2014 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION INDIA
Sept 25-26, 2014
Hotel Park Plaza
Bangalore, India

SERDES Rx CDR Verification using Jitter, Spread-spectrum clocking (SSC) stimulus

Somasunder Sreenath, Raghuram Kolipaka, Chirag Shah, Parag Lonkar

Cadence, Bangalore

***Abstract*— Most SERDES standards, e.g USB3.0/PCIE/MPHY/SATA/DP, need a Clock-Data-Recovery (CDR) module to recover clock information from the incoming data-stream in order to sample the data. In some implementations, the block may have Analog and Digital components where part of the circuit is Analog (typically the Equalizer, phase-interpolator, Samplers etc) and the CDR algorithm is implemented in digital RTL. As a first step, the architecture level analysis is done in tools like Matlab(Simulink)/Octave. This is followed by implementation of the analog and digital blocks. At some point, the implementation diverges from architecture due to practical limitations/better understanding/minor issues in the architecture that need to be ironed out. The implemented Analog and Digital RTL of the CDR need to be verified to ensure that they meet the jitter tolerance and can track the frequency offset in ppm(parts-per-million) and Spread-spectrum clock(SSC) limits set by the protocol. In this paper we present the methodology to verify the CDR in the conventional Digital verification flow. The Analog portion of the design can be modelled in Verilog and/or SystemVerilog. The Verification environment is developed using HVL (we have both UVM-SV and UVM-e flavours on different designs).**

In this paper we present a methodology to verify SERDES Rx CDR with jitter and spread-spectrum clocking stimulus. The paper will discuss in detail the methods of adding various types on jitter in a SV module that can be re-used across various protocol standards (IP designs). The sign-off criteria for the simulations are also discussed. Simulation results from recently verified design IP(s) are also presented in detail. In addition to the CDR, the methodology also helps in stress-testing the Elastic Buffer (EB) in protocols like USB3.0/PCIE/SATA. Performance of the EB depends on the input Clock Tolerance Compensation (CTC) pattern (usually called as SKP) and the difference in the internal clock frequency with respect to the CDR recovered clock frequency. Accurate modelling of the jitter, ppm offset and SSC allows testing of the EB functionality limits.

Keywords—CDR Verification; Clock-Data-Recovery; jitter; ssc; ppm; USB3.0; PCIe; SATA;CTC

I. INTRODUCTION

The High Speed SERDES circuits are getting complex resulting in increased complexity of the verification environment. The CDR portion of the SERDES is one of the most critical parts and hence verifying it thoroughly is an involved task. The CDR verification goes beyond the realm of data integrity check across the PHY inputs/outputs. CDR specifications describe the expected jitter performance in terms of jitter tolerance characteristics across frequency (deterministic jitter) with jitter budgeting for the various types of jitter. In addition, the support requirements for Spread-spectrum clocking (SSC) are also common across various standards.

This paper will discuss in detail the various forms of Jitter, SSC and the methodology to stress the CDR with the desired stimulus. The methodology to incorporate and model jitter, SSC in the Verification Environment is presented in detail. The simulation results for the various scenarios are also discussed.

In order to verify the CDR functionality, following critical points are considered. The PLL model inside the Design Under Test (DUT) needs to model SSC as needed by the various protocols. From a Verification environment perspective, the input clock used in the Serial-Line UVC (which drives the serial differential lines) should be super-imposition of various types of Jitter (Deterministic sine-wave Jitter, DJ; Random Jitter, RJ), ppm offset, SSC with Down-spread. There should be provision for adding multiple-tone DJ to stress the CDR loop. In this paper the two methods of generating DJ, Absolute and Period Jitter, are discussed in detail. In addition to this, the Phase-Difference/Offset of the UVC clock and PLL clock needs to be varied to stress the CDR with worst case scenarios.

The signoff criteria on the verification include the following. Firstly, data-integrity check at the output of the CDR. A scoreboard is implemented to compare the serial and parallel data for checking correctness. In addition to this, the recovered clock sampling position is checked to be within expected tolerance, based on design margins, with the ideal sampling position at the centre of the Eye. These checks are enabled for the entire regression, the coverage space of which is described in detail in the Results section of this paper. In addition to this, specific tests are run to measure the bandwidth of the CDR by giving appropriate phase steps and varied data patterns.

In addition, regressions are run across combinations of ppm offsets, jitter frequencies, jitter amplitudes, phase offsets, SSC (if applicable) to extract/verify the jitter tolerance of the design to the desired granularity. The setup also allows for robust verification of the USB3.0/PCIE/SATA Elastic Buffers for Clock Tolerance Compensation (CTC) by providing the various input scenarios to stress the design.

II. CDR OVERVIEW

Many SERDES protocols (USB3.0, PCIe, SATA, DP, MPHY) use the embedded clock. i.e., there is no clock signal supplied to the receiver. The clock at the far-end is recovered from the incoming data itself. The data t driven on serial lines generally follows encoding (8b-10b, 128b-130b,..) which guarantees minimum transition density. The block diagram of the basic CDR architecture is shown in Figure 1.

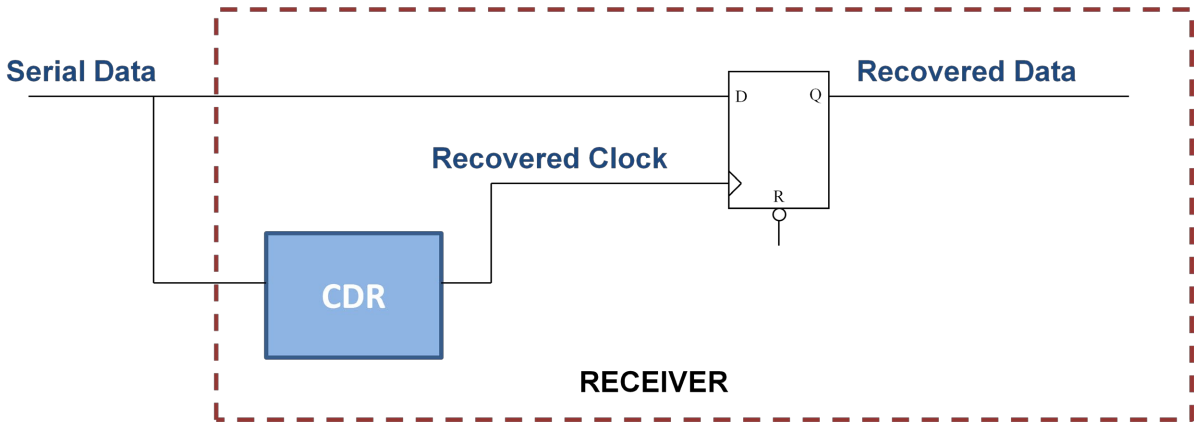

Figure 1: CDR block diagram

The CDR recovered clock needs to track the Jitter, ppm offset in the incoming serial data in order to maintain data integrity. The Jitter tolerance graph of the CDR, Figure 2, exhibits Low pass characteristics with corner frequency (f_c) around 5-10 MHz range for most protocols. The Jitter Tolerance (JTOL) is higher for low-frequency Jitter as the CDR loop tracks the jitter within the bandwidth. In order to guarantee the JTOL performance of the CDR, an exhaustive set of regressions need to be run. This is discussed in the Results section of this paper in detail.

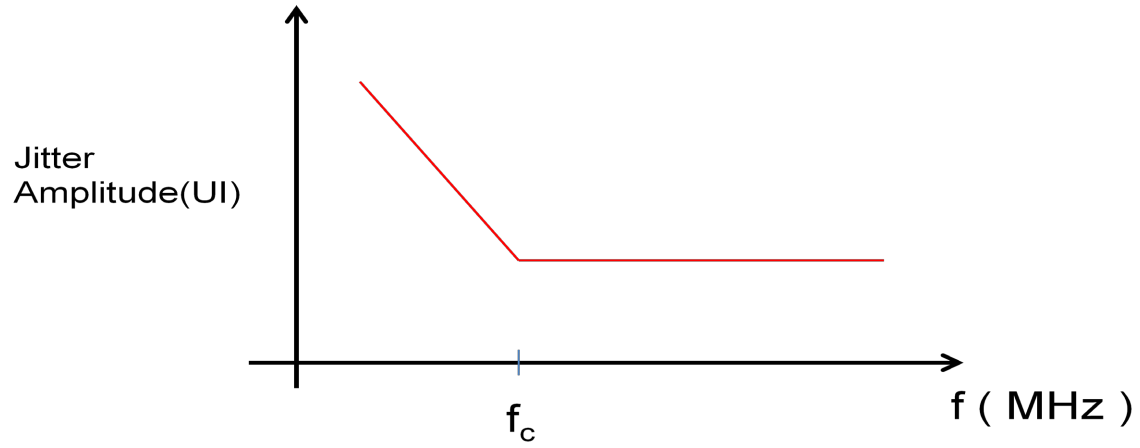

Figure 2: CDR Jitter tolerance

III. VERIFICATION ENVIRONMENT

The verification environment (Figure 3) makes use of a Specman UVM-e based HVL environment to verify the design. There is a SystemVerilog(SV) module to generate a clock with Jitter, ppm offset, SSC and random phase offset. The generated clock is used in the UVC to drive the traffic on the Receiver serial lines. An exhaustive regression is run to determine the Jitter Tolerance (JTOL) across various frequencies and Jitter Amplitudes. Also, simulations are done with multiple tones, one at low frequency say 1MHz and another at high frequency of say $F_s/2$ (which is used to account for the high frequency switching activity in the design and model the Inter-Symbol Interference ISI). The environment also generates stimulus to check for the stringent lock time conditions in MPHY based on Sync-length capability and lock-time/jitter tolerance with SSC and jitter in USB3/PCIE/SATA protocols. The environment checks the data integrity and the accuracy of the recovered clock in the design with the ideal sampling point based on design margins.

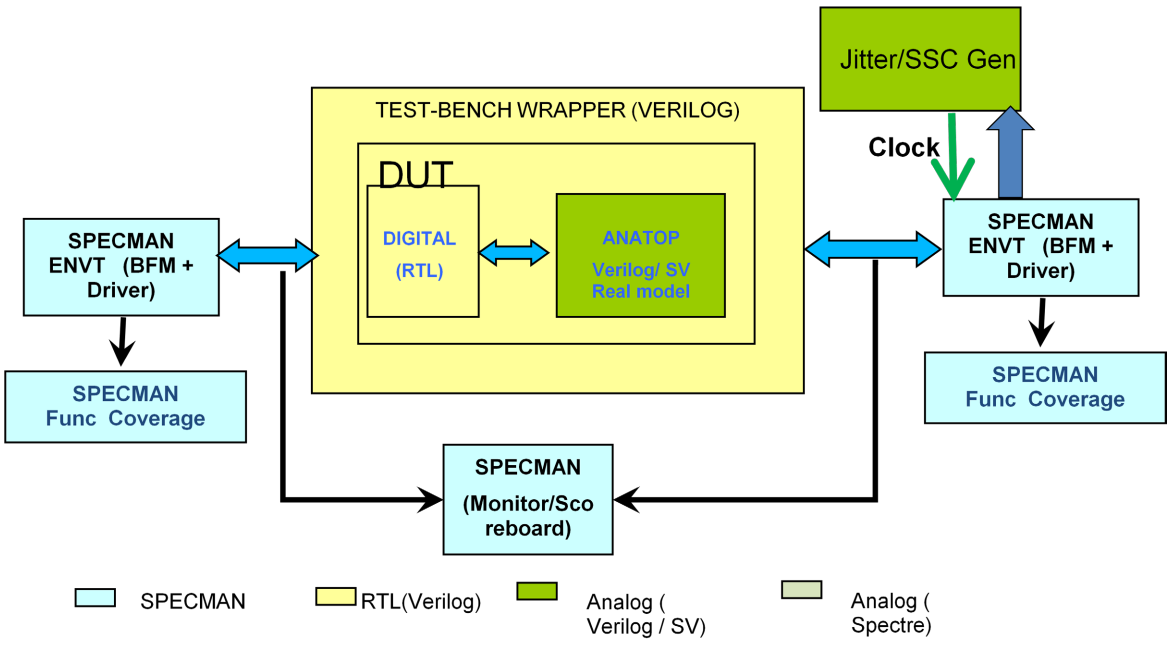

Figure 3: Verification environment architecture

In order to do the simulations in an accurate fashion, the simulation resolution needs to be 1 fs (Femto seconds) to have fine granularity of the ppm offset and Jitter addition. The analog model (including the PLL) needs to operate at 1 fs resolution.

In addition, to gain confidence on the implementation, we also run simulations by replacing the analog portion with the Spectre netlist and run the critical tests in a mixed-mode simulation using the same setup and automated checking of data and clock position. This however is out of scope of this paper and is not discussed.

IV. VERIFICATION METHODOLOGY

This section describes the Verification methodology used to verify the CDR in detail. Below are the things that are considered for verifying the CDR,

A. Clock Generation

Clock generation unit is the important component of the environment that is responsible for generating the clock to stress the CDR with all combinations of inputs. The Clock Generation block incorporates the following,

- Jitter – Deterministic (DJ) sine-wave jitter and Random Jitter(RJ)
- SSC – Spread Spectrum triangular wave with down-spread
- PPM offset on the clock with respect to the internal pll clock

B. Deterministic Jitter(DJ)Flavours

- **Period jitter**

  The Period Jitter is simpler to inject onto any clock. The method of adding period jitter to a clock is described below. To start with, a sampling clock having period equal to UI width of the underlying protocol is used, for USB3.0 it is 200ps, PCIe Gen1 it is 400ps, PCIe Gen2 it is 200ps and so on. A sine wave of desired Jitter Frequency is generated with the necessary Amplitude (AMPL). This results in a sine jitter with range +AMPL to –AMPL. At every posedge of the sampling clock, the sine wave amplitude is sampled and stored in a variable (jitter_val). This variable is then used to alter the period of the sampling clock, making the variable period equal to the period of sample clock with jitter_val added to it. As a result, the jitter_clk period varies according to the period jitter added through jitter_val.

  Basic Equations & Code:

```
always @(posedge jitter_clk) begin
        jitter_val = AMPL * sin(2π*freq*time_sec);
end

forever begin
        var_period = UI/2 + jitter_val/2;
        jitter_clk = #(var_period)  ~ jitter_clk;
end
```

  The following math_package.sv calls SV DPI to get sine & cosine wave functions used in above code.

```
package math_pkg;
        import "DPI" pure function real cos (input real rTheta);
        import "DPI" pure function real sin (input real rTheta);
endpackage : math_pkg
```

  The main advantage of Period Jitter is the ease of implementation and lesser number of parameters to handle whereas there are few limitations which make it less effective, like:

  - Slow ppm drift with time: Inconsistent sample step sizes during +ive half and –ive half sine wave cause ‘Undesirable Jitter Accumulation’ after each cycle. To overcome this to a certain extent, the method employed is to keep accumulating the jitter over one Sine Wave cycle and then subtract this residual value from the first sample of next cycle. Thus making sure not to carry forward this residue in the next cycle and keeping the next sample as close to zero as possible, Figure 4. This results in difference in number of samples in each half and ultimately causing residual jitter accumulation after one complete cycle.

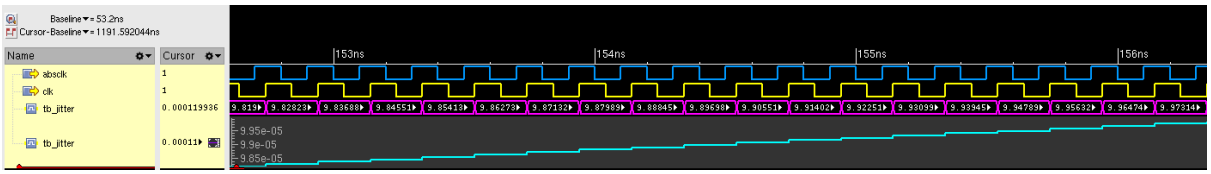

Figure 4: Variable Step Sizes within a sine wave cycle.

- Inconsistent Jitter Addition: Jitter applied in 2 consecutive Sine-wave cycles may not necessarily be always same as shown in Figure 5. The signals in the wave indicate:

  Residue: Jitter Accumulation; Zero_Cross : Zero Crossing Points of sine wave
  tb_jitter: Uncompensated Jitter wave; tb_jitter_comp : compensated jitter wave.
  tb_jitter_comp = First Jitter Sample of present Cycle – Residue (jitter accumulation of last cycle: mostly non zero)
  cntPtoN/cntNtoP : num of samples taken in Positive & Negative half cycle (Which are 2499 and 2501 respectively)

- It is not easy to add multi-tone jitter using this method as the residue correction function becomes complicated

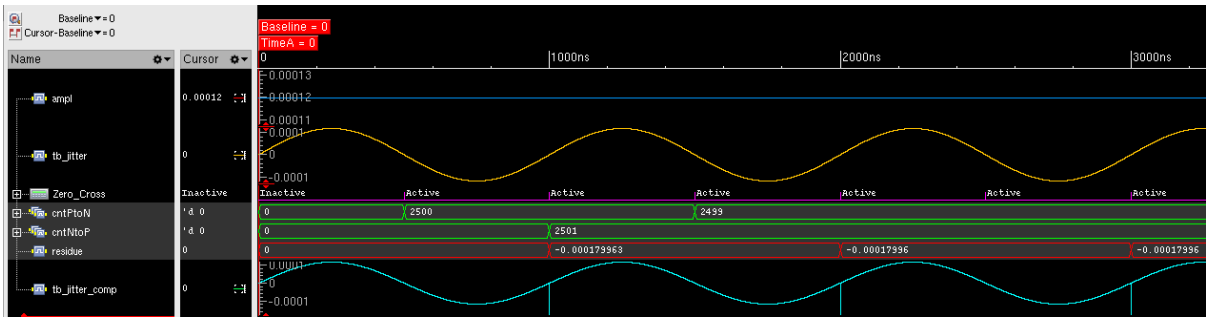

Figure 5: Shows Jitter Accumulation after each cycle and Number of Samples taken during Positive and Negative half of sine wave.

- **Absolute Jitter**

Absolute Jitter can be introduced in the clock using the following technique. In order to generate Absolute Jitter on a clock, the clock rising edge needs to be varied in accordance with the Jitter function intended. The sine-wave function traverses both positive and negative excursions thereby making it unsuitable for addition of Absolute jitter. The function 1 – cos(x) , Figure 6, has the similar characteristics of the sine-wave but the function is always positive. This enables addition of jitter using transport delay.

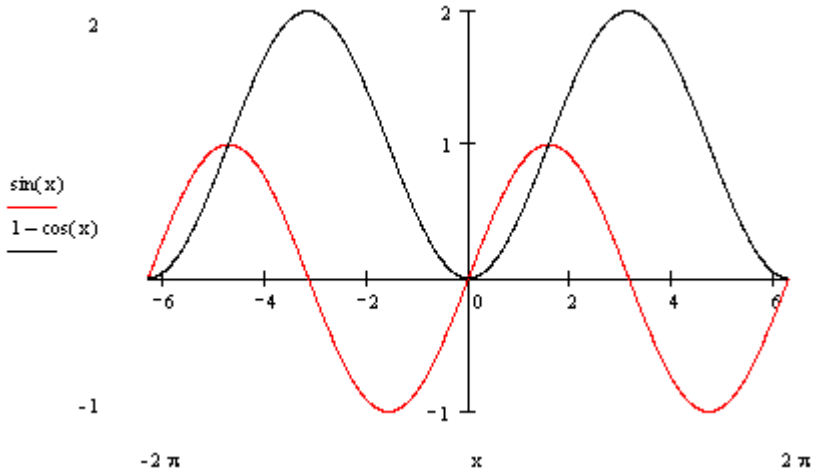

Figure 6: Absolute jitter vs. Period Jitter

The expression for the same is

*jitter_primary = (UI/2) * (DJpp1_UI * (1-cos(2.0*π*(jitter_freq))*(time_s))));*

Similarly, we can add multi tone jitter by adding the jitter contributions from the various tonal frequencies using the formula above. jitter_primary, jitter_secondary are jitter contributions are two different frequencies.

delay_jitter = 0.002+ (jitter_primary + jitter_secondary + ...);
jitter_clk <= #(delay_jitter) abs_clk;

The simulation timescale used is 1 ns/ 1fs. The component 0.002 is equivalent to 2 ps additional delay to make sure the delay is always positive.

C. PPM(Parts Per Million) Offset and SSC :

Spread Spectrum Clocking in case of USB3 and SATA have a modulation frequency of 30 KHz to 33 Khz , with a Frequency Down-spread of 4000 and 5000 ppm. SSC and ppm offset are embedded on the absolute clock before injection of jitter.

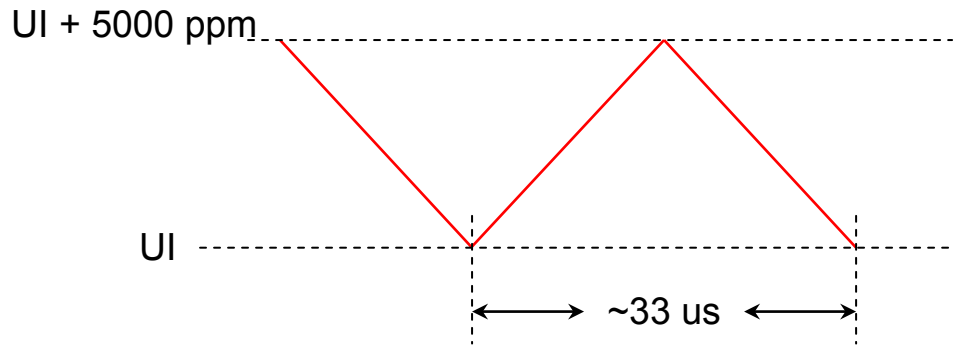

Figure 7: SSC with ppm offset

ssc_os = SSC varying in 1000 steps (First increasing then decreasing in the loop)
var_period = UI/2 +(offset_ppm/(10^6)+(ssc_os)) ;

Below are the typical PPM offset and SSC values acceptable for different protocols,

- Ppm offset – Frequency variation across 2 sides of the link for example., 300 ppm : 300* Freq/10^6
- SSC – Modulation rate : 30 to 33 kHz
  Frequency Deviation : -4000 to -5000 ppm (DOWN-SPREAD)
- USB3 : +/- 300 ppm offset, upto -5000 ppm SSC
- PCIE (common refclk) : +/- 300 ppm offset
- PCIE (Separate refclk) : +/- 300 ppm offset, upto -5000 ppm SSC
- SATA : +/- 350 ppm offset, upto -5000 ppm SSC

D. Elastic Buffer(EB) and Clock-Tolerance Compensation (CTC) :

Performance of the EB depends on the input Clock Tolerance Compensation (CTC) pattern (usually called as SKP) and the difference in the internal clock frequency with respect to the CDR recovered clock frequency. Accurate modelling of the jitter, ppm offset and SSC allows testing of the EB functionality limits. The method of adding jitter, SSC and ppm offset allow generating worst case movement of the read/write-pointers in the EB. In addition to this, the SKP position and insertion in the Verification environment is varied to test the EB in a robust manner.

In USB3 specification, the SKP Ordered Set (SKP OS), which consists of 2 SKP symbols K28.1 K28.1, is inserted on a average every 354 symbols. In the scenario of worst case instantaneous ppm offset between the internal clock and the recovered clock, the EB pointers can move apart by ~2 symbols. Adding or removing the SKP OS helps the design to maintain the EB in the nominal hall-full position (which is the default mode of operation of the EB). However, the EB should be able to starve or hold upto 8 symbols as in the worst-case scenario 4 SKP OS may be inserted in the incoming data-stream every 1416 symbols. The Verification

environment is capable of generating all possible traffic in conjunction, which in conjunction with the SSC,jitter, ppm offset UVC clock enables testing the EB functionality in a robust manner.

V. RESULTS

A. *Scenario1*

Simulation result with 5000 ppm offset, 1MHz single tone sine-jitter are shown in Figure 8,9. The sample simulation is run at a data rate of ~5.8 Gbps data-rate. The loop dynamics are captured in Figures and explained below for reference.

- Loop tracking the offset and sine wave(LOOP_DYN1)
- Recovered clock at the centre of the eye(REC_CLK_POS)
- ppm offset – Recovered clock position sweeps like a ramp across quadrants(LOOP_DYN2)

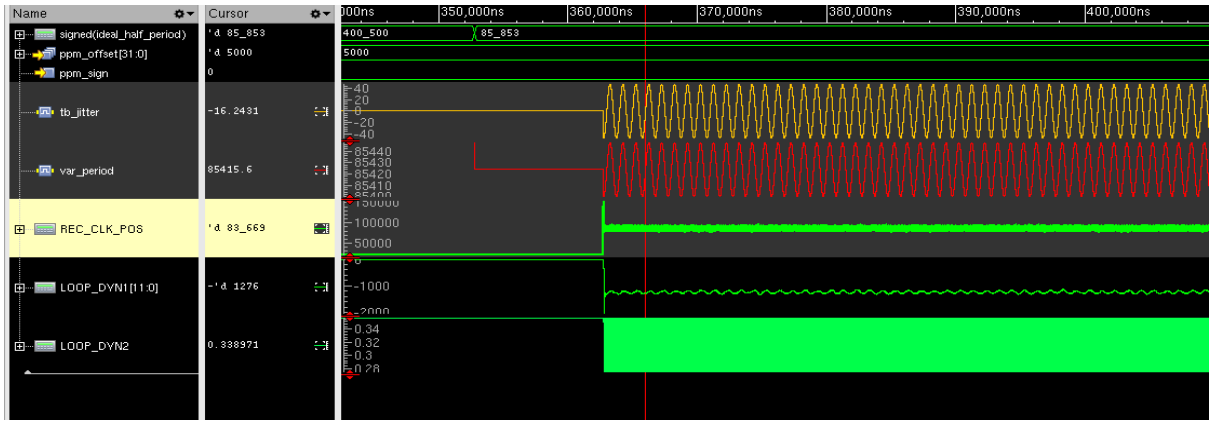

Figure 8: Simulation results for CDR locking

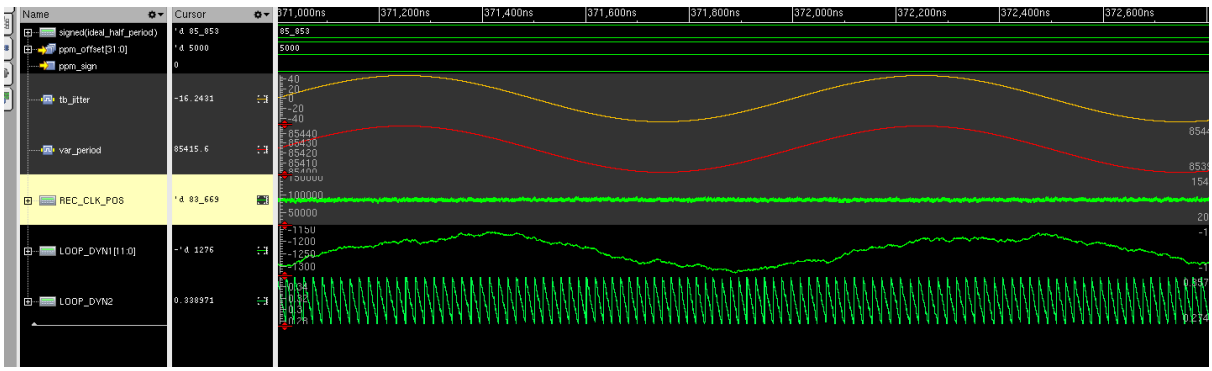

Figure 9: Simulation results for CDR locking- Zoomed version

B. *Scenario 2*

Simulation result for CDR operating at 5 Gbps data rate, -5000 ppm SSC down-spread, single tone DJ (1*UI @ 1Mhz) is shown in Figure 10. In this result, the CDR loop internal dynamics is tracking the DJ super-imposed on the SSC signaling. Also, the signal ***c0p*** represents the REC_CLK_POS (recovered clock position) which is at the centre of the eye-diagram indicated by ~0.1 ns real valued signal.

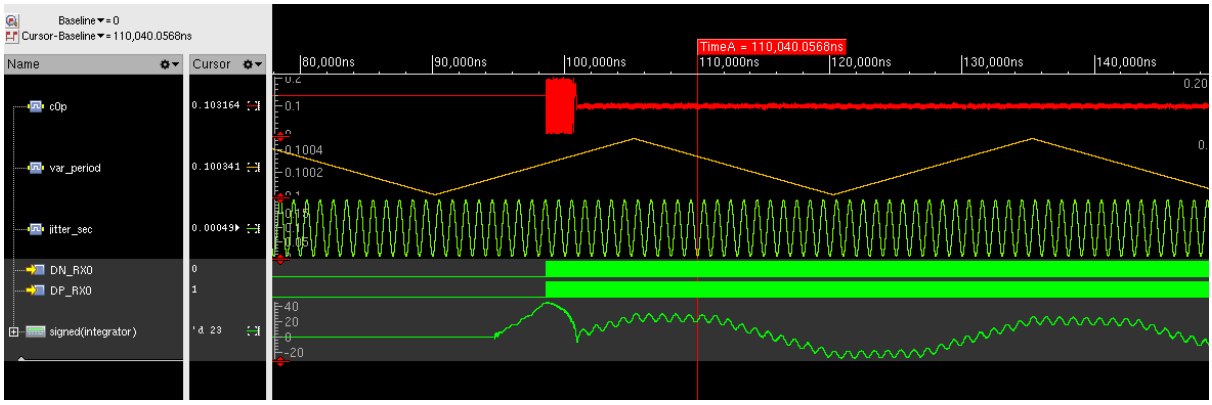

Figure 10: Simulation results for single-tone jitter with -5000 ppm ssc down-spread

C. Jitter Tolerance

Regressions were run for around 100000 seeds with various combinations of ppm offset, jitter(no jitter, single-tone jitter, 2-tone jitter, 2-tone jitter+ RJ), Jitter frequencies(1,2,4,6,8,10,100,500,1200Mhz),Jitter Amplitude(steps of 0.1UI, better resolution of higher jitter frequencies), SSC Modulation rate(30,33Khz), SSC Frequency Deviation(No SSC, -4000ppm, -5000ppm), SSC Phase Difference, Data Pattern(TSEQ,TS1, TS2, CP0, Clock pattern(D10.2),MPHY Sync pattern(D10.5,D26.5), SATA ALIGN, PRBS7, PRBS31,CRPAT,CJTPAT, Random data)

For each Jitter frequency, the jitter amplitude is swept in small steps of desired resolution (in terms of UI) to determine the failure point in terms of data integrity and recovered clock position check. In order to improve the confidence on the results and remove outliers in the regression, multiple simulations for the same jitter amplitude. Figure 11 shows the jitter tolerance graph obtained with these regressions. The simulation JTOL is plotted with the specification JTOL to compare the design performance.

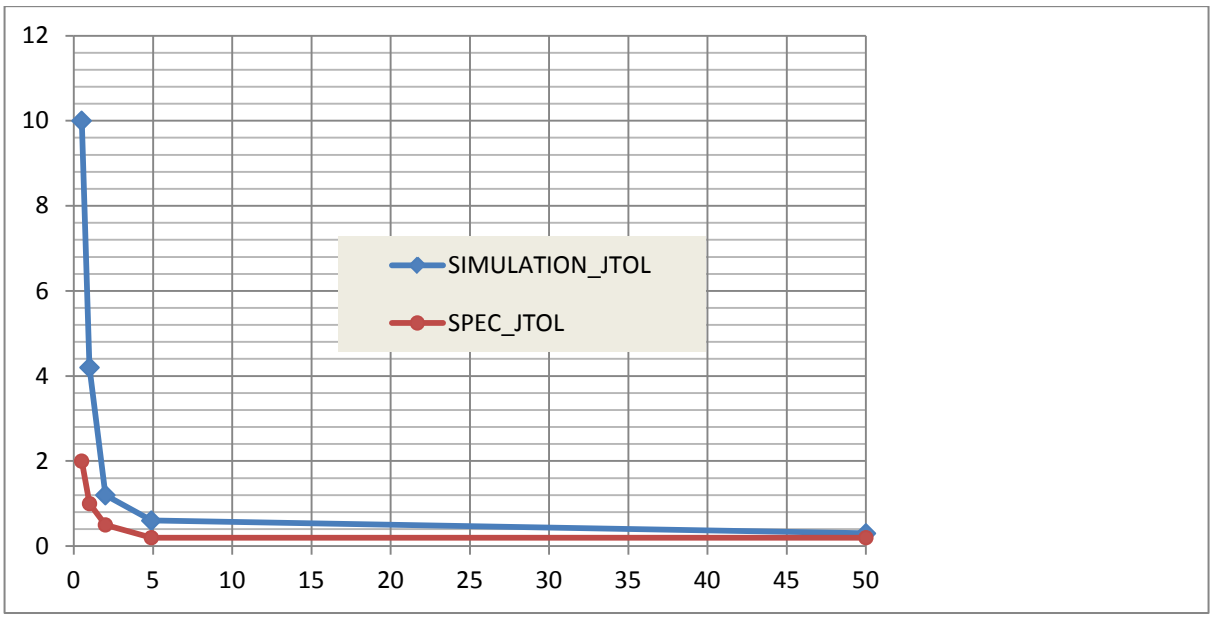

Figure 11: Jitter tolerance graph

REFERENCES

[1] MIPI MPHY v3.0 r03 spec

[2] USB3.0 Spec : 1.0 June 6, 2011

[3] PCIE : PCI Express® Base Specification Revision 2.1 March 4, 2009