Introduction to PCIe Standard

11.09.2020

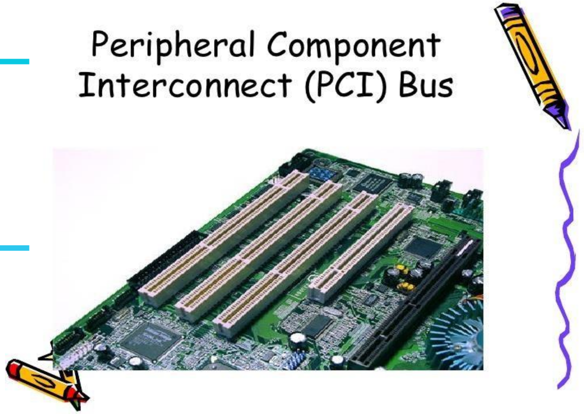
Introduction

Conventional PCI, often shortened to **PCI**, is a local computer bus for attaching hardware devices in a computer.

Attached devices can take either the form of an integrated circuit fitted onto the motherboard itself (called a planar device in the PCI specification) or an expansion card that fits into a slot.

Devices connected to the PCI bus appear to a bus master to be connected directly to its own bus and are assigned addresses in the processor's address space. It is a parallel bus, synchronous to a single bus clock.

Created by Intel (later **PCI-SIG**) to bring together best ideas from prior bus architectures i.e**. ISA (Industry Standard Architecture)** Bus and **VL (Video Electronics Standards Association (VESA) Local)** Bus..



Brief History

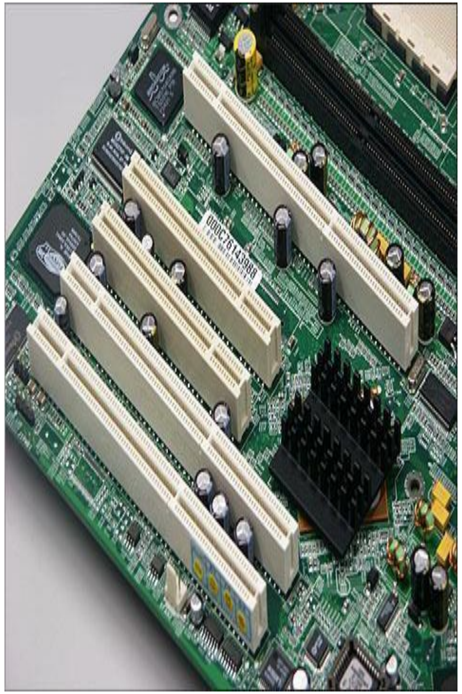
❖ PCI™ (1992/1993)

Revolutionary

- Plug and Play jumper less 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



Brief History

❖ 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

Brief History

PCI Express – aka PCIe® (2002)

Revolutionary

- Unprecedented bandwidth
  - x1: up to 1GB/sec in *EACH* direction
  - x16: up to 16GB/sec in *EACH* direction
- “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
- PCIe 2.0 (2006) doubled per-lane bandwidth: 250MB/s to 500MB/s
- PCIe 3.0 (2010) doubled again to 1GB/s/lane
- PCIe 4.0(2011) double again to 2GB/s/lane
- PCIe 5.0 (2017) double again to 4GB/s/lane

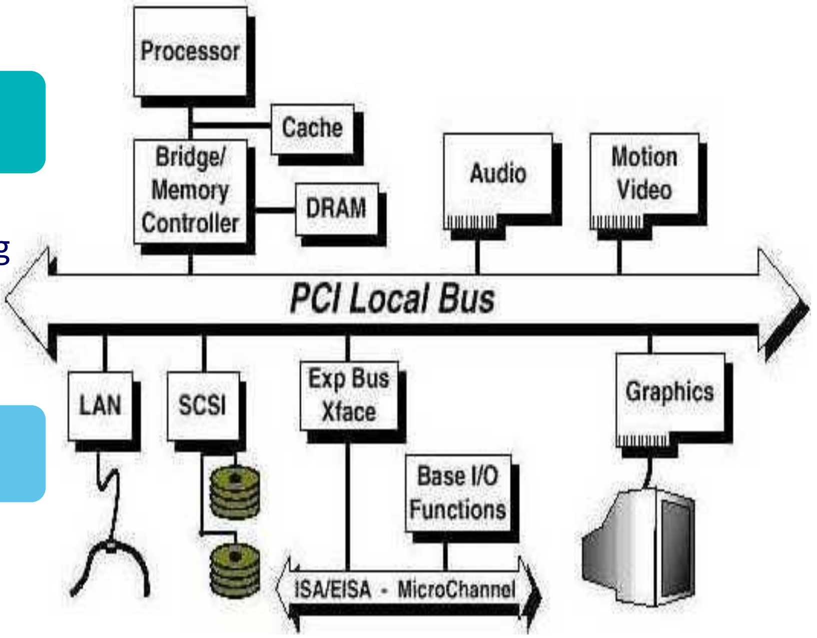
PCI Concepts

PCI Architecture

- The PCI Bus was originally 33MHz and then changed to 66MHz.
- PCI Bus became big with the release of Windows 95 with “Plug and Play” technology
- “Plug and Play” utilized the PCI bus concept.

PCI System Bus Performance

- Synchronous Bus Architecture
- 64 Bit Addressing
- Linear Burst Mode Data Transfer
- Large Bandwidth
- Full Bus Mastering



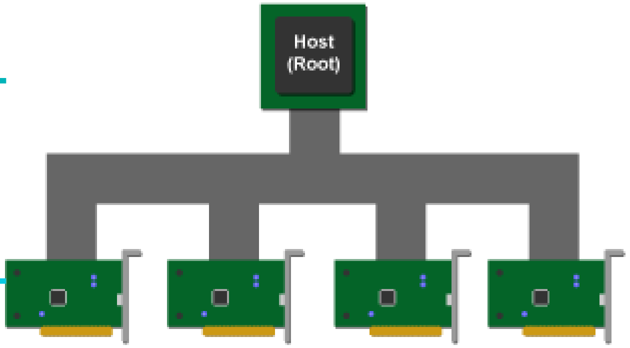
PCI Concepts

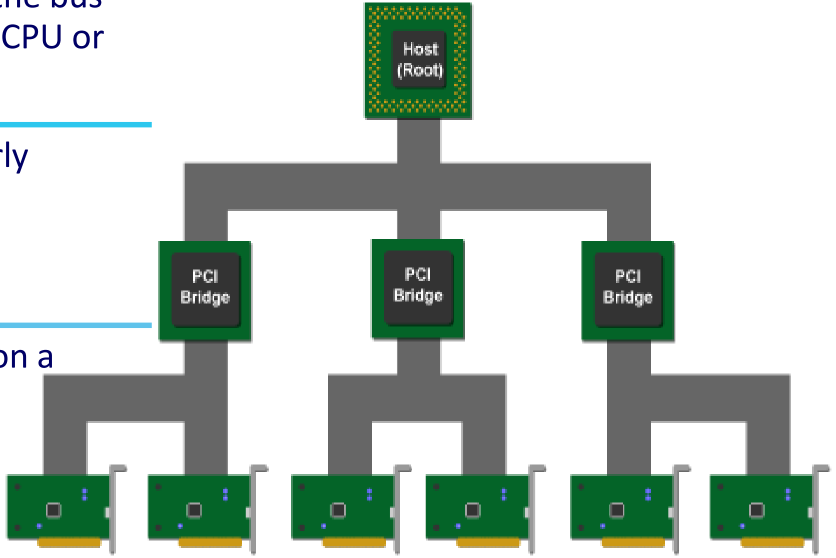
PCI uses a shared bus topology to allow for communication among the different devices on the bus;

Because all of the devices attached to the bus must share it among themselves. Once a device has control of the bus, it becomes the bus master, which means that it can use the PCI bus to talk to the CPU or memory via the chipset's Southbridge.

From the CPU's perspective, PCI devices are accessible via a fairly straightforward load-store mechanism.

In real life is that if you want to put more than five PCI devices on a system, then you must use PCI-to-PCI bridge chips configured



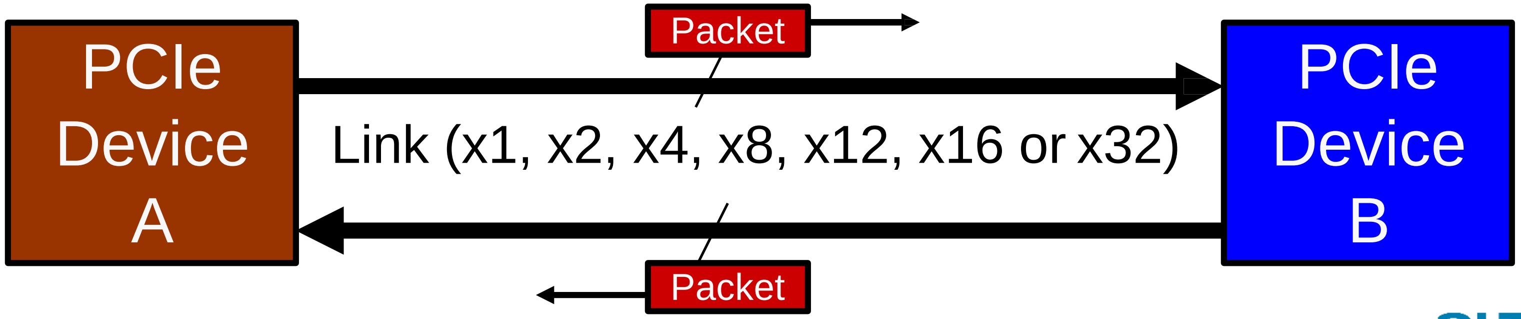


PCI Limitation

- Limited Bandwidth
  - PCI-X and Advanced Graphics Port (AGP) for higher frequency
  - Reduction of distance
- Bandwidth shared between all devices
- Limited host pin-count
- Lack of support for real time data transfer
- Stringent routing rules
- Lack of scaling with frequency and voltage
- Absence of power management
- PCI-X-- an enhancement of the 32-bit PCI Local Bus for a higher bandwidth demand.
  - a double-wide version of PCI, running at up to four times the clock speed

PCIe Concepts

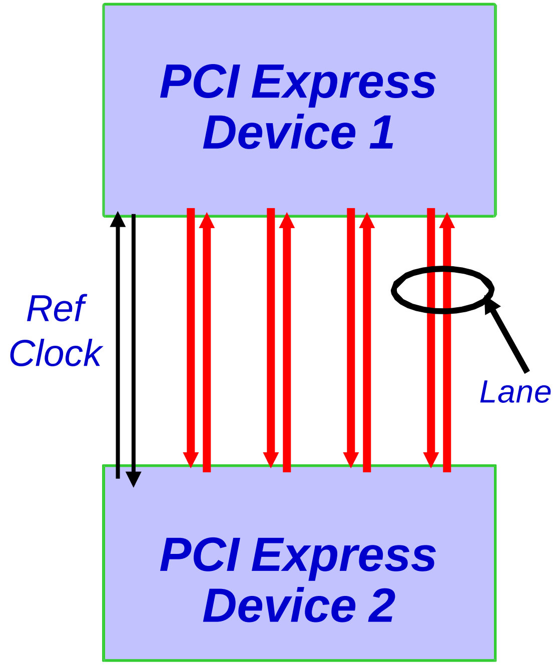
- Dual Simplex point-to-point serial connection
  - Independent transmit and receive sides
- Scalable Link Widths
  - x1, x2, x4, x8, *x12*, x16, *x32*
- Scalable Link Speeds
  - 2.5, 5.0 and 8.0GT/s (16GT/s coming in 4.0)
- Packet based transaction protocol



PCIe Basics

- **Serial, point-to-point, Low Voltage Differential Signaling**
- **2.5GHz full duplex lanes (2.5Gb/s)**
- **PCIe Gen 2 = 5Gb/s**
- **Scaleable links – x1, x4, x8, x16**
- **Packet based transaction protocol**
- **Software compatible but with higher speeds**

- **Built-in Quality of Service provisions**
  - Virtual Channels
  - Traffic Classes
- **Reliability, Availability and Serviceability**
  - End-to-End CRC (Cyclic redundant checking)
  - Poison Packet
  - Native Hot Plug support
- **Flow Control**
- **Advance error reporting**



x4 Link Example

PCIe Performance

Link Width	X1	X2	X4	X8	X12	X16	x32
Bandwidth in Gbits/s (Tx and Rx)	5	10	20	40	60	80	160
Throughput in GB/s (Tx and Rx)	.5	1	2	4	6	8	16
Throughput in GB/s (per direction)	🟨 .25	🟩 .5	🟥 1	2	3	4	8

Raw: Assuming 100% efficiency with no payload overhead.

🟨 ***= PCI 32/66***

🟩 ***= PCI or PCI-X 64/66***

🟥 ***= PCI-X 64/133***

PCIe Electrical Characteristics

- Data rates 2.5GT/s, 5GT/s and 8GT/s
- 10-12 bit error ratio
- AC coupled
- Link widths 1, 2, 4, 8, 16, 32 lanes
- Hot swap capable
- 2.5 and 5GT/s scrambled + 8b10b
- 8GT/s scrambled + 128/130
- Power management

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 3.0 “8 GT/s”	1	2	4	8	16
PCIe 4.0 “16GT/s”	*2*	*4*	*8*	*16*	*32*

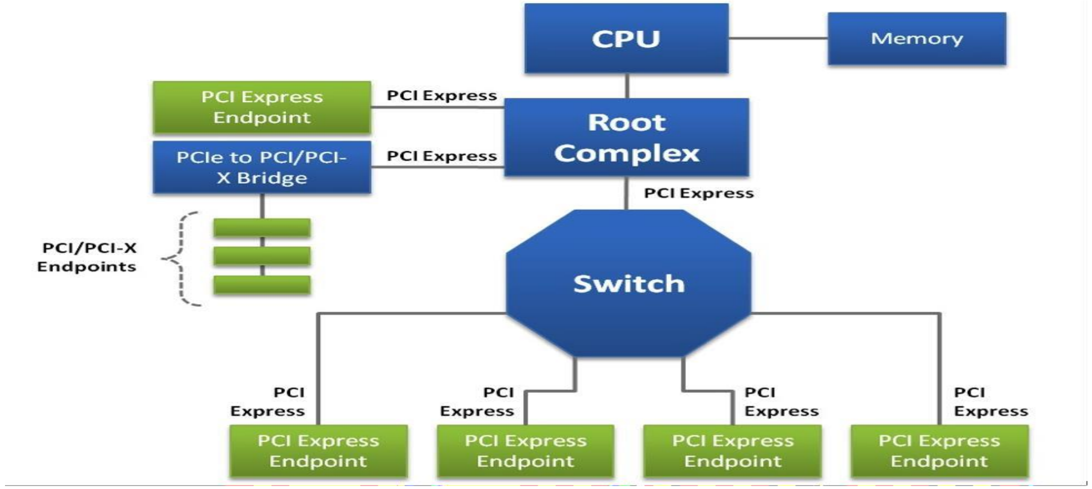
Derivation of these numbers:

- 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 4.0

PCIe transactions and Topology

PCIe Components:

- Root Complex
- Endpoints
- PCI Express-to-PCI(-X) Bridge
- Requester
- Completer
- Port
- Switch



Root Complex (RC)

- Root Complex (RC) – The interface between the CPU and the PCIe buses may contain several components (processor interface, DRAM interface, etc.)
- Logically aggregates PCIe hierarchy domains into one single PCIe hierarchy
- Single fabric instance referred to as a hierarchy composed of a RC , multiple Endpoints (I/O devices), a Switch, and a PCI Express to PCI/PCI-X Bridge, all interconnected via PCI Express Links

PCI Address Space

A PCI target can implement up to three different types of address spaces

I/O space

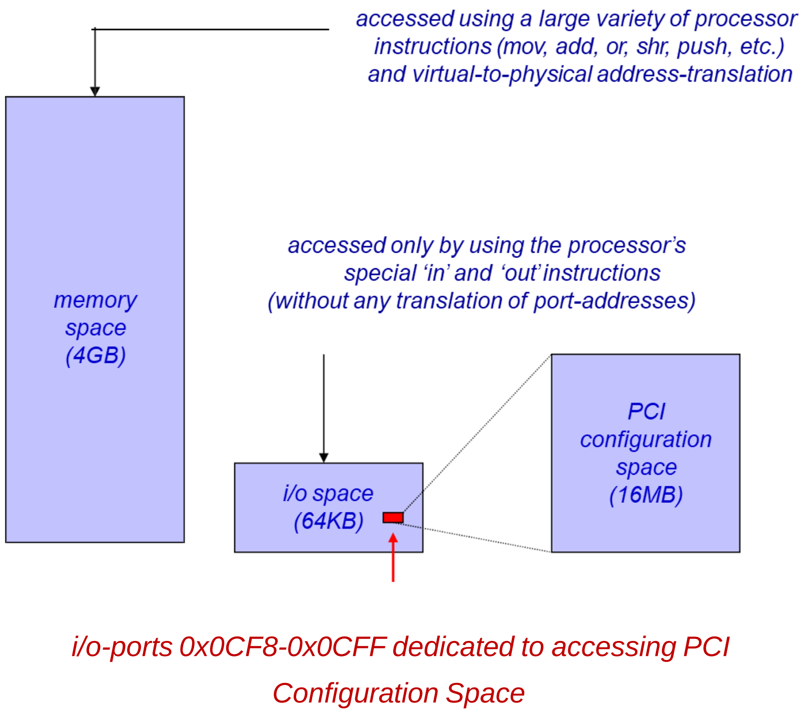
- ❖ Stores basic information about the device
- ❖ Allows the central resource or O/S to program a device with operational settings

Memory space

- ❖ Used mainly with PC peripherals and not much else

Configuration space

- Used for just about everything else


accessed using a large variety of processor instructions (mov, add, or, shr, push, etc.) and virtual-to-physical address-translation
memory space (4GB)
accessed only by using the processor's special 'in' and 'out' instructions (without any translation of port-addresses)
i/o space (64KB)
PCI configuration space (16MB)
i/o-ports 0x0CF8-0x0CFF dedicated to accessing PCI Configuration Space

PCI Configuration Address space

- Contains 256 bytes of basic device information,
  - addressable by 8-bit PCI bus, 5-bit device, and 3-bit function numbers for the device
    - 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
  - the first 64 bytes (00h – 3Fh) make up the standard configuration header, including PCI ID, i.e. vendor ID and device ID registers, to identify the device
  - the remaining 192 bytes (40h – FFh) represent user-definable configuration space, such as the information specific to a PC card for use by its accompanying software driver
- Also permits Plug-N-Play
  - base address registers allow an agent to be mapped dynamically into memory or I/O space
  - a programmable interrupt-line setting allows a software driver to program a PC card with an IRQ upon power-up

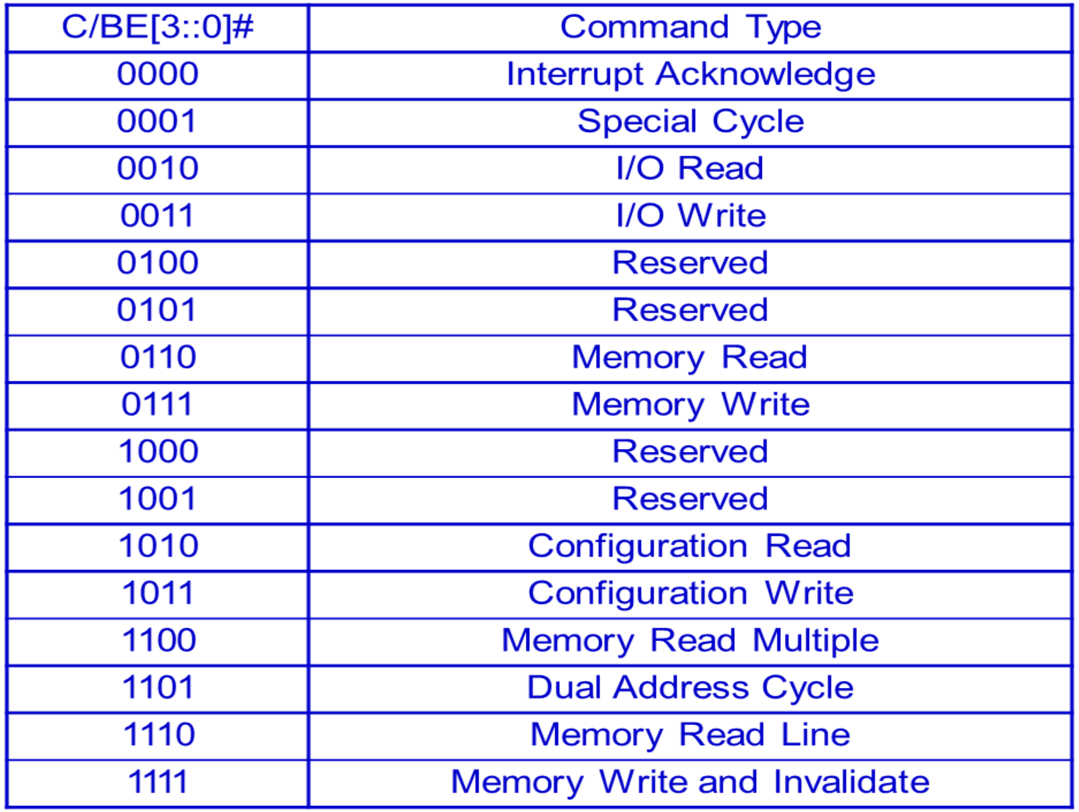
PCI Memory and I/O spaces

- Memory space is used by most everything else – it's the general-purpose address space
  - The PCI spec recommends that a device use memory space, even if it is a peripheral
  - An agent can request between 16 bytes and 2GB of memory space. The PCI spec recommends that an agent use at least 4kB of memory space, to reduce the width of the agent's address decoder

- IO space is where basic PC peripherals (keyboard, serial port, etc.) are mapped
  - The PCI spec allows an agent to request 4 bytes to 2GB of I/O space
  - For x86 systems, the maximum is 256 bytes because of legacy ISA issues

PCI commands

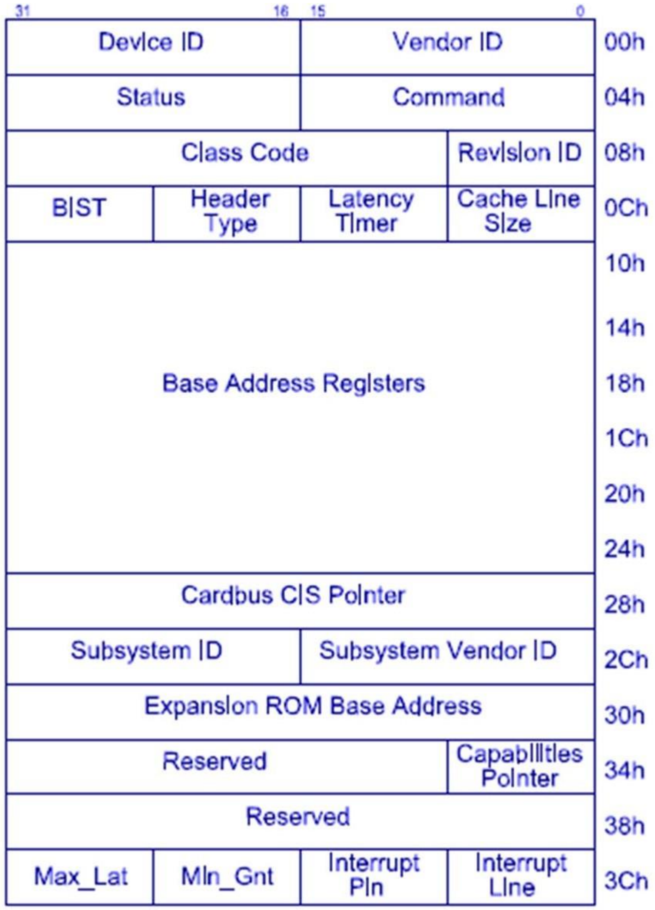
- PCI allows the use of up to 16 different 4-bit commands
  - Configuration commands
  - Memory commands
  - I/O commands
  - Special-purpose commands



C/BE[3::0]#	Command Type
0000	Interrupt Acknowledge
0001	Special Cycle
0010	I/O Read
0011	I/O Write
0100	Reserved
0101	Reserved
0110	Memory Read
0111	Memory Write
1000	Reserved
1001	Reserved
1010	Configuration Read
1011	Configuration Write
1100	Memory Read Multiple
1101	Dual Address Cycle
1110	Memory Read Line
1111	Memory Write and Invalidate

Config Space Header

- ❑ 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



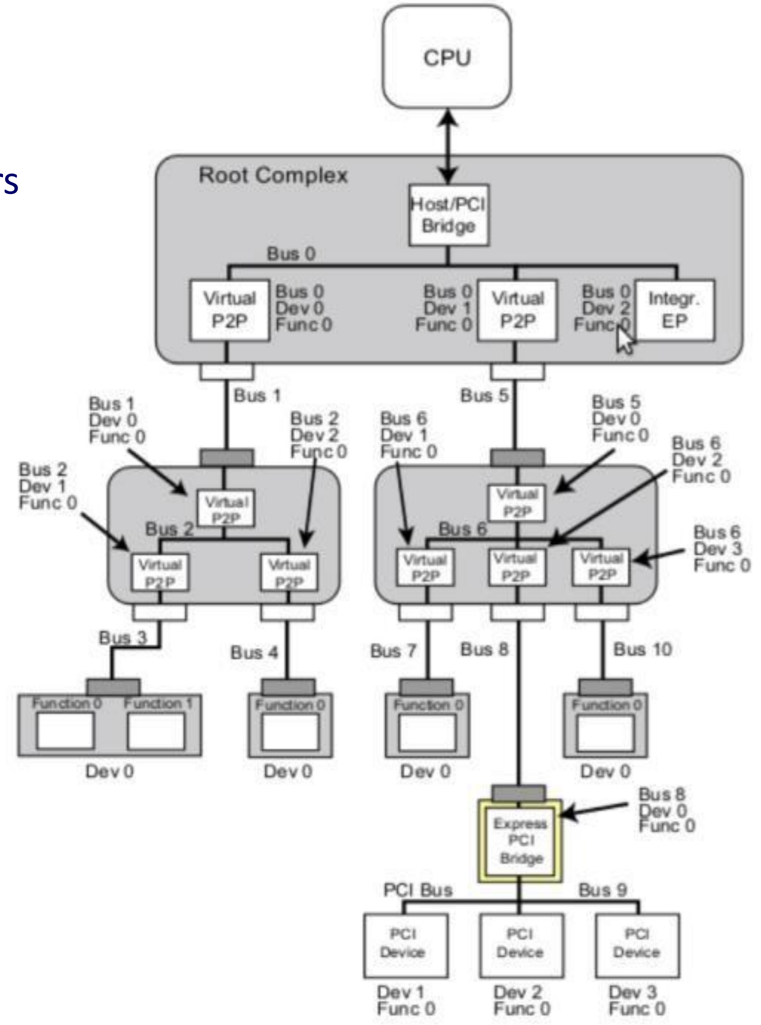
31	16	15	0	
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Latency Timer	Cache Line Size	0Ch
Base Address Registers				10h
				14h
				18h
				1Ch
				20h
				24h
Cardbus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			Capabilities Pointer	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

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

Enumeration

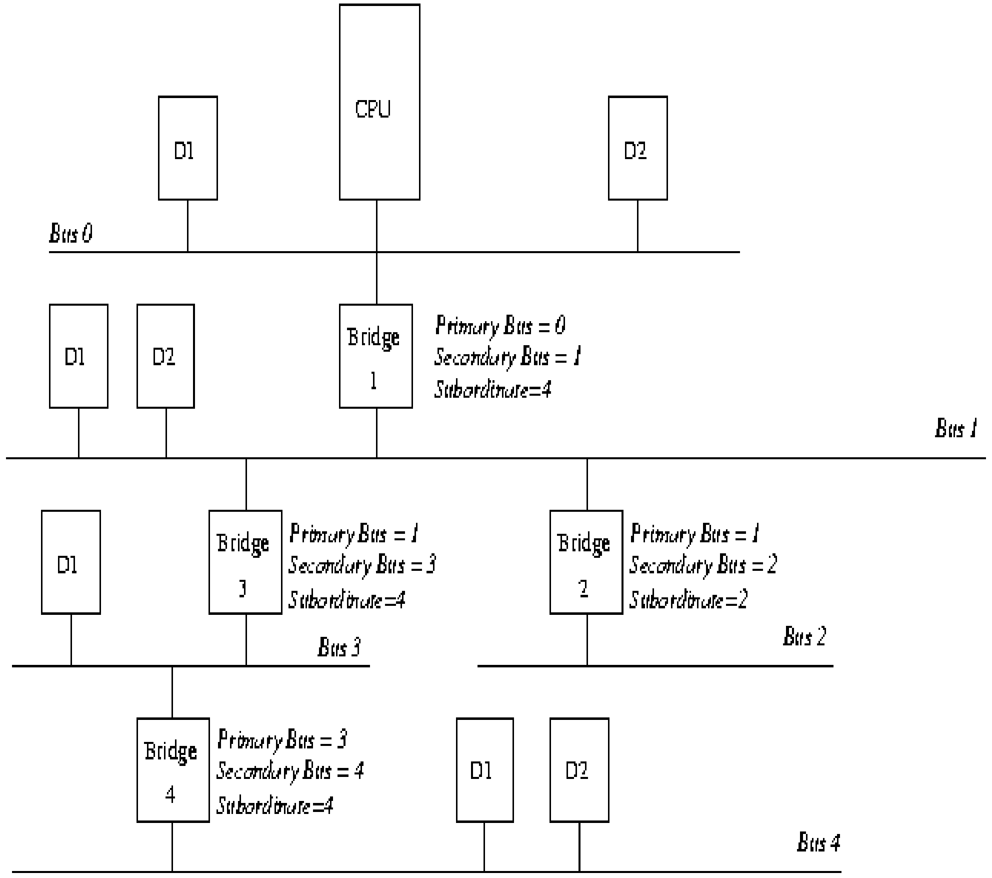
- The process by which configuration software discovers the system topology and assigns bus numbers and system resources.
- On x86 PCIe hierarchy enumeration done by BIOS on hardware initialization state – all registers configured before bootloader.
- System software can re-assign enumeration according to enumeration rules.



Enumeration

Assigning PCIe Bus Numbers :

- **Primary Bus Number** The bus number immediately upstream of the PCI-PCI Bridge
- **Secondary Bus Number** The bus number immediately downstream of the PCI-PCI Bridge
- **Subordinate Bus Number** The highest bus number of all of the busses that can be reached downstream of the bridge

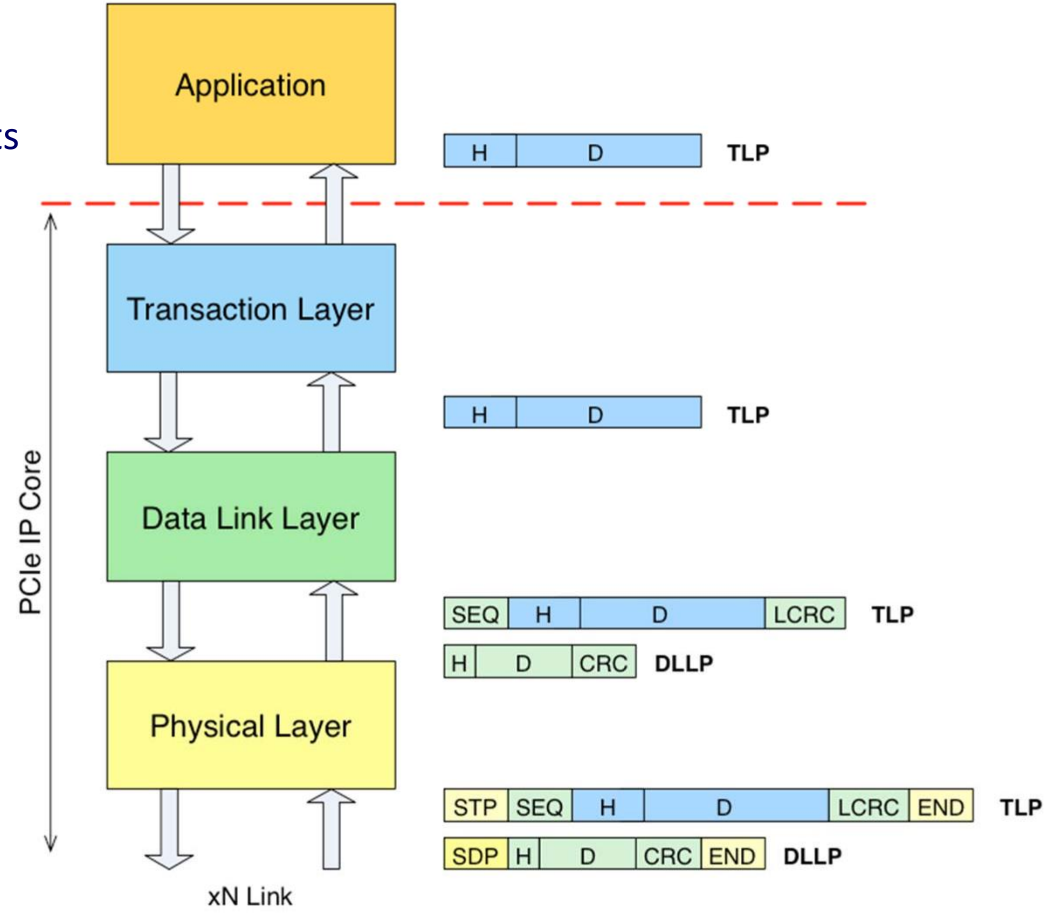


Base Address Register

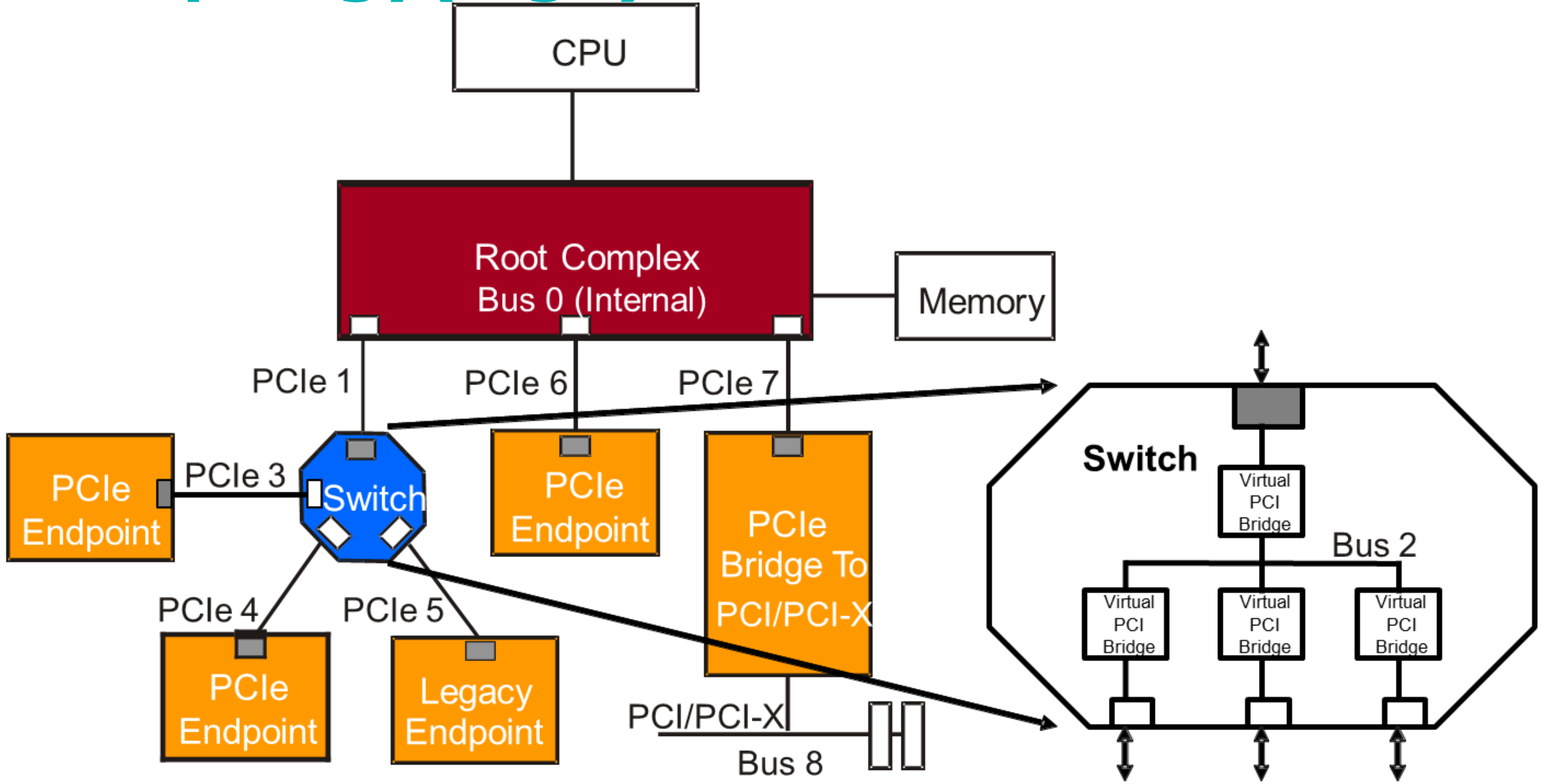
- Base Address Registers point to the location in the system address space where the PCI device will be Mapped. The BAR essentially defines how much memory space your device needs.
- The device RAM, etc. (anything really, per the vendor)
- BARs are R/W and the BIOS programs them to set up the Memory Map
- PCI Configuration Registers provides space for up to 6 BARs (bytes 10h thru 27h)
- BAR[0-5]
- Each BAR is 32-bits wide to support 32-bit address space locations
- Concatenating two 32-bit BARs provides 64-bit addressing capability
- At boot time, the root complex probes the entire PCIE tree. It eventually reaches our device, and attempts to Map all the BARs into its Device Memory Map.

PCIe Layers

- Layered architecture
- Application Data transferred via packets
  - Transaction Layer Packet (TLP)
- PCIe core usually implement the lower three layers
- Protocol handling
  - connection establishing
  - link control
  - flow control
  - power management
  - error detection and reporting



PCIe Topology(E.g..) Root & Switch



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:

Memory Read or Memory Write.

- Used to transfer data from or to a memory mapped location
- also supports a locked memory read transaction variant.

I/O Read or I/O Write.

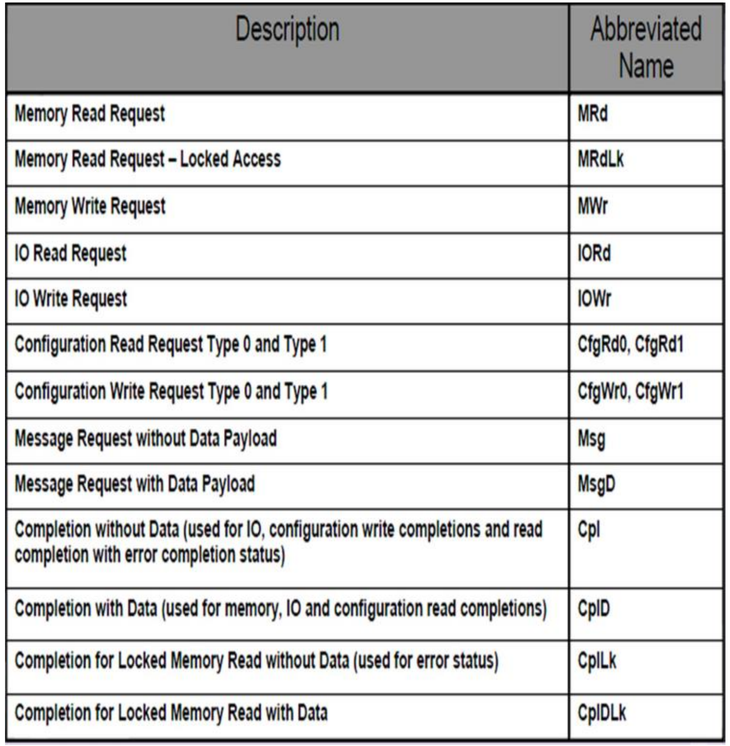
- Used to transfer data from or to an I/O location
- restricted to supporting legacy endpoint devices.

Configuration Read or Configuration Write

- Used to discover device capabilities, program features, and check status in the 4KB PCI Express configuration space.

Messages.

- Handled like posted writes. Used for event signaling and general purpose messaging.

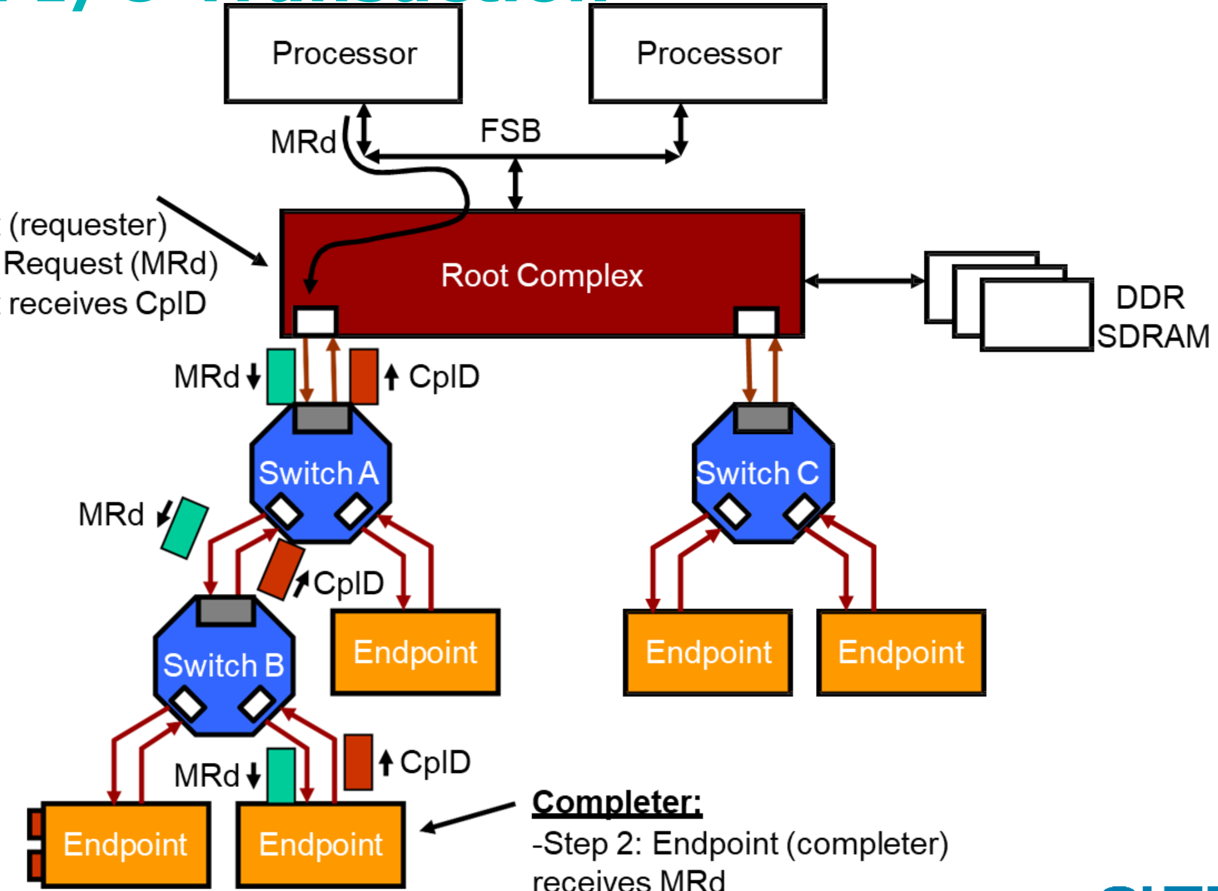


Description	Abbreviated Name
Memory Read Request	MRd
Memory Read Request – Locked Access	MRdLk
Memory Write Request	MWr
IO Read Request	IORd
IO Write Request	IOWr
Configuration Read Request Type 0 and Type 1	CfgRd0, CfgRd1
Configuration Write Request Type 0 and Type 1	CfgWr0, CfgWr1
Message Request without Data Payload	Msg
Message Request with Data Payload	MsgD
Completion without Data (used for IO, configuration write completions and read completion with error completion status)	Cpl
Completion with Data (used for memory, IO and configuration read completions)	CplD
Completion for Locked Memory Read without Data (used for error status)	CplLk
Completion for Locked Memory Read with Data	CplDLk

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

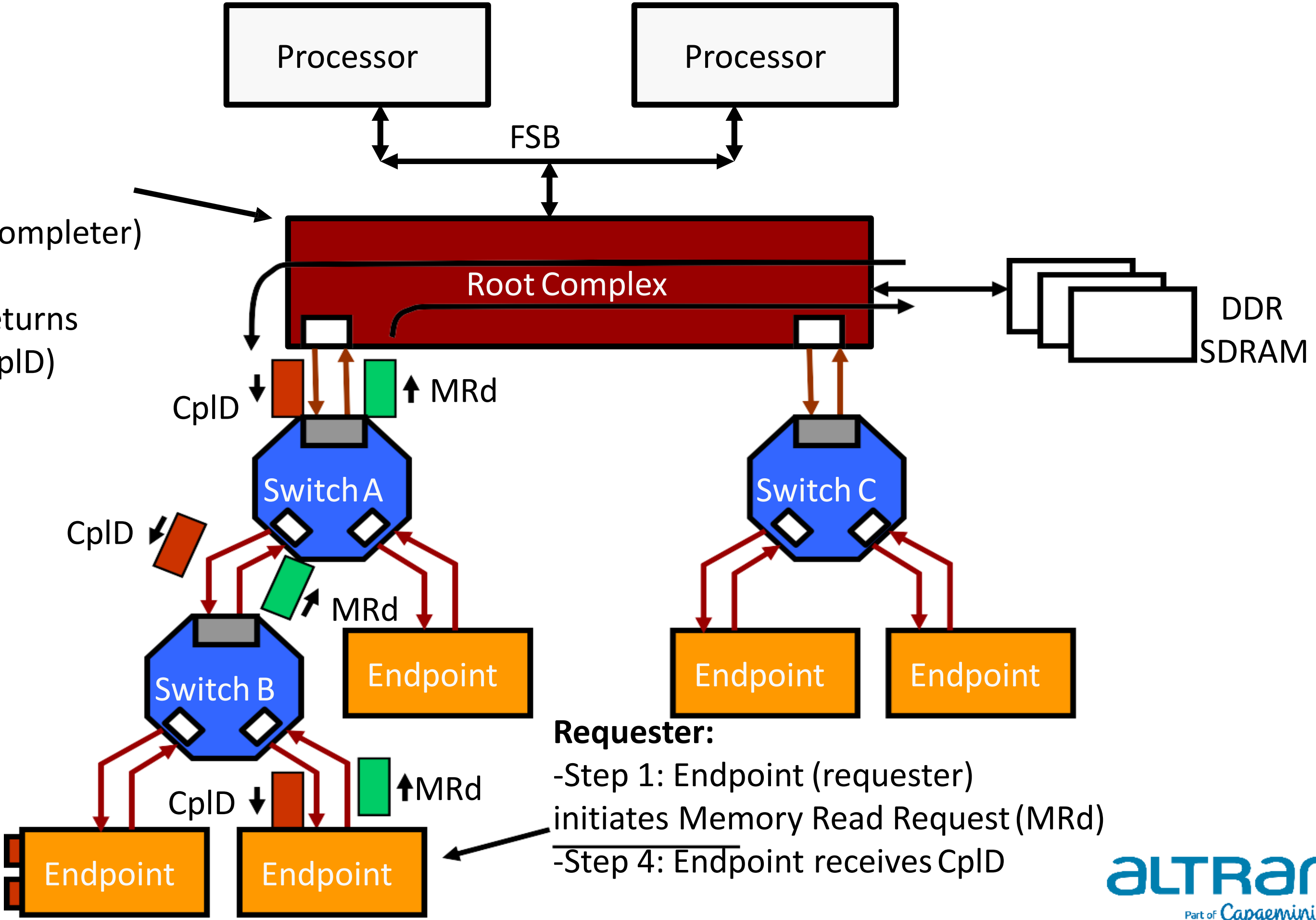


Requester:
-Step 1: Root Complex (requester) initiates Memory Read Request (MRd)
-Step 4: Root Complex receives CplD

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

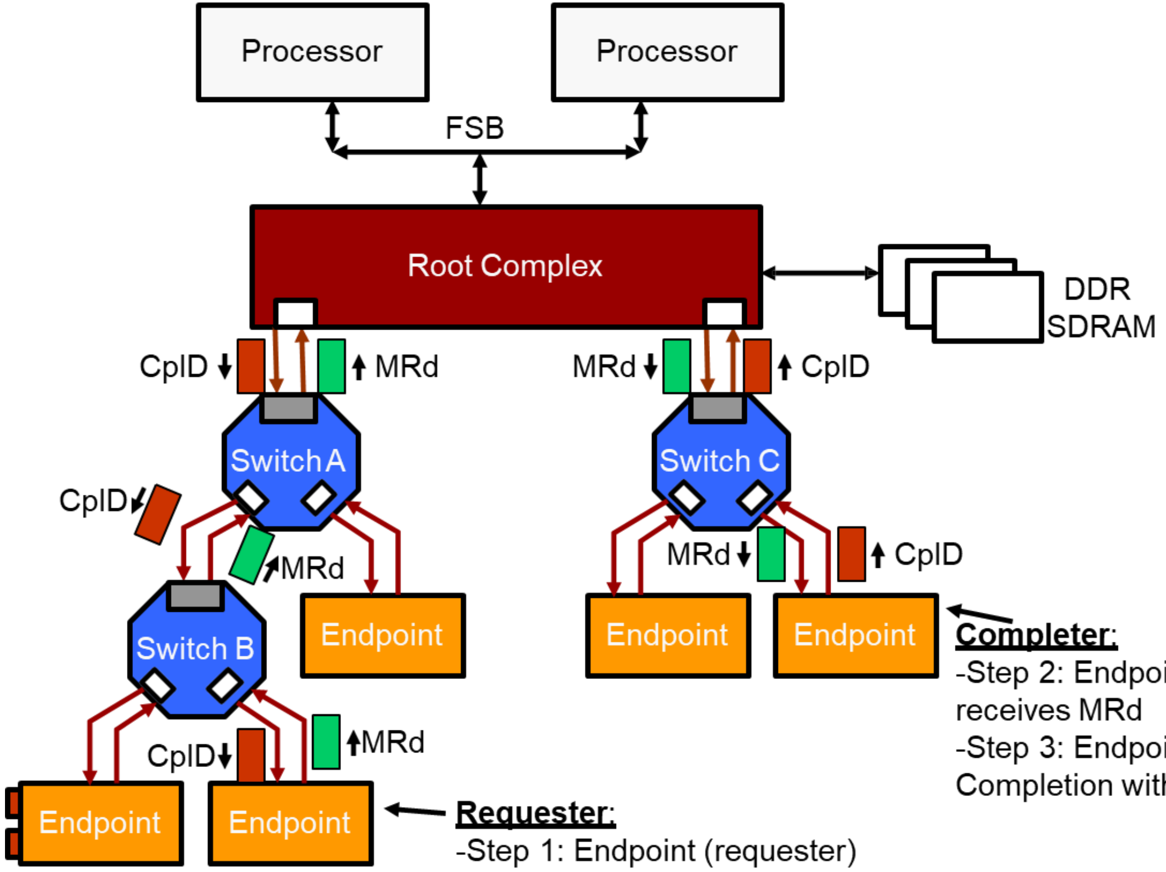
DMA Transactions

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

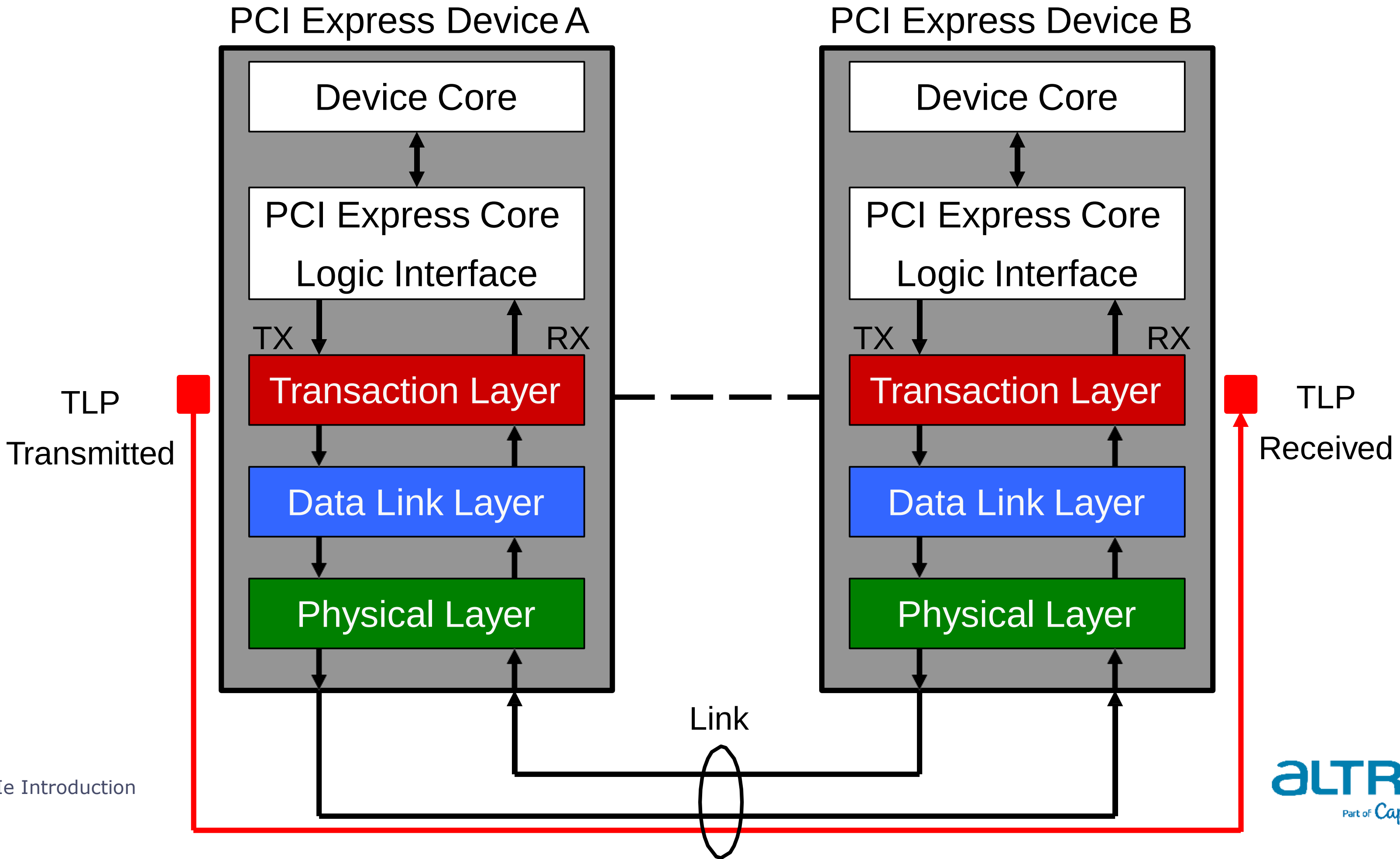
Peer to Peer Transactions



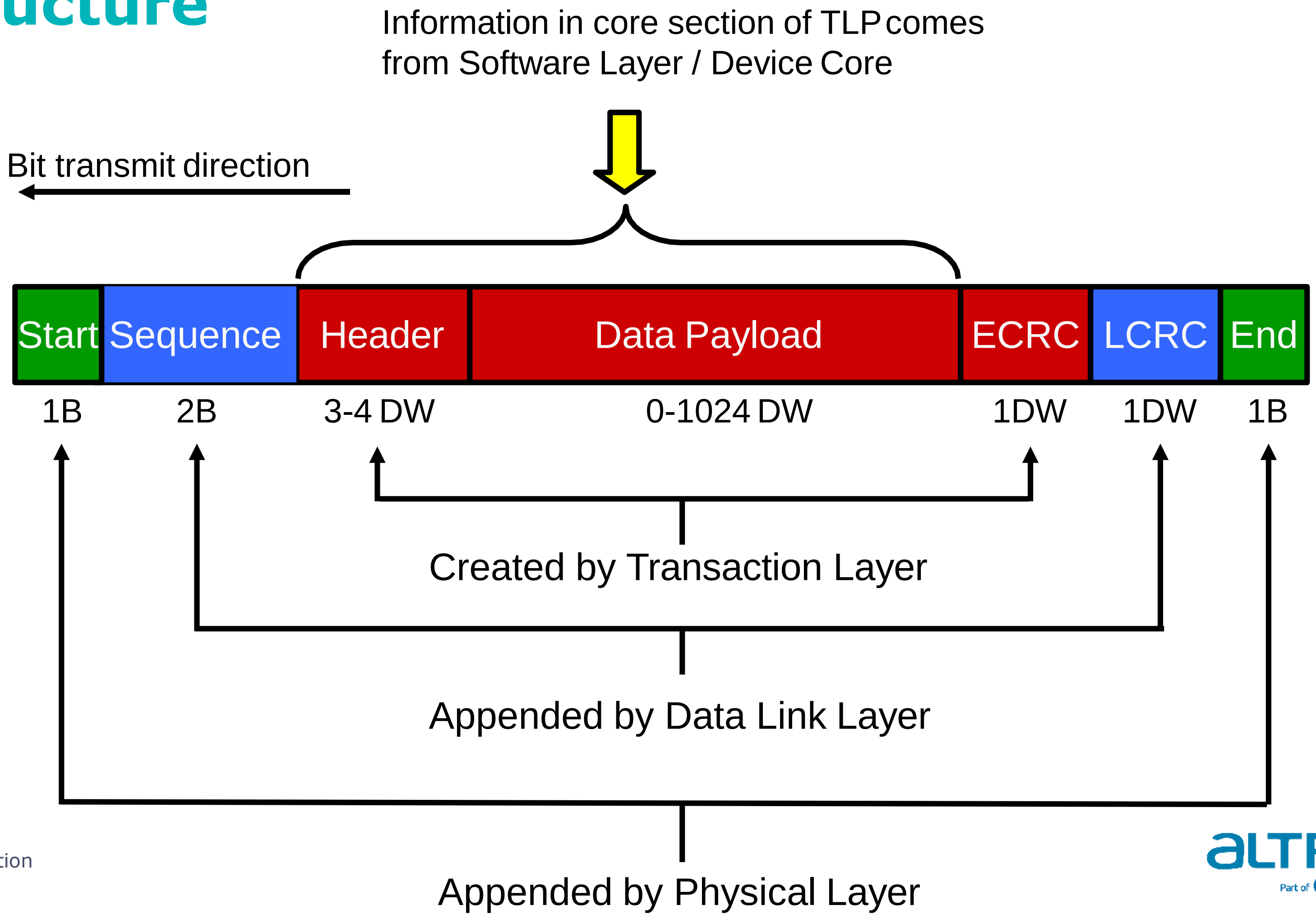
Completer:
-Step 2: Endpoint (completer) receives MRd
-Step 3: Endpoint returns Completion with data (CplD)

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

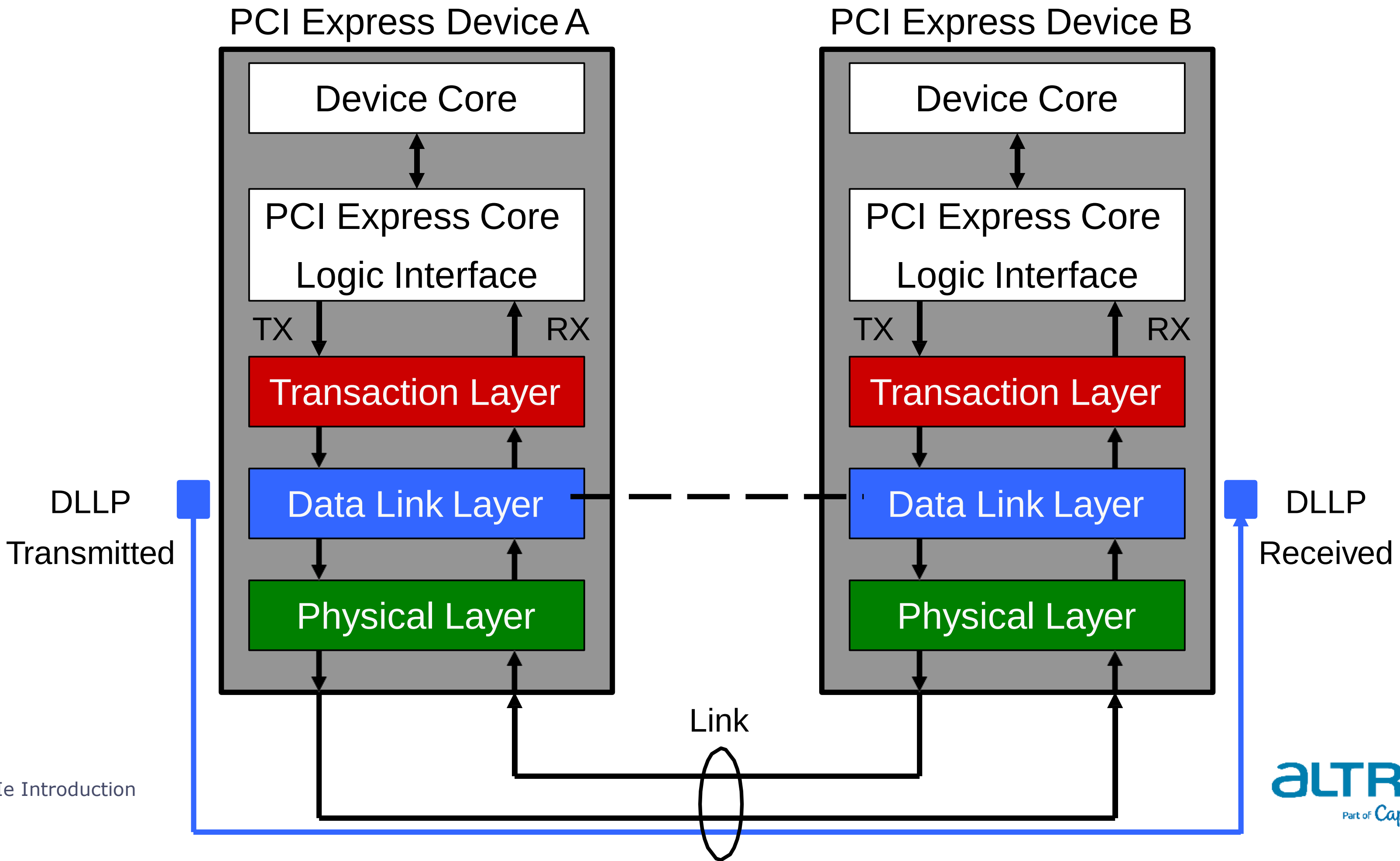
TLP Origin and Destination



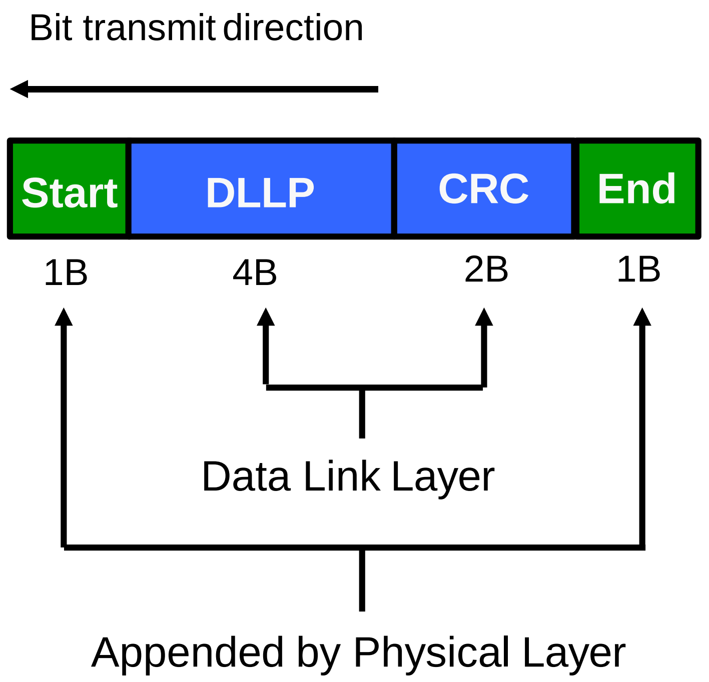
TLP Structure


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

DLLP Origin and Destination

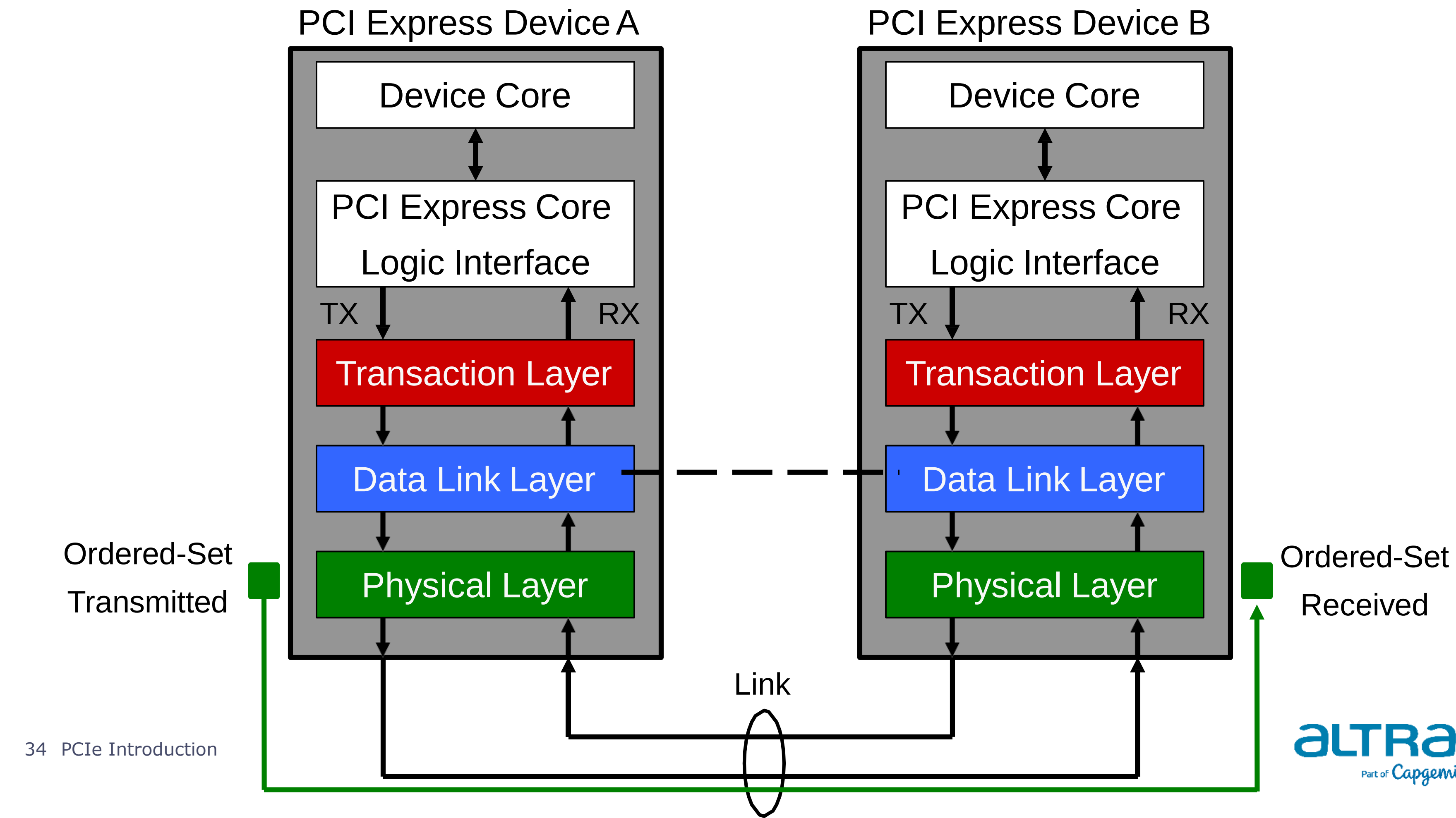


DLLP Structure

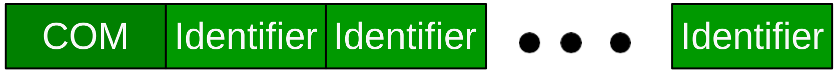


- ACK / NAK Packets
- Flow Control Packets
- Power Management Packets
- Vendor Defined Packets

Ordered-set Origin and Destination



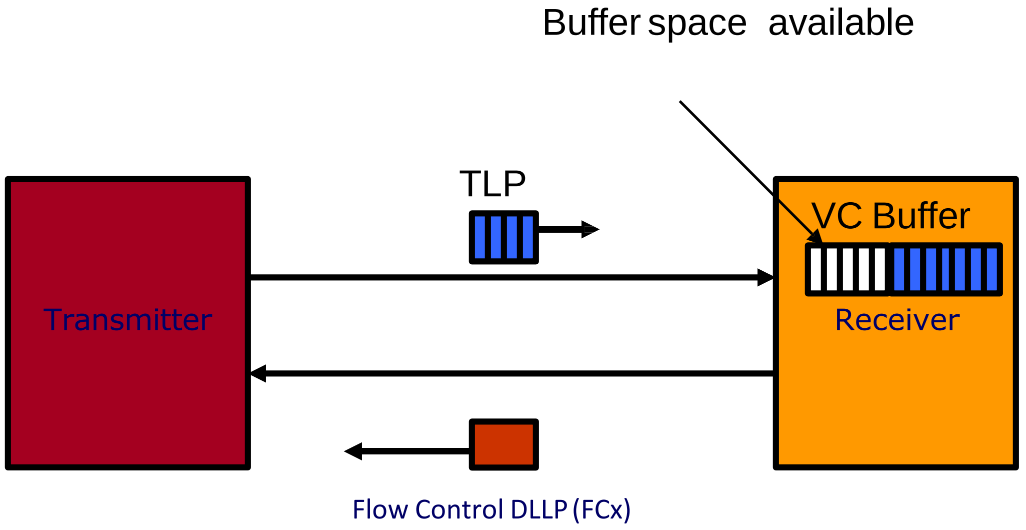
Ordered-set Structure



- 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)
  - 16 characters

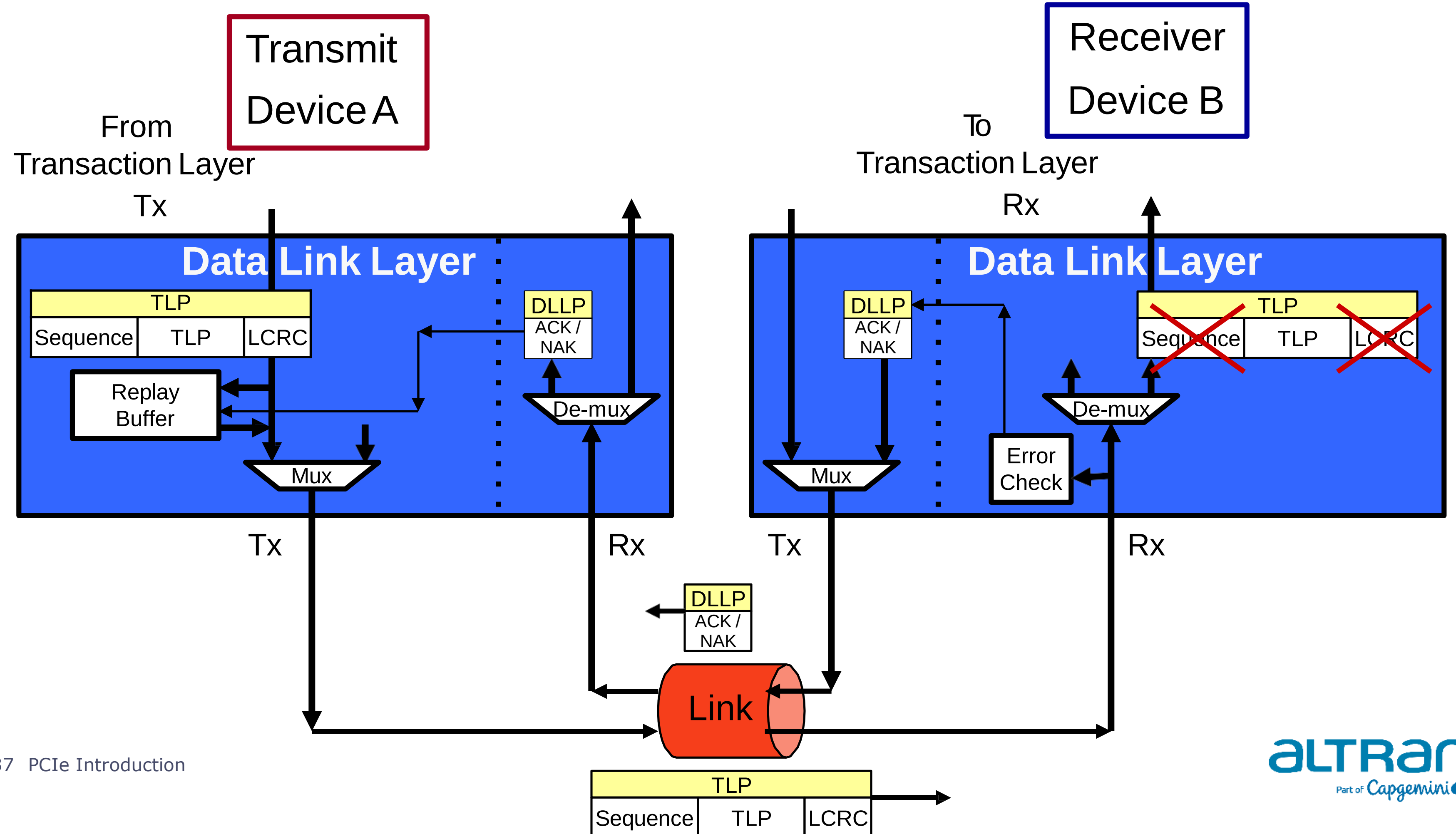
PCIe Flow Control

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

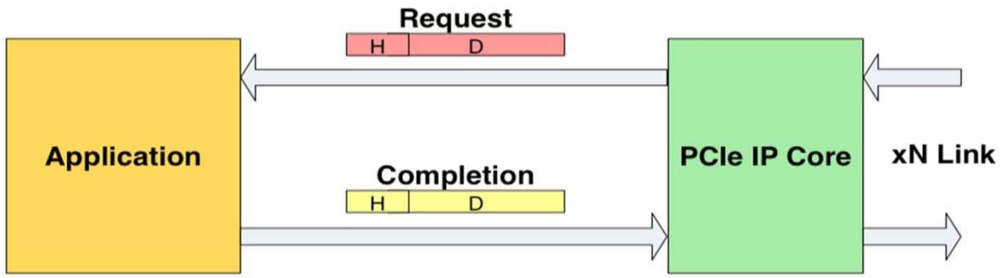


Receiver sends Flow Control Packets (FCP) which are a type of DLLP (Data Link Layer Packet) to provide the transmitter with credits so that it can transmit packets to the receiver

ACK/NACK Protocol Overview

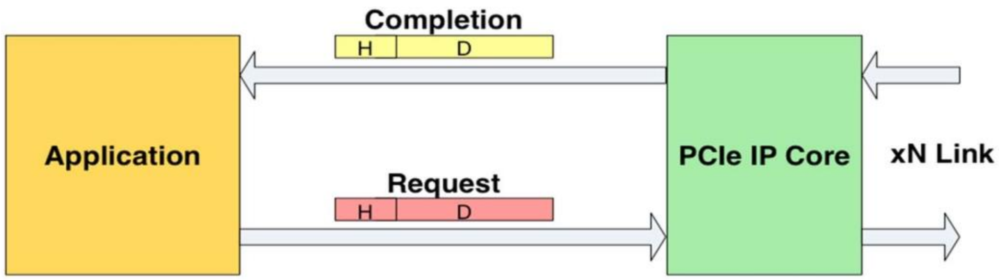


Incoming Requests



- **Incoming requests perform local subsystem read or write**
- **Some incoming requests require sending completion TLP**
- **Completion TLP rules**
  - Must form completion packets with respect to Max_Payload and Read Completion Boundary
  - Must correctly encode fields in completion TLP
  - Completion address in packet differs (I/O x Memory)
- **Application must correctly report a request processing problem to the core**

Outgoing Requests



- ❑ **Outgoing Requests are generated by the application**
- ❑ **There is a set of rules for forming outgoing request TLP**
  - ➢ Must be identified by unique Tag
  - ➢ Read requests restricted by Max_Read_Request_Size
  - ➢ Write requests restricted by Max_Payload
  - ➢ Must not cross 4KB address boundary
- ❑ **Violations will result in request being discarded and error detected**
- ❑ **Completion request processing**
  - ➢ Completions for multiple outstanding requests must be processed by Tag
  - ➢ Must have correct values in lower addresses to process multiple TLPs
  - ➢ Must process both Unsupported Request and Completer Abort responses

How PCI Works

Once a new device has been inserted into a PCI slot on the motherboard

Operating System Basic Input/output System (BIOS) initiates Plug and Play (PnP) BIOS.

PnP BIOS scans the PCI bus for any new hardware connected to the bus. If new hardware is found, it will ask for identification.

The device will respond with its identification and send its device ID to the BIOS through the bus.

PnP checks the Extended System Configuration Data (ESCD) to make sure the configuration data already exists for the card. (If the card is new, then there will be no data for it.)

PnP will assign an Interrupt Request Line, Direct Memory Access, memory address and Input/output settings to the card, then stores the information in the ESCD.

When the Windows software loads, it will check the PCI bus and the ESCD to see if there is new hardware.

Windows will alert the user that new hardware has been found if there is new hardware installed and will also identify the hardware.

Windows will determine the device and attempt to install its driver.

PCI Applications

- PCI Express operates in consumer, server, and industrial applications, as a motherboard-level interconnect (to link motherboard-mounted peripherals), a passive backplane interconnect and as an expansion card interface for add-in boards.
  - External GPUs
    - 3D Graphic Cards
    - Sound cards
  - Network
    - 10G or multigigabit cards
    - 802.11ax Wifi cards
  - Enterprise Storage
    - RAID Cards
    - SSD drives
    - Flash memories

PCIe 6.0

- Planned to be released in 2021
- Bandwidth is expected to increase to 64 GT/s, yielding 126 GB/s in each direction and 256GB/s(bidirectional) in a 16-lane configuration
- 4-level pulse-amplitude modulation (PAM-4) signalling will be used
- Reed—Solomon forward error correction(RS-FEC), a predictive error correction.
- Maintains backward compatibility to previous generations
- PCI-Express 6.0 is also likely to be useful in self-driving systems, the industrial IoT, and any system that juggles multiple sensors to combine input from multiple peripherals into a cohesive whole.

REVISION HISTORY

Rev No.	Date	Description of Change	Author	Review and Approved By
1.0		Initial Draft		Tanmoy B (L&D)

altran

Part of Capgemini