Single Root IO Virtualization

David Kahn (Sun Microsystems)

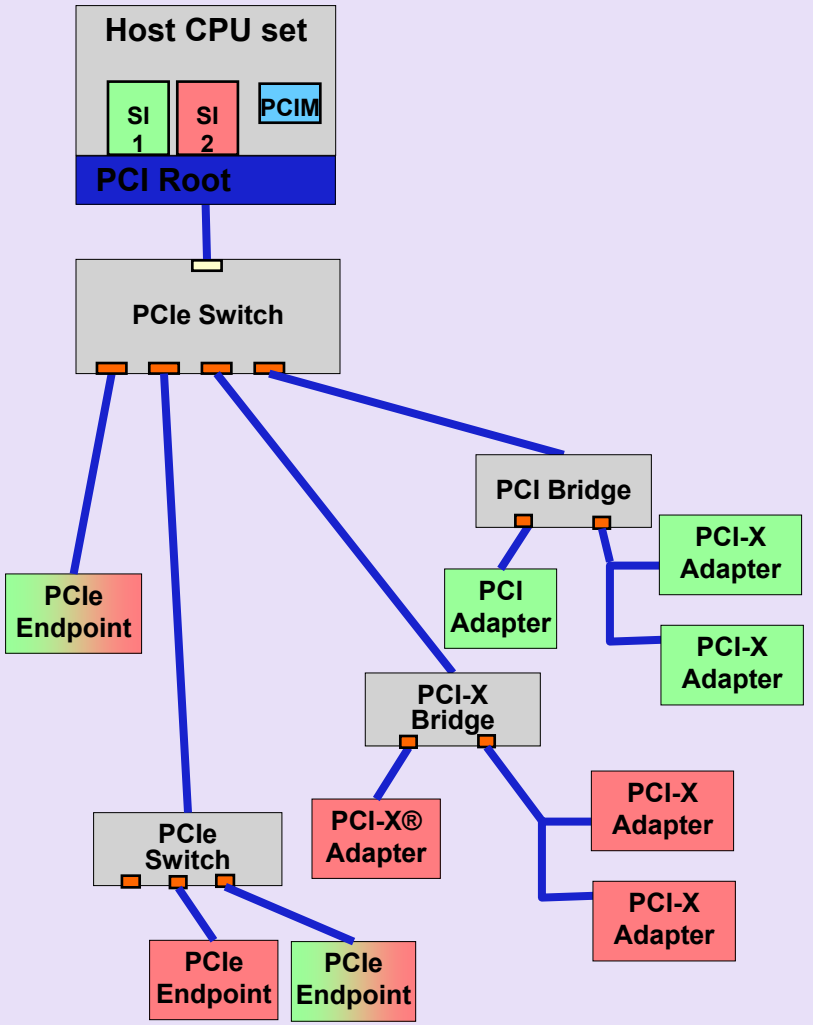
Work In Progress

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.

Outline

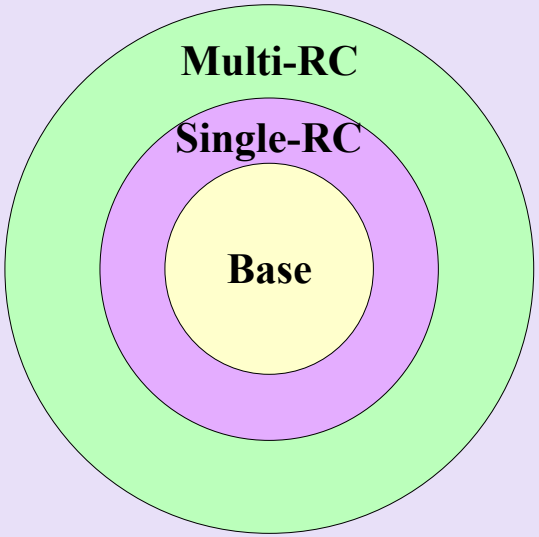
- Single Root Overview
- Single Root Resource Allocation
- SR PCIM – VF Creation and addressing
- System Image use of devices
- Single Root Configuration Space Details

Single Root Overview



- A single Root Complex with multiple System Images sharing SR-IOV aware devices.
- A single root fabric consists of a single set of PCI address spaces (just like PCI Express® base)
- A VI is required to manage access to the fabric (permissions, etc.)

Single Root Overview (Cont.)



- SR is built on the PCI-Express base protocol.
- SR requires no changes to the root complex or the PCI Express fabric.
- Some implementations may decide to include some optional changes to switches and possibly the root complex to implement SR. (examples: ARI, ATPT). Note: ATPT is not specified or required by any IOV specification.
- Changes to CPU complex to support virtualization. (protection, etc.) Note: CPU changes to support virtualization are not specified by the IOV specifications.

SR Overview – PF/VF

- Physical Function (PF)
  - ✓ Simply, a name for a PCI function as defined by the base spec.
  - ✓ An SR iov-aware PF contains IOV capabilities for configuration and management of the PF.
  - ✓ Used by SR PCIM to manage a set of virtual functions.
- Virtual Function (VF)
  - ✓ Simply, a name for a virtual view of the device.
  - ✓ Used by SIs to access resources on the endpoint.
  - ✓ The VF is created/managed by SR-PCIM
  - ✓ The VF is associated with a single PF
  - ✓ Once created, it can be probed and accessed through the root complex using normal access methods.

SR Overview – SR PCIM

- SR PCI Manager (SR-PCIM)
  - ✓ The entity responsible for configuration and management of an iov-enabled fabric and devices.
  - ✓ Creates and manages VFs
  - ✓ Handles events that cannot be associated with a single VF/SI.

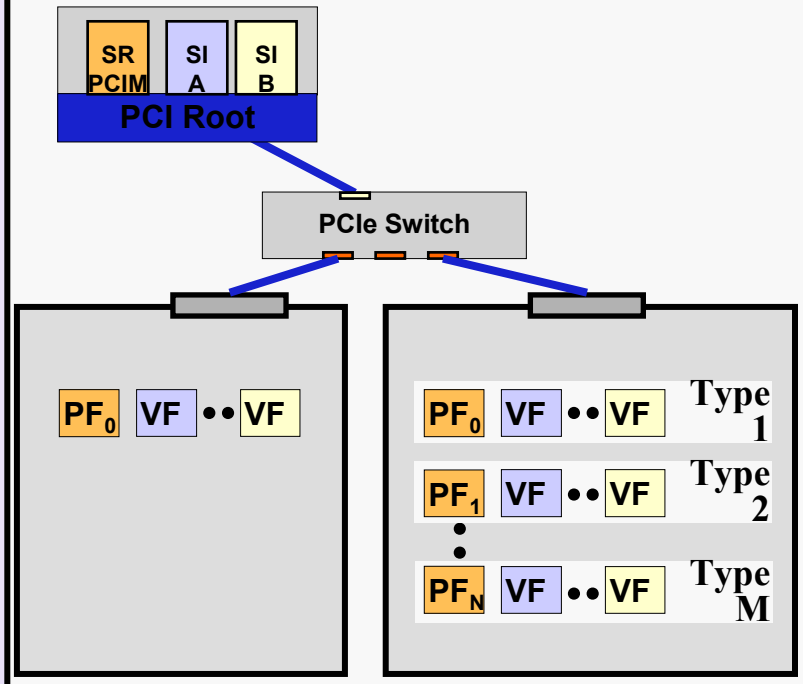
SR Overview – Roll of the VI

- Provides protection between SIs
  - ✓ AKA hypervisor, etc.
  - ✓ Physical resources (memory, devices, privileged registers)
  - ✓ PCI resources (memory, io, config space)
  - ✓ DMA addresses
  - ✓ Routing of messages (Interrupts, etc)
  - ✓ Can be (and usually will be) a combination of software and hardware

SR Resource Allocation

Single-Root IOV Function Types and Terms

SR Topology



Terms

For SR topologies,

Physical Function is used by SR-PCIM to manage a set of Virtual Functions.

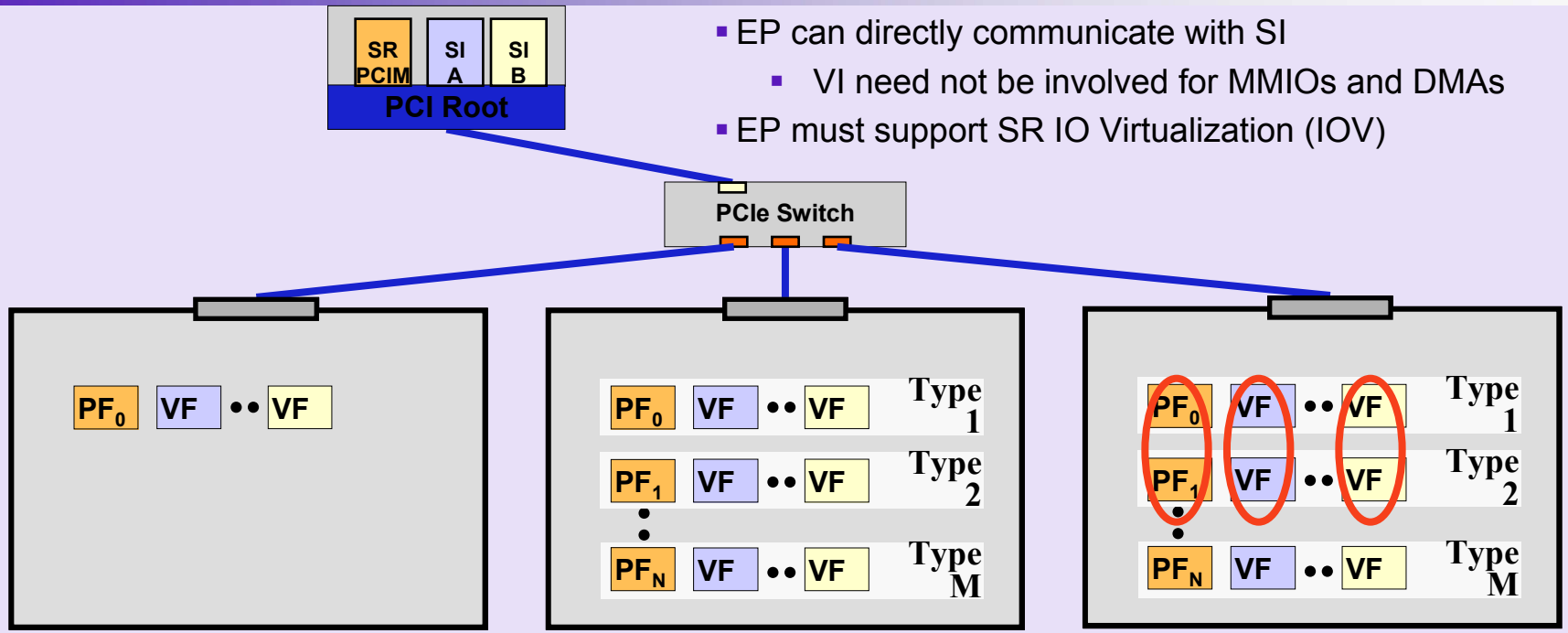
Physical Function 0 (PF_0) is also used to manage EP functions, such as physical errors and events.

Virtual Function is used by SIs to access resources on the EP.

More details on function assignment, capabilities, etc… later.

EP enabled for SR IO Virtualization

- EP can directly communicate with SI
  - VI need not be involved for MMIOs and DMAs
- EP must support SR IO Virtualization (IOV)

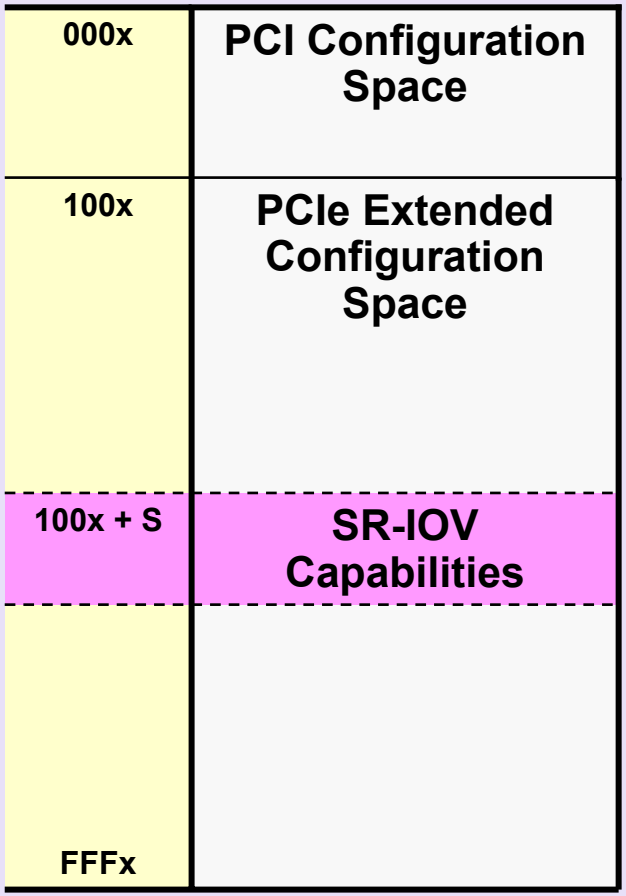


- Use by EP with one PF
- PF describes VF capabilities, i.e.:
  - ✓ Maximum number of VFs.
  - ✓ … *more later*

- Use by EP with multiple independent functions
- Each PF describes its VF capabilities, i.e.:
  - ✓ Maximum number of VFs for PF type.
  - ✓ … *more later*

- Use by EP with multiple dependent functions
- Each PF describes its VF capabilities, i.e.:
  - ✓ Maximum number of VFs for PF type.
  - ✓ Combinations
  - ✓ … *more later*

Configuration Space Overview



- SR IOV adds a new, optional PCIe® Extended Capability for each Physical Function that supports SR-IOV.

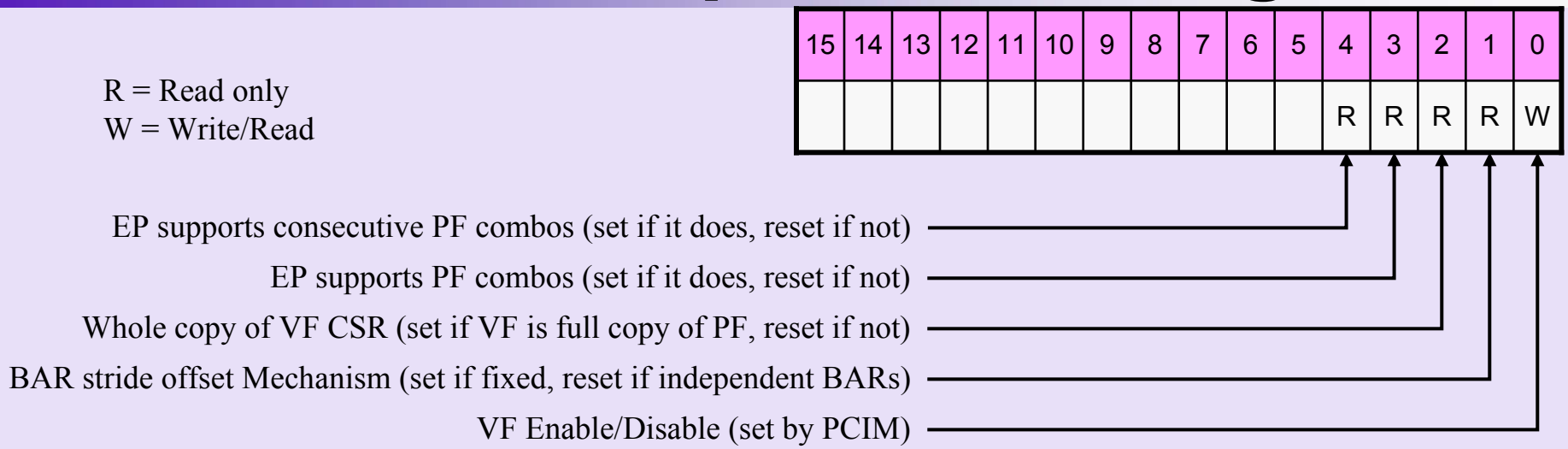
✓ SR-IOV Capability - used by SR-PCIM to:
- Determine EP is SR-IOV capable
- Enable/disable and configure the EP's SR-IOV capability

SR-IOV Capabilities

SR-IOV PCIe Extended Configuration Space		
SR-IOV Capabilities	Next Cap Ptr	Version
RID Space Allocation Register(s) (TBD by RID subteam)		
Mapping Dependent Fields (covered on the slides that follow)		

- IOV Capabilities; Next Capabilities Pointer; Version
- RID Space Allocation Register(s) to be defined by RID subteam
- Mapping Dependent Fields - Contents differ depending on which EP architecture approach is used:
  - ✓ EPs with a single PF type
  - ✓ EPs with multiple independent PF types
  - ✓ EPs with multiple dependent PF combinations

SR-IOV Capabilities Register



- VF Enable/Disable - Used by PCIM to set SR-IOV mode.
- BAR Stride Offset Mechanism
  - ✓ Set if EP supports the same, fixed offset for all the VFs' BAR.
  - ✓ Reset if EP supports a Independent BARs for each VF.
- Whole copy of VF - Set if VFs share **all fields** with their associated PF.
- EP supports PF combos - Set if EP supports dependent functions.
- EP supports consecutive PF combos - For an EP with dependent functions, set if dependent functions must be sequentially assigned.

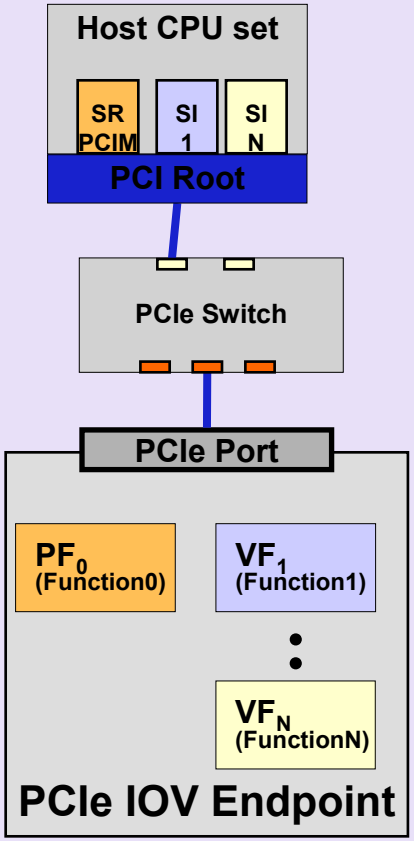
EPs with one VF type

SR-IOV PCIe Extended Configuration Space	
:	
Max VFs (RO)	Num VFs (RW)
Max Stride (RO)	
Min Stride (RO)	
Stride (RW)	

- Max VFs - Defines the maximum number of VFs supported by the EP.
- Num VFs - The number of VFs assigned to the PF by SR-PCIM (must be equal to or smaller than Max VFs.
- If EP supports fixed BAR stride:
  - ✓ Max Stride* - Defines the maximum BAR offset for all VFs.
  - ✓ Min Stride* - Defines the minimum BAR offset for all VFs (vendor defined minimum amount of MMIO space needed by each VF).
  - ✓ Stride - Defines the BAR offset for all VFs
  - ✓ Min Stride < Stride < Max Stride and all Stride fields must be a power of 2.
  - ✓ *For EPs with multiple BARs, the fixed stride applies to all BARs.*
- *If the EP supports Independent BARs, then each VF has full BAR(s) and the Stride fields are not used.*

*Note: each of these fields has a documented limit in the specification, which EPs must support.

Example of an EP with fixed Stride & one VF Type in an SR-IOV Topology



SR-IOV PCIe Extended Configuration Space		
SR-IOV Capabilities	Next Cap Ptr	Version
Max VFs	Num VFs	
Max Stride		
Min Stride		
Stride		

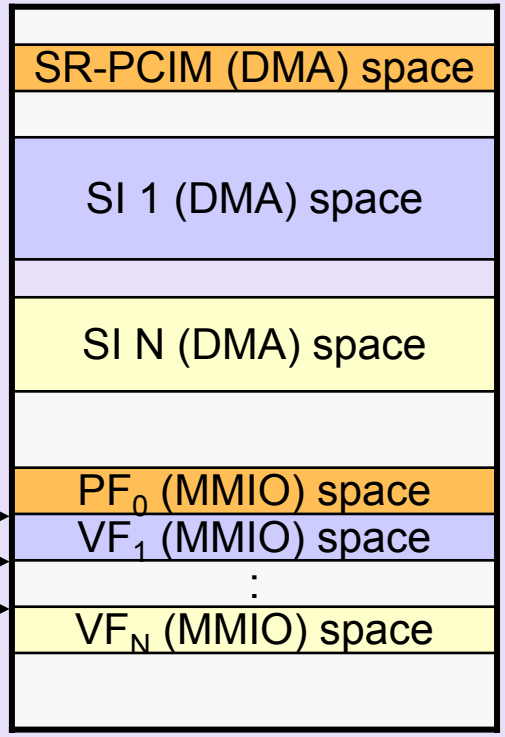
The Assigned Stride equals the offset into the PF BAR for each VF, such that:

- **VF_1 starts at base BAR + Stride**
- **VF_2 starts at base BAR + [(2) x (Stride)]**

…

- **VF_N starts at base BAR + [(N) x (Stride)]**

Sample PCI Address Layout



Implementation note: Firmware can use the Max VFs and Max Stride fields to calculate the amount of MMIO space to allocate to the EP. For example, the MMIO space for the EP's port could be set to: (Max Stride) x (Max VFs + 1).

EPs with multiple, independent VF types

SR-IOV PCIe Extended Configuration Space		
SR-IOV Capabilities	Next Cap Ptr	Version
Max VFs	Num VFs	VF Offset
Max Stride		
Min Stride		
Stride		

- Fields for an EP with multiple, independent VF types have the same definitions as for an EP with one VF type. One additional field is used:
  - ✓ VF Offset - Virtual function pointer (see next page).

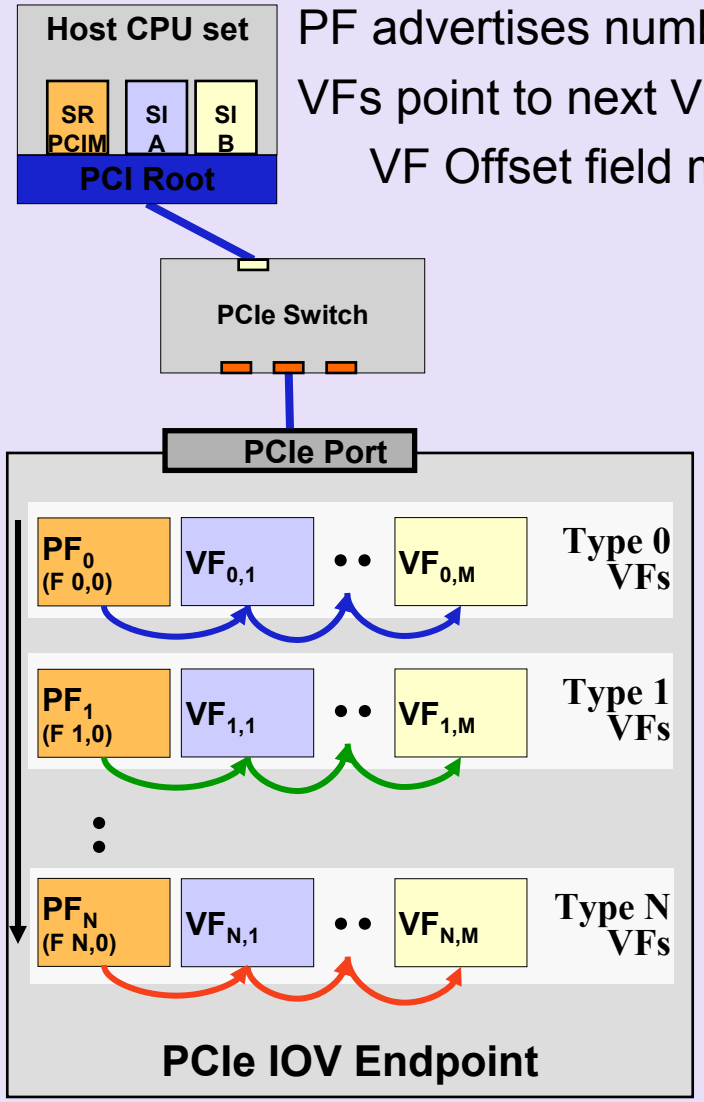
- Note: each VF type must have a physical function.
  - ✓ The PF must contains the above SR-IOV PCIe Extended Configuration Space that describes the capabilities of the VF instances of the PF.

Overview of EPs with multiple, independent VF types

PF advertises number of VFs and points to 1st VF

VFs point to next VF – advertise relative offset:

VF Offset field must be in the VF's CSRs.



Configuration Space

$PF_{0,0}$	$NumVFs_{0,0}$	VF Offset
$PF_{1,0}$	$NumVFs_{1,0}$	VF Offset
:		
$PF_{N,0}$	$NumVFs_{N,0}$	VF Offset
$VF_{0,1}$ space		VF Offset
:		
$VF_{0,M}$ space		VF Offset
$VF_{N,M}$ space		VF Offset
:		
$VF_{1,1}$ space		VF Offset
:		
$VF_{0,M}$ space		VF Offset
:		
$VF_{1,M}$ space		VF Offset

Legend

- Type 1
- Type 2
- ..
- Type M

EPs with multiple, fixed interdependent functions

SR-IOV PCIe Extended Configuration Space	
SR-IOV Capabilities	Next Cap Ptr
RID Space Allocation Register(s) (TBD by RID subteam)	
Max VFs	Num VFs
Nominal Combo Number (RW)	
Combo Instance Number (RW)	
Supported Combinations (RO)	
$Stride_1$	
:	
$Stride_N$	

- IOV Capabilities; Next Capabilities Pointer; Version; and RID Space, Max VFs, and Num VFs are as defined before.
- Nominal Combo Number - Describes the combination to be used by the Function.
- Combo Instance Number - A unique identifier for the specific combination instance the Function is in.
- Supported Combinations - Function truth table that describes which function combinations the EP supports.
- For each VF, Stride fields are as defined previously.

Supported Combinations Combo Truth Table

Nominal Combo Number	$Type_N$	$Type_{N-1}$	. . .	$Type_1$	$Type_0$	Supported Combinations
0	0	0		0	0	0
1	0	0		0	1	0/1
2	0	0		1	1	0/1
3	0	0		0	0	0/1
…						
X-2	1	1		0	0	0/1
X-1	1	1		0	1	0/1
X	1	1		1	1	0/1

Function Types Supported by the EP

EP supported combinations are 1

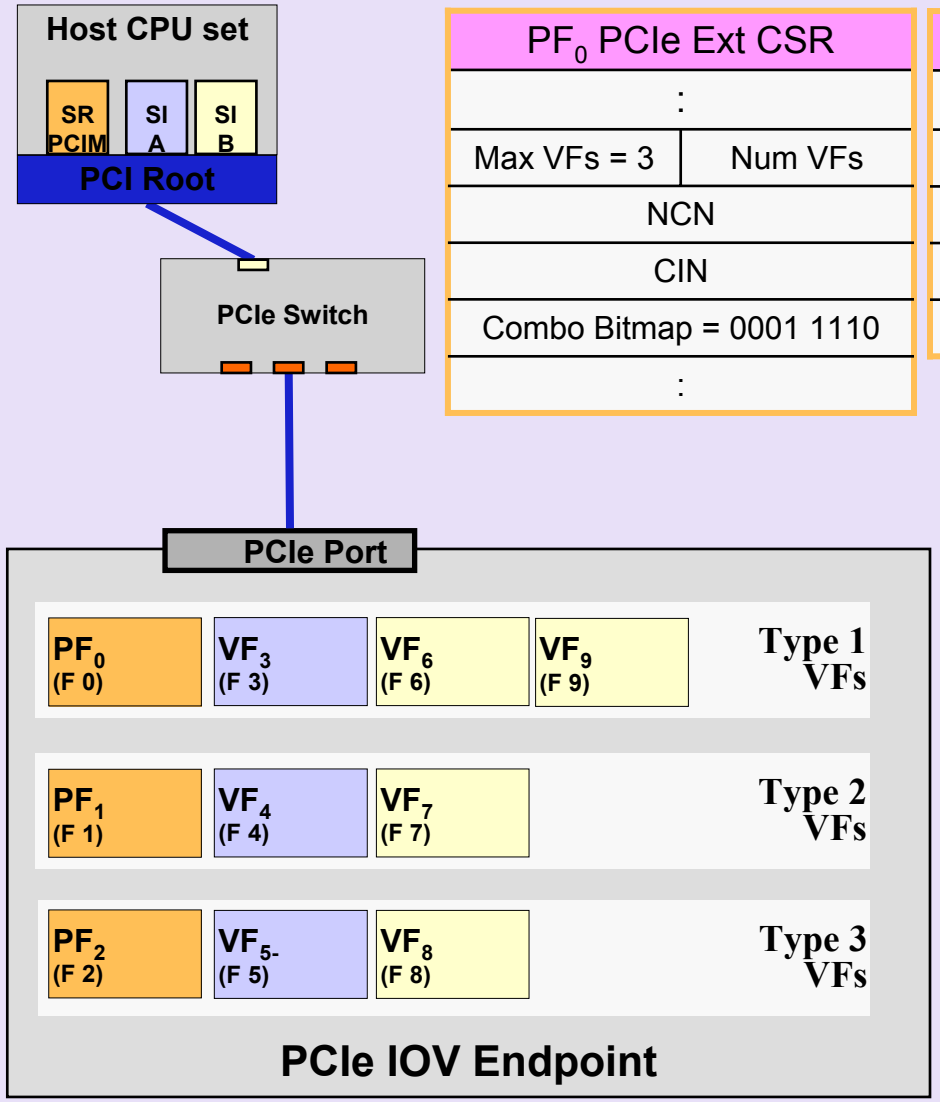
EP unsupported combinations are 0

Example

- EP with 3 Function types, where:
  - ✓ Function $Type_0$ can be used independently.
  - ✓ Function $Type_1$ can be used independently.
  - ✓ Function $Type_2$ can be used independently.
  - ✓ Function $Type_0$ and $Type_1$ can be used together.

Nominal Combo Num	2	1	0	Supported
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

Example - discovery



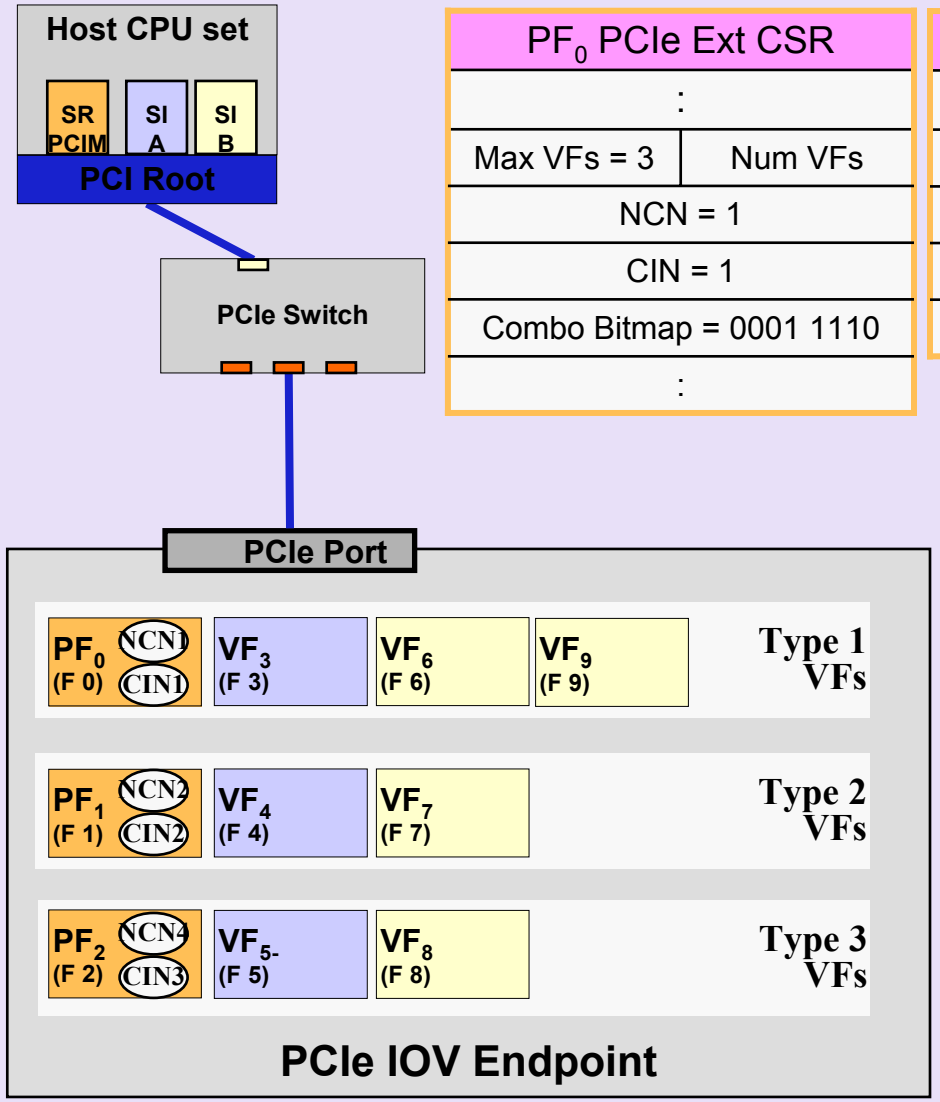
PF_0 PCIe Ext CSR	
:	
Max VFs = 3	Num VFs
NCN	
CIN	
Combo Bitmap = 0001 1110	
:	

PF_1 PCIe Ext CSR	
:	
Max VFs = 2	Num VFs
NCN	
CIN	
:	

PF_2 PCIe Ext CSR	
:	
Max VFs = 2	Num VFs
NCN	
CIN	
:	

- SR-PCIM reads EP:
  - ✓ PF_0 CSRs to determine
    - SR-IOV Cap = **Consecutive** FNs
    - Max VFs = 3
    - **PF Combinations**
    - …
  - ✓ PF_1 CSRs to determine
    - Existence of PF_1
    - Max VFs
  - ✓ PF_2 CSRs to determine
    - Existence of PF_2
    - Max VFs

Example - PF Combination setting



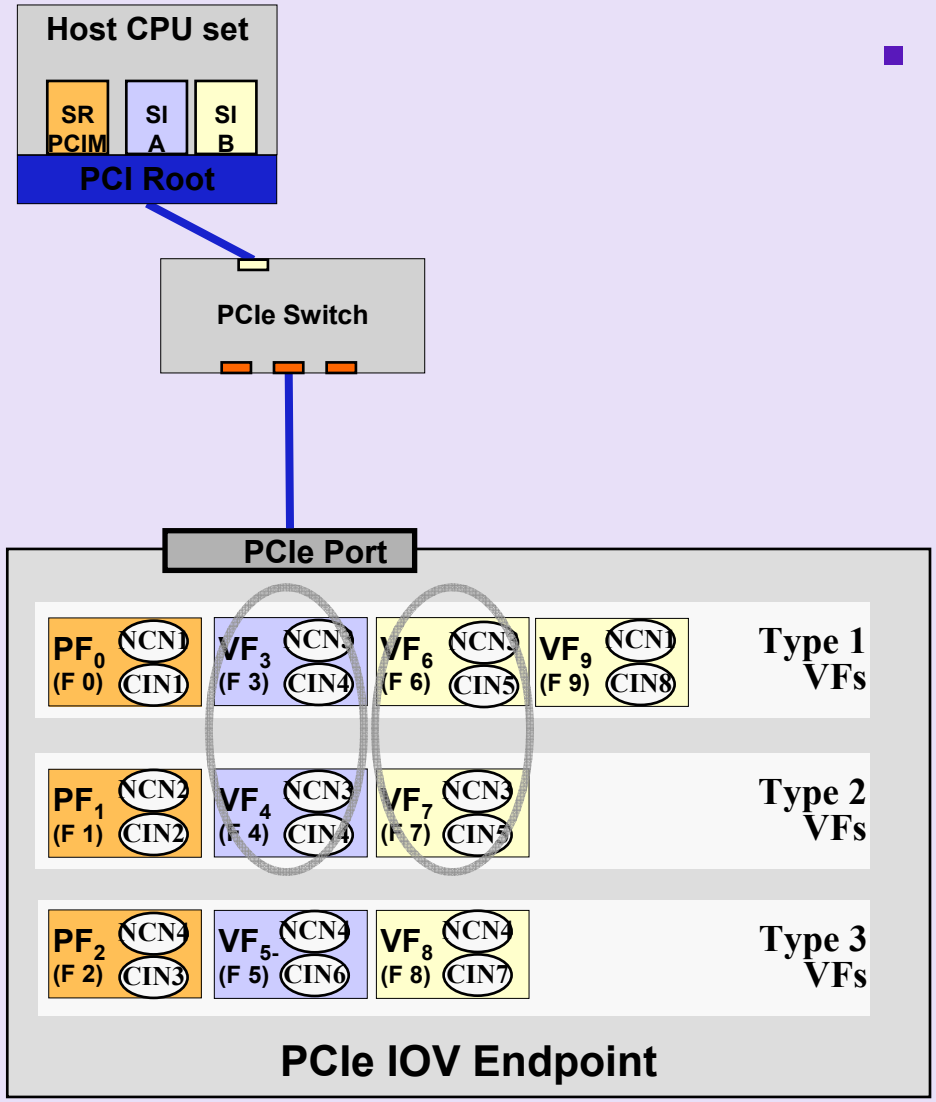
PF_0 PCIe Ext CSR	
:	
Max VFs = 3	Num VFs
NCN = 1	
CIN = 1	
Combo Bitmap = 0001 1110	
:	

PF_1 PCIe Ext CSR	
:	
Max VFs = 2	Num VFs
NCN = 3	
CIN = 2	
:	

PF_2 PCIe Ext CSR	
:	
Max VFs = 2	Num VFs
NCN = 4	
CIN = 3	
:	

- SR-PCIM sets combos for PFs:
  - ✓ Writes to PF_0 CSRs
    - Num VFs = 3
    - **NCN = 1**
    - **CIN =1**
  - ✓ Writes to PF_1 CSRs
    - Num VFs = 2
    - **NCN = 2**
    - **CIN = 2**
  - ✓ Writes to PF_2 CSRs
    - Num VFs = 2
    - **NCN = 4**
    - **CIN = 3**

Example - VF Combination setting



- SR-PCIM sets combos for VFs:
  - ✓ Writes to VF_3 CSRs
    - **NCN = 3**
    - **CIN = 4**
  - ✓ Writes to VF_4 CSRs
    - **NCN = 3**
    - **CIN = 4**
  - ✓ …
  - ✓ Writes to VF_9 CSRs
    - **NCN = 1**
    - **CIN = 8**

SR PCIM Capability

- Lives in the PF configuration space.
- SR PCIM key fields (from Resource Allocation)
  - ✓ Maximum number of VFs supported by this PF.
  - ✓ Fixed stride or independent BARs supported.
  - ✓ Minimum, maximum stride supported (fixed stride only).
    - Supports pagesize alignment of BARs in the “fixed stride” option.
  - ✓ Actual stride (fixed stride only).
  - ✓ #VFs
    - Set by SR-PCIM to create VF instances.
  - ✓ VFs enabled
    - Set by SR-PCIM after #VFs set to enable access to VF instances
- SR PCIM creates VFs using the information from the capability (with instructions from *somewhere else*).

SR PCIM – RID Assignment/Routing

- Once assigned, each VF has a unique Requester ID (RID) allowing it to be accessed by the host.
- RIDs must be assigned within the BDF range passed to this endpoint, as defined by the base spec or the base spec as modified by ARI.
- PF→VF mapping is fixed once configured.
  - ✓ Reconfiguration events are supported.

SR PCIM – Address Assignment/Routing

- Packets routed by address:

  - ✓ Fixed Stride (Shared Decoders)
    - PCI Memory and I/O BARs are relative to and decoded by the PFs BARs.
    - Devices can use a single set of address decoders for the entire set of VFs assigned to a PF.
    - Use stride and original BAR size to determine the VF instance.
    - BAR assignments must be consistent with the address routing rules defined in the PCI Express base specification.

  - ✓ Independent BARs.
    - PCI Memory and I/O BARs are independently decoded by each PF & VF.
    - Each PF and VF BAR can be set independently
    - BAR assignments must be consistent with the address routing rules defined in the PCI Express base specification.

SR PCIM – Interrupts/Msgs

- Each VF gets its own set of MSI/X resources.
  - ✓ MSI messages include the Requester ID of the sender.
  - ✓ Each MSI can be assigned unique address/data values.
- Messages must contain the Requester ID of the sender.
- We're looking into several other methods for supporting interrupts which reduce the VF space requirements.

SR: SIs use of VFs

SI – Role of the VI

- Existing SIs can still probe the fabric.
- The VI is required to arbitrate access to the PCI address space, specifically configuration space.
  - ✓ If an SI attempts to probe/access a function using a Requester ID that it doesn't own, the VI must reject that access.
  - ✓ If the VI simply returns an error, the SI can assume that the function does not exist.
  - ✓ Some config space fields require VI emulation. (details later).
- As with any other physical resource, the VI must protect SIs from accessing PCI memory and IO space that they don't own.
- The VI should protect virtual functions from accessing system memory that they don't have access to. (DMA).

SR – Key Config Space Requirements

- SR PCIM must be able to discover PFs and configure them.
  - ✓ SR PCIM Capability
- Each VF must have a unique Requester ID.
  - ✓ Unique configuration space address to discover the VF instance.
  - ✓ Unique Requester ID in interrupts, messages, R/W requests, etc.
- Compatibility with the PCI Express Base
  - ✓ Retain header layout for type 0 and 1 headers.
  - ✓ No need to implement all bits.
  - ✓ Maintain configuration space read/write semantics. (ordering …)
  - ✓ Maintain routing rules defined by the base spec.
- Minimize bits that must be implemented per VF.
  - ✓ Alias bits where possible.
  - ✓ Implement bits where required.
  - ✓ VI emulation where “alias” or “implement” is not practical.

SR - Register Blocks of Interest

- Type 0 Header
- PCI Express Capability
- MSI and MSI-X Capabilities
- VPD Capability
- Power Management Capability
- AER Extended Capability
- Function Level Reset (FLR) Capability
- VC Extended Capability
- MFVC Extended Capability
- Power Budgeting Extended Capability

SR - Implementation Codes

- I — VF/PF – **I**mplement if capability present
- R — VF/PF – **R**ead-only value (VF: same value as PF)
- Z — VF – **Z**ero constant, Read-only
- E — VF – **E**mulate field in VI – HW field is “don’t care”

- Iram — Implement as a R/W register with no side effects
- O — Optional – Do not implement in VF if not implemented in PF.
- RO — Read-Only field
- RW — Read/Write field

- We’re still working on details
  - ✓ Code assignments are roughly cast in Jello™ at this point

SR – Type 0 Header (1 of 2)

Field	Use	PF	VF
Vendor ID	RO	R	R
Device ID	RO	R	R
Subsystem VID	RO	R	R
Subsystem DID	RO	R	R
Revision ID	RO	R	R
Class Code	RO	R	R
Header Type	RO	R	R
Capabilities Pointer	RO	R	R
Capabilities List	RO	R	R
IO Space Enable	RW	I	I
Mem Space Enable	RW	I	I
Bus Master Enable	RW	I	I

Field	Use	PF	VF
BAR0 .. BAR6	RW	I	***
Expansion ROM BAR (address)	RW	O	E
Expansion ROM Enable	RW	O	E
Interrupt Line	RW	Iram	E
Interrupt Disable	RW	I	I
Interrupt Status	RO	I	I
Interrupt Pin	RO	R	R
BIST	RW	O	Z
Cache Line Size	RW	Iram	E

SR – Memory & I/O BARs

- Fixed Stride (Shared Decoders).
  - ✓ Recall that the SR PCIM capability defines maximum and minimum stride supported and that SR PCIM can set the actual stride.
  - ✓ VF BAR Size = MAX(Original_PF_BAR_size, stride)
    - – “Stride” allows pagesize aligned BARs.
  - ✓ Once configured and VFs are enabled, the PFs BARs expand to decode the entire address space for the PF and all configured VFs.
  - ✓ This method allows a single decode per PF BAR, which covers all VFs BARs
  - ✓ VF BARs are emulated (E) by the VI when accessed by the SI.
- Independent BARs.
  - ✓ SR-PCIM capability indicates that VFs have BARs that are independently decoded.
  - ✓ VF BARs are implemented (I) per VF.

SR – Type 0 Header (2 of 2)

Field	Use	PF	VF
Parity Error Response	RW	I	I
SERR# Enable	RW	I	I
Master Data Parity Error	RW 1C	I	I
Signaled Target Abort	RW 1C	I	I
Signaled System Error	RW 1C	I	I
Received Target Abort	RW 1C	I	I
Received Master Abort	RW 1C	I	I
Detected Parity Error	RW 1C	I	I

- See Error Discussion.
- Not all bits need to be “I”.
- Bits that indicate device error (vs VF specific errors) should be handled in the PF registers.

SR – PCI Express Capability (1 of 3)

Field	Use	PF	VF
Capability Hdr Stuff	RO	R	R
Dev / Port Type	RO	R	R
Interrupt Msg Num	RO	R	R
Max Payload Supported	RO	R	R
Max Payload Size	RW	I	E
Max Read Req Size	RW	I	E
Phantom Function Supported	RO	R	Z
Phantom Function Enable	RW	I	Z
Extended Tag Supported	RO	R	R
Extended Tag Enable	RW	I	E

Field	Use	PF	VF
Enable No Snoop	RW	I	I
Enable Relaxed Ordering	RW	I	I
Role Based Error Reporting	RO	R	R
Max Link Speed	RO	R	R
Max Link Width	RO	R	R
Link Speed	RO	I	R
Negotiated Link Width	RO	I	R
Read Completion Boundary	RW	I	E
Extended Sync	RW	I	E
Port Number	RO	R	R

SR – PCI Express Capability (2 of 3)

Field	Use	PF	VF
Captured Slot Power Limit & Scale	RO	I	R
Clock / Pwr Mgmt	RO	R	Z
Enable Clock Power Mgmt	RW	I	E
ASPM Supported	RO	R	Z
ASPM Control	RW	I	Z
AUX Power PM Enable	RW S	I	E
AUX Power Detected	RO	I	E

Field	Use	PF	VF
EP L0s & L1 Acceptable Latency	RO	R	R
L0s / L1 Exit Latency	RO	R	R
Common Clock Config	RW	I	E
Transactions Pending	RO	I	I
Slot Configuration	RO	I	R

SR – PCI Express Capability (3 of 3)

Field	Use	PF	VF
Correctable Error Reporting Enable	RW	I	I
Non-Fatal Error Reporting Enable	RW	I	I
Fatal Error Reporting Enable	RW	I	I
Unsupported Request Error Reporting Enable	RW	I	I
Correctable Error Detected	RW 1C	I	I
Non-Fatal Error Detected	RW 1C	I	I
Fatal Error Detected	RW 1C	I	I

Field	Use	PF	VF
Unsupported Request Error Detected	RW 1C	I	I

- See Error Discussion.
- Not all bits need to be “I”.
- Bits that indicate device error (vs VF specific errors) should be handled in the PF registers.

SR – MSI and MSI-X Capabilities

- MSI or MSI-X is required by the PCI-Express base.
  - ✓ Either one or both may be implemented.
- PF and VF both get standard “full featured” capabilities
  - ✓ Nothing changed from Base spec
- We’re looking into other options to reduce VF space requirements.

SR – VPD Capability

- Optional Capability
- VPD is optional in PFs and VFs.
  - ✓ It's up to the device vendor if it's implemented in PFs and VFs or not.
  - ✓ Interpretation of the data is device specific.
  - ✓ If implemented, the device provides any required access controls.
  - ✓ If implemented in PFs or VFs, the device must ensure that data leakage through writable areas is not permitted.
    - If there's a writable area, it must be per VF (?)
    - The device decides how to associate VPD fields with VFs.
  - ✓ Error logging, etc, should probably be done using the PF.
  - ✓ The VI can't emulate VPD access without device-specific knowledge.

SR – Power Management Capability

- Power State / Power Status fully implemented per VF
  - ✓ e.g. VF implements “virtual D3” so things like wake-on-LAN can be implemented in a VF
- Data_Select / Data / Data_Scale emulated by VI
  - ✓ software provides PF Data values to associated VFs
- For PF, we may need to distinguish between:
  - ✓ virtual power state (“virtual D3” state of just the PF) and
  - ✓ real power state (actual power state of the PF and associated VFs)
- This is only one of several possibilities. We’ll require/provide state where it’s needed.

SR – Serial Number Extended Capability

- Optional Capability

- Provide the same 64 bit unique number to all PFs and VFs
  - ✓ Up to the device vendor?

SR – Power Budgeting Extended Capability

- Optional Capability
- Present in PF
- Not Present in VF
  - ✓ Emulated by VI if desired

SR – AER Extended Capability

- Optional Capability
- Up to the IHV if AER is implemented per VF or not.
- See Error Discussion

SR – Function Level Reset

- Provides reset capability for a single function in a multi-function device
- Currently FLR is an ECR in review.
- FLR can be used to provide VF and PF reset.
- Intent is to require FLR in VFs or to have similar per-VF functionality in the PF.

SR – VC Extended Capability

- Optional Capability

- VC Arbitration Table + Associated Control Bits
  - ✓ Optional Table in Base
- VC Enable
  - ✓ Enables non-zero VCs
- VC Negotiation Pending
  - ✓ Reflects state of non-zero VCs
- VC / TC Map
  - ✓ Maps TC to/from VC

- No details decided about VF / PF usage of VCs, but looking to reuse the existing infrastructure as much as possible.

MFVC Extended Capability

- Optional Capability
- Like VC Capability but adds:

- Multi-Function Arbitration table
  - ✓ Arbitrates between functions within a given VC

- No details decided about VF / PF usage of VCs
- No details decided about VF / PF Arbitration
- Looking to reuse the existing infrastructure as much as possible

Questions

Thank you for attending.

For more information please go to
www.pcisig.com

Single Root IO Virtualization

David Kahn (Sun Microsystems)