IOV Error, Interrupt and Event Handling

Mahesh Wagh

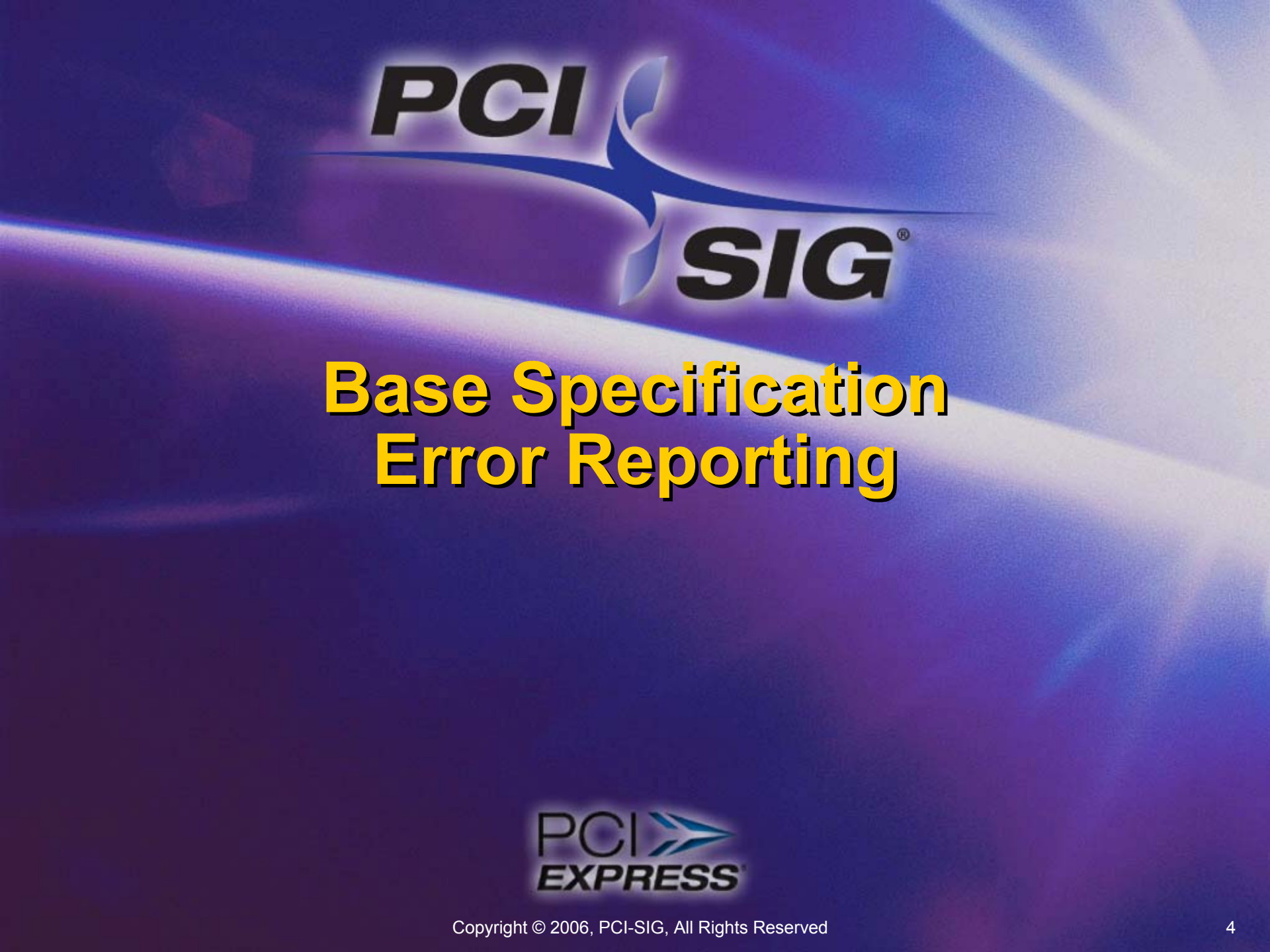
(Intel)

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

- Error Reporting
- Interrupts
  - ✓ INTx
  - ✓ MSI / MSI-X
- Other Events



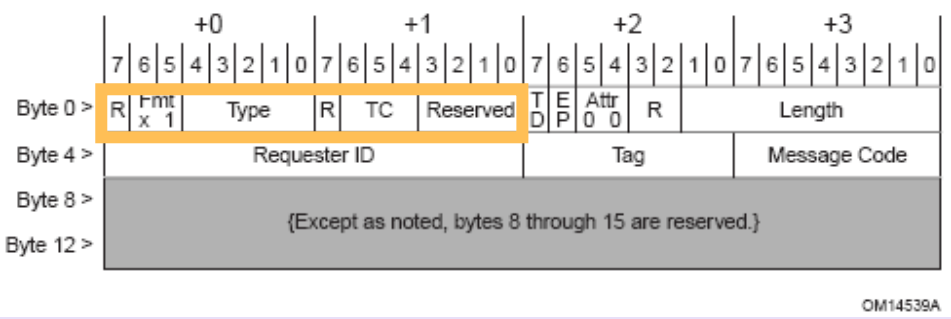
Base Specification Error Reporting

Background

- Two error reporting paradigms
  - ✓ the baseline capability –required by all PCIe® devices
  - ✓ the Advanced Error Reporting capability - optional
- Baseline error reporting capabilities define the minimum error reporting requirements.
- The Advanced Error Reporting is defined for more robust error reporting
- All PCI Express® errors map to existing PCI reporting mechanism.

Error Messages

- Error Messages are sent to the Root
  - ✓ ERR_COR
  - ✓ ERR_NONFATAL
  - ✓ ERR_FATAL
- Initiator of the Message identified by the Requestor ID of the message header
  - ✓ Error Source Identification Register – with Advanced Error Reporting
- Errors Messages translated into platform level events
  - ✓ System Error
  - ✓ Error Interrupt – with Advanced Error Reporting



Baseline Error Reporting

- Required by all PCI Express devices
- Uses conventional PCI error status bits
  - ✓ Master Abort, Signaled Target Abort, Master Data Parity Error, etc.

Advanced Error Reporting

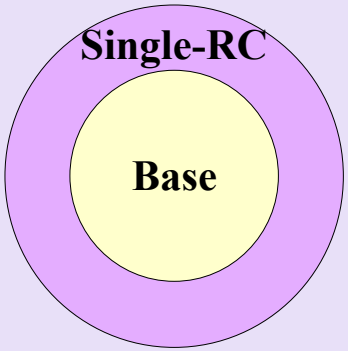
- Device Logs Header of TLP that caused the Error
  - ✓ Correctable / Uncorrectable status bits are set
  - ✓ Conventional status bits are also set (Master Abort, Target Abort, etc.)
- Root Port logs the Source ID found in the Error Message
  - ✓ Logged in the Error Source Identification register.
  - ✓ Records the Requester ID of the first ERR_NONFATAL/ERR_FATAL (uncorrectable errors) and ERR_COR (correctable errors)

Multi-Function Device

- In a multi-function device, errors that are not related to any specific function, are logged in the status and logging registers of all functions.
- The following PCI Express errors are not function-specific:
  - ✓ All Physical Layer errors
  - ✓ All Data Link Layer errors
  - ✓ These Transaction Layer errors:
    - ECRC Fail
    - UR, when caused by no function claiming a TLP
    - Receiver Overflow
    - Flow Control Protocol Error
    - Malformed TLP
- On the detection of one of these errors, a multi-function device should generate at most one error reporting Message of a given severity.
- Software is responsible for scanning all functions in a multi-function device when it detects one of these errors.

Single Root Error Virtualization

SR Error Logging



- Want to identify the affected SI
  - ✓ Which Virtual Function
- Evolutionary approach over Base
- IOV devices are fundamentally Multi-Function Devices
  - ✓ Physical functions plus Virtual Functions
- Errors are logged according to Base Specification rules

Single Root Error logging

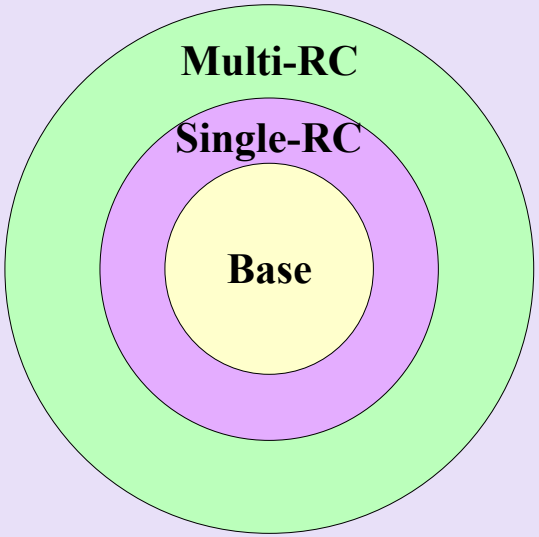
- Legacy Control and Status bits replicated
  - ✓ Per VM control
  - ✓ Per VM error detection
  - ✓ Help with fault isolation
- Advanced Error Reporting still optional
  - ✓ Improves fault detection/isolation

Question:

- Do all legacy control/status bits need to be replicated?
- If Implemented, can VFs share the same Adv Error Header Log with the PF?
  - ✓ Could Eliminate AER Header Log (128bits) per VF
  - ✓ Reads of VF return log from PF
  - ✓ Header log includes RID which identifies VF
  - ✓ For Multi-Function Errors, the Header would be identical

Multi-Root Error Virtualization

Multi-Root Error Logging



- Want to alert the affect Root Port
  - ✓ Which Virtual Hierarchy
- Builds on Base and SR-PCIM.
  - ✓ Concentric circles
- MR devices are fundamentally Multi-Function Devices
  - ✓ Multiple Physical functions
  - ✓ May include Virtual Functions
- Leverage Multi-Function mechanisms

Multi-Root Error logging

- Each Virtual Hierarchy has a Physical Function
  - ✓ Physical functions have independent configuration space
  - ✓ Conventional Error control and status bits are present (required)
  - ✓ Advanced Error Reporting may be present (optional)
- Assume a dedicated Virtual Hierarchy for MR-PCIM
  - ✓ VH0 is a special case of a Virtual Hierarchy

Base Errors

- Errors that are not related to any specific **virtual hierarchy:**
  - ✓ are logged in the status and logging registers of all active virtual hierarchies including MR-PCIM (VH0).
  - ✓ The following PCI Express errors are not root-specific:
    - All Physical Layer errors
    - All Data Link Layer errors
- Most Transaction layer errors are VH Specific
  - ✓ Contain a valid VH number
  - ✓ Error sent to affect Root Port
  - ✓ Base Multi-Function rules apply within the Virtual Hierarchy
- Some Transaction Layer errors may affect MR-PCIM
  - ✓ Send to affected Root Port(s) and MR-PCIM
  - ✓ Receiver overflow
  - ✓ Flow Control Protocol Error

Multi-Root-Specific Errors

- Some errors impact all hierarchies
  - ✓ Sent to all active VH including MR-PCIM
    - Invalid VH number
    - Error in training/flow control
    - Additional errors
  - ✓ Analogous to Multi-Function Error in base
- Some errors impact a single VH
  - ✓ Sent to affect VH and MR-PCIM
    - Error in training/flow control
- Some errors used to notify MR-PCIM of fabric issues
  - ✓ Sent only to MR-PCIM
    - Errors in new Multi-Root DLLPs

Summary

		Base	SR	MR
Physical	Receiver Error	RC	RC	Active RC & MR PCIM
Link	Bad TLP	RC	RC	Active RC & MR PCIM
	Bad DLLP	RC	RC	Active RC & MR PCIM
	Replay Timeout	RC	RC	Active RC & MR PCIM
	Repaly Num Rollover	RC	RC	Active RC & MR PCIM
	Data Link Layer Protocol Error	RC	RC	Active RC & MR PCIM
Transaction	Poisoned TLP Received	RC	RC	Affected RC
	ECRC Check Failed	RC	RC	Affected RC
	Unsupported Request	RC	RC	Affected RC
	Completion Timeout	RC	RC	Affected RC
	Completer Abort	RC	RC	Affected RC
	Unexpected Completion	RC	RC	Affected RC
	Receiver Overflow	RC	RC	Affected RC(s) & PCIM
	Flow Control Protocol Error	RC	RC	Affected RC(s) & PCIM
	Malformed TLP	RC	RC	Affected RC
VP Errors	Invalid VP	n/a	n/a	Active RC & MR PCIM
	Flow Control Error	n/a	n/a	Affected RC & MR PCIM
	Other Errors	n/a	n/a	MR PCIM

Virtualizing PCI Express Interrupts

Interrupt Value Proposition

- Provide one or more vectors per VF
  - ✓ scalability/performance
- Identify the interrupt source
  - ✓ for correct operation
  - ✓ for efficiency
- Limit VMM's work
  - ✓ reduce the need to maintain state
- Compatibility with existing software models
  - ✓ Work with existing Guest OS / VM software
  - ✓ No requirement to change driver model

PCI Express Interrupt Support

- The PCI Express interrupt model supports two mechanisms:
  - ✓ Legacy INTx emulation
  - ✓ Message Signaled Interrupt (MSI/MSI-X)
- The rationale of dual approach include:
  - ✓ Compatibility with existing PCI Software Models
  - ✓ Direct support for boot devices
  - ✓ Easier End of Life (EOL) for INTx legacy mechanism.
- Each device is required to differentiate between INTx and MSI modes of operation.
- Existing software model can already differentiate INTx vs MSI modes of operation.
- MSI, MSI-X, or both is required for native PCI Express endpoint devices

Legacy INTx

- PCIe supports the PCI interrupts as defined in the *PCI Local Bus Specification, Revision 3.0*
- Level Triggered mechanism
  - ✓ Two messages are defined, Assert_INTx and Deassert_INTx
  - ✓ Assert/Deassert messages used in pairs to emulate level-triggered signaling.
- Switches collect and combine these virtual wires.
  - ✓ Switch uses is own RID when forwarding message to upstream port.
  - ✓ Routed to the Root which maps the virtual wires to system interrupt sources.
- Software model matches that of PCI.
  - ✓ For System BIOS reporting of chipset/platform interrupt mapping
  - ✓ For association of a device's interrupt with PCI interrupt lines

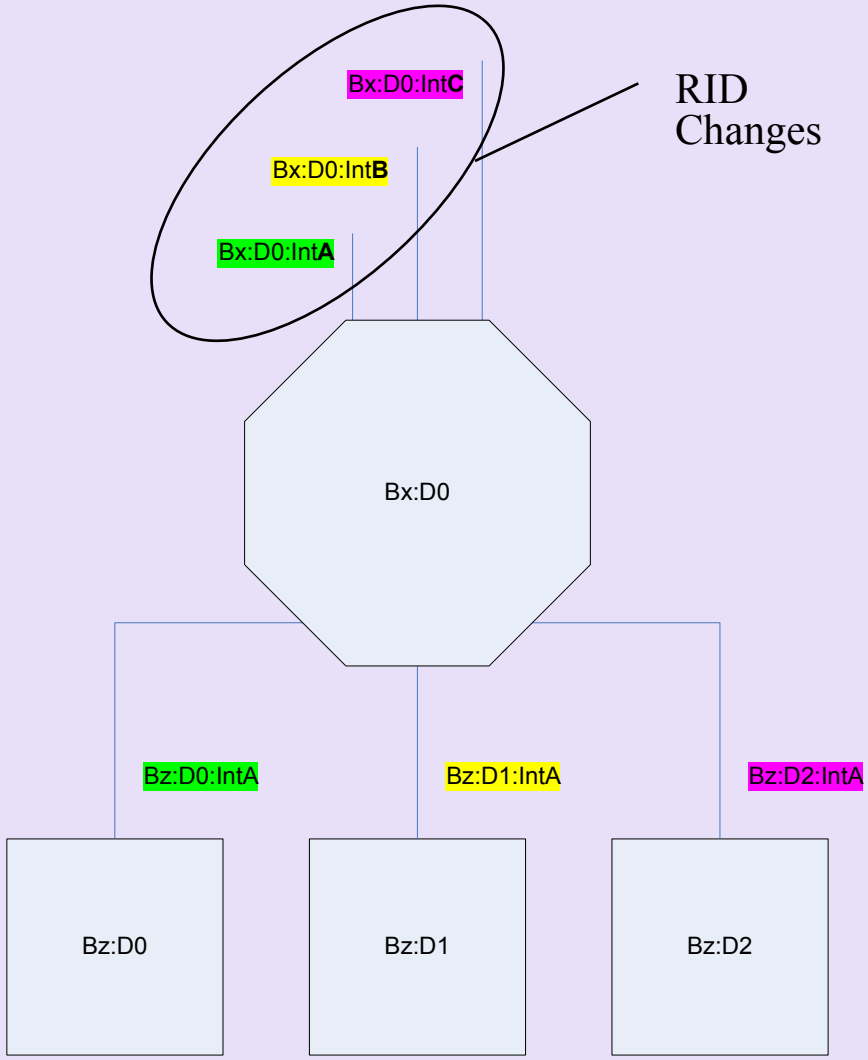
Bridge Mapping for INTx

INTx Message changes as it passes through switches.

- ✓ Bz:D0:IntA -> Bx:D0:IntA
- ✓ Bz:D1:IntA -> Bx:D0:IntB
- ✓ Bz:D2:IntA -> Bx:D0:IntC

Table 2-13: Bridge Mapping for INTx Virtual Wires

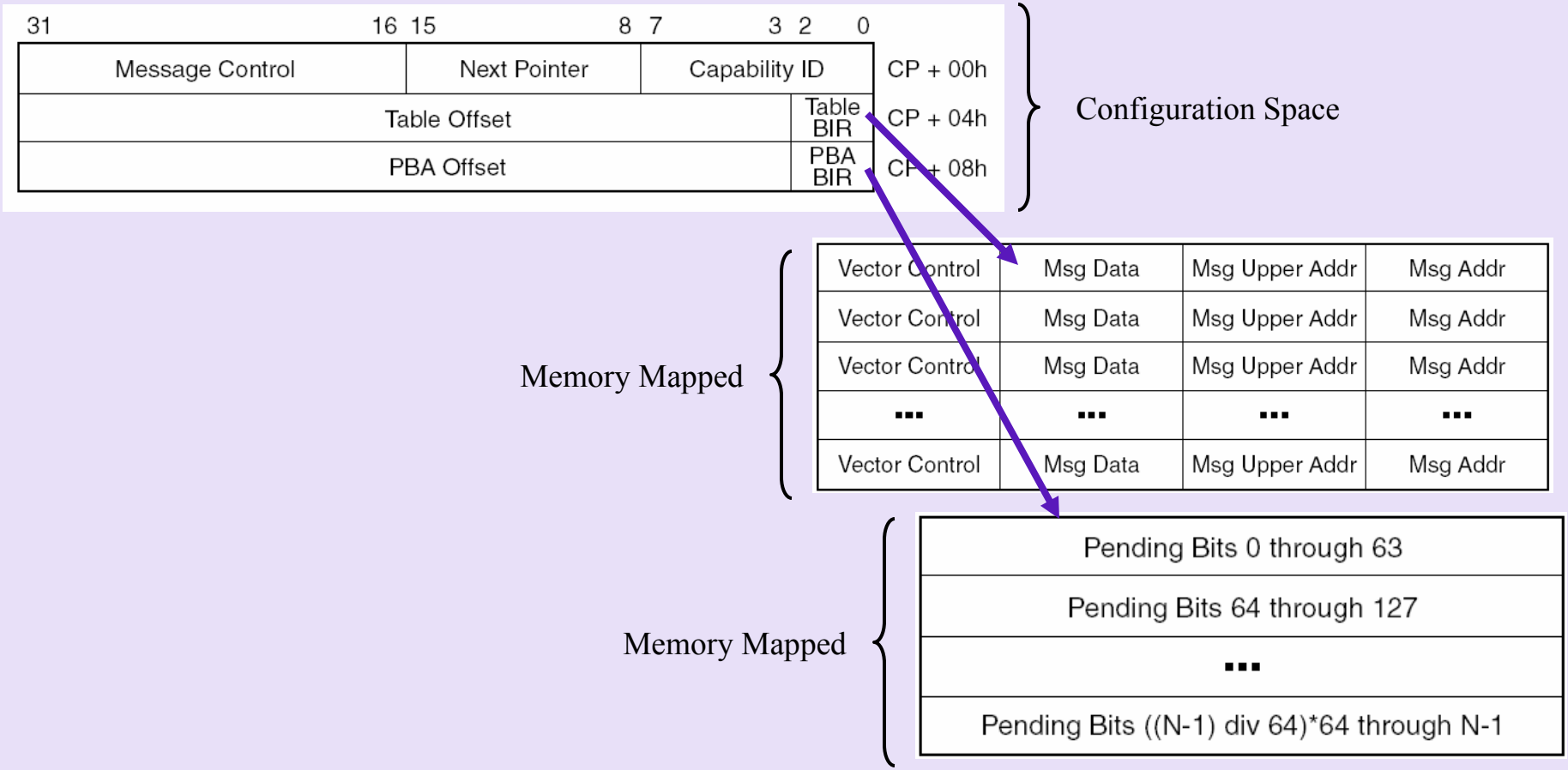
Device Number for Device on Secondary Side of Bridge (Interrupt Source)	INTx Virtual Wire on Secondary Side of Bridge	Mapping to INTx Virtual Wire on Primary Side of Bridge
0,4,8,12,16,20,24,28	INTA	INTA
	INTB	INTB
	INTC	INTC
	INTD	INTD
1,5,9,13,17,21,25,29	INTA	INTB
	INTB	INTC
	INTC	INTD
	INTD	INTA
2,6,10,14,18,22,26,30	INTA	INTC
	INTB	INTD
	INTC	INTA
	INTD	INTB
3,7,11,15,19,23,27,31	INTA	INTD
	INTB	INTA
	INTC	INTB
	INTD	INTC



Message Signaled Interrupt

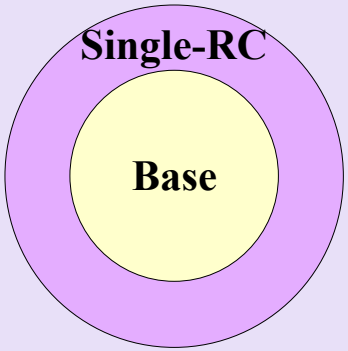
- MSI/MSI-X interrupt support required for PCIe devices
- Delivers interrupts via memory write transactions
  - ✓ Edge-triggered mechanism
  - ✓ 64-bit Message Address version of MSI is required.
- MSI and MSI-X have Capability Structures in Configuration Space.
- MSI-X requires MMIO region
  - ✓ MSI-X tables and Pending Bit Array are located in MMIO space
  - ✓ Offset relative to a BAR
  - ✓ MSI-X Table is recommended to be in its own 4KB page.

MSI-X Capability



Legacy INTx Virtualization

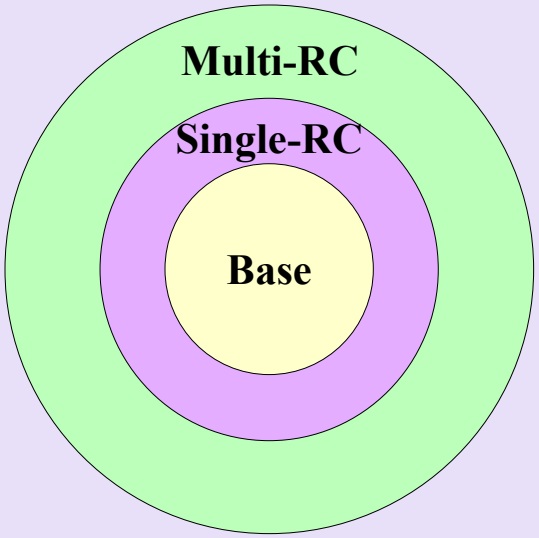
Single Root INTx Routing

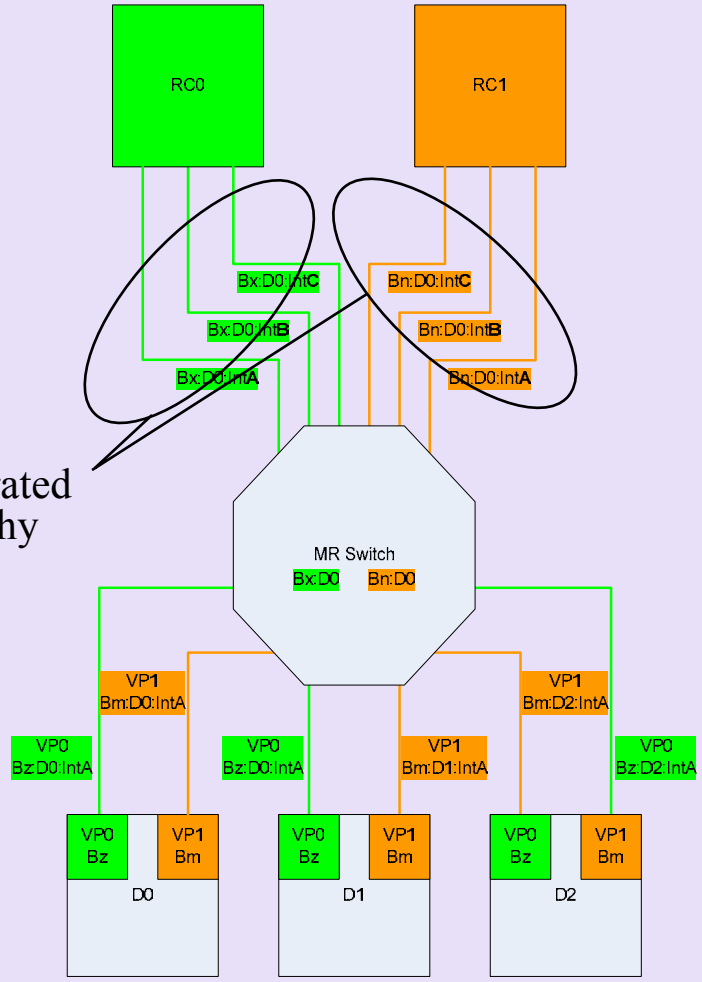


- No changes on the Wire
- INTx delivered to interrupt controller
- Source device unknown
  - ✓ Intervening switches take ownership
- VI involved in interrupt delivery
  - ✓ Similar to OS role for a shared interrupt
  - ✓ VI calls SIs that share that Interrupt line

Multi-Root INTx Routing

- Virtual Hierarchy Identifier
  - ✓ Applied at endpoint
  - ✓ Used by switch to route TLP
- INTx routes with a VH
  - ✓ Switch collects and combines per VH
  - ✓ SR rules apply within a VH



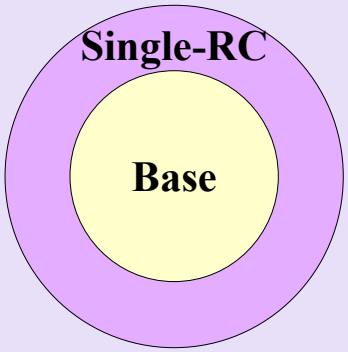


Why Virtualize INTx

- Compatibility with existing software model
- Some device models rely on level sensitive semantics
- Most existing OSes do not support MSI/MSI-X
- Devices already designed to manage the INTx vs. MSI-X on a per function basis.
- Why not emulate INTx using MSI
  - ✓ MSI has edge sensitive semantics
  - ✓ Would need to invent new scheme to replicate level semantics
    - – i.e. End point would reissue the MSI after some time out
  - ✓ Why invent a new scheme when the legacy mode works
    - – but at a performance penalty

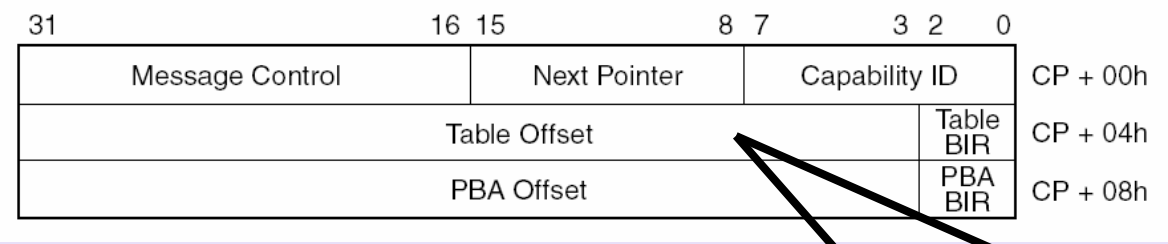
MSI and MSI-X Virtualization

Single Root MSI/MSI-X

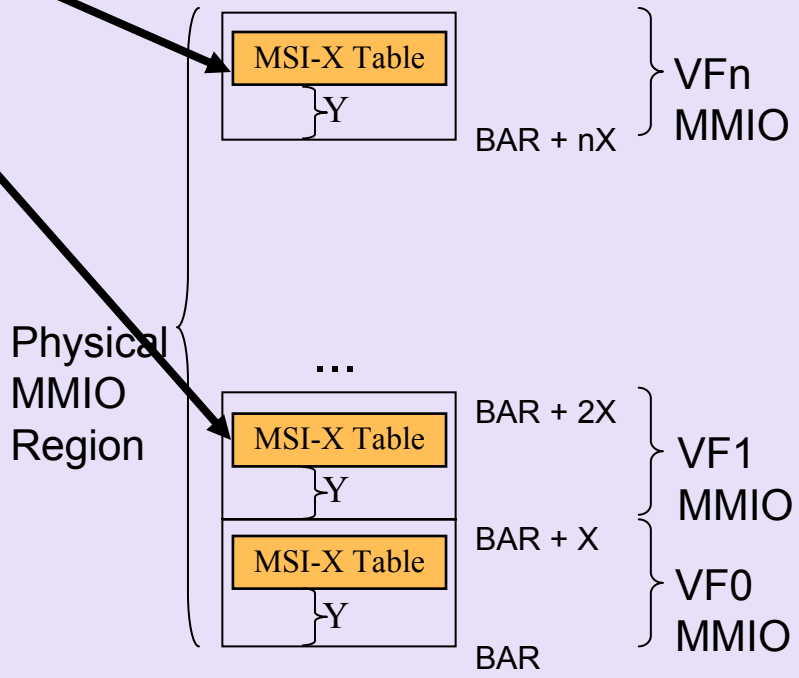


- No changes on the Wire
  - ✓ Memory mapped routing applies
- Source Identification Retained
  - ✓ TLP includes Requester ID
  - ✓ RID preserved throughout hierarchy
- MSI-X Table / PBA
  - ✓ Replicated per VF
  - ✓ Located relative to VF MMIO regions

Access to MSI-X Table

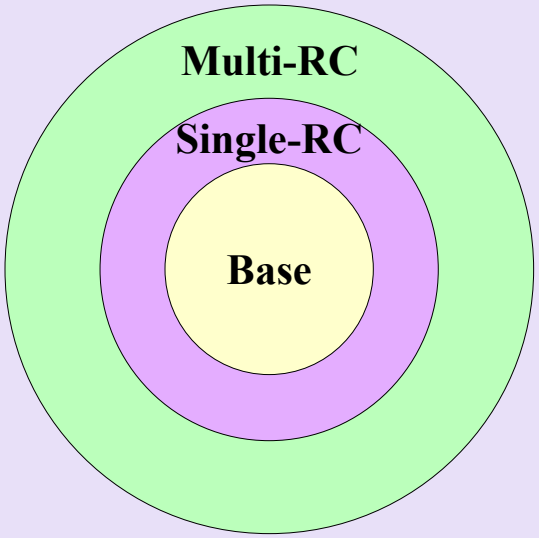


- BIR points to PF BAR
- Offset is relative to VF Base Address



X is VF offset
Y is Table Offset

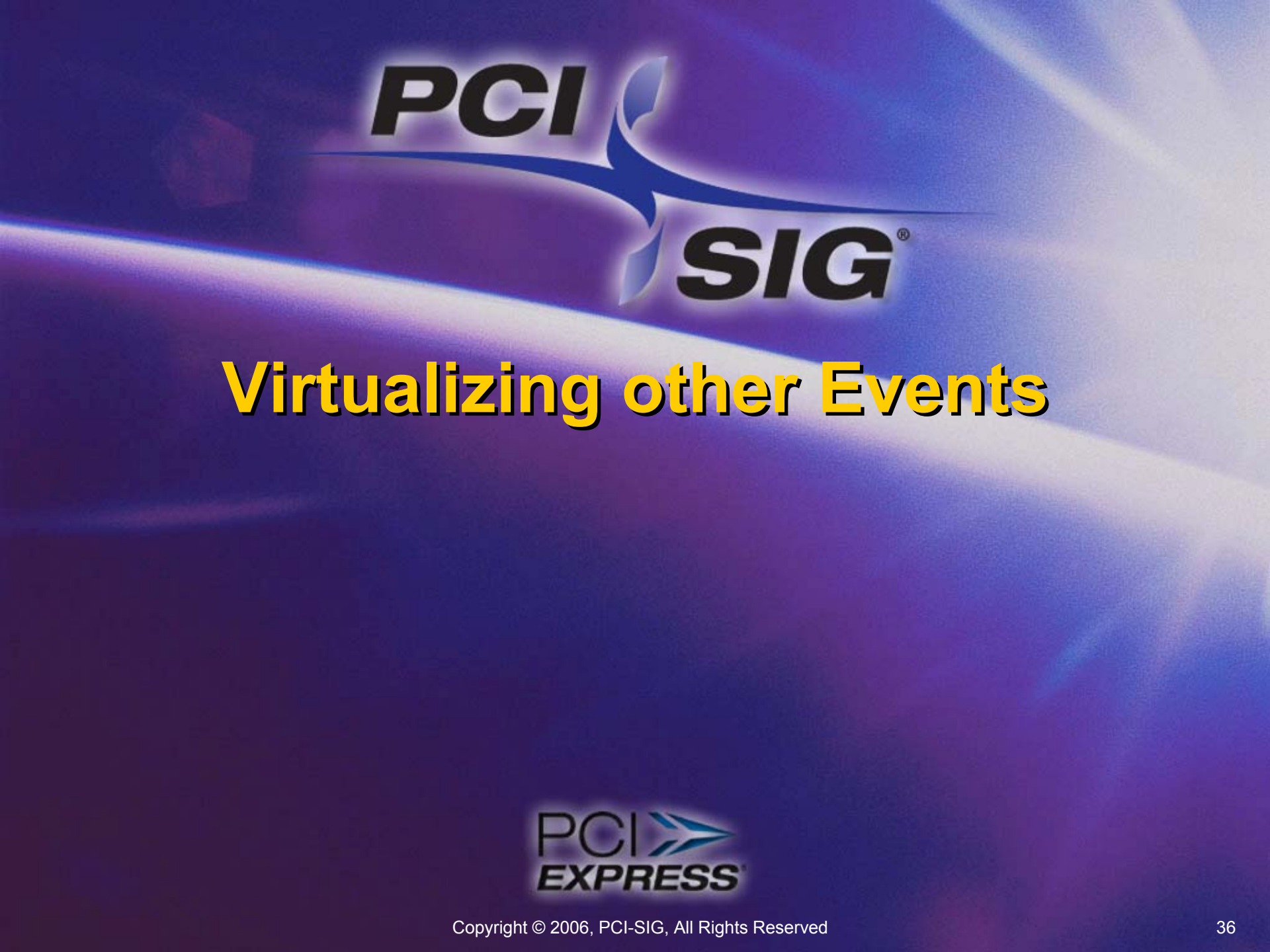
Multi Root MSI/MSI-X



- TLP routed within a Virtual Hierarchy
  - ✓ VH Tag is part of routing mechanism
- SR rules apply within the VH
- Source Identification Retained
  - ✓ TLP includes Requester ID
  - ✓ RID preserved throughout hierarchy

Why Virtualize MSI / MSI-X

- Source Identification retained
  - ✓ Reduces VI state
  - ✓ Posted Write Transactions
  - ✓ Behaves like any other write
  - ✓ RID preserved through hierarchy
  - ✓ Better performance over INTx
- Multiple Vectors per device
  - ✓ Targets 64-bit address space
- Vectors are limited resource
  - ✓ Not Shared
  - ✓ What happens when vector pool is exhausted
- MSI, MSI-X, or both required by PCIe Base Specification



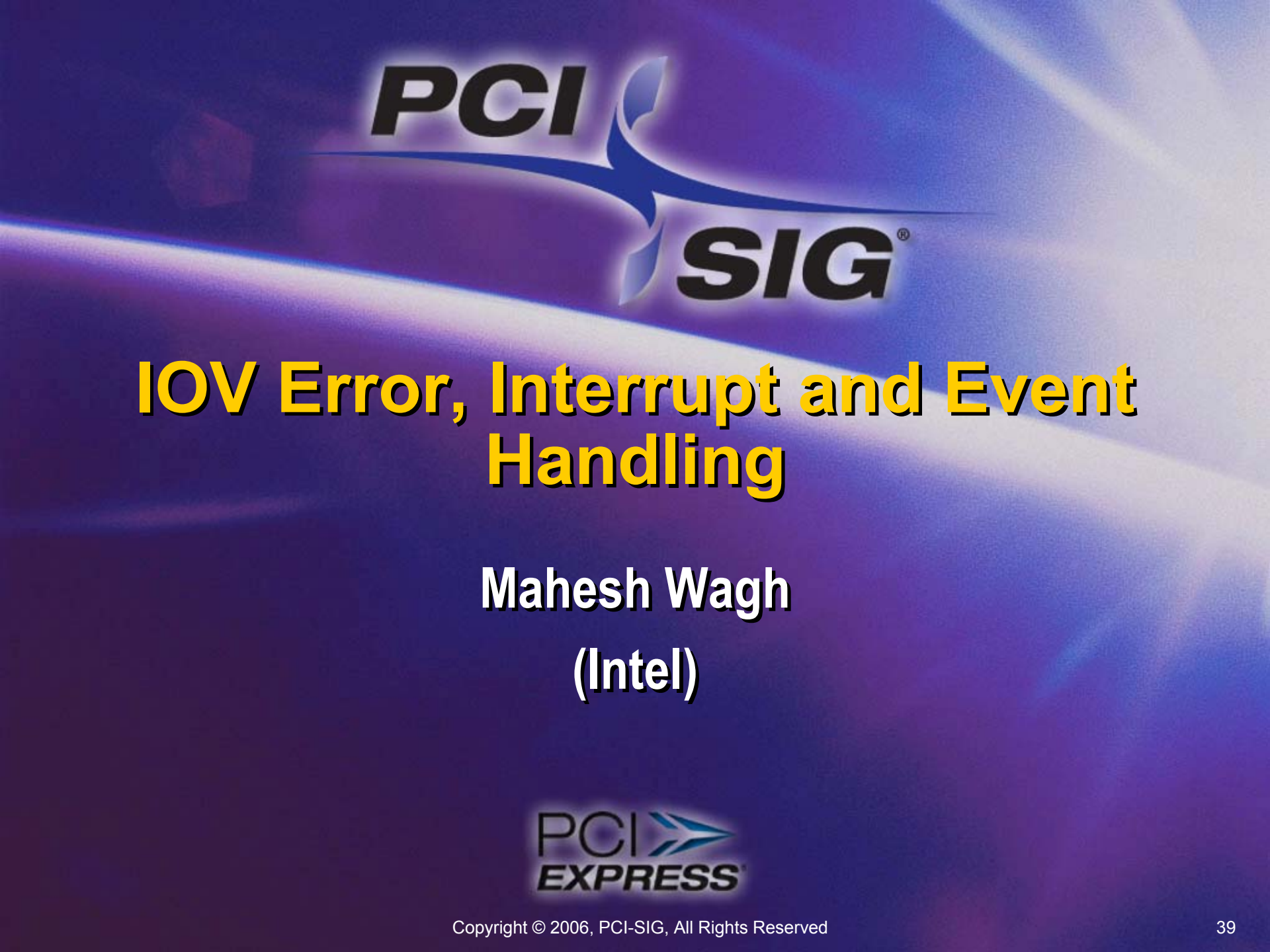
Virtualizing other Events

Events

- Event signaling in PCIe is performed via Message TLPs
  - ✓ INTx, Error, PME
- Routed based on Message Routing code
  - 000 Route to Root
  - 001 Route by Address
  - 010 Routed by ID
  - 011 Broadcast from Root
  - 100 Local – Terminate at Receiver
  - 101 Gather and Route to Root
  - 110-111 Reserved
- Single Root
  - ✓ Message Routing retains current behavior
- Multi Root
  - ✓ Message Routing stays within a Virtual Hierarchy
  - ✓ SR rules apply within the VH

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

For more information please go to
www.pcisig.com



IOV Error, Interrupt and Event Handling

Mahesh Wagh

(Intel)