PCI TOI

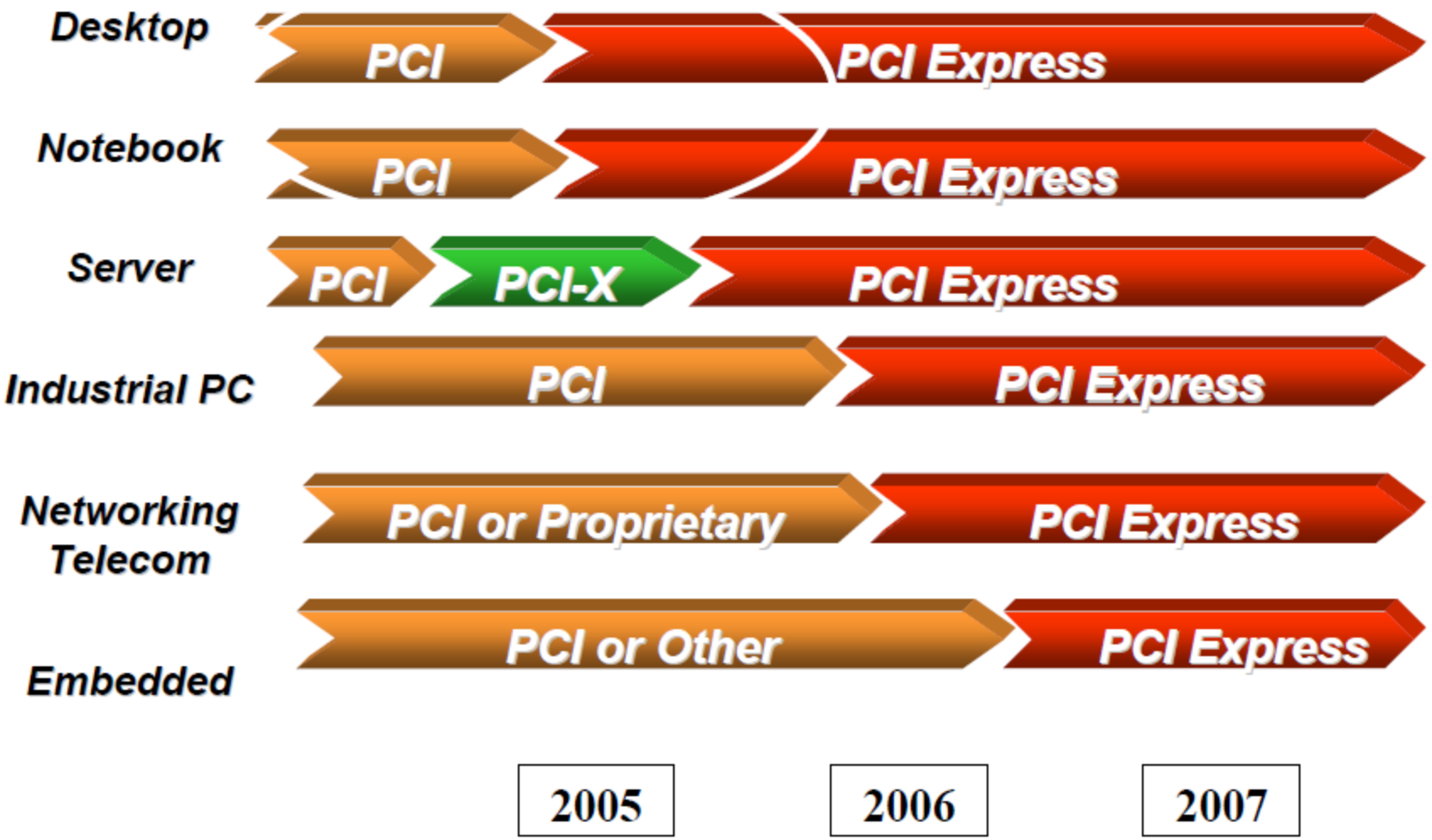
Compiled by Oliver Yang

Nov, 2017

Agenda

- **PCI/PCIe Basic**
- **PCIe errors handling**
- **Case study - analysis PCIe errors**
- **References**

Transition from PCI/PCI-X to PCIe



PCI Throughput

- Bandwidth(Gbps) = Frequency(Mhz) x bit(Bytes)

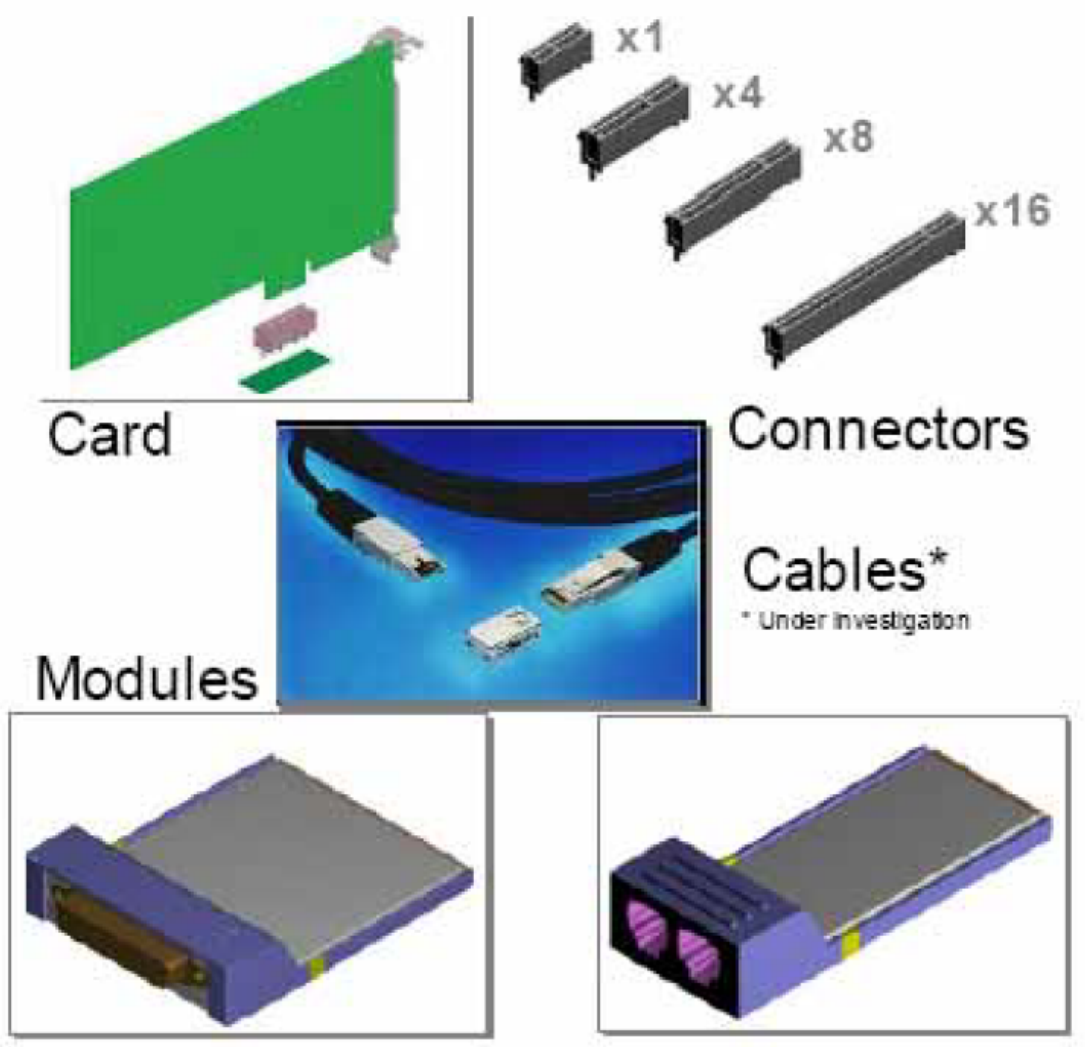
PCI / PCI-X	32bit 33Mhz	32bit 66Mhz	64bit 66Mhz	64bit 133Mhz
Bandwidth	1.056 Gbps	2.112 Gbps	4.224 Gbps	8.512 Gbps

PCIe Throughput

Link Bandwidth	Raw Bit Rate x1	x1	x8	x32
PCIe 1.x	2.5 GT/s	2Gbps	16Gps	64Gps
PCIe 2.x	5.0 GT/s	4Gbps	32Gps	128Gps
PCIe 3.x	8.0 GT/s	8Gbps	64Gps	256Gps

- GT/s vs. Gbps
  - PCIe 1.x and 2.x use 8b/10b encoding
  - PCIe 3.x uses 128/130 encoding.
- Bandwidth(Gbps) = Per lane(Gbps) x Number of Lane
- Real bandwidth is much slower because of overhead of PCIe itself
  - One x8 PCIe 2.x slot only could hold dual ports 10G NIC
  - One x8 PCIe 3.x slot only could hold four ports 10G NIC

PCI Express Mechanicals

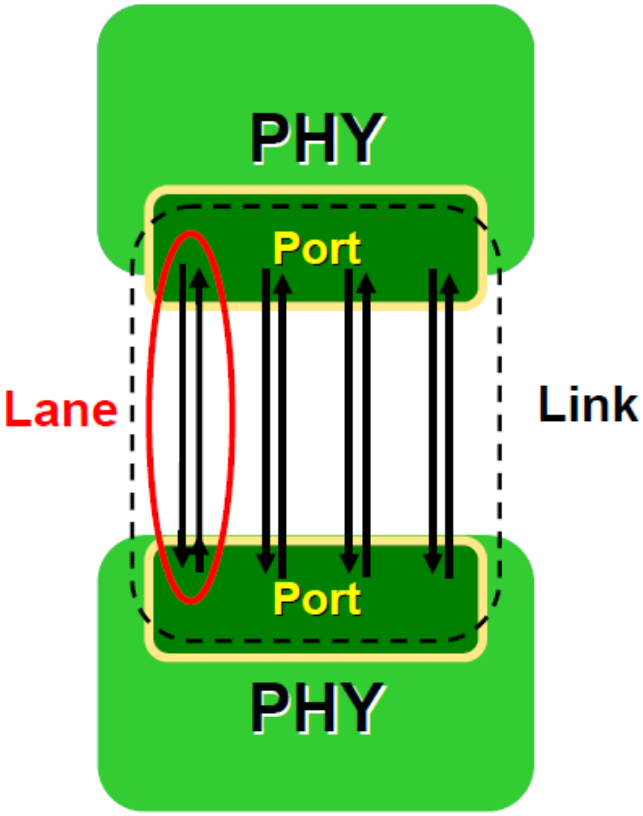


PCI vs. PCIe

- PCI uses a shared parallel bus
  - Bandwidth shared between devices on the bus
  - Only one device may own the bus at any time
  - Large number of parallel signals
  - Wait states may be added by Initiator or Target
- PCI-Express uses a serial point to point interconnect
  - Full bandwidth dedicated to that link
  - No need for arbitration
  - Low number of signals
  - No wait states

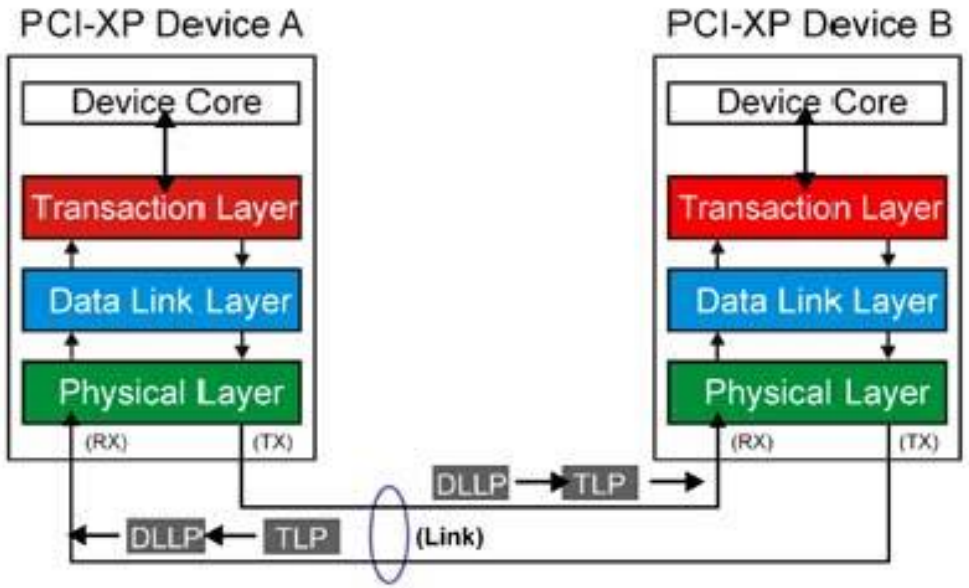
PCIe Link

- Port
  - A group of transmitters and receivers located on the same chip that define a link
- Lane
  - A set of differential signal pairs, one pair for transmission and one pair for reception
- Link
  - A dual-simplex communications path between two components
  - A xN link is composed of N lanes



Example: 4 Lanes

PCIe – Packet Based Protocol



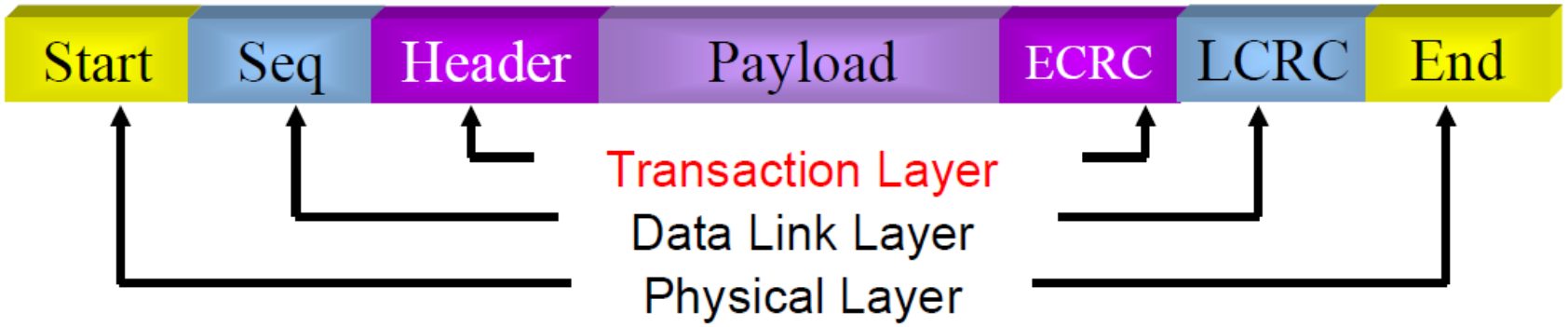
Link Transaction Layer Packet (TLP)

Data Link Layer Packet (DLLP)

DLLP Types:
TLP Ack/Nak
Power Management
Link Flow Control
Vendor-Specific

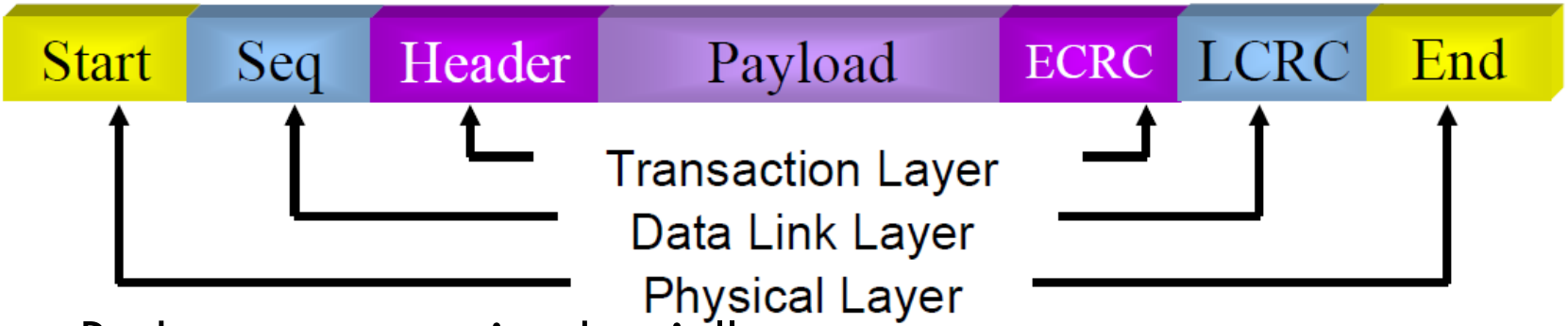
TLP Types:
Memory Read-Unlocked
Memory Read-Locked
Memory Write
IO Read
IO Write
Configuration Read-Type 0,
Configuration Write-Type 0,
Completion-Unlocked
Completion-Locked
Completion w/data-Unlocke
Completion w/data-Locked
Message Request
Message Request w/data

Transaction Layer



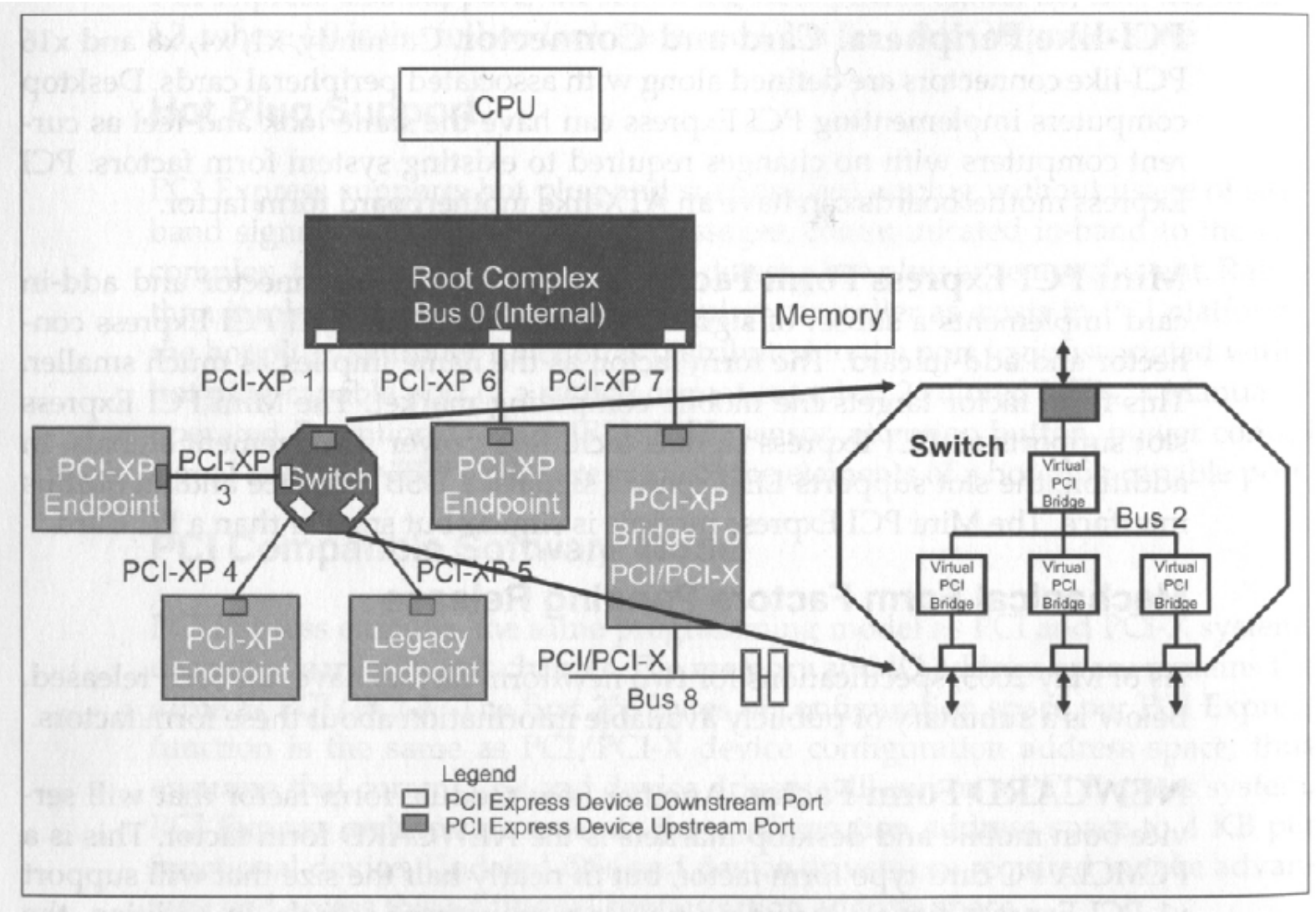
- Responsible for
  - Post and Non-Posted transactions
    - ✓ Memory read/write, IO read/write, Configuration read/write etc.
  - Managing link flow control
  - Enforcing ordering and Quality of Service(QoS)
  - Power management control/status
- Header Information may include
  - Address/Routing
  - Data transfer Length
  - Transaction descriptor
- End to End CRC checking provides additional security(optional)

Physical Link and Data Link Layers



- Packets are transmitted serially
- Credit based Flow Control
  - Each device transmits periodic flow control packets to inform transmitter of buffer space
  - Ensures that receiving device has buffer space for the packet
- ACK/NAK ensure correct delivery of data
- Different link status for power management
- Can re-train the link in the event of catastrophic failure
  - Only one or some of the lanes may be corrupted
  - Retraining may allow fall back to lower bandwidth system.

PCIe Fabric

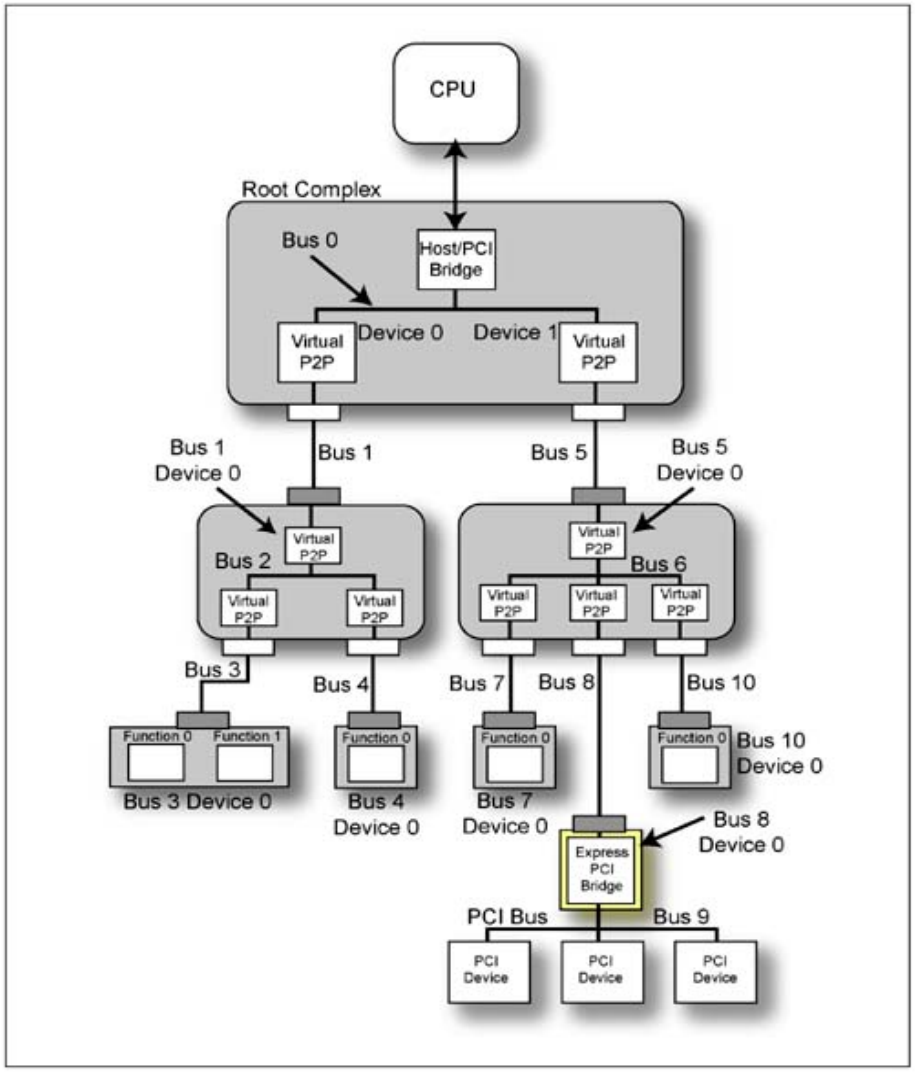

CPU
Root Complex
Bus 0 (Internal)
Memory
PCI-XP 1
PCI-XP 6
PCI-XP 7
PCI-XP Endpoint
PCI-XP 3
Switch
PCI-XP Endpoint
PCI-XP Bridge To PCI/PCI-X
PCI-XP 4
PCI-XP 5
PCI-XP Endpoint
Legacy Endpoint
PCI/PCI-X
Bus 8
Switch
Virtual PCI Bridge
Bus 2
Virtual PCI Bridge
Virtual PCI Bridge
Virtual PCI Bridge
Legend
PCI Express Device Downstream Port
PCI Express Device Upstream Port

PCIe Terminologies

- RC (Root Complex)
  - A defined PCIe System Element that includes
    - ✓ A Host Bridge
    - ✓ Zero or more Root Complex Integrated Endpoints
    - ✓ Zero or more Root Complex Event Collectors (Support RC EPs)
    - ✓ One or more Root Ports.
- Root Port
  - A PCI Express Port on a Root Complex that maps a portion of the Hierarchy through an associated virtual PCI-PCI Bridge.
- Switch
  - A defined System Element that connects two or more Ports to allow Packets to be routed from one Port to another.
  - To configuration software, a *Switch appears* as a collection of virtual PCI-to-PCI Bridges.
- EP (Endpoint)
  - A Function that has a Type 00h Configuration Space header.

Domain:Bus:Device.Function

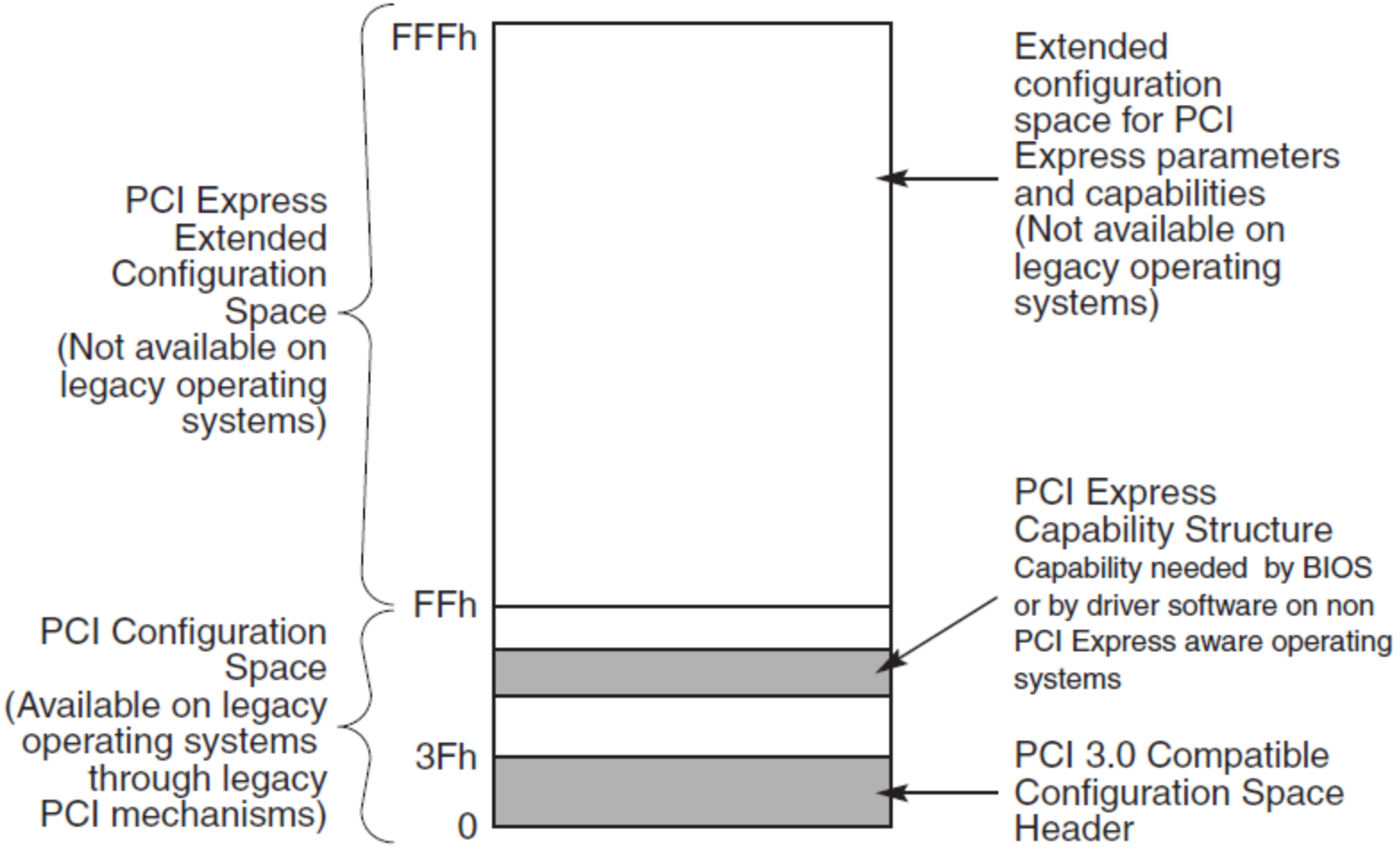
- Domain
  - Address range[0-65535]
  - Each Domain have its separate bus range [0-255]
  - x86 only have one PCI Domain: 0
- Bus
  - Address range [0-255]
  - Devices from RC is attached to bus 0
  - x86 could have multiple RCs, but all of RCs shared [0-255] bus range.
- Device
  - Address range [0-31]
- Function
  - Address range [0-7]



PCI Address spaces

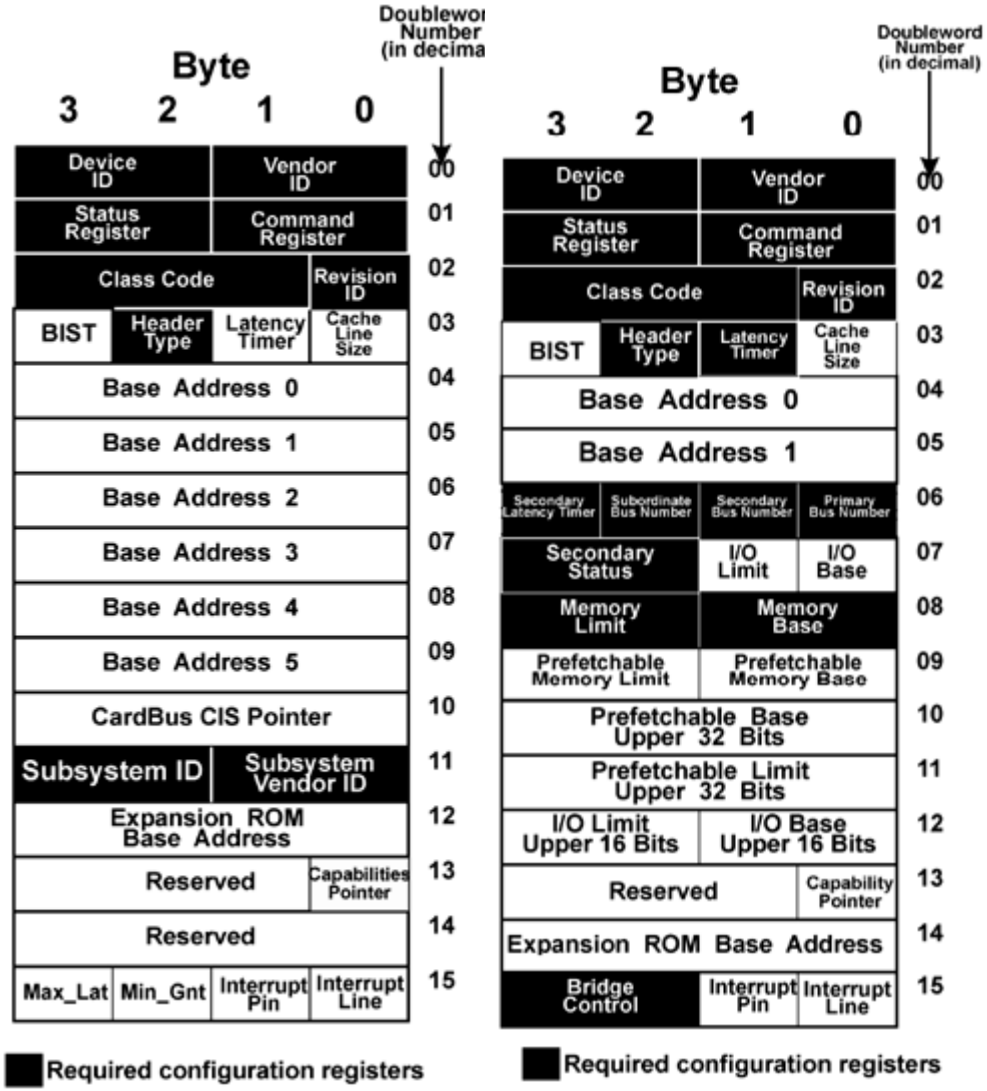
- Address spaces
  - Configuration Space (Accessed by Bus/Device/Function)
    - ✓ Configuration Access Mechanism#1
      - Legacy in/out to IO ports 0xCF8 and 0xCFC.
    - ✓ MMIO (memory mapped IO)
      - ACPI MCFG table provide the base address.
  - Memory Space (Accessed mov instructions)
    - ✓ Pointed by 32bit or 64bit BARs which are initialized by BIOS or OS
  - IO Space (Accessed by in/out instructions)
    - ✓ Pointed by IO BARs which are initialized by BIOS or OS
    - ✓ Legacy space with size limitations.
- Usages
  - Config Space is standard for OS device enumeration and management.
  - Memory and IO Spaces are device private

4K PCIe Config Space Layout



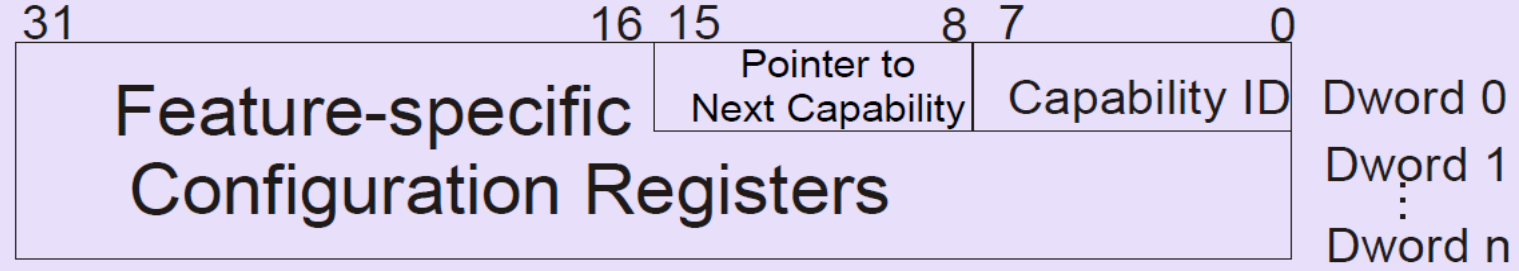
PCI Config Space [0 - 63]

- Two header types
  - Type 0 - for regular EPs
  - Type 1 - for switches and bridges
- Device Identification
  - VendorID: PCI-SIG assigned
  - DeviceID: Vendor self-assigned
  - Subsystem VendorID: PCI-SIG
  - Subsystem DeviceID: Vendor
- Address Decode controls
  - Software reads/writes BARs to determine required size and maps appropriately
  - Memory, I/O, and bus-master enables



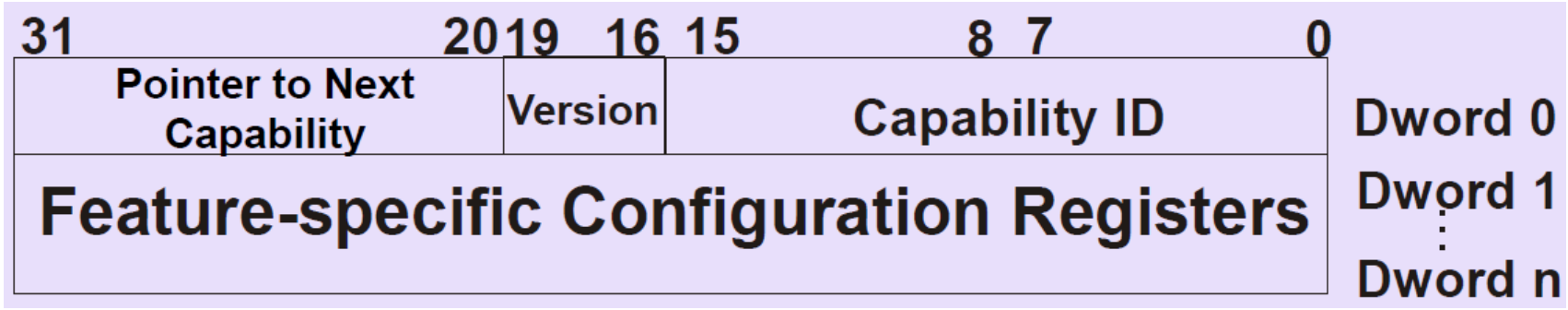
PCI Config Space [63 - 255]

- Linked list
  - Follow the list! Cannot assume fixed location of any given feature in any given device
  - First entry in list is stored at offset of 0x34h in PCI config space.
  - Features defined in their related specs:
    - ✓ PCI-X
    - ✓ PCIe
    - ✓ PCI Power Management
    - ✓ Etc…

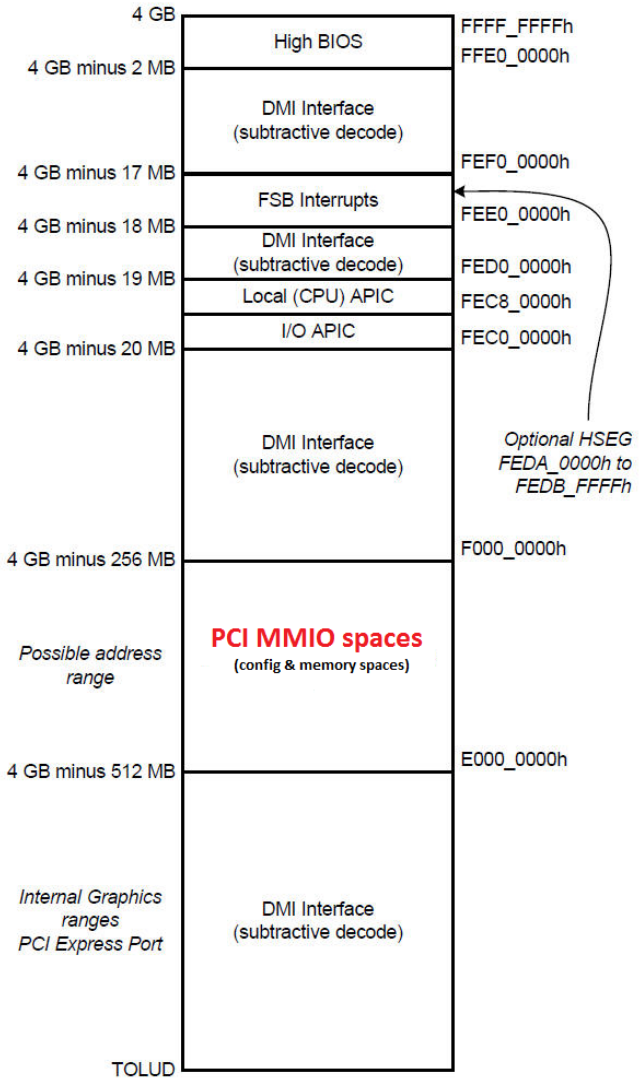


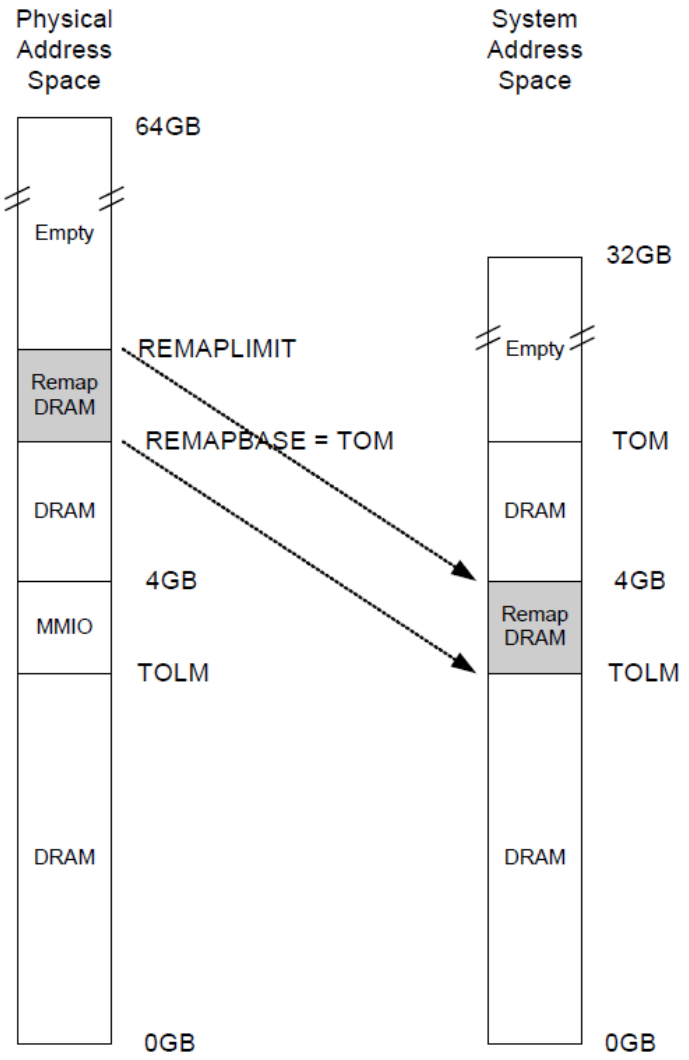
PCI Config Space [256 - 4095]

- PCI Express only
- Linked list
  - ✓ Follow the list! Cannot assume fixed location of any given feature in any given device
  - ✓ First entry in list is *always* at 100h



PCI Hole and DRAM Remap

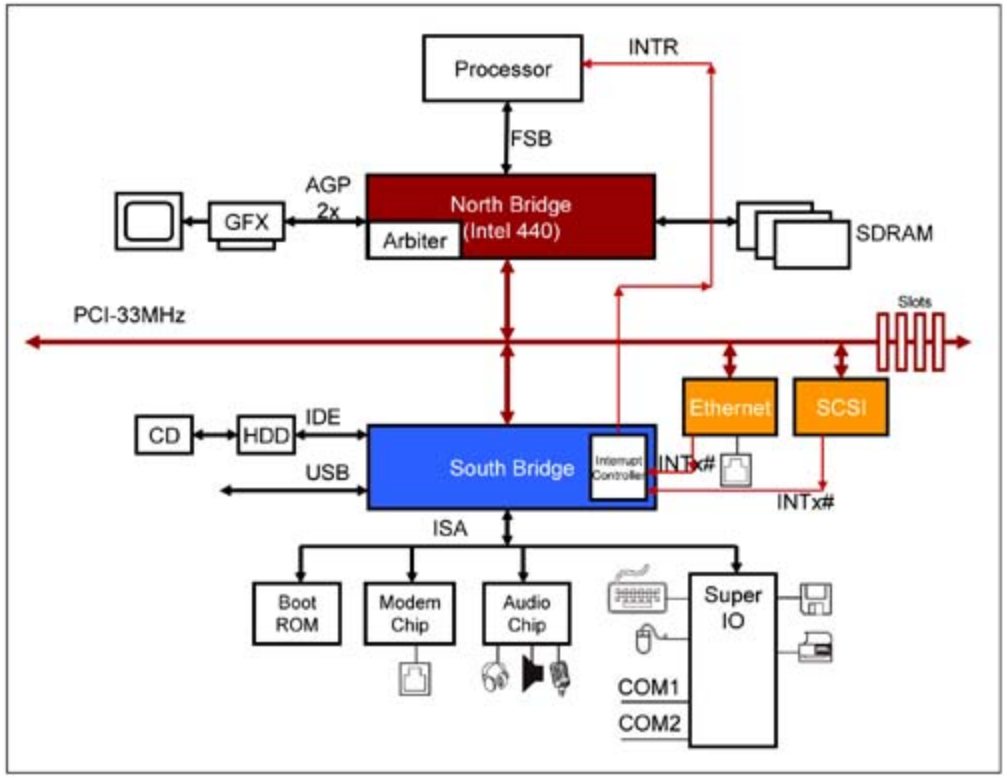




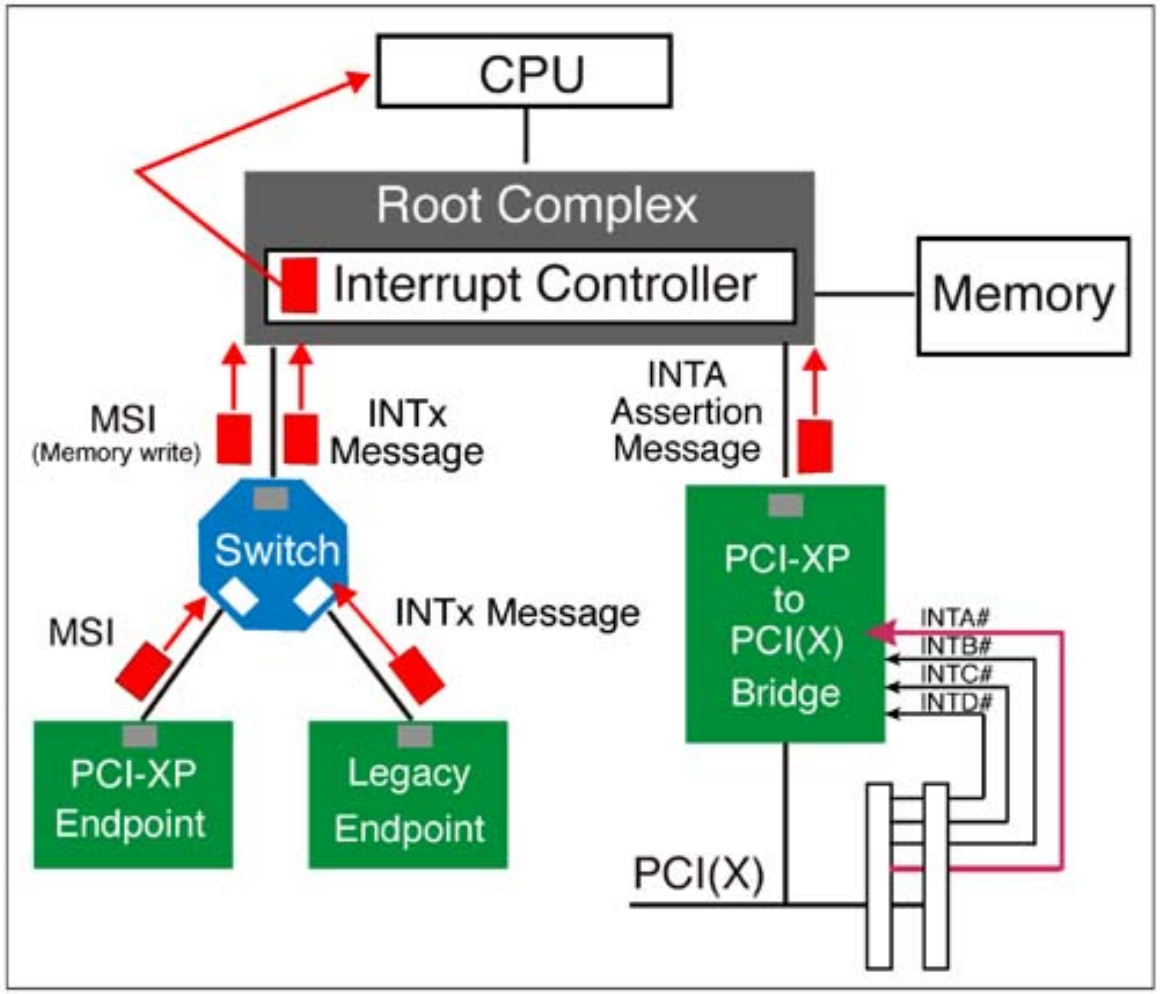
PCIe INTx/MSI/MSI-X Differences

- INTx
  - PCIe defined the INTx messages to emulate INTx (INTA/B/C/D pins) assertions
  - On x86, INTx messages will target to IOAPIC in PCH/South Bridge
  - On x86 IOAPIC sends a Intel defined interrupt message to North Bridge
  - LAPIC within the CPU die will receive INTx eventually
- MSI
  - Each PCIe functions config space has a PCIe MSI capability structure which allow up to 32 MSI vectors. All of 32 vectors share a same MSI address and MSI data pair, which means all 32 vectors target a same CPU core.
  - MSI address and MSI data are initialized by OS interrupt vector allocation code
  - PCIe device write the MSI data to MSI address by using a PCIe DMA write
  - On x86, MSI address is the LAPIC address
  - On x86, MSI data a Intel defined format, which defined interrupt delivery mode.
- MSI-X
  - Similar with MSI, but MSI-X capability structure allows up to 2048 vectors, and each MSI-X vectors defines the separate MSI address and data pairs, which could target to different CPU cores.

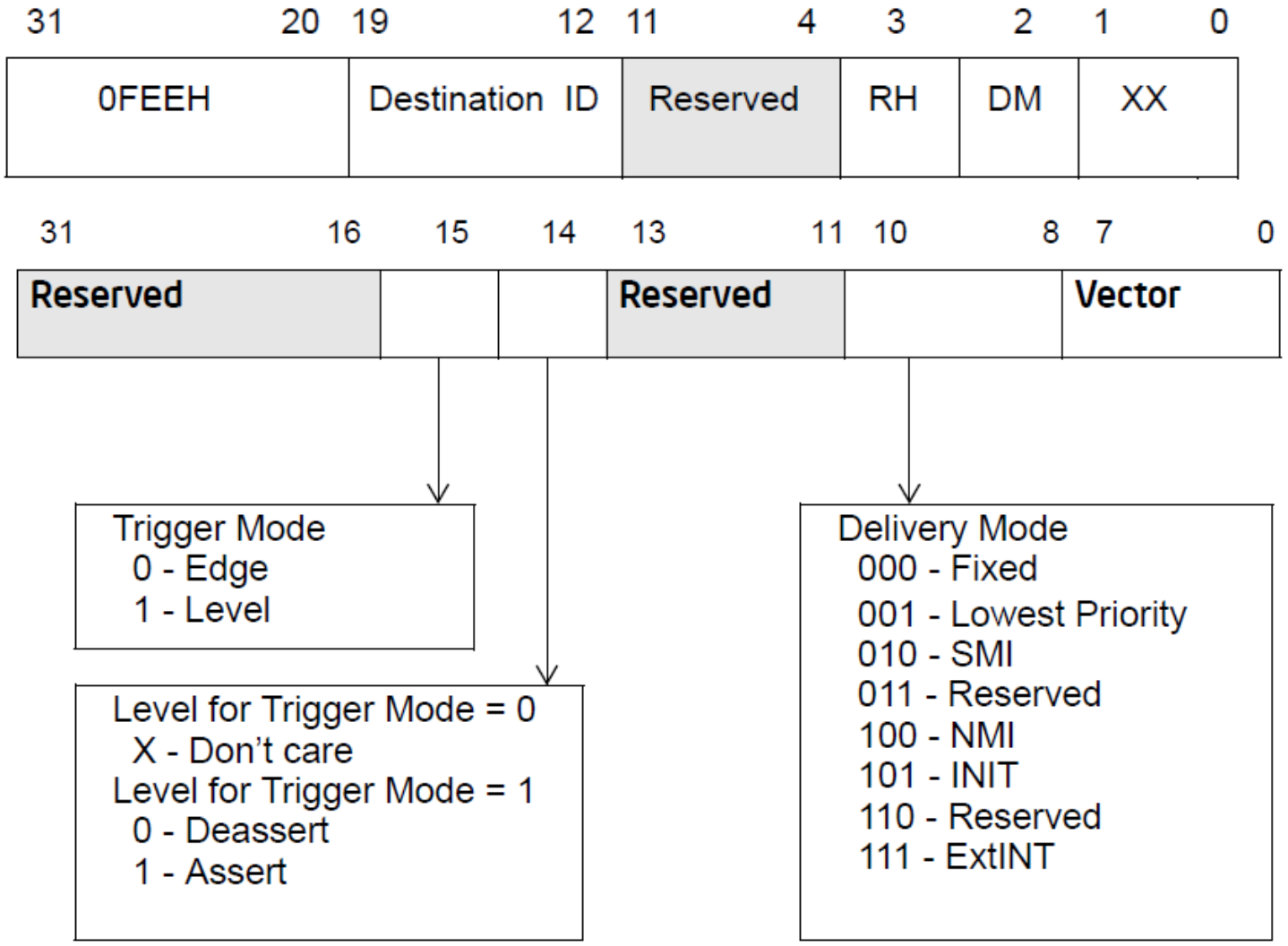
Legacy PCI Interrupt on x86



Message Based PCIe Interrupts



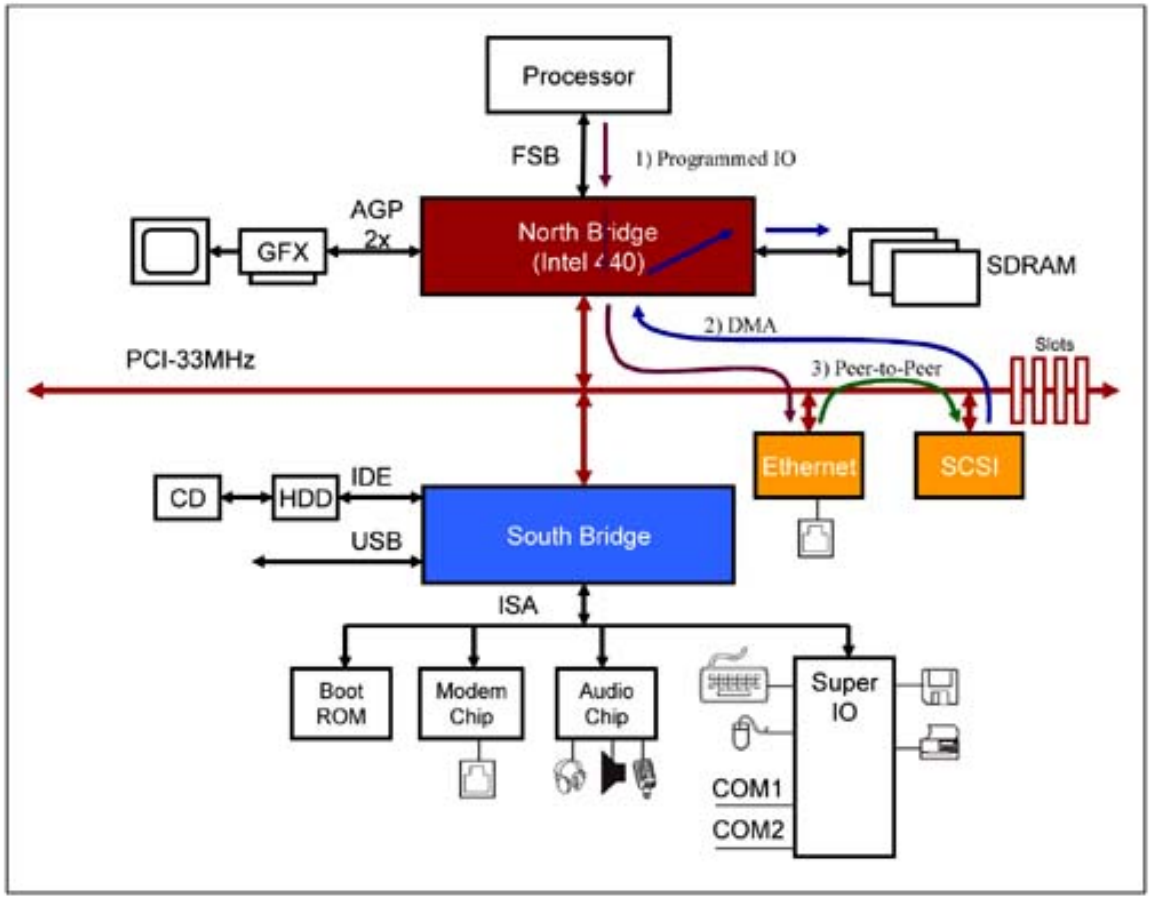
MSI Address and Data on x86



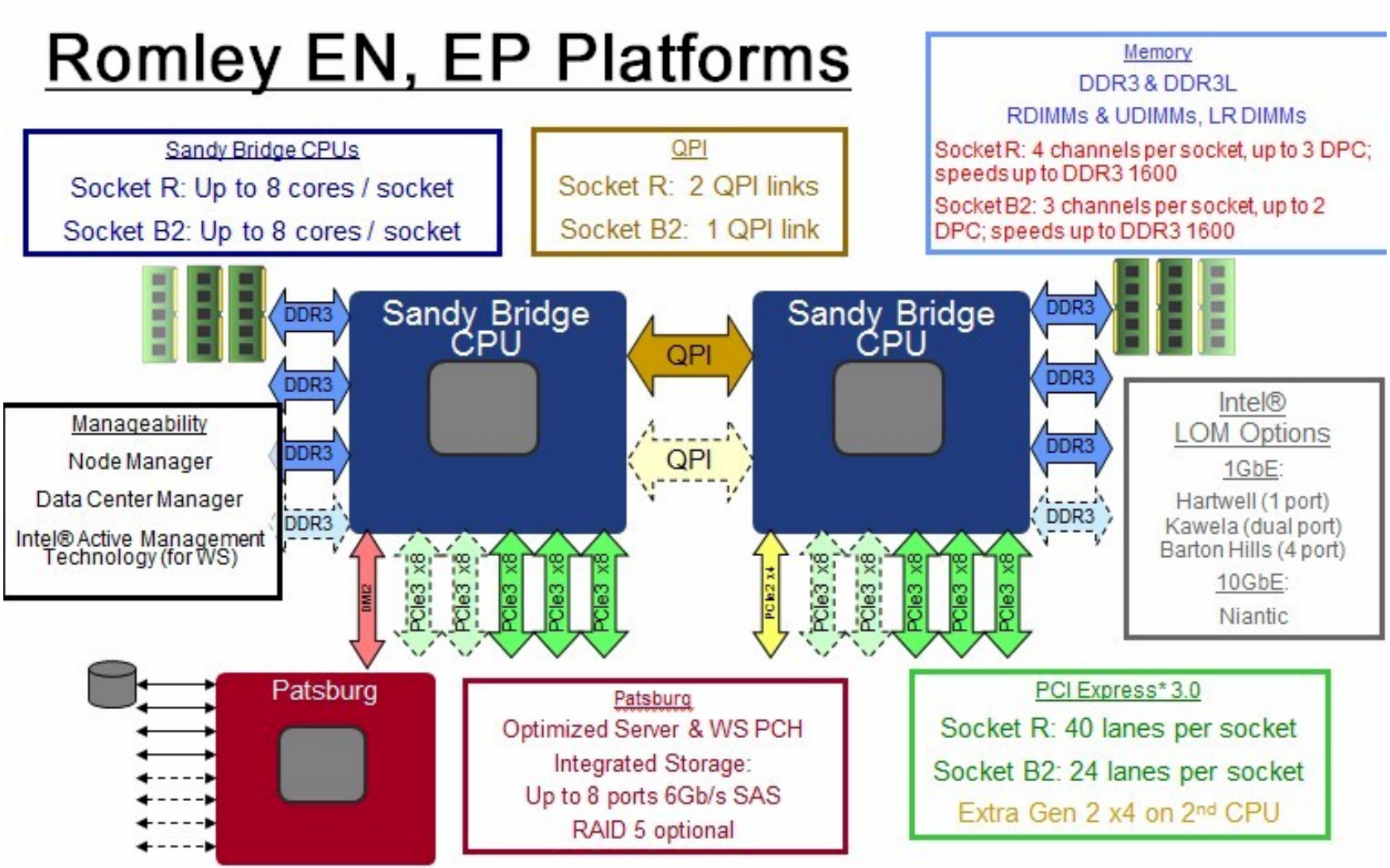
x86 DMA Transactions in PCIe

- There are 4 possible DMA transactions
  - Device memory to host memory
    - ✓ PCIe DMA write transaction, memory write TLP
  - Host memory to device memory
    - ✓ PCIe DMA read transaction, memory read TLP
  - Device memory to device memory
    - ✓ PCIe peer to peer
  - Host memory to host memory
    - ✓ Not PCIe related.
    - ✓ Supported by Intel CPU uncore DMA engine since Nehalem
- PCIe DMA
  - PCIe devices are bus master devices who can start a DMA transaction
  - Each PCIe devices might have multiple built-in DMA engines
  - On x86, without IOMMU support, DMA address is equal to MMU physical memory address
  - If IOMMU enabled, a DMA address is an IO address. IOMMU translate an IO address to a MMU physical address

PCI Bus Transactions



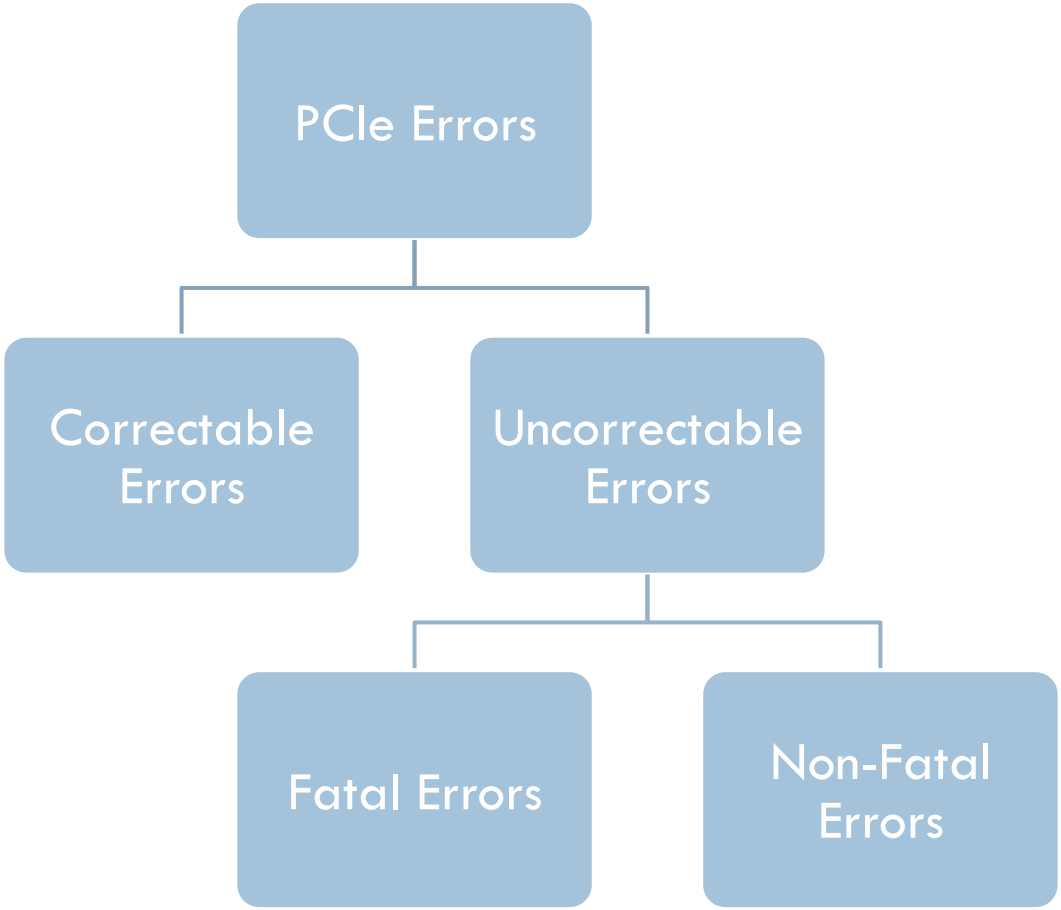
Sandy Bridge Board



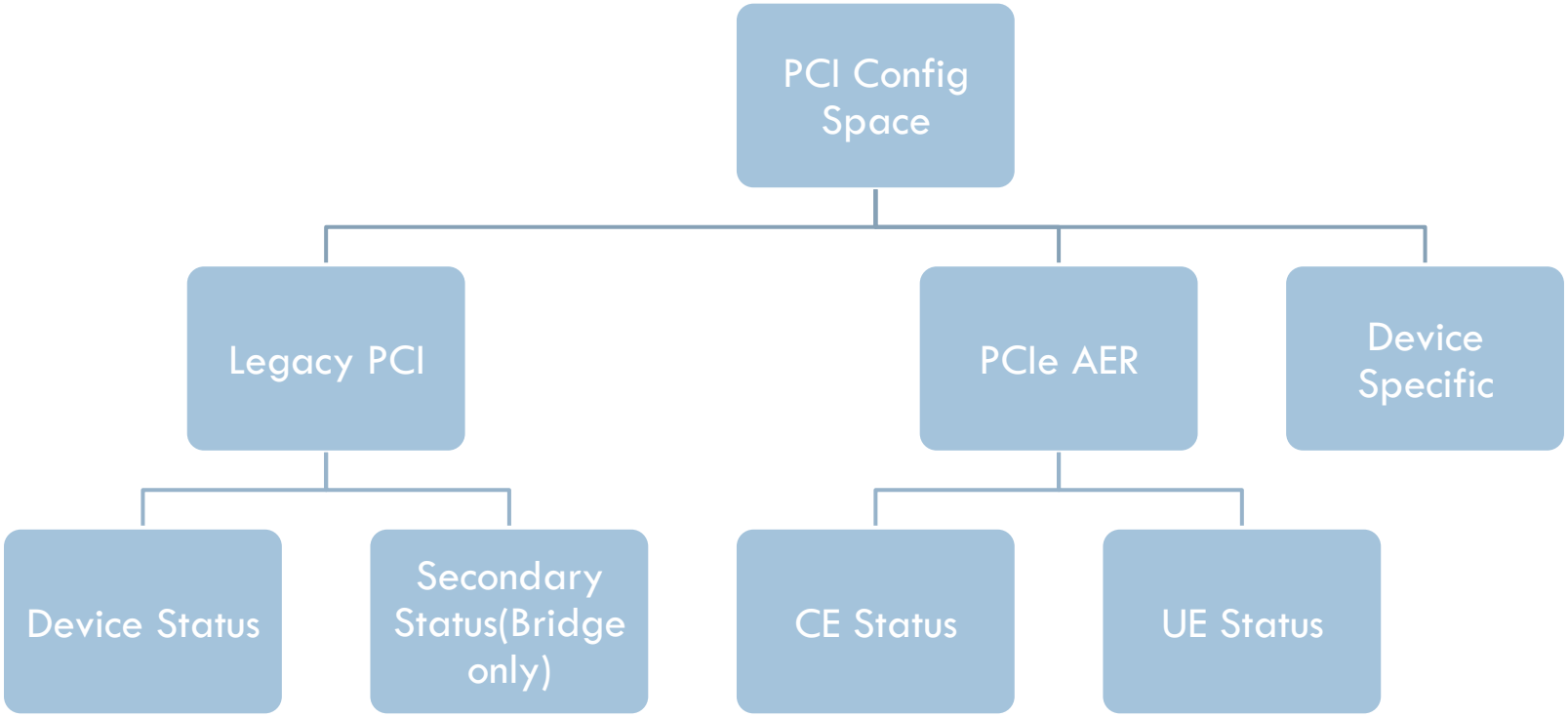
Agenda

- PCI/PCIe Basic
- PCIe errors handling
- Case study - analysis PCIe errors
- References

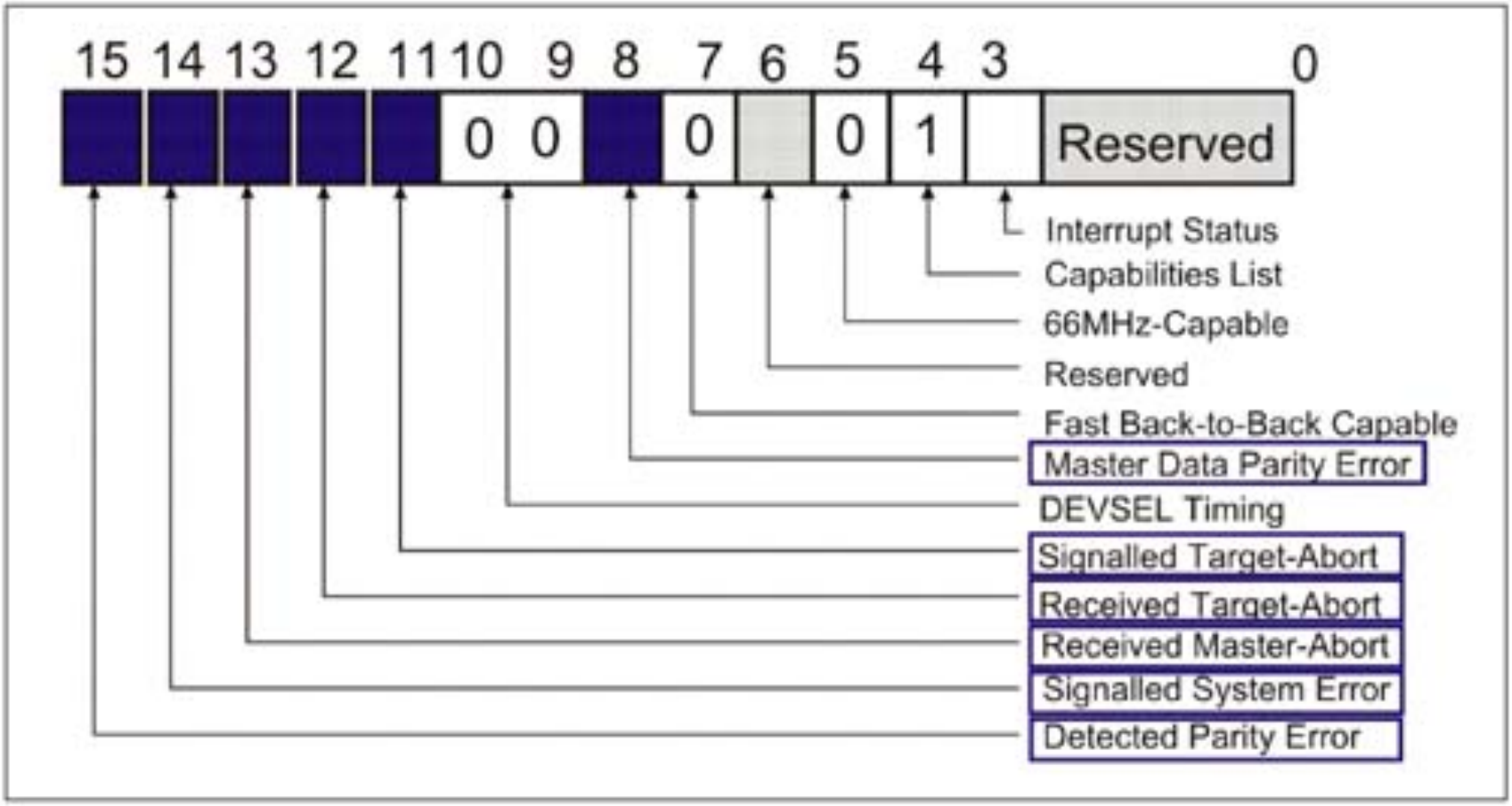
PCIe Errors Classification



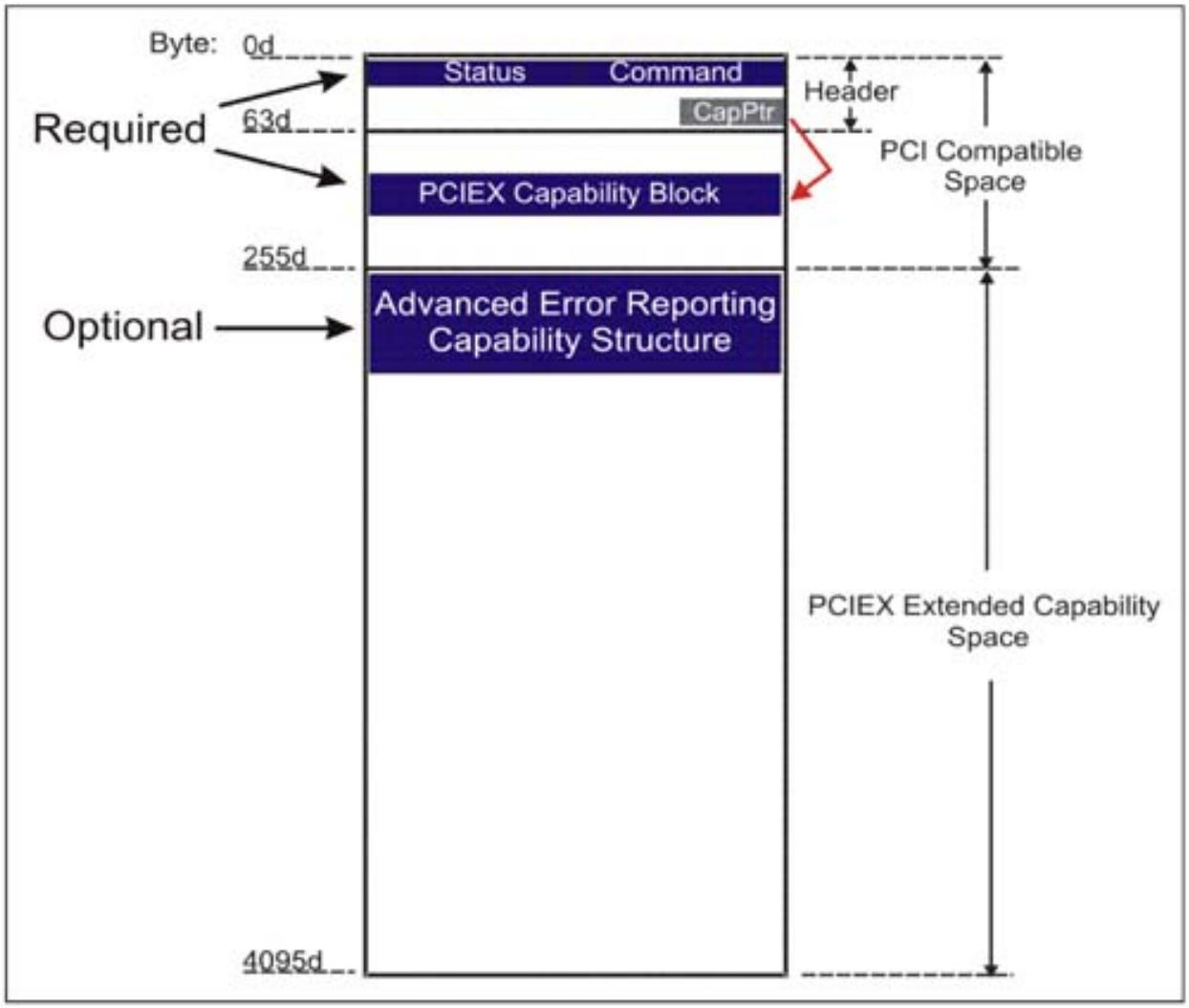
PCIe Errors Registers in Config Space



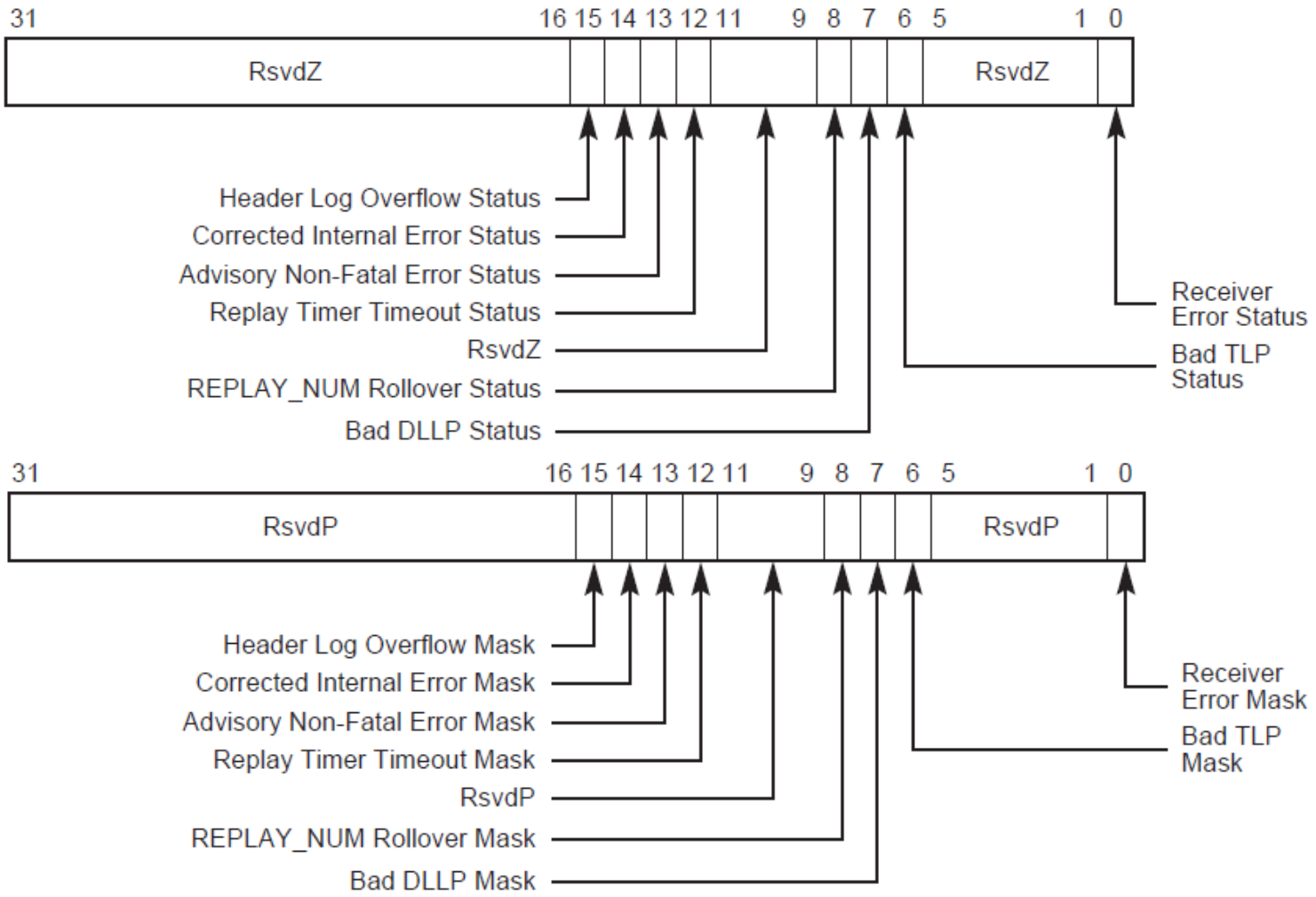
PCI Device Status Register



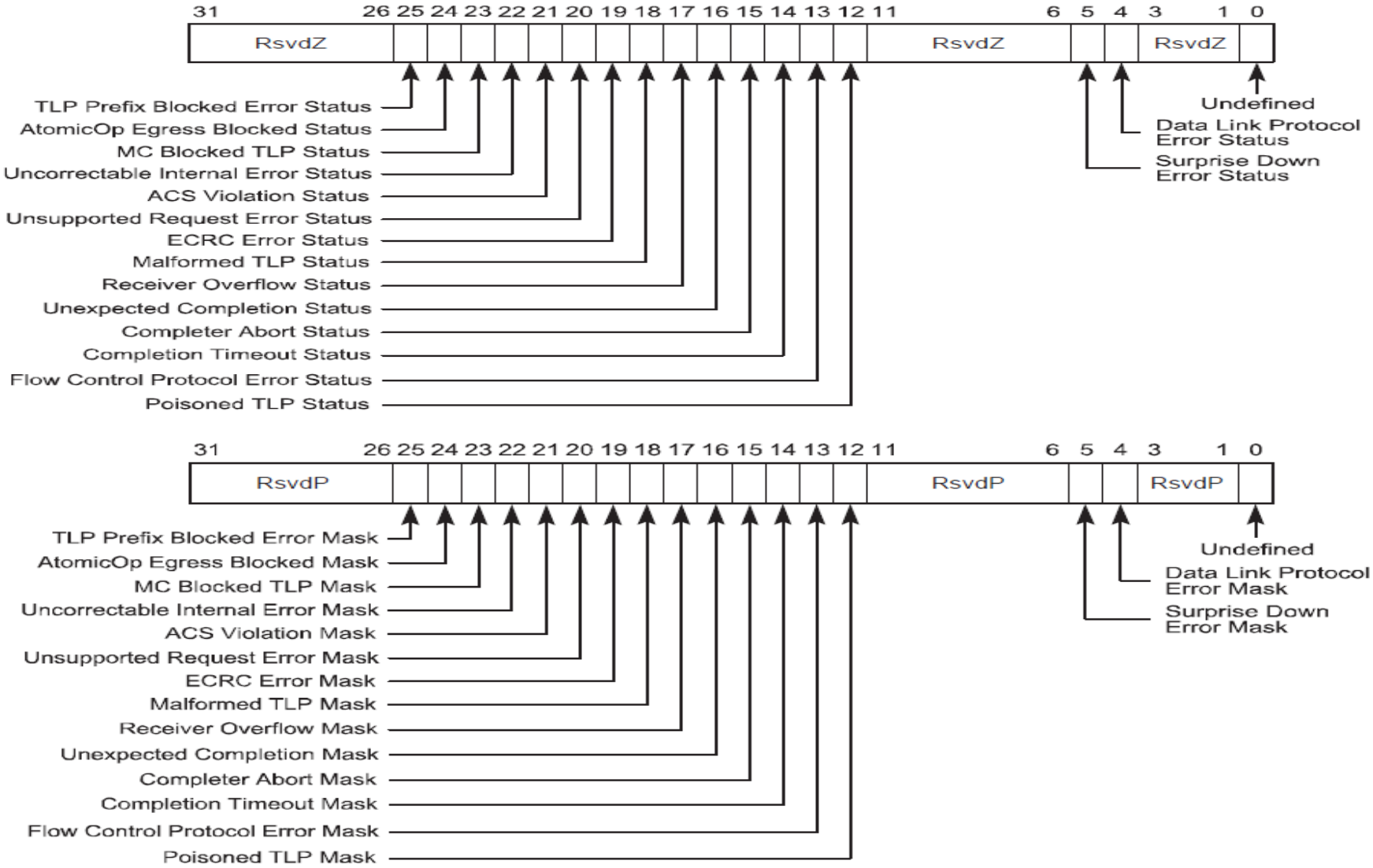
PCIe AER(Advance Error Reporting) Registers



AER CE Status and Mask Registers

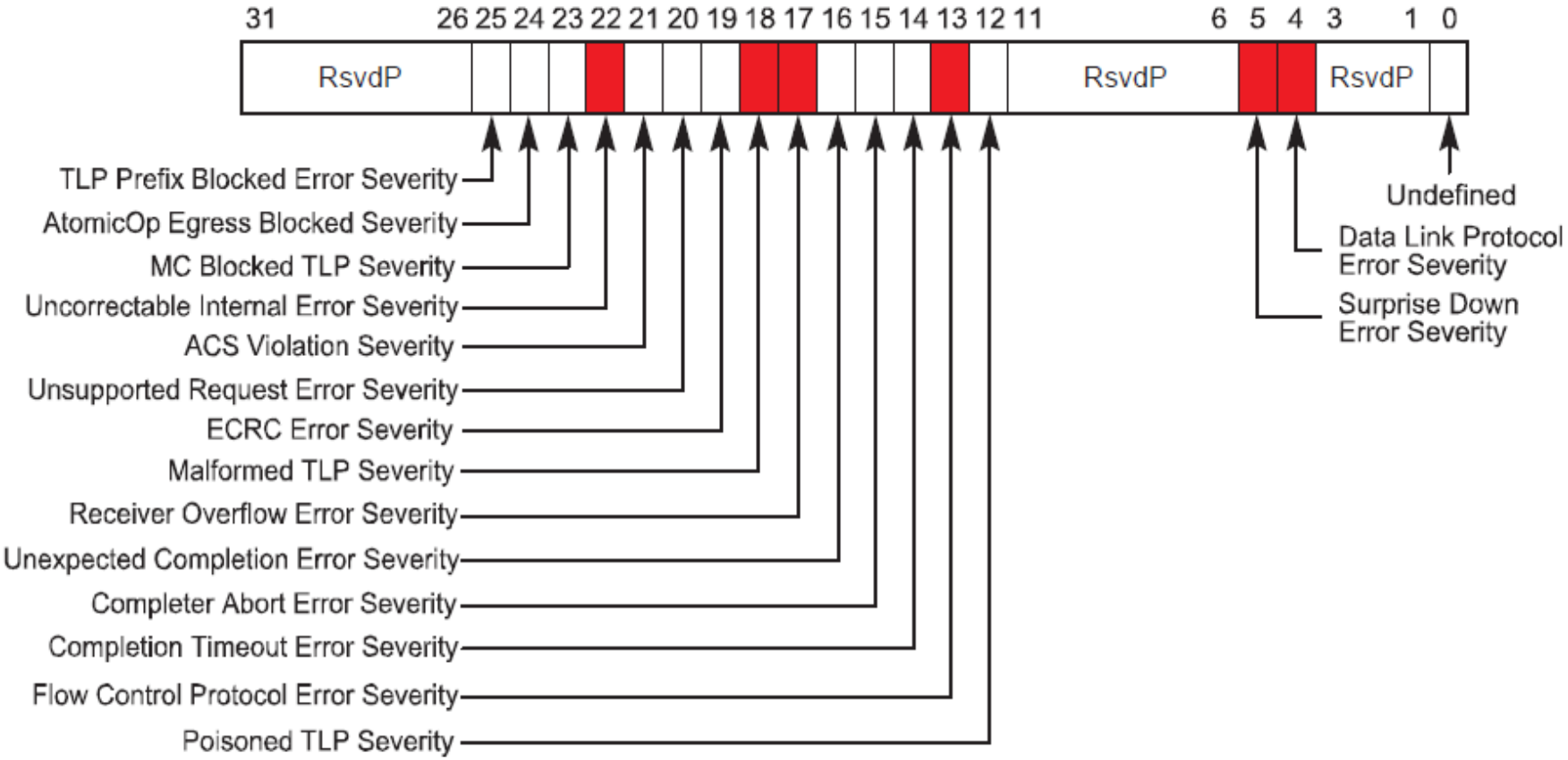


AER UE Status and Mask Registers



AER UE Severity Registers

- Red color indicates the Fatal UEs

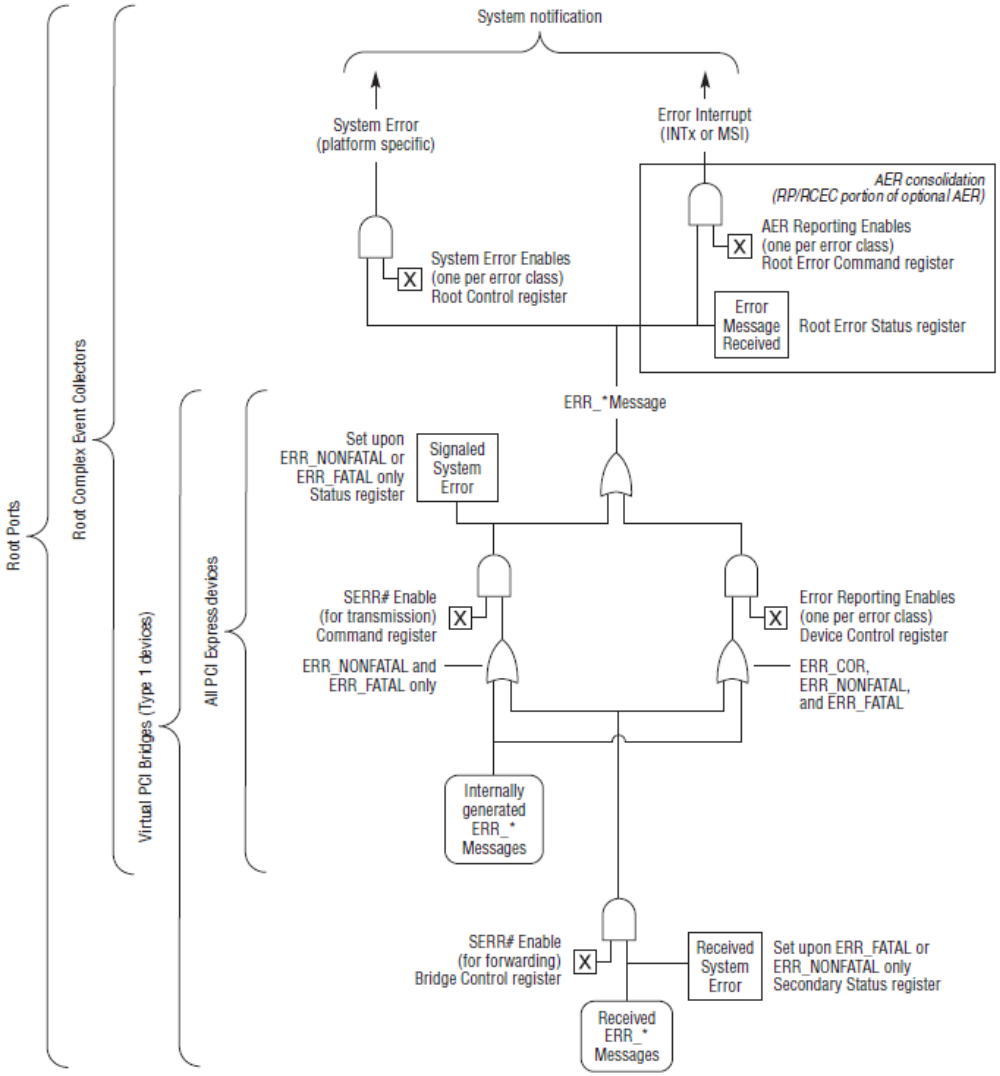


PCIe Error Signaling

- PCIe error signaling
  - Completion Status other than success: UR, CA, CRS
  - Error Messages: ERR_COR, ERR_NONFATAL, ERR_FATAL
  - Error Forwarding, aka Data Poisoning
- Error Interrupts
  - System interrupts
    - SMI - Handled by BIOS SMI interrupt handler
    - NMI – Handled by OS NMI interrupt handler
      - ✓ x86 standard NMI
      - ✓ Vendor customized NMI
  - Root Port device interrupts
    - ✓ MSI or INTx handled by OS PCIe AER bus driver

PCIe error signaling control

- PCIe errors signaling could be controlled at different levels
  - EP, Switch, Root Port
    - ✓ AER CE/UE mask registers
    - ✓ #SERR enable in PCI command register
    - ✓ Device control registers in PCIe capability
  - Root Port only
    - ✓ Root Error Command register
    - ✓ Root Control register



PCIe Error Handling on Linux

- PCIe AER bus driver handles all of PCIe CEs and UEs
  - Negotiate with BIOS by ACPI _OSC method and get error handling control
  - Disable SMI and register per Root Port MSI or INTx interrupts handlers.
  - Handled errors interrupts with below strategies
    - ✓ CE - Error logging and clear the registers
    - ✓ Non-fatal UE
      - Disable all drivers associated within the device hierarchy
      - Error logging and registers dump
      - Call driver callbacks to reset the PCIe devices
      - Enable all affected drivers again if recover success
    - ✓ Fatal UE
      - Disable all drivers within the device hierarchy
      - Error logging and registers dump
      - Call driver callbacks to log errors, and reset PCIe links, devices
      - Enable all affected drivers again if recover success

Agenda

- PCI/PCIe Basic
- PCIe error handling
- Case study - analysis PCIe errors
- References

Kernel panic for PCIe UEs – part I

- On a x86 box, below command could trigger a kernel panic

  # dd if=/dev/mem of=/dev/null

- The panic just means the system is panicked by HW errors, which could be either a HW fault or a software(driver) bug.
- Look at the SEL logs to understand what HW errors are detected
  - It was the completion abort PCIe UEs reported by device 84:0.0

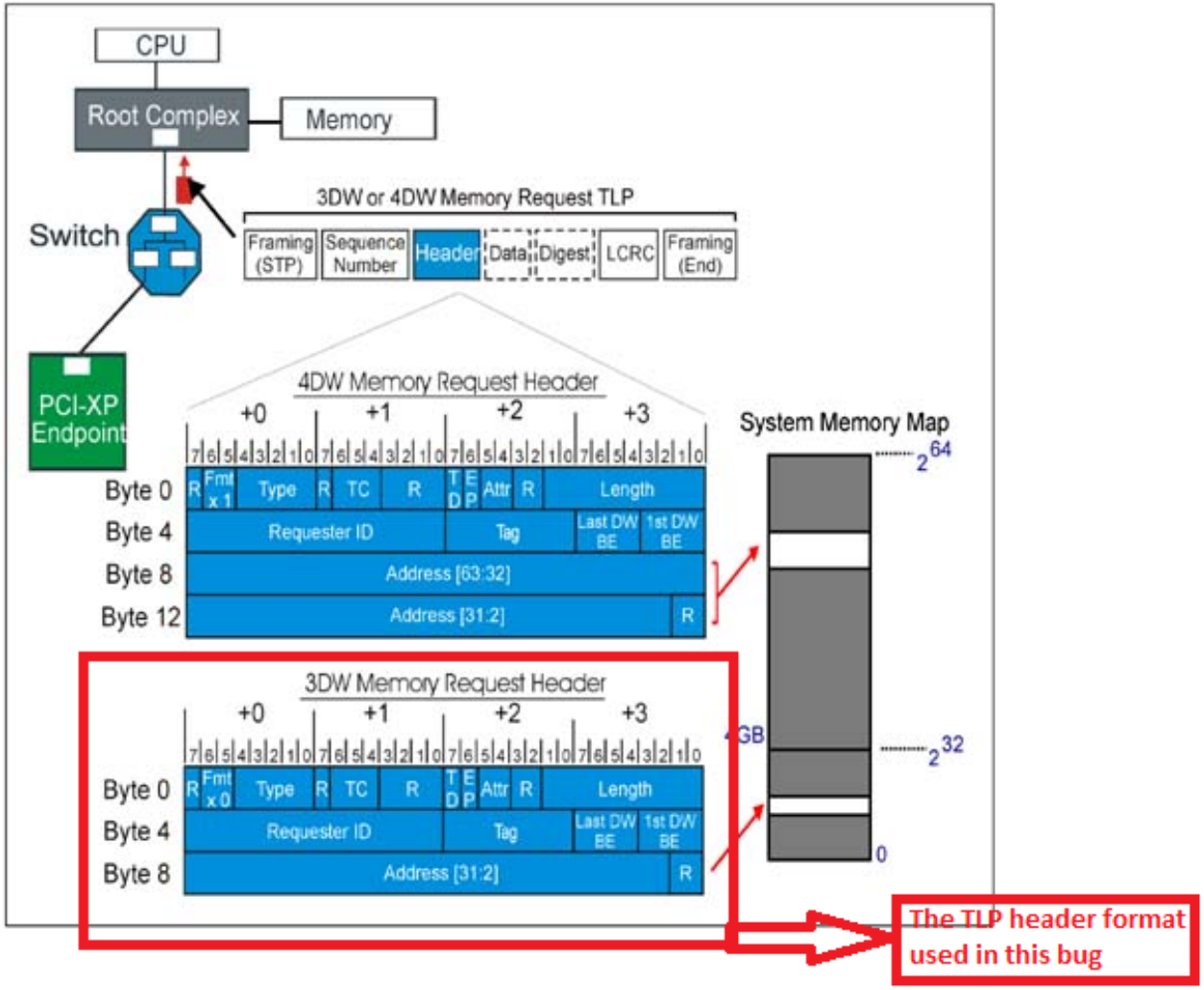
    # grep HL bios.txt

    ELOG(109) -N- Completer Abort Status Bus:84H Dev:00H Fn:00H SID:FFFFFFFFH FEP:0FH HL:D0CF1D27CE200000800808FF00000002H
  - The HL field in SEL log is PCIe TLP header log dump, which is actually from the TLP packet that caused the error.
  - Now decode the header log: D0CF1D27CE200000800808FF00000002
    - ✓ 00000002 is the first 4(0-3) bytes of TLP header which tells us this TLP is a Memory Read Request TLP, and packet header has 3DW length(12bytes). So the D0CF1D27 is useless.
    - ✓ 800808FF tells us the PCIe TLP Requester(device) ID is 80:1.0 which is the PCIe Root Port.
    - ✓ CE200000 is 32bit memory address according to Memory Read Request TLP packet format.
    - ✓ Run lspci, we found that CE200000 is the start address in PLX switch memory(BAR0).

      # lspci -s 84:0.0 -vvv | grep -i reg

      Region 0: Memory at ce200000 (32-bit, non-prefetchable) [size=128K]
    - ✓ Then we know PLX switch(84:0:0) send PCIe Root Port(80:1.0) a completion abort TLP, indicate the its device register read failure happened at memory address CE200000
  - This HW failure should be HW/FW problems, as PLX bar0 is not owned by Linux OS.

Kernel panic for PCIe UEs – part II



Agenda

- PCI/PCIe Basic
- PCIe error handling
- Case study - analysis PCIe errors
- References

References for this TOI material

- PCI Express System Architecture (Book)
- PCI Express Base Specification Revision 3.0
- PCI Local Bus Specification Revision 3.0
- The PCI Express Advanced Error Reporting Driver Guide HOWTO

Q&A