Implementing PCI I/O Virtualization Standards

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

Copyright © 2007, PCI-SIG, All Rights Reserved

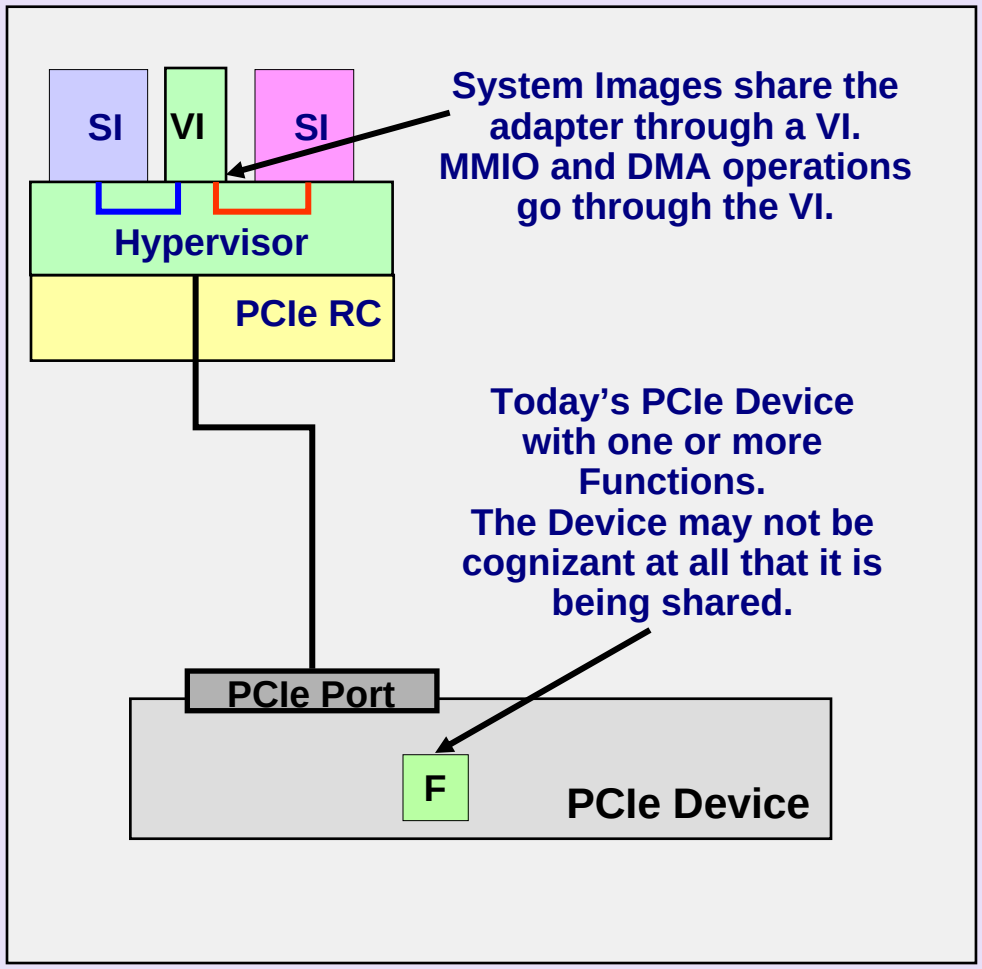
Agenda

- Virtualization Overview
  - ✓ Terminology
  - ✓ Single-Root (SR) IOV
  - ✓ SR-IOV Usage Exampes
  - ✓ Multi-Root (MR) IOV
- Specification Status

Copyright © 2007, PCI-SIG, All Rights Reserved

Today's Approach to IOV

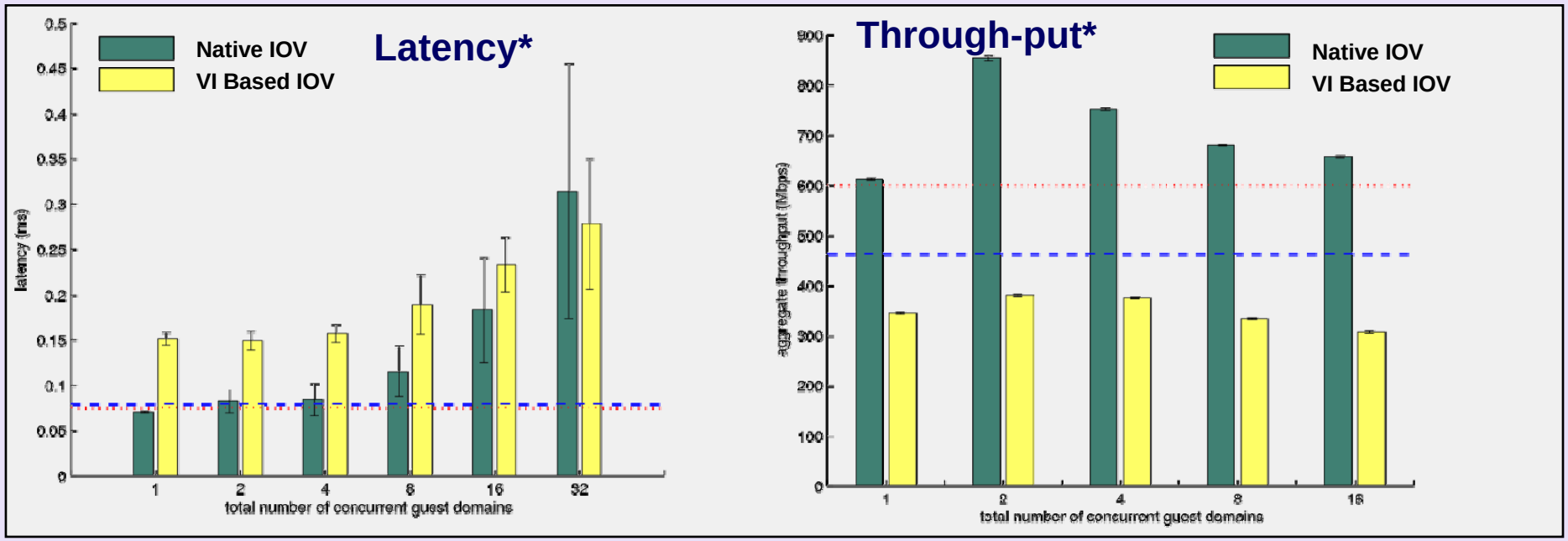
VI Based PCI Device Sharing Example



- 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.

Copyright © 2007, PCI-SIG, All Rights Reserved

Performance of Today's IOV

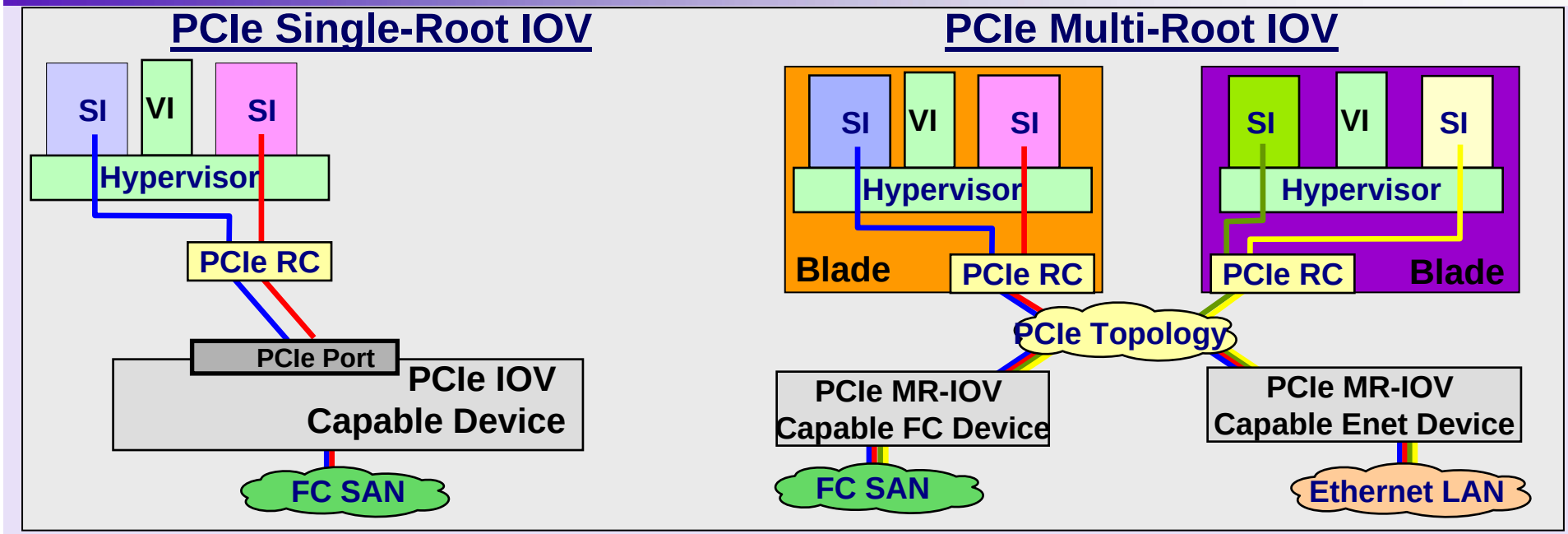


- VI based IOV adds path length on every IO operation.
- Native IOV significantly improves performance
  - ✓ For the example above Native IOV doubles throughput and reduces latency by up to half.
- Several factors are increasing the use of virtualization (e.g. more cores per socket, customer simplification requirements, …).
  - ✓ Making Native IOV even more important in the future.

***Source:** Self-Virtualized I/O: High Performance, Scalable I/O Virtualization in Multi-core Systems; R. Himanshu, I. Ganev, K. Schwan - Georgia Tech and J. Xenidis - IBM

Copyright © 2007, PCI-SIG, All Rights Reserved

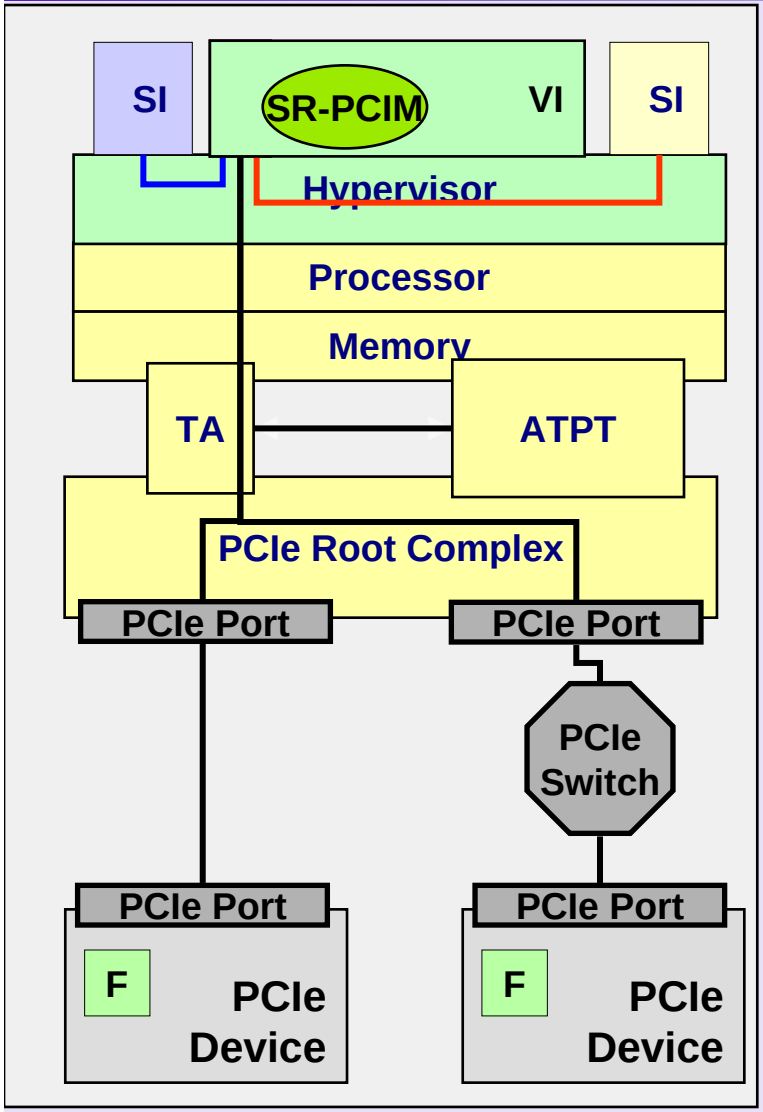
PCI-SIG IOV Overview



- PCI-SIG is standardizing 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 cover later in this deck.

Copyright © 2007, PCI-SIG, All Rights Reserved

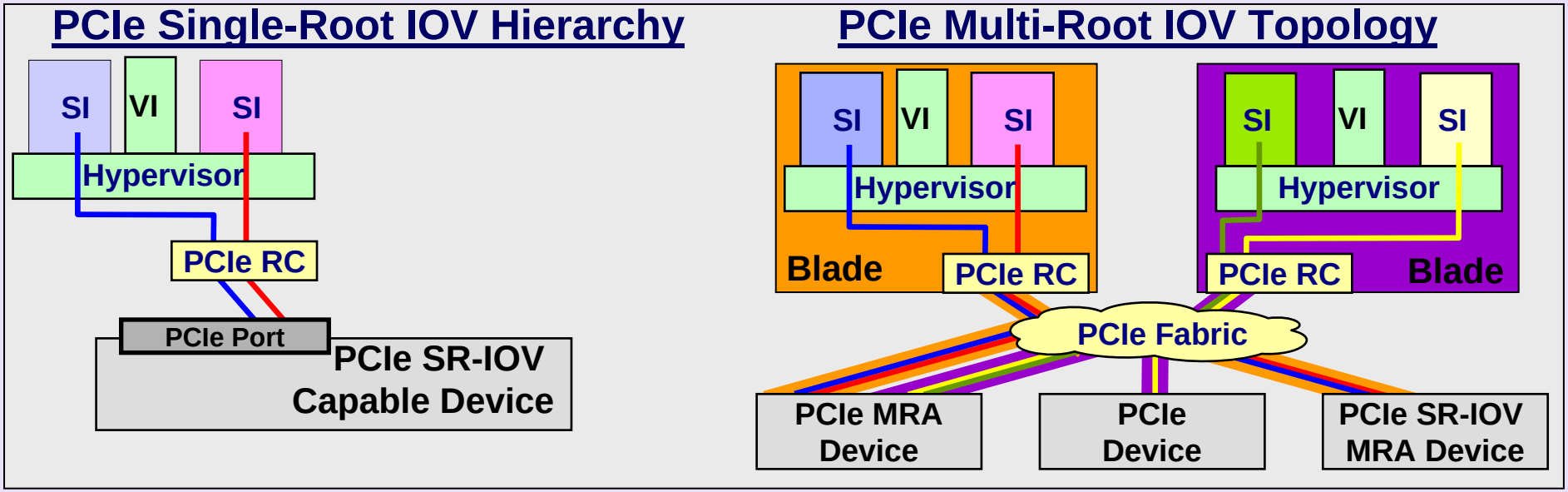
Terminology



- System Image (SI)
  - ✓ *SW, e.g. a guest OS, to which specific Functions, PFs and VFs (more later) can be assigned*
- Virtual Intermediary (VI)
  - ✓ *Performs resource allocation, isolation, management and event handling.*
- PCIM – PCI Manager
  - ✓ Configures and manages SR-IOV capability and associated error handling.
  - ✓ May be in SW and/or Firmware.
  - ✓ May be integrated into a VI.
- Translation Agent (TA)
  - ✓ Uses ATPT to translates PCI Bus Addresses into platform addresses.
- Address Translation and Protection Table (ATPT)
  - ✓ Validates access rights of incoming PCI memory transactions.
  - ✓ Translates PCI Address into platform physical addresses.

Copyright © 2007, PCI-SIG, All Rights Reserved

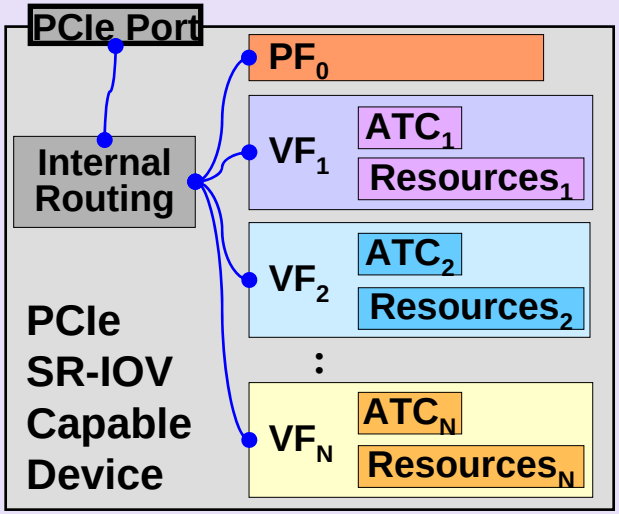
Terminology… Continued

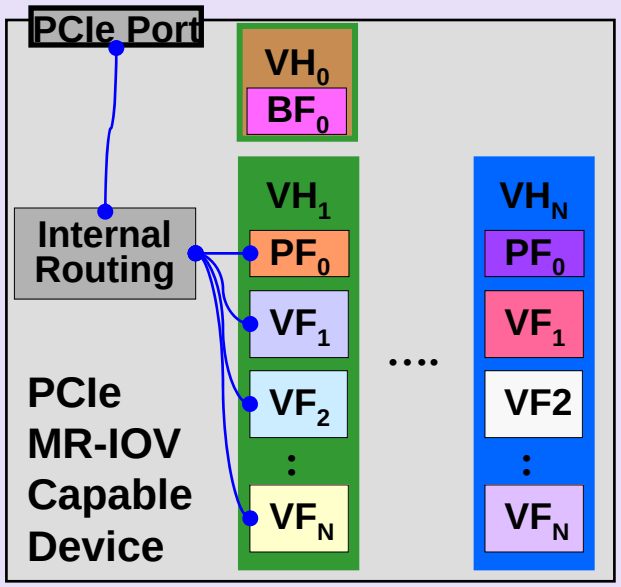


- Single-Root IOV (SR-IOV) - A function that supports the SR-IOV capability.
- Multi-Root Aware (MRA) - A PCIe component that supports the MR-IOV extensions.
- Multi-Root IOV (MR-IOV) - A function that supports the MR-IOV capability.
  - ✓ Virtual Hierarchy (VH) - A portion of an MR Topology assigned to a PCIe hierarchy. Each VH has its own PCI Memory, IO, and Configuration Space.

Copyright © 2007, PCI-SIG, All Rights Reserved

Terminology… Continued





- Address Translation Cache (ATC)
  - ✓ Cache of recent translations
- Physical Function (PF)
  - ✓ Function that supports SR-IOV.
  - ✓ Used to manage VFs.
- Virtual Function (VF)
  - ✓ Function that supports SR-IOV and shares resources with the PF it associated with.
- Base Function (BF)
  - ✓ Function that supports MR-IOV.
  - ✓ Used to manage an MRA Device's features (e.g. VHs, PFs and VFs).

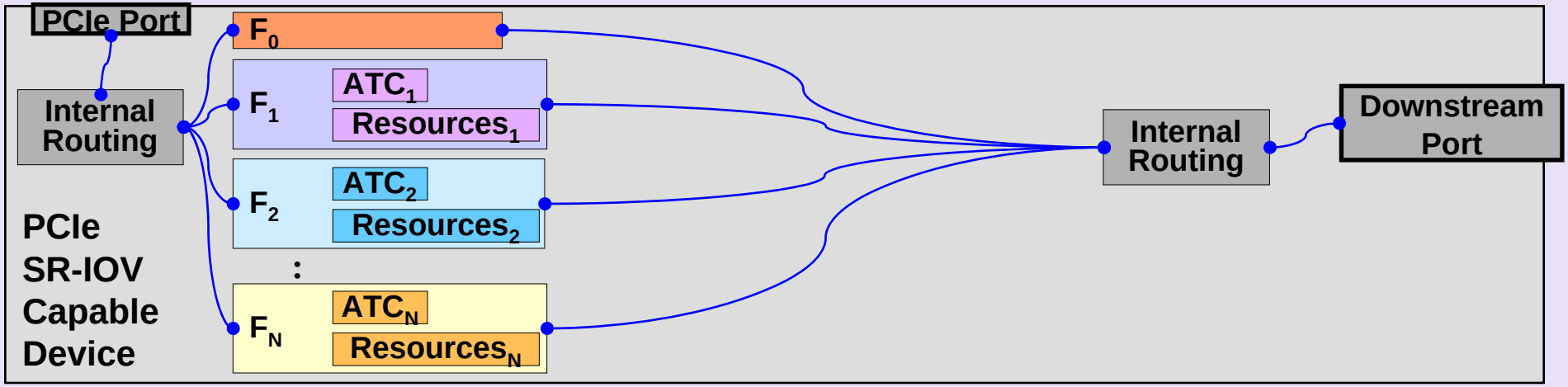
Copyright © 2007, PCI-SIG, All Rights Reserved

PCI Single-Root IOV

Overview

Copyright © 2007, PCI-SIG, All Rights Reserved

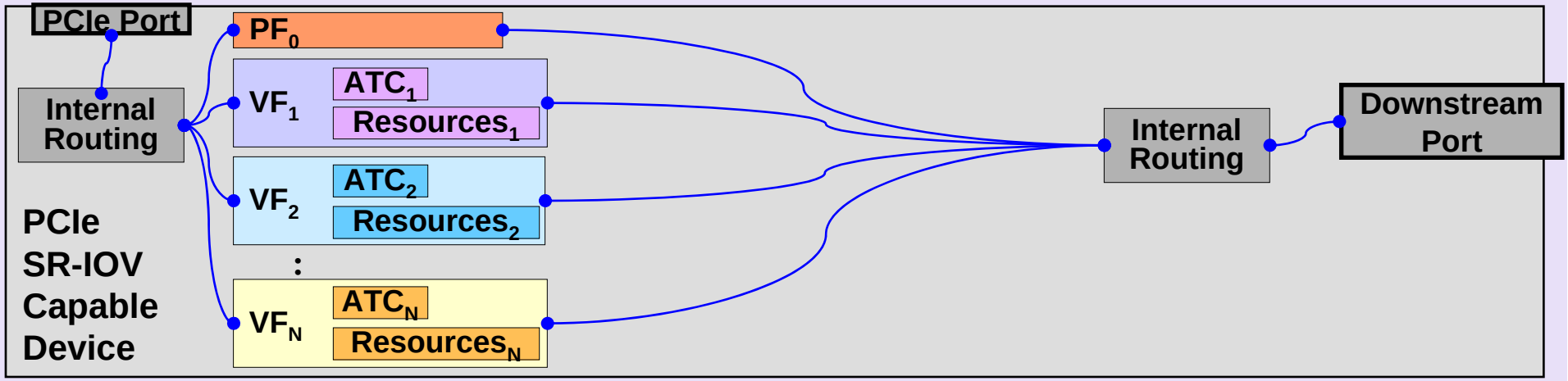
Today's PCI Device



- Function 0 is required
- Overview of Function Attributes
  - ✓ Each Function has a its own configuration and PCIe memory address space
  - ✓ Up to 8 PCI Functions with unique configuration space / BAR / etc.
    - ARI Capability enables up to 256 Functions to be supported.
  - ✓ Support INTx, MSI, MSI-X or combination of MSI and MSI-X.
  - ✓ Function dependencies through vendor specific mechanisms
  - ✓ Cannot be directly shared by SIs.
  - ✓ Vendor specific mechanisms associate Functions to "South-side" resources.

Copyright © 2007, PCI-SIG, All Rights Reserved

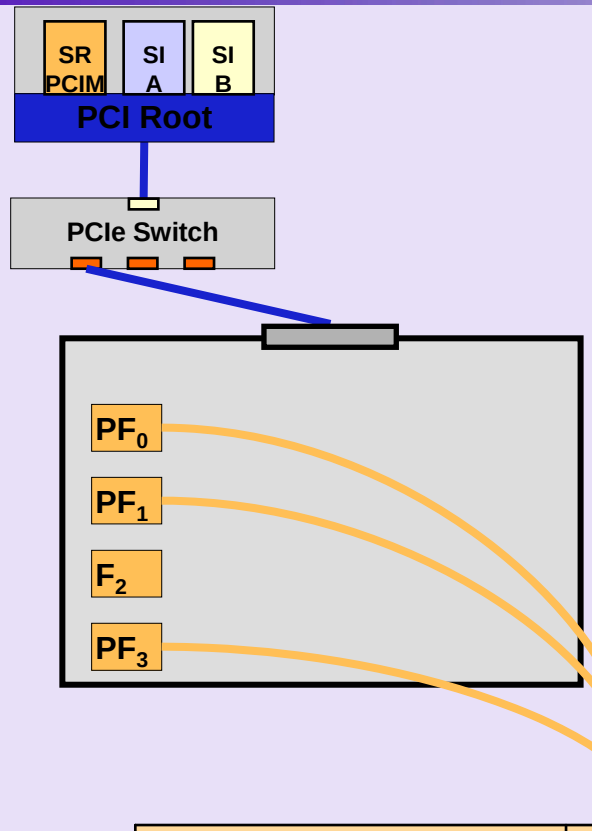
SR-IOV Device Overview



- Function 0 is required
- Overview of VF attributes
  - ✓ Each VF has a its own configuration and PCIe memory address space
    - VFs share a contiguous PCIe memory address space
  - ✓ Up to ~2^{16} Virtual Functions
    - ARI enables up to 256
    - IOV enables additional Bus Numbers to be associated
  - ✓ Function dependencies defined through standard mechanism.
  - ✓ Support MSI and MSI-X
  - ✓ Can be directly shared by SIs.
  - ✓ Vendor specific mechanisms to associate Functions to “South-side” resources.

Copyright © 2007, PCI-SIG, All Rights Reserved

SR-IOV Capability Discovery



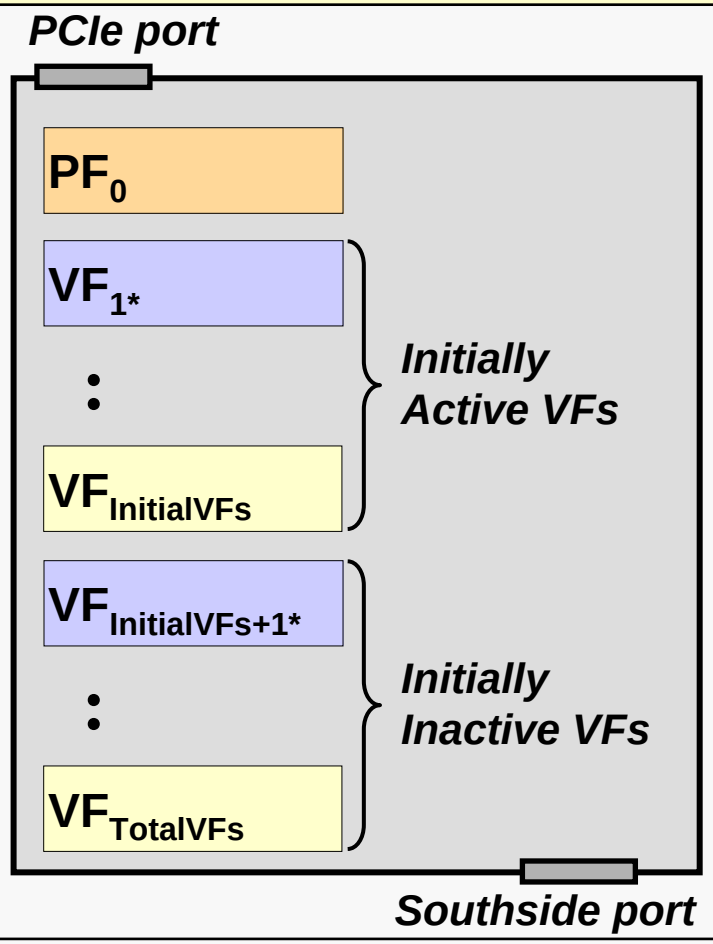
- Each PF must have an SR-IOV Extended Capability structure.
- A Device may have a mix of Functions and PFs.
- Each Function and PF consumes one Routing Identifier (RID).
- For a Device that:
  - ✓ isn't ARI capable, Functions and PFs must be in the first 8 functions;
  - ✓ is ARI capable, number of Functions and PFs supported is as defined in the base specification.

31 20	19 16	15 0	Offset
Next Capability Offset	Cap. Vrsn.	PCIe Extended Capability ID	00h
SR IOV Capabilities			04h
:			:

Copyright © 2007, PCI-SIG, All Rights Reserved

Terminology

SR Topology



Terms

Active VF is a VF that must respond to PCIe Configuration transactions and may respond to Memory Transactions that target that VF. Additionally, an Active VF may issue PCIe transactions if Bus Master Enable is Set.

In other words, an Active VF is associated with underlying south side context and can be used to perform mission work to that context.

Inactive VF is a VF that must respond with Unrecognized Request (UR) to any transaction that targets that VF.

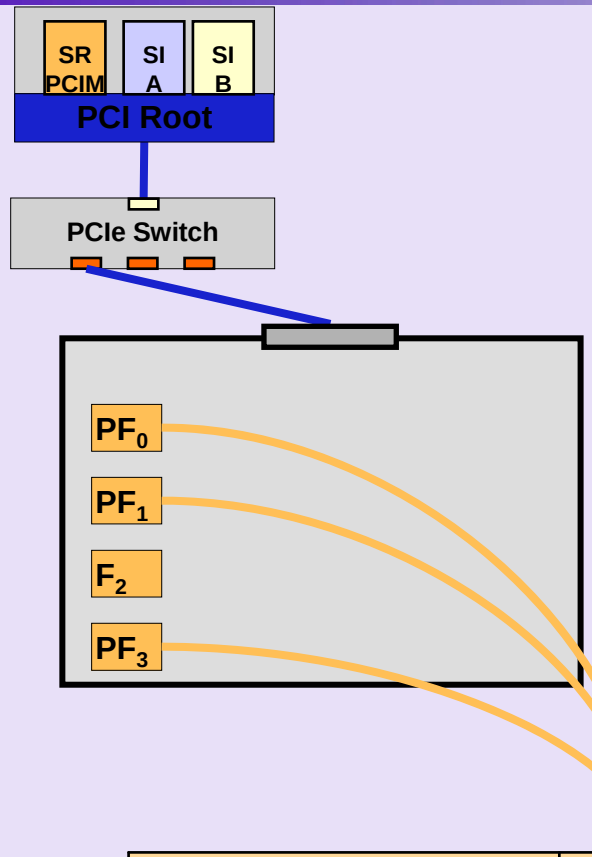
*In other words, an Inactive VF is **not** associated with underlying south side context and **cannot** be used to perform mission work.*

This deck will describe how, during migration, a specific VF's state can move between these 2 states.

**Note: The VF numbering depicted on this slide is with respect to the PF and is not intended to reflect RID numbering. For a Device with multiple PFs, the RID associated with a VF may not be sequential (more on this later).*

Copyright © 2007, PCI-SIG, All Rights Reserved

VF Discovery - Part 1



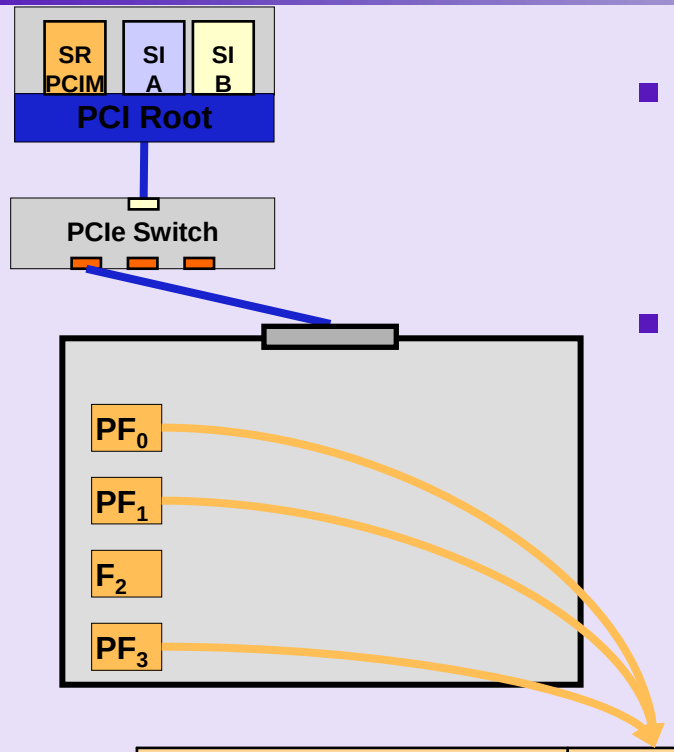
- The ***TotalVFs*** field is used to discover the number of ***Active VFs*** that a PF ***could have***.
- The ***InitialVFs*** field is used to discover the number of ***Active VFs*** that a PF ***initially has***.
- Note for a Device that **isn't** MR capable*:
  1 ≤ InitialVFs = TotalVFs

31 20	19 16	15 0	Offset
:			:
TotalVFs (RO)		InitialVFs (RO)	0Ch
:			:

**Note: More later on MR capable Devices.*

Copyright © 2007, PCI-SIG, All Rights Reserved

BAR Discovery

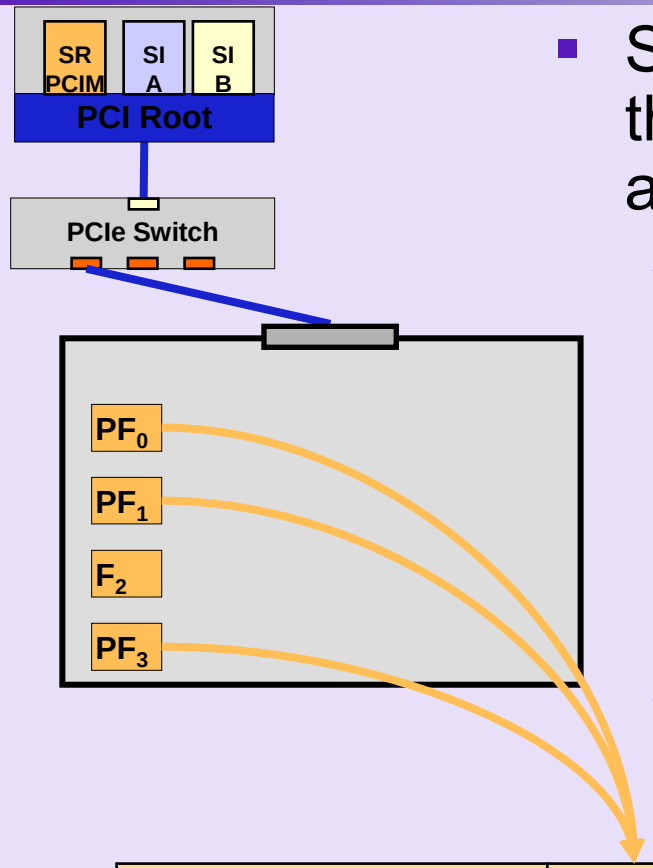


- Each PF has its own independent set of BARs in its standard configuration space and an MSE bit for those BARs.
- The VFs share a BAR set *(more later)* and have an MSE bit that controls the memory space of **all** the VFs.
  - ✓ The BAR set that is shared by all the VFs resides in the PF's SR-IOV capabilities

31 20	19 16	15 0	Offset
:			:
VF BAR0 (RW)			24h
:			:
VF BAR5 (RW)			38h
:			:

Copyright © 2007, PCI-SIG, All Rights Reserved

Discovering Supported Page Size

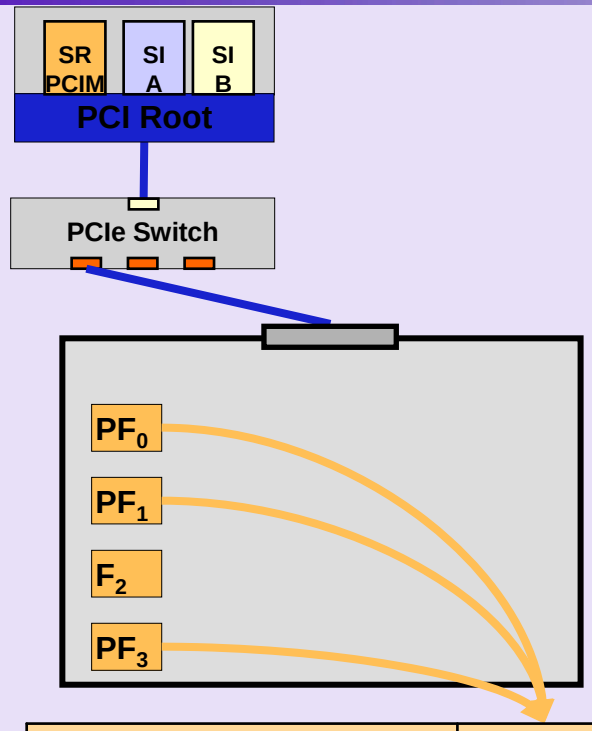


- Supported Page Sizes is used to discover the page sizes supported by the VFs associated with the PF.
  - ✓ When this field is read, the PF must return the page sizes it can support.
    - If bit n is Set in this field, the PF supports a page size of 2^{n+12} (e.g. if bit 0 is Set, the PF supports 4 KB pages).
    - The PF must support 4 KB, 8 KB, 64 KB, 256 KB, 1 MB and 4 MB pages.
  - ✓ This field will be used during the IOV configuration phase to align VF BAR apertures on system page boundaries (more later).

31 20	19 16	15 0	Offset
:			:
Supported Page Sizes (RO)			1Ch
:			:

Copyright © 2007, PCI-SIG, All Rights Reserved

Configuring VFs



- A PF's VFs are enabled by Setting "VF Enable".
  - ✓ NumVFs defines the number of VFs enabled and must be 1 ≤ NumVFs ≤ TotalVFs
  - ✓ When VF's are enabled, the PCIe Device must associate NumVFs worth of VFs with the PF.
- If VF Migration Capable is Set, the Device is operating under a migration capable MR-PCIM.
  - ✓ In this case, InitialVFs may be less than TotalVFs to allow for migration (more later).
- If VF Migration Enable is Set by SR-PCIM, then MR-PCIM is allowed to perform migration of VFs associated with the PF (more later).

31 20	19 16	15 0	Offset
:			:
SR IOV Capabilities (RO)			04h
SR IOV Status		SR IOV Control (RW)	04h
TotalVFs (RO)		InitialVFs (RO)	0Ch
Reserved	Func Dep Link	NumVFs	10h
:			:

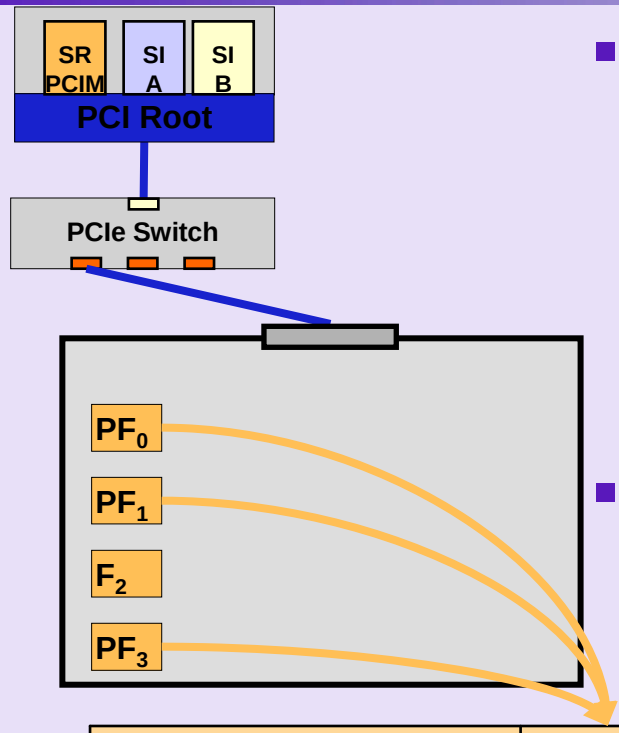
VF Migration Capable

VF Migration Enable

VF Enable

Copyright © 2007, PCI-SIG, All Rights Reserved

Configuring the VF's BARs

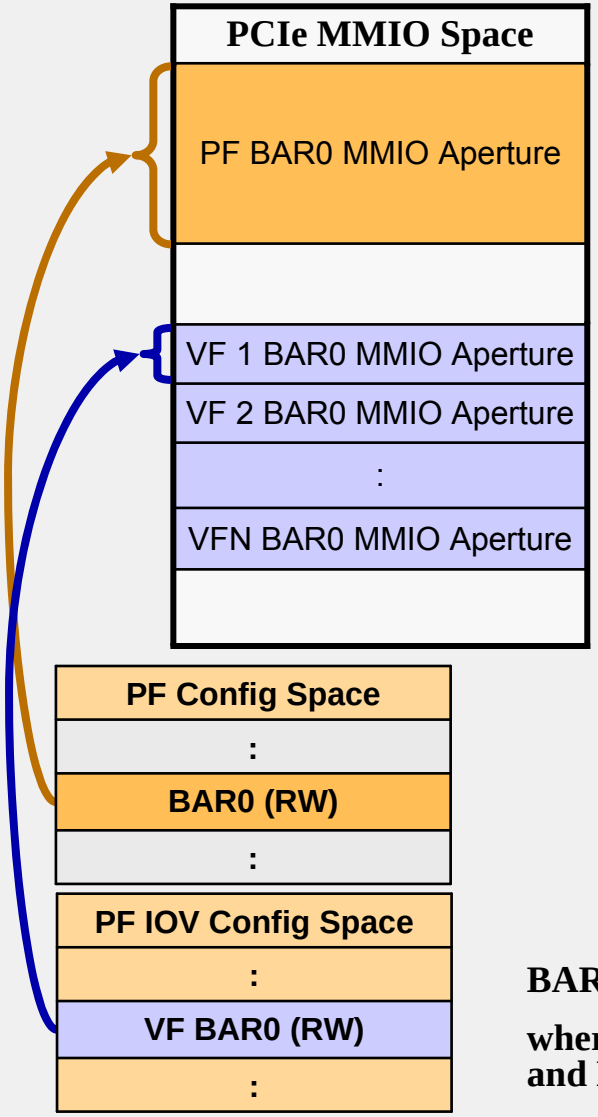


- System Page Size defines the page size the system will use.
  - ✓ Value of System Page Size must be one of the Supported Page Sizes.
  - ✓ System Page Size is used by the PF to align the MMIO aperture defined by each BAR to a system page boundary.
- VF BARs behave as in PCI 3.0 Spec's PCI BARs, except that a VF BAR describes the aperture for multiple VFs (see next page).

31 20	19 16	15 0	Offset
:			:
System Page Sizes (RW)			20h
VF BAR0 (RW)			24h
:			:
VF BAR5 (RW)			38h
:			:

Copyright © 2007, PCI-SIG, All Rights Reserved

PF and VF BAR Semantics



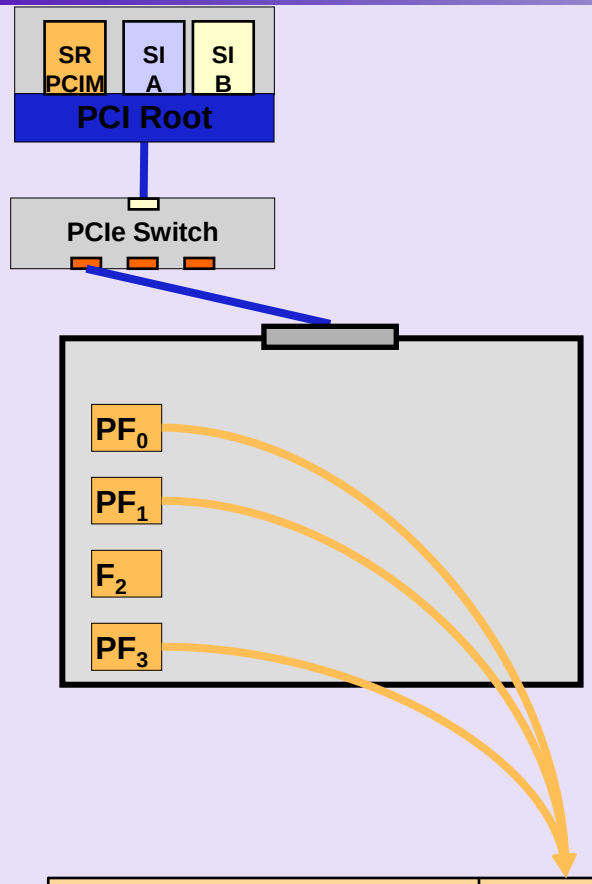
- The memory aperture required for each VF BAR can be determined by writing all "1"s and then reading the VF BAR.
  - ✓ Behaves as in the base PCI Spec.
- The address written into each VF BAR is used by the Device to set the starting address for that BAR on the first VF.
- The differences between VF BARs and PCI 3.0 Spec BARs are:
  - ✓ For each VF BAR, the memory space associated with the 2nd and higher VFs is derived from the starting address of the first VF and the memory space aperture.
  - ✓ The VF BAR's MMIO space is not enabled until both VF Enable and VF MSE have been set.

BAR1 VF N SA = BAR1 VF 1 SA + N x (BAR1 VF MA) - 1

where BAR1 VF 1 SA is the address written into VF BAR1 and MA is the memory aperture of VF BAR1.

Copyright © 2007, PCI-SIG, All Rights Reserved

VF Discovery - Part 2



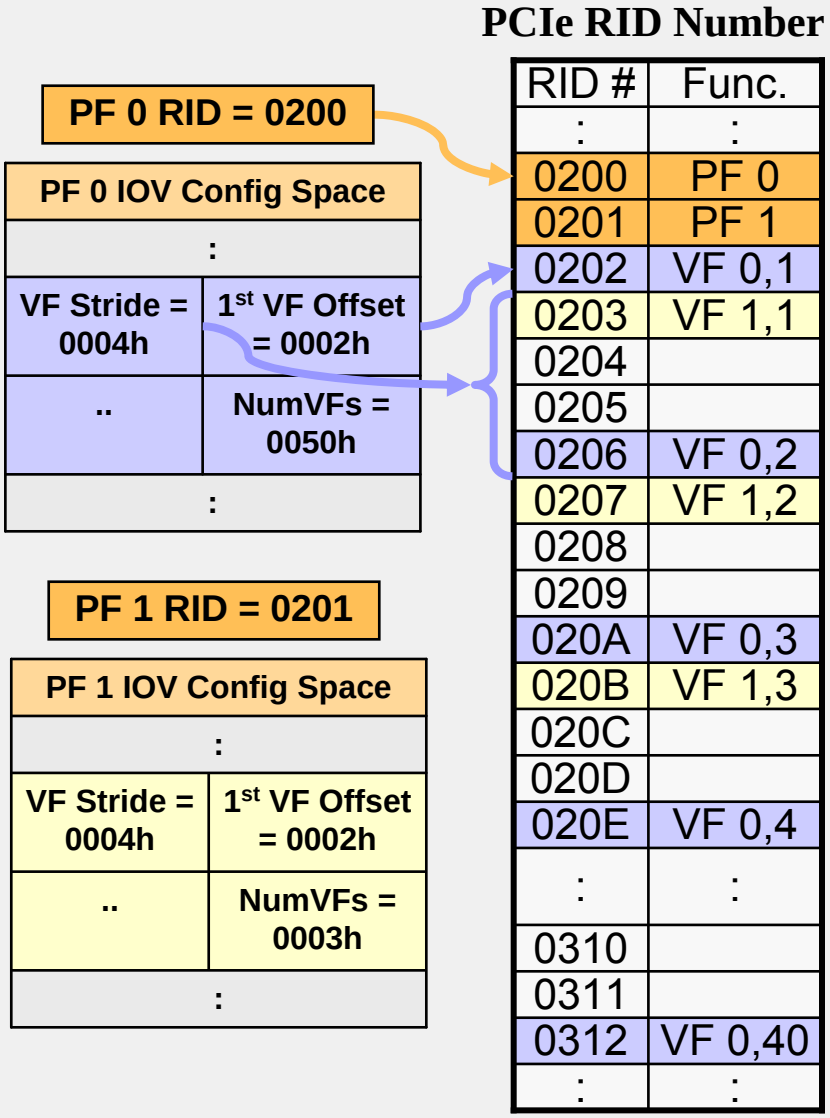
- First VF Offset defines the Routing ID offset of the first VF associated with the PF.
- VF Stride defines the Routing ID offset from one VF to the next one for all VFs associated with the PF.
- The RID of VF N can be determined by:

  VF N = [PF RID + VF Offset + (N-1) * VF Stride] Modulo 2^{16}
  where all arithmetic used is 16 bit unsigned dropping all carries.
- The values in NumVFs and VF ARI Enable affect the values for First VF Offset and VF Stride.
  - ✓ *After setting NumVFs and VF ARI Enable, SR-PCIM can read these First VF Offset and VF Stride to determine how many busses will be consumed by the PF's VFs.*

31 20	19 16	15 0	Offset
:			:
VF Stride (RO)		First VF Offset (RO)	14h
:			:

Copyright © 2007, PCI-SIG, All Rights Reserved

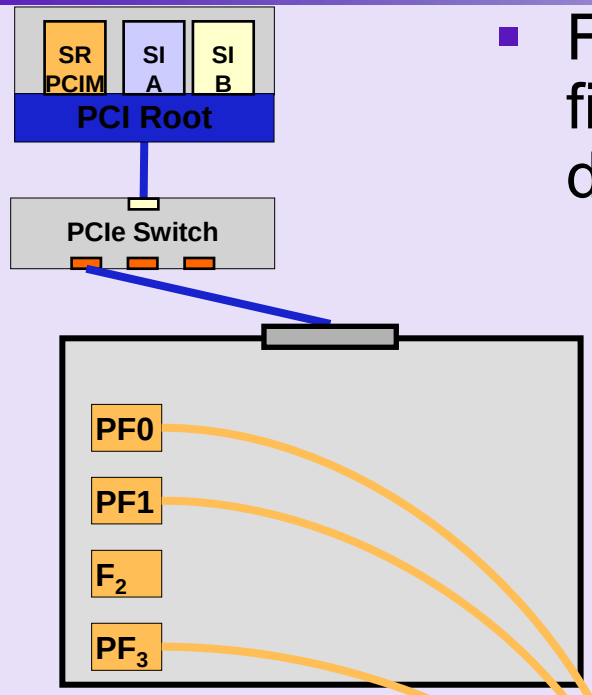
PF and VF RID Semantics



- PF and VF RIDs must not overlap given any valid NumVFs setting across all PFs of a Device.
- As in base, an SR-IOV Device captures the Bus # from any Type 0 config request.
  - ✓ VFs may reside on different bus number(s) than the associated PF.
- If the switch above the Device:
  - ✓ supports the ARI Forwarding bit, RIDs on 1st Bus are interpreted as 8 bit Bus # and 8 bit Device #.
  - ✓ doesn't support ARI, RIDs on 1st bus are interpreted as BDF#.
- Bus # can be used to assign VFs, but PFs must be on 1st Bus assigned to Device.

Copyright © 2007, PCI-SIG, All Rights Reserved

Function Dependencies



- Function Dependency Link is an optional field used to describe vendor specific dependencies between sets of functions.
  - ✓ If a PF is independent from other PFs of a Device, the Function Dependency Link field must contain the PF's Function Number.
  - ✓ If a PF is dependent on other PFs of a Device, the Function Dependency Link field must contain the Function Number of the next PF in the same Function Dependency List (more on next page).

31	16	15	0	Offset
:				:
Reserved	Func Dep Link	NumVFs		10h
:				:

Copyright © 2007, PCI-SIG, All Rights Reserved

Function Dependency Link Example

Example	PF 0	PF 1	PF 3
Function Dependency Link	1	0	2
NumVFs	4	4	6
First VF Offset	4	4	4
VF Stride	3	3	3

Each of these is a pair of dependent VFs, e.g.:
PF 1 is dependent on PF 0
VF 1,0 is dependent on VF 0,0
where for n,m: n is the PF's number and
m is the VF's number.

Function #	Description	Independent
0	PF 0	No
1	PF 1	No
2	PF 2	Yes
3	Function not present	
4	VF 0,1 (aka PF 0 VF 1)	No
5	VF 1,1 (aka PF 1 VF 1)	No
6	VF 2,1 (aka PF 2 VF 1)	Yes
7	VF 0,2	No
8	VF 1,2	No
9	VF 2,2	Yes
10	VF 0,3	No
11	VF 1,3	No
12	VF 2,3	Yes
13	VF 0,4	No
14	VF 1,4	No
15	VF 2,4	Yes
16 to 17	Functions not present	
18	VF 2,5	Yes
19 to 20	Functions not present	
21	VF 2,6	Yes
22 to 255	Functions not present	

Copyright © 2007, PCI-SIG, All Rights Reserved

Reset Mechanisms

- Three reset mechanisms are supported:
  - ✓ Conventional Reset - Resets all PF and VF state.
  - ✓ FLR that targets a VF - Resets a single VF.
  - ✓ FLR that targets a PF - Resets a PF and its associated VFs.

Copyright © 2007, PCI-SIG, All Rights Reserved

Conventional Reset

- A Fundamental or Hot Reset to an SR-IOV Device shall cause all Functions, PFs, and VFs context to be reset to their original, power-on state.
- If a PF has its VFs enabled and a Fundamental or Hot Reset is issued to the Device, the Device must reset all PF and VF state:
  - ✓ The PF must disable its SR-IOV capabilities and reverts back to being a PCI Function.
  - ✓ Settable SR-IOV capabilities are reset to default values.
  - ✓ MSE and BME are both off.
  - ✓ All BAR values used by the PF and VFs are indeterminate.
  - ✓ All interrupts are disabled.

Copyright © 2007, PCI-SIG, All Rights Reserved

FLRs to VFs and PFs

- An FLR that targets a VF must be supported.
  - ✓ Software may use FLR to reset a VF.
  - ✓ An FLR that targets a VF must:
    - Not affect the VFs existence (e.g. it still consumes a RID)
    - Not affect any address mapping assigned to it. That is, the VF's BAR registers and MSE are unaffected by FLR.
- An FLR that targets a PF must be supported.
  - ✓ Software may use an FLR to reset a PF.
  - ✓ An FLR that targets a PF must:
    - Reset the PF's SR-IOV Extended Capabilities.
    - The VFs are no longer allocated or enabled.

Copyright © 2007, PCI-SIG, All Rights Reserved

PCI SR-IOV

Usage Examples

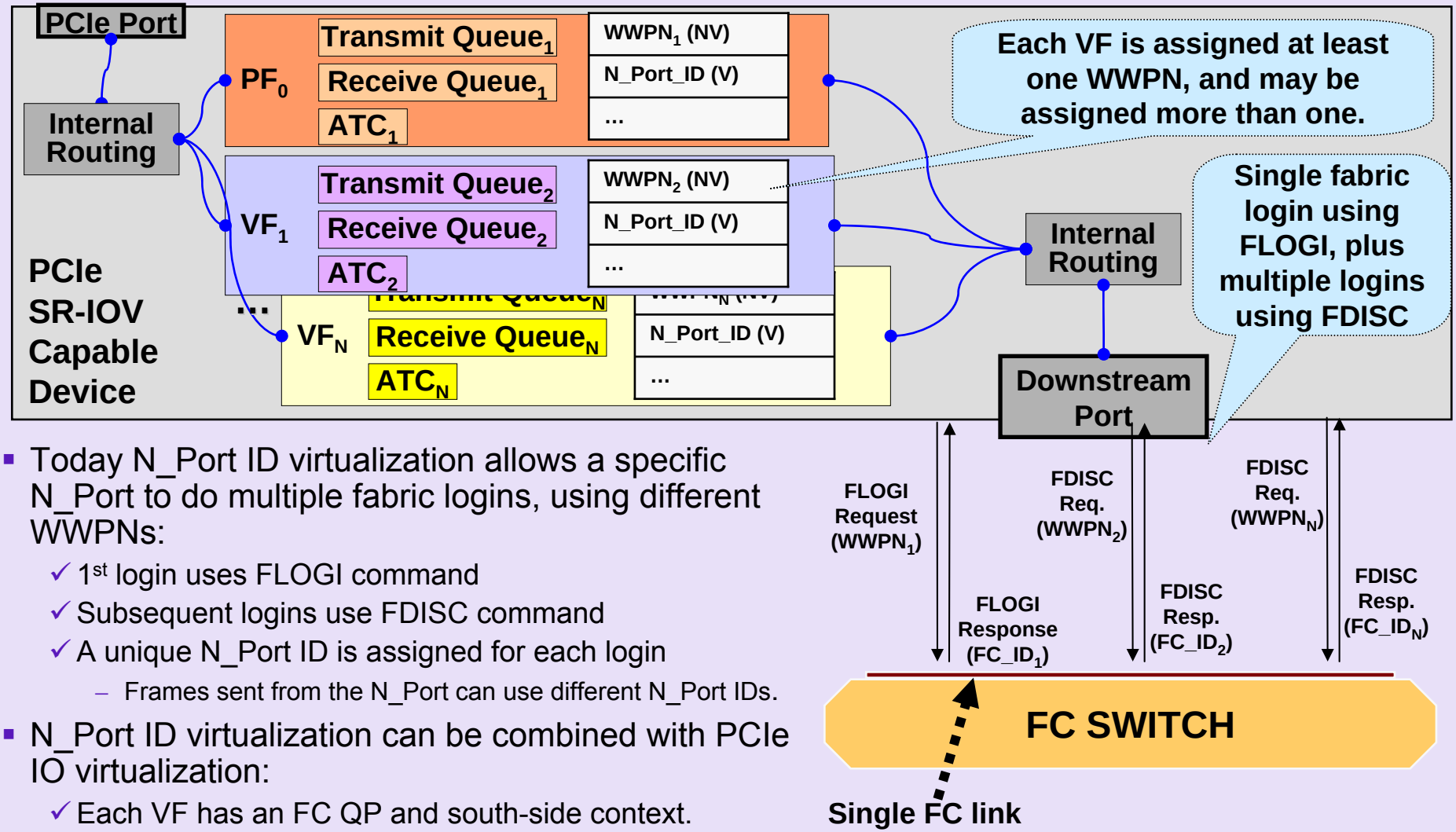
Copyright © 2007, PCI-SIG, All Rights Reserved

SR-IOV Usage Examples

- South side logic virtualization examples:
  1. An FC adapter.
  2. An Ethernet adapter.
  3. An iWARP RDMA enabled NIC.
  4. An InfiniBand Host Channel Adapter.

Copyright © 2007, PCI-SIG, All Rights Reserved

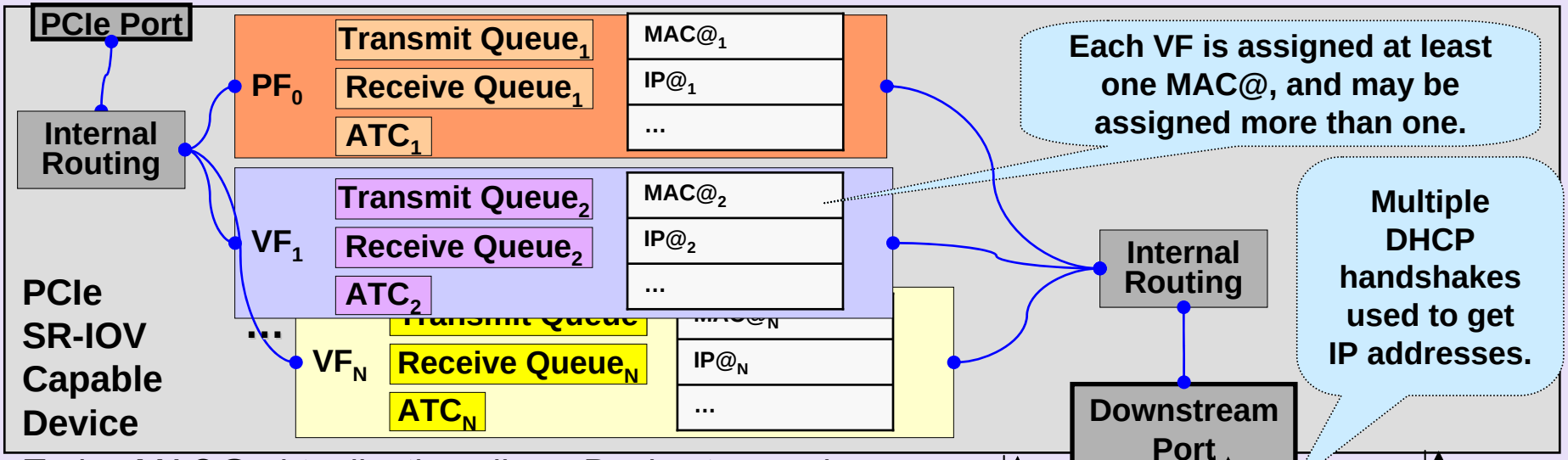
Example 1 - FC HBA



- Today N_Port ID virtualization allows a specific N_Port to do multiple fabric logins, using different WWPNs:
  - ✓ 1st login uses FLOGI command
  - ✓ Subsequent logins use FDISC command
  - ✓ A unique N_Port ID is assigned for each login
    - Frames sent from the N_Port can use different N_Port IDs.
- N_Port ID virtualization can be combined with PCIe IO virtualization:
  - ✓ Each VF has an FC QP and south-side context.
  - ✓ SI can communicate directly with VF's FC QP.

Copyright © 2007, PCI-SIG, All Rights Reserved

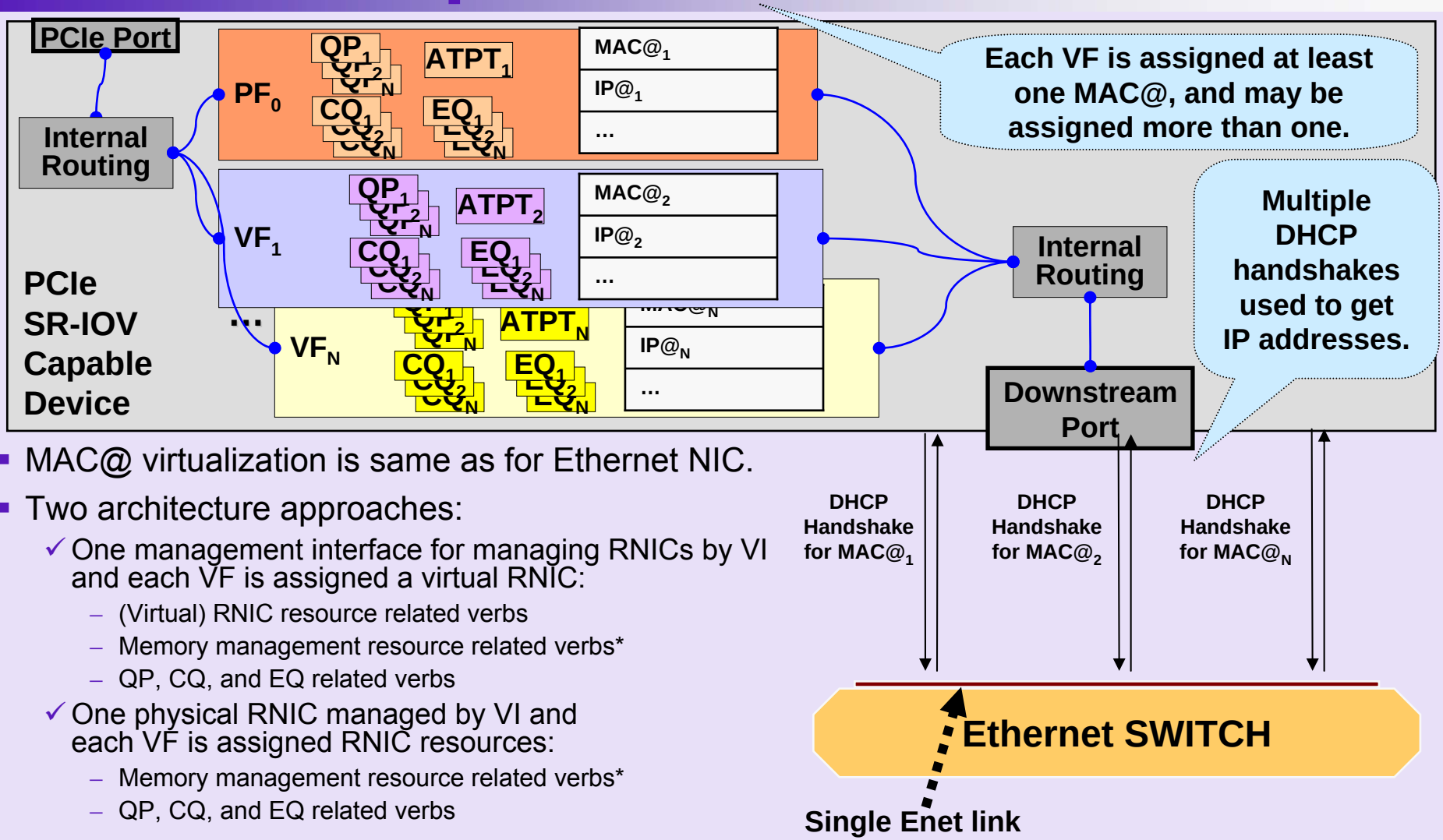
Example 2 - Ethernet NIC



- Today MAC@ virtualization allows Device to acquire an IP address per MAC@. For example, using DHCP each "virtual" MAC@:
  - ✓ Issues a DHCP Discover to initiate DHCP.
  - ✓ After the DHCP Server responds with DHCP Offer, a DHCP Request is sent to acquire the IP Address.
  - ✓ The DHCP Server then responds with the IP address associated with the MAC@.
- MAC@ virtualization can be combined with PCIe IO virtualization:
  - ✓ Each VF has an Enet QP and south-side context.
  - ✓ SI can communicate directly with VF's Enet QP.

Copyright © 2007, PCI-SIG, All Rights Reserved

Example 3 - iWARP RNIC

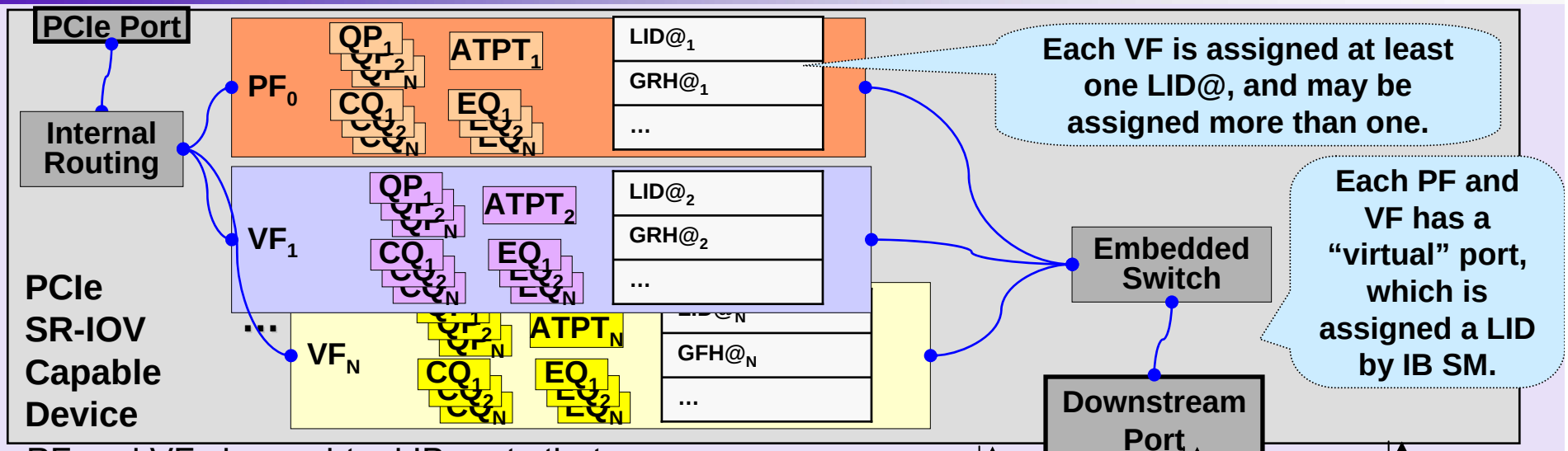


- MAC@ virtualization is same as for Ethernet NIC.
- Two architecture approaches:
  - ✓ One management interface for managing RNICs by VI and each VF is assigned a virtual RNIC:
    - (Virtual) RNIC resource related verbs
    - Memory management resource related verbs*
    - QP, CQ, and EQ related verbs
  - ✓ One physical RNIC managed by VI and each VF is assigned RNIC resources:
    - Memory management resource related verbs*
    - QP, CQ, and EQ related verbs

*Note: depending on Device's ATC implementation, memory management can be either a PF or VF function.

Copyright © 2007, PCI-SIG, All Rights Reserved

Example 4 - IB HCA



- PF and VFs have virtual IB ports that appear as physical ports on the external fabric.
- Two architecture approaches:
  - ✓ One management interface for managing HCAs by VI and each VF is assigned a virtual HCA:
    - (Virtual) HCA resource related verbs
    - Memory management resource related verbs*
    - QP, CQ, and EQ related verbs
  - ✓ One physical HCA managed by VI and each VF is assigned a HCA resources:
    - Memory management resource related verbs*
    - QP (with VF's LID), CQ, and EQ related verbs

*Note: depending on Device's ATC implementation, memory management can be either a PF or VF function.

Copyright © 2007, PCI-SIG, All Rights Reserved

PCI Multi-Root IOV

Overview

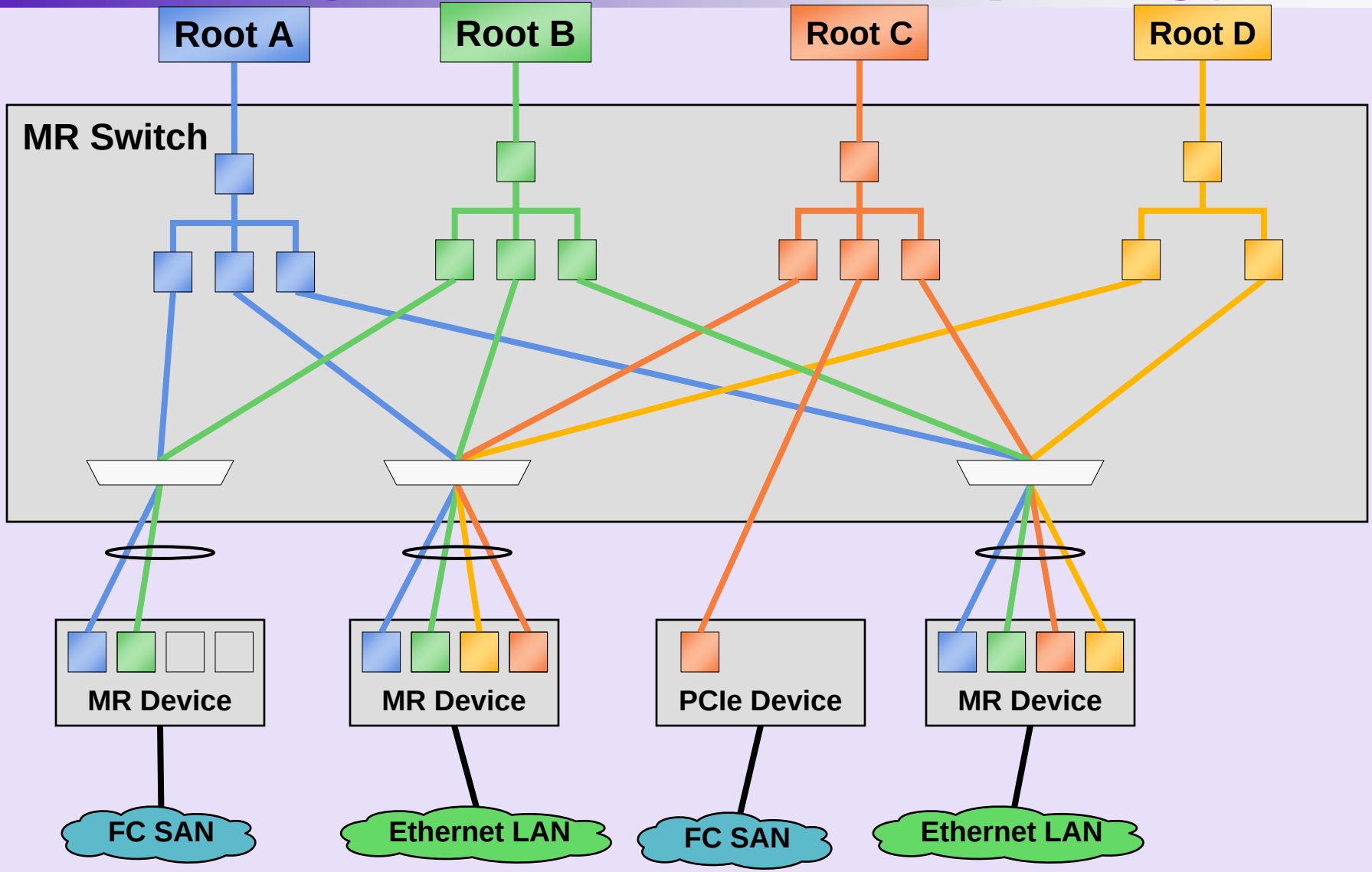
Copyright © 2007, PCI-SIG, All Rights Reserved

Today's PCIe Topology

- Strict Tree
  - ✓ Root Ports Connect to Switches
  - ✓ Switches Connect to Devices
  - ✓ Roots Ports can also connect to Devices
  - ✓ Switches can also connect to more Switches
- Single Software Management Entity
  - ✓ Runs above the Root
- Implicit Message Routing
  - ✓ Route to Root
  - ✓ Broadcast from Root

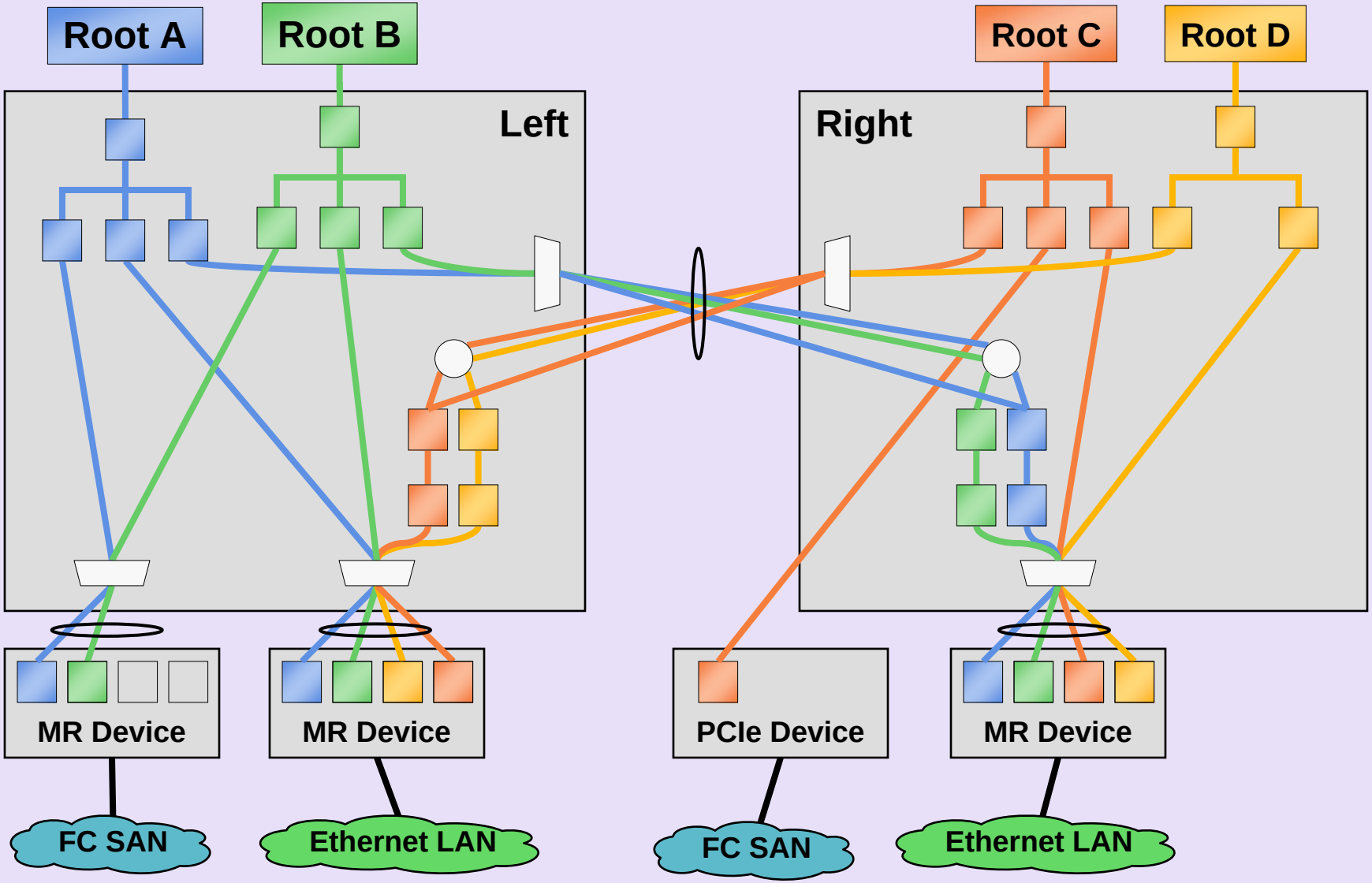
Copyright © 2007, PCI-SIG, All Rights Reserved

Single Switch MR Topology



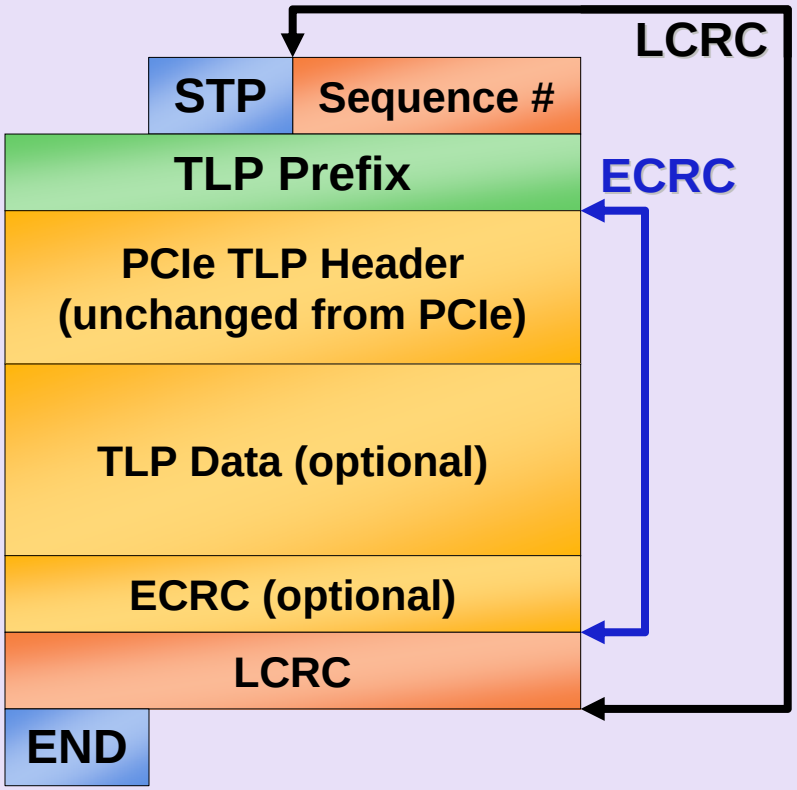
Copyright © 2007, PCI-SIG, All Rights Reserved

Two Switch MR Topology



Copyright © 2007, PCI-SIG, All Rights Reserved

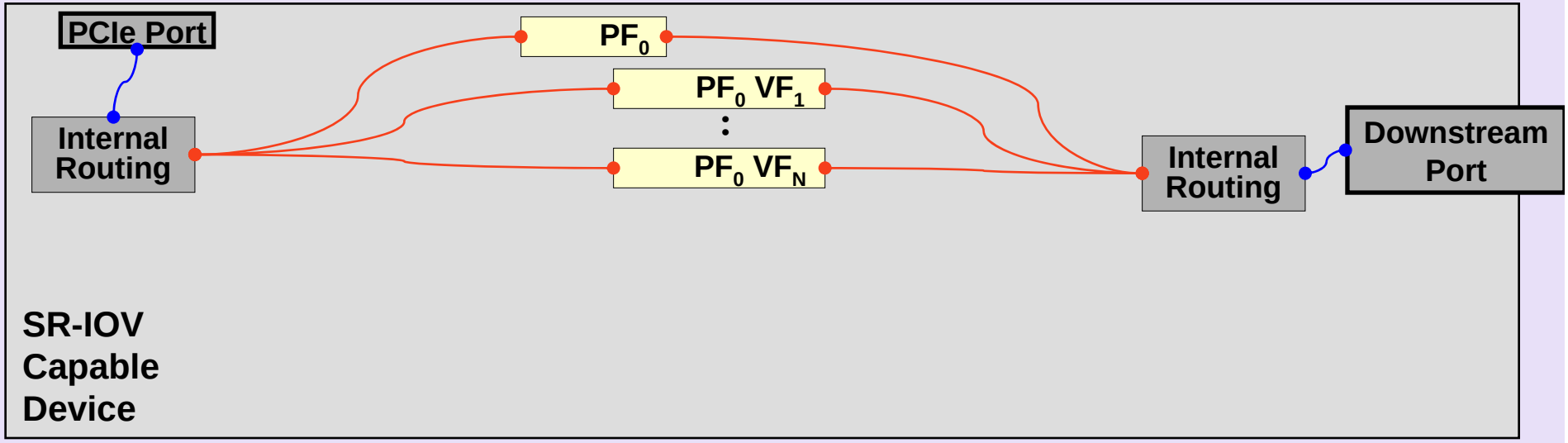
TLP Labeling



- TLP Prefix on MR Links
  - ✓ After Seq #, before TLP Hdr
  - ✓ Sent / Resent with TLP
- TLP Prefix Contains:
  - ✓ VH Number (link local)
  - ✓ VL Number (flow control)
  - ✓ Global Key (error checking)
- Added at MR Ingress
  - ✓ First MR Component seen
- Dropped at MR Egress
  - ✓ Last MR Component seen

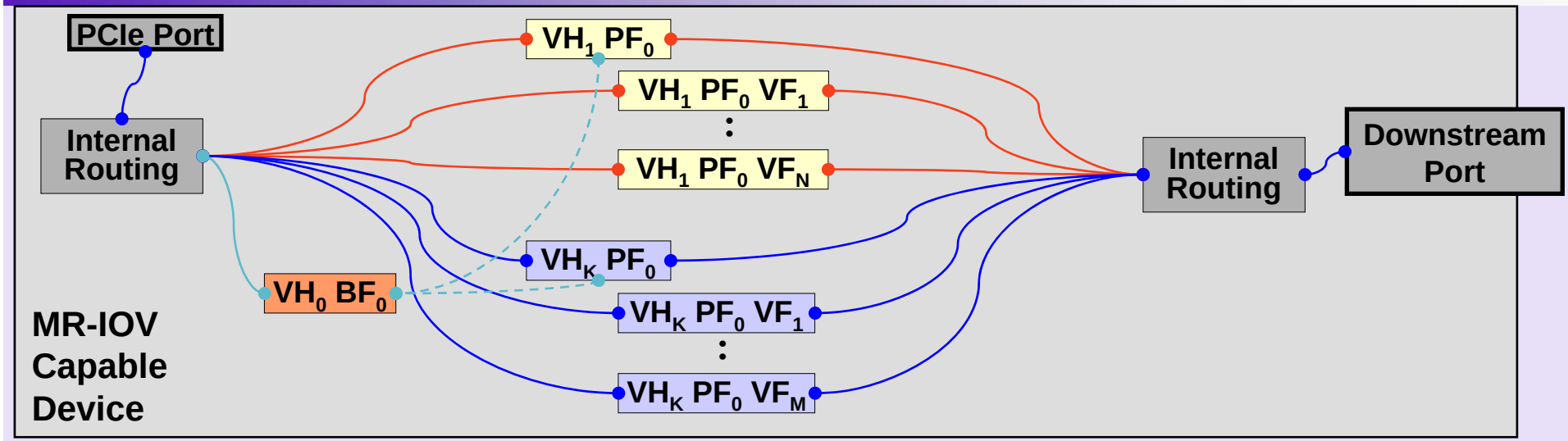
Copyright © 2007, PCI-SIG, All Rights Reserved

SR-IOV Device Overview



Copyright © 2007, PCI-SIG, All Rights Reserved

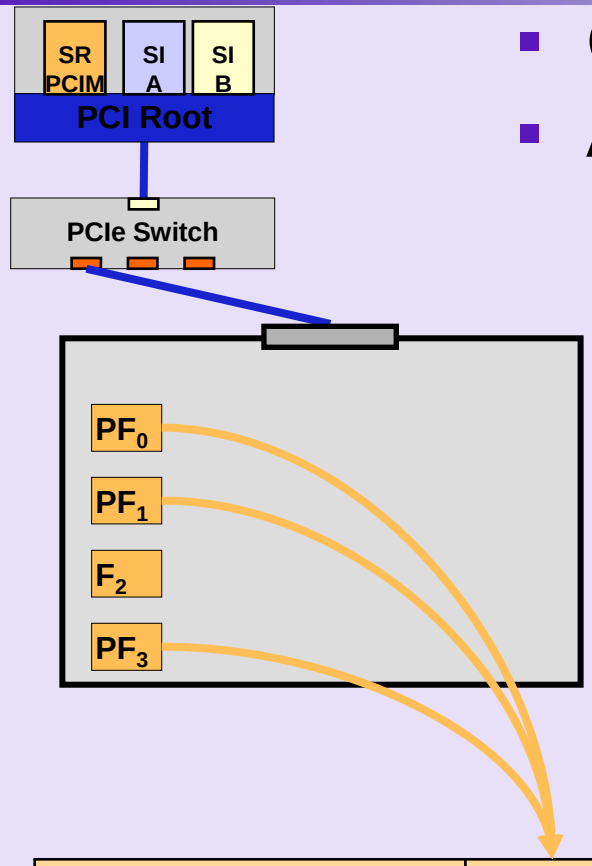
MR-IOV Device Overview



- Each Virtual Hierarchy (VH) has a full PCIe address space
  - ✓ Configuration, Memory and I/O Space
- Up to 2^8 Virtual Hierarchies
- VH_0 is used to Manage MR-IOV features of the Device
  - ✓ BF in VH_0 manages associated PFs and VFs
  - ✓ Other Functions optional in VH_0
- VH_1 to VH_{max} have same PFs at the same Function #s
  - ✓ Each PF in each VH has it's own values for InitialVFs, TotalVFs, NumVFs, VF Stride and VF Offset.

Copyright © 2007, PCI-SIG, All Rights Reserved

IOV Re-initialization and Reallocation



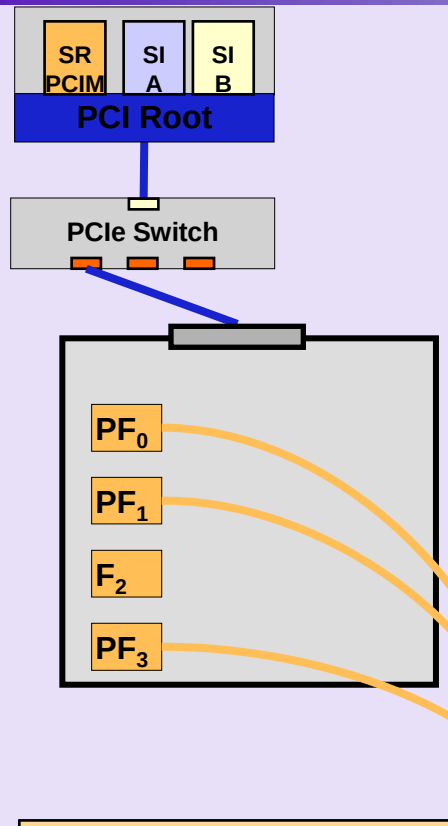
- Clearing “VF Enable” disables a PF’s VFs.
- After a PF’s VFs are disabled:
  - ✓ The VFs must no longer:
    - Issue PCIe transactions,
    - Respond to Configuration Space accesses,
    - Respond to Memory Space accesses,
    - Retain any context, or
    - Log any new VF errors.
  - ✓ Any errors already logged via PF error reporting registers, remain logged.

31 20	19 16	15 0	Offset
:			:
SR IOV Status		SR IOV Control	04h
:			:

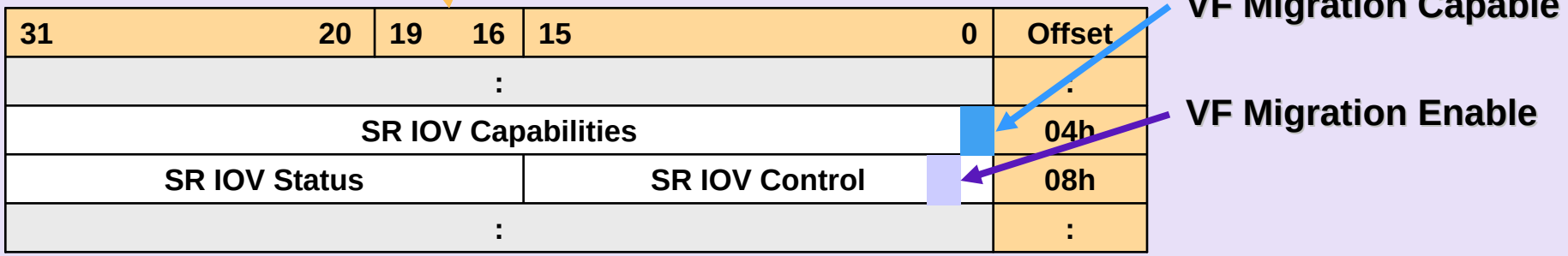
VF Enable

Copyright © 2007, PCI-SIG, All Rights Reserved

VF Migration



- VF Migration does not occur in SR systems.
- VF Migration is an optional Device feature defined to support the migration of VFs between Virtual Hierarchies.
  - ✓ If the Device is in an MR topology and migration is enabled, the “VF Migration Capable” bit must be set in the SR-IOV Capabilities Register.
- VF Migration must only occur if it has been enabled through the PF’s SR-IOV capabilities by SR software.
  - ✓ If migration operations are disabled in the PF’s SR-IOV capabilities, VFs are always Active, and the VF State array is undefined (more on this later).



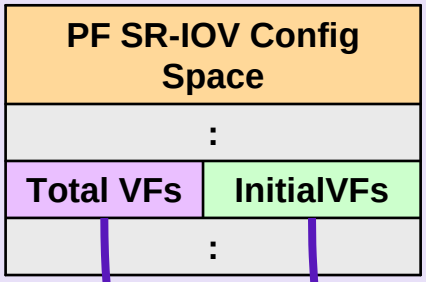
Copyright © 2007, PCI-SIG, All Rights Reserved

VF Migration Enabled → Software Requirements

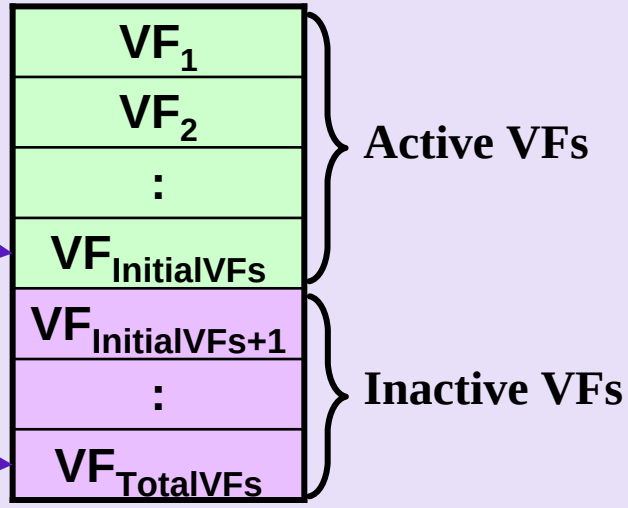
- SR software (e.g. SR-PCIM) must:
  - ✓ Determine if a VF is *Active, Dormant* or *Inactive*.
    - An active VF that is must be available for use by SR.
    - A Dormant VF can be configured by SR but will not issue transactions
    - Inactive VFs must not be available for use by SR.
  - ✓ Participate in *Migrate In* operations.
    - A Migrate In operation is used by MR-PCIM to offer a VF to SR software.
    - SR software may Accept a Migrate In operation making the VF Active.
      - Until accepted, VF is not Active.
  - ✓ Participate in *Migrate Out* operations.
    - A Migrate Out operation is used to migrate a VF from Active to Inactive.
    - This supports the graceful removal of a VF from use by SR.
    - SR Software may Accept a Migrate Out making the VF Inactive.
      - Until accepted, VF remains Active.
  - ✓ Handle *Migrate In (and Migrate Out) Retractions*.
    - Used by MR-PCIM to retract a Migration.
    - Migrate In Retraction moves the VF back to Inactive.

Copyright © 2007, PCI-SIG, All Rights Reserved

Initial VF Migration State

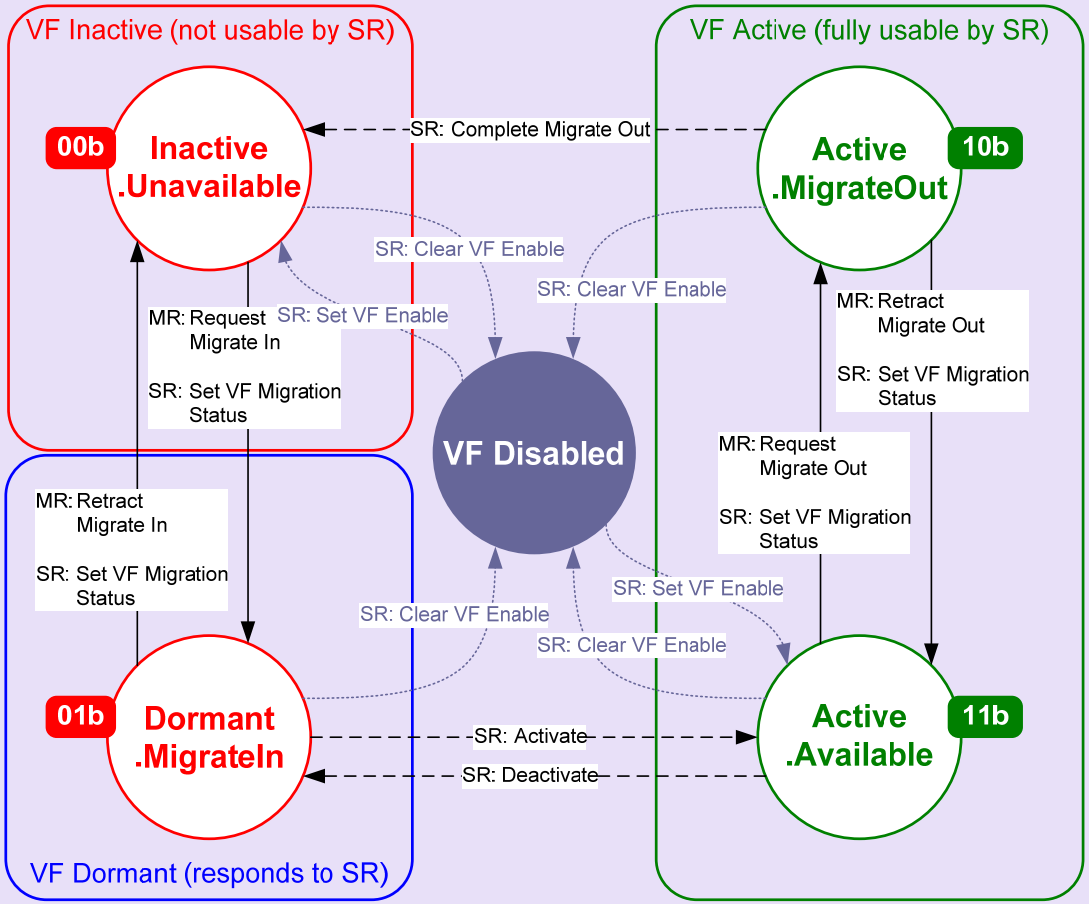


- If a Device is MR capable, then:
  - ✓ 1 ≤ InitialVFs ≤ TotalVFs
  - ✓ The initial VF states are:
    - Active VFs: VF_1 to $VF_{InitialVFs}$
    - Inactive VFs $VF_{InitialVFs+1}$ to $VF_{TotalVFs}$
      (Inactive VFs are “Holes” for VF Migration*)



Copyright © 2007, PCI-SIG, All Rights Reserved

VF Migration State Diagram



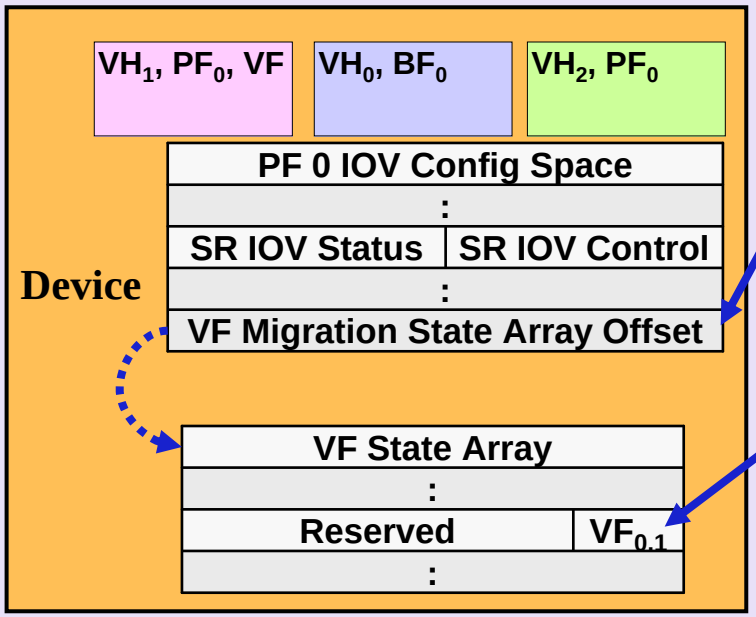
VF State	VF Exists	Description
00b*	No	***Inactive.Unavailable*** – VF does not exist to SR nor is it being migrated in or out.
01b	No	***Dormant.MigrateIn*** – VF is available for use by SR. VF exists but cannot initiate transactions.
10b	Yes	***Active.MigrateOut*** – SR has been requested to relinquish use of the VF.
11b	Yes	***Active.Available*** – Fully functional. Could be assigned to an SI.
*Initial states can be either of these		

- MR-PCIM initiates state transitions indicated by solid lines.
- SR-PCIM initiates state transitions indicated by dashed lines by writing the new state value to the VF State Array.

Copyright © 2007, PCI-SIG, All Rights Reserved

VF Migration

- VF Migration Status (in SR IOV Status) is used by MR-PCIM to indicate to SR-PCIM that a VF Migration In or Migration Out Request has been issued.
  - ✓ SR-PCIM must read the VF State Array (more next) to determine which VF(s) are being migrated in or out.
  - ✓ The VF Migration State Array Offset is a BAR relative offset to the location that contains the VF State Array.



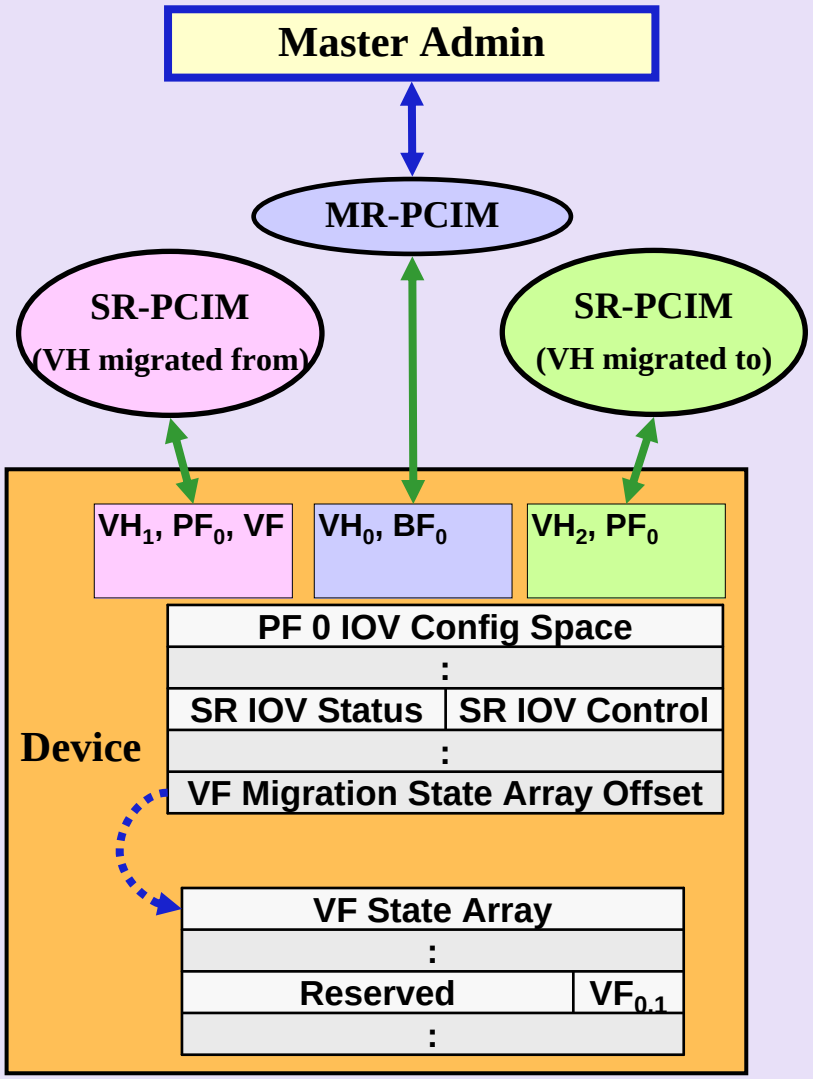
Lower 3 bits define the BAR to use for the offset.

Upper 29 bits are the offset into VF state array.

SR-PCIM Initiated VF Migration State Transitions		
Current	**New**	**Description**
Dormant. MigrateIn	Active. Available	**VF Activate** VF now exists.
Active. Available	Dormant. MigrateIn	**VF Deactivate** VF no longer exists.
Active. MigrateOut	Inactive. Unavailable	**VF Migrate Out Complete** VF no longer exists.

Copyright © 2007, PCI-SIG, All Rights Reserved

VF Migration Example



The next two slides will use the following nomenclature for the communications necessary to migrate a VF from one SR-PCIM to another SR-PCIM:

- ↔ HAL with SR-PCIMs and MR-PCIM
- ↔ Dev with SR-PCIMs and MR-PCIM

Copyright © 2007, PCI-SIG, All Rights Reserved

VF *Migrate Out* Example

5) VF Migrate Out interrupt to SR-PCIM, SR-PCIM informs SI of Migrate Out. SI accepts Migrate Out.

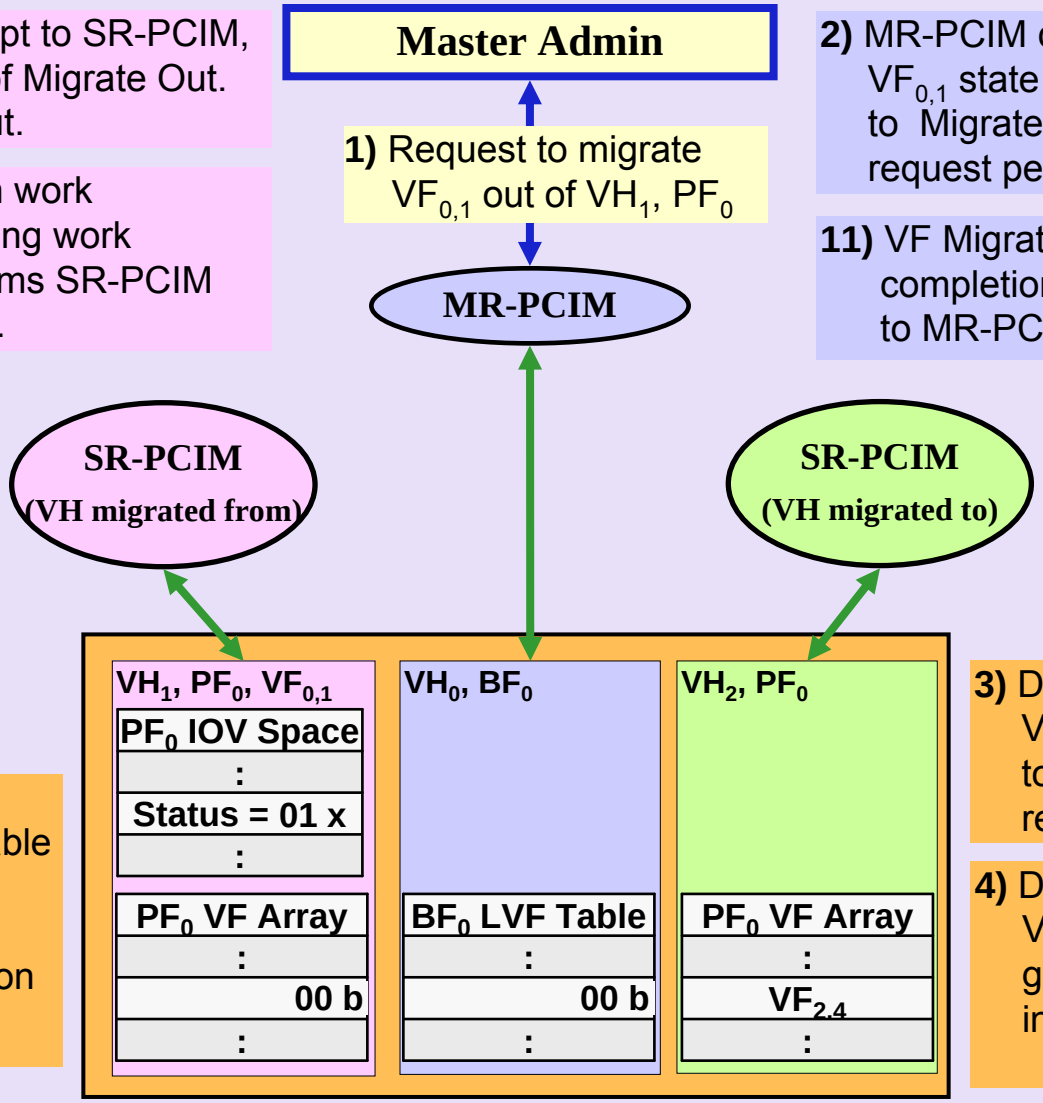
6) SI performs shut down work (e.g. wait for outstanding work to complete) and informs SR-PCIM VF is no longer in use.

7) SR-PCIM issues FLR to reset $VF_{0,1}$.

9) SR-PCIM changes $VF_{0,1}$ state to Inactive VF.

8) FLR Resets VF_0.

10) Device changes $VF_{0,1}$ state in LVF Table and sets *MR* VF Migration Status and generates VF Migration Interrupt on BF_0.



2) MR-PCIM changes $VF_{0,1}$ state in LVF Table to Migrate Out request pending.

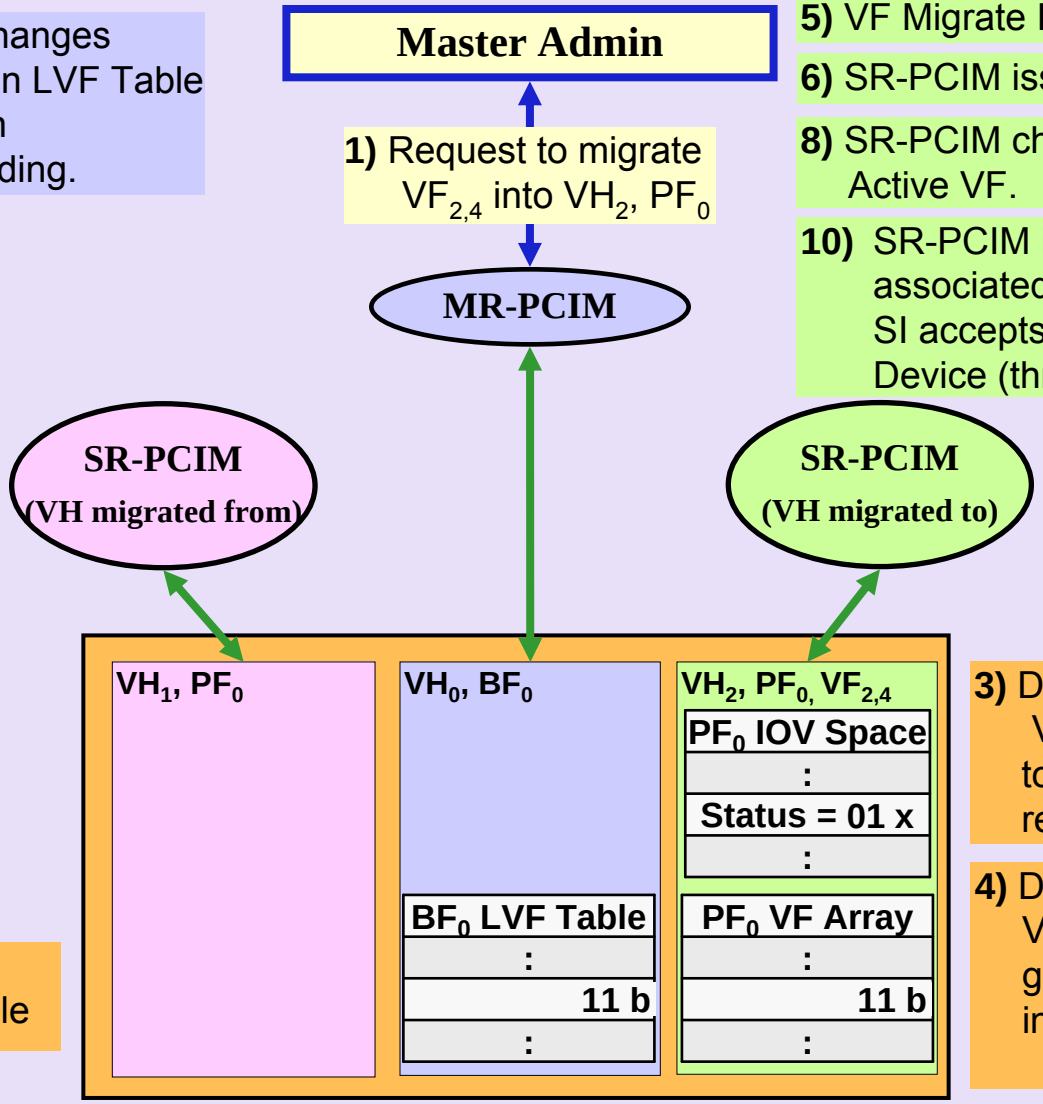
11) VF Migrate Out completion interrupt to MR-PCIM.

3) Device changes $VF_{0,1}$ state of VH_1, PF_0 to Migrate Out request pending.

4) Device sets VF Migration Status and generates VF Migrate Out interrupt on VH_1, PF_0.

Copyright © 2007, PCI-SIG, All Rights Reserved

VF *Migrate In* Example



2) MR-PCIM changes $VF_{2,4}$ state in LVF Table to Migrate In request pending.

5) VF Migrate In interrupt to SR-PCIM

6) SR-PCIM issues FLR to reset $VF_{2,4}$.

8) SR-PCIM changes $VF_{2,4}$ state to Active VF.

10) SR-PCIM informs SI that is to be associated with VF of Migrate In. SI accepts Migrate In & "discovers" Device (through SR-PCIM).

3) Device changes $VF_{2,4}$ state of VH_2, PF_0 to Migrate In request pending.

4) Device sets VF Migration Status and generates VF Migrate In interrupt on VH_2, PF_0.

7) FLR Resets $VF_{2,4}$.

9) Device changes $VF_{2,4}$ state in LVF Table

Copyright © 2007, PCI-SIG, All Rights Reserved

PCI Specification Schedule

ATS

SR-IOV

MR-IOV

Copyright © 2007, PCI-SIG, All Rights Reserved

Schedules

- ATS Specification released 3/8/2007
- SR-IOV Specification released 9/11/2007
- MR-IOV Specification
  - ✓ Following PCI-SIG Specification Process.
  - ✓ Draft 0.5 Completed
  - ✓ Draft 0.7 Completed
  - ✓ Draft 0.9 late 3Q/2007
  - ✓ Version 1.0 1/2008

Copyright © 2007, PCI-SIG, All Rights Reserved

Call To Action

- Please participate in the PCI-SIG Specification Development Process

Copyright © 2007, PCI-SIG, All Rights Reserved

Thank you for attending the PCI-SIG Developers Conference Europe 2007

For more information please go to www.pcisig.com

Copyright © 2007, PCI-SIG, All Rights Reserved

Workgroup Members

- AMD
- Broadcom
- Emulex
- HP
- IBM
- IDT
- Intel
- LSI Logic
- Microsoft
- NextIO
- Neterion
- Nvidia
- PLX
- Qlogic
- Stargen
- Sun
- VMWare

Copyright © 2007, PCI-SIG, All Rights Reserved

Implementing PCI I/O Virtualization Standards

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

Copyright © 2007, PCI-SIG, All Rights Reserved