

# 基于 FPGA 的嵌入式多核处理器及 SUSAN 算法并行化

王 洁 张淑燕 刘 涛 季振洲 胡铭曾

(哈尔滨工业大学计算机科学与技术学院 哈尔滨 150001)

**摘 要** 给出了四核心嵌入式并行处理器 FPEP 的结构设计并建立了 FPGA 验证平台. 为了对多核处理器平台性能进行评测, 提出了基于 OpenMP 的 3 种可行的图像处理领域的经典算法 SUSAN 算法的并行化方法: 直接并行化 SUSAN、图像分块处理和多图像并行处理, 并对这 3 种并行算法在 Intel 四核心平台和 FPEP 的 FPGA 验证平台上进行性能测试. 实验表明, 3 种并行算法在两种四核心平台下均可获得接近 3.0 的加速比, 多图像并行处理在 FPEP 的 FPGA 验证平台可以获得接近 4.0 的加速比.

**关键词** SUSAN; FPGA; OpenMP; 多核处理器; 图像处理

**中图法分类号** TP302 **DOI 号**: 10.3724/SP.J.1016.2008.01995

## Multi-Core Embedded Processor Based on FPGA and Parallelization of SUSAN Algorithm

WANG Jie ZHANG Shu-Yan LIU Tao JI Zhen-Zhou HU Ming-Zeng

(Department of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001)

**Abstract** Structure of 4-core embedded parallel processor is presented and FPGA validation platform that is called FPEP is built. To evaluate performance of multi-core processor platform, three feasible parallel algorithms with OpenMP on SUSAN that is a classic image processing algorithm: direct parallel SUSAN, image block processing and parallel processing of multiple images are proposed, and performance testing of these three parallel algorithms execute on Intel 4-core platform and FPGA validation platform of FPEP. Experimental results demonstrate that speedup ratio of three parallel algorithms on both 4-Core platforms is approximate 3.0, and speedup ratio of parallel processing of multiple images on FPGA validation platform of FPEP is approximate 4.0.

**Keywords** SUSAN; FPGA; OpenMP; multi-core processor; image processing

## 1 引 言

近年来多核处理器的研究已经成为主要的研究热点之一, 而基于嵌入式多核处理器平台的应用日

益受到重视.

以 Intel、IBM 等为代表的商用处理器制造商已经推出了面向包括桌面应用等众多领域的多核处理器产品, 大大提高了计算机系统的性能<sup>[1-2]</sup>. 特别是面向航空航天等应用领域, 嵌入式多核处理器在实

收稿日期: 2008-05-31; 最终修改稿收到日期: 2008-09-08. 本课题得到山东省科技发展计划项目基金(2007GG10001020)资助. 王 洁, 男, 1979 年生, 博士研究生, 研究方向为并行计算、FPGA 验证技术. E-mail: wj@pac518.hit.edu.cn. 张淑燕, 女, 1986 年生, 硕士研究生, 研究方向为嵌入式并行计算和并行算法. 刘 涛, 男, 1979 年生, 博士研究生, 研究方向为嵌入式多核处理器. 季振洲, 男, 1965 年生, 教授, 博士生导师, 研究领域包括计算机体系结构、网络安全、并行计算等. 胡铭曾, 男, 1935 年生, 教授, 博士生导师, 研究领域包括计算机体系结构、网络安全、并行计算等.

时计算、空间图像处理、视频解码等方面有着广泛的应用前景<sup>[3-5]</sup>。目前多核心嵌入式处理主要的研究方向包括总线架构技术、IP 可复用、软硬件协同、SoC 验证、可测性设计、低功耗设计、超深亚微米电路实现、嵌入式软件的开发和移植等。

随着半导体制造技术不断的前进和相应的设计规模以及复杂度飞速的增长,传统的软件仿真工具已不可能完全解决功能验证的问题,一些需要处理大量实时数据的应用也越来越多,因此要求能够在接近实时的条件下进行功能验证。随着微电子技术和工艺的进步,高端的 FPGA(现场可编程门阵列)产品已经成为多核处理器技术研究的主要支撑平台<sup>[6-7]</sup>。Xilinx 公司的 Vertex-4 和 Vertex-5 系列 FPGA 芯片可以达到千万门级,支持内部几百兆的时钟频率,拥有巨大的内部存储器资源和寄存器资源,基于 SPARC 结构的嵌入式多核处理器可以直接在 FPGA 芯片平台上进行设计和验证,并且具有良好的可重构性。

目前,已经成熟的应用软件和基准程序测试集都是基于单核心系统的,并不能充分发挥和评测并行硬件体系结构的优势和性能,因此并行软件的开发和测试对并行计算的发展起着至关重要的作用。

不管是通用处理器还是嵌入式处理器,目前国际上都有公认的性能测试标准,即 SPECCPU2000<sup>[8]</sup>和 EEMBC<sup>[9]</sup>。标准测试程序是在全世界范围内向实际用户征求来的,经过了严格选择。比如 SPECCPU2000 共 14 个大程序,代码量达 600MB,包括科学计算、图像压缩、编译等各种应用;EEMBC 包括家电、网络、汽车、办公等几类。其中 SPECCPU2000 覆盖的应用面最广,包括文件压缩、FPGA 布局布线、编译器、组合优化、国际象棋、文字处理、计算机视觉、编程语言、解释器、数据库、布局布线模拟器、量子动力学、浅水模型、三维势场求解、偏微分方程、三维图形库、计算流体动力学、图像识别/神经网络、地震波传播模拟、计算化学、数论/素数测试等等。但对于特定应用领域的 ASIC 芯片, SPEC 并不适应。目前,我国主管部门和行业协会还没有对国产 CPU/SoC 进行一次性能标准测试,也没有制定本国的标准测试程序。因此,对于处理器性能评测,除了常用的一些速度参数、功耗、加速比外,针对并行处理器芯片的应用领域,需要做常用并行应用算法验证,并给出部分关键性能的对比。

SUSAN 算法是来源于 MiBench 基准测试集<sup>[10]</sup>

的一个图像识别算法,是一个经典的图像边缘检测、角点检测和图像平滑的算法,主要测试基本的算术运算。MiBench 是由 Michigan 大学的 Guthaus 等人开发的一个开源的嵌入式基准测试集,由 35 个嵌入式应用程序组成。SUSAN 算法的应用非常广泛,如用于图像的自动配准、遥感图像的边缘检测、图像的快速角点检测和指纹细化等。

对 SUSAN 算法的并行化有着重要的意义,一方面为嵌入式并行系统提供性能更好的测试,另一方面对图像的处理通常是非常耗时的,图像处理的数据量大,SUSAN 并行化算法能够高效地处理图像以提高性能。因此本文以 SUSAN 算法为例采用 OpenMP 并行编程语言进行并行化,并在 Intel 四核心平台下和嵌入式四核心处理器(Four-core Parallel Embedded Processor, FPEP)的 FPGA 嵌入式平台下进行测试和分析比对,也作为嵌入式多核处理评测技术研究的一个重要组成部分。

## 2 基于 FPGA 的嵌入式多核处理器系统

嵌入式处理器越来越广泛地应用于空间图像处理、空间计算等需要高性能、高可靠性、低功耗处理器<sup>[11]</sup>的领域。SoC 技术的飞速发展芯片级并行处理计算平台的研究提供了坚实的技术基础。FPGA 验证成为 SoC 设计流程中重要的一个环节,一方面作为硬件验证工具,可以将所设计的 RTL 级代码综合实现后写入 FPGA 芯片进行调试检错;另一方面可以进行软件部分的并行开发,在验证板上检测驱动程序,启动操作系统。FPGA 验证的流程相当于一个 FPGA 设计的主要流程,它主要分为设计输入、综合、功能仿真(前仿真)、实现、时序仿真(后仿真)、配置下载、下载后板级调试检错这几个步骤。FPGA 验证是整个 SoC 设计中一个重要而且有效的验证步骤,用来改进 RTL 级设计代码,验证功能的正确和完整性,提高 SoC 流片成功率。

### 2.1 四核心嵌入式并行处理器 FPEP 的结构

嵌入式多核处理器并行计算体系结构主要的研究内容包括:处理器结构选择和并行模式选择、IP 核选择复用、片内总线选择以及总线仲裁机制选择、可配置的 Cache 和 Cache 一致性保证机制、Cache 交换算法选择、外部中断分配机制、时钟中断的分配方式、处理器间并行机制的实现、片上设备集成方式和系统调试方式等<sup>[12-14]</sup>。

在 SPARC V8 结构的 Gaisler LEON3 软核<sup>[15]</sup>基础上提出四核心嵌入式并行处理器 FPEP 的设计结构,并在 FPGA 平台上进行验证.

采用 SMP 架构,共享存储器和 I/O,采用

AMBA 总线规范. 每一个 CPU 都作为 AHB 总线上的一个设备存在,它们平等地拥有访问 SDRAM 和其它外设的权利,通过 AHB 总线的仲裁机制,可以避免访问冲突的发生. 其主要架构如图 1 所示.

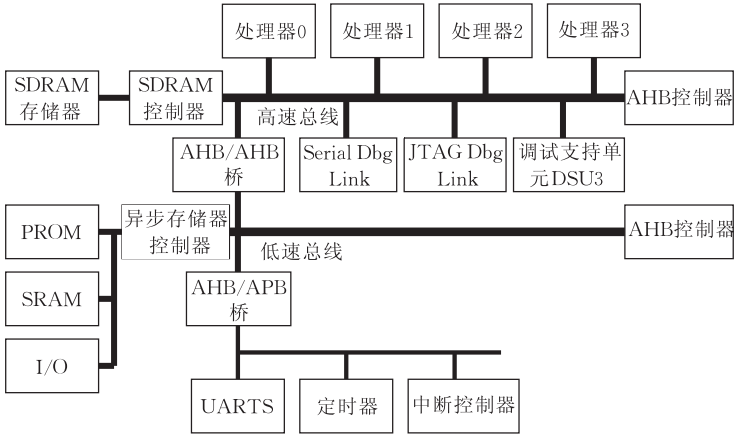


图 1 四核心嵌入式处理器 FPEP 主要架构

所有处理器通过一条高速总线 AHB 在同一芯片中紧密耦合. 处理器共享同样的全局内存、磁盘和 I/O 设备. 只有一份操作系统的副本跨所有处理器运行,并且操作系统必须设计为能利用这种体系结构(多线程操作系统).

每个核的结构如图 2 所示,具备以下性能和

特色:

- (1) 兼容 SPARC V8 指令集;
- (2) 32 位整型单元,五级流水线;
- (3) 64 位(双精度)浮点处理器;
- (4) 266MIPS/66MFLOPS@266MHz;
- (5) 支持嵌入式实时多任务操作系统.

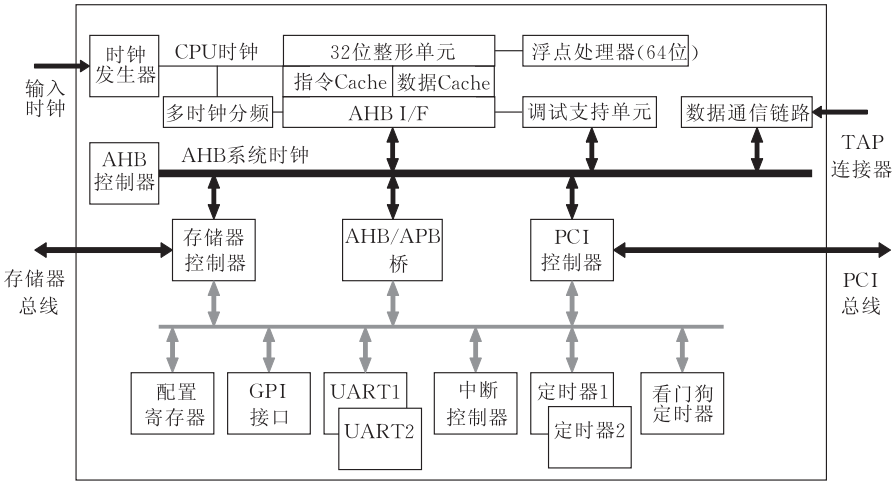


图 2 FPEP 核结构

2.1.1 Cache 一致性算法

指令 Cache 在行重填充过程中使用流使填充延迟最小化. 数据 Cache 使用透写策略并实现了一个双字写缓冲区. 数据 Cache 还可以在 AHB 总线上执行总线窥视,保持 Cache 和存储器数据的一致性. 一个便笺 RAM 可以加到指令 Cache 和数据 Cache 的控制器上,以允许没有数据回写的 0 等待存储器存取时间. 为了保证 CPU 访问时有较高的命中率, Cache 中的内容应该按一定的算法替换. 本文采用

的 Cache 交换算法为一种较常用的算法,是“最近最少使用算法”(LRU 算法),它是将最近一段时间内最少被访问过的行淘汰出局. 映射机制选用直接映射(DIR-MAP).

2.1.2 中断控制方式

中断控制器是作为一个 APB 从设备连接到 AHB 总线的. 它监视和组合中断信号,并对中断信号进行排序,屏蔽和传输给一个或者多个处理器. 每个处理器都有独立的中断屏蔽寄存器,当一个中断

出现在 AMBA 总线上时,中断控制器计算优先级,为每个处理器进行中断屏蔽处理,并发送中断信号给相应的处理器. 当一个处理器响应中断时,对应的中断保持位被清除.

2.1.3 高速与低速设备连接

SDRAM 是系统运行时的主要数据存储,因此,它直接通过 SDRAM 控制器挂在 AHB 总线上. 这样,SDRAM 就可以处在一个和 CPU 最近的位置,有利于提高数据吞吐量.

其它高速设备,则通过 AHB/APB 桥连接在 AHB 总线上,包括同步存储器控制器,可以挂接 PROM、SRAM、FLASH 和一些高速的 IO 设备,以及串行调试端口、JTAG 调试端口、以太网 MAC 等.

低速设备是连接在 APB 总线上的. APB 总线通过 AHB、APB 桥连接到 AHB 总线上. 系统现有的低速设备有 UART、定时器、中断控制器等.

2.1.4 系统引导

当处理器复位的时候,从 0x0 地址开始执行. 引导代码就存放在这个地址,处理器在综合时在每个处理器的 PSR 寄存器中分配一个 ID 号. 如果处理器 ID 号大于 0,则进入 loop 模式. ID 号为 0 的处理器将启动并执行一些任务. 其它从处理器等待分布在 AHB 总线上的 BP 寄存器. 主处理器在初始化内存、串口等设备后,等待加载程序到内存,之后将堆栈寄存器值设置完成,并置 BOOT 寄存器. 从处理器在检测到这个寄存器的值之后,开始启动,按各自

的 ID 号为自己设置堆栈空间,大小各为 20KB. 然后跳转到 BP 寄存器制定的地址处开始运行.

2.1.5 处理器同步与总线仲裁

处理器之间同步锁放在 AHB 总线上,用 5 个寄存器来实现. 可以支持一个周期的读写功能. 这些锁用来支持互斥和 RTOS 用来设置互斥信号量.

总线仲裁采用轮训协议,初始从 ID 为 3 的处理器开始,依次往下,然后再往复循环.

2.2 FPGA 验证平台

用于 ASIC 原型验证的 FPGA 硬件平台如图 3 所示,以大规模 FPGA 器件为中心,周围集成了大量的外部设备,主要包括以下主要组件:

- FPGA 器件: XC2V6000-FG676-4;
- 配置 prom: 6×18V04-VQ44;
- SRAM 存储器: 8Mbit (256k×32bit);
- SDRAM 存储器: 128Mbit (32M×32bit);
- FLASH 存储器: 64Mbit (2M×32bit);
- DSU I/F: Debug Support Unit 接口;
- Memory expansion: 存储器扩展接口;
- User I/O: FPGA 用户自定义 I/O 接口;
- Ethernet I/F: 以太网网卡物理层 10/100 transceiver 接口;
- PIO expansion: 通用 I/O 接口;
- Power Supply: 5V 输入,板载高性能 3.3V 和 1.5V 转换模块;
- PCI I/F: PCI 接口.

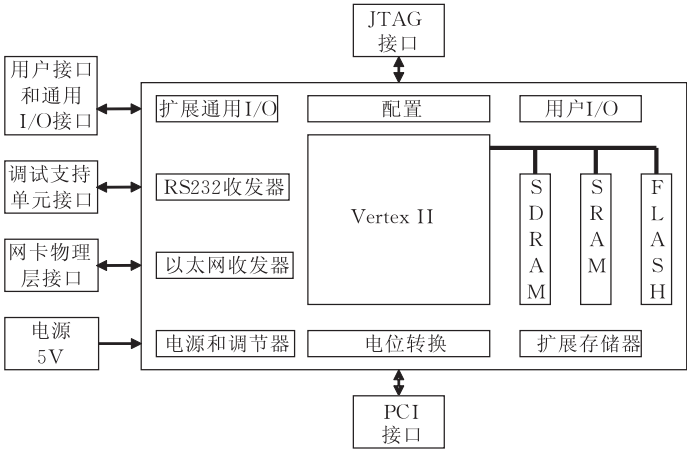


图 3 FPGA 验证平台结构

2.3 嵌入式并行操作系统

并行处理器的应用不仅取决于其硬件结构,与之相配合的软件系统也起着决定性的作用. RTEMS 是一个高性能的实时嵌入式操作系统环境,支持多处理器系统. 它主要有以下特点:

- (1) 具备多任务处理能力;
- (2) 适用于同类或异类多处理器系统;
- (3) 事件驱动、基于优先级的抢占式调度;
- (4) 可选的单调速率调度机制;
- (5) 任务间通信和同步;

- (6) 优先级继承;
- (7) 中断响应管理;
- (8) 动态内存分配;
- (9) 高级用户配置.

在 RTEMS 实时操作系统或实时执行任务的基础上,构建应用系统. 对于 RTEMS 操作系统的移植,包括 GNU 工具对于目标系统的支持;移植 RTEMS 执行代码;开发并行多处理器板级支持包;实现一组 RTEMS 所依赖于处理器的程序组.

### 3 SUSAN 算法及参数设定

SUSAN 算法是由英国牛津大学的 Smith、Brady 首先提出的<sup>[16]</sup>,它主要包括图像的边缘检测、角点检测和图像的平滑 3 个子算法. 并行处理技术可以有效提高图像识别算法的效率,本文将以 SUSAN 算法为例研究基于 FPGA 的多核处理器平台 FPEP 在图像识别应用中的并行化及效率分析.

#### 3.1 SUSAN 特征检测原理

SUSAN 算法采用圆形模板,如图 1 所示,用此模板在图像上移动,若模板内像素的灰度与模板中心像素(称为核 Nucleus)灰度的差值小于一定阈值,则认为该点与核具有相同(或相近)的灰度,由满足这样条件的像素组成的区域称为 USAN(Univalue Segment Assimilating Nucleus).

图 4 中的圆形模板  $a$  完全处在图像中,模板  $b$  完全处于背景中,此时 USAN 区域面积最大;当模板移向图像边缘时,USAN 区域逐渐变小(如图 4 中  $c$ );如图 4 中的  $d$ ,当模板中心处于边缘时,USAN 区域很小;当模板中心处于角点时,USAN 区域最小(如图 4 中的  $e$ ). 可以看出,在边缘处像素的 USAN 值都小于或等于其最大值的一半. 因此,计算图像中每一个像素的 USAN 值,通过设定一个 USAN 阈值,查找小于阈值的像素点,即可确定为边缘点、角点或其它.

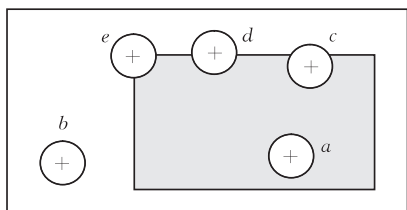


图 4 SUSAN 特征检测原理

#### 3.2 SUSAN 算法

SUSAN 模板在图像上滑动,在每个位置上,比

较模板内各图像像素的亮度与模板核的亮度,如式(1)<sup>[16]</sup>所示.

$$c(\mathbf{r}, \mathbf{r}_0) = \begin{cases} 1, & |l(\mathbf{r}) - l(\mathbf{r}_0)| \leq t \\ 0, & |l(\mathbf{r}) - l(\mathbf{r}_0)| > t \end{cases} \quad (1)$$

其中,  $\mathbf{r}_0$  是核在二位图像中的位置;  $\mathbf{r}$  是模板内除核之外的任意一个点的位置;  $l(\mathbf{r})$  为  $\mathbf{r}$  的亮度;  $l(\mathbf{r}_0)$  为核点的亮度;  $t$  为亮度差的阈值,它控制生成角点的数量;  $c(\mathbf{r}, \mathbf{r}_0)$  为亮度比较的结果.

通常对于式(1)采用更稳定可靠的形式,如式(2)<sup>[16]</sup>所示:

$$c(\mathbf{r}, \mathbf{r}_0) = e^{-\left(\frac{|l(\mathbf{r}) - l(\mathbf{r}_0)|}{t}\right)^6} \quad (2)$$

采用式(3)<sup>[16]</sup>计算模板内所有点(共  $n$  个)与核亮度比较的和

$$n(\mathbf{r}_0) = \sum_{\mathbf{r}} c(\mathbf{r}, \mathbf{r}_0) \quad (3)$$

然后  $n(\mathbf{r}_0)$  与一个给定阈值  $g$  (成为几何阈值)比较,得到图像的边缘响应,如式(4)<sup>[16]</sup>所示:

$$R(\mathbf{r}_0) = \begin{cases} g - n(\mathbf{r}_0), & |n(\mathbf{r}_0)| < g \\ 0, & |n(\mathbf{r}_0)| \geq g \end{cases} \quad (4)$$

当  $n(\mathbf{r}_0) < g$  时,所检测到像素位置  $\mathbf{r}_0$  可以认为是一个边缘点、角点等.

#### 3.3 模板的选取

由于图像的数字化,无法实现真正的圆形模板,用近似圆代替. 在程序实现中,采用了两种模板,一个是 37 像素的圆形模板,一个是  $3 \times 3$  的小模板,如图 5 所示. 模板较小时,如果门限选取不当,可能会发生边缘点漏检的情况,模板选取过大会增大计算量. 这也跟具体的图像有关,本文的设计中选用了 37 像素的模板.

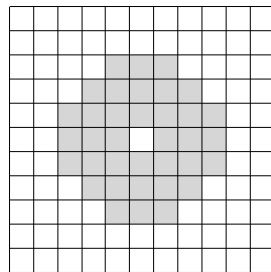


图 5 SUSAN37 像素模板

#### 3.4 门限值 $t, g$ 的确定

门限  $g$  决定了边缘点的 USAN 区域的最大值,即只要图像中的像素的 USAN 值小于  $g$ ,该点就被判定为边缘点.  $g$  过大时,边缘点附近的像素可能作为边缘被提出,过小则会漏检边缘点.

门限  $t$  表示所能检测边缘点的最小对比度,是



能忽略的噪声的最大容限.  $t$  越小, 可从对比度越低的图像中提取特征. 对于不同的对比度和噪声情况的图像, 应取不同的值. 在本文的设计中, 默认值  $t$  为 20, 针对不同的图像修改  $t$  值为 10.

当区域边界模糊时, SUSAN 算法会产生虚假角点, 通常使用式(5)

$$\bar{r}(r_0) = \frac{\sum_r r c(r, r_0)}{\sum_r c(r, r_0)} \tag{5}$$

消除虚假角点. 使用非最大抑制方法, 即通过将一个边缘点作为  $3 \times 3$  模板的中心, 在邻域范围内进行亮度比较找出角点.

4 SUSAN 算法并行化设计与实现

OpenMP 是可移植多线程应用程序开发的行业标准, 在细粒度与粗粒度线程技术上具有很高的效率. 对于将串行应用程序转换成并行应用程序, OpenMP 指令是一种容易使用且作用强大的工具, 具有使应用程序因为在对称多处理器或多核系统上并行执行而获得大幅性能提升的潜力<sup>[17]</sup>.

本文对 SUSAN 算法的并行化采用 OpenMP 共享存储并行标记语言. OpenMP 通过分析串行程序中存在潜在并行化可能的地方, 通过增加并行标记语句的形式将程序转化为并行程序, 而原串行程序结构不发生变化或仅在需要对数据存储形式进行调整的时候发生局部变化. OpenMP 采用 fork-join 执行模型, 以单线程开始执行, 遇到并行区创建子线程进行并行处理, 并行区结束退出, 继续以单线程处理. OpenMP 的优势是循环语句的并行化. SUSAN 并行算法中的结构图如图 6 所示.

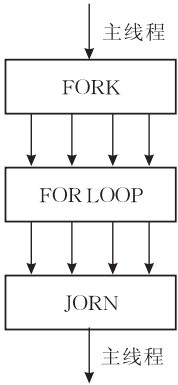


图 6 SUSAN 并行化算法结构

通常来讲, 并行算法是在给定并行模型下的一种具体明确的计算方法和步骤. 其分类有不同的分类方

法. 根据并行计算任务的大小分类, 可以分为 3 类:

(1) 粗粒度并行算法; 它所含的计算任务有较大的计算量和较复杂的计算程序.

(2) 细粒度并行算法; 它所含的计算任务有较小的计算量和较短的计算程序.

(3) 中粒度并行算法; 它所含的计算任务的大小和计算作为并行处理计算, 程序的长短在粗粒度和细粒度两种类型的算法之间.

综上所述, 根据不同粒度的划分, 我们给出下列 SUSAN 算法并行化的 3 个改进算法.

4.1 SUSAN 算子改进算法

SUSAN 算法包括 3 个子算法, 边缘检测、角点检测和平滑, 这 3 个子算法都是基于 SUSAN 算子进行的具体应用, 本文中采用 OpenMP 并行编程语言对 SUSAN 算子进行并行化标记, 找到算法内部存在的可并行区, 实现并行, SUSAN 算子是 SUSAN 算法的核心. 另外由 SUSAN 算子扩展的边缘检测算法也可实现部分并行化, 角点检测和平滑算法由于存在数据相关, 还需进一步研究, 以确定可否利用 OpenMP 对其标记实现并行.

由前述 SUSAN 算法特征检测原理及对程序的分析可以得到, SUSAN 算法是对图像中的每个点逐点利用前述公式计算图像的边缘响应, 以确定其是边缘点、角点、内部点和外部点. 这些计算只是访问原图像每个像素点的值, 并没有修改, 即对原图像只读不写, 不存在读写相关问题, 并行处理结束后串行保存结果到文件中. 对 37 点模板每个像素点被计算 36 次, 在算法实现时是用嵌套 for 循环来控制图像中每行每列像素点的位置, 并行化循环是 OpenMP 的强项, 将算法中的 for 循环标记为并行化处理, 并保证处理后的结果是正确的. 此算法记为 4.1 算法. 对原程序做并行标记例子如下:

```
susan_principle:
# pragma omp parallel for private(i,j)
for (i=3; i<y_size-3; i++)
for (j=3; j<x_size-3; j++)
```

等.

4.2 SUSAN 内部分块并行算法

从图像角度出发, 将一个大图像分成几块相同的部分, 把每一部分分配给一个线程处理, 最后再将每个线程单独处理的小图像合并到一个大图像中. 这也是一种内部并行的方法, 是图像并行化中常用的方法. 此算法记为 4.2 算法. 该算法的设计关键如下.

### (1) 图像的分块数与均匀分割

具体图像的分块数目与系统线程数有关,且根据 SUSAN 算法的特点,计算的复杂度取决于图像的分辨率,与图像内像素点的灰度值无关,因此将图像分成大小相同的块就可以保证每个线程负载的平衡.图像的均匀分割是图像处理中一个常见的问题,对有些图像格式很难找到等分点,SUSAN 算法中图像为 pgm 格式,在文件头中指明了图像的分辨率,分割时只要将此分辨率等分分割就可以等分图像.

### (2) 图像的分块处理

每块图像的边缘处需要一定的像素点重合,以保证图像特殊点的正确提取,防止漏掉在小图像的边缘线处的点.如果分块的小图像没有重合的话,通常会丢失两个图像相接处的像素点的信息.在本算法设计中是采用边缘处各重叠 5 个像素点距离(与采用的模板大小有关)的区域,在图像分割和写入时都要重复处理这段区域.因此,当图像分辨率较小时,这段重叠区域将占整个图像的很大部分,对图像的处理大部分集中在这段重复处理上,是不适合使用这种方法进行并行化处理的.

常用的等分图像的方法如图 7 所示,图 7(a)为将图像水平均分为四部分,阴影部分为各块边缘需要重复处理的部分,设  $x\_size$  为图像的宽度, $y\_size$  为图像的高度, $d$  为重复处理的像素点距离,则图 7(a)中重复处理的像素数为  $3 \times x\_size \times d$ ;图 7(b)为将图像按水平竖直均分为四部分,每一块需重复处理与已处理块的边缘重叠部分,图 7(b)中重复处理的像素数为  $(x\_size + y\_size) \times d$ ;图 7(c)为将图像按竖直方向均分为四部分,由图可知,图 7(c)中重复处理的像素数为  $3 \times y\_size \times d$ .因此,可根据图像的具体像素值,来选择最优的分块方式以使得重复处理的像素数最小.本文采用图 7(a)的方式划分图像.

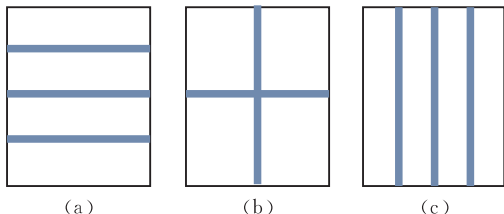


图 7 SUSAN 内部分块图像分割方式

### (3) 图像的预处理时间

图像的预处理时间增加,降低了并行化的效果.在对图像进行处理之前,获得图像的分辨率和图像所存放的首地址,算法中需要先将图像分割成小块,

利用 SUSAN 算法中读写图像的函数,将这些小块图像完整的保存,再将图像读出以获得这些值,这期间进行了大量的读写存储器操作,在不同平台上,读写存储器时间会影响并行效果.

图像分块并行化方法跟具体的算法和图像格式有关,如果图像分割开销较小,则适合采用这种方法,如果图像分割需要增加相当大的开销,这种方法的实际可用性将会降低.

### 4.3 SUSAN 外部多个图像并行处理

前述两种方法在数据量很小的情况下,由于并行粒度小,不一定适用.一次输入多个图像,将每个或每几个图像分给一个线程处理,以这种思想实现多个线程的并行.该算法思想和实现比较简单,通常情况下是最稳定最高效的算法.这是一种外部并行的方法.此算法记为 4.3 算法.

## 5 SUSAN 并行算法效率分析

分别使用 Intel 四核心处理器平台和四核心嵌入式处理器 FPEP 的 FPGA 验证平台对 SUSAN 并行算法进行测试,分析其效率提升,并进行比较分析.

对于多核嵌入式处理器平台,分别通过模拟器仿真和 FPGA 验证平台测试,对该验证平台的试剂性能进行评测.

### 5.1 Intel 四核心处理器平台测试

测试环境为

(1) 硬件环境: Intel® Core(TM) 2 Quad Q6600 @ 2.40GHz, 2GB 内存.

(2) 软件环境 Microsoft VC++ 2005, 图像查看器, 图像转换器.

对上一节中所述 3 个算法的测试结果如下文中各表所示,表中算法的运行时间单位为毫秒(ms),用 openmp 的库函数 `omp_get_wtime()` 获得的时间可精确到微秒( $\mu s$ )级.每个算法采用 3 种图像大小进行测试,测试时间是算法的计算时间,输入输出时间没有进行比较,因为这部分是串行执行,读写存储器需要的时间很长且不稳定.在算法测试过程中,每个时间值波动很小,说明系统环境包括软件和硬件环境对算法的影响较小,测试所得的时间是可信的.每个时间值都是 10 次测量结果的平均值,以求得稳定值.实验中以加速比<sup>[7]</sup>作为衡量算法效率提升的指标,算法 A 相对算法 B 加速比的计算公式为  $S = t_B / t_A$ ,其中,  $t_B$  为算法 B 的执行时间,  $t_A$  为算法 A 的

执行时间.

5.1.1 各算法的效率提升分析

表 1 为 4.1 算法在 Intel 四核系统上单线程和 4 个线程的运行效率提升表. 所选择的图像大小依次增大,当图像较小时,加速比较小,随着图像的增大,加速比有所增大,当图像大小超过一定阈值时,加速比达到最大值并趋于稳定,该算法所达到的加速比为 2.98. 事实上,当图像很小时,并行化的开销所占总计算时间很大比率,即并行化的粒度需足够大才有并行的意义,否则反而会增加算法的执行时间.

表 1 SUSAN 算法(4.1)效率提升表

图像(MB)	算法		加速比
	Susanp1	Susanp4	
0.109	7.214	3.072	2.35
2.277	133.889	45.739	2.93
9.291	587.528	196.995	2.98

注: Susanp1 指 4.1 算法单线程执行; Susanp4 指 4.1 算法 4 个线程执行.

表 2 为 4.2 算法在 Intel 四核系统上将图像分为 4 块的运行效率提升表. 处理图像大小依次增大,当图像较小时,加速比较小,随着图像的增大,加速比增大,并逐渐趋于稳定,当图像大小超过一定域值时,加速比达到最大值并保持稳定不变. 该算法的加速比为 2.88. 这组数据也显示了并行开销和并行化粒度之间的关系. 用 OpenMP 实现的共享存储的显示并行化处理程序适合大粒度的算法.

表 2 SUSAN 算法(4.2)效率提升表

图像(MB)	算法		加速比
	Susans	Susanm4	
0.109	6.800	3.397	2.00
2.277	129.944	45.546	2.79
9.291	202.206	581.613	2.88

注: Susans 指原串行 Susan 算法; Susanm4 为 4.2 算法将图像等分为 4 块.

表 3 为 4.3 算法在 Intel 四核系统上处理 4 个文件与串行处理的运行效率提升表. 处理的图像大小依次增大,当图像较小时反而显示出较大的加速比,随着图像的增大趋于稳定值,此算法中并行化的开销比较小,只有开启 4 个线程的开销,各线程间相互独立,无需等待,所以加速比与图像大小的关系不明显.

表 3 SUSAN 算法(4.3)效率提升表

图像(MB)	算法		加速比
	Susans	Susanm4	
0.109	6.800	8.974	3.03
2.277	129.944	181.570	2.86
9.291	202.206	787.288	2.95

注: Susans 指原串行 Susan 算法; Susanm4 指 4.3 算法同时处理 4 个图像.

5.1.2 各算法并行效率的比较

如图 8 所示(其中,Susanp4 是 4.1 算法开启 4 个线程,Susann4 是 4.2 算法划分 4 个部分,Susanm4 是 4.3 算法同时处理 4 个图像),分别采用 3 种数据量大小进行测试.

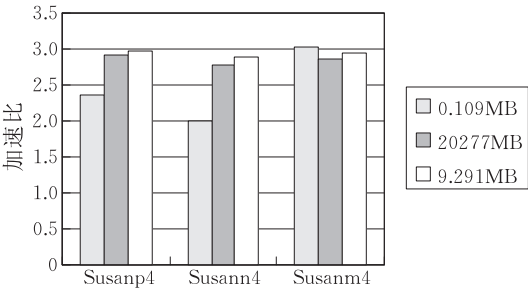


图 8 SUSAN 各并行化算法加速比比较

当图像较小时,Susanm4 的效率最好,这说明 Susanm4 的并行开销是最小的,对处理数据量的要求较小. 而 Susanp4 和 Susann4 在图像较小时,程序并行的粒度较小,并行开销大,加速比较小. 随着图像的增大,3 种算法的效率提升相差不大,Susanp4 和 Susanm4 略有优势,另外,Susann4 增加了图像的预处理步骤,需要较长的读写存储器的时间,因此总体的效率提升并不明显(并没有显示在图 8 中),所以 Susann4 算法的实际可用性还跟具体的环境有关.

5.1.3 OpenMP 开销分析

从表 4 可以看出 Susans 是原来的串行算法,程序中没有并行语句开销,串行程序单进程执行, Susanp1 是并程序,但是程序运行时只开启了一个线程,这样程序在执行时仍执行线程的并行语句,它比 Susans 程序开销更大. 由表 4 可看出,并行开销并不是很大,对总的并行效率影响不大,说明 Susanp 算法的设计比较合理. 随着图像的增大,并行开销有所增加,但是并行开销所占程序的总开销的比率逐渐减小. 随着并行粒度的增加,并行开销可忽略.

表 4 SUSAN 串行算法与单线程算法效率比较

图像(MB)	算法		开销	开销比率/%
	Susanp1	Susans		
0.109	7.214	6.800	0.414	5.74
2.277	133.889	129.944	3.945	2.94
9.291	587.528	581.613	5.915	1.00

注: Susanp1 指 4.1 算法开启 1 个线程; Susans 指原串行 Susan 算法.

5.2 四核心嵌入式处理器仿真测试

OpenMP 编译器使用 Omni 编译器作前端,



sparc 编译工具链作后端<sup>[3]</sup>,采用 Gaisler 公司的 GRSIM 模拟器,GRSIM 是 LEON 核心 SMP 系统模拟器,它可以模拟多处理器 LEON3 系统,在实验中将其配置为四核心的 FPEP 平台,GRMON 调试连接器用于连接目标板进行测试。

表 5 为 3 种 SUSAN 并行化算法在四核心平台的 GRSIM 模拟器上进行性能分析的结果,所采用的计时器为 ecos 内核支持的 *cyg\_current\_time()* 函数。由于嵌入式平台处理速度慢,存储空间有限,因此实验中首先将 SUSAN 算法简化,只对 SUSAN 算子进行并行化,并选用大小为 8KByte 的小图像进行测试。在嵌入式多处理器平台下进行图像的载入和存储空间的分配是较难处理的部分。由表 5 可知,算法 Susanp4 和 Susann4 的效率相当,但是较非嵌入式平台加速比低,还需进一步的时间和空间上的优化。Susanm4 算法加速比较高,因为只是重复简单的四倍的 Susans 的计算量,OpenMP 在此平台下的开销较小。

表 5 SUSAN 并行化算法 GRSIM 仿真效率比较

算法	处理时间	加速比
Susans	370	1.00
Susanp4	131	2.82
Susann4	140	2.64
Susanm4	380(4)	3.89

注: Susans 为串行算法; Susanp4 为 4.1 算法; Susann4 为 4.2 算法; Susanm4 为 4.3 算法。

5.3 FPE 的 FPGA 验证平台测试

使用四核心嵌入式处理器 FPEP 的 FPGA 验证平台,为了与模拟器环境进行比较,采用与 5.2 节同样的图像和简化算法。表 6 为 3 种 SUSAN 并行化算法在FPGA验证平台的进行性能分析的结果。

表 6 SUSAN 并行化算法 FPE 的 FPGA 验证平台效率比较

算法	处理时间	加速比
Susans	383	1.00
Susanp4	142	2.69
Susann4	149	2.57
Susanm4	401(4)	3.82

注: Susans 为串行算法; Susanp4 为 4.1 算法; Susann4 为 4.2 算法; Susanm4 为 4.3 算法。

实验表明,FPGA 验证平台的性能略低于 GRSIM 模拟器的仿真结果,但仍有明显的性能提高,加速比超过 2.5。

6 结 论

本文提出了四核心嵌入式并行处理器 FPEP 的结构设计,并建立 FPGA 验证平台,通过典型的图

像识别算法 SUSAN 的并行化设计和分析进行性能评测。

在 OpenMP 并行标记语言的基础上,详细描述了对 SUSAN 算法在不同粒度划分下进行并行化的 3 种实现方法,并分析 3 种算法在 Intel 四核平台和四核心嵌入式并行处理器 FPEP 平台下的效率。通过分析可知:

- (1) 内部直接并行算法和分块处理算法适合于大图像的处理,如有多个图像时可采用多图像并行处理,也可 3 种算法结合起来使用。
- (2) 3 种算法对四核系统能达到 3.0 左右的加速比,效率提高明显,但是算法本身还存在一定缺陷,需要进一步优化以提高加速比。
- (3) 四核心嵌入式并行处理器 FPEP 在执行内部直接并行算法和分块处理算法时性能低于 Intel 四核心处理平台,但仍能达到 2.5 以上;而在多图像并行处理算法时性能高于 Intel 四核心处理平台,接近 4.0。

参 考 文 献

[1] McNairy C, Bhatia R. Montecito: A dual-core, dual-thread itanium processor. IEEE Micro, 2005, 25(2): 10-20

[2] Kalla R. IBM power5 chip: A dual-core multithreaded processor. IEEE Micro, 2004, 24(2): 40-47

[3] Mladen B, Hans J S, Peter P. Multicore system-onchip architecture for MPEG-4 streaming video. IEEE Transactions on Circuits and Systems for Video Technology, 2002, 12(8): 688-699

[4] Lee Trong-Yen, Fan Yang-Hsin, Cheng Yu-Min et al. Hardware-oriented partition for embedded multiprocessor FPGA systems//Proceedings of the 2th International Conference on Innovative Computing, Information and Control (ICICIC 2007). Kumamoto, Japan, 2007

[5] Toledo F, Martinez J, Ferrandez J. FPGA-based platform for image and video processing embedded systems//Proceedings of the 2007 3rd Southern Conference on Programmable Logic (SPL'07). Mar del Plata, Argentina, 2007: 171-176

[6] Hofmann Andreas, Waldschmidt Klaus. SDVMR: A scalable firmware for FPGA-based multi-core Systems-on-Chip//Proceedings—IEEE Computer Society Annual Symposium on VLSI: Trends in VLSI Technology and Design, ISVLSI. Montpellier, France, 2008: 387-392

[7] Noseworthy Joshua, Leeser Miriam. Efficient communication between the embedded processor and the reconfigurable logic on an FPGA. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2008, 16(8): 1083-1090

[8] Kleinosowski A, Lilja D J. MinneSPEC: A new SPEC benchmark workload for simulation-based computer architecture research. University of Minnesota ARCTiC Lab, 2002-10

- [9] Levy Markus. Keynote 1: Using EEMBC benchmarks to understand processor behavior in embedded applications//Proceedings of the 1st International Conference on High Performance Embedded Architectures and Compilers (HiPEAC 2005). Lecture Notes in Computer Science 3793. Barcelona, Spain, 2005: 3-4
- [10] Guthaus M R, Ringenberg J S, Ernst D et al. MiBench: A free, commercially representative embedded benchmark suite//Proceedings of the IEEE International Workshop on Workload Characterization. Los Alamitos, CA: IEEE Computer Society Press, 2000: 266-277
- [11] Li Yong, Wang Zhi-Ying, Zhao Xue-Mi et al. Design of a low-power embedded processor architecture using asynchronous function units//Proceedings of the 12th Asia-Pacific Computer Systems Architecture Conference (ACSAC 2007). Lecture Notes in Computer Science 4697. Seoul, South Korea, 2007: 354-363
- [12] Yan Li-Ke, Shi Qing-Song, Chen Tian-Zhou et al. An on-chip communication mechanism design in the embedded heterogeneous multi-core architecture//Proceedings of the 2008 IEEE International Conference on Networking, Sensing and Control (ICNSC). Sanya, China, 2008: 1842-1845
- [13] Park Gi-Ho, Lee Kil-Whan, Han Tack-Don et al. Cooperative cache system: A low power cache system for embedded processors. IEICE Transactions on Electronics, 2007, E90-C(4): 708-717
- [14] Vermeulen Bart. Functional debug techniques for embedded systems. IEEE Design and Test of Computers, 2008, 25(3): 208-215
- [15] Gaisler J. A portable and fault-tolerant microprocessor based on the SPARC v8 architecture//Proceedings of the International Conference on Dependable Systems and Networks. Washington, DC, United States, 2002: 409-415
- [16] Smith S M, Brady J M. SUSAN — A new approach to low level image processing. International Journal of Computer, 1997, 25(1): 45-78
- [17] Cuvillo J D, Zhu W, Gao G R. Landing OpenMP on Cyclops-64: An efficient mapping of OpenMP to a many-core system-on-a-chip//Proceedings of the CF'06. Ischia, Italy, 2006: 41-50



**WANG Jie**, born in 1979, Ph. D. candidate. His research interests focus on parallel computing and FPGA validation.

**ZHANG Shu-Yan**, born in 1986, master candidate. Her research interests focus on embedded parallel computing and

parallel algorithm.

**LIU Tao**, born in 1979, Ph. D. candidate. His research interests focus on embedded multi-core processor.

**JI Zhen-Zhou**, born in 1965, professor, Ph. D. supervisor. His research interests include computer architecture, network security and parallel computing.

**HU Ming-Zeng**, born in 1935, professor, Ph. D. supervisor. His research interests include computer architecture, network security and parallel computing.