**SoC相关知识点梳理**

**更改历史**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本** | **状态** | **日期** | **责任人** | **更改原因** |
| 1.0 | 草稿 | 2025-03-25 | liuyongqi (刘永琪) | 创建 |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1 介绍 3](#_Toc193895084)

[1.1 文档目的 3](#_Toc193895085)

[2 术语和定义 3](#_Toc193895086)

[2.1 术语 3](#_Toc193895087)

[2.2 定义 3](#_Toc193895088)

[2.2.1 DMA 3](#_Toc193895089)

[2.2.2 TCM 3](#_Toc193895090)

[2.2.3 ITCM 3](#_Toc193895091)

[2.2.4 DTCM 3](#_Toc193895092)

[2.2.5 Cache 3](#_Toc193895093)

[2.2.6 DMA与Cache的冲突 3](#_Toc193895094)

[2.2.7 SoC存储结构 4](#_Toc193895095)

[3 参考文档 4](#_Toc193895096)

[4 SoC系统和架构设计 5](#_Toc193895097)

[4.1 概述 5](#_Toc193895098)

[4.1.1 SoC组成 5](#_Toc193895099)

[4.1.2 架构设计 5](#_Toc193895100)

[4.2 处理器子系统设计 5](#_Toc193895101)

[4.2.1 多核处理器 5](#_Toc193895102)

[4.2.2 处理器子系统 6](#_Toc193895103)

[4.3 存储子系统设计 6](#_Toc193895104)

[4.3.1 存储结构 7](#_Toc193895105)

[4.3.2 存储器件 7](#_Toc193895106)

[4.3.3 存储器映射与重映射 8](#_Toc193895107)

[4.4 互连子系统设计 8](#_Toc193895108)

[4.4.1 互连类型 8](#_Toc193895109)

[4.4.2 系统总线 9](#_Toc193895110)

[4.5 芯片管理设计 10](#_Toc193895111)

[4.5.1 电源管理设计 10](#_Toc193895112)

[4.5.2 时钟管理设计 10](#_Toc193895113)

[5 全文更新索引的方法： 10](#_Toc193895114)

# 介绍

## 文档目的

本文档用于记录SoC的相关知识点，包括SoC系统、架构设计、现代微处理器架构、存储系统等，为入门SoC提供概要的知识点，可用于梳理SoC相关功能、整体系统设计，增加对SoC或子系统的完整理解，加深对SoC设计全流程的认识，掌握常用的SoC相关的专业术语，提高问题描述和讨论的质量和效率。

# 术语和定义

## 术语

1. VLSI：Very Large Scale Integration，超大规模集成
2. GPU：Graphics Processing Unit，图形处理器
3. CPU：Central Processing Unit，中央处理器
4. ASIC：Application Specific Integrated Circuit，专用集成电路芯片
5. FPGA：Field Programmable Gate Array，现场可编程门阵列
6. ALU：Arithmetic and Logic Unit，算术逻辑单元，实现多组算术运算和逻辑运算的组合逻辑电路
7. ROM：Read-Only Memory，只读存储器
8. RAM：Random Access Memory，随机存取存储器，常见的是SRAM和DRAM；
9. DRAM：Dynamic RAM，动态RAM
10. TPU：Tensor Processing Unit，张量处理器，谷歌专门为提高神经网络计算能力而研发的一款ASIC；
11. DMA：Direct Memory Access，直接存储器访问
12. AHB：Advanced High Performance Bus，高级高性能总线、
13. APB：Advanced Peripheral Bus，外围总线，该总线协议是ARM公司提出的AMBA总线结构之一，几乎已成为一种标准的片上总线结构；
14. USB：Universal Serial Bus，通用串口总线
15. UART：Universal Asynchronous Receiver/Transmitter，通用异步收发器，是一种通用**串行数据总线**，用于异步通信。该总线双向通信，可以实现全双工传输和接收。在嵌入式设计中，UART用来与PC进行通信；
16. AMBA 全称叫 Advanced Microcontroller Bus Architecture，即高级处理总线架构。其下包含了**AHB、ASB、APB、AXI**等几种总线
17. PMIC：Power Management IC，电源管理芯片，SoC外部的电源管理模块称为电源管理芯片；
18. PMU：Power Management Unit，电源管理单元，SoC内部的电源管理模块称为电源管理单元；
19. PLL：Phase-Locked Loop，是一种输出一定频率信号的振电路，也称为相位同步环(回路)。该回路利用外部施加的基准信号与PLL 回路内的振荡器输出的相位差恒定的反馈控制来产生振荡信号。

## 定义

### DMA

Direct Memory Access，直接内存访问，一种**硬件加速的数据传输机制**，绕过CPU直接操作总线完成内存与外设、内存与内存之间的杉树搬运。避免CPU因频繁搬运数据而发生阻塞，提升系统整体效率，**DMA无法访问TCM**，需要确保DMA的源地址和目的地址位于普通内存区域；

### TCM

Tightly Coupled Memory，紧密耦合内存，**集成在CPU内部**的高速存储器，分为ITCM（指令TCM）和DTCM（数据TCM），访问速度与CPU内核同步，延迟极低。TCM容量小（通常KB级别），需要谨慎分配给关键代码和数据；

### ITCM

Instruction TCM，存储实时性要求高的代码（如中断处理函数、加解密函数），通过将编译器属性标签将代码段映射到ITCM地址

### DTCM

Data TCM，存储频繁访问的数据（如栈、全局变量、实时处理缓冲区），通过**链接脚本定义数据段地址**，或直接操作固定映射地址

### Cache

1. CPU内部的多级高速缓存（如L1/L2/L3），通过存储近期访问数据的副本减少内存访问延迟；
2. 读操作：**优先从Cache读取数据**，若未命中则访问内存；
3. 写操作。**数据先写入Cache**，待Cache满或特定条件**触发时回写内存（Write-Back策略）**

### DMA与Cache的冲突

#### 数据访问路径差异

1. DMA直接操作物理内存（绕过CPU），而Cache是CPU的**本地高速缓存**，两者对内存的访问路径不同；
2. 当CPU通过Cache读写数据时，若DMA直接修改内存，会导致**Cache中保存的数据与内存实际数据不一致**；

#### 数据过时的具体场景

1. CPU写操作未同步到内存：若CPU修改了Cache中的数据（采用“写回”策略），但未及时回写到内存，此时DMA从内存读取的数据是旧值；
2. DMA写操作未同步到Cache：若DMA将新数据写入内存，但Cache中仍保留旧数据，CPU后续读取Cache会得到错误数据[9](https://blog.csdn.net/weixin_43850980/article/details/128625032)；

#### Cache手动刷新（Flush）与失效（Invalidate）的场景

1. 刷新（Flush）操作

作用：将Cache中修改过的数据（Dirty数据）强制写回内存，确保内存与Cache数据一致。

适用场景：

1. DMA向外设发送数据前：若数据由CPU生成并存储在Cache中，需先刷新Cache，确保DMA从内存获取最新数据。
2. 多核系统中共享内存操作：避免其他核的Cache数据未更新导致冲突。

2. 失效（Invalidate）操作

作用：**标记Cache中特定地址的数据为无效**，强制后续访问时**从内存重新加载**。

适用场景：

1. DMA从外设接收数据后：若DMA将新数据写入内存，需失效Cache中对应的旧数据，避免CPU读取过时数据。
2. 动态配置内存区域时：例如重新分配DMA缓冲区前，需失效原有Cache数据2。

### SoC存储结构



图 2‑1 不同层次的存储器访问速度和容量

# 参考文档

1. SoC设计高级教程：系统架构——张庆
2. SoC设计高级教程：技术实现——张庆
3. 3GPP TS 36.321: " Medium Access Control (MAC) protocol specification"

# SoC系统和架构设计

## 概述

### SoC组成

1. SoC由处理器、存储、互联、外设和应用等子系统组成，并且包含芯片管理设计、低功耗设计、可测性设计等模块，系统设计将确定各个子系统和模块的硬件特性；
2. **处理器子系统**至少包含一个微控制器或微处理器，也可以包含CPU、GPU等同构或异构的计算单元；
3. **存储子系统**采用只读存储器（ROM）、随机存储器（RAM）、Flash中的一种或多种；
4. **互联子系统**分为星型连接、环状连接和网状连接，以及基于多**种总线拓扑**的混合连接；
5. **外设子系统**包括系统外设和不同标准的**I/O接口**；
6. **芯片管理**设计包括稳压器等**电源**管理设计、**振荡器和锁相环**等**时钟管理**设计，以及复位管理设计；
7. **低功耗设计**目的在于降低芯片整体的静态和动态功耗；
8. **可测性设计**是在保证功能的前提下，加入特殊的测试结构，以便芯片制造完成后进行测试，**筛选**出有瑕疵的芯片；

### 架构设计

1. 架构设计分为微架构设计和SoC架构设计；
2. 微架构设计偏重于**模块级**的架构，一般用于**IP或小型系统**的架构设计；
3. SoC架构设计狭义上指SoC的软硬件架构设计，包括**计算架构、总线互联架构、存储架构**等，广义上，包括SoC的工艺和封装选型、PPA（性能、功耗和面积）估算、需求确认和分解、IP选型、软硬件划分、硬件架构规划、ESL（电子系统级）评估、低功耗设计、信息安全策略规划、可测性目标建议；

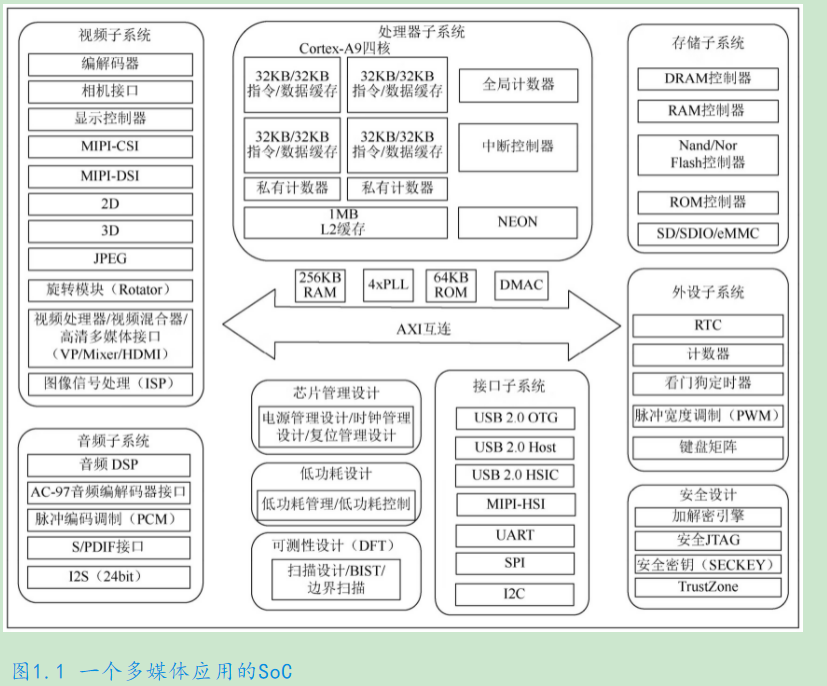


图 4‑1 一个多媒体应用SoC架构示例

## 处理器子系统设计

### 多核处理器

#### CPU内核

1. 多核处理器包括CPU内核、高速缓存、内存管理单元、总线接口单元、通用中断控制器、调试与跟踪单元、侦听控制单元、加速器一致性端口等；
2. CPU的计算速度与主频、流水线（Pipeline）和总线等方面的性能指标有关，**主频是CPU内核的时钟频率，可以理解为运算速度**，主频越大计算速度越快，主频的单位是MHz或GHz；
3. 流水线通常是**多级流水线**，冯· 诺依曼型处理器都基于5个基本操作：取指（Instruction Fetch, IF）、译码（Instruction Decode, ID）、执行（Execute,EXE）、访存（Memory Access, MEM）、写回（Write Back,WB），这些操作可以顺序执行也可以乱序执行；
4. 总线频率影响CPU与外部的数据交换，总线的带宽取决于**总线频率**和**传输数据的位宽**；

#### 缓存

1. 目前的通用处理器均采用片上多级缓存方式解决处理器与内存之间的速度不匹配问题；
2. 对于多核处理器系统，若共享内存，则存在缓存一致性问题，可由软件或缓存一致性管理硬件来处理；

#### 内存管理单元

1. 内存中的每个单元都有一个唯一的编号，即内存地址，或成为物理地址；
2. 内存地址的集合称为内存地址空间或物理地址空间；
3. 源程序经过汇编或编译后形成的目标程序，目标程序中的地址成为逻辑地址或虚拟地址，并且每个目标程序都是**从0地址开始编址**；
4. 内存管理单元（MMU：Memory Management Unit）负责**将用户程序指令或数据的逻辑地址**转换为**存储空间中的物理地址**，这个过程成为**内存映射**；

#### 总线接口单元

总线接口单元（BIU：Bus Interface Unit）负责处理器与外部总线、内存之间的**数据传输**，以实现高带宽和低延迟。

#### 通用中断控制器

1. 中断是芯片的重要功能，当产生中断时，处理器的中断函数可以打断主程序的执行，并在执行完成后重选返回到主程序的执行；
2. 中断可以用于处理一些实时性要求高的场景，如按键、核减交互等；
3. 通用中断控制器（GIC：Generic Interrupt Controller）提供了一种强大且灵活的方式，实现处理器之间的通信、路由系统中断和优先级确定；

#### 调试与跟踪单元

1. 处理器运行需要实时观察内核状态，片上跟踪功能指通过专用硬件**非入侵地实时记录程序执行路径和数据读写信息**，并将这些信息压缩成跟踪数据流，通过专用数据通道和输出接口传输至调试主机；
2. 较早期的处理器采用**JTAG**标准调试，现在ARM处理器统一使用**内存映射方式**进行调试
3. 多个处理器的调试与跟踪单元（Debug and Trace Unit）可以分别外链，或者内部以菊花链（Daisy Chain）方式链接后再连出；

#### 侦听控制单元

1. 侦听控制单元（SCU：Snoop Control Unit）维护**多处理器内核与下级共享缓存**之间的一致性，降低在各个操作系统**驱动程序中维持软件一致性**所涉及的软件复杂性；
2. Snoop：调查、窥探

#### 加速器一致性单元

1. 加速器一致性接口（ACP：Accelerator Coherency Port），AMR处理器多核架构下定义的一种端口，用于管理**不带缓存的外设**，从而提高处理器运行效率并与外部数据源达成可靠的高速缓存一致性；
2. Coherency：/koˈhɪrənsi/ n. 一致

### 处理器子系统

处理器子系统拥有两个或多个紧密通信的处理器、共享总线、内存和外设等，主要关注**缓存一致性**和启动。

#### 多核处理器和多处理器

1. SoC有多个处理器内核或多个处理器，以允许**不同进程同时运行**，提高系统速度；
2. 具有多个内核处理器被称为多核处理器（Multi-Core Processor），物理内核越多，性能越强；
3. 多处理器（Multi-Processor）：芯片上包含两个或多个同构或异构的处理器，分为**单芯片多处理器**（CMP：Chip Multi-Processor）和**片上多处理器系统**（MPSoc：Multi-Processor System-on-Chip）；
4. CMP用于通用计算类型SoC，如服务器芯片，MPSoC用于复杂计算类型SoC，如媒体芯片；

#### 芯片启动源

1. SoC芯片启动时，会从内部ROM（Internal ROM，IROM）运行程序，当不能从IROM中启动时，可以从其他源启动，增强芯片的容错能力；
2. 可以基于不同的需求从不同的源启动，栓NorFlash提供的XIP（eXecute In Place，就地执行）模式；
3. ROM内部的数据是在制造工艺中使用特殊方法烧录进入的，其内容只能读不能修改，用户只能验证写入数据的正确性；

#### 异构处理器

1. 异构计算：通过在单一系统中利用不同类型的处理器，如CPU、GPU、ASIC、FPGA等协同工作，执行特定任务，以优化性能和效率，更高效地利用不同类型的计算资源，满足不同的计算需求；
2. 目前，异构计算主要的大类：CPU+GPU、CPU+FPGA、CPU+ASIC，CPU执行通用计算，其他架构处理器完成特定领域的计算，目前GPU是商用最广泛的AI芯片；
3. XPU：各种类型的PU，如CPU、GPU等；
4. GPU：Graphics Processing Unit，图像处理器，采用数量众多的计算单元和超长的流水线，解决处理器在大规模并行计算中所遭遇的难题，提高数据处理速度。GPU不能单独工作，必须由处理器控制调用才行。当处理器需要处理大量类型统一的数据时，可以调用GPU进行并行计算；
5. GPU在图像处理方面能力非常强，因为图像上每一个像素点都有处理需求，而且处理过程和方式十分相似，可以并行计算；
6. GPU不限于图像处理，还可广泛用于科学计算、密码破解、数值分析和海量数据处理等大规模并行计算的领域；
7. 与CPU相比，GPU中拥有大量的ALU（算数逻辑单元）用于并行的数据运算；

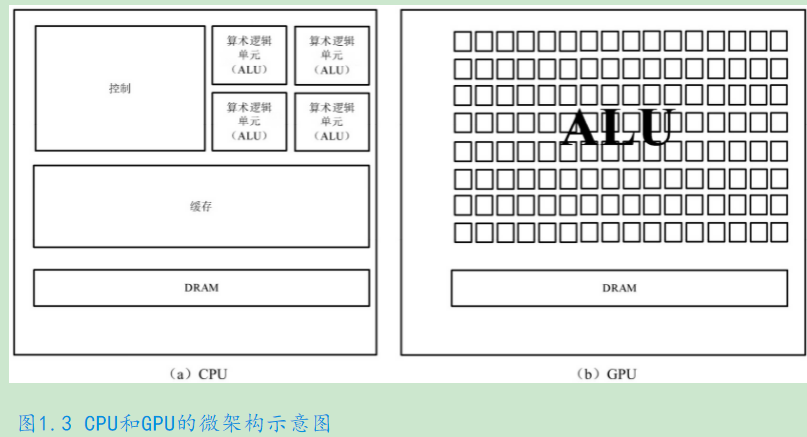


图 4‑2 CPU和GPU的架构图

## 存储子系统设计

1. 存储子系统可分为内（主）存和外（辅）存两类；
2. 内存是直接受处理器控制与管理，并只能暂存信息的存储器；
3. 外存是可以永久性保存信息的存储器，外存中的程序必须调入内存才能运行；

### 存储结构

1. SoC存储结构是分层的，离处理器越近的存储器速度越快，容量越小；
2. 不同级别的缓存访问的延迟不同，访问延迟：L1<L2<L3…；
3. 处理器可以通过缓存和**便笺存储器**（ScratchPad Memory，**SPM**）访问内存；

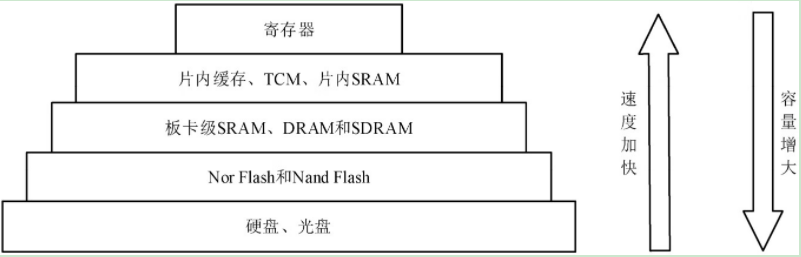


图 4‑3 SoC存储结构

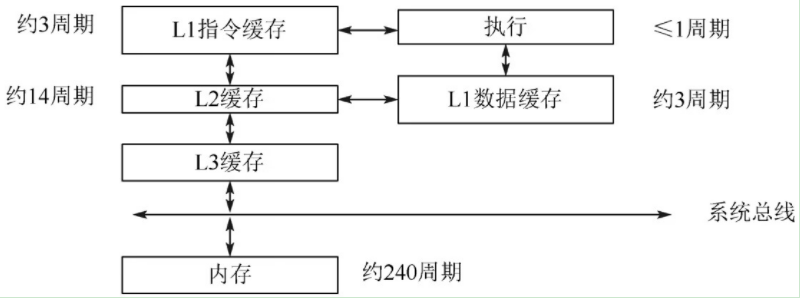


图 4‑4 不同级别的缓存访问延迟不同

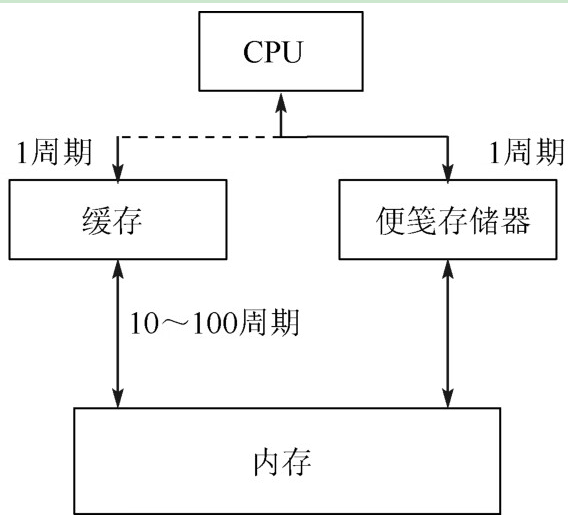


图 4‑5 处理器数据缓存结构

#### 缓存

1. 缓存速度在一定程度上影响着系统的性能，为了进一步提升系统性能，引入多级缓存；
2. L1缓存为CPU专用，L2缓存为多个CPU共享，L2缓存则位于L2缓存和内存之间；
3. CPU取数据时优先在缓存中找，若缓存中没有则需要访问内存；
4. **缓存行（Cache Line）**是缓存与内存之间数据传输的最小单位，在缓存缺失（Cache Miss）的情况下，即使处理器试图加载1Byte数据，缓存行从内存中至少加载一个缓存行大小的内存数据，如缓存行大小设置为8Byte，则一次缓存从内存中至少加载8Byte的数据；

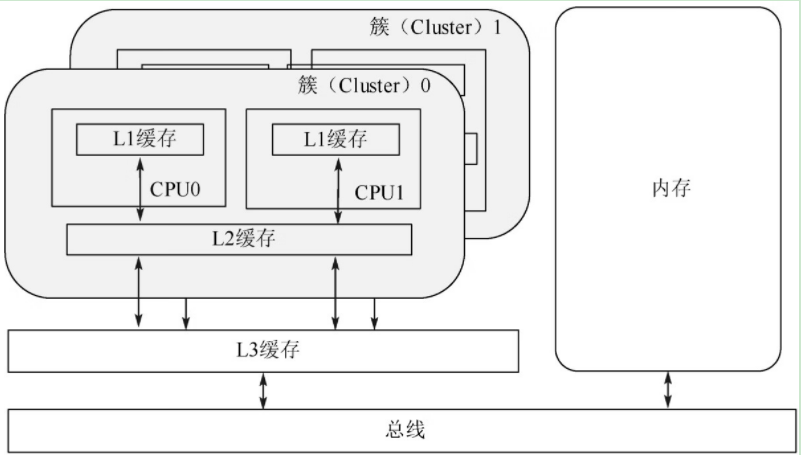


图 4‑6 多级缓存

#### 便笺存储器——SPM

1. 便笺存储器（ScratchPad Memory，SPM），与内存统一编址，处理器可以直接对其进行访问，不会出现访问缓存时缓存缺失（类似于Cache Miss）现象；
2. SPM不需要标记存储器（Tag RAM）就可以直接对其进行访问，起硬件构造相对简单，同一制造工艺下面积仅为缓存的65%，因此功耗低，速度快；
3. 若要实现大量的数据处理，可以使用DMA多次将待处理数据搬运到SPM中，分批处理；
4. SPM与缓存的主要区别：
   1. 缓存主要解决的是用最小的面积解决大部分连续访问的效率问题，只要缓存命中就不用去远处的内存中读取数据，提高处理器的性能，若缓存没命中，则需要读取内存，因此数据存取的延迟不确定，不太适合对计算延迟敏感的场景；
   2. SPM针对实时计算的应用，常见与DSP，若处理器内核附近集成了SPM，则读写延迟固定，计算时间相对可控，适用于高精度工业控制、航空航天、汽车制动等实时性要求高的场景；

### 存储器件

1. 缓存、SPM、内存都是按照应用场景来区分的；
2. 按照存储器件的类型分类，可以分为：**SRAM**（Static Random Access Memory，静态随机访问存储器）、**DRAM**（Dynamic Random Access Memory，动态随机访问存储器）、**Flash**、**OTP**（One Time Programmable，一次性可编程存储器）

#### SRAM

1. SRAM（Static Random Access Memory，静态随机访问存储器），只要不掉电，SRAM中存储的数据就**不会丢失，不需要刷新**；
2. RAM中的R，随机，是指SRAM中的内容可以按照任何顺序访问，与前一次访问的位置无关；
3. 读写速度非常快，但功耗较高且成本昂贵；
4. 通常用于**高速缓存**（如CPU的L1、L2缓存），对速度要求极高的场景；

#### DRAM

1. DRAM（Dynamic Random Access Memory，动态随机访问存储器），动态的意思是指即使在通电的状态下，也只能使数据保持很短的时间，需要通过**不断刷新来保证数据不丢失**；
2. DARM的数据以电容器上的电荷形式存储，由于电容器会失去其数据（电荷），DRAM必须每秒刷新几百次以保持数据，DRAM读写速度比SRAM慢，但**成本较低，集成度高**，是计算机内存的主要组成部分；
3. SDRAM（Synchronous Dynamic Random Access Memory，同步动态随机访问存储器），同步是指具有一个与CPU同步的时钟，内部的指令发送与数据传输都以其为基准；
4. DDR SDRAM（Double Data Rate SDRAM，双倍数据速率SDRAM），也可以简称为DDR，可以在一个周期内读写两次数据。后续存在其他的改进版本DDR2、DDR3、DDR4等；

#### Flash

1. Flash是一种长寿命的非易失性存储器，在断电的情况下任然能够保持所存储的数据信息；
2. Nand Flash和Nor Flash是两种主流的非易失性存储器；
3. Nand Flash**容量较大，改写速度快，廉价**，但是用户**不能直接运行存储于其上的代码**；
4. Nor Flash可以通过处理器总线随机访问，其读取类似SDRAM，但**不支持随机写操作**，提供XIP模式，应用程序可以**直接运行于其上，成本高**，只有小容量的Nor Flash获得的收益较大；

#### OTP存储器

1. 一种特殊的存储器件，只允许一次性写入，一旦写入便不能修改，读取操作不受限制；
2. OTP存储器擦写和读取的速度都比较慢，特点是一旦写入不能被修改，常用于存储芯片的标识符ID、批次、版本、安全Boot密钥等不允许被改写的特定内容；
3. OTP存储器的内容一般在自动化测试阶段通过机台写入，并于芯片启动后读取使用；

#### 存储内容

1. 各种RAM（SRAM、DRAM、SDRAM、DDR SDRAM）都支持随机读和随机写，而ROM和Nor Flash仅支持随机读，不支持随机写；
2. 程序可以存储于支持掉电保存数据的存储器中，如ROM、Nand Flash、Nor Flash，当需要被执行时加载到相应的存储器中，需要安排RAM存储器执行ROM和Nor Flash上的程序；
3. 较大的程序可以运行在动态存储器中，如在多阶段启动中，首先将操作系统从启动源闪存复制到DRAM中，然后启动DRAM中的操作系统；

### 存储器映射与重映射

1. 十六进制的地址编码与存储器之间的对应关系就是存储器映射（Memory Map），也称为内存映射、地址映射（Address Map）；
2. 存储器映射是一个逻辑概念，在芯片上电后才建立起来；
3. 指令代码和数据都位于同一内存地址空间，但可以将他们限制在不同的地址范围中，如通过链接脚本设置.txt和.data；
4. 存储器重映射是对此前已经确立的存储器映射的再次修改，重映射发生在系统启动及运行的过程中，与处理器中断处理密切相关；
5. 当中断产生时，处理器在硬件的驱动机制下跳转到预先设定的存储器，取出相应的中断处理程序的入口地址，执行相应的中断服务程序；

## 互连子系统设计

互连子系统的设计原则：满足系统**延迟和带宽**的要求，同时**减少**主要互连模块的**接口数量**和顶层高速信号的**走线数量**。

### 互连类型

#### 总线

1. 总线是一组信号（连线）的集合，多个需要相互通信的主/从设备连接其上；
2. 共享总线通过时分复用机制实现不同设备之间的通信，当出现多个主设备同时访问一条总线时，需要由仲裁机制决定总线的所有权；
3. 总线的结构简单，硬件代价小，但其带宽有限且无法随设备的增多而扩展；

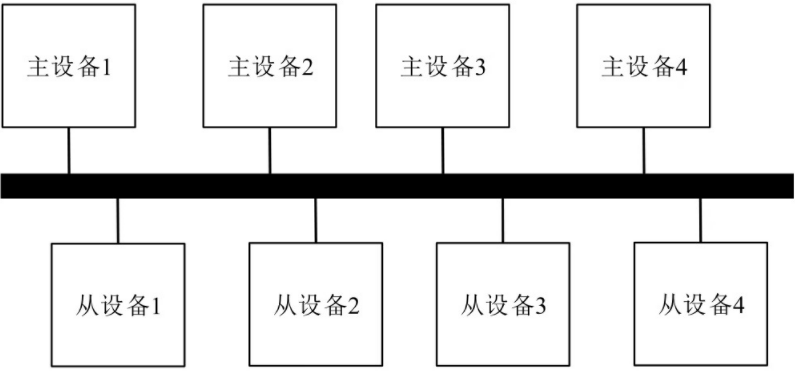


图 4‑7 总线

#### 交叉矩阵

1. 交叉矩阵保证了多路通信可以同时进行，结构简单，互联部分延迟小，适用于数量不多的设备互连；
2. 随着设备数量的增加，交叉矩阵的规模呈几何数增长，导致内部走线非常多，不利于物理实现；

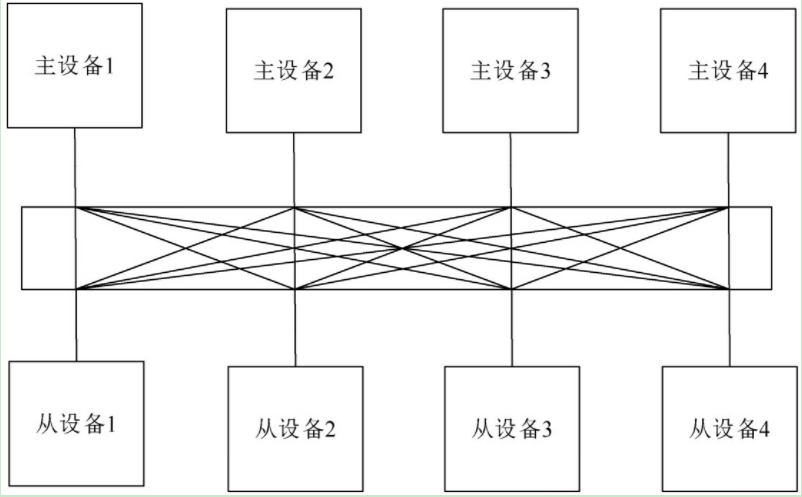


图 4‑8 交叉矩阵

#### 片上网络——NoC

1. 在基于报文交换的片上网络（Network on Chip，NoC）中，每个IP（处理单元）与路由器相连，IP之间的通信转换为路由器之间的通信；
2. NoC实现了更好的扩展性，在吞吐量和带宽方面尤为突出；
3. 设计相对复杂，需要考虑拓扑、路由、流量控制等方面的问题；

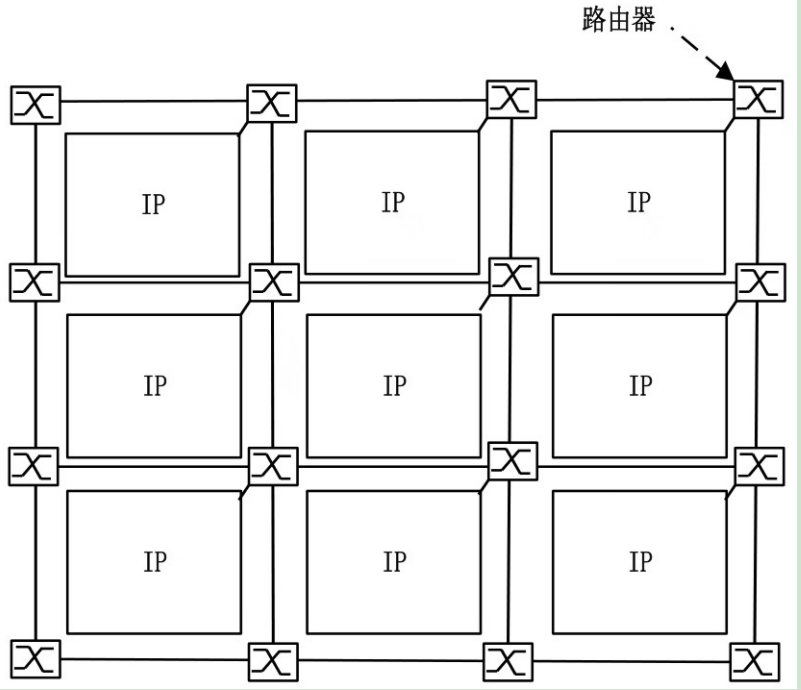


图 4‑9 NoC

#### 缓存一致性总线

1. 缓存一致性总线用于维护不同处理器之间的缓存一致性；
2. 处理器簇内部的L2缓存维护两个L1缓存之间的缓存一致性，而外部的缓存一致性互连（CCI）维护两个处理器簇之间的缓存一致性；

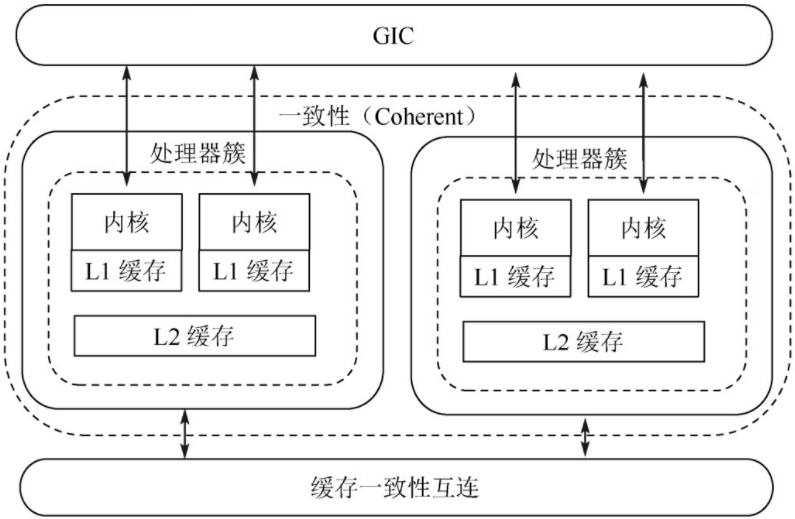


图 4‑10 缓存一致性总线

### 系统总线

系统总线包括数据总线（Data Bus）和寄存器总线（Register Bus）。

#### 数据总线

#### 寄存器总线

寄存器总线用于模块的寄存器配置和状态读取

#### 可靠性、可用性和可维护性

可靠性：系统必须尽可能可靠、不会意外崩溃、重启甚至出现物理损坏，可靠的系统应能对某些小错误做到自修复，以保障系统其余部分正常运转；

可用性：系统必须能够确保尽可能长时间工作而不下线，即使出现了一些小问题也不会影响整个系统的正常运行；

可维护性：系统能够提供便利的诊断功能，采用系统日志、动态检测等收到对系统进行诊断和维护

RAS：Reliable Available Serviceability，用于确保整个系统尽可能长期可靠运行而不下线，并具备足够强大的容错机制；

## 芯片管理设计

芯片管理设计主要包括电源管理设计、时钟管理设计和复位管理设计。

### 电源管理设计

#### 电源的需求

1. SoC的功能越来越负责，包含的IP也越来越多，如处理器、射频模块、音/视频模块、内存控制器和外设等，不同的IP带来了多档电源的需求；
2. 为满足低功耗的需求，SoC通常将被分为多个电源域，不同的电源域可以独立供电，根据芯片工作场景确定供电电压；

#### 电源管理

1. 电源信号可由外部直接输入、或者由片内电路产生；
2. SoC外部的电源管理模块称为电源管理芯片（PMIC：Power Management IC）；
3. SoC内部的电源管理模块称为电源管理单元（PMU：Power Management Unit）；
4. 电源管理芯片具有高集成度，将多路输出电源封装在一个芯片内，使得多电源应用场景的效率更高，体积更小；

#### 电压调节器

1. SoC通常使用两种电压调节器，低压差线性稳压器和开关稳压器；
2. 低压差稳压器（LDO：Low Dropout Regulator），是一种能够在**输入电压**与**输出电压差值较小的情况下**，仍能稳定输出电压的线性稳压器。其核心原理基于**反馈控制和线性调节机制；**
3. 开关稳压器（Switching Regulator）使用输出级，**重复切换“开”和“关”状态**，与能量存贮部件（电容器和感应器）一起产生输出电压。它的调整是通过根据输出电压的反馈样本来调整切换定时来实现的；

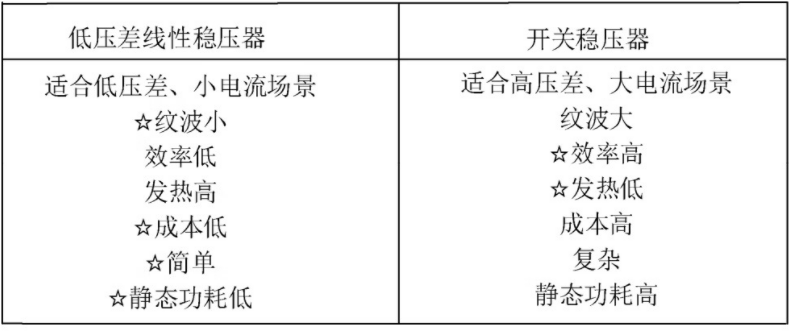


图 4‑11 两种电压调节器的比较

### 时钟管理设计

#### 时钟需求

根据芯片工作场景的需要，确定所需时钟及频率。

#### 时钟源

1. 时钟源信号可由外部直接输入，也可以外接晶体/陶瓷振荡器和内部时钟发生器产生；
2. 内部RC振荡器能够产生精度较差的系统时钟，而外部锁相环（PLL）可以产生高频、高精度的系统时钟；
3. 锁相环路简称锁相环( PLL )，是一种输出一定频率信号的振电路，也称为相位同步环( 回路)。该回路利用外部施加的**基准信号**与PLL 回路内的振荡器输出的**相位差恒定的反馈控制**来产生振荡信号。

# 全文更新索引的方法：

全文选中，按下F9即可