Extending the NVMHCI Standard to Enterprise

Amber Huffman
Principal Engineer
Intel Corporation

Outline

- *Remember: What is NVMHCI*
- PCIe SSDs Coming… with Challenges
- Enterprise Extensions to NVMHCI
- Summary

Remember NVMHCI: An Optimized Interface for NVM

- NVMHCI: Non-Volatile Memory Host Controller Interface
- NVMHCI is a clean and optimized interface for SSDs and caches
- NVM equivalent of the SATA AHCI controller interface

Quick Links | Home | Worldwide

Microsoft

Search Microsoft.com for: Go

PressPass - Information for Journalists

PressPass Home | PR Contacts | Fast Facts About Microsoft | Site Map | Advanced Search | RSS Feeds

Microsoft News
Product News
Consumer News
International Contacts
Legal News
Security & Privacy News
Events
News Archive

Corporate Information
Microsoft Executives
Fast Facts About Microsoft
Image Gallery
Broadcast Room

Related Sites
Analyst Relations
Community Affairs
Essays on Technology
Executive E-Mail
Global Citizenship
Investor Relations
Microsoft Research

The PressPass Broadcast Room
Broadcast-standard media for download

PressPass Subscriptions RSS

Dell, Intel and Microsoft Join Forces to Increase Adoption of NAND-Based Flash Memory in PC Platforms

Newly formed group to provide standard interface for nonvolatile memory subsystems.

REDMOND, Wash. — May 30, 2007 — Broad adoption of NAND flash memory technology in the PC platform received a boost with the formation of the Non-Volatile Memory Host Controller Interface (NVMHCI) Working Group. The NVMHCI Working Group is chaired by Intel Corporation with core contributors including Dell Inc. and Microsoft Corp.

Related Links
External Resources:
- Dell Web site
- Intel Web site

NVMHCI will provide a standard software programming interface for nonvolatile memory subsystems. The interface would be used by operating system drivers to access NAND flash memory storage in applications such as hard drive caching and solid-state drives.

"Several NAND solutions are coming on the scene to take advantage of the ReadyBoost™ and ReadyDrive™ features of the Windows Vista® operating system," said Bob Rinne, general manager of Windows Hardware Ecosystem at Microsoft. "Standardizing on a common controller interface will enable more integrated operating system support of these solutions moving forward."

Industry momentum for standardization in NAND storage solutions is building, especially as NAND moves into the PC platform. NVMHCI complements standardization work being done in the Open NAND Flash Interface (ONFI) Working Group.

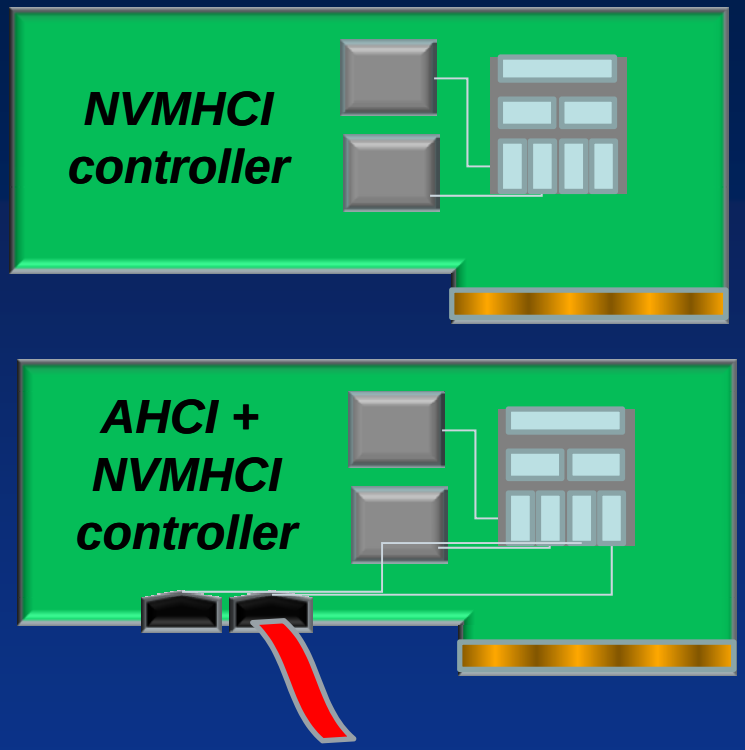
"We've got a performance-enhancing NAND-based product in the market with our new Centrino mobile technology platform called Intel Turbo memory, and this newly formed working group will help make that and a number of other NAND-based solutions more prolific, faster," said Rick Coulson, senior fellow and director of I/O Architecture at Intel. "ONFI formed last year to standardize the interface between the Flash controller and the NAND itself, and standardizing the register level interface between the Flash controller and the operating system driver is the logical next step."

"Nonvolatile memory solutions enable better system performance and lower power consumption as well as facilitate additional benefits such as smaller form factors, quieter systems and improved robustness," said Liam Quinn, director of communications for technology strategy and architecture at Dell. "Dell looks forward to working with industry partners and extending the benefits NVMHCI will bring to our customers."

The group is actively expanding its membership to include other industry-leading companies

Technical Essence of NVMHCI

- NVMHCI defines a standard programming interface for non-volatile memory subsystems
- Leverage AHCI to provide best infrastructure for caching
  - One driver for HDDs and NAND
- Allows NVMHCI registers to appear as:
  - A separate PCI device
  - A port within an existing AHCI controller
- NVMHCI is a logical interface
  - All NAND management abstracted out: NAND technology changes too quickly
  - All caching algorithms are outside the spec: NVMHCI only defines how caching software gets access to the NAND
- Optimized interface for both cache and SSD usage models



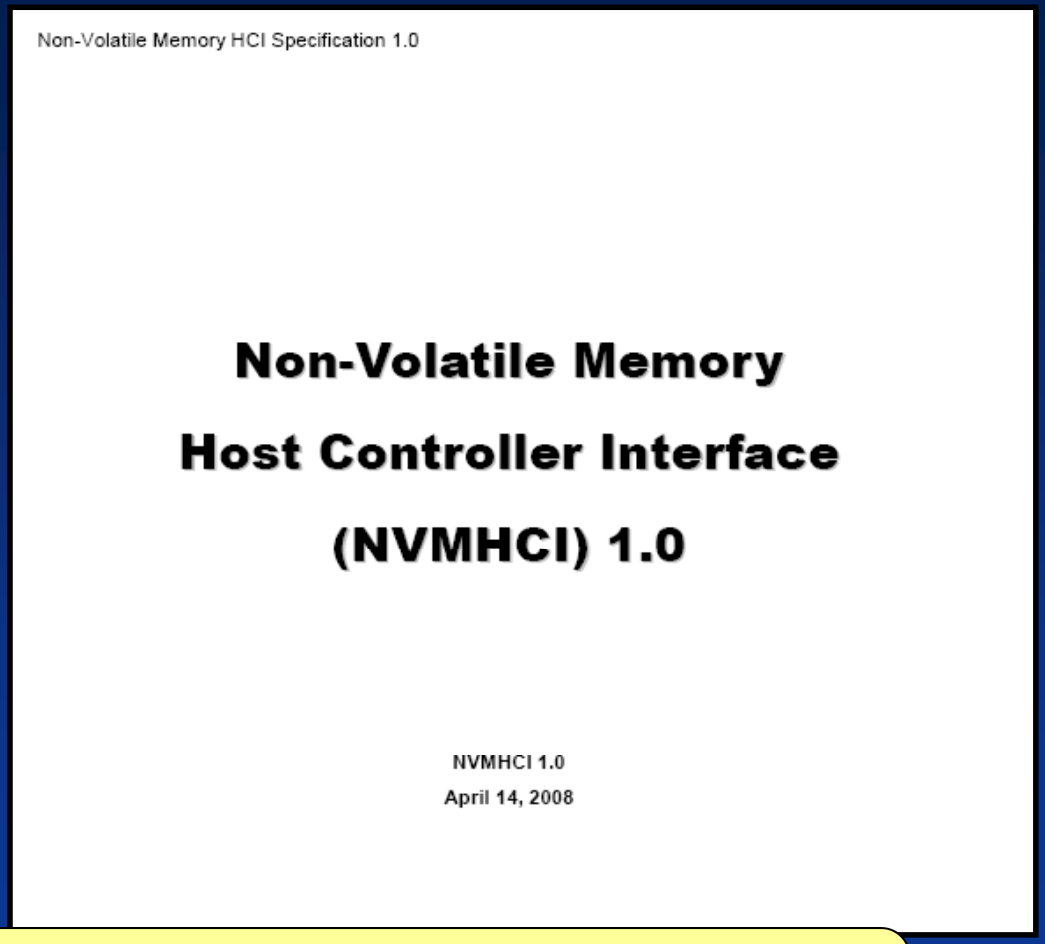
Companies Driving NVMHCI

The NVMHCI Workgroup includes 40+ members, focused on delivering streamlined NVM solutions.

*Other names and brands may be claimed as the property of others

NVMHCI 1.0 Spec Delivered

- NVMHCI 1.0 completed in April 2008
- Less than one year from team formation to ratification
- Includes registers, DMA engine, and command set



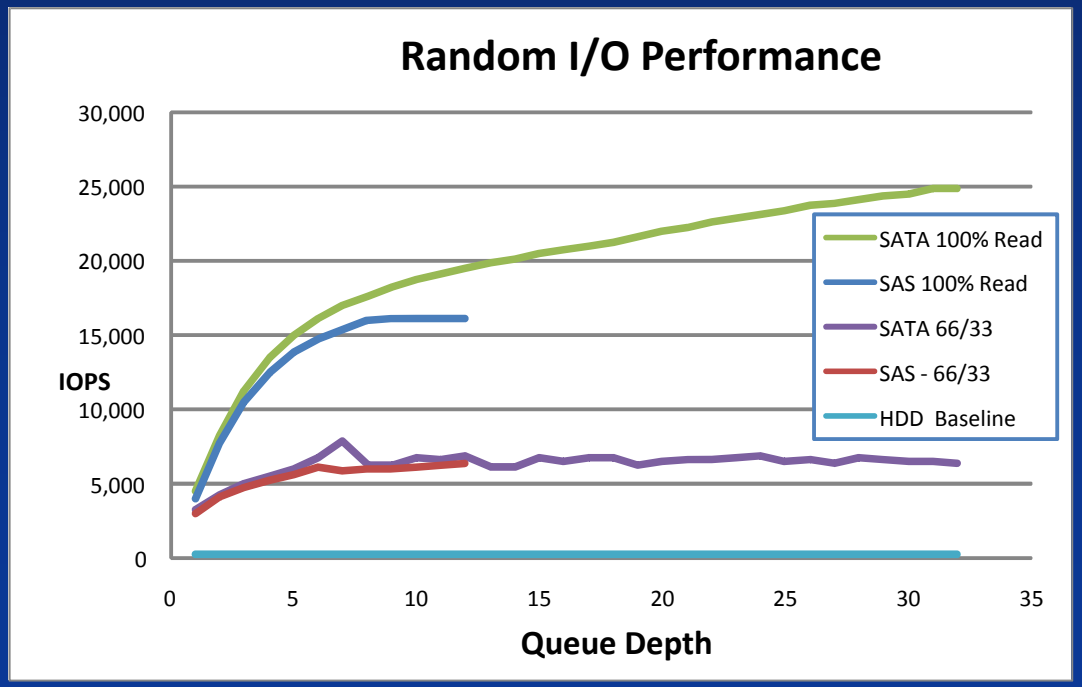
Available for download at:
http://www.intel.com/standards/nvmhci

Outline

- Remember: What is NVMHCI
- *PCIe SSDs Coming… with Challenges*
- Enterprise Extensions to NVMHCI
- Summary

PCIe SSD Value Proposition

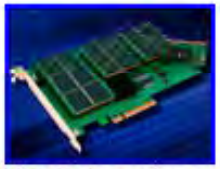
- PCIe SSDs are attractive for some segments
  - Eliminates SAS infrastructure cost and bottleneck *(being addressed)*
  - Platform connectivity is from PCIe, plenty of PCIe lanes
  - Bandwidth performance can be concentrated into fewer devices (6Gb/sec SATA/SAS vs multi-lane PCIe Gen2)
  - Lower latency (µsec matter)



SAS Bottleneck

- Via SATA chipset direct attach, one SSD delivers outstanding I/O performance
  - > 24,000 Read IOPs
  - > 6,000 OLTP IOPs
- However, attached to SAS the IOPs are limited by the SAS controller
  - *SAS controllers are being updated for an SSD world*

SMART Modular Technologies Announces Enterprise Class PCI Express Storage Solution



Click for larger image

Delivering 140K random performance and 200x power reduction, the new 400GB XceedIOPS PCIe raises the bar in next-generation solid-state storage.

NEWARK, CA, June 1, 2009 - SMART Modular Technologies (WWH), Inc. ("SMART" or the "Company") (NASDAQ: SMOD), a leading independent manufacturer ... subsystems, ... (NASDAQ: MR... solutions, to d... first in a line of p... applications.

The Register®
Biting the hand that feeds IT

Micron barges into PCIe SSD business

Partners up with IDT

By **Chris Mellor** • **Get more from this author**

Posted in Storage, 28th July 2009 12:51 GMT

Fusion-io unveils 80GB ioXtreme PCI Express SSD

TECHSPOT
PC TECHNOLOGY NEWS AND ANALYSIS

By Matthew DeCarlo, TechSpot.com
Published: June 8, 2009, 9:15 AM EST

Fusion-io is launching a new "Fatal1ty" branded product as they deliver an enthusiast-oriented PCI Express solid state drive. The ioXtreme SSD will make use of the PCI-E x4 interface and bear a non-volatile 80GB capacity based on MLC NAND technology.

RamSan-20 highlights:

- 450GB Flash storage
- 120,000 IOPS
- 700 MB/s random sustained external throughput
- ECC and RAID
- Embedded CPU controller.

To order, please contact Texas Memory Systems Sales

TexasMemorySystems

Super Talent Debuts PCI Express SSD RAIDDrive

Wednesday, April 01, 2009 - by Shawn Oliver

Please Super Talent, let this not be a joke. Here on the alw... April, Super Talent has announced its reaction to Fusion-io'... the patented RAID... lent's o... bring quick, durabl... mothe... interface.

HOT HARDWARE
THE HOTTEST TECH, TESTED AND BURNED IN

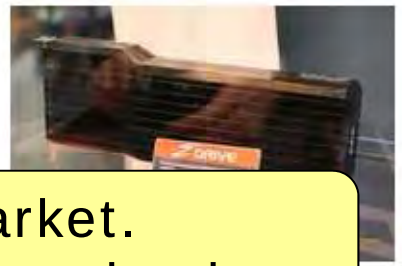
OCZ's New Blazing Fast 1TB Z SSD Drive

tom's hardware

7:00 PM - March 4, 2009 by Steve Seguin

OCZ was demonstrating its new Z Drive with 1 TB of storage at CeBIT 2009 this week.

The new Z Drive from OCZ is a storage device that connects to an x8 PCIe slot and offers 1 terabyte of storage capacity. The Z Drive is about the same size a dual-slot graphics card, so its not exactly small, but the device is stated to offer maximum read and write speeds of up to 600 MB/sec. and 500 MB/sec., respectively.



Enterprise class PCIe SSDs are coming to market.
However, they have proprietary interfaces impacting adoption.

PCIe SSDs are Coming, With Challenges

- Question: What OS and driver infrastructure is in place for PCIe SSDs?
- Answer: None. PCIe SSDs do not have a standard host controller interface. Solutions are delivered with proprietary interfaces & drivers.
  - Note: NVMHCI is available for use in client PCIe SSDs.
- Lack of a standard host controller interface impacts PCIe SSD adoption
  - Requires each SSD vendor to provide a driver for each OS, rather than focusing on delivering a great SSD solution
  - OEM features, like error logs, are implemented in an inconsistent fashion
  - Rather than qualifying one driver with multiple SSDs, requires OEM to validate each SSD with it's own driver, increasing validation time and cost
- What if we made the PCIe SSD look like a SATA or SAS SSD?
  - SATA has a standard host controller interface, AHCI. However, AHCI is not optimized for NVM and emulating a SATA device adds cost and complexity.
  - SAS does not have a standard host controller interface. Only proprietary HCIs with their associated RAID driver stacks are available.

Enterprise class PCIe SSDs would greatly benefit from a standard host controller interface.

Extend NVMHCI for Enterprise class PCIe SSDs

- Extend NVMHCI to meet the needs of Enterprise PCIe SSDs
  - Address Enterprise server scenarios
  - Enables SSD vendors to focus on building a great SSD
  - Enables OS vendors to deliver a great driver for all PCIe SSDs
  - Enables OEMs to qualify a single driver on each OS, with features implemented in a consistent fashion, reducing time to market
- Leverage NVMHCI interface, software infrastructure, and Workgroup to fill this gap quickly with a streamlined solution
  - Make NVMHCI an ideal interface for Enterprise PCIe SSDs
  - Take advantage of drivers already written or underway
  - Take advantage of existing Workgroup, an efficient team that can execute quickly

One Member's View of Benefit

"A standardized interface functions as a foundation, enabling a volume market for technology innovation while avoiding the compatibility issues that arise from multiple, proprietary interfaces. Enterprise customers are requesting standard interfaces be used on non-volatile-memory products as an enabler to assist broad adoption."

Microsoft

John Loveall, Microsoft
Director of Program Management

Enterprise Extensions: Architectural Goals

- Question: What could be delivered if the NVM device had more resources (e.g., buffering)
- Answer: A highly parallel and low latency solution, leading to new levels of performance.

- Architectural Goals for Enterprise extensions
  - Increase parallelism & number of units of work per port
  - Streamline protocol (e.g., eliminate unnecessary DRAM accesses in command issue, collapse status completion, etc)
  - Extensible manageability architecture
  - Add Enterprise class features, e.g.,:
    - End-to-end data integrity
    - Error injection, fast to fail indication, enhanced status codes, SMART
    - Interrupt coalescing
    - MSI-X
    - Secure Erase at a device level

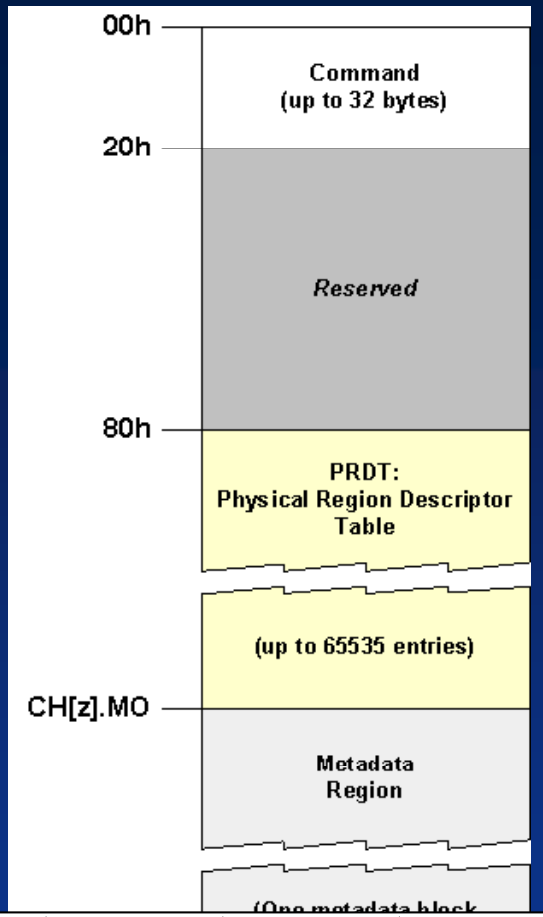
Outline

- Remember: What is NVMHCI
- PCIe SSDs Coming… with Challenges
- *Enterprise Extensions to NVMHCI*
  - *Approaches to:*
    - *Increase parallelism & number of units of work per port*
    - *Streamline protocol (e.g., eliminate unnecessary DRAM accesses in command issue, collapse status completion, etc)*
- Summary

NVMHCI 1.0 Command Issue Recap

- To issue a data command in NVMHCI 1.0 requires hardware to:
  - Fetch command header at PxCLB[CH(z)]
  - Fetch command from CTBA[0]
  - Fetch PRD from CTBA[80h]
  - More fetches if Metadata Region or PRD Index Table is present
- This is a good approach when hardware memory resources are limited
  - E.g., fetch PRD entries as needed
- For Enterprise class SSDs with more resources, important to minimize the number of fetches to increase IOPs

Command Table

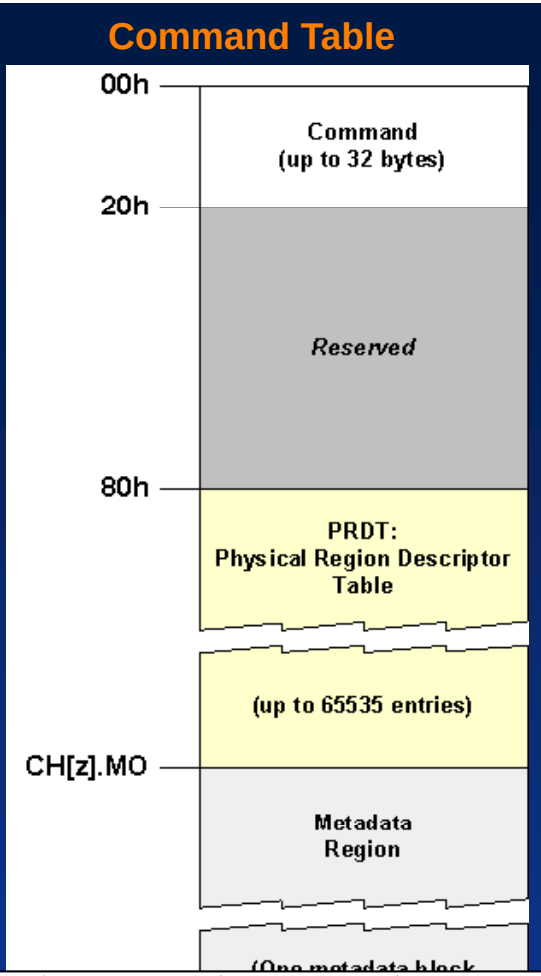


	31	23	15	7	0
DW0	PRDTL		*Reserved*	DT W I R	CL
DW1	PRD Index Table Offset				
DW2	CTBA0: Command Table Base Address				*Reserved*
DW3	CTBAU0: Command Table Base Adr Upper 32-bits				
DW4	Metadata Offset				
DW5	*Reserved*				
DW6	Command Status				
DW7	*Reserved*				

Command Header

Streamlining Approach: Packed Commands

- If the device has sufficient memory resources, it is more efficient to transmit all command details in one DMA transfer
- Approach: Develop “packed commands” that include the key details of the command header and command table



	31	23	15	7	0
DW0	PRDTL		Reserved	DT W I R	CL
DW1	PRD Index Table Offset				
DW2	CTBA0: Command Table Base Address				Reserved
DW3	CTBAU0: Command Table Base Adr Upper 32-bits				
DW4	Metadata Offset				
DW5	Reserved				
DW6	Command Status				
DW7	Reserved				

Command Header

Streamlining the Command Header

- Eliminate need for the Metadata Offset and PRD Offset by requiring a fixed region ordering and packing regions together
  - The device calculates the number of NVM pages touched for a command, and thus knows the size of the Metadata and PRD Index Table Regions
- Require the streamlined command header and command table be joined together, eliminating the Command Table Base Address

	31	23	15	7					0
DW0	PRDTL		*Reserved*	DT	W	I	R	CL	
DW1	PRD Index Table Offset								
DW2	CTBA0: Command Table Base Address							*Reserved*	
DW3	CTBAU0: Command Table Base Adr Upper 32-bits								
DW4	Metadata Offset								
DW5	*Reserved*								
DW6	Command Status								
DW7	*Reserved*								

← Add bit to indicate PRD Index Table present

← Eliminate

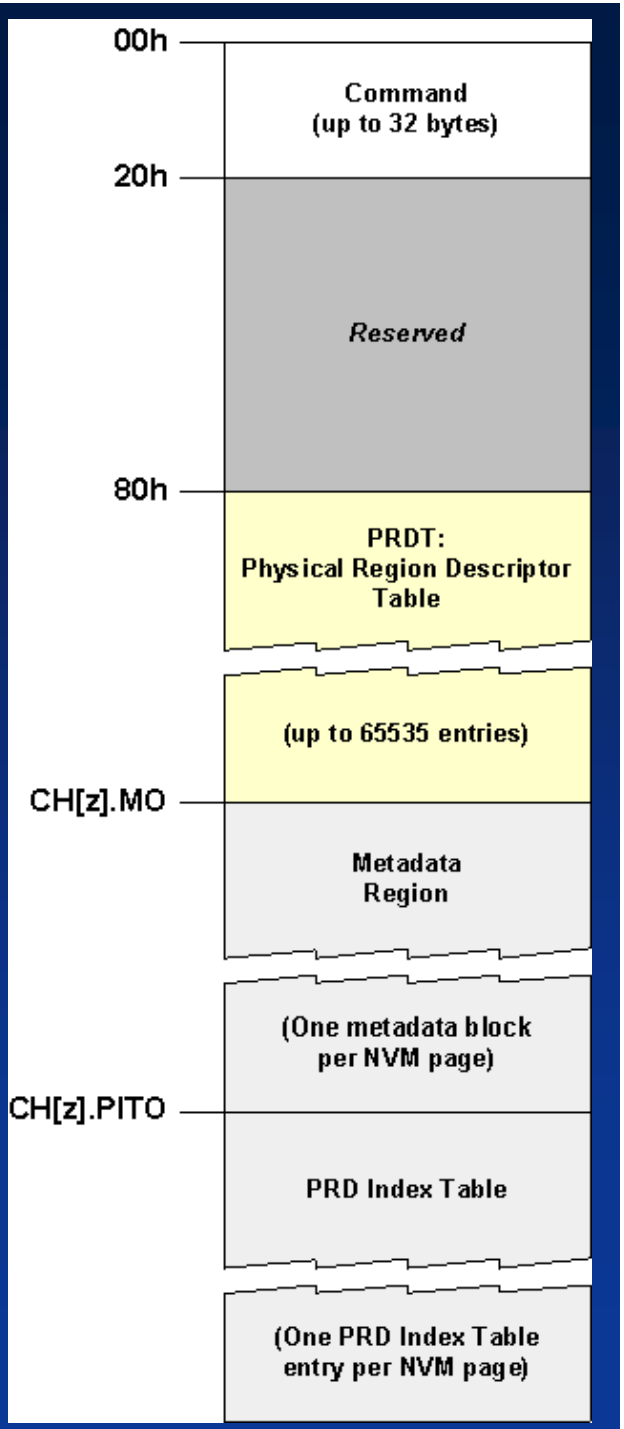
← Eliminate, packed

← Eliminate, packed

← Eliminate

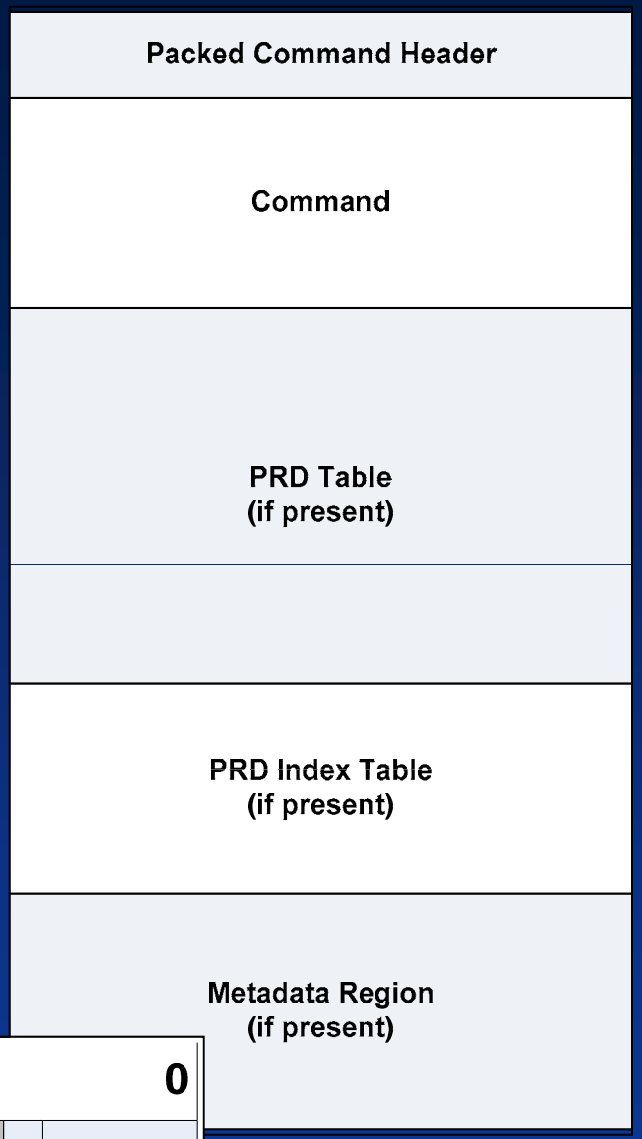
Command Table Changes

- Pack all regions that are present together in a fixed order
- Fixed Order: Command, PRD Table, PRD Index Table, Metadata Region
- If PRD Table, Metadata Region and/or PRD Index Table are present, it is indicated in the packed command header



Packed Command Structure

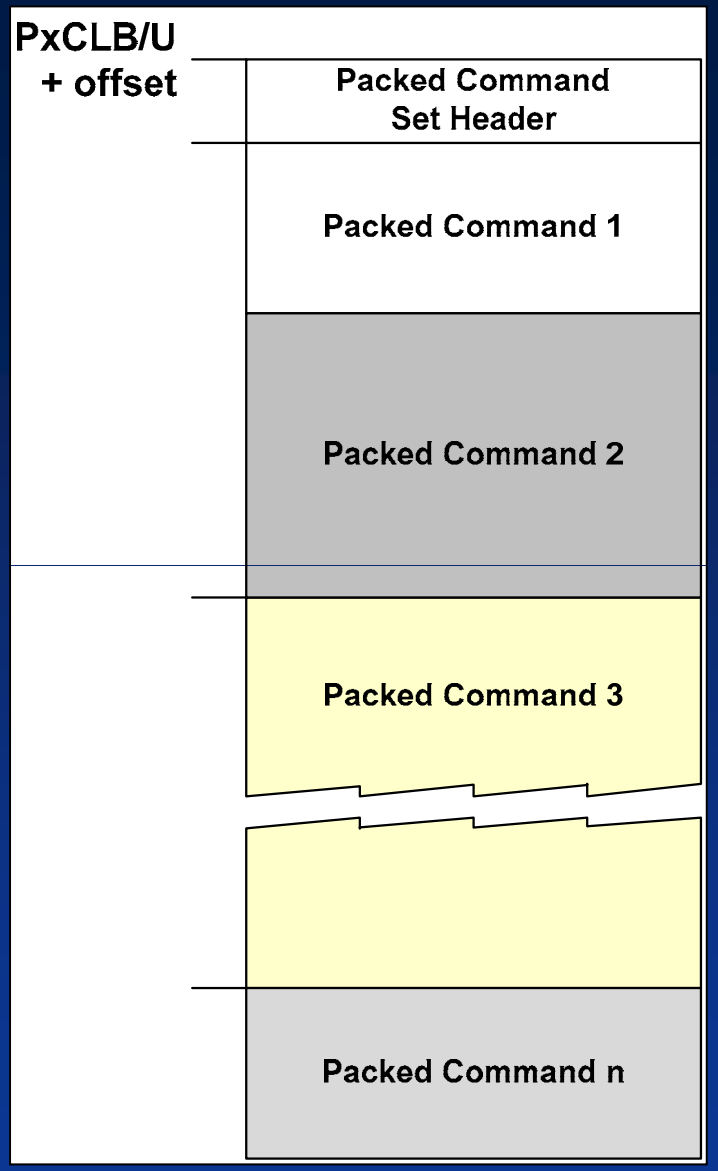
- The packed command structure enables hardware to fetch the entire command & associated data structures in one DMA operation
- The packed command header is an optimized version of the original command header
  - The structure size is configurable, for expansion

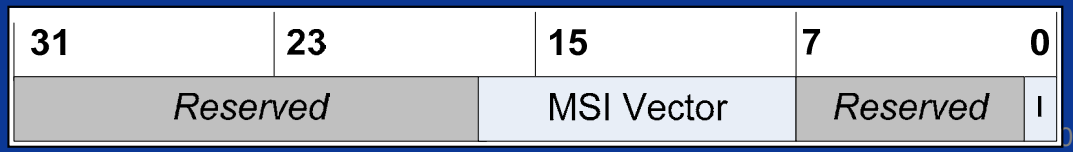


	31	23	15	7				0
DW0	PRDTL		*Reserved*	DT	W	R	P	CL
DW1	Command Status							

Packed Command Sets

- NVMHCI allows 32 commands per port
  - Suitable for client, but more is needed for Enterprise usage models
- To address this, allow software to issue packed command ***sets***
  - A set is a group of packed commands, joined together
- Packed command sets easily grow the number of commands per port
  - E.g., software places 8 commands in each set yields 256 commands for the port
- Advantage: Interrupt/status coalescing
  - This mechanism naturally lends itself to one interrupt and one successful status when the entire set completes





Packed Command Set Header

Issuing a Packed Command Set

- Use PxCI to track sets of commands that have been issued (hardware sets & clears)
- Use a new register "Command Set Launch" to launch a set of commands associated with a PxCI bit
  - Command set starts at an offset from PxCLB/PxCLBU
  - Indicates how much hardware should fetch from location

Command Issue (PxCI)

Bit	Type	Reset	Description
31:0	RO	0	**Commands Issued (CI):** This field is bit significant. Each bit corresponds to a packed set of commands issued by software. When set to '1', the corresponding set of commands is outstanding. When cleared to '0', the corresponding set of commands has been completed.

Command Set Launch (PxCSL)

Bit	Type	Reset	Description
31:20	RW	0h	**Command Set Size (CSS):** Indicates the size of the set of commands to be issued. This size is in 16 byte units.
19:08	RW	0h	**Command Base Offset (CBO):** Indicates the offset from the PxCLB/PxCLBU address that the command set starts at. This offset is in 1KB units.
07:05	RO	0h	Reserved
04:00	RW	0h	**Tracking Index (TI):** Indicates the tracking index used for this set of commands in the Commands Issued register. Hardware sets the index bit in PxCI to '1' when this register is written. Hardware cleared the index bit in PxCI to '0' when this set of commands is complete.

Achieving Very High IOPs

- 4KB read performance is a key metric
- To achieve high IOPs for 4KB reads, need to have a simple, small command
- With streamlining, it's possible to fit a 4KB read within a *64B* packed command

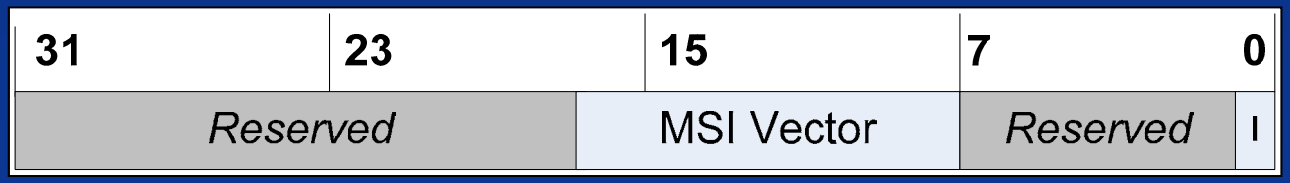
	31	23	15	7	0
DW0	*Reserved*		MSI Vector	*Reserved*	I
DW1	PRDTL		*Reserved*	DT W R P	CL
DW2	Command Status				
DW3					
DW4	**Command (20 bytes)**				
DW5					
DW6					
DW7					
DW8	**PRD Table (2 entries = 32 bytes)**				
DW9					
DW10					
DW11					
DW12					
DW13					
DW14					
DW15					

Another Key: The Completion Path

- In NVMHCI 1.0, the completion flow is:
  - Software reads the IS register, determines port that has interrupt
  - Software reads the PxIS register, determines interrupt type
  - Software reads the PxCI register to determine the commands that have completed
  - For each command that has completed, software reads the status to determine success, failure, or additional actions required
  - Software writes the PxIS register to clear the interrupt
  - Software writes the IS register to clear the interrupt
- Items to streamline:
  - Accesses to the IS register should be avoided (by using MSI-X)
  - Reads to MMIO registers should be optimized to minimum possible (each read takes > 2000 clocks)
  - For success, looking at specific command status for each command (especially in a command set) is inefficient

Improved Efficiency with MSI-X

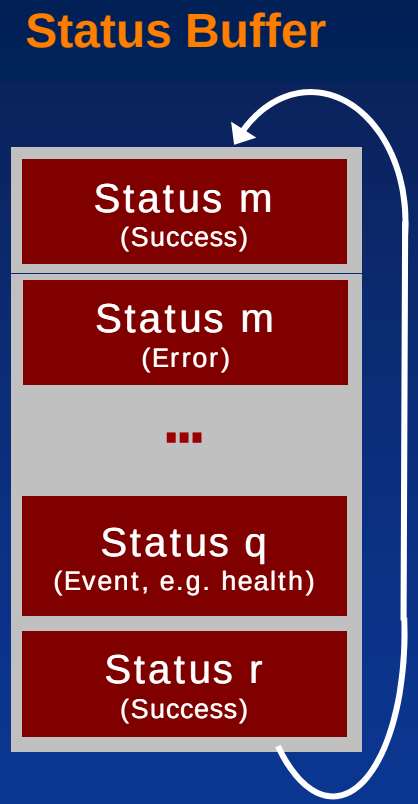
- The Interrupt Status (IS) register indicates the port that had the interrupt
  - When an interrupt is received, software checks this register and then starts reading the appropriate port registers
- This functionality can be replaced with MSI-X, where a particular MSI vector is specified that identifies the interrupting port
  - Eliminates the need to use / read the Interrupt Status register
- Software indicates the MSI vector that hardware should use in the packed command set header



Packed Command Set Header

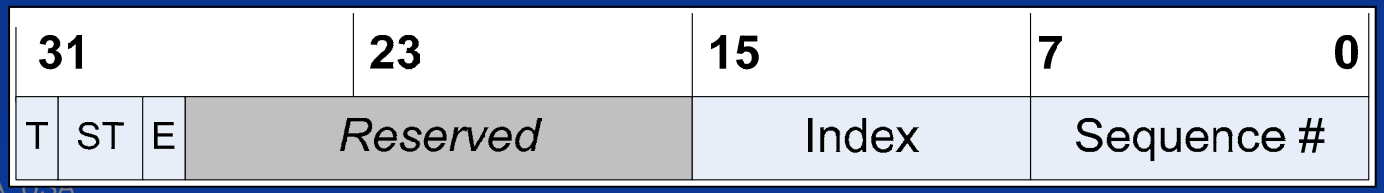
Streamlining Completions

- Optimize reading the command issue register and command status values into a single memory read
- Create a status buffer, a simple circular queue
  - All status items posted are a single Dword
- Event types:
  - Success: 1 Dword read that indicates an entire command set was successfully completed
  - Error: 1 Dword indicates particular command set had errors, refer to Status field in each command to determine individual status
  - Event: 1 Dword indicating that an event has occurred, software should then use the Get Status capability



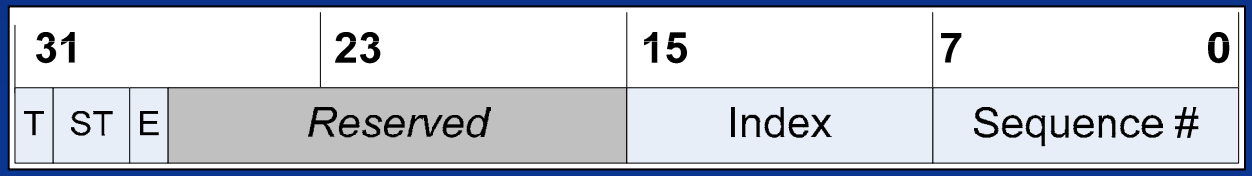
Status Buffer Entries

- For each event or command set completion, hardware inserts a Status Header Entry into the status buffer
- Status Header Entry fields:
  - Type (T): Whether this is a completion or an event
  - Status (ST): If this is a completion, specifies success or failure. If this is an event, specifies criticalness of the event.
  - Event (E): If this is a command completion, indicates if there is an event that still needs service
  - Index: If this is a command entry, this field is the bit in the Command Issue register that corresponds to this command set. If this is an event entry, this is the page type to read with the Get Status command.
  - Sequence #: Incremented by hardware on posting of each new status entry; used to determine if next entry is valid



Successful Command Set Completion

- Successful completion flow:
  - MSI vector is received, indicating port that has the interrupt
  - PxIS is read by software, indicating command completion
  - PxIS is cleared by software
  - Each valid entry in status buffer is read
  - Each success entry has the following attributes: Type = 1 (command), Status = 01b (success), Index = PxCI associated
- Completion is for all commands in the set with a single memory read



Successful completion is efficient: 1 interrupt, 1 MMIO read, 1 MMIO write, 1 memory read, 1 memory write.

Outline

- Remember: What is NVMHCI
- PCIe SSDs Coming… with Challenges
- Enterprise Extensions to NVMHCI
- *Summary*

Summary

- Lack of a standard host controller i/f impacts PCIe SSD adoption
- To address this gap, extend NVMHCI to meet the needs of Enterprise PCIe SSDs
  - Enables SSD vendors to focus on building a great SSD
  - Enables OS vendors to deliver a great driver for all PCIe SSDs
  - Enables OEMs to qualify a single driver on each OS, with features implemented in a consistent fashion, reducing time to market
- Solid approaches identified for scaling NVMHCI to high IOPs, given an SSD with richer resources
- Enterprise extensions to NVMHCI targeted for completion in Q1

Join the NVMHCI Workgroup and participate in the Enterprise Extensions development effort.
Visit http://www.intel.com/standards/nvmhci.