jEye: A Statistical Channel Compliance Tool

Emerson S. Fang[1], Gerry Talbot[2] and Douglas Wallace[3]

Advanced Micro Devices

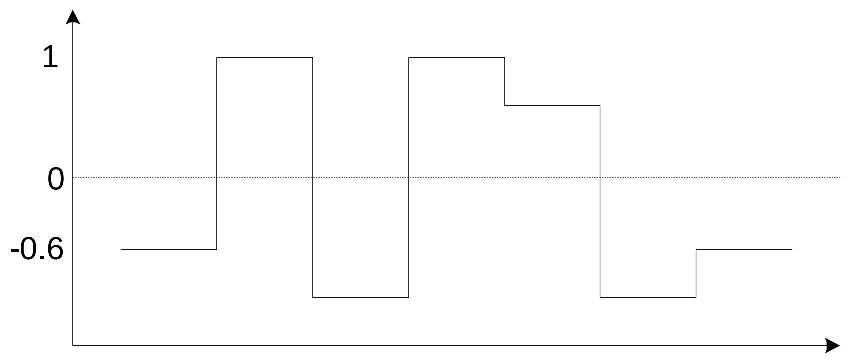
Introduction

In a system utilizing a standardized I/O interface like a Hypertransport™ link, the process of determining specification compliance is key to insuring successful system integration. The transmitter, channel, and receiver all have their own specification and methods defined to verify compliance with the specification. For channel specification, a frequency domain method based on an S-parameter mask (as used in XAUI for 10G Ethernet) can be used. However it is hard to translate frequency domain parameters to the BER performance metric, and other imperfections like jitter and DCD are typically measured and expressed as time domain parameters. Experimentally, the measured eye statistics can be easily mapped to BER data. It is therefore desirable to have the channel specification defined in terms of the eye diagram (i.e. the shape of the eye opening at some specific bit error rate). In order to verify a channel design will meet the specification, one has to be able to compute the eye. Many eye computation techniques use a set of representative patterns or other heuristics, but they are not very good at predicting the eye statistics at the low BER of 10^{-12}. Sanders, Resso, and D'Ambrosia [1] proposed a statistical method called StatEye as a computation technique based on pulse response and statistical convolution of a probability density function (PDF). In this paper, we described the techniques used in the HyperTransport 3.0 channel compliance tool called jEye which also computes a statistical eye but is based on step response.

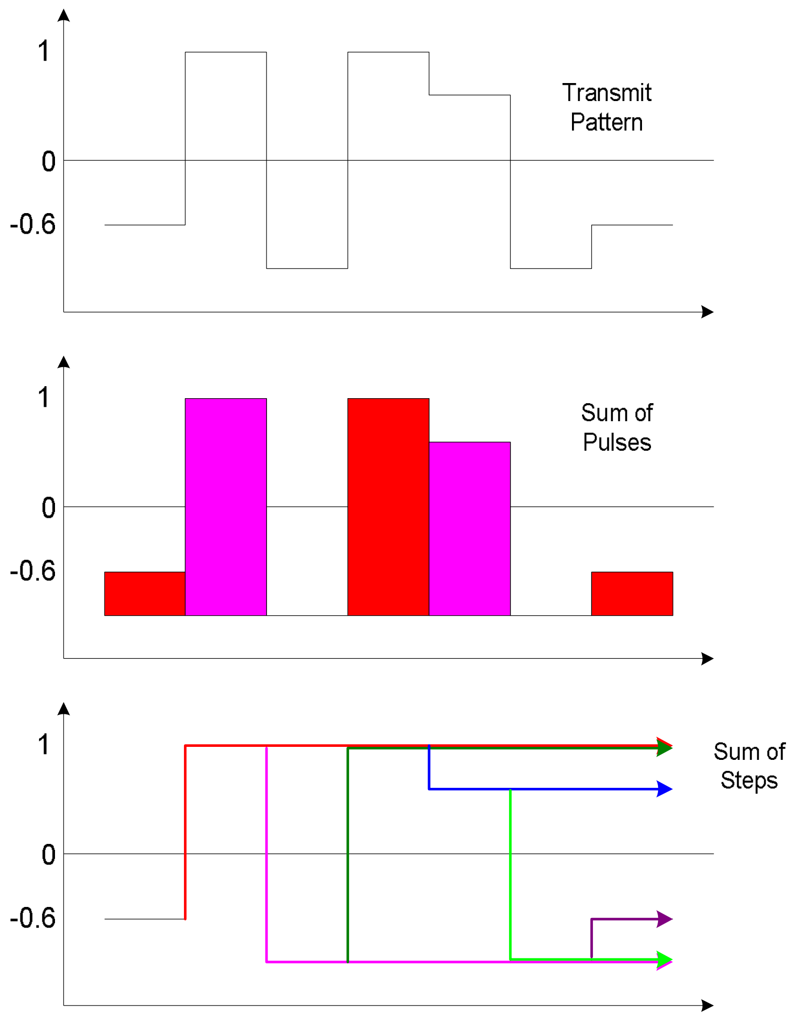
[1] *AMD, Sunnyvale, CA*
[2] *AMD, Boston, MA*
[3] *AMD, Austin, TX*

Section 1. Representation of Channel Response

The majority of present-day transmitters in a high speed transceiver support de-emphasis equalization. A transmit de-emphasis circuit is typically the easiest equalization to implement and is effective for combating ISI distortion in a transmission medium when transmitting NRZ signals. To calculate the channel output in response to a transmitter's excitation, the impact of the de-emphasis equalization has to be modeled correctly. The figure below is an example of transmit waveform for the digital sequence of 0101100.



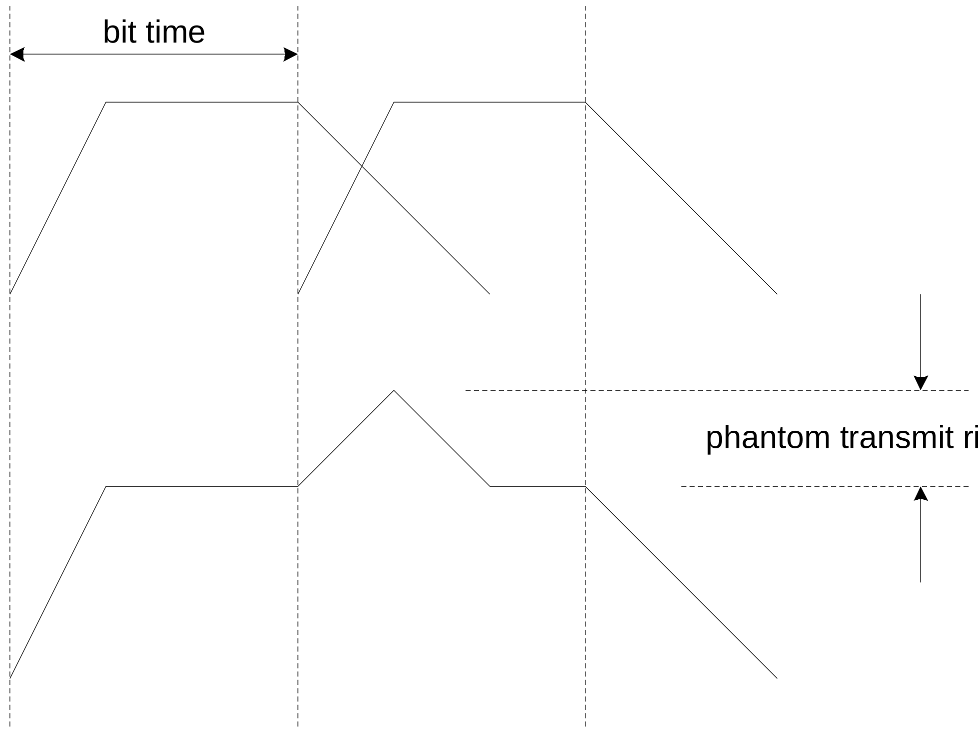
The example transmit filter used has one delay tap. The transmit waveform can be considered as the superposition of a series of pulse trains or a sequence of step waveforms of various amplitudes, as shown below.



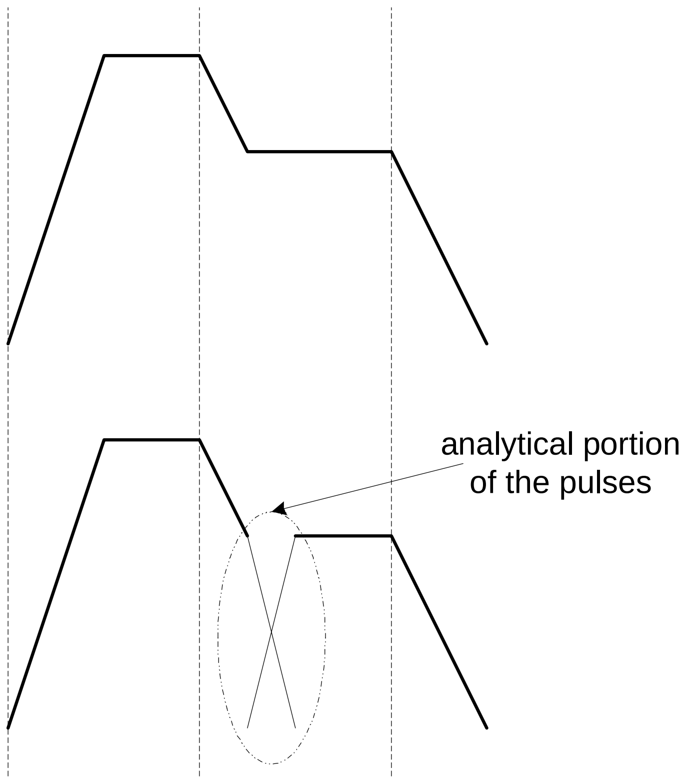
If the transmission channel is linear and time invariant, then by the superposition principle we can sum the channel responses to the individual pulses or steps. Given that the number of distinct pulses or steps is very limited, those pulse or step responses can be pre-computed, saved, and later used in a lookup table. Furthermore if a smaller amplitude pulse or step response waveform can be approximated as a linear scaled version of the largest one, then even fewer responses are needed (the minimum is one for the pulse response approach, and two for the step response approach; one for the rise step and one for the fall step).

In theory the two approaches are equivalent. However there are a few practical issues that make the step response approach easier to use for modeling the overall channel response accurately. One practical issue is the extraction of the basic responses. It is relatively easy to extract the responses using a field solver based simulator, but it is not always possible to get accurate models for the transmit driver, package, socket and other components that one would need to generate the responses using a simulator. Extracting the responses from an actual transmitter driving a real channel is certainly desirable and has higher fidelity. The transmitter driver itself may not be very linear, and some of the non-linear behavior would be captured in the extracted response. Although it may be possible to make a transmitter generate a pulse, it is problematic to make the pulse width of precise width. Furthermore, one has to get a collection of pulse responses for modeling at different data rates. The step response approach does not have these shortfalls. A more important advantage of the step response approach is the ease of incorporating transition-edge to transition-edge timing variation. The transmitter's cycle-to-cycle jitter and transmit clock duty cycle distortion can both result in such timing imperfections. A pulse response method would have to either maintain many different pulse width pulse responses or make some approximations to model this effect. This is rather cumbersome.

Even if one is willing to bear the burden of extra work, there is another issue with the pulse response method that limits its accuracy: its sensitivity to rise and fall transition matching. Consider the case of no transmit equalization, a long run of logic one is represented by a consecutive stream of pulses. The rise waveform of the pulse r(t), and fall waveform of the pulse f(t) need to be matched (i.e. $r(t) + f(t) = 1$), otherwise the signal at the transmit end will not be level and it will appear to have ripples at bit time boundaries as shown below. A transmit driver device is not perfectly linear and symmetric, particularly for single ended signaling. The transmit rise and fall waveforms can differ by a significant amount. A pulse response method using pulse responses extracted under these conditions will include phantom transmit ripples that affect the accuracy of the calculated channel response.



When transmit equalization is introduced, there are additional places where the pulse response method can introduce similar distortions. One example is shown below.

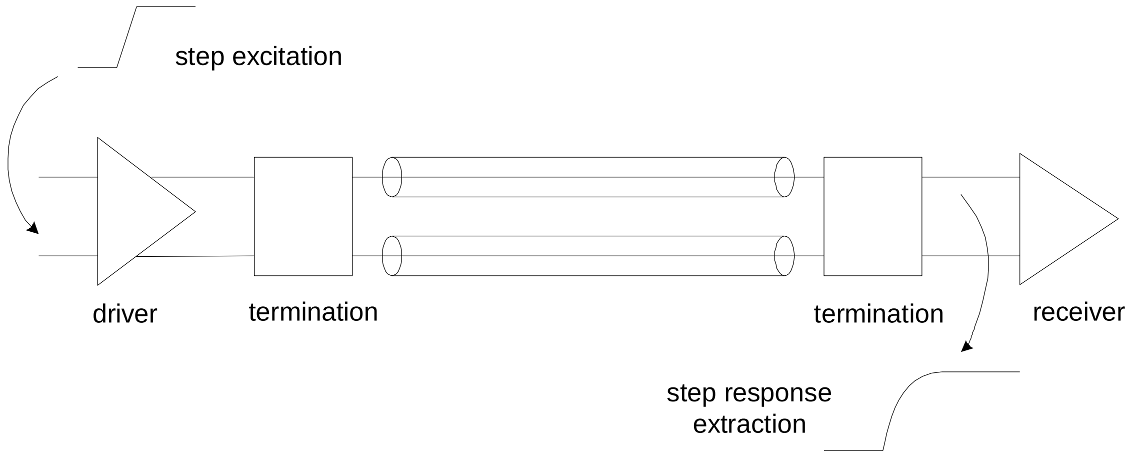


The two consecutive logic ones generated will cause the transmitter to generate a small falling step before the second one is outputted. The pulse response method will use two pulses to represent the two one outputs. A portion of the pulse has to be extracted from a real transmitter driving the channel, and a portion has to be constructed analytically since the transmitter never sent that portion and we need to make sure this portion does not cause unrealistic channel behavior like the phantom transmit ripple described above.

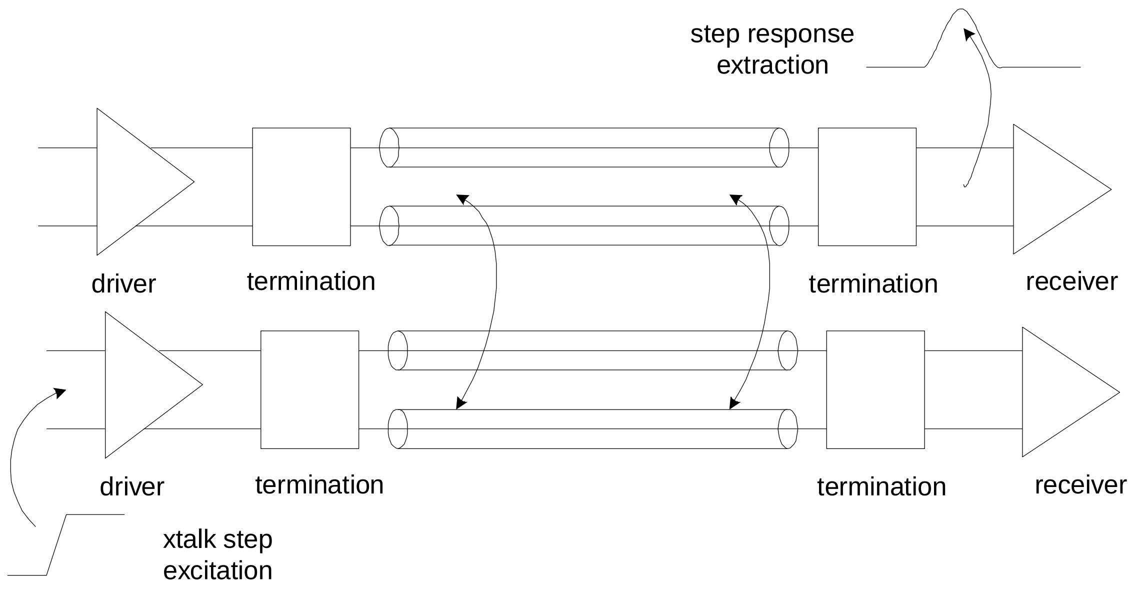
We should mention that there are frequency domain methods based on FFT which can produce the channel response as well. Channel characteristics are usually obtained based on frequency domain measurements (i.e. S-parameters). One could take the transmit waveform and transform it into the frequency domain and perform the channel response calculation by multiplication in the frequency domain and transform the result back to the time domain. This paper describes a time domain based approach, and therefore will not attempt to provide a detailed comparison between time domain eye calculation methods and frequency domain methods. However most of the channel compliance specifications are given in the time domain and it is non-trivial to convert them to the frequency domain, therefore we believe the time domain approach is the better choice.

Section 1.1. Extraction and Representation of Step Response

To accurately represent the channel, the following circuit topology should be used. In addition to the channel, one should include the driver with its termination network at the transmitting end, and include the receiver loading and termination at the receiving end.



Depending on the level of detail one wants to model, one can either extract just the differential step responses, or extract both the differential and common mode step responses. Of course, unless one models the common mode dependence of the receiver, one could not use the extracted common mode step responses. Crosstalk behavior in multiple coupled channels can also be modeled using step response as the example shows below. The channel response is then the sum of all the responses.



The step response will usually have a long tail. For optimal simulation time, one should limit the duration of the tail by forcing the step response to the final DC value after the step response has reached a value when the difference between it and the final DC value is small compared to the error tolerance. One should keep in mind that the current bit can be influenced by a step many cycles earlier and the influence is cumulative, so a relatively small error in the step response can result in a significant error in the simulated channel output. Each extracted step response is stored as a piece-wise linear (PWL) waveform in a file. The extracted and stored step response should have as high a resolution as possible. The jEye program has a filter and sub-sampling mechanism to allow a user to make the speed verses accuracy tradeoff at run time.

Section 2. Statistical Eye and Its Dependence on Past History and Transmit Equalization

At the end of a transmission line, the waveform of a receive bit is influenced by past bits and future bits. We will use a lossy but uniform transmission line as an example to help understand the number of past and future bits we need to include to calculate an accurate eye pattern. A signal or wave propagating along a distance z down a transmission line can be represented by [2]

$$W(z) = W(0) \times e^{-az} \times e^{-jbz}$$

where a is the attenuation constant and b is the phase constant. The phase velocity is given by

$$v_p = \frac{w}{b}$$

The parameters a and b are given by

$$a + jb = \sqrt{(R + jwL)(G + jwC)}$$

The loss constant a can be expressed as

$$a = \mathrm{Re}\left(w\sqrt{LC} \times \sqrt{\left(\frac{R}{wL}\frac{G}{wC} - 1\right) + j \times \left(\frac{R}{wL} + \frac{G}{wC}\right)} \right)$$

The phase constant b can be expressed as

$$b = \mathrm{Im}\left(w\sqrt{LC} \times \sqrt{\left(\frac{R}{wL}\frac{G}{wC} - 1\right) + j \times \left(\frac{R}{wL} + \frac{G}{wC}\right)} \right)$$

Furthermore, we have

$$\frac{1}{\sqrt{LC}} = \frac{3\times 10^{8}}{\sqrt{e_r}}$$

$$\sqrt{\frac{L}{C}} = 50\,Ohm$$

The relative permittivity of FR4 is about 4.7, which translates to

C = 144.5 pF/m

and

L = 361.3 nH/m

The α term represents loss, and frequency dependent components of β represent dispersion. The dielectric loss is characterized by loss tangent defined as

$$\tan d_D = \frac{G}{wC}$$

It is approximately constant with frequency. For FR4 material, the typical loss tangent value ranges from 0.015 to 0.035, a very good material like non-woven glass can achieve a loss tangent as low as 0.001. We will use a loss tangent of 0.02 for FR4 in our example here (A note of caution, a constant complex permittivity will cause a non-causal time domain response. This example is only used to illustrate the ISI duration not as a method to model or calculate channel response).

The resistive loss of a metal trace is very sensitive to skin effect. The DC resistance for a 4 mil wide and 2 mil thick trace is given by

$$R_0 = \frac{1}{s\,wt} = 3.3\,Ohm/m$$

σ is 5.8×10^{7} S/m for copper, and w and t are the trace width and thickness respectively. The skin penetration depth of copper is given by

$$d = \frac{0.066}{\sqrt{f}}$$

At a frequency of 2.5 GHz, the penetration depth is only 1.3 um. The effective metal thickness is between δ and 2δ, depending on if we have a microstrip or a stripline structure. We can therefore approximate R in the skin effect frequency region as

$$R = R_0 \times \left(1.0 + \frac{t}{0.066k} \times \sqrt{f} \right)$$

where R_0 is the DC resistance value, and k is a value between 1 and 2. Assume k=2, we obtained the following table of α and β values at different frequency points.

f	$\frac{R}{R_0}$	$\frac{R}{wL}$	$\left(\frac{g}{w\sqrt{LC}}\right)^2$	a	$\frac{b}{w\sqrt{LC}}$
10.0 GHz	*39.50*	*0.00574*	*-0.9999+j0.02574*	*5.84*	*1.0000*
5.0 GHz	*28.22*	*0.00820*	*-0.9998+j0.02820*	*3.20*	*1.0000*
2.5 GHz	*20.25*	*0.01177*	*-0.9998+j0.03177*	*1.80*	*1.0000*
1.25 GHz	*14.61*	*0.01699*	*-0.9997+j0.03699*	*1.05*	*1.0000*
625 MHz	*10.63*	*0.02472*	*-0.9995+j0.04472*	*0.634*	*1.0000*
250 MHz	*7.09*	*0.04123*	*-0.9992+j0.06123*	*0.347*	*1.0001*
125 MHz	*5.30*	*0.06164*	*-0.9988+j0.08164*	*0.232*	*1.0002*
62.5 MHz	*4.04*	*0.09397*	*-0.9981+j0.11397*	*0.162*	*1.0007*
25 MHz	*2.93*	*0.17037*	*-0.9966+j0.19037*	*0.108*	*1.0030*
12.5 MHz	*2.36*	*0.27445*	*-0.9945+j0.29445*	*0.083*	*1.0079*
6.25 MHz	*1.96*	*0.45587*	*-0.9909+j0.47587*	*0.011*	*1.0223*
2.5 MHz	*1.61*	*0.93616*	*-0.9813+j0.95616*	*0.050*	*1.0843*
1.25 MHz	*1.43*	*1.66300*	*-0.9667+j1.68300*	*0.040*	*1.2057*
625 KHz	*1.30*	*3.02364*	*-0.9395+j3.04364*	*0.030*	*1.4361*

With the exception of the frequency dependent β term which represents dispersion, the loss behavior is similar to a RC low pass network. At data rate of 5 Gbps, the frequency components near 2.5GHz shows very little dispersion at a distance of 1 meter or less. At 1 meter, the attenuation factor of our example FR4 line is 0.165 or 15.6dB at 2.5GHz. The loss characteristics can be approximated roughly as a low-pass RC filter with RC time constant of 380ps. It is well known that the response of an RC network to a pulse excitation at time zero of width T_{PW} is $1 - e^{-t/RC}$ for $t <= T_{PW}$, and $\left(e^{T_{PW}/RC} - 1\right)e^{-t/RC}$ for $t > T_{PW}$. The peak occurs at time T_{PW}, so the optimal sampling point is shifted by the pulse width from the leading edge compared to the unfiltered signal where the optimal sampling point is at $T_{PW}/2$. This shows that we have a pre-cursor duration of $T_{PW}/2$. The real transmission line will show slightly more pre-cursor duration due to more complex loss characteristics and dispersion, but it is usually within 1 bit time duration for a distance of within 1 meter and data rate of 10 Gbps or less.

The post-cursor waveform lasts much longer. The question we want to answer is: "Beyond which post-cursor bit we can safely ignore the past history effect on the cursor bit?" We again use the RC filter approximation to give an estimate. The contribution by post-cursor bits that are k bits or earlier than the cursor bit can be expressed as

$$\sum_{i=k}^{\infty}\left(e^{T_{PW}/RC}-1\right)e^{-iT_{PW}/RC}=\sum_{i=0}^{\infty}\left(e^{T_{PW}/RC}-1\right)e^{-kT_{PW}/RC}e^{-iT_{PW}/RC}=e^{-(k-1)T_{PW}/RC}$$

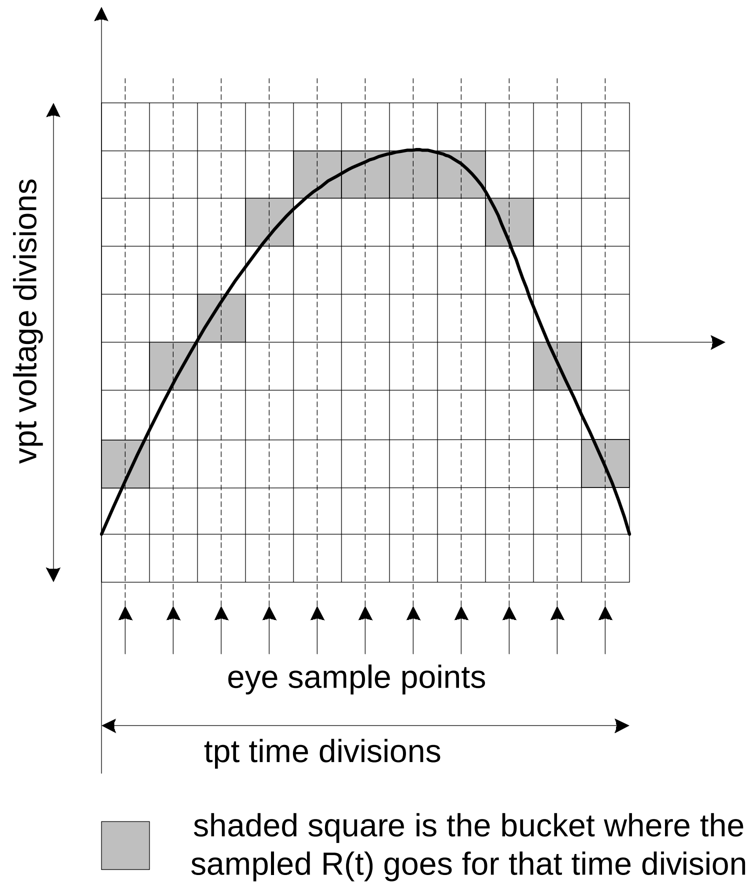
If we set the criteria that the post-cursor bits we ignore should contribute no more than 1% of the received cursor bit peak amplitude, we can solve for *k* as

$$k = 1-\frac{RC}{T_{PW}}\ln(0.01\times(1-e^{-T_{PW}/RC}))$$

At 5 Gbps, *k* comes out to be 12 for our example. The actual transmission line is more complex; higher frequency loss is steeper and there is less low frequency attenuation, which leads to a longer ISI tail than the simple RC estimation. One therefore should set *k* to 2 to 4 times the amount predicted here to guarantee accurate results. A practical number to use at or below 5 Gbps would be about 40 and it goes up to about 80 as the data rate approaches 10 Gbps. For shorter or lower-loss lines with discontinuities, the round trip delay of reflection has to be taken into account. This can further increase the duration of past history we need to retain. The measured or extracted step response can give a good indication of the magnitude of the reflection and the time for reflection to settle out.

Section 2.1. Statistical Eye for a Random Data Stream

Our previous analysis shows that the channel response of the cursor bit can be determined from the response to a finite number of bit durations (we call it the search length ***n***) that includes the cursor plus one or two pre-cursor bits and a set of post-cursor bits. For every one of the possible *2^n* patterns, if we calculate the response ***R(t)*** around the sampling point, and map into a two dimensional discretized voltage-time PDF map as shown below, we will obtain the statistical eye after all the combinations have been accounted for.


vpt voltage divisions
eye sample points
tpt time divisions
shaded square is the bucket where the sampled R(t) goes for that time division

The crosstalk channels' PDFs can be calculated this way as well. The PDFs of all the channels can be convolved together to form the final eye. Of course for *n* at 40 or larger, the brute force search of all 2^n patterns is impractical. We can divide the n-bit search length into groups of *r*-bits each. The PDF of each individual *r*-bit group can be calculated and the PDFs of all the *r*-bit groups can be convolved together to give the PDF for that channel. The extreme case is with *r=1*. However in order to minimize quantization error and the accumulation of errors in the convolution process, we have to use a reasonably fine voltage grid which makes convolution computationally expensive. The optimal size of *r* is usually a small number but larger than 1. Sanders et al. [1] provide a discussion on the theory and the convolution process of PDFs, and showed that quantization error and its accumulation over the convolution process will result in an error variance of $C\frac{e_{\max}^2}{12}$, where C is the number of convolutions and $e_{\max}$ is the peak quantization error per voltage bin or half of a voltage grid.

Section 2.2. The Impact of Transmit Equalization on Eye Pattern Search

The channel response to an *n*-bit data excitation can be expressed as

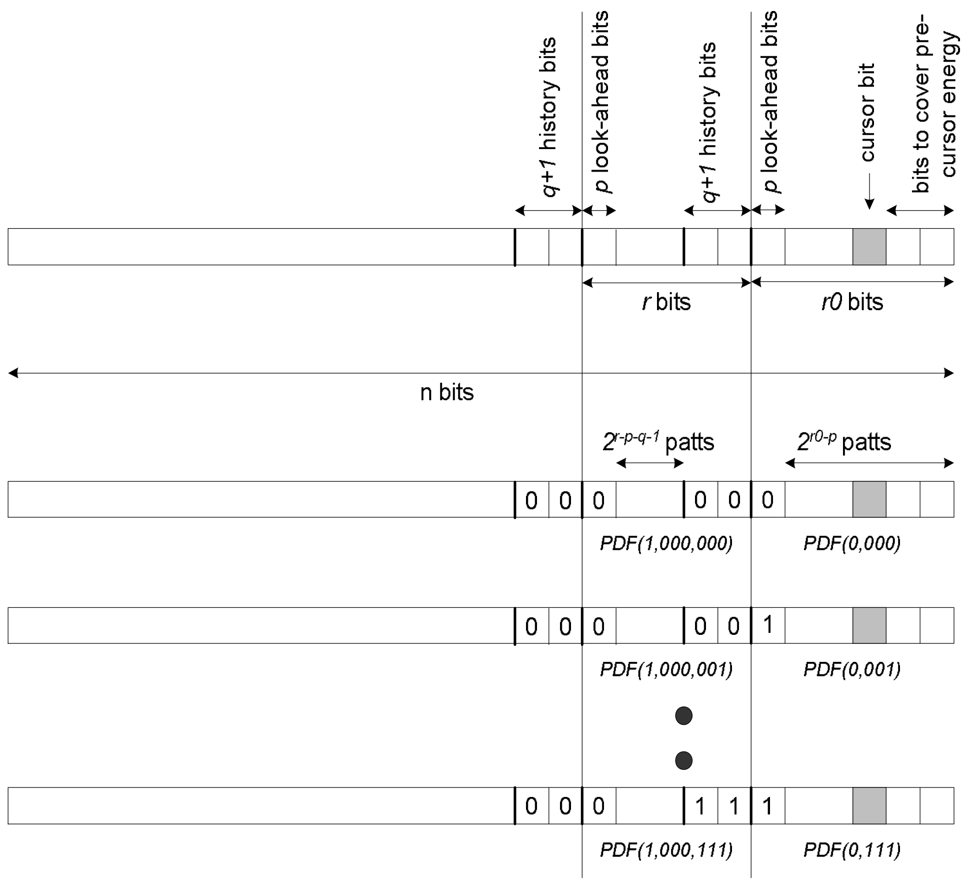
$$R(t) = \sum_{i=0}^{n-1} d_i P(t - iT_{bit})$$

where d_i is the *i*-th data bit in the *n*-bit search pattern and *P(t)* is the pulse response of the channel. With a transmit de-emphasis equalizer of *p*-tap pre-cursor and *q*-tap post-cursor equalization, the channel response changes to

$$R(t) = \sum_{i=0}^{n-1} \left(\sum_{j=i-q}^{i+p} c_{j-i} d_j \right) P(t - iT_{bit})$$

where c_k's are the coefficients of the equalizer. One may argue instead of two possible pulse responses for a given bit, we now have 2^{p+q+1} possible pulse responses with equal probability of occurring on a random data stream. If we set our search group size *r* to one, then we have to evaluate 2^{p+q+1} pulse responses to generate a PDF for the *r*-bit subgroup instead of 2 and we can still convolve *n* times to get the final PDF for the *n*-bit length pattern. However there is a serious flaw in this assertion, because in order to obtain a PDF through convolution, any two adjacent *r*-bit search subgroups have to be statistically independent. Clearly with an equalizer calculating the current value based on the past and the future, assuming statistical independence of adjacent search subgroups is no longer correct.

In order to make the convolution of PDFs valid, we have to separate out the correlation term by tracking the history that couples the two r-bit search subgroups together. The history tracking is illustrated in the figure below for the first two r-bit groups. The first r-bit group contains the cursor bit. The history length is *p+q+1* bit long. The pre-cursor history bits belong in the current *r*-bit group and the *q+1* bits of history belong to the next *r*-bit group. We assume *r* > *p+q+1*, and allow the first *r*-bit group (labeled as 0-th group) to have *r0* bit long with *r0* at least as large as *r*.



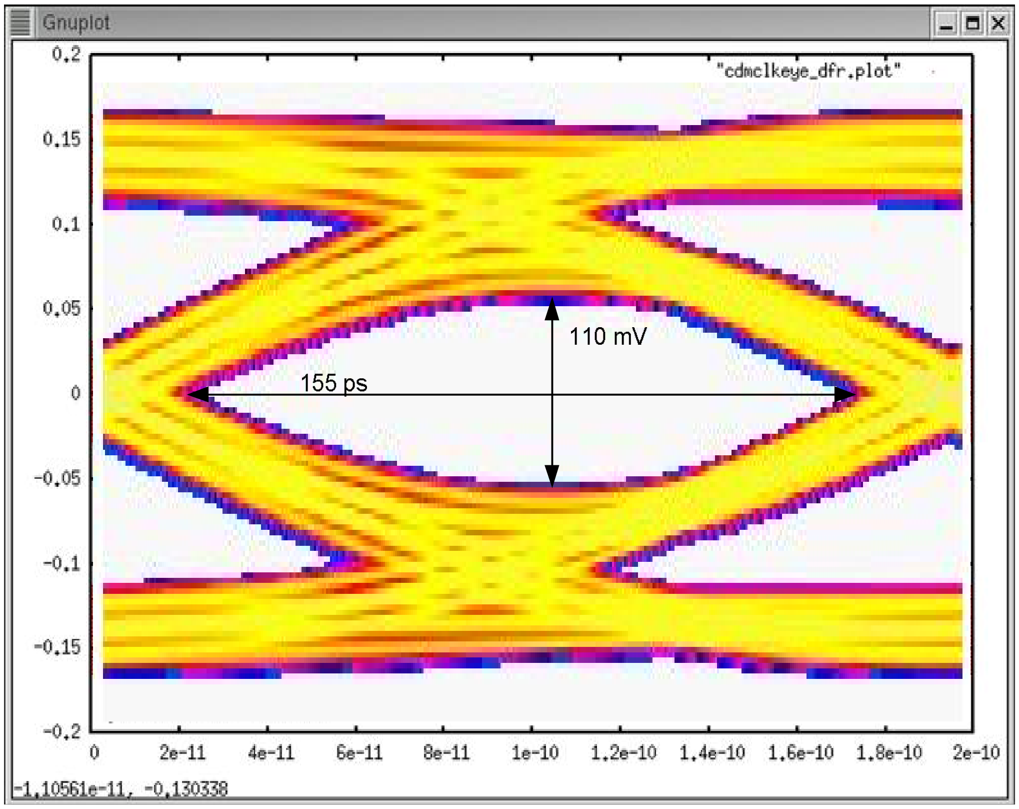
For the 0-th r-bit group, we calculate the r-bit PDF for each of the unique history patterns, we label each PDF as *PDF(0, x)* where 0 is the *r*-bit group index and *x* is the history pattern index. This process will evaluate 2^{r+q+1} patterns. The next *r*-bit group PDFs are calculated according to the following. For each of its history pattern *y*, we calculate the PDFs such that the pattern terminates with each of the previous *r*-bit group's history pattern *x*, so the PDFs for the next *r*-bit groups are labeled as *PDF(1, y, x)* where *y* is this group's history pattern and *x* is this group's terminating pattern or the previous group's history pattern. It will take $2^{r+p+q+1}$ patterns to generate all the *PDF(1, y, x)*. The two *r*-bit groups' PDFs can be combined to form a two *r*-bit group PDF through convolution given by

$$PDF(0+1,y) = Normalize\left(\sum_{x=firsthist}^{lasthist} PDF(1,y,x) \otimes PDF(0,x)\right)$$

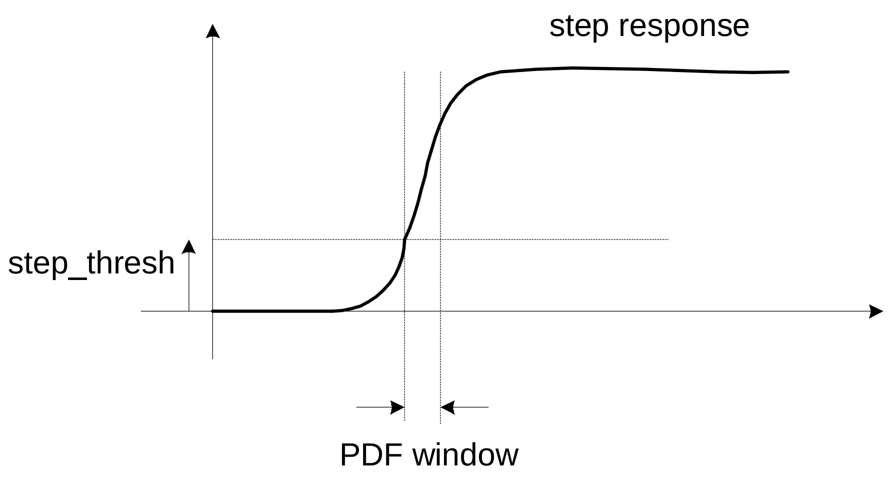
It takes 2^{p+q+1} convolution to combine the two groups together. One easily infers that the process can be repeated to obtain the PDF for the n-bit search length. The final step is to sum and normalize:

$$PDF_{channel} = Normalize\left(\sum_{x=firsthist}^{lasthist} PDF(0+1+\mathbf{K},x)\right)$$

The computation complexity is $O(2^{r+p+q+1})$ pattern searches and $O(2^{p+q+1})$ convolutions of PDFs. The storage requirement is $O(2^{p+q+1}\times vpt\times tpt)$ of float. The figure below is an example of an eye pattern generated by jEye.

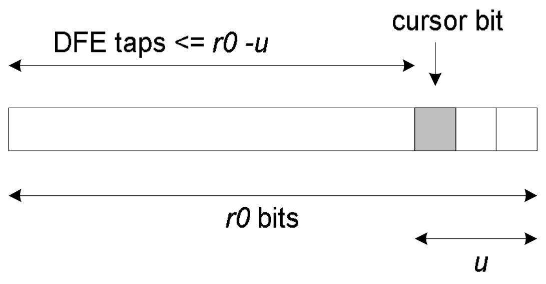


Due to storage and computation time limitation, the calculated PDF is typically limited over a time interval of 1 to 2 UI. This requires that the PDF window overlays the cursor eye region reasonably well. The PDF window is specified relative to the step response through the step_thresh parameter as illustrated in the figure. By moving the step_thresh value up or down, one can shift the relative position of the PDF window to the step response, and center the cursor eye into the PDF window.



Section 3. Receive Equalization with DFE and 8B10B Coded Data Stream

Decision feedback equalization is a popular receive-side equalization method. The effective eye when DFE is applied can be easily included if we set the first r-bit group length, r0, to cover the taps of the equalizer as shown.



The DFE timing is specified through the dfe_start parameter. It is a relative offset from the start of the PDF or eye window. One normally generates the eye pattern without DFE, then determines the cursor eye location and sets the dfe_start parameter to generate the eye pattern with DFE turned on. It should be noted that the DFE feature in jEye does not account for the error propagation effect (it assumes the receiver always samples the data correctly regardless of the size of the eye opening), so one has to take that into account to calculate the final BER.

When a data stream is encoded with 8B10B code, the bit patterns are no longer completely random but rather correlated by the 8B10B coding rule. The pattern search has to select the patterns that are valid 8B10B codes. The jEye tool accomplishes this by forcing *r* to be 10 and *r0* to be 16, since the encoded 10-bit code is a concatenation of a 6-bit code followed by a 4-bit code [3]. We follow the normal *r*-bit pattern search process to generate the PDFs, except the patterns are restricted to valid 8B10B codes, and the history tracking has to remember an additional piece of history which is the 8B10B running disparity. The disparity history is tacked on as an imaginary transmit equalization tap.

Section 4. Inclusion of Timing Jitter in the Eye Calculation

Transmit timing jitter can be separated into DJ and RJ components. The DJ component is typically modeled using the Dual Dirac Jitter approach, or DDJ. The RJ component is treated as Gaussian. If we consider the transmit jitter's spectral content, the portion of the jitter spectrum that gets attenuated by the channel will modify the magnitude of the jitter in the time domain. The net effect of this is a magnification of transmit pulse width jitter by the channel and creates some complication in correctly modeling its effects with a statistical tool. A good approximation is to separate the transmitter's jitter components spectrally by creating a transmit specification which is pulse width jitter included within the normal DDJ component. Furthermore RJ has most of its spectral energy at low frequency and so passes through the channel without modification. Once we have a specification of transmit pulse width jitter we can make the conservative assumption that this occurs on adjacent bit times, i.e. a transmitted bit width alternates between minimum and maximum pulse width, equivalent to transmit DCD. In practice we note that a real transmitter will show a distribution between the two pulse width extremes however, as a typical transmitter tends to have a bi-modal distribution, the approximation of pulse width jitter as DCD jitter at a BER of 10^{-12} is reasonable and conservative.

Transmitter DCD jitter timing can be easily included in the eye search process. To obtain the eye with DCD jitter, we perform the eye calculation twice. The first time we calculate the eye PDF with a duty distortion that is $(0.5-\delta)/(0.5+\delta)$, then we obtain the second eye PDF with duty distortion set at $(0.5+\delta)/(0.5-\delta)$ for a peak-to-peak DDJ of 4δ fraction of a UI. The two eye PDFs are summed and normalized to give the eye with DCD jitter or pulse width jitter.

This leaves the rest of the DDJ and RJ from the transmitter. As these two distortions pass through the channel without any distortion they can be convolved with final PDF from jEye to give the final eye closure as seen at the receiver.

Section 5. Conclusion

We have presented the step response approach to represent a channel response, and discussed the advantage of step response over pulse response. Various methods can be used to obtain the step response needed for jEye, some of which are shown in this paper. The algorithm for PDF calculation and a way to de-correlate data patterns caused by de-emphasis filtering are shown. In addition, methods to include receive side DFE equalization and statistical eye generation involving 8B10B coded data stream are presented. The jEye tool suite (known as jEyeII [4]) is available for HyperTransport^{TM} consortium members.

References

[1]. Anthony Sanders, Mike Resso, John D'Ambrosia, "Channel Compliance Testing Utilizing Novel Statistical Eye Methodology", DesignCon 2004.

[2]. Ramo, Whinnery, VanDuzer, "Fields and Waves in Communication Electronics", Wiley, 1994.

[3]. A. X. Widmer, P. A. Franaszek, "A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code", IBM J. Res. Develop. Vol. 27. No. 5, Sept. 1983.

[4]. http://www.hypertransport.org/members/docucontrol_viewprevious.cfm?docnumber=HTC20061028-0097-0001