异构平台上基于 OpenCL 的 FFT 实现与优化

李　焱[1,2,3]　张云泉[1,2]　王　可[1,2]　赵美超[1,2,3]
（中国科学院软件研究所并行软件与计算科学实验室　北京 100190）[1]
（中国科学院软件研究所计算机科学国家重点实验室　北京 100190）[2]
（中国科学院研究生院　北京 100190）[3]

摘　要　快速傅立叶变换作为 20 世纪公认的最重要的基础算法之一，在大规模科学计算处理、数字信号处理、图形图像仿真等众多领域有着广泛的应用。OpenCL 是首个面向异构系统通用的并行编程标准，为软件开发人员提供了统一的面向异构系统的并行编程环境。首先，在异构平台 Cell 和 GPU 上使用 OpenCL 实现了基于 2 的幂一维 FFT，并对其进行了测试和分析，在 Cell 平台上当数据规模适中时它能够达到 SDK 性能的 65%，当数据规模继续增大时，相对性能有所降低。此外，针对 Nvidia Fermi 平台，手工调优了小因子的 FFT，使其性能接近于 CUFFT 的 140%。

关键词　FFT，OpenCL，Cell，CUDA，GPU，快速傅立叶变换
中图法分类号　TP311　　**文献标识码**　A

Implementation and Optimization of the FFT Using OpenCL on Heterogeneous Platforms

LI Yan[1,2,3]　ZHANG Yun-quan[1,2]　WANG Ke[1,2]　ZHAO Mei-chao[1,2,3]
(Lab. of Parallel Software and Computational Science, ISCAS, Beijing 100190, China)[1]
(State Key Lab. of Computer Science, CAS, Beijing 100190, China)[2]
(Graduate University of Chinese Academy of Sciences, Beijing 100190, China)[3]

Abstract　Fourier methods have revolutionized fields of science and engineering, from astronomy to medical imaging, from seismology to spectroscopy. A fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT) and its inverse. OpenCL (Open Computing Language) is a new framework for writing programs that execute across heterogeneous platforms consisting of CPUs, GPUs, and other processors, and it provides parallel computing using task-based and data-based parallelism. In this paper, we first implemented FFT with OpenCL, then tested and analyzed the performance of it on heterogeneous multi-core platforms like Cell, NVIDIA GPU. The performance we achieved is about 65% of Cell SDK, and 75% of CUDA CUFFT, and it needs to improve in the near further. Furthermore, we acquire unprecedented performance results that nearly 140% of CUFFT on Fermi GPU by exploiting hardware features when the size of FFT is small.

Keywords　FFT, OpenCL, Cell, CUDA, GPU, Fast fourier transform

1　引言

1.1　FFT 的原理和应用

快速傅里叶变换[1-4]（Fast Fourier Transform，FFT），是离散傅里叶变换（Discrete Fourier Transform，DFT）的快速算法，具有广泛的应用，常被用于数字滤波、求解偏微分方程和信号分解。近年来具有 FFT 特点的谱方法已被成功地应用于解决计算流体力学（CFD）问题，而求解 CFD 问题广泛地被应用于气象预报、模拟地球物理学以及量子力学等方面。FFT 在数据分析领域中的应用包括计算机断层成像、数据过滤和重构以及流体结构相互间作用的分析和可视化等。FFT 算法设计的基本思想，就是充分利用 DFT 的周期性和对称性，减少重复的计算量；并把 N 点长序列分成几个短序列，递归地减少每个序列长度从而降低计算量和计算复杂度。

1.2　GPU 和 Cell

硬件方面，集成电路产业一直按照摩尔定律（Moore's Law）发展，计算机硬件环境日新月异，高性能计算机不断挑战更高难度数量级的性能。作为高性能计算机的基本组成部件，CPU、GPU 、IBM Cell 等各种高性能处理器的发展也十分迅猛。其中，GPU 在通用计算方面的发展尤为引人注目。2006 年 11 月，AMD 开始研发“close to metal”，后来演进为 ATI Stream 技术。同时，NVIDIA 的计算统一设备架构（Compute Unified Device Architecture，CUDA）与 G80 显卡同时公开，并于 2010 年 3 月发布了 CUDA 3.0。

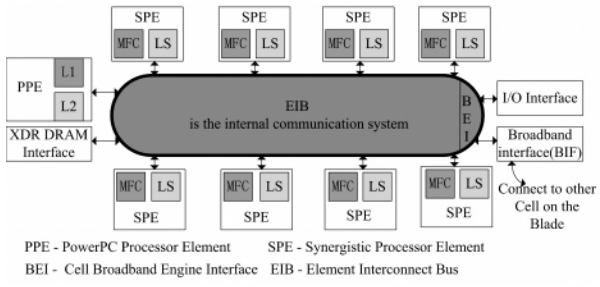

图 1　Cell 硬件结构图

Cell[5-7] 是由 IBM、东芝(Toshiba)和 Sony 共同研发的异构多核处理器，具备多工、多效以及庞大的浮点运算功能。Cell 处理器(如图 1 所示)包括 1 个基于 PowerPC 架构的控制处理单元 Power Processing Element (PPE)以及 8 个 SIMD 的协处理器单元 Synergistic Processing Elements (SPE)。此外，采用高速环形数据总线 Element Interconnect Bus (EIB)连接 PPE、输入输出单元和 SPE，同时还提供了 DMA 指令和控制机制以用于在不同处理单元之间提供高效的通信。SPE 访问主存储器的方法则是利用 DMA 命令在主存储器和没有高速缓存(Cache)的私有本地内存(LS，Local Store)之间移动数据和指令，而不是共享主存储器。

1.3　OpenCL 简介

OpenCL[8] (Open Computing Language，开放计算语言)是首个面向异构系统通用的并行编程标准，具有免费、跨平台的特点和很好的互操作性。OpenCL 为软件开发人员提供了统一的编程环境，便于为高性能计算服务器、桌面计算系统、手持设备编写高效轻便的代码，而且广泛适用于多核处理器(CPU)、图形处理器(GPU)、Cell 类型架构以及数字信号处理器(DSP)等其他并行处理器，因此在游戏、娱乐、科研、医疗等各种领域都有广阔的发展前景。

OpenCL 的架构如图 2 所示，一个主机(Host)可以连接到多个 OpenCL 设备(即 Compute Device)上，这些设备可以是 CPU、GPU、Cell 甚至 DSP 等。每个 OpenCL 计算设备被看作一组计算单元(Compute Unit)的集合，每个计算单元又划分为多个处理部件(Compute Element)。对于 GPU 而言，一个计算单元是指流多处理器(Streaming Multiprocessor)，一个处理部件是指一个流处理器(Streaming Processor)。OpenCL 程序由主机程序(Host program)和内核程序(Kernel)组成。主机程序为 Kernel 定义了上下文并管理 Kernel 代码的执行，Kernel 则在支持 OpenCL 的计算设备(Compute Device)如 GPU、Cell 等上执行具体的计算任务。

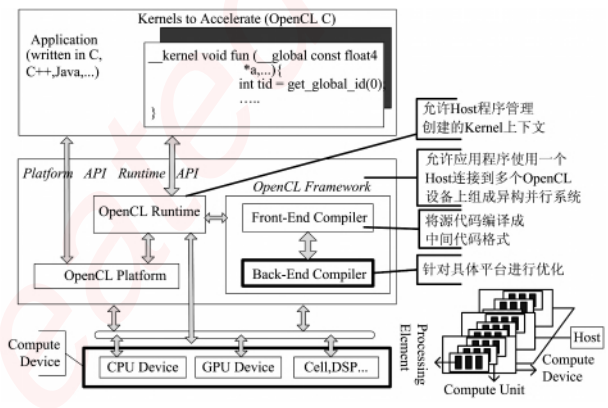

图 2　OpenCL 的框架图[8]

2　基于 OpenCL 的 FFT 算法设计与实现

对于复数序列 $x_0, x_1, \cdots, x_{n-1}$，离散傅里叶变换 DFT 的计算公式为：

$$y_k = \sum_{j=0}^{N-1} \omega_N^{jk} x_j \tag{1}$$

式中，$\omega_N = \exp(-\frac{2\pi i}{N})$，$i = \sqrt{-1}$；$K = 0, 1, \cdots, N-1$

计算每个 y_k 须作 N 次复数乘法及 $N-1$ 次复数加法，要完成整个 DFT 变换共需 N^2 次复数乘法及 $N(N-1)$次复数加法。经典的 FFT 算法是由 Cooley-Tukey[4] 提出的，该算法考虑到 FFT 的周期性，以分治法为策略递归地将长度为 $N = r \times m$ 的 DFT 分解为长度分别为 N_1 和 N_2 的两个较短序列的 DFT。

$$j = j_1 r + j_2 \quad 0 \leqslant j_2 < r, 0 \leqslant j_1 < m$$
$$k = k_1 + k_2 m \quad 0 \leqslant k_1 < m, 0 \leqslant k_2 < r$$

于是：

$$\begin{aligned} y_{k_1+k_2 m} &= \sum_{j_2=0}^{r-1} \sum_{j_1=0}^{m-1} \omega_N^{(k_1+k_2 m)(j_1 r+j_2)} x_{j_1 r+j_2} \\ &= \sum_{j_2=0}^{r-1} \left(\left(\sum_{j_1=0}^{m-1} \omega_N^{k_1 j_1} x_{j_1 r+j_2} \right) \omega_N^{k_1 j_2} \right) \omega_r^{k_2 j_2} \end{aligned} \tag{2}$$

用张量积表示为：

$$Y_l = (W_r \otimes I_m) T_m^N (I_r \otimes W_m) \prod_{N,rX} \tag{3}$$

式中，T_m^N 是一个对角线“旋转因子”矩阵，$\prod_{N,rX}$ 是一种模 r 置换矩阵，这是 FFT 的核心思想，以上式(3)就是著名的 Cooley-Tukey 混合基分解算法。

采用混合基分解 DFT 可将其复数计算量由 N^2 下降到 $N(r+m)$，当 N 为 r 的幂时，计算复杂度降低为 $2r\log_r N$。当 $N=8$ 时，DFT 的计算示意图如图 3 所示。

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \begin{bmatrix} l & l & l & l & l & l & l & l \\ l & u & -i & v & -l & -u & i & -v \\ l & -i & -l & i & l & -i & -l & i \\ l & v & i & u & -l & -v & -i & -u \\ l & -l & l & -l & l & -l & l & -l \\ l & -w & -i & -v & -l & u & i & v \\ l & i & -l & -i & l & i & -l & -i \\ l & -v & i & -u & -l & v & -i & u \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}$$

图 3　8 个点 DFT 计算示意图[3]

其中，$u = \omega_8 = (1-i)/\sqrt{2}$，$v = \omega_8^3 = (-1-i)/\sqrt{2}$。对于以上 $N=8$ 时的计算序列，可以将其分解为 Radix-2 FFT(见图 4)，所对应的蝶式计算图如图 5 所示。

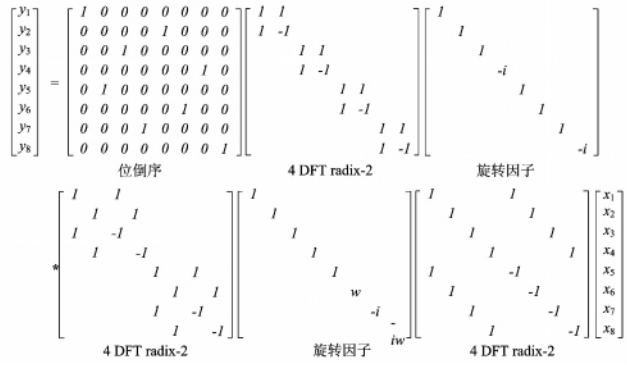

图 4　$N=8$ 时 FFT 计算分解示意图[3]

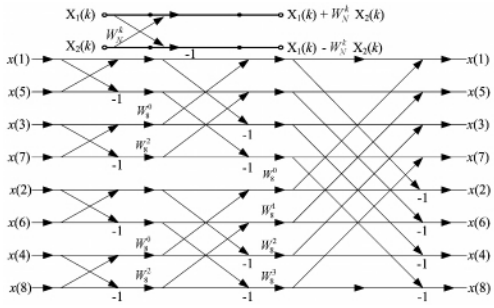

图 5　$N=8$ 时 FFT 蝶式计算图

目前，基于 CPU 版的 FFT 库的研究与发展比较成熟，各大硬件厂商 Intel、IBM 以及 AMD 也相应推出了针对各自平台优化的 FFT 库。FFTW（Fast Fourier Transform in the West）是由麻省理工学院开发的开源免费的自适应优化 FFT 软件包，同时支持共享存储多线程并行和分布式存储 MPI 并行，其运算性能远远领先于目前已有的其它 FFT 软件，甚至优于商用软件。FFTW 遴选了过去近 40 年的各种好的 FFT 算法，包括 Cooley-Tukey 算法以及各种变式、素因子、Rader 算法、分裂基算法等。在软件包安装时通过 Ocaml 语言针对安装平台的硬件特性产生一系列优化了的小因子 FFT（codelets），例如 Radix-2、Radix-3、Radix-5、Radix-8 等。在运行时通过搜索和组合这些小因子 FFT 来选择较优或者最优的针对指定规模的 FFT 算法[13,14]。

但是，针对 GPU 优化的 OpenCL 版 FFT 库在各个平台（包括 Cell、GPU）基本上还处于开发阶段，OpenCL 的应用和使用还处于起步阶段。本文中基于 OpenCL 的 FFT 算法实现采用的是时间抽选 FFT 算法（Decimation in frequency，DIF），先将输入元素重新排列成位序颠倒的次序，然后进行蝶形算法输出自然序列。数据的实部和虚部采用混合存储模式即实部和虚部连续存储。本文所采用的 Cooley-Tukey 算法框架如下：

- 将输入数据按照列优先的顺序存储在 $r\times m$ 的数组中。
- 对矩阵中的每一行数据并行做 FFT radix-r 计算。
- 将矩阵乘以相关的旋转因子。
- 对矩阵进行转置操作。
- 对矩阵中的每一行数据并行地进行 FFT radix-m 计算。

算法框架所对应的图示如图 6 所示。

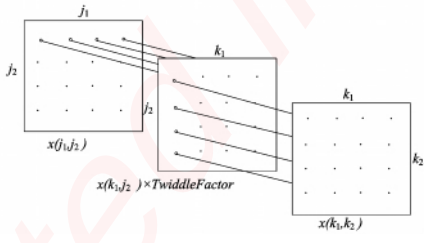

图 6　Cooley-Tukey 算法框架

3　实验结果及分析

3.1　测试环境

本节通过实验分别进行了 Cell SDK 以及 NVIDIA（CUFFT[10]）的 FFT 库与实现的 OpenCl FFT 性能的比较，实验的平台如表 1 所列。AMD 中 SDK 2.1 的 Sample 中 FFT 只支持 1024 规模，而且其对应 ATI GPU（AMD Core Math Library for Graphic Processors，ACML-GPU）上没有可用的 FFT 函数库[9]，由于缺少可比性，因此实验平台暂时没有考虑 AMD GPU。

表 1　实验平台

平台	参数
Cell	Cell 3.2GHZ，2GB Memory，256K LS 32/32 KB L1（data/instruction）Cache，512KB L2 Cache Cell OpenCL 0.1.1＋Cell SDK 3.1 编译器：XLC 9.0
GPU	Tesla C1060 NVIDIA OpenCL 1.0＋ CUDA SDK 3.0 编译器：NVCC 3.0 Fermi（Tesla C2050） NVIDIA OpenCL 1.0＋ CUDA SDK 3.1.1 编译器：NVCC 3.1

测试结果中的时间包括 IO 传输时间、执行时间以及 OpenCL Kernel/CUFFT/Cell libfft 执行上下文的创建时间。由于 Cell 体系结构的异构特点[6,7]譬如支持 16M 的大页表，我们在 Host program 端使用了大页表来减少 TLB 缺失，并且数据内存分配采用 128bytes 地址对齐，MFC 虽然支持自然对齐的 DMA 传输，大小分别为 1，2，4，8 和 16 字节，以及 16 字节整数倍，但当 DMA 传输的源地址和目的地址都是 128 字节对齐且传输数据的大小是 128 字节的倍数时，传输的性能可以达到峰值（DMA 访问带宽是 128bytes/cycle）。考虑到 Radix-2 FFT 的特点，结合 Cell 上 SPE 大容量向量寄存器的特点，在 OpenCL kernel 中数据类型尽量使用 float2 和 float4 类型来提高寄存器的利用率。

3.2　测试结果与分析

OpenCL 在 Cell 平台上的测试结果如图 7 所示，若数据规模为 1024 至 8192，OpenCL 版的 FFT 的性能约为 SDK 的 65%，当数据规模较大如 size 为 65536 时，性能只有 SDK 的 35%左右。在 Kernel 端，虽然 SPE 上没有 Cache，SPE 从容量为 256kB 的 LS 中读取指令和数据，但是，其内部包含一个有 128 条目的 128 位寄存器堆（Register file），而且具有两条（奇、偶）执行流水线，可在单周期内发射并完成两条指令，因此，在 Kernel 中应该尽量使用 float4 类型进行运算和操作。但是 FFT 的实现无论采用 DIT 还是先迭代后进行位序颠倒的频率抽选法（Decimation in frequency，DIF）算法，其内部实现的还是两个元素进行运算和操作，OpenCL 1.0 标准中没有类似于 SPE SIMD 指令中的 shuffle 指令来重组元素，因此寄存器的使用率最多只能达到 50%。由于一个 128 位寄存器刚好能够容纳 2 个 double 类型数据，即一个 double complex（对应于 double2 类型），但是，目前 Cell 上该 OpenCL 版本对 OpenCL 标准中 double2 类型的操作的支持还不是很完善，譬如三角函数，故只能期待在下一版本中进行测试。另外，在 Kernel 端不能有效地使用双缓冲机制（Double buffer）来分摊通信开销，因此，随着计算数据量（size）的增大，势必会影响性能，从图 9 可以看出当数据规模增加时，虽然 Kernel 所占的时间比例明显增加，但性能与 SDK 的差距反而增大。

在 NVIDIA Tesla C1060 平台上的测试结果如图 8 所示，平均性能接近 CUFFT 的 75%。Fermi 是 Nvidia 专门为高性能计算优化的新一代处理器架构，基于该架构的 Tesla GPU 处理器拥有 30 亿颗晶体管、512 个计算内核，双精度性能相

（下转第 296 页）

[10] Wolfram S. Cellular automata as models of complexity[J]. Nature,1984,311:419-424

[11] Pinheiro E,Bianchini R,Dubnicki C. Exploiting redundancy to conserve energy in storage systems[J]. ACM SIGMETRICS Performance Evaluation Review,ACM,2006:15-26

[12] Colarelli D,Grunwald D. Massive arrays of idle disks for storage archives[C]// Proceedings of the 2002 ACM/IEEE Conference on Supercomputing. ACM,2002:1-11

[13] Pinheiro E,Bianchini R. Energy conservation techniques for disk array-based servers[C]// Proceedings of the ACM/IEEE Conference on Supercomputing. ACM,2004:88-95

[14] Narayanan D,Donnelly A,Rowstron A. Write Off-loading:Practical Power Management forEnterprise Storage[C]// the 6th USENIX Conference on File and Storage Technologies (FAST'08). USENIX Association,2008:253-267

[15] Adami C. Introduction to artificial life[M]. New York:Springer-Verlag,1998:94-98

[16] Lee W,Scheuermann P,Vingralek R. File Assignment in Parallel I/O Systems with Minimal Variance of Service Time[J]. IEEE Trans. Computers,2000,49(2):127-140

(上接第 286 页)

对前一代产品 GT200 提升 8 倍,带有 ECC 校验功能,增加了 L1 和 L2 两级缓存,内存带宽提高两倍。Fermi (Tesla C2050)每个 SM 中的寄存器文件数量为 32K[11,12],对于小规模的 FFT,每个线程拥有的寄存器数量非常充足。因此,在 Fermi 平台,我们对小规模的 FFT 完全展开。此外,Fermi 支持高精度的乘加指令 FMA(fused multiply-add)[11],FFT 中复数相乘的操作非常频繁,有效使用 FMA 指令优化复数运算能够改善运算精度和运算效率。线程组成 warp 进行访存优化,减少线程之间的同步,同时通过转置操作优化全局内存的对齐访问,对于 block 内部线程之间尽量使用 local memory (对应于 CUDA 中的 Shared memory)来进行 block 内全局通信。在 Fermi 上通过采用如上技术优化 Radix-8,Radix-64 以及 Radix-512,使其性能大大提升,实验结果如图 10 所示。

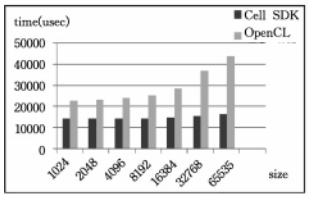

图 7 Cell 平台上的测试结果

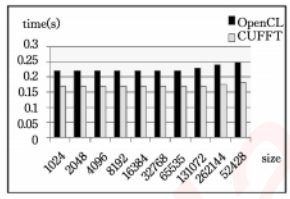

图 8 Tesla C1060 平台上的测试结果

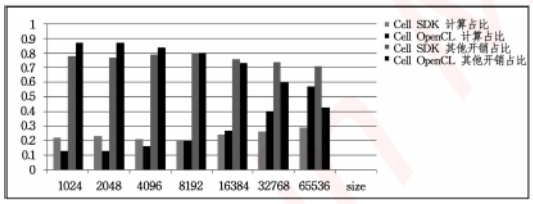

图 9 OpenCL FFT 在 Cell 平台上的计算和开销占比

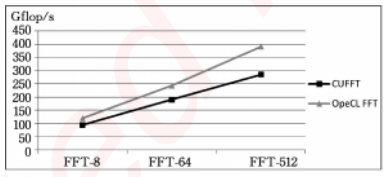

图 10 Fermi 平台上的小规模 FFT 优化结果

结束语 本文测试和分析了面向异构平台基于 OpenCL 的一维 FFT,实验结果表明,在 Cell 平台上当数据规模适中时能达到 SDK 中提供的 FFT 性能的 65%,当数据规模继续增大时,性能有所降低了,仅为 SDK 的 35% 左右。在 NVIDIA 平台上能够到达 CUFFT 性能的 75%,并对小规模的 FFT 在 Fermi 进行了手工调优,性能接近 CUFFT 的 140%。另外,在 Cell 上由于 SPE 具有大容量向量寄存器的特点,下一步工作会利用循环展开和寄存器分块来有效利用寄存器;在 NVIDIA 平台上借鉴 CUDA 程序的优化技术来改善 OpenCL Kernel 中的性能,例如流技术。此外,将针对 Fermi 平台实现和优化更多的小因子 FFT(codelets),并对一定规模大小的 FFT 采用多种搜索策略和匹配 codelets 算法来得到较优的性能。

总之,OpenCL 为软件开发人员提供了统一的面向异构系统的并行编程环境,尤其在 Cell 上,屏蔽了基于 SDK 编程的很多复杂的硬件细节,提高了并行编程的生产率,因此,OpenCL 在并行计算领域具有良好的应用前景。

结束语 本文在异构平台 Cell 和 Nvidia GPU 上使用 OpenCL 实现和优化了一维 FFT,并测试和分析了该版本 FFT 的性能。此外,针对 Fermi 的存储层次和硬件特点,通过手动调优小因子 FFT,使其性能接近 CUFFT 的 140%。

参 考 文 献

[1] Bracewell R N. The Fourier Transform and Its Applications(3rd ed)[M]. McGraw-Hill,1999

[2] Govindaraju N K,Lloyd B,Dotsenko Y,et al. High Performance Discrete Fourier Transforms on Graphics Processors[C]// SC08. November 2008

[3] Volkov V,Kazian B. Fitting FFT onto the G80 architecture[R]. http://www.cs.berkeley.edu/~kubitron/courses/cs258-S08/projects/reports/project6_report.pdf,May 2008

[4] Cooley J W,Tukey J W. An algorithm for the machine calculation of complex Fourier series [J]. Mathematics of Computation,1965,19(90):297-301

[5] IBM Corporation. Cell Broadband Engine Programming Handbook[S]. Version 1.12,Apr. 2009

[6] IBM Corporation. Software Development Kit 2.1 Programmer's Guide[S]. Version 2.1,March 2007

[7] IBM Corporation. Cell Broadband Engine Programming Tutorial [S]. Version 2.0,Dec. 2006

[8] Khronos OpenCL Working Group. The OpenCL Specification [S]. Version 1.0,Oct. 2009

[9] AMD Corporation. AMD Core Math Library for Graphic Processors Release Notes for Version 1.0[S]. March 2009

[10] CUDA Programming Guide Version 3.0:NVIDIA Corporation [S]. Feb. 2010

[11] NVIDIA's Next Generation CUDA Compute Architecture:Fermi. NVIDIA Corporation[S]. 2009

[12] Glaskowsky P N. NVIDIA's Fermi:The First Complete GPU Computing Architecture[Z]. September 2009

[13] Frigo M,Johnson S G. The Design and Implementation of FFTW3 [J]. Proceedings of the IEEE,2005,93 (2):216-231

[14] Frigo M. A Fast Fourier Transform Compiler[C]// Proceedings of the 1999 ACM SIGPLAN Conference on Programming Language Design and Implementation. Feb. 1999

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

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

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

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

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

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

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

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

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.