

基于 NEON 优化技术的视频处理系统设计

陈 炎^{1,2}, 袁国顺¹, 刘小强^{1,2}

(1 中国科学院微电子研究所, 北京 100029; 2 中国科学院大学, 北京 100049)

摘 要: 针对软件方式的嵌入式图像处理, ARM Cortex-A 系列的处理器集成了基于 SIMD 指令架构的 NEON 协处理器引擎, 能够实现一条指令同时对多个数据进行并行化处理, 可以有效加速多媒体和信号处理算法. 本文设计了一种基于 FPGA 的视频处理系统, 充分利用 NEON 技术对 Sobel 算法进行优化加速, 并与普通软件方法实现算法的方式进行对比, 结果显示, 采用 NEON 技术后图像处理加速显著, 且视频流畅度好.

关键词: NEON 技术; SIMD 指令架构; Sobel 算法

中图分类号: TP312

文献标识码: A

文章编号: 1000-7180(2018)07-0062-05

The Design of Video Processing System Based on NEON Optimization Technique

CHEN Yan^{1,2}, YUAN Guo-Shun¹, LIU Xiao-Qiang^{1,2}

(1 Institute of Microelectronics of China Academy of Sciences, Beijing 100029, China;

2 University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Aimed at ARM-based embedded image processing, the ARM Cortex-A series of processors are integrated the NEON coprocessor engine which is based on the SIMD instruction architecture, enabling an instruction to parallelize multiple data simultaneously, which can accelerate multimedia and signaling processing algorithms (such as video codecs, image processing techniques) effectively. In this paper, we design a video processing system based on FPGA, which can make full use of NEON technology to optimize the Sobel algorithm, compared with the way of ordinary software method to implement the algorithm, the results show that NEON technology accelerates image processing significantly, decoded the video more fluent.

Key words: NEON technology; SIMD instruction architecture; sobel algorithm

1 引言

随着移动通信技术的发展,人们越来越多地在手机等移动端设备进行多媒体影音的访问,然而在移动端视频处理系统中,一方面由于动态图像数据处理的运算量较为庞大,且多为并行计算,而传统的嵌入式视频处理平台大多是基于 DSP 等顺序处理引擎,并不太适合做并行度高且较为复杂的图像处理算法,处理视频时不但效果和流畅度相对较差,也会消耗大量的硬件资源;另一方面,目前在移动平台上许多密集型应用,如音视频应用、2D/3D 游戏等对嵌入式处理器的峰值计算能力提出了很高要求,

但这类应用中的数据长度往往少于 32 位,当数据进行运算时,32 位通用寄存器中的一些运算单元并不需要参与,这容易导致电量的浪费,对电量有限的移动设备来说是非常严重的问题^[1]. 所以高效率和低功耗的应用方案目前仍是移动端多媒体访问最迫切的需求.

本文所设计的基于 FPGA 的视频处理系统,采用 ARM Cortex-A9 NEON 加速引擎,利用 NEON 引擎具有扩展的 SIMD 指令集和专用的寄存器系统,极大改善了嵌入式系统在实际视频流的处理性能,并尽量采用片上资源降低成本和开发调试难度,为打算使用 NEON 技术进行算法加速的研究人员

和应用开发者提供参考和借鉴。

2 NEON 技术原理

2.1 ARM Cortex-A9

ARM Cortex-A9 是基于 ARMv7 架构,具有高效率、长度动态可变、多指令执行的超标量的体系结构^[2]。作为嵌入式处理器,Cortex-A9 既可以用作单核处理器也可配置成多核处理器,同时可提供可合成或宏实现,从而成为需要在低功耗或散热受限的成本敏感型设备中提供高性能设计的理想解决方案。

2.2 SIMD 指令架构和 NEON 技术

SIMD(Single Instruction Multiple Data)即单指令多数据流的指令架构,是相对于传统处理器上使用 SISD(Single Instruction Single Data)单指令单数据流而言的。从图 1 可以看出 SIMD 能够采用一条指令来处理多个数据,实现针对同种数据类型和长度的多个数据进行并行操作。从时间上来说一条指令执行的时间,对于 SIMD 和 SISD 是差不多的,由于 SIMD 一次可以处理 N 个数据,所以其消耗的时间也就缩短到了 SISD 的 1/N。

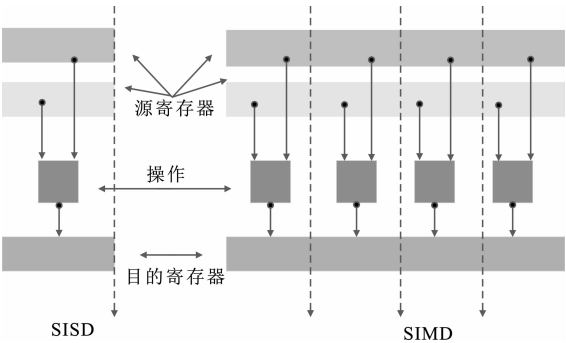


图 1 SISD 和 SIMD 指令架构对比

SIMD 指令集技术于 1996 年由 Intel 在奔腾处理器中首次引入,旨在增强处理器在多媒体方面的处理能力。ARM 公司于 ARMv6 架构中首次引入了简单的 SIMD 指令集,具体表现即是 NEON。

ARM Cortex-A9 的 NEON 协处理器的设计主要基于 ARMv7 SIMD 和向量浮点计算 VFPv3 (Vector Floating Point)指令集进行的,在具体的芯片设计中 NEON 组件是可选的,该组件主要面向视频音频编解码、图像语音信号处理和其他一些基带信息处理领域^[3]。NEON 协处理器包括 16 个 128 位的寄存器,拥有独立的寄存器系统和硬件执行单元,支持 8 位、16 位、32 位、64 位等数据类型的向量运算。此外 NEON 指令集能与 ARM 或 Thumb 指

令混合使用,ARM 或 Thumb 指令集管理所有的程序流和同步,而 NEON 指令集可以实现结构化的存储访问、NEON 寄存器和通用寄存器之间的数据拷贝、数据类型的转换、算数和逻辑并行处理等操作^[4]。

3 基于 FPGA 的视频处理系统设计

3.1 基于 FPGA 的视频处理系统整体结构

本文所用的 FPGA 平台为 Altera Cyclone V,具体的器件为 DE10-Standard 5CSXFC6D6F31C6,视频采集系统如图 2 左边所示,将采集到的分辨率为 800×600 的视频信号通过一个 HDMI 转 HSMC 的转接板输入到 FPGA 中,通过一个异步 FIFO 隔离不同的时钟域,利用 Avalon 总线特有的流传输功能(Avalon-ST),通过调用一个 DMA 模块将 FIFO 中的数据搬运到 Cortex-A9 核所控制的 DDR3 中,然后通过 A9 中的 NEON 加速引擎对图像进行边缘检测处理,优化 Sobel 算子的计算过程。将图像处理后的结果送到 VGA 缓冲器(SDRAM)暂存,实例化一个 VGA 系统 IP(内部有 DMA 控制器,双时钟 FIFO,VGA 控制器)再次使用流传输将分辨率为 800×600 图像边缘检测的计算结果打包成 VGA 协议的数据格式送到 LCD 显示器上,显示出边缘检测处理后的图像。

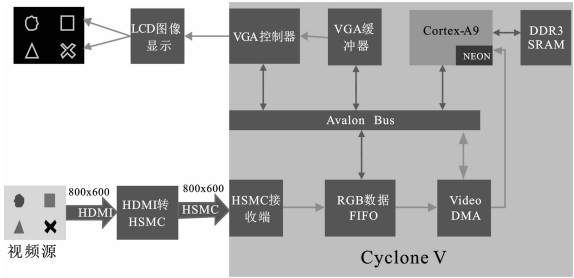


图 2 基于 FPGA 的视频处理系统整体结构

3.2 Sobel 边缘检测算法

图像的边缘包涵了该幅图像丰富的特征信息,因此对图像进行边缘检测是图像处理中非常重要的内容。如图 3 所示,Sobel 边缘检测算子主要利用 3×3 的像素模板,对中心像素点和周围 8 个像素点进行卷积运算,求出水平和垂直方向的梯度,即得出了该中心像素点的梯度值,再与设定的阈值作比较,根据图像的边缘检测效果再进行相应的调整。

图 4(a)为一帧图像 3×3 的区域,P 为各点的灰度值,(b)和(c)分别为 Sobel 算子的水平、垂直方向的算子。

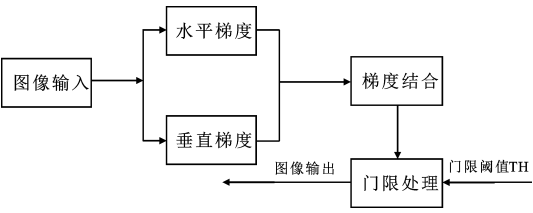


图 3 Sobel 边缘算法

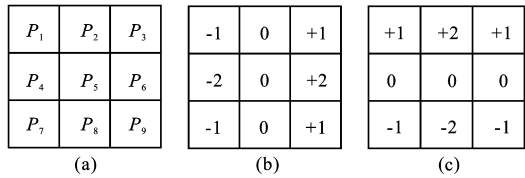


图 4 一帧图像的 3×3 区域和 Sobel 算子

将(a)中的9个点的像素灰度值分别与 Sobel 水平(b)、垂直(c)两个方向的算子做卷积运算可得 P_5 点的水平、垂直方向的梯度分量 f_x, f_y 。

$$f_x = (P_3 + 2P_6 + P_9) - (P_1 + 2P_4 + P_7) \tag{1}$$

$$f_y = (P_1 + 2P_2 + P_3) - (P_7 + 2P_8 + P_9) \tag{2}$$

$$|F_{P_5}| = \sqrt{f_x^2 + f_y^2} \tag{3}$$

将 $|F_{P_5}|$ 与设定的梯度阈值 TH 作比较,如果 $|F_{P_5}| > TH$,则判定 P5 点为边缘点灰度值为 0,否则为内部点灰度值为 255. 然后根据结果调整阈值,得到比较好的图像边缘效果^[4]。

3.3 NEON 技术具体实现方法

NEON 技术主要有四种应用方法:自动行量化、NEON 内联函数、汇编代码和第三方的 NEON 库来实现具体的应用. 其中,汇编代码的方法优化性能最高,但是编写出最优化 NEON 汇编代码的难度较高;自动行量化的方法最为简单,但优化性能最差;第三方函数库的方法平台可移植性最差;而 NEON 内联函数的方法编写程序较容易,优化性能相对较高,可移植性好. 故本文采用 NEON 内联函数(Intrinsics)的方法进行 NEON 技术的应用。

NEON 内联函数是由 ARM 定义的一组全新的数据类型和内联函数,便于使用 C 语言直接访问 NEON 寄存器. 虽然在 C 程序中,内联函数和普通的 C 函数使用一样,但是在编译时,这些内联函数会直接映射为 NEON 提供的向量指令. 由于 GCC 编译器和 ARM 编译器都支持 NEON 内联函数,我们需要在文件开头加入“arm_neon.h”头文件,以便编译器能够识别 NEON 类型的变量。

由于 ARM Cotex-A9 处理器的 NEON 为可选功能,故在使用之前要使能 NEON 编译器选项. 开

启 ARM 提供的 NEON 向量化编译器,需要编译时添加适当的命令行参数,指明所用的 CPU 类型、浮点处理器和 ABI 接口类型、NEON 和 VFP 类型等:

```
-mcpu=cortex-a9 -mfloat-abi=softfp -mfpu=neon-fp16
```

开启 NEON 功能后,就可以用 NEON 内联函数编写代码进行算法优化. 算法主要实现步骤如下:

(1)分别将 R,G,B 三个通道的数据按照一定的格式和顺序加载到 NEON 寄存器中. DDR3 中存储的每个像素点包括 R,G,B 三通道各 8 位,通过 NEON 指令提供的交叉存取的功能可以实现各像素点红、绿、蓝通道数据的分离. 由于 NEON 寄存器最大为 128 bit,考虑到接下来计算灰度时可能会出现溢出,故一次可以实现将 8 个像素点转成 3 个 NEON 向量,每个点占 8 位,共 8 路. 过程如图 5 所示:

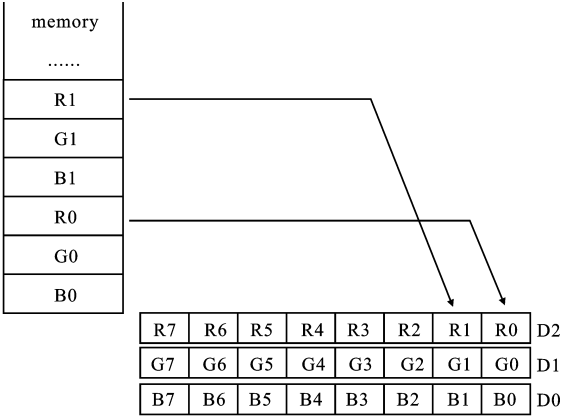


图 5 RGB 图像经过交叉存储成了三个向量

(2)计算输出图像的灰度. 按照公式(4)将 RGB 图像转化为灰度图. 与 CPU 计算的不同之处在于, NEON 寄存器将存储的数据视为矢量,各元素的运算互不干扰,能够对同种数据类型和长度的多个数据进行并行操作. 考虑到在计算灰度的过程中可能会有进位的情况,所以先把 RGB 转成的 8 bit 宽 8 通道向量扩大成 16 bit 宽 8 通道向量,然后计算灰度,灰度计算完后,再通过缩减位宽,转回成 8 bit 宽 8 通道向量,这样就完成了如图 6 所示的一次计算 8 个点的灰度值。

$$\text{Gray} = (R * 77 + G * 150 + B * 29) >>> 8 \tag{4}$$

(3)取三行灰度图的像素点,转成成如图 7 的三个向量,考虑到之后的计算过程中可能会出现负值,先把灰度像素点转成的 8 bit 宽 8 通道的三个向量扩大成 16 bit 宽 8 通道的三个向量,再把这三个无符号型的向量转换成有符号向量. 由于每个像素点的 Sobel 算子梯度计算需要周围 8 个点的数

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24

图 6 每次计算 8 个像素点的灰度值

利用 NEON 寄存器计算一次可以得到连续 6 个点的 Sobel 算子梯度值。

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24

图 7 每次计算 8 个灰度像素点的 Sobel 算子梯度值

(4) 将 Sobel 算子梯度值的计算结果存入 DDR3 中, 将得到梯度值的像素点与阈值按公式(3)进行比较, 如果大于阈值, 判定 P5 点为边缘点灰度值为 0, 否则为内部点灰度值为 255. 然后根据结果调整阈值, 得到比较好的图像边缘效果。

4 NEON 加速效果演示与分析

4.1 阈值调整效果演示

按图 2 搭建好 FPGA 视频处理硬件平台, 将边缘检测的软件程序下载到 FPGA 中, 在一台显示器上放视频, 将采集到的信号通过 FPGA 视频处理系统处理后, 使用 VGA 输出计算结果到另一台显示器上, 播放处理后的视频, 采集的图像和输出的图像分辨率都是 800×600. 根据图像的边缘效果进行阈值调整过程如图 8 所示。

图 8 里(a)是原始 RGB 图像, (b)是处理后阈值较小的图像, (c)为阈值较大的图像, (d)是阈值较为适中的图像. 图像的阈值设定要根据做完 Sobel 运算后图像的期望和标准差来设定, 为简化处理, 视频系统中对于每帧图像的边缘检测处理采用统一的全局阈值. 图像的边缘检测效果与阈值的设定值密切相关, 可以看出, 阈值大时, 提取的信息较少, 如人物和背景的一些细节都被过滤掉了, 而阈值较小时提取的边缘信息较多, 图像的细节比较丰富. 阈值调整同时也要根据对边缘信息的需求去操作。

4.2 NEON 加速效果演示

图 9(a)、(b)上图都是采集的原始 RGB 图像, 中图都是使用 CPU 软件计算出的边缘检测图像,



图 8 图像的阈值调整与边缘效果



图 9 NEON 运算处理后和 CPU 处理后的图像效果

下图是使用 NEON 引擎算出的边缘检测图, 可以看到图像的边缘轮廓很清晰的显现出来, 符合预期的效果, 因为算法一致的原因, 只是实现手段不同, 二种方法对图像的处理效果总体相差不大. 但是 NEON 处理的速度明显比单纯 CPU 软件计算的方法快, 视频流畅度也较好。

4.3 NEON 加速性能对比和分析

本系统使用 CPU 自带定时器对两种处理方式计时, 观察到由 CPU 和 NEON 引擎分别完成一帧

图像的边缘检测所需的时间如表 1 所示:

表 1 NEON 处理和 CPU 处理性能对比

图像处理方法	CPU	NEON
时间/ms	1403	121

由表 1 可以看出使用 NEON 引擎计算速度,是 CPU 软件计算的 11.6 倍,速度有明显提升.对分辨率为 800×600 ,帧率为 60 fps 的输入图像处理速度达到了 8.3 fps,而使用 CPU 处理的速度仅为 0.7 fps.

本系统仅由 CPU 软件实现的算法程序是使用 C 语言编写的,NEON 方法的算法程序是使 NEON C 内联函数实现的.两者均为软件处理图像的方法.由于 NEON 协处理器由于拥有独立的寄存器系统和硬件执行单元,可以实现并行运算,大大提高了图像处理速度.本设计中 Sobel 算法并不复杂,程序代码不多,所以 NEON 的处理速度大约是传统软件方式处理的 11.6 倍,可以预见两者的差距会随着数据量的增加和算法复杂度的升级而加大.

5 结束语

本文针对嵌入式软件方式的视频处理研究,提出一种基于 NEON 优化技术的视频处理系统,尽量采用基于 FPGA 的片上资源,充分利用了 Avalon 总线特有的流传输功能,有效的缩短了数据传输的时间;采用 NEON 技术对算法进行优化,与仅用

CPU 软件实现算法的性能对比,从结果上可以看出 NEON 技术优化后的算法,不仅计算时间明显缩短,处理后视频流畅度较好.同时,本文所设计的视频处理系统很好地降低嵌入式视频处理系统的开发调试难度,节约了成本,为打算使用 NEON 技术进行算法加速的研究人员和应用开发者提供参考和借鉴.

参考文献:

[1] 陈泽新. 基于 NEON 引擎的 JM 解码器并行优化实现[D]. 成都:西南交通大学,2010.

[2] 简欢. 基于 ARM NEON 的 H. 265 解码器优化[J]. 电视技术,2014,38(15):102-105.

[3] 陈泽新,杨斌. ARM Cortex-A9 的 NEON 技术研究及应用[J]. 器件技术,2013(10):46-49.

[4] 杨新华,寇为刚. 基于 FPGA 的 Sobel 算子图像边缘检测技术[J]. 表技术与传感器,2013(1):102-104.

作者简介:

陈 炎 男,(1991-),硕士研究生.研究方向为 SOC 设计.
E-mail: chenyan790779514@163.com.

袁国顺 男,(1966-),硕士,研究员,博士生导师.研究方向为数模混合集成电路设计.

刘小强 男,(1990-),博士研究生.研究方向为图像处理和 SOC 设计.

(上接第 61 页)

[5] 万晓凤,康利平,余运俊,等. 互补滤波算法在四旋翼飞行器姿态解算中的应用[J]. 测控技术,2015,34(2):8-11.

[6] 景丽. 基于卡尔曼滤波组合导航算法的计算量与精度分析[D]. 哈尔滨:哈尔滨工业大学,2014.

[7] 秦永元. 惯性导航[M]. 北京:科学出版社,2006:287-304.

[8] 王励扬,翟昆朋,何文涛,等. 四阶龙格库塔算法在捷联惯性导航中的应用[J]. 计算机仿真,2014,31(11):56-59.

[9] 徐泽远,伊国兴,魏振楠. MEMS-SINS/GPS 组合导航

系统设计[J]. 自动化仪表,2016,37(12):67-70.

[10] Bhogadi L R, Gottapu S B R, Konala V R. MicroBlaze implementation of GPS/INS integrated system on Virtex-6 FPGA[J]. Springerplus, 2015, 4(1):629.

作者简介:

刘强军 男,(1993-),硕士研究生.研究方向为数字 IC 设计.

张 岩(通讯作者) 男,(1969-),教授.研究方向为专用处理器设计. E-mail: ianzh@foxmail.com.