PCI Express I/O Virtualization Explained

Richard Solomon
LSI Corporation

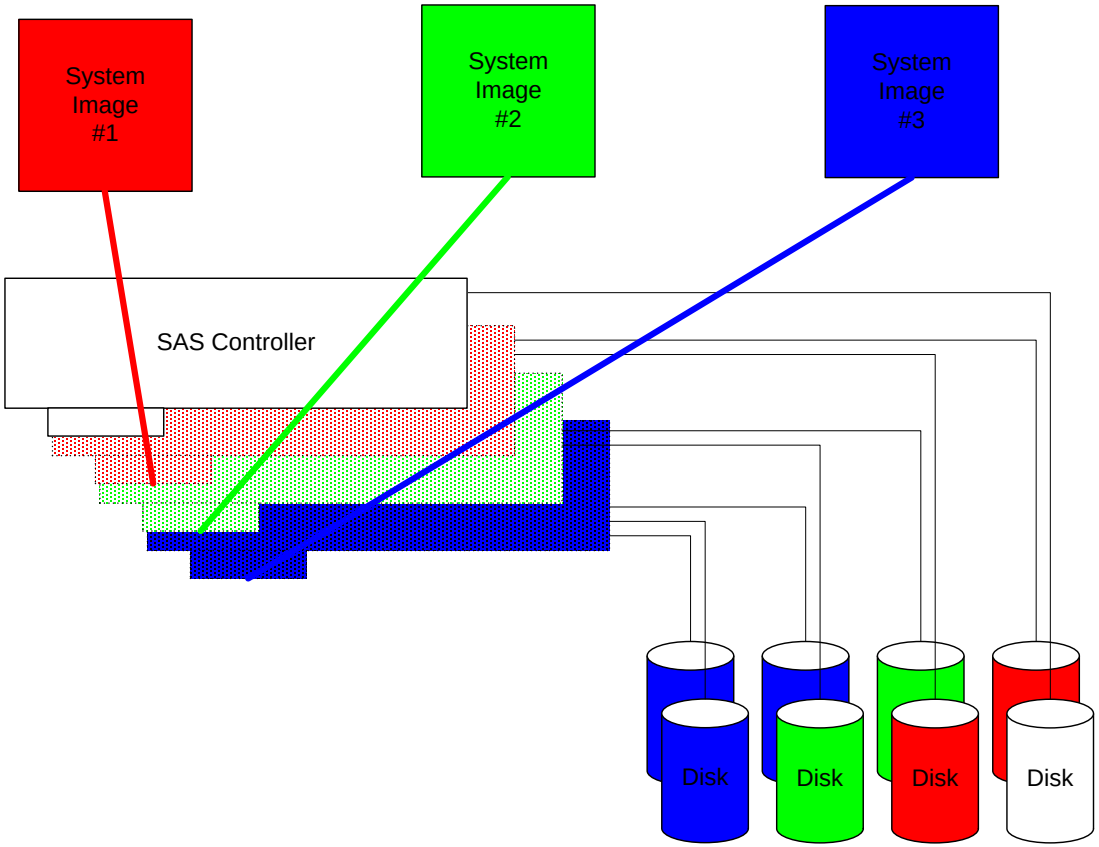
Agenda

- PCI Express® I/O Virtualization (IOV) Overview
  - Single Root (SR-IOV)
  - Multi-Root (MR-IOV)
- Secret Three Letter Acronym (TLA) Decoder Ring
- Programming IOV Devices
  - Configuration Space Mapping
  - Memory Space Mapping
  - Programming Registers

PCI Express I/O Virtualization

IOV Overview

- Makes one device "look" like multiple devices
- Generally motivated by cost
- Seek performance *within* the cost envelope



IOV Overview

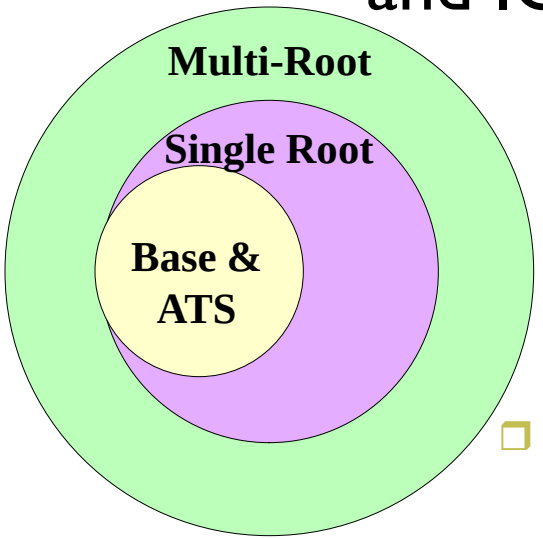
- From an adapter point of view:
  - One physical device looks like multiple devices
  - Virtual devices appear completely independent
    - May occupy different PCI memory ranges
    - May have different settings for various PCI Configuration registers
    - Need to keep cross-"device" traffic isolated
    - Each virtual device *may* be a multi-function device

IOV Overview

- 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
      - 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" (SI) needs to "see" it's own PCI hierarchy
    - Even if NO end devices are actually shared
    - Only its "portion" of shared end devices

IOV Overview

- Attachment of existing PCIe 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 Base specification
  - Multi-Root capabilities are superset of the Single Root capabilities

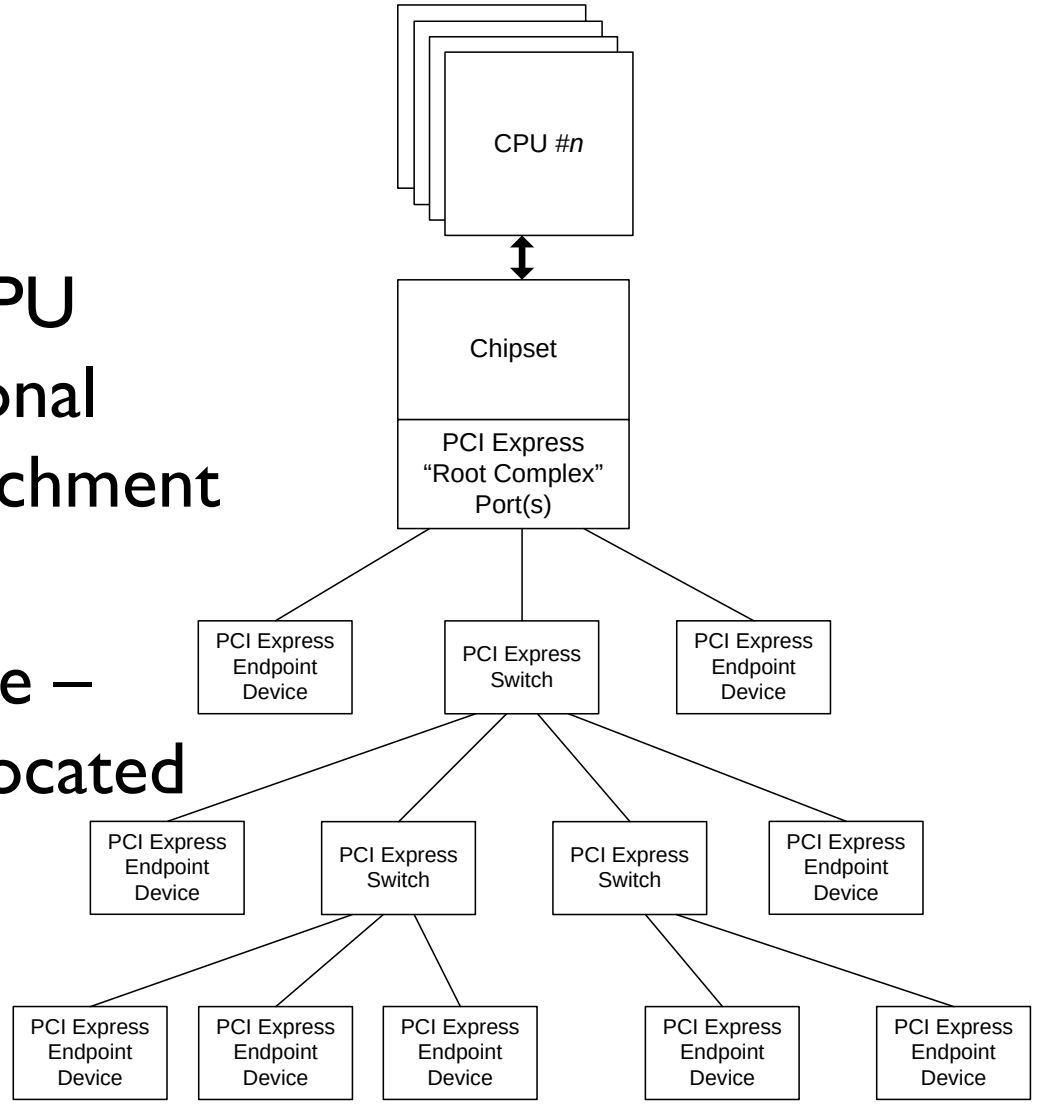


"Concentric Circles" model

- 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

Single Root IOV (SR-IOV)

- Fits into existing PCI hierarchies today
  - Single and Multi-CPU boxes with traditional single point of attachment to PCI
  - Same address space – partitioned and allocated "above" the Root Complex



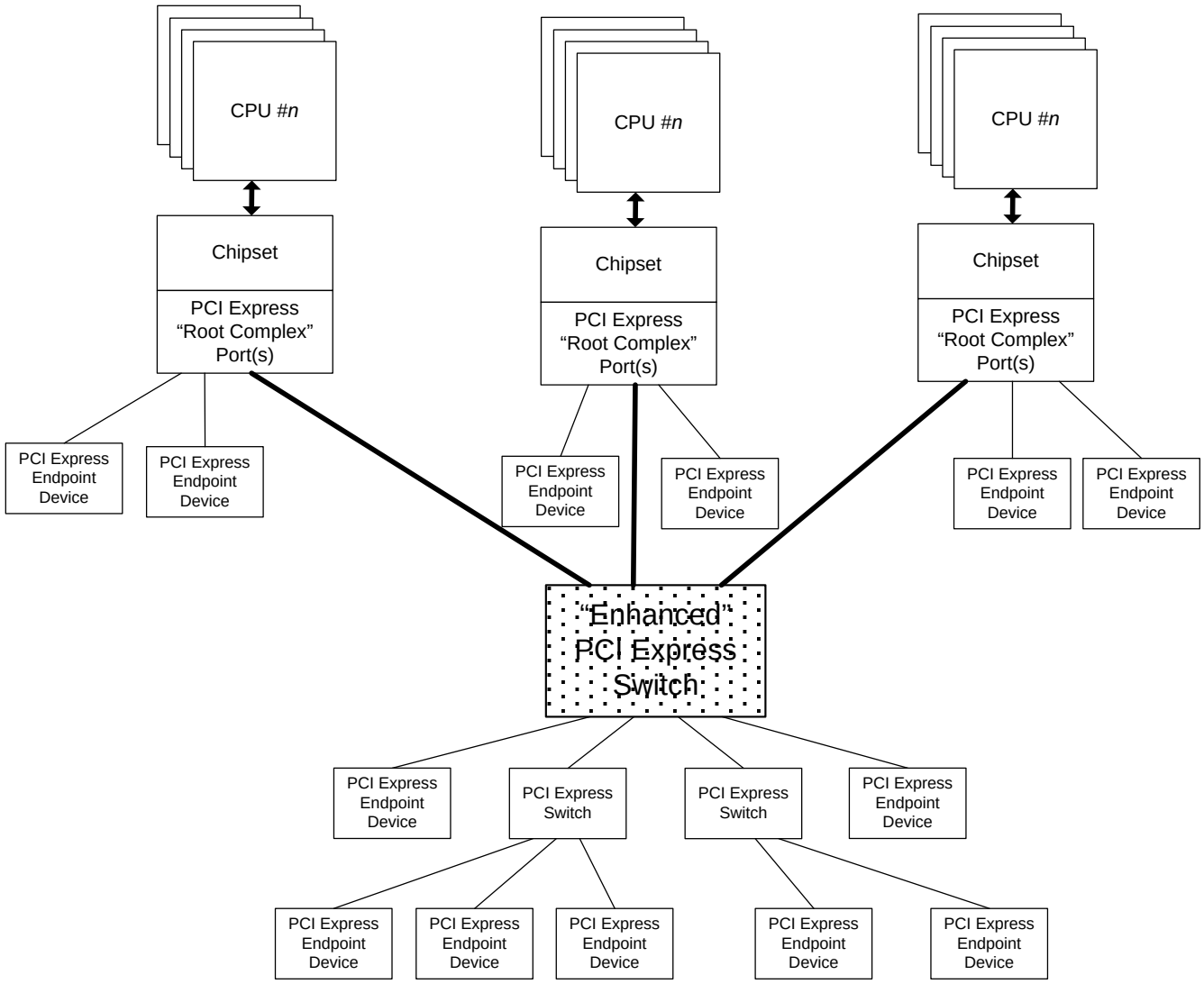
Single Root IOV (SR-IOV)

- Existing or absolutely minimally changed Root Complex (i.e. chipset) and Switch silicon
- New Endpoint silicon
- Presumes existence of a Virtualization Intermediary (VI) aka a Hypervisor
  - Direct result of "*don't change the chipset!*" philosophy
  - Opens market to lots of existing or simply-derived systems
    - E.g. might need new BIOS or chipset revision
  - Shifts ***substantial burden*** to software

Multi-Root IOV (MR-IOV)

- Most obvious example is a blade server with a PCIe "backplane"
- New PCIe hierarchy construct - (mini) fabric
  - Logically partitions the hierarchy into multiple Virtual Hierarchies (VHs) all sharing the same physical hierarchy
  - Targets "small" systems (16-32 Root Ports likely max)
    - Workgroup saying "Our yardstick is a yardstick" i.e. the typical implementation is a system occupying not more than about 3 feet cubed
    - Architected to allow larger, but not optimized that way

MR-IOV Hierarchy



Multi-Root IOV (MR-IOV)

- 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
- Management model

Secret

Three Letter Acronym

(TLA) Decoder Ring

Decoder Ring

- Function
  - The PCIe-addressed portion of a device devoted to one distinct "chunk" of that device's operation
    - E.g. one SAS port, or the ethernet side of a combination ethernet/SAS device
- Virtual Function (VF)
  - A "lightweight" implementation of one "view" of a single device Function which is being virtualized to appear as multiple

Decoder Ring (cont'd)

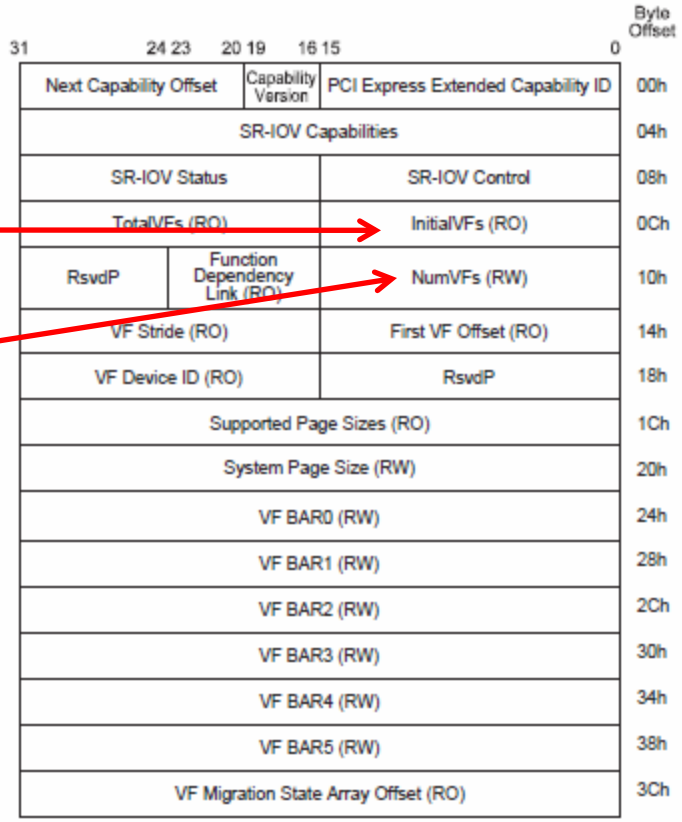
- Physical Function (PF)
  - Contains the SR-IOV control structure
  - Used to manage a set of associated Virtual Functions
- Base Function (BF) *[MR-IOV *ONLY*]*
  - Contains the MR-IOV control structure
  - Used to manage Virtual Hierarchies and Physical Functions
  - Is *NOT* usable for "real" work of the device

Programming IOV Devices

Configuration Space Mapping

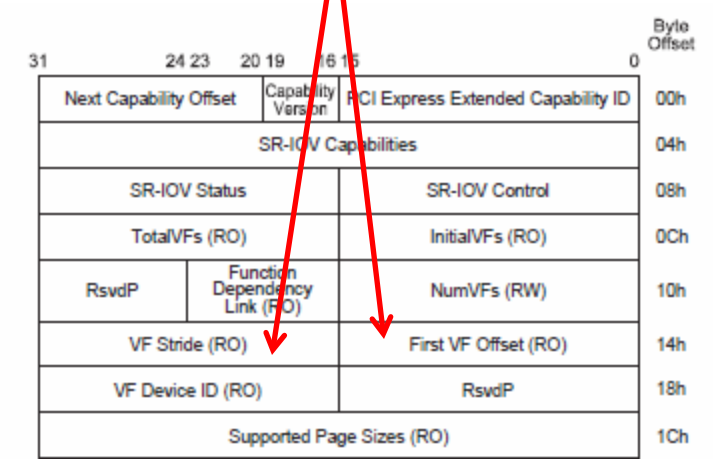
Configuration Space Mapping

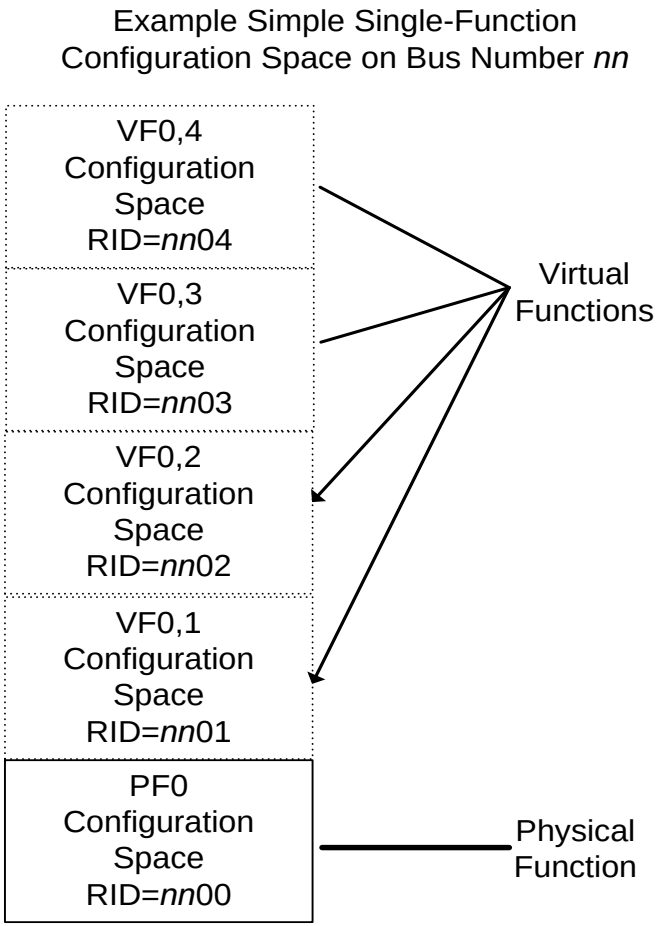
- Determine desired number of Virtual Functions from *InitialVFs* field
- Program *NumVFs* field to match
- Multi-Root adds a further layer where configuration software ***first*** allocates VFs to Virtual Hierarchies – thus *InitialVFs* may be less than *TotalVFs*



Configuration Space Mapping

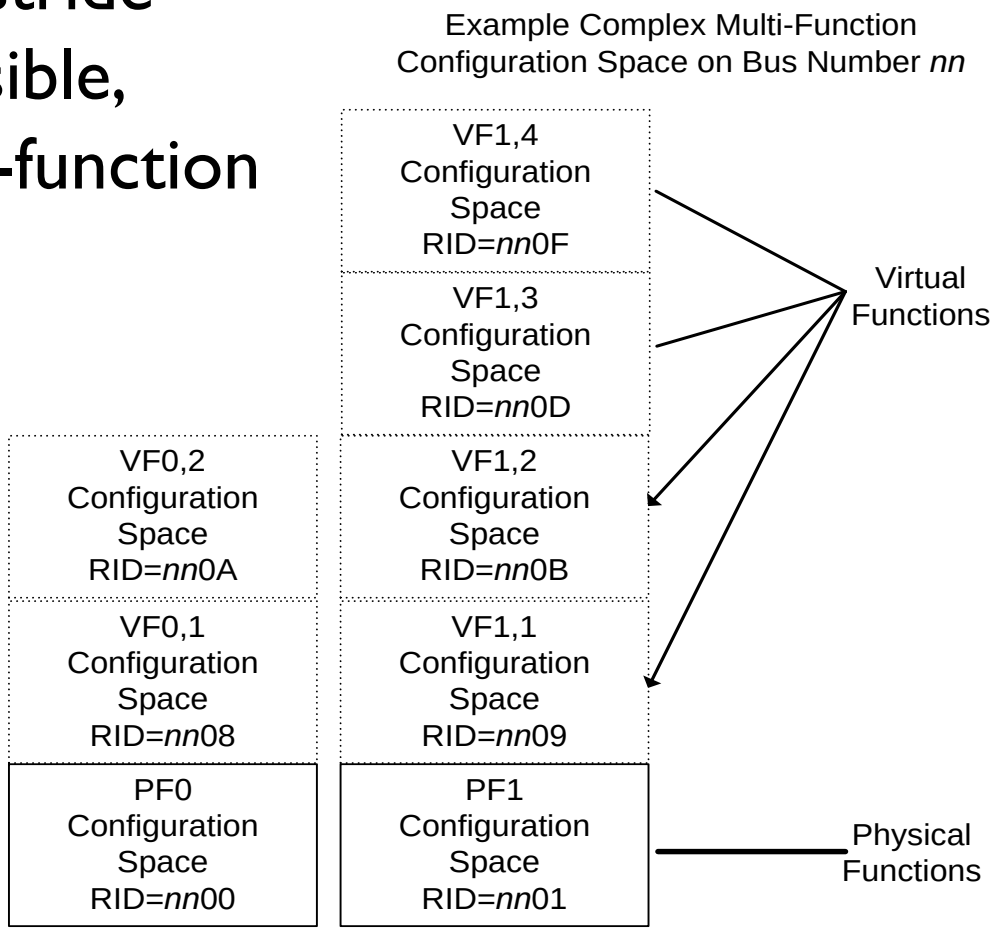
- Bus Number / Device Number / Function Number (BDF) field now known as RoutingID (RID)
- RIDs of VFs found from SR-IOV configuration



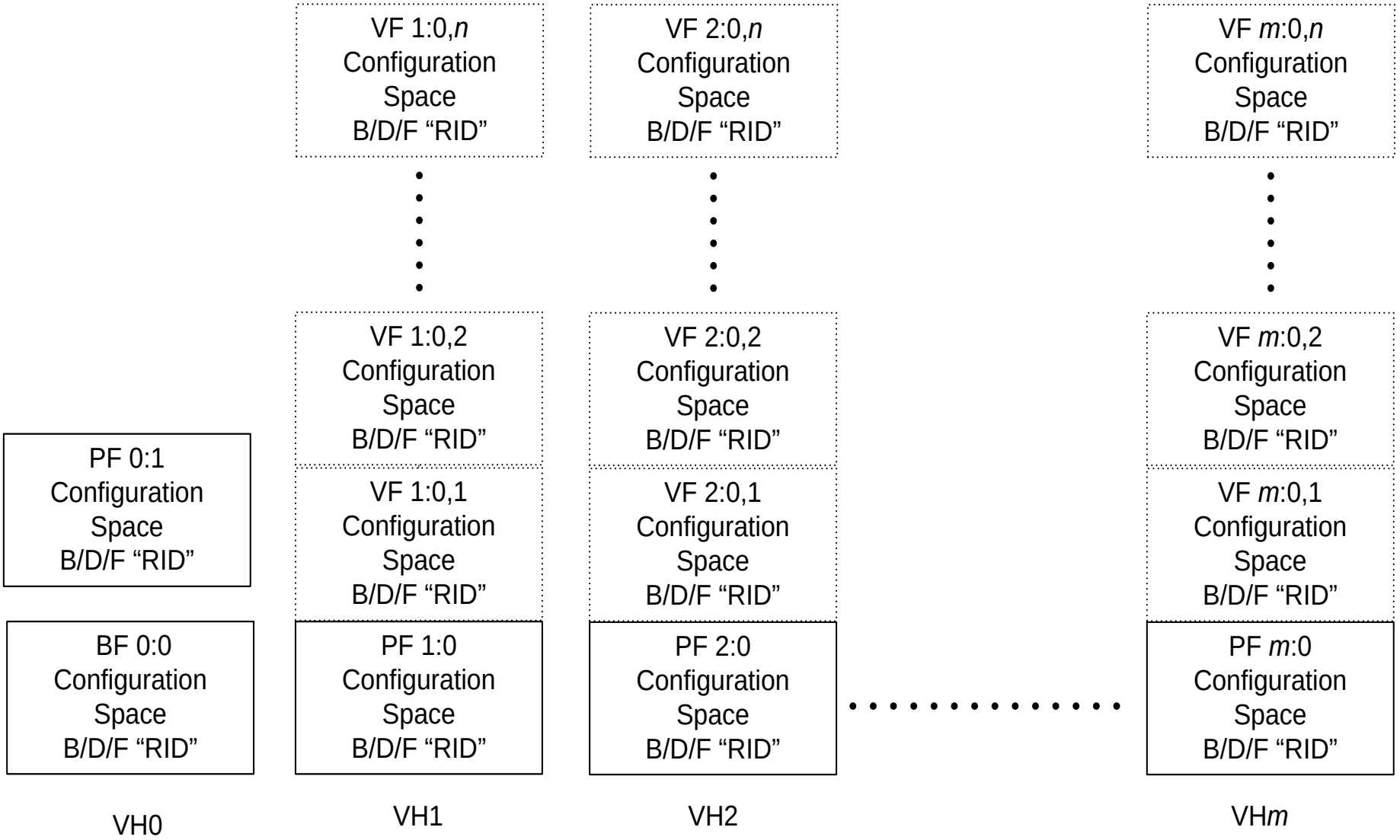


Configuration Space Mapping

- More complex offset/stride combinations are possible, particularly with multi-function devices
- E.g. PF0 and PF1 both set to:
  - Offset=8
  - Stride=2



Configuration Space Mapping - Multi-Root



Memory Space Mapping

Memory Space Mapping

- Virtual Function memory mapped somewhat differently than Physical Function
- All VFs share single set of Base Address Registers

PF:

(PCI)

31 16	15 0	
Device ID	Vendor ID	00h
Status	Command	04h
Class Code	Revision ID	08h
BIST / Header Type / Latency Timer	Cacheline Size	0Ch
Base Address Registers		10h–24h
Cardbus CIS Pointer		28h
Subsystem ID	Subsystem Vendor ID	2Ch
Expansion ROM Base Address		30h
Reserved	Capabilities Pointer	34h
Reserved		38h
Max_Lat / Min_Gnt / Interrupt Pin	Interrupt Line	3Ch

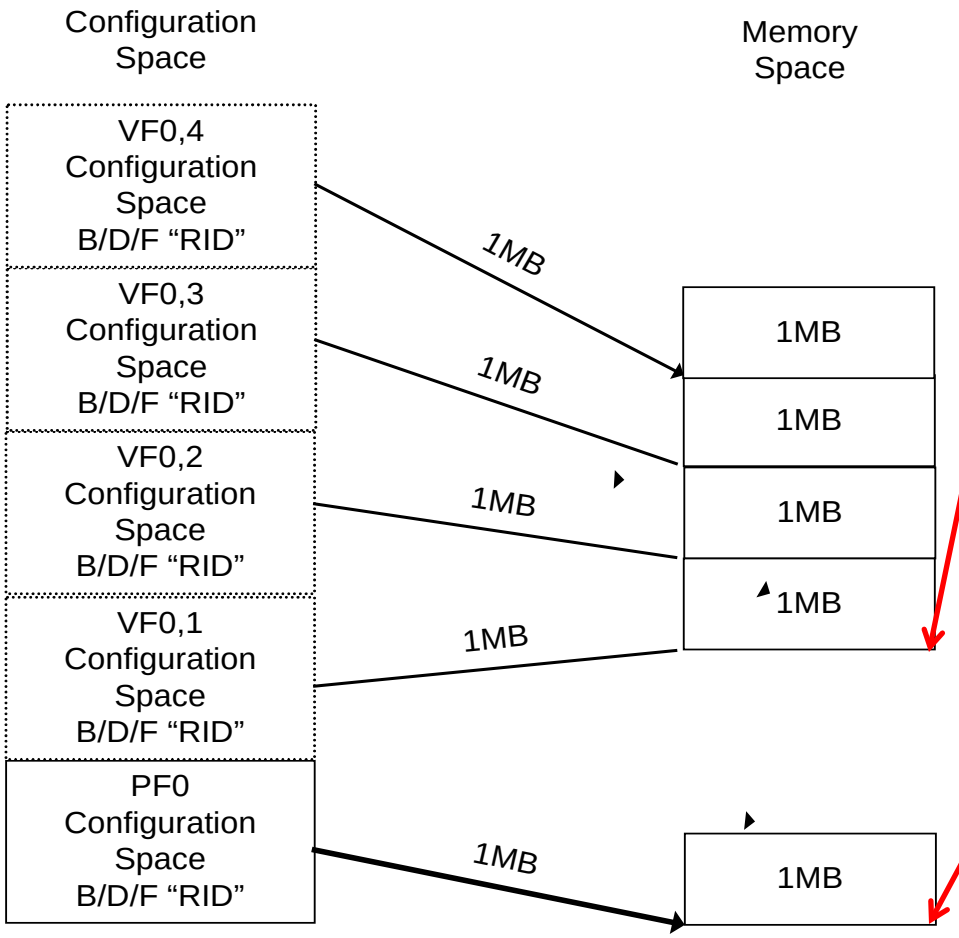
VF:

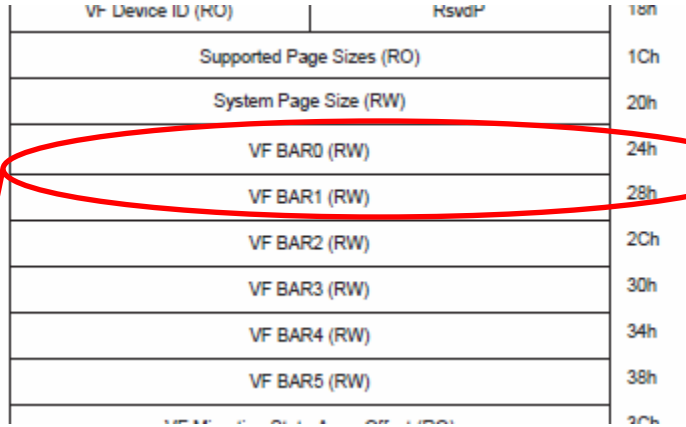
(SR-IOV)

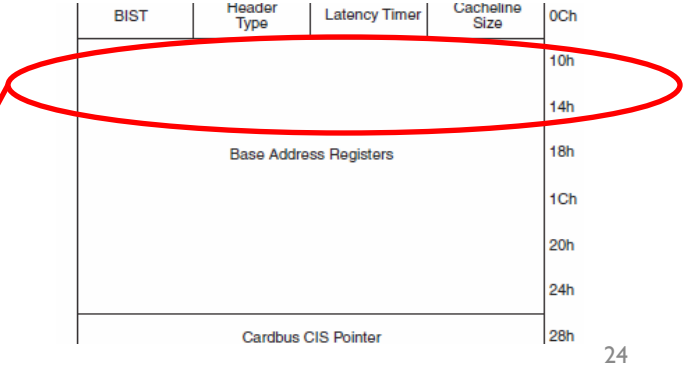
31 24	23 20	19 16	15 0	Byte Offset
Next Capability Offset	Capability Version		PCI Express Extended Capability ID	00h
SR-IOV Capabilities				04h
SR-IOV Status			SR-IOV Control	08h
TotalVFs (RO)			InitialVFs (RO)	0Ch
RsvdP	Function Dependency Link (RO)		NumVFs (RW)	10h
VF Stride (RO)			First VF Offset (RO)	14h
VF Device ID (RO)			RsvdP	18h
Supported Page Sizes (RO)				1Ch
System Page Size (RW)				20h
VF BAR0 (RW)				24h
VF BAR1 (RW)				28h
VF BAR2 (RW)				2Ch
VF BAR3 (RW)				30h
VF BAR4 (RW)				34h
VF BAR5 (RW)				38h
VF Migration State Array Offset (RO)				3Ch

Memory Space Mapping

Simple Example Single-Function Device with 1MB Memory Requirement

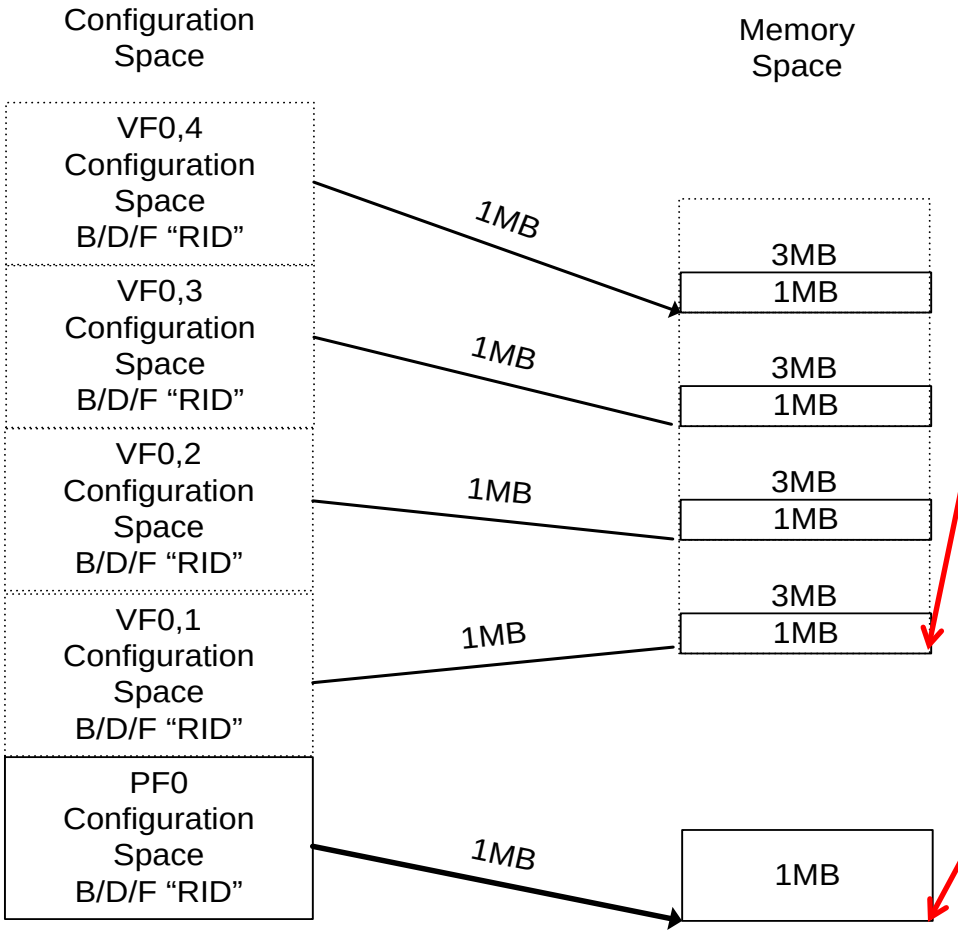


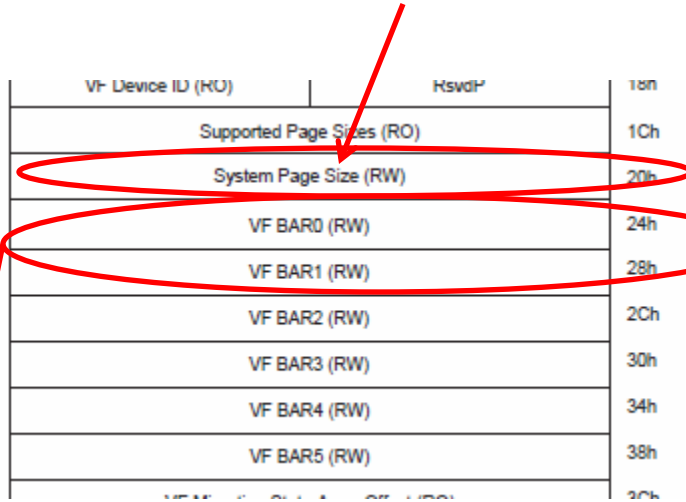


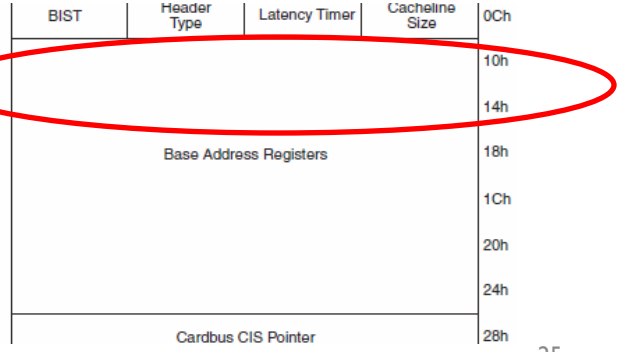


Memory Space Mapping

Single-Function Device with 1MB Memory Requirement on System w/4MB Pages



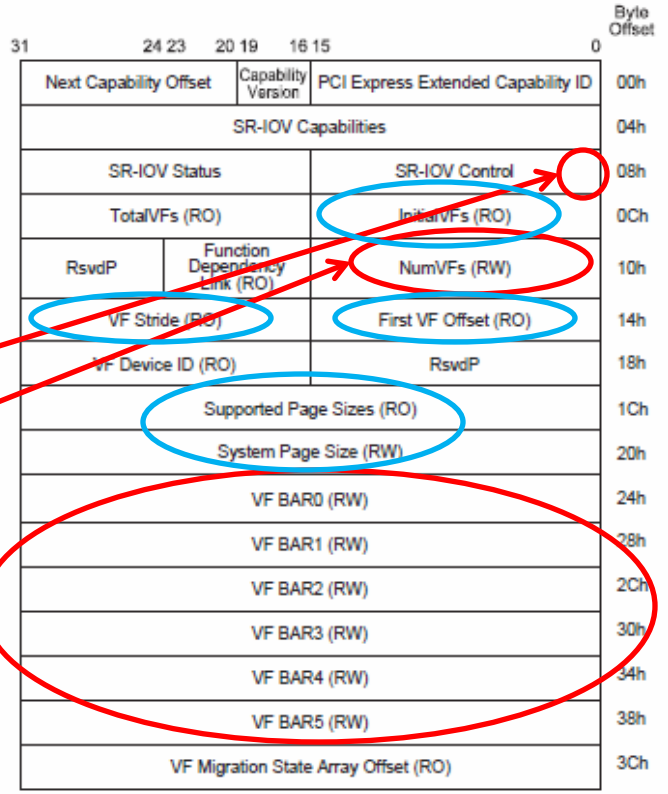




Programming Registers

PF Configuration Space

- Generally owned by Hypervisor
  - "Normal" PCIe config registers programmed by BIOS and Hypervisor's base OS
  - SR-IOV config registers programmed by Hypervisor
  - Key fields written:
    - *VF Enable*
    - *NumVFs*
    - *VF BARn*



VF Configuration Space

- Virtual Functions implement "lightweight" version of standard PCIe configuration space
  - Goal was to minimize hardware "cost" of VFs
  - Various register fields are either:
    - Base – operate as "normal"
    - Reserved – have no meaning in SR-IOV
    - Hardwired – have meaning as "normal" but aren't changeable
  - SR-IOV 1.1 spec calls out specific bit-by-bit definitions in Sections 3.4 through 3.7

VF Configuration Space

- Each Virtual Function is owned by a guest OS
- Hypervisor responsible for emulating / faking any undefined behavior
  - E.g. VF BARs – guest obviously cannot control true memory address of its VF, but believes it needs to do so
  - E.g. PCIe Link Control registers – guest cannot be allowed to bring the Link down for instance!

VF Configuration Space

- VendorID / DeviceID / Subsystem IDs
  - Work as expected – DeviceID may vary from PF
- PCI Command Register
  - I/O & Mem Enables hard-coded to 0
    - All VFs share single Memory Space Enable (MSE) bit in the VF capability structure (in the PF's config space)
  - Bus Master Enable works on the VF as expected
- PCI Base Address Registers
  - Read-only 0 in VFs
    - VFs memory mapped via mechanism previously discussed using VF BARs in the VF capability structure

Resets

- Hypervisor must block true PCIe bus reset from guest Oses
- Function Level Reset (FLR) used to reset single VF
- FLR to PF will reset ALL VFs just like a bus reset

Device-Specific Registers

- Vendor and device-specific
- "Normal" memory-mapped registers generally appear in VFs nearly identically to the PF
  - Goal of SR-IOV is guest OS "ignorance" of IOV
  - Existing device drivers should run as expected in guest

Sources of More Information

Where to find more information

- Source for all things PCI Express:

http://www.pcisig.com

(Company must be a member to access most content)

- Single Root and Multi-Root IOV Specs:

http://www.pcisig.com/members/downloads/specifications/iov/sr-iov1_1_20Jan10.pdf
http://www.pcisig.com/members/downloads/specifications/iov/mr-iov1.0_12May08.pdf

- Other IOV and PCIe Training Materials:

http://www.pcisig.com/events/devcon_10/agenda/