DesignCon 2021

Hidden/ Secrets of IBIS Sampling Specifications

Hansel Desmond Dsilva, Achronix Semiconductor
hanseldsilva@achronix.com

Adam Gregory, Samtec
Adam.Gregory@samtec.com

Todd Bermensolo, Keysight
todd.bermensolo@keysight.com

Michael Mirmak, IBIS enthusiast
michael.mirmak@ieee.org

Abstract

The I/O Buffer Information Specification-Algorithmic Modeling Interface (IBIS-AMI) enables sharing of a model, which encompasses the complexity of the transmitter and/or receiver blocks. The IBIS-AMI model outputs an equalized waveform along with sampling information to the electronic design automation (EDA) tool. This paper gives an overview of receiver sampling assumptions in the main IBIS-AMI functions, AMI_Init and AMI_GetWave, along with insights into IBIS reserved parameters usage. Results from seven EDA tools, modelling different sampling mechanisms, show the importance of accurate representation of sampling information when modelling through IBIS. This is an attempt to make model developers and model users aware of the nuances of sampling under IBIS when running channel simulations.

Author (s) biography

Hansel Desmond Dsilva is a Staff Signal Integrity Engineer at Achronix Semiconductor Corporation. He received a Master of Science degree (with thesis) in Electrical Engineering from San Diego State University in 2015 and a Bachelor of Engineering degree in Electronics and Telecommunication Engineering from Don Bosco Institute of Technology, Mumbai (Bombay) University in 2013. He believes in innovating through collaboration and never shies from listening to another's thought process in challenging his own.

Adam Gregory is a Signal Integrity Engineer at Samtec. He is involved in modeling and analysis of high-speed differential signaling channels. He received a BSEE and MSEE at the University of South Carolina.

Todd Bermensolo is an Application Engineer at Keysight Technologies, in their Customer Success Acceleration service team. Prior experience at Intel Corporation in the Enterprise Platform Signal Integrity team. He received his BS degree in EE from University of Idaho in 1998 and his MS degree in EE from University of Illinois in 2005.

Michael Mirmak is a Platform Applications Engineering manager with Intel's Data Center Group, supporting signal integrity (SI) modeling and analysis. He has been involved with SI since 1996. He is a past chair of the IBIS Open Forum, the organization that manages the I/O Buffer Information Specification and the Touchstone specification.

Background

Today's high-speed serial-differential (SerDes) interface design involves sharing of an I/O Buffer Information Specification-Algorithmic (IBIS-AMI) model of the transmitter and receiver in determining channel reach. The different blocks of the transmitter and receiver can include Feed-Forward Equalizer (FFE), Automatic Gain Control (AGC), Continuous Time Linear Equalization (CTLE), Decision Feedback Equalizer (DFE) and Clock Data Recovery (CDR) circuits. The Clock Data Recovery (CDR) circuit block is of particular importance given it determines how the waveform is sampled, which in turn affects the configuration of the various equalization coefficients of the different blocks.

An IBIS-AMI model outputs an equalized waveform, in some cases along with sampling information, to the electronic design automation (EDA) tool. The sampling information provided by the IBIS-AMI model to the EDA tool is critical in determining the margin about the sampling point (for example, eye height top, eye height bottom, eye width left and eye width right) and therefore assessing the performance of the devices in the system design.

The IBIS-AMI specification defines several simulation flows and their modeling functions, including AMI_Init (for statistical simulations) and AMI_GetWave (for time-domain or bit-by-bit simulations). These functions define the equalization, if any, the transmitters and receivers implement, operating on impulse responses or voltage-versus time waveforms respectively, plus parameters passed between the EDA tool and the model. The information passed between the receiver IBIS-AMI model and the EDA tool for the AMI_GetWave flow includes clocking information in the form of "clock_ticks", which helps to define how the model samples the incoming waveform.

However, the sampling mechanism and exchange of data about it between the tool and model are not explicitly defined for statistical models and the AMI_Init flow in IBIS. As a result, performance calculations from statistical IBIS simulations must rely on sampling assumptions made by the EDA tool instead of explicit data from the model. Recognizing this gap in IBIS, the IBIS Open Forum added a new AMI reserved parameter called Rx_Decision_Time in IBIS 7.1 (still a draft as of this writing). This paper compares and contrasts the behavior of the AMI_Init and AMI_GetWave flows in the IBIS specification in terms of sampling, along with providing insight into reserved parameters usage for these flows.

I. Problem statement

The IBIS specification defines the input and output interface to any given model. The EDA tool is responsible to use the output of the model in calculating margin information such as eye height, eye width, bit error rate (BER), etc. and sharing this with the user.

This work presents results of a comparative study on the results of seven EDA tools in running channel simulation using IBIS-AMI models for the statistical and bit-by-bit flow. This work revisits that presented in [1] by fixing the input waveform along with sampling

information to the EDA tool in noting the eye plotting & margining capability through the eye contour. The results below compare seven EDA tools supporting IBIS-AMI, where channel simulations are performed for different channel losses and different sampling mechanisms. To demonstrate sampling behaviors and eye margin performance separate from channel characterization, a methodology is shown, developed into an IBIS-AMI model, which bypasses the impulse_matrix part of the IBIS flow using a comma-separated value (CSV) file of the waveform. This ensures the same input waveforms are applied to the IBIS-AMI models across the different EDA tools, thus helping to focus on the eye calculations of the EDA tools. The tool-generated eyes are compared against an expected reference which is generated using code shared in Appendix A, to show the importance of the receiver model passing explicit sampling information along with the waveform data.

The results from this work will bring attention to changes needed in serial-differential interface analysis as understood by the industry, whether these comprise changes to EDA tool algorithms, IBIS, or interface electrical specifications.

II. Input/ Output Buffer Information Specification (IBIS)

IBIS (Input/Output Buffer Information Specification) is a method of providing the input/output device characteristics buffer through behavioral data without disclosing any circuit or process information. It can be thought of as a behavioral modeling specification suitable for use in transmission line-based simulations of digital systems and applicable to most digital components. There exists an IBIS Open forum which is the industry organization responsible for the management of the IBIS specifications and standards including IBIS, IBIS-AMI, IBIS-ISS, ICM, and Touchstone.

With version 5.0 of IBIS, an algorithmic modeling component was introduced. This enables digital signal processing through executable models and additional data exchange between the model and EDA tool as part of the flow. The analog behavior is captured through traditional IBIS tables while the digital signal processing blocks are part of the Algorithmic Modeling Interface (AMI). In this way, IBIS-AMI enables a means to more accurately model and simulate high-speed interface performance.

IBIS defines separate statistical and bit-by-bit flows in running channel simulations. The statistical flow involves use of an impulse response while the bit-by-bit flow involves use of wave. Table 1 compares the statistical and bit-by-bit flow.

Table 1. Overview of the statistical and bit-by-bit flow.

	Statistical Flow	Bit-by-bit Flow
Inputs	1] Analog channel impulse response 2] Algorithmic Models (AMI_Init)	1] Channel and buffer impulse response 2] User-defined input stimulus 3] Algorithmic Models (AMI_GetWave)

Analysis Method	Convolution Engine	Waveform Processing & Convolution
Outputs	1] Statistical eye diagram 2] Eye height & width measurements 3] Eye contour 4] Equalized & unequalized response	1] Bit-pattern eye diagram 2] Eye height & width measurements 3] Eye contour 4] Equalized & unequalized response

Figure 1 gives an overview of the statistical flow, which calls out an impulse response to be passed by the EDA tool to the model. The EDA tool generates the channel response in combination with the transmitter and receiver analog portion part of the IBIS models, which is in turn given as an input to the transmitter AMI. The output of the transmitter AMI (TX ir_out) is given as an input to the receiver AMI (RX ir_in). The receiver AMI outputs an impulse response (RX ir_out). In IBIS 7.1 (see below), sampling information using Rx_Decision_Time will be provided to the EDA tool. The EDA tool uses the impulse response along with Rx_Decision_Time in plotting the eye and reporting the margins.

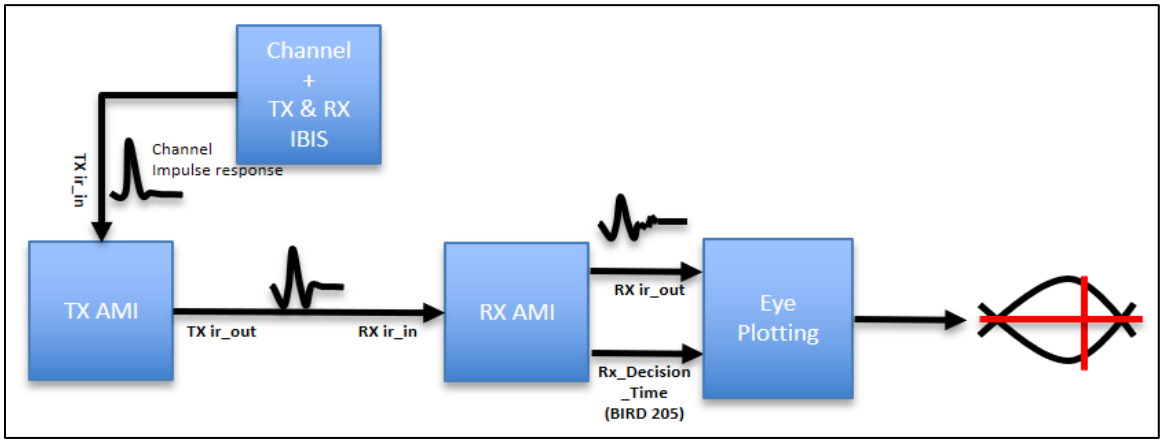

Figure 1. Statistical flow.

Figure 2 shows the bit-by-bit flow, which calls out a wave to be passed by the EDA tool to the model. The EDA tool generates a bit-pattern stimulus which is given to the transmitter AMI. The EDA tool convolves the transmitter AMI outputs a wave (TX wave_out) with the channel response along with the transmitter and receiver analog portion part of the IBIS in generating the input wave for the receiver AMI (RX wave_in). The receiver AMI outputs a wave (RX wave_out) along with sampling information using clock_times to the EDA tool. The EDA tool uses the wave along with clock_times in plotting the eye and reporting the margin. It is important to note that there are a number of combinations of the bit-by-bit flow [2].

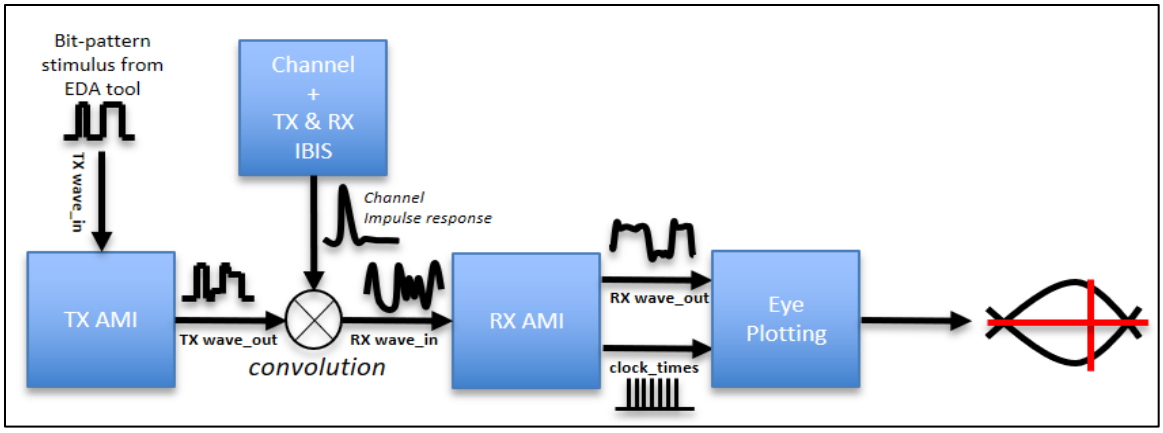

Figure 2. Bit-by-bit flow.

II. Importance of sampling for eye margining

Sampling information is one aspect that many overlook when running channel simulation using IBIS-AMI models. Given the same waveform, one may arrive at different margin value depending on the sampling. Figure 3 shows the impact of sampling on BER. From this, one can see that optimal sampling may lead to optimistic margin, which may not be representative of the real hardware. Ideally, the EDA tool would use the sampling information given by the receiver IBIS-AMI model, which in turn would ideally represent the actual receiver hardware.

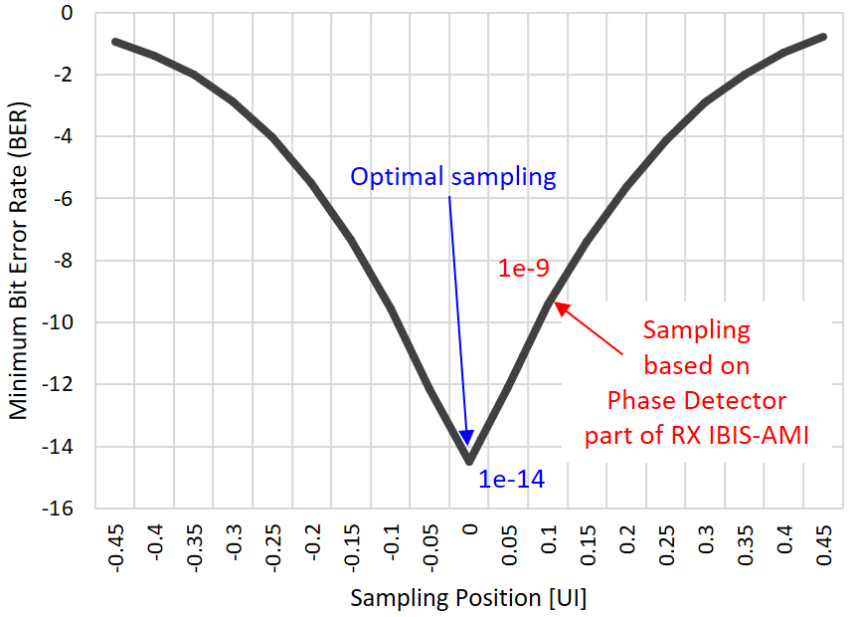

Figure 3. Dependency of minimum BER on sampling.

III. Eye generation using two different phase detectors

Eye generation using the given waveform and sampling information from the models is an important part of the channel simulation. Under IBIS-AMI, the receiver model will output an equalized waveform, in some cases along with sampling information. The actual eye diagram will be generated by the EDA tool itself using this information.

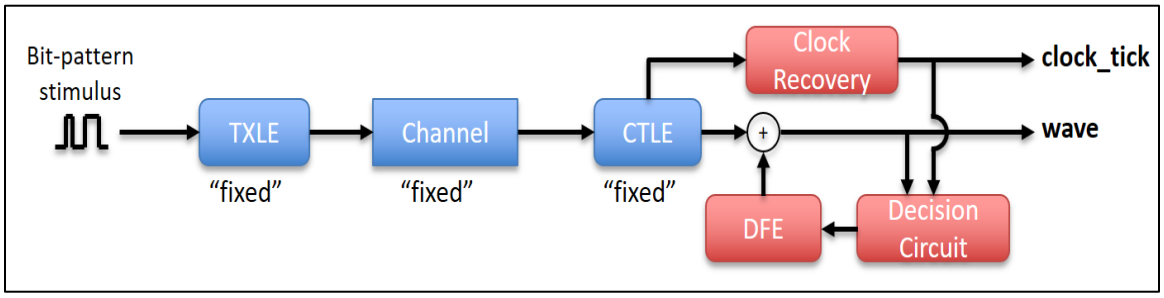

Figure 3. Block diagram overview of the setup.

Figure 3 shows a block diagram of an example system running at a speed of 32 Gbps, using NRZ signaling and modeled using 32 samples per UI. The transmitter linear equalizer (TXLE), channel, and receiver continuous time linear equalizer (CTLE) are fixed. For the purposes of comparison, the receiver clock recovery circuits along with the decision circuit and zero-forcing DFE are varied by using two phase detector (PD) types:

1. Mueller-Muller (MM)
2. Modified Mueller-Muller (Mod-MM)

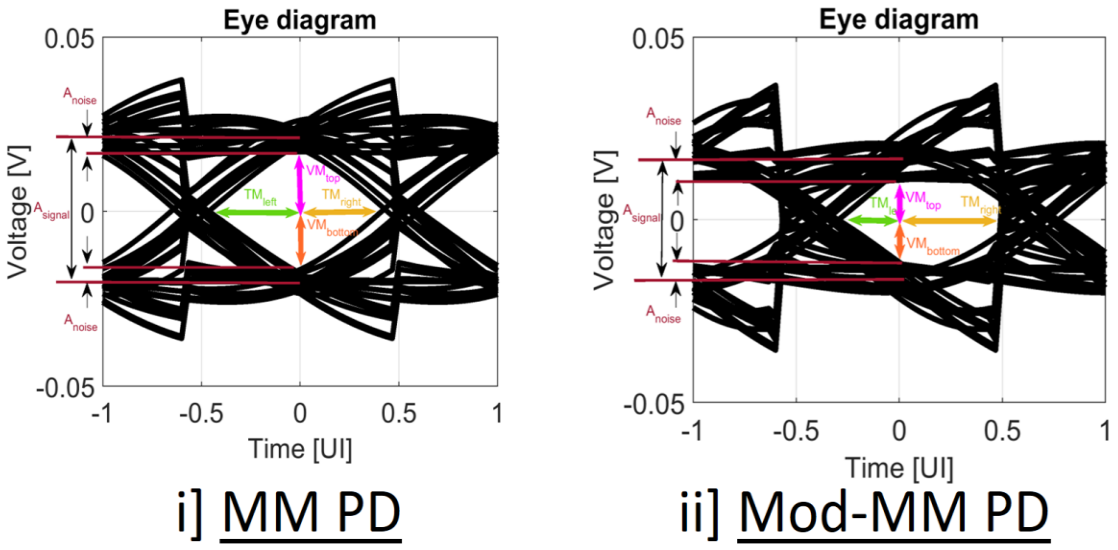

Figure 3. Eye diagram when using MM PD and Mod-MM PD.

Eye height and eye width are metrics commonly used in judging the margin with regards to the eye opening minimum requirement defined for the interface. The voltage margin and timing margin take into account the sampling and may be defined against the outer edges of the eyes (top and bottom for voltage, left and right for timing). The goodness of sampling can be deduced through the values of voltage margin top, voltage margin bottom, timing margin left and timing margin right. Figure 4 shows the eye diagrams when using the MM PD and Mod-MM PD.

Table 2. Margin when using MM PD and Mod-MM PD.

32 Gbps NRZ	MM PD	Mod-MM PD
EH [mV]	33.76	22.56
EW [UI]	0.900	0.757
VM_{top} [mV]	17.16	10.91
VM_{bottom} [mV]	16.60	11.65
TM_{left} [UI]	0.46667	0.26667
TM_{right} [UI]	0.43333	0.49000

Table 2 shows the margin when using the MM PD and Mod-MM PD. While the eye height and eye width show a minor difference in margin when using MM PD and Mod-MM PD, the voltage and timing margin show that the MM PD gives a well-balanced eye.

The eye density along with signal-to-noise ratio (SNR) are good metrics for checking the amount of noise due to residual inter-symbol interference (ISI) and jitter. The eye may be wide open, yet the noise may have a significant impact on the performance of the receiver. Figure 5 shows the eye density along with calculated SNR when using the MM PD and Mod-MM PD. One may observe increased noise and corresponding lower SNR when using the Mod-MM. Thus a wide-open eye may be misleading.

In conclusion, the MM PD provides a much more symmetric eye along with better SNR when compared to the Mod-MM PD.

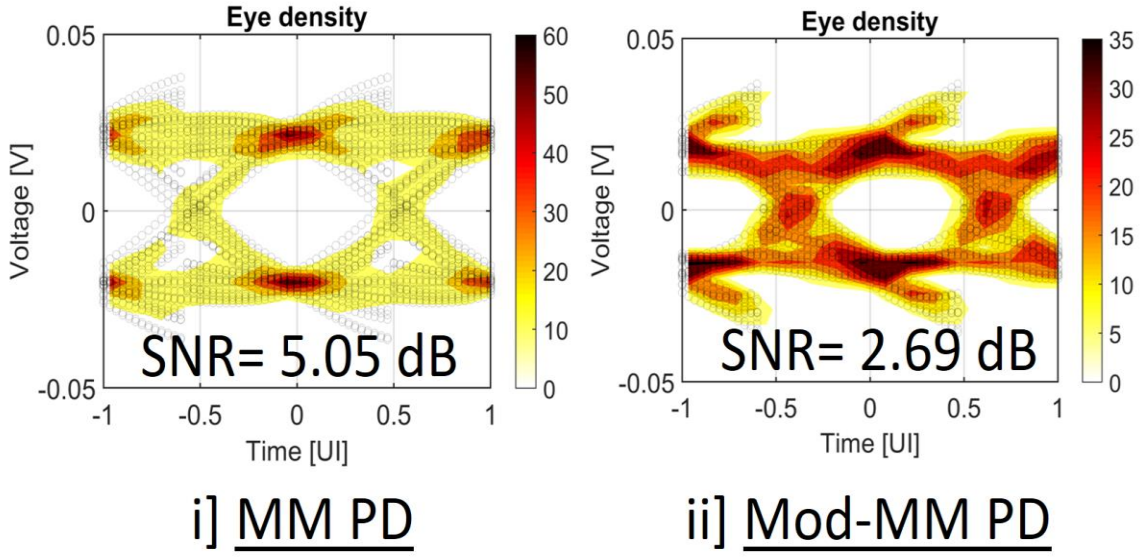

Figure 4. Eye density when using MM PD and Mod-MM PD.

IV. Sampling controls in the IBIS specification

The IBIS specification calls out clock_times and, in IBIS 7.1, Rx_Decision_Time, to enable sampling information to provided by the receiver IBIS-AMI model to the EDA tool when in the bit-by-bit and statistical flows, respectively.

In case the receiver model does not return sampling information to the EDA tool, the IBIS specification calls out the following 'RECEIVER RECOVERED CLOCK RESERVED PARAMETERS'.

- Rx_Clock_PDF
- Rx_Clock_Recovery_Mean
- Rx_Clock_Recovery_Rj
- Rx_Clock_Revovery_Dj
- Rx_Clock_Recovery_Sj
- Rx_Clock_Recovery_DCD

Table 3 gives an overview on the sampling controls in IBIS with regards to the statistical and bit-by-bit flows. For further details, recommended is the IBIS specification [3].

Table 3. Overview of sampling controls in IBIS.

Statistical flow	Bit-by-bit flow	Comment
Rx_Decision_Time - BIRD 205; status: Accepted	clock_times	Tells the EDA tool about the sampling position
Rx_Clock_PDF*	Rx_Clock_PDF*	Tells the EDA tool about the probability density function of the recovered clock (PDF)
Rx_Clock_Recovery_Mean*	Rx_Clock_Recovery_Mean*	Tells the EDA tool about the static offset between the recovered clock and the point half way between the PDF medians of consecutive edge threshold crossing times
Rx_Clock_Recovery_Rj*	Rx_Clock_Recovery_Rj*	Tells the EDA tool about the standard deviation of a Gaussian phase noise exhibited by the recovered clock
Rx_Clock_Recovery_Dj*	Rx_Clock_Recovery_Dj*	Tells the EDA tool about the worst-case half the peak to peak variation of the recovered clock
Rx_Clock_Recovery_Sj*	Rx_Clock_Recovery_Sj*	Tells the EDA tool about the half of the peak to peak variation of a sinusoidal phase noise exhibited by the recovered clock
Rx_Clock_Recovery_DCD*	Rx_Clock_Recovery_DCD*	Tells the EDA tool about the half of the peak to peak variation of a clock duty cycle distortion exhibited by the recovered clock

Note (*). In case the receiver model does not return sampling information to the EDA tool, the IBIS specification calls out these 'RECEIVER RECOVERED CLOCK RESERVED PARAMETERS'.

V. Channel simulation across seven different EDA tools

Running channel simulation across different EDA tools has shown large variation in results even when the same channel and transmitter and receiver IBIS-AMI models are used [1]. Once can observe variation in the channel response generated by the EDA tool and variation in operation of the IBIS-AMI model when performing equalization. This makes debugging difficult across different EDA tools.

For running channel simulation across different EDA tools, a receiver IBIS-AMI model was developed that bypasses the EDA input with that from a common separated value (CSV) file.

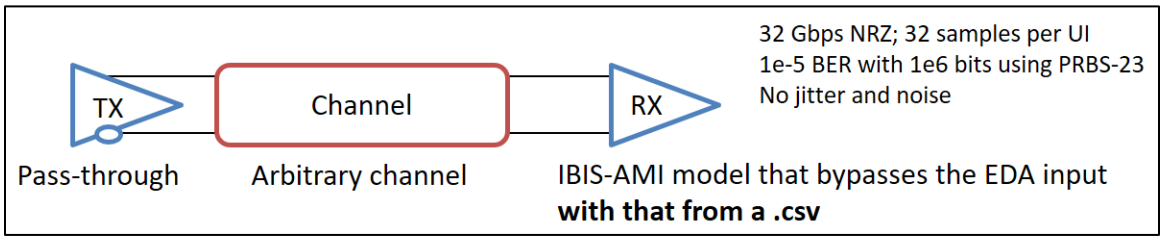

Figure 6. Channel simulation setup.

Figure 6 gives an overview of the channel simulation setup for 32 Gbps NRZ operation with 32 samples per UI, running 1e6 bits using PRBS-23. This interface targets 1e-5 BER, with no extrapolation. Jitter and noise are zeroed out as part of this work.

The transmitter IBIS-AMI model is a pass-through; no equalization is imposed. The receiver IBIS-AMI model bypasses the EDA input by doing the following.

1. When running the statistical flow, the model reads the impulse_matrix which contains the channel, TXLE, CTLE and DFE from a CSV and generates sampling information.
2. When running the bit-by-bit flow, the model reads the impulse_matrix which contains the channel, TXLE, CTLE and DFE and then convolves the result with an ideal bit-pattern, then generates sampling information.

The channel model is taken from the IEEE 802.3 public area [4], which uses a backplane cable. Figure 7 shows the frequency response of the channel, exhibiting a loss of -28.53 dB at 16 GHz. The equalized impulse_matrix is generated using the Channel Operating Margin (COM) v2.75 tool [4] with a fixed TXLE, CTLE and 3-tap DFE using a MM PD and Mod-MM PD. Figure 8 shows the COM spreadsheet which was used.

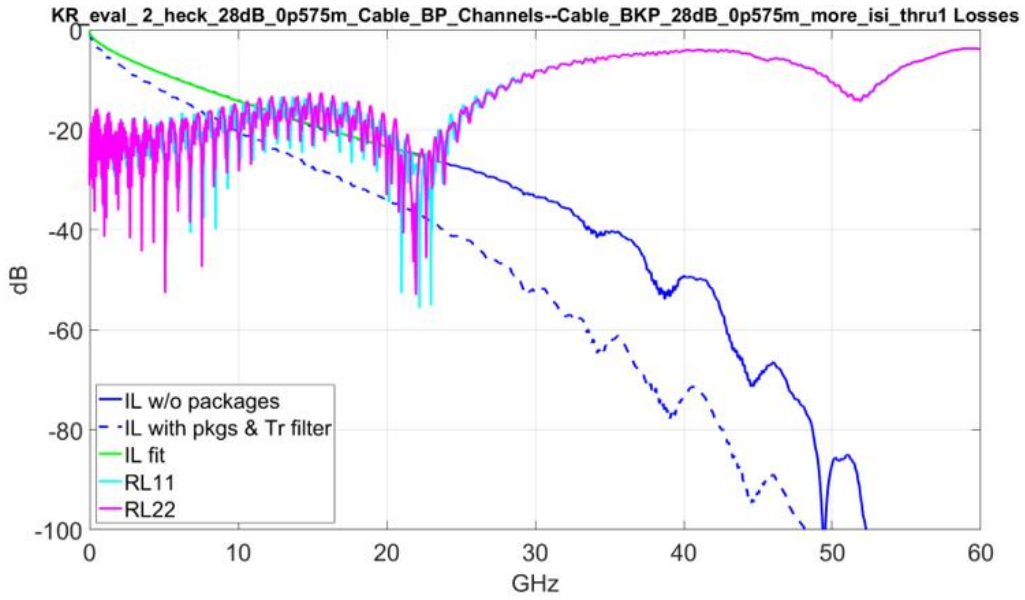

Figure 7. Frequency response of the channel.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Table 93A-1 parameters					I/O control				Table 93A–3 parameters		
2	**Parameter**	**Setting**	**Units**	**Information**		DIAGNOSTICS	1	logical		**Parameter**	**Setting**	**Units**
3	f_b	32	GBd			DISPLAY_WINDOW	1	logical		package_tl_gamma0_a1_a2	[0 0.002 0.0003]	
4	f_min	0.05	GHz			CSV_REPORT	1	logical		package_tl_tau	6.141E-03	ns/mm
5	Delta_f	0.01	GHz			RESULT_DIR	.\results\100GEL_1_PK_KR_{date}			package_Z_c	[87.5 87.5 ; 92.5 92.5]	Ohm
6	C_d	[1.2e-4 1.2e-4]	nF	[TX RX]		SAVE_FIGURES	0	logical		benartsi_3ck_01_0119 & mellitz_3ck_01_0119		
7	L_s	[0 0]	nH	[TX RX]		Port Order	[1 3 2 4]			Table 92–12 parameters		
8	C_b	[0 0]	nF	[TX RX]		RUNTAG	KR_eval_			**Parameter**	**Setting**	
9	z_p select	[2]		[test cases to run]		COM_CONTRIBUTION	0	logical		board_tl_gamma0_a1_a2	*[0 3.8206e-04 9.5909e-05]*	
10	z_p (TX)	[12 31; 1.8 1.8]	mm	[test cases]		Operational				board_tl_tau	*5.790E-03*	ns/mm
11	z_p (NEXT)	[12 29; 1.8 1.8]	mm	[test cases]		COM Pass threshold	3	dB		board_Z_c	100	Ohm
12	z_p (FEXT)	[12 31; 1.8 1.8]	mm	[test cases]		ERL Pass threshold	10.5	dB		z_bp (TX)	*110.3*	mm
13	z_p (RX)	[12 29; 1.8 1.8]	mm	[test cases]		DER_0	1.00E-05			z_bp (NEXT)	*110.3*	mm
14	C_p	[0.87e-4 0.87e-4]	nF	[TX RX]		T_r	6.16E-03	ns		z_bp (FEXT)	*110.3*	mm
15	R_0	50	Ohm			FORCE_TR	1	logical		z_bp (RX)	*110.3*	mm
16	R_d	[50 50]	Ohm	[TX RX]						C_0	[0.29e-4]	nF
17	A_v	0.415	V			TDR and ERL options				C_1	[0.19e-4]	nF
18	A_fe	0.415	V			TDR	1	logical		**Include PCB**	**0**	**logical**
19	A_ne	0.608	V			ERL	1	logical				
20	L	2				ERL_ONLY	0	logical		Floating Tap Control		
21	M	32				TR_TDR	0.01	ns		N_bg	0	0 1 2 or 3 groups
22	filter and Eq					N	3000			N_bf	0	taps per group
23	f_r	0.75	*fb			beta_x	2.3407E+09			N_f	0	UI span for floating taps
24	c(0)	0.54		min		rho_x	0.19			bmaxg	0.2	max DFE value for floating taps
25	c(-1)	[-0.1667]		[min:step:max]		fixture delay time	[0 0]	[port1 port2]		cable assemblies require this for each HCB		
26	c(-2)	[0]		[min:step:max]		TDR_W_TXPKG	0			ICN parameters (v2.73)		
27	c(-3)	[0]		[min:step:max]		N_bx	3	UI		f_f	21.448	
28	c(1)	[-0.0417]		[min:step:max]		Receiver testing				f_n	21.448	
29	N_b	3	UI			RX_CALIBRATION	0	logical		f_2	24.000	
30	b_max(1)	0.8				Sigma BBN step	5.00E-03	V		A_ft	0.600	
31	b_max(2..N_b)	0.3				Noise, jitter				**A_nt**	**0.600**	
32	g_DC	[-15]	dB	[min:step:max]		sigma_RJ	0	UI		heck_3ck_03b_0319	**Adopted Mar 2019**	
33	f_z	12.8	GHz			A_DD	0	UI		**walker_3ck_01d_0719**	**Adopted July 2019**	
34	f_p1	12.8	GHz			eta_0	8.2E-09	V^2/GHz		**result of R_d=50**		
35	f_p2	32	GHz			SNR_TX	100	dB		**benartsi_3ck_01a_0719**	**no used for KR**	
36	g_DC_HP	[0]		[min:step:max]		R_LM	1			**mellitz_3ck_03_0919**		
37	f_HP_PZ	0.4	GHz			CDR	Mod-MM	M or Mod-MM		under consideration		

Figure 8. COM spreadsheet snapshot.

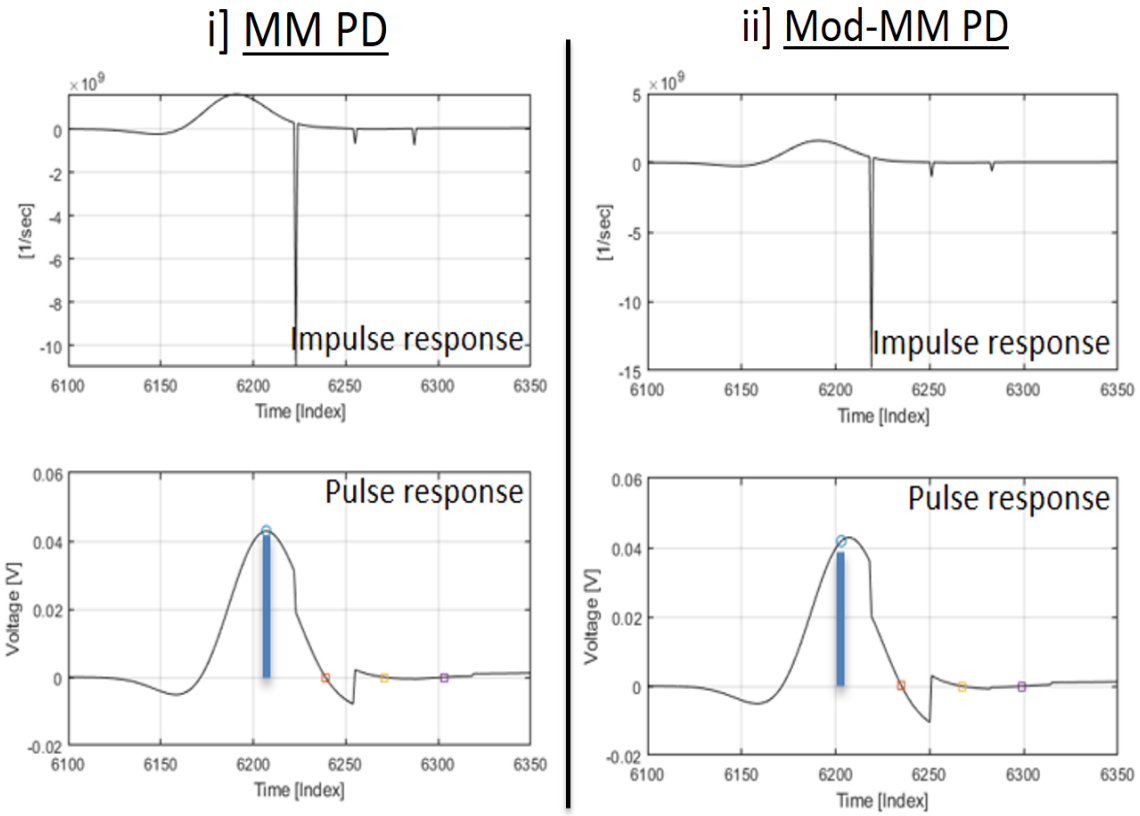

Figure 5. Equalized impulse and pulse response when using MM PD and Mod-MM PD.

Figure 9 shows the equalized impulse and pulse response when using the MM PD and Mod-MM PD. The wave part of the bit-by-bit flow is generated by convolution of the equalized pulse response with an ideal bit-pattern of PRBS-23. The convolution engine is part of the receiver IBIS-AMI model. This ensures the same waveform (impulse_matrix for statistical flow and wave for bit-by-bit flow) is gives to the different EDA tools, wherein the receiver IBIS-AMI reads the impulse_matrix from a CSV and then convolves it with an ideal bit-pattern pf PRSB-23 to generate the wave. Further, given the same waveform read from a CSV, the sampling information generated by the receiver IBIS-AMI model remains the same when running across the different EDA tools.

The results of the different EDA tools is compared to a reference, which is generated using the code given in Appendix A. In this way, this work presents a fair comparison among the different EDA tools for eye shape and eye margin along with position of the sampling point.

A. Results of the bit-by-bit flow across the different EDA tools

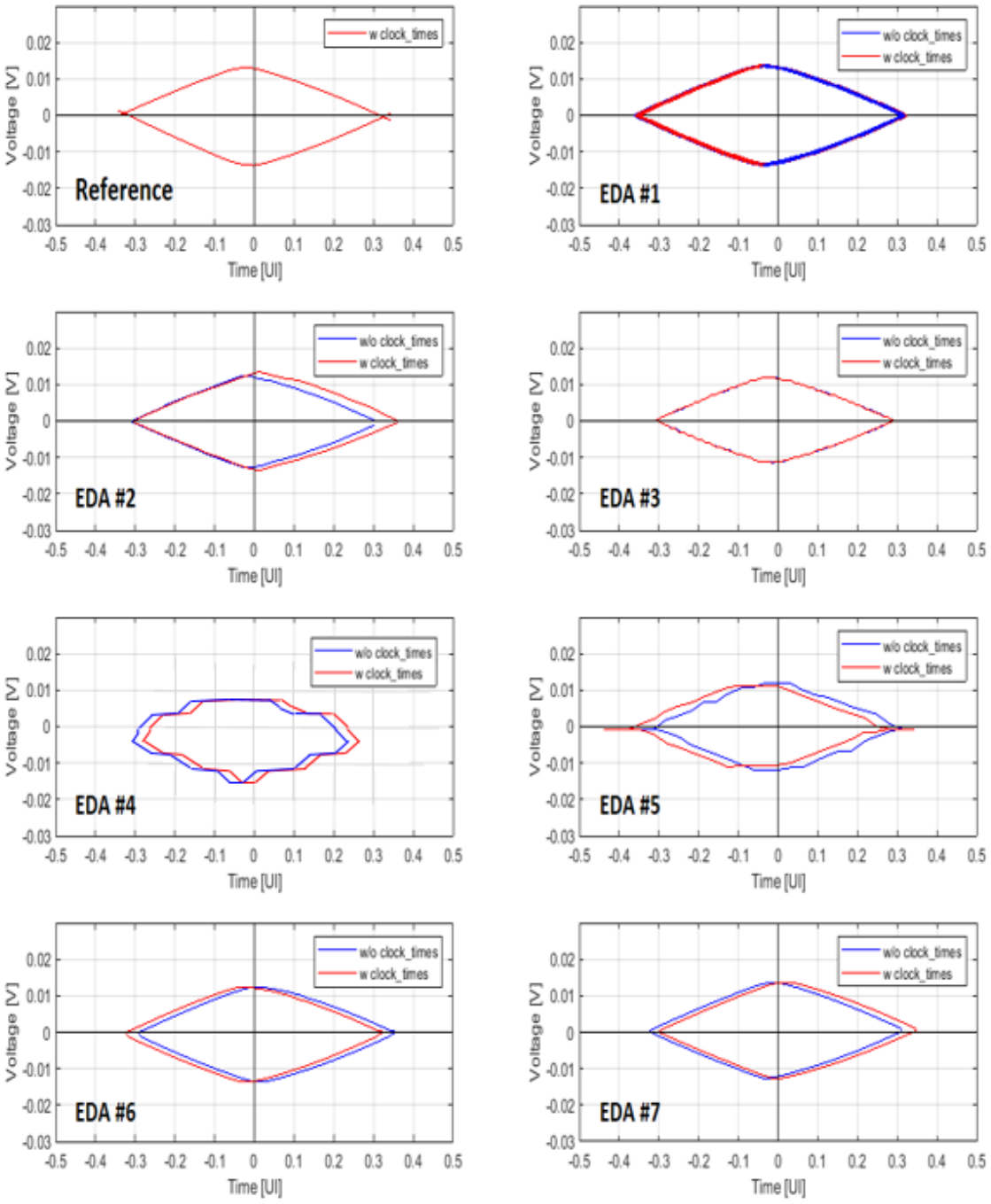

Figure 9. Eye contour with the bit-by-bit flow when using MM PD.

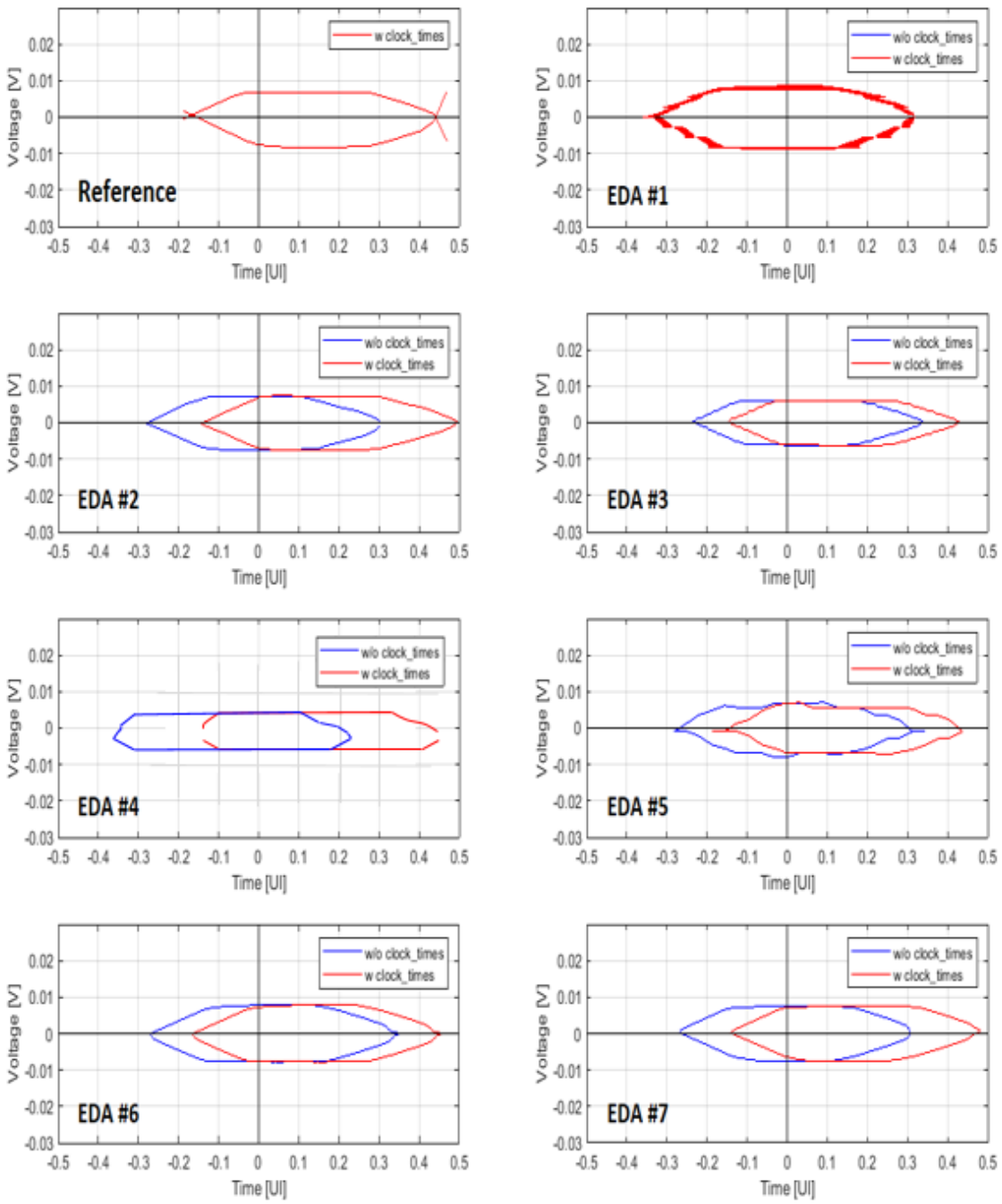

Figure 10. Eye contour with the bit-by-bit flow when using Mod-MM PD.

Table 4. Bit-by-bit BER contour margin with clock_times returned by the receiver IBIS-AMI.

Sampling Method	EDA tool	EH [mV]	EW [UI]
MM-PD	Reference	26.6	0.66
	EDA #1	25.7	0.64
	EDA #2	26.4	0.67
	EDA #3	22.3	0.60
	EDA #4	24.7	0.54
	EDA #5	21.7	0.59
	EDA #6	25.6	0.65
	EDA #7	26.8	0.64
Mod-MM PD	Reference	14.4	0.64
	EDA #1	15.4	0.65
	EDA #2	13.8	0.64
	EDA #3	12.1	0.58
	EDA #4	12.0	0.50
	EDA #5	14.7	0.65
	EDA #6	15.5	0.62
	EDA #7	14.3	0.61

Figures 9 and 10 show the results of the bit-by-bit flow across the different EDA tools for MM PD and Mod-MM PD respectively. When the receiver IBIS-AMI does not return clock_times then the EDA tool is responsible to determine the clock_times. In the case of the MM PD, one may observe reasonable matching of the EDA tools to get a similar positioned eye to the reference. Further, none of the EDA tools are able to get a similar positioned eye matching the reference in the absence of sampling information from the receiver IBIS-AMI (without clock_times).

Table 4 shows the BER contour margin for the bit-by-bit flow when using MM PD and Mod-MM PD, with clock_times returned by the receiver IBIS-AMI. The eye height and eye width with respect to the sampling point (0 UI) are noted. The results among the different EDA tools are “in the ballpark” of the reference. A standard deviation of 2.0 mV/0.05 UI can be observed for the set of seven EDA tools.

B. Results of the statistical flow across the different EDA `tools

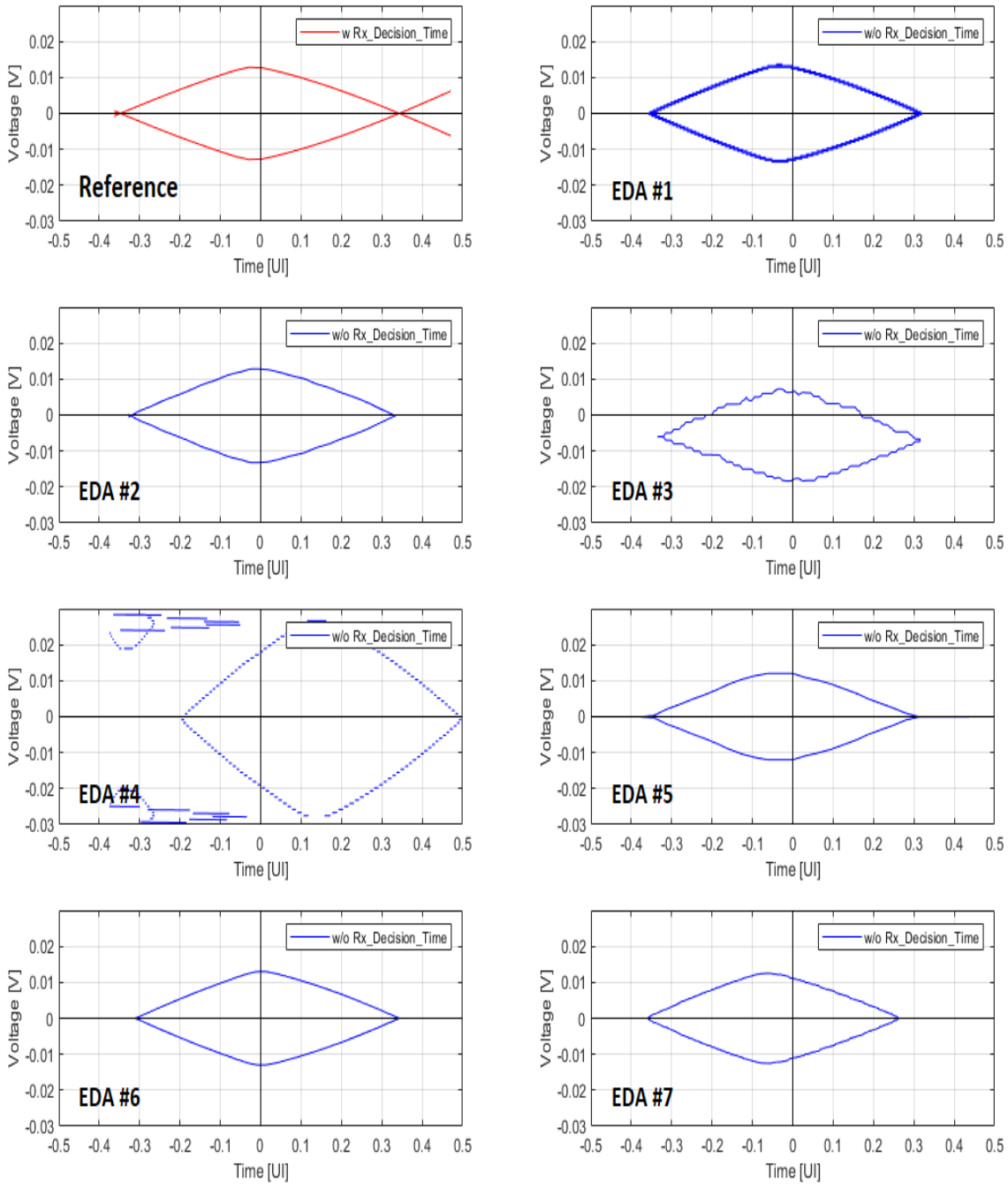

Figure 6. Eye contour with the statistical flow when using MM PD.

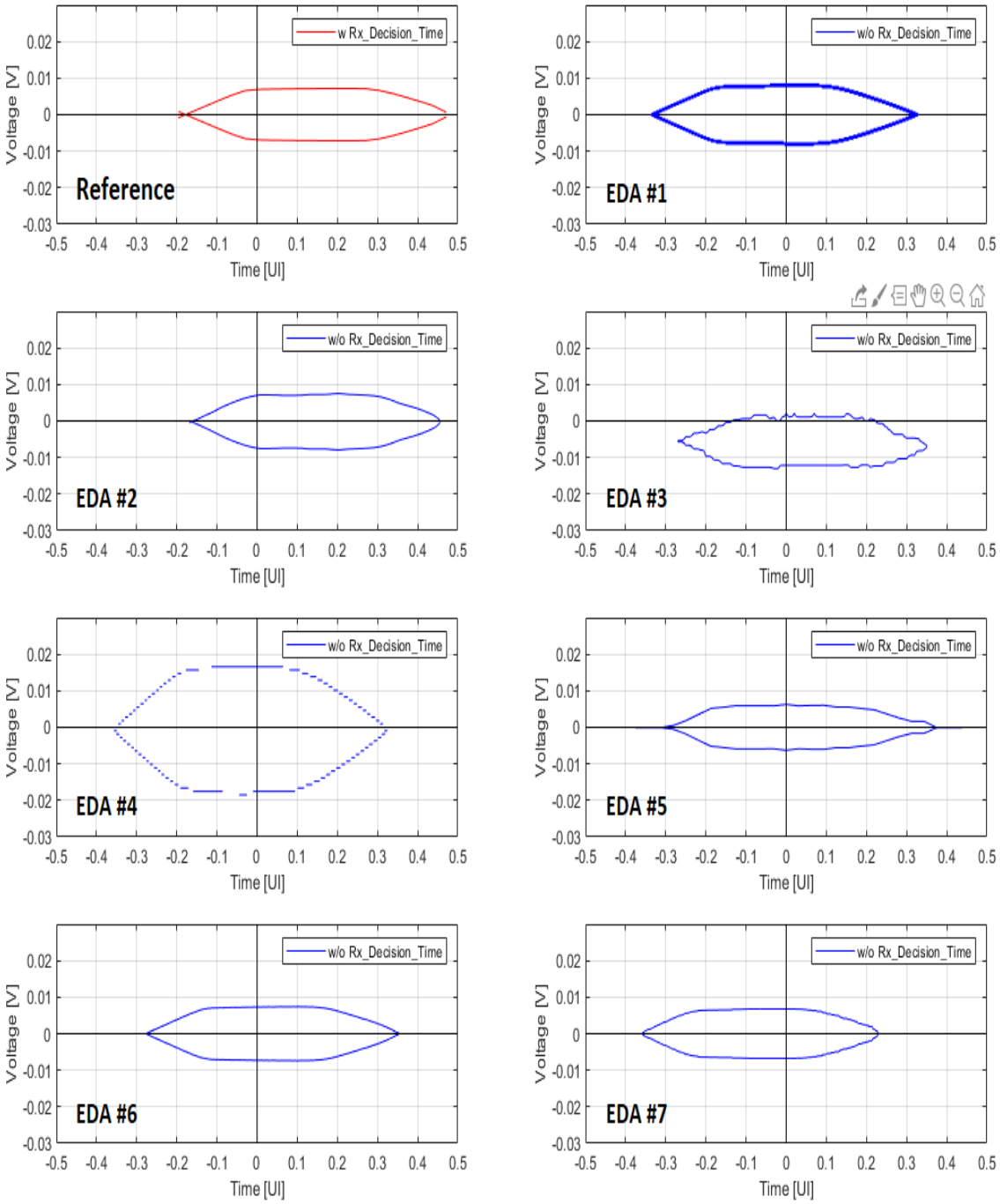

Figure 12. Eye contour with the statistical flow when using Mod-MM PD.

Table 5. Statistical BER contour margin without sampling information (Rx_Decision_Time) returned by the receiver IBIS-AMI.

Sampling Method	EDA tool	Max. EH [mV]	EW [UI]
MM-PD	Reference	25.6	0.67
	EDA #1	25.9	0.64
	EDA #2	25.9	0.66
	EDA #3	25.4	0.65
	EDA #4	53.3	0.66
	EDA #5	24.0	0.69
	EDA #6	26.0	0.65
	EDA #7	25.1	0.62
Mod-MM PD	Reference	14.2	0.67
	EDA #1	15.4	0.63
	EDA #2	14.9	0.61
	EDA #3	13.2	0.62
	EDA #4	33.2	0.67
	EDA #5	11.8	0.69
	EDA #6	14.7	0.63
	EDA #7	14.4	0.58

Figure 11 and 12 show the results of the statistical flow across the different EDA tools for MM PD and Mod-MM PD respectively. In the case of the statistical flow, Buffer Issue Resolution Document (BIRD) 205 introduced Rx_Decision_Time, which addresses the passing of sampling information by the receiver IBIS-AMI to the EDA tool. IBIS 7.1 is expected to be released in the later end of 2021. Current EDA tools which refer to IBIS 7.0 do not support sampling information by the receiver IBIS-AMI.

When the receiver IBIS-AMI does not provide sampling information through a mechanism such as Rx_Decision_Time, then the EDA tool is responsible to determine when to sample the equalized response (impulse, pulse, or step). From the results, one can observe that only a few of the EDA tools is able to get a similarly-positioned eye matching the reference in the absence of sampling information from the receiver IBIS-AMI (without Rx_Decision_Time).

EDA tool #4 appears to be scaling the differential waveform by two and may be ignored when noting the margin difference across the different EDA tools.

Table 5 shows the BER contour margin for the statistical flow when using MM PD and Mod-MM PD with clock_times returned by the receiver IBIS-AMI. Given the absence of sampling information by the receiver IBIS-AMI model, the maximum eye height and eye width is noted. The results among the different EDA tools are in the ballpark of the reference. Observed is a standard deviation of 1.3 mV/ 0.04 UI for the set of six EDA tools.

Conclusion

This paper presented the importance of sampling information when running channel simulation with algorithmic models under the IBIS-AMI specification. A brief overview of the sampling controls (Rx_Decision_Time and clock_times) in the statistical and bit-by-bit flow along with 'RECEIVER RECOVERED CLOCK RESERVED PARAMETERS' was shared.

Eye generation using the Mueller-Muller (MM) and Modified Mueller-Muller (Mod-MM) phase detector algorithm in different IBIS-AMI models was compared across EDA tools. This work will help the reader to think beyond eye height and eye width by presenting the importance of eye symmetry and eye margin with respect to sampling information (voltage margin top, voltage margin bottom, timing margin left and timing margin right), eye density and signal-to-noise ratio.

Results of the bit-by-bit and statistical flows across seven EDA tools was presented. A receiver IBIS-AMI model was generated that bypasses the input from the EDA tool, instead using that from a CSV file for the output impulse_matrix and wave in the statistical and bit-by-bit flows respectively. The idea was to provide an identical waveform along with sampling information (Rx_Decision_Time and clock_times) to the different EDA tools and note the eye margin difference among the EDA tools. The results across the different EDA tools was similar to the reference. One can observe a standard deviation of 2.0 mV/ 0.05 UI for the set of seven EDA tools when running the bit-by-bit flow for 1e-5 BER given 1e6 bits using PRBS-23. Also observable is a standard deviation of 1.3 mV/ 0.04 UI for the set of six EDA tools when running statistical flow for 1e-5 BER.

The results of this work show that there is a difference in margin for the different EDA tools values despite the same input waveform and sampling information. Despite these differences, the results across the different EDA tools come generally close to that of the reference. This work was carried out by working with the different EDA tool vendors in running the simulation..

References

[1] Romi Mayder et al., “IBIS-AMI Model Simulations Over Six EDA Platforms”, DesignCon 2015, Santa Clara, California, USA.

[2] Douglas Burns et al., “Understanding IBIS-AMI Simulations”, DesignCon 2015, Santa Clara, California, USA.

[3] IBIS, ’(I/O Buffer Information Specification)’, 15-March-2019. [Online]. Available: https://ibis.org/ver7.0/ver7_0.pdf. [Accessed: 25- May-2021].

[4] ieee802.org, ’IEEE P802.3ck 100 Gb/s, 200 Gb/s, and 400 Gb/s Electrical Interfaces Task Force Public Area’, 2019. [Online]. Available: http://www.ieee802.org/3/ck/public/index.html. [Accessed: 12- July-2019].

Appendix A. BER contour script

```
clc;
clear all;
close all;

%Author: Adam Gregory
%artificial impulse response
%triangle wave with risetime twice as fast as fall time
%followed by random noise with sigma= 0.5 mV combined with decaying
%exponential so noise falls to zero as delay increases

ir= [zeros(1,100) (0.1:0.04:1)*2.5e-3 (1-0.02:-0.02:0)*2.5e-3
randn(1,1000)*5e-4.*exp(-5e-3*[1:1000])];
UI= 32;
num_bits= 1e3;
pulse= filter(ones(1,UI),1,ir);
%set [-0.5 0.5] bit pattern (no DC)
bit_pattern= round(rand(1,num_bits))-0.5;
%set bit pattern to UI increments to conv with pulse
bit_pattern_UI(1:UI:UI*num_bits)= bit_pattern;
wave= conv(pulse,bit_pattern_UI);
%easy sample at peak of pulse
[tmp,ts]= max(pulse);
%clock times are UI increments of ts
clock_times= (1:UI:length(bit_pattern)*UI)+ts-1;

%eye contour
half_UI= ceil(UI/2);
%eye contour must know whether is a 1 or 0
%if the pattern was unknown, this can also be discovered by checking if
the
%sample voltage [wave(clock_times)] is positive/negative.  However that
%only works for open eye.
sv= wave(clock_times);
ones_idx= find(bit_pattern>0);
zeros_idx= find(bit_pattern<0);
for j=1:UI
    sample_vector= wave(clock_times-half_UI+j);
    %1st column=1 contour
    %2nd column= 0 contour
    eye_contour(j,1)= min(sample_vector(ones_idx));
    eye_contour(j,2)= max(sample_vector(zeros_idx));
end
%full eye density
for j=1:UI
    sample_vector= wave(clock_times-half_UI+j);
    eye_density(1:num_bits,j)= sample_vector;
end

figure;
plot(eye_contour);
figure;
plot(eye_density');
```