PCI Overview

PCI vs PCI Express

Marcel Apfelbaum
Red Hat
February 09, 2016

Agenda (PCI vs PCIe)

- **PCI Evolution**
- Technologies overview
- Topology differences
- Configuration
- PCI Hot Plug
- Interrupts handling

PCI Evolution

- Peripheral Component Interconnect (PCI) is a **local computer bus** for attaching hardware devices in a computer.

Bus type	Specification Release	Date of Release	Maximum Transfer Rate
PCI 33 MHz	2.0	1993	266 MB/s
PCI 66 MHz	2.1	1995	533 MB/s
PCI-X 66 MHz and 133 MHz	1.0	1999	1,066 MB/s
PCI-X 266 MHz and 533 MHz	2.0	Q1, 2002	4,266 MB/s

PCI Evolution

- PCI Express (Peripheral Component Interconnect Express), officially abbreviated as PCIe, is a high-speed serial computer expansion bus standard, designed to replace the older PCI, PCI-X, and AGP bus standards.

PCI Express	Date Of Release	Line encoding	Transfer rate	Bandwidth	
				Per lane	X16 lane slot
1.0	Q2, 2002	8b/10b	2.5 GT/s	2 Gbit/s (250 MB/s)	32 Gbit/s (4 GB/s)
2.0	2007	8b/10b	5 GT/s	4 Gbit/s (500 MB/s)	64 Gbit/s (8 GB/s)
3.0	2010	128b/130b	8 GT/s	7.877 Gbit/s (984.6 MB/s)	126.032 Gbit/s (15.754 GB/s)
4.0	Early 2017 ?	128b/130b	16 GT/s	15.754 Gbit/s (1969.2 MB/s)	252.064 Gbit/s (31.508 GB/s)

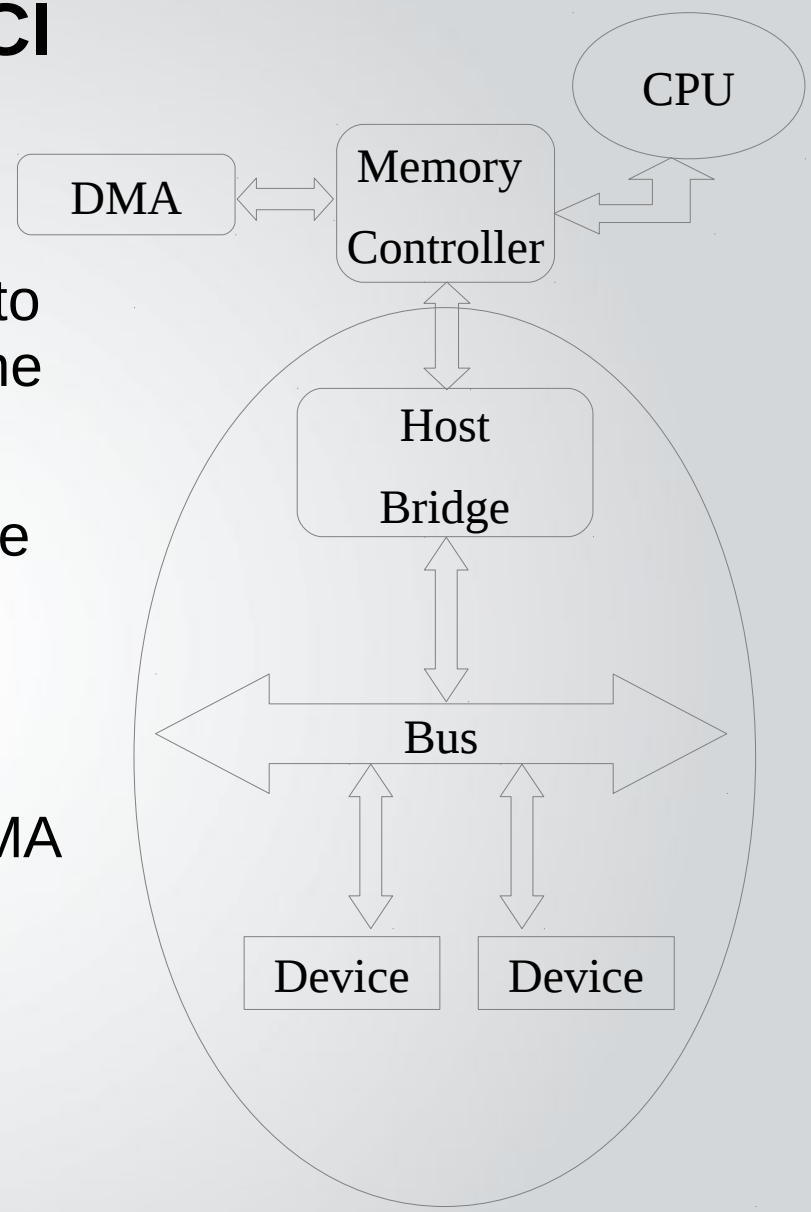
Agenda (PCI vs PCIe)

- PCI Evolution
- **Technologies overview**
- Topology differences
- Configuration
- PCI Hot Plug
- Interrupts handling

Technologies overview – PCI

The basics

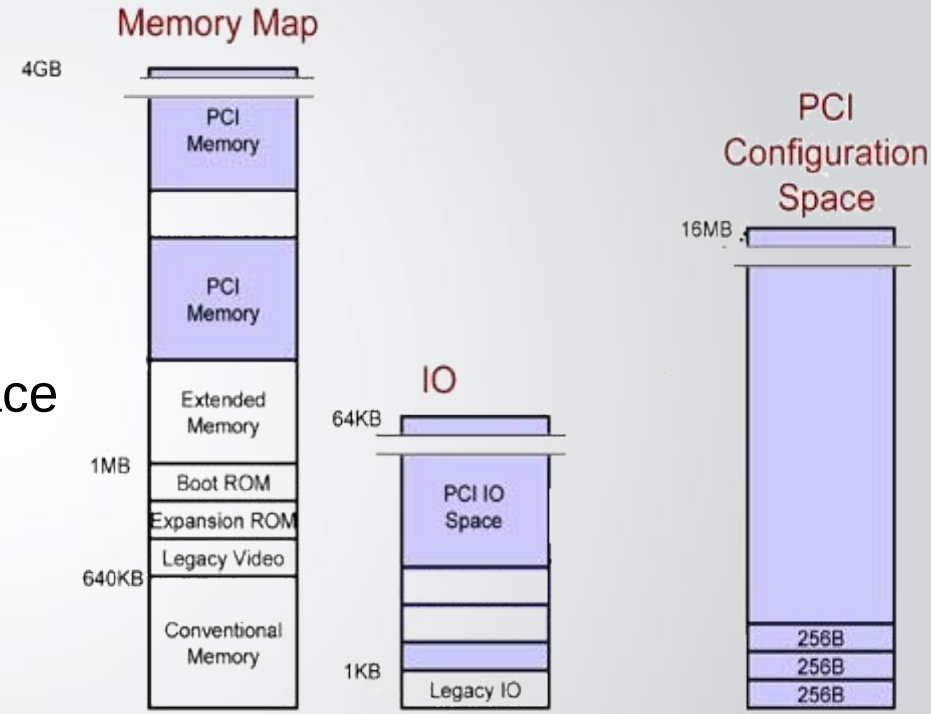
- PCI uses a **shared bus** topology to allow for communication among the different devices on the bus.
- There is a **bus arbitration** scheme in place for deciding who gets access to the bus and when.
- The **Host Bridge** provides an interconnect between the CPU/DMA and peripheral components



Technologies overview – PCI

The CPU point of view

- PCI devices are accessible via a fairly straightforward load-store mechanism.
  - **Memory** address(MMIO) space
  - **I/O** address spaces
  - **Configuration** space.



Technologies overview – PCI

Bus traffic

- Command traffic (configuration)
- Read/Write traffic

C/BE#	Command
0000	Interrupt Acknowledge
0001	Special Cycle
0010	I/O Read
0011	I/O Write
0100	Reserved
0101	Reserved
0110	Memory Read
0111	Memory Write
1000	Reserved
1001	Reserved
1010	Configuration Read
1011	Configuration Write
1100	Memory Read Multiple
1101	Dual Address Cycle
1110	Memory Read Line
1111	Memory Write and Invalidate

Memory

I/O

Configuration

Special-Purpose

Reserved

Technologies overview – PCI

PCI to PCI Bridges

- Maximum 32 multi-function devices per bus.
- A PCI-to-PCI bridge provides a connection path between two independent PCI buses.
- A bridge has two PCI interfaces, the primary and secondary.

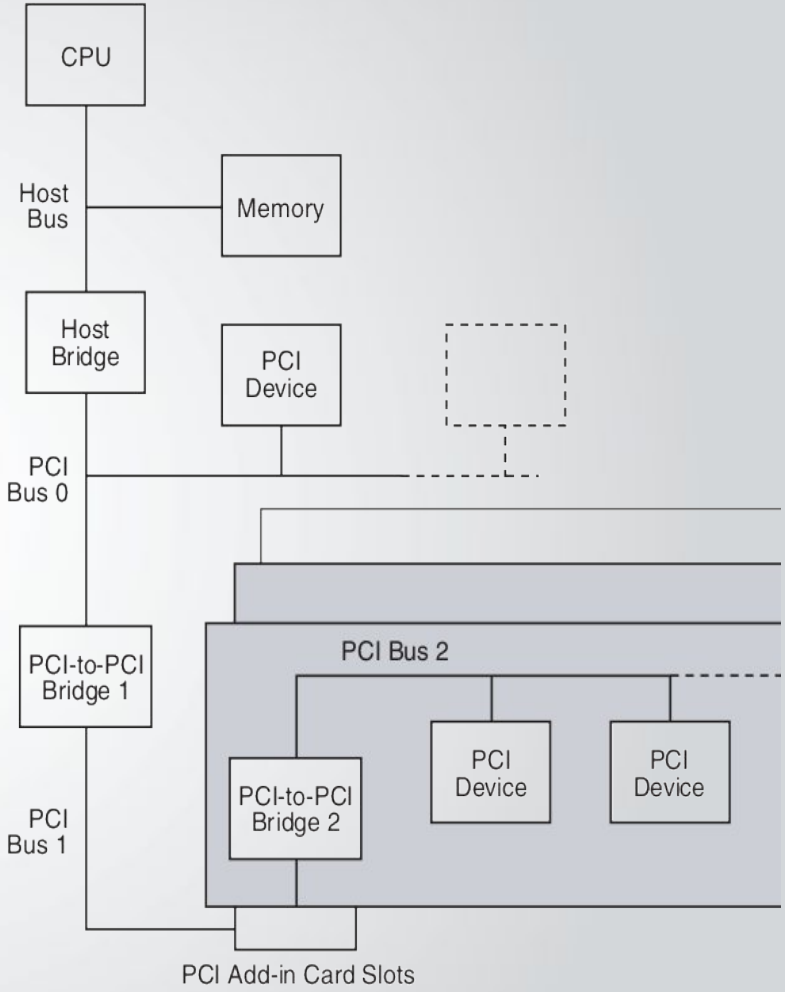


Figure 1-1: Typical Bridge Applications

Technologies overview – PCI

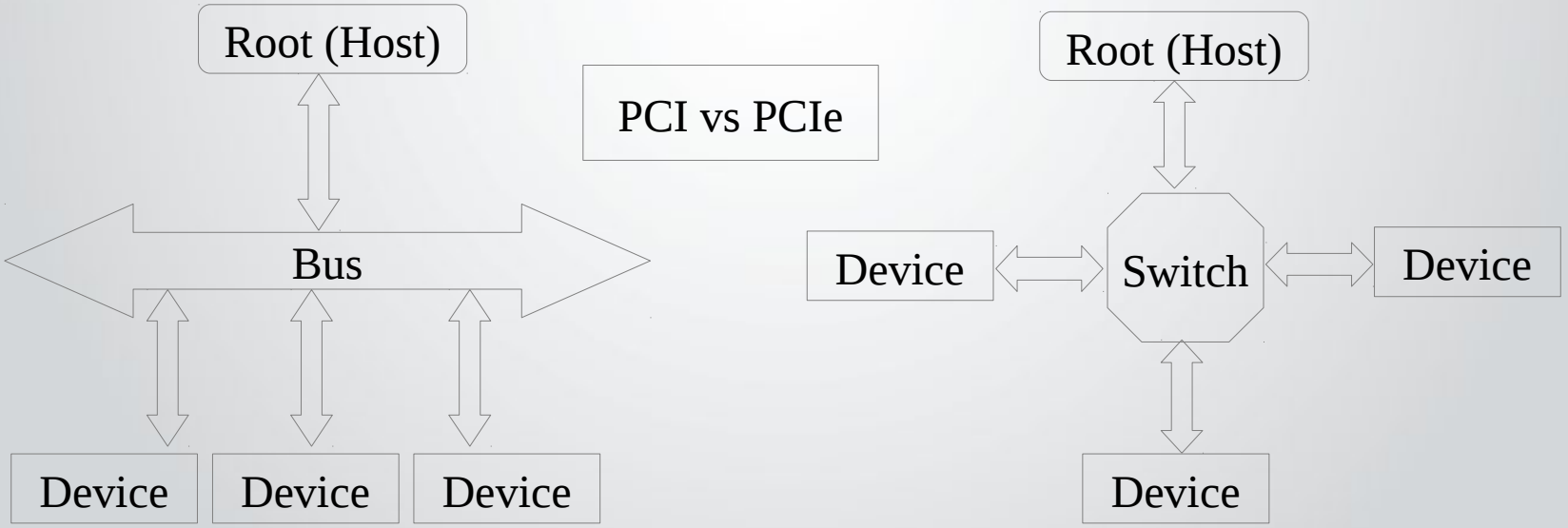
PCI limitations

- Its highly parallel shared-bus architecture holds it back by limiting its bus speed and scalability.
- Its simple, load-store, flat memory-based communications model is less robust and extensible than a routed, packet-based model.
- PCI-X: wider and faster, but still outdated.

Technologies overview – PCIe

The basics

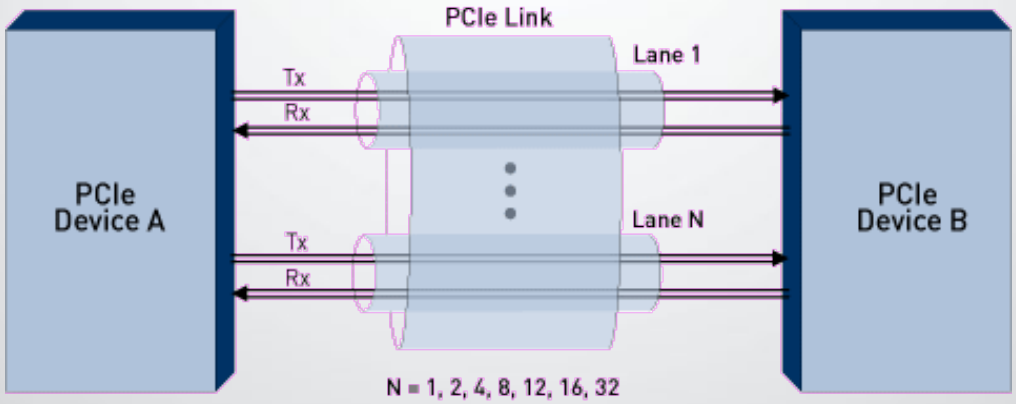
- PCIe's most obvious improvement over PCI is its **point-to-point bus topology**.
- Each device sits on its own **dedicated bus**, which in PCIe lingo is called a **link**.



Technologies overview – PCIe

Links and lanes

- A **lane** represents a set of differential signal pairs (one pair for Tx, one pair for Rx).
- A **link** represents a dual-simplex communications channel between two components.
- To scale bandwidth, a **link** may aggregate multiple **lanes** denoted by xN (x1, x2, x4, x8, x12, x16, and x32).



Technologies overview – PCIe
Transactions

- PCIe implements the first layers of the OSI stack.
  - Much more robust than PCI

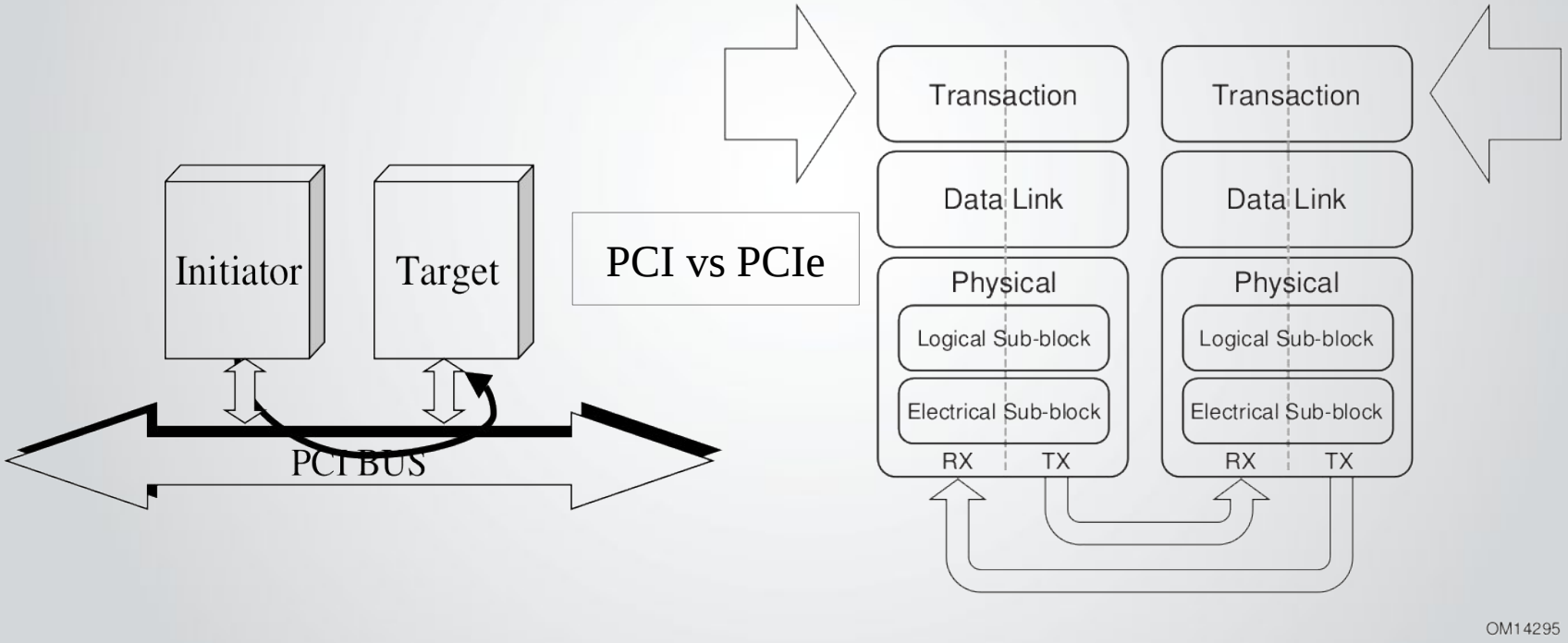
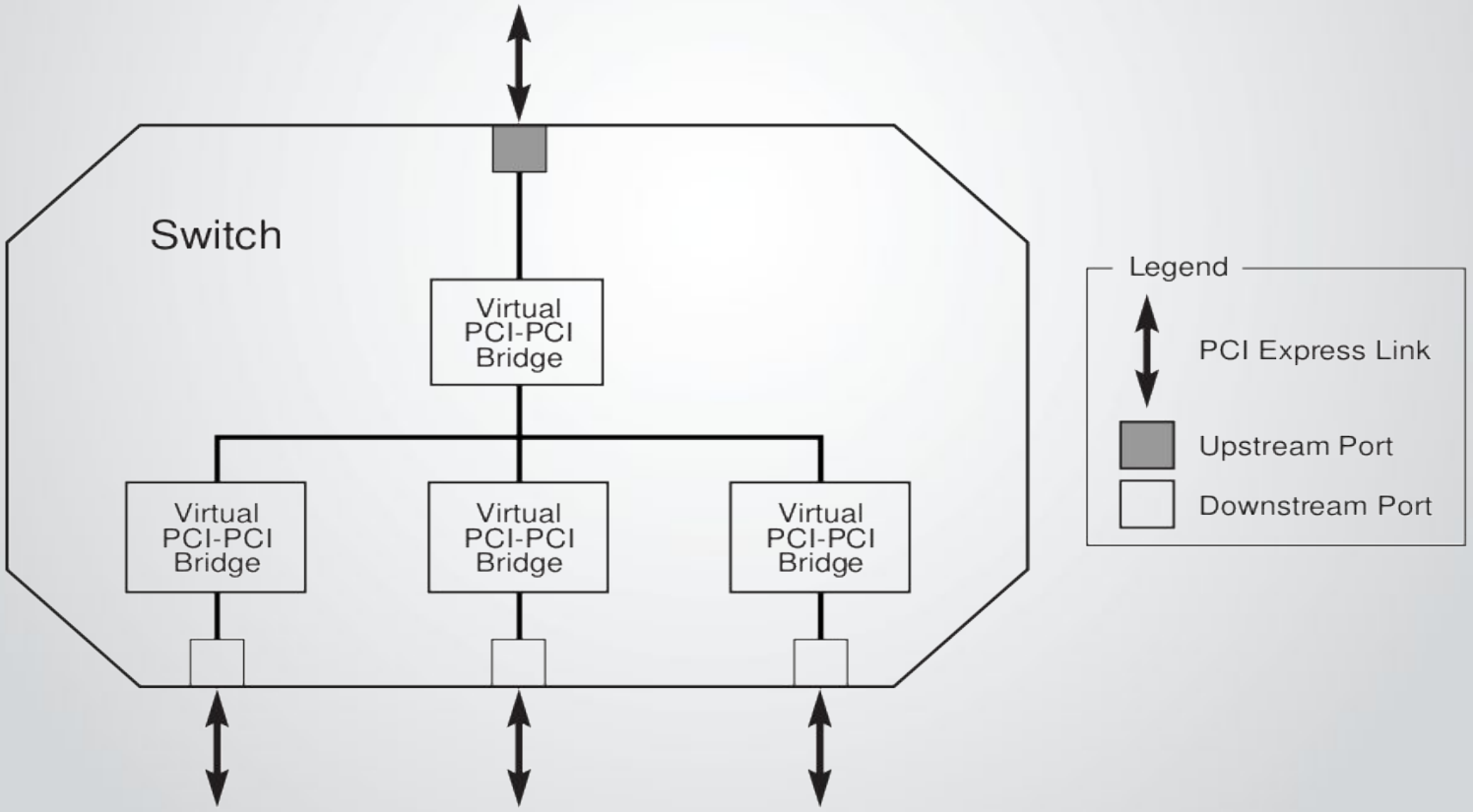


Figure 2-1: Layering Diagram Highlighting the Transaction Layer

Technologies overview – PCIe
Switches

- A **Switch** is defined as a logical assembly of multiple virtual PCI-to-PCI Bridge devices.



Agenda (PCI vs PCIe)

- PCI Evolution
- Technologies overview
- **Topology differences**
- Configuration
- PCI Hot Plug
- Interrupts handling

Topology - PCI

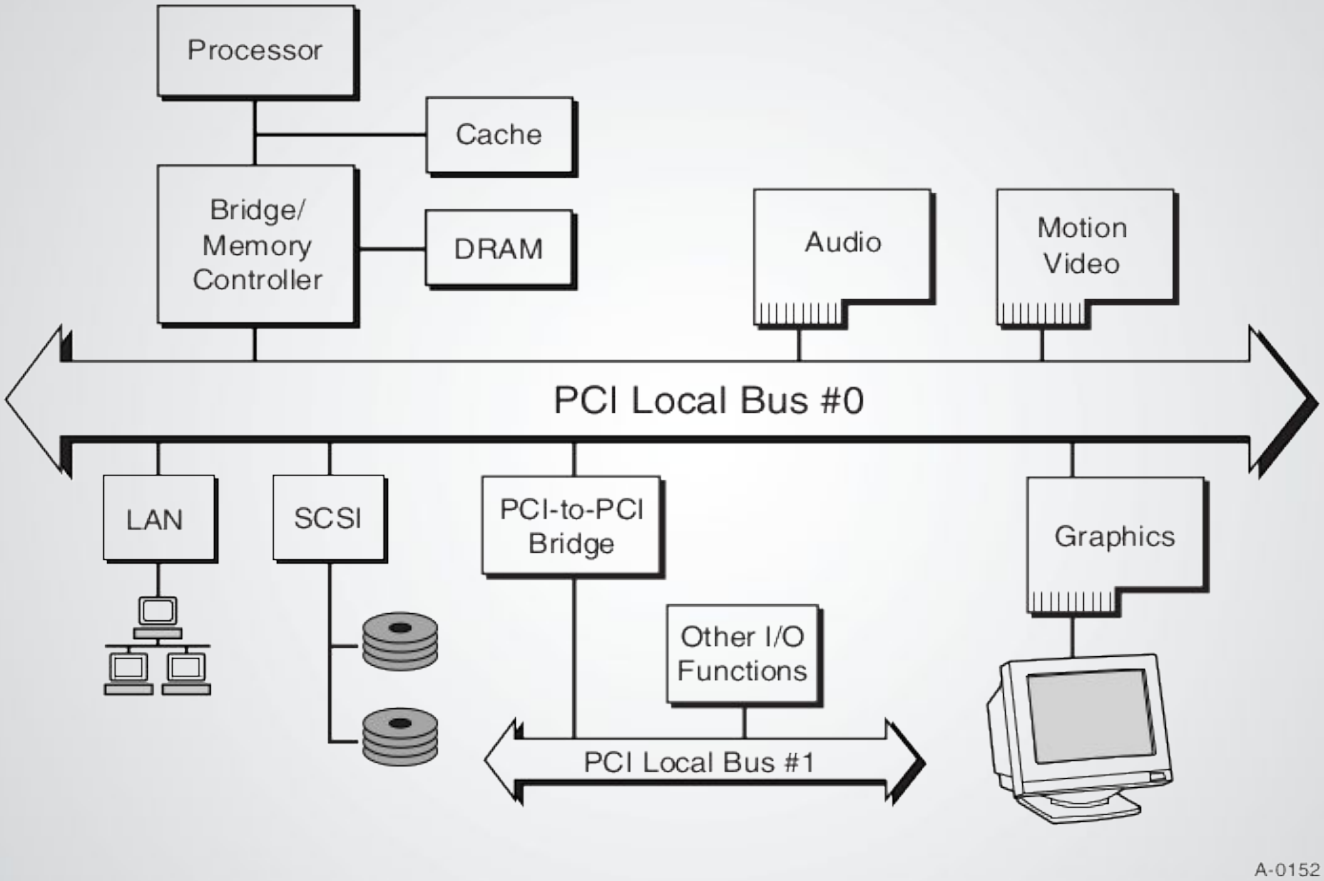


Figure 1-2: PCI System Block Diagram

Topology - PCIe

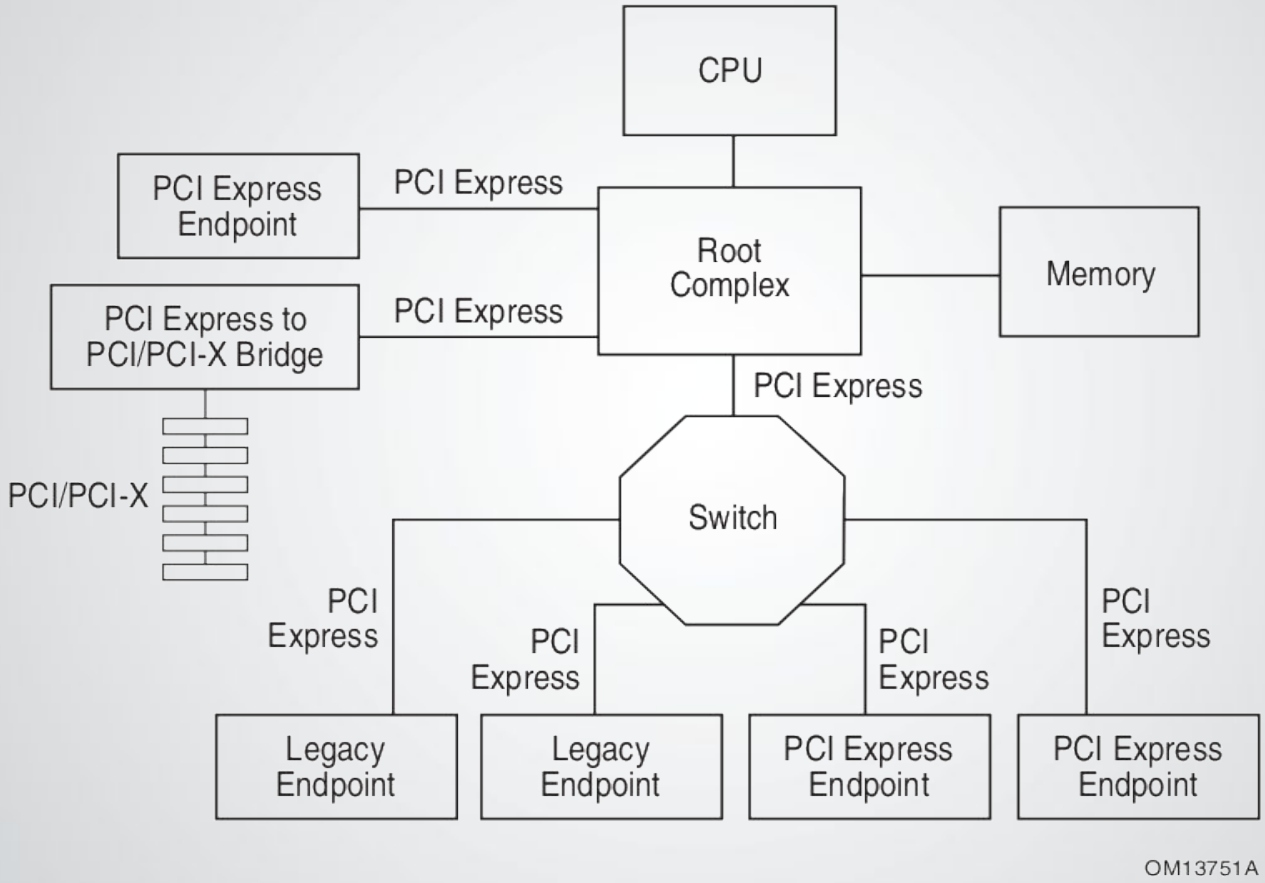
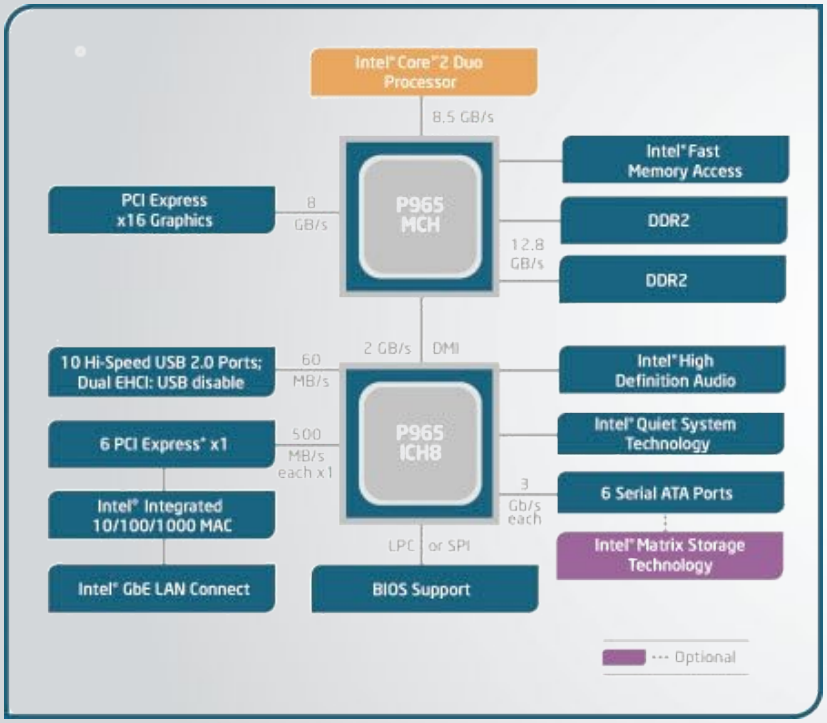
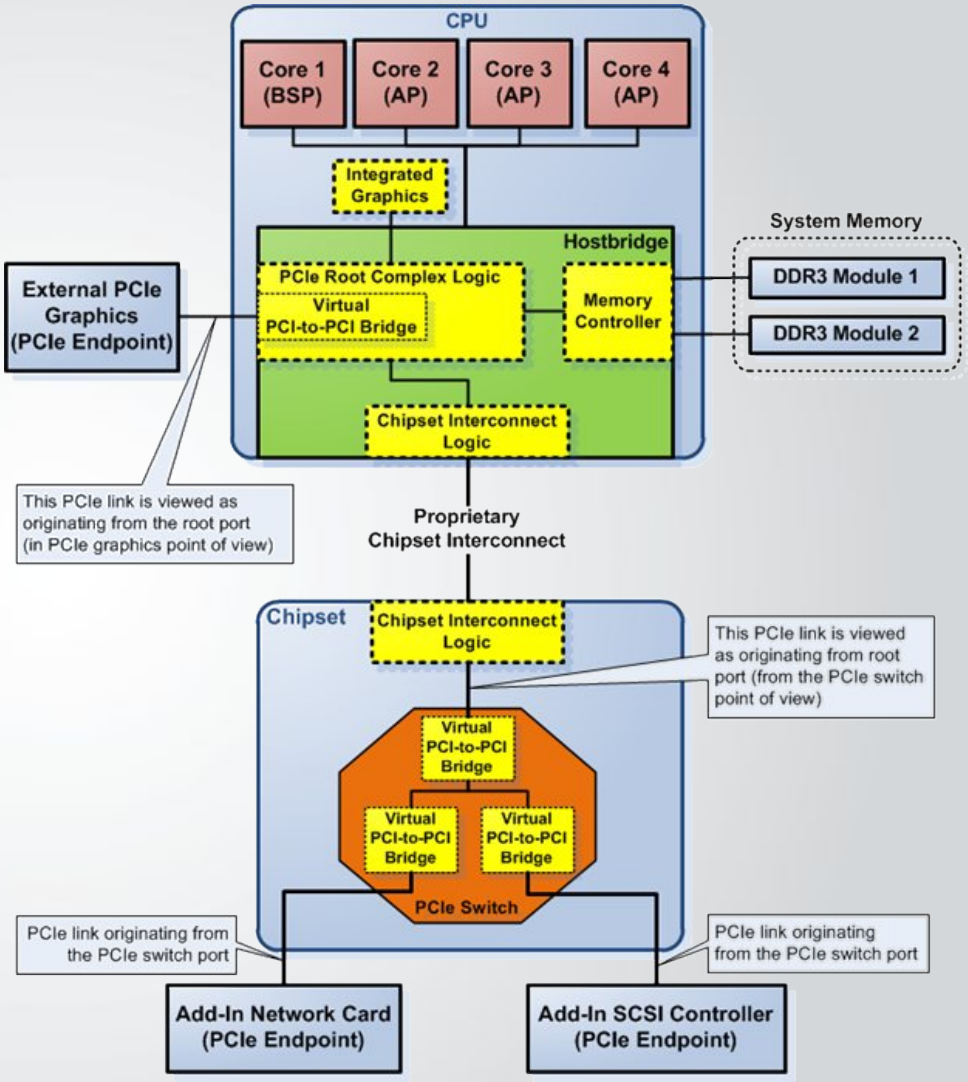


Figure 1-2: Example Topology

Topology – PCIe
Root Complex location





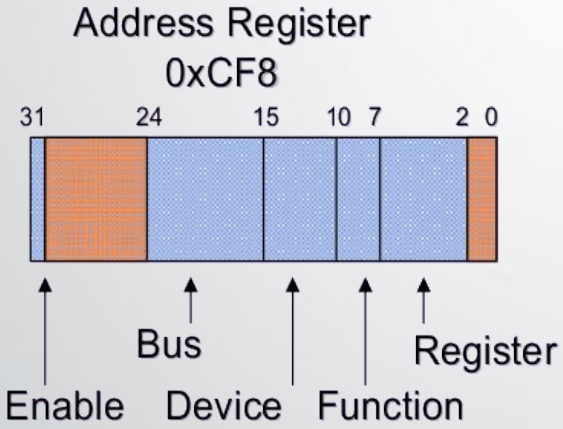
Agenda (PCI vs PCIe)

- PCI Evolution
- Technologies overview
- Topology differences
- **Configuration**
- PCI Hot Plug
- Interrupts handling

Configuration

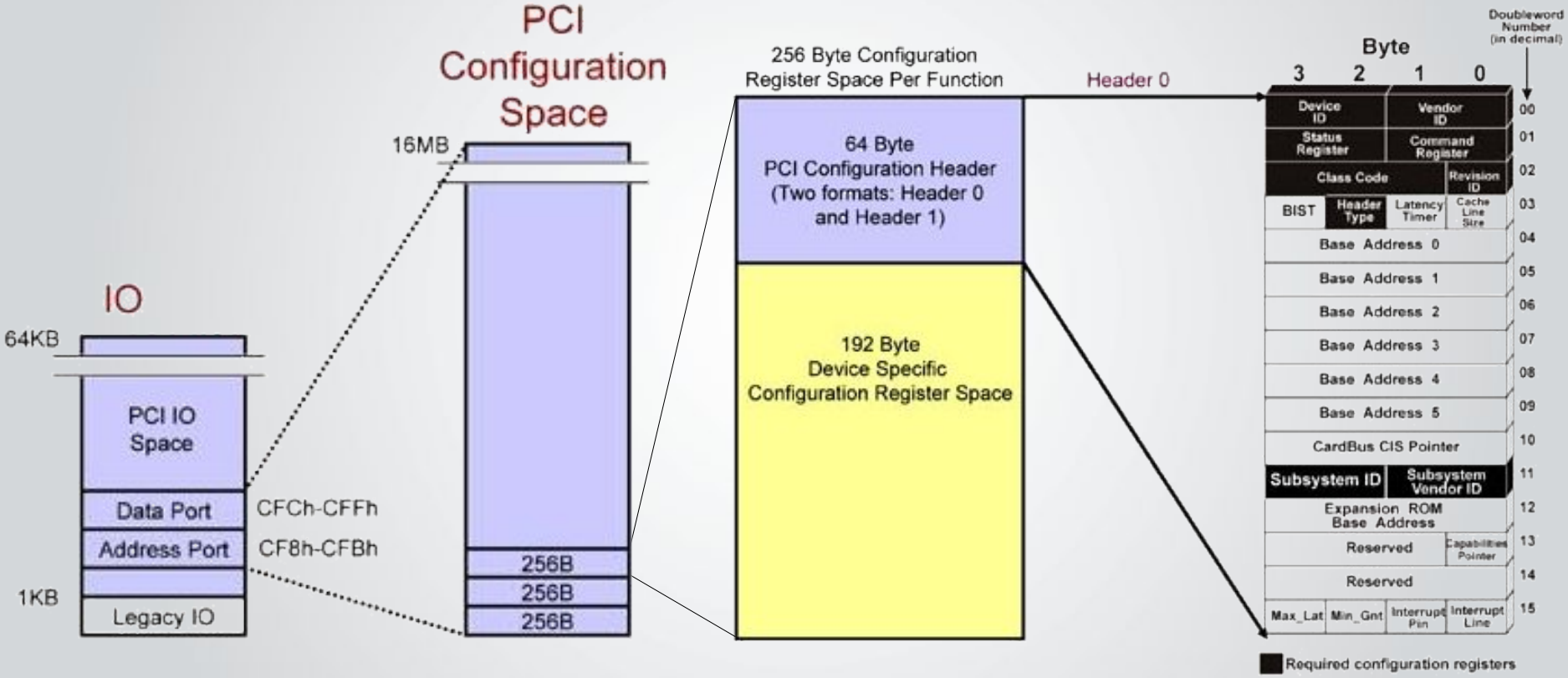
PCI Configuration space triggering

- I/O mapped
- Addresses identical across machines
- Only supports 256 bytes/device
- Only supports 256 buses



Configuration

PCI Configuration space



Configuration

PCI Configuration – Command register

- Provides control over a device's ability to generate and respond to PCI cycles.
- 0 => device is logically **disconnected** from the PCI bus.

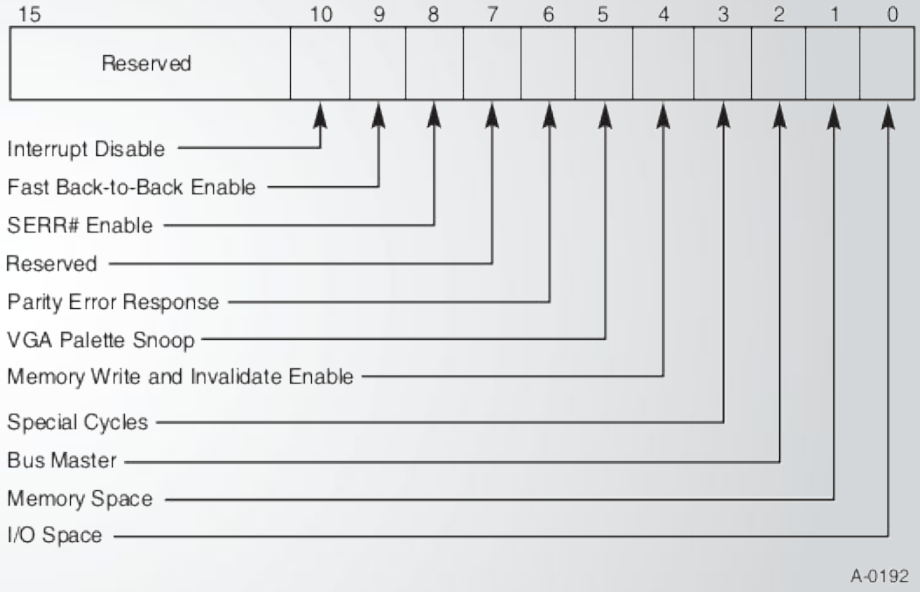


Figure 6-2: Command Register Layout

Configuration

PCI Configuration – Status register

- The Status register is used to record status information for PCI bus related events.
- Devices may not need to implement all bits, depending on device functionality.

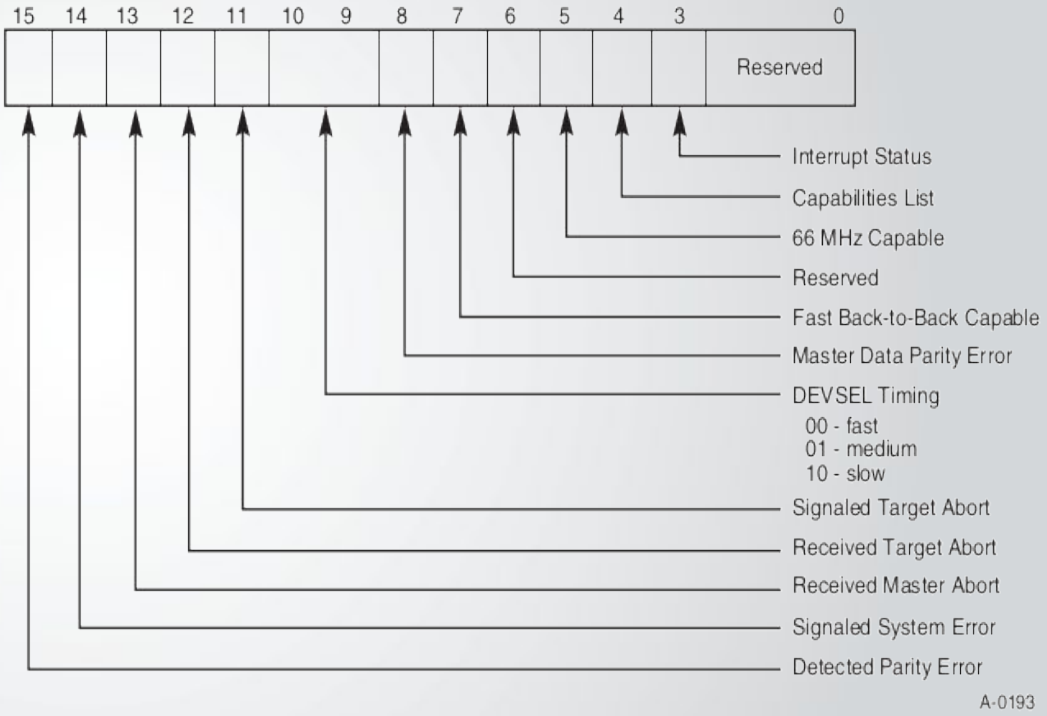


Figure 6-3: Status Register Layout

Configuration

PCI Configuration – Capabilities

- A set of registers added to a linked list
- Enabled if “Capabilities” bit in Status Register is “ON”
- Capability Examples:
  - MSI / MSI-X
  - PCI-X
  - PCIe

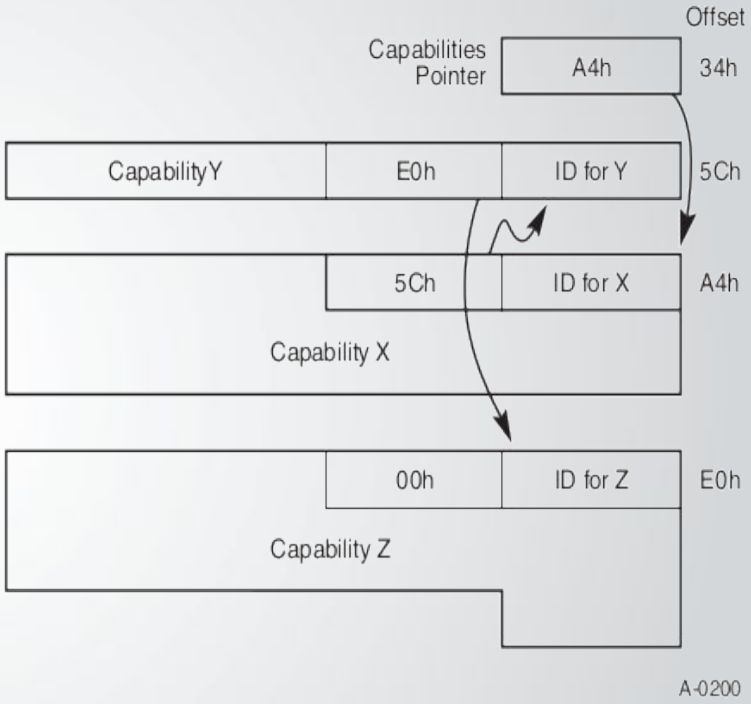


Figure 6-8: Example Capabilities List

Configuration

PCI Configuration – Base Address Registers (BARs)

- BAR Dual usage:
  - Used to determine how much memory space or I/O space the device requires.
  - Stores the base address of the memory region which is used to access the device registers.

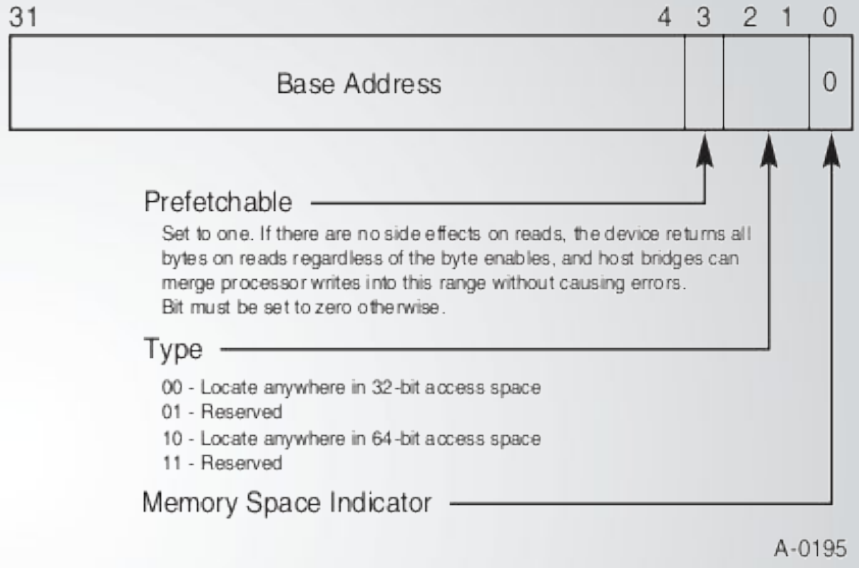
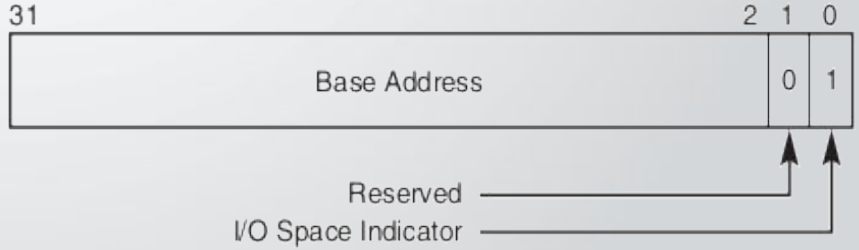


Figure 6-5: Base Address Register for Memory



Configuration

Demo

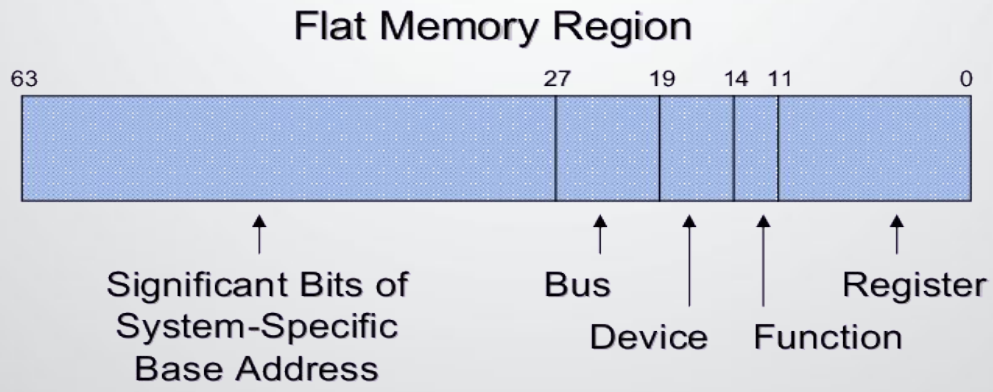
```
[root@localhost ~]# lspci -v -s 01:00.0
01:00.0 Ethernet controller: Red Hat, Inc Device 1041 (rev 01)
        Subsystem: Red Hat, Inc Device 1100
        Flags: bus master, fast devsel, latency 0, IRQ 23
        Memory at c1800000 (32-bit, non-prefetchable) [size=4K]
        Memory at c1000000 (64-bit, prefetchable) [size=8M]
        Expansion ROM at c1840000 [disabled] [size=256K]
        Capabilities: [dc] MSI-X: Enable+ Count=3 Masked-
        Capabilities: [c8] Vendor Specific Information: VirtIO: <unknown>
        Capabilities: [b4] Vendor Specific Information: VirtIO: Notify
        Capabilities: [a4] Vendor Specific Information: VirtIO: DeviceCfg
        Capabilities: [94] Vendor Specific Information: VirtIO: ISR
        Capabilities: [84] Vendor Specific Information: VirtIO: CommonCfg
        Capabilities: [7c] Power Management version 3
        Capabilities: [40] Express Endpoint, MSI 00
        Kernel driver in use: virtio-pci
        Kernel modules: virtio_pci

[root@localhost ~]# setpci -s 01:00.0 COMMAND
0407
[root@localhost ~]# setpci -s 01:00.0 STATUS
0010
[root@localhost ~]# setpci -s 01:00.0 BASE_ADDRESS_0
00000000
[root@localhost ~]# setpci -s 01:00.0 BASE_ADDRESS_1
c1800000
[root@localhost ~]# setpci -s 01:00.0 BASE_ADDRESS_2
00000000
[root@localhost ~]# setpci -s 01:00.0 BASE_ADDRESS_3
00000000
[root@localhost ~]# setpci -s 01:00.0 BASE_ADDRESS_4
c100000c
[root@localhost ~]# setpci -s 01:00.0 BASE_ADDRESS_5
00000000
```

Configuration

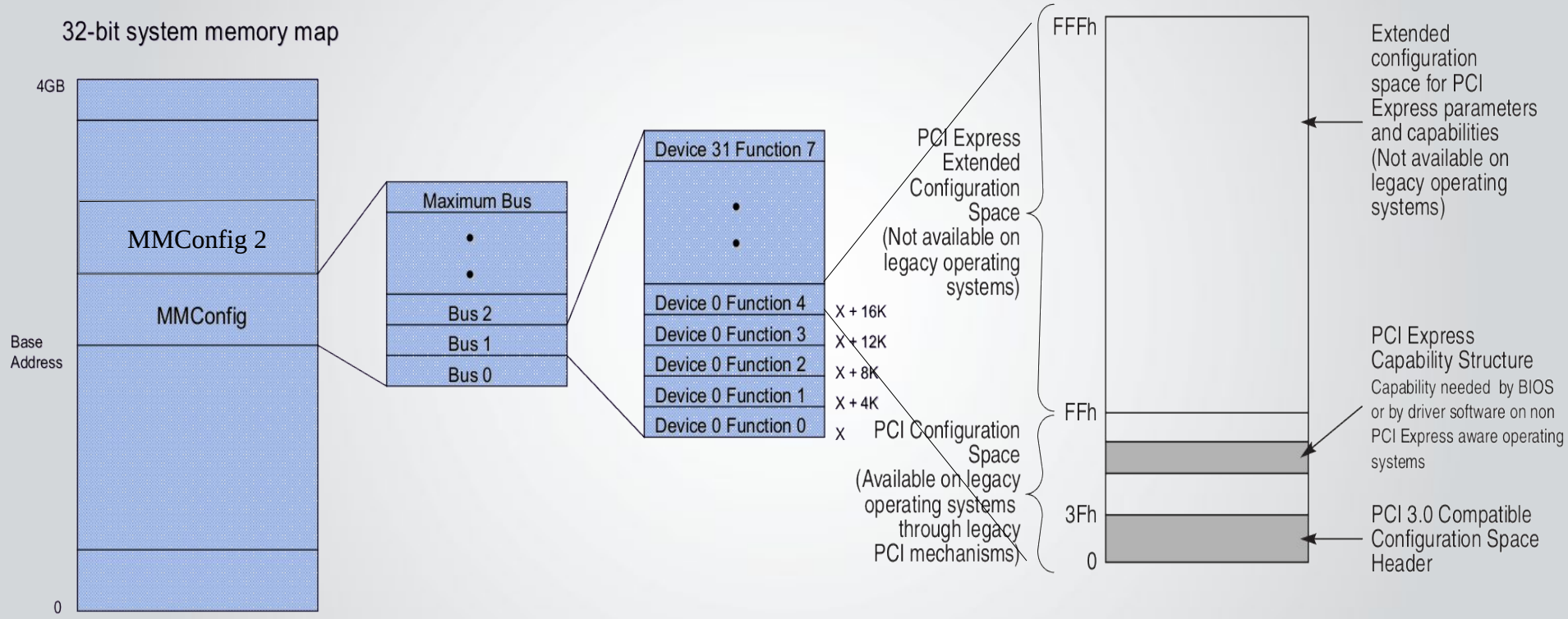
Configuration space triggering – PCIe (ECAM)

- Memory Mapped
- Supports 4K bytes/device
  - Each device has its own 4K memory page.
- Requires up to 28 bits of memory
  - 256 MB
- Supports 256 buses **per base address.**



Configuration

PCIe Configuration space



Configuration
PCIe Configuration – PCIe Extended Capabilities

- It is not the PCI Express “capability”.
- Begins at offset 100h.
- Resembles PCI Capability structure.

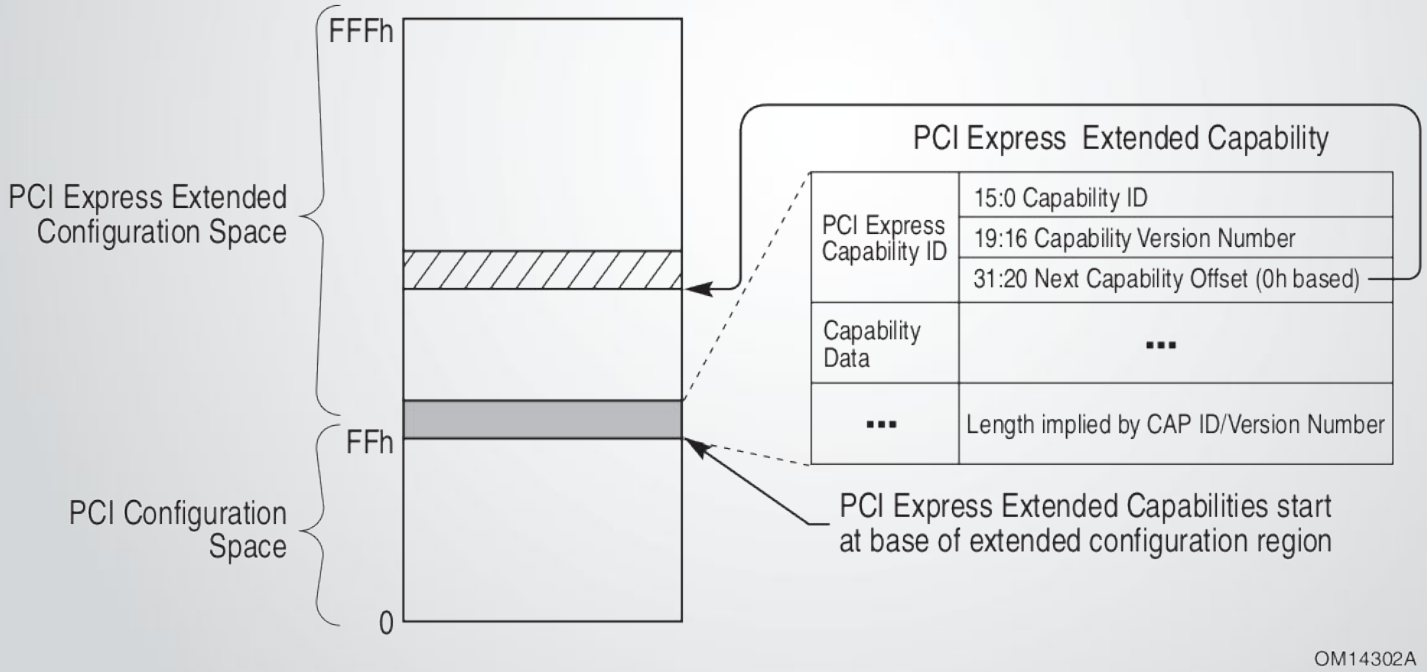


Figure 7-30: PCI Express Extended Configuration Space Layout

Configuration

PCIe Configuration – ARI Extended Capability

- ARI (Alternative Routing-ID Interpretation) Device
- Motivation
  - PCI: 32 dev * 8 functions per bus
  - PCIe: 1 dev * 8 functions per bus
- Device is implied 0
  - <bus, device, function> replaced by <bus, function>

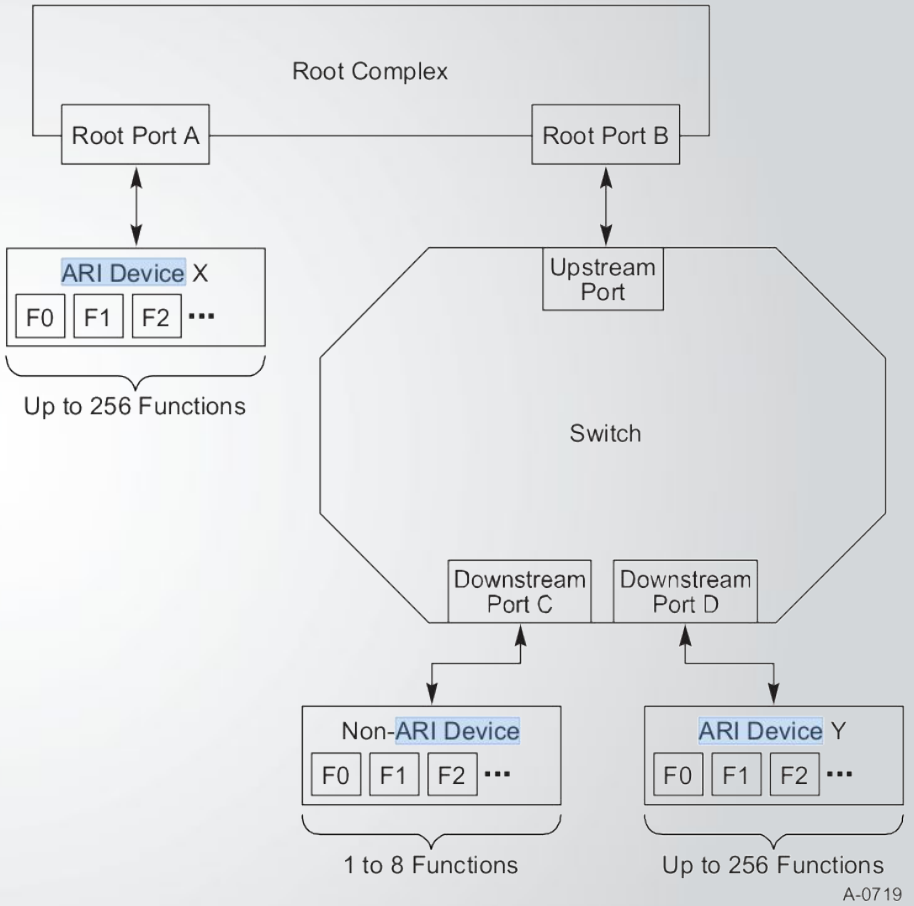


Figure 6-13: Example System Topology with ARI Devices

Configuration

PCIe Configuration – SR-IOV Extended Capability

- Configure a device to appear in PCI configuration space as multiple functions.
- Two new function types:
  - PFs: full functions
  - Vfs: “lightweight” functions
- Leverages ... ARI

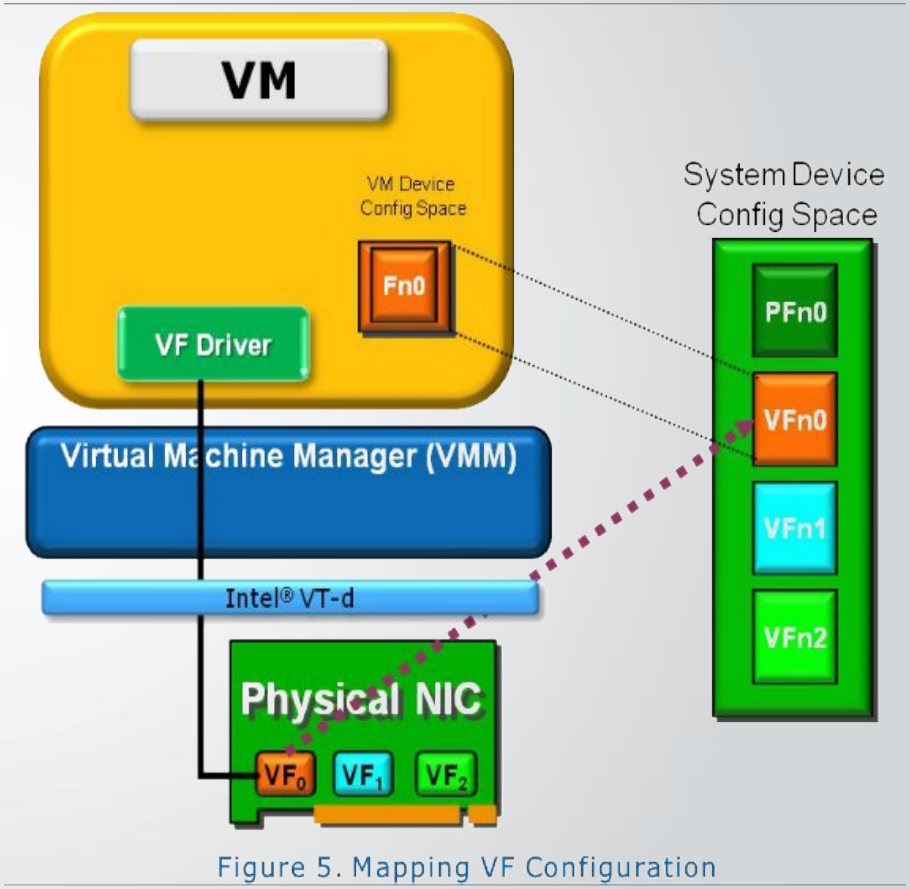


Figure 5. Mapping VF Configuration

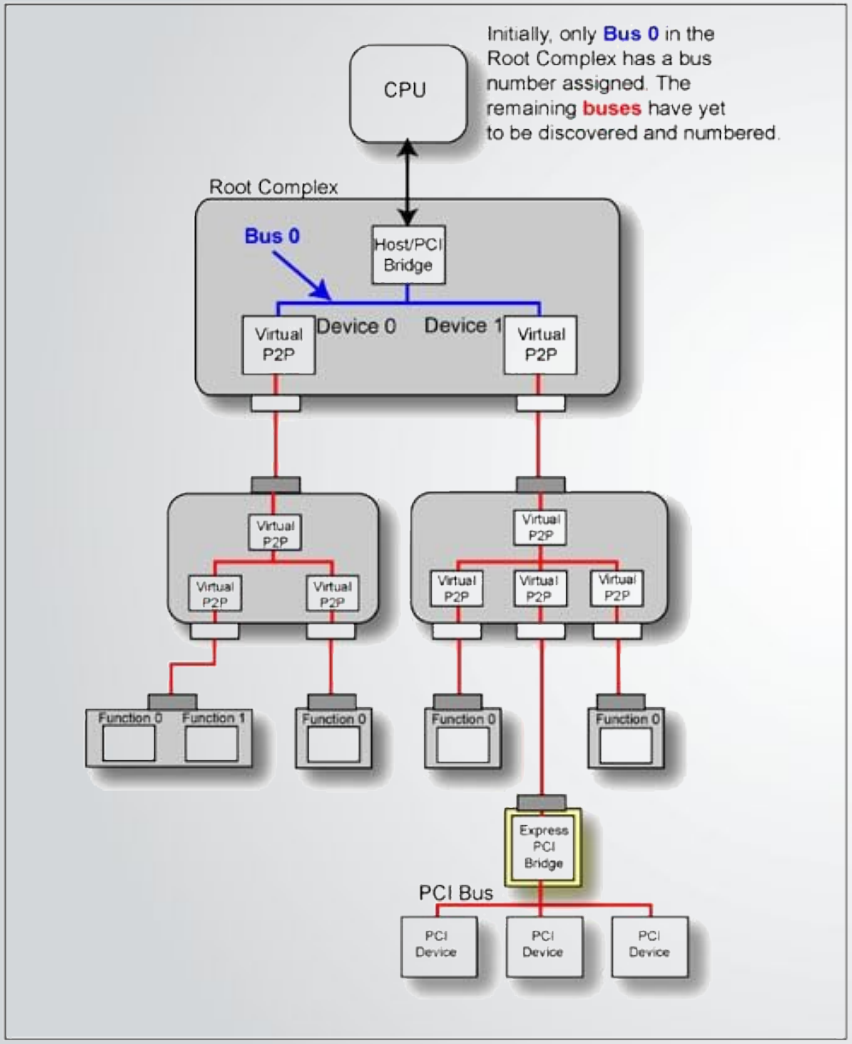
Configuration

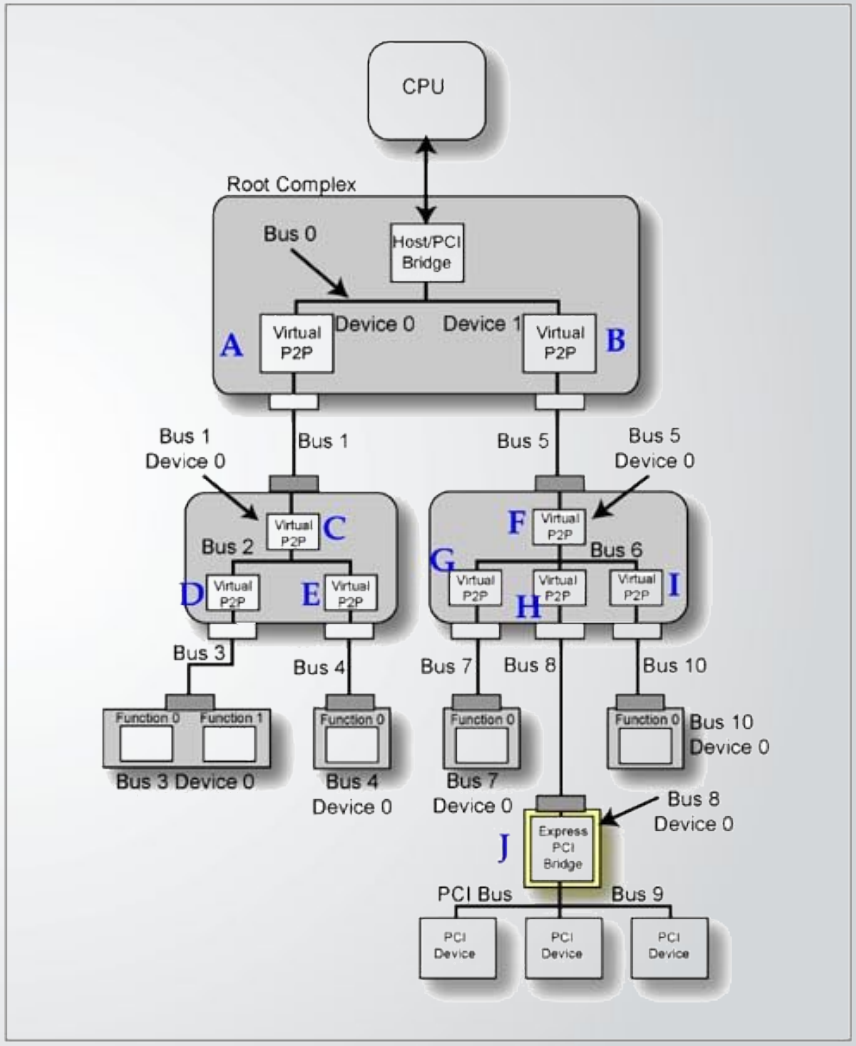
PCIe Configuration – Other Extended Capabilities

- Function Level Reset (FLR)
- Advanced Error Reporting (AER)
- Access Control Services (ACS)
- ...

Configuration

Enumeration & Resources allocation

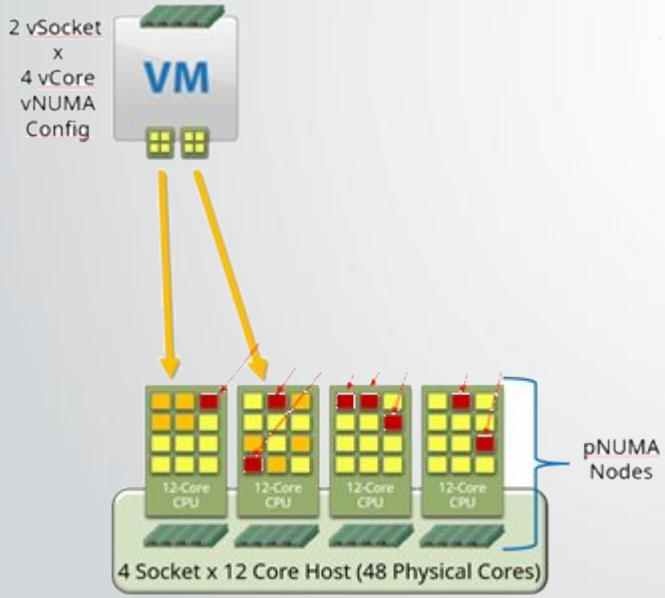


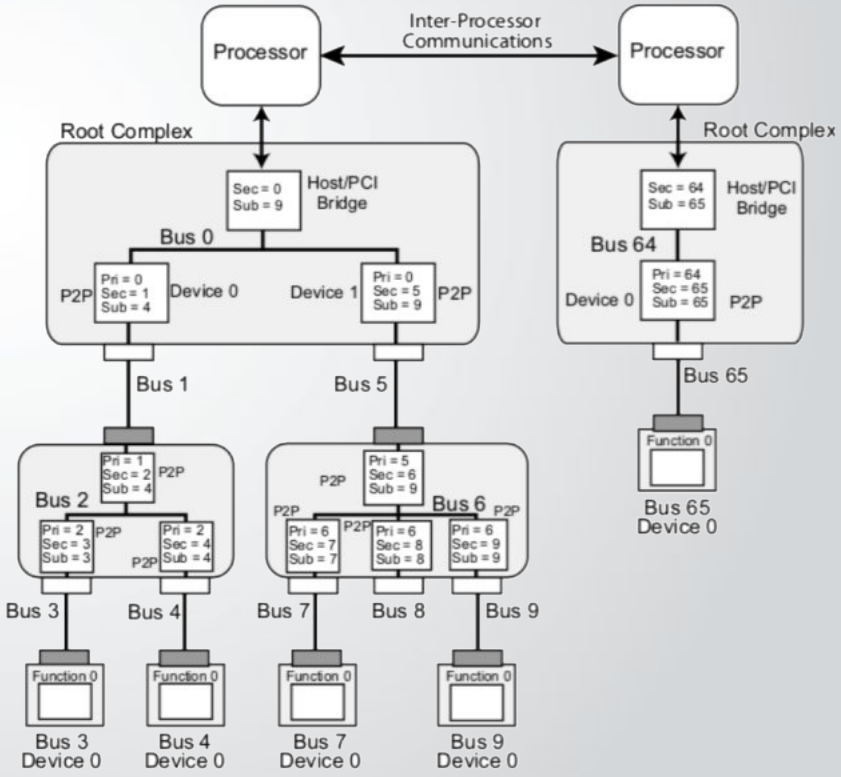


Configuration

Multiple host bridges / Root complexes

- NUMA – how servers scales up
- A PCI Host Bridge can be connected to a single NUMA node
- Device assignment for a VM with multiple NUMA nodes





Agenda (PCI vs PCIe)

- PCI Evolution
- Technologies overview
- Topology differences
- Configuration
- **PCI Hot Plug**
- Interrupts handling

PCI Hot Plug
Introduction

- PCI Hot Plug (PHP) is the concept of removing or inserting a standard PCI adapter card from a system without interrupting normal system operation or powering-down the system as a whole.
- History
  - 1997: PCI Hot Plug introduced
  - 2001: Standardized Hot Plug Usage Model and the hot plug controller(SHPC)
  - 2002: PCIe Hot plug developed as part of PCIe base specification
- ACPI Hot plug, as part of ACPI spec
  - Out of the scope of this presentation

PCI Hot Plug
SHPC

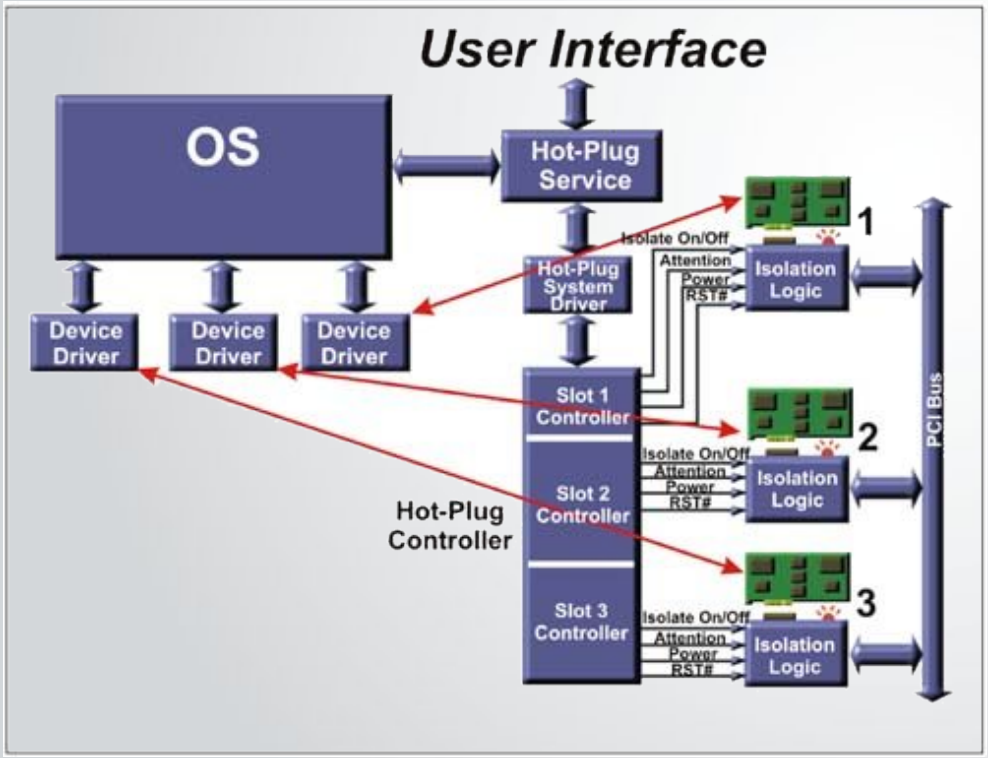
The SHPC must be integrated into a PCI-to-PCI bridge or host bridge.

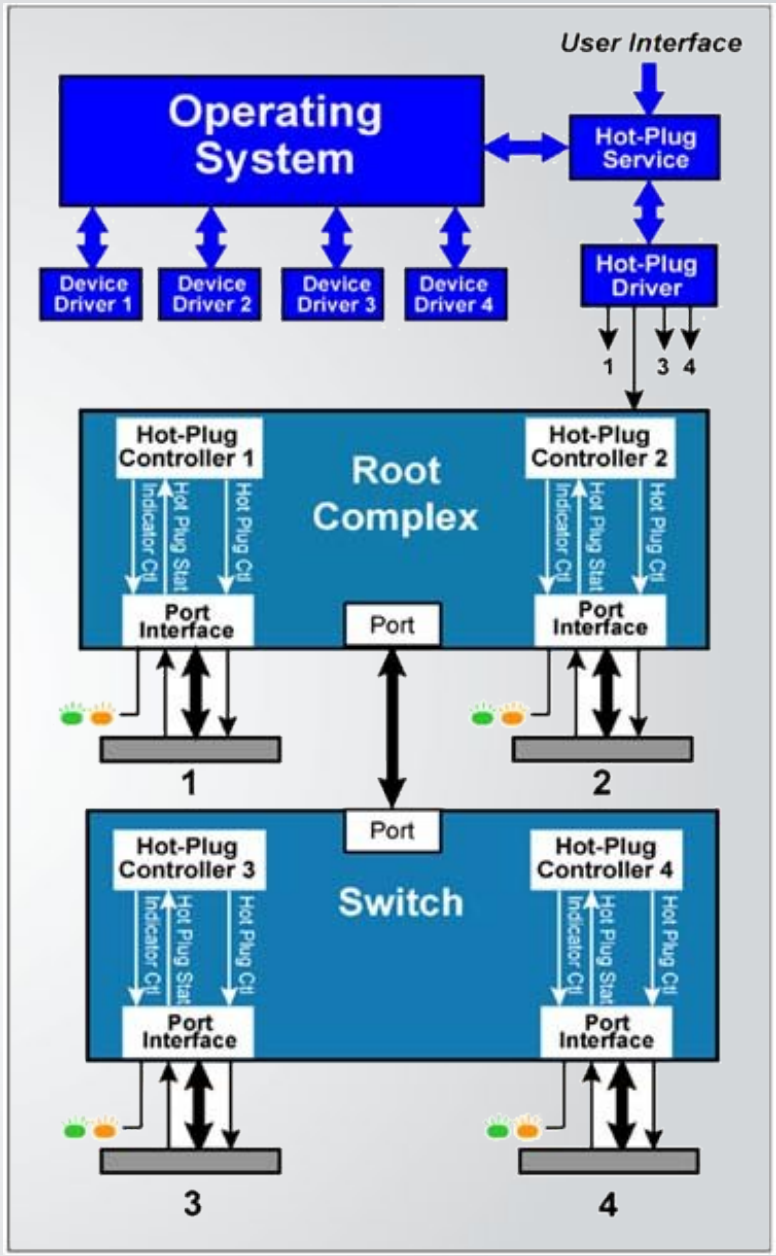
Hot Plug controllers must:

- assert and deassert the reset signal to the PCI Express card
- remove or apply power to the card connector
- turn on or turn off the Power and Attention Indicators associated with a specific card connector to draw the user's attention to the connector and advertise whether power is applied to the slot.
- Monitor slot events (e.g. card removal) and report these events to software via interrupts.

PCI Hot Plug

SHPC - PCI vs PCIe





PCI Hot Plug

Software elements of Standard Hot Plug Usage Model

User Interface	Allows user to request hot-plug operations
Hot-Plug System Driver / Hot-Plug Service	Receives requests issued by the OS and Interacts with the hardware Hot-Plug Controllers to accomplish requests. • provide slot identifiers • turn device On/Off • turn Attention Indicator On/Off • set current state of slot On/Off
Device Driver	Hot-Plug-specific capabilities must be incorporated in a Hot-Plug capable device driver • support for the Quiesce command
Firmware	Ensure unused IO/MEM regions remain for the use of the new PCI devices

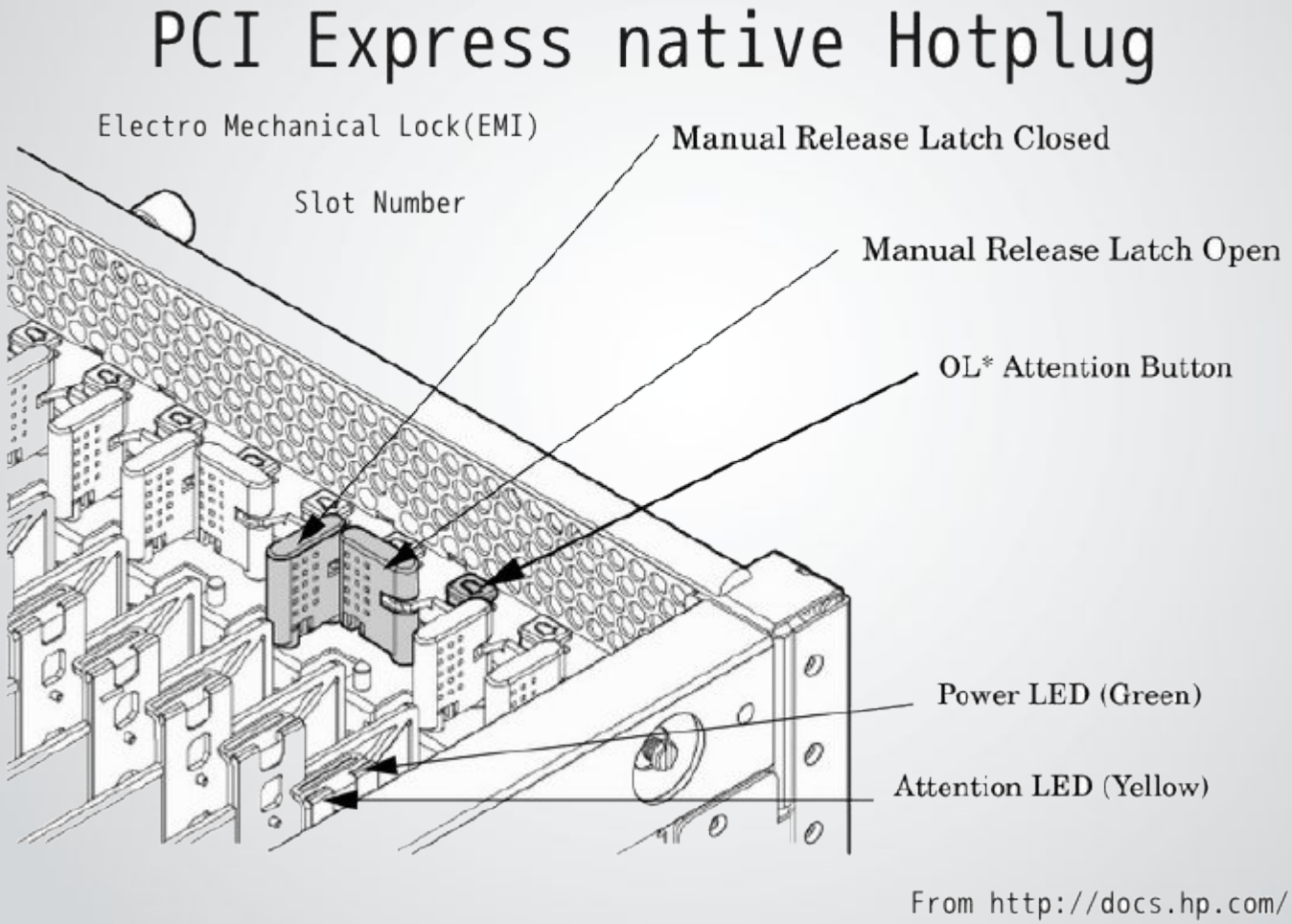
PCI Hot Plug

Hardware Elements of Standard Hot Plug Usage Model

Hot-Plug Controller	Receives and processes commands issued by the Hot-Plug Device Driver
Power Indicator	Indicates the slot/card for service
Attention Indicator	The Attention Indicator is used to draw the attention of the operator or to indicate a Hot Plug problem or failure.
Attention Button	Allows user to request hot-plug operations
Manually-operated Retention Latch (MRL)	Holds add-in cards in place
MRL Sensor	Allows the port and system software to detect the MRL being opened
Electromechanical Interlock	Prevents removal of add-in cards while slot is powered

PCI Hot Plug

Hardware Elements of Standard Hot Plug Usage Model



PCI Hot Plug
Hot Add Sequence

- The operator installs the card and secures the MRL
- The MRL sensor(if present), causes a system interrupt to be sent to the Root Complex signaling the SHPC driver that the latch is closed
- User initiates hot add using Attention Button or UI
- The SHPC driver causes the slot's Power Indicator to blink for 5 seconds, this being the window to cancel the operation and also the time for the software to validate the request.
- The SHPC driver issues a request to the hot plug controller to power on the slot
- Once link training is complete, the Hot plug driver causes re-enumeration of the slot bus; The hot added device is found, configured and driver is loaded.

PCI Hot Plug
Hot Remove Sequence

- The user requests hot removal by pressing the Attention Button on the corresponding slot or by a software request.
- The SHPC detects this event and delivers an interrupt to the Root Complex. As a result of the interrupt, the Hot Plug System Driver reads slot status information and detects the Attention Button request.
- Power LED blinks to indicate transition state. The operator is granted a 5 second interval to cancel the request and the Hot Plug software validates the request.
- OS offlines the PCI Express device: the Hot-Plug System Driver commands the card's device driver to quiesce.
- Next, software commands the Hot Plug Controller to turn the slot off.
- User opens the MRL and the card can now be removed.
- The OS deallocates the memory space, IO space, interrupt line, etc.

PCI Hot Plug
Hot Plug Registers

- PCI Express allows buttons/indicators to be placed either on the chassis or the card
  - Buttons/Indicators on card can further be handled side-band or in-band
- Buttons/Indicators implemented on the chassis are indicated using **Slot Capabilities Register**
  - Other Hot Plug related Registers: **Slot Control Register, Slot Status Register**
- Buttons/Indicators implemented on the card are indicated using **Device Capabilities Register**

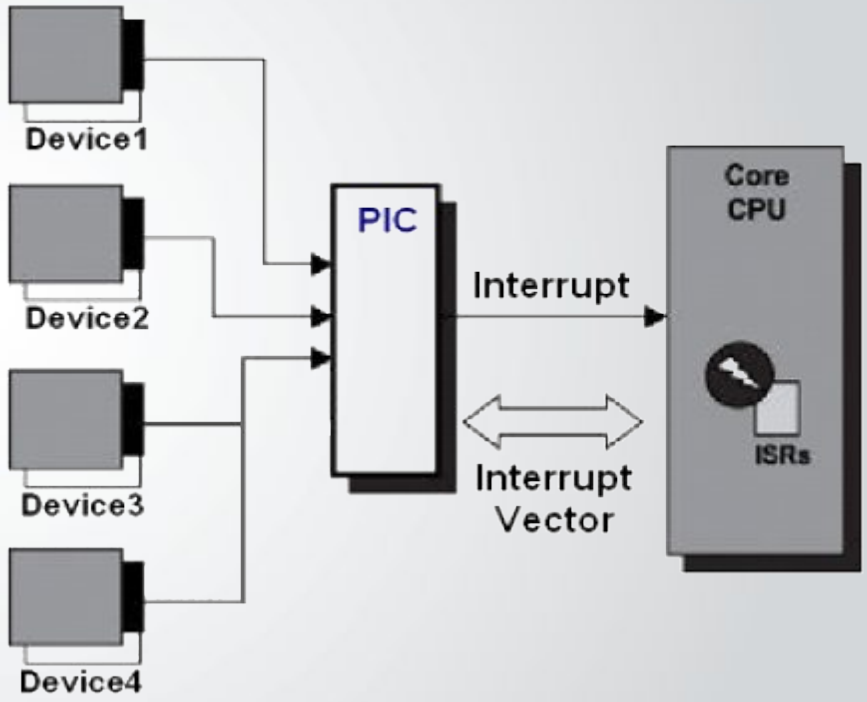
Agenda (PCI vs PCIe)

- PCI Evolution
- Technologies overview
- Topology differences
- Configuration
- PCI Hot Plug
- **Interrupts handling**

Interrupts handling

Legacy interrupts

- Legacy interrupts are dedicated side-band signals.
- Multiple interrupt signals may share a single physical line.



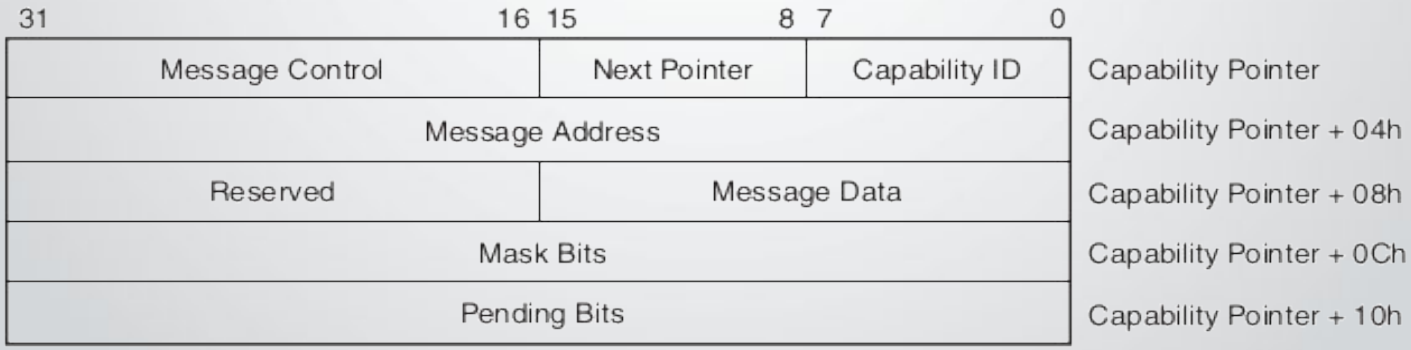
Interrupts handling
MSI

- In-band messages are implemented as memory writes to an address with a data value, as specified by software.
  - Required for PCI Express devices, optional for PCI devices
  - Maximum of 32 MSIs per function
- MSI has a number of distinct advantages over INTx
  - Sharing of interrupt vectors is eliminated
  - Devices may have multiple interrupts per function

Interrupts handling
MSI capability structure

- To request service, an MSI function writes the contents of the Message Data register to the address specified by the contents of the Message Address register.
- Per-vector masking (optional) is managed through a **mask** and **pending** bit pair per MSI vector

Capability Structure for 32-bit Message Address and Per-vector Masking



Interrupts handling
MSI-X

- MSI-X has the same features as MSI, the key differences are:
  - Maximum of 2048 MSI-Xs per function
  - MMIO region required for MSI-X tables and Pending Bit Arrays
  - Per function vector masking and per vector masking (optional for MSI)

Interrupts handling
MSI-X capability structure

- The capability structure points to:
  - A MSI-X Table structure
  - A Pending Bit Array (PBA) structure.
- Each structure is mapped by a BAR.

DWORD 3	DWORD 2	DWORD 1	DWORD 0		
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	entry 0	Base
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	entry 1	Base + 1*16
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	entry 2	Base + 2*16
...	...	...	...	...	...
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	entry (N-1)	Base + (N-1)*1

A-038

Figure 6-11: MSI-X Table Structure

63	0		
Pending Bits 0 through 63		QWORD 0	Base
Pending Bits 64 through 127		QWORD 1	Base + 1*8
...		...	...
Pending Bits ((N-1) div 64)*64 through N-1		QWORD ((N-1) div 64)	Base + ((N-1) div 64)*8

Thank you!