Multi-Root IOV

Chris Pettey

NextIO

Work In Progress

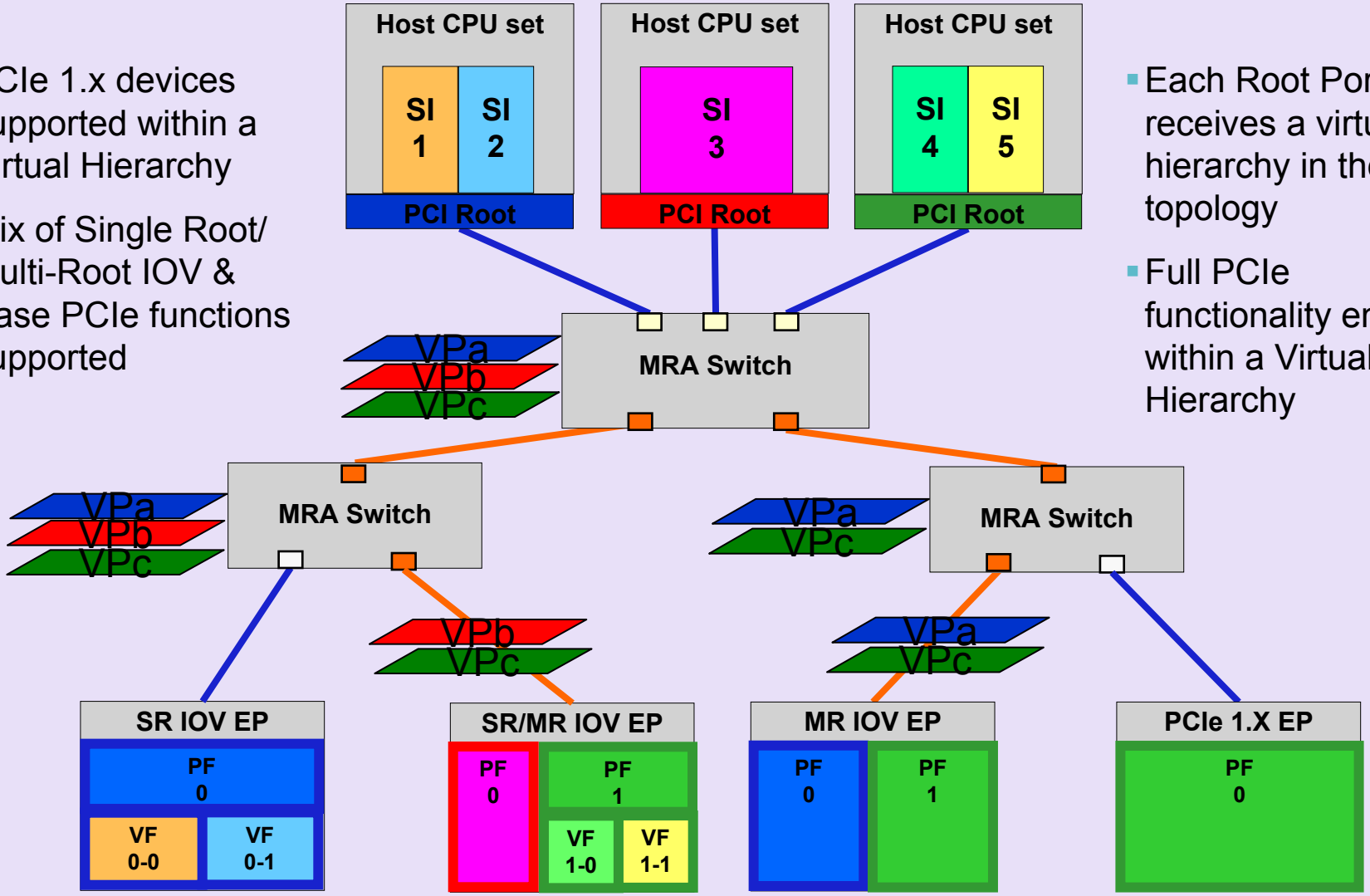
NOTE: The information in this presentation refers to a specification still in the development process. This presentation reflects the current thinking of the workgroup, but all material is subject to change before the specification is released.

Multi-Root (MR) Introduction

- MR components group to create Virtual Hierarchies (VH)
  - ✓ Virtual Hierarchy = a logical PCIe® hierarchy within a MR topology
  - ✓ Extends from a Root Port (RP) to all its Endpoints (EPs)
- MR links are carved into Virtual Planes (VP)
  - ✓ Virtual Plane = a slice of a link which operates as a logical PCIe link
    - – Includes protocol changes to enable multiple VP on a link
- MR PCIM manages the new Multi-Root Aware (MRA) elements
  - ✓ MR PCIM = The SW configuration entity which controls the MR portions of the MRA topology
- PCIe Base RPs are assigned a Virtual Hierarchy
  - ✓ RP controls the operation of the devices within a VH

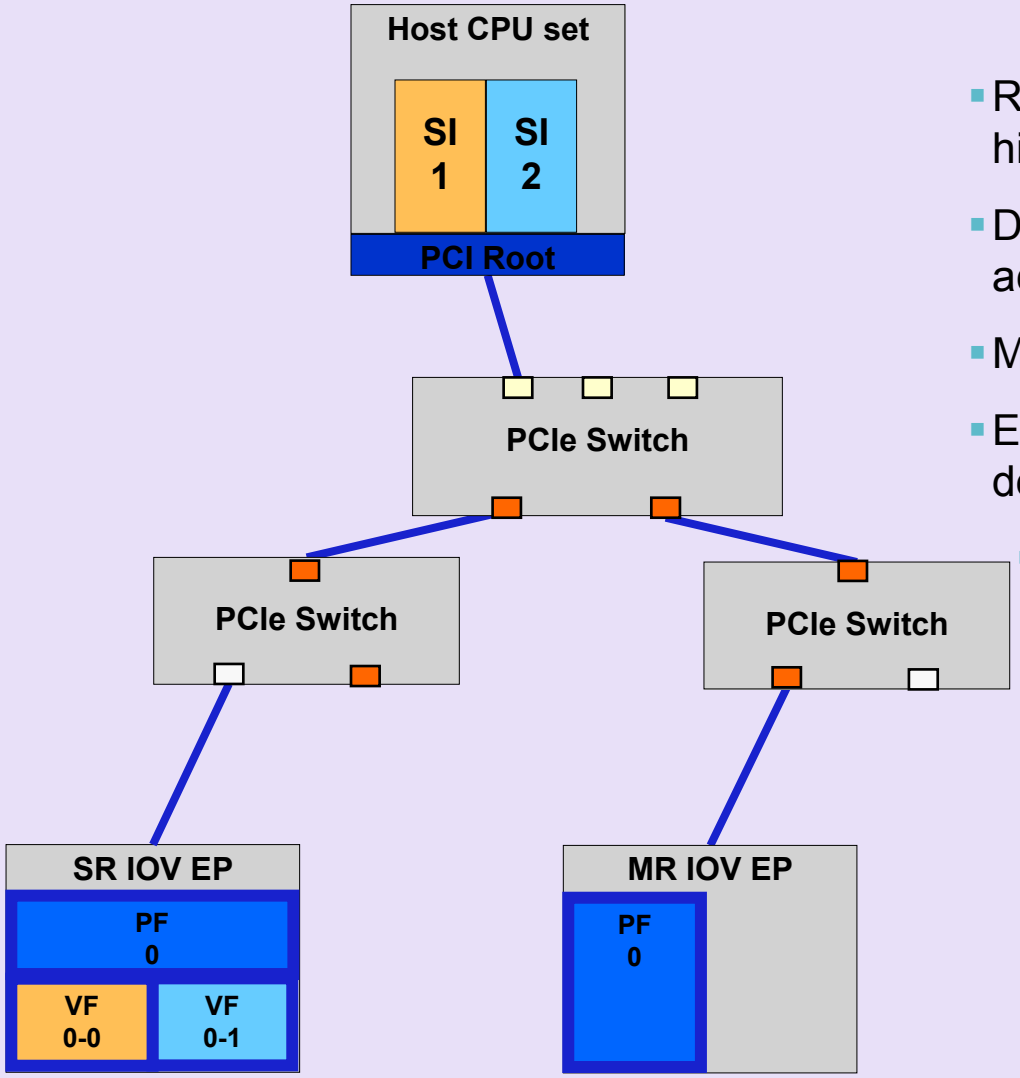
Multi-Root Aware (MRA) Topology Overview

- PCIe 1.x devices supported within a Virtual Hierarchy
- Mix of Single Root/ Multi-Root IOV & Base PCIe functions supported



- Each Root Port receives a virtual hierarchy in the topology
- Full PCIe functionality enabled within a Virtual Hierarchy

Root Port's View of a Virtual Hierarchy



- Root Port "sees" only devices within its hierarchy
- Devices in other hierarchies are not accessible
- MRA switches appear as PCIe 1.X switches
- Endpoints appear as Single Root equivalent devices
  - PCIe 1.X devices with IOV capabilities

Multi-Root Solution Impact

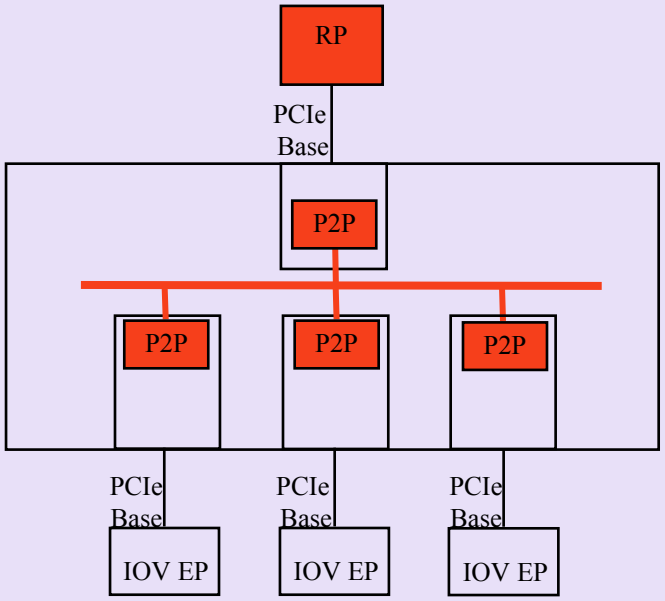
- Independent Virtual PCIe Hierarchy (VH) per Root Port
  - ✓ Virtual PCIe Hierarchy extends from Root Port to endpoints
  - ✓ All valid PCIe operations exist within virtual hierarchy
    - Operations may only have meaning in virtual context – No physical action
  - ✓ VHs are separate, independently controlled resources
    - Each Root Port only has access to the resources of its VH
  - ✓ PCIe routing rules apply within a VH
- Protocol Changes Required for implementation
  - ✓ Tag each PCIe TLP on link with a Virtual Plane (VP) identifier
    - Added and removed at DLL
  - ✓ Per plane reset within DLL
- Component impact
  - ✓ Root Port – No (uses PCIe base)
  - ✓ Switch – Yes (implements virtual PCIe switch per VP)
  - ✓ Endpoint – Yes (implements Type0 Config Space per VP)
  - ✓ RP Software – No (utilizes PCIe SW programming model)

New Multi-Root Aware (MRA) Components

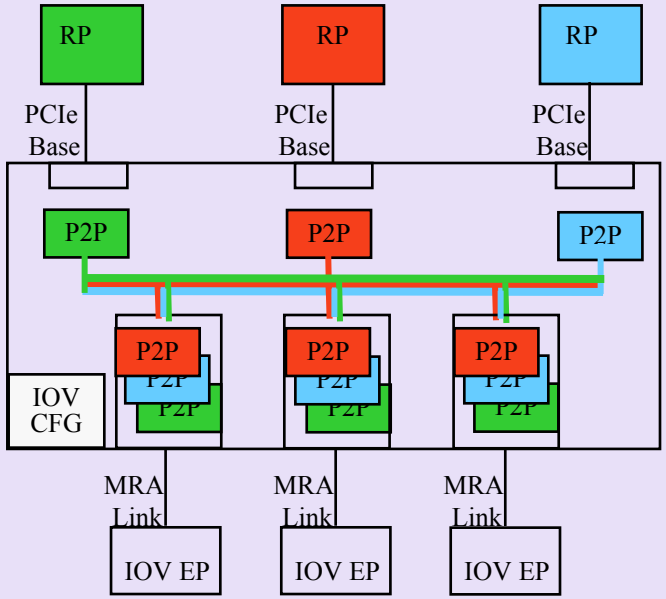
- MRA Switches
  - ✓ Switches that participate in multiple Virtual Hierarchies
    - – Utilize VP protocol extensions
  - ✓ Include a virtual PCIe switch for each Virtual Hierarchies
    - – Type1 Config space (aka Physical Function or PF) for each VH
    - – Virtual operations for RESET, Hot Plug, Power Management, etc
- MRA Endpoints
  - ✓ Endpoints that participate in multiple Virtual Hierarchies
    - – Utilize VP protocol extensions
  - ✓ Include a Physical Function (PF) for each Virtual Hierarchy
    - – Type0 Confg space for each VH
- MR PCIM
  - ✓ Management Software that is aware of IOV MRA features
  - ✓ Controls operations which span multiple Virtual Hierarchies

Multi-Root Aware Switches

PCIe 1.X Switch



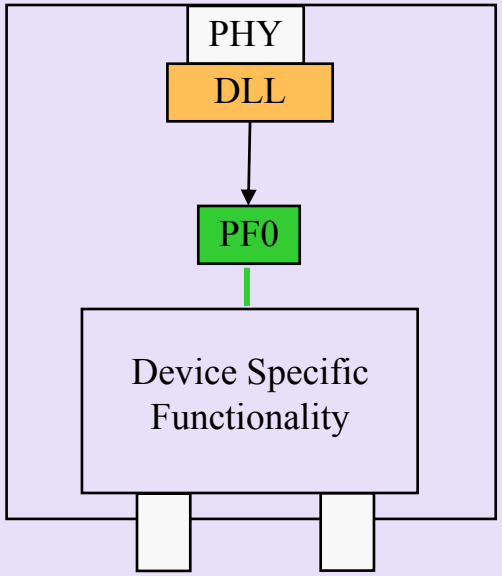
MRA Switch



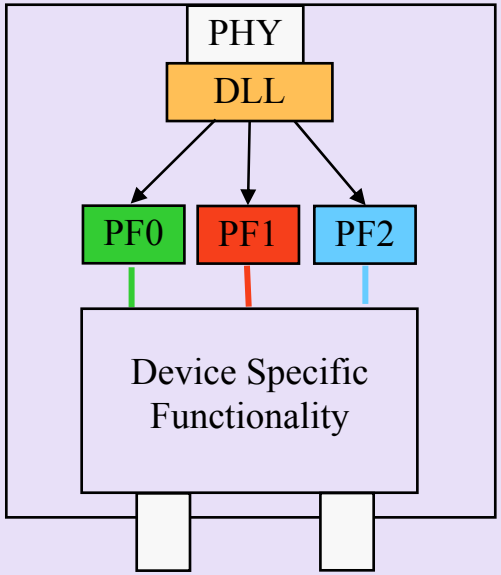
- Virtual Switch per Virtual Hierarchy
  - ✓ PCIe Type1 Config space per VH
  - ✓ Virtual Hot plug control, Power management tracking, Isolation, Error containment & processing per VH
- Virtual Plane protocol support
  - ✓ Insert/Remove VP label & Process Reset DLLP
- PCIM interface
  - ✓ SW registers model for MR PCIM control of shared resources

Multi-Root Aware Endpoint

PCIe 1.X Endpoint

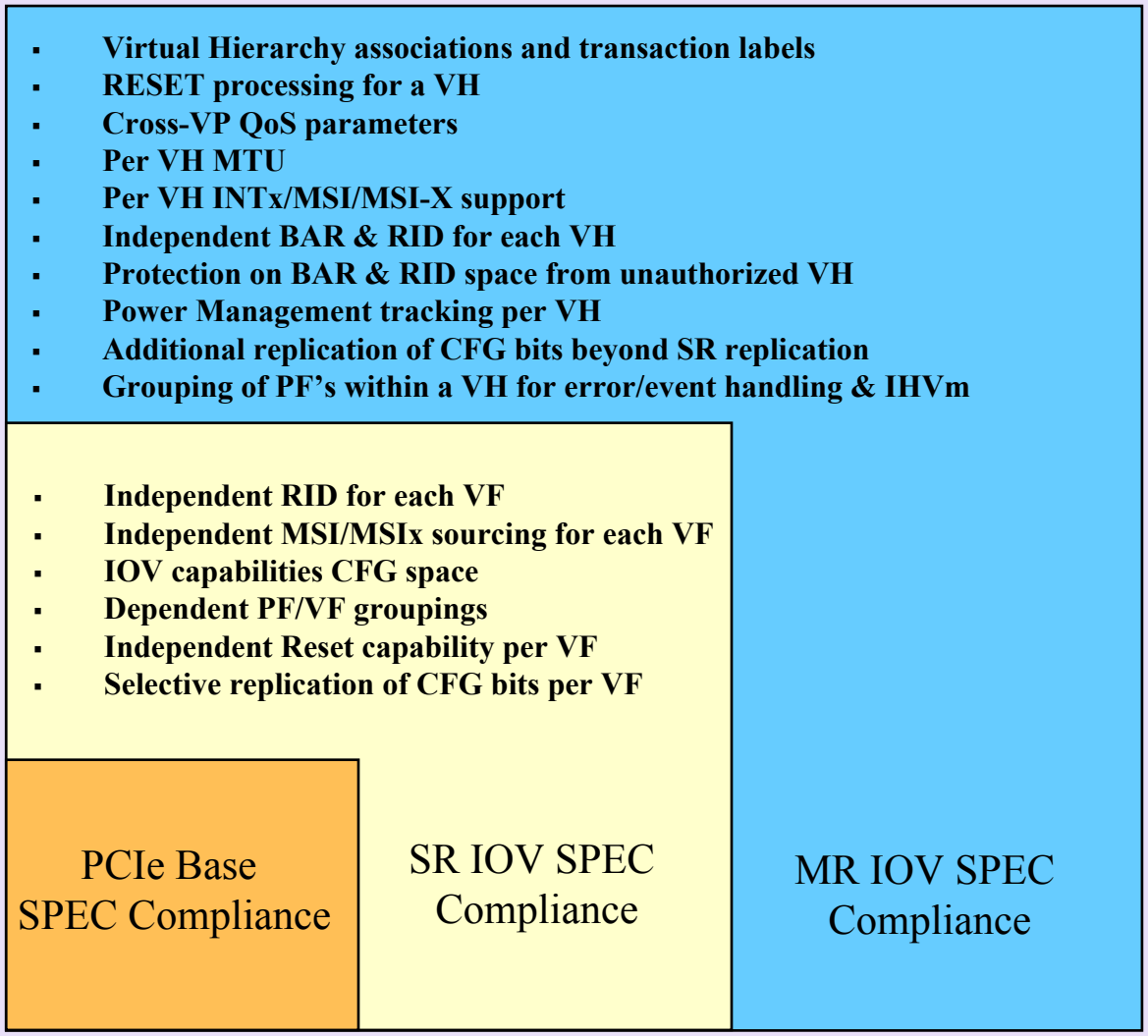


MRA Endpoint



- Virtual Plane protocol support
  - ✓ Insert/Remove VP label at DLL
  - ✓ Process DLLP for VP RESET
- Virtual Endpoint(s) per Virtual Hierarchy
  - ✓ Type 0 Config space (aka Physical Function or PF) per VH
  - ✓ IOV capabilities within VH for SR IOV support
- PCIM interface
  - ✓ Registers for MR PCIM control
  - ✓ Included in IOV CFG space of PF

Base->SR->MR Endpoint Progression



- MR Endpoints function as SR or Base Endpoints
  - ✓ MR capabilities unused by non-MRA SW
  - ✓ All SR capabilities included in MR device
- SR Endpoints function as Base Endpoints
  - ✓ IOV capability register ignored by PCIe Base SW
- Goal is to minimize cost of new functions
  - ✓ Minimal Config space replication
  - ✓ Minimal logic impact of new functions

Multi-Root Aware Switch Type1 Config Space

- Fixed Type1 Header Bits – RO or Not Used
  - ✓ Vendor ID
  - ✓ Device ID
  - ✓ Revision ID
  - ✓ Class Code
  - ✓ Latency Timer
  - ✓ Header Type
  - ✓ BIST
  - ✓ Cap Pointer
- Variable Type1 Header Bits – Selective Replication per Physical Function
  - ✓ Command
  - ✓ Status
  - ✓ Cache Line Size
  - ✓ BAR 0 & 1
  - ✓ Primary, Secondary, Subordinate Bus Numbers
  - ✓ Secondary Status
  - ✓ Memory Base/Limit, Prefetchable Memory Base/Limit, Prefetchable Base/Limit Upper
  - ✓ I/O Base/Limit, I/O Base/Limit Upper
  - ✓ Bridge Control

Multi-Root Aware Endpoint Type0 Config Space

- Fixed Type0 Header Bits – RO or Not Used
  - ✓ Vendor ID
  - ✓ Device ID
  - ✓ Revision ID
  - ✓ Class Code
  - ✓ Latency Timer
  - ✓ Header Type
  - ✓ BIST
  - ✓ Cap Pointer
- Variable Type0 Header Bits – Selective Replication per Physical Function
  - ✓ Command
  - ✓ Status
  - ✓ Cache Line Size
  - ✓ BAR 0, 1, 2, 3, 4, 5
  - ✓ Expansion ROM

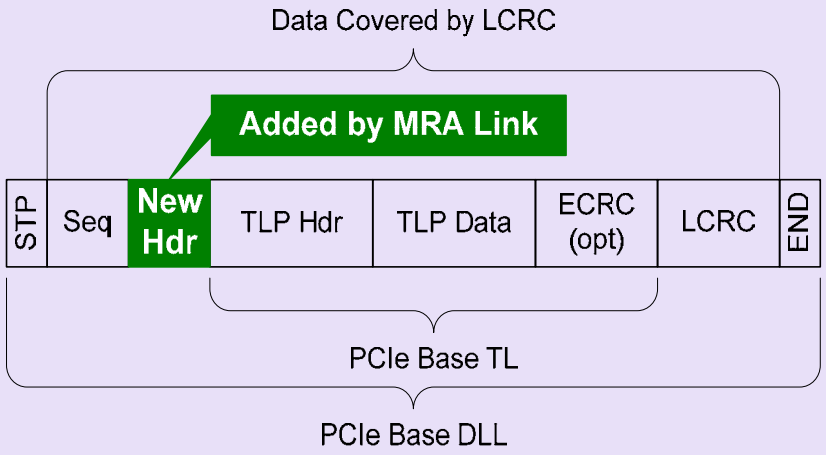
Multi-Root Aware PCIe Capability Structure

- Fixed Header Bits – RO or Not Used
  - ✓ PCIe CAP ID
  - ✓ Next CAP Pointer
  - ✓ PCIe Capabilities Register
  - ✓ Device Capabilities
- Variable Header Bits – Selective Replication per Physical Function
  - ✓ Device Control
  - ✓ Device Status
  - ✓ Link Capabilities
  - ✓ Link Control
  - ✓ Link Status
  - ✓ Slot Capabilities
  - ✓ Slot Control
  - ✓ Slot Status

Virtual Plane Protocol

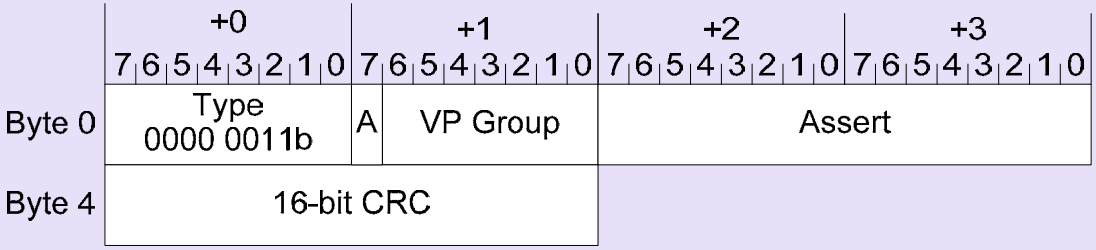
- TLP Tag
  - ✓ Inserted/removed at DLL
    - TLP is unmodified
    - TLP processing uses PCIe 1.X rules
  - ✓ Targeted for support of 256 Virtual Planes
    - Room for expansion to more VP or more functions
    - Devices may support from 1 to 256 VP
- RESET DLLP
  - ✓ Per plane RESET DLLP
    - Guaranteed progress under all topology conditions
    - TLP messages can stall due to FC congestion
  - ✓ Propagates using RESET logical rules
    - Provides a mirror of PCIe RESET within a plane

Tagging TLPs in Multi-Root



- Header for all TLPs on MR link
  - ✓ Not included on PCIe 1.X links
- Header included on all TLPs
  - ✓ Stable during retransmissions
- Located between Sequence # and TLP Hdr
- ACK / Sequence # concept remains per link
  - ✓ Not affected by Virtual Plane changes
  - ✓ Like Virtual Channels today
- Header covered by LCRC
  - ✓ VP# bits are not part of ECRC

RESET DLLP



A: 0 = Request
1 = Acknowledge

VP Group: upper bits of VP number representing a group of 16 VPs

Assert: Bit field representing the VPs within group of 16.
1 = Assert Hot Reset
0 = Deassert Hot Reset

- Provides RESET assert/clear for up to 16 VP in single DLLP
  - ✓ RESET function remains a level event
  - ✓ RESET state is stored and maintained by each link partner
- Handshake for complete reliability of RESET propagation
- Guarantees buffer flush of VH within intermediate switches
- Utilizes currently RSVD DLLP

Virtual Plane Operation

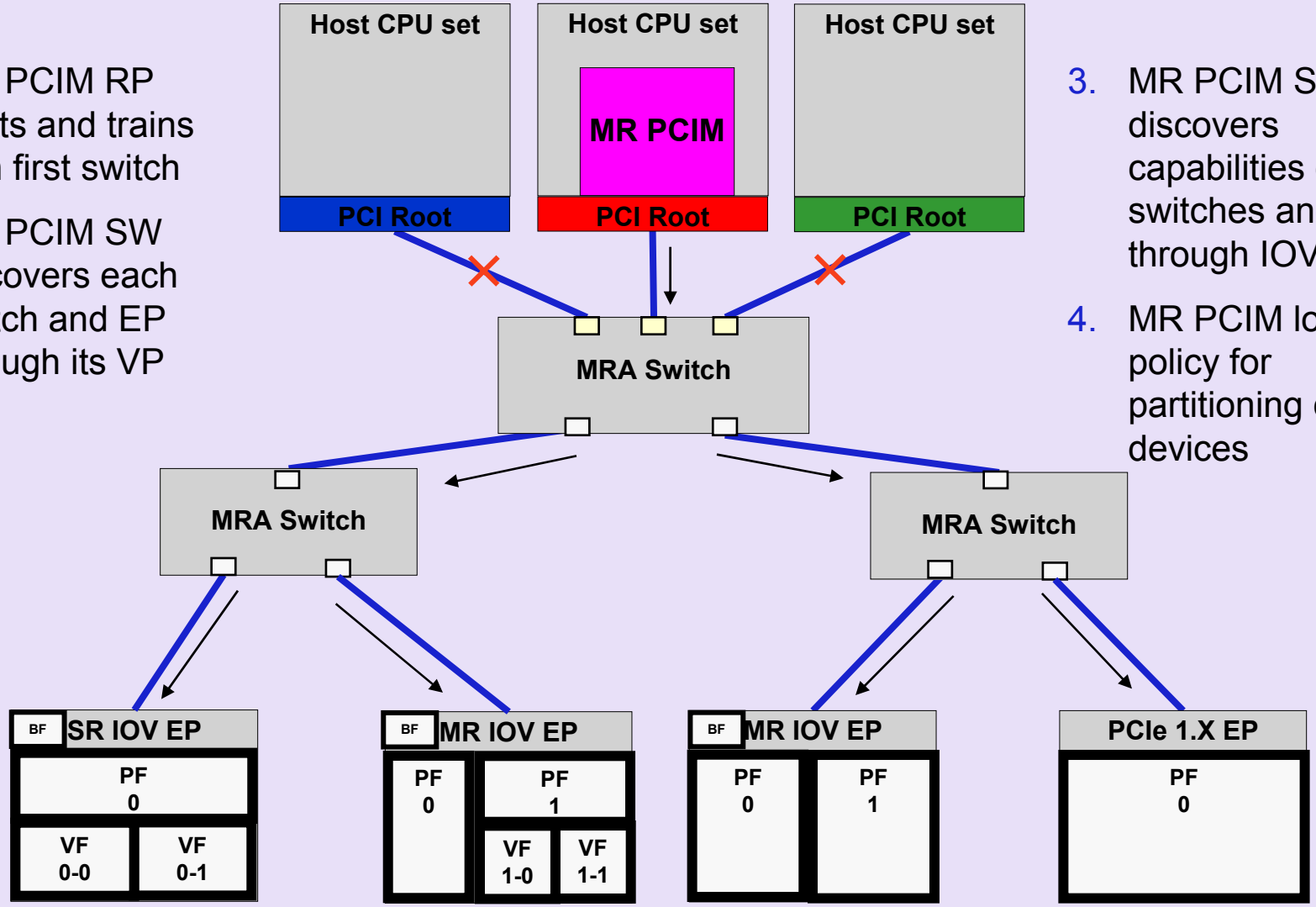
- Initialization & Enumeration
  - ✓ MR PCIM discovers MR, SR, and Base devices
  - ✓ MR PCIM devices and PFs to RPs
  - ✓ MR PCIM programs MR switch and EP tables with MR assignments
  - ✓ SR SW enumerates within its Virtual Hierarchies)
- Traffic Flow
  - ✓ Base RP & Base/SR EP utilize PCIe Base protocol
  - ✓ MR EP inserts VP tag at DLL for appropriate PF on Base TLP
  - ✓ Switch utilizes VP tag to index correct Type1 CFG headers
  - ✓ PCIe Base routing rules utilized within a Virtual Hierarchy
- RESET
  - ✓ Base RP asserts RESET as TS1 or Fundamental RESET
  - ✓ Switch propagates on MR link as RESET DLLP within a VP
  - ✓ Switch propagates on Base link as TS1
  - ✓ MR Switch or EP receiving RESET DLLP must flush according to PCIe rules

Protocol Selection

- MRA protocol usage must be enabled per link
  - ✓ Base, SR, and MR devices all coexist
- Base devices must work unchanged
  - ✓ New protocol should either be ignored or
  - ✓ Enabled only after both link partners are determined as capable
- Two options under consideration
  - ✓ Auto-negotiation at DLL
    - – Does not modify the PHY Layer training
  - ✓ SW enable
- Auto negotiation modifies the DLL training SM
  - ✓ New DLLPs used during DLL training to indicate capabilities of link partners
  - ✓ New DLLPs dropped by base devices during training
    - – Once dropped, link reverts to PCIe Base operation
- SW enable controlled by MR PCIM
  - ✓ MR PCIM uses PCIe Base protocol for initial discovery
  - ✓ MRA enabled during discovery and initialization by MR PCIM

MR PCIM Initialization

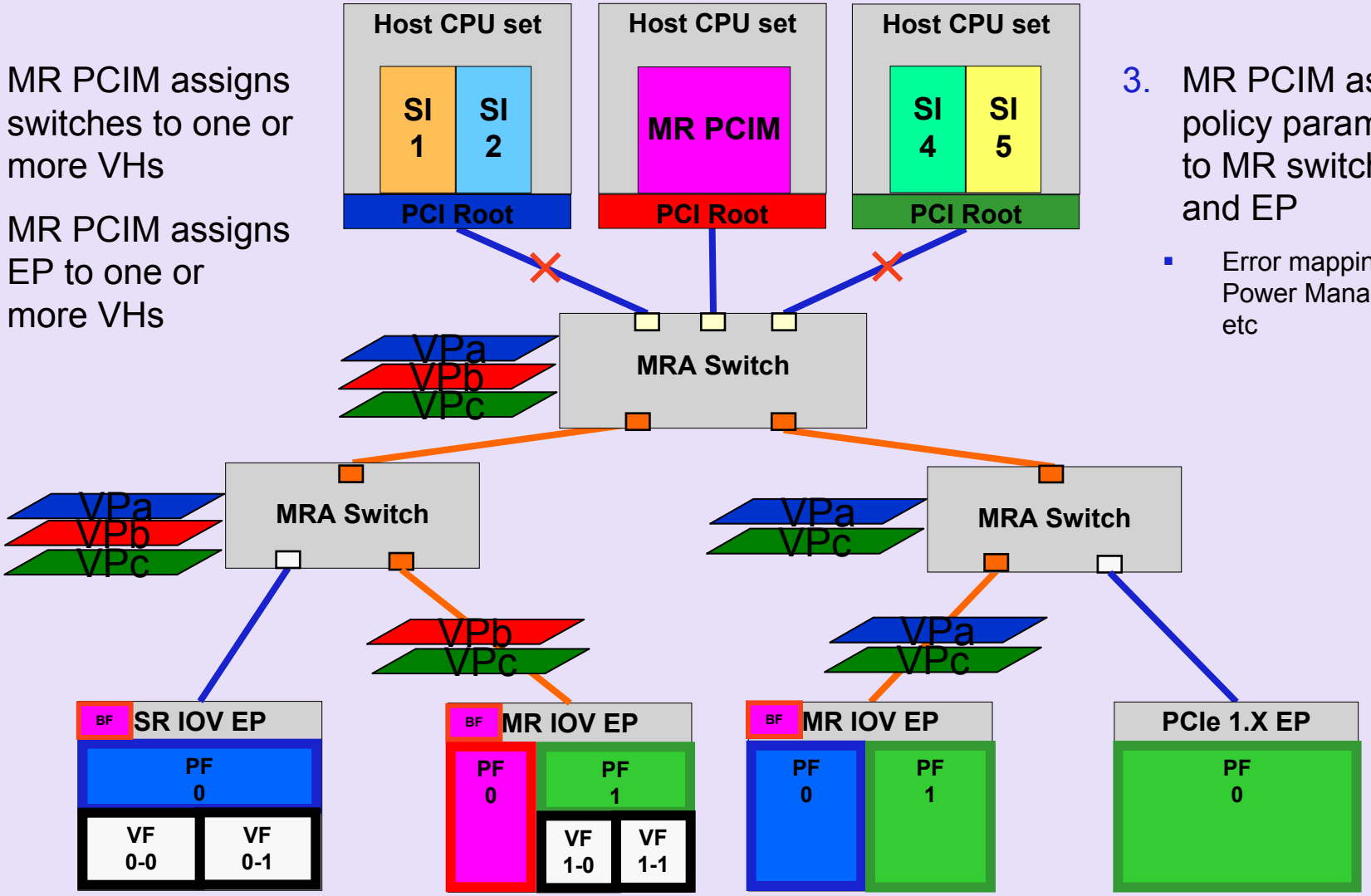
1. MR PCIM RP boots and trains with first switch
2. MR PCIM SW discovers each switch and EP through its VP



3. MR PCIM SW discovers capabilities of switches and EP through IOV CAP
4. MR PCIM loads policy for partitioning of devices

MR PCIM Assignment

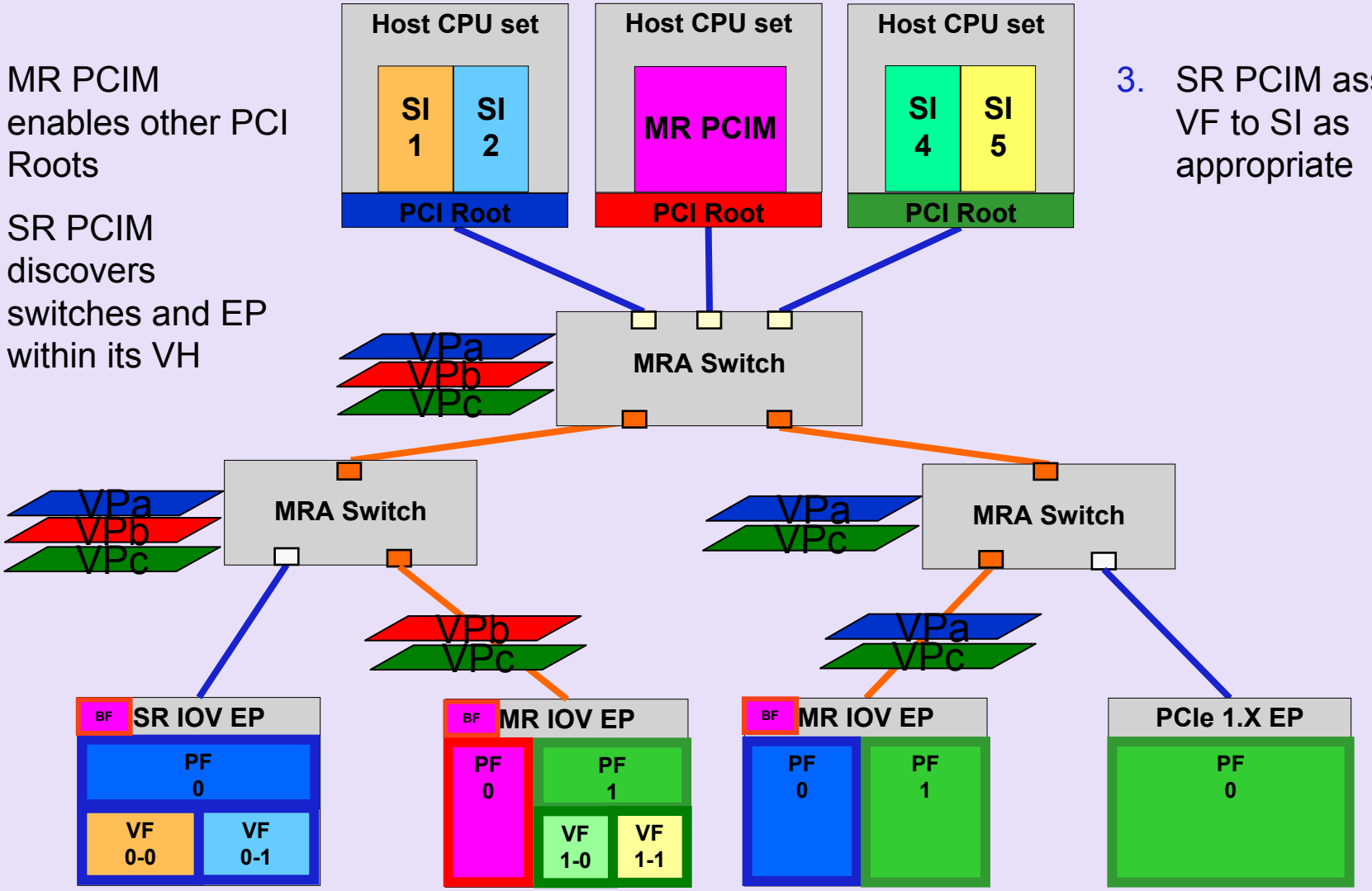
1. MR PCIM assigns switches to one or more VHs
2. MR PCIM assigns EP to one or more VHs



3. MR PCIM assigns policy parameters to MR switches and EP
   - Error mapping, Power Management, etc

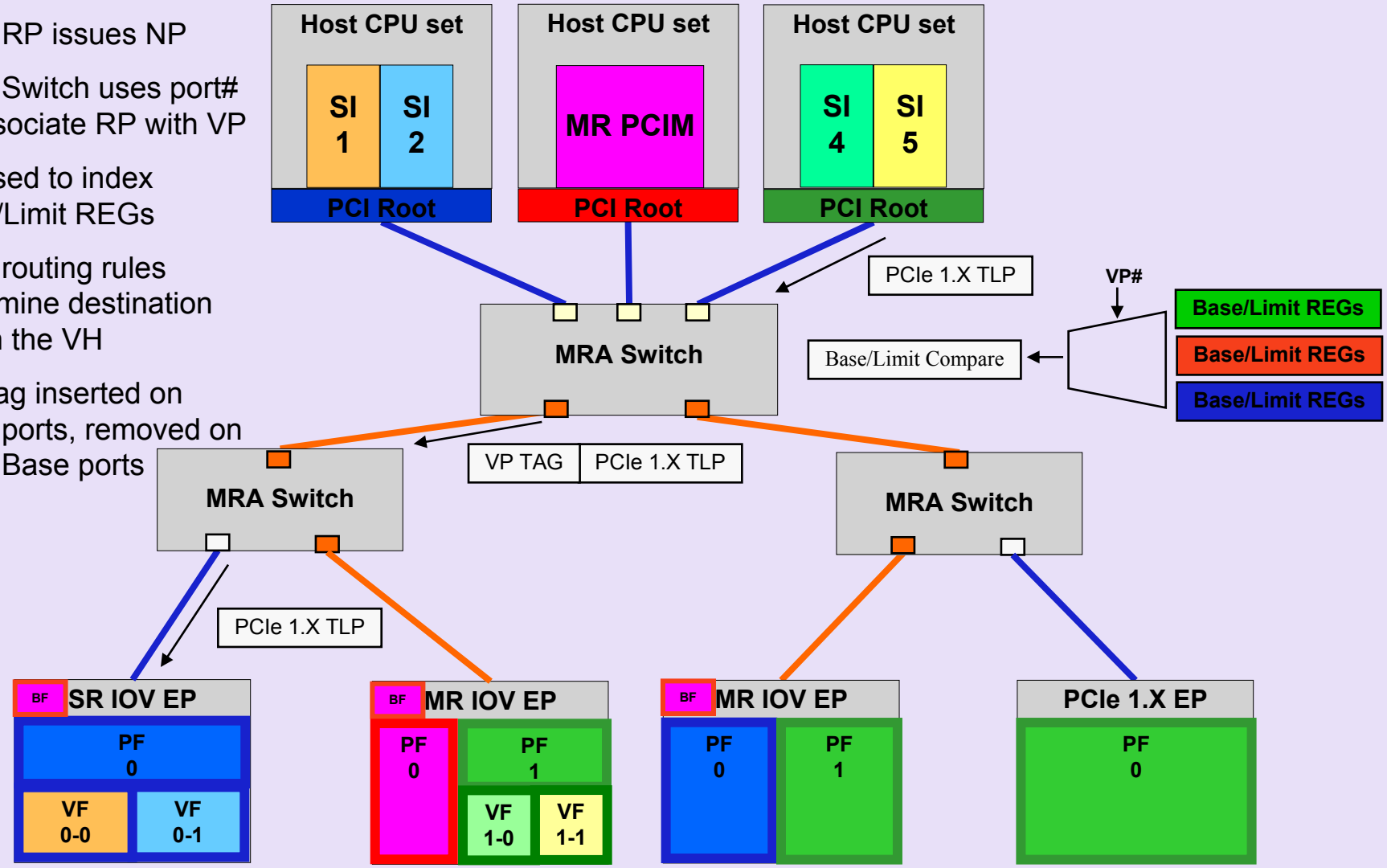
SR PCIM Assignment

1. MR PCIM enables other PCI Roots
2. SR PCIM discovers switches and EP within its VH
3. SR PCIM assigns VF to SI as appropriate



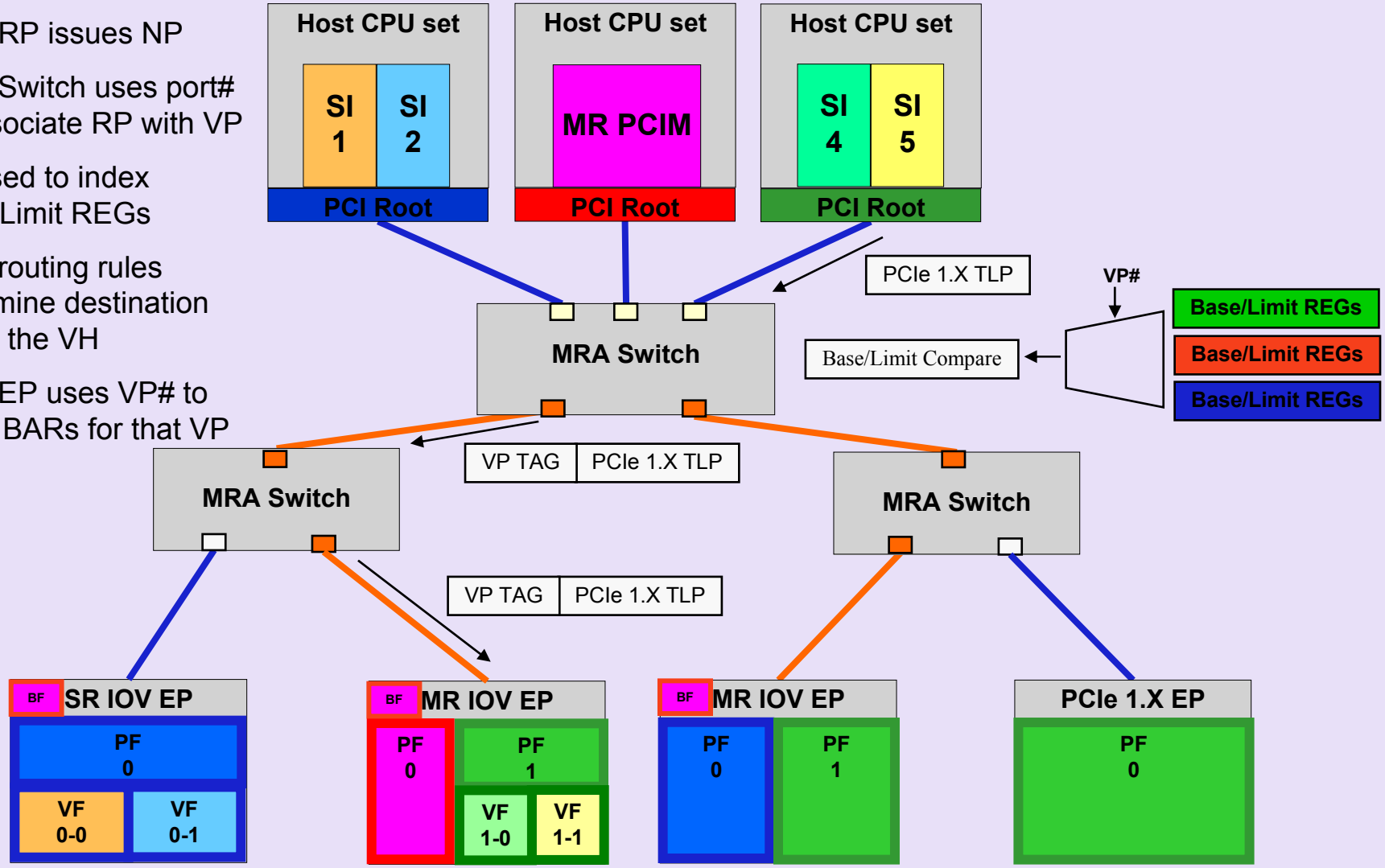
Non-Posted (NP) Request from PCIe Base to PCIe Base

1. PCIe RP issues NP
2. MRA Switch uses port# to associate RP with VP
3. VP used to index Base/Limit REGs
4. PCIe routing rules determine destination within the VH
5. VP Tag inserted on MRA ports, removed on PCIe Base ports



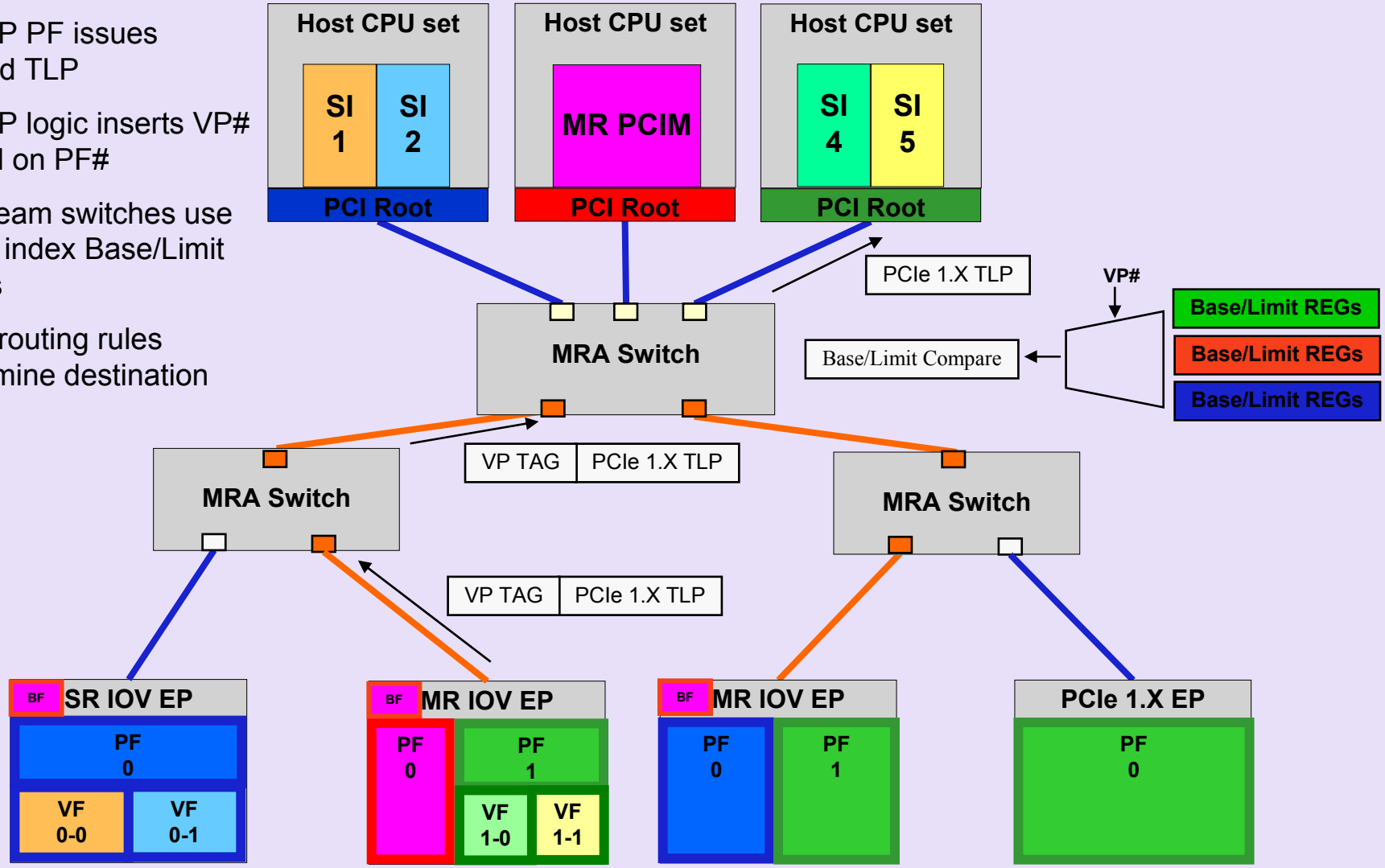
NP from PCIe Base to MRA

1. PCIe RP issues NP
2. MRA Switch uses port# to associate RP with VP
3. VP used to index Base/Limit REGs
4. PCIe routing rules determine destination within the VH
5. MRA EP uses VP# to index BARs for that VP



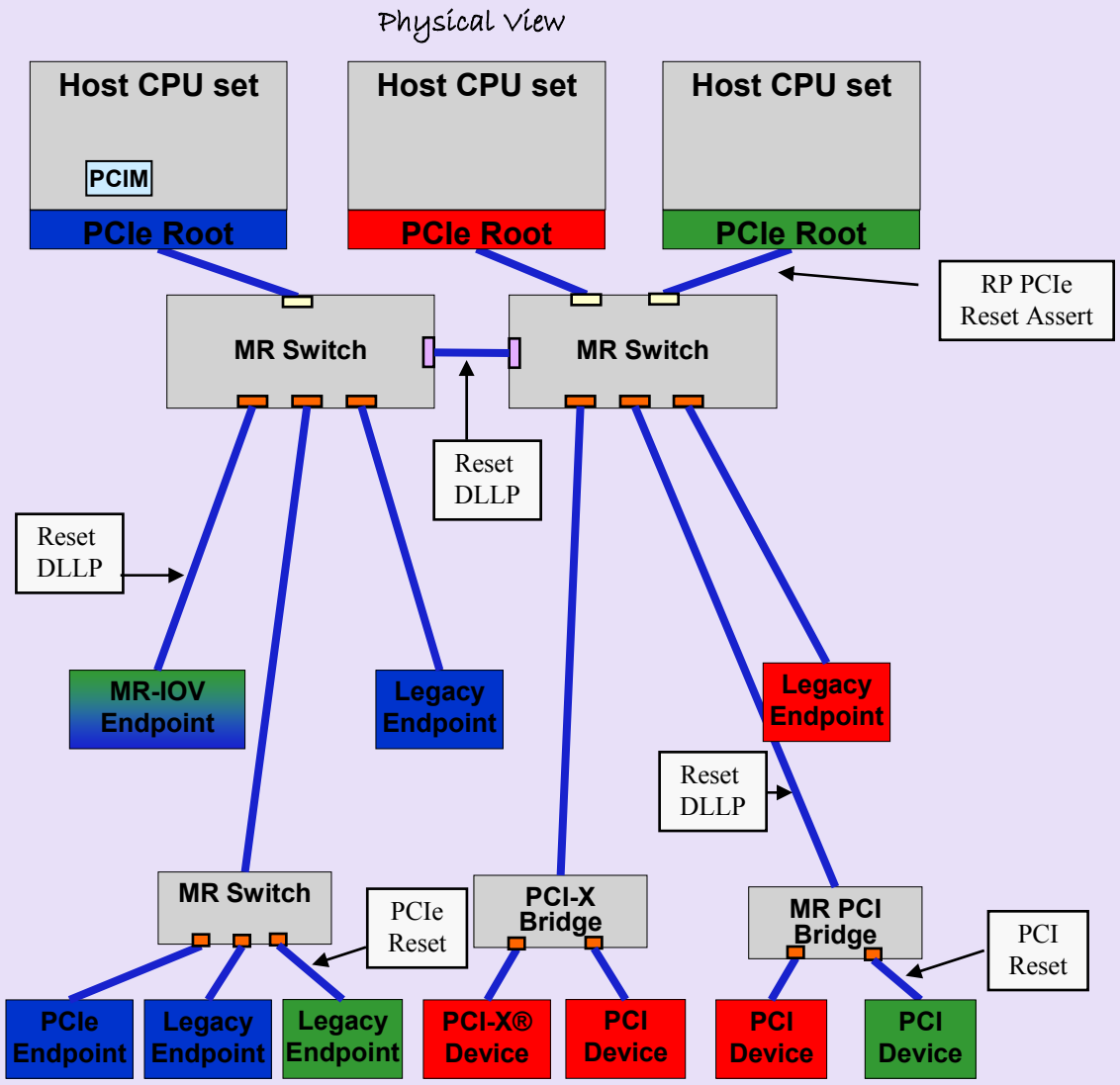
Posted from MRA to PCIe Base

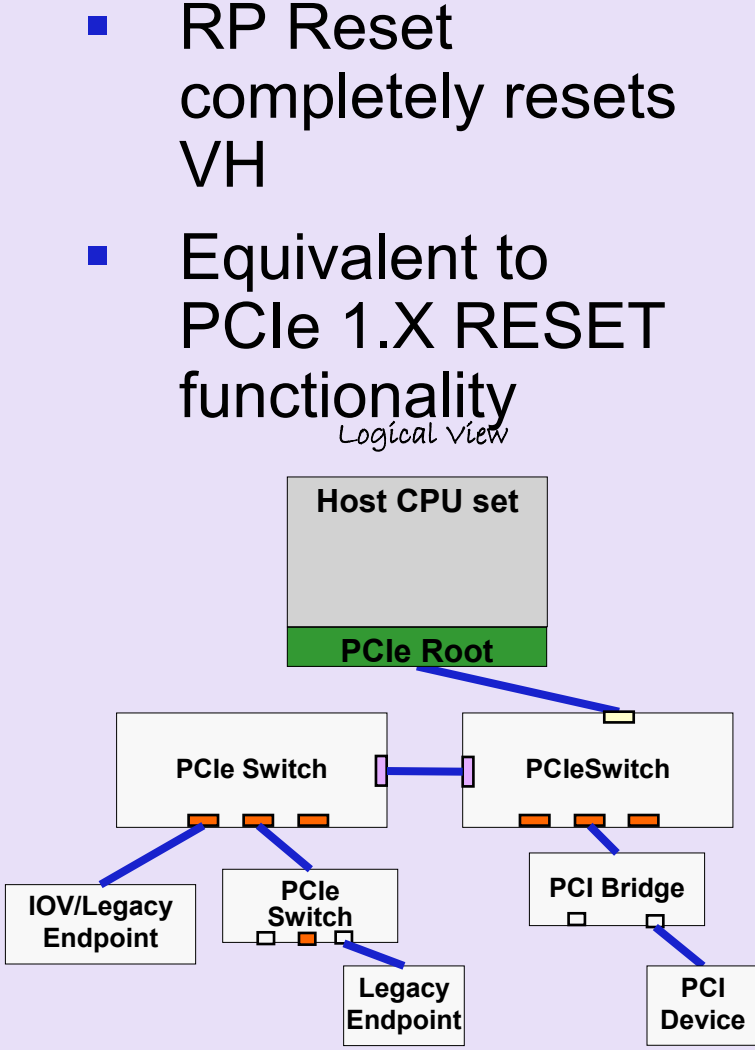
1. MR EP PF issues Posted TLP
2. MR EP logic inserts VP# based on PF#
3. Upstream switches use VP to index Base/Limit REGs
4. PCIe routing rules determine destination



RESET Propagation

- RP Reset completely resets VH
- Equivalent to PCIe 1.X RESET functionality

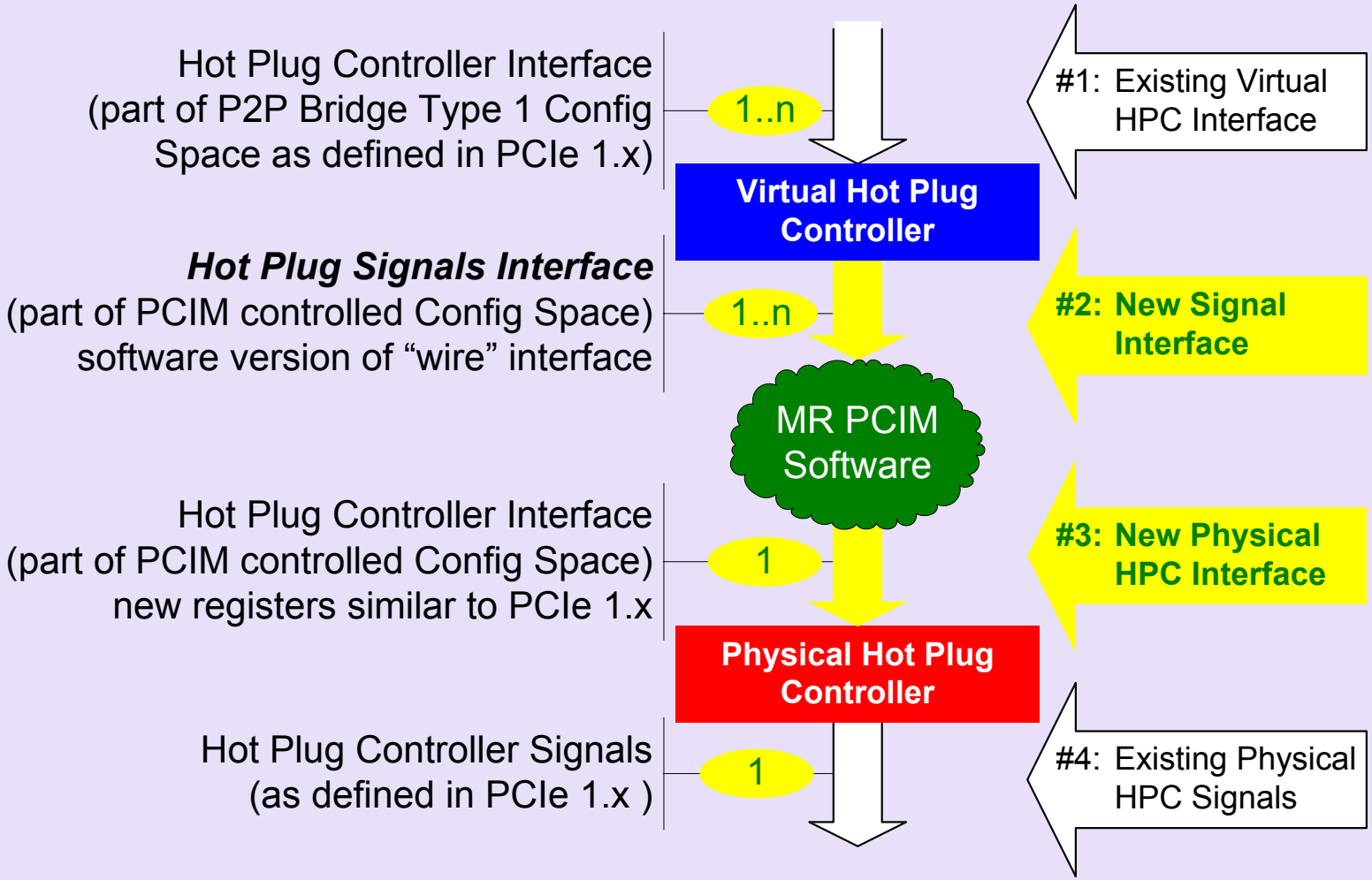




Hot Plug in Multi-Root

- MRA Switches have two Hot Plug Controllers (HPC)
  - ✓ Physical controller (1 per link)
    - – Controls events on the link
    - – Owned by MR PCIM
  - ✓ Virtual controller (1 per VP per link)
    - – Controls events within a VH
    - – Owned by SR PCIM, coordinated with MR PCIM
      - • New registers added for MR PCIM access point
- Hot Plug in MR consists of two event types
  - ✓ Physical events coordinated by MR PCIM and physical HPC (pHPC)
  - ✓ Virtual events coordinated by MR PCIM, SR PCIM, and virtual HPC (vHPC)
- HPC SW interface same as PCIe Base
  - ✓ Utilizes PCIe Hot Plug specification within switches

MR Hot Plug Interfaces

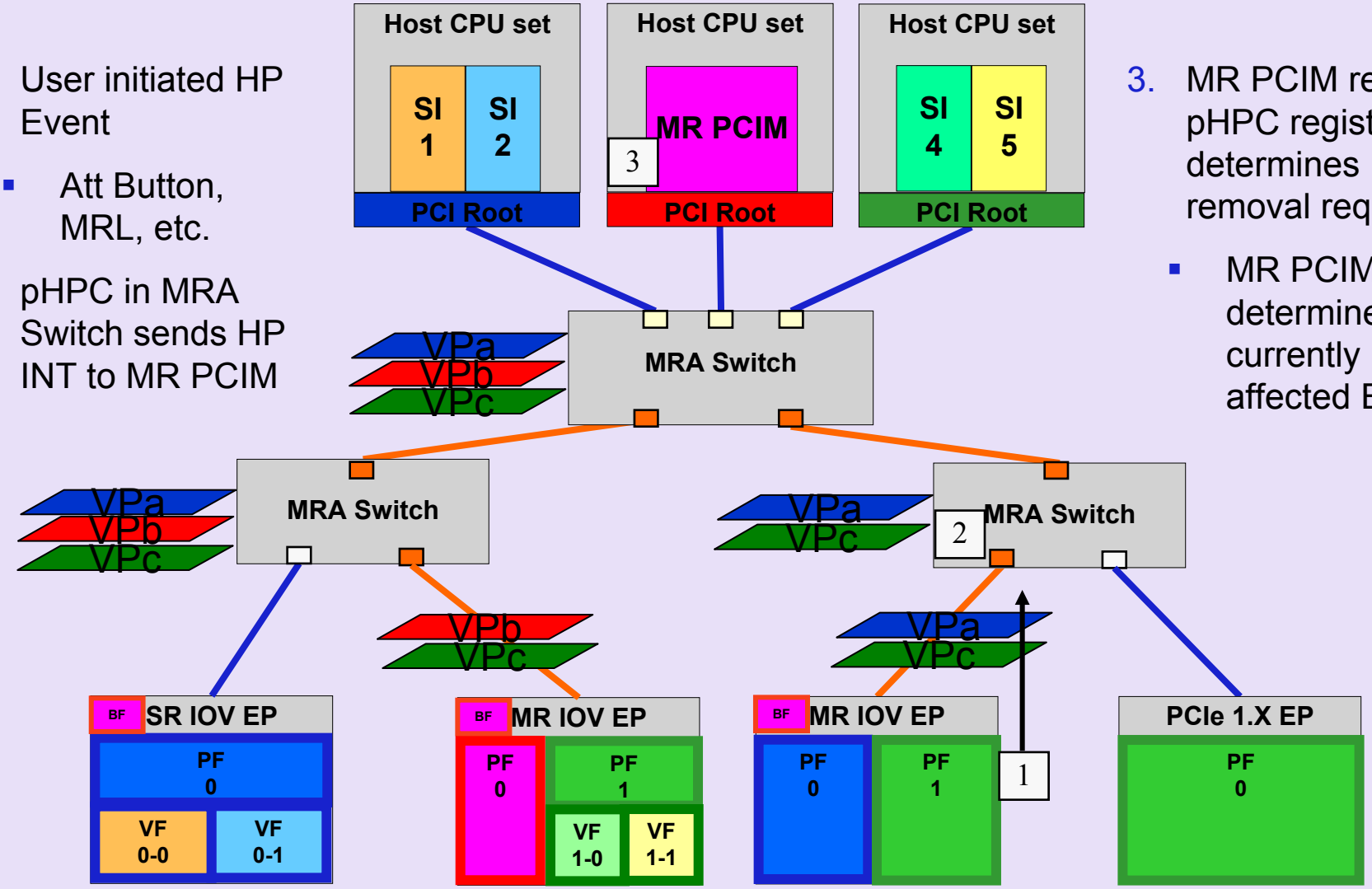


Hot Plug Event Steps

- User event initiated
  - ✓ Example: Slot Attention Button Press for device removal
- pHPC notifies MR PCIM of event
  - ✓ Uses MR PCIM pHPC interface (#3)
- MR PCIM initiated virtual HP events through vHPC
  - ✓ Uses MR PCIM vHPC interface (#2)
- vHPC notifies SR PCIM(s) of event
  - ✓ Uses SR PCIM vHPC interface (#1)
- SR PCIM processes event and updates state of vHPC
  - ✓ Uses SR PCIM vPHC interface (#1)
- vHPC notifies MR PCIM of state change for that vHPC
  - ✓ Uses MR PCIM vHPC interface (#2)
- MR PCIM processes all state changes and updates state of pHPC
  - ✓ Uses MR PCIM pHPC interface (#3)
- pHPC indicates device is safe for removal
  - ✓ Example: Slot power removed & LED state change

Hot Removal Event Initiated

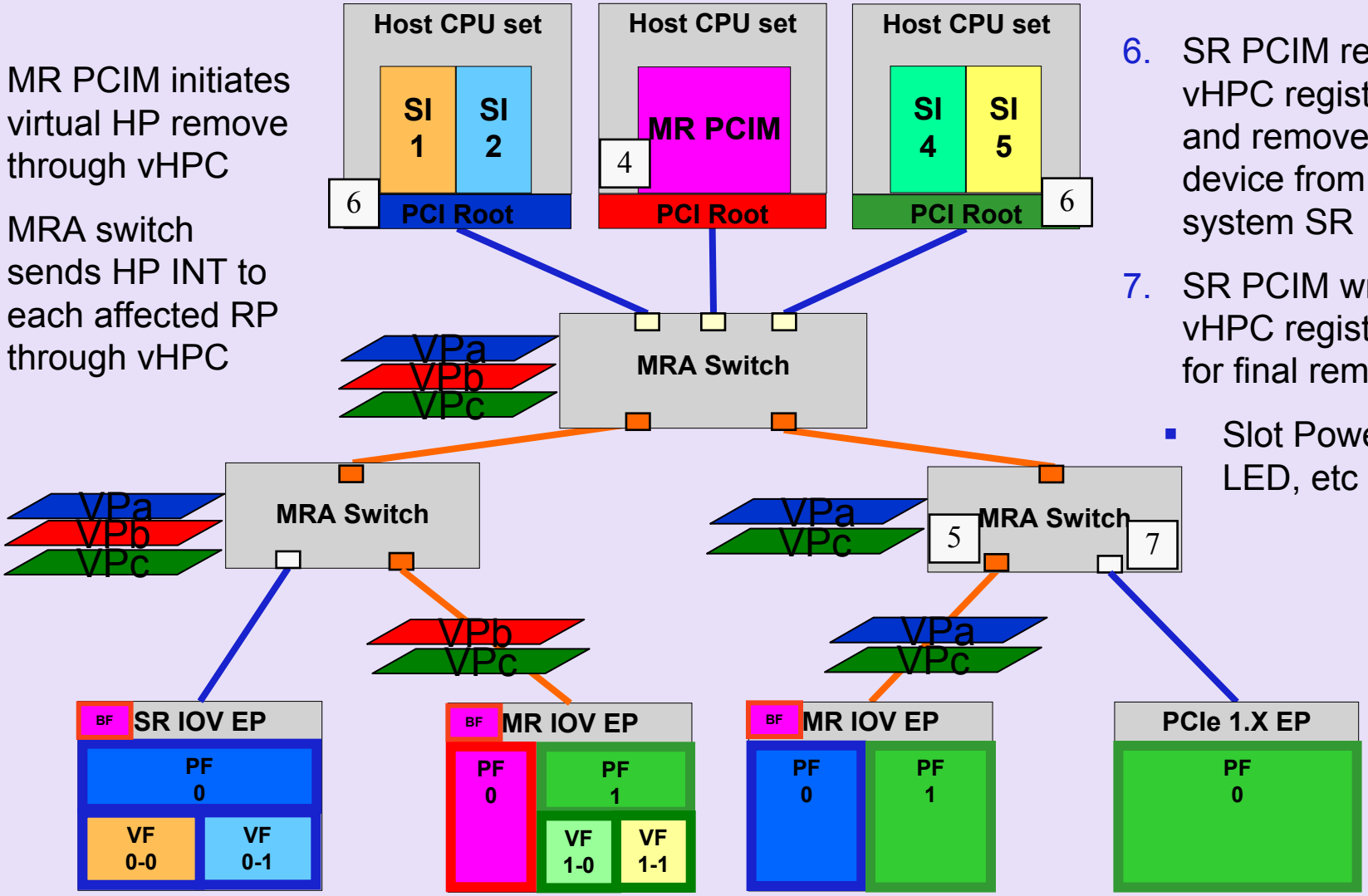
1. User initiated HP Event
   - Att Button, MRL, etc.
2. pHPC in MRA Switch sends HP INT to MR PCIM



3. MR PCIM reads pHPC registers & determines removal required
   - MR PCIM determines VH currently using affected EP

Hot Removal Coordination

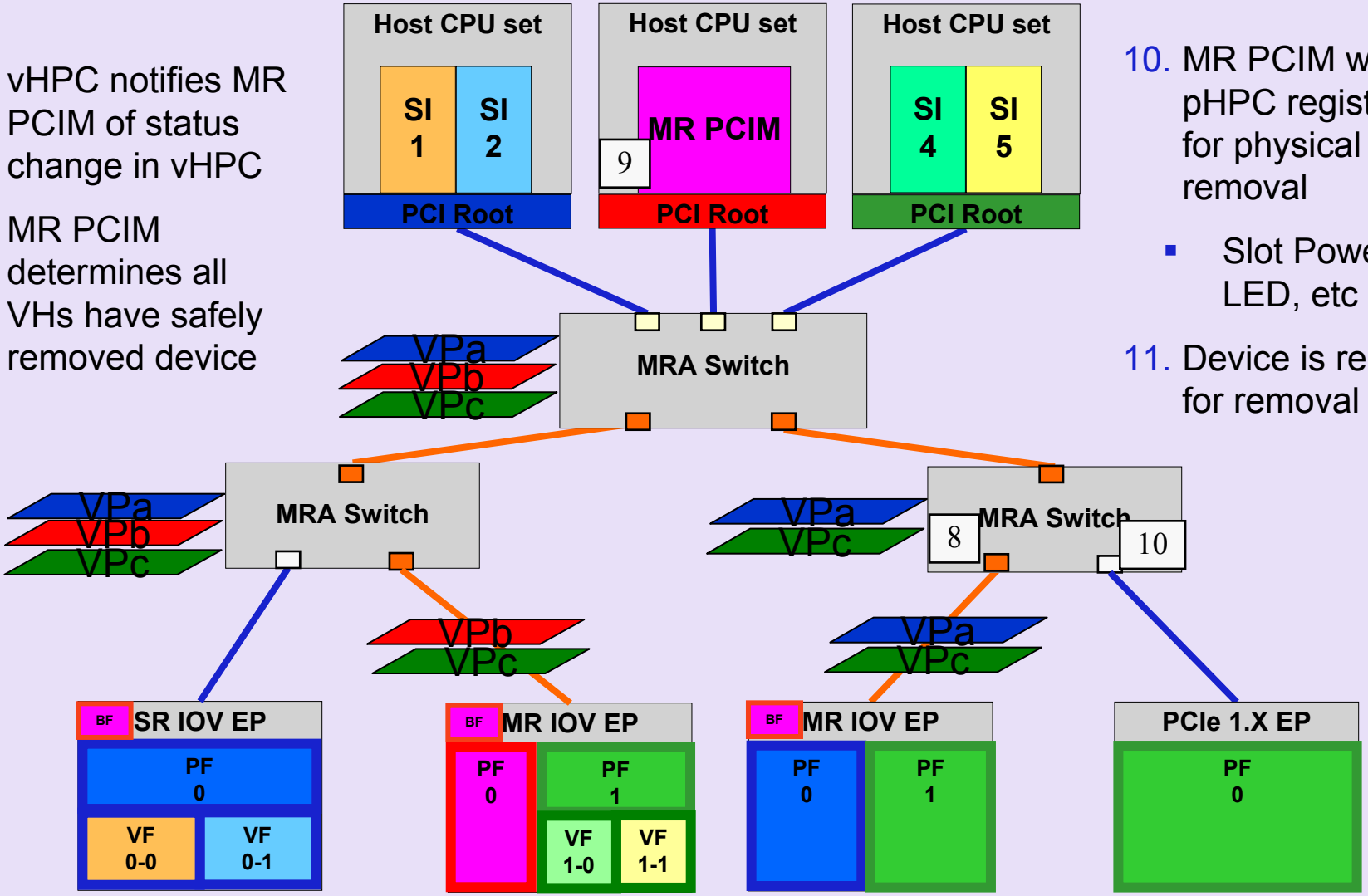
4. MR PCIM initiates virtual HP remove through vHPC
5. MRA switch sends HP INT to each affected RP through vHPC



6. SR PCIM reads vHPC registers and removes device from system SR
7. SR PCIM writes vHPC registers for final removal
   - Slot Power, LED, etc

Hot Removal Completion

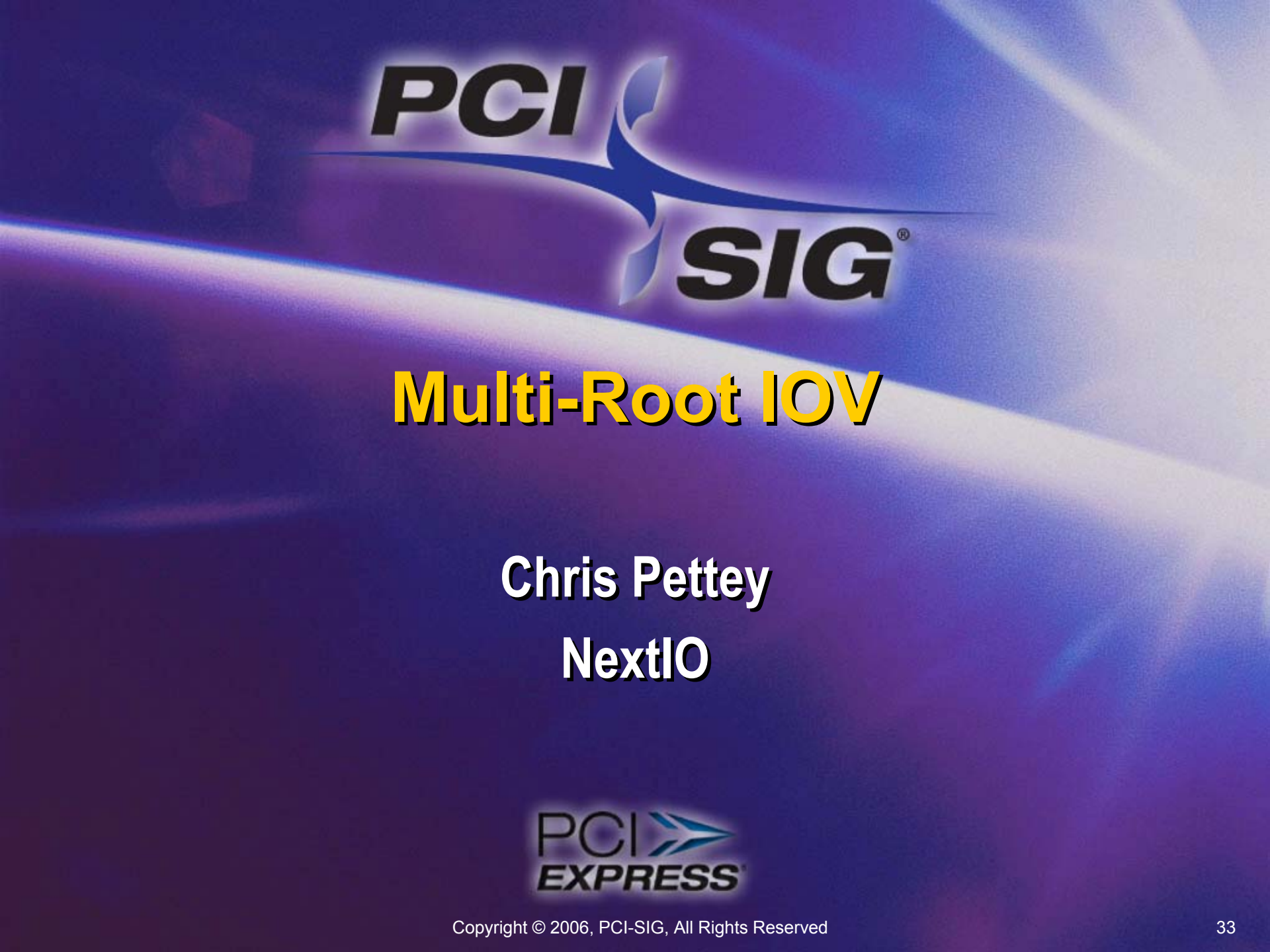
8. vHPC notifies MR PCIM of status change in vHPC
9. MR PCIM determines all VHs have safely removed device



10. MR PCIM writes pHPC registers for physical removal
    - Slot Power, LED, etc
11. Device is ready for removal

Thank you for attending the
PCI-SIG Developers Conference 2006.

For more information please go to
www.pcisig.com



Multi-Root IOV

Chris Pettey

NextIO