UVM Based Verification Environment for USB 3.0 Physical layer and LTSSM of Link layer

Arpit Patel
Department of VLSI & Embedded System Design
GTU PG School,
Ahmedabad – India
arpitpatel360@gmail.com

***Abstract*—** USB 3.0 is the protocol used for connecting computers to electronic devices, which supports data transfer rate of 5 Gbit/s. The physical layer of USB 3.0 performs 8b/10b encoding-decoding, serialization-deserialization and scrambling-descrambling. Link Transition and Status State Machine(LTSSM) is used for managing its end of the physical connection between Transmitter and Receiver. Verification environment is a configurable structure built, which can be interfaced with the design and test sequences can be applied to the design through it. SystemVerilog(SV) with Universal Verification Methodology(UVM) is used widely for verification. Both Physical layer and LTSSM are verified using Verification Environment developed for them using SV-UVM.

Keywords- *USB 3.0; UVM; Verification Environment; DUT; functional verification*

*****

I. INTRODUCTION

USB 3.0(SuperSpeed) architecture has these elements: Host, hub and device. Host and device are two ends of communication. Host is a source or sink of information. Devices are sources or sinks of information exchanges. Hubs are the interconnection points between Host and device. USB 3.0 is host-directed protocol and device will notify its readiness using "ready" packet. Device can request service from host asynchronously.

It has 3 layers: Physical, Link and Protocol as shown in Figure 1.

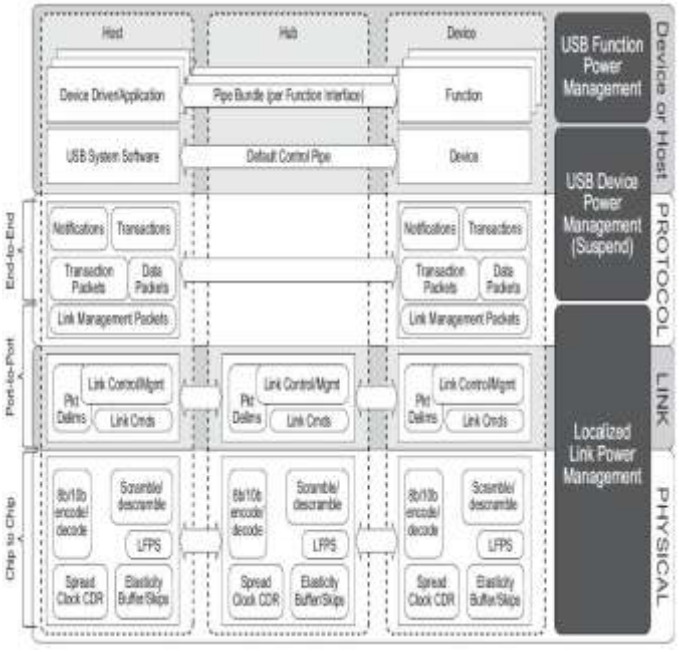

Figure 1 SuperSpeed Bus Communications Layers and Power Management Elements

The physical layer provides physical connection of downstream port of host or hub and upstream port of device. It does 8b/10b encoding-decoding to ensure enough transitions to recover clock and data on Receiver side, scrambling-descrambling to reduce EMI emissions and channel performance improvising techniques like transmitter de-emphasis and equalization.

The link layer manages physical layer initialization and events, i.e., connect, removal, and link power management using LTSSM. It implements protocol for flow control.

The protocol layer defines rules for communication between a host and device using Packet headers. Packet headers contain the information to guide link layer to manage flow of data in those packets from port to port [1].

Verification Environment's main advantage is reuse capability. It can be horizontal between designs or vertical from blocks in design to integrated blocks to whole design [2].

SystemVerilog(SV) is advantageous because of OOP concepts, constrained randomization, regression and coverage capability [3].

Plus, Universal Verification Methodology(UVM) has advantages of base class library, object factory, transaction level modelling(TLM), sequences, macros and configured database. UVM class library is divided mainly in two types: uvm_component and uvm_transaction. uvm_component models the testbench components like monitor and driver. uvm_transaction models sequences. Object factory is for creating generic reusable code and provide final specification of the object at run time. UVM components are connected with each other through TLM interfaces for reuse purpose. TLM uses port and socket methods for interfacing. Sequences with constrained randomization are applied from agent to stimulate the test cases. UVM provides macros to generate routine code for registering in factory or utility methods automatically. Database can be prepared for any data type in UVM, which can be accessed by a string name. This database can be configured differently for different test cases [4].

First verification environment for Physical layer is designed by developing component classes like driver, monitor, scoreboard with expected result and functional coverage collector. Then by reusing some of these classes environment to verify LTSSM has been designed. These verification environment are interfaced with design through bus functional model interface and code and functional coverage are measured for both of them.

 ISSN: 2321-8169
2811 - 2815

The remaining paper is directed like this. Part II gives detailed description of design of PHY layer and LTSSM. In part III UVM verification environment architecture is explained. In part IV the results achieved by applying this environment to the design are shown. At last part V concludes the work.

II. USB 3.0 DESIGN

Design of Physical layer and LTSSM are described in detail here. The pin diagram of both of them containing input, output, status and control pins are shown in Figure 2 and Figure 3.

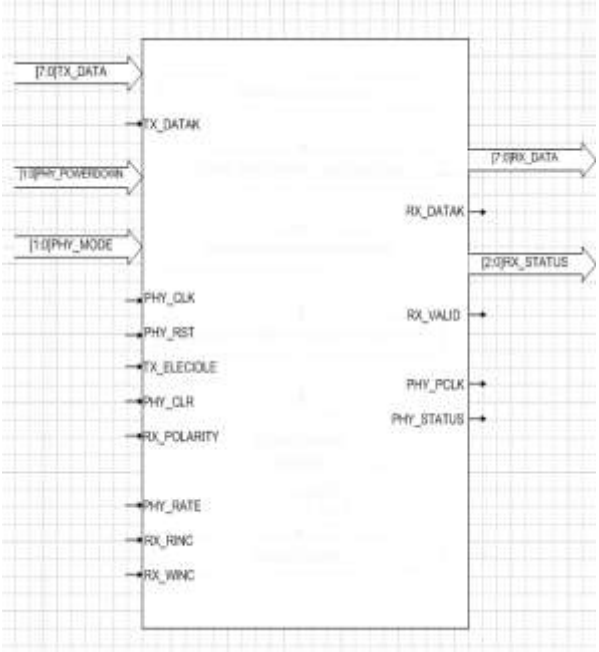

Figure 2 PHY pin diagram

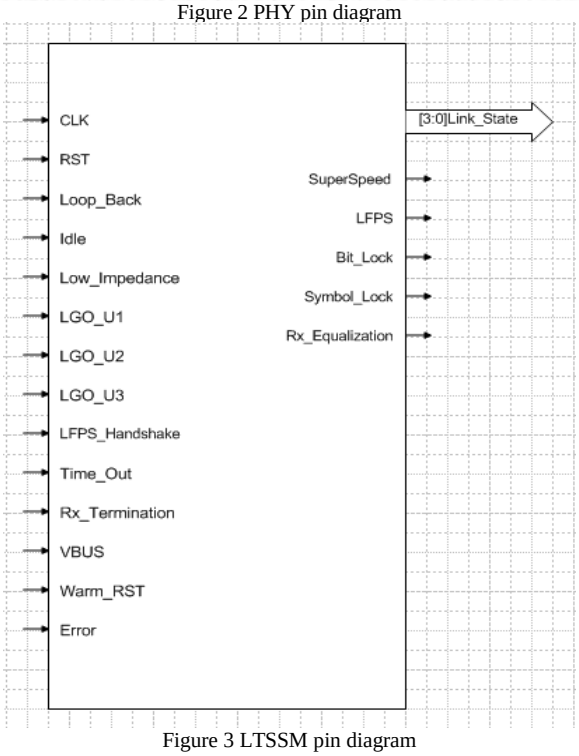

Figure 3 LTSSM pin diagram

A. *Physical layer*

The transmitter and receiver block diagram of phy layer are shown in Figure 4 and Figure 5.

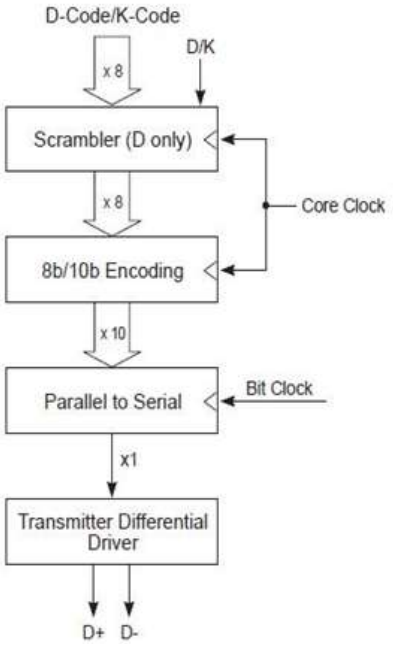

Figure 4 Transmitter Block diagram

In design from [5],data is transferred either on 2.5GT/s or on 5.0GT/s depending upon the mode and rate. The design uses core clock of either 125MHz or 250MHz that is used to transfer data on parallel interface

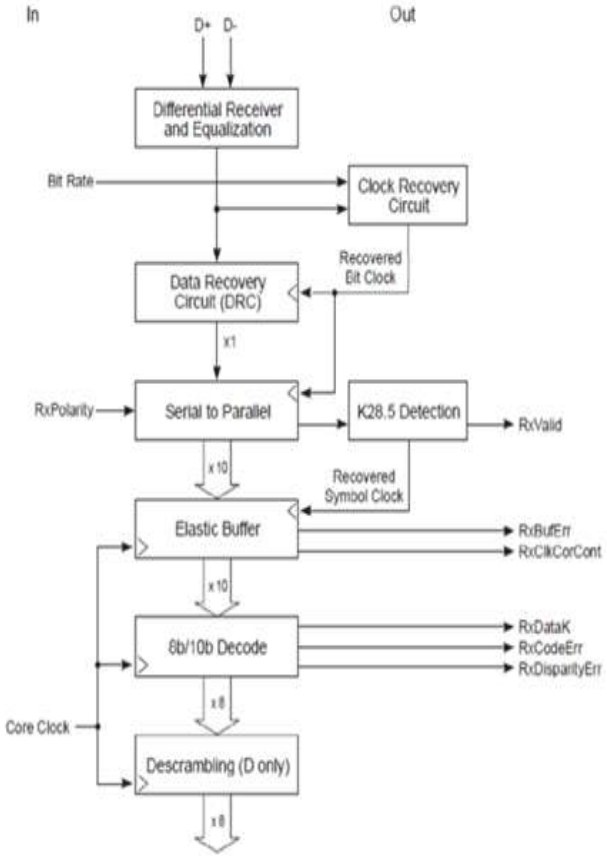

Figure 5 Receiver Block Diagram

On Receiver side incoming asynchronous serial data is captured and receiver clock is locked with incoming data. In Receiver FIFO is used for buffering the incoming data and status of received data is shown in RxStatus.

ISSN: 2321-8169
2811 - 2815

B. LTSSM

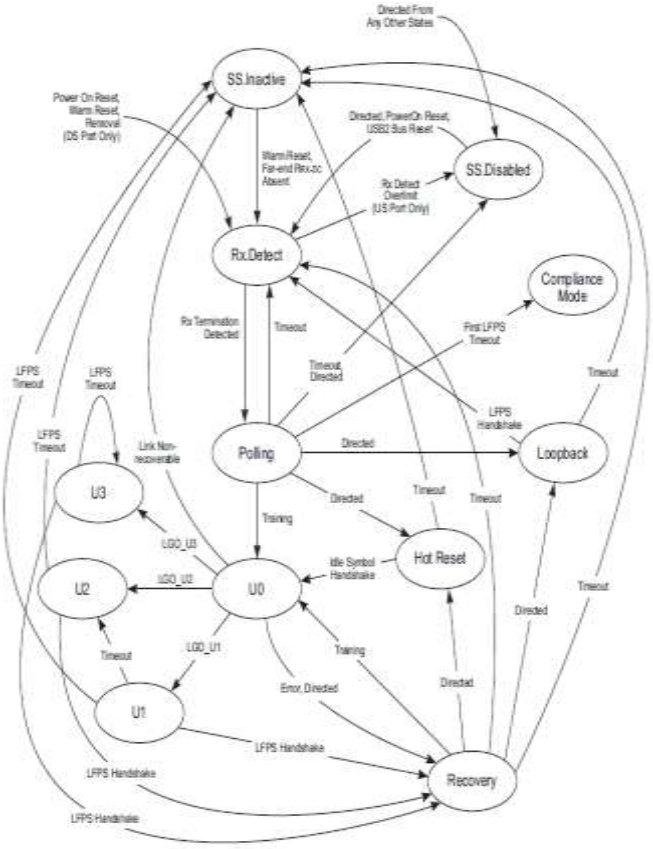

Figure 6 State Diagram of the Link Training and Status State Machine

There are 12 states in LTSSM. U0, U1, U2 and U3 are four operational link states. Rx_Detect, Polling, Recovery and Hot_Reset are four link states. LoopBack and Compliance_mode are for bit error test and transmitter compliance test.SS_Inactive and SS_Disable are link error states.

III. VERIFICATION ENVIRONMENT ARCHITECTURE

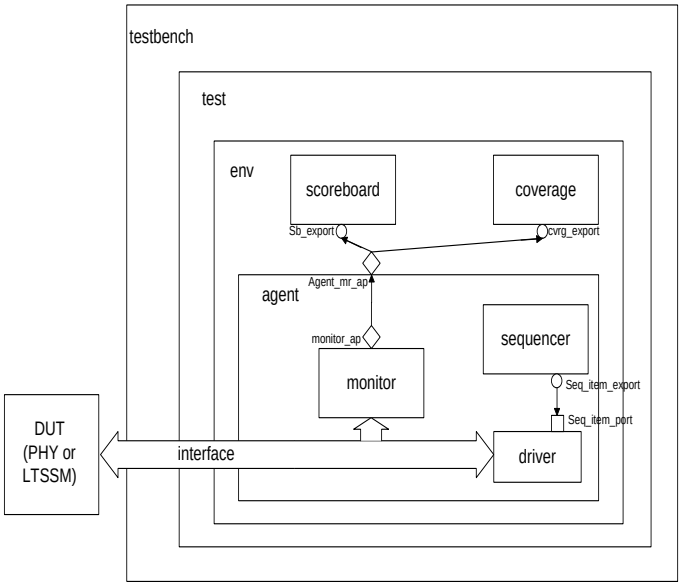

Figure 7 UVM testbench

Interface : It connects the driver and monitor to the Design. It takes incoming input from driver and applies it to design pins and sends input applied and updated output received from design into monitor.

Port : It is used to send transaction in another component's export.

Export : It receives transaction from another component's port.

Analysis port : From one port transaction can be sent only in single export. But to send it to multiple exports analysis port is useful.

Driver : It converts the transaction received from sequencer into signal level and sends it to interface.

Monitor : It takes signal level data from interface and converts it into transaction and sends it forward to scoreboard for checking correctness and coverage to keep track of functionality covered.

Sequence_item : It is constraint randomized transaction for particular test case. It contains all inputs to be applied to design and outputs to be read from design.

Sequence : It is bunch of sequence_item collected together and co-ordinated for checking a test scenario.

Sequencer : It applies the sequeces to the driver from agent using TLM port-export methods.

Agent : It contains driver, monitor and sequencer. It instantiates and configures these components and then makes TLM port connection between them. Different agents can be used for different purpose, for ex. Transmitter agent needs driver while Receiver agent needs only monitor.

Scoreboard : It checks correctness of functional behaviour of design. It gathers transactions from monitor into tlm_fifo and then compares the output in them to output received from reference model of design by giving input from these transactions.

```
UVM_INFO tb/scoreboard.svh(900) @ 95891: uvm_test_top.env_h.scoreboard []
Predicted State = Polling
Actual State = Polling
SELF CHECKER :: PASS

UVM_INFO tb/scoreboard.svh(900) @ 95911: uvm_test_top.env_h.scoreboard []
Predicted State = Rx_Detect
Actual State = Rx_Detect
SELF CHECKER :: PASS

UVM_INFO tb/scoreboard.svh(900) @ 95931: uvm_test_top.env_h.scoreboard []
Predicted State = SS_Disable
Actual State = SS_Disable
SELF CHECKER :: PASS
```

Figure 8 scoreboard from LTSSM

Coverage : It keeps track of functionality verified. It gathers transaction like scoreboard, and from input it updates it's covergroups and coverpoints. By checking coverage, constraints can be updated for regression testing to reach to corner cases.

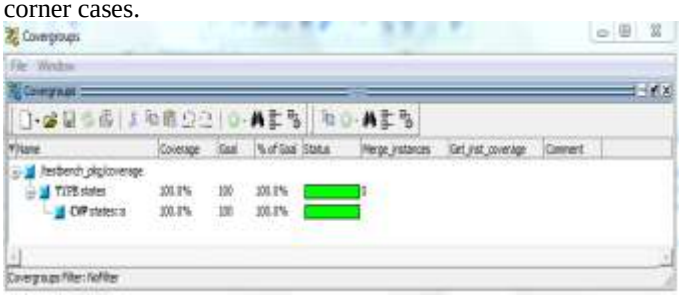

Figure 9 Functional coverage for state transitions for LTSSM

Environment : It contains agent, scoreboard and coverage. It instantiates and configures these components and then makes TLM port connection between them.

Test : It deals with employment of sequences for particular test scenario into sequencer of the environment. It can be selected from commnd line also.

Testbench : It is the top module of testbench. It connects design with the interface and then applies various tests designed to check various test scenarios to the design.

 ISSN: 2321-8169
2811 - 2815

IV. RESULT

A. PHY layer

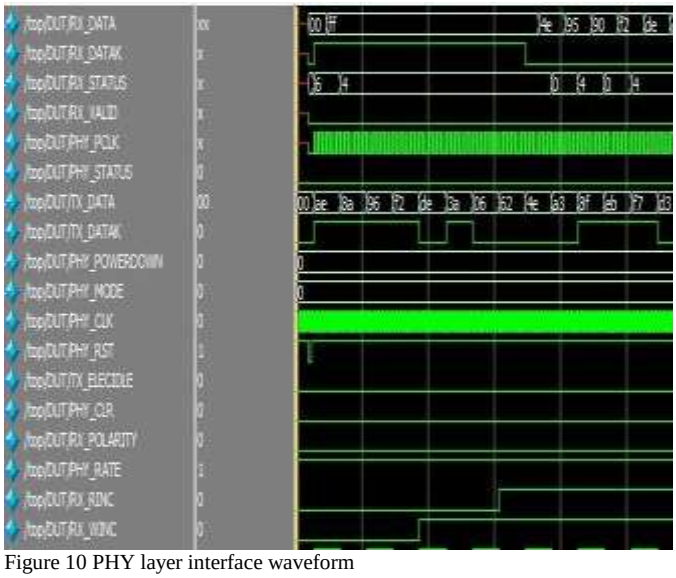

Figure 10 PHY layer interface waveform

In Figure 10, for different control signals, according to current mode, data transmission from transmitter to receiver is visible. Current operational mode can be changed by changing phy_mode, phy_powerdown and phy_rate signals. rx_polarity signal can be used to invert the polarity of data to be decoded.

```
Coverage Report Summary Data by file
File: AASD.v
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                          4         4         0     100.0
File: CLOCK_DIV.v
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                          6         6         0     100.0
File: CLOCK_GEN.v
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                         12        12         0     100.0
File: DECODE.v
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                         51        51         0     100.0
File: DFF.v
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                          7         6         1      85.7
File: DUT.v
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                          0         0         0     100.0
File: ENCODER_1.v
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                         60        60         0     100.0
File: FIFO_v.v
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                         44        41         3      93.1
File: PartoSerial.v
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                          8         8         0     100.0
File: Rx_STATUS1.v
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                         15        12         3      80.0
File: SertoPar.v
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                         11        11         0     100.0
File: Synchronizer.v
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                          6         6         0     100.0

NEVER FAILED: 100.0%  ASSERTIONS: 22
```

Figure 11 Code coverage for PHY layer

Figure 11 shows the code coverage for the test sequences applied to PHY layer to cover all operational modes and normal/inverted data.

B. LTSSM

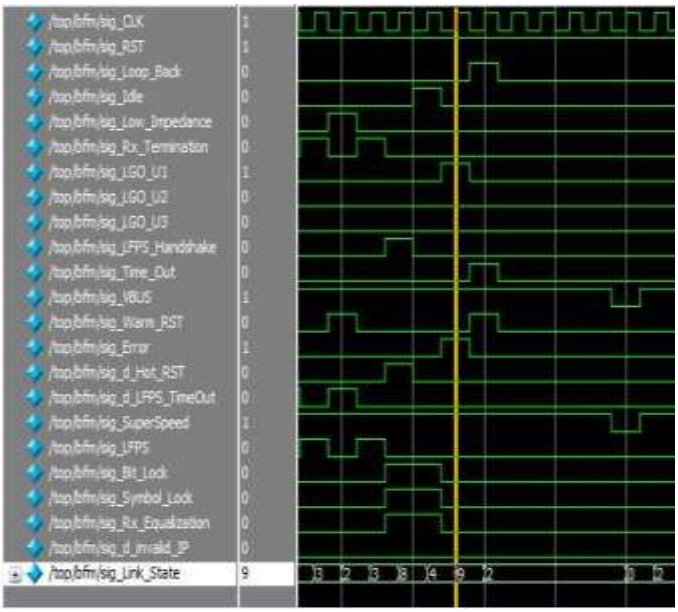

Figure 12 LTSSM interface waveform

In fig 12, for different control signals, output values and link state are shown. To change the state proper constrained transaction of control signals based on current state needs to be applied.

```
Coverage Report Summary Data by file


File: DUT.v
    Enabled Coverage          Active      Hits    Misses % Covered
    ----------------          ------      ----    ------ ---------
    Stmts                        285       285         0     100.0

TOTAL COVERGROUP COVERAGE: 100.0%  COVERGROUP TYPES: 1


NEVER FAILED: 100.0%  ASSERTIONS: 22
```

Figure 13 code coverage for LTSSM

Figure 13 shows the code coverage for the test sequences applied to LTSSM to cover all state transitions.

V. CONCLUSION

Verification is too much important to get bug-free designs and SV-UVM is best for that purpose. Using SystemVerilog with UVM, Verification environment to verify USB 3.0 Physical layer and LTSSM functionality has been designed. After verifying the designs with these Verification Environments, maximum code and functional coverage possible with these test scenarios and test cases has been achieved. These verification environments can be reused for verifying systems which uses USB 3.0 protocol.

ACKNOWLEDGMENT

I would like to express my gratitude to Mr. Ashish Prabhu, CDAC-ACTS, pune and Gujarat Technological University for their guidance and resources.

REFERENCES

[1] Universal Serial Bus 3.0 Specification (Rev 1.0), Available at http://www.usb.org/developers/docs/documents_archive/

[2] Francesconi, J. ; Agustin Rodriguez, J. ; Julian, P.M., "UVM based testbench architecture for unit verification", IEEE conference on Micro-Na noelectronics, Technology and Applications (EAMTA), 2014 on Publication Year: 2014 , Page(s): 89 – 94

[3] Flake, Peter, "Why SystemVeriog? ", IEEE confer ence on Specification & Design Languages (FDL), 2013 Forum on Publication Year: 2013 , Page(s): 1 – 6

[4] Bromley, Jonathan, "If SystemVerilog is so good , why do we need the UVM?", IEEE conference on Specification & Design Languages (FDL), 2013 Forum on Publication Year: 2013 , Page(s): 1 – 7

[5] Design synthesizable USB 3.0 using Verilog HDL and simulate design using Cadence, Available at http://scholarworks.csun.edu/bitstream/handle/10211.2/1060/USB_3_Final.pdf?sequence=1

[6] Bhaumik Vaidya, NayanPithadiya, "An Introduction to Universal Verification Methodology", Journal Of Information, Knowledge And Research In Electronics And Communication Engineering, Nov 12 To Oct 13 , Volume – 02, Issue – 02, Page(s). 420-424

[7] http://colorlesscube.com/

[8] Accellera, UVM User Guid

[9] Accellera, UVM Class Reference

[10] Accellera, UVM VIP

[11] https://verificationacademy.com/

[12] www.usb.org

[13] http://www.mindshare.com/files/resources/MindShare_Intro_to_USB_3.0.pdf

[14] Chris_Spear, Greg Tumbush, "SystemVerilog for Verification",Springer