

# USB 3.0 CDR Model White Paper

## Revision 0.5

January 15, 2009

DRAFT

### **INTELLECTUAL PROPERTY DISCLAIMER**

THIS WHITE PAPER IS PROVIDED TO YOU "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE. THE AUTHORS OF THIS WHITE PAPER DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY PROPRIETARY RIGHTS, RELATING TO USE OR IMPLEMENTATION OF INFORMATION IN THIS WHITE PAPER. THE PROVISION OF THIS WHITE PAPER TO YOU DOES NOT PROVIDE YOU WITH ANY LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS.

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

Copyright © 2009, Hewlett-Packard Company, Intel Corporation, Microsoft Corporation, NEC Corporation, ST-NXP Wireless, and Texas Instruments.

All rights reserved.

## About This Document

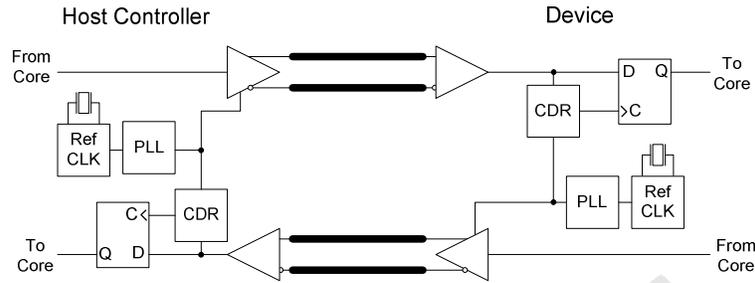
This paper describes the origin of the Jitter Transfer Function (JTF) and the slew rate limit for SuperSpeed USB. Traditionally serial architectures based the JTF on linear PLL models. In SuperSpeed USB the JTF is based on a digital Clock and Data Recovery circuit (CDR).

Spread spectrum clocks (SSC) present special challenges to the clock and data recovery circuit (CDR). This is due to the large, low frequency difference between the local clock and the incoming data. New to the USB 3.0 specification is the phase jitter slew rate requirement that helps bound the impact of the SSC on the CDR. An abstract model of the CDR is presented and the impact of SSC to this circuit is shown.

The overlap between the JTF and the phase jitter slew rate limit is shown, and a Matlab® script for performing the phase jitter slew rate measurement is provided.

DRAFT

# SuperSpeed USB Clocking and Jitter



**Figure 1: SuperSpeed USB clocking and jitter architecture**

An overview of SuperSpeed USB clocking is shown in Figure 1. The host and device each have a separate reference clock that includes spread spectrum. Spread spectrum is described in detail later. The reference clock frequency is multiplied up to the bit rate by the PLL and used to transmit the data at the bit rate. It is also sent to the clock recovery circuit and used to generate the recovered clock. The recovered clock is then used to sample the data.

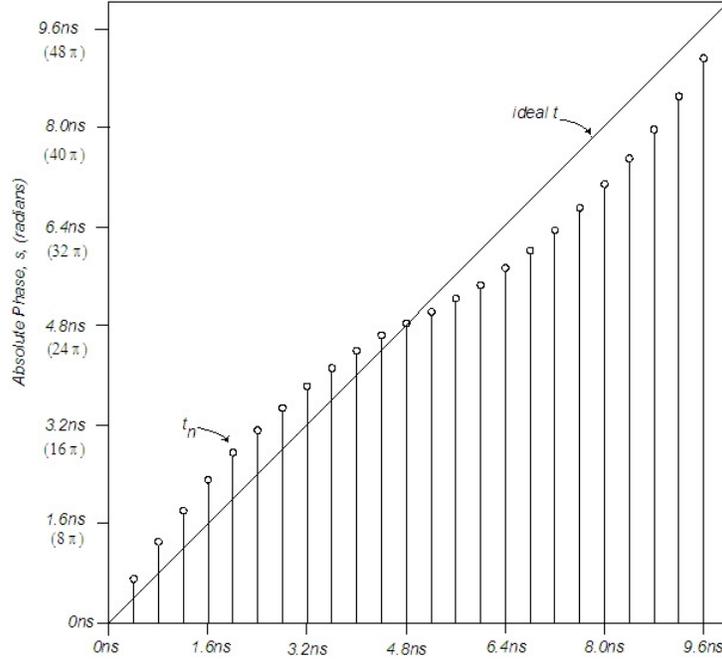
*Jitter is the timing error between the actual sampling edge and the ideal sampling point for the data.*

Jitter is always a relative measurement between the sampling clock and the ideal sampling point for the data. In an oscilloscope the sampling clock is a high precision time base. In a serial link, the sampling clock is the output of the clock recovery circuit. The job of the clock recovery circuit is to line up the sampling clock to the center of the data eyes.

The clock recovery circuit has a transfer function from the input data edges to the recovered clock. The function is called the clock recovery function and it sets the minimum behavior required of the clock recovery circuit. This function is designated as  $H_{CDR}$ . It is this function  $H_{CDR}$  that is used to determine the Jitter Transfer Function (JTF) of the serial specification.

$H_{CDR}$  is critically important to the specification of the serial link. It establishes a minimum behavior of the receiver and the transmitter that is required for interoperability between two components. If the receiver meets or exceeds the clock recovery function,  $H_{CDR}$ , and the transmitter outputs less jitter than the JTF, then the devices will interoperate properly. If the receiver does not meet the clock recovery function  $H_{CDR}$  or the transmitter outputs excessive jitter, the link will have an increased bit error rate or will not work at all.

In generating the recovered clock the clock recovery circuit looks at the data transitions. It is the data transitions that define the phase of the data. The times of the data transitions can be recorded into a continuous record,  $t_n$ , of the times of the edge crossings for each transition bit. For modeling purposes an edge can be interpolated even if a transition did not occur. This is shown in Figure 2.



**Figure 2: Time stamps of the edge crossings**

The input to the clock recovery circuit is the edge crossings,  $t_n$ , while the output is the edge crossings of the recovered clock,  $r_n$ . The clock recovery function,  $H_{CDR}$ , is the transfer function of  $t_n$  to  $r_n$ .

The Z- transform of  $t_n$  is  $T(z)$ , and is just written  $T$ . Likewise, the transform of  $r_n$  is  $R(z)$  and is written as  $R$ . These will be used to model the clock recovery circuit and to derive the JTF. The transfer function,  $H_{CDR}$ , of the clock recovery circuit is then

$$(1) \quad H_{CDR} = \frac{R}{T}$$

or

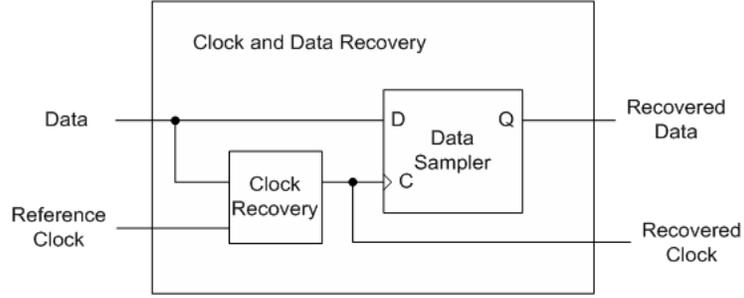
$$(2) \quad R = T \cdot H_{CDR}$$

## SuperSpeed USB Digital CDR

In a communications link, data is sampled into a latch by a clock. The difference in time between the center of the data eye and the sampling clock is the jitter. In a perfect system, jitter is non-existent and the data is sampled at the optimum location in the center of the data bit. In a practical system, the clock and data are misaligned due to various physical processes. The jitter budget allocates the error to each of the components of the transmitter, the media and the receiver. This error component is also called the eye closure. Of course, the total jitter budget must be less than the total unit interval (UI) or a sampling error will occur.

In USB the receiver recovers the clock from the data by digitally adjusting the phase of the local clock to try to match the phase of the incoming data as closely as possible. The

difference of the phase of the recovered clock and the data is a timing error, or jitter. The clock recovery circuit is shown in the following block diagram.



**Figure 3: Receiver clock and data recovery**

Generally we assume that the clock recovery circuit is well behaved and has a frequency domain response,  $H_{CDR}$ . Given the record of edge crossings of the data,  $t_n$ , and the transform of  $t_n$ ,  $T$ , we can calculate the eye closure,  $\Delta_{eye}$ , in the frequency domain as

$$(3) \quad \Delta_{eye} = T - R,$$

applying (2),

$$(4) \quad \Delta_{eye} = (T - T \cdot H_{CDR})$$

factoring,

$$(5) \quad \Delta_{eye} = T(1 - H_{CDR})$$

The jitter transfer function is defined as

$$(6) \quad H_{JTF} = (1 - H_{CDR})$$

This allows us to write express the eye closure in terms of the data edge crossings and the jitter transfer function:

$$(7) \quad \Delta_{eye} = T \cdot H_{JTF}$$

The clock recovery circuit's transfer function is low pass, since the clock recovery circuit needs to look at many samples in order to create the recovered clock. In the simplest case this amounts to an averaging function which is a low pass response. The low pass nature of the clock recovery circuit gives an eye closure that is high pass. High frequency jitter in the data will not be tracked and will directly close the eye, low frequency jitter that is not tracked will close the eye. Low frequency jitter that can be perfectly tracked does not close the eye.

The eye closure caused by a transmitter is measured by applying the JTF to the jitter generated by the transmitter. The frequency domain JTF and the separation between tracked and untracked regions is shown in Figure .

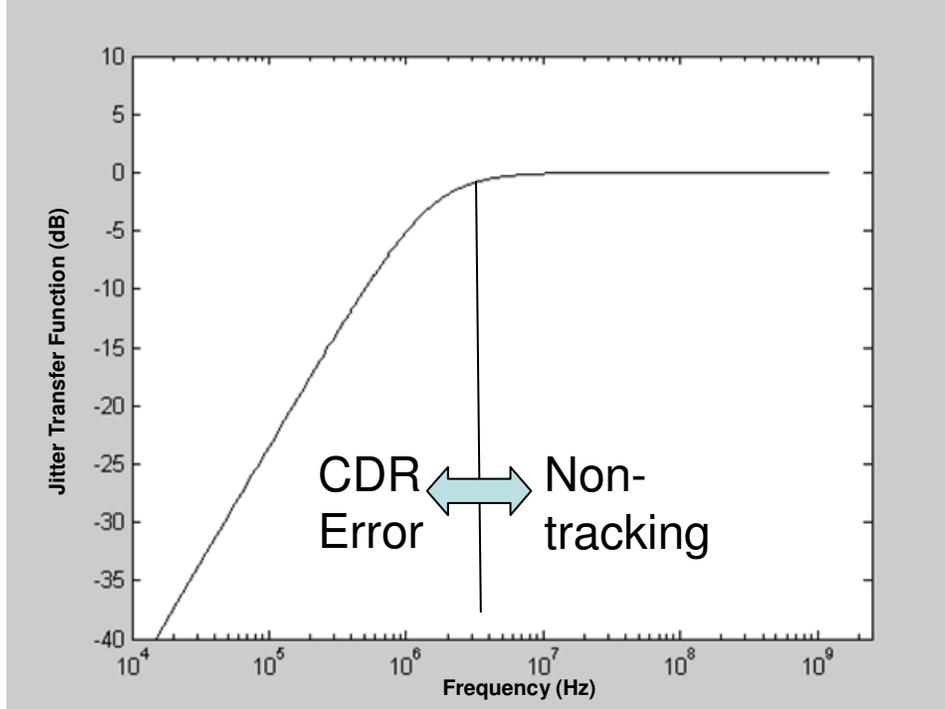


Figure 4: Jitter Transfer Function

The total jitter (eye closure at the sampling latch) is due to the CDR tracking error on the left of the graph **plus** the higher frequency jitter that the CDR is not expected to track.

Clearly, the transfer function of the clock recovery circuit is critical in developing the SuperSpeed USB budgets. In the USB 3.0 specification  $H_{CDR}$  is given as a second order low pass function with a -3dB corner frequency of 10 MHz, this gives the JTF as a second order high pass function with a -3dB corner frequency of approximately 4.9 MHz.

The actual implementation of the clock recovery circuit is digital and non-linear, so a linear model does not adequately describe the behavior of a typical digital implementation. This means that the jitter measured through the JTF could meet the jitter specification but the actual implementations of the clock recovery circuit would fail to track the jitter properly. The phase jitter slew rate specification was added to further limit the transmitter jitter and media jitter in such a way that a digital clock recovery circuit can recover the clock. This is described in detail later in this paper.

## Spread Spectrum Clocks

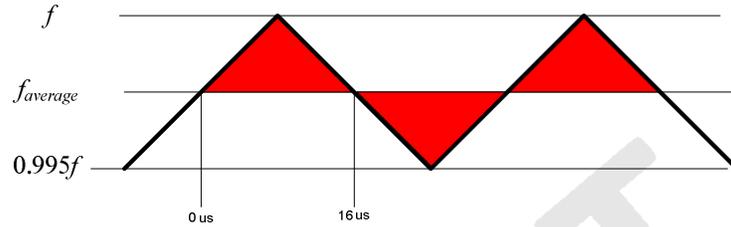
A significant challenge of the clock recovery circuit is to track the jitter at the spread spectrum frequency. This large phase jitter is important to the behavior of H, and therefore the JTF.

Spread spectrum clocks (SSC) are used to lower electromagnetic emissions at any one particular frequency for FCC compliance. The clock frequency is varied between  $f$ , the reference clock frequency, and  $0.995f$  at a rate between 30 KHz and 33 KHz. The average frequency is

$$(8) \quad f_{avg} = \frac{f + 0.995f}{2} = 0.9975f$$

The maximum phase jitter of the SSC can be calculated by integrating the instantaneous frequency and subtracting out the average for all the times the UI is over (or under) the average UI.

This amounts to integrating the shaded area of one of the triangles in Figure with  $f_{average} = 0$ .



**Figure 5: SSC Frequency Modulation**

The area of each shaded triangle in Figure is one-half of the product of the base and the height:

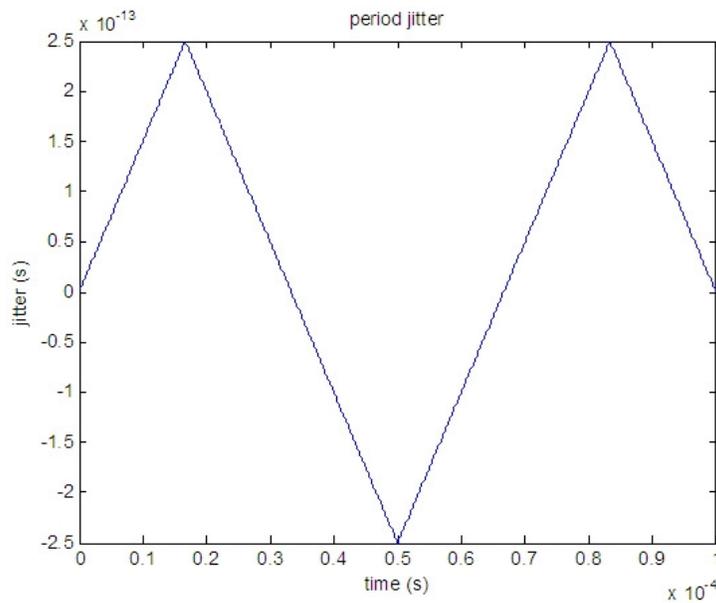
$$(9) \quad \phi_{\max} = 20 \text{ ns} \cdot f ,$$

in units of radians. The phase starts from 0 and increases to 20 ns, then decreases back to 0. This can be approximated by a sinusoid of amplitude 10 ns.

The change in period can be calculated as the inverse of the frequency. This gives

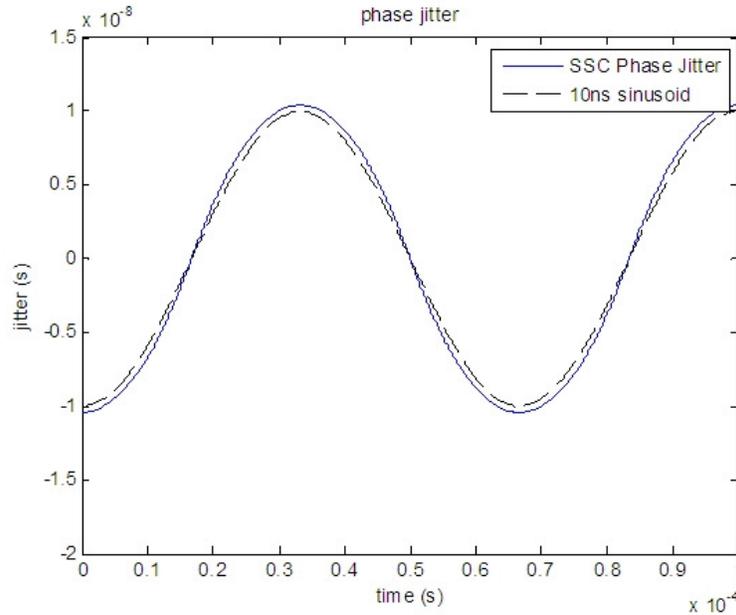
$$(10) \quad \Delta T = \frac{1}{f - 0.005f} \cong 1.005T$$

The maximum period change is therefore  $\Phi'_{\max} \cong 0.005T$ . Graphically, the period jitter is shown in Figure .



**Figure 6: Period Jitter of SSC**

Taking the cumulative sum of the period jitter gives the phase jitter. This is shown in Figure along with the sinusoid approximation.



**Figure 7: Phase Jitter of SSC**

For the remainder we assume that the two sides of the link have independent SSC domains. This doubles the amplitude of the SSC phase jitter that must be tracked to 20 ns. Therefore, in the above example the maximum slew rate due only to the SSC is computed to be:

$$(11) \quad \left. \frac{d}{dt} [(20ns)\sin(2 * pi * 33,000 * t)] \right|_{t=0} = 4.1667 \frac{m}{s}$$

However, additional margin is required to take into consideration the use of other SSC profiles and for the superposition of other jitter components. We generate this margin by doubling the SSC phase jitter amplitude to 40 ns at the maximum SSC frequency of 33 KHz. With these assumptions the maximum slew rate that must be tracked is once again found by taking the derivative of the phase jitter at the zero crossing,

$$(12) \quad \left. \frac{d}{dt} [(40ns)\sin(2 * pi * 33,000 * t)] \right|_{t=0} = 8.3 \frac{m}{s}$$

## Digital Clock Recovery

### Introduction

In this section the model of the digital clock recovery is presented. The conclusion is that for modern digital clock recovery circuits, there is a frequency band where the slew rate of the phase becomes the limiting factor of the clock recovery

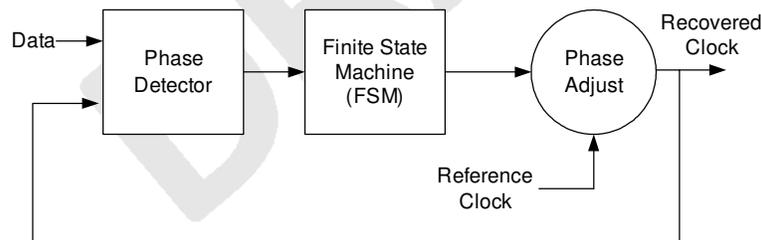
## Clock Recovery

The first serial architectures used an analog phase locked loop to generate the recovered clock. Modern low cost serial architectures are digital based. Two popular digital CDR architectures are phase interpolation and over-sampling types.

The specification of  $H_{CDR}$ , and therefore the JTF, is not intended to define a particular implementation. However, investigation into typical implementations of these circuits showed that the low frequency circuit performance is limited by the phase jitter slew rate. A simple model of the clock recovery circuit is helpful in understanding this requirement.

Conceptually a digital tracking loop is shown in Figure . A binary phase detector (also called a bang-bang phase detector) looks at the current phase of the data compared to the current phase of the recovered clock. The phase detector does not give magnitude information about how far the phase is misaligned, only that the phase is early or late.

This comparison is repeated over many unit intervals (UI) and the early/late results are accumulated. Some type of filter algorithm is applied to the results and an adjustment to the recovered clock is made as necessary. In a phase interpolator this involves moving the position of the reference clock; in an over sampling architecture it involves selecting a different sample for the data. Both architectures can be thought of as moving a fixed step in time. In general, the step size is a fixed quantity.



**Figure 8: Phase Interpolator Block Diagram**

For a periodic function, the frequency difference is the derivative of the phase difference. The frequency of the reference clock is bound with respect to the data clock by the specification, so the rate of the phase adjustment must be sufficient to overcome frequency differences between the incoming data and the local reference clock.

The maximum rate of change that can be generated by the FSM is equal to the product of the step size and the update rate. However, practically it is often not possible to achieve this rate due to the fact that normal data does not have a transition on every bit (only a Nyquist pattern has this feature).

Large step sizes will track greater frequency deltas, since the phase will move farther with each adjustment. Since the phase detector is of the bang-bang type, the FSM will always dither by at least one step, this is also called self-noise. This dither is an error in the sampling location and is a consideration when choosing the maximum allowed step size. Smaller step sizes give smaller dither errors but cannot track larger frequency offsets.

The rate of the early/late decisions coming into the FSM is determined by the number of transitions in the data. Since this is a random quantity the FSM is not a regularly sampled

system if it operates on every possible transition and cannot be easily modeled. For 8b10b encoding the maximum run length is 5 UI and the average edge density is 30%.

## Step Size

Assuming that one step can be made every  $m$  unit intervals, the maximum rate of change is

$$(13) \quad \left( \frac{d}{dt} \phi \right)_{\max} = \frac{\text{stepsize}}{m \cdot UI}.$$

Where  $\text{stepsize}$  is the number of ps that the phase changes per step,  $m$  is the number of UI that pass (on average) before a step occurs, and  $UI$  is the UI size.

For example, with a 5 ps step size, an  $m$  value of 5, and a 200 ps  $UI$  value the maximum slew rate of the CDR is

$$(14) \quad SR_{\max} = \frac{5 \cdot 10^{-12} s}{5(200 \cdot 10^{-12} s)} = 5 \frac{ms}{s}$$

This is not sufficient to track the SSC profile described above, as it had a maximum slew rate of 8.3 ms/s. This is plotted in Figure .

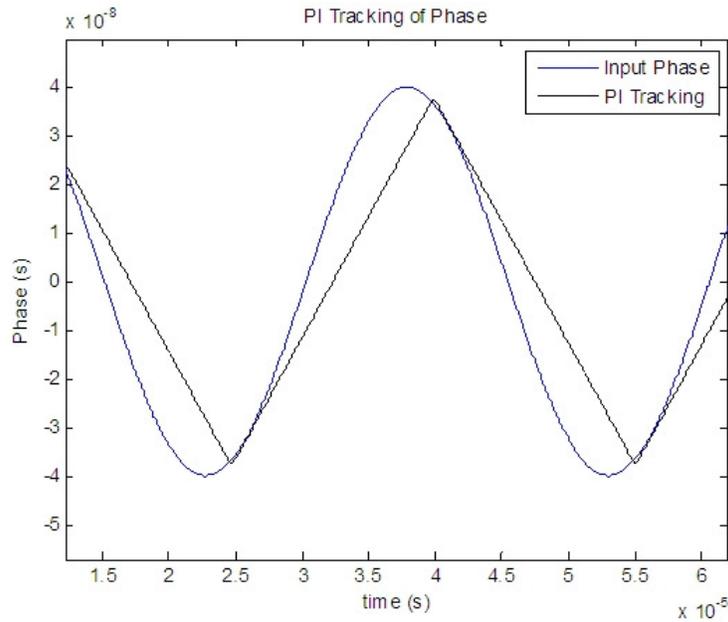
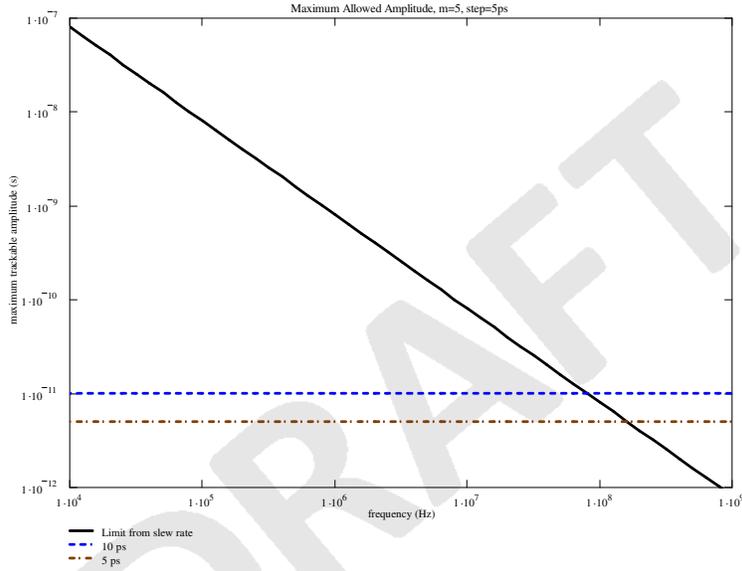


Figure 9: Tracking 40ns SSC with 5 ps step size,  $m=5$

In this figure we can clearly see that the CDR can't slew fast enough to keep up with the sinusoid. Therefore, for "perfect" tracking the CDR must be able to slew as fast as the maximum slewing point on the sinusoid. As we noted earlier, the maximum slew rate of a sinusoid is equal to the sinusoid amplitude multiplied its frequency (in radians). From this, the maximum slew rate of a sinusoid is given by equation (14):

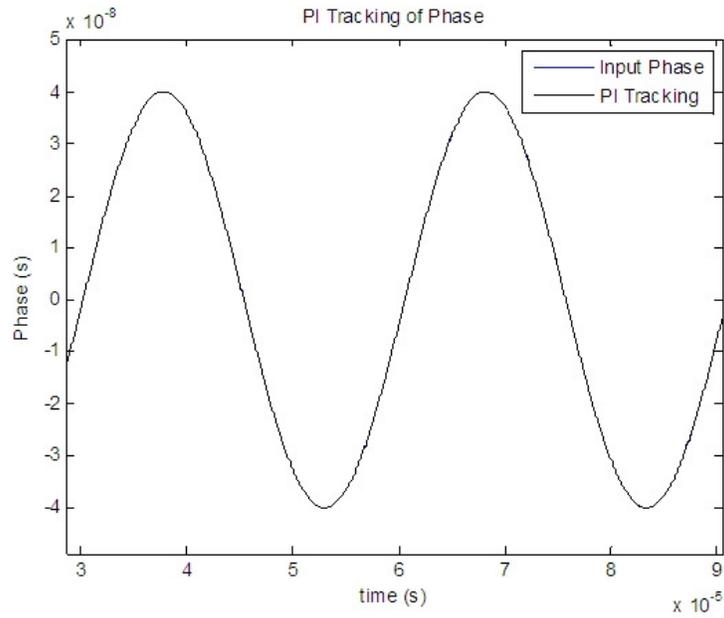
$$(14) \quad A_{\max} = \frac{0.005}{2\pi f}$$

and is shown in Figure . In this figure, the first horizontal line is an amplitude of 5 ps or one step, this occurs at about 100 MHz. The second horizontal line is 10 ps. This figure shows that for a given slew rate (5 ms/s) the maximum amplitude of a sinusoid at given frequency that can be tracked is fixed.



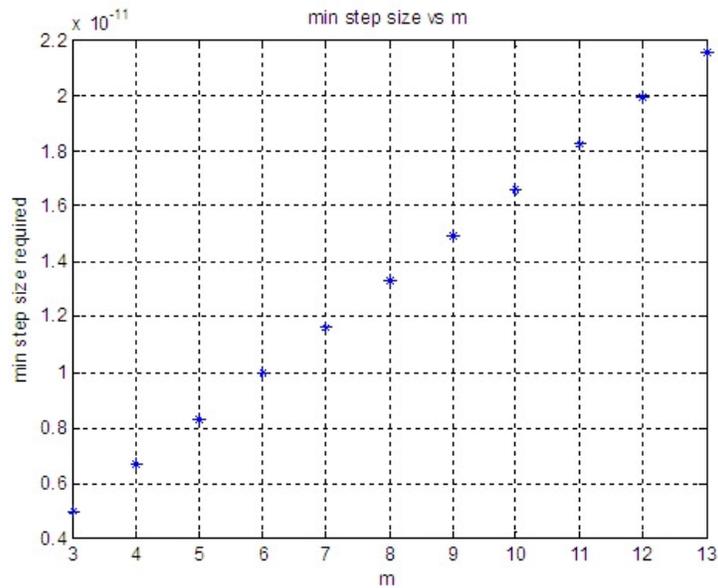
**Figure 10: Maximum amplitude traced by a 5 ps step with  $m=5$**

Assuming the maximum SSC slew rate of 8.3 ms/s, a 200 ps UI, and  $m=5$ , equation (13) gives the minimum step size of 8.3 ps. A CDR with this step size is shown to be properly tracking the SSC profile in Figure 4.



**Figure 4: Minimum step size tracking of SSC**

The larger the under sampling value  $m$ , the larger the step size required to be able to track SSC unless some form of gain is introduced into the control loop. For a 200 ps UI, the graph in Figure 5 provides the minimum step size required to track SSC as a function of the under sampling integer,  $m$ .



**Figure 5: Minimum step size required to track SSC vs under sampling integer  $m$**

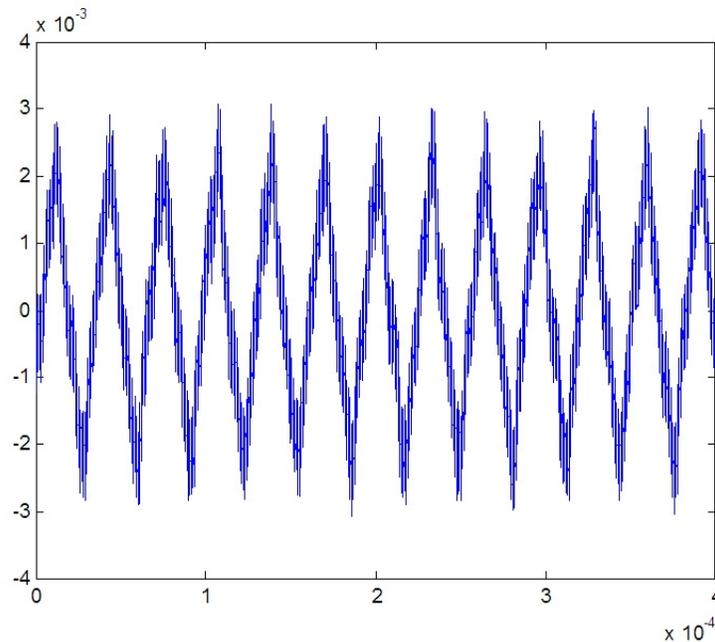
Although the larger step size does a better job of tracking SSC, it also causes a larger dither component that closes the eye. The amount of acceptable dithering at steady state sets the upper limit of the step size.

The desire of good response at low frequencies and attenuation of dither at high frequencies can be optimized with the proper filter design. Proper filter design can add gain to the low frequencies and attenuation at the higher frequencies, enabling small step sizes to track SSC.

## Slew Rate Limit

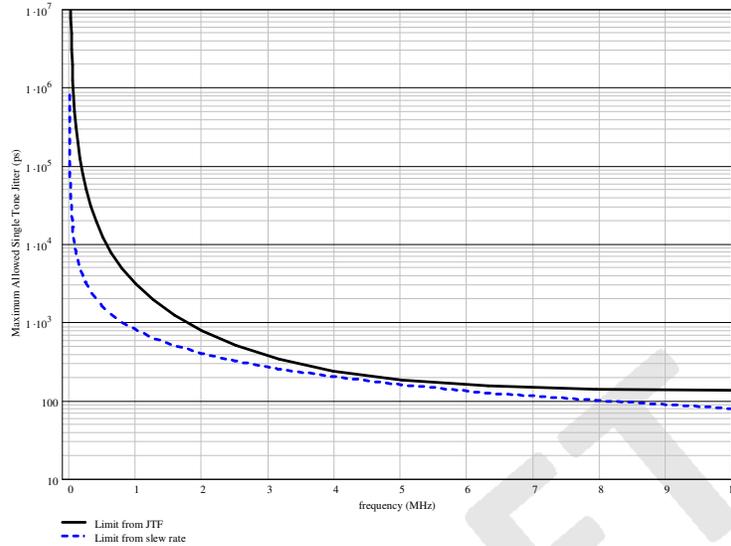
It should now be clear that the slew rate limit of the clock recovery circuit also sets a performance limit. The USB 3.0 specification requires a transmitter to output less than 10 milliseconds per second of phase jitter slew rate for any frequency component that lies within the bandwidth of the clock recovery circuit. This is relative to the worst case SSC clock. This limit allows the typical SSC profile to pass but prevents pathological SSC profiles from breaking the clock recovery circuit.

The slew rate limit is measured by using  $H$  on the time stamps,  $t_n$ , to obtain the phase jitter within the tracking bandwidth of the clock recovery function. The first difference is then taken to get the maximum slew rate. The maximum slew rate must not exceed the USB 3.0 specification of 10 ms/s for a two sided measurement, or 5 ms/s for a single sided measurement. An example of the slew rate limit measurement from the time record is shown below, the Matlab® code is given in appendix A.



**Figure 6: Example slew rate measurement**

The implications for the single tone jitter tolerance are that the slew rate limit sets the maximum allowed jitter amplitude in the tracking band of the clock recovery circuit, as Figure 7 shows.



**Figure 7: JTF and Slew Rate Limit single tone**

## Summary

The purpose of a Jitter Transfer Function (JTF) is to ensure that the jitter created by a transmitter can be tracked by a receiver. The jitter budget is important for interoperability between different components. It provides the design targets for the transmitter, system and receiver designers.

A key component of deriving the jitter budget is to understand the clock recovery model and the role of clock recovery in tracking the incoming jitter. The digital nature and limits of the clock recovery function has led to the introduction of the phase jitter slew rate limit and measurement. The clock recovery model used for the CDR is digital-based.

## Appendix A: Slew Rate Measurement

```
*****
%*
%*      Copyright (c) 2008 Intel Corp.
%*      Andy Martwick
%*
%*      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.
*****

close all
clear all
load('Test1.mat')           % load the file of the zero crossing times
T=mean(TDS_y_data); % this is the average period, measured against an absolute.
phi=cumsum(TDS_y_data-T); % integrate the period jitter to get the phase jitter

N=length(phi);           % N must be odd for the transforms to work
t=0:T:(N-1)*T;          % build the time record
F=1/(2*T);              % nyquist
df=F/(N/2);
f=0:df:(N/2-1)*df;      % build the frequency record
%phi=20e-9*sin(2*pi*33e3*t)'; % use this synthetic SSC to confirm the
transforms
X=fft(phi)/N;           % take the transform

% now build the transfer function from the spec
s=2*pi*f*(-1)^.5;
zeta=0.707
fc=10e6;
wn=(fc*2*pi)/(1+2*zeta^2+((1+2*zeta^2)^2+1)^.5)^.5
H=(2*s*zeta*wn+wn^2)./(s.^2+2*zeta*wn*s+wn.^2);
Y=X.* [1 H/2 conj(fliplr(H/2))]; % apply the transfer function
y=ifft(2*Y)*N;         % take the inverse and scale

slr=diff(y)/T;
plot(t(100:N-1000),slr(100:N-1000)) % ignore the beginning and end
```

## Appendix B: Slew rate Limit vs JTF

```
%*****
%*
%*      Copyright (c) 2008 Intel Corp.
%*      Andy Martwick
%*
%*      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.
%*****

zeta=0.707
fc=10e6;
wn=(fc*2*pi)/(1+2*zeta^2+((1+2*zeta^2)^2+1)^.5)^.5
f=0:1e3:50e6;

i=(-1)^.5
s=2*pi*i*f;

% these are from the spec:

H=(2*s*zeta*wn+wn^2)./(s.^2+2*zeta*wn*s+wn.^2);
ssc=40e-9* 1-(1-abs(2*pi*fc./(2*pi*i*33e3 + 2*pi*fc)))

JTF=1-H;
semilogx (f,20*log10(abs(JTF)), 'b');
hold on;
semilogx (f,20*log10(abs(H)), 'k');

axis([1e3 500e6 -18 3]);
grid on
% this is the slew rate limit spec.
srl=0.005/2/pi./f;
figure;

% from the spec this is what the eye closure can be at 1e-12
semilogy(f, abs(150e-12./(JTF)))
hold on
semilogy(f,srl,'k');
axis([-1e6 10e6 1e-11 1e-6]);

legend('Limit from JTF', 'Limit from PPM')
ylabel('Maximum Single Tone Amplitude Allowed')
xlabel('Frequency (Hz)')
```