INTRODUCTION
Non-Volatile Memory Express® (NVMe®) is a new software interface optimized for PCIe® Solid State Drives (SSD). It significantly improves both random and sequential performance by reducing latency, enabling high levels of parallelism, and streamlining the command set while providing support for security, end-to-end data protection, and other client and enterprise features. NVMe provides a standards-based approach, enabling broad ecosystem adoption and PCIe SSD interoperability.

This article provides an overview of the NVMe specification and examines some of its key features. We will discuss its pros and cons, compare it to conventional technologies, and point out key areas to focus on during its verification. In particular, we will describe how NVMe Questa® Verification IP (QVIP) effectively contributes and accelerates verification of PCIe-based SSDs that use NVMe interfaces.

TECHNOLOGY OVERVIEW
The NVMe specification by the NVM Express Workgroup is an optimized, high performance, scalable host interface for connecting SSDs in personal computers. However, SATA was designed primarily for interfacing with mechanical hard disk drives (HDDs) and has become increasingly inadequate as SSDs have improved. For example, unlike hard disk drives, some SSDs are limited by the maximum throughput of SATA.

SSDs based on NAND flash memory deliver revolutionary performance versus HDDs: a good client SSD achieves 20,000 input/output operations per second; whereas an HDD is limited to about 500 IOPS on a good day.

A client SSD can theoretically deliver more than 1GBps in read throughput. However, SSDs have been bottlenecked by the volume client interface, SATA v3.0, which has a maximum speed of 600 MBps. Scaling SATA substantially beyond this requires multi-lane support, which involves chipset and connector changes and potentially a modified software stack.

To enable higher-speed client SSDs in the near term, the only choice was PCIe, which supports up to 1GBps per lane and has software-transparent multi-lane support for scalability (×2, ×4, ×8, and so on).

These limitations motivated development of a new software interface optimized for PCIe SSDs, NVMe.
controller interface with a streamlined register interface and command set designed for enterprise, datacenter, and client systems that use non-volatile memory (NVM) storage.

An NVMe controller is associated with a single PCI function. The capabilities and settings that apply to the entire controller are indicated in the controller capabilities (CAP) register and the identify controller data structure.

NVMe is based on a paired submission and completion queue mechanism. Commands are placed by host software into a submission queue. Completions are placed into the associated completion queue by the controller. Multiple submission queues may utilize the same completion queue. Submission and completion queues are allocated in memory.

An admin submission and the associated completion queue exist for the purpose of controller management and control (e.g., creation and deletion of I/O submission and completion queues, aborting commands, etc.). Only commands that are part of the admin command set may be submitted to the admin submission queue.

The interface provides optimized command submission and completion paths. It includes support for parallel operation by supporting up to 65,535 I/O queues with up to 64K outstanding commands per I/O queue.

Some of the key NVMe features are:
- Does not require un-cacheable / MMIO register reads in the command submission or completion path
- A maximum of one MMIO register write is necessary in the command submission path
- Support for up to 65,535 I/O queues, with each I/O queue supporting up to 64K outstanding commands
- Priority associated with each I/O queue with well-defined arbitration mechanism
- All information to complete a 4KB read request is included in the 64B command itself, ensuring efficient small I/O operation
- Efficient and streamlined command set
- Support for MSI/MSI-X and interrupt aggregation
- Support for multiple namespaces
- Efficient support for I/O virtualization architectures, like SR-IOV
- Robust error reporting and management capabilities
- Support for multi-path I/O and namespace sharing
- Developed by open industry consortium

The following figure explains the steps involved in NVMe command processing.

**Figure 3: Queue pair example, n:1 mapping**

**Figure 4: Command processing**
NVME QVIP FEATURES

1. Plug and play
In order to make the NVMe QVIP work out-of-the-box and easier to deploy, a number of integration kits are available for some commonly used PCIe design IP. These quick starter kits can be easily downloaded for serial or pipe link and are pre-tested with commonly used IP vendors. These kits are basically a reference resource to allow smooth integration of DUTs with UVM testbenches having QVIP and running first stimulus.

Also, single and multiple NVMe controller use-case examples are part of the standard deliverable database.

2. In-depth protocol feature support
NVMe QVIP provides extensive protocol feature support, allowing a wider degree of stimuli generation. Some of the supported features are:

- Multipath IO and namespace sharing
- Configurable queue depth, number and n:1 queue pairing
- Pin, MSI, and MSI-X interrupts with/without masking

3. Multiple controllers support
NVMe QVIP supports multiple controller operations. Since each PCIe physical or virtual, function, single root I/O virtualization (SR-IOV) can act as an NVMe controller, NVMe QVIP as a host can generate stimuli on the target controller or respond as a controller based on its location in the bus topology (bus, device, function number). In Figure 6 below is an example of multiple controllers attached to virtual functions of a physical function.

4. PI based stimuli
NVMe QVIP provides a rich set of APIs for easier stimuli generation. These APIs can be categorized as transactional, attribute/command specific, and utility based. They provide a simple interface to initiate traffic from a higher abstraction (transactional) or precise control at the command (command specific) or byte levels. Users can modify stimulus at the byte abstraction level for intentional error injection.
5. UVM based register modeling

NVMe QVIP adopts UVM register modeling concepts and provides a way of tracking the register content of a DUT and a convenience layer for accessing register and memory locations within the DUT. In order to support the use of the UVM register package, QVIP has an adapter class, which is responsible for translating between the UVM register packages generic register sequence items and the PCIe bus specific sequence items (Figure 8).

A register model matching DUT properties can be built using register generator applications like Questa’s UVM register assistant or be handwritten in the csv format as shown in Figure 9 on the following page.

Register-based modeling allows users to write reusable sequences that access hardware registers and areas of memory. It is organized to reflect the DUT hierarchy and makes it easier to write abstract and reusable stimuli in terms of hardware blocks, memories, registers, and fields rather than working at a lower bit pattern level of abstraction. The model contains a number of access methods which sequences use to read and write registers.

The UVM package also contains a library of built-in test sequences which can be used to do most of the basic register and memory tests, such as checking register reset values and checking the register and memory data paths.
6. UVM reporting based protocol checker
To prove protocol compliance of the DUT, VIP must provide protocol checking. NVMe QVIP has built-in protocol checkers that continuously monitor ongoing bus traffic for protocol compliance and flashes a UVM message when there is undesired activity. These protocol checks cover the NVMe specification in depth and can be individually enabled or disabled at run time.

7. Easy to read text-based logs
NVMe related protocol traffic is captured in real time from the PCIe bus and printed in a well formatted, easy-to-read view. Each column in the log can be enabled or disabled for the test case and provides valuable input for post simulation debug.
PCle transaction-layer packet logging can also be enabled from a test.

8. Flexibility and reusability
NVMe QVIP can be integrated with PCIe VIP since it uses flexible and configurable generic APIs to interact with PCIe BFM.

These APIs are also protocol independent and allow users to extend default behavior and implement user specific functionalities.

This provides hooks to use for implementing custom behavior or upcoming standards/extensions; such as the NVMe management interface and NVMe over fabrics, etc.

9. Sample functional coverage and compliance test suite
Since coverage driven verification is a natural complement to constrained random verification (CRT), QVIP comes with a sample functional coverage plan and integrated coverage model. It helps in making decisions regarding “are we done yet” with stimuli generation. NVMe coverage matrices are supplied with “must have” and “nice to have” coverpoints and crosses.

Questa VIP is also accompanied with test suite targeting scenarios as specified by industry leading IOL’s NVMe Conformance Test suite v1.2. These sequences are pre-tested and can be used as is within user simulation environments.

CONCLUSION
NVMe is designed to optimize the processor’s driver stack so it can handle the high IOPS associated with flash storage. It also puts the SSD close to the server chipset, reducing latency and further increasing performance while keeping the traditional form factor that IT managers are comfortable servicing. It brings PCIe SSDs into the mainstream with a streamlined protocol that is efficient and scalable and with an ease of deployment through industry standard support.
NVMe QVIP perfectly complements verification engineering efforts for timely design closure by providing easy stimuli generation and debugging interfaces, so they can focus more on protocol-level intricacies.

REFERENCES

1. NVM Express revision 1.2 specification
2. Serial ATA Advanced Host Controller Interface (AHCI) 1.3.1 specification
3. PCI Express® Base Specification Revision 4.0 Version 0.5
4. Register Package guide for UVM, COOKBOOK
8. “The Nonvolatile Memory Transformation of Client Storage”, Amber Huffman and Dale Juenemann, Intel, 2013 IEEE, Published by the IEEE Computer Society
The Most Comprehensive Resource for Verification Training

25 Video Courses Available Covering
- Formal Verification
- Intelligent Testbench Automation
- Metrics in SoC Verification
- Verification Planning
- Introductory, Basic, and Advanced UVM
- Assertion-Based Verification
- FPGA Verification
- Testbench Acceleration
- PowerAware Verification
- Analog Mixed-Signal Verification

UVM and Coverage Online Methodology Cookbooks
Discussion Forum with more than 6500 topics
UVM Connect and UVM Express Kits

www.verificationacademy.com