PCI-SIG®
Architecture Overview

Richard Solomon
LSI Corporation

What's all this PCI stuff anyway?

- Presentation will cover basic concepts and their evolution from PCI™ through PCI-X™ to PCI Express®
  - ✓ Specs written assuming designers have these key background concepts
  - ✓ High level overview of PCI, PCI-X, PCI Express, and I/O Virtualization
  - ✓ Brief description of compliance program

PCI Background

Revolutionary AND Evolutionary

- PCI
  - ✓ Revolutionary
    - Plug and Play jumperless configuration (BARs)
    - Unprecedented bandwidth
      - 32-bit / 33MHz – 133MB/sec
      - 64-bit / 66MHz – 533MB/sec
    - Designed from day 1 for bus-mastering adapters
  - ✓ Evolutionary
    - System BIOS maps devices then operating systems boot and run without further knowledge of PCI
    - PCI-aware O/S could gain improved functionality

Revolutionary AND Evolutionary

- PCI-X
  - ✓ Revolutionary
    - Unprecedented bandwidth
      - Up to 1066MB/sec with 64-bit / 133MHz
    - Registered bus protocol
      - Eased electrical timing requirements
    - Brought split transactions into PCI “world”
  - ✓ Evolutionary
    - PCI compatible at hardware *AND* software levels
    - PCI-X 266/533 added as “mid-life” performance bump
      - 2133MB/sec at PCI-X 266 and 4266MB/sec at PCI-X 533

Revolutionary AND Evolutionary

- PCI Express (aka PCIe®)
  - ✓ Revolutionary
    - Unprecedented bandwidth
      - x1: 500MB/sec in *EACH* direction
      - x16: 8000MB/sec in *EACH* direction
    - "Relaxed" electricals due to serial bus architecture
      - Point-to-point, low voltage, dual simplex with embedded clocking
  - ✓ Evolutionary
    - PCI compatible at software level
      - Configuration space, Power Management, etc
      - Of course, PCIe-aware O/S can get more functionality
    - Transaction layer familiar to PCI/PCI-X designers
    - System topology matches PCI/PCI-X
    - PCIe 2.0 doubled bandwidth from 250MB/s/lane to 500MB/s/lane
    - PCIe 3.0 will double again to 1GB/s/lane!

PCI Concepts

PCI Concepts

- Address spaces
  - ✓ Memory – 64-bit
  - ✓ I/O – 32-bit (non-burstable since PCI-X)
  - ✓ Configuration (“Config”) – Bus/Device/Function
- Key configuration space regs/concepts
  - ✓ *B*ase *A*ddress *R*egisters (BARs)
    - 64-bit vs 32-bit addressing
  - ✓ Linked list of capabilities

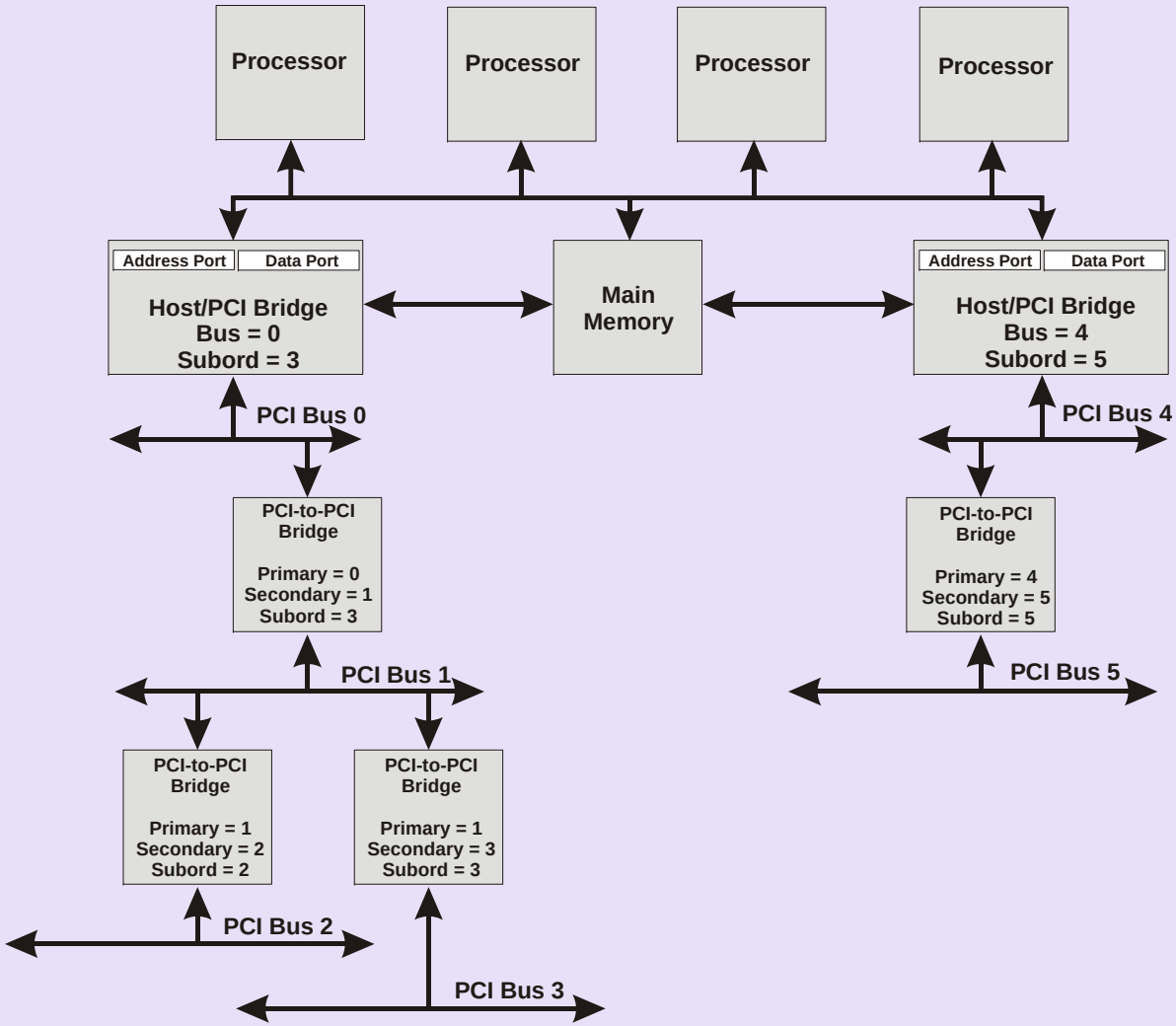
Address Spaces – Memory & I/O

- Memory space mapped cleanly to CPU semantics
  - ✓ 32-bits of address space initially
  - ✓ 64-bits introduced via Dual-Address Cycles (DAC)
    - Extra clock of address time on PCI/PCI-X
    - 4DWORD header in PCI Express
  - ✓ Burstable
- I/O space mapped cleanly to CPU semantics
  - ✓ 32-bits of address space
    - Actually much larger than CPUs of the time
  - ✓ Non-burstable
    - Most PCI implementations didn't support
    - PCI-X codified
    - Carries forward to PCI Express

Address Spaces – Configuration

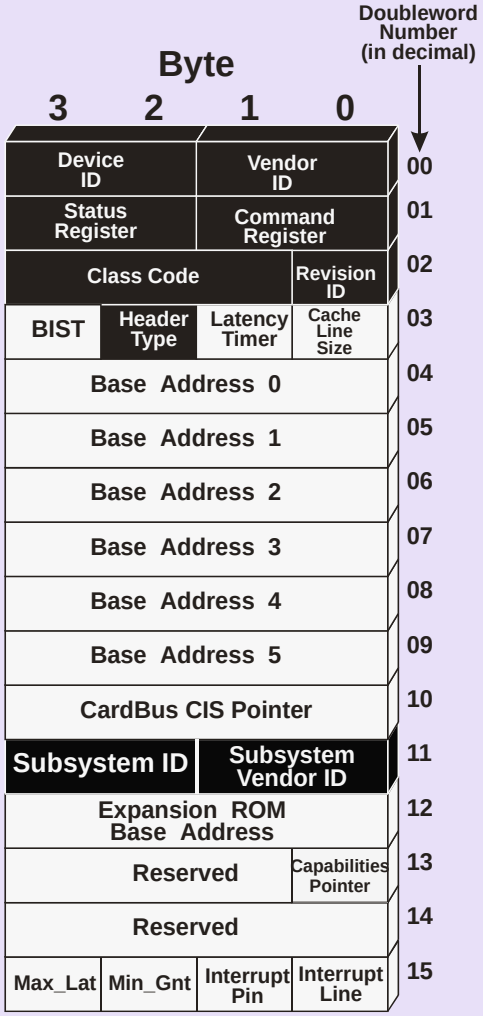
- Configuration space???
  - ✓ Allows control of devices' address decodes without conflict
  - ✓ No conceptual mapping to CPU address space
    - Memory-based access mechanisms introduced with PCI-X and PCIe
  - ✓ Bus / Device / Function (aka BDF) form hierarchy-based address
    - "Functions" allow multiple, logically independent agents in one physical device.
      - E.g. combination SCSI + Ethernet device
      - 256 bytes or 4K bytes of configuration space per device
    - PCI/PCI-X bridges form hierarchy
    - PCIe switches form hierarchy
      - Look like PCI-PCI bridges to software
  - ✓ "Type 0" and "Type 1" configuration cycles
    - Type 0: to same bus segment
    - Type 1: to another bus segment

Configuration Space (cont'd)

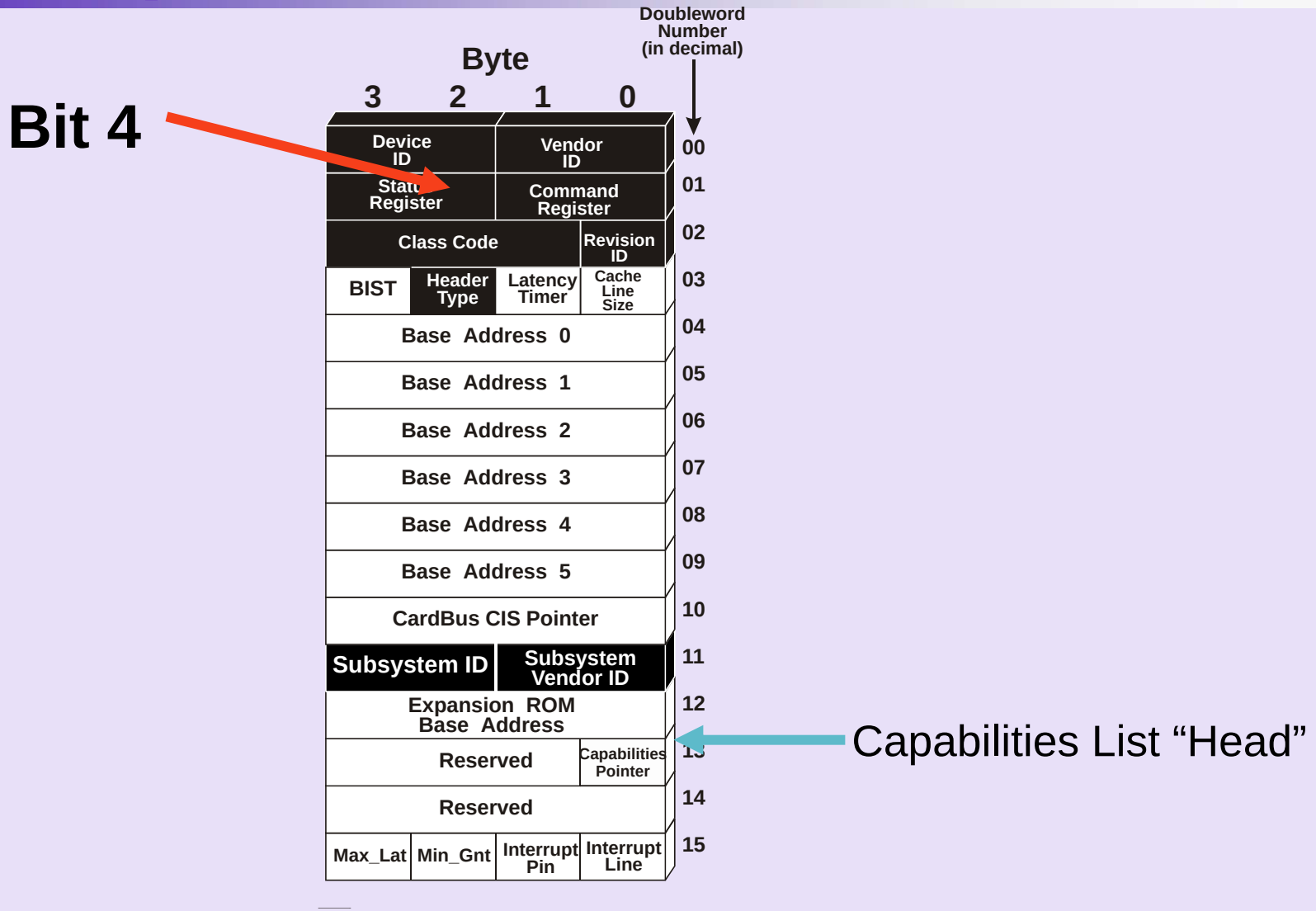


Using Configuration Space

- 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
- Other bus-oriented controls

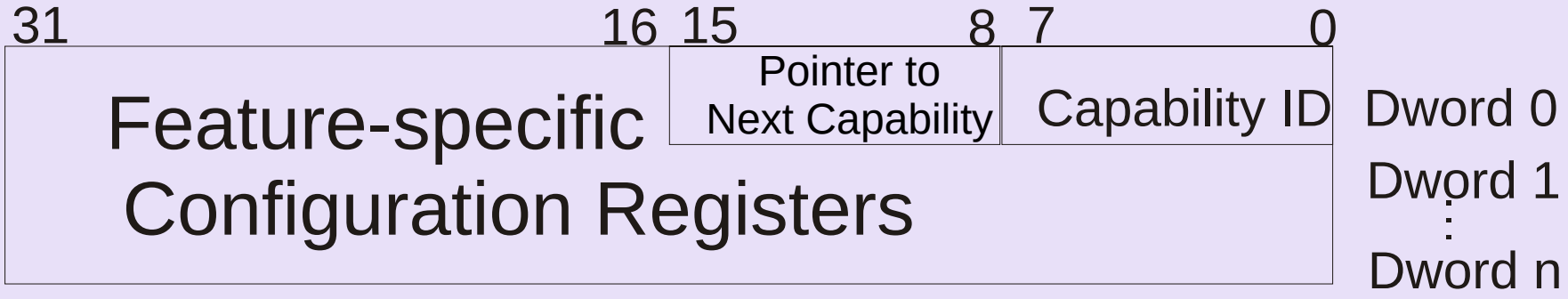


Using Configuration Space – Capabilities List



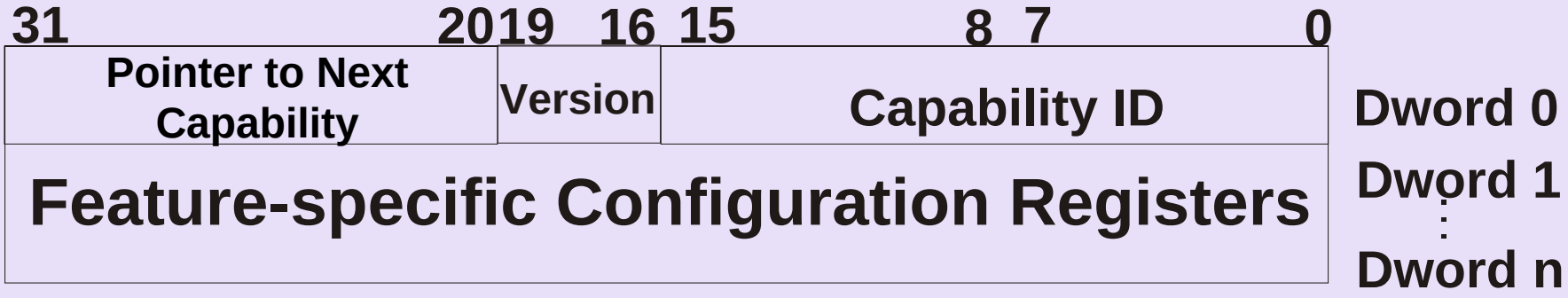
Using Configuration Space – Capabilities List (cont'd)

- Linked list
  - ✓ Follow the list! Cannot assume fixed location of any given feature in any given device
  - ✓ Features defined in their related specs:
    - PCI-X
    - PCIe
    - PCI Power Management
    - Etc…



Using Configuration Space – Extended Capabilities List

- 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
  - ✓ Features defined in PCI Express specification



Interrupts

- PCI introduced INTA#, INTB#, INTC#, INTD# - collectively referred to as INTx
  - ✓ Level sensitive
  - ✓ Decoupled device from CPU interrupt
  - ✓ System controlled INTx to CPU interrupt mapping
  - ✓ Configuration registers
    - report A/B/C/D
    - programmed with CPU interrupt number
- PCI Express mimics this via “virtual wire” messages
  - ✓ Assert_INTx and Deassert_INTx

What are MSI and MSI-X?

- Memory Write replaces previous interrupt semantics
  - ✓ PCI and PCI-X devices stop asserting INTA, INTB, INTC, INTD once MSI or MSI-X mode is enabled
  - ✓ PCI Express devices stop sending Assert_INTx and Deassert_INTx TLPs once MSI or MSI-X mode is enabled

- NOTE: *Boot devices* and any device intended for a non-MSI operating system generally must still support the appropriate INTx signaling!

MSI vs MSI-X

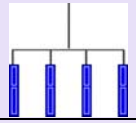
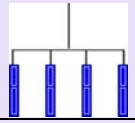
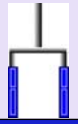
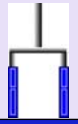
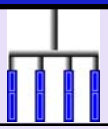
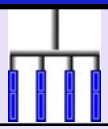
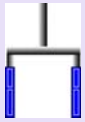
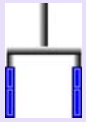
- MSI uses one address with a variable data value indicating which “vector” is asserting
- MSI-X uses a table of independent address and data pairs for each “vector”
  - ✓ Allows software to control aliasing (when fewer vectors are allocated than requested)
  - ✓ Table size supports more vectors than MSI structure allowed

PCI-X Explained

What is PCI-X?

- "PCI-X is high-performance backward compatible PCI"
  - ✓ PCI-X uses the same PCI architecture
  - ✓ PCI-X leverages the same base protocols as PCI
  - ✓ PCI-X leverages the same BIOS as PCI
  - ✓ PCI-X uses the same connector as PCI.
  - ✓ PCI-X and PCI products are interoperable
  - ✓ PCI-X uses same software driver models as PCI
- PCI-X is faster PCI
  - ✓ PCI-X 533 is up to 32 times faster than the original version of PCI
  - ✓ PCI-X protocol is more efficient than conventional PCI

PCI-X Modes and Speeds

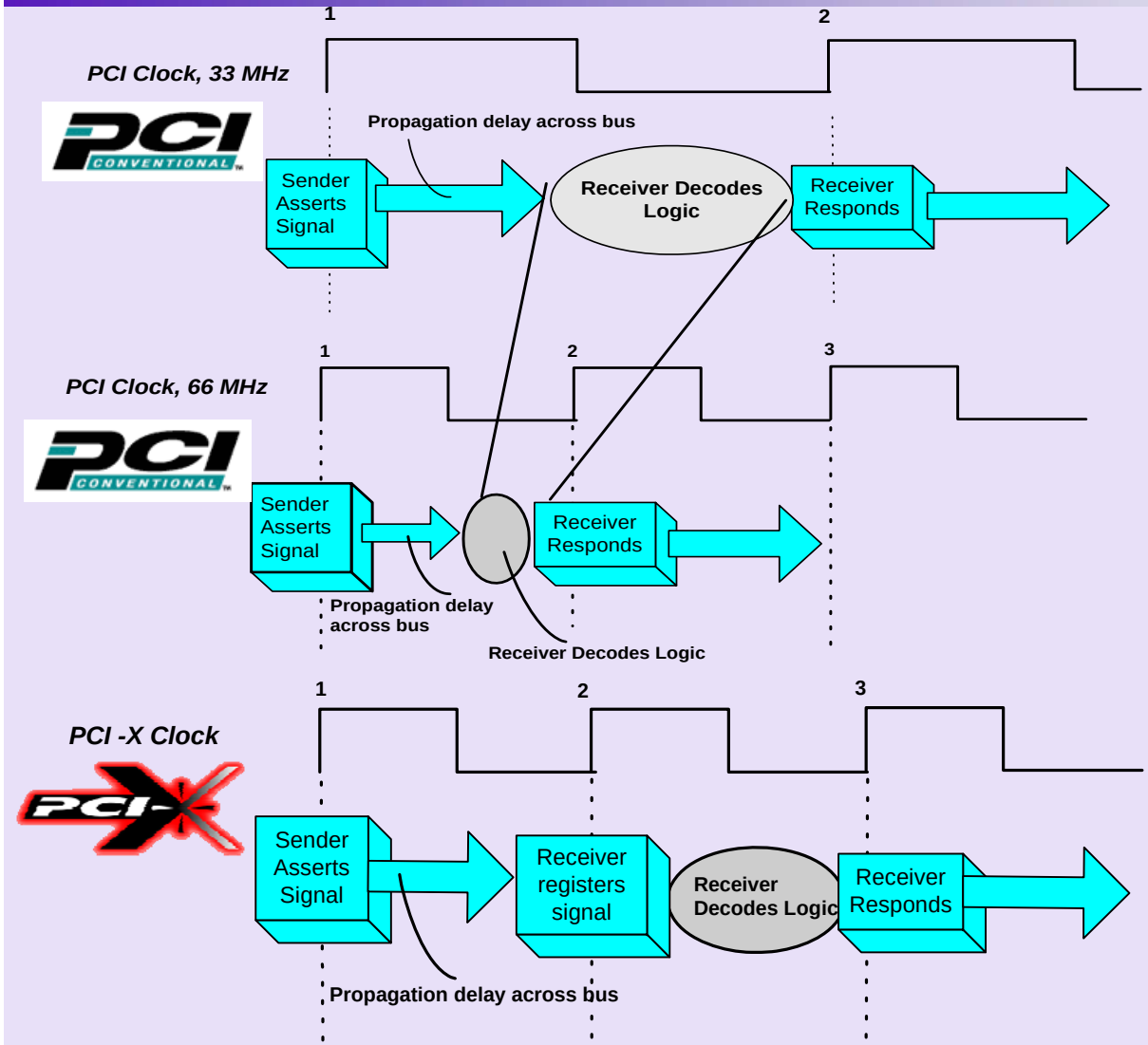
Mode	$V_{I/O}$	64-Bit Slots*	64-Bit MB/s	32-Bit Slots*	32-Bit MB/s	16-Bit	Error Prot	Conf Bytes	DIM
PCI 33	5V/3.3V		266		133	N/A	par	256	N/A
PCI 66	3.3V		533		266	N/A	par	256	N/A
PCI-X 66	3.3V		533		266	N/A	par or ECC	256	yes
PCI-X 133 (operating at 100 MHz)	3.3V		800		400	N/A	par or ECC	256	yes
PCI-X 133	3.3V		1066		533	N/A	par or ECC	256	yes
PCI-X 266	1.5V		2133		1066	533	ECC	4K	yes
PCI-X 533	1.5V		4266		2133	1066	ECC	4K	yes

Mode 1

Mode 2

* For lower bus speeds, # slots / bus is implementation choice to share bandwidth

Registered Bus Protocol

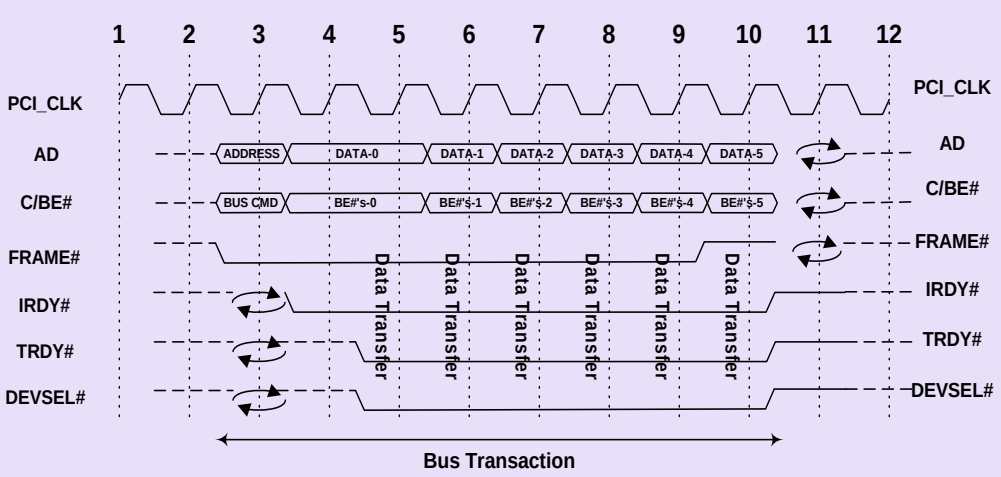


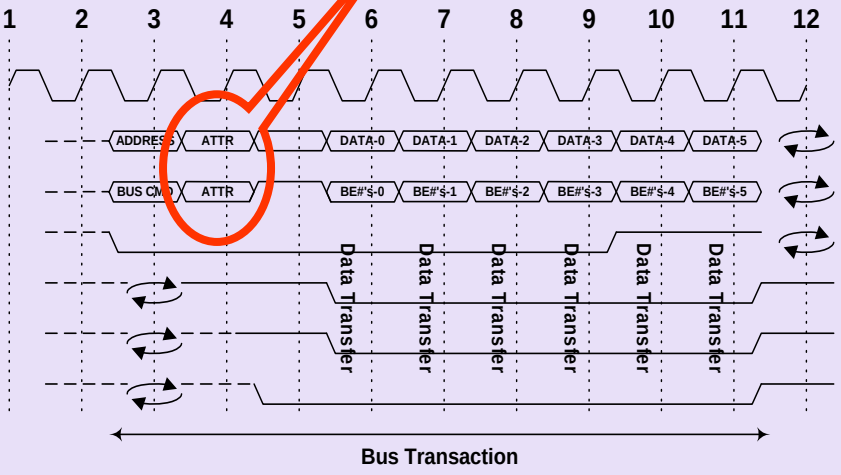
- PCI @ 33MHz
  - ✓30 ns period
  - ✓7 ns setup time
- PCI @ 66MHz
  - ✓ 15 ns period
  - ✓ 3ns setup time
- PCI-X registered protocol allocates a full clock period for logic decision
  - ✓ @ 66MHz - 15ns
  - ✓ @ 133MHz - 7.5ns

PCI 2.x/3.0 vs. PCI-X Mode 1

- Same bus and control signals
- Evolutionary protocol changes
- Clock frequency up to 133 MHz

New "Attribute" phase for enhanced features



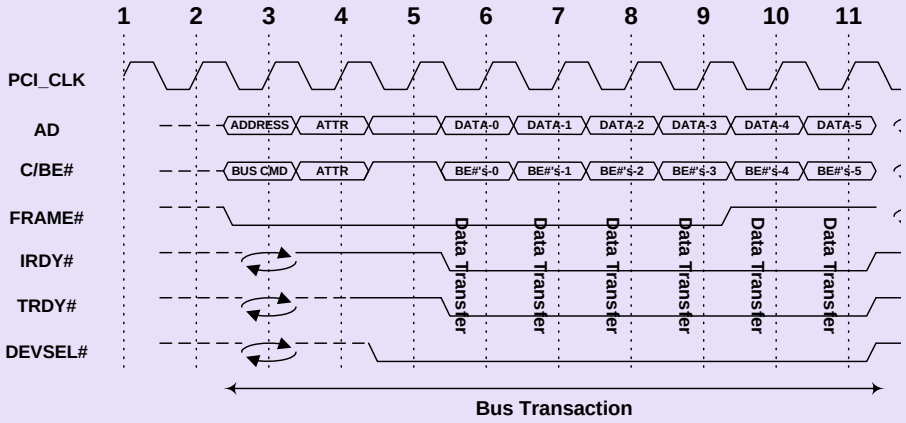


PCI CONVENTIONAL

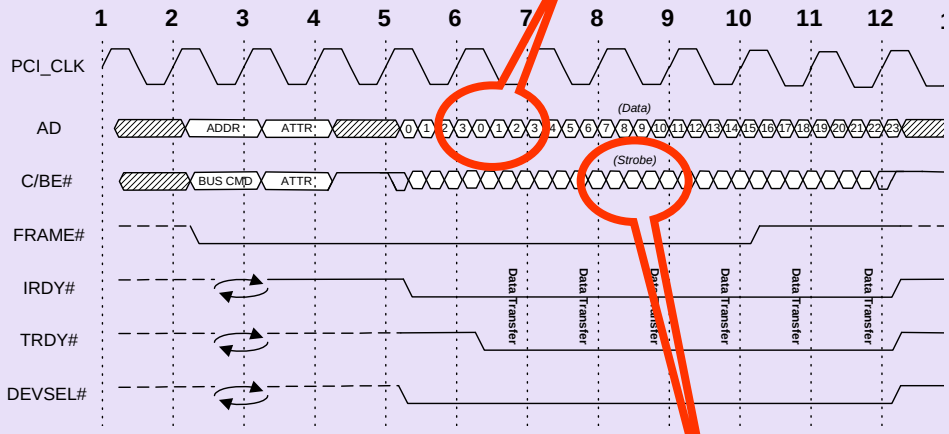
PCI-X (Common clock)

PCI-X 66/133 (Mode 1) vs. PCI-X 266/533 (Mode 2)

- Same bus and control signals
- PCI-X 266 moves 2x the data
  PCI-X 533 moves 4x the data
- Clock frequency up to 133 MHz



PCI-X 66/133 (Mode 1)



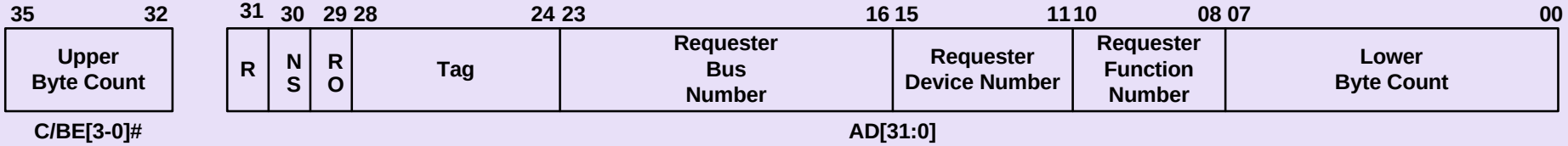
PCI-X 533 (Mode 2)

4 transfers per clock cycle

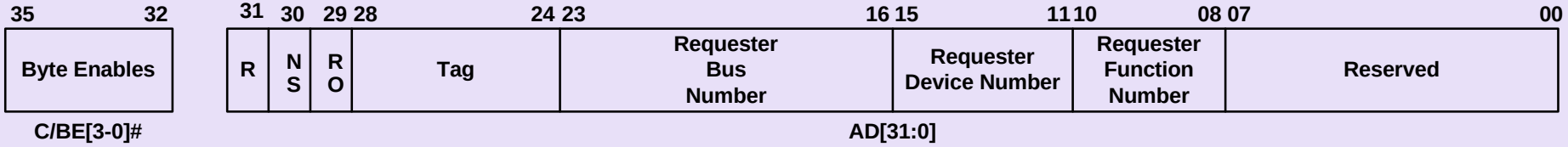
source-synchronous data strobes share C/BE pins

Transaction Attributes

Requester Attributes for Burst Transactions



Requester Attributes for DWORD Transactions



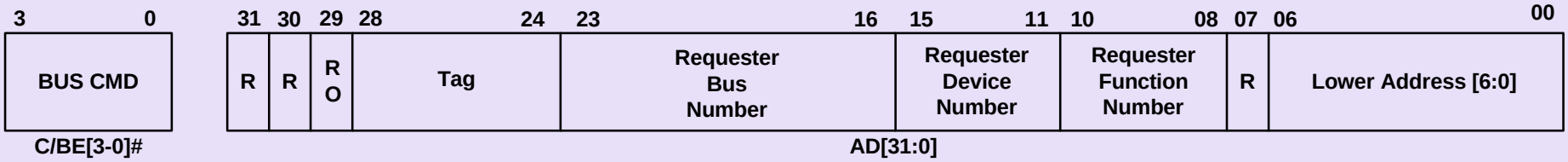
RO -- Relax ordering

NS -- No Snoop

R -- Reserved

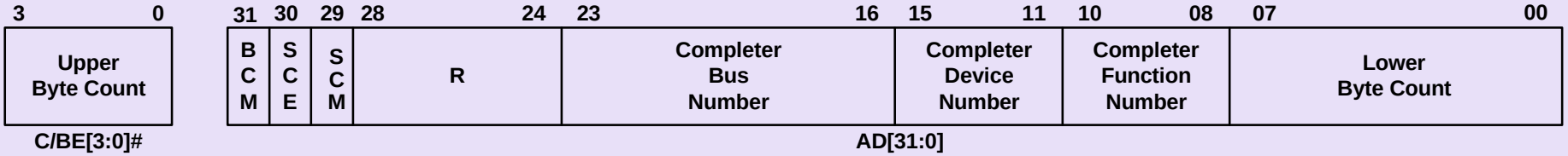
Transaction Attributes

Split Completion Address



RO -- Relaxed ordering

Completer Attributes



SCM -- Split Completion Message

SCE -- Split Completion Error

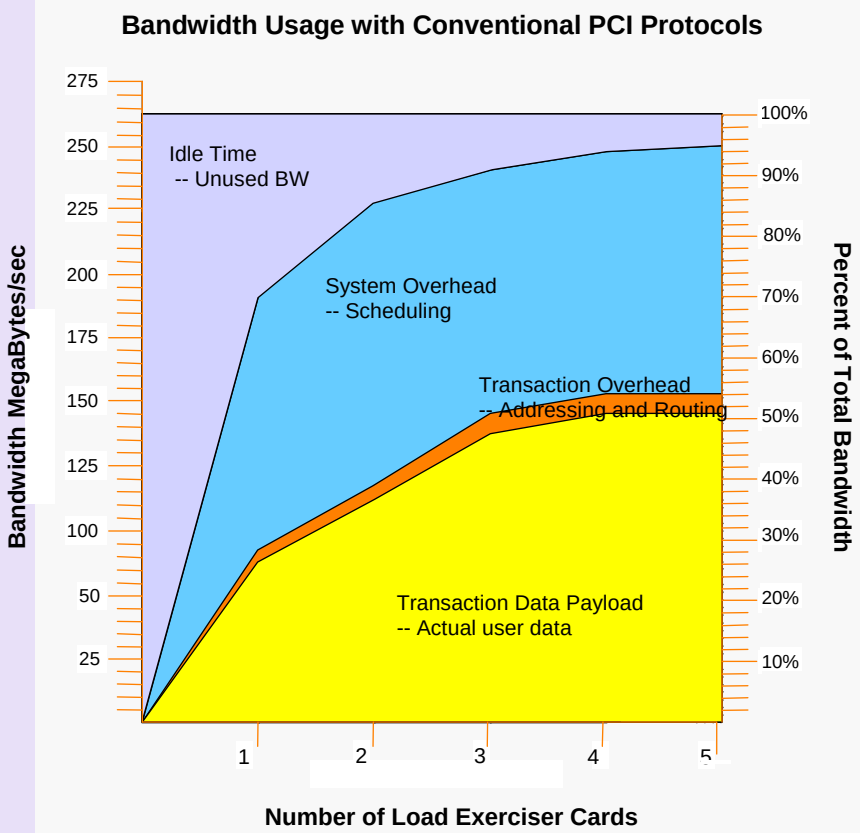
BCM -- Byte Count Modified

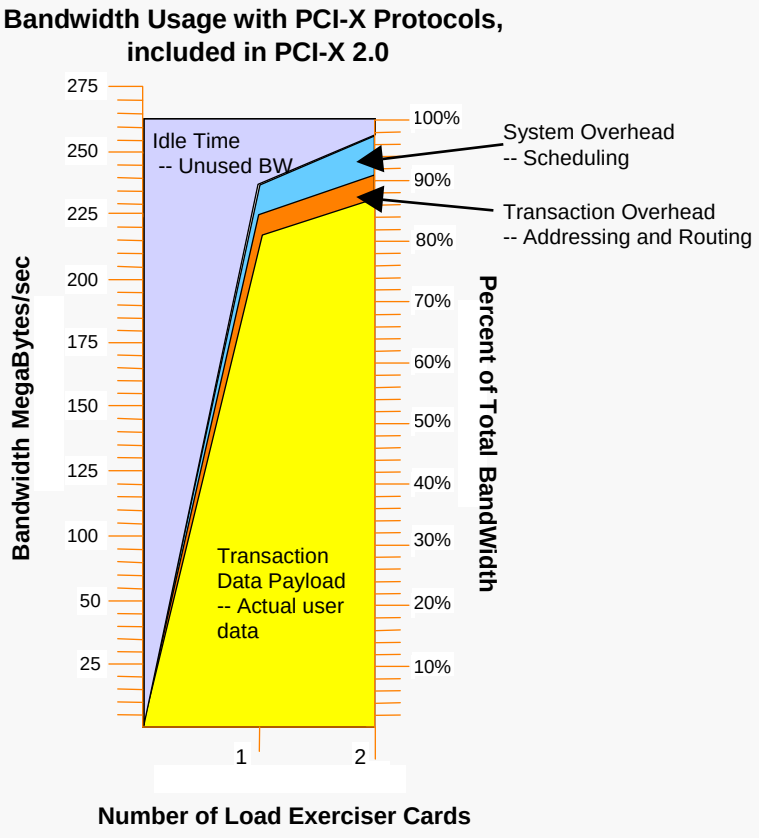
R -- Reserved

Split Transactions

- Bus efficiency of Read almost as good as Write
- Split Completion routed back to requester across bridges using initiator's number and bus number
- Split Transaction components
  - ✓ Step 1. Requester requests bus and arbiter grants bus
  - ✓ Step 2. Requester initiates transaction
  - ✓ Step 3. Target (completer) communicates intent with new target termination, Split Response
  - ✓ Step 4. Completer executes transaction internally
  - ✓ Step 5. Completer requests bus and arbiter grants bus
  - ✓ Step 6. Completer initiates Split Completion

Efficient PCI-X Protocol





The PCI-X protocol is more efficient than traditional PCI.

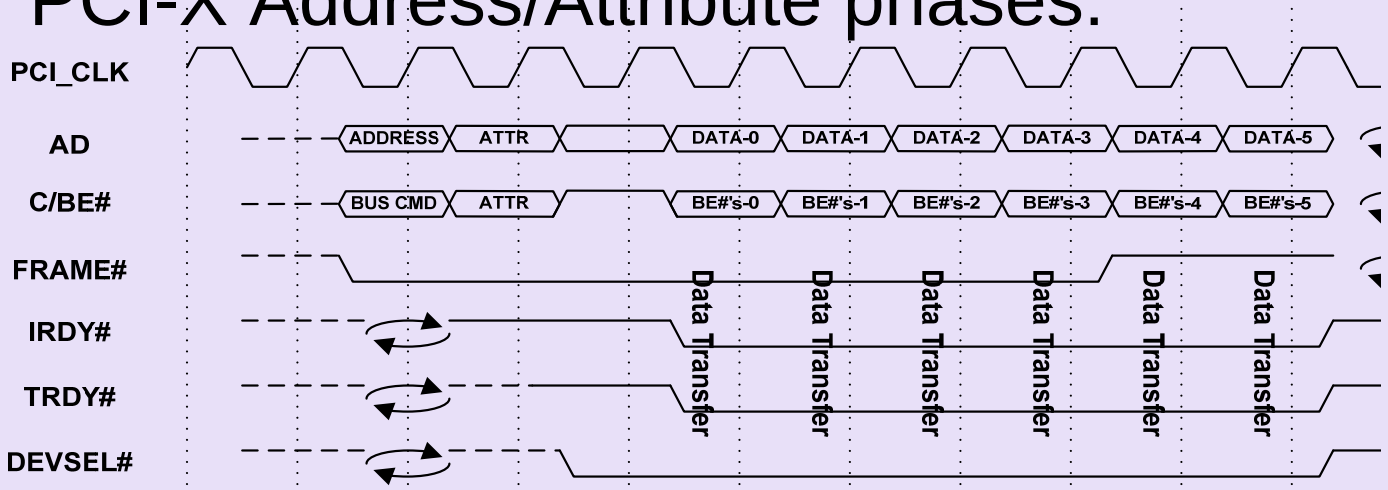
PCI Express Overview

PCIe Architecture Features

- **PCI Compatibility**
  - ✓ Configuration and PCI software driver model
  - ✓ PCI power management software compatible
- **Performance**
  - ✓ Scalable frequency (2.5-5GT/s)
  - ✓ Scalable width (x1, x4, x8, x16)
  - ✓ Low latency and highest utilization (Bandwidth/pin)
- **Physical Interface**
  - ✓ Point-to-point, dual-simplex
  - ✓ Differential low voltage signaling
  - ✓ Embedded clocking
  - ✓ Supports connectors, modules, cables
- **Protocol**
  - ✓ Fully packetized split-transaction
  - ✓ Credit-based flow control
  - ✓ Hierarchical topology support
  - ✓ Virtual channel mechanism
- **Advanced Capabilities**
  - ✓ CRC-based data integrity, hot plug, error logging
- **Enhanced Configuration Space**
  - ✓ Extensions and bridges into other architectures

PCIe Protocol Overview

- PCI-X Address/Attribute phases:

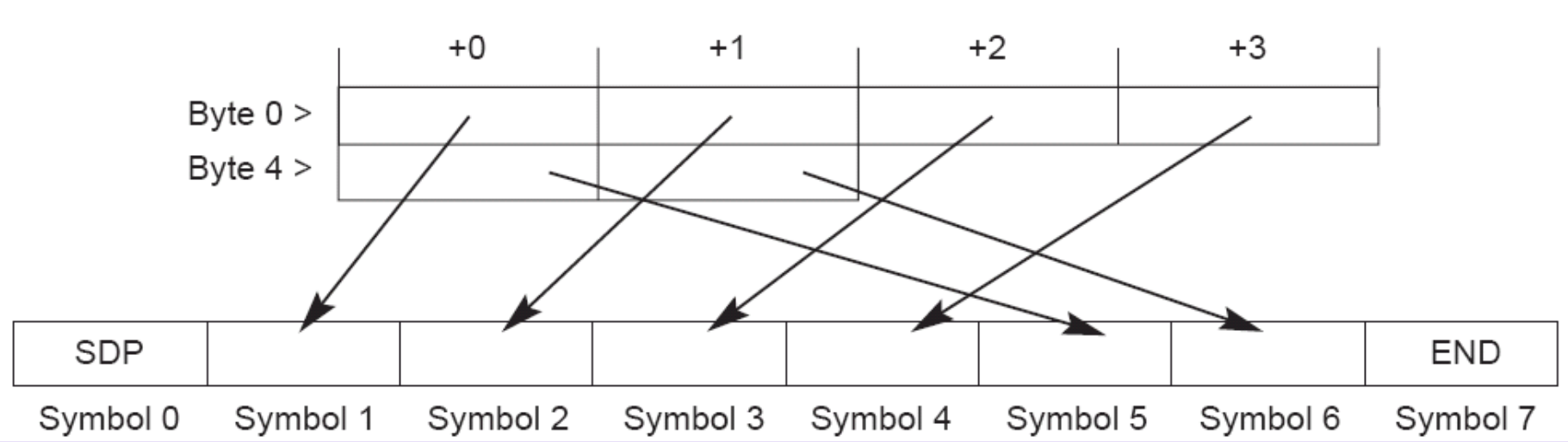


- Evolved into the PCIe Packet Header:

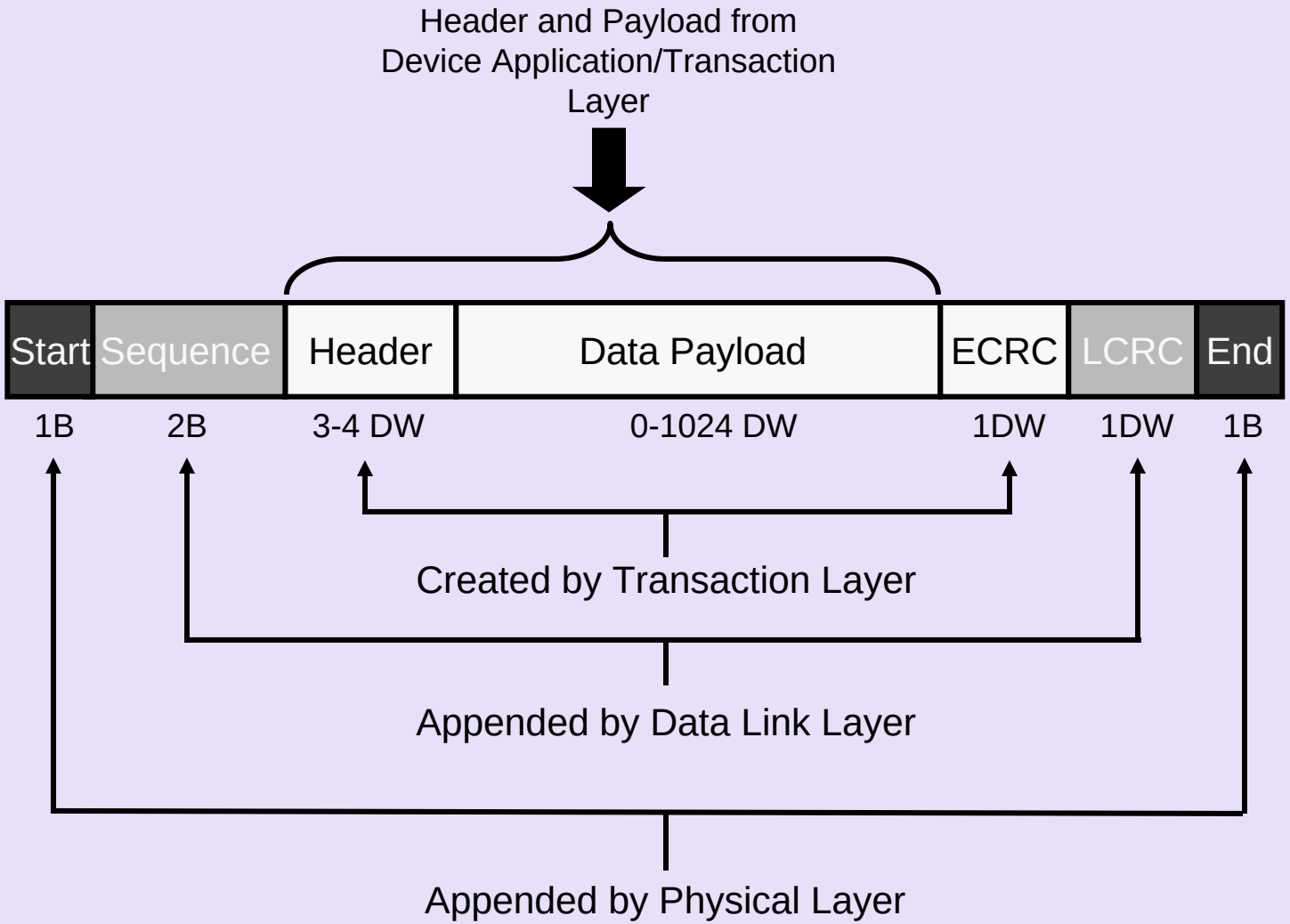
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0 >	R	Fmt x 1			Type				R	TC			Reserved				TD	EP	Attr		R				Length							
Byte 4 >	Requester ID																Tag								Last DW BE				1st DW BE			
Byte 8 >	Address[63:32]																															
Byte 12 >	Address[31:2]																								R							

PCIe Protocol Overview

- The packet bytes get converted to 8b/10b and serialized

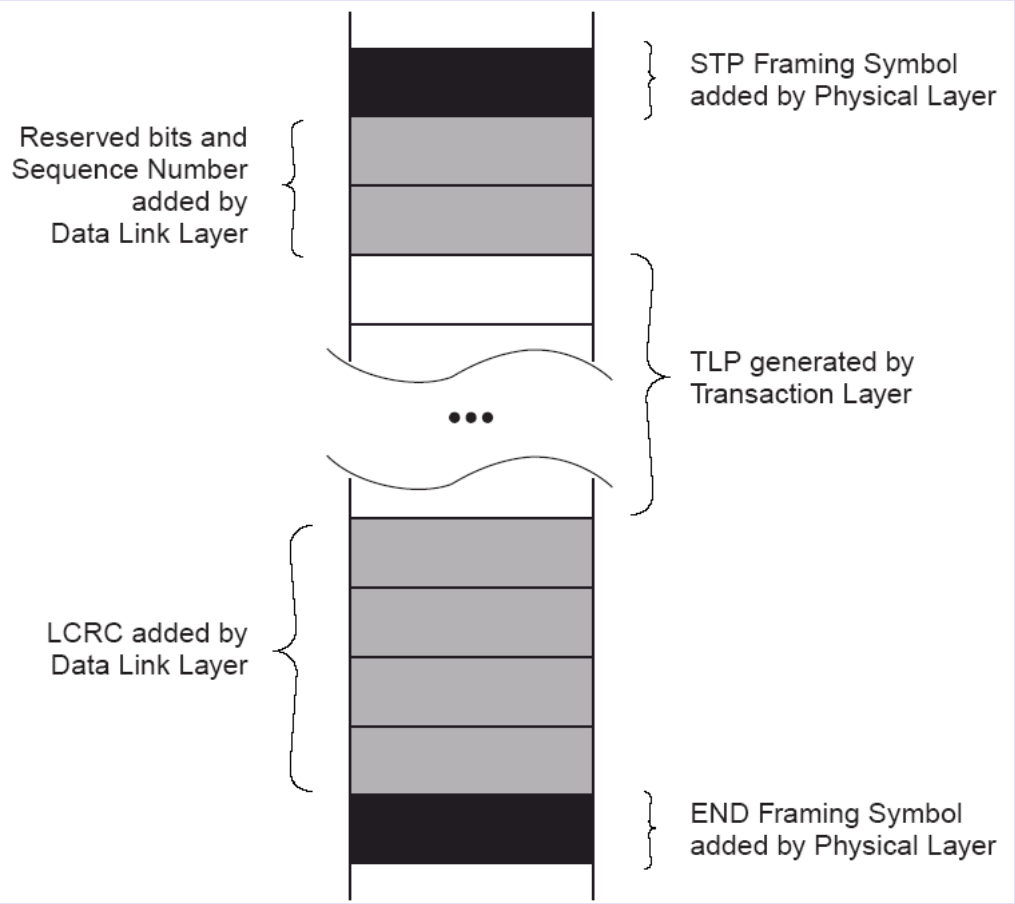


PCIe Protocol Overview



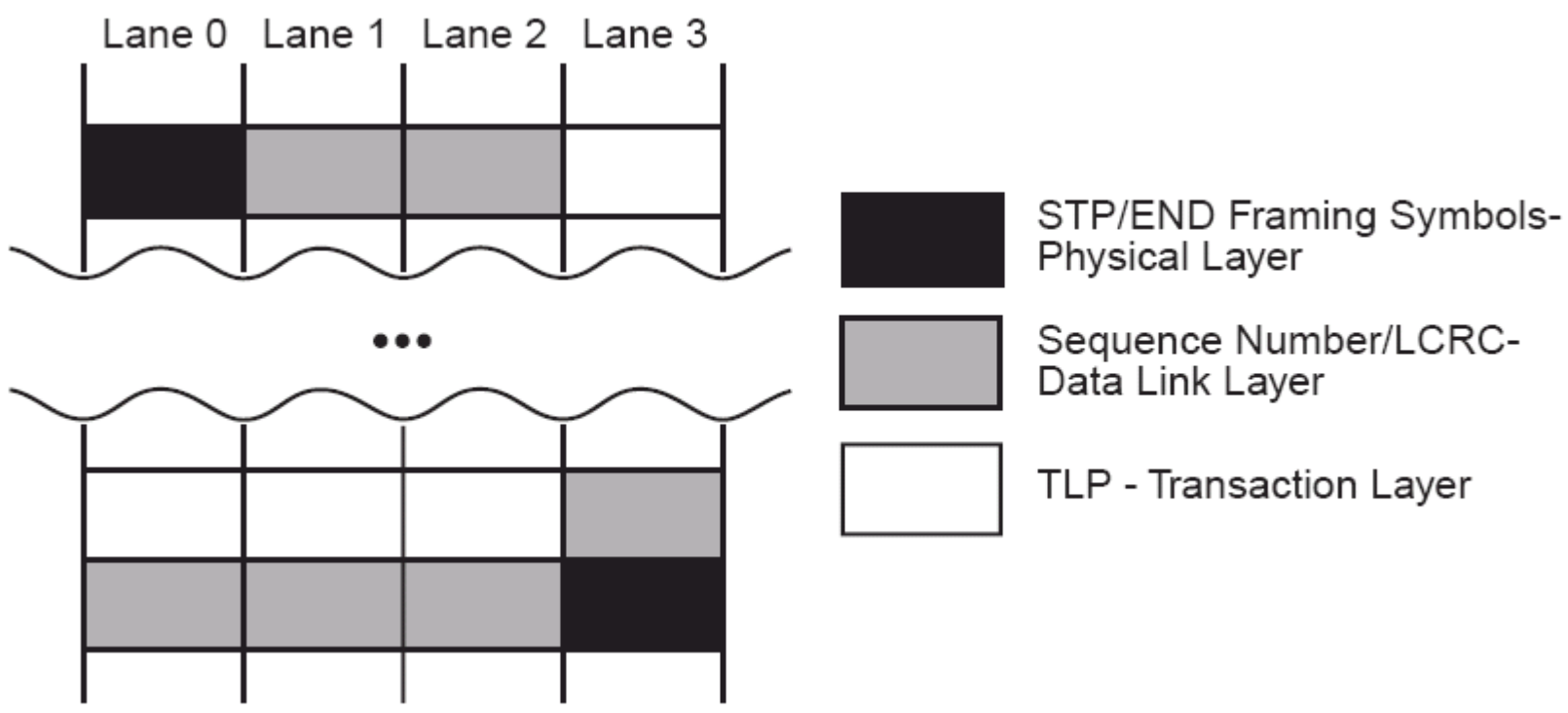
PCIe Protocol Overview

- Framing varies depending on link width
  - ✓ x1



PCIe Protocol Overview

- Framing varies depending on link width
  - ✓ x4



PCIe Power Management

- PCI Power Management – Device Layer
  - ✓ Introduced with Conventional PCI
  - ✓ Software-driven, addresses ***devices not in use***
  - ✓ D0 – operating normally
  - ✓ D1 – reduced power
  - ✓ D2 – reduced power
    - – In both D1/D2, the device can't *do* anything on the bus except respond to configuration cycles
  - ✓ $D3_{cold}$ – device powered off
  - ✓ $D3_{hot}$ – device all but powered off

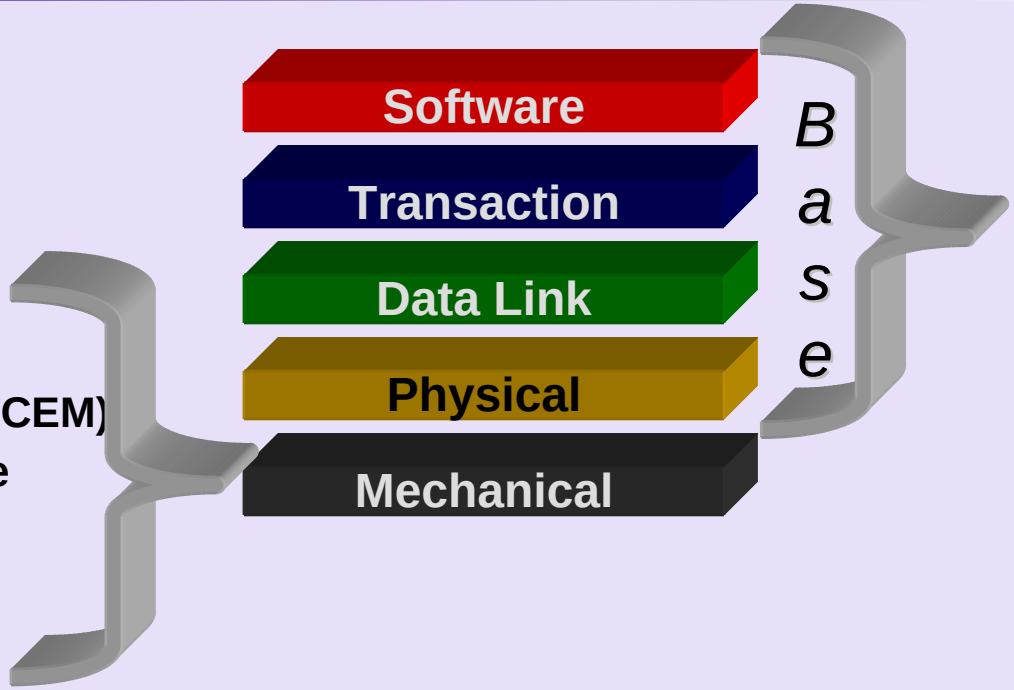
PCIe Power Management

- Link Layer
  - ✓ L0 – operating normally
  - ✓ L0s – one or both sides in low power mode
  - ✓ L1 – lower powered mode
  - ✓ L2 – deep power save mode but Vaux provided
  - ✓ L3 – powered off
- Active State Power Management (ASPM)
  - ✓ Software-invisible, addresses ***idle devices***
  - ✓ L0s required – enter on timer, exit on activity
  - ✓ L1 optional – enter on timer or other conditions, exit on activity

PCIe Power Management

- Dynamic Power Allocation (DPA)
  - ✓ New configuration space reporting and selecting different power “envelopes” for each device function
    - ECN against PCIe 2.0 (appears in PCIe 2.1 and 3.0)
    - Endpoint-only
  - ✓ Addresses ***fully functioning*** devices
  - ✓ Software-set
    - Allows trading off performance for power
    - Exact “cost” of reduced power is device-dependent

PCIe Specifications



Form Factors:

- Card (CEM)
- Mini Card (MiniCEM)
- ExpressModule
- Cable

- Base
  - ✓ Electrical
  - ✓ Protocol
  - ✓ Configuration
- Bridge
- I/O Virtualization

- Layered, scalable architecture
- Performance matched to applications
- Innovative form factors

I/O Virtualization Overview

I/O Virtualization – What?

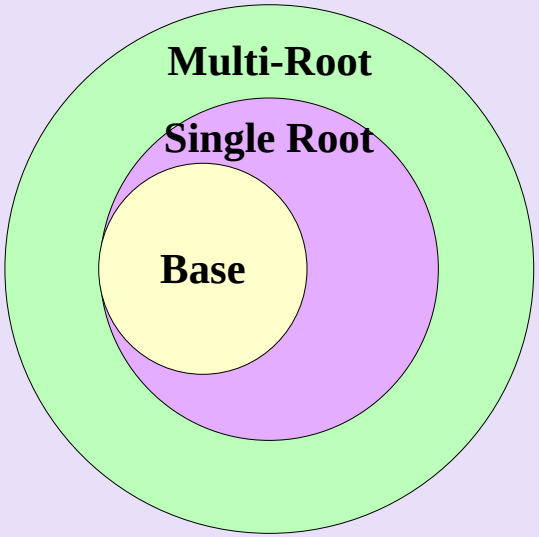
From an adapter point of view:

- One physical device looks like multiple devices
- Virtual devices appear completely independent
  - ✓ May occupy different PCI memory ranges
  - ✓ May have different settings for PCI Configuration registers

From a system point of view:

- "System Image" is a real or virtual system of CPU(s), Memory, O/S, I/O, etc
  - ✓ Multiples may run on one or more sets of hardware
- Each "System Image" (SI) needs to "see" it's own PCI hierarchy
  - ✓ Even if NO end devices are actually shared
  - ✓ Only its "portion" of shared end devices

I/O Virtualization – How?



"Concentric Circles" model

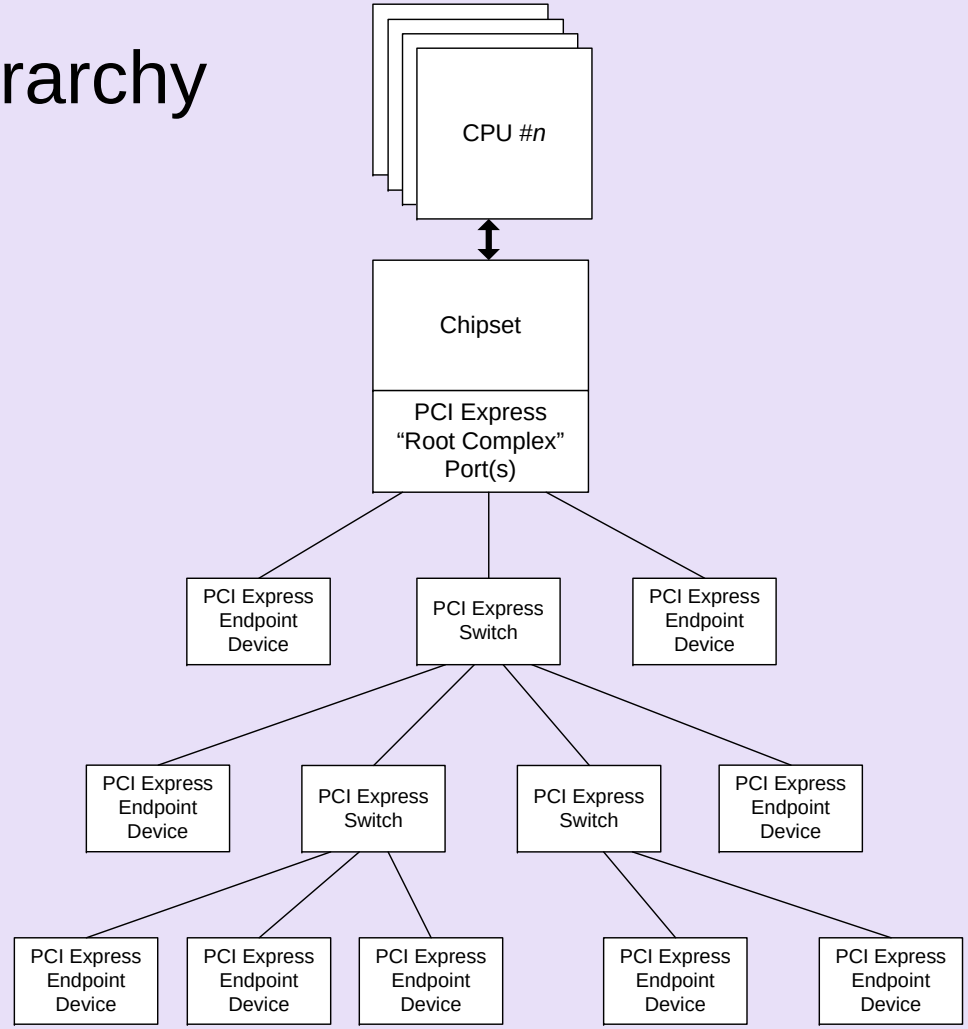
- Attachment of existing PCIe 1.x Base components
  - ✓ Root Complexes, Switches, Endpoints, and Bridges.
- A solution to use a combination of existing base and IOV-aware components:
  - ✓ Single Root capabilities are a superset of the PCIe 1.x Base specification.
  - ✓ Multi-Root capabilities build upon the Base and Single Root capabilities.
- IOV-capable components are backwards compatible with existing software.
  - ✓ Although some or all of the new IOV capabilities may not be supported in these circumstances.

I/O Virtualization "Flavors"

- Single Root (SR IOV)
  - ✓ Fits into existing PCI hierarchies today, with address space partitioned/allocated "above" the Root Complex
    - Uses RoutingID (formerly the Bus/Device/Function field) in packets to track back to appropriate System Image
  - ✓ Existing or absolutely minimally changed Root Complex silicon
  - ✓ Existing or minimally changed Switch silicon
  - ✓ New Endpoint silicon
  - ✓ Presumes existence of a Virtualization Intermediary
    - Opens market to lots of existing or simply-derived systems
    - Shifts substantial burden to software

I/O Virtualization "Flavors"

- Single Root Hierarchy

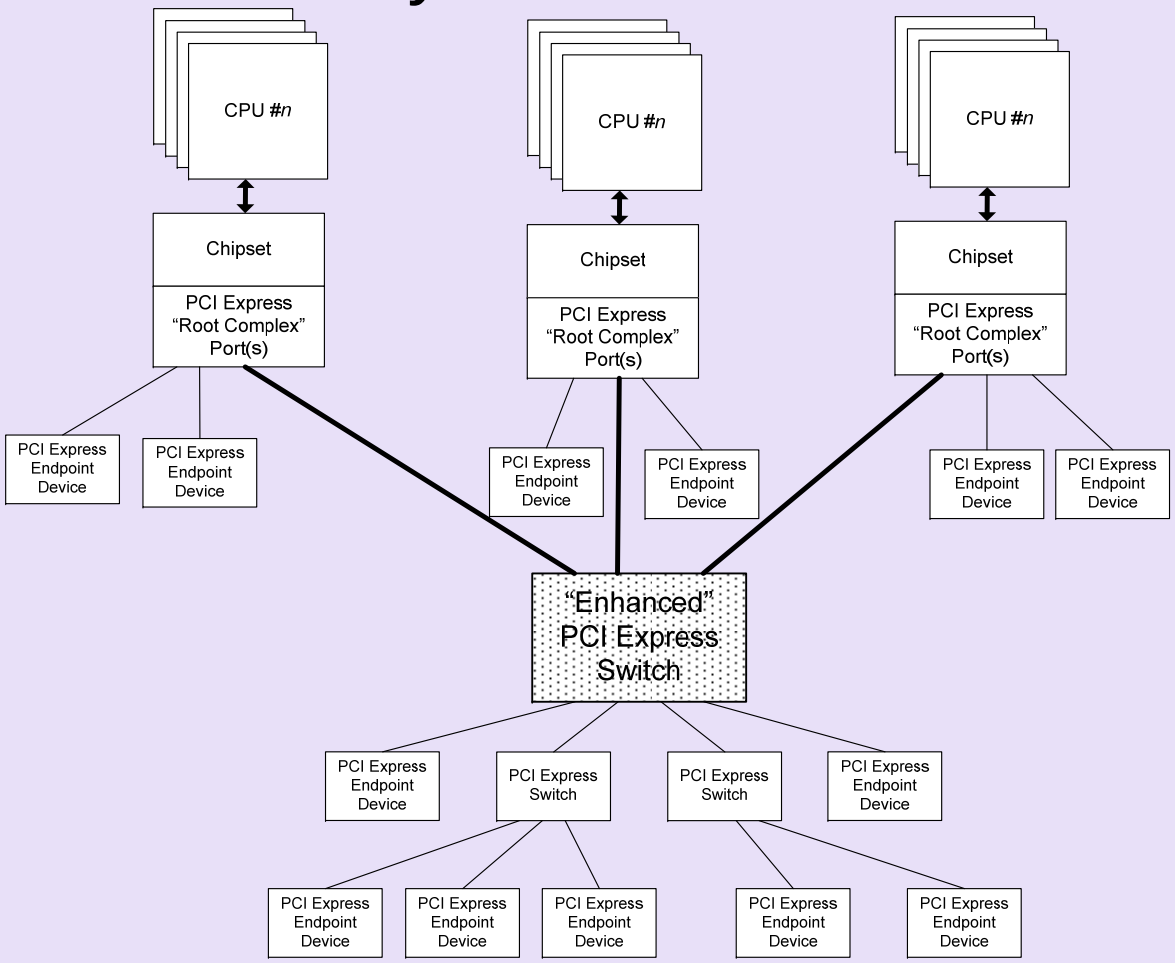


I/O Virtualization "Flavors" (cont'd)

- Multi-Root (MR IOV)
  - ✓ Most obvious example is a blade server with a PCIe "backplane"
  - ✓ New PCIe hierarchy construct
    - Effectively a (mini) fabric
    - Logically partitions the hierarchy into multiple Virtual Planes (VPs) all sharing the same physical hierarchy
  - ✓ Existing or absolutely minimally changed Root Complex (i.e. chipset) silicon
  - ✓ New Switch silicon
    - Allows for use of existing or minimally changed switches in a reduced capacity in certain places
  - ✓ New Endpoint silicon

I/O Virtualization "Flavors" (cont'd)

- Multi-Root Hierarchy



I/O Virtualization "Flavors" (cont'd)

- Address Translation Services (ATS)
  - ✓ Defines a set of transactions PCIe components can use to exchange and share translated addresses
    - With an I/O MMU, bus addresses map to different system addresses based on the "identity" of the agent using them
      - Allows each SI to appear to use the entire address space
      - System's I/O MMU does translation of "normal" addresses
        - Expensive in performance terms
        - Impossible to size I/O MMU's TLB or cache for all applications
      - ATS-aware devices can translate an address range and bypass I/O MMU
    - New PCIe commands for translation Requests, Completions, and Invalidations
  - ✓ Implementation is optional even for IOV devices

Thank you for attending the
PCI-SIG Developers Conference 2009

For more information please go to
www.pcisig.com