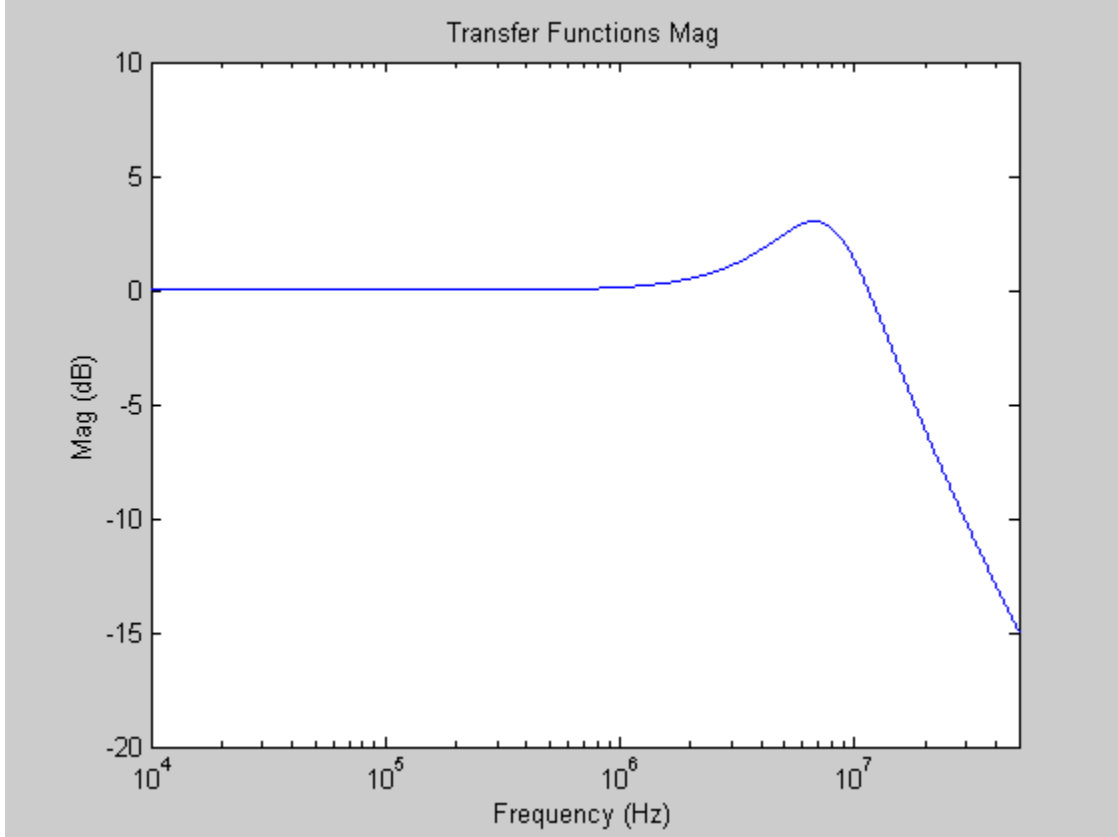
Figure 9: Transfer Functions for f_3dB = 15 MHz and Zeta = 0.54

4.3 Digital CDR System Transfer Function

Referring to the system model diagram in Figure 10, the input phase jitter signal x(t) is transformed to the S domain as X(s) and is acted on by the transfer functions of the system model as discussed in Section 3.2. The system model can be simplified and drawn as shown in Figure 10. The derivation of the phase interpolator function is given in appendix B.

20
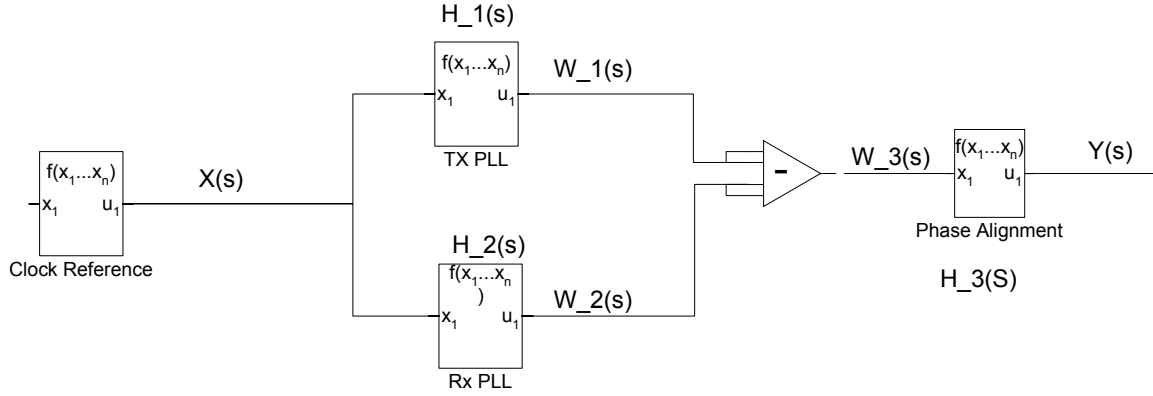
Figure 10: Simplified System Model for Input Jitter to Receiver Eye Closure

Here X(s) is the input signal representing the input phase jitter; H_1(s) is the transfer function of the Tx PLL; H_2(s) is the transfer function of the Rx PLL; H_3(s) is the high pass transfer function of the data recovery circuit; and Y(s) is the output signal that represents the Rx eye closure at the receiver caused only by the jitter on the reference clock propagating through the system.

The total transfer function of this system is given by

$$\textbf{(4.6)} \qquad H_t(s) = [H_1(s) - H_2(s)] * H3(s).$$

The output signal can then be calculated for any *X(s)* as *Y(s) = Ht(s) * X(s)*. Note that there is a phase delay that is not considered here and will be discussed later.

The transfer function H_3(s) is the response of the CDR circuit. In general, H_3(s) should have a high pass frequency response with minimal peaking in the pass band. This allows the CDR circuit to track phase jitter while rejecting the lower frequency SSC and transport delay effects.

The transfer function of H3 is a single pole high pass filter that is given by

$$\textbf{(4.7)} \qquad H_3(s) = \frac{s}{s + \omega_3}$$

where ω_3 is simply

$$\omega_3 = 2 * \pi * f3\_3dB .$$

It is clear that if *H1(s)* and *H2(s)* are perfectly matched; no Rx eye closure occurs regardless of the phase jitter content of the reference clock. When there is a mismatch in the transfer function, an Rx eye closure occurs that is dependent on the phase jitter content of the reference clock and the difference of the transfer functions. In practice, the control of the transfer function is inexact and the variance is large even between two devices of the exact same design, process, and manufacturing lot.

Figure 11 gives the transfer function of reference clock phase jitter to receiver eye closure. All of the phase jitter frequencies present in an input signal X(s) in the frequency range of ~ 400 kHz to 10 MHz will cause the eye to close at Y(s), with attenuation occurring outside this range. Due to the peaking, there is gain in this range, and the phase noise gets amplified. Only a portion of the phase noise outside this range contributes to the Rx eye closure at Y(s).
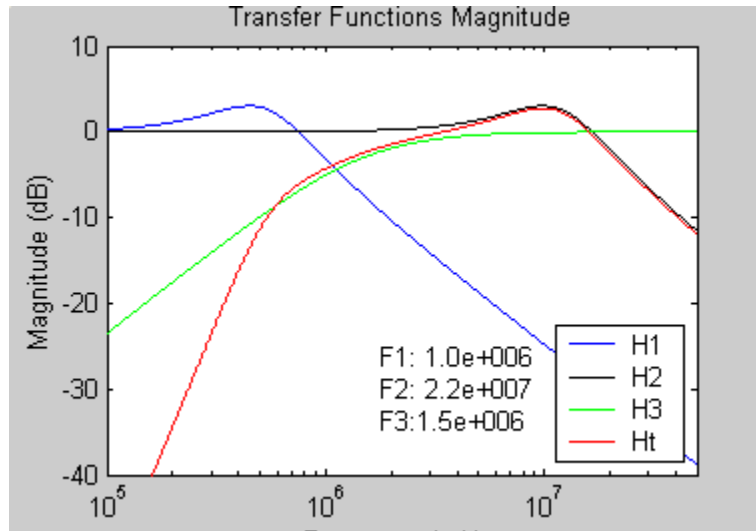
Figure 11: Input Jitter to Rx Eye Closure for F1=1 MHz, F2=22 MHz and F3=1.5 MHz, Zeta = 0.54

This means that low frequency on the clock reference phase jitter is tracked equally by both the Tx and Rx devices and does not contribute to Rx eye closure. Jitter at ~ 400 kHz, the f_3 dB lower corner of the bandpass, transfers ~0.707 of its magnitude to Rx eye closure. Jitter in the range of approximately 400 kHz to 10 MHz is amplified and contributes directly to Rx eye closure.

Setting the minimum f_3 dB frequency to 7 MHz and the maximum f_3 dB set by the clock rate at 22 MHz, we get the transfer function shown in Figure 12. This is the transfer function that is to be applied to the reference clock phase jitter X(s) to produce Y(s) in order to model this range of PLLs in the Tx and Rx. The upper limit of 22 MHz was chosen as the theoretical limit of stability for a PLL with a 100 MHz input reference. The resulting peak value in y(t) is the peak jitter output of the reference clock.
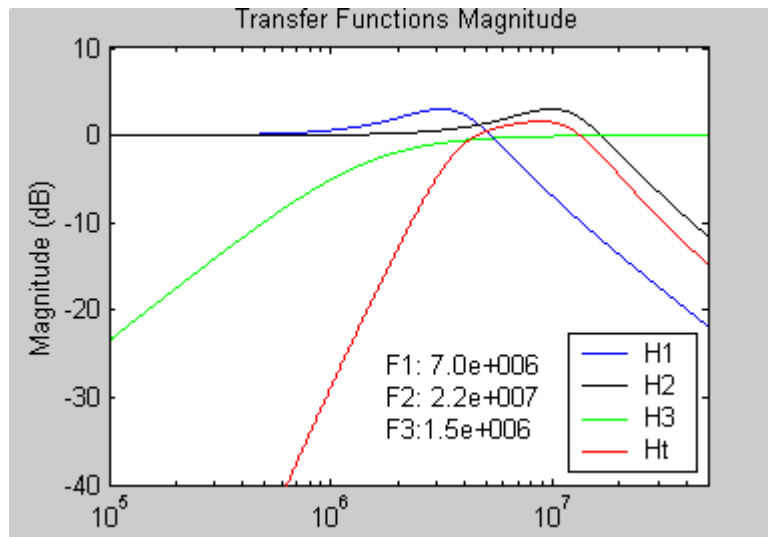
Figure 12: Input Jitter to Rx Eye Closure for F1=7 MHz, F2=22 MHz and F3=1.5 MHz, Zeta = 0.54

22

4.4 PLL CDR System Transfer Function

A simplified PLL clock recovery system model is shown in Figure 13 where X(s) is the input signal, H_1(s) is the transfer function of the Tx PLL, H_2(s) is the transfer function of the Rx PLL, and Y(s) is the output.
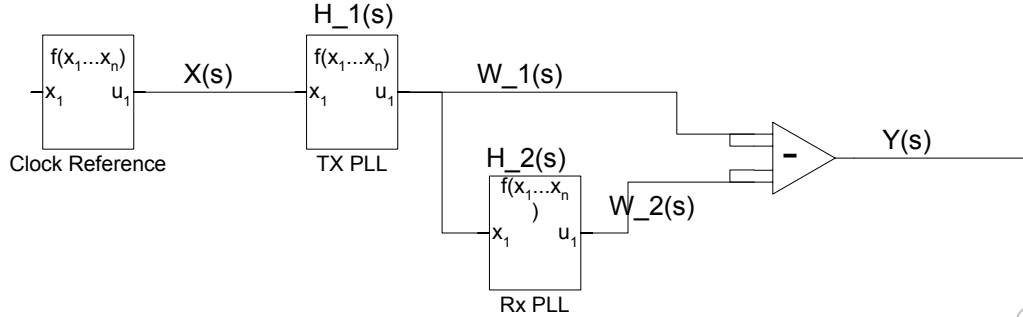
Figure 13: Simplified System Transfer Function for a PLL Based Clock Recovery Circuit

The total transfer function is then

$$ (4.8) \qquad H_t(s) = H_1(s)\left(1 - H_2(s)\right). $$

In this case, a solution that minimizes H(s) is when H1 is smaller than H2. Using the PLL Clock Recovery clock transfer function and the frequencies of f_1 = 7 MHz and f_2 = 22 MHz, the total system transfer function is shown in Figure 14.
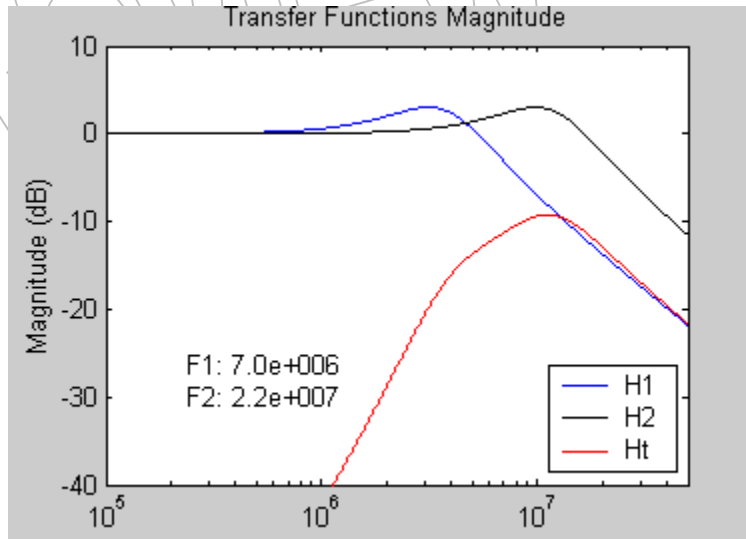
Figure 14: Input Jitter to Rx Eye Closure for a PLL CDR with f_3 dB at 7 MHz and 22 MHz

Setting f_1 to 22 MHz and f_2 to 7 MHz, we get the results shown in Figure 15.
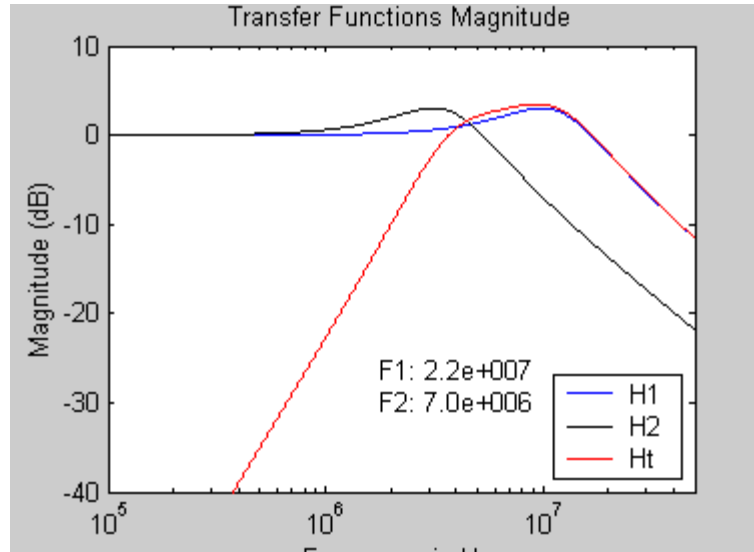
Figure 15: Input Jitter to Rx Eye Closure for a PLL CDR with f_3 dB of 22 MHz and 7 MHz

This results in the similar performance as the digital clock recovery model. Clearly the case of having the Rx bandwidth higher than the Tx bandwidth provides better overall jitter rejection, but as previously shown, this comes at the expense of interoperability with digital clock recovery mechanisms. Obviously, to minimize the effect of the reference clock noise, it is an advantage for the PLL based CDR to have a high Rx bandwidth.

4.5 Delay and System Phase Response

The delay and phase response also contributes to timing error. One component of the phase response is the fixed phase delay due to the routing length differences of the 100 MHz clock and the data channel, as shown in Figure 16. Another component of the phase response is the differences in the transfer functions of the Tx and Rx PLLs. Additional delay may come from the insertion delay of the chips themselves. This is all grouped into the category of phase difference.
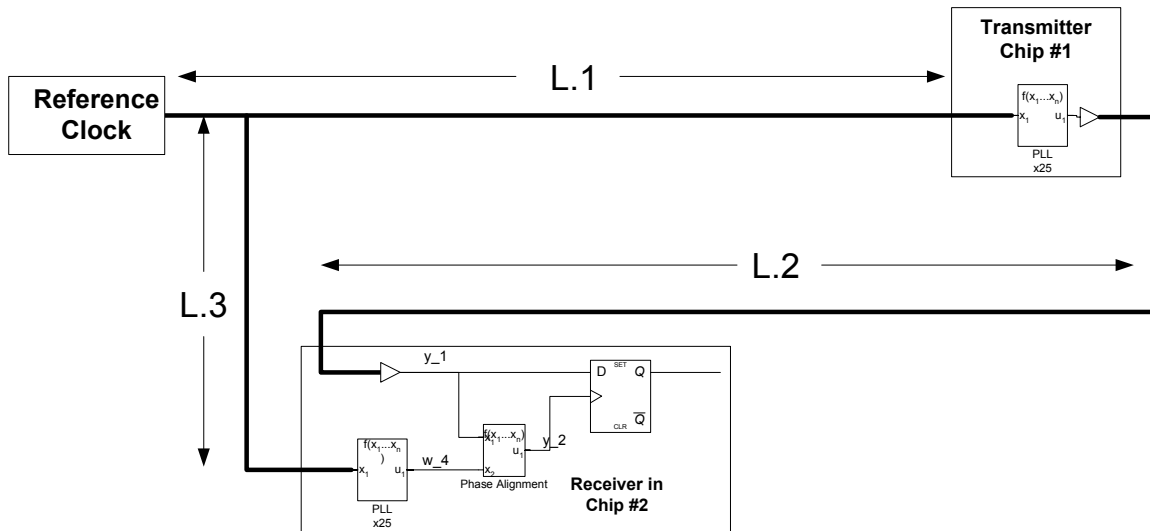
Figure 16: Gain Induced by Routing Phase Delay

The phase difference can cause additional Rx eye closure. The Rx eye closure can be modeled by multiplying either H1 or H2 by exp(-s*t_delay), where t_delay is the maximum time for the interconnect phase delay and is the relative difference in flight time. This is shown in Figure 16 as [(L1 + L2) – L3] / v_prop, where v_prop is the propagation velocity of the signal. An example of the phase delay effect can be seen in the transfer function as shown in Figure 17.
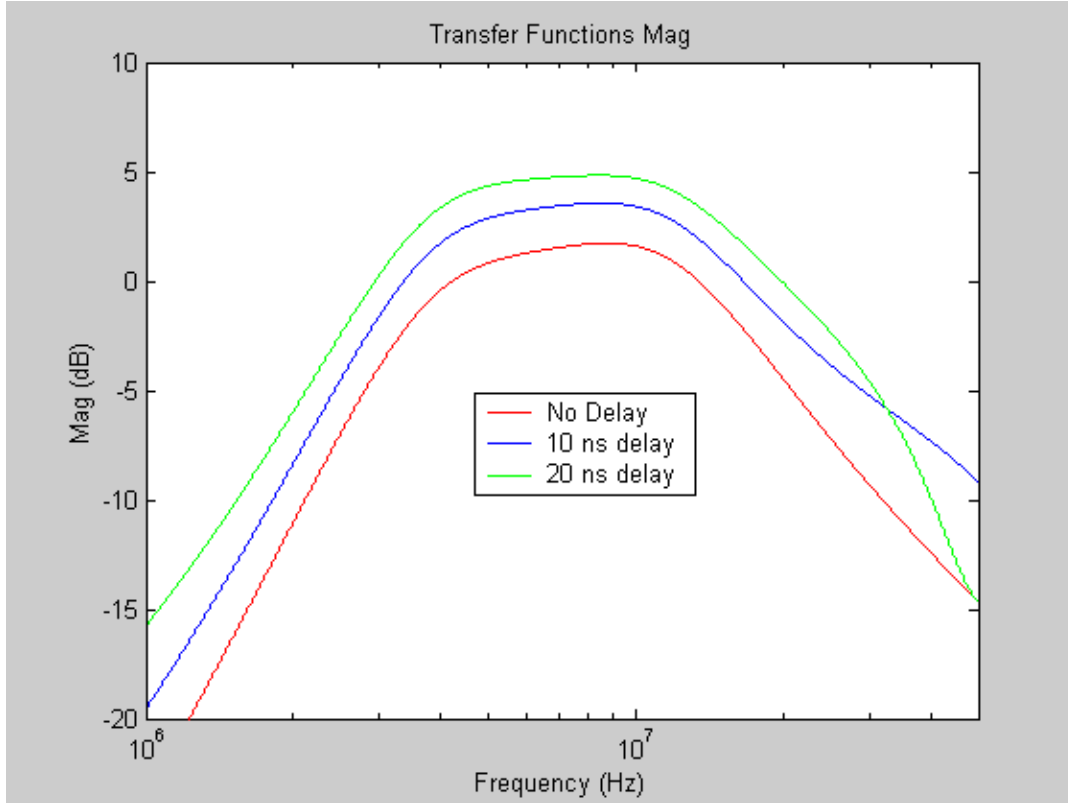
Figure 17: Effect of Phase Delay on Transfer Function

There is an assumption that the total delay difference for PCI Express does not exceed 30 ns. As the delay between the paths increase beyond 30 ns, the reference clocks become uncorrelated at the higher frequencies and the delay effects become dominant in the transfer functions. In the case of delay beyond 30 ns, the Rx eye closure can exceed 6 dB at the lower frequencies. The minimum response of H3 serves to keep the delay effects from dominating the low frequency response. This is shown in Figure 18 where the delayed transfer function, shown in green, rolls off to the left at -40 dB/decade due to the addition of H3. Without H3, the delay becomes dominant at the low end and the roll-off is -20 dB/decade.
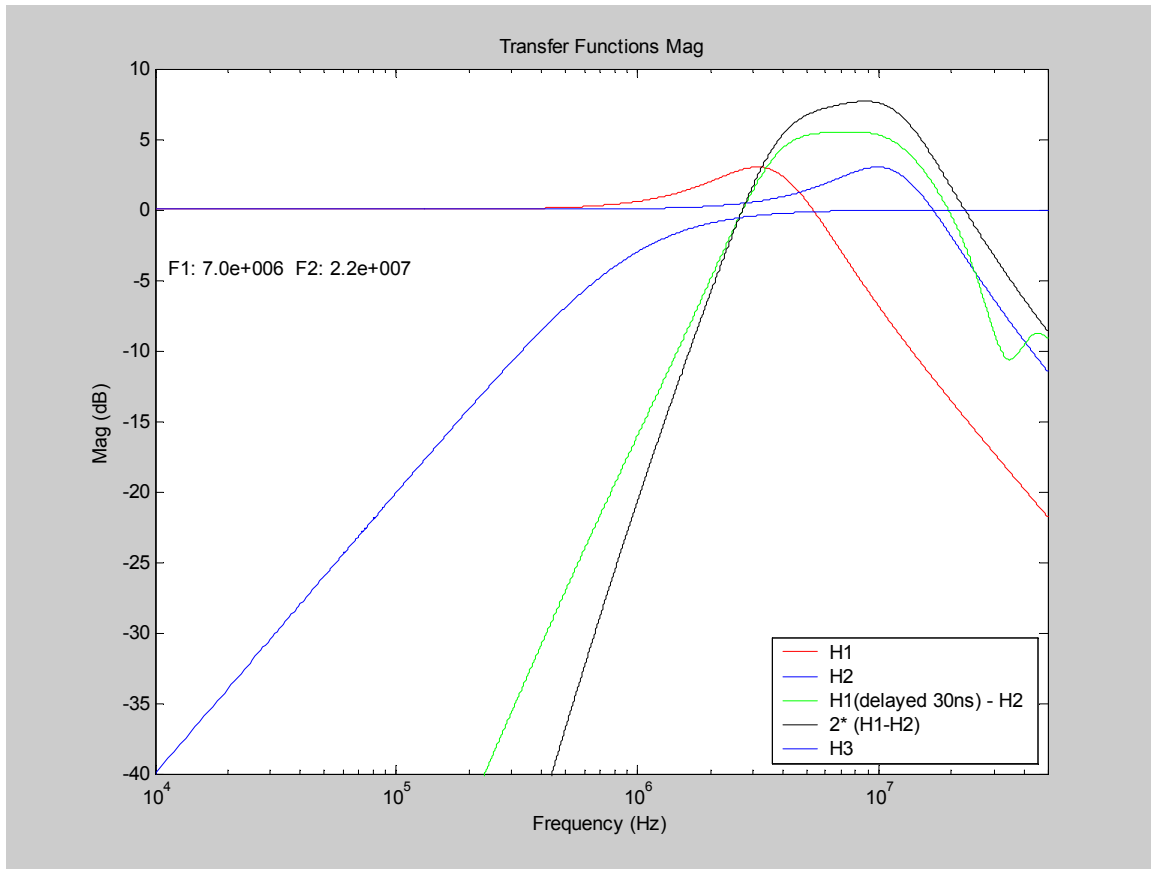
Figure 18: 30 ns Delay Effect on Rx Eye Closure

4.6 Platform Reference Clock Analysis

The transfer of the platform reference clock jitter to Rx eye closure can be calculated. Code for this analysis has been developed in Matlab*, the source code is given in Appendix A.

An example application of the code follows. The phase modulation of a clock reference chip is shown in Figure 19, where the 33 kHz SSC which dominates the phase modulation can be seen. The SSC component has a magnitude of ~10 ns.
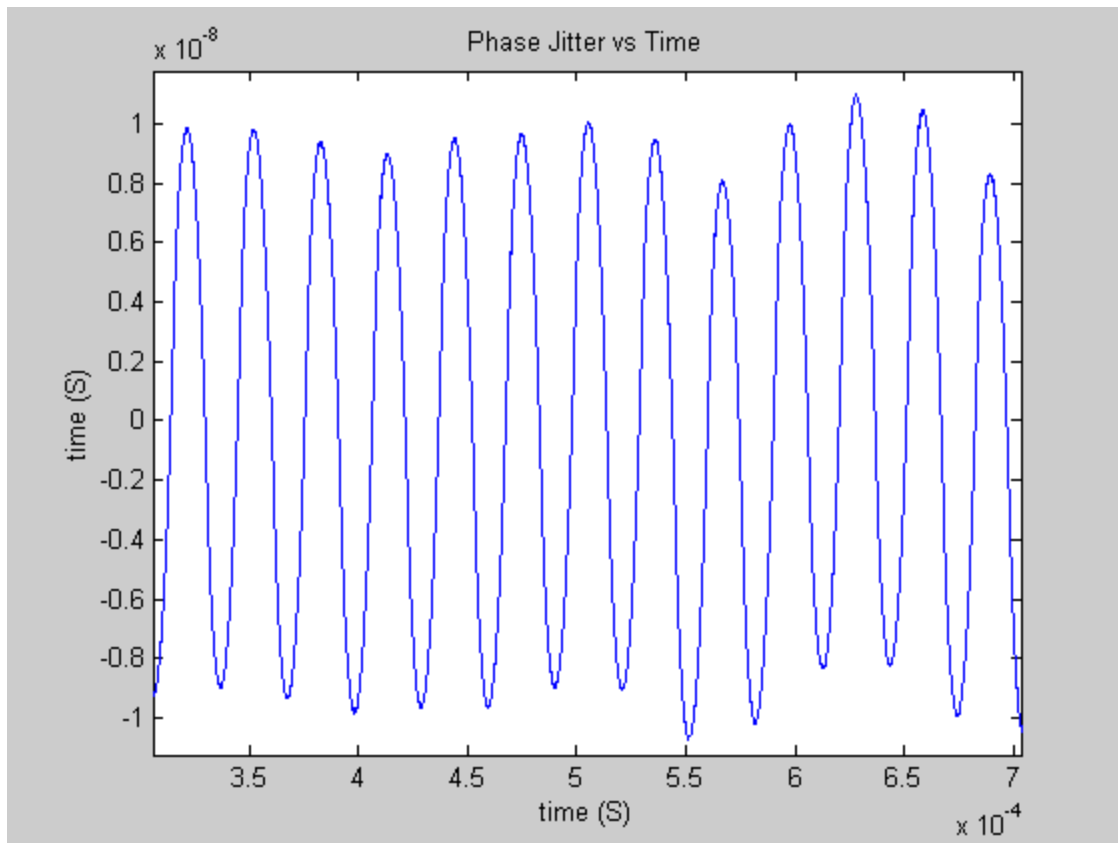
Figure 19: Time Domain Phase Jitter

The spectrum of this reference clock before and after the difference function is applied is shown in Figure 20.
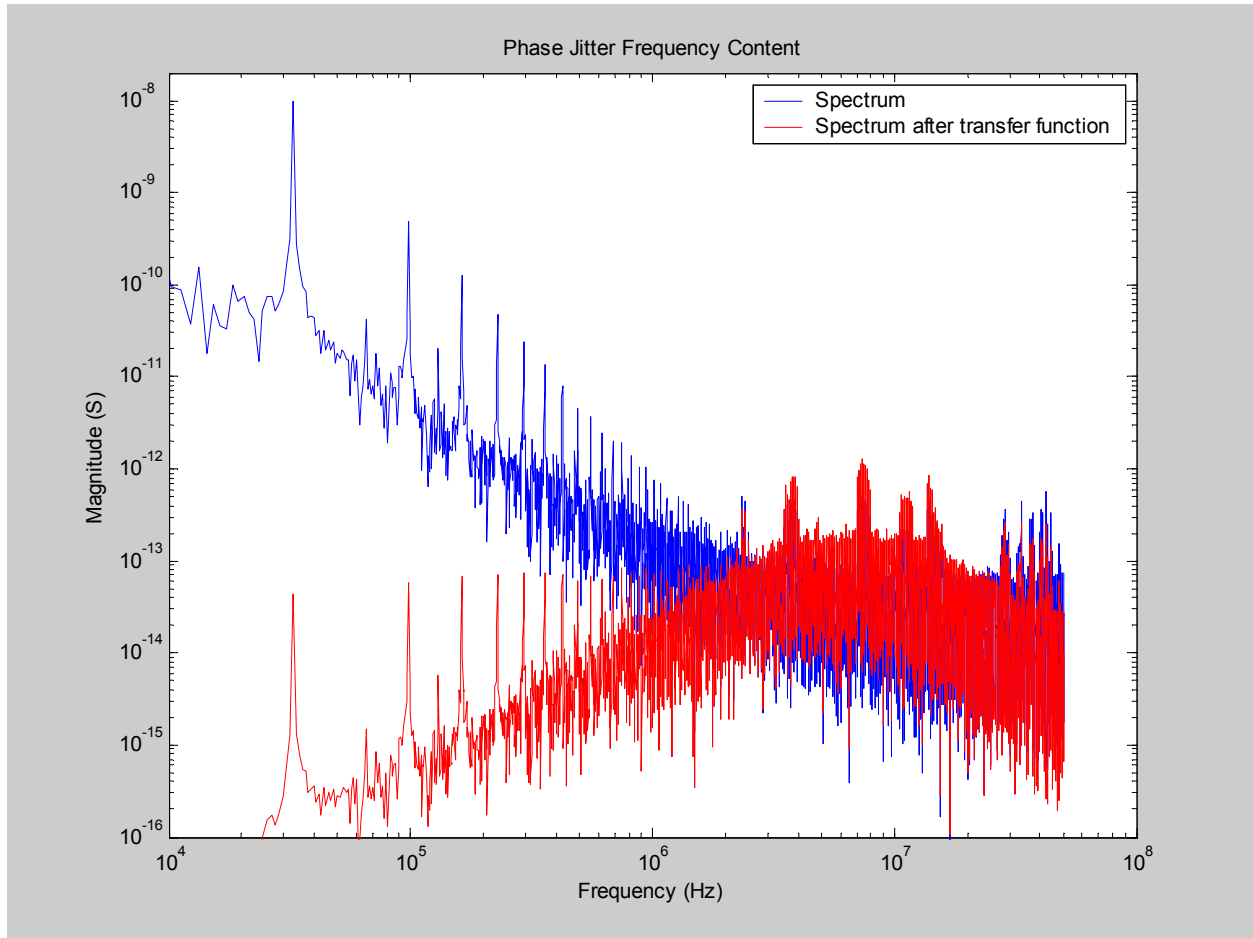
Figure 20: Phase Jitter Spectrum Before and After the System Transfer Function

The inverse FFT of the spectrum after the difference function of the Rx eye closure is shown in Figure 21. Here the total eye peak to peak closure for this particular sample is on the order of 80 ps.
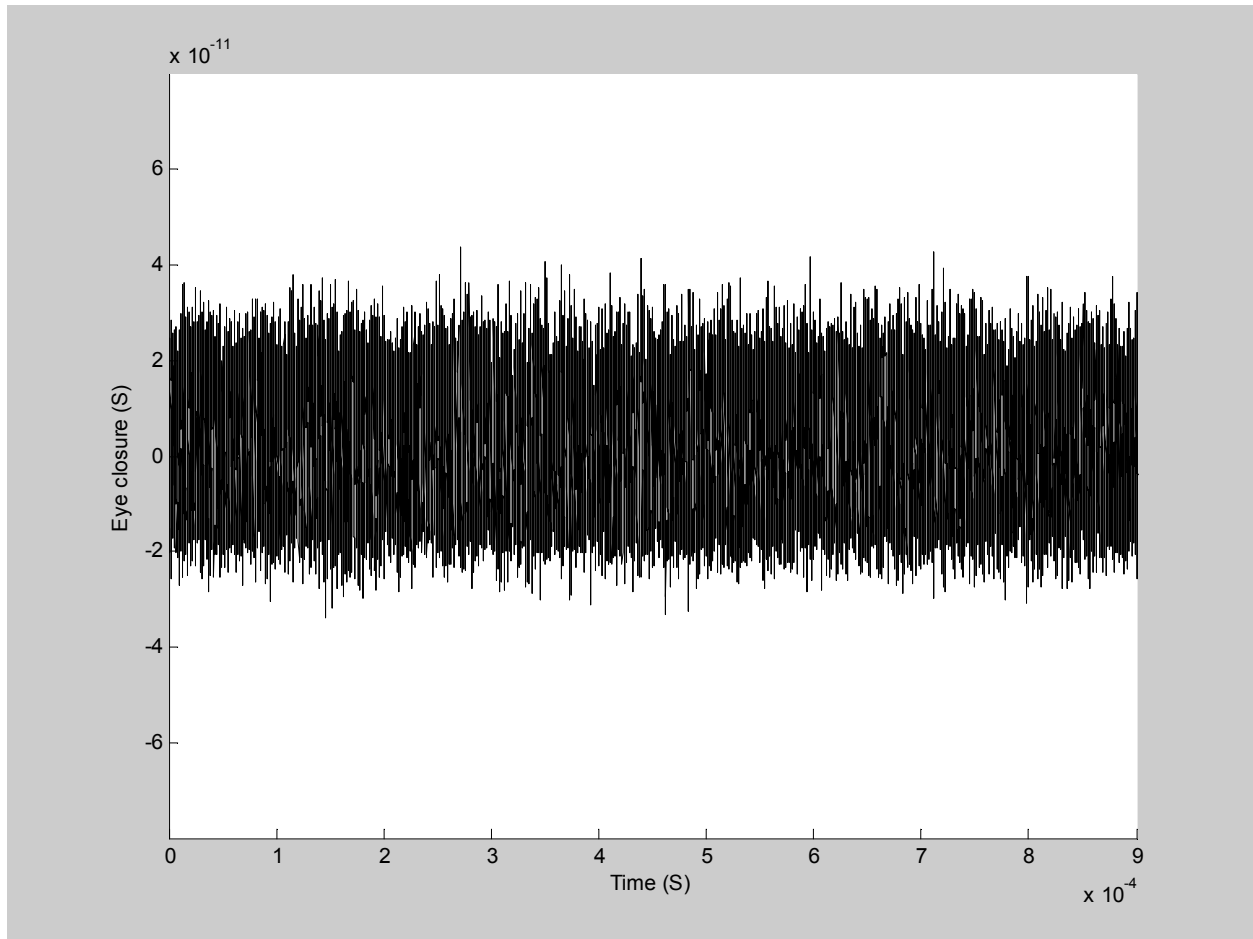
Figure 21: Time Domain of the Rx Eye Closure for 10^5 samples

5 Measurement Eye Closure

In the previous sections, we developed the model for calculating the phase jitter, y(t), at the sampling device which we have called the Rx eye closure. This is a comprehensive model that accounts for the relative behaviors of the transmitter and the receiver. There is another type of eye closure defined by the *PCI Express Base Specification* [1]. This is the eye closure measured by the 250/3500 method as defined in the specification and applies to the eye closure created by the Tx and media output into the 250/3500 UI method. This will be referred to as the measurement eye closure.

5.1 Phase Jitter to Measured Eye Closure

In this section, we will show that Phase Jitter, as defined in equation (2.2), is a direct measure of the peak to peak measurement eye closure for a particular sample. This relationship is most easily shown with an example using an ideal clock. The actual phase jitter calculated depends on the recovered clock used; the model will be extended to include this in the following section.

In the following example, we use the definition of phase jitter to get the measured Tx eye closure. In Figure 22, we see some records of an ideal clock and fictitious measured data threshold crossings using a repeating 101010 data pattern.
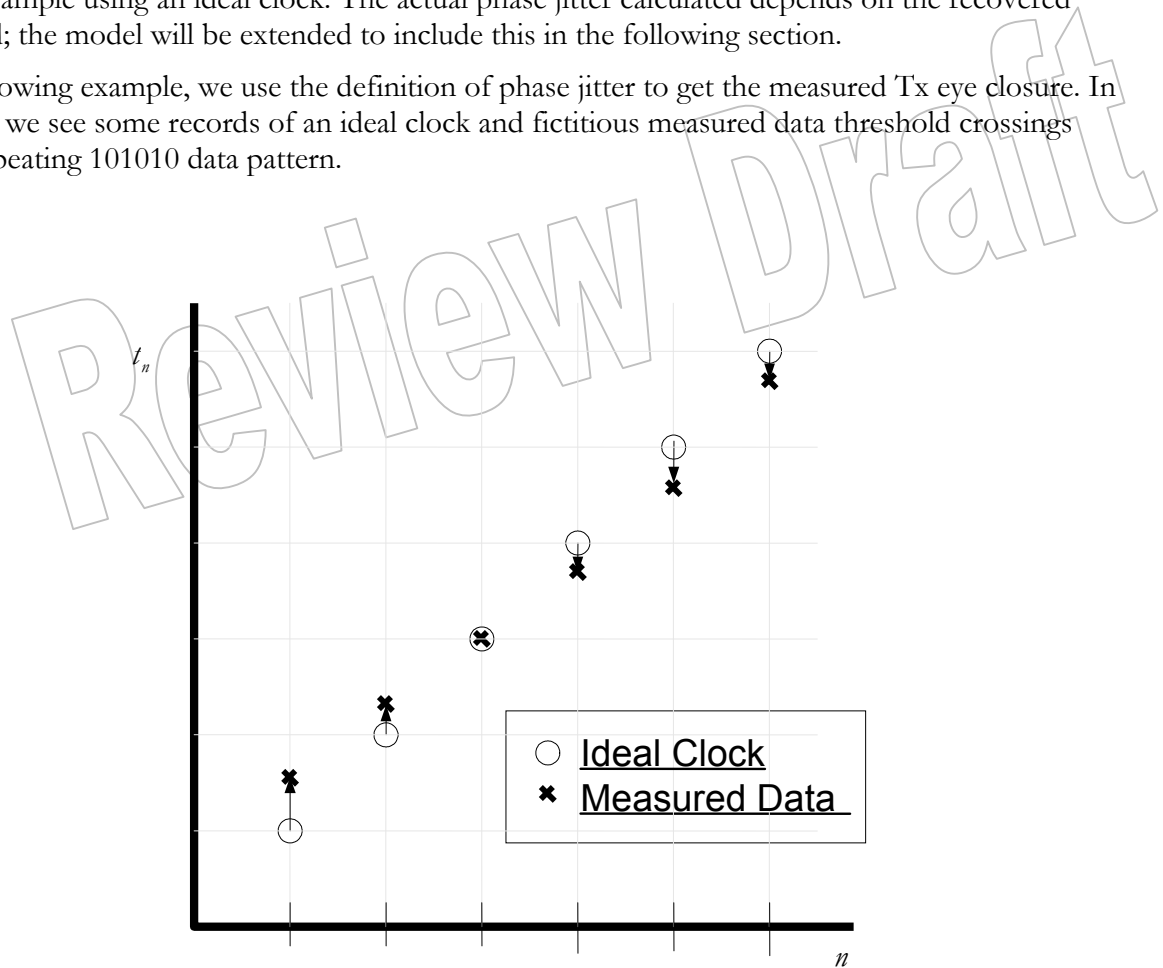
Figure 22: Example of Measured Record

We can convert this record into phase jitter by applying equation (2.2). We label the positive phase jitter measurements ϕ_{n+} and the negative measurements are labeled ϕ_{n-}. This is shown in Figure 23.
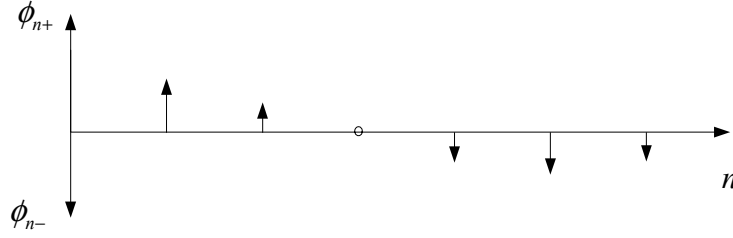
Figure 23: Example of Phase Jitter

The arrows pointing down are where the phase lags the ideal clock; the arrows pointing up are where the phase leads the ideal clock. We can plot this using the ideal clock as a reference template by collecting all the phase jitter signals and plotting them on the edge of the ideal period, as shown in Figure 24.
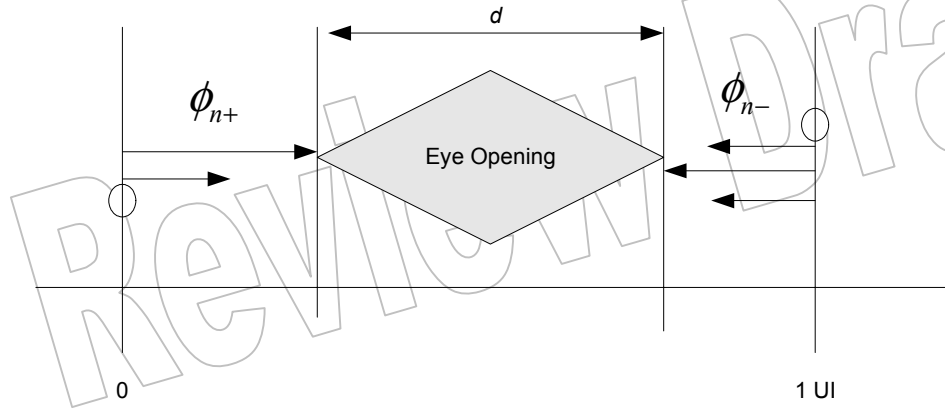
Figure 24: Phase Jitter to Eye Opening, "d"

The eye opening is now shown by the shaded diamond with a magnitude of *d*. In this fictitious example, we are calculating the eye opening over 6 UI. This is the jitter eye mask that is referenced in the PCI Express specification [2].[3]

This shows that the eye opening is the recovered clock period (in this case using an ideal period) minus the difference between the maximum positive and maximum negative phase jitters over the measurement interval, or

$$(5.1) \qquad d = T - \left(\max(\phi_{n+}) - \max(\phi_{n-})\right) \qquad n = 1,2,\ldots,N$$

where *d* is the difference between the maximum positive and maximum negative phase jitters taken over N unit intervals. In this case, *d* is the "eye opening", and *T-d* is the "eye closure." We will

---

[3] Note: This description does not account for the more complicated issue of the jitter median. Our simplified model here assumes a symmetric distribution of threshold crossings; this assumption does not alter the relationship of phase jitter to eye closure. For a complete description of the eye measurement including the jitter median and the "jitter median to max" specification, see the *PCI Express Base Specification* [1].

define the eye closure to be $Ec = T{-}d$, since this is relevant to the transfer function that we wish to derive. Substituting this expression for eye closure into (5.1) gives

$$(5.2) \qquad Ec = \left(\max(\phi_{n+}) - \max(\phi_{n-})\right) \qquad\qquad n = 1,2,\ldots,N$$

If we define J to be the index to the particular UI within a consecutive UI record where the maximum positive deviation exists and K to be index to the particular UI where the maximum negative deviation exists, we can form an expression for the maximum eye closure. Substituting J and K into equation (5.2) for n+ and n- gives

$$(5.3) \qquad Ec = (\phi_{n+J}) - (\phi_{n+K})$$

By using the definition of phase jitter in equation (2.1), we can find the phase jitter directly from the record of measurements given by:

$$(5.4) \qquad E_c = (t_{n+J} - t_{n+K}) - (J - K) * T \qquad\qquad J = \text{max+ and K = max-, } n = 1,2,\ldots,N$$

assuming we know the particular indexes of the min and max, J and K. For any 250 UI ensemble, the measured eye closure is given by equation (5.4) where J is the particular maximum and K is the particular minimum. N is the number of unit interval over which the measurement is taken. This is a peak (max) to peak (min) measurement.

5.2 Recovered Clock

So far, we have used some ideal T for the period of the ideal clock. This would measure all components of phase jitter relative to this ideal recovered clock, with all frequencies of phase jitter weighted equally. In practice, there is no ideal clock and the period used is recovered from the data itself, generally over a wider range than the measurement. In the PCI Express specification, the recovered clock is defined as being recovered over a 3500 UI period. This leads to the expression for T of

$$(5.5) \qquad T = \frac{(t_{n+1750} - t_{n-1750})}{3500}.$$

This averaging "bends" the ideal recovered clock to follow the data phase for low frequency jitter, thereby rejecting low frequency phase jitter while measuring high frequency jitter. This concept is illustrated in Figure 25 for a T averaged over a few cycles.

It is important to note that equation (5.5) is not the clock recovery algorithm specified by the PCI Express specification. The PCI Express specification recommends a minimization merit function as supported by a white paper on the subject [7]. However, for the purpose of deriving the frequency response, the average approach will be approximately equal to the fit function since our derivation is using ideal measurements and ideal periods.
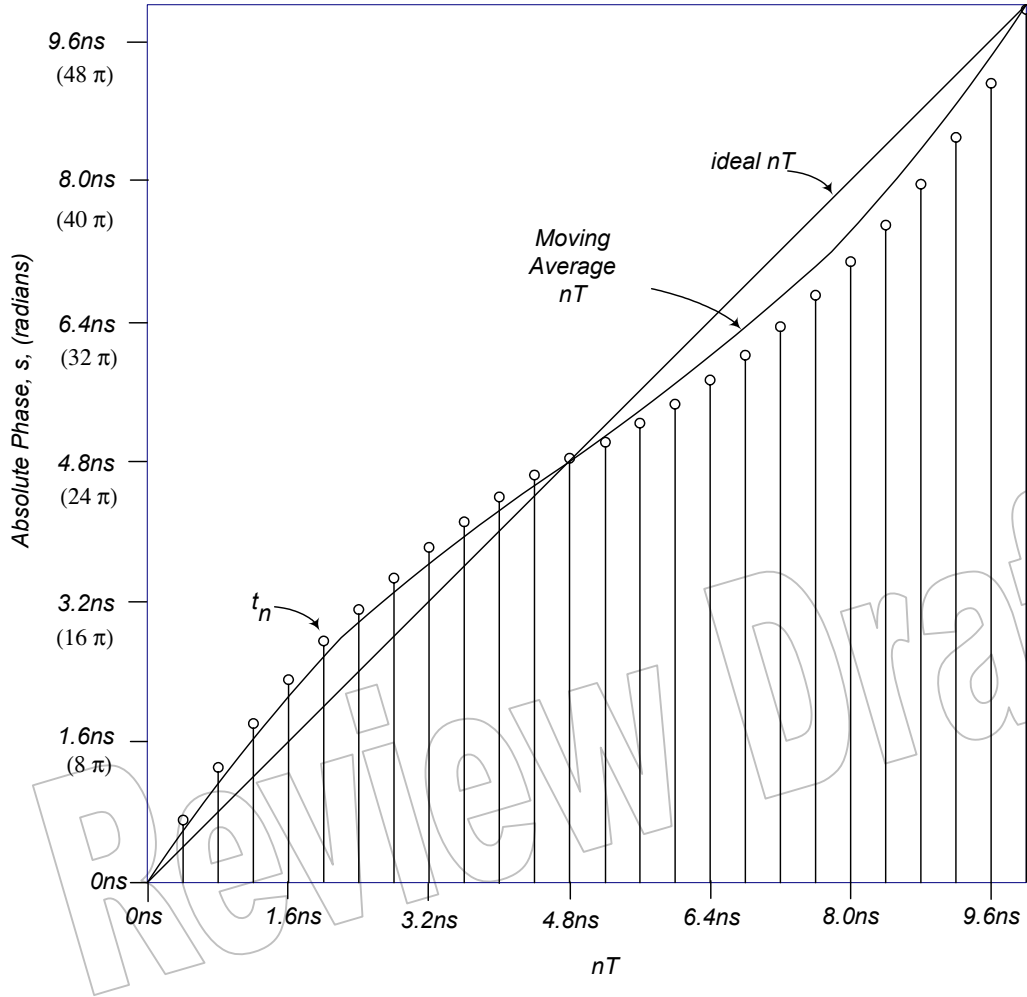
Figure 25: Recovered Clock

Substitution of (5.5) into (5.4) gives

$$(5.6) \qquad Ec = |\, t_{n-1750+J} - t_{n-1750+K} - \frac{J-K}{3500} * (t_n - t_{n-3500})\,|\,, \qquad \text{J= max, K = min.}$$

It can be shown that by the repeated application of the delay model (see appendix C), the transfer function of this equation is given by (5.7).

$$(5.7) \qquad H = [\exp(-s * (t_{1750} + j * T)) - \exp(-s * (t_{1750} + k * T))] - \frac{j-k}{3500}[1 - \exp(-s * t_{3500})].$$

The transfer function of this measurement is shown in Figure 26.

34
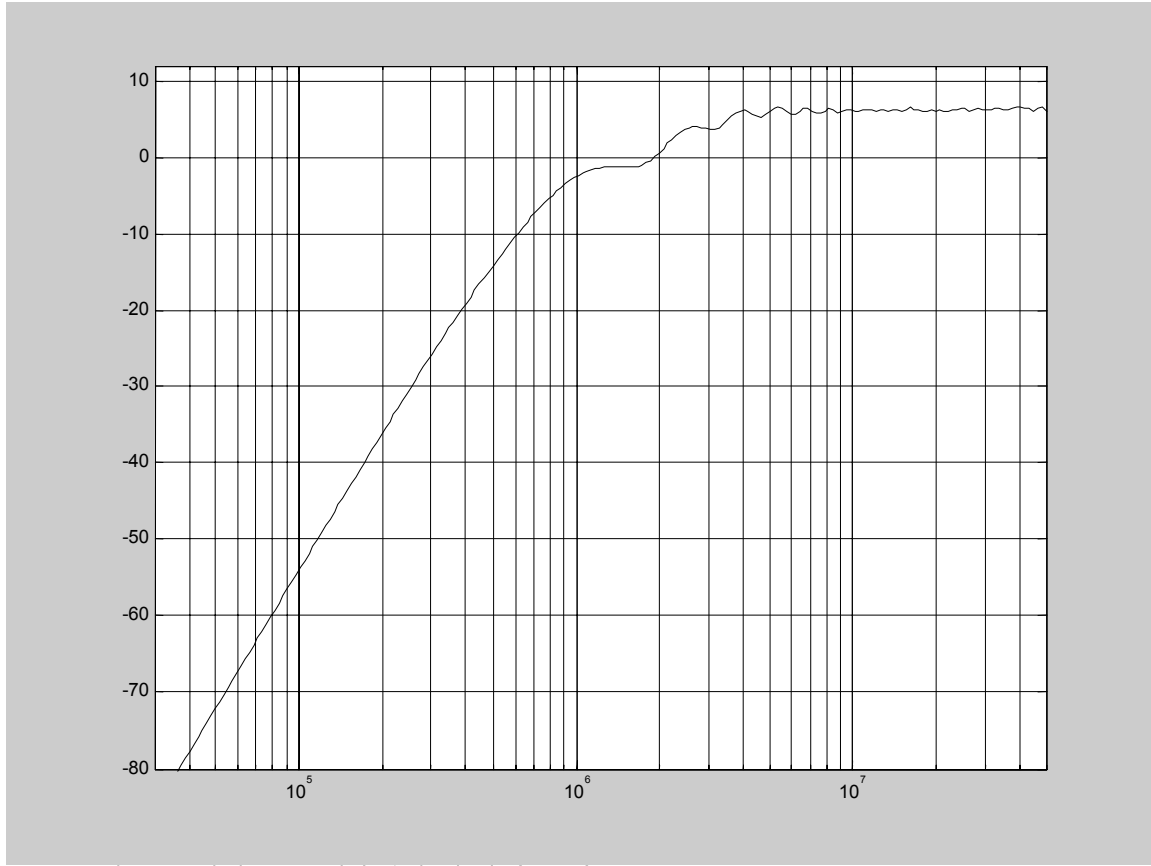
Figure 26: Frequency Response of 250 UI Measurement Method

The transfer function is defined as the ratio of the eye closure Ec(s) to the input phase jitter $\phi(s)$ as given in (5.8). This provides the frequency response of the 250 UI measurement method showing the rejection of low frequency jitter and the measurement of the higher frequency jitter.

$$(5.8) \qquad\qquad H(s) \approx \frac{Ec(s)}{\phi(s)}$$

It can be seen in Figure 27 that the definition of the recovered clock can have a large impact on the results.
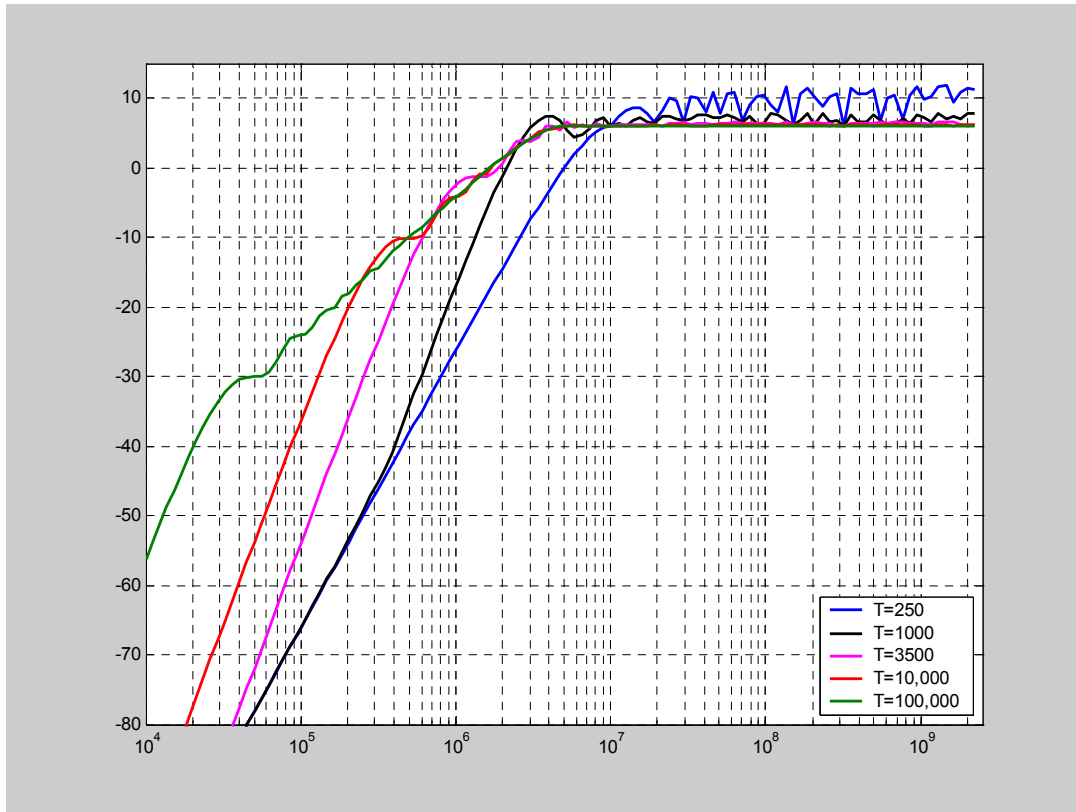
Figure 27: Variation in Clock Recovery Response with Different Average Periods

5.3 250/3500 UI Approximation

Phase jitter is measured with a recovered clock as defined by the 250/3500 UI measurement method of the PCI Express specification. The recovered clock has a frequency response shown by Figure 26.

The response of the 250/3500 measurement method can be closely approximated by a 3-pole high pass with a f_3db frequency of 1 MHz and a gain of 2.

This is shown in Figure 28.
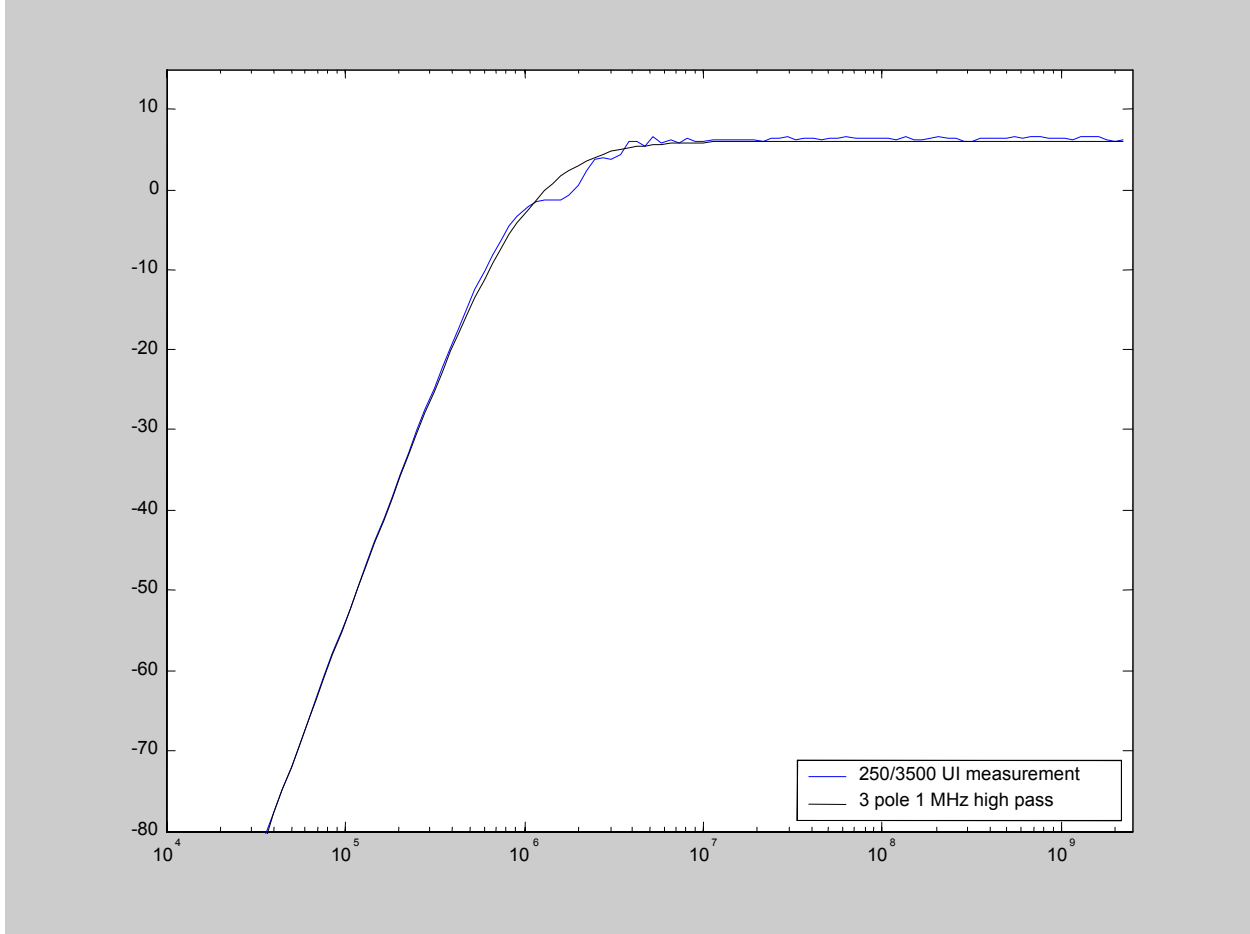
Figure 28: Approximation of 250/3500 UI Window

The magnitude of this transfer function is given by

$$(5.9) \qquad H_{250}(s) = \left| 2 * \left[\frac{s}{s + 2 * pi * 10^6} \right]^3 \right|$$

5.4 Phase of the 250/3500 UI

The physical behavior of single pole high pass filter, such as an RC network, is to create a phase shift of 90 degrees in the stop band and 0 degrees in the pass band. This is due to the physics of the components that comprise the filter.

There is no physics in the 250/3500 UI measurement method that creates any type of phase shift of the frequency components. The phase of the measurement transfer function must be 0 for all frequency components.

In order to apply a 0 phase transfer function, the third order approximation must be modified to zero the phase. This is accomplished by taking only the magnitude of the transfer function into the real part, as shown in Equation (5.9).

6 Discussion

6.1 Comparison of Measurement and Rx Eye Closures

It is relatively straight forward to use the previous results to show that the eye closure at the receiver of the digital CDR can exceed the eye closure as measured by the 250/3500 UI measurement, depending on the BW assumptions of H_1 and H_2.

Figure 29 shows the case with the H_2 f_3dB at 2 MHz and H_1 f_3dB at 22 MHz. In this figure, the transfer function of the measurement method H1*H250 is approximately the same the Rx eye closure.

Figure 30 shows that when H_1 f_3dB < H_2 f_3dB, this is not the case; that the measurement will return a lower value than the actual eye-closure.

Figure 31 shows that with H_1 f_3dB at 7 MHz and H_2 f_3dB at 22 MHz, the eye closure is indeterminate and depends on the noise content of the 100 MHz reference clock.

To have the 250/3500 UI method measured eye-closure as the upper upper-bound, the following inequality must be satisfied for all values of s.

$$(6.1) \qquad [H_1 - H_2] * H_3 \; < \; |H_1 * H_{250}|$$

where we have explicitly shown the magnitude of the difference function is to be taken. The ratio of H_3 to H_{250} introduces a frequency dependent scaling factor.

$$(6.2) \qquad |H_1 - H_2| \; < \; \left| H_1 * \frac{H_{250}}{H_3} \right|$$

For this inequality to hold requires that H_2 never exceeds H_1, adjusted by a scaling factor. If H_2 exceeds H_1, then the measured eye closure (RHS of equation (6.2) will be less than the Rx eye closure (LHS of equation (6.2). This proves that the eye closure given by the measurement method can be exceeded by the eye closure at the receiver when the receiver transfer function, H_2, exceeds the measured transmitter function, H_1.

For reference, both the H250 transfer function and the H1 – H2 transfer functions are plotted in Figure 32.
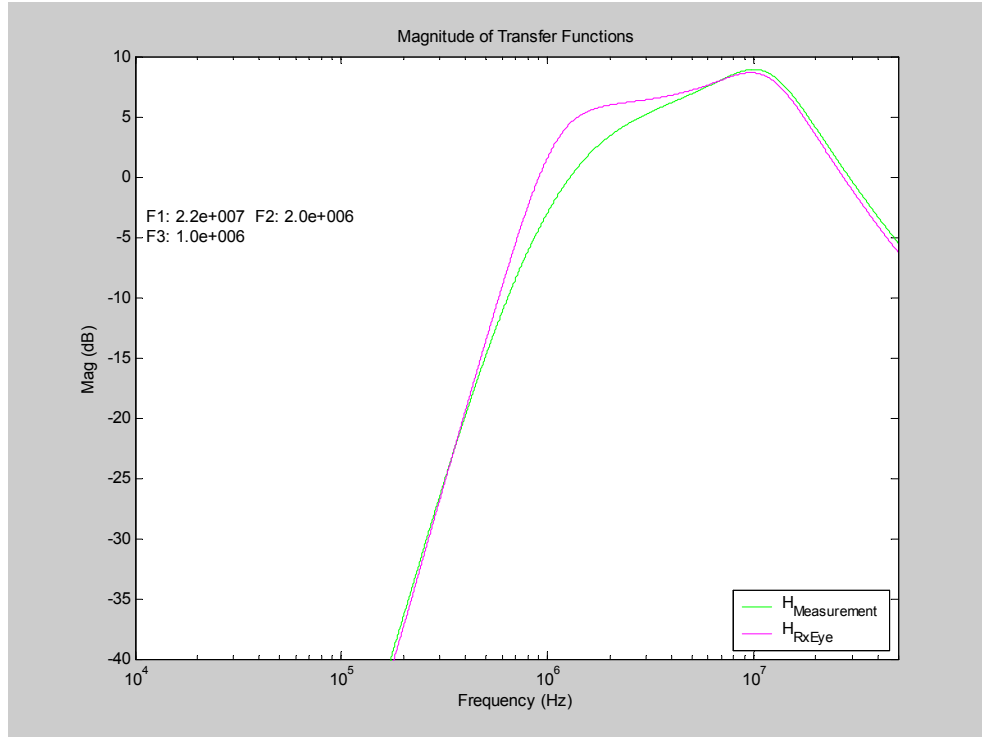
Figure 29: Comparison of Rx Eye Closure [H1-H2]*H3 and the Measurement Eye Closure H1 * H250, H1=22 MHz and H2 = 2 MHz
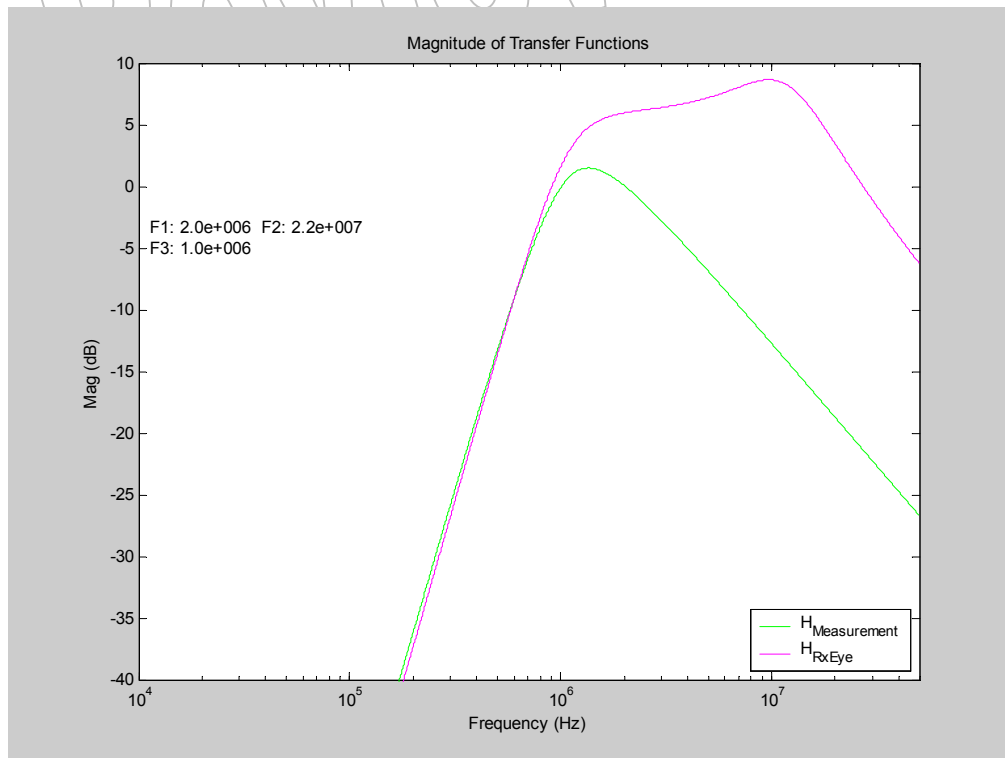
Figure 30: Comparison of the Measurement Eye Closure and the Rx Eye Closure, H1 f_3dB = 2 MHz and H2 f_3dB = 22 MHz
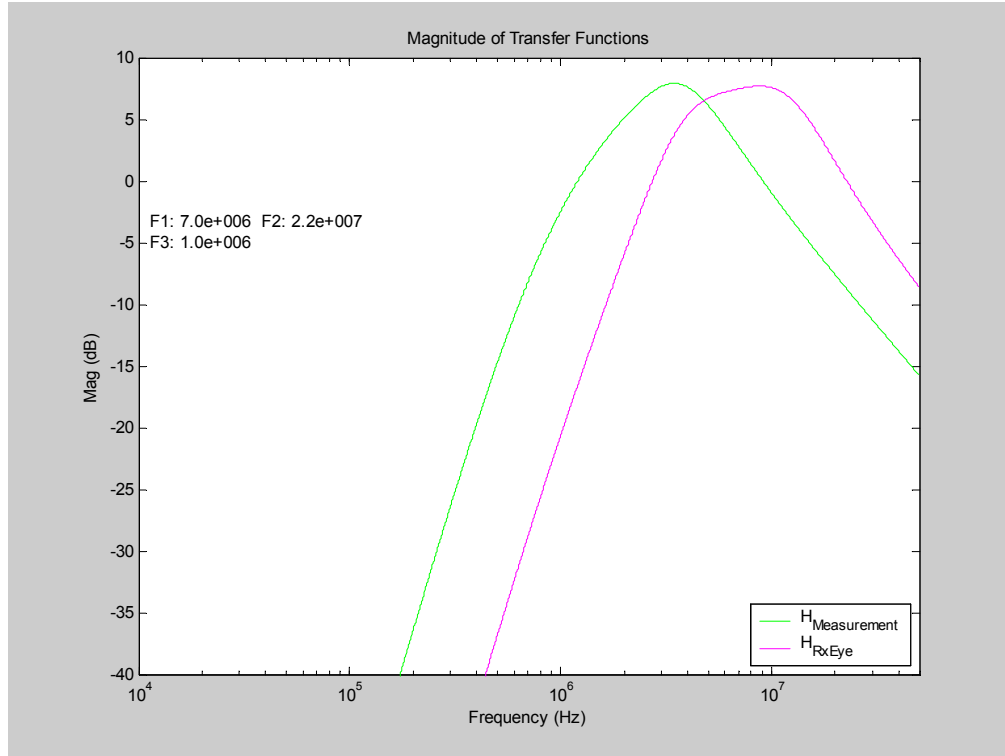
Figure 31: Comparison of the Measurement Eye Closure and the Rx Eye Closure, H1 f_3dB = 7 MHz and H2 f_3dB = 22 MHz
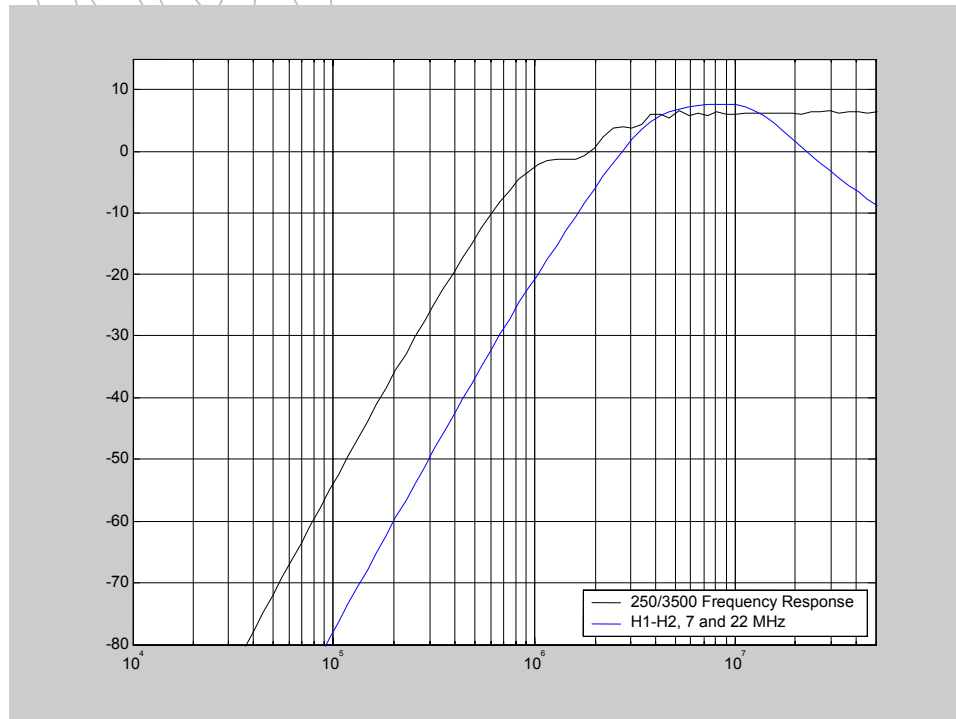
Figure 32: Comparison of H250 and Ht, H1 f_3dB = 7 MHz, H2 f_3dB = 22 MHz

6.2 Design Implications

We have shown there is an impact of the reference clock on the system jitter budget. There is an engineering change request (ECR) that has been submitted for the 1.0a specification to allocate a budget to the reference clock. This ECR is expected to be included in the PCI Express revision 1.1 specification. Refer to the PCI SIG website for the latest information and the status of the 1.1 specification.

Peaking can be a major contributor to eye closure and must be controlled. PLL Designers should ensure the peaking of the PLL does not exceed 3 dB, a generally accepted good design practice. Future generations of PCI Express are likely to also limit PLL bandwidths in order to minimize the impact of the reference clock jitter on the total system budget.

One of the contributors to reference clock jitter is power supply noise at the reference clock. System board designers must ensure that adequate decoupling and filtering is provided to the reference clock. Failure to follow the clock generator's filter recommendations can lead to excessive eye closure and bit errors. A reference clock specification in the form of an ECR is being proposed for the CEM specification to limit the amount of reference clock jitter. Refer to the PCI SIG website for the latest information and the status of the CEM specification.

The transmitter design must limit the self-induced jitter. This means following good design practices for power supply noise rejection, adequate power deliver, etc.

7 Summary

Jitter can be characterized in a number of ways; phase jitter, period jitter, or cycle-to-cycle jitter. Period jitter is roughly equivalent to the first derivative of phase jitter, and cycle-to-cycle jitter is roughly equivalent to the second derivative of phase jitter. Given a continuous record in time of jitter in any one of the forms, the other two can be derived. However, if only a peak-to-peak value for one form of jitter is known, determination of the peak-to-peak values of the other forms of jitter is not possible.

Transfer functions are a traditional way of accurately predicting a system behavior and have been developed to model the performance of PCI Express devices when including the effects of reference clock noise. This paper provides the information and tools to properly determine the performance of the system given the expected reference clock noise and the characteristics of the parts of the system (PLL bandwidths). This supports the requirement for the reference clock specification and for extended compliance testing that includes the behavior of phase jitter at the system level.

Jitter considerations for serial communication differ from the jitter considerations for parallel communication buses. In parallel communications architectures, phase jitter is not a key metric, cycle to cycle and period jitters more important. On the other hand, for serial communications phase jitter is the number one metric. The recovered clock used to calculate the phase jitter is also important, since it changes the magnitude of the phase jitter measurement.

PCI Express differs from classical communications architectures in that it is not an entirely clock recovered architecture. Extensive use of phase interpolators and SSC infer a hybrid approach, where we have shown it to be a pseudo-clock forwarded architecture that involves two PLLs communicating with each other. Care is required by the system and component designers to ensure reference clock jitter is properly understood, measured, and managed.

8 References

1. PCI-SIG, *PCI Express Base Specification, Rev. 1.0a*, (www.pcisig.com).

2. PCI-SIG, *PCI Express Card Electromechanical Specification, Rev. 1.0a*, (www.pcisig.com).

3. William J. Dally and John W. Poulton, *Digital Systems Engineering,* (Cambridge University Press, 1998).

4. Gordon E. Carlson, *Signals and Linear System Analysis*, second edition, (John Wiley and Sons, 1998).

5. Ronald E. Mickens, *Difference Equations, Theory and Applications,* second edition, (Van Nostrand Reinhold, 1990).

6. Roland E. Best, *Phase-Locked Loops, Design, Simulation, and Applications*, fourth edition, (McGraw-Hill, 1999).

7. Per (Pelle) Fornberg, Dan Froelich, Andy Volk, *Transmitter Clock Recovery and Jitter Analysis Methodology For PCI Express Signaling,* Revision 0.8, Intel Corporation

Appendix A

```
%*****************************************************************************
%*
%*      This is a script to process edge crossings of a clock
%*
%*      The data samples are the period measurements as defined in the whitepaper.
%*      Copyright (c) 2003 Intel Corp.
%*
%*      Send comments to andy.martwick@intel.com and duane.quiet@intel.com
%*      This program has been developed by Intel Corporation.
%*
%*      Intel specifically disclaims all warranties, express or
%*      implied, and all liability, including consequential and other
%*      indirect damages, for the use of this code, including liability
%*      for infringement of any proprietary rights, and including the
%*      warranties of merchantability and fitness for a particular
%*      purpose.  Intel does not assume any responsibility for any
%*      errors which may appear in this code nor any responsibility to
%*      update it.
%*****************************************************************************
%
%------------------------------------------------------------------------------%
%               ----- Data Format -----
%
%       The program expects a (1 x N) matrix of 100 MHz consecutive clock
%       periods from a deep memory scope at a high sampling rate.  N needs
%       to be > 50K points, and ~150K points is desired.
%
%       "Dataset1" is the (1 x N) matrix used in the program and must be
%       loaded prior to running the script.
%
%       Load the ASCII period data with the following:
%           load -ascii 'filename.mat'
%           Dataset1 = filename;
%
%       Some jitter analysis programs attach the time index to the period
%       data resulting in a (2 x N) ASCII matrix.  If needed, remove the
%       first column time index with the following:
%           Dataset1 = Dataset1(:,2);
```

```
%
%--------------------------------------------------------------------------------%
%            ----- Setup -----
        close all;            % close all existing figures
        clear L M N Phi t1 pj Ht H1 H2 H3 timeaxis x;



        YES=1;                % an orthogonal basis set
        NO=0;
        colordef black        % set the graphing backgrounds to black


        SP=10e-9;             % sample period for the 100 MHz reference clock
        N=150000;             % number of samples. Need >50k to get a good spectrum.


        Res = round(100e6/N);   % resolution of single freq point


        f=1:Res:Res*N/2;      % set the frequency sweep from 1 Hz to 50MHz (N/2 points)
        s=f*2*pi*j;           % set the xfer function to complex radians
```

%--%

%   Define transfer functions:

%   H1 is the Tx PLL, and is where the delay can be added

%   H2 is the Rx PLL

%   H3 is Rx Phase interpolator

%   see white paper for natural frequency documentation

```
        f1 = 22e6;          % define 3dB corner frequency for H1
        zeta1 = 0.54;       % define peaking for H1


        f2 = 7e6;           % define 3dB corner frequency for H2
        zeta2 = 0.54;       % define peaking for H2


        f3 = 1.0e6;         % define 3dB corner frequency for H3



        w1=2*pi*f1/((1+2*zeta1^2+((1+2*zeta1^2)^2+1)^.5)^.5);          % w1 = natural frequency
        H1=((s.*2*(zeta1)*w1+w1.^2)./(s.^2+2*(zeta1)*w1.*s+w1.^2));    % H1 transfer function
        H_1=20*log10(abs(H1));                                        % printable version

        w2=2*pi*f2/((1+2*zeta2^2+((1+2*zeta2^2)^2+1)^.5)^.5);          % w2 = natural frequency
        H2=((s.*2*(zeta2)*w2+w2.^2)./(s.^2+2*(zeta2)*w2.*s+w2.^2));    % H2 transfer function
        H_2=20*log10(abs(H2));                                        % printable version

        w3=2*pi*f3;                 % w3 = 3dB point for a single pole high pass function.
        H3=(s./(s+w3));             % the H3 xfer function is a high pass
        H_3=20*log10(abs(H3));      % a printable version on a log scale
```

%--%

%   Add a phase delay to one of the xfer functions. It should be to the

%   xfer function with a phase change in the range of the data, ie, the

%   f_3dB needs to be in the 5-50 MHz range otherwise it has no effect.

%   Only use the delay or if the delay is not known use a 2x worst case, set delay to 0

```
        delay=0e-12;                % ideal transfer has default delay = 0
        H1 = H1.*((exp(-delay*s)));  % delay H1 with respect to H2
```

%--%

%   Build the final xfer function Ht -- NOTE 2X MULTIPLIER TO ACCOUNT FOR UNKNOWN PHASE DELAY

```
        Ht =2*(H1-H2).*H3;              % Final transfer based on the difference functions
                                        % note 2X multiplier -- see section 6.5 of whitepaper


        %   Ht=(H1.*(1-H2));            % use if PLL based clock recovery (not worst case)
        H_t=20*log10(abs(Ht));          % create H_t for plotting
```

%---%
%   plot the magnitude of the transfer functions
%

```
        figure (1);
        semilogx(f,(H_1(1:length(H_1))),'r')    % magnitude of H1 (Tx PLL)
        hold on;
        semilogx(f,(H_2(1:length(H_2))),'b')    % magnitude of H2 (Rx PLL)
        semilogx(f,(H_3(1:length(H_3))),'g')    % magnitude of H2 (Rx phase interpolator)
        semilogx(f,(H_t(1:length(H_t))),'y')    % magnitude of combined difference function
        axis([1e4 50e6 -40 10])
        legend('H1', 'H2', 'H3', '2*((H1-H2)*H3)', 4);
        title('Magnitude of Transfer Functions');
        xlabel ('Frequency (Hz)');
        ylabel ('Mag (dB)');
        S=sprintf('  F1: %1.1e  F2: %1.1e \n  F3: %1.1e\n' ,f1,f2,f3)
        text(1e4,-5,S)
```

%---%
%   plot the phase of the transfer functions

```
        figure (2);
        semilogx (f,(atan(imag(H1)./real(H1)))/(2*pi)*360, 'r')    % phase of H1
        hold on;
        semilogx (f,(atan(imag(H2)./real(H2)))/(2*pi)*360, 'b')    % phase of H2
        semilogx (f,(atan(imag(H3)./real(H3)))/(2*pi)*360, 'g')    % phase of H3
        semilogx (f,(atan(imag(Ht)./real(Ht)))/(2*pi)*360, 'y')
        axis ([1e5 50e6 -100 100]);
        legend ('H1', 'H2', 'H3', '2*((H1-H2)*H3)' , 4)
        title('Phase of Transfer Functions');
        ylabel ('Phase (Degree)');
        xlabel ('Freq (Hz)');
```

%---%
%   process the data to get the phase jitter

```
pj = Dataset1(1:N);     % read in data

pj = pj-mean(pj);       % remove the DC offset to get the period jitter (Phi prime)

Phi = cumsum(pj);       % integrate to get the phase jitter

Phi = Phi - mean(Phi);  % center at 0
```

%--%
%   plot the phase jitter

```
figure (3)

timeaxis=(0:SP:(N-1)*SP);  % scale the time axis

plot(timeaxis,Phi,'b')

title('Phase Jitter vs Time');

xlabel ('time (S)');

ylabel ('time (S)');
```

%--%
%   take complex DFT of the phase jitter

```
K=fft(Phi,N);   % take the complex DFT of the phase noise from 1:N

L=K/N;          % scale (both + and - frequencies)
```

%--%
%   apply transfer function Ht to data set L and store in M

```
M=zeros(N,1);                   % set up array of zeros

Hfinal = Ht;


for i=1:1:N/2-1

        M(i) = L(i) .* Hfinal(i);   % apply to the positive frequencies

end


for i=N/2:1:N-1

        M(i)=L(i) .* Hfinal(N-i);   % handle the negative frequencies

end
```

%--%
%   plot the noise spectrum before and after the 2*Ht transform is applied

```
figure(4);
loglog(f,(2*abs(L(1:N/2))),'c') % plot single side(mult by 2 so magnitude is right)
hold on
loglog(f,2*abs(M(1:N/2)),'m')   % show the spectrum after the filter is applied
axis([10e3 1e8 1e-14 2e-8]);
legend('Spectrum', 'Spectrum after transfer function');
title('Phase Jitter Frequency Content')
xlabel ('Frequency (Hz)');
ylabel ('Magnitude (S)');
text(1e4,-5,S)
```

%--%
%   calculate IFFT to get eye closure magnitude

```
x=ifft(M,N);          % take the inverse transform
x=(x*N);              % fix scaling
```

%--%
%   plot eye closure magnitude

```
figure(5);
timeaxis=(0:SP:(N-1)*SP);       % scale the time axis
plot(timeaxis,real(x),'y');   % show the time magnitude of the closure around 0
xlabel('Time (S)');
ylabel('Eye closure (S)');
```

%--%
%   return maximum peak jitter and projected eye closure due to 2*(H1-H2)*H3 for each

```
peak_close = max(abs(real(x))) % gives maximum positive or negative phase jitter

eye_closure = peak_close
```

return
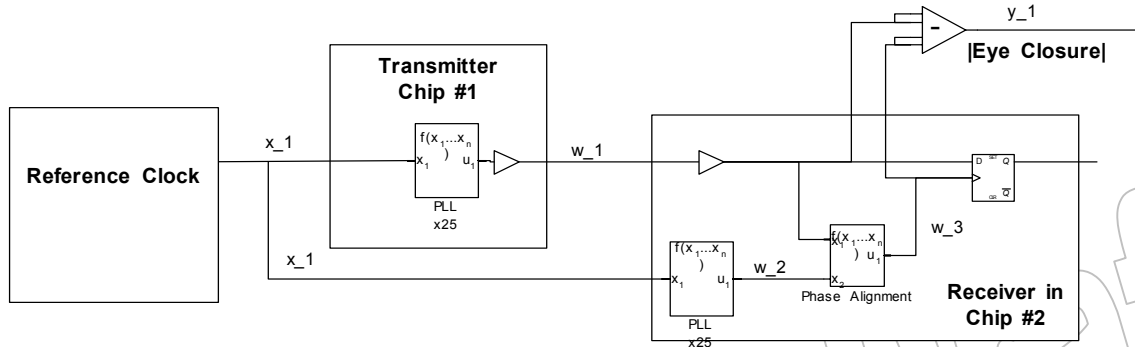
Appendix B: Derivation of the PI High Pass

We derive this result that allows the PI to become a high pass and move to the R.H.S. of the difference function.

System Model

The full physical model of the system is:

The input signal is x_1. We will assign the transmitter chip the transfer function H1, the receiver chip's PLL will have the transfer function H2, and the phase interpolator in the receiver will have the low pass transfer function H3.

The Phase Interpolator has two inputs, $x\_1 * H1$ and $x\_1 * H2$. These are labeled on the previous figure as w_1 and w_2, respectively. These intermediate signals are defined by

(1) $w\_1 = x\_1 * H1$

(2) $w\_2 = x\_1 * H2$

The phase interpolator looks at the magnitude of the difference between these two inputs and adjusts w_2 by the difference ($w\_1 – w\_2$) to produce w_3. The eye closure is then given by the difference of w_3 and w_1. This is labeled y_1 in the diagram.

If $w\_1 > w\_2$, then the difference between w_1 and w_2, a positive quantity, will be added to w_2 in an attempt to align w_2 to w_1. If $w\_1 < w\_2$, then the difference between w_1 and w_2, a negative quantity, will be added to w_2 in order to align w_2 to w_1. The difference between w_1 and w_2 is always added to w_2.

This is shown in the following diagram:
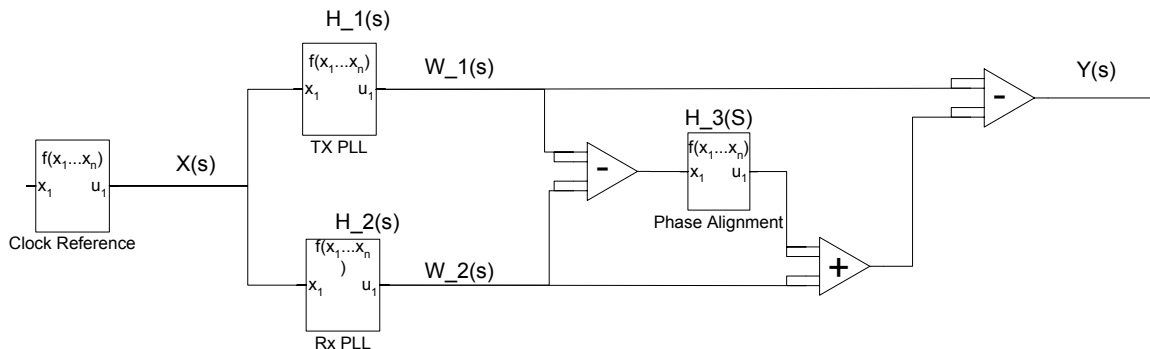
Based on this model and substituting equations (1) and (2), the total system equation is:

(3) $Y = X\{H1-[(H1-H2)*H3+H2]$

This is arrived at by first factoring out the input signal, X. The rest reads that the difference is H1 minus the response of the phase interpolator added to H2, in order to align to H1, but limited by the low pass nature of the PI, H3.

To simplify this equation we want to represent H3 as a high pass instead of a low pass. Define H3' as the high pass function of the phase interpolator, then equation (4) gives the relationship between H3 and H3'.
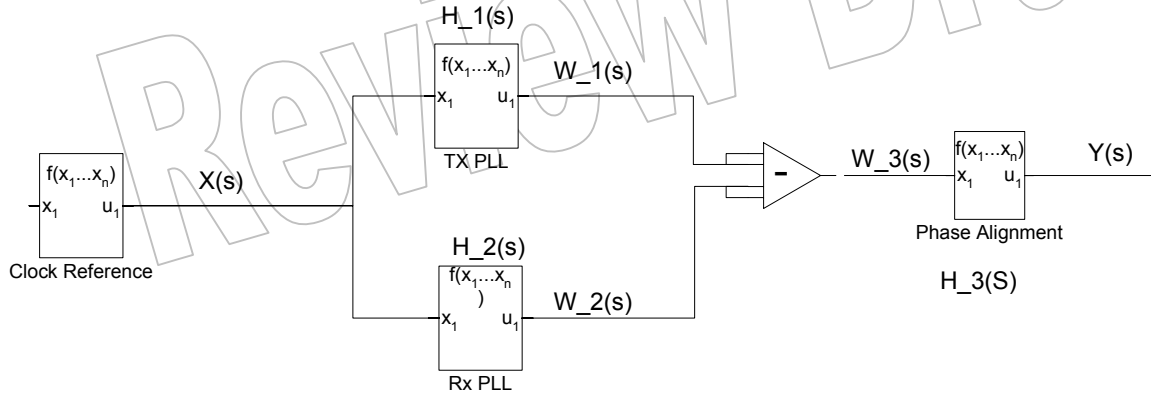
(4)   H3' = 1-H3

Dividing (3) by X and substituting (4) into (3) gives

(5) $\dfrac{Y}{X} = H1-[(H1-H2)*(1-H3')+H2]$

letting Ht = Y/X and simplifying gives the result:
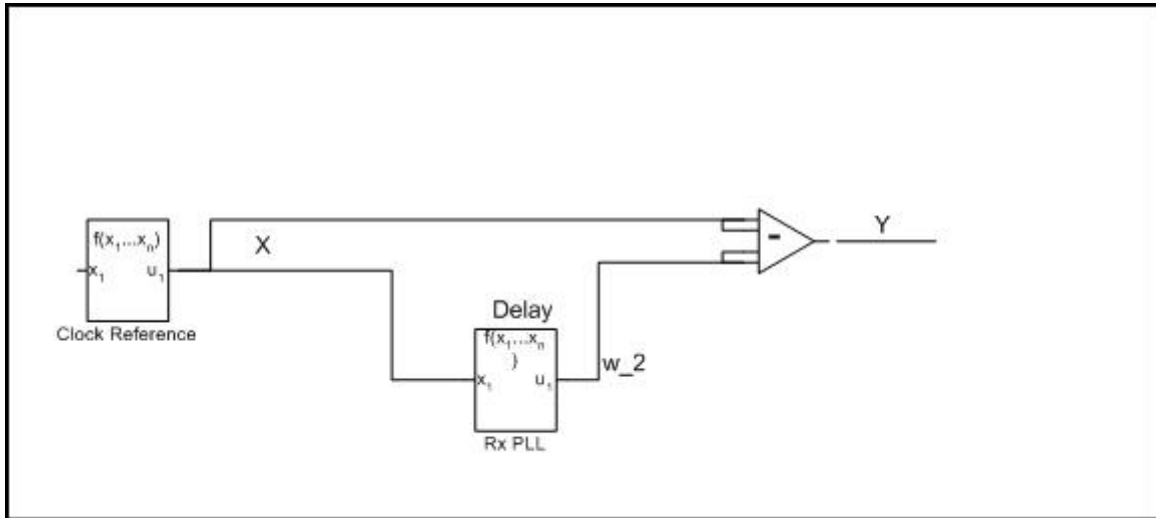
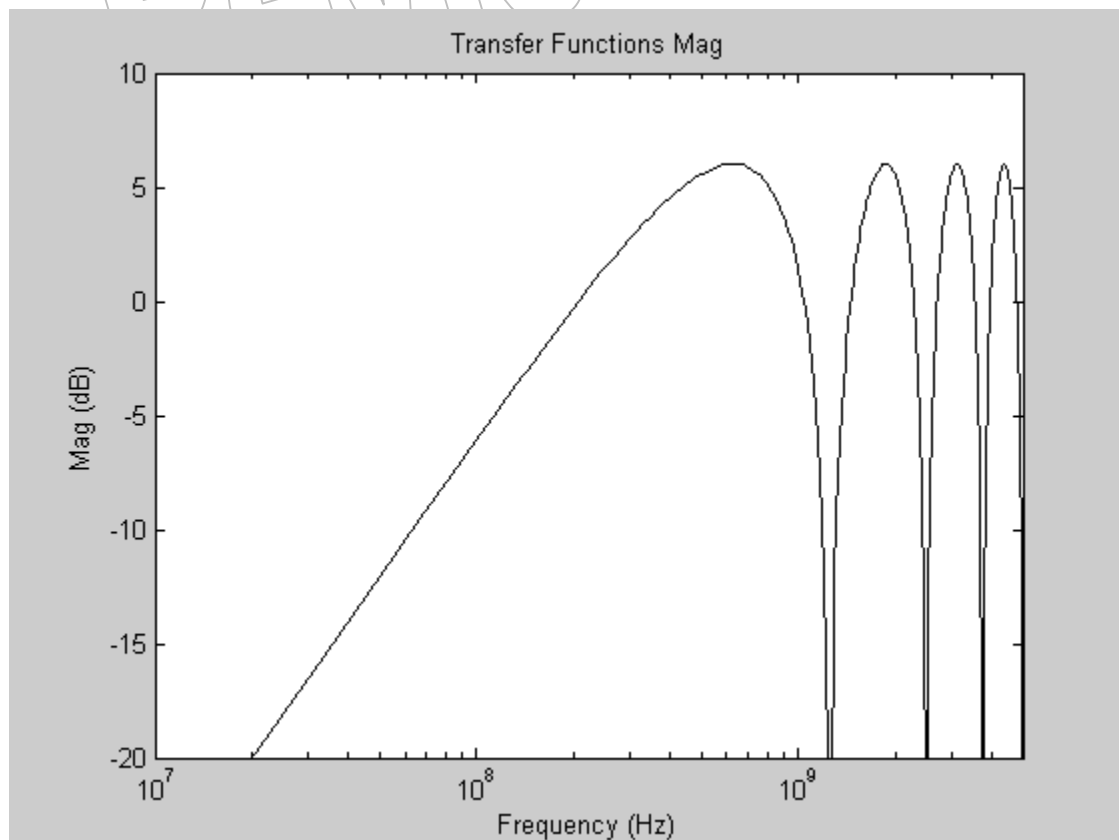(6) $Ht = (H1-H2)*H3'$

and is shown in the following diagram:

where H3' is the high pass transfer function of the low pass phase interpolator.

Appendix C: Delay Model

Consider an input signal X subtracted with a delayed version of itself.

This is modeled by H(s) = (1 – exp(-s * t_delay)). An example delay of an 800 ps delay is: