RAID 系统中 RAID 级别的具体实现算法

陈　赟①杨根科②吴智铭③

摘　要　RAID 系统通过不同的 RAID 级别提供不同的响应速率和容错性。本文介绍了正在开发中的以 Vxworks 嵌入式实时操作系统为开发工具的 RAID 系统软件模型，包括 RAID 诸级别，并详细说明了在实现 RAID 各个级别的具体算法中的几个关键数据结构和算法流程。

关键词　RAID　磁盘阵列　Cache　算法实现

一、引言

1988 年美国加州大学的 David A. Patterson 提出了磁盘阵列的概念。RAID(Redundant Array of Independent Disks)是一种把多块独立的物理硬盘按不同方式组合起来形成一个逻辑硬盘，从而提供比单个硬盘更高的存储性能和提供数据冗余的技术。组成磁盘阵列的不同方式称为 RAID 级别。RAID 技术提供了比通常的磁盘存储更高的性能指标、数据完整性和数据可用性。在当今面临的 I/O 总是滞后于 CPU 性能的瓶颈问题越来越突出的情况下，RAID 是有效弥补这个缺陷的技术之一[1]。

RAID 被定义为 0～5 共 6 个级别，各级别针对不同的用途进行了性能上的优化。某些级别组合也可以形成新的级别，如 RAID0+1。RAID2－4 冗余信息计算开销较大，而容错性不高，未被广泛应用。RAID 0：磁盘分段(Striped Disks)，将数据块分成多个段，以并行的方式分别写入相应的 RAID 成员盘中。数据的读写可以并行地在多个通道甚至多个硬盘上同时进行，可以成倍的提高系统读写速度。RAID1：磁盘镜像(Mirrored Disks)，双盘结构。数据均有镜象映射备份，但系统的读写速度没有提高。RAID0+1：镜像十分段。RAID5：分散校验并行读写(Independent Access RAID with Distributed Parity)，采用了奇偶校验码作为冗余信息。这种方法利用校验位代替数据备份，提高了磁盘的利用率。JBOD 和 SPANNING：JBOD(Just Bndle Of Disks)是"简单磁盘捆绑"，不采用任何的冗余算法，JBOD 不是标准的 RAID 级别，它只是在近几年才被一些厂家提出，并被广泛采用。而 Span 是在逻辑上把几个物理磁盘一个接一个串联到一起，从而提供一个大的逻辑磁盘。Span 也不提供数据安全保障。

在一个 RAID 系统中可以包含数个 RAID 级别，在 SCSI 总线下最多能够包括 8 个逻辑单元(由若干个磁盘构成的某个 RAID 级别)，满足用户不同的需求。在一个 RAID 系统中安装多个级别是很有意义的。比如，在有两个驱动器的系统中，一个驱动器使用 RAID0 给临时性文件提供高速的 I/O 性能；另一个使用 RAID1 给重要数据提供安全的存储[1]。

二、RAID 系统软件模型

RAID 系统进行按功能模块进行分拆，是一件比较困难的事情。根据以往的经验、建模的可视性和实现的可行性，我们将 RAID 系统分为四个功能块：启动模块(Initiator)、目标模块(Target)、算法实现模块(Vdisk)和缓存模块(Cache)，另外附加一个配置模块。模块间存在着密切的数据信息交换和控制信息交换，如图 1 所示[5]：

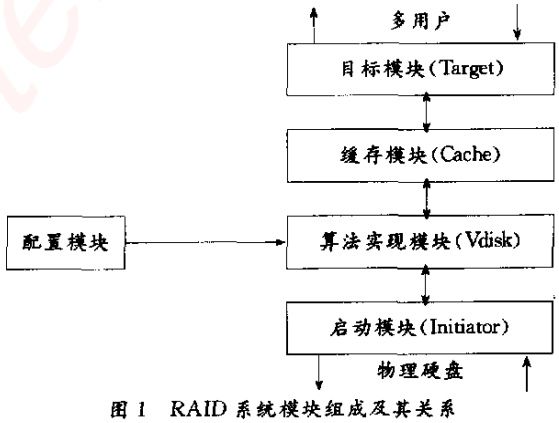

图 1　RAID 系统模块组成及其关系

启动模块(Initiator)：该模块主要负责对物理磁盘进行管理，完成读写操作、错误处理、响应磁盘中断和系统阵列初始化等功能。

目标模块(Target)：该模块主要和用户接口进行信息交互。它主要负责对从用户那里传来的命令进行"翻译"、管理，并将命令传给 RAID 缓存模块。对多用户发送的多命令要进行高效管理，对用户的状态答复要满足接口协议等。

算法实现模块(Vdisk)：该模块是系统最具智能的功能模块，它主要完成两个功能：RAID 虚拟盘与实际物理盘的地址映射。出错处理。出错处理能力是 RAID 安代性和稳定性的一个重要度量。因为实际物理硬盘在任一时刻只能处理一个读

① 陈　赟　上海交通大学自动化系　上海　200030
② 杨根科　上海交通大学自动化系　教授　上海　200030
③ 吴智铭　上海交通大学自动化系　教授　博士生导师　上海　200030

写任务，而 Cache 对此并不知情，会源源不断地往 Vdisk 模块发任务，所以 Vdisk 模块还缓存由 Cache 模块发来的读写任务，直到物理硬盘空时，才往下转发。

缓存模块(Cache)：该模块完成预读功能，就是将系统需要的数据预先读进缓存；快速寻址功能就是将需要的数据地址尽快的找到；应变能力，就是能够有效地处理服务要求不同"大块"数据和"小条"数据。

配置模块(Configure)：在系统初始化阶段，用户要根据物理硬件的情况和自身对于系统的要求来配置 RAID 系统级别及其相关信息，系统将根据这些配置信息进行工作。

三、RAID 级别的实现算法

本实验室开发的 RAID 系统是以 VxWorks 嵌入式操作系统的应用程序来实现的。VxWorks 是具有微内核、高性能、可伸缩的实时操作系统。VxWorks 的所有功能都由 C 程序库提供，VxWorks 的开发环境 Tornado 的命令和调试环境中可以调用 C，VxWorks 又支持广泛的网络通信协议，所以在 VxWorks 下能较方便地进行实时操作系统的开发[2]。

1. Vdisk 模块对 Cache 模块和 Initiator 模块的接口描述

系统间各模块是通过消息队列的方式传递信息，定义如下的读写任务体和消息体。

读写任务体：

```
typef struct{
uint32 task—blockstart//读写任务的起始地址
uint32 task—blockcount//读写任务的数据块数
uint8 task—rwdev//读写任务的设备号
uint8 task—type//读任务 OR 写任务
uint8 task—status//用作记录任务的状态
uint32 task—buffer//用来在 Vdisk 模块中保存要读的或要写的数据块
}TrwTaskitem;
```

消息体：

```
typedef struct{
uint8 msg—id;
uint8 msg—status;
uint8 msg—content[MAX—MSG—CONTENT—LEN];
}TmsgItem;
```

其中 TrwTaskItem 作为内容填充 TmsgItem 的 msg—content。

从 Cache 模块发来的消息中的读写任务中 task—blockstart、task—blockcount 和 task—rwdev 对应的是读写任务在逻辑磁盘上的数据块起始地址、数据块数和读写设备号。相应的有 Vdisk 模块发向 Initiator 模块的消息中的读写任务中 task—blockstart、task—blockcount 和 task—rwdev 对应的都是读写任务在实际物理磁盘上的数据块地址、数据块数和读写设备号。

另外由于拆分任务的工作在 Cache 模块完成，因此每一个从 Cache 模块发来的消息中读写任务的数据块数都为 1，这就是说任何一个读写任务只对应于一个物理硬盘，不会跨多个物理硬盘。

2. 各个 RAID 级别算法的具体实现

在 Vdisk 模块中，RAID 级别算法的具体实现包括逻辑磁盘地址与物理磁盘地址之间的相互转换、读写任务的产生以及差错控制。下述算法从原理上说明了实现的具体方法，但同时也省去了一些繁杂的控制细节[1]。

设由 Cache 模块发来的读写任务为 pCacheTask。由 Vdisk 模块发往 Iniatior 模块的读写任务为 pInitiatorTask。物理磁盘阵列由一个数组纪录 PhyDisk[MaxPhyDiskCount]，设每一个物理磁盘的容量相同，即磁盘的数据块数相等，大小为 PhyDiskBlockCount。总共物理磁盘的个数为 pPhyDiskCount 个。NtempStart 是用来表示物理磁盘起始地址的临时变量，初值为零。PPhyDisk 是一个 PhyDisk 结构体的一个对象，PhyDisk 结构体定义如下

```
typedef struct{
uint32 pPhyDisk—blockstart;//物理硬盘的起始地址
uint32 pPhyDisk—blockcount;//物理硬盘上的数据块数
uint32 pPhyDisk—blocksize;//每个数据块的实际大小
}PhyDisk;
```

(1) JBOD 方式

因为 JBOD 不采用任何的冗余算法，所以逻辑磁盘和物理磁盘的地址实际上是完全一样的，因为 pCacheTask 无需做任何改动，直接传给 Initiator 模块即可。在这种模式下 Vdisk 模块实际上只是对 Cache 模块发下来的任务起缓存作用，并不对地址做任何修改。

因为不采用任何的冗余算法，不能提供任何的容错性，因此 Vdisk 模块在磁盘发生错误时，出错信息会直接向上发到 Cache 模块和 Target 模块，最后告诉 Host 的操作系统，本次读写任务发生错误，由操作系统做相应的出错处理。

(2) Spanning 方式

Spanning 是在逻辑上把几个物理磁盘一个接一个串联到一起，一个逻辑磁盘可能是有多个物理磁盘构成，需要地址转换。寻址地址的算法如下所示

```
for(I=0;I<MaxPhyDiskCount;i++)
{
pInitiatorTask—rwdev=PhyDisk[i];
pPhyDisk = GetPhyDiskPointer (pInitiatorTask—rwdev);
if((nTempStart<=pCacheTask.task—blockstart)&&(pCacheTask.task—blockstart+pCacheTask.
task—blockcount<=pPhyDisk->pPhyDisk—block-
```

count+nTempStart))

break;//找到了实际读写的物理磁盘,退出循环

nTempStart+=pPhyDisk->pPhyDisk—blockcount;

}

pInitiatorTask. task—blockstart = pCacheTask. task—blockstart-nTempStart;

pInitiatorTask. task—blockcount = pCacheTask. task—blockcount;

因为不采用任何的冗余算法,不能提供任何的容错性,因此在 Vdisk 在磁盘发生错误时,错误处理同 JBOD 方式。

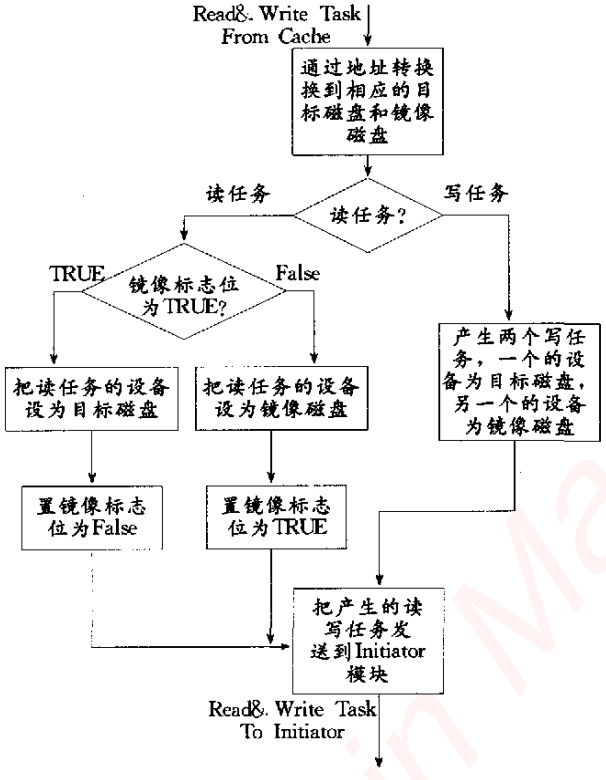

图 2 Vdisk 模块中 RAID1

(3)RAID0

RAID0 将数据块分成多个段,那么逻辑地址连续的数据块就必定分布在不同的物理磁盘上,需要 Vdisk 模块进行地址转换[4]。寻址地址的算法如下所示:

PInitiatorTask — rwdev = pCacheTask. task — blockstart%pPhyDiskCount;

PInitiatorTask. task—blockstart = pCacheTask. task—blockstart/pPhyDiskCount;

PinitiatorTask. task—blockcount = pCacheTask. task—blockcount;

因为 RAID0 不采用任何的冗余算法,不能提供任何的容错性,因此在 Vdisk 在磁盘发生错误时,错误处理同 JBOD 方式。

(4) RAID1

RAID1 中的数据块以镜像的方式分别存储在两组磁盘中,以达到保护数据的目的,因此对于目标磁盘(非镜像磁盘)的读写任务而言,无需作地址转换。由于镜像磁盘与目标磁盘结构、大小等完全一致,因此对于镜像磁盘的读写任务也无须作地址转换,只要把任务的 pCacheTask. task—rwdev 磁盘设备号改为镜像磁盘的序号就可以。

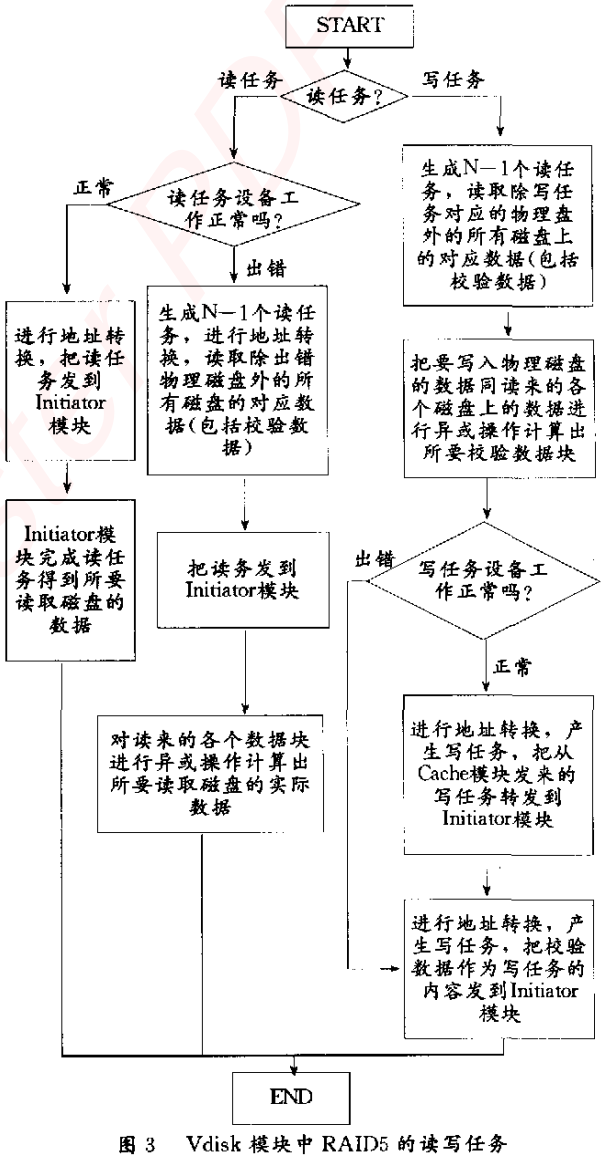

图 3 Vdisk 模块中 RAID5 的读写任务产生及出错处理流程图

然而,由于 RAID1 中的数据在磁盘阵列中有完全相同的备份。为了提高 RAID 系统的读写速度和容错性,读写任务的产生及处理方式不同于上述 RAID 级别。流程图如图 2 所示。

在实现读任务时,为了实现并行操作,程序设置了一个标

志位为 MirrorFlag。如果它的值为 True，表明上次多任务是从镜像磁盘读的，则当前读任务就从目标磁盘读，同时设 MirrorFlag 为 False。反之，则从镜像磁盘读，同时设 MirrorFlag 为 False。这样做就可以使得多个物理磁盘(包括镜像磁盘和目标磁盘)同时工作，实现了 RAID1 的并行化，回快了 I/O 的响应速度。

在实现写任务时，为了保证镜像磁盘和目标磁盘完全一致，Vdisk 模块在接受到由 Cache 模块发出的写任务后，自身产生两个写任务发往 Initiator 模块，一个读写设备为目标磁盘，一个读写设备镜像磁盘。同时 Vdisk 只有从 Initiator 模块受到来自于目标磁盘和镜像磁盘写任务的两个 reply 时，才告知 Cache 模块，写任务已经结束。

当在执行读任务时，Vdisk 模块受到 Initiator 模块发出的出错信息后，并不直接传递给 Cache 模块，而是再发起一个读任务，读写设备为拥有镜像数据的那个物理磁盘。当目标磁盘损坏的时候，那再发起一个读任务的设备就是镜像磁盘；若镜像磁盘损坏的时候，那再发起一个读任务的设备就是目标磁盘。如果后一个读任务成功，则 Vdisk 模块告诉 Cache 模块本次读任务成功，同时告知配置模块一个物理磁盘已经损坏，RAID 等级降为 DEGRADE 模式，不再提供容错性。若后一个读任务失败，Vdisk 模块告诉 Cache 模块本次读任务失败，同时告知配置模块镜像磁盘和目标磁盘已经损坏，无法适用。

当在执行写任务时，Vdisk 模块受到 Initiator 模块发出的来自镜像磁盘和目标磁盘出错信息后，Vdisk 模块告诉 Cache 模块，本次写任务失败，镜像磁盘和目标磁盘已经损坏，无法适用。若有一个消息告知写操作成功的话，模块告诉 Cache 模块本次写任务成功，同时告知配置模块一个物理磁盘已经损坏，RAID 等级降为 DEGRADE 模式[3]。

(5)RAID0+1

RAID0+1 实际上是 RAID0 和 RAID1 的综合，因此地址转换算法同 RAID0 一致，而产生读写任务的算法同 RAID1 一致，在此不再述。

(6)RAID5

RAID5 分散校验并行读写采用了奇偶校验码作为冗余信息，把校验数据分布在工作磁盘本身上，结果每个磁盘可同时从事读或者写而不必等待校验盘空闲。因此 Vdisk 模块需要进行地址转换[4]。在多个物理磁盘中的每一行总有一个磁盘的数据是校验数据。其中 pParityDiskNo 表示当前行中记录校验数据的物理磁盘号。录址地址的算法如下所示：

PInitiatorTask. task—blockstart = pCacheTask. task—blockstart/(pPhyDiskCount－1)；

PParityDiskNo = pInitiatorTask. task — blockstart% pPhyDiskCount；

PInitiatorTask—redev = pCacheTak. task—blockstart% (pPhyDiskCount－1)

If (pInitiatorTask—rwdev>=pParityDiskNo)

PInitiatorTask—rwdev＋＋；

PInitiatorTask. task—blockcount = pCacheTask. task—blockcount；

流程图如图 3 所示。RAID5 无论是读任务还是写任务都会涉及到产生多个子任务，因此 Vdisk 模块中必须有充分的缓存空间供存放这些数据以及计算奇偶校验数据。由于读写任务都不再是单一的单个任务，因此需要等到所有相关的子任务结束以后才能真正确认一个读(写)任务已经成功或失败。如果在一个任务的进行过程中，从 Initiator 模块收到了两个或两个以上的出错信息则告知配置模块 RAID5 磁盘组已经损坏。

四、结束语

本文工作是在我们原开发系统基础上，对 RAID 管理级别和优化算法的进一步完善。在整个系统中，模块中 RAID 级别实现算法的优劣的关系到 RAID 系统速度、吞吐率和鲁棒性的关键因素，因此将来的工作需要对上述算法和差错控制算法进行不断的完善和优化，以期达到高速数据传输速率和较好系统稳定性。

参考文献

[1]Paul Massiglia. The RAID Book—— A Storage System Technology Handbook, 6th edition, RAID Advisory Board, 1997

[2]WindRiver. VxWorks Programmer'sGuide, 5. 4 1999

[3] M. Holland, G. A. Gibson and D. P. Soeoprek. Fast on—line failure recovery in redundant disk array, IEEE 23th international symposium of fault—tolerance computing, 1993, pp422—431

[4] 高峻，吴智铭，蒋智平，"RAID 系统设计中的一个关键技术问题"，电子计算机与外部设备，2002，24(1)：5—8

[5] 尚涛，施亮，v 吴智铭，"RAID 系统的结构化模型"，微型电脑应用，2001，17(1)：29—31

(收到日期：2003 年 1 月 28 日)

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计
32. Modbus 协议在串口通讯中的研究及应用
33. 高可用的磁盘阵列 Cache 的设计和实现
34. RAID 阵列中高速 Cache 管理的优化

35. 一种新的基于 RAID 的 CACHE 技术研究与实现
36. 基于 PCIE-104 总线的高速数据接口设计
37. 基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计
38. 北斗卫星系统在海洋工程中的应用
39. 北斗卫星系统在远洋船舶上应用的研究
40. 基于 CPCI 总线的红外实时信号处理系统
41. 硬件实现 RAID 与软件实现 RAID 的比较
42. 基于 PCI Express 总线系统的热插拔设计
43. 基于 RAID5 的磁盘阵列 Cache 的研究与实现
44. 基于 PCI 总线的 MPEG2 码流播放卡驱动程序开发
45. 基于磁盘异或引擎的 RAID5 小写性能优化
46. 基于 IEEE1588 的时钟同步技术研究
47. 基于 Davinci 平台的 SD 卡读写优化
48. 基于 PCI 总线的图像处理及传输系统的设计
49. 串口和以太网通信技术在油液在线监测系统中的应用
50. USB30 数据传输协议分析及实现
51. IEEE 1588 协议在工业以太网中的实现
52. 基于 USB30 的设备自定义请求实现方法
53. IEEE1588 协议在网络测控系统中的应用
54. USB30 物理层中弹性缓冲的设计与实现
55. USB30 的高速信息传输瓶颈研究
56. 基于 IPv6 的 UDP 通信的实现
57. 一种基于 IPv6 的流媒体传送方案研究与实现
58. 基于 IPv4-IPv6 双栈的 MODBUS-TCP 协议实现
59. RS485CAN 网关设计与实现
60. MVB 周期信息的实时调度
61. RS485 和 PROFINET 网关设计
62. 基于 IPv6 的 Socket 通信的实现
63. MVB 网络重复器的设计
64. 一种新型 MVB 通信板的探究
65. 具有 MVB 接口的输入输出设备的分析
66. 基于 STM32 的 GSM 模块综合应用
67. 基于 ARM7 的 MVB CAN 网关设计
68. 机车车辆的 MVB CAN 总线网关设计
69. 智能变电站冗余网络中 IEEE1588 协议的应用
70. CAN 总线的浅析 CANopen 协议
71. 基于 CANopen 协议实现多电机系统实时控制
72. 以太网时钟同步协议的研究
73. 基于 CANopen 的列车通信网络实现研究
74. 基于 SJA1000 的 CAN 总线智能控制系统设计
75. 基于 CANopen 的运动控制单元的设计
76. 基于 STM32F107VC 的 IEEE 1588 精密时钟同步分析与实现

77. 分布式控制系统精确时钟同步技术
78. 基于 IEEE 1588 的时钟同步技术在分布式系统中应用
79. 基于 SJA1000 的 CAN 总线通讯模块的实现
80. 嵌入式设备的精确时钟同步技术的研究与实现
81. 基于 SJA1000 的 CAN 网桥设计
82. 基于 CAN 总线分布式温室监控系统的设计与实现
83. 基于 DSP 的 CANopen 通讯协议的实现
84. 基于 PCI9656 控制芯片的高速网卡 DMA 设计
85. 基于以太网及串口的数据采集模块设计
86. MVB1 类设备控制器的 FPGA 设计
87. MVB 接口彩色液晶显示诊断单元的显示应用软件设计
88. IPv6 新型套接字的网络编程剖析
89. 基于规则的 IPv4 源程序到 IPv6 源程序的移植方法
90. MVB 网络接口单元的 SOC 解决方案
91. 基于 IPSec 协议的 IPv6 安全研究
92. 具有 VME 总线的车载安全计算机 MVB 通信板卡
93. SD 卡的传输协议和读写程序
94. 基于 SCTP 的 TLS 应用
95. 基于 IPv6 的静态路由实验设计
96. 基于 MVB 的地铁列车司机显示系统研究
97. 基于参数优化批处理的 TLS 协议
98. SSD 数据结构与算法综述
99. 大容量 NAND Flash 文件系统中的地址映射算法研究
100. 基于 MVB 总线的动车组门控系统的设计与仿真研究
101. 多功能车辆总线 MVB 控制
102. 基于 LabVIEW 的 MVB 和 WTB 帧解码方法
103. 基于双 FPGA 的 MVB 通用接口研制
104. RAID 中 Cache 的设计与实现

VxWorks:

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库

8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析
22. 基于 VxBus 的高速数据采集卡驱动程序开发
23. Vxworks 下的冗余 CAN 通讯模块设计
24. WindML 工业平台下开发 S1d13506 驱动及显示功能的实现
25. WindML 中 Mesa 的应用
26. VxWorks 下图形用户界面开发中双缓冲技术应用
27. VxWorks 上的一种 GUI 系统的设计与实现
28. VxWorks 环境下 socket 的实现
29. VxWorks 的 WindML 图形界面程序的框架分析
30. VxWorks 实时操作系统及其在 PC104 下以太网编程的应用
31. 实时操作系统任务调度策略的研究与设计
32. 军事指挥系统中 VxWorks 下汉字显示技术
33. 基于 VxWorks 实时控制系统中文交互界面开发平台
34. 基于 VxWorks 操作系统的 WindML 图形操控界面实现方法
35. 基于 GPU FPGA 芯片原型的 VxWorks 下驱动软件开发
36. VxWorks 下的多串口卡设计
37. VxWorks 内存管理机制的研究
38. T9 输入法在 Tilcon 下的实现
39. 基于 VxWorks 的 WindML 图形界面开发方法
40. 基于 Tilcon 的 IO 控制板可视化测试软件的设计和实现
41. 基于 VxWorks 的通信服务器实时多任务软件设计
42. 基于 VXWORKS 的 RS485MVB 网关的设计与实现
43. 实时操作系统 VxWorks 在微机保护中的应用
44. 基于 VxWorks 的多任务程序设计及通信管理
45. 基于 Tilcon 的 VxWorks 图形界面开发技术
46. 嵌入式图形系统 Tilcon 及应用研究
47. 基于 VxWorks 的数据采集与重演软件的图形界面的设计与实现
48. 基于嵌入式的 Tilcon 用户图形界面设计与开发
49. 基于 Tilcon 的交互式多页面的设计

50. 基于 Tilcon 的嵌入式系统人机界面开发技术
51. 基于 Tilcon 的指控系统多任务人机交互软件设计
52. 基于 Tilcon 航海标绘台界面设计
53. 基于 Tornado 和 Tilcon 的嵌入式 GIS 图形编辑软件的开发
54. VxWorks 环境下内存文件系统的应用
55. VxWorks 下的多重定时器设计
56. Freescale 的 MPC8641D 的 VxWorks BSP
57. VxWorks 实验五[时间片轮转调度]
58. 解决 VmWare 下下载大型工程.out 出现 WTX Error 0x100de 的问题
59. 基于 VxWorks 系统的 MiniGUI 图形界面开发
60. VxWorks BSP 开发中的 PCI 配置方法
61. VxWorks 在 S3C2410 上的 BSP 设计
62. VxWorks 操作系统中 PCI 总线驱动程序的设计与实现
63. VxWorks 概述
64. 基于 AT91RM9200 的 VxWorks END 网络驱动开发
65. 基于 EBD9200 的 VxWorks BSP 设计和实现
66. 基于 VxWorks 的 BSP 技术分析
67. ARM LPC2210 的 VxWorks BSP 源码
68. 基于 LPC2210 的 VxWorks BSP 移植
69. 基于 VxWorks 平台的 SCTP 协议软件设计实现
70. VxWorks 快速启动的实现方法[上电到应用程序 1 秒]

Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版

16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现
24. 掌握 Android 7.0 新增特性 Quick Settings
25. Android 应用逆向分析方法研究
26. Android 操作系统的课程教学
27. Android 智能手机操作系统的研究
28. Android 英文朗读功能的实现
29. 基于 Yocto 订制嵌入式 Linux 发行版
30. 基于嵌入式 Linux 的网络设备驱动设计与实现
31. 如何高效学习嵌入式
32. 基于 Android 平台的 GPS 定位系统的设计与实现
33. LINUX ARM 下的 USB 驱动开发
34. Linux 下基于 I2C 协议的 RTC 驱动开发
35. 嵌入式下 Linux 系统设备驱动程序的开发
36. 基于嵌入式 Linux 的 SD 卡驱动程序的设计与实现
37. Linux 系统中进程调度策略
38. 嵌入式 Linux 实时性方法
39. 基于实时 Linux 计算机联锁系统实时性分析与改进
40. 基于嵌入式 Linux 下的 USB30 驱动程序开发方法研究
41. Android 手机应用开发之音乐资源播放器
42. Linux 下以太网的 IPv6 隧道技术的实现
43. Research and design of mobile learning platform based on Android
44. 基于 linux 和 Qt 的串口通信调试器调的设计及应用
45. 在 Linux 平台上基于 QT 的动态图像采集系统的设计
46. 基于 Android 平台的医护查房系统的研究与设计
47. 基于 Android 平台的软件自动化监控工具的设计开发
48. 基于 Android 的视频软硬解码及渲染的对比研究与实现
49. 基于 Android 移动设备的加速度传感器技术研究
50. 基于 Android 系统振动测试仪研究
51. 基于缓存竞争优化的 Linux 进程调度策略
52. Linux 基于 W83697 和 W83977 的 UART 串口驱动开发文档
53. 基于 AT91RM9200 的嵌入式 Linux 系统的移植与实现
54. 路由信息协议在 Linux 平台上的实现
55. Linux 下 IPv6 高级路由器的实现
56. 基于 Android 平台的嵌入式视频监控系统设计

Windows CE:

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究
17. Windows CE 环境下无线网卡的自动安装
18. 基于 Windows CE 的可视电话的研究与实现
19. 基于 WinCE 的嵌入式图像采集系统设计
20. 基于 ARM 与 WinCE 的掌纹鉴别系统
21. DCOM 协议在网络冗余环境下的应用
22. Windows XP Embedded 在变电站通信管理机中的应用
23. XPE 在多功能显控台上的开发与应用
24. 基于 Windows XP Embedded 的 LKJ2000 仿真系统设计与实现
25. 虚拟仪器的 Windows XP Embedded 操作系统开发
26. 基于 EVC 的嵌入式导航电子地图设计
27. 基于 XPEmbedded 的警务区 SMS 指挥平台的设计与实现
28. 基于 XPE 的数字残币兑换工具开发
29. Windows CENET 下 ADC 驱动开发设计
30. Windows CE 下 USB 设备流驱动开发与设计
31. Windows 驱动程序设计
32. 基于 Windows CE 的 GPS 应用
33. 基于 Windows CE 下大像素图像分块显示算法的研究
34. 基于 Windows CE 的数控软件开发与实现
35. NAND FLASH 在 WINCENET 系统中的应用设计

PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植
12. 基于 PowerPC 双核处理器嵌入式系统 UBoot 移植
13. 基于 PowerPC 的雷达通用处理机设计
14. PowerPC 平台引导加载程序的移植
15. 基于 PowerPC 嵌入式内核的多串口通信扩展设计
16. 基于 PowerPC 的多网口系统抗干扰设计
17. 基于 MPC860T 与 VxWorks 的图形界面设计
18. 基于 MPC8260 处理器的 PPMC 系统
19. 基于 PowerPC 的控制器研究与设计
20. 基于 PowerPC 的模拟量输入接口扩展
21. 基于 PowerPC 的车载通信系统设计
22. 基于 PowerPC 的嵌入式系统中通用 IO 口的扩展方法
23. 基于 PowerPC440GP 型微控制器的嵌入式系统设计与研究
24. 基于双 PowerPC 7447A 处理器的嵌入式系统硬件设计
25. 基于 PowerPC603e 通用处理模块的设计与实现
26. 嵌入式微机 MPC555 驻留片内监控器的开发与实现
27. 基于 PowerPC 和 DSP 的电能质量在线监测装置的研制
28. 基于 PowerPC 架构多核处理器嵌入式系统硬件设计
29. 基于 PowerPC 的多屏系统设计
30. 基于 PowerPC 的嵌入式 SMP 系统设计
31. 基于 MPC850 的多功能通信管理器
32. 基于 MPC8640D 处理系统的技术研究
33. 基于双核 MPC8641D 处理器的计算机模块设计
34. 基于 MPC8641D 处理器的对称多处理技术研究
35. PowerPC 处理器原理

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植
15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现
16. 基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计
17. ARM S3C2440 Linux ADC 驱动
18. ARM S3C2440 Linux 触摸屏驱动
19. Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计
20. Nand Flash 启动模式下的 Uboot 移植
21. 基于 ARM 处理器的 UART 设计
22. ARM CortexM3 处理器故障的分析与处理
23. ARM 微处理器启动和调试浅析
24. 基于 ARM 系统下映像文件的执行与中断运行机制的实现
25. 中断调用方式的 ARM 二次开发接口设计
26. ARM11 嵌入式系统 Linux 下 LCD 的驱动设计
27. Uboot 在 S3C2440 上的移植
28. 基于 ARM11 的嵌入式无线视频终端的设计
29. 基于 S3C6410 的 Uboot 分析与移植
30. 基于 ARM 嵌入式系统的高保真无损音乐播放器设计
31. UBoot 在 Mini6410 上的移植
32. 基于 ARM11 的嵌入式 Linux NAND FLASH 模拟 U 盘挂载分析与实现
33. 基于 ARM11 的电源完整性分析
34. 基于 ARM S3C6410 的 uboot 分析与移植
35. 基于 S5PC100 移动视频监控终端的设计与实现
36. UBoot 在 AT91RM9200 上的移植简析
37. 基于工控级 AT91RM9200 开发板的 UBoot 移植分析

38. 基于 ARM11 和 Zigbee 的人员定位防丢器
39. 基于 NAND FLASH 的嵌入式系统启动速度的研究
40. μCOSⅡ在 ARM7 上的移植
41. 基于 ARM11 的嵌入式视频采集系统设计
42. 基于 ARM11 的视频监控系统设计
43. μC-OSII 在 LPC2210 上的移植研究

Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计
10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究
14. MIPS 架构计算机平台的支持固件研究
15. 基于 UEFI 固件的攻击验证技术研究
16. 基于 UEFI 的 Application 和 Driver 的分析与开发
17. 基于 UEFI 的可信 BIOS 研究与实现
18. 基于 UEFI 的国产计算机平台 BIOS 研究
19. 基于 UEFI 的安全模块设计分析
20. 基于 FPGA NiosⅡ的等精度频率计设计
21. 基于 FPGA 的 SOPC 设计
22. 基于 SOPC 基本信号产生器的设计与实现
23. 基于龙芯平台的 PMON 研究与开发
24. 基于 X86 平台的嵌入式 BIOS 可配置设计
25. 基于龙芯 2F 架构的 PMON 分析与优化
26. CPU 与 GPU 之间接口电路的设计与实现
27. 基于龙芯 1A 平台的 PMON 源码编译和启动分析
28. 基于 PC104 工控机的嵌入式直流监控装置的设计
29. GPGPU 技术研究与发展
30. GPU 实现的高速 FIR 数字滤波算法

31. 一种基于 CPUGPU 异构计算的混合编程模型
32. 面向 OpenCL 模型的 GPU 性能优化
33. 基于 GPU 的 FDTD 算法
34. 基于 GPU 的瑕疵检测
35. 基于 GPU 通用计算的分析与研究
36. 面向 OpenCL 架构的 GPGPU 量化性能模型
37. 基于 OpenCL 的图像积分图算法优化研究
38. 基于 OpenCL 的均值平移算法在多个众核平台的性能优化研究
39. 基于 OpenCL 的异构系统并行编程
40. 嵌入式系统中热备份双机切换技术研究
41. EFI-Tiano 环境下的 AES 算法应用模型
42. EFI 及其安全性研究
43. 基于 UEFI Shell 的 PreOS Application 的开发与研究
44. UEFI Bootkit 模型与分析
45. UEFI 计算机系统快速调试方法的实现
46. 基于 EFI 系统的多文件系统解决方案
47. 基于 UEFI 的可信 Tiano 设计与研究

Programming:

1. 计算机软件基础数据结构 - 算法
2. 高级数据结构对算法的优化
3. 零基础学算法
4. Linux 环境下基于 TCP 的 Socket 编程浅析
5. Linux 环境下基于 UDP 的 socket 编程浅析
6. 基于 Socket 的网络编程技术及其实现
7. 数据结构考题 - 第 1 章 绪论
8. 数据结构考题 - 第 2 章 线性表
9. 数据结构考题 - 第 2 章 线性表 - 答案
10. 基于小波变换与偏微分方程的图像分解及边缘检测
11. 基于图像能量的布匹瑕疵检测方法
12. 基于 OpenCL 的拉普拉斯图像增强算法优化研究
13. 异构平台上基于 OpenCL 的 FFT 实现与优化
14. 数据结构考题 - 第 4 章 串
15. 数据结构考题 - 第 4 章 串答案
16. 用 IPv6 编程接口实现有连接通信的方法
17. 一种战棋游戏的 AI 算法设计与实现浅析
18. 基于 TLS 协议的 ECC 扩展研究

FPGA / CPLD:

1. 一种基于并行处理器的快速车道线检测系统及 FPGA 实现
2. 基于 FPGA 和 DSP 的 DBF 实现
3. 高速浮点运算单元的 FPGA 实现
4. DLMS 算法的脉动阵结构设计及 FPGA 实现
5. 一种基于 FPGA 的 3DES 加密算法实现
6. 可编程 FIR 滤波器的 FPGA 实现
7. 基于 FPGA 的 AES 加密算法的高速实现
8. 基于 FPGA 的精确时钟同步方法
9. 应用分布式算法在 FPGA 平台实现 FIR 低通滤波器
10. 流水线技术在用 FPGA 实现高速 DSP 运算中的应用
11. 基于 FPGA 的 CAN 总线通信节点设计
12. 基于 FPGA 的高速时钟数据恢复电路的实现
13. 基于 FPGA 的高阶高速 FIR 滤波器设计与实现
14. 基于 FPGA 高效实现 FIR 滤波器的研究
15. FPGA 的 VHDL 设计策略
16. 用 FPGA 实现串口通信的设计
17. GPIB 接口的 FPGA 实现
18. 一种基于 FPGA 的 FFT 阵列处理器
19. 基于 FPGA 的 FFT 信号处理器的硬件实现
20. CPLD 在 CAN 通讯卡中的应用
21. 用 CPLD 实现同步串口与异步串口的转换
22. CPLD 在 LED 网络控制器中的应用
23. 基于 CPLD 的双口 RAM 设计与应用
24. CPLD 在有源电力滤波器中的应用