PCI Express 3.0 Overview

Jasmin Ajanovic

Sr. Principal Engineer

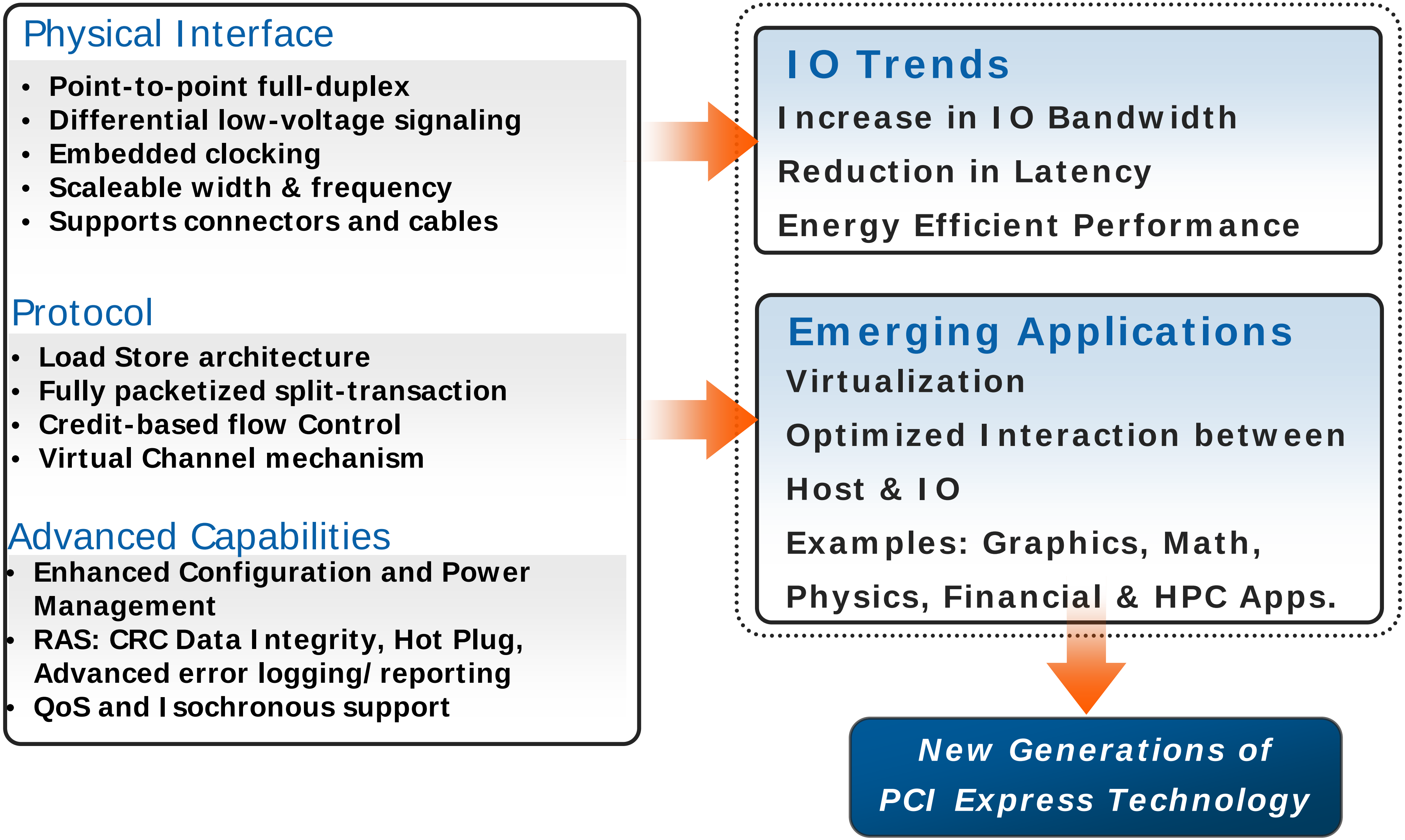
Intel Corp.

HotChips - Aug 23, 2009

Agenda

- PCIe Architecture Overview
- PCIe 3.0 Electrical Optimizations
- PCIe 3.0 PHY Encoding and Challenges
- New PCIe Protocol Features
- Summary & Call to action

PCI Express* (PCIe) Interconnect



Physical Interface

- Point-to-point full-duplex
- Differential low-voltage signaling
- Embedded clocking
- Scaleable width & frequency
- Supports connectors and cables

Protocol

- Load Store architecture
- Fully packetized split-transaction
- Credit-based flow Control
- Virtual Channel mechanism

Advanced Capabilities

- Enhanced Configuration and Power Management
- RAS: CRC Data Integrity, Hot Plug, Advanced error logging/ reporting
- QoS and Isochronous support

IO Trends

Increase in IO Bandwidth

Reduction in Latency

Energy Efficient Performance

Emerging Applications

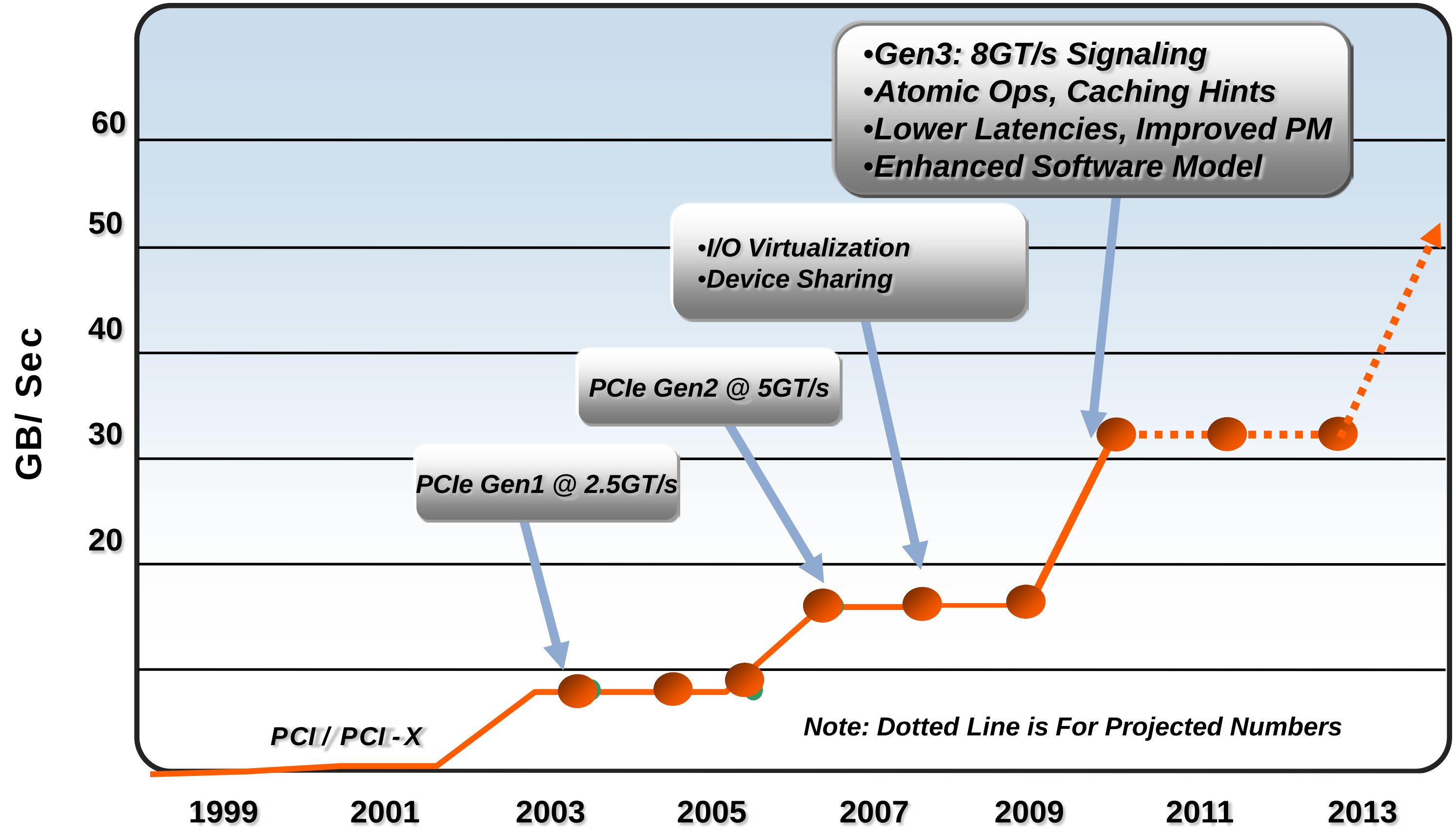
Virtualization

Optimized Interaction between Host & IO

Examples: Graphics, Math, Physics, Financial & HPC Apps.

New Generations of PCI Express Technology

PCIe Technology Roadmap



Note: Dotted Line is For Projected Numbers

	Raw Bit Rate	Link BW	BW/lane/way	BW x16
PCIe 1.x	2.5GT/s	2Gb/s	~250MB/s	~8GB/s
PCIe 2.0	5.0GT/s	4Gb/s	~500MB/s	~16GB/s
PCIe 3.0	8.0GT/s	8Gb/s	~1GB/s	~32GB/s

Based on x16 PCIe channel

All dates time frames and products are subject to change without further notification

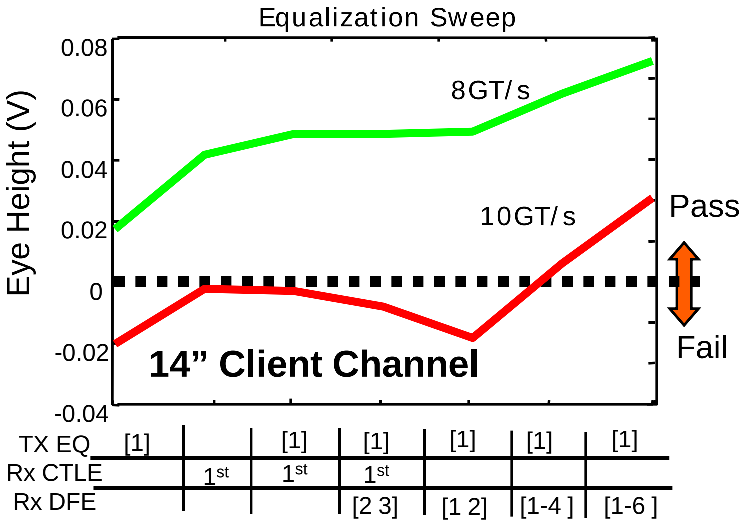
Continuous Improvement: Doubling Bandwidth & Improving Capabilities Every 3-4 Years!

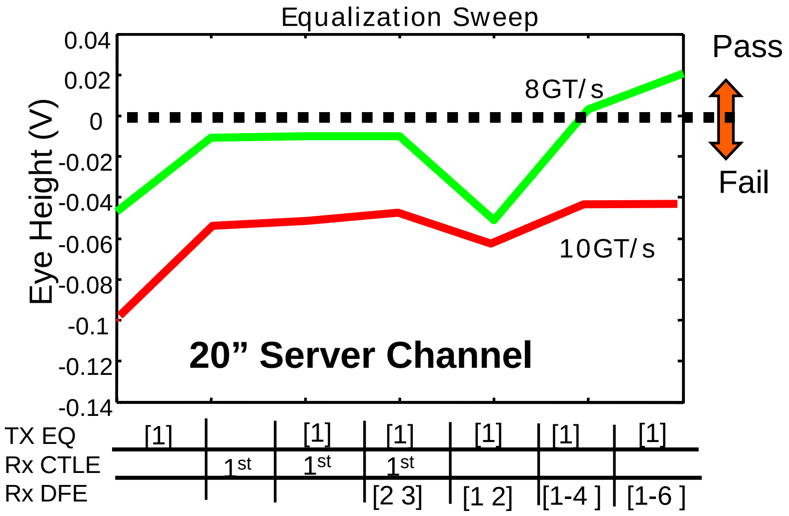
PCIe 3.0 Electrical Interface

PCIe 3.0 Electrical Requirements

- **Compatibility with PCIe 1.x, 2.0**
- **2x payload performance bandwidth over PCIe 2.0**
- **Similar cost structure (i.e. no significant cost adders)**
- **Preserve existing data clocked and common clock architecture support**
- **Maximum reuse of HVM ingredients**
  - FR4, reference clocks, etc.
- **Strive for similar channel reach in high-volume topologies**
  - Mobile: 8”, 1 connector
  - Desktop: 14”, 1 connector
  - Server: 20”, 2 connectors

PCIe Gen3 Solution Space





Source: Intel Corporation

- **Solution space exists to satisfy 8GT/s client and server channels requirements**
  – Power, channel loss and distortion much worse at 10GT/s
  – Similar findings by PCI-SIG members corroborated Intel analysis

- **PCI-SIG approved 8GT/s as PCIe 3.0 bit rate**

CTLE= Continuous Time Linear Equalizer

DFE= Decision Feedback Equalizer

Enabling Factors for 8G

- **Scrambling permits 2x payload rate increase wrt. Gen2 with 8 GT/s data rate**
  - Scrambling eliminates 25% coding overhead of 8b/10b
  - 8G chosen over 10G due to eye margin considerations
- **More capable Tx de-emphasis**
  - One post cursor tap and one pre cursor tap (2.5 and 5G has 1 post cursor tap)
  - Six selectable presets cover most equalization requirements
  - Finer Tx equalization control available by adjusting coefficients
- **Receiver equalization**
  - 1st order LE (linear eq.) is assumed as minimum Rx equalization
  - Designs may implement more complex Rx equalization to maximize margins
  - Back channel allowing Rx to select fine resolution Tx equalization settings
- **BW optimizations for Tx, Rx PLLs and CDR**
  - PLL BW reduced, CDR (Clock Data Recovery) jitter tracking increased
  - CDR BW > 10 MHz, PLL BW 2-4 MHz

PCIe 3.0 Encoding/Signaling

Problem Statement

- **PCI Express* (PCIe) 3.0 data rate decision: 8 GT/s**
  - High Volume Manufacturing channel for client/ servers
    - Same channels and length for backwards compatibility
    - Low power and ease of design - avoid using complicated receiver equalization, etc.

- **Requirement: Double Bandwidth from Gen 2**
  - PCIe 1.0a data rate: 2.5 GT/s
  - PCIe 2.0 data rate: 5 GT/s
    - Doubled the bandwidth from Gen 1 to Gen 2 by doubling the data rate
  - Data rate gives us a 60% boost in bandwidth
  - Rest will come from Encoding
    - Replace 8b/10b encoding with a scrambling-only encoding scheme when operating at PCIe 3.0 data rate

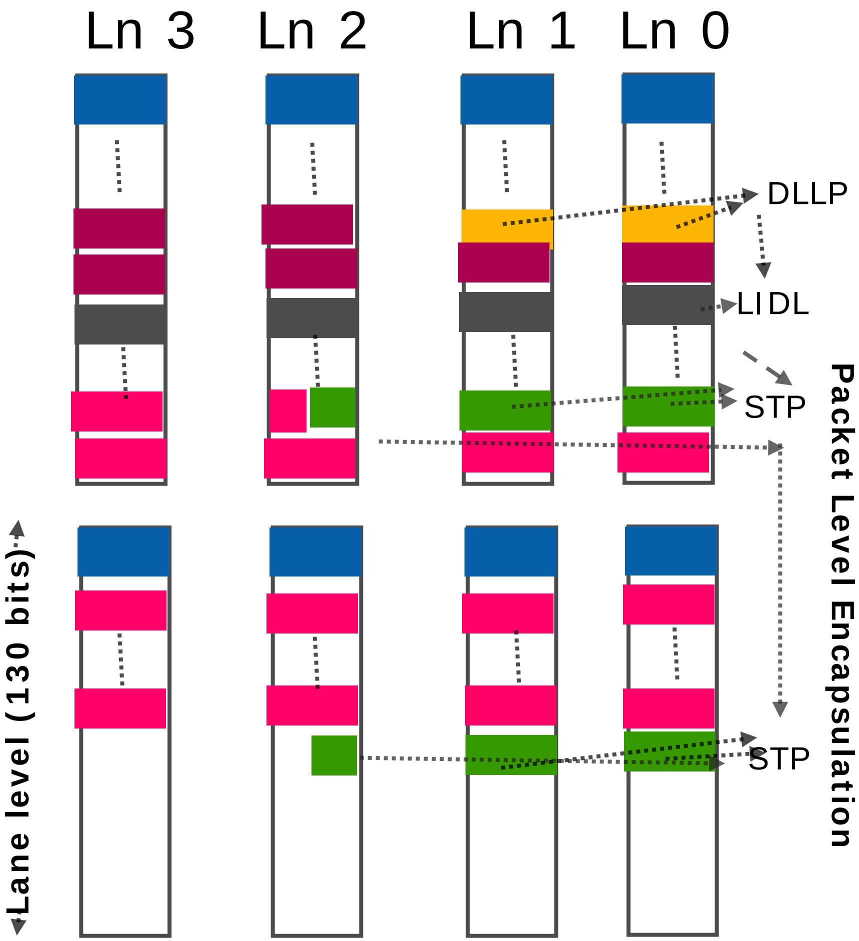
- **Double B/W: Encoding efficiency 1.25 X data rate 1.6 = 2X**

Challenge: New Encoding Scheme to cover

256 data plus 12 K-codes with 8 bits

New Encoding Scheme

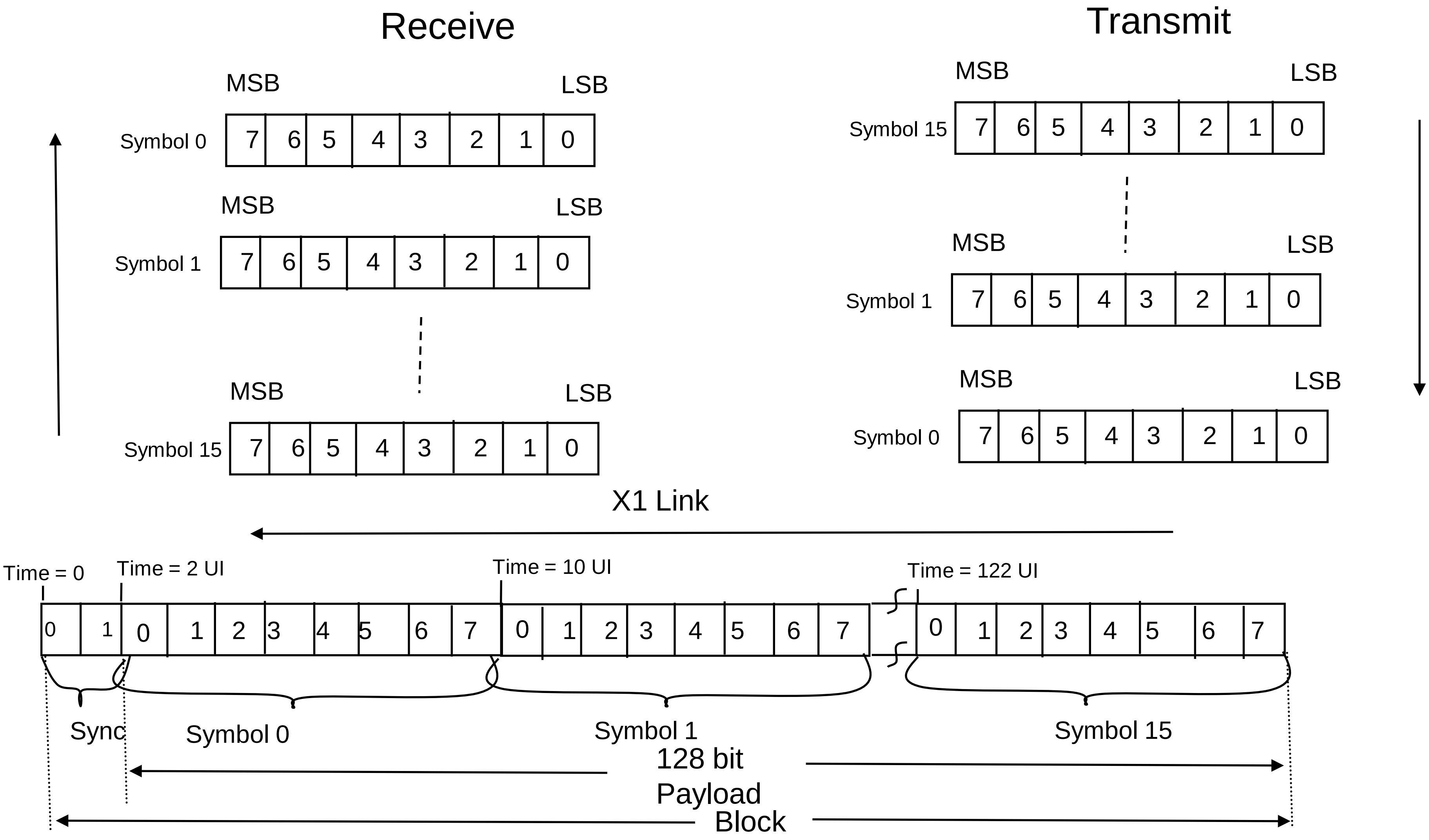
- Two levels of encapsulation
  - Lane Level (mostly 128/130)
  - Packet Level to identify packet boundaries
    - Point to where next packet begins
- Additive Scrambling only (no 8b/10b) to provide edge density
  - Data Packets scrambled
    - TLP/ DLLP/ LIDL
  - Ordered Sets mostly not scrambled
  - Electrical Idle Exit Ordered Set resets scrambler (Recovery/ Config)



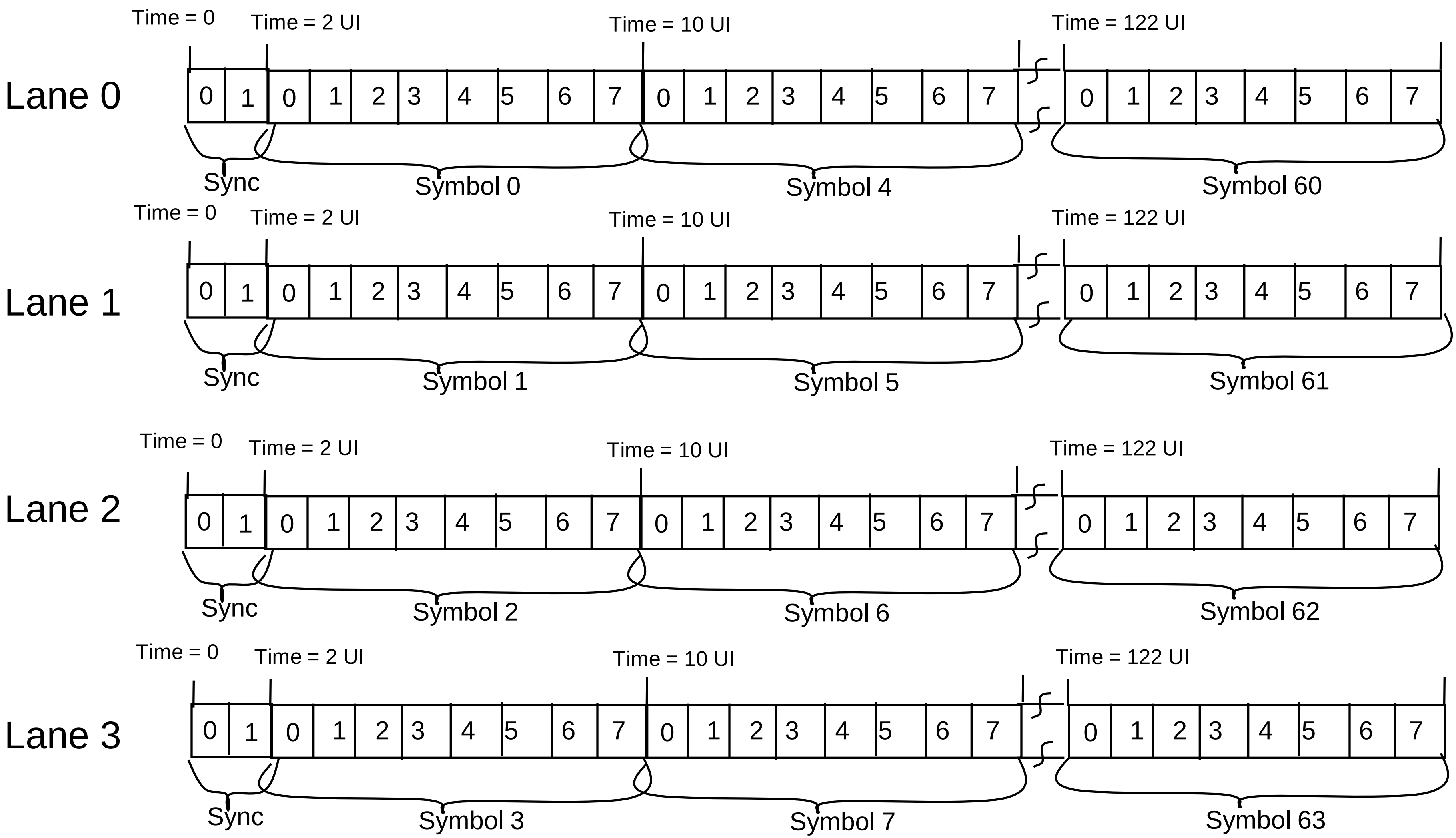
Source: Intel Corporation

Scrambling with two levels of encapsulation

Mapping of bits on a x1 Link



Mapping of bits on a x4 Link



P-Layer Encapsulation: TLP

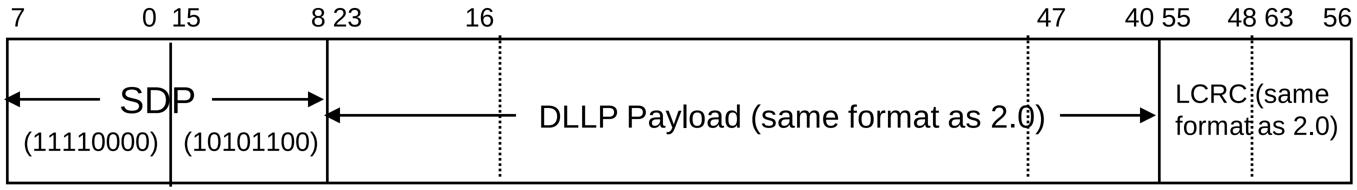
7 4	3 0	15	14 8	23 20	19 16	31 24	39 32 …	n-1 n-8
Len[3:0]	STP (1111)	P	Len [10:4]	Frame CRC [3:0]	Seq No [11:8]	Seq No [7:0]	← TLP Payload (same format as 2.0) →	LCRC (4B, same format as 2.0)

[Len[10:0]: length of the TLP in DWs, Frame CRC[4:0]: Check Bits covering Length[0:10], P: Frame Parity, No END]

- **Length known from the first 3 Symbols**
  - First 4 bits are 1111 (bit[0:3] = 4'b1111)
  - Bits 4:14 has the length of the TLP (valid values: 5 to 1031)*
  - Bits 15 and 20:23 is check bits to cover the TLP Length field
    - Primitive Polynomial (X4 + X + 1) protects 15 bit field
      - Provides double bit flip detection guarantee (length 11 bits + CRC 4 bits)
    - Odd parity covers the 15 bits (length 11 bits + CRC 4 bits)
      - Guaranteed detection of triple bit errors (over 16 bits)
- **Sequence Number occupies bits 16:19 and 24:31**
- **TLP payload is from the 4th Symbol position (same as 2.0)**
- **No explicit END - Check 1st Symbol after TLP for implicit END vs. an explicit EDB => Ensures triple bit flip detection**
- **All Symbols are scrambled/ de-scrambled**

*Note: Valid values for a TLP Prefix is 5 to ~ 1039 (Max value depends on type of TLP Prefix)

P-Layer Encapsulation: DLLP



7 0 15 8 23 16 47 40 55 48 63 56

SDP

(11110000) (10101100)

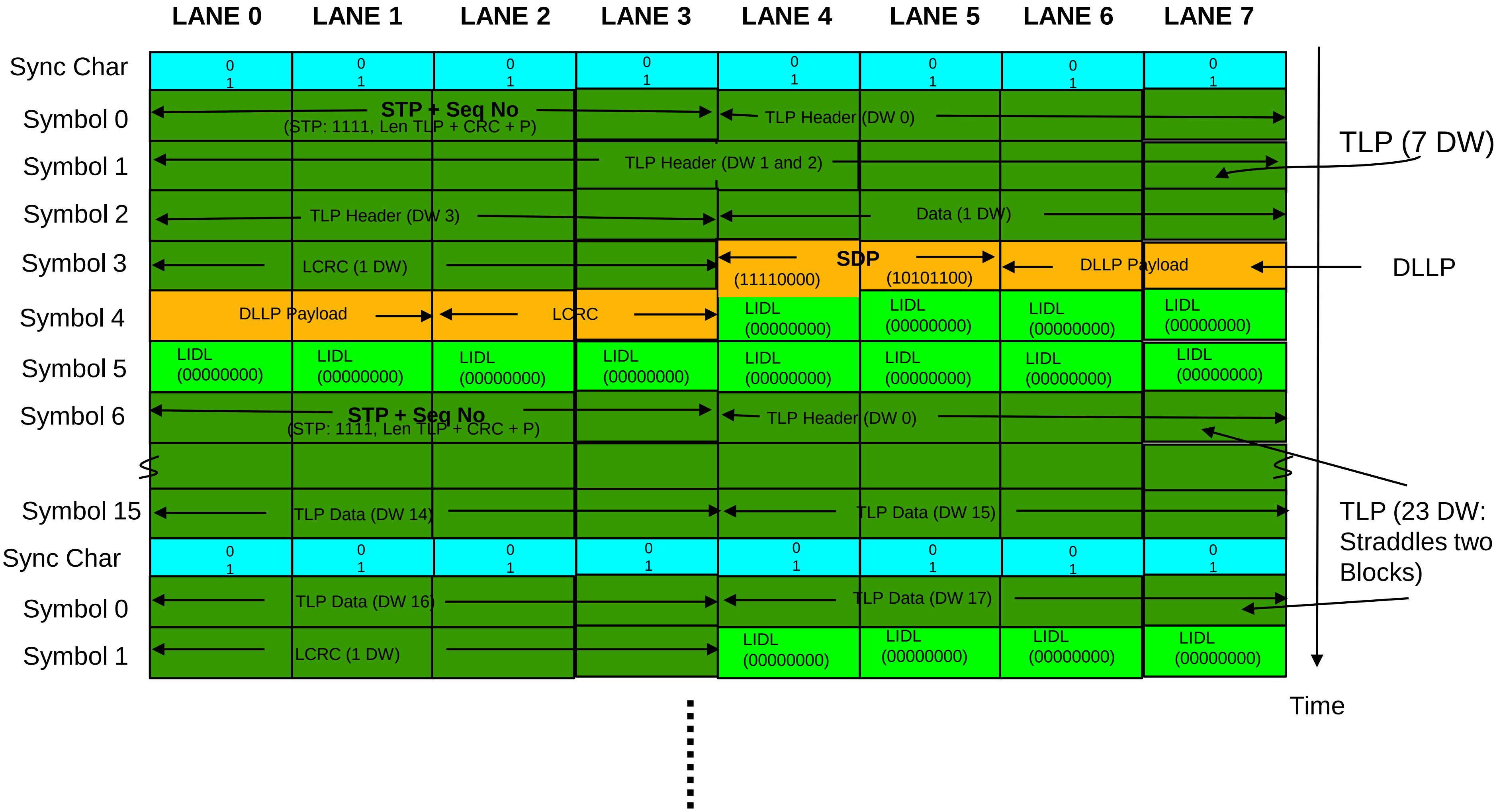
DLLP Payload (same format as 2.0)

LCRC (same format as 2.0)

(DLLP Layout)

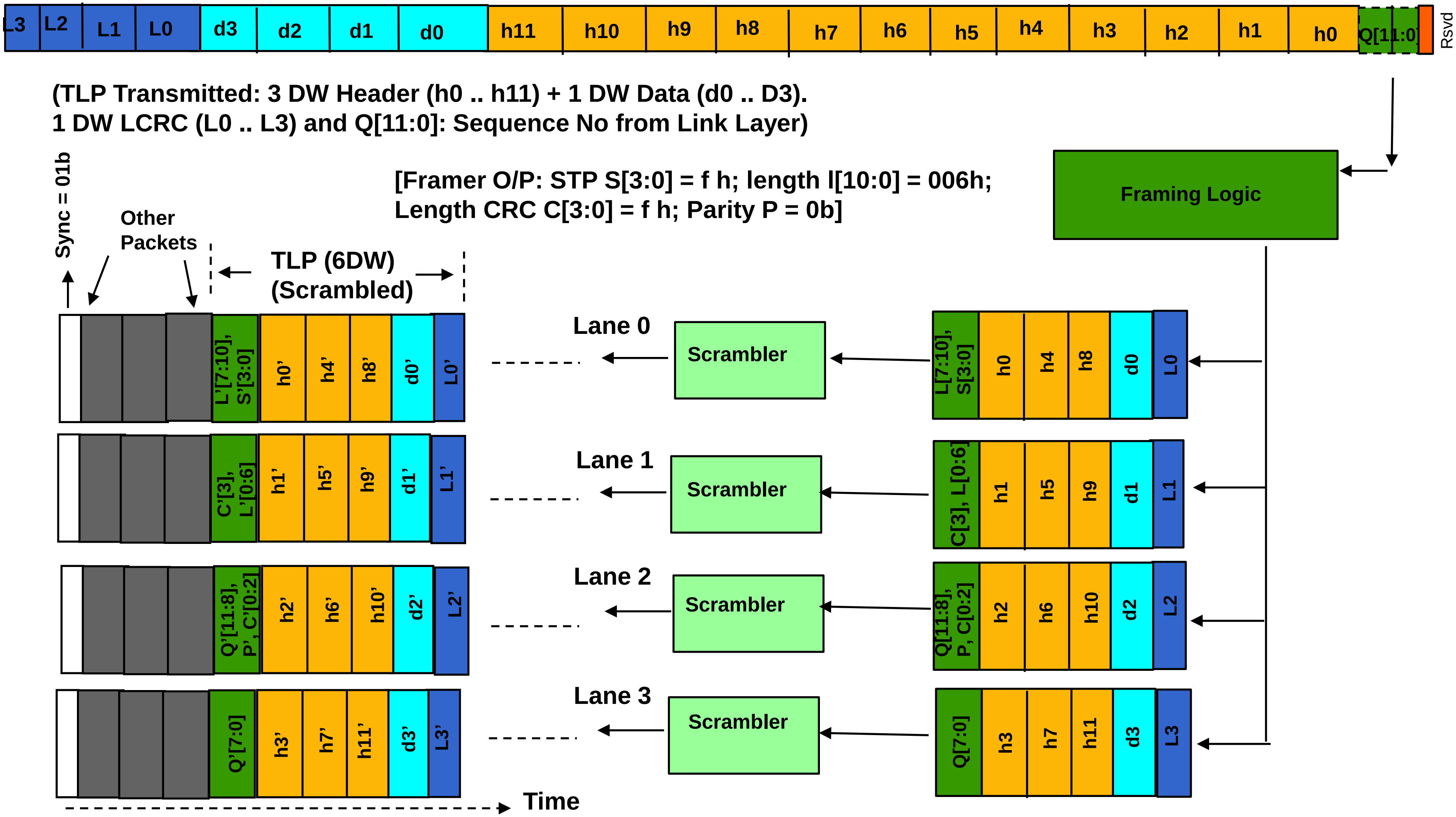
- **Preserve DLLP layout of 2.0 spec**
- **First Symbol is F0h**
- **Second Symbol is ACh**
- **Next 4 Symbols (2 through 5) are the DLLP layout**
- **Next 2 Symbols (6 and 7): LCRC (identical to 2.0)**
- **No explicit END**
- **All Symbols are scrambled/ de-scrambled**

Ex: TLP/ DLLP/ IDLs in x8



	LANE 0	LANE 1	LANE 2	LANE 3	LANE 4	LANE 5	LANE 6	LANE 7
Sync Char	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1
Symbol 0	**STP + Seq No** (STP: 1111, Len TLP + CRC + P)				TLP Header (DW 0)			
Symbol 1	TLP Header (DW 1 and 2)							
Symbol 2	TLP Header (DW 3)				Data (1 DW)			
Symbol 3	LCRC (1 DW)				**SDP** (11110000)	(10101100)	DLLP Payload	
Symbol 4	DLLP Payload		LCRC		LIDL (00000000)	LIDL (00000000)	LIDL (00000000)	LIDL (00000000)
Symbol 5	LIDL (00000000)	LIDL (00000000)	LIDL (00000000)	LIDL (00000000)	LIDL (00000000)	LIDL (00000000)	LIDL (00000000)	LIDL (00000000)
Symbol 6	**STP + Seq No** (STP: 1111, Len TLP + CRC + P)				TLP Header (DW 0)			
Symbol 15	TLP Data (DW 14)				TLP Data (DW 15)			
Sync Char	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1
Symbol 0	TLP Data (DW 16)				TLP Data (DW 17)			
Symbol 1	LCRC (1 DW)				LIDL (00000000)	LIDL (00000000)	LIDL (00000000)	LIDL (00000000)

TLP Transmission in a X4 Link

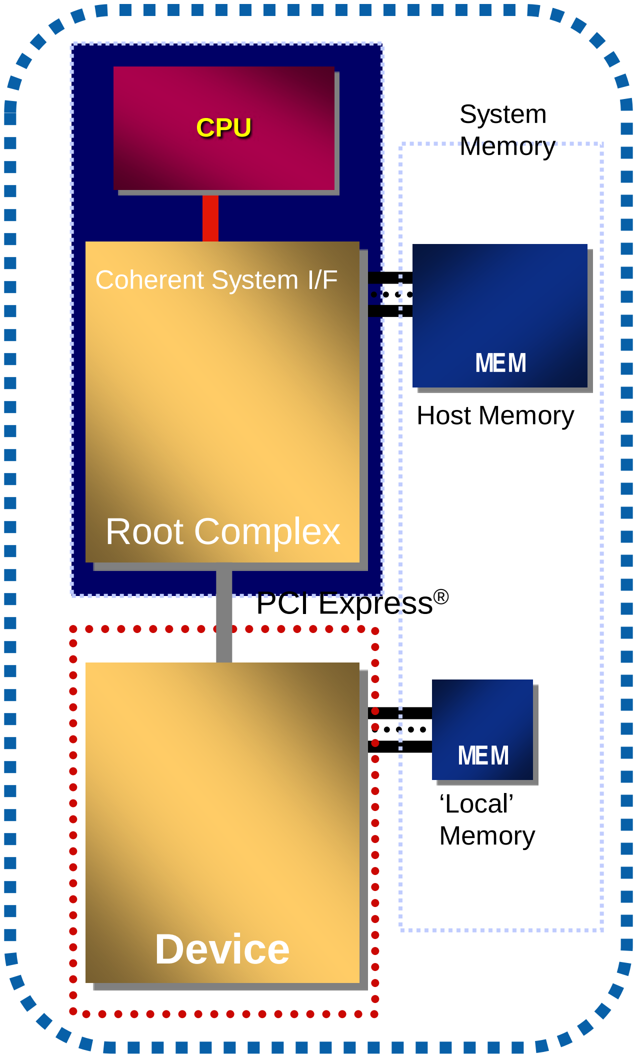


(TLP Transmitted: 3 DW Header (h0 .. h11) + 1 DW Data (d0 .. D3).
1 DW LCRC (L0 .. L3) and Q[11:0]: Sequence No from Link Layer)

[Framer O/P: STP S[3:0] = f h; length l[10:0] = 006h;
Length CRC C[3:0] = f h; Parity P = 0b]

PCIe 3.0 Protocol Extensions

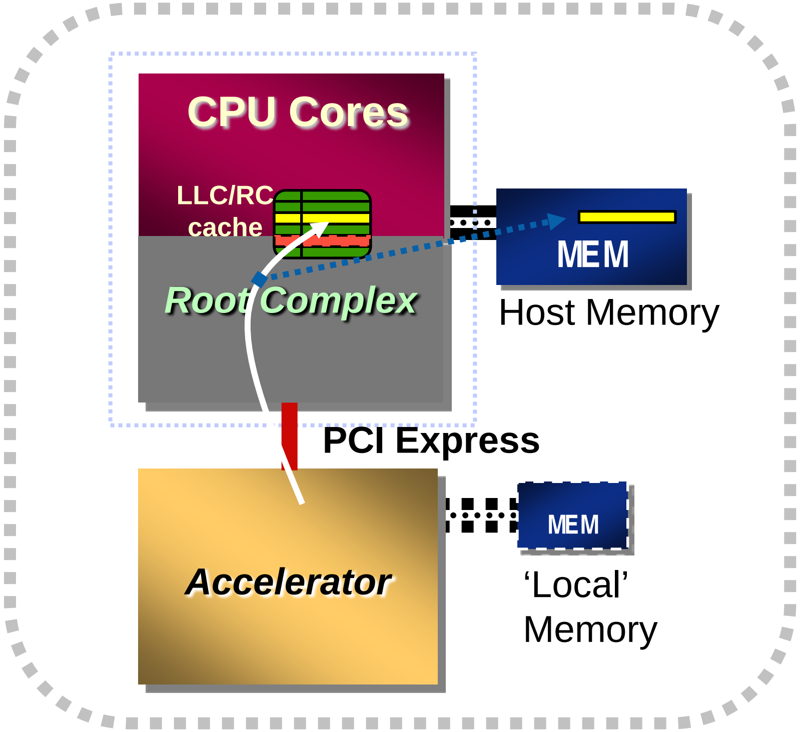
Protocol Extensions

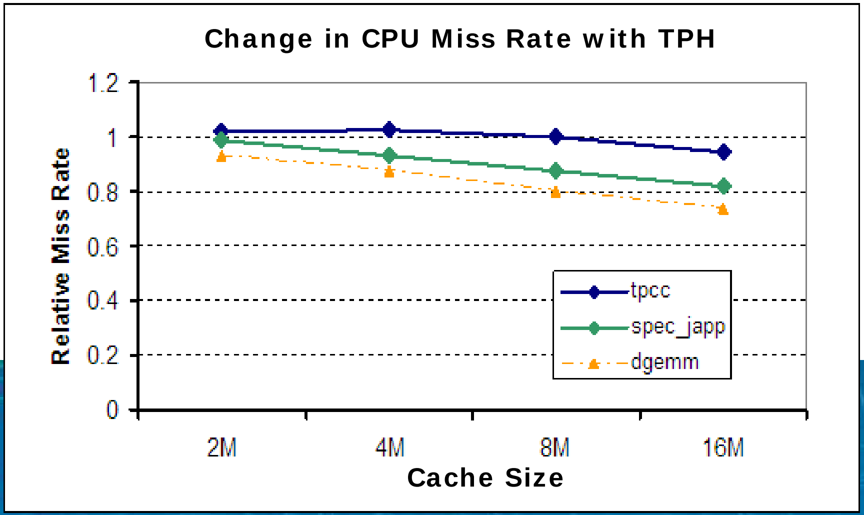


- **Performance Improvements**
  - **TLP Processing Hints** – hints to optimize system resources and performance
  - **TLP Prefix** – mech to extend TLP headers for TLP Processing Hints, MR-IOV, and future extensions
  - **ID-Based Ordering** – Transaction-level attribute/hint to optimize ordering within RC and memory subsystem
  - **Extended Tag Enable Default** – permits default for Extended Tag Enable bit to be Function-specific
- **Software Model Improvements**
  - **Atomic Operations** – new atomic transactions to reduce synchronization overhead
  - **Page Request Interface** – mech in ATS 1.1 for a device to request faulted pages to be made available (not covered)
- **Communication Model Enhancements**
  - **Multicast** – mechanism to transfer common data or commands sent from one source to multiple recipients
- **Power Management**
  - **Dynamic Power Allocation** – support for dynamic power operational modes through standard configuration mech
  - **Latency Tolerance Reporting** – Endpoints report service latency requirements for improved platform power mgmt
  - **Optimized Buffer Flush/Fill** – Mechs for devices to align DMA activity for improved platform power mgmt
- **Configuration Enhancements**
  - **Resizable BAR**– Mechanism to support BAR size negotiation
  - **Internal Error Reporting**– Extend AER to report component internal errors and record multiple error logs

TLP Processing Hints (TPH)

Transaction Processing Hints

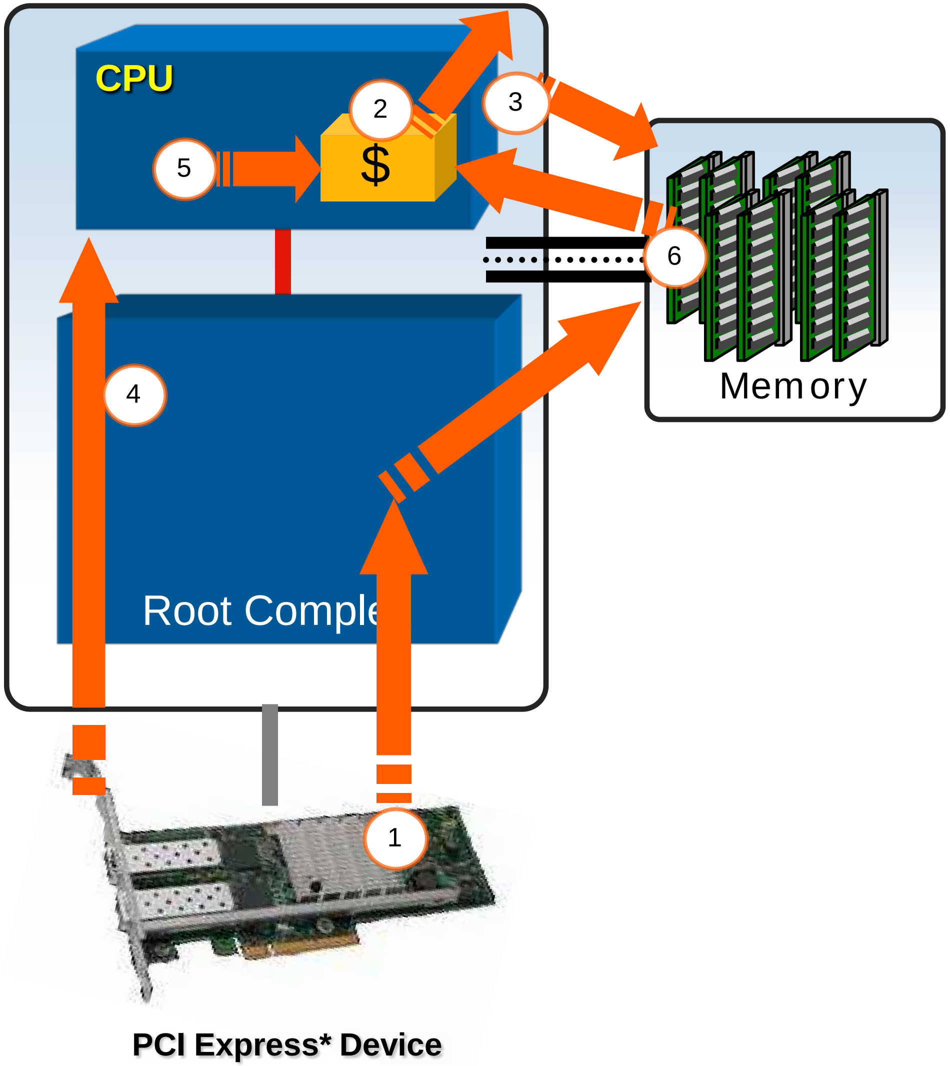


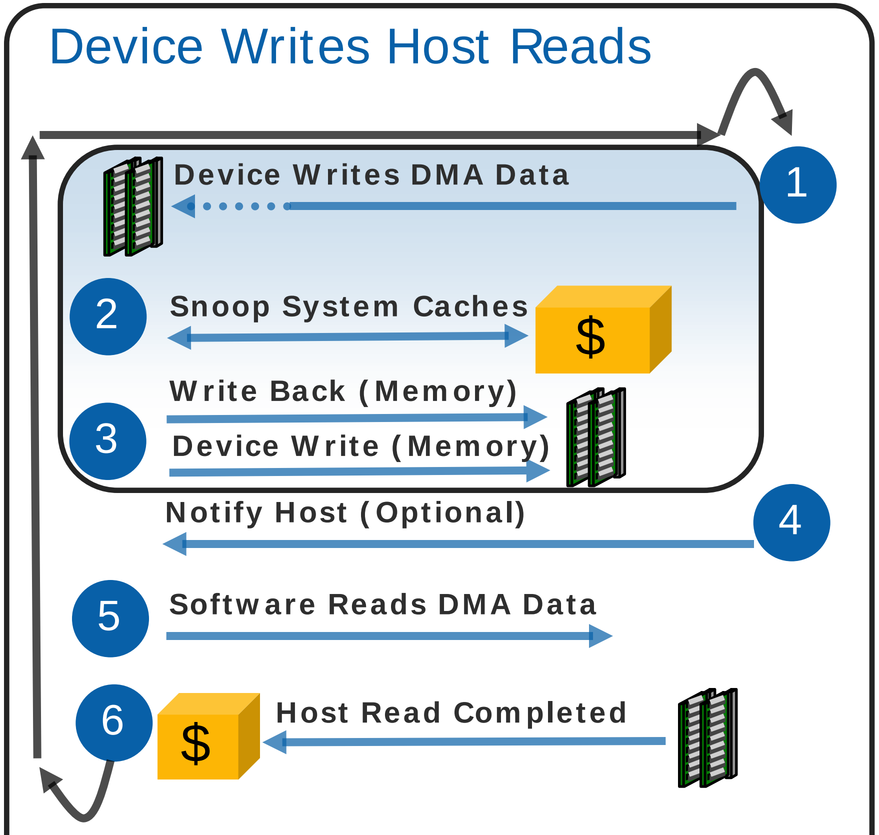


- Background:
  - Small IO Caches implemented in server platforms
    - Ineffective w/o info about intended use of IO data
- Feature:
  - TPH= hints on a transaction basis
    - Allocation & temporal reuse
  - More direct CPU<->IO collaboration
    - Control structures (headers, descriptors) and data payloads
- Benefits:
  - Reduced access latencies
    - Improved data retention/allocation
  - Reduced mem & QPI BW/ power
    - Avoiding data copies
  - New applications
    - Comm adapters for HPC and DB clusters, Computational Accelerators,…

Provides stronger coupling between Host Cache/Memory hierarchy and IO

Basic Device Writes





Device Writes Host Reads

1 Device Writes DMA Data

2 Snoop System Caches

3 Write Back (Memory)

Device Write (Memory)

4 Notify Host (Optional)

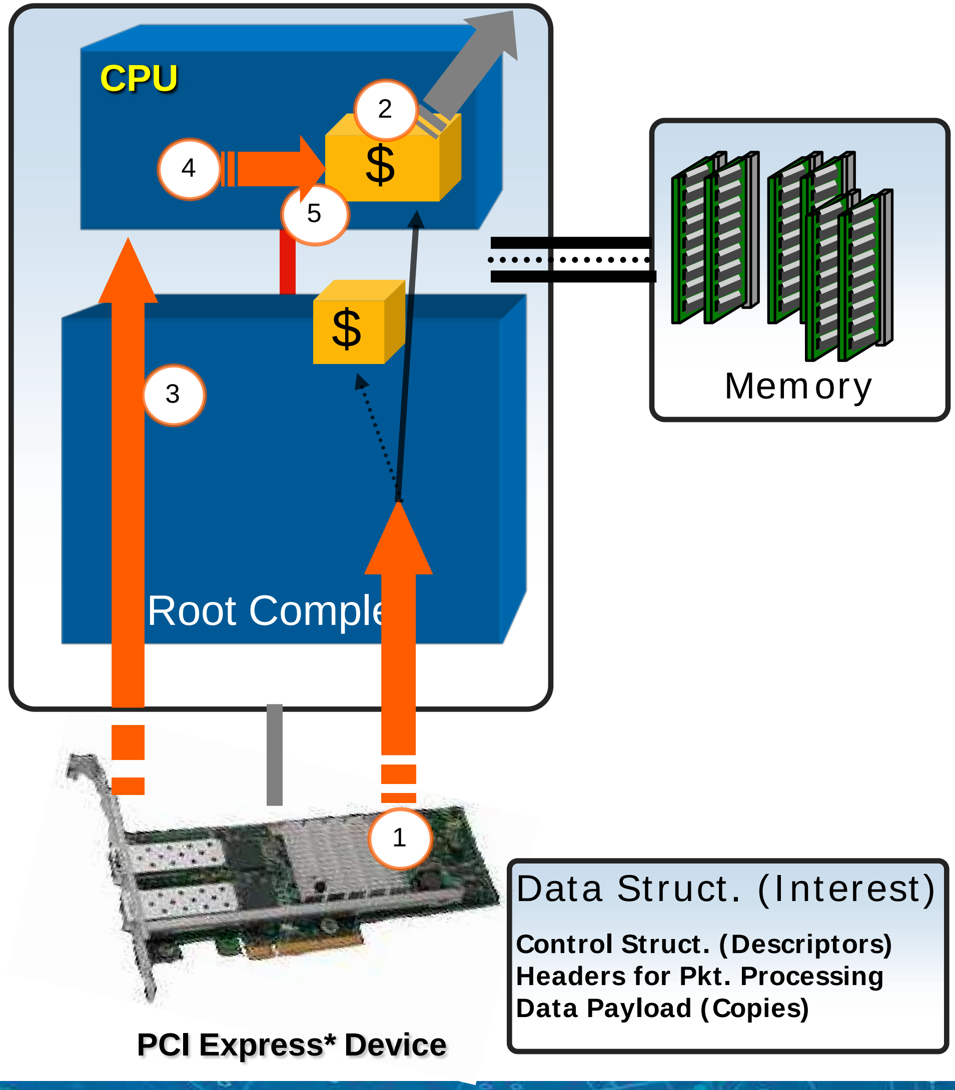
5 Software Reads DMA Data

6 Host Read Completed

Transaction Flow does not take full advantage of System Resources

- **System Caches**
- **System Interconnect**

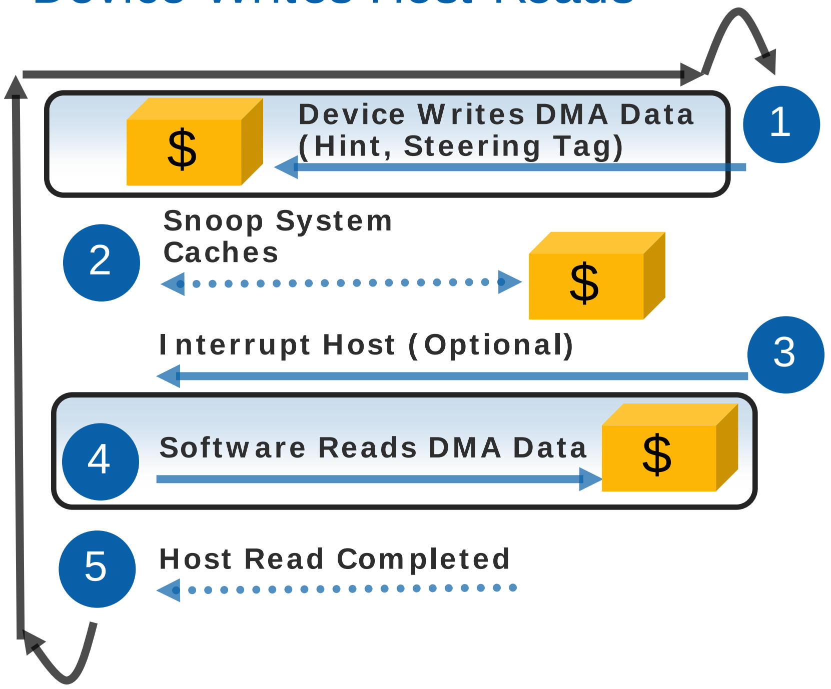
Device Writes with TPH



Data Struct. (Interest)

Control Struct. (Descriptors)
Headers for Pkt. Processing
Data Payload (Copies)

Device Writes Host Reads



1 Device Writes DMA Data (Hint, Steering Tag)

2 Snoop System Caches

3 Interrupt Host (Optional)

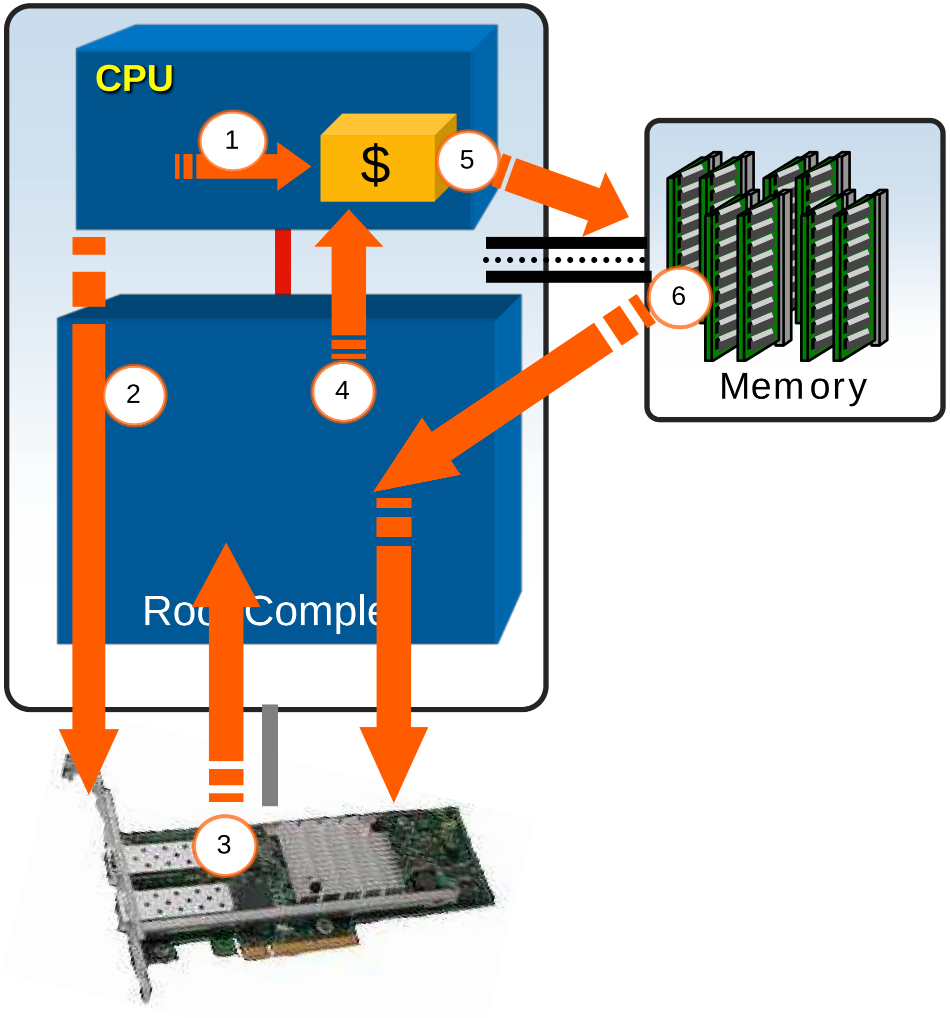
4 Software Reads DMA Data

5 Host Read Completed

Effective Use of System Resources

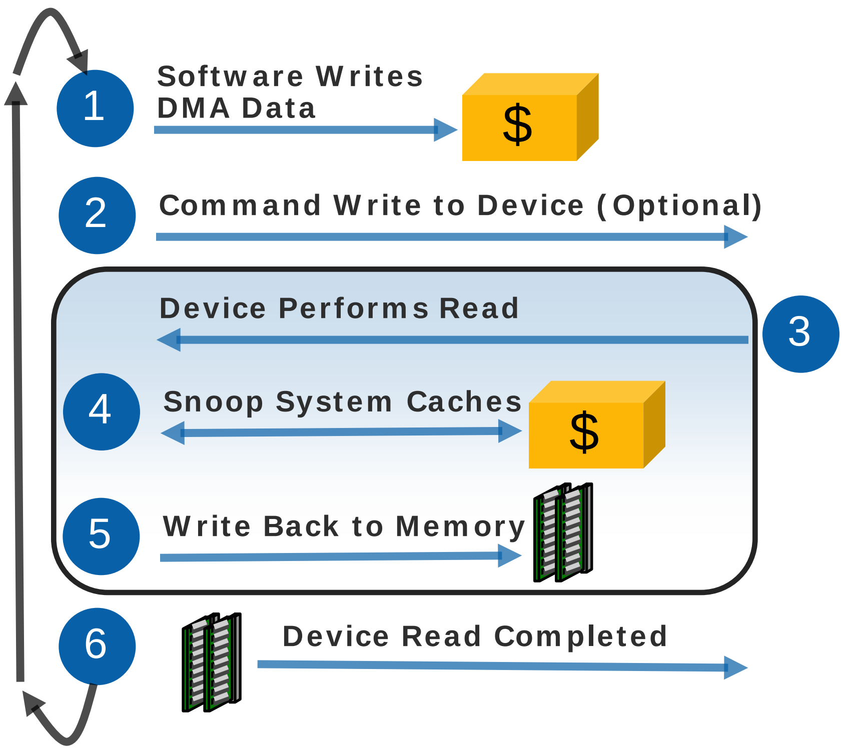
- **Reduce Access latency to system memory**
- **Reduce Memory & system interconnect BW & Power**

Basic Device Reads



PCI Express* Device

Host Writes Device Reads

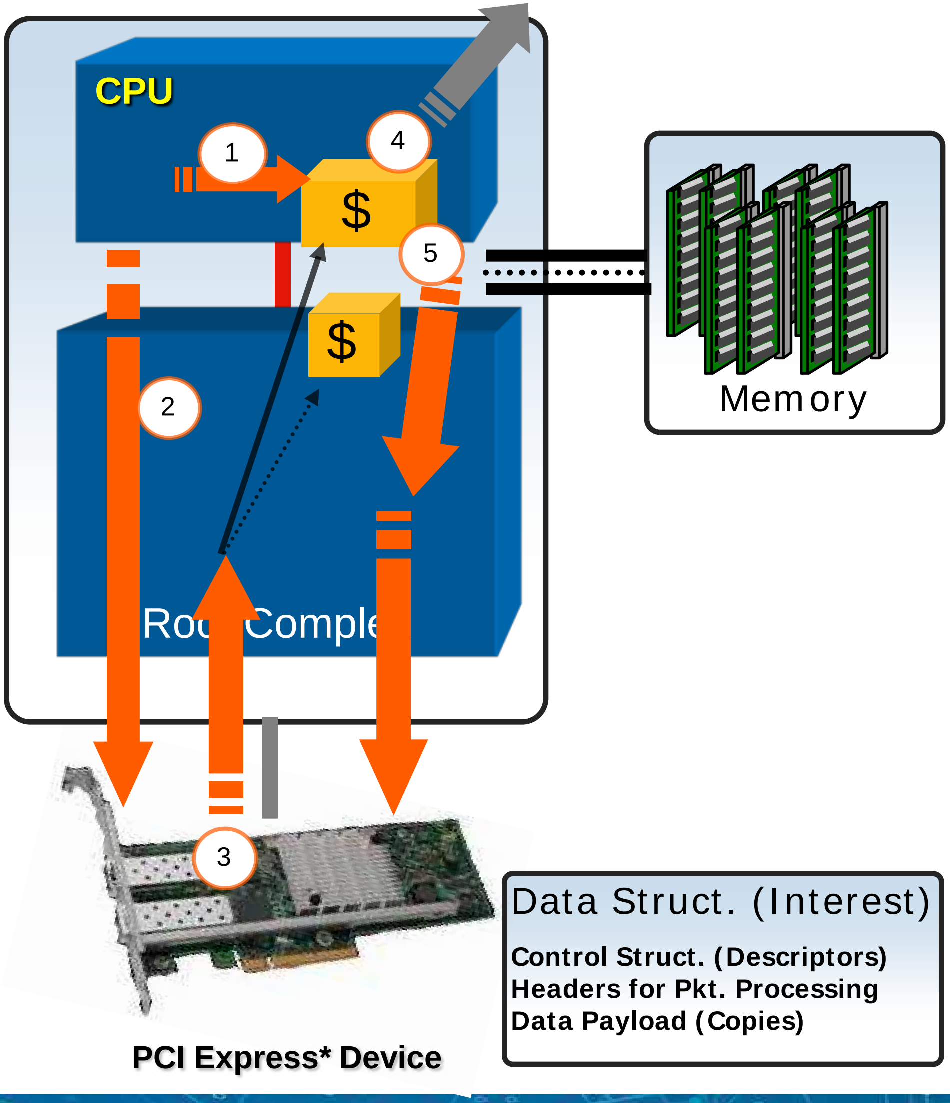


1. Software Writes DMA Data
2. Command Write to Device (Optional)
3. Device Performs Read
4. Snoop System Caches
5. Write Back to Memory
6. Device Read Completed

Transaction flow does not take full advantage of System Resources

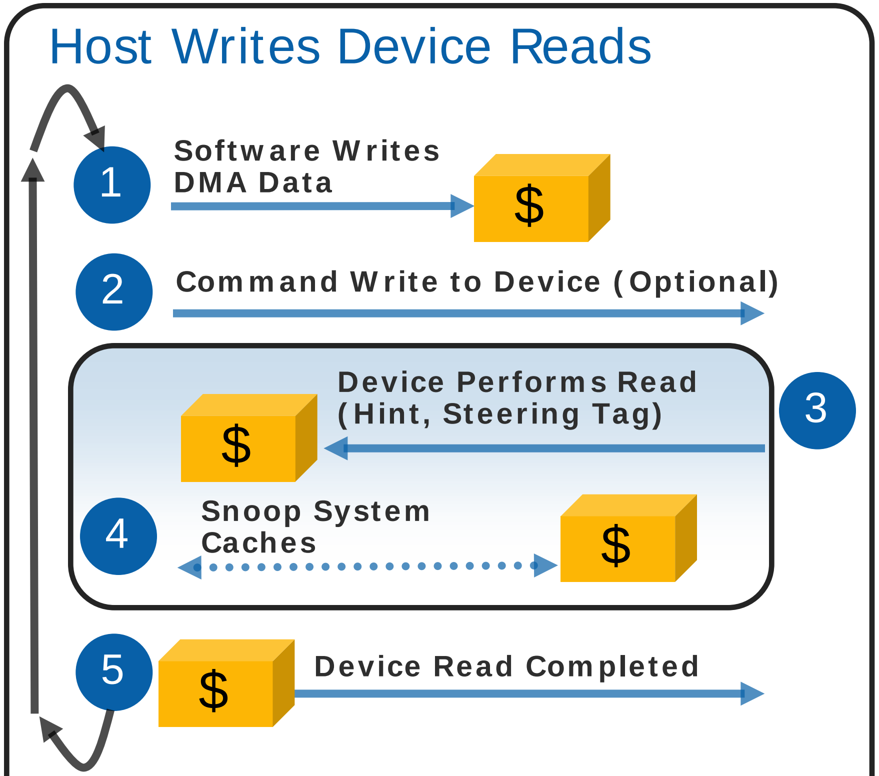
- **System Caches**
- **System Interconnect**

Device Reads with TPH



Data Struct. (Interest)

Control Struct. (Descriptors)
Headers for Pkt. Processing
Data Payload (Copies)

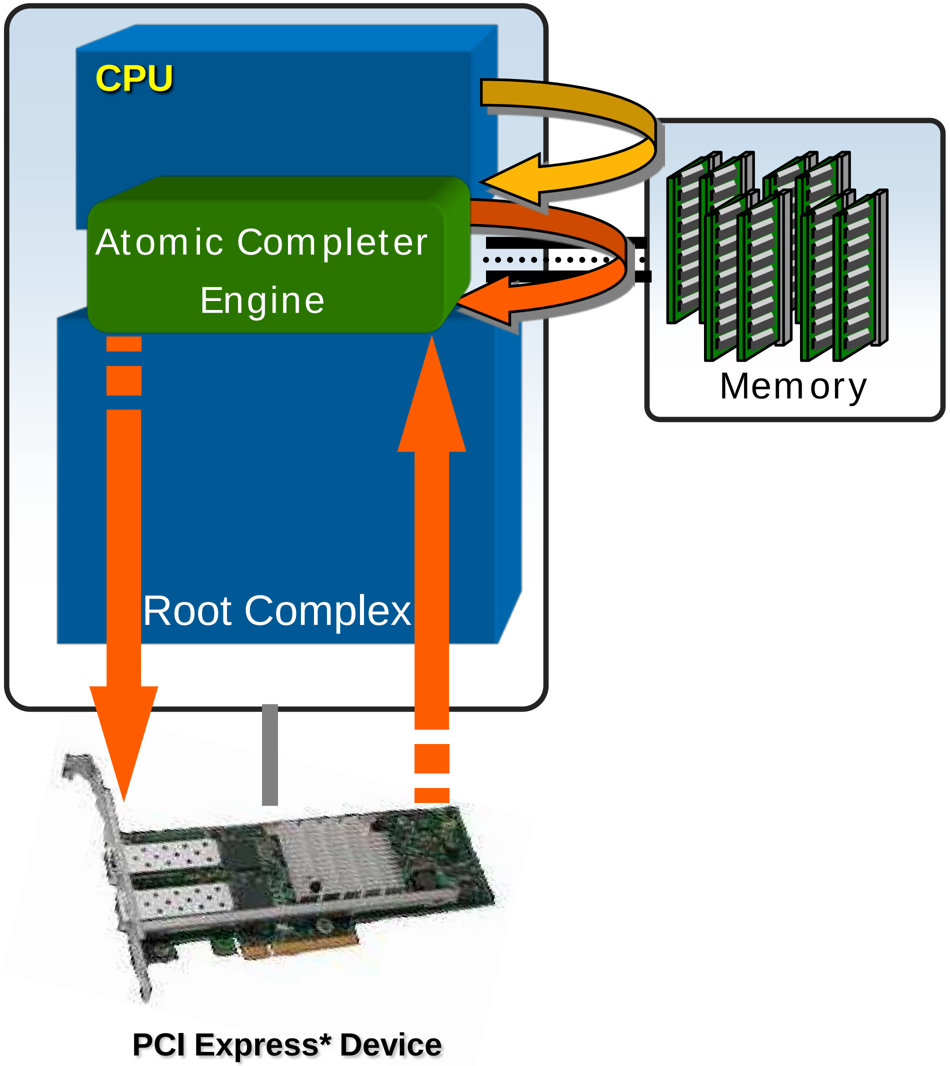


Effective Use of System Resources

- **Reduce Access latency to system memory**
- **Reduce Memory & system interconnect BW & Power**

Atomic Operations (AtomicOps)

Synchronization

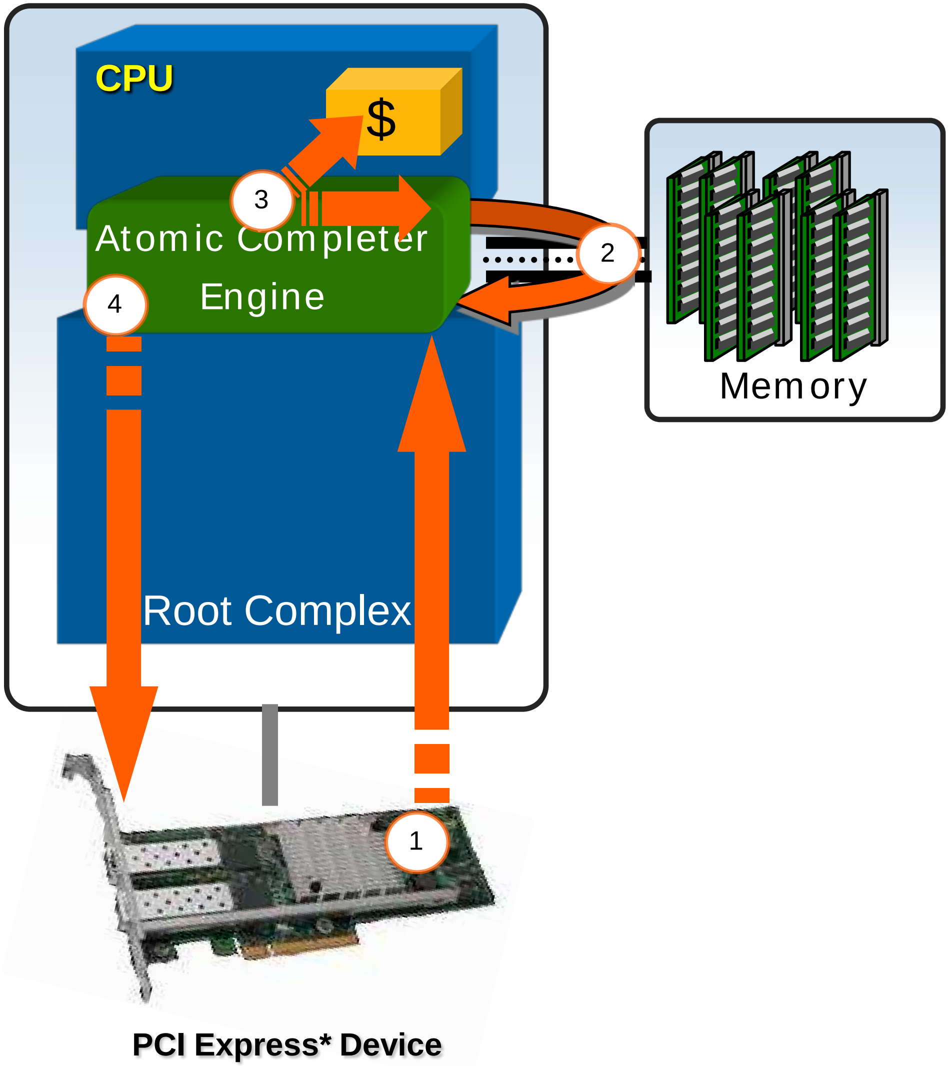


PCI Express* Device

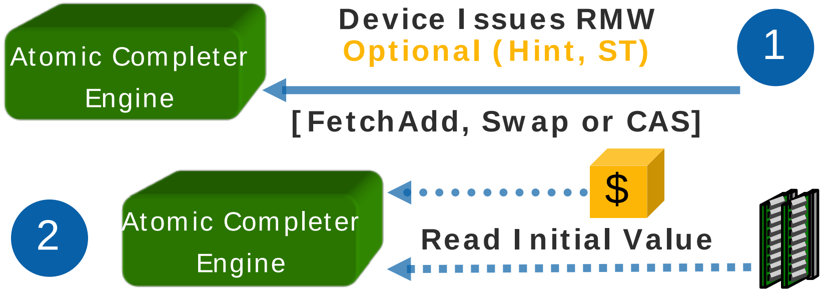
Atomic Read-Modify-Write

- **Atomic transaction support for Host update of main memory exists today**
  - Useful for synchronization without interrupts
  - Rich library of proven algorithms in this area
- **Benefit in extending existing inter-processor primitives for data sharing/ synchronization to PCI e interconnect domain**
  - Low overhead critical sections
  - Non-Blocking algorithms for managing data structures e.g. Task lists
  - Lock-Free Statistics e.g. counter updates
- **I mprove existing application performance**
  - Faster packet arrival rates create demand for faster synchronization
- **Emerging applications benefit from Atomic RMW**
  - Multiple Producer – Multiple Consumer support
  - Example: Math, Visualization, Content Processing etc

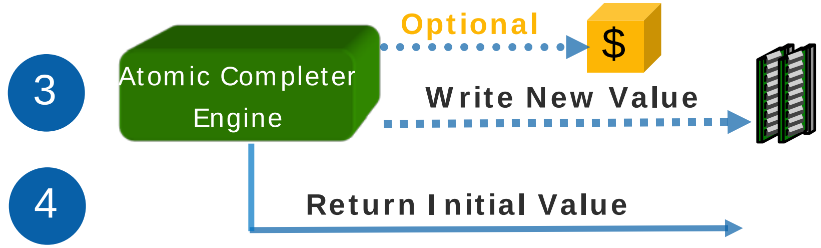
Atomic Read-Modify-Write (RMW)



Atomic RMW Operation



Request	Description
FetchAdd	**Data(Addr) = Data(Addr) + AddData**
Swap	**Data(Addr) = SwapData**
CAS	**If (CompareData == Data(Addr)) then** **Data(Addr) = SwapData**



Power Management Enhancements

Dynamic Power Allocation(DPA)

Optimized Buffer Flush (OBFF)

Latency Tolerance Reporting (LTR)

Dynamic Power Allocation

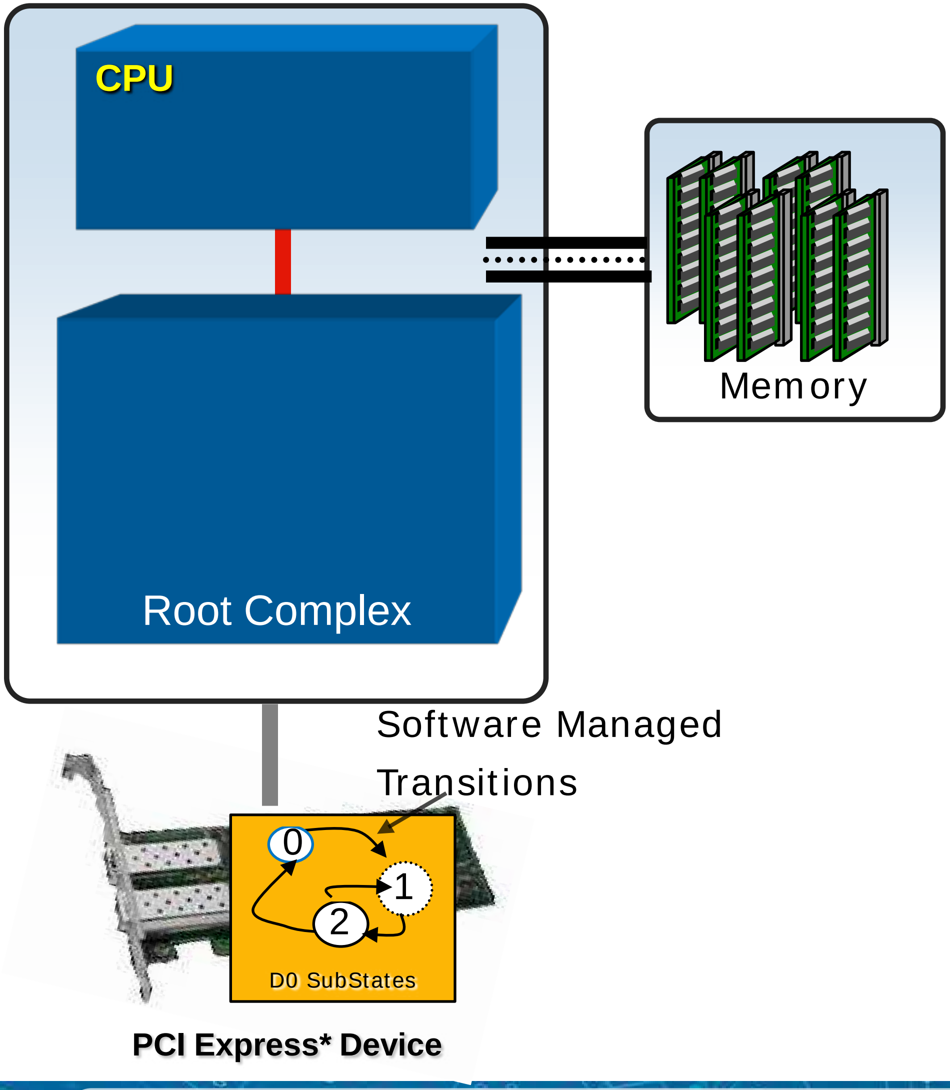
Background

- **PCIe 1.x provided standard Device & Link-level Power Management**
- **PCIe 2.0 adds mechanisms for dynamic scaling of Link width/speed**
- **No architected mechanism for dynamic control of device thermal/power budgets**

Problem Statement

- **Devices are increasingly higher consumers of system power & thermal budget**
  - Emerging 300W Add-In Cards
- **New Customer & Regulatory Operating Requirements**
  - On-going Industry wide efforts e.g. ENERGY STAR* Compliance
  - Battery Life/Enclosure Power Management
    - Mobile, Servers & Embedded Platforms

Dynamic Power Allocation (DPA)

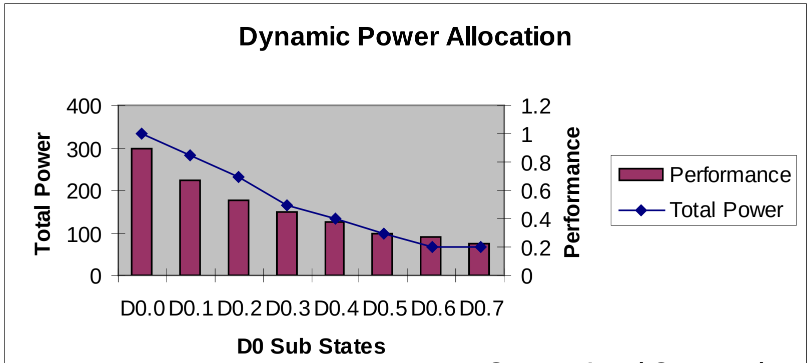


DPA Capability

- **Extend Existing PCI Device PM to provide Active (D0) substates**
  - Up to 32 substates supported
- **Dynamic Control of D0 Active Substates**

Benefits

- **Platform Cost Reduction**
  - Pwr/Thermal Management
- **Platform Optimizations**
  - Battery Life (Mobile)/Power(Servers)



Source: Intel Corporation

Enables New Platform Level Flexibility in Power/Thermal Resource Management

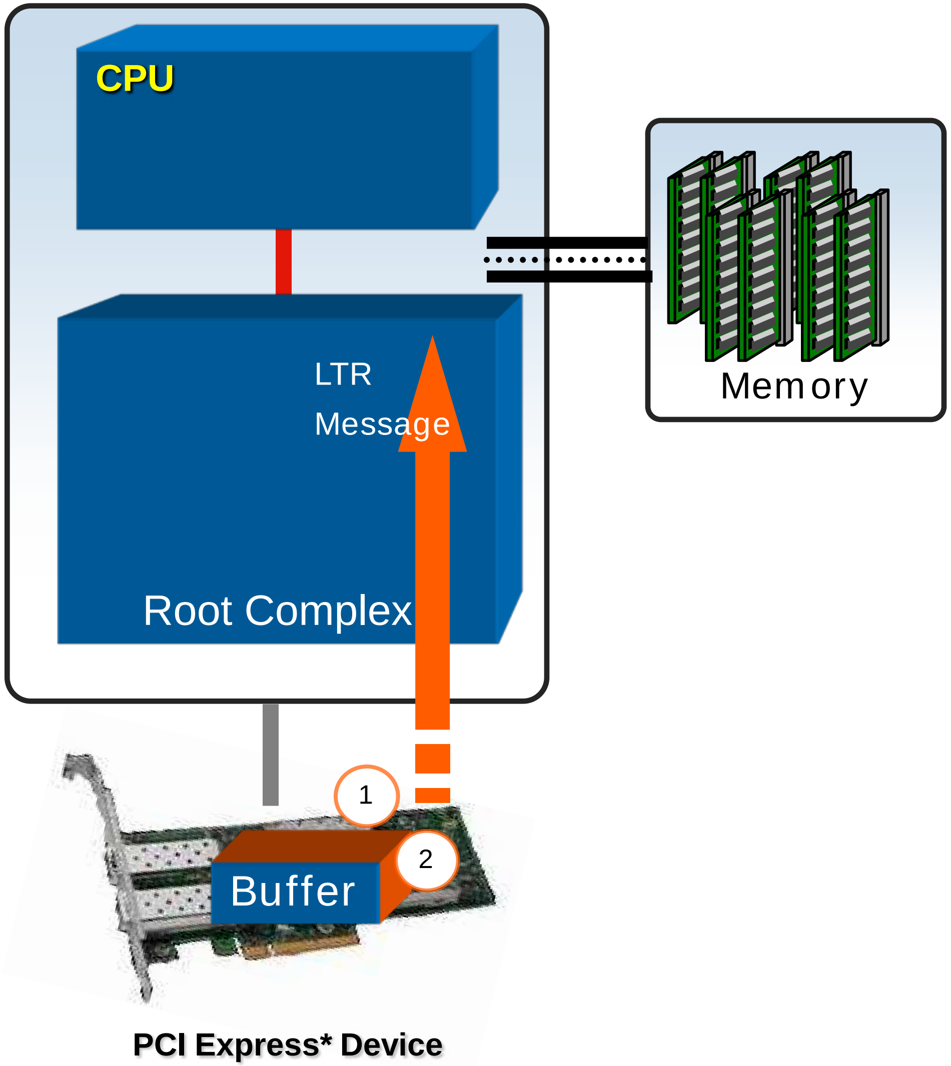
Latency Tolerance Reporting

Problem: Current Platforms PM policies guesstimate when devices are idle (e.g. w/inactivity timers)

- **Guessing wrong can cause performance issues, or even HW failures**
- **Worst case: PM disabled to allow functionality at cost to power**
- **Even best case not good – reluctance to power down leaves some PM opportunities on the table**
  - Tough balancing act between performance / functionality and power

Wanted: Mechanism for platform to tune PM based on actual device service requirements

Latency Tolerance Reporting (LTR)



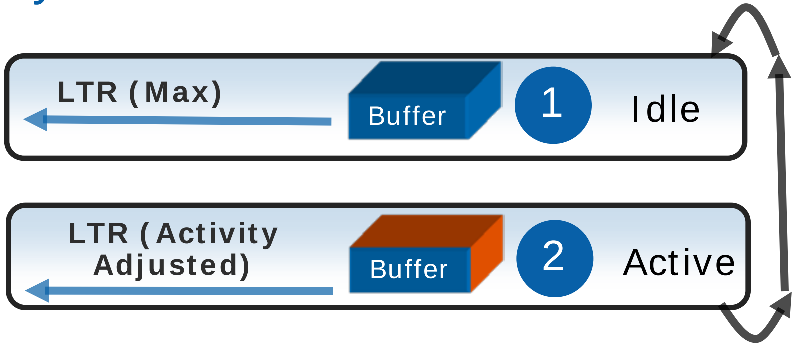
LTR Mechanism

- **PCIe Message sent by Endpoint with tolerable latency**
  - Capability to report both snooped & non-snooped values
  - “Terminate at Receiver” routing, MFD & Switch send aggregated message

Benefits

- **Provides Device Benefit: Dynamically tune platform PM state as a function of Device activity level**
- **Platform benefit: Enables greater power savings without impact to performance/ functionality**

Dynamic LTR

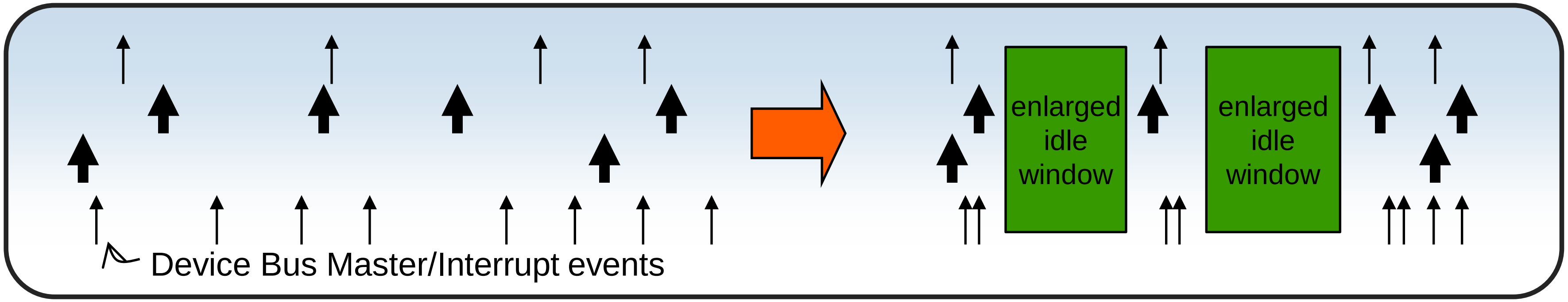


LTR enables dynamic power vs. performance tradeoffs at minimal cost impact

Optimized Buffer Flush/Fill

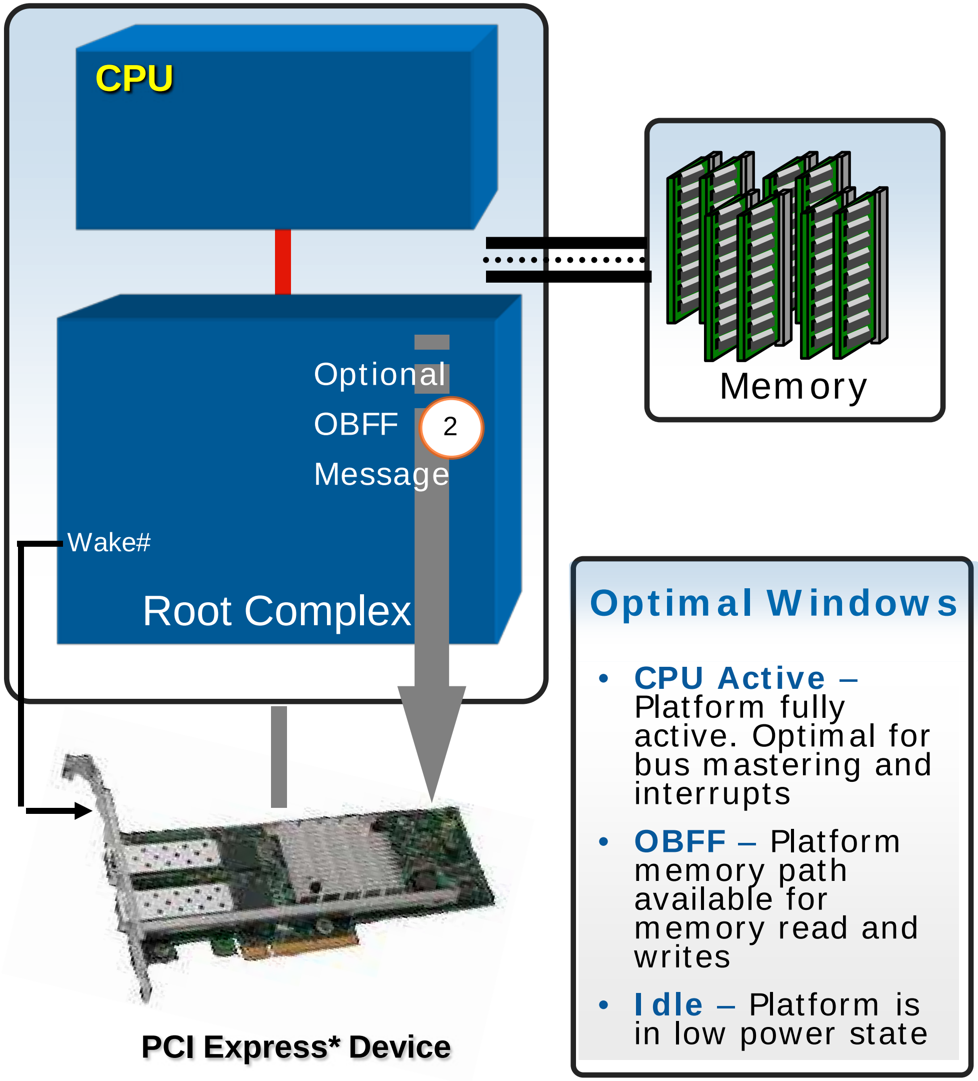
Problem: Devices do not know power state of central resources

- **"Asynchronous" device activity prevents optimal power management of memory, CPU, RC internals by idle window fragmentation**
- **Premise: If devices knew when to talk, most could easily optimize their Request patterns**
  - Result: System would stay in lower power states for longer periods of time with no impact on overall performance
- **Optimized Buffer Flush/Fill (OBFF) - a mechanism for broadcasting PM hint to device**



Wanted: Mechanism for <u>Align</u> Device Activity with Platform PM events

Optimized Buffer Flush/Fill (OBFF)



OBFF

- **Notify all Endpoints of optimal windows with minimal power impact**

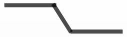
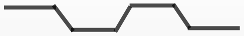
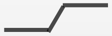
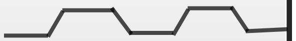
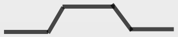
Solution1: When possible, use WAKE# with new wire semantics

Solution2: WAKE# not available – Use PCIe Message

Optimal Windows

- **CPU Active** – Platform fully active. Optimal for bus mastering and interrupts
- **OBFF** – Platform memory path available for memory read and writes
- **Idle** – Platform is in low power state

WAKE# Waveforms

Transition Event	WAKE#
Idle → OBFF	
Idle → CPU Active	
OBFF/CPU Active → Idle	
OBFF → CPU Active	
CPU Active → OBFF	

Greatest Potential Improvement When Implemented by All Platform Devices

Other Protocol Enhancements

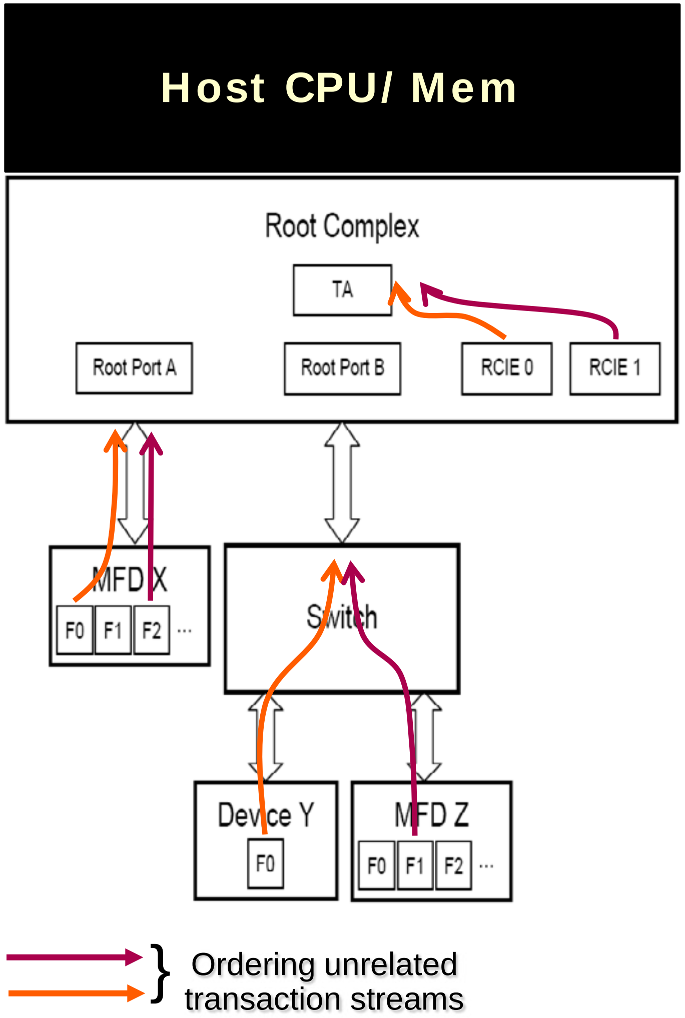
ID-based Transaction Ordering

IO Page Fault Mechanism

Resizable BAR

Multicast

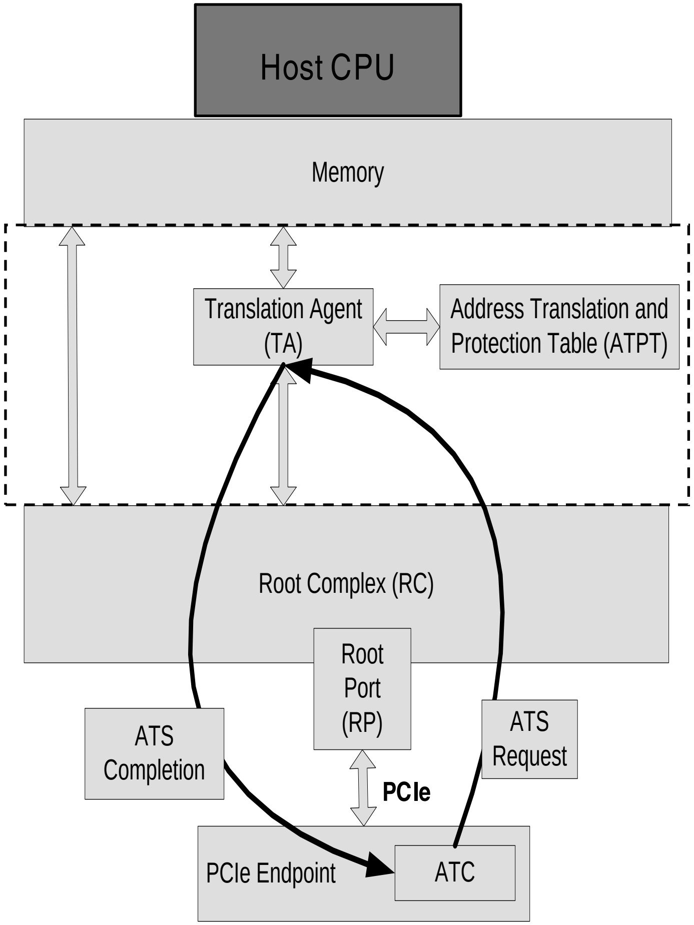
Transaction Ordering Enhancement



- Background:
  - Strong ordering == unnecessry stalls
  - Transactions from different Requestors carry different IDs
- Feature:
  - New Transaction Attribute bit to indicate ID-based ordering relaxation
    - Permission to reorder transactions between different ID streams
  - Applies to unrelated streams within:
    - MF Devices, Root Complex, Switches
- Benefits:
  - Improves latency/ power/ BW

Reduces transaction latencies in the system.

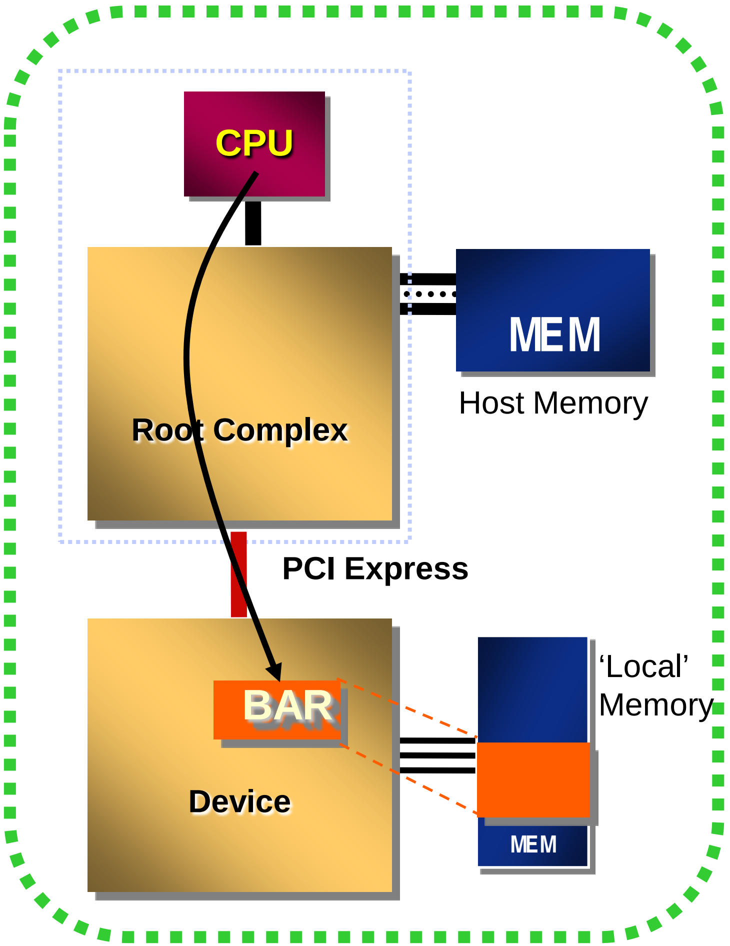
IO Page Fault Mechanism



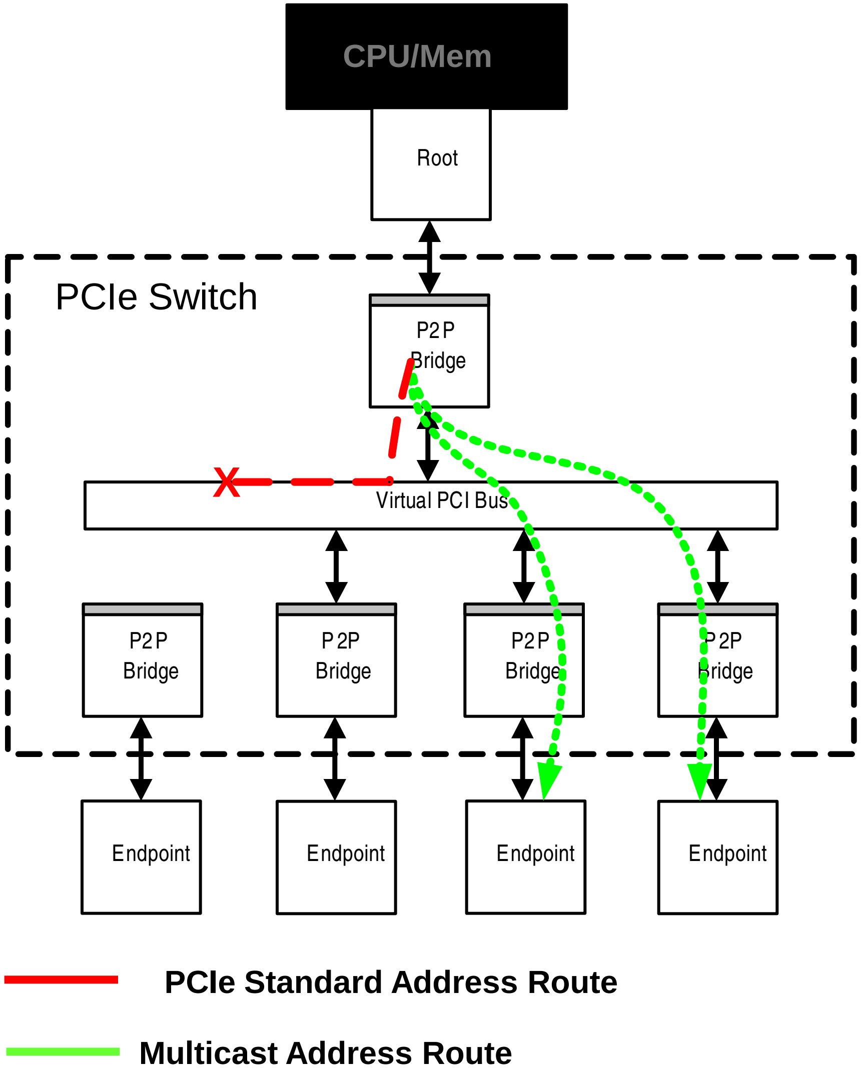
- Background:
  - Emmerging trend: Platform Virtualization
  - Increases pressure on memory resources making page “pinning” very expensive
- Feature:
  - Built upon PCIe Address Translation Services (ATS) Mechanism
  - Notify IO devices when IO page faults occur
    - Device pause/resume on page faults
    - Faulted pages requested to be made available
- Benefits:
  - OS/Hypervisor gets ability to maintain overall system performance by over-commiting memory

Critical for future IO Virtualization application scaling.

Resizable BAR & Multicast



BAR == Base Address Register – PCI mechanism for mapping device memory into sys. address space



Improved platform addres space management -- solves current problems with gfx/accel

Multicast provides perf. scaling of existing apps (e.g. multi Gfx) -- opens new usages for PCIe in embedded space

Summary

- **8.0 GT/s silicon design is challenging but achievable**
- **Double B/W: Encoding efficiency 1.25 X data rate 1.6 = 2X**
- **Next Generation PCIe Protocol Extensions Deliver**
  - Energy Efficient Performance,
  - Software Model Improvements and
  - Architecture Scalability
- **Specification Status:**
  - Rev 0.5 spec delivered to PCI SIG in Q1'09
  - Rev 0.7 targeting Sept. '09 & Rev 0.9 early Q1'10

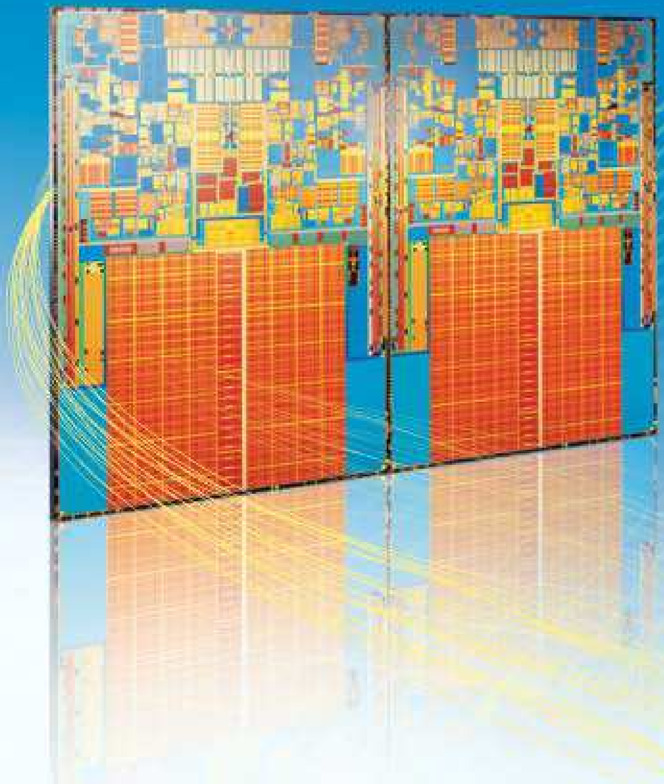
Continuous Improvement: Doubling Bandwidth & Improving Capabilities Every 3-4 Years!

Call to Action & Referrences

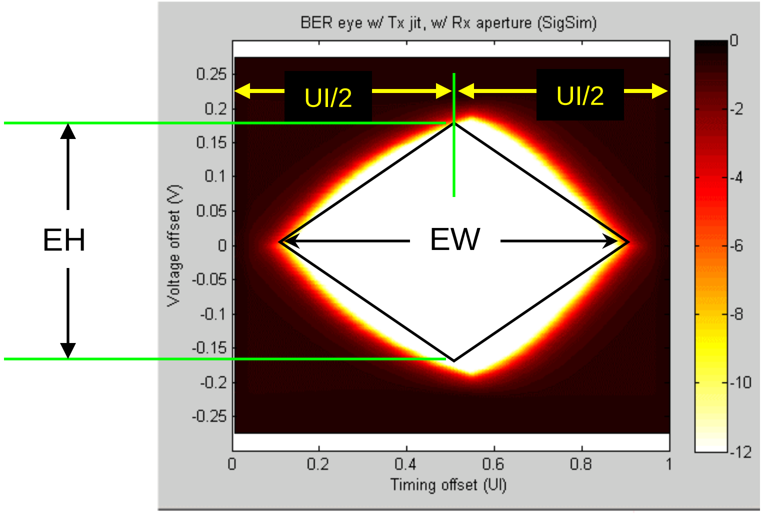
- **Contribute to the evolution of PCI Express architecture**
  - Review and provide feedback on PCIe 3.0 specs
  - Innovate and differentiate your products with PCIe 3.0 industry standard

- **Visit:**
  - ***www.pcisig.com*** for PCI Express specification updates
  - ***http://download.intel.com/technology/pciexpress/devnet/docs/PCIe3_Accelerator-Features_WP.pdf*** for white-paper on PCIe Accelerator Features

Backup



Example of a Eye As Seen At Receiver Input Latch



Eye aperture defines Tj at 10^{-12}

Eye margins reflect CDR tracking and Rx equalization

Scrambling vs. 8b/10b coding

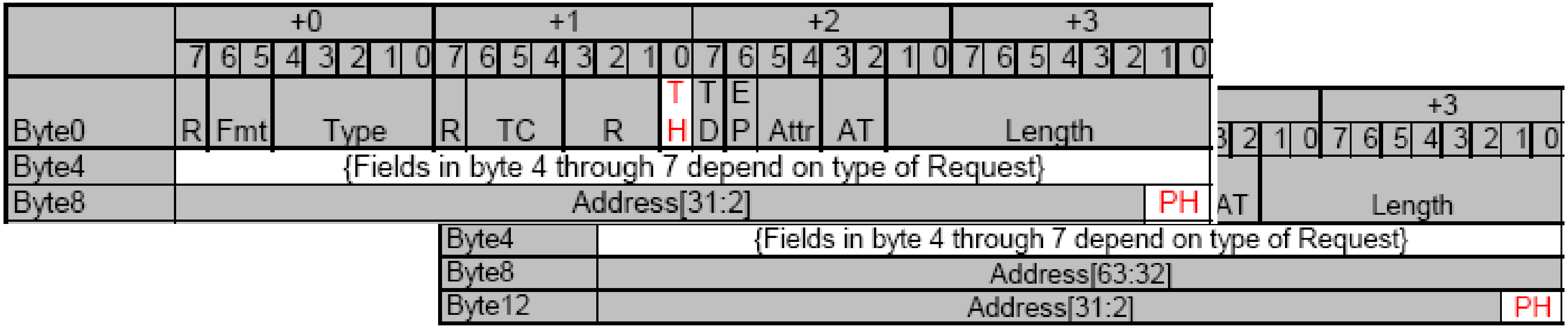
- **8GT/s uses scrambled data to improve signaling efficiency over 8b10b encoding used in 2.5GT/s and 5GT/s, yielding 2x payload data rate wrt. 5 GT/s**
- **Unlike 8b10b a maximal length PRBS generated by an LFSR does not preserve DC balance**
  - The average voltage level over a constant period of time varies slowly based on the pattern of the PRBS
  - In an AC coupled system this creates a slowly changing differential offset that that reduces eye height
- **Different PRBS polynomials have different average run lengths through their pattern and so different peak differential offsets**
  - There exists a best case PRBS23 polynomial yielding minimum DC wander of ~ 4.5 mVPP: $x^{23} + x^{21} + x^{18} + x^{15} + x^{7} + x^{2}$
- **Large number of taps tends to break up long runs of 0s or 1s (a common case)**
  - Pathological match between PRBS and data pattern have very low probability
  - Retry mechanism changes polynomial starting point to prevent pathological data pattern from failing repeatedly

Gen3 Signaling: Error Detection & Recovery

- **Framing error is detected by the physical layer**
  - The first byte of a packet is not one of the allowed sets (e.g., TLP, DLLP, LIDL)
  - Sync character is not 01 or 10
  - Same sync character not present in all lanes after deskew
  - CRC error in the length field of a TLP
  - Ordered set not one of the allowed encodings or not all lanes sending the same ordered set after deskew (if applicable)
  - 10 sync header received after 01 sync header without a marker packet in the 01 sync header OR received a marker packet in the 01 sync header and the subsequent sync header in any lane not 10
- **Any framing error requires directing LTSSM to Recovery**
  - Stop processing any received TLP/ DLLP after error until we get through Recovery
  - Block lock acquired with EIEOS
  - Scrambler reset with each EIEOS
- **Error Detection Guarantees**
  - Triple bit flip detection within each TLP/ DLLP/ IDL/ OS

TLP Processing Hints (TPH)

TPH Mechanism



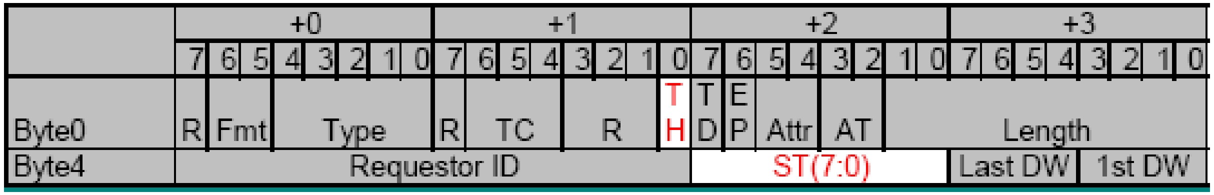
	+0			+1				+2					+3
	7	6 5	4 3 2 1 0	7	6 5 4	3 2 1	0	7	6	5 4	3 2	1 0	7 6 5 4 3 2 1 0
Byte0	R	Fmt	Type	R	TC	R	TH	TD	EP	Attr	AT	Length	
Byte4	{Fields in byte 4 through 7 depend on type of Request}												
Byte8	Address[31:2]												PH

		+3
	3 2 1 0	7 6 5 4 3 2 1 0
	AT	Length
Byte4	{Fields in byte 4 through 7 depend on type of Request}	
Byte8	Address[63:32]	
Byte12	Address[31:2]	PH

- **Mechanism to provide processing hints on per TLP basis for Requests that target Memory Space**
  - Enable system hardware (ex: Root-Complex) to optimize on a per TLP basis
  - Applicable to Memory Read/Write and Atomic Operations

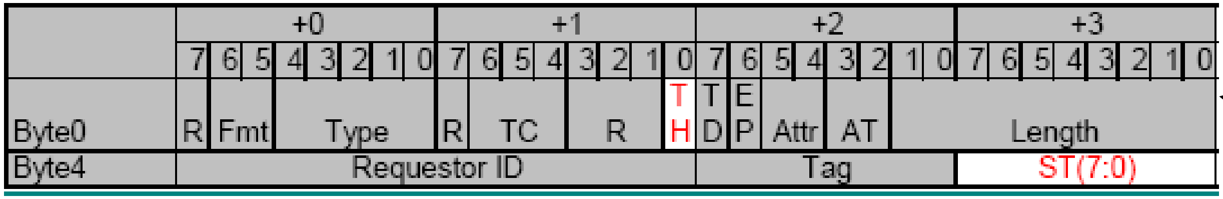
PH[1:0]	Processing Hint	Usage Model
00	Bi-directional data structure	Bi-Directional data structure
01	Requestor	D*D*
10	Target	DWHR HWDR
11	Target with Priority	DWHR (Prioritized) HWDR (Prioritized)

Steering Tag (ST)



	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte0	R	Fmt		Type					R	TC			R			T H	T D	E P	Attr		AT		Length									
Byte4	Requestor ID																ST(7:0)								Last DW				1st DW			

Memory Write TLP



	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte0	R	Fmt		Type					R	TC			R			T H	T D	E P	Attr		AT		Length									
Byte4	Requestor ID																Tag								ST(7:0)							

Memory Read or

AtomicOperation TLPs

- **ST: 8 bits defined in header to carry System specific Steering Tag values**
  - Use of Steering Tags is optional – ‘No preference’ value used to indicate no steering tag preference
  - Architected Steering Table for software to program system specific steering tag values

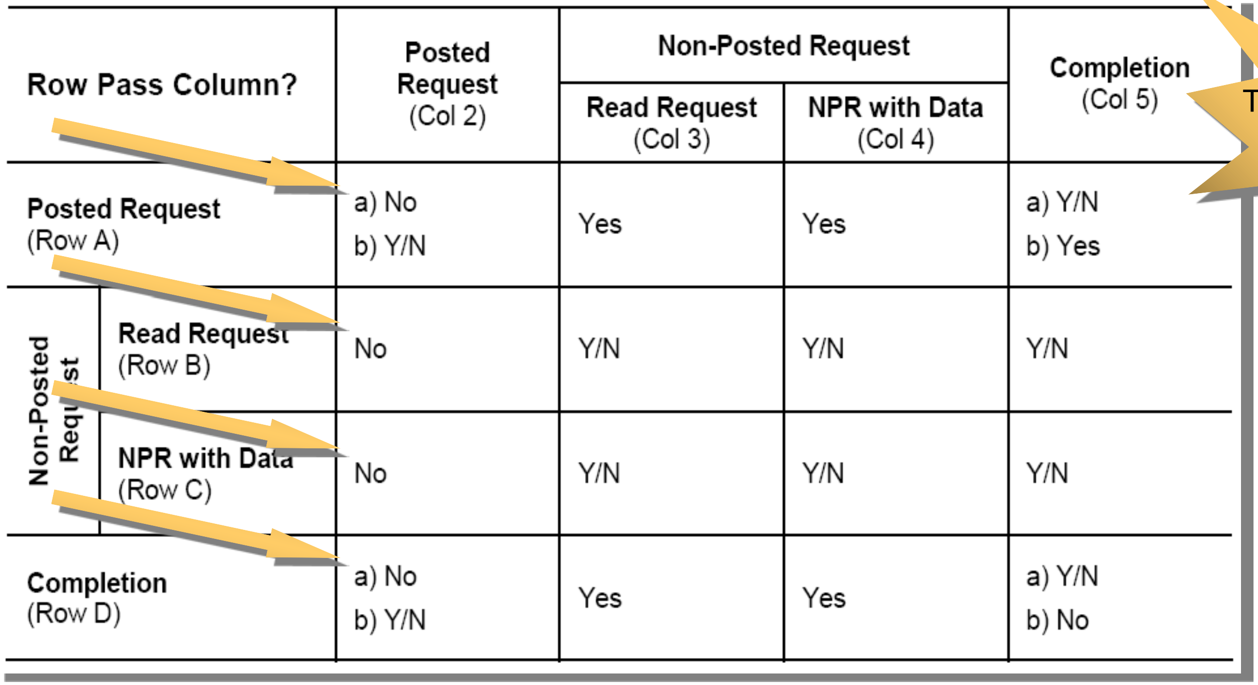
TPH Summary

- **Mechanism to make effective use of system fabric and improve system efficiency**
  - Reduce variability in access to system memory
  - Reduce memory & system interconnect BW & power consumption
- **Ecosystem Impact**
  - Software impact is under investigation - minimally may require software support to retrieve hints from system hardware
  - Endpoints take advantage only as needed → No cost if not used
  - Root Complex can make implementation tradeoffs
  - Minimal impact to Switches
- **Architected software discovery, identification, and control of capabilities**
  - RC support for processing hints
  - Endpoint enabling to issue hints

ID-Based Ordering (IDO)

Review: PCI e Ordering Rules

“No” entries caused by Producer/ Consumer restrictions



Row Pass Column?		Posted Request (Col 2)	Non-Posted Request		Completion (Col 5)
			Read Request (Col 3)	NPR with Data (Col 4)	
Posted Request (Row A)		a) No b) Y/N	Yes	Yes	a) Y/N b) Yes
Non-Posted Request	Read Request (Row B)	No	Y/N	Y/N	Y/N
	NPR with Data (Row C)	No	Y/N	Y/N	Y/N
Completion (Row D)		a) No b) Y/N	Yes	Yes	a) Y/N b) No

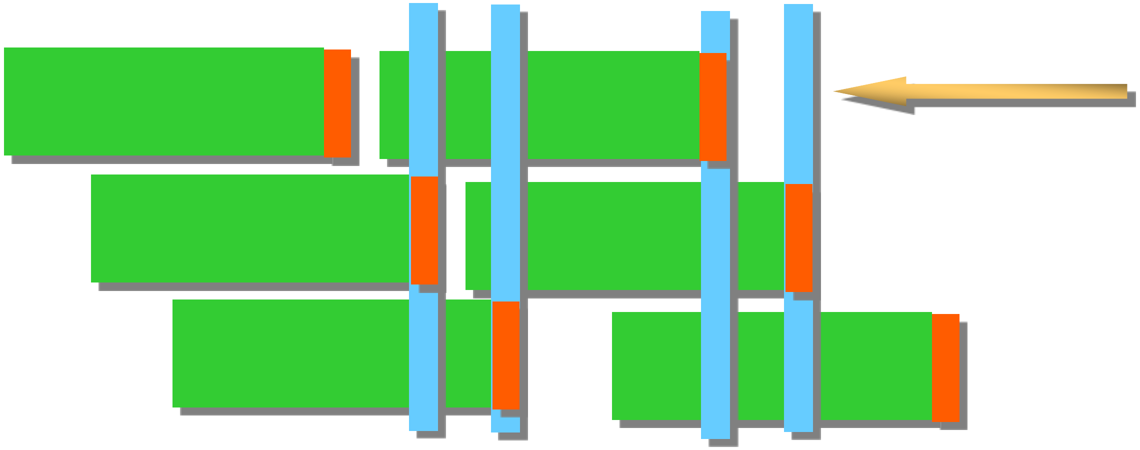
Table is based on new 2.0 errata!

“Yes” entries are required for deadlock avoidance

- **Maximum theoretical flexibility: All entries are “Y/ N”**
- **Traditional Relaxed Ordering (RO) enables A2 & D2 “Y/ N” cases**
  - AtomicOps ECR defines an RO-enabled C2 “Y/ N” case
- **I D-Based Ordering (I DO) enables A2, B2, C2, & D2 “Y/ N” cases**

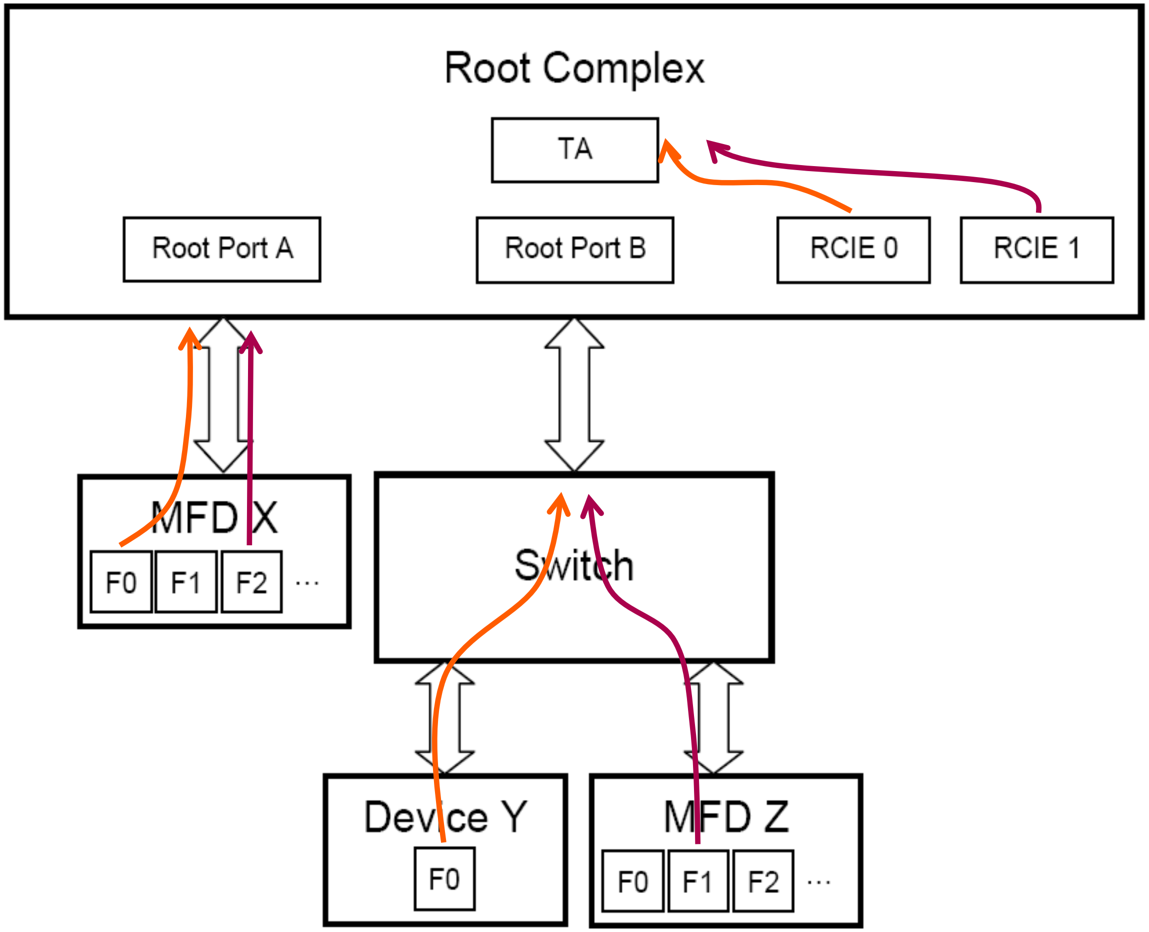
Motivation

- **RO works well for single-stream models where a data buffer is written once, consumed, and then recycled**
  - Not OK for buffers that will be written more than once because writes are not guaranteed to complete in order issued
  - Does not take advantage of the fact that ordering doesn't need to be enforced between unrelated streams
- **Conventional Ordering (CO) can cause significant stalls**
  - Observed stalls in the 10's to 100's of ns are seen
  - Worst case behavior may see such stalls repeatedly for a Request stream
- **Consider case of NIC or disk controller with multiple streams of writes:**



Each CO Flag Write serializes & adds latency to traffic from unrelated streams

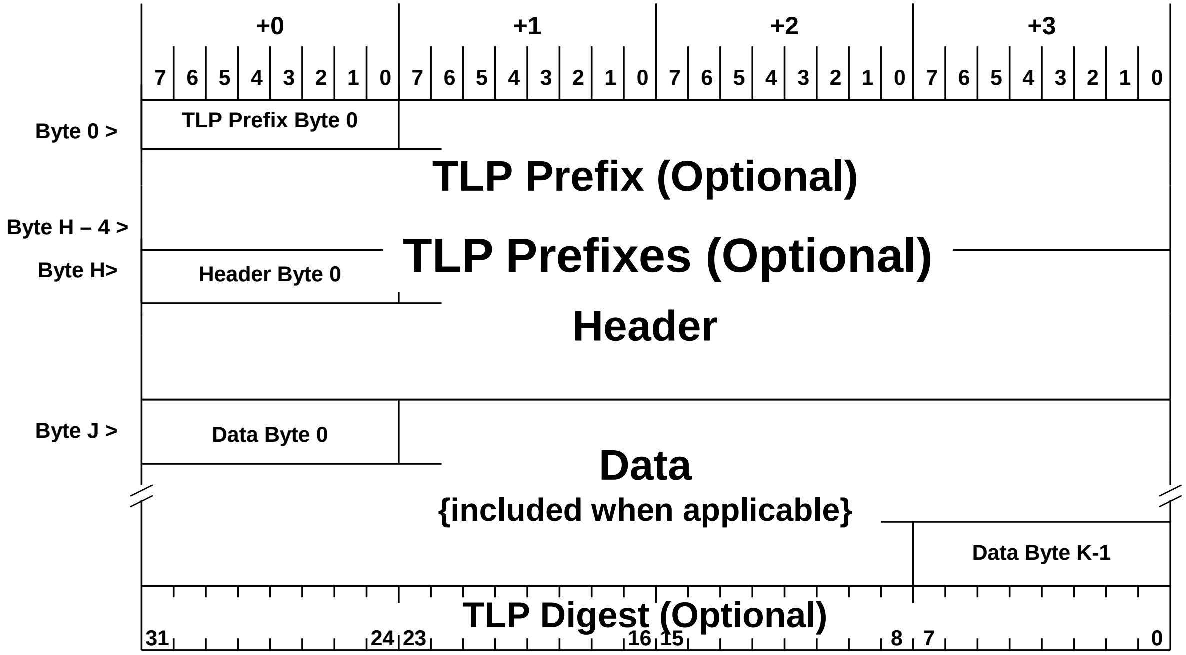
IDO: Perf Optimizations for Unrelated TLP Streams



- **TLP Stream: a set of TLPs that all have the same originator**
- **Optimizations possible for unrelated TLP Streams, notably with:**
  - Multi-Function device (MFD) / Root Port Direct Connect
  - Switched Environments
  - Multiple RC Integrated Endpoints (RCIEs)
- **IDO permits passing between TLPs in different streams**
- **Particularly beneficial when a Translation Agent (TA) stalls TLP streams temporarily**

TLP Prefix

Motivation



- **Emerging usage models require increase in header size to carry new information**
  - Example: Multi-Root IOV, Extended TPH
- **TLP Prefix mechanism extends the header sizes by adding DWORDs to the front of headers**

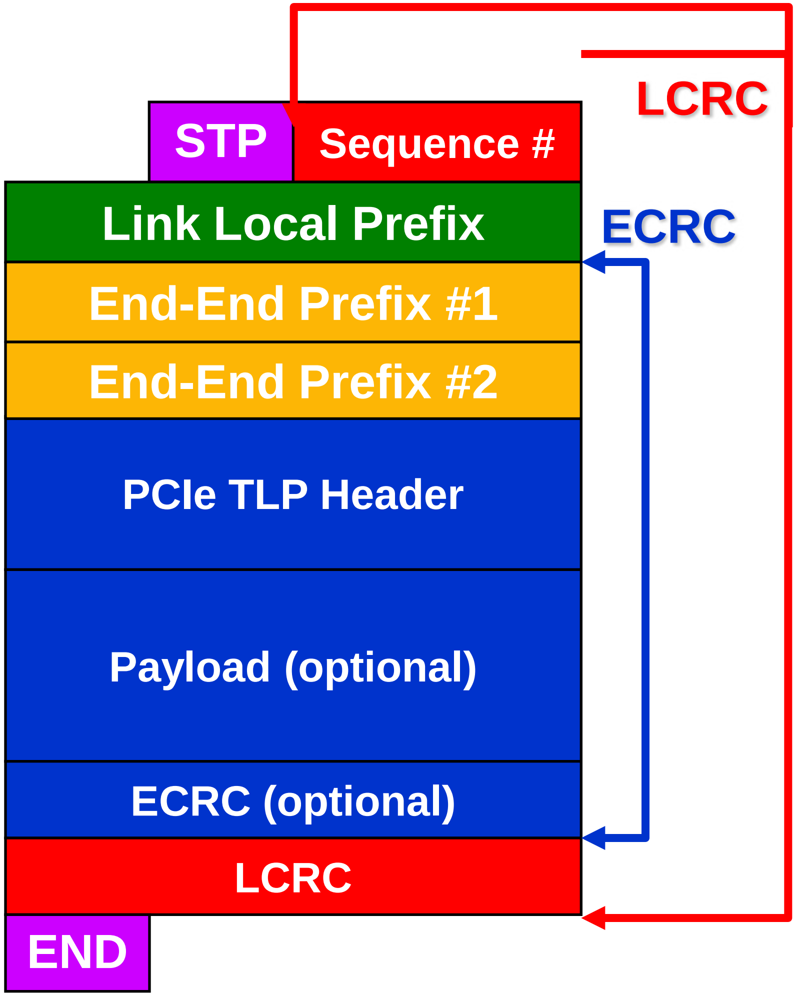
Prefix Encoding

	+0			+1	+2	+3
	7 6 5	4	3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0	100	0	Type	Local Prefix Contents		

	+0			+1	+2	+3
	7 6 5	4	3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0	100	1	Type	End-End Prefix Contents		

- **Base TLP Prefix Size – 1 DW**
  - Appended to TLP headers
- **TLP Prefixes can stacked or repeated**
  - More than one TLP Prefix supported
- **Link Local – Where routing elements may process the TLP for routing or other purposes.**
  - Only usable when both ends understand and are enabled to handle link local TLP Prefix
  - ECRC not applicable
- **End-End TLP Prefix**
  - Requires support between the Requester, Completer and routing elements
  - End-End TLP Prefix not required to but is permitted to be protected by ECRC
    - If underlying Base TLP is protected by ECRC then End-End TLP Prefix is also protected by ECRC
  - Upper bound of 4DWORDs (16 Bytes) for End-End TLP Prefix
- **Fmt field grows to 3 bits**
  - New error behavior defined
  - Undefined Fmt and/or Type values results in Malformed TLP
  - “Extended Fmt Field Supported” capability bit indicates support for 3 bit Fmt
    - Support is recommended for all components (independent of Prefix support)

Stacked Prefix Example:



- **Link Local is first**
  - Starts at 0
  - Type$_{L1}$
- **End-End #1 follows Link Local**
  - Starts at 4
  - Type$_{E1}$
- **End-End #2 follows End-End #1**
  - Starts at 8
  - Type$_{E2}$
- **PCIe Header follows End-End #2**
  - Starts at 12

- **Switch routes using Link Local and PCIe Header**
  - … and possibly additional Link Local DWORDs
    - if more extension bits needed
  - Malformed TLP if don't understand

- **Switch forwards End-End Prefixes unaltered**
  - End-End Prefixes do not affect routing
  - Up to 4 DWORDs (16 Bytes) of End-End Prefix

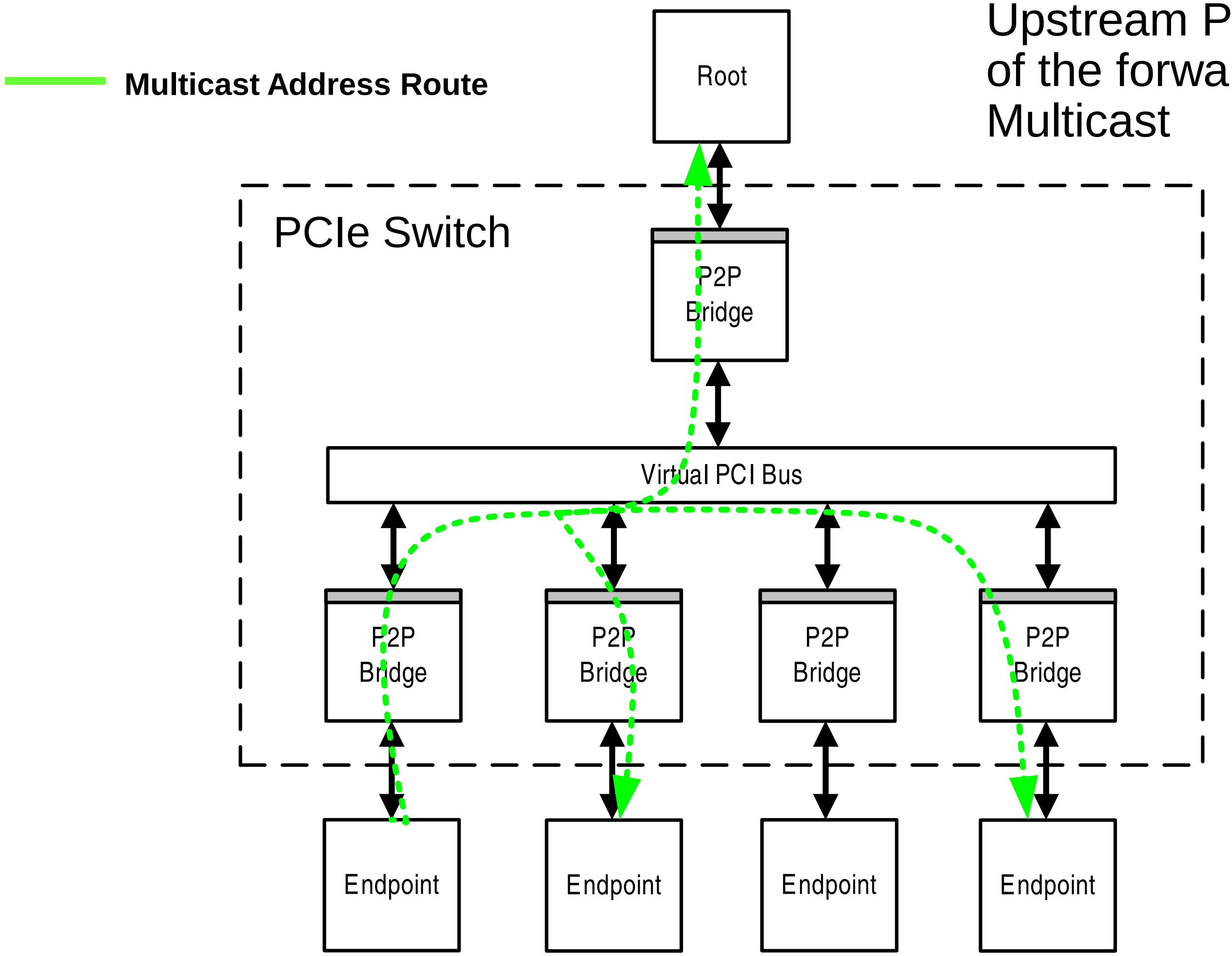
- **End-End Prefixes are optional**
  - Different End-End Prefixes sequence are unordered
    - affects ECRC but does not affect meaning
  - Repeated End-End Prefix sequence must be ordered
    - e.g. 1st Extended TPH vs. 2nd Extended TPH attribute
    - meaning of this is defined by each End-End Prefix

Multicast

Multicast Motivation & Mechanism Basics

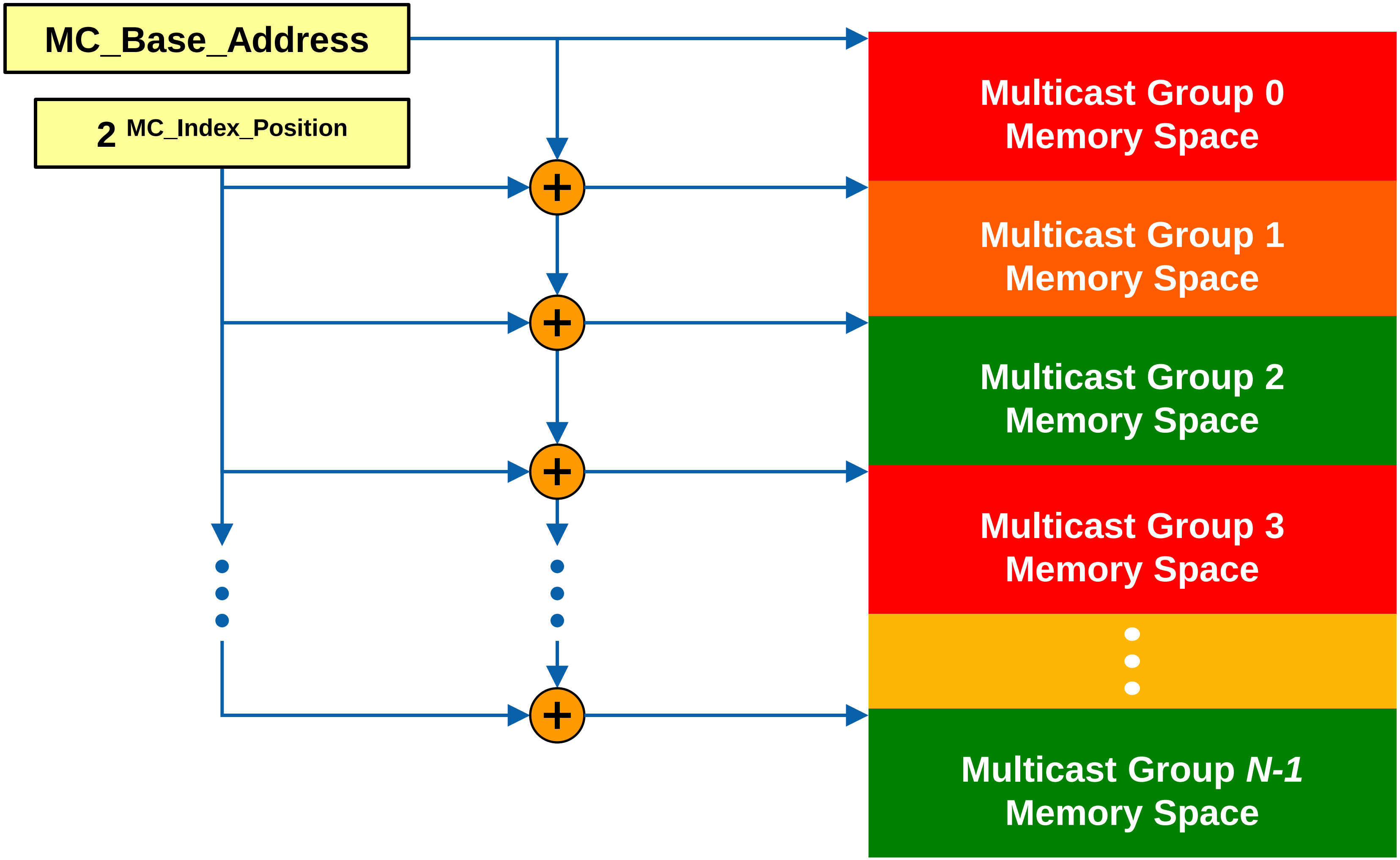
- **Several key applications benefit from Multicast**
  - Communications backplane (e.g. route table updates, support of IP Multicast)
  - Storage (e.g., mirroring, RAID)
  - Multi-headed graphics
- **PCIe architecture extended to support address-based Multicast**
  - New Multicast BAR to define Multicast address space
  - New Multicast Capability structure to configure routing elements and Endpoints for Multicast address decode and routing
  - New Multicast Overlay mechanism in Egress Ports allow Endpoints to receive Multicast TLPs without requiring Endpoint Multicast Capability structure
- **Supports only Posted, address-routed transactions (e.g., Memory Writes)**
  - Supports both RCs and EPs as both targets and initiators
  - Compatible with systems employing Address Translation Services (ATS) and Access Control Services (ACS)
  - Multicast capability permitted at any point in a PCIe hierarchy

Multicast Example



- Address route Upstream– Upstream Port must be part of the forwarding Ports for Multicast

Multicast Memory Space



MC_Base_Address

$2^{\text{MC\_Index\_Position}}$

Multicast Group 0 Memory Space

Multicast Group 1 Memory Space

Multicast Group 2 Memory Space

Multicast Group 3 Memory Space

Multicast Group *N-1* Memory Space