
PCI
SIG®

PCIe™ Error Reporting ECN

Joe Cowan
Computer Systems Architect
Hewlett-Packard Company

Copyright © 2005, PCI-SIG, All Rights Reserved

Presentation Overview

- Role-Based Error Reporting Background
- PCIe Error Reporting Basics
- PCIe Architected Error Handling
- “The Critical Issue” That Needed Fixing
- Can Software Contain Non-Fatal Errors?
- Preserving the Value of Advanced Error Reporting
- Other Issues That the ECN Addresses
- The Five Advisory Non-Fatal Error Cases
- Error Message Controls: New Figure
- Key Hardware & Software Impacts by the ECN
- PCIe Bridges & Role-Based Error Reporting
- Recommendations

Copyright © 2005, PCI-SIG, All Rights Reserved

Role-Based Error Reporting Background

- Protocol & Software Workgroups (PWG & SWG) began exploring “the critical issue” in August 2004
  - ✓ Identified a number of other error reporting issues
  - ✓ PWG became the owning WG; SWG kept in close sync
  - ✓ Ultimately agreed that the ECN was critical enough to warrant delaying the PCIe Base 1.1 release
  - ✓ New semantics are referred to as “Role-Based Error Reporting”
- Draft ECN against PCIe Base 1.0a began its 30-day Member Review in November 2004
- Final 1.0a ECN approved by PCI SIG Board of Directors in March 2005
- Final 1.0a ECN has been incorporated into PCIe Base 1.1
  - ✓ All 1.1-compliant parts must implement Role-Based Error Reporting

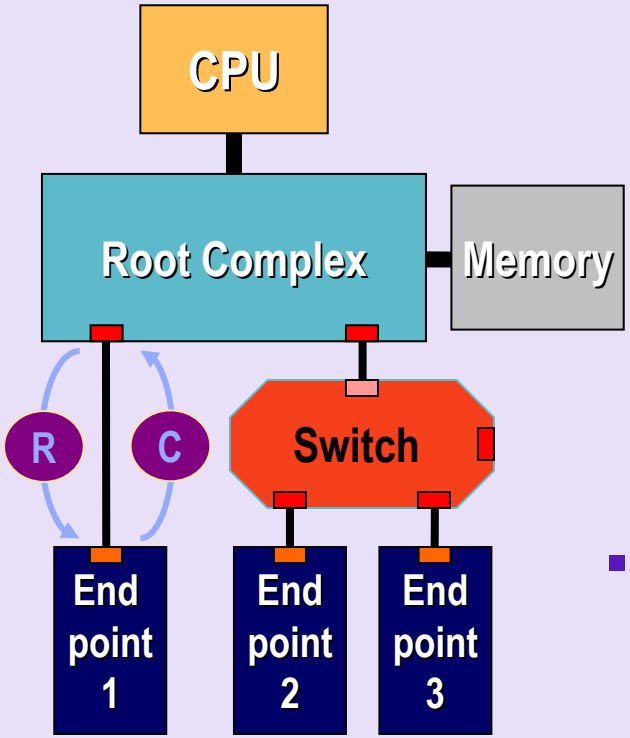
Copyright © 2005, PCI-SIG, All Rights Reserved

PCIe Error Reporting Basics: Error Types and Signaling Mechs

- PCIe error **types**
  - ✓ Correctable
    - A detected error that has the potential of being corrected
    - Not an indication that the error has actually been corrected yet!
    - If uncorrected, will be reported later as an uncorrectable error
  - ✓ Uncorrectable: either Non-Fatal or Fatal
    - Non-Fatal: single transaction failed; hierarchy health assumed OK
    - Fatal: single transaction failed; hierarchy health is threatened
- The three PCIe error **signaling mechanisms:**
  - ✓ Completion Status other than Success: UR, CA, CRS
  - ✓ Error Messages: ERR_COR, ERR_NONFATAL, ERR_FATAL
  - ✓ Error Forwarding, aka Data Poisoning
- Except for Error Forwarding, the error detector logs the error in various status bits and (if implemented) the Advanced Error Reporting (AER) structure

Copyright © 2005, PCI-SIG, All Rights Reserved

PCIe Error Reporting Basics: Posted and Non-Posted Requests



- Non-Posted Requests ("normal split transactions")
  - ✓ Requester sends Request TLP to Completer
  - ✓ Completer sends Completion TLP back to Requester
  - ✓ Includes: all Reads; Config & I/O Writes
  - ✓ Completer uses the Completion Status to signal an error back to the Requester (unless the Completion is poisoned instead)
  - ✓ Completer in addition may send an Error Message to signal the error to the Root Complex
  - ✓ Generally, the Requester decides how to handle the error
- Posted Requests ("fire & forget")
  - ✓ Requester sends Request TLP to Completer
  - ✓ No associated Completion TLP is returned
  - ✓ Includes: Memory Writes; Messages
  - ✓ Completer uses an Error Message to signal an error to the Root Complex
  - ✓ **Root Complex must handle the error since the Requester doesn't know about it!**

Copyright © 2005, PCI-SIG, All Rights Reserved

PCIe Architected Error Handling

- Hardware Containment: System Error
  - ✓ Root Port can be programmed to trigger System Error upon receipt of an Error Message (separate controls each type)
    - ERR_FATAL “obviously” should trigger hardware containment
    - ERR_COR “obviously” should not trigger hardware containment
    - ERR_NONFATAL envisioned not to trigger hardware containment by the original PCIe spec writers, who thought that software could recover from these
      - **This does not work out in practice!**
      - This topic is explored on a subsequent slide
  - ✓ No PCIe architected recovery from System Error other than reboot
  - ✓ Some system vendors implement proprietary hardware containment mechanisms to support recovery without requiring a reboot
- Software can service Error Messages under interrupt/polling
  - ✓ Root Port can be programmed to trigger an interrupt upon receipt of an Error Message (separate controls for each type)
- Non-success Completion Status for PIO Requests
  - ✓ Implementation Note documents returning -1 for PIO reads w/ UR error
  - ✓ An essential feature for “bus” probing/enumeration

Copyright © 2005, PCI-SIG, All Rights Reserved

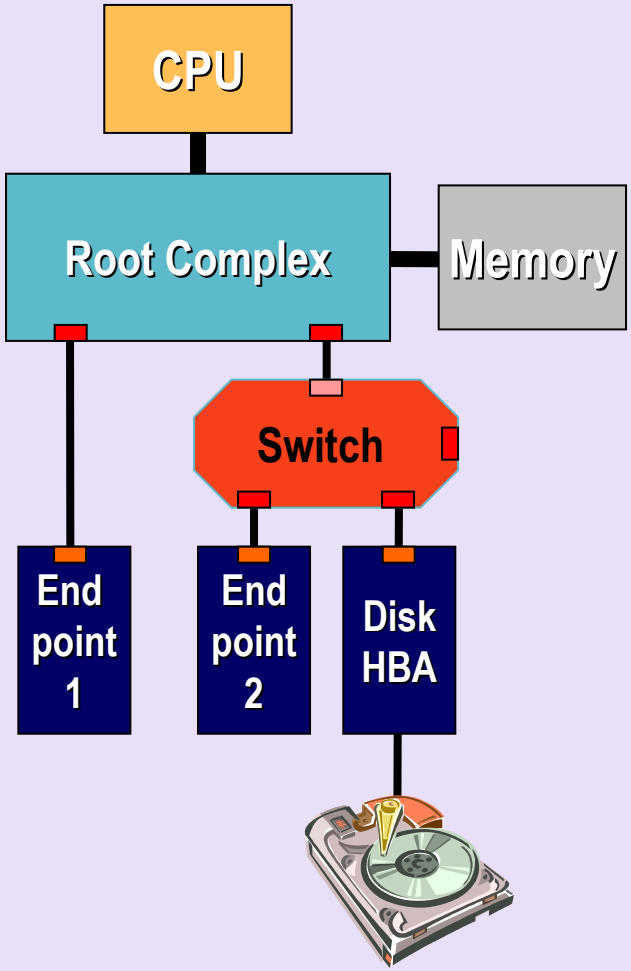
"The Critical Issue" That Needed Fixing

- "Unsupported Request" (UR) error reporting
  - ✓ UR in PCIe corresponds to "Master Abort" in PCI/PCI-X®
  - ✓ For many/most platforms, UR **must not** trigger hardware containment for Config Read/Write transactions (Non-Posted Requests) in order to support normal "bus" probing & enumeration
  - ✓ For platforms supporting robust error handling, UR **must** trigger hardware containment for Memory Write transactions (Posted Requests) in order to avoid silent data corruption
- PCI/PCI-X permit Master Abort to be signaled differently with Non-Posted vs Posted Requests
  - ✓ Non-Posted Requests: signaled via the Completion status (only)
  - ✓ Posted Requests: signaled by asserting SERR#
- PCIe 1.0a components cannot be configured to achieve this
  - ✓ On most platforms, must leave UR reporting totally disabled in order for normal probing to work
  - ✓ Risk silent data corruption if/when Memory Write transactions encounter UR errors

Copyright © 2005, PCI-SIG, All Rights Reserved

Can Software Contain Non-Fatal Errors?

- Assumption in 1.0a was “yes”
  - ✓ Only Fatal errors need be enabled to trigger hardware containment (System Error)
  - ✓ Non-Fatal errors only trigger an interrupt that software processes
- **Does not work out in practice!**
  Example – application doing a disk read:
  - ✓ HBA does DMA write of payload data targeting host memory, but the Root Complex detects a UR error; *host memory does not get updated with payload data*
  - ✓ Root Complex generates internal ERR_NONFATAL, which triggers an interrupt to the OS
  - ✓ Meanwhile the disk read operation continues: HBA does DMA write of completion indication to host memory; HBA generates an interrupt to the driver
  - ✓ There is now a race between the OS tracking down the associated IO operation and the driver reporting the *presumed successful* disk read completion to the OS/app.
  - ✓ If the driver wins, silent data corruption results!
- Bottom line: **Non-Fatal errors must trigger hardware containment** in order to avoid this race condition and potential silent data corruption



Copyright © 2005, PCI-SIG, All Rights Reserved

Preserving the Value of Advanced Error Reporting (AER)

- Initial proposed (simple) fix: cut way back on the error cases logged & signaled by AER
  - ✓ Have AER never log or signal UR/CA for Non-Posted Requests, since the UR/CA error is signaled instead via the associated Completion
  - ✓ Some PWG members rejected this approach, since it dramatically reduced the number of errors the OS can observe via guaranteed architected means
- Adopted fix: “Advisory Non-Fatal Errors” (Advisory NFEs)
  - ✓ Advisory NFEs: errors that should not be reported as uncorrectable by the detecting agent
  - ✓ Components with AER still log and signal as many error cases as before, but signal Advisory NFEs with ERR_COR instead of ERR_NONFATAL
  - ✓ Several other Advisory NFE cases were identified beyond the initial “Completer Sending a Completion with UR/CA Status”. These are covered on a subsequent slide.

Copyright © 2005, PCI-SIG, All Rights Reserved

Other Issues That the ECN Addresses

- Improved tracking of TLPs with poison or bad ECRC
  - ✓ Achieved using new Advisory NFE cases
- Recovery by Requesters when Completions are lost or extremely delayed
  - ✓ Enabled using new Advisory NFE cases
- UR Reporting could not be enabled using “legacy” controls
  - ✓ Compatibility with PCI/PCI-X in this area is now restored
- “Unreasonable” error reporting requirement for Root Ports when their associated Link is down
  - ✓ Specific requirement is now optional
- Clarifications wrt multiple levels of Error Message controls
  - ✓ Discussed on a subsequent slide
- Requester receiving a UR/CA Completion now explicitly prohibited from reporting it using Error Messages
  - ✓ If necessary, Requester must report it using other mechs, e.g., via its driver

Copyright © 2005, PCI-SIG, All Rights Reserved

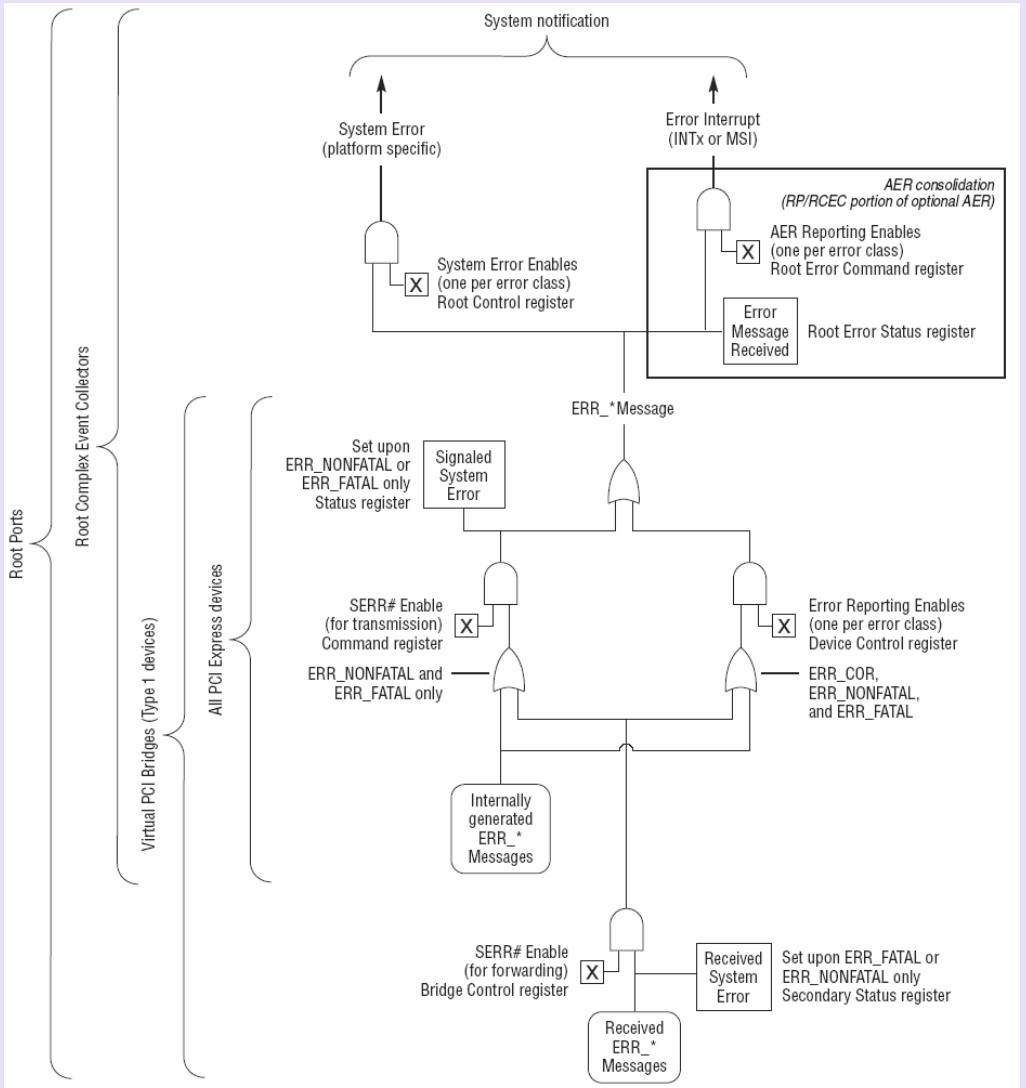
The Five Advisory Non-Fatal Error Cases

- Completer Sending a Completion with UR/CA Status
  - ✓ “The Critical Issue” already covered in this presentation
- Intermediate Receiver
  - ✓ Occurs in Switches that propagate poisoned TLPs or TLPs with bad ECRC
  - ✓ Permits the ultimate Receiver to determine whether hardware containment is triggered
- Ultimate PCIe Receiver of a Poisoned TLP
  - ✓ Occurs in “ultimate PCIe Receivers” that choose to propagate poison instead of triggering hardware containment
- Requester with Completion Timeout
  - ✓ Permits a Requester to attempt recovery from Completion Timeouts if a Completion TLP is mis-routed, dropped, or extremely delayed
- Receiver of an Unexpected Completion
  - ✓ Avoids the Receiver from interfering with the previous case

Copyright © 2005, PCI-SIG, All Rights Reserved

Error Message Controls: New Figure

- Figure 6-3 in 1.1 Base spec
- “Sometimes a picture says a thousand words”
- Complements/reinforces text clarifications made in 1.0a errata
- Notable clarifications:
  - ✓ Which controls exist in which components
  - ✓ Which control bits control which Error Messages
  - ✓ Interactions between legacy and new PCIe controls
- One significant change:
  - ✓ Forwarded ERR_* Message transmission can be enabled via legacy or new PCIe bits
  - ✓ Transmission previously enabled only via legacy bits



Copyright © 2005, PCI-SIG, All Rights Reserved

Key Hardware Impacts by the ECN

- Completer logic must change slightly in handling NFEs with Non-Posted Requests
- Intermediate Receiver (e.g., Switch) logic that detects TLPs with poison or bad ECRC must change slightly
- The *SERR# Enable* bit in Command reg now implicitly enables UR reporting even when the *UR Reporting Enable* bit in the Device Control reg is clear
- Control logic in virtual PCI Bridges (Root Ports & Switches) must change slightly regarding the transmission of Error Messages forwarded from the secondary interface to the primary interface
- For devices implementing AER, there is a new *Advisory NFE* bit in the Correctable Error Status & Mask regs
- A new capability bit, *Role-Based Error Reporting*, has been assigned in the Device Capabilities reg to indicate to software if a device implements the semantics defined by this ECN

Copyright © 2005, PCI-SIG, All Rights Reserved

Key Software Impacts by the ECN

- Legacy software:
  - ✓ Key error-reporting compatibility with PCI/PCI-X is now restored
- 1.0a-based PCIe-aware software:
  - ✓ Software that intentionally leaves UR Reporting disabled **is not broken**, but must change in order to take advantage of UR reporting with Posted Requests
  - ✓ Software that assumes all ERR_COR messages report Link-level correctable errors **is not broken**, since Advisory NFEs are masked by default
- Software aware of Role-Based Error Reporting:
  - ✓ Must check the *Role-Based Error Reporting* bit in devices before relying on the new semantics
  - ✓ If desired, can enable Advisory NFE logging/signaling on devices that implement AER

Copyright © 2005, PCI-SIG, All Rights Reserved

PCIe Bridges & Role-Based Error Reporting

- The current PCIe to PCI/PCI-X Bridge spec is still based on PCIe Base 1.0a
- "The Critical Issue" that drove the ECN for the Base spec is not as critical with PCIe Bridges
  - ✓ Posted Requests that Master Abort on the secondary can be signaled correctly with SERR#, without breaking probing
- PCIe Bridges would benefit somewhat with Role-Based Error Reporting semantics
  - ✓ Silent data corruption is currently a risk for downstream Posted Requests that encounter a UR error on the primary of a PCIe Bridge
  - ✓ A new Advisory NFE case for PCIe Bridges would improve parity/ECC error tracking slightly
- Currently there is no plan to create an Error Reporting ECN for the PCIe Bridge spec, nor update the PCIe Bridge spec to be based on PCIe Base 1.1

Copyright © 2005, PCI-SIG, All Rights Reserved

Recommendations

- PCIe component vendors:
  - ✓ Base all new components on the PCIe Base 1.1 spec, which already incorporates Role-Based Error Reporting
  - ✓ Review the 1.0a ECN and/or 1.1 carefully to ensure adherence to the new semantics and many clarifications of the old semantics
  - ✓ Implement Advanced Error Reporting (AER)
- PCIe software vendors:
  - ✓ Update old software and base new software on Role-Based Error Reporting semantics
  - ✓ Check the *Role-Based Error Reporting* bit, and continue to support components without the new semantics

Copyright © 2005, PCI-SIG, All Rights Reserved

Presentation Conclusions

- The ECN fixes a critical error reporting issue involving silent data corruption, restoring compatibility in this area with PCI/PCI-X
- The ECN addresses several other error reporting issues
- The ECN preserves the value of Advanced Error Reporting
- The ECN makes many clarifications of existing error reporting semantics
- ECN hardware impacts are challenging conceptually, but relatively modest as far as actual changes
- New hardware is backwards compatible with both legacy and 1.0a-based PCIe-aware software

Copyright © 2005, PCI-SIG, All Rights Reserved

Thank you for attending the
PCI-SIG Developers Conference 2005.

For more information please go to
www.pcisig.com

Copyright © 2005, PCI-SIG, All Rights Reserved


PCI
SIG®