

# 基于 CUDA 平台的 FIR 滤波算法的设计与优化

郭海凤<sup>1,2</sup>, 李莉<sup>1</sup>

(1. 金陵科技学院 信息技术学院, 江苏 南京 211169;  
2. 江苏省信息分析工程实验室, 江苏 南京 211169)

**摘要:**针对目前基于普通 DSP 的 FIR 算法速度低、扩展性差的缺点,提出并实现基于 CUDA 平台实现的 FIR 滤波算法。由于在 CUDA 中程序可以直接操作数据而无需借助于图形系统的 API,使开发者能够在 GPU 强大计算能力的基础上建立起一种效率更高的密集数据计算解决方案。该算法将 CUDA 用于 FIR 滤波器输入输出关系计算,采用矩阵乘法的并行运算技术,在 GPU 上建立并行滤波模型,并对算法进行了优化。实验结果表明,在 Tesla C1060 平台上,和传统的基于 DSP 的 FIR 滤波算法计算速度相比,基于 CUDA 平台计算 FIR 滤波算法时,其加速比可接近 30,解决了传统基于 DSP 计算 FIR 滤波算法速度较慢、扩展性差的问题。

**关键词:**FIR 滤波算法;并行计算;GPU 计算;CUDA 平台;矩阵乘法

中图分类号:TP391

文献标识码:A

文章编号:1673-629X(2014)03-0102-04

doi:10.3969/j.issn.1673-629X.2014.03.026

## Design and Optimization of FIR Filtering Algorithm Based on CUDA Platform

GUO Hai-feng<sup>1,2</sup>, LI Li<sup>1</sup>

(1. College of Information Technology, Jinling Institute of Technology, Nanjing 211169, China;  
2. Information Analysis Engineering Laboratory of Jiangsu Province, Nanjing 211169, China)

**Abstract:** It is well known that FIR algorithm based on normal DSP has low computing speed and extensive capabilities. In order to overcome these, present a new FIR filter algorithm based on CUDA platform. Since in CUDA program can directly manipulate data without graphics API of the system, enables developers on the basis of the powerful GPU computing power to set up a efficient dense data computing solutions. The algorithm adopts CUDA for FIR filter calculation of input and output relationship, using the parallel computing technology of matrix multiplication, on the GPU the parallel filtering model is established, and the algorithm is optimized. Experiment on Tesla C1060 shows that, compared with traditional FIR filter algorithm's speed based on DSP, it can accelerate its computation speed up to 30 times, solving conventional FIR filter's defect based on DSP of low speed and bad extending capabilities.

**Key words:** FIR filtering algorithm; parallel computing; GPU computing; CUDA platform; matrix multiplication

## 0 引言

CUDA 是 NVIDIA 公司为 GPU 开发的一个基于 C 语言的并行计算架构,在 CUDA 中程序可以直接操作数据而无需借助于图形系统的 API。使开发者能够在 GPU 强大计算能力的基础上建立起一种效率更高的密集数据计算解决方案, GPU 是一种高性能的多核处理器,可以用来加速许多应用。目前,它在石油天然气、地震处理、金融风险管控、产品设计、媒体图像以及科学研究等领域具有重要应用<sup>[1]</sup>。

CUDA 支持的 GPU 包含 GeForce、Quadro 和 Tesla 三个系列(如表 1): GeForce 是 NVIDIA 公司面向娱乐消费市场的 GPU 产品,如 GeForce GTX580、GeForce 8800GS 等; Quadro 是面向专业图形的 GPU 产品,如 Quadro FX4800、Quadro FX5800 等;而 Tesla 系列是专门面向高性能计算的产品,如 Tesla C1060、Tesla C2050、C2075、Tesla S2090 等<sup>[2]</sup>。

GeForce 采用了图形加速度技术,因此降低了双精度浮点运算的性能; Quadro 是面向专业图形的 GPU 产品;如 Tesla 不支持 OpenGL 加速技术,而增强了双

收稿日期:2013-06-02

修回日期:2013-09-06

网络出版时间:2014-01-07

基金项目:江苏省高校自然科学基金项目(11KJD520006)

作者简介:郭海凤(1980-),女,江苏南京人,在读博士,讲师,研究方向为信息处理、信息检索、图像检索。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20140107.1511.001.html>

精度浮点运算功能,它支持异步传输、并行 datacache 技术等。总体来说,GeForce 专为图形处理而设计,Tesla 专为并行计算而设计<sup>[3]</sup>。

表1 部分支持 CUDA 并行计算的 GPU		
产品型号	核心数量	计算能力
GeForce 8800GTX	128	1.0
GeForce 9800GX2	2×128	1.1
GeForce GT240	96	1.2
GeForce GTX 260	192	1.3
GeForce GTX 470	448	2.0
Quadro FX 4600	96	1.0
Quadro FX 3700	112	1.1
Quadro FX1800M	72	1.2
Quadro FX 4800	192	1.3
Quadro 5000M	320	2.0
Tesla C1060	240	1.3
Tesla C2050/C2075	448	2.0
Tesla S2050/S2070	448×4	2.0

### 1 CUDA 执行模型

CUDA 是 NVIDIA 推出的一种高性能 GPU 体系架构,支持 CUDA 的 GPU 与传统 GPU 的最大区别在于它拥有片内共享存储空间用以减小片外访存延迟。CUDA 执行模型以 C 语言为基础,在显示芯片上执行的程序,程序员在编程时不需要去学习特定的显示芯片指令或结构,只需要掌握基本的 C 语言知识,按照 CUDA 提供的 API 函数去编写程序即可完成并行化的计算<sup>[4]</sup>。CUDA 硬件架构如图 1 所示。

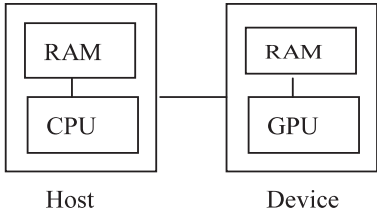


图1 硬件架构

在 CUDA 架构下,一个程序分为两个部分:Host 端和 Device 端。Host 端是指在 CPU 上执行的部分,而 Device 端则是在显示芯片上执行的部分。Device 端的程序又称为“kernel”。通常 Host 端程序会将所有输入数据准备好后,复制到显卡内存中,再由显示芯片执行 Device 端程序,计算完成后,需要 Host 端程序将计算结果从显卡内存中取回。所以从本质上来说,CUDA 执行模型是两个程序架构的综合体,CPU 则承担着数据初始化、内存申请、数据传递、结果显示及处理等一系列操作,而 GPU 则是利用大量的执行单元进

行并行计算。  
显示芯片执行时的最小单位是 thread。数个 thread 可以组成一个 block。每个 block 中的 thread 能够存取同一块共享的内存,而且可以快速进行同步的动作。

### 2 FIR 滤波器

在现代电子系统中,FIR 滤波器以其良好的线性特性被广泛使用,它可以在保证任意幅频特性的同时具有严格的线性相频特性,同时其单位抽样响应是有限长的,因而滤波器是稳定的系统。  
有限长脉冲响应(FIR)滤波器的系统函数只有零点,除原点外,没有极点,因而 FIR 滤波器是稳定的<sup>[5-6]</sup>。如果其他单位脉冲响应是非因果的,总能够方便地通过适当的移位得到因果的单位脉冲响应,所以 FIR 滤波不存在稳定性和是否可实现的问题。它的另一个突出优点是在满足一定的对称条件时,可以实现严格的线性相位。由于线性相位滤波器不会改变输入信号的形状,而只是时域上使信号延时,因此线性相位特性在工程实际中具有非常重要的意义。如在数据通信、图像处理等应用领域,往往要求信号在传输和处理过程中不能有明显的相位失真,因而线性相位 FIR 滤波器在通信、图像处理、模式识别等领域都有着广泛的应用<sup>[6]</sup>。

长度为  $M$  的因果有限冲激响应滤波器由传输函数  $H(z)$  描述,如式(1)所示:

$$H(z) = \sum_{k=0}^M h(k)z^{-k}$$

(1)

式(1)是次数为  $M$  的  $z^{-1}$  的一个多项式,在时域中,式(1)中有限冲激响应滤波器的输入输出关系为:

$$y(n) = \sum_{i=0}^M h(k)x(n-k)$$

(2)

其中,  $y(n)$  是输出序列;  $x(n)$  是输入序列;  $M$  是滤波器长度。该式中没有考虑 FIR 数字滤波器的瞬间响应过程。

### 3 基于 CUDA 的 FIR 滤波算法

在计算 FIR 滤波器中有限冲激响应滤波器的输入输出关系时,普通的 DSP 在计算时,随着输入序列长度以及滤波器长度的增加,其累加之和也是不小的数字,速度会变的越来越慢,差别也越来越明显,在很多应用领域中,计算时间已不能满足系统的需求。若要加快输出响应速度,则需要采用并行处理的方式。

CUDA 是 NVIDIA 的 GPGPU 模型,为加速图像的实时处理而设计的一种运行在 GPU 上的开发平台,现代芯片已经具有高度的可程序化能力,并且具有相

当高的内存带宽以及大量的浮点计算单元,可以大量处理并行化的问题,尤其是在大型浮点型数据计算方面优势比较显著<sup>[7-9]</sup>。CUDA 是一种由 NVIDIA 推出的通用并行计算架构,若将其用于 FIR 滤波器输入输出关系计算,势必可以起到事半功倍的效果。

### 3.1 线程数目选择

在 CUDA 的平台下,一个并行程序总的来说可以分为两个部分:Host 和 Device。程序执行时,会将 Host 端的数据复制到显卡的内存中,由显卡芯片来执行计算 Device 端的程序,待计算完成后,Host 端程序将计算结果从显卡芯片中取回。因此在计算很少的执行单元时,其 Host 端与 Device 端的互存的复杂度和仅仅在 CPU 上计算相当,甚至还要低,因此对于少量的计算来说,GPU 计算并不能够显示出绝对的优势。在 CUDA 程序中,总共启动的线程数目可按照以下公式计算<sup>[10-12]</sup>:

$N$  个线程块  $\times$  1 个线程/线程块 =  $N$  个并行线程

事实上也可启动  $N/2$  个线程块,每个线程块包含 2 个线程,或者  $N/4$  个线程块,每个线程块启动 4 个线程,以此类推。

对于线程数目的确定,要本着“高度并行化”的原则,即线程相互之间尽可能相互独立,尽可能地启动较多的线程,同时又不能造成资源的浪费。

假设计算程序能够分解的线程数目为  $N$ ,在 GPU 中每个线程块启动的线程数目为  $T$ ,则 block(线程块)的数目为  $(N + T - 1)/T$ ,总共启动的线程数目为  $[N + T - 1/T] \times T$ ,其中  $[N + T - 1/T]$  表示对  $N + T - 1/T$  取整。这样就确保当  $N$  不是  $T$  的倍数时能够启动足够多的线程,又不会造成资源的过度浪费。

### 3.2 FIR 滤波算法并行化

CUDA 的并行执行模型是由线程、线程块及网格组成,分别支持一维、二维、三维的线程组织,线程是 CUDA 语言的基本单位,具有独立的寄存空间;线程块是一组互相合作的线程,可以通过共享内存存储空间交换数据<sup>[7-8]</sup>。

基于式(2)中响应信号  $y(n)$  之间的计算相互独立,在计算时可将  $y(n)$  分成单个的线程。文中选用一维线程组织,令响应信号的长度为  $N$ ,算法步骤如下:

Step1:为  $h$ 、 $x$ 、 $y$  申请 GPU 存储空间,数据类型为浮点型,长度分别为  $M$ 、 $M + N$ 、 $N$ 。

Step2:拷贝数据,将数据复制到  $h$ 、 $x$  中(主内存复制到显卡)。

Step3:分配线程数目,包括 block 数目及 threads 数目,由于式(2)中  $y(n) = \sum_{i=0}^M h(k)x(n-k)$  的计算包括

$h(k)x(n-k)$  的累加过程, $n$  的最大取值范围是  $N$ ,所以总共的计算次数为  $M \times N$ ,即最大线程个数为  $M \times N$ 。另 threads 数目为  $T$ ,block 数目为  $B$ 。针对不同 GPU 特性(文中使用 GPU 型号为 Tesla C2070), $M \times N > 1024$  时, $T$  的取值可设置为 1024, $M \times N > 512$  时, $T$  的取值可设置为 512, $B$  的取值为  $(M \times N + T - 1)/T$ 。

Step4:分配线程,总的线程数目为  $M \times N$ ,针对每个  $y(i)$ ,分配  $M + 1$  个线程进行计算,其中  $M$  个线程负责计算,待  $M$  个线程完成计算后,启动另外 1 个线程负责累加。逐次计算  $y(i)$ 。

Step5:拷贝数据  $y$ (显卡复制到主内存)。

### 3.3 程序优化

由于 CPU 存取显卡内存数据时只能通过 PCI Express 接口进行传输,在 CPU 和设备之间传输数据要比在设备和全局内存之间传输数据的带宽低很多,因此尽可能减少 CPU 和设备之间数据传输的次数<sup>[13-14]</sup>;针对 FIR 滤波算法来说,其输入数据为  $b$ 、 $x$ ,输出数据只有  $y$ 。针对 GPU 计算涉及的数据,CPU 和设备之间数据传输只需要 2 次即可(拷入、拷出)。

一个 block 内的 thread 可以有共享的内存,也可以进行同步,利用这一点,将数据  $x$  存入共享变量(shared memory),在 GPU 进行计算时,能够加快读取  $x$  的速度。因此在 3.2 节 Step3 步骤中,将每个累加线程设置共享变量,减少读取内存的次数,同时达到同步的目的。

显卡上的内存是 DRAM(动态随机存取存储器),因此最有效率的存取方式,是以连续的方式存取。在大量的 thread(线程)执行程序时,要考虑到 thread 的执行方式。如果在同一个 thread 中连续存取内存,在实际执行时反而不是连续了,势必延缓了计算时间。最佳的执行方式是当一个 thread 在等待内存的数据时,GPU 会切换到下一个 thread。也就是说,实际上执行的顺序按照线程的顺序,即让 thread 0 计算  $y(i)$ ,thread 1 计算  $y(i+1)$ ...依此类推。

通过上述分析,优化后的 FIR 滤波算法并行化计算过程如以下步骤所示:

Step1:为  $h$ 、 $x$ 、 $y$  申请 GPU 存储空间,数据类型为浮点型,长度分别为  $M$ 、 $M + N$ 、 $N$ 。

Step2:拷贝数据,将数据复制到  $h$ 、 $x$  中(主内存复制到显卡)。

Step3:分配线程数目, $y(n) = \sum_{i=0}^M h(k)x(n-k)$  计算中,总的线程数目为  $M \times N$ ,由于  $n$  的最大取值范围是  $N$ ,可达到百万数量级,而  $M$  的取值范围在 200 以下,并且每个  $y(n)$  的计算是对  $h(k)x(n-k)$  的累加,

所以将  $h(k)$  数据设置为共享变量,可以减少系统内存对  $h(k)$  的访问次数。这样总的线程数目为  $N$ ,令每个 block 块中 threads 的数目为  $T$ ,则 block 的数目为  $(N+T-1)/T$ ,  $T$  的数值根据  $N$  的大小确定,不同型号的 GPU,每个 block 块中拥有 threads 数目范围不同,根据具体型号选择  $T$  的最佳值。

Step4:总的线程次数为  $N$ ,启动线程数目  $T_{\text{num}} = T \times [(N+T-1)/T]$ ,线程号分别用  $T_i(i=0,1,\cdots,T_{\text{num}}-1)$  表示,分配线程规则如下:

$$\begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_{\text{num}} \end{bmatrix} = >$$

$$\begin{bmatrix} y(0), y(T_{\text{num}}), y(2T_{\text{num}}), \cdots, y((N/T_{\text{num}}-1)T_{\text{num}}) \\ y(1), y(T_{\text{num}}+1), y(2T_{\text{num}}+1), \cdots, y((N/T_{\text{num}}-1)+1) \\ y(2), y(T_{\text{num}}+2), y(2T_{\text{num}}+2), \cdots, y((N/T_{\text{num}}-1)T_{\text{num}}+2) \\ \cdots \\ y(T_{\text{num}}), y(N/T_{\text{num}}+T_{\text{num}}), \cdots, y((N/T_{\text{num}}-1)T_{\text{num}}+T_{\text{num}}) \end{bmatrix}$$

其中,  $T_0$  负责计算:  $y(0), y(T_{\text{num}}), y(2T_{\text{num}}), \cdots, y((N/T_{\text{num}}-1)T_{\text{num}})$ ,其他线程分配计算任务依次类推。当  $T_i$  计算的任务数目超过  $N$  时,返回。

Step5:拷贝数据  $y$  (显卡复制到主内存)。

4 实验结果

实验平台是 Intel(R) core(TM) i7, 4 GB RAM, Tesla C2075,操作系统是 32 位的 Windows XP, CUDA 工具包的版本是 4.0。

为了便于统计和比较,在 GPU 上计算时,将系统初始化时间、数据拷贝时间以及计算时间分别记录;同样的数据在 DSP 上计算并统计时间,测试记录部分结果如表 2 所示。

表 2 测试记录(时间单位为  $\mu\text{s}$ )

H 长度	Y 长度	DSP 时间	GPU 计算时间				
			总时间	初始化	GPU 传输	计算	结果返回
200	1 024	760	51 198	50 688	78	16.4	415
200	2 048	1 561	50 199	49 787	41	16.2	355
200	4 096	3 807	51 505	51 018	44	16.2	425
200	8 192	6 015	51 286	50 851	52	16.4	365
200	16 384	12 188	53 821	53 129	64	16.4	611
200	32 768	24 256	51 688	50 847	86	16.5	738
200	65 536	48 438	53 183	51 855	130	16.8	1 181
200	131 072	96 822	54 225	51 610	412	16.2	2 185
200	262 144	193 947	51 982	50 643	565	20.8	752
200	524 288	393 618	54 033	51 883	786	20.9	1 342
200	1 048 576	780 725	54 022	50 578	1 509	24.5	1 910
200	2 097 152	1 560 596	59 580	49 396	2 057	25.9	3 494

通过表 2 可以看出,当计算量成倍增加时,CPU 计算的时间也随着计算量的增加而增加;而对于 GPU 来说,计算时间基本上成平稳趋势,和 CPU 相比,计算速度可提高 6 000 多倍,但其总的运行时间最多相差 32 倍,当  $y$  为 65 536 时,CPU 和 GPU 总的计算运行时间相当,随着  $y$  的增加,GPU 的并行计算优势才逐渐体现出来。

通过比较 GPU 计算各个属性的运行时间可以发现,在 GPU 计算中,系统初始化时间占据了主要部分,CPU 与 GPU 之间的数据拷贝也随着数据量的增加而增加,占据着 15% 左右的比重。

5 结束语

文中探讨了 CUDA 平台,利用 GPU 强大并行处理能力进行计算。实验结果表明,针对 FIR 滤波算法来说,GPU 计算加速比可达 30 倍。

FIR 滤波器在多个科技领域有重要的应用,对于其计算速度的研究意义深远,通过实验比较,文中提出的基于 CUDA 平台计算 FIR 滤波算法时其速率可提高近 30 倍,具有一定的实用价值。

但目前基于 CUDA 平台的 GPU 并行计算方面尚有许多限制,在进行小数据量的计算时并不能够显示出 GPU 计算的优势,若想更深层次的应用,仍需要进一步的学习和探讨。

参考文献:

[1] Smith S M. Feature based image sequence understanding[D]. Oxford:Oxford University,1992.

[2] Northcutt S. Inside network perimeter security[M]. [s. l. ]: Pearson Education,2004.

[3] Wong M L, Wong T T. Parallel hybrid genetic algorithms on consumer level graphics hardware[C]//Proc of CEC. [s. l. ]:[s. n. ],2006:2973-2980.

[4] 陈国良. 并行算法的设计与分析[M]. 北京:高等教育出版社,2002.

[5] 李莹,路卫军,于敦山,等. 一种在 FPGA 上实现 FIR 数字滤波器的资源优化算法[J]. 北京大学学报(自然科学版),2009,45(2):222-226.

[6] Cheng C, Parhi K K. Hardware efficient fast parallel FIR filter structures based on iterated short convolution[J]. IEEE transactions on circuits and systems I: Regular papers, 2004, 51(8):1492-1500.

[7] 吴恩华,柳有权. 基于图形处理器(GPU)的通用计算[J]. 计算机辅助设计与图形学学报,2004,16(5):601-612.

[8] Voronenko Y, Püschel M. Multiplierless multiple constant multiplication[J]. ACM transactions on algorithms, 2007, 3(2): 11-20.

通过对电压偏差的真实值、时序预测值和神经网络预测值三者的曲线图的比较,可以很清晰地看出神经网络算法的预测结果的曲线图与真实值的曲线图更接近。

再通过表 1 对五项常规指标的时间序列预测算法的预测结果误差和神经网络算法的预测结果误差进行比较,从而更准确地反映这两种算法应用于五项常规指标的预测的差别。

表 1 两种不同算法的预测结果平均误差 %		
指标	时序算法预测结果 平均误差	神经网络算法预测 结果平均误差
电压偏差	17.89	3.38
频率偏差	55.89	19.42
短时波动与闪变	54.35	19.21
电压负序不平衡	52.67	19.83
总谐波畸变率	26.46	8.91

从表 1 可以看出神经网络算法应用于电能质量的五项常规指标的预测误差明显低于时序算法的预测误差,且平均误差能够满足工程上的要求。

### 5 结束语

文中通过分析时间序列算法和人工神经网络算法的优缺点和适用范围,并结合分析有功功率与电能质量稳态指标中的五项常规指标的相关性,最终提出了 ARIMA 算法应用于有功功率的预测和神经网络应用于电能质量五项常规指标的预测这一方案。根据五项常规指标的时间序列预测算法的预测结果和神经网络算法的预测结果以及这两种算法的预测结果误差进行比较,确定人工神经网络算法的预测准确度要明显高于时序算法的预测准确度。

通过实例研究可知,该方法可以有效预测出电能

质量指标序列的变化趋势,平均误差均在 20% 以内,具有工程可行性和实用性。

### 参考文献:

[1] Han Hongke, Qi Linhai. Application and research of multidimensional data analysis in power quality[C]//Proc of ICC-DA. Qinhuangdao:[s. n.],2010.

[2] Basu M, Basu B. Analysis of power quality (PQ) signals by continuous wavelet transform[C]//Proc of power electronics specialists conference. Orlando, FL:IEEE,2007:2614-2618.

[3] 庄后顺,孟庆海,袁 振. 浅谈电能质量问题[J]. 科技信息,2010(35):282-282.

[4] 陈红坤,黄 娟. 数据挖掘及其在电能质量分析中的应用[J]. 电力系统及其自动化学报,2009,21(5):51-55.

[5] 杜 奕. 时间序列挖掘相关算法研究及应用[D]. 合肥:中国科学技术大学,2007.

[6] 张保稳. 时间序列数据挖掘研究[D]. 西安:西北工业大学,2002.

[7] 黄 超,朱扬勇. 基于 ARMA 模型的联机时间序列数据分割算法[J]. 模式识别与人工智能,2005,18(2):129-134.

[8] Srinivasan D, Ng W S, Liew A C. Neural-network-based signature recognition for harmonic source identification[J]. IEEE trans on power delivery,2006,21(1):398-405.

[9] Yan Shaojin, Peng Yonging, Guo Guang. Neuroid BP type model applied to the study of monthly rainfall forecasting[J]. AAS,1995,12(3):336-342.

[10] 侯 慧,游大海,尹项根,等. 人工智能技术在电能质量控制中的应用[J]. 武汉大学学报(工学版),2004,37(3):114-118.

[11] 徐 群,陶 顺,肖湘宁,等. 新能源发电功率与电压偏差的相关性研究[J]. 电测与仪表,2011,48(6):1-5.

[12] 张 利. 基于时间序列 ARIMA 模型的分析预测算法研究及系统实现[D]. 镇江:江苏大学,2008.

(上接第 105 页)

[9] Bakhoda A, Yuan G L, Fung W W L, et al. Analyzing CUDA workloads using a detailed GPU simulator[C]//Proceedings of IEEE international symposium on performance analysis of systems and software. New York:IEEE Press,2009:163-174.

[10] 姚 平. CUDA 平台上的 CPU/GPU 异步计算模式[D]. 合肥:中国科学技术大学,2010.

[11] 刘 琳,何剑锋,王红玲. GPU 加速数据挖掘算法的研究

[J]. 郑州大学学报(理学版),2010,42(2):31-34.

[12] 白洪涛. 基于 GPU 的高性能并行算法研究[D]. 长春:吉林大学,2010.

[13] 朱二周. 基于 CPU/GPU 平台的虚拟化技术研究[D]. 上海:上海交通大学,2012.

[14] 肖 汉. 基于 CPU+GPU 的影像匹配高效能异构并行计算研究[D]. 武汉:武汉大学,2011.

基于CUDA平台的FIR滤波算法的设计与优化

作者:	郭海凤, 李莉, GUO Hai-feng, LI Li
作者单位:	郭海凤, GUO Hai-feng(金陵科技学院 信息技术学院, 江苏 南京 211169; 江苏省信息分析工程实验室, 江苏 南京 211169), 李莉, LI Li(金陵科技学院 信息技术学院, 江苏 南京, 211169)
刊名:	计算机技术与发展
	ISTIC
英文刊名:	Computer Technology and Development
年, 卷(期):	2014(3)

本文链接: [http://d.wanfangdata.com.cn/Periodical\\_wjfz201403026.aspx](http://d.wanfangdata.com.cn/Periodical_wjfz201403026.aspx)