
# Using Functional Verification in Testing NVMe SSD Controller Designs

August 2019

Vikas Tomar

Product Engineer

Questa Verification IP


# Agenda

- Functional verification
  - Definition
  - Methods
- NVMe SSD Controller
  - Typical Communication in NVMe Controller
  - What needs verification
- NVMe Controller Verification Challenges
  - Controller configuration
  - Extensive features
  - Conformance and interoperability
  - Test creation and Debug
- Components of verification solution
  - Test plan
- Technique
  - Effective coverage closure
  - Effective debug


# Functional Verification

- Verify RTL's confirmation to specification
  - "Does design do what is intended"
  - Most time and effort consuming part of Design Verification process
  - Various steps included "None sufficient"

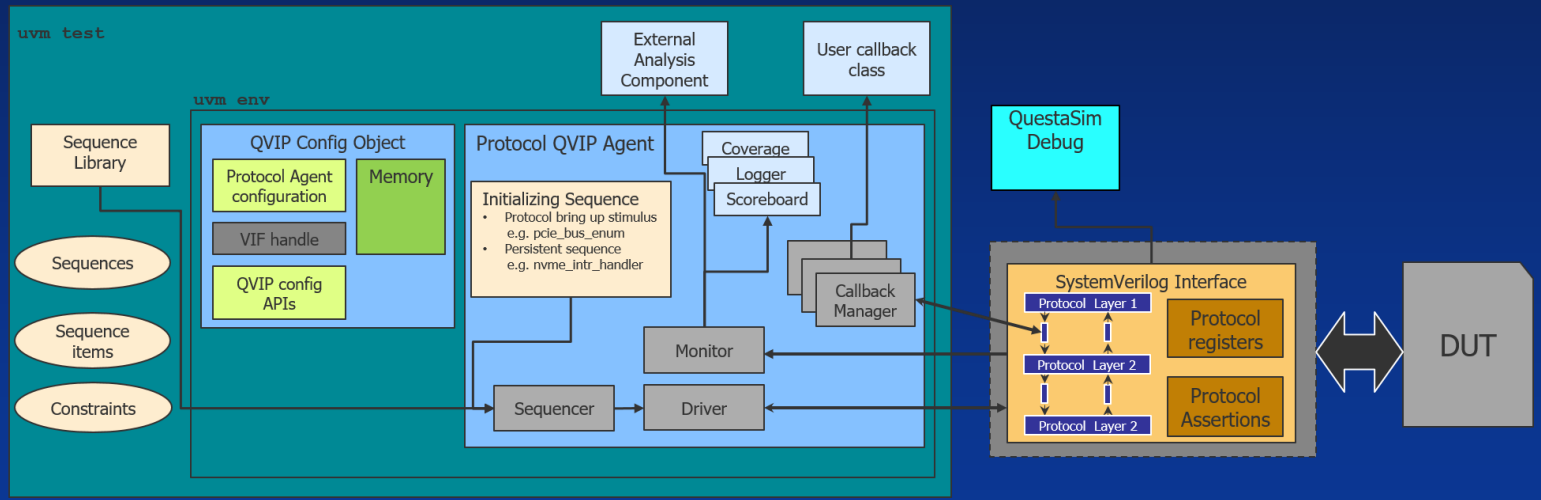
- Methods
  - Simulation
  - Emulation
  - Formal

# Functional Verification -Simulation

- Simulation helps in verifying the design early
  - Major components
    - Testplan => Define verification
    - Stimulus => Generating scenarios(+ive and -ive)
    - Assertions => Protocol adherence
    - Coverage => Verification closure
  - Benefits
    - Start early
    - Standard methodology and verification components available

# Functional Verification -Simulation

- Generic UVM based Simulation Testbench



# NVMe SSD controller

- NVMe controller provides
  - Queue based access to Non volatile media
  - Data transfer is conducted using Register read writes
  - For Data transfer NVMe promises
    - Lower Latency
    - High throughput
    - High number of IOPS
- NVMe SSDs are benchmarked
  - Combination of above under various test loads
- Functional verification for NVMe Controller SSD
  - Specification adherence and competitive performance

# NVMe SSD controller

- Typical communications in NVMe SSD controller
  - PCIe related communication
    - Discovery of controller (PCIe PF and VF)
    - Interrupt management
    - Register implementation and mapping
    - TX and RX data paths
  - Data transfer to the Flash interfaces
  - Data transfer to DDR interfaces (on chip memory)
  - On-chip communications

# What to verify ??

- Complete NVMe SSD subsystem verification can be divided into below categories
  - Link Level verification (PCIe)
    - Interrupts (MSI, MSIx)
    - PCIe power management (Various Power saving states)
    - Resets
      - PCIe and NVMe resets
  - NVMe Controller Register Level verification
    - Register values
    - Action on register access
  - Queue Interface
    - Queue creation/deletion, Doorbell, Empty/Full conditions
    - Queue location and data access
    - Queue starving*
  - Data transfer between Host and controller
    - Data Access direction
    - Extra RD/WR on PCIe interface*



* performance impacts


# What to verify ??

- Command Level
  - Admin and IO command
  - Autonomous commands like Abort, Event notifications
  - Possible completion status
- Data structure access
  - PRP (Offsets for PRP1 and PRP2)
  - SGL (Various Descriptors)
- Data structure Values
  - Identify data structures
  - Name space data structures
  - Log pages access
- Feature verification
- Error handling verification

# NVMe Controller verification Challenges

- Large Configurations space
  - Behavior of a NVMe operation depends on the combination of various parameter
    - SSD Namespace characteristics
    - Controller and Identify data structures
  - Similarly NVMe SSD can show different performance statistics depending upon
    - Feature enabled by host
    - Queues created by host
    - Parameters related to data transfer selected by host
  - The combination of all above parameters can exponentially increase
    - Number of test cases
    - Time and effort
  - Such large combination is very hard to
    - Create and cover with fast deadlines
    - Estimate the verification closure time

# NVMe Controller verification Challenges

- Extensive feature support
  - Almost 40 TPs added in NVMe 1.4 specification
  - 27(Not including Fabrics) number of TP's are in various development stages
  - Challenges:
    - Features affecting existing features
      - Features like CMB and PMR changed the direction of data access.
      - These operation affects the existing test scenarios and expand the verification space
    - Feature verification
      - Each feature requires extensive planning
- Interoperability and Conformance
  - Affected by
    - Large configuration space
    - New features
    - Various platforms

# NVMe Controller verification Challenges

- Stimulus generation
  - With so many parameter in picture
    - Impossible to create directed scenarios
    - Randomization helps but do not solve the problem
      - Commands field interdependency
- Debug
  - Hard to investigate a suspicious transaction
    - Traffic on PCIe bus
    - Data transfer for commands running in parallel for multiple queues
    - Address based transactions
    - Hard to relate a PCIe transaction to a NVMe command.

# Components of Verification solution

- Test plan
- Coverage
- Stimulus
  - Random
  - Feature wise
- Assertions
- Callbacks
- Monitor
- Debugger/Logger


# Test plans

- Test plan Requirement
  - Controller configuration Test plan
    - SSD Name space (NS DS)
    - Command support
  - Host configuration Test plan
    - Covering the possible host configurations
  - NVMe Protocol Events Test Plan
    - Specification mapped feature wise Test plan covering
      - Controller register space field access
      - Queue operations
      - PCIe Features
      - NVMe Features
  - Standard Compliance Testplan

# Sample Test Plan

| Section | Title | Description | Link | Type | Weight | Goal |
|---|---|---|---|---|---|---|
| 1 | **Introduction** | | | | 0 | 0 |
| 2 | **System Bus (PCI Express) Registers** | | | | 0 | 0 |
| 3 | **Controller Registers** | **This section describes the Controller registers that are** | | | 1 | 100 |
| 3.1 | **Register Definition** | | | | 0 | 0 |
| 3.1.1 | Offset 00h: CAP – Controller Capabilities | This register indicates basic capabilities of the controller to host | | | 0 | 0 |
| 3.1.1.1 | Offset 00h: CAP – MPSMAX (RO Bits 55:52) | This field indicates the maximum host memory page size that | ctrl_reg_cvg,*:cap_mpsmax | Coverpoint | 1 | 100 |
| 3.1.1.2 | Offset 00h: CAP – MPSMIN (RO Bits 51:48) | This field indicates the minimum host memory page size that | ctrl_reg_cvg,*:cap_mpsmin | Coverpoint | 1 | 100 |
| 3.1.1.3 | Offset 00h: CAP – CSS (RO Bits 44:37) | This field indicates the I/O Command Set(s) that the controller | ctrl_reg_cvg,*:cap_css | Coverpoint | 1 | 100 |
| 3.1.1.4 | Offset 00h: CAP – NSSRS (RO Bit 36) | This field indicates whether the controller supports the NVM | ctrl_reg_cvg,*:cap_nssrs | Coverpoint | 1 | 100 |
| 3.1.1.5 | Offset 00h: CAP – DSTRD (RO Bits 35:32) | Each Submission Queue and Completion Queue Doorbell | ctrl_reg_cvg,*:cap_dstrd | Coverpoint | 1 | 100 |
| 3.1.1.6 | Offset 00h: CAP – Timeout (RO Bits 31:24) | This is the worst case time that host software shall wait for | ctrl_reg_cvg,*:cap_timeout | Coverpoint | 1 | 100 |
| 3.1.1.7 | Offset 00h: CAP – AMS (RO Bits 18:17) | This field is bit significant and indicates the optional arbitration | ctrl_reg_cvg,*:cap_ams | Coverpoint | 1 | 100 |
| 3.1.1.8 | Offset 00h: CAP – CQR (RO Bit 16) | This field is set to '1' if the controller requires that I/O | ctrl_reg_cvg,*:cap_cqr | Coverpoint | 1 | 100 |
| 3.1.1.9 | Offset 00h: CAP – MQES (RO Bits 15:0) | | | | | |
| 3.1.2 | Offset 08h: VS – Version | | | | | |
| 3.1.3 | Offset 0Ch: INTMS – Interrupt Mask Set | | | | | |
| 3.1.4 | Offset 10h: INTMC – Interrupt Mask Clear | | | | | |
| 3.1.5 | Offset 14h: CC – Controller Configuration | | | | | |
| 3.1.5.1 | Offset 14h: CC - I/O Completion Queue Entry Size (RW bits | | | | | |
| 3.1.5.2 | Offset 14h: CC - I/O Submission Queue Entry Size (RW bit-19:16) | | | | | |
| 3.1.5.3 | Offset 14h: CC - Shutdown Notification (RW bits 15:14) | | | | | |
| 3.1.5.4 | Offset 14h: CC - Arbitration Mechanism Selected (RW bits | | | | | |
| 3.1.5.5 | Offset 14h: CC - Memory Page size (RW bits 10:7) | | | | | |
| 3.1.5.6 | Offset 14h: CC - I/O Command Set Selected (RW bits 6:4) | | | | | |
| 3.1.5.7 | Offset 14h: CC - Enable Set (RW bit 0) | | | | | |
| 3.1.5.8 | Offset 14h: CC - Enable Clear (RW bit 0) | | | | | |
| **4.1** | **Submission Queue & Completion Queue Definition** | **This section describes Head and Tail entry pointers which correspond to the Completion Queue Head Doorbells and the Submission Queue Tail Doorbells registers. The** | | | 1 | 100 |
| 4.1 | Submission Tail doorbell update. | The submitter of entries to a queue uses the current Tail entry pointer to identify the next open queue entry space. The submitter increments the Tail entry pointer after submitting the | mem_strct_cvg,*:updt_tail_doorbell | Coverpoint | 1 | 100 |
| 4.1.01 | IO Submission Queue Tail Entry rollover. | If the Tail entry pointer increment exceeds the queue size, the Tail entry shall roll to zero. | mem_strct_cvg,*:zero_sq_tail_entry | Coverpoint | 1 | 100 |
| 4.1.02 | Completion Queue Head doorbell update. | The consumer of entries on a queue uses the current Head entry pointer to identify the next entry to be pulled off the queue. The consumer increments the Head entry pointer after retrieving the | mem_strct_cvg,*:updt_head_doorbell | Coverpoint | 1 | 100 |
| 4.1.03 | IO Completion Queue Head Entry rollover. | If the Head entry pointer increment exceeds the queue size, the Head entry pointer shall roll to zero. | mem_strct_cvg,*:zero_cq_head_entry | Coverpoint | 1 | 100 |
| 4.1.04 | Order of Submission Queue and Completion Queue creation. | Host software shall create the Completion Queue before creating any associated Submission Queue. Submission Queues may be created at any time after the associated Completion Queue | mem_strct_cvg,*:sq_id_after_cq_id | Coverpoint | 1 | 100 |
| 4.1.05 | Order of Submission Queue and Completion Queue deletion. | Host software shall delete all associated Submission Queues prior to deleting a Completion Queue. (Knowledge of SQ ID associated with CQ ID is must). | mem_strct_cvg,*:cq_id_del_cmd | Coverpoint | 1 | 100 |
| 4.1.06 | Empty Queue | The queue is Empty when the Head entry pointer equals the Tail entry pointer. | mem_strct_cvg,*:empty_queue | Coverpoint | 1 | 100 |
| 4.1.2 | Full Queue | The queue is Full when the Head equals one more than the Tail. The number of entries in a queue when full is one less than the queue size. | mem_strct_cvg,*:full_queue | Coverpoint | 1 | 100 |
| 4.1.3 | Queue Size | | | | 0 | 0 |
| 4.1.4 | Queue Identifier | Each queue is identified through a 16-bit ID value that is assigned to the queue when it is created. | mem_strct_cvg,*:diff_sq_id | Coverpoint | 1 | 100 |
| 4.1.5 | Queue Priority | | | | 0 | 0 |

# Techniques for effective coverage closure

- Testbench configuration:
  - Should be generated using a constrained random class
  - Benefits
    - ✓ Constraints can be used for generating only valid configurations
    - ✓ Controllability to  generated valid number of predictable configurations
    - ✓ Any coverage closure tool can be used to have closure on verification from configuration aspect
- Configurable Stimulus
  - Initialization Sequences
    - Num queues, MPS
    - Queue location
    - Interrupt
  - All Sequences
    - BDF/Controller ID
    - NS ID
  - Callback control for error handling
    - Command
    - Data and data structure

# Techniques for Effective Debug

- Monitor:
  - Should watch address space independently
  - Should check for any unnecessary PCIe RD/WR.
- Logger
  - Should be able to correlate all pcie transactions under single NVMe transaction
  - Should be able to highlight any unknown address access
  - Should show the direction of the transfer
- Performance statistics
  - Latency, throughput and Iops can be calculated.
- Configurable assertion
  - E.g. Assertions can be added for checking latency for a queue entry with timeout

# Loggers

- Intuitive loggers can reduce the debug time.

| DEBUG ID | BDF | S R C | R W | TYPE | SQID | CQID | REG_NAME / QENTRY | CMD / REG_DATA | MISC | CID | PS DT | PRP2 / SGL1[39:32] | PRP1 / SGL1[31:24] | NSID | STS/SLBA/PC | ADDR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ---------- | 0100 | H | R | REG | ---- | ---- | CAP | 0102FFFF | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C00000000000 |
| ---------- | 0100 | H | R | REG | ---- | ---- | | 00100030 | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C00000000004 |
| ---------- | 0100 | H | R | REG | ---- | ---- | CSTS | 00000000 | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C0000000001C |
| ---------- | 0100 | H | R | REG | ---- | ---- | CMBSZ | 00000000 | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C0000000003C |
| ---------- | 0100 | H | W | REG | ---- | ---- | AQA | 000F000F | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C00000000024 |
| ---------- | 0100 | H | W | REG | ---- | ---- | ASQ | 00000000 | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C00000000028 |
| ---------- | 0100 | H | W | REG | ---- | ---- | | 00000010 | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C0000000002C |
| ---------- | 0100 | H | W | REG | ---- | ---- | ACQ | 00000000 | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C00000000030 |
| ---------- | 0100 | H | W | REG | ---- | ---- | | 0000000F | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C00000000034 |
| ---------- | 0100 | H | R | REG | ---- | ---- | CC | 00000000 | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C00000000014 |
| ---------- | 0100 | H | W | REG | ---- | ---- | CC | 00460001 | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C00000000014 |
| ---------- | 0100 | H | R | REG | ---- | ---- | CSTS | 00000000 | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C0000000001C |
| ---------- | 0100 | H | R | REG | ---- | ---- | CSTS | 00000001 | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C0000000001C |
| ---------- | 0100 | H | W | REG | ---- | ---- | SQ0TDBL | 00000001 | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C00000001000 |
| 01000000000000000000000001 | 0100 | D | R | ASQ | ---- | ---- | 0 | IDENTIFY | CDS | 0000 | PRP | 0000000000000000 | 0000000000000000 | 00000000 | ---------- | 0000001000000000 |
| 01000000000000000000000001 | 0100 | D | W | CDS | ---- | ---- | -------- | ---------- | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000000000000000 |
| 01000000000000000000000001 | 0100 | D | W | ACQ | ---- | ---- | 0 | ---------- | 00001 | 0000 | --- | ---------- | ---------- | -------- | GEN00 | 0000000F00000000 |
| ---------- | 0100 | D | W | INTR | ---- | ---- | -------- | ---------- | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000001C00000000 |
| ---------- | 0100 | H | W | REG | ---- | ---- | INTMS | 00000001 | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C0000000000C |
| ---------- | 0100 | H | W | REG | ---- | ---- | CQ0HDBL | 00000001 | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C00000001004 |
| ---------- | 0100 | H | W | REG | ---- | ---- | SQ0TDBL | 00000002 | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C00000001000 |
| ---------- | 0100 | H | W | REG | ---- | ---- | INTMC | 00000001 | ----- | ---- | --- | ---------- | ---------- | -------- | ---------- | 0000C00000000010 |

# Performance loggers

- Performance logging

| | |
|---|---|
| TDB TIME | TAIL DOORBELL REGISTER UPDATE TIME |
| FSQE TIME | SQE FETCH TIME |
| TDB2SQE | TAILDOORBELL TO SQE FETCH TIME |
| AVG TDB2SQE | AVERAGE TDB2SQE TIME PER QUEUE |
| IOPS | NUMBER OF IO WR/RD OPERATIONS PER SECOND |
| THROUGHPUT | AMOUNT OF DATA TRANSFERRED IN MBPS |
| TDB2CQE | ROUND TRIP TIME OF IO WR/RD COMMAND |

| QID | PRIORITY | AVG TDB2SQE |
|---|---|---|
| 0000 | URGENT | 396071 |
| 0001 | URGENT | 426875 |

| TDB2CQE(S) | | |
|---|---|---|
| BDF | CMD | AVG. TDB2CQE |
| 0100 | WR | 2427500 |
| 0100 | RD | 1988751 |
| Average IO | | 2208125 |

| DEBUG ID | BDF | QID | PRIORITY | CMD | CID | TDB TIME | FSQE TIME | TDB2SQE |
|---|---|---|---|---|---|---|---|---|
| 01000000000000000000001 | 0100 | 0000 | URGENT | IDENTIFY | 0000 | 141782950 | 142180450 | 397500 |
| 01000000000000000000002 | 0100 | 0000 | URGENT | IDENTIFY | 0001 | 153167950 | 153565450 | 397500 |
| 01000000000000000000003 | 0100 | 0000 | URGENT | IDENTIFY | 0002 | 164747950 | 165142950 | 395000 |
| 01000000000000000000004 | 0100 | 0000 | URGENT | IDENTIFY | 0003 | 176142950 | 176537950 | 395000 |
| 01000000000000000000005 | 0100 | 0000 | URGENT | SET_FR | 0004 | 187527950 | 187922950 | 395000 |
| 01000000000000000000006 | 0100 | 0000 | URGENT | CRE_IO_CQ | 0005 | 190207950 | 190605450 | 397500 |
| 01000000000000000000007 | 0100 | 0000 | URGENT | CRE_IO_CQ | 0006 | 203797950 | 204192950 | 395000 |
| 01000000000000000000008 | 0100 | 0000 | URGENT | CRE_IO_CQ | 0007 | 217325450 | 217722950 | 397500 |
| 01000000000000000000009 | 0100 | 0000 | URGENT | CRE_IO_CQ | 0008 | 230640450 | 231035450 | 395000 |
| | 0100 | 0000 | URGENT | CRE_IO_CQ | 0009 | 242605450 | 243000450 | 395000 |
| | 0100 | 0000 | URGENT | CRE_IO_CQ | 000A | 254620450 | 255015450 | 395000 |
| | 0100 | 0000 | URGENT | CRE_IO_CQ | 000B | 268340450 | 268735450 | 395000 |
| | 0100 | 0000 | URGENT | CRE_IO_CQ | 000C | 281360450 | 281757950 | 397500 |
| | 0100 | 0000 | URGENT | CRE_IO_SQ | 000D | 294950450 | 295345450 | 395000 |
| | 0100 | 0000 | URGENT | CRE_IO_SQ | 000E | 308182950 | 308580450 | 397500 |
| | 0100 | 0000 | URGENT | CRE_IO_SQ | 000F | 321212950 | 321610450 | 397500 |
| | 0100 | 0000 | URGENT | CRE_IO_SQ | 0000 | 334315450 | 334710450 | 395000 |
| | 0100 | 0000 | URGENT | CRE_IO_SQ | 0001 | 347895450 | 348290450 | 395000 |
| | 0100 | 0000 | URGENT | CRE_IO_SQ | 0002 | 360570450 | 360965450 | 395000 |
| | 0100 | 0000 | URGENT | CRE_IO_SQ | 0003 | 373660450 | 374057950 | 397500 |
| | 0100 | 0000 | URGENT | CRE_IO_SQ | 0004 | 386690450 | 387087950 | 397500 |
| | 0100 | 0001 | URGENT | WR | 0000 | 400350450 | 400747950 | 397500 |
| | 0100 | 0001 | URGENT | RD | 0001 | 403102950 | 403500450 | 397500 |
| | 0100 | 0001 | URGENT | WR | 0002 | 405460450 | 405977950 | 517500 |
| | 0100 | 0001 | URGENT | RD | 0003 | 408335450 | 408730450 | 395000 |

# QVIP @ Mentor

- Complete Verification Solution:


# Thanks

Visit us at Booth #136