IOV 1.1 Update and Overview

Richard Solomon
LSI Corporation
Member I/O Virtualization Workgroup

What's New in SR-IOV 1.1

SR-IOV 1.1 – Don't Panic!

- Most changes are editorial
  - ✓ Additions to definitions
  - ✓ Clarified and consistent PF *M*, VF *M,N* numbering usage
  - ✓ Rewording for clarity
  - ✓ Terminology consistency
- Updates to reflect PCI Express Base 2.1
  - ✓ Atomic Operations
  - ✓ Address Translation Services (ATS) Page Request Interface
  - ✓ Dynamic Power Allocation
  - ✓ Multicast
  - ✓ TLP Processing Hints
- Additional information about Access Control Services and usage
  - ✓ New implementation note
- Additional information about ARI-capable hierarchy
  - ✓ One new bit – SR-IOV control register

SR-IOV 1.1: Editorial/Terminology

- Clarified and consistent Function numbering usage
  - ✓ **PF *M*** designates the Physical Function at Function number *M*
  - ✓ **VF *M,N*** designates the *N*th Virtual Function associated with PF *M*
  - ✓ VFs are numbered starting with 1 so the first VF associated with PF *M* is VF *M*,1
  - ✓ The Routing ID for each VF is determined using the Routing ID of its associated PF and fields in that PF's SR-IOV Capability

- Specification references updated
  - ✓ PCI Express Base Specification, Revision 2.1
    - Formerly referenced Revision 1.1 and Revision 2.0
  - ✓ PCI Local Bus Specification
    - Formerly referenced Revision 3.0, now generic
  - ✓ Address Translation Services, Revision 1.1
    - Formerly referenced Revision 1.0

SR-IOV 1.1: PCIe 2.1 & ACS Updates

- Register definitions updated to include Atomic Operations controls
  - ✓ Section 3.5.9 Device Capabilities 2 Register
  - ✓ Section 3.5.10 Device Control 2 Register
- Section 3.7 updated to include new Extended Capabilities
  - ✓ Section 3.7.6 Multicast
  - ✓ Section 3.7.7 Address Translation Services (ATS) Page Request Interface
  - ✓ Section 3.7.8 Dynamic Power Allocation
  - ✓ Section 3.7.9 TLP Processing Hints
- Section 3.7.2 updated with more information on using Access Control Services (ACS)
  - ✓ New implementation note
  - ✓ Remember that a “VI” is a “Virtualization Intermediary” aka “Hypervisor”

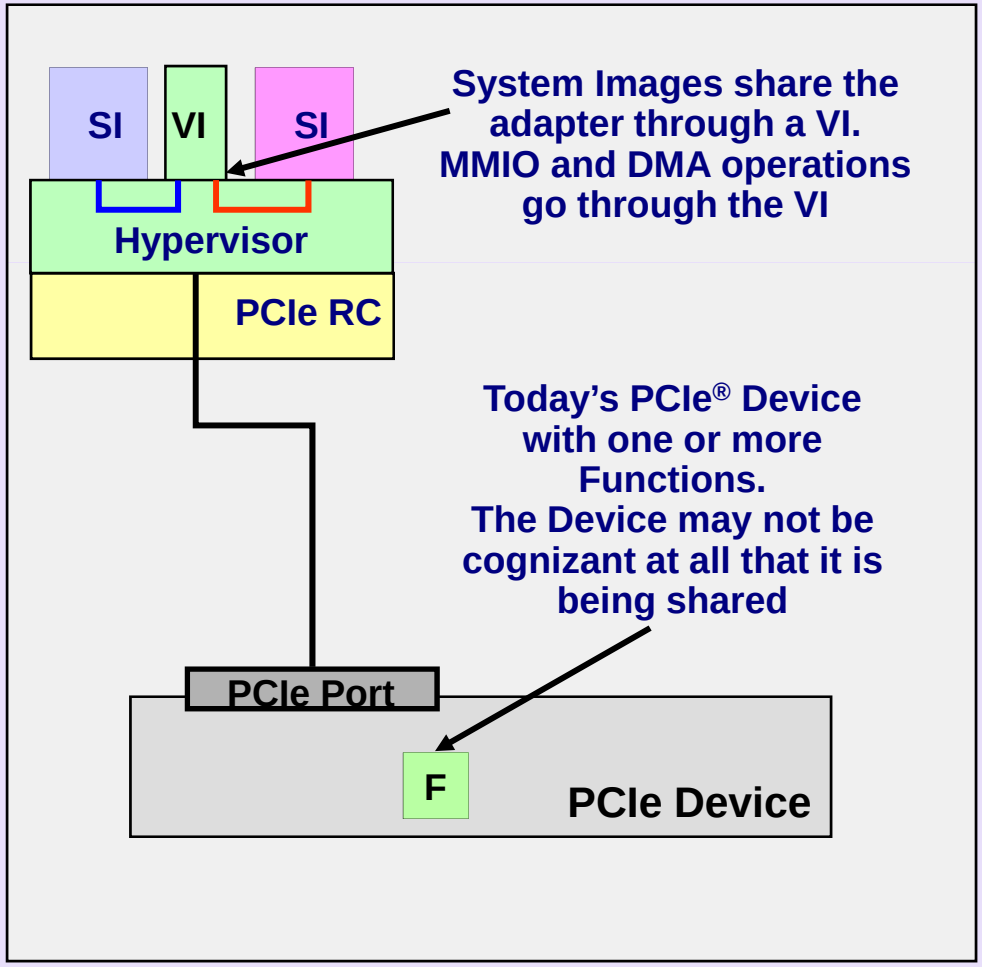
SR-IOV 1.1: One new bit…

- ARI-Capable Hierarchy Preserved
  - ✓ Section 3.3.2: Bit 1 in the SR-IOV Capabilities Register
  - ✓ Indicates whether device preserves Bit 4 (ARI-Capable Hierarchy) in the SR-IOV Control Register across $D3_{hot}$ to D0 power state transitions
    - Bit likely set by BIOS or other system initialization software
    - Existing software wouldn't know (or be able) to set the bit again after power management
    - Unclear how a device using this bit would react to having it change state after enumeration
  - ✓ Base specification strongly encourages implementing the PCI Power Management No_Soft_Reset bit
    - Doing so likely made preserving the ARI-Capable Hierarchy bit "automatic"
  - ✓ SR-IOV 1.1 strongly encourages devices to preserve ARI-Capable Hierarchy bit even if they do NOT support No_Soft_Reset (i.e. do an internal soft-reset when programmed from $D3_{hot}$ to D0)
  - ✓ Software must check **both** ARI-Capable Hierarchy Preserved **and** No_Soft_Reset to accurately determine whether the device preserves ARI-Capable Hierarchy
    - Components that only Set No_Soft_Reset are SR-IOV 1.1 compliant

I/O Virtualization Overview

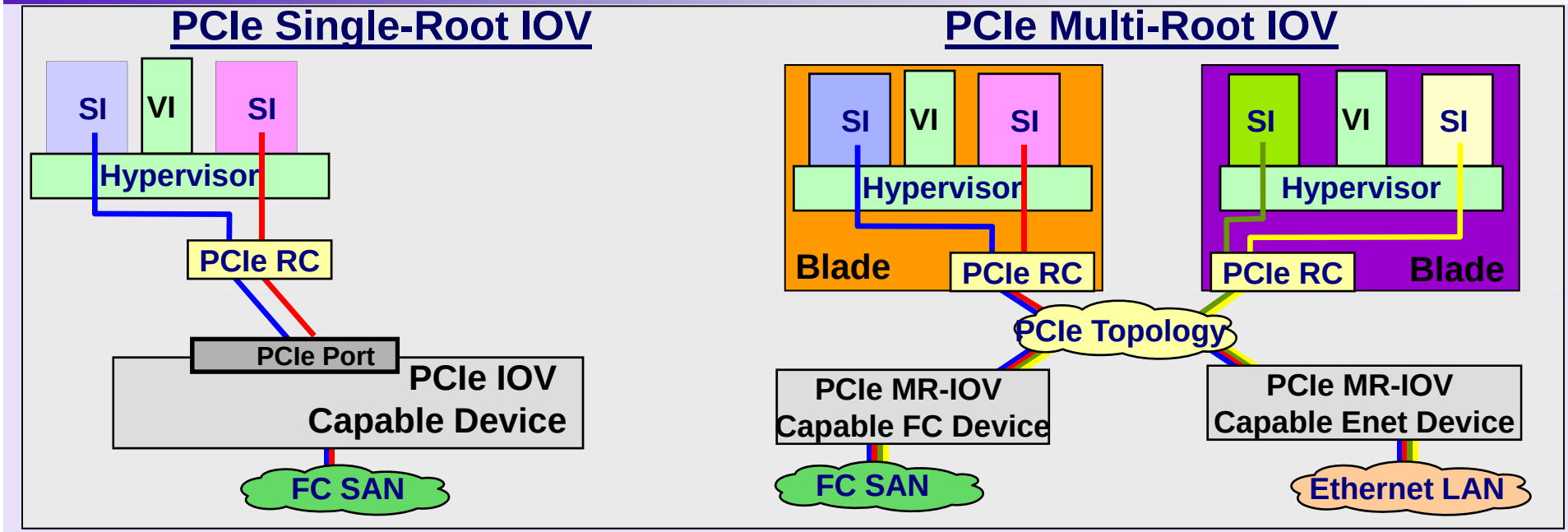
Prior Approach to IOV

VI Based PCI Device Sharing Example



- Virtualization Intermediaries (VI) are used to safely share IO
- Under this approach:
  - ✓ 1 or more System Images (SI) share the PCI device via a VI
  - ✓ Virtualization enablers are not needed in either the Root Complex (RC) or PCIe Device
  - ✓ The VI is involved in all IO transactions and performs all IO Virtualization Functions

PCI-SIG® IOV Overview



- PCI-SIG standardized mechanisms that enable PCIe Devices to be directly shared, with no run-time overheads:
  - ✓ Single-Root IOV - Direct sharing between SIs on a single system
  - ✓ Multi-Root IOV - Direct sharing between SIs on multiple systems
- PCI-SIG IOV Specification only covers Device's "north-side"
  - ✓ Some example usage models will be covered in other presentations

I/O Virtualization – What?

From an adapter point of view:

- One physical device looks like multiple devices
  - ✓ Multiple views of the same physical device
    - E.g., multiple views of a SAS controller or NIC
- Virtual devices appear completely independent
  - ✓ Each virtual device is assigned its own PCIe addresses
  - ✓ Each virtual device's BARs can be separated by the system page size.
  - ✓ Each virtual device appears as a separate PCIe Function in configuration space
  - ✓ May have different settings for various Configuration registers
  - ✓ Need to keep cross-"device" traffic isolated
  - ✓ An endpoint may export a mix of "base" Functions and IOV-enabled Functions in SR-IOV

I/O Virtualization – What?

From a system point of view:

- *System Image* (SI) is a real or virtual system of CPU(s), Memory, O/S, I/O, etc
  - ✓ SI is just another name for an OS running on top of a Virtualization Manager. (e.g., An instance of linux running in a Xen Dom0 or DomU).
  - ✓ Multiple SIs may run on one or more sets of hardware
    - E.g. VMWare running Win32 & Linux on a single CPU
    - E.g. Blade server running multi-OS each on a single blade
- Each System Image *sees* it's own PCI hierarchy
  - ✓ Even if NO end devices are actually shared
  - ✓ Only its *portion* of shared end devices

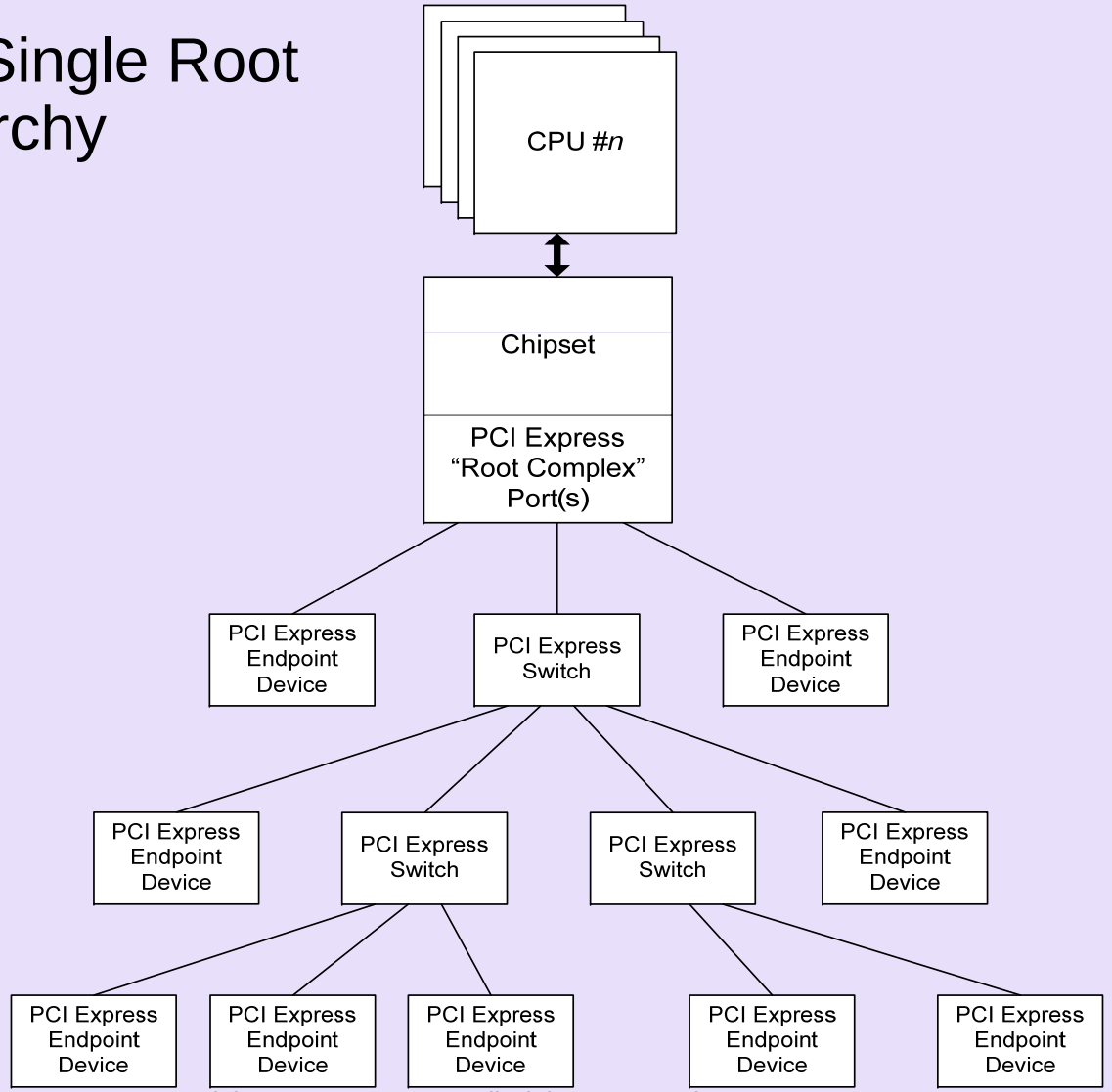
Definitions

I/O Virtualization – Definitions

- IOV – I/O Virtualization
  - ✓ A hardware method of exporting multiple views of the same *device*
- SR-IOV – Single Root I/O Virtualization
  - ✓ PCI-SIG defined methods for exporting multiple views of the same device in a traditional PCIe base fabric
  - ✓ No PCIe protocol changes. Works with existing PCIe fabrics
  - ✓ May be ignored by SR-IOV unaware software/firmware
    - In this case, it works just like base PCIe

I/O Virtualization – Definitions

- Example Single Root IOV Hierarchy

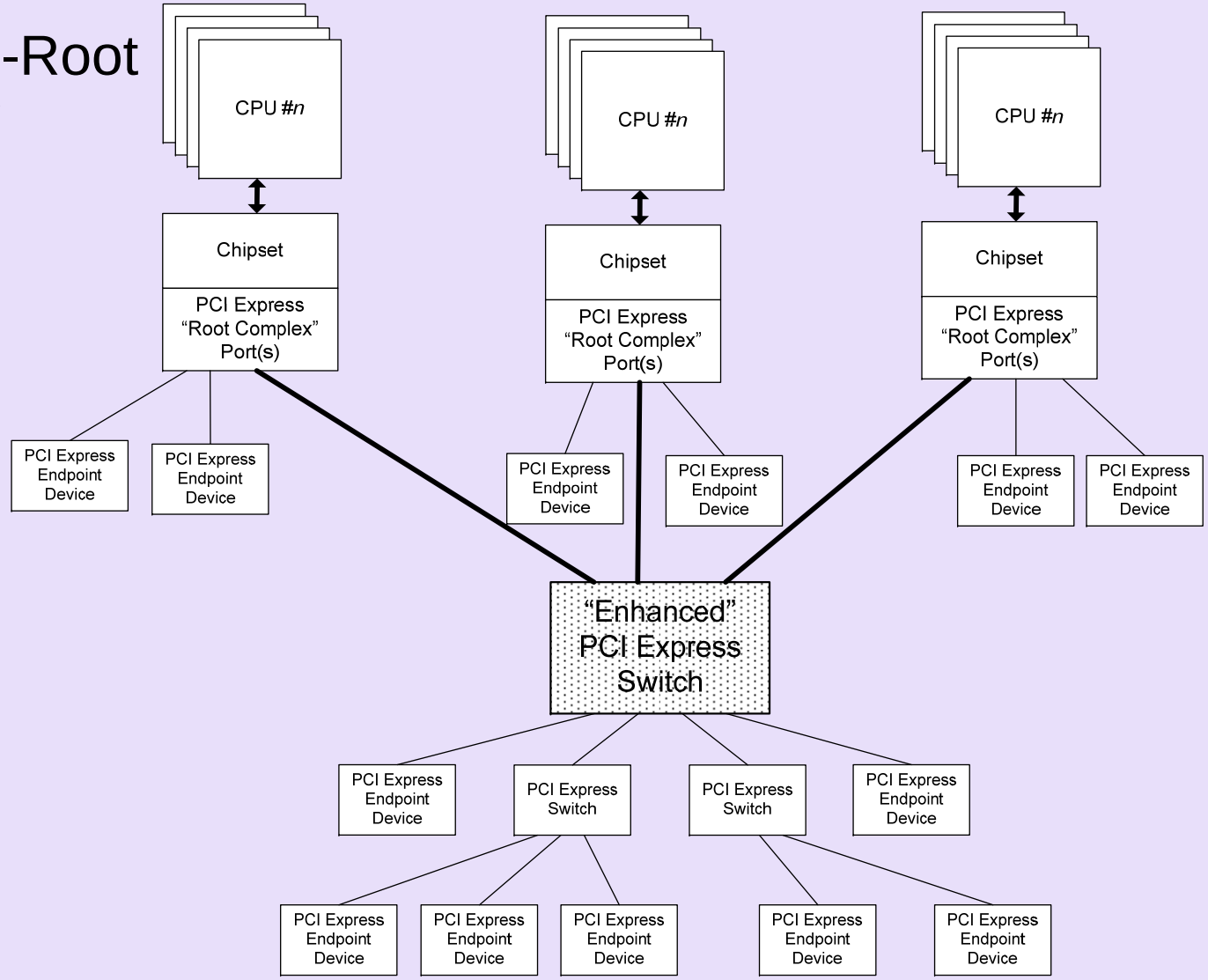


I/O Virtualization – Definitions

- MR-IOV – Multi-Root I/O Virtualization
  - ✓ PCI-SIG defined methods for exporting and managing multiple views of the same device in a PCIe fabric that can be connected to more than one root port
  - ✓ From each root ports' perspective, the fabric appears to be a traditional PCIe base fabric
  - ✓ No change to Root Complex. No change to non-MR endpoints
  - ✓ Changes to MR aware switches and endpoints to support multiple *Virtual Hierarchies*
  - ✓ PCIe protocol changes (TLP Header) and new automatic training sequences for MR aware switches and endpoints
  - ✓ Requires configuration and management:
    - Create and configure *Virtual Hierarchies*
    - Handle certain errors and events

I/O Virtualization – Definitions

- Example Multi-Root IOV Hierarchy



I/O Virtualization – Definitions

- Function
  - ✓ PCIe base-defined function as with base PCI and PCIe
- Virtual Function (VF)
  - ✓ A *virtual view* of a physical device
    - – Looks like a Function: RID, configuration space, memory space, etc.
- Physical Function (PF)
  - ✓ A Function that includes the SR-IOV capability.
    - – An anchor for creating *Virtual Functions* and reporting errors and events that cannot be attributed to a single *virtual function*
- Base Function (BF)
  - ✓ A Function that includes the MR-IOV capability.
    - – An anchor for creating/managing *Virtual Hierarchies* and PFs within VHs and reporting errors and events that cannot be attributed to a single VH
    - – Does not implement the *mission function* of the device

I/O Virtualization – Definitions

- System Image (SI)
  - ✓ The OS running in a domain (e.g., linux in Dom0 or Domu)
- Virtual Intermediary (VI)
  - ✓ Hypervisor that provides physical resource protection between SIs running on a system
- Single Root PCI Manager (SR-PCIM)
  - ✓ The SR-IOV specification's *name* for the configuration, management and protection software/firmware for SR-IOV implementations
- Multi Root PCI Manager (MR-PCIM)
  - ✓ The MR-IOV specification's *name* for the configuration and management software/firmware for MR-IOV implementations

Types of I/O Virtualization

Single Root (SR-IOV)

I/O Virtualization: SR-IOV

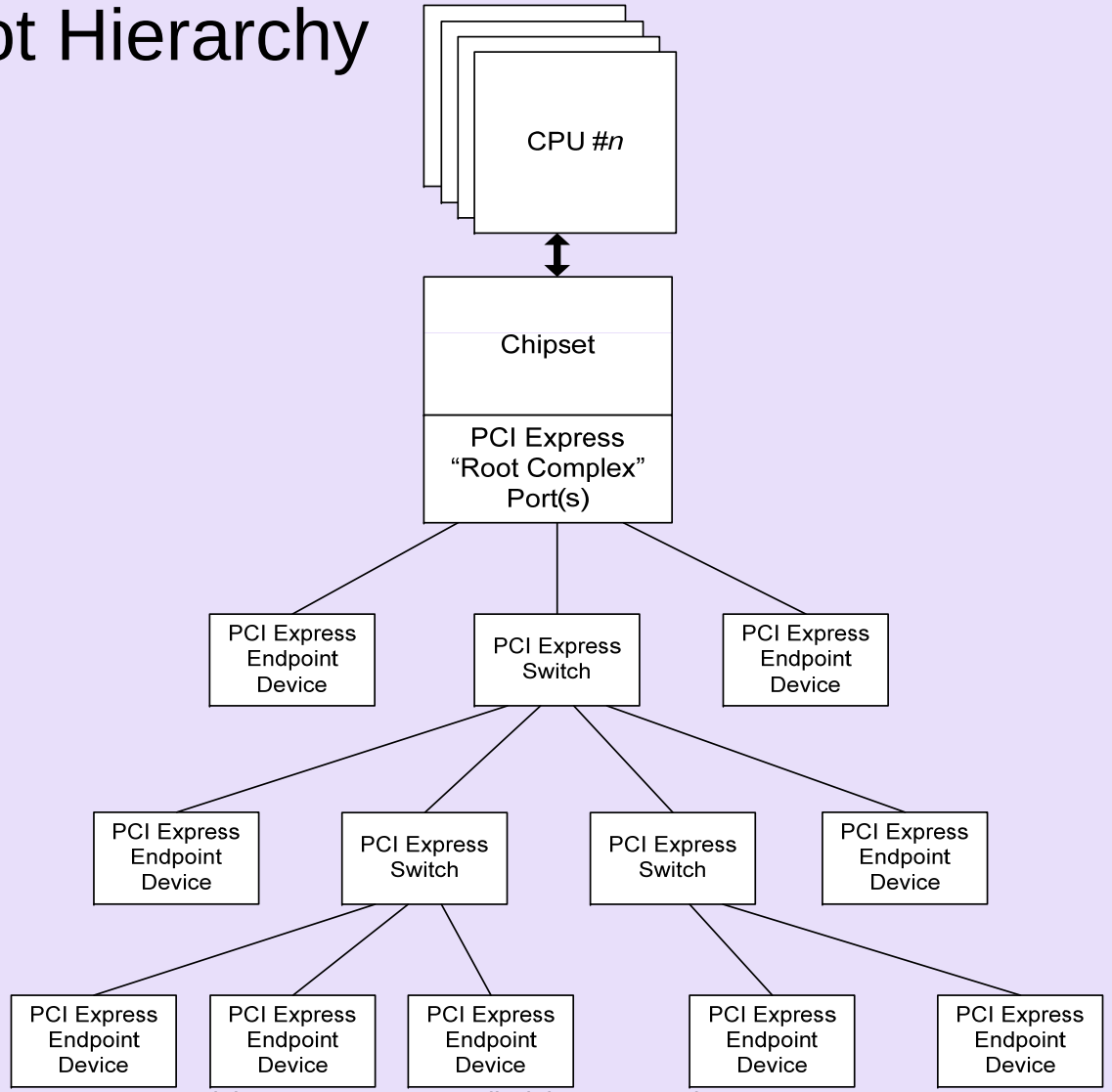
- Single Root IOV (SR-IOV)
  - ✓ Fits into existing PCIe hierarchies today
    - *Systems* with traditional single point of attachment to a PCIe fabric
    - Single PCIe address spaces – partitioned and allocated *above* the Root Complex
      - Uses Routing ID (the Bus/Device/Function number) in packets to track transactions back to the appropriate SI
  - ✓ Same PCIe base protocol
  - ✓ Supports up to 256 or several thousand virtual functions per device
    - More than 256 requires extra work

I/O Virtualization: SR-IOV

- Single Root IOV (Continued)
  - ✓ Existing Root Complex (+Optional ATC support)
  - ✓ Existing Switches (+Optional ARI support)
  - ✓ New Endpoint silicon
  - ✓ Presumes existence of a Virtualization Intermediary (VI) aka Hypervisor
    - Direct result of “don’t change the chipset!” philosophy
    - Opens market to existing systems
    - Shifts substantial burden to software

I/O Virtualization: SR-IOV

- Single Root Hierarchy

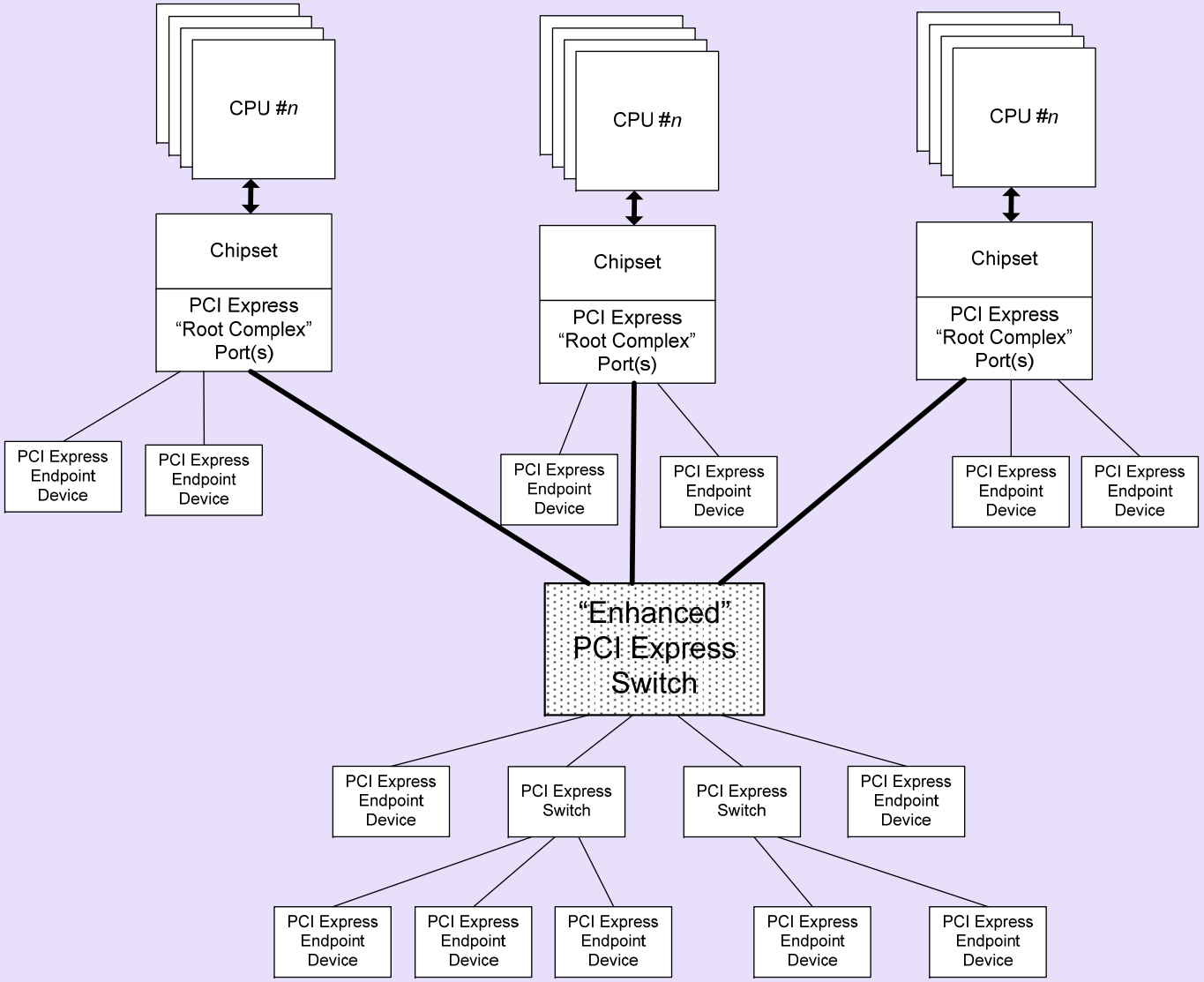


Multi-Root (MR-IOV)

I/O Virtualization: MR-IOV

- Most obvious example is a blade server with a PCIe “backplane”
- New PCIe hierarchy construct
  - ✓ Effectively a (mini) fabric
  - ✓ Logically partitions the PCIe fabric into multiple Virtual Hierarchies (VHs) all sharing the same physical hierarchy
  - ✓ Targets “small” systems with 16-32 Root Ports as likely max
  - ✓ A “special” management hierarchy does initial configuration

I/O Virtualization: MR-IOV



I/O Virtualization: MR-IOV

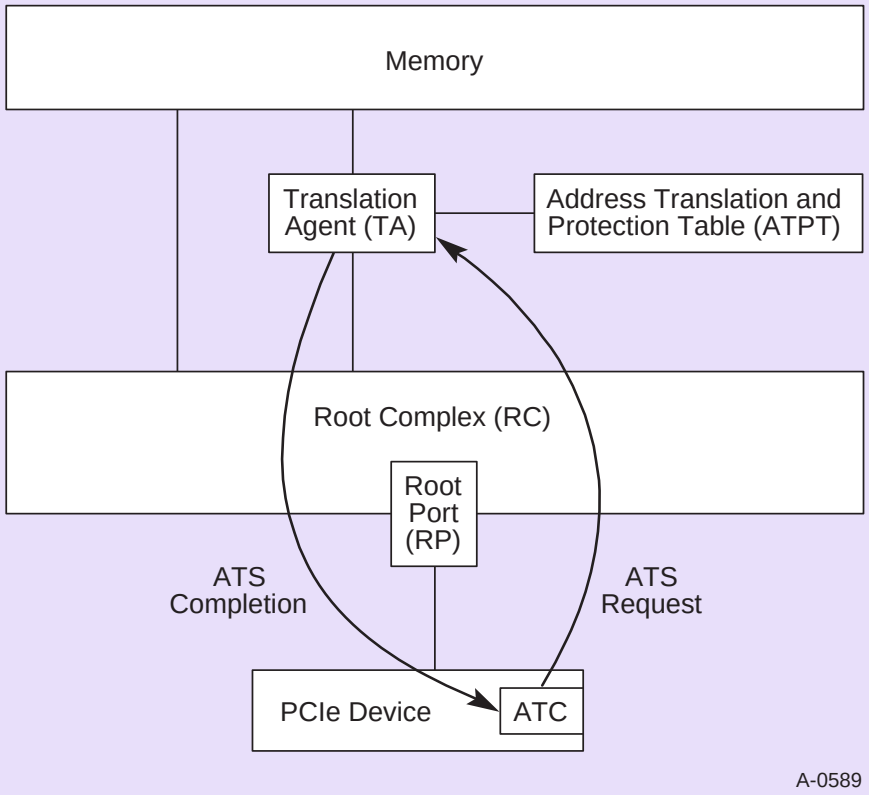
- New protocol
  - ✓ TLP header for MR-IOV
    - Used for routing TLPs within the MR-aware fabric
  - ✓ New training sequence
    - Try new MR training sequence first
    - Try PCIe base training sequence second
- New Switch silicon
  - Support for new TLP header and new training sequences
  - Initial MRA point for discovering MR-aware devices
  - Lots of configuration, management and error handling "knobs". (virtual switches, virtual hot-plug support, …)
  - Support for multiple Virtual Hierarchies (routing tables, etc.)

I/O Virtualization: MR-IOV

- New Endpoint silicon
  - ✓ TLP header for MR-IOV
    - Which VH is being targeted by the TLP
  - ✓ New training sequence
    - Try new MR training sequence first
    - Try base training sequence second
  - ✓ Support for multiple Virtual Hierachies
  - ✓ Optional support for SR-IOV within each VH

Address Translation Services (ATS)

Address Translation Services



Optional normative cooperative protocol to extend address translations to “trusted” devices

TLP bit indicates translated addresses in read/write transactions

Must be enabled by software via the ATS capability in Functions

Address Translation Services

- Defines a set of transactions PCIe components can use to exchange and share translated addresses
  - ✓ With an I/O MMU, PCIe bus addresses can map to different system physical 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 untranslated 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 if ATS is also supported by the root complex.
  - ✓ New PCIe transactions for translation Requests, Completions, and Invalidations
  - ✓ Modified PCIe TLP header (2 reserved bits) to indicate whether a given address is pre-translated or not

Address Translation Services

- Implementation is optional even for IOV aware devices and the root complex (Optional Normative)
- After a reset, ATS is disabled and may be enabled by ATS aware system software (If the root complex also supports ATS)
- System software is required to send invalidates downstream to any ATS-enabled device for any transaction cached by that device's ATC when that translation changes or is removed (TLB flush)
- Trust issues: Since enabling ATS allows the Function to generate read/write transactions that bypass the I/O MMU protection, the system software *should* trust that Function and its driver before enabling ATS

Endpoint Impact

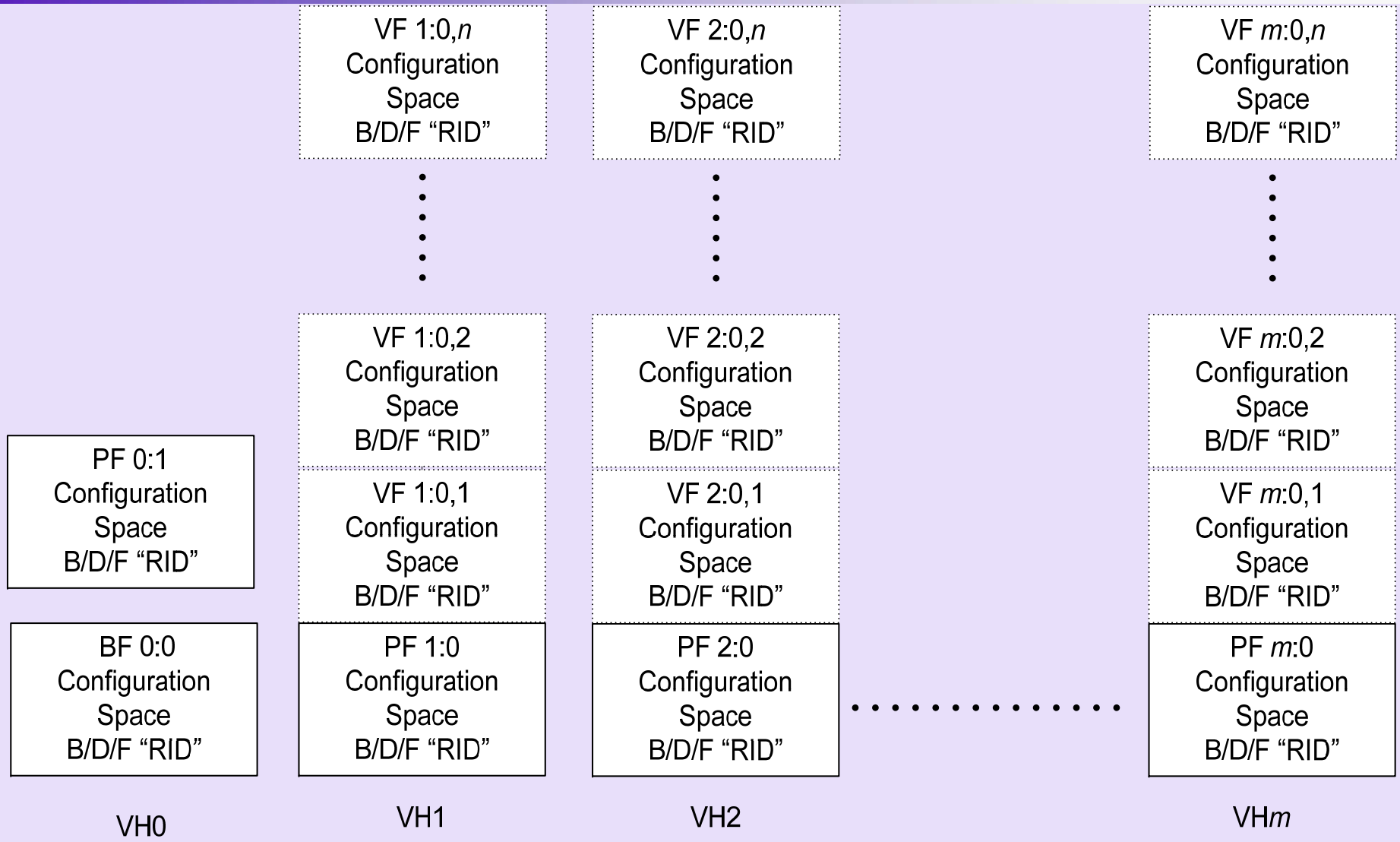
Endpoint Impact

- Each SI that needs direct access to the Function needs its own Virtual Function
  - ✓ The device needs *n* sets of configuration space to support *n* VFs
  - ✓ The device needs *n* sets of registers to support *n* VFs
  - ✓ The device may need *n* sets of internal logic to support *n* VFs
- VFs and PFs must support Function Level Reset (FLR)
  - ✓ FLR targeting a VF *resets* only that VF. Other VFs and the PF are not affected
  - ✓ FLR targeting a PF *resets* the PF including the VF Enable bit in the SR-IOV cap. VFs are destroyed if the PF is reset

Endpoint Impact

- Single Root PCI Manager (SR-PCIM) creates, manages and assigns VFs to SIs
- SR-PCIM uses the PFs SR-IOV extended capability to create VFs
  - ✓ Number of VFs that the PF should create (NumVFs)
  - ✓ VF BAR registers (one set of BARs for all VF's memory space. BARs are *not* replicated in VF config space.
  - ✓ VF State Table and State Change status and interrupt
    - Supports VF migration between VHs if the device is MR-IOV aware
    - Optional and may be disabled by SR-PCIM

Endpoint Impact – Example Multi-Root View of Configuration Space (Single Function Device)



Endpoint Impact

- **BF: Base Function** – A Function that appears only in the Multi-Root management hierarchy
  - ✓ Used to manage and configure the device's MR-IOV environment
    - the active Virtual Hierarchies and number of Physical Functions
    - allocations of Virtual Functions to each Virtual Hierarchy
  - ✓ Not a *mission* function (e.g. can't be used for booting or work)
    - Exists only for MR-IOV purposes
  - ✓ Handles all errors that cannot be attributed directly to a particular Virtual Hierarchy
    - E.g. link errors and errors that may affect more than one Virtual Hierarchy are sent via the Base Function to MR-PCIM in most cases

Endpoint Impact

- **PF: Physical Function** – A PCIe Function that includes the SR-IOV extended capability
  - ✓ Used to manage and configure the device's SR-IOV environment
    - the active Virtual Functions
  - ✓ Is a *mission* Function. (e.g., can be used for boot or work)
  - ✓ Single function device when used in:
    - Single Root only: has one
    - Multi-Root: Has *m* when there are *m* Virtual Hierarchies
  - ✓ Multi-function device with *x* Base Functions when used in:
    - Single root only: has *x*
    - Multi-root: Has *m*x* when there are *m* Virtual Hierarchies
  - ✓ Error reporting for any error that CAN be attributed to this Virtual Hierarchy but CANNOT be attributed directly to a particular Virtual Function

Endpoint Impact

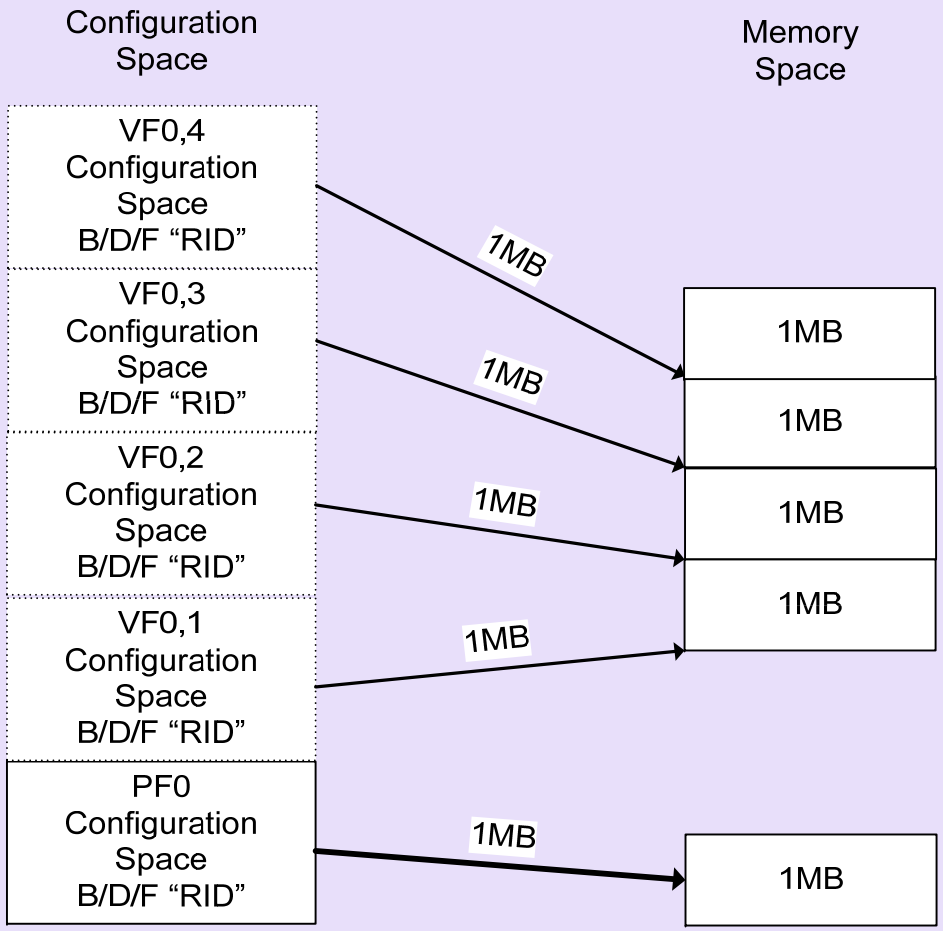
- **VF: Virtual Function** – A *virtual view* of a Physical Function
  - ✓ Has configuration space and its own unique Requestor ID
    - Type 0 header
    - Many fields are implemented directly
    - Some fields are read-only copies of the Physical Function
    - Some fields do not exist (e.g. VF BARs are implemented in the PF.)
    - It is expected that the VI (Hypervisor) will provide access to fields not implemented in the VF config space
  - ✓ May have PCIe memory space (e.g. *Mission* registers)
  - ✓ Must not have PCIe I/O space
  - ✓ Has its own set of MSI and/or MSI-X registers for interrupts

Endpoint Impact

- VF Base Address Registers (BARs)
  - ✓ BARs are not implemented in each VF
  - ✓ Instead, they are implemented as a single set of BAR-like registers in the SR-IOV extended capability (VF BARx)
    - Additional decoders now required for a VF-specific set of BARs
    - Individual VF BARs may have required large CAMs (Cost)
  - ✓ Logically concatenates Virtual Functions into one contiguous chunk of PCIe memory space per implemented VF BAR
    - Note that the System Page Size field in the SR-IOV capability allows SR-PCIM to specify the minimum stride between VFs address spaces
- Expansion ROM BAR not implemented for VFs
  - ✓ Use the PFs expansion ROM BAR
    - PCI ROM BAR shared decoding is not permitted (A PCI legacy feature)

Endpoint Impact – Single Root view of Configuration Space

- Example VF BARx Memory Mapping with 1MB Size



Protocol Details

Protocol Details – Single Root

- No new TLPs
  - ✓ The same PCIe base protocol that we know and love
- Routing ID (RID) indicates who sent the packet
  - ✓ MemRead and MemWrite include the sender's RID
  - ✓ Root Complex *may* use the RID to help translate address in the context of the SI that the device is assigned to
  - ✓ Packets sent downstream that are routed by address are still routed by address to the appropriate function in the endpoint
  - ✓ Packets sent downstream that are routed by RID are decoded in the endpoint and internally routed to the appropriate function

Protocol Details – Multi-Root

- New TLP Header – Virtual Hierarchy identifier
  - ✓ Every TLP will include the new TLP header in an MR-A fabric
    - Local to that bus segment
    - Tied to a particular root complex by the programming of routing table in MR-A switch hierarchies
  - ✓ New DLLPs
    - Per-plane reset & Additional flow control
- New flow control
  - ✓ Virtual Link (VL) concept
    - MR IOV equivalent of Virtual Channels (VCs)
    - Provides wide range of implementation flexibility

Thank you for attending the
PCI-SIG Technology Seminar

For more information please go to
www.pcisig.com