RT Embedded http://www.kontronn.com

基于 OpenCL 的异构系统并行编程

詹　云，赵新灿，谭同德

（郑州大学 信息工程学院，河南 郑州 450001）

摘　要：针对异构处理器在传统通用计算中利用率低的问题，提出基于开放计算语言 OpenCL（open computing language）的新的通用计算技术，它提供了统一的编程模型。介绍了 OpenCL 的特点、架构及实现原理等，并提出 OpenCL 性能优化策略。将 OpenCL 与计算统一设备架构 CUDA（compute unified device architecture）及其它通用计算技术进行对比。对比结果表明，OpenCL 能够充分发挥异构处理平台上各种处理器的性能潜力，充分合理地分配任务，为进行大规模并行计算提供了新的强有力的工具。

关键词：异构处理器；通用计算；开放计算语言（OpenCL）；性能优化；计算统一设备架构（CUDA）

中图法分类号：TP391　　**文献标识号**：A　　**文章编号**：1000-7024（2012）11-4191-05

Parallel programming of heterogeneous system based on OpenCL

ZHAN Yun, ZHAO Xin-can, TAN Tong-de

(School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China)

Abstract: Aiming at the problem of the low utilization of heterogeneous processors in traditional general purpose computing, a new general purpose computing technology based on OpenCL (open computing language) is proposed to provide a unified programming model. Firstly, OpenCL features, framework and performance principles are described, and the performance optimization strategies of OpenCL are proposed. At last, OpenCL and CUDA (compute unified device architecture) are compared with other computing technologies. The result shows that OpenCL can fully excavate various processors' potential in heterogeneous processing platforms, and distribute tasks reasonably, and a new powerful tool for large-scale parallel computing is provided.

Key words: heterogeneous processors; general purpose computing; OpenCL; performance optimization; CUDA

0　引　言

使用 GPU 进行通用计算是近些年的热点研究领域。传统的通用计算主要依靠图形 API，这给不熟悉图形应用的用户带来了极大的不便。为了克服该缺陷，出现了 NVIDIA CUDA[1] 和 ATI Stream[2] 等编程模型，给编程带来了极大的灵活性。但是在程序移植性方面，NVIDIA GPU 和 AMD GPU 互不兼容。并且今天的计算机系统通常包含高度并行的 CPU、GPU 和其它类型的处理器，让软件开发人员充分合理的利用这些异构处理平台的优势变得非常重要。

针对上述问题，Khronos Group 推出开放计算语言 OpenCL[3-4]。OpenCL 是一种新的并行计算技术，使用它可以调用计算机内全部计算资源，包括 CPU、GPU 和其它处理器，为软件开发人员能够方便高效的利用异构处理平台、充分挖掘计算机中所有计算设备的性能潜力提供了充分保障。OpenCL 包含一个用来协调异构处理器间并行计算的 API，和一个基于 ISO C99 跨平台的编程语言，且能与 OpenGL、OpenGL ES 和其它图形类 API 高效互通，具有跨平台、兼容性好等特点，极大地方便了软件开发人员的编程工作，将大大地推动并行计算的发展。在诸如分子模拟、流体模拟、碰撞模拟、生物应用以及电信、金融、证券数据分析、医疗等各种领域都将具有良好的应用前景。本文以 OpenCL 的架构、软件框架及实现原理等为基础，对 OpenCL 进行描述，并将 OpenCL 与 CUDA 等通用计算技术进行对比，突出 OpenCL 的优越性。

1　OpenCL 架构

为体现 OpenCL 的核心思想，可以用 4 个模型来描述

Creator PDF Master Editor

RT Embedded http://www.kontronn.com

其架构[5-7]。

1.1 平台模型

该模型描述内部单元之间的关系，如图1所示。主机（host）可以是个人计算机或超级计算机。OpenCL设备（device）可以是CPU、GPU、DSP或其它处理器。每个OpenCL设备包含若干计算单元（compute unit，CU），每个计算单元又由若干处理单元（processing element，PE）组成。

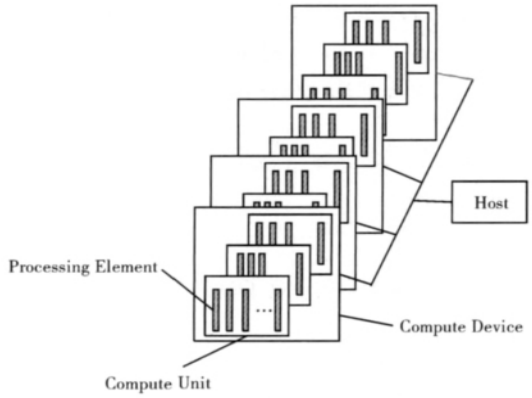

图1 平台模型

OpenCL通过平台实现主机与设备间的交互操作。主机管理着整个平台上的所有计算资源，所有OpenCL应用程序都是从主机端启动并在主机端结束的。应用程序运行时由主机提交命令（command），在设备上的处理单元中执行计算。每个计算单元内所有的处理单元都会执行相同的一套指令流程。每个处理单元以单指令多数据SIMD或单程序多数据SPMD模式运行指令流。

1.2 执行模型

OpenCL执行两类程序：内核程序（kernel）和主机程序（host program）；前者由若干个OpenCL设备执行，后者由主机执行。OpenCL通过主机程序定义上下文（context）并创建一个被称为命令队列（command-queue）的数据结构来管理内核程序的执行。在命令队列中，内核程序可顺序执行（In-order Execution）也可乱序执行（Out-of-order Execution）。

每当主机提交内核程序到设备上执行时，系统便会创建一个N维（N可取1，2，3）的索引空间NDRange。如图2所示。内核程序将索引空间中的每一点用一个工作项（work-item）来表示，将若干个工作项划分成一个工作组（work-group）。在一个计算单元内可运行同一工作组中的工作项，并且该组内的工作项可以并发执行在多个处理单元上。

1.3 内存模型

设备上有4块存储区域可以提供给工作项进行访问：

（1）全局内存：所有工作项对其中的任意数据都可以读写，容量较大，但访问延迟较高。

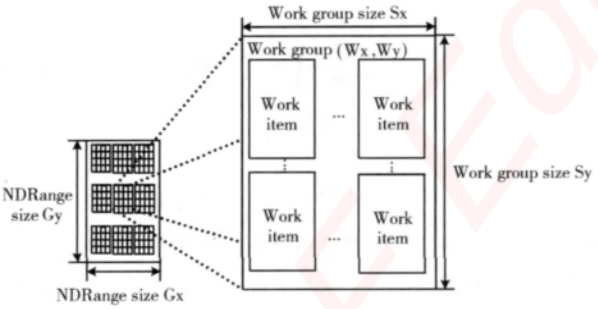

图2 执行模型索引空间

（2）常数内存：全局内存的一部分，但工作项对其中的任意数据只能进行读操作。

（3）局部内存：对特定工作组可见，该工作组中所有工作项可以对其中的任意数据进行读写操作。局部内存通常位于片上，访问速度较快，但容量有限。

（4）私有内存：该区域中的数据只对单独的工作项可见。内存模型如图3所示。

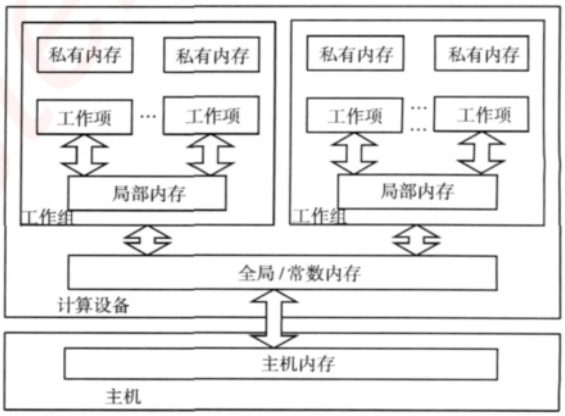

图3 内存模型

一个kernel既不能访问主机内存也不能动态分配全局内存和常数内存，所有的内存都是由主机进行管理。表1描述了内核与主机对内存区域的分配以及访问情况。

表1 存储区域比较

		全局内存	常数内存	局部内存	私有内存
主机	分配	动态	动态	动态	无
	访问	读写	读写	无	无
内核	分配	无	静态	静态	静态
	访问	读写	只读	读写	读写

1.4 编程模型

数据并行和任务并行是OpenCL可以支持的两种并行编程模型，同时两者的混合模型也得到支持。通常情况下，OpenCL采用的首要模型是数据并行，而对多核CPU主要采用任务并行。

RT Embedded http://www.kontronn.com

在数据并行编程模型中，一系列的指令会作用到内存对象（memory object）的多个元素上。严格来说，数据并行要求内存对象单元与工作项保持一对一的映射，而在实际应用中，并不要求严格按照这种方式。在数据并行编程模型中，OpenCL 又提供了一种分级方式，有两种方法：显式分级模型和隐式分级模型；前者要求开发人员指出工作项的总数和工作项所属的工作组；而后者仅需要开发人员定义工作项的总数，对于工作项的划分则根据 OpenCL 的实现来管理。在任务并行编程模型上，每个工作项都相当于在一个单一的计算单元内，该单元内只有单一工作组，该工作组只有该工作项本身在执行。

2 OpenCL 软件框架

OpenCL 软件框架包含三部分：OpenCL 平台层、OpenCL 运行时和 OpenCL 编译器。如图 4 所示。在 OpenCL 平台层上，开发人员可以查询系统中的平台数目并选定运行平台，在指定的平台上选择必要的计算设备并对它们进行初始化，然后可以建立上下文，并创建命令队列。执行内核程序、读、写及复制缓冲区和同步操作等都是通过命令队列中的命令实现的。一个命令队列和一个 OpenCL 设备是一对一的关系。在 OpenCL 运行时中，开发人员建立内核实例，并将其映射到正确的内存空间中，接着在命令队列中排队执行内核。OpenCL 编译器负责编译运行在设备上的程序，并创建可执行程序。

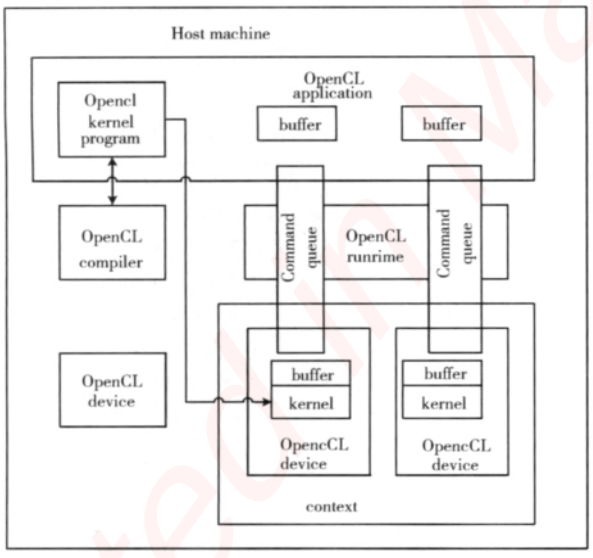

图 4 OpenCL 软件框架

3 OpenCL 实现原理

3.1 实现原理

从编程的角度来说，OpenCL 中的操作都是和一个给定的上下文环境有关，每个上下文环境中都有若干个相关的设备。在上下文环境中，OpenCL 能够保证设备之间的内存一致性。OpenCL 使用 buffers（1 维的内存块）和 images（2 维和 3 维的内存块）来存储内核数据。一旦分配了存放内核数据的内存空间，并且指定了运行内核程序的设备，就需要装载和编译内核程序。为了执行内核程序，程序员必须建立一个内核对象（kernel object），并且设置内核参数。之后，内核就可以在命令队列中排队执行了。

通常情况下，OpenCL 设计的大致流程为：

（1）创建并初始化 OpenCL 设备和上下文环境，建立命令队列；

（2）创建并编译源程序，建立内核句柄；

（3）分配数据所需内存空间，并将数据复制到 OpenCL 设备上；

（4）设置内核参数；

（5）执行内核程序；

（6）将计算结果从 OpenCL 设备复制到主机中；

（7）释放系统所占资源。

3.2 OpenCL 程序框架

以向量加法运算为例，介绍 OpenCL 的程序框架。其中内核代码部分如下：

```
__kernel void vectoradd（__global const float * a，__global const float * b，__global float * c）
{//向量元素索引
int nindex=get__global__id（0）；
//将向量 a 和 b 的和存储到向量 c 中
c［nindex］=a［nindex］+b［nindex］；
}
```

OpenCL 的主机代码关键部分如下：

（1）初始化设备和上下文：OpenCL 中所有的计算都是在上下文中进行的，先获得 OpenCL 的平台 clGetPlatformIDs（……）；

clGetPlatformInfo（……）；

在此平台上选定 OpenCL 设备，CL__DEVICE__TYPE__GPU表明选择的设备类型是 GPU

clGetDeviceIDs（PlatformIDs［NvPlatform］，CL__DEVICE__TYPE__GPU，……）；

接着创建上下文环境，并建立命令队列用以执行内核实例

hContext=clCreateContext（……）；

hCmdQueue = clCreateCommandQueue（hContext，……）；

（2）建立内核：内核代码存放在 sProgramSource 所指的字符串中，首先创建一个程序对象

hProgram=clCreateProgramWithSource（hContext，1，sProgramSource，……）；

再通过 clBuildProgram（）将程序编译成二进制代码，

RT Embedded http://www.kontronn.com

最后建立内核实例，vectorAdd 为内核函数名。

clBuildProgram（hProgram，……）；

hKernel = clCreateKernel（hProgram，“vectorAdd”，0）；

（3）分配设备内存：OpenCL 的设备内存都是由 buffer 对象来管理的，用 clCreateBuffer（）函数来建立，以向量 A 为例

hDeviceMemA=clCreateBuffer（hContext，……）；

（4）设置内核参数：clSetKernelArg（hKernel，0，sizeof（cl_mem），（void *）&hDeviceMemA）；……

（5）执行内核程序：OpenCL 的内核在各种条件都满足的情况下，在命令队列中排队执行

clEnqueueNDRangeKernel（hCmdQueue，hKernel，1，0，&cnDimension，0，0，0，0）；

（6）返回结果：将结果从设备复制到主机中

clEnqueueReadBuffer（hContext，……）；

（7）释放资源：clReleaseKernel（hKernel）；……

4 OpenCL 性能优化策略

在实际的应用阶段，开发人员在编写 OpenCL 程序时都要考虑程序的性能，充分合理地利用各种资源，因此要对 OpenCL 程序进行优化。通常情况下，OpenCL 性能优化可主要从以下三方面进行：①数据传输优化；②内存访问的优化；③计算及控制流优化。

目前，大多数显卡是通过 PCI-E 和主存进行数据交换的，然而 PCI-E 的数据传输带宽远小于 GPU 访问设备内存的带宽，因此合理的数据传输策略对通用计算性能的提高是至关重要的。数据传输优化需要注意两点：

（1）合理使用 Buffer 类型，避免无谓的数据传输；

（2）在传输数据量不变的情况下，提高在 GPU 上计算量的比重。

对内存访问进行优化可以使读取、写入速度成倍增长，大大提高程序的整体执行效率，可以对全局存储器和局部存储器进行访问优化。影响全局存储器访问性能的因素有访存次数和访存模式。如果访问全局存储器过于频繁，这会大大降低执行的效率。因此为了改善性能，可以将只需在工作组中共享的数据存放在局部存储器中，需要的时候再读出来即可。在访问全局存储器时设计好的访存模式也对性能的提高至关重要，这就需要编程人员设计好的算法。局部存储器是每个工作组私有的一片存储区域，访问速度比全局存储器快，因此，可以合理利用这片区域以减少对全局存储器的读写，并且减少工作组内工作项之间的数据交换，从而大大提高程序的执行效率。

OpenCL 的内核程序是 OpenCL 设备上的多个工作组并行执行的，不同大小的工作组执行内核程序的效率是不同的，因此可以通过设置 clEnqueueNDRangeKernel（）函数的参数来指定工作组的大小和总的工作项数量，这样就可以通过控制流来改善程序的性能。

5 常见通用计算技术及其对比

由于通用计算的广阔发展前景，以及 GPU 等各种处理器在硬件制造工艺上的不断改进，目前许多厂商都推出了自己的通用计算技术，除本文介绍的 OpenCL 外，还有 NVIDIA 公司的 CUDA 以及其它公司的通用计算技术。

5.1 CUDA 简介

CUDA 是 NVIDIA 公司于 2007 年正式发布的一种通用计算技术。在 CUDA 中，CPU 作为主机，GPU 作为协处理器（Co-processor）或设备。一个系统中可以有一个主机和若干个设备。GPU 上有大量的流多处理器（SM），每个 SM 中又有 8 个流处理器（SP）。每个 SP 对应一个线程，每个 SM 对应一个或多个线程块。CUDA 中最小的执行单位是线程（thread），用 threadIdx 标识。每个线程都有自己的局部存储器和私有寄存器。所有的线程共享一份全局存储器、常数存储器和纹理存储器。若干个线程的集合组成一个线程块（block），每个线程块有其唯一的标识（blockIdx）。同一线程块中的所有线程共享一块共享存储器，可实现快速的同步操作。不同线程块是并行执行的，无法直接通信。执行相同程序的若干个 block 组成一个网格（grid）。同一个 kernel 程序可以并行运行在一个网格所包含的所有线程块中的线程上[8-10]。

5.2 OpenCL 与 CUDA 对比

OpenCL 与 CUDA 在术语表达及特点上有不同，分别如表 2、表 3 所示。

表 2 OpenCL 与 CUDA 架构术语比较

CUDA	OpenCL
设备（GPU）	设备（CPU、GPU 或其它处理器）
流多处理器（SM）	计算单元（CU）
流处理器（SP）	处理单元（PE）
全局存储器	全局存储器
常数存储器	常数存储器
纹理存储器	暂无对应
共享存储器	局部存储器
局部存储器	私有存储器
网格	索引空间
线程块	工作组
线程	工作项

从表 3 可以看出，两者采用了不同的开发语言：

（1）CUDA 采用的是 CUDA C 作为开发语言，是一种类 C 的编程语言，它包含对 C 语言的最小扩展集和一个运行时库，编写的文件由 NVCC 编译器编译。CUDA C 对 C 语言的扩展集引入了变量类型限定符、函数类型限定符等，

RT Embedded http://www.kontronn.com

表 3　OpenCL 与 CUDA 特点对比

	CUDA	OpenCL
支持者	NVIDIA	Apple、AMD、NVIDIA 等
硬件要求	NVIDIA GPU	CPU、GPU 及其它处理器
系统平台	Windows、Linux、Mac OS	Windows、Linux、Mac OS
软件架构	CUDA 开发库 运行时 API 驱动 API	OpenCL 平台层 OpenCL 运行时 OpenCL 编译器
开发语言	CUDA C	OpenCL C
编译方式	离线编译	在线＋离线编译
数学函数库	私有	已确定、开放
设备与上下文	一个上下文一个设备	一个上下文多个设备
扩展机制	私有	已确定、开放
执行模式	SIMT	SIMD、SPMD
特点	支持 OpenCL、简单高效、较好的便利性和扩展性、封闭	高效轻便、跨平台、通用性好、兼容性好、开放标准

但都有一定的限制，如 __global__ 函数类型限定符用于声明内核函数，只能在设备上执行，从主机调用。

（2）OpenCL 采用的是基于 ISO C99 的 OpenCL C 语言，也是一种类 C 的编程语言。但 OpenCL C 引入了一些函数限定符、变量限定符，并且支持 C 语言中原有的一些数据类型，还增加了一些新的数据类型如 half 类型、内建的矢量数据类型等，OpenCL C 还提供了丰富的内建函数，其中有些内建函数名和 C 语言的库函数相同，只是实现有所不同。OpenCL C 为开发者提供的是统一的编程语言，适合在各种处理器上实现通用计算，并且程序移植性好。

5.3　其它通用计算技术

除 CUDA 和 OpenCL 之外，用于通用计算的还有 AMD 公司的 Stream 技术以及微软的 DirectCompute[11] 技术等。其中，ATI Stream 通用计算模型类似于 CUDA 模型，和 CUDA 一样它也只能运行在自家的显卡上，且只支持 Windows vista、Windows XP、Windows 7 和部分 Linux 操作系统。ATI Stream 采用 Brook＋作为上层的应用程序接口，有很多的限制，并有过多的封装，效率不高，不适用于开发复杂的项目。DirectCompute 是用于 GPU 通用计算的应用编程接口，集成在 Microsoft DirectX 内，平台兼容性差，只能用于 Windows 操作系统，不利于在其它操作系统上进行推广。

6　结束语

本文针对异构处理器在传统通用计算中利用率低的问题，提出了 OpenCL，为异构处理器提供了统一的编程模型。文章首先对 OpenCL 的特点、架构模型等进行了介绍，提出了 OpenCL 性能优化策略，并将 OpenCL 和 CUDA 及其它通用计算技术进行对比。从对比中可以看出，OpenCL 在异构处理平台上具有很大的优势，不再局限于某种特定的硬件平台，可兼容不同厂家的各种设备，充分挖掘各种处理器的性能潜力，充分合理地分配任务，为并行计算提供了新的强有力的工具。鉴于 OpenCL 的优势，将其运用到增强现实三维注册中，实现更加实时的效果是下一步欲尝试的研究。

参考文献：

[1] NVIDIA . CUDA [EB/OL] . [2007-10-08]. http://www.nvidia.com/cuda.

[2] AMD. Stream[EB/OL]. [2009-03-12]. http://www.amd.com/stream.

[3] Khronos Group. The OpenCL specification [EB/OL]. [2010-09-20]. http://www.khronos.org/opencl/.

[4] Jens Breitbart, Claudia Fohry. OpenCL-an effective programming model for data parallel computations at the cell broadband engine [C]. Los Alamitos: IEEE Computer Society Press, 2010.

[5] John E Stone, David Gohara, Guochun Shi. OpenCL: A parallel programming standard for heterogeneous computing systems [C]. Los Alamitos: IEEE Computer Society Press, 2010: 66-73.

[6] CHEN Gang, WU Bai-feng. GPU performance optimization targeting OpenCL model [J]. Journal of Computer-Aided Design & Computer Graphics, 2011, 23 (4): 571-581 (in Chinese). [陈钢，吴百锋. 面向 OpenCL 模型的 GPU 性能优化 [J]. 计算机辅助设计与图形学学报，2011，23 (4)：571-581.]

[7] Martin Jurecko, Jana Kocisova. Evaluation framework for GPU performance based on OpenCL standard [C]. Los Alamitos: IEEE Computer Society Press, 2010: 256-261.

[8] ZHAO Li-li, ZHANG Sheng-bing, ZHANG Meng. High performance FFT computation based on CUDA [J]. Application Research of Computers, 2011, 28 (4): 1556-1559 (in Chinese). [赵丽丽，张盛兵，张萌. 基于 CUDA 的高速 FFT 计算 [J]. 计算机应用研究，2011，28 (4)：1556-1559.]

[9] YI Song, LIU FU-yan, LI Xue-min, et al. Ocean surface simulation based on CUDA platform [J]. Computer Engineering and Design, 2011, 32 (3): 998-1001 (in Chinese). [易松，刘福岩，李雪敏，等. 基于 CUDA 平台的海洋表面模拟 [J]. 计算机工程与设计，2011，32 (3)：998-1001.]

[10] DONG Luo, GE Wan-cheng, CHEN Kang-li. Study on application of parallel computation on CUDA [J]. Information Technoligy, 2010 (4): 11-15 (in Chinese). [董荦，葛万成，陈康力. CUDA 并行计算的应用研究 [J]. 信息技术，2010 (4)：11-15.]

[11] Microsoft. DirectCompute [EB/OL]. http://msdn.microsoft.com/zh-cn/directx, 2010.

RT Embedded http://www.kontronn.com

参考文献：

[1] HU Shunyun, HUANG Tinghao, CHANG Shaochen, et al. FLoD: A framework for peer-to-peer 3D streaming [C]. Phoenix, AZ, USA: IEEE INFOCOM, 2008: 2047-2055.

[2] LV Zhihan, YIN Tengfei, HAN Yong, et al. WebVR-web virtual reality engine based on P2P network [J]. Journal of Networks, 2011, 6 (7): 990-998.

[3] WANG Wei, JIA Jinyuan, ZHANG Chenxi, et al. Survey on progressive transmission strategy of large-scale virtual scenes [J]. Computer Science, 2010, 37 (2): 38-43 (in Chinese). [王伟，贾金原，张晨曦，等. 大规模虚拟场景渐进式传输的研究进展 [J]. 计算机科学，2010，37 (2)：38-43.]

[4] WANG Qinghui, LI Jingrong. Interactive visuali-zation of complex dynamic virtual environments for industrial assemblies [J]. Computers in Industry, 2006, 57 (4): 366-377.

[5] Dirk Staneker, Dirk Bartz, Wolfgang Strafler. Occlusion-driven scene sorting for efficient culling [C]. Proceedings of the 4th international conference on Computer graphics, virtual reality, visualization and interaction, 2006: 99-106.

[6] WANG C, Garcia A, SHEN H W. Interactive level-of-detail selection using image-based quality metric for large volume visualization [C]. North Carolina USA: IEEE Visualization and Computer Graphics, 2007: 122-134.

[7] WANG Yan, LI Zhaokui, SHI Xiangbin. MMOG AOI multicast advanced algorithm based on P2P [J]. Computer Engineering, 2010, 37 (2): 247-249 (in Chinese). [王岩，李照奎，石祥滨. 基于 P2P 的 MMOG 兴趣域内多播改进算法 [J]. 计算机工程，2010，37 (2)：247-249.]

[8] ZHANG Changming, ZHANG Hong. LOD automatic generation algorithm based on edge collapse [J]. Computer Engineering and Design, 2005, 26 (11): 3109-3111 (in Chinese). [张昌明，张虹. 一种基于边折叠的 LOD 自动生成算法 [J]. 计算机工程与设计，2005，26 (11)：3109-3111.]

[9] MEN Xiaopeng, LV Xiaofeng, MA Dengwu, et al. Research on primitive geometric elements intersection test in virtual environment [J]. Journal of Naval Aeronautical Engineering Institute, 2006, 21 (3): 379-382 (in Chinese). [门晓鹏，吕晓峰，马登武，等. 虚拟场景中基本几何元素相交测试技术 [J]. 海军航空工程学院学报，2006，21 (3)：379-382.]

[10] ZHANG Zhongxiang, WANG Shitong. Fast intersection test for triangle to triangle [J]. Computer Engineering and Design, 2010, 31 (4): 869-871 (in Chinese). [张忠祥，王士同. 三角形对的快速相交测试 [J]. 计算机工程与设计，2010，31 (4)：869-871.]

(上接第 4195 页)

[12] LI Sen, LI Xin-liang, WANG Long, et al. Performance analysis of OpenCL parallel acceleration on cavity flow problem [J]. Application Research of Computers, 2011, 28 (4): 1401-1404 (in Chinese). [李森，李新亮，王龙，等. 基于 OpenCL 的并行方腔流加速性能分析 [J]. 计算机应用研究，2011，28 (4)：1401-1404.]

[13] Ryo Aoki, Shuichi Oikawa, Ryo Tshchiyama, et al. Hybrid OpenCL: Connecting different OpenCL implementations over network [C]. Los Alamitos: IEEE Computer Society Press, 2010: 2729-2735.

[14] Amnon Barak, Tal Ben-Num, Ely Levy, et al. A package for OpenCL based heterogeneous computing on clusters with many GPU devices [C]. Los Alamitos: IEEE Computer Society Press, 2010.

[15] XU Yan-qin, CHEN Qing-kui. Research and implementation of MPI+CUDA model based on SMP clusters [J]. Computer Engineering and Design, 2010, 31 (15): 3408-3412 (in Chinese). [许彦芹，陈庆奎. 基于 SMP 集群的 MPI+CUDA 模型的研究与实现 [J]. 计算机工程与设计，2010，31 (15)：3408-3412.]

Created in Master PDF Editor

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

Creator Master PDF Editor

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

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

Created with Master PDF Editor

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

Creator PDF Editor

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

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

Creat by Master PDF Editor

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

Create PDF Master PDF Editor

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

Creator Master PDF Editor

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

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

Creator Master PDF Editor

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 Nios II 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

Create by Master PDF Editor

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

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.

Created in Master PDF Editor