Reliability and Serviceability Features in a PCIe® Controller

Trupti Gowda

PLDA

Disclaimer

Presentation Disclaimer: All opinions, judgments, recommendations, etc. that are presented herein are the opinions of the presenter of the material and do not necessarily reflect the opinions of the PCI-SIG®.

Summary

- **Introduction**
- **Test features**
- **Diagnostic features**
- **Serviceability features**
- **Usability**
- **Conclusion**

Introduction

- **Implementing testing and debugging features is nothing new to PCI Express® device designers.**

  PCIe® protocol contains several of these features and improvements and new mechanisms are regularly added to the specification. Industry actors are also constantly on the look-out for features that can make their devices more reliable and easier to test and debug if needed.

- **This presentation will provide a guided tour of some of these features and discuss their benefits versus drawbacks.**

Introduction

- **These features include:**

  - Test features
    .. to create certain situations to check device / system behavior

  - Diagnostic features
    .. to report problems and help find the causes

  - Serviceability features
    .. to contain, fix or work around the problem

Summary

- Introduction
- **Test features**
- Diagnostic features
- Serviceability features
- Usability
- Conclusion

Link State Changes

- **LTSSM has many states and state transitions**

- **A number of them can be reached / exercised with PCIe specification defined configuration registers:**

Feature	Downstream port	Upstream port
Disabled	✓	n/a
Hot Reset	✓	n/a
Polling.Compliance	✓	✓
Link retraining	✓	
Redo equalization	✓	n/a
Change link speed	✓	
Low power states	✓(as applicable)	✓(as applicable)

Link State Changes

- **However there is no standard way to exercise some features:**

Feature	Downstream port	Upstream port
L2	✗	n/a
Loopback	✗	✗
Request equalization	n/a	✗
Link retraining	✓	✗
Change link width	✗	✗

- **It is important to implement registers to be able to control these features: this is both easier and much cheaper than other solutions.**

Error Injection

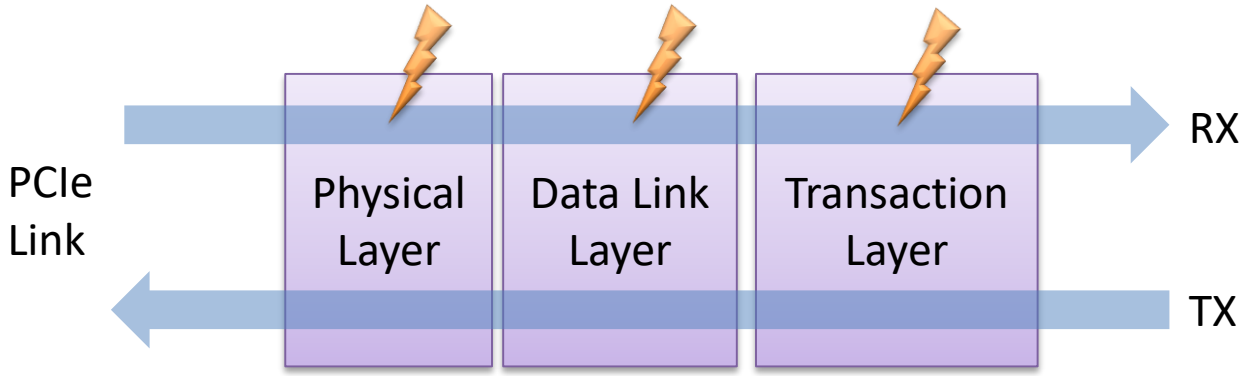
- **Error Injection possibilities are nearly endless**

- **Injecting errors properly can be complex:**
  - Quite often it is complex to introduce expected error, and just this error.
  - Some errors can create deadlocks, have undesirable side-effects…
  - As a result cost in logic & verification can be high

- **So their scope must be clearly defined!**

Error Injection

- **Error injection on Receiver side has many benefits:**
  - Can help test hard to produce error cases
  - Test error detection / reporting, and device logic behavior
  - Help test device firmware / software



Error Injection

- **Example: Adding ability to trigger all applicable AER uncorrectable & correctable errors:**
  - Goal was to be able to test all these errors and check how logic & system software would handle them
  - And REALLY cause the error to happen, not just set corresponding AER register!

- **One of the most challenging error to inject cleanly was ‘Unsupported Request’:**
  - Adding a TLP to the receive flow can be complex and have various side effects, so must modify a received TLP.
  - UR rules are different for upstream / downstream ports, and vary based on TLP type
  - Cannot be done on most TLPs without risk of causing another type of error

  - Solution:
    - Downstream port : replace next received MRd with MRdLk
    - Upstream port : replace next received Cpl/CplD with CplLk/CpldLk

Summary

- Introduction
- Test features
- **Diagnostic features**
- Serviceability features
- Usability
- Conclusion

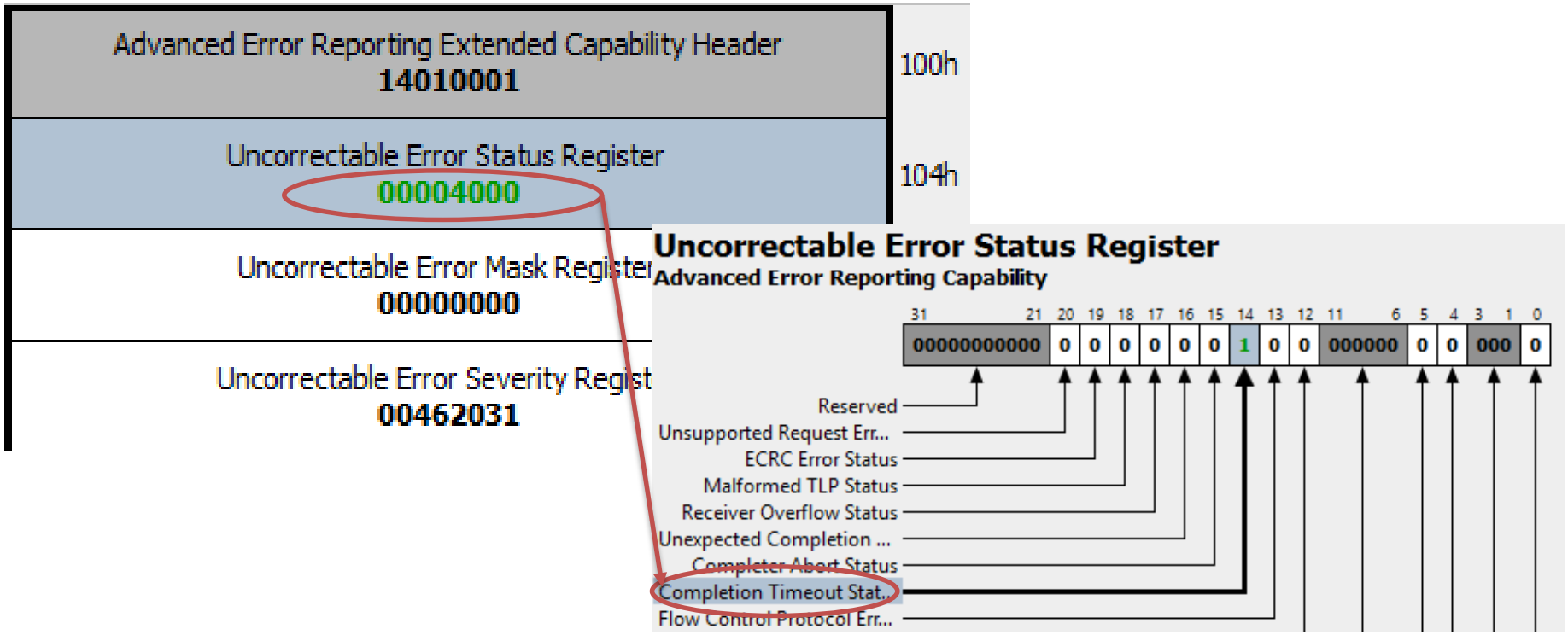
Advanced Error Reporting

- **AER is an optional feature of PCIe protocol that provides detailed information about errors that occur in a device:**
  - Separate status/mask/severity bit for each type of error
  - Offending TLP header is logged when applicable
  - Supports internal (i.e. application specific) errors

Advanced Error Reporting

Example : “My device performs R/W to system memory, but it stops transferring data after running for several minutes…”

AER registers report that a Completion Timeout occurred:



Advanced Error Reporting

- **AER gate count may be high in some configurations**

- **Example: a device with 500 Virtual Functions:**

  AER includes status / mask / severity / header log registers, ~200 flip-flops in each functions

  200 x 500 = 10000 register bits just for AER registers!

Recovery Reason

- **Unexpected entry in Recovery is a very common problem, but there can be multiple reasons for this.**

- **A protocol analyzer link trace and some configuration space registers error/status bits can give some clues, but quite often identifying the reason is challenging.**

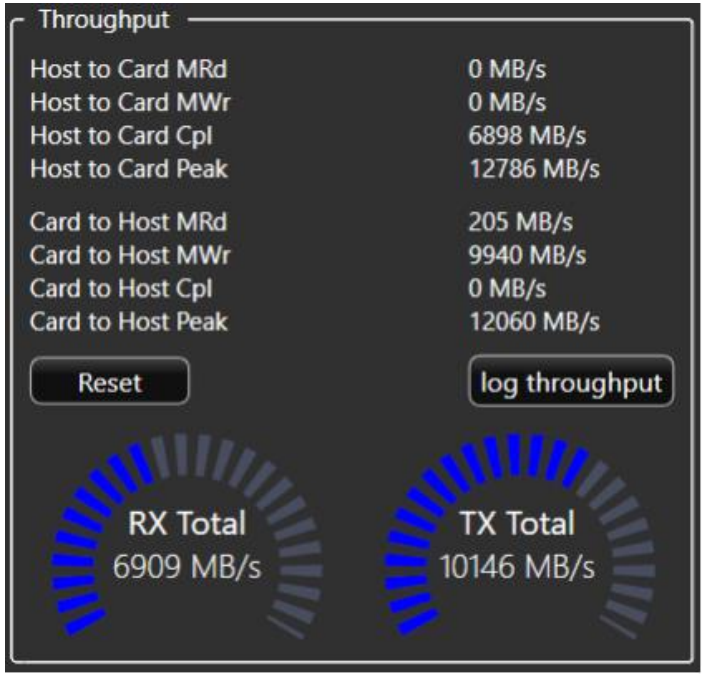
Recovery Reason

- **Having an internal source that indicates why the device entered Recovery is invaluable:**

  - 1: retrain_link bit is set
  - 2: directed to perform autonomous link width change
  - 3: directed to perform autonomous link speed change
  - 4: directed to enter Hot Reset
  - 5: link_disable bit is set
  - 6: replay error
  - 7: flow-control timeout
  - 8: deskew error
  - 9: framing error
  - 10: receiver encoding error
  - 11: request equalization
  - 12: TS1/2 received from link partner
  - 13: EIEOS received from link partner
  - 14: N_FTS timeout
  - 15: Electrical idle inferred on all lanes in L0 and no EIOS was received
  - 16: Directed to do speed change

Performance Measurement

- **Having a way to measure RX / TX throughput is important to have:**
  - Helps check that device / system performances are as expected
  - Lower than expected throughput can reveal unsuspected problems
  - Makes it possible to measure the effect when changing some settings

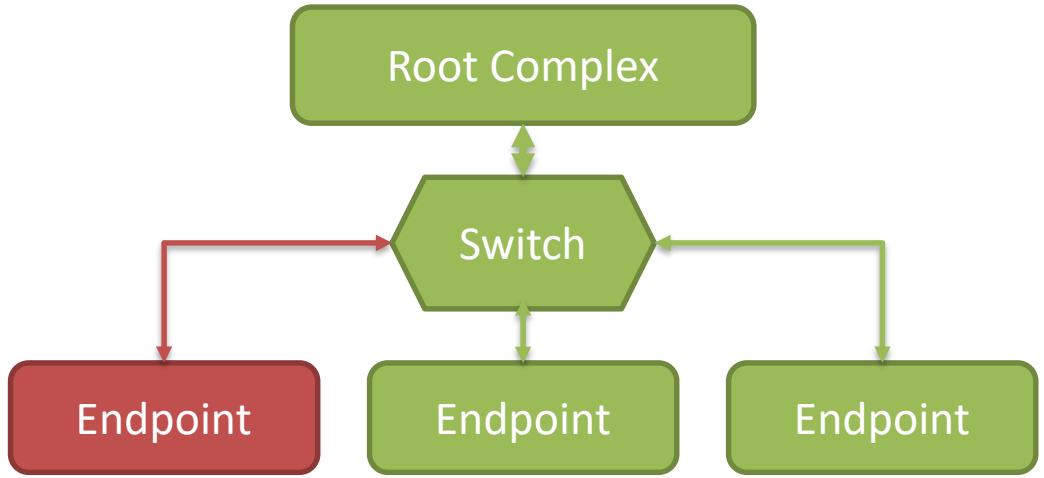


Summary

- Introduction
- Test features
- Diagnostic features
- **Serviceability features**
- Usability
- Conclusion

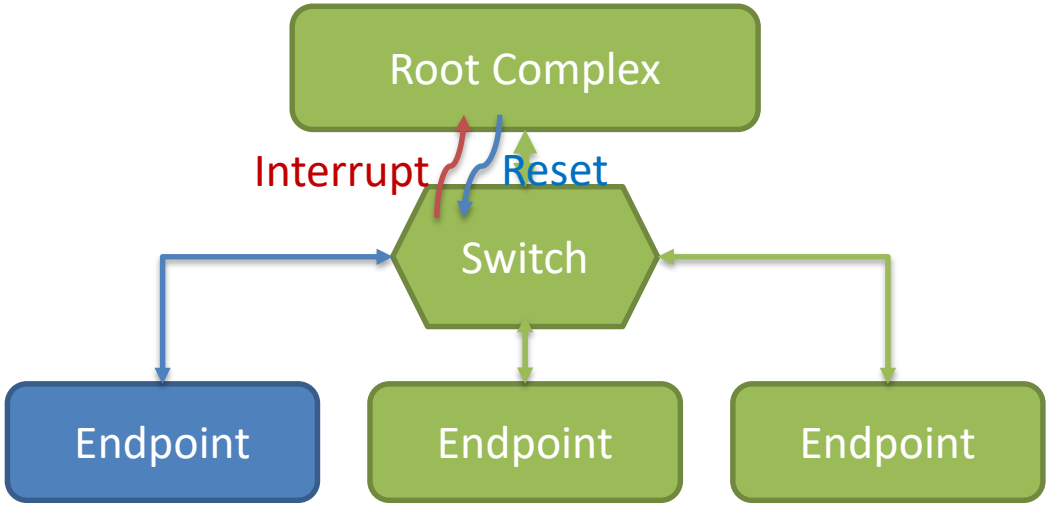
Downstream Port Containment

- **DPC is an optional feature of PCIe protocol that can automatically disable a link below a Downstream Port when an uncorrectable error occurs:**
  - This isolates the faulty link or device
  - Can prevent the potential spread of errors & data corruption



Downstream Port Containment

- Event is reported to Root Complex by an interrupt
- System may reset faulty link / device



Programmable Timers (LTSSM)

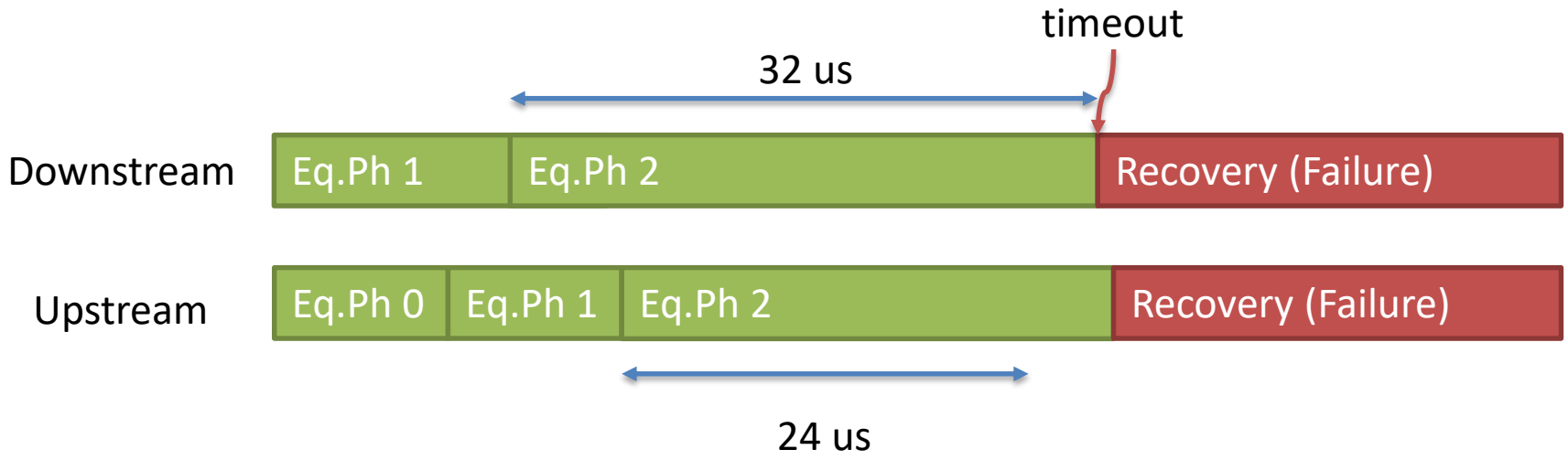
- **PCIe® specification defines many delay and timeout values for LTSSM operations, usually with a -0%/+50% tolerance.**

- **It happens occasionally that two devices fail to link-up or operate properly because one of them has a problem with some of these delays or timeout values.**

- **This can be due to a variety of reasons :**
  - Design issue
  - PHY issue
  - Clock issue

Programmable Timers (LTSSM)

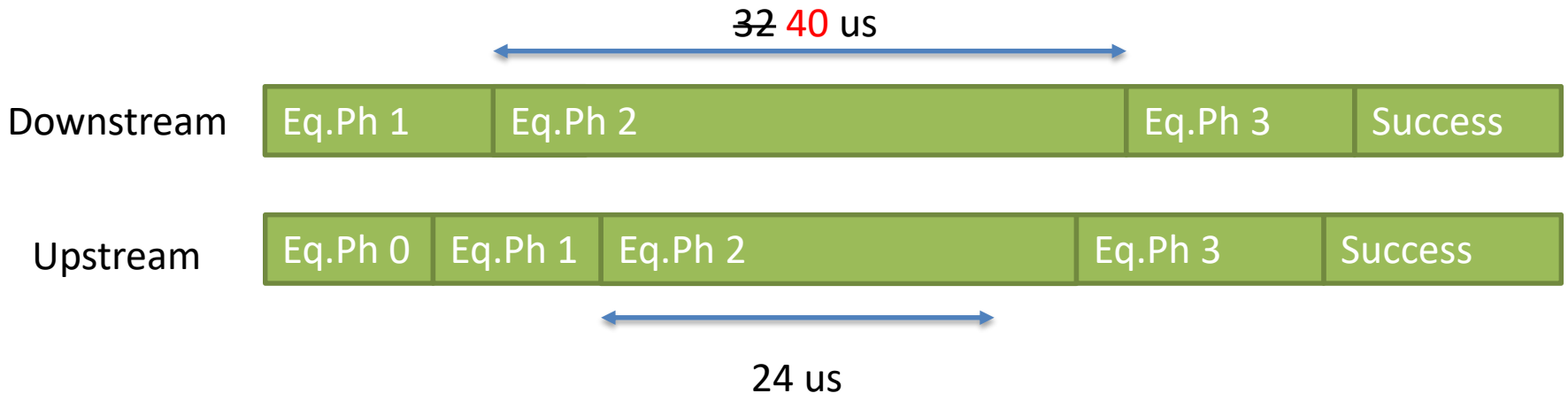
- **Example: Downstream port times out in Equalization because Upstream port takes too long in Equalization phase #2**



- **Result: equalization always fails and there is no way to get the link operate at expected speed.**

Programmable Timers (LTSSM)

- **Having the ability to extend Equalization Phase #2 timeout on Downstream port side can solve the issue :**



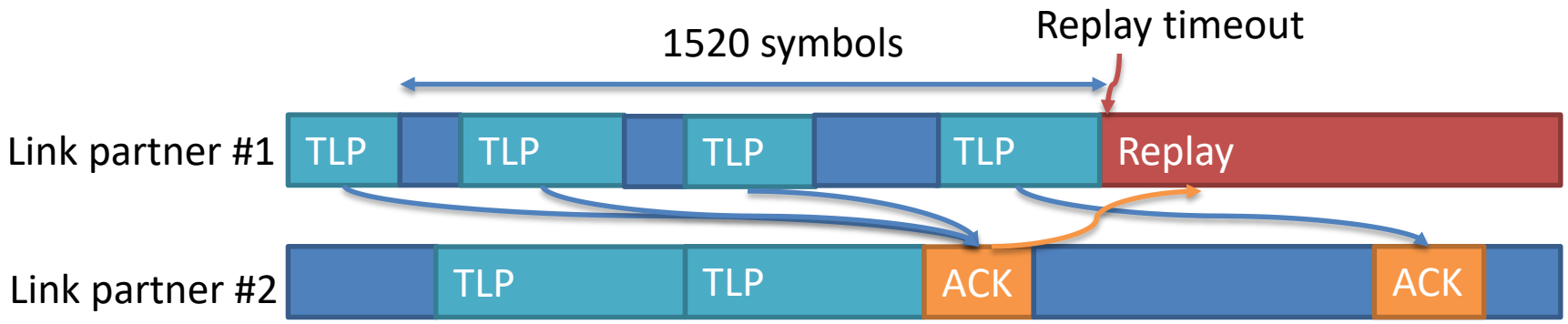
Programmable Timers (LTSSM)

- **We have currently identified more than 28 values that can help troubleshoot interoperability issue:**

  - **Detect.Quiet timeout**
  - **Detect.Active timeout**
  - **Polling.Active timeout**
  - **Polling.Compliance timeout**
  - **Polling.Config timeout**
  - **Config.LinkWidthStart timeout**
  - **Config.LinkWidthAccept timeout**
  - **Config.LaneNumWait timeout**
  - **Config.LaneNumAccept timeout**
  - **Config.Complete timeout**
  - **Config.Idle timeout**
  - **Recovery.ReceiverLock timeout**
  - **Recovery.Speed timeout**
  - **Recovery.ReceiverConfig timeout**
  - **Recovery.Idle timeout**
  - **Recovery.Equalization (upstream phase 3) timeout**
  - **Recovery.Equalization (upstream phase 2) timeout**
  - **Recovery.Equalization (upstream phase 1) timeout**
  - **Recovery.Equalization (downstream phase3) timeout**
  - **Recovery.Equalization (downstream phase2) timeout**
  - **Recovery.Equalization (downstream phase1) timeout**
  - **Recovery.Equalization (downstream phase0) timeout**
  - **Disabled timeout**
  - **Loopback.Entry to Loopback.Exit timeout (loopback master)**
  - **Loopback.Entry to Loopback.Active timeout (loopback master)**
  - **Loopback.Exit timeout**
  - **Hot Reset timeout**

Programmable Timers (Tuning)

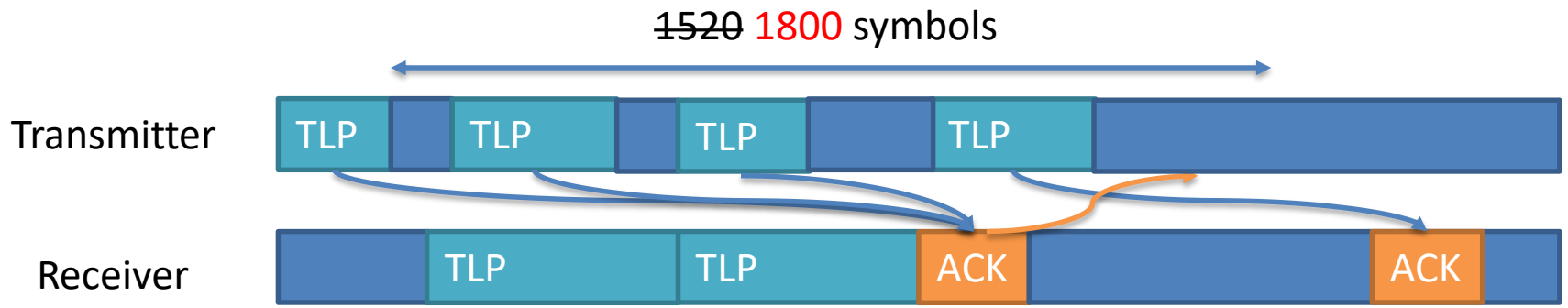
- **Some other delays are not critical for proper operation but may be tuned to overcome performance loss issues.**
- **Example: Unwanted replays happen because link partner sometimes takes too long to transmit ACK after receiving TLP(s):**



- **Can be due to design issue, larger than expected PHY latency..**

Programmable Timers (Tuning)

- **Increasing replay timer timeout value overcomes this problem easily and performance loss is eliminated:**



Programmable Timers (Tuning)

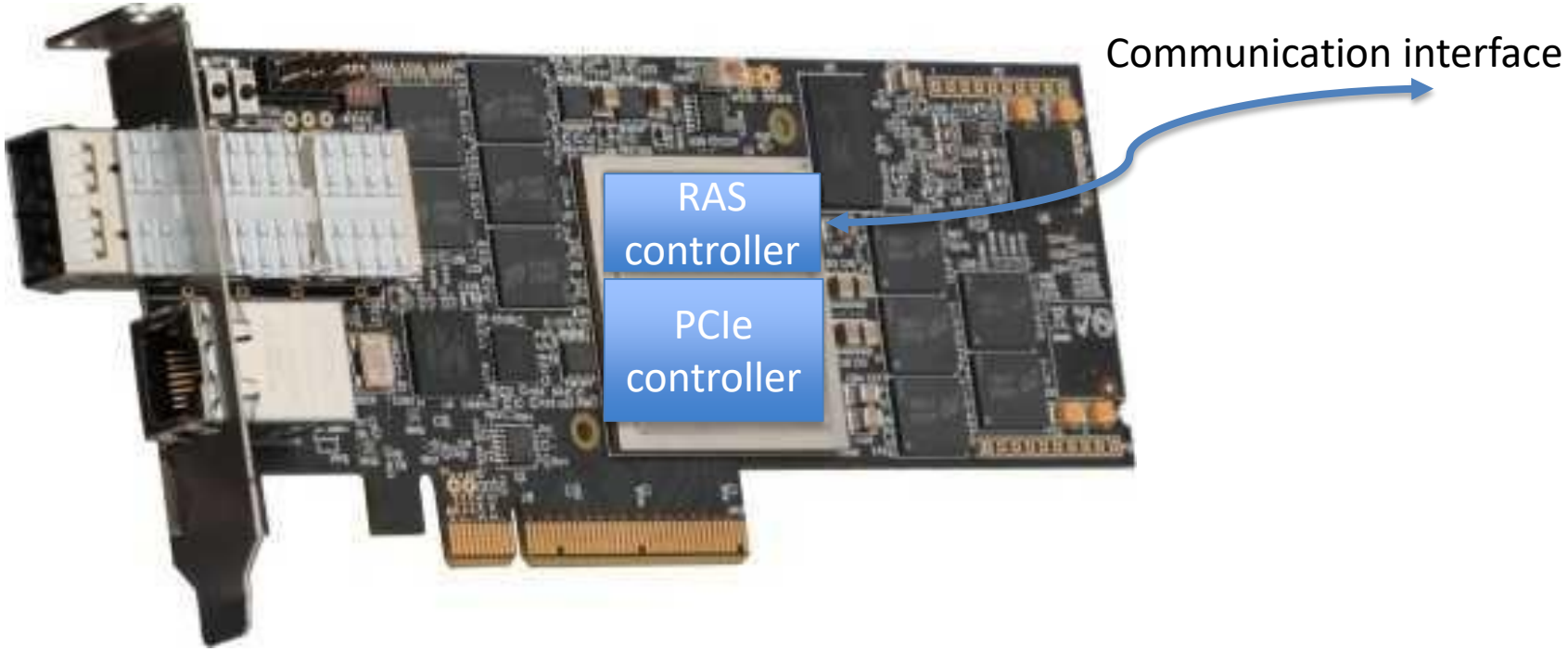
- **Replay timer timeout**
  - Increasing value can prevent undesirable replays from occurring
- **Max delay to transmit Flow-control updates**
  - Decreasing delay can help free credits more quickly
- **ACK/NAK latency timeout**
  - Decreasing values can help link partner free replay buffer space more quickly
- **Delay to enter low-power states (ASPM L0s / L1)**
  - Can be tuned to make low-power less/more aggressive

Summary

- Introduction
- Test features
- Diagnostic features
- Serviceability features
- **Usability**
- Conclusion

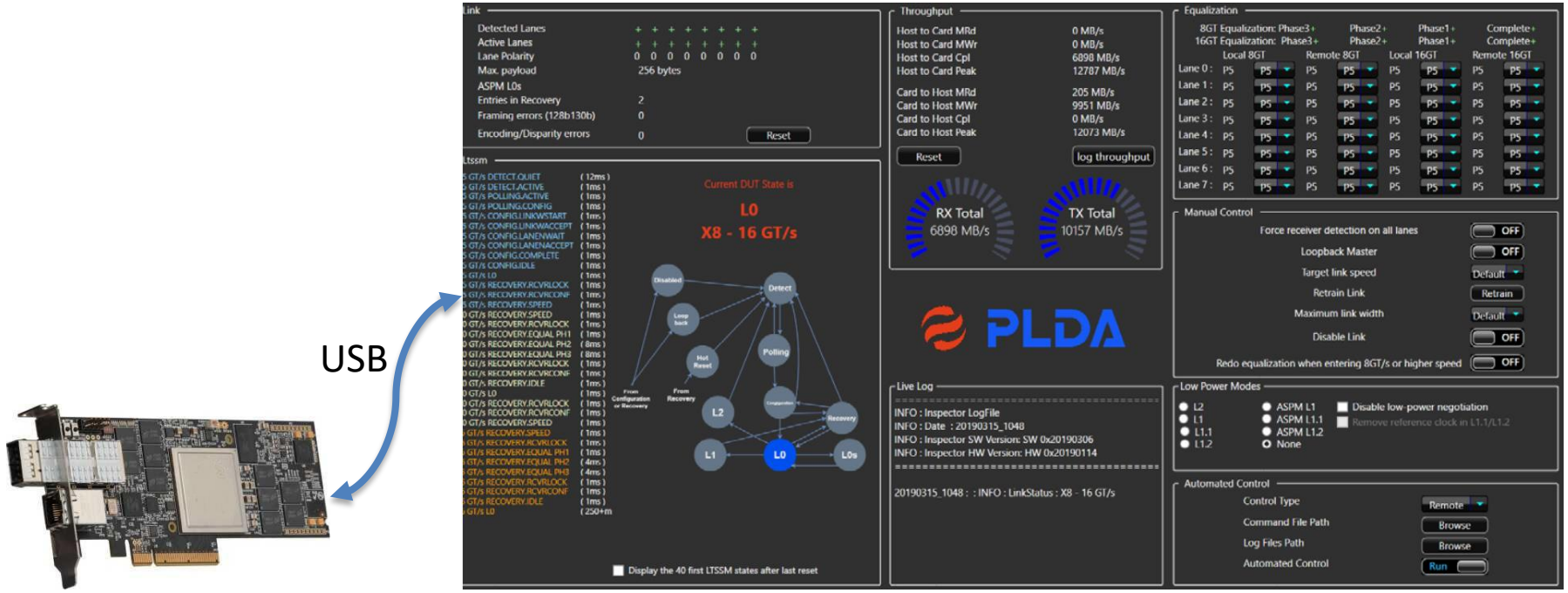
Usability

- **Implementing a communication interface that can provide easy access to RAS features is a valuable feature:**



Usability

- **These features can become even more powerful and useful with an interface to monitor & control the device:**



Summary

- Introduction
- Test features
- Diagnostic features
- Serviceability features
- Usability
- **Conclusion**

Conclusion

- **Pros:**
  - Better testing
  - Much easier & faster debugging
  - Ability to workaround / contain issues

- **Cons:**
  - Higher design effort & gate count
  - More verification effort
  - Often difficult to use (but does not have to !)

Thank you for attending the PCI-SIG Developers Conference Israel 2019.

For more information please go to

www.pcisig.com