



# I/O Virtualization and Sharing

**Michael Krause (HP, co-chair)**  
**Renato Recio (IBM, co-chair)**



# Outline

- Virtualization Technology Overview and Terminology
- Scope of Work
  - ✓ Overview
  - ✓ Single Root Requirements
  - ✓ Multi-Root Requirements
  - ✓ Address Translation Services Requirements

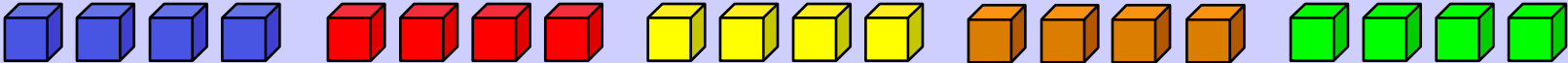


# Overview



# What is Virtualization?

- **System Virtualization** - *The division of a physical system's processors, memory, I/O, and storage, where each such set of resources operates independently with its own System Image instance and applications.*



**Virtual Resources**

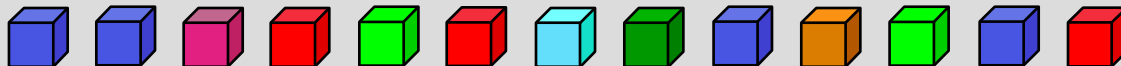
- Proxies for physical resources and have the **same external interfaces and functions**.
- Composed from physical resources.

## Virtualization Intermediary

- Creates virtual resources and "maps" them to physical resources.
- Provides isolation between Virtual Resources.
- Accomplished through a combination of software, firmware, and hardware mechanisms.

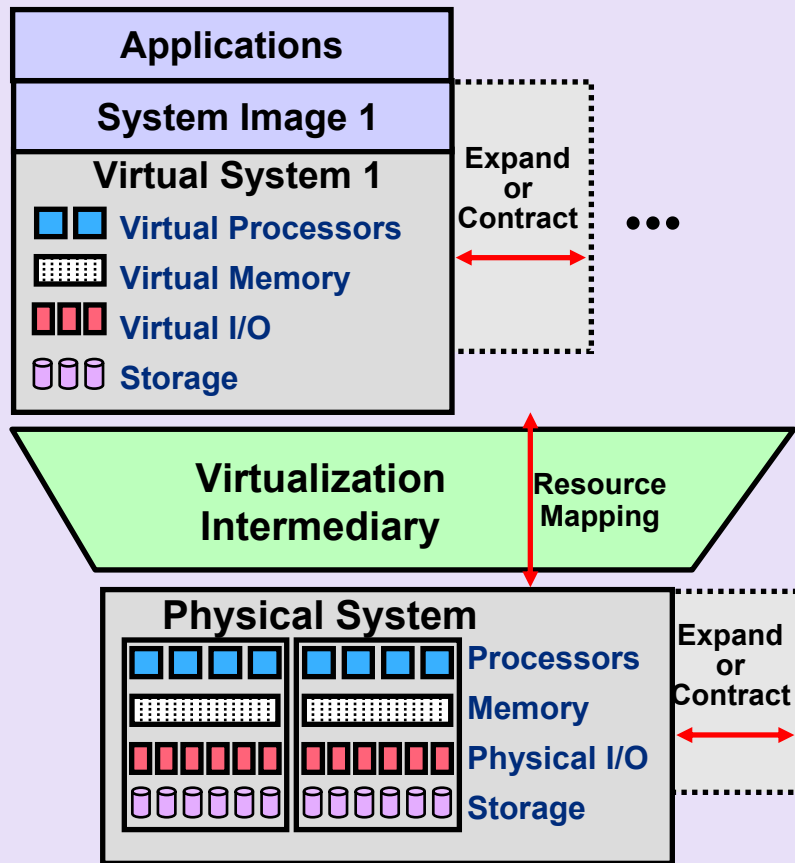
## Physical Resources

- Hardware components with **architected interfaces/functions**.
- Examples: memory, disk drives, networks, servers.



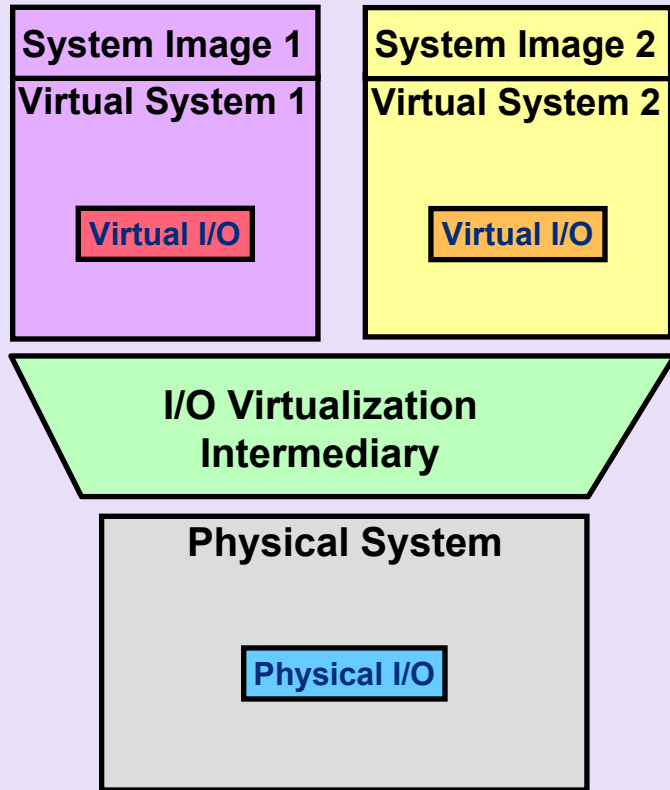
# Terminology

- **System Image (SI)** - A software component, such as a general or special purpose Operating System, to which specific virtual and physical devices can be assigned.
- **Virtualization Intermediary (VI)** - A component that manages the allocation of resources to an SI and isolates resources assigned to a System Image from access by other System Images.
- **Virtual System (VS)** - The physical or virtualized resources necessary to run a single SI instance. Virtual resources typically consist of: processors, memory, I/O, and storage.





# Terminology (cont.)

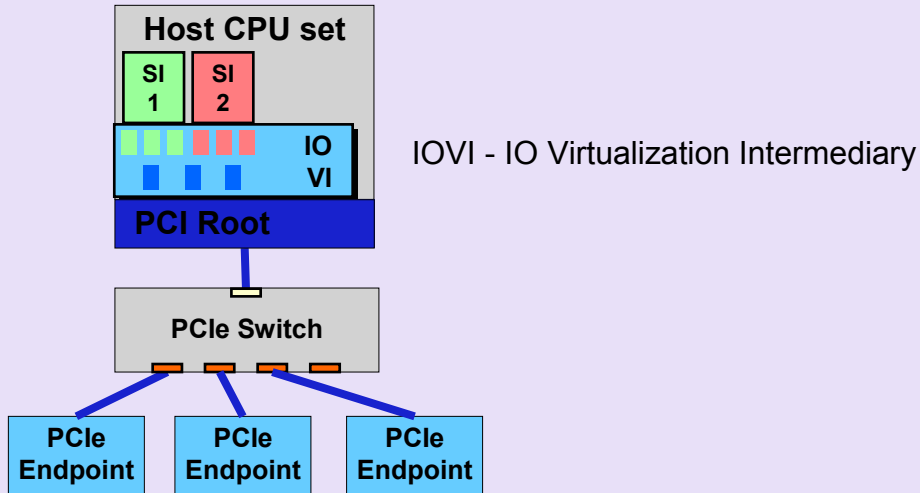


- ***I/O Virtualization (IOV)*** - The capability for a single physical I/O unit to be shared by more than one System Image.
- ***I/O Virtualization Intermediary (IOVI)*** - Software or firmware that is used to support IOV by intervening on one or more of the following: Configuration, I/O, and Memory operations from a System Image; and DMA, completion, and interrupt operations to a System Image.

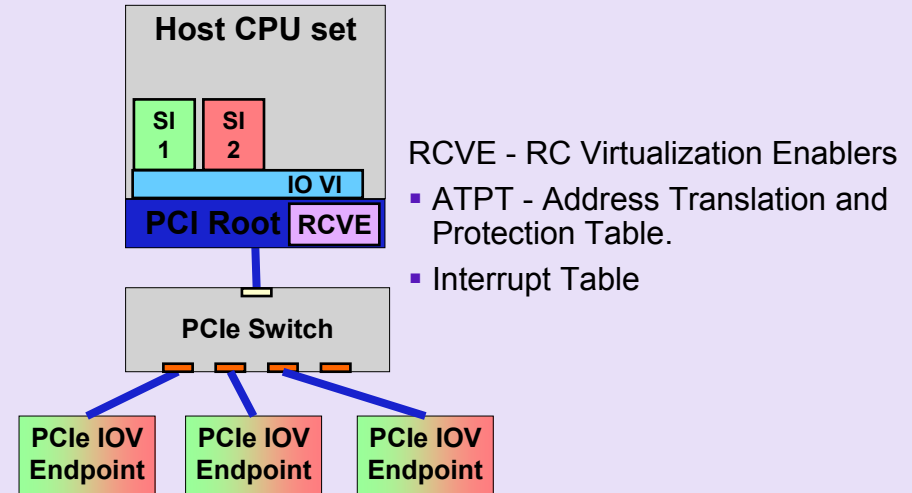
Several mechanisms have evolved in the industry, each with varying performance and function characteristics.

Strictly for background, some of the key ones will be covered next.

# PCI Sharing Approaches



- Adapter Shared Thru Intermediary
  - ✓ No RC virtualization enablers needed
  - ✓ One or more System Images
  - ✓ PCIe® EPs shared through IO VI:
    - IOVI is involved in all IO transactions and performs all IO Virtualization Functions, for example:
      - Multiplexes SIs' IO queues onto a single queue in the adapter.
    - PCIe EP is not required to support any virtualization functions.



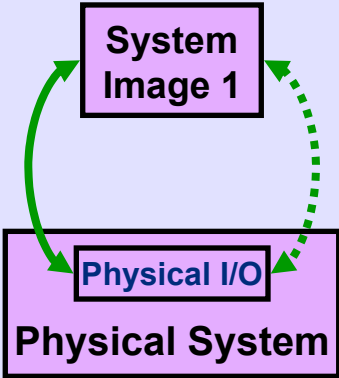
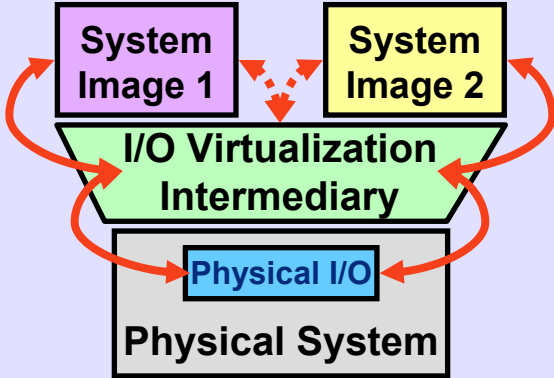
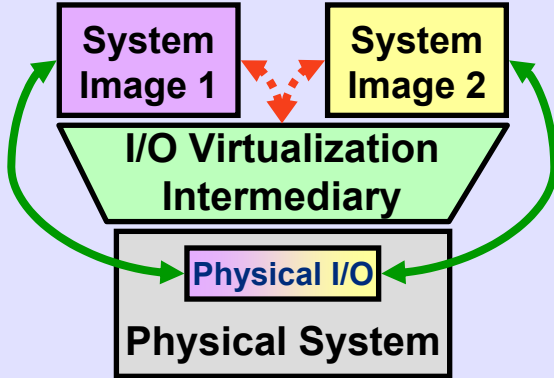
- Natively Shared Adapters
  - ✓ RC has virtualization enablers
  - ✓ One or more System Images
  - ✓ PCIe EPs shared through IO VI.
    - Same as IOVI without ATPT
  - ✓ PCIe IOV enabled EPs are directly shared.
    - IOVI is involved in configuration transactions.
    - Data transfers are direct between PCIe adapter and OS.
    - Requires PCIe adapter to support PCI SIG IOV specification.

# Terminology (cont.)

- **Configuration Time VI Involvement** – Refers to the operations performed by the VI to configure a Virtual Function and associate it with an SI.
- **Run-Time VI Involvement** – Refers to the intervention by a VI on data movement and interrupt operations performed between an SI and a physical PCI Function (versus a Virtual Function).
- **Native based PCI IOV** – An IOV mechanism where:
  - ✓ A physical PCI Endpoint supports one or more Virtual Functions.
  - ✓ Each VF of an IOV enabled Endpoint is associated with an SI.
  - ✓ IOV enabled Endpoint configuration ops are performed by the IOVI.
  - ✓ The Virtual System may support mechanisms that allow data movement and interrupt operations to be performed directly between an SI and a VF, without VI involvement.



# Adapter IOV Mechanisms within a Single Physical System

	Dedicated Adapter (No Virtualization)	Adapter Shared Through Intermediary	Natively Shared Adapter
Graphic Depiction			
Intermediary Role	None	Virtualizes physical I/O by intervening on configuration and data transfer operations	Manages assignment of Virtual Resources by intervening on configuration operations
Configuration Operation Path	SI direct to Adapter	VI serves as proxy (SI to VI; VI to Adapter)	VI serves as proxy (SI to VI; VI to Adapter)
Data Transfer Operation Path	SI direct to Adapter	VI serves as proxy (SI to VI; VI to Adapter)	SI direct to Adapter

# PCI Root Complex Virtualization Enablers

## ■ Address Translation and Protection Table

- ✓ Translates and protects system memory accessed through the PCI bus.
- ✓ Consists of the following functions:
  - On incoming DMA from a PCI adapter, translates the PCI Bus Address into an address that can be used to directly access host memory.
  - Provides **Physical Adapter Isolation** by assuring that when a PCIe function performs a DMA, the operation targets a System Image (SI) associated with that function.

## ■ Interrupt Table

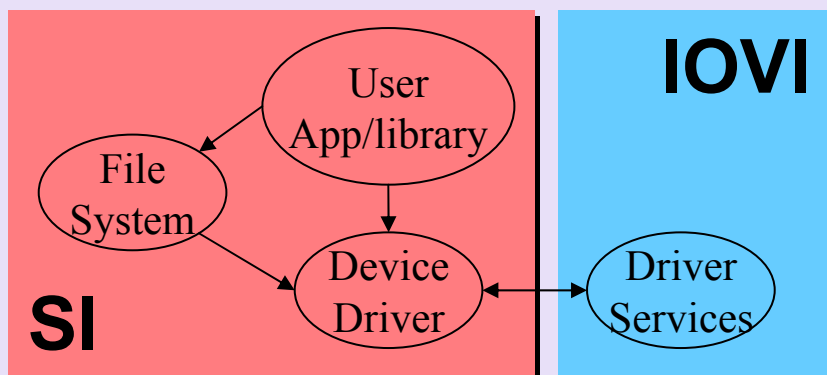
- ✓ Routes PCI interrupts to the System Image associated with the PCI BDF# that generated the interrupt.

## ■ Note: the PCI-SIG is not standardizing ATPT or Interrupt Tables.

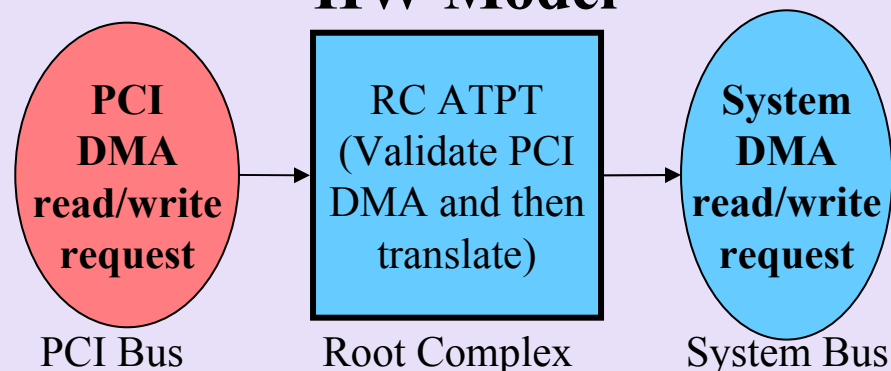
- ✓ They are covered here merely to provide context for the architectural responsibilities of SR-PCIM and SIs.

# ATPT based DMA

## SW Model



## HW Model



DMA Services performed by IOVI:

### ■ Initialization

- ✓ Pins host memory.
- ✓ Assigns DMA Addresses
- ✓ Programs the ATPT entries
- ✓ Returns a PCI memory address to DD.

### ■ Completion

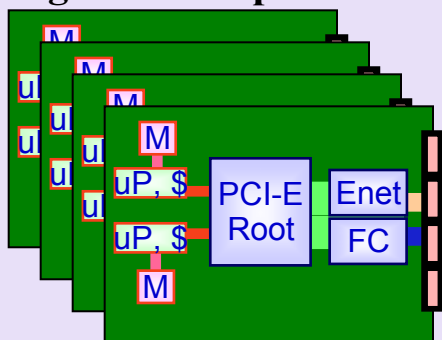
- ✓ Invalidates ATPT Entries
- ✓ Unpins host memory.
- ✓ Releases DMA Addresses.

### ■ Address Translation and Protection performed at or above the RC:

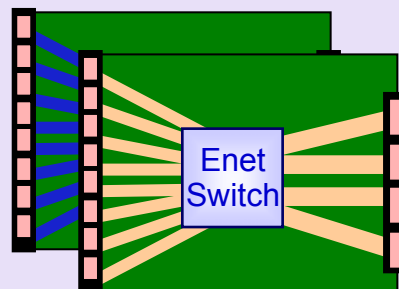
- ✓ Validates incoming PCI memory accesses, for example by limiting the memory addresses accessible by a PCIe BDF#.
- ✓ Converts incoming PCI memory address to the associated RC system bus memory address.

# Multi-Host Mechanisms

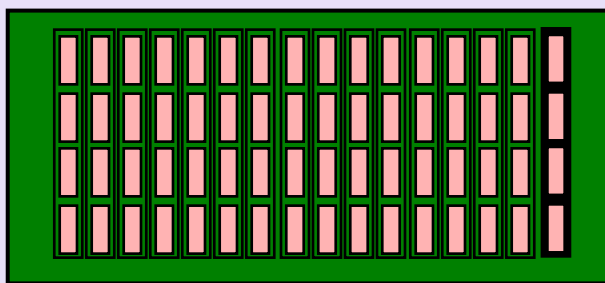
Server with  
Integrated Adapter Blades



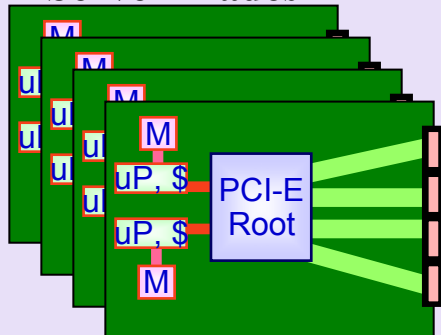
Switch blades



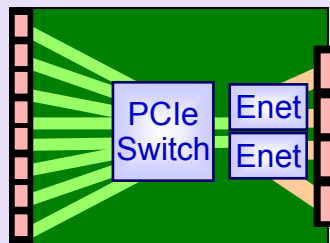
Midplane  
(N fabrics)



Server Blades



Adapter blades



- **Today:** PCI adapters used to access external networks are integrated into blades.

✓ Pros/Cons:

- + Uses well established network management infrastructures.
- Adapters cannot be shared.
- SAN/LANs link rates may not scale with CPU performance over time.

- **Multi-host PCI Topologies:** PCI fabric used to share adapters.

✓ Pros/Cons:

- + Adapters can be shared.
- + PCI family generations have higher link rate than SAN/LAN links.
- New network management infrastructure.

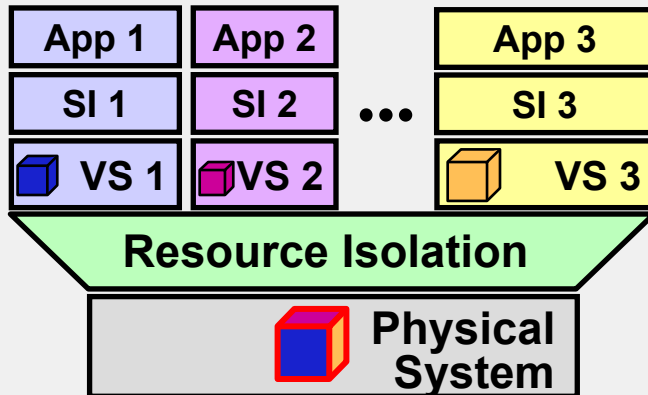


# Scope of Work Overview

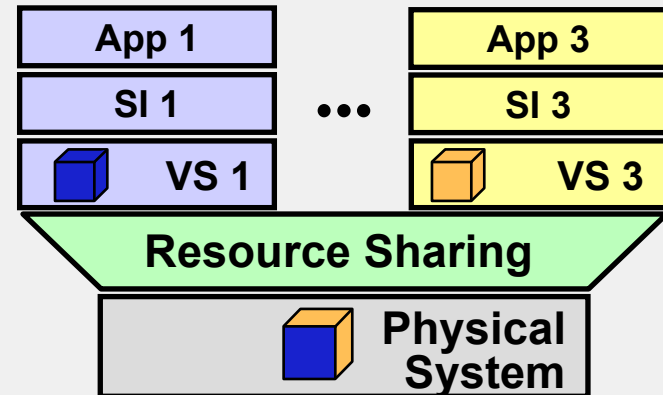




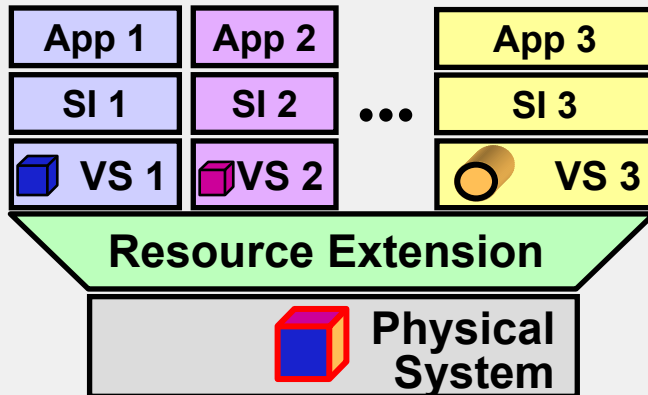
# Problem Statement



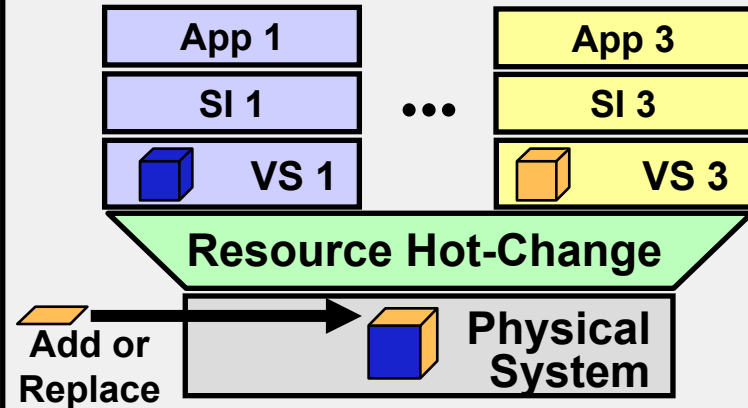
Isolation allows application co-location within a shared physical system.



Resource sharing allows lower TCO by utilizing resources more efficiently.



Extension of functions available to applications without physical system changes.

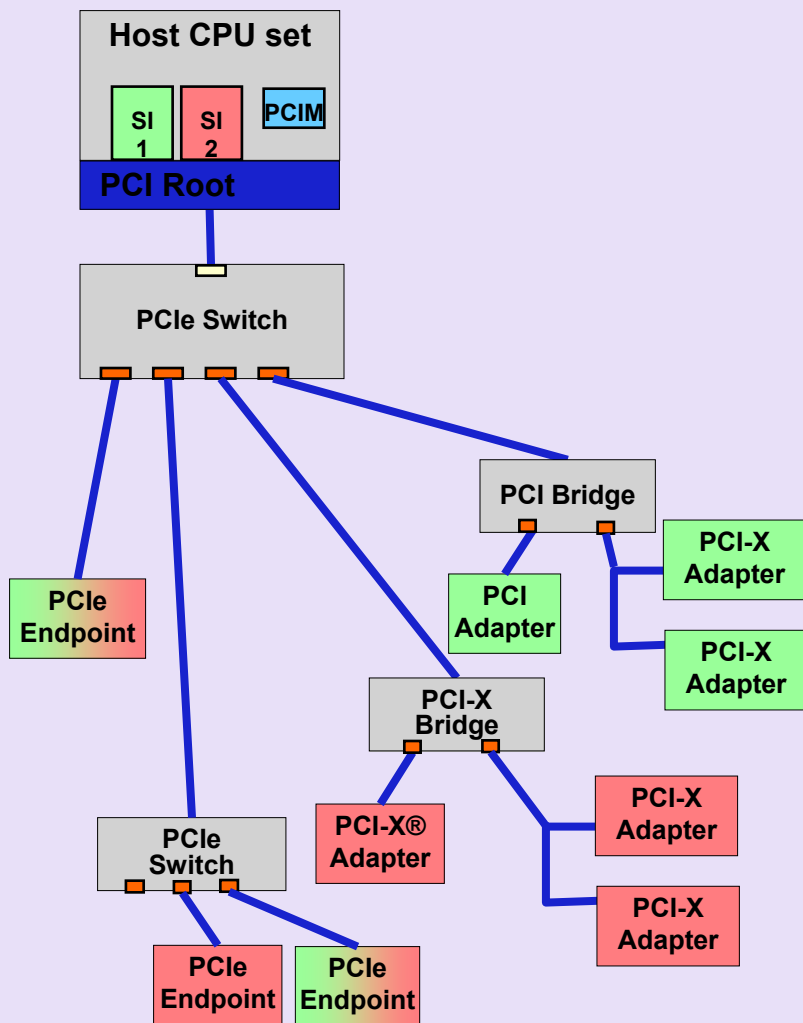


Resources can be transparently added or replaced beneath running applications.

# Problem Statement (Cont.)

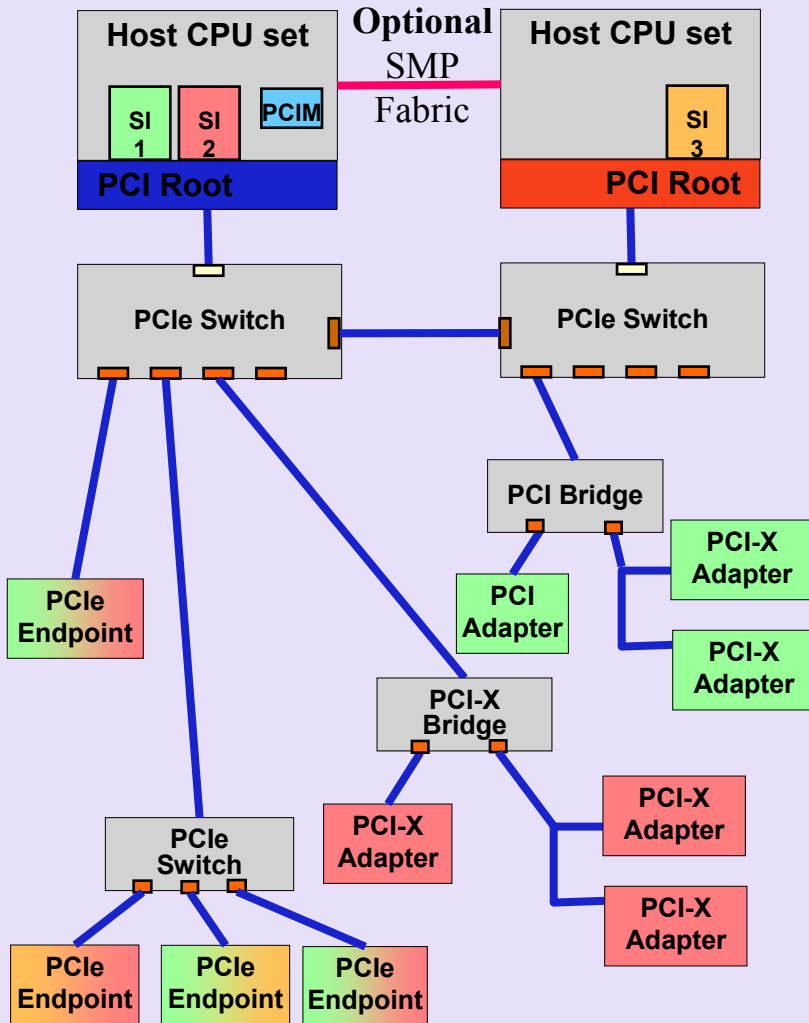
- Problem:
  - ✓ The industry has made significant strides in developing processor and memory virtualization technology.
  - ✓ Due to I/O's dependencies on industry standards, such as PCI, Memory Mapped I/O Virtualization has remained largely undefined or proprietary.
- Workgroup Consensus
  - ✓ I/O virtualization and I/O sharing specifications are needed for point-to-point and switch-based configurations.
  - ✓ These specifications will enable the industry to create interoperable components – chipsets, switches, endpoints, and bridges.
- I/O Sharing – Sharing of an I/O component by multiple SIs.  
Sharing types:
  - ✓ **Serial Sharing** – ability to transfer exclusive I/O component usage from one SI to another.
  - ✓ **Simultaneous Sharing** – ability to share an I/O component by multiple SIs in parallel.

# High-Level Requirements (cont.)



- The scope **will** enable an I/O Endpoint to be serially shared or simultaneously shared by multiple SIs (e.g. OS guests) in a single hierarchy domain (single RC).
  - ✓ **Single Root (SR) IOV Enabled Endpoint** – An Endpoint that supports the required IOV extensions defined in this specification.
- The scope **will** investigate whether new mechanisms are required to prevent, or selectively prevent, peer-to-peer operations within a single RC hierarchy.

# High-Level Requirements (cont.)



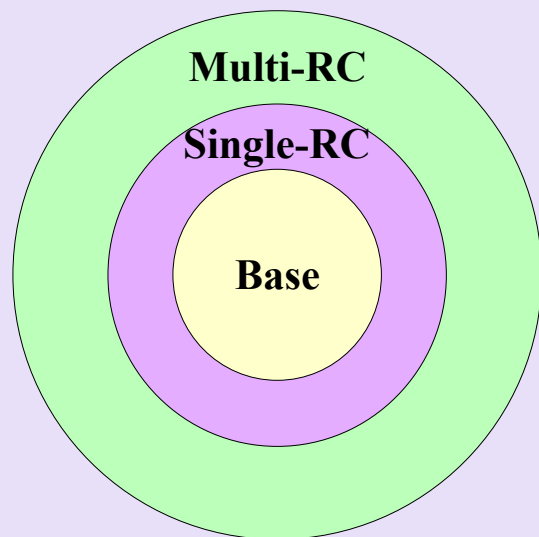
- The scope **will** enable an I/O Endpoint to be serially shared or simultaneously shared by multiple SIs (e.g. OS guests) in a single hierarchy domain (single RC) or in a multi-hierarchy domain (multiple RC connected to a common PCIe switch fabric).

  - ✓ **Multi-Root (MR) IOV Enabled Endpoint** – An Endpoint that supports the required IOV multi-root extensions defined in this specification.

# High-Level Requirements (cont.)

The scope **will** enable:

- Attachment of existing PCIe 1.x Base components – RC, Switches, Endpoints, and Bridges.
- A solution to use a combination of existing base and IOV-aware components:

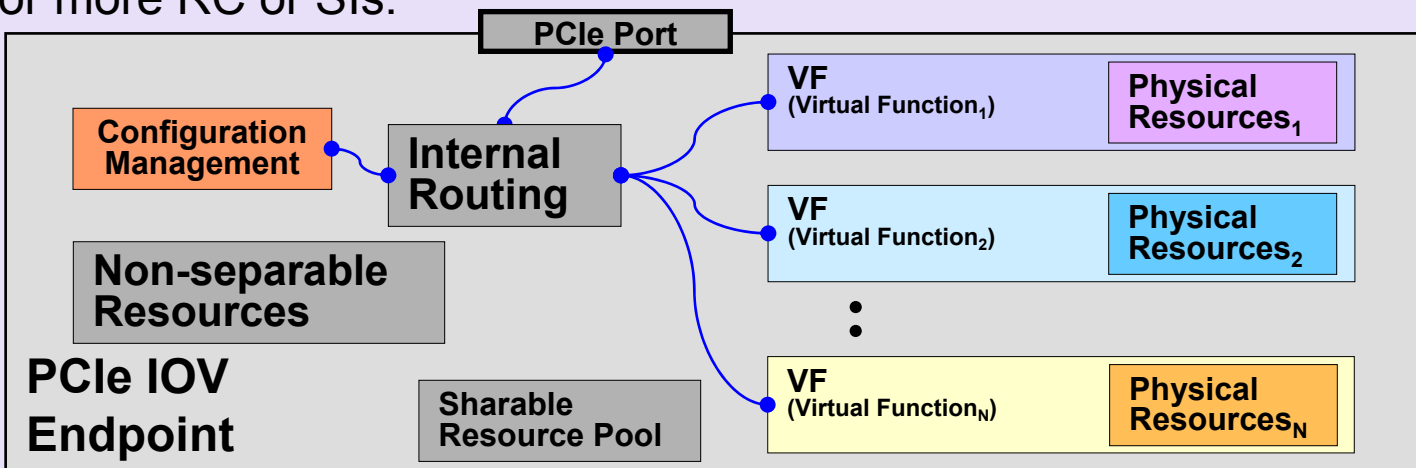


- ✓ Single-RC capabilities **shall** be a superset of the PCIe 1.x Base specification.
- ✓ Multi-RC capabilities **shall** be a superset of the single-RC capabilities.
- ✓ Objective is to maximize component interoperability across multiple topologies, e.g.:
  - An I/O component designed to support multi-RC **shall** support the single-RC capabilities.
- IOV-capable components to be backwards compatible with existing software.
  - ✓ Although some or all of the new IOV capabilities may not be supported in these circumstances.



# High-Level Requirements (cont.)

- The scope **will** include support for the creation, management, and destruction of resources associated with an IOV enabled Endpoint.
  - ✓ An IOV enabled Endpoint may be serially or simultaneously shared by one or more RC or SIs.

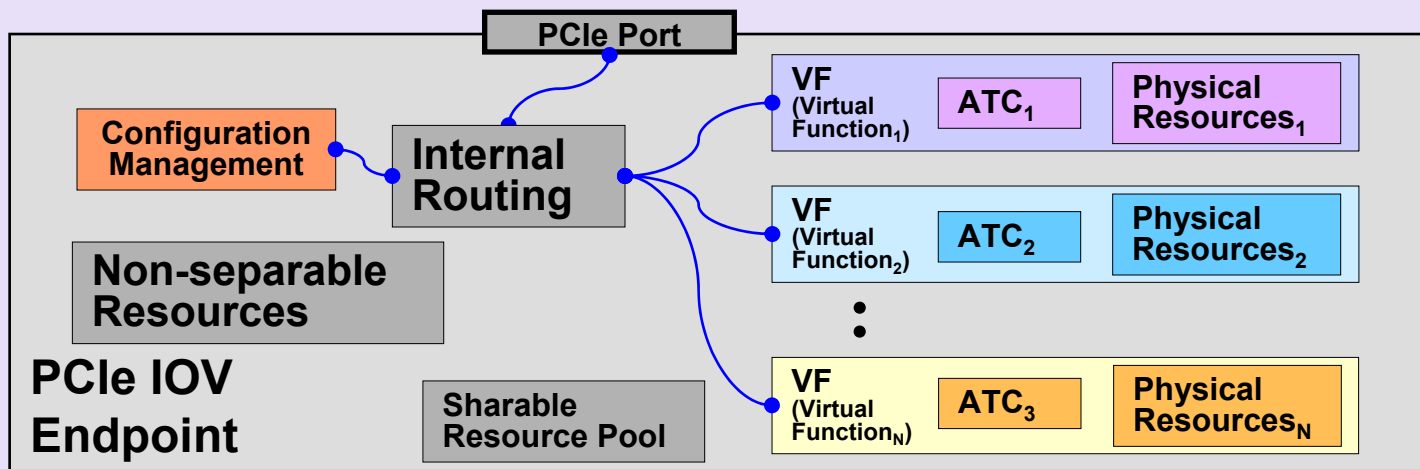


- **Virtual Function (VF)** - A Function, within an IOV enabled Endpoint, that shares one or more physical Endpoint resources, such as a link, with another and can, without run-time intervention by a VI, directly:
  - ✓ sink I/O and Memory operations from an SI; and
  - ✓ source DMA, completion and interrupt operations to an SI.

An IOVI is still required for configuration operations.

# High-Level Requirements (cont.)

- The scope **will** include support for an Endpoint to request translated addresses prior to injecting a request to a RC, e.g. a DMA Read or DMA Write operation.
- The minimum expected operations are:
  1. Request a translation,
  2. Return a translated address, and
  3. Invalidate a prior translation.



- **Address Translation Services (ATS)** – The caching of information used within an ATC (Address Translation Cache) by the RC to translate a PCI bus address in order to access host memory.

# Goals

The scope of this work **will** enable:

- The development of interoperable components – RC, switches, endpoints, and bridges – that provide I/O virtualization and sharing capabilities.
  - ✓ These components may be jointly deployed with existing PCIe 1.x based components to provide varying degrees of functionality .
- Multiple SIs to:
  - ✓ Be co-located onto a shared physical domain; and
  - ✓ Provide attestable trust and fault isolation, when co-located.
- Support for a variety of physical topologies enabling the technology to be deployed across multiple market segments and usage models.

# Non-Goals

Based on the votes taken to date, the following areas were deemed as not in scope:

- The scope of this work **does not, in principle**, touch the physical layer.
  - ✓ Subsequent investigation (e.g. Cross-linking requirement analysis) could result in a change to the physical layer, but our objective is to avoid any impact if at all possible.
- The scope of this work **will not include**:
  - ✓ Host address translation or protection mechanisms, i.e. an ATPT;
  - ✓ The ability of multiple SIs to share a device attached to a PCIe Bridge; and
  - ✓ Support for peer-to-peer communication (e.g., IPC) between different RC.
- The scope of this work **will not define** policy management or other types of management functions used to provide higher-level management of the components specified by this workgroup. That is:
  - ✓ The scope will be conceptually focused on PCIe operation types, such as: configuration operations, event notification operations, etc...



# **Scope of Work**

## **Single Root Requirements**

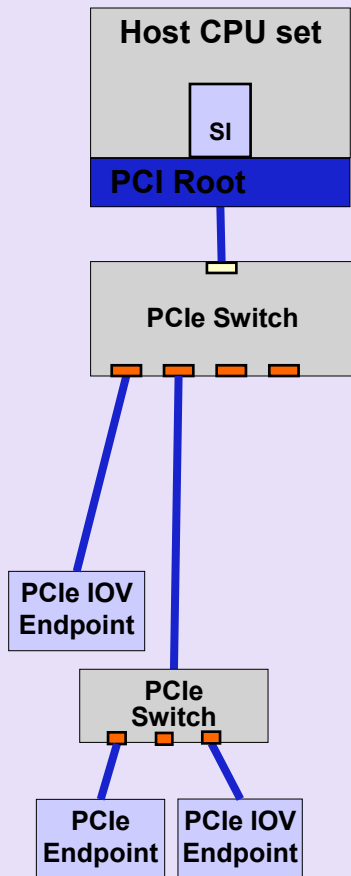




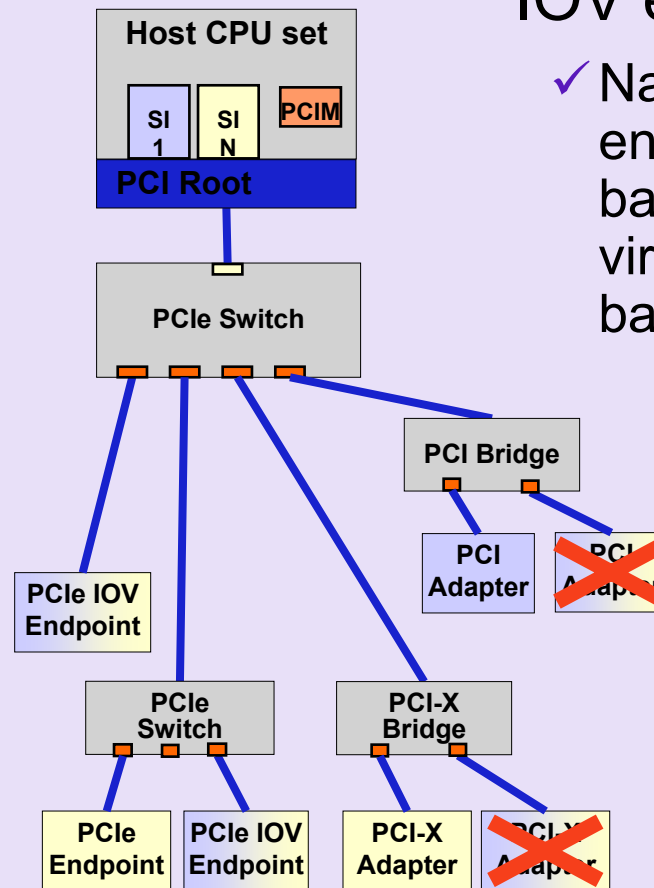
# Single RC PCIe IOV enabled Endpoint Requirements

- Only PCIe endpoints shall be specified for IOV enabled Endpoints.
  - Native based PCI SR OV enabled Endpoints shall be backwards compatible, in a non virtualized mode, with PCIe base 1.x SPEC.

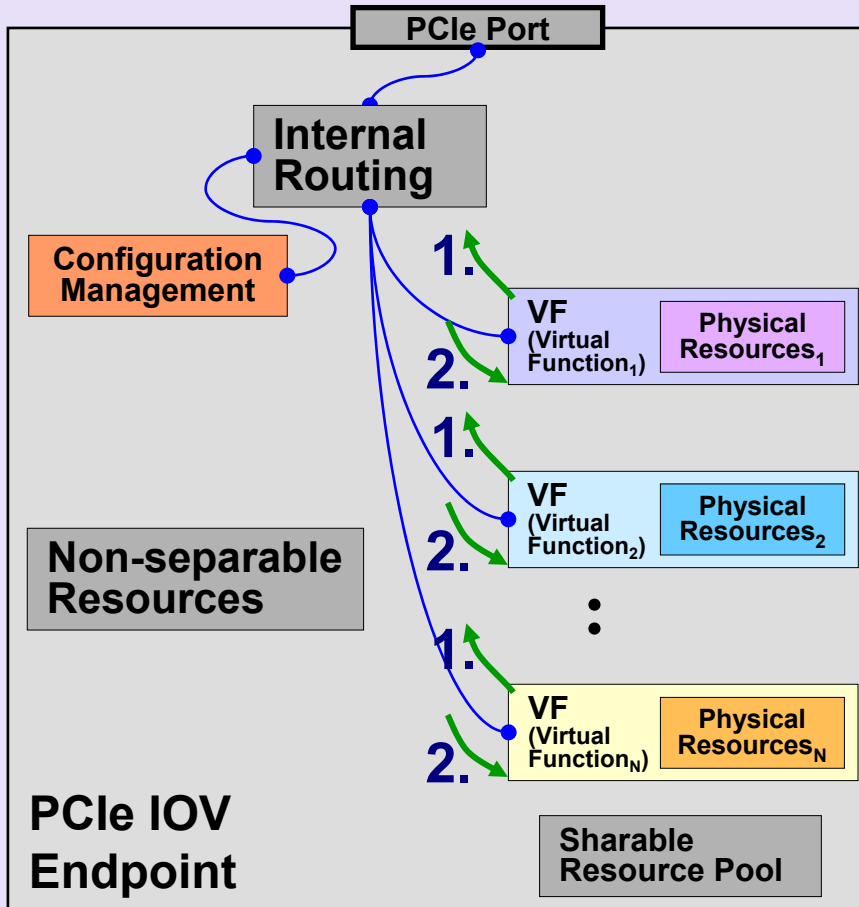
Base PCIe 1.x System



IOV Enabled PCIe System



# Single Root PCIe IOV Endpoint Reqs (cont.)

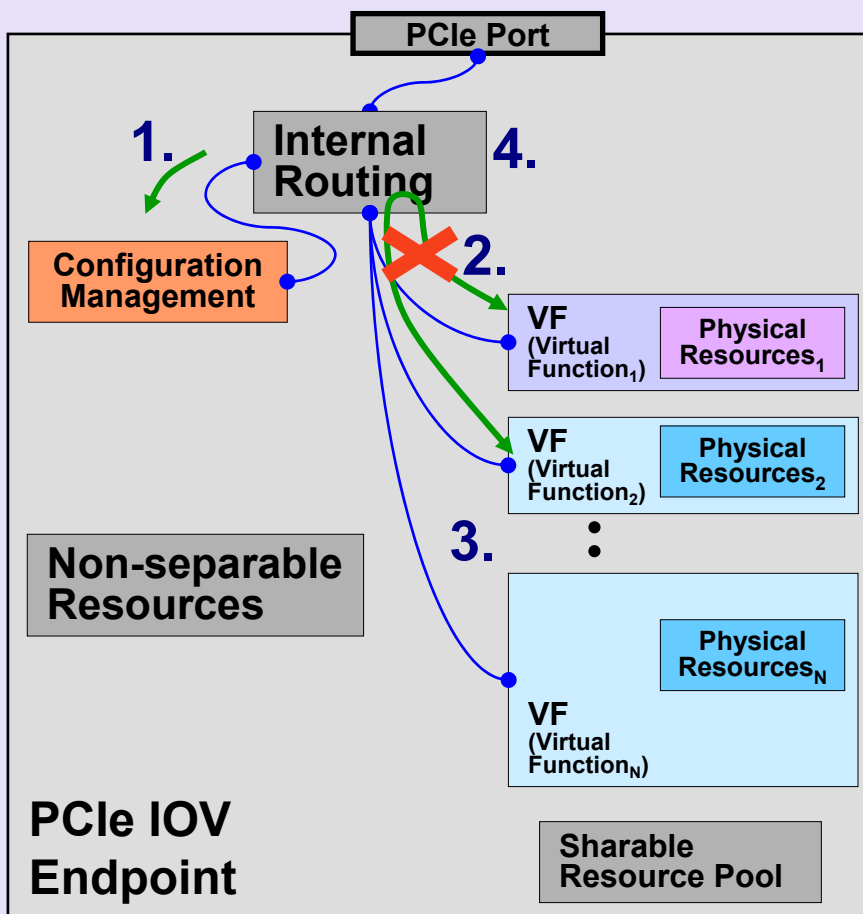


- Each VF shall have the capability to independently:
  1. Source DMA, completion, and interrupt operations within the system as an initiator;
  2. Sink configuration, I/O, and memory operations within the system as a target.
- Each VF shall appear in the fabric as a PCIe function.
- Except for rare errors conditions (spec will provide examples):
  - ✓ Each transaction issued from the PCIe IOV enabled Endpoint shall be uniquely identifiable with the VF that sourced the transaction.
  - ✓ The PCIe IOV enabled Endpoint shall be able to associate each transaction it sinks with the appropriate VF.

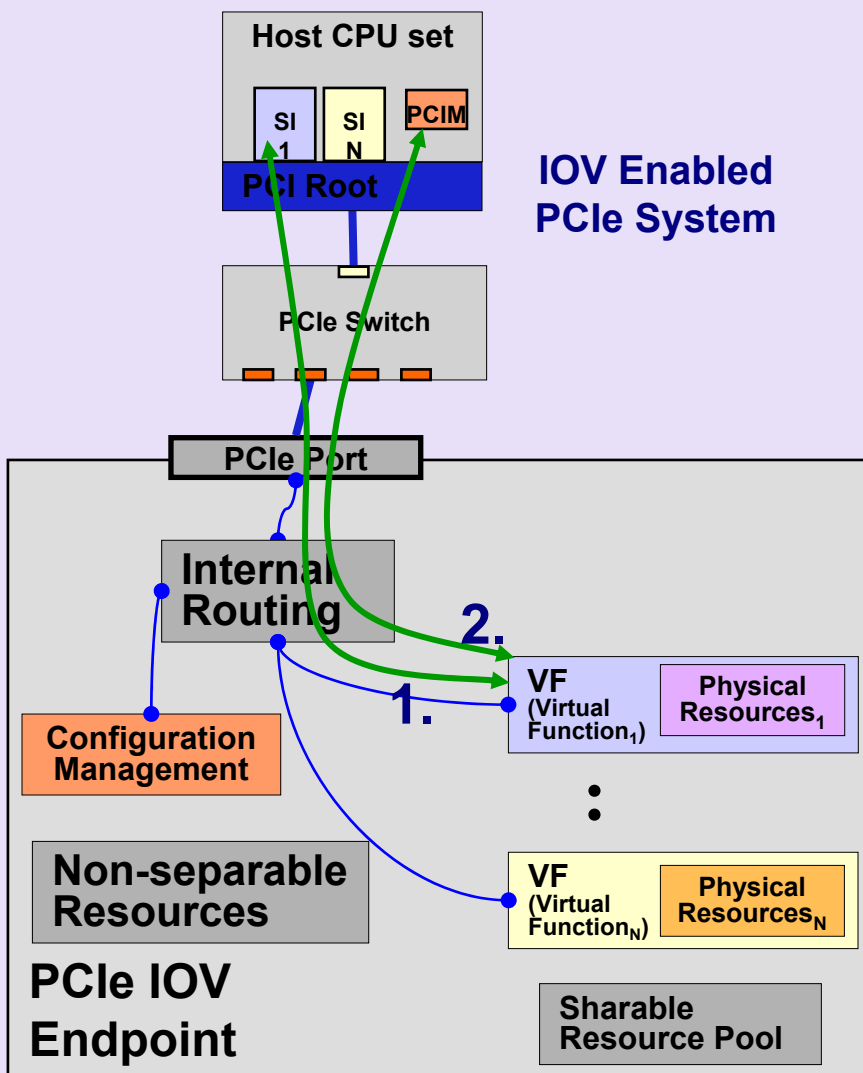
# Single Root PCIe IOV Endpoint Reqs (cont.)

- IOV enabled Endpoints shall:

1. Support a standard, in-band, mechanism for managing the IOV endpoint;
2. Include functionality to disable their peer to peer traffic to other functions within the IOV Endpoint.
3. Be allowed to support multiple functions assigned to the same SI with the same level of functionality given to a current multi-function device.
4. Support defined arbitration mechanisms between the VF's within an IOV enabled endpoint
5. Support the capability to have each VF independently managed, including:
  - ✓ Functional reset (i.e. full device function state),
  - ✓ Enable/Disable (a.k.a. Create/Destroy)
  - ✓ Having a VF's PCIe state and resources modified, for example:
    - ✓ TC assignment,
    - ✓ Bus Master Enable, ...
6. Support the specification of the minimum and maximum number of resources available to each VF.



# Single Root PCIe IOV Endpoint Reqs (cont.)



1. A mechanism shall be provided to allow a VF to be associated with an SI, such that data movement operations are enabled and can be performed directly between the SI and its associated VF, without VI involvement.
2. The virtualization mechanisms defined in this specification may require a VI (such as a PCI Configuration Manager) to be involved for configuration operations performed on a VF.



# **Scope of Work**

## **Multi-Root Requirements**

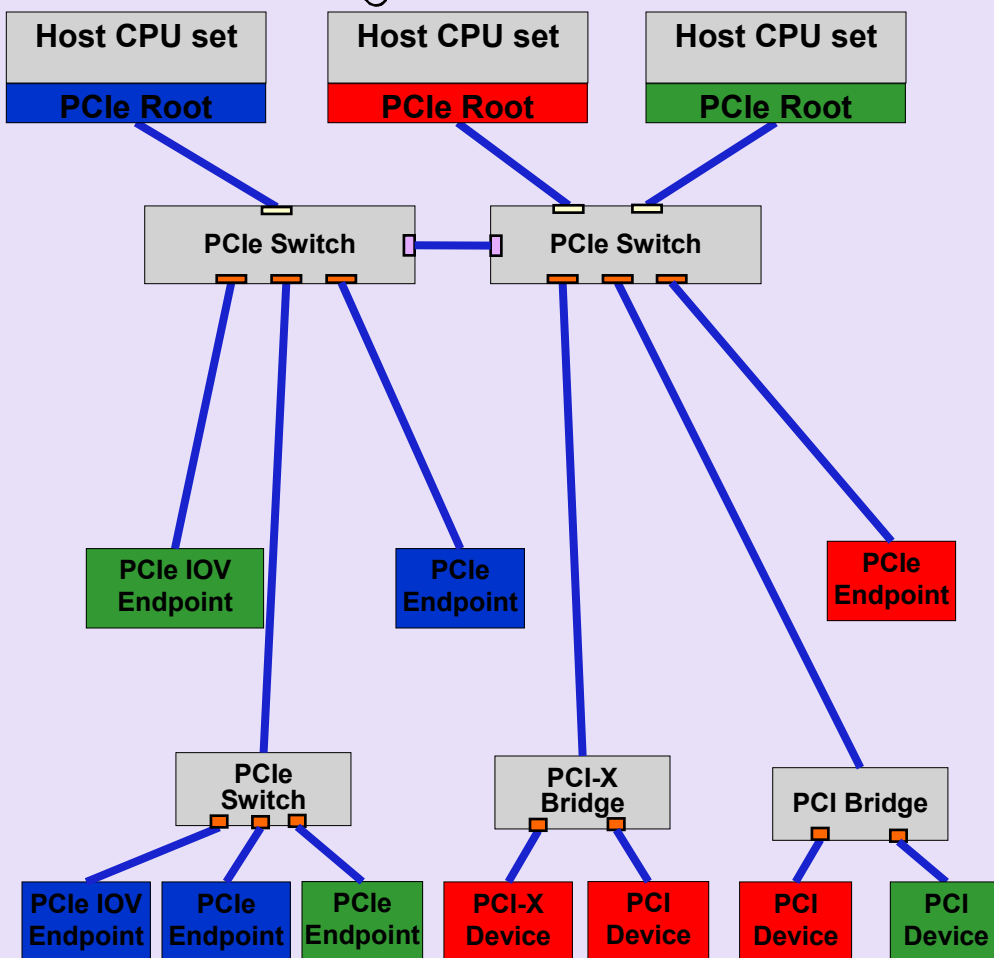




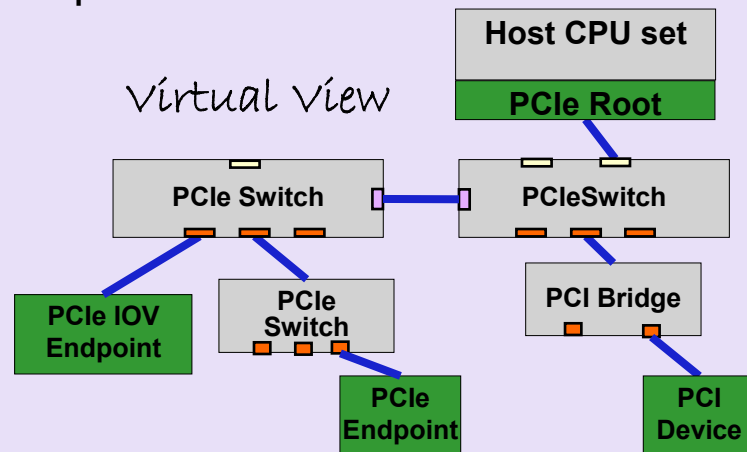
# Multi-Root PCIe IOV Endpoint Requirements

- The multi-root solution
  - ✓ Shall give each RC its own Virtual Hierarchy.
  - ✓ Should enable SW written for PCI, PCI-X, or PCIe enumeration, configuration, and management to run, unchanged on each RC within the multi-root system.
  - ✓ Shall enable each switch, bridge, function, and VF to be uniquely represented in the configuration space of each RC.

*Physical view*



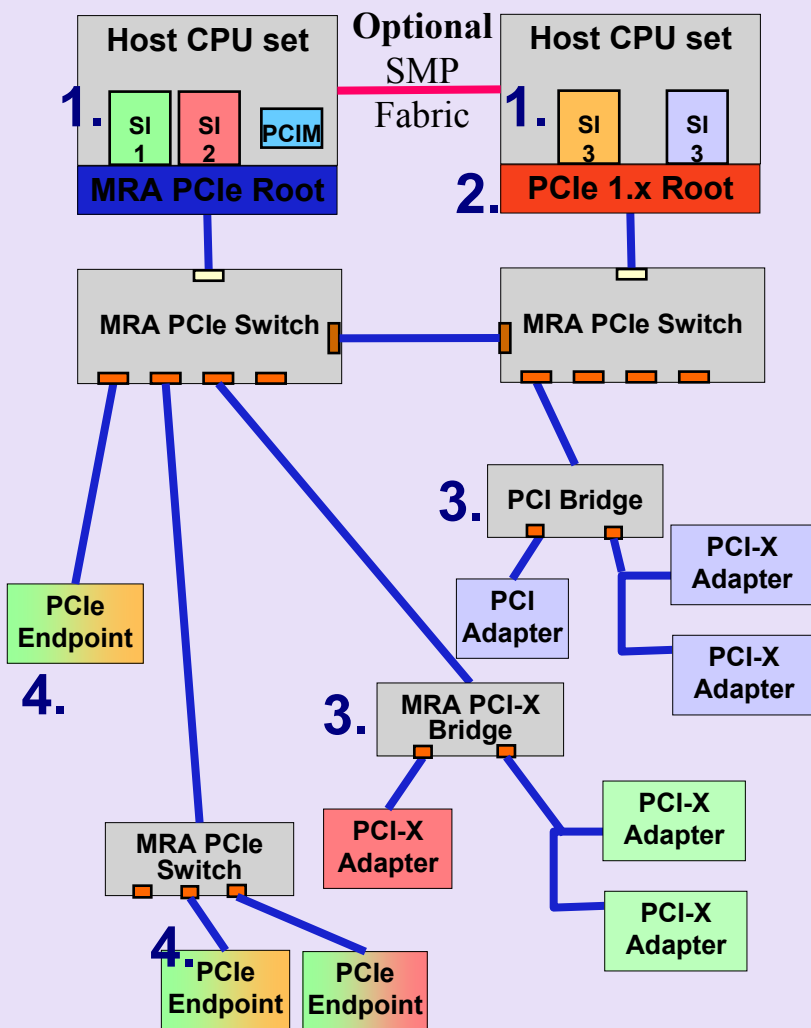
*Virtual view*



# Multi-Root PCIe IOV Endpoint Reqs (cont.)

- The multi-root solution shall:

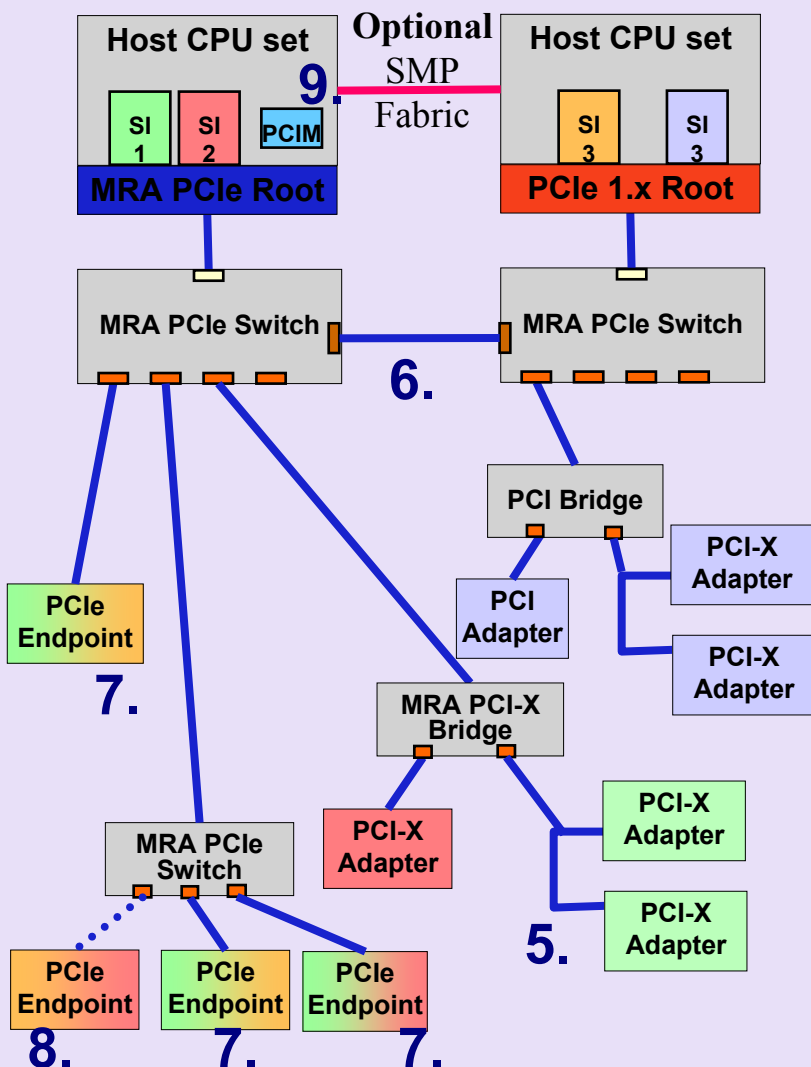
1. Provide the same characteristics to its IOV enabled Endpoints as the single-root solution relative to separate SIs.
2. Enable use of existing PCIe 1.x or later RC.
3. Enable existing PCIe 1.x Switches, Endpoints, and PCIe to PCI/PCI-X Bridges to each be bound to a single RC.
4. Enable an IOV enabled endpoint to be shared amongst multiple RC's using a **Multi-Root Aware (MRA)** PCIe switch.



# Multi-Root PCIe IOV Endpoint Reqs (cont.)

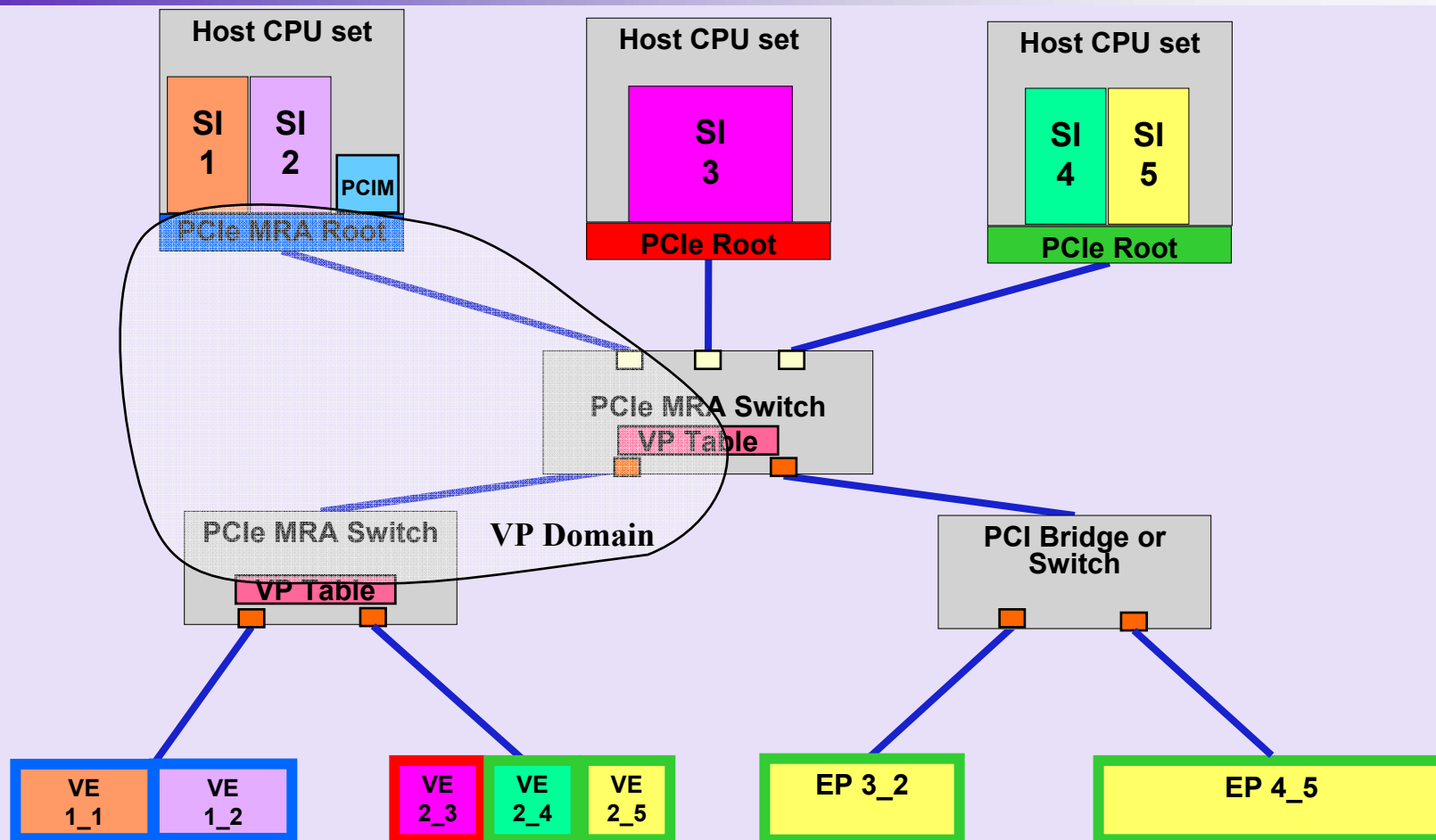
- The multi-root solution shall:

5. Enable a PCIe 1.x MRA bridge to PCI/PCI-X to assign each of its independent busses to a separate RC.
6. Enable multiple MRA switches to be connected together.
7. Enable an IOV enabled Endpoint to be shared by multiple SIs within one or more RC's.
8. Enable hot-plug of physical EPs and VFs.
9. Enable a management entity (PCI Configuration Manager, PCIM) to manage the fabric.



# Virtual Plane based MR

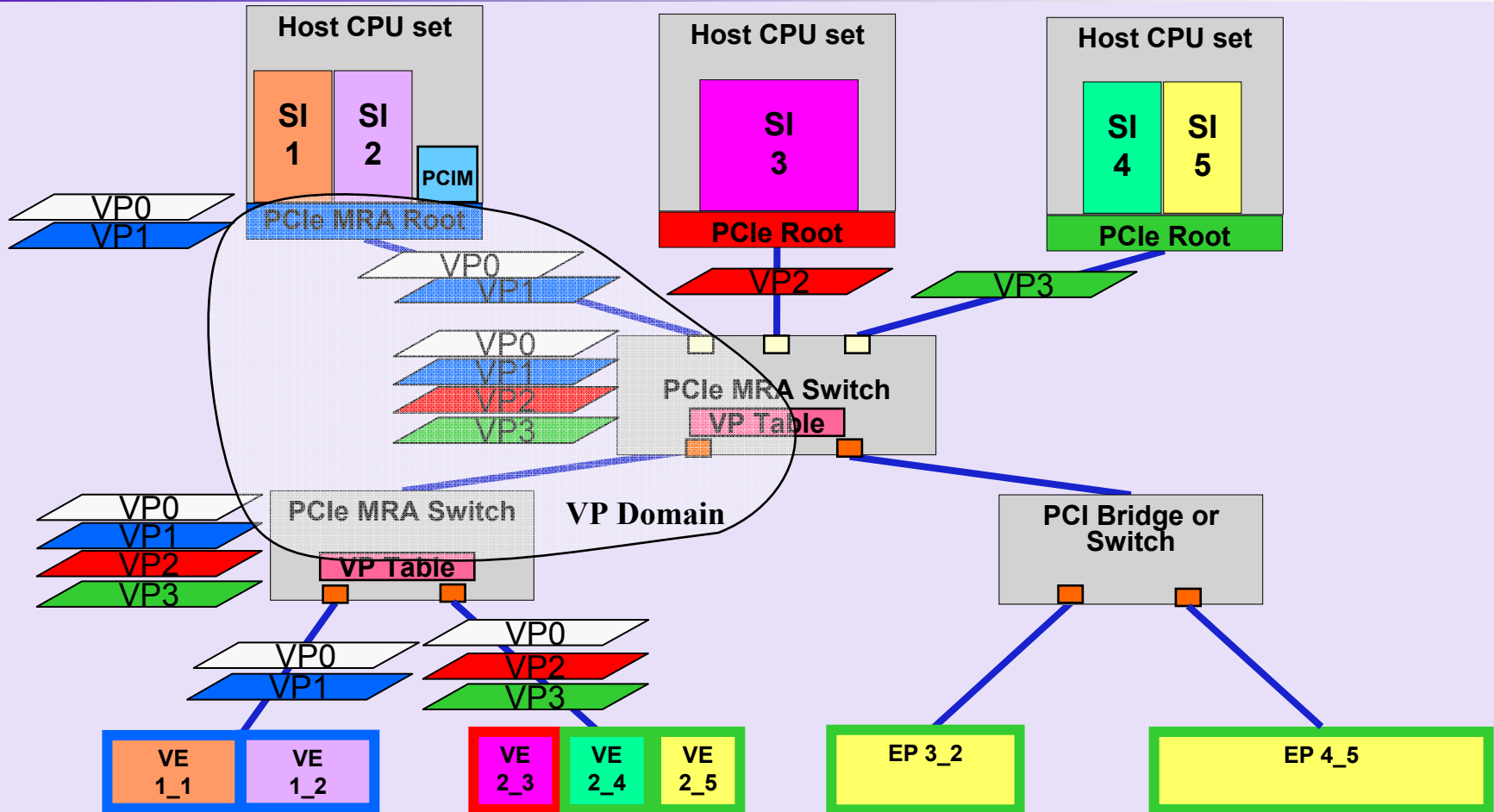
## Showing mix of legacy and MRA Components



Legend:

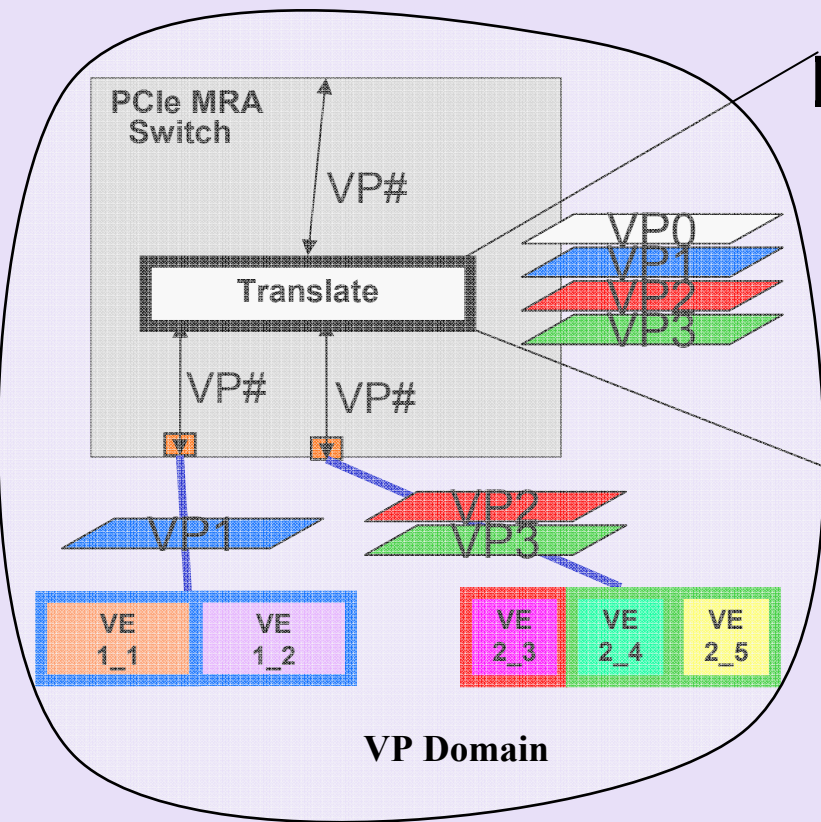
- VE N\_M = Virtual EP in Physical EP N and associated with Image M
- For Adapter – Fill Color shows SI association; Line Color shows root complex association.
- VP = Virtual Plane as defined by a new VP field which is associated with an RC.

# Virtual Plane based MR

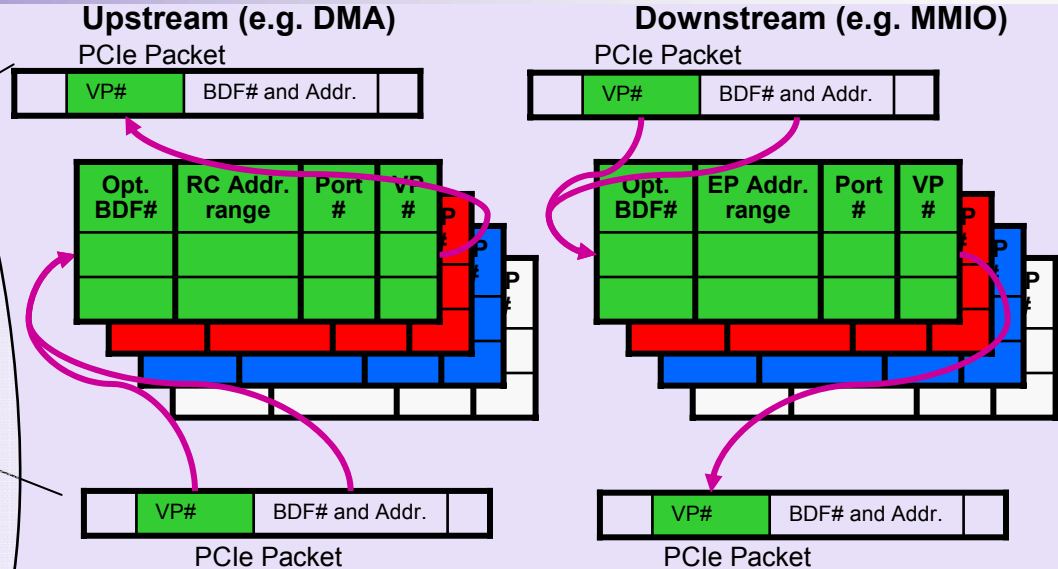


Note: Each VP has its own upstream and downstream memory addresses (more on the next pages).  
 Each legacy RC is assigned its own VP.  
 If in-band management is used, then VP is dedicated for PCIM on an MRA RC.  
 Otherwise out-of-band management can be used.

# Multi-Root Aware Switch and EP Support



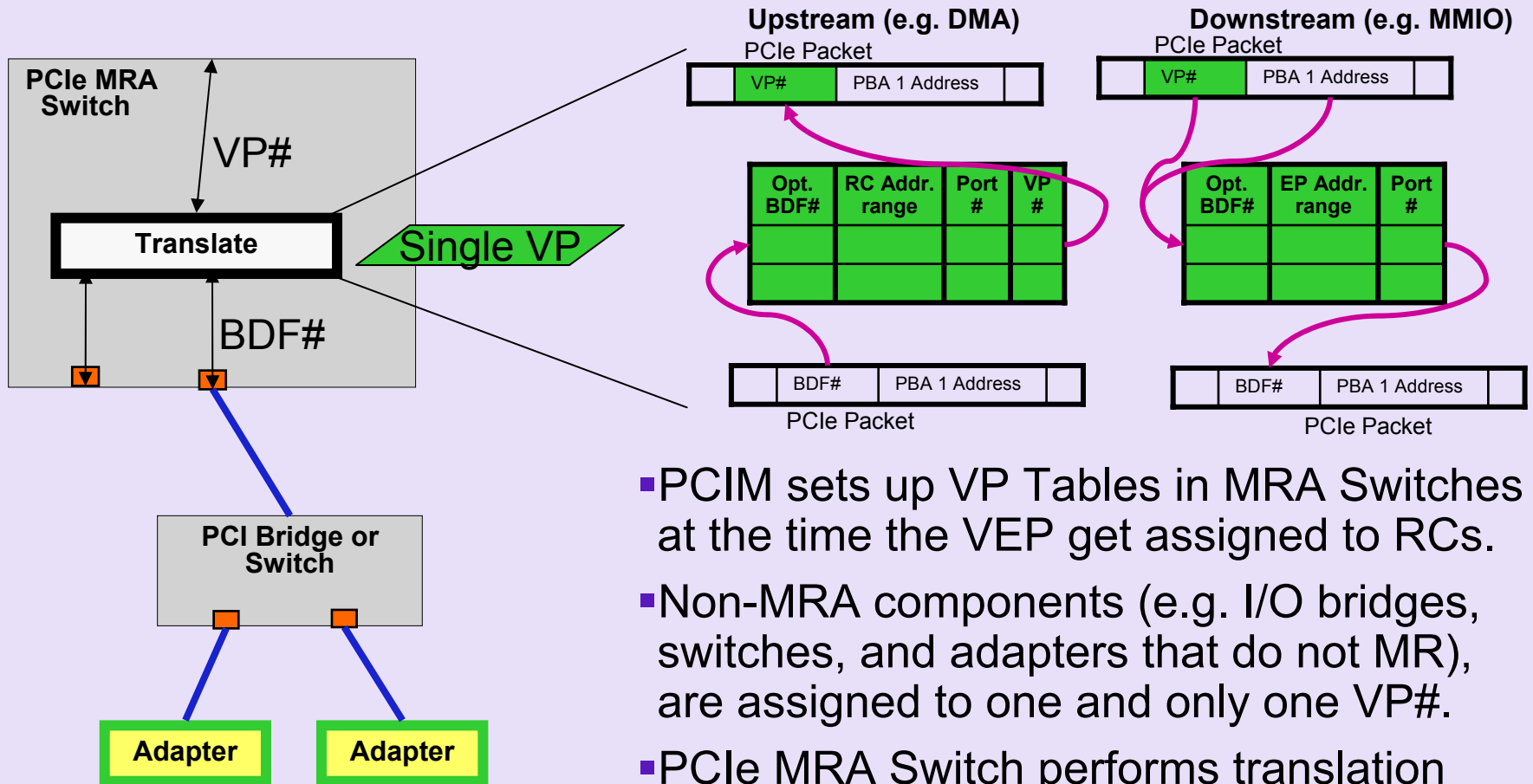
**Legend:** VP = Virtual Plane as defined by a new VP field which is associated with an RC.



- PCIM sets up VP Tables in MRA Switches at the time the VE get assigned to RCs.
- For MRA switches the VP table is used to determine outbound port.
- Current PCI IOV WG direction is that the VP# is at DLLP (i.e. not part of TLP).



# Legacy Switch and EP Support



**Legend:** VP = Virtual (address) Plane as defined by a new VP field which is associated with each BDF#.

- PCIM sets up VP Tables in MRA Switches at the time the VEP get assigned to RCs.
  - Non-MRA components (e.g. I/O bridges, switches, and adapters that do not MR), are assigned to one and only one VP#.
  - PCIe MRA Switch performs translation when entering MRA domain.
- ✓ Inserts VP# associated with component.

# PCIM Requirements

The PCIM shall be able to:

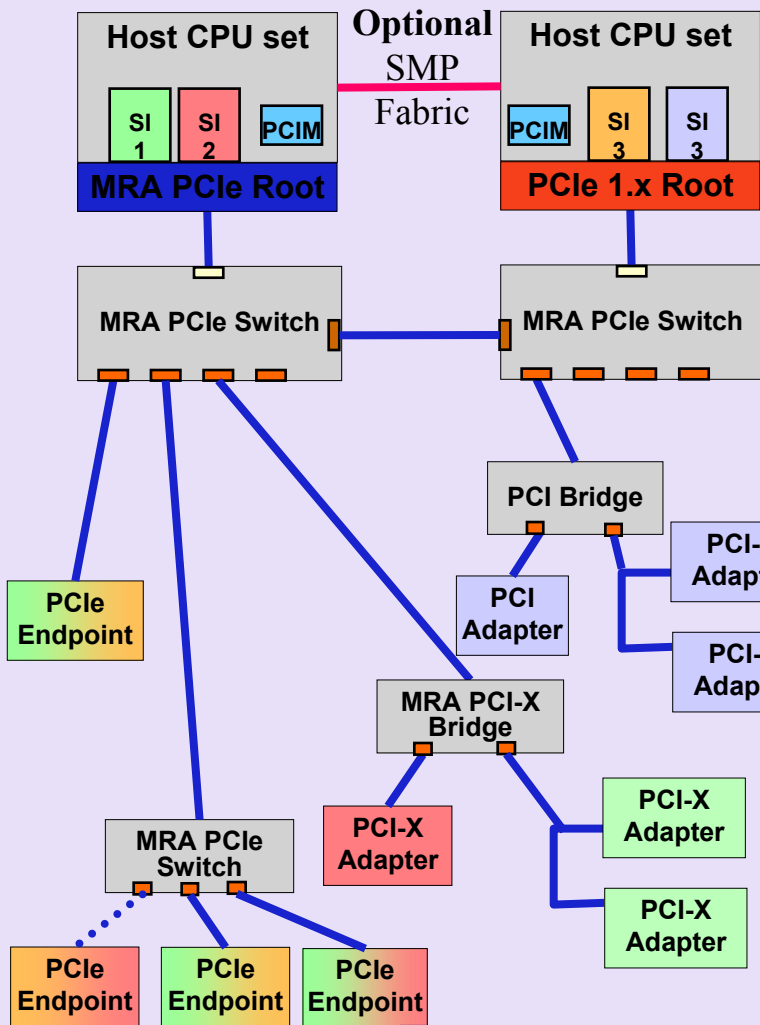
- ✓ Discover and configure IOV enabled RC's, switches, bridges, and endpoints using a defined PCIM interface.
- ✓ Discover and configure non-IOV components using existing interfaces.
- ✓ Control and manage errors within the fabric.

The solution supports one or more PCIM entities (with coordination between them).

A method will be defined:

- ✓ To establish a trust relationship between a PCIM and a given PCIe component.
- ✓ For the PCIM to control and limit access to IOV management functions within PCIe IOV enabled Endpoints by entities other than the primary PCIM.

The PCI Express® Multi-Root Configuration provides multiple configuration spaces for each switch, bridge and endpoint.





## **Scope of Work**

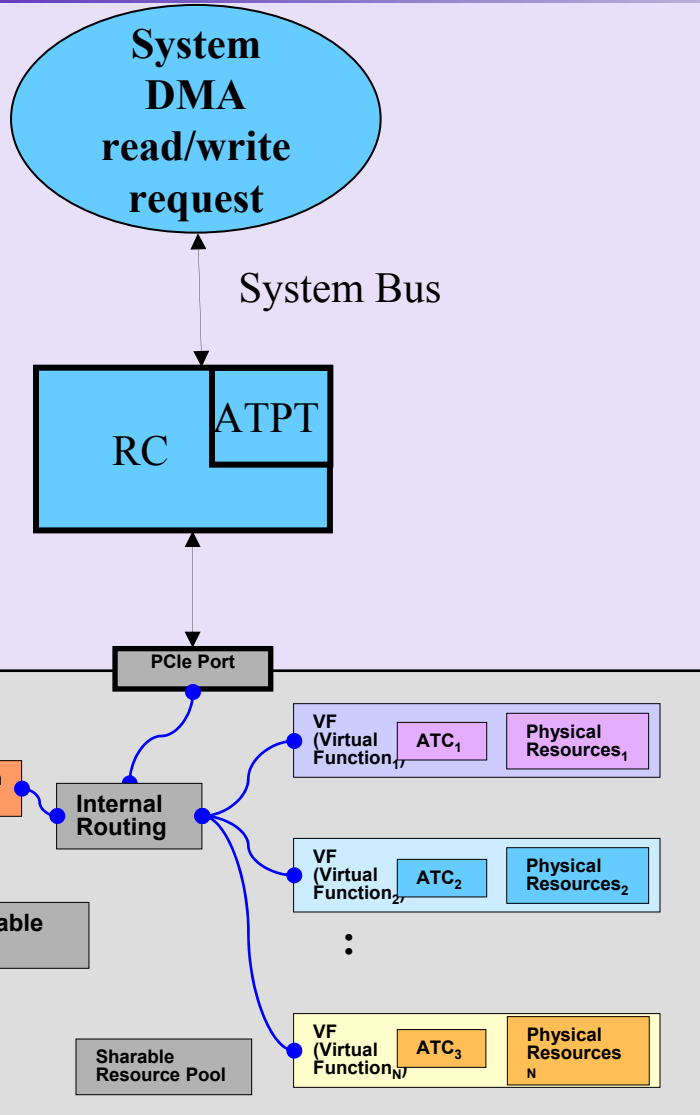
# **Address Translation Services Requirements**



# ATS Requirements

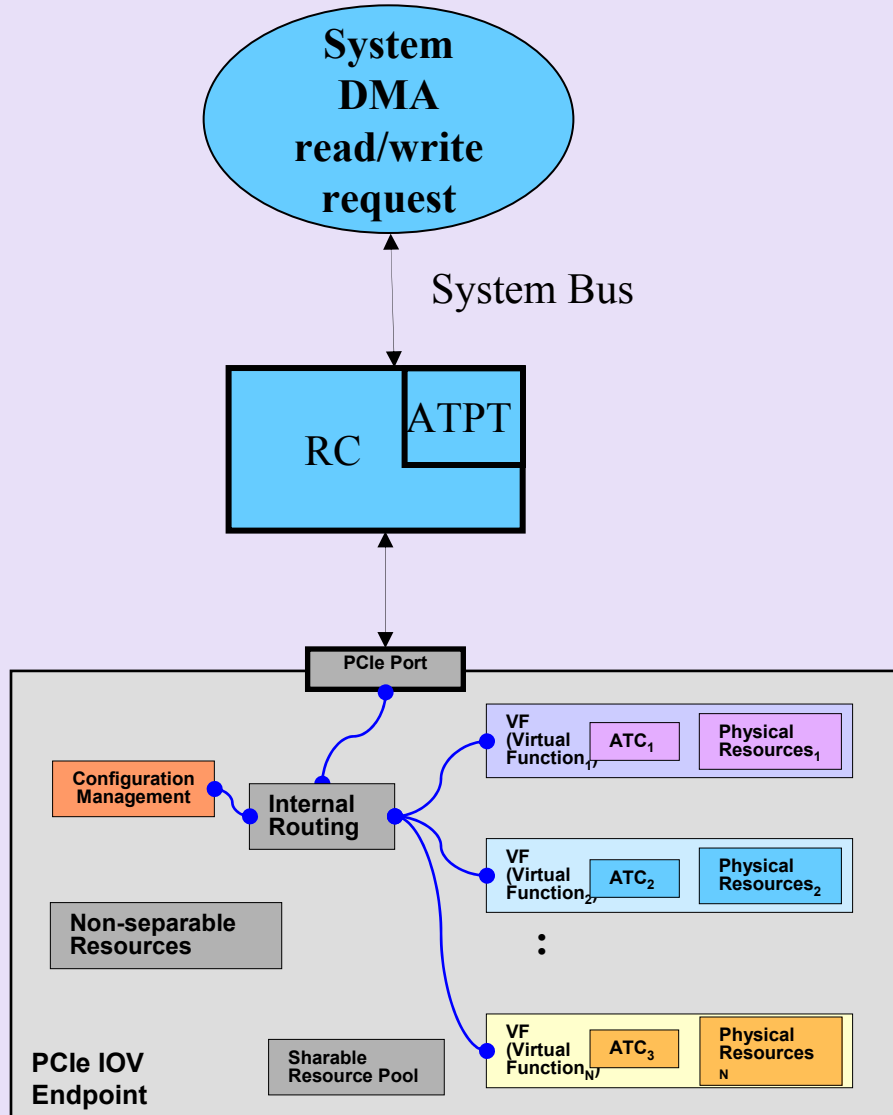
- Address Translation Services (ATS) will consist of a set of transactions to enable an Endpoint to:
  - ✓ Request a translation of a given address.
  - ✓ Indicate that a subsequent transaction, e.g. a DMA Read or DMA Write, has been translated.
  - ✓ Invalidation response to indicate that a translation invalidation request has been processed.
- ATS will consist of a set of transactions to enable a host Address Translation and Protection Table (ATPT) via an RC to:
  - ✓ Issue a translation completion of a given address.
  - ✓ Issue a translation invalidation request for a given address.
  - ✓ Ascertain whether an invalidation has been processed by an Endpoint.

# ATS Request



- **ATS Request**
  - ✓ Used by the Endpoint to request translated address
    - Address can be 32-bit or 64-bit
  - ✓ Requests can flow on any TC
  - ✓ Multiple ATS Requests may be pipelined
- **PCIe Endpoint Responsibilities**
  - ✓ Endpoint implements an ATC
    - ATC is a cache of translated addresses
    - ATC may be implemented independent of IOV support
  - ✓ Endpoint must only populate cache via ATS protocol
  - ✓ Endpoint must not allow ATC to be modified or accessed by software (local or remote) to prevent data leakage
  - ✓ For each ATS Request, there will be a corresponding ATS Completion.

# ATS Completion

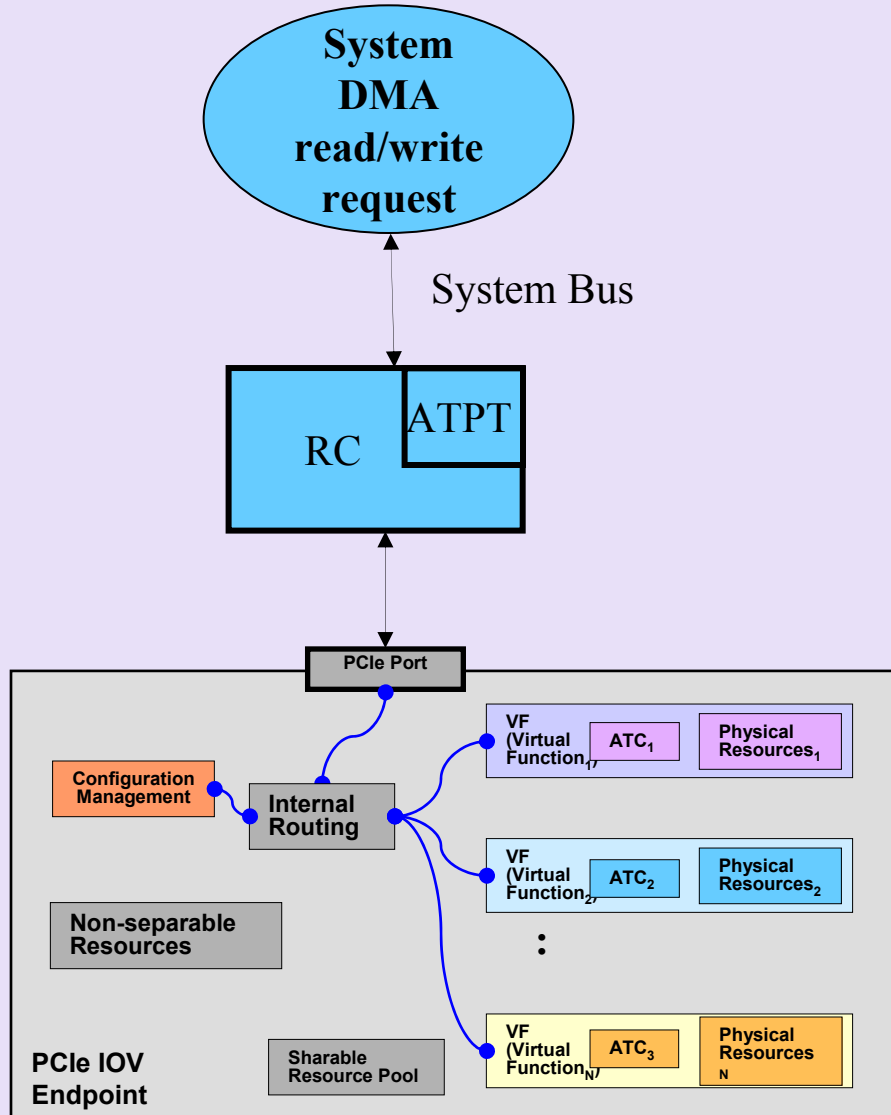


## ■ ATS Completion

- ✓ Used to return either:
  - A translated address which may cover a range of pages
  - A translation failure
- ✓ An ATS Completion must be generated for each ATS Request
  - Multiple ATS Completions may be in-flight
  - ATS Completions may be returned in any order relative to the ATS Requests
- ✓ An ATS Completion must use the same TC as the associated ATS Request
- ✓ Completions must be sent using the same TC as the ATS Request
- ✓ ATS Completions return access rights
  - Read-only, Write-only, Read / Write
    - Read = Write = 0 indicates there is a hole in the translation space
  - An Endpoint can only issue subsequent TLP Requests match the access rights on the associated address range

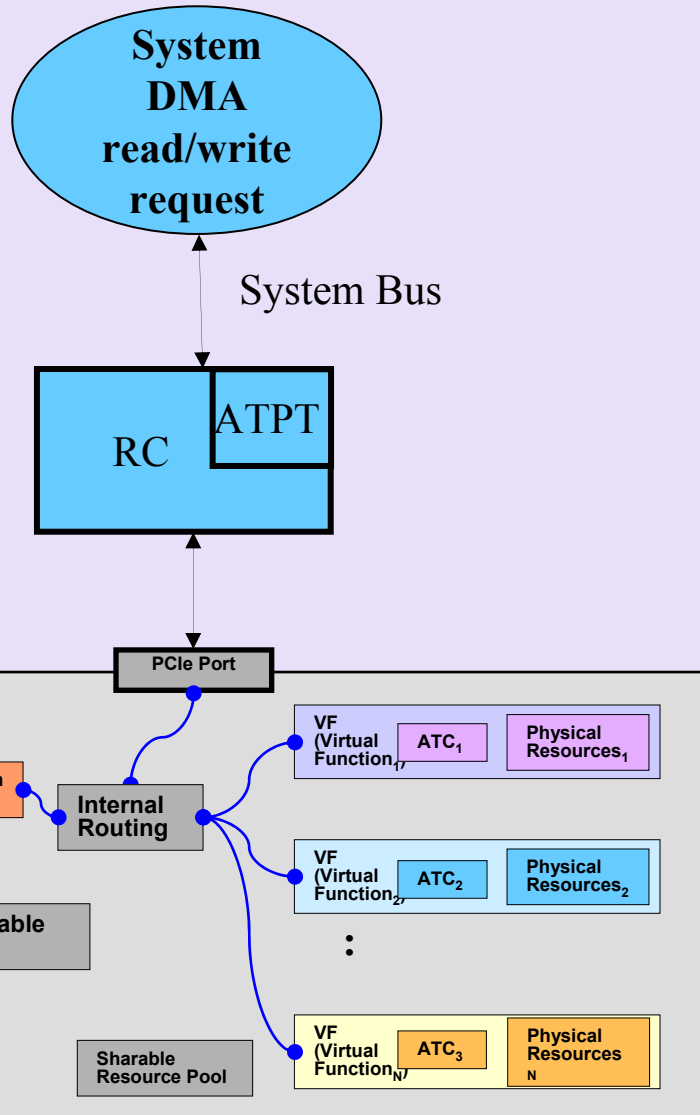


# ATS Completion (cont.)



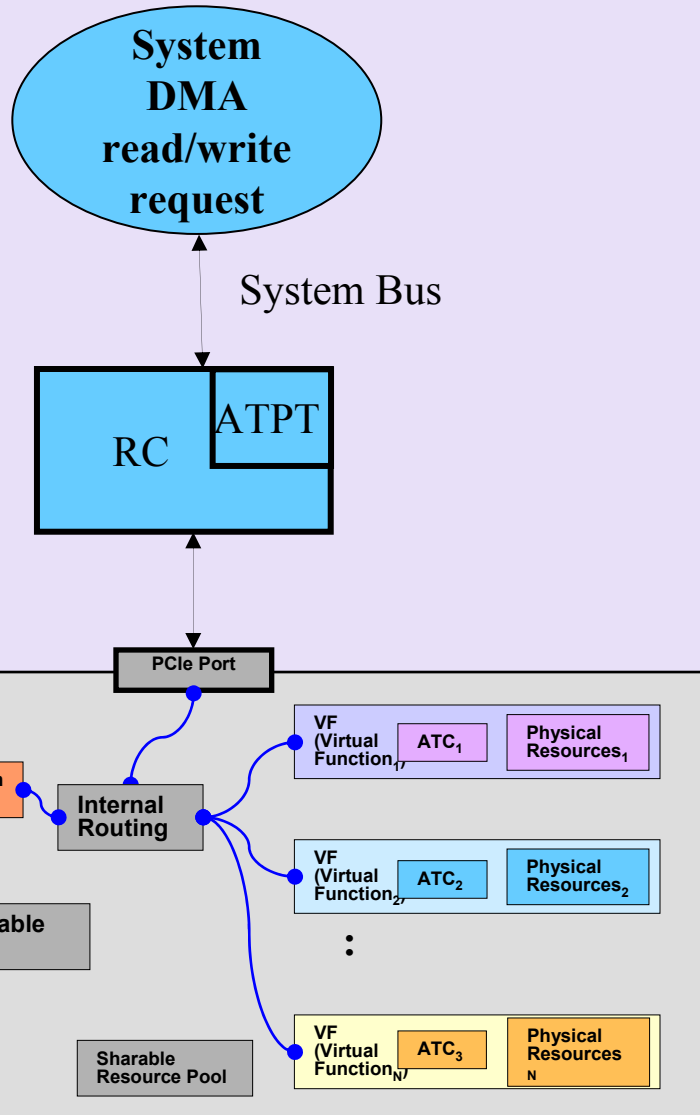
- An Endpoint responsibilities:
  - ✓ Be capable of sinking all ATS Completions without causing backpressure on the PCIe Link
  - ✓ Be able to discard ATS Completions that are stale
    - If an ATS Invalidation is received prior to the ATS Completion, then the subsequent Completion is stale and should not be used

# ATS Invalidation



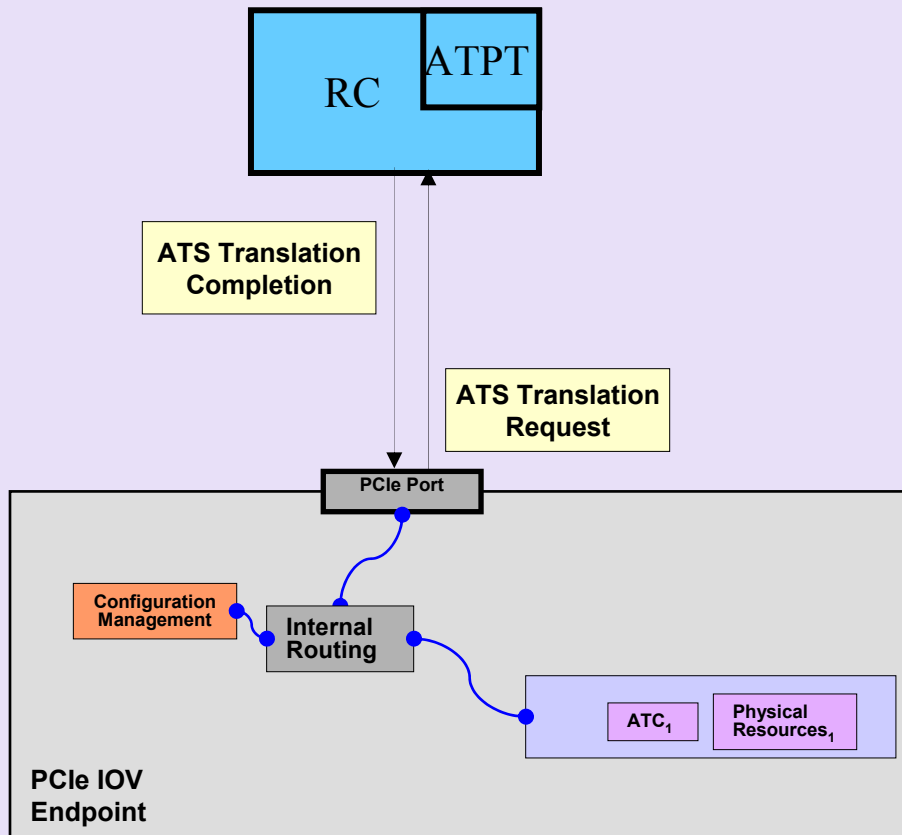
- **ATS Invalidation Request**
  - ✓ Used to maintain consistency between the ATPT Translation Agent (TA) and the ATC
  - ✓ ATS Invalidation Requests invalidate translations within an ATC
    - A Request may cover a range of address
  - ✓ ATS Invalidation Request may be issued at any time
    - When a translation is changed within the ATPT, the TA issues an ATS Invalidation Request to the impacted ATC
  - ✓ ATS Invalidation Requests may be issued on any TC
    - The TA does not track the TC of prior ATS Request(s)
  - ✓ Each ATS Invalidation Request has an ITAG which uniquely identifies the Request
    - The associated Invalidation Completion returns this ITAG to enable the TA to quickly associate the Completion with the prior Request

# ATS Invalidation (cont.)



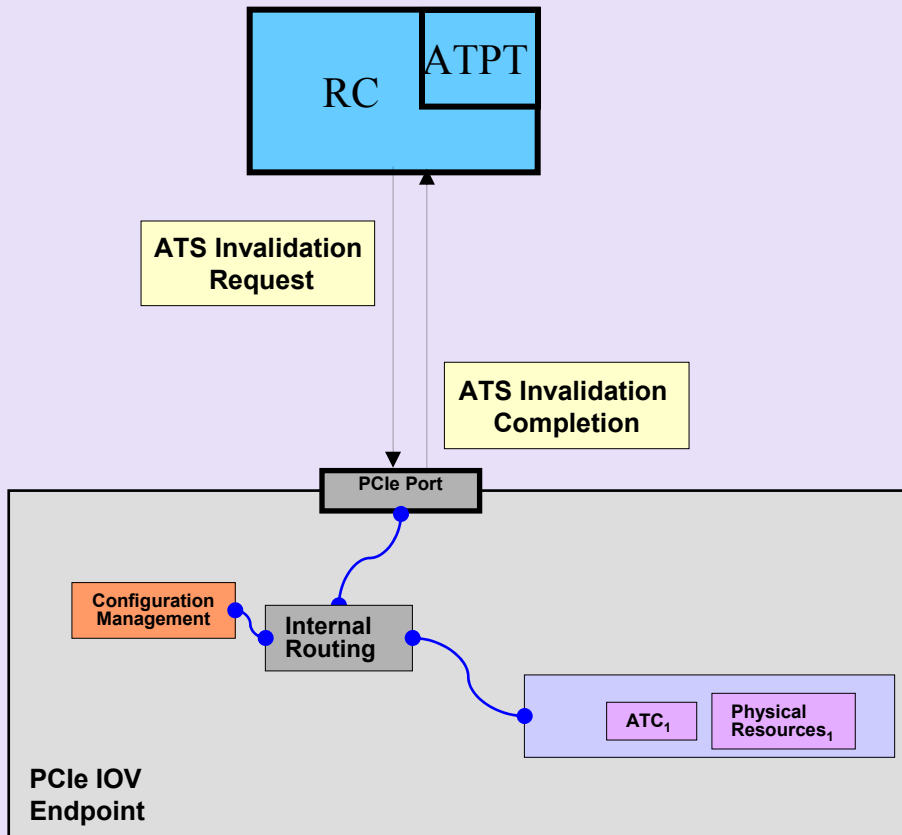
- Endpoint responsibilities:
  - ✓ Must return an ATS Invalidation Completion for each Request
  - ✓ Must not indicate an invalidation has completed until all outstanding Read Requests that reference the associated translated address have been retired.
  - ✓ Must insure that the invalidation completion indication to the RC will arrive at the RC after any previously posted writes that use the 'stale' address.
  - ✓ Is NOT required to immediately flush all pending requests upon receipt of an Invalidate Request. If transactions are in a queue waiting to be sent, it is not necessary for the device to expunge requests from the queue even if those transaction use an address that is being invalidated.

# Example ATS Request Flow



1. Endpoint determines address ranges that will benefit from ATS
  1. For example, an Endpoint may want to translate addresses associated with work queues while not for single-use scatter-gather addresses
2. Endpoint issues an ATS Request
3. Translation Agent (TA) examines ATPT to determine if a translation exists
4. If translation exists, issues an ATS Completion with associated translation and access rights
  1. A TA may selectively ignore Requests for a given Function even if a translation exists
5. If translation does not exist, issues an ATS Completion indicating no such translation
  1. Endpoint may still issue DMA TLP but must not set the "Translation bit"
6. Endpoint issues DMA TLP to access associated address range

# Example ATS Invalidation Flow



- The ATPT requires a translation update
- TA issues ATS Invalidation Request
- Endpoint may flush or complete all pending transactions associated with the effected address range
- Once the Endpoint has cleaned up all state associated with the effected address range, it returns an ATS Invalidation Completion
- TA processes ATS Invalidation Completion and updates ATPT accordingly



# Alternative RID Interpretation ECR





# Problem Statement

- PCIe Base specification constraints impeding innovation
  - ✓ New process technology is enabling developers to integrate an increased number of I/O functions within a single Endpoint
    - PCI-SIG member companies prefer to avoid incorporating one or more virtual switches to provide increased fan-out as noted in the specification.
      - Do not want to incur virtual switch cost / complexity
      - Do not want to consume additional Bus Numbers
  - ✓ Virtualization requires each SI to see a unique I/O identifier
    - Each Virtual Endpoint must be assigned a unique RID
      - Multiple Endpoints may be assigned per SI
      - Multiple Virtual Endpoints may be assigned per SI
    - Given advent of multi-core processors, sub-CPU sharing, etc. the number of SI per RC will increase making the current PCIe constraints untenable in the long-term

# Requester ID (RID) Review

- A Requester ID is a 16-bit field composed of three elements:
  - ✓ Bus Number – 8 bits in length
  - ✓ Device Number – 5 bits in length
  - ✓ Function Number – 3 bits in length
- Functions must capture the Bus and Device Numbers supplied with all Configuration Write Requests (Type 0) completed by the function and supply these numbers in the Bus and Device Number fields of the Requester ID for all Requests initiated by the device/function.
  - ✓ The Bus Number and Device Number may be changed at run time, and so it is necessary to re-capture this information with each and every Type 0 Configuration Write Request.
- Each function associated with a logical device must be designed to respond to a unique Function Number for Configuration Requests addressing that logical device.
  - ✓ Each logical device may contain up to eight logical functions.

# PCIe Constraints

- PCIe Base specification v1.1 states:
  - ✓ All PCI Express components are restricted to implementing a single Device Number on their primary interface (Upstream Port), but may implement up to eight independent functions within that Device Number.
  - ✓ Switches and Root Complexes must associate only Device Number 0 with the Endpoint attached to the logical bus representing the Link from a Switch Downstream Port or a Root Port.
  - ✓ Configuration Requests specifying all other Device Numbers (1-31) must be terminated by the Switch Downstream Port or the Root Port with an Unsupported Request Completion Status (equivalent to Master Abort in PCI).

# Problem Statement

- PCIe Base specification constraints impeding innovation
  - ✓ New process technology is enabling developers to integrate an increased number of I/O functions within a single Endpoint
    - PCI-SIG member companies prefer to avoid incorporating one or more virtual switches to provide increased fan-out as noted in the specification.
      - Do not want to incur virtual switch cost / complexity
      - Do not want to consume additional Bus Numbers
  - ✓ Virtualization requires each SI to see a unique I/O identifier
    - Each Virtual Endpoint must be assigned a unique RID
      - Multiple Endpoints may be assigned per SI
      - Multiple Virtual Endpoints may be assigned per SI
    - Given advent of multi-core processors, sub-CPU sharing, etc. the number of SI per RC will increase making the current PCIe constraints non-tenable in the long-term

# Goals

- No change to the PCIe TLP wire protocol
  - ✓ Requester ID remains a 16-bit field
- Minimally, bound number of enumeration operations
  - ✓ Avoid issuing enumeration operations to potentially large RID space per Endpoint
- Optimally, define a deterministic set of Functions
  - ✓ Enumeration discovers all Functions through defined method
- Enable IHV to optimize resources if Functions are not used

# High-level Approach

- A Requester ID is a 16-bit field composed of two elements:
  - ✓ Bus Number – 8 bits in length
  - ✓ Function Number – 8 bits in length
- Function 0 is required
  - ✓ This is the equivalent of Device Number = 0, Function Number = 0
    - Required to enable variety of PCIe configuration operations to continue to operate as they do today
- An Endpoint can be assigned multiple Bus Numbers
  - ✓ Additional Bus Number bits are concatenated with the Function Number bits to identify a Function
    - Sparse Function Number space is allowed
    - Number of Functions per Bus Number is variable
      - Not required to be uniform across all Bus Numbers
  - ✓ Endpoint would be required to respond to Type 1 Configuration transactions on the additional Bus Numbers



# Questions



Thank you for attending.

For more information please go to  
[www.pcisig.com](http://www.pcisig.com)

# Workgroup Members

- AMD
- ATI
- Broadcom
- Emulex
- HP
- IBM
- IDT
- Intel
- LSI Logic
- Microsoft
- Next I/O
- Neterion
- Nvidia
- PLX
- Qlogic
- Stargen
- Sun
- VMWare



# **I/O Virtualization and Sharing**

**Michael Krause (HP, co-chair)**  
**Renato Recio (IBM, co-chair)**

