IBIS-AMI Modeling and Simulation of DDR5 Systems

Fangyi Rao, Hee-Soo Lee and Jing-Tao Liu, Keysight

Wendem Beyene, Intel

DesignCon IBIS Summit
Santa Clara, California
January 31, 2020

DDR5 Highlights

- Data rate is increased to 3200-6400 MT/s in DDR5, resulting in higher ISI
- Equalizations including CTLE and DFE are utilized in memory controller and DRAM to mitigate ISI
- Timing and voltage margins are specified at extremely low BER. Jitter and noise becomes critical factors.
- Millions of bits are required to reliably estimate margins
- Solution: extend AMI methodology from differential signal in SerDes channel to single-ended signal in DDR channel

Challenges in DDR AMI Simulation

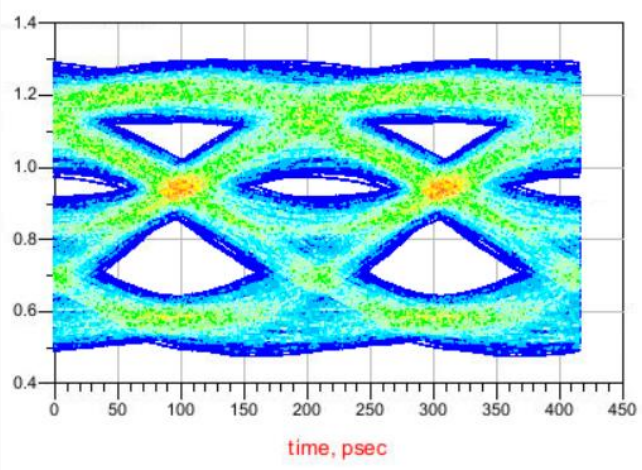
- DQ and CAC are single-ended (SE) signals that have both differential and common modes
- SE signal has asymmetric rise and fall edges
- DDR Rx does not have embedded CDR as in SerDes. DQ Rx DFE is clocked by forwarded clock
- AMI modeling is new to most DDR designers
- …

Common Mode in Single-ended Signal

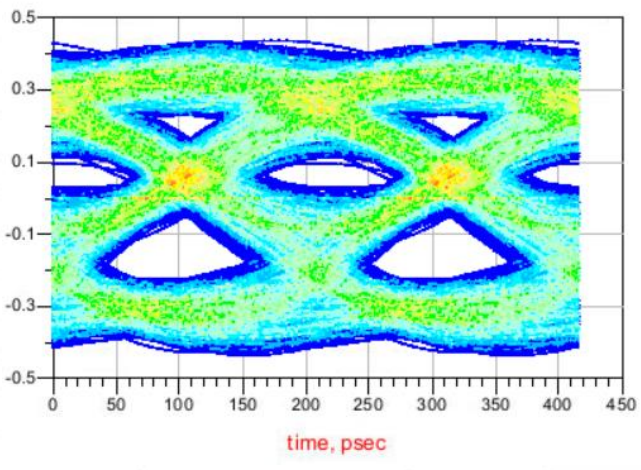
- Resolved in BIRD197.7 with the introduction of a new reserved parameter DC_Offset
- Physical (SE) waveform at Rx DLL input node = Rx GetWave input waveform + DC_Offset
- Rx GetWave input waveform centers around 0V, as in SerDes
- DC_Offset value is a constant that is characterized and passed into Rx Init by EDA tool
- Rx DLL can choose to internally reconstruct the physical input waveform by adding DC_Offset to GetWave input waveform
- Rx GetWave output waveform centers around 0V, same as in SerDes
- EDA tool can choose to add DC_Offset to Rx GetWave output waveform for display

Common Mode in Single-ended Signal (cont'd)

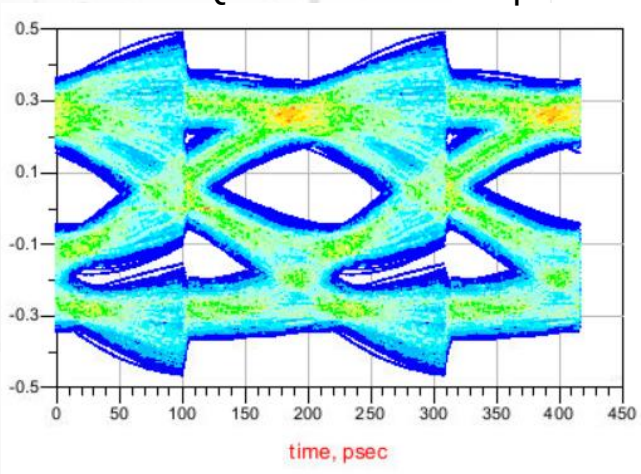
At DQ Rx package



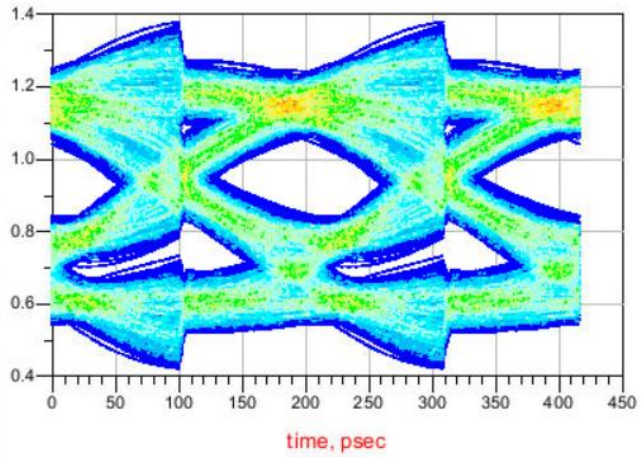
DQ Rx GetWave input



Raw DQ Rx GetWave output

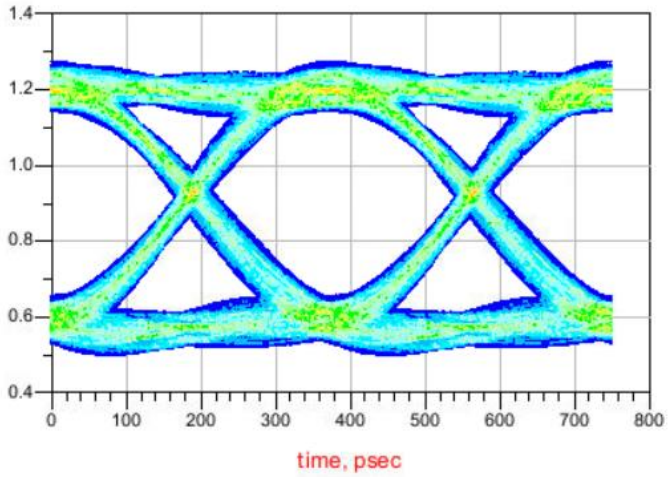


DC_Offset is to DQ Rx GetWave output

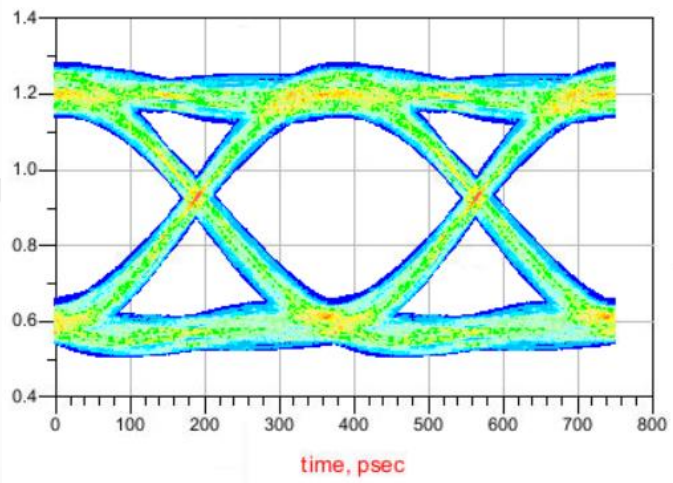


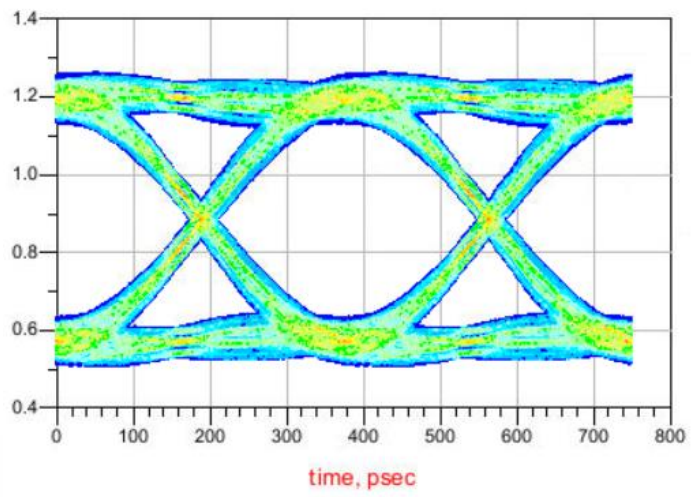
Asymmetric Rise and Fall Edges in Single-ended Signal

SPICE simulation



AMI simulation using asymmetric edges



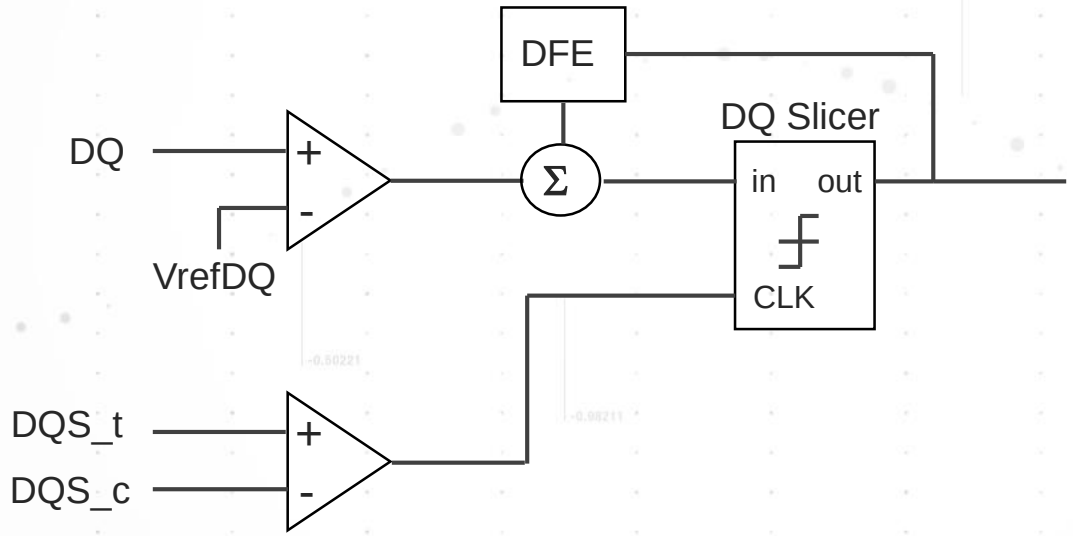


AMI simulation using symmetric edges

Simulation using symmetrical edges yields unrealistically symmetrical eye, resulting in inaccurate Vref determination and timing and voltage margin measurements

DFE and Clock Forwarding Architectures in Data Buffer

- DQ Rx utilizes DFE to mitigate ISI
- DQ DFE is clocked by DQS

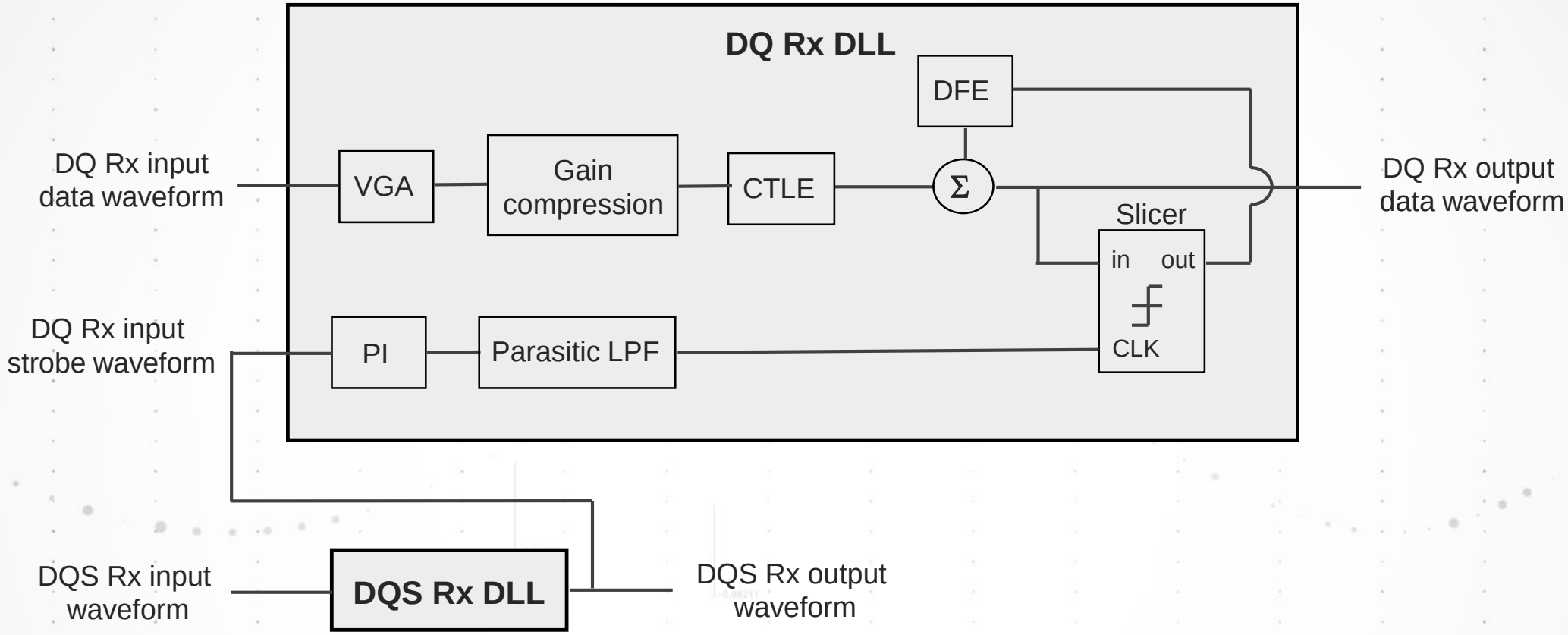


Proposal of A New GetWave API to Model Clock Forwarding

```
long AMI_GetWave2(double *wave,
                  double *wave_ref,
                  long wave_size,
                  double *clock_times,
                  char **AMI_parameters_out,
                  void *AMI_memory);
```

- The new GetWave2 function is implemented by DQ Rx model
- Two input waveforms and one output waveform
- wave: input and output data waveforms
- wave_ref: input strobe waveform, which is the output waveform of DQS Rx GetWave
- Sizes of wave and wave_ref are the same
- Rx model can choose to process wave_ref, typically in controller DQ Rx model
- clock_times: output clock times generated by Rx model based on wave_ref

Controller DQ Rx Model Example

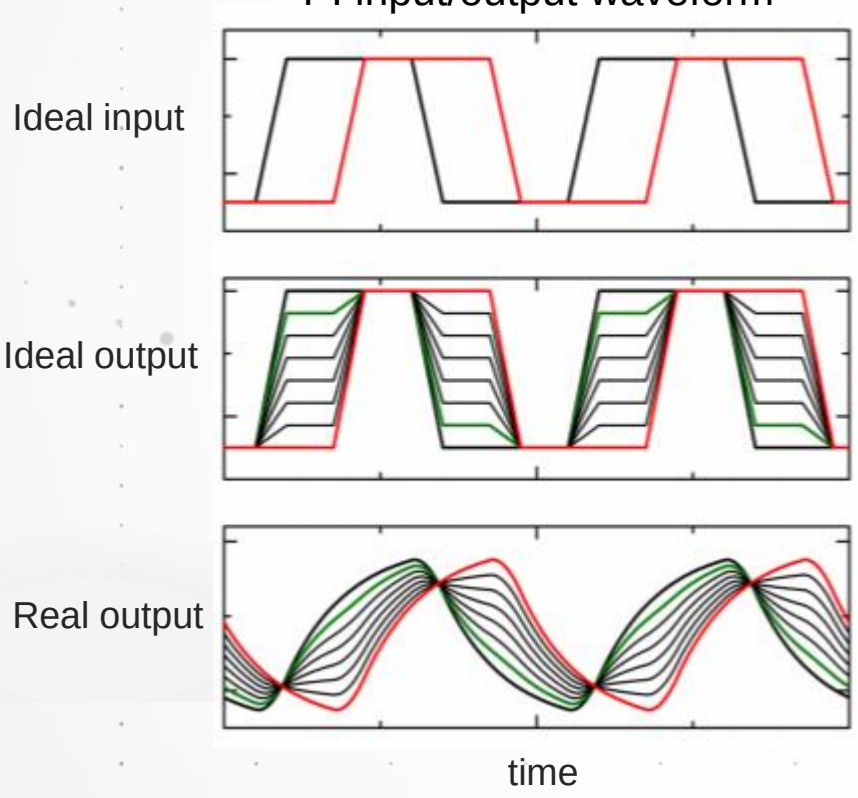


- Input strobe waveform is processed by a phase interpolator (PI)
- PI is dynamically trained by Rx DLL to adjust data-strobe skew for optimal DFE clocking
- Adaptive DFE
- Both PI training and DFE adaptation stop after Ignore_Bits to emulate system start-up training

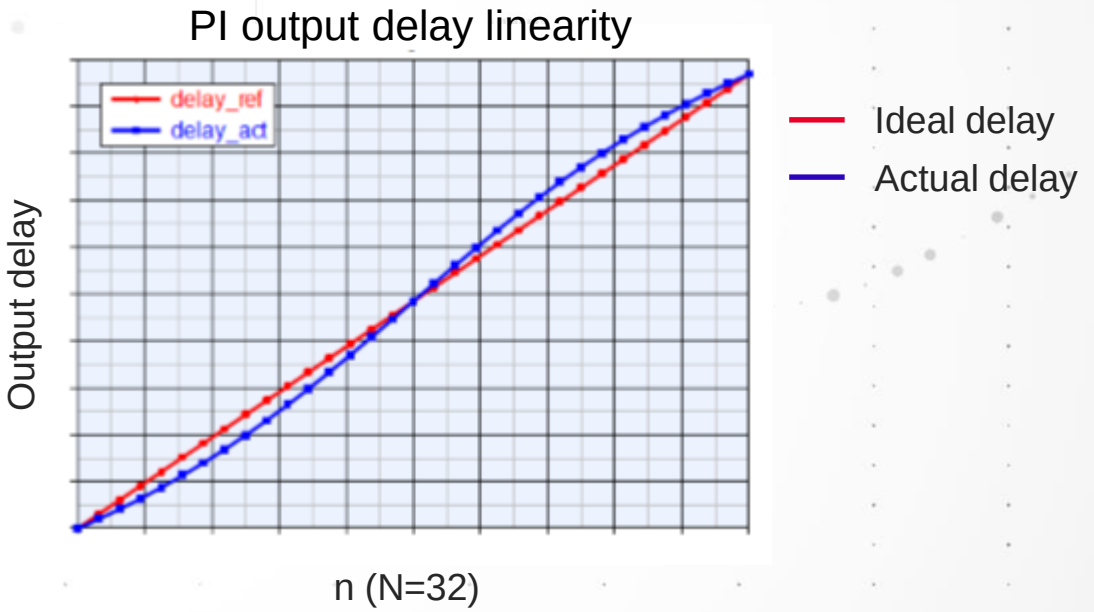
Phase Interpolator Review

$$v_{out}(t) = \frac{n}{N} v_{in}(t - \tau_1) + \frac{N - n}{N} v_{in}(t - \tau_2), \qquad n = 0,1, \dots, N$$

PI input/output waveform



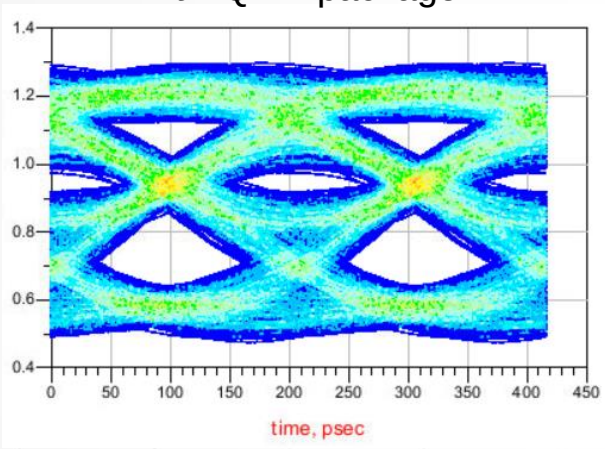
PI output delay linearity



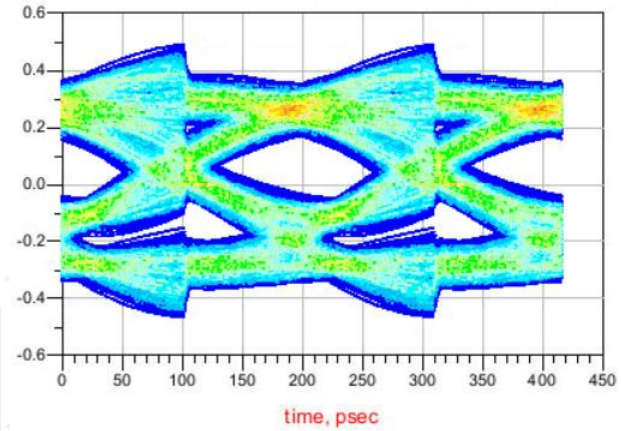
Phase Interpolator Training in Controller DQ Rx Model

Controller DQ Rx model can internally train the phase interpolator to adjust data-strobe skew for optimal DFE clocking

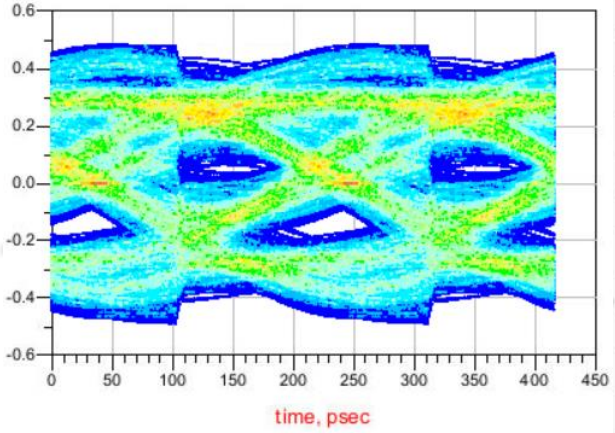
At DQ Rx package

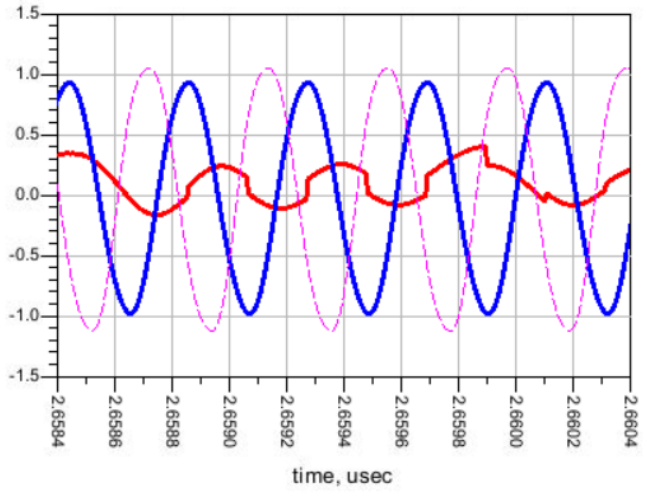


DQ Rx GetWave2 output
(with phase interpolator training)



DQ Rx GetWave2 output
(without phase interpolator training)





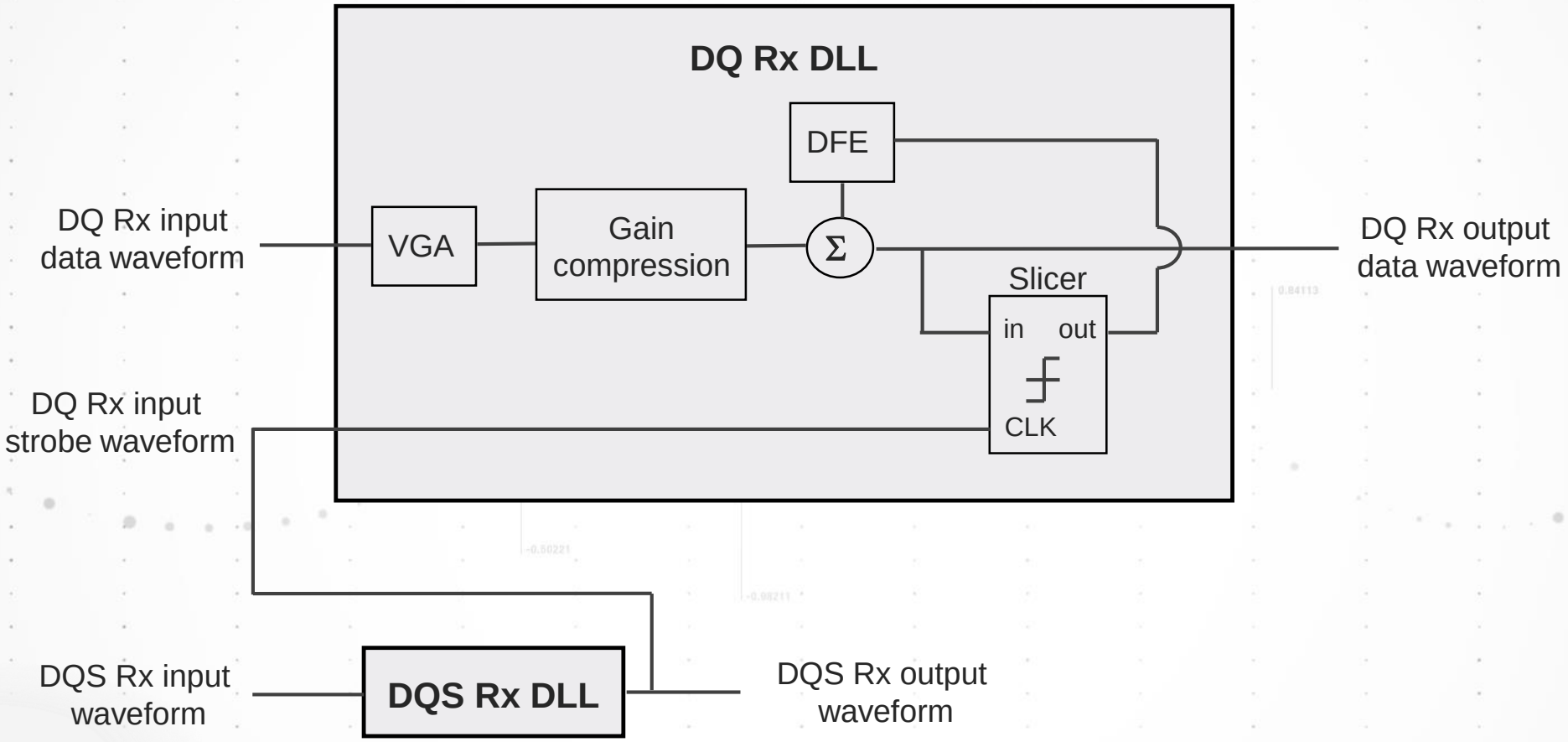
DQ Rx GetWave2 output

DQ Rx GetWave2 strobe input

Strobe after phase interpolator

Data and post-phase interpolator strobe are center-aligned

DRAM DQ Rx Model Example



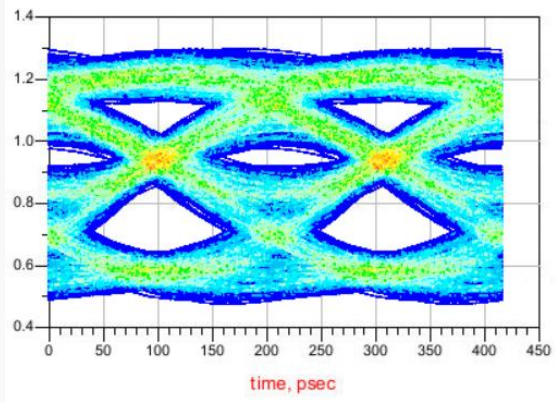
- Input strobe waveform is directly used to clock DFE
- DQ-DQS skew is optimized by controller write leveling

Jitter Tracking and Unmatched IO Rx

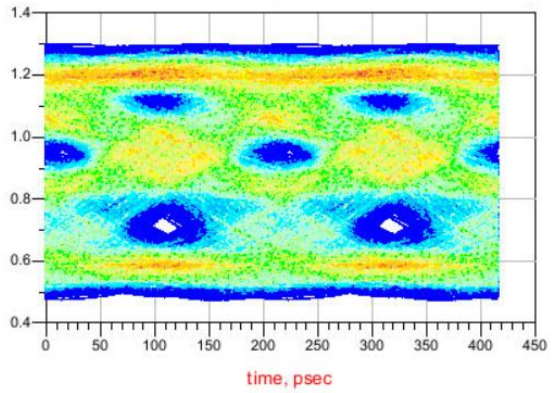
- Correlated jitters in DQ and DQS can be tracked in DQ Rx by clock forwarding
- DDR4 (and previous) DRAMs pad the DQ path to match DQ Rx to DQS
- DDR5 supports unmatched DQ and DQS Rx on both DRAM and controller sides
- Unmatched Rx reduces DQ-DQS jitter correlation and adversely impacts DQ Rx jitter tracking and DFE performance

Jitter Tracking and Unmatched IO Rx (cont'd)

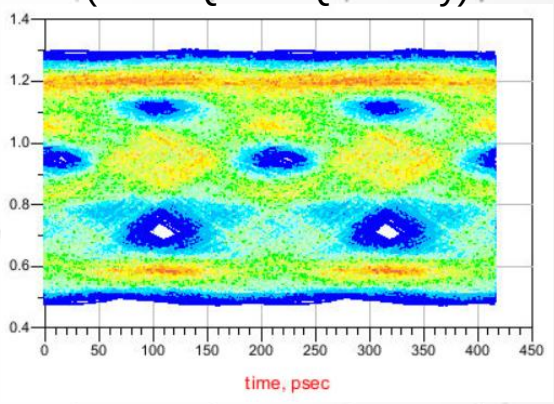
DQ Rx package without DQ & DQS Tx SJ



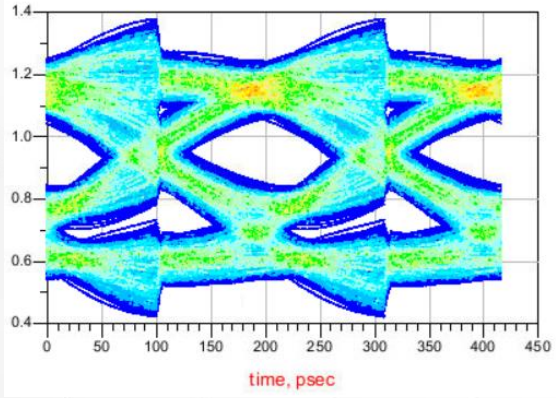
DQ Rx package with DQ & DQS Tx SJ
(0 DQS-to-DQ delay)



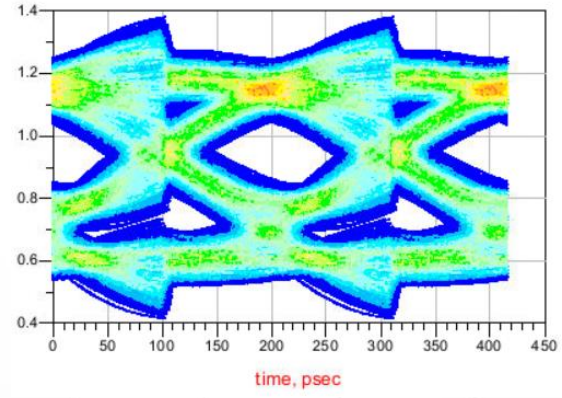
DQ Rx package with DQ & DQS Tx SJ
(5UI DQ-to-DQS delay)



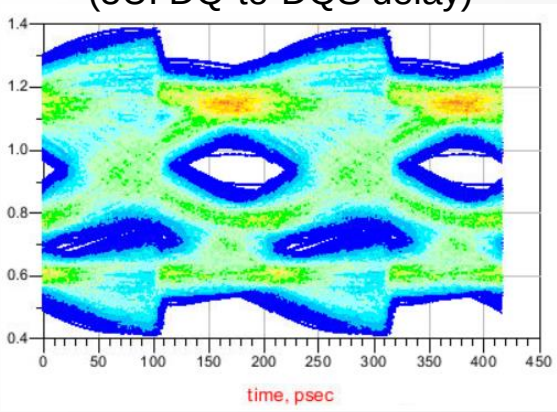
DQ Rx output without DQ & DQS Tx SJ



DQ Rx output with DQ & DQS Tx SJ
(0 DQS-to-DQ delay)



DQ Rx output with DQ & DQS Tx SJ
(5UI DQ-to-DQS delay)



KEYSIGHT TECHNOLOGIES intel

Summary

- Asymmetric rise and fall edges need to be taken into account in single-ended signal calculations
- Data buffer DFE is clocked by strobe
- Propose a new GetWave function with two input waveforms for data and strobe to model clock forwarding
- Controller DQ Rx model can internally adjust data-strobe skew to optimize DFE clocking
- Jitter tracking in DFE and unmatched IO Rx are captured by the new GetWave function