电子信息与电气学科规划教材

基于多核平台的嵌入式系统设计方法

林继鹏 编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY http://www.phei.com.cn

电子信息与电气学科规划教材

基于多核平台的嵌入式系统设计方法

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

本书主要介绍多线程编程及其处理方法、自动并行化程序设计、Intel 的 IPP 性能原语用于信号处理、多核程序设计的评估和调试方法、多核平台程序设计的任务分解和函数分解方法；线程构建模块的基本算法和高级算法、线程和内存检测工具 Inspector 的使用和原理、以及热点分析工具 Amplifier 的原理和应用等。

本书的特点是理论与实践相结合，重在实践能力的培养；书中有大量的源代码可供参考。本书可作为电子信息相关专业高年级本科生和研究生的教材。

图书在版编目（CIP）数据

基于多核平台的嵌入式系统设计方法 / 林继鹏编著. —北京：电子工业出版社，2011.1
（电子信息与电气学科规划教材）
ISBN 978-7-121-12228-6

Ⅰ.①基… Ⅱ.①林… Ⅲ.①微型计算机—系统设计 Ⅳ.①TP360.21

中国版本图书馆 CIP 数据核字（2010）第 217135 号

责任编辑：竺南直　　文字编辑：雷洪勤
印　　刷：涿州市京南印刷厂
装　　订：涿州市桃园装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1 092　1/16　印张：16.75　字数：428 千字
印　　次：2011 年 1 月第 1 次印刷
印　　数：4 000 册　　定价：29.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

2006年，当第一台双核计算机出现的时候，对大多数人来说，“双核”是一个新鲜事物。在同一块CPU内集成两个或多个计算引擎的单元，无论是从理论上还是直观感觉上似乎能数倍提高计算机的运行速度，但实际上并非如此。根本的原因是软件的开发速度已经远远滞后于硬件的发展，当前还没有完全支持多核平台的操作系统，要想充分利用多计算引擎带来的实惠，掌握软件开发的理论和提高实践能力迫在眉睫。本书正是在这种背景下酝酿而成。本书的写作是基于“长安大学-Intel多核技术联合实验室”所提供的多核平台，并得到学校质量工程的资助。

本书共分为三个部分。第一部分包括第1章到第3章，详细阐述了多核系统的组成、软件的性能评估方法和并行程序设计模型。作为并行程序设计的主要途径之一，在第3章论述了多线程的基础知识和设计方法。

第二部分包括第4章到第8章，是多核平台程序设计的核心部分，主要内容包括自动并行化程序设计、Intel的IPP性能原语用于信号处理、线程构建模块的基本算法和高级算法，以及基于任务分解和函数分解的并行程序设计及调试方法。通过本部分的学习，读者应改变传统的程序设计思想，逐步建立并行程序设计的基本理论和方法，为提高就业竞争力打下坚实基础。

第三部分包括第9章到第11章，本部分的主要内容是基于Intel在2009年6月份推出的全新并行程序设计parallel studio来展开的，包括线程和内存检测工具Inspector的原理和使用、以及热点分析工具 Amplifier 的原理和应用等。通过本部分的学习，读者应掌握并行套件的原理和使用，能自主开发出高性能的并行程序。

本书第3章由王平编写，第4章由茹锋博士编写，第5章由博士生陈金平编写，其余部分由林继鹏编写。研究生龚结龙、秦海兵参与了第7～11章的调试与整理工作。本书编写过程中，还得到巨永锋教授、张启水研究员和汪贵平教授的指导，博士生李刚对本书的编写也付出诸多汗水，在此对他们表示衷心的感谢。

本书的特点是基本理论与实践环节结合，重在实践能力的培养。书中有大量的源代码供参考，图文并茂，易于学习。

作　者

目　录

第 1 章　从多处理器系统到多核系统

早前的研究主要集中在单处理器系统上，但随着商业应用和工业发展的要求，越来越多的领域需要使用到多片微处理器。多处理器系统可以将工作量分配到不同的处理器上，从而实现冗余、加速、模块化甚至是简化代码。

使用多处理器的理由主要是由项目的要求决定的。如图 1-1 所示，假设系统需要实现显示、键盘、响应事件、控制执行器和与主机 Host 通信功能；以对 Events 的响应要求为例，执行器 Actuators 需要根据 Events 来实时做出动作，但 Events 的数据量很小。如果采用单片 CPU，CPU 的处理速度必须足够快以致对 Events 的开销不会影响高速 Events 的性能（如中断），也不会因为 Events 的快速重复速率降低密集消息处理函数的数量。这样可能会过分追求高性能的 CPU 而增加系统的成本。这种情况看起来并不是那么直观：对于彼此独立的任务群而言，每一个处理器的处理负载要比由一个处理器处理的负载小得多。

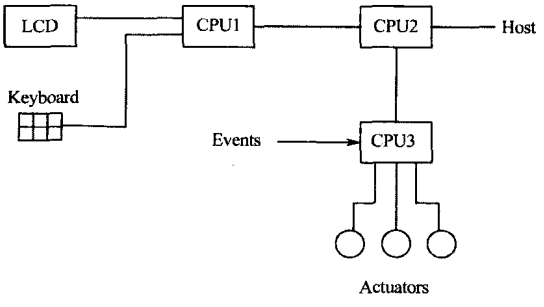

图 1-1　多处理器系统示意图

在图 1-1 中，每一个 CPU 都可以有自己独立的 PROM、RAM 和 I/O，其中 CPU1 的功能是实现人机交互，CPU3 用于控制执行器和处理外部事件，CPU2 用于连通 CPU1 和 CPU3 以及上层主机。系统中需要采用多少个处理器依赖于：

- 软件的相互依赖关系；
- 处理器的吞吐量；
- 处理器的位置。

1.1　板内处理器间的通信

实现多处理器嵌入式系统的一个难点是处理器之间的通信，根据处理器物理位置的不同，可以分为板内通信和板间通信；根据通信方式的不同，又可以分为寄存器通信和串行通信等。最为简单和廉价的通信方式是利用数字锁存器。如图 1-2 所示是双向通信寄存器的

原理图。

假设数据流是从 CPU1 到 CPU2。寄存器每次传送 8 位数据，总线切换时钟（Bus Exchange Clock）实现双向数据流向的控制。当时钟信号从低变成高的时候，74LS374 的输出 Q_n 跟踪其输入 D_n，即此时数据处于更新状态；而当时钟信号为低电平时，不管 D_n 如何变化，Q_n 始终维持不变，即此时数据处于锁定状态。74LS374 的这一性能有助于让 CPU2 有充足的时间来获取 CPU1 发送来的数据，但缺点是 CPU1 却不知道 CPU2 是否获取数据，也不知道自己需要等待多长时间来更新数据。同样的问题也存在于当 CPU2 向 CPU1 发送数据时。第二个缺点是，CPU1 的处理速度严重受限于 CPU2，因为它一直要等待 CPU2 成功接收数据，特别是 CPU2 被某种中断引起循环时。要解决这一问题，需要给两片 74LS374 分别加一个更新和读完的状态信号，状态信号可以利用 74LS374 的状态缓冲、CPU 端口的某一位或者是扩展芯片的某个引脚。为了提高系统的处理速度，建议将读完的状态信号直接连接到两 CPU 的外部中断引脚上，这样当一方确认读完发送来的信息，就可以及时通知到发送信息方。

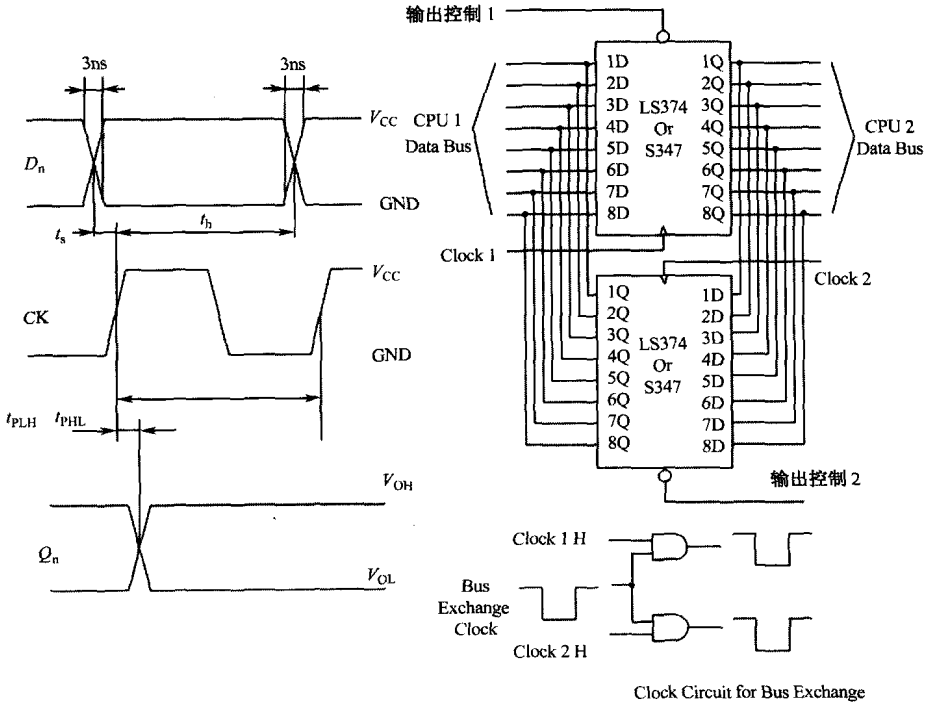

图 1-2 双向通信寄存器原理图

如果所对应的 CPU 是内置 DMA 控制器的，那么传输的速度还可以进一步得到提高，这是因为 DMA 不需要 CPU 本身的干预。简单的说，此时将更新完的状态信号连接到 CPU2 的 DMA 请求信号线上，而将读完的状态信号连接到 CPU1 的 DMA 请求信号线上。最快的 DMA 方式是单周期和单地址的 flyby 模式（另外一种模式是 fetch-and-deposit），在这种模式下，一个总线周期可以同时完成从源地址取数并写入目的地址的过程。DMA 传送数据的过程如下：

（1）CPU 利用指令向 DMA 控制器发出 DMA 传送请求；

（2）DMA 控制器向 CPU 发出总线请求，要求 CPU 交出总线管理和使用权，并在接到 CPU 的总线响应后接管系统总线的管理和使用权，从而变成系统的主设备；

（3）DMA 控制器将被访问的存储器单元地址送到地址总线上；

（4）向存储器和进行 DMA 传送的外部设备（CPU2）发出读写命令，则存储器和外部设备通过数据总线进行数据交换；

（5）若需要继续传送，则将 DMA 请求信号继续保持高电平；

（6）如果数据传送完成，DMA 撤销对 CPU 总线的请求，交回系统总线的管理和使用权。

DMA 传送的时序图如图 1-3 所示。

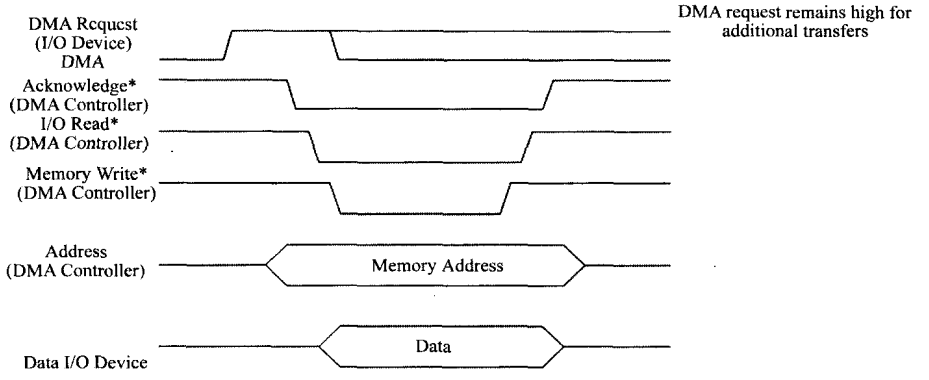

图 1-3　DMA 传送的时序图

必须指出的是，仅当其中一个 CPU 具有 DMA 控制器时也可以实现 DMA 传送，图 1-4 给出了一个 DMA 传输的实用电路。

在图 1-4 中，CPU1 将需要发送的数据发送到内存中，编程利用 DMA 控制器传送，而 CPU2 编程实现其 DMA 控制器从寄存器中读取数据并存储到内存中。问题是：CPU2 无法确定传送数据的大小，通常有以下三种方法解决这一问题：

（1）第一种方法是使用固定的数据量，比如每次只传送 256 个字节，如果数据量少于 256 个字节，可在数据后补 0 来达到固定长度。

（2）第二种方法是采用报头，即 CPU1 通过 DMA 传送的第一位数据是其要传送的数据长度信息，CPU2 的 DMA 接收到该位信息后即再次引发 DMA 中断开始接收数据。

（3）第三种方法需要在两个 CPU 之间开辟一条额外的中断路径。首先 CPU2 建立其 DMA 控制器来接收比实际信息更多的字节；之后 CPU1 建立其 DMA 传送机制并传送数据，一旦传送完成，即通过独立的中断路径通知 CPU2，CPU2 随即开始接收并处理数据。在这种模式下，CPU2 是不会产生中断的，因为它不需要处理信息量。

尽管利用 DMA 实现 CPU 之间的互连可以大大提高其通信速率，但如果 CPU 之间的运行速度相差太大，也会存在问题。如果 CPU1 的速度小于 CPU2 的速度，CPU2 可能会检测到数据更新完的状态并开始读取数据，而此时 CPU1 仍然处在写使能状态；反过来，如果 CPU1 的速度大于 CPU2 的速度，可能会导致 CPU1 在写新数据位的时候，CPU2 仍然处在

读使能状态。这两种情况的出现，均会导致信息的误读或重读。要避免这一情况的发生，最好的方法是在 CPU 间加上 FIFO。

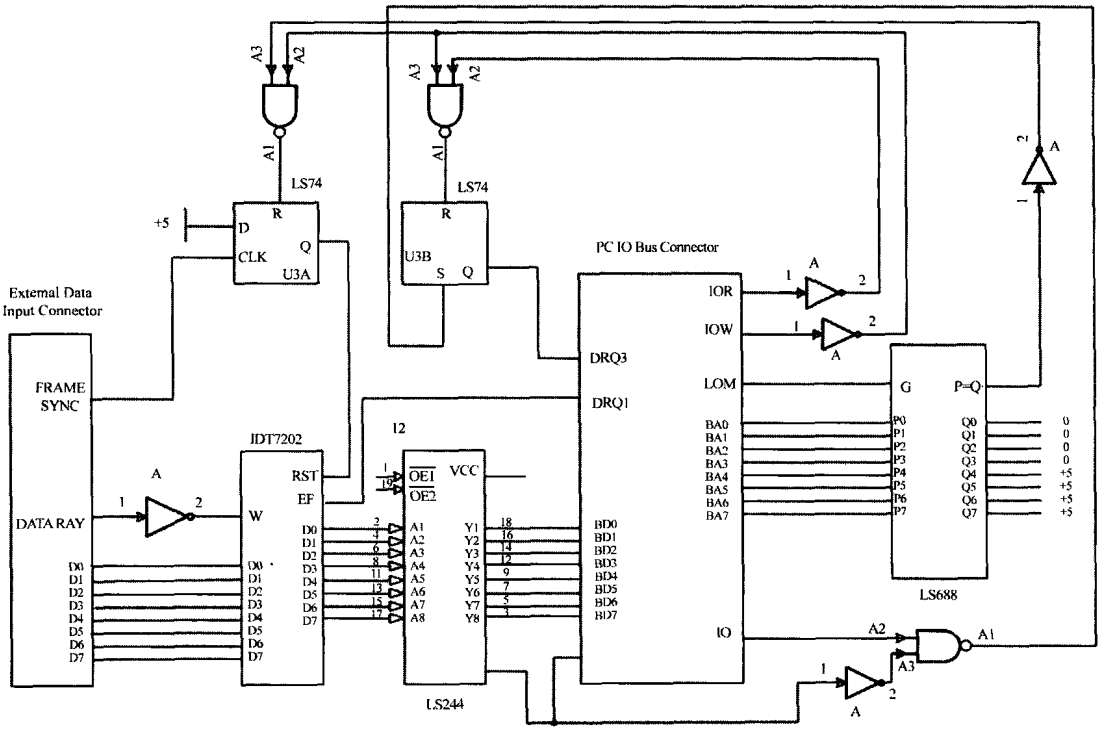

图 1-4 DMA 接口示意图

FIFO（First In First Out）的全称是先进先出存储器。先进先出也是 FIFO 的主要特点。20 世纪 80 年代早期，FIFO 芯片是基于移位寄存器的中规模逻辑器件。容量为 n 的这种 FIFO 中，输入的数据逐个寄存器移位，经 n 次移位才能输出。因此，这种 FIFO 的输入到输出的延时与容量成正比，工作效率得到限制。为了提高 FIFO 的容量和减小输出延时，现在 FIFO 的内部存储器均采用双端口 RAM，数据从输入到读出的延迟大大缩小。以通用的 IDT7202 为例，结构框图如图 1-5 所示。输入和输出具有两套数据线。独立的读/写地址指针在读/写脉冲的控制下顺序地从双端口 RAM 读/写数据，读/写指针均从第一个存储单元开始，到最后一个存储单元，然后又回到第一个存储单元。标志逻辑部分即内部仲裁电路通过对读指针和写指针的比较，相应给出双端口 RAM 的空（EF）和满（FF）状态指示，甚至还有中间指示（XO/HF）。如果内部仲裁仅提供空和满状态指示，FIFO 的传输效率得不到充分的发挥。新型的 FIFO 提供可编程标志功能，例如，可以设置空加 4 或满减 4 的标志输出。目前，为了使容量得到更大提高，存储单元采用动态 RAM 代替静态 RAM，并将刷新电路集成在芯片内，且内部仲裁单元决定器件的输入、读出及自动刷新操作。FIFO 只允许两端一个写、一个读，因此 FIFO 是一种半共享式存储器。在双 CPU 系统中，只允许一个 CPU 往 FIFO 写数据，另一个 CPU 从 FIFO 读数据。而且，只要注意标志输出，空指示不写，满指示不读，就不会发生写入数据丢失和读出数据无效的现象。

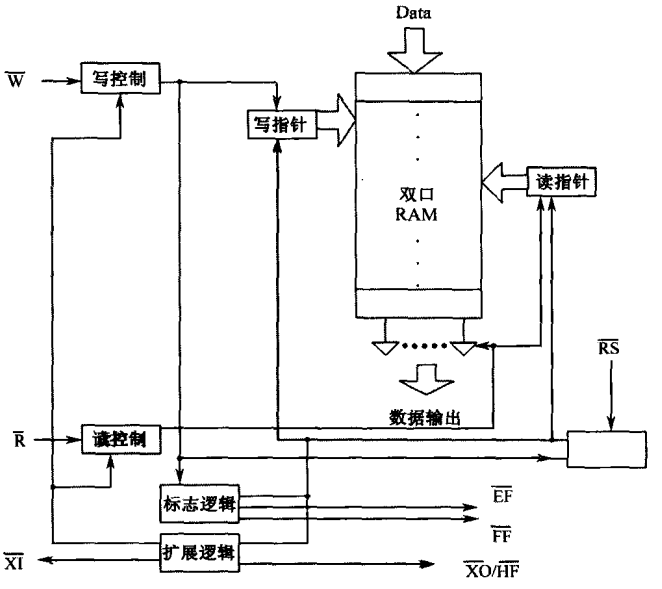

图 1-5　基于双端口 RAM 的 FIFO

数字信号处理器（DSP）能实时快速地实现各种数字信号处理算法，而 DSP 的控制功能不强，可以采用 8051 单片机控制数据采集板，将采集的原始数据送给 DSP 处理并将处理结果传送给 8 位单片机。图 1-6 给出了利用 1 片数字信号处理器 TMS320F206（以下简称 DSP）和 2 片 AT89C51 单片机（以下简称 MCU）构成多机数据采集系统的接口图。

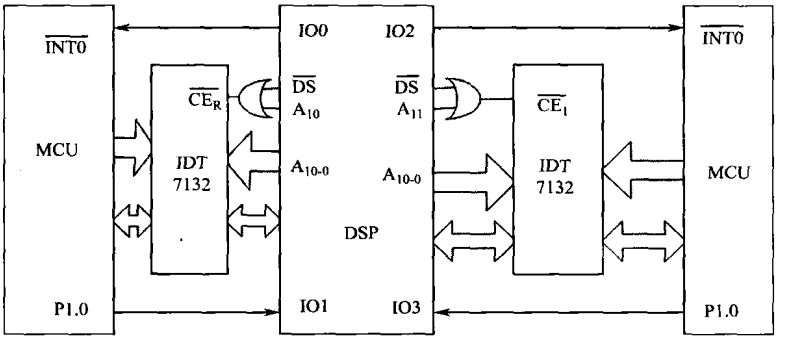

图 1-6　数字信号处理器和单片机通信系统

系统采用两片 CMOS 静态双端口 RAM（IDT7132）实现 MCU 和 DSP 的数据双向传递。双端口 RAM 作为 DSP 的片外数据存储器，即用外部数据存储器选通信号 DS 和高位地址信号经高速或门输出选通双端口 RAM 的片选信号。这样可以利用 DSP 的重复操作指令（RPT）和数据存储器块移动指令（BLDD）减少数据传送时间，双端口 RAM 的 8 位数据总线接在 DSP 的低 8 位。IDT7132 的仲裁逻辑控制只提供 Busy 逻辑输出，而由于 MCU 无 Busy 功能，只能采用自行设计的软件协议仲裁方法。将双端口 RAM 划分为两块：上行数据区（DSP 接收 MCU 采集的数据区）和下行数据区（DSP 输出处理结果区）。此处的上

行数据区将远大于下行数据区。采用 DSP 的 4 个 I/O 口与 MCU 中断口和 I/O 口相连，并在数据区中规定一个信令交换单元。以 DSP 采集右端 MCU 上行数据为例来说明仲裁流程：

（1）初始化时，DSP 置 IO3 为输出口，保持高电平，IO2 为输入口（MCU 使其初始化为低电平）；

（2）DSP 需要采集 MCU 数据时，向右端 IDT7132 下行数据区的下行信令字单元（此处设为 00H）写入需要取数的信令字，再向右端 MCU 发中断，置 IO3 为低电平，然后查询 IO2 等待 MCU 应答；

（3）MCU 及时响应中断后，则先从 IDT7132 的下行数据区的下行信令字单元读取 DSP 请求信息，检测为 DSP 需要取数的下行信令。然后，向 IDT7132 上行数据区的上行信令字单元写入数据，准备好需要 DSP 取数据的信令（00H）或数据未准备好的信令（01H）。最后，向 DSP 发送应答信号，置 IO2 为高电平（此处高电平的持续时间只要 DSP 可以检测到即可）；

（4）DSP 检测到 IO2 为高电平，表明 MCU 应答，立即读取 IDT7132 上行数据区的上行信令字单元。若为可以取数据的上行信令，则从 IDT7132 上行数据区取出采集数据。完成后，需要向右端 MCU 发送采集结束下行信令（01H）；若为数据未准备好的上行信令（01H），则跳转至与左端 MCU 通信程序中。

1.2 板间通信

板与板之间的通信主要采用串行通信方式。根据其协议的不同，可以分为 RS-485/232C、I^2C、CAN 或并口等形式。图 1-7 是采用 RS-485 总线实现三片 CPU 之间数据互连的示意图，CPU 之间共享两条串行总线。

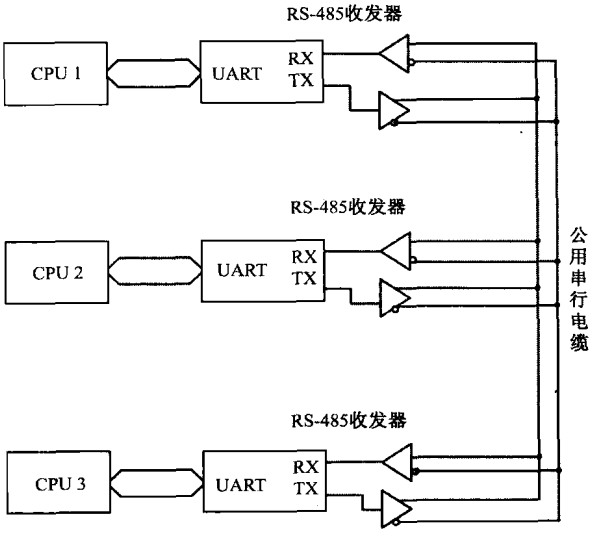

图 1-7 基于 RS-485 的多 CPU 互连示意图

第二种常用的板间通信采用 CAN（Controller Area Network）协议，CAN 起源于汽车的应用需求。与图 1-7 不同的是，CAN 支持多主机模式，每一个主机节点的地位都是等同的（即对等式的现场总线），而且并不指定具体的地址。主机的地址信息包含在被传送信息中，而且在系统工作时可以添加或移动节点。

CAN 总线采用 120 欧姆的差分公用线，其传输速率随传输距离的不同而不同，10km 的时候其传输速率为 5kbits/s，在 40m 的时候传输速率为 1Mbits/s。CAN 总线最多可以有 110 个节点，传输介质可以是双绞线、同轴电缆和光纤。CAN 总线遵循 ISO/OSI 网络标准模式，但只使用了其中的物理层和数据链路层。CAN 在 1991 年 9 月制定并发布了其第二版，该版本包括 A 和 B 两个部分，其中 A 版本给出了其早期 1.2 版本中的报文格式，而 B 版本给出了其标准和扩展的两种报文格式。这导致 CAN 有两种帧格式，其中一种是含有 11 位标识符的标准格式和 29 位标识符的扩展格式。其标准格式如图 1-8 所示。

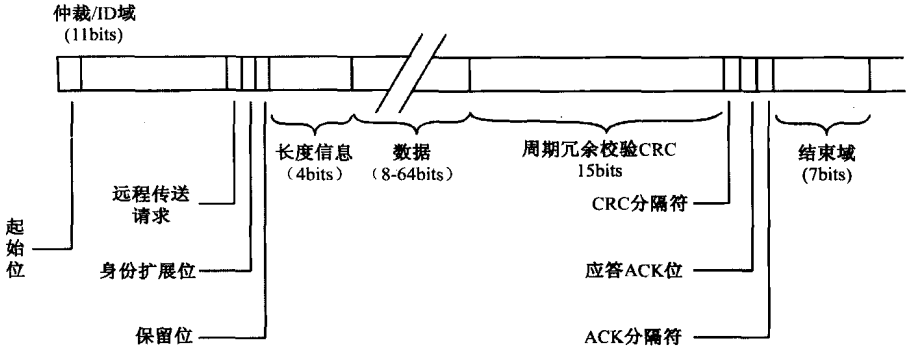

图 1-8　CAN 的 11 位标识符的帧格式

CAN 使用非归零码（Non-return to Zero），其信号线上存在的两种状态电平如图 1-9 所示。

从图 1-9 可知，CAN 上的信号存在两种状态，当其处在状态 1 时，差分信号线上的电平均为 2.5V，而处于状态 0 时，差分线上的其中一根的电平上拉至 3.5V，另一根被下拉到 1.2V。现在很多的微处理器芯片如 Silabs 的 C8051F040 都内置 CAN 接口模块。

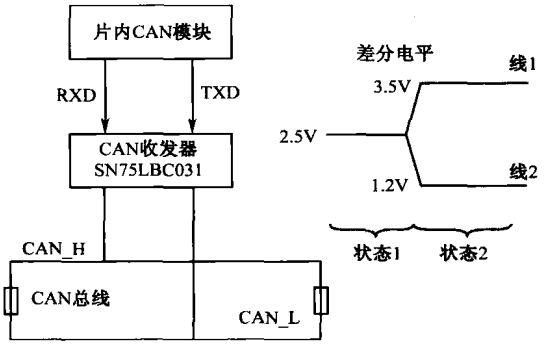

图 1-9　CAN 差分电平示意图

计算机并口的工作模式有 5 种：标准并行接口（SPP）、简单双向接口（PS2，即 EPP 1.7）、