Single Root Resource/Initialization/Event Management

Renato Recio

Disclaimer

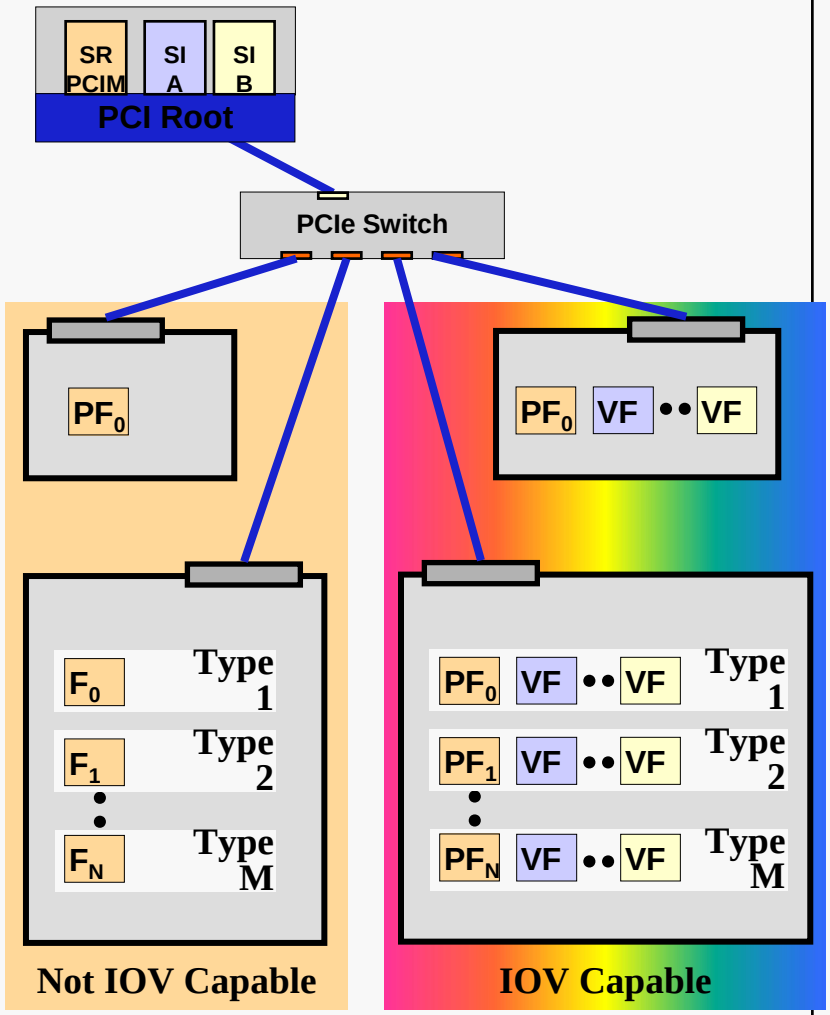
- NOTE: The information in this presentation refers to a specification still in the development process. This presentation reflects the current thinking of the workgroup, but all material is subject to change before the specification is released.

Contents

- Function types and terms
- Discovering a PCIe® Device's SR-IOV Capabilities
- Configuring a PCIe Device's SR-IOV Capabilities
- Discovering the Virtual Functions associated with a configured Physical Function
- Reset and Re-Initialization Mechanisms
- Migration of a Virtual Function across Virtual Hierarchies
- Usage Examples

Single-Root IOV Function Types and Terms

SR Topology



Terms

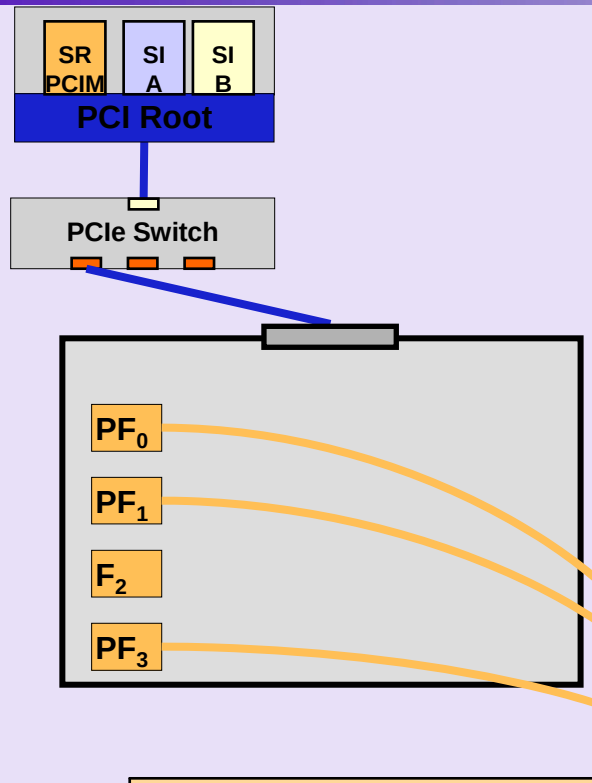
For SR topologies:

Physical Function (PF) is a PCIe Function that supports the SR-IOV capabilities.
A PF is used by SR-PCIM to discover and configure the set of Virtual Functions associated with the PF.

Physical Function 0 (PF_0) is also used to manage Device functions, such as link errors and events.

Virtual Function (VF) is a Function that is associated with a PF.
A VF shares one or more physical resources, such as a link, with other functions (PFs or VFs) on the same Device. A VF supports Native IO Virtualization.

SR-IOV Capability Discovery

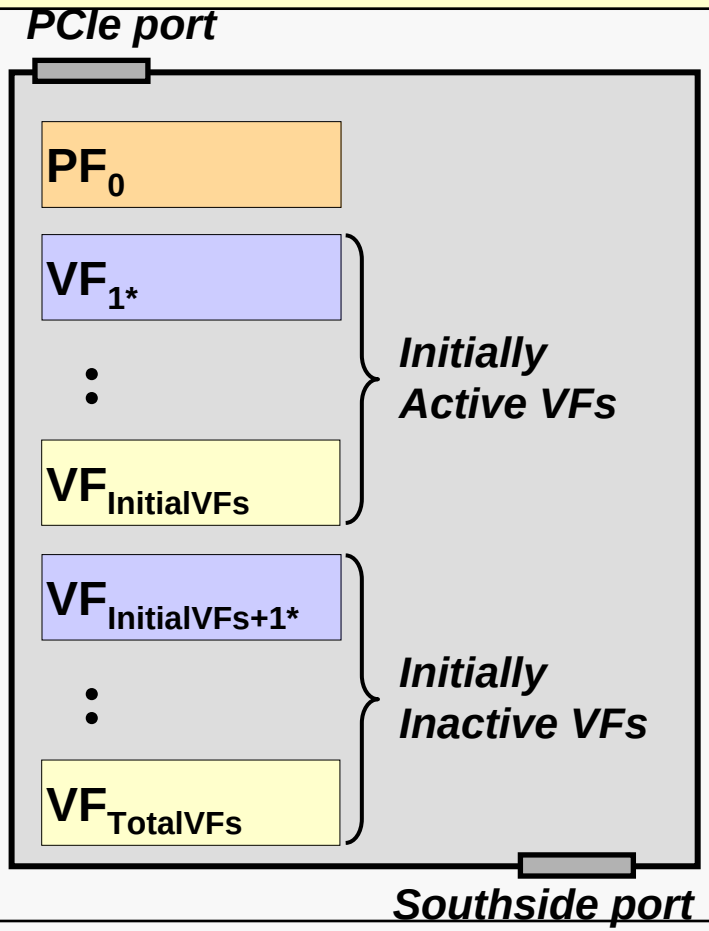


- Each PF must have an SR-IOV Extended Capability structure.
- A Device may have a mix of Functions and PFs.
- Each Function and PF consumes one Routing Identifier (RID).
- For a Device that:
  - ✓ isn't ARI capable, Functions and PFs must be in the first 8 functions;
  - ✓ is ARI capable, number of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

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

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 to that context.

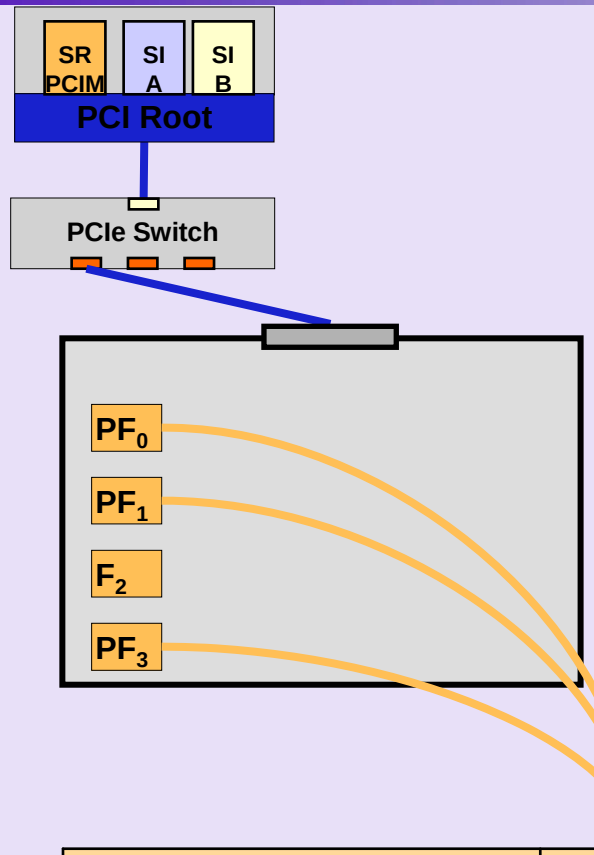
Inactive VF is a VF that must respond with Unrecognized Request (UR) to any transaction that targets that VF.

*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).*

VF Discovery - Part 1

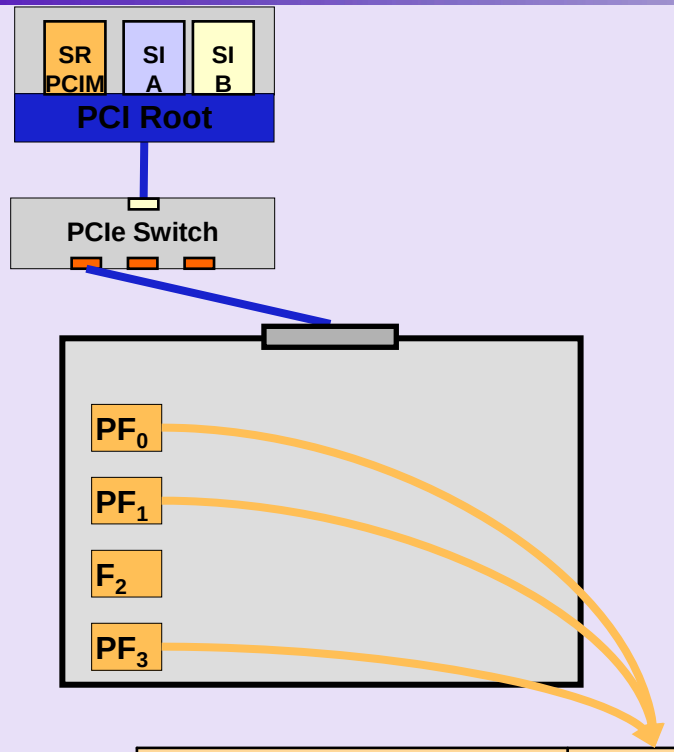


- The ***TotalVFs*** field is used to discover the number of ***Active VFs*** that a PF ***could have***.
- The ***InitialVFs*** field is used to discover the number of ***Active VFs*** that a PF ***initially has***.
- Note for a Device that **isn't** MR capable*:
  1 ≤ InitialVFs = TotalVFs

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

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

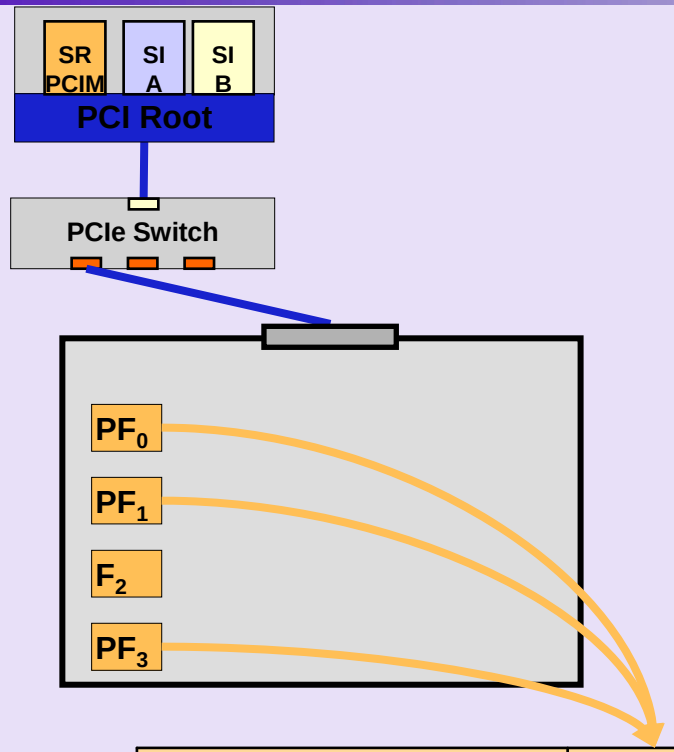
BAR Discovery



- 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)			20h
:			:
VF BAR5 (RW)			34h
:			:

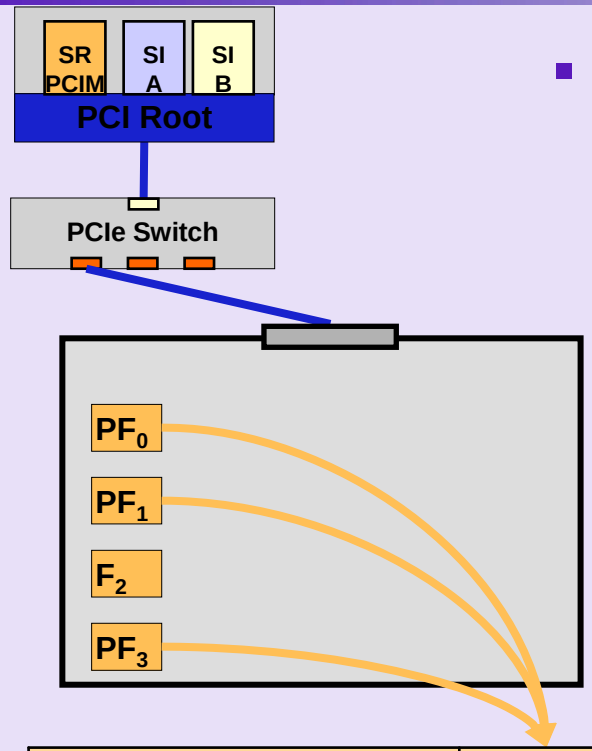
Discovering Supported Page Size



- 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.
  - ✓ 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)			18h
:			:

Configuring VFs



- VFs for a PF are enabled by writing NumVFs and then Setting the “VF Enable” bit.
  - ✓ When VF’s are enabled, the PCIe Device must associate NumVFs worth of VFs with the PF.
    - – If VF Migration Capable and VF Migration Enabled set, then NumVFs must be
      $1 \le NumVFs \le TotalVFs$
    - – Otherwise NumVFs must be
      $1 \le NumVFs \le InitialVFs$

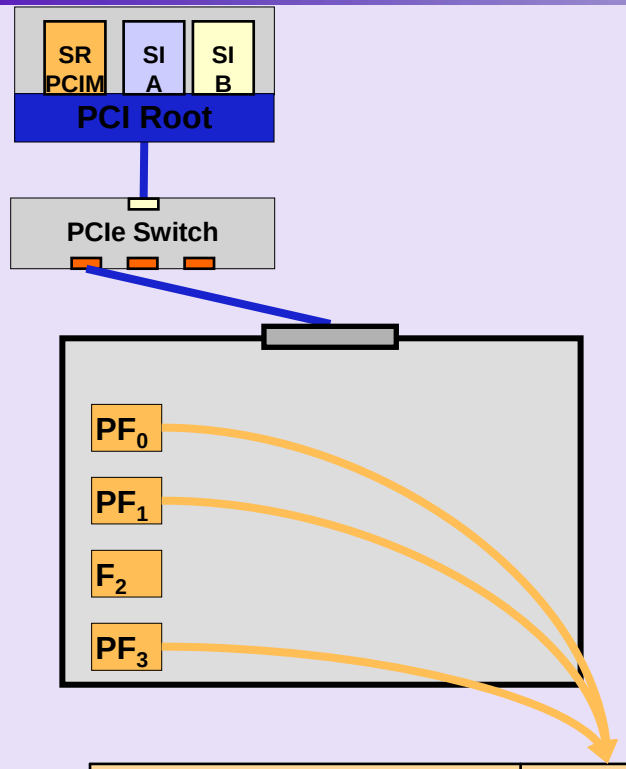
31 20	19 16	15 0	Offset
:			:
SR IOV Capabilities (RO)			04h
SR IOV Status		SR IOV Control (RW)	08h
Reserved	Func Dep Link	NumVFs	10h
:			:

VF Migration Capable

VF Migration Enable

VF Enable

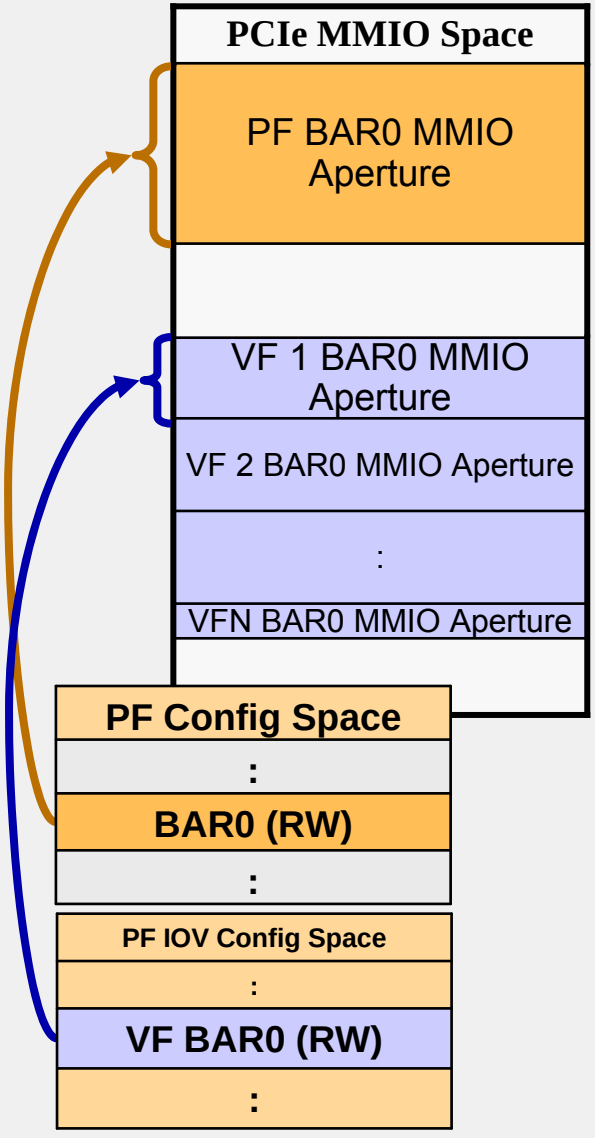
Configuring the VF's BARs



- 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
:			:
VF BAR0 (RW)			20h
:			:
VF BAR5 (RW)			34h
:			:

PF and VF BAR Semantics

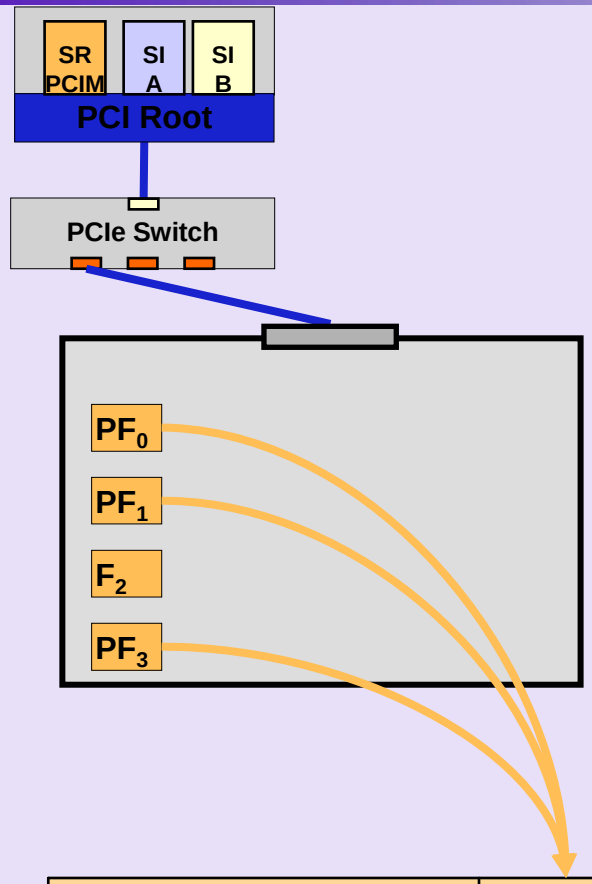


- The memory aperture required for each VF BAR can be determined by writing all "1"s and then reading the VF BAR.
  - ✓ Behaves as in the base PCI Spec.
- The address written into each VF BAR is used by the Device to set the starting address for that BAR on the first VF.
- The differences between VF 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 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.

VF Discovery - Part 2



- The values in NumVFs and VF ARI Enable affect the values for First VF Offset and VF Stride*.
  - ✓ First VF Offset and VF Stride are defined by the Device.
  - ✓ Following is the equation for determining the RID of VF N:

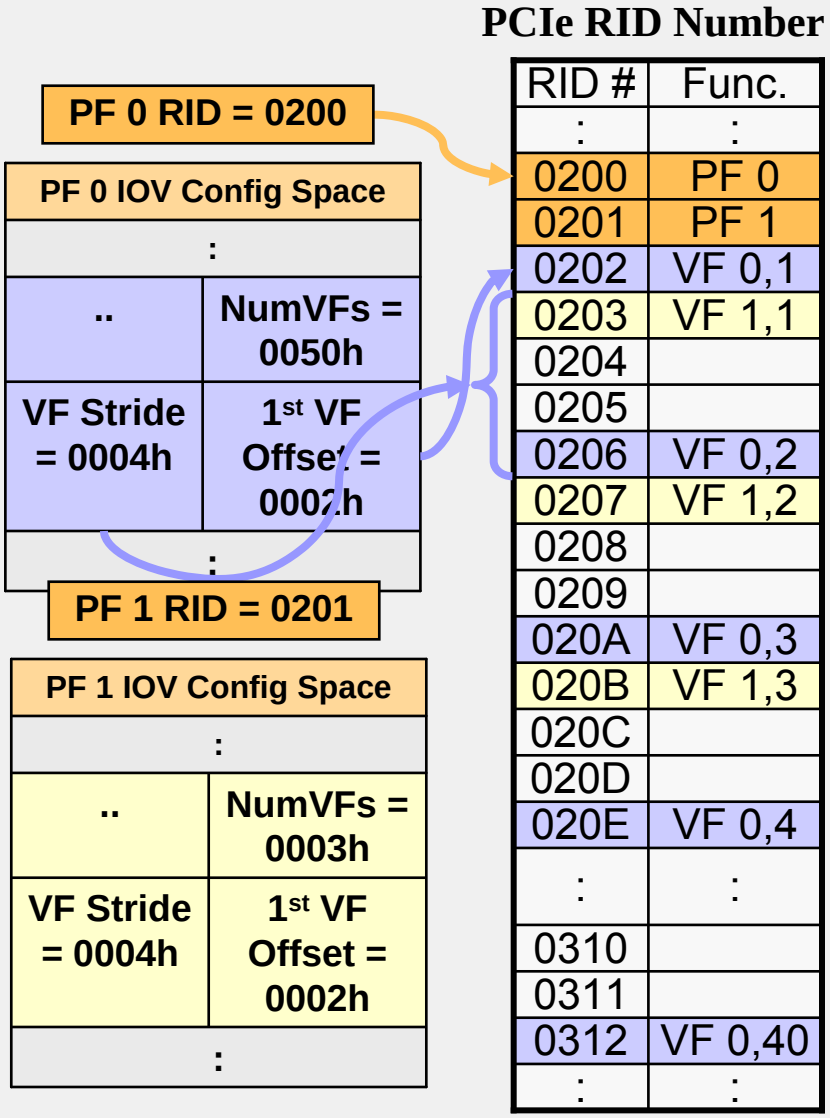
    VF N = [PF RID + VF Offset + (N-1) * VF Stride] Modulo 2^{16}

    where all arithmetic used is 16 bit unsigned dropping all carries.

**Note: After setting NumVFs and VF ARI Enable, SR-PCIM can read these two fields to determine how many busses will be consumed by the PF's VFs.*

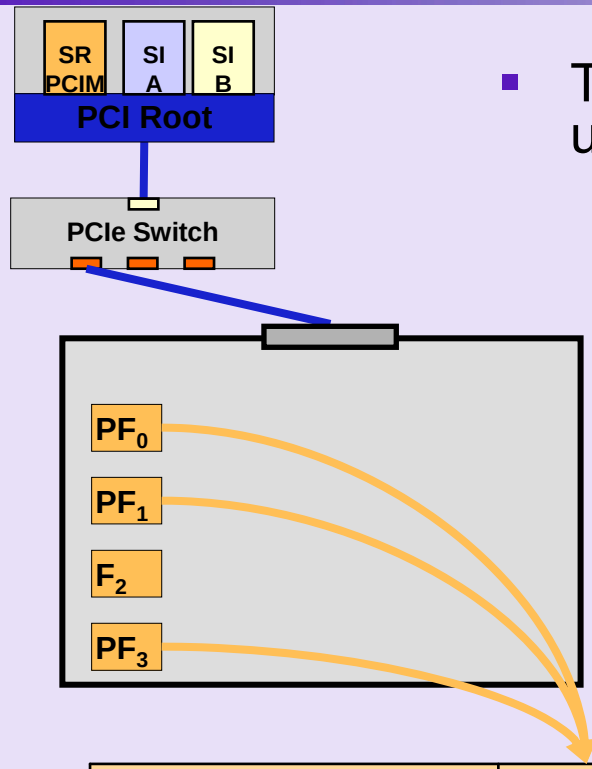
31 20	19 16	15 0	Offset
:			:
VF Stride (RO)		First VF Offset (RO)	14h
:			:

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 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 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.

Function Dependencies



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

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

Example	PF 0	PF 1	PF 2
Function Dependency Link	1	0	2
NumVFs	4	4	6

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.

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.

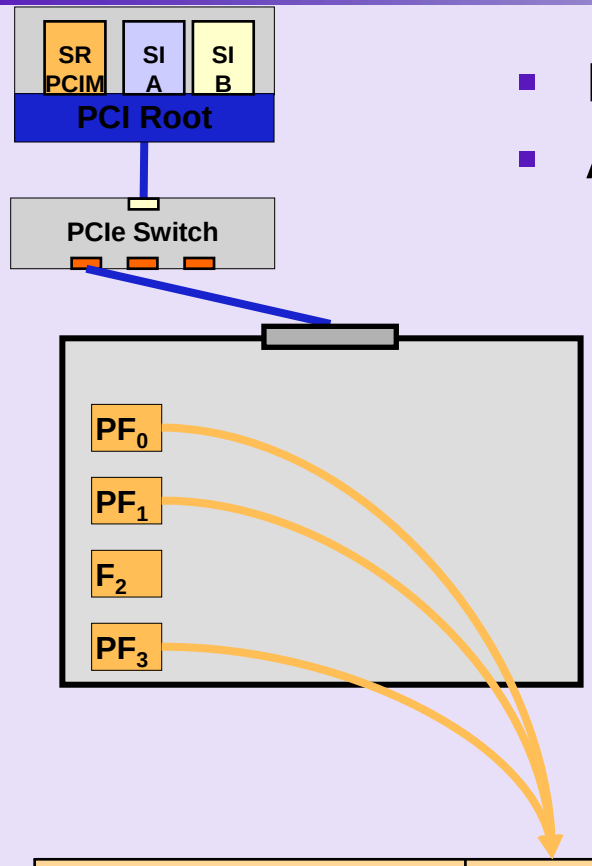
Conventional Reset

- 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.

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.

IOV Re-initialization and Reallocation

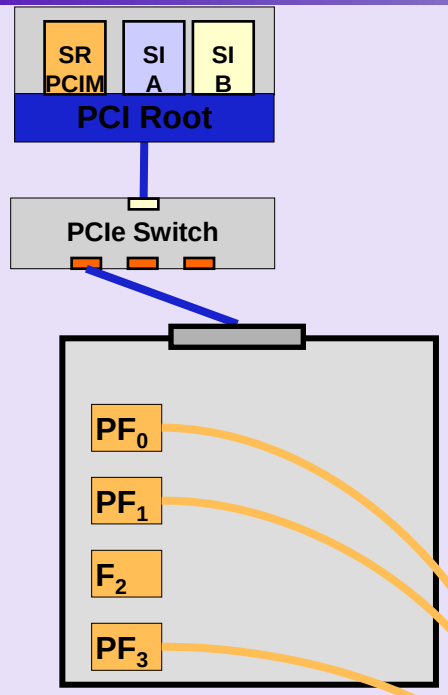


- Resetting a “VF Enable” disable a PF’s VFs.
- After a PF’s VFs are disabled:
  - ✓ The VFs must no longer:
    - Issue PCIe transactions,
    - Exist in the RID space,
    - Exist in the memory mapped address space,
    - Retain any context, or
    - Log any new VF errors.
  - ✓ Any errors already logged via PF error reporting registers, remain logged.

31 20	19 16	15 0	Offset
:			:
SR IOV Status		SR IOV Control	08h
:			:

VF Enable

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).

31 20	19 16	15 0	Offset
:			:
SR IOV Capabilities			04h
SR IOV Status		SR IOV Control	08h
:			:

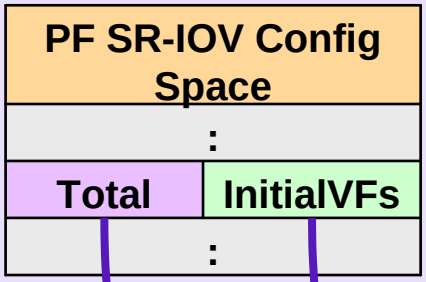
VF Migration Capable

VF Migration Enable

VF Migration Software Requirements

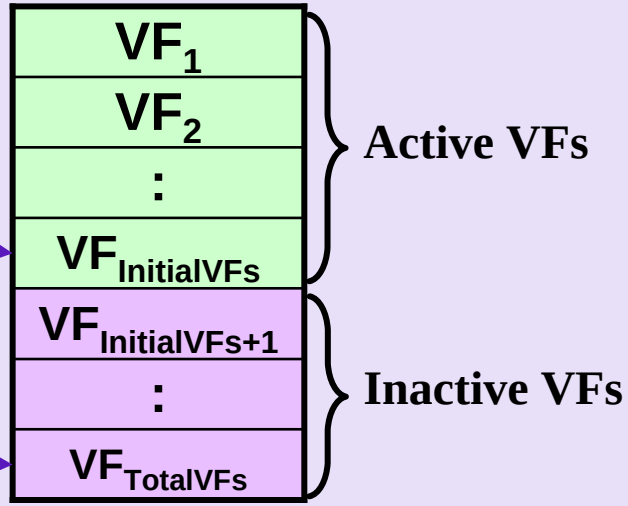
- SR software (e.g. SR-PCIM) must:
  - ✓ Determine if a VF is *Active* or *Inactive*.
    - A VF that is active must be available for use by SR.
    - An Inactive VFs must not be available for use by SR.
  - ✓ Participate in *Migrate In* operations.
    - A Migrate In operation is used by MR-PCIM to 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.

Initial VF Migration State

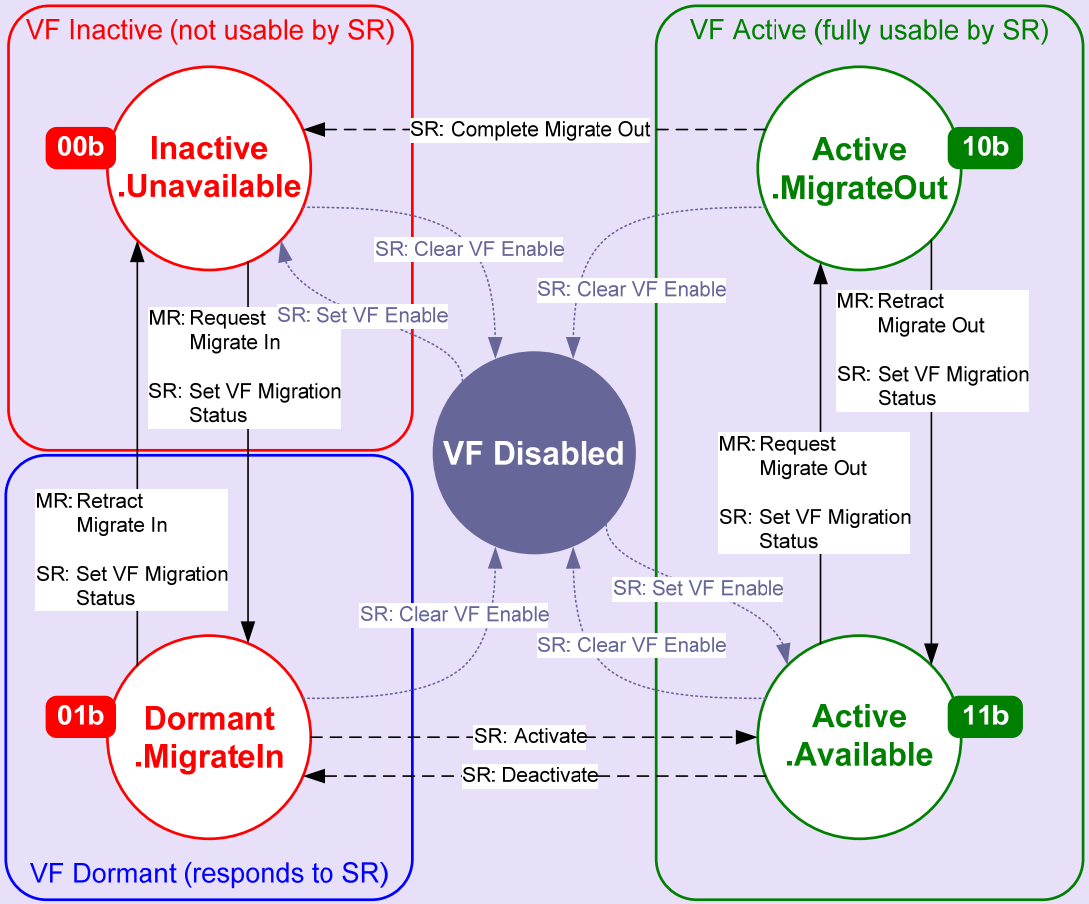


- If a Device is MR capable, then:
  - ✓ $1 \leq \text{InitialVFs} \leq \text{TotalVFs}$
  - ✓ The initial VF states are:
    - Active VFs VF_1 to $VF_{InitialVFs}$
    - Inactive VFs $VF_{InitialVFs+1}$ to $VF_{TotalVFs}$

    (Inactive VFs are "Holes" for VF Migration*)



VF Migration State Diagram



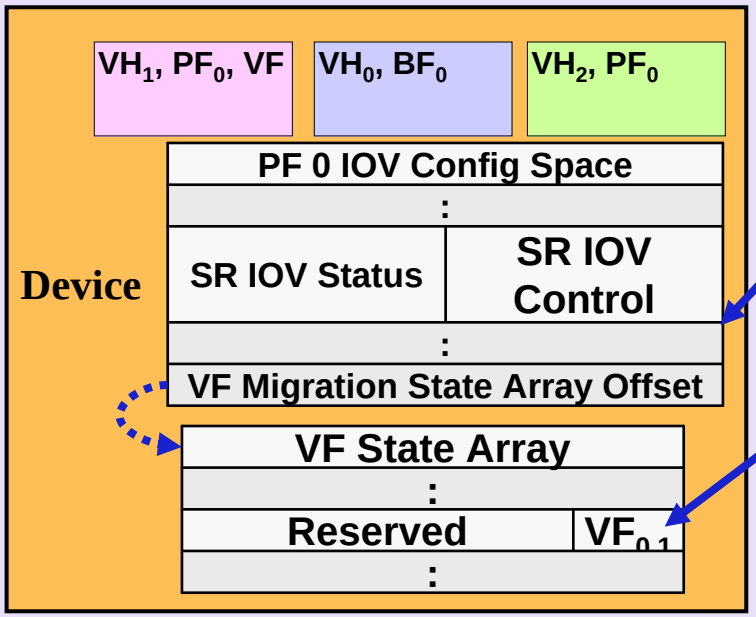
- When VF Enable is set,
  the initial VF states are:
  - ✓ Active, or
  - ✓ Inactive

Current State	Written State	Meaning
01b	11b	SR Activate VF
11b	01b	SR Deactivate VF
10b	00b	SR Complete Migrate Out

- 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.

VF Migration

- The VF Migration Status field 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.



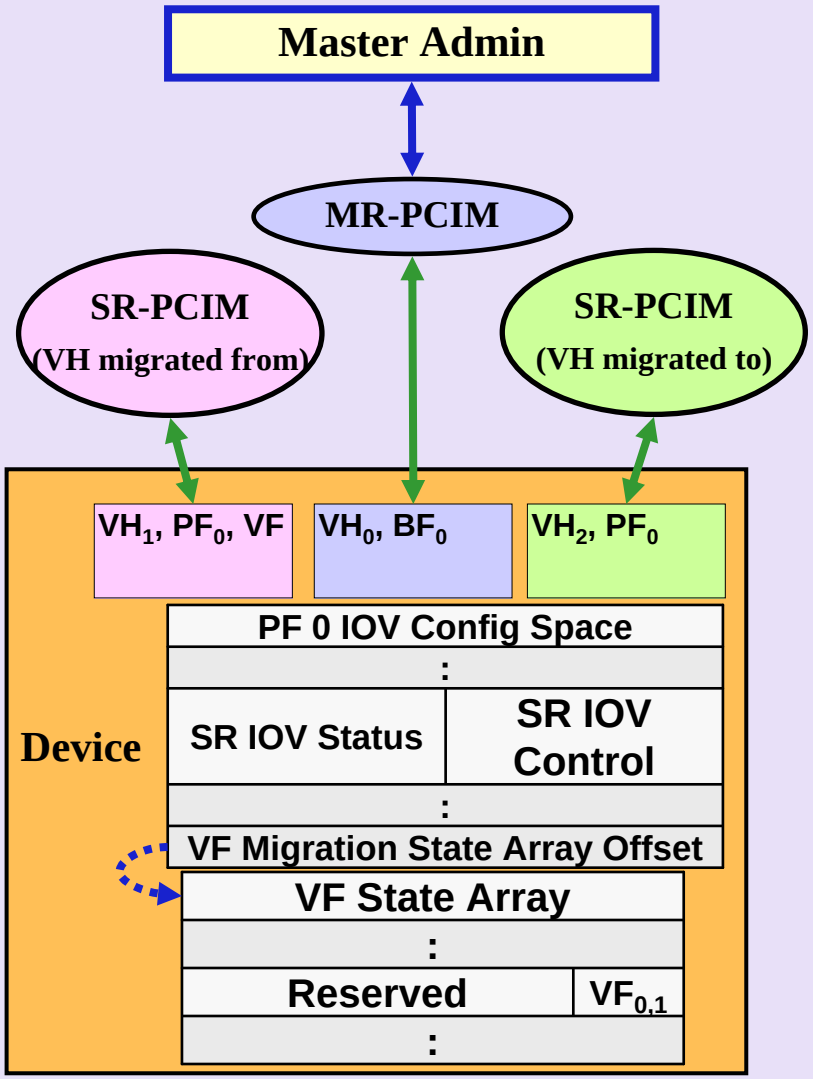
Lower 3 bits define the BAR to use for the offset.

Upper 29 bits are the offset into VF state array.

VF State	VF Exists	Description
00b	No	**Inactive.Unavailable**
01b	Partially*	**Dormant.MigrateIn**
10b	Yes	**Active.MigrateOut**
11b	Yes	**Active.Available**

*Note: SR-PCIM can issue an FLR to a VF in this state, but until the Migrate is accepted, the VF doesn't fully exist.

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

VF *Migrate Out* Example

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

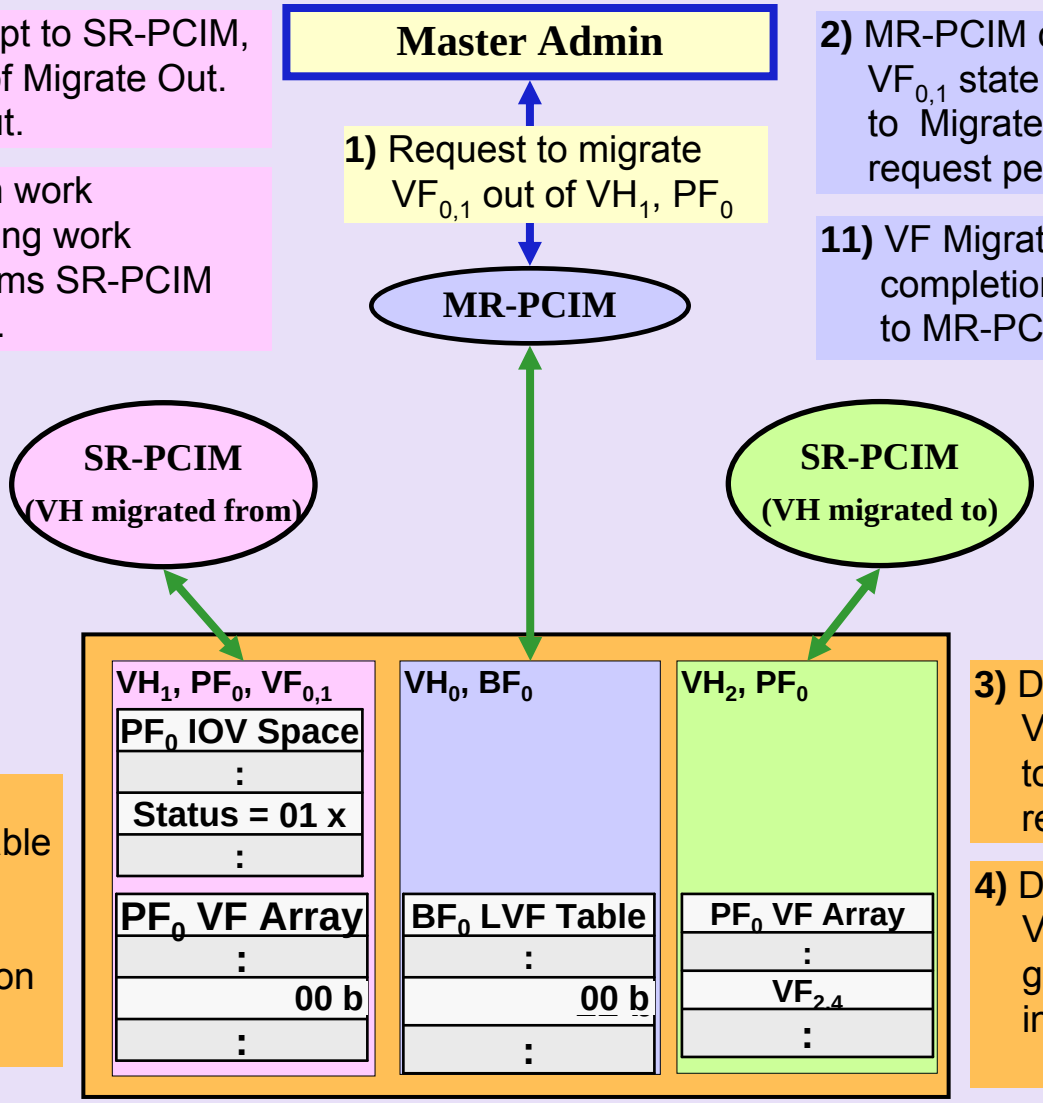
6) SI performs shut down work (e.g. wait for outstanding work to complete) and informs SR-PCIM VF is no longer in use.

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.

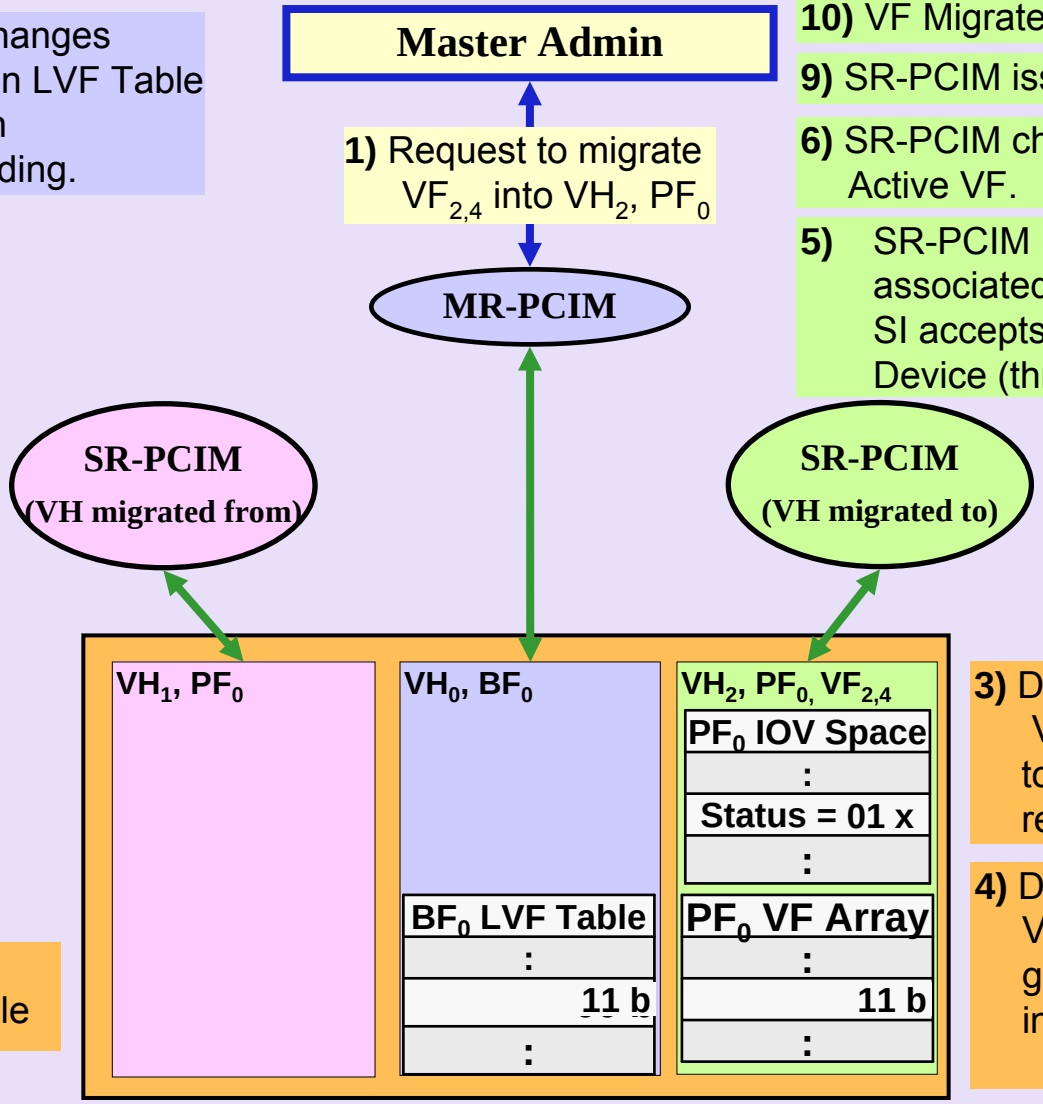
11) VF Migrate Out completion interrupt to MR-PCIM.

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.

VF *Migrate In* Example

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



10) VF Migrate In interrupt to SR-PCIM

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

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

5) 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.

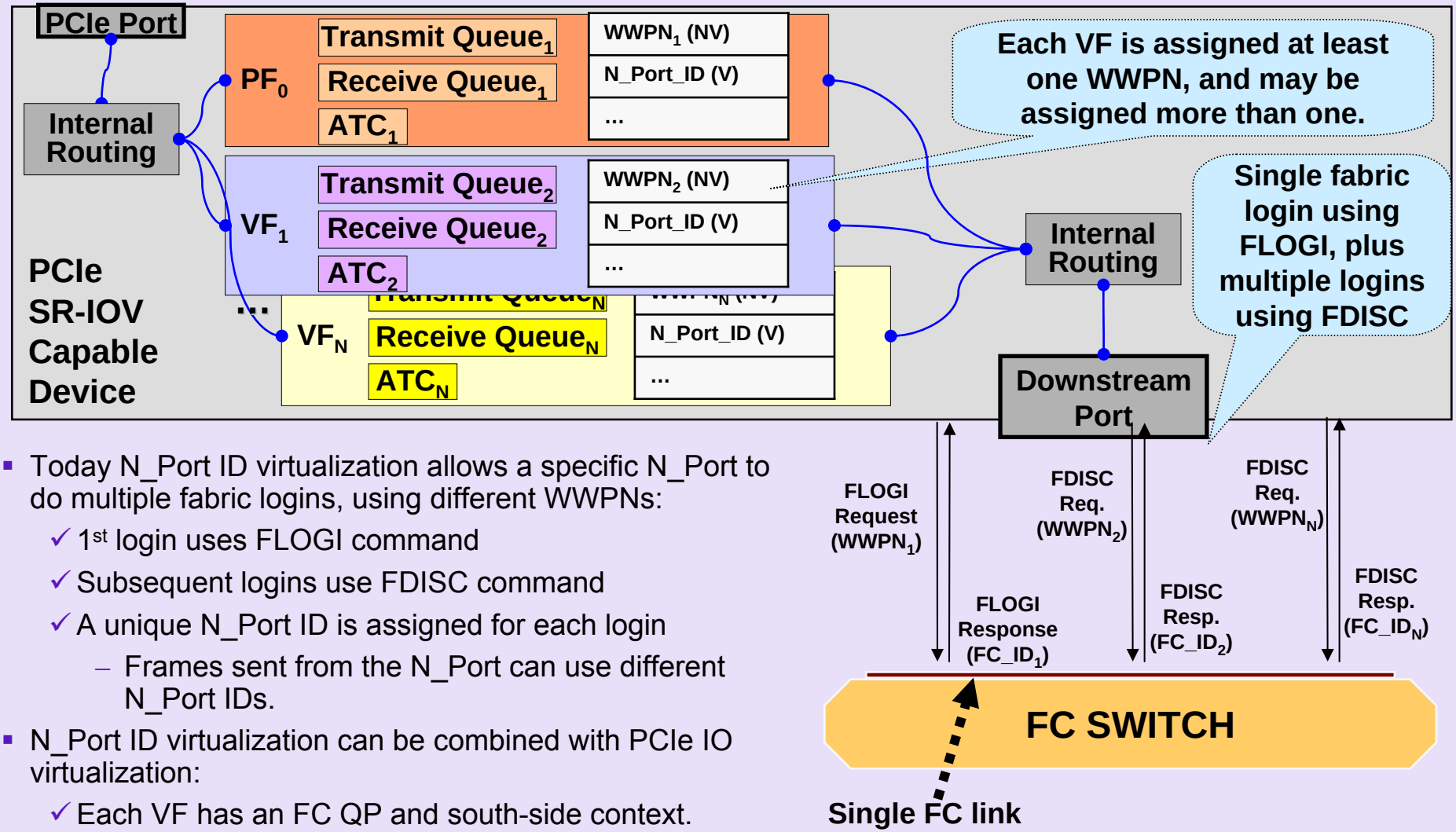
7) FLR Resets $VF_{2,4}$.

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

SR-IOV Usage Examples

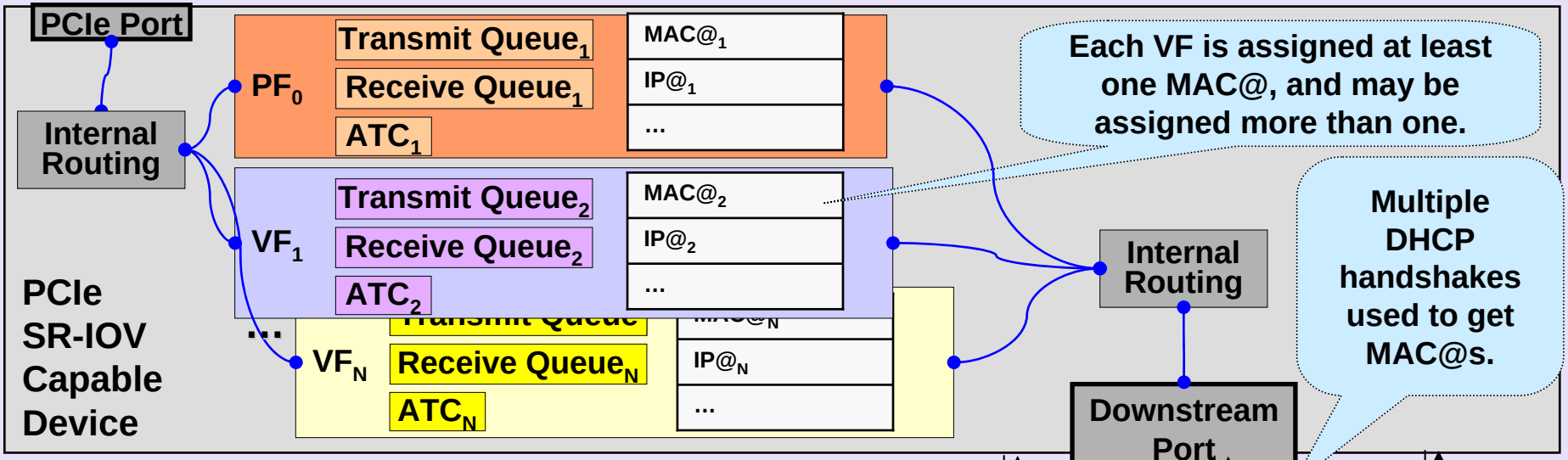
- Example 1
  - ✓ Virtualization of south side logic used by an FC adapter.
- Example 2
  - ✓ Virtualization of south side logic used by an Ethernet adapter.
- Example 3
  - ✓ Virtualization of south side logic used by an iWARP RNIC.
- Example 4
  - ✓ Virtualization of south side logic used by an IB HCA.

Example 1 - FC HBA



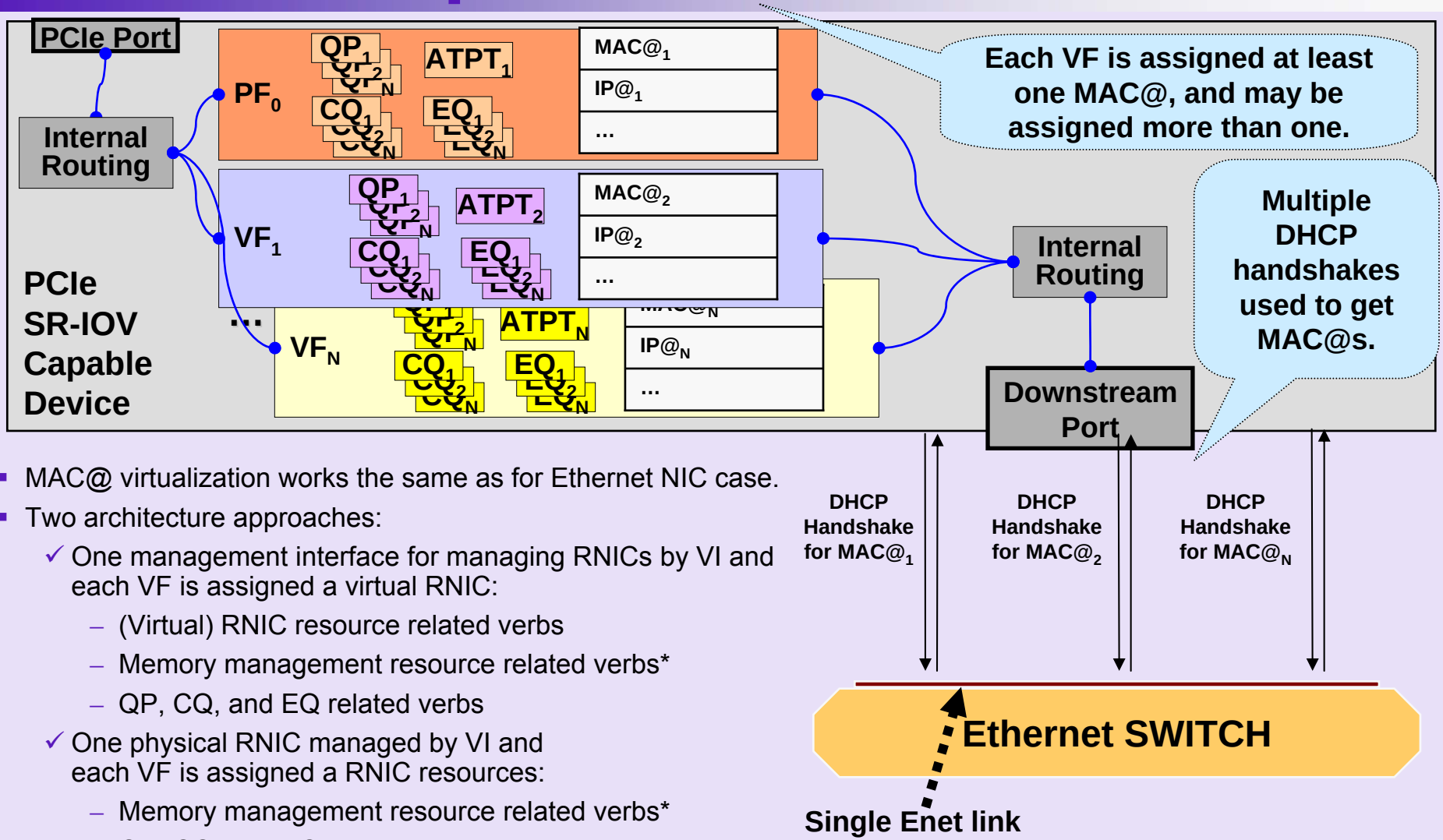
- Today N_Port ID virtualization allows a specific N_Port to do multiple fabric logins, using different WWPNs:
  - ✓ 1st login uses FLOGI command
  - ✓ Subsequent logins use FDISC command
  - ✓ A unique N_Port ID is assigned for each login
    - Frames sent from the N_Port can use different N_Port IDs.
- 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.

Example 2 - Ethernet NIC



- Today MAC@ virtualization allows DV 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.

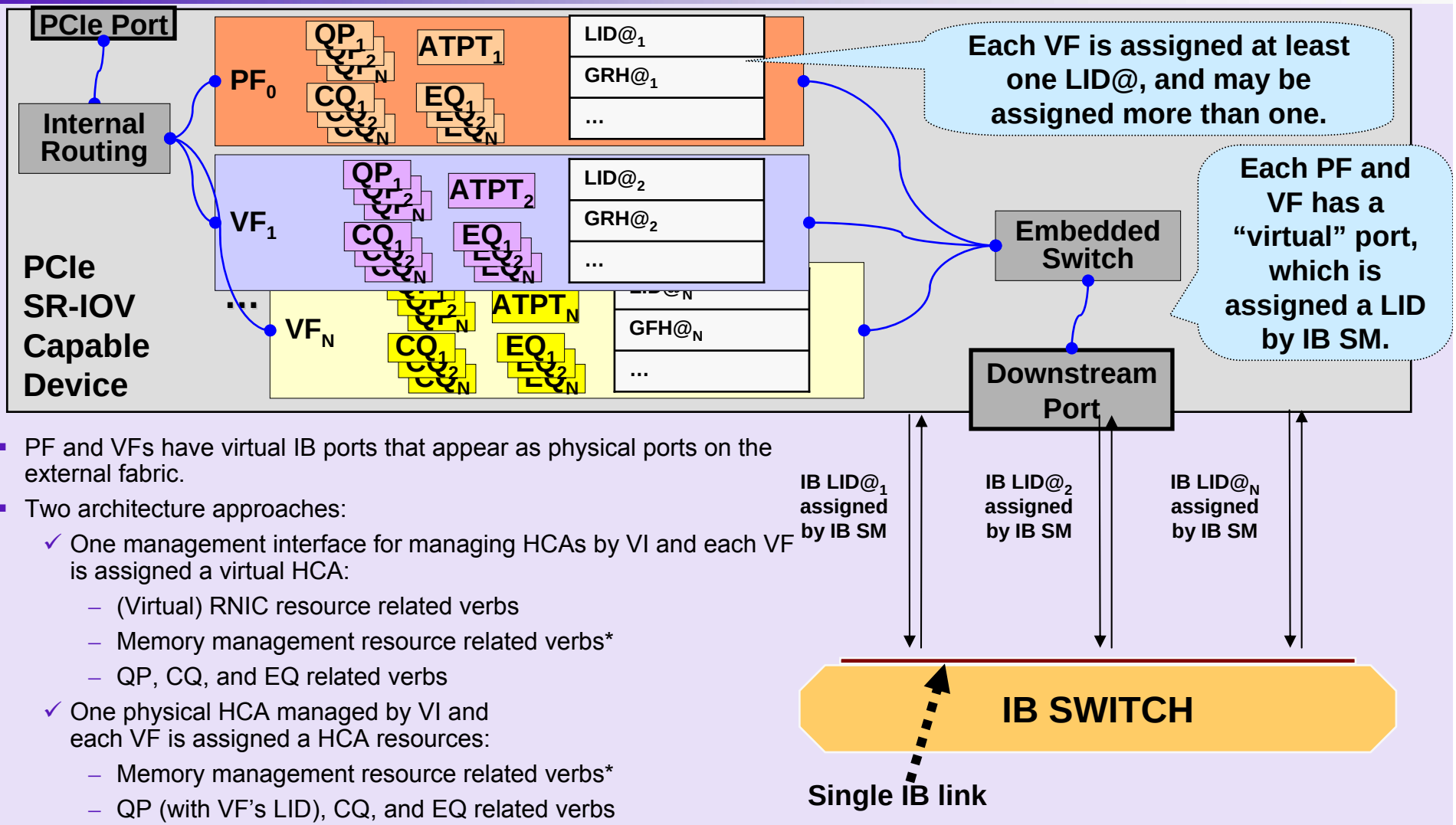
Example 3 - iWARP RNIC



- MAC@ virtualization works the same as for Ethernet NIC case.
- 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 a RNIC resources:
    - Memory management resource related verbs*
    - QP, CQ, and EQ related verbs

*Note: memory management is PF function, not a VF function if ATC example 5 is used.

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) RNIC 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: memory management is PF function, not a VF function if ATC example 5 is used.

Thank you for attending the PCI-SIG Developers Conference 2007

For more information please go to
www.pcisig.com

PCI-SIG® Developers Conference 2007

May 21-22, 2007