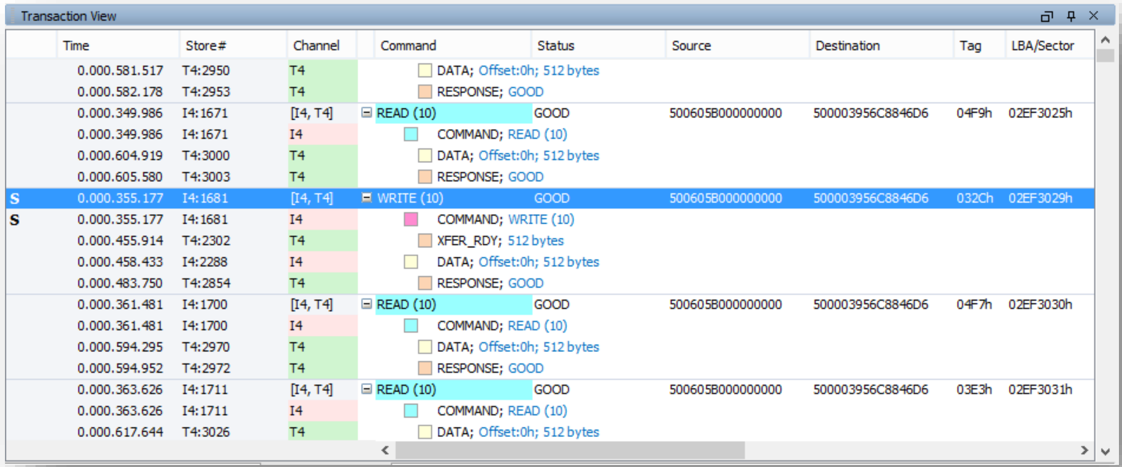
Validation and Testing Challenges of NVMe Subsystems

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 straight forward grouping frames into “transactions”



Transaction View

	Time	Store#	Channel	Command	Status	Source	Destination	Tag	LBA/Sector
	0.000.581.517	T4:2950	T4	DATA; Offset:0h; 512 bytes					
	0.000.582.178	T4:2953	T4	RESPONSE; GOOD					
	0.000.349.986	I4:1671	[I4, T4]	⊟ READ (10)	GOOD	500605B000000000	500003956C8846D6	04F9h	02EF3025h
	0.000.349.986	I4:1671	I4	COMMAND; READ (10)					
	0.000.604.919	T4:3000	T4	DATA; Offset:0h; 512 bytes					
	0.000.605.580	T4:3003	T4	RESPONSE; GOOD					
S	0.000.355.177	I4:1681	[I4, T4]	⊟ WRITE (10)	GOOD	500605B000000000	500003956C8846D6	032Ch	02EF3029h
S	0.000.355.177	I4:1681	I4	COMMAND; WRITE (10)					
	0.000.455.914	T4:2302	T4	XFER_RDY; 512 bytes					
	0.000.458.433	I4:2288	I4	DATA; Offset:0h; 512 bytes					
	0.000.483.750	T4:2854	T4	RESPONSE; GOOD					
	0.000.361.481	I4:1700	[I4, T4]	⊟ READ (10)	GOOD	500605B000000000	500003956C8846D6	04F7h	02EF3030h
	0.000.361.481	I4:1700	I4	COMMAND; READ (10)					
	0.000.594.295	T4:2970	T4	DATA; Offset:0h; 512 bytes					
	0.000.594.952	T4:2972	T4	RESPONSE; GOOD					
	0.000.363.626	I4:1711	[I4, T4]	⊟ READ (10)	GOOD	500605B000000000	500003956C8846D6	03E3h	02EF3031h
	0.000.363.626	I4:1711	I4	COMMAND; READ (10)					
	0.000.617.644	T4:3026	T4	DATA; Offset:0h; 512 bytes					

- 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

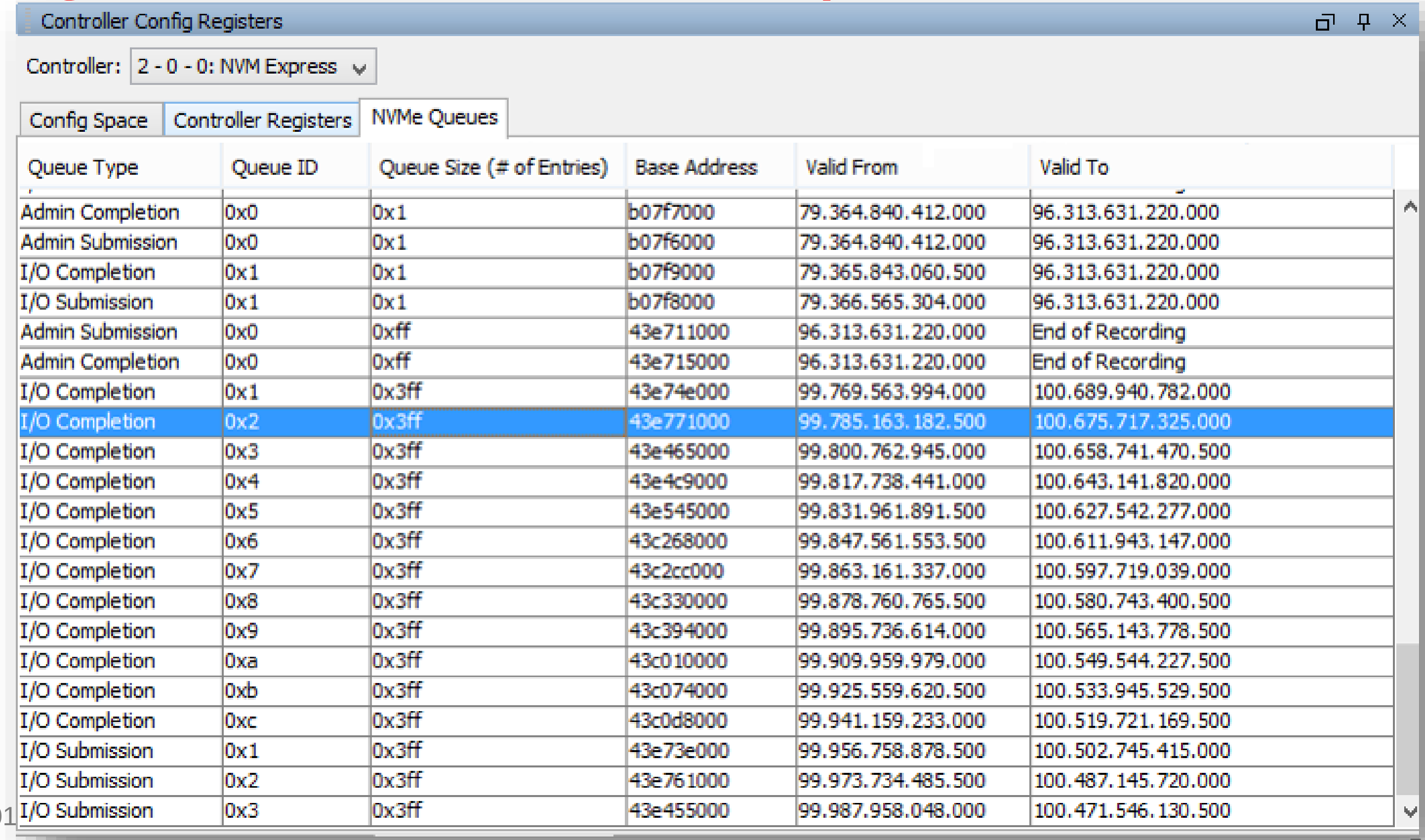
NVMe Validation/Testing Challenges

- 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

Created by the BIOS at boot up

Created by the driver



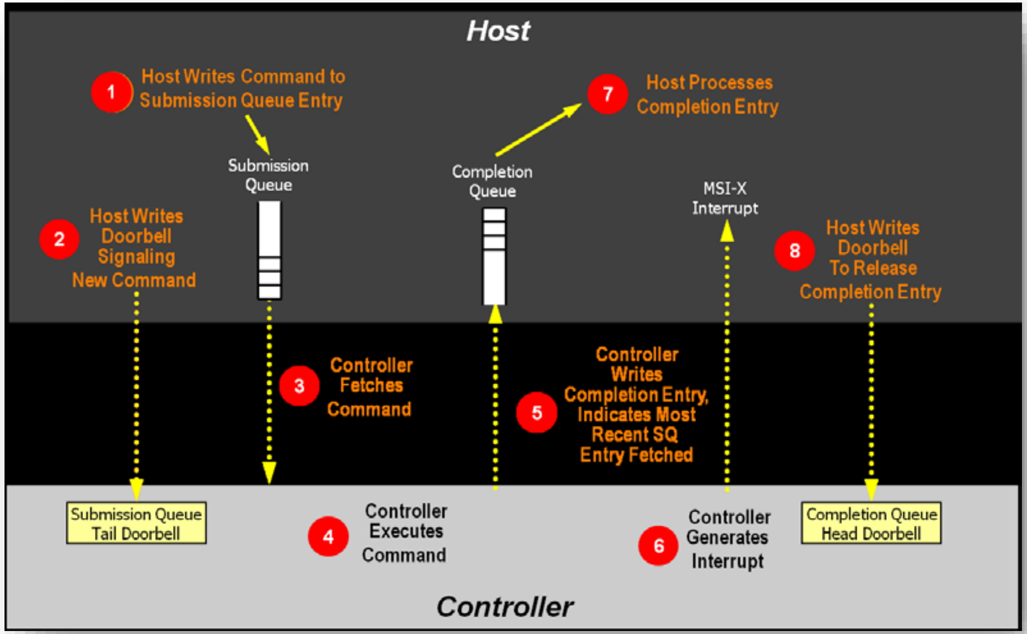
Controller Config Registers

Controller: 2 - 0 - 0: NVM Express

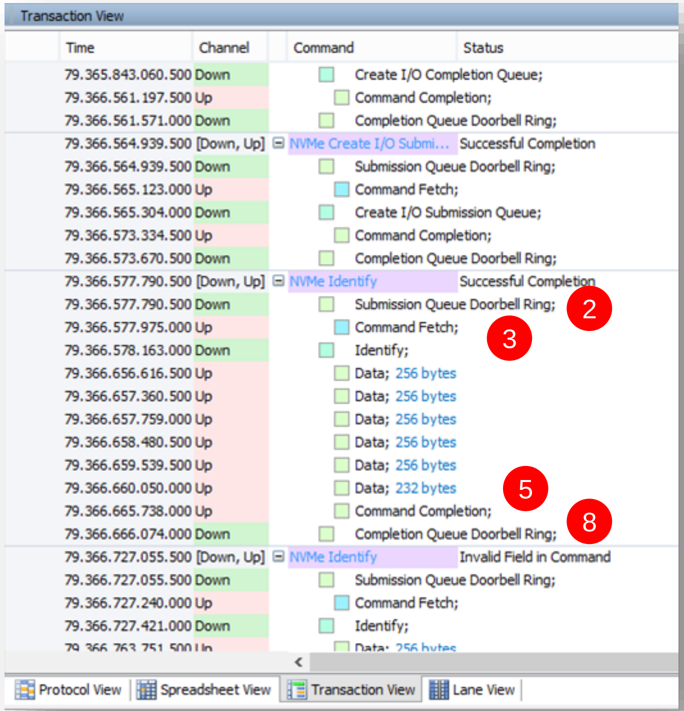
Config Space | Controller Registers | NVMe Queues

Queue Type	Queue ID	Queue Size (# of Entries)	Base Address	Valid From	Valid To
Admin Completion	0x0	0x1	b07f7000	79.364.840.412.000	96.313.631.220.000
Admin Submission	0x0	0x1	b07f6000	79.364.840.412.000	96.313.631.220.000
I/O Completion	0x1	0x1	b07f9000	79.365.843.060.500	96.313.631.220.000
I/O Submission	0x1	0x1	b07f8000	79.366.565.304.000	96.313.631.220.000
Admin Submission	0x0	0xff	43e711000	96.313.631.220.000	End of Recording
Admin Completion	0x0	0xff	43e715000	96.313.631.220.000	End of Recording
I/O Completion	0x1	0x3ff	43e74e000	99.769.563.994.000	100.689.940.782.000
I/O Completion	0x2	0x3ff	43e771000	99.785.163.182.500	100.675.717.325.000
I/O Completion	0x3	0x3ff	43e465000	99.800.762.945.000	100.658.741.470.500
I/O Completion	0x4	0x3ff	43e4c9000	99.817.738.441.000	100.643.141.820.000
I/O Completion	0x5	0x3ff	43e545000	99.831.961.891.500	100.627.542.277.000
I/O Completion	0x6	0x3ff	43c268000	99.847.561.553.500	100.611.943.147.000
I/O Completion	0x7	0x3ff	43c2cc000	99.863.161.337.000	100.597.719.039.000
I/O Completion	0x8	0x3ff	43c330000	99.878.760.765.500	100.580.743.400.500
I/O Completion	0x9	0x3ff	43c394000	99.895.736.614.000	100.565.143.778.500
I/O Completion	0xa	0x3ff	43c010000	99.909.959.979.000	100.549.544.227.500
I/O Completion	0xb	0x3ff	43c074000	99.925.559.620.500	100.533.945.529.500
I/O Completion	0xc	0x3ff	43c0d8000	99.941.159.233.000	100.519.721.169.500
I/O Submission	0x1	0x3ff	43e73e000	99.956.758.878.500	100.502.745.415.000
I/O Submission	0x2	0x3ff	43e761000	99.973.734.485.500	100.487.145.720.000
I/O Submission	0x3	0x3ff	43e455000	99.987.958.048.000	100.471.546.130.500

NVMe Command Processing / Analyzer Decoding of Command Processing


Host
1 Host Writes Command to Submission Queue Entry
7 Host Processes Completion Entry
Submission Queue
Completion Queue
MSI-X Interrupt
2 Host Writes Doorbell Signaling New Command
8 Host Writes Doorbell To Release Completion Entry
3 Controller Fetches Command
5 Controller Writes Completion Entry, Indicates Most Recent SQ Entry Fetched
Submission Queue Tail Doorbell
4 Controller Executes Command
6 Controller Generates Interrupt
Completion Queue Head Doorbell
Controller

Source: NVMe v1.2 specification



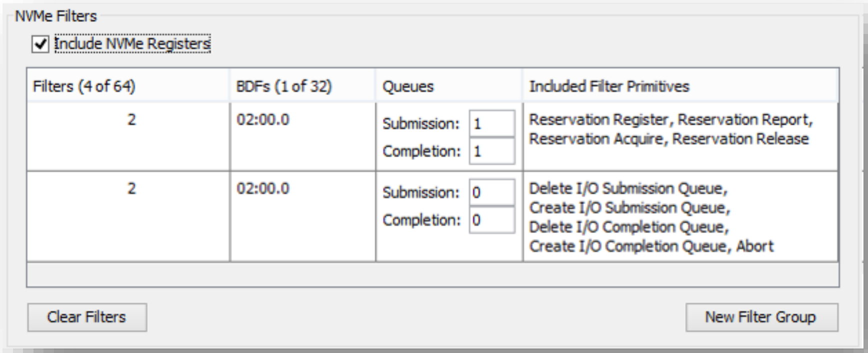
Time	Channel	Command	Status
79.365.843.060.500	Down	Create I/O Completion Queue;	
79.366.561.197.500	Up	Command Completion;	
79.366.561.571.000	Down	Completion Queue Doorbell Ring;	
79.366.564.939.500	[Down, Up]	NVMe Create I/O Submi...	Successful Completion
79.366.564.939.500	Down	Submission Queue Doorbell Ring;	
79.366.565.123.000	Up	Command Fetch;	
79.366.565.304.000	Down	Create I/O Submission Queue;	
79.366.573.334.500	Up	Command Completion;	
79.366.573.670.500	Down	Completion Queue Doorbell Ring;	
79.366.577.790.500	[Down, Up]	NVMe Identify	Successful Completion
79.366.577.790.500	Down	Submission Queue Doorbell Ring;	
79.366.577.975.000	Up	Command Fetch;	
79.366.578.163.000	Down	Identify;	
79.366.656.616.500	Up	Data; 256 bytes	
79.366.657.360.500	Up	Data; 256 bytes	
79.366.657.759.000	Up	Data; 256 bytes	
79.366.658.480.500	Up	Data; 256 bytes	
79.366.659.539.500	Up	Data; 256 bytes	
79.366.660.050.000	Up	Data; 232 bytes	
79.366.665.738.000	Up	Command Completion;	
79.366.666.074.000	Down	Completion Queue Doorbell Ring;	
79.366.727.055.500	[Down, Up]	NVMe Identify	Invalid Field in Command
79.366.727.055.500	Down	Submission Queue Doorbell Ring;	
79.366.727.240.000	Up	Command Fetch;	
79.366.727.421.000	Down	Identify;	
79.366.763.751.500	Up	Data; 256 bytes	

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



NVMe Filters

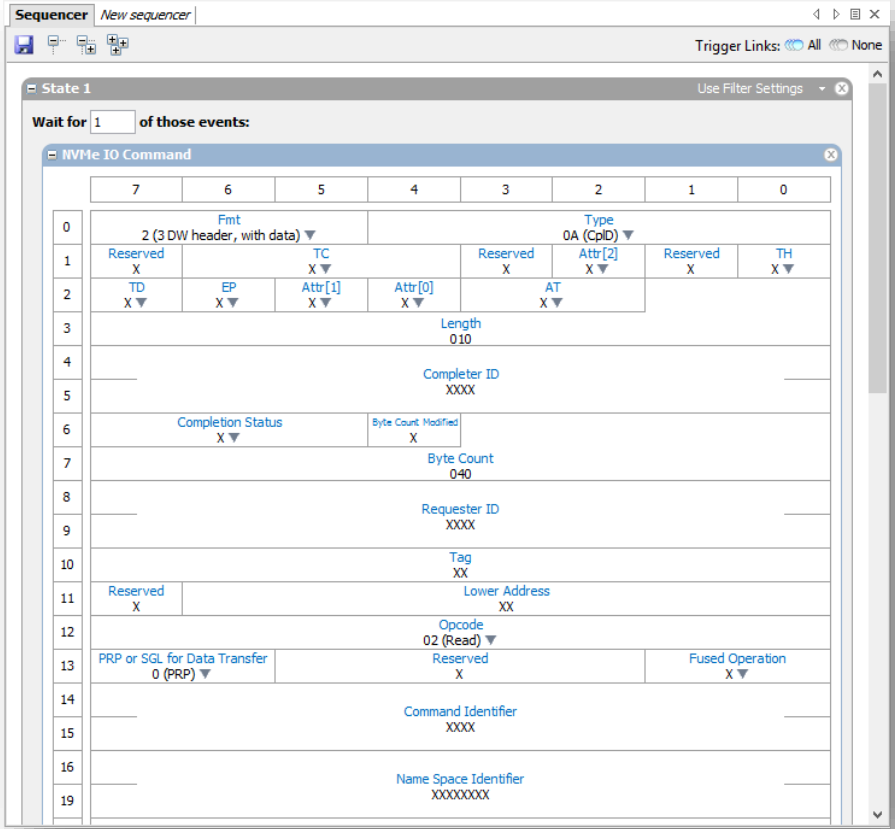
☑ Include NVMe Registers

Filters (4 of 64)	BDFs (1 of 32)	Queues	Included Filter Primitives
2	02:00.0	Submission: 1 Completion: 1	Reservation Register, Reservation Report, Reservation Acquire, Reservation Release
2	02:00.0	Submission: 0 Completion: 0	Delete I/O Submission Queue, Create I/O Submission Queue, Delete I/O Completion Queue, Create I/O Completion Queue, Abort

- 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

- 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



Sequencer | *New sequencer*

Trigger Links: All None

State 1 — Use Filter Settings

Wait for 1 of those events:

NVMe IO Command

	7	6	5	4	3	2	1	0
0	Fmt 2 (3 DW header, with data)			Type 0A (CplD)				
1	Reserved X	TC X			Reserved X	Attr[2] X	Reserved X	TH X
2	TD X	EP X	Attr[1] X	Attr[0] X	AT X			
3	Length 010							
4	Completer ID XXXX							
5								
6	Completion Status X			Byte Count Modified X				
7	Byte Count 040							
8	Requester ID XXXX							
9								
10	Tag XX							
11	Reserved X	Lower Address XX						
12	Opcode 02 (Read)							
13	PRP or SGL for Data Transfer 0 (PRP)		Reserved X				Fused Operation X	
14	Command Identifier XXXX							
15								
16	Name Space Identifier XXXXXXXX							
19								

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