基于 Mailbox 的图像误差扩散并行算法的 SOPC 实现

郑升华 蒋本珊
（北京理工大学计算机科学技术学院　北京 100081）

摘要

本文介绍了一种图像误差扩散并行算法的原理，在 Altera 公司的 cyclone II 系列 FPGA 上利用 NIOS II 软核处理器搭建了一个以 Mailbox 为基础的 SOPC 系统，并且在这个多核系统上实现了图像误差扩散并行算法，提高了图像误差扩散算法对图像数据的处理速度，优化了系统的性能。

关键词

图像误差扩散并行算法；NIOS II 软核处理器；多核；SOPC

The image error diffusion parallel algorithms based on Mailbox using SOPC technology

Zheng sheng-hua，Jiang ben-shan
(Institute of Computer Science and Technology, Beijing Institute of Technology, Beijing, 100081, China)

Abstract

In this paper, the image error diffusion parallel algorithms is narrated briefly, and then using the NIOS II soft-core processors to built a Mailbox-based SOPC system with the Altera cyclone II FPGA, on this multi-core system, we carry out the image error diffusion parallel algorithms successfully, increasing the speed that deal with the image data using error diffusion algorithms, optimizing the performance of the system.

Key word

Image error diffusion parallel algorithms; NIOS II soft-core processor; multi-core; SOPC

1.引言

在日常的图形图像的转换中，将多阶灰度的图像转变为 1 阶图像的工作比较频繁。一般来说，为了将图像的细节显示得更为清楚，图像的灰度等级越高越好，但是在部分只能处理 1 阶图像信息的设备上，就必须将多阶灰度的图片资源转变为只有 1 阶信息的图片，这样才能够正确显示。

2.算法原理

这里将用 8 阶灰度图像转换为 1 阶图像作为算法实现的例子。8 阶显示图像的图形信息范围为[0，255]，一般认为对这样的数值范围取中间值即 127 作为量化的阀值，小于 127 的像素点值统一量化为 0，而大于 127 的像素点值统一量化为 1，从而将整个 8 阶图像的图像矩阵信息量化为只有 0，1 两种数值的矩阵。

但是这种量化方法很粗糙，目前流行的转化方法是 1975 年由 Floyd 和 Steinberg 提出的图像误差扩散算法，可以实现在有限色彩显示范围的设备上显示连续色彩范围的数字图像[1, 3]。

其处理图像误差的流程分为三步：

（1）量化。首先必须将 8 阶图像数据量化为 1 阶图像数据。这时候将所有处于[0，127]范围内的像素值量化为 0，将所有处于[128，255]范围内的像素值量化为 1。量化后的值称为量化值。

（2）误差计算。这是误差扩散算法的核心所在，要进行误差计算，首先必须将量纲统一。8 阶图像的显示数值量纲和 1 阶图像的显示数值量纲不同，必须将它们统一化。这里采取的方法是：在量化过程中量化值为 0 的像素点，对应的 8 阶数值量为 0，而量化值为 1 的像素点，对应的 8 阶数值量为 255。误差就是原来的 8 阶像素值和量化值对应的 8 阶量值的差。

（3）最后，将误差值分布到区域中相邻的像素上去[1], [3]。如图 1 所示：

图 1 采用的就是 Floyd-Steinberg 误差扩散算法将误差传播到相邻的像素上。7/16 的误差被加到正在处理的当前像素右侧的像素上，5/16 的误差被加到下一行中当前像素正下方的像素上。其余的误差以同样的方式被加到其他的相邻的像素上。图像中所有的像素都要通过上述误差扩散算法的处理。

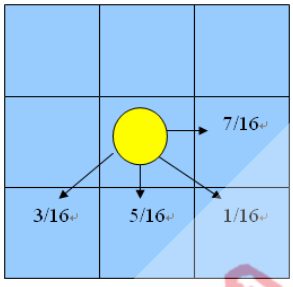

图 1 误差扩散算法示意图

误差扩散算法看上去本质上是一个典型的串行算法：在误差被计算出来之后就将其分布到相邻的像素上去，因此要计算下一个像素的值就必须知道前面元素的误差值，这种相关性暗示了程序在一个时刻只能处理一个像素。但是可以进行另一个角度的思考：一个被扩散误差的数据，只要它所接受的四个误差值都准备好了，就马上可以开始。图 2 是从接受像素的观点来查看误差扩散算法的[2]。

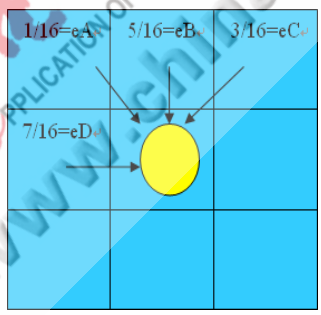

图 2 从接受像素的角度来观察误差扩散算法

有了这样的基础，就可以将这种原来是串行计算的算法转换为可以并行执行的算法。如果在系统中存在着两个处理任务，当一个任务处理数据完全产生了对下一行某个数据的三个误差值时，另一个任务就可以马上处理相应的数据了。这时，在系统里，同时有两个任务在进行数据的处理，实现了误差数据的并行操作。

3.基于Mailbox的双NIOS实现方案

NIOS II 是 Altera 特有的基于通用 FPGA 构架的软 CPU 内核，是一个可灵活配置的通用 32 位 RISC 嵌入式处理器。NIOS II 处理器内核有 3 种类型，用来满足不同的设计要求。它们分别是快速型（NIOS II/f，Fast）、经济型（NIOS II/e，Economy）和标准型（NIOS II/s，Standard）。快速型 NIOS II 内核具有最高的性能，经济型 NIOS II 内核具有最低的资源占用，而标准型 NIOS II 在性能和面积之间做了一个平衡。NIOS II 具有指令和数据缓冲，高达 2GB 的外部地址空间，支持流水线和动态/静态分支预测，实现了硬件的乘除法，具有强大的功能。NIOS II 处理器的指令架构十分适合嵌入式软件的编译，其具有灵活

的寻址方式、快速的中断处理和对内部存储器文件的高效使用。另外它对数值计算进行了模块化的加速，特有的 MSTEP 指令和 MUL 指令提高了整数乘法运算的计算效率[4]。

NIOS II 软核处理器具有高灵活性，丰富的可配置外设，存储器和接口，高速的数据处理能力等特点。

1. 高灵活性：使用 NIOS II 软核处理器，可以在 SOPC Builder 中添加自定义的外设，实现随时根据需要添加外设的能力。
2. 高速的数据处理能力：NIOS II/s（标准）拥有 5 级流水线，动态支路预测，可设置指令及数据缓冲，动态支路预测等功能来提升性能。NIOS II 处理器定制指令扩展了 CPU 指令集，采用用户自定义指令可以实现传统处理器无法达到的最佳系统性能。NIOS II 系列处理器支持多达 256 条的用户自定义指令，可以使用硬件方式来对通常由软件实现的逻辑和复杂数学算法进行加速。
3. Avalon 交换架构：Avalon 交换架构的同时多主机体系结构提高了系统带宽，消除了带宽瓶颈。每当系统加入模块或者外设接入优先权改变时，SOPC Builder 利用最少的 FPGA 资源，产生新的最佳 Avalon 交换架构。Avalon 交换架构支持多种系统体系结构，如单处理器/多处理器系统，可实现数据在外设与性能最佳数据通道之间的无缝传输[4]。

Altera 公司为构建基于 NIOS II 多核系统提供了两个类型的组件，一个是 Mutex，一个是 Mailbox。Mutex 被用于保证共享资源的互斥使用，而 Mailbox 用于多个处理器之间的消息通讯。在进行多处理器之间消息通讯和共享时，由于 Mutex 只能在同一时刻使一个且仅有一个处理器应用处于活动状态，Mailbox 能够保证在消息有效时，系统中有多个处理器应用处于活动状态，因而使用 Mailbox 的效率会比 Mutex 高。在这次设计中，双核系统的通讯工具就采用了 Mailbox。图 3 所示为基于 Mailbox 的 NIOS II 多软核处理器系统的硬件设计框图。

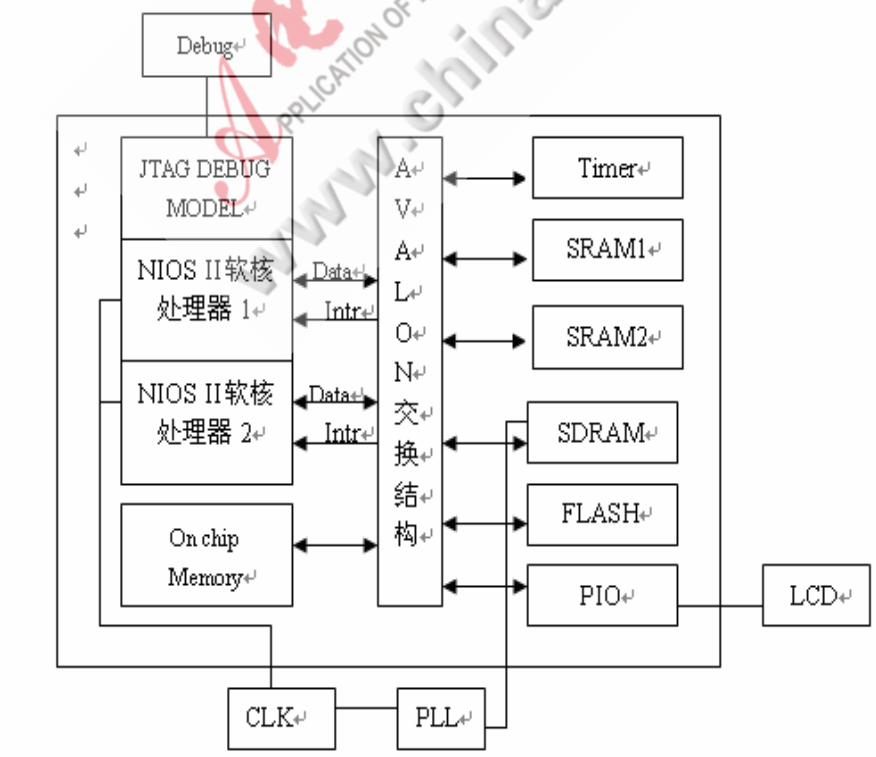

图 3 基于 Mailbox 的多 NIOS 系统设计框图

根据并行化误差扩散算法的特点，采用了 Mailbox 作为传递上一行处理数据时产生的

误差值向下一行传递的工具。系统中存在两个 NIOS II 软核处理器，其中一个存储有原来 8 阶图像数值矩阵的所有奇数行数据，另外一个存储原来 8 阶图像数值矩阵的所有偶数行数据，双方通过两个 Mailbox 进行数据通讯。其中一个 Mailbox 传递奇数行产生的误差值给存储有偶数行数据值的 NIOS II 软核处理器，这个 Mailbox 被命名为 even mailbox，使用系统中的 SRAM1 存储器作为存储介质，另外的一个 Mailbox 传递偶数行产生的误差值给存储有奇数行数据值的 NIOS II 软核处理器，这个 Mailbox 被命名为 odd mailbox，使用系统中的 SRAM2 存储器作为存储介质。双 NIOS II 并行工作的模式如图 4 所示。

系统的工作流程分为下面的几步：

（1）NIOS 处理器 1 顺序扫描第一个奇数行的数据，处理第一个奇数行里面的每一个像素，将产生的误差值送进 even mailbox；

（2）NIOS 处理器 2 检查 even mailbox，发现数据立即将数据放入误差数值矩阵中，启动扫描误差数值矩阵的循环，检查是否有某个偶数行像素点所需要的误差全部准备好的。如果有，进行相应偶数行像素点的处理，这时偶数行像素点也会产生相应的误差值，通过 odd mailbox 向 NIOS 处理器 1 发送；

（3）NIOS 处理器 1 检查 odd mailbox，发现数据立即将数据放入误差数值矩阵中，启动扫描误差数值矩阵的循环，检查是否有某个奇数行像素点所需要的误差全部准备好的。如

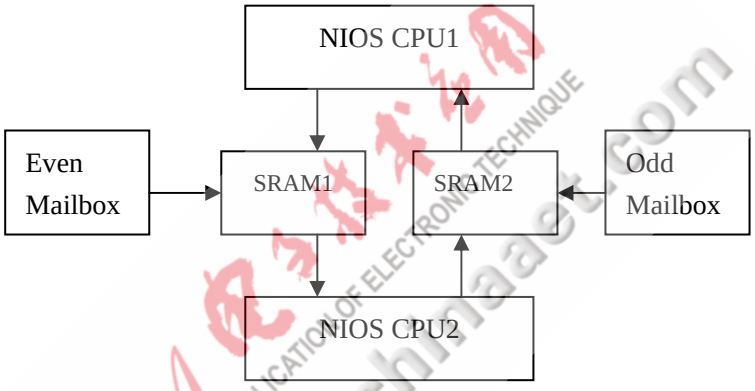

图 4 双 NIOS 并行工作原理

果有，进行相应奇数行像素点的处理，这时奇数行像素点也会产生相应的误差值，通过 even mailbox 向 NIOS 处理器 2 发送。

这样进行循环处理，就可以将所有的数据处理完毕。下面说明扫描误差数值矩阵的循环是如何发现某个元素的误差值已经全部准备好的。图 5 显示某行数据处理后产生的误差值的顺序。

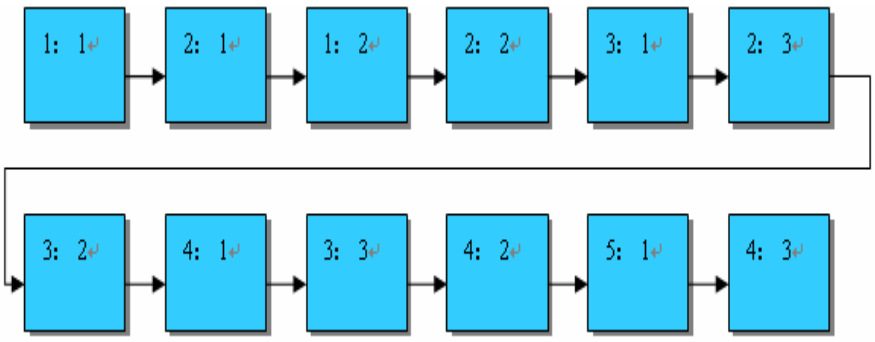

图 5 误差产生的顺序

一行数行产生的 3 个对下一行有影响的误差值（其中首个元素只产生 2 个），数据框中的第一个 1 说明产生的这个误差值是对下一行的第一个元素起作用，相应的如果是 2 就

是对下一行的第二个元素起作用，冒号后面的数字表示产生的数值具体对应与哪一个误差因子，如果是 1，对应的误差因子就是 3/16，如果是 2，对应的误差因子就是 5/16；如果是 3，对应的误差因子就是 1/16。图 5 是完全按照误差产生的数据顺序来排列的。下一行的相应的数据元素如果 3 个误差扩散因子都产生了的话，NIOS II 软核处理器就可以进行数据处理了。根据观察，我们可以发现，每当产生 3 个误差数据的时候，下一行的一个数据元素一定有一个可以进行处理------这是我们发现误差数值准备好的基础。

因此，mailbox 传递信息的数据结构如下设置：

```
typedef struct {
                        int flag-i;   //指明放入误差矩阵的行
                        int flag-j;   //指明放入误差矩阵的列
                        int data;     //用于存放误差数据
} share_struct
```

而误差矩阵初始化所有的元素值为 1000，处理完毕后将对应的矩阵位置的元素设置为 2000。这两个数值在 8 阶转换为 1 阶的过程中都是非法值。误差矩阵的行号指明这一行的误差值是对于哪一行有效。

下面是接收数据函数的部分代码：

```
Receiver( )                                          //接收误差值任务
 { Share=altera_avalon_mailbox_open("/dev/evenmailbox");
   i=altera_avalon_mailbox_pend(Share->flag-i);   //取出行顺序值
   j=altera_avalon_mailbox_pend(Share->flag-j);   //取出列顺序值
   value=altera_avalon_mailbox_post(Share->data); //取出误差值
   ErrorData[i][j]=value;}                        //按照顺序存放误差值
```

下面是接收误差数据后处理图像元素的部分代码：

```
for(i=0;i<Length;i++)
  for(j=3;j<Length1;j+3)
   { k=j/3;
     if(ErrorData[i][j-1]!=1000&&ErrorData[i][j-2]!=1000&&rrorData[i][j-4]!
=1000&&ErrorData[i][j-1]!=2000&&ErrorData[i][j-2]!=2000&&ErrorData[i][j-4]!
=2000)
      { Data[k]=Data[k]+DataError[i][j-1]+DataError[i][j-2]+DataError[i][j-4]
        if(Data[k]<128)
          { Tem=data[k];     //放入相应的临时变量
            data[k]=0;
            Error0=Tem*3/16;//产生误差因子为3/16的误差值
            Error1=Tem*5/16;//产生误差因子为5/16的误差值
            Error2=Tem*1/16;//产生误差因子为1/16的误差值
            Error3=Tem*7/16;//产生误差因子为7/16的误差值
            data[i][k+1]+=Error3;
            Share=altera_avalon_mailbox_open("/dev/evenmailbox");
            altera_avalon_mailbox_post(Share->flag-i,i+1);
            altera_avalon_mailbox_post(Share->flag-j,2+(j-1)*3);
            altera_avalon_mailbox_post(Share->data,Error0);
            altera_avalon_mailbox_post(Share->flag-i,i+1);
            altera_avalon_mailbox_post(Share->flag,2+(j-1)*3+1);
            altera_avalon_mailbox_post(Share->data,Error1);
            altera_avalon_mailbox_post(Share->flag-i,i+1);
            altera_avalon_mailbox_post(Share->flag,2+(j-1)*3+1+1);
            altera_avalon_mailbox_post(Share->data,Error2);}
      }
```

数据处理的流程如图 6 所示：

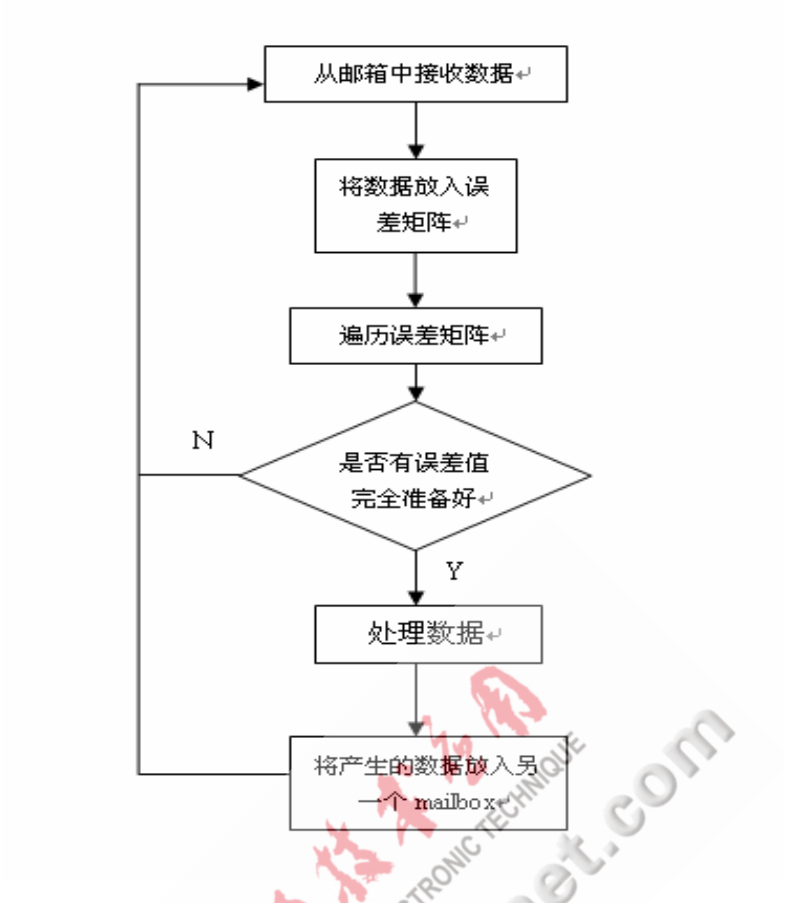

图 6 数据处理流程

4.系统特点

基于 Mailbox 的 NIOS II 双软核系统，能够很好的利用 Mailbox 传输数据速度快、特别适合处理器任务为生产者/消费者的特点，对图像误差扩散算法进行了并行化的设计，实现了预期的目的。由于在系统运行时，有两个处理器任务处于活动状态，从而在硬件和软件两个层面上实现了多核并行。

5.结语

随着嵌入式应用飞速发展，嵌入式系统面临的处理任务和处理数据也日益庞大，传统的以单个处理器为核心的系统已经不能满足应用的需求，需要进行多核系统开发。而使用硬核进行多核系统构建，有着更新难、可配置性能低、价格昂贵的缺点，使用 NIOS II 软核处理器和 SOPC 技术，能够在性能和价格取得平衡，使多核系统更有效，更具灵活性。但是我们可以看到，虽然多核的硬件设计比较容易实现，但是软件的设计模式需要改变，以适应多核的运行环境。在未来的嵌入式软件设计中，并行化将会是一个重要的主题。

参考文献

[1] R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial grey scale. Proc. Soc. Inf. Display, 17:75-77, 1976

[2] Shameem Akhter，Jason Roberts。Multi-Core programming[M]。USA：Intel press，2007

[3] R. W. Floyd and L. Steinberg, "An adaptive algorithm for spatial grey scale," in Proc. Dig.SID International Symp., pp. 36–37, Los Angeles, California (1975).

[4] 王建校，危建国。SOPC 设计基础与实践[M]。西安：西安电子科技大学出版社，2006

[5] 张志刚。FPGA 与 SOPC 设计教程—DE2 实践[M]。西安：西安电子科技大学出版社，2007

作者简介：
郑升华（1984－），男，北京理工大学计算机科学技术学院，硕士生，计算机系统结构。
蒋本珊（1955－），女，北京理工大学计算机科学技术学院副教授，硕士，计算机应用技术、计算机系统结构
Biography:
Zheng shenghua. Male, Institute of Computer Science and Technology, Beijing Institute of Technology, master, computer architecture
Jiang Ben-shan, Female, Institute of Computer Science and Technology, Beijing Institute of Technology, master, computer application, computer architecture

联系人：蒋本珊，
单位：北京理工大学计算机科学技术学院（100081），
电话：13520785629
电子邮箱：bs.jiang@163.com

研究成果声明

本人郑重声明：所提交的论文是我本人在指导教师的指导下进行的研究工作获得的研究成果。尽我所知，文中除特别标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果。

签　　名：郑升华　　日期：2008 年 6 月 19 日

导师签名：蒋本珊　　日期：2008 年 6 月 19 日