NoteBook

zhenqiufu

Contents

MyNote

-
-
-
-

by Zhenqiu

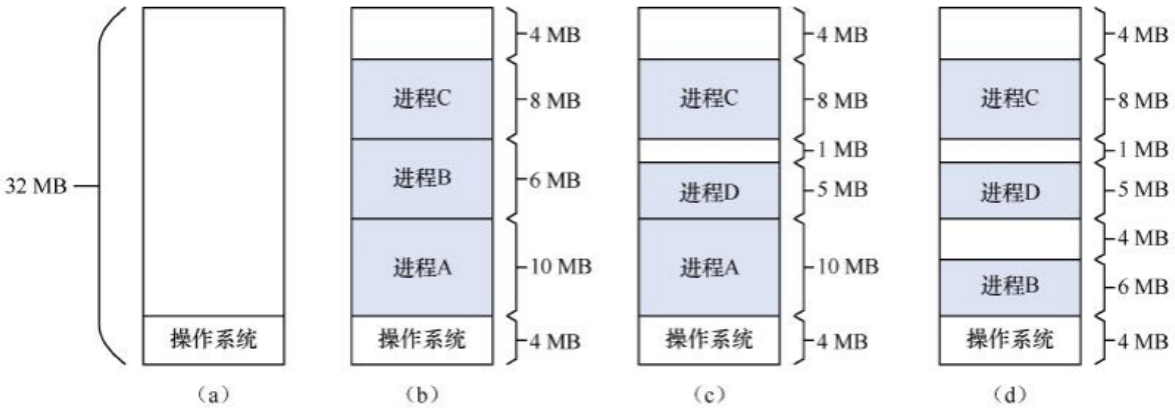
Part I

Chapter 1

1.1

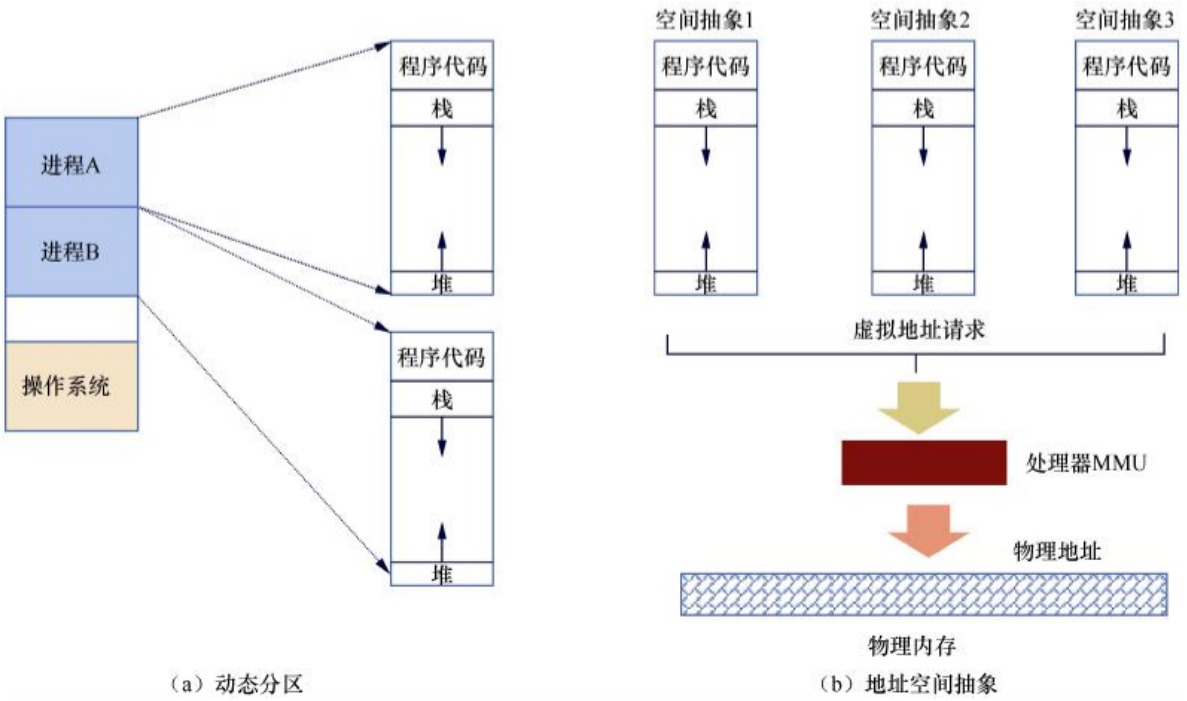
1.1.1

[toc]

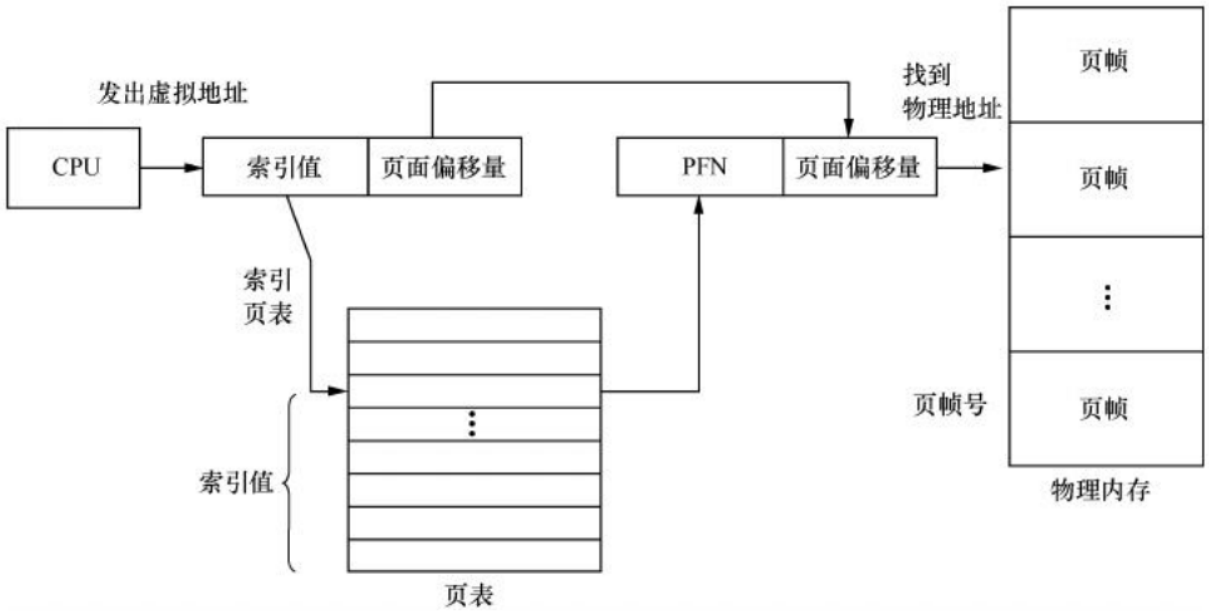


▲图 14.1　动态分区示意图

- A B
-
-



▲图 14.2　动态分区和地址空间抽象

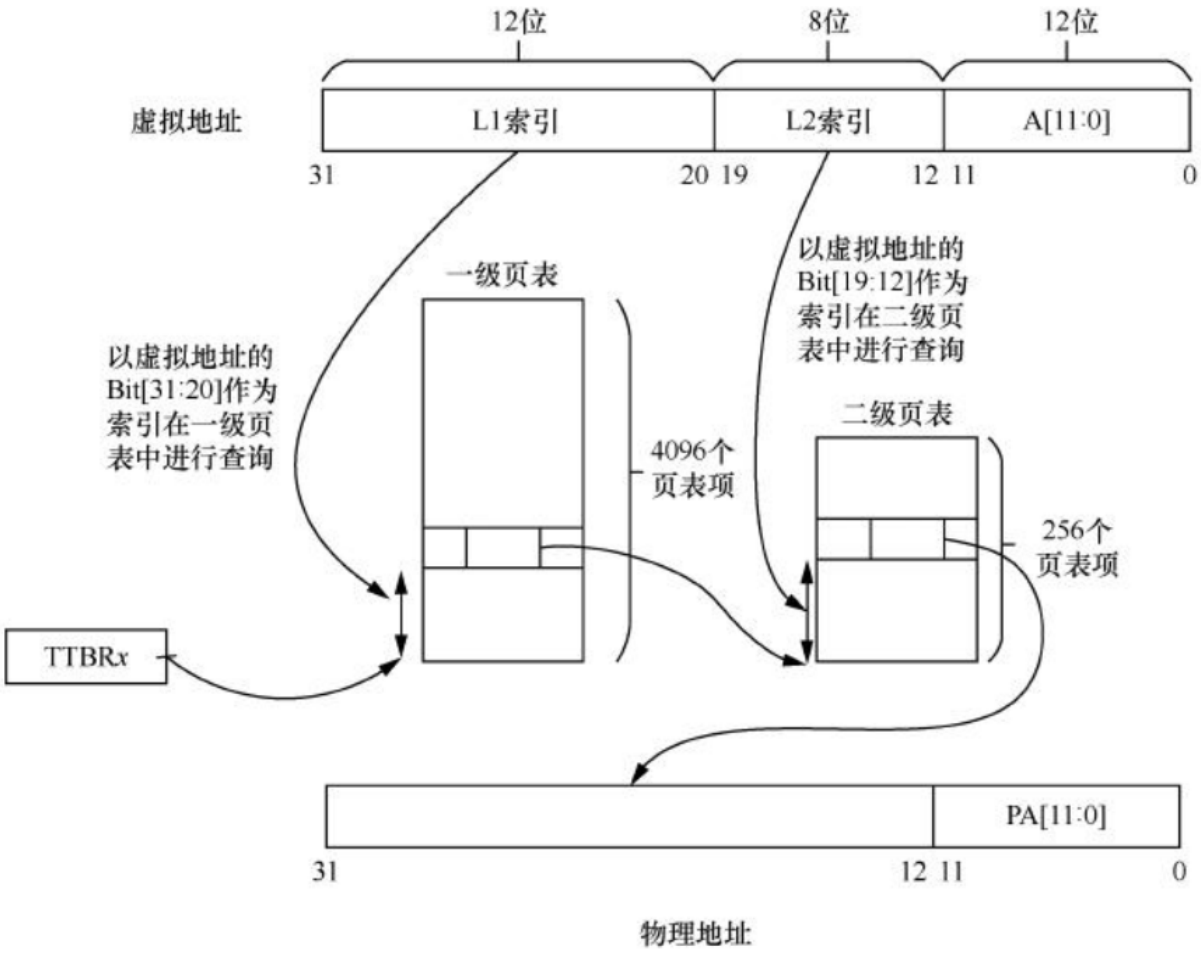


▲图 14.3　页表查询过程

4G　4K　　2^20　　4B　　4M

4M table　　table

32　　4 GB　　512 MB



▲图 14.5　二级页表查询过程

14.5　　VA[31:20]　　12　　4096　　VA[19:12]　　8　　256　　**16 KB**　　4096　　4　　16 KB　　PFN

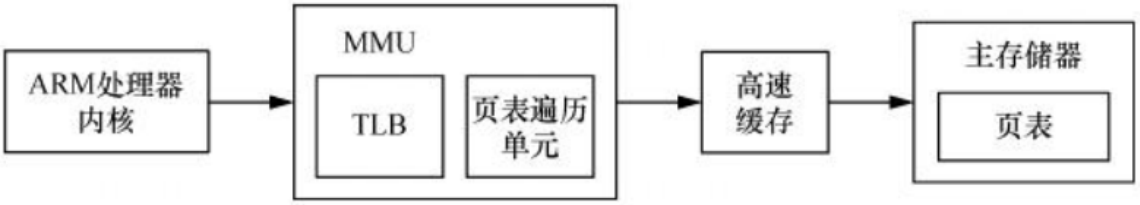
entry　　5　　KB

ARM64　　3　4

X86　5

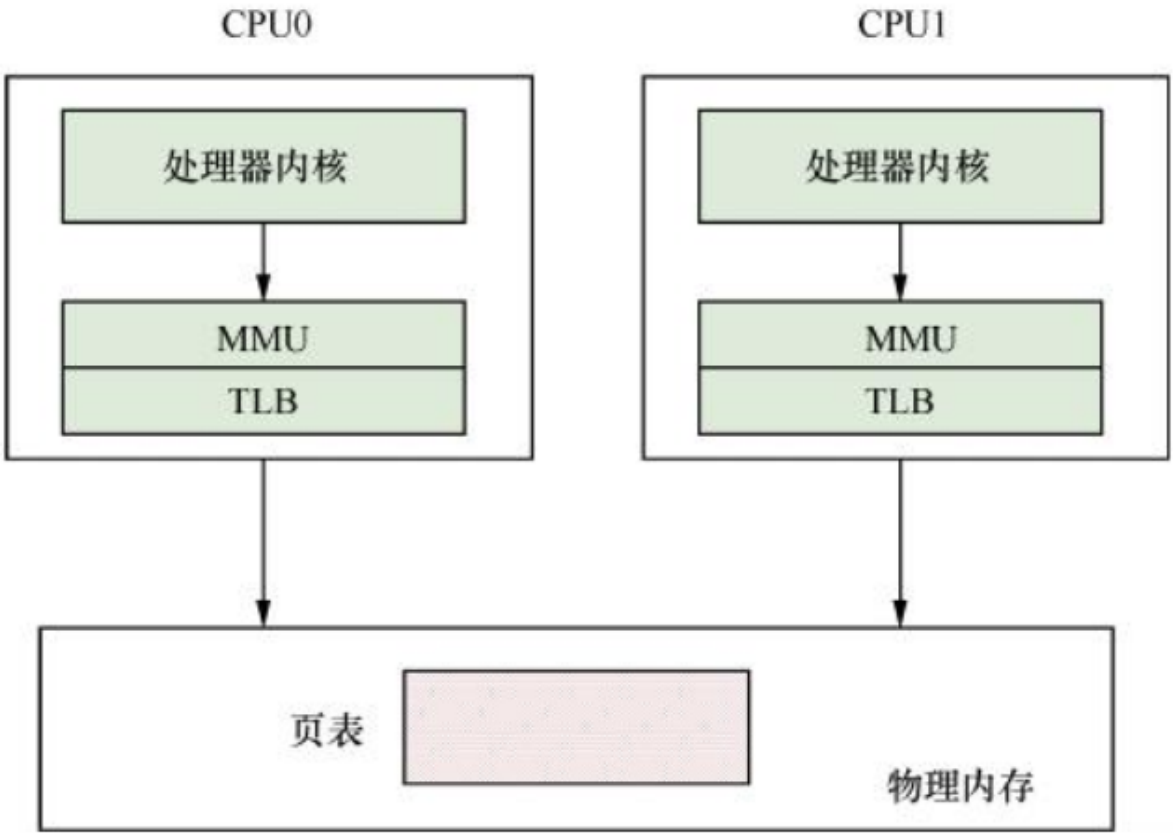
ARM64

ARM64　　MMU　TLB　　Table Walk Unit　TWU



▲图 14.8　ARM 处理器的内存管理体系结构

SMP　Symmetric Multi-Processor　　MMU　TLB



▲图 14.10　SMP 系统与 MMU

AArch64　　MMU

VA　　PA

1　　Intermediate Physical Address　IPA　　2　IPA　　PA

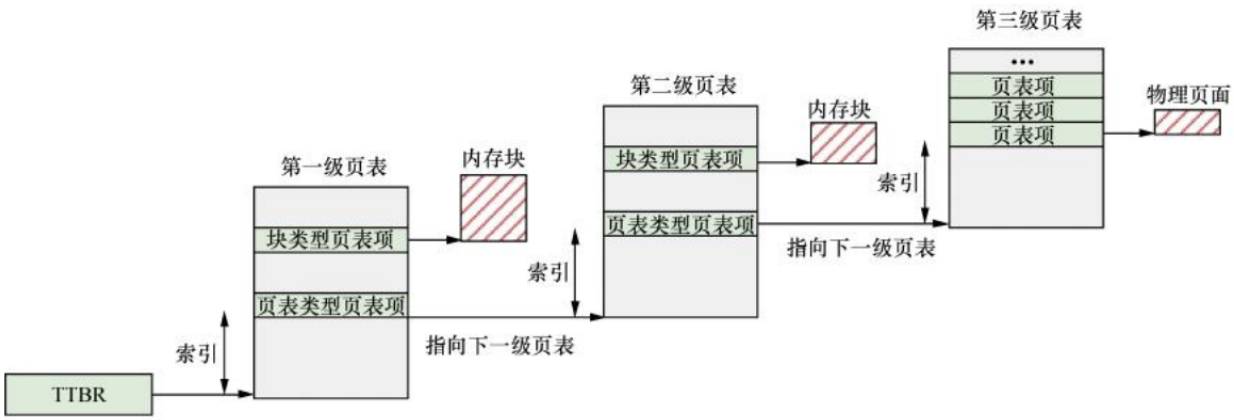
ARMv8　　4 KB　16 KB　64 KB　3

AArch64　　48　　VA　　256 TB

- 0x0000000000000000　0x0000FFFFFFFFFFFF

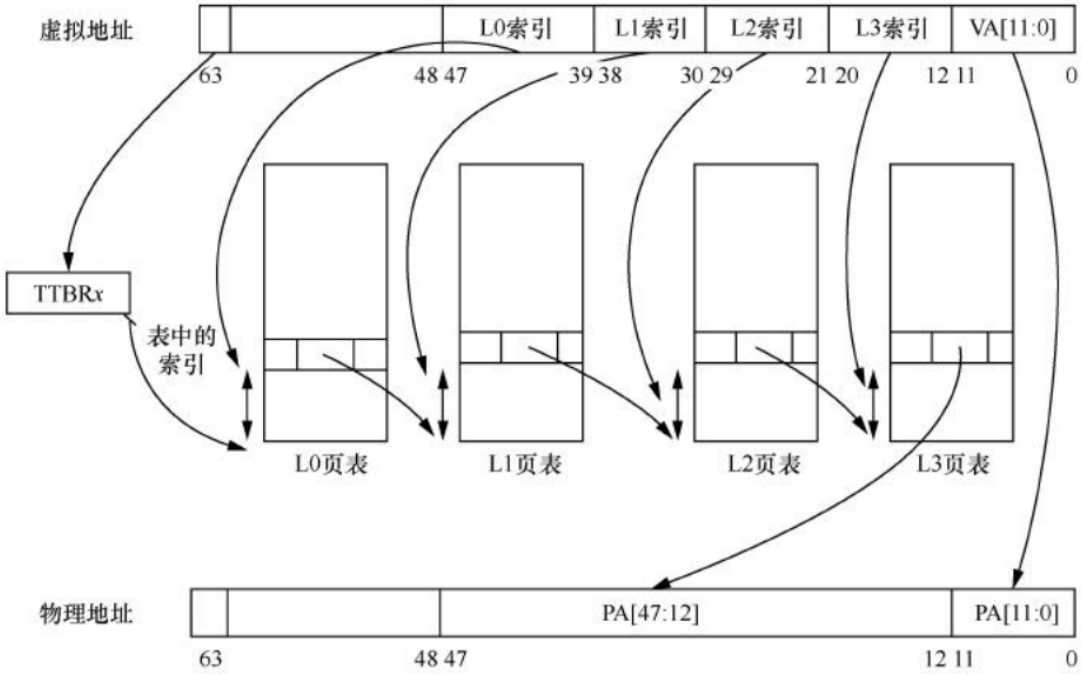
  0　　**TTBR0_ELx**

- 0xFFFF000000000000　0xFFFFFFFFFFFFFFFF　　1　　**TTBR1_ELx**



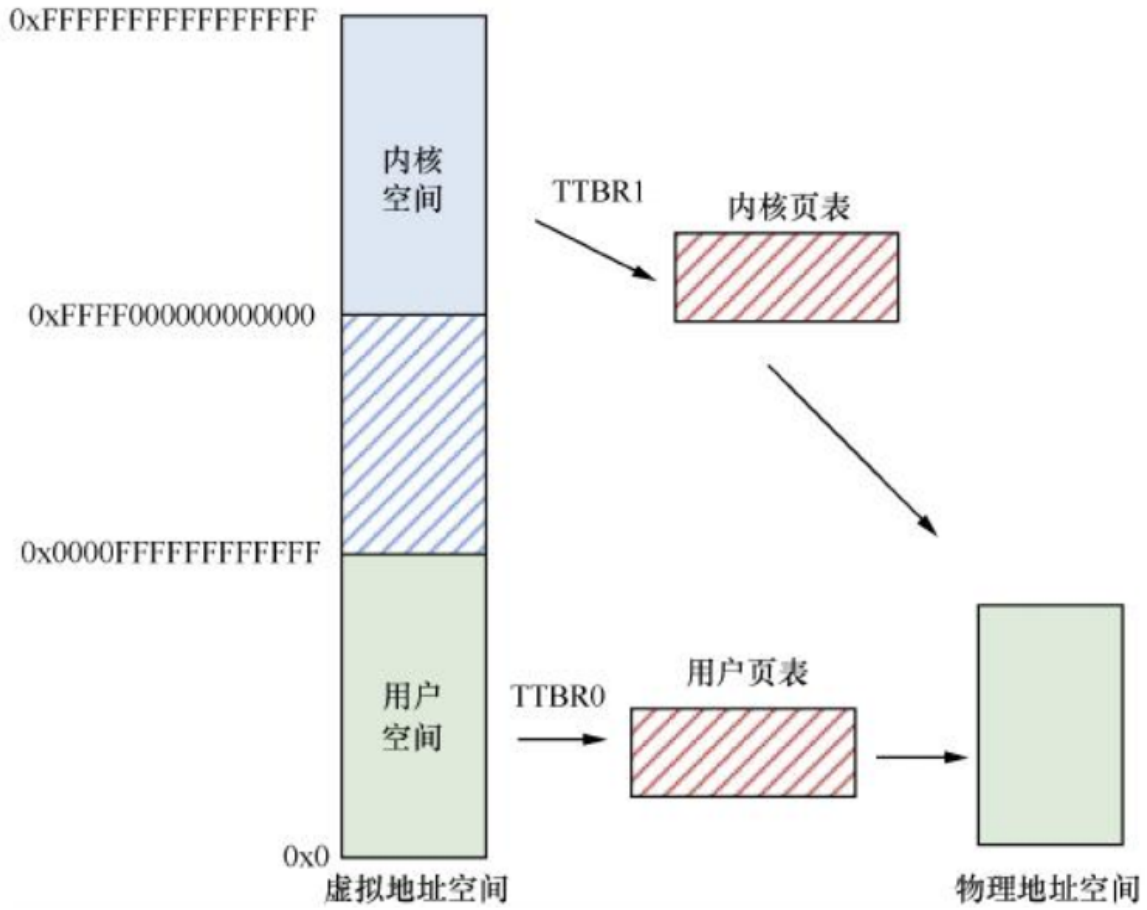
▲图 14.11　页表三级映射的示意图

48 4K



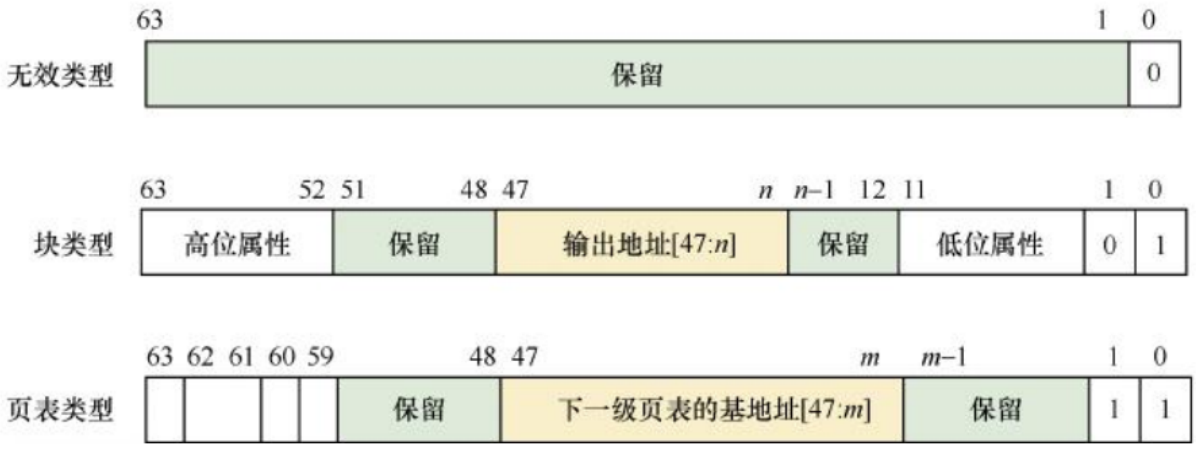
▲图 14.12 页表映射的查询过程

x86_64 AArch64 14.16 3 CPU MMU TTBR0 CPU MMU TTBR1

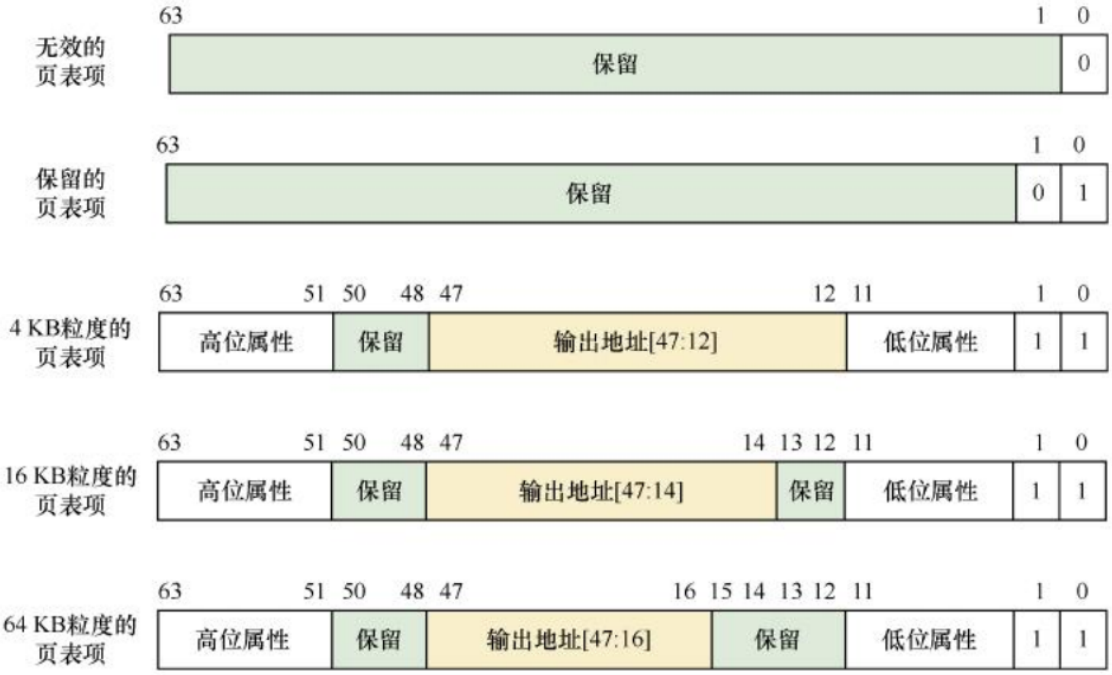


▲图 14.16　两套页表

L0-L2 Entry　AArch64　　L0 L3　　L0 L2　　14.17



▲图 14.17　L0～L2 页表项描述符

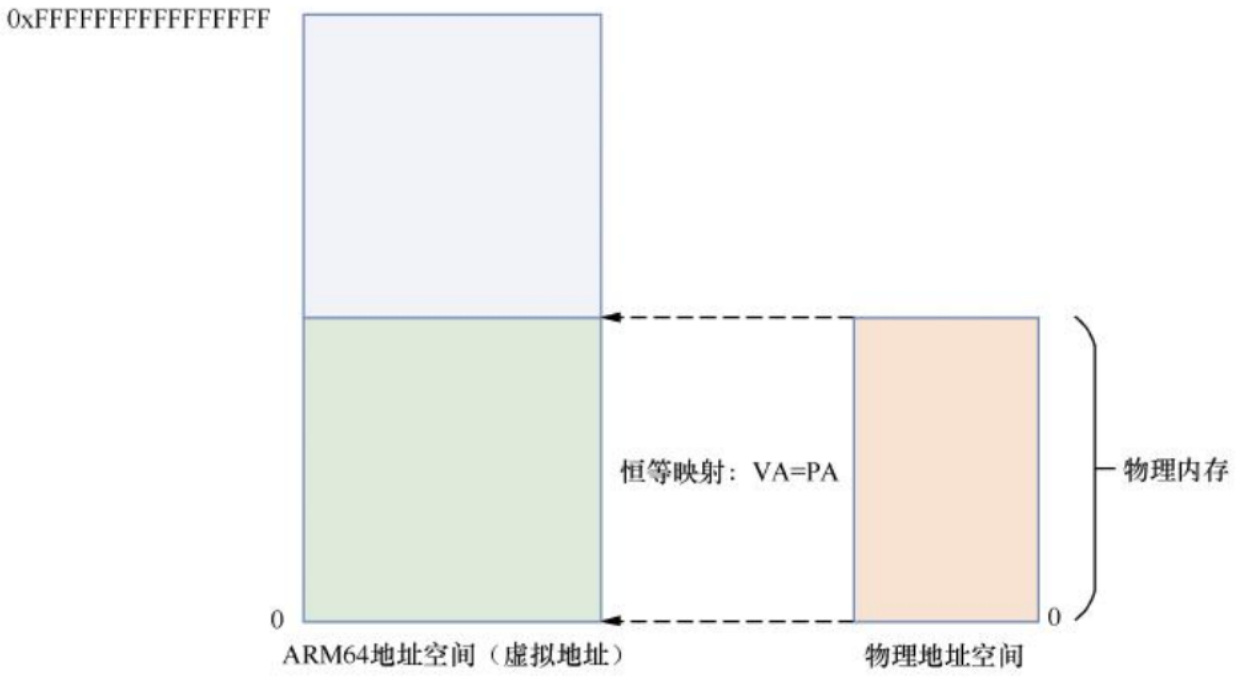


▲图 14.18　L3 页表项描述符

L3 Entry

Translation Control Register TCR System Control Register SCTLR Translation Table Base Register TTBR

VA = PA



▲图 14.24　恒等映射

1.1.2 SMMU

[toc]

SMMU

(SMMU, System Memory Management Unit) SMMU

1.
   - SMMU
   - **I/O** SMMU I/O
2.
   - SMMU
   - SMMU
3.
   - SMMU I/O
4.
   - SMMU
   - SMMU

MMU-700

ARM SMMU IO IOMMU MMU-700 IP

SMMU Spec MMU-700 TRM

MMU SMMU

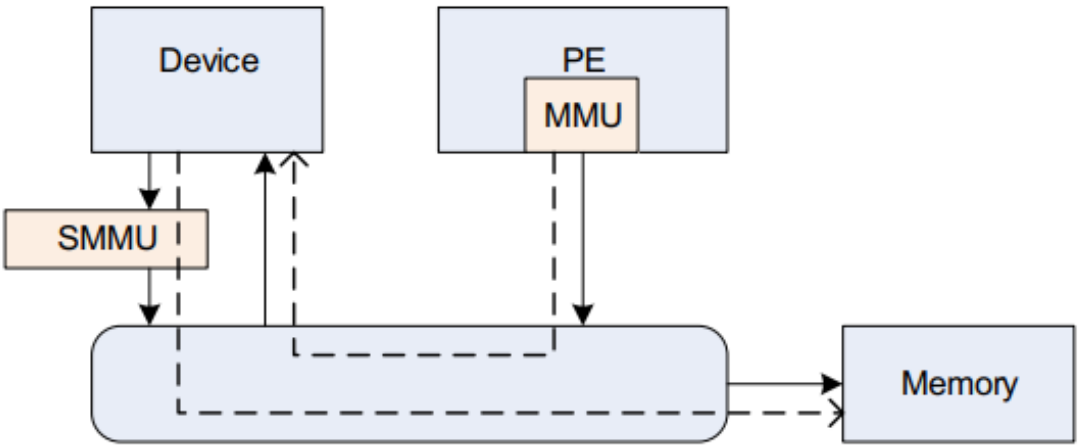


Figure 2.1: System MMU in DMA traffic

SMMU

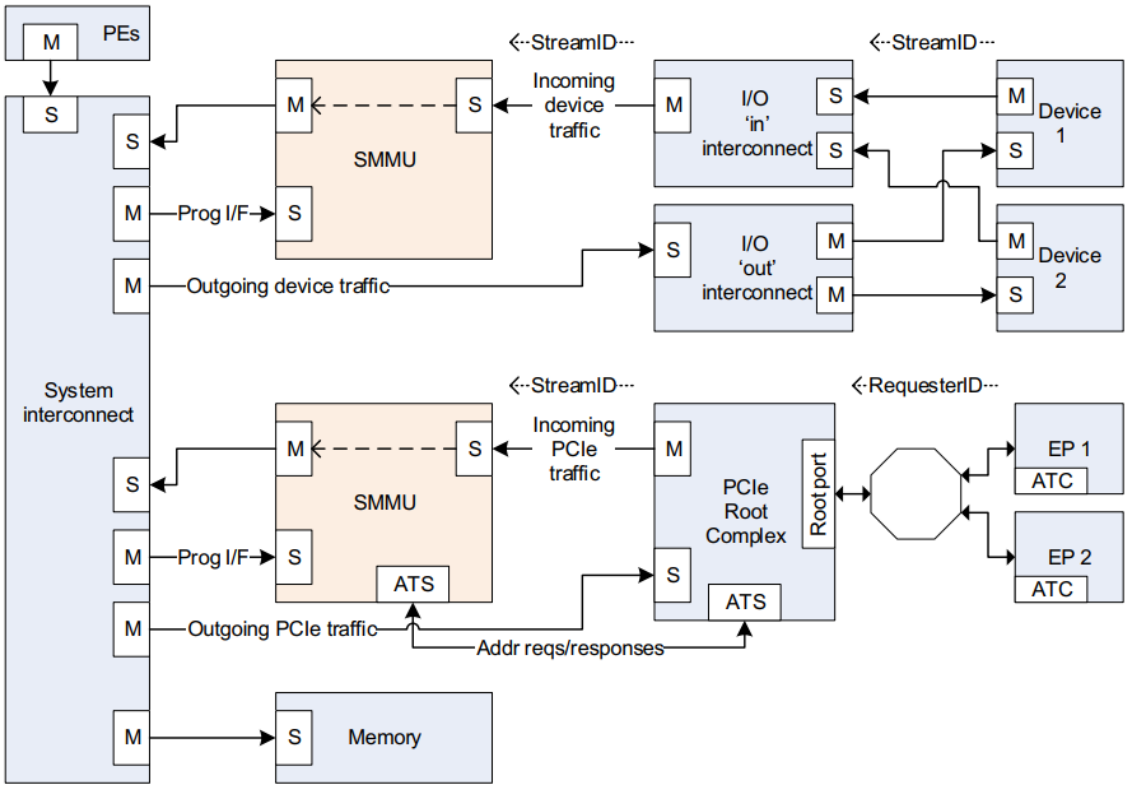


Figure 2.2: SMMU placement in an example system

1 SMMU DMA device device1 VA smmu VA PA

2 SMMU PCIe Root Complex (which itself hosts a network of endpoints) PCIE ATS

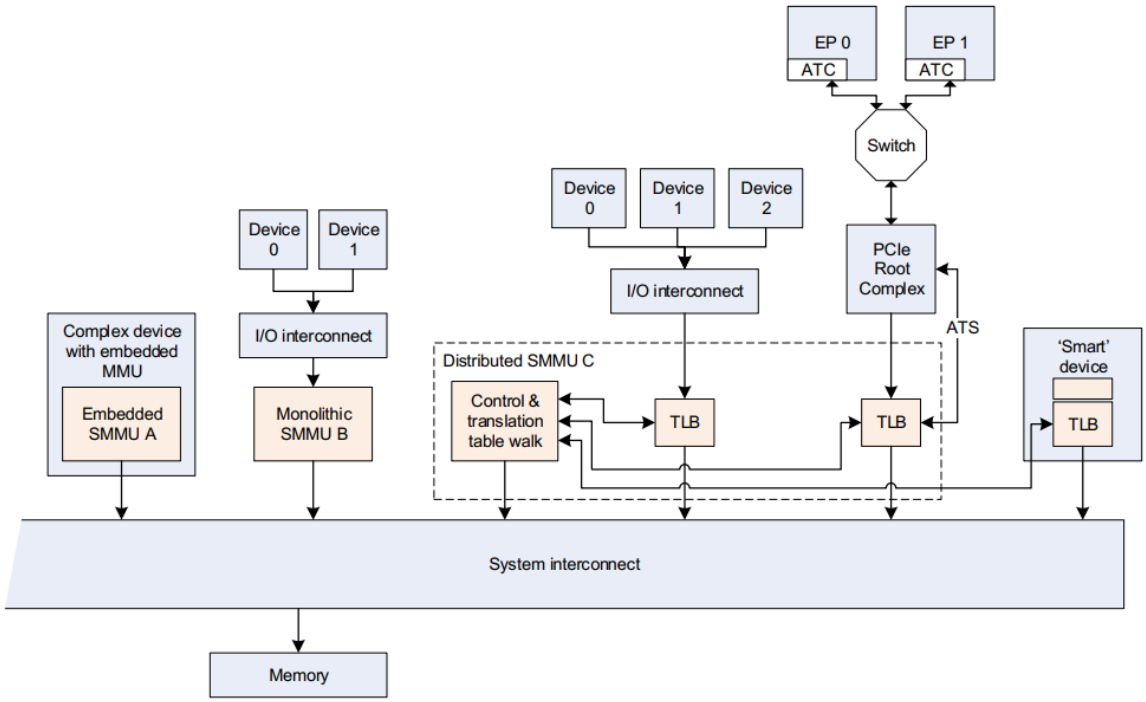


Figure 2.3: Example SMMU implementations

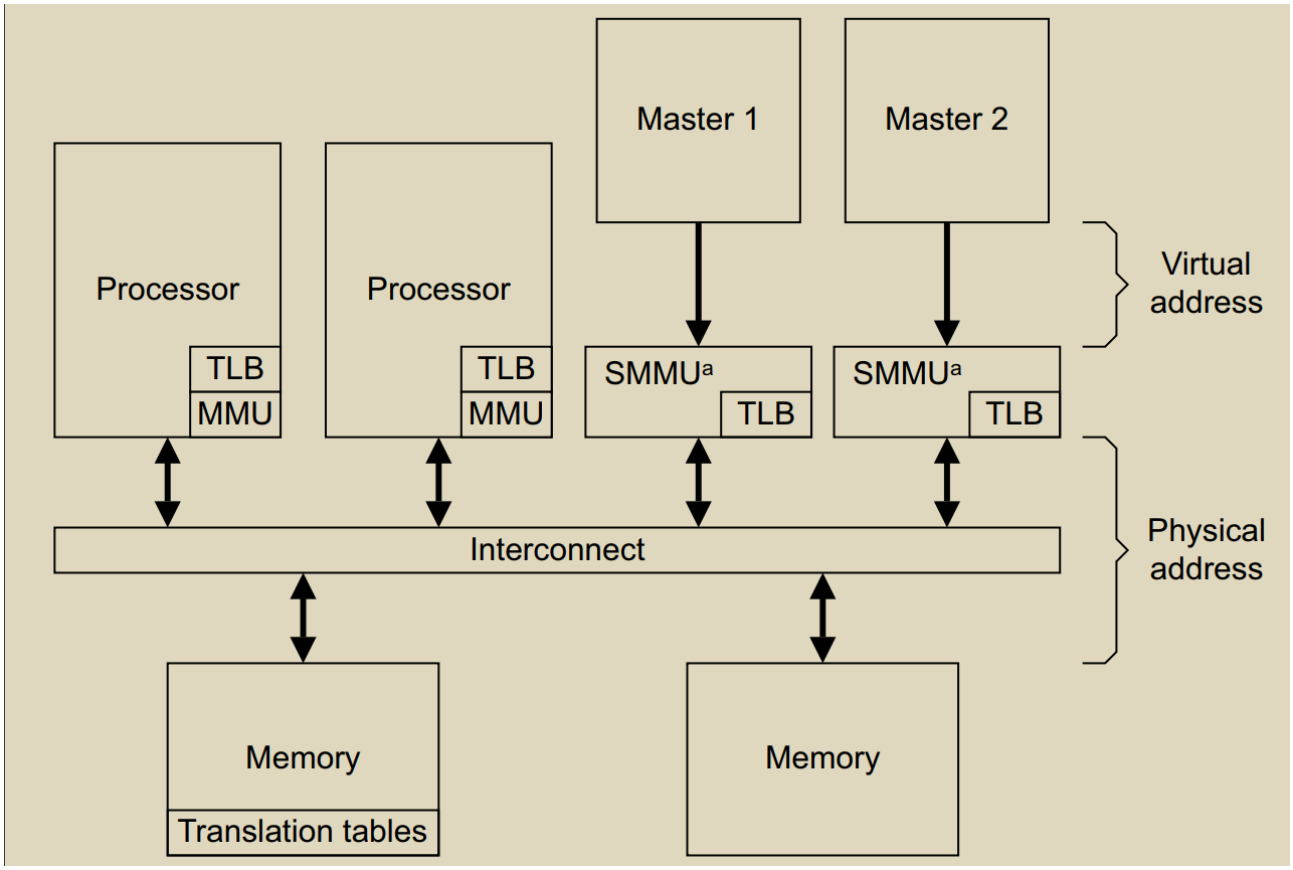
1 SMMU A complex device

2 SMMU B device I/O DMA

3 SMMU C

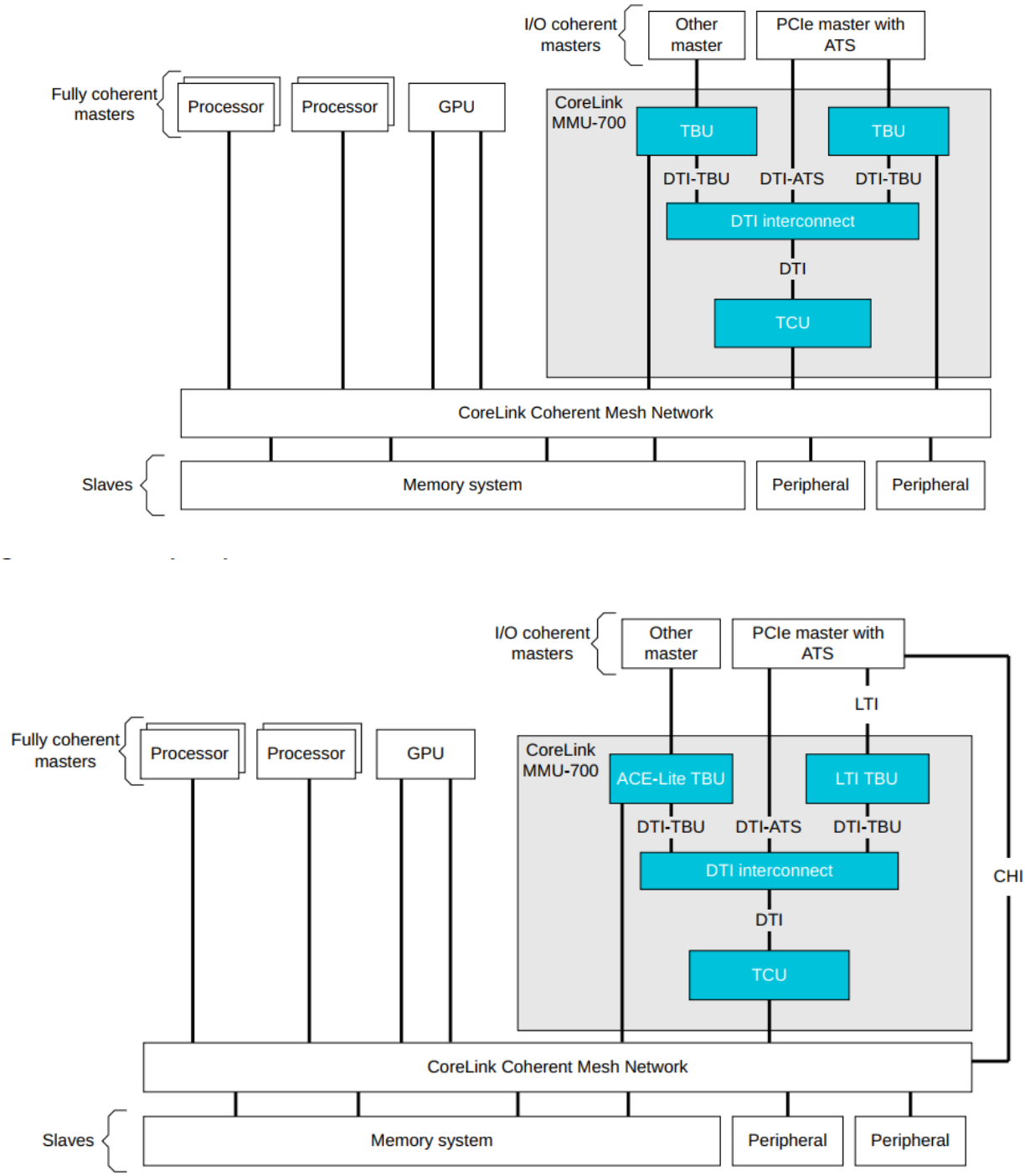
a. central translation table walker

b. device PCIe



MMU-700

Figure 3-1: Example system with the MMU-700



The MMU-700 contains the following key components: **Translation Buffer Unit (TBU)** The TBU contains Translation Lookaside Buffers (TLBs) that cache translation tables. The MMU-700 implements a TBU that can be connected to single master or multiple masters. It is also possible to connect multiple TBUs to a single master to improve performance. These TBUs are local to the corresponding master and can be one of the following:

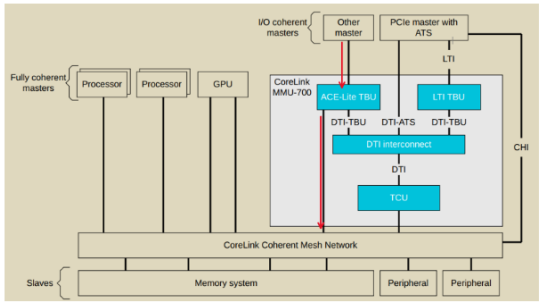
- ACE-Lite TBU
- LTI TBU

Translation Control Unit (TCU) The TCU controls and manages the address translations. The MMU-700 implements a single TCU. In MMU-700-based systems, the AMBA® DTI protocol defines the standard for communicating with the TCU. See the AMBA® DTI Protocol Specification.

DTI interconnect The DTI interconnect connects multiple TBUs to the TCU

TBU **TBU ACE PA transaction c2000 TBU LTI transaction PA pcie controller**

step3.SMMU使用PA发起transaction



step3.SMMU把PA返回给pcie controller，由pcie controller发起transaction（pcie<->ni700<->cmn）

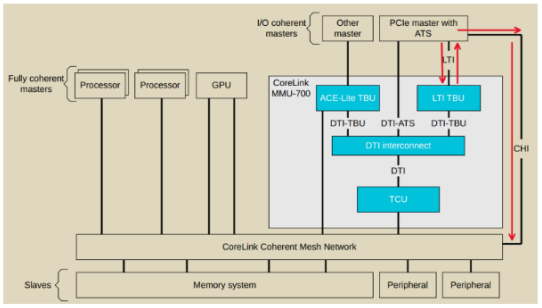


Figure 3-3: MMU-700 ACE-Lite TBU

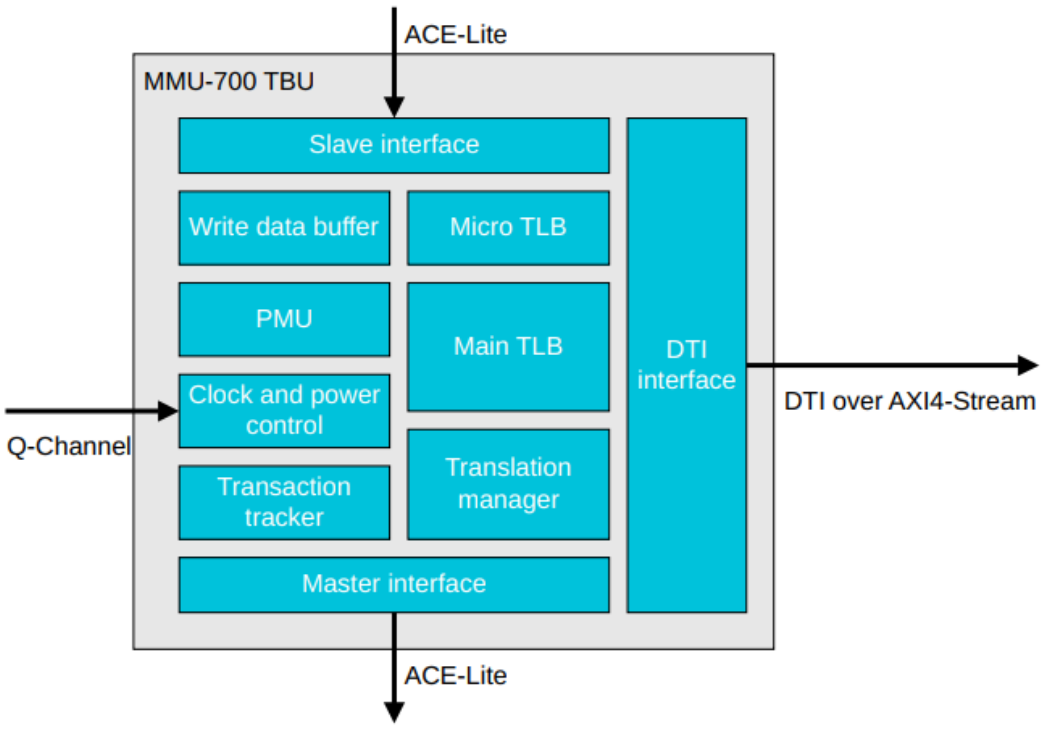
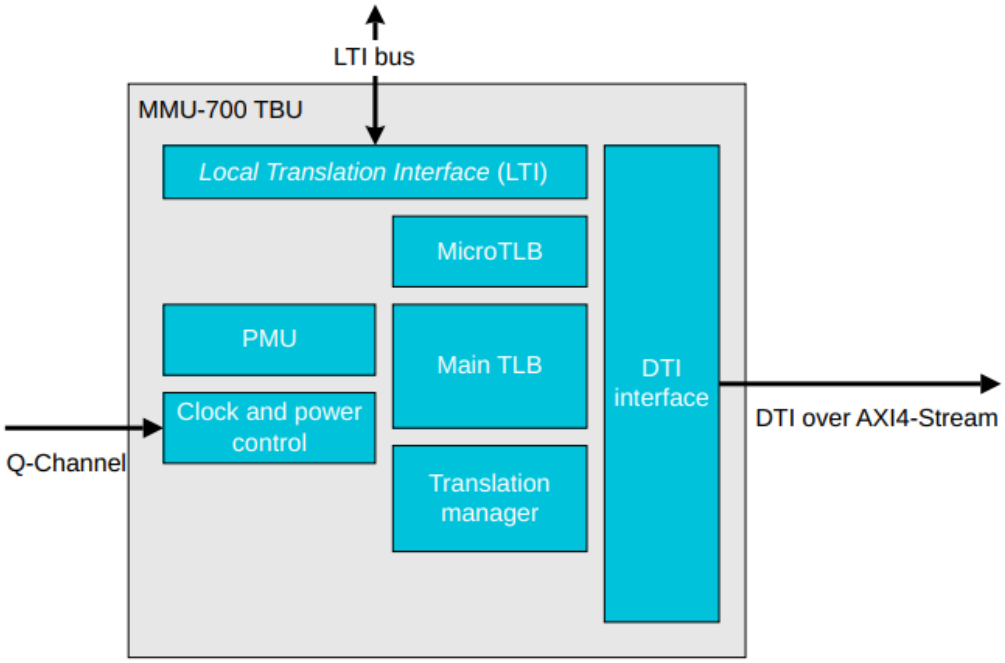


Figure 3-4: MMU-700 LTI TBU



TLB

The TBU contains Translation Lookaside Buffers (TLBs) that cache translation tables.

TBU cache micro TLB mainTLB

PA = F(streamID, substreamID, Address)

step1.master pcie controller VA transaction

step2.SMMU VA SMMU_CRO.SMMUEN SMMU, bypass, VA PA DMA , StreamlD STE;

STE.Config Stage1 , VA IPA Stage2 , STE.S1ContextPtr SubStreamID CD, Stage 1 translation table ASID, Stage 1 translation table walk, VA IPA Stage 2 ;

STE.Config Stage2 , IPA PA SMMU , STE.S2TTB Stage 2 translation table VMID, Stage 2 translation table walk, IPA PA SMMU ,

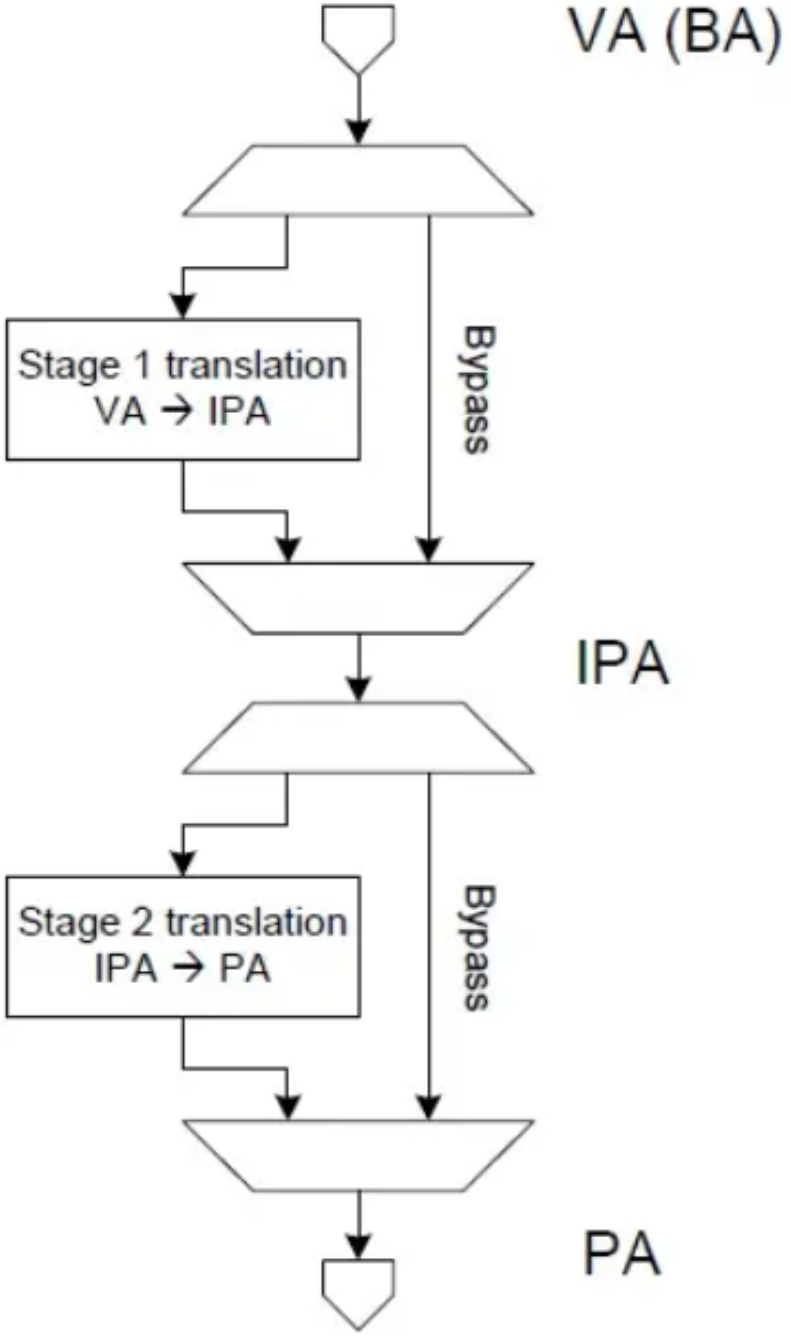
Note: Stage 1/2 translation table walk , SMMU TLB , , translation table walk translation table walk TLB

untranslated transaction, PCIe ATS translation request/PRI

PA

- TBU TLB PA
- TBU TLB TBU TCU transaction table walk memory translation table VA PA

step3.SMMU PA transaction



SMMU enable SMMU

StreamID	SMMU SMMU Endpoint SMMU StreamID (SubstreamID PCIe PASID)
STE	Stream Table Entry, STE stage2 CD Context Descriptor .
CD	Context Descriptor, stage1

STE CD

STE stream table entry stage1 CD **CD stage1** stage2 stage2

3.3.1.1 Linear Stream table

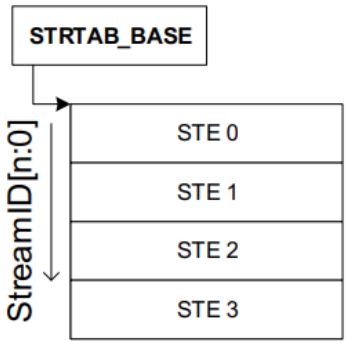


Figure 3.1: Linear Stream table

STE

3.3.1.2 2-level Stream table

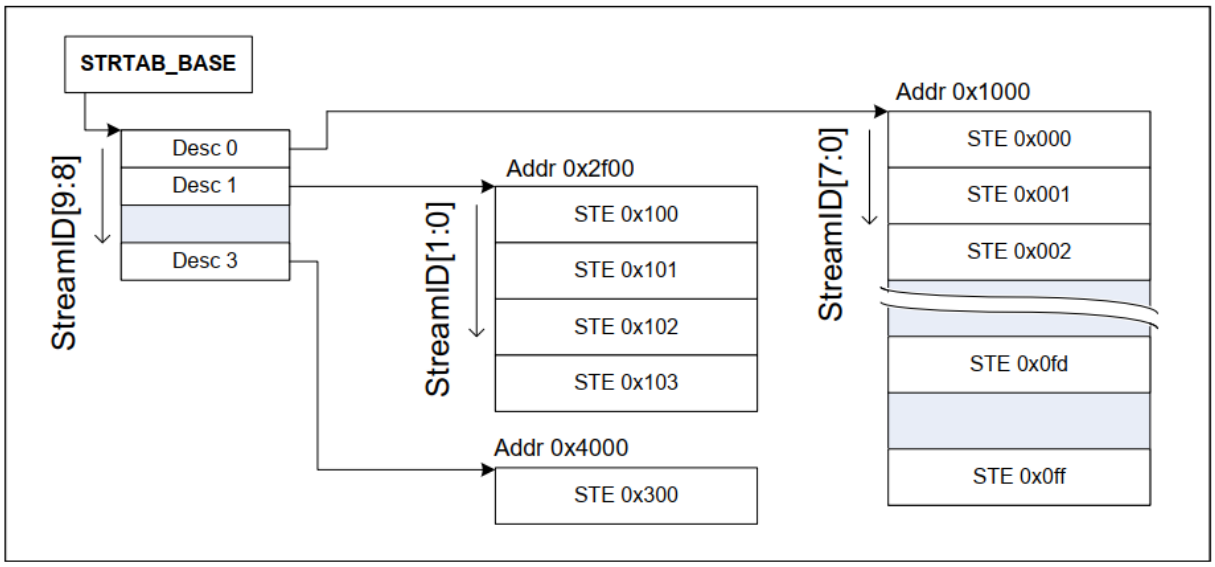
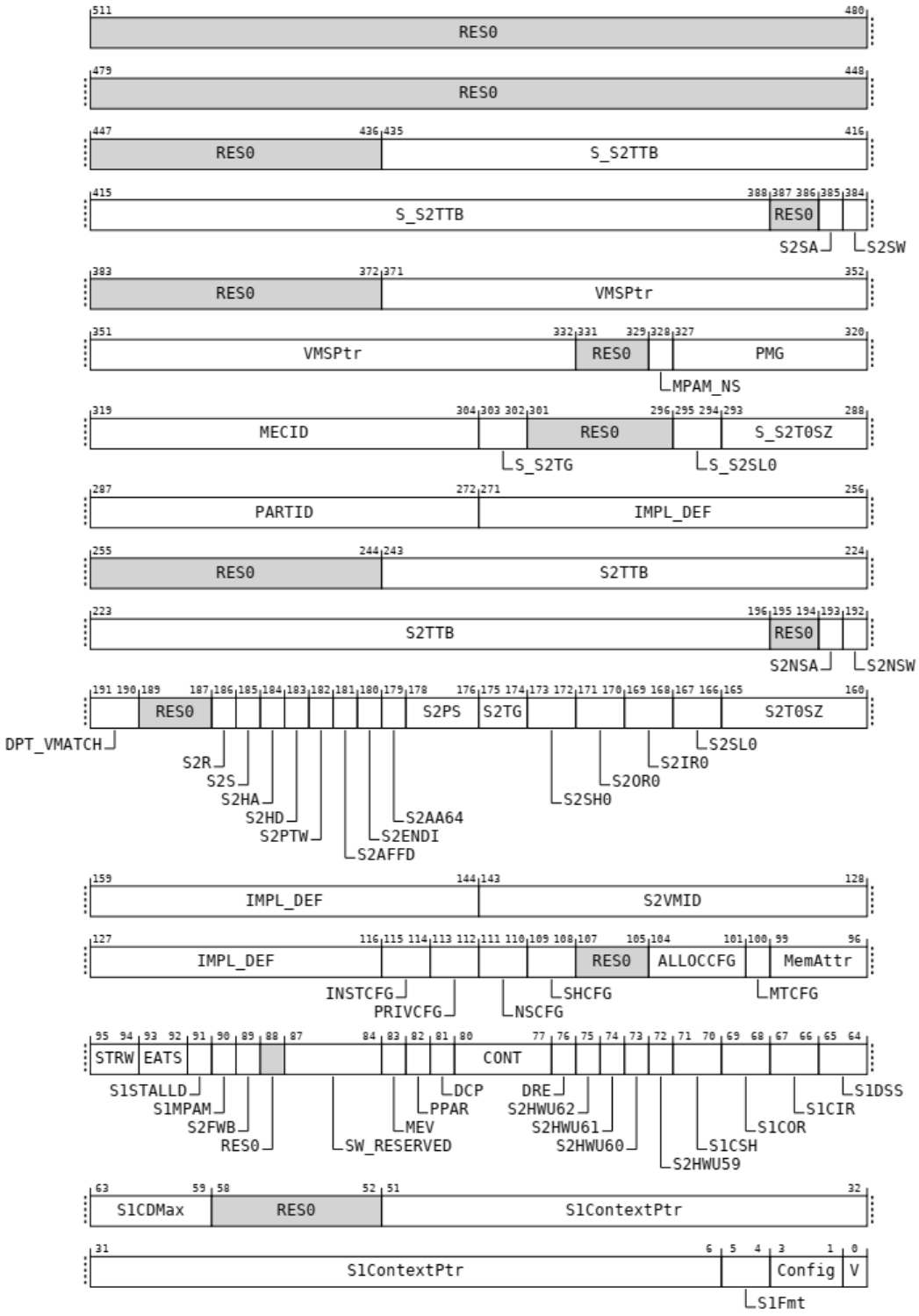


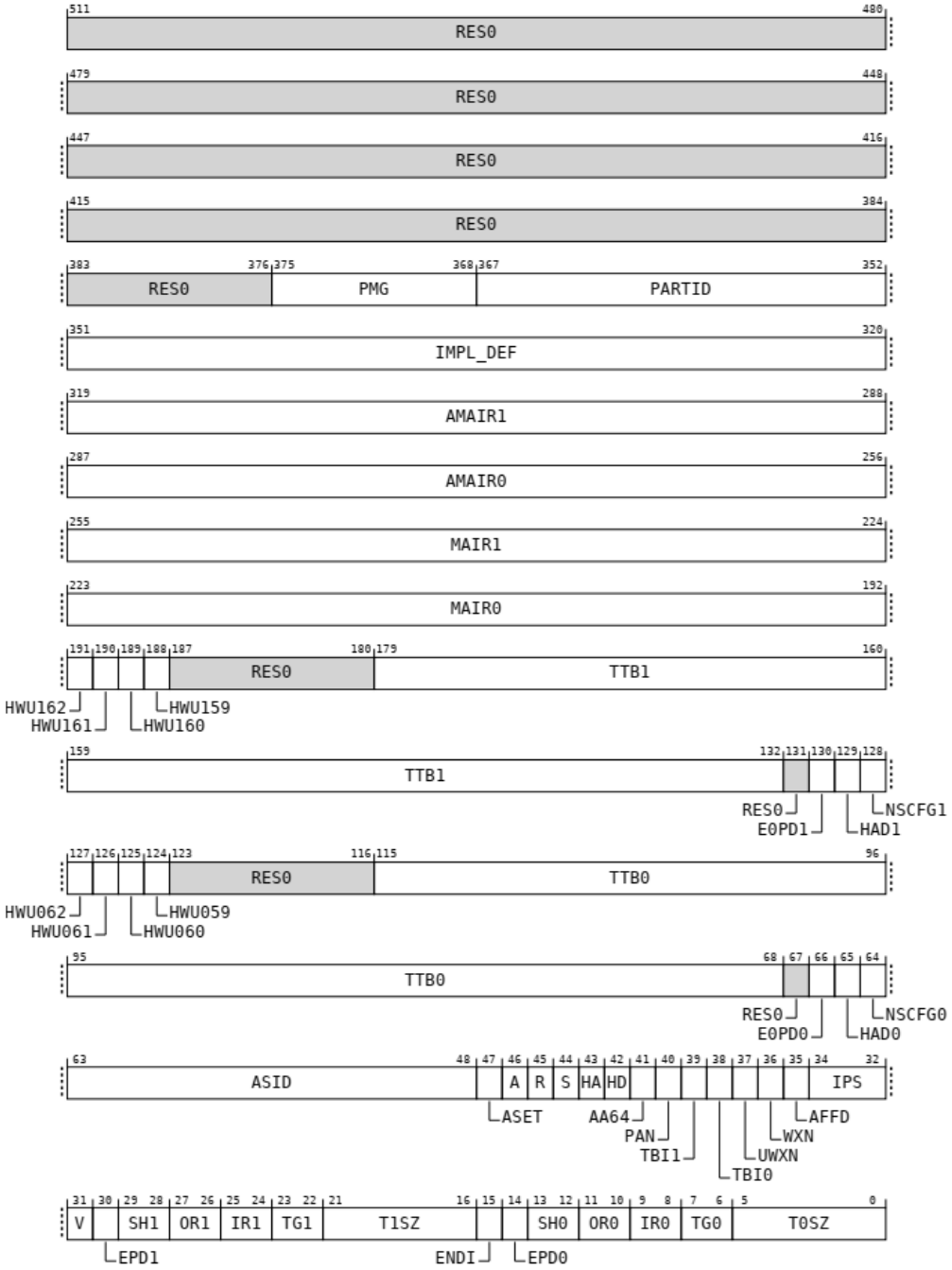
Figure 3.2: Example Two level Stream table with SPLIT == 8

STE 64B SMMU STRTAB_BASE STE STRTAB_BASE STE Stream id STE linear STRTAB_BASE + sid * 64 STE 64B STE 2-level sid L1_STD STRTAB_BASE + sid[9:8] * 8, L1_STD 8B , L1_STD sid STE l2ptr + sid[7:0] * 64 .

STE stream bypass, stage1, stage2 stage1 + stage2

5.2 Stream Table Entry




511 480 RES0
479 448 RES0
447 416 RES0
415 384 RES0
383 376 375 368 367 352 RES0 PMG PARTID
351 320 IMPL_DEF
319 288 AMAIR1
287 256 AMAIR0
255 224 MAIR1
223 192 MAIR0
191 190 189 188 187 180 179 160 RES0 TTB1
HWU162 HWU161 HWU160 HWU159
159 132 131 130 129 128 TTB1 RES0 E0PD1 HAD1 NSCFG1
127 126 125 124 123 116 115 96 RES0 TTB0
HWU062 HWU061 HWU060 HWU059
95 68 67 66 65 64 TTB0 RES0 E0PD0 HAD0 NSCFG0
63 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 32 ASID A R S HA HD IPS
ASET AA64 PAN TBI1 TBI0 UWXN WXN AFFD
31 30 29 28 27 26 25 24 23 22 21 16 15 14 13 12 11 10 9 8 7 6 5 0 V SH1 OR1 IR1 TG1 T1SZ SH0 OR0 IR0 TG0 T0SZ
EPD1 ENDI EPD0

CD

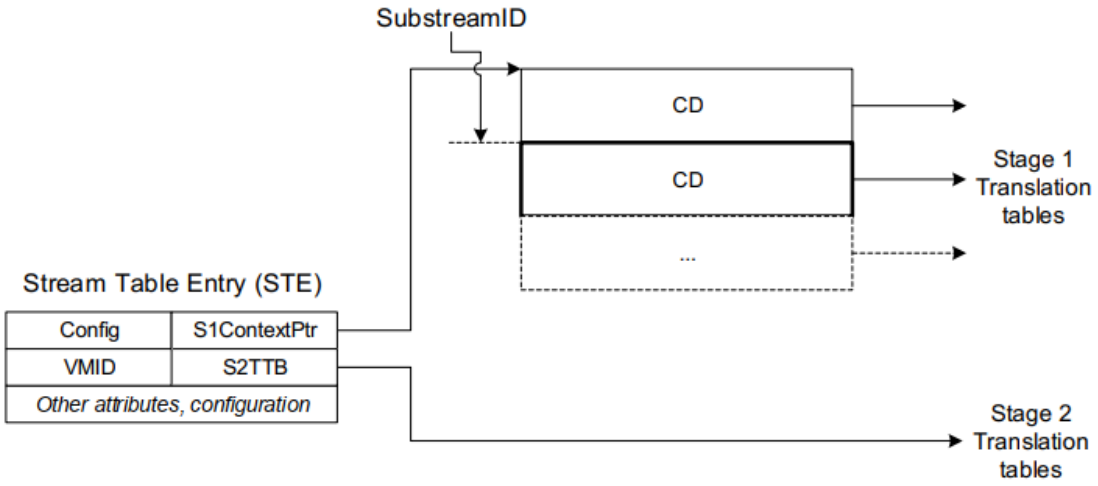


Figure 3.4: Multiple Context Descriptors for Substreams

SubStreamID CD SubStreamID SubStreamID

SMMU command queue event queue

Command Queue Event Queue SMMU 2 Queue Command queue SMMU command queue, SMMU command queue Event Queue SMMU SMMU Event queue Event queue

streamID substreamID

StreamID

StreamID is generated from the PCI RequesterID so that **StreamID[15:0] ==RequesterID[15:0]**

SubstreamID

The SubstreamID is equivalent to a **PCIe PASID**. Because the concept can be applied to non-PCIe systems, it has been given a more generic name in the SMMU. The maximum size of SubstreamID, 20 bits, matches the maximum size of a PCIe PASID.

c2k StreamID N 24 StreamID[23:0]

StreamID[15:0] pcie RequesterID[15:0].

StreamID[23:16] Root Complex c2000 root complex id :

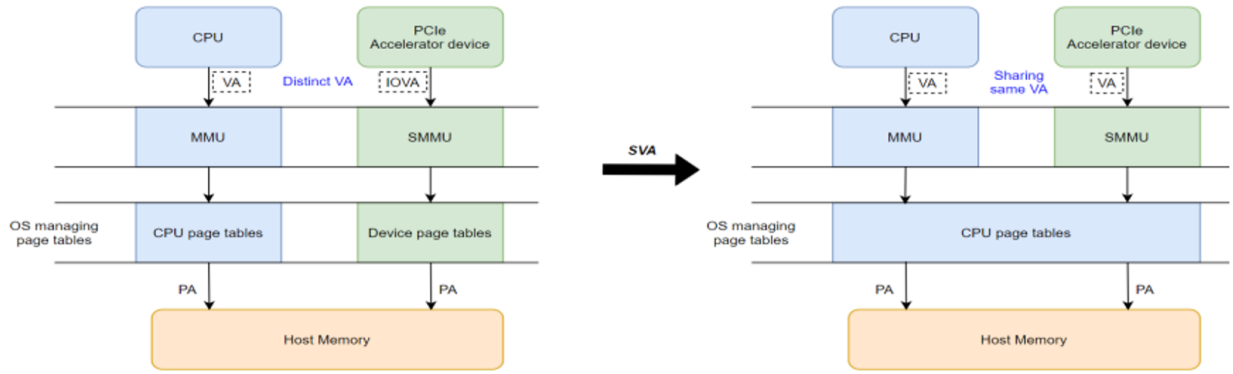
SubstreamID pcie PASID

1.1.3 SVA

[toc]

SVA

SVA (Shared Virtual Addressing) SVA SVM (Shared Virtual Memory) SVA Secure Virtual Machine



SVA

- CPU GPU/NPU/SmartNIC/FPGA/ASIC/Accelerator
- device cpu
- DMA cpu pin DMA page

SVA

1. PASID
2. IO page fault PCIe PRI
3. **MMU IOMMU**

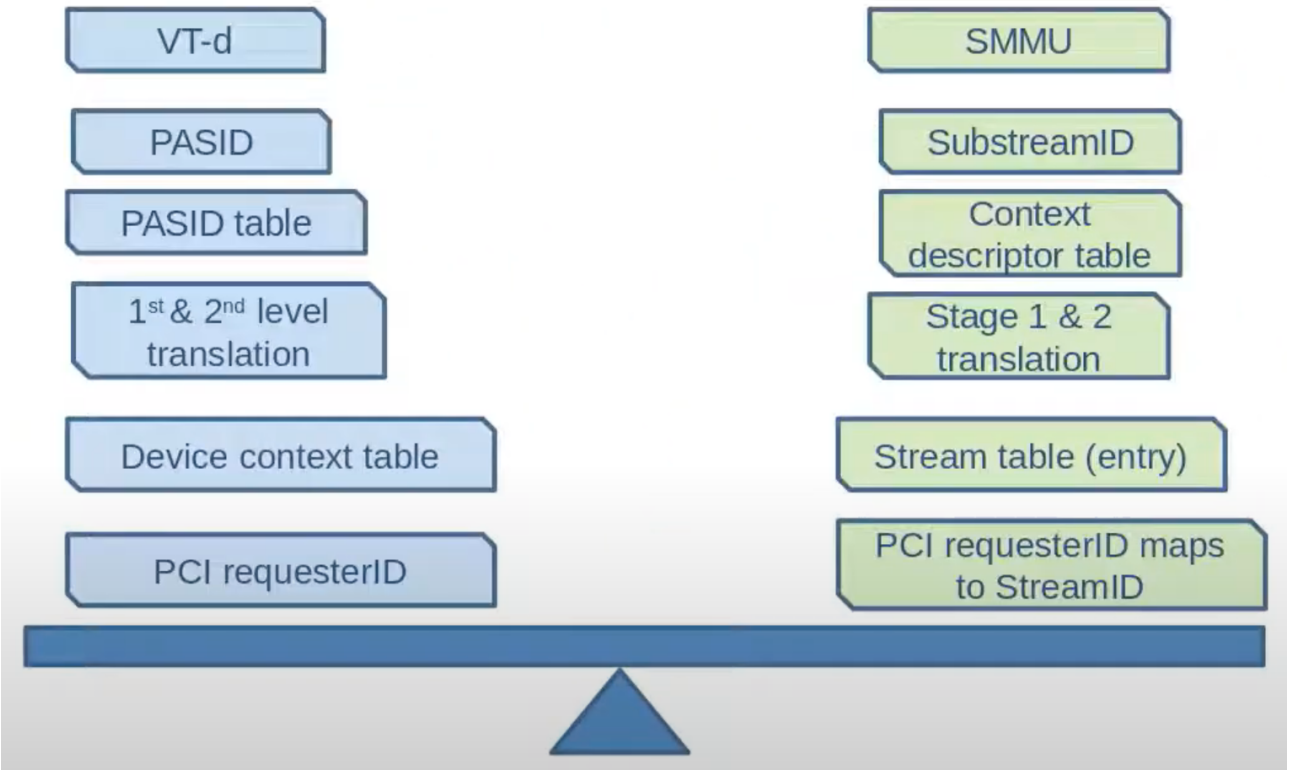
C2K PPC MMU ARM SMMU

ARM SMMU

PPC MMU

SVA MMU SMMU page table SMMU 64bit EA MMU 64bit EA SLB 78bit VA

SVA Intel ARM



device BDF streamID BDF CPU process PASID DMA process PASID

PASID

PASID PCIe spec PASID TLP prefix 20 bit PASID value **process** Requeseter ID (BDF) Requester ID PASID Requester ID PASID PASID prefix EP PASID ATS PRI PASID TLP prefix

6.20.2 PASID TLP Layout

A TLP may contain at most one PASID TLP Prefix.

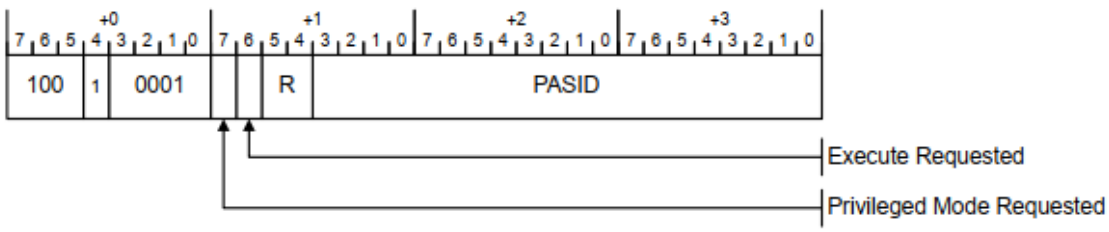


Figure 6-20 PASID TLP Prefix

Intel IOMMU VT-d

- Intel: (Share Virtual Memory)
  - CPU access DDR through MMU
  - Device access DDR through IOMMU
  - MMU share the same page table with IOMMU
  - (used by AMD' s Secure Virtual Machine in Linux)

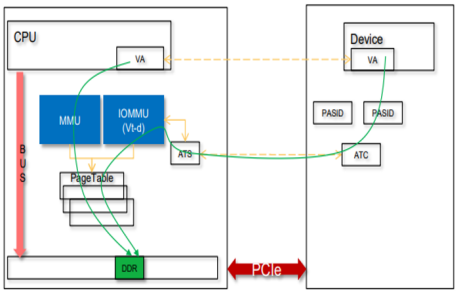


Figure 3-5 illustrates the paging structure for translating a 48-bit address to a 4-KByte page.

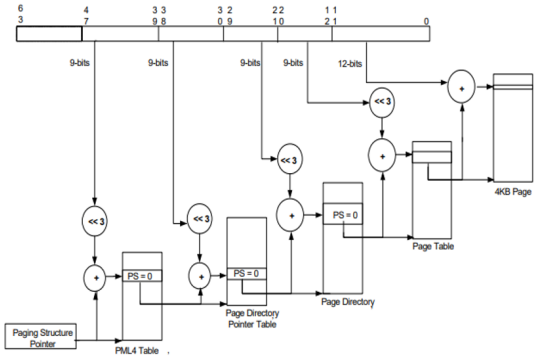
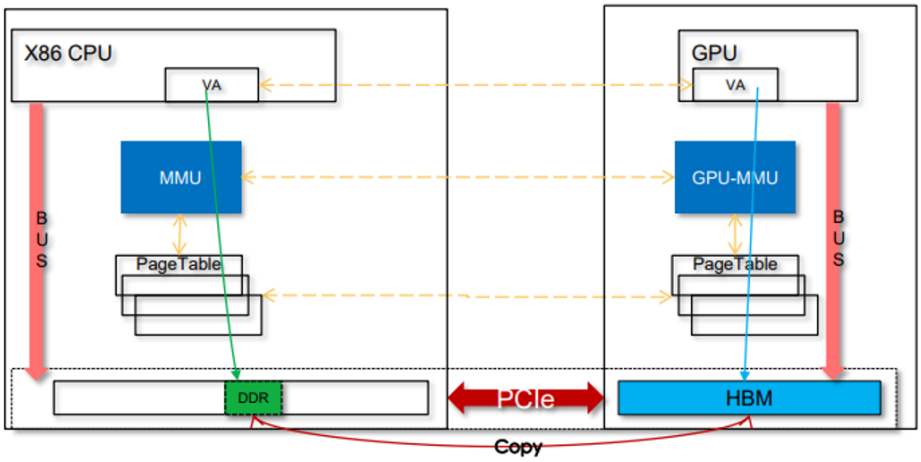


Figure 3-5. Address Translation to a 4-KByte Page

Nvidia UVA

- Nvidia: (UVA: Unified Virtual addressing)
  - GPU has its own MMU
  - CPU can only access DDR while GPU can only access HBM
  - GPU MMU mirror the Page Table of CPU
  - When GPU access a VA not populated in HBM it will copy the data from DDR , and vice versa.



1.2 NoC

1.2.1 CHI

[toc]

CHI

SoC

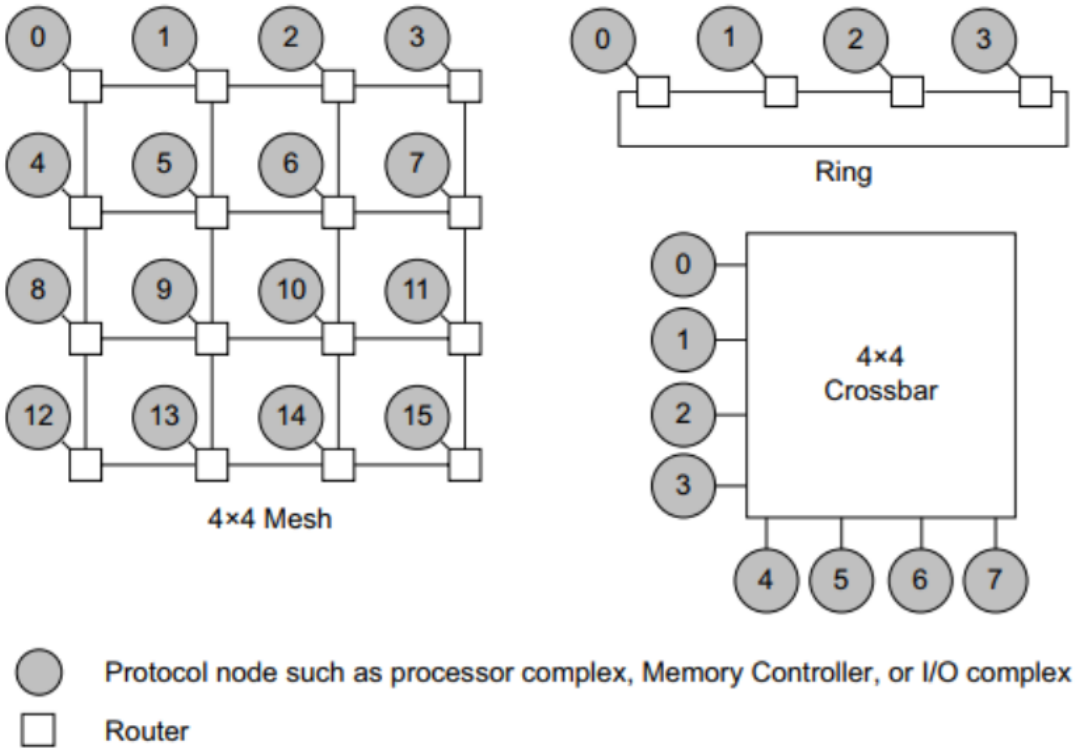


Figure 1-1 Example interconnect topologies

Component

**RN **

**HN ** cache CMN700 HNF SLC Combined PoS/PoC(ordering) Snoop Filter

**SN ** HN

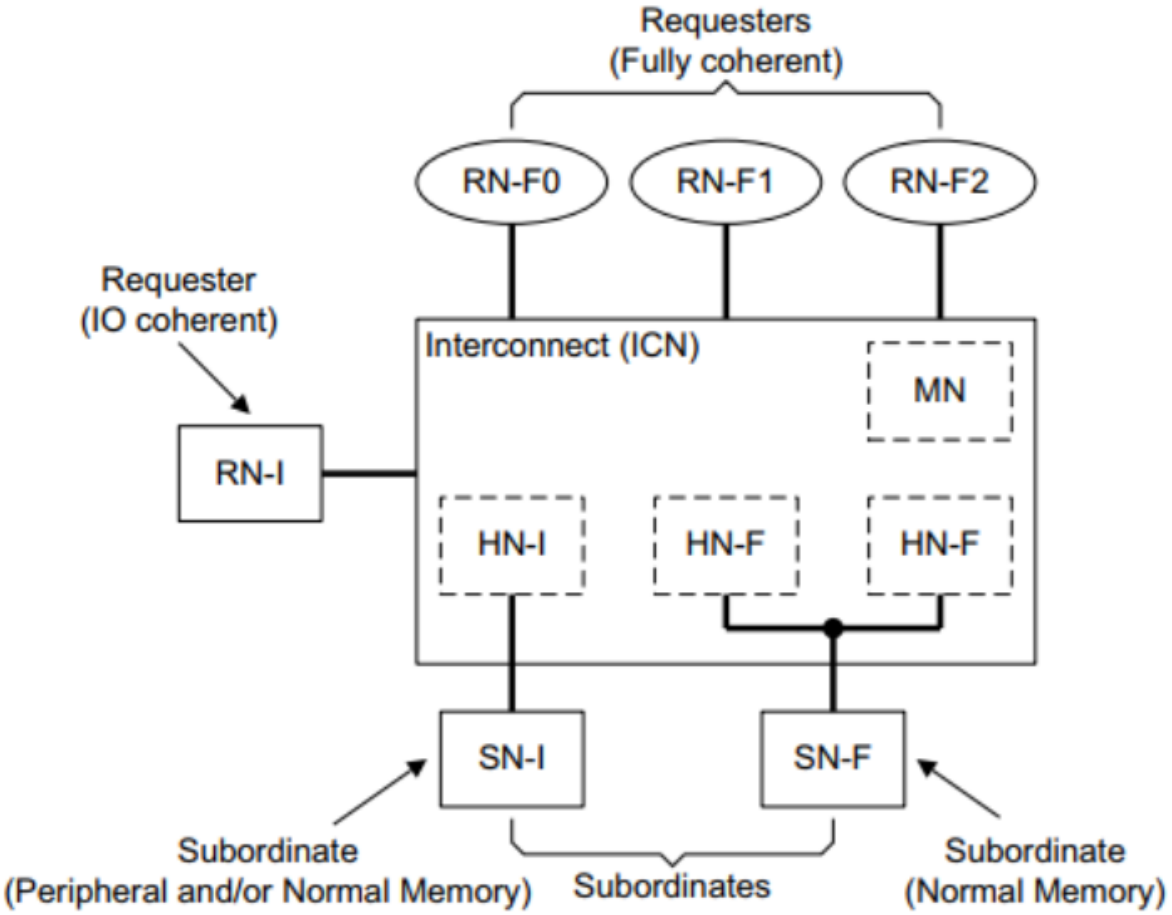


Figure 1-4 Protocol node examples

PCIe

Table 1-1 Layers of the CHI architecture

Layer	Communication granularity	Primary function
Protocol	Transaction	The Protocol layer is the topmost layer in the CHI architecture. The function of the Protocol layer is to: • Generate and process requests and responses at the protocol nodes. • Define the permitted cache state transitions at the protocol nodes that include caches. • Define the transaction flows for each request type. • Manage the protocol level flow control.
Network	Packet	The function of the Network layer is to: • Packetize the protocol message. • Determine the source and target Node IDs required to route the packet over the interconnect to the required destination and add to the packet.
Link	Flit	The function of the Link layer is to: • Provide flow control between network devices. • Manage link channels to provide deadlock-free switching across the network.

**Protocol ** transaction

**Network ** transaction packet SAM flit

**Link ** Flit 4 6 channel

Protocol

Transaction CHI tansaction read write snoop atomic snoop

Type	Transaction Type	Description
Read	**ReadNoSnoop**	
Read	**ReadOnce**	
Read	**ReadShared**	
Read	**ReadClean**	
Read	**ReadNotSharedDirty**	
Read	**ReadUnique**	
Snoop	**CleanUnique**	
Snoop	**CleanShared**	
Snoop	**MakeUnique**	
Snoop	**Evict**	
Snoop	**StashOnce**	
Snoop	**StashUnique**	
Snoop	**DVM Operations**	DVM TLB
Write	**WriteNoSnoop**	
Write	**WriteUnique**	
Write	**WriteLineUnique**	
Write	**WriteBack**	
Write	**WriteClean**	

Type	Transaction Type	Description
Write	**WriteEvictFull**	
Atomic	**Atomic Operations**	- -

transaction flow

5.1.1 Read transactions with DMT and without snoops

For Read transactions without snoops, this specification recommends the use of *Direct Memory Transfer* (DMT).

Figure 5-1 shows an example DMT transaction flow using the ReadShared transaction.

In Figure 5-1, a response from SN-F to HN-F is not required because CompAck from the Requester is used to deallocate the request at Home.

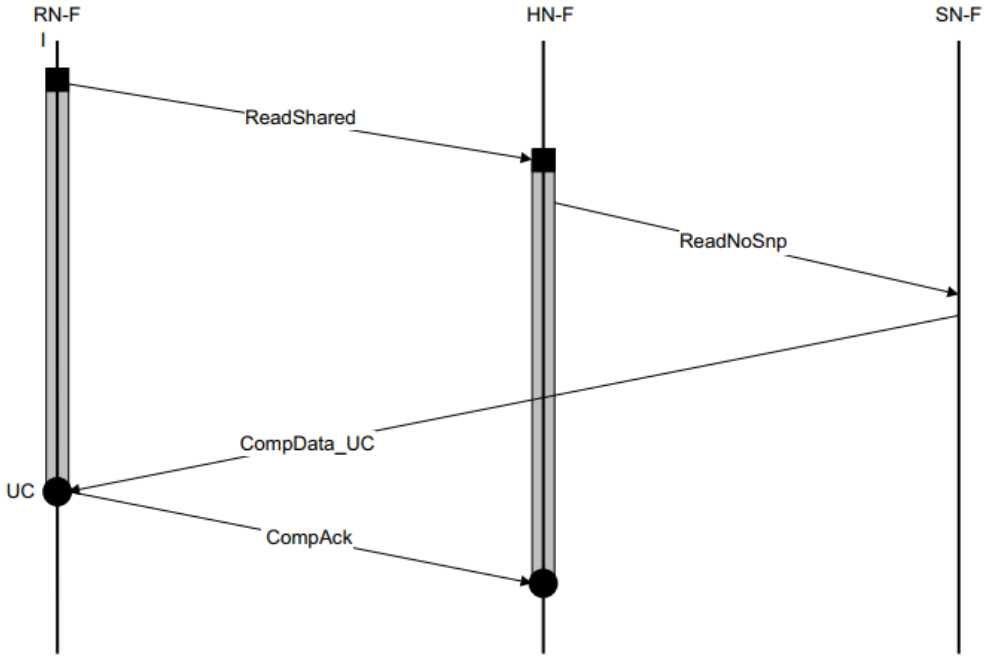


Figure 5-1 DMT Read transaction example without snoops

1. RN-F sends a Read request to HN-F.
2. HN-F sends a Read request to SN-F

   The ID field values in the Read request are based on where the data response is to be sent. Data can be sent to the Requester or to the HN-F.
3. SN-F sends a data response directly to RN-F.
4. RN-F sends CompAck to HN-F as the request is ReadShared and requires CompAck to complete the transaction.

Network

tansaction SAM packet srcid tgtid

SAM RNSAM, HNSAM

power of 2 hashing hieracical hashing

Link

4 6 channel

TX: RSP DATA REQ

RX: RSP DATA SNP

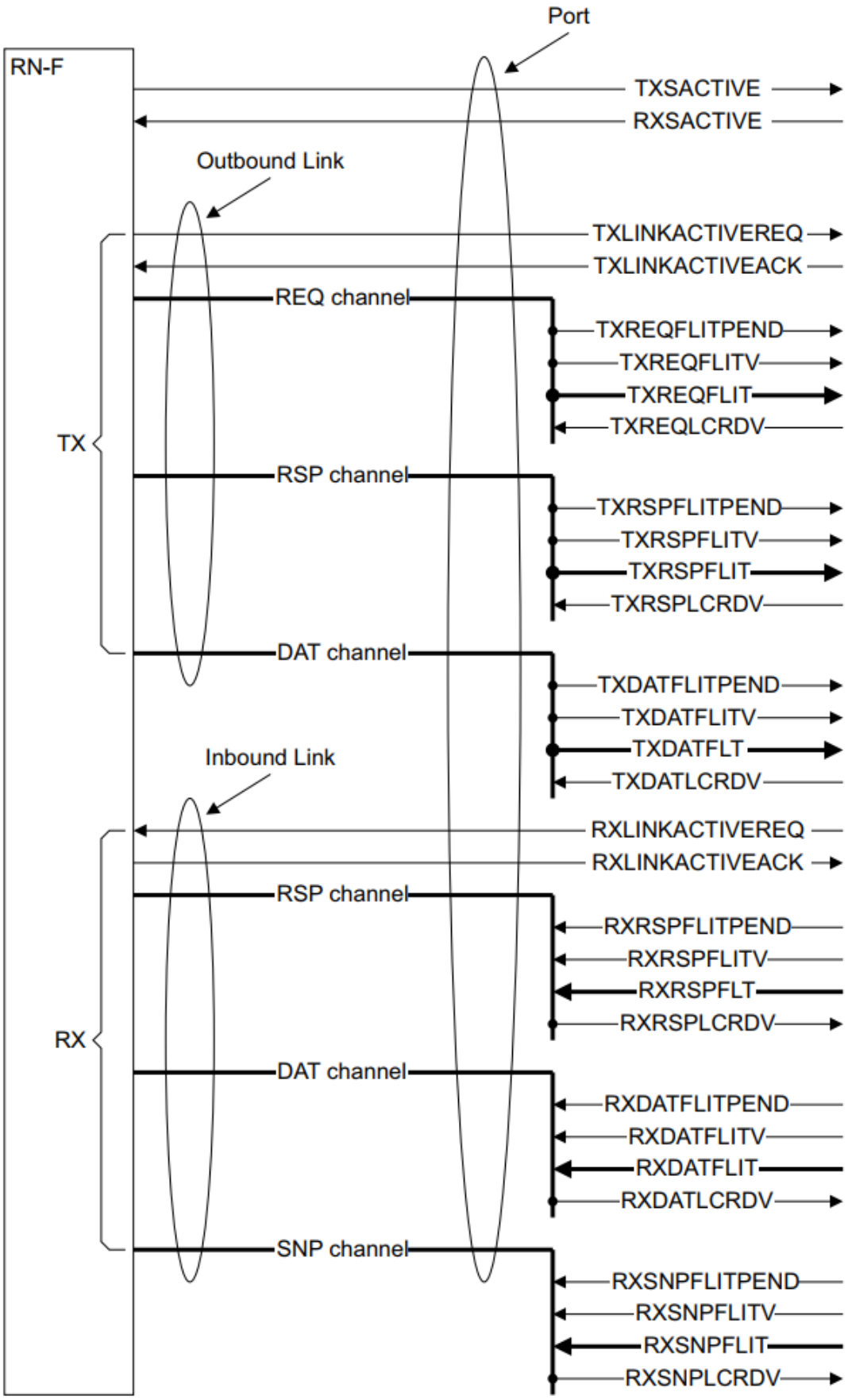
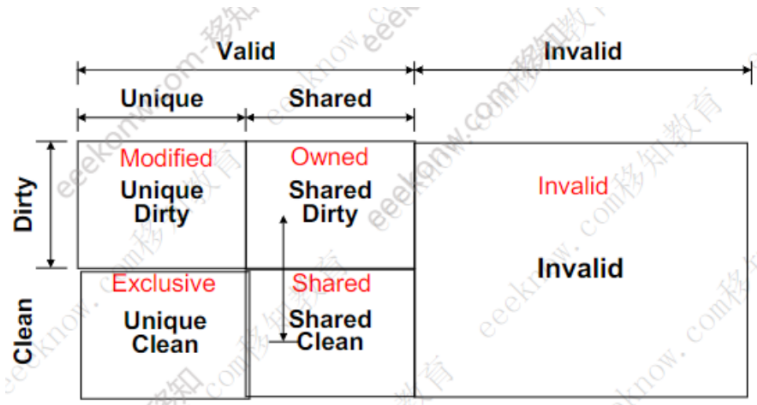


Figure 13-4 Relationship between links, channels, and port

cache cache



Unique – The cache line is only in this cache

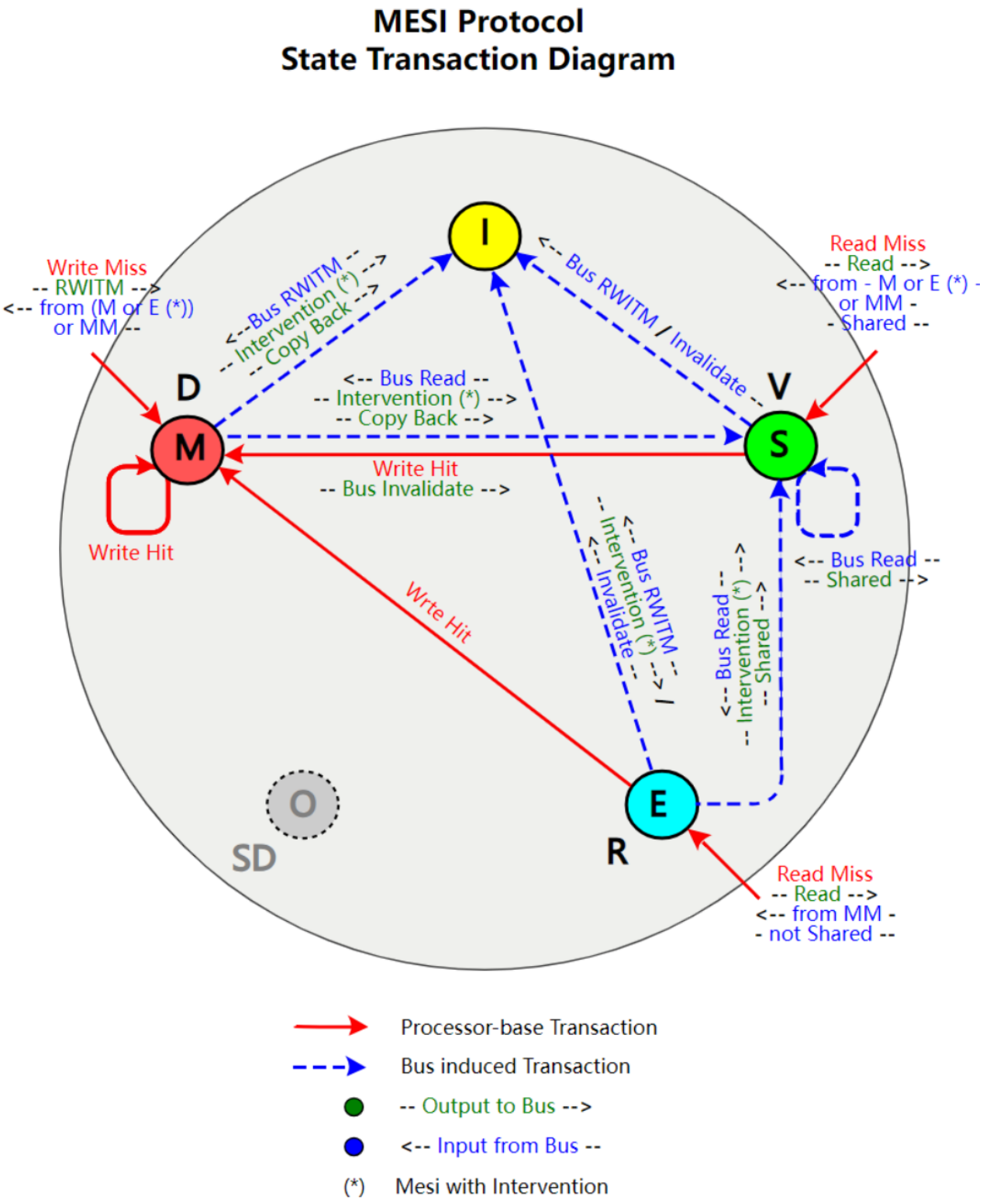
Shared – The cache line may be in multiple ca

Dirty – The cache line has been mo
respect to main memory and this cache must
main memory is
eventually updated

Clean – The cache line has the same
main memory or the value of shared dirty dat
written
to main memory

Invalid – The cache line is not valid

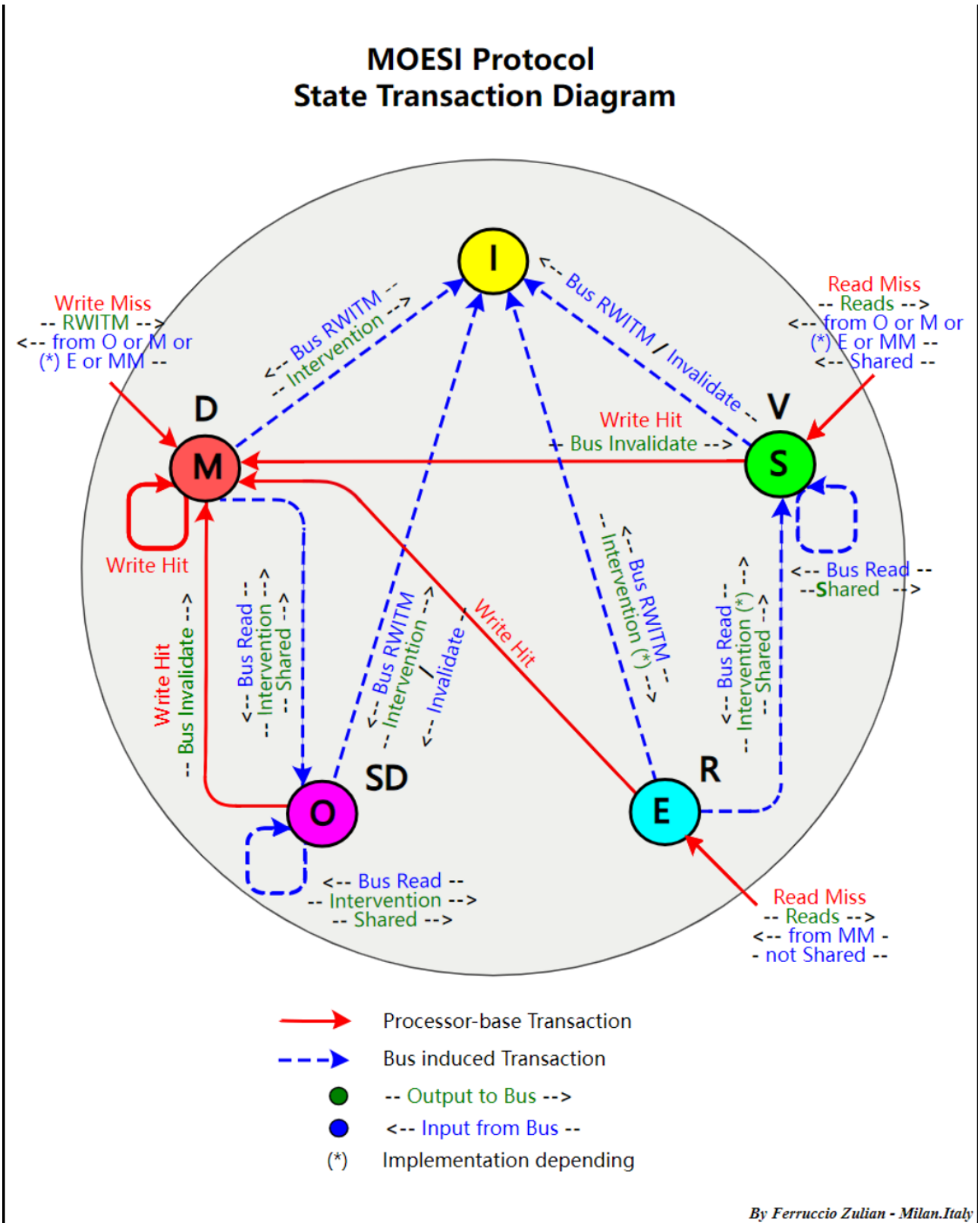
Cache line states



MESI

By Ferruccio Zulian - Milan.Italy

MOESI MESI M CPU MOESI Owned O M M O MOESI O S E M I MESI MESI S MOESI O O MOESI S S CPU O CPU O



	address	data
DDR	0x100	0xa
Cache 1	0x100	0xa
Cache 2	0x100	0xa

core 2 cache

	address	data
DDR	0x100	0xa
Cache 1	0x100	0xa
Cache 2	0x100	**0xa --> 0xb**

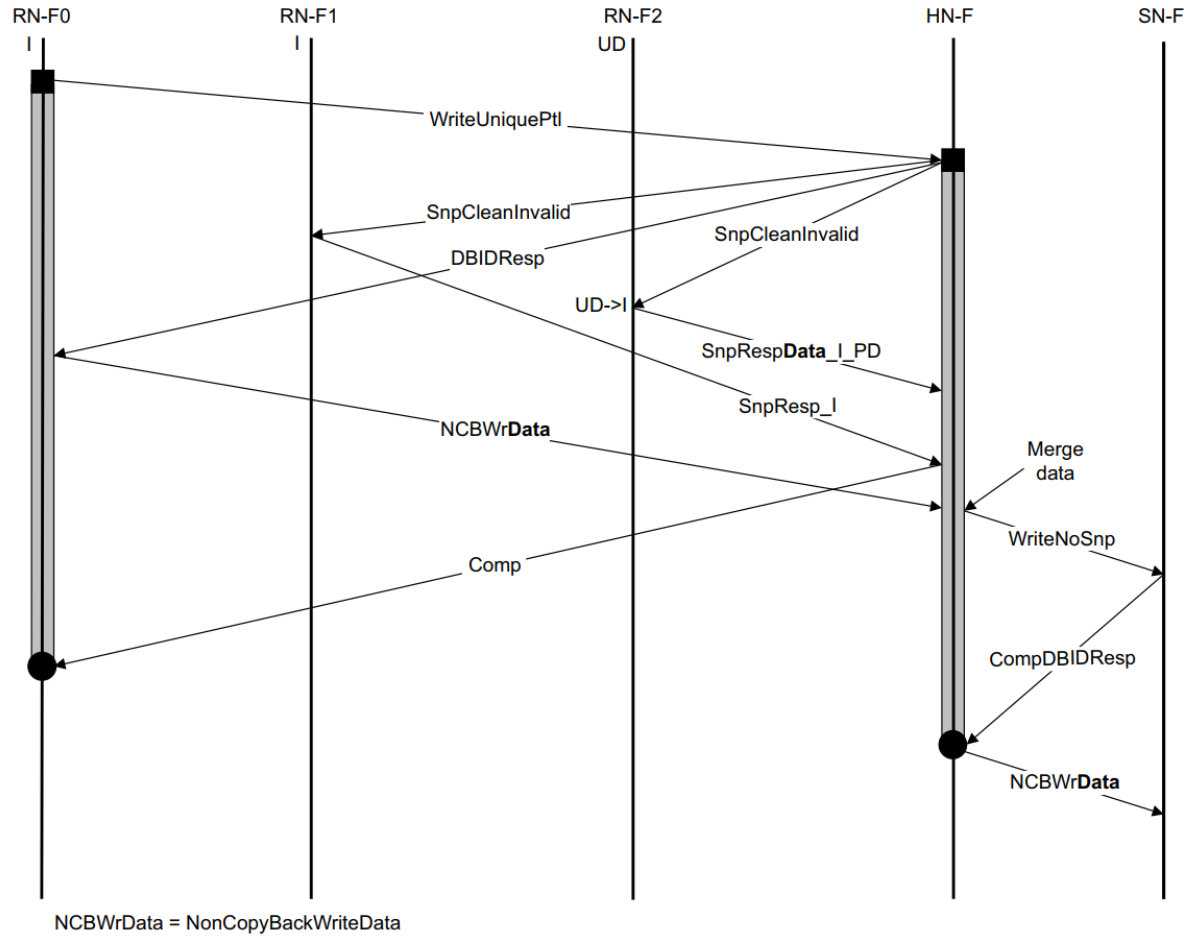
HN directory snoop cache ddr

5.3.2 Write transaction with snoop and separate responses

An example of this type of flow is a WriteUniquePtl transaction.

The Comp_I response from HN-F must be sent when the coherency activity is complete at HN-F.

Figure 5-16 shows the transaction flow. The copy of data being transferred is marked in bold.



CHI - snooping directory-based AMBA CHI Coherent Hub Interface directory-based

-
- /

-
-

  CHI

-
-

- /
-

-
-

-
-

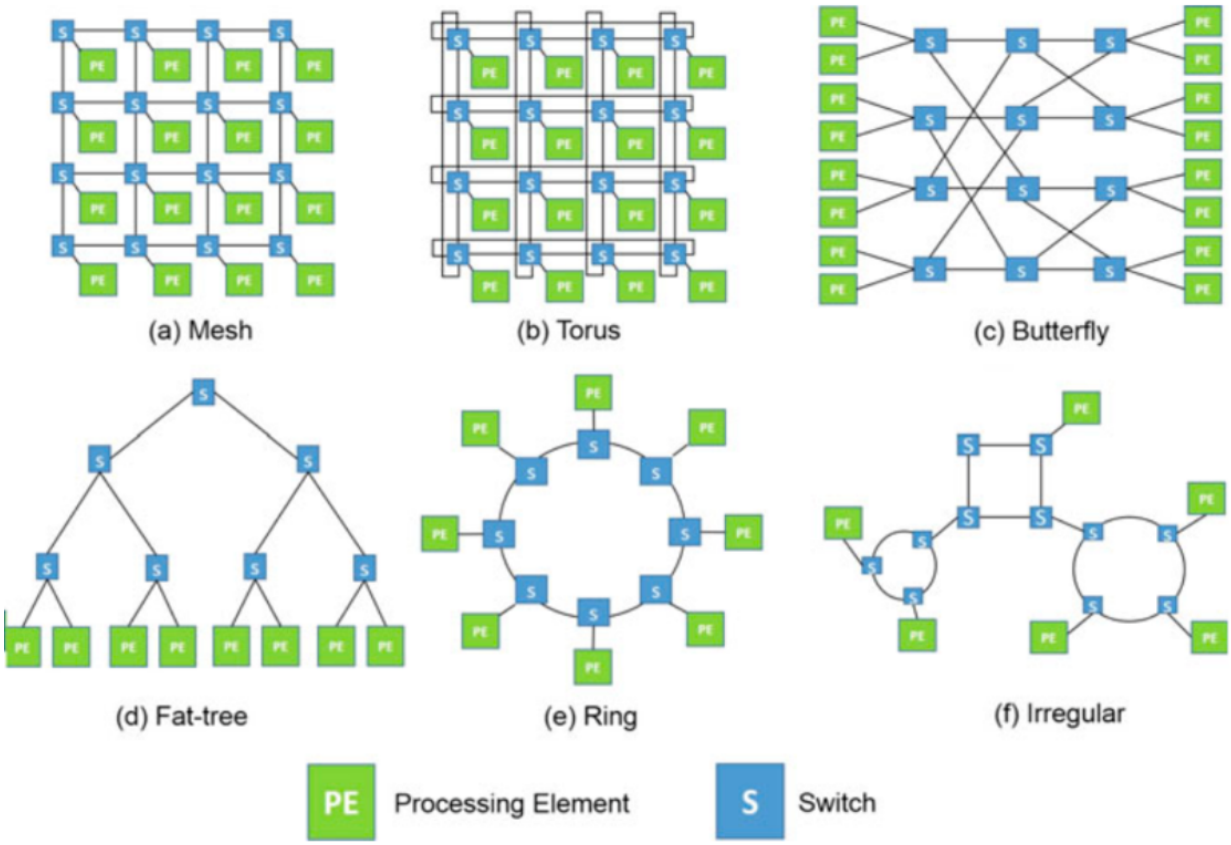
- DVM

1.2.2 NoC

[toc]

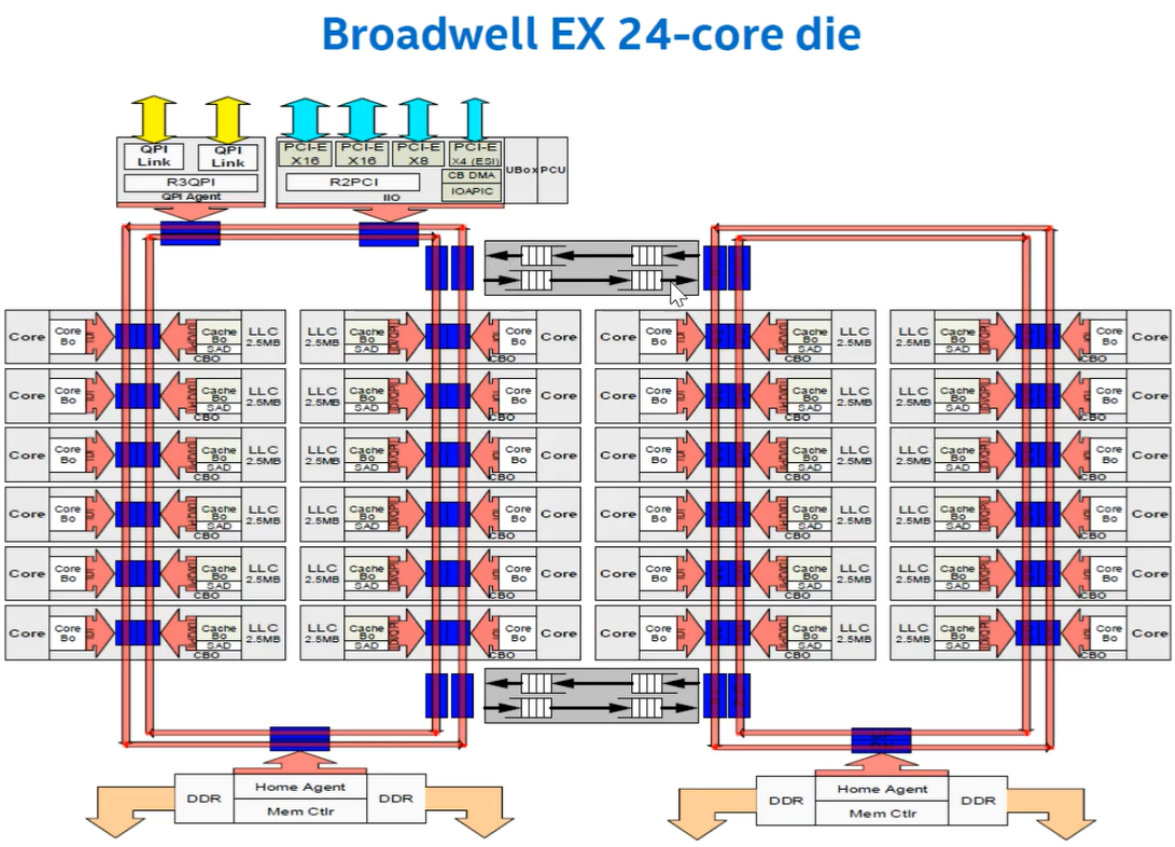
CPU On-Chip Interconnect

1. Shared Bus a. b. c.
2. Crossbar Switch a. b. c.
3. Ring Interconnect a. b. c.
4. Mesh Torus Mesh and Torus Interconnects a. b. c.
5. Network-on-Chip, NoC a. b. c.

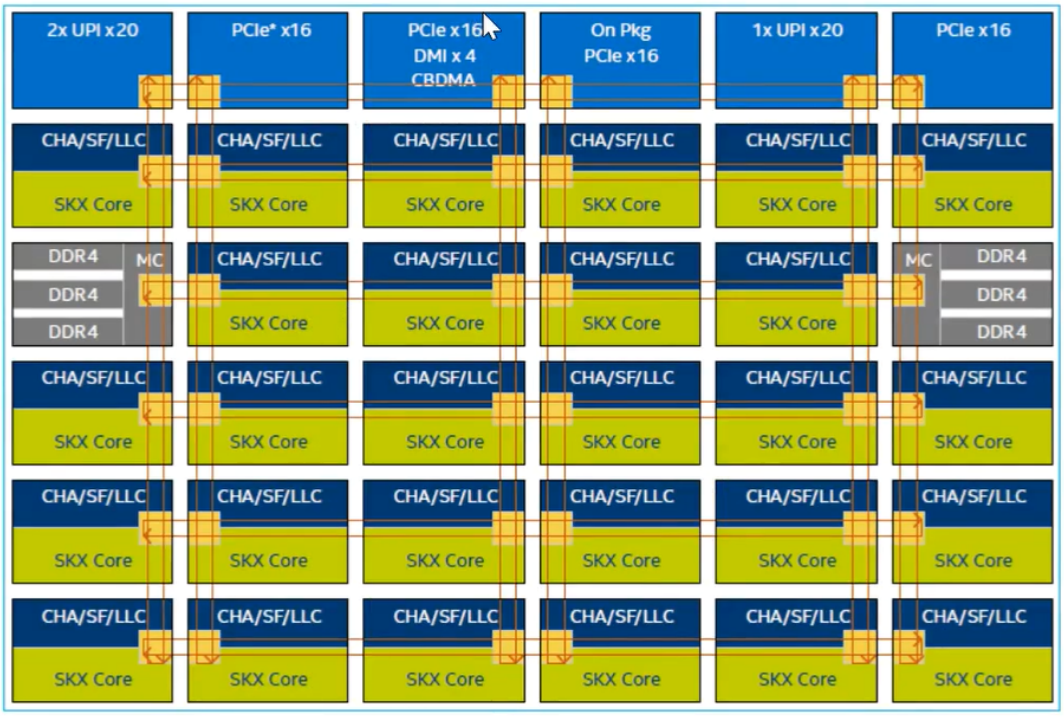


1.

Intel Broadwell ring core core FIFO



Intel Skylake Mesh



1. ——
2. Efficient Barrier Implementation on the POWER8 Processor
3.

1.2.3 CMN700

[toc]

CMN-700 HNF/HNI/HNP

Home Node HN Home Node HN SAM HNF HN

HNF DRAM HNF

HNF

1. **System Level Cache** The System Level Cache (SLC) is a last-level cache. The SLC allocation policy is exclusive for data lines, except where sharing patterns are detected and pseudo-inclusive for code lines, as indicated by the RN-Fs. All code lines can be allocated into the SLC on the initial request. When MTE is enabled, SLC stores data and tags.
2. **Combined PoS/PoC** The combined Point-of-Serialization/Point-of-Coherency (PoS/PoC) is responsible for the ordering of all memory requests sent to the HN-F. Ordering includes serialization of multiple outstanding requests and actions to the same line, and request ordering as required by the RN-F.
3. **Snoop Filter** The Snoop Filter (SF) tracks cachelines that are present in the RN-Fs. It reduces snoop traffic in the system by favoring directed snoops over snoop broadcasts when possible. This approach substantially reduces the snoop response traffic that might otherwise be required.

snoop Snoop Filter

Ans HN invalid

HNI The I/O coherent Home Node (HN-I) is a Home Node for all CHI transactions targeting AMBA subordinate devices

The HN-I acts as a proxy for all the RNs of CMN-700, converting CHI transactions to ACE5-Lite transactions. The HN-I includes support for the correct ordering of Arm device types.

The HN-I does not support caching of any data read from or written to the downstream ACE5-Lite I/O subordinate subsystem. Any cacheable request that is sent to the HN-I does not result in any snoops being sent to RN-Fs in the system. Instead, the request is converted to the appropriate ACE5-Lite read or write command and sent to the downstream ACE5-Lite subsystem.

If an RN-F caches data that is read from or written to the downstream ACE5-Lite I/O subordinate subsystem, coherency is not maintained. Any subsequent access to that data reads from or writes to the ACE5-Lite I/O subordinate subsystem directly, ignoring the cached data

There are HN-I types with extra functionality. These types include:

- HN-T HN-I that has a debug trace controller and DVM Node. CMN-700 can have zero or more HN-I and HN-T instances.
- HN-D HN-I that has a debug trace controller, DVM Node, and configuration subordinate. CMN-700 must have exactly one HN-D instance.
- HN-P HN-I that is optimized for peer-to-peer PCIe traffic.
- HN-V A device that includes an HN-I and DVM Node (DN).

HNP The I/O coherent Home Node with PCIe optimization (HN-P) is a device that includes the HN-I functionality and dedicated trackers for PCIe peer-to-peer traffic. HN-P can only be used to connect to PCIe subordinates.

	HNF	HNI	HNP
	System Level Cache Combined PoS/PoC Snoop Filter	CHI ACE5-Lite	HNI PCIe P2P
	Yes		PCIe
		ACE5-Lite	PCIe
	NA	AMBA	PCIe
			PCIe

1.2.4 CMN700

1.2.5

[toc]

ARM ARMv8 EL3 EL2 EL1

ARMv7 **ARMv7** **0x00000000** **4** ARMv8 128 32
ARMv8 64 32 64

ARMv8 Vector Base Address Register VBAR

ARM M7 Arm v7 32

地址	异常	进入模式
0x00000000	复位	管理模式
0x00000004	未定义指令	未定义模式
0x00000008	软件中断	管理模式
0x0000000C	中止（预取）	中止模式
0x00000010	中止（数据）	中止模式
0x00000014	保留	保留
0x00000018	中断 IRQ	中断模式
0x0000001C	快中断 FIQ	快中断模式

```
Address      Exception Type
0x00000000   Reset
0x00000004   Undefined Instruction
0x00000008   Software Interrupt (SWI)
0x0000000C   Prefetch Abort
0x00000010   Data Abort
0x00000014   Reserved
0x00000018   IRQ
0x0000001C   FIQ
```

Interrupt I/O

1.
2. IRQ
3.
4. NVIC GIC
5.

-
-
-

Exception

1.
2.
3.
4.

5.

-
-
-

Interrupt	Exception

ARM

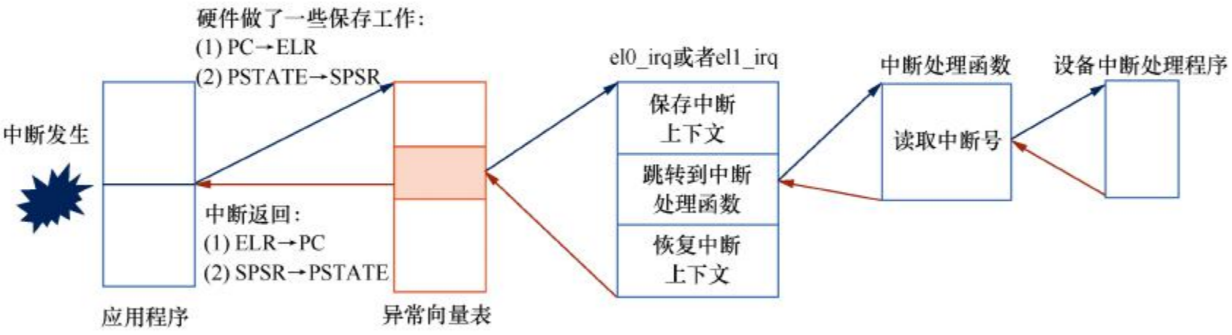
ARM64 ——nIRQ nFIQ

ARM64 ARM IRQ Interrupt Request FIQ Fast Interrupt Request

PSTATE CPU I IRQ F FIQ

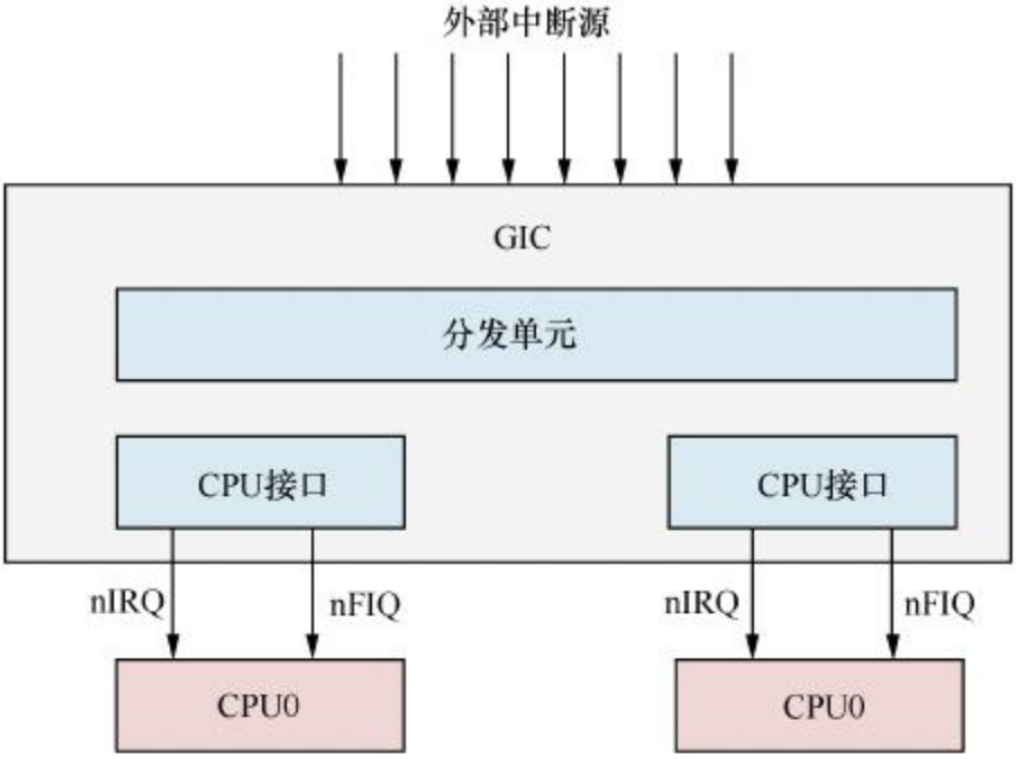
表 12.1 中断信号线

信　号	类　型	说　明
nIRQ	输入	IRQ 信号，每个 CPU 内核都有一根 nIRQ 信号线。它是一个低电平有效的信号线。低电平表示要激活这个 IRQ；高电平表示不要激活这个 IRQ。nIRQ 一直保持高电平直到触发 IRQ
nFIQ	输入	FIQ 信号，每个 CPU 内核都有一根 nFIQ 信号线。它是一个低电平有效的信号线。低电平表示要激活这个 FIQ；高电平表示不要激活这个 FIQ。nFIQ 一直保持高电平直到触发 FIQ



▲图 12.2 中断处理过程

NVIC GIC



▲图 12.1　中断控制器

NVIC Nested Vectored Interrupt Controller　NVIC　　Cortex-M　　NVIC

1.
   -
   -
2.
   - NVIC Cortex-M
   -
3.
   - NVIC
4.
   - 　　VTOR
   - 240　　Cortex-M

GIC Generic Interrupt Controller　GIC　　Cortex-A　　GIC

1.
   - GIC
   - 　　Distributor CPU　CPU Interface
2.
   -
   -
3.
   - 　　1020
   - GICv3 GICv4
4.

- GIC SoC
-

- **NVIC**
  - **Cortex-M**
  -
  -
- **GIC**
  - **Cortex-A**
  -
  -

1.2.6 ARM GIC

[toc]

GIC

ARM ARM7 ARM9 64 32
Interrupt Enable Register IER Interrupt Status Register ISR

SoC

-
- CPU PCIe
-

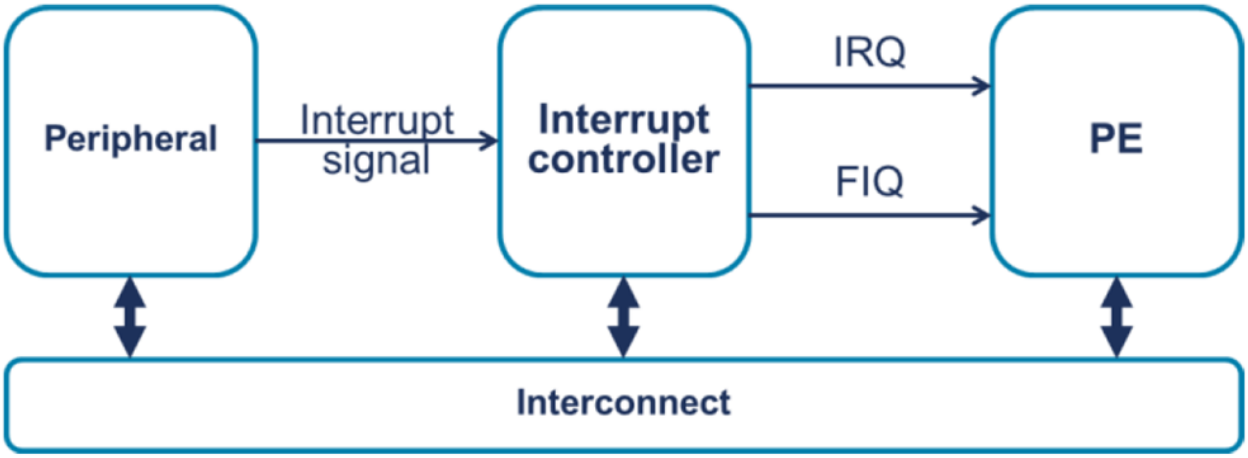
core core PCIe 500

ARM GIC V4 IP GIC-700 GIC 13.1 GIC-V1 GIC-390 IP ARM IP

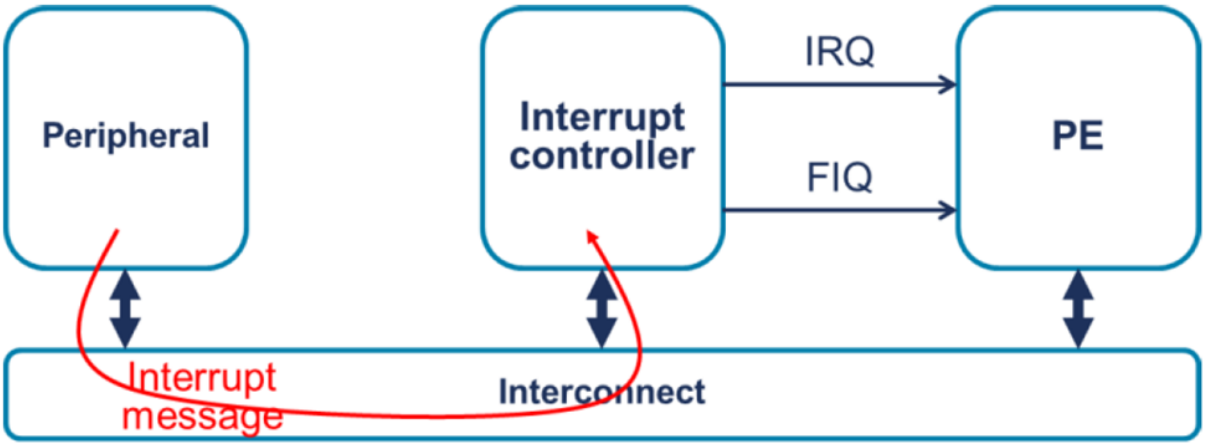
表 13.1 GIC 的发展史

版本	GIC-V1	GIC-V2	GIC-V3	GIC-V4
新增功能	❑ 支持 8 核， ❑ 支持多达 1020 个中断源， ❑ 支持用 8 位二进制数表示的优先级， ❑ 支持软件触发中断， ❑ 支持 TrustZone	❑ 支持虚拟化 ❑ 改进对安全软件的支持	❑ 支持的 CPU 核数大于 8 ❑ 支持基于消息的中断 ❑ 支持更多的中断 ID	支持注入虚拟中断
IP 核心	GIC-390	GIC-400	GIC-500、GIC-600	GIC-700
应用场景	Cortex-A9 MPCore	Cortex-A7/A9 MPCore	Cortex-A76 MPCore	

C2K GIC V4.1.



MSI



GIC-V3 GIC-V3 PCIe MSIX PCIe PCIe GIC-V3 CPU 8 GIC-V3 IP GIC-500 GIC-600

GIC-V4 GICv4 GICv3

1. **vPE**

GICv4 vPE

- GICv4 vPE Hypervisor

2.

GICv4 Direct Injection

- GICv4

3.

GICv4

- **vPE** GICv4 vPE

4.

GICv4

- **vLPI Control** GICv4

5.

GICv4

- GICv4

6.

GICv4

- GICv4

4 inactive pending CPU active CPU active and pending CPU

edge-triggered level-triggered

GIC

GIC GIC

表 13.2 GIC 分配的中断号范围

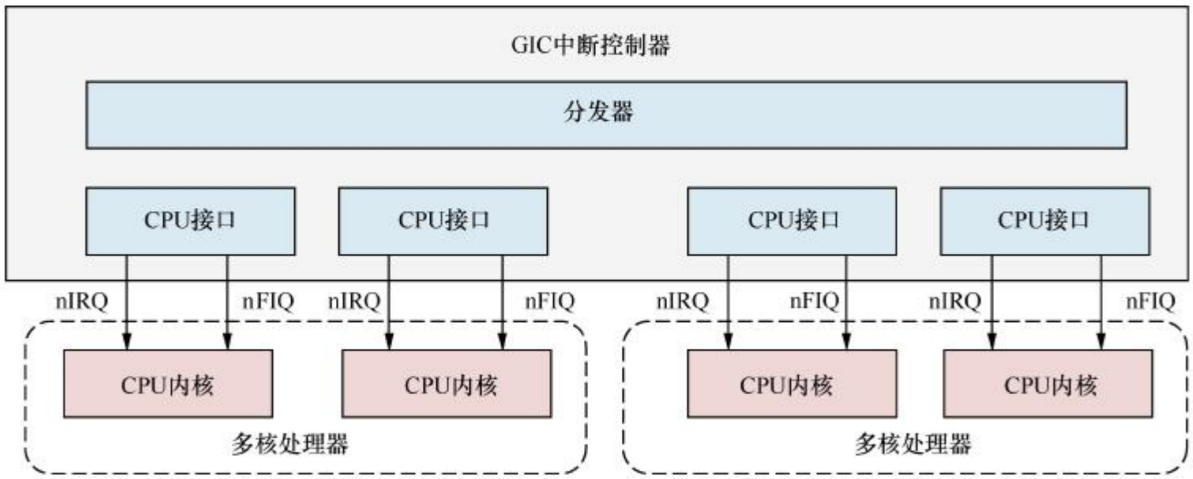
中 断 类 型	中断号范围
软件触发中断（SGI）	0～15
私有外设中断（PPI）	16～31
共享外设中断（SPI）	32～1019

SoC GIC-V2

- SGI SGI Linux Inter-Processor Interrupt IPI CPU
- PPI PPI CPU CPU local timer
- SPI

SGI PPI CPU SPI CPU

GIC distributor CPU



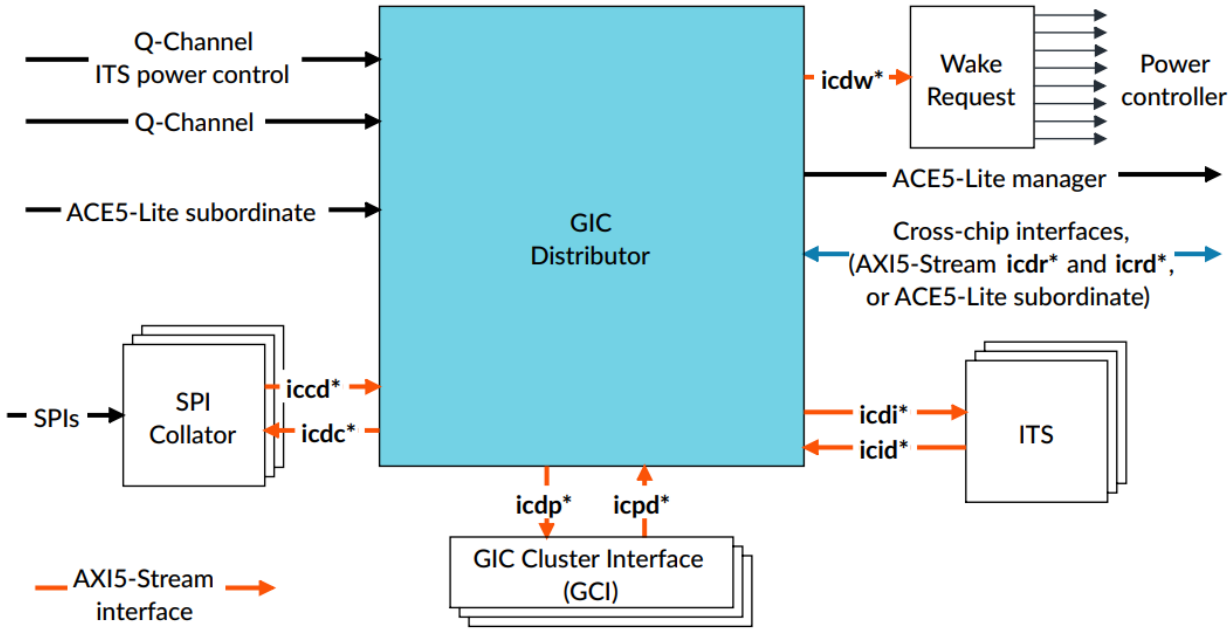
▲图 13.2 GIC-V2 的内部结构

GIC-V2 CPU CPU interface 13.2 CPU CPU CPU CPU nIRQ nFIQ CPU

GIC-700

GIC-700

Figure 3-1: GIC-700 Distributor



gic-v3 Distributor Redistributor CPU interface ITS Interrupt Translation Service

- Distributor SPI priority affinity trigger-type sgi spi Redistributor
- Redistributor PPI LPI priority affinity trigger-type PPI LPI CPU interface
- CPU interface Redistributor CPU CPU Distributor Redistributor
- ITS msi msi LPI Redistributor

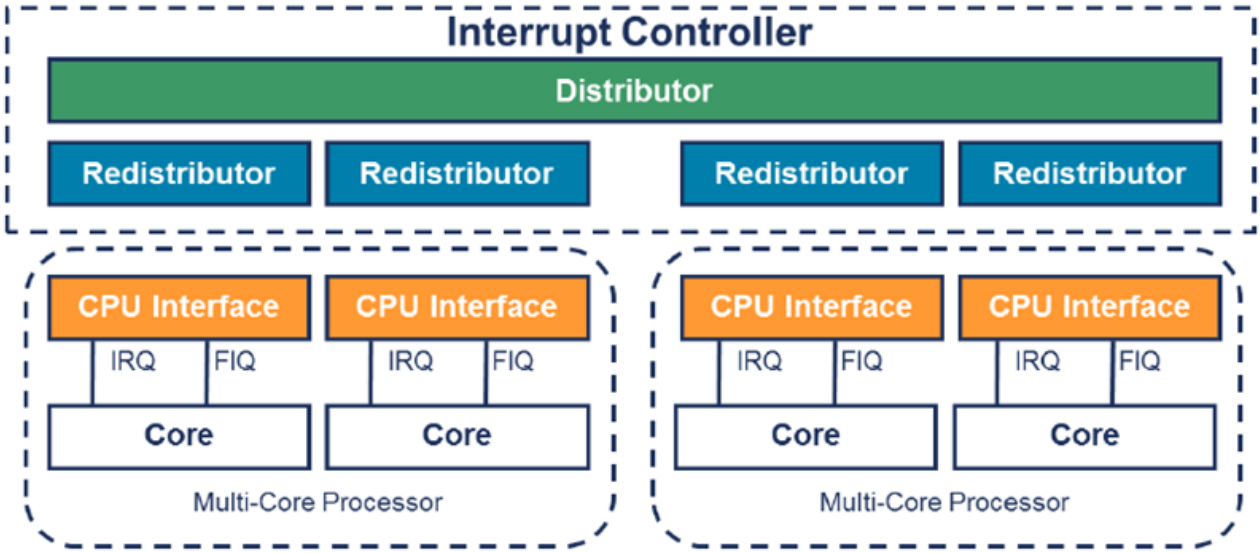
GIC	General Interrupt Controller
SGI	Software Generated Interrupt
PPI	Private Peripheral Interrupt
SPI	Shared Peripheral Interrupt
LPI	Locality-specific Peripheral Interrupt
PE	Process Element
ITS	Interrupt Translation Service

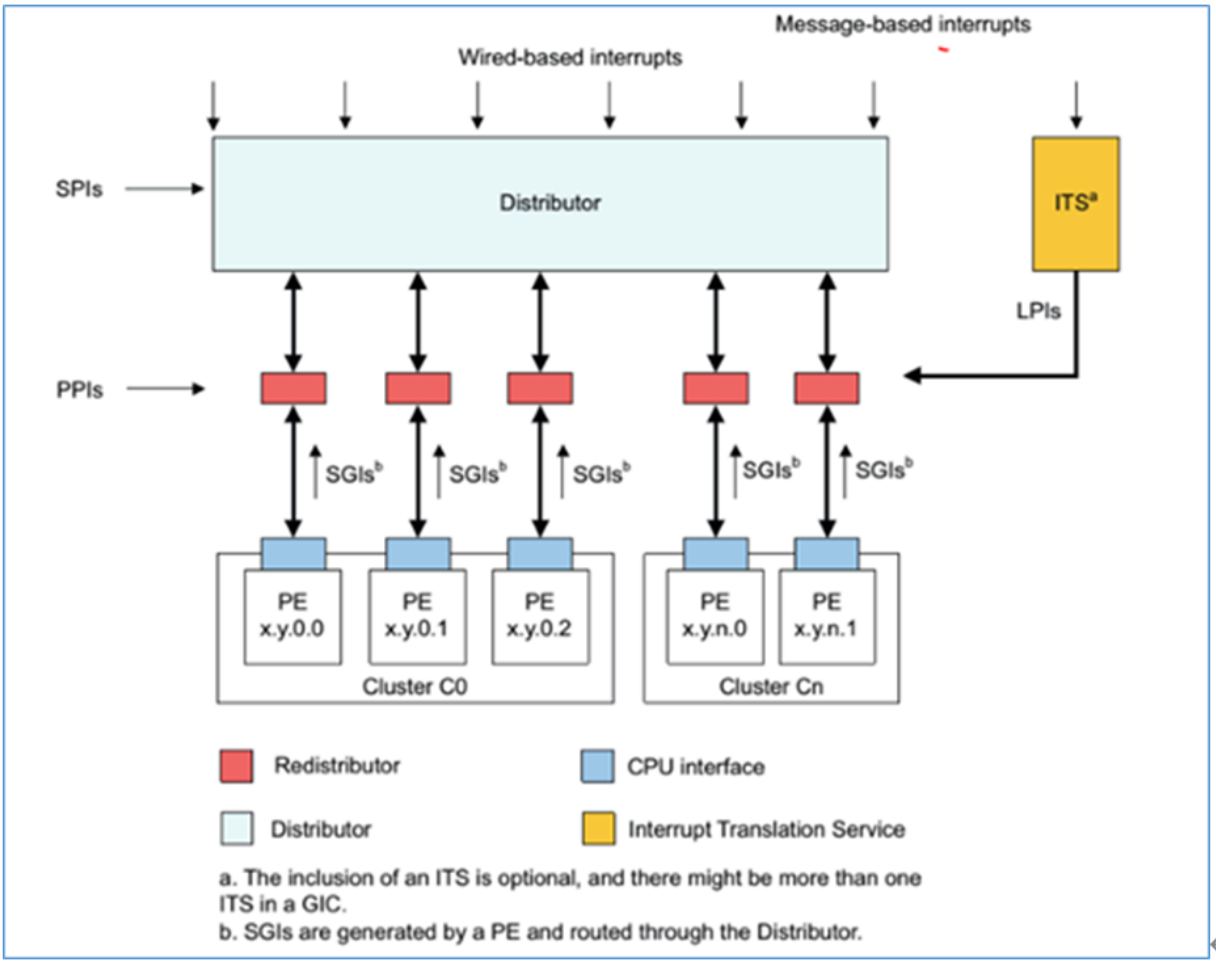
SGI	SGI security group PE PE / PE
PPI	PPI
SPI	PE PE / PE
LPI	LPI forwarding /ITS

linux kernel 5.10

SGI	0 - 15	PE
PPI	16 – 31	PE
SPI	32 - 1019	
	1020 - 1023	
	1024 - 1055	
PPI	1056 -1119	GICv3.1
	1120 - 4095	
SPI	4096 - 5119	GICv3.1
	5120 - 8191	
LPI	8192 -	

LPI 8192 LPI GICD_TYPER IDbits field LPI





GCI GIC Cluster Interface cluster Redistributor c2000 EXG cluster core 8 thread GCI 8 Redistributor c2k ICI ARM CPU interface

SGI core A CPU interface GICD threadde GICR interface thread

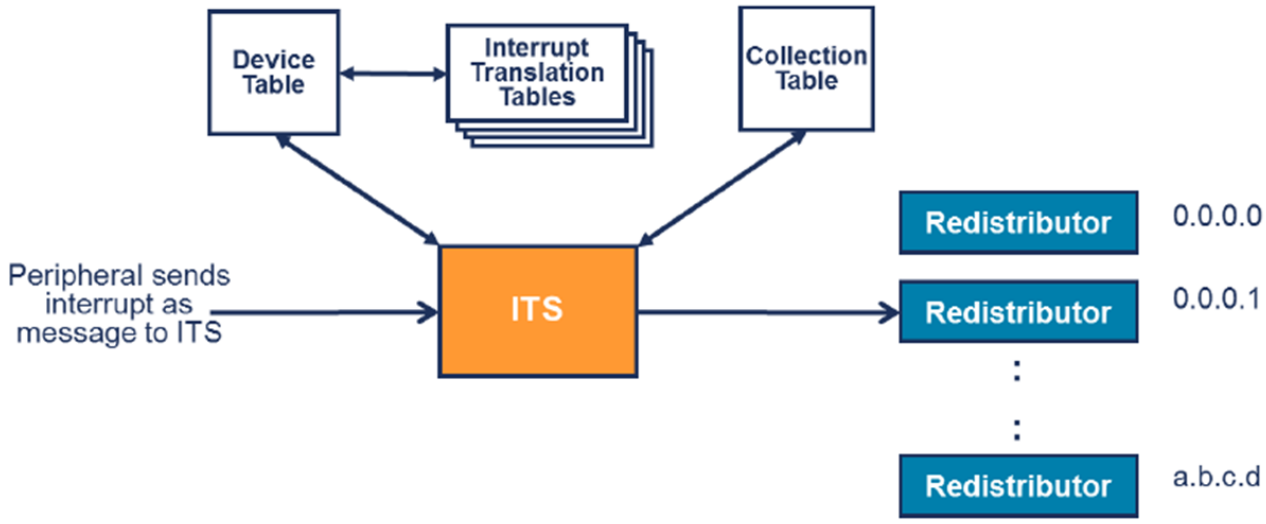
PPI PPI GICR interface thread

EXG timer

SPI SPI GICD GICR GICR interface thread

PCIe

LPI MSI ITS GICR GICR interface thread



1. ITS device id event id device id event id

2. ITS GICR CPU

a) Device table device ITT

b) ITT event id collection id

c) Collection table collection id GICR

3. ITS ITS ITS CMDQ

c2k MSI its->gicd->gicr arm its redistributor GICR distributor(GICD)

GICD ITS GICD GICD ddr GICR c2k arm

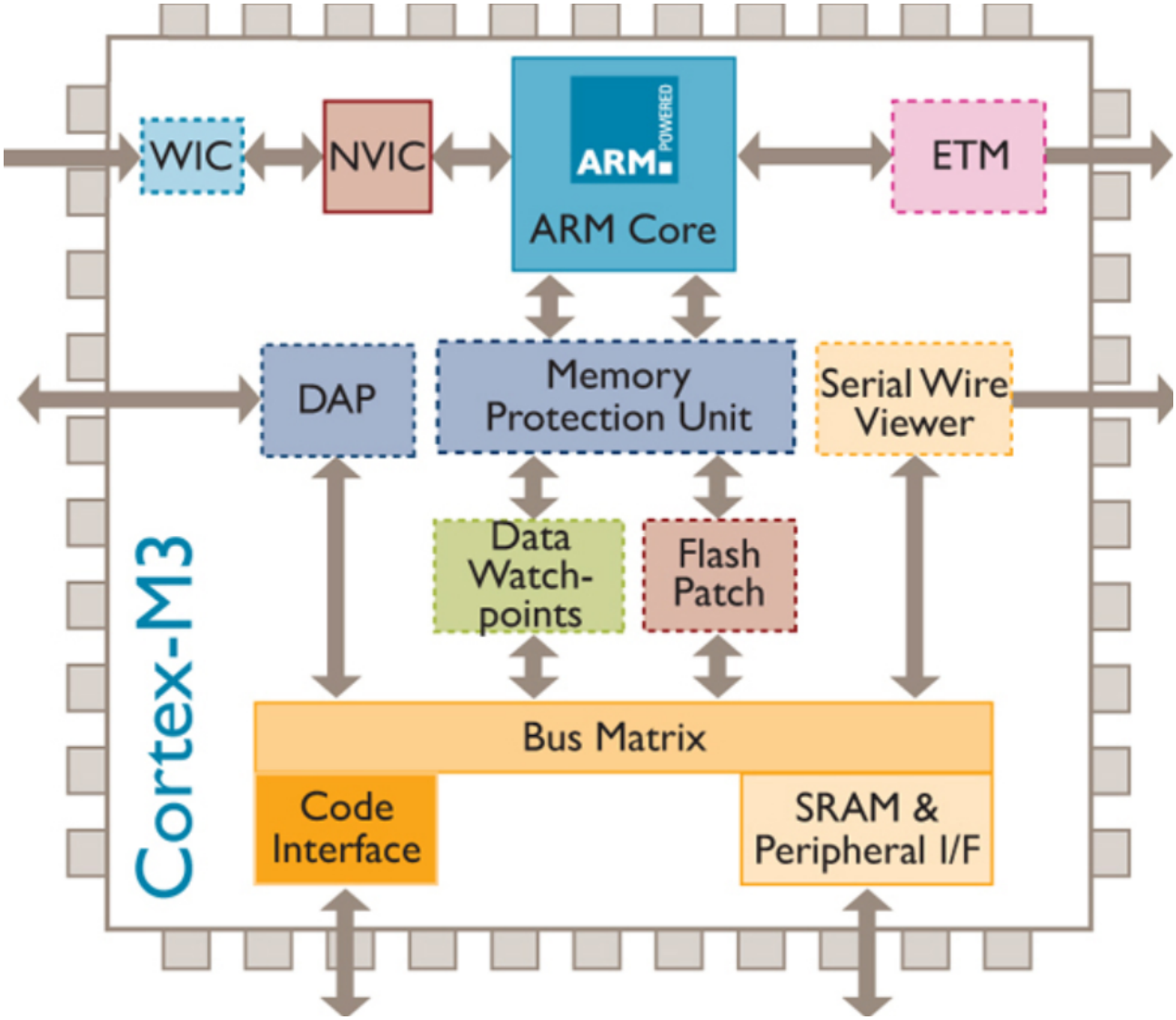
1.2.7 RISCV AIA

[toc]

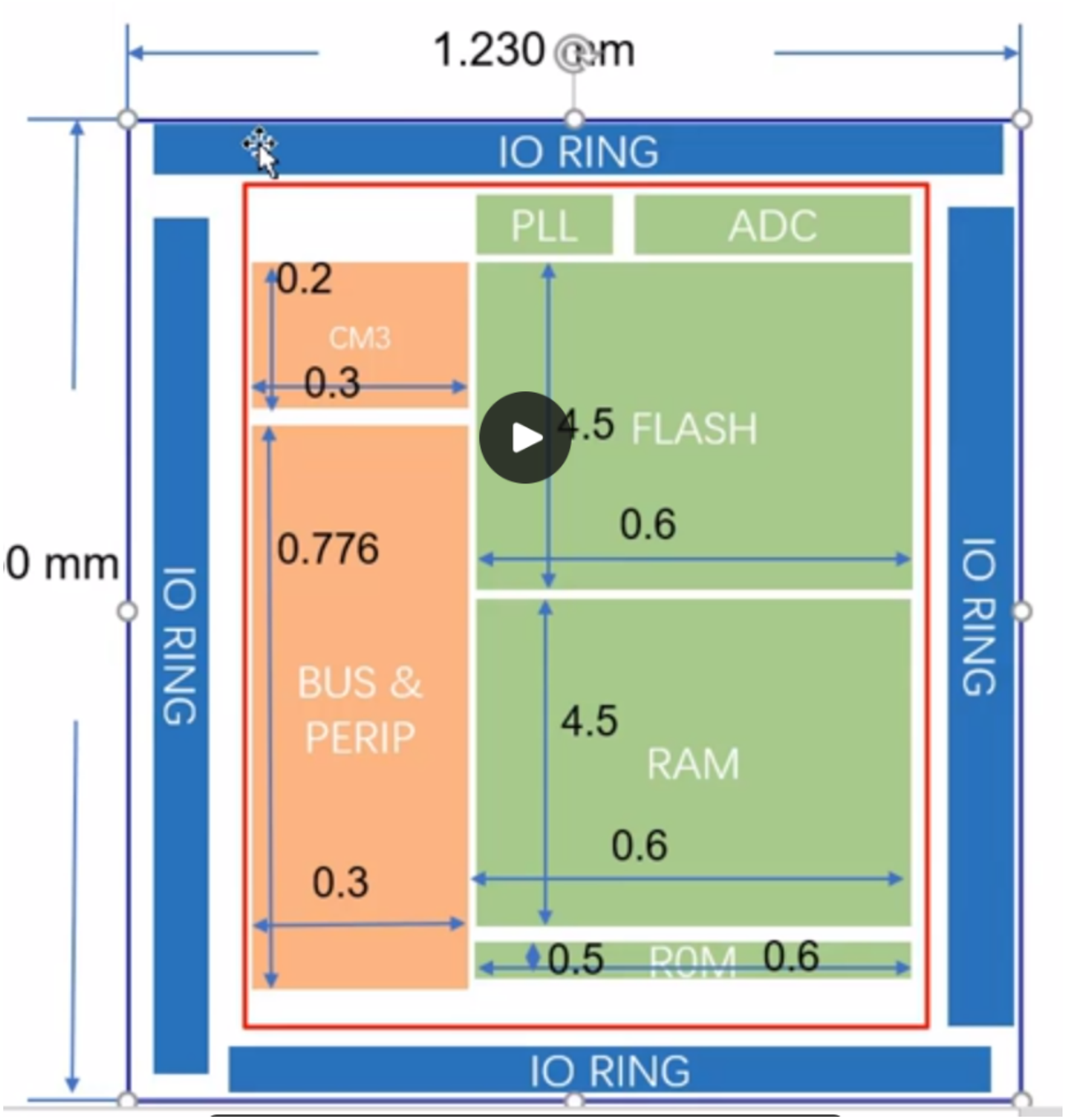
1.2.8 Cortex M3 MCU

[toc]

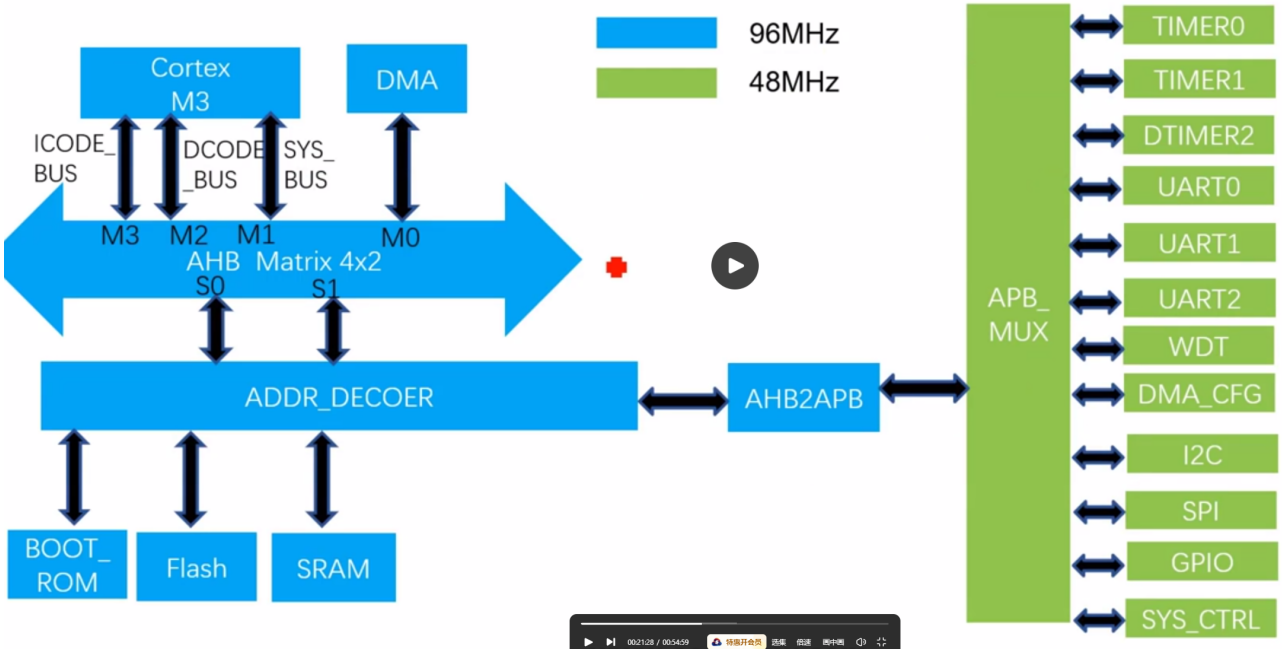
M3

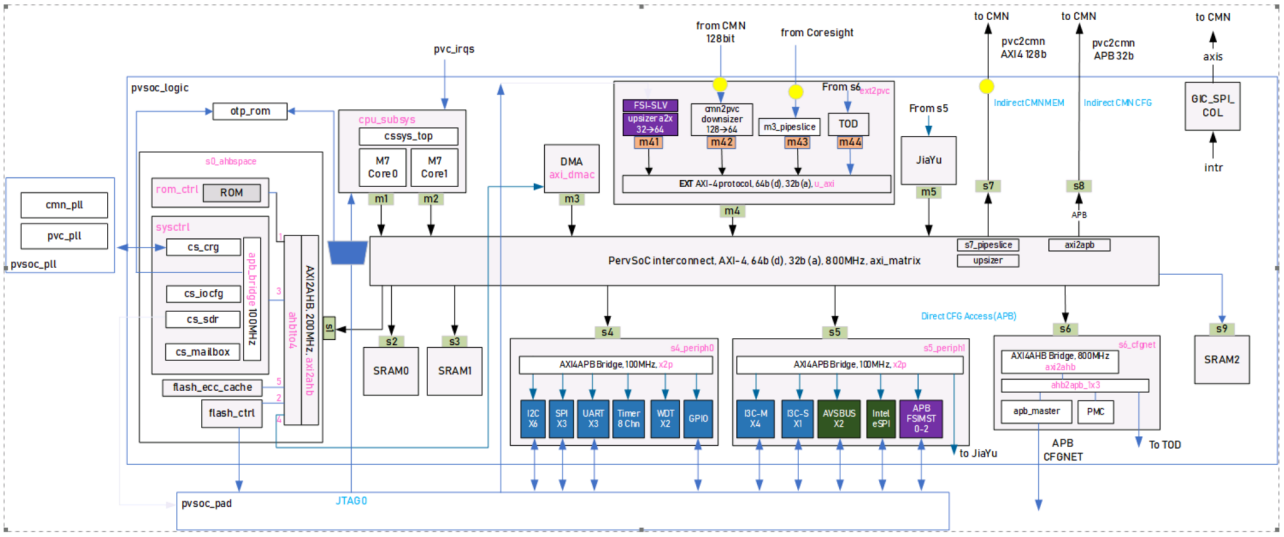

WIC
NVIC
ARM
POWERED
ARM Core
ETM
DAP
Memory
Protection Unit
Serial Wire
Viewer
Data
Watch-
points
Flash
Patch
Cortex-M3
Bus Matrix
Code
Interface
SRAM &
Peripheral I/F

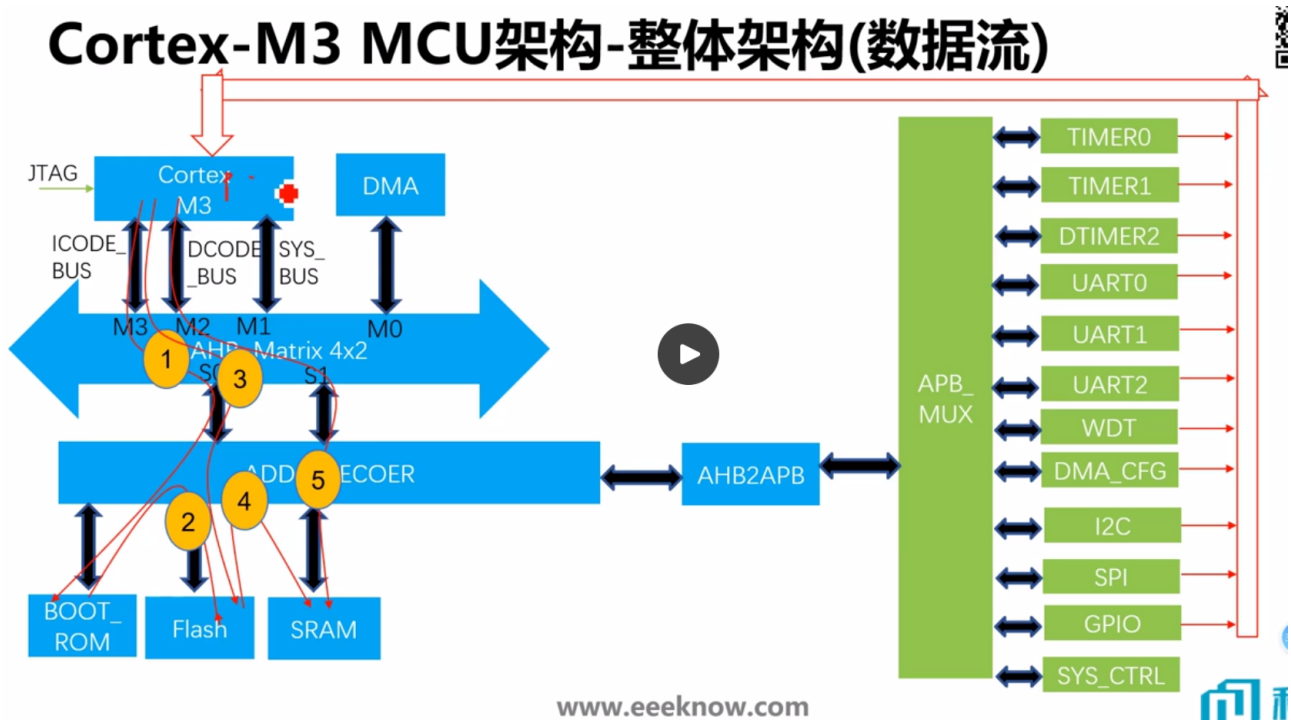
MCU


1.230 cm
IO RING
PLL
ADC
0.2
CM3
0.3
4.5
FLASH
0.6
0.776
BUS &
PERIP
4.5
RAM
0.6
0.3
0.5
ROM
0.6
IO RING
IO RING
IO RING
0 mm

Cortex-M3 MCU架构-整体架构







Cortex-M3 MCU架构-Memory Map

SUB	名字	结束地址	起始地址	大小（KBYTE）
存储系统	Program	0000FFFF	00000000	64
	Reserved	00FFFFFF	00010000	16320
	Boot Loader	0100FFFF	01000000	64
	Reserved	1FFFFFFF	01010000	507840
	RAM	2000FFFF	20000000	64
	Reserved	3FFFFFFF	20010000	524224
APB 系统	TIMER0	40000FFF	40000000	4
	TIMER1	40001FFF	40001000	4
	DTIMER	40002FFF	40002000	4
	SYS_CTRL	40003FFF	40003000	4
	UART 0	40004FFF	40004000	4
	UART 1	40005FFF	40005000	4
	UART 2	40006FFF	40006000	4
	SPI	40007FFF	40007000	4
	WDT	40008FFF	40008000	4
	I2C0	40009FFF	40009000	4
	I2C1	4000AFFF	4000A000	4
	APB_TEST	4000BFFF	4000B000	4
	ADC	4000CFFF	4000C000	4
	RTC	4000DFFF	4000D000	4
	Reserved	4000EFFF	4000E000	4
	DMA	4000FFFF	4000F000	4
保留	Reserved	40010FFF	40010000	4
	Reserved	40011FFF	40011000	4
	Reserved	4001EFFF	40012000	52
	Reserved	4001FFFF	4001F000	4
CM3	Cortex3	FFFFFFFF	40020000	3145600

1.3 ARM N2

ARM N2 ARM Neoverse N2 reference design ARM Neoverse N2 ARM CPU

1. N2 HPC

2. Neoverse N2
3. ARM N2
4. N2
5. Neoverse N2 ARM ARM

ARM N2

ARM Neoverse N2 ARM IP IP

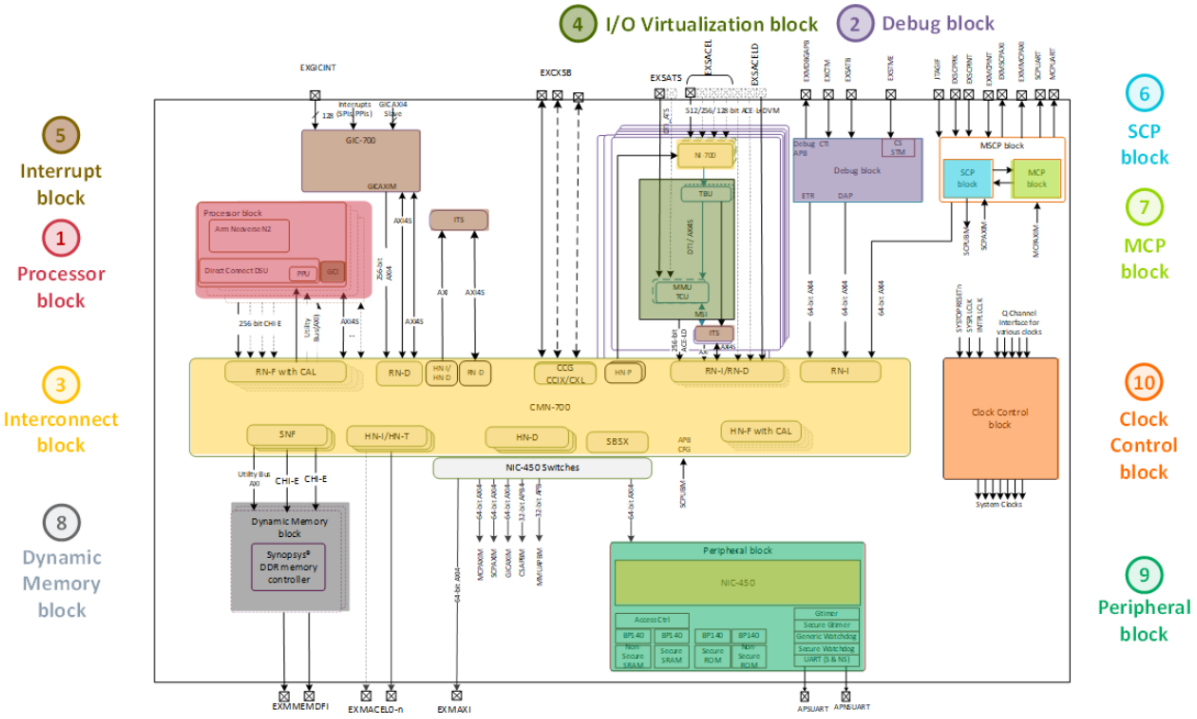
1. **ARM Neoverse N2 CPU** N2 N2 ARMv9
2. **System Level Cache (SLC)** ARM N2 SLC
3. **CCIX CXL** N2 CCIX Cache Coherent Interconnect for Accelerators CXL Compute Express Link
4. **AMBA (Advanced Microcontroller Bus Architecture)** AMBA SoC IP N2 AMBA
5. **TrustZone** ARM TrustZone
6. **Mali GPU** N2 CPU ARM Mali GPU
7. **SMMU (System Memory Management Unit)**

IP ARM Neoverse N2

High Level

Arm Neoverse N2 reference design Technical Overview

Figure 3-1: RD-N2 architecture block diagram



CMN-700 PICe SMMU NI-700 RNI SCP GIC SNF memory controller core

1.4

1.4.1 ARM

core

1. BMC C2000 CHIP POR_N pervsoc
2. otprom valid SCP SRAM BISR PVO(SCP) ROM otprom
3. SCP SRAM
4. SCP ROM FW GPIO CHIP ID, CHIP ID CHIP;
5. I2C/FSI
6. CHIP SCP QSPI;
7. CHIP SCP CHIP CHIP
8. CHIP SCP CHIP SCP
9. CHIP SCP QSPI SCP RAM FW CHIP SRAM;
10. CHIP SCP 8 SCP RAM FW SCP, SCP SRAM:
11. CHIP SCP SRAM, SCP RAM FW:
12. CHIP SCP QSPI C2C SCP
13. CHIP SCP CHIP SCP C2C C2C;
14. SCP CHIP ID CMN OCM(SLC) SAM;
15. CHIP SCP MCP RAM FW, MCP( ) +
16. SCP CHIP L2 NCU SCOM/FIR;+
17. CHIP SCP Hostboot base CHIP OCM(SLC);+
18. CHIP SCP CHIP
19. srest 0x100

core-hostboot

1. kernel other thread spinlock, thread
2. hostboot kernel C/C++ main +
3. hostboot kernel
4. MMU(SLB/TLB), DIMM OCM(SLC, 28 * 2MB=56MB 3/4
5. TB CPU
6. Scratch FSP/BMC;
7. CPU init main
8. kernel other thread spinlock 1 thread
9. hostboot task task user kernel hypersior
10. init main VFS;
11. VFS base modules

init

12. initservice

13. initservice g_taskinfolist

14. secureboot_base/ipmibase/pnor/vfs/extinitsvc

15. hostboot extended image

16. extinitsvc g_exttaskinfolist

17. istepdisp hostboot

18. istep hostboot

19. istep 21.03 payload(skiboot,OPAL);+

20. payload(skiboot,OPAL)

core-skiboot

1. C2000 skiboot

2. thread skiboot relocate thread

3. thread boot flag=1, thread

4. thread main cpu entry, thread secondary cpu entry;

5. uart(console);

6. skiboot /

7. skiboot

8. opal call table;

9. hostboot device tree HDAT;

10. DT SCOM/eSPI(LPC):

11. DT C2000 DIMM

12. DT homer C2000

13. DT CPU(thread);

14. opal DT;

15.

16. BMC ipmi/BT/UART;

17. GIC

18. CPU thread

19. sreset CPU

20. TOD TOD

21. sensor OPAL

22.

23.NV

24. jiayu

25. TPM

26. opal console(UART)

27. PCIE HB PCIE (RC CXL PCIe device tree)

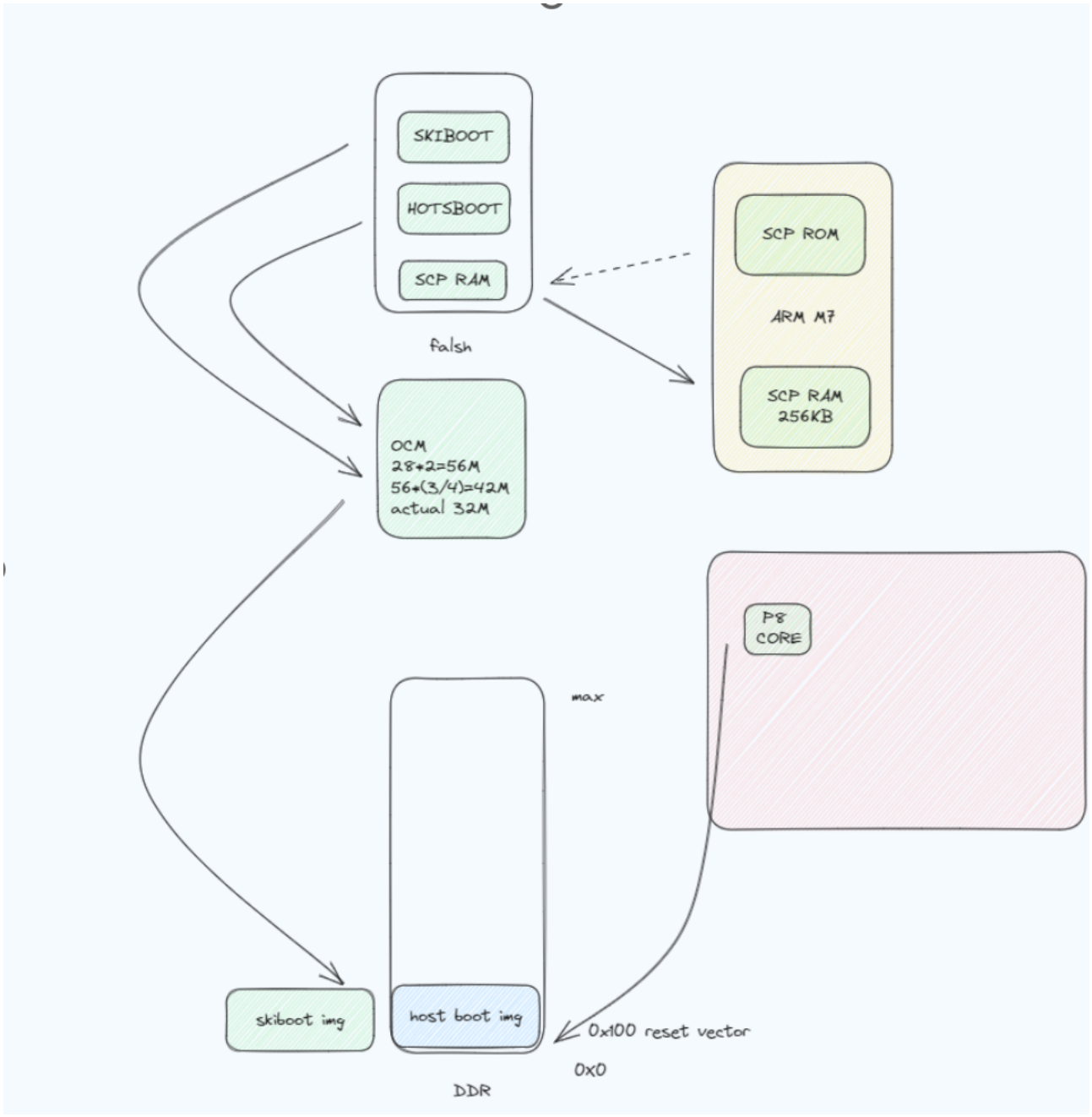
28.PCIE SLOT

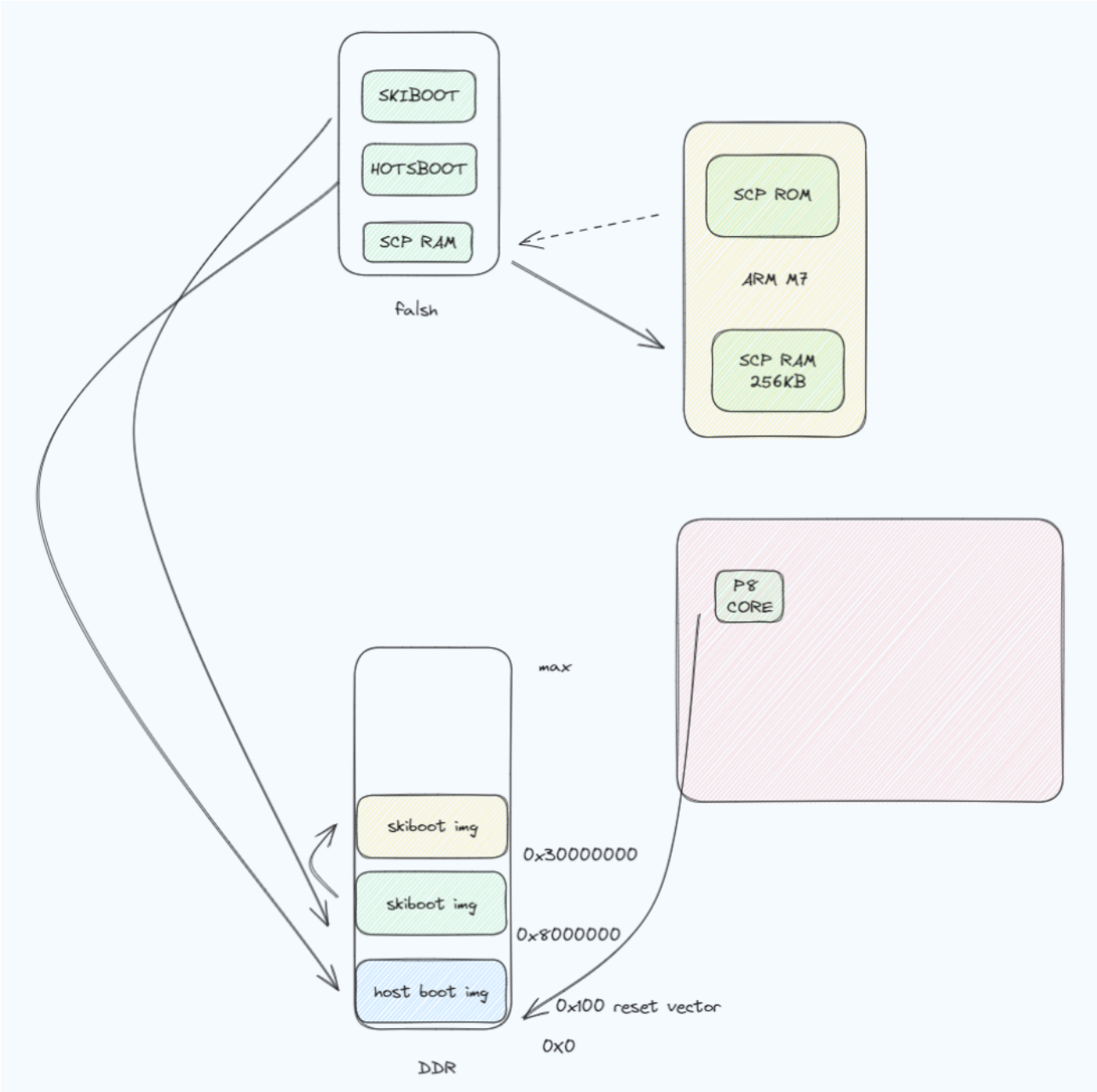
29.

30.

31. zImage(boot loader petitboot);

32. boot loader





CMN OCM(on chip memory) L3 cache DDR Intel cache as ram

OCM memory CMN OCM(on chip memory) 28*2=56M 3/4 32M

SCP ram DDR training OCM flash DDR

OCM SNF

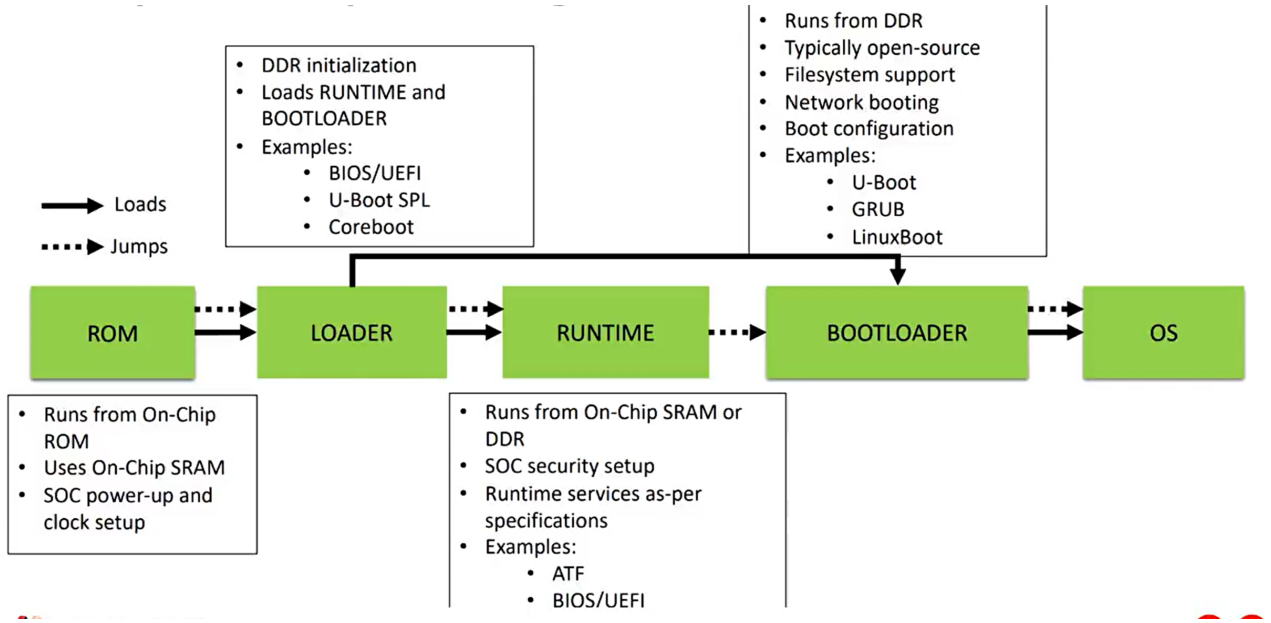
SCP OCM SCP RAM hostboot OCM skiboot OCM skiboot ddr training DDR OCM OCM cache

hostboot skiboot 0x800 0000

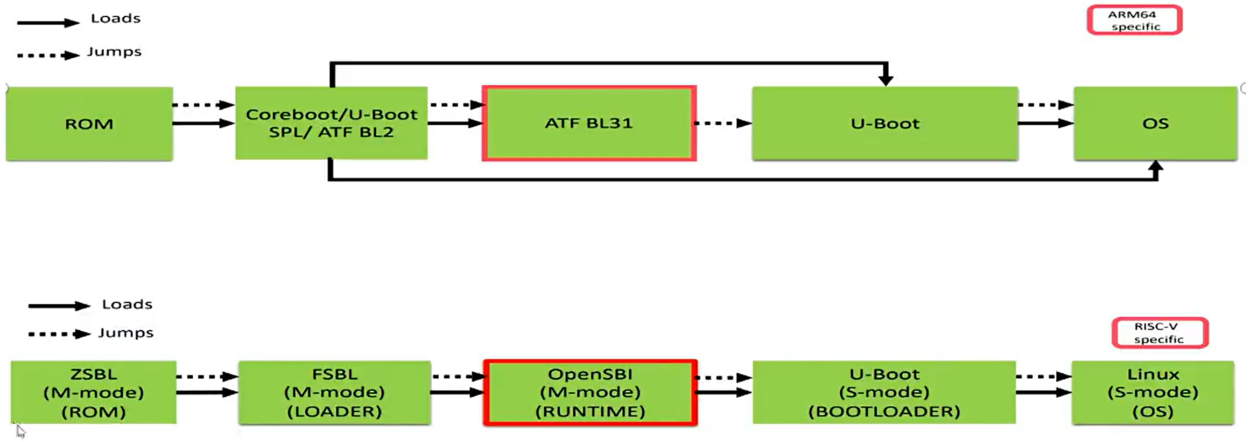
32M 0x200 0000, skiboot 0x3000 0000 OCM 0x0 0x0

1.4.2 RISC-V Firmware

Firmware

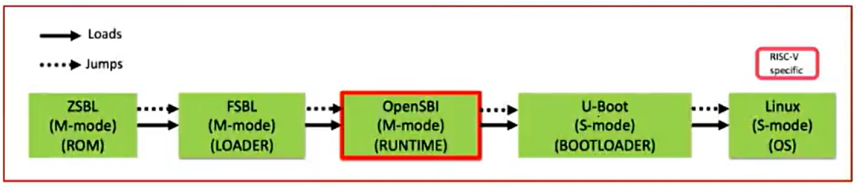


ARM64 vs RISC-V



uboot EDK2 PEI DXE

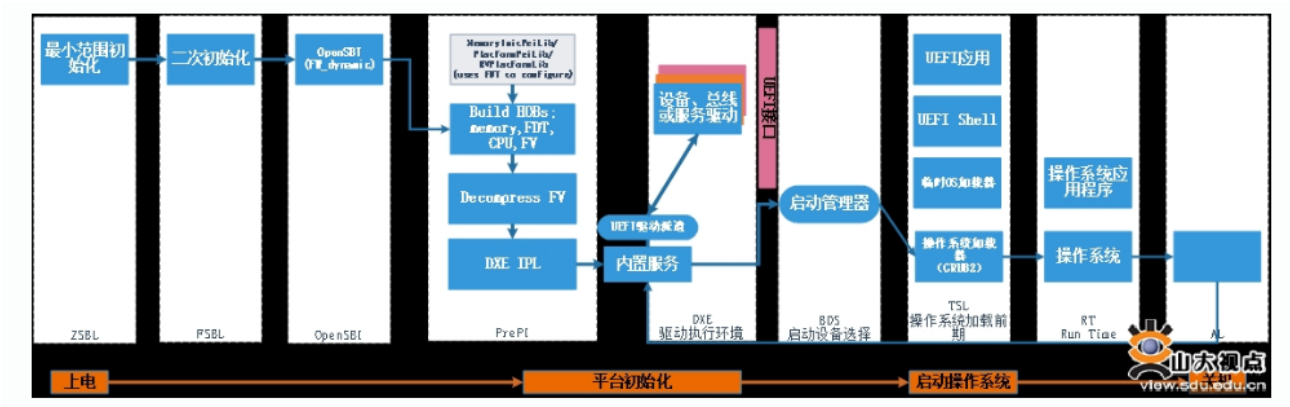
RISC-V linux启动流程如图，启动流程：ZSBL-->FSBL-->OpenSBI-->u-boot(UEFI)-->Linux



1. ZSBL（第0阶段Bootloader）at 0x00010000 （ROM in SOC）
2. FSBL（第1阶段Bootloader）
3. OpenSBI (runtime servics) M模式
4. U-boot/UEFI (bootloader)，
5. Linux启动linux需要以下文件：
   a. /boot/extlinux/extlinux.conf:包含启动菜单、默认启动等
   b. Image：Linux镜像
   c. Initrd:Linux执行需要的ramdisk，执行其中的init文件并进行进一步处理
   d. 设备树(.dtb)（可选）:未指定时会使用默认的dtb

ZSBL: SCP ROM

FSBL: SCP RAM



1.4.3 Cache as RAM and On Chip Memory

x86 Cache as RAM, CAR;

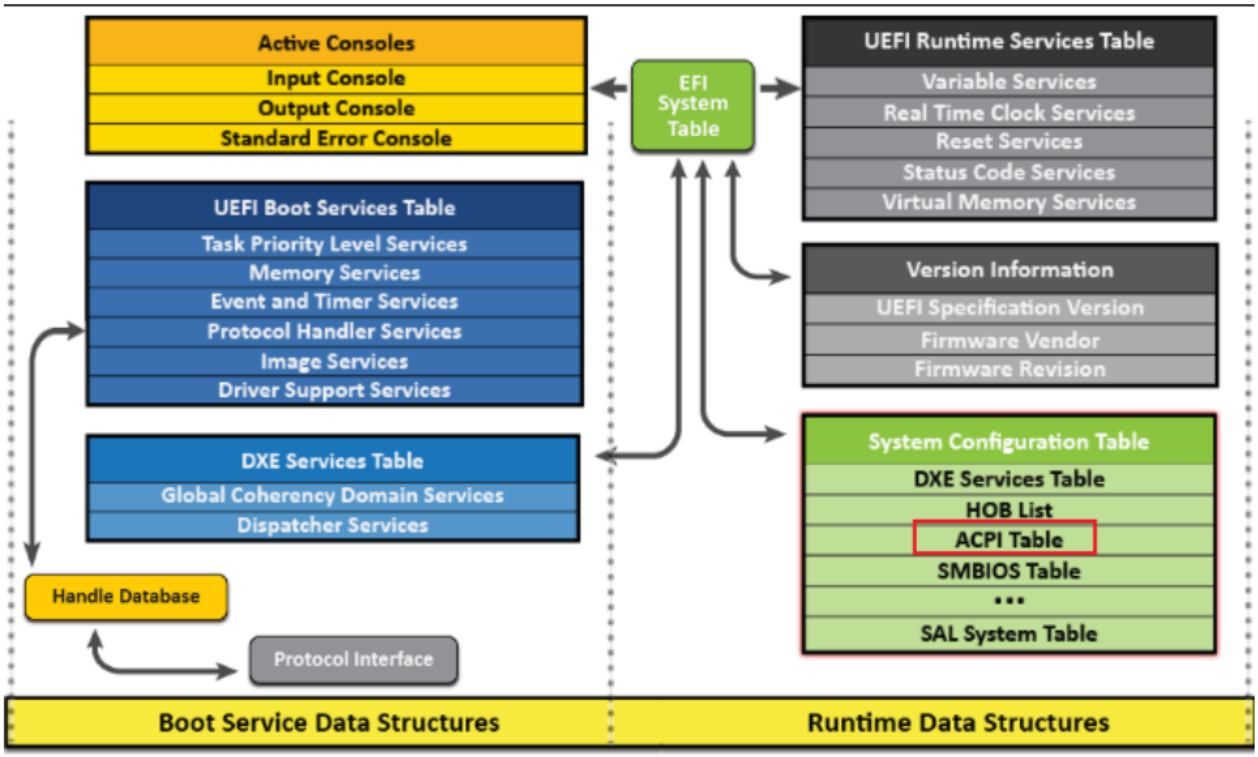
ARM On Chip Memory OCM

DDR cache RAM C

CAR

OCM

1.4.4 System Table



MdeModulePkg\Core\Dxe\DxeMain\DxeMain.c

```
  //
  // Allocate the EFI System Table and EFI Runtime Service Table from EfiRuntimeServicesData
  // Use the templates to initialize the contents of the EFI System Table and EFI Runtime
    ↪ Services Table
  //
  gDxeCoreST = AllocateRuntimeCopyPool (sizeof (EFI_SYSTEM_TABLE),
↪   &mEfiSystemTableTemplate);
  ASSERT (gDxeCoreST != NULL);

  gDxeCoreRT = AllocateRuntimeCopyPool (sizeof (EFI_RUNTIME_SERVICES),
↪   &mEfiRuntimeServicesTableTemplate);
  ASSERT (gDxeCoreRT != NULL);

  gDxeCoreST->RuntimeServices = gDxeCoreRT;


///
/// EFI System Table
///
typedef struct {
  ///
  /// The table header for the EFI System Table.
  ///
  EFI_TABLE_HEADER                  Hdr;
  ///
  /// A pointer to a null terminated string that identifies the vendor
```

```
/// that produces the system firmware for the platform.
///
CHAR16                            *FirmwareVendor;
///
/// A firmware vendor specific value that identifies the revision
/// of the system firmware for the platform.
///
UINT32                            FirmwareRevision;
///
/// The handle for the active console input device. This handle must support
/// EFI_SIMPLE_TEXT_INPUT_PROTOCOL and EFI_SIMPLE_TEXT_INPUT_EX_PROTOCOL.
///
EFI_HANDLE                        ConsoleInHandle;
///
/// A pointer to the EFI_SIMPLE_TEXT_INPUT_PROTOCOL interface that is
/// associated with ConsoleInHandle.
///
EFI_SIMPLE_TEXT_INPUT_PROTOCOL    *ConIn;
///
/// The handle for the active console output device.
///
EFI_HANDLE                        ConsoleOutHandle;
///
/// A pointer to the EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL interface
/// that is associated with ConsoleOutHandle.
///
EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL   *ConOut;
///
/// The handle for the active standard error console device.
/// This handle must support the EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL.
///
EFI_HANDLE                        StandardErrorHandle;
///
/// A pointer to the EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL interface
/// that is associated with StandardErrorHandle.
///
EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL   *StdErr;
///
/// A pointer to the EFI Runtime Services Table.
///
EFI_RUNTIME_SERVICES              *RuntimeServices;
///
/// A pointer to the EFI Boot Services Table.
///
EFI_BOOT_SERVICES                 *BootServices;
///
/// The number of system configuration tables in the buffer ConfigurationTable.
///
UINTN                             NumberOfTableEntries;
```

```
  ///
  /// A pointer to the system configuration tables.
  /// The number of entries in the table is NumberOfTableEntries.
  ///
  EFI_CONFIGURATION_TABLE           *ConfigurationTable;
} EFI_SYSTEM_TABLE;
```

1.4.5 EDK2 EVENT

[toc]

UEFI

/ intel 8253/8254 PC IC MSC8051,

Timer cpu

PC/AT UEFI 8254 Timer mode 3 1.1931816MHz tick mTimerPeriod 1ms 10000 tick =10000*100ns pin 8259 IRQ0 CPU Timer

cpu archprotocol Legacy8259Protocol , cpu IDT Timer TimerInterruptHandler

Timer

Timer

1.

2.

3.

4. Timer CoreTimerTick(mTimerPeriod)/CoreTimerTick(100ms) TimerArchprotocol install EVT_NOTI

5. Sytem time mEfiSystemTime =mEfiSystemTime+ 1Tick(100ms)

6. timer event list event BS->SignalEvent signal

BS->SignalEvent--->CoreNotifyEvent-->CoreRestoreTpl-->CoreDispatchEventNotifies-->A. EVT_NOTIFY_SIGNAL >SignalCount = 0 Event->NotifyFunction (Event, Event->NotifyContext) Tpl gEventPending mask bit event

7.

8.

1. gEventQueue--- A list of event's to notify for each priority level

2. mEfiTimerList---- timer event

3. gEventSignalQueue --- A list of events to signal based on EventGroup type

4. EVT_NOTIFY_WAIT event BS-> **CoreCheckEvent**->CoreNotifyEvent-->CoreRestoreTpl-->CoreDispatchEventNotifies gEventQueue IEvent

gEventPending gEventQueue 1 0

gEventPending 32 1

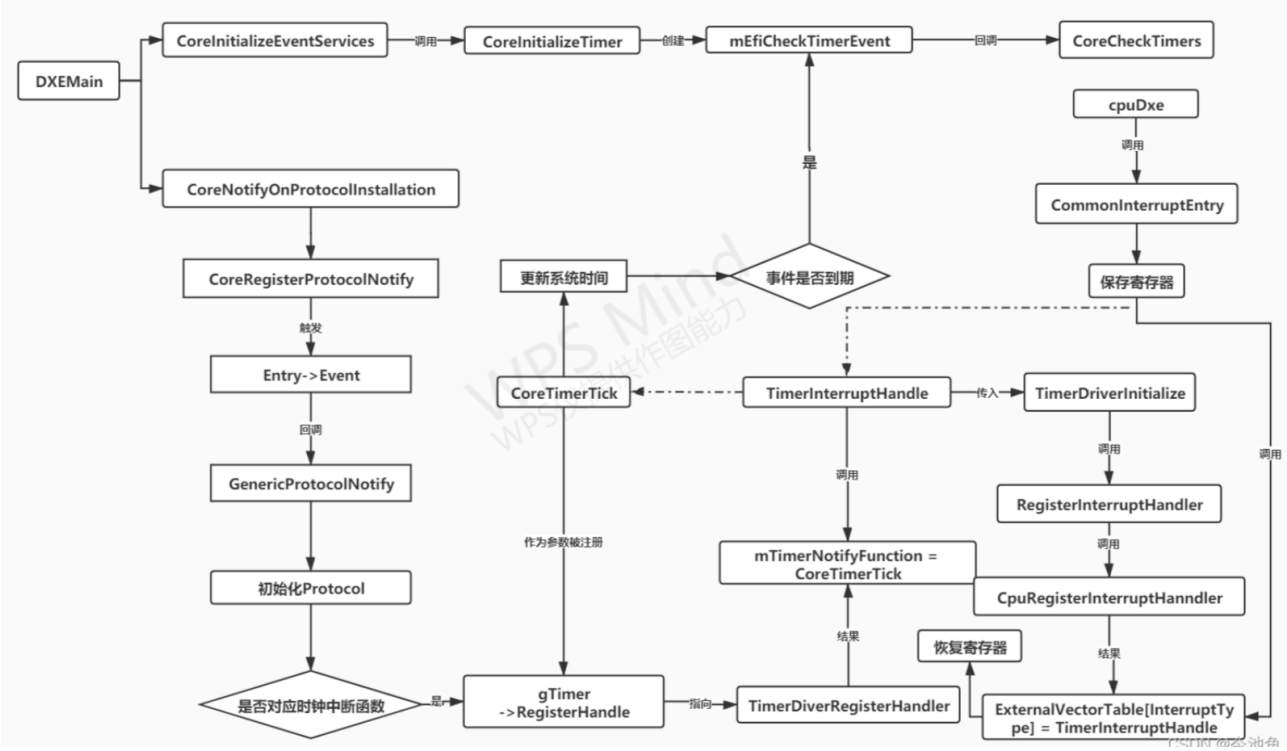
x86 IRQ

- IRQ0:
- IRQ1:
- IRQ2:
- IRQ3: 2 COM2
- IRQ4: 1 COM1
- IRQ5:
- IRQ6:
- IRQ7: 1 LPT1
- IRQ

event

```
//
// Task priority level
//
#define TPL_APPLICATION       4
#define TPL_CALLBACK          8
#define TPL_NOTIFY            16
#define TPL_HIGH_LEVEL        31
```

31



DXE main

```
//
// Initialize the Event Services
```

```
  //
  Status = CoreInitializeEventServices ();

...

  //
  // Register for the GUIDs of the Architectural Protocols, so the rest of the
  // EFI Boot Services and EFI Runtime Services tables can be filled in.
  // Also register for the GUIDs of optional protocols.
  //
  CoreNotifyOnProtocolInstallation ();
```

event

```
/**
  Called by the platform code to process a tick.

  @param  Duration              The number of 100ns elapsed since the last call
                                to TimerTick

**/
VOID
EFIAPI
CoreTimerTick (
  IN UINT64   Duration
  )
{
  IEVENT          *Event;

  //
  // Check runtiem flag in case there are ticks while exiting boot services
  //
  CoreAcquireLock (&mEfiSystemTimeLock);

  //
  // Update the system time
  //
  mEfiSystemTime += Duration;

  //
  // If the head of the list is expired, fire the timer event
  // to process it
  //
  if (!IsListEmpty (&mEfiTimerList)) {
    Event = CR (mEfiTimerList.ForwardLink, IEVENT, Timer.Link, EVENT_SIGNATURE);

    if (Event->Timer.TriggerTime <= mEfiSystemTime) {
      CoreSignalEvent (mEfiCheckTimerEvent);
```

```
    }
  }

  CoreReleaseLock (&mEfiSystemTimeLock);
}
```

timer event

```
/**
  Initializes "event" support.

  @retval EFI_SUCCESS            Always return success

**/
EFI_STATUS
CoreInitializeEventServices (
  VOID
  )
{
  UINTN        Index;

  for (Index=0; Index <= TPL_HIGH_LEVEL; Index++) {
    InitializeListHead (&gEventQueue[Index]);
  }

  CoreInitializeTimer ();

  CoreCreateEventEx (
    EVT_NOTIFY_SIGNAL,
    TPL_NOTIFY,
    EfiEventEmptyFunction,
    NULL,
    &gIdleLoopEventGuid,
    &gIdleLoopEvent
    );

  return EFI_SUCCESS;
}
```

- gEventQueue--- A list of event's to notify for each priority level
- CoreInitializeTimer

```
/**
  Initializes timer support.

**/
VOID
CoreInitializeTimer (
  VOID
  )
{
```

```
  EFI_STATUS  Status;

  Status = CoreCreateEventInternal (
             EVT_NOTIFY_SIGNAL,
             TPL_HIGH_LEVEL - 1,
             CoreCheckTimers,
             NULL,
             NULL,
             &mEfiCheckTimerEvent
             );
  ASSERT_EFI_ERROR (Status);
}
```

```
#include <Uefi.h>
#include <Library/UefiBootServicesTableLib.h>
#include <Library/UefiLib.h>

EFI_EVENT TimerEvent;
EFI_EVENT ExitBootServicesEvent;
BOOLEAN TimerTriggered = FALSE;

VOID EFIAPI TimerEventHandler(
    IN EFI_EVENT Event,
    IN VOID *Context
) {
    TimerTriggered = TRUE;
    Print(L"Timer event triggered!\n");
}

EFI_STATUS
EFIAPI
UefiMain(
    IN EFI_HANDLE ImageHandle,
    IN EFI_SYSTEM_TABLE *SystemTable
) {
    EFI_STATUS Status;

    //         10000000 100ns (1 )
    Status = gBS->CreateEvent(
        EVT_TIMER | EVT_NOTIFY_SIGNAL,
        TPL_CALLBACK,
        TimerEventHandler,
        NULL,
        &TimerEvent
    );
    if (EFI_ERROR(Status)) {
```

```
        Print(L"Failed to create event: %r\n", Status);
        return Status;
    }

    //    1
    Status = gBS->SetTimer(
        TimerEvent,
        TimerRelative,
        10000000 // 100ns  10000000 = 1
    );
    if (EFI_ERROR(Status)) {
        Print(L"Failed to set timer: %r\n", Status);
        gBS->CloseEvent(TimerEvent);
        return Status;
    }

    //
    while (!TimerTriggered) {
        gBS->Stall(100000); //  100ms
    }

    //
    gBS->CloseEvent(TimerEvent);

    return EFI_SUCCESS;
}
```

1

gBS->CreateEvent

```
typedef struct {
    UINT32 Type;
    EFI_TPL NotifyTpl;
    EFI_EVENT_NOTIFY NotifyFunction;
    VOID *NotifyContext;
    BOOLEAN Triggered;
} EFI_EVENT_INTERNAL;

EFI_STATUS
EFIAPI
CreateEvent (
    IN UINT32 Type,
    IN EFI_TPL NotifyTpl,
    IN EFI_EVENT_NOTIFY NotifyFunction,
    IN VOID *NotifyContext,
    OUT EFI_EVENT *Event
) {
    EFI_STATUS Status;
    EFI_EVENT_INTERNAL *NewEvent;
```

```
    //
    if (Event == NULL || (Type & EVT_NOTIFY_SIGNAL && NotifyFunction == NULL)) {
        return EFI_INVALID_PARAMETER;
    }

    //
    NewEvent = AllocateZeroPool(sizeof(EFI_EVENT_INTERNAL));
    if (NewEvent == NULL) {
        return EFI_OUT_OF_RESOURCES;
    }

    //
    NewEvent->Type = Type;
    NewEvent->NotifyTpl = NotifyTpl;
    NewEvent->NotifyFunction = NotifyFunction;
    NewEvent->NotifyContext = NotifyContext;
    NewEvent->Triggered = FALSE;

    //
    Status = RegisterEvent(NewEvent);
    if (EFI_ERROR(Status)) {
        FreePool(NewEvent);
        return Status;
    }

    //
    *Event = (EFI_EVENT)NewEvent;

    return EFI_SUCCESS;
}
```

event

event

```
  if ((Type & EVT_NOTIFY_SIGNAL) != 0x00000000) {
    //
    // The Event's NotifyFunction must be queued whenever the event is signaled
    //
    InsertHeadList (&gEventSignalQueue, &IEvent->SignalLink);
  }
```

32 bitmap 1

SignalEvent event 32

```
/**
  Signals the event.  Queues the event to be notified if needed.

  @param  UserEvent               The event to signal .
```

```
  @retval EFI_INVALID_PARAMETER  Parameters are not valid.
  @retval EFI_SUCCESS            The event was signaled.

**/
EFI_STATUS
EFIAPI
CoreSignalEvent (
  IN EFI_EVENT    UserEvent
  )
/**
  Queues the event's notification function to fire.

  @param  Event                  The Event to notify

**/
VOID
CoreNotifyEvent (
  IN  IEVENT      *Event
  )
{

  //
  // Event database must be locked
  //
  ASSERT_LOCKED (&gEventQueueLock);

  //
  // If the event is queued somewhere, remove it
  //

  if (Event->NotifyLink.ForwardLink != NULL) {
    RemoveEntryList (&Event->NotifyLink);
    Event->NotifyLink.ForwardLink = NULL;
  }

  //
  // Queue the event to the pending notification list
  //

  InsertTailList (&gEventQueue[Event->NotifyTpl], &Event->NotifyLink);
  gEventPending |= (UINTN)(1 << Event->NotifyTpl);
}
```

Duang CoreTimerTick TPL restore TPL

timer event mEfiTimerList

TPL

```
  while (gEventPending != 0)
```

```
{
    PendingTpl = (UINTN) HighBitSet64 (gEventPending);
    if (PendingTpl <= NewTpl) {
      break;
    }
    gEfiCurrentTpl = PendingTpl;
    if (gEfiCurrentTpl < TPL_HIGH_LEVEL) {
      CoreSetInterruptState (TRUE);
    }
    CoreDispatchEventNotifies (gEfiCurrentTpl);
    // gEventQueue [gEfiCurrentTpl]   event Notification
  }
```

gEventPending 32 1

1.

2. TPL TPL 16 17 16 TPL 17

3. pending pending TPL 16 15 16

——

waitforevent 4 10ms 31 RestoreTpl 4, 31 4 30 5 4

CoreTimerTick

```
/**
  Lowers the task priority to the previous value.   If the new
  priority unmasks events at a higher priority, they are dispatched.

  @param  NewTpl  New, lower, task priority

**/
VOID
EFIAPI
CoreRestoreTpl (
  IN EFI_TPL NewTpl
  )


{
  EFI_TPL     OldTpl;
  EFI_TPL     PendingTpl;
```

- OldTpl:
- PendingTpl:

```
  OldTpl = gEfiCurrentTpl;
  if (NewTpl > OldTpl) {
    DEBUG ((EFI_D_ERROR, "FATAL ERROR - RestoreTpl with NewTpl(0x%x) > OldTpl(0x%x)\n",
 ↪  NewTpl, OldTpl));
```

```
    ASSERT (FALSE);
  }
  ASSERT (VALID_TPL (NewTpl));
```

- OldTpl = gEfiCurrentTpl:
- NewTpl NewTpl OldTpl

```
  if (OldTpl >= TPL_HIGH_LEVEL  &&  NewTpl < TPL_HIGH_LEVEL) {
    gEfiCurrentTpl = TPL_HIGH_LEVEL;
  }
```

- TPL_HIGH_LEVEL TPL_HIGH_LEVEL TPL_HIGH_LEVEL

```
  while (gEventPending != 0) {
    PendingTpl = (UINTN) HighBitSet64 (gEventPending);
    if (PendingTpl <= NewTpl) {
      break;
    }

    gEfiCurrentTpl = PendingTpl;
    if (gEfiCurrentTpl < TPL_HIGH_LEVEL) {
      CoreSetInterruptState (TRUE);
    }
    CoreDispatchEventNotifies (gEfiCurrentTpl);
  }
```

- while (gEventPending != 0):
- PendingTpl = (UINTN) HighBitSet64 (gEventPending):
- PendingTpl NewTpl
- PendingTpl TPL_HIGH_LEVEL
- CoreDispatchEventNotifies

```
  gEfiCurrentTpl = NewTpl;

  if (gEfiCurrentTpl < TPL_HIGH_LEVEL) {
    CoreSetInterruptState (TRUE);
  }

}
```

- NewTpl
- TPL_HIGH_LEVEL

CoreDispatchEventNotifies

```
/**
  Dispatches all pending events.

  @param  Priority               The task priority level of event notifications
```

```
                                to dispatch

**/
VOID
CoreDispatchEventNotifies (
  IN EFI_TPL      Priority
  )


{
  IEVENT          *Event;
  LIST_ENTRY      *Head;
```

- Event:
- Head:

```
  CoreAcquireEventLock ();
  ASSERT (gEventQueueLock.OwnerTpl == Priority);
  Head = &gEventQueue[Priority];
```

- CoreAcquireEventLock ():
- ASSERT (gEventQueueLock.OwnerTpl == Priority):
- Head = &gEventQueue[Priority]:

```
  while (!IsListEmpty (Head)) {

    Event = CR (Head->ForwardLink, IEVENT, NotifyLink, EVENT_SIGNATURE);
    RemoveEntryList (&Event->NotifyLink);

    Event->NotifyLink.ForwardLink = NULL;
```

- while (!IsListEmpty (Head)):
- Event = CR (Head->ForwardLink, IEVENT, NotifyLink, EVENT_SIGNATURE):
- RemoveEntryList (&Event->NotifyLink):
- Event->NotifyLink.ForwardLink = NULL:

SIGNAL

```
    if ((Event->Type & EVT_NOTIFY_SIGNAL) != 0) {
      Event->SignalCount = 0;
    }
```

- EVT_NOTIFY_SIGNAL SignalCount

```
    CoreReleaseEventLock ();

    ASSERT (Event->NotifyFunction != NULL);
    Event->NotifyFunction (Event, Event->NotifyContext);
```

- CoreReleaseEventLock ():
- ASSERT (Event->NotifyFunction != NULL):

- `Event->NotifyFunction (Event, Event->NotifyContext)`:

```
  CoreAcquireEventLock ();
}
```

- `CoreAcquireEventLock ()`:

```
  gEventPending &= ~(UINTN)(1 << Priority);
  CoreReleaseEventLock ();
}
```

- `gEventPending &= ~(UINTN)(1 << Priority)`:
- `CoreReleaseEventLock ()`:

- 31TPL
-

Blog

Blog

1.4.6 EDK2 MEMORY MAP

GetMemoryMap

https://blog.csdn.net/xiaopangzi313/article/details/109928878

https://www.lab-z.com/stu5/

1.5

1.5.1

[toc]

Windows

1. **FAT32 File Allocation Table 32**
   - 2TB 4GB
   - USB SD
2. **NTFS New Technology File System**
   -
   - Windows SSD

macOS

1. **HFS+ Hierarchical File System Plus**
   -
   - macOS APFS
2. **APFS Apple File System**
   - SSD
   - macOS 10.13 Mac iOS

Linux

1. **ext4 Fourth Extended File System**
   -
   - Linux
2. **XFS**
   -
   -
3. **Btrfs B-tree File System**
   -
   -

1. exFAT Extended File Allocation Table

   - FAT32 NTFS
   - USB Windows macOS

1. ZFS Zettabyte File System

   - RAID-Z
   -

1.5.2 RAID

[toc]

RAID Redundant Array of Independent Disks RAID

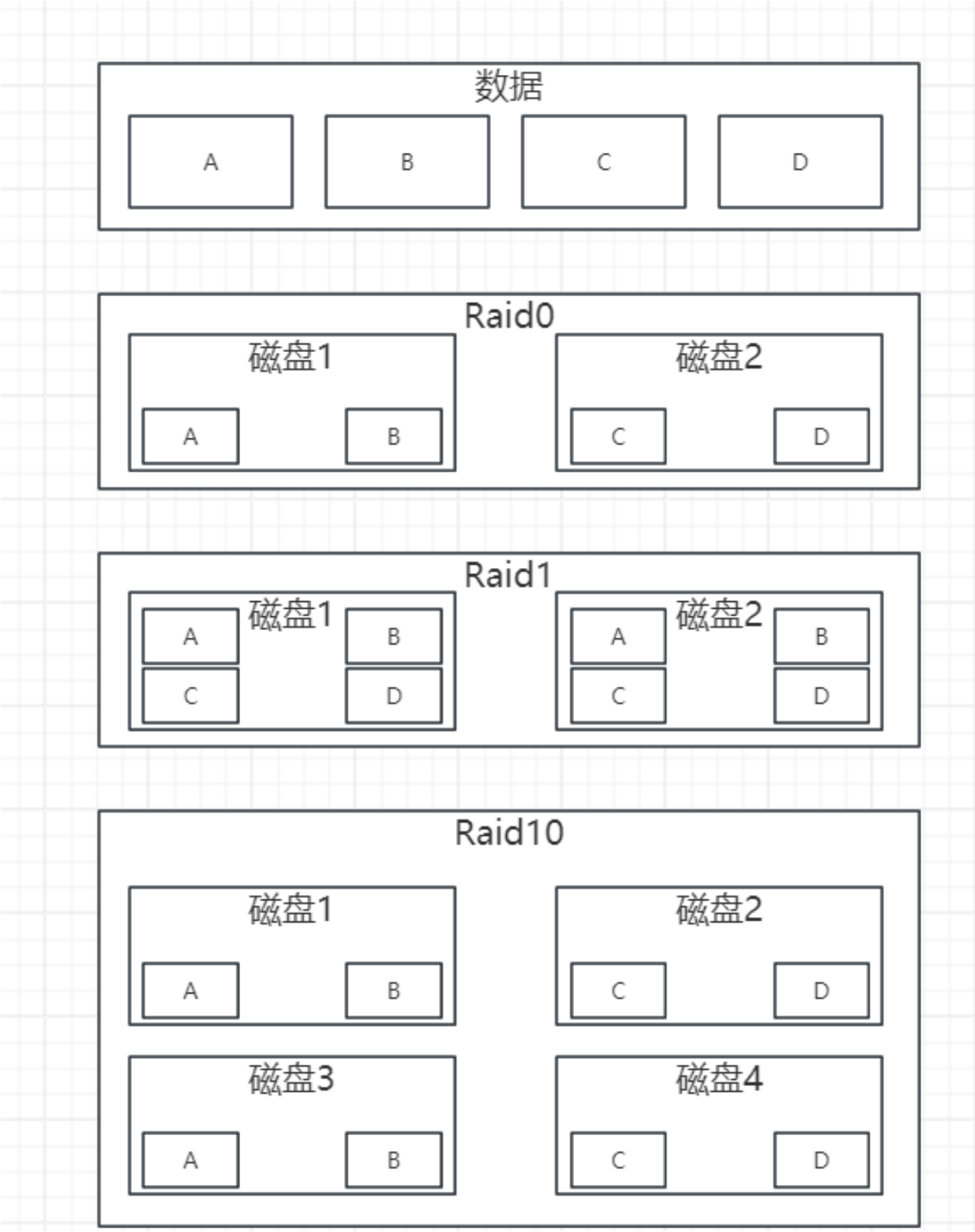
1. RAID RAID 0
2. RAID 1 RAID 5 RAID 6 RAID RAID 1
3. RAID RAID 5

RAID

- **RAID 0**
- **RAID 1**
- **RAID 5** RAID 0
- **RAID 6** RAID 5

- **RAID 10** RAID 1+0 RAID 0 RAID 1

0 1 10



RAID 3

- RAID 3
- RAID 3
-

-

RAID 5

- RAID 5
- RAID 0
- (\frac{(n-1)}{n}) n RAID 1
-
- RAID 3

RAID 6

- RAID 6 RAID 5
- RAID 6 RAID 5
- RAID 5
- **RAID 5** (\frac{(n-2)}{n}) n
- RAID 5

- **RAID 3**
- **RAID 5**
- **RAID 6**

1.5.3 SSD

[toc]

SSD SOC

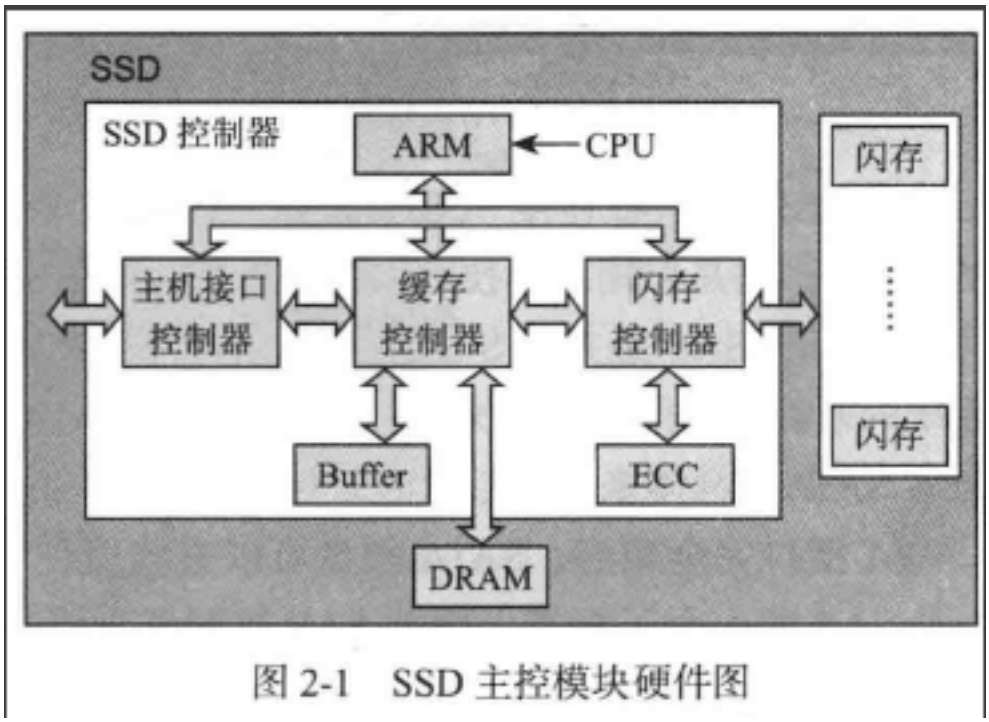


图 2-1 SSD 主控模块硬件图

SSD controller FTL nand

PCIE/SATA host FTL

FTL flash translation layer LBA SSD

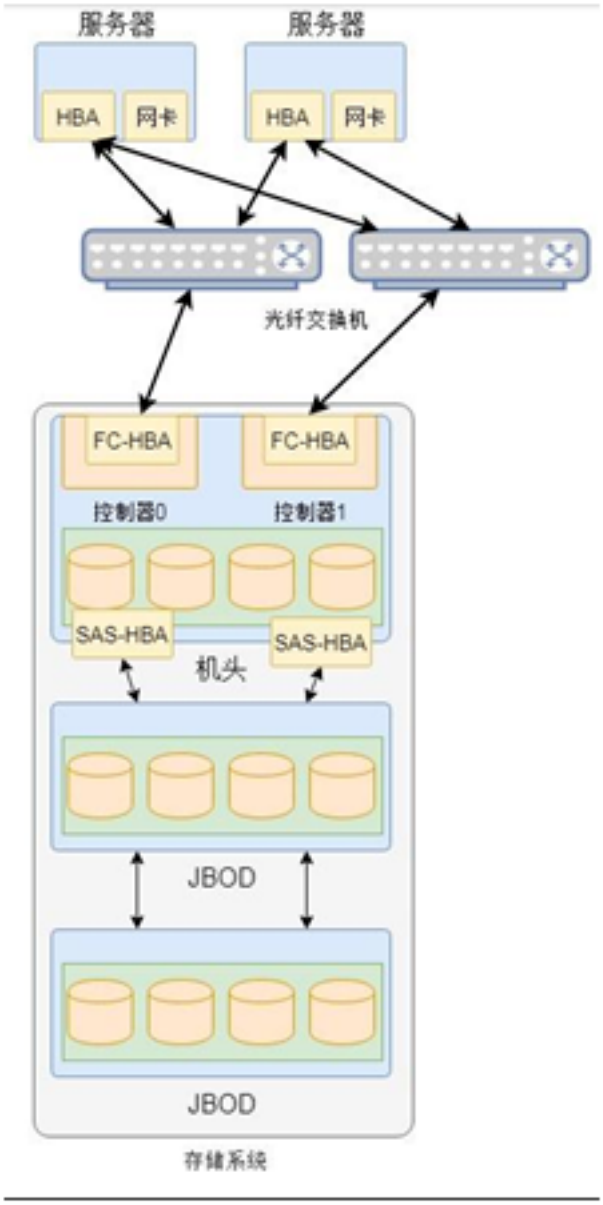
flash controller ECC

1.5.4

[toc]

( ) (JBOD)

LUN

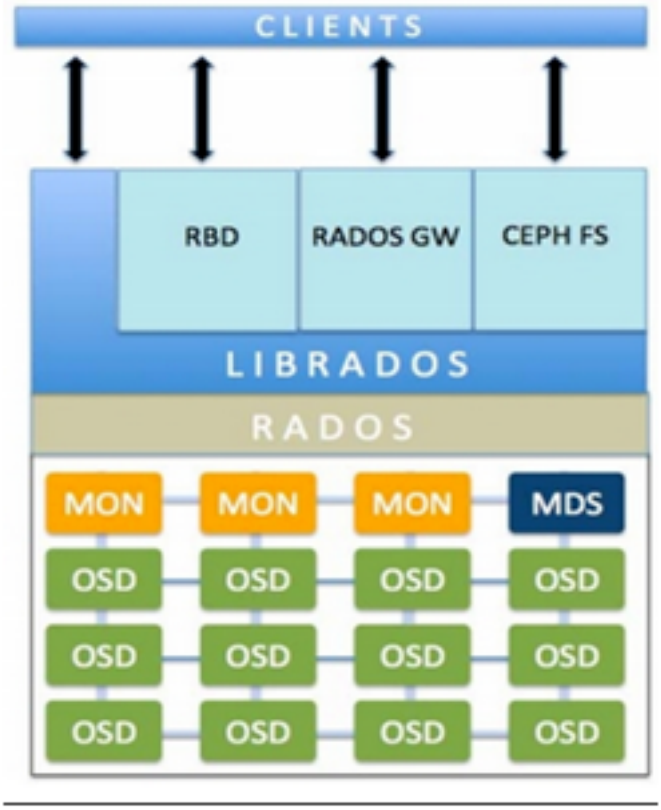


Web

–

Ceph HDFS

CMN SAM Ceph swift



Ceph MON OSD MDS

(1) MON MON

(2) OSD OSD

(3) MDS CephFS Ceph RADOS Ceph MDS

(4) RADOS RADOS ceph Ceph RADOS RADOS

(5) RBD ( ) RADOS

(6) CephFS Ceph Ceph POSIX

(7) Librados libRADOS PHP RUBY Java Python C++ RADOS

(8) RADOS GW RGW Ceph RGW Amazon S3 openstack Swift RUSTFUL API

RADOS GW MON ( ) CephFS

Blog

1.6

1.6.1 Hypervisor

[toc]

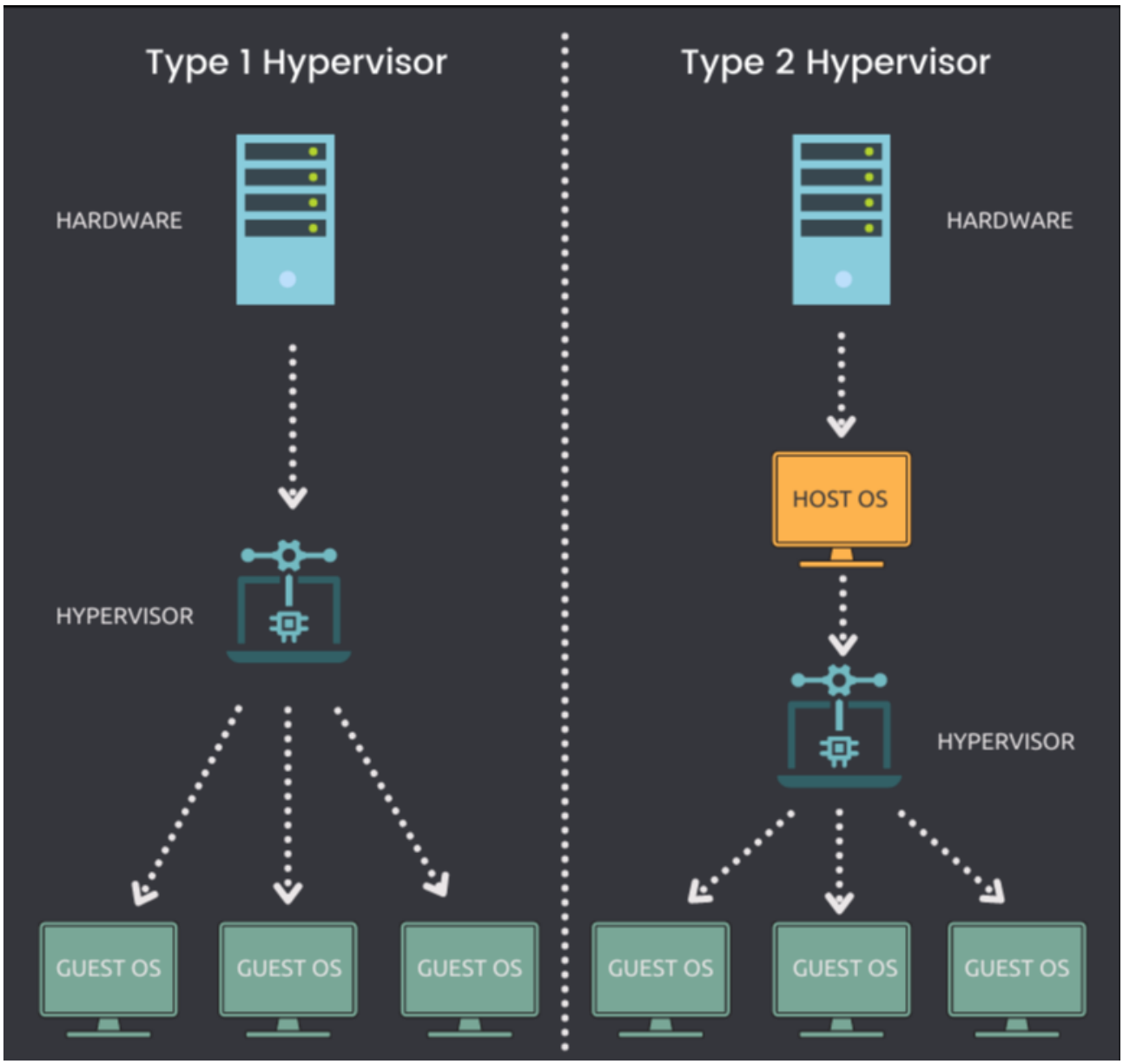
Introduction

Type 1 Hypervisor Hypervisor Type 1 Hypervisor

- Type 1 Hypervisor
- **(OS)**
- CPU
-
-
- Type 1 Hypervisor
-
- VMware ESXi Citrix Hypervisor Microsoft Hyper-V Type 1 Hypervisor

Type 2 Hypervisor 2

- Type 2 hypervisor
-
- CPU
- (OS)
-
- Type 2 hypervisor
-
- VMware Workstation Player VMware Workstation Pro VirtualBox Type 2 hypervisor



RISC-V Type-1 Hypervisor

RVirt RVirt MIT PDOS 2018 Rust **RISC-V Type-1 hypervisor** RVirt RISC-V MIT PDOS RVirt S-mode U-mode CSR hypervisor hypervisor RVirt QEMU virt Hifive Unleased fedora RVirt M S U RISC-V

Bao-hypervisor Bao hypervisor hypervisor Bao hypervisor

Bao

Bao hypervisor Bao Bao

Bao RISC-V Hypervisor Extension rocket chip CVA6

Xvisor Xvisor Xvisor VirtIO CPU IO Xvisor Normal vCPUs

2019 Xvisor RISC-V RISC-V RISC-V Hypervisor Extension 1.0 Xvisor Xvisor

seL4 seL4 NICTA CSIRO data61 seL4 seL4

seL4

- seL4
- seL4
- seL4
- seL4 GPLv2
- seL4
- seL4 RISC-V

KVM KVM Kernel-based Virtual Machine Linux VM VM KVM Intel VT AMD-V VM

KVM Red Hat IBM Intel KVM Linux Windows BSD KVM virt-manager virsh

KVM Linux Linux Linux

RISC-V Type-1 Hypervisor

1.6.2 MIT 6.824 Distributed Systems

6.824 MIT Golang

B

https://pdos.csail.mit.edu/6.82

1.7 Memory consistency and cache coherence

1.7.1 Memory Consistency

[toc]

consistency model 152 252
152 252 “ ”

consistency model FENCE

```
static int x = 0, y = 0;
static int r1, r2;


static void int thread_cpu0(void)
{
        x = 1;     /* 1) */
```

```
        r1 = y;    /* 2) */
}


static void int thread_cpu1(void)
{
        y = 1;     /* 3) */
        r2 = x;    /* 4) */
}


static void check_after_assign(void)
{
        printk("r1 = %d, r2 = %d\n", r1, r2);
}
```

4 6

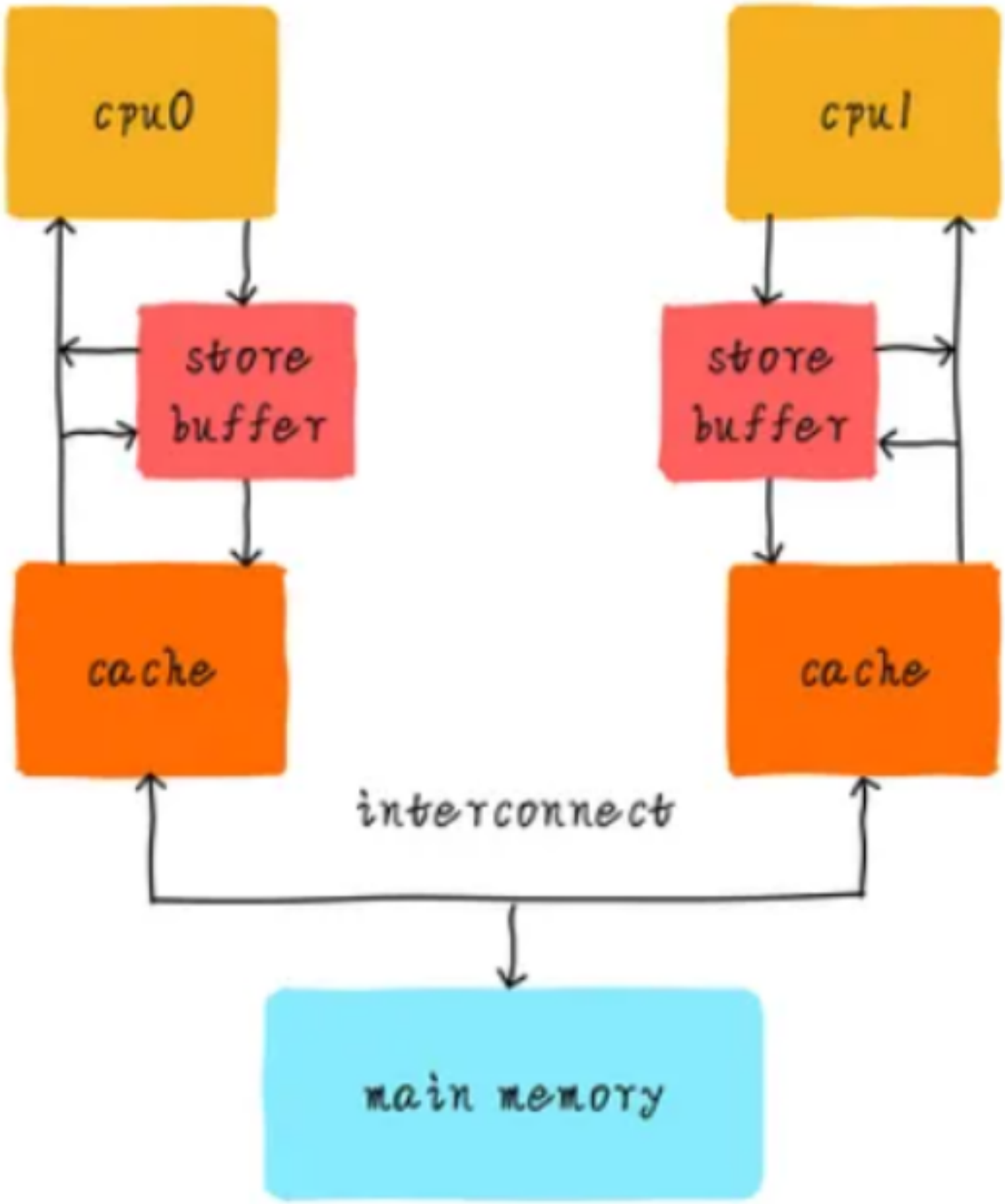
```
1) 2) 3) 4)
1) 3) 2) 4)
1) 3) 4) 2)
3) 4) 1) 2)
3) 1) 4) 2)
3) 1) 2) 4)
```

thread_cpu0 thread_cpu1 check_after_assign() 6 3

```
r1 = 1, r2 = 1
r1 = 0, r2 = 1    // 1) 2) 3) 4)
r1 = 1, r2 = 0    // 3) 4) 1) 2)
```

r1 r2 ” ” write buffer prefetching multiple cache banks

cache write buffer cache cache



write buffer 1) 3) 2) 4) case write buffer core 0

`r1 = 0, r2 = 0`

`3) 4) 1) 2)`

** ** TSO

Memory Consistency Model

“ ” —— core 0 write buffer core1

memory consistency and cache coherence

cache CPU

CPU

4 Store-Load Store-Store Load-Store Load-Load

Sequential Consistency SC Total Store Order TSO x86-64 Part Store Order PSO Relax Memory Order RMO

1. Sequential Consistency, SC

-
-

```
1 -> 3 -> 2 -> 4
3 -> 1 -> 4 -> 2
1 -> 3 -> 4 -> 2 //
3 -> 1 -> 2 -> 4
```

-
-

`1 -> 3 -> 4 -> 2`

1. `1: x = 1` CPU 0
2. `3: y = 1` CPU 1
3. `4: r2 = x` CPU 1
4. `2: r1 = y` CPU 0

`CPU 0` `CPU 1`

- `CPU 0` `1 -> 2`
- `CPU 1` `3 -> 4`

`1 -> 3 -> 4 -> 2`

- `4` `x` `1` `r2 = 1`
- `2` `y` `3` `r1 = 1`

2. Total Store Order, TSO TSO store load 4 store-store store-load load-load load-store TSO **store-load** 3

-
-

```
1 -> 3 -> 2 -> 4
3 -> 1 -> 4 -> 2
1 -> 2 -> 3 -> 4
3 -> 4 -> 1 -> 2
```

3. Partial Store Order, PSO PSO

Processor Consistency, PC Relaxed Consistency, RC

-
-

```
1 -> 3 -> 2 -> 4
3 -> 1 -> 4 -> 2
1 -> 2 -> 3 -> 4
3 -> 4 -> 1 -> 2
2 -> 1 -> 4 -> 3
4 -> 3 -> 2 -> 1
```

4. Relaxed Memory Order, RMO RMO

-
-

```
1 -> 3 -> 2 -> 4
3 -> 1 -> 4 -> 2
1 -> 2 -> 3 -> 4
3 -> 4 -> 1 -> 2
2 -> 1 -> 4 -> 3
4 -> 3 -> 2 -> 1
2 -> 4 -> 1 -> 3
4 -> 2 -> 3 -> 1
```

	SC	TSO	PSO	RMO
Store-Store			Yes	Yes
Store-Load		Yes	Yes	Yes
Load-Load				Yes
Load-Store				Yes

`r1 = 0, r2 = 0` Linux `smp_mb()` `smp_mb()`

- A Primer on Memory Consistency and Cache Coherence (2nd Edition)
- https://blog.csdn.net/qq_29328443/article/details/107616795
- https://blog.csdn.net/anyegongjuezjd/article/details/125954805
- https://zhuanlan.zhihu.com/p/141655129?from_voters_page=true
-

Chapter 2

2.1 ARM

core

1. BMC C2000 CHIP POR_N pervsoc
2. otprom valid SCP SRAM BISR PVO(SCP) ROM otprom
3. SCP SRAM
4. SCP ROM FW GPIO CHIP ID, CHIP ID CHIP;
5. I2C/FSI
6. CHIP SCP QSPI;
7. CHIP SCP CHIP CHIP
8. CHIP SCP CHIP SCP
9. CHIP SCP QSPI SCP RAM FW CHIP SRAM;
10. CHIP SCP 8 SCP RAM FW SCP, SCP SRAM:
11. CHIP SCP SRAM, SCP RAM FW:
12. CHIP SCP QSPI C2C SCP
13. CHIP SCP CHIP SCP C2C C2C;
14. SCP CHIP ID CMN OCM(SLC) SAM;
15. CHIP SCP MCP RAM FW, MCP( ) +
16. SCP CHIP L2 NCU SCOM/FIR;+
17. CHIP SCP Hostboot base CHIP OCM(SLC);+
18. CHIP SCP CHIP
19. srest 0x100

core-hostboot

1. kernel other thread spinlock, thread
2. hostboot kernel C/C++ main +

3. hostboot kernel
4. MMU(SLB/TLB), DIMM OCM(SLC, 28 * 2MB=56MB 3/4
5. TB CPU
6. Scratch FSP/BMC;
7. CPU init main
8. kernel other thread spinlock 1 thread
9. hostboot task task user kernel hypersior
10. init main VFS;
11. VFS base modules

init

12. initservice
13. initservice g_taskinfolist
14. secureboot_base/ipmibase/pnor/vfs/extinitsvc
15. hostboot extended image
16. extinitsvc g_exttaskinfolist
17. istepdisp hostboot
18. istep hostboot
19. istep 21.03 payload(skiboot,OPAL);+
20. payload(skiboot,OPAL)

core-skiboot

1. C2000 skiboot
2. thread skiboot relocate thread
3. thread boot flag=1, thread
4. thread main cpu entry, thread secondary cpu entry;
5. uart(console);
6. skiboot /
7. skiboot
8. opal call table;
9. hostboot device tree HDAT;
10. DT SCOM/eSPI(LPC):
11. DT C2000 DIMM
12. DT homer C2000
13. DT CPU(thread);
14. opal DT;

15.

16. BMC ipmi/BT/UART;

17. GIC

18. CPU thread

19. sreset CPU

20. TOD TOD

21. sensor OPAL

22.

23.NV

24. jiayu

25. TPM

26. opal console(UART)

27. PCIE HB PCIE (RC CXL PCIe device tree)

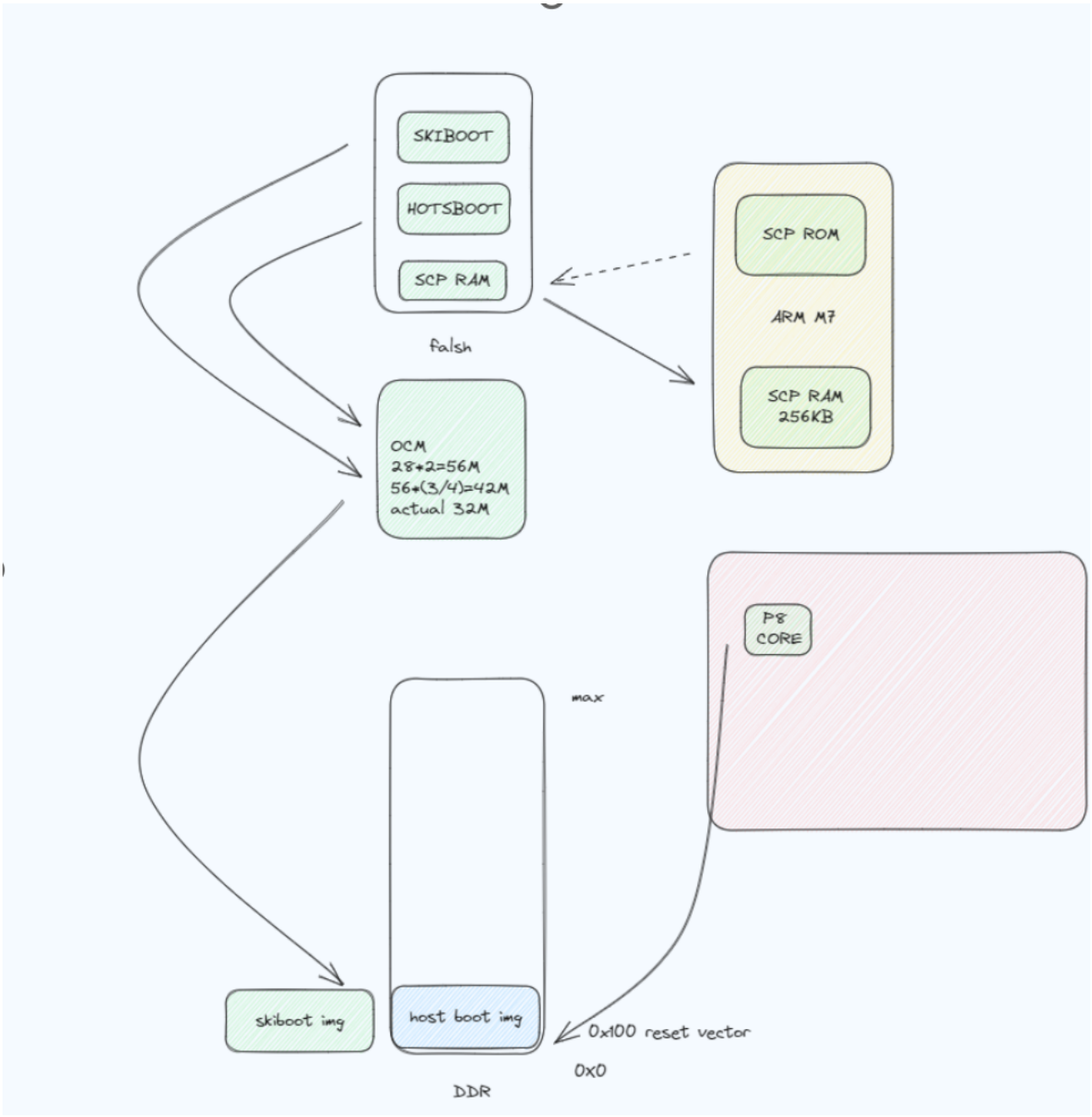
28.PCIE SLOT

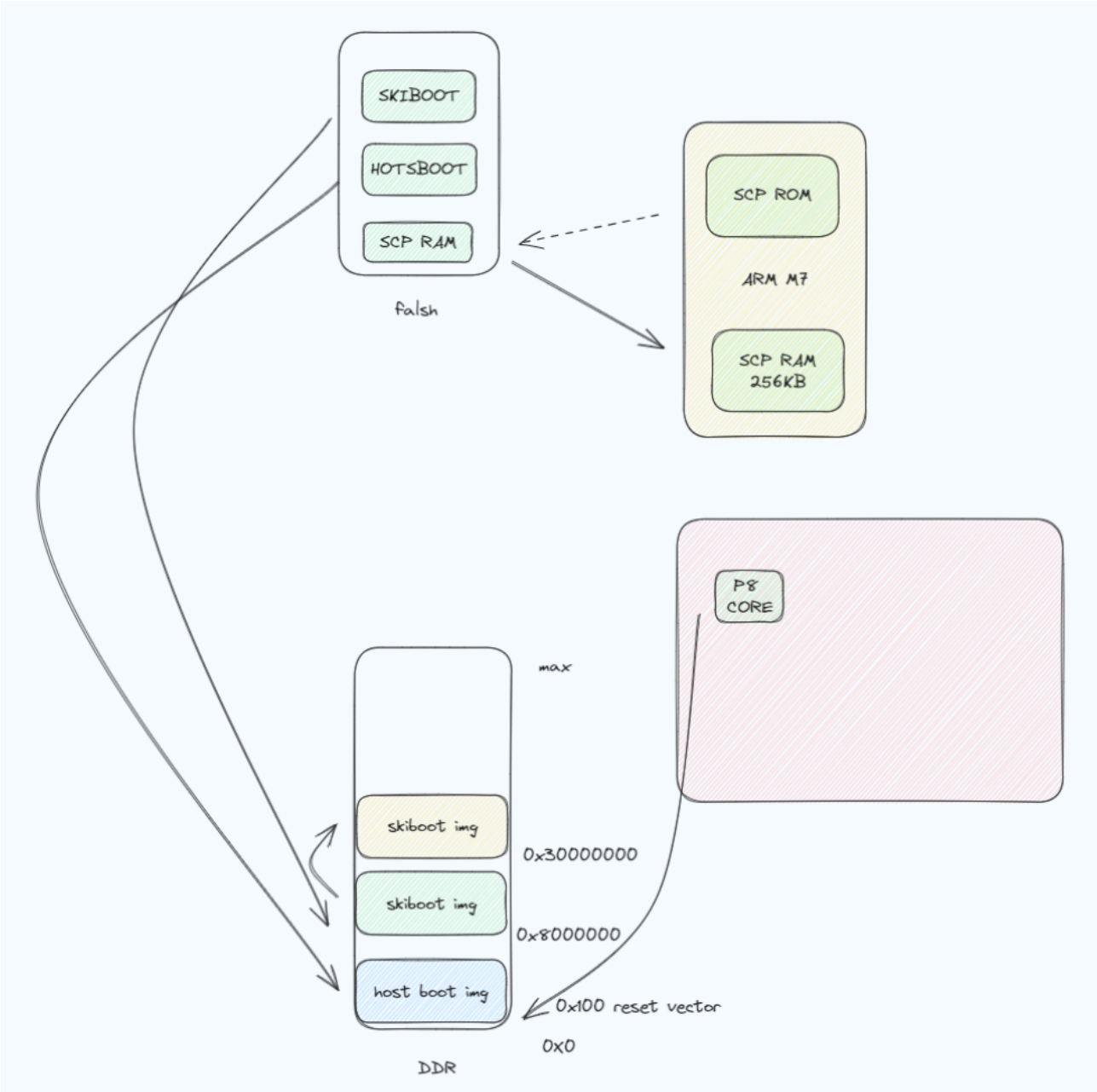
29.

30.

31. zImage(boot loader petitboot);

32. boot loader


SKIBOOT
HOTSBOOT
SCP RAM
falsh
SCP ROM
ARM M7
SCP RAM
256KB
OCM
28+2=56M
56+(3/4)=42M
actual 32M
P8
CORE
max
skiboot img
host boot img
0x100 reset vector
0x0
DDR



CMN OCM(on chip memory) L3 cache DDR Intel cache as ram

OCM memory CMN OCM(on chip memory) 28*2=56M 3/4 32M

SCP ram DDR training OCM flash DDR

OCM SNF

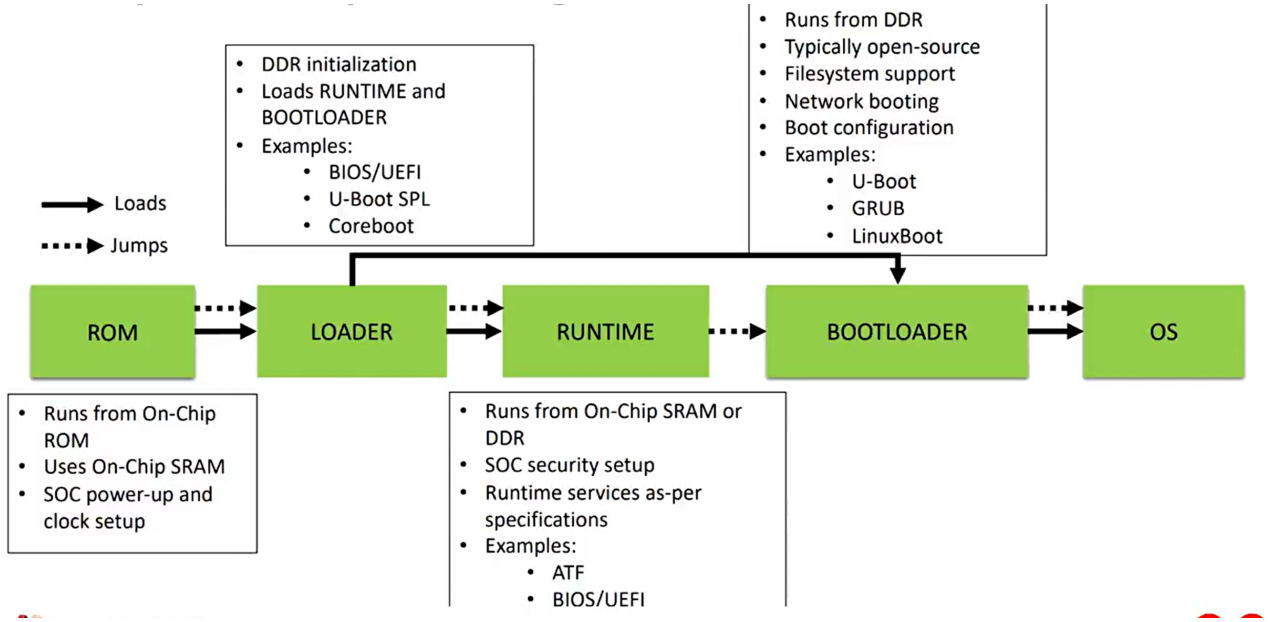
SCP OCM SCP RAM hostboot OCM skiboot OCM skiboot ddr training DDR OCM OCM cache

hostboot skiboot 0x800 0000

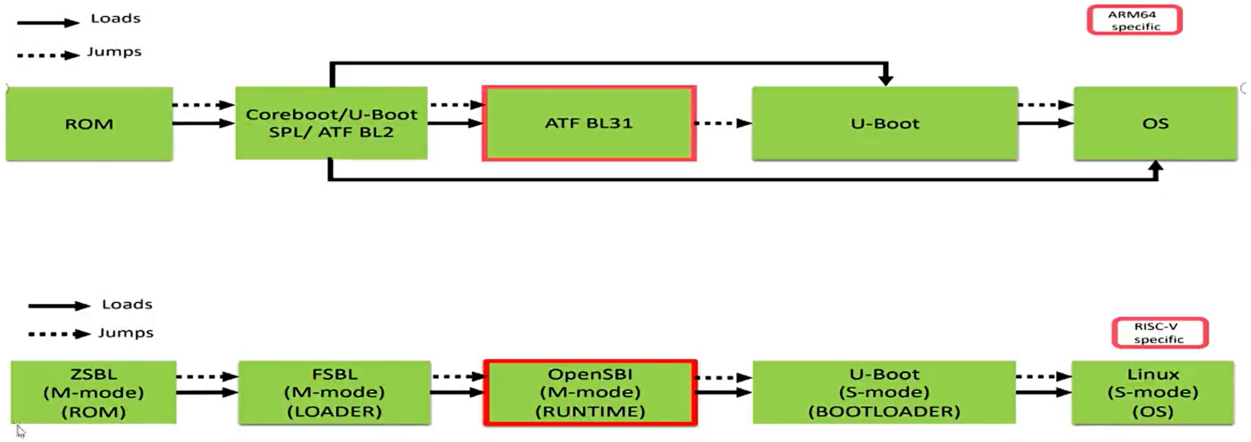
32M 0x200 0000, skiboot 0x3000 0000 OCM 0x0 0x0

2.2 RISC-V Firmware

Firmware

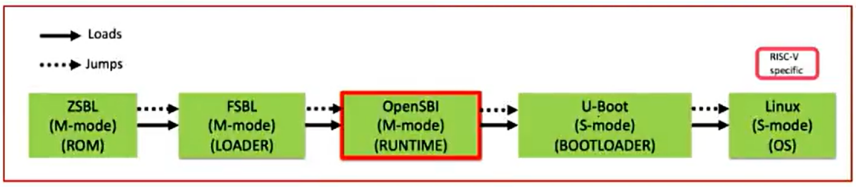


ARM64 vs RISC-V



uboot EDK2 PEI DXE

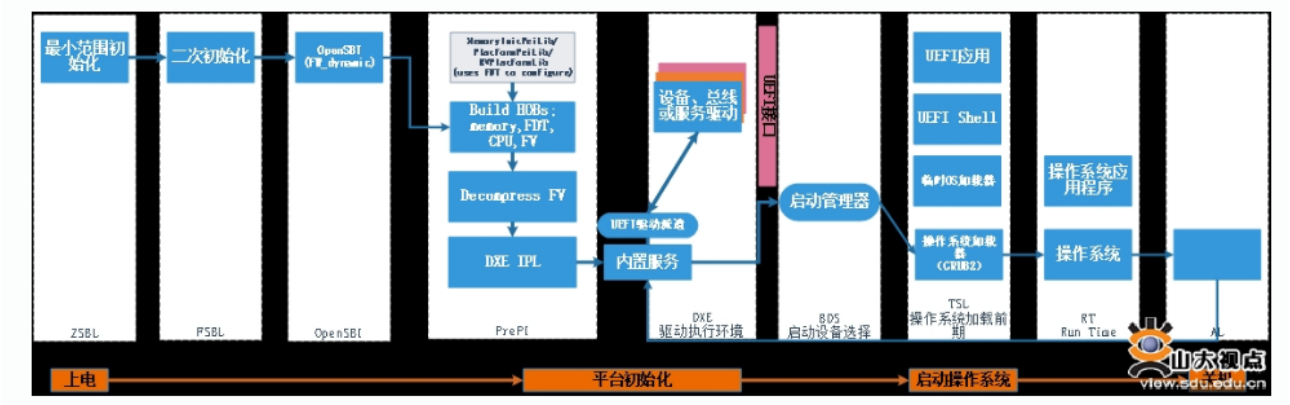
RISC-V linux启动流程如图，启动流程：ZSBL-->FSBL-->OpenSBI-->u-boot(UEFI)-->Linux



1. ZSBL（第0阶段Bootloader）at 0x00010000 （ROM in SOC）
2. FSBL（第1阶段Bootloader）
3. OpenSBI (runtime servics) M模式
4. U-boot/UEFI (bootloader)，
5. Linux启动linux需要以下文件：
   a. /boot/extlinux/extlinux.conf:包含启动菜单、默认启动等
   b. Image：Linux镜像
   c. Initrd:Linux执行需要的ramdisk，执行其中的init文件并进行进一步处理
   d. 设备树(.dtb)（可选）:未指定时会使用默认的dtb

ZSBL: SCP ROM

FSBL: SCP RAM



2.3 Cache as RAM and On Chip Memory

x86 Cache as RAM, CAR;

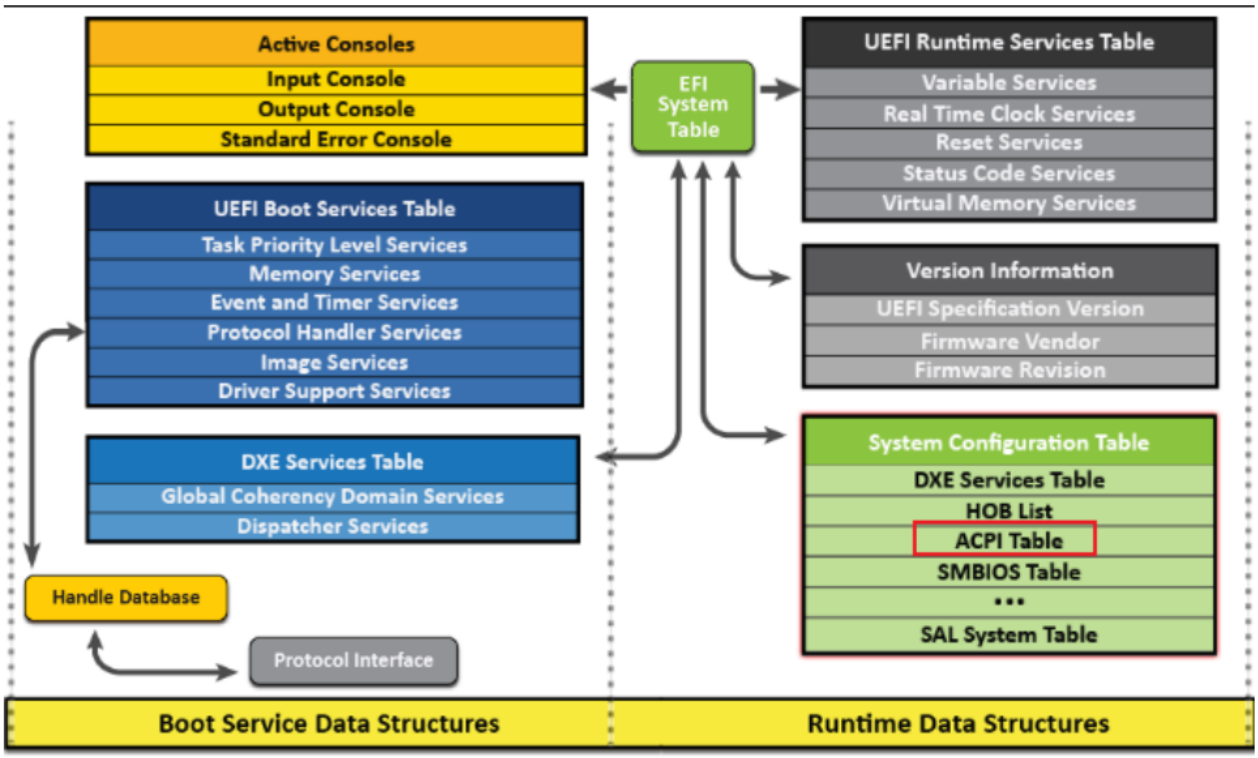
ARM On Chip Memory OCM

DDR cache RAM C

CAR

OCM

2.4 System Table



MdeModulePkg\Core\Dxe\DxeMain\DxeMain.c

```
  //
  // Allocate the EFI System Table and EFI Runtime Service Table from EfiRuntimeServicesData
  // Use the templates to initialize the contents of the EFI System Table and EFI Runtime
    ↪  Services Table
  //
  gDxeCoreST = AllocateRuntimeCopyPool (sizeof (EFI_SYSTEM_TABLE),
↪   &mEfiSystemTableTemplate);
  ASSERT (gDxeCoreST != NULL);

  gDxeCoreRT = AllocateRuntimeCopyPool (sizeof (EFI_RUNTIME_SERVICES),
↪   &mEfiRuntimeServicesTableTemplate);
  ASSERT (gDxeCoreRT != NULL);

  gDxeCoreST->RuntimeServices = gDxeCoreRT;


///
/// EFI System Table
///
typedef struct {
  ///
  /// The table header for the EFI System Table.
```

```
///
EFI_TABLE_HEADER                  Hdr;
///
/// A pointer to a null terminated string that identifies the vendor
/// that produces the system firmware for the platform.
///
CHAR16                            *FirmwareVendor;
///
/// A firmware vendor specific value that identifies the revision
/// of the system firmware for the platform.
///
UINT32                            FirmwareRevision;
///
/// The handle for the active console input device. This handle must support
/// EFI_SIMPLE_TEXT_INPUT_PROTOCOL and EFI_SIMPLE_TEXT_INPUT_EX_PROTOCOL.
///
EFI_HANDLE                        ConsoleInHandle;
///
/// A pointer to the EFI_SIMPLE_TEXT_INPUT_PROTOCOL interface that is
/// associated with ConsoleInHandle.
///
EFI_SIMPLE_TEXT_INPUT_PROTOCOL    *ConIn;
///
/// The handle for the active console output device.
///
EFI_HANDLE                        ConsoleOutHandle;
///
/// A pointer to the EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL interface
/// that is associated with ConsoleOutHandle.
///
EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL   *ConOut;
///
/// The handle for the active standard error console device.
/// This handle must support the EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL.
///
EFI_HANDLE                        StandardErrorHandle;
///
/// A pointer to the EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL interface
/// that is associated with StandardErrorHandle.
///
EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL   *StdErr;
///
/// A pointer to the EFI Runtime Services Table.
///
EFI_RUNTIME_SERVICES              *RuntimeServices;
///
/// A pointer to the EFI Boot Services Table.
///
EFI_BOOT_SERVICES                 *BootServices;
```

```
  ///
  /// The number of system configuration tables in the buffer ConfigurationTable.
  ///
  UINTN                             NumberOfTableEntries;
  ///
  /// A pointer to the system configuration tables.
  /// The number of entries in the table is NumberOfTableEntries.
  ///
  EFI_CONFIGURATION_TABLE           *ConfigurationTable;
} EFI_SYSTEM_TABLE;
```

2.5 EDK2 EVENT

[toc]

UEFI

/ intel 8253/8254 PC IC MSC8051,

Timer cpu

PC/AT UEFI 8254 Timer mode 3 1.1931816MHz tick mTimerPeriod 1ms 10000 tick =10000*100ns pin 8259 IRQ0 CPU Timer

cpu archprotocol Legacy8259Protocol , cpu IDT Timer TimerInterruptHandler

Timer

Timer

1.

2.

3.

4. Timer CoreTimerTick(mTimerPeriod)/CoreTimerTick(100ms) TimerArchprotocol install EVT_NOTI

5. Sytem time mEfiSystemTime =mEfiSystemTime+ 1Tick(100ms)

6. timer event list event BS->SignalEvent signal

BS->SignalEvent--->CoreNotifyEvent-->CoreRestoreTpl-->CoreDispatchEventNotifies-->A. EVT_NOTIFY_SIGNAL >SignalCount = 0 Event->NotifyFunction (Event, Event->NotifyContext) Tpl gEventPending mask bit event

7.

8.

1. gEventQueue--- A list of event's to notify for each priority level
2. mEfiTimerList---- timer event

3. gEventSignalQueue --- A list of events to signal based on EventGroup type

4. EVT_NOTIFY_WAIT event BS-> **CoreCheckEvent**->CoreNotifyEvent-->CoreRestoreTpl-->CoreDispatchEventNotifies gEventQueue IEvent

gEventPending gEventQueue 1 0

gEventPending 32 1

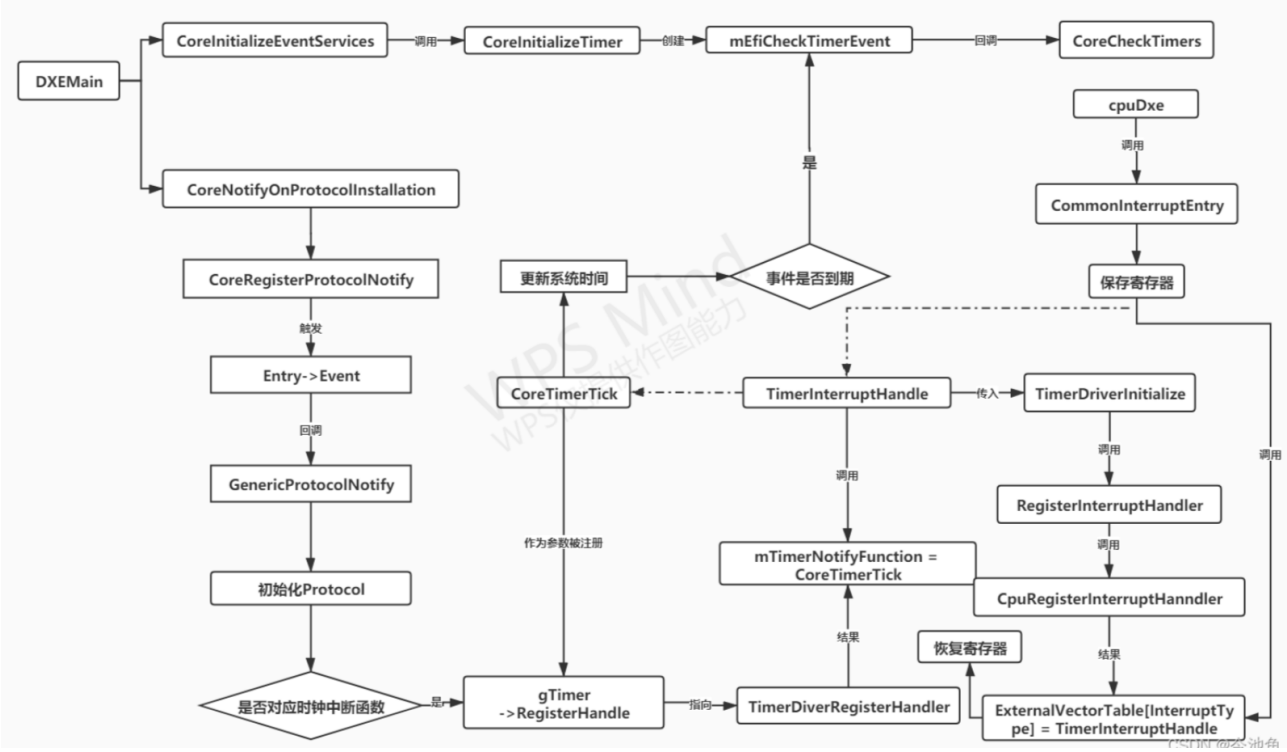
x86 IRQ

- IRQ0:
- IRQ1:
- IRQ2:
- IRQ3: 2 COM2
- IRQ4: 1 COM1
- IRQ5:
- IRQ6:
- IRQ7: 1 LPT1
- IRQ

event

```
//
// Task priority level
//
#define TPL_APPLICATION       4
#define TPL_CALLBACK          8
#define TPL_NOTIFY            16
#define TPL_HIGH_LEVEL        31
```

31



DXE main

```
  //
  // Initialize the Event Services
  //
  Status = CoreInitializeEventServices ();

...

  //
  // Register for the GUIDs of the Architectural Protocols, so the rest of the
  // EFI Boot Services and EFI Runtime Services tables can be filled in.
  // Also register for the GUIDs of optional protocols.
  //
  CoreNotifyOnProtocolInstallation ();
```

event

```
/**
  Called by the platform code to process a tick.

  @param  Duration              The number of 100ns elapsed since the last call
                                to TimerTick

**/
VOID
EFIAPI
CoreTimerTick (
```

```
  IN UINT64   Duration
  )
{
  IEVENT          *Event;

  //
  // Check runtiem flag in case there are ticks while exiting boot services
  //
  CoreAcquireLock (&mEfiSystemTimeLock);

  //
  // Update the system time
  //
  mEfiSystemTime += Duration;

  //
  // If the head of the list is expired, fire the timer event
  // to process it
  //
  if (!IsListEmpty (&mEfiTimerList)) {
    Event = CR (mEfiTimerList.ForwardLink, IEVENT, Timer.Link, EVENT_SIGNATURE);

    if (Event->Timer.TriggerTime <= mEfiSystemTime) {
      CoreSignalEvent (mEfiCheckTimerEvent);
    }
  }

  CoreReleaseLock (&mEfiSystemTimeLock);
}
```

timer event

```
/**
  Initializes "event" support.

  @retval EFI_SUCCESS            Always return success

**/
EFI_STATUS
CoreInitializeEventServices (
  VOID
  )
{
  UINTN        Index;

  for (Index=0; Index <= TPL_HIGH_LEVEL; Index++) {
    InitializeListHead (&gEventQueue[Index]);
  }

  CoreInitializeTimer ();
```

```
  CoreCreateEventEx (
    EVT_NOTIFY_SIGNAL,
    TPL_NOTIFY,
    EfiEventEmptyFunction,
    NULL,
    &gIdleLoopEventGuid,
    &gIdleLoopEvent
    );

  return EFI_SUCCESS;
}
```

- gEventQueue--- A list of event's to notify for each priority level
- CoreInitializeTimer

```
/**
  Initializes timer support.

**/
VOID
CoreInitializeTimer (
  VOID
  )
{
  EFI_STATUS  Status;

  Status = CoreCreateEventInternal (
             EVT_NOTIFY_SIGNAL,
             TPL_HIGH_LEVEL - 1,
             CoreCheckTimers,
             NULL,
             NULL,
             &mEfiCheckTimerEvent
             );
  ASSERT_EFI_ERROR (Status);
}
```

```
#include <Uefi.h>
#include <Library/UefiBootServicesTableLib.h>
#include <Library/UefiLib.h>

EFI_EVENT TimerEvent;
EFI_EVENT ExitBootServicesEvent;
BOOLEAN TimerTriggered = FALSE;

VOID EFIAPI TimerEventHandler(
```

```
    IN EFI_EVENT Event,
    IN VOID *Context
) {
    TimerTriggered = TRUE;
    Print(L"Timer event triggered!\n");
}

EFI_STATUS
EFIAPI
UefiMain(
    IN EFI_HANDLE ImageHandle,
    IN EFI_SYSTEM_TABLE *SystemTable
) {
    EFI_STATUS Status;

    //         10000000 100ns (1 )
    Status = gBS->CreateEvent(
        EVT_TIMER | EVT_NOTIFY_SIGNAL,
        TPL_CALLBACK,
        TimerEventHandler,
        NULL,
        &TimerEvent
    );
    if (EFI_ERROR(Status)) {
        Print(L"Failed to create event: %r\n", Status);
        return Status;
    }

    //    1
    Status = gBS->SetTimer(
        TimerEvent,
        TimerRelative,
        10000000 // 100ns  10000000 = 1
    );
    if (EFI_ERROR(Status)) {
        Print(L"Failed to set timer: %r\n", Status);
        gBS->CloseEvent(TimerEvent);
        return Status;
    }

    //
    while (!TimerTriggered) {
        gBS->Stall(100000); //  100ms
    }

    //
    gBS->CloseEvent(TimerEvent);

    return EFI_SUCCESS;
```

```
}
```

1

gBS->CreateEvent

```
typedef struct {
    UINT32 Type;
    EFI_TPL NotifyTpl;
    EFI_EVENT_NOTIFY NotifyFunction;
    VOID *NotifyContext;
    BOOLEAN Triggered;
} EFI_EVENT_INTERNAL;

EFI_STATUS
EFIAPI
CreateEvent (
    IN UINT32 Type,
    IN EFI_TPL NotifyTpl,
    IN EFI_EVENT_NOTIFY NotifyFunction,
    IN VOID *NotifyContext,
    OUT EFI_EVENT *Event
) {
    EFI_STATUS Status;
    EFI_EVENT_INTERNAL *NewEvent;

    //
    if (Event == NULL || (Type & EVT_NOTIFY_SIGNAL && NotifyFunction == NULL)) {
        return EFI_INVALID_PARAMETER;
    }

    //
    NewEvent = AllocateZeroPool(sizeof(EFI_EVENT_INTERNAL));
    if (NewEvent == NULL) {
        return EFI_OUT_OF_RESOURCES;
    }

    //
    NewEvent->Type = Type;
    NewEvent->NotifyTpl = NotifyTpl;
    NewEvent->NotifyFunction = NotifyFunction;
    NewEvent->NotifyContext = NotifyContext;
    NewEvent->Triggered = FALSE;

    //
    Status = RegisterEvent(NewEvent);
    if (EFI_ERROR(Status)) {
        FreePool(NewEvent);
        return Status;
    }
```

```
    //
    *Event = (EFI_EVENT)NewEvent;

    return EFI_SUCCESS;
}
```

event

event

```
  if ((Type & EVT_NOTIFY_SIGNAL) != 0x00000000) {
    //
    // The Event's NotifyFunction must be queued whenever the event is signaled
    //
    InsertHeadList (&gEventSignalQueue, &IEvent->SignalLink);
  }
```

32 bitmap 1

SignalEvent event 32

```
/**
  Signals the event.  Queues the event to be notified if needed.

  @param  UserEvent               The event to signal .

  @retval EFI_INVALID_PARAMETER  Parameters are not valid.
  @retval EFI_SUCCESS            The event was signaled.

**/
EFI_STATUS
EFIAPI
CoreSignalEvent (
  IN EFI_EVENT    UserEvent
  )
/**
  Queues the event's notification function to fire.

  @param  Event                   The Event to notify

**/
VOID
CoreNotifyEvent (
  IN  IEVENT      *Event
  )
{

  //
  // Event database must be locked
```

```
  //
  ASSERT_LOCKED (&gEventQueueLock);

  //
  // If the event is queued somewhere, remove it
  //

  if (Event->NotifyLink.ForwardLink != NULL) {
    RemoveEntryList (&Event->NotifyLink);
    Event->NotifyLink.ForwardLink = NULL;
  }

  //
  // Queue the event to the pending notification list
  //

  InsertTailList (&gEventQueue[Event->NotifyTpl], &Event->NotifyLink);
  gEventPending |= (UINTN)(1 << Event->NotifyTpl);
}
```

Duang CoreTimerTick TPL restore TPL

timer event mEfiTimerList

TPL

```
  while (gEventPending != 0)
{
    PendingTpl = (UINTN) HighBitSet64 (gEventPending);
    if (PendingTpl <= NewTpl) {
      break;
    }
    gEfiCurrentTpl = PendingTpl;
    if (gEfiCurrentTpl < TPL_HIGH_LEVEL) {
      CoreSetInterruptState (TRUE);
    }
    CoreDispatchEventNotifies (gEfiCurrentTpl);
    // gEventQueue [gEfiCurrentTpl]   event Notification
  }
```

gEventPending 32 1

1.

2. TPL TPL 16 17 16 TPL 17

3. pending pending TPL 16 15 16

---

waitforevent 4 10ms 31 RestoreTpl 4, 31 4 30 5 4

CoreTimerTick

```
/**
  Lowers the task priority to the previous value.   If the new
  priority unmasks events at a higher priority, they are dispatched.

  @param  NewTpl  New, lower, task priority

**/
VOID
EFIAPI
CoreRestoreTpl (
  IN EFI_TPL NewTpl
  )



{
  EFI_TPL     OldTpl;
  EFI_TPL     PendingTpl;
```

- OldTpl:
- PendingTpl:

```
  OldTpl = gEfiCurrentTpl;
  if (NewTpl > OldTpl) {
    DEBUG ((EFI_D_ERROR, "FATAL ERROR - RestoreTpl with NewTpl(0x%x) > OldTpl(0x%x)\n",
↪   NewTpl, OldTpl));
    ASSERT (FALSE);
  }
  ASSERT (VALID_TPL (NewTpl));
```

- OldTpl = gEfiCurrentTpl:
- NewTpl NewTpl OldTpl

```
  if (OldTpl >= TPL_HIGH_LEVEL  &&  NewTpl < TPL_HIGH_LEVEL) {
    gEfiCurrentTpl = TPL_HIGH_LEVEL;
  }
```

- TPL_HIGH_LEVEL TPL_HIGH_LEVEL TPL_HIGH_LEVEL

```
  while (gEventPending != 0) {
    PendingTpl = (UINTN) HighBitSet64 (gEventPending);
    if (PendingTpl <= NewTpl) {
      break;
    }
```

```
    gEfiCurrentTpl = PendingTpl;
    if (gEfiCurrentTpl < TPL_HIGH_LEVEL) {
      CoreSetInterruptState (TRUE);
    }
    CoreDispatchEventNotifies (gEfiCurrentTpl);
  }
```

- while (gEventPending != 0):
- PendingTpl = (UINTN) HighBitSet64 (gEventPending):
- PendingTpl NewTpl
- PendingTpl TPL_HIGH_LEVEL
- CoreDispatchEventNotifies

```
  gEfiCurrentTpl = NewTpl;

  if (gEfiCurrentTpl < TPL_HIGH_LEVEL) {
    CoreSetInterruptState (TRUE);
  }

}
```

- NewTpl
- TPL_HIGH_LEVEL

CoreDispatchEventNotifies

```
/**
  Dispatches all pending events.

  @param  Priority              The task priority level of event notifications
                                to dispatch

**/
VOID
CoreDispatchEventNotifies (
  IN EFI_TPL      Priority
  )


{
  IEVENT          *Event;
  LIST_ENTRY      *Head;
```

- Event:
- Head:

```
  CoreAcquireEventLock ();
  ASSERT (gEventQueueLock.OwnerTpl == Priority);
  Head = &gEventQueue[Priority];
```

- CoreAcquireEventLock ():
- ASSERT (gEventQueueLock.OwnerTpl == Priority):
- Head = &gEventQueue[Priority]:

```
  while (!IsListEmpty (Head)) {

    Event = CR (Head->ForwardLink, IEVENT, NotifyLink, EVENT_SIGNATURE);
    RemoveEntryList (&Event->NotifyLink);

    Event->NotifyLink.ForwardLink = NULL;
```

- while (!IsListEmpty (Head)):
- Event = CR (Head->ForwardLink, IEVENT, NotifyLink, EVENT_SIGNATURE):
- RemoveEntryList (&Event->NotifyLink):
- Event->NotifyLink.ForwardLink = NULL:

SIGNAL

```
    if ((Event->Type & EVT_NOTIFY_SIGNAL) != 0) {
      Event->SignalCount = 0;
    }
```

- EVT_NOTIFY_SIGNAL SignalCount

```
    CoreReleaseEventLock ();

    ASSERT (Event->NotifyFunction != NULL);
    Event->NotifyFunction (Event, Event->NotifyContext);
```

- CoreReleaseEventLock ():
- ASSERT (Event->NotifyFunction != NULL):
- Event->NotifyFunction (Event, Event->NotifyContext):

```
    CoreAcquireEventLock ();
  }
```

- CoreAcquireEventLock ():

```
  gEventPending &= ~(UINTN)(1 << Priority);
  CoreReleaseEventLock ();
}
```

- gEventPending &= ~(UINTN)(1 << Priority):
- CoreReleaseEventLock ():

- 31TPL
-

Blog

Blog

2.6 EDK2 MEMORY MAP

GetMemoryMap

https://blog.csdn.net/xiaopangzi313/article/details/109928878

https://www.lab-z.com/stu5/

Chapter 3

OS

3.1 FreeRTOS

[TOC]

1. Determinism

RTOS RTOS

2. Low Latency

RTOS

3. Priority Scheduling

RTOS CPU Priority Preemptive Scheduling Round-Robin Scheduling

4. Interrupt Handling

RTOS ISR

5. Time Management

RTOS

6. Resource Management

RTOS

Linux

1.

- Linux
- Linux

2.

- Linux CPU Linux SCHED_FIFO SCHED_RR
- Linux

3.

- Linux
- **Swapping** Linux

4.

- Linux
-

5.

- Linux CPU
-

6.

- **vs** Linux Linux

Part II

Chapter 4

ACPI

4.1 ACPI Introduction and Overview

[toc]

History of ACPI

ACPI 1990 Phoenix ACPI BIOS( ) BIOS BIOS / (I/C
and play) BIOS

BIOS

ACPI

What is ACPI?

ACPI ( )

ACPI ACPI (BIOS) ” 1”

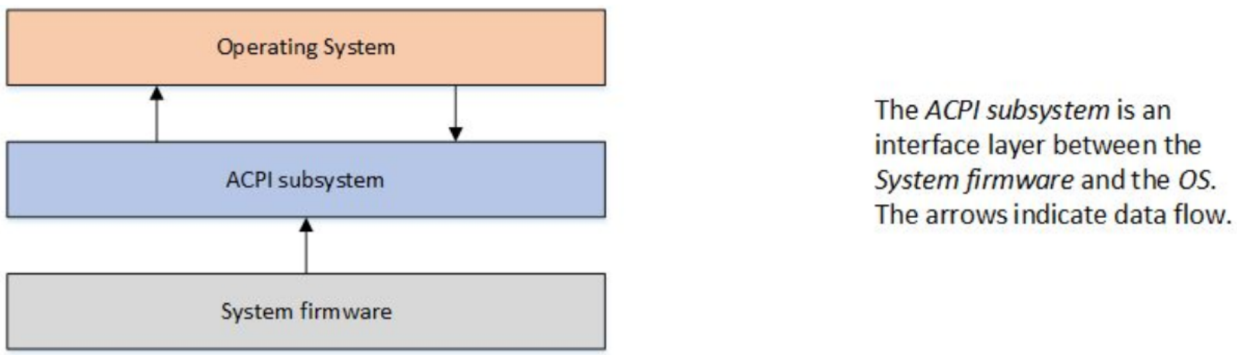
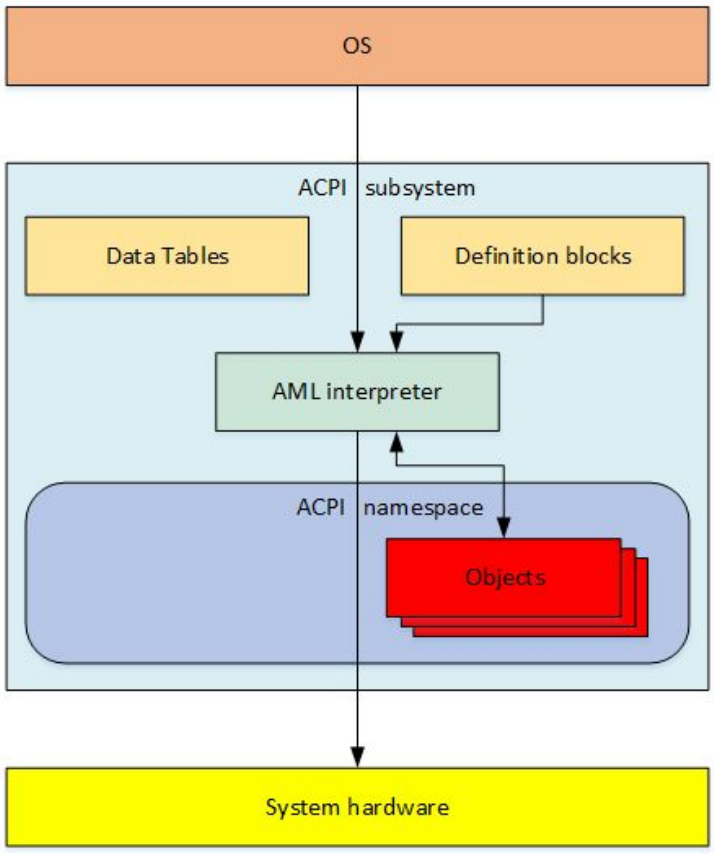


Figure 1: ACPI overview

ACPI (data table) (definition block) ” 2”



ACPI子系统由两类数据结构组成：
数据表(data table)和定义块(definition block)

初始化时，AML解释器(AML interpreter)
将定义块中的字节码提取为可枚举对象

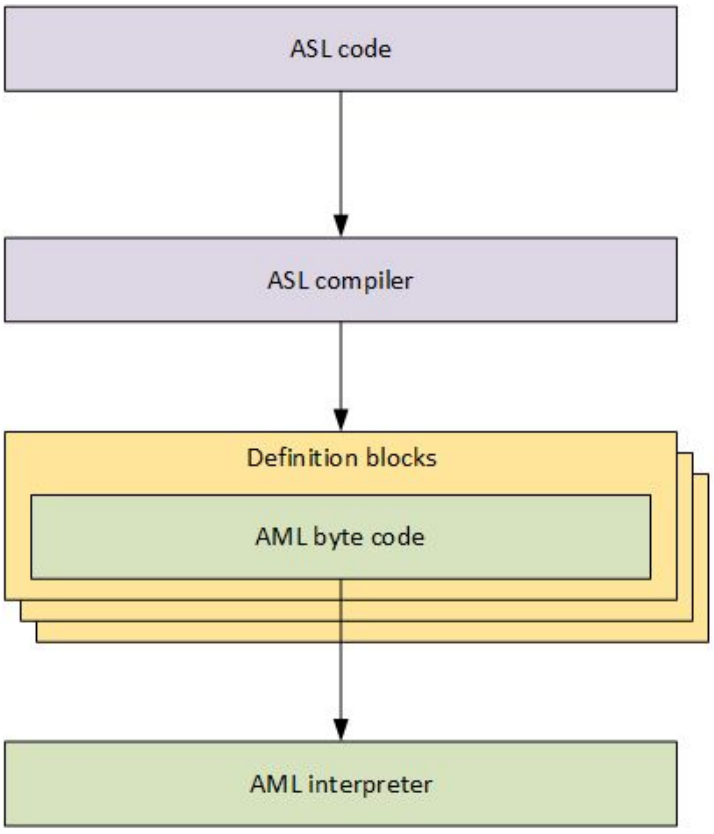
可枚举对象的集合最终构成了操作系统内的
"ACPI名字空间"(ACPI namespace)

可枚举对象(Object)
要么拥有一个直接定义的值，
要么必须由AML解释器执行

由操作系统控制的AML解释器，
按照可枚举对象的指引与硬件交互，
让硬件(System hardware)执行各种操作

图2：ACPI结构

ASL(ACPI Source Language) ASL ACPI ASL ASL AML(ACPI) AML ACPI AML ”3”



ASL代码(ASL code)定义了对象与控制方法

ASL编译器(ASL compiler)将ASL代码
编译为AML字节码(AML byte code)

每个ACPI定义块(definition block)都由
一个标识表头(identifying table header)和
一段能被AML解释器(AML interpreter)执行的
AML字节码(AML byte code)组成

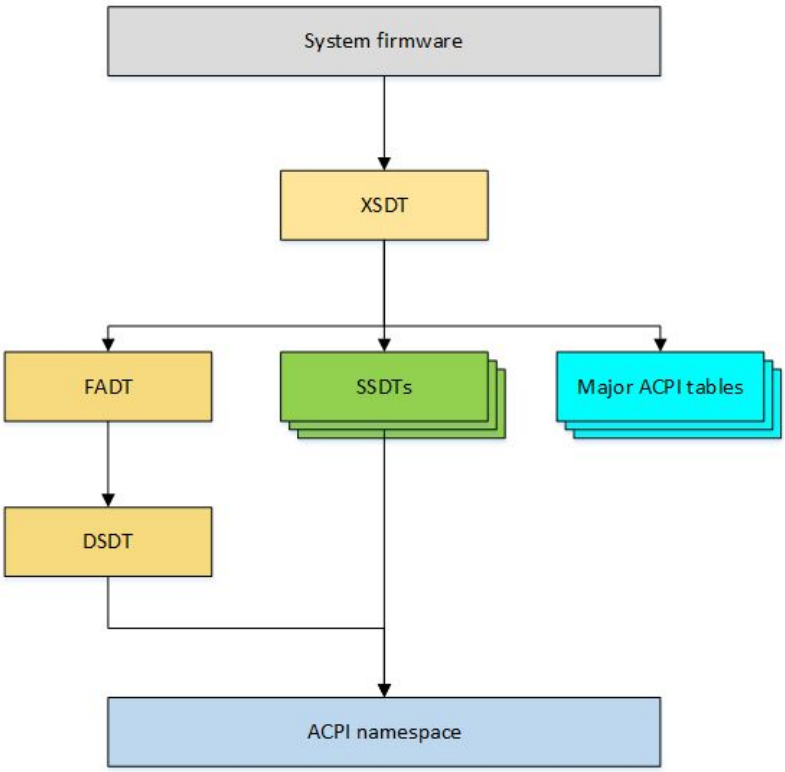
图3：ASL与AML

AML AML / I/O PCI ” ” AML

”ACPI ” ACPI ACPI (PCI USB) ACPI
_HID ( ) ACPI _ADR ( ) ACPI ACPI _ADR ACPI
PCI PCI ACPI ACPI PCI
ACPI ACPI ACPI

ACPI initialization

ACPI ” 4” :



System firmware updates the ACPI tables as necessary with information only available at runtime before handing off control to the boostrap loader.

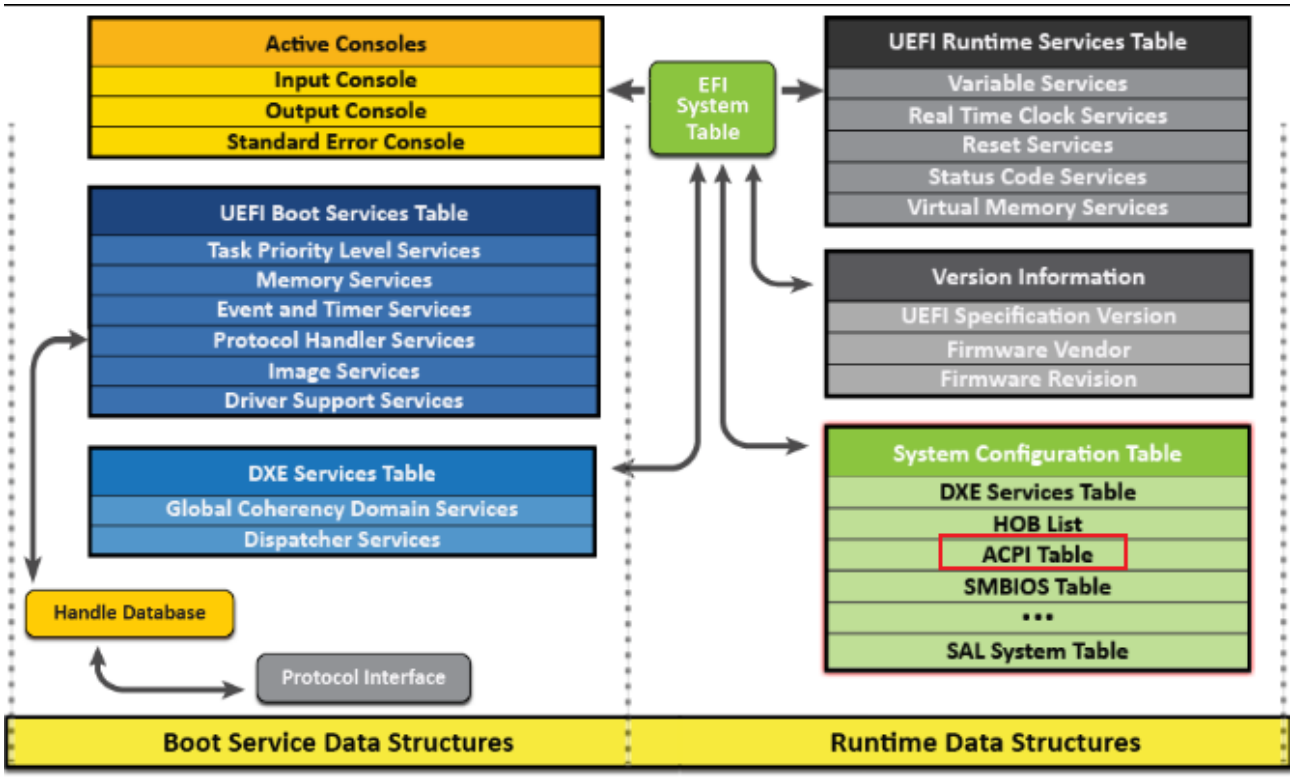
The *XSDT* is the first table used by the OS's ACPI subsystem and contains the addresses of most of the other ACPI tables on the system.

The *XSDT* points to the *FADT*, the *SSDTs*, and other *major ACPI tables*.

The FADT directs the ACPI subsystem to the *DSDT*, which is the beginning of the namespace by virtue of being the first table that contains a definition block.

The ACPI subsystem then consumes the DSDT and begins building the *ACPI namespace* from the definition blocks. The XSDT also points to the SSDTs and adds them to the namespace.

ACPI (XSDT) ACPI ACPI XSDT ACPI (FADT)
ACPI DSDT ACPI XSDT (SSDT) ACPI
ACPI _HID



ACPI EDK2

system table -> system configuration table -> ACPI table

system table OS OS

ACPI table system configuration table entry entry GUID

```
///
/// Contains a set of GUID/pointer pairs comprised of the ConfigurationTable field in the
/// EFI System Table.
///
typedef struct {
  ///
  /// The 128-bit GUID value that uniquely identifies the system configuration table.
  ///
  EFI_GUID                            VendorGuid;
  ///
  /// A pointer to the table associated with VendorGuid.
  ///
  VOID                                *VendorTable;
} EFI_CONFIGURATION_TABLE;
```

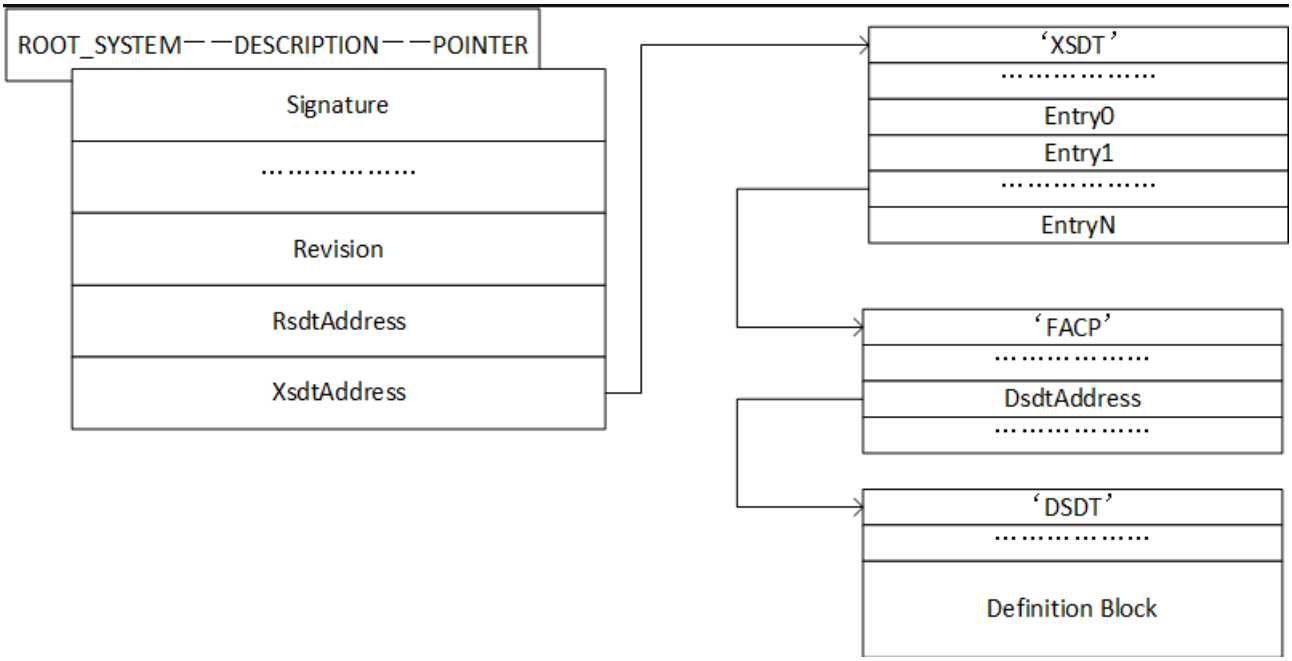
ACPI GUID MdePkg\Include\Guid\Acpi.h

```
#define ACPI_TABLE_GUID \
  { \
    0xeb9d2d30, 0x2d88, 0x11d3, {0x9a, 0x16, 0x0, 0x90, 0x27, 0x3f, 0xc1, 0x4d } \
  }

#define EFI_ACPI_TABLE_GUID \
  { \
```

```
    0x8868e871, 0xe4f1, 0x11d3, {0xbc, 0x22, 0x0, 0x80, 0xc7, 0x3c, 0x88, 0x81 } \
  }

#define ACPI_10_TABLE_GUID     ACPI_TABLE_GUID

//
// ACPI 2.0 or newer tables should use EFI_ACPI_TABLE_GUID.
//
#define EFI_ACPI_20_TABLE_GUID EFI_ACPI_TABLE_GUID

extern EFI_GUID gEfiAcpiTableGuid;
extern EFI_GUID gEfiAcpi10TableGuid;
extern EFI_GUID gEfiAcpi20TableGuid;
```

ACPI table

1.0 2.0



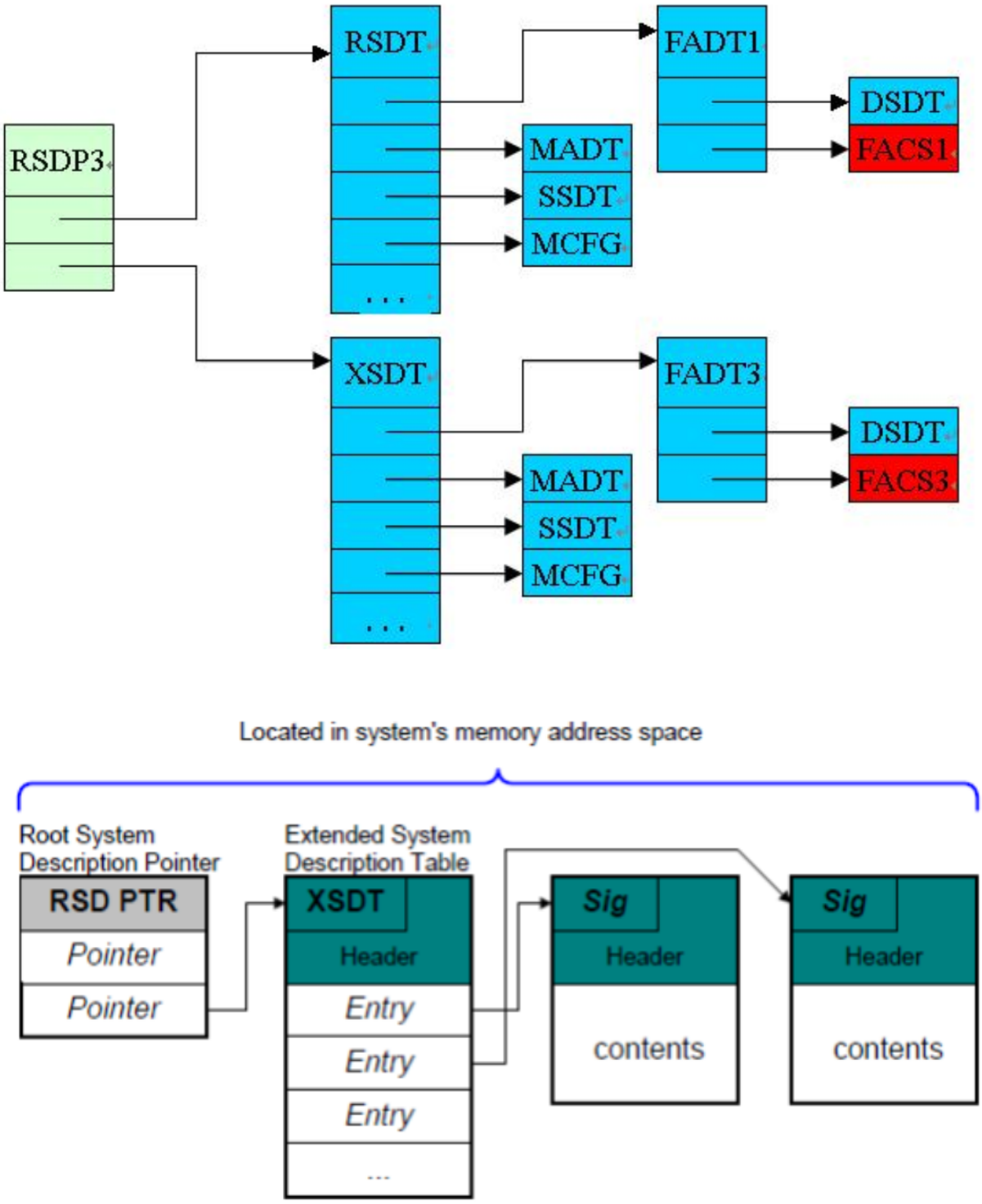


Fig. 5.1: Root System Description Pointer and Table

ACPI table entry RSDP RSDP XSDT

RSDT XSDT

- RSDT 32 4GB ACPI
- XSDT 64 64

- RSDT ACPI 1.0 32
- XSDT ACPI 2.0 64 32

- RSDT 32
- XSDT 64

Linux Kernel ACPI

ACPI

ACPI table

ACPI table XSDT 36 entry RSDP XSDT FADT Fixed ACPI Description Table DSDT Differentiated System Description Table FACS Firmware ACPI Control Structure

```
Signature   "XSDT"
Length  0x00000144 (324)
Revision    0x01 (1)
Checksum    0x34 (52)
OEM ID  "LENOVO"
OEM Table ID    "CB-01   "
OEM Revision    0x00000001 (1)
Creator ID  "    "
Creator Revision    0x01000013 (16777235)
Entry0  0x0000000044BC3000 (FACP)
Entry1  0x0000000044B2E000 (UEFI)
Entry2  0x0000000044BF3000 (SSDT)
Entry3  0x0000000044BF2000 (SSDT)
Entry4  0x0000000044BEC000 (SSDT)
Entry5  0x0000000044BE8000 (SSDT)
Entry6  0x0000000044BE4000 (SSDT)
Entry7  0x0000000044BD6000 (SSDT)
Entry8  0x0000000044BD5000 (SSDT)
Entry9  0x0000000044BD4000 (TPM2)
Entry10 0x0000000044BD3000 (SSDT)
Entry11 0x0000000044BD2000 (SSDT)
Entry12 0x0000000044BD1000 (MSDM)
Entry13 0x0000000044BCF000 (SSDT)
Entry14 0x0000000044BCE000 (LPIT)
Entry15 0x0000000044BCD000 (WSMT)
Entry16 0x0000000044BCC000 (SSDT)
Entry17 0x0000000044BC9000 (SSDT)
Entry18 0x0000000044BC8000 (DBGP)
Entry19 0x0000000044BC7000 (DBG2)
Entry20 0x0000000044BC4000 (NHLT)
Entry21 0x0000000044BFD000 (ECDT)
Entry22 0x0000000044BC2000 (HPET)
Entry23 0x0000000044BC1000 (APIC)
Entry24 0x0000000044BC0000 (MCFG)
```

```
Entry25 0x0000000044B58000 (SSDT)
Entry26 0x0000000044B56000 (SSDT)
Entry27 0x0000000044B55000 ($H2O)
Entry28 0x0000000044B54000 (DMAR)
Entry29 0x0000000044B53000 (SSDT)
Entry30 0x0000000044B4F000 (SSDT)
Entry31 0x0000000044B4B000 (SSDT)
Entry32 0x0000000044B4A000 (SSDT)
Entry33 0x0000000044B49000 (FPDT)
Entry34 0x0000000044B48000 (BGRT)
Entry35 0x0000000044B47000 (PHAT)
```

RSDP Root System Description Pointer

- RSDT XSDT
-

RSDT Root System Description Table XSDT Extended System Description Table

- ACPI
- RSDT 32 XSDT 64

FADT Fixed ACPI Description Table

-
-

DSDT Differentiated System Description Table

- AML
-

FACS Firmware ACPI Control Structure

-
-

Memory ACPI table

ACPI `SRAT` System Resource Affinity Table `SLIT` System Locality Information Table

MCFG Memory Configuration Table

- MCFG ACPI DRAM Dynamic Random Access Memory MMIO Memory-Mapped I/O / MCFG DRAM MMIO

SRAT System Resource Affinity Table

`SRAT` CPU I/O NUMA Non-Uniform Memory Access `SRAT`

-
-
- CPU
- I/O

`SRAT`

```
struct ACPI_TABLE_SRAT {
    struct ACPI_TABLE_HEADER Header;     // ACPI
    uint32_t Reserved1;                  //
    uint64_t Reserved2;                  //
    struct ACPI_SRAT_ENTRY Entries[];    //
};

struct ACPI_SRAT_ENTRY {
    uint8_t Type;                        //    CPU
    uint8_t Length;                      //
    //        Type
};

//
struct ACPI_SRAT_MEM_AFFINITY {
    uint8_t Type;                        //    0x01
    uint8_t Length;                      //
    uint32_t ProximityDomain;            //
    uint16_t Reserved1;                  //
    uint16_t Flags;                      //
    uint64_t BaseAddress;                //
    uint64_t Length;                     //
    uint32_t Reserved2;                  //
    uint32_t Reserved3;                  //
};
```

SLIT System Locality Information Table

SLIT SLIT

```
struct ACPI_TABLE_SLIT {
    struct ACPI_TABLE_HEADER Header;     // ACPI
    uint64_t LocalityCount;              //
    uint8_t Entry[];                     //
};
```

CXL ACPI table

CEDT ACPI CXL Early Discovery Table CXL

CEDT CXL CXL

CEDT CEDT CXL CXL CXL

CEDT CEDT CXL Entries

1. **CXL Host Bridge Structure** CXL
2. **CXL Device Structure** CXL ID
3. **CXL Switch Structure** CXL Switch

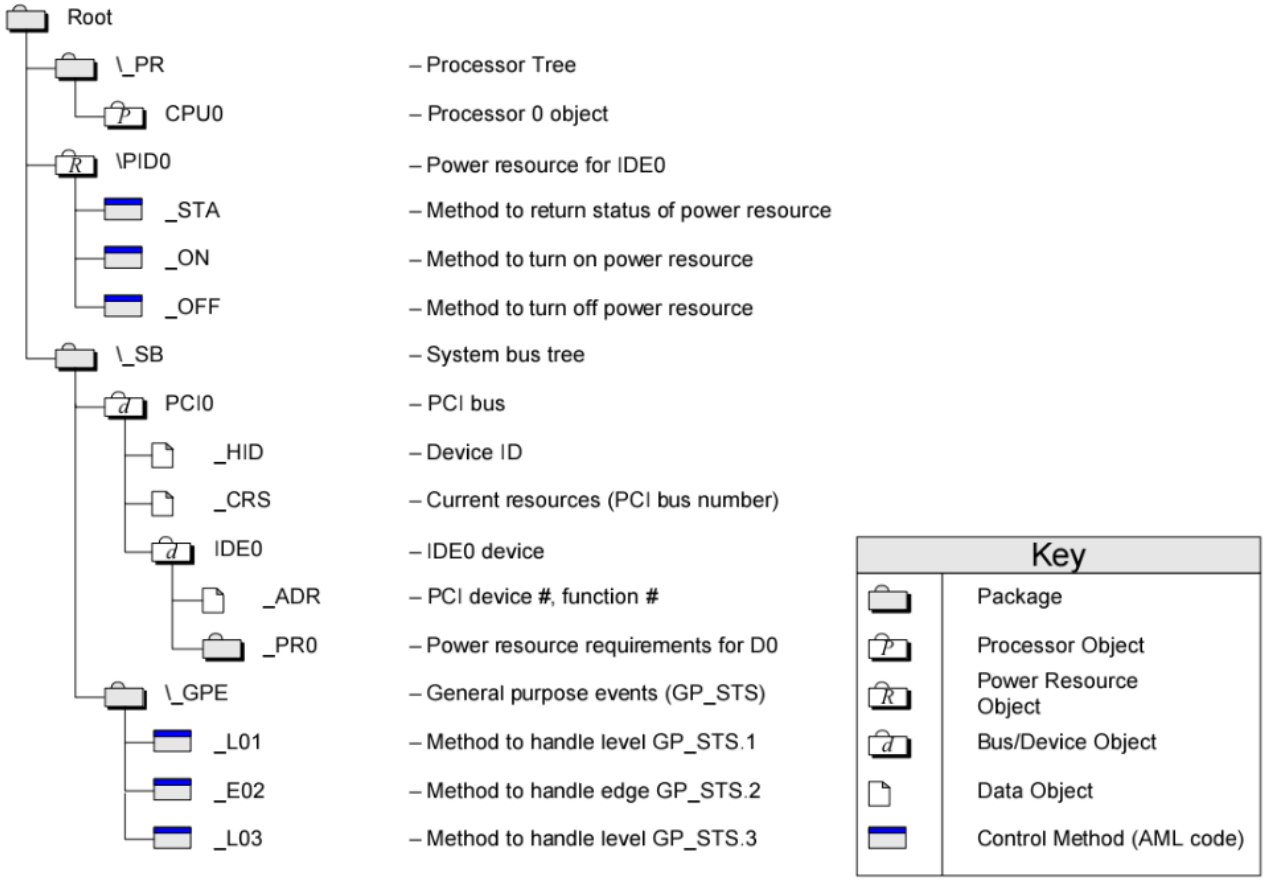
CXL CXL

CEDT

```
struct cedt_header {
    uint32_t signature;    // 'CEDT'
    uint32_t length;       // Length of the entire table
    uint8_t revision;      // Revision of the CEDT table
    uint8_t checksum;      // Checksum of the entire table
    char oem_id[6];        // OEM ID
    char oem_table_id[8];  // OEM Table ID
    uint32_t oem_revision; // OEM Revision
    uint32_t creator_id;   // Creator ID
    uint32_t creator_rev;  // Creator Revision
};

struct cxl_host_bridge {
    uint16_t type;         // Entry type (CXL Host Bridge)
    uint16_t length;       // Length of this entry
    uint32_t host_bridge_id; // Host Bridge ID
    // Additional fields specific to the CXL Host Bridge
};

struct cxl_device {
    uint16_t type;         // Entry type (CXL Device)
    uint16_t length;       // Length of this entry
    uint32_t device_id;    // Device ID
    // Additional fields specific to the CXL Device
};

// Main CEDT Table containing multiple entries
struct cedt_table {
    struct cedt_header header;
    struct cxl_host_bridge host_bridge;
    struct cxl_device device;
    // Other entries as needed
};
```

DSDT SSDT

ACPI namespace



Name	Description
_GPE	General events in GPE register block.

Name	Description
_PR	ACPI 1.0 Processor Namespace. ACPI 1.0 requires all Processor objects to be defined under this namespace. ACPI 2.0 and later allow Processor object definitions under the _SB namespace. Platforms may maintain the _PR namespace for compatibility with ACPI 1.0 operating systems, but it is otherwise deprecated. see the compatibility note in Section 5.2.12.12. An ACPI-compatible namespace may define Processor objects in either the _SB or _PR scope but not both. For more information about defining Processor objects, see Section 8, “Processor Configuration and Control.”
_SB	All Device/Bus Objects are defined under this namespace.
_SI	System indicator objects are defined under this namespace. For more information about defining system indicators, see Section 9.2, _SI System Indicators.”
_TZ	ACPI 1.0 Thermal Zone namespace. ACPI 1.0 requires all Thermal Zone objects to be defined under this namespace. Thermal Zone object definitions may now be defined under the _SB namespace. ACPI-compatible systems may maintain the _TZ namespace for compatibility with ACPI 1.0 operating systems. An ACPI-compatible namespace may define Thermal Zone objects in either the _SB or _TZ scope but not both. For more information about defining Thermal Zone objects, see Section 11, “Thermal Management.”

table

asl aml aml raw raw ffs

ffs aml memory protocol install

```
// Example SSDT: SSDT_MYCUSTOM.aml

DefinitionBlock ("SSDT_MYCUSTOM.aml", "SSDT", 1, "YOURID", "YOURID", 1)
{
    // Insert your ACPI table content here
    // Example:
    Method (_STA, 0, NotSerialized)  // _STA: Status
    {
        Return (0x0F) // indicate device is present and working
    }
}
```

1. **DefinitionBlock**
   - `DefinitionBlock` ACPI `SSDT_MYCUSTOM.aml` `SSDT` Secondary System Description Table ACPI `1` OEM ID `"YOURID"` OEM ID `"YOURID"`
2. **Method (_STA)**
   - ACPI `Method (_STA, 0, NotSerialized)` `_STA` `0` `NotSerialized`
   - `_STA` `Return (0x0F)` `0x0F`

EDK2 protocol

Status = gBS->LocateProtocol (&gEfiAcpiTableProtocolGuid, NULL, (VOID **) &AcpiTable);

Status = AcpiTable->InstallAcpiTable (AcpiTable, Table, TableSize, &TableKey); uninstall ACPI table

table device tree

https://www.nirsoft.net/utils/firmware_tables_view.html

RW

UEFI Shell https://acpica.org/downloads/uefi-support

acpi windwos tool download link: `https://acpica.org/downloads/binary-tools`

UEFI Shell https://github.com/andreiw/UefiToolsPkg/tree/master/Applications

UEFI Shell acpiview from UefiShellAcpiViewCommandLib.inf

Advanced Configuration and Power Interface (ACPI) Introduction and Overview

(ACPI)

ACPI Spec

https://www.lab-z.com/revmem/

ACPI Source Language (ASL) Tutorial

4.2 CEDT

[toc]

CEDT ACPI CXL Early Discovery Table CXL

CEDT CXL CXL

CEDT

CEDT CXL CXL CXL

CEDT

CEDT CXL Entries

1. **CXL Host Bridge Structure** CXL
2. **CXL Device Structure** CXL ID
3. **CXL Switch Structure** CXL Switch

CXL CXL

CEDT

Table 9-19. CEDT Header

Field	Length in Bytes	Byte Offset	Description
Header:			
Signature	4	00h	'CEDT'. Signature for the CXL Early Discovery Table.
Length	4	04h	Length, in bytes, of the entire CEDT.
Revision	1	08h	Value is 1.
Checksum	1	09h	Entire table must sum to 0.
OEM ID	6	0Ah	OEM ID
OEM Table ID	8	10h	Manufacturer Model ID
OEM Revision	4	18h	OEM Revision
Creator ID	4	1Ch	Vendor ID of the utility that created the table.
Creator Revision	4	20h	Revision of the utility that created the table.
CEDT Structure[n]	Varies	24h	A list of CEDT structures for this implementation.

Table 9-20. CEDT Structure Types

Value	Description
0	CXL Host Bridge Structure (CHBS)
1	CXL Fixed Memory Window Structure (CFMWS)
2	CXL XOR Interleave Math Structure (CXIMS)
3	RCEC Downstream Port Association Structure (RDPAS)
4-255	Reserved

Table 9-21. CHBS Structure

Field	Length in Bytes	Byte Offset	Description
Type	1	00h	=0 to indicate that this is a CHBS entry
Reserved	1	01h	Reserved
Record Length	2	02h	Length of this record (20h).
UID	4	04h	CXL Host Bridge Unique ID. Used to associate a CHBS instance with a CXL Host Bridge instance. The value of this field shall match the output of _UID under the associated CXL Host Bridge in ACPI namespace.
CXL Version	4	08h	• 0000 0000h: RCH • 0000 0001h: Host Bridge that is associated with one or more CXL root ports
Reserved	4	0Ch	Reserved
Base	8	10h	• If CXL Version = 0000 0000h, this represents the base address of the RCH Downstream Port RCRB • If CXL Version = 0000 0001h, this represents the base address of the CHBCR See Table 8-17 for more details.
Length	8	18h	• If CXL Version = 0000 0000h, this field must be set to 8 KB (2000h) • If CXL Version = 0000 0001h, this field must be set to 64 KB (1 0000h)

```
#define INTERLEAVE_TARGETS  1

#pragma pack (1)

typedef struct {
  EFI_ACPI_CEDT_CFMWS_STRUCTURE  Cfmws;
  UINT32  InterleaveTarget[INTERLEAVE_TARGETS];
} EFI_ACPI_CEDT_CFMWS_AND_INTERLEAVE_TARGET_STRUCTURE;

typedef struct {
  EFI_ACPI_DESCRIPTION_HEADER  Header;
  EFI_ACPI_CEDT_CFMWS_AND_INTERLEAVE_TARGET_STRUCTURE CfmwsTarget;
  EFI_ACPI_CEDT_CHBS_STRUCTURE  Chbs;
} EFI_ACPI_CEDT_STRUCTURE_TABLE;

#pragma pack ()

STATIC EFI_ACPI_CEDT_STRUCTURE_TABLE Cedt = {
  ARM_ACPI_HEADER (
    EFI_ACPI_6_4_CEDT_SIGNATURE,
    EFI_ACPI_CEDT_STRUCTURE_TABLE,
    1
  ),
  {
    {
    // CFMWS
      {
        1,
        EFI_ACPI_RESERVED_BYTE,
```

```
        sizeof (EFI_ACPI_CEDT_CFMWS_AND_INTERLEAVE_TARGET_STRUCTURE)
      },
      EFI_ACPI_RESERVED_DWORD,
      0x3fe00000000,
      0x200000000,
      0,
      0,
      EFI_ACPI_RESERVED_WORD,
      4,
      2,
      0
    },
    {
    //Interleave target list
      1
    }
  },
  {
  // CHBS
    {
      0,
      EFI_ACPI_RESERVED_BYTE,
      sizeof (EFI_ACPI_CEDT_CHBS_STRUCTURE)
    },
    1,
    1,
    EFI_ACPI_RESERVED_DWORD,
    0x10D0000000,
    0x1000
  }
};

//
// Reference the table being generated to prevent the optimizer from removing
// the data structure from the executable
//
VOID* CONST ReferenceAcpiTable = &Cedt;
```

CXL Type-3 device discovery, configuration in firmware and prepare ACPI tables for kernel usage

EDK2 CEDT Table

CXL Spec3.0 Chapter 9.17

Chapter 5

AMBA

5.1 PCIe

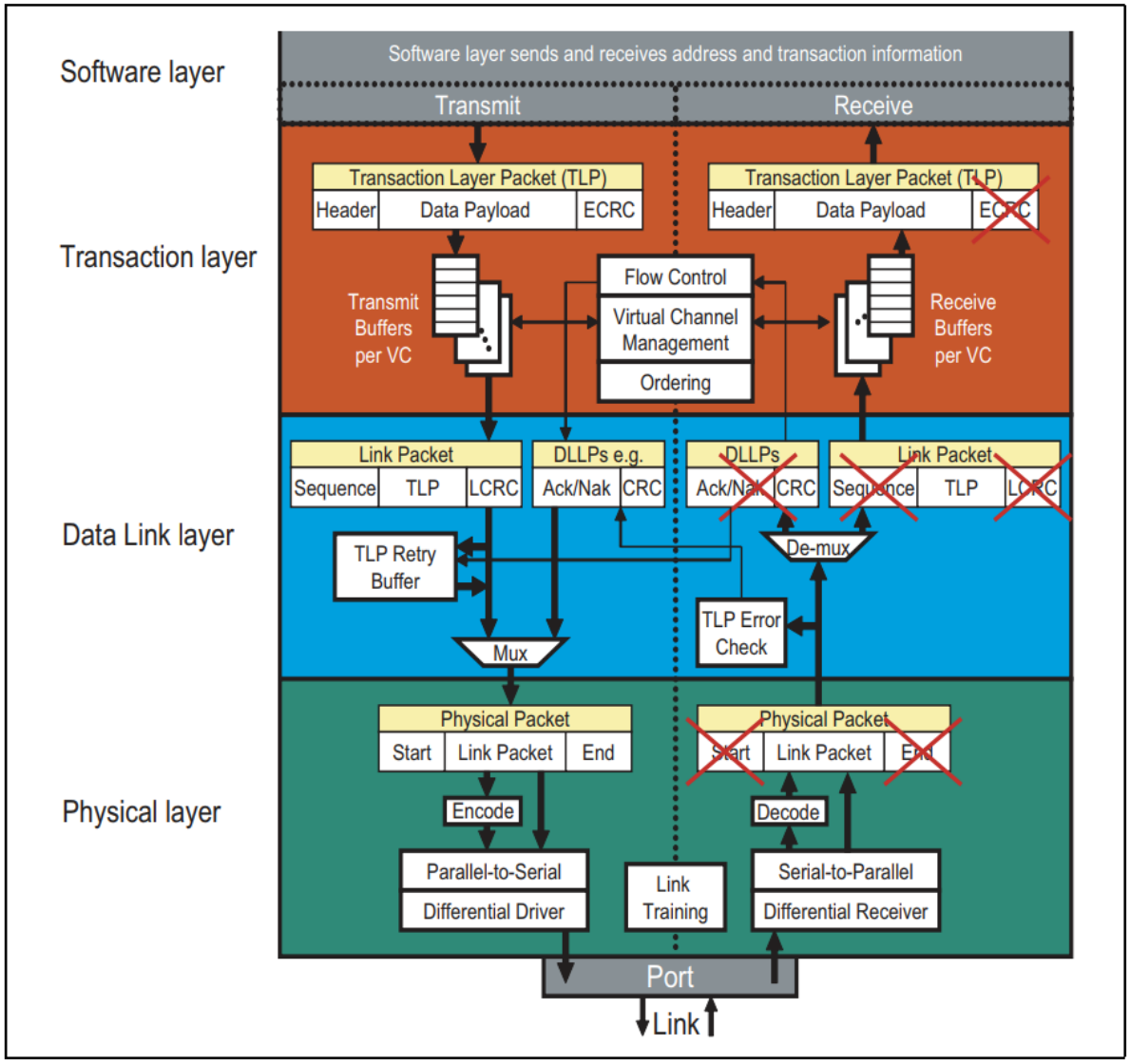
5.2 PCIe

5.2.1

Link Lane EQ LTSSM

Introduction

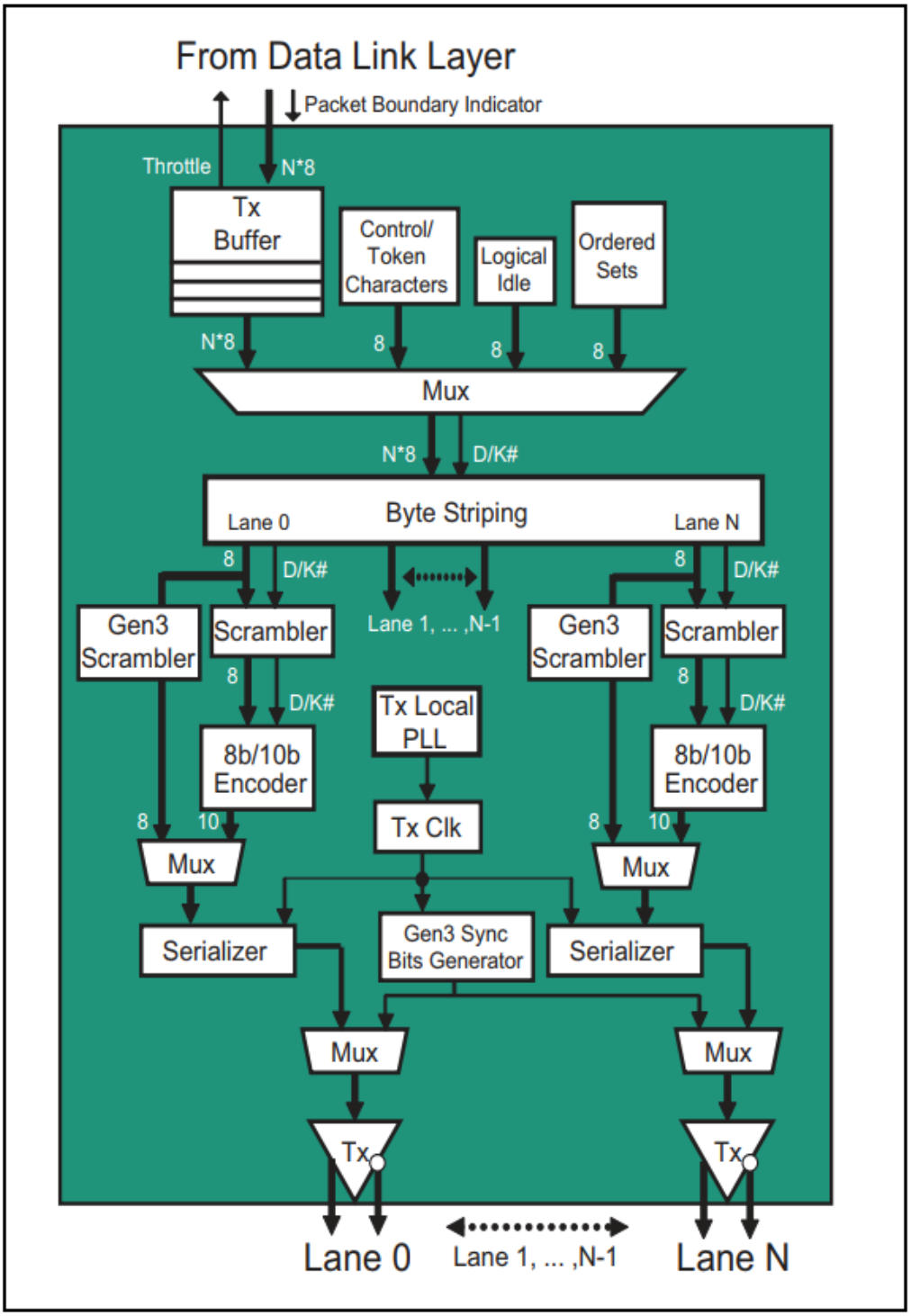
Figure 11-1: PCIe Port Layers



- Logical Sub-Block
- Electrical Sub-Block

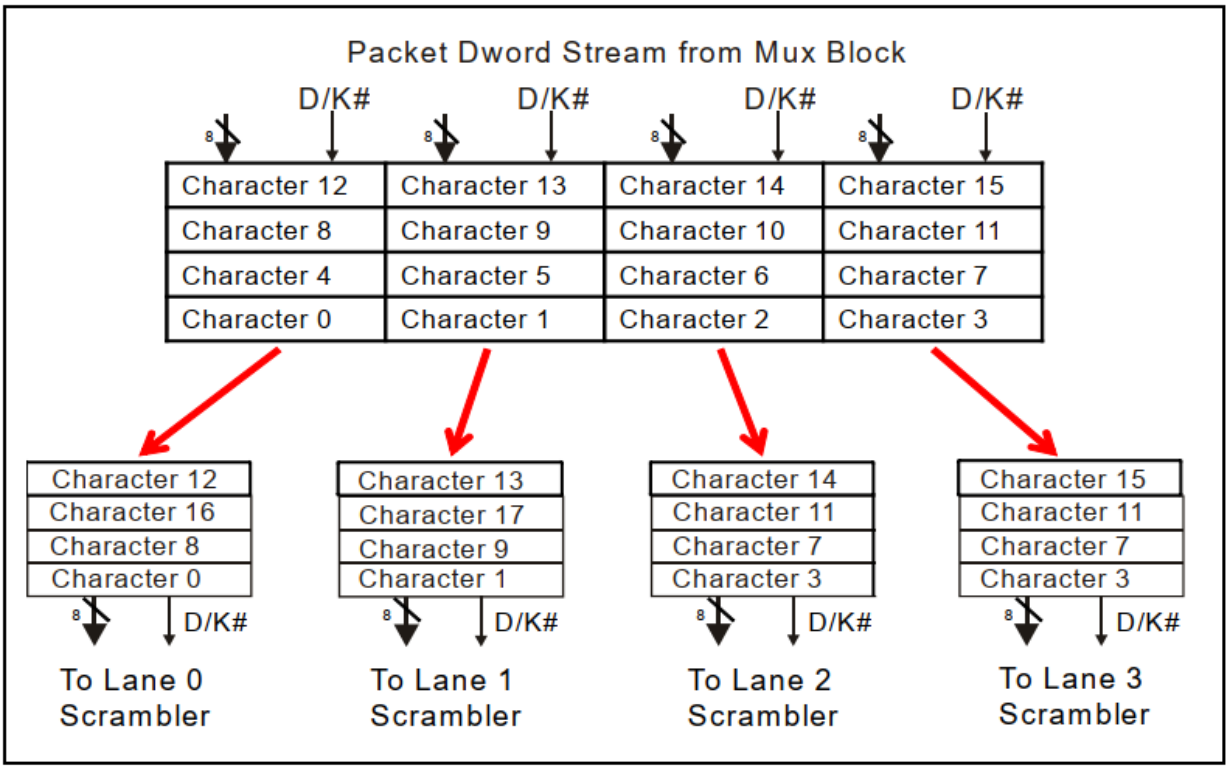
Logical Sub-Block

Figure 11-3: Physical Layer Transmit Details



Tx Buffer & Mux buffer DL Mux DL skip buffer

Byte Striping lane



Data Scrambling Scrambling pseudo-random number generator

EMI 0 1 01

PCIe —— **XOR** **LFSR Linear-feedback shift register**

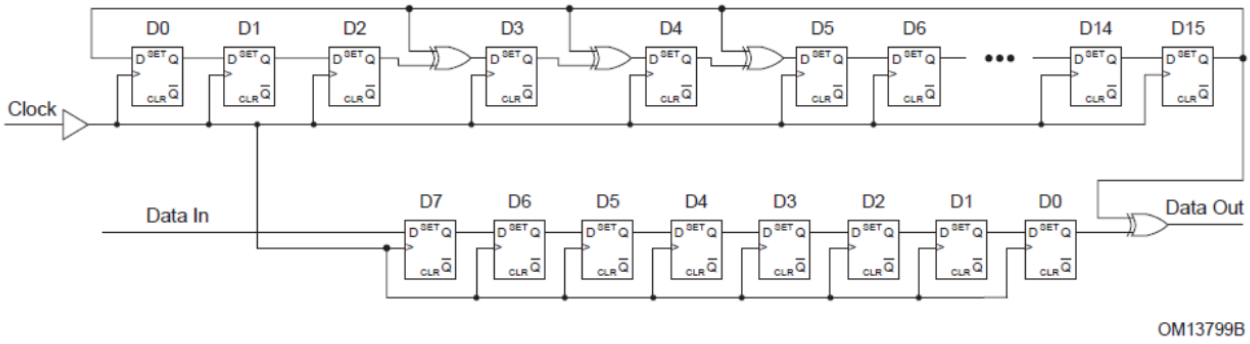


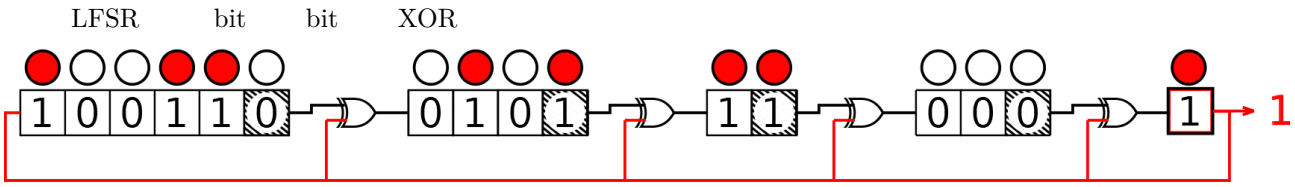
Figure 4-10 LFSR with 8b/10b Scrambling Polynomial

PCIe Galois LFSR PCIe 1.0 2.0 16 LFSR

$$G(X) = X^{16} + X^5 + X^4 + X^3 + 1$$

3.0 23 LFSR

$$G(X) = X^{23} + X^{21} + X^{16} + X^8 + X^5 + X^2 + 1$$



PCIe1.0 2.0 LFSR Seed 0xFFFF PCIe3.0 Lane Lane LFSR Seed 8 Lane ID 8

Lane	Seed
0	1DBFBCh
1	0607BBh
2	1EC760h
3	18C0DBh
4	010F12h
5	19CFC9h
6	0277CEh
7	1BB807h

XOR XOR Seed LFSR

Encoding Encoding 0 1 DC DC Balance PCIe Clock Recovery

PCIe 8b/10b 128b/130b 242B/256B FLIT PCIe Lane PCIe 1.0 8b/10b 8bit 10b

$$\frac{2.5GT/s \times \frac{8bits}{10bits}}{8bits} = 250MB/s$$

PCIe

PCIe Version	Line code	Transfer rate per lane	Throughput x1	Throughput x16
1.0	8b/10b	2.5 GT/s	250 MB/s	4 GB/s
2.0	8b/10b	5 GT/s	500 MB/s	8 GB/s
3.0	128b/130b	8 GT/s	984.6 MB/s	15.75 GB/s
4.0	128b/130b	16 GT/s	1.969 GB/s	31.51 GB/s
5.0	128b/130b	32 GT/s	3.938 GB/s	63.02 GB/s
6.0	1b/1b 242B/256B FLIT	64 GT/s	7.564 GB/s	121.00 GB/s

242B/256B B bit Byte

8b/10b 8b/10b PCIe 1.0 2.0 2.5GT/s 5GT/s [4] 8bits 5bits 3bits 6bits 4bits

8b/10b K D ABCDE FGH D.<ABCDE>.<FGH> K.<ABCDE>.<FGH>

5b/6b code (abcdei) [edit]

Input		RD = −1	RD = +1
Code	EDCBA	a b c d e i	
D.00	00000	100111	011000
D.01	00001	011101	100010
D.02	00010	101101	010010
D.03	00011	110001	
D.04	00100	110101	001010
D.05	00101	101001	
D.06	00110	011001	
D.07	00111	111000	000111
D.08	01000	111001	000110
D.09	01001	100101	
D.10	01010	010101	
D.11	01011	110100	
D.12	01100	001101	
D.13	01101	101100	
D.14	01110	011100	
D.15	01111	010111	101000
not used		**111**100	**000**011

Input		RD = −1	RD = +1
Code	EDCBA	a b c d e i	
D.16	10000	011011	100100
D.17	10001	100011	
D.18	10010	010011	
D.19	10011	110010	
D.20	10100	001011	
D.21	10101	101010	
D.22	10110	011010	
D.23 †	10111	111010	000101
D.24	11000	110011	001100
D.25	11001	100110	
D.26	11010	010110	
D.27 †	11011	110110	001001
D.28	11100	001110	
D.29 †	11101	101110	010001
D.30 †	11110	011110	100001
D.31	11111	101011	010100
K.28 ‡	11100	00**1111**	11**0000**

3b/4b code (fghj) [edit]

Input		RD = –1	RD = +1
Code	HGF	f g h j	
D.x.0	000	1011	0100
D.x.1	001	1001	
D.x.2	010	0101	
D.x.3	011	1100	0011
D.x.4	100	1101	0010
D.x.5	101	1010	
D.x.6	110	0110	
D.x.P7 †	111	1110	0001
D.x.A7 †		0111	1000

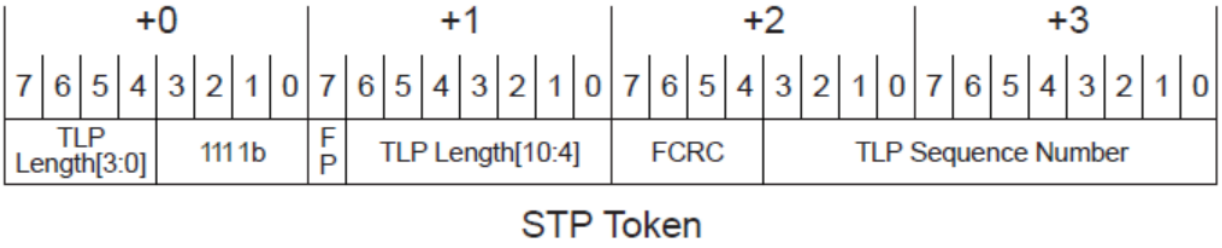
Input		RD = –1	RD = +1
Code	HGF	f g h j	
K.x.0	000	1011	0100
K.x.1 ‡	001	**0**110	**1**001
K.x.2	010	1010	0101
K.x.3	011	1100	0011
K.x.4	100	1101	0010
K.x.5 ‡	101	**0**101	**1**010
K.x.6	110	1001	0110
K.x.7 ‡	111	**0**111	**1**000

**RD Running Disparity ** 1 0 01 8b/10b RD ±1 RD ±2 8b/10b 01 DC 20% PCIe 3.0 8b/10b

128b/130b PCIe 3.0 PCIe 128b/130b 130 bits 2 bits 98.46% 128b/130b 64b/66b [5] payload 64b -> 128b 8b/10b 3.0 LFSR DC LFSR 128b/130b Framing Encoding Framing

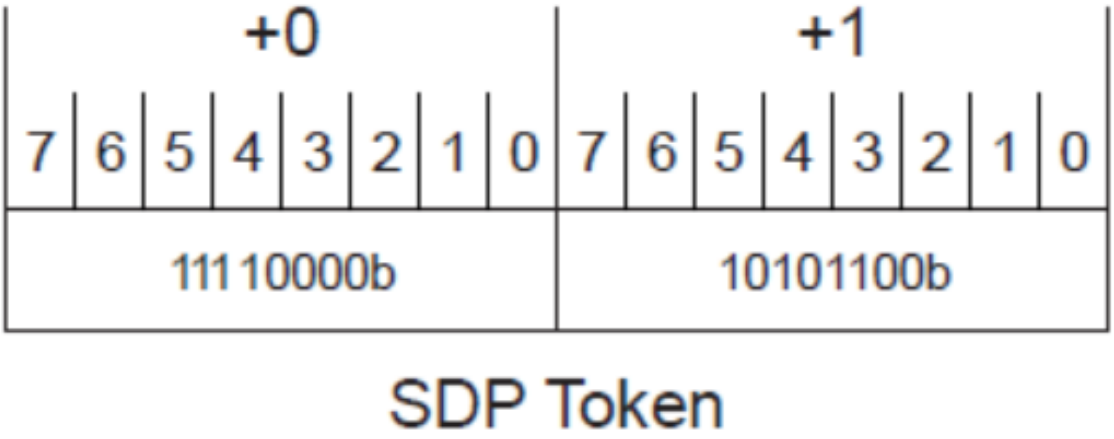
128b/130b block Token

- TLP 2 4 bits FCRC Seq 4 token —— STP Start of TLP

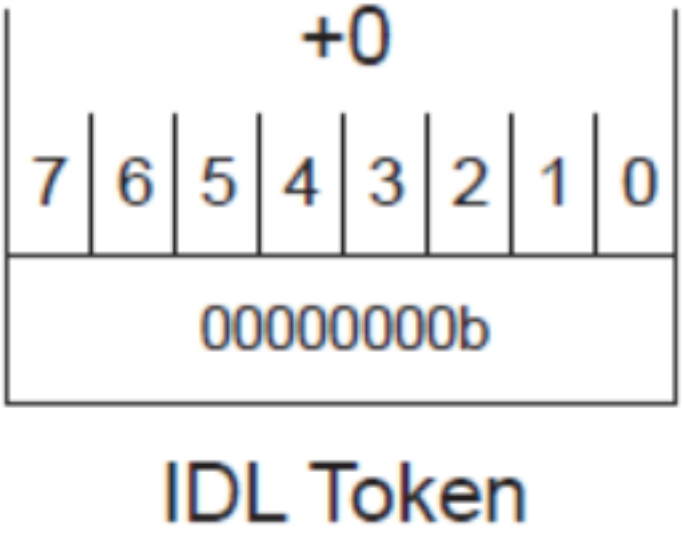


STP Token

- DLLP 2 Token - SDP Start of DLLP



Token Lane 0 Byte 0 IDL Logical Idle Nullify TLP EDB EnD Bad EDS End of Data Stream

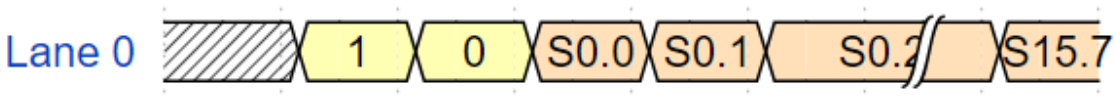


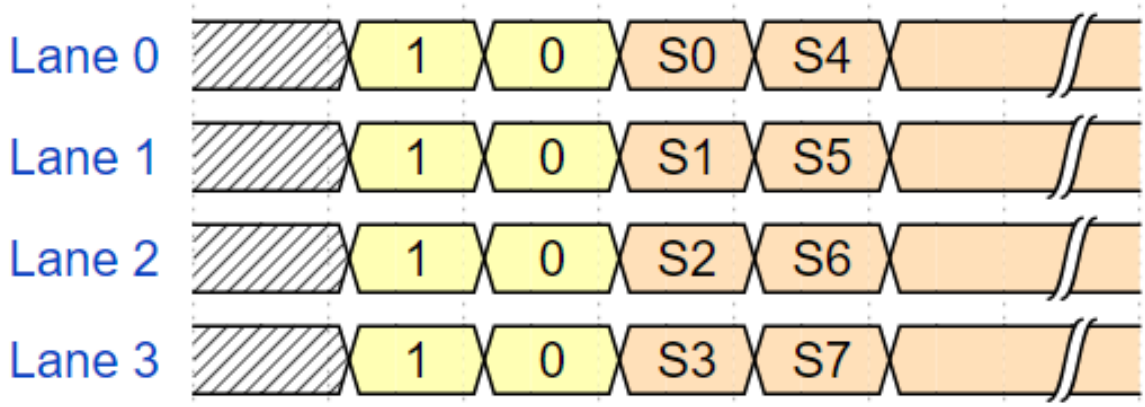
Encoding

128b/130b 128bit block 128bits payload 2bits SyncHeader payload

- 01b payload Data Block 128bits
- 10b payload Ordered Set Block 128bits

16 S0-S15 bit .0 .7 01b 10b





242B/256B FLIT 128b/130b PCIe6.0 PCIe1.0 NRZ Non-Return-To-Zero PAM4 Pulse Amplitude Modulation 4 PCIe6.0 PCIe 242B/256B FLIT

242B/256B FLIT FLIT 256 bits

x8 Lanes	0	1	2	3	4	5	6	7
256 UI								
TLP Bytes	0	1	2	3	4	5	6	7
(0-299)	8	9	10	11	12	13	14	15
	16	17	18	19	20	21	22	23
	24	25	26	27	28	29	30	31
	32	33	34	35	36	37	38	39
	40	41	42	43	44	45	46	47
	48	49	50	51	52	53	54	55
	56	57	58	59	60	61	62	63
	64	65	66	67	68	69	70	71
	72	73	74	75	76	77	78	79
	80	81	82	83	84	85	86	87
	88	89	90	91	92	93	94	95
	96	97	98	99	100	101	102	103
	104	105	106	107	108	109	110	111
	112	113	114	115	116	117	118	119
	120	121	122	123	124	125	126	127
	128	129	130	131	132	133	134	135
	136	137	138	139	140	141	142	143
	144	145	146	147	148	149	150	151
	152	153	154	155	156	157	158	159
	160	161	162	163	164	165	166	167
	168	169	170	171	172	173	174	175
	176	177	178	179	180	181	182	183
	184	185	186	187	188	189	190	191
	192	193	194	195	196	197	198	199
	200	201	202	203	204	205	206	207
	208	209	210	211	212	213	214	215
	216	217	218	219	220	221	222	223
	224	225	226	227	228	229	230	231
	232	233	234	235	dlp0	dlp1	dlp2	dlp3
	dlp4	dlp5	crc0	crc1	crc2	crc3	crc4	crc5
	crc6	crc7	ecc0	ecc0	ecc0	ecc1	ecc1	ecc1

236 TLP 6 DLLP Data Link Layer Packet 8 CRC 6 FEC Forward Error Correc-

tion FLIT CRC DLLP TLP CRC FEC FLIT

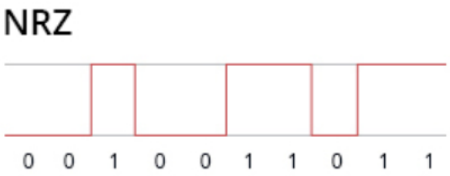
FLIT PCIe FLIT NRZ 2.5 GT/s 5.0 GT/s 8.0 GT/s 16.0 GT/s 32.0 GT/s NRZ

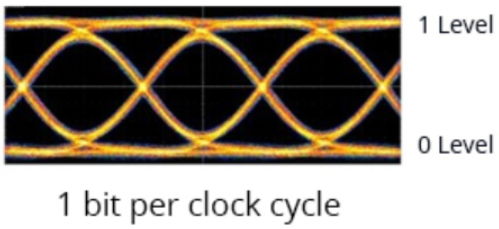
Electrical Sub-block

Parallel to Serial Parallel to Serial

Modulation

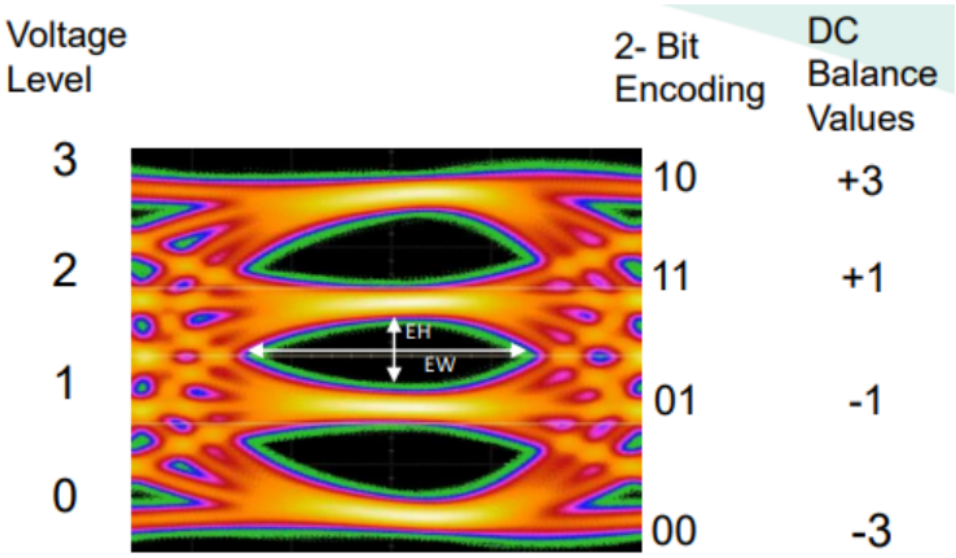
NRZ Non-Return-to-Zero PCIe1.0 5.0 PCIe NRZ Non-Return-to-Zero [10] RZ Return-to-Zero 1 PCIe Bipolar NRZ level 0 −V 1 +V



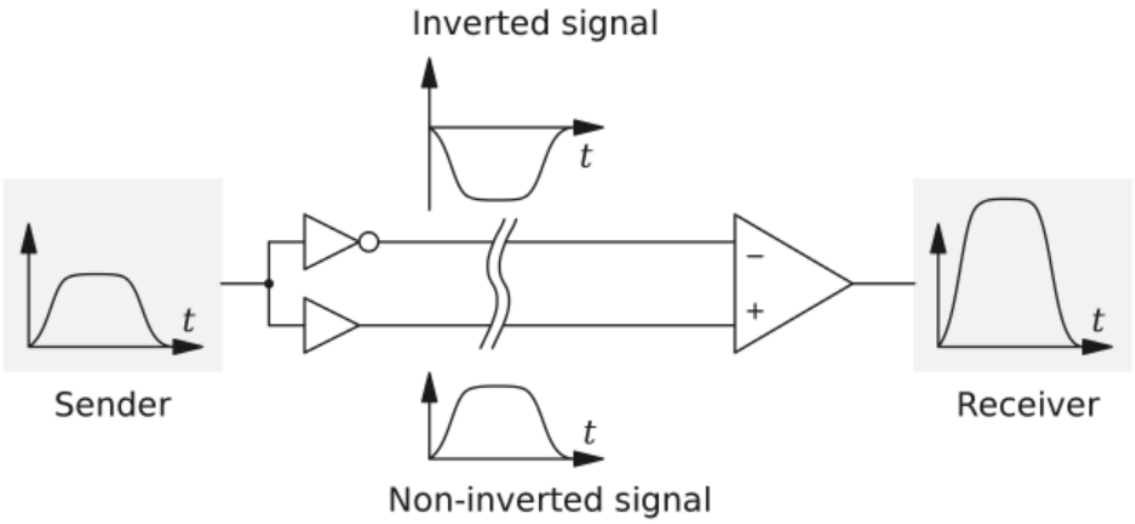


Encoding Scrambling

PAM4 PCIe6.0 PAM4 Pulse Amplitude Modulation 4 00 −V 10 +V 01 −V/3 11 +V/3



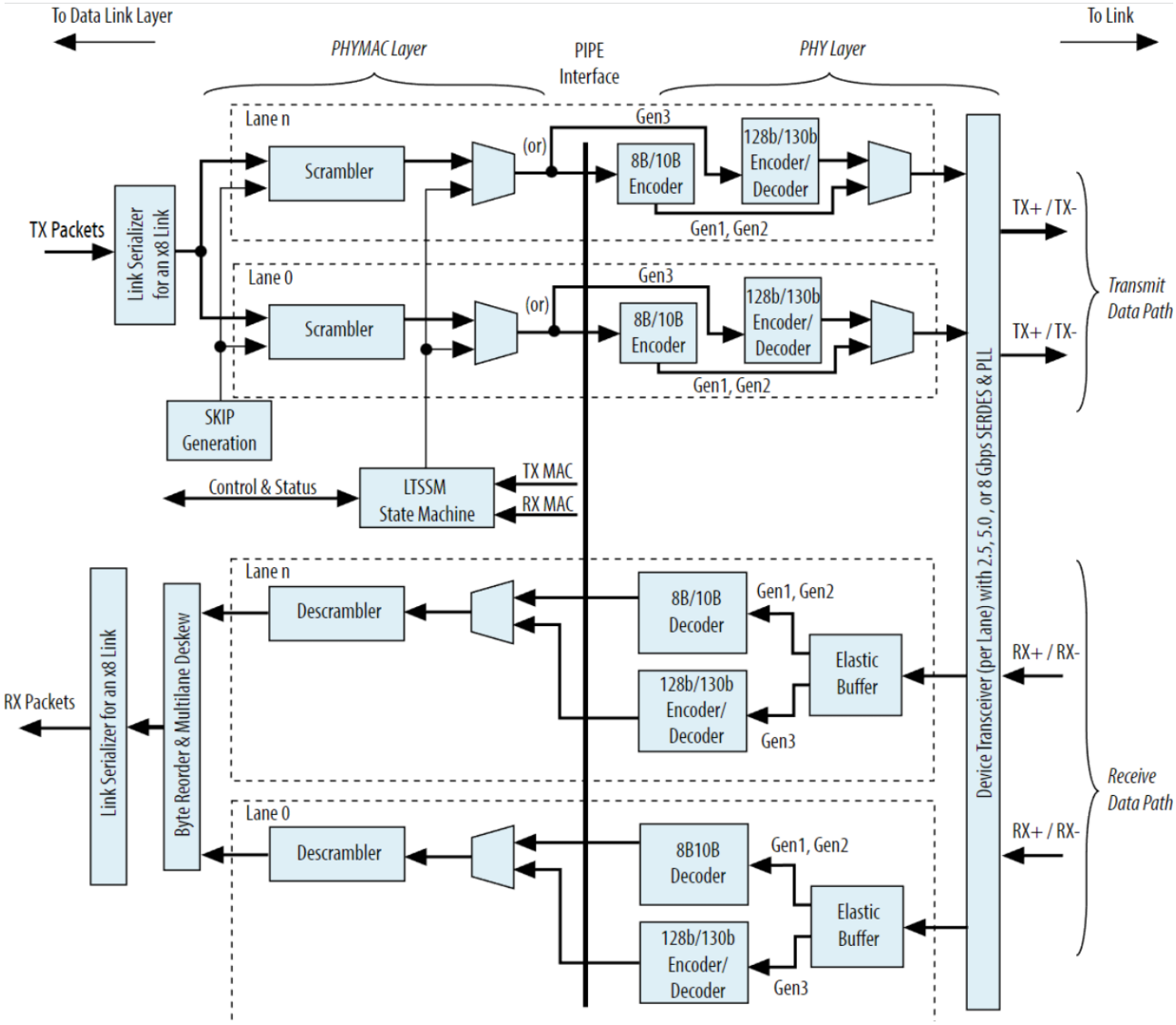
Differential Signal PCIe Differential Signaling Pinout Pair



我们假设原始信号中的电压为V_{Tx}，经过差分处理的两路信号为$+\frac{V_{Tx}}{2}$和$-\frac{V_{Tx}}{2}$，干扰导致的电压变化为ΔV_{noise}，那么忽略传输的损耗，在接收端收到的电压就是：

$$V_{Rx} = (+\frac{V_{Tx}}{2} + \Delta V_{noise}) - (-\frac{V_{Tx}}{2} + \Delta V_{noise}) = V_{Tx}$$

这样，通过差分信号，我们就抵消了信道上的干扰。



Blog

5.2.2 Data Link Layer

Introduction

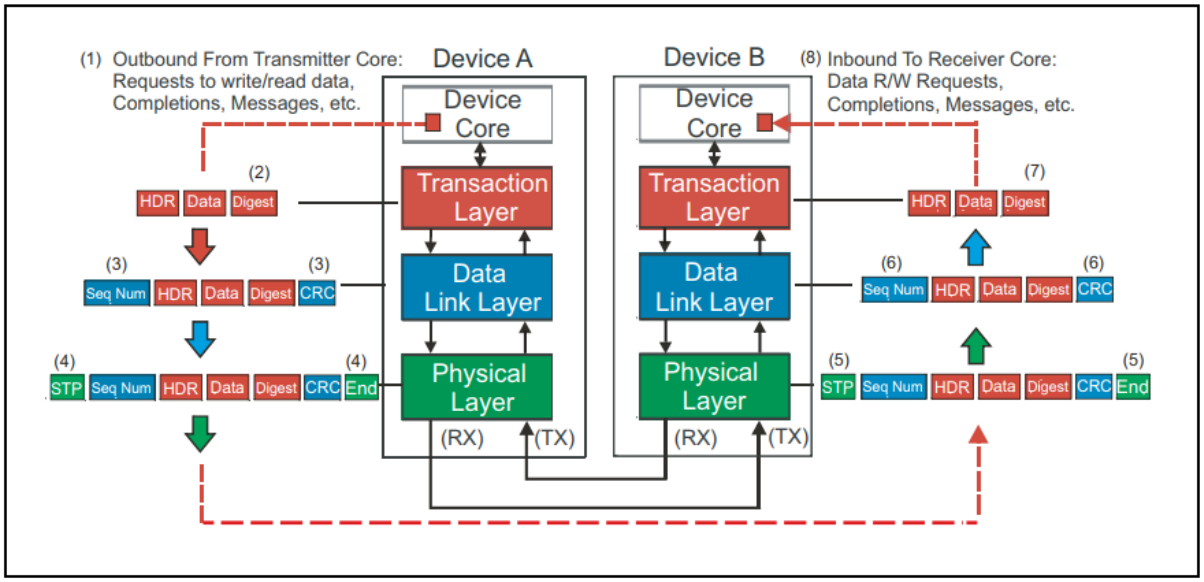
TLP Ack/Nack

Data Link Layer

TLP Feature ack/nack Token STP Start of TLP TLP SDP Start of DLLP DLLP Data Link Layer Packet

Transaction layer TLP TLP buffer TLP

Figure 5-2: PCIe TLP Assembly/Disassembly



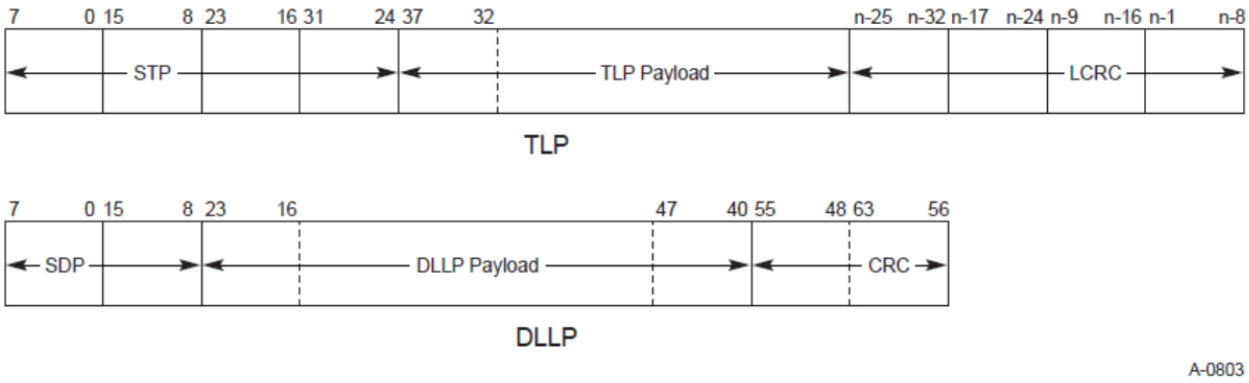


Figure 4-14 TLP and DLLP Layout

TLP

1. Sequence Number 2 Link `NEXT_RCV_SEQ`
2. CRC LCRC Link CRC 4 CRC

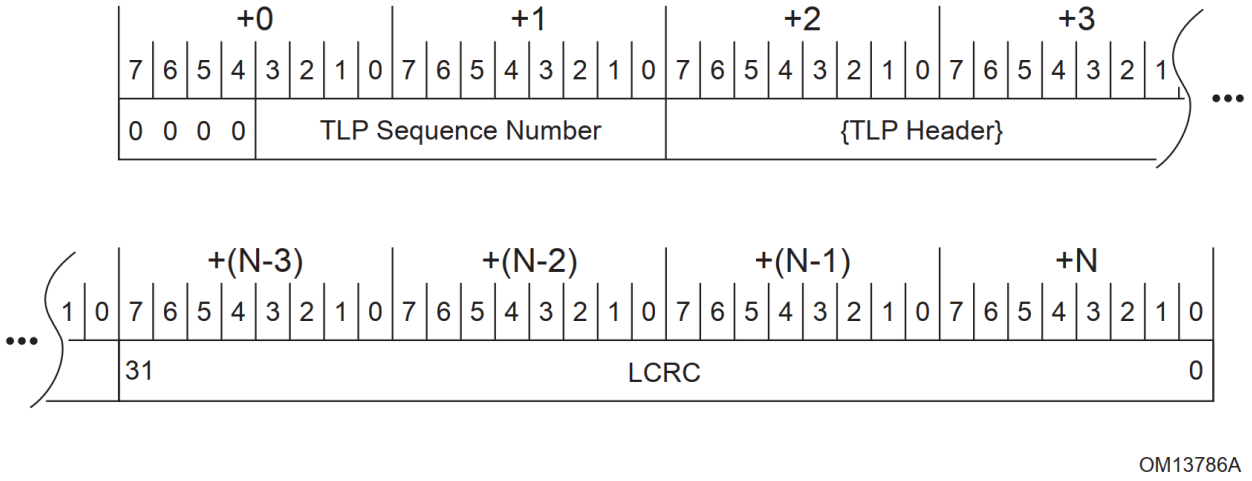


Figure 3-14 TLP with LCRC and TLP Sequence Number Applied

Retry Buffer Physical Layer ACK Seq CRC Retry Buffer

CRC Nak Ack ACK Retry Buffer-TLP

DLLP

TLP Ack Nak DLLP Data Link Layer Packet 8B DLLP 6B 2B (gen1&2 SDP end gen3 2B SDP) 8B

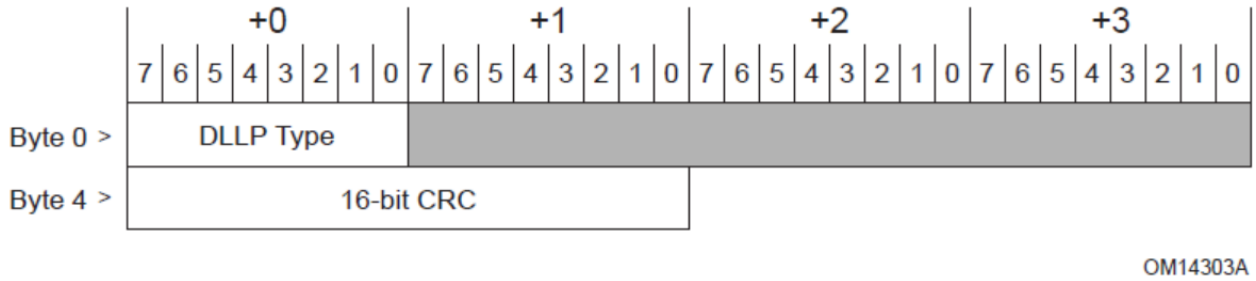


Figure 3-4 DLLP Type and CRC Fields

Table 4-1 shows the Special Symbols used for PCI Express and provides a brief description for each. These Symbols will be discussed in greater detail in following sections. Each of these Special Symbols, as well as the data Symbols, must be interpreted by looking at the 10-bit Symbol in its entirety.

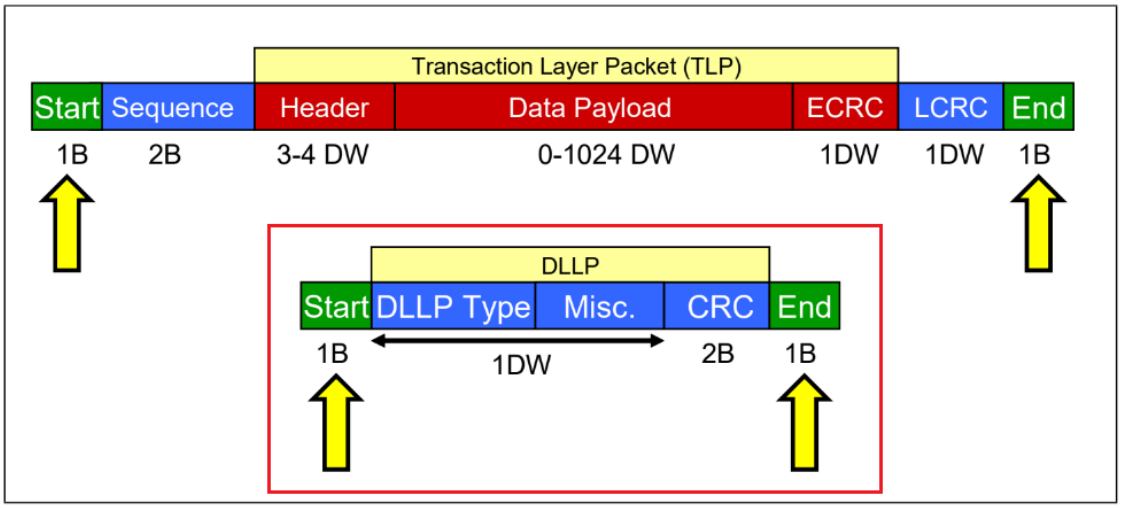
Table 4-1 Special Symbols

Encoding	Symbol	Name	Description
K28.5	COM	Comma	Used for Lane and Link initialization and management
K27.7	STP	Start TLP	Marks the start of a Transaction Layer Packet
K28.2	SDP	Start DLLP	Marks the start of a Data Link Layer Packet
K29.7	END	End	Marks the end of a Transaction Layer Packet or a Data Link Layer Packet
K30.7	EDB	EnD Bad	Marks the end of a nullified TLP
K23.7	PAD	Pad	Used in Framing and Link Width and Lane ordering negotiations
K28.0	SKP	Skip	Used for compensating for different bit rates for two communicating Ports
K28.1	FTS	Fast Training Sequence	Used within an Ordered Set to exit from L0s to L0
K28.3	IDL	Idle	Used in the Electrical Idle Ordered Set (EIOS)
K28.4			Reserved
K28.6			Reserved

Physical Layer - Logical

TLPs and DLLPs from the Data Link Layer are clocked into a buffer in the Physical Layer, where Start and End characters are added to facilitate detection of the packet boundaries at the receiver. Since the Start and End characters appear on both ends of a packet they are also called "framing" characters. The framing characters are shown appended to a TLP and DLLP in Figure 2-28 on page 77, which also shows the size of each field.

Figure 2-28: TLP and DLLP Structure at the Physical Layer



Within this layer, each byte of a packet is split out across all of the lanes in use

TLP DLL CRC 32bit DLLP CRC 16bit

DLLP DLLP Type 16 CRC

	Type	
Ack	00000000b	TLP
Nak	00010000	TLP
<InitFC1/InitFC2/UpdateFC>-<P/NP/Cpl>	Type	P/NP/Cpl
MRInitFC1/MRInitFC2/MRUpdateFC	0111/1111/1011>0xxxb	P/NP/Cpl
PM_*	00100xxxb	
NOP	00110001b	
Data_Link_Feature	00000010b	Scaled Flow Control
Vendor-specific	00110000b	DLLP

Ack/Nak TLP Ack Nak DLLP Ack Nak

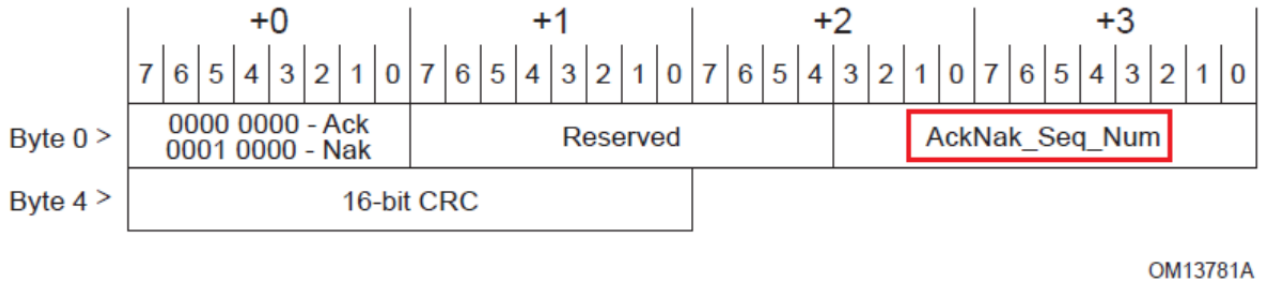


Figure 3-5 Data Link Layer Packet Format for Ack and Nak

AckNak_Seq_Num TCP PCIe Ack Nak Ack Nak Retry Buffer Ack/Nak Buffer

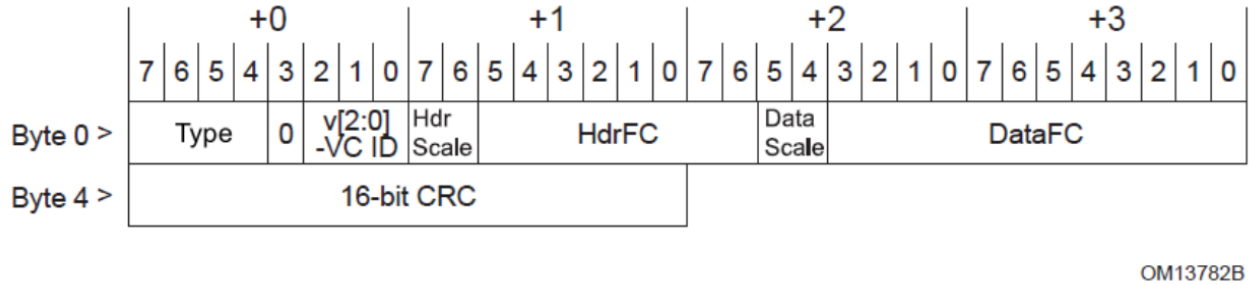
DDLP 4 retrain

VC Virtual Channel TLP PCIe TC Traffic Class VC Virtual Channel VC

1. TLP Posted P Non-Posted NP Completion Cpl
2. VC Link Link VC VC

N/NP/Cpl

- **InitFC1-P/NP/Cpl**
- **InitFC2-P/NP/Cpl** InitFC1 InitFC1
- **UpdateFC-P/NP/Cpl**



- **Type** ID

Type	Id
InitFC1-P	0100b
InitFC1-NP	0101b
InitFC1-Cpl	0110b
InitFC2-P	1100b
InitFC2-NP	1101b
InitFC2-Cpl	1110b
UpdateFC-P	1000b
UpdateFC-NP	1001b
UpdateFC-Cpl	1010b

- **VC ID v[2:0]** Virtual Channel Id Id 3 8 VC
- **HdrFC** TLP Credit TLP Header Credit TLP
- **DataFC** TLP Credit DW Double Word 4 Data Credit

64 128 4 DW TLP 128 Payload 1 DW TLP Digest 1 Header Credit (128 + 4) / 4 = 33 Data Credit

UpdateFC 30us CRC

Scaled Flow Control 2

Table 3-2 Scaled Flow Control Scaling Factors

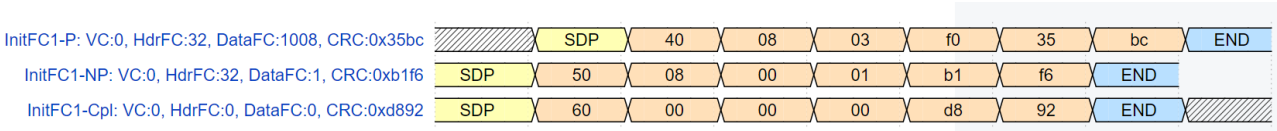
Scale Factor	Scaled Flow Control Supported	Credit Type	Min Credits	Max Credits	Field Width	FC DLLP field	
						Transmitted	Received
00b	No	Hdr	1	127	8 bits	HdrFC	HdrFC
		Data	1	2,047	12 bits	DataFC	DataFC
01b	Yes	Hdr	1	127	8 bits	HdrFC	HdrFC
		Data	1	2,047	12 bits	DataFC	DataFC
10b	Yes	Hdr	4	508	10 bits	HdrFC >> 2	HdrFC << 2
		Data	4	8,188	14 bits	DataFC >> 2	DataFC << 2
11b	Yes	Hdr	16	2,032	12 bits	HdrFC >> 4	HdrFC << 4
		Data	16	32,752	16 bits	DataFC >> 4	DataFC << 4

AVIP log credit 6 stack post non-post cpl head and data

```
|PH     |PD     |NPH    |NPD    |CPL    |CPLD   |
VC:0    |0x390  |0x5dd  |0x390  |0xe4   |0      |0      |
VC:0x1  |0      |0      |0      |0      |0      |0      |
TC->VC map
TC:  0 1 2 3 4 5 6 7
VC:  0 0 0 0 0 0 0 0
```

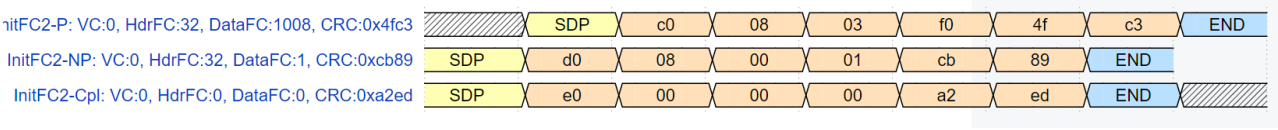
HdrFC DataFC

1. PCIe Endpoint Switch



2. Switch Endpoint Switch Endpoint

3. Switch InitFC1 DLLP InitFC2 DLLP



Blog

5.2.3 Transaction Layer

[toc]

VC Ordering

Introduction

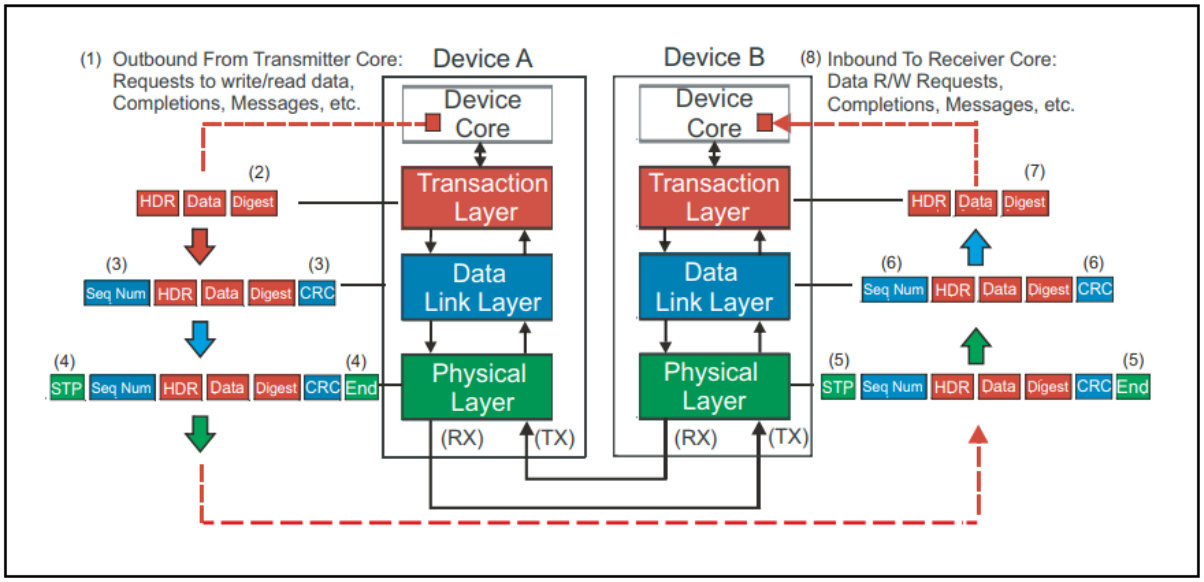
PCIe Transaction

- Memory Transaction
- IO IO Transaction
- Configuration Transaction
- Message Transaction

- **Non-Posted** Completion
- **Posted** Fire and forget

Non-Posted NP Posted P Completion Cpl

Figure 5-2: PCIe TLP Assembly/Disassembly



TLP Transaction Layer Packet

PCIe TLP Transaction Layer Packet

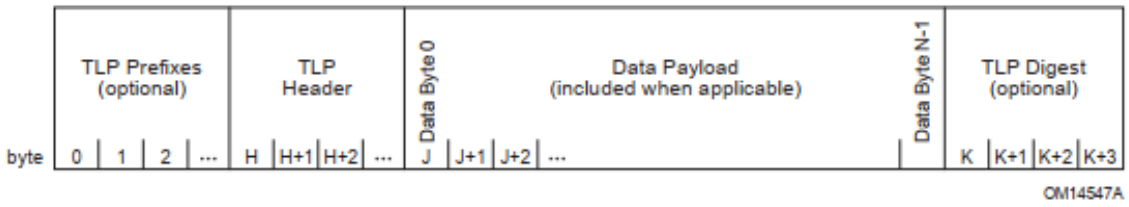


Figure 2-2 Serial View of a TLP

- **TLP Prefix** Precision Time Measurement
  SMMU substreamID PASID
- **TLP Digest** 4 CRC
  DLLP CRC CRC switch DLLP CRC TLP switch TLP TLP
- **TLP Header** TLP
- **TLP Payload** TLP Payload Payload `Max_Payload_Size` 4096
  buffer buffer `Max_Payload_Size`

TLP TLP 12 3DW 16 4DW 4 DW 8 DW

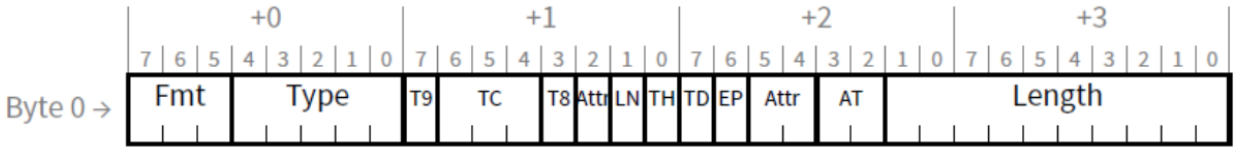


Figure 2-5 Fields Present in All TLP Headers

- **Fmt**

  : TLP

  - **Bit 7** 1 Fmt 100 TLP Prefix
  - **Bit 6** 1 = TLP Payload 0 = TLP Payload
  - **Bit 5** 1 = 32 12 3DW Header 0 = 64 16 4DW Header
- **Type** IO

  TLP
- **LN Lightweight Notification**
- **TH TLP Hints** TPH TLP Processing Hint TPH TLP Prefix
- **TD TLP Digest** 1 = TLP Digest 0 = TLP Digest
- **EP Error Poisoning** 1 = 0 =
- **AT Address Type** ATS 00 = 01 = 10 = 11 =
- **Length** Payload **DW Double Word** 1DW = 4

Transaction Descriptor TLP ID Requester ID Tag Attr TC TL

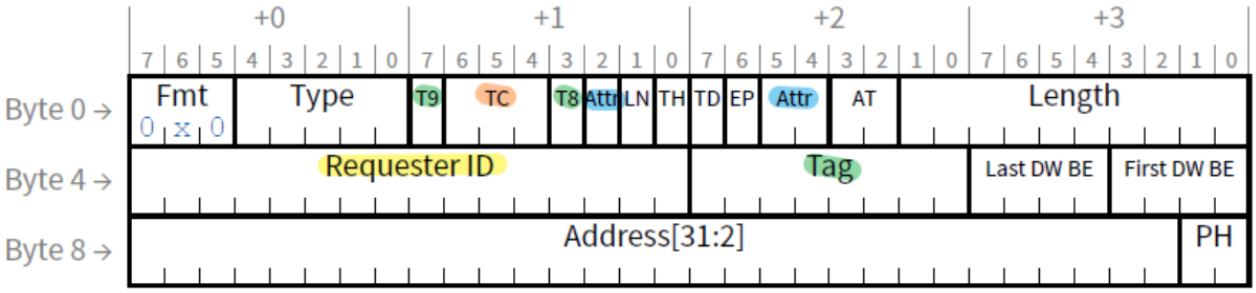


Figure 2-18 Request Header Format for 32-bit Addressing of Memory

ID Transaction ID

ID Requester ID Tag

- **Requester ID** 16 bit BDF 8:5:3.
- **Tag** 10 bit TLP PCIe T8 T9 bits tag bits `10-Bit Tag Requester Enable`

Attributes

bits Attr[2:1] Byte 1 - Bit 2 Byte 2 - Bit 5 Attr[0] Byte 2 - Bit 4 Coherency

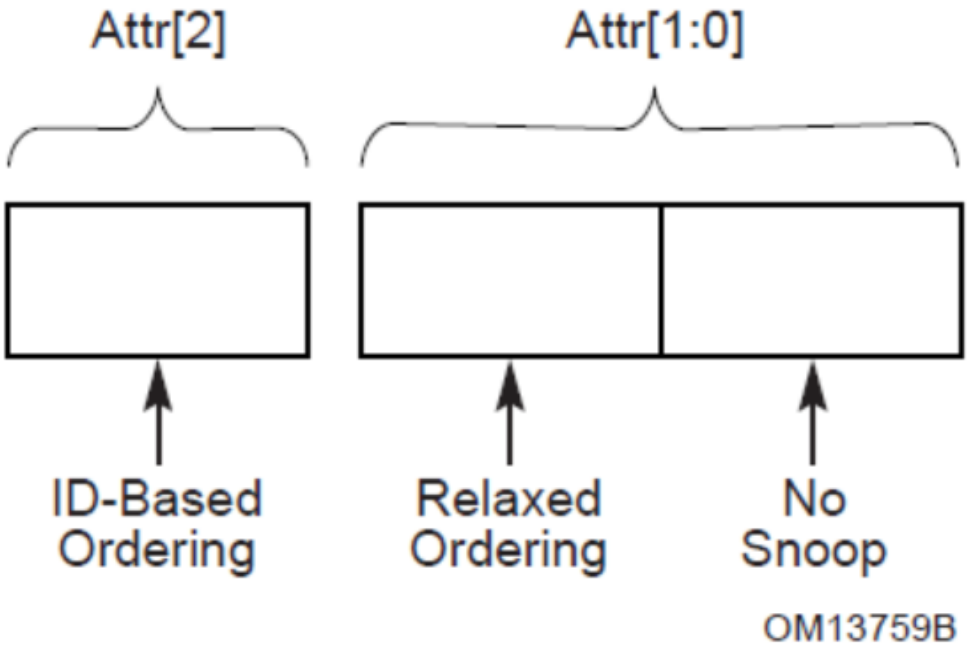


Figure 2-16 Attributes Field of Transaction Descriptor

Ordering

PCIe virtual channel TLP TC buffer VC VC TC TLP ordering

Figure 2-47 provides a graphical illustration of TC to VC mapping in several different Link configurations. For additional considerations on TC/VC, refer to Section 6.3 .

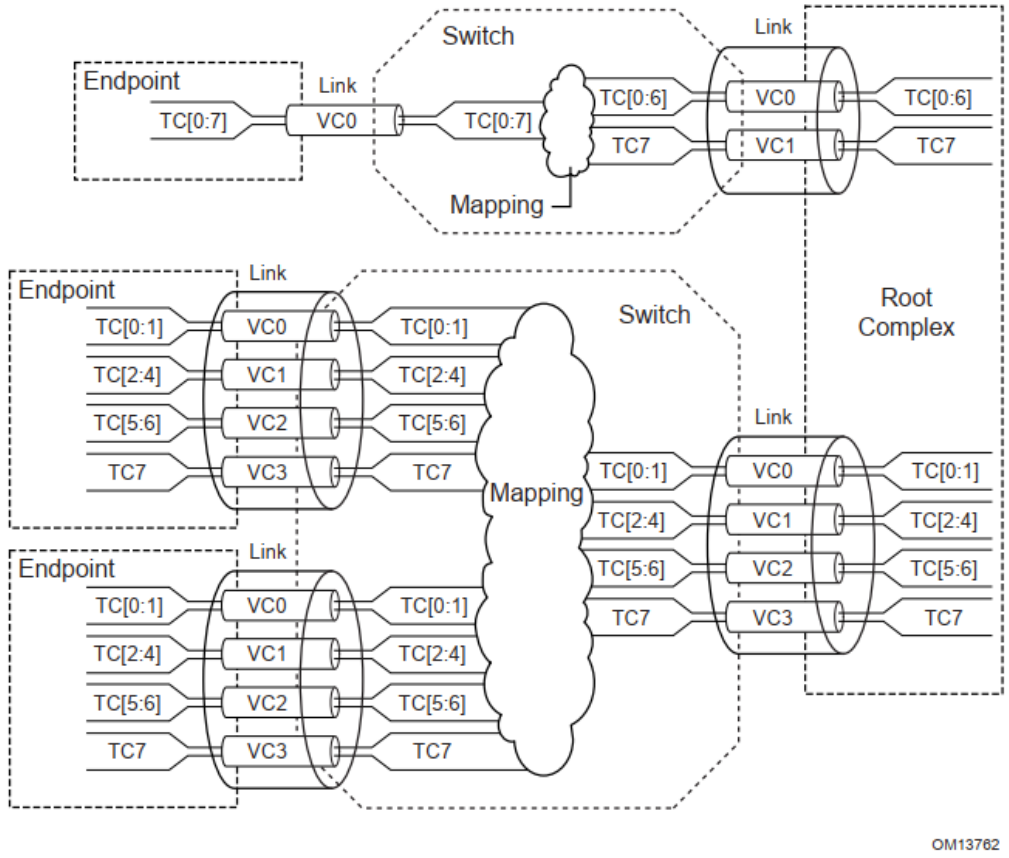


Figure 2-47 An Example of TC/VC Configurations

2.5.3 VC and TC Rules

Attr[2:1] Bits

Attr[2]	Attr[1]	
0	0	
0	1	Relaxed Ordering
1	0	ID-based Ordering
1	1	Relaxed Ordering ID-based Ordering

No Snoop

NoSnoop Attr[0] 0 PCIe Cache 1 PCIe Cache Cache

flag flag

IO MSI DMA

Traffic Class

Traffic Class 3 bit 8

TC VC Virtual Channel

1. PCIe Link VC Virtual Channel VC

2. TC VC TLP TC TLP VC
3. VC VC Credit Pool VC Credit 0 TLP Credit VC Credit
4. TC 0 Hardcoded VC0 TC TLP VC0
5. VC TC

TLP ID

PCIe

Address-Based Routing Memory Transaction IO IO Transaction

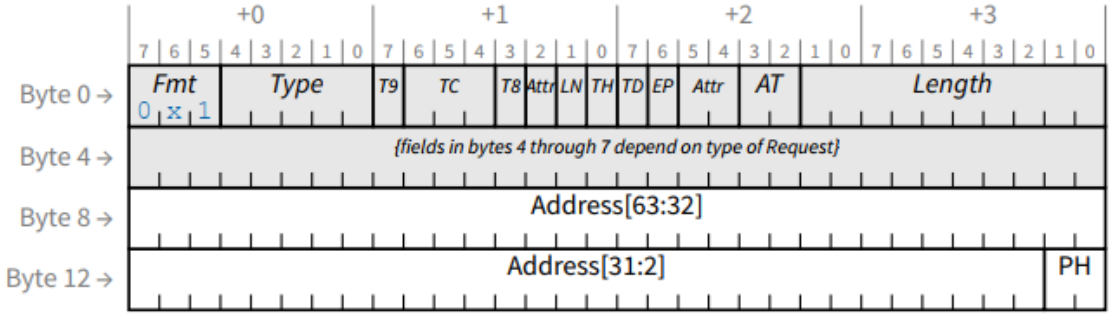


Figure 2-7 64-bit Address Routing

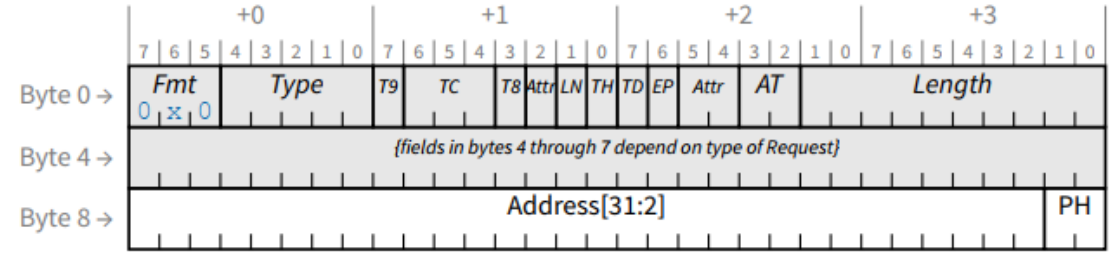


Figure 2-8 32-bit Address Routing

BDF **ID** **ID Based Routing** Configuration Transaction Message Transaction Completion

Implicit routing is used only with **Message Requests**, and is covered in Section 2.2.8

5.2.4 PCIe

[toc]

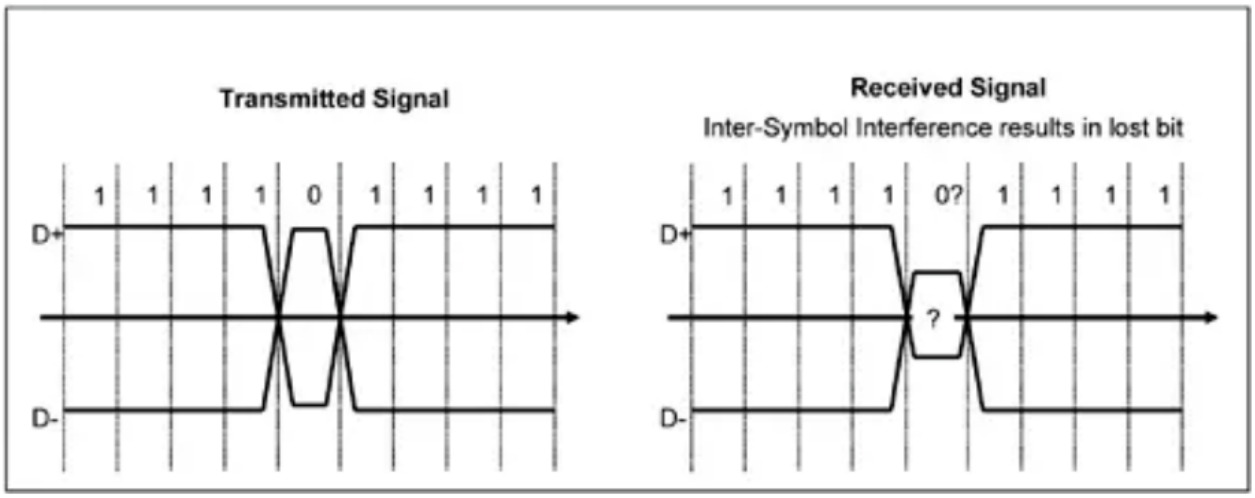
Overview

( )

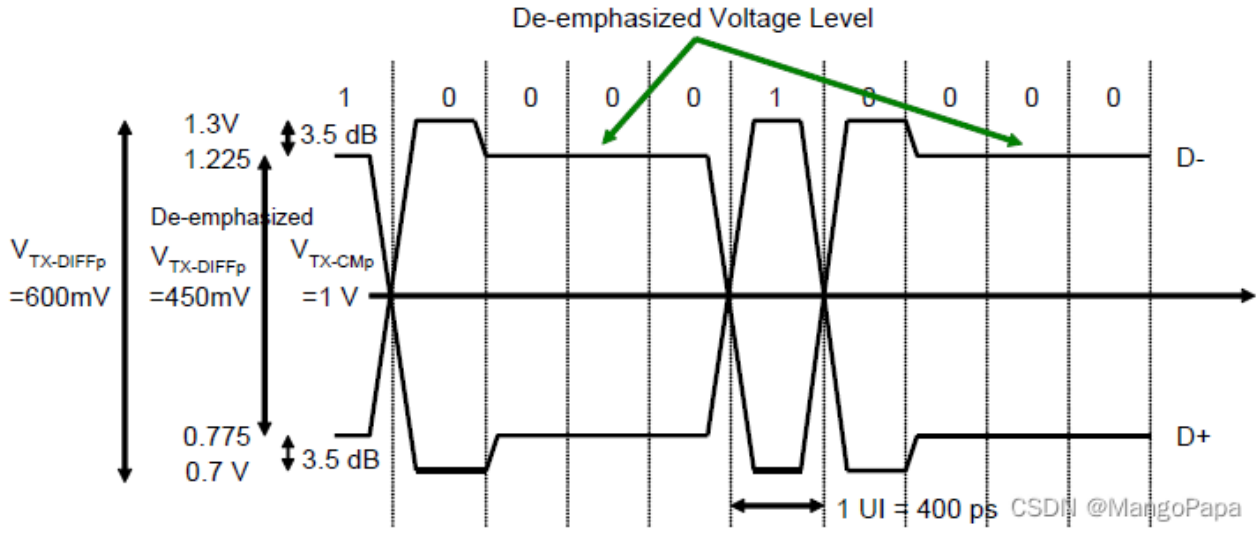
De-emphasis Pre-shoot

Feed forward Equalizer FFE

111101111 **0** **0** **1** **0**



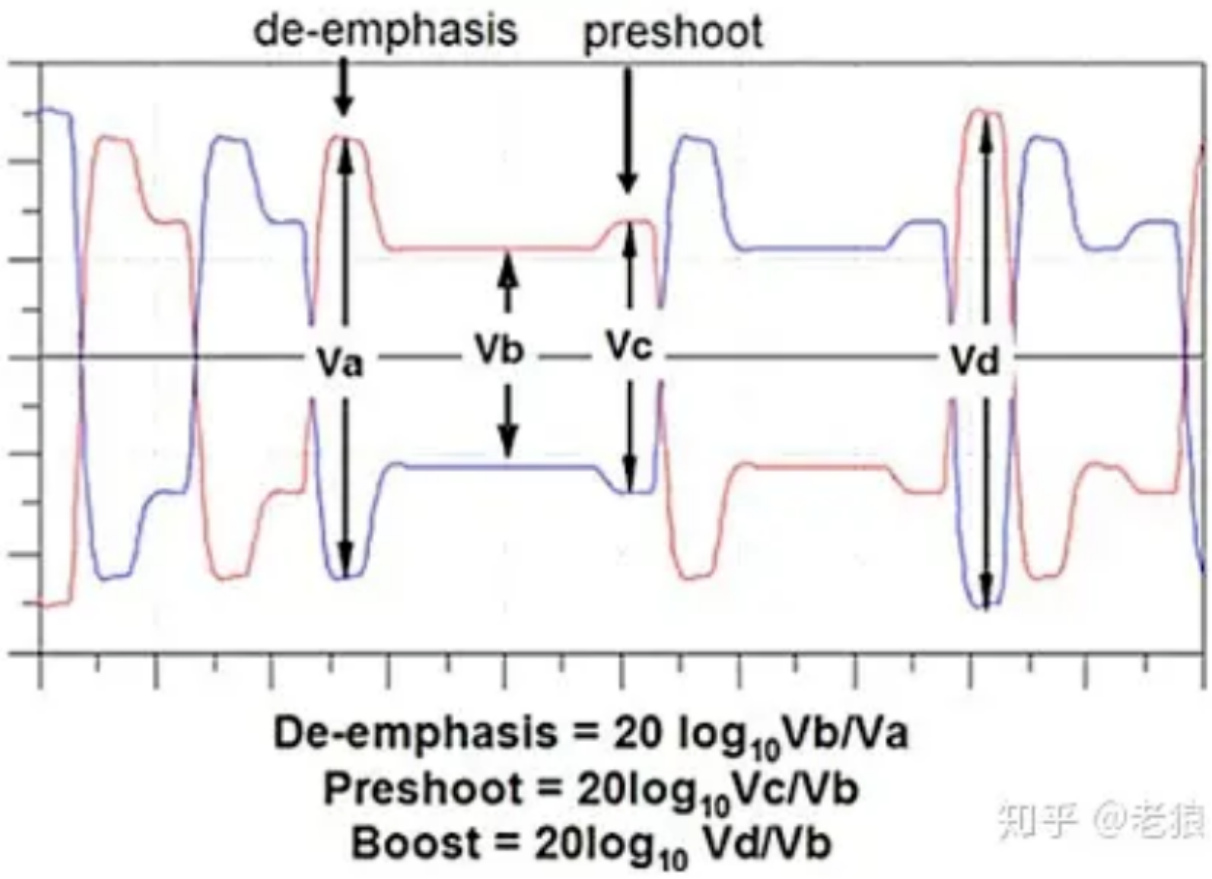
1 **0**



▲ 图 4: 去加重规则示意图

1.
2.

Preshoot De-emphasis



FIR 8 GT/s Finite Impulse Response, FIR

FIR 5

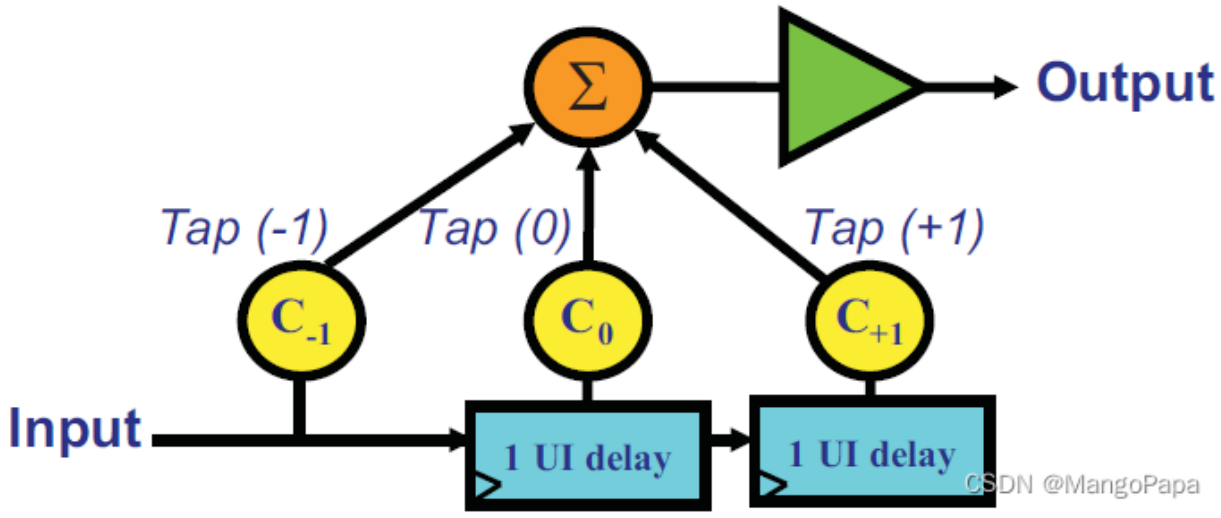
$$V_{out}(n) = C_{-1}V_{in}(n-1) + C_0 V_{in}(n) + C_{+1}V_{in}(n+1)$$

C-1 C0 C+1 FIR Pre-cursor Cusor Post-cursor Pre-cursor Pre-shoot Post-cursor

$$|C_{-1}| + |C_0| + |C_{+1}| = 1, \quad C_{-1}$$

Pre-cursor Cusor Post-cursor

Pre-cursor Post-cursor Cursor



de-emphasize preshoot

preset

▼ 表 1: Tx Preset Encoding, Ratios and Corresponding Coefficient Values

Encoding	Preset #	Pre-shoot (dB)	De-emphasis (dB)	C_{-1}	C_{+1}	Va/Vd	Vb/Vd	Vc/Vd
0000b	P0	0.0	-6.0 ± 1.5 dB	0.000	-0.250	1.000	0.500	0.500
0001b	P1	0.0	-3.5 ± 1 dB	0.000	-0.167	1.000	0.668	0.668
0010b	P2	0.0	-4.4 ± 1.5 dB	0.000	-0.200	1.000	0.600	0.600
0011b	P3	0.0	-2.5±1dB	0.000	-0.125	1.000	0.750	0.750
0100b	P4	0.0	0.0	0.000	0.000	1.000	1.000	1.000
0101b	P5	1.9 ± 1 dB	0.0	-0.100	0.000	0.800	0.800	1.000
0110b	P6	2.5 ± 1 dB	0.0	-0.125	0.000	0.750	0.750	1.000
0111b	P7	3.5 ± 1 dB	-6.0 ± 1.5 dB	-0.100	-0.200	0.800	0.400	0.600
1000b	P8	3.5 ± 1 dB	-3.5 ± 1 dB	-0.125	-0.125	0.750	0.500	0.750
1001b	P9	3.5 ± 1 dB	0.0	-0.166	0.000	0.668	0.668	1.000
1010b	P10	0.0	不定	0.000	不定	1.000	不定	不定
1011b~1111b	Reserved							

Continuous-Time Linear Equalizaion, CTLE Decision Feedback Equalization DFE

CTLE

DFE DFE

DFE

DFE

2.5 GT/s -3.5dB

5 GT/s -3.5dB -6dB

8 GT/s 16 GT/s 32 GT/s gen1, gen2

gen3 FIR CTLE DFE

PCIe

PCIe

8 GT/s 16 GT/s 32 GT/s

L0 PCIe EQ EQ DLLP

- Link Control 3 Register Perform Equalization
- Link Control 2 Register Target Link Speed 8 GT/s
- Link Control Register Retrain Link

DLLP Blocking

Full equalization mode PCIe

training

PCIe EQ 100 ms EQ PCIe Gen5 PCIe

LTSSM Configuration TS Equalization bypass to highest rate

32GT Capability Reg Equalization bypass to highest rate

Bypass equalization to highest rate 2.5 GT/s L0 Recovery 32 GT/s 1 EQ

32 GT/s No equalization needed 5 GT/s EQ EQ EQ Gen1 gen5

Phase PCIe Controller link status Equalization Phase Successful

Phase 3 Equalization Complete

8GT link status

16 32GT 16 32GT status

EQ

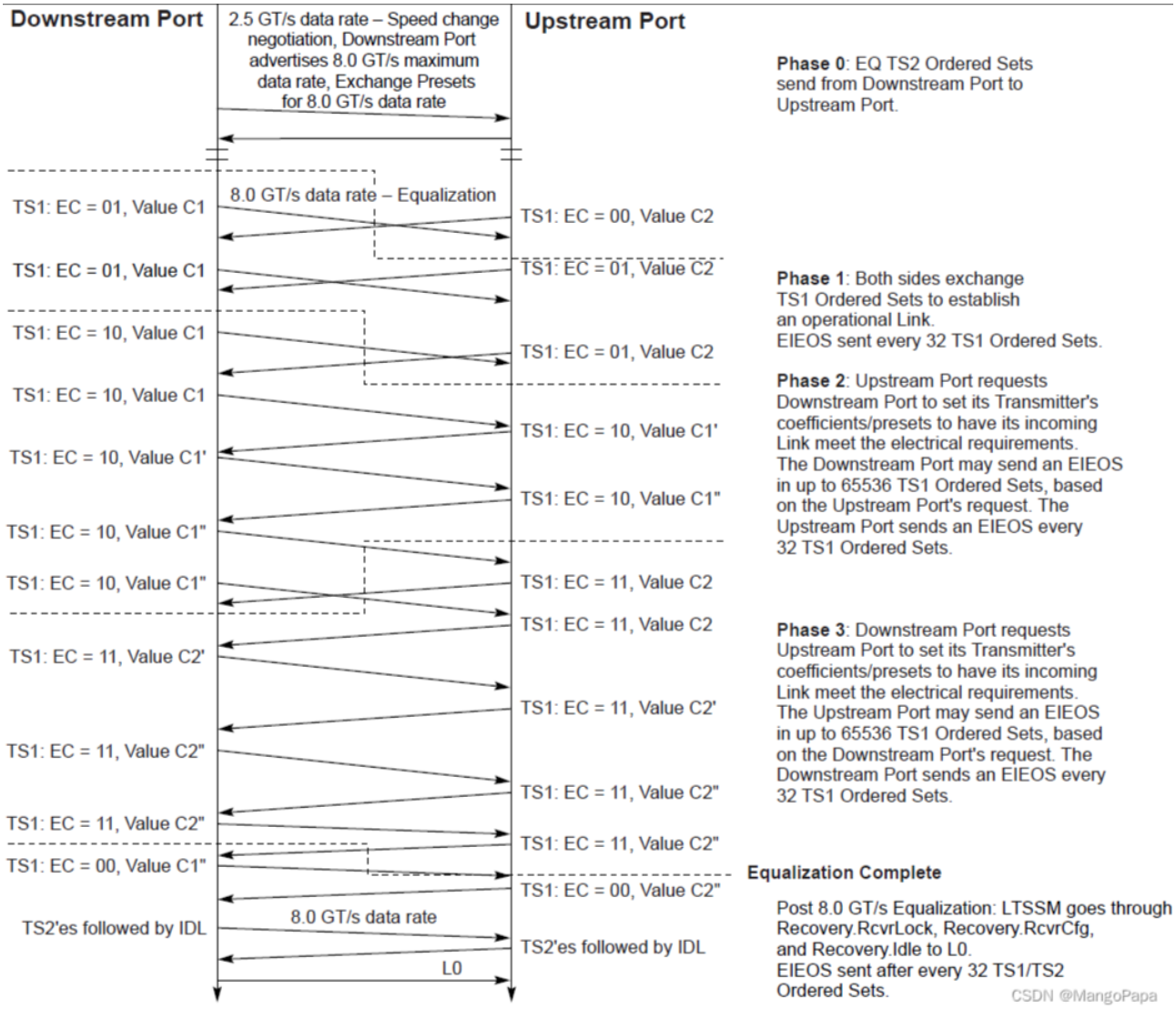
1. EQ
2. EQ
3.

Port EQ Port EQ

8 GT/s Link Status 2 Regiser Link Equanlization Requset 8 GT/s 16 GT/s 16 GT/s Status Register Link Equanlization Requset 16 GT/s 32 GT/s 32 GT/s Status Register Link Equanlization Requset 32 GT/s

EQ

1. gen1 DSP gen3 preset
2. EQ Phase 0 EC=0 8 GT/s USP Preset Coefficients DSP **DSP Phase 0**
3. EQ Phase 1 EC=1 DSP USP LF (Low Frequency symbol7) FS (Full Swing symbol8) Post-cursor (symbol9) Phase 2/3 32 TS1 EIEOS
4. EQ Phase 2 EC=2 USP Master DSP Tx Preset USP USP Preset coefficient USP 32 TS1 EIEOS DSP USP 65536 TS1 EIEOS
5. EQ Phase 3 EC=3 DSP Master USP Tx Preset DSP DSP Preset coefficient DSP 32 TS1 EIEOS USP DSP 65536 TS1 EIEOS
6. EQ EC=0 LTSSM Recovery.RcvrLock -> Recovery.RcvrCfg -> Recovery.Idle -> L0 L0 32 TS1/TS2 EIEOS



PCIe EQ DSP phase1

phase0

https://mangopapa.blog.csdn.net/article/details/124539607

5.3 ARM N2

ARM N2 ARM Neoverse N2 reference design ARM Neoverse N2 ARM CPU

1. N2 HPC
2. Neoverse N2
3. ARM N2
4. N2
5. Neoverse N2 ARM ARM

ARM N2

ARM Neoverse N2 ARM IP IP

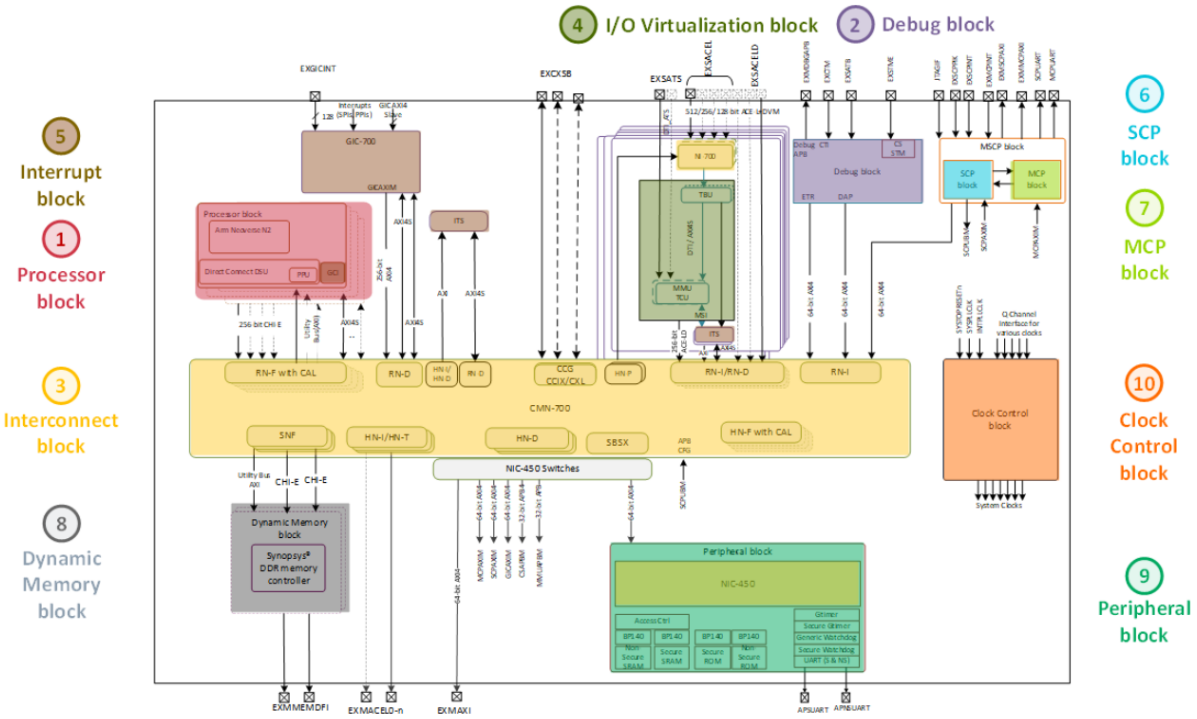
1. **ARM Neoverse N2 CPU** N2 N2 ARMv9
2. **System Level Cache (SLC)** ARM N2 SLC
3. **CCIX CXL** N2 CCIX Cache Coherent Interconnect for Accelerators CXL Compute Express Link
4. **AMBA (Advanced Microcontroller Bus Architecture)** AMBA SoC IP N2 AMBA
5. **TrustZone** ARM TrustZone
6. **Mali GPU** N2 CPU ARM Mali GPU
7. **SMMU (System Memory Management Unit)**

IP ARM Neoverse N2

High Level

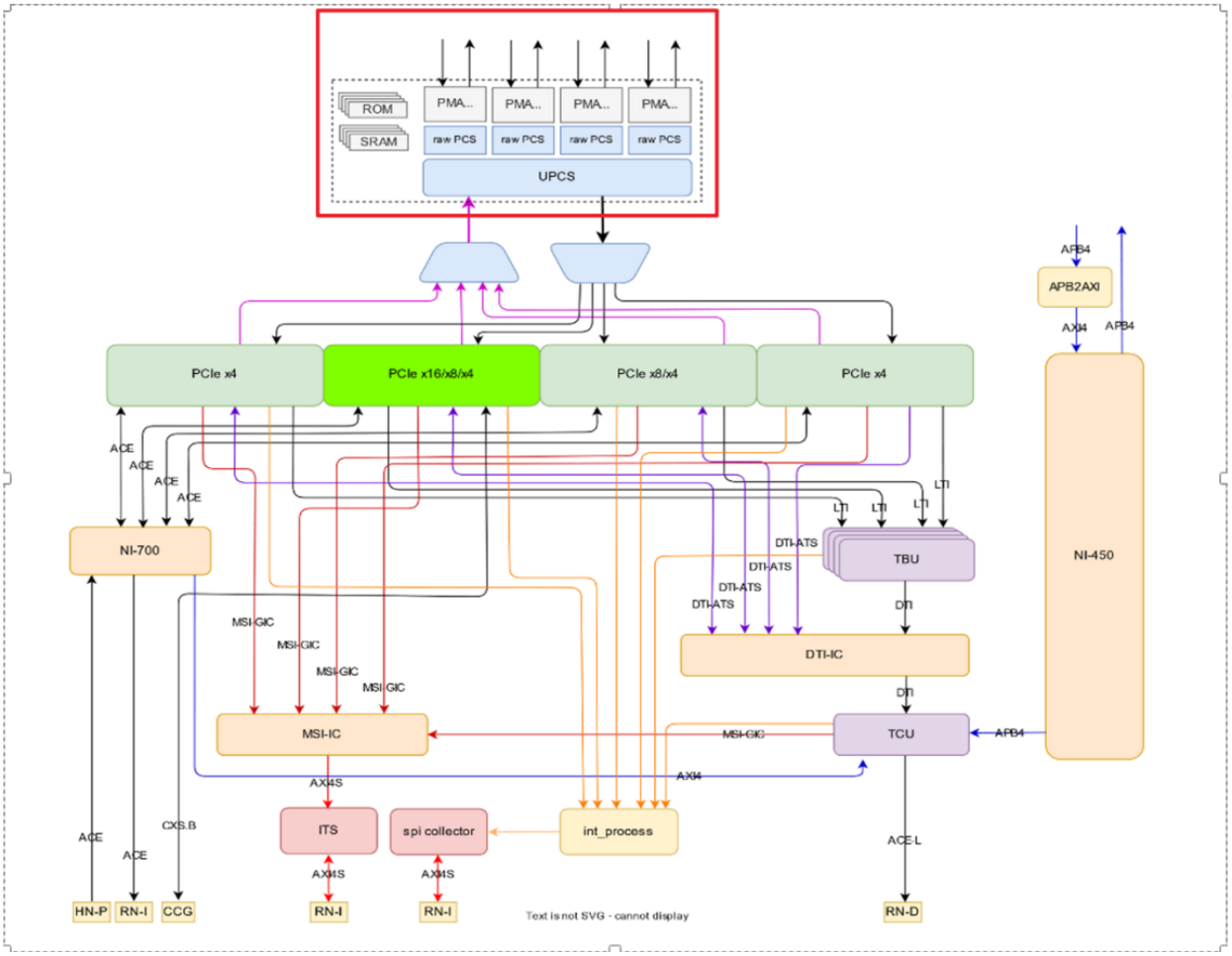
Arm Neoverse N2 reference design Technical Overview

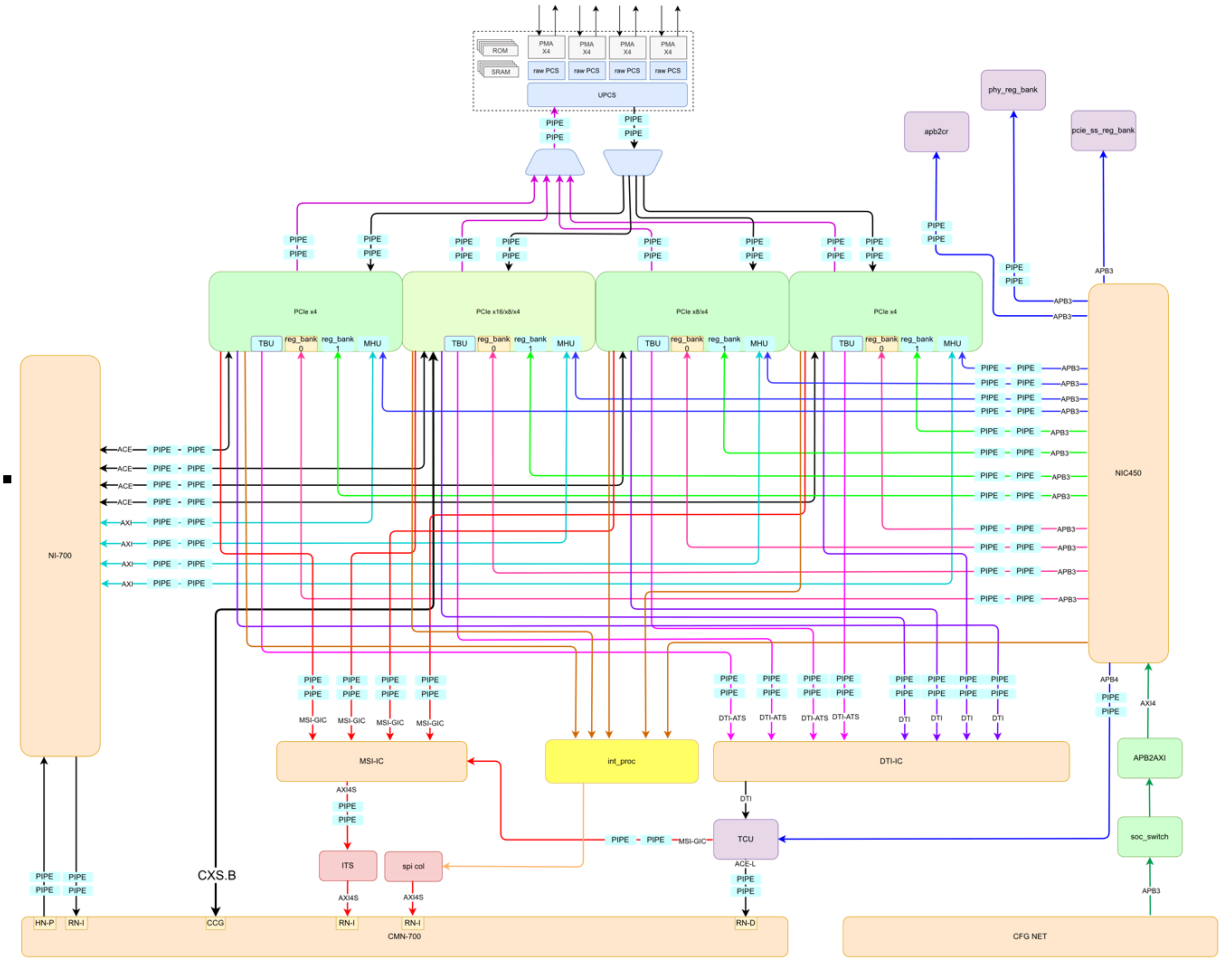
Figure 3-1: RD-N2 architecture block diagram



CMN-700 PICe SMMU NI-700 RNI SCP GIC SNF memory controller core

5.3.1 PCIe


ROM
SRAM
PMA...
PMA...
PMA...
PMA...
raw PCS
raw PCS
raw PCS
raw PCS
UPCS
APB4
APB2AXI
AXI4
APB4
PCIe x4
PCIe x16/x8/x4
PCIe x8/x4
PCIe x4
ACE
ACE
ACE
ACE
LTI
LTI
LTI
LTI
NI-700
DTI-ATS
DTI-ATS
DTI-ATS
DTI-ATS
TBU
NI-450
DTI
DTI-IC
DTI
MSI-GIC
MSI-GIC
MSI-GIC
MSI-GIC
MSI-IC
MSI-GIC
TCU
APB4
AXI4
AXI4S
ACE
ACE
CXS.B
ITS
spi collector
int_process
ACE L
AXI4S
AXI4S
HN-P
RN-I
CCG
RN-I
RN-I
RN-D
Text is not SVG - cannot display



4 S CXL CXL PCIe x16 x4 4 controller bifircation

CMN RNI HNP CCG NI-700 NI-700 controller X16 CCG CXL CCIX

SMMU-700 MMU TBU TCU DTIIC

MSI-IC MSI switch controller MSI MSI

ITS PCIe MSI GIC ITS GIC

int process

NIC-450 NI-700 NIC-450 core cfg controller NI-700 PCIe AMBA ACE

5.4 PCIe

5.4.1 Resource Degration

[toc]

BAR 5 resource pool

图1 是Memory BAR寄存器的结构，图2是IO类型的BAR寄存器

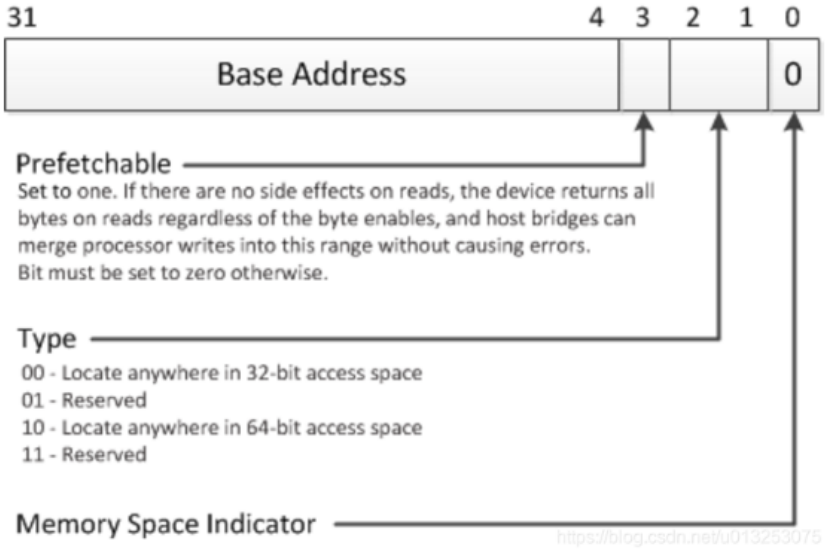


图1 Base Address Register for Memory

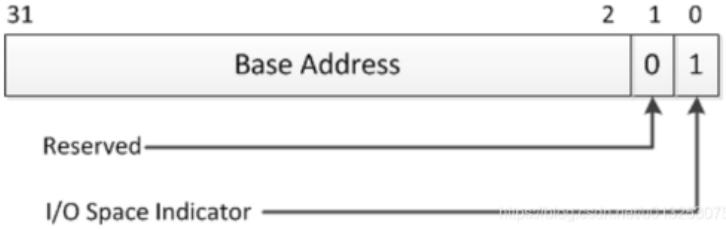


图2 Base Address Register for I/O

bit0:表示设备寄存器是映射到memory（0）还是IO（1)空间。
bit1: reserved 0
bit2: 在base adress register for Memory 中0表示32位地址空间，1表示64位地址空间。
bit3:在memory BAR中用来表示该设备是否允许prefetch，1表示可以预取，0表示不可以预区。
bit4~31:用来表示设备需要占用的地址空间大小。

```
InitializeResourcePool (&IoPool, PciBarTypeIo16);
InitializeResourcePool (&Mem32Pool, PciBarTypeMem32);
InitializeResourcePool (&PMem32Pool, PciBarTypePMem32);
InitializeResourcePool (&Mem64Pool, PciBarTypeMem64);
InitializeResourcePool (&PMem64Pool, PciBarTypePMem64);
```

RC prefetchable.

```
//
// The base address of MMIO Registers
//
#define AC01_PCIE_MMIO_BASE_LIST         0x300000000000, 0x340000000000, 0x380000000000, 0x3C000

//
// The size of MMIO space
//
#define AC01_PCIE_MMIO_SIZE_LIST         0x3FFE0000000, 0x3FFE0000000, 0x3FFE0000000, 0x3FFE0000

//
// The base address of MMIO32 Registers
//
#define AC01_PCIE_MMIO32_BASE_LIST       0x000020000000, 0x000028000000, 0x000030000000, 0x00003

//
// The size of MMIO32 space
//
#define AC01_PCIE_MMIO32_SIZE_LIST       0x8000000, 0x8000000, 0x8000000, 0x8000000, 0x4000000,
```

BAR 11 Prefetchable vs. Non-Prefetchable Memory Space

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

PCIe Technology 3.0 Chapter 4.1.2.2

1. DegradeResource()

1.1 Degrade Resource For OptionRom

If any child device has both option ROM and 64-bit BAR, degrade its PMEM64/MEM64 requests in case that if a legacy option ROM image can not access 64-bit resources.

PCD `PcdPciDegradeResourceForOptionRom`

- X64 ---- TRUE
- X32 ARM AARCH64 ---- FALSE

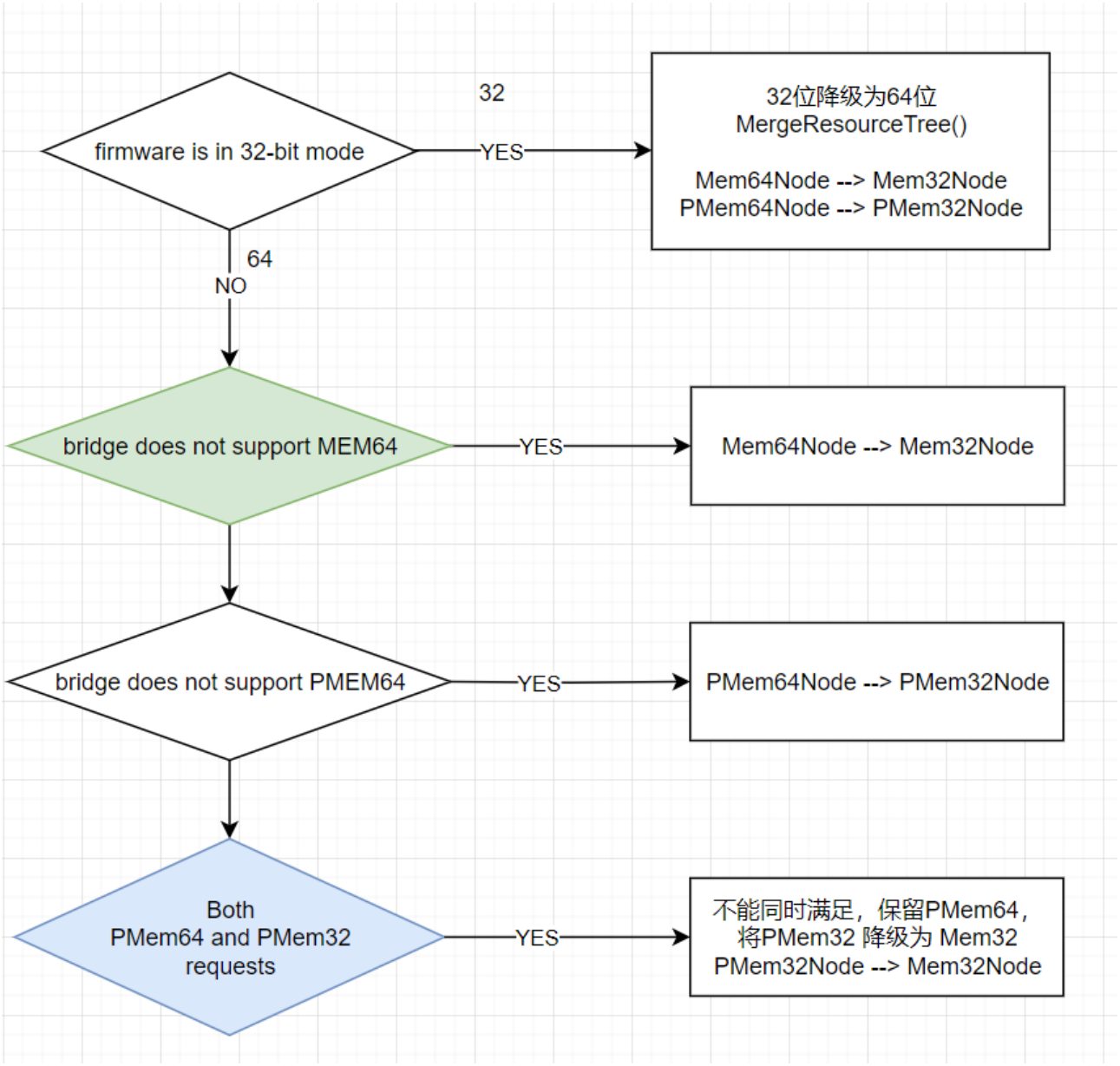
root bridge bar0 mem32 bar0 bar 64bit

1.2 Degrade Resource For Others

```
//
// Degrade resource if necessary
//
DegradeResource (Bridge, Mem32Node, PMem32Node, Mem64Node, PMem64Node);
```

`BridgeSupportResourceDecode()` bridge

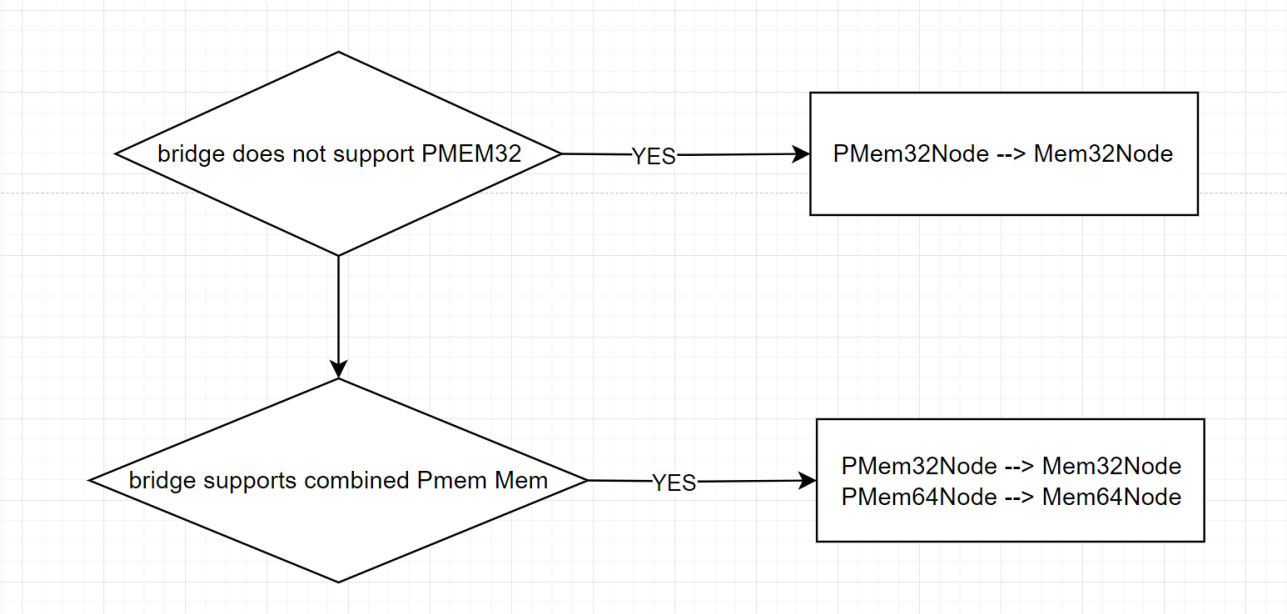
`MergeResourceTree()` resoure tree



PMem64 PMem32 bridge

NP-Mem64 32 32

Secondary Status	I/O Limit	I/O Base
Memory Limit	Memory Base	
Prefetchable Memory Limit	Prefecthable Memory Base	
Prefetchable Base Upper 32 Bits		
Prefetchable Base Limit 32 Bits		
I/O Limit Upper 16 Bits	I/O Base Upper 16 Bits	

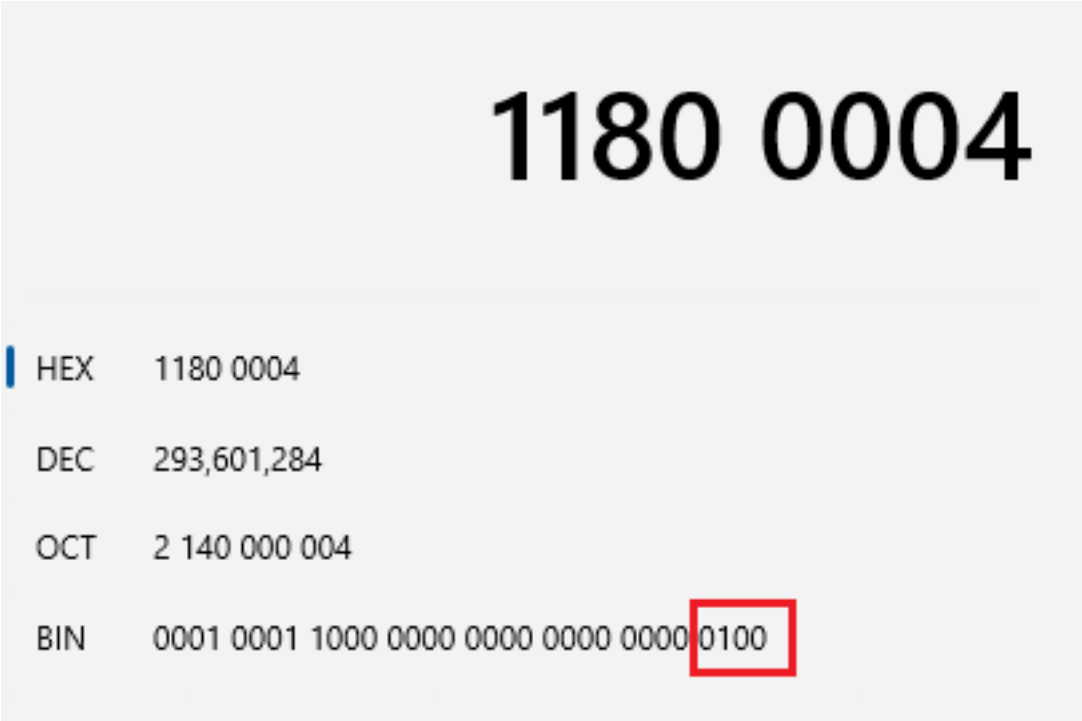


2.

```
0004:03:00.0 USB controller: Renesas Technology Corp. uPD720201 USB 3.0 Host Controller (rev 03) (prog-
    Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR+ FastB2B- DisINT
    Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx
    Latency: 0, Cache Line Size: 64 bytes
    Interrupt: pin A routed to IRQ 198
    NUMA node: 0
    IOMMU group: 0
    Region 0: Memory at 11800000 (64-bit, non-prefetchable) [size=8K]
    Capabilities: [50] Power Management version 3
        Flags: PMEClk- DSI- D1- D2- AuxCurrent=375mA PME(D0+,D1-,D2-,D3hot+,D3cold+)
        Status: D0 NoSoftRst+ PME-Enable- DSel=0 DScale=0 PME-
    Capabilities: [70] MSI: Enable- Count=1/8 Maskable- 64bit+
```

2.1 Ampere PCI log

```
0004:03:00.0 USB controller: Renesas Technology Corp. 
00: 12 19 14 00 06 05 10 00 03 30 03 0c 10 00 00 00
10: 04 00 80 11 00 00 00 00 00 00 00 00 00 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff
30: 00 00 00 00 50 00 00 00 00 00 00 00 ff 01 00 00
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



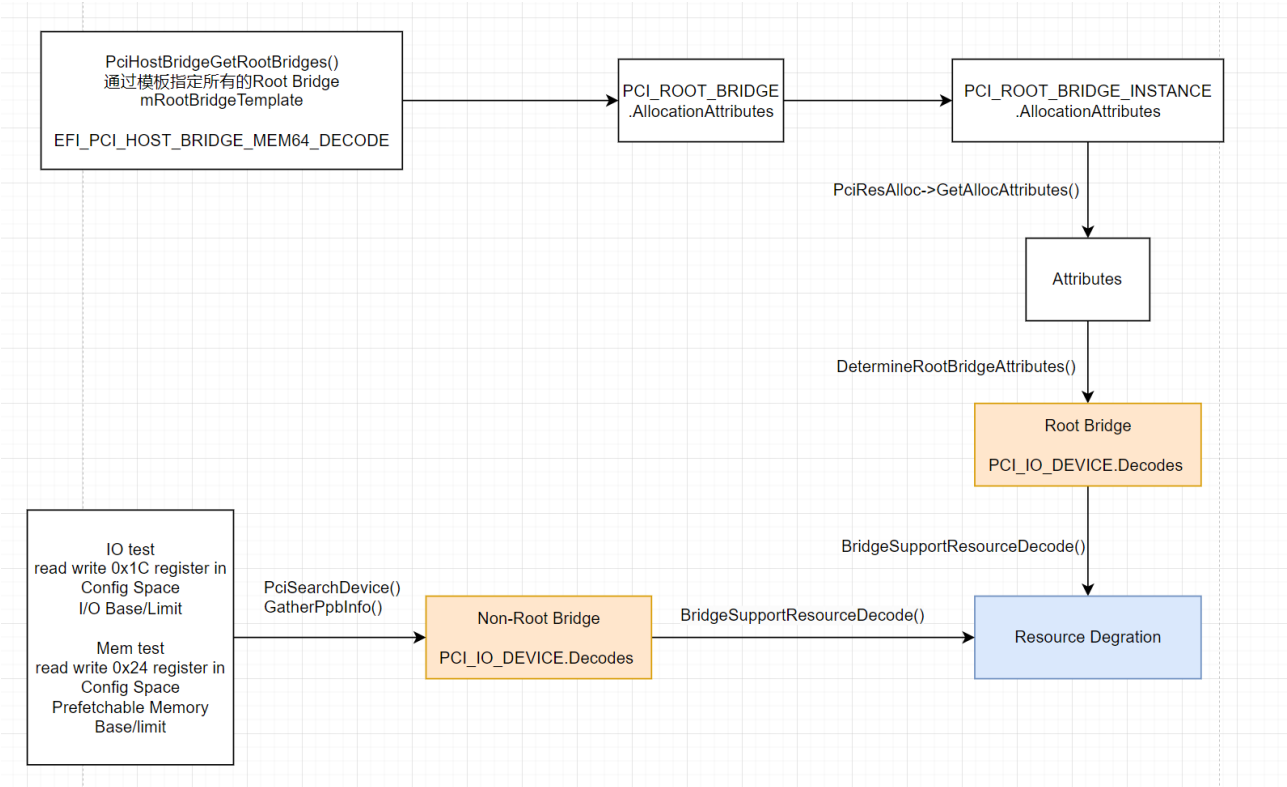
bit0 - 0 memory

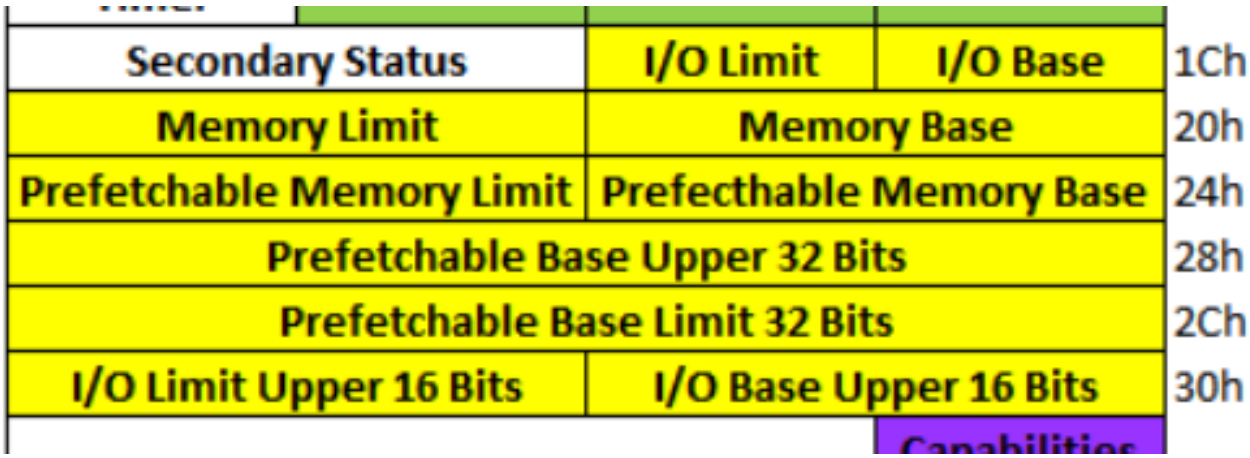
bit 2:1 memory 00 32bit memory 10- 64 bit memory B

bit3 - 0 Non-prefetchable

NP-Mem64

2.2 Degrade Resource `BridgeSupportResourceDecode()` PCI_IO_DEVICE.Decodes





bridge

```
#define EFI_BRIDGE_IO32_DECODE_SUPPORTED      0x0001
#define EFI_BRIDGE_PMEM32_DECODE_SUPPORTED    0x0002
#define EFI_BRIDGE_PMEM64_DECODE_SUPPORTED    0x0004
#define EFI_BRIDGE_IO16_DECODE_SUPPORTED      0x0008
#define EFI_BRIDGE_PMEM_MEM_COMBINE_SUPPORTED 0x0010
#define EFI_BRIDGE_MEM64_DECODE_SUPPORTED     0x0020
#define EFI_BRIDGE_MEM32_DECODE_SUPPORTED     0x0040

struct _PCI_IO_DEVICE {
   ...
  //
  // The resource decode the bridge supports
  //
  UINT32                                    Decodes;
  ...
}
```

PCI_IO_DEVICE.Decodes DetermineRootBridgeAttributes() protocol

```
PciResAlloc->GetAllocAttributes (
                          PciResAlloc,
                          RootBridgeHandle,
                          &Attributes
                          );
```

- root bridge RC root bridge
- bridge bridge

Table 7-10 I/O Addressing Capability

Bits 3:0	I/O Addressing Capability
0h	16-bit I/O addressing
1h	32-bit I/O addressing
2h-Fh	Reserved

- IO capability -- ACPI spec 7.5.1.3.6
- Mem capability -- PCIe spec 7.5.1.3.9

  The bottom 4 bits of both the Prefetchable Memory Base 0x24 and Prefetchable Memory Limit registers are read-only, contain the same value, and encode whether or not the bridge supports 64-bit addresses

  * 0 - the bridge supports only 32 bit addresses.
  * 1 - the bridge supports 64-bit addresses. bridge

```
0004:00:02.0 PCI bridge: Ampere Computing, LLC Altra PCI Express Root Port a1 (rev 04)
00: ef 1d 02 e1 07 01 10 00 04 00 04 06 10 00 01 00
10: 00 00 00 00 00 00 00 00 00 03 03 00 f0 00 00 00
20: 80 11 90 11 21 00 31 00 00 28 00 00 00 28 00 00
30: 00 00 00 00 40 00 00 00 00 00 00 00 ff 01 02 00
40: 01 70 c3 5f 08 00 00 00 00 00 00 00 00 00 00 00
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Prefetchable Memory Base 0x002**1**

Prefetchable Memory Limit 0x003**1**

0x28 32 PMEM32 PMEM64 flag PMEM32 PMEM32 MEM32 PMEM64 PMEM32 MEM32

PCIe Spec EDK2 Root Bridge bridge MEM64 Flag bridge NP-MEM64 NP-MEM64 tree merge NP-MEM32 tree

RC

```
InitializeResourcePool (&IoPool, PciBarTypeIo16);
InitializeResourcePool (&Mem32Pool, PciBarTypeMem32);
// InitializeResourcePool (&PMem32Pool, PciBarTypePMem32);
// InitializeResourcePool (&Mem64Pool, PciBarTypeMem64);
InitializeResourcePool (&PMem64Pool, PciBarTypePMem64);
```

2.3 RC RC Root Bridge

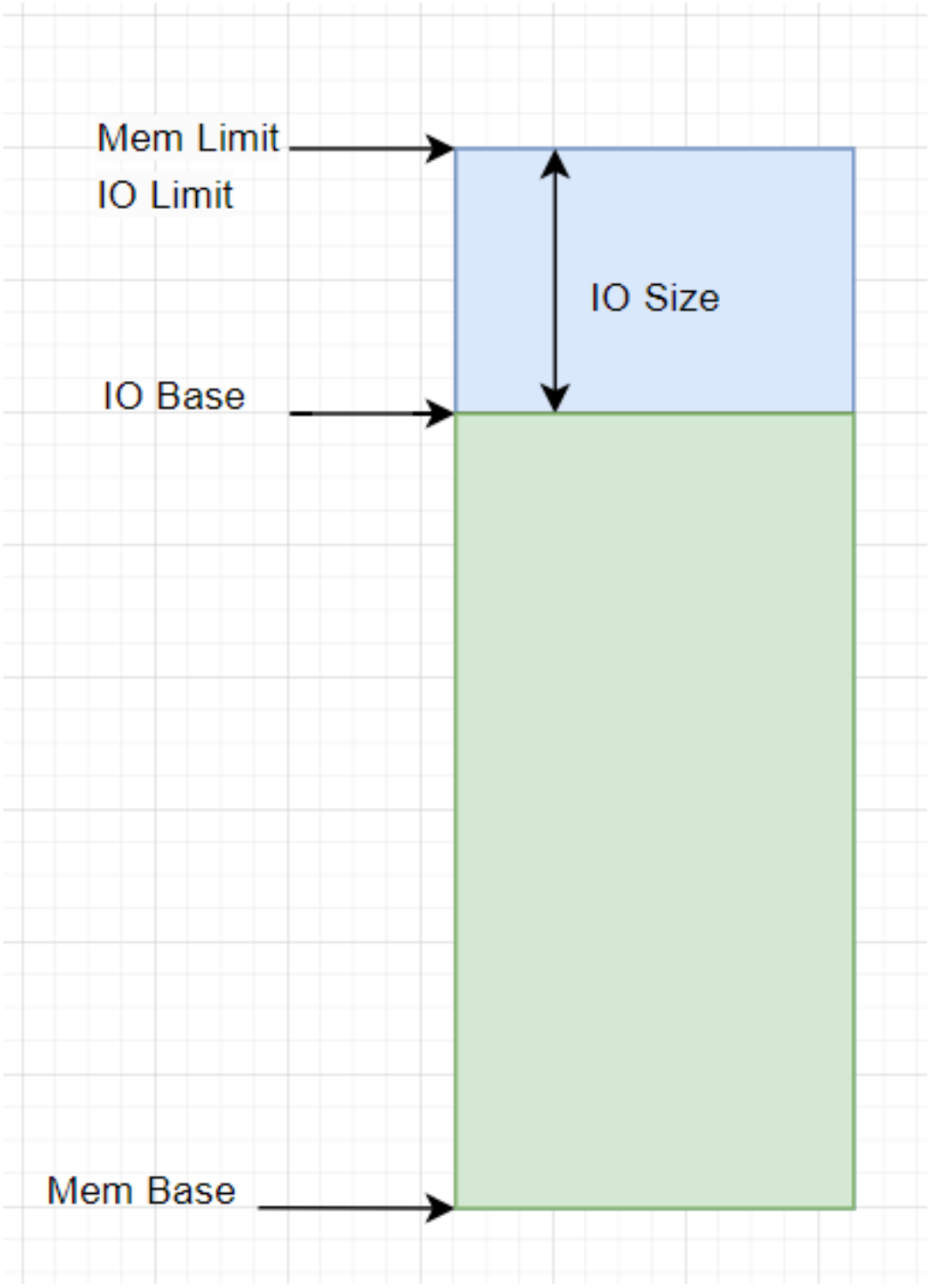
Mem PMem =**Mmio32Base**

Io.Base = Mem Limit IO_SIZE

PMemAbove4G = **MmioBase**

MemAbove4G

```
{  // PMem
 MAX_UINT64,
 0,
 0
}
```



```
CopyMem (&RootBridge->Bus, &Bridge->Bus, sizeof (PCI_ROOT_BRIDGE_APERTURE));
CopyMem (&RootBridge->Io, &Bridge->Io, sizeof (PCI_ROOT_BRIDGE_APERTURE));
CopyMem (&RootBridge->Mem, &Bridge->Mem, sizeof (PCI_ROOT_BRIDGE_APERTURE));
CopyMem (&RootBridge->MemAbove4G, &Bridge->MemAbove4G, sizeof (PCI_ROOT_BRIDGE_APERTURE));
CopyMem (&RootBridge->PMem, &Bridge->PMem, sizeof (PCI_ROOT_BRIDGE_APERTURE));
CopyMem (&RootBridge->PMemAbove4G, &Bridge->PMemAbove4G, sizeof (PCI_ROOT_BRIDGE_APERTURE));
```

```
  if (RootComplex->Mmio32Base != 0) {
      RootBridge->Mem.Base = RootComplex->Mmio32Base;
      RootBridge->Mem.Limit = RootComplex->Mmio32Base + RootComplex->Mmio32Size - 1;
      RootBridge->PMem.Base = RootBridge->Mem.Base;
      RootBridge->PMem.Limit = RootBridge->Mem.Limit;
      RootBridge->Io.Base = RootComplex->Mmio32Base + RootComplex->Mmio32Size -
↪   AC01_PCIE_IO_SIZE;
      RootBridge->Io.Limit = RootBridge->Mem.Limit;
    }

    if (RootComplex->MmioBase != 0) {
      RootBridge->PMemAbove4G.Base = RootComplex->MmioBase;
      RootBridge->PMemAbove4G.Limit = RootComplex->MmioBase + RootComplex->MmioSize - 1;
    }
if ((Attributes & EFI_PCI_HOST_BRIDGE_COMBINE_MEM_PMEM) != 0) {
    RootBridgeDev->Decodes |= EFI_BRIDGE_PMEM_MEM_COMBINE_SUPPORTED;
  }

  if ((Attributes & EFI_PCI_HOST_BRIDGE_MEM64_DECODE) != 0) {
    RootBridgeDev->Decodes |= EFI_BRIDGE_MEM64_DECODE_SUPPORTED;
    RootBridgeDev->Decodes |= EFI_BRIDGE_PMEM64_DECODE_SUPPORTED;
  }

  RootBridgeDev->Decodes |= EFI_BRIDGE_MEM32_DECODE_SUPPORTED;
  RootBridgeDev->Decodes |= EFI_BRIDGE_PMEM32_DECODE_SUPPORTED;
  RootBridgeDev->Decodes |= EFI_BRIDGE_IO16_DECODE_SUPPORTED;
```

NP-MEM64 NP-MEM32 bar NP-MEM32

Linux kernel 5.18.0 EDK2

PCIe Spec NP-MEM64

1.3.2.2 PCI Express Endpoint Rules

- A PCI Express Endpoint must be a Function with a Type 00h Configuration Space header.
- A PCI Express Endpoint must support Configuration Requests as a Completer.
- A PCI Express Endpoint must not depend on operating system allocation of I/O resources claimed through BAR(s).
- A PCI Express Endpoint must not generate I/O Requests.
- A PCI Express Endpoint must not support Locked Requests as a Completer or generate them as a Requester. PCI Express-compliant software drivers and applications must be written to prevent the use of lock semantics when accessing a PCI Express Endpoint.
- A PCI Express Endpoint operating as the Requester of a Memory Transaction is required to be capable of generating addresses greater than 4 GB.
- A PCI Express Endpoint is required to support MSI or MSI-X or both if an interrupt resource is requested., If MSI is implemented, a PCI Express Endpoint must support the 64-bit Message Address version of the MSI Capability structure.
- A PCI Express Endpoint requesting memory resources through a BAR must set the BAR's Prefetchable bit unless the range contains locations with read side-effects or locations in which the Function does not tolerate write merging. See Section 7.5.1.2.1 for additional guidance on having the Prefetchable bit Set.
- For a PCI Express Endpoint, 64-bit addressing must be supported for all BARs that have the Prefetchable bit Set. 32-bit addressing is permitted for all BARs that do not have the Prefetchable bit Set.
- The minimum memory address range requested by a BAR is 128 bytes.
- A PCI Express Endpoint must appear within one of the hierarchy domains originated by the Root Complex.

PCIe 3.0&4.0 Spec

- For a PCI Express Endpoint, 64-bit addressing must be supported for all BARs that have the Prefetchable bit Set. 32-bit addressing is permitted for all BARs that do not have the Prefetchable bit Set.

5.4.2 EDK2 PCIe OpRom

[toc]

1. OpRom

1.1 0x30 Expansion ROM Base Address Register option rom Bridge 0x38

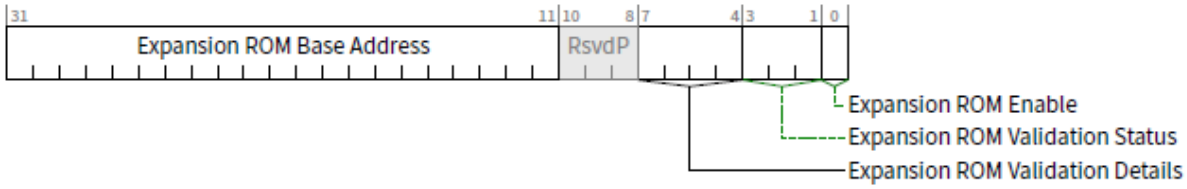


Figure 7-13 Expansion ROM Base Address Register

21bit 8bit

1.2 OpRom Plug and Play BIOS Specification

EFI takes advantage of both the **PCI Firmware Specification** and the PE/COFF Specification to store EFI images in a PCI Option ROM. There are several rules that must be followed when constructing a PCI Option ROM:

From UEFI Spec 14.4.21 PCI Option ROMs

PCI Firmware Specification 3.0

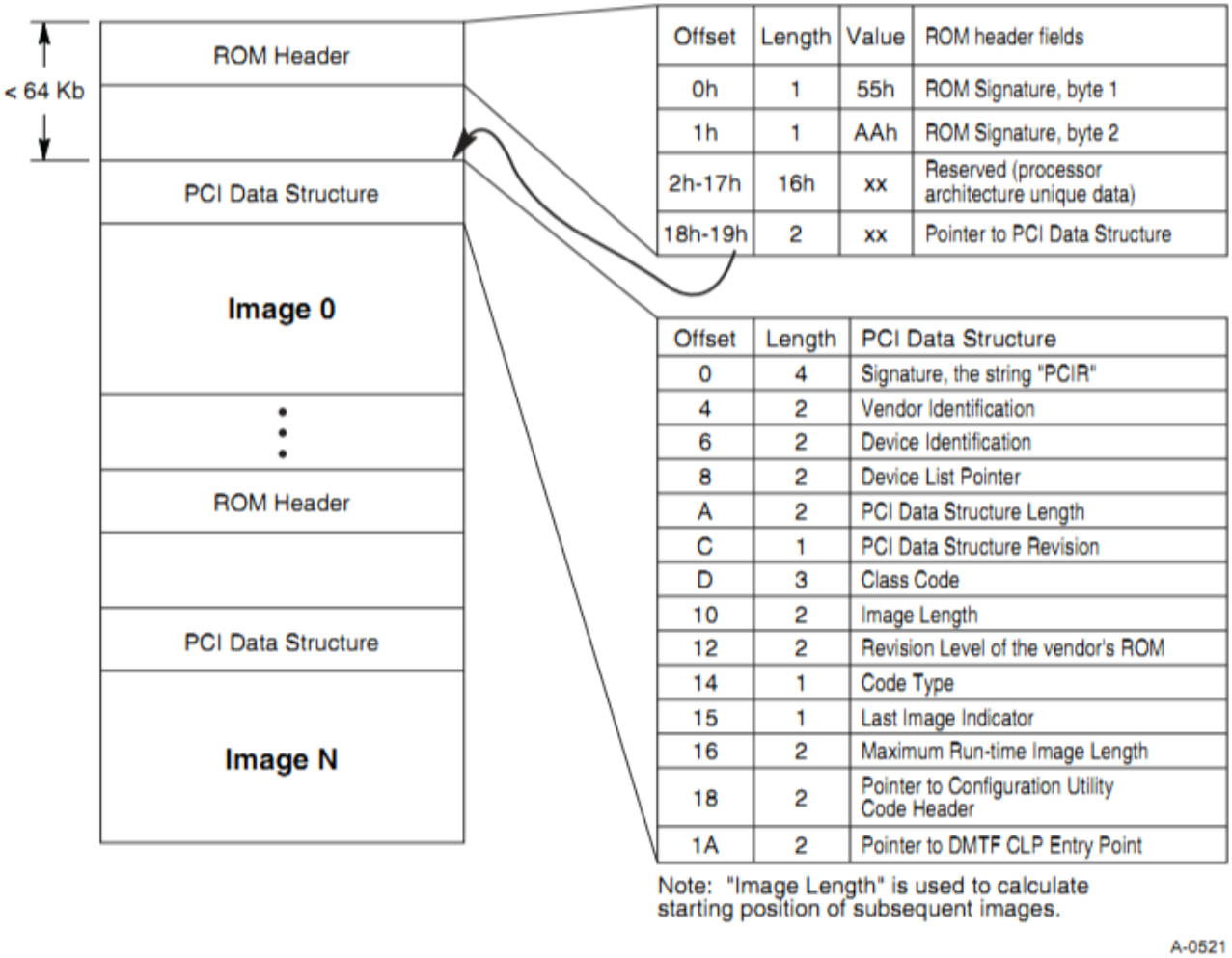


Figure 5-2: Image and Header Organization

2. OpRomImage

OpRom `PciSearchDevice() --> GetOpRomInfo()` OpRom 1 RomSize PCI_IO_DEVICE 0 1 NOT FOUND.

OpRom root bridge BAR BAR `ProcessOptionRom()`, root optionRom `LoadOpRomImage` pci

`LoadOpRomImage` Rom PCIR rom root bridge root bridge option rom size root bridge bar 0 root bridge BAR0 OptionRom Addr option rom shadow

shadow oprom program bar oprom bar oprom root bridge bar0 shadow oprom

OpRom bar oprom bar

rombar 0x55aa 512 RomSize 3.1

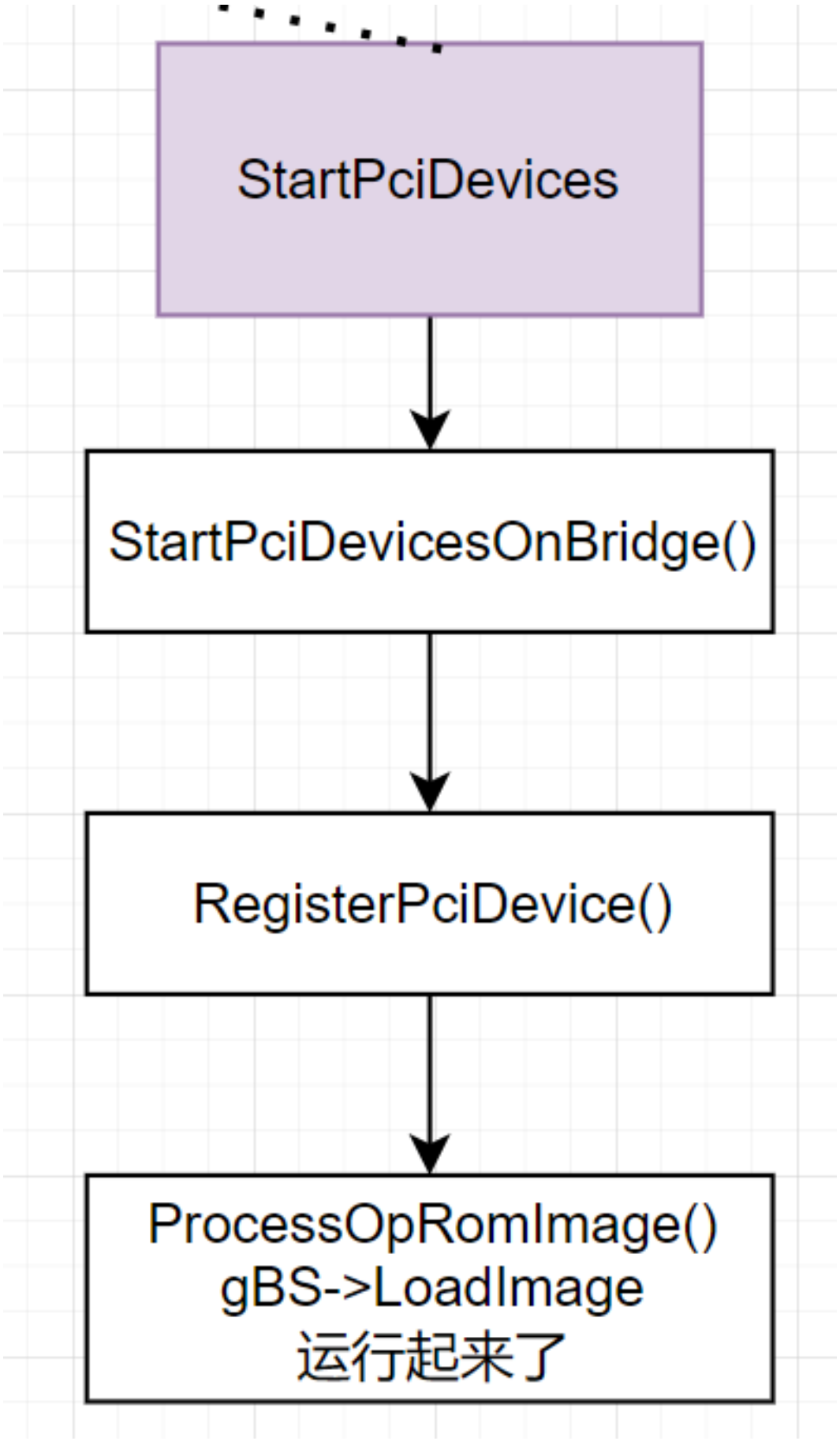
optionRom FirstCheck FALSE Pcir 0 4

PCIr image legacy LegacyImageLength

Legacy Rom size rom 0 rom

rom PciDevice->PciIo optionRom

OpRom Image `StartPciDevices` image



\edk2\MdeModulePkg\Bus\Pci\PciBusDxe\PciOptionRomSupport.c

Resources

PCIe 5.0 Spec Chapter 7.5.1.2.4

https://blog.csdn.net/robinsongsog/article/details/51785335

https://zhuanlan.zhihu.com/p/343464819

https://www.cnblogs.com/free-1122/p/16611254.html

5.5 PCIe

5.5.1 PCIe AER

5.5.2 PCIe Interrupt

5.5.3 PCIe Hot-Plug

5.5.4 PCIe Power Management

5.5.5 *DPC*

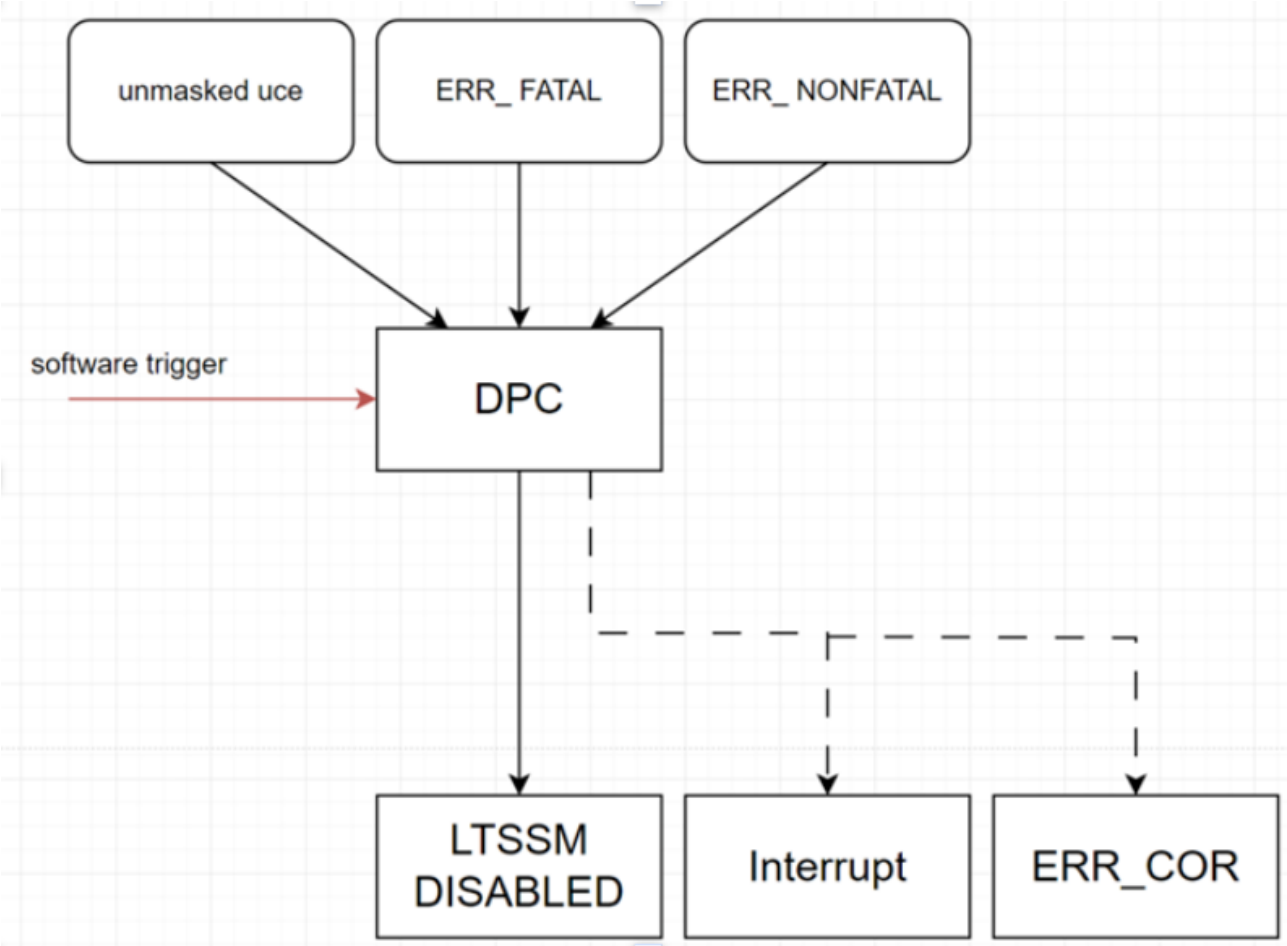
DPC downstream port containment PCIe Spec 3.1 root port pcie switch halt DPC

DPC unmasked uncorrectable error ERR_FATAL ERR_NONFATAL DPC

DPC DP DPC Trigger Status bit and DPC Trigger Reason field LTSSM disable link LTSSM disable
Trigger Status bit

DPC LTSSM detect

DPC DPC ERR_COR message



DPC Extended Capability structure

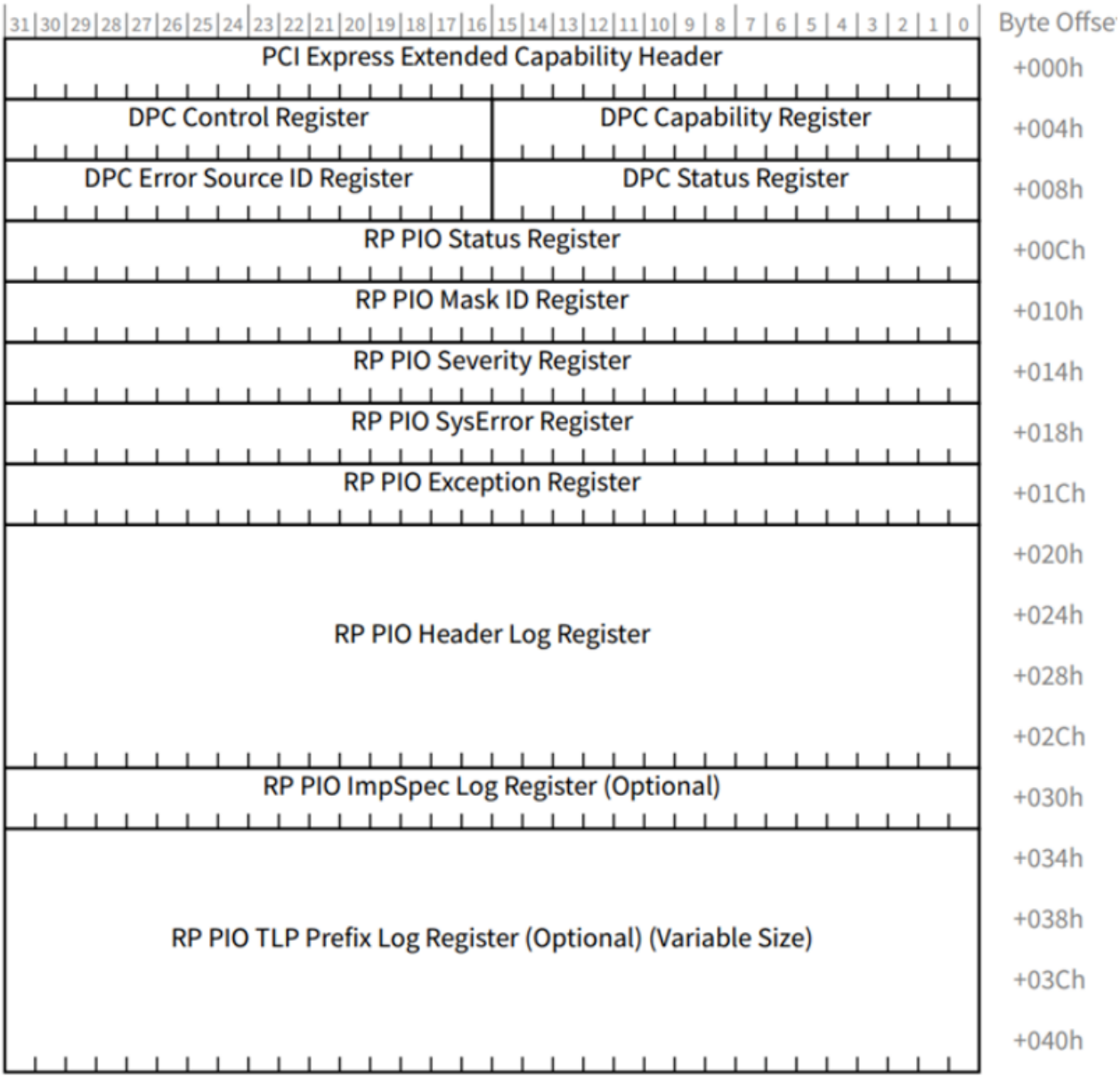


Figure 7-249 DPC Extended Capability

DPC extended capability ID 1Dh

RP PIO

DPC

DPC DPC control register

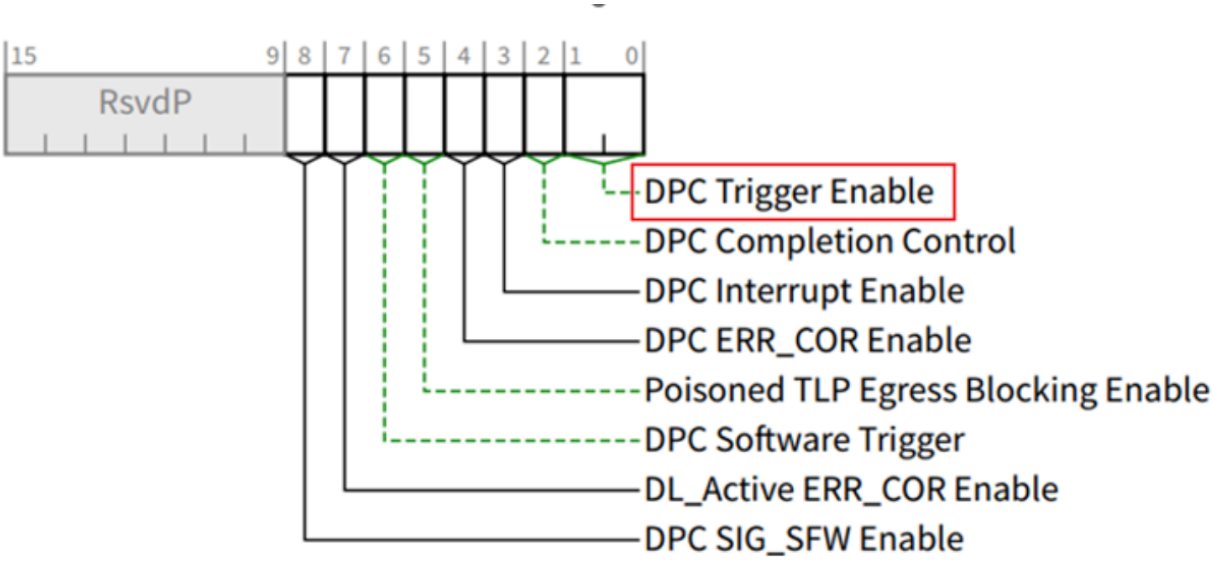


Figure 7-252 DPC Control Register

DPC Trigger Enable DPC

Bit Location	Register Description	Attributes
1:0	***DPC Trigger Enable*** - This field enables DPC and controls the conditions that cause DPC to be triggered. Defined encodings are: **00b** DPC is disabled **01b** DPC is enabled and is triggered when the Downstream Port detects an unmasked uncorrectable error or when the Downstream Port receives an ERR_FATAL Message	RW

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Bit Location	Register Description	Attributes
	10b DPC is enabled and is triggered when the Downstream Port detects an unmasked uncorrectable error or when the Downstream Port receives an ERR_NONFATAL or ERR_FATAL Message **11b** Reserved Default value of this field is 00b.	

DPC Trigger Enable field 01b DPC ERR_FATAL Message DPC

DPC Trigger Enable field 10b DPC ERR_NONFATAL or ERR_FATAL Message DPC

DPC

DPC DPC

DPC enable

Port DPC

DPC Software Triggering Supported bit in DPC Capability register 0x1

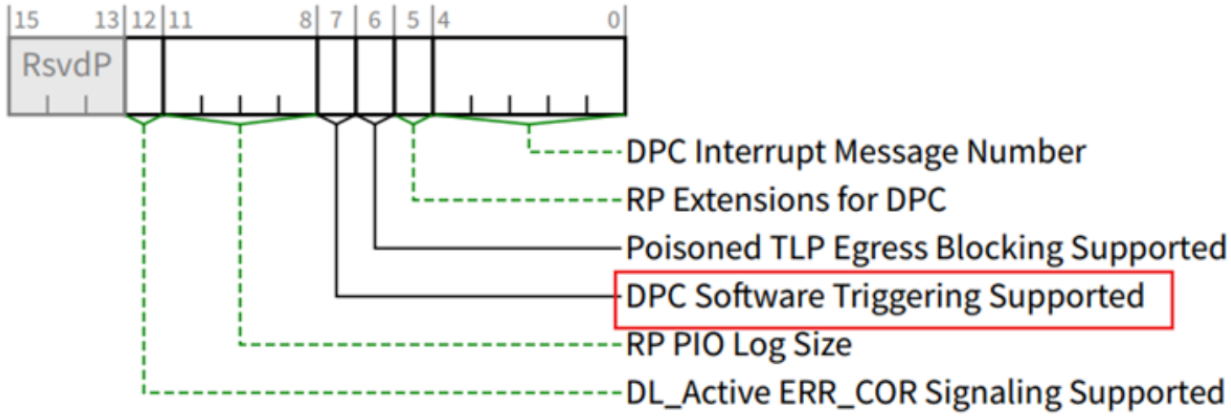


Figure 7-251 DPC Capability Register

DPC

1b DPC Software Trigger bit in the DPC Control Register Port DPC

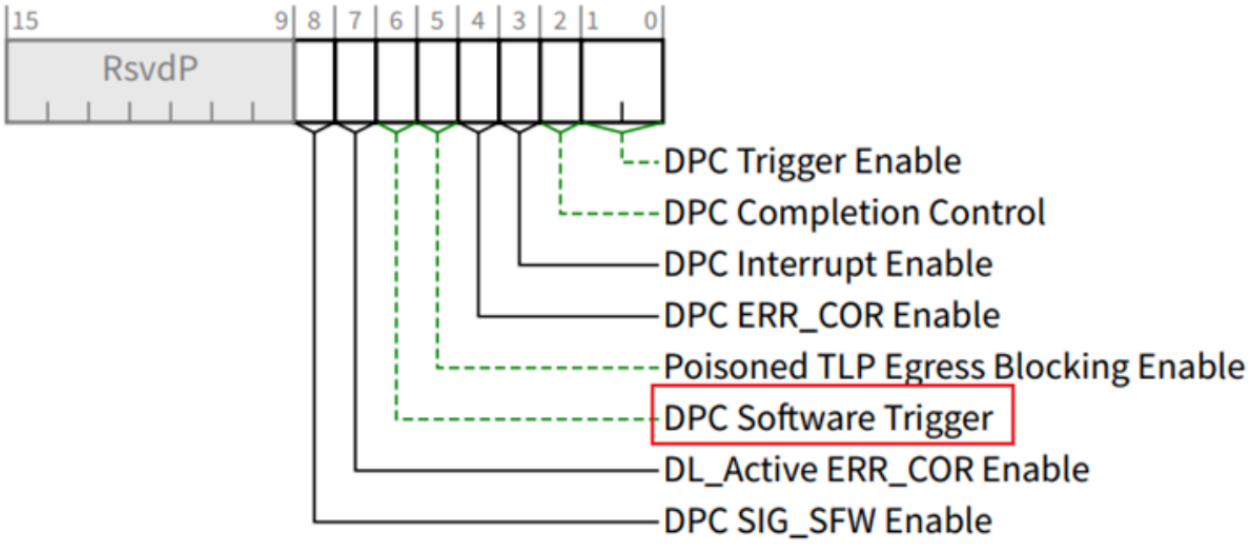


Figure 7-252 DPC Control Register

DPC Status Register

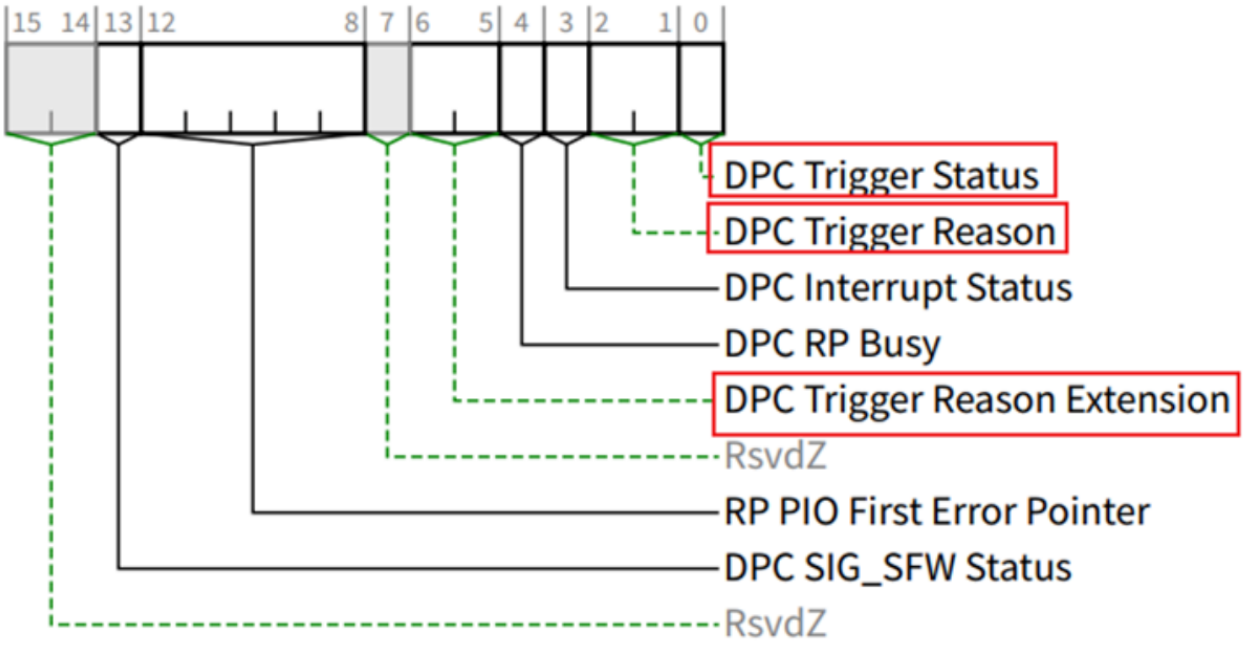


Figure 7-253 DPC Status Register

DPC Trigger Status 1 DPC

DPC Trigger Reason 0x11 DPC Trigger Reason Extension

DPC Trigger Reason Extension 0x01 DPC software Trigger

DPC

DPC DSP DPC

DPC DPC Control Register DPC Interrupt Enable bit

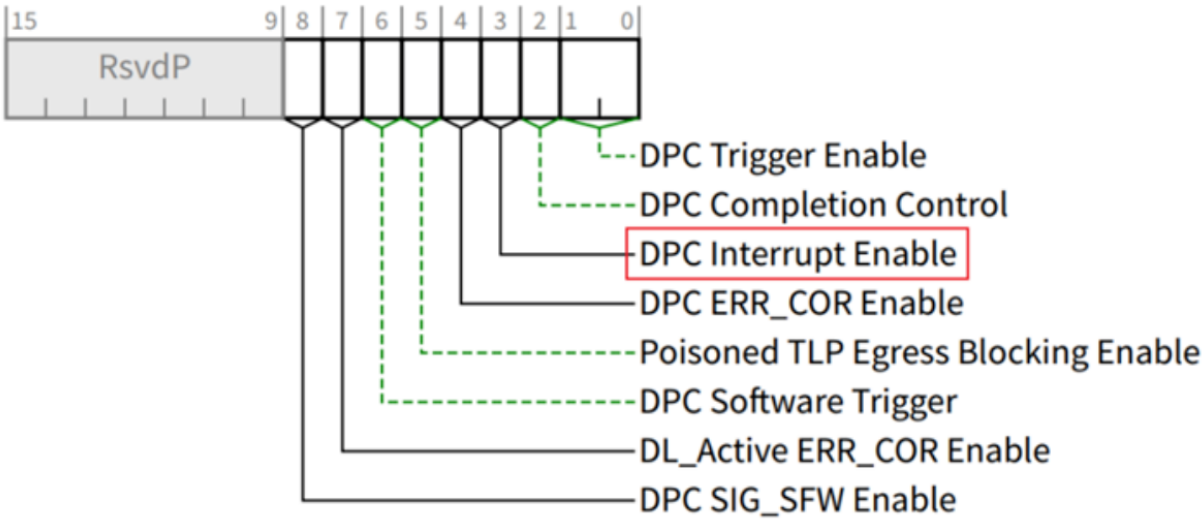
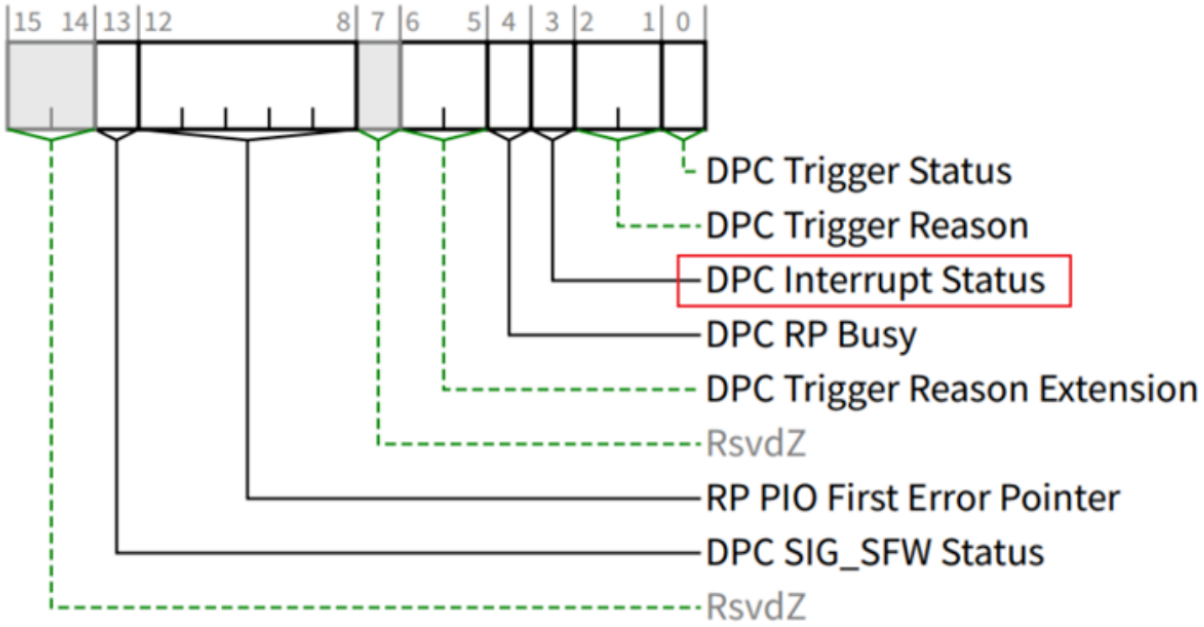


Figure 7-252 DPC Control Register

3	***DPC Interrupt Enable*** - When Set, this bit enables the generation of an interrupt to indicate that DPC has been triggered. See Section 6.2.10.1 . Default value of this bit is 0b.	RW

DPC DPC Status Register DPC Interrupt Status bit



3	***DPC Interrupt Status*** - This bit is Set if DPC is triggered while the DPC Interrupt Enable bit is Set. This may cause the generation of an interrupt. See Section 6.2.10.1 . Default value of this bit is 0b.	RW1CS

DPC ERR_COR signaling

DPC DSP ERR_COR Advanced Error Reporting (AER)

DPC ERR_COR DPC

DPC ERR_COR signaling is enabled by the DPC ERR_COR Enable bit in the DPC Control Register.

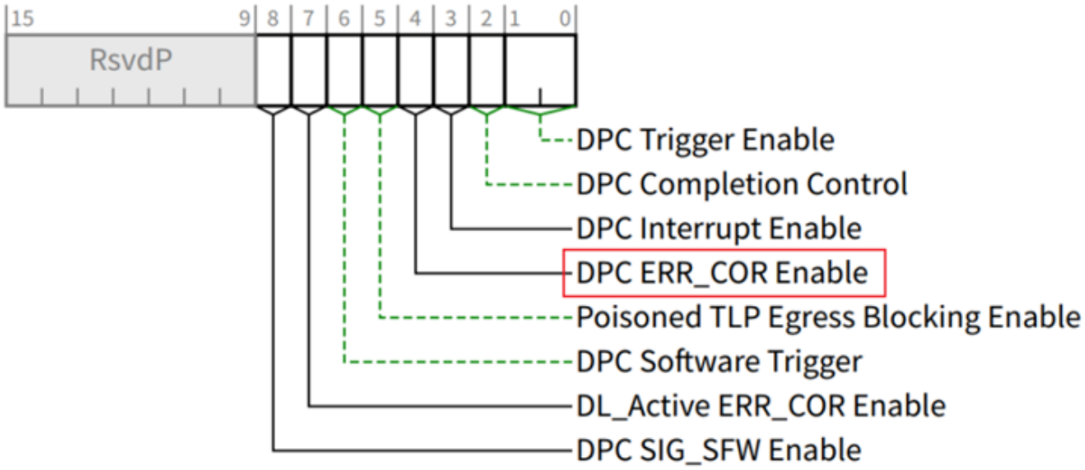


Figure 7-252 DPC Control Register

the Correctable Error Reporting Enable bit in the Device Control Register

or the DPC SIG_SFW Enable bit in the DPC Control Register is Set

PCIe Spec OS DPC FW ERR_COR signaling

DPC

DPC Trigger Status bit in DPC Status Register Root port DPC

LTSSM disbale Data Link Layer Link Active bit in the Link Status Register reads 0b DPC

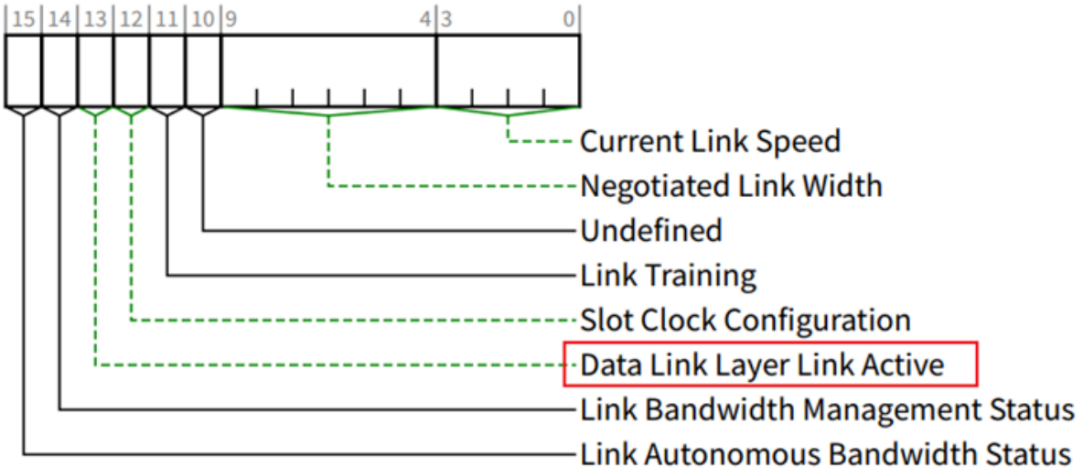


Figure 7-29 Link Status Register

Root Port RP DPC DPC RP Busy bit reads 0b.

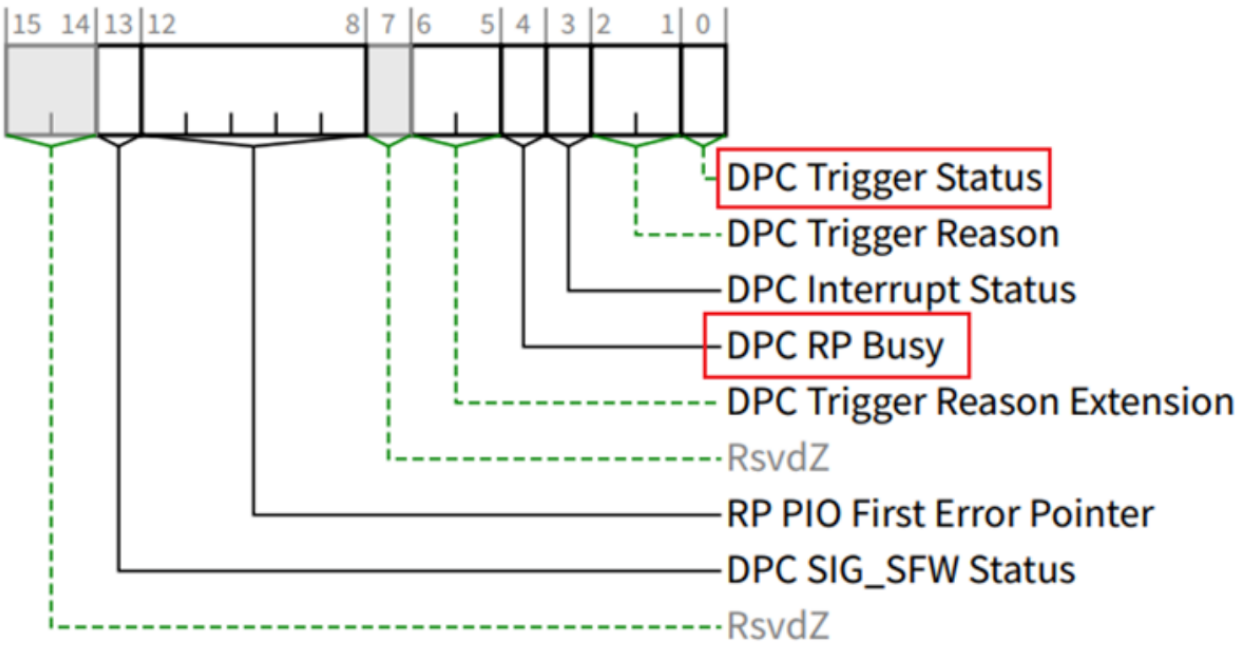


Figure 7-253 DPC Status Register

DL_Active ERR_COR signaling

DPC active DL_Active ERR_COR

DL_Active the Data Link Layer Link Active bit in the Link Status Register DPC bit

ERR_COR ERR_COR active

DL_Active ERR_COR signaling DPC control :

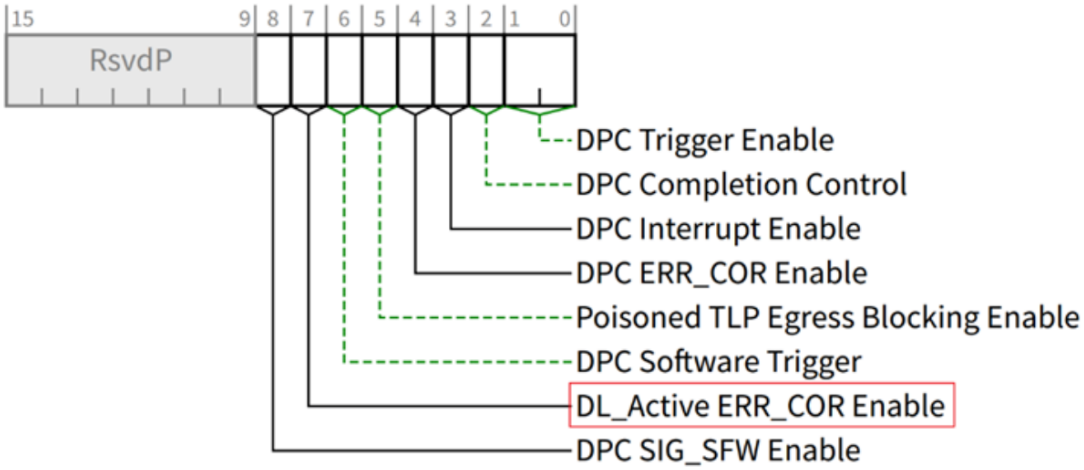


Figure 7-252 DPC Control Register

DPC ERR_COR

the Correctable Error Reporting Enable bit in the Device Control Register

or the DPC SIG_SFW Enable bit in the DPC Control Register is Set

DL_ACTIVE ERR_COR MSI/MSI-X MSI/MSI-X ERR_COR

INTx DL_ACTIVE INTx ERR_COR

eDPC

eDPC DPC PCIe Spec 3.1 DPC DPC RP PIO Root Port Programmed I/O

RP PIO PCIe RP PIO

DPC capability bit5 RP eDPC

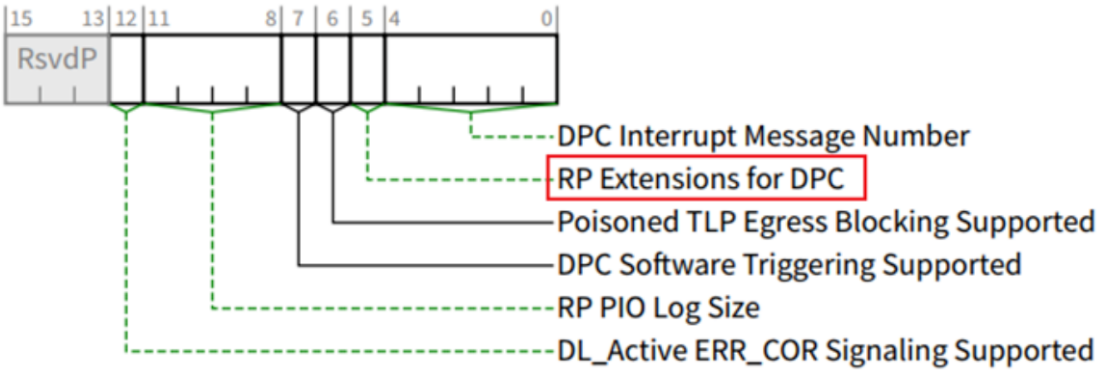
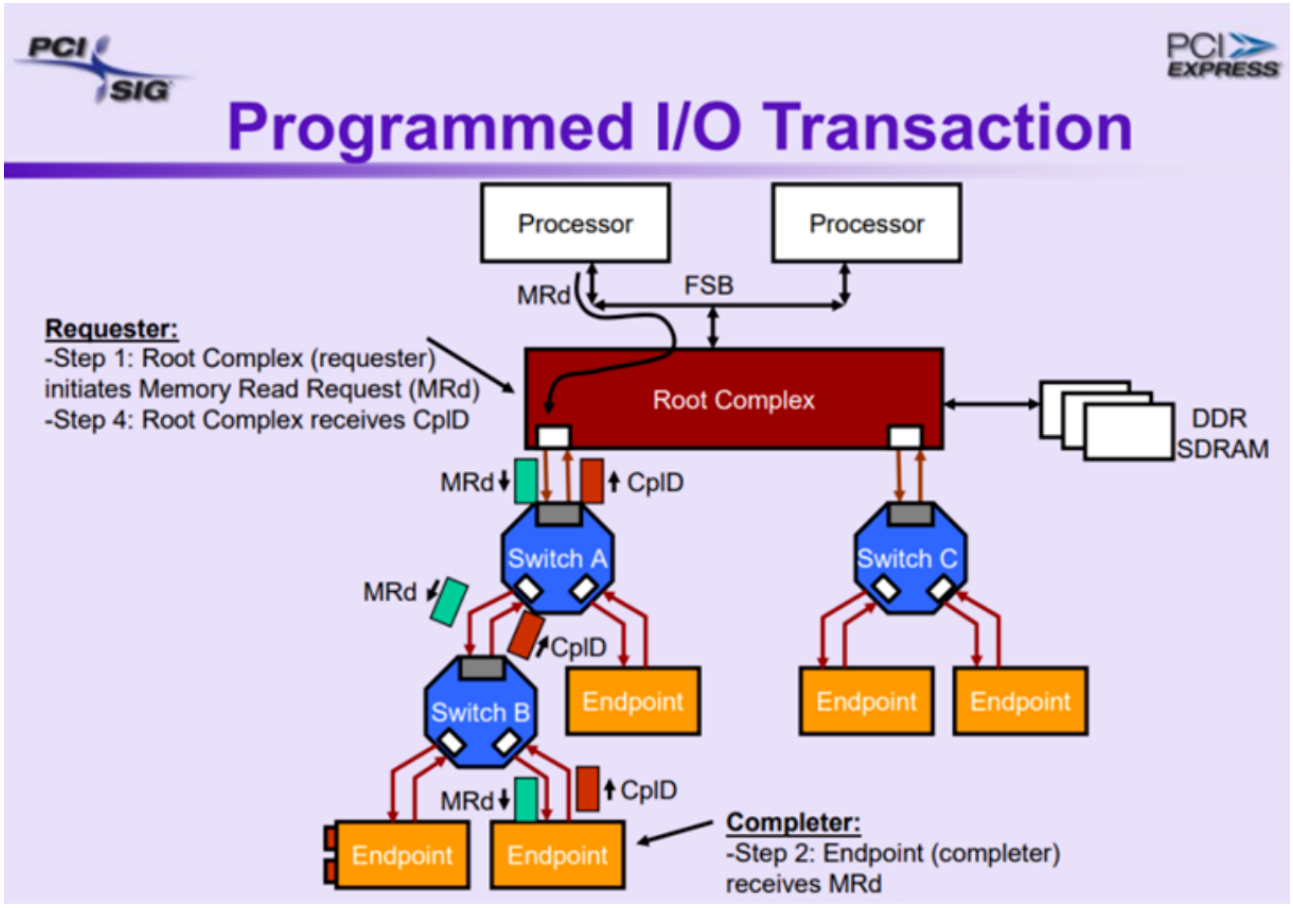
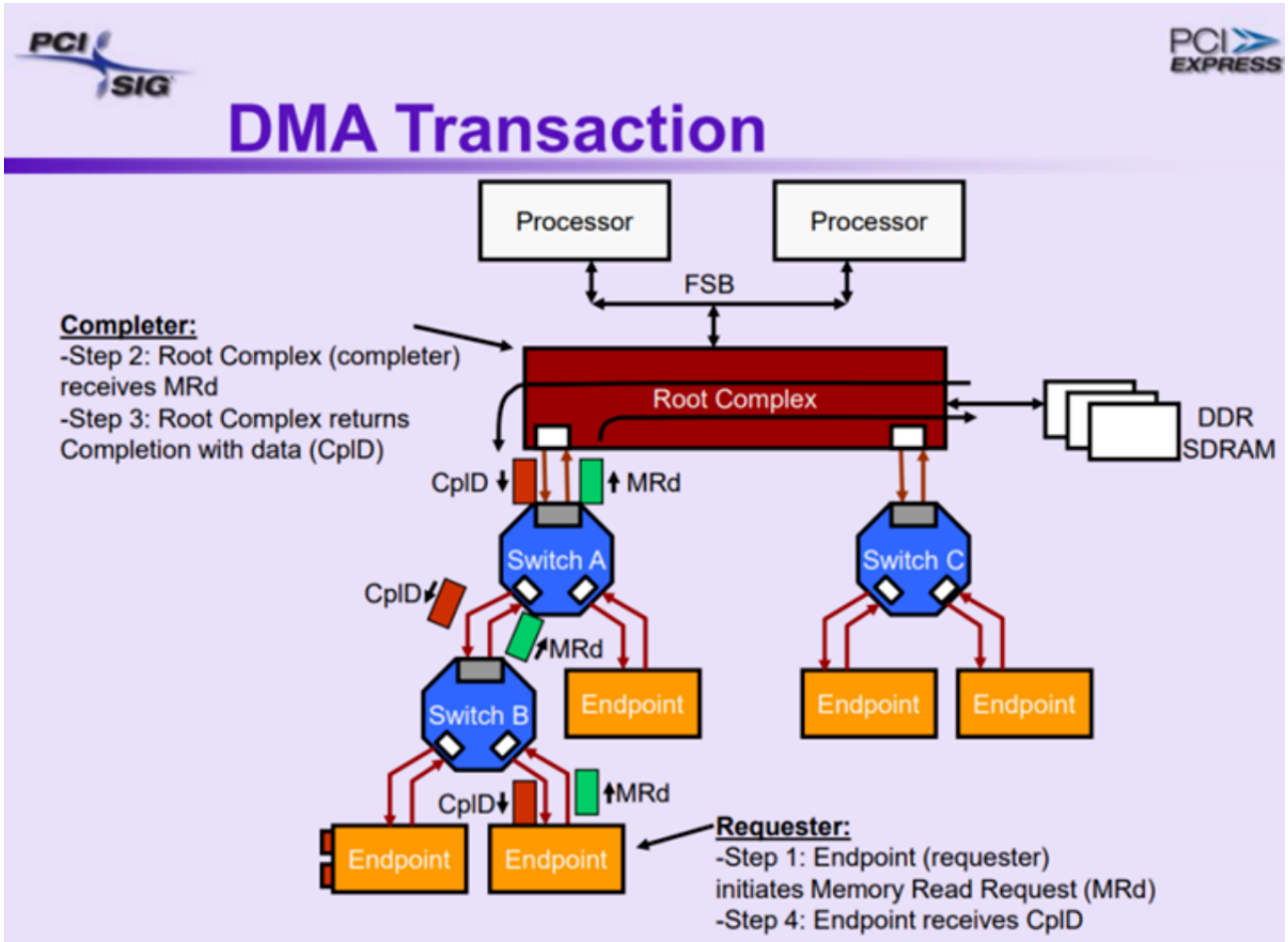


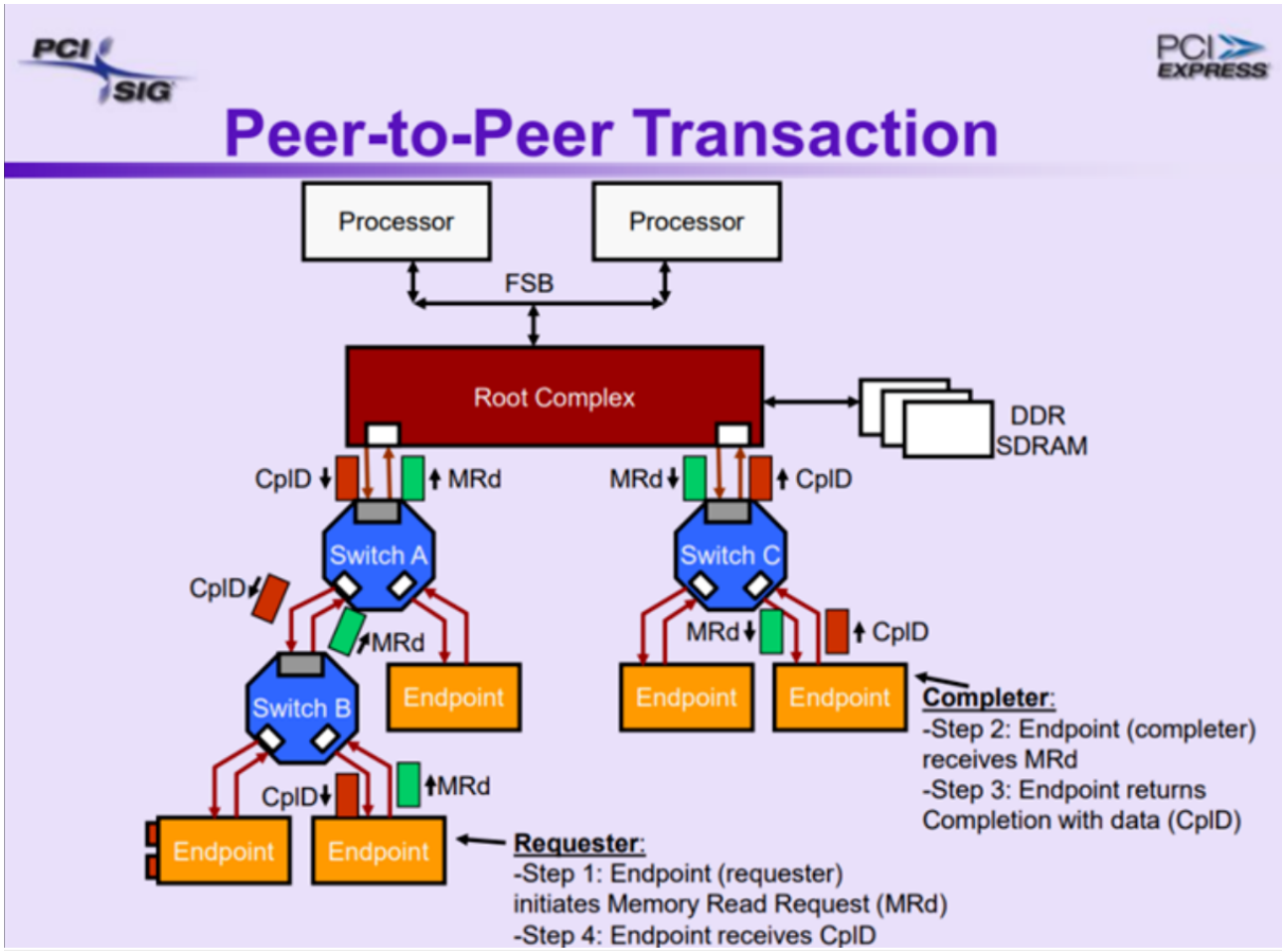
Figure 7-251 DPC Capability Register

PCIe

PIO PCIe DMA P2P


Programmed I/O Transaction
Processor
Processor
MRd
FSB
Requester:
-Step 1: Root Complex (requester) initiates Memory Read Request (MRd)
-Step 4: Root Complex receives CplD
Root Complex
DDR SDRAM
MRd
CplD
Switch A
Switch C
MRd
CplD
Switch B
Endpoint
Endpoint
Endpoint
MRd
CplD
Endpoint
Endpoint
Completer:
-Step 2: Endpoint (completer) receives MRd


DMA Transaction
Processor
Processor
FSB
Completer:
-Step 2: Root Complex (completer) receives MRd
-Step 3: Root Complex returns Completion with data (CplD)
Root Complex
DDR SDRAM
CplD
MRd
Switch A
Switch C
CplD
MRd
Switch B
Endpoint
Endpoint
Endpoint
CplD
MRd
Endpoint
Endpoint
Requester:
-Step 1: Endpoint (requester) initiates Memory Read Request (MRd)
-Step 4: Endpoint receives CplD



RP PIO

Completion with Unsupported Request status (UR Cpl)

Completion with Completer Abort status (CA Cpl)

Completion Timeout (CTO) errors

Configuration Requests I/O Requests Memory Requests RP PIO 9

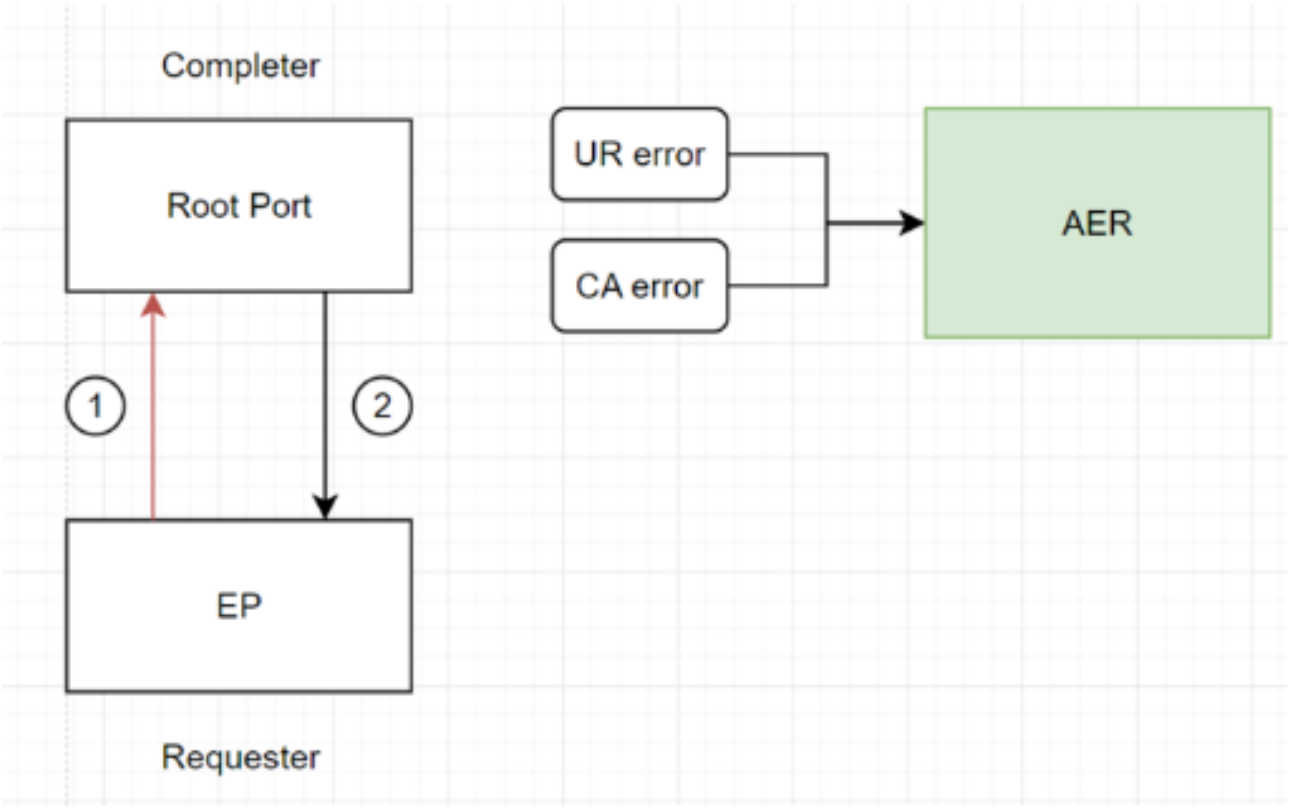
Cfg UR Cpl - Configuration Request received UR Completion
Cfg CA Cpl - Configuration Request received CA Completion
Cfg CTO - Configuration Request Completion Timeout
I/O UR Cpl - I/O Request received UR Completion
I/O CA Cpl - I/O Request received CA Completion
I/O CTO - I/O Request Completion Timeout
Mem UR Cpl - Memory Request received UR Completion
Mem CA Cpl - Memory Request received CA Completion
Mem CTO - Memory Request Completion Timeout

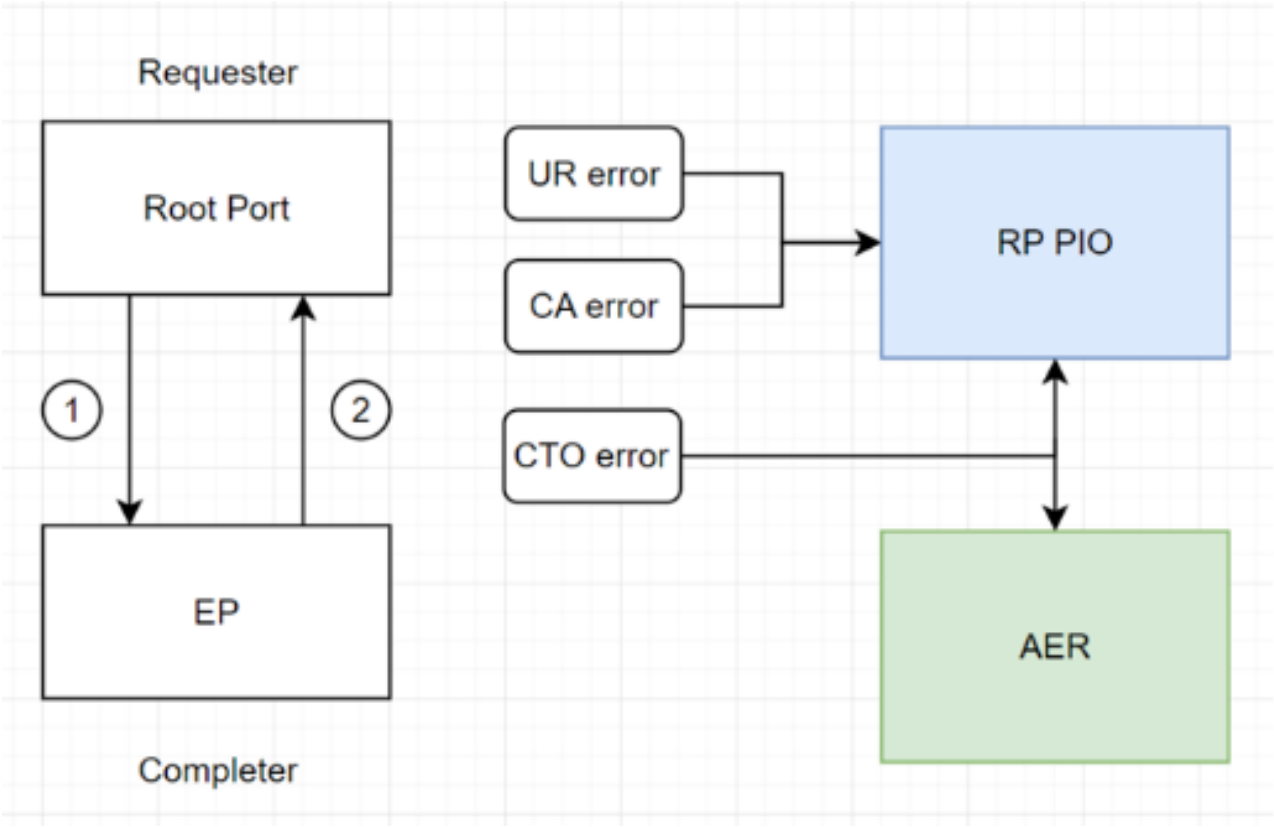
AER UR CA CTO RP PIO error control

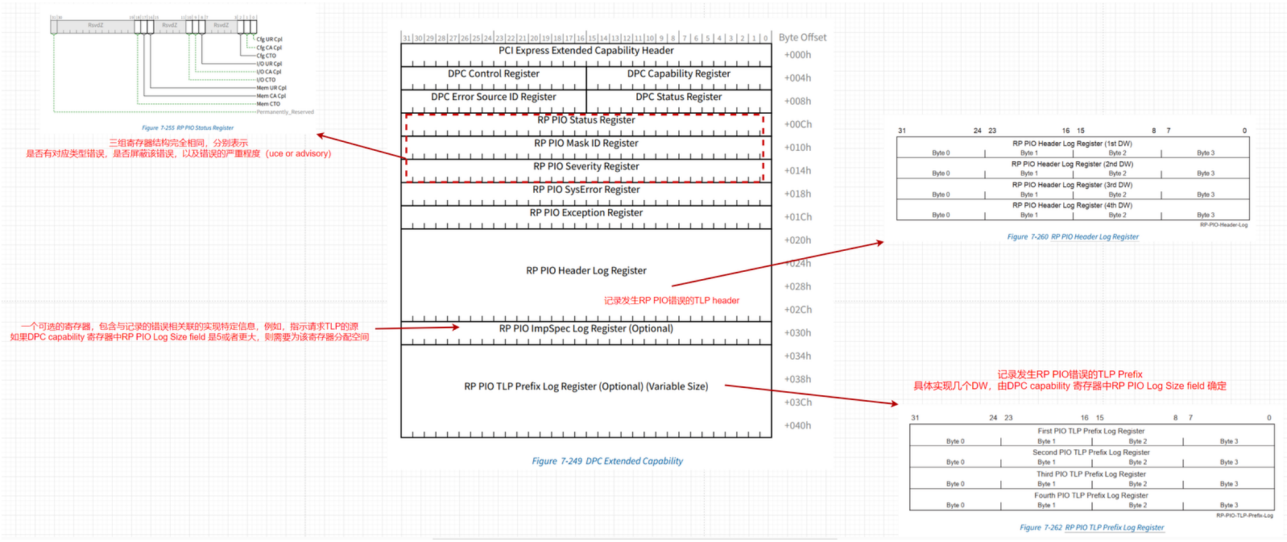
RP Completer UR CA error AER

RP Requester UR CA error RP PIO CTO error AER RP PIO

RP PIO CTO error AER CTO error


Completer
Root Port
1
2
EP
Requester
UR error
CA error
AER


Requester
Root Port
1
2
EP
Completer
UR error
CA error
CTO error
RP PIO
AER



RP PIO error control DPC extended capability RP PIO AER capability

The RP PIO Status, Mask, and Severity registers AER the Uncorrectable Error Status, Mask, and Severity registers

RP PIO RP PIO Status Register RP PIO log registers RP PIO

RP PIO RP PIO Status Register DPC RP PIO log registers

RP PIO uncorrectable or advisory RP PIO Severity Register

Severity bit uncorrectable DPC(DPC) DPC / ERR_COR(if enabled)

Severity bit Clear (DPC) ERR_COR(if enabled)

The RP PIO Header Log Register, RP PIO ImpSpec Log Register, and RP PIO TLP Prefix Log Registers RP PIO log registers

The RP PIO First Error Pointer, RP PIO Header Log, and RP PIO TLP Prefix Log AER the First Error Pointer, Header Log, and TLP Prefix Log

RP PIO Header Log RP PIO error TLP header;

RP PIO ImpSpec Log Register request TLP DPC capability RP Log Size field 5

RP PIO TLP Prefix Log Register RP PIO error TLP Prefix DW DPC capability RP Log Size field

Number = RP Log Size - 5 if RP Log Size <=9;

Number =4 if RP Log Size >9;

Linux Kernel DPC

DPC

Linux kernel 5.10.201 DPC RAS C2000 DPC RAS

```c
static struct pcie_port_service_driver dpcdriver = {
    .name       = "dpc",
    .port_type  = PCIE_ANY_PORT,
    .service    = PCIE_PORT_SERVICE_DPC,
    .probe      = dpc_probe,
    .remove     = dpc_remove,
};
```

dpc_probe DPC

```c
349 static int dpc_probe(struct pcie_device *dev)
350 {
351     struct pci_dev *pdev = dev->port;
352     struct device *device = &dev->device;
353     int status;
354     u16 ctl, cap;
355
356     if (!pcie_aer_is_native(pdev) && !pcie_ports_dpc_native)
357         return -ENOTSUPP;
358
359     status = devm_request_threaded_irq(device, dev->irq, dpc_irq,
360                                        dpc_handler, IRQF_SHARED,
361                                        "pcie-dpc", pdev);
362     if (status) {
363         pci_warn(pdev, "request IRQ%d failed: %d\n", dev->irq,
364                  status);
365         return status;
366     }
367
368     pci_read_config_word(pdev, pdev->dpc_cap + PCI_EXP_DPC_CAP, &cap);
369     pci_read_config_word(pdev, pdev->dpc_cap + PCI_EXP_DPC_CTL, &ctl);
370
371     ctl = (ctl & 0xfff4) | PCI_EXP_DPC_CTL_EN_FATAL | PCI_EXP_DPC_CTL_INT_EN;
372     pci_write_config_word(pdev, pdev->dpc_cap + PCI_EXP_DPC_CTL, ctl);
373     pci_info(pdev, "enabled with IRQ %d\n", dev->irq);
374
375     pci_info(pdev, "error containment capabilities: Int Msg #%d, RPExt%c PoisonedTLP%c SwTrigger%c RP PIO Log %d, DL_ActiveErr%c\n",
376              cap & PCI_EXP_DPC_IRQ, FLAG(cap, PCI_EXP_DPC_CAP_RP_EXT),
377              FLAG(cap, PCI_EXP_DPC_CAP_POISONED_TLP),
378              FLAG(cap, PCI_EXP_DPC_CAP_SW_TRIGGER), pdev->dpc_rp_log_size,
379              FLAG(cap, PCI_EXP_DPC_CAP_DL_ACTIVE));
380
381     pci_add_ext_cap_save_buffer(pdev, PCI_EXT_CAP_ID_DPC, sizeof(u16));
382     return status;
383 }
```

line356 PCIe AER DPC -ENOTSUPP

line359 devm_request_threaded_irq dev->irq status

devm_request_threaded_irq Linux IRQ dpc_irq dpc_handler top half bottom half

line362

line368 DPC capability control

line371 control enable Fatal error control

dpc_irq

```c
static irqreturn_t dpc_irq(int irq, void *context)
{
    struct pci_dev *pdev = context;
    u16 cap = pdev->dpc_cap, status;

    pci_read_config_word(pdev, cap + PCI_EXP_DPC_STATUS, &status);

    if (!(status & PCI_EXP_DPC_STATUS_INTERRUPT) || status == (u16)(~0))
        return IRQ_NONE;

    pci_write_config_word(pdev, cap + PCI_EXP_DPC_STATUS,
                          PCI_EXP_DPC_STATUS_INTERRUPT);
    if (status & PCI_EXP_DPC_STATUS_TRIGGER)
        return IRQ_WAKE_THREAD;
    return IRQ_HANDLED;
}
```

DPC status status 1

status 0x8 dpc 1b

DPC return IRQ_WAKE_THREAD;

IRQ_WAKE_THREAD

IRQ_HANDLED

dpc_handler

```c
298  static irqreturn_t dpc_handler(int irq, void *context)
299  {
300      struct pci_dev *pdev = context;
301
302      dpc_process_error(pdev);
303
304      /* We configure DPC so it only triggers on ERR_FATAL */
305      pcie_do_recovery(pdev, pci_channel_io_frozen, dpc_reset_link);
306
307      return IRQ_HANDLED;
308  }
```

dpc_process_error pdev DPC

pcie_do_recovery pdev pci_channel_io_frozen dpc_reset_link DPC

IRQ_HANDLED

```
265  void dpc_process_error(struct pci_dev *pdev)
266  {
267      u16 cap = pdev->dpc_cap, status, source, reason, ext_reason;
268      struct aer_err_info info;
269
270      pci_read_config_word(pdev, cap + PCI_EXP_DPC_STATUS, &status);
271      pci_read_config_word(pdev, cap + PCI_EXP_DPC_SOURCE_ID, &source);
272
273      pci_info(pdev, "containment event, status:%#06x source:%#06x\n",
274               status, source);
275
276      reason = (status & PCI_EXP_DPC_STATUS_TRIGGER_RSN) >> 1;
277      ext_reason = (status & PCI_EXP_DPC_STATUS_TRIGGER_RSN_EXT) >> 5;
278      pci_warn(pdev, "%s detected\n",
279               (reason == 0) ? "unmasked uncorrectable error" :
280               (reason == 1) ? "ERR_NONFATAL" :
281               (reason == 2) ? "ERR_FATAL" :
282               (ext_reason == 0) ? "RP PIO error" :
283               (ext_reason == 1) ? "software trigger" :
284                                   "reserved error");
285
286      /* show RP PIO error detail information */
287      if (pdev->dpc_rp_extensions && reason == 3 && ext_reason == 0)
288          dpc_process_rp_pio_error(pdev);
289      else if (reason == 0 &&
290               dpc_get_aer_uncorrect_severity(pdev, &info) &&
291               aer_get_device_error_info(pdev, &info)) {
292          aer_print_error(pdev, &info);
293          pci_aer_clear_nonfatal_status(pdev);
294          pci_aer_clear_fatal_status(pdev);
295      }
```

status source ID

raw data

line287 DPC RP reason 3 ext_reason 0 DPC RP PIO dpc_process_rp_pio_error

line289 reason 0 unmasked uncorrectable error AER Advanced Error Reporting uce AER

dpc_process_rp_pio_error

pcie_do_recovery(pdev, pci_channel_io_frozen, dpc_reset_link);

err.c PCIe

I/O

/ I/O /

“ ”

dpc_reset_link dpc.c DPC

reset_subordinates

PCIe pcie_do_recovery

Root Port Downstream Port RCEC

Endpoint Port

reset_subordinates

dpc_reset_link

```
146  pci_ers_result_t dpc_reset_link(struct pci_dev *pdev)
147  {
148      pci_ers_result_t ret;
149      u16 cap;
150
151      set_bit(PCI_DPC_RECOVERING, &pdev->priv_flags);
152
153      /*
154       * DPC disables the Link automatically in hardware, so it has
155       * already been reset by the time we get here.
156       */
157      cap = pdev->dpc_cap;
158
159      /*
160       * Wait until the Link is inactive, then clear DPC Trigger Status
161       * to allow the Port to leave DPC.
162       */
163      if (!pcie_wait_for_link(pdev, false))
164          pci_info(pdev, "Data Link Layer Link Active not cleared in 1000 msec\n");
165
166      if (pdev->dpc_rp_extensions && dpc_wait_rp_inactive(pdev)) {
167          clear_bit(PCI_DPC_RECOVERED, &pdev->priv_flags);
168          ret = PCI_ERS_RESULT_DISCONNECT;
169          goto out;
170      }
171
172      pci_write_config_word(pdev, cap + PCI_EXP_DPC_STATUS,
173                            PCI_EXP_DPC_STATUS_TRIGGER);
174
175      if (pci_bridge_wait_for_secondary_bus(pdev, "DPC",
176                                            PCIE_RESET_READY_POLL_MS)) {
177          clear_bit(PCI_DPC_RECOVERED, &pdev->priv_flags);
178          ret = PCI_ERS_RESULT_DISCONNECT;
179      } else {
180          set_bit(PCI_DPC_RECOVERED, &pdev->priv_flags);
181          ret = PCI_ERS_RESULT_RECOVERED;
182      }
183  out:
184      clear_bit(PCI_DPC_RECOVERING, &pdev->priv_flags);
185      wake_up_all(&dpc_completed_waitqueue);
186      return ret;
187  }
```

line151 DPC

line163 DPC pcie_wait_for_link inactive 100 true false

line172 status bit0 1b DPC **PCIe 5.0 Spec** **spec** **disabled** **DPC** **100ms** **warni**

Disabled state or at least to bring the Link down under a variety of error conditions, software must leave the Downstream Port in DPC until the Data Link Layer Link Active bit in the Link Status Register reads 0b; otherwise, the result is undefined. See Section 7.5.3.8 . See Section 2.9.3 for other important details on Transaction Layer behavior during DPC.

line175 PCI_DPC_RECOVERED PCI_ERS_RESULT_DISCONNECT

line179 PCI_DPC_RECOVERED PCI_ERS_RESULT_RECOVERED

line184 DPC

Chapter 6

CXL

6.1 CXL

[toc]

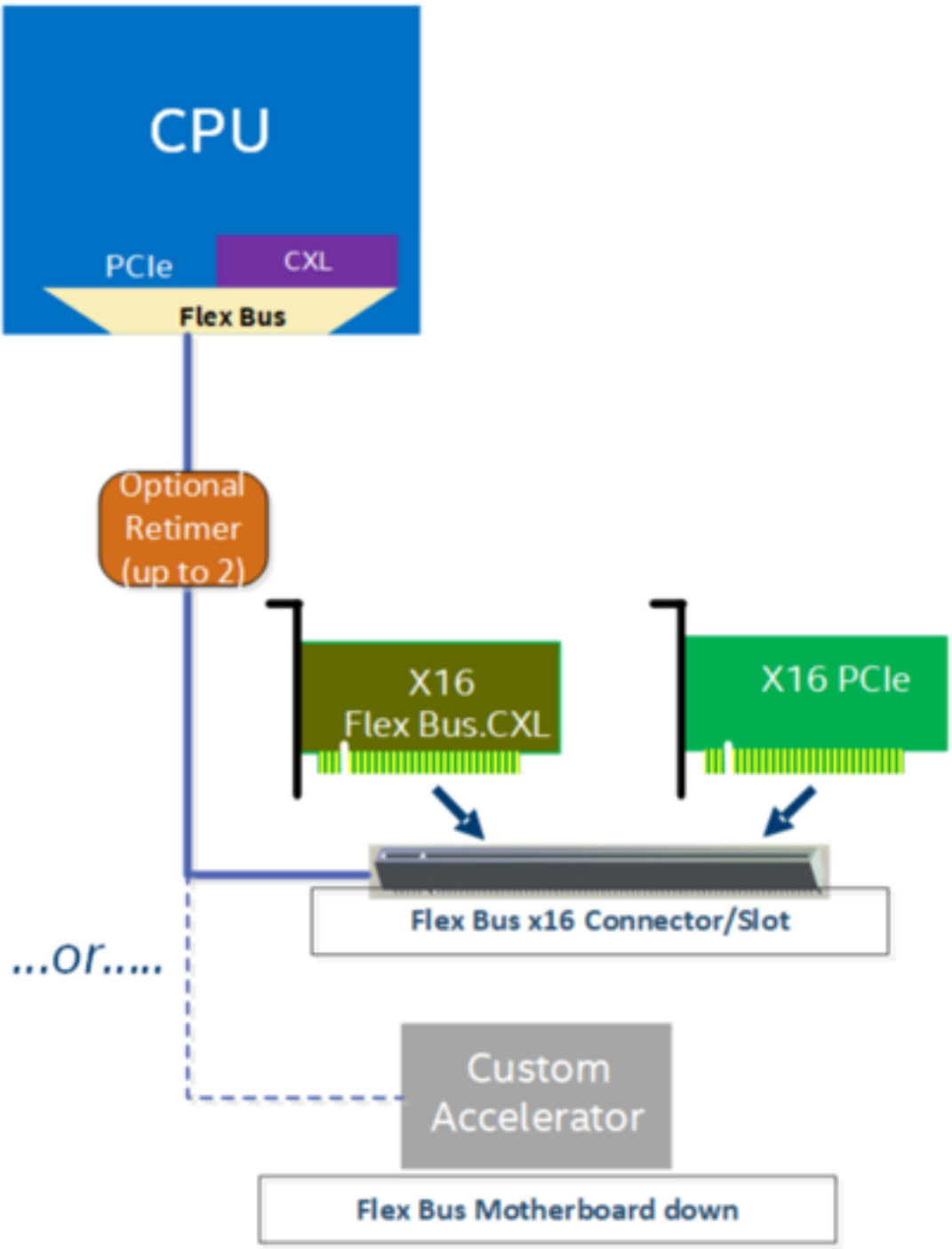
Overview

CXL PCI Express 5.0

CXL I/O(CXL.io PCIe) (CXL.cache) (CXL.memory)

CXL I/O(CXL.io PCIe) (CXL.cache) (CXL.memory)

CXL PCIe CXL PCIe CXL PCIe Gen1 Link Training PCIe 5.0 Alternate Negotiation Protocol CXL CXL PCIe 5.0 CXL PHY



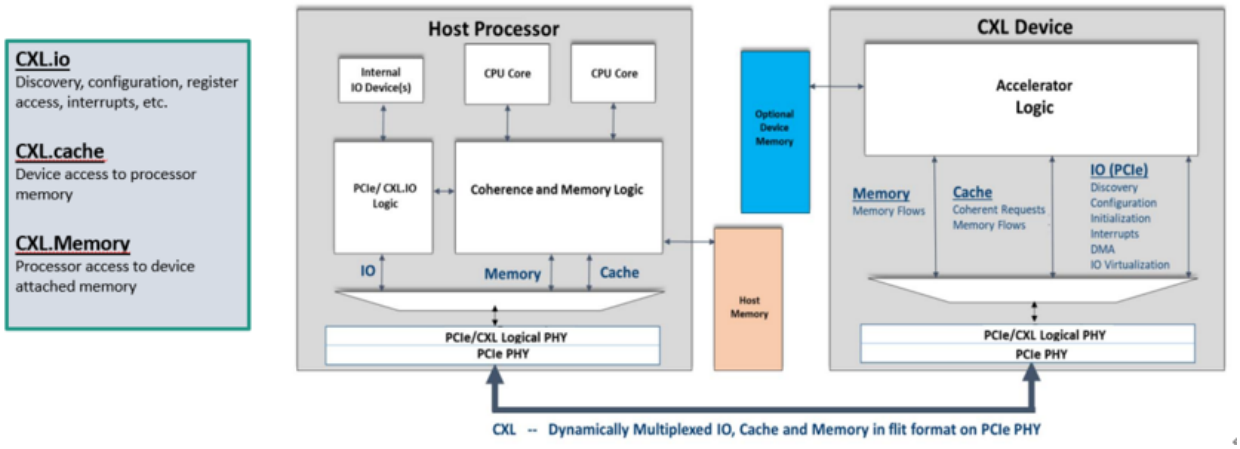
CXL Device Type

CXL Device CXL.io CXL.cache CXL.mem CXL.io CXL.cache CXL.io CXL.mem

CXL.io PCIe I/O (DMA) - PCIe CXL.cache CXL.mem

CXL.cache Snoop Messages

CXL.mem CXL Memroy Device CXL.mem Load Store

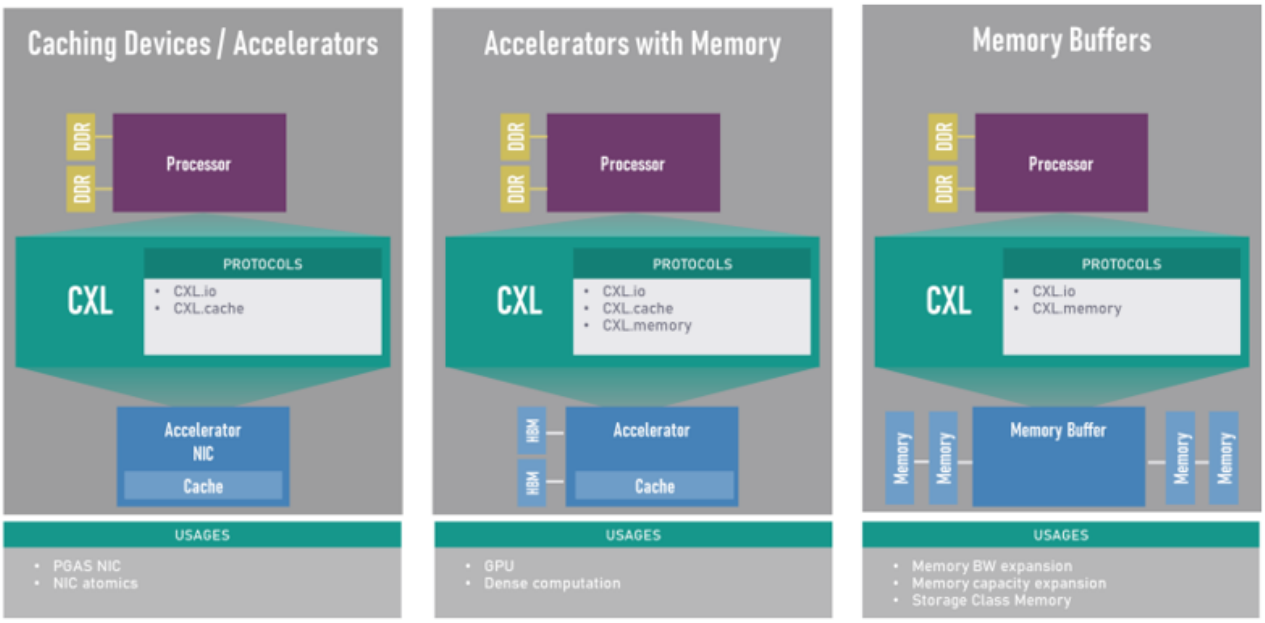


CXL Spec CXL Device CXL Device

Type 1 CXL Device: CXL.io CXL.cache CXL Device

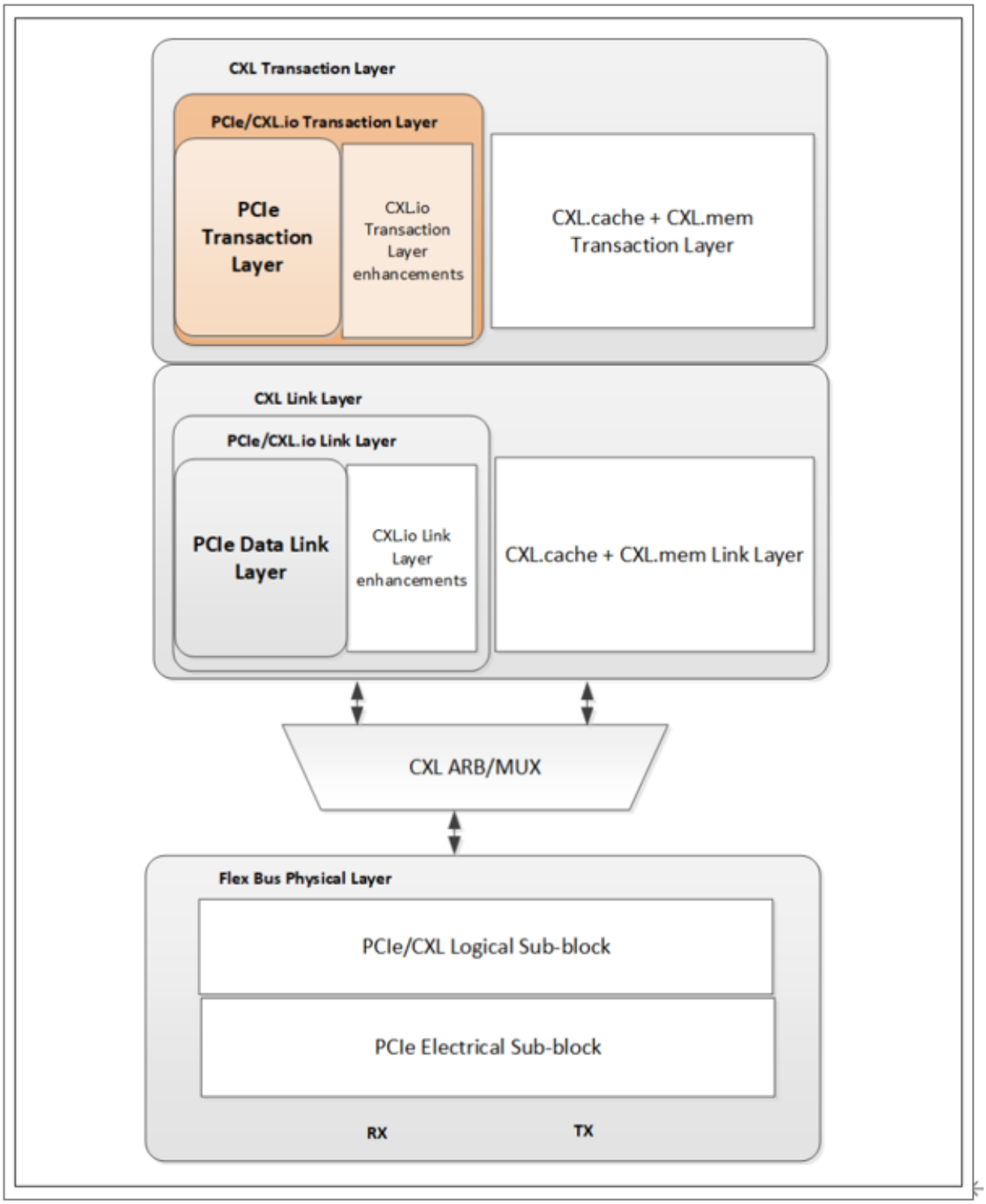
Type 2 CXL Device: CXL Device HBM GPU

Type 3 CXL Device: CXL.io CXL.mem CXL Device CXL Memory Expander



CXL.io Overview

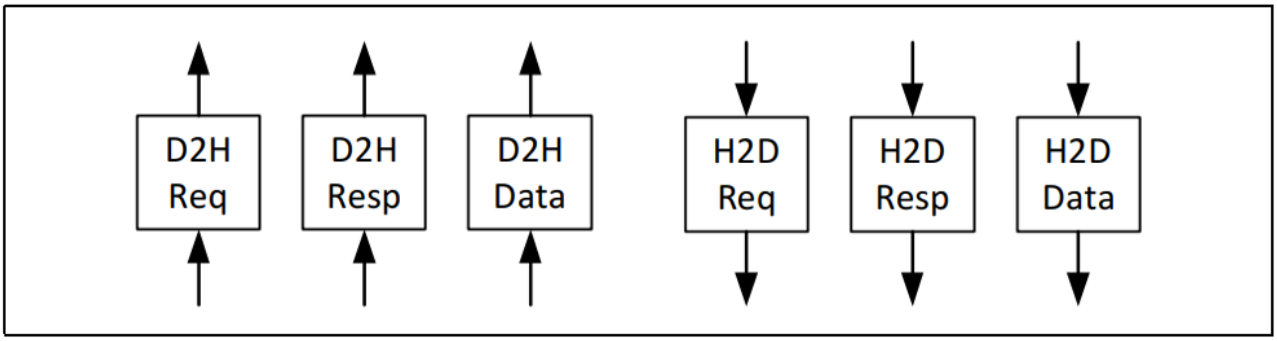
CXL.io I/O non-coherent load/store CXL.io Flex Bus PCIe PCI Express 5.0 Base Specification



CXL.cache Overview

CXL.cache CXL.cache Channels D2H H2D

CXL.cache CXL.cache Channels D2H H2D



H2D Request snoops snoops H2D Response ordering messages Overview

CXL.mem Overview

CXL Memory Protocol CXL.mem CPU Device Memory Die Compute Express Link(CXL) Phy Link Layer Memory Controller Host CPU Memory Controller Accelerator Memory Controller Volat Persistent Flat Hierarchical

CPU Coherency Engine CXL.mem Request Response CPU Coherency Engine CXL.mem Master Mem Device CXL.mem Subordinate CXL.mem Master CXL.mem CXL.mem Subordinate CXL.mem

Subordinate Device Accelerator CXL.mem Protocol Device Coherency Engine(DCOH) DCOH CXL.mem

CXL.mem Master Subordinate "M2S" Subordinate Master "S2M"

M2S

- -- Req
- -- RwD

S2M

- -- (NDR)
- -- DRS

CXL

Features	CXL 1.0 / 1.1	CXL 2.0	CXL 3.0
Release date	2019	2020	2022
Max link rate	32GTs	32GTs	64GTs
Flit 68 byte (up to 32 GTs)	✓	✓	✓
Flit 256 byte (up to 64 GTs)			✓
Type 1, Type 2 and Type 3 Devices	✓	✓	✓
Memory Pooling w/ MLDs		✓	✓
Global Persistent Flush		✓	✓
CXL IDE		✓	✓
Switching (Single-level)		✓	✓
Switching (Multi-level)			✓
Direct memory access for peer-to-peer			✓
Enhanced coherency (256 byte flit)			✓
Memory sharing (256 byte flit)			✓
Multiple Type 1/Type 2 devices per root port			✓
Fabrics (256 byte flit)			✓

Not supported
✓ Supported

Firmware Device

- CXL2.0 Device Component Register HDM Decoder Component Register CXL HDM Decoder Global Control Register (Offset 04h) ->HDM Decoder Enable DVSEC ID 0 Range Size Range Base Memory Address
- CXL 1.1 Device DVSEC ID 0 Range Size Range Base
- CXL Root Port CXL Switch Upstream Port HDM Decoder
- PCIe DVSEC for CXL Device CXL 2.0 Extensions DVSEC for Ports Mem_Enable

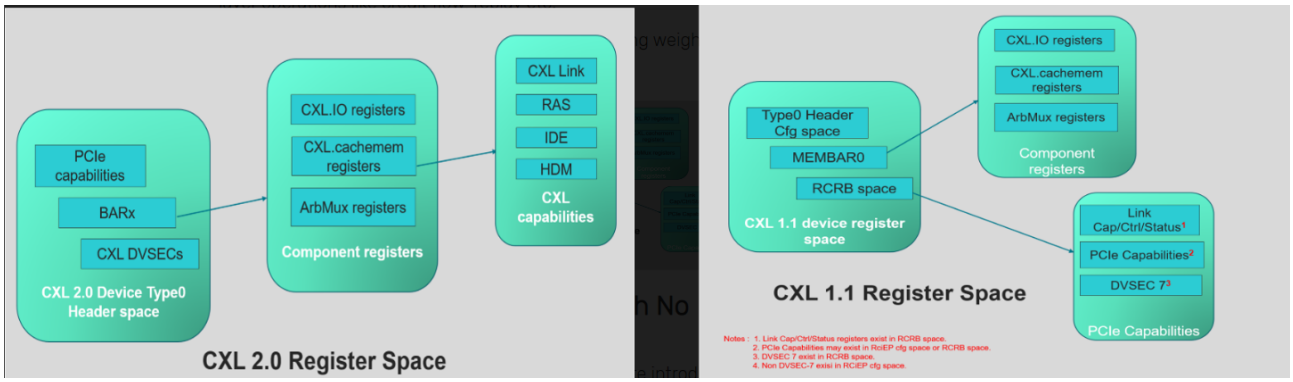
CXL HDM Decoder

- HDM Decoder Host HDM Decoder
- HDM Decoder
- HDM Decoder DRAM NVRAM

6.2 CXL

CXL

PCIe Configuration Space MMIO Memory Mapped Space Memory



DVSEC

PCIe DVSEC CXL CXL CXL vendor ID 0x1E98.

CXL 3.0 9 CXL DVSEC

- PCIe DVSEC for CXL Devices CXL PCIe DVSEC RCD LD SLD FMLD CXL Device D0F0 DVSEC Device DVSEC Device PCIe Device CXL Device
- Non-CXL Function Map DVSEC Device Function CXL.cachemem Device Function
- CXL Extensions DVSEC for Ports RP DSP USP PCIe RCH-RCD
- GPF DVSEC for CXL Ports CXL Port GPF GPF Phase1 Phase2 Timeout
- GPF DVSEC for CXL Devices CXL Device GPF GPF Phase2 Timeout Phase2
- PCIe DVSEC for Flex Bus Port Flex Bus Port DVSEC Flex Bus Multi-Device CXL 1.1 CXL Port/Device CXL Modified TS Flex Bus Modified TS Info DVSEC RCRB RCH/RCD DVSEC RCRB
- Register Locator DVSEC CXL Entry

- MLD DVSEC FM LD MLD Capability LD LD-ID
- PCIe DVSEC for Test Capability CXL Compliance CXL Compliance DVSEC DVSEC

9 CXL DVSEC DVSEC ID

CXL Capability (Name, DVSEC ID) \ Devices		PCIe	RCH DP	CXL RP	RCD	RCD UP	LD	SLD	FMLD	USP[1]	DSP[2]
PCIe DVSEC for CXL Devices	0x0000	F	F	F	M	F	M	M	M	F	F
Non-CXL Function Map DVSEC	0x0002	F	F	F	O	F	O	O	O	O	F
CXL Extensions DVSEC for Ports	0x0003	F	F	M	F	F	F	F	F	M	M
GPF DVSEC for CXL Ports	0x0004	F	F	M	F	F	F	F	F	F	M
GPF DVSEC for CXL Devices	0x0005	F	F	F	O	F	M	M	F	F	F
PCIe DVSEC for Flex Bus Port	0x0007	F	M	M	M	M	M	M	M	M	M
Register Locator DVSEC	0x0008	F	O	M	O	O	M	M	M	M	M
MLD DVSEC	0x0009	F	F	F	F	F	F	F	M	F	F
PCIe DVSEC for Test Capability	0x000a	F	F	F	M	F	F	O	F	F	F

[1]USP: Upstream Switch Port
[2]DSP: Downstream Switch Port

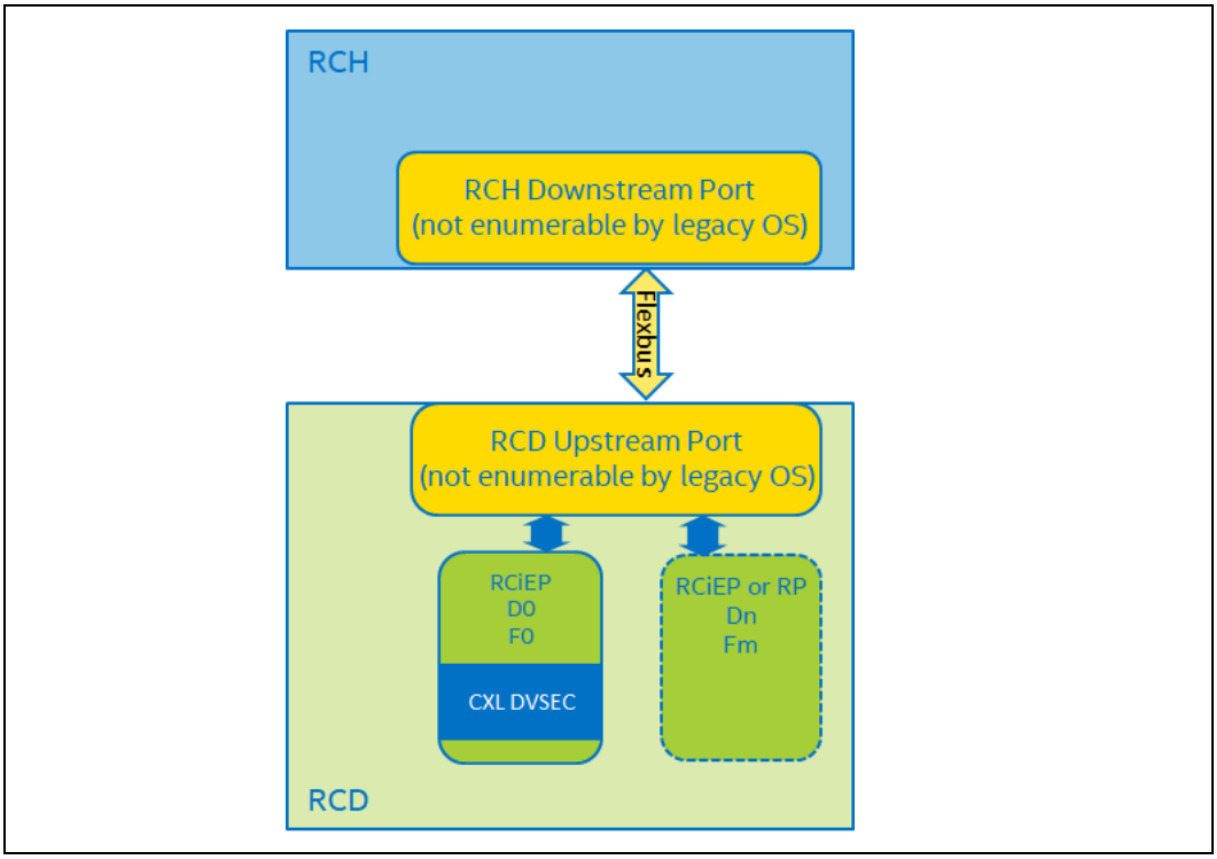
Mandatory Forbidden Optional

CXL Capability	DVSECID	HighestDVSECRevisionID	Mandatory	Not Permitted	Optional
PCIe DVSEC for CXL Device (Section 8.1.3)	0	1	D1, D2, LD, FMLD	P, UP1, DP1, R, USP, DSP	
Non-CXL Function Map DVSEC(Section 8.1.4)	2	0	P, UP1, DP1, R, DSP	D1. D2, LD, FMLD, USP	
CXL 2.0 Extensions DVSEC for Ports (Section 8.1.5)	3	0	R, USP, DSP	P, D1, D2, LD, FMLD, UP1, DP1	
GPF DVSEC for CXL Ports (Section 8.1.6)	4	0	R, DSP	P, D1, D2, LD, FMLD, UP1, DP1, USP	
GPF DVSEC for CXL Devices (Section 8.1.7)	5	0	D2, LD	P, UP1, DP1, R, USP, DSP, FMLD	D1
PCIe DVSEC for Flex Bus Port (Section 8.1.8)	7	1	D1, D2, LD, FMLD, UP1, DP1, R, USP, DSP	P	
Register Locator DVSEC(Section 8.1.9)	8	0	D2, LD, FMLD, R, USP, DSP	P	D1, UP1, DP1
MLD DVSEC (Section 8.1.10)	9	0	FMLD	P, D1, D2, LD,UP1, DP1, R, USP, DSP	
PCIe DVSEC for Test Capability (Section 14.16.1)	0Ah	0	D1	P, LD, FMLD, DP1, UP1, R, USP, DSP	D2

DVSEC

PCIe DVSEC for CXL Device ID=0	CXL 2.0 Extensions DVSEC for Ports ID=3
DVSEC CXL Capability (Offset 0Ah)	CXL Port Extension Status (Offset 0Ah)
DVSEC CXL Control (Offset 0Ch)	Port Control Extensions (Offset 0Ch)
DVSEC CXL Status (Offset 0Eh)	Alternate Bus Base (Offset 0Eh)
DVSEC CXL Control2 (Offset 10h)	Alternate Bus Limit (Offset 0Fh)
DVSEC CXL Status2 (Offset 12h)	Alternate Memory Base (Offset 10h)
DVSEC CXL Lock (Offset 14h)	Alternate Memory Limit (Offset 12h)
DVSEC CXL Capability2 (Offset 16h)	Alternate Prefetchable Memory Base (Offset 14h)
DVSEC CXL Range 1 Size High (Offset 18h)	Alternate Prefetchable Memory Limit (Offset 16h)
DVSEC CXL Range1 Size Low (Offset 1Ch)	Alternate Memory Prefetchable Base High (Offset 18h)
DVSEC CXL Range 1 Base High (Offset 20h)	Alternate Prefetchable Memory Limit High (Offset 1Ch)
DVSEC CXL Range 1 Base Low (Offset 24h)	CXL RCRB Base (Offset 20h)
	CXL RCRB Base High (Offset 24h)
GPF DVSEC for CXL Port ID=4	**GPF DVSEC for CXL Device ID=5**
GPF Phase 1 Control (Offset 0Ch)	GPF Phase 2 Duration (Offset 0Ah)
GPF Phase 2 Control (Offset 0Eh)	GPF Phase 2 Power (Offset 0Ch)
PCIe DVSEC for Flex Bus Port ID=7	**Register Locator DVSEC ID=8**
DVSEC Flex Bus Port Capability (Offset 0Ah)	Register Offset Low (Block 1)
DVSEC Flex Bus Port Control (Offset 0Ch)	Register Offset High (Block 1)
DVSEC Flex Bus Port Status (Offset 0Eh)	Register Offset Low (Block 2)
DVSEC Flex Bus Port Received Modified TS Data Phase1 (Offset 10h)	Register Offset High (Block 2)
	Register Offset Low (Block 3)
	Register Offset High (Block 3)
MLD DVSEC ID=9	**CXL Device Test Capability Advertisement DVSEC ID=A**
Number of LD Supported (Offset 0Ah)	DVSEC CXL Test Lock (offset 0Ah)
LD-ID Hot Reset Vector (Offset 0Ch)	DVSEC CXL Test Capability1 (offset 0Ch)
	Device CXL Test Capability2 (Offset 10h)
	DVSEC CXL Test Configuration Base Low (Offset 14h)
	DVSEC CXL Test Configuration Base High (Offset 18h)

PCIe Software View of an RCH and RCD



Because the CXL link is not exposed to CXL-unaware OSs, the System Firmware view of the hierarchy is different than that of the CXL-unaware OS.

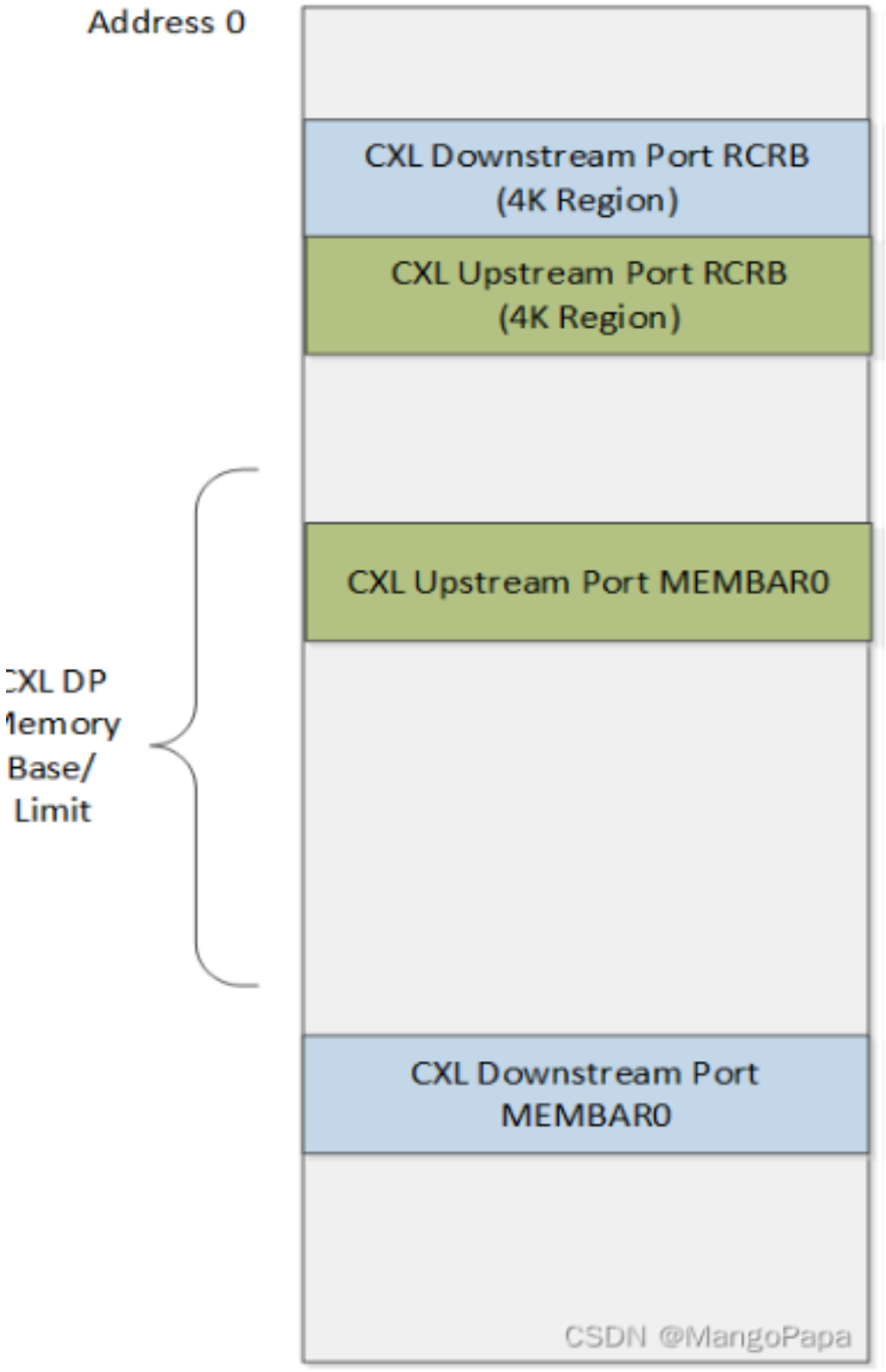
Reg in MMIO

9 PCIe CXL DVSEC CXL PCIe PCIe 4KB CXL Component Register Size CXL DVSEC Component Register Memory Mapped MMIO Memory

CXL 3.0 CXL 6 MMIO

- **RCH DP RCRB** RCH 4KB MEMBAR0 ACPI
- **RCH DP Component Register** CXL ARB/MUX 64KB RCH DP RCRB 64b MEMBAR0 RCH
- **RCD UP RCRB** RCD 4KB RCH DP RCRB 4KB RCH DP Memory MEMBAR0 RCD UP RCRB RCD RCH DP MRd CXL 1.1 MRd/MWr
- **RCD UP Component Register** CXL ARB/MUX 64KB RCRB RCD RCD UP RCRB 64b MEMBAR0 RCD RCH DP Memory MEMBAR0 RCRB RCD Register Locator DVSEC
- **Component Register for ALL Other CXL Components** CXL Component Register Locator DVSEC
- **CXL Host Bridge Component Register (CHBCR)** CXL Host Bridge RP Memory Interleaving ACPI CEDT

RCRB Components Register CHBCR Register Locator DVSEC Component Registers for All Other CXL Components



PCIe CXL blog

https://blog.csdn.net/weixin_40357487/article/details/132553156

6.3 CXL in CMN

[toc]

CXS

ARM RD_N2 chip to chip CCIX2.0 CMN-700 TRM ARM CML_SMP

S CXL CXS CCIX2.0 CML_SMP CCIX2.0

CMN CCG chip CXL

Coherent Multichip Link (CML) device = CCG

A given multi-chip link can be used for: • SMP (CML_SMP) connection • CXL device attachment

A CML device (CCG) can be configured to be used for **CML_SMP** connection or **CXL device attachment**.

For SMP systems, CCG block is required to enable multi-chip SMP communications over a **CXS issue B interface**.

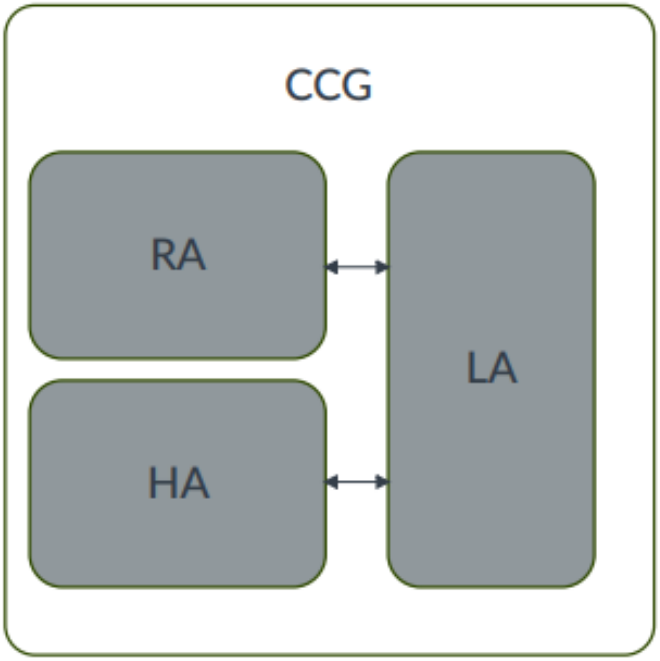
CXL **CXS issue A interface**

CCG

CML components

The following figure shows a CCG block diagram with RA, HA, and LA.

Figure 3-17: CCG block diagram with RA, HA, and LA

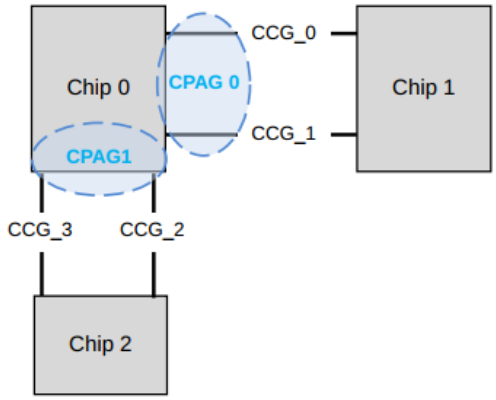


3.1.3.2 CML Port Aggregation Groups

The CMN-700 CML configuration supports up to 32 gateway blocks (CCGs). These CML gateways can be grouped in up to 16 *CML Port Aggregation Groups* (CPAGs).

This feature can be used when connecting two or more chips together with multiple ports between the chips. For example, the following figure shows three chips that are connected by four CCGs that are grouped into two CPAGs.

Figure 3-19: CML Port Aggregation Groups



CCG

The **CML RA** node type has the following MPAM modes:

- SMP mode The CML RA passes the MPAM field on the USER field of the request. When snooped, the CML RA receives the MPAM field and passes it through the CHI SNP MPAM field.
- Non-SMP mode The CML RA drops the MPAM field that is received on the CHI request. The CML RA also does not receive MPAM field on CML snoops in this mode.
- CXSA mode The CML RA passes the MPAM field on the USER field of the request, even though CXSA is in non-SMP mode. You can use a configuration bit to enable passing of MPAM attributes when in CXSA mode.

The **CML HA** node type has the following MPAM modes:

- SMP mode The CML HA receives MPAM fields through the USER field of the request and passes them through the CHI MPAM field. On incoming CHI transactions, the CML HA passes CHI MPAM values through on the USER field.
- Non-SMP mode The CML HA drops the MPAM values that it receives on CHI snoop. The CML HA does not receive the MPAM attributes on CML requests.

CXL enable **CXSA** mode in RA

LA enable CXL

CCG SAM

CMN CCG CXL CMN700 type3 CXL type3 memory CMN SAM CCG

6.4 CXL in SCP and EDK2

SCP CXL

SCP EDK2

CCG

- SHA-1: bc14e780c120fc656ce905d6e219078a0006da88
  - module/cmn700: configuring CCG for mapping Host address to CXL mem A Memory region is reserved for CXL Memory. CXL.Mem (0x3fe_0000_0000) comes under 4TB Chip-0 memory and the whole region is by default configured as SCG. Configured CXL.Mem region in HNF-SAM HTG and CCG SA node IDs for HTGs in following order -

  HNF_SAM_CCG_SA_NODEID_REG HNF_SAM_HTG_CFG3_MEMREGION HNF_SAM_HTG_CFG2_MEM HNF_SAM_HTG_CFG1_MEMREGION

  CXL Memory region is accessible as Normal memory with above configuration.

  This patch maps Host address space to CXL device mem area through CCG node, based on the CXL device memory size, which is discovered by CXL module. CXL module invokes runtime CMN700 API for mapping the host address space and configuring CCG node.

  This patch also adds a flag "cxl_mem" in CCG_Config structure for identifying host region and configuration reserved for CXL device memory purpose and thus differentiating from Remote chip memory.

  Signed-off-by: Sayanta Pattanayak sayanta.pattanayak@arm.com Change-Id: I988f471db6a6a55f97320519413daedd8ea

config_cmn700.c

N2 0-0x40000000000 SCG

HNSAM 0-0x3fe00000000 CCG-cxl_mem

```
#if (PLATFORM_VARIANT == 1)
static const struct mod_cmn700_ccg_config ccg_config_table_chip_0[] = {
    {
        .remote_rnf_count = 0,
        .remote_mmap_table = {
            .base = UINT64_C(0x3fe00000000),
            .size = UINT64_C(8) * FWK_GIB,
            .type = MOD_CMN700_REGION_TYPE_CCG,
        },
        .ra_mmap_table = {
            {
                .base = UINT64_C(0x3fe00000000),
                .size = UINT64_C(8) * FWK_GIB,
                .remote_haid = 0,
            },
            { 0 }
        },
        .remote_agentid_to_linkid_map = {
            {
                .remote_agentid_start = (RNF_PER_CHIP * CHIP_0),
                .remote_agentid_end = (RNF_PER_CHIP * CHIP_0)
```

```
            },
        },
        .smp_mode = false,
        .cxl_mem  = true,
    },
};
#endif
```

entry 8G flag cxl_mem CCG

```
static int cmn700_setup_rnsam_ccg_regions(void)
{
    const struct mod_cmn700_config *config;
    const struct mod_cmn700_mem_region_map *region;
    struct cmn700_rnsam_reg *rnsam;
    unsigned int count;
    unsigned int cxra_ldid;
    unsigned int cxra_node_id;
    unsigned int idx;
    uint32_t bit_pos;
    uint32_t group;

    config = ctx->config;
    /* Do configuration for CCG Nodes */
    for (idx = 0; idx < config->ccg_table_count; idx++) {
        region = &config->ccg_config_table[idx].remote_mmap_table;
        if (region->type != MOD_CMN700_REGION_TYPE_CCG) {
            return FWK_E_DATA;
        }

        FWK_LOG_INFO(
            MOD_NAME "  [0x%llx - 0x%llx] %s",
            region->base,
            region->base + region->size - 1,
            mmap_type_name[region->type]);

        /* If the region is for extended memory area like CXL.Mem
         * and connected through CCG then the region shouldn't be
         * marked as Non-Hash region. CXL.Mem region should be part
         * of Hashed cache group area.
         */
        if (config->ccg_config_table[idx].cxl_mem == true)
            continue;

        for (count = 0; count < ctx->internal_rnsam_count; count++) {
            rnsam = ctx->internal_rnsam_table[count];
```

CCG CXL.Mem non-hash hash

```
static int map_ccg_for_cxl_mem(uint64_t size)
{
```

```
    uint32_t idx;
    const struct mod_cmn700_config *config = ctx->config;
    const struct mod_cmn700_ccg_config *ccg_config;

    cmn700_rnsam_stall();

    /* Do configuration of CCG Node for mapping remote CXL Mem area. */
    for (idx = 0; idx < config->ccg_table_count; idx++) {
        ccg_config = &(config->ccg_config_table[idx]);
        if (ccg_config->cxl_mem == true)
            ccg_setup_for_remote_mem(size, ctx, ccg_config);
    }

    cmn700_rnsam_unstall();

    return FWK_SUCCESS;
}
```

API CCG CXL.Mem

rnsam stall unstall rnsam default node

STATUS 0x2 unstall enable RNSAM

online RNSAM stall

Figure 4-647: rnsam_status

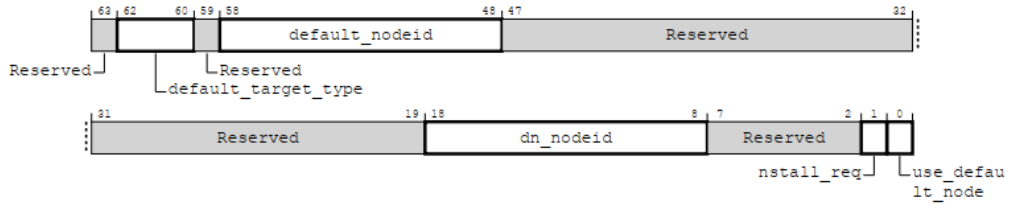


Table 4-663: rnsam_status attributes

Bits	Name	Description	Type	Reset
[63]	Reserved	Reserved	RO	-
[62:60]	default_target_type	Indicates node type **3'b000** HN-F **3'b001** HN-I **3'b010** CXRA **3'b011** HN-P **3'b100** PCI-CXRA **3'b101** HN-S **Others** Reserved CONSTRAINT: Only applicable for RN-I	RW	3'b001
[59]	Reserved	Reserved	RO	-
[58:48]	default_nodeid	Default Node ID	RW	Configuration dependent
[47:19]	Reserved	Reserved	RO	-
[18:8]	dn_nodeid	DN Node ID for DN operations	RW	11'b0
[7:2]	Reserved	Reserved	RO	-
[1]	nstall_req	Indicates RN SAM is programmed and ready **1'b0** STALL requests **1'b1** UNSTALL requests	RW	1'b0
[0]	use_default_node	Indicates target ID selection mode **1'b0** Enables RN SAM to hash address bits and generate target ID **1'b1** Uses default target ID	RW	1'b1

```
static void cmn700_rnsam_stall(void)
{
    struct cmn700_rnsam_reg *rnsam;
    uint32_t rnsam_idx;

    for (rnsam_idx = 0; rnsam_idx < ctx->internal_rnsam_count; rnsam_idx++) {
        rnsam = ctx->internal_rnsam_table[rnsam_idx];
        rnsam->STATUS = (rnsam->STATUS & ~CMN700_RNSAM_STATUS_UNSTALL) |
            CMN700_RNSAM_STATUS_USE_DEFAULT_TARGET_ID;
    }
    __DSB();
}

static void cmn700_rnsam_unstall(void)
{
    struct cmn700_rnsam_reg *rnsam;
    uint32_t rnsam_idx;

    for (rnsam_idx = 0; rnsam_idx < ctx->internal_rnsam_count; rnsam_idx++) {
        rnsam = ctx->internal_rnsam_table[rnsam_idx];
        rnsam->STATUS = (rnsam->STATUS | CMN700_RNSAM_STATUS_UNSTALL) &
            ~(CMN700_RNSAM_STATUS_USE_DEFAULT_TARGET_ID);
    }
    __DSB();
}
```

ccg_setup_for_remote_mem

```
int ccg_setup_for_remote_mem(
    uint64_t size,
    struct cmn700_device_ctx *ctx,
    const struct mod_cmn700_ccg_config *ccg_config)
{
    uint64_t reg_val;
    unsigned int index;
    unsigned int ccg_ldid;
    struct cmn700_hnf_reg *hnf_reg;
    struct cmn700_ccg_ra_reg *ccg_ra_reg;

    cmn700_ccg_ctx.is_prog_for_port_agg = false;
    /* Enable CXSA */
    ccg_ldid = get_ldid(ctx, cmn700_ccg_ctx.is_prog_for_port_agg);
    ccg_ra_reg = ctx->ccg_ra_reg_table[ccg_ldid].ccg_ra_reg;

    for (index = 0; index < ctx->hnf_count; index++) {
        /* Programming sequence to enable CXL.mem regions inside HNSAM */

        hnf_reg = (struct cmn700_hnf_reg *)ctx->hnf_node[index];

        /* Configuring CCG SA node IDs for HTGs in the HNSAM */
        reg_val = hnf_reg->HNF_SAM_CCG_SA_NODEID_REG[0];
        reg_val &= ~(CMN700_HNF_SAM_CCG_SA_NODEID_MASK);
        reg_val |= get_node_id(ccg_ra_reg);
```

```
        hnf_reg->HNF_SAM_CCG_SA_NODEID_REG[0] = reg_val;

        /* CXSA/CXLSA aggregated SA selection function */
        /*    region   SN NODE 8SN node,   CXSA   */
        hnf_reg->HNF_SAM_HTG_CFG3_MEMREGION[0] |=
            (CMN700_HNF_SAM_HTG_MODE_CXSA <<
            CMN700_HNF_SAM_HTG_SN_MODE_POS);

        /* 64B interleaved */
        /* CXSA    interleaved            0 64B */
        reg_val = hnf_reg->HNF_SAM_HTG_CFG3_MEMREGION[0];
        reg_val &= ~(CMN700_HNF_SAM_HTG_SA_DEVICE_INTERLEAVE_CNTL_MASK <<
               CMN700_HNF_SAM_HTG_SA_DEVICE_INTERLEAVE_CNTL_POS);
        hnf_reg->HNF_SAM_HTG_CFG3_MEMREGION[0] |= reg_val;

        /* 1 CXSA/CXLSA port used */
        /*          0 */
        reg_val = hnf_reg->HNF_SAM_HTG_CFG3_MEMREGION[0];
        reg_val &= ~(CMN700_HNF_SAM_HTG_SA_PORTS_CNT_MASK <<
               CMN700_HNF_SAM_HTG_SA_PORTS_CNT_POS);
        hnf_reg->HNF_SAM_HTG_CFG3_MEMREGION[0] |= reg_val;

        /* htg_region_end_addr[51:20] = end address of HTG region */
        reg_val = ccg_config->ra_mmap_table[0].base +
              (size - 1);
        reg_val &= ~(CMN700_HNF_SAM_HTG_REGION_ADDR_RES_MASK);
        hnf_reg->HNF_SAM_HTG_CFG2_MEMREGION[0] |= reg_val;

        /* htg_region_base_addr[51:20] = start address of HTG region */
        reg_val = ccg_config->ra_mmap_table[0].base;
        reg_val &= ~(CMN700_HNF_SAM_HTG_REGION_ADDR_RES_MASK);
        hnf_reg->HNF_SAM_HTG_CFG1_MEMREGION[0] |= reg_val;

        /* Configuring HTG region as Valid */
        hnf_reg->HNF_SAM_HTG_CFG1_MEMREGION[0] |=
            (UINT64_C(0x1) << CMN700_HNF_SAM_HTG_REGION_VALID_POS);
    }

    /*
     * Program the CXRA SAM with the address range and the corresponding
     * remote HAID.
     */
    program_ccg_ra_sam_addr_region(ctx, ccg_config);

    update_cxl_mem_region(ccg_config->ra_mmap_table[0].base,
                          ccg_config->ra_mmap_table[0].size);

    return FWK_SUCCESS;
}
```

link

cxl cxsa enable

- RA SAM region:
- RA SAM valid valid

```
ccg_ra_reg->CCG_RA_SAM_ADDR_REGION_REG[0] |=
            fwk_math_log2(size) |
            (cxl_cfg->base & CCG_RA_REMOTE_SAM_BASE_ADDR_MASK) |
            ((uint64_t)0 << CCG_RA_REMOTE_SAM_TGT_HAID_POS);

ccg_ra_reg->CCG_RA_SAM_ADDR_REGION_REG[0] |= CCG_RA_REMOTE_SAM_VALID_MASK;
```

[5]	cxl_cache_en	Enable CXL .cache mode, by default is disabled	RW	1'b0
[4]	cxl_mem_en	Enable CXL .mem mode, by default is enabled	RW	1'b1
[3:2]	cxl_type	Used to program CXL Type for RA **2'b00** Reserved **2'b01** Type1 **2'b10** Type2 **2'b11** Type3	RW	2'b11
[1]	la_device_mode_en	Enable the Device mode, by default set to Host mode	RW	1'b0
[0]	la_cxl_mode_en	When set enables CXL mode	RW	1'b0

- CCLA flex bus control MEM CACHE
- CCLA config control MEM CACHE

```
reg_val1 = ccla_reg->CCLA_DVSEC_FLEX_BUS_PORT_CONTROL;
reg_val2 = ccla_reg->CCLA_CFG_CTL;

if (cxl_cfg->is_cache_en) {
    reg_val1 |= CCLA_CFG_CTL_CXL_CACHE_EN_MASK;
    reg_val2 |= CCLA_DVSEC_FLEX_BUS_PORT_CTRL_CACHE_EN_MASK;
}
if (cxl_cfg->is_mem_en) {
    reg_val1 |= CCLA_CFG_CTL_CXL_MEM_EN_MASK;
    reg_val2 |= CCLA_DVSEC_FLEX_BUS_PORT_CTRL_MEM_EN_MASK;
}
if (cxl_cfg->is_device_mode) {
    reg_val1 |= CCLA_CFG_CTL_LA_DEVICE_MODE_EN_MASK;
}
```

```
if (cxl_cfg->is_multi_logical_dev) {
    reg_val2 |= CCLA_DVSEC_FLEX_BUS_PORT_CTRL_MULTI_LOGICAL_DEVICE_EN_MASK;
}
```

```
reg_val1 |= (cxl_cfg->cxl_type << CCLA_CFG_CTL_CXL_TYPE_POS);
reg_val1 |= CCLA_CFG_CTL_LA_CXL_MODE_EN_MASK;
```

```
ccg_ra_reg->CCG_RA_CFG_CTRL |= 0x200;
```

[9]	cxsa_mode_en	When set, enables the CCIX Subordinate Agent mode.In this mode RA functions as a CCIX Subordinate Agent **1'b1** CCIX Subordinate Agent **1'b0** CCIX Requesting Agent	RW	1'b0

link

```
ccg_ra->LINK_REGS[0].CCG_CCPRTCL_LINK_CTRL |= CCPRTCL_LINK_CTRL_LINK_EN_MASK;
ccg_ha->LINK_REGS[0].CCG_CCPRTCL_LINK_CTRL |= CCPRTCL_LINK_CTRL_LINK_EN_MASK;
```

		agent may still be in LINK_UP state.		
[0]	lnk0_link_en	Enables CCIX Link 0 when set **1'b0** Link is disabled **1'b1** Link is enabled	RW	1'b0

```
int exchange_protocol_credit
```

[7:4]	lnk0_num_snpcrds	Controls the number of CCIX snoop credits assigned to Link 0 **4'h0** Total credits are equally divided across all links **4'h1** 25% of credits assigned **4'h2** 50% of credits assigned **4'h3** 75% of credits assigned **4'h4** 100% of credits assigned **4'hF** 0% of credits assigned	RW	4'b0

6.5 CXL

[toc]

CXL

CXL PCIe

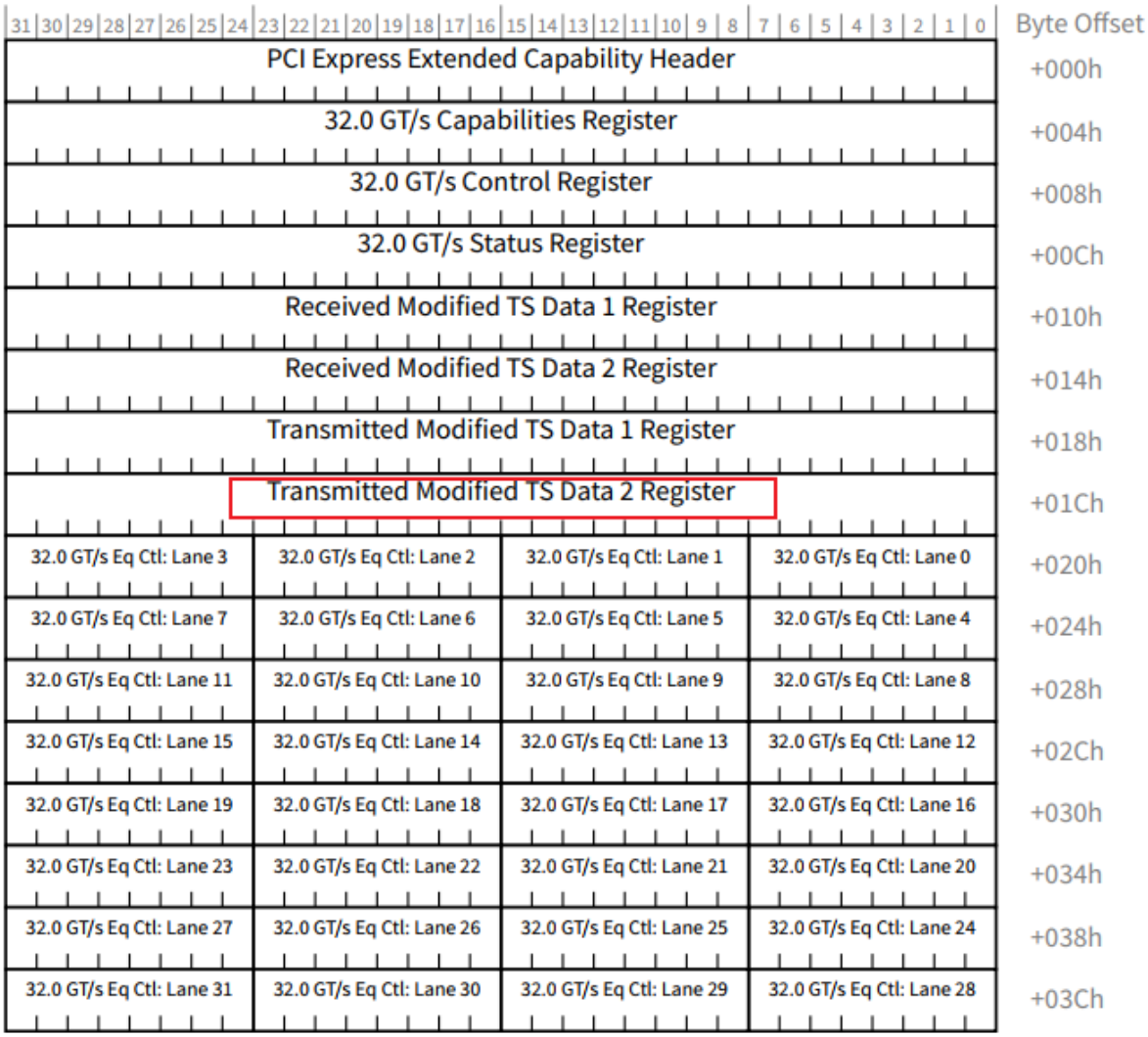


Figure 7-87 Physical Layer 32.0 GT/s Extended Capability

Table 7-77 Transmitted Modified TS Data 2 Register

Bit Location	Description	Attributes
23:0	***Transmitted Modified TS Information 2*** - If Modified TS Received is Set, this field contains the Modified TS Information 2 field from the Modified TS2 Ordered Set transmitted during the Configuration.Complete LTSSM State (see Section 4.2.6.3.6). Bits 23:16 contain the value of Symbol 14. Bits 16:8 contain the value of Symbol 13. Bits 7:0 contain the value of Symbol 12. If PCI Express (Usage Mode 0) is the only one supported, this field is permitted to be hardwired to 00 0000h. Default is 00 0000h.	RO
25:24	***Alternate Protocol Negotiation Status*** - Indicates the status of the Alternate Protocol Negotiation. Encodings are: **00b** Alternate Protocol Negotiation not supported - Modified TS Usage Mode 2 Supported - Alternate Protocol is Clear. **01b** Alternate Protocol Negotiation disabled - Modified TS Usage Mode 2 Supported - Alternate Protocol is Set but Modified TS Usage Mode Selected was not 2 during the appropriate LTSSM State. **10b** Alternate Protocol Negotiation failed - Alternate Protocol Negotiation was attempted and did not locate a protocol that was supported on both ends of the Link. **11b** Alternate rotocol Negotiation succeeded - Alternate Protocol Negotiation located one or more protocols that were supported on both ends of the Link and the Downstream Port selected one of those protocols for use. If Set, Alternate Protocol Negotiation completed succesfully. If Clear, Alternate Protocol Negotiation negotiation has not completed succesfully. If Modified TS Usage Mode 1 Supported - Training Set Message and Modified TS Usage Mode 2 Supported - Alternate Protocol are both Clear, this register is permitted to be hardwired to 0000 0000h. If Modified TS Usage Mode 2 Supported - Alternate Protocol is Clear, this bit is hardwired to 0b. If Modified TS Usage Mode Selected does not equal 2, this bit contains 0b. This bit is Cleared on Detect LTSSM State. Default is 0b.	RO

PCIe DVSEC for CXL Devices CXL PCIe DVSEC RCD LD SLD FMLD CXL Device D0F0 DVSEC Device DVSEC Device PCIe Device CXL Device

CXL2.0

PCIe

CXL1.1

RCiEP

HDM Decoder

6.6 DDR Introduction

[toc]

Introduction

SDRAM vs. SRAM

- Synchronous Dynamic Random Access Memory

- Static RAM

Type of RAM	Cost per cell	Size of each cell	Power Dissipated	Speed
Dynamic RAM	Low	Very small	High	Slow
Static RAM	High	Large	Low	High

SRAM SDRAM

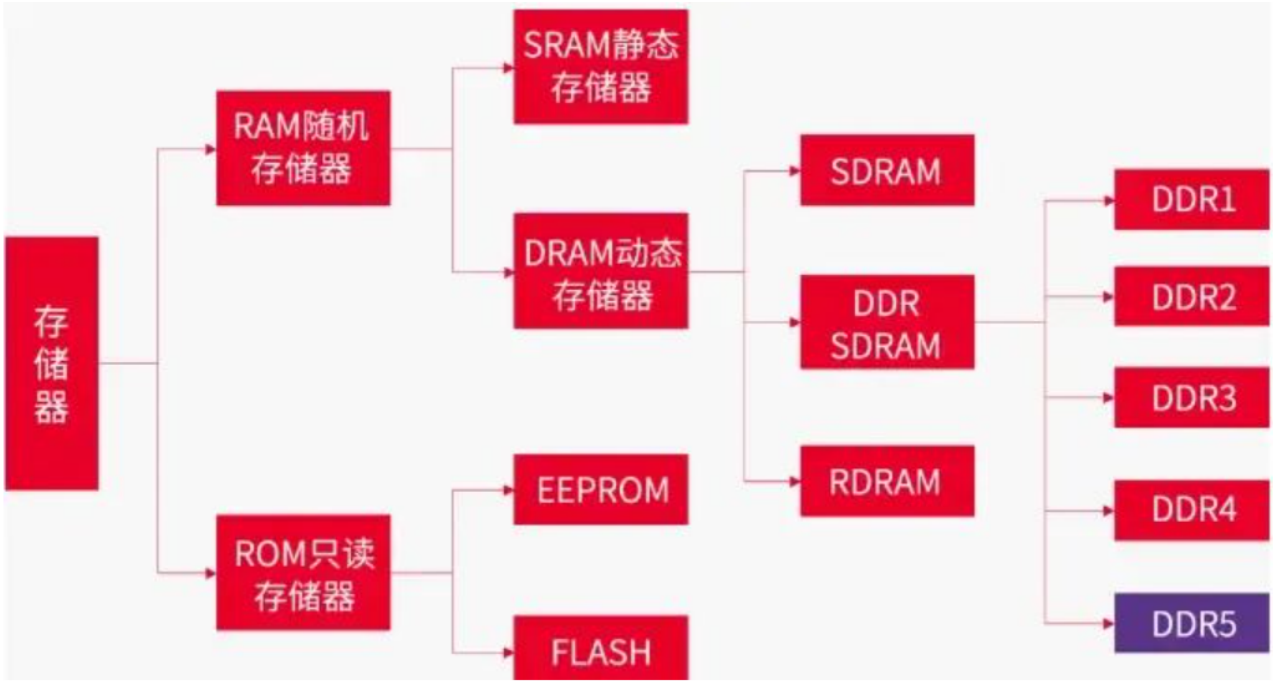
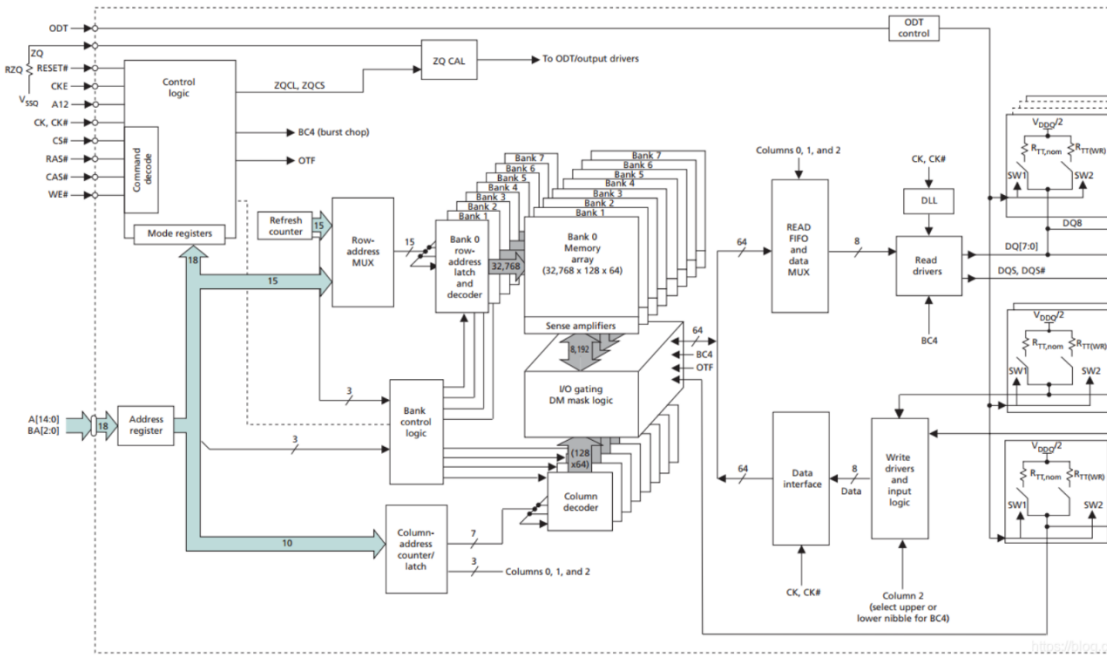


Figure 4: 256 Meg x 8 Functional Block Diagram



DDR3 SDRAM

- 18b

-
- 8 bank -> BA[2:0], 3 bit 8
- A 15b
- 10b;
- 8192

Row Address 16 A0-A15

Column Address 10 A0-A9

Bank Bank Address 3 BA0-BA2

4 8

512Mb, 1Gb, 2Gb, 4Gb, 8Gb

1Gb

Configuration	256Mb x 4	128Mb x 8	64Mb x 16
# of Banks	8	8	8
Bank Address	BA0 - BA2	BA0 - BA2	BA0 - BA2
Auto precharge	A10/AP	A10/AP	A10/AP
BC switch on the fly	A12/BC#	A12/BC#	A12/BC#
Row Address	A0 - A13	A0 - A13	A0 - A12
Column Address	A0 - A9,A11	A0 - A9	A0 - A9
Page size [1]	1 KB	1 KB	2 KB

2Gb

Configuration	512Mb x 4	256Mb x 8	128Mb x 16
# of Banks	8	8	8
Bank Address	BA0 - BA2	BA0 - BA2	BA0 - BA2
Auto precharge	A10/AP	A10/AP	A10/AP
BC switch on the fly	A12/BC#	A12/BC#	A12/BC#
Row Address	A0 - A14	A0 - A14	A0 - A13
Column Address	A0 - A9,A11	A0 - A9	A0 - A9
Page size [1]	1 KB	1 KB	2 KB

256Mb Configuration

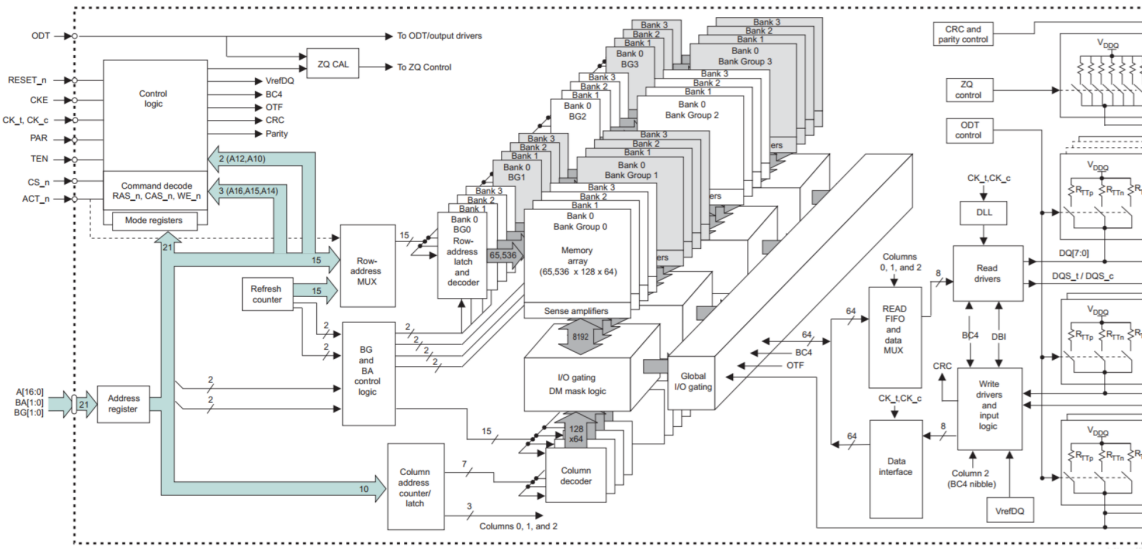
	256Mb x 4	128Mb x 8	64Mb x 16
# of Banks	8	8	8
Bank Address	BA0 - BA2	BA0 - BA2	BA0 - BA2
Auto precharge	A10/AP	A10/AP	A10/AP
BC switch on the fly	A12/BC#	A12/BC#	A12/BC#
Row Address	A0 - A13	A0 - A13	A0 - A12
Column Address	A0 - A9,A11	A0 - A9	A0 - A9
Page size	1 KB	1 KB	2 KB

2Gb Configuration

	512Mb x 4	256Mb x 8	128Mb x 16
# of Banks	8	8	8
Bank Address	BA0 - BA2	BA0 - BA2	BA0 - BA2
Auto precharge	A10/AP	A10/AP	A10/AP

	512Mb x 4	256Mb x 8	128Mb x 16
BC switch on the fly	A12/BC#	A12/BC#	A12/BC#
Row Address	A0 - A14	A0 - A14	A0 - A13
Column Address	A0 - A9,A11	A0 - A9	A0 - A9
Page size	1 KB	1 KB	2 KB

Figure 3: 512 Meg x 8 Functional Block Diagram



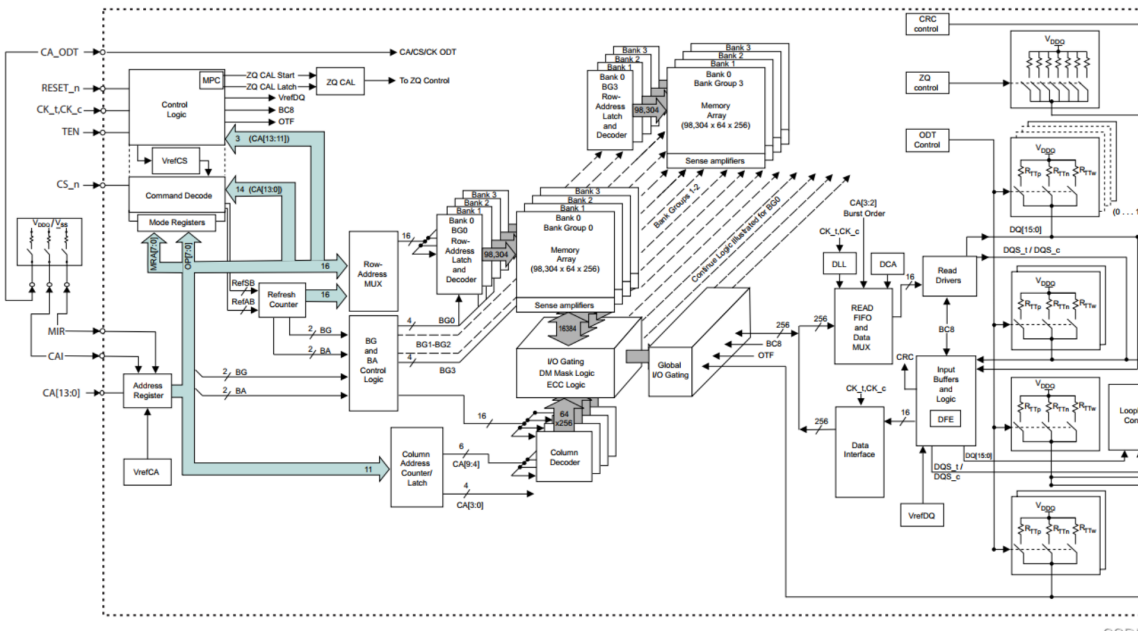
DDR4 SDRAM

Row Address 16 A0-A15

Column Address 10 A0-A9

Bank Bank Address 2 BA0-BA1

Bank Bank Group Address 2 BG0-BG1



DDR5 SDRAM

Row Address 16 A0-A15

Column Address 10 A0-A9

Bank Bank Group Address 2 BA0-BA1

Bank Bank Address 3 BG0-BG2

Command Address CA

1. Channel
2. Module Channel
3. Rank DDR SDRAM Module Rank Rank Rank CS_n SDRAM
4. Chip DDR SDRAM 64 Rank 8 x8 Chip
5. Bank Group DDR4 Bank Group DDR4 Bank Group tCCD_L
6. Bank Bank Row Bank Activate/Precharge
7. Row Activate/Precharge
8. Column Column n Cell n SDRAM
9. Cell Cell 1 bit

Page

- **Row Column** DDR
- **Page**
- 1024 1024

2K page 1024 x16 2 B 2KB

8N

32 4 8 8N

32

DQ DQS DM

Figure: Point-to-Point (DQ/DQS) and Fly-by (CAC) Routing

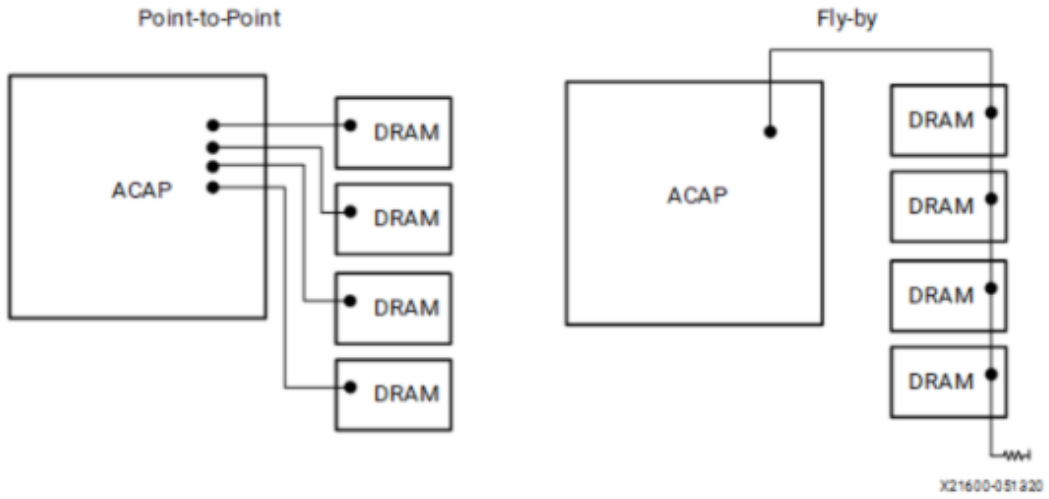


Figure 9: SDRAM 的两种信号连接方式（图源 Versal ACAP PCB Design User Guide (UG863)）

fly-by DRAM

CLK/CLK# DQS AD CMD

DQ

DQ 8 kua DQ

Write Leveling

Fly-by Topology

SDRAM DQS CK DQ DQS

DQS Data Strobe CK Clock

DQS

CK

DQ DQ

DQS DQS-DQS# CK DQ DDR3 01010101 DQS-DQS# DQ 0 1 Write leveling

1. SDRAM Write Leveling SDRAM DQS CK DQ
2. DQS DQS DQ 0-1 00111111111111111110000 0 1 0
3. DQS DQ 0 1 DQS CK
4. SDRAM Write Leveling

Read Leveling

SDRAM

1. SDRAM Pattern
2. 0
3.
4. 1 0
5. 1

0-1 Write Leveling 0-1 0-1 Read Leveling
Read Centering

Spec

SDRAM JEDEC

- JESD79F: DDR SDRAM
- JESD79-2F: DDR2 SDRAM
- JESD79-3F: DDR3 SDRAM
- [JESD79-4D: DDR4 SDRAM](https://www.jedec.org/document_search?search_api_views_fulltext=jesd79-4 ddr4)
- JESD79-5B: DDR5 SDRAM

DDR LPDDR

- JESD209B: LPDDR SDRAM
- JESD209-2F: LPDDR2 SDRAM
- JESD209-3C: LPDDR3 SDRAM
- JESD209-4D: LPDDR4 SDRAM
- JESD209-5B: LPDDR5 SDRAM

HBM SDRAM

- JESD235D: HBM
- JESD238A: HBM3

GDDR SGRAM

- SDRAM3.11.5.8 R16.01: GDDR4 SGRAM
- JESD212C.01: GDDR5 SGRAM
- JESD232A.01: GDDR5X SGRAM
- JESD250D: GDDR6 SGRAM

https://www.cnblogs.com/sky-heaven/p/15948268.html

Blog

https://zhuanlan.zhihu.com/p/26327347

Part III

Chapter 7

LeetCode

(Bubble Sort)	O(n)	O(n^2)	O(n^2)	O(1)
(Selection Sort)	O(n^2)	O(n^2)	O(n^2)	O(1)
(Insertion Sort)	O(n)	O(n^2)	O(n^2)	O(1)
(Quick Sort)	O(n log n)	O(n log n)	O(n^2)	O(log n)
(Merge Sort)	O(n log n)	O(n log n)	O(n log n)	O(n)
(Heap Sort)	O(n log n)	O(n log n)	O(n log n)	O(1)

7.1 C basic

[toc]

C

static

extern

C

C

C

C

2. 头文件中的全局变量声明

正确的做法是在头文件中声明全局变量，并在一个源文件中定义它。使用 `extern` 关键字声明全局变量，而不是定义它。

示例：

c 复制代码

```c
// global.h
extern int globalVar;  // 声明全局变量
```

c 复制代码

```c
// file1.c
#include "global.h"

int globalVar = 42;  // 定义全局变量
```

c 复制代码

```c
// file2.c
#include "global.h"

void someFunction() {
    globalVar = 100;  // 使用全局变量
}
```

这样，`globalVar` 只在 `file1.c` 中定义一次，但可以在其他任何包含 `global.h` 的源文件中使用。

C

```c
#include <stdio.h>

int check_endianness() {
    union {
        unsigned int i;
        unsigned char c[4];
    } test_union;

    test_union.i = 0x01020304;

    if (test_union.c[0] == 1) {
        return 1; //
    } else {
        return 0; //
```

```
    }
}

int main() {
    if (check_endianness()) {
        printf("  (Big-endian)\n");
    } else {
        printf("  (Little-endian)\n");
    }
    return 0;
}
```

test_union.i 0x01020304 32 4

Big-endian 01 02 03 04

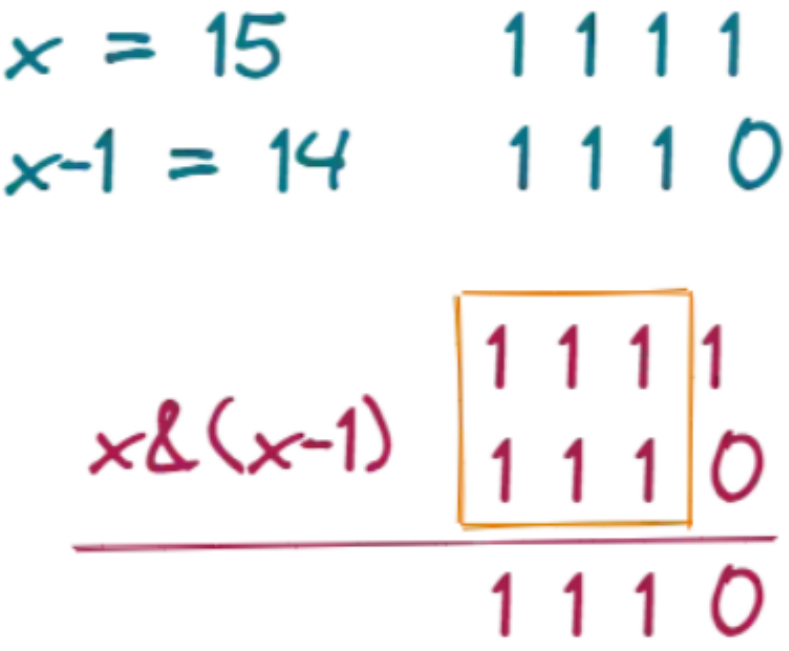
Little-endian 04 03 02 01

64 0 1

Brian Kernighan 1

Brian Kernighan

`x=x&(x-1)`



对数字x去除二进制最右侧的1

```
#include <stdio.h>
#include <stdint.h>

void count_bits(uint64_t number, int *count_zeros, int *count_ones) {
    *count_ones = 0;
```

```
    while (number) {
        number &= (number - 1);
        (*count_ones)++;
    }
    *count_zeros = 64 - *count_ones;
}

int main() {
    uint64_t number = 0b11010101;  //
    int count_zeros, count_ones;
    count_bits(number, &count_zeros, &count_ones);
    printf("Number of 0s: %d, Number of 1s: %d\n", count_zeros, count_ones);
    return 0;
}
```

MSB LSB

64 Highest Significant Bit, MSB Least Significant Bit, LSB

```
#include <stdio.h>
#include <stdint.h>

//     LSB
int find_lsb(uint64_t number) {
    if (number == 0) return -1;  //   0
    int position = 0;
    while ((number & 1) == 0) {
        number >>= 1;
        position++;
    }
    return position;
}

//     MSB
int find_msb(uint64_t number) {
    if (number == 0) return -1;  //   0
    int position = 63;
    while ((number & (1ULL << position)) == 0) {
        position--;
    }
    return position;
}

int main() {
    uint64_t number = 0b11010101;  //
    int lsb_position = find_lsb(number);
    int msb_position = find_msb(number);

    if (lsb_position != -1) {
        printf("LSB position: %d\n", lsb_position);
```

```
    } else {
        printf("The number has no LSB (number is 0).\n");
    }

    if (msb_position != -1) {
        printf("MSB position: %d\n", msb_position);
    } else {
        printf("The number has no MSB (number is 0).\n");
    }

    return 0;
}
```

memory copy

```
561 void *memcpy(void *dest, const void *src, size_t count)
562 {
563         char *tmp = dest;
564         const char *s = src;
565
566         while (count--)
567                 *tmp++ = *s++;
568         return dest;
569 }
```

7.2 Bit

[toc]

338.

```
vector<int> ans;
    int countOnes(int nums) {
        int numOnes = 0;
        while(nums){
            nums &= (nums - 1);
            numOnes++;
        }
        return numOnes;
    }
    vector<int> countBits(int n) {
        for (int index = 0; index <= n; index++) {
            ans.push_back(countOnes(index));
        }
        return ans;

    }
```

Brian Kernighan 1

Brian Kernighan

`x=x&(x-1)`

7.3 LeetCode

[toc]

35. while

- mid
-

```
class Solution {
public:
    int searchInsert(vector<int>& nums, int target) {
        int left = 0;
        int right = nums.size() - 1;
        while(left<=right) {
            //     left + right    int max
            //
            int mid = left + ((right -left) >> 1);
            if (nums[mid] > target) {
                //
                right = mid - 1;
            } else if (nums[mid] < target) {
                //
                left = mid + 1;
            } else {
                return mid;
            }
        }
        //
        return left;

    }
};
```

74. for

mid

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        int left = 0;
        int right = matrix.size() * matrix[0].size() - 1;

        while (left <= right) {
            int mid = left + ((right - left) >> 1);
```

```
            int mid_value = matrix[mid/matrix[0].size()][mid%matrix[0].size()];
            // printf("%d \n", mid_value);

            if (target == mid_value) {
                return true;
            } else if (target > mid_value) {
                //right
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        // for (int x = 0; x < matrix.size(); x++) {
        //     for (int y = 0; y < matrix[0].size(); y++) {
        //         // printf("x-y: %d - %d   %d\n", x, y, matrix[x][y]);
        //         if (target == matrix[x][y]) {
        //             return true;
        //         }
        //     }
        // }
        return false;

    }
};
```

34.

first last

```
class Solution {
public:
    vector<int>  ans;
    vector<int> searchRange(vector<int>& nums, int target) {
        int left = 0;
        int right = nums.size() - 1;

        int first = -1;
        int last = -1;



        //
        if (nums.size() == 0) {
            ans.push_back(-1);
            ans.push_back(-1);
            return ans;
        }

        while (left<=right)
        {
```

```
            /* code */
            int mid = left + ((right - left) >> 1);
            if (nums[mid] == target){
                first = mid;
                right = mid -1;
            } else if (nums[mid] > target) {
                //
                right = mid -1;
            } else {
                left = mid + 1;
            }
        }

        left = 0;
        right = nums.size() - 1;

        while (left<=right)
        {
            /* code */
            int mid = left + ((right - left) >> 1);
            if (nums[mid] == target){
                last = mid;
                left = mid +1;
            } else if (nums[mid] > target) {
                //
                right = mid -1;
            } else {
                left = mid + 1;
            }
        }

        ans.push_back(first);
        ans.push_back(last);
        return ans;


    }
};
```

33.

```
class Solution {
public:
    int search(vector<int>& nums, int target) {
        int left=0,right=nums.size()-1;
        while(left<=right){
```

```
            int mid=left+(right-left)/2;
            if(target==nums[mid]) return mid;
            //
            if(nums[left]<=nums[mid]){
                //
                if(target>=nums[left]&&target<nums[mid]){
                    right=mid-1;
                }else{
                    //
                    left=mid+1;
                }
            }else{
                //
                if(target>nums[mid]&&target<=nums[right]){
                    left=mid+1;
                }else{
                    right=mid-1;
                }
            }
        }
        return -1;
    }
};
```

153.

7.4

[toc]

160.

-

```
class Solution {
public:
    ListNode *getIntersectionNode(ListNode *headA, ListNode *headB) {

        if (headA == NULL || headB == NULL) {
            return NULL;
        }
        ListNode *pa = headA;
        ListNode *pb = headB;
        while (pa != pb) {
            pa = (pa == NULL ? headB : pa->next);
            pb = (pb == NULL ? headA : pb->next);
```

```
        }
        return pa;
    }
};
```

206.

```
ListNode* reverseList(ListNode* head) {
        ListNode *pre = nullptr;
        ListNode *curr = head;

        while (curr  = NULL) {
            /*      temp         */
            ListNode * temp = curr->next;
            curr->next = pre;
            pre = curr;
            curr = temp;
        }

        return pre;
```

92. II

234.

index

```
vector<int> nums;

        ListNode* p = head;

        while (p != nullptr) {
            nums.push_back(p->val);
            p = p->next;
        }

        for (int i = 0, j = (int)nums.size() - 1; i < nums.size()/2; i ++, j--) {
            if (nums[i] != nums[j]) {
                return false;
```

```
            }
        }
        return true;
```

141.

false.

```
        if (head == NULL || head->next == NULL) {
            return false;
        }

        ListNode * fast = head->next;
        ListNode * slow = head;

        while(fast != slow) {
            if (fast == NULL || fast->next == NULL) {
                return false;
            }

            fast = fast->next->next;
            slow = slow->next;
        }
        return true;
```

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        unordered_set<ListNode*> table;
        while(head) {
            if (table.find(head) != table.end()) {
                return true;
            }
            table.insert(head);
            head = head->next;
        }
        return false;

    }
};
```

142. II

a b

f=2s

f=s+nb f s n

s=nb

Jb

```
class Solution {
public:
    ListNode *detectCycle(ListNode *head) {
        if (head == NULL || head->next == NULL) {
            return NULL;
        }
        ListNode* fast = head;
        ListNode* slow = head;

        while(true) {
            if (fast == NULL || fast->next == NULL) {
                return NULL;
            }
            fast = fast->next->next;
            slow = slow->next;
            if (fast == slow) {
                break;
            }
        }

        if (fast == slow) {
            ListNode* ptr = head;
            while (ptr != slow) {
                slow = slow->next;
                ptr = ptr->next;
            }
            return ptr;
        }
        return NULL;
    }
};
```

```
class Solution {
public:
    ListNode *detectCycle(ListNode *head) {
        unordered_set<ListNode*> table;
        while(head){
            if (table.find(head) != table.end()) {
                return head;
            }
            table.insert(head);
            head = head->next;
        }
        return NULL;

    }
};
```

21.

```
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        if (list1 == nullptr) {
            return list2;
        } else if (list2 == nullptr) {
            return list1;
        } else if (list1->val < list2->val) {
            list1->next = mergeTwoLists(list1->next, list2);
            return list1;
        } else {
            list2->next = mergeTwoLists(list1, list2->next);
            return list2;
        }
    }
```

24.

1.
2.
3.

```
class Solution {
public:
    ListNode* swapPairs(ListNode* head) {
        if (head == nullptr || head->next == nullptr) {
            return head;
        }
        //
        ListNode* newHead = head->next;
        //          newhead
        head->next = swapPairs(newHead->next);
```

```
        //
        newHead->next = head;
        //
        return newHead;
    }
};
```

25. K

K head

K

```
class Solution {
public:
    ListNode* reverseKGroup(ListNode* head, int k) {
        ListNode *p = head;
        //      *K*     head
        //    K
        for(int i = 0; i < k; i++) {
            if(!p) return head;
            p = p->next;
        }

        // *K*
        ListNode *q = head;
        ListNode *pre = nullptr;
        while(q != p) {
            ListNode *tmp = q->next;
            q->next = pre;
            pre = q;
            q = tmp;
        }

        //
        head->next = reverseKGroup(p, k);
        return pre;
    }
};
```

2.

```
l1 = [2,4,3], l2 = [5,6,4]
[7,0,8]
342 + 465 = 807.
```

```
class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
```

```
        ListNode *head = nullptr, *tail = nullptr;
        int carry = 0;
        while (l1 || l2) { //
            //              0
            int n1 = l1 ? l1->val: 0;
            int n2 = l2 ? l2->val: 0;
            //
            int sum = n1 + n2 + carry;
            //  head
            // 10    10
            if (!head) {
                head = tail = new ListNode(sum % 10);
            } else {
                tail->next = new ListNode(sum % 10);
                tail = tail->next;
            }
            //
            carry = sum / 10;
            //
            if (l1) {
                l1 = l1->next;
            }
            if (l2) {
                l2 = l2->next;
            }
        }
        //
        if (carry > 0) {
            tail->next = new ListNode(carry);
        }
        return head;
    }
};
```

19. N

dummy head

```
class Solution {
public:
    ListNode* removeNthFromEnd(ListNode* head, int n) {
        if (head == nullptr || head->next == nullptr) {
            return nullptr;
        }
        ListNode* dummy = new ListNode(0, head);
        ListNode* p1 = head;
        ListNode* p2 = dummy;

        for (int i = 0; i < n; i++) {
            p1 = p1->next;
```

```
        }

        while(p1){
            p1 = p1->next;
            p2 = p2->next;
        }

        p2->next = p2->next->next;

        ListNode* ans = dummy->next;
        delete dummy;

        return ans;

    }
};
```

146. LRU

138.

hash value

DNA

```
class Solution {
public:
    unordered_map<Node*,Node*>hmap;
    Node* copyRandomList(Node* head) {
        Node *p=head;
        //          DNA
        //   value
        while(p){
            hmap.insert({p,new Node(p->val)});
            p=p->next;
        }
        //
        p=head;
        while(p){
            //   next
            hmap[p]->next=hmap[p->next];
            //   random
            hmap[p]->random=hmap[p->random];
            p=p->next;
        }
        return hmap[head];
```

```
    }
};
```

147.

```
class Solution {
public:
    ListNode* insertionSortList(ListNode* head) {
        //
        if (head == nullptr) {
            return head;
        }

        //
        ListNode* dummyHead = new ListNode(0);
        dummyHead->next = head; //

        ListNode* lastSorted = head; //
        ListNode* curr = head->next; //

        //
        while (curr != nullptr) {
            //                lastSorted
            if (lastSorted->val <= curr->val) {
                lastSorted = lastSorted->next;
            } else {
                //
                ListNode* prev = dummyHead;
                while (prev->next->val <= curr->val) {
                    prev = prev->next;
                }

                //
                lastSorted->next = curr->next;
                curr->next = prev->next;
                prev->next = curr;
            }

            //
            curr = lastSorted->next;
        }

        //
        return dummyHead->next;
    }
};
```

PCIe

148.

```
head = [4,2,1,3]
[1,2,3,4]
```

23. K

7.5

C **ctype.h**

---

int isalpha(int c)

---

int isdigit(int c)
int islower(int c)
int tolower(int c)
int toupper(int c)

125.

```
: s = "A man, a plan, a canal: Panama"
true
"amanaplanacanalpanama"
```

```
class Solution {
public:
    bool isPalindrome(string s) {
        string temp;
        for (auto c:s) {
            if (isdigit(c)) temp += c;
            if (islower(c))  temp += c;
            if (isupper(c))  temp += tolower(c);
        }
        std::cout<<temp<<std::endl;

        int left = 0;
        int right = temp.length() - 1;
        while (left < right) {
            if (temp[left] == temp[right]) {
                left++;
                right--;
            } else {
                return false;
            }
        }
        return true;
```

```
    }
};
```

392.

"ace" "abcde" "aec"

i j s t i j s j i t s

```
class Solution {
public:
    bool isSubsequence(string s, string t) {
        int l_s = s.length();
        int l_t = t.length();
        int i = 0, j = 0;

        while(i < l_s && j < l_t) {
            if (s[i] == t[j]) {
                i++;
            }
            j++;
        }
        return i == l_s;

    }
};
```

7.6

[toc]

1.

nums = [2,7,11,15], target = 9

[0,1]

nums[0] + nums[1] == 9 [0, 1]

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int ,int> table;
        for (int i = 0; i < nums.size(); i++) {
            auto it = table.find(target - nums[i]);
            if (it != table.end()) {
                return {it->second, i};
            }

            table[nums[i]] = i;

        }
```

```
        return {};

    }
};
```

49.

```
 : strs = ["eat", "tea", "tan", "ate", "nat", "bat"]
 : [["bat"],["nat","tan"],["ate","eat","tea"]]
```

```
class Solution {
public:
    vector<vector<string>> groupAnagrams(vector<string>& strs) {

        //      key     value
        unordered_map<string, vector<string>> map;

        //
        for (auto str:strs){
            string key = str;
            sort(key.begin(), key.end());
            map[key].push_back(str);
        }

        //          ans
        vector<vector<string>> ans;
        for (auto it = map.begin(); it != map.end(); it++) {
            ans.push_back(it->second);
        }
        return ans;

    }
};
```

128.

```
 nums = [100,4,200,1,3,2]
 4
       [1, 2, 3, 4]    4
```

```
class Solution {
public:
    int longestConsecutive(vector<int>& nums) {
        //     ,
        unordered_set<int> num_set;
        for (const int& num : nums) {
            num_set.insert(num);
        }

        int longestStreak = 0;
```

```
        for (const int& num : num_set) {
            //   num - 1   num_set    num
            if (!num_set.count(num - 1)) {
                int currentNum = num;
                int currentStreak = 1;

                //   num
                //        currentStreak
                while (num_set.count(currentNum + 1)) {
                    currentNum += 1;
                    currentStreak += 1;
                }

                //   max    longestStreak
                //
                longestStreak = max(longestStreak, currentStreak);
            }
        }

        return longestStreak;
    }
};
```

7.7

[toc]

283.

    nums   0

 : nums = [0,1,0,3,12]
 : [1,3,12,0,0]

  0   0   0

```
class Solution {
public:
    void moveZeroes(vector<int>& nums) {
        int n = nums.size(), left = 0, right = 0;
        while (right < n) {
            if (nums[right]) {
                swap(nums[left], nums[right]);
                left++;
            }
            right++;
        }
    }
};
```

11.

```
class Solution {
public:
    //
    int maxArea(vector<int>& height) {
        int left = 0, right = height.size() - 1;  //
        int maxArea = 0;  //      0

        //
        while (left < right) {
            //
            int currentArea = (right - left) * min(height[left], height[right]);
            //
            maxArea = max(maxArea, currentArea);

            //
            if (height[left] < height[right]) {
                left++;
            } else {
                right--;
            }
        }

        return maxArea;  //
    }
};
```

15.

7.8 Stack

[toc]

20.

```
    '(' ')' '{' '}' '[' ']'    s

          ([{})}


class Solution {
public:
    bool isValid(string s) {
        int n = s.size();
        if (n % 2 == 1) {  //          false
            return false;
        }

        //
        unordered_map<char, char> pairs = {
```

```
            {')', '('},
            {']', '['},
            {'}', '{'}
        };

        stack<char> stk;  //

        //
        for (char ch: s) {
            if (pairs.count(ch)) {  //
                //              false
                if (stk.empty() || stk.top() != pairs[ch]) {
                    return false;
                }
                stk.pop();  //
            } else {
                stk.push(ch);  //
            }
        }

        return stk.empty();  //
    }
};
```

155.

push pop top

```
class MinStack {
public:
    stack<int> x_stack;
    stack<int> min_stack;

    MinStack() {
        min_stack.push(INT_MAX);
    }

    void push(int val) {
        x_stack.push(val);
        min_stack.push(min(min_stack.top(), val));
    }

    void pop() {
        x_stack.pop();
        min_stack.pop();
    }
```

```
    int top() {
        return x_stack.top();
    }

    int getMin() {
        return min_stack.top();
    }
};

/**
 * Your MinStack object will be instantiated and called as such:
 * MinStack* obj = new MinStack();
 * obj->push(val);
 * obj->pop();
 * int param_3 = obj->top();
 * int param_4 = obj->getMin();
 */
```

394.

```
 s = "3[a]2[bc]"
 "aaabcbc"
```

```
string decodeString(string s) {
    string ans = "";  //

    stack<int> num_stk;     //
    stack<string> str_stk;  //

    int temp_num = 0;       //
    int len_s = s.size();   //

    for (int i = 0; i < len_s; i++) {
        if (isdigit(s[i])) {
            temp_num = temp_num * 10 + s[i] - '0';  //
        } else if (isalpha(s[i])) {
            ans = ans + s[i];  //
        } else if (s[i] == '[') {
            num_stk.push(temp_num);     //
            temp_num = 0;               //   temp_num
            str_stk.push(ans);          //    ans
            ans = "";                   //   ans
        } else if (s[i] == ']') {
            int times = num_stk.top();      //
            num_stk.pop();                  //
            string temp = str_stk.top();    //
            str_stk.pop();                  //

            //   ans    times     temp
            for (int j = 0; j < times; ++j) {
```

```
                temp += ans;
            }

            ans = temp;  //   ans
        }
    }

    return ans;  //
}
```

739.

```
 : temperatures = [73,74,75,71,69,72,76,73]
 : [1,1,4,2,1,1,0,0]
```

```
vector<int> dailyTemperatures(vector<int>& temperatures) {
    int n = temperatures.size();  //

    vector<int> ans(n, 0);  //         0
    stack<int> st;  //

    for (int i = 0; i < temperatures.size(); i++) {
        //
        while (!st.empty() && temperatures[i] > temperatures[st.top()]) {
            auto t = st.top();  //
            st.pop();  //
            ans[t] = i - t;  //
        }
        st.push(i);  //
    }

    return ans;  //
}
```

Chapter 8

System Design

system-design-primer

Tool

8.1 Git

git bash git status

git `core.quotepath` false quotepath `--global`

git bash

```
git config --global core.quotepath false
```

Part IV

Chapter 9

[toc]

1.
2.
3.
4.
5.
6.
7.

Meeting Minutes

1.
2. Jira HSD unit test
3. BIOS OS BIOS

One on One

manager manager second manager arch Jun one one

Chapter 10

2024

[toc]

1566

——

—— GPT-4 ChatGPT

——

ARM

——