International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 05 | May 2021 www.irjet.net p-ISSN: 2395-0072

VERIFICATION OF FIRMWARE CONTROLLED NVME HOST

Bishwapa Sanyal[1], Namita Palecha[2]

[1]UG Student, Department of Electronics and Communication, RV College of Engineering, Bengaluru, India.
[2]Assistant Professor, Department of Electronics and Communication, RV College of Engineering, Bengaluru, India.

Abstract - *NVMe™ (Non Volatile Memory Express®) is a new storage access and transport protocol for flash and next-generation solid-state drives (SSDs) that delivers the highest throughput and fastest response times. The NVMe protocol accesses flash storage via a PCIe bus, which is much faster than hard disks and traditional flash architectures and delivers high bandwidth. Functional Verification is a task that ensures the implemented design conforms to the specification. The Universal Verification Methodology (UVM) is a standardized methodology for verifying integrated circuit designs and assembling test environments utilizing constrained random stimulus generation and functional coverage methodologies of SystemVerilog. The primary objective of this paper is to achieve a complete functional verification of the NVMe Soft host and ensuring that the subsystem meets all the requirements and features as described in the specification document. In this paper, the NVMe soft host implements an internal PCIe Device that is intended to be able to receive TLPs routed from other switch ports, and to generate TLPs destined for PCIe targets. Universal Verification Methodology (UVM) is used for verification and simulations are carried out using Cadence NCsim tool.*

Key Words: **Non-Volatile Memory Express (NVME), Peripheral Component Interconnect Express (PCIE), Functional Verification, Universal Verification Methodology (UVM), Layered Testbench Architecture.**

1. INTRODUCTION

Functional verification is defined as a process in which the functional correctness of a design is determined with respect to the specifications of the design. It verifies that the RTL design meets the specifications from a functional perspective. It assumes that the design specification is correct and cannot confirm the correctness of the design specifications. The fundamental purpose of functional verification is that it ensures the implemented design conforms to the specification and the failures are detected. The identified bugs need to be corrected before the design gets shipped to costumer.

The UVM is an open source standardized method developed and maintained by Accellera which is used for the verification of IC designs. It is mainly used for the purpose of universal Verification IP interoperability. UVM is closely related OVM from which it is mainly derived to a large extent which is further related to the eRM Language. The UVM class library has many modifications as compared to the standard SystemVerilog language like in packages, formation of sequences etc. UVM is developed based on an Accellera standard and supports several company vendors like Xilinx Simulator(XSIM), Aldec, Mentor Graphics, Cadence, Synopsys unlike the previous methodologies. The UVM Class Library also provides configuration database, component hierarchy, transaction library model (TLM), etc. This enables the user to virtually create any structure required for the testbench. The main objective of the Universal Verification Methodology (UVM) is to improve productivity of the design by providing a platform that facilitates easier verification of the design components with a standardized representation which can be used with different verification tools as required.

NVM Express (Non-volatile Memory Express) commonly known as NVMe is a logical-device interface specification and transport protocol which can access solid-state drives (SSDs) and flash devices. Hence it is used for accessing a computer's non-volatile storage media. The advantages of NVMe are it delivers highest throughput and fastest response times. PCI Express (PCIe) bus can be used for the access of flash storage in NVMe protocol via a which helps it to deliver a high-bandwidth and low-latency. NVMe can be used for SSDs, multicore CPUs and gigabytes of memory. NVMe is used in any fields for various applications like AI, big data, advanced analytics apps and ML and in real-time interaction platforms of e-commerce, finance, sales and DevOps as it can complete more number of iterations in lesser time. NVMe devices are generally available in the form of standard-sized PCIe expansion cards which uses a U.2 connector and provides a four-lane PCIe interface. The Fig - 1 shows an Intel P3608 NVMe flash SSD, PCI-E add-in card.

PCI Express (Peripheral Component Interconnect Express) is also known as PCIe or PCI-e. It is a serial point-to-point protocol which has a very high speed capacity. It is better as compared to the older bus standards like AGP, PCI and PCI-X bus and has individual serial interfaces which connects every device or the endpoint to the Root complex. It has many widespread applications like in personal computers(PC), hard adapters of disk drives, graphics cards, Ethernet, Wi-Fi, etc. Some of the advantages of PCIe over the older standards are lesser number of I/O pins, maximum bus capacity is higher and scales better when connected to other bus devices. It also has an AER mechanism for error detection and an elaborate report generation. The specifications of the format are made by the PCI-SIG.

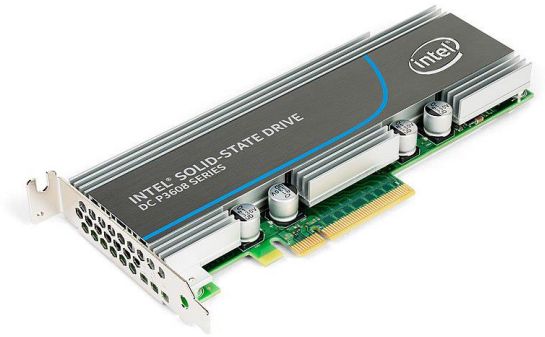

Fig -1: Intel P3608 NVMe flash SSD, PCI-E add-in card (source: https://en.wikipedia.org/wiki/PCI_Express)

2. LITERATURE SURVEY

The document [1] describes the hardware independent firmware interface for managing PCI, PCI-X, and PCI Express™ systems in a host computer. The authors of [2] describe all verification features of the SystemVerilog language. The advantages and disadvantages of different language features along with descriptions of UVM features are also given. The authors of [3] have developed a scheme to avoid the timing out of sync problem when PCI Bus transmits data. This has the capability to solve the problem of high-speed data transmission between Field Programmable Gate Array (FPGA) and PC. In [4], a few approaches to minimize channel impedance discontinuity and near-end/far-end crosstalk (NEXT/FEXT) are discussed. Two channels with optimal design practices and regular design practices are also compared and contrasted. In [5], the verification of the PCI Express Gen5.0 transactions between MAC and PHY layer is proposed in the work. The RTL of PCI Express Gen5.0 is designed in SystemVerilog language and Universal Verification Methodology (UVM) is used for verification. Design and implementation of the interconnection network using PCI Express is proposed in paper [6]. The accuracy of the initial model is also verified. In paper [7], a universal verification methodology based verification environment for PCIe data link layer is built. The coverage goals are achieved by enhancing the performance of verification using methods such as reusability, overriding mechanisms. The authors of [8] have verified the performance of SOC on a dedicated channel between peripheral component interconnect express (PCI-e) end point and memory. The authors of [9] have reviewed the different types of bus architectures (AGP, PCI, PCI-X, PCIe). Also the description of how data transfer takes place between the CPU to the destination in PCIe architecture is given in this paper. Design and verification of several blocks of physical layer for PCI Express and USB is done in paper [10].

3. OVERVIEW OF PCIe

3.1 PCIe Topology

In PCIe, a simple tree topology is present to facilitate compatibility with older versions of PCI. The A simple PCIe topology is shown in Fig- 2. CPU is at the top of the PCIe topology hierarchy.

Root complex is present at the root if the tree of the PCIe hierarchy is inverted. It communicates as the CPU with the other components of the system. It behaves as an interface that might be present between PCIe buses like processor interface, DRAM interface and CPU.

Endpoints exist at the root of the branches of tree hierarchy. They act as the initiators and completers on the bus for various types of transactions. They implement only a single upstream port and no downstream port. Legacy PCIe Endpoint is a device that was made when the older bus like PCI-X has to be operated on newer PCIe interface. These devices should have an added PCIe interface.

Switch allows more devices to connect to attach to a single PCIe port. They route the packets and decide the paths in which the packets are directed based on the address. Bridges are used provide an interface from PCIe to other older buses like PCI-X, PCI, etc.

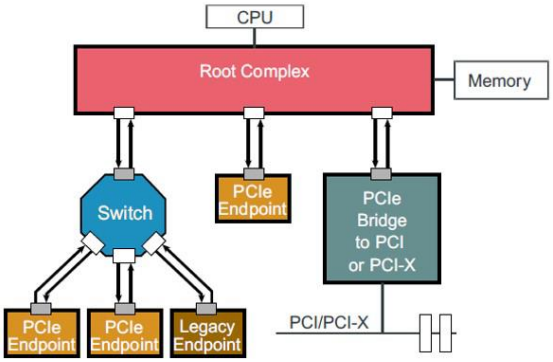

Fig -2: PCIe Topology Example (source: https://www.mindshare.com/Books/Titles/PCI_Express_Technology_3.0)

3.2 PCIe Device Layers

The architecture present in PCIe is layered architecture. The layers operate independently and hence it is easier to adopt to newer specifications. PCIe has three main layers as shown in Fig- 3.

Transaction layer is responsible for Transaction Layer Packet (TLP) creation on the transmit side and TLP decoding on the receive side. It also manages flow control and transaction reordering.

Data Link Layer is responsible for Data Link Layer Packet (DLLP) creation on the transmit side and DLLP decoding on the receive side. It is also responsible for link error detection and correction. It processes the TLP received from transaction layer and forwards it to the physical layer after addition of CRC and some other bits.

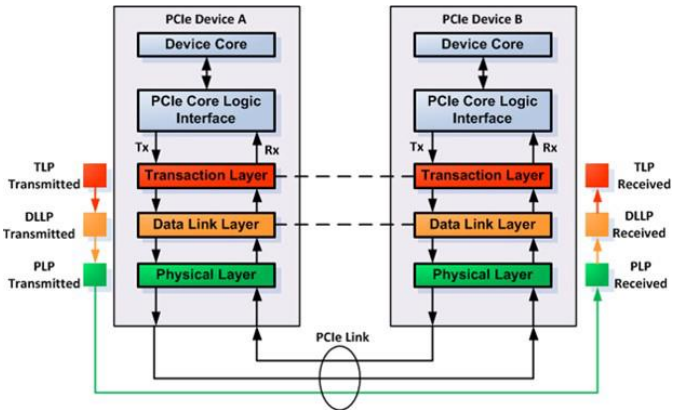

Fig -3: PCIe Device Layers (source: https://www.mindshare.com/Books/Titles/PCI_Express_Technology_3.0)

Physical Layer processes all the different types of packets (TLP, DLLP) transmitted to or received from the link. It differentially clocks out the packets at the link speed and receives it on the other side. The link initialization happens according to the Link Training and Status State Machine (LTSSM).

3.3 PCIe Packet Types

All PCIe TLPs falls in either of the two categories, posted or non-posted TLP type. Posted TLPs may contain data request in the TLP but do not return any completion packet to the requester after the transaction is complete. For Non-posted TLPs, the requester waits for the completion TLP to confirm the completion of the data or control information transfer. The Memory write (MWr) and Message with and without data (Msg/MsgD) TLPs in PCIe are posted. Other TLPs like Configuration read/write (CfgRd/CfgWr), Memory read (MRd) and completion TLPs are non-posted type.

4. BASIC METHODOLOGY

In order to design a complete testbench environment for functional verification of NVMe, the primary task is to design a flowchart for the verification methodology. The flow of verification is shown in Fig- 4. The individual steps in the verification flow shown in Fig- 4 are described below:

1. The first step in the verification flow is to understand the specifications of Design under Test to be verified.
2. After understanding the specifications, the next step is to prepare a test case document. This test case document should contain all the test cases possible for the design.
3. Once the test case document covers more than three-fourth of the functionalities, the preparation of testbench architecture document is started.
4. The main components of the testbench architecture are designed. Some of the components are Transaction Generator, Agent, Driver, Monitor, Scoreboard, Checkers, etc.
5. After the documents are prepared, the coding for each of the components in both the documents are done.

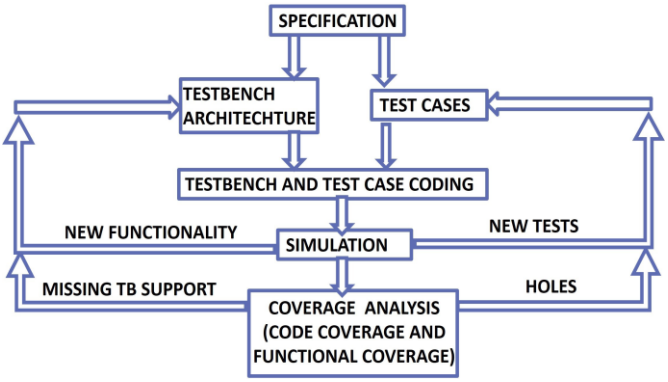

Fig -4: Basic Methodology Flowchart

6. Then simulation is carried out for the environment.
7. If any new tests are found to be added, then it is added in the test cases document whereas if any new functionality is found to be added, then it is added in the testbench architecture document. After addition it has to be coded again and simulated.
8. After the simulation is completed without any errors, the coverage analysis is performed.
9. While the automatic tests are executed, the code coverage calculates the number of lines or branch or expressions evaluated and executed. Code coverage includes Line Coverage, Branch Coverage, Expression Coverage, Toggle Coverage and FSM Coverage.
10. If any holes are found in the coverage model, then the required test cases are added in the test cases document whereas if any missing testbench support is found the coverage model, then the required testbench is added in the testbench architecture document. After addition it is coded, simulated and coverage analysis is performed again.
11. If the coverage report is in the acceptable range, then the verification flow is terminated otherwise modifications are performed to bring the report to acceptable range.

4.1 Testbench Architecture

The testbench architecture used for the verification of NVMe soft host is shown in Fig -5 below.

For inbound path, the agent on the right side allocates TLP space in slave memory and captures write transactions and stores the TLPs into respective FIFOs based on address ranges. For the outbound path, it writes TLPs into Slave memory and captures read transactions to verify addresses.

The slave memory has separate space for TLP and MFA. MFA contains information about the TLP sent like address, TLP type, TLP length, etc.

The driver on the left side will drive inbound data traffic and outbound credit updates and the monitor captures outbound data traffic and inbound credit updates.

The Scoreboards are used to monitor the TLPs and credits. The reference models are used to check the inbound and outbound traffic.

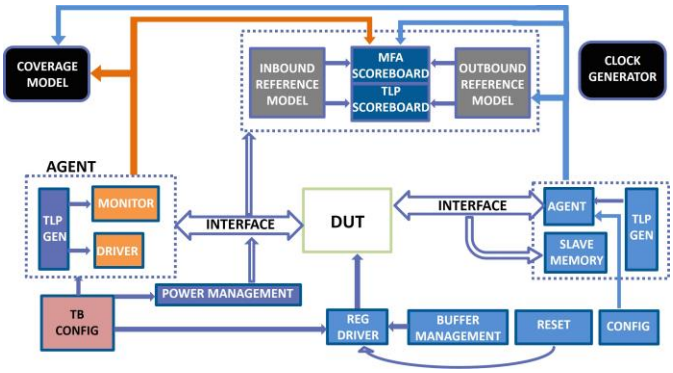

Fig -5: Testbench Architecture

4.2 Test Case Coding

For coding each of the test cases for verification, a standard flow is followed. The Standard UVM phases are considered while writing the flow. The general flow for coding test cases is given below in Fig- 6.

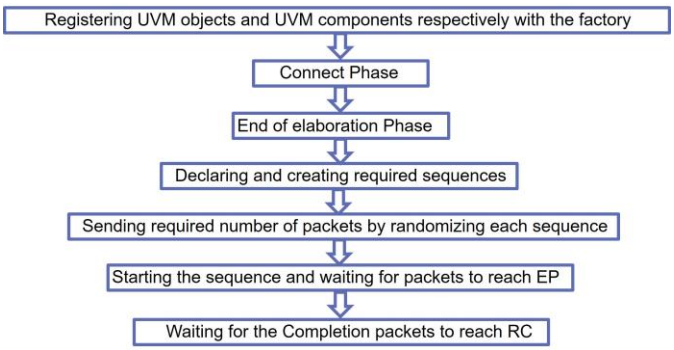

Fig- 6: Flow of test case coding

5. RESULTS

The verification is done for most of the TLP types like configuration read (CfgRd), memory read (MRd), memory write (MWr), message with and without data (Msg/MsgD) and Completion (CplD). The waveform for the simulated test is shown below in Fig- 7. At the bottom of the waveform window, the edge counts for both TX packtes and RX packets are also shown. The LTSSM states are also visible. The transactions start only after the LTSSM is in L0 and Data Link Layer is up. This verifies that the test is functioning as required. The test cases are simulated using Cadence NCsim tool and Universal Verification Methodology (UVM) is used for verification.

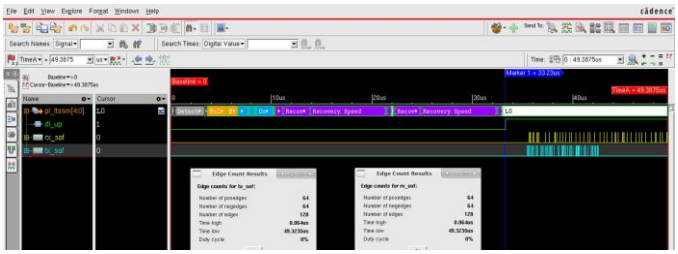

Fig- 7: Waveform obtained after simulation

6. CONCLUSIONS

The paper mainly focuses on the verification of NVME soft host. The primary objective of the proposed work is to develop a complete testbench environment for functional verification of the NVMe Soft host (DUT) using SystemVerilog and Universal Verification Methodology (UVM). The secondary objective is to verify that the subsystem meets all the requirements and features as mentioned in the specification document.

The testbench is designed keeping in mind all the crucial components of a general UVM layered testbench and test case coding is done with the SystemVerilog and UVM. The simulations obtained using the Cadence NCsim reveal that the TLPs are successfully transmitted and received and the functionality is successfully acheived. The states achieved by the LTSSM are also shown. Hence the design under test is successfully verified in the given work.

REFERENCES

[1] PCI Firmware Specification Revision 3.3, https://pcisig.com/specifications, 2021.

[2] Christian B. Spear, “SystemVerilog for Verification: A guide to learning the Test Bench Language Features”, 3rd Edition, Springer Publications, 2012.

[3] M. Vasa, C. -L. Liao, S. Kumar, C. -H. Chen and B. Mutnury, "PCIe Gen-5 Design Challenges of High-Speed Servers," IEEE 29th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), 2020, San Jose, CA, USA, pp. 1-3, doi:10.1109/EPEPS48591.2020.9231458

[4] Y. Tao et al., "Design and implementation of high speed encryption and decryption system based on PCIE bus,“ IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT), Weihai, China, 2020, pp. 369-372, doi: 10.1109/ICCASIT50869.2020.9368599.

[5] G. Rohilla, D. Mathur and U. Ghanekar, "Functional Verification of MAC-PHY Layer of PCI Express Gen5.0 with PIPE Interface using UVM," International Conference for Emerging Technology (INCET), Belgaum, India, 2020, pp. 1-5, doi: 10.1109/INCET49848.2020.9154176.

[6] Shim, C., Cha, Kh. & Choi, M., “Design and implementation of initial OpenSHMEM on PCIe NTB

based cloud computing", Cluster Comput 22, Springer, 2019, https://doi.org/10.1007/s10586-018-1707-0.

[7] Dr. T. C. Thanuja , Akshata, "Universal verification methodology based verification Environment for PCIe data link layer" International Research Journal of Engineering and Technology (IRJET), Vol: 04, 2017.

[8] S. R. Mantripragada and P. Mopuri, "Verifying performance of PCI express in a system for multi giga byte per second data transmission," International Conference on Communication and Electronics Systems (ICCES), 2017 Coimbatore, India, pp. 1-5, doi: 10.1109/CESYS.2016.7889889.

[9] Pamula, Vinay Kumar & Mantripragada, Sai., "Implementation and verification of PCI express interface in a SoC", IEICE Communications Express, 2017, 6. 10.1587/comex.2017XBL0056.

[10] Verma, Anuj & Dahiya, Pawan, "PCIe BUS: A State-of-the-Art-Review", IOSR Journal of VLSI and Signal Processing (IOSR-JVSP), 2017. 7. 24-28. 10.9790/4200-0704012428.

[11] Richard A Prasad, Madhusudan Kulkarni, "Design and verification of phy interface for PCIe GEN 3.0 and USB gen 3.1 using uvm methodology," International Research Journal of Engineering and Technology (IRJET), 2017, Volume: 04 Issue: 10

[12] M. AbdElSalam, "NVMe Solid State Drive verification solution using HW Emulation and Virtual Device Technologies," 11th International Design & Test Symposium (IDT), Hammamet, Tunisia, 2016, pp. 47-52, doi: 10.1109/IDT.2016.7843013.

[13] Y. T. Jin, S. Ahn, and S. Lee, \Performance analysis of nvme ssd-based all-flash array systems," in 2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2018, pp. 12-21. doi: 10.1109/ISPASS.2018.00010.

[14] V. K. Pamula and S. Mantripragada, \Implementation and verication of pci express interface in a soc," IEICE Communications Express, vol. 6, Jun. 2017. doi:10.1587/comex.2017XBL0056.

[15] W. Ni and J. Zhang, Research of reusability based on uvm verication," in 2015 IEEE 11th International Conference on ASIC (ASICON), 2015, pp. 1{4. doi: 10.1109/ASICON.2015.7517189.