Design of the Image Accelerator Based on FPGA

Jia Liang[1,a], Yang Huichao[2,b,*] and Yin Yi[3,c]

[1]Electronic and Information Engineering, Shenyang Aerospace University, China

[2] Electronic and Information Engineering, Shenyang Aerospace University, China

[3] Electronic and Information Engineering, Shenyang Aerospace University, China

[a]342747947@qq.com, [b]1440697154@qq.com, [c]410529437@qq.com

* Yang Huichao

Keywords: FPGA; Hardware acceleration; Image edge detection.

Abstract. This thesis introduces a kind of image accelerator design based on FPGA, Hardware acceleration achieve the purpose of accelerating by increasing operation parallelism, the DMA controller accesse memory, and the DMA controller gives address and control signals, in order to avoid conflict due to multiple main equipment access the memory at the same time, therefore introducting arbitrator (Arbiter), who decided which module to get access to the memory bus control, processing the image through the edge detection accelerator,the experimental results show that it can be real-time and efficiently complete the image processing, it can be applied to the image, video, and other fields.

基于FPGA的图像加速器的设计

贾亮[1, a]，杨慧超[2,b,*]，尹伊[3,c]

[1]沈阳航空航天大学电子信息工程学院，沈阳，辽宁，中国

[2]沈阳航空航天大学电子信息工程学院，沈阳，辽宁，中国

[3]沈阳航空航天大学电子信息工程学院，沈阳，辽宁，中国

[a]342747947@qq.com , [b]1440697154@qq.com, [c]410529437@qq.com

[*]杨慧超

关键词: FPGA；硬件加速；图像边缘检测；fpga流水设计

中文摘要. 介绍了一种基于FPGA的图像加速器的系统开发。通过fpga的并行运算特性来进行硬件加速，由DMA控制器发起对内存的访问，并由DMA控制器给出地址和控制信号，在系统开发的过程中加入分配硬件资源的模块(Arbiter)，从而处理多个模块共同访问cpu的问题，使得系统有序的运行，决定谁取得访问内存的总线控制权，之后经由边缘检测加速器来处理图像，实验结果表明它可以实时、高效地完成图像处理，可以应用到图像、视频等领域。

1. 引言

在工业应用中,我们经常选取DSP或者ARM作为处理器完成信息的处理.为了能够处理更多的数据和复杂的功能，单一类型的处理器已经无法解决这些问题，因此需要将不同的处理

Copyright © 2018, the Authors. Published by Atlantis Press.
This is an open access article under the CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/).

器结合起来，将系统任务进行划分，交给不同的处理器完成，利用FPGA的高速特性，来进行控制和数据传输，dsp具有数字运算的特性完成相应的数字运算。

在嵌入式项目开发中，首先要考虑使用什么处理器进行开发，包括图像处理、压缩、通信。不同的项目开发需要不同的处理器，有些情况需要使用专用的处理器来进行开发，即使使用某种处理器的最新系列也不一定能够完成，能够完成结果也不会很好。而用合适的处理器不仅加快项目开发，而且能够较少资金的不必要开支。

2．总体架构设计

该架构既支持cpu读写memory，也支持cpu读写acc，acc读写memory，其中CPU为8位，地址总线16位，虚拟CPU（内部不写指令，仅做一次长整加速器的运算）唯一的加速器设备：长整加速器，之所以加入长整形加速器，那是因为在fpga内部无法对浮点进行运算，虽然没有浮点处理机制，但是fpga仍然可以处理浮点，那是因为可以通过移位使其进行运算处理，之后在将其恢复即可，因此为了在获得CPU授权（访问它的内部寄存器）后，通过对Memory资源的直接访问，完成64位无符号长整的加减乘除算术运算加速器运算结束后，在系统中有处理器和DMA物理器件，当它们共同向存储器发出访问命令，那么就会产生错误，一个设备同一个时刻只能有一个设备进行访问，为了使设备都能够进行访问同一个设备，那么就必须对不同的访问指令进行优先级处理，之后根据指令先后次序对被访问设备进行时分复用，使用中断信号通知CPU仲裁器采用唯一申请者获得仲裁的仲裁策略，当其他主设备要访问内存，必须先向处理器发出请求信号，由该处理器决定哪个设备可以访问内存，进行长整型运算时，将源（Source）与目标（Target）读入加速器，执行64位的运算，并将64位的运算结果写回目标（Target）存储器数据总线8位，该总体架构设计如图1所示。

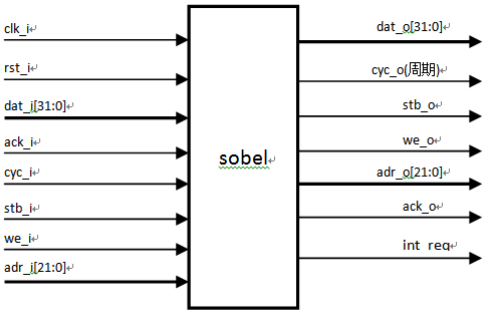

图1　总体架构设计

3．算法分析

3.1　算法对比

设执行算法核心所占的时间比例为f。那么在整个运行周期内余下的时间为(1- f)，所以可以得出下式：

$$t = ft + (1-f) t \tag{1}$$

当在该系统下运行算法时，处理所有的操作比之前快s倍，那么时间就相应的减少为原来的1/s，而上式中余下的时间并不发生变化，因此可以得出总的运行时间如下：

$$t' = ft/s + (1- f) t \tag{2}$$

通过(1)可以得出不使用硬件加速的时间，(2)式可以得出硬件加速之后的总时间，两者的比例系数就是总体性能的提高比例。提高的比例如下：

$$S' = 1 / (f/s+(1-f)) \tag{3}$$

最后得到的公式叫Amdahl法则，Amdahl是因为Gene Amdahl而出名，Amdahl是并行计算的先驱之一。该式表明算法核心在整个算法执行时间中所占的比例很大程度影响核心算法通过加速处理所带来的整体性能的提高。

假设在嵌入式处理器上运行的算法可以分成不同的部分，执行每一部分所花费的时间可以估算出来，而该算法可分为两个部分，即分成两个核心（kernel）算法，一个占总时间的80%，而另一个只占总时间的20%。若用一个硬件加速器来加速算法核心，将运行速度提高到原来的10倍，那么可以得出之前分出的算法加速后的性能提升：

$$1/((0.8/10)+(1-0.8)) =3.57 \tag{4}$$

对第二个算法核心进行加速处理带来的总体性能提升为：

$$1/((0.2/10)+(1-0.2)) =1.25 \tag{5}$$

3.2 算法对比

从结果对比可以看到，虽然第二个算法核心提高的运算速度是第一个的10倍，但是由于它在原来总运行时间里所占的比例较低，对其进行加速处理也只能带来较小的性能改善。而对第一个算法核心进行加速处理会对总体性能产生更显著的效果。所以我们在开始正式设计硬件时必须找到主要矛盾，即占总运行时间比例较高的部分，针对这部分算法程序进行硬件加速器的设计 ，以期望取得显著效果，算法模块设计如图2所示。

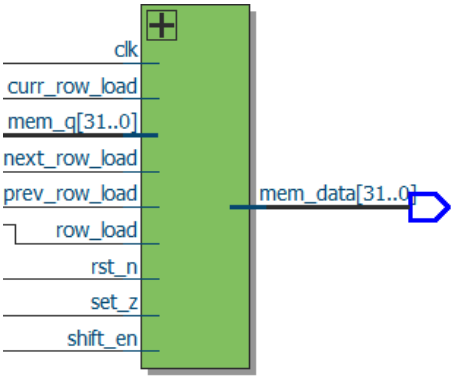

图2 算法模块设计

4. 图像边缘检测器

图像边缘检测法的原理是中心像素点和它周围的八个像素点都同时乘以一个系数(通常是一个 卷积表（convolution mask）)后相加的一个卷积过程，通过该过程来把每个像素点在每个方向上的导数值估计出来。该Sobel卷积表的Gx和 Gy可以分别用于计算x和y方向的导数值。Dx和 Dy是中心像素点和它周围的八个像素点都同时乘以一个系数(通常是一个卷积表（convolution mask）)后相加，这样就把x和y方向的偏导数值分别求了出来，Gx和 Gy 如下图3所示。

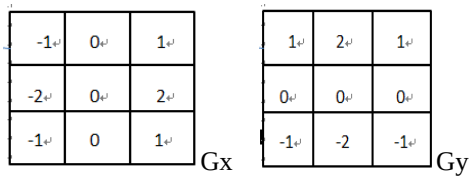

图3 sobel卷积表

然后，利用(6)式将中心像素点的导数求出来。计算公式如下式所示：

$$|D| = \sqrt{D_x^{\,2} + D_y^{\,2}} \tag{6}$$

为了便于硬件模型的设计，减少资源的使用，并且使误差在允许的范围内，将上式进行简化：

$$|D|=|D_x|+|D_y| \tag{7}$$

图4展示了每个操作步骤中的具体数据，通过并行操作将原始图像从上层像素开始到底部像素的导数计算出来。对于系数值是0的部分积我们已经省略，因为该操作没有任何意义。在模块设计时通过状态机发出不同指令，依次产生不同行的地址，每次取32位数据进行流水操作，这里采用的是全流水操作，将每一拍都进行合理的规划，产生的导数根据地址产生模块向存储器发出的地址进行相应的数据存储。

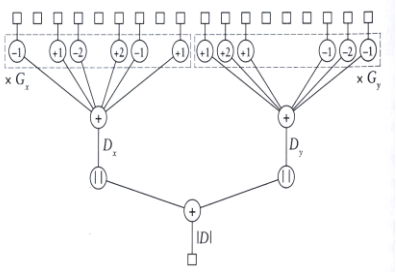

图4　sobel数据操作

该架构是独立的图像边缘检测硬件，原始图像（亮度），储存在memory中，忽略摄像头与memory的写显存操作，用偏导数的绝对值计算，仍然采用水平偏导Dx和垂直偏导Dy绝对值之和，全速流水作业，加速作业时，无任何空闲周期，其架构如图5
所示，在该架构中图像采用的是600*400大小，在存进memory时，它的起始地址为0，导数的起始地址为100000H，在接受到start命令后开始对图像进行读取，并开始流水作业，通过sobel算法模块进行边缘处理。

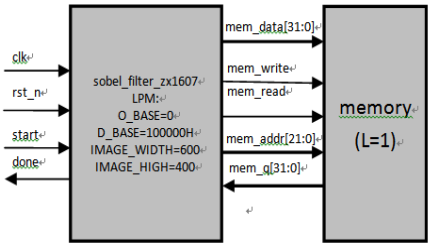

图5　sobel_filter架构设计

5．系统仿真及结果测试

使用Verilog语言编写，利用 Altera公司quartus II开发软件进行设计调试，首先通过Photoshop CS3对图片的灰度进行处理,首先设置图像的位宽为8，灰度按位宽的数值大小将由黑到白的区域为0-256，得到灰度图,为方便处理将格式修改为bmp格式,在bmp格式选项中将位深度改为8位，之后通过Matlab软件将灰度图读取并转换成Verilog的存储器初始化txt格式,在quartus的Sobel_Filter模块输出亮度倒数图像txt文件,再通过Matlab转换为导数图像显示出来，为了更好的进行研究，对图像进行阈值切割得到边缘图像，为了更加清晰的看到各模块的层次结构，用word将各个模块的放在一起，各模块分别是状态机、地址生成模块和算法模块，当该整体模块接受到上层发出的start命令后，状态机开始进行工作，首先对地址模块发出使能信号，使其产生地址，并对存储模块进行读取，发出相应的图像数据，之后将数据存到相应的行寄存器中，状态机发出相应的命令控制算法模块进行工作。

最后开始分析波形仿真图，如图6所示。

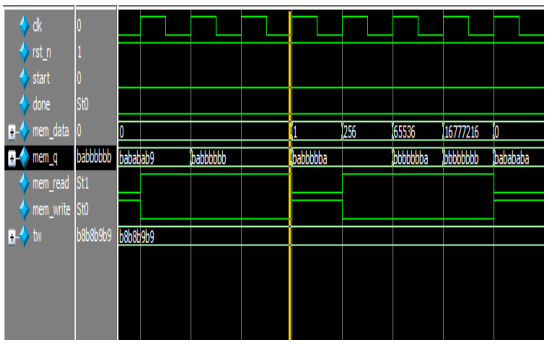

图6 波形仿真图

mem_q[31:0]代表从memory中读取的数据，是在状态机向memory发出mem_read信号后，memory发出的数据，done信号表示该模块是否完成，当一副图像的全部像素都转换为亮度导数后，done置为1，mem_q数据进过sobel算法模块处理后，输出mem_data,并且状态机会发出对应的地址控制数据的存储位置，将mem_data波形的十六进制数据与亮度导数图像txt文件进行对比数据没有发生错误。

6. 结 论

每计算一个像素的导数需要周围8个像素的数值，计算全部像素必须与视频播放同步，即计算一帧像素的时间必须小于等于视频播放时一帧的时间（帧周期），即计算一个像素的平均时间必须小于等于视频播放时一个像素的传输时间,“带宽”的概念即是在一个像素的平均传输时间内（100ns）完成导数计算所需要的时间比例。或者说，是在一帧视频播放周期内，完成导数计算诸操作所需要的时间比例如果每次单独读取全部9个像素，需要9×20ns=180ns视频播放时每像素的平均传输时间只有100ns这样，导数的计算就跟不上视频的播放了,此时，仅读像素的操作就占据180/100=180%，现在的做法是：

首先分别读入相邻三行的一个字（32bit=4byte）这用去了三个读周期，即3×20ns=60ns,然后将行寄存器中的最左列的三个像素放入计算阵列中,然后流水线计算出导数将行寄存器中左移动一次将行寄存器最左端移入计算阵列,计算阵列同样左移一次,使用流水线计算出该导数将行寄存器中的第3列移入计算阵列,使用流水线计算出该导数将行寄存器中的第4列移入计算阵列,使用流水线计算出该导数，到目前为止，完成4个像素的导数计算，使用了三个字的读周期即60ns随后继续读入三个字，重复上述过程，计算4个像素这样平均400ns周期内，读内存+读视频+写导数=100ns，占100/400=25%剩余的75%可以用于其他操作。通过计算可以发现通过硬件来进行单独的算法时，尽管会增加额为的硬件资源，但是不可否认，运算速度会大大增加，这样cpu可以有更多的空闲周期，这样能够处理更多的任务。

References

[1] Zhang Yuxi, Wang Jun, Yin Han. Development and research of experimental platform of FPGA+MCU [J]. Industrial and informatization education. 2015(05):3-7

[2] Wei Yun. Design of PCIe bus DMA platform based on FPGA [D]. Wuhan university of technology, 2013

[3] Xia min. Design of high-speed data transmission system based on PCI Express interface [D]. Beijing jiaotong university 2012

[4] Ren Lianfang. Data transmission and storage based on PCI Express bus [D]. Nanjing university of technology 2010

[5] Ji ZhongKai. Design of high-speed serial data transmission system based on FPGA with flow control mechanism [J]. Electronics world. 2012(04) : 12-14

[6] Wei Jiamin, Verilog programming art [M]. Electronic industry press, 2014

[7] Yu Deyu. Design of A/D data acquisition system based on FPGA [D]. Heilongjiang university 2012

[8] Zhang Chunxia, Zhou Chunmei, Dong Wenjie. A study on embedded fault injection test system combined with hardware and software [J]. Nuclear power technology and detection technology. 2013(05):149-153

[9] Li Ming, Ma yue, Yin zhenyu. Design of embedded security PLC based on ARM+FPGA [J]. Computer system application. 2017(03):99-100

[10] Wu Jihua, Cui Cheng. Altera FPGA/CPLD design (basic) [M]. Beijing: people's post and telecommunications press, 2005.