PCI Express® Basics & Background

Richard Solomon

Synopsys

Acknowledgements

Thanks are due to Ravi Budruk, Mindshare, Inc. for much of the material on PCI Express® Basics

PCI Express Background

Revolutionary AND Evolutionary

○ PCI™ (1992/1993)

- Revolutionary
  - Plug and Play jumperless configuration (BARs)
  - Unprecedented bandwidth
    - 32-bit / 33MHz – 133MB/sec
    - 64-bit / 66MHz – 533MB/sec
  - Designed from day 1 for bus-mastering adapters

- Evolutionary
  - System BIOS maps devices then operating systems boot and run without further knowledge of PCI
  - PCI-aware O/S could gain improved functionality
  - PCI 2.1 (1995) doubled bandwidth with 66MHz mode

Revolutionary AND Evolutionary

○ **PCI-X™ (1999)**

- Revolutionary
  - Unprecedented bandwidth
    - Up to 1066MB/sec with 64-bit / 133MHz
  - Registered bus protocol
    - Eased electrical timing requirements
  - Brought split transactions into PCI “world”

- Evolutionary
  - PCI compatible at hardware *AND* software levels
  - PCI-X 2.0 (2003) doubled bandwidth
    - 2133MB/sec at PCI-X 266 and 4266MB/sec at PCI-X 533

Revolutionary AND Evolutionary

- **PCI Express – aka PCIe® (2002)**
  - Revolutionary
    - Unprecedented bandwidth
      - x1: up to 4GB/sec in *EACH* direction (PCIe 5.0)
      - x16: up to 64GB/sec in *EACH* direction (PCIe 5.0)
    - “Relaxed” electricals due to serial bus architecture
      - Point-to-point, low voltage, dual simplex with embedded clocking
  - Evolutionary
    - PCI compatible at software level
      - Configuration space, Power Management, etc.
      - Of course, PCIe-aware O/S can get more functionality
    - Transaction layer familiar to PCI/PCI-X designers
    - System topology matches PCI/PCI-X
    - Doubling of bandwidth each generation (from 250MB/s/lane):
      - PCIe 2.0 (2006) 500MB/s/lane
      - PCIe 3.0 (2010) ~1GB/s/lane
      - PCIe 4.0 (2017) ~2GB/s/lane
      - PCIe 5.0 (2019) ~4GB/s/lane

PCI Concepts

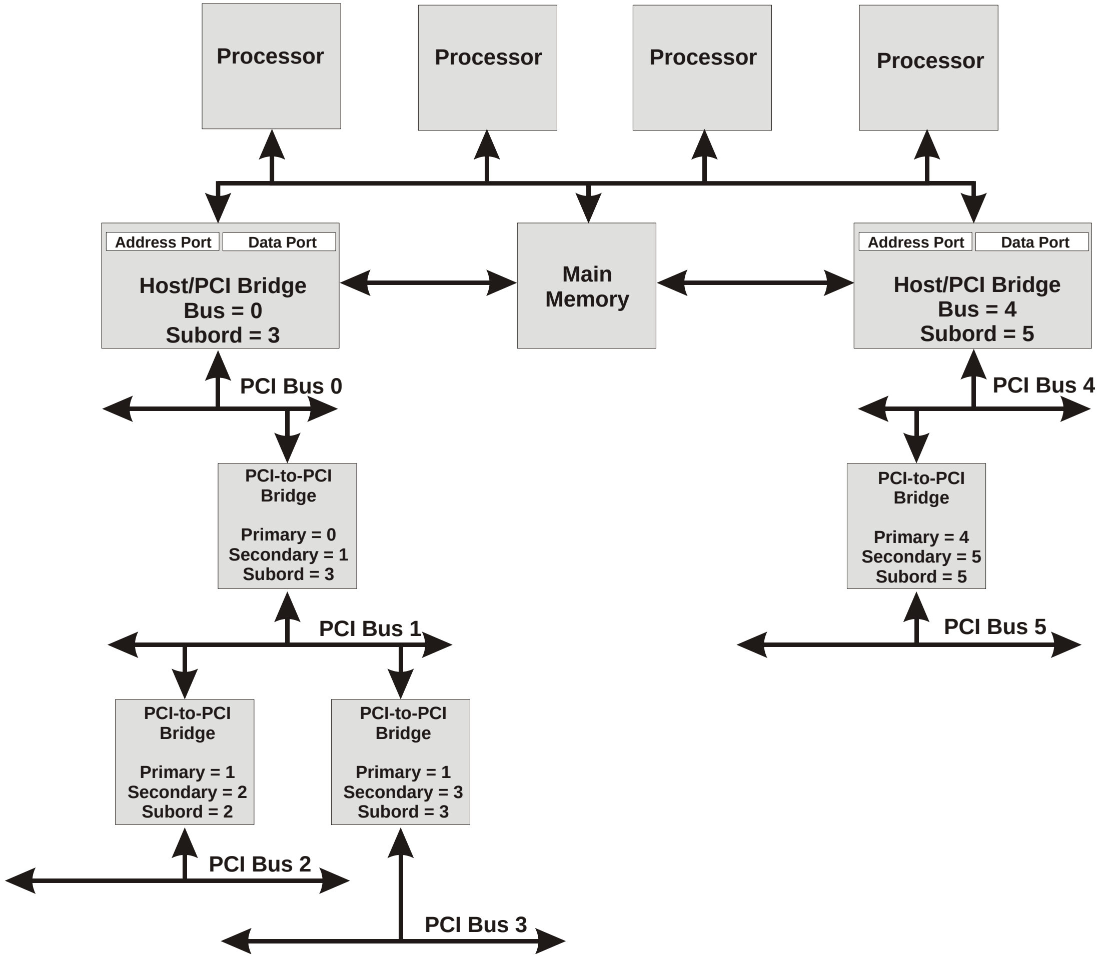
Address Spaces – Memory & I/O

- **Memory space mapped cleanly to CPU semantics**
  - 32-bits of address space initially
  - 64-bits introduced via Dual-Address Cycles (DAC)
    - Extra clock of address time on PCI/PCI-X
    - 4 DWORD header in PCI Express
  - Burstable
- **I/O space mapped cleanly to CPU semantics**
  - 32-bits of address space
    - Actually much larger than CPUs of the time
  - Non-burstable
    - Most PCI implementations didn’t support
    - PCI-X codified
    - Carries forward to PCI Express

Address Spaces – Configuration

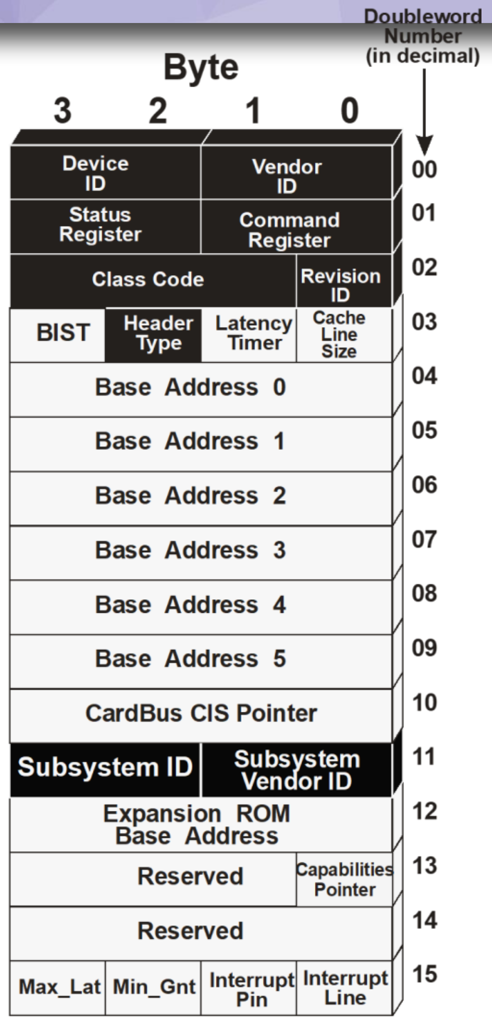
- **Configuration space???**
  - Allows control of devices' address decodes without conflict
  - No conceptual mapping to CPU address space
    - Memory-based access mechanisms in PCI-X and PCIe
  - Bus / Device / Function (aka BDF) form hierarchy-based address (PCIe 3.0 calls this "Routing ID")
    - "Functions" allow multiple, logically independent agents in one physical device
      - E.g. combination SCSI + Ethernet device
      - 256 bytes or 4K bytes of configuration space per device
    - PCI/PCI-X bridges form hierarchy
    - PCIe switches form hierarchy
      - Look like PCI-PCI bridges to software
  - "Type 0" and "Type 1" configuration cycles
    - Type 0: to same bus segment
    - Type 1: to another bus segment

Configuration Space (cont'd)



Configuration Space

- **Device Identification**
  - VendorID: PCI-SIG® assigned
  - DeviceID: Vendor self-assigned
  - Subsystem VendorID: PCI-SIG
  - Subsystem DeviceID: Vendor
- **Address Decode controls**
  - Software reads/writes BARs to determine required size and maps appropriately
  - Memory, I/O, and bus-master enables
- **Other bus-oriented controls**

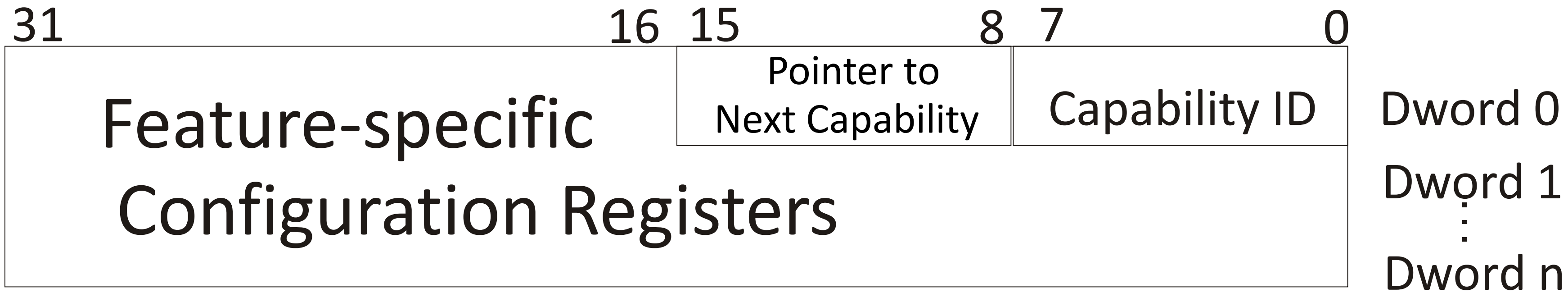


Byte				Doubleword Number (in decimal)
3	2	1	0	
Device ID		Vendor ID		00
Status Register		Command Register		01
Class Code			Revision ID	02
BIST	Header Type	Latency Timer	Cache Line Size	03
Base Address 0				04
Base Address 1				05
Base Address 2				06
Base Address 3				07
Base Address 4				08
Base Address 5				09
CardBus CIS Pointer				10
Subsystem ID		Subsystem Vendor ID		11
Expansion ROM Base Address				12
Reserved			Capabilities Pointer	13
Reserved				14
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	15

Configuration Space – Capabilities List

○ Linked list

- Follow the list! Cannot assume fixed location of any given feature in any given device
- Features defined in their related specs:
  - PCI-X
  - PCIe
  - PCI Power Management
  - Etc.



31 16 15 8 7 0

Feature-specific Configuration Registers

Pointer to Next Capability

Capability ID

Dword 0

Dword 1

⋮

Dword n

Configuration Space – Extended Capabilities List

Linked list – new with PCI Express

- Follow the list! Cannot assume fixed location of any given feature in any given device
- First entry in list is *always* at 100h
- Features defined in PCI Express and related (e.g. MR-IOV, SR-IOV) specifications
- Consolidated in ***PCI Code and ID Assignment Spec***

31 20	19 16	15 8 7 0	
Pointer to Next Capability	Version	Capability ID	Dword 0
Feature-specific Configuration Registers			Dword 1 ⋮ Dword n

Interrupts

- **PCI introduced INTA#, INTB#, INTC#, INTD# - collectively referred to as INTx**
  - Level sensitive
  - Decoupled device from CPU interrupt
  - System controlled INTx to CPU interrupt mapping
  - Configuration registers
    - report A/B/C/D
    - programmed with CPU interrupt number
- **PCI Express mimics this via “virtual wire” messages**
  - Assert_INTx and Deassert_INTx

What are MSI and MSI-X?

- **Memory Write replaces previous interrupt semantics**
  - PCI and PCI-X devices stop asserting INTA/B/C/D and PCI Express devices stop sending Assert_INTx messages once MSI or MSI-X mode is enabled
  - MSI uses one address with a variable data value indicating which “vector” is asserting
  - MSI-X uses a table of independent address and data pairs for each “vector”
- **NOTE: *Boot devices* and any device intended for a non-MSI operating system generally must still support the appropriate INTx signaling!**

Split Transactions – Background

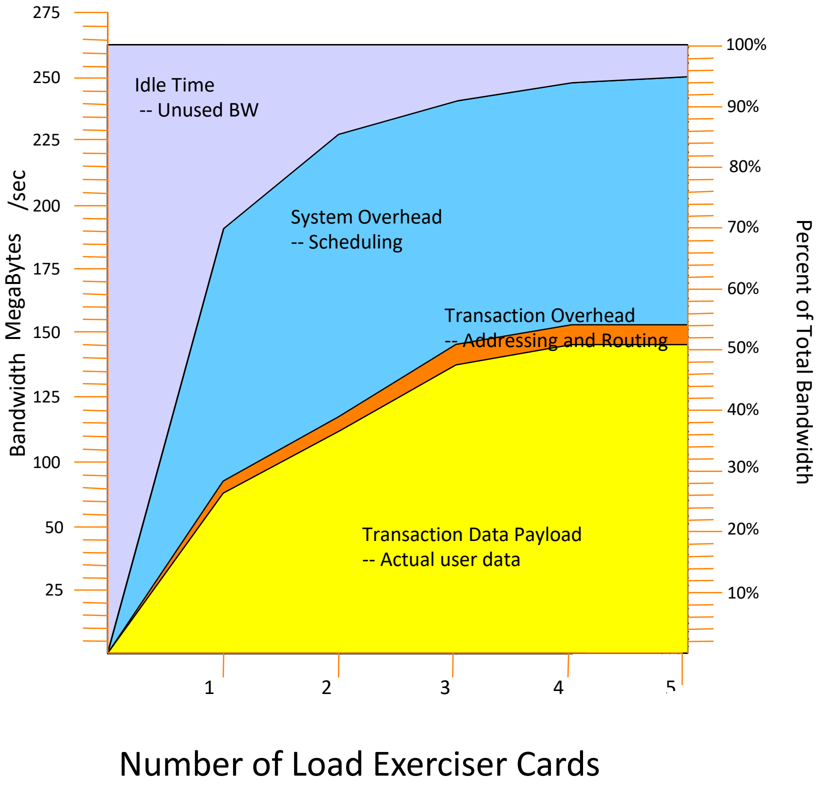
- **PCI commands contained no length**
  - Bus allowed disconnects and retries
  - Difficult data management for target device
    - Writes overflow buffers
    - Reads require pre-fetch
      - How much to pre-fetch? When to discard? Prevent stale data?
- **PCI commands contained no initiator information**
  - No way for target device to begin communication with the initiator
  - Peer-to-peer requires knowledge of system-assigned addresses

Split Transactions

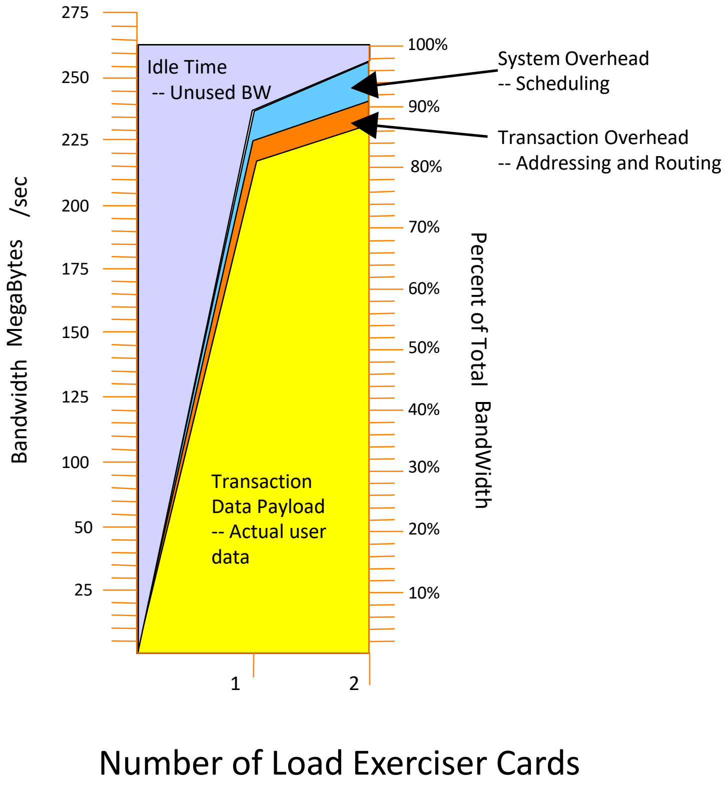
- **PCI-X commands added length and Routing ID of initiator**
  - Writes: allow target device to allocate buffers
  - Reads: Pre-fetch now deterministic
- **PCI-X retains “retry” & “disconnect”, adds “split”**
- **Telephone analogy**
  - Retry: “I’m busy go away”
    - Delayed transactions are complicated
  - Split: “I’ll call you back”
    - Simple
    - More efficient

Benefits of Split Transactions

Bandwidth Usage with Conventional PCI Protocols



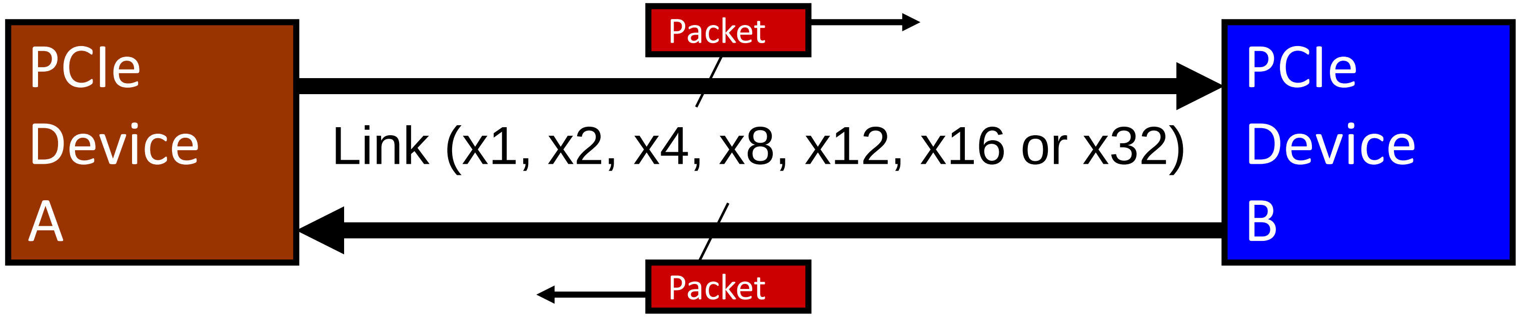
Bandwidth Usage with PCI-X Enhancements



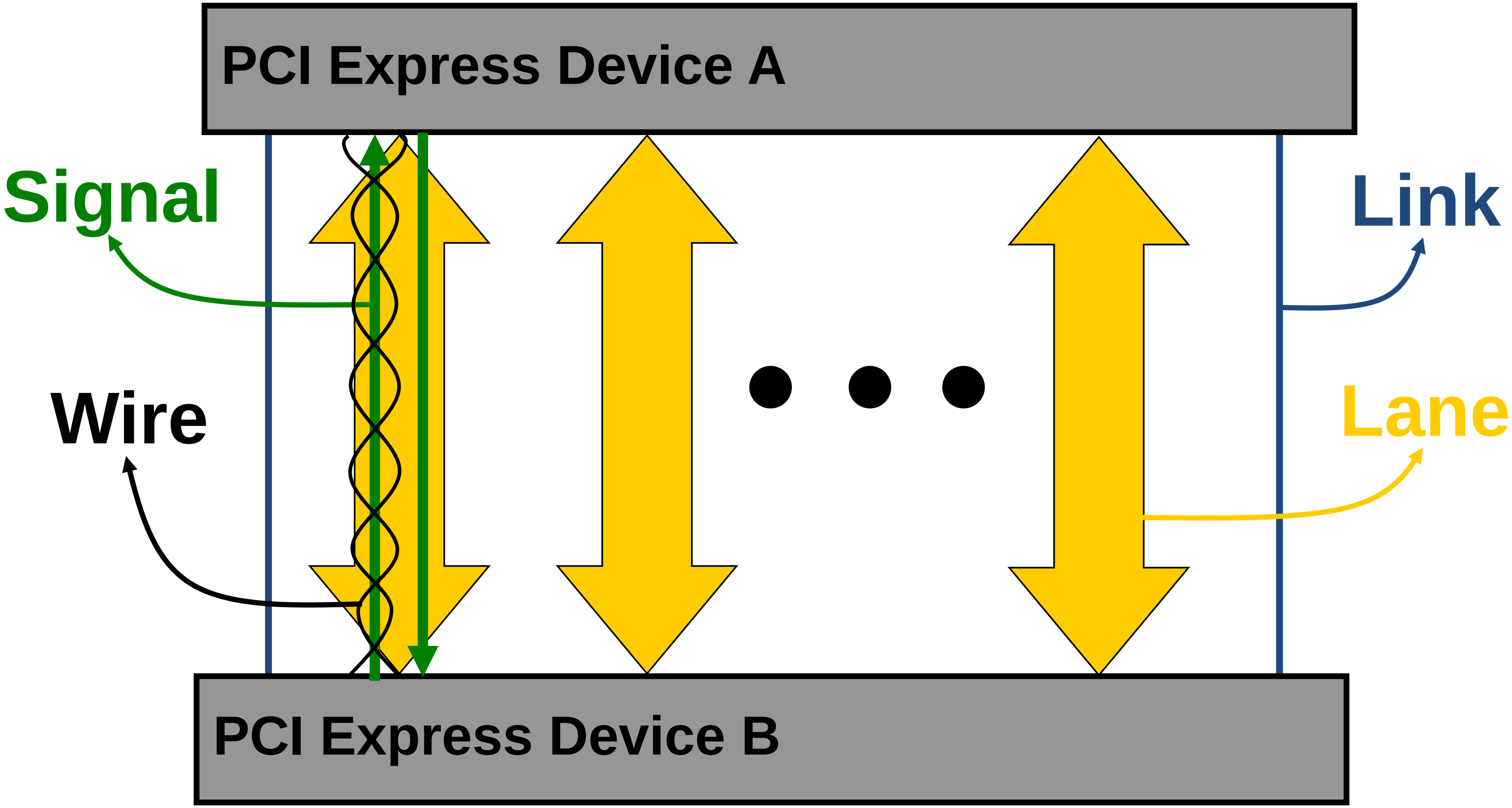
PCI Express Basics

PCI Express Features

- **Dual Simplex point-to-point serial connection**
  - Independent transmit and receive sides
- **Scalable Link Widths**
  - x1, x2, x4, x8, *x12*, x16, *x32*
- **Scalable Link Speeds**
  - 2.5, 5.0, 8.0, 16.0 GT/s, 32GT/s
- **Packet based transaction protocol**

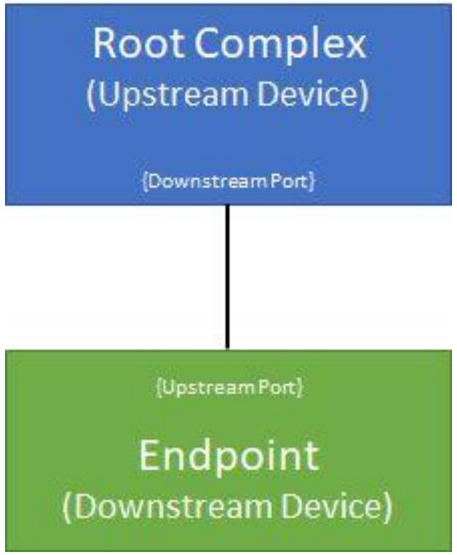


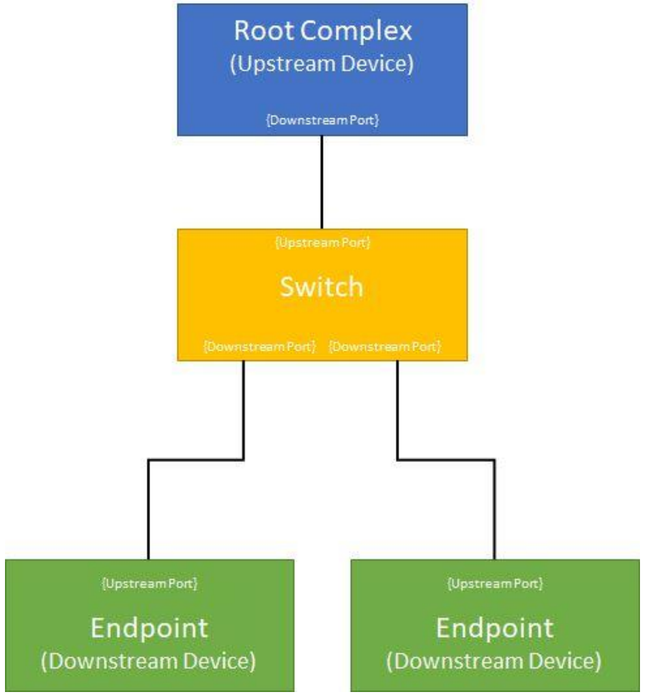
PCI Express Terminology



Upstream/Downstream

- **Relative to root – up is towards, down is away**
- **Note “streamness” of devices vs their ports**
  - The direction a gremlin standing on the device looks…





PCI Express Throughput

Bandwidth (GB/s)	Link Width				
	x1	x2	x4	x8	x16
PCIe 1.x “2.5 GT/s”	0.25	0.5	1	2	4
PCIe 2.x “5 GT/s”	0.5	1	2	4	8
PCIe 3.x “8 GT/s”	~1	~2	~4	~8	~16
PCIe 4.0 “16GT/s”	~2	~4	~8	~16	~32
PCIe 5.0 “32GT/s”	~4	~8	~16	~32	~64

Derivation of these numbers:

- ○ **20% overhead due to 8b/10b encoding in 1.x and 2.x**
- ○ **Note: ~1.5% overhead due to 128/130 encoding not reflected above in 3.x and beyond**

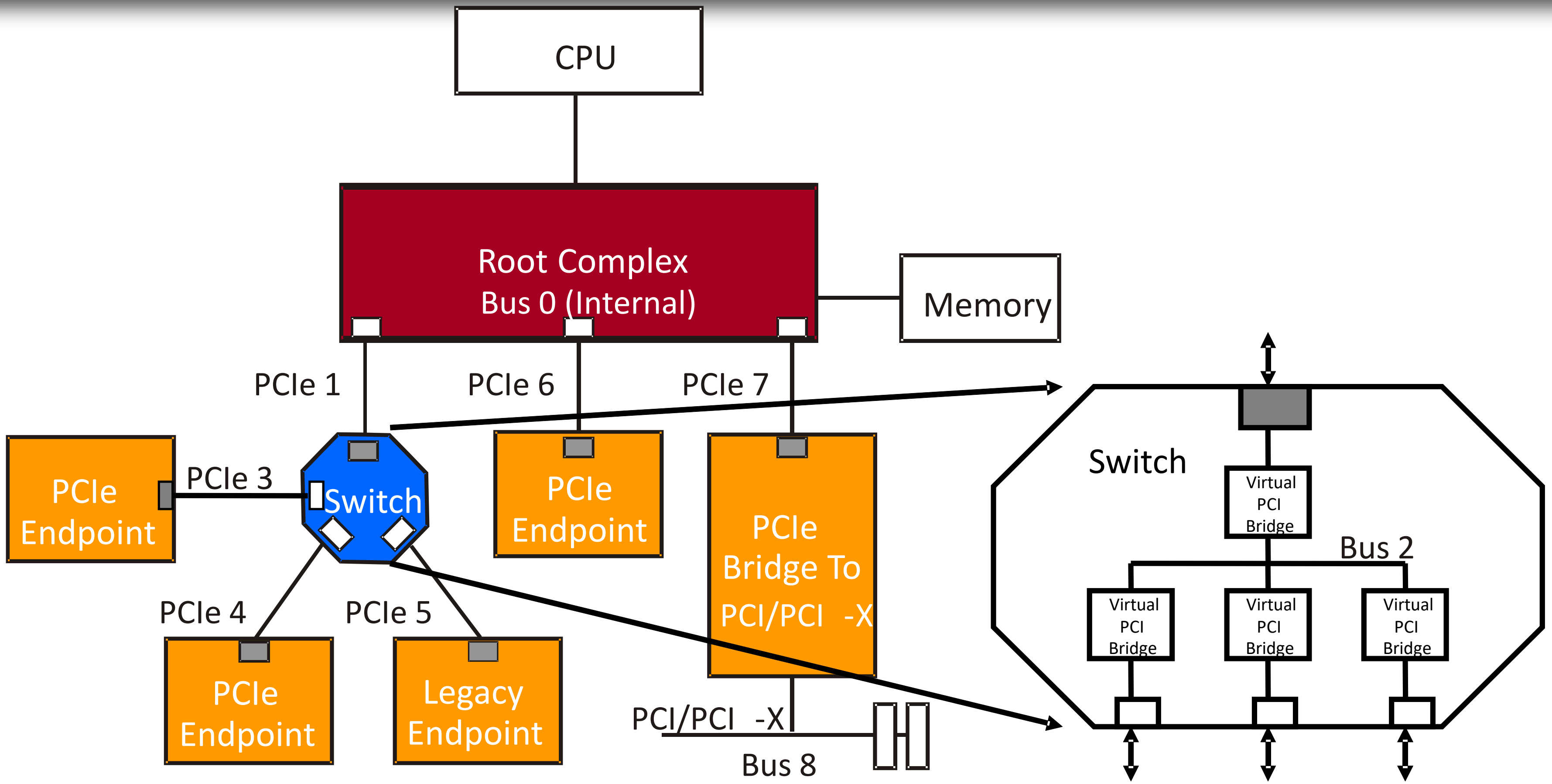
Additional Features

- **Data Integrity and Error Handling**
  - Link-level “LCRC”
  - Link-level “ACK/NAK”
  - End-to-end “ECRC”
- **Credit-based Flow Control**
  - No retry as in PCI
- **MSI/MSI-X style interrupt handling**
  - Also supports legacy PCI interrupt handling in-band
- **Advanced power management**
  - Active State PM
  - PCI compatible PM

Additional Features

- **Evolutionary PCI-compatible software model**
  - PCI configuration and enumeration software can be used to enumerate PCI Express hardware
  - PCI Express system will boot “PCI” OS
  - PCI Express supports “PCI” device drivers
  - New additional configuration address space requires OS and driver update
    - Advanced Error Reporting (AER)
    - PCI Express Link Controls

PCI Express Topology



Legend

□ PCI Express Device Downstream Port

▨ PCI Express Device Upstream Port

Transaction Types, Address Spaces

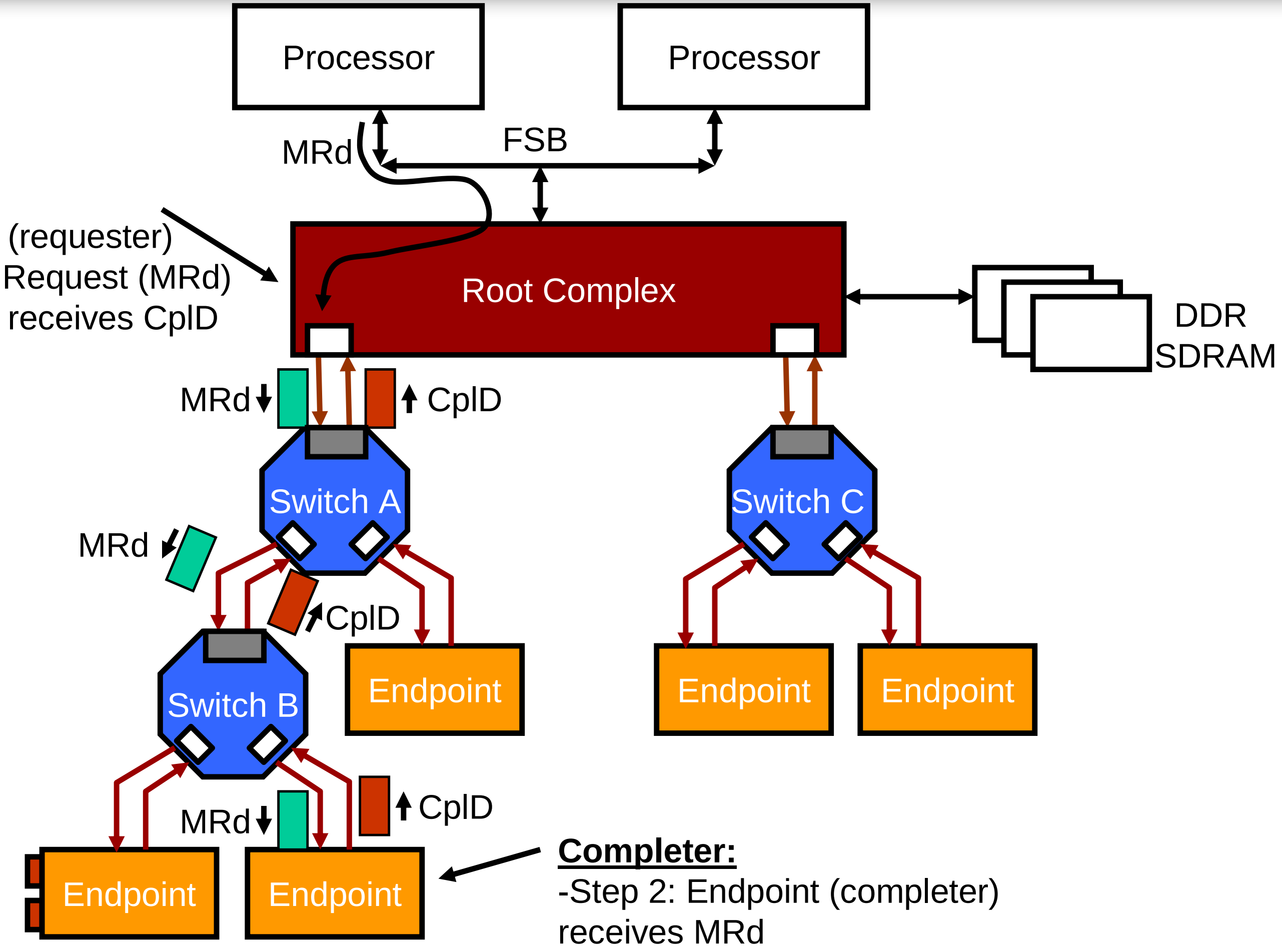
- **Request are translated to one of four transaction types by the Transaction Layer:**

1. **Memory Read or Memory Write**. Used to transfer data from or to a memory mapped location.
   - The protocol also supports a *locked memory read* transaction variant
2. **I/O Read or I/O Write.** Used to transfer data from or to an I/O location.
   - These transactions are restricted to supporting legacy endpoint devices
3. **Configuration Read or Configuration Write**. Used to discover device capabilities, program features, and check status in the 4KB PCI Express configuration space.
4. **Messages.** Handled like posted writes. Used for event signaling and general purpose messaging.

Three Methods For Packet Routing

- **Each request or completion header is tagged as to its *type*, and each of the packet types is routed based on one of three schemes:**
  - Address Routing
  - ID Routing
  - Implicit Routing
- **Memory and IO requests use address routing**
- **Completions and Configuration cycles use ID routing**
- **Message requests have selectable routing based on a 3-bit code in the message routing sub-field of the header type field**

Programmed I/O Transaction

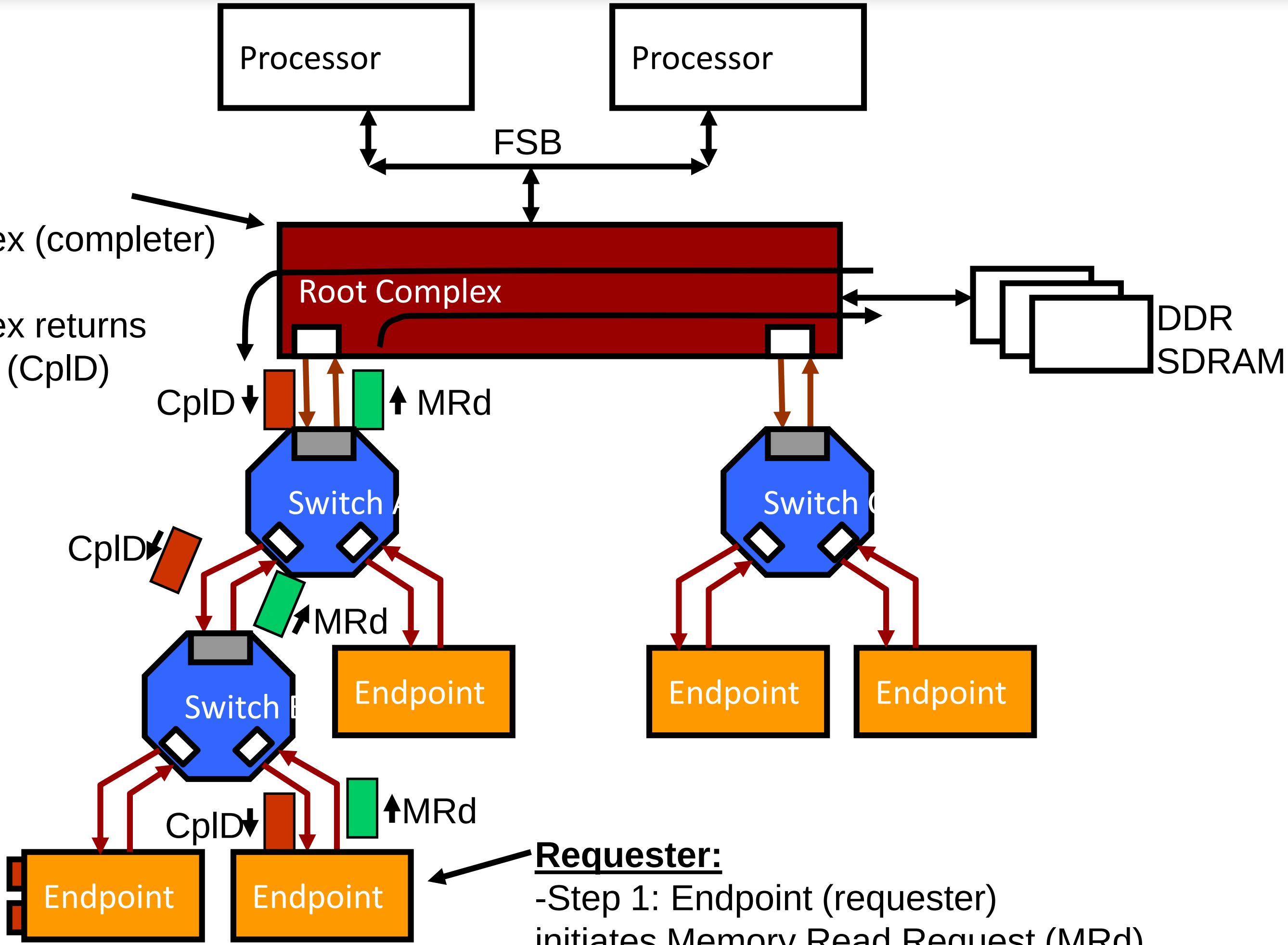


Requester:
-Step 1: Root Complex (requester) initiates Memory Read Request (MRd)
-Step 4: Root Complex receives CplD

Completer:
-Step 2: Endpoint (completer) receives MRd
-Step 3: Endpoint returns Completion with data (CplD)

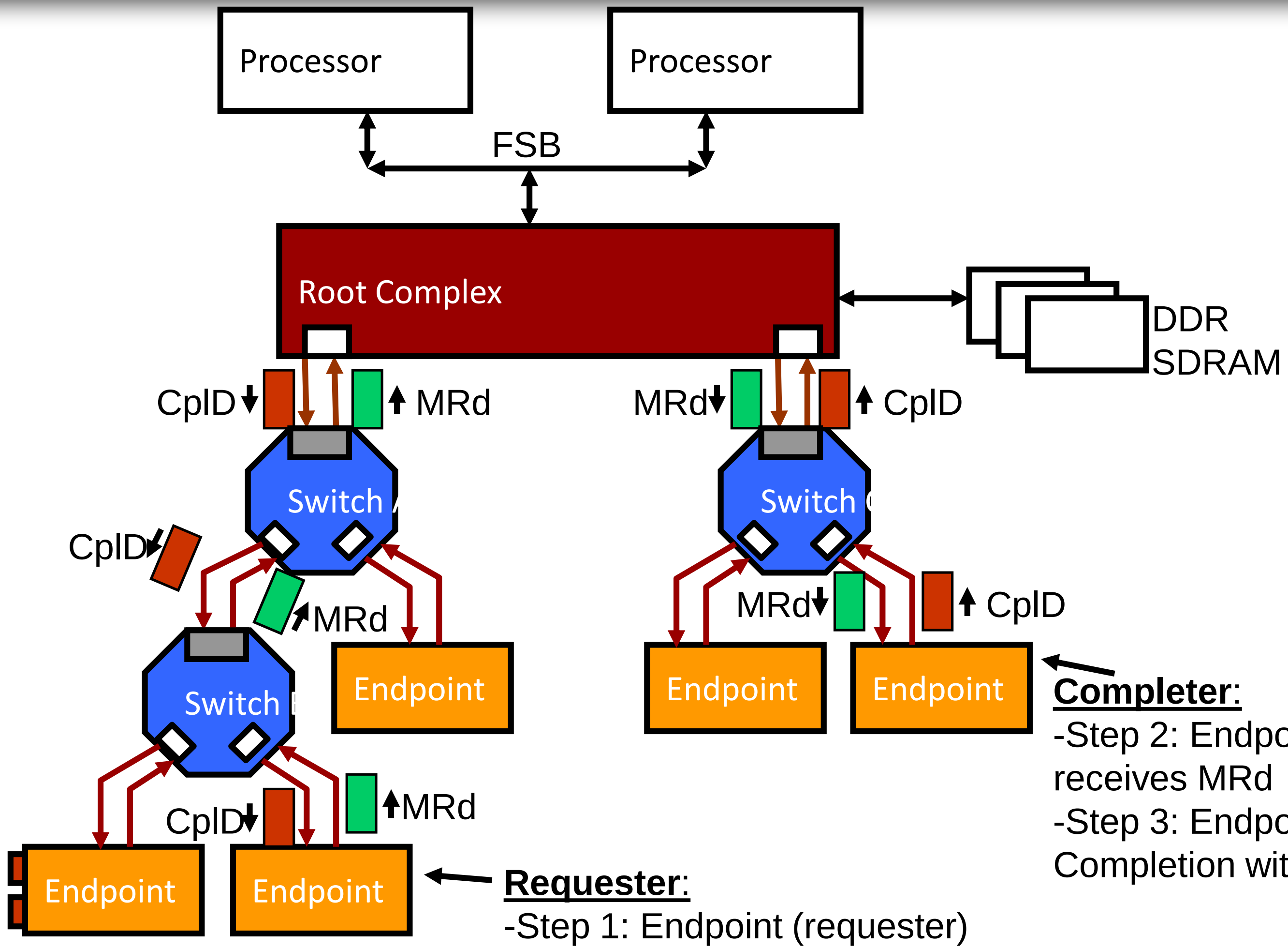
DMA Transaction

Completer:
-Step 2: Root Complex (completer) receives MRd
-Step 3: Root Complex returns Completion with data (CplD)



Requester:
-Step 1: Endpoint (requester) initiates Memory Read Request (MRd)
-Step 4: Endpoint receives CplD

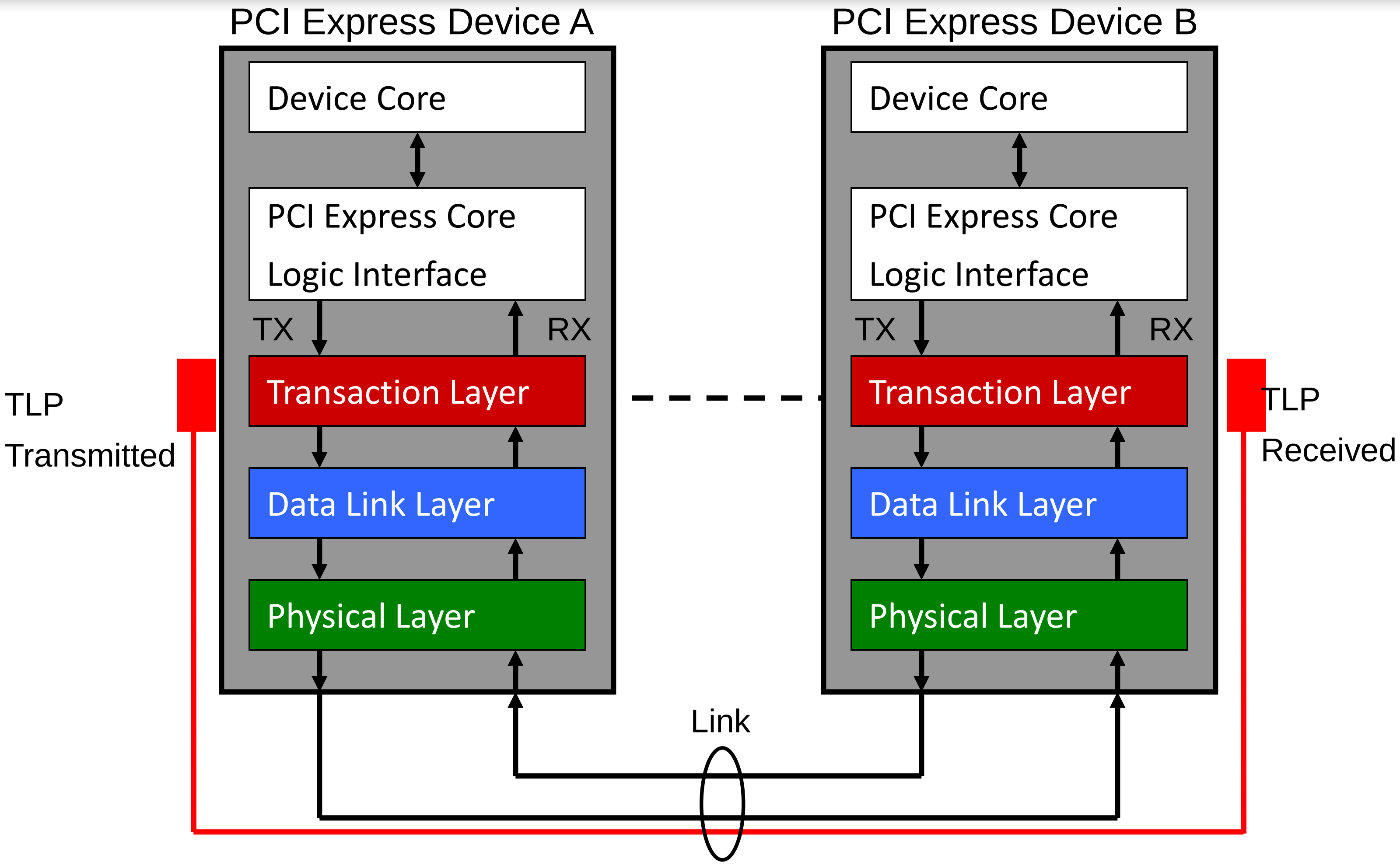
Peer-to-Peer Transaction



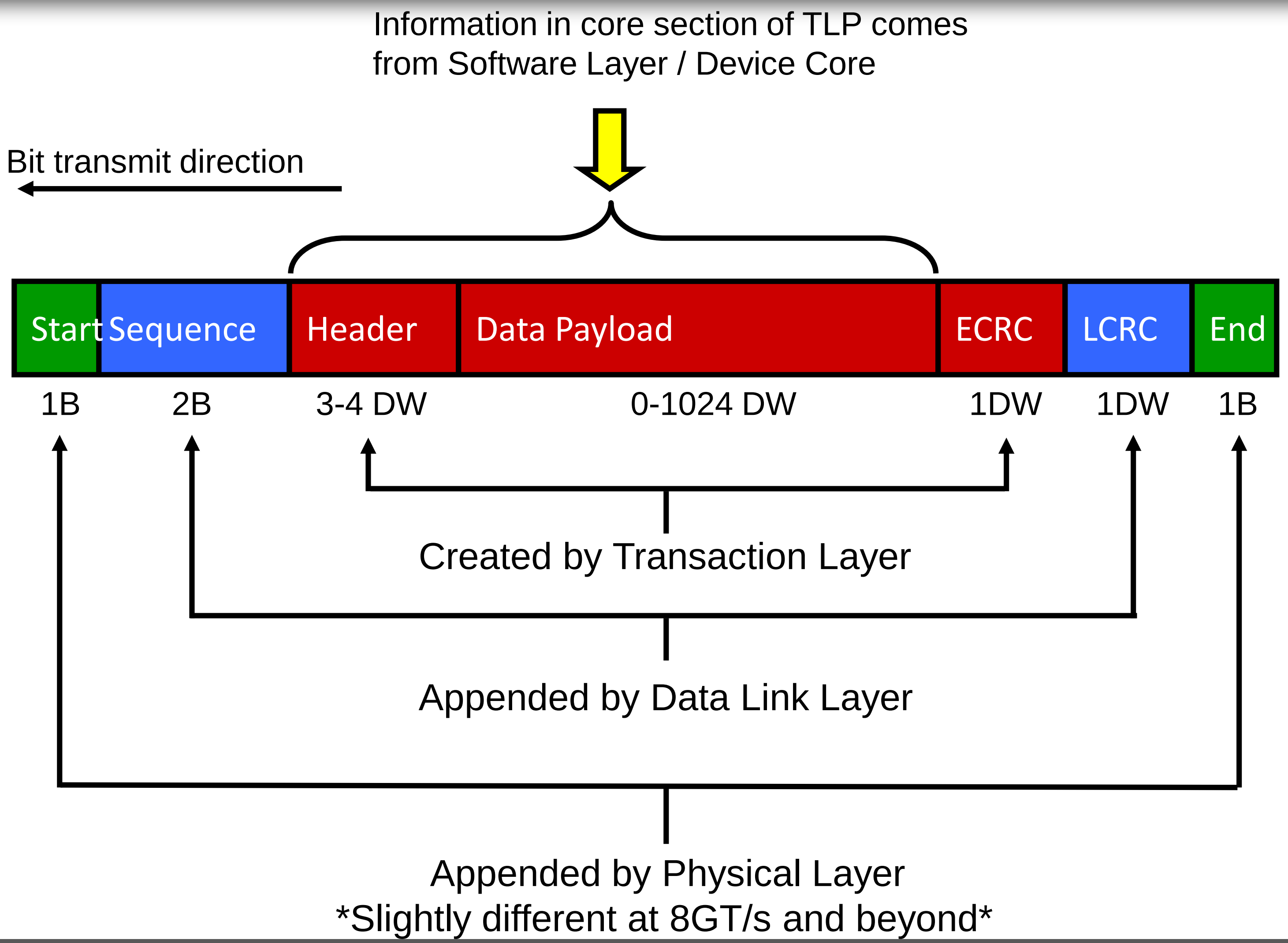
Completer:
-Step 2: Endpoint (completer) receives MRd
-Step 3: Endpoint returns Completion with data (CplD)

Requester:
-Step 1: Endpoint (requester) initiates Memory Read Request (MRd)
-Step 4: Endpoint receives CplD

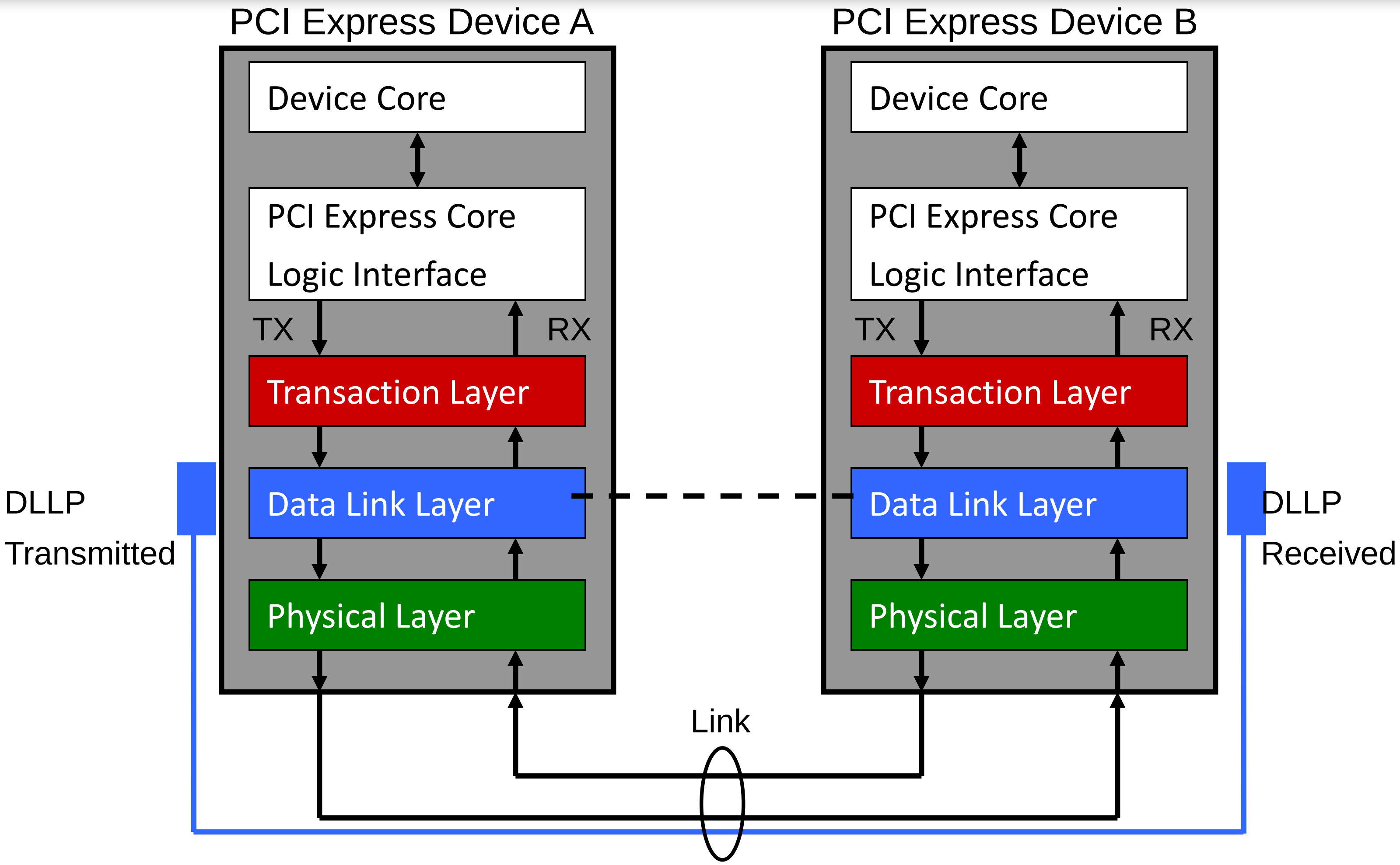
TLP Origin and Destination



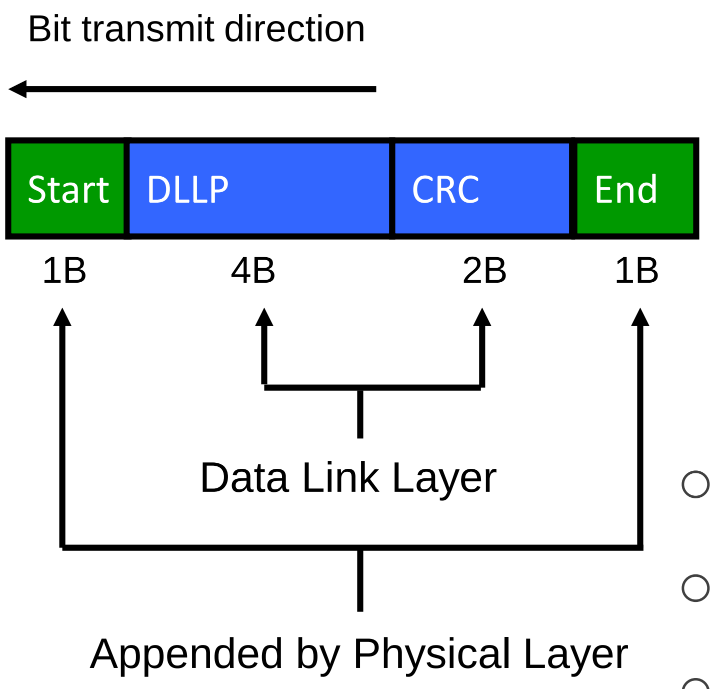
TLP Structure


Information in core section of TLP comes from Software Layer / Device Core
Bit transmit direction
Start
Sequence
Header
Data Payload
ECRC
LCRC
End
1B
2B
3-4 DW
0-1024 DW
1DW
1DW
1B
Created by Transaction Layer
Appended by Data Link Layer
Appended by Physical Layer
Slightly different at 8GT/s and beyond

DLLP Origin and Destination

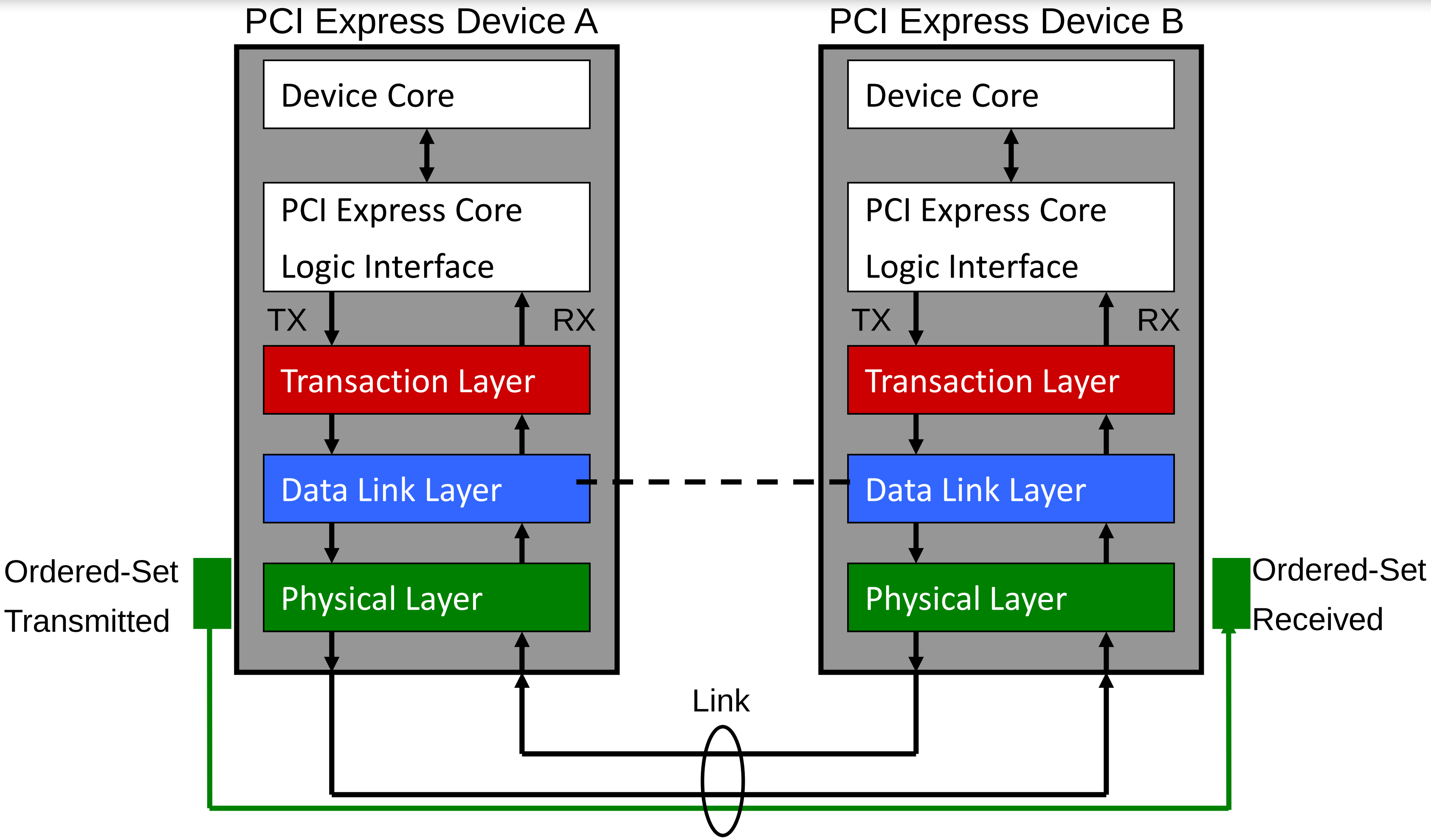


DLLP Structure

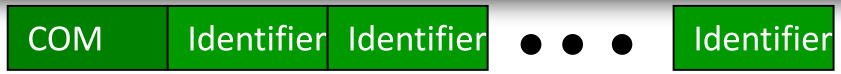


- ○ **ACK / NAK Packets**
- ○ **Flow Control Packets**
- ○ **Power Management Packets**
- ○ **Vendor Defined Packets**

Ordered-Set Origin and Destination



Ordered-Set Structure

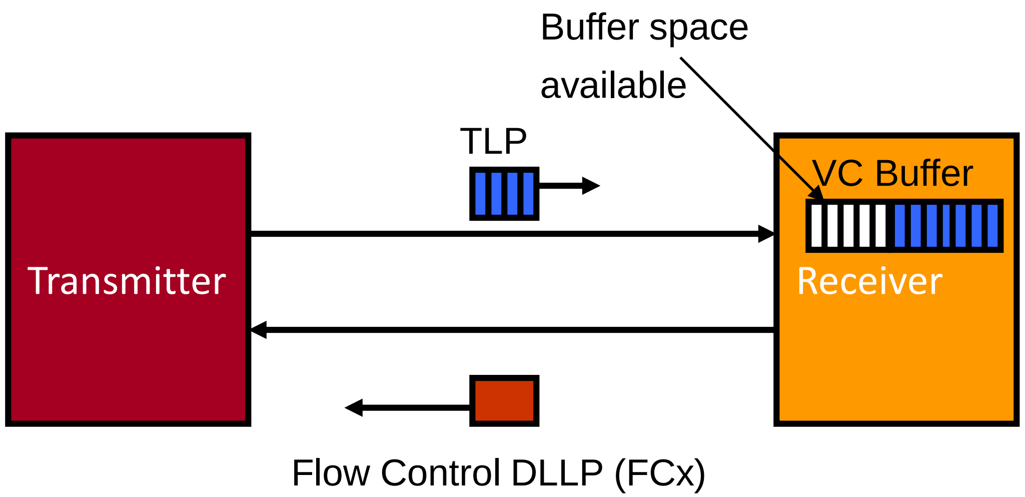


COM | Identifier | Identifier | • • • | Identifier

- ○ **Training Sequence One (TS1)**
  - 16 character set: 1 COM, 15 TS1 data characters
- ○ **Training Sequence Two (TS2)**
  - 16 character set: 1 COM, 15 TS2 data characters
- ○ **SKIP**
  - 4 character set: 1 COM followed by 3 SKP identifiers
- ○ **Fast Training Sequence (FTS)**
  - 4 characters: 1 COM followed by 3 FTS identifiers
- ○ **Electrical Idle (IDLE)**
  - 4 characters: 1 COM followed by 3 IDL identifiers
- ○ **Electrical Idle Exit (EIEOS) (new to 2.0 spec)**
  - 16 characters

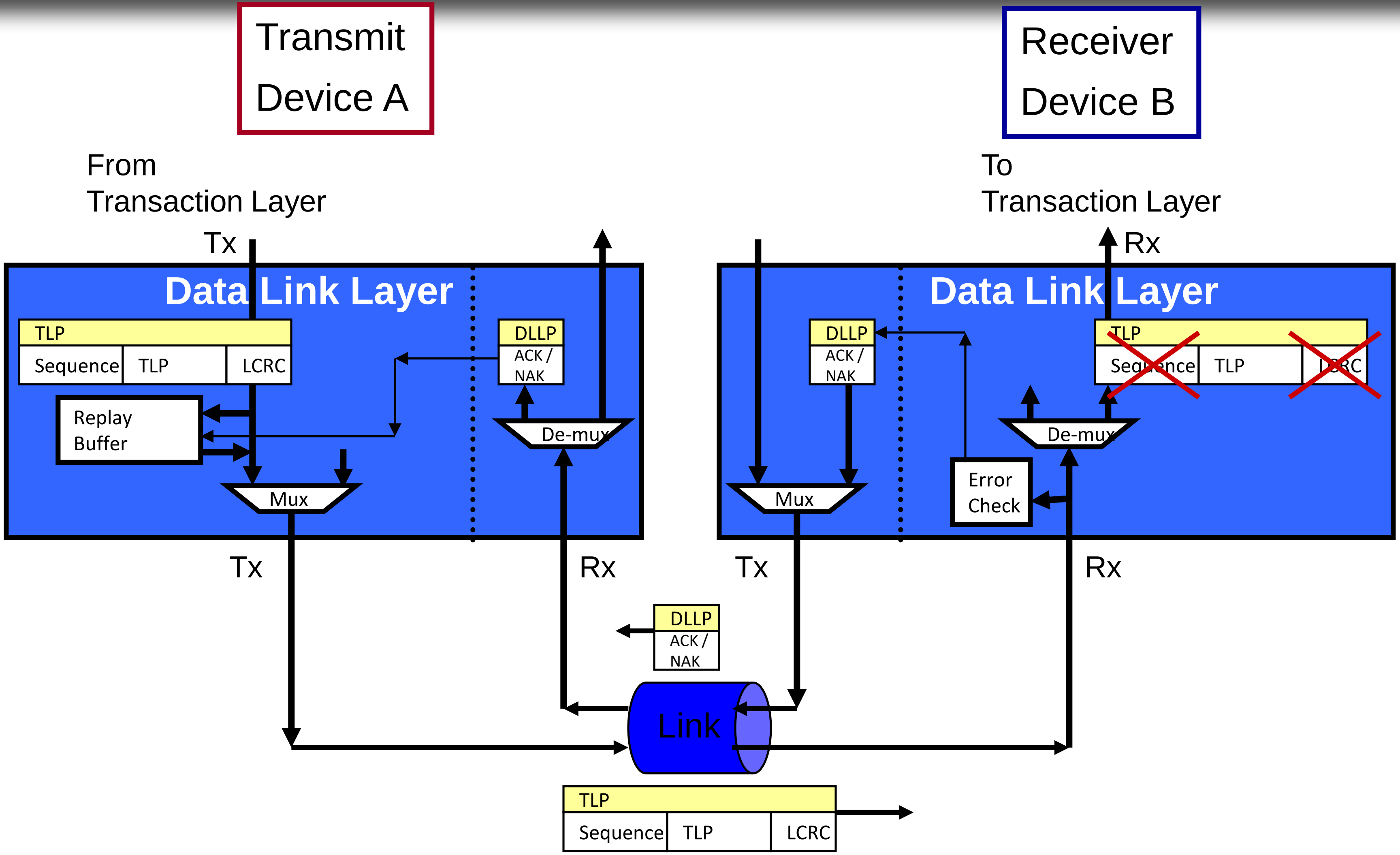
PCI Express Flow Control

Credit-based *flow control* is point-to-point based, not end-to-end

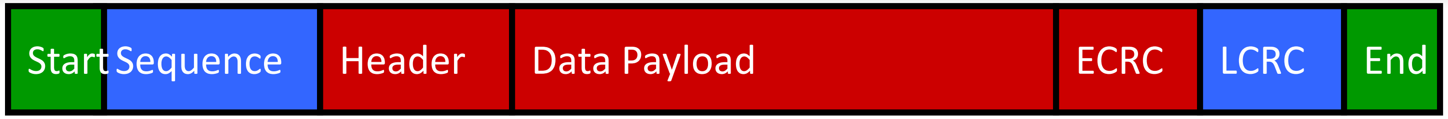


Receiver sends Flow Control Packets (FCP) which are a type of DLLP (Data Link Layer Packet) to provide the transmitter with credits so that it can transmit packets to the receiver

ACK/NAK Protocol Overview



ECRC Overview



Start	Sequence	Header	Data Payload	ECRC	LCRC	End

- **“End-to-End CRC” AKA the “I Don’t Trust Switches” feature**
  - Part of the TLP, therefore it’s covered by the LCRC
  - Covers “invariant” parts of the TLP (almost all bits)
  - Intended for the ultimate recipient, but allowed to be checked along the way
  - Switches pass value unmodified (Multi-cast complicates)
- **Loosely defined behavior when mismatched**
  - Log and report the error like any other (including AER)
  - Requests w/bad ECRC are “*strongly recommended*” to return Unsupported Request (UR) status
  - Even credit updates are only “*strongly recommended*” on Tx/Rx of bad ECRC packet

Thank you for attending the
PCI-SIG Developers Conference

Asia-Pacific Tour 2019.

For more information please go to
www.pcisig.com