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
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

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
<th>Type</th>
<th>Weight</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introduction</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>System Bus (PC/1 Express) Registers</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Controller Registers</strong></td>
<td>This section describes the Controller registers that are</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Offset 0xh: CAP - Controller Capabilities</strong></td>
<td>This register indicates basic capabilities of the controller to host</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This field indicates the maximum host memory page size that</td>
<td><strong>c3f_rgl_cig * cap _ capasersmax</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Corepoint</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This field indicates the minimum host memory page size that</td>
<td><strong>c3f_rgl_cig * cap _ capasersmen</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Corepoint</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This field indicates the IO Command Set(s) that the controller</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>c3f_rgl_cig * cap _ capases</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Corepoint</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This field indicates whether the controller supports the NVMM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>c3f_rgl_cig * cap _ nvmms</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Corepoint</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each Submission Queue and Completion Queue DASHBOARD</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>c3f_rgl_cig * cap _ dashbd</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Corepoint</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This is the worst case time that host software shall wait for</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>c3f_rgl_cig * cap _ arriaval</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Corepoint</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This field is bit significant and indicates the optional arbitration</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>c3f_rgl_cig * cap _ arriavals</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Corepoint</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This field is set in ‘1’ if the controller requires that DO</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>c3f_rgl_cig * cap _ arriavals</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Corepoint</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Offset 0xh: CAP - MGE2 (RO Bits 15-0)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Offset 0xh: VM - Version</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Offset 0xh: INFMS - Interrupt Mask Set</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Offset 0xh: BTMC - Interrupt Mask Clear</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Offset 14h: CC - Controller Configuration</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Offset 14h: CC - IO Completion Queue Entry Size (RW bits 19-15)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Offset 14h: CC - IO Submission Queue Entry Size (RW bits 15-11)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Offset 14h: CC - Direct Start Notification (RW bits 15-14)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Offset 14h: CC - Allocation Mechanism Selected (RW bits 15-14)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Offset 14h: CC - Memory Page size (RW bits 10-7)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Offset 14h: CC - IO Command Set Selected (RW bits 6-4)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Offset 14h: CC - Enable Set (RW bits 6-0)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Offset 14h: CC - Enable Clear (RW bits 5-0)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Santa Clara, CA**  
**August 2019**  

---

15
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.

<table>
<thead>
<tr>
<th>DEBUG ID</th>
<th>SDF</th>
<th>S</th>
<th>R</th>
<th>W</th>
<th>TYPE</th>
<th>SOID</th>
<th>COID</th>
<th>REG_NAME</th>
<th>CMD</th>
<th>MISC</th>
<th>CID</th>
<th>PS</th>
<th>PRP2</th>
<th>PRP1</th>
<th>NSID</th>
<th>STS/SLBA/PC</th>
<th>ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0109</td>
<td>H</td>
<td>R</td>
<td>REG</td>
<td></td>
<td></td>
<td>0109</td>
<td></td>
<td>CAP</td>
<td>0102</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0109</td>
<td>H</td>
<td>R</td>
<td>REG</td>
<td></td>
<td></td>
<td>0109</td>
<td></td>
<td>CSTS</td>
<td>00100030</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0109</td>
<td>H</td>
<td>R</td>
<td>REG</td>
<td></td>
<td></td>
<td>0109</td>
<td></td>
<td>CMBSZ</td>
<td>000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0109</td>
<td>H</td>
<td>R</td>
<td>REG</td>
<td></td>
<td></td>
<td>0109</td>
<td></td>
<td>AQG</td>
<td>0000000006F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0109</td>
<td>H</td>
<td>W</td>
<td>REG</td>
<td></td>
<td></td>
<td>0109</td>
<td></td>
<td>A50</td>
<td>000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0109</td>
<td>H</td>
<td>W</td>
<td>REG</td>
<td></td>
<td></td>
<td>0109</td>
<td></td>
<td>A5Q</td>
<td>000000810</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0109</td>
<td>H</td>
<td>W</td>
<td>REG</td>
<td></td>
<td></td>
<td>0109</td>
<td></td>
<td>ASQ</td>
<td>000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0109</td>
<td>H</td>
<td>W</td>
<td>REG</td>
<td></td>
<td></td>
<td>0109</td>
<td></td>
<td>ACQ</td>
<td>000000060</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0109</td>
<td>H</td>
<td>W</td>
<td>REG</td>
<td></td>
<td></td>
<td>0109</td>
<td></td>
<td>A00</td>
<td>0000000006F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0109</td>
<td>H</td>
<td>R</td>
<td>REG</td>
<td></td>
<td></td>
<td>0109</td>
<td></td>
<td>CC</td>
<td>000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0109</td>
<td>H</td>
<td>R</td>
<td>REG</td>
<td></td>
<td></td>
<td>0109</td>
<td></td>
<td>CC</td>
<td>00000000001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0109</td>
<td>H</td>
<td>R</td>
<td>REG</td>
<td></td>
<td></td>
<td>0109</td>
<td></td>
<td>CSS</td>
<td>000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0109</td>
<td>H</td>
<td>R</td>
<td>REG</td>
<td></td>
<td></td>
<td>0109</td>
<td></td>
<td>CSS</td>
<td>00000000001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0109</td>
<td>H</td>
<td>W</td>
<td>REG</td>
<td></td>
<td></td>
<td>0109</td>
<td></td>
<td>SQDTDBL</td>
<td>000000000001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The table above is a sample of debug data from an event. The specific data shown is for illustrative purposes and may not be representative of real-world debug scenarios.
Performance loggers

- Performance logging
• Complete Verification Solution:
Thanks

Visit us at Booth #136