An Enhanced DV Approach for Effectively Verifying High Speed, Low Power MIPI-MPHY5.0 Designs.

Eldhose P.M, Sagar Jayakrishnan, Suraj Vijay Shetty, Kuntal Pandya, Parag S. Lonkar
Samsung Semiconductor India Research (SSIR) Bangalore, India
(eldhose.pm, sagar.j, shetty.suraj, k.pandya, parag.lonkar)@samsung.com

***Abstract*- The version 5.0 is a major update to the MIPI MPHY PHY layer providing high throughput and low latency feature upgrades to Flash Storage and 5G Mobile ecosystems. Moreover, the design complexity has increased double fold with the addition of high speed Fifth Gear with data-rate up-to 23Gbps. To mitigate these challenges, many advancements are done in existing MPHY test bench to achieve better quality compared to its predecessors.**

***Keywords*- MPHY, Verification, Enhancements, Assertions, Coverage, FSM, Jitter**

I. INTRODUCTION

The MIPI MPHY specification is becoming a popular physical layer solution for Mobile and 5G ecosystems. The existing specification is revised to meet the rising industrial demands for higher throughput and superior performance with lower power consumption. High speed designs with technology node advancements requires high quality verification. Integrating designs, with logic updates to meet node expectations along with protocol upgrades into existing Test Benches always pose a challenge to verification teams. The main challenge is the high amount of churn that existing Test Environment has to go through to accommodate the upgrades and developments by meeting the industrial expectations. In this paper, we will discuss various aspects and challenges in verifying high speed MIPI MPHY 5.0 IP, along with scalability and reusability in focus. Previously, verification environment was built focusing towards data path and multiple iteration of regressions to meet functional and code coverage requirements set for the MIPI MPHY protocol. But, in the newer scheme for MPHY 5.0 specification update, a series of enhancements are done on the top of existing infrastructure. The approach discussed in this paper can be used as a good reference for verification of any serial specifications in the industry like PCIE, USB etc. To illustrate the same let us visit how conventional Test Bench was upgraded to help us perform qualitative verification for MPHY 5.0 protocol in a step by step-fashion.

A. FSM Inspired Scalable Test Bench Architecture.

Modularize each MPHY state requirement as a separate sub-sequence for increasing scalability and reusability. *B. Automated Python Script for Test Case Generation.*

Development of a python script to ease the test case sequence creation effort in environment with the use of an excel template.

C. Approach for faster Functional Coverage Closure.

Usage of different methods for reducing functional coverage closure time.

D. Interface Connectivity and Timing Checks.

Deployment of various schemes for robust timing and protocol handshake checks.

E. Increasing Regression Throughput using EDA Tool Option Save-Restore.

Saving critical simulation time using simulator option of Save-Restore. *F. Addition of CTS Tests and Error/Corner Test Scenarios.*

Developing and simulating Conformance tests and complex Error/Corner cases for early bug hunting.

G. Usage of High Precision Analog Models.

Verification of high precision analog models.

H. Jittered Clock Generation Module for CDR Stress Testing.

Development of a configurable Jitter engine for high quality stress testing of receiver block. *I. Parameterized Test Sequence.*

Command-line option support in testbench to avoid multiple compilation and quick directed scenario creation.

All the above said enhancements are discussed in detail in the following sections.

II. Conventional Test Bench Architecture

The conventional approach of verifying any serial protocol is connecting various verification components across the boundaries of Design Under Test (DUT) and applying stimulus at input ports, then output response is checked against a golden reference as shown in Fig. 1.

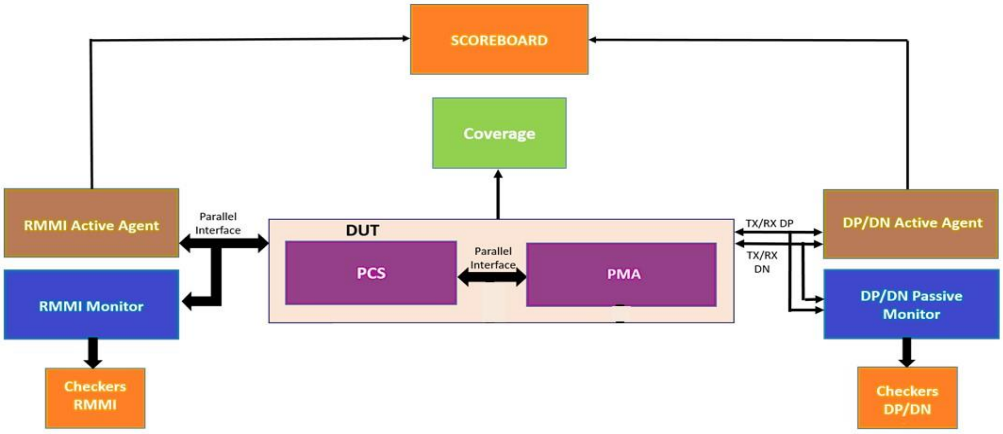

Fig. 1 Conventional Test Bench Architecture

III. Improved Test Bench Architecture

Fig. 2 illustrates enhanced Test Bench architecture to ensure Design Verification closure of ultra-high speed and low power MPHY 5.0 Protocol. A sophisticated SSC-Jitter module is connected in between Test Bench clock generation module and third party VIP to generate modulated data, which in turn is connected to receiver serial interface of Design. In addition, this Fig. also explains about the placements of Unknown and High impedance checker, PCS-PMA Interface and PMA Analog-Digital Interface checkers at various stages of Design.

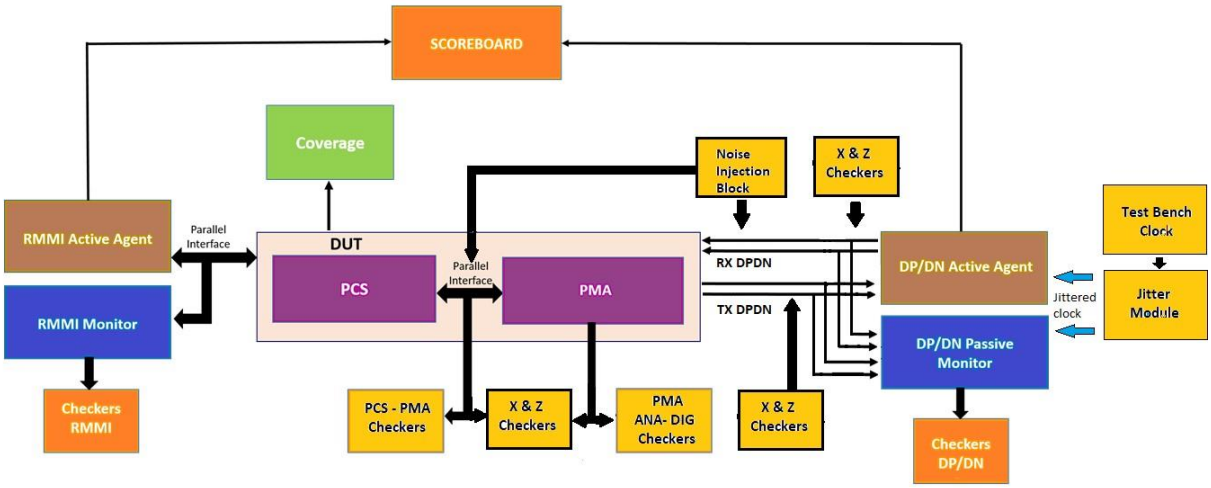

Fig. 2 Advanced Test Bench Architecture

This paper also focuses on test sequence enhancements based on MPHY specification FSM providing flexibility to user for controlling Test Bench, development of noise and CTS test scenarios, approaches to achieve faster coverage, increasing regression throughput, thorough check of interface handshakes etc. as well.

IV. Enhancements And Results

In this section, we will discuss about the enhancements done in Test Bench and results/improvements achieved based on implementation.

A. FSM Inspired Scalable Test Bench Architecture

In standard Test Bench, test sequences were written keeping focus only towards data path transactions.

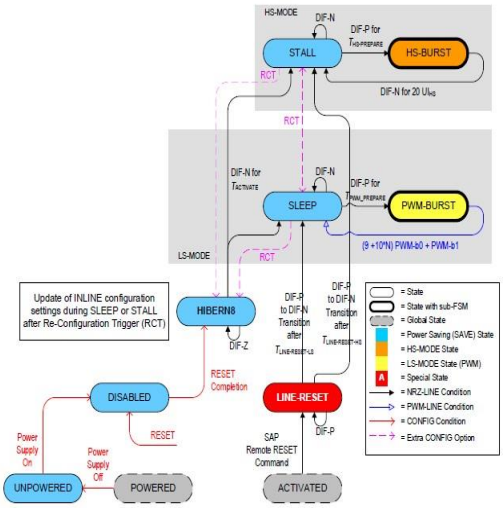

Fig. 3(a) MPHY Tx FSM State Diagram

```
class burst_sequence extends mphyVirtualSequence;
  `uvm_object_utils(burst_sequence)
  rand int no_of_bursts;
  rand bit hibern8_en,linereset_en,adapt_burst_en,sleep_en,stall_en;

function new(string name="burst_sequence");
  super.new(name);
endfunction
virtual task body();
  super.body();
  if (hibern8_en==1)
  begin
    //Register writes for Hibern8 Entry Exit
    if (stall_en==1)
    begin
      //Register writes for Stall
    end
    if (linereset_en==1)
    begin
      //VIP configuration Sequence for Linereset
    end
    if (no_of_bursts >=1)
    begin
      //Send Burst Sequence
    end
  end
endtask
endclass
```

Fig. 3(b) Pseudo Conventional Sequence Code

For example, Fig. 3(a) below represents MIPI MPHY Tx FSM state diagram, which represents various paths to send the data/bursts. Coming to implementation part, as seen in Fig. 3(b), each state machine is coded by using conditional statements.

The difficulty lies when additional settings, applicable for each state are required to be incorporated. Hence, this approach makes Test Bench more complex and considerable amount of time is required on enhancements. This inefficiency reduces scalability and reusability of Test Bench.

To overcome this challenge in verification, sequences are extensively enhanced and modularized based on MPHY states. Each state requirements has been defined as a separate sub-sequence, which in turn becomes part of the more complex main sequence. Fig. 3(c) represents the method where FSM states are mapped to individual class sequences and respective sequence can be called from test cases.

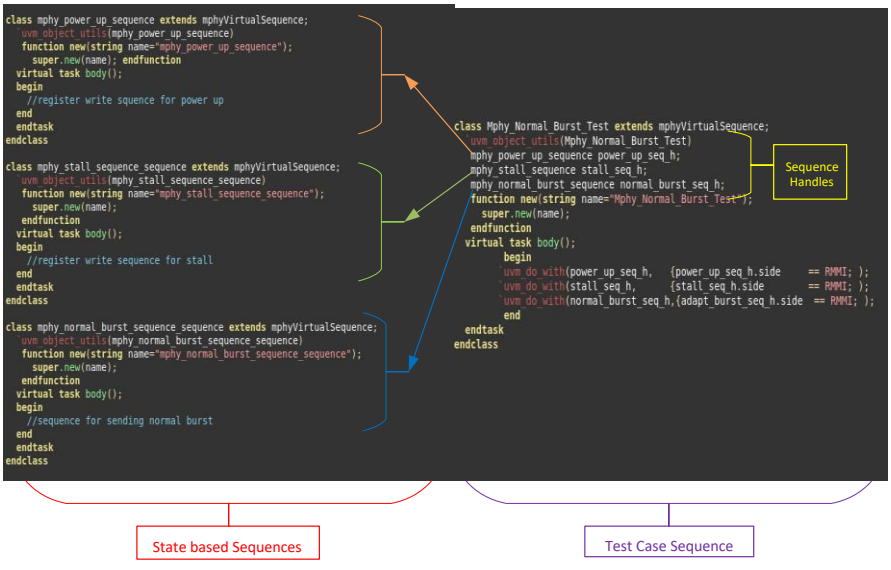

Fig. 3(c) Pseudo Code for State based Sequences

This improves the scalability of Test Bench and sequences became highly reusable.

The added advantages of this method are:

1. Easy accommodation of future updates as per specification and increased readability.
2. Debugging fail cases becomes much easier with modular sequences as compared to single complex sequence.
3. Test bench is more structured and in line with exact specification FSM flow.

B. Automated Python Script for Test Case Generation

With the specification update, complexity of design also becomes double fold compared to its predecessor. Hence, this mandates additional protocol feature verification leading to various new tests and their combinations. Manually writing test cases from scratch for each new scenario identified from specification is time consuming and inefficient.

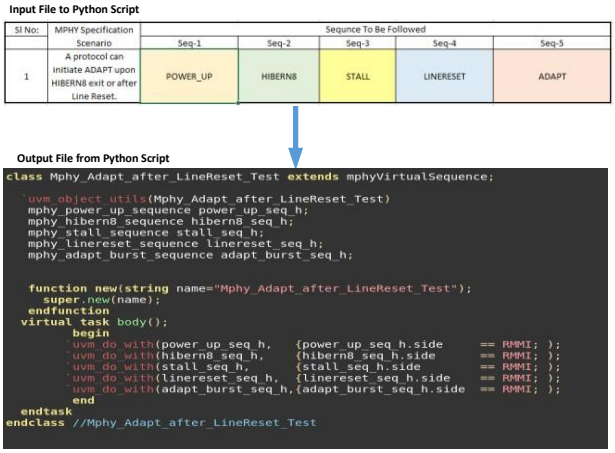

Fig. 4 Automated Test Generation

To mitigate the efforts, a test generation setup has been designed using Python Script and Microsoft Excel. Since, the Test Bench Architecture is based on FSM state diagram (as discussed in previous point), the identified test cases gets translated to respective FSM based sequence. Python script takes excel derived file as an input where each FSM state is mapped to corresponding DUT level sequence. Once the sequence order is finalized, the python script is evoked to create the required test case. Fig. 4 refers to input excel format and sample output file from python script.

The advantages of this enhancement are: -

1. Very handy and efficient in creating multiple scenarios in quick time.
2. A week time of efforts to create new tests and sequence updates has been reduced to a single day.

C. Approach for faster Functional Coverage Closure

Regression with constraint randomization is an integral part of Design Verification process as 100% functional coverage is an important milestone for closure. If cross coverage groups involve several cover bins, the amount of time it takes to meet the 100% functional coverage is much higher. Fig. 6 describes the usual method of functional coverage closure.

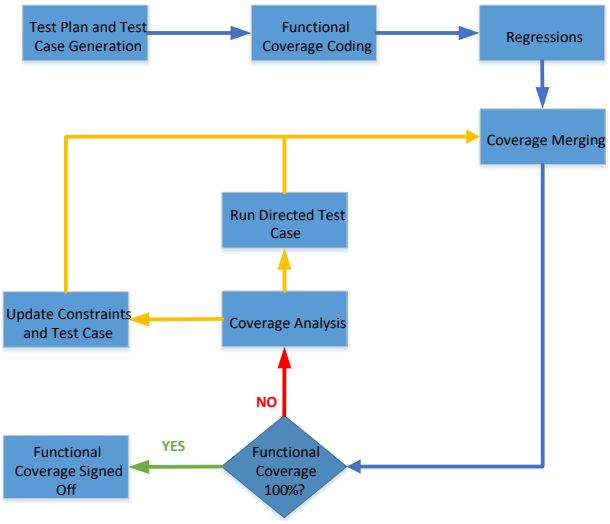

Fig. 6 Functional Coverage Sign Off Process

While random regressions run in background to make sure no new failures are seen, following measures have been taken to overcome this issue and to achieve maximum functional coverage with minimal efforts:

1. The test cases contributing to maximum coverage are ranked and custom regression suite is generated based on the results. With this method, redundant test cases contributing to same bins are discarded.
2. Weighted distribution on cover bins is added wherever possible for quick convergence of complex cross coverages, resulting in faster coverage closure.

Additional to this, to get maximum Functional Coverage with minimal iterations, an array based coverage collection method has been implemented. In this method, each cross coverage bin is mapped to an index based multidimensional array in coverage instance. Then, in post randomization block test sequence variables corresponding to each cross coverage bins are checked against the coverage tracking array values and manipulated to create a unique cross coverage bin. An example to demonstrate the method used is depicted in below algorithm:

Supported Gear = {1,2,3,4}, Rate = {0,1};

Then, let coverage tracker variable be reg [3:0][1:0] gear_rate:

Step-1: Start send burst sequence.

Step-2: Randomize gear & rate.

Step-3: In post randomization block, if gear_rate[gear][rate] == 0, if yes, then go to *Step*-5, else go to *Step-4*

Step-4: Iterate through gear_rate[gear][rate] using for loop and find the un-hit combination i.e. gear_rate[i][j] ==0.Once a un-hit combination is found, then override the gear and rate values and proceed to *Step-5*. If match is not found, it means all cross cover bins are already covered; goto *Step-6.*

Step-5: Register current gear and rate in coverage tracker. gear_rate[gear][rate] =1. go to *Step-6*

Step-6: Proceed with execution of test case.

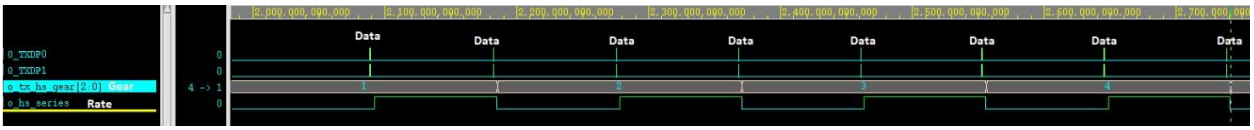

Fig. 7 Simulation results with coverage tracking sequence logic deployed

Fig. 7, illustrates an example of simulation results achieved with above algorithm for a cross coverage with two variables. As evident from the dump, all bins are hit within first eight sequence calls.

The improvements seen with this enhancement are:

1. Ranked regression suite plus parametrized direct test approach reduced the functional coverage time by 50%.
2. Could identify duplicate test cases in Test suite which are not adding much value to coverage.

D. Interface Connectivity and Timing Checks

MPHY Design consists of Physical Coding Sub-Layer (PCS) and Physical Media Attachment Layer (PMA). In addition, PMA further deals with analog and digital logic within. Therefore, connectivity and protocol related checks are applied at different levels of design.

Different ways to verify the interface logic are:

1) **Third Party Formal Connectivity tool:** To ensure the physical connections between interface signal, the tool takes comma-separated-values (CSV) file and register transfer logic (RTL) as input and automatically generates stimulus to verify the connection. As it provides interactive debug environment, the results are faster and easy to debug missing connections.
2) **Interface Timing Verification:** As discussed above the PCS-PMA interface and PMA Analog-Digital interface are crucial part of such designs and ensuring their intactness is critical for quality of IP. We ensure enough checks, on value and signal timings, are applied at these interfaces to ascertain signal exchange between two complex logics (PCS-PMA or PMA Analog-Digital) always happens as expected.
3) **Unknown (X) and High Impedance (Z) checker:** Having a big Analog logic may at times result in intended or unintended X/Z being driven on interface signals for certain cases, which can potentially affect digital simulations. So having such checks is crucial at these interfaces. These checks are clock independent and hence are able to flag any unexpected transitions (X or Z) at any point of simulation.

The improvement seen after doing this enhancement are:

1. Early bug detection which can save cost to company.
2. Revealed critical design flaws which could have led to excess power consumption.
3. Formal connectivity checks exposed interface signal width mismatches.
4. More confidence on design, with thorough check of protocol handshake at interfaces.
5. Could validate the effect of X/Z originated from PMA analog block in PCS functionality.

Table 1 shown below are the statistic w.r.t. the bugs found in design by implementing above discussed methods.

Total no of interface signals	73
No of critical bugs found with formal connectivity checks.	2
No of critical bugs found with X/Z checkers	6
No of critical bugs found with interface timing checkers	23

E. Increasing Regression Throughput using EDA Tool Option Save-Restore

In majority of test cases initial configuration and power-up sequence is common. As highlighted in Fig. 8, design need to exercise the power supply sequence. Moreover, MIPI MPHY specification mandates a warm up wait time of 1.5ms on Hibernate exit.

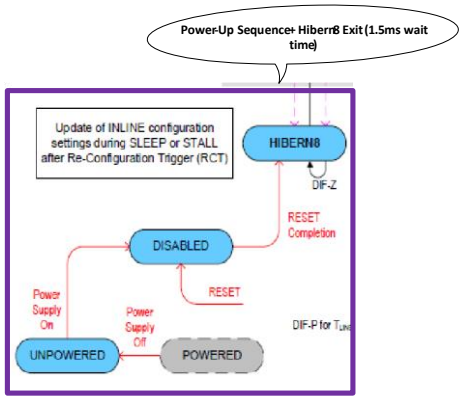

Fig. 8 Region of Save-Restore method applied in Advanced Test Bench

On analysis of majority test scenarios, this common sequence consumes approximately one third of total simulation time. If we can pre-load the simulation to reach this design state quickly it gives a huge possibility of saving critical simulation time. The simulator has a resume option, in which the saved snapshots can be reused to start a new sequence from the point of last saved data. This feature is implemented over all tests in regression suite to increase the throughput by saving good amount of regression time. We have seen **5.1% to 30%** reduction in CPU time for normal tests to long run tests respectively. And on an average, it has been observed an approximate **15%** reduction in regression run time after implementing this method and tested with a test suite of 1100 runs.

F. Addition of CTS Tests and Error/Corner Test Scenarios

Conformance tests are early tests executed in silicon once the product is out of fabrication laboratory. Silicon is qualified based on the results of CTS tests. So, it's important to check the design functionality in that front. Special tests were implemented for Frequency, Unit Interval, De-emphasis, Amplitude, Slew Rate and Receiver Jitter Tolerance checks. Though it's hard to simulate entire analog level functionality like amplitude, de-emphasis, slew rates etc. from digital level, it's possible to check the analog controls corresponding to these functions. For simulating silicon BERT test, retimed loopback mode is programmed in DUT and data integrity checks are performed with jittered loopback patterns.

In Silicon, its normal that spurious noises interfere with data patterns or interface control signals and may lead to abnormal behavior. So it's important to simulate the DUT behavior by injecting noises at interface signal levels and receiver serial inputs at various DUT power states. Enough time has been spent with design team to find all possible noise cases. For this activity past silicon bug cases also has been referred. Around 30 noise test scenarios were identified and test scenarios were created for all of them. The improvements seen with this enhancement are: -

1. Noise scenario stress testing and CTS tests increased confidence on design.
2. 5 potential bugs were found at early stage with noise tests and CTS testing exposed 2 critical bugs.

G. Usage of High Precision Analog Models

For typical AMS designs like MPHY, for digital simulation purpose, analog behavior is modelled at a high level to mimic the functionality. However, for an efficient testing of analog features, it always helps to have models

having real data types and related logic, taking them much closer to real analog behavior. Such models were added as final verification signoff criterion and multiple features like Eye Monitoring, Jitter Tolerance, calibration (de-emphasis) etc. were qualified using these models. For example, Eye Monitoring test is performed by sweeping sampling clock phase shifter in horizontal axis and voltage reference in vertical axis. For each of their values, Pass/Fail criterion is captured and later used for viewing the Eye Opening in graphical format.

H. Jittered Clock Generation Module for CDR Stress Testing

MPHY protocol follows a typical SerDes architecture with a serial transmitter and a serial receiver. The transmitted serial data then passes through the channel and gets exposed to various distortions say ISI, Noise, Jitter etc., resulting in mild to severe modulation w.r.t. actual data. This makes the task of CDR (Clock and Data Recovery), at receiver side, much complex to faithfully reproduce the transmitted data. Hence, it becomes a crucial test scenario to be added in verification suite to ascertain Design works well within tolerance limits set by Protocol. A specific SSC-Jitter module is hooked in Test Bench, having the capability to modulate any serial data stream, based on input parameters, before feeding the same to receiver logic of DUT. Fig. 9 is graphical representation of Jitter Tolerance regression results achieved against specification settings.

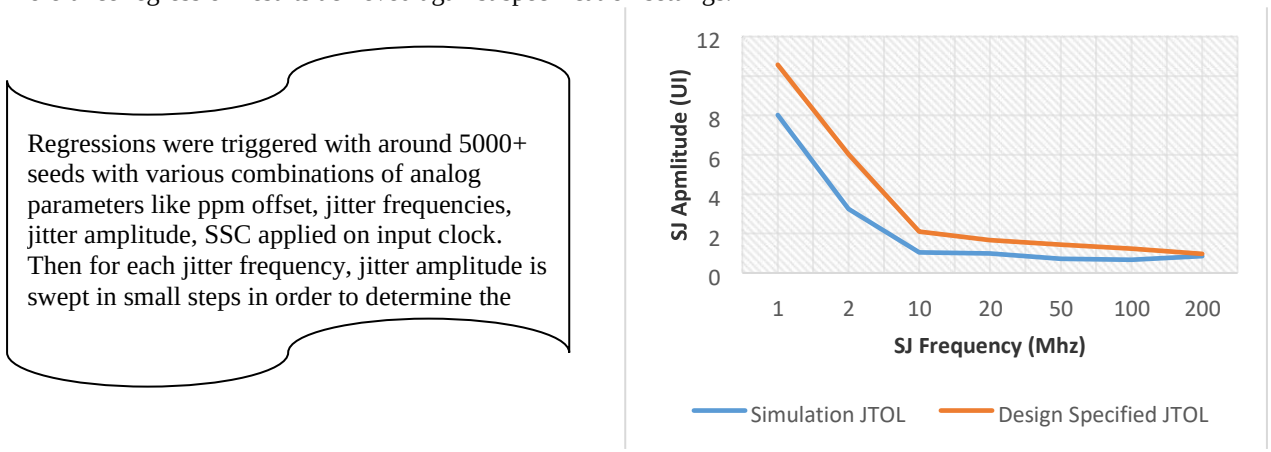

Fig. 9 Jitter Tolerance Curve

I. Parameterized Test Sequence

High Speed MPHY protocol uses highly sensitive CDR, BIST, Calibration designs with different modes of operations, which requires verification with various design settings. Previously, the verification was done by creating multiple compilation builds for each feature. However, maintenance and functional coverage closure with such Test Bench architecture is difficult where quick sign off with coverage is required.

Another, problem statement was to reproduce any directed scenario in such a complex and highly randomized Test Bench architecture.

In order to verify different modes supported within each features with minimal changes in Test Bench as well as reproducing any scenario, **`define** macros are replaced by **$value$plusargs API's**. This helps in generating directed scenarios without changing test sequence manually and hence targeted coverage scenario can be achieved without compiling the Test Bench.

For example, MPHY CDR feature has various parameters like jitter (random and sinusoidal), ppm, ssc. In order to reproduce any scenario post compilation, the command line switches are passed, which in turn feeds value to internal Test Bench variables through **$value$plusargs API's.**

Jitter		**SSC**	**Frequency Offset**
RJ	**SJ**		
0	0	0	0
0.12UI	0.25UI	0	+300 ppm

In above table, SJ (Sinusoidal Jitter) supports 2 values, similarly RJ (Random Jitter) and PPM. Each supported value is converted to a command line switch and selected switch will be assigned to internal Test bench variables.

Hence, helps in generating desired scenario without modifying any test case or sequence. Fig. 10, below explains the directed scenario with SJ=0.25UI.

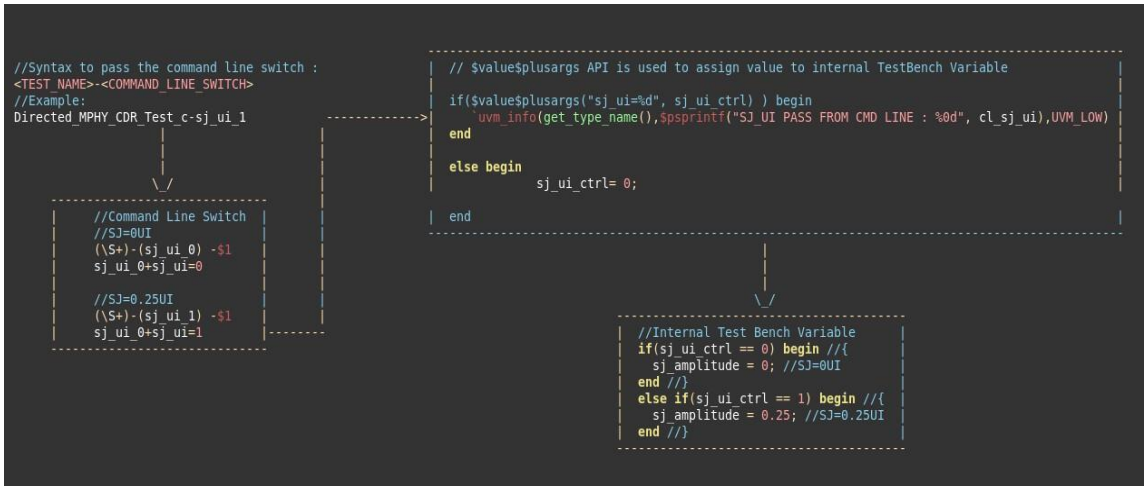

Fig. 10 Parameterized Test Sequence

CONCLUSION

In this paper, we have tried to demonstrate various steps taken and enhancements done in our existing Test Bench to make it much more efficient as well as effective in fulfilling the design challenges with higher speeds and much complex logics. For ex. Interface assertions helped us to unearth some really critical bugs, which could have led to excess power consumption, regressive CDR testing ensured high quality jitter tolerant receiver logic while at the same time with runtime switches, save and restore, quicker turn around on coverage closure helped us big time in closure of various verification activities. Moreover, the enhancements done are not only restricted to MPHY protocol, but can be applied widely for verifying other PHY's.

FUTURE SCOPE

The current enhancements have significantly improved the scalability and reusability of the test bench while helping in quicker coverage signoff. On top of these, we are working on some more enhancements to improve coverage collection even further. Changes (minor or major) in existing designs to accommodate derivatives and customer requests are quite common, though the design impact is less but accommodating these features into coverage framework requires considerable effort. A flexible coverage setup capable for providing mechanism to easily add/drop relevant cover bin is always an added advantage.

The few such identified areas are:

1) Support in test bench for sending PRBS7 and PRBS15 patterns.
2) RAL model implementation for RMMI Interface.

The enhancement works related to these improvement items are in progress.

REFERENCES

[1] mipi_M-phy_specification_v5-0.pdf, by MIPI Alliance

[2] SystemVerilog 3.1a Language reference Manual

[3] https://www.academia.edu/30128384/SERDES_Rx_CDR_Verification_using_Jitter_Spread_spectrum_clocking_SSC_stimulus