# Constrained Random Verification of PCIe Transaction Layer

**Sapna**[1], **Chandrashekar.C** [2]

*[1] Student, Department of Electronics & Communication Engineering B.N.M.I.T. College of Engineering, Bangalore-560 070, Karnataka, India*
*[2] Assistant Professor, Department of Electronics & Communication Engineering B.N.M.I.T. College of Engineering, Bangalore-560 070, Karnataka, India*

---------------------------------------------------------------------***---------------------------------------------------------------------

**Abstract -***Modern electronic devices are demanding for high bandwidth. This can be achieved by (Peripheral Component Interconnect Express) PCI Express. PCI Express uses packet based communication, packets can be generated in transaction layer. PCI Express is used where the high speed is required for the transfer of data. Transaction Layer supports different types of transactions such as Posted and Non-Posted transactions. PCI Express uses the serial interface through which clock skews can be reduced and higher throughput can be achieved which is useful in chip verification because chip verification depends on throughput rate. This paper aim is to do the constrained random verification of PCI Express Transaction Layer (TL). Universal Verification Methodology (UVM) is used for the verification of Transaction Layer Packet . In this, verification environment is created using four Universal Verification Component such as Advanced Peripheral Bus (APB) UVC, Advanced eXtensible Interface (AXI) UVC, Data Link Layer Transmit (DLL_ TX) UVC and Data Link Layer Receive (DLL_RX) UVC. These UVC components are going to help to drive the transactions. Design Under Test (DUT) consists of Transaction Layer Packet. Constrained random verification used for complex system. In this paper the verification is done for the PCI Express Transaction Layer to verify the transactions. Simulation results are shown in Questasim tool.*

***Key Words***: **PCIe, TL, UVM, APB, AXI, DUT.**

## 1. INTRODUCTION

Development in electronic industry, have increased the data processing ability of a single chip. To satisfy the required bandwidth, a new technology called PCI Express was introduced to replace PCI, PCI-X, and AGP. PCI Express is used in consumer, server, and industrial applications. PCI Express uses the packet based communication hence the Packets are generated in Transaction Layer. Here the only targeted test cases are generated and hence constrained random verification is used.

### 1.1 Transaction Layer

Responsibility of Transaction Layer is the assembly and disassembly of Transaction Layer Packets (TLPs).TLPs is used to communicate transactions such ass read and write and certain types of events. In this case AXI is used for generating transactions such as write, read transactions. It supports the split transactions. Components of a TLP are: Header, Data Payload and an optional ECRC. Figure 1 shows layering diagram Highlighting the Transaction Layer. Figure 2 shows the fields present in TLP Headers. The packet format supports different forms of addressing depending on the type of the transaction (Memory, I/O, Configuration, and Message). Table1 shows the PCI Express posted and non-posted transactions. Transaction Layer is having different TLP Packet types such as memory read request, memory write request , IO read request, IO write request, Configuration read, configuration write, completion without data and completion with data.

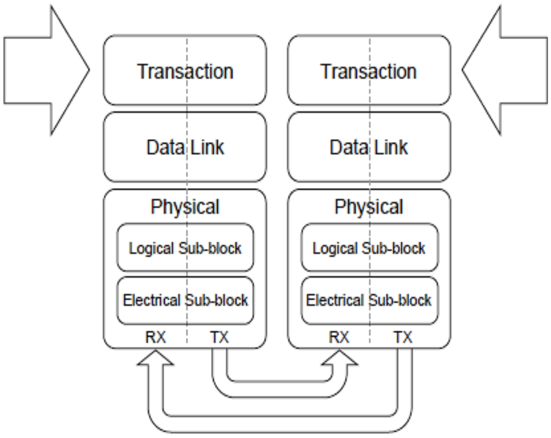


**Figure 1.** Layering Diagram Highlighting the Transaction Layer

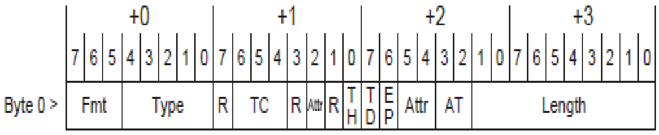


**Figure 2.** Fields Present in All TLP Headers

**Table1.** PCI Express Posted and Non-Posted Transactions

| Transaction Type | Non-Posted or Posted |
|---|---|
| Memory Read | Non-Posted |
| Memory Write | Posted |
| Memory Read Lock | Non-Posted |
| IO Read | Non-Posted |
| IO Write | Non-Posted |
| Configuration Read(Type0 and Type1) | Non-Posted |
| Configuration Write (Type0 and Type1) | Non-Posted |
| Message | Posted |

## 2. METHODOLOGY

UVM methodology is used for the verification of different transactions of PCI Express Transaction Layer. Figure 3 shows the UVM Verification environment.

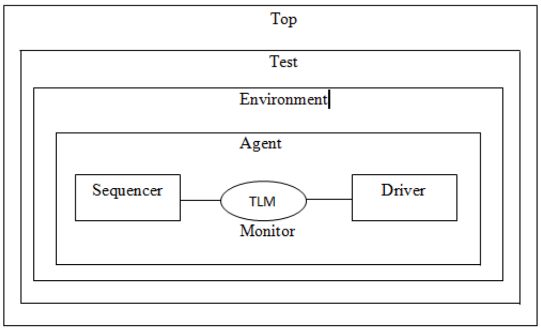


**Figure 3.** UVM Verification Environment

### Sequencer

It is used as generator and its function is to control the data item provided to the driver for execution.

### Driver (BFM)

Function of the driver is receives the data item and drives it to the DUT.

### Monitor

Monitor collects the data items and samples DUT signals.

### Agent

More than one agent can be used in the verification components. Agents can be classified as active agent and passive agent. Active agent drives the transactions whereas passive agent monitors the activity of DUT.

### Environment

It is the top level component of the verification component and contains one or more agents.

### Transaction Level Modeling

Communication between different components of UVM is provided by TLM interfaces.

## 3. ARCHITECTURE DETAILS

### 3.1 Architecture of Transaction Layer

Figure 4 shows the architecture of transaction layer, at transmit side processor will load descriptors (both transmit and receive) to external memory program, Processor is the only component which will program the descriptors, always knows where to write the data (to be used for transmit purpose) and from where to read the data (to be used for receive purpose), TX descriptor processing engine will start fetching descriptors, it is passed to DMA block then it will create a TLP. To create TLP it will get information from the configuration register to fill header fields. Than whole TLP is framed and ECRC is generated and is forwarded to transmit block. Transmit block will transmit 32 bit per clock cycle. At receive side receive block will receive the TLP frame and then it is decoded, passed to DMA receive block than it can be sent to the RX descriptor processing block. Descriptor format is 64 bit first 32 bit is used to find the address where to send TLP and second 32 bit is used find direction as either TX or RX.

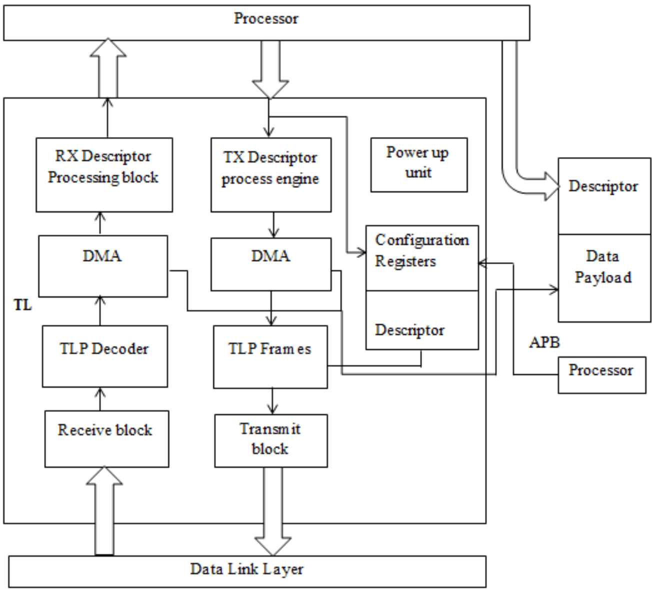


**Figure 4.** Architecthure of Transaction Layer

### 3.2 Test bench Architecture

Figure 5 shows the test bench architecture which is going to be used for the verification of different transactions or TLP packet types of transaction layer. Test bench architecture consists of different UVC components such as APB UVC, AXI UVC, DLL RX UVC and DLL TX UVC. All these UVC components are used for driving the transactions. AXI UVC is used to generate the write and read transactions. Using the information of different fields present in TLP header, test cases can be generated for different TLP Packet types.

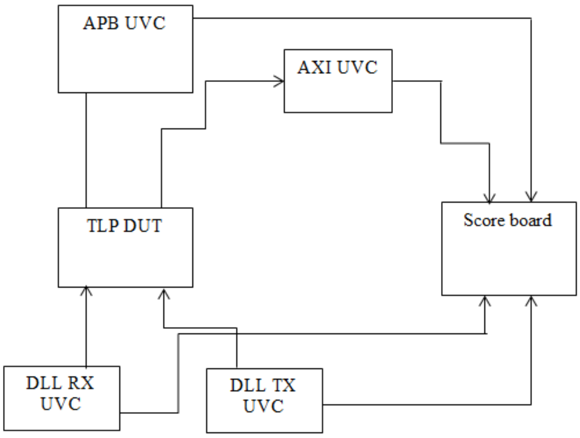


**Figure5**. Test bench Architecture

## 4.SIMULATION RESULTS

Following waveforms can be obtained after passing the memory write_read testcase.Similar results can be obtained by passing different testcases.

Figure 6. shows the write transactions of the AXI, arrow mark reprents the wid and bid values must be same when the awready and awvalid signals are high.Thus the write transactions vcan be verified.

Figure 8. Shows the waveform of APB UVC which represents the information of pwdata, paddr, prdata, pclk, prstn, pready.

Figure 9. Shows the waveform of DLL TX and RX UVC having the information of the tx data and rx data.

Figure 10 shows the different fields present in the TLP header and figure 11 represents the type of TLP verified.

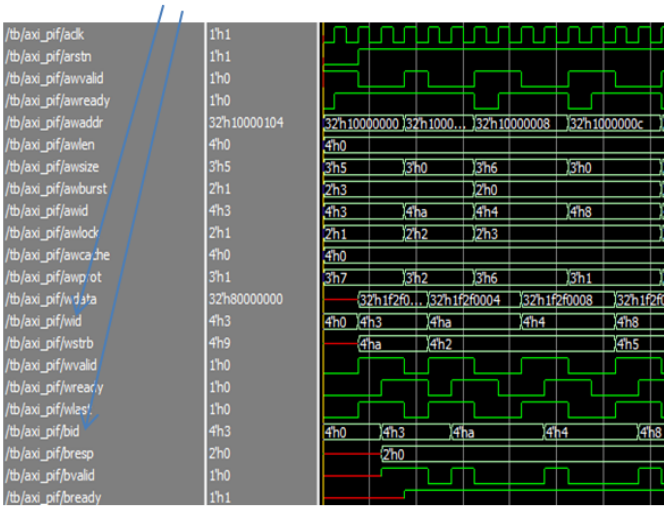


**Figure 6**. Waveform of write transactions

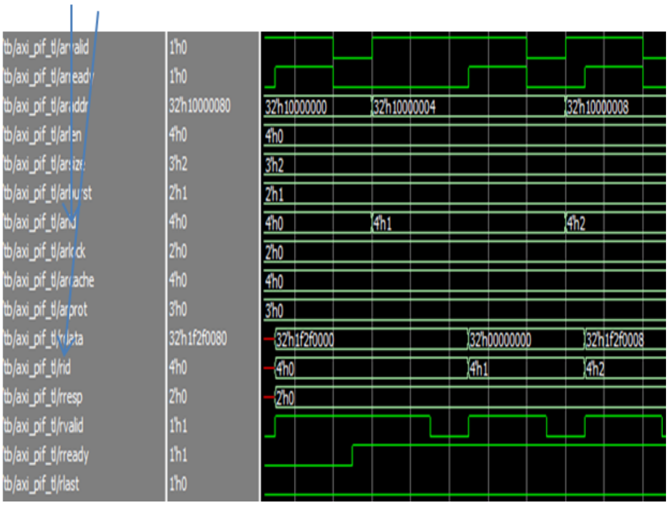


**Figure 7**. Waveform of read transactions

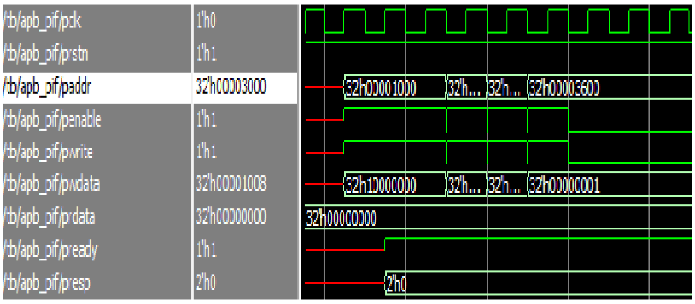


**Figure 8**. Waveform of APB UVC

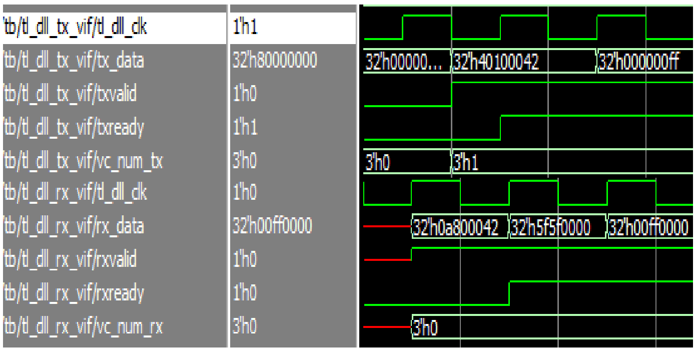
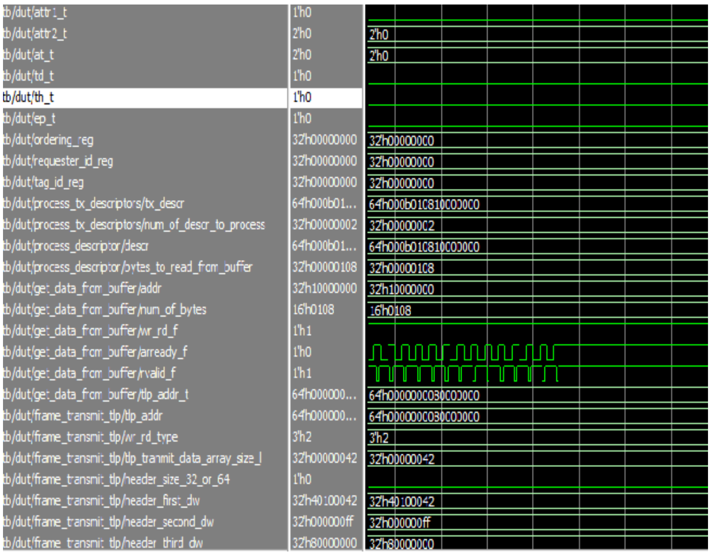
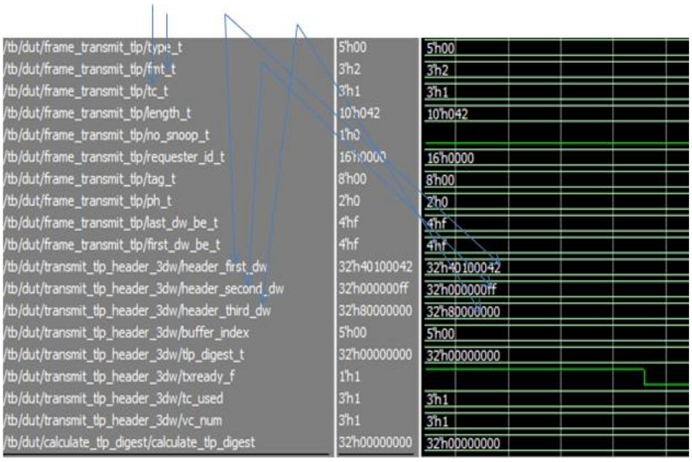


**Figure 9**. Waveform of DLL TX and RX UVC



```
tb/dut/attr1_t                                             1'h0
tb/dut/attr2_t                                             2'h0          2'h0
tb/dut/at_t                                                2'h0          2'h0
tb/dut/td_t                                                1'h0
tb/dut/th_t                                                1'h0
tb/dut/ep_t                                                1'h0
tb/dut/ordering_reg                                        32'h00000000  32'h00000000
tb/dut/requester_id_reg                                    32'h00000000  32'h00000000
tb/dut/tag_id_reg                                          32'h00000000  32'h00000000
tb/dut/process_tx_descriptors/tx_descr                     64'h000b01... 64'h000b010810000000
tb/dut/process_tx_descriptors/num_of_descr_to_process      32'h00000002  32'h00000002
tb/dut/process_descriptor/descr                            64'h000b01... 64'h000b010810000000
tb/dut/process_descriptor/bytes_to_read_from_buffer        32'h00000108  32'h00000108
tb/dut/get_data_from_buffer/addr                           32'h10000000  32'h10000000
tb/dut/get_data_from_buffer/num_of_bytes                   16'h0108      16'h0108
tb/dut/get_data_from_buffer/wr_rd_f                        1'h1
tb/dut/get_data_from_buffer/arready_f                      1'h0
tb/dut/get_data_from_buffer/rvalid_f                       1'h1
tb/dut/get_data_from_buffer/tlp_addr_t                     64'h000000... 64'h0000000030000000
tb/dut/frame_transmit_tlp/tlp_addr                         64'h000000... 64'h0000000030000000
tb/dut/frame_transmit_tlp/wr_rd_type                       3'h2          3'h2
tb/dut/frame_transmit_tlp/tlp_tranmit_data_array_size_l    32'h00000042  32'h00000042
tb/dut/frame_transmit_tlp/header_size_32_or_64             1'h0
tb/dut/frame_transmit_tlp/header_first_dw                  32'h40100042  32'h40100042
tb/dut/frame_transmit_tlp/header_second_dw                 32'h000000ff  32'h000000ff
tb/dut/frame_transmit_tlp/header_third_dw                  32'h80000000  32'h80000000
```

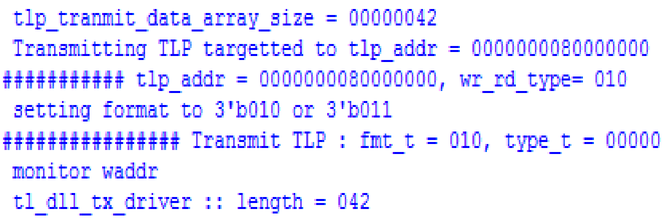
**Figure 10**. Waveform of different fields of TLP



```
/tb/dut/frame_transmit_tlp/type_t                          5'h00         5'h00
/tb/dut/frame_transmit_tlp/fmt_t                           3'h2          3'h2
/tb/dut/frame_transmit_tlp/tc_t                            3'h1          3'h1
/tb/dut/frame_transmit_tlp/length_t                        10'h042       10'h042
/tb/dut/frame_transmit_tlp/no_snoop_t                      1'h0
/tb/dut/frame_transmit_tlp/requester_id_t                  16'h0000      16'h0000
/tb/dut/frame_transmit_tlp/tag_t                           8'h00         8'h00
/tb/dut/frame_transmit_tlp/ph_t                            2'h0          2'h0
/tb/dut/frame_transmit_tlp/last_dw_be_t                    4'hf          4'hf
/tb/dut/frame_transmit_tlp/first_dw_be_t                   4'hf          4'hf
/tb/dut/transmit_tlp_header_3dw/header_first_dw            32'h40100042  32'h40100042
/tb/dut/transmit_tlp_header_3dw/header_second_dw           32'h000000ff  32'h000000ff
/tb/dut/transmit_tlp_header_3dw/header_third_dw            32'h80000000  32'h80000000
/tb/dut/transmit_tlp_header_3dw/buffer_index               5'h00         5'h00
/tb/dut/transmit_tlp_header_3dw/tlp_digest_t               32'h00000000  32'h00000000
/tb/dut/transmit_tlp_header_3dw/txready_f                  1'h1
/tb/dut/transmit_tlp_header_3dw/tc_used                    3'h1          3'h1
/tb/dut/transmit_tlp_header_3dw/vc_num                     3'h1          3'h1
/tb/dut/calculate_tlp_digest/calculate_tlp_digest          32'h00000000  32'h00000000
```

**Figure 11**.Waveform of different fields of TLP and representing the type of TLP Verified.

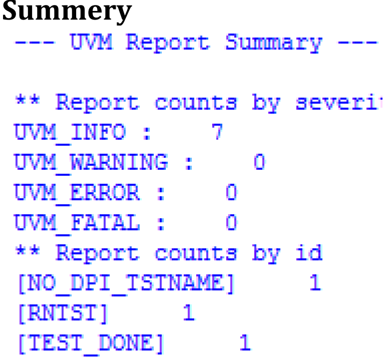
## 4.1Transcript window result of the test case passed



```
 tlp_tranmit_data_array_size = 00000042
 Transmitting TLP targetted to tlp_addr = 0000000080000000
########### tlp_addr = 0000000080000000, wr_rd_type= 010
 setting format to 3'b010 or 3'b011
################ Transmit TLP : fmt_t = 010, type_t = 00000
 monitor waddr
 tl_dll_tx_driver :: length = 042
```

This represents the transcript window result , in which it can be seen that the targeted test case is write read type having the type_t=00000 and fmt=010 hence the write request TLP is verified and fmt=010 represents the TLP format having three DW header with data as represented by the arrow as shown in figure 11.

## 4.2 Transcript window result of UVM Report Summery



```
 --- UVM Report Summary ---

 ** Report counts by severi
 UVM_INFO :     7
 UVM_WARNING :     0
 UVM_ERROR :     0
 UVM_FATAL :     0
 ** Report counts by id
 [NO_DPI_TSTNAME]      1
 [RNTST]      1
 [TEST_DONE]      1
```

This represents the UVM report summery having the information likes zero errors, zero warnings and one test done for the targeted test case.

## 5. CONCLUSION

PCI Express Transaction Layer supports the split transaction. These are verified by using AXI UVC where write and read transactions are verified and also the targeted test cases are generated for targeted TLPs which have been verified. In this way constrained random verification of PCI Express Transaction Layer is done.

## REFERENCES

[1] PCI Express Base Specification Revision3.0 November10-2010

[2] AMBA Specification (Rev 2.0)

[3] AMBA AXI and ACE Protocol Specification

[4] P. Naveen Kalyan, K. Jaya Swaroop, “AMBA-AXI protocol verification by using UVM**”,** International Journal of Electronics and Communication Engineering and Technology (IJECET) Volume 7, Issue 4, July-August 2016

[5] Min-An Song Lee-Ming Institute of Technology Department of Electronic Engineering “System Level Assertion-Based Verification Environment for PCI/PCI-X and PCI-Express” 2007 International Conference on Computational Intelligence and Security

[6] Universal Verification Methodology (UVM) Class Reference June 2011