DesignCon 2017

IBIS-AMI Modeling of Asynchronous High Speed Link Systems

Hongtao Zhang, Xilinx Inc.
hongtao@xilinx.com

Fangyi Rao, Keysight Technologies
fangyi_rao@keysight.com

Daniel (Zhaoyin) Wu, Xilinx Inc.
danielw@xilinx.com

Geoff Zhang, Xilinx Inc.
geoff.zhang@xilinx.com

Abstract

IBIS-AMI modeling for high-speed serial link systems becomes the de facto standard in the industry. The evolution of IBIS-AMI modeling standard started from a pure THRU channel modeling for NRZ signaling and expanded to the inclusion of crosstalk aggressors, to links containing repeaters, to back-channel transmitter (TX) and receiver (RX) equalizer training process, to the modeling of PAM4 and duobinary signaling.

However, to date IBIS-AMI modeling can only deal with a synchronous system, indicating that the TX and RX share a common reference clock source. In real applications, there are many more systems that are designed for asynchronous operations, i.e., there exists certain frequency offset between the reference clocks for the transmitter and the receiver. Thus, clock-data-recovery (CDR) behavior in the case of frequency offset between the TX and the RX is not verified through the standard IBIS-AMI simulation. As a result, the impact of the frequency offset is not rigorously evaluated, leading to potential overoptimistic estimation of system performance.

In this paper we propose an approach such that an asynchronous high-speed link system can be modeled within the existing IBIS-AMI framework, making it possible to study the dynamics of the CDR under the asynchronous condition through time domain simulations. The paper explains what it takes to perform asynchronous link system simulations. Details of enhancements to the existing modeling and simulation practice required to capture the asynchronous effect are described. Specific examples of asynchronous links are analyzed using the proposed approach, and simulation results are presented. The behavior of the CDR in the presence of reference clock frequency offset is demonstrated, and the consequent timing impairment is measured. The system tolerance to frequency ppm offset is investigated with channels having different loss profiles at different data rates. Finally, the impact of asynchronous TX and RX clocks on link budget and system performance is discussed.

Authors Biography

Hongtao Zhang received his Ph.D. degree in Electrical and Computer Engineering from University of California, San Diego in 2006. He joined Xilinx in 2013 and is now a senior staff Design Engineer, working on SerDes architecture development and circuit design. From 2010 to 2013, he was with SerDes design team at Oracle Corporation, where he worked on circuit design and architecture modeling. Prior to that, he worked on SerDes characterization at Texas Instruments, Dallas. His current interests are SerDes architecture development and modeling, high speed mixed-signal circuit design and optimization, and system level modeling.

Fangyi Rao is a master R&D engineer at Keysight Technologies. He received his Ph.D. degree in theoretical physics from Northwestern University. He joined Agilent/Keysight EEsoft in 2006 and works on Analog/RF and SI simulation technologies in ADS. From 2003 to 2006 he was with Cadence Design Systems, where he developed SpectreRF Harmonic Balance technology and perturbation analysis of nonlinear circuits. Prior to 2003 he worked in the areas of EM simulation, nonlinear device modeling, and medical imaging.

Daniel (Zhaoyin) Wu is the Senior Staff Design Engineer in the SerDes technology Group at Xilinx, Inc since 2010. He is working on SerDes modeling/architecture design and system simulation with SOC and 3D-IC. He was also in charge of electromagnetic design/modeling of spiral inductor, finger capacitor (MOMcap), LC-tank, and on-chip signal/clock routing in Xilinx. He was with Ansoft Corp, AltraBroadband Inc., and ITRI before joining Xilinx. He is experienced on both RFIC circuit and passive component designs such as UHF RFID Tag, UWB Transceiver, GSM900/1800, On-Chip Spiral/Transformer/Balun, Multilayered LTCC/Organic RF Filter, Planar Antenna, and etc. He has 5 published papers in the area of circuit/electromagnetic design, and 2 US patents for spiral inductor and 1 US patents for MoM capacitor respectively.

Geoff Zhang received his Ph.D. in 1997 in microwave engineering and signal processing from Iowa State University, Ames, Iowa. He joined Xilinx Inc. in June, 2013. Geoff is currently Distinguished Engineer and Supervisor, in transceiver architecture and modeling under SerDes Technology Group. Prior to joining Xilinx he has employment experiences with HiSilicon, Huawei Technologies, LSI, Agere Systems, Lucent Technologies, and Texas Instruments. His current interest is in transceiver architecture modeling and system level end-to-end simulation, both electrical and optical.

1. Introduction

This paper introduces Input/Output Buffer Information Specification Algorithmic Model Interface (IBIS-AMI) modeling and simulation of asynchronous high speed link systems. However, to date IBIS-AMI modeling can only deal with a synchronous system, indicating that the TX and RX share a common reference clock source. In real applications, there are many more systems that are designed for asynchronous operations, i.e., there exists certain frequency offset between the reference clocks for the transmitter and the receiver. Thus, clock-data-recovery (CDR) behavior in the case of a frequency offset between the TX and the RX is not verified through simulations. Hence, the impact of the frequency offset is not rigorously evaluated, leading to potential overoptimistic estimation of system performance.

To accurately simulate a serial link, it is critical to model the link channel and SerDes (or other active component, such as repeaters or retimers) accurately, besides setting up system parameters such as jitter and noise. Inside the SerDes we need to model the behavior of feed-forward-equalization (FFE), continuous-time-linear-equalization (CTLE), decision-feedback-equalization (DFE) and CDR, both hardware and adaptations. Impairments and design trade-offs also have to be represented inside the model. However, such information is typically proprietary for SerDes vendors, thus unavailable to system engineers. This poses a challenge to the system simulation.

Another challenge is model interoperability. As a third-party SerDes IP is usually required in many application scenarios, there is a need to establish a common interface standard for interoperability simulation. Still another challenge is simulation speed. As most design specs are defined at bit-error-ratio (BER) of 1e-12 or lower, designers need to run millions of bits in order to predict a link performance at very low BER levels with good statistical confidence.

Above challenges are addressed by the IBIS-AMI standard. By defining a common SerDes model interface, the standard allows SerDes vendors to encapsulate SerDes behaviors in model executables without exposing their IP. The models can be used by system designers to perform end-to-end link simulations. Furthermore, in AMI simulations analog channels are assumed to be linear-time-invariant (LTI) and can be represented by impulse responses. The highly efficient convolution method can be applied to calculating signal waveforms at channel outputs. As a result, millions of bits can be simulated in minutes, allowing accurate predictions of link performance at low BER with good confidence.

This paper proposes details of modifications to the current IBIS-AMI simulation flow so that asynchronous systems can be simulated for a given frequency ppm offset. The paper provides background information of IBIS-AMI basics, highlight of serial interface link analysis, and PLL and CDR fundamentals. The paper then prescribes details on how the existing system can be modified such that the simulation scope can be extended from the synchronous system to both synchronous and asynchronous systems. By doing so the IBIS-AMI modeling becomes a step forward toward more comprehensive solution.

The paper provides examples using the modified platform to analyze asynchronous links. An example is also presented to show link margin is over-optimistically represented if CDR performance in the presence of frequency offset is ignored. The newly proposed IBIS-AMI platform will help the industry to better analyze link performance and budget system designs.

2. IBIS-AMI Modeling Basics

A brief discussion of IBIS-AMI modeling for an NRZ link is presented in this section. AMI defines the SerDes behavioral modeling interface and an efficient channel simulation methodology. A serial link consists of a TX, a physical channel and a RX. Each SerDes device (TX or RX) is represented by an IBIS-AMI model, which contains analog and algorithmic portions. The analog portion is a regular IBIS model, and the algorithmic part is a Dynamic Link Library (DLL) executable of a data flow model.

In a typical TX model, the analog portion models rise and fall waveforms and the output impedance and the DLL the de-emphasis. In a typical RX model, the analog portion represents the input termination and the DLL the functionalities of AGC, equalization (such as CTLE, FFE and DFE) and CDR. The TX DLL output is considered an ideal voltage source, and the RX DLL input is assumed to have high impedance. Therefore, DLLs are electrically decoupled from the analog channel, which includes the TX analog model, the physical channel and the RX analog model. Furthermore, the analog channel is assumed to be LTI, thus can be represented by a combined analog channel impulse response. A graphical representation is given in Figure 1.

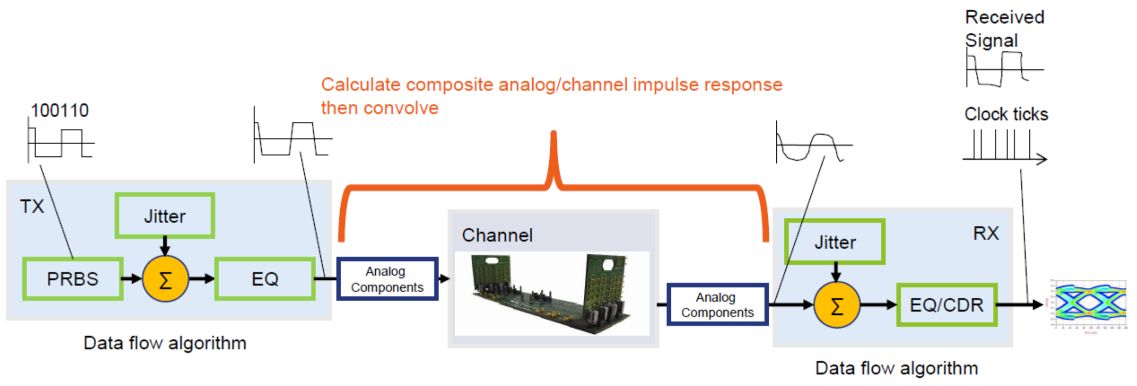


Figure 1. Graphical representation of IBIS-AMI modeling.

In AMI simulations, the TX DLL input is a square wave switching between 0.5V and -0.5V that represents the data pattern. The transmitted data rate is controlled through AMI parameter bit_time and sample_interval. The TX output is convolved with the analog channel impulse response. The highly efficient FFT algorithm can be employed in the convolution calculation. The resulting signal is the input to the RX DLL, which applies equalizations and CDR to it and returns the equalized signal and the recovered clocks.

The expected data rate on the RX side is again determined from the AMI parameter bit_time and sample_interval. The RX output is sampled at each clock time and compared with the reference voltage at 0V and the transmitted bit to compute the BER. If the RX DLL has the *AMI_GetWave* function, the RX signal processing is performed inside the function. In a typical setting, the RX input waveform is divided into segments. The simulator repeatedly calls *AMI_GetWave*, using sequentially the waveform of each segment as the input of each function call until all segments are processed. The recovered clock tick information is passed from the RX DLL to the EDA tool through the AMI parameter clock_times.

It is noted that the frequency offset between the TX and the RX is not explicitly handled, thus the existing platform cannot handle asynchronous link simulations. This is what this paper is discussing and is trying to accomplish.

3. Synchronous and Asynchronous Systems

In the context of this paper we only deal with embedded clock I/O architecture such as SerDes. SerDes is an interface IC that converts n-bit parallel data into serial data at the transmitter and converts this serial data back to n-bit parallel data at the receiver. The transmitter side of the SerDes has serializer, de-emphasis, and line drivers. The receiver side has clock and data recovery (CDR), equalizer, and deserializer. This is captured in Figure 2.

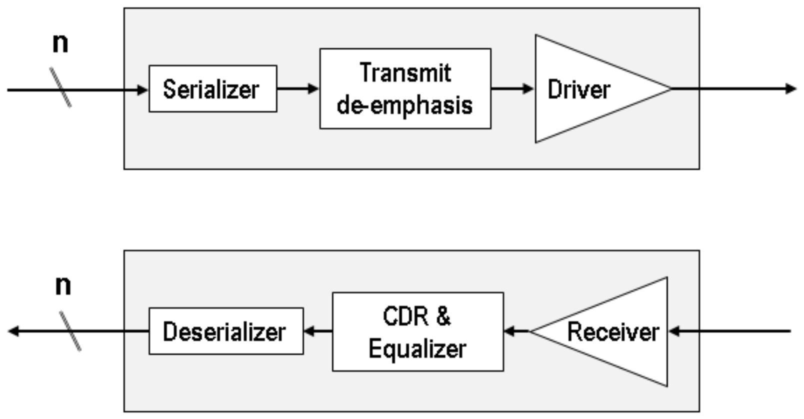


Figure 2. SerDes block diagram.

3.1 Synchronous and Asynchronous Systems

In this architecture of SerDes, there is no explicit clock being transmitted together with the data; the clock information, due to data transitions, is embedded in the data. There are protocols designed for different applications. SONET, USB, Ethernet, and PCIe are some well-known serial interfaces. Each of these serial interfaces can be sorted into one of two groups, synchronous or asynchronous.

As illustrated in Figure 3, a synchronous serial interface has both the transmitter and the receiver share a common reference clock. That is, the normalized reference clock frequency is identical to both the TX and the RX. Over time, the synthesizer output clock frequency might drift due to environment changes such as temperature or voltage, but the frequency seen on the RX side follows that seen on the TX side. The CDR in the RX would only need to track the instantaneous phase variations between the TX and the RX due to noise and jitter.

Asynchronous serial interface, on the other hand, implies that the transmitter and the receiver have different reference clocks which could differ by some small values, usually defined as ppm (parts per million, defined as the difference between the TX and RX normalized frequencies over the normalized frequency). When the TX and the RX reference clocks come from different on-board clock synthesizers, this system is considered to be an asynchronous system (or plesio-chronous system). The RX clock frequency cannot be guaranteed to be the same as the TX clock frequency even over a short period of time. Due to temperature, voltage or process variations between the two clock synthesizers, an on-average ppm offset could exist between the two clock frequencies. The CDR in the RX not only needs to track the instantaneous phase difference, but also needs to compensate for the frequency difference between the two sides.

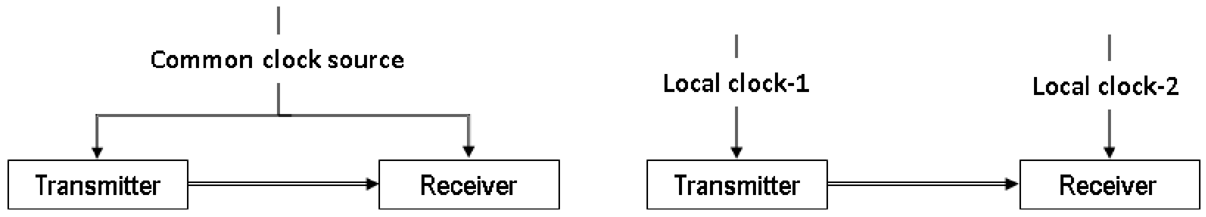


Figure 3. Two clocking architectures.

3.2 Clock Data Recovery

A PLL in SerDes is used to sync the local clock frequency with the reference clock frequency and also to multiply the reference clock frequency up for the actual data rate. In a synchronous system, the receiver PLL and the transmitter PLL gets the same reference clock, while in an asynchronous system, the reference clock frequency to the RX PLL and to the TX PLL could be different. One approach for clock recovery is to have a phase-frequency detector (PFD) on the RX side to directly adjust the PLL frequency. When there is no data or no lock the internal PLL will normally frequency/ phase lock onto the reference clock. Once the SerDes receiver starts to receives, the PLL clock is compared in frequency and phase in a PFD to generate the error signal, which is then used to adjust the PLL in such a way that the incoming data is more or less sampled around the bit center. Figure 4 shows the general concept for timing recovery.

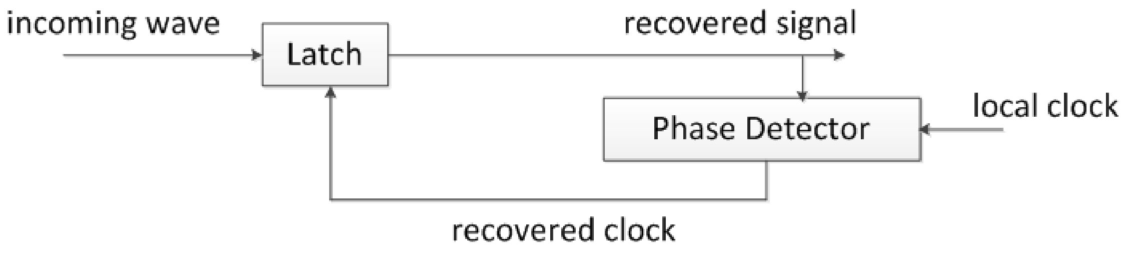


Figure 4. General concept for timing recovery.

Another commonly used approach is to use a phase interpolator (PI) based block to adjust the RX sampling clock phase from a local PLL to track and match the TX frequency and phase. The block diagram for these two approaches are shown in Figure 5. PD is the phase detector and are generally very different for different architectures.

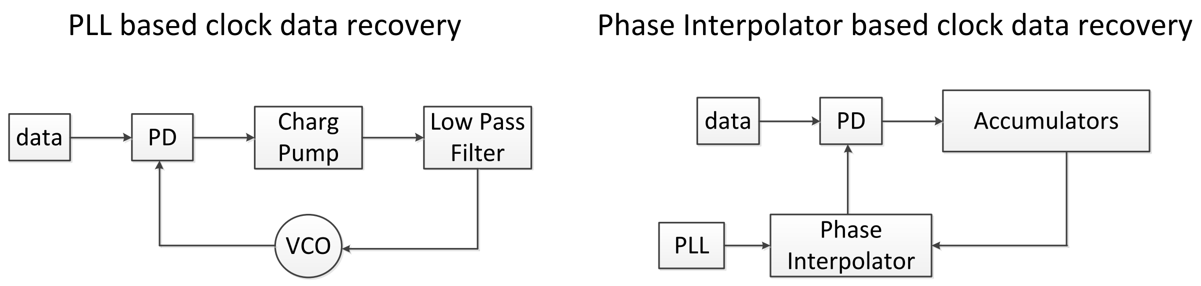

PLL based clock data recovery
data
PD
Charg Pump
Low Pass Filter
VCO
Phase Interpolator based clock data recovery
data
PD
Accumulators
PLL
Phase Interpolator

Figure 5. PLL based versus Phase Interpolator based clock data recovery

The commonly used CDR can be categorized into either a first order or a second order CDR. A first order CDR can track phase offsets within its loop bandwidth and loop gain range, but would result in phase error if a frequency offset exists in the system. The larger the frequency offset is, the larger the phase error becomes. A second order CDR, on the hand, can track the frequency offset without phase errors within its tracking range. In this paper, we will use the PI based clock data recovery as an example to explain the timing recovery behavior in an asynchronous system. The analysis of PLL based timing recovery is very similar.

3.3 System Performance Impact

For an asynchronous system, frequency offset ppm has to be well controlled. When system margin is small, a very small ppm offset needs to be guaranteed and/or a second order CDR has to be used to reduce the margin loss.

A PI-based 2nd-order CDR architecture is typically modeled as shown in Figure 6. Gp is the 1st order path gain and Gf is the 2nd order path gain. PD is the phase detector and PI is the phase interpolator. It is straightforward to analyze the system behavior when we can reasonably linearize the PD. When the CDR loop is in lock, the PD output, the phase error, is expected to be 0 in the ideal case. It is worth mentioning that when Gf is 0, this architecture becomes a 1st order CDR.

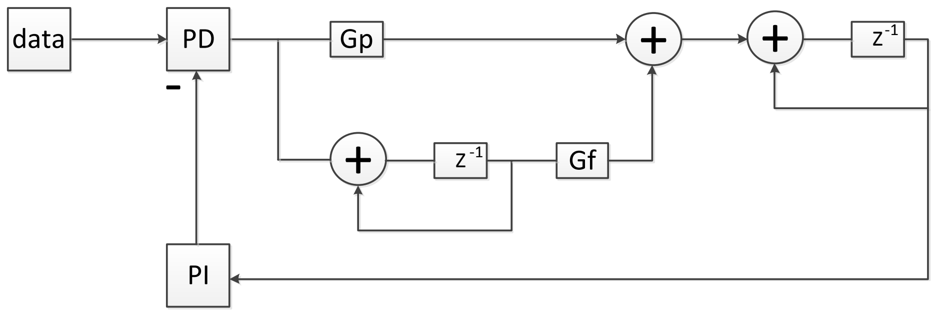


Figure 6. Block diagram of a 2nd order CDR.

With a small frequency offset, the PD output in the steady state does not approach 0 if the CDR loop is only of the 1st order. This phase error is proportional to the ppm offset magnitude and inversely proportional to Gp and Gpi (the PI gain) as shown in Equation (1). However, when the 2nd order CDR is used, the phase error can be reduced to a small number within the PI resolution as in Equation (2). Here, P_e is the phase error and Δf is the frequency ppm offset.

$$P_e^{1st-order} \sim \frac{\Delta f}{G_{pi} \times G_p} \qquad \textbf{Equation (1)}$$

$$P_e^{2nd-order} \sim 0 \qquad \textbf{Equation (2)}$$

In practice, the PD cannot track large frequency offset due to its phase wrapping-around nature, resulting in incorrect phase detection output. The CDR loop will not converge properly even with the 2nd order path. Thus, when the frequency offset is out of the CDR capture range, the CDR can no longer lock to the incoming data and the eye is closed. This behavior can be seen from the time domain simulation in Figure 7 based on the simple architecture model given in Figure 6.

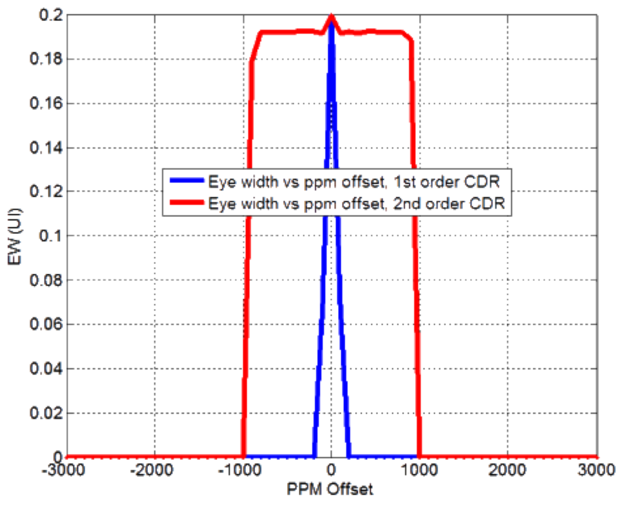


Figure 7. Eye width as a function of ppm offset for the 1st order and the 2nd order CDR.

Another insight we can derive from this simple model is for jitter tolerance. It is well known that the CDR can track low frequency jitter up to its bandwidth. The 2nd order CDR loop can especially help with very low frequency jitter because of the 2nd pole it introduced in its jitter transfer function. In addition, the 1st order CDR jitter tolerance is compromised in the asynchronous system while the 2nd order CDR is barely affected. This can be observed in either the small signal analysis or a simple time domain simulation as shown in Figure 8.

Although a simple architecture model can predict, to a large degree, the behavior of a CDR in the presence of a frequency offset and a sinusoidal jitter, a time domain simulation is desired for higher accuracy and to account for the convergence complexities and interactions among various adaptation loops in the SerDes system. This is where an IBIS-AMI simulation can help.

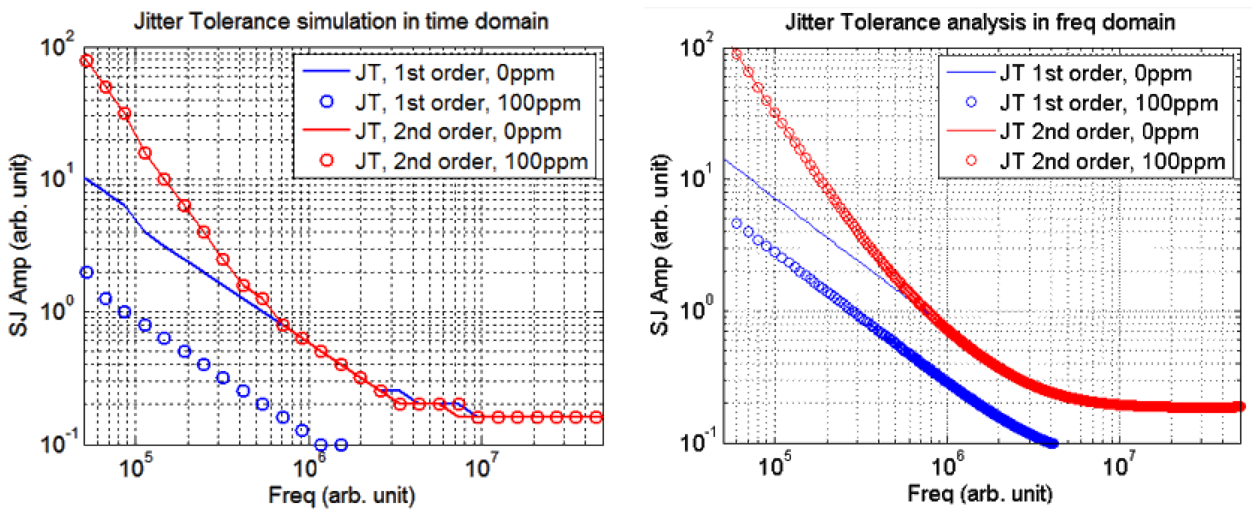


Figure 8. Jitter Tolerance analysis in time domain and frequency domain.

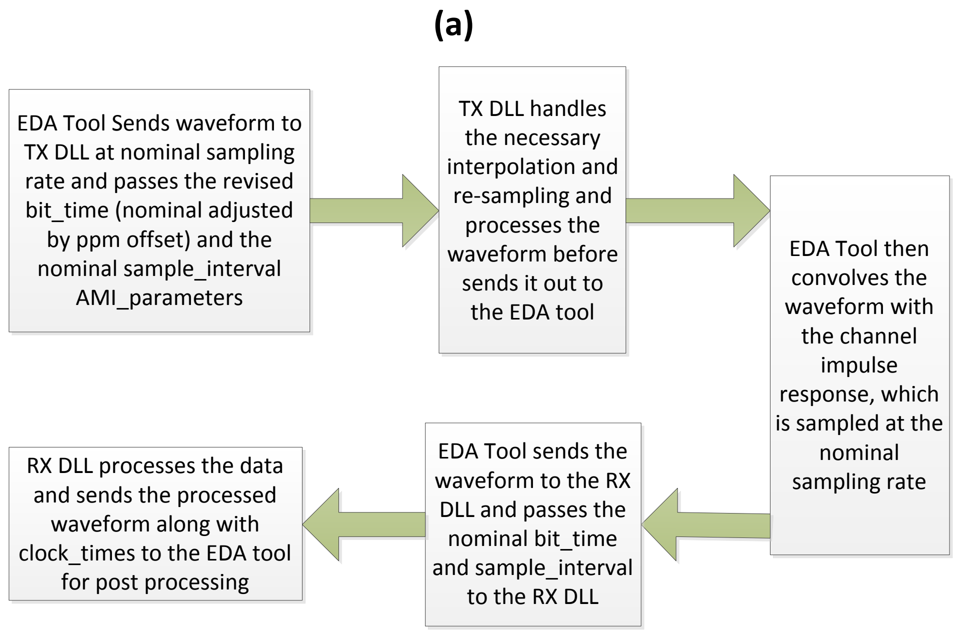
4. IBIS-AMI Simulation for Asynchronous Systems

The existing IBIS-AMI standard supports system level simulations but does not explicitly require its application in an asynchronous system. We can take advantage of the independency between the RX model and the TX model to apply the asynchronous reference clocks through the EDA platform. The reference clock frequency offset can be applied on either the TX or the RX side by varying the AMI parameter input bit_time and sample_interval. If it is applied on the TX side, the bit_time argument value in the AMI_Init call is the actual TX reference clock frequency determined by the TX frequency offset relative to the nominal data rate. In addition, the ideal unit interval (without TX jitter) of the input digital stimulus to the TX DLL is equal to the inverse of the actual TX reference clock frequency. If it is applied on the RX side, the bit_time argument value in the AMI_Init call is the actual RX reference clock frequency determined by the RX frequency offset relative to the nominal data rate. It should be pointed out that in both cases the sample_interval argument value in the AMI_Init call does not need to depend on either actual reference clock frequencies.

If the same sample_interval is used by the TX and the RX, the samples per bit could be different between the TX and the RX due to the small ppm offset and could be non-integer. Although the IBIS-AMI standard requires the DLL to handle the re-sampling, it is possible that many DLLs expect an integer samples per bit and could fail with an exit code 0.

On the other hand, if different sample_intervals are used by the TX and the RX to keep the samples per bit an integer number for both sides, then the EDA tool needs to handle the re-sampling. For example, the EDA tool sends the waveform to the TX at the TX sample rate, receives the TX DLL output and convolves it with the channel impulse response, which is sampled at the same rate as the TX. Then the EDA tool interpolates the convolved output waveform and re-sample it to match the RX side sampling rate before sending the waveform to the RX DLL.

The two approaches are illustrated in Figure 9(a) and 9(b) respectively, assuming offset is added on the TX side. It is not the intention of this paper to endorse or to dictate which approach should be adopted.


(a)
EDA Tool Sends waveform to TX DLL at nominal sampling rate and passes the revised bit_time (nominal adjusted by ppm offset) and the nominal sample_interval AMI_parameters
TX DLL handles the necessary interpolation and re-sampling and processes the waveform before sends it out to the EDA tool
EDA Tool then convolves the waveform with the channel impulse response, which is sampled at the nominal sampling rate
EDA Tool sends the waveform to the RX DLL and passes the nominal bit_time and sample_interval to the RX DLL
RX DLL processes the data and sends the processed waveform along with clock_times to the EDA tool for post processing

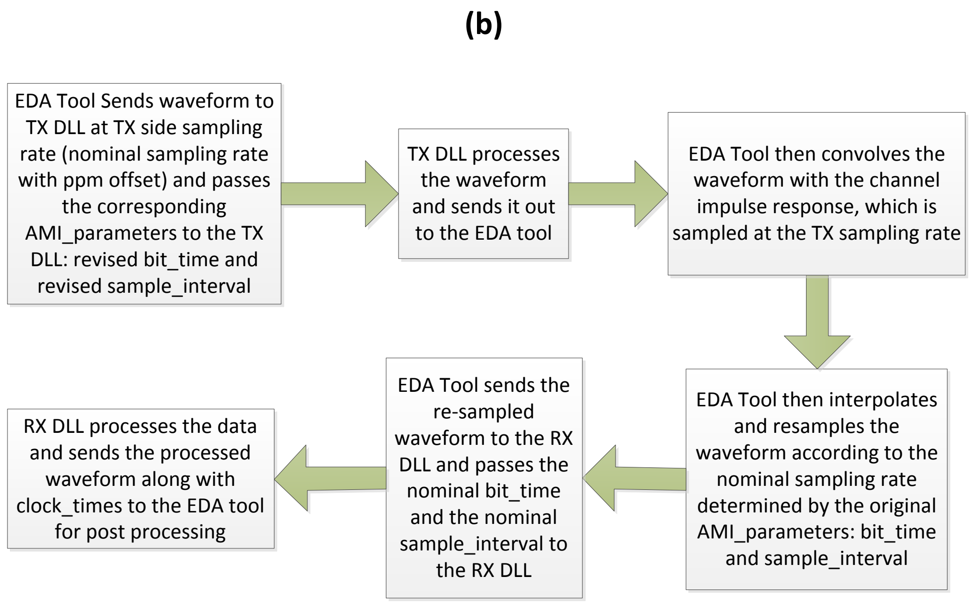

(b)
EDA Tool Sends waveform to TX DLL at TX side sampling rate (nominal sampling rate with ppm offset) and passes the corresponding AMI_parameters to the TX DLL: revised bit_time and revised sample_interval
TX DLL processes the waveform and sends it out to the EDA tool
EDA Tool then convolves the waveform with the channel impulse response, which is sampled at the TX sampling rate
EDA Tool then interpolates and resamples the waveform according to the nominal sampling rate determined by the original AMI_parameters: bit_time and sample_interval
EDA Tool sends the re-sampled waveform to the RX DLL and passes the nominal bit_time and the nominal sample_interval to the RX DLL
RX DLL processes the data and sends the processed waveform along with clock_times to the EDA tool for post processing

Figure 9. (a) Frequency offset added on the TX side and the TX DLL handles the re-sampling. (b) Frequency offset added on the TX side and EDA tools handles the re-sampling.

5. Asynchronous System Simulation and Measurement

Three channels, with high, medium and low losses, are analyzed. Insertion and return losses of these channels are plotted in Figure 10 and Figure 11. Insertions losses of the three channels are 36dB, 30dB and 18dB, respectively.

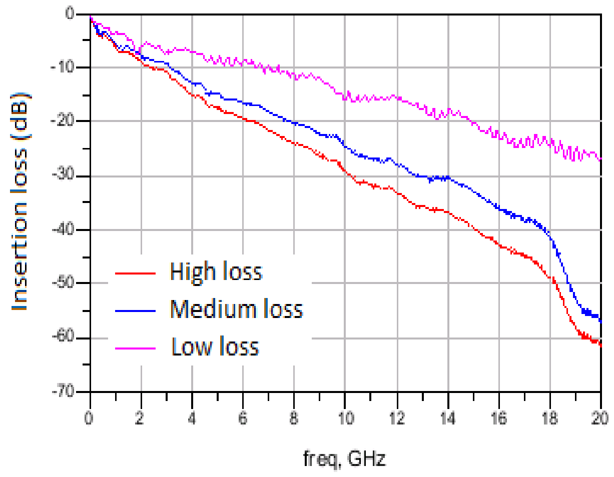


Figure 10. Channel insertion losses.

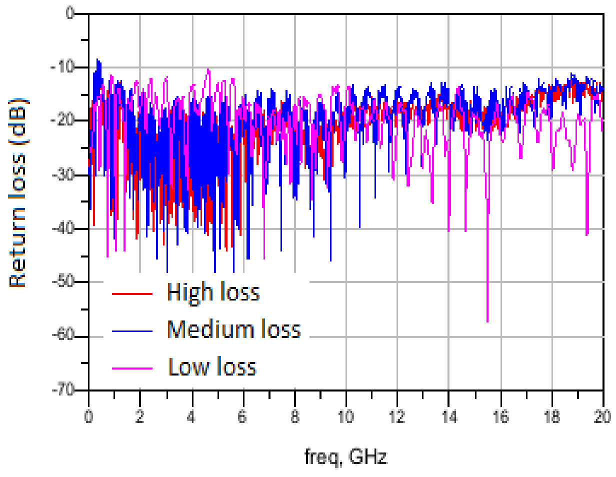


Figure 11. Channel return losses.

The simulation setup is shown in Figure 12. The TX package, the physical channel and the RX package are represented by 4-port S-parameters, which are cascaded to model the link. The nominal data rate is 28Gbps. Each simulation runs one million PRBS23 bits. The frequency ppm offset is set inside the EDA tool, so the interface of the simulation bench looks the same as that for the conventional IBIS-AMI simulations for synchronous links.

The TX model implements a 3-tap FIR for de-emphasis. The RX model implements adaptive CTLE, AGC and DFE. In this study the high and medium loss channels use the same TX equalizer setting and the low loss channel uses a different setting. The default CDR in the RX model is 2nd order. The model has a parameter knob to switch the CDR between the 1st order and the 2nd order. In this example the reference clock offset is applied on the TX side through the EDA. Positive offset means TX side data rate is higher than that on the RX side. All the denoted frequency offsets are measured relative to a reference offset Δf.

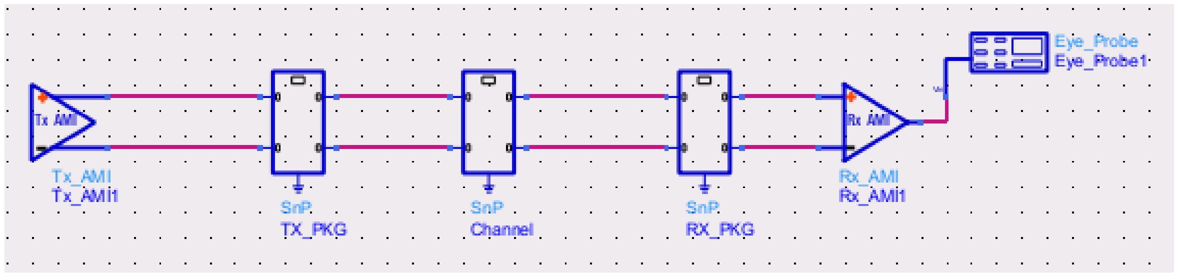


Figure 12. Simulation setup for an asynchronous link.

5.1 High Loss Channel

The CDR phase shifts as a function of time in the high loss channel with frequency offset relative to the nominal data rate at 5 and -5 reference offset Δf are shown in Figure 13. The CDR phase shift is defined as

$$\boldsymbol{phase\ shift(n) = \frac{t_{clk}(n) - t_{clk}(0)}{T_0} - n} \qquad \textbf{Equation (3)}$$

where $t_{clk}(n)$ is the nth clock time returned by the RX AMI_GetWave call and T_0 is the nominal unit interval, which is 1/28G, or 35.71 ps, in this study. Figure 13 shows that the CDR tracks the TX frequency offset at 5 and -5 Δf. As a result, the eyes are open and similar to the eye without frequency offset (in a synchronous channel), as shown with the eye diagrams in Figure 14.

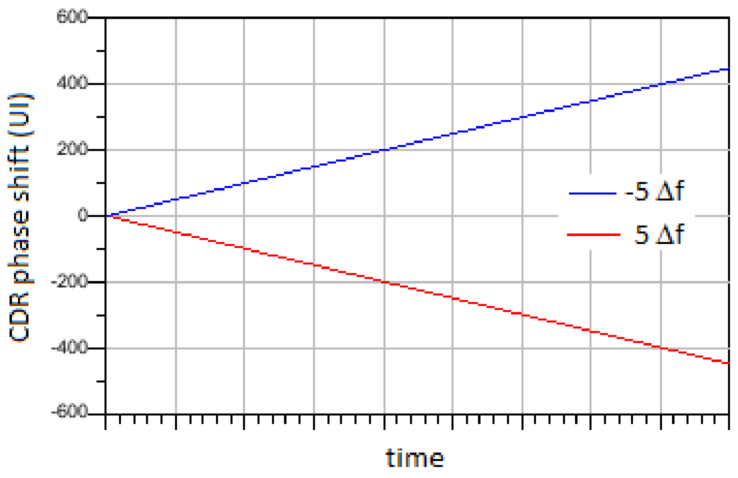


Figure 13. CDR phase shift in the high loss channel.

Figure 14 also shows that at 8.5 Δf frequency offset the eye is closed. The CDR phase shift at this offset, plotted in Figure 15, indicates that the CDR fails to fully track the TX frequency offset and is output of lock with the data, resulting in a closed eye. The red line is the actual CDR output phase shift. The blue line is the hypothetical ideal phase shift if the CDR tracks the TX frequency offset perfectly.

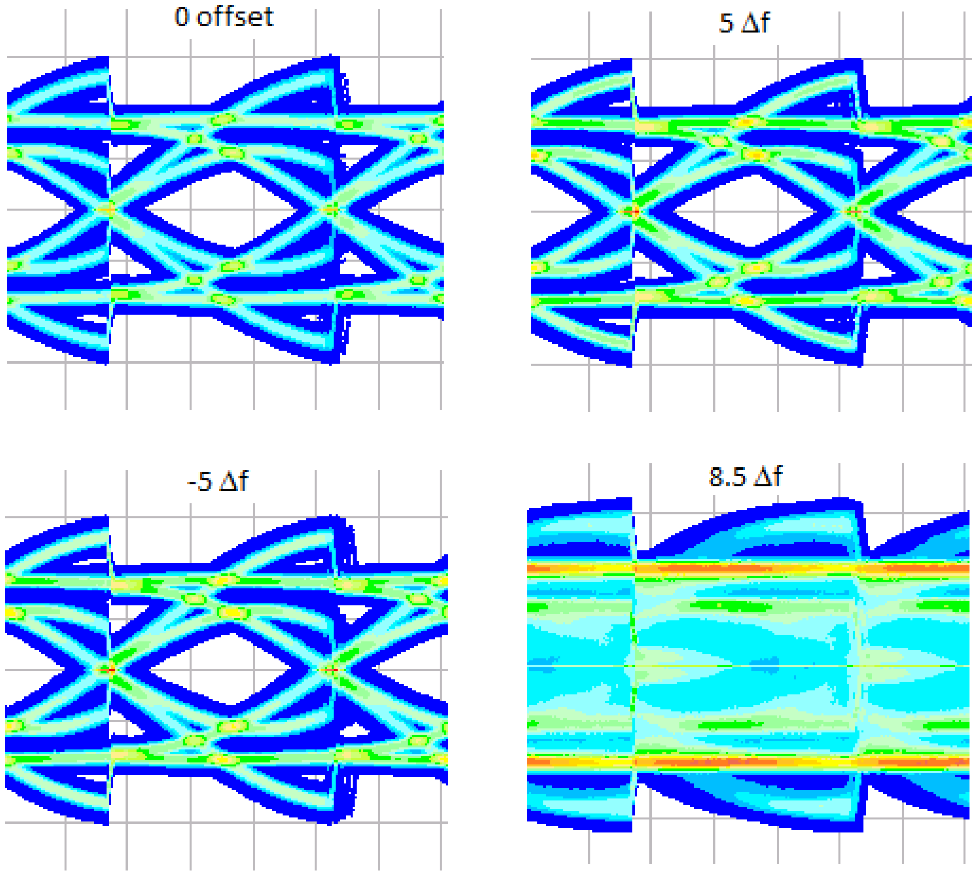


Figure 14. High loss channel eyes with frequency offset at 0, 5, -5 and 8.5 reference offset Δf.

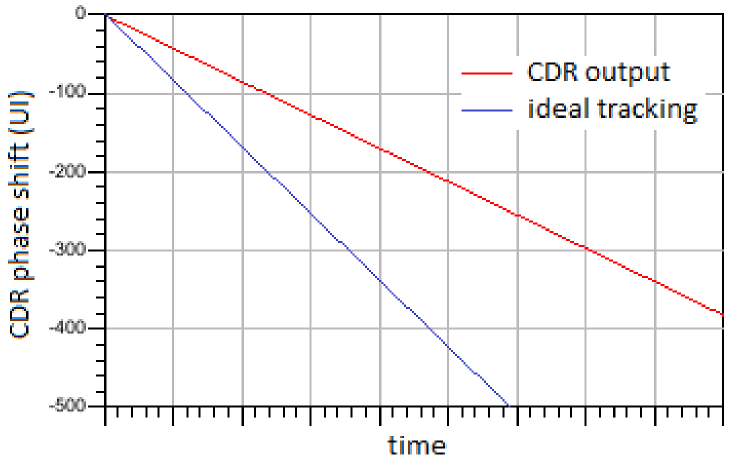


Figure 15. CDR phase shift in the high loss channel with frequency offset at 8.5 reference offset Δf.

Timing bathtub curves and eye contours at 10^{-12} BER with frequency offset at 0, 5 and -5 reference offset Δf are plotted in Figure 16 and Figure 17, respectively. Comparison with the data in the synchronous channel (with 0 offset) shows that in the asynchronous channels (with 5 and -5 Δf offsets) the frequency offset introduces a slight additional timing closure.

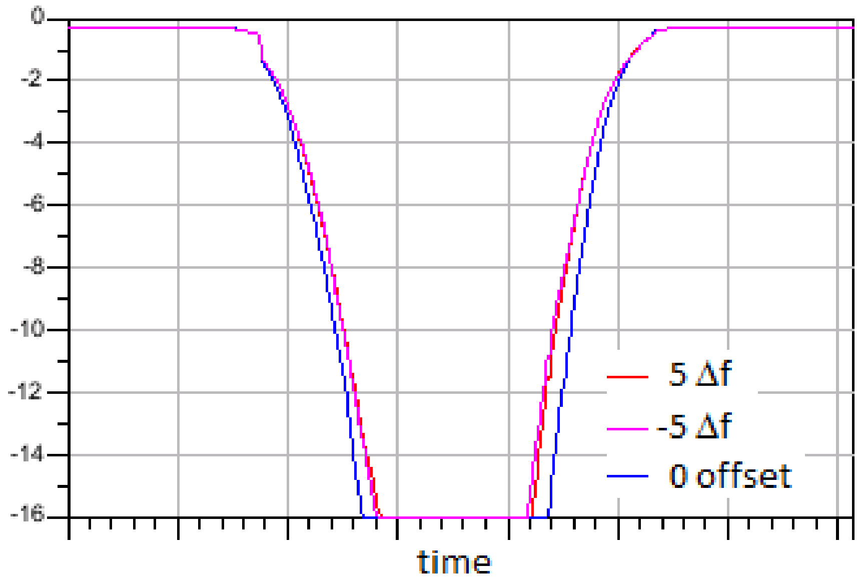


Figure 16. Timing bathtub curves in the high loss channel.

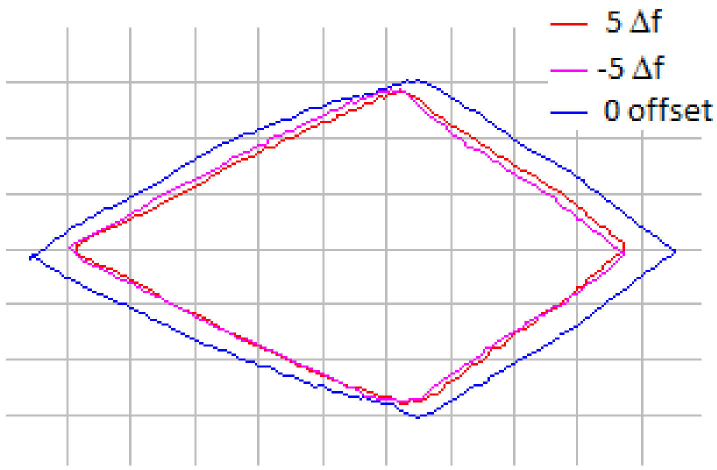


Figure 17. Eye contours at 10^{-12} BER in the high loss channel.

5.2 Medium Loss Channel

Figure 18 shows the eyes for the medium channel with frequency offset at 0, 5, -5 and 10 reference offset Δf. At 10 Δf. The CDR fails to track the frequency offset and the eye is closed.

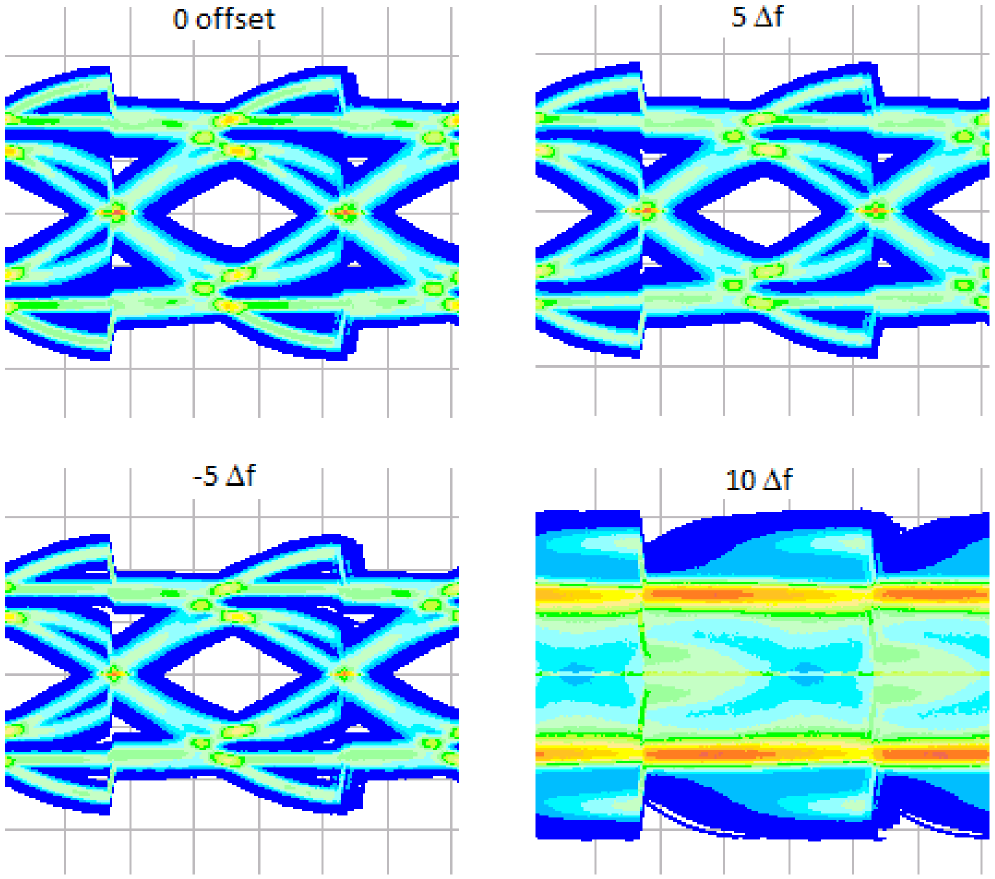


Figure 18. Medium loss channel eyes with different frequency offsets.

Timing bathtub curves and eye contours at 10^{-12} BER with frequency offset at 0, 5 and -5 reference offset Δf are plotted in Figure 19 and Figure 20, respectively. Again, we can see that with 2nd order CDR loop, as long as it is within the frequency capture range, margin loss with frequency offset is small.

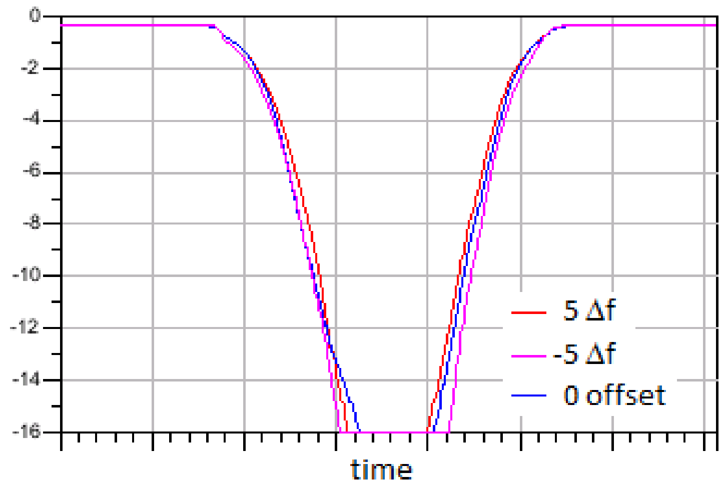


Figure 19. Timing bathtub curves in the medium loss channel.

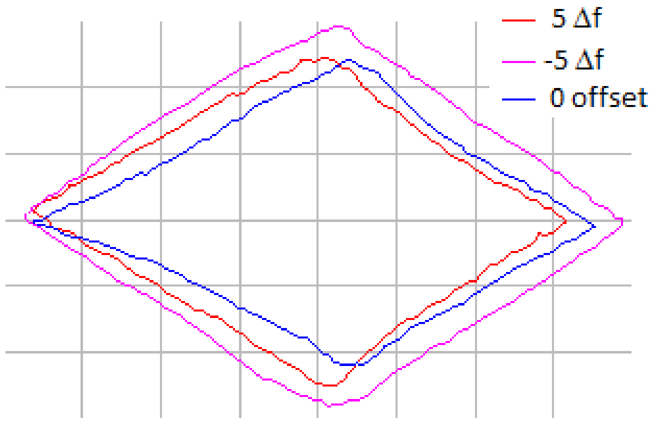


Figure 20. Eye contours at 10^{-12} BER in the medium loss channel.

5.3 Low Loss Channel

Figure 21 shows the eyes in the low channel with frequency offset at 0, 5, -5 and 10 reference offset Δf. At 10 Δf the RX CDR fails to track the TX frequency offset and the eye is closed. Timing bathtub curves and eye contours at 10^{-12} BER with frequency offset at 0, 5 and -5 reference offset Δf are plotted in Figure 22 and Figure 23, respectively. The same conclusion on the margin loss is drawn here. Also worth noting is the very small phase shift with positive and negative ppm offset.

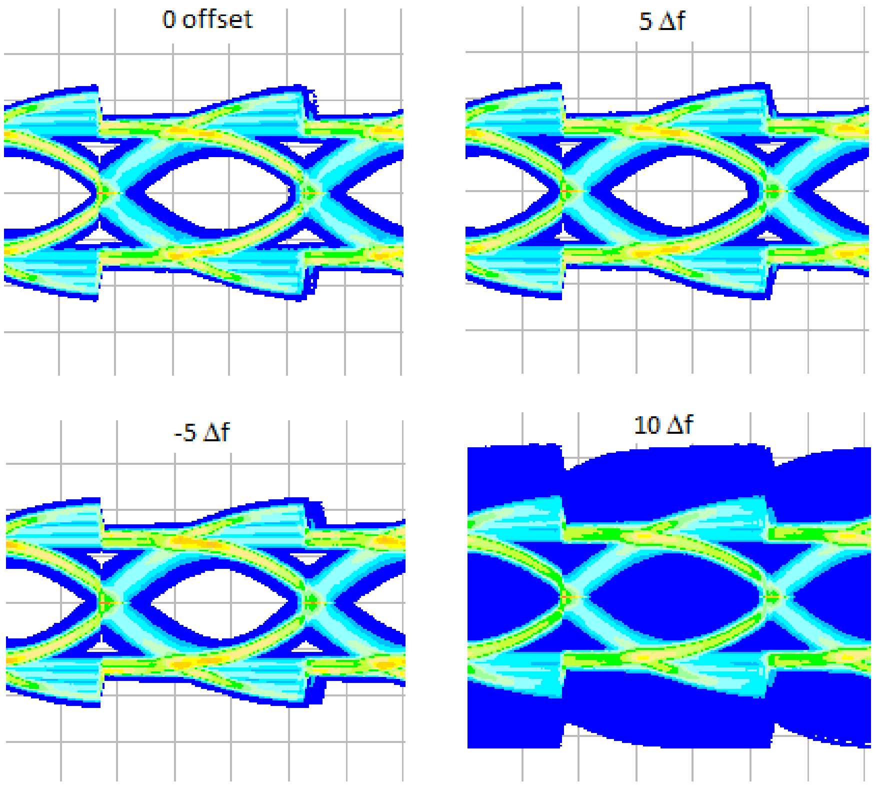


Figure 21. Low loss channel eyes with different frequency offsets.

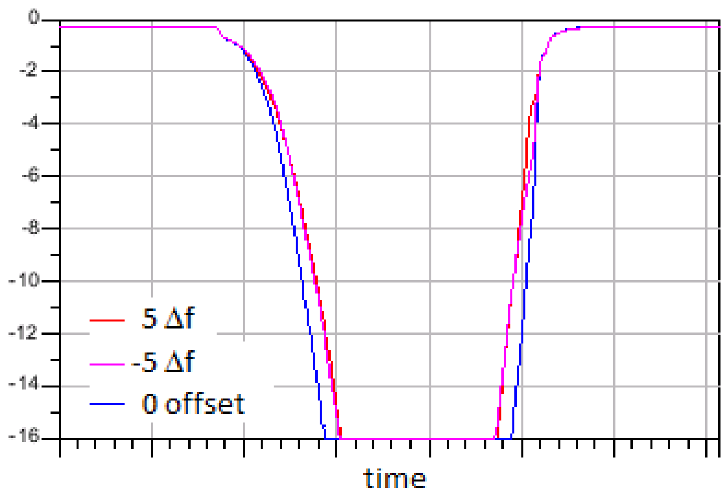


Figure 22. Timing bathtub curves in the low loss channel.

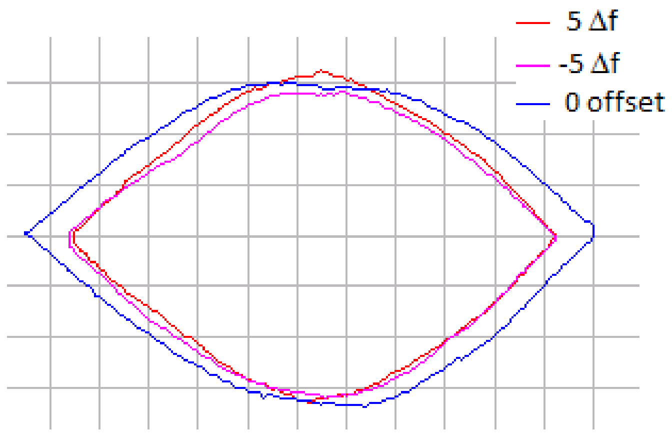


Figure 23. Eye contours at 10^{-12} BER in the low loss channel.

5.4 Frequency Offset Tolerance

Eye widths at 10^{-12} BER vs frequency offset are plotted in Figure 24 for the three channels. It is seen that the CDR is able to track the offset between -11 Δf and 8 Δf. The high loss channel seems better equalized than the medium loss channel with the equalization settings from the TX and the RX side, leading to a slightly larger eye width.

Timing bathtubs and eye contours at 10^{-12} BER between +/-15 Δf frequency offset are shown in Figure 25 and Figure 26. Again, we can see that within the capture range, frequency offset impact to a second order CDR system exists but is relatively small.

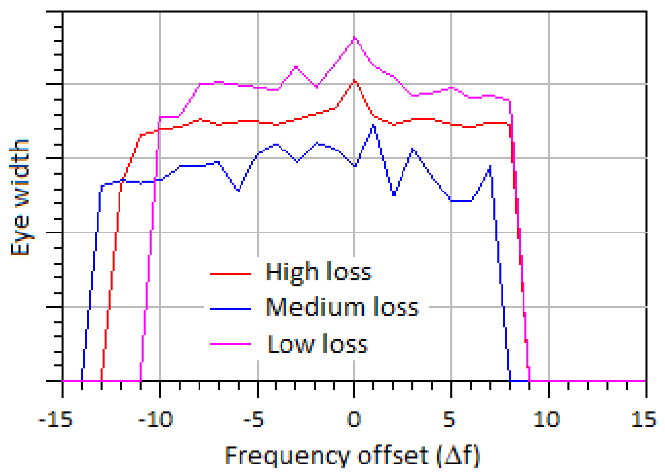


Figure 24. Eye width at 10^{-12} BER vs frequency offset.

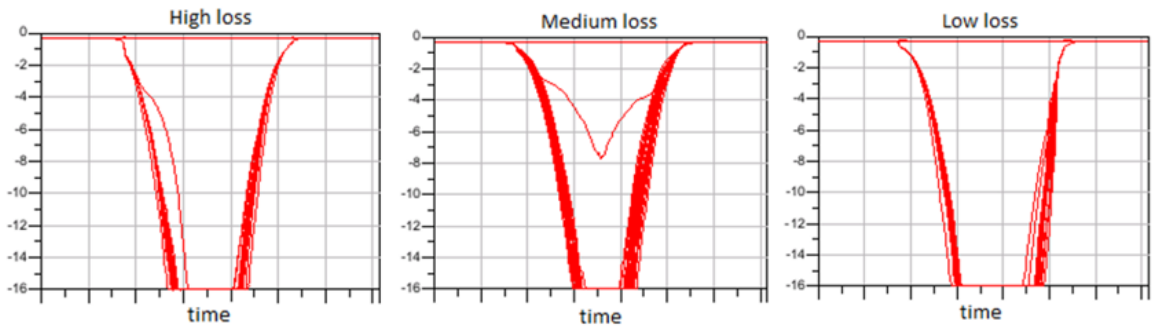


Figure 25. Timing bathtub curves in three channels with different frequency offsets.

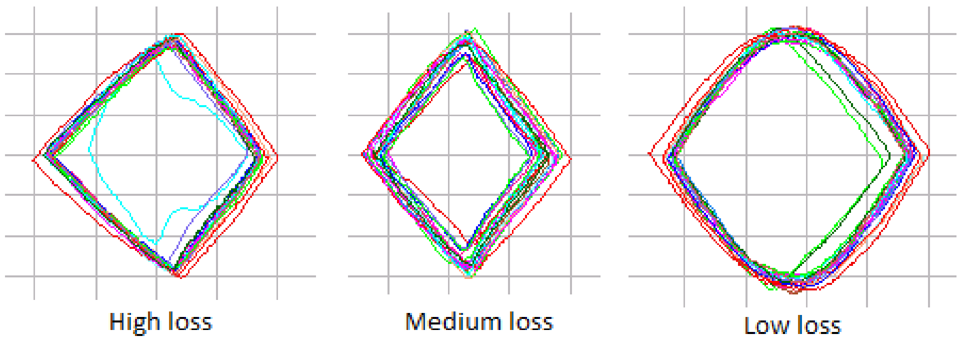


Figure 26. Eye contours at 10^{-12} BER in three channels with frequency offset between -15 and 15.

To compare the frequency offset tolerance between the 1st order CDR and the 2nd order CDR, eye widths at 10^{-12} BER with 1st order CDR vs frequency offset are plotted in Figure 27. The result indicates that,for all three channels, the 1st order CDR can only track the offset between +/- 1.3 Δf. The frequency offset tolerance is much lower in the 1st order CDR than in the 2nd order CDR. Please note that the EW seems flat within the capture range, but it is actually not (as expected in the theoretical analysis in Figure 6). The locking point is severely impacted by the ppm offset (Figure 28), so the effective EW degrades proportionally to the ppm offset.

Timing bathtubs and eye contours at 10^{-12} BER with the 1st order CDR are shown in Figure 26 with frequency offset between -1.5 and 1.5 reference offset Δf and Figure 29 with frequency offset between -1.5 and 1.5 reference offset Δf. Closed eyes at the BER are not shown. As ppm offset becomes more positive, the eye is shifted to the left, resulting less and less horizontal eye margin on the right. Similarly, as ppm offset becomes more negative, the eye is shifted to the right, resulting less and less horizontal eye margin on the left. This is in contrast to a second order CDR system, where eye is not shifted as long as the ppm offset is within the capture range.

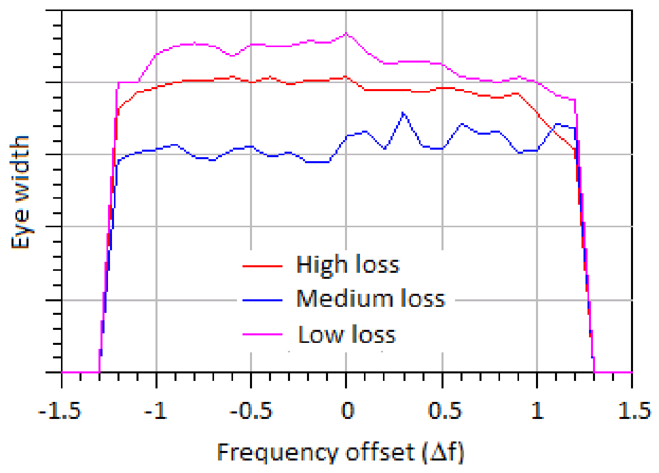


Figure 27. Eye width at 1e-12 BER with 1st order CDR vs frequency offset.

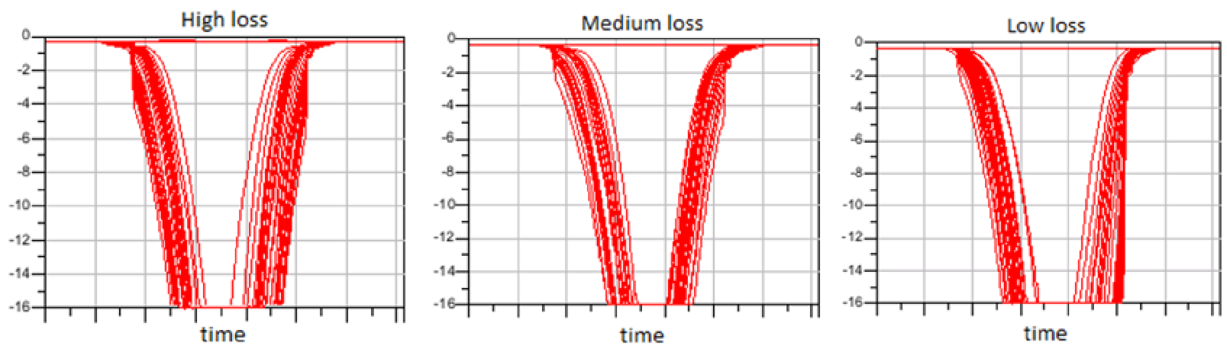


Figure 28. Timing bathtub curves for the three channels with the 1st order CDR.

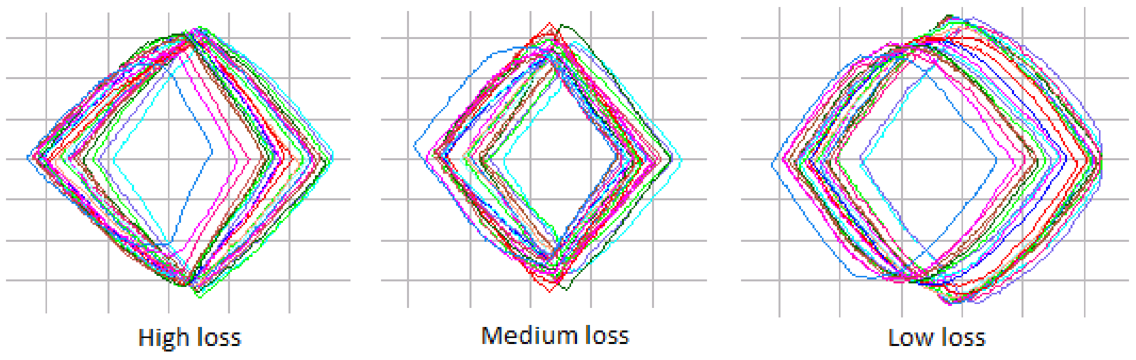


Figure 29. Eye contours at 1e-12 BER for three channels with 1st order CDR.

5.5 Frequency Offset Impact on Jitter Tolerance

To investigate the frequency offset impact on RX jitter tolerance, a sinusoidal jitter (SJ) is injected into the TX data stream. In this paper, all SJ frequencies are measured relative to a reference F_{SJ}, and all SJ amplitudes are measured relative to a reference A_{SJ}.

The CDR phase shift in the high loss channel with frequency offset at 5 reference offset Δf and TX SJ frequency at 10 reference SJ frequency FSJ is plotted in Figure 30. The result shows that the CDR tracked both the TX frequency offset and the TX SJ.

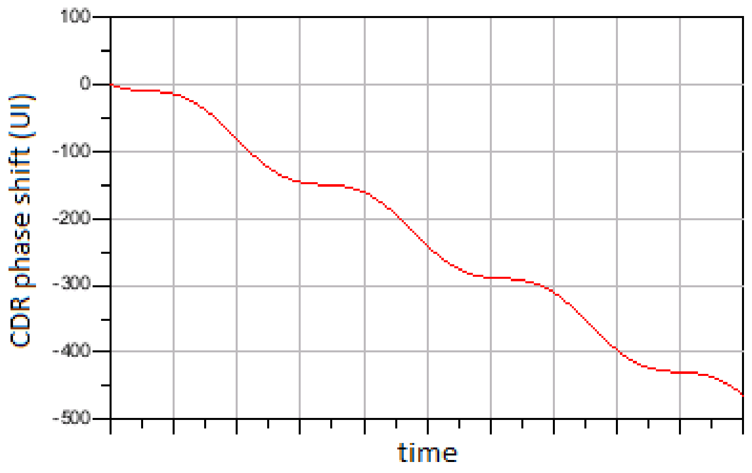


Figure 30. CDR phase shift in the high loss channel with frequency offset.

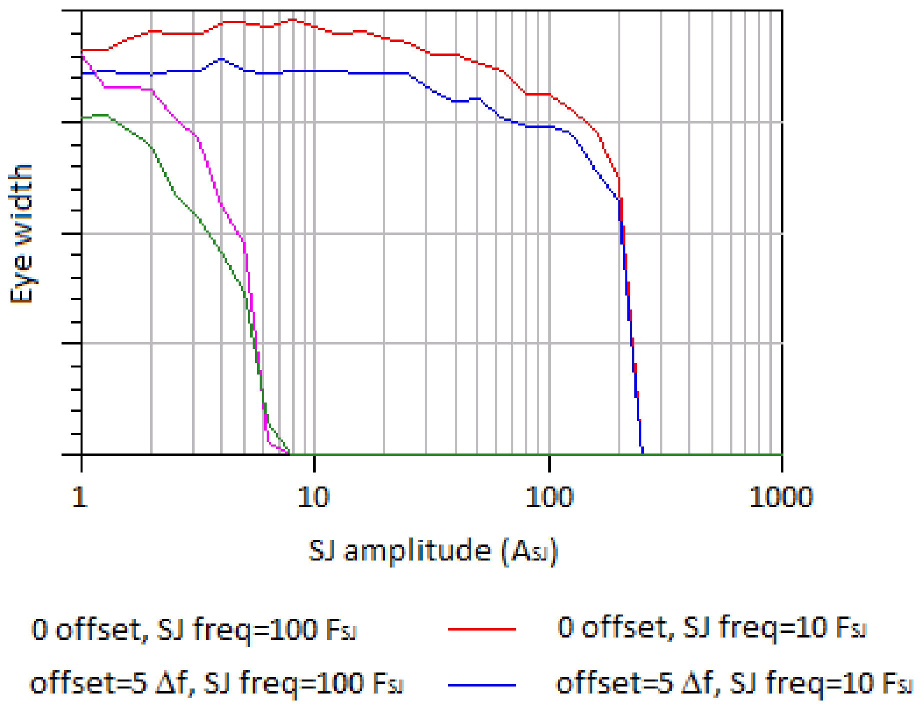


Figure 31. Eye width at 10^{-12} BER vs TX SJ amplitude for the high loss channel.

The TX SJ amplitude is swept between 1 A_{SJ} and 1000 A_{SJ} for the high loss channel with 10 and 100 F_{SJ} TX SJ frequencies and with 0 and 5 Δf frequency offsets. Eye widths at 10^{-12} BER vs TX SJ amplitude are plotted in Figure 31. At both SJ frequencies the frequency offset reduces the eye width. For a certain eye width threshold, the maximum SJ amplitude tolerable by the CDR is around 200 A_{SJ} at 10 F_{SJ} and 5 A_{SJ} at 100 F_{SJ} SJ frequency for both frequency offsets.

Figure 32 shows the CDR jitter tolerance for the high loss channel for the 2nd order CDR and the 1st order CDR. For the 2nd order CDR the frequency offset is found to negligibly impact the jitter tolerance. For the 1st order CDR the impact of the frequency offset is more prominent and occurs in the entire TX SJ frequency range.

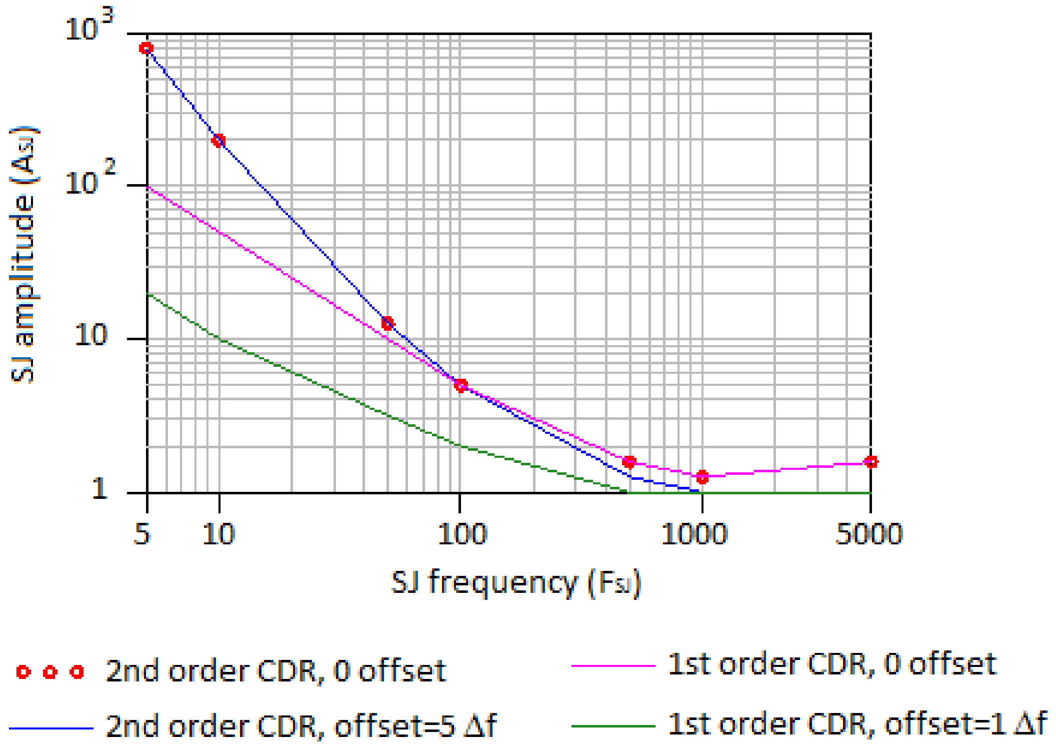


Figure 32. Maximum TX SJ amplitude tolerable by CDR vs SJ frequency for the high loss channel.

6. Conclusions and Future Work

In this paper, we proposed a modified IBIS-AMI model simulation flow for asynchronous link systems. The reference clock frequency offset between TX and RX can be modeled on either the TX or the RX sides. The bit_time input value to the AMI_Init function is the actual reference clock period, and the nominal unit interval of the input digital stimulus to the TX DLL is the inverse of the actual TX reference clock frequency.

The feasibility of the proposed approach is demonstrated by simulating three asynchronous channels with different losses. The results show that the frequency offset can cause timing impairments and reduce link margin. The 2nd order CDR is found to have much better frequency offset tolerance than the 1st order CDR. The frequency offset also impacts the CDR jitter tolerance as predicated.

The capability of simulating system performance in the presence of jitter and frequency offset is valuable for system level timing budgeting. Doing so we are closer to more realistic results than simply treating the link as synchronous. One approximation we did make is by assuming the channel response changes negligibly with ppm offset, which is reasonable given the small ppm offset in typical systems.

The principles of the proposed simulation approach can be extended to the case of spread spectrum clocking (SSC) when the signal frequency is modulated by a low frequency periodic signal to reduce the electromagnetic interference (EMI). SSC is permitted in specifications such as PCIe and CCIX. The complication is that the frequency is no longer constant but changes periodically. As a result, the bit_time (and maybe sample_interval) needs to be updated periodically by the EDA tool. The DLL needs to be able to handle this dynamic change. This would be a good topic for future study.

References

[1] IBIS Specifications 5.0

[2] IEEE P802.3bj™/D3.2, “Draft Standard for Ethernet Amendment 2: Physical Layer Specifications and Management Parameters for 100 Gb/s Operation Over Backplanes and Copper Cables

[3] David Robert Stauffer, Jeanne Trinko Mechler, Michael A. Sorna, Kent Dramstad, Clarence Rosser Ogilvie, Amanullah Mohammad, James Donald Rockrohr, “High Speed Serdes Devices and Application”, Springer Science & Business Media, 2008

[4] Borivoje Nikolić, “Advanced Digital Integrated Circuits”, http://bwrcs.eecs.berkeley.edu/Classes/icdesign/ee241_s04/lectures/Lecture21-Timing.pdf

[5] Ming-ta Hsieh and Gerald E. Sobelman, “Architectures for Multi-Gigabit Wire-Linked Clock and Data Recovery”, IEEE circuits and systems magazine, 4th quarter 2008