
PCI
SIG™

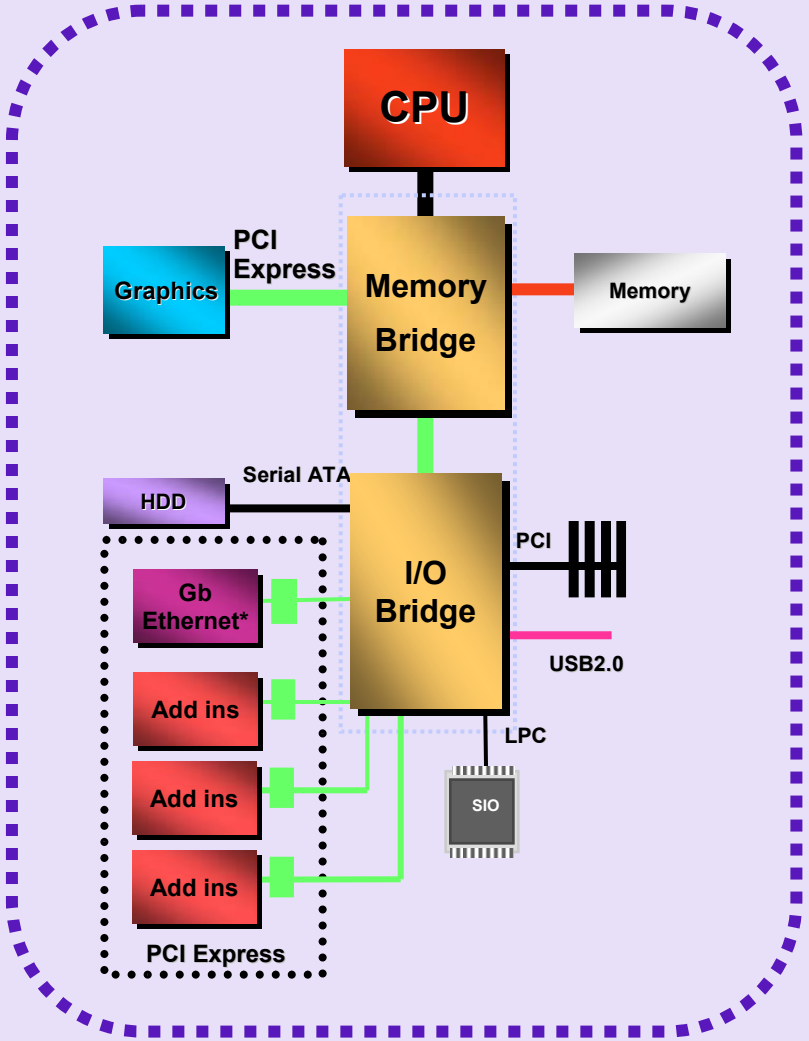
PCI Express™ Advanced Hardware Topics & Specification Updates

Jasmin Ajanovic – Chair, PCI Express Protocol Workgroup

Carl Jackson – Member, PCI Express Protocol Workgroup

PCI Express* - Selected Features Summary

- Physical Interface:
  - ✓ Point-to-point full-duplex
- Protocol:
  - ✓ Load Store architecture
  - ✓ Fully packetized split-transaction
  - ✓ Credit-based flow Control
- PCI Compatibility:
  - ✓ Configuration model and PCI Software Driver model
- Advanced Capabilities:
  - ✓ Data Integrity
  - ✓ Error Reporting
  - ✓ Hot-Plug
  - ✓ Power Management
  - ✓ Virtual Channels



Agenda

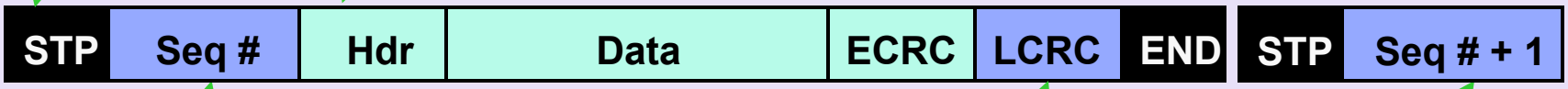
- Link Data Integrity & End-To-End CRC
- Error Signaling & Logging
- Flow Control & Buffering Optimizations
- Hot-Plug
- Power Management
- Specification Update
- Summary & Call to Action

Error Coverage

Unambiguous Framing with 8b/10b

Explicit error forwarding mechanism

End-to-end 32b CRC coverage



False/missed start correction

Transaction Layer Packet protected with 32b CRC

False/missed termination correction

Both Local and End-to-End Robust Error Coverage

Data Integrity Support

- Requirements for Robust Data Integrity
- Data Link Layer Mechanisms (Link/local):
  - ✓ TLPs protected using 32bit CRC
  - ✓ DLLPs protected using 16bit CRC
  - ✓ TLP error recovery through Data Link-level retry
  - ✓ Supplemental coverage through 8b/10b
  - ✓ Loss of packets detected using Sequence Numbers
- Transaction Layer Mechanisms (End-to-End):
  - ✓ Optional coverage using 32bit CRC
  - ✓ Data Poisoning capability

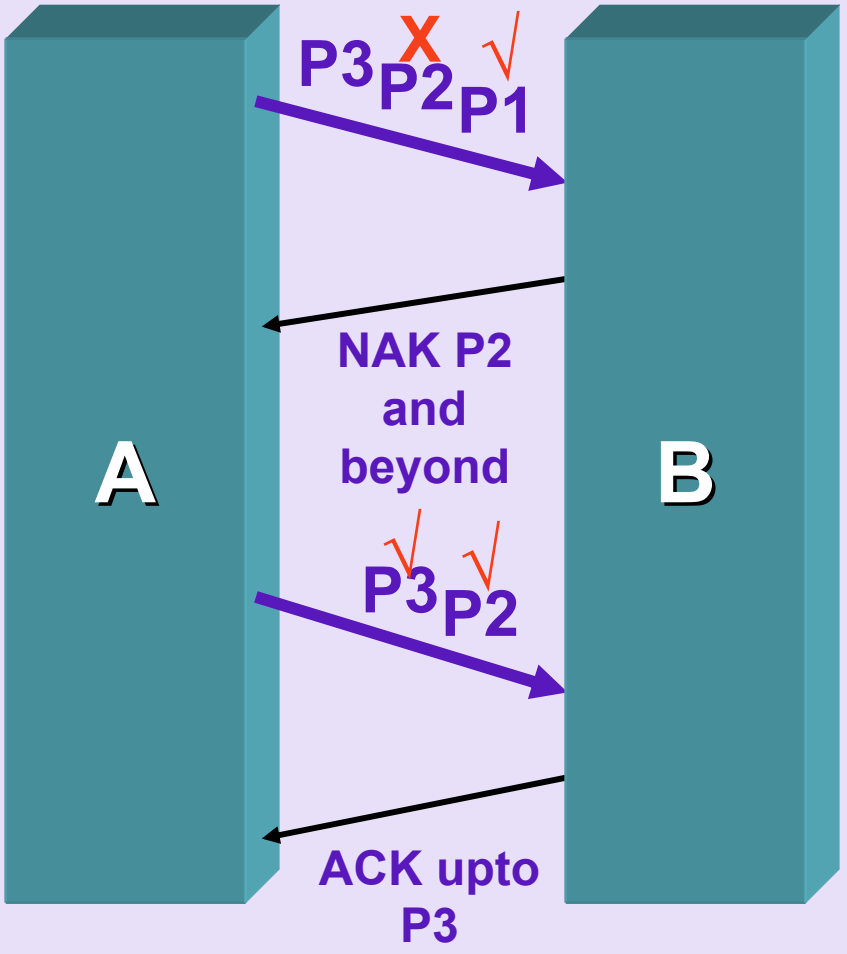
Robust Data Integrity Allows for Signaling Frequency Headroom

Link Data Integrity for TLPs

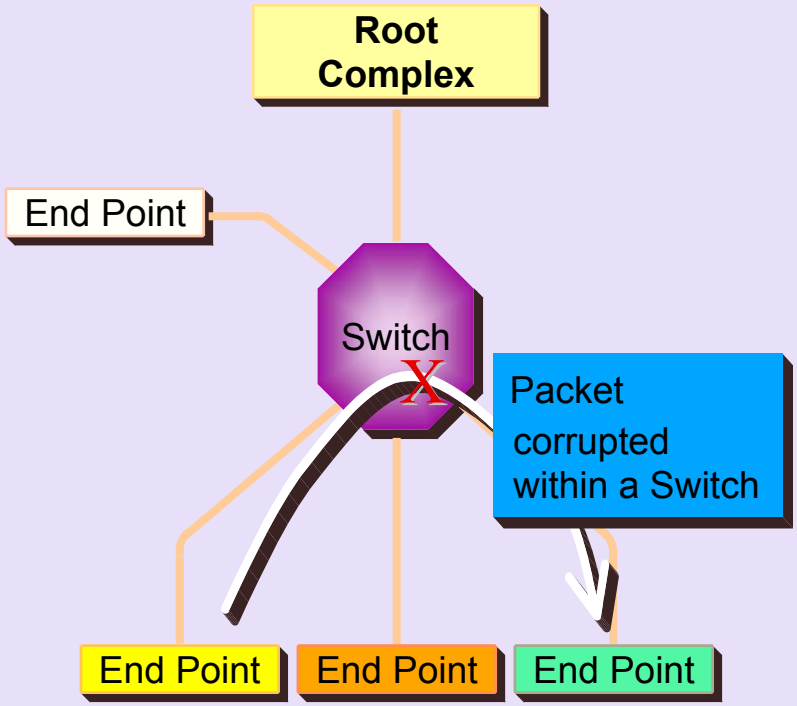
- Covers integrity of Link between two directly attached PCI Express devices across one Link
- Transmit side :
  - ✓ Applies 32bit CRC and Sequence # to Transaction Layer Packets
  - ✓ Buffers TLPs to allow retransmission
- Receive side:
  - ✓ Validates received TLPs by:
    - Checking the CRC code
    - Checking the Packet Sequence Number
    - Checking Phy Layer status for 8b/10b errors and framing errors
  - ✓ In the case of error:
    - Affected packet and following packets are discarded
    - NACK DLLP is sent to Transmitter to request retransmission
- Transmitter time-out causes re-transmission if TLP completely lost

Link Data Integrity – Retry Example

1. Three TLPs sent from ***A*** to ***B***
2. Packet **P2** corrupted
3. ***B*** detects corruption and issues Nak DLLP
4. ***A*** resends Packet **P2** and subsequent Packet
5. ***B*** acknowledges successful receipt of Packets



End-to-End Data Integrity - ECRC



- Component internal errors are critical
  - ✓ Header errors → TLP misrouting
  - ✓ Data corruption → application and system failure
- End-to-end data integrity using ECRC
  - ✓ Protecting from system-wide errors
  - ✓ Enabling upper layers error recovery
- ECRC basics:
  - ✓ Optional Capability – additional 32bit field (part of TLP)
  - ✓ Generated by the source component – applies to all invariant TLP fields
  - ✓ Switches must pass ECRC unchanged
  - ✓ Checked in the destination component – resulting behavior is device specific

Agenda

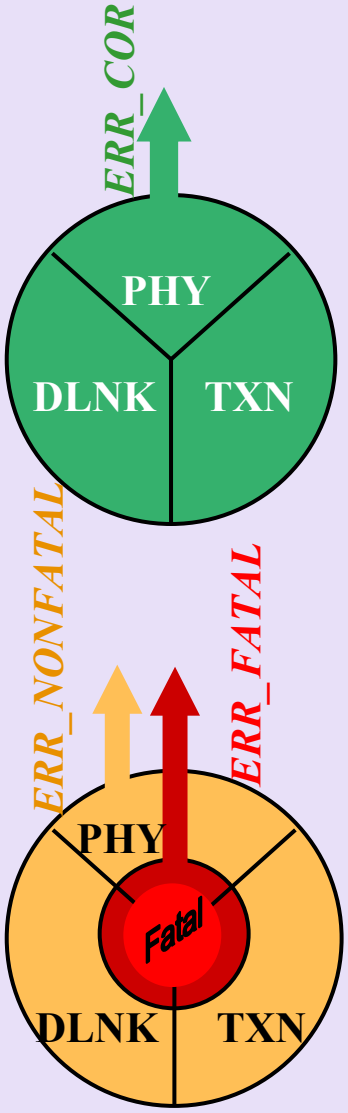
- Link Data Integrity & End-To-End CRC
- Error Signaling & Logging
- Flow Control & Buffering Optimizations
- Hot-Plug
- Power Management
- Specification Update
- Summary & Call to Action

Error Signaling and Logging

- Consistent mechanism for managing PCI Express errors
  - ✓Signaling using Completion Status
  - ✓Signaling using Error Messages
  - ✓Control and Logging using configuration registers
  - ✓Baseline and Advanced Error Reporting
- Error Classification and Mapping
  - ✓Correctable, Uncorrectable (Fatal, Non-Fatal)
  - ✓Mapping of Physical, Data Link and Transaction Layer Errors
- Advanced Error Reporting expands management / reporting capability and number of managed events

Robust Data Integrity Complemented With Standardized Error Signaling/logging

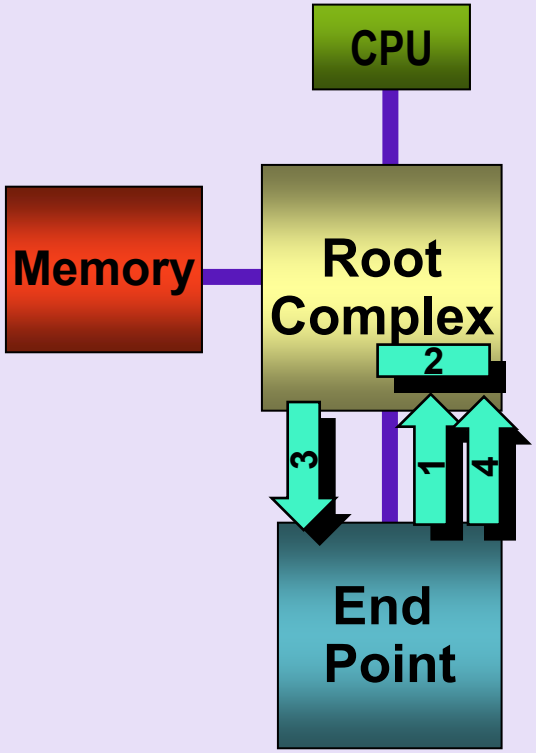
Error Classification



Programmable Severity with Advanced Error Reporting

- Correctable Errors (Msg: ERR_COR)
  - ✓ Hardware corrects the error without software impact beyond performance
  - ✓ Logging permits port integrity profiling
  - ✓ Examples: Invalid Symbol in packet; CRC error
- **Uncorrectable – Non-Fatal (Msg: ERR_NONFATAL)**
  - ✓ **Hardware cannot correct the error**
  - ✓ **Transaction lost; impacts the software**
  - ✓ **Port otherwise fully functional**
  - ✓ **Examples: Unsupported Request; Completer Abort**
- Uncorrectable – Fatal (Msg: ERR_FATAL)
  - ✓ Hardware cannot correct the error
  - ✓ Port reliability compromised
  - ✓ Likely component/hierarchy reset required
  - ✓ Example: Malformed Packet

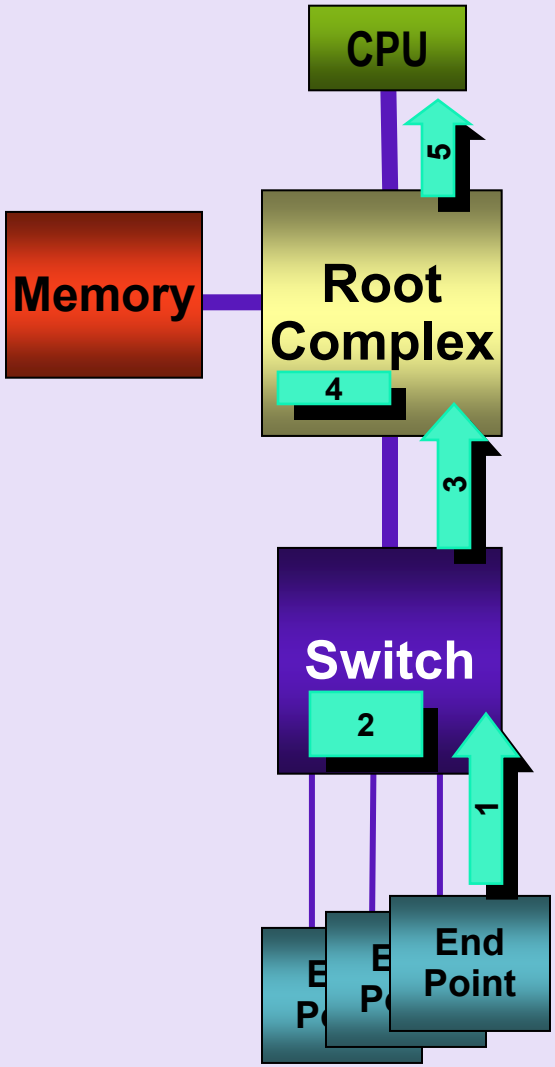
Correctable Error Handling



1. End Point issues TLP to Root Complex (Seq# = N)
2. Root Complex detects LCRC error
3. Root Complex records correctable error and responds to TLP with a Nak DLLP identifying packet N (Link-level retry)
4. End Point reattempts all packets starting with TLP N.

Retry to REPLAY_NUM limit. If *still* not corrected, the Endpoint records an error and retrains the Link.

Uncorrectable Error Handling



1. Malformed TLP Detected at Switch

2. Switch records:
   -Uncorrectable Error *(Device Status)*
   -Malformed TLP *(Uncorrectable Error Status)*
   -First Error Ptr into *Uncorrectable Error Status (Advanced Error Capability and Control)*
   -TLP Header Encoding *(Header Log)*

3. Switch issues ERR_FATAL message to Root

4. Root Complex records reception of error Message and RequesterID of message

5. Root Complex triggers MSI or system-specific alert

Error Reporting/Control Registers

	Base Reporting Registers	Advanced Error Reporting Registers (in addition to Base)
Error Type Detected	**Device Status**	**Uncorrectable Error Status** **Correctable Error Status** **Header Log Registers** **Advanced Error Capability and Control Register** ***Root Error Status****
Error Mask	**Device Control**	**Uncorrectable Error Mask** **Correctable Error Mask**
System Alert Method	***Root Control****	***Root Error Command****
Severity	**N/A**	**Uncorrectable Error Severity**
Error Source		***Error Source ID Register****

*** Root Complex registers**

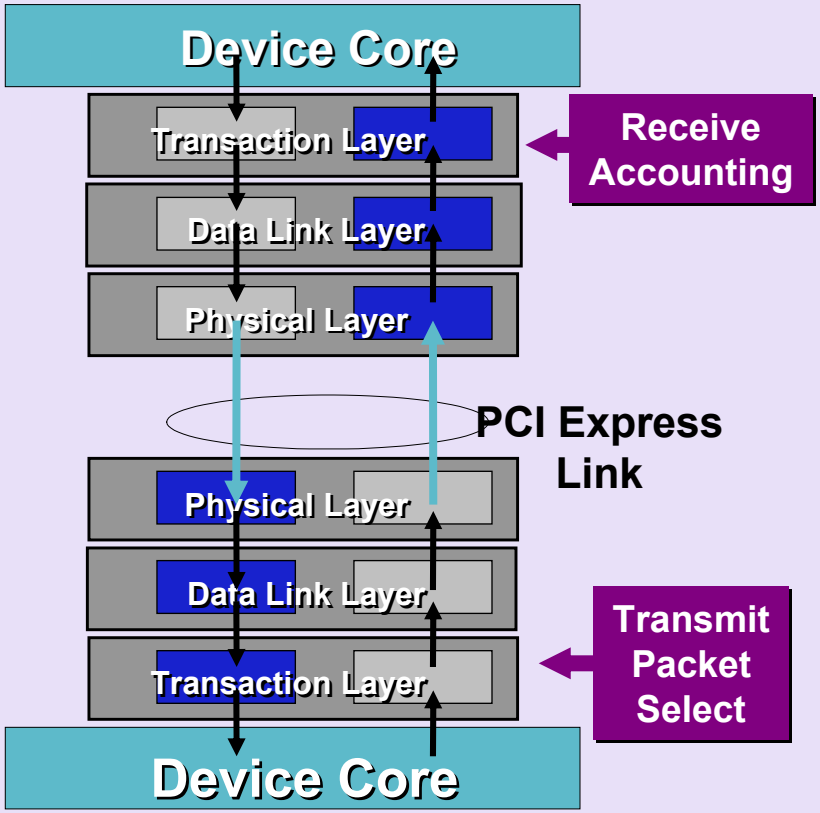
Advanced Error Reporting Provides Severity-level Detail

Agenda

- Link Data Integrity & End-To-End CRC
- Error Signaling & Logging
- Flow Control & Buffering Optimizations
- Hot-Plug
- Power Management
- Specification Update
- Summary & Call to Action

Flow Control

- Transaction to Transaction Layer across one Link
- Prevents overflow of receiver buffers
- Enables compliance with ordering rules
- Credit-based scheme
  - ✓ Transmitter throttles according to its supply of credits
  - ✓ Requesters can not use FC to throttle completions



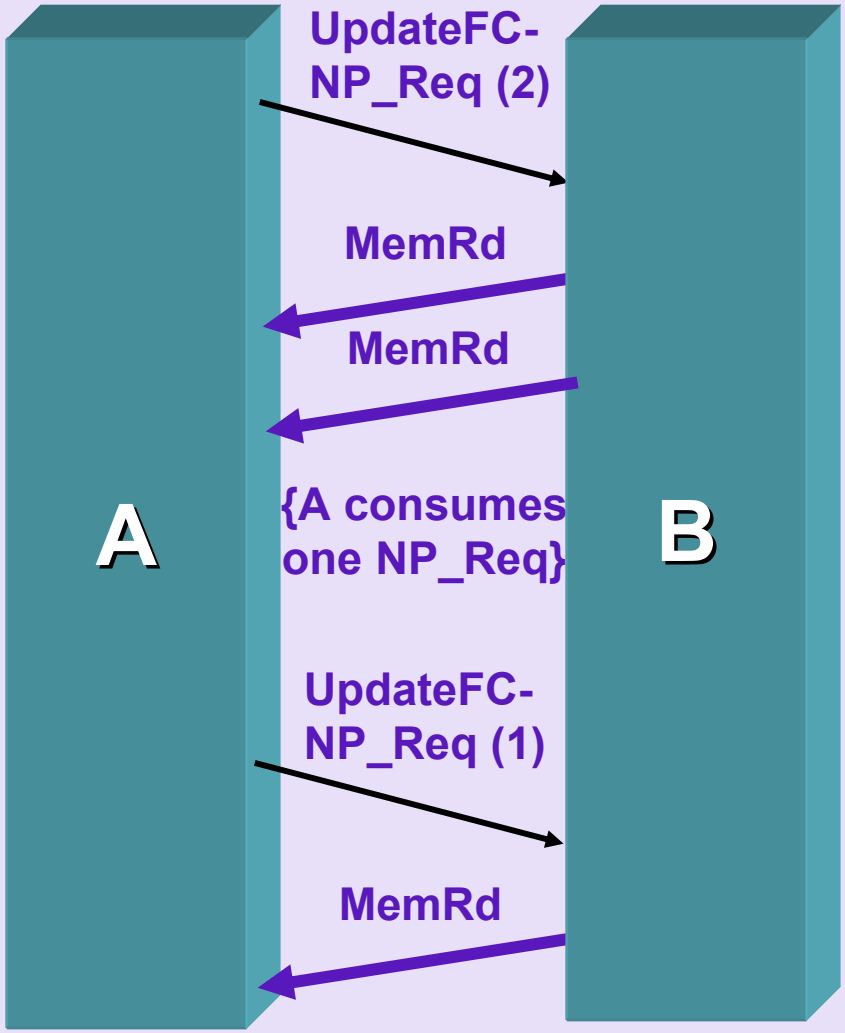
Flow control and ordering rules enable deadlock-free operations

Flow Control (cont.)

- Handled by the Transaction Layer in cooperation with the Data Link Layer
  - ✓ DLLPs used to exchange FC information
- FC information covers separately:
  - ✓ Posted and Non-Posted Request Queues, Completion Queues
  - ✓ Separate Header vs. Data Queues
- FC is orthogonal to the data integrity mechanisms
- FC Ack does not imply Request completion

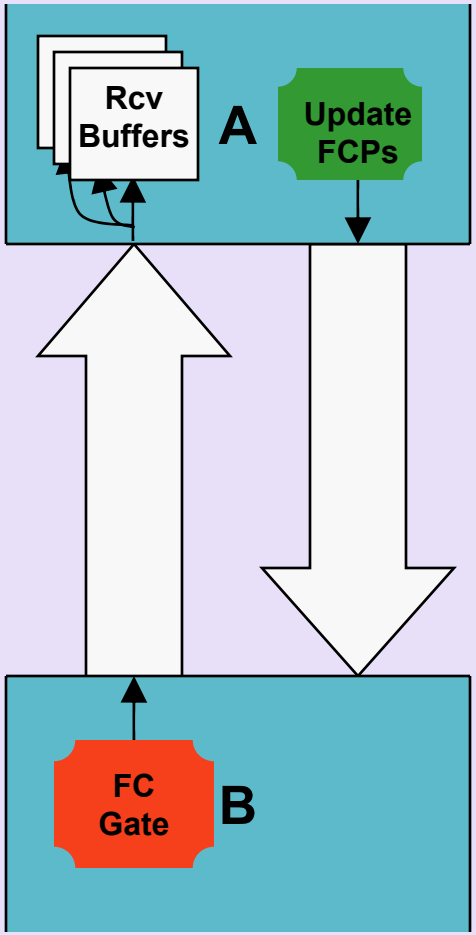
Flow Control Example

1. A advertises buffer space for two Non-Posted Requests
2. B sends two Memory Read Requests
3. A consumes one of the Non-Posted Requests
4. A advertises the released buffer space to B
5. B sends another Memory Read Request



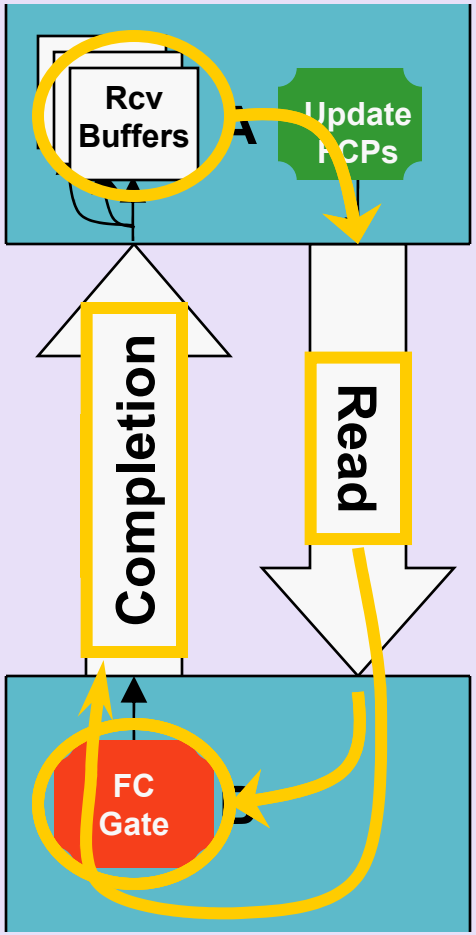
Receive Buffer Sizing and Flow Control Credit Return Policy

- Receive Buffer sizing and Flow Control (FC) credit return policy have a significant effect on performance
  - ✓ Much inbound traffic → Receive Buffer optimization critical
- Tradeoff: Bandwidth for Update Flow Control Packets (UpdateFCPs) vs. Receive Buffer Size
  - ✓ UpdateFCPs affect *outbound* traffic
- Width & Max/Typical Payload size must be considered



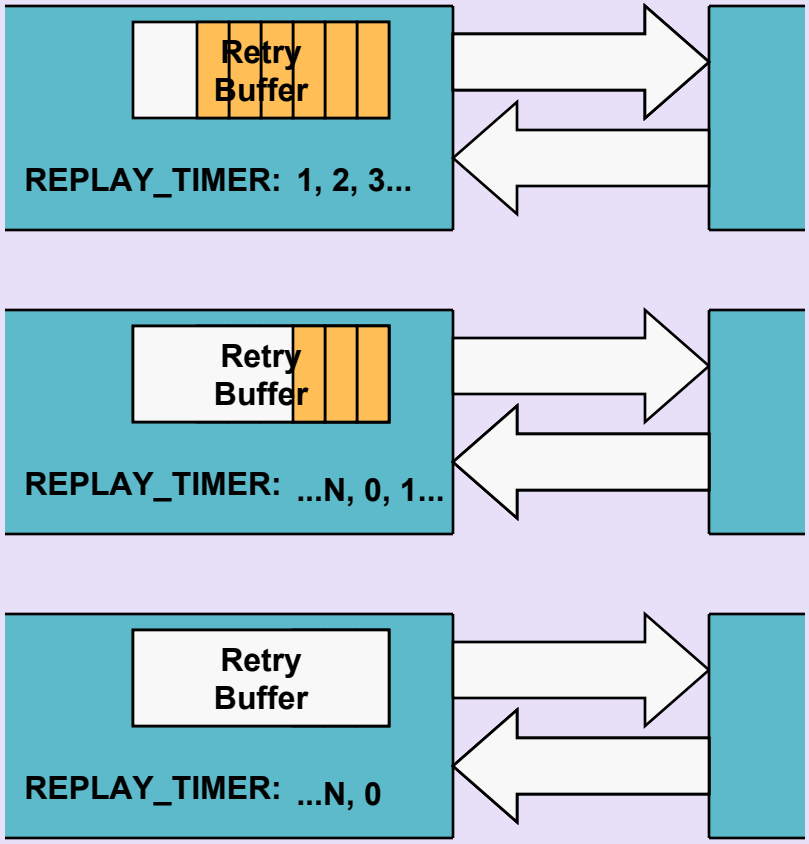
Flow Control Credit Return Policy

- Flow Control Credit Return (Update) policy = F(Receive Buffer Size, Round-Trip Delay)
- Consider:
  - ✓ Delay to transmit, potentially including L0s exit
  - ✓ Time for other component to process update
  - ✓ Credit consumed by incoming TLPs “in flight”
- Additional considerations for Completions:
  - ✓ Outbound Non-Posted Requests determine performance requirements
  - ✓ Key factors: Requested data size, Number of Requests simultaneously pending
- Balance Performance against Cost
  - ✓ Some applications may not require optimal behavior
    - – Example: Incoming control writes may be rare



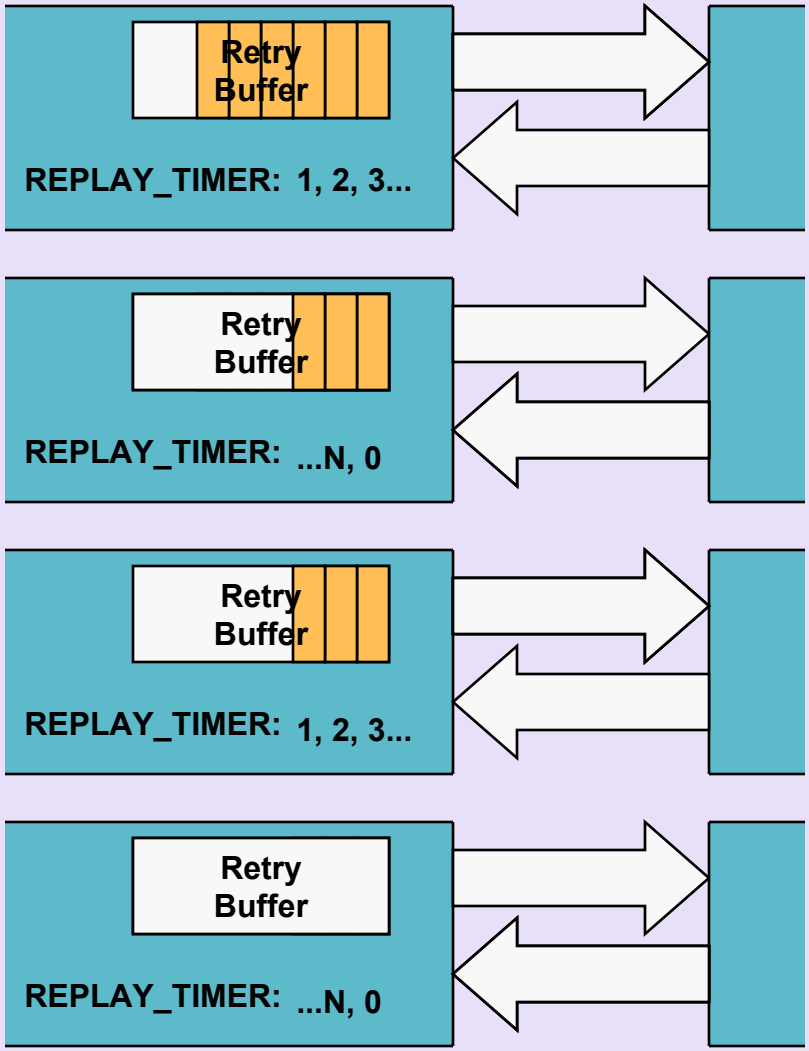
Retry Buffer & REPLAY_TIMER Operation – Example 1

- Six TLPs are transmitted
  - ✓ REPLAY_TIMER started
- An Ack is received that acknowledges three of the transmitted TLPs
  - ✓ REPLAY_TIMER reset and restarted, because there are still outstanding unacknowledged TLPs
- ... eventually, the remaining TLPs are Ack'd
  - ✓ REPLAY_TIMER resets and holds – no outstanding unacknowledged TLPs



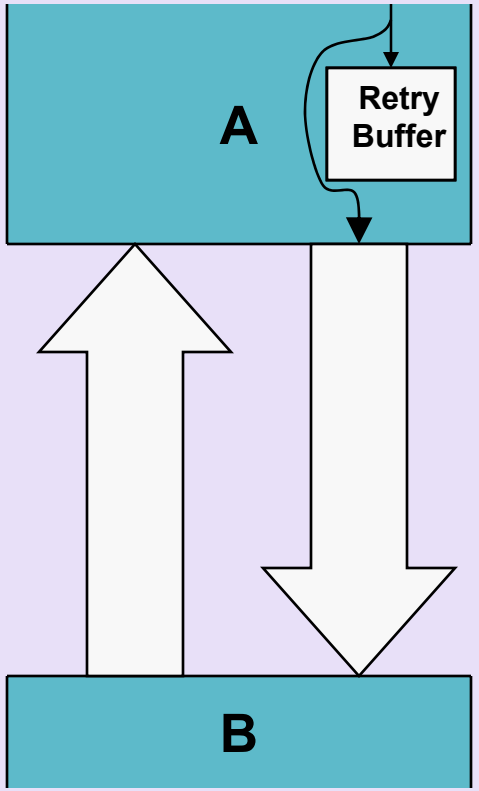
Retry Buffer & REPLAY_TIMER Operation – Example 2

- Six TLPs are transmitted
  - ✓ REPLAY_TIMER started
- *Nak* acknowledges three of the transmitted TLPs...
  - ✓ REPLAY_TIMER resets and holds
- ... causes retransmission of other three
  - ✓ REPLAY_TIMER restarted
- ... eventually, the remaining TLPs are Ack'd
  - ✓ REPLAY_TIMER resets and holds – no outstanding unacknowledged TLPs



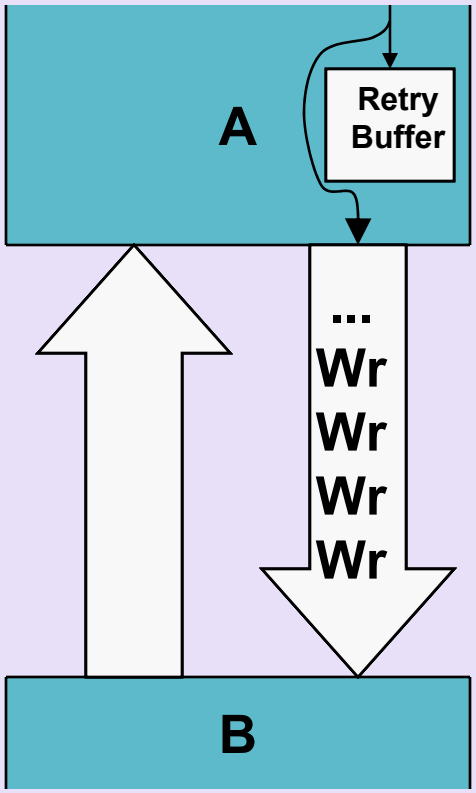
Retry Buffer Sizing

- Transmitter out of Retry Buffer space → Stop transmitting
  - ✓ All transmitted TLPs must be kept in Retry Buffer until Ack'd by the other component on the Link
    - Note that this includes Completions as well as Requests
  - ✓ Much outbound traffic → Retry Buffer optimization critical
- Ack policy and L0s also affect optimal Retry Buffer sizing



Avoid Stalls due to Slow L0s Exit

- Initially, both *A*→*B* and *B*→*A* are in L0s
- *A*'s transmitter exits L0s and starts a long burst of write Requests
- *B*'s transmitter initiates L0s exit to send Ack...
- ...but the *A's* receiver L0s exit time is large, and *A* stalls due to lack of Retry Buffer space
  - ✓ *A* is stalled by *A's* own slow L0s exit time
- *B*→*A* returns to L0, *B* transmits an Ack DLLP to *A*, and the stall is resolved



Traffic Types & Receive Buffer Management

- Posted Requests (Memory Writes)
  - ✓ Many incoming writes → Optimize Performance
    - Example: frame buffer
  - ✓ Typical control operations → Optimize Cost
- Non-Posted Requests (Reads)
  - ✓ Many incoming reads → Optimize Performance
    - Example: Root Complex (main memory)
  - ✓ Typical control operations → Optimize Cost
- Completions
  - ✓ Endpoint/Root Complex: Match to *outbound* Reads
  - ✓ Switch: Support full Link Bandwidth

Related Efficiency Considerations

- Flow Control data credit granularity also has an effect on utilization
  - ✓Payloads of multiples of 4DW are optimal
- For best service, 128B address alignment highly desirable
- For B/W efficiency reasons, requests of 128B or larger are recommended
  - ✓Local buffering/caching may be desirable

Agenda

- Link Data Integrity & End-To-End CRC
- Error Signaling & Logging
- Flow Control & Buffering Optimizations
- Hot-Plug
- Power Management
- Specification Update
- Summary & Call to Action

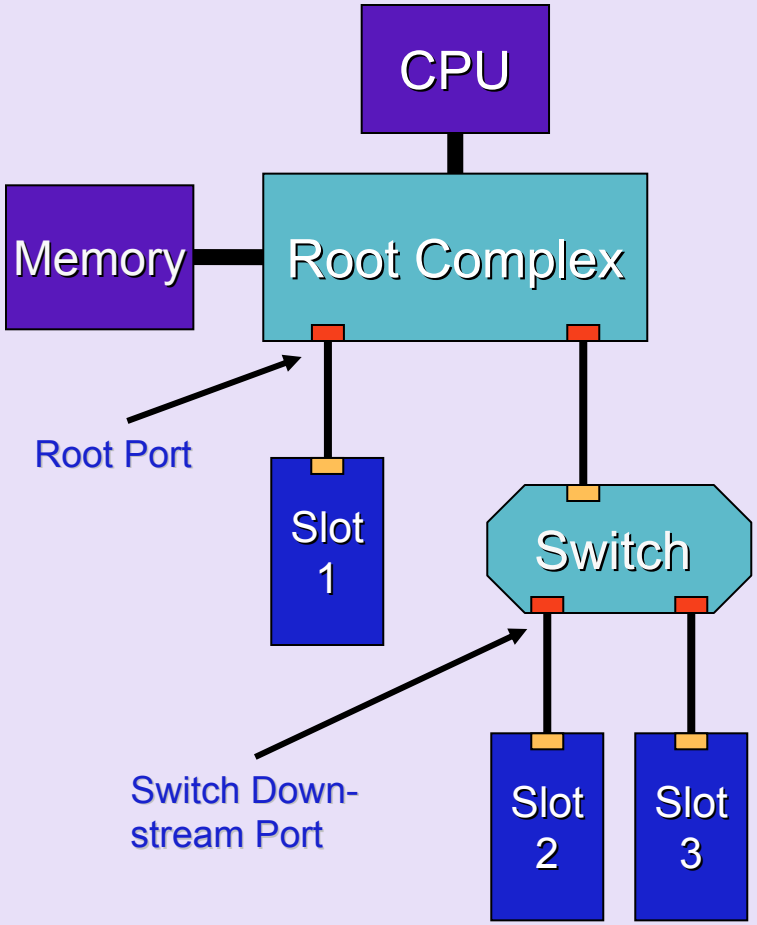
PCI Hot-Plug Refresher

- PCI Hot-Plug enables I/O adapter insertion/removal without requiring system downtime or interruption
  - ✓ Burden placed on platform – minimal impact to adapter
- 1997: PCI Hot-Plug 1.0 introduced
- 2001: SHPC 1.0 & PCI H-P 1.1 standardized hot-plug controller functionality and usage model
  - ✓ Enables OSV support of hot-plug without platform specific software
- 2002: PCI Express hot-plug developed as integral part of the PCI Express Base 1.0 specification
  - ✓ Defines native “toolbox” of hot-plug mechanisms. Each form factor spec defines which are required, optional, or not applicable
  - ✓ Currently leveraged by CEM, SIOM, & ExpressCard* form factors

PCI Express hot-plug builds on 5+ years of SIG experience with hot-plug

PCI Express Topology

- Hot-Plug enabled in Root Ports and Switch Downstream Ports
- Ports appear to software as PCI-to-PCI Bridges
- Each Port implementing a slot contains its own Hot-Plug register set in the Bridge's Configuration Space
- Hot-Plug register set reports presence or absence of defined hot-plug mechanisms



Hot-Plug Elements

Element	Purpose
Power Indicator – Green	Visually indicates the power state (on/off/transitioning) of the slot/card
Attention Indicator – Yellow/Amber	Indicates operational problem (blink) Identifies slot/card for service (on)
Attention Button	Press to initiate hot-plug operation Press again within 5 sec to cancel
Hot-Plug Signaling Messages (in-band)	Permit electrical control of Indicators, Attention Button by device on adapter
Manually-operated Retention Latch (MRL)	Rigidly holds adapter in place Stability for cable attach/detach, etc
MRL Sensor	Reports position of MRL. Triggers slot power removal if unexpectedly opened
Electromechanical Interlock	Prevents adapter removal while power is on. Optionally provides physical security for adapter.

Unified set of hot-plug mechanisms

Hot-Plug Elements (cont.)

Element	Purpose
Presence Detect	Reports presence/absence of card. Both pin-based and in-band (Beacon) mechanisms supported.
Power Controller	Controls main power for an adapter. Also reports main & aux power faults.
Slot Numbering	Uniquely identifies slot in a system. PCIe Physical Slot Number register plus optional Chassis Number.
Software User Interface	OSV specific. Method to monitor slot status. Alternate means to initiate hot-plug operations.

Hot-Plug Events

- Hot-plug events reported via Slot Status Register fields
- Slot Events:
  - ✓ Attention button pressed
  - ✓ Power fault detected
  - ✓ MRL sensor changed
  - ✓ Presence detect changed
- Command completed event:
  - ✓ Indicates hot-plug controller ready to accept new command
  - ✓ Does NOT guarantee all actions for previous command have completed
- Hot-plug events may trigger software notification
  - ✓ Global enable plus individual enables per event
  - ✓ Generates a system interrupt for OS with native hot-plug support
    - – Redirect to a GPE for ACPI-based legacy OS
- Hot-plug events may also initiate platform wakeup

Wakeup and PME

- PCI PME# – Used both to wake system and to request power management state change
- PCI Express approach – Break functionality into two separate components *while maintaining compatibility with PCI-PM*:
  - ✓ Platform wakeup via WAKE# sideband signal or in-band Beacon
  - ✓ PM state change request via in-band PM_PME message
- Don't signal Wakeup to OS
  - ✓ Make Wakeup visible to platform power controller only
  - ✓ Must not be combined with PCI PME#
- System and adapter support for Wakeup is optional
  - ✓ Vaux must be enabled for components participating in Wakeup
    - – Includes components that propagate Beacon
- Device support for PCI PM D0 & D3 states is required
  - ✓ Support for additional states is optional

Wakeup and PME (cont.)

- PM_PME sent by device requesting PM state change
  - ✓ Follows platform wakeup when requesting D3 ➔ D0 transition
  - ✓ Link must be active (L0 state) to transmit PM_PME
- PM_PME is routed upstream to Root Complex
  - ✓ TLP header uniquely identifies source via Requester ID
  - ✓ Exception: PCI devices behind a bridge may not be uniquely identified
- Root Complex stores PM_PME Requester ID in Root Status Register and sets PME Status bit
  - ✓ Generates a system interrupt for OS with native PME support
    - – Redirect to a GPE for ACPI-based legacy OS
- Deadlock avoidance requires Root Complex to discard PM_PME messages if buffer space overflows
  - ✓ Initiator re-sends PM_PME if unserviced after 100 msec timeout

I/O Adapter Reset

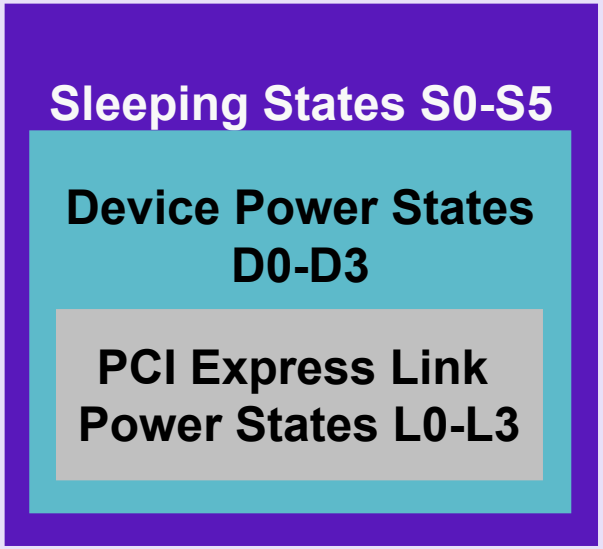
- PCI Express Cold, Warm, and Hot Reset
  - ✓ Cold Reset: Fundamental Reset accompanying application of power Generally signaled via PERST#
  - ✓ Warm Reset: Fundamental Reset without power removal and re-application
  - ✓ Hot Reset: Signaled in-band (physical layer)
- All device state must be re-initialized by Fundamental Reset
  - ✓ Exception: Specified Power Management context when Vaux is enabled
- "Sticky" registers excluded from Hot reset
  - ✓ E.g., many Advanced Error Reporting registers
- I/O adapters with on-board voltage regulation must be properly reset with PERST# assertion
  - ✓ On-board voltages may not decay sufficiently to trigger self-detected reset when slot power is cycled

Agenda

- Link Data Integrity & End-To-End CRC
- Error Signaling & Logging
- Flow Control & Buffering Optimizations
- Hot-Plug
- Power Management
- Specification Update
- Summary & Call to Action

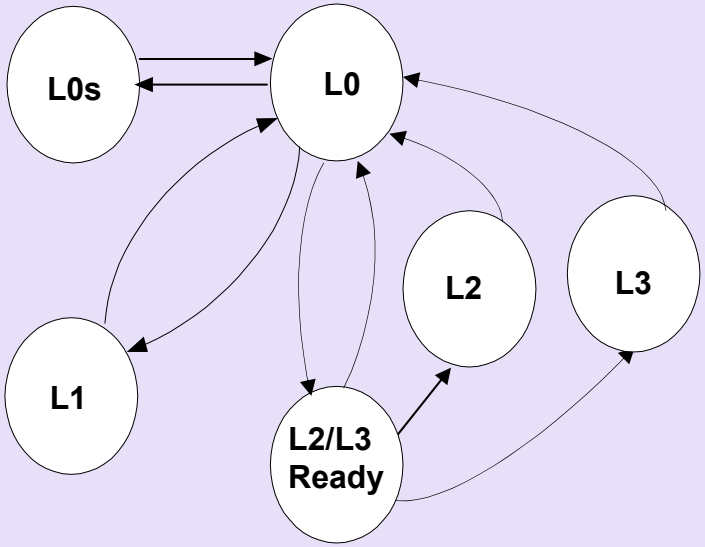
PCI Express Power Management

- Builds on PCI Power Management (PM)
  - ✓ Compatible with existing PCI PM software stacks
- System Level, Device Level and Link Level PM States
- Enhanced PM capabilities
  - ✓ Aggressive power reduction through Active State PM (L0s, L1)
  - ✓ Improved PME using in-band messaging
  - ✓ Improved definition and SW control of Vaux



PCI Express Advances Platform PM While Preserving Software Investment

Link PM States Summary



Downstream Component D-State	Permissible Link Power State
D0	L0, L0s, L1
D1	L1
D2	L1
$D3_{hot}$	L1
$D3_{cold}$	L2, L3

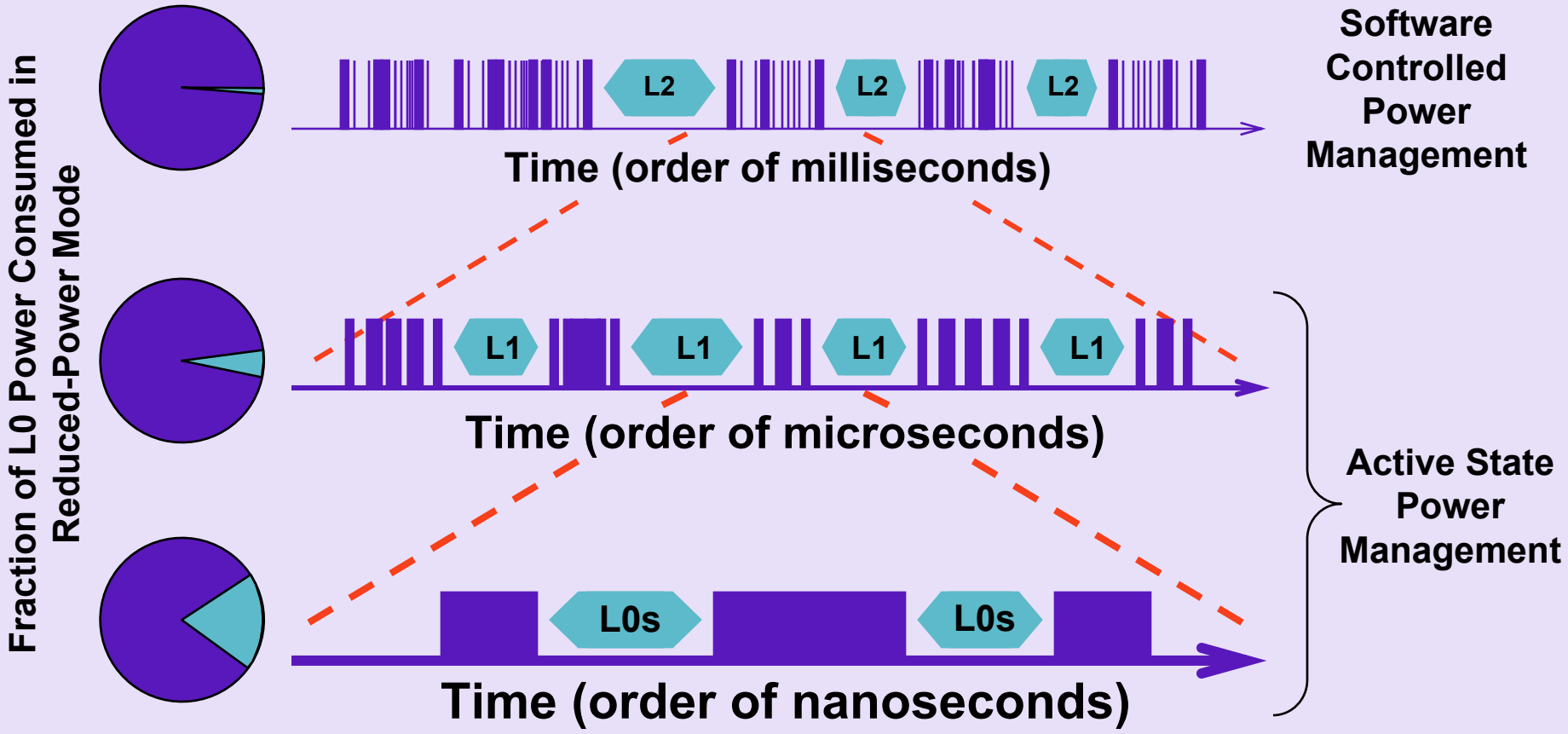
Software Controlled Power States

Active State PM States

Active State Power Management

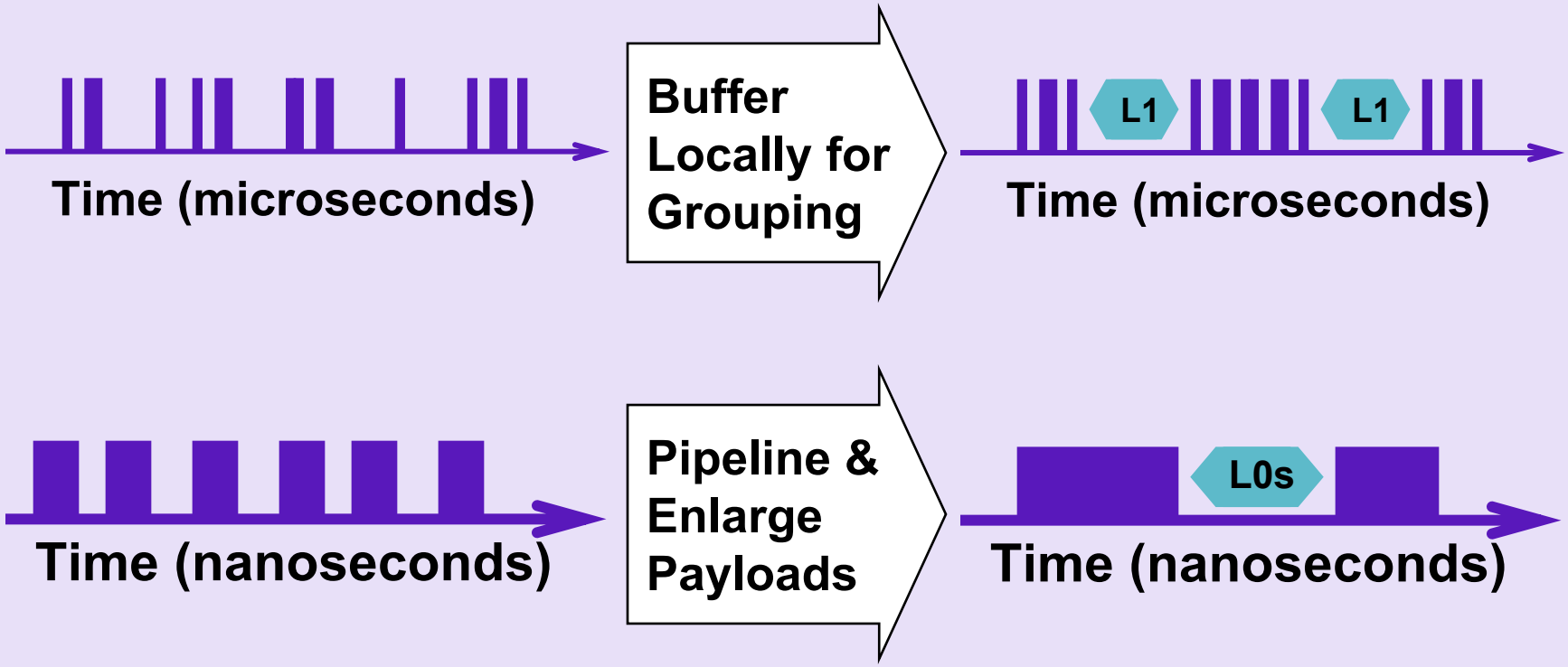
- Software controlled power management (PCI-PM) provides mechanisms for intelligent PM
  - ✓ Important in conventional PCI, important in PCI Express
  - ✓ PCI Express is PCI-PM compatible
- Active State Power Management (ASPM) provides *additional* benefit
  - ✓ Low latency – minimum impact on performance
  - ✓ Finer granularity of control – more opportunity for power savings compared to software controlled PM
- Serial signaling technology consumes power when not sending data
  - ✓ “idle” = logical idle, not electrical idle
  - ✓ PCI Express uses ASPM to reduce power in logical idle
- ASPM is important to minimize power consumption with minimum performance impact

Active State Power Management Fills the Gaps



- ASPM Provides “Transparent” Power Savings
- Transition aggressively to lowest power mode

Use Pipelining and Grouping to Maximize Power Savings



- Minimize handshake overhead to maximize bandwidth utilization → Reduce Power
  - ✓ Ack multiple packets at once
  - ✓ Enlarge Receive Buffers → Fewer FC updates
  - ✓ Recommend 3-4 transactions per FC update/Ack

Agenda

- Link Data Integrity & End-To-End CRC
- Error Signaling & Logging
- Flow Control & Buffering Optimizations
- Hot-Plug
- Power Management
- Specification Update
- Summary & Call to Action

PCI Express Base 1.1

- Coming Soon! – New spec revision incorporating published ECNs and errata to Rev 1.0a
- Upcoming ECNs:
  - ✓ Hot-Plug: Addresses hot-plug messages, software-hardware interactions, EM Interlock support
  - ✓ Surprise Down: Reports surprise link down errors, adds Data Link Active status bit
  - ✓ FC Init: Clarifies/corrects requirements for flow control initialization. Incorporates suggestions from earlier review.
  - ✓ Posted Request Acceptance: Imposes 10 μsec acceptance limit similar to that in conventional PCI
- Recently published ECNs
  - ✓ PME Turn Off (May ’04)
  - ✓ Platform Ref Clock PM Mechanism (Apr ’04)
  - ✓ VSEC (Apr ’04)
  - ✓ CRS Software Visibility (Apr ’04)

PCI Express Base 1.1 (cont.)

- Recent spec Errata:
  - ✓ PME_Turn_Off – Clarifies behavior entering and exiting link "off" states
  - ✓ Link & VC Initialization – Clarifies rules regarding TLP transmission & reception during link & VC init
  - ✓ Hot-Plug – Clarifies hot-plug controller behavior, removes form factor specific language
  - ✓ Multi-function devices – several clarifications regarding the behavior of multi-function devices
  - ✓ Errata to the Root Complex Integrated Devices & Event Collector and the Root Complex Topology Discovery ECNs
  - ✓ Miscellaneous additional clarifications

Agenda

- Link Data Integrity & End-To-End CRC
- Error Signaling & Logging
- Flow Control & Buffering Optimizations
- Hot-Plug
- Power Management
- Specification Update
- Summary & Call to Action

Summary & Call to Action

- PCI Express advances overall platform capabilities while preserving PCI architecture and software investments
- New advanced capabilities/features enable important emerging applications
- Call to Action:
  - ✓ Optimize your baseline PCI Express products and extend their features
  - ✓ Engage with PCI-SIG to take the advantage of technical enabling
  - ✓ Stay up-to-date: Monitor the PCI SIG website for new ECNs and errata. Participate in the member comment periods
  - ✓ Participate in PCI-SIG compliance and interoperability forums

Thank you for attending the
PCI-SIG Developers Conference 2004.

For more information please go to
www.pcisig.com

PCI Express™ Advanced Hardware Topics & Specification Updates

Jasmin Ajanovic – Chair,PCI Express Protocol Workgroup

Carl Jackson – Member, PCI Express Protocol Workgroup


PCI
SIG™