PCIe® Post-3.0 Protocol Changes

Mahesh Wagh
Protocol Workgroup Member
Intel Corporation

Disclaimer

The information in this presentation refers to specifications still in the development process. This presentation reflects the current thinking of various PCI-SIG® workgroups, but all material is subject to change before the specifications are released.

PCIe® Post-3.0 Protocol Update

- ECNs/ECRs Against the PCIe® 3.0 Base Spec
  - ✓ Process Address Space ID (PASID)
  - ✓ 8.0 GT/s Receiver Impedance
  - ✓ Lightweight Notification (LN) Protocol
  - ✓ L1 PM Substates with CLKREQ#
  - ✓ Downstream Port Containment (DPC)
  - ✓ Enhanced DPC (eDPC) – under development
  - ✓ Fast Link Training (FLT) – under development
  - ✓ Function Readiness Status (FRS) – under development
  - ✓ Separate RefClk Independent SSC (SRIS) – under development
  - ✓ Precision Time Measurement (PTM) – under development
- PCI Code and ID Assignments Spec & Associated ECNs
  - ✓ Class Code & Capability ID Extraction ECN
  - ✓ PCI Code and ID Assignment Specification
  - ✓ PCI Code & ID Assignment Specification Update ECN
  - ✓ Additional PCI Code & ID Assignment Spec ECNs

Process Address Space ID (PASID)

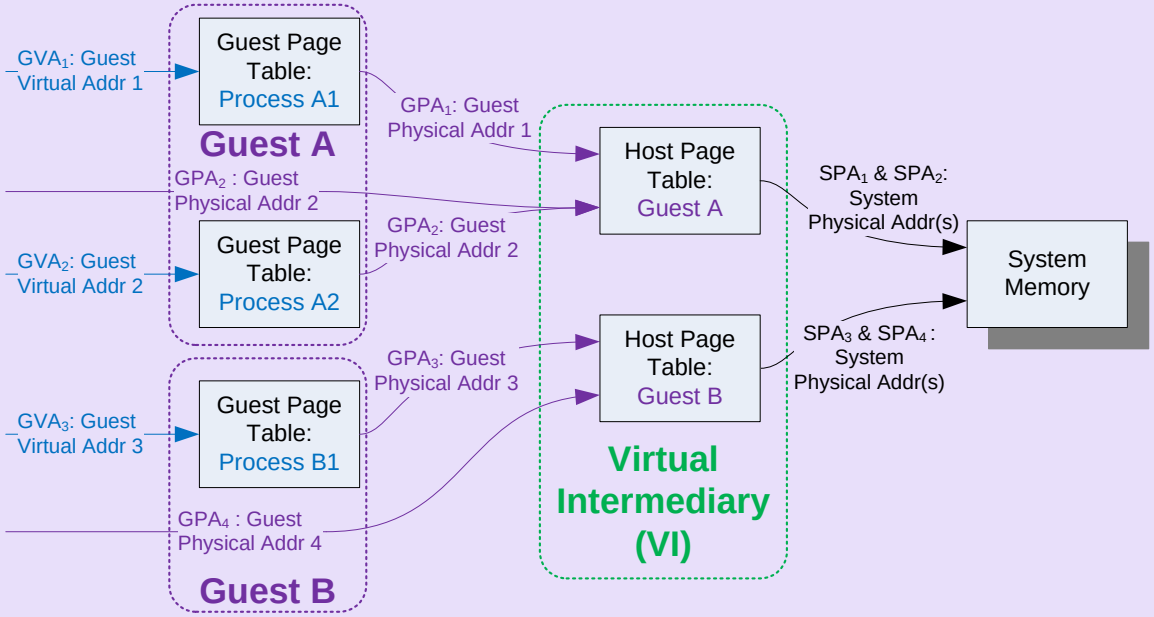
IOV Background

- Address Translation Services (ATS) supports:
  - ✓ Performance optimization for direct assignment of a Function to a Guest OS running on a Virtual Intermediary (Hypervisor)
- Page Request Interface (PRI) supports:
  - ✓ Functions that can raise a Page Fault
- Single Root-I/O Virtualization (SR-IOV) supports:
  - ✓ Light-weight Functions (Virtual Functions)
  - ✓ Large numbers of Functions (multiple Bus Numbers)

PASID Overview

- Supports Direct Assignment of I/O to a User Process running on a Guest OS running on a Virtual Intermediary
  - ✓ Untranslated Memory Requests
  - ✓ Translation Requests
  - ✓ Translation Invalidations
  - ✓ Page Requests
- Supports Execute Permission
- Supports Privileged Mode

PASID Address Mapping



Address Translation Cache (either in TA *or* in Function's ATC)

$GVA_1/A1 \rightarrow SPA_1$	
$GVA_2/A2 \rightarrow SPA_2$	
$GPA_2 \rightarrow SPA_2$	
$GVA_3/B1 \rightarrow SPA_3$	
$GPA_4 \rightarrow SPA_4$	

Cache Entry Type	*Meaning*
GVA/PASID → SPA	*TA / ATC Entry with PASID*
GPA → SPA	*TA / ATC Entry without PASID*

- 3 User Processes
- 2 Guests
- Cache Example:
  - ✓ 3 GVA Entries
    - GVA_1 / A1 → → SPA_1
    - GVA_2 / A2 → → SPA_2
    - GVA_3 / B1 → → SPA_3
    - Intermediate GPA not cached
  - ✓ 2 GPA Entries
    - GPA_2 → SPA_2
    - GPA_4 → SPA_4
- Directly Assigned to User Process
  - ✓ GVA_1 / GVA_2 / GVA_3
- Directly Assigned to Guest
  - ✓ GPA_2 / GPA_4
- … → SPA_2 cached twice
  - ✓ GVA_2 / A2 → → SPA_2
  - ✓ GPA_2 → SPA_2

PASID is TWO ECNs

- Process Address Space ID ECN
  (PCI Express Base 3.0)
  - ✓ PASID TLP Prefix
  - ✓ Usage on Untranslated Memory Requests
  - ✓ PASID Capability
- PASID Translation ECN
  (Address Translation Services 1.1)
  - ✓ Usage on ATS Requests
  - ✓ Usage on ATS Invalidation Requests
  - ✓ Usage on PRI Requests
  - ✓ Usage on PRG Responses
  - ✓ ATS Invalidation rules

PASID TLP Prefix

- Permitted on:
  - ✓ Untranslated Memory Request
    - Including Untranslated AtomicOP Requests
  - ✓ ATS Translation Request
  - ✓ ATS Invalidation Request
  - ✓ Page Request Interface Request (PRI)
  - ✓ Page Request Group Response (PRG)

Base ECN

ATS ECN

- PASID does not require ATS support
  - ✓ Functions can use only Untranslated Memory Requests
  - ✓ Without ATS support, pages must be pinned
- PASID does not require PRI support
  - ✓ PRI permits paging and late pinning
    - User Process/Guest OS level
      or
    - Guest OS/Hypervisor level
  - ✓ Without PRI support, pages must be pinned

Complete presentation on PASID given by IOV Workgroup in past DevCons

8.0 GT/s Receiver Impedance

8.0 GT/s Receiver Impedance ECN

- Impacts Receivers that operate at 8.0 GT/s with an impedance other than the range defined by the Z_{RX-DC} parameter for 2.5 GT/s (40-60 Ohms)
- Adds new requirements to avoid deadlock scenarios caused by such Receivers not being detected by an opposite component in the LTSSM Detect State
- Makes minor changes to several LTSSM States:
  - ✓ Polling.Compliance
  - ✓ Polling.Configuration
  - ✓ Rx_L0s.Idle, L1.Idle, & L2.Idle
  - ✓ Disabled

Lightweight Notification (LN) Protocol

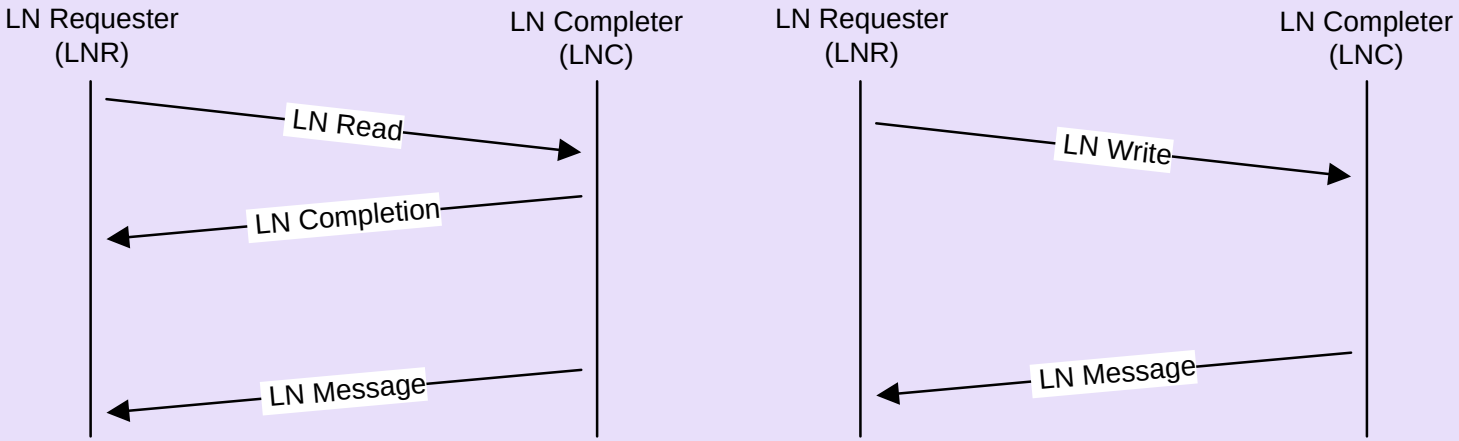
LN Protocol: Overview

- An optional-normative simple protocol:
  - ✓ A device can *register* one or more cachelines in host memory, and later be notified by a hardware mechanism when any registered cachelines are updated
  - ✓ Architected support for 64-byte & 128-byte cachelines
  - ✓ New LN Requester Capability structure for software to discover and manage LN Requester capabilities in Endpoints
  - ✓ New field in Device Capabilities 2 register to inform software of LN Completer capabilities in the host
- Transactions
  - ✓ LN Reads/Completions/Writes – special forms of Memory Reads/Completions/Writes with registration semantics
  - ✓ LN Messages – SIG-defined Vendor-Specific Messages to convey notifications regarding existing registrations
  - ✓ No changes required for PCIe Switches

LN Protocol: Example Benefits

- Reduction of I/O bandwidth consumption & I/O latency:
  - ✓ Device caching can significantly reduce I/O bandwidth consumption and I/O latency for some applications
  - ✓ Reducing I/O bandwidth consumption also reduces host memory subsystem bandwidth consumption
- Lightweight signaling:
  - ✓ LN Protocol enables host user-space software to signal a device by updating a cacheline as opposed to performing a PIO operation, which has higher software overhead and synchronization/flow-control issues
- Dynamic device associations:
  - ✓ VM guest drivers communicating with a device via host memory structures enables easier VM guest migration and switching between virtualized and direct I/O for that device

LN Protocol: Basic Operation



- LNR reads & registers a cacheline using LN Read
- LNC acknowledges registration & returns data with LN Completion
- Later, LNC notifies LNR with LN Message when cacheline is updated

- LNR writes & registers a cacheline using LN Write
- Later, LNC notifies LNR with LN Message when cacheline is updated

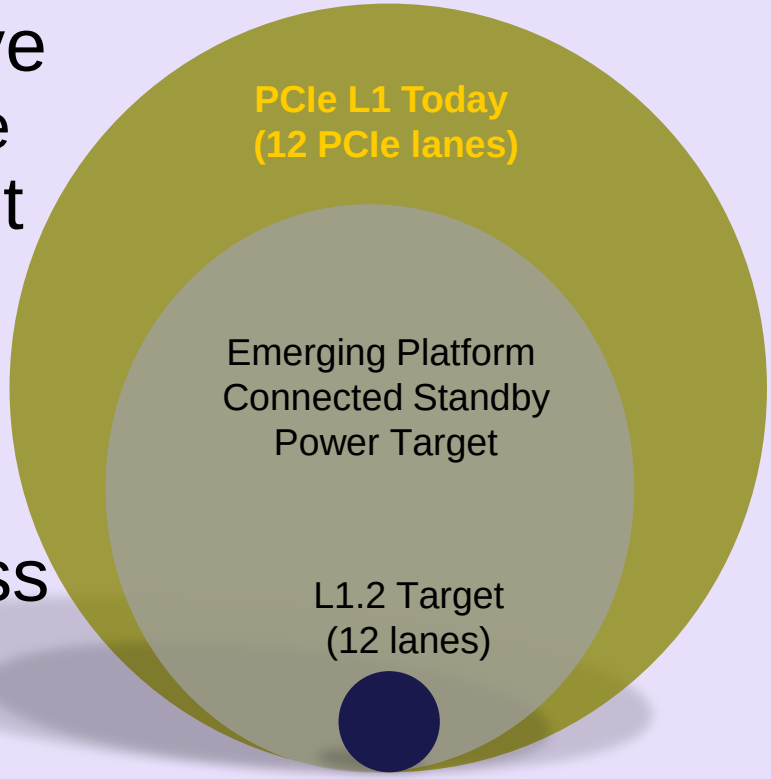
LN Protocol: Additional Attributes

- Evictions: LN Completer can evict registrations when it runs short on resources
- LN Message Notification Reasons:
  - ✓ Registered cacheline was updated
  - ✓ Registered cacheline was evicted
  - ✓ All registered cachelines for this LNR were evicted
- Zero-length LN Reads for probing without registering
  - ✓ LN-capable host not required to support LN for all memory regions
  - ✓ LN-capable hosts are required to support LN with 4KB granularity
- Zero-length LN Writes for explicit deregistration
  - ✓ Enables LN Requester to manage its outstanding registrations
- LN Requester Capability Structure provides:
  - ✓ Advertisement of maximum outstanding registrations
  - ✓ Software specified limit for maximum outstanding registrations

L1 PM Substates with CLKREQ#

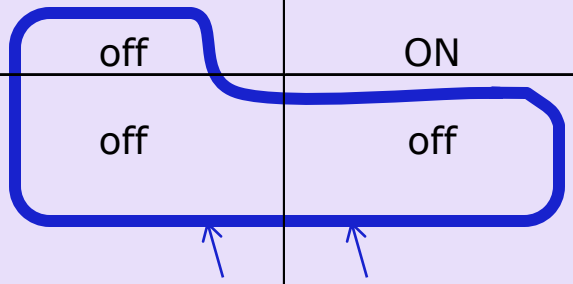
L1 Power Management (PM) Substates Motivation

- Link idle power ~10% of Active
  - ✓ On the order of 10 mW per lane
- PCIe L1 Power does not meet stand-by power requirements for emerging thin and light form factor markets
- Regulatory requirements are driving down idle power across multiple market segments
- Platforms require idle power near zero
- Retain backwards compatibility
- Add minimum cost



Power and Latency Solutions

	Port Circuit Power On/Off			Target Results*	
Sub-State	**PLL**	**Rx/Tx**	**Common-Mode Keepers**	**x1 Port Power**	**Exit Latency**
L1 (unmodified)	ON	off/idle	ON	25mW	2µs (retrain)
L1+CLKREQ (unmodified)	off	off/idle	ON	10mW	20µs (PLL)
L1.1	off	off	ON	300µW	20µs (PLL)
L1.2	off	off	off	10µW	70µs (Common mode restore + other delays)



Note: Power savings will provide near linear scaling for multi-lane links.

*** These are targets for power and latency, not specified results.**

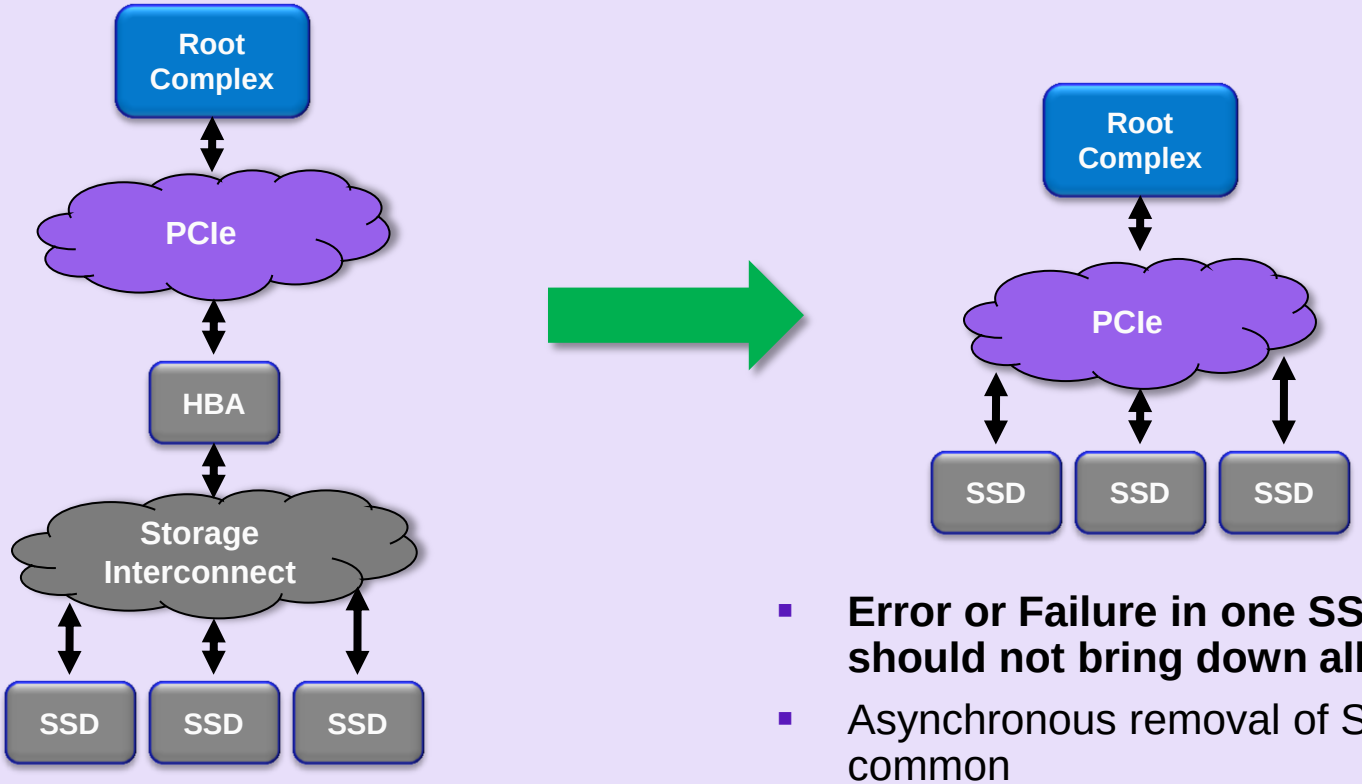
L1 PM Substates High-Level Overview

- New power management substates of L1 Link state
  - ✓ Enables dramatically lower idle power by removing power from high-speed circuits
  - ✓ Applicable to both ASPM & PCI-PM L1 Link states
  - ✓ Entire ECR & various aspects of it are optional normative
- ECR history
  - ✓ Original ECR utilized **CLKREQ#** sideband signal for new purpose to manage L1 PM Substates
  - ✓ ECR later expanded to define a Low Frequency Activate Signaling **(LFAS)** inband mechanism to manage L1 PM Substates for platforms where CLKREQ# is not available
  - ✓ L1 PM Substates with CLKREQ# ECN completed membership review & LFAS ECR is under development

Downstream Port Containment (DPC)

DPC Motivation

- Emerging PCIe usage models are creating a need for improved error containment/recovery and support for asynchronous removal (a.k.a. hot-swap)



- **Error or Failure in one SSD should not bring down all SSDs**
- Asynchronous removal of SSDs is common

Details of Downstream Port Containment

- Downstream Port Containment (DPC) is triggered when:
  - ✓ An unmasked uncorrectable error is detected by the Downstream Port, or…
  - ✓ The Downstream Port receives an uncorrectable error Message (ERR_NONFATAL or ERR_FATAL). There is a mode to pass through ERR_NONFATAL w/o triggering DPC.
- When DPC is triggered:
  - ✓ The Link is immediately Disabled (LTSSM Disabled state) and subsequent TLPs are blocked (including the Error message if it triggered DPC)
  - ✓ The cause of DPC is recorded in the DPC Trigger Reason field
  - ✓ The Downstream Port can signal this event by sending an interrupt, an ERR_COR Message, or both
- During DPC the Link remains Disabled. The Downstream Port:
  - ✓ Completes Non-Posted Requests with either a UR or CA Completion Status (controlled by a configuration bit)
  - ✓ Participates in PME_Turn_Off handshake protocol
  - ✓ Handles Vendor Defined Requests in the same way as Link Down
  - ✓ Silently discards all other Posted Requests
- DPC is exited and normal operation resumes when host software clears the DPC Trigger Status field

Summary of Overall DPC ECN

- Is optional normative, applying to Switch Downstream Ports and Root Ports
- Defines an error containment mechanism, automatically disabling a Link when an uncorrectable error is detected, preventing potential spread of corrupted data
- Defines an optional ability for Requesters to log the TLP prefix/header of the Request associated with a Completion Timeout. (Completion Timeouts will occasionally occur as a side-effect of asynchronous removal.)
- Defines an optional ability to prevent the automatic transmission of a Set_Slot_Power_Limit Message upon the Link transition to DL_Up. This can help avoid power surges when many devices power up concurrently.
- Does a minor cleanup of hot-plug terminology
  - ✓ Changes “hot-swap” references to “hot-plug”
  - ✓ Defines the concept of asynchronous removal and adds a section describing the system implications
  - ✓ Generalizes the “presence detect pin” to “out-of-band presence detect”
- A subsequent ECN is under development for defining DPC extensions that are specific to Root Ports

Enhanced DPC (eDPC)
(under development)

eDPC Overview

- DPC, covered earlier, provides uncorrectable error containment for Root Ports (RPs) and Switch DPs
- eDPC defines DPC extensions that are specific to RPs
  - ✓ Fine-grained controls for managing RP Programmed I/O (RP PIO) errors, notably if/when they trigger DPC
  - ✓ Completion synthesis for outstanding Non-Posted Requests when DPC is triggered, to avoid Completion Timeouts
  - ✓ DPC RP Busy bit so software can determine how long to leave a DP in DPC following a DPC trigger event
- Firmware WG needs to explore how System Firmware and the OS will arbitrate ownership of the DPC Capability struct

RP PIO Error Controls

Fine-grained controls for when Root Port PIOs encounter uncorrectable errors

- Mask & Status bits for Unsupported Request (UR), Completer Abort (CA), & Completion Timeout (CTO)
- Individual sets of UR/CA/CTO bits for Config Space, I/O Space, and Memory Space
- First Error Pointer (FEP) field & Log registers similar to (but distinct from) those in AER
- SysError bits to control which errors generate a System Error vs normal DPC containment
- Exception bits to control which errors generate a synchronous exception vs returning a value of all 1's
  - ✓ "Exception" is a generic term for interrupt, trap, machine check, etc.

Fast Link Training (FLT)
(under development)

FLT Motivation & History

- Goal: To reduce training time and subsequent training times to improve system performance, stability, efficiency, and experience
- Has been explored in the Protocol WG for many months, with major evolutions of the ECR
- Spun off a separate effort which resulted in the Function Readiness Status (FRS) ECR, covered in subsequent slides
- ECR based on Link components retaining state from earlier Link training, and using that state to streamline and accelerate future Link training

Current FLT Summary

- Control mechs for saving & reusing training state
  - ✓ Can be applied independently on each end of the Link
  - ✓ FLT info can be stored in either non-volatile memory or volatile (sticky) memory
- Optimizations for LTSSM Detect.Active
  - ✓ Avoids 12ms delay caused when there is a width mismatch between Upstream & Downstream components
- Optimizations for LTSSM Recovery.Equalization
  - ✓ Potentially dramatic reduction in Phase 2 & 3 of Tx EQ
  - ✓ For both Phase 2 & 3, reduction from a max of 24ms to a max of 2ms with a typical of 50µsec
  - ✓ Actual improvement highly dependent upon specific LTSSM implementations

Function Readiness Status (FRS)
(under development)

FRS Motivation & Approach

- Avoid relatively long architected fixed delays following various forms of reset before software is permitted to perform its first Configuration access
  - ✓ **1 second** if Configuration Retry Status (CRS) is not used
  - ✓ **100ms** for most cases if CRS is used
- Avoid the complexity of using the CRS mechanism, potentially polling periodically up to 1 second following reset
- Specific cases for device or Function to become ready:
  - ✓ **Device** becoming ready following DL_Down to DL_Up
    - Exit from Cold Reset (initial power-up, hot-add, or D3cold)
    - Exit from Warm Reset, Hot Reset, or Disabled
    - Exit from L2/L3 Ready (D3hot w/ PME_Turn_Off)
  - ✓ **Function** becoming ready following D3hot/D0 transition or FLR
- Approach: have the device or Function send a Message in these cases when it's ready for Configuration with no CRS

FRS & DRS Messages

- Two distinct types of Messages
  - ✓ Device Ready Status (DRS)
  - ✓ Function Ready Status (FRS)
  - ✓ Both are PCI-SIG defined Type 1 VDMs with no payload
  - ✓ Receivers of these Messages ignore them if unrecognized
- DRS Message: uses local routing
  - ✓ Sent following DL_Down to DL_Up transition when device becomes ready for 1st Config access
  - ✓ In an MFD, all Functions (other than VFs) must be ready
  - ✓ Being “ready” also means device won’t send any CRS Completions
- FRS Message: uses route-to-root routing
  - ✓ Sent when an individual Function becomes ready
  - ✓ Requester ID indicates which Function became ready
  - ✓ FRS Reason field in Message indicates why Function became ready

FRS Functionality in Different Components

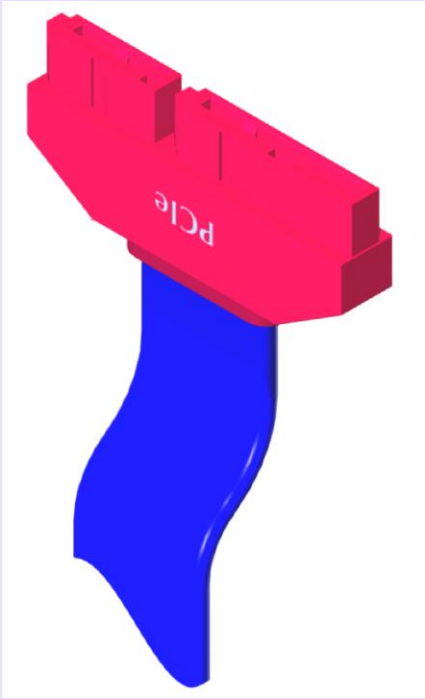
- Functionality in Switch Upstream Ports & Endpoints
  - ✓ Ability to send **FRS & DRS Messages** Upstream for cases described earlier
- Functionality in both RPs & Switch Downstream Ports
  - ✓ **DRS Message Received** bit – indicates this event
  - ✓ **DRS Signaling Enable** bit – when set, enables Downstream Port to signal the Root Port when it receives a DRS Message, by sending its own FRS Message
- Additional Functionality in Root Complex
  - ✓ A queuing mechanism for received FRS Messages
  - ✓ Queue depth between 1 & 255
  - ✓ Records Function ID & Reason Field for each FRS Message
  - ✓ Supports both interrupt and polling models

Separate RefClk Independent SSC (SRIS)
(under development)

Motivation

- Requirement: Need in-box low cost cabling to support existing physical partitionings (e.g. in desktop)
- Challenge: PCIe spec does not support independent clock with spread spectrum
- PCIe base spec changes needed to enable feature:
  1) Requires use of larger elasticity buffer
  2) Requires more frequent insertion of SKIP ordered set
  3) Requires overall system jitter budget modifications

Example of Possible PCIe x2 Cable

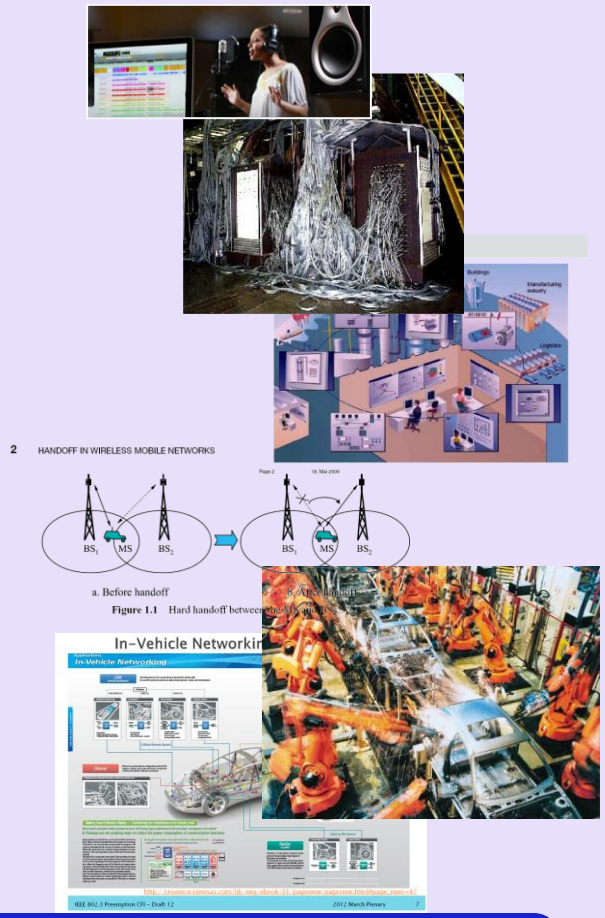


New Terms: SRIS (5600ppm) and Separate RefClk With No SSC – SRNS (600ppm)

Precision Time Measurement (PTM)
(under development)

Motivation

- Precise Time Synchronization is a foundational technology enabling a broad array of applications
  - ✓ Continues to gain momentum in IEEE 1588-2008, 802.1AS, 802.11v …
- Required in many applications:
  - ✓ Synchronized A/V streaming (802.1 AVB)—studios, autos, and home
  - ✓ Instrumentation—distributed data acquisition, logic analyzers
  - ✓ Telecom—cell towers synchronized for seamless handoff
  - ✓ Industrial Automation/control—robots mustn't crash into each other or hit people
  - ✓ Many others…
- PCIe is the ubiquitous "local bus" technology interconnecting IO controllers / Hosts



PCIe Time Synchronization Mechanism Required to "Connect the Time Islands"

PCI Code and ID Assignments Spec & Associated ECNs

Class Code & Capability ID Extraction ECN

Class Code & Capability ID Extraction ECN

- An ECN against the *PCI Local Bus Specification*, Rev 3.0
- Extracts the Class Code definitions from Appendix D
- Extracts the Capability ID definitions from Appendix H
- Enables the consolidation of these definitions into a new standalone document that's easier to maintain
- The new document is the *PCI Code and ID Assignment Specification*
- Consolidating these and other definitions makes them easier to find and manage, and reduces the chance of lost or duplicate assignments
- New IDs for specifications/ECNs, or new Class Codes will trigger updates to the new specification

PCI Code and ID Assignment Specification

PCI Code and ID Assignment Specification

- Consolidates:
  - ✓ Class Code definitions from the *PCI Local Bus Specification* Appendix D
  - ✓ Capability ID definitions from the *PCI Local Bus Specification* Appendix H
  - ✓ Extended Capability ID definitions from the *PCI Express Base* & I/O Virtualization specifications
- Includes:
  - ✓ New Class Code and Capability ID assignments made since the *PCI Local Bus Specification*, Revision 3.0
  - ✓ Some cleanup of formatting & terminology

PCI Code & ID Assignment Specification Update ECN

PCI Code & ID Assignment Specification Update ECN

- A very simple ECN against the new *PCI Code & ID Assignment Specification*
  - ✓ Intentionally chose to separate these “new changes” from the initial version so they could receive proper visibility and review
  - ✓ Wanted the initial version not to include any semantic changes
- Adds a new Programming Interface assignment to Base Class 01h (Storage Controllers) / Sub-Class 08h (Flash Controllers)
  - ✓ 02h – Solid State Storage Controller – NVM Express
- Changes “Flash Controller” references in existing Class Code definitions to “Solid State Storage Controller”
- Adds new “Null Capability” ID for both standard Capabilities and Extended Capabilities
  - ✓ Intended for use by hypervisors that intentionally choose not to expose selected Capabilities in the virtualized Configuration Space for VMs
- Assigns Extended Capability ID 001Ah for Protocol Multiplexing (PMUX)

Additional PCI Code & ID Assignment Spec ECNs

Additional *PCI Code & ID Assignment Spec* ECNs

- Changing Class Code for InfiniBand Adapter ECN
  - ✓ Adds new Sub-Class 07h (InfiniBand Controller) to Base Class 02h (Network Controllers)
  - ✓ Deprecates Sub-Class 06h (InfiniBand) in Base Class 0Ch (Serial Bus Controllers)
- Accelerator Class Code ECN
  - ✓ Defines a new Base Class 12h for processing accelerators
  - ✓ No Sub-Classes or Programming Interfaces are defined
- Note: There's now a more streamlined update process for the *PCI Code & ID Assignment Specification*
  - ✓ No ECN is required for simply assigning new codes or IDs
  - ✓ The Errata Process can be used for making simple clarifications or editorial fixes
  - ✓ The ECN Process is still required for making non-errata changes to existing text

Thank you for attending the PCI-SIG Developers Conference Europe 2012

For more information please go to www.pcisig.com