Modeling Forwarded Clock Interfaces with IBIS-AMI

Justin Butterfield

DesignCon 2019 IBIS Summit

Santa Clara, California

February 1, 2019

Agenda

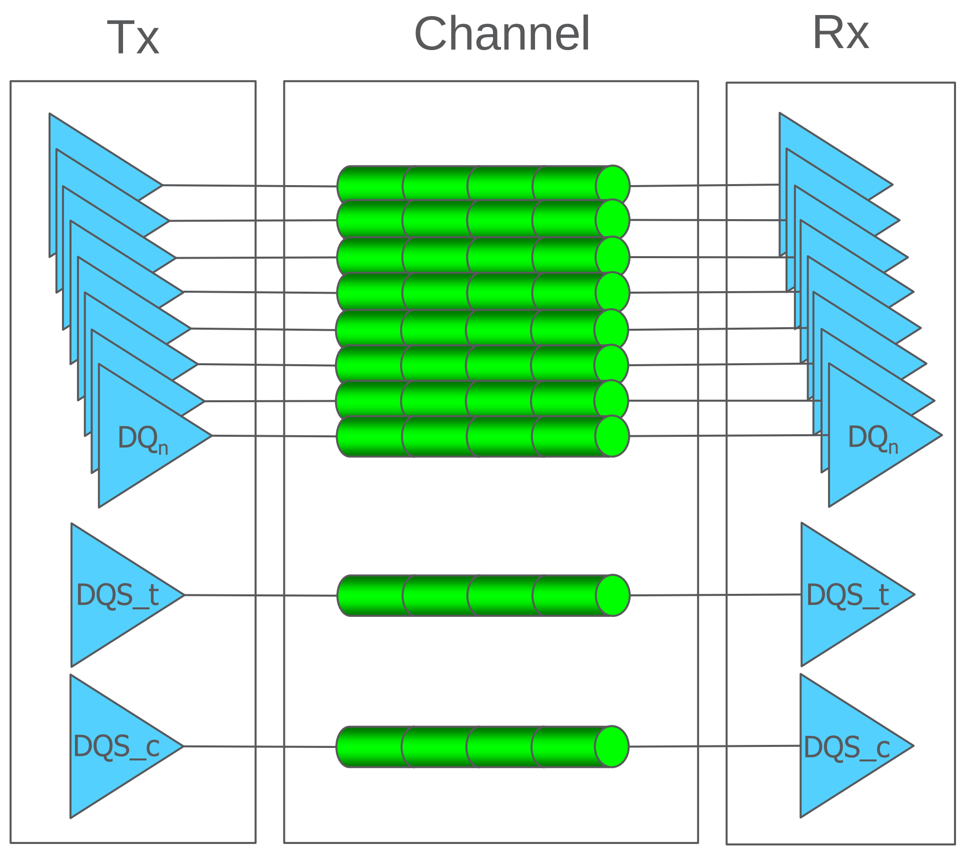
- Upcoming DRAM interfaces
- Forwarded clock architecture
- Currently proposed modeling approaches
- DRAM Rx circuit
- Correlated jitter
- Sinusoidal jitter example
- Using existing IBIS-AMI jitter parameters
- Summary and future discussion topics

Upcoming DRAM Interfaces

- Expected to implement equalization
  - Both at the DRAM and Controller side
    - DDR5
    - GDDR6
    - Other next-generation DDR
  - Likely include one or more EQ capabilities:
    - FFE, CTLE, DFE, etc.
- New simulation techniques required
  - Statistical analysis to predict behavior over millions of bits
  - IBIS-AMI can be used to model the equalization
- Forwarded clock problem
  - Traditional IBIS-AMI applications include clock data recovery (CDR)
  - DDR interfaces will continue to use a forwarded clock architecture
  - IBIS-AMI function APIs do not support inputting a clock

Forwarded Clock Architecture

- Forwarded Clock (Source Synchronous System)
  - DQ is clocked by DQS at DRAM Rx
  - DQS driven parallel to DQ
- Challenges for IBIS-AMI
  - IBIS-AMI assumes a CDR
  - DQ and DQS are independent paths
    - Each will have different jitter characteristics
    - Jitter from the Tx source (Rj, Dj, Sj, DCD)
    - Jitter at the Rx (Rj, Dj, Sj, DCD)
    - Intersymbol interference (ISI)
    - Crosstalk
  - Need careful accounting of jitter
    - Avoid double counting
    - Consider jitter budget for Controller and DRAM



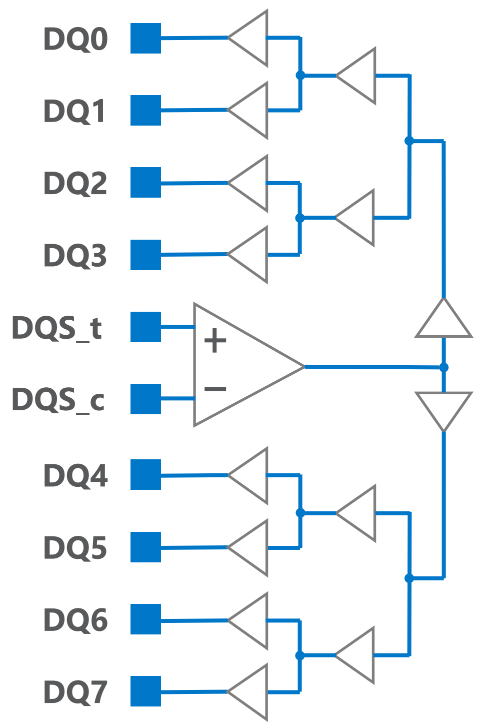
Currently Proposed Modeling Approaches

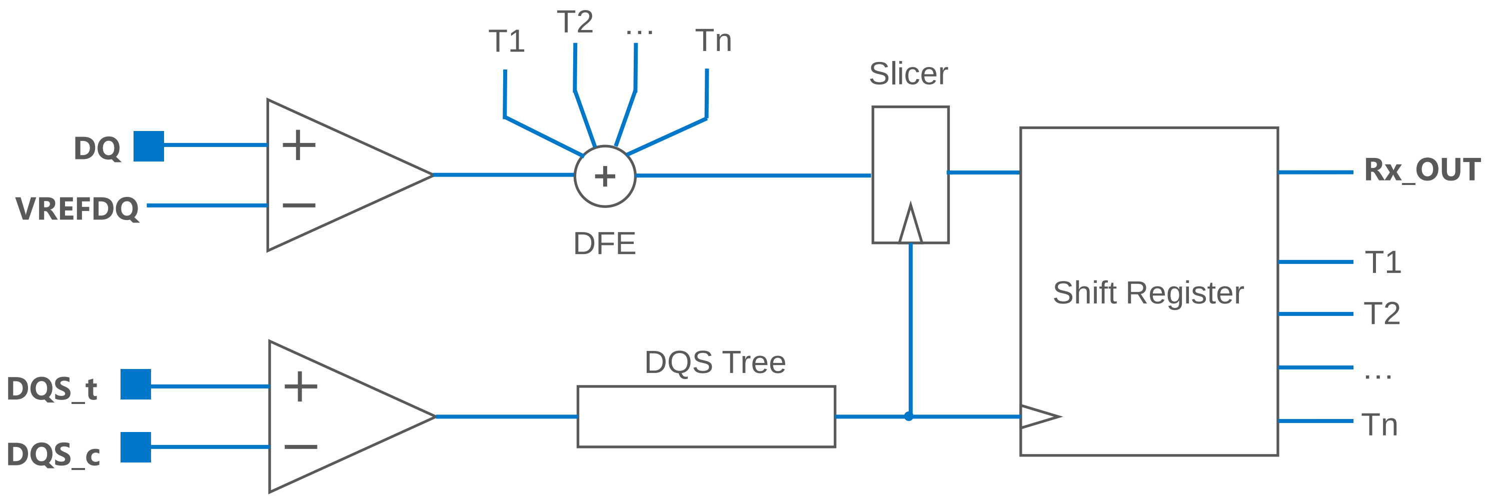
- 1. CDR as modeling construct
  - Concept:
    - Use existing model flows with a CDR built into the executable
    - Determines DFE decision point
    - Rely on existing Rx / Rx_Clock_Recovery jitter parameters
  - Pros:
    - Simple
    - Use existing flows
    - Rx model can be self contained
  - Cons:
    - Not a physical representation
    - Jitter modeling must make assumptions

- 2. clock_times as an input vector
  - Concept:
    - clock_times vector as input clock
    - Could use a new BIRD to use clock_times vector as input
    - Jitter modeling could be handled by the executable
  - Pros:
    - More physical representation
    - Potentially more accurate jitter modeling
  - Cons:
    - More complex
    - Requires new flows
    - clock_times from a source outside the executable
    - Need to handle the phase training

DRAM Rx Circuit

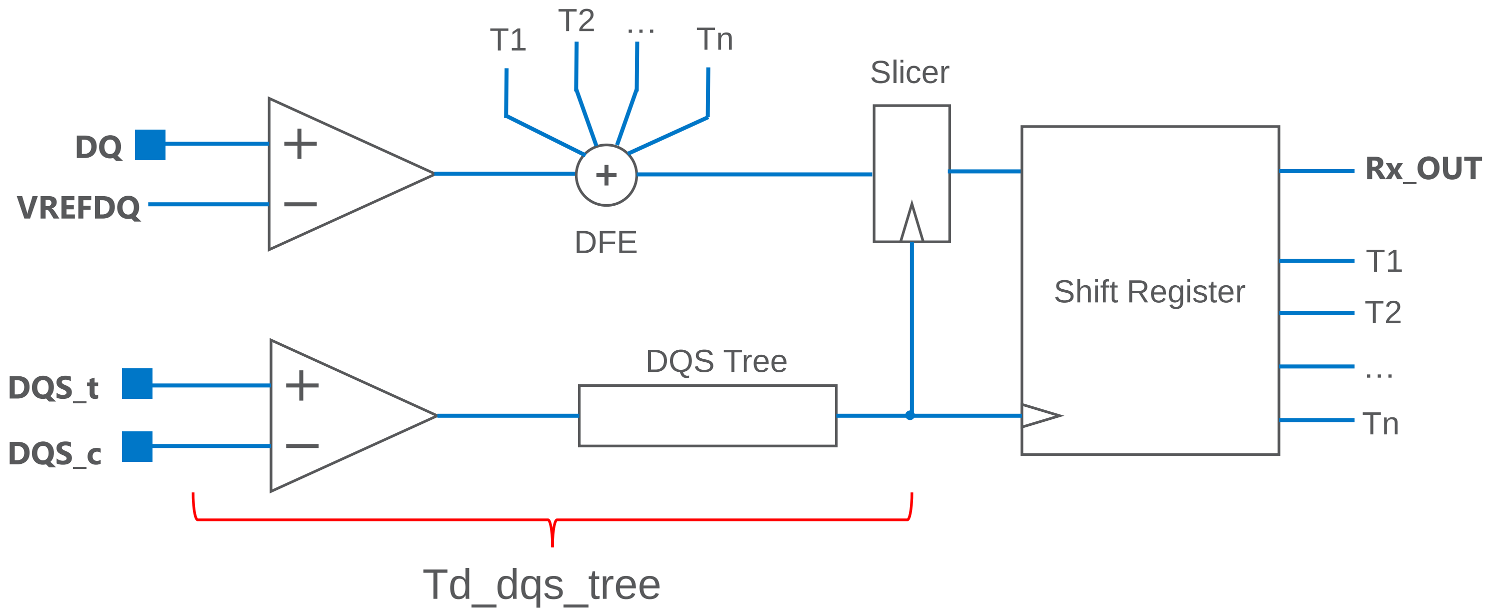
- DRAM Rx may include
  - VREFDQ, Analog Gain Stage, CTLE, DFE, etc.
  - Slicer clocked with DQS
- DQS Tree
  - DQS clock must be driven to each DQ in the byte
  - Significant DQS delay from pad to Slicer at each DQ
    - Due to physical length
    - On the order of 1ns (covers multiple UI)
    - Controller expected to train the phase alignment at Slicer





Correlated Jitter

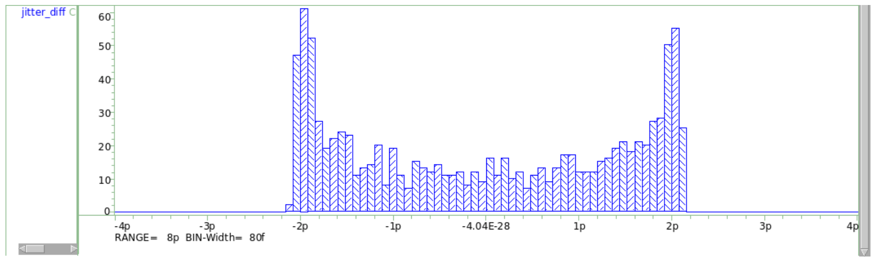
- Theory:
  - Should correlated jitter on both DQ and DQS cancel out?
  - Due to DQS tree time delay (Td_dqs_tree), this is **not** always the case
  - Jitter cancelation only below the jitter cutoff frequency (Jitter_cutoff_freq)
    - Td_dqs_tree ~= 1ns
    - Jitter_cutoff_freq << 1 / Td_dqs_tree
    - Jitter_cutoff_freq ~= 100MHz



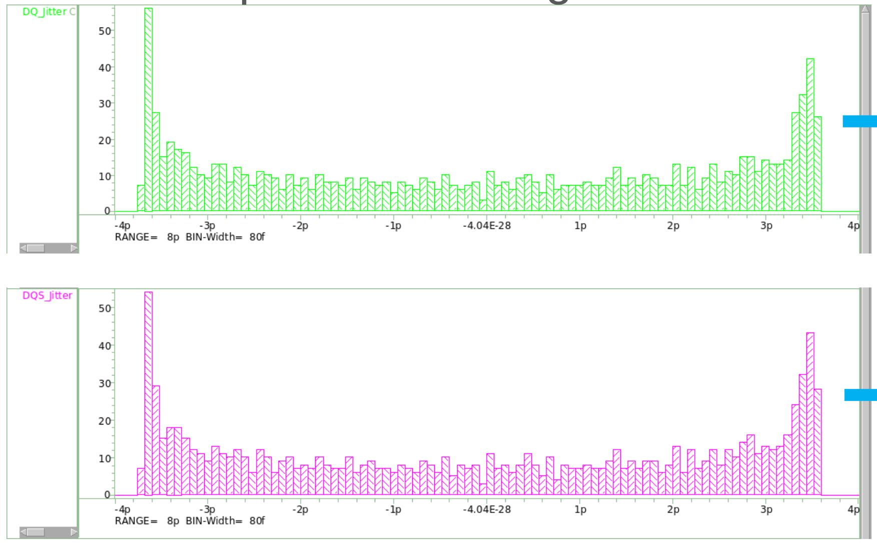
Sinusoidal Jitter Example

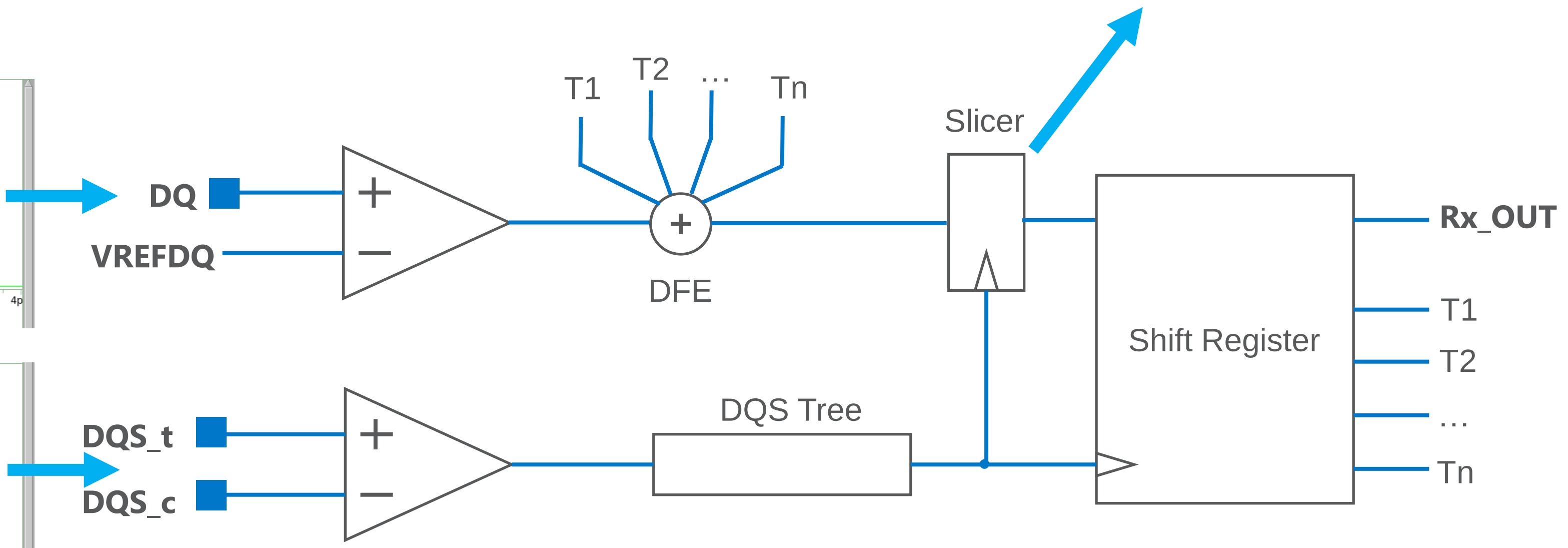
- Case 1: "Low Frequency SJ"
  - SJ frequency = **95MHz**
  - Td_dqs_tree ~= 1ns
  - ~+/-4ps correlated jitter input at DQ and DQS pads
  - Result is partial jitter cancelation
    - ~+/-2ps effective jitter at the Slicer

Effective Jitter Histogram at Slicer



Input Jitter Histograms

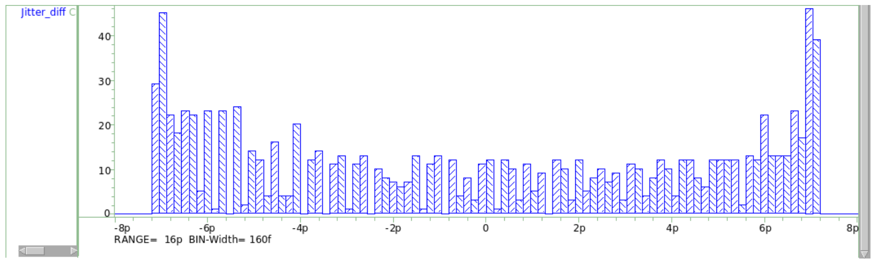




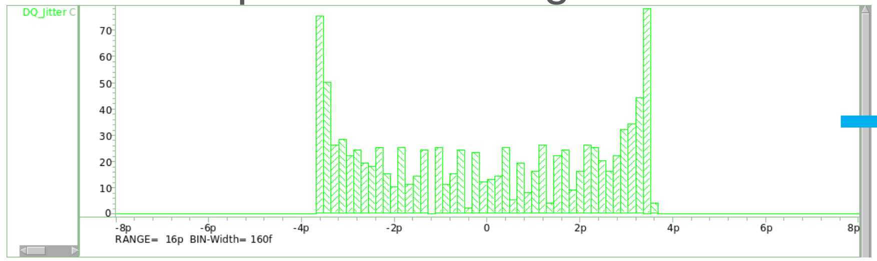
Sinusoidal Jitter Example

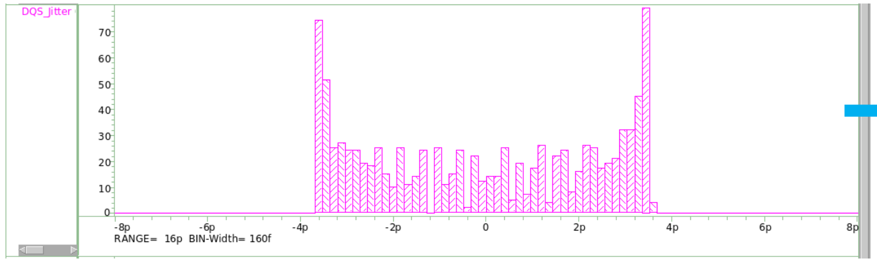
- Case 2: “High Frequency SJ”
  - SJ frequency = **475MHz**
  - Td_dqs_tree ~= 1ns
  - ~+/-4ps correlated jitter input at DQ and DQS pads
  - Result is additive jitter **not** cancelation
    - ~+/-7ps effective jitter at the Slicer

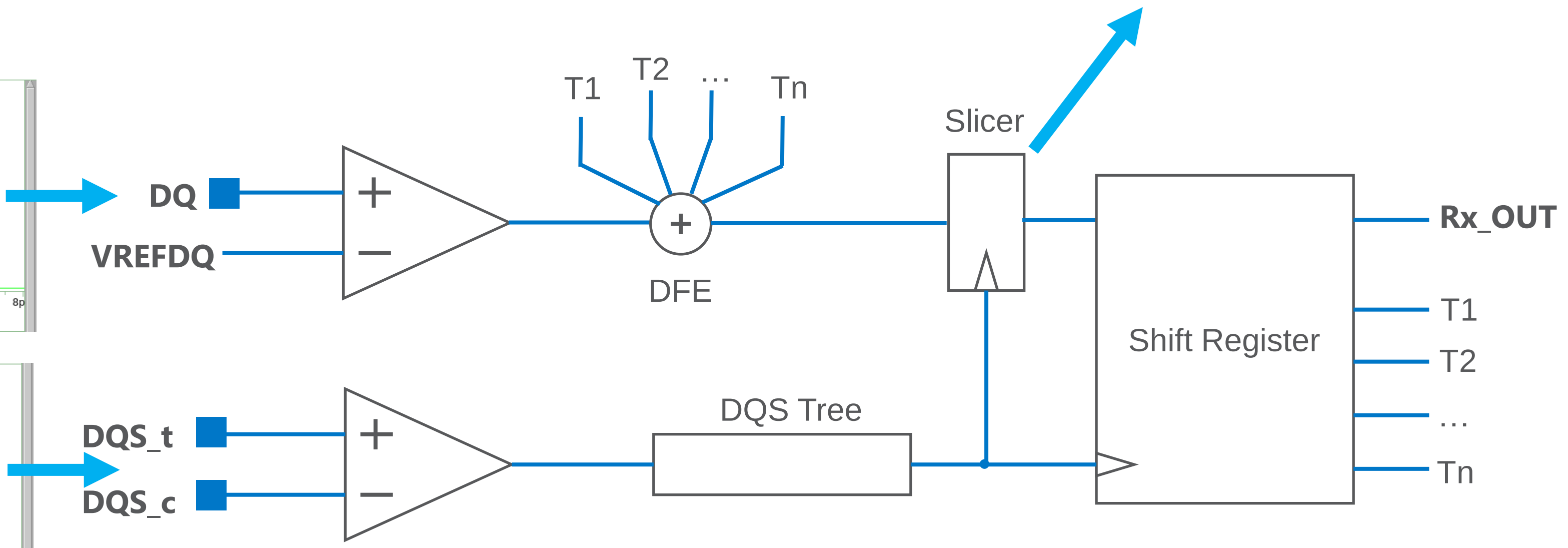
Effective Jitter Histogram at Slicer



Input Jitter Histograms



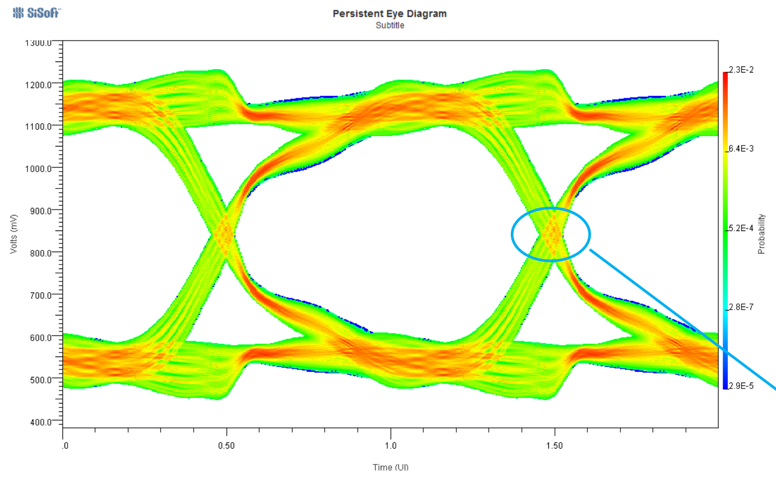




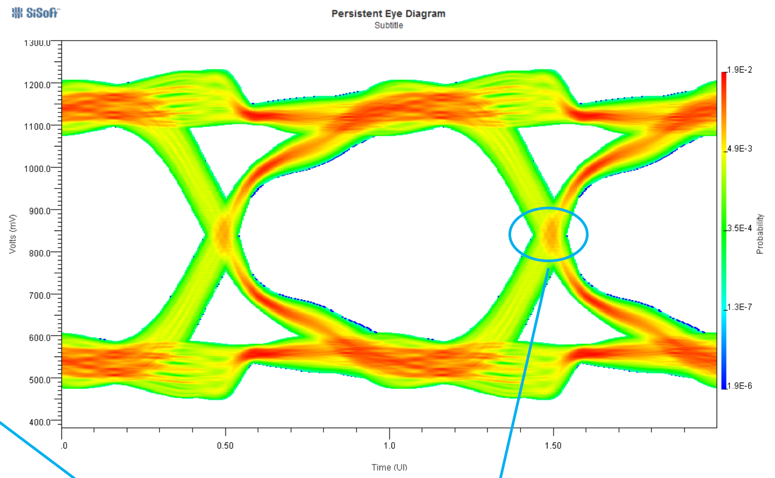
Using Existing IBIS-AMI Jitter Parameters

- 2-Rank DDR5 system, 4Gb/s, +/-4ps SJ

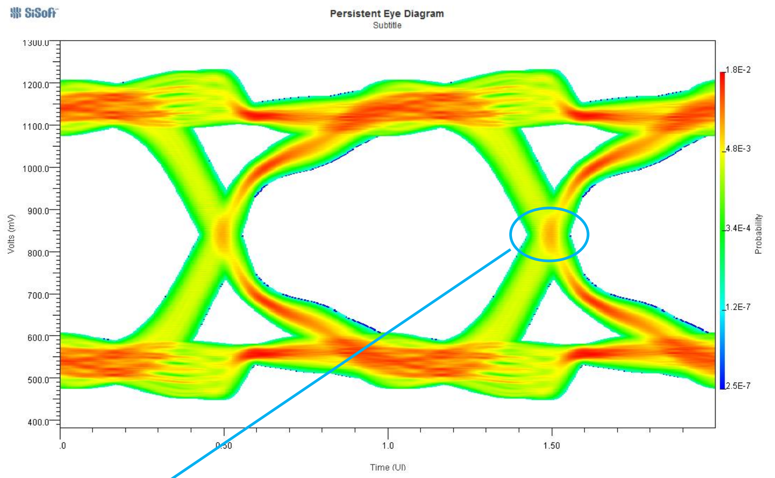
No Jitter



Tx_Sj=4ps

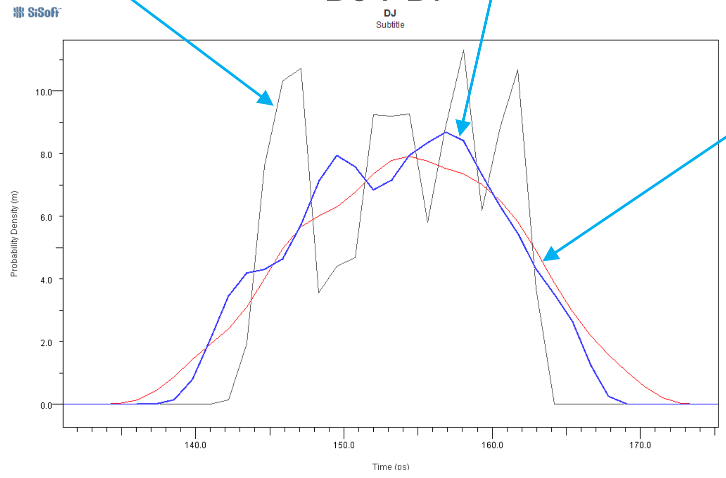


Tx_Sj=4ps +
Rx_Clock_Recovery_Sj=4ps



- DQ / DQS Jitter that becomes uncorrelated at the Slicer
  - DQS Jitter modeled with Rx_Clock_Recovery_Sj
  - Jitter cut off frequency of the DRAM Rx needs to be known

DJ PDF



No Jitter

Tx_Sj=4ps

Tx_Sj=4ps +
Rx_Clock_Recovery_Sj=4ps

Summary and Future Discussion

- Summary
  - Challenges exist to accurately model jitter in forwarded clock IBIS-AMI models
    - Due to separate paths for DQ and DQS
  - Two modeling approaches under consideration
    - CDR + Existing jitter parameters
    - clock_times as an input vector
  - Jitter that appears correlated at DQ / DQS pads can become uncorrelated at the Slicer
    - Depends on the frequency of the jitter and electrical length of the DQS tree
- Future Discussion Topics:
  - What is the frequency content of the system jitter?
    - Is it low enough to where this is a don’t care?
  - Would using existing jitter parameters be accurate enough?
    - Can inputting clock_times be a better approach?
    - Other possible approaches? Possible two step analysis approach?

Micron®