Validation and Testing Challenges of NVMe Subsystems

Eric Lanning
Hardware Engineering Manager, SerialTek
SAS/SATA Analysis

- SAS/SATA is a well-understood protocol
- Analyzers can interpret SAS/SATA protocols with relative ease
  - Each frame has a type (i.e. Command, Response, Data …)
  - It is straightforward grouping frames into “transactions”

**Command frames start a transaction**

**Response frames end the transaction**

**Data frames are transmitted in between**
NVMe Advantages

- Scalable performance with low latency
  - No need to convert native PCIe to SAS or SATA
- Efficient and streamlined command set
- Support for up to 64K I/O queues, with each I/O queue supporting 64K commands
- Priority associated with each I/O queue with a well-defined arbitration mechanism
- All information to complete a 4KB read request is included in the 64 byte command itself
- Support for multiple namespaces
- These features give us lower latency and higher performance, but makes it more difficult to analyze
NVMe Validation/Testing Challenges

- Validating and analyzing NVMe designs is challenging
  - Controller registers are addresses that are an offset of the device’s BAR0/BAR1
  - Commands and completions are stored in queues which occupy a range of addresses
  - BIOS and driver can create and destroy queues as needed
- In order to decode, filter and trigger, analyzers need to know the BAR address, queue addresses and sizes
  - This information can be entered manually, but this method can be error prone and difficult if queues are created and destroyed multiple times in the same trace
  - If the initialization was captured, it can be extrapolated from the trace
  - The analyzer can monitor the traffic and archive all necessary information in hardware
- The physical memory locations for data transfers are specified through pointers
Queue Archive Example

<table>
<thead>
<tr>
<th>Queue Type</th>
<th>Queue ID</th>
<th>Queue Size (# of Entries)</th>
<th>Base Address</th>
<th>Valid From</th>
<th>Valid To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin Completion</td>
<td>0x0</td>
<td>0x1</td>
<td>b07f7000</td>
<td>79.364.840.412.000</td>
<td>96.313.631.220.000</td>
</tr>
<tr>
<td>Admin Submission</td>
<td>0x0</td>
<td>0x1</td>
<td>b07f6000</td>
<td>79.364.840.412.000</td>
<td>96.313.631.220.000</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0x1</td>
<td>0x1</td>
<td>b07f0000</td>
<td>79.365.063.060.500</td>
<td>96.313.631.220.000</td>
</tr>
<tr>
<td>I/O Submission</td>
<td>0x0</td>
<td>0xff</td>
<td>43e711000</td>
<td>96.313.631.220.000</td>
<td>End of Recording</td>
</tr>
<tr>
<td>Admin Completion</td>
<td>0x0</td>
<td>0xff</td>
<td>43e715000</td>
<td>96.313.631.220.000</td>
<td>End of Recording</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0x1</td>
<td>0x3ff</td>
<td>43e740000</td>
<td>99.769.563.994.000</td>
<td>100.689.940.782.000</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0x2</td>
<td>0x3ff</td>
<td>43e771000</td>
<td>99.785.163.182.500</td>
<td>100.675.717.325.000</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0x3</td>
<td>0x3ff</td>
<td>43e465000</td>
<td>99.800.762.945.000</td>
<td>100.658.741.470.500</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0x4</td>
<td>0x3ff</td>
<td>43e4c9000</td>
<td>99.817.738.441.000</td>
<td>100.643.141.820.000</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0x5</td>
<td>0x3ff</td>
<td>43e545000</td>
<td>99.831.961.891.500</td>
<td>100.627.542.277.000</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0x6</td>
<td>0x3ff</td>
<td>43c268000</td>
<td>99.847.561.553.500</td>
<td>100.611.943.147.000</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0x7</td>
<td>0x3ff</td>
<td>43c2c0000</td>
<td>99.863.161.337.000</td>
<td>100.597.719.039.000</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0x8</td>
<td>0x3ff</td>
<td>43c330000</td>
<td>99.878.760.765.500</td>
<td>100.580.743.400.000</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0x9</td>
<td>0x3ff</td>
<td>43c394000</td>
<td>99.893.736.614.000</td>
<td>100.565.143.778.500</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0xa</td>
<td>0x3ff</td>
<td>43c0a0000</td>
<td>99.909.959.979.000</td>
<td>100.549.544.227.000</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0xb</td>
<td>0x3ff</td>
<td>43c074000</td>
<td>99.925.559.620.500</td>
<td>100.533.945.529.000</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0xc</td>
<td>0x3ff</td>
<td>43c0d8000</td>
<td>99.941.159.233.000</td>
<td>100.519.721.169.000</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0xd</td>
<td>0x3ff</td>
<td>43e73e000</td>
<td>99.956.758.878.500</td>
<td>100.502.745.415.000</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0xe</td>
<td>0x3ff</td>
<td>43e761000</td>
<td>99.973.734.485.500</td>
<td>100.487.145.720.000</td>
</tr>
<tr>
<td>I/O Completion</td>
<td>0xf</td>
<td>0x3ff</td>
<td>43e455000</td>
<td>99.987.958.048.000</td>
<td>100.471.546.130.500</td>
</tr>
</tbody>
</table>
NVMe Command Processing / Analyzer
Decoding of Command Processing

Source: NVMe v1.2 specification
Filtering/Triggering Challenges

- Decoding NVMe traffic does not require the BAR0/1 address and queue address ranges real-time.
- Decoding can take the information after the capture is complete and decode all of the Doorbell, Admin and I/O Commands and Completions.
- In order to trigger and filter on commands, the BAR0/1 address and the queue address ranges need to be stored in hardware so the analyzer can act on the traffic in real-time.
- Triggering on data transfers is difficult because the pointers specifying the physical memory address are dynamic and are probably not known ahead of time.
Filtering Example

- Filtering is also an important analyzer capability used to prevent the capturing of information that is unimportant to the user.
- At 8 GT/s and multiple lanes wide, the analyzer will fill its buffer in a second or two.
- By using the BAR0/1 addresses and the queue address ranges, the analyzer can compare the address of Memory Write and know whether the data being fetched is command or completion.
- Once it is known that the data is a command or completion, the analyzer can filter those packets depending on the command opcode.
Triggering is a standard and important analyzer capability used to stop the capturing of a trace because of a certain event or condition (e.g. error).

To trigger on NVMe events, analyzers have to first determine whether a TLP contains NVMe events/information.

If the address of the TLP packet is not compared against the queue address ranges, then knowing whether the data being transferred is part of an NVMe command or some other data transfer becomes a “guess”.

- There is no way to distinguish between Admin and IO commands.
- The size and type of the TLP are used to try and trigger on the command or completion.
- This can lead to false positives.
Summary and Key Take-Aways

- Without accurate and current admin/IO queue address ranges and BAR0/1 addresses, PCIe/NVMe protocol analysis is extremely challenging
  - Key analyzer filtering and triggering functions work less reliably
  - This impacts development and testing cycle and often results in product release delays
  - Manual input of queue information into analyzer software can alleviate some issues but process is tedious and error-prone
- Automated discovery and capture of this information is what is required to overcome these challenges unique to NVMe protocol analysis