PCIe® Controller Design Architectural Challenges

Anujan Varma

Cadence Design Systems

Disclaimer

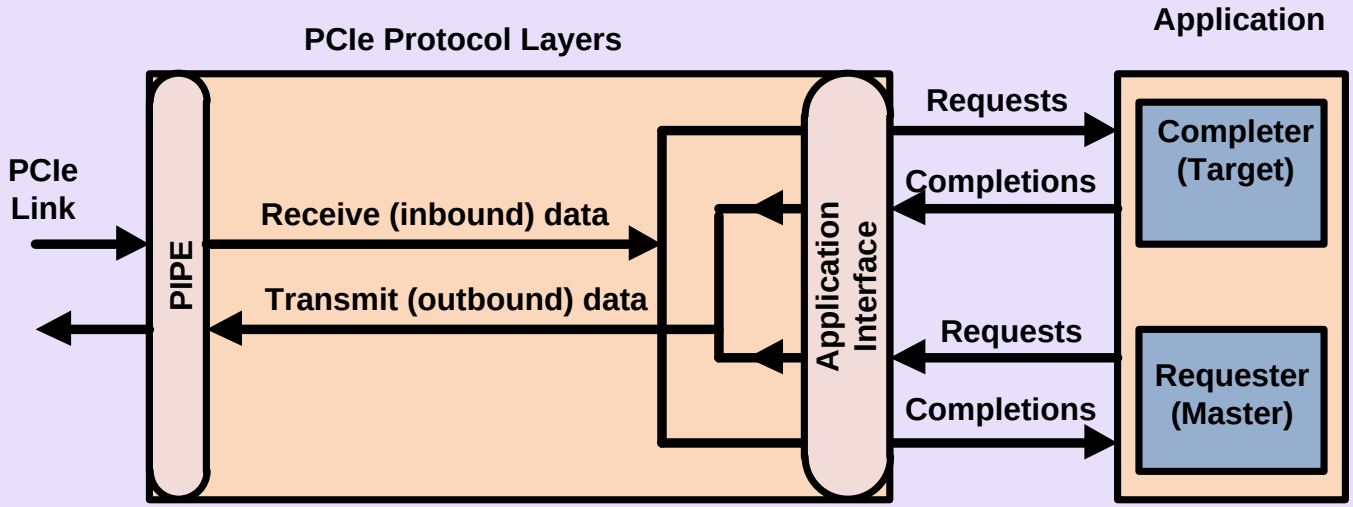
Presentation Disclaimer: All opinions, judgments, recommendations, etc. that are presented herein are the opinions of the presenter of the material and do not necessarily reflect the opinions of the PCI-SIG®.

Motivation

- PCIe 2.x x16 designs may not scale well to PCIe 3.0
  - ✓ Designs with 128-bit data paths need to run at a minimum clock rate of 1 GHz
  - ✓ Outside the constraints of most current ASIC starts
- 500 MHz operation requires 256-bit data paths
  - ✓ Also applies to PCIe 3.0 x8 designs running at 250 MHz
- Introduces new architectural challenges

Terminology

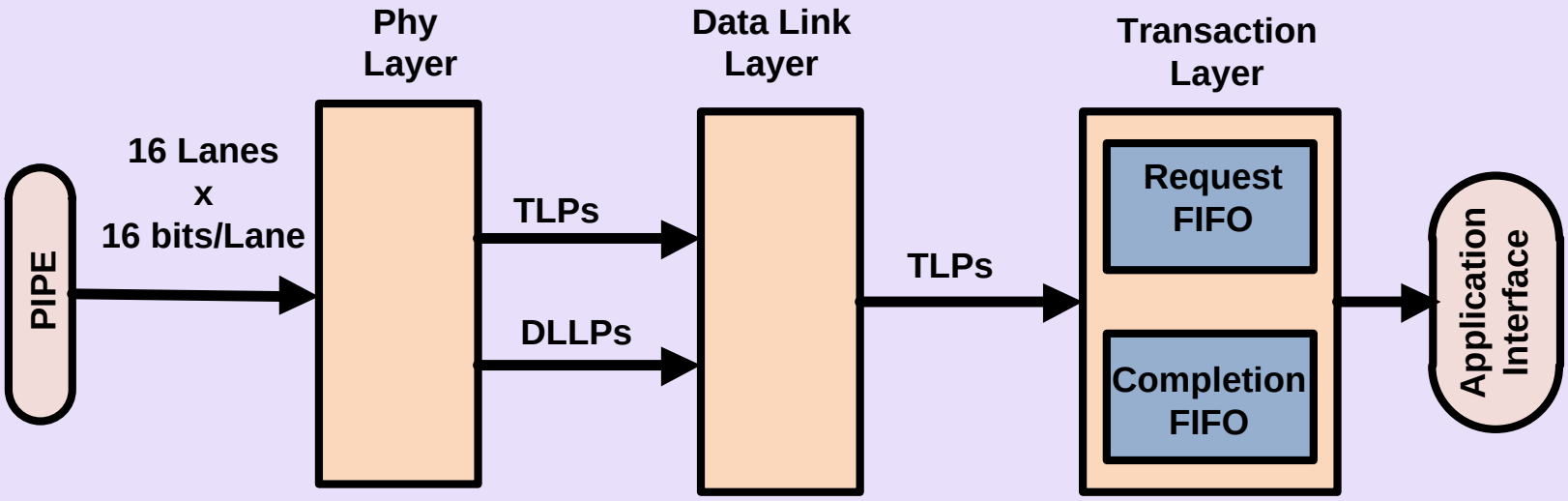
- Receive (inbound) data: Data received from the link and delivered to application
- Transmit (outbound) data: Data sent by the application to the link
  - ✓ Data may be Requests or Completions



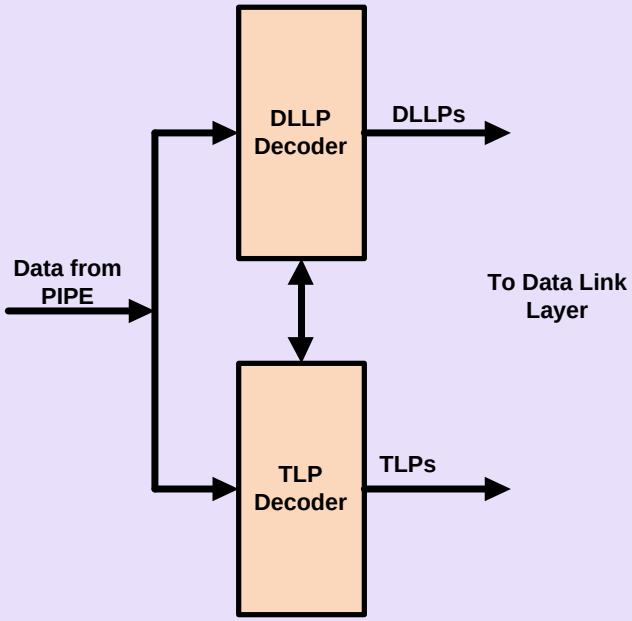
Challenges

- Need to process multiple TLPs and DLLPs per clock cycle
  - ✓ Receive data path must support full bandwidth from PIPE to Transaction Layer
  - ✓ Transmit data path can be designed for single TLP per cycle, with some performance impact
- Maintaining transaction order
- Receive FIFO organization
- Avoiding performance bottlenecks at client application interface

Typical Receive Datapath



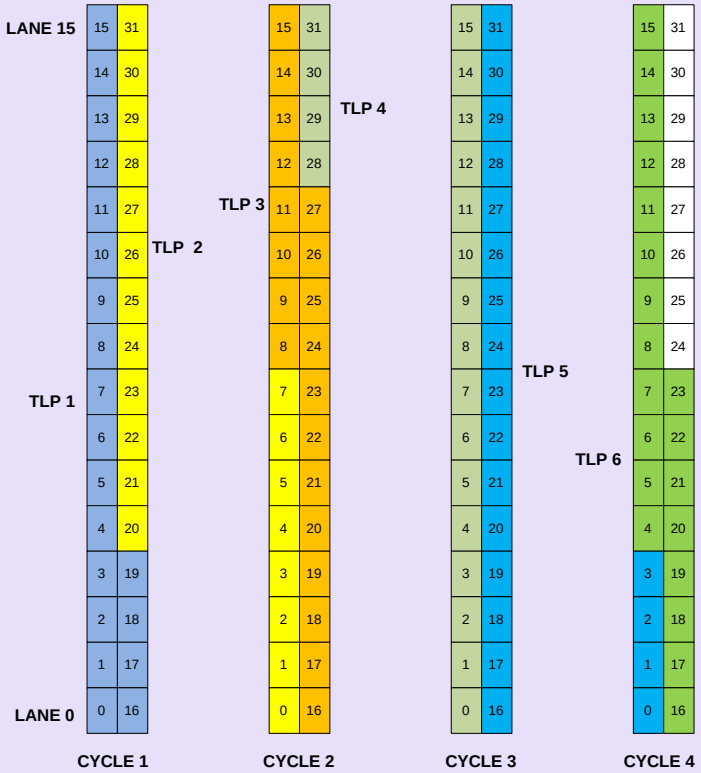
Decoding TLPs and DLLPs



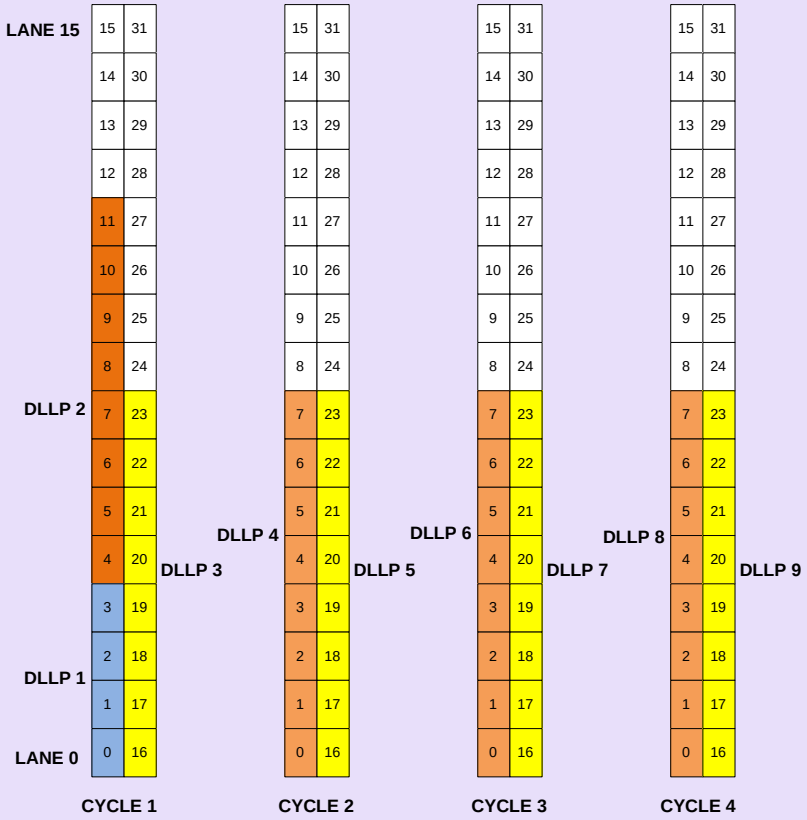
- TLPs and DLLPs can be decoded separately
  - ✓ Some state exchanged between the decoders
- Single TLP, dual DLLP per cycle adequate for 128-bit data path

TLP Processing in a x16 Device

- Smallest TLP is 5 Dwords (20 bytes) long
- TLP decoder must forward more than one TLP to Data Link Layer in some cycles

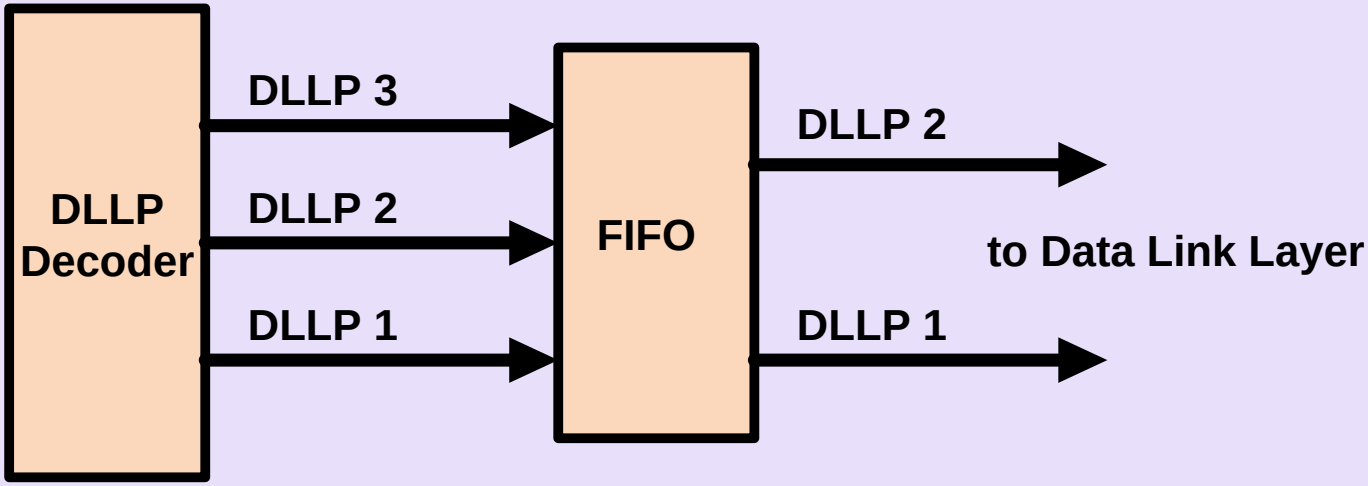


DLLP Processing in a x16 Device



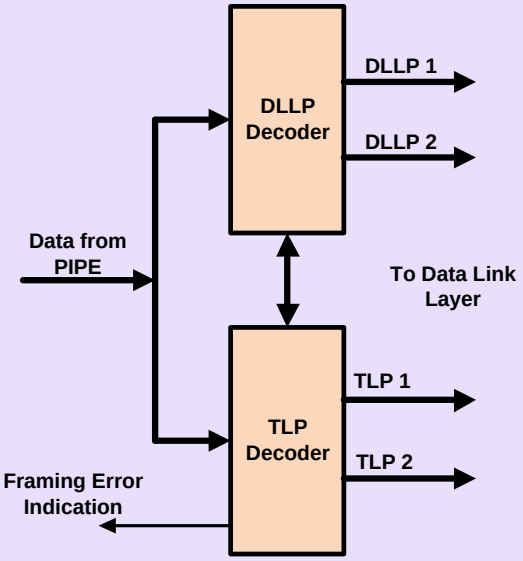
- Maximum of two DLLPs can start in one cycle
- May need to process three DLLPs in some cycles

DLLP Processing in a x16 Device



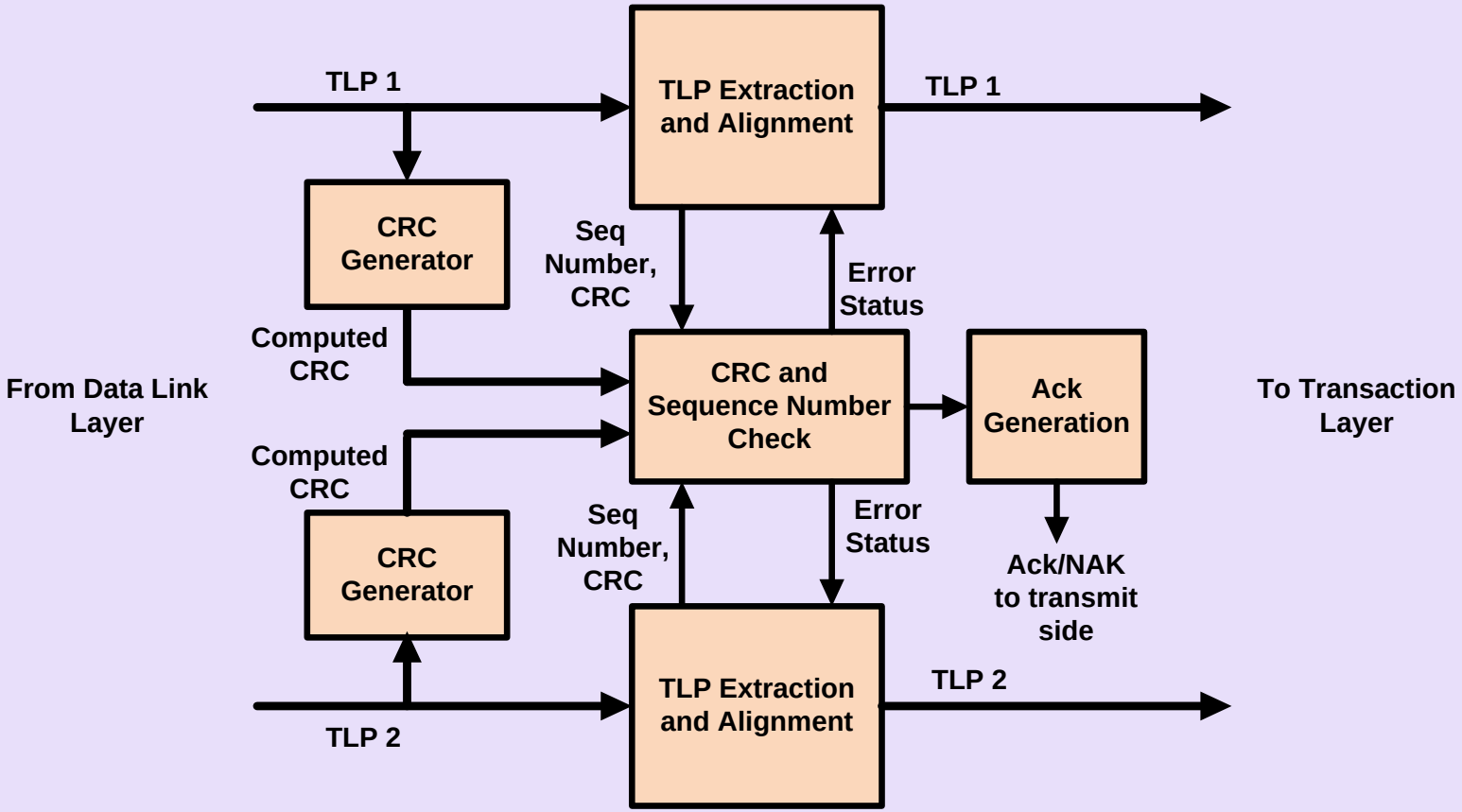
- DL needs to process max 2 DLLPs per cycle
- No additional complexity compared to 128-bit data path

TLP/DLLP Decoder in x16 Device



- TLP 2 may start in the same cycle when TLP 1 ends
  - ✓ Two TLPs may straddle same bus or could have separate buses
- Only one TLP in progress when no SOP/EOP present

Receive TLP Processing in Data Link Layer



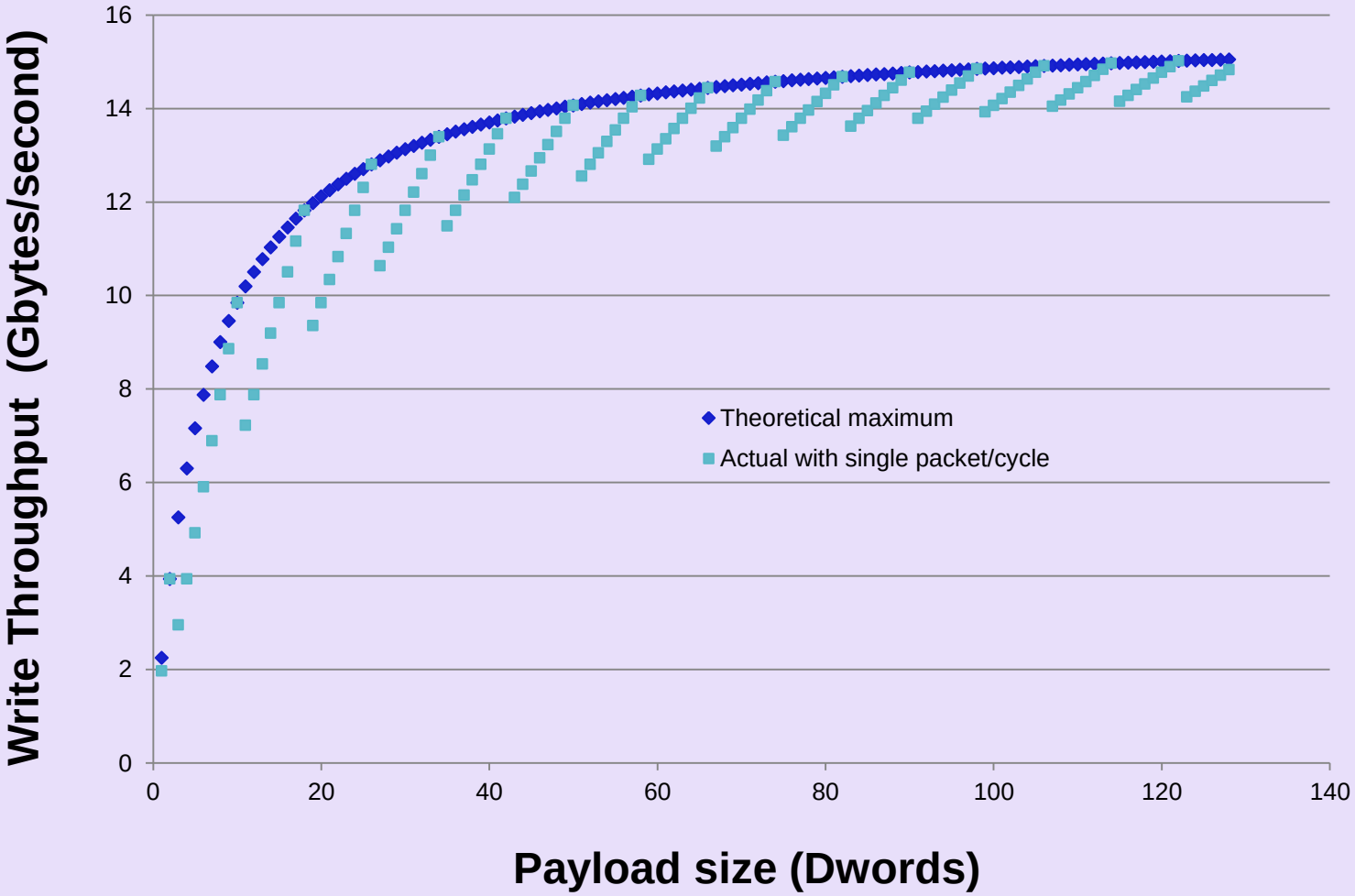
Transmit Data Path Design

- Can work with single TLP per cycle with some throughput degradation
  - ✓ Performance impact depends on TLP size and nature of traffic
- Dual DLLPs per cycle easy to support

Impact of Single-TLP Transmit Path on Throughput

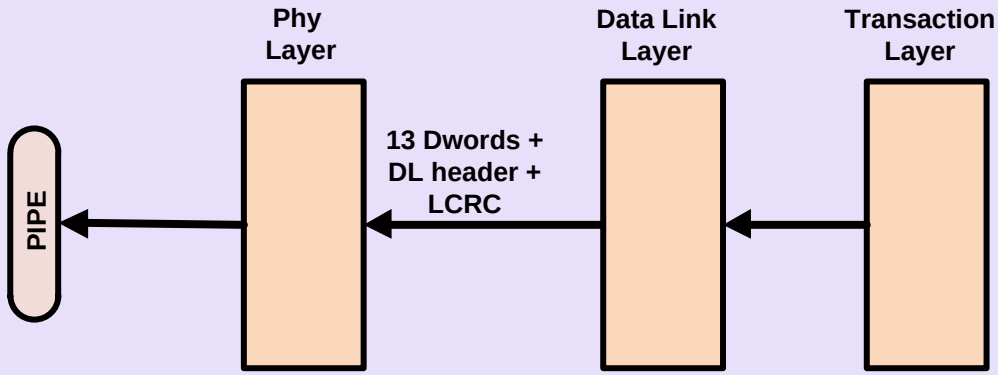
- Memory Writes with 64-bit address, no ECRC
- Ignore DLLPs and SKPs
- Max theoretical throughput can be achieved with dual-TLP transmit path
  - ✓ Throughput = B/(24+B) * (128/130) * 16 Gbytes/sec, where B = payload size in bytes
- Throughput for single TLP/cycle calculated from number of cycles to send TLP.
- Actual difference will be less because DLLPs can fill the gaps between TLPs

Impact of Single-TLP Transmit Path on Throughput



TX Performance with Wider Data Paths

- Throughput of single-TLP pipeline can be improved by widening the data path between TL and the phy layer.

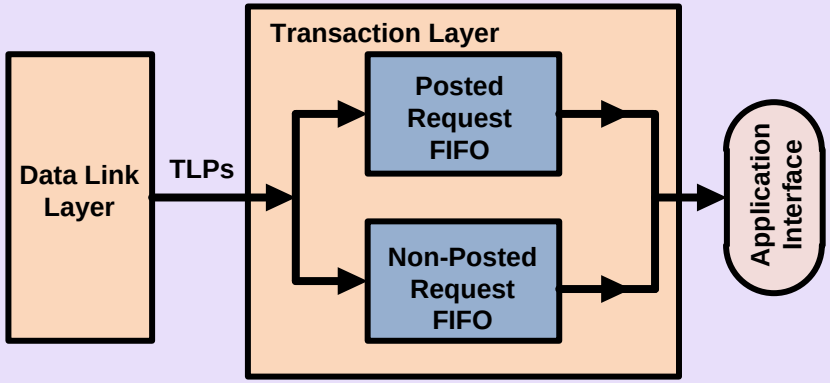


- ✓ Achieves theoretical maximum throughput when payload size > 1 Dword.
- ✓ May increase complexity of CRC computations

TX Performance with Bidirectional Traffic

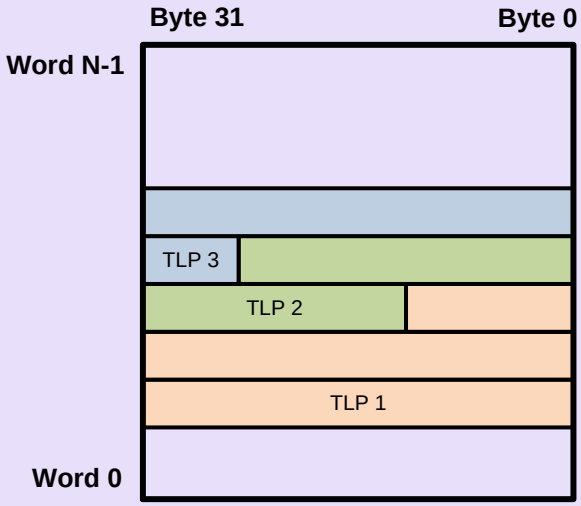
- Throughput impact is less with traffic in both directions
  - ✓ Acks and FC Updates can be scheduled in gaps between TLPs
- Actual throughput depends on Ack frequency and credit update freqeuency.

Receive FIFO Organization



- Request FIFO must be split into two physical RAMs for Posted and Non-Posted.
  - ✓ Single RAM with partitions adequate with 128-bit designs
- TLPs may straddle the same word in RAM

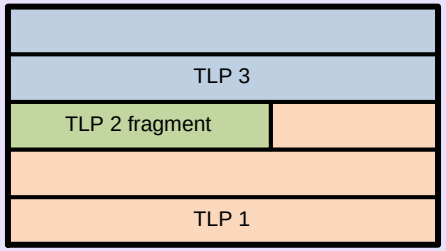
FIFO RAM Structure



- How to discard TLPs with errors, or duplicates?
  - ✓ Some errors won't be detected until the last payload cycle.
- Read-modify-writes not practical

Discarding from Receive FIFO

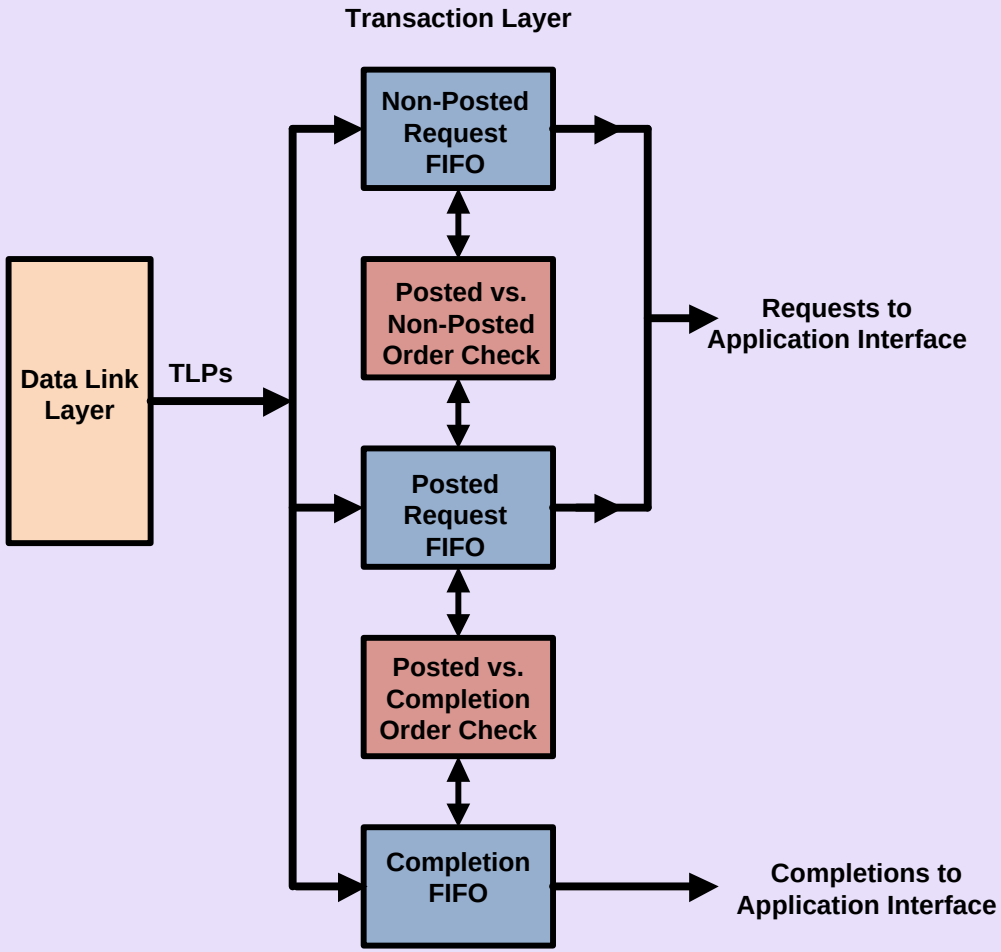
- Solution 1: Discard bad TLPs before reaching FIFO.
  - ✓ Requires single-TLP store-and-forward buffer before FIFO
  - ✓ Adds latency
- Solution 2: Leave fragments of discarded TLPs in FIFO and maintain state



Maintaining Transaction Order

- Dual TLPs makes order checking more complex
- Two primary checks on RX side
  - ✓ Posted vs. Non-Posted
  - ✓ Posted vs. Completions
- Must support ID-based ordering

Checking Transaction Order



Ordering of Posted Versus Non-Posted Requests

- Posted requests must be allowed to proceed out of order when Non-Posted requests are blocked at the application interface
  - ✓ Prioritizing Posted over Non-Posted can lead to starvation
  - ✓ Logic must handle two requests queued and two requests delivered in the same cycle

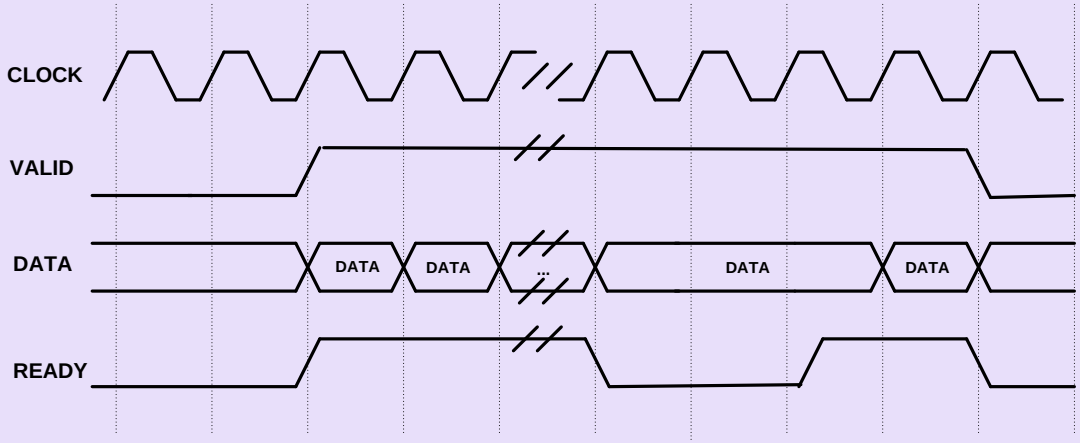
Ordering of Posted Requests Versus Completions

- Completions must maintain ordering with Posted requests when strict ordering is required.
- ID-based ordering check requires RIDs to be matched
  - ✓ One comparator for each Posted request in the FIFO
- Logic must handle two Posted requests/ Completions queued/delivered in the same cycle

Application Interface Design

- Application interfaces can become bottleneck
  - ✓ Slower Request interface impacts memory read/write throughput
  - ✓ Slower inbound Completion interface could result in data loss
- Key mechanisms
  - ✓ Ready/Valid-based protocol instead of request-ack protocol
  - ✓ Selective flow control for Non-Posted to avoid HOL blocking
  - ✓ Ability to straddle multiple TLPs per cycle

Valid-Ready Handshake Protocol

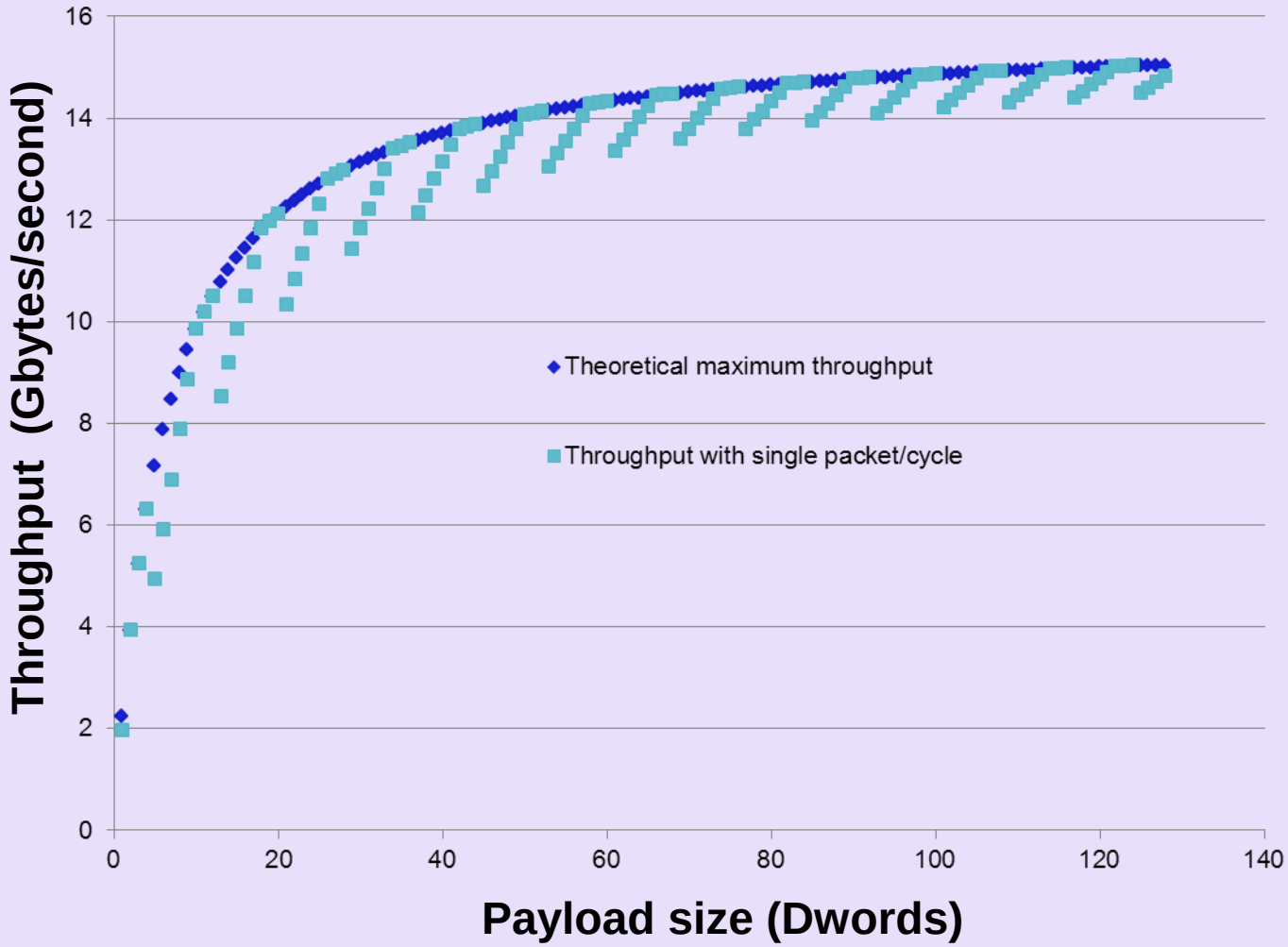


- Valid controlled by source of data, ready by destination
  - ✓ Data transferred when both valid and ready are high
- Potential impact on timing closure
  - ✓ High fan-out
  - ✓ Long ready chains in pipeline

Inbound Request Interface

- 256-bit interface with single packet per cycle can lead to some performance loss.
  - ✓ Bandwidth loss depends on packet size and alignment
- Assume 128-bit header, followed by payload
- Worst case is for payload size = 5 Dwords
- Also needs to support selective backpressure or flow control for Non-Posted requests

Performance Impact of Single-Packet Interface



Inbound Completion Interface

- 256-bit interface with single packet per cycle can lead to FIFO overflow
  - ✓ EndPoints required to advertise infinite Completion credit
- Straddle capability avoids loss of bandwidth
- Assume 96-bit header, followed by payload
- No bandwidth loss if packets are allowed to start at Dword positions 0, 3, 4, 5.

Outbound Request Interface

- Valid/ready handshake protocol does not allow application to re-schedule requests
  - ✓ Application must check available credit before presenting request to avoid HOL blocking
  - ✓ Also must check tag availability for Non-Posted requests
- Straddle capability can improve performance as in the inbound case.
  - ✓ Many options: Two Posted requests, Two NP requests, one of each kind, etc.
  - ✓ Adds significant complexity

Conclusions

- PCIe 3.0 x16 designs are not incremental
  - ✓ Using 256-bit data paths efficiently requires processing multiple packets per clock cycle
  - ✓ Architectural challenges in many parts of the design
  - ✓ Requires careful analysis of complexity Vs. performance tradeoffs
- Fundamental problem: Clock speed in ASIC flows is not scaling as fast as PCIe generations
  - ✓ Plenty of gates to build wide data paths and complex logic, but how to use them?
- Preview of challenges to come with future generations of PCIe

Thank you for attending the
PCI-SIG Developers Conference 2011.

For more information please go to
www.pcisig.com