

## 第2章

# 驱动设计的硬件基础

### 本章导读

本章讲述底层驱动工程师必备的硬件基础，给出了嵌入式系统硬件原理及分析方法的一个完整而简洁的全景视图。

2.1 节描述了微控制器、微处理器、数字信号处理器以及应用于特定领域的处理器各自的特点，分析了处理器的体系结构和指令集。

2.2 节对嵌入式系统中所使用的各类存储器与 CPU 的接口、应用领域及特点进行了归纳整理。

2.3 节分析了常见的外设接口与总线的工作方式，包括串口、I<sup>2</sup>C、SPI、USB、以太网接口、PCI 和 PCI-E、SD 和 SDIO 等。

嵌入式系统硬件电路中经常会使用 CPLD 和 FPGA，作为驱动工程师，我们不需要掌握 CPLD 和 FPGA 的开发方法，但是需要知道它们在电路中能完成什么工作，2.4 节讲解了这项内容。

2.5 ~ 2.7 节给出了在实际项目开发过程中硬件分析的方法，包括如何进行原理图分析、时序分析及如何快速地从芯片数据手册中获取有效信息。

2.8 节讲解了调试过程中常用仪器仪表的使用方法，涉及万用表、示波器和逻辑分析仪。

## 2.1 处理器

### 2.1.1 通用处理器

目前主流的通用处理器（GPP）多采用 SoC（片上系统）的芯片设计方法，集成了各种功能模块，每一种功能都是由硬件描述语言设计程序，然后在 SoC 内由电路实现的。在 SoC 中，每一个模块不是一个已经设计成熟的 ASIC 器件，而是利用芯片的一部分资源去实现某种传统的功能，将各种组件采用类似搭积木的方法组合在一起。

ARM 内核的设计技术被授权给数百家半导体厂商，做成不同的 SoC 芯片。ARM 的功耗很低，在当今最活跃的无线局域网、3G、手机终端、手持设备、有线网络通信设备等中应用非常广泛。至本书编写时，市面上绝大多数智能手机、平板电脑都使用 ARM SoC 作为主控

芯片。很多 ARM 主控芯片的集成度非常高,除了集成多核 ARM 以外,还可能集成图形处理器、视频编解码器、浮点协处理器、GPS、WiFi、蓝牙、基带、Camera 等一系列功能。比如,高通的 Snapdragon 810 就集成了如图 2.1 所示的各种模块。

主流的 ARM 移动处理芯片供应商包括高通 (Qualcomm)、三星 (Samsung)、英伟达 (Nvidia)、美满 (Marvell)、联发科 (MTK)、海思 (HiSilicon)、展讯 (Spreadtrum) 等。德州仪器 (TI)、博通 (Broadcom) 则已淡出手机芯片业务。

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。Intel 公司的中央处理器、ARM 的 ARM7、MIPS 公司的 MIPS 处理器采用了冯·诺依曼结构;而 AVR、ARM9、ARM10、ARM11 以及 Cortex A 系列等则采用了哈佛结构。

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数据宽度相同。而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 and 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。图 2.2 描述了冯·诺依曼结构和哈佛结构的区别。

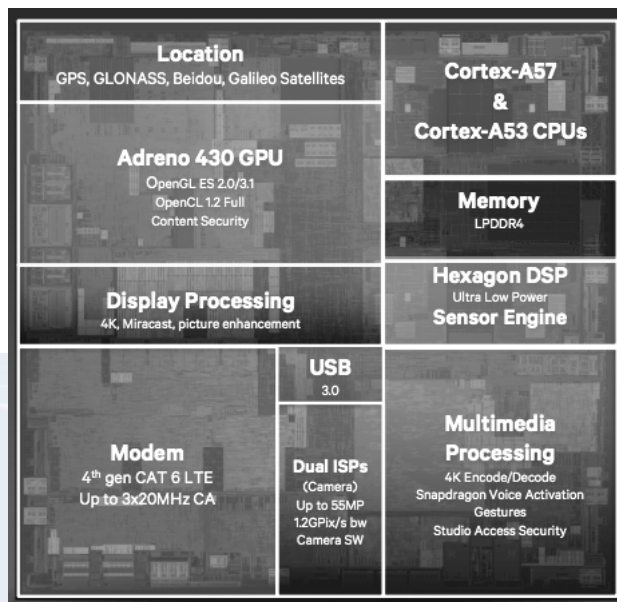


图 2.1 ARM SoC 范例: Snapdragon 810

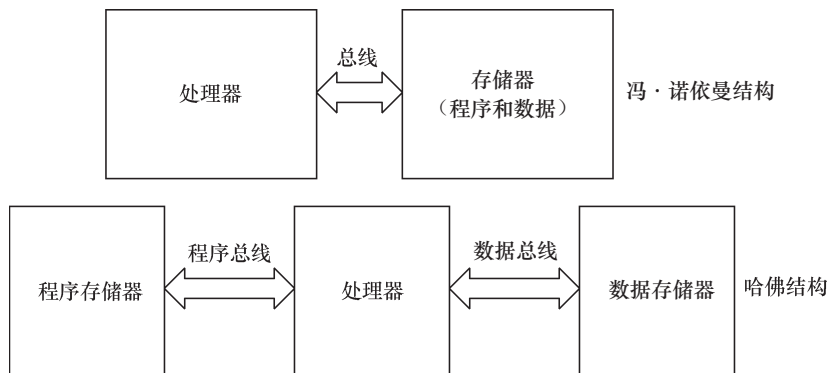


图 2.2 冯·诺依曼结构与哈佛结构

## 22 Linux设备驱动开发详解：基于最新的Linux 4.0内核

许多芯片采用的是如图 2.3 所示的改进的哈佛架构，它具有独立的地址总线 and 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。

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。

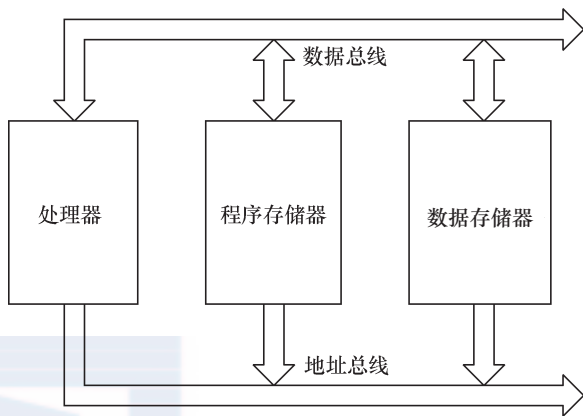


图 2.3 改进的哈佛结构

### 2.1.2 数字信号处理器

数字信号处理器（DSP）针对通信、图像、语音和视频处理等领域的算法而设计。它包含独立的硬件乘法器。DSP 的乘法指令一般在单周期内完成，且优化了卷积、数字滤波、FFT（快速傅里叶变换）、相关矩阵运算等算法中的大量重复乘法。

DSP 分为两类，一类是定点 DSP，另一类是浮点 DSP。浮点 DSP 的浮点运算用硬件来实现，可以在单周期内完成，因而其浮点运算处理速度高于定点 DSP。而定点 DSP 只能用定点运算模拟浮点运算。

德州仪器（TI）、美国模拟器件公司（ADI）是全球 DSP 的两大主要厂商。

TI 的 TMS320™ DSP 平台包含了功能不同的多个系列，如 2000 系列、3000 系列、4000 系列、5000 系列、6000 系列，工程师也习惯称其为 2x、3x、4x、5x、6x。2010 年 5 月，TI 已经宣布为其 C64x 系列数字信号处理器与多核片上系统提供 Linux 内核支持，以充分满足通信与关键任务基础设施、医疗诊断以及高性能测量测试等应用需求。TI 也推出了软件可编程多核 ARM + DSP SoC，即 KeyStone 多核 ARM+DSP 处理器，以满足医疗成像应用、任务关键应用、测试和自动化应用的需求。

ADI 主要有 16 位定点的 21xx 系列、32 位浮点的 SHARC 系列、从 SHARC 系列发展而来的 TigerSHARC 系列，以及高性能 16 位 DSP 信号处理能力与通用微控制器方便性相结合的 blackfin 系列等。ADI 的 blackfin 不含 MMU，完整支持 Linux，是没有 MMU 情况下 Linux 的典型示例，其官方网站为 <http://blackfin.uclinux.org>，目前 blackfin 的 Linux 开发保持了与 Linux mainline 的同步。

通用处理器和数字信号处理器也有相互融合以取长补短的趋势，如数字信号控制器（DSC）即为 MCU+DSP，ADI 的 blackfin 系列就属于 DSC。目前，芯片厂商也推出了许多 ARM+DSP 的双核以及多核处理器，如 TI 公司的 OMAP 4 平台就包括 4 个主要处理引擎：ARM Cortex-A9 MPCore、PowerVR SGX 540 GPU（Graphic Processing Unit）、C64x DSP 和 ISP（Image Signal Processor）。

除了上面讲述的通用微控制器和数字信号处理器外，还有一些针对特定领域而设计的专用处理器（ASP），它们都是针对一些特定应用而设计的，如用于 HDTV、ADSL、Cable Modem 等的专用处理器。

网络处理器是一种可编程器件，它应用于电信领域的各种任务，如包处理、协议分析、路由查找、声音 / 数据的汇聚、防火墙、QoS 等。网络处理器器件内部通常由若干个微码处理器和若干硬件协处理器组成，多个微码处理器在网络处理器内部并行处理，通过预先编制的微码来控制处理流程。而对于一些复杂的标准操作（如内存操作、路由表查找算法、QoS 的拥塞控制算法、流量调度算法等），则采用硬件协处理器来进一步提高处理性能，从而实现了业务灵活性和高性能的有机结合。

对于某些应用场合，使用 ASIC（专用集成电路）往往是低成本且高性能的方案。ASIC 专门针对特定应用而设计，不具备也不需要灵活的编程能力。使用 ASIC 完成同样的功能往往比直接使用 CPU 资源或 CPLD（复杂可编程逻辑器件）/FPGA（现场可编程门阵列）来得更廉价且高效。

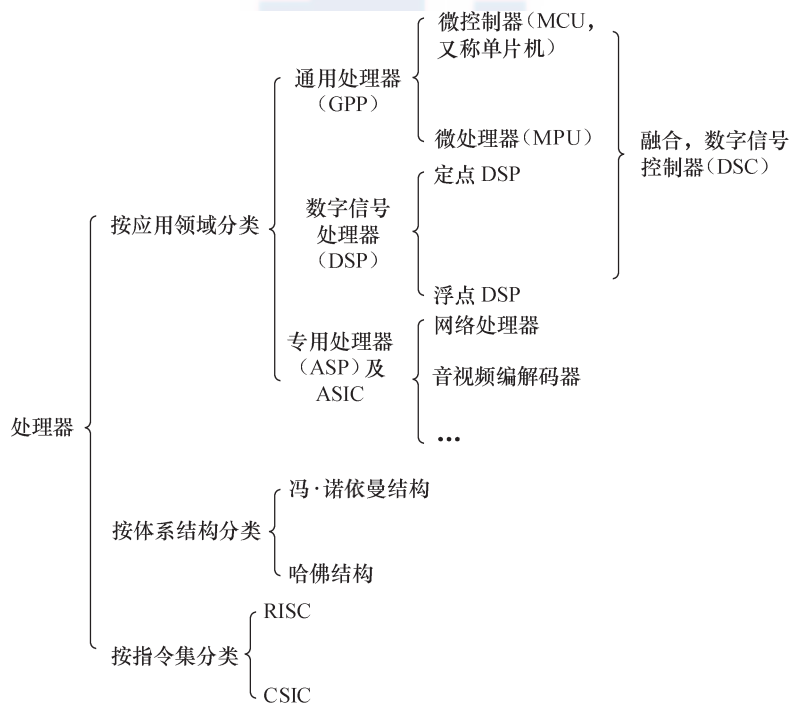


图 2.4 处理器分类

## 24 Linux设备驱动开发详解：基于最新的Linux 4.0内核

在实际项目的硬件方案中，往往会根据应用的需求选择通用处理器、数字信号处理器、特定领域处理器、CPLD/FPGA 或 ASIC 之一的解决方案，在复杂的系统中，这些芯片可能会同时存在，协同合作，各自发挥自己的长处。如在一款智能手机中，可使用 MCU 处理图形用户界面和用户的按键输入并运行多任务操作系统，使用 DSP 进行音视频编解码，而在射频方面则采用 ASIC。

综合 2.1 节的内容，可得出如图 2.4 所示的处理器分类。

## 2.2 存储器

存储器主要可分类为只读存储器（ROM）、闪存（Flash）、随机存取存储器（RAM）、光/磁介质存储器。

ROM 还可再细分为不可编程 ROM、可编程 ROM（PROM）、可擦除可编程 ROM（EPROM）和电可擦除可编程 ROM（E<sup>2</sup>PROM），E<sup>2</sup>PROM 完全可以用软件来擦写，已经非常方便了。

NOR（或非）和 NAND（与非）是市场上两种主要的 Flash 闪存技术。Intel 于 1988 年首先开发出 NOR Flash 技术，彻底改变了原先由 EPROM 和 EEPROM 一统天下的局面。紧接着，1989 年，东芝公司发表了 NAND Flash 结构，每位的成本被大大降低。

NOR Flash 和 CPU 的接口属于典型的类 SRAM 接口（如图 2.5 所示），不需要增加额外的控制电路。NOR Flash 的特点是可在芯片内执行（eXecute In Place, XIP），程序可以直接在 NOR 内运行。而 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。

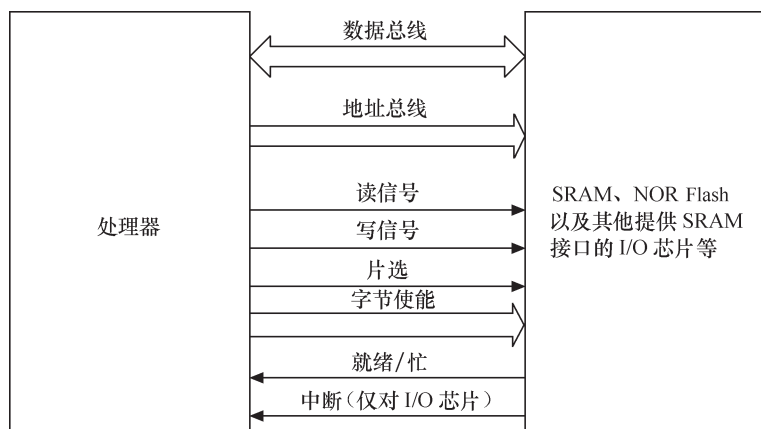


图 2.5 典型的类 SRAM 接口



公共闪存接口 (Common Flash Interface, CFI) 是一个从 NOR Flash 器件中读取数据的公开、标准接口。它可以使系统软件查询已安装的 Flash 器件的各种参数, 包括器件阵列结构参数、电气和时间参数以及器件支持的功能等。如果芯片不支持 CFI, 就需使用 JEDEC (Joint Electron Device Engineering Council, 电子电器设备联合会) 了。JEDEC 规范的 NOR 则无法直接通过命令来读出容量等信息, 需要读出制造商 ID 和设备 ID, 以确定 Flash 的大小。

与 NOR Flash 的类 SRAM 接口不同, 一个 NAND Flash 的接口主要包含如下信号。

- I/O 总线: 地址、指令和数据通过这组总线传输, 一般为 8 位或 16 位。
- 芯片启动 (Chip Enable, CE#): 如果没有检测到 CE 信号, NAND 器件就保持待机模式, 不对任何控制信号做出响应。
- 写使能 (Write Enable, WE#): WE# 负责将数据、地址或指令写入 NAND 之中。
- 读使能 (Read Enable, RE#): RE# 允许数据输出。
- 指令锁存使能 (Command Latch Enable, CLE): 当 CLE 为高电平时, 在 WE# 信号的上升沿, 指令将被锁存到 NAND 指令寄存器中。
- 地址锁存使能 (Address Latch Enable, ALE): 当 ALE 为高电平时, 在 WE# 信号的上升沿, 地址将被锁存到 NAND 地址寄存器中。
- 就绪 / 忙 (Ready/Busy, R/B#): 如果 NAND 器件忙, R/B# 信号将变为低电平。该信号是漏极开路, 需要采用上拉电阻。

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, 偶尔会产生 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 的编程原理都是只能将 1 写为 0, 而不能将 0 写为 1。因此在 Flash 编程之前, 必须将对应的块擦除, 而擦除的过程就是把所有位都写为 1 的过程, 块内的所有字节变为 0xFF。另外, 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。

IDE (Integrated Drive Electronics) 接口可连接硬盘控制器或光驱, IDE 接口的信号与 SRAM 类似。人们通常也把 IDE 接口称为 ATA (Advanced Technology Attachment) 接口, 不过, 从技术角度而言, 这并不准确。其实, ATA 接口发展至今, 已经经历了 ATA-1 (IDE)、ATA-2 (Enhanced IDE/Fast ATA, EIDE)、ATA-3 (FastATA-2)、Ultra ATA、Ultra ATA/33、Ultra ATA/66、Ultra ATA/100 及 Serial ATA (SATA) 的发展过程。

很多 SoC 集成了一个 eFuse 电编程熔丝作为 OTP (One-Time Programmable, 一次性可编

程) 存储器。eFuse 可以通过计算机对芯片内部的参数和功能进行配置, 这一般是在芯片出厂的时候已经设置好了。

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。

RAM 也可再分为静态 RAM (SRAM) 和动态 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。SRAM 是静态的, 只要供电它就会保持一个值, SRAM 没有刷新周期。每个 SRAM 存储单元由 6 个晶体管组成, 而 DRAM 存储单元由 1 个晶体管和 1 个电容器组成。

通常所说的 SDRAM、DDR SDRAM 皆属于 DRAM 的范畴, 它们采用与 CPU 外存控制器同步的时钟工作 (注意, 不是与 CPU 的工作频率一致)。与 SDRAM 相比, DDR SDRAM 同时利用了时钟脉冲的上升沿和下降沿传输数据, 因此在时钟频率不变的情况下, 数据传输频率加倍。此外, 还存在使用 RSL (Rambus Signaling Level, Rambus 发信电平) 技术的 RDRAM (Rambus DRAM) 和 Direct RDRAM。

针对许多特定场合的应用, 嵌入式系统中往往还使用了一些特定类型的 RAM。

### 1. DPRAM: 双端口 RAM

DPRAM 的特点是可以两个端口同时访问, 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 2.6 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。由于双 CPU 同时访问 DPRAM 时的仲裁逻辑电路集成在 DPRAM 内部, 所以需要硬件工程师设计的电路原理比较简单。

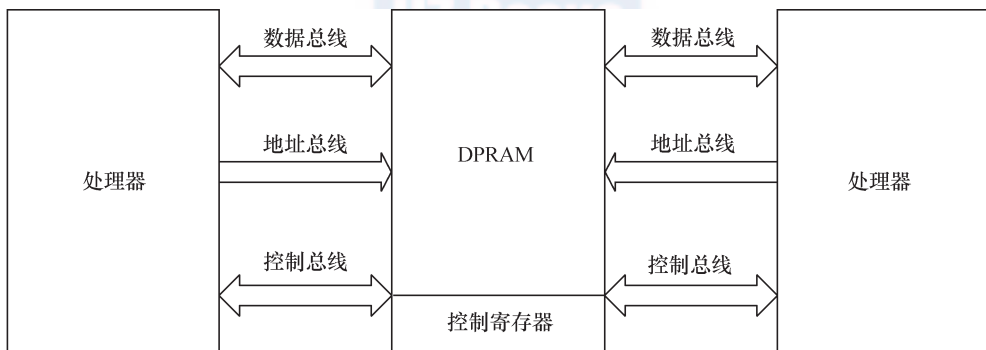


图 2.6 双端口 RAM

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。

### 2. CAM: 内容寻址 RAM

CAM 是以内容进行寻址的存储器, 是一种特殊的存储阵列 RAM, 它的主要工作机制就是同时将一个输入数据项与存储在 CAM 中的所有数据项自动进行比较, 判别该输入数据项

与 CAM 中存储的数据项是否相匹配，并输出该数据项对应的匹配信息。

如图 2.7 所示，在 CAM 中，输入的是所要查询的数据，输出的是数据地址和匹配标志。若匹配（即搜寻到数据），则输出数据地址。CAM 用于数据检索的优势是软件无法比拟的，它可以极大地提高系统性能。

### 3. 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。FIFO 和 DPRAM 类似，具有两个访问

端口，但是 FIFO 两边的端口并不对等，某一时刻只能设置为一边作为输入，一边作为输出。

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。

总结 2.2 节的内容，可得出如图 2.8 所示的存储器分类。

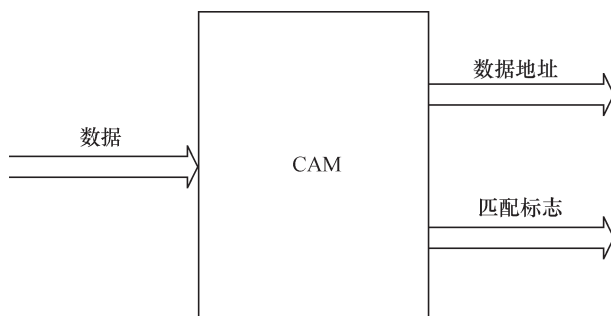


图 2.7 CAM 的输入与输出

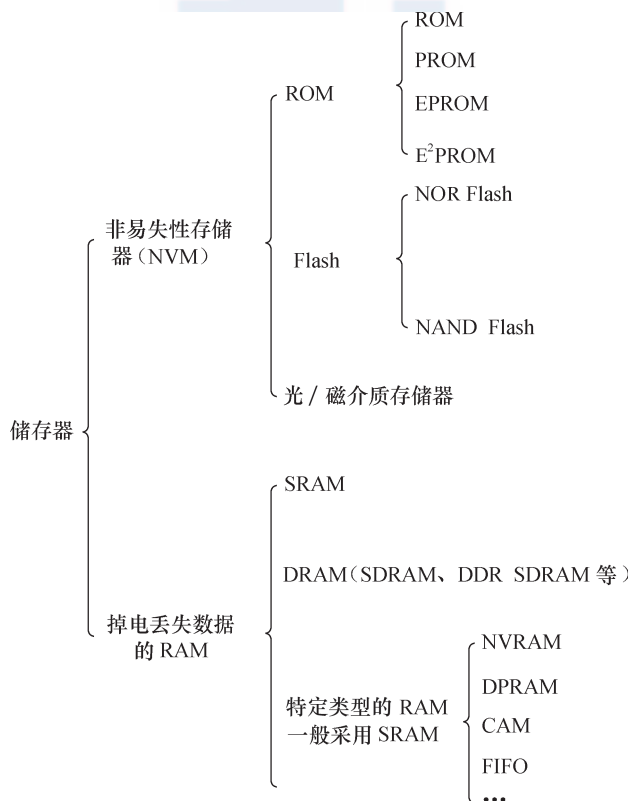


图 2.8 存储器分类



## 2.3 接口与总线

### 2.3.1 串口

RS-232、RS-422 与 RS-485 都是串行数据接口标准，最初都是由电子工业协会（EIA）制订并发布的。

RS-232 在 1962 年发布，命名为 EIA-232-E。之后发布的 RS-422 定义了一种平衡通信接口，它是一种单机发送、多机接收的单向、平衡传输规范，被命名为 TIA/EIA-422-A 标准。RS-422 改进了 RS-232 通信距离短、速率低的缺点。为进一步扩展应用范围，EIA 又于 1983 年在 RS-422 的基础上制定了 RS-485 标准，增加了多点、双向通信能力，即允许多个发送器连接到同一条总线上，同时增加了发送器的驱动能力和冲突保护特性，并扩展了总线共模范围，被命名为 TIA/EIA-485-A 标准。

1969 年发布的 RS-232 修改版 RS-232C 是嵌入式系统中应用最广泛的串行接口，它为连接 DTE（数据终端设备）与 DCE（数据通信设备）而制定。RS-232C 标准接口有 25 条线（4 条数据线、11 条控制线、3 条定时线、7 条备用和未定义线），常用的只有 9 根，它们是 RTS/CTS（请求发送 / 清除发送流控制）、RxD/TxD（数据收发）、DSR/DTR（数据终端就绪 / 数据设置就绪流控制）、DCD（数据载波检测，也称 RLSD，即接收线信号检出）、Ringing-RI（振铃指示）、SG（信号地）信号。RTS/CTS、RxD / TxD、DSR/DTR 等信号的定义如下。

- RTS：用来表示 DTE 请求 DCE 发送数据，当终端要发送数据时，使该信号有效。
- CTS：用来表示 DCE 准备好接收 DTE 发来的数据，是对 RTS 的响应信号。
- RxD：DTE 通过 RxD 接收从 DCE 发来的串行数据。
- TxD：DTE 通过 TxD 将串行数据发送到 DCE。
- DSR：有效（ON 状态）表明 DCE 可以使用。
- DTR：有效（ON 状态）表明 DTE 可以使用。
- DCD：当本地 DCE 设备收到对方 DCE 设备送来的载波信号时，使 DCD 有效，通知 DTE 准备接收，并且由 DCE 将接收到的载波信号解调为数字信号，经 RxD 线送给 DTE。
- Ringing-RI：当调制解调器收到交换台送来的振铃呼叫信号时，使该信号有效（ON 状态），通知终端，已被呼叫。

最简单的 RS-232C 串口只需要连接 RxD、TxD、SG 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 2.9 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。

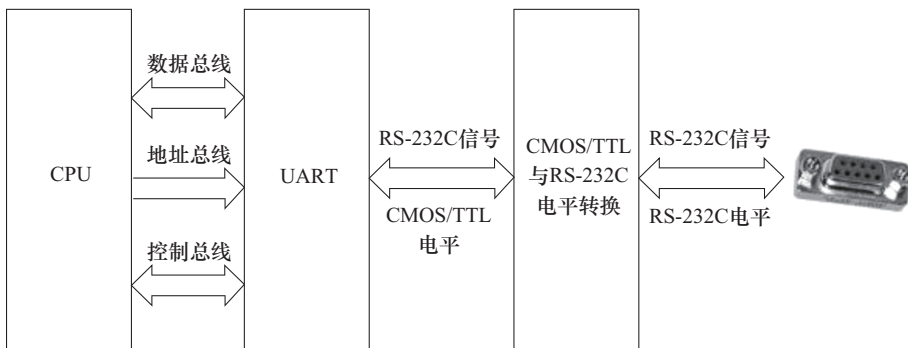


图 2.9 RS-232C 串口电路原理

### 2.3.2 I<sup>2</sup>C

I<sup>2</sup>C（内置集成电路）总线是由 Philips 公司开发的两线式串行总线，产生于 20 世纪 80 年代，用于连接微控制器及其外围设备。I<sup>2</sup>C 总线简单而有效，占用的 PCB（印制电路板）空间很小，芯片引脚数量少，设计成本低。I<sup>2</sup>C 总线支持多主控（Multi-Mastering）模式，任何能够进行发送和接收的设备都可以成为主设备。主控能够控制数据的传输和时钟频率，在任意时刻只能有一个主控。

组成 I<sup>2</sup>C 总线的两个信号为数据线 SDA 和时钟 SCL。为了避免总线信号的混乱，要求各设备连接到总线的输出端必须是开漏输出或集电极开路输出的结构。总线空闲时，上拉电阻使 SDA 和 SCL 线都保持高电平。根据开漏输出或集电极开路输出信号的“线与”逻辑，I<sup>2</sup>C 总线上任意器件输出低电平都会使相应总线上的信号线变低。



“线与”逻辑指的是两个或两个以上的输出直接互连就可以实现“与”的逻辑功能，只有输出端是开漏（对于 CMOS 器件）输出或集电极开路（对于 TTL 器件）输出时才满足此条件。工程师一般以“OC 门”简称开漏或集电极开路。

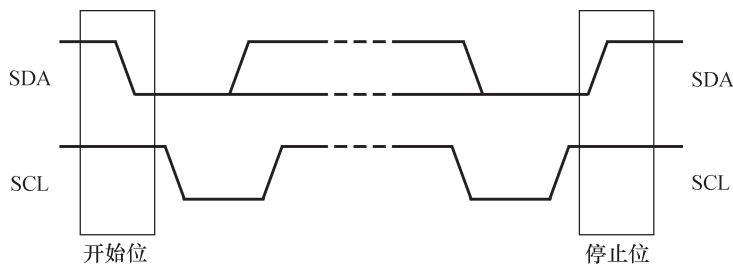
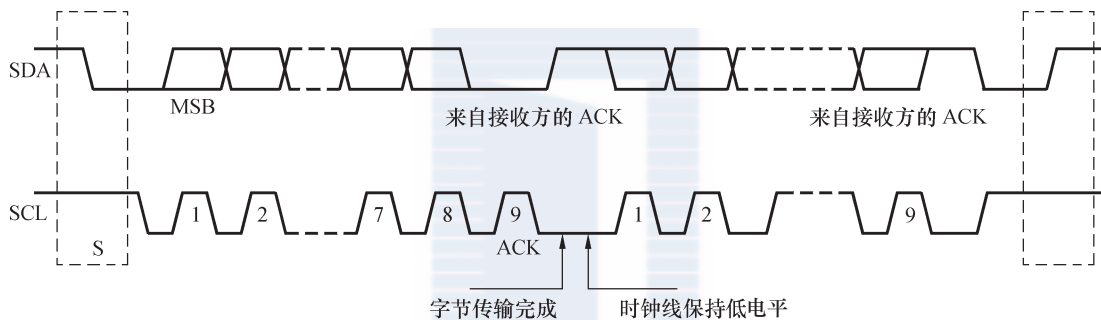
I<sup>2</sup>C 设备上的串行数据线 SDA 接口电路是双向的，输出电路用于向总线上发送数据，输入电路用于接收总线上的数据。同样地，串行时钟线 SCL 也是双向的，作为控制总线数据传送的主机要通过 SCL 输出电路发送时钟信号，并检测总线上 SCL 上的电平以决定什么时候发下一个时钟脉冲电平；作为接收主机命令的从设备需按总线上 SCL 的信号发送或接收 SDA 上的信号，它也可以向 SCL 线发出低电平信号以延长总线时钟信号周期。

当 SCL 稳定在高电平时，SDA 由高到低的变化将产生一个开始位，而由低到高的变化则产生一个停止位，如图 2.10 所示。

开始位和停止位都由 I<sup>2</sup>C 主设备产生。在选择从设备时，如果从设备采用 7 位地址，则主设备在发起传输过程前，需先发送 1 字节的地址信息，前 7 位为设备地址，最后 1 位为读写标志。之后，每次传输的数据也是 1 字节，从 MSB 开始传输。每个字节传完后，在 SCL

## 30 Linux设备驱动开发详解：基于最新的Linux 4.0内核

的第9个上升沿到来之前，接收方应该发出1个ACK位。SCL上的时钟脉冲由I<sup>2</sup>C主控方发出，在第8个时钟周期之后，主控方应该释放SDA，I<sup>2</sup>C总线的时序如图2.11所示。

图 2.10 I<sup>2</sup>C 总线的开始位和停止位图 2.11 I<sup>2</sup>C 总线的时序

### 2.3.3 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）、主机输入/从机输出数据线 MISO、主机输出/从机输入数据线 MOSI 和低电平有效的从机选择线 SS（在不同的文献里，也常称为 nCS、CS、CSB、CSN、nSS、STE、SYNC 等）。图 2.12 演示了 1 个主机连接 3 个 SPI 外设的硬件连接图。

如图 2.13 所示，在 SPI 总线的传输中，SS 信号是低电平有效的，当我们要与

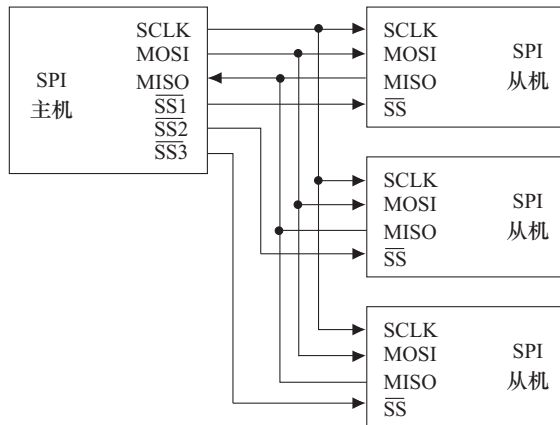


图 2.12 SPI 主、从硬件连接图

某外设通信的时候, 需要将该外设上的 SS 线置低。此外, 特别要注意 SPI 从设备支持的 SPI 总线最高时钟频率 (决定了 SCK 的频率) 以及外设的 CPHA、CPOL 模式, 这决定了数据与时钟之间的偏移、采样的时刻以及触发的边沿是上升沿还是下降沿。

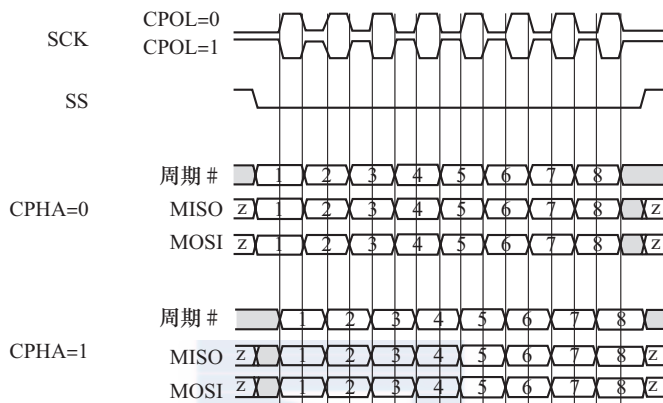


图 2.13 SPI 总线的时序

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。

#### 2.3.4 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。而 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。

USB 2.0 总线的机械连接非常简单, 采用 4 芯的屏蔽线, 一对差分线 (D+、D-) 传送信号, 另一对 (VBUS、电源地) 传送 +5V 的直流电。USB 3.0 线缆则设计了 8 条内部线路, 除 VBUS、电源地之外, 其余 3 对均为数据传输线路。其中保留了 D+ 与 D- 这两条兼容 USB 2.0 的线路, 新增了 SSRX 与 SSTX 专为 USB 3.0 所设的线路。

在嵌入式系统中, 电路板若需要挂接 USB 设备, 则需提供 USB 主机 (Host) 控制器和

## 32 Linux设备驱动开发详解：基于最新的Linux 4.0内核

连接器；若电路板需要作为 USB 设备，则需提供 USB 设备适配器和连接器。目前，大多数 SoC 集成了 USB 主机控制器（以连接 USB 外设）和设备适配器（以将本嵌入式系统作为其他计算机系统的 USB 外设，如手机充当 U 盘）。由 USB 主机、设备和 Hub 组成的 USB 系统的物理拓扑结构如图 2.14 所示。

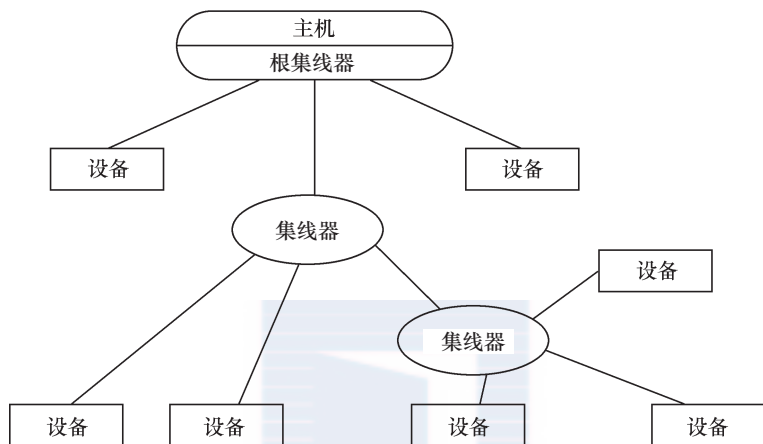


图 2.14 USB 的物理拓扑结构

每一个 USB 设备会有一个或者多个逻辑连接点在里面，每个连接点叫端点。USB 提供了多种传输方式以适应各种设备的需要，一个端点可以选择如下一种传输方式。

### 1. 控制 (Control) 传输方式

控制传输是双向传输，数据量通常较小，主要用来进行查询、配置和给 USB 设备发送通用命令。所有 USB 设备必须支持标准请求 (Standard Request)，控制传输方式和端点 0。

### 2. 同步 (Isochronous) 传输方式

同步传输提供了确定的带宽和间隔时间，它用于时间要求严格并具有较强的容错性的流数据传输，或者用于要求恒定数据传送率的即时应用。例如进行语音业务传输时，使用同步传输方式是很好的选择。同步传输也常称为“Streaming Real-time”传输。

### 3. 中断 (Interrupt) 传输方式

中断方式传送是单向的，对于 USB 主机而言，只有输入。中断传输方式主要用于定时查询设备是否有中断数据要传送，该传输方式应用在少量分散的、不可预测的数据传输场合，键盘、游戏杆和鼠标属于这一类型。

### 4. 批量 (Bulk) 传输方式

批量传输主要应用在没有带宽、间隔时间要求的批量数据的传送和接收中，它要求保证传输。打印机和扫描仪属于这种类型。

而 USB 3.0 则增加了一种 Bulk Streams 传输模式，USB 2.0 的 Bulk 模式只支持 1 个数据流，而 Bulk Streams 传输模式则可以支持多个数据流，每个数据流被分配一个 Stream ID



(SID), 每个 SID 与一个主机缓冲区对应。

在 USB 架构中, 集线器负责检测设备的连接和断开, 利用其中断 IN 端点 (Interrupt IN Endpoint) 来向主机报告。一旦获悉有新设备连接上来, 主机就会发送一系列请求给设备所挂载的集线器, 再由集线器建立起一条连接主机和设备之间的通信通道。然后主机以控制传输的方式, 通过端点 0 对设备发送各种请求, 设备收到主机发来的请求后回复相应的信息, 进行枚举 (Enumerate) 操作。因此 USB 总线具备热插拔的能力。

### 2.3.5 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 IEEE 802.3 以太网标准定义, 实现了数据链路层。常用的 MAC 支持 10Mbit/s 或 100Mbit/s 两种速率。吉比特以太网 (也称为千兆位以太网) 是快速以太网的下一代技术, 将网速提高到了 1000 Mbit/s。千兆位以太网以 IEEE 802.3z 和 802.3ab 发布, 作为 IEEE 802.3 标准的补充。

MAC 和 PHY 之间采用 MII (媒体独立接口) 连接, 它是 IEEE-802.3 定义的以太网行业标准, 包括 1 个数据接口与 MAC 和 PHY 之间的 1 个管理接口。数据接口包括分别用于发送和接收的两条独立信道, 每条信道都有自己的数据、时钟和控制信号, MII 数据接口总共需要 16 个信号。MII 管理接口包含两个信号, 一个是时钟信号, 另一个是数据信号。通过管理接口, 上层能监视和控制 PHY。

一个以太网接口的硬件电路原理如图 2.15 所示, 从 CPU 到最终接口依次为 CPU、MAC、PHY、以太网隔离变压器、RJ45 插座。以太网隔离变压器是以太网收发芯片与连接器之间的磁性组件, 在其两者之间起着信号传输、阻抗匹配、波形修复、信号杂波抑制和高电压隔离作用。

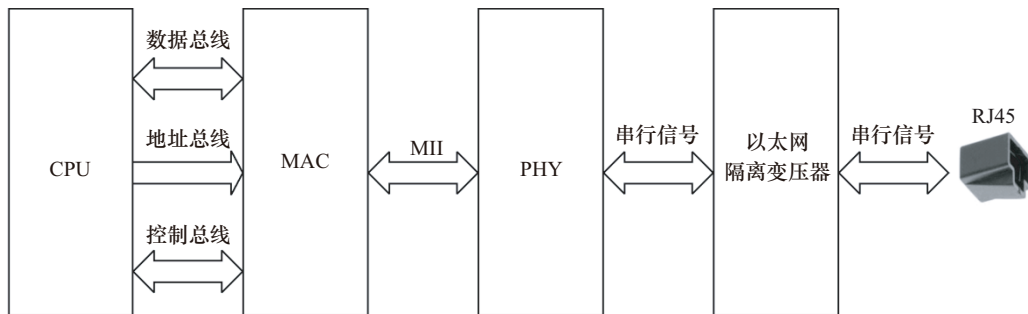


图 2.15 以太网接口的硬件电路原理

许多处理器内部集成了 MAC 或同时集成了 MAC 和 PHY, 另有许多以太网控制芯片也集成了 MAC 和 PHY。

### 2.3.6 PCI 和 PCI-E

PCI（外围部件互连）是由 Intel 于 1991 年推出的一种局部总线，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。

图 2.16 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。

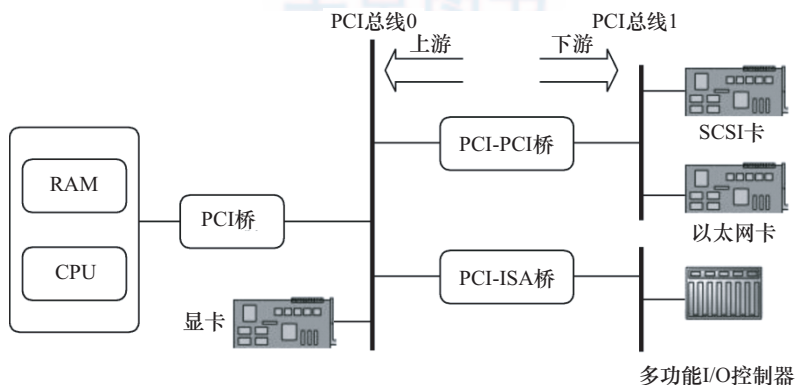


图 2.16 基于 PCI 总线的计算机系统逻辑示意图

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。偏移 3Dh 处为中断引脚寄存器，其值表明 PCI 设备使用了哪一个中断引脚，对应关系为 1—INTA#、2—INTB#、3—INTC#、4—INTD#。

31		16		15		0		
设备标识				制造商标识				00H
状态				命令				04H
分类码						修正标志		08H
BIST		头类型		延迟定时器		行大小		0CH
基址寄存器								10H
								14H
								18H
								1CH
								20H
								24H
卡总线 CIS 指针								28H
子系统标识				子系统制造商标识				2CH
扩展 ROM 基址								30H
保留						容量指针		34H
保留								38H
Max_Lat		Max_Gnt		中断引脚		中断线		3CH

图 2.17 PCI 配置空间

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

## 36 Linux设备驱动开发详解：基于最新的Linux 4.0内核

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。

PCI Express 在软件层面上兼容目前的 PCI 技术和设备，支持 PCI 设备和内存模组的初始化，也就是说无须推倒目前的驱动程序、操作系统，就可以支持 PCI Express 设备。

### 2.3.7 SD 和 SDIO

SD (Secure Digital) 是一种关于 Flash 存储卡的标准，也就是一般常见的 SD 记忆卡，在设计上与 MMC (Multi-Media Card) 保持了兼容。SDHC (SD High Capacity) 是大容量 SD 卡，支持的最大容量为 32GB。2009 年发布的 SDXC (SD eXtended Capacity) 则支持最大 2TB 大小的容量。

SDIO (Secure Digital Input and Output Card, 安全数字输入输出卡) 在 SD 标准的基础上，定义了除存储卡以外的外设接口。SDIO 主要有两类应用——可移动和不可移动。不可移动设备遵循相同的电气标准，但不要求符合物理标准。现在已经有非常多的手机或者手持装置都支持 SDIO 的功能，以连接 WiFi、蓝牙、GPS 等模块。

一般情况下，芯片内部集成的 SD 控制器同时支持 MMC、SD 卡，又支持 SDIO 卡，但是 SD 和 SDIO 的协议还是有不一样的地方，支持的命令也会有不同。

SD/SDIO 的传输模式有：

- SPI 模式
- 1 位模式
- 4 位模式

表 2.1 显示了 SDIO 接口的引脚定义。其中 CLK 为时钟引脚，每个时钟周期传输一个命令或数据位；CMD 是命令引脚，命令在 CMD 线上串行传输，是双向半双工的（命令从主机到从卡，而命令的响应是从卡发送到主机）；DAT[0]~DAT[3] 为数据线引脚；在 SPI 模式中，第 8 脚位被当成中断信号。图 2.18 给出了一个 SDIO 单模块读、写的典型时序。

表 2.1 SDIO 接口引脚定义

引脚 #	SD 4 位 模式		SD 1 位 模式		SPI 模式	
1	CD/DAT[3]	数据线 3	N/C	未使用	CS	片选
2	CMD	命令线	CMD	命令线	DI	数据输入
3	VSS1	地	VSS1	地	VSS1	地
4	VDD	电源电压	VDD	电源电压	VDD	电源电压
5	CLK	时钟	CLK	时钟	SCLK	时钟
6	Vss2	地	Vss2	地	Vss2	地
7	DAT[0]	数据线 0	DATA	数据线	DO	数据输出
8	DAT[1]	数据线 1 / 中断	IRQ	中断	IRQ	中断
9	DAT[2]	数据线 2 / 读等待	RW	读等待	NC	未使用

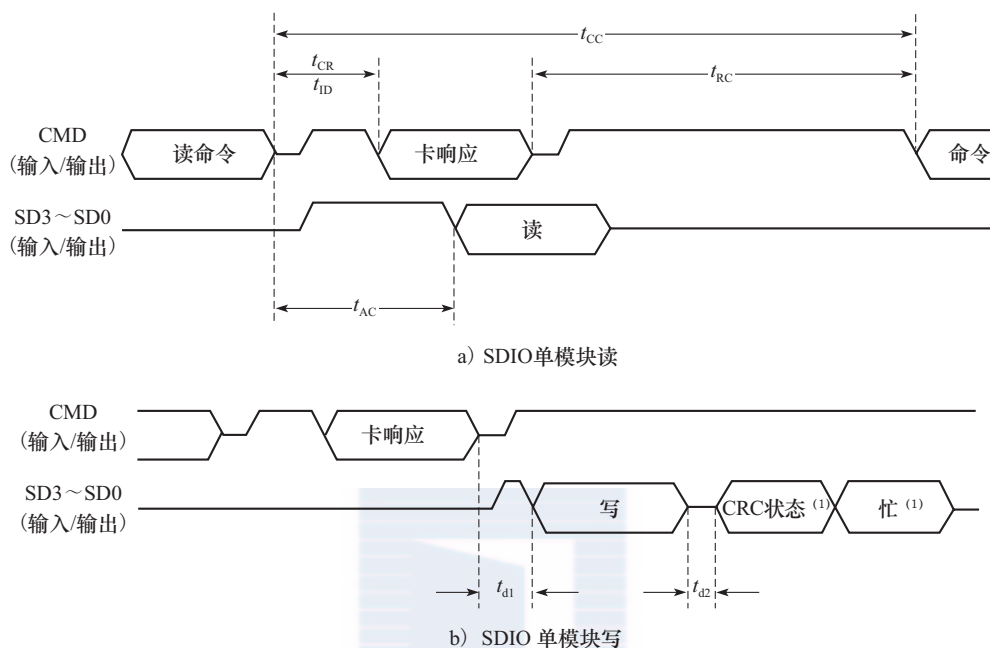


图 2.18 SDIO 单模块读、写的典型时序

eMMC (Embedded Multi Media Card) 是当前移动设备本地存储的主流解决方案，目的在于简化手机存储器的设计。eMMC 就是 NAND Flash、闪存控制芯片和标准接口封装的集合，它把 NAND 和控制芯片直接封装在一起成为一个多芯片封装 (Multi-Chip Package, MCP) 芯片。eMMC 支持 DAT[0]~DAT[7] 8 位的数据线。上电或者复位后，默认处于 1 位模式，只使用 DAT[0]，后续可以配置为 4 位或者 8 位模式。

## 2.4 CPLD 和 FPGA

CPLD (复杂可编程逻辑器件) 由完全可编程的与或门阵列以及宏单元构成。

CPLD 中的基本逻辑单元是宏单元，宏单元由一些“与或”阵列加上触发器构成，其中“与或”阵列完成组合逻辑功能，触发器完成时序逻辑功能。宏单元中与阵列的输出称为乘积项，其数量标示着 CPLD 的容量。乘积项阵列实际上就是一个“与或”阵列，每一个交叉点都是一个可编程熔丝，如果导通就是实现“与”逻辑。在“与”阵列后一般还有一个“或”阵列，用以完成最小逻辑表达式中的“或”关系。图 2.19 所示为非常典型的 CPLD 的单个宏单元结构。

图 2.20 给出了一个典型 CPLD 的整体结构。这个 CPLD 由 LAB (逻辑阵列模块，由多个宏单元组成) 通过 PIA (可编程互连阵列) 互连组成，而 CPLD 与外部的接口则由 I/O 控制模块提供。



38 Linux设备驱动开发详解：基于最新的Linux 4.0内核

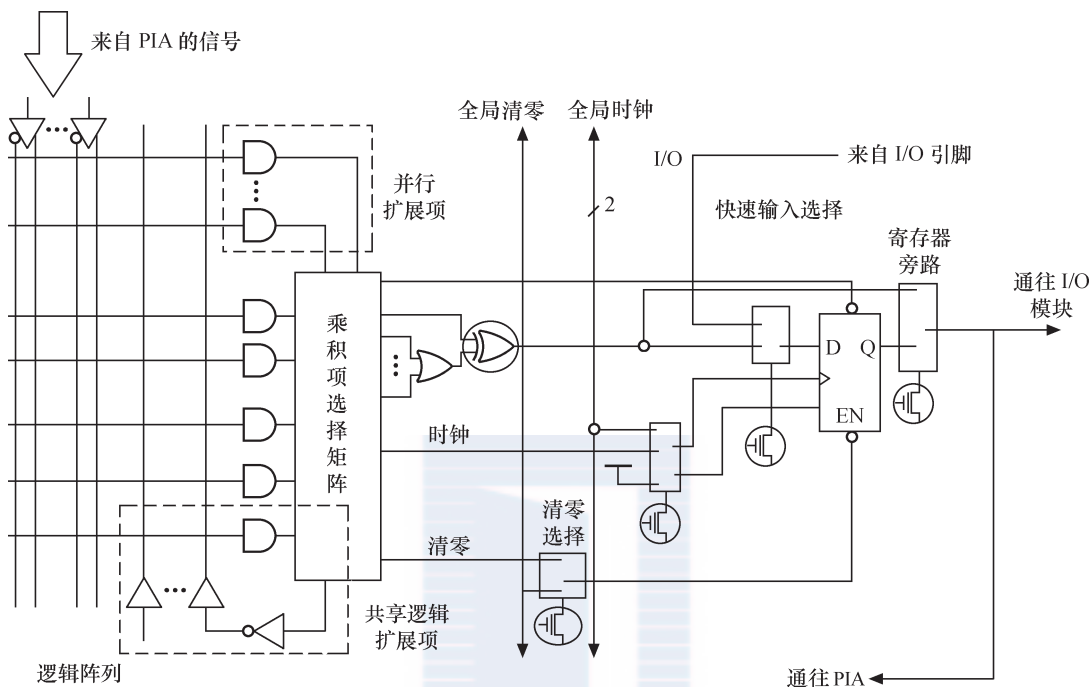


图 2.19 典型的 CPLD 的单个宏单元结构

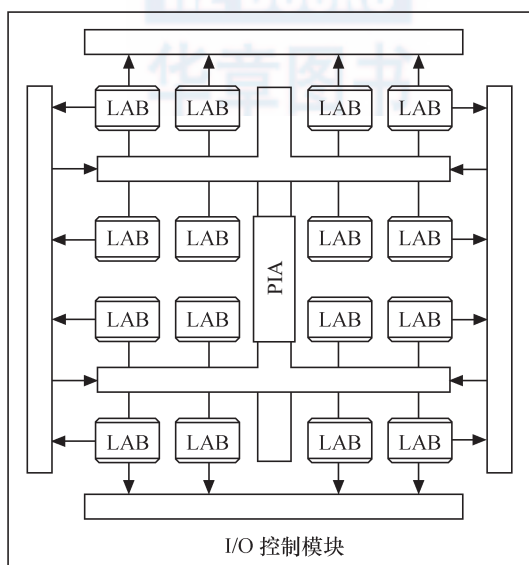


图 2.20 典型的 CPLD 整体架构

图 2.20 中宏单元的输出会经 I/O 控制块送至 I/O 引脚, I/O 控制块控制每一个 I/O 引脚的工作模式, 决定其为输入、输出还是双向引脚, 并决定其三态输出的使能端控制。

与 CPLD 不同, FPGA (现场可编程门阵列) 基于 LUT (查找表) 工艺。查找表本质上是一片 RAM, 当用户通过原理图或 HDL (硬件描述语言) 描述了一个逻辑电路以后, FPGA 开发软件会自动计算逻辑电路所有可能的结果, 并把结果事先写入 RAM。这样, 输入一组信号进行逻辑运算就等于输入一个地址进行查表以输出对应地址的内容。

图 2.21 所示为一个典型 FPGA 的内部结构。这个 FPGA 由 IOC (输入/输出控制模块)、EAB (嵌入式阵列块)、LAB 和快速通道互连构成。

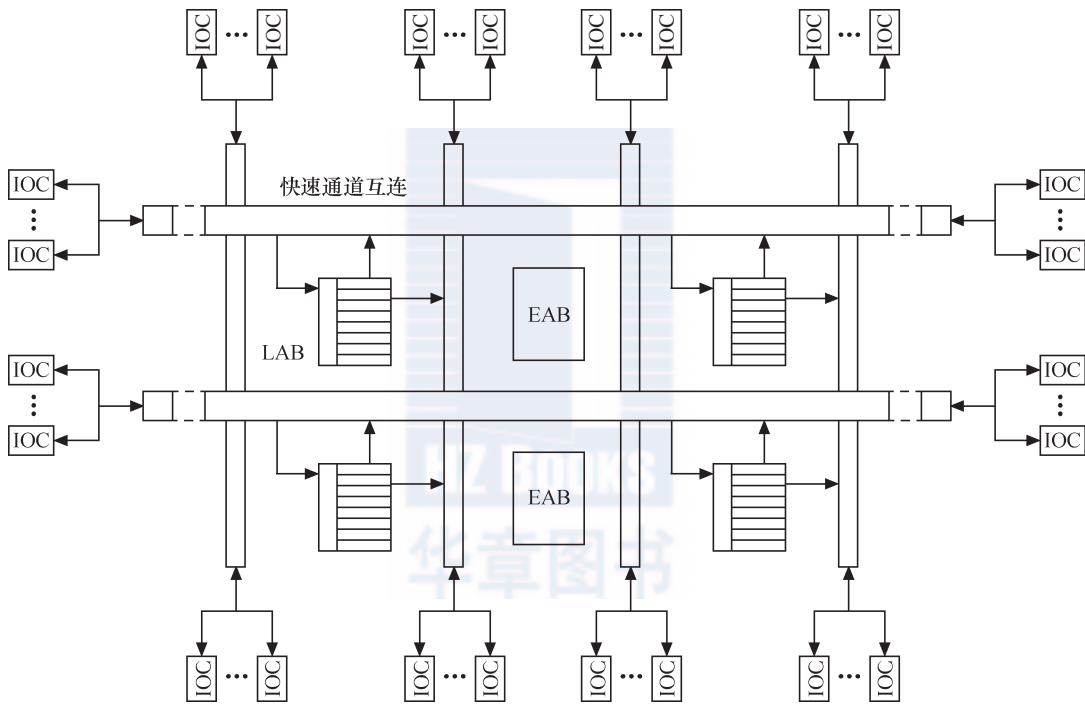


图 2.21 典型的 FPGA 内部结构

IOC 是内部信号到 I/O 引脚的接口, 它位于快速通道的行和列的末端, 每个 IOC 包含一个双向 I/O 缓冲器和一个既可作为输入寄存器也可作为输出寄存器的触发器。

EAB (嵌入式存储块) 是一种输入输出端带有寄存器的非常灵活的 RAM。EAB 不仅可以用作存储器, 还可以事先写入查表值以用来构成如乘法器、纠错逻辑等电路。当用于 RAM 时, EAB 可配制成 8 位、4 位、2 位和 1 位长度的数据格式。

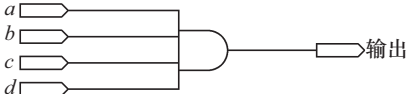
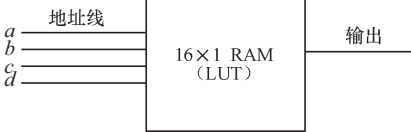
LAB 主要用于逻辑电路设计, 一个 LAB 包括多个 LE (逻辑单元), 每个 LE 包括组合逻辑及一个可编程触发器。一系列 LAB 构成的逻辑阵列可实现普通逻辑功能, 如计数器、加法器、状态机等。

## 40 Linux设备驱动开发详解：基于最新的Linux 4.0内核

器件内部信号的互连和器件引出端之间的信号互连由快速通道连线提供，快速通道遍布于整个 FPGA 器件中，是一系列水平和垂直走向的连续式布线通道。

表 2.2 所示为一个 4 输入 LUT 的实际逻辑电路与 LUT 实现方式的对应关系。

表 2.2 实际逻辑电路与查找表的实现

LUT 的实际逻辑电路		LUT 的实现方式	
			
abcd 输入	逻辑输出	地 址	RAM 中存储的内容
0000	0	0000	0
0001	0	0001	0
...	0	...	0
1111	1	1111	1

CPLD 和 FPGA 的主要厂商有 Altera、Xilinx 和 Lattice 等，它们采用专门的开发流程，在设计阶段使用 HDL（如 VHDL、Verilog HDL）编程。它们可以实现许多复杂的功能，如实现 UART、I<sup>2</sup>C 等 I/O 控制芯片、通信算法、音视频编解码算法等，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，我们只需要这样看待 CPLD 和 FPGA：如果它完成的是特定的接口和控制功能，我们就直接把它当成由很多逻辑门（与、非、或、D 触发器）组成的可完成一系列时序逻辑和组合逻辑的 ASIC；如果它完成的是 CPU 的功能，我们就直接把它当成 CPU。驱动工程师眼里的硬件比 IC 设计师要宏观。

值得一提的是，Xilinx 公司还推出了 ZYNQ 芯片，内部同时集成了两个 Cortex™-A9 ARM 多处理器子系统和可编程逻辑 FPGA，同时可编程逻辑可由用户配置。

## 2.5 原理图分析

原理图分析的含义是指通过阅读电路板的原理图获得各种存储器、外设所使用的硬件资源、接口和引脚连接关系。若要整体理解整个电路板的硬件组成，原理图的分析方法是以主 CPU 为中心向存储器和外设辐射，步骤如下。

1) 阅读 CPU 部分，获知 CPU 的哪些片选、中断和集成的外设控制器在使用，列出这些元素 a、b、c、…。

CPU 引脚比较多时，芯片可能会被分成几个模块并单独画在原理图的不同页上，这

时应该把相应的部分都分析到位。

2) 对第1步中列出的元素,从原理图中对应的外设和存储器电路中分析出实际的使用情况。

硬件原理图中包含如下元素。

- 符号 (symbol)。符号描述芯片的外围引脚以及引脚的信号,对于复杂的芯片,可能会被分割为几个符号。在符号中,一般把属于同一个信号群的引脚排列在一起。图 2.22 所示为 NOR Flash AM29LV160DB 和 NAND Flash K9F2G08 的符号。

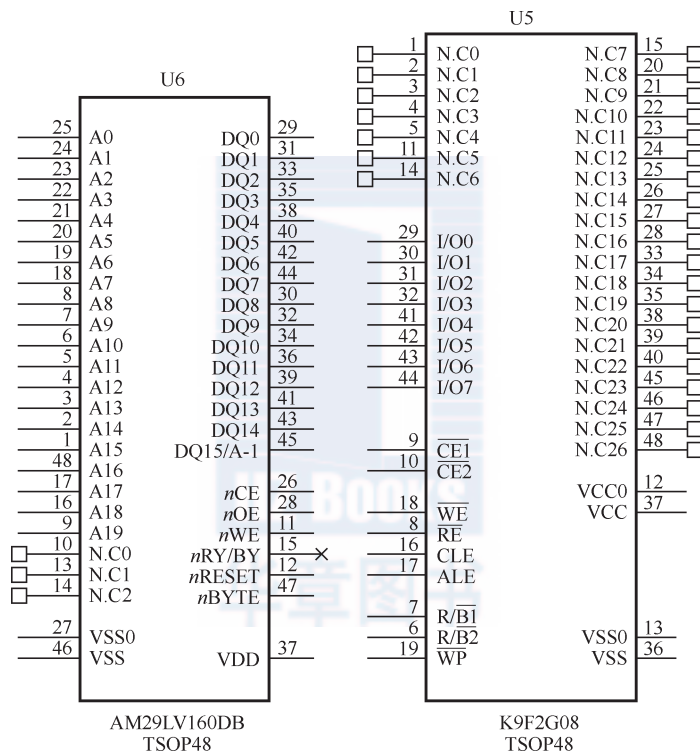


图 2.22 原理图中的符号

- 网络 (net)。描述芯片、接插件和分离元器件引脚之间的互连关系,每个网络需要根据信号的定义赋予一个合适的名字,如果没有给网络取名字,EDA 软件会自动添加一个默认的网络名。添加网络后的 AM29LV160DB 如图 2.23 所示。
- 描述。原理图中会添加一些文字来辅助描述原理图(类似源代码中的注释),如每页页脚会有该页的功能描述,对重要的信号,在原理图的相应符号和网络中也会附带文字说明。图 2.24 中给出了原理图中的描述示例。

## 42 Linux设备驱动开发详解：基于最新的Linux 4.0内核

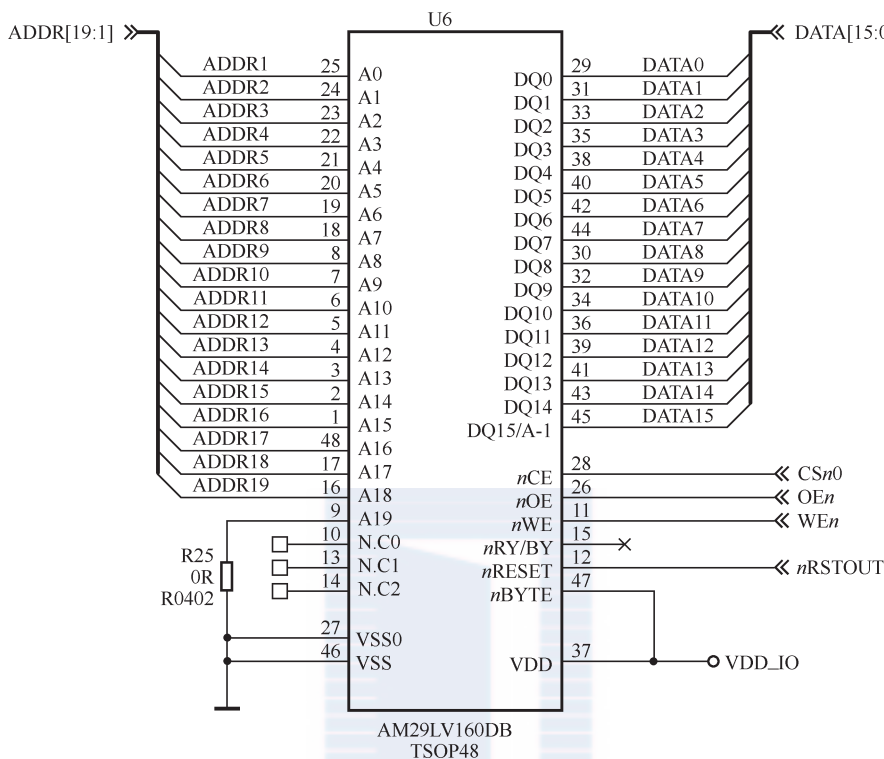


图 2.23 原理图中的网络

### LDDD3 Evaluation Board Schematics

Revision	Data	Description
V1	2014.3.26	

图 2.24 原理图中的描述示例

## 2.6 硬件时序分析

### 2.6.1 时序分析的概念

驱动工程师一般不需要分析硬件的时序，但是鉴于许多企业内驱动工程师还需要承担电路板调试的任务，因此，掌握时序分析的方法也就比较必要了。

对驱动工程师或硬件工程师而言，时序分析的意思是让芯片之间的访问满足芯片数据手



册中时序图信号有效的先后顺序、采样建立时间 (Setup Time) 和保持时间 (Hold Time) 的要求, 在电路板工作不正常的时候, 准确地定位时序方面的问题。

建立时间是指在触发器的时钟信号边沿到来以前, 数据已经保持稳定不变的时间, 如果建立时间不够, 数据将不能在这个时钟边沿被打入触发器; 保持时间是指在触发器的时钟信号边沿到来以后, 数据还需稳定不变的时间, 如果保持时间不够, 数据同样不能被打入触发器。如图 2.25 所示, 数据稳定传输必须满足建立时间和保持时间的要求, 当然, 在一些情况下, 建立时间和保持时间的值可以为零。

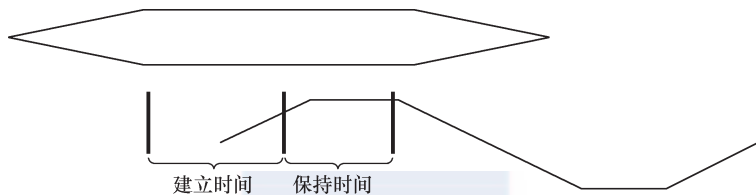


图 2.25 建立时间和保持时间

### 2.6.2 典型的硬件时序

最典型的硬件时序是 SRAM 的读写时序, 在读 / 写过程中涉及的信号包括地址、数据、片选、读 / 写、字节使能和就绪 / 忙。对于一个 16 位、32 位 (甚至 64 位) 的 SRAM, 字节使能表明哪些字节被读写。

图 2.26 给出了 SRAM 的读时序, 写时序与此相似。首先, 地址总线上输出要读 (写) 的地址, 然后发出 SRAM 片选信号, 接着输出读 (写) 信号, 之后读 (写) 信号要经历数个等待周期。当 SRAM 读 (写) 速度比较慢时, 等待周期可以由 MCU 的相应寄存器设置, 也可以通过设备就绪 / 忙 (如图 2.27 中的  $nWait$ ) 向 CPU 报告, 这样, 读写过程中会自动添加等待周期。

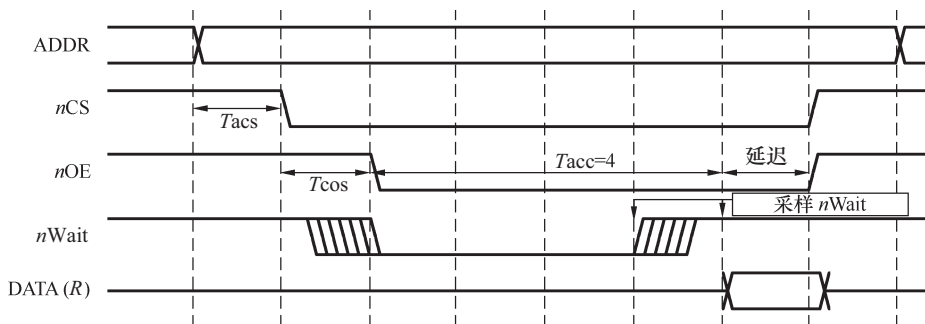


图 2.26 SRAM 读时序图

NOR Flash 和许多外设控制芯片都使用了类似 SRAM 的访问时序, 因此, 牢固掌握这个时序意义重大。一般, 在芯片数据手册给出的时序图中, 会给出图中各段时间的含义和要求, 真实的电路板必须满足芯片数据手册中描述的建立时间和保持时间的最小要求。

## 2.7 芯片数据手册阅读方法

芯片数据手册往往长达数百页，甚至上千页，而且全部是英文，从头到尾不加区分地阅读需要花费非常长的时间，而且不一定能获取对设计设备驱动有帮助的信息。芯片数据手册的正确阅读方法是快速而准确地定位有用信息，重点阅读这些信息，忽略无关内容。下面以S3C6410A的数据手册为例来分析阅读方法，为了直观地反映阅读过程，本节的图都是直接从数据手册中抓屏而得到的。

打开S3C6410A的数据手册，发现页数为1378页，从头读到尾是不现实的。

S3C6410A数据手册的第1章“PRODUCT OVERVIEW”（产品综述）是必读的，通过阅读这一部分可以获知整个芯片的组成。这一章往往会给出一个芯片的整体结构图，并对芯片内的主要模块进行一个简洁的描述。S3C6410A的整体结构图如图2.27所示（见数据手册第61页）。

### 1.2 FEATURES

This section summarizes the features of the S3C6410X. Figure 1-1 is an overall block diagram of the S3C6410X.

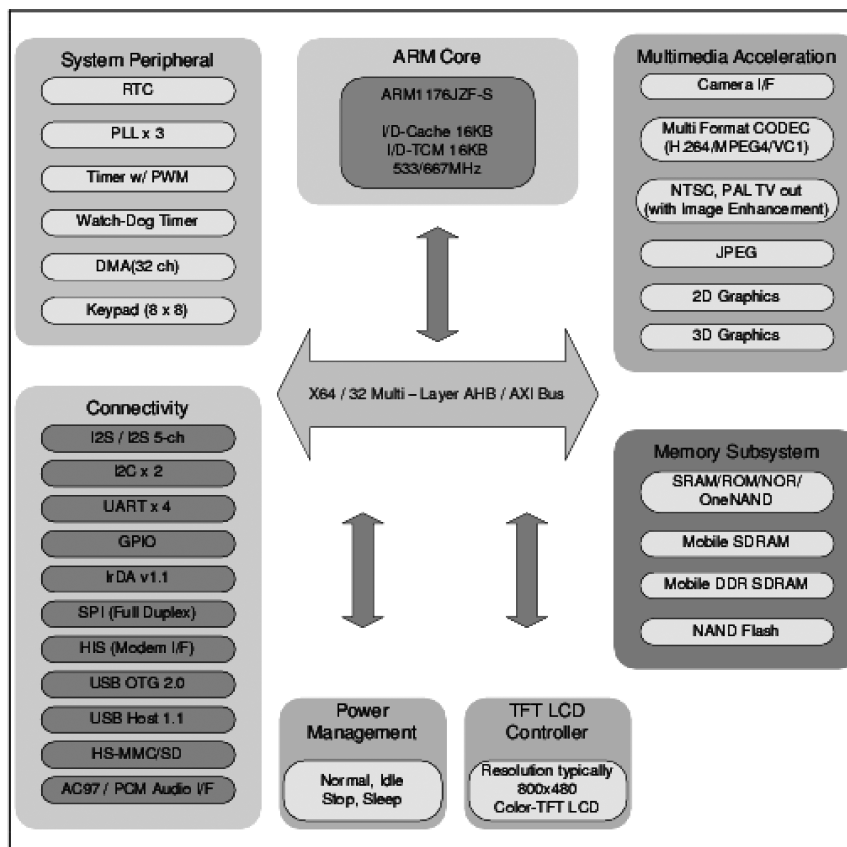


图 2.27 S3C6410A 数据手册中的芯片结构图

第2 ~ 43章中的每一章都对应S3C6410A整体结构图中的一个模块,图2.28为从Adobe Acrobat中直接抓取的S3C6410A数据手册的目录结构图。

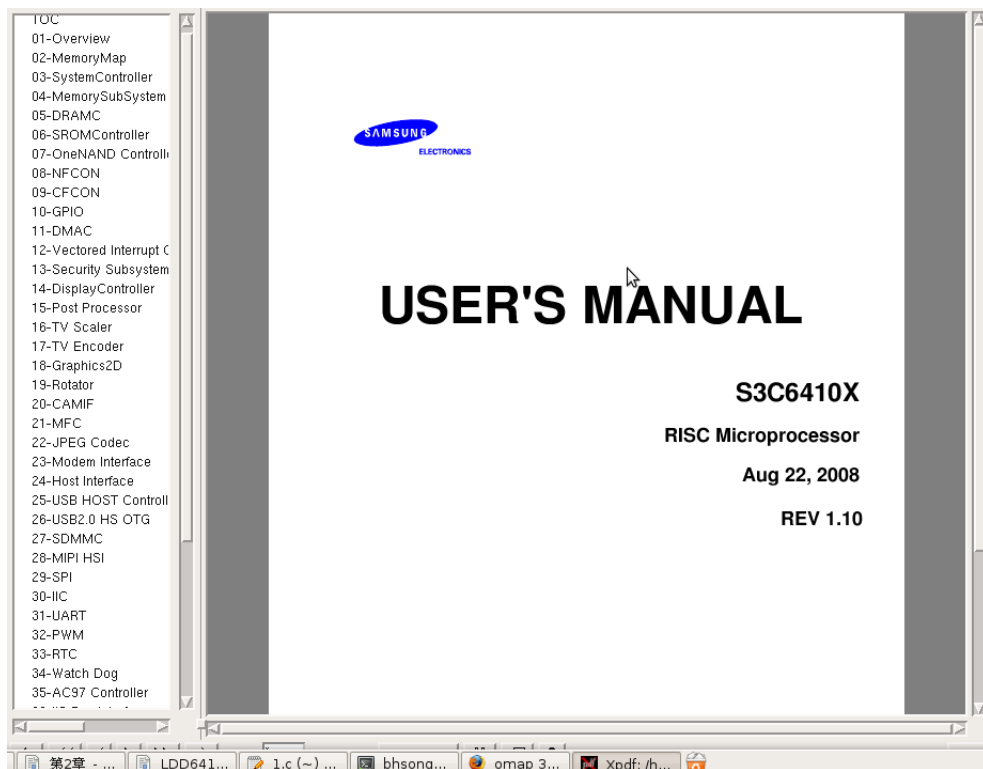


图 2.28 S3C6410A 数据手册的目录结构

第2章“MemoryMap”(内存映射)比较关键,对于定位存储器和外设所对应的基址有直接指导意义,这一部分应该细看。

第3 ~ 34章对应于CPU内部集成的外设或总线控制器,当具体编写某接口的驱动时,应该详细阅读,主要是分析数据、控制、地址寄存器(数据手册中一般会以表格列出)的访问控制和具体设备的操作流程(数据手册中会给出步骤,有的还会给出流程图)。譬如为了编写S3C6410A的I<sup>2</sup>C控制器驱动,我们需要详细阅读类似图2.29的寄存器定义表格和图2.30的操作流程图。

第44章“ELECTRICAL DATA”(对于电气数据,在图2.28中未画出),描述芯片的电气特性,如电压、电流和各种工作模式下的时序、建立时间和保持时间的要求。所有的数据手册都会包含类似章节,这一章对于硬件工程师比较关键,但是,一般来说,驱动工程师并不需要阅读。

第45章“MECHANICAL DATA”(机械数据)描述芯片的物理特性、尺寸和封装,硬件工程师会依据这一章绘制芯片的封装(Footprint),但是,驱动工程师无须阅读。

## 46 Linux设备驱动开发详解：基于最新的Linux 4.0内核

## 30.11.1 MULTI-MASTER IIC-BUS CONTROL (IICCON) REGISTER

Register	Address	R/W	Description	Reset Value
IICCON	0x7F004000	R/W	IIC Channel 0 Bus control register	0x0X
	0x7F00F000	R/W	IIC Channel 1 Bus control register	0x0X

IICCON	Bit	Description	Initial State
Acknowledge generation (1)	[7]	IIC-bus acknowledge (ACK) enable bit. 0: Disable 1: Enable In Tx mode, the IICSDA is free in the ACK time. In Rx mode, the IICSDA is L in the ACK time.	0
Tx clock source selection	[6]	Source clock of IIC-bus transmit clock prescaler selection bit. 0: IICCLK = PCLK /16 1: IICCLK = PCLK /512	0
Tx/Rx Interrupt (5)	[5]	IIC-Bus Tx/Rx interrupt enable/disable bit. 0: Disable, 1: Enable	0
Interrupt pending flag (2) (3)	[4]	IIC-bus Tx/Rx interrupt pending flag. This bit cannot be written to 1. When this bit is read as 1, the IICSDA is tied to L and the IIC is stopped. To resume the operation, clear this bit as 0. 0: 1) No interrupt pending (when read). 2) Clear pending condition & Resume the operation (when write). 1: 1) Interrupt is pending (when read) 2) N/A (when write)	0
Transmit clock value (4)	[3:0]	IIC-Bus transmit clock prescaler. IIC-Bus transmit clock frequency is determined by this 4-bit prescaler value, according to the following formula: $Tx\ clock = IICCLK / (IICCON[3:0] + 1)$ .	Undefined

图 2.29 芯片数据手册中以表格形式列出的寄存器定义

## 30.10 FLOWCHARTS OF OPERATIONS IN EACH MODE

The following steps must be executed before any IIC Tx/Rx operations.

1. Write own slave address on IICADD register, if needed.
2. Set IICCON register.
  - a) Enable interrupts
  - b) Define SCL period
3. Set IICSTAT to enable Serial Output

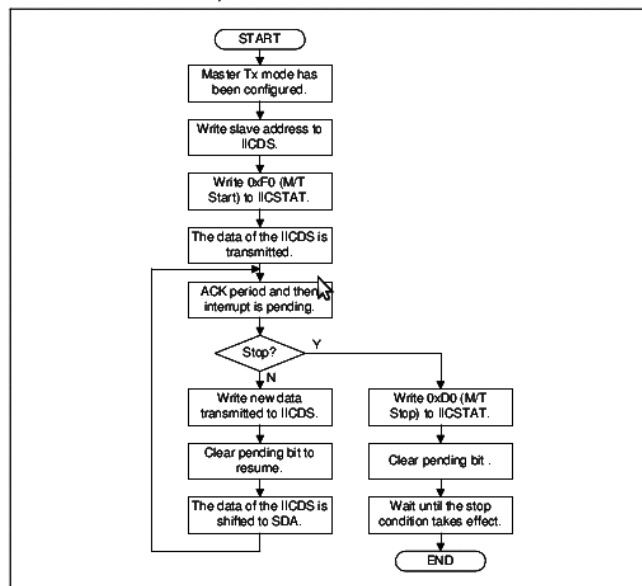


图 2.30 芯片数据手册中给出的外设控制器的操作流程

## 2.8 仪器仪表使用

### 2.8.1 万用表

在电路板调试过程中主要使用万用表的两个功能。

- 测量电平。
- 使用二极管挡测量电路板上网络的连通性，当示波器被设置在二极管挡，测量连通的网络会发出“嘀嘀”的鸣叫，否则，没有连通。

### 2.8.2 示波器

示波器是利用电子示波管的特性，将人眼无法直接观测的交变电信号转换成图像，显示在荧光屏上以便测量的电子仪器。它是观察数字电路实验现象、分析实验中的问题、测量实验结果必不可少的重要仪器。

使用示波器时应主要注意调节垂直偏转因数选择（VOLTS/DIV）和微调、时基选择（TIME/DIV）和微调以及触发方式。

如果 VOLTS/DIV 设置不合理，则可能造成电压幅度超出整个屏幕或在屏幕上变动太过微小以致无法观测的现象。图 2.31 所示为同一个波形在 VOLTS/DIV 设置由大到小变化过程中的示意图。

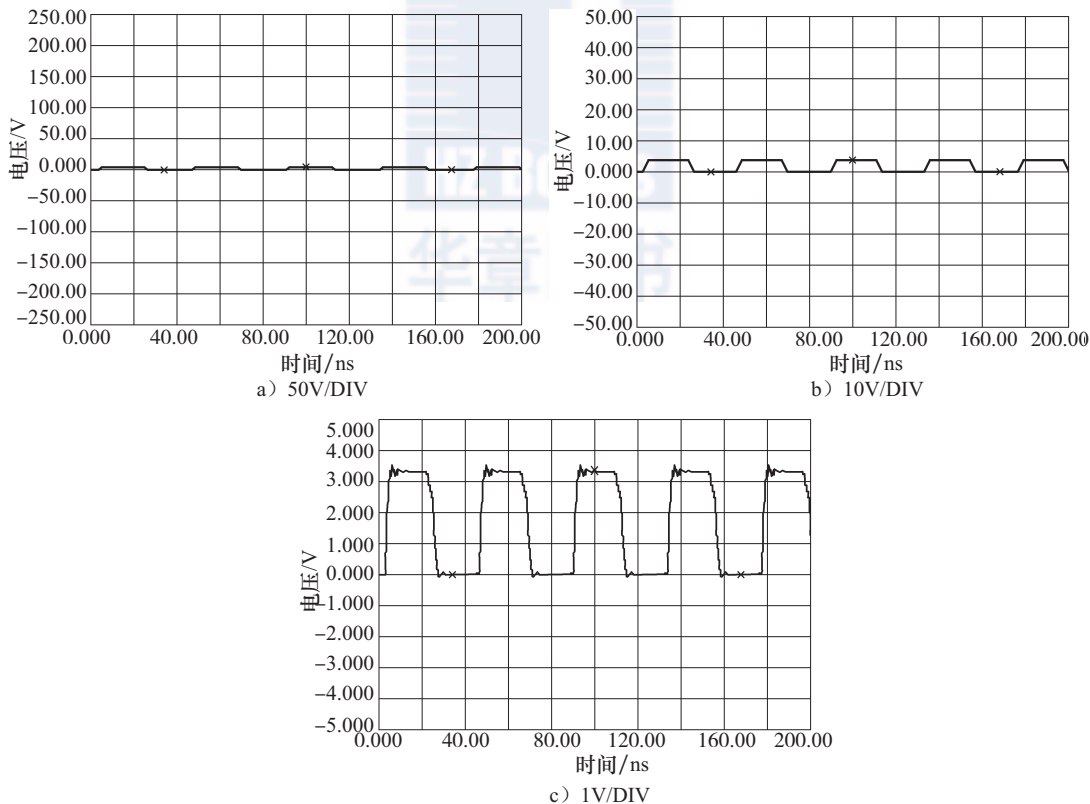


图 2.31 示波器的 VOLTS/DIV 设置与波形



## 48 Linux设备驱动开发详解：基于最新的Linux 4.0内核

如果 TIME/DIV 设置不合适,则可能造成波形混迭。混迭意味着屏幕上显示的波形频率低于信号实际频率。这时候,可以通过缓慢改变扫速 TIME/DIV 到较快的时基挡提高波形频率,如果波形频率参数急剧改变或者晃动的波形在某个较快的时基挡稳定下来,说明之前发生了波形混迭。根据奈奎斯特定理,采样速率至少高于信号高频分量的两倍才不会发生混迭,如一个 500MHz 的信号,至少需要 1GS/s 的采样速率。图 2.32 所示为同一个波形在 TIME/DIV 设置由小到大变化过程中的示意图。

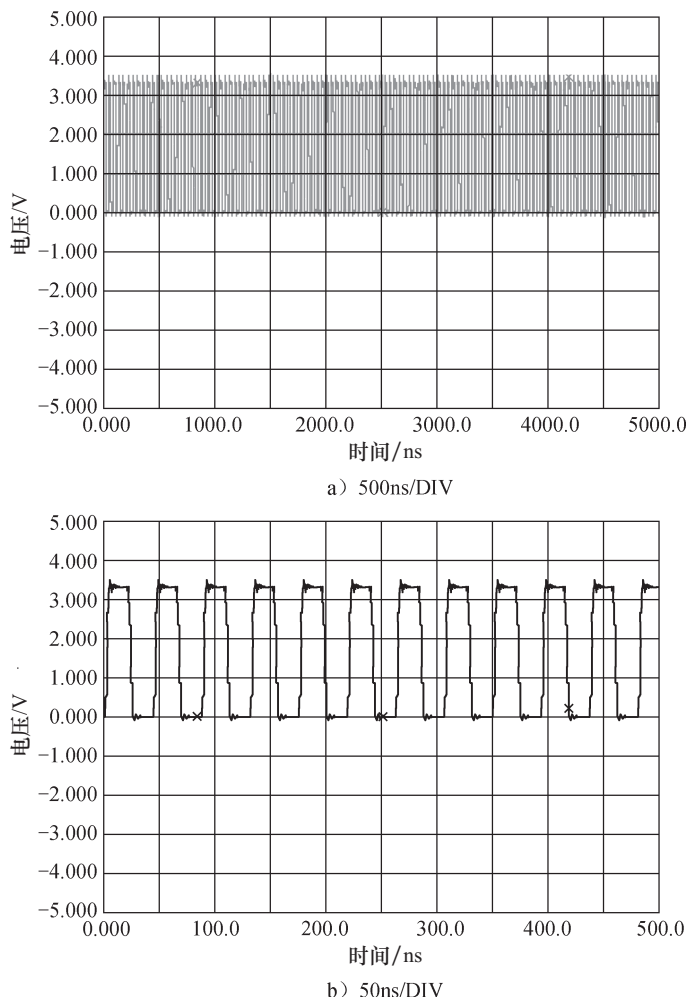


图 2.32 示波器的 TIME/DIV 设置与波形



奈奎斯特定理即为采样定理,指当采样频率 $f_{\text{smax}}$ 大于信号中最高频率 $f_{\text{max}}$ 的两倍时,即 $f_{\text{smax}} \geq 2f_{\text{max}}$ 时,采样之后的数字信号可完整地保留原始信息。这条定理在信号处理领域中的地位相当高,大致相当于物理学领域中的牛顿定律。

在示波器的使用过程中,要设置触发方式和触发模式。触发的目的是为了在每次显示的时候都从波形的同一位置开始,波形可以稳定显示。一般示波器都支持边沿触发,在某些情况下,我们也要使用视频触发、毛刺触发、脉宽触发、斜率触发、码型触发等。设定正确的触发,可以大大提高测试过程的灵活性,并简化工作。

示波器一般支持3种触发模式:自动模式、常规模式和单次模式。

- 自动模式(示波器面板上的 AUTO 按钮)。在这种模式下,当触发没有发生时,示波器的扫描系统会根据设定的扫描速率自动进行扫描;而当有触发发生时,扫描系统会尽量按信号的频率进行扫描。因此, AUTO 模式下,不论触发条件是否满足,示波器都会产生扫描,都可以在屏幕上看到有变化的扫描线,这是这种模式的特点。一般来说,在对信号的特点不是很了解的时候,可先选择自动模式。
- 常规模式(示波器面板上的 NORM 或 NORMAL 按钮)。在这种模式下,示波器只有当触发条件满足了才进行扫描,如果没有触发,就不进行扫描。因此在这种模式下,如果没有触发,对于模拟示波器而言,用户不会看到扫描线,对于数字示波器而言,不会看到波形更新。
- 单次模式(示波器面板上的 SIGL 或 SINGLE 按钮)。这种模式与 NORMAL 模式有一点类似,就是只有当触发条件满足时才产生扫描,否则不扫描。而不同在于,这种扫描一旦产生并完成后,示波器的扫描系统即进入一种休止状态,即使后面再有满足触发条件的信号出现也不再进行扫描,也就是触发一次只扫描一次。实际工作中,可能要根据情况在自动、常规和单次模式之间进行切换。

### 2.8.3 逻辑分析仪

逻辑分析仪是利用时钟从测试设备上采集数字信号并进行显示的仪器,其最主要的作用是用于时序的判定。与示波器不同,逻辑分析仪并不具备许多电压等级,通常只显示两个电压(逻辑1和0)。在设定了参考电压之后,逻辑分析仪通过比较器来判定待测试信号,高于参考电压者为1,低于参考电压者为0。

例如,如果以  $n$  MHz 采样率测量一个信号,逻辑分析仪会以  $1000/n$  ns 为周期采样信号,当参考电压设定为 1.5V 时,超过 1.5V 则判定为1,低于 1.5V 则为0,将逻辑1和0连接成连续的波形,工程师依据此连续波形可寻找时序问题。

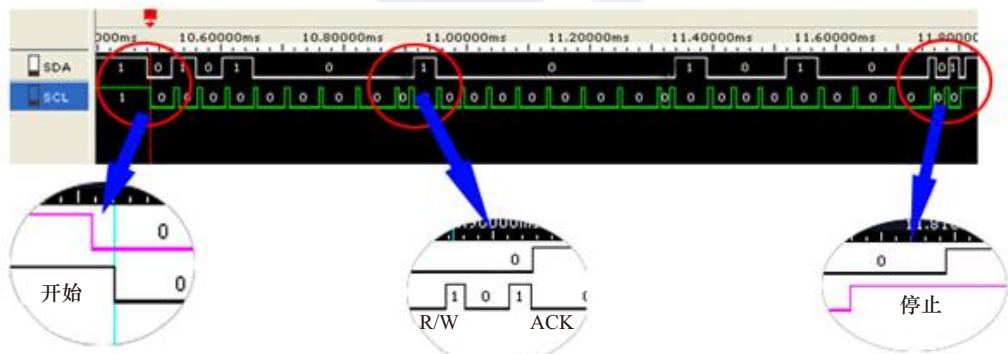
高端逻辑分析仪会安装 Windows 操作系统并提供非常友善的逻辑分析应用软件,在其中可方便地编辑探针、信号并查看波形。这种逻辑分析仪一般称为传统逻辑分析仪,其功能强大,数据采集、分析和波形显示融于一身,但是价格十分昂贵。有的逻辑分析仪则没有图形界面,但是可以通过 USB 等接口与 PC 连接,分析软件则工作在 PC 上。这种逻辑分析仪一般称为虚拟逻辑分析仪,它是 PC 技术和测量技术结合的产物,触发和记录功能由虚拟逻辑分析仪硬件完成,波形显示、输入设置等功能由 PC 完成,因此比较廉价。图 2.33 给出了两种逻辑分析仪。

## 50 Linux设备驱动开发详解：基于最新的Linux 4.0内核



图 2.33 逻辑分析仪

逻辑分析仪的波形可以显示地址、数据、控制信号及任意外部探头信号的变化轨迹，在使用之前应先编辑每个探头的信号名。之后，根据波形还原出总线的工作时序，图 2.34 给出了一个 I<sup>2</sup>C 的例子。目前，很多逻辑分析仪都自带了协议分析能力，可以自动分析出总线上传输的命令、地址和数据等信息。

图 2.34 从逻辑分析仪波形还原 I<sup>2</sup>C 总线

逻辑分析仪具有超强的逻辑跟踪分析功能，它可以捕获并记录嵌入式处理器的总线周期，也可以捕获如实时跟踪用的 ETM 接口的程序执行信息，并对这些记录进行分析、译码且还原出应用程序的执行过程。因此，可使用逻辑分析仪通过触发接口与 ICD（在线调试器）协调工作以补充 ICD 在跟踪功能方面的不足。逻辑分析仪与 ICD 协作可为工程师提供断点、触发和跟踪调试手段，如图 2.35 所示。



ICD 是一个容易与 ICE（在线仿真器）混淆的概念，ICE 本身需要完全仿真 CPU 的行为，可以从物理上完全替代 CPU，而 ICD 则只是与芯片内部的嵌入式 ICE 单元通过 JTAG 等接口互通。因此，对 ICD 的硬件性能要求远低于 ICE。目前市面上出现的很多号称为 ICE 的产品实际上是 ICD 等，但是人们一般也称它们为 ICE。

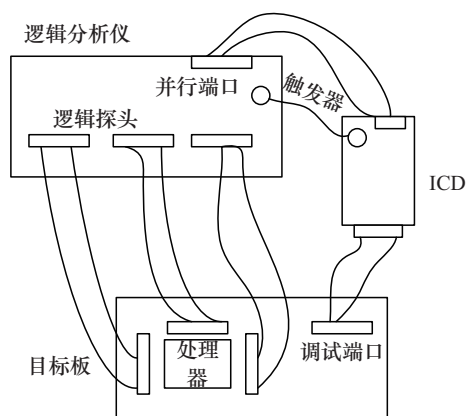


图 2.35 逻辑分析仪与 ICD 协作

## 2.9 总结

本章简单地讲解了驱动软件工程师必备的硬件基础知识，描述了处理器、存储器的分类以及各种处理器、存储器的原理与用途，并分析了常见的外围设备接口与总线的工作方式。

此外，本章还讲述了对驱动工程师进行实际项目开发有帮助的原理图、硬件时序分析方法，芯片数据手册阅读方法以及万用表、示波器和逻辑分析仪的使用方法。