**处理器与接口**

|  |  |
| --- | --- |
| 编制： | Lizy |
| 校对： | - |
| 2023.4.15 | |

更改记录

|  |  |
| --- | --- |
| **发布日期** | **更改描述** |
| 20230415 | 第一版发布 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 引言

目前，CPU和MPU两类芯片的发展值得关注：

CPU（Central Processing Unit，中央处理器）作为信息产业中最基础的核心部件，是信息处理、程序运行的最终执行单元，其设计技术门槛高、研发周期长，具有极高的生态壁垒，其国产化具有重要的战略意义。

MPU（Microprocessor Unit，微处理器）是把计算机的CPU功能在单片或几片集成电路中实现的器件，是[微型计算机](https://www.zhihu.com/search?q=%E5%BE%AE%E5%9E%8B%E8%AE%A1%E7%AE%97%E6%9C%BA&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra=%7b)的核心部件。随着终端功能的需求逐渐多样化，它增加了图形、视频，甚至新兴的人工智能（AI）应用等越来越多的系统级功能，可以用于各种系统的嵌入式处理，如[网络设备](https://www.zhihu.com/search?q=%E7%BD%91%E7%BB%9C%E8%AE%BE%E5%A4%87&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra=%7b)、计算机外设、医疗和工业设备、汽车、电视、机顶盒、视频游戏机、可穿戴产品和物联网应用等。伴随这些产品应用的迭代升级，国产MPU芯片进入高成长周期。

# 国产CPU

CPU作为计算机系统的核心和“大脑”，堪称国之重器，其自主创新是实现安全可控的核心。近年来，在国家集成电路产业政策和大基金投资等多重措施支持下，在云计算、大数据、物联网等产业的推动下，一批CPU设计企业逐渐成长起来。

CPU主要由大量的运算器、控制器、寄存器组成，它是对计算机的所有硬件资源（如存储器、 输入输出单元）进行控制调配、执行通用运算的核心硬件单元；同时，计算机系统中所有软件层的操作，最终都将通过指令系统映射为CPU的操作。

从硬件层面，CPU中的硬件系统主要是为了实现每一条指令的功能，解决指令之间的连接关系，因此指令系统是计算机硬件的语言系统，它决定了计算机的基本功能。而指令系统需要通过处理器核进行实现，最终形成芯片产品。

计算机的程序最终需要转化成“指令”才能在CPU上运行，所以采用的**指令集**对于CPU的设计尤为重要。按照采用的指令集，CPU可以分为**复杂指令集（CISC）和精简指令集（RISC）**两大类。x86架构是复杂指令集的代表，而Arm架构、MIPS架构、Alpha、PowerPC架构等是精简指令集的代表，各架构特点比较如下：



从软件层面，软件是由按一定规则组织起来的许多条指令组成，完成一定的数据运算或者事务处理功能。而操作系统是管理电脑硬件与软件资源的程序，能够在硬件管理中处于支配地位；应用软件是利用计算机解决某类问题而设计的程序的集合，需要依赖于操作系统的支持才能运行；只有具备操作系统等关键基础软件的开发能力，才能为搭建全新的基于该指令系统的应用软件生态提供支撑。

综上，CPU生态包含软硬件两个方面，从指令系统出发，硬件上通过IP核形成芯片，并最终用于板卡、整机厂商等不同领域的应用终端；软件上形成包括操作系统、编译器、Java、.NET 等基础软件，最终实现应用于政企、教育、能源、交通等不同领域的应用软件。CPU生态体系是硬件和软件的结合，是产业上下游交互的产物，因此生态壁垒一旦建立便是长期稳定牢固的。x86处理器起步较早，生态较其他处理器具有明显优势，x86处理器的服务器销售额占全部服务器销售额的比例约为91%，销售量占比超过97%，处于显著领先地位。

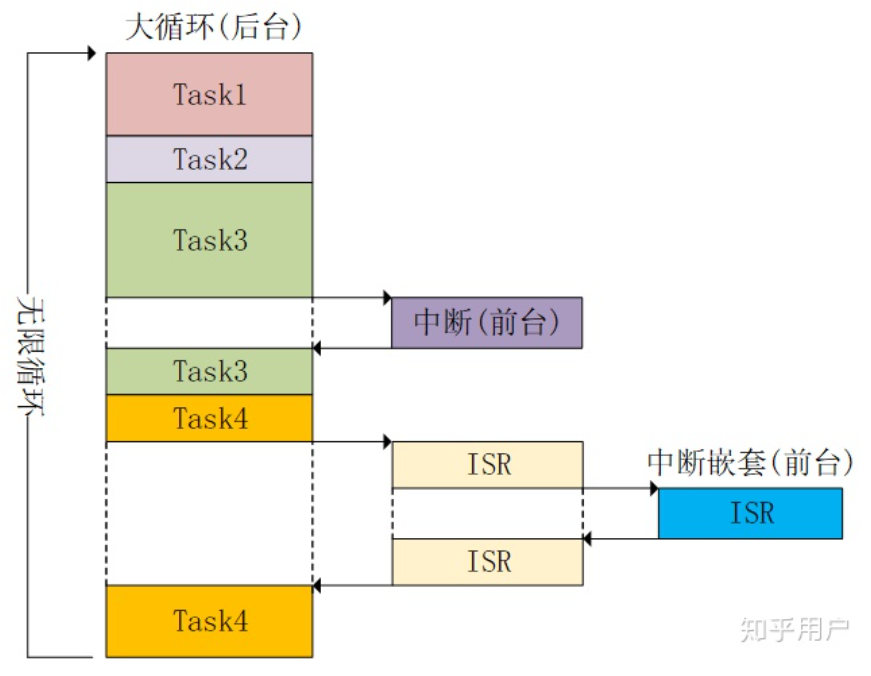
# 操作系统

一般**嵌入式系统**分为两种：**前后台系统**和**实时系统**。

## 前后台系统

早期嵌入式开发没有嵌入式操作系统的概念，直接操作**裸机**，在裸机上写程序，比如用51单片机基本就没有操作系统的概念。通常把程序分为两部分：**前台系统**和**后台系统**。

简单的小系统通常是前后台系统，这样的程序包括**一个死循环**和**若千个中断**服务程序:**应用程序**是一个无限循环，循环中调用API函数完成所需的操作，这个大循环就叫做**后台系统**。中断服务程序用于处理系统的**异步事件**，也就是**前台系统**。**前台是中断级，后台是任务级**。**前台任务优先级高于后台任务**。



在控制系统需求相对简单，控制逻辑不复杂的情况下，前后台系统因为代码设计逻辑简单，对ROM和RAM的占用低，可靠性高等特点得到了快速的普及和推广。但其不具备多任务管理和调度机制，不具备内存处理功能等。

从上面前后台系统的执行流程来看，当**软件控制对实时性要求提高**或者**软件逻辑过程过多**的时候，该前后台设计方案将不再符合要求。例如不执行完前面的逻辑（任务1,2），后面的任务3就无法执行。

同时如果**中断处理逻辑不合理**，也会导致后面的逻辑**实时性降低**。

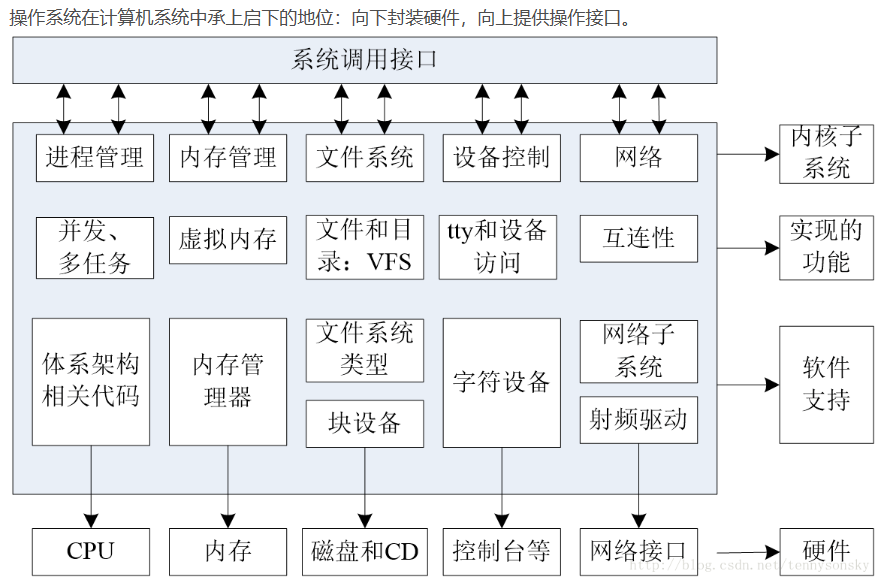
因此我们需要操作程序来作为一个**协调者**——操作系统；

## 操作系统引入

操作系统最主要的作用是协调资源、管理硬件（“抹平”硬件差异）：

* 操作系统可以为所有程序分配资源，一切程序要调用计算机资源都需要向操作系统申请权限，这样就能**最大程度利用计算机资源同时让各种程序不互相冲突**；
* 操作系统提供一系列**底层接口和上层标准**，“抹平”不同电脑之间的**硬件差异**，这样只要操作系统相同，硬件能达到软件运行的最低标准，理论上这个软件就能互通。

总体而言，虽然操作系统本身也会占用一部分计算机资源，但是总体上它让计算机**运行更为稳定**，同时也**减少了软件开发者的工作量**，因为程序猿只需要考虑操作系统的标准接口，而不需要考虑硬件系统的底层差异。



## 操作系统分类

按实时性操作系统分为2类：

实时操作系统：uCOS/VxWorks/RTLinux

非实时操作系统：Linux/Windows/OSX

### 实时操作系统

RTOS全称为：Real Time OS，就是实时操作系统，**强调的是：实时性**；

实时操作系统要求用户任务可以得到及时响应，多用于嵌入式设备；

实时操作系统又分为硬实时和软实时：

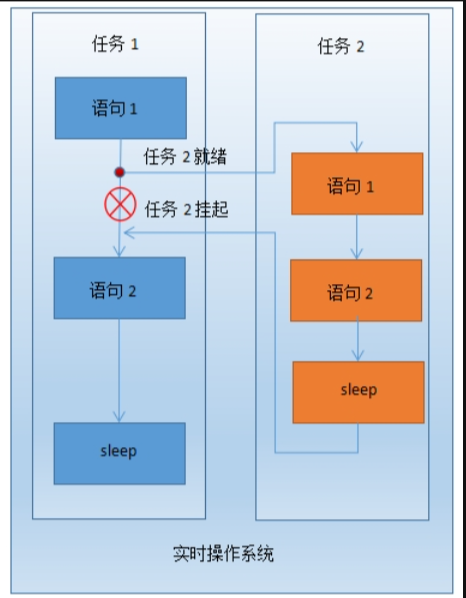
* **硬实时**要求在规定的时间内必须完成操作，硬实时系统**不允许超时**，否则会导致不可挽回的结果。
* 在**软实时**里面处理过程超时的后果就没有那么严格。

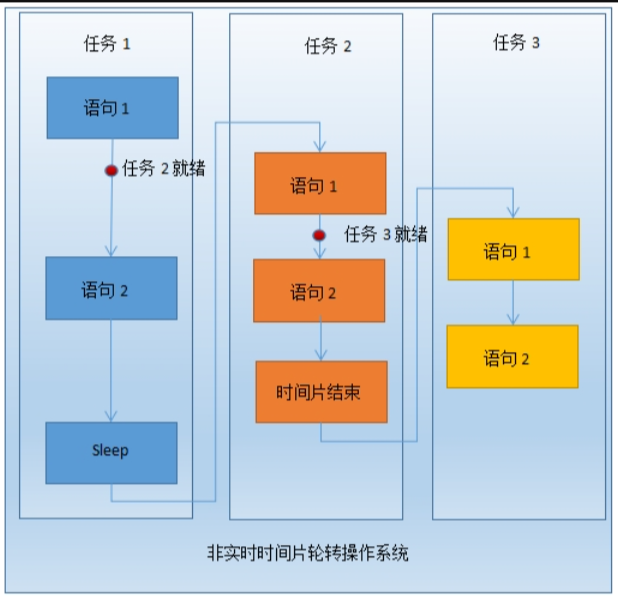
RTOS 运行的永远是**处于就绪态的最高优先级的那个任务**。

### 分实操作系统

通用型操作系统（如Windows、linux），更注重数据吞吐量而不是系统的响应时间；linux是分时操作系统，但随着它的升级，也有了实时调度算法的补丁。

### 区别





在上面的图中右边的任务优先级**高于**左边的任务。

实时操作系统：当优先级更高的任务2就绪的时候，即便任务1正在运行中，也必须**立刻交出CPU的使用权**，就跟中断一样，先执行任务2，等任务2执行完或者主动挂起(sleep)让出CPU的时候，任务1才能接着运行。

非实时时间片操作系统：即便高优先级的任务2就绪了，也**不能马上中断**低优先级任务1而得到执行，必须要等到低优先级任务1主动挂起(sleep)或者时间片结束才能得到执行；

区别在于何时调用Schedule；实时操作系统是优先级更高的任务就绪时，直接进行调度，像中断一样直接切换到优先级更高的任务，CPU可以抢占；而非实时时间片操作系统只有当当前任务执行完毕后，才调用Schedule，切换任务，CPU不可抢占；

实时操作系统**严格按照任务优先级抢占执行**，**快速的中断响应；**

## 航天操作系统

中国航天用的SpaceOS主要内容是仿造美国风河系统公司的VxWorks653（**653是产品名**，并非版本号）。

先解释为什么用这个系统不用Linux：

航天器的内存和CPU都非常弱，弱到什么程度呢：天宫一号的CPU是10MHz的，内存是2M，这种配置跑Linux比较费劲，虽然也并非不可能但要裁剪Linux内核确实太麻烦了。

而Linux的致命缺陷就是Linux**不是一个硬实时操作系统**。实时操作系统（RTOS）有一系列严格的定义：包括**严格按照任务优先级抢占执行**，**快速的中断响应**，对关中断的时间有非常严格的控制，Linux是一个民用的操作系统，其内核设计并没有考虑到这些点，所以Linux不适用于航天领域。

实际上，Linux的主要应用还是在**通用计算机**和**服务器**方面，在**工业控制、自动化**方面还有各种各样的操作系统：ucos，nuclues，threadX，greenhill，VxWorks等等。

有人提到了RT-Linux，这个说的简单点，就是底下是一层RTOS，上面是Linux，这种Linux复杂度太高，也不能裁剪的太小。

## 国内研究所操作系统应用

|  |  |  |  |
| --- | --- | --- | --- |
| **单位** | **操作系统** | **适航取证/公司** | **项目** |
| 631 | 天脉 | SOI#2 随机取证 |  |
| 618 | TTOS |  | 新舟700、AG600 |
| 615 | vxworks653 | SOI#4 随机取证 | C919 |
| 607 | vxworks7.0/vxworks6.9 |  |  |
| 飞控 | DEOS653 | 霍尼韦尔自研 | C919 |
| INTEGRITY | Green Hills | B787 |

国外处理器一般被认为安全性e-6；

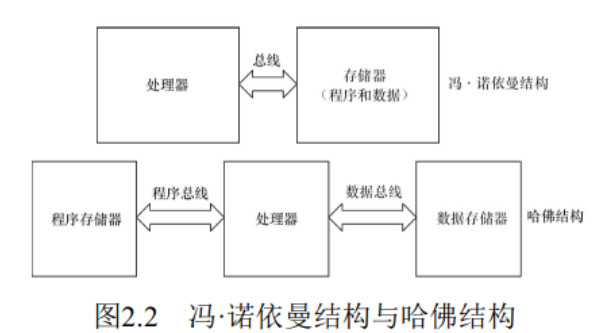
# 处理器

1. 处理器按**应用领域**可分为通用处理器(单片机，ARM)，数字处理器（DSP），其他专用处理器（FPGA）；

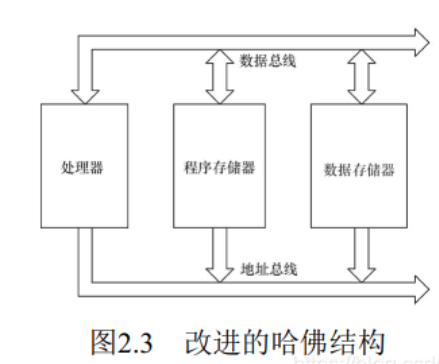
在当今通用处理器芯片大多数采用ARM架构并且多采用SOC的芯片设计方法，集成了各种功能模块（图形处理器，视频解码器，浮点协处理器，GPS，WIFI等），每一种功能都是由硬件描述语言设计程序，然后在Soc内由电路连接实现。

1. 中央处理器的**体系结构**可以分为两类， 一类为冯·诺依曼结构， 另一类为哈佛结构：

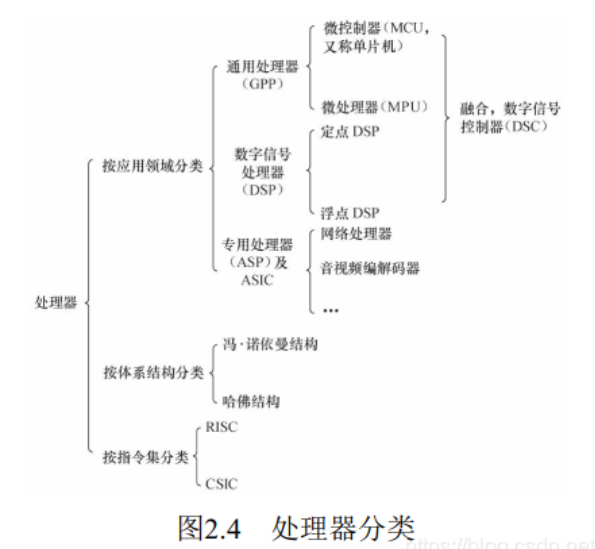
* **冯·诺依曼结构**也称普林斯顿结构， 是一种将**程序指令存储器**和**数据存储器合并**在一起的**存储器结构**。程序指令存储地址和数据存储地址指向同一个存储器的不同物理位置， 因此程**序指令和数据的宽度相同**。
* **哈佛结构**将**程序指令**和**数据分开**存储， 指令和数据可以有**不同的数据宽度**。 此外， 哈佛结构还采用了独立的程序总线和数据总线， 分别作为CPU与每个存储器之间的专用通信路径， 具有**较高的执行效率**。



改进的哈佛架构， 它具有独立的地址总线和数据总线， 两条总线由程序存储器和数据存储器**分时共用**。因此，改进的哈佛结构针对程序和数据，其实没有独立的总线，而是使用公用数据总线来完成程序存储模块或数据存储模块与CPU之间的数据传输，公用的地址总线来寻址程序和数据。

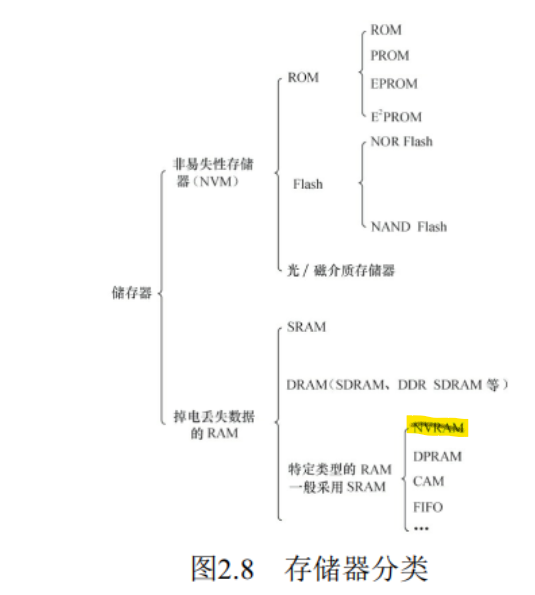


1. 从**指令集**的角度来讲，中央处理器也可以分为两类， 即RISC（**精简指令集**计算机） 和CISC（**复杂指令集**计算机）。CSIC强调增强指令的能力、 减少目标代码的数量，但是指令复杂， 指令周期长；而RISC强调尽量减少指令集、指令单周期执行，但是目标代码会更大。ARM、MIPS、PowerPC等CPU内核都采用了RISC指令集。目前，RISC和CSIC两者的融合非常明显。



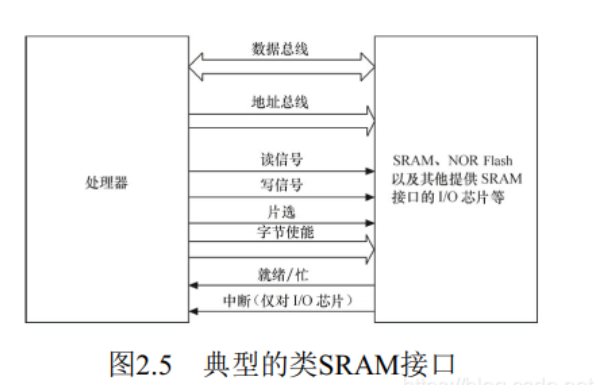
# 存储器

## 存储器分类



存储器主要可分为：

* 只读储存器（ROM），ROM还可再细分为：
* 不可编程ROM（MROM）；
* 可编程ROM（PROM）；
* 可擦除可编程ROM（EPROM）；
* 电可擦除可编程ROM（E2PROM），E2PROM完全可以用软件来擦写， 已经非常方便了。
* 闪存（Flash）：NOR（或非） 和NAND（与非） 是市场上两种主要的Flash闪存技术：
* NOR Flash和CPU的接口属于典型的**类SRAM接口**（如下图所示） ， **不需要增加额外**的控制电路。NOR Flash的特点是可芯片内执行（execute In Place,XIP），程序可以直接在NOR内运行。
* NAND Flash和CPU的接口必须由相应的**控制电路**进行转换， 当然也可以通过地址线或GPIO产生NAND Flash接口的信号。 NAND Flash以块方式进行访问， 不支持芯片内执行。
* 随机存取存储器（RAM）：速度快，价格高；易失存储器。RAM也可再分为：
* 静态RAM（SRAM）：SRAM是静态的，以触发器存储数据。只要供电它就会保持一个值，SRAM没有刷新周期。
* 动态RAM（DRAM）： DRAM以电荷形式进行存储， 数据存储在电容器中。由于电容器会因漏电而出现电荷丢失，所以DRAM器件需要**定期刷新**。
* 另外：RAM和NVRAM的区别在于，**掉电后RAM数据消失**，NVRAM数据保留：NVRAM(非易失性随机存取存储器)是指一种在断电后仍能保存数据的RAM。
* 光/磁介质储存器。



## NAND Flash与NOR Flash

NAND Flash较NOR Flash容量大，价格低；NAND Flash中每个块的最大擦写次数是100万次，而NOR的擦写次数是10万次； NAND Flash的擦除、 编程速度**远超过**NOR Flash。

由于Flash固有的电器特性，在读写数据过程中，偶然会产生1位或几位数据错误，即**位反转**，NAND Flash发生位反转的概率要**远大于**NOR Flash。**位反转无法避免**，因此，使用NAND Flash的同时，应采用错误探测/错误更正（**EDC/ECC**） 算法。

Flash还存在一个负载均衡的问题，不能老是在同一块位置进行擦除和写的动作，这样容易导致坏块。

值得一提的是， 目前NOR Flash可以使用SPI接口进行访问以节省引脚。 相对于传统的**并行**NOR Flash而言，SPI NOR Flash只需要6个引脚就能够实现单I/O、双I/O和4个I/O口的接口通信，有的SPI NOR Flash还支持DDR模式， 能进一步提高访问速度到80MB/s。

以上所述的各种ROM、Flash和磁介质存储器都属于非易失性存储器（NVM） 的范畴，掉电时信息不会丢失，而RAM则与此相反。

## 补充：单粒子翻转（Single Event Upset）

单粒子翻转是宇宙中单个高能粒子射入半导体器件灵敏区，使器件逻辑状态翻转的现象。

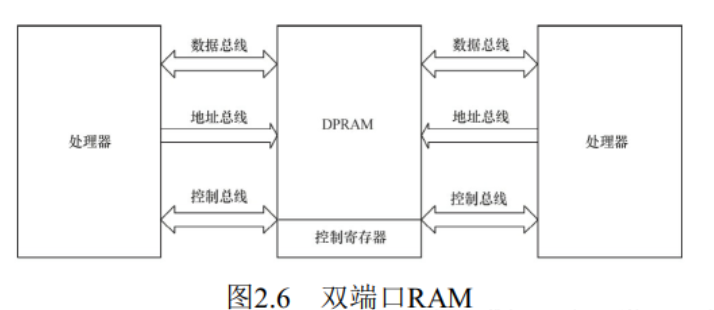
最容易发生单粒子翻转的是像**RAM**这种利用双稳态进行存储的器件，其次是CPU，再其次是其它的接口电路。随着芯片集成度的增加，发生SEU错误的可能性也随之增大。

缓解SEU的方案：

* 选取具有ECC功能的RAM器件；
* 选取基于Flash的存储器件;
* 通过软件设计ECC编码;
* 禁用处理器的cache

## 双端口RAM

双口RAM（DPRAM）的特点是可以通过**两个端口同时**访问，具有**两套**完全独立的数据总线、 地址总线和读写控制线， 通常用于两个处理器之间交互数据，如图所示。当一端被写入数据后，另一端可以通过轮询或中断获知，并读取其写入的数据。



DPRAM的优点是通信速度快、实时性强、接口简单，而且两边处理器都可主动进行数据传输。除了双端口RAM以外，目前IDT等芯片厂商还推出了多端口RAM，可以供3个以上的处理器互通数据。

## FIFO先进先出队列

FIFO存储器的特点是先进先出，进出有序，FIFO多用于**数据缓冲**。

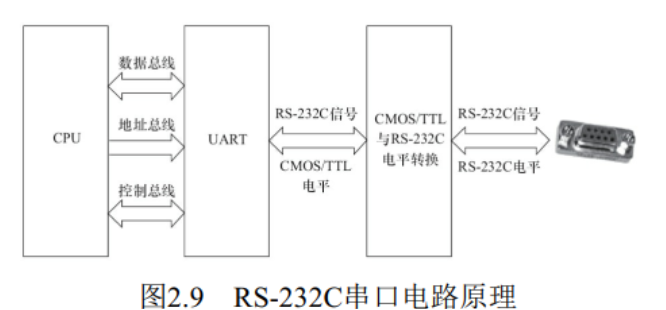
如果FIFO的区域共有n个字节，我们只能通过循环n次读取同一个地址才能将该片区域读出，不能指定偏移地址。对于有n个数据的FIFO，当循环读取m次之后，下一次读时会自动读取到第m+1个数据，这是由FIFO本身的特性决定的。

# 接口与总线

## 串口

最简单的RS-232C串口只需要连接RxD、 TxD、 SG（地）这3个信号， 并使用XON/XOFF软件流控。

组成一个RS-232C串口的硬件原理如图所示， 从CPU到连接器依次为CPU、 UART（通用异步接收器发送器， 作用是完成并/串转换） 、CMOS/TTL电平与RS-232C电平转换、 DB9/DB25或自定义连接器。



## I²C

I²C是内部整合电路的称呼，是一种串行通讯总线，使用多主从架构。

相应时序：当SCL稳定在高电平时，SDA由高向低的变化将产生一个开始位，而且由低到高变化，则产生一个停止位。（都有注主设备产生）

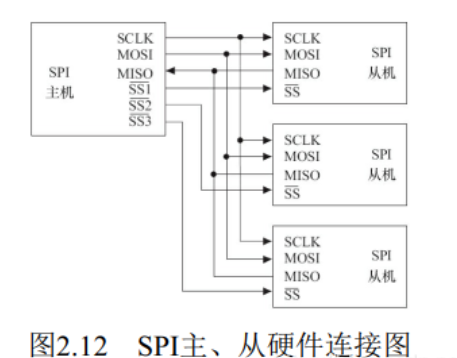
在选择设别时：主设备需要首先发送一个字节的地址信息，前7位代表地址信息，最后一位代表读写信息。

数据通信：在第9个上升沿来到之前，从设备应该发出一个ACK位。结束时，SCL稳定保持在高电平期间，SDA从低向高，产生停止信号。

## SPI

SPI（Serial Peripheral Interface，串行外设接口）总线系统是一种**同步串行**外设接口，它可以使CPU与各种外围设备以**串行**方式进行通信以交换信息。 一般主控SoC作为SPI的“主”， 而外设作为SPI的“从”。

SPI接口一般使用**4**条线： 串行**时钟线**（SCLK）、主入从出数据线**MISO**、 主出从入数据线**MOSI**和低电平有效的**从机选择线**SS。下图演示了1个主机连接3个SPI外设的硬件连接图。



在SPI总线的传输中， 片选SS信号是**低**电平有效的，当我们要与某外设通信的时候，需要将该外设上的SS线置低。

SPI模块为了和外设进行数据交换，根据外设工作要求，其输出串行同步时钟**极性**（CPOL）和**相位**（CPHA）可以进行配置。如果CPOL=0，串行同步时钟的空闲状态为低电平； 如果CPOL=1， 串行同步时钟的空闲状态为高电平。 如果CPHA=0，在串行同步时钟的第一个跳变沿（上升或下降）数据被采样；如果CPHA=1，在串行同步时钟的第二个跳变沿（上升或下降）数据被采样。

## USB

USB（通用串行总线）是Intel、Microsoft等厂商为解决计算机外设种类的日益增加与有限的主板**插槽和端口**之间的矛盾而于1995年提出的，它具有数据**传输率高**、**易扩展**、支持**即插即用**和**热插拔**的优点，目前已得到广泛应用。

USB 1.1包含全速和低速两种模式，低速方式的速率为1.5Mbit/s， 支持一些不需要很大数据吞吐量和很高实时性的设备，如鼠标等。全速模式为12Mbit/s， 可以外接速率更高的外设。

在**USB 2.0**中，增加了一种高速方式，数据传输率达到**480Mbit/s**，**半双工**， 可以满足**更高速**外设的需要。

**USB 3.0**（也被认为是Super Speed USB） 的最大传输带宽高达**5.0Gbit/s**（即640MB/s），**全双工**。

## 以太网接口

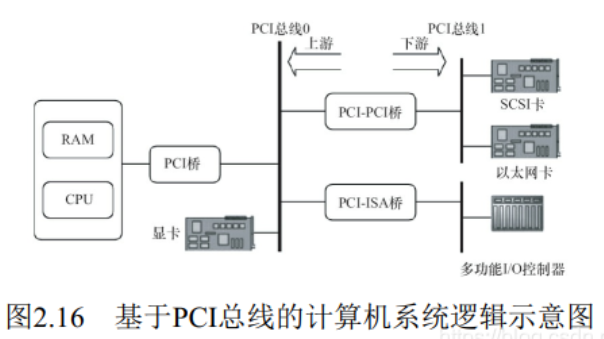
以太网接口由MAC(以太网媒体接入控制器)和PHY(物理接口收发器)组成。以太网MAC由IEEE802.3以太网标准定义，实现数据链路层。常用的MAC支持10Mbit/s和100Mbit/s两种速率。吉比特网是快速以太网的下一代，速度将高达1000Mbit/s。

## PCI和PCI-E

PCI（外围部件互连） 是一种**局部总线**，作为一种通用的总线接口标准，它在目前的计算机系统中得到了非常广泛应用。PCI总线具有如下特点：

* 数据总线为32位， 可扩充到64位；
* 可进行突发（Burst）模式传输。**突发方式**传输是指取得总线控制权后连续进行多个数据的传输。突发传输时，只需要给出目的地的首地址，访问第1个数据后，第2~n个数据会在首地址的基础上按一定规则自动寻址和传输。 与突发方式对应的是**单周期方式**，它在1个总线周期只传送1个数据；
* 总线操作与处理器—存储器子系统操作并行；
* 采用中央**集中式总线仲裁**；
* 支持全自动配置、资源分配，PCI卡内有设备信息寄存器组为系统提供卡的信息， 可实现**即插即用；**
* PCI总线规范**独立于**微处理器， 通用性好；
* PCI设备可以完全作为**主控设备控制总线**。

下图给出了一个典型的基于PCI总线的计算机系统逻辑示意图，系统的各个部分通过PCI总线和PCI-PCI桥连接在一起。 CPU和RAM通过PCI桥连接到PCI总线0（即主PCI总线），而具有PCI接口的显卡则可以直接连接到主PCI总线上。PCI-PCI桥是一个特殊的PCI设备，它负责将PCI总线0和PCI总线1（即从PCI主线）连接在一起， 通常PCI总线1称为PCI-PCI桥的下游（Downstream），而PCI总线0则称为PCI-PCI桥的上游（Upstream）。为了兼容旧的ISA总线标准， PCI总线还可以通过PCI-ISA桥来连接ISA总线，从而支持以前的ISA设备。



当PCI卡刚加电时，卡上配置空间即可以被访问。PCI配置空间保存着该卡工作时所需的所有信息，如厂家、卡功能、资源要求、处理能力、功能模块数量、主控卡能力等。通过对这个空间信息的读取与编程，可完成对PCI卡的配置。如图2.17所示，PCI配置空间共为256字节，主要包括如下信息：

制造商标识（Vendor ID）：由PCI组织分配给厂家。

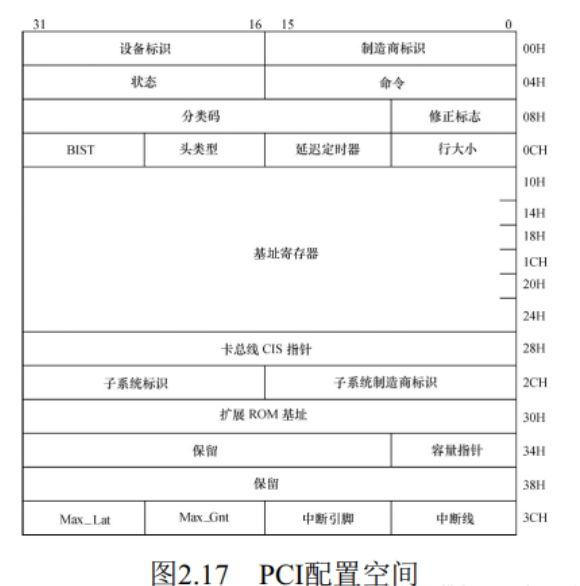
设备标识（Device ID）：按产品分类给本卡的编号。

分类码（Class Code）：本卡功能的分类码，如图卡、显示卡、解压卡等。

申请存储器空间：PCI卡内有存储器或以存储器编址的寄存器和I/O空间， 为使驱动程序和应用程序能访问它们，需申请CPU的一段存储区域以进行定位。 配置空间的**基地址**寄存器用于此目的。

申请I/O空间：配置空间中的基地址寄存器用来进行系统I/O空间的申请。

中断资源申请：配置空间中的中断引脚和中断线用来向系统申请中断资源。



PCI-E（PCI Express）是Intel公司提出的新一代的总线接口，PCI Express采用了目前业内流行的**点对点串行**连接，比起PCI以及更早的计算机总线的共享并行架构， 每个设备都有自己的**专用**连接，采用**串行**方式传输数据，**不需要向整个总线请求带宽**，并可以把数据**传输率**提高到一个很高的频率，达到PCI所不能提供的高带宽。

# CPLD和FPGA

CPLD由**完全可编程**的**与或门**阵列以及**宏单元**构成。与CPLD不同，FPGA（现场可编程门阵列）基于LUT（查找表）工艺。CPLD和FPGA的主要厂商有Altera,Xilinx和Lattice等，它们专门的开发流程，在设计阶段使用HDL编程。它们可实现许多复杂功能，如实现USART，I2C等I/O控制芯片，通信算法，音视频解码算法等。甚至还可以直接集成ARM等CPU内核和外围电路。对于驱动工程师而言，我们就直接把它看成由很多逻辑门（与 或 非）组成的可完成一系列功能的芯片。如果完成的功能是CPU，我们就直接把它看成是CPU。驱动工程师眼里的硬件要比IC设计师要宏观。

# 驱动

软件层面的驱动广义上就是指：这一段代码操作硬件去动，所以这一段代码就叫硬件的驱动程序。（本质上是电力提供了动力，而驱动程序提供了操作逻辑方法）。

狭义上驱动程序就是专指操作系统中用来操控硬件的**逻辑方法部分代码**。