# 基于 Android 的视频软硬解码及渲染的对比研究与实现

李炜锋，于鸿洋

（电子科技大学 电子科学技术研究院，四川 成都 611731）

【摘　要】因为 Android 设备的硬件平台差异比较大，所以不同硬件平台的 Android 设备支持的视频处理方式也不一样。在 CPU 为高通 MSM8260 的 Android 手机平台下对比了 FFmpeg 实现的软件解码和 OpenMax 实现的硬件解码的效率，同时也对不同渲染方式的效率做了对比。结果表明，硬件解码在对高清视频处理时比有 NEON 优化的软件解码效率高了 30% 左右，不同的渲染方式效率也不一样。

【关键词】软件解码；硬件解码；OpenMax；Android；FFmpeg

【中图分类号】TN911.73　　【文献标志码】A

## Research and Implementation of Software Hardware Video Decoding with Different Rendering Methods in Android

LI Weifeng, YU Hongyang

(*Research Institute of Electronic Science and Technology, University of Electronic Science and Technology of China, Chengdu 611731, China*)

【**Abstract**】Because of the significant difference among hardware platform for Android devices, the method of decoding video differs on different hardware platform. A comparison of efficiency between software video decoding with FFmpeg and hardware video decoding with OpenMax in Android Mobile Phone Platform based on Qualcomm's snapdragon MSM8260 is presented. Furthermore, the efficiency of different render methods is also demonstrated. The conclusion is that the efficiency of hardware decoding on high resolution videos is 30% greater than the efficiency of software decoding with NEON optimization. Besides, the efficiency of different render methods also varies.

【**Key words**】software video decoding; hardware video decoding; OpenMax; Android; FFmpeg

Android 系统是一种基于 Linux 的自由及开放源代码的操作平台[1-3]，它具有开放、不受运营商束缚、硬件选择多等特点。随着搭载 Android 系统的智能手机用户的快速增长，手机的硬件性能也有了显著的进步。如今，手机都是向大屏幕高分辨率方向发展，同时人们对高品质数字视频的要求也越来越高。是否能够流畅的播放高清视频已经成为播放器优劣的一个标准。

软件解码是指通过软件让 CPU 进行视频解码处理，直接通过 CPU 来对视频处理进行大量的运算。软件解码的移植性好，可以实现各种各样的解码器，可以满足很多用户观看视频的需求。而硬件解码则是不依赖于 CPU，通过专用的设备单独完成视频解码。现在的高清解码芯片都是主芯片 + DSP 结构，解码的工作都是通过 DSP 来做。芯片中 DSP 硬件编解码的能力通过 OpenMax[4] 标准接口呈现出来，提供上层播放器来使用。

为了分析 Android 手机上音视频播放的效率问题，本文在 CPU 为高通 MSM8260 的手机平台上进行研究，设计了一款支持软件解码和硬件解码并且具有不同渲染方式的播放器。然后先对播放器基本功能进行了验证，最后对解码和渲染的效率进行了对比。

## 1　视频解码

### 1.1　播放器的设计

播放器的原型一般如图 1 所示。数据源可以是 http://、rtsp://、本地地址或者本地文件描述符 fd 等。对数据源中的数据进行解复用得到视频数据和音频数据并送至相对应的解码器进行真正意义上的解码。解码之后音频得到的是 PCM 数据，视频得到的一般是 YUV 数据。如果平台支持 YUV 数据的直接渲染，则只要将视频和音频做同步处理并输出即可。如果不支持 YUV 直接显示，则需要在同步输出之前对视频数据进行颜色空间转换。

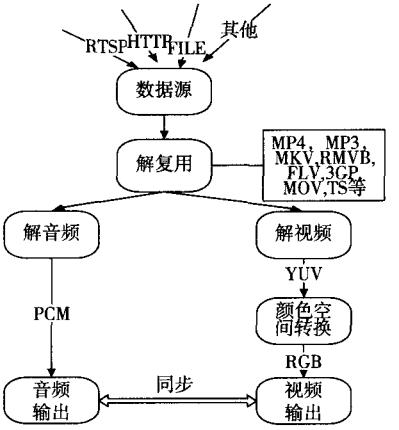


图 1　播放器的原型

为了对视频软硬解码效率和渲染效率做对比研究，本文中的视频播放器初始化的设计流程如图 2 所示。一般而言，硬件解码速度快而且不占用 CPU 资源，如果手机平台支持硬件解码则优先选择硬件解码，否则就考虑 FFmpeg 的软件解码。由于 FFmpeg 软件解码部分自带有 NEON 指令集优化，因此软件解码时还需考虑手机平台是否支持 NEON 指令集优化。这里采用载入不同的库来区分 Andriod 平台是否支持 NEON。Cpufeatures 是 Android 自带的一个用来检测 CPU 类型的静态库，通过这个静态库我们可以编写简单的程序来检测平台是否支持 NEON。

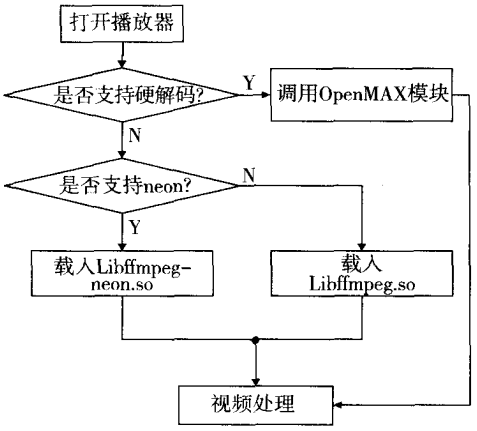


图 2　视频播放器初始化的流程

### 1.2　软件解码

本文中软件解码主要是使用 FFmpeg，它是一个开源跨平台的多媒体数据解决方案。FFmpeg 是由 C 语言编写的，因此需要使用 JNI 技术（Java Native Interface，中文为 Java 本地调用，一种允许 Java 代码和其他语言（如 C、C++）写的代码进行交互的技术）将其移植到 Android 上来。图 3 是本文 Android 播放器的 FFmpeg 软件解码的调用过程。

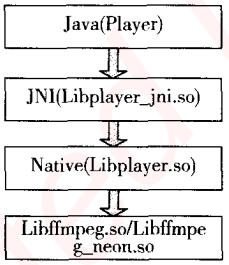


图 3　FFmpeg 软件解码的调用过程

Java 层对应的是 Player 类，该类是播放器在 Java 层的实现，主要负责界面等功能，该类中的某些方法（如播放、暂停等）由 Native 层进行具体实现。JNI 层链接着 Java 层和 Native 层，它对应的是 libffmpeg_jni. so 动态库，该动态库是专门给 Java 层提供 API 的。播放器解码等功能的具体实现是在 Native 层的 libplayer. so 动态库中，该 libplayer. so 动态库由 FFmpeg 源码编译成的动态库（libffmpeg. so 或者是 libffmpeg-neon. so）跟编写的代码重新封装而成。因为编写的代码调用了 FFmpeg 中的相关接口，所以 FFmpeg 源码编译成的动态库是调用过程中最底层的库。

### 1.3　硬件解码

OpenMax 是跨平台的应用程序接口 API，是一个多媒体应用程序的标准，其中 OpenMax IL（集成层）目前已经成为多媒体框架标准。本文中利用 Android 自带的作为多媒体引擎插件[5]的 OpenMax IL 来实现 Android 平台下视频的解码功能。

OpenMax IL 的基本单元是组件 component，它的数据主要通过 port 来进行交互，port 通过和 OMXCodec 共享 buffer 来进行解码。本文硬件解码的抽象结构如图 4 所示。OMXCodec 调用 EmptyThisBuffer 传递未解码的 buffer 数据给 component，component 从 buffer 中读完数据会调用 EmptyBufferDone 来通知 OMXCodec 去接收新的视频数据。OMXCodec 使用 OMX_FillThisBuffer 传递空的 buffer 给 component 用于存储解码后的帧，component 收到该命令后将解码好的帧数据复制到该 buffer 上，然后调用 FillBufferDone 通知 OMXCodec 并让其输出。视频正常播放时，从第一次调用 EmptyThisBuffer 开始计时直到最后一次调用 FillBufferDone，计时得到的时间便是解码对应数量视频帧的总时间。

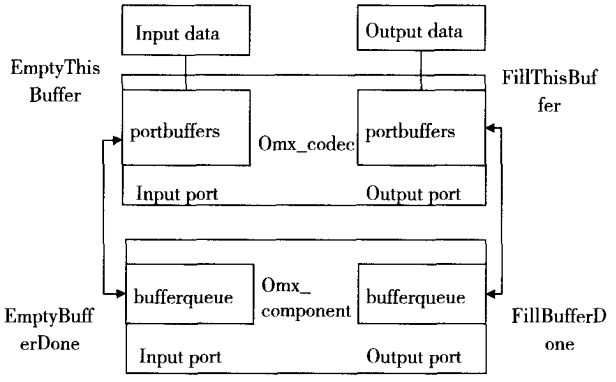


图 4　硬件解码的抽象结构

## 2　视频数据的显示

### 2.1　颜色空间转换

本文中不管是软件解码还是硬件解码，解码出来的都是 YUV 数据，但是 Android 中现在不支持直接渲染 YUV 数据，必须把 YUV 数据转换成 Android 支持的 RGB 数据来渲染[5-6]。而 YUV 数据转 RGB 数据所消耗的时间是不容忽视的。FFmpeg 虽然有自带的颜色空间转换，

但是效率太低。本文使用的是另一开源方案，该开源方案有 C 语言和汇编两种方法，但是这两种方案都不在 FFmpeg 优化之内。针对测试平台为 ARMV7 + NEON，还可以对此进行 NEON 指令集优化来提高颜色空间转换的效率。可在 Android. mk 文件中加入 LOCAL_ARM_NEON: = true 开启。

### 2.2 渲染

Android 中渲染视频数据有以下几种方式：

1）Java Surface JNI，Java 层创建一个 buffer，将该 buffer 的 handle 传至 Native C 代码中，然后将所要渲染的 RGB 数据 copy 至该 buffer 中，最后通过 JNI 调用 Java 中的相关方法进行渲染。缺点是不如其他方法的效率高。

2）Private C++ API，Android 源码中提供的私有的 C++接口，也是渲染 RGB 数据，效率与 ANativeWindow API 差不多，而且对版本没什么限制。只是 Google 团队不建议使用该私人接口。

3）ANativeWindow API，在 Native 层中直接渲染 RGB 数据，这种方法比较简单且效率高。缺点是只能针对 Android2.3 以后的版本。这里渲染的是 RGB565 格式的数据，相关程序段如下

```
void jni_player_render(JNIEnv * env. jobject thiz,jobject surface)
{
...
ANative Window_setBuffersGeometry(window,etx - >width,ctx - >hetght,WINDOW_FORMAT_RGB_565;  //设置 buffer
If(ANativeWindow_lock(window,&buffer,NULL)! =0){
//锁住 ANativeWindow
pthread_mutex_unlock(&player - >mutex_queue);
...
ANativeWindow_unlockAndpost(window);  //显示
...
}
```

4）OpenGL ES 2.0，当平台有 GPU 的情况下，并且 Android 版本 2.2 之后，可以使用 OpenGL ES 2.0[7] 来显示视频数据。利用 OpenGL ES 2.0 在 Android 底层的 API 接口，将 YUV 数据直接交给 Native 层来显示。它有点特殊，虽然也是将 YUV 数据转 RGB 数据再显示，但是是通过 GPU 来处理的，不同于利用 CPU 来进行颜色空间转换，速度快了很多。glViewport(…)是设置显示窗口的大小和位置，bindTexture(…)是分别绑定 *Y*，*U* 和 *V* 数据，相关程序段如下

```
{
...
glViewport(0,0,wwidth,wheight);
bindTexture(g_texYId,buffer,width,height);
bindTexture(g_texUId,buffer + width * height,width/2,height/2);
bindTexture(g_texVId,buffer + width * height *  /5/4,width/2,height/2);
renderFrame();
...
}
```

## 3 测试与结论

### 3.1 功能验证和效率测试

测试程序是用 Java，C/C++编写，编写环境是 Ubuntu11.10，工具是 ADT(Android Development Tools)22.3.0 - 887826。测试平台是 CPU 为高通 MSM8260(GPU 为 ADRENO220)的 Android 手机，屏幕分辨率为 854 × 480，Android OS 版本为 4.03，内存大小为 768 Mbyte。测试所用的视频分辨率为 1 280 × 720(720 p)，码率为 4 554 kbit/s，每秒 25 帧。测试程序流程图如图 5 所示。

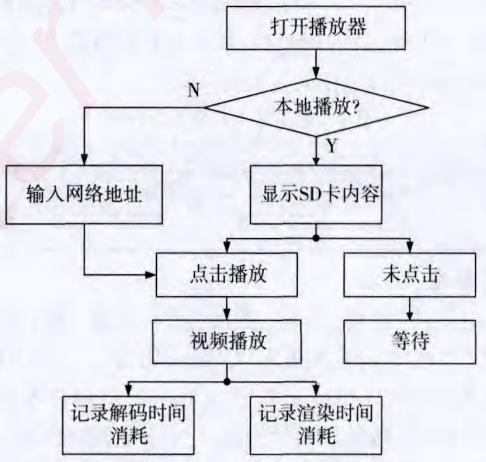


图 5 播放器测试程序流程图

测试 1，测试本文播放器的基本功能，主要是多媒体文件格式，视音频编码方式及网络协议的支持性测试，如表 1。其中，HTTP 用的是 Apache HTTP Server，RTSP 用的是 Darwin Streaming Server。

表 1 播放器基本功能测试

| 多媒体文件格式 | | | 视频编码 | | 音频编码 | | 网络协议 | |
|---|---|---|---|---|---|---|---|---|
| MP4 | MKV | AVI | MPEG - 4 | H. 264 | MP3 | AAC | HTTP | RTSP |
| 支持 | 支持 | 支持 | 支持 | 支持 | 支持 | 支持 | 支持 | 支持 |

测试 2，测试本文播放器各个环节的效率。

1）测试软件解码和硬件解码对 720p 视频的解码速度，结果如表 2 所示。硬件解码速度比有 NEON 优化的软件解码速度快了 30% 左右，是没有 NEON 优化的软件解码时间的 1/3。

表 2　解码每一帧的时间消耗

| 测试视频分辨率 | 720p |
|---|---|
| 硬件解码 | 20 ms |
| 软件解码(NEON) | 28 ms |
| 软件解码(非 NEON) | 60 ms |

2) 720p 测试视频的 YUV 数据转 RGB 数据的时间消耗如表 3 所示,因此颜色空间转换最快的方式应该是同时使用汇编和 NEON 优化。

表 3　YUV 数据转 RGB 数据的时间消耗

| 消耗时间 |
|---|
| NEON 优化 < 非 NEON 优化 |
| 汇编 < C 代码 |

3) 渲染 720 p 视频帧的时间如表 4 所示,渲染 RGB 数据的 Private C++ API 和 ANativeWindow API 两种方法速度都为 7 ms,其中 OpenGL ES 2.0 比较特殊,13 ms 的时间消耗还包括颜色空间转换的时间。

表 4　渲染每一帧需要的时间　　ms

| OpenGL ES 2.0 | Java Surface JNI | Private C++ API | ANativeWindow |
|---|---|---|---|
| 13 | 10 | 7 | 7 |

### 3.2　结论

由测试 1 可得,本文的播放器设计方案支持主流的多媒体文件格式、音视频编码方式和网络协议。由测试 2 中可得,硬件解码和 OpenGL ES 2.0 同时使用是效率最高的。在不支持硬件解码的情况下,对于软件解码或是颜色空间转换,NEON 优化的代码效率比非 NEON 优化的代码效率快了很多。因此,本文的播放器设计方案能满足大部分的音视频播放需求,具有一定的可行性和有效性。图 6 为本文视频播放器的流畅播放截图。

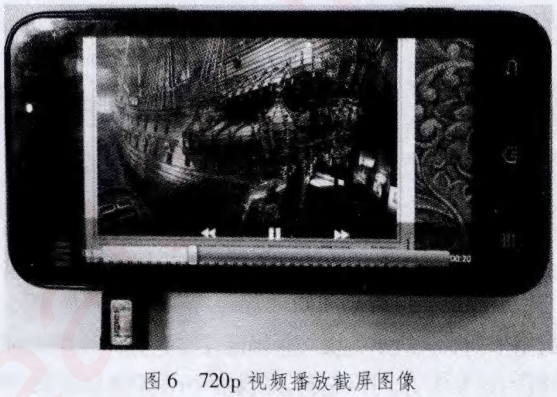

图 6　720p 视频播放截屏图像

## 4　小结

Android 的开放性使得手机设备有了丰富的硬件选择,虽然非 NEON 优化的软件解码和 Private C++ API 渲染方式几乎可以适用于任何 Android 手机设备,但是对于很多 Android 手机设备来说,这种视频播放方式的效率并不高。本文利用 FFmpeg 和 Android 自带的 OpenMax 设计实现了一款 Android 平台的播放器,并对软件解码和硬件解码的效率进行了对比,同时也对多种渲染方式进行了对比,这对于提高 Android 手机平台上高清视频的播放效率是有一定意义的。

**参考文献:**

[1] Android SDK Document [EB/OL]. [2013-11-21]. http://developer.android.com.

[2] 陈熙,于鸿洋. 基于 Android 高清图像的视音频监控终端的实现[J]. 电视技术,2013,37(23):222-225.

[3] 季光献. Android 系统环境下应用前景与开发研究[J]. 软件,2011(10):50-51.

[4] 沈永增,李晓凤,吴冬林. Android 下 OpenMax IL 框架的研究和应用[J]. 计算机应用与软件,2012(8):265-268.

[5] Android open source project [EB/OL]. [2013-12-15]. https://android.googlesource.com/.

[6] 李刚. 疯狂 Android 讲义[M]. 北京:电子工业出版社,2011.

[7] 冯贤全. 基于 Andorid 和 OpenGL 的多媒体播放器的研究[D]. 成都:电子科技大学,2012.

作者简介:

李炜锋(1988— ),硕士生,主研 Android 相关;

于鸿洋(1964— ),副教授,主要研究方向为数字电视广播电视网络技术相关,三维技术和算法。

责任编辑:魏雨博　　收稿日期:2014-01-03

# 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

# 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

# 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

# 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

# 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

# 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

# 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

# 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