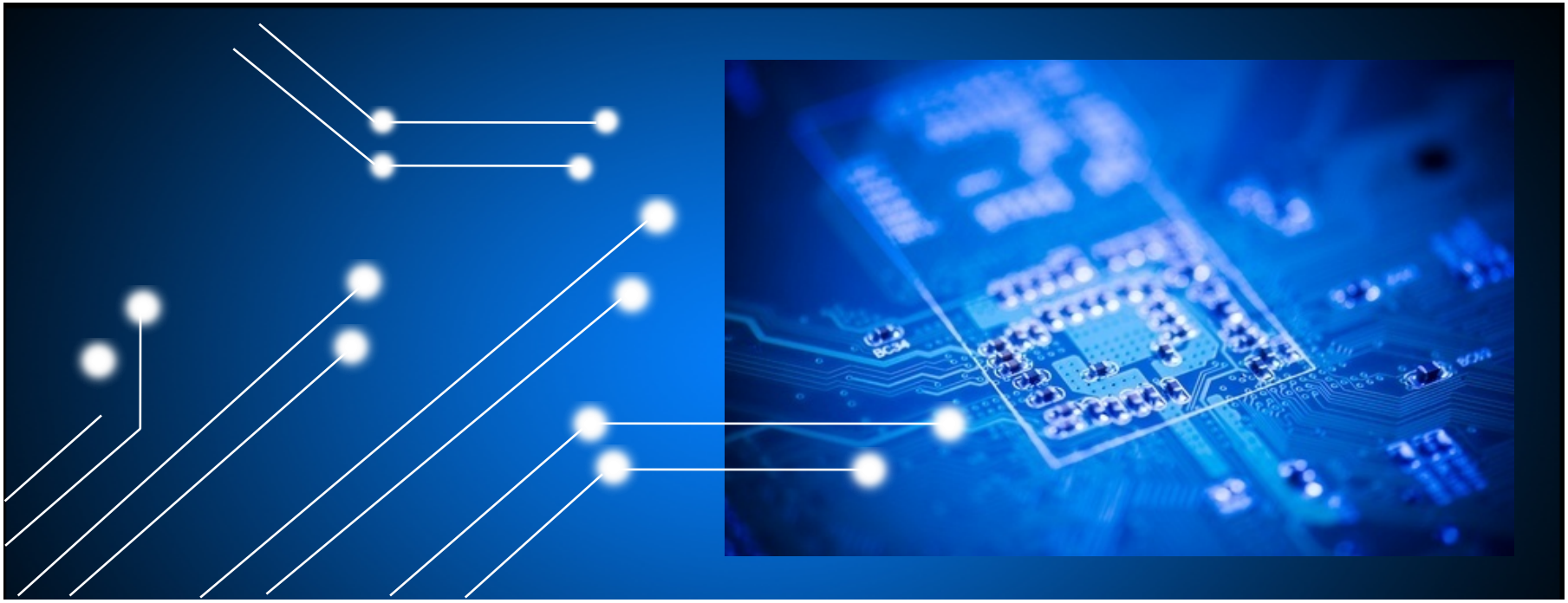


PCIe Protocol And Linux Driver Introduction

Shawn Lin
2016.06.12

Rockchip

www.rock-chips.com
TEL: 86-591-83991906 FAX: 86-591-83951833

Agenda

- **Introduction**
- Layering Overview
- Transaction Basics
- Configuration Overview
- PCIE Capability
- Address Space and TLP Routing
- Linux Software Stack(RC)
- Linux Driver(EP)

Introduction

Protocol Evolution

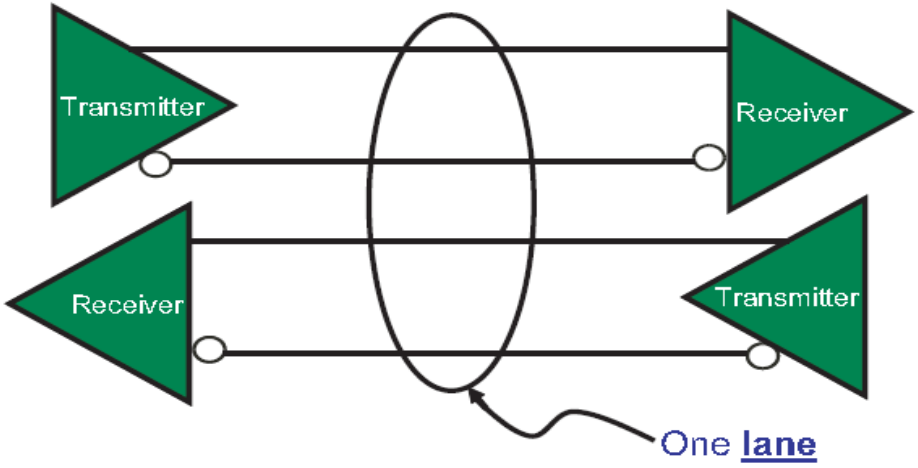
PCI Express (**Peripheral Component Interconnect Express**), officially abbreviated as PCIe, is a high-speed serial computer expansion bus standard designed to replace the older PCI, PCI-X, and AGP bus standards .

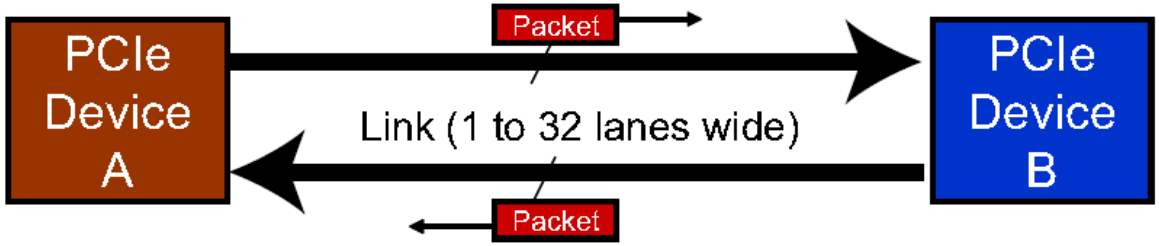
Scalable, Flexible
High performance interconnect

Introduction

Physical link and lane

One lane consists of a pair of differential TX and a pair of differential RX. It allows a Link to be made up 1, 2, 4, 8, 12, 16, or 32 Lanes， lane counts is link width. But, X32 （32*4=128） lanes is Unusual, due to pin cost is intolerability.





Dual-simplex link

Introduction

- Performance

PCI Express link performance[27][28]

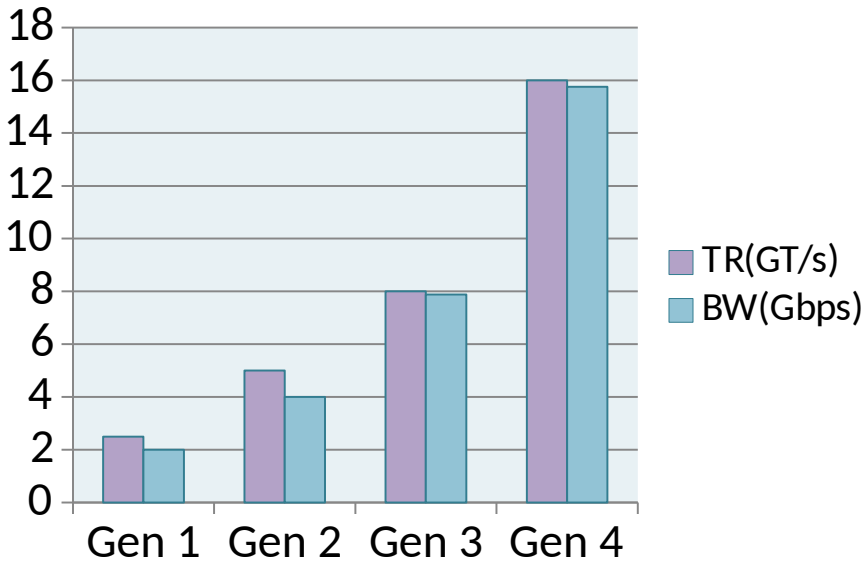
PCI Express version	Line code	Transfer rate[a]	Throughput[a]			
			×1	×4	×8	×16
1.0	8b/10b	2.5 GT/s	250 MB/s	1 GB/s	2 GB/s	4 GB/s
2.0	8b/10b	5 GT/s	500 MB/s	2 GB/s	4 GB/s	8 GB/s
3.0	128b/130b	8 GT/s	984.6 MB/s	3.938 GB/s	7.877 GB/s	15.754 GB/s
4.0 (future)	*128b/130b*	*16 GT/s*	*1.969 GB/s*	*7.877 GB/s*	*15.754 GB/s*	*31.508 GB/s*

Gen1/Gen2 :8b/10b

Gen3 : 128b/130b

Gen4 : 128b/130b ?

one lane one direction



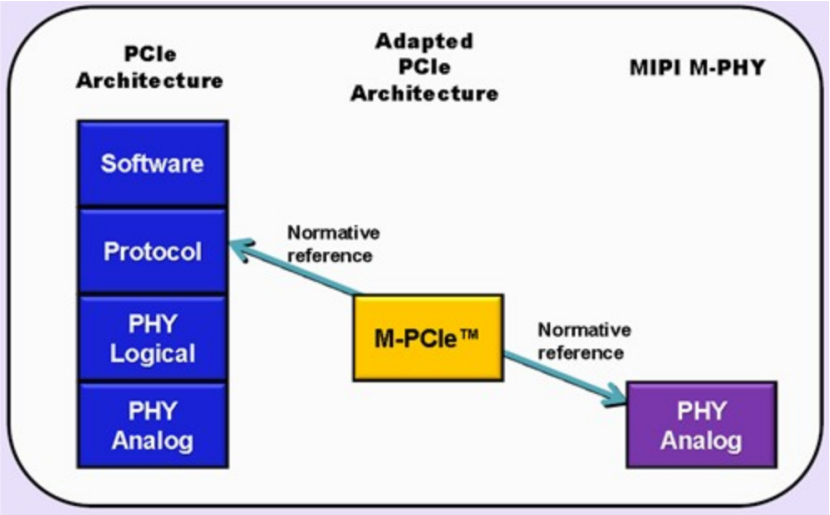
M.2(NGFF) SSD

Link Width	x1	x2	x4	x8	x12	x16	x32
Gen1 Bandwidth (GB /s)	0.5	1	2	4	6	8	16
Gen2 Bandwidth (GB/s)	1	2	4	8	12	16	32
Gen3 Bandwidth (GB/s)	2	4	8	16	24	32	64

Introduction

- Markets and Applications

Storage
RAID
NVMe
SSD

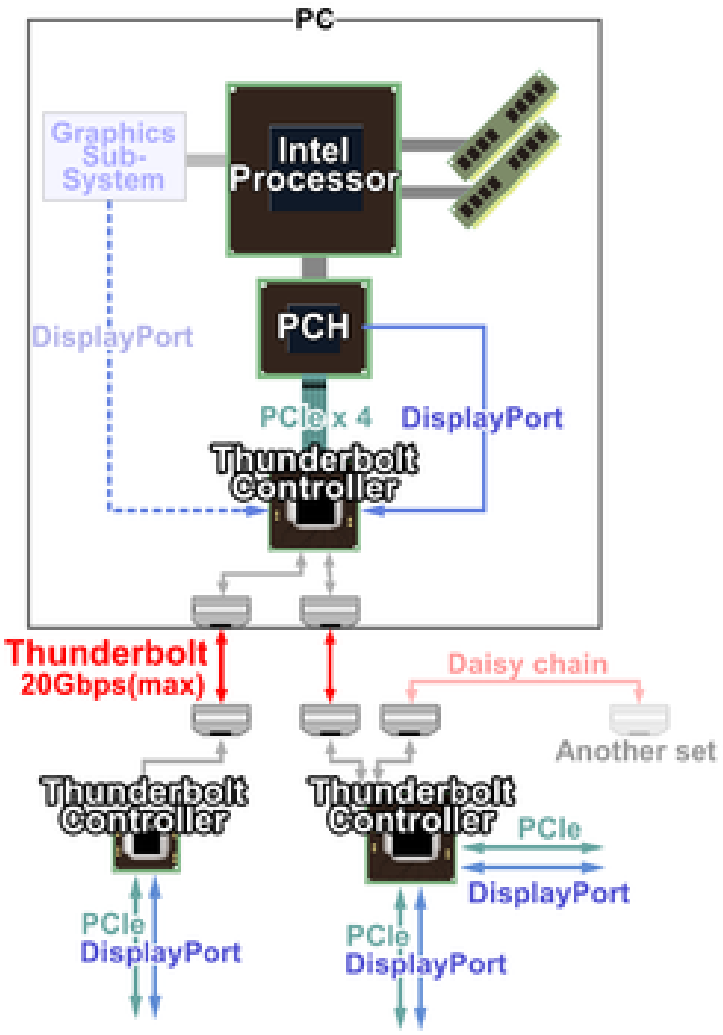


RATE A-series (Mbps)	RATE B-series[1] (Mbps)	High-Speed GEARs
1248	1457.6	HS-G1 (A/B)
2496	2915.2	HS-G2 (A/B)
4992	5830.4	HS-G3 (A/B)
9984	11660.8	HS-G4 (A/B)

THUNDERBOLT™

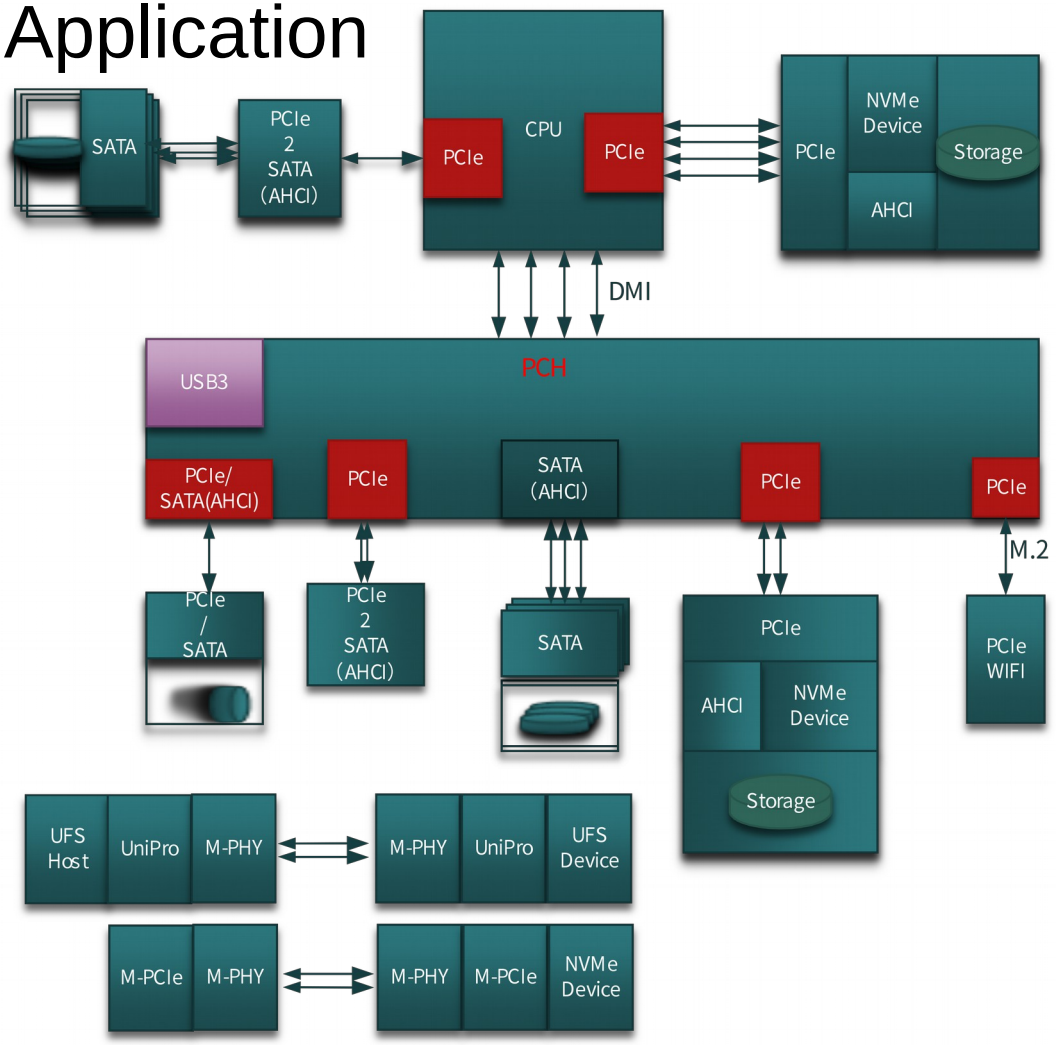
Thunderbolt is the brand name of a hardware interface that allows the connection of external peripherals to a computer. Thunderbolt 1 and 2 use the same connector as Mini DisplayPort (MDP), while Thunderbolt 3 uses USB Type-C. It was initially developed and marketed under the name Light Peak,[1] and first sold as part of a consumer product on February 24, 2011.[2]

Thunderbolt combines PCI Express (PCIe) and DisplayPort (DP) into one serial signal, and additionally provides DC power, all in one cable. Up to six peripherals may be supported by one connector through various topologies.



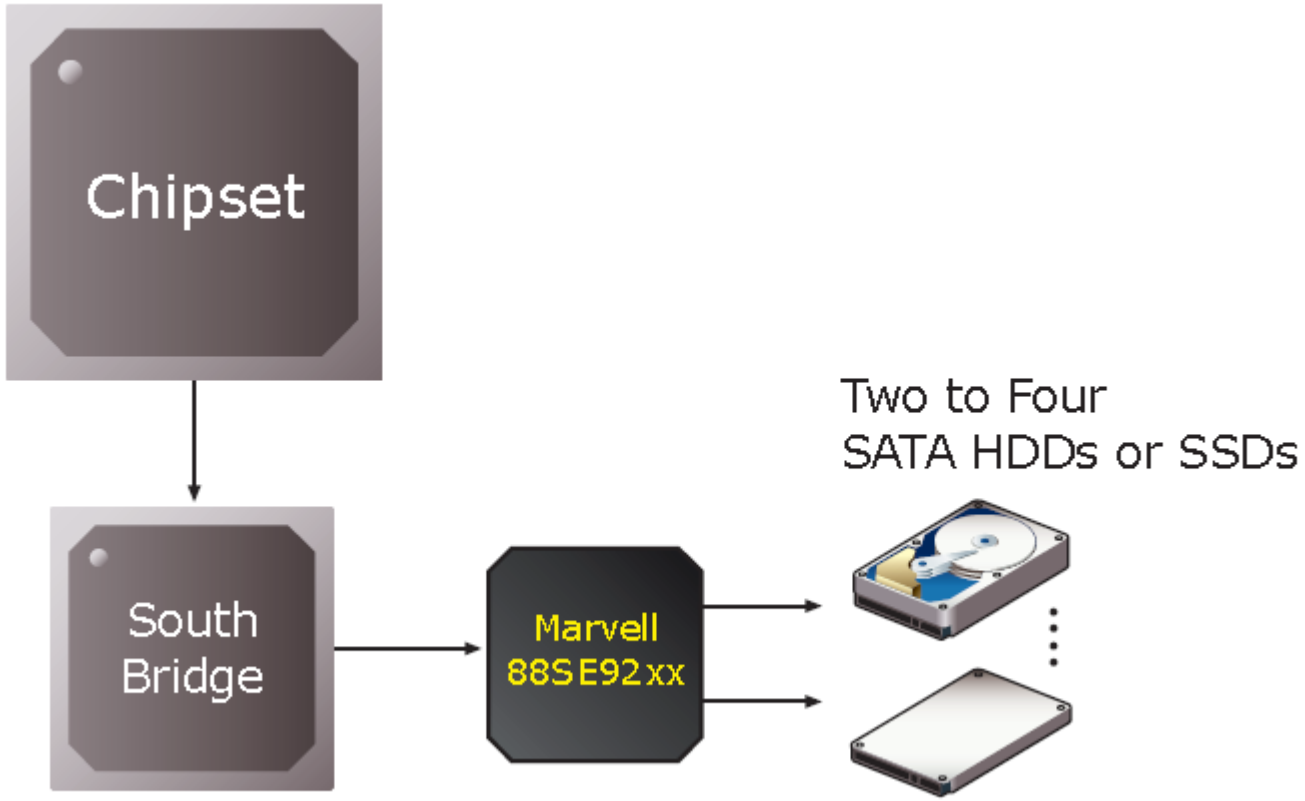
Introduction

- Storage Application



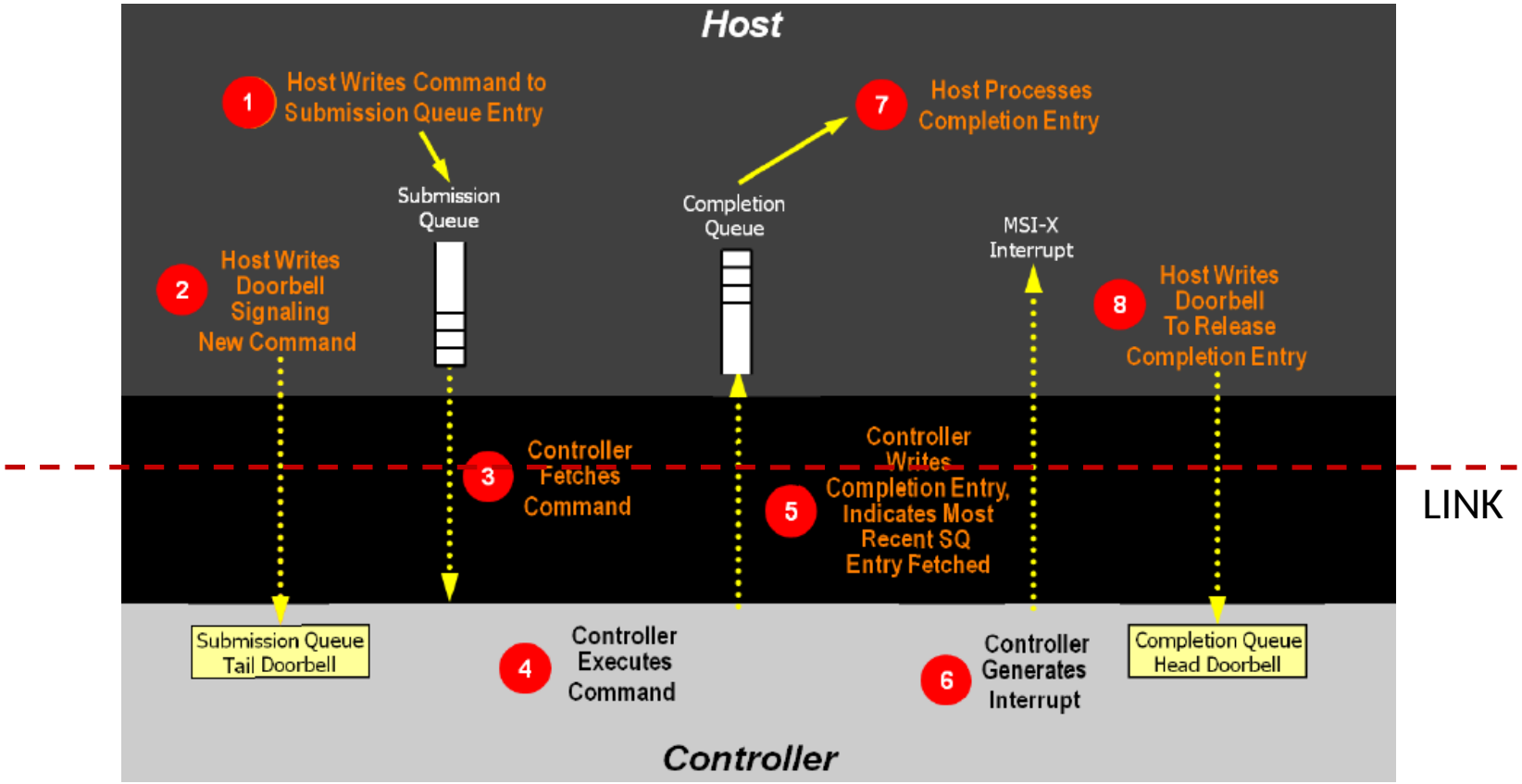
Introduction

- PCIe to SATA



Introduction

- NVMe



Introduction

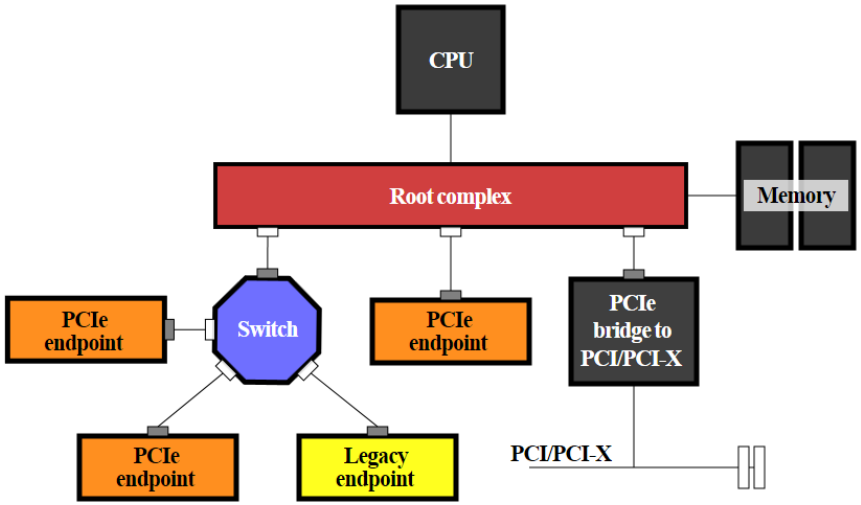
A Simple PCIe Topology

✓**Root Complex**: The Root Complex can be understood as the interface between the system CPU and the PCIe topology, with PCIe Ports labeled as "Root Ports" in configuration space

✓**Switch**: Fanout or aggregation capability, act as router

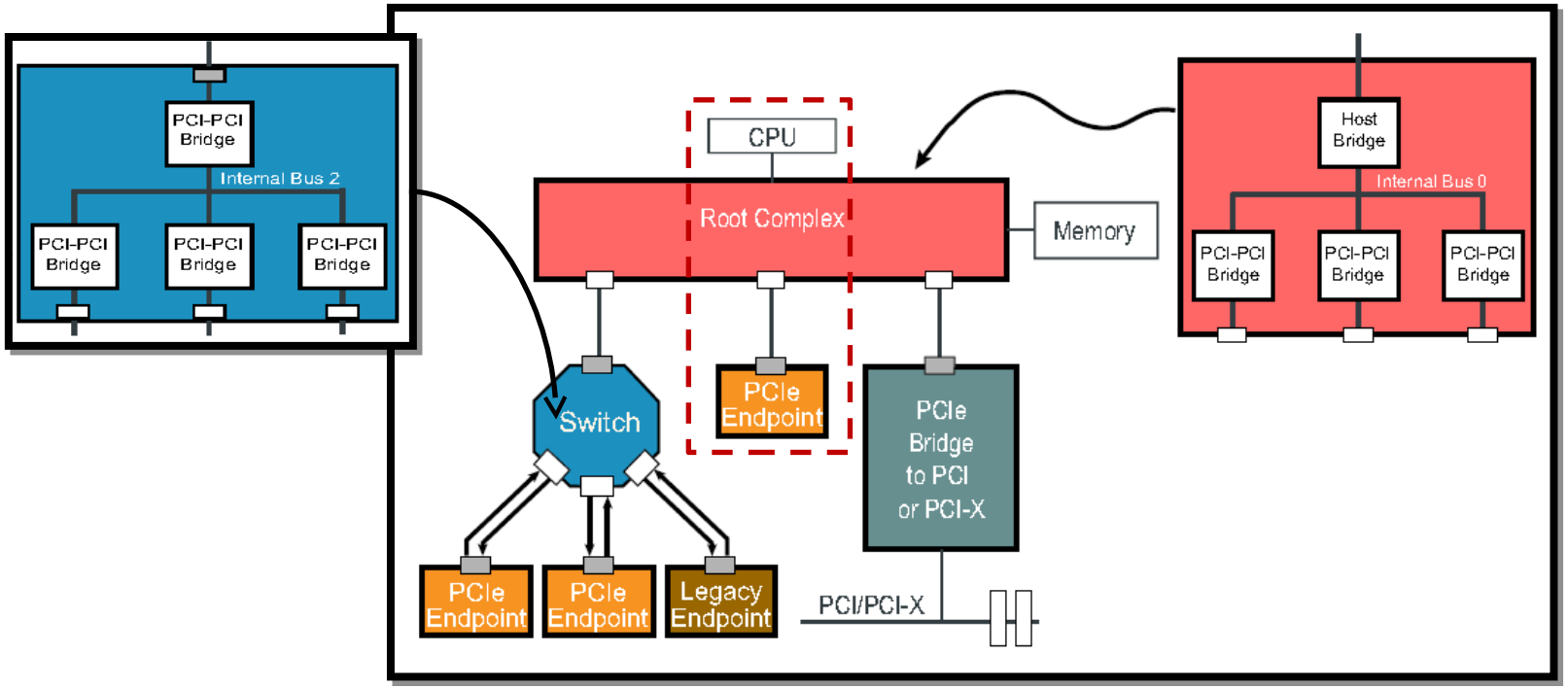
✓**Bridge**: Interface to other buses, such as PCI/PCI-X or PCIe

✓**Endpoint**: Act as initiators and Completers of transactions on the bus. Have legacy EP and native EP



Introduction

- A Simple PCIe Topology



Introduction

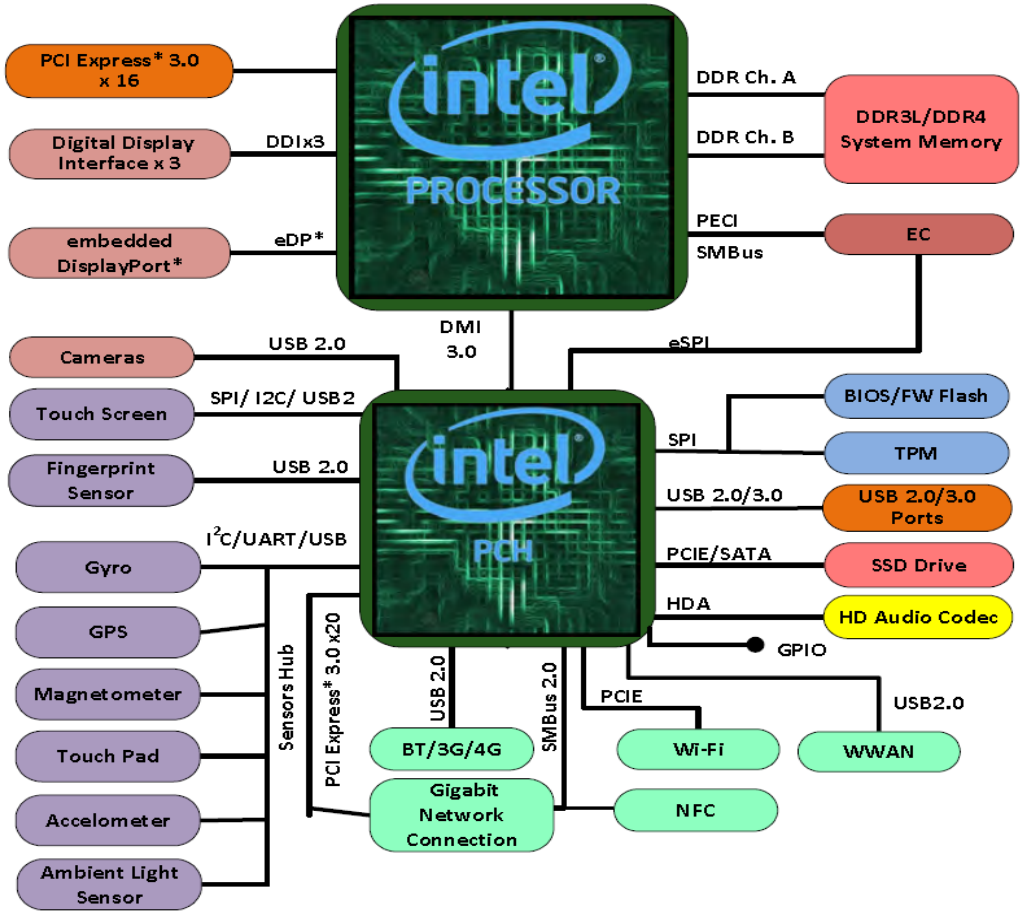
A Complex PCIe Topology

Uncore: Un core

QPI: quick path interconnect

DMI: Direct Media Interface
FDI： Flexible Display Interface

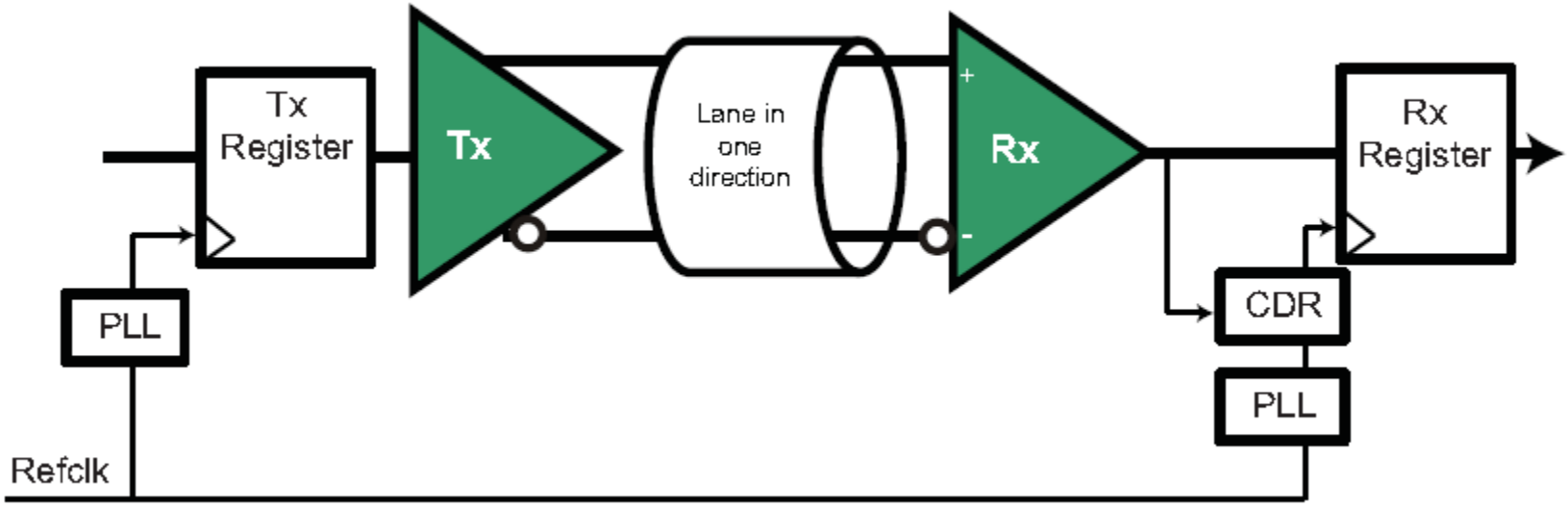
IOH: South Bridge
PCH： platform Controller Hub



Introduction

Refclk Architecture overview

- **Common Refclk Architecture**
- Data Clocked Rx Architecture
- Separate Refclk Architecture

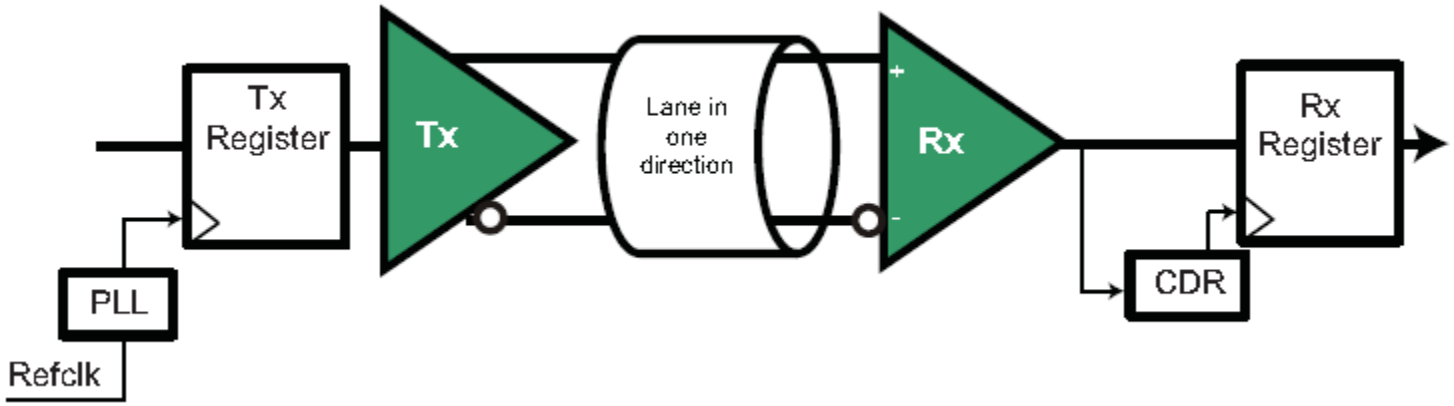


Common Refclk Architecture

Introduction

Refclk Architecture overview

- Common Refclk Architecture
- **Data Clocked Rx Architecture**
- Separate Refclk Architecture

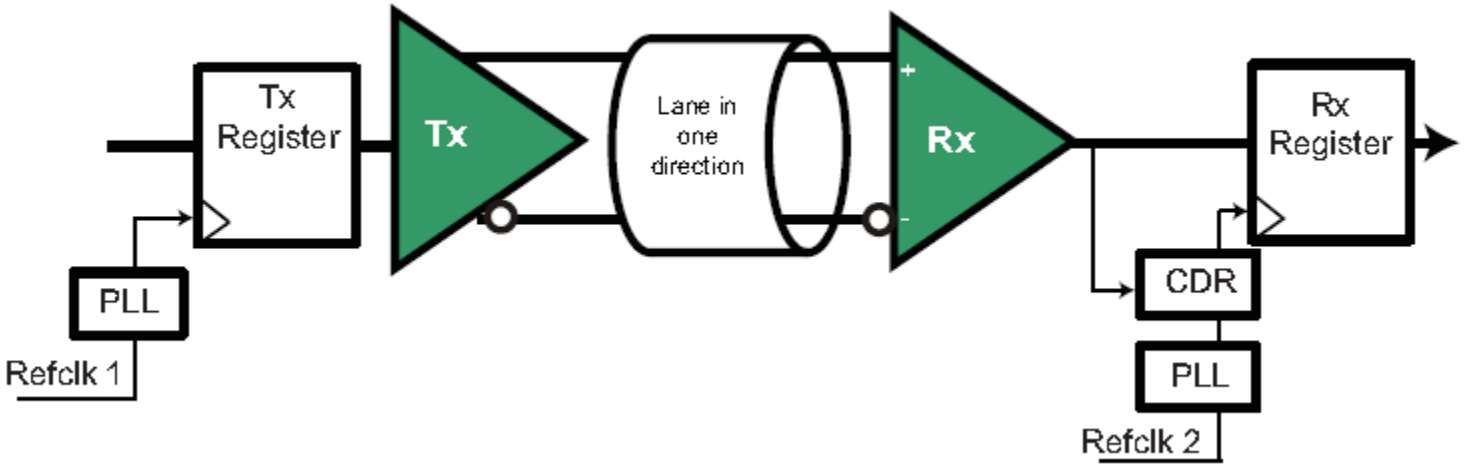


Data Clocked Rx Architecture

Introduction

Refclk Architecture overview

- Common Refclk Architecture
- Data Clocked Rx Architecture
- **Separate Refclk Architecture**



Separate Refclk Architecture

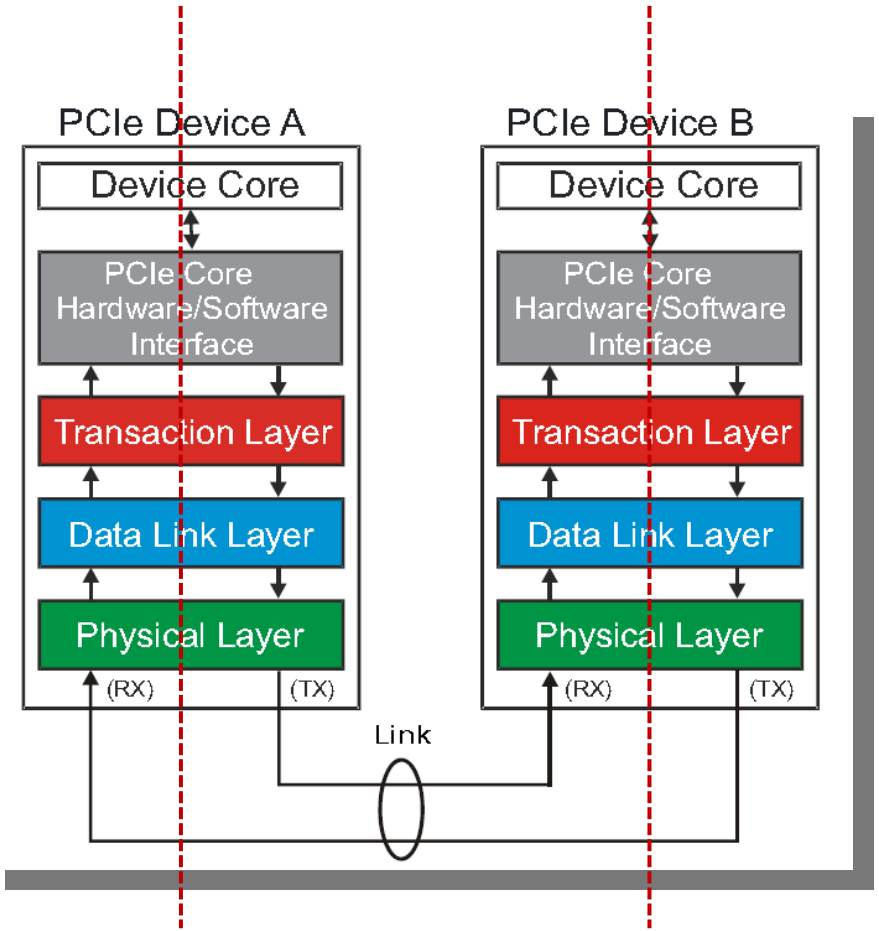
Agenda

- Introduction
- **Layering Overview**
- Transaction Basics
- Configuration Overview
- Address Space and TLP Routing
- IP Integration
- IP Verification
- IP Application

PCIe Layering Overview

Device Layers

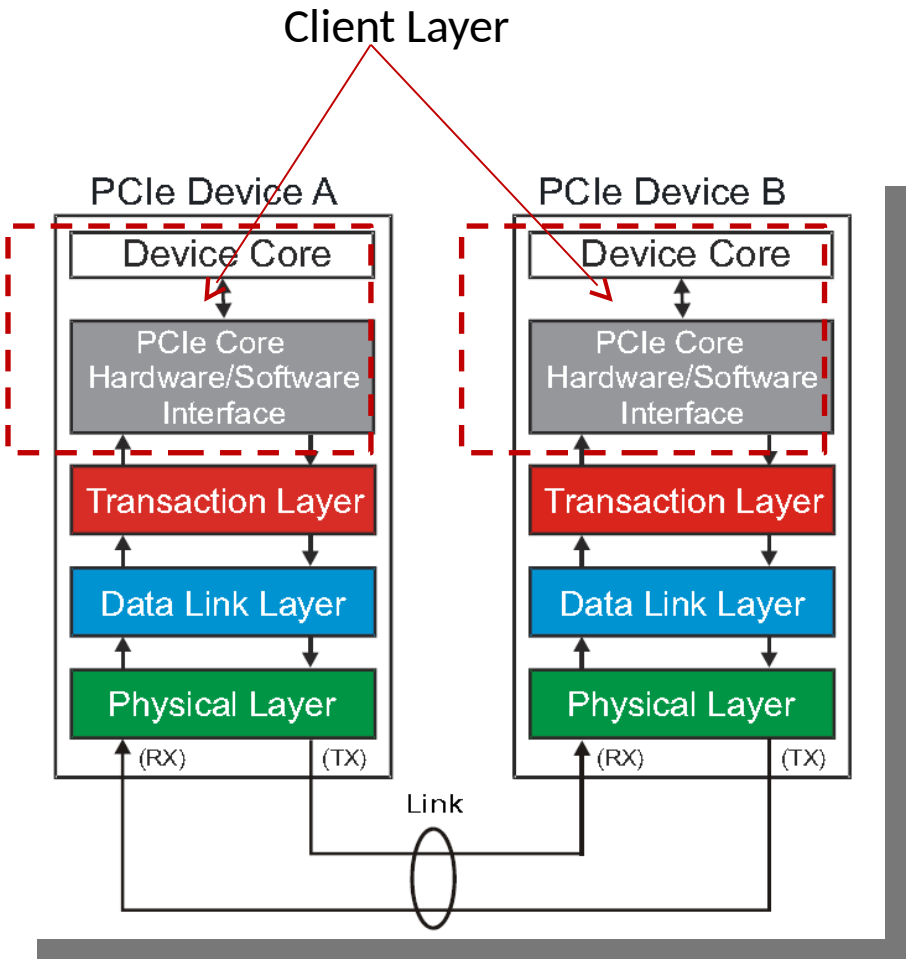
PCIe defines a layered architecture as illustration. The layers can be considered as being logically split into two parts that operate independently because they each have a transmit side for outbound traffic and a receive side for inbound traffic. The layered approach has some advantages for hardware designers because, if the logic is partitioned carefully, it can be easier to migrate to new versions of the spec by changing one layer of an existing design while leaving the others unaffected.



PCIe Layering Overview

- Client Layer

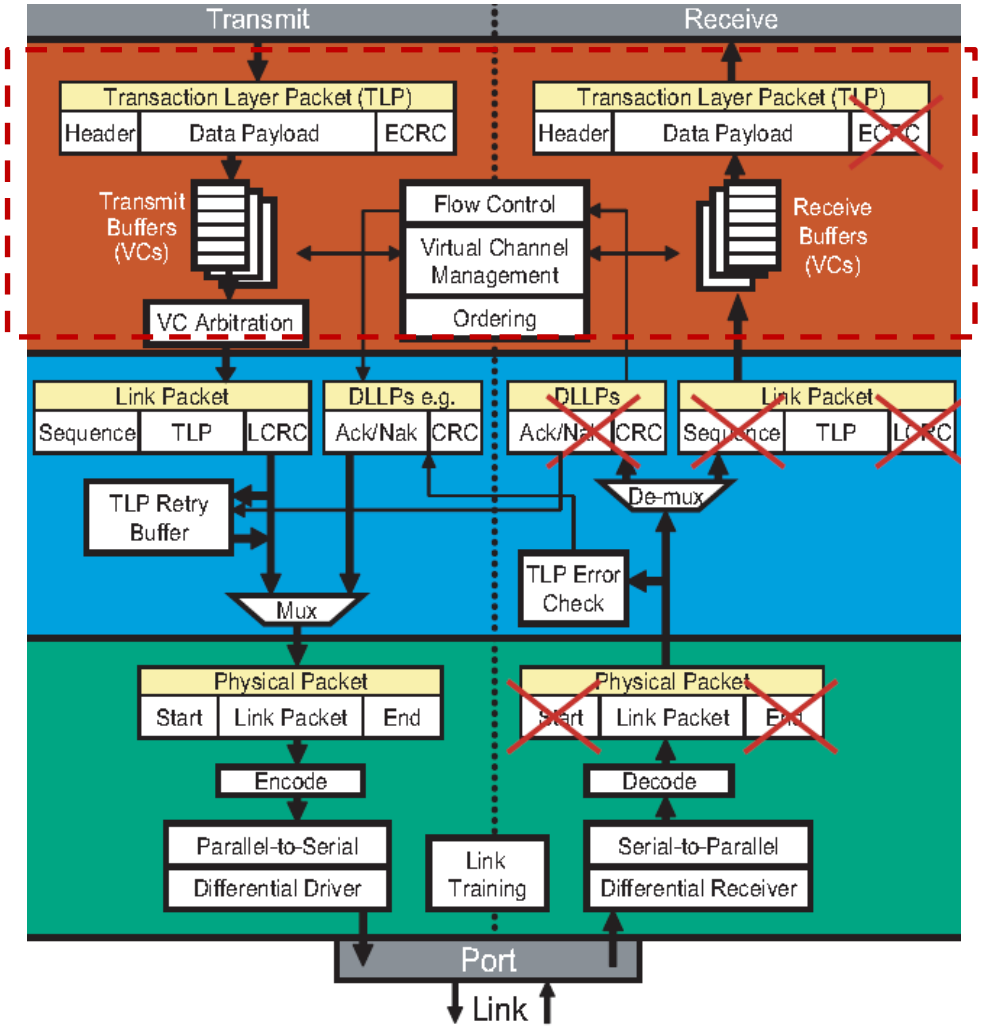
Device core and interface to Transaction Layer. The core implements the main functionality of the device. If the device is an endpoint, it may consist of up to 8 functions, each function implementing its own configuration pace. If the device is a switch, the switch core consists of packet routing logic and an internal bus for accomplishing this goal. If the device is a root, the root core implements a virtual PCI bus 0 on which resides all the chipset embedded endpoints and virtual bridges



PCIe Layering Overview

Transaction Layer

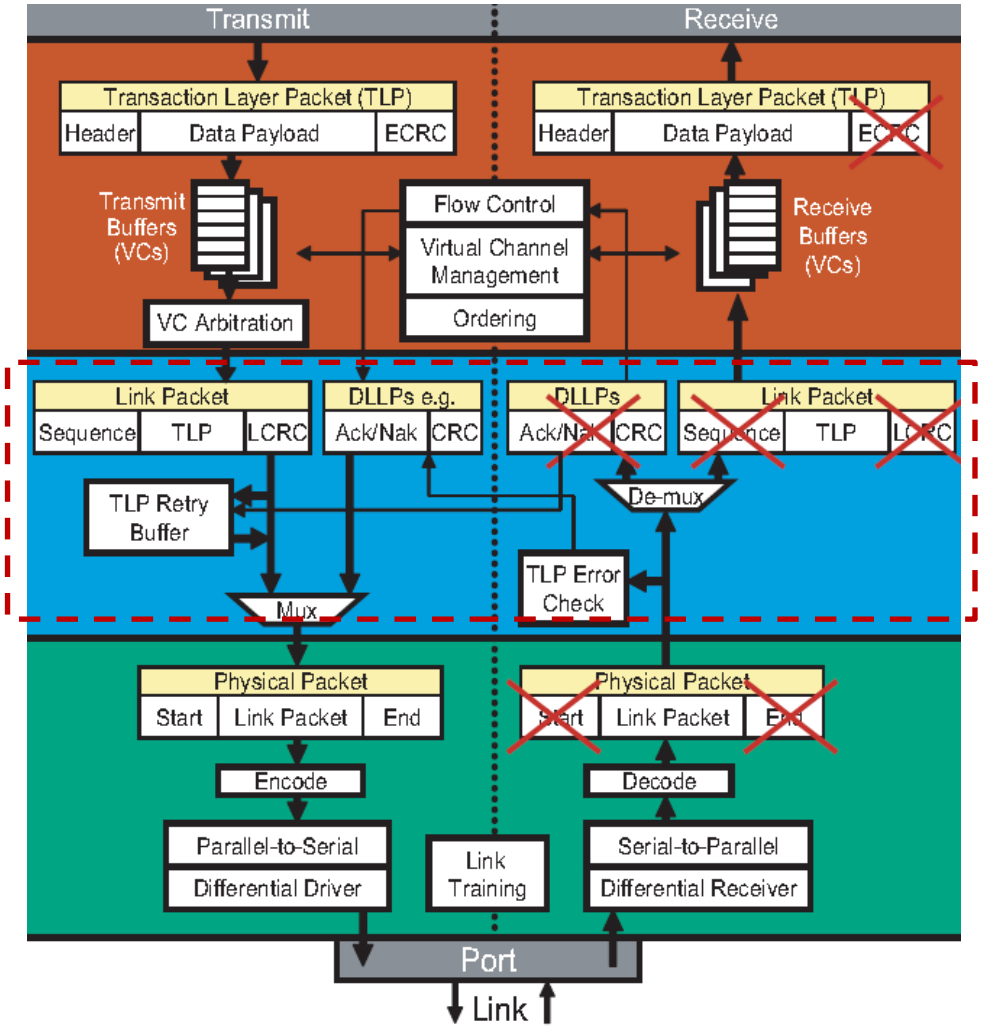
This layer is responsible for Transaction Layer Packet(TLP) creation on the transmit side and TLP decoding on the receive side. This layer is also responsible for Quality of Service functionality, Flow Control functionality and Transaction Ordering functionality



PCIe Layering Overview

Data Link Layer

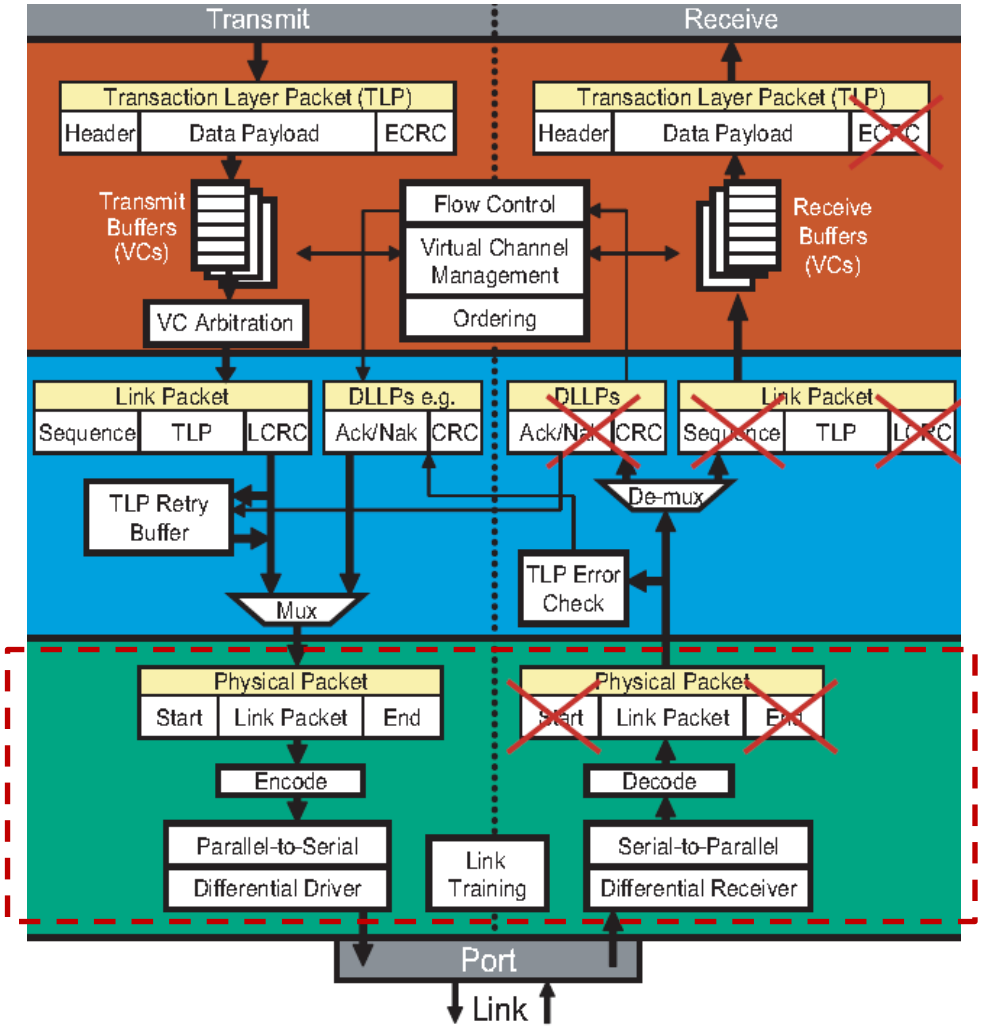
This layer is responsible for Data Link Layer Packet (DLLP) creation on the transmit side and decoding on the receive side. This layer is also responsible for Link error detection and correction. This Data Link Layer function is referred to as the Ack/Nak protocol



PCIe Layering Overview

- Physical Layer

The Physical Layer includes all circuitry for interface operation, including driver and input buffers, parallel-to-serial and serial-to-parallel conversion, PLL(s),CDR, Byte Strip, elasticity buffer, 8b/10b encoder and decode, 128b/130b encoder/decoder, scrambler and impedance matching circuitry, etc.



Agenda

- Introduction
- Layering Overview
- **Transaction Basics**
- Configuration Overview
- Address Space and TLP Routing
- IP Integration
- IP Verification
- IP Application

Transaction Basics

Transaction Request

A Transaction is defined as the combination of a Request packet that a delivers a command to a targeted device, together with any Completion packets the target sends back in reply.

Request categories

Address Space	Types	Basic usage	N/P types
Memory	Write	Transfer data to/from a memory mapped location	Posted
	Read		Non-Posted
	Read Lock		Non-Posted
IO	Write	Transfer data to/from an I/O-mapped location	Non-Posted
	Read		Non-Posted
Configuration (Type 0 and Type 1)	Write	Device Function configuration/setup	Non-Posted
	Read		Non-Posted
Message	Write	From event signaling mechanism to general purpose messaging	Posted

NOTE： The first three of these were already supported in PCI and PCI-X, but messages are a new type for PCIe.

Transaction Basics

Transaction Request

◆Non-Posted

A Requester sends a packet for which a Completer should generate a response in the form of a Completion packet.

◆Posted

The targeted device does not return a completion TLP to the Requester.

Improve performance, likelihood of a failure is small and the performance gain is significant.

But do still participate in the Ack/Nak protocol in the Data Link Layer that ensures reliable packet delivery.

In summary, non-posted transactions require a completion. Posted transactions do not require, and should never receive, a completion.

Address Space	Types	N/P types
Memory	Write	Posted
	Read	Non-Posted
	Read Lock	Non-Posted
IO	Write	Non-Posted
	Read	Non-Posted
Configuration	Write	Non-Posted
	Read	Non-Posted
Message	Write	Posted

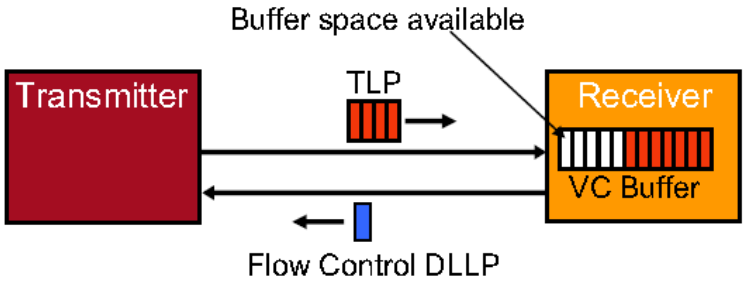
Transaction Basics

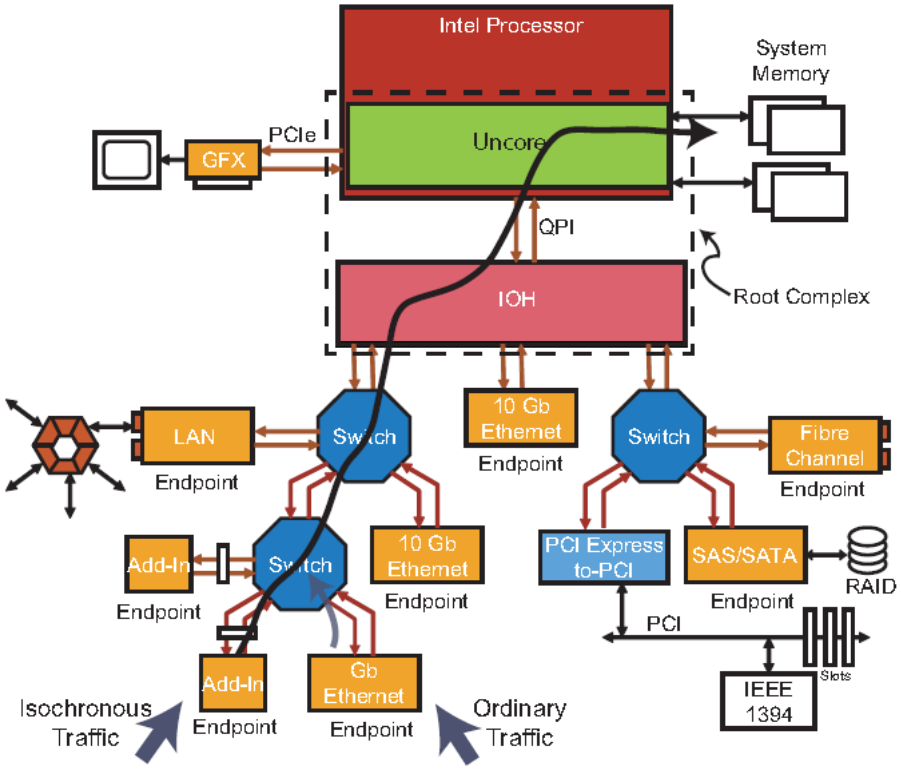
TLP Packet Types

TLP Packet Types	Abbreviated Name
Memory Read Request	MRd
Memory Read Request - Locked access	MRdLk
Memory Write Request	MWr
IO Read	IORd
IO Write	IOWr
Configuration Read (Type 0 and Type 1)	CfgRd0,CfgRd1
Configuration Write (Type 0 and Type 1)	CfgWr0,CfgWr1
Message Request without Data	Msg
Message Request with Data	MsgD
Completion without Data	Cpl
Completion with Data	CplD
Completion without Data - associated with Locked Memory Read Requests	CplLk
Completion with Data - associated with Locked Memory Read Requests	CplDLk

Transaction Basics

- Transaction basics
  - **Quality of Service (QoS)**
  - **Flow Control**
  - **Transaction Ordering**



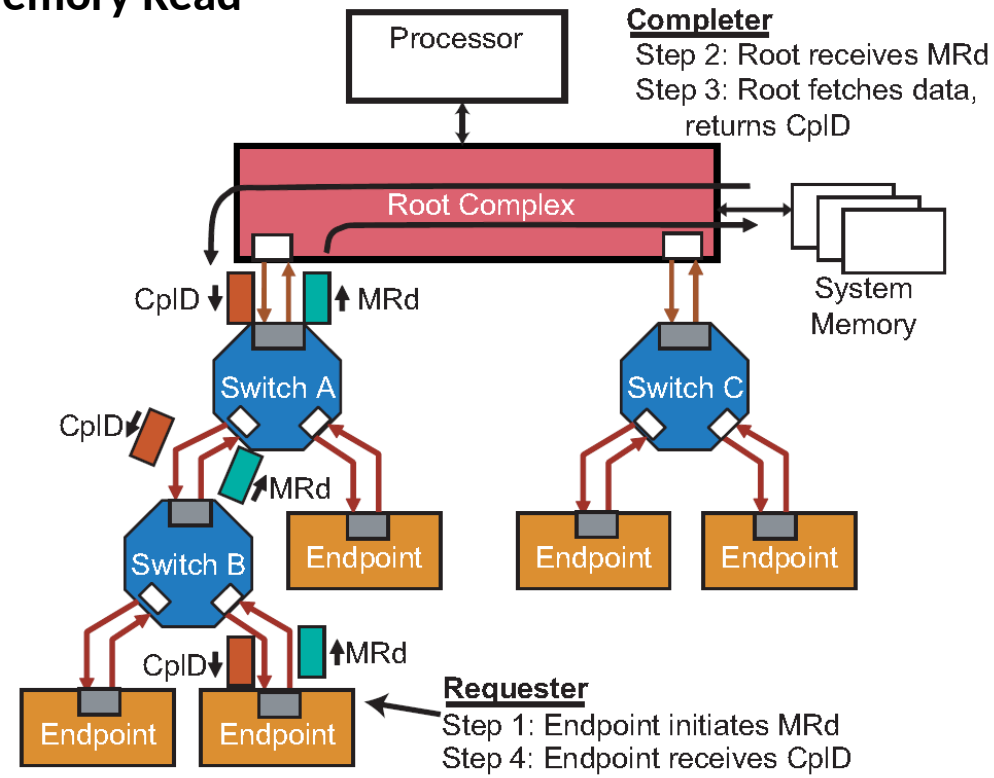


QoS

Transaction Basics

Transaction Example

Non-Posted Memory Read

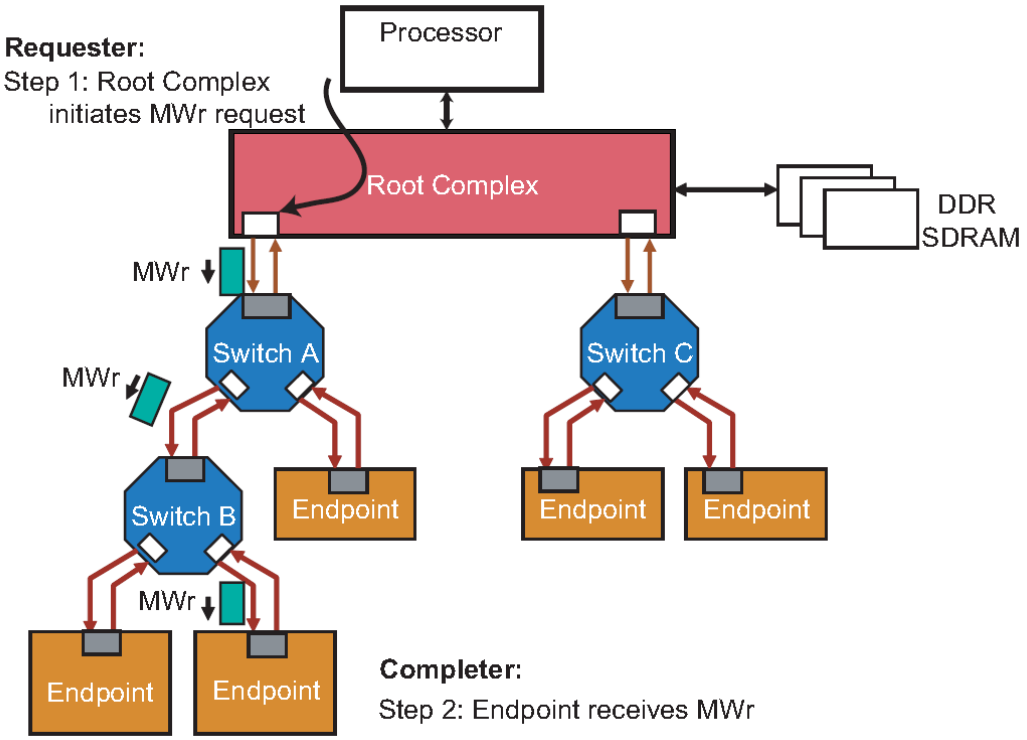


Non-Posted Memory Read(Ordinary)

Transaction Basics

Transaction Example

- **Memory Posted Writes**
- **Message Posted Writes**

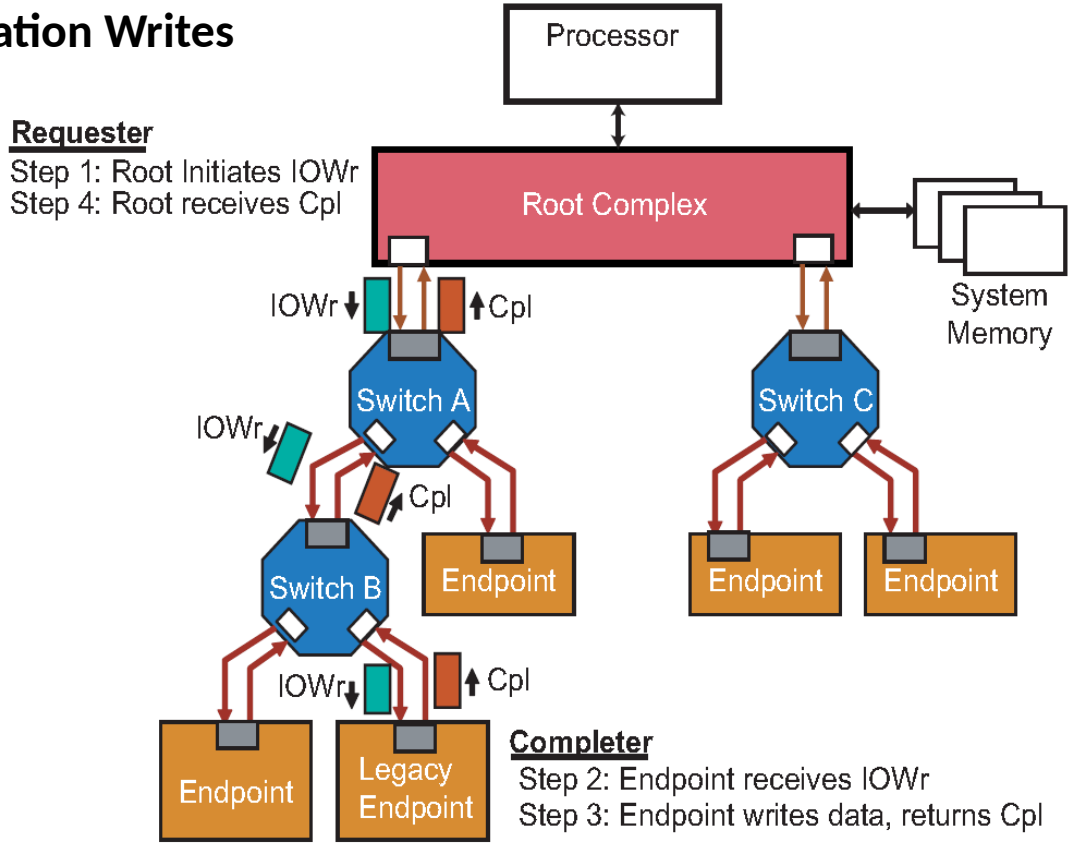


Memory Posted Writes

Transaction Basics

Transaction Example

IO and Configuration Writes



IO Writes

Agenda

- Introduction
- Layering Overview
- Transaction Basics
- **Configuration Overview**
- Address Space and TLP Routing
- IP Integration
- IP Verification
- IP Application

Configuration Overview

BDF Definition

◆PCIe Buses

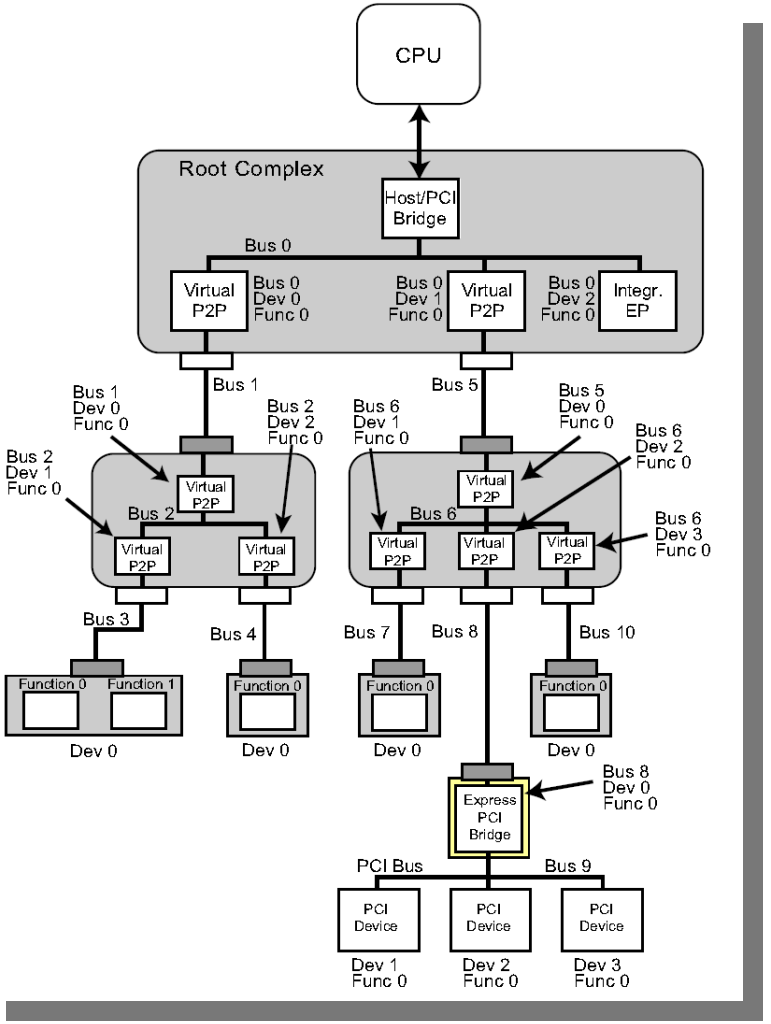
Up to 256 Bus, Bus 0, is typically assigned by hardware to the Root Complex

◆PCIe Devices

up to 32 device attachments on a single PCI bus.

◆PCIe Functions

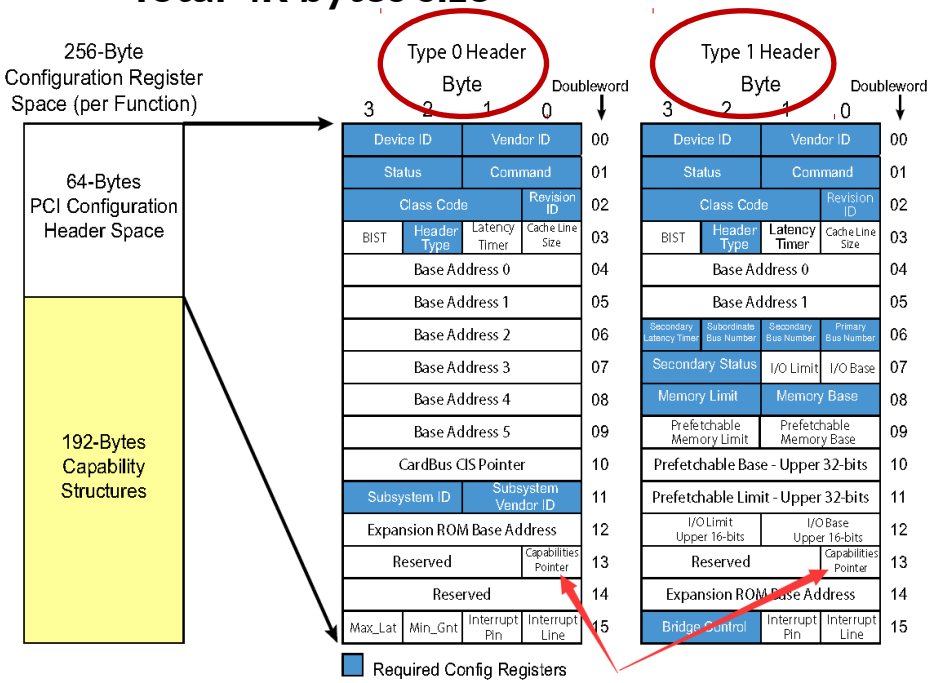
Each Function also has its own configuration address space that is used to setup the resources associated with the Function, up to 8 functions each PCIe Device

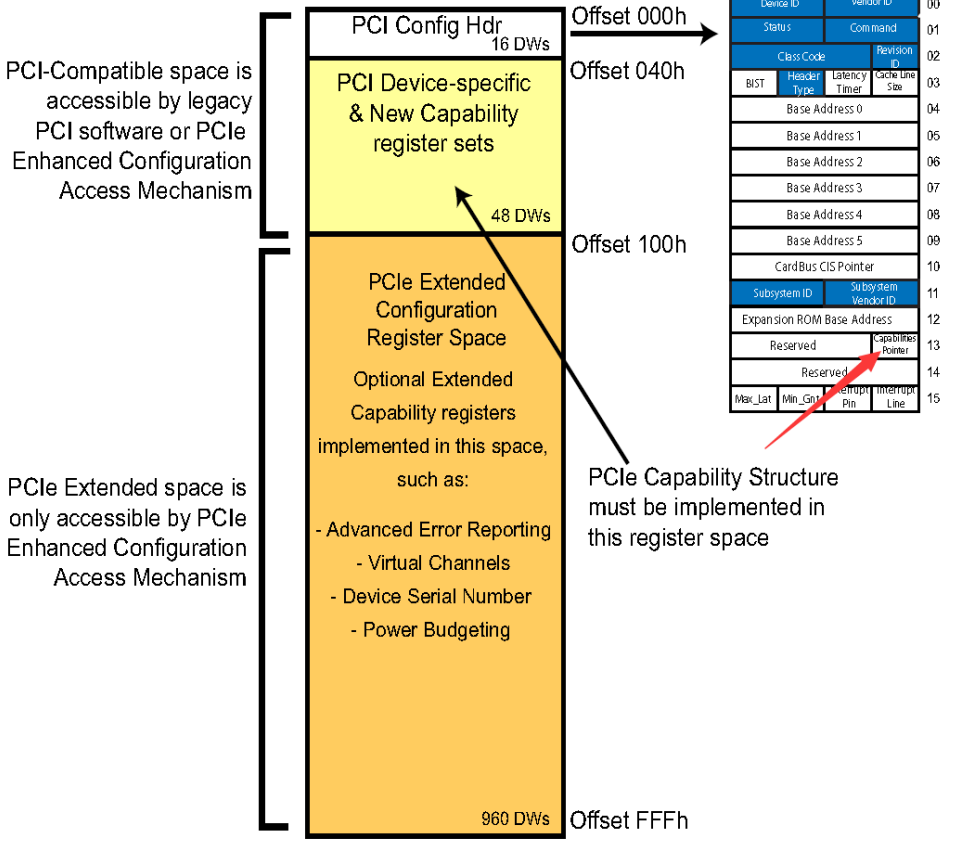


Configuration Overview

Configuration Address Space

- **PCI-Compatible Space**
- **Extended Configuration Space**
- **Total 4K bytes size**





PCI Compatible Configuration Register Space **4KB Configuration Space per Function**

Configuration Overview

Generating Configuration Transactions

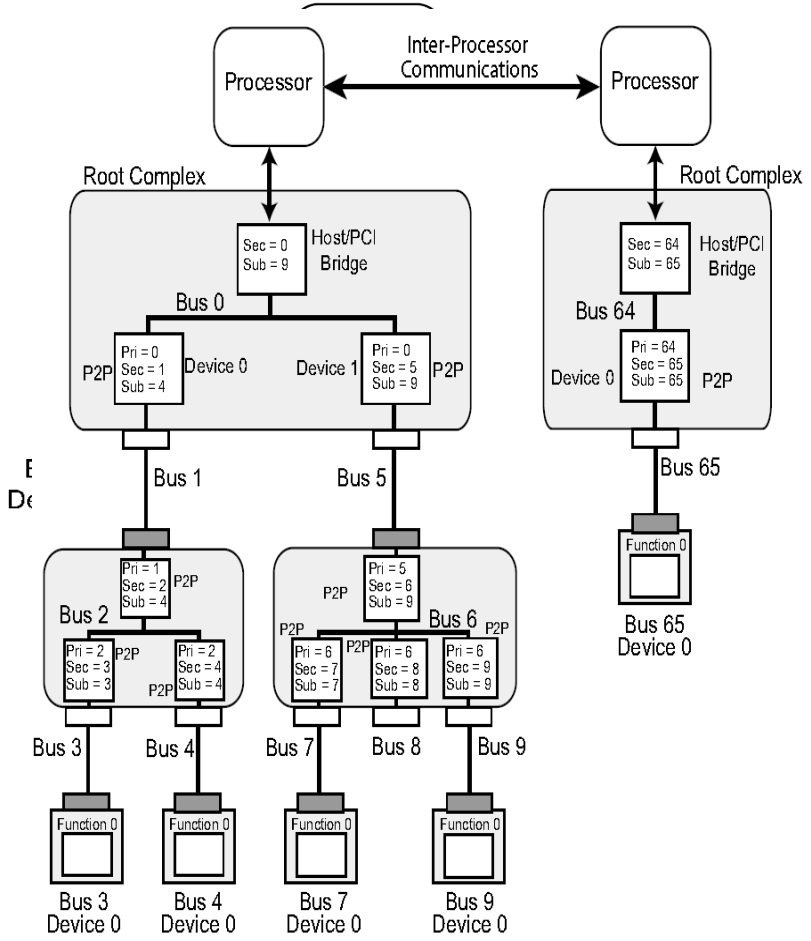
◆**The legacy PCI configuration mechanism, using IO-indirect accesses Extended Configuration Space**

① Configuration Address Port

② Bus Compare and Data Port Usage

◆**The enhanced configuration mechanism, using memory-mapped accesses(ECAM)**

Memory Address	Description
A[27:20]	Target Bus Number (0 - 255).
A[19:15]	Target Device Number (0 - 31).
A[14:12]	Target Function Number (0 - 7).
A[11:2]	A[11:2] this range can address one of 1024 dwords, whereas the legacy method is limited to only address one of 64 dwords.
A[1:0]	Defines the access size and the Byte Enable setting.



Configuration Overview

Configuration Requests

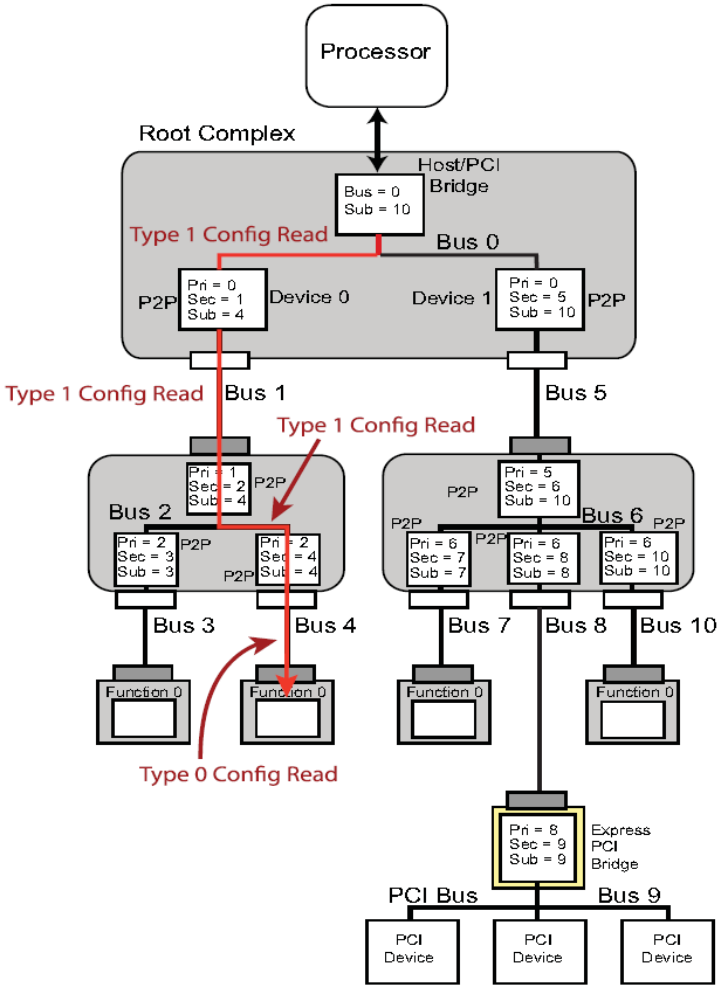
◆Type 0 Configuration Request

If the target bus number matches the Secondary Bus Number, a Type 0 configuration read or write is forwarded to the secondary bus

◆Type 1 Configuration Request

When a bridge sees a configuration access whose target bus number does not match its Secondary Bus Number but is in the range between its Secondary and Subordinate Bus Numbers, it forwards the packet as a Type 1 Request to its Secondary Bus

NOTE: Type is part of request Header



Configuration Overview

Enumeration - Discovering the Topology

◆**Device not Present**

Even though this timeout or UR result would be seen as an error during runtime, it's an expected result that isn't considered an error during enumeration

◆**Device not Ready**

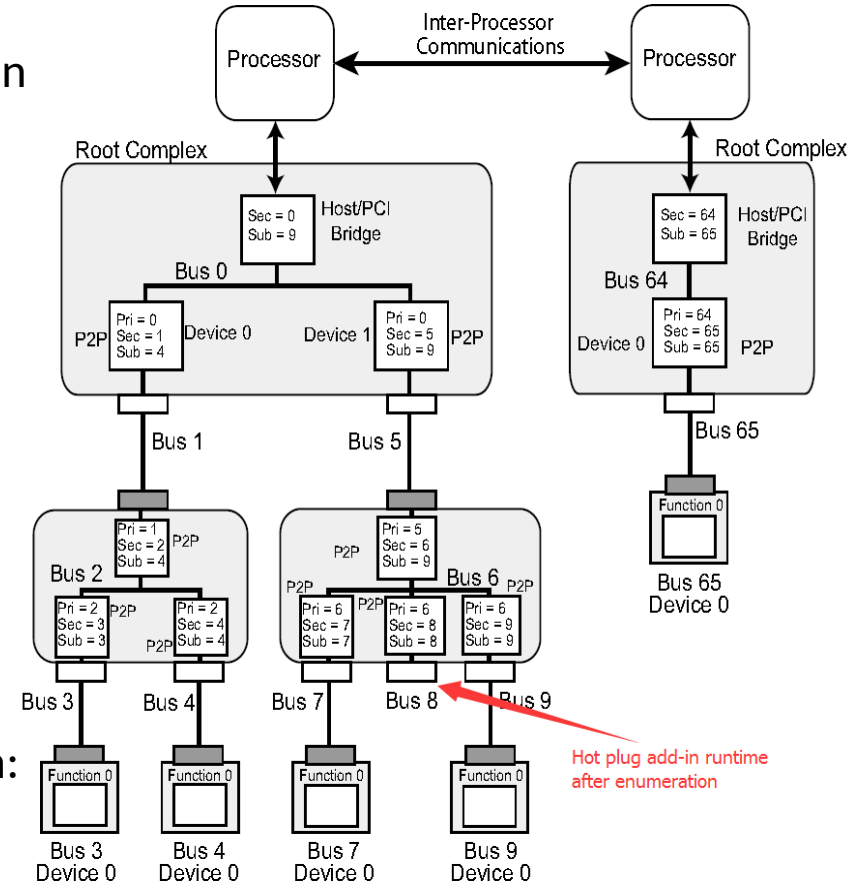
Request Retry Status (CRS) completion . CRS Only vaild one second after reset

◆**Determining if a Function is an Endpoint or Bridge**

The lower 7 bits of the Header Type register (offset 0Eh in config space header) identify the basic category of the Function

◆**Hot-Plug Considerations**

There is a simpler solution to this potential problem: simply leave a bus number gap whenever an unpopulated slot is found



Agenda

- Introduction
- Layering Overview
- Transaction Basics
- Configuration Overview
- **Address Space and TLP Routing**
- IP Integration
- IP Verification
- IP Application

Address Space and TLP Routing

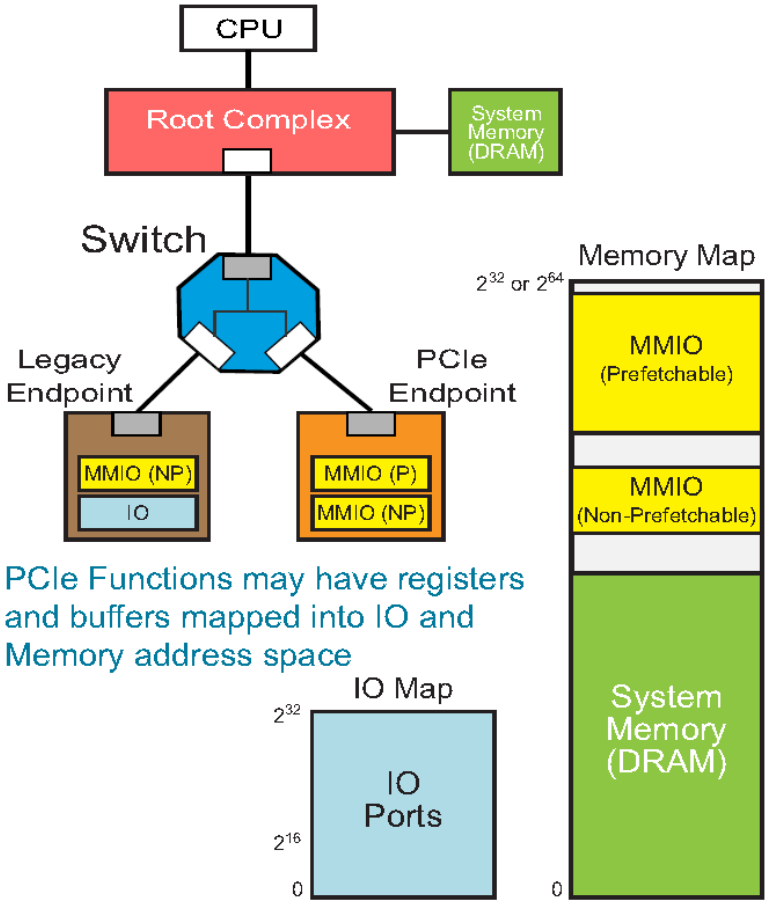
Address Space

- **Configuration**
- **Memory**

  Prefetchable vs. Non-prefetchable Memory Space
- **IO**

Prefetchable space has two very well defined attributes:

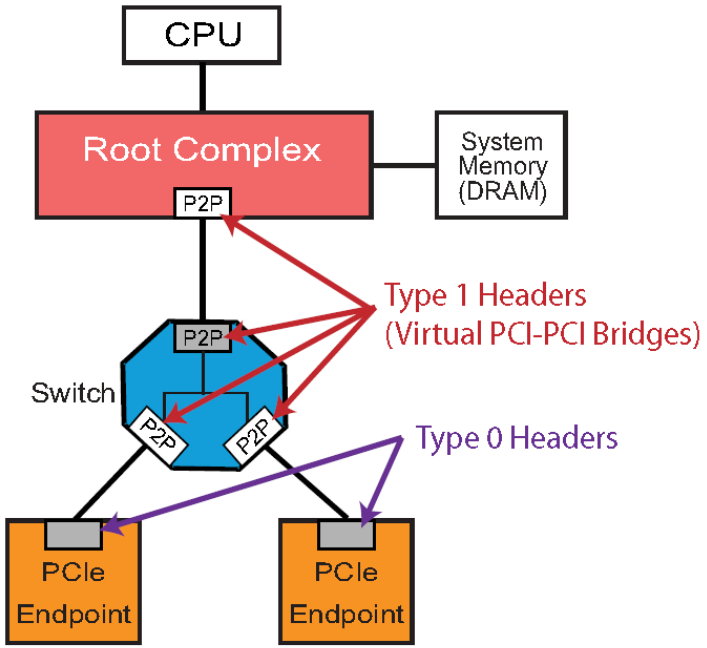
- Reads do not have side effects
- Write merging is allowed



Address Space and TLP Routing

Base Address Registers (BARs)

- **System software must first determine the size and type of address space being requested by a device**
- **Resizable BAR Capability**



Type 0 Header

31 23 15 7 0				
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Rev ID	08h
BIST	Header Type	Latency Timer	Cache Line Size	0Ch
Base Address 0 (BAR0)				10h
Base Address 1 (BAR1)				14h
Base Address 2 (BAR2)				18h
Base Address 3 (BAR3)				1Ch
Base Address 4 (BAR4)				20h
Base Address 5 (BAR5)				24h
CardBus CIS Pointer				28h
Subsystem Device ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			Capability Pointer	34h
Reserved				38h
Max Lat	Min Gnt	Interrupt Pin	Interrupt Line	3Ch

Type 1 Header

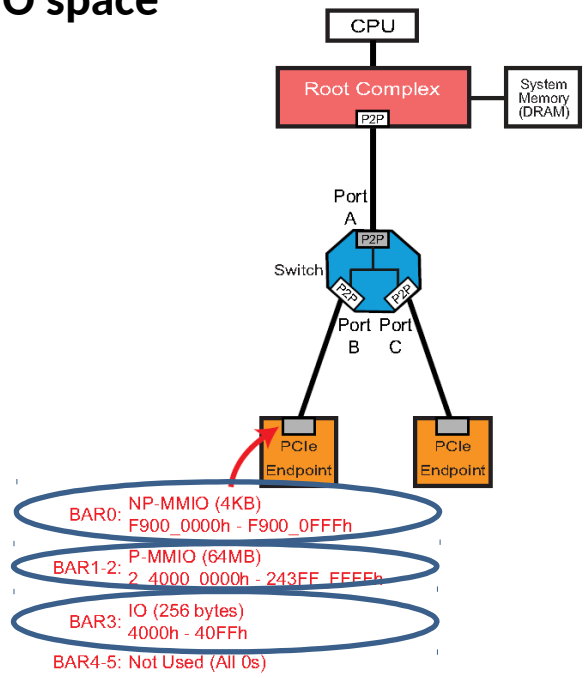
31 23 15 7 0				
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Rev ID	08h
BIST	Header Type	Latency Timer	Cache Line Size	0Ch
Base Address 0 (BAR0)				10h
Base Address 1 (BAR1)				14h
Secondary Lat Timer	Subordinate Bus #	Secondary Bus #	Primary Bus #	18h
Secondary Status		IO Limit	IO Base	1Ch
(Non-Prefetchable) Memory Limit		(Non-Prefetchable) Memory Base		20h
Prefetchable Memory Limit		Prefetchable Memory Base		24h
Prefetchable Memory Base Upper 32 Bits				28h
Prefetchable Memory Limit Upper 32 Bits				2Ch
IO Limit Upper 16 Bits		IO Base Upper 16 Bits		30h
Reserved			Capability Pointer	34h
Expansion ROM Base Address				38h
Bridge Control		Interrupt Pin	Interrupt Line	3Ch

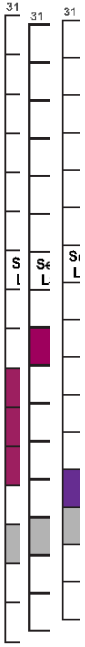
Address Space and TLP Routing

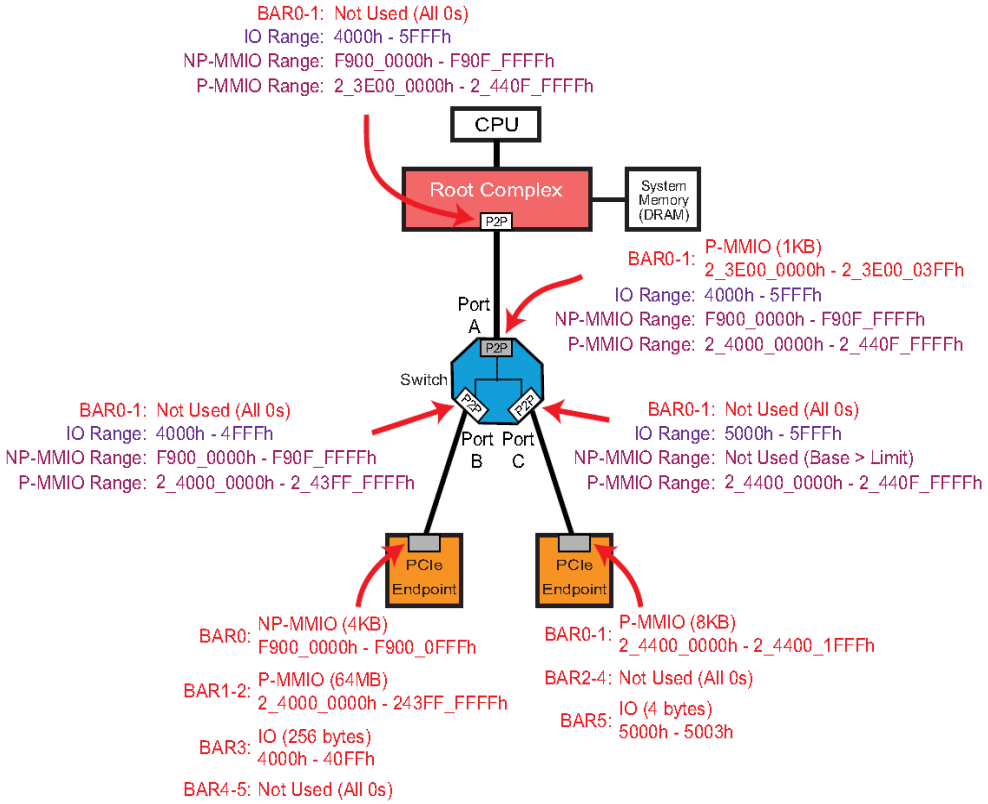
Base and Limit Registers

There are the three sets of Base and Limit registers found in each Type 1 header:

- **Prefetchable Memory space**
- **Non-Prefetchable Memory space**
- **IO space**







Address Space and TLP Routing

TLP Routing Basics

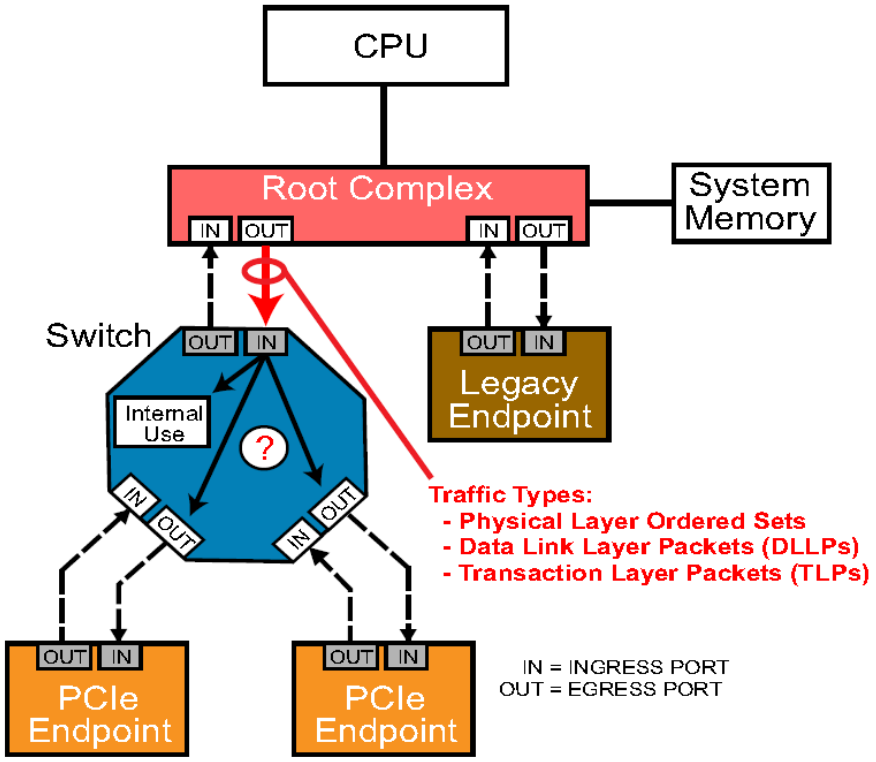
- **Three traffic types**
- **Three decisions**

Accept internally 、 Forward 、 reject

- **Three Methods of TLP Routing**

Address Routing, ID Routing, Implicit routing

TLP Type	Routing Method
MR/MW/Atomic	Address Routing
IOR/IOW	Address Routing
CR/CW	ID Routing
Message, Message With Data	Support ALL Routing
Completion, Completion With Data	ID Routing

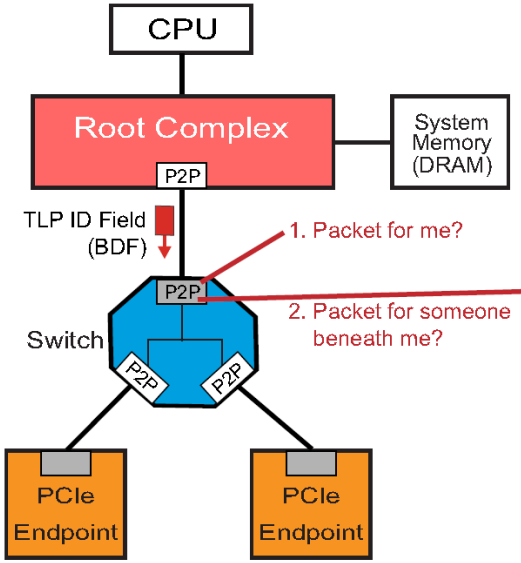


Address Space and TLP Routing

ID Routing

ID routing is used to target the logical position - Bus Number, Device Number, Function Number (typically referred to as BDF), of a Function within the topology

TLP Type	Routing Method
MR/MW/Atomic	Address Routing
IOR/IOW	Address Routing
CR/CW	ID Routing
Message, Message With Data	Support ALL Routing
Completion, Completion With Data	ID Routing



Type 1 Header

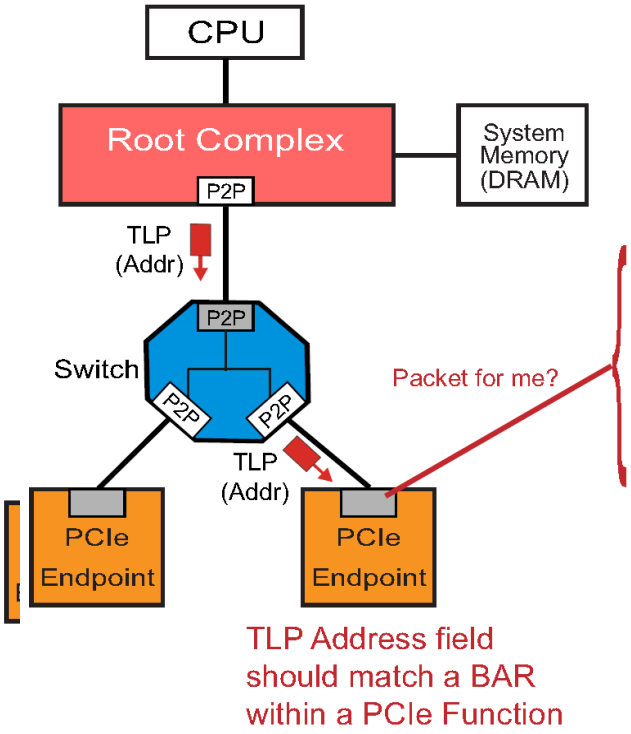
31	23	15	7	0	
Device ID		Vendor ID			00h
Status		Command			04h
Class Code				Rev ID	08h
BIST	Header Type	Latency Timer		Cache Line Size	0Ch
Base Address 0 (BAR0)					10h
Base Address 1 (BAR1)					14h
Secondary Lat Timer	Subordinate Bus #	Secondary Bus #		Primary Bus #	18h
Secondary Status		IO Limit		IO Base	1Ch
(Non-Prefetchable) Memory Limit		(Non-Prefetchable) Memory Base			20h
Prefetchable Memory Limit		Prefetchable Memory Base			24h
Prefetchable Memory Base Upper 32 Bits					28h
Prefetchable Memory Limit Upper 32 Bits					2Ch
IO Limit Upper 16 Bits		IO Base Upper 16 Bits			30h
Reserved				Capability Pointer	34h
Expansion ROM Base Address					38h
Bridge Control		Interrupt Pin		Interrupt Line	3Ch

Address Space and TLP Routing

Address Routing

TLPs that use address routing refer to the same memory (system memory and memory-mapped IO) and IO address maps that PCI and PCI-X transactions do

TLP Type	Routing Method
MR/MW/Atomic	Address Routing
IOR/IOW	Address Routing
CR/CW	ID Routing
Message, Message With Data	ALL Routing
Completion, Completion With Data	ID Routing



Type 0 Header

31	23	15	7	0	
Device ID			Vendor ID		00h
Status			Command		04h
Class Code				Rev ID	08h
BIST	Header Type	Latency Timer		Cache Line Size	0Ch
Base Address 0 (BAR0)					10h
Base Address 1 (BAR1)					14h
Base Address 2 (BAR2)					18h
Base Address 3 (BAR3)					1Ch
Base Address 4 (BAR4)					20h
Base Address 5 (BAR5)					24h
CardBus CIS Pointer					28h
Subsystem Device ID			Subsystem Vendor ID		2Ch
Expansion ROM Base Address					30h
Reserved				Capability Pointer	34h
Reserved					38h
Max Lat	Min Gnt	Interrupt Pin		Interrupt Line	3Ch

Address Space and TLP Routing

Implict Routing

Implicit routing, used in some message packets, is based on the awareness of routing elements that the topology has upstream and downstream directions and a single Root Complex at the top.

Message:

- INTx Interrupt Signaling
- Power Management
- Error Signaling
- Locked Transaction Support
- Slot Power Limit Support
- Vendor-Defined Messages
- Ignored Messages(Hot-plug)
- LTR Messages
- OBFF Messages

TLP Type	Routing Method
MR/MW/Atomic	Address Routing
IOR/IOW	Address Routing
CR/CW	ID Routing
Message, Message With Data	Support ALL Routing
Completion, Completion With Data	ID Routing

Agenda

- Introduction
- Layering Overview
- Transaction Basics
- Configuration Overview
- Address Space and TLP Routing
- PCIE Capability
- Linux Software Stack(RC)

PCIE Capability

PCI Express CAP

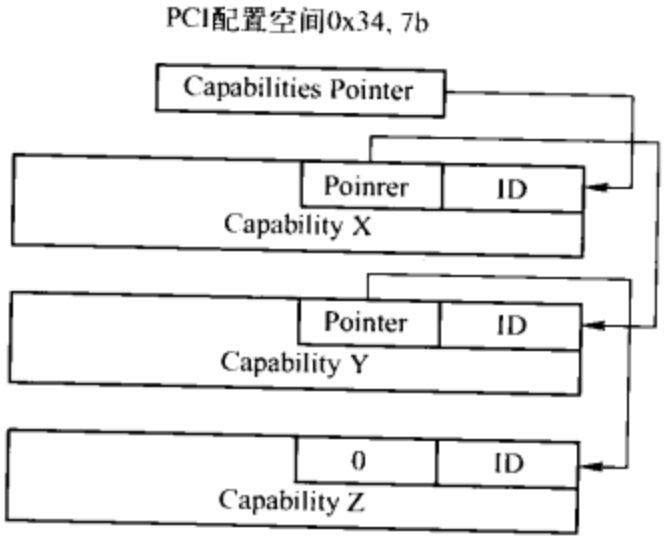


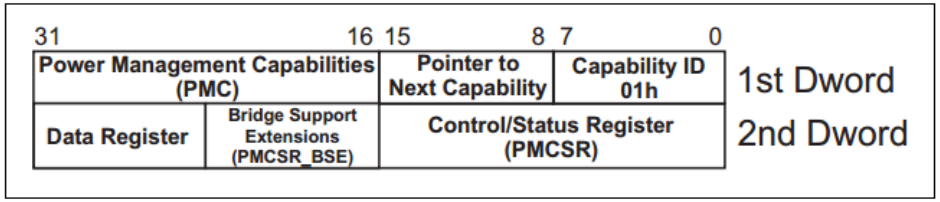
图 4-14　PCIe 总线 Capability 结构的组成

- PCIe 总线规范要求 PCIE 设备必须支持 Capability 结构。在 PCI 总线的基本配置空间中，包含一个 Capability Pointer 寄存器，寄存器存放 Capability 结构链表的头指针。在一个 PCIe 设备中，可能包含有多个 Capability 结构，这些寄存器组成一个链表。

- 每一个 Capability 结构都有唯一的 ID 号，每一个 Capability 寄存器都有一个指针，这个指针指向下一个 Capability ，组成一个单向链表结构。重点讲解下 Power Management 和 MSI/MSI-X Capability 结构。

PCIE Capability

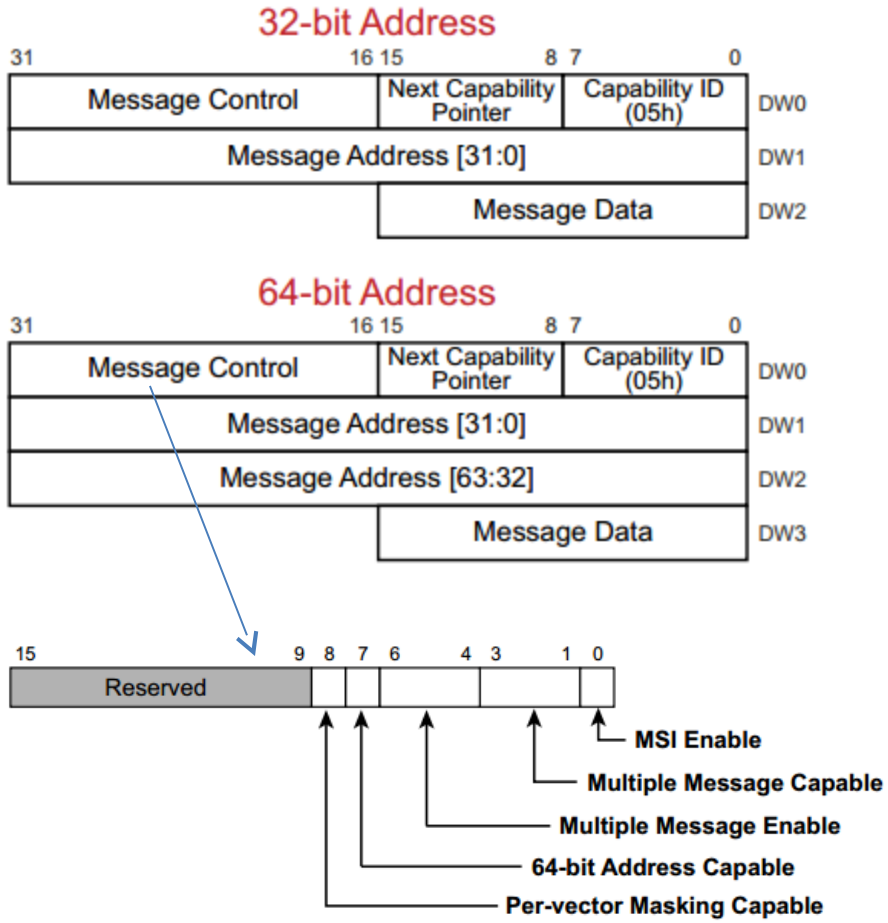
- Power Management Capability



The PCI-PM spec defines the Power Management Capability configuration registers. These registers were optional for PCI, but required for PCIe, and are located in the PCI-compatible configuration space with a Capability ID of 01h.

PCIE Capability

- MSI Capability



The MSI Capability Structure resides in the PCI-compatible config space area (first 256 bytes). There are four variations of the MSI Capability Structure based on whether it supports 64-bit addressing or only 32-bit and whether it supports per vector masking or not. Native PCIe devices are required to support 64-bit addressing.

Agenda

- Introduction
- Layering Overview
- Transaction Basics
- Configuration Overview
- Address Space and TLP Routing
- PCIE Capability
- Linux Software Stack

Linux Software Stack

- DTS

phys.hi cell: npt000ss bbbbbbbb dddddfff rrrrrrrr
n: relocatable region flag (doesn't play a role here)
p: prefetchable (cacheable) region flag
t: aliased address flag
ss: space code
    00: configuration space
    01: I/O space
    10: 32 bit memory space
    11: 64 bit memory space

```
#address-cells = <3>;
#size-cells = <2>;
ranges = < 0x82000000 0 0xfa000000 0x0 0xfa000000 0 0x600000
           0x81000000 0 0xfa600000 0x0 0xfa600000 0 0x100000 >;
```

bbbbbbbb: The PCI bus number. PCI may be structured hierarchically. So we may have PCI/PCI bridges which will define sub busses.
ddddd: The device number, typically associated with IDSEL signal connections.
fff: The function number. Used for multifunction PCI devices.
rrrrrrrr: Register number; used for configuration cycles.

Linux Software Stack

- DTS

```
#interrupt-cells = <1>;
interrupt-controller;
interrupt-map-mask = <0 0 0 7>;
interrupt-map = <0 0 0 1 &pcie0 1>,
                <0 0 0 2 &pcie0 2>,
                <0 0 0 3 &pcie0 3>,
                <0 0 0 4 &pcie0 4>;
```

In this example board, we have 1 PCI slots with 4 interrupt lines, respectively, so we have to map 4 interrupt lines to the interrupt controller. This is done using the interrupt-map property. The exact procedure for interrupt mapping is described in.

legacylegac 中断

```
        irq_set_chained_handler_and_data(irq,
                                         rockchip_pcie_legacy_int_handler,
                                         port);
```

```
static void rockchip_pcie_legacy_int_handler(struct irq_desc *desc)
{
        struct irq_chip *chip = irq_desc_get_chip(desc);
        struct rockchip_pcie_port *port;
        u32 reg;

        chained_irq_enter(chip, desc);
        port = irq_desc_get_handler_data(desc);

        reg = pcie_read(port, PCIE_CLIENT_INT_STATUS);
        reg = (reg & ROCKCHIP_PCIE_RPIFR1_INTR_MASK) >>
              ROCKCHIP_PCIE_RPIFR1_INTR_SHIFT;
        if (reg)
                generic_handle_irq(irq_find_mapping(port->irq_domain,
                                                    ffs(reg)));
        chained_irq_exit(chip, desc);
}
```

MSI 中断

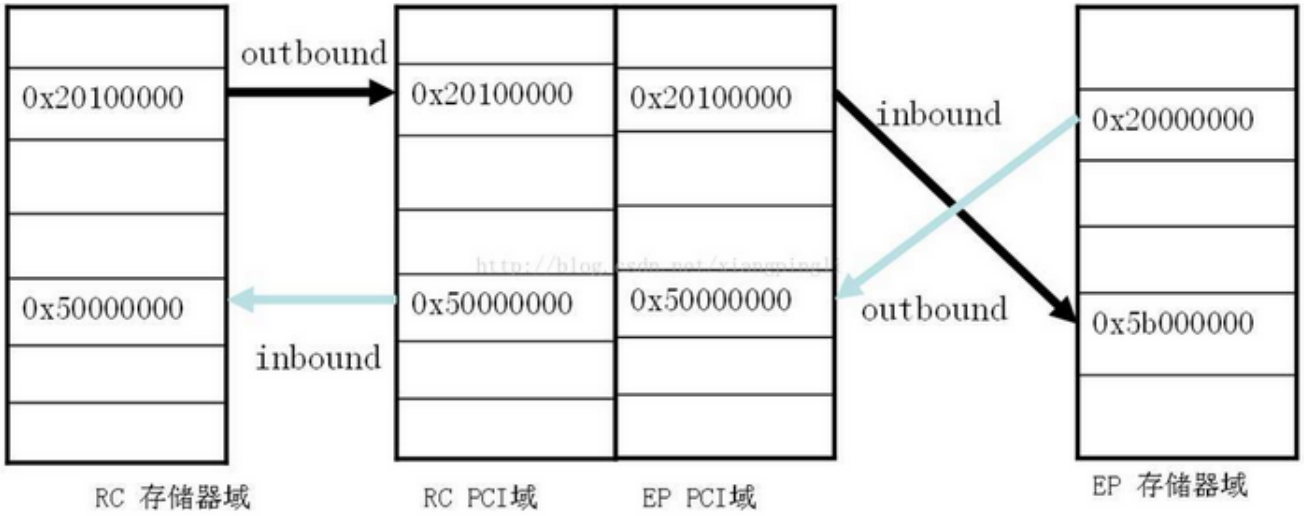
```
gic: interrupt-controller@fee00000 {
        compatible = "arm,gic-v3";
        #interrupt-cells = <3>;
        #address-cells = <2>;
        #size-cells = <2>;
        ranges;
        interrupt-controller;

        reg = <0x0 0xfee00000 0 0x10000>, /* GICD */
              <0x0 0xfef00000 0 0xc0000>, /* GICR */
              <0x0 0xfff00000 0 0x10000>, /* GICC */
              <0x0 0xfff10000 0 0x10000>, /* GICH */
              <0x0 0xfff20000 0 0x10000>; /* GICV */
        interrupts = <GIC_PPI 9 IRQ_TYPE_LEVEL_HIGH>;
        its: interrupt-controller@fee20000 {
                compatible = "arm,gic-v3-its";
                msi-controller;
                reg = <0x0 0xfee20000 0x0 0x20000>;
        };
};
```

```
msi-map = <0x0 &its 0x0 0x1000>;
```

```
- msi-map: Maps a Requester ID to an MSI controller and associated
  msi-specifier data. The property is an arbitrary number of tuples of
  (rid-base,msi-controller,msi-base,length), where:

  * rid-base is a single cell describing the first RID matched by the entry.

  * msi-controller is a single phandle to an MSI controller

  * msi-base is an msi-specifier describing the msi-specifier produced for the
    first RID matched by the entry.

  * length is a single cell describing how many consecutive RIDs are matched
    following the rid-base.

  Any RID r in the interval [rid-base, rid-base + length) is associated with
  the listed msi-controller, with the msi-specifier (r - rid-base + msi-base).

- msi-map-mask: A mask to be applied to each Requester ID prior to being mapped
  to an msi-specifier per the msi-map property.

- msi-parent: Describes the MSI parent of the root complex itself. Where
  the root complex and MSI controller do not pass sideband data with MSI
  writes, this property may be used to describe the MSI controller(s)
  used by PCI devices under the root complex, if defined as such in the
  binding for the root complex.
```

Linux Software Stack

- ATU



Thank You!