Functional Verification of CSI-2 Rx-PHY using AMS Co-simulations

Ratheesh Mekkadan, Advanced Micro Devices, Inc., Bangalore, India
(ratheesh.mekkadan@amd.com)

***Abstract*— The physical layer of the MIPI-camera serial interface 2 (CSI-2) protocol, MIPI DPHY, is an anolog mixed signal (AMS) IP as it consists of analog blocks (e.g., receiver blocks) and digital blocks (e.g., finite state machines). The digital blocks are synthesizable modules, while behavioral RTL models are developed for the analog blocks for IP level verification. A universal verification methodology (UVM) based constrained random verification environment is used to exhaustively verify the IP.**

However, certain functional modes, such as data transfer mode in CSI protocol which makes use of LP (low power high swing) sequencing followed by HS (low swing differential) signal levels, cannot be effectively verified using register-transfer level (RTL) simulations. In the digital simulations, both the LP and HS level signal states are represented by '1' (logic high) or '0' (logic low), and proper functionality of the HS receiver and LP receiver cannot be validated. Further HS termination within the PHY, which gets enabled during HS burst/data reception, has no effect in RTL simulations.

To overcome the above challenges, an analog mixed signal (co-simulation) environment is setup for the functional verification of the CSI-2 PHY. Also, the extensive UVM based RTL simulation environment can be reused as is with a few additional components added for co-simulations.

Keywords—Analog Mixed Signal; MIPI; CSI-2.

I. INTRODUCTION

The MIPI camera serial interface 2 (CSI-2) specification defines a standard interface between a peripheral device (camera/transmitter (Tx)) and a host processor (receiver (Rx)) for mobile platforms. This is depicted in Figure 1. As compared to existing camera interface solutions, that were either parallel interfaces (which made scalability difficult) or serial interfaces (incompatible and proprietary), the MIPI CSI-2 has emerged as a standard, scalable, low-power, high-speed, cost-effective interface that supports a wide range of image applications.

The physical layer of this interface is the MIPI D-PHY, which provides the connection between master (Tx) and slave (Rx). This PHY layer is composed of up to four data lanes and two clock lanes, and supports a high-speed (HS) signaling mode for fast-data traffic and a low-power (LP) signaling mode for control sequencing. The HS mode supports a bit rate of 80 to 1500 Mbps per lane, while the maximum data rate in low-power mode is 10 Mbps.

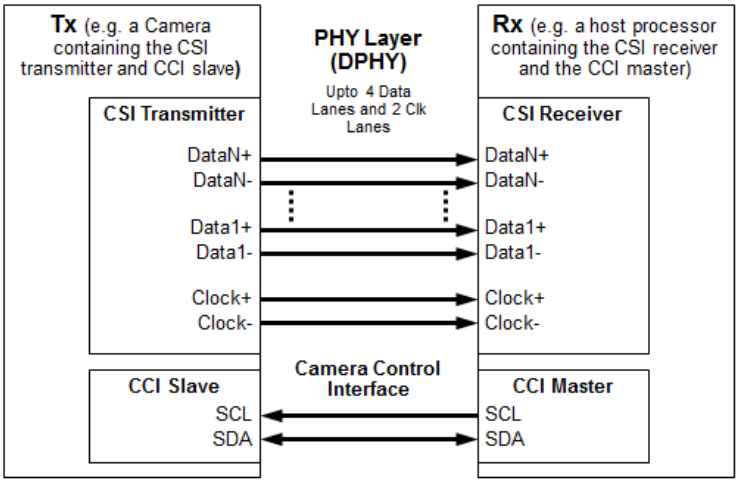

Figure 1: Components of the CSI-2 protocol.

A. CSI-2 PHY – An AMS IP

The CSI-2-PHY (or DPHY) is an example for an AMS IP. Each of its lanes (either clock or data) of the Rx-PHY has analog blocks, such as:

a) LP and HS Rx blocks

b) Termination block for HS mode

c) Circuits for determining “clk-miss detect” and “clk settle” parameters

The digital blocks in each lane are:

a) Finite state machines (FSM) that control the logic

b) Clock division blocks

c) De-serializer and synchronization

d) Design for testability (DFT) and observation blocks

This is depicted in Figure 2. At IP and SoC level simulations, RTL behavioral models are developed and used for all of the analog blocks.

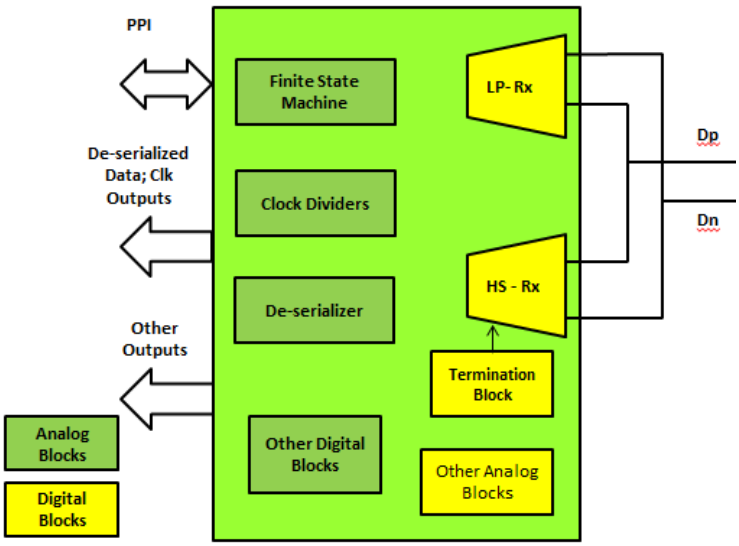

Figure 2: Block diagram of DPHY showing internal blocks.

B. LP and HS mode signalling

The LP mode transmission is through single-ended high swing signaling (typically 1.2V high and 0V low) while HS mode is differential low swing signaling (typically 300mV high, 100mV low, and 200mV common mode voltage). Thus there are two possible high-speed lane states: Differential-0 (HS-0) and Differential-1 (HS1) and four possible low-power lane states (i.e., LP-00, LP-01, LP-10, LP-11), as seen in Table 1. LP-receivers interprets high-speed line states as LP-00

Table 1: HS, LP line states.

Line State	**Line Voltage Level**	
	Dp	**Dn**
HS-0	HS Low	HS High
HS-1	HS High	HS Low
LP-00	LP Low	LP Low
LP-01	LP Low	LP High
LP-10	LP High	LP Low
LP-11	LP High	LP High

C. Conventional RTL Verification and its Challenges

For data transfer in the CSI-2 protocol, the DPHY HS mode is used, with data being serialized in the transmitting PHY and de-serialized in the Rx-PHY. The sequencing as seen for data transmission is as below. During the HS mode, the CSI-2 PHY design enables line termination.

LP-11 (StopState) ➔ LP-01(HS-Request) ➔ LP-00(HS- Prepare) ➔ HS-0 ➔ [HS Sync Pattern -- HS Payload Data -- HS Trailer] ➔LP-11 (StopState)

An exhaustive UVM-based environment is created and used to validate the functionality of the design. This creates data bursts of random length and random data, error injection into valid line LP, and HS sequences during entry into or exit from HS burst mode or control modes or into the HS Sync Pattern. However, the RTL simulation environment is inadequate with respect to the below items:

a) Functionality of the LP-Rx and the HS-Rx blocks are not verified since these blocks are represented by RTL behavioral models in IP level or SoC level simulations (these behavioral models do not model all of the analog functionality). For example, the LP and HS line voltage levels are different, but they cannot be distinguished in a digital simulation, as they are both represented as 1's and 0's.

b) Functionality of the HS line termination, which gets enabled during HS mode, cannot be validated in digital simulations.

II. AMS VERIFICATION APPROACH

To overcome the above inadequacies of the RTL simulation setup, an analog mixed signal simulation setup is created for the CSI-2 PHY. The tool used for the co-simulation is XA-VCS tool from Synopsys. The co-simulation environment has the UVM testbench as its top layer. The design under test (DUT) is configured to have SPICE description for its analog blocks and RTL for the rest of the logic. A2D and D2A rules are suitable defined to allow analog and digital blocks to interact with each other. Below are a few salient highlights of the setup:

A. Analog and Digital block partitioning for co-simulations

To resolve the inadequacies of the RTL behavioral models of analog blocks, these are replaced by their SPICE description, in the co-simulation environment. The blocks include HS-Rx, LP-Rx blocks, termination block, Clk settle, and miss circuits to measure DPHY time parameters, etc.

B. Signal Source Impedance

The HS mode within the CSI-2 Rx PHY enables termination. On the stimulus side, the testbench models signal source impedance. A 50ohm resistor is added to both the Dp and Dn differential lines during the HS mode.

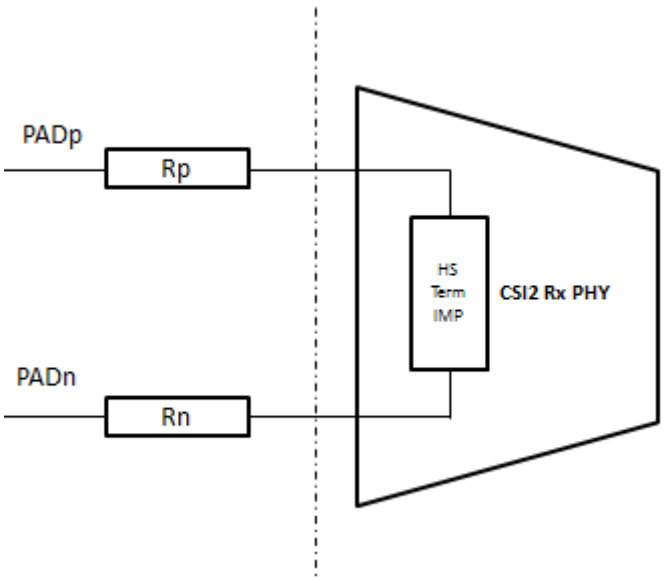

Figure 3: Signal Source Impedance.

C. Converter Block

We want to reuse the same exhaustive UVM testbench build for the RTL simulations for co-simulation. A "converter" block is inserted between the UVM testbench and the actual design, as the Dp/Dn pads need to see

dynamic change in “pad” signal voltage when transitioning between HS and LP line states. The XA-VCS command **$snps_force_volt** is used to drive voltages on the source impedance (HS mode) or directly on the “pads” (LP mode)

The digital testbench generates the required sequence (LP/HS) for entering HS mode for data transmission, random payload data and random burst length. The UVM driver component uses a testbench signal “hs_mode_en” to distinguish between generated bit sequence (consisting ‘1’s and ‘0’s) for both HS and LP modes.

Pseudo code for the converter block is as below:

```
always@(tb.hs_mode_en  or tb.pad_csi_dl_p) begin
      if(tb.hs_mode_en==1 && tb.pad_csi_dl_p==1)
         $snps_force_volt (<hierarchy>.PADPp, <hs_hi_voltage>);
      else if(tb.hs_mode_en==1 && tb.pad_csi_dl_p==0)
         $snps_force_volt (<hierarchy>.PADPp, <hs_low_voltage>);
      else if(tb.hs_mode_en==0 && tb.pad_csi_dl_p==1)
         $snps_force_volt (<hierarchy>.vpadp, <lp_hi_voltage>);
      else if(tb.hs_mode_en==0 && tb.pad_csi_dl_p==0)
         $snps_force_volt (<hierarchy>.vpadp, <lp_low_voltage>);
end

always@(tb.hs_mode_en  or tb.pad_csi_dl_n) begin
      if(tb.hs_mode_en==1 && tb.pad_csi_dl_n==1)
         $snps_force_volt (<hierarchy>.PADNn, <hs_hi_voltage>);
      else if(tb.hs_mode_en==1 && tb.pad_csi_dl_n==0)
         $snps_force_volt (<hierarchy>.PADNn, <hs_low_voltage>);
      else if(tb.hs_mode_en==0 && tb.pad_csi_dl_n==1)
         $snps_force_volt (<hierarchy>.vpadn, <lp_hi_voltage>);
      else if(tb.hs_mode_en==0 && tb.pad_csi_dl_n==0)
         $snps_force_volt (<hierarchy>.vpadn, <lp_low_voltage>);
end
```

As seen above, depending on whether we are in the HS mode (tb.hs_mode_en==1) or LP mode (tb.hs_mode_en==0), and depending on the bit sequence (1’s and 0’s) driven by the testbench, these are translated to appropriate voltage levels (HS or LP) to be driven on the pads.

D. Testbench architecture

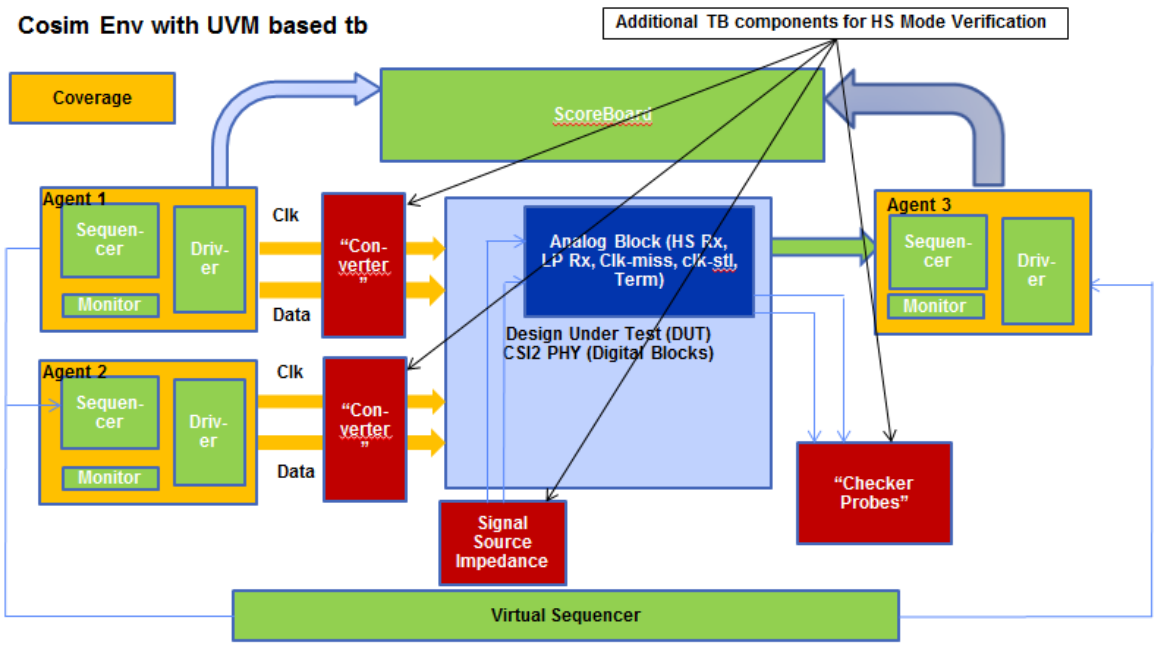

Figure 3: UVM based testbench used for co-simulations.

III. APPLICATION AND RESULTS

A. *Validating Data Transfer Mode*

Functionality of the LP Rx and HS Rx blocks are validated for "data transfer" mode which uses the LP to HS mode sequencing. The stimulus generated in the digital UVM tb is translated to appropriate voltage levels for the analog blocks. The sequencing, as seen for data transmission, is as below.

LP-11 (StopState) ➔ LP-01(HS-Request) ➔ LP-00(HS- Prepare) ➔ HS-0 ➔ [HS Sync Pattern -- HS Payload Data -- HS Trailer] ➔LP-11 (StopState)

The "converter block" ensures that the stimulus generated by the UVM testbench (digital stimulus in terms of 1's and 0's) translates to valid voltage levels, depending on HS or LP mode. **The regular "d2a" connect rules (in the co-simulation tool suite), which is generally used for "digital" to "analog" conversion, will not work here as the PADs can dynamically change voltage levels (when the line states on the PAD transitions from LP to HS and then to LP modes again).** The co-simulation environment helped validate that data generated in the UVM testbench and driven to the Rx PHY appears as expected on the PHY protocol interface (PPI) as de-serialized data. The UVM scoreboard does the checking.

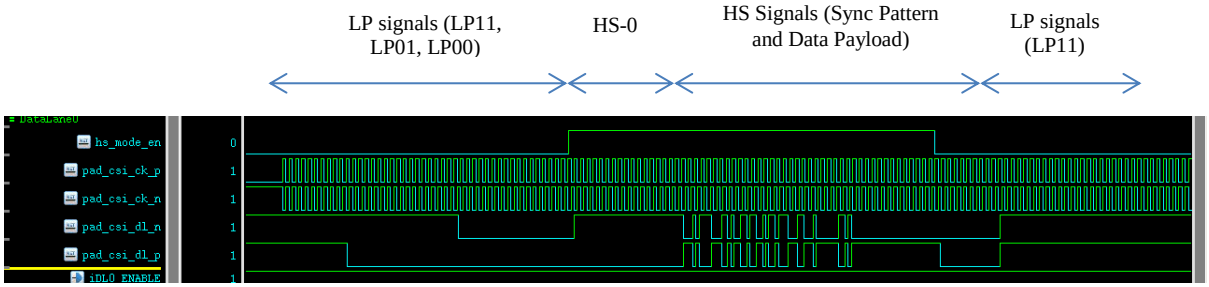

Figure 4: Stimulus from the digital UVM testbench.

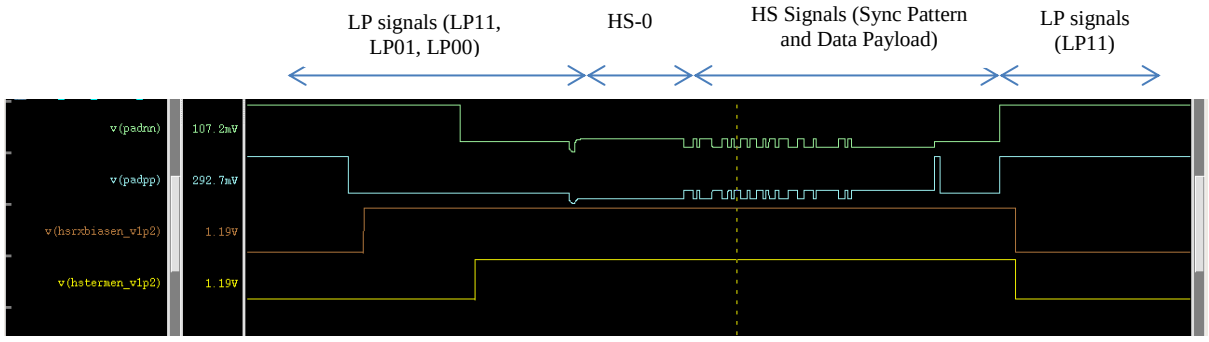

Figure 5: Actual LP and HS voltage levels at the PADs.

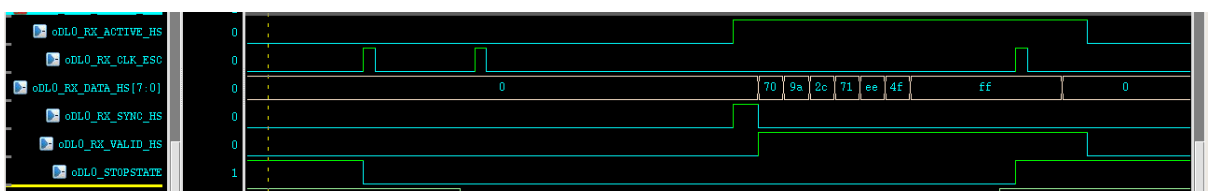

Figure 6: De-serialized data out of the CSI-2 Rx-PHY.

B. Rx response to valid voltage levels on pads

The voltage levels driven by the "converter block" onto the pads is randomly varied to ensure that various valid voltage levels (within specs) on the lines are recognized by the LP-Rx and HS-Rx blocks. Below is the pseudo code used to generate the LP "voltages" to be forced onto the pads. At the start of any simulation, these values are generated and used by the test case.

```
real  lp_hi_int_v, lp_hi_frac_v, lp_hi_v;
real  lp_low_int_v, lp_low_frac_v, lp_low_v;
lp_hi_int_v = $urandom_range(<LP_1_max>,<LP_1_min>);
lp_hi_frac_v = $urandom_range(999,1) / 1000;
lp_hi_v = lp_hi_int_v + lp_hi_frac_v;

lp_low_int_v = $urandom_range(<LP_0_max>,<LP_0_min>);
lp_low_frac_v = $urandom_range(999,1) / 1000;
lp_low_v = lp_low_int_v + lp_low_frac_v;
```

The LP-Rx block is validated to respond well to minimum low and high voltage levels specified in DPHY spec.

C. Validating configurable HS termination though control and status registers (CSRs)

The design has a configurable termination impedance block, where depending on the CSR configuration setting, the HS termination value gets set during HS data transmission mode. The design has a 7 bit CSR register which supports thermometer coding. Below is the pseudo code for the test sequence.

1) The Rx-PHY is powered-up and initialized
2) HS mode is enabled (so that the HS termination is "ON")
3) Drive fixed (HS differential level) voltages on the PADs. (e.g., PADp = 250mv, PADn = 150mv)

```
always@(Core_dft_pwr_intf.iPWRGD_PWROK) begin
   if(Core_dft_pwr_intf.iPWRGD_PWROK===1'b1)  begin
         $snps_force_volt (<hierarchy>.PADP, 0.25);
         $snps_force_volt (<hierarchy>.PADN, 0.15);
   end
 end
```

4) Iterate through the different CSR settings to configure different HS termination values. In each case, measure the PAD voltage as well as PAD current and determine the termination resistance. This checker is within the digital UVM testbench and achieves this checking by probing internal analog nodes.

```
for (int i=0; i<8; i++) begin
      //thermometer coding; valid code values are [decimal] 0,1,3,7,15,31,63,127
      Term_CSR[6:0]={ Term_CSR [5:0], 1'b1};
end

always@(<tb_signal_for_CSR_Term_cfg_done>) begin
   if(Core_dft_pwr_intf.iPWRGD_PWROK===1'b1)  begin
      #<delay_value>;
      voltage_padp   = $snps_get_volt(<hierarchy>.PADP);
      voltage_padn   = $snps_get_volt(<hierarchy>.PADN);
      current_padp   = $snps_get_port_current(<hierarchy>.PADP);
      term_impedance = (voltage_padp - voltage_padn) / current_padp;
   end
end
```

5) Sample log file and waveform dump from the simulation run:

```
UVM_INFO at  6015971: HS_TERM_IMP = 0, PADp(V) = 0.250000, PADn(V) = 0.150000, PADp
current = 0.000884 :: Term-impedance = 113.066436
UVM_INFO at  6916491: HS_TERM_IMP = 1, PADp(V) = 0.250000, PADn(V) = 0.150000, PADp
current = 0.000981 :: Term-impedance = 101.906082
UVM_INFO at  7817011: HS_TERM_IMP = 11, PADp(V) = 0.250000, PADn(V) = 0.150000, PADp
current = 0.001067 :: Term-impedance = 93.686777
```

```
UVM_INFO at  8717531: HS_TERM_IMP = 111, PADp(V) = 0.250000, PADn(V) = 0.150000, PADp
current = 0.001151 :: Term-impedance = 86.908291
UVM_INFO at  9618051: HS_TERM_IMP = 1111, PADp(V) = 0.250000, PADn(V) = 0.150000, PADp
current = 0.001242 :: Term-impedance = 80.540675
UVM_INFO at 10518571: HS_TERM_IMP = 11111, PADp(V) = 0.250000, PADn(V) = 0.150000,
PADp current = 0.001332 :: Term-impedance = 75.091791
```

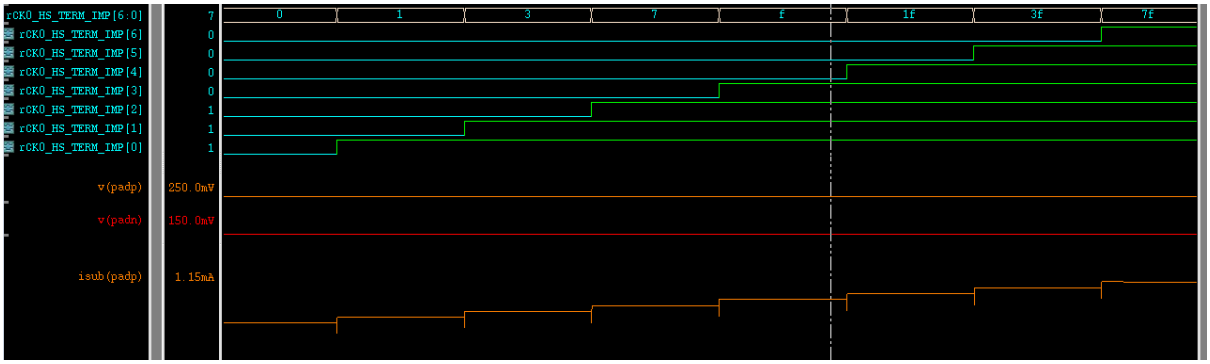

Figure 7: PAD voltage, CSR termination impedance configuration and PADp current draw.

D. Always "on" checker probes on analog blocks

Checker probes can be placed within UVM based testbench, accessing internal nodes, to validate signal timing within the analog blocks. The CSI-2-PHY design has measurement circuits to determine the "clk-settle" time, as well as the "clk-miss" time parameters in the DPHY standard. The checkers monitor the rise of the "clk-settle_start" and "clk-miss-start" signals, the fall of the "clk-settle_done," and "clk-miss_done" to determine the "clk-settle" time, as well as the "clk-miss" time. The pseudo code is as below. These checkers are always on and turn on during the mission mode simulations.

```
always @ (snps_cross($snps_get_volt(Csi_top.uPAD_CSI_CL.ickmisstart)-1.0 ,1) )
    //Detect ckmisstart exceeding 1v
    Ckmisstart_time = $time;
always @ (snps_cross($snps_get_volt(Csi_top.uPAD_CSI_CL.ickmisdone)-1.0 ,1) )
    //Detect ckmisdone exceeding 1v
    Ckmisdone_time = $time;
Ckmiss_time = Ckmisdone_time - Ckmisstart_time;
```

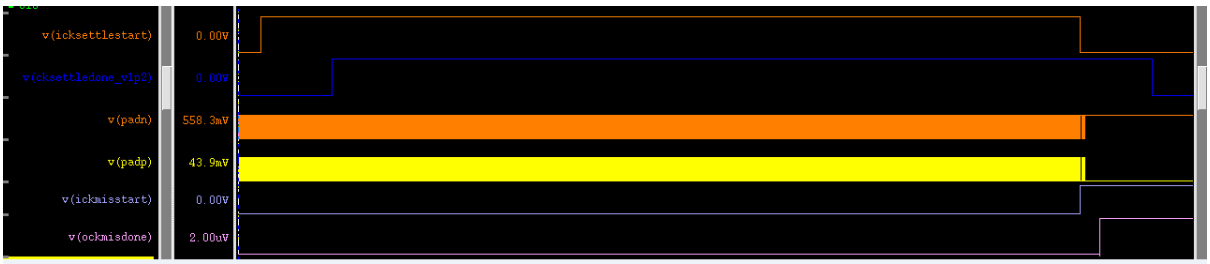

Figure 8: Probing internal nodes for "clkmiss" time and "clk_settle" time parameters.

E. Validating proper design response to corner case scenarios

1) The design is validated for data bursts of different lengths and random payload data. The UVM testbench, in conjunction with the "converter block," is used to achieve this and random data generated is driven as appropriate voltage levels on the pads. The design is also validated to respond to error induced into bit sequences.

2) Range of valid "data rates" supported by the DPHY standard for HS mode data transmission.

3) Pulse width requirement for LP mode bit pattern validated by varying the pulse width of the LP mode bit pattern in the digital UVM testbench.

F. Key issues found

1) Co-simulations helped identify an issue where an internal state machine, working on EscClk (generated based on the pad state), was found to stall and not recover. It was found that a glitch on the Lane-Enable signal could cause a short pulse width line state on the pads and this failed to generate an EscClk pulse. The state machine, which was reset to the "OFF" state by Lane-Enable deassertion, gets stuck (due to absence of further EscClk) and continues to remain in the OFF state. This is depicted in Figure 9.

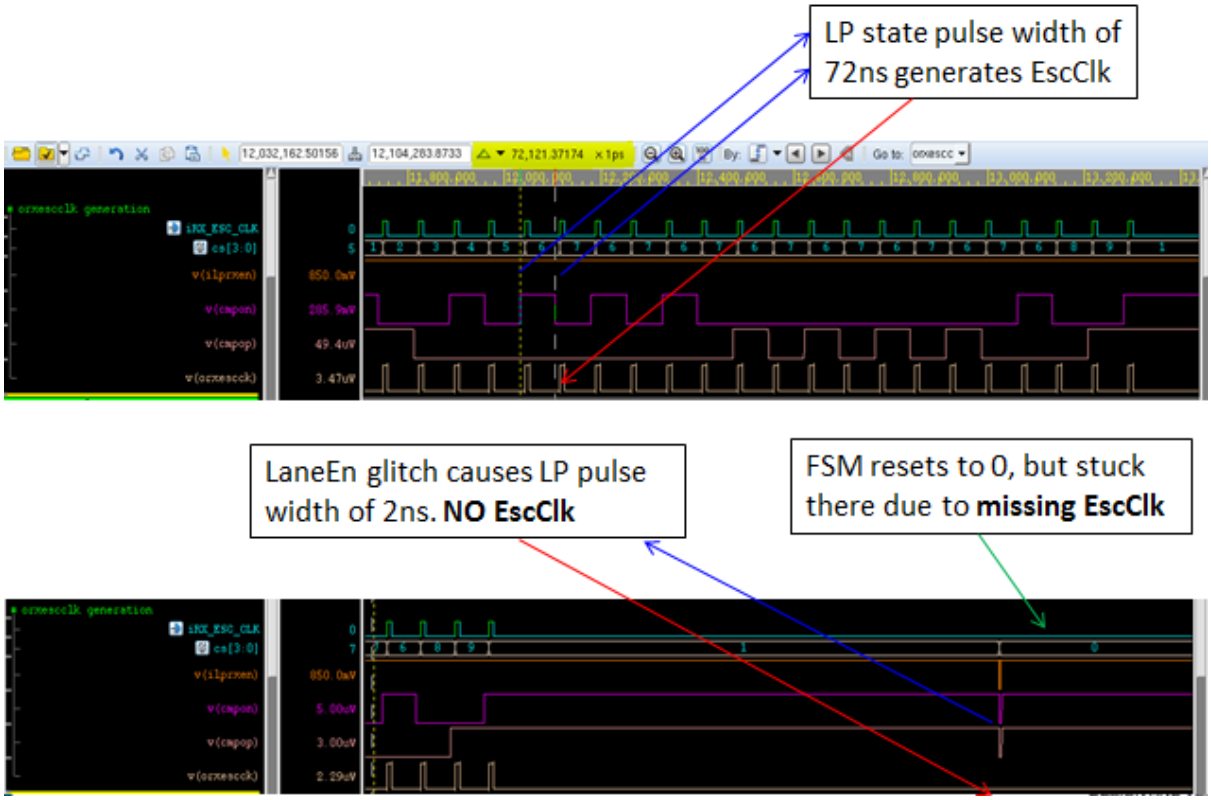

Figure 9: Probing internal nodes for "clkmiss" time and "clk_settle" time parameters.

2) Correlating RTL simulation and co-simulation results identified inaccuracies in RTL model of analog blocks in the modeling "clk-settle" and "clk-miss" parameters.

IV. CONCLUSION

A. Advantages of the co-simulation based approach in verifying the functinoality of the Rx PHY

1) The UVM tb was effectively reused in the co-simulation environment. Testbench components to mimic camera side stimulus, ScoreBoard checking, and other protocol related checks were reused as is from the verification suite developed for the RTL verification.
2) Validated the LP-Rx and HS-Rx functionality by simulating the transition between LP Mode → HS Mode → LP Mode seen during data transfer mode.
3) Co-simulations are faster than full SPICE simulations. A full co-simulation run completes in 4hours.
    - This enabled re-running a large number of tests that covered various random scenarios in the UVM tb, in the co-simulation world.

4) Automatic checking/monitoring of ‘clk-miss’ and ‘clk-settle’ DPHY parameters with the help of the “checker probes.”
5) Automatic checking of termination resistance calibration in the digital UVM testbench and indicating pass / fail criterion.

ACKNOWLEDGMENT

I would like to thank my management at AMD, Sachin Kulkarni and Vivek Sabnis for inspiring and encouraging me to publish my ideas as a DVCON paper. I also want to thank my design team and David Block in particular for the suggestions and inputs for co-simulations on the CSI-2PHY. Thanks are also due to AMD PR and Legal approvers for their feedback.

REFERENCES

[1] *D-PHY Specification*, **MIPI® Alliance , Version 1.1 – 7, November 2011.**
[2] *Camera Serial Interface 2 (CSI-2) Specification*, **MIPI® Alliance , Version 1.01.00 – 9, November 2010.**
[3] *D-PHY Physical Layer Conformance Test Suite*, **MIPI® Alliance, Version 1.1r04 – 12, June 2013.**
[4] *Discovery™ AMS: Mixed Signal Simulation User Guide*, **Synopsys, Version H-2013.03, March 2013.**
[5] *CustomSim™ XA Command Reference*, **Synopsys, Version H-2013.03, March 2013**.
[6] *CustomSim™ XA User Guide*, **Synopsys, Version H-2013.03, March 2013**.
[7] Synopsys Solvnet. [Online]. Available: **https://sso.synopsys.com. Retrieved September, 2014.**

ATTRIBUTION