



東南大學

毕业设计(论文)

外文资料翻译

翻译资料名称(外文) Accelerating colour space conversion

翻译资料名称(中文) 色彩空间转换的可重构硬件实现

院（系）： 仪器科学与工程学院

专 业： 测控技术与仪器

姓 名： 戴天宇

学 号： 22011229

指导教师： 王立辉

完成日期： 2015 年 3 月 20 日

色彩空间转换的可重构硬件实现

作者: F.Bensaali*,A.Amira

英国贝尔法斯特女王大学, 计算机科学院。

接受于 2004.06.27; 修订版本在 2005.03.25 收到; 在 2005.03.30 被接收。

概要

色彩空间转换对于包括视频压缩在内的许多种图像处理应用而言都是十分重要的。这一操作在高度优化的解码器中的功耗比例接近于 40%。这也就是说, 一种能够有效实行这种转换的方法是为我们所期望的。这篇论文针对色彩空间转换的有效实行提出了两个新的结构, 它们都适用于现场和编程逻辑阵列 (FPGA) 和 VLSI 的实现。这些结构都是基于分布式算法 (DA) ROM 累加器原理, 并且已经被完成, 还在 Celoxica RC1000 FPGA 开发板上完成了验证。此外, 它们是平台无关的, 同时拥有着低延迟 (八个周期)。第一个结构拥有高度的吞吐量, 第二个则是完全流水化的, 同时拥有一个能够将数据速率持续保持在 234 百万次转换每秒的吞吐量。

2005 Elsevier 有限公司拥有所有权利。

关键词: 色彩空间转换; 现场可编程逻辑阵列; 分布式算法

1. 前言

色彩是视网膜上的可见区域对于光所引起的频谱变化的一种视觉感受。由于人类的视觉系统拥有三种类型的视锥细胞, 所以对于描述色彩而言, 三个不同的成分是充分并且必要的[1]。

色彩空间 (也被称为色彩模式或者色彩系统) 是一种我们可以用于定义、创造和构想颜色的方法。现在已经有许多色彩空间存在, 并且其中的绝大多数都是以每一种颜色作为一个三维坐标系统的一个点。每种色彩空间都是为了已经定义好的应用领域而优化的[3]。其中三个最流行的色彩模式是 RGB (用于色彩打印领域)。所有的色彩空间都能够由被现实摄像头、扫描仪这样的硬件所支持的 RGB 信息派生出来。

在 RGB 色彩空间内——一个由若干 RGB 值表示的像素点构成的集合内, 对图像进行处理并不是最高效的。为了加速许多处理过程, 许多广播、视频和图像使用亮度和色差信号标准, 比如 YCrCb, 这使得在两种制式之间必须存在一个转换机制。一些可以实现 RGB 到 YCrCb 转换的核已经能够在市面上找到, 就像是 Amphion Ltd[4], CAST.Inc[5] 和 ALMA.Tech[6] 建议的那样, 它们被设计为 FPGA 实现。另一种转换器也被提出[7], 它已经被实现在由一个带有基于 FPGA 的可重构功能单元的 TriMedia 处理器实现的混合系统中。

作为在贝尔法斯特皇后大学正在进行的一个课题项目的一部分——针对于图像和信号处理算法, 去开发一个基于矩阵运算的硬件加速器[8-11]。本文提出了基于 DA 流水线, 并且使用 FPGA 进行 RGB 和 YCrCb 色彩空间转换的、一个低成本的加速器, DA 流水线是去隐藏乘法的一个乘法运算的比特级重排。这两个建议的结构都基于一些像素的串行和并行操作。

实现和验证建议的结构的目标硬件是配有 Xilinx 公司 XCV2000E Virtex FPGA[12,13]的 Celoxica RC1000 PCI 开发板。本文其他部分的组成如下: 第二节给出了从 RGB 到 YCrCb 的一个综述, 第三节和第四节则是两种结构所涉及的数学背景和描述。硬件实现的结果和分析则是被列于第五和第六节。最终的结束语在第六节被给出。

2. 色彩空间转换: 一个综述

在前言中许多色彩模式都被提到了, 每一种都倾向于支持一种特定的任务或者解决某个针对性的问题。下面被描述的是两种被选择于我们研究的对象, 它们被用于许多图像处理应用中。

2.1. RGB 色彩空间

RGB 色彩空间是一种简单和健壮的色彩定义形式。RGB 使用三个数字部分去描述一种色彩。这种色彩空间能够被认为是一种三维空间坐标系统, 坐标系的每个轴对应色彩的每一种成分, R 或者红色, G 或者绿色, B 或者蓝色。RGB 是一种适用于电脑显示的色彩空间, 它的对应非常接近于人类眼睛的行为[1]。RGB 也是一种加

色系统，三种基本的色彩“红”、“绿”、“蓝”被叠加在一起去组成期望的色彩。对于真彩色，每个像素的红绿蓝三个部分都需要八位的位宽。总体来看，大概需要 16 百万（ 2^{24} ）可能的颜色，而每个部分的范围都在 0-255 之间，当每个部分都是 0 的时候为黑，都为 255 的时候为白[1]。在本文其他的部分，被伽马校正过的 RGB 值被标记为 R'G'B'。

2.2. Y'CrCb 色彩空间

T'CrCb 色彩空间是 TUV 色彩空间的一种偏移和缩放，它基于亮度和色度，这对应于明度和色彩。在 R'G'B'色彩空间内就是分离到亮度部分（Y'）和两个色度部分（Cr 和 Cb）。Y'被定义在范围 16-235 内，Cb 和 Cr 的范围则是 16-240[1,2]。

2.3. 从 R'G'B'转换到 Y'CrCb

分解一个 R'G'B'色彩空间的图像到亮度图像以及两个色度图像是许多商业应用中最常用的方法[14,15]，比如人脸检测，以及 JPEG 和 MPEG 图像标准[16-18]。

估算来讲，从 R'G'B'合成转换到 Y'CrCb 合成的功耗占据了高度优化解码器的 40%[16]。加速这个操作对于整个运算的加速将会十分有用。将一个 R'G'B'色彩空间的颜色转换到 Y'CrCb 色彩空间会用到以下等式：

$$\begin{pmatrix} Y' \\ Cr \\ Cb \end{pmatrix} = \begin{pmatrix} 0.257 & 0.504 & 0.098 & 16 \\ 0.439 & -0.368 & -0.071 & 128 \\ -0.148 & -0.291 & 0.439 & 128 \end{pmatrix} \begin{pmatrix} R' \\ G' \\ B' \\ 1 \end{pmatrix} \quad (1)$$

随后其反变换可以从以下等式得出：

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{pmatrix} 1.164 & 1.596 & 0.0 & -222.912 \\ 1.164 & -0.813 & -0.392 & 135.616 \\ 1.164 & 0.0 & 2.017 & -276.8 \end{pmatrix} \begin{pmatrix} Y' \\ Cr \\ Cb \\ 1 \end{pmatrix} \quad (2)$$

3. 基于串行处理方法的结

3.1 数学背景

自从色彩空间转换可以被表述为一个矩阵向量（MV）操作，一种基于 DA 的新算法在本节中本提出。

DA，分布式算术运算而不是将它们分组相乘。常规的 DA 被称为基于 ROM 的 DA，通过分解内积的可见输入变量到比特级，去生成预处理的数据。基于 ROM 的 DA 使用一个查找表去存储预处理数据，这种方法能够在 VLSI 实现中使得芯片面积被完全和有效的使用。基于 DA 的 ROM 方法的优势在于它的实现效率。基本的需求是一个 ROM 序列、加法、减法和对于输入数据序列的移位操作[19]。一个 DA 用法的例子可以在这些参考文献中被找到[19-21]。

考虑以下等式的矩阵向量积：

$$C_i = \sum_{k=0}^{N-1} A_{ik} B_k \quad (3)$$

公式中的{A_{ik}}’s 是一个 Lbits 的常量，{B_k}’s 则是被写在一个无符号二进制数表达式，如式中所见。

$$B_k = \sum_{m=0}^{W-1} b_{k,m} 2^m \quad (4)$$

式中 $b_{k,l}$ 表示 B_k 的第 m 位，0 或者 1， W 表示每个像素的颜色分量解析度的位宽。

将 (4) 带入 (3)：

$$C_i \sum_{k=0}^{N-1} A_{ik} \left(\sum_{m=0}^{W-1} b_{k,m} 2^m \right) = \sum_{m=0}^{W-1} \left(\sum_{k=0}^{N-1} A_{ik} (b_{k,m} 2^m) \right) \quad (5)$$

定义：

$$Z_m = \sum_{k=0}^{N-1} A_{ik} b_{k,m} \quad (6)$$

从而 C_i 可以用一下公式计算：

$$C_i = \sum_{m=0}^{W-1} Z_m 2^m \quad (7)$$

这种想法的可行性来自于到元素 Z_m 依赖于 k,m 并且仅仅有 2^N 个可能的值，这保证了一个 ROM 是有可能对它们进行预处理和存储操作的。ROM 的内容不同且依赖于一个静态的系数矩阵 A 。这些中间结果在 W 个时钟周期被积累，并且生成 C_i 系数矩阵。

3.2 个案研究：在 $R'G'B$ 和 $Y'CrCb$ 之间转换

CSC 核利用以下数学公式去将一个空间转换为另一个：

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} A_{00} & A_{01} & A_{02} & A_{03} \\ A_{10} & A_{11} & A_{12} & A_{13} \\ A_{20} & A_{21} & A_{22} & A_{23} \end{pmatrix} \begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ 1 \end{pmatrix} \quad (8)$$

公式中的 C_i ($0 \leq i \leq 2$)， B_i ($0 \leq i \leq 3$) 顺序地表示输入和输出色彩组成。由于所有的组成部分都拥有 0-255 这个范围，只需要八位的位宽便可以满足要求。在我们的应用中 ($N=4$ 且 $W=8$)， C_i 能够被以下公式计算出来：

$$C_i = \sum_{m=0}^7 Z_m 2^m \quad (9)$$

其中：

$$Z_m = \sum_{k=0}^3 A_{ik} b_{k,m} \quad (10)$$

$$C_i = \begin{array}{ccccccc} & k=0 & k=1 & k=2 & k=3 & & \\ & (A_{i0} \times b_{0,0} + & A_{i1} \times b_{1,0} + & A_{i2} \times b_{2,0} + & A_{i3} \times b_{3,0}) & \times 2^0 + & m=0 \\ & (A_{i0} \times b_{0,1} + & A_{i1} \times b_{1,1} + & A_{i2} \times b_{2,1} + & A_{i3} \times b_{3,1}) & \times 2^1 + & m=1 \\ & (A_{i0} \times b_{0,2} + & A_{i1} \times b_{1,2} + & A_{i2} \times b_{2,2} + & A_{i3} \times b_{3,2}) & \times 2^2 + & m=2 \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & (A_{i0} \times b_{0,7} + & A_{i1} \times b_{1,7} + & A_{i2} \times b_{2,7} + & A_{i3} \times b_{3,7}) & \times 2^7 + & m=7 \end{array} \quad (11)$$

需要三个位宽为 $2^N=24=16$ 的 ROM（一个被用于每一个 A 矩阵的行）去存储预处理 2^4 个可能的部分乘积值。由于 B 向量的最后一个元素等于 1:

$$b_{3,m} = \begin{cases} 1 & \text{for } m = 0 \\ 0 & \text{for } m \neq 0 \end{cases} \quad (12)$$

Eq. (9) can be rewritten as

$$C_i = \sum_{m=0}^7 Z_i^* 2^m + A_{i3} \quad (13)$$

where

$$Z_m^* = \sum_{k=0}^2 A_{ik} b_{k,m} \quad (14)$$

$$C_i = \begin{array}{ccccccc} & k=0 & k=1 & k=2 & & & \\ & (A_{i0} \times b_{0,0} + & A_{i1} \times b_{1,0} + & A_{i2} \times b_{2,0}) & \times 2^0 + & m=0 \\ & (A_{i0} \times b_{0,1} + & A_{i1} \times b_{1,1} + & A_{i2} \times b_{2,1}) & \times 2^1 + & m=1 \\ & (A_{i0} \times b_{0,2} + & A_{i1} \times b_{1,2} + & A_{i2} \times b_{2,2}) & \times 2^2 + & m=2 \\ & \vdots & \vdots & \vdots & \vdots & \vdots \\ & A_{i0} \times b_{0,7} & A_{i1} \times b_{1,7} & A_{i2} \times b_{2,7} & \times 2^7 + & m=7 \\ & A_{i3} & & & & \end{array} \quad (15)$$

这是一个非常有价值的行为，它使得 ROM 的大小被削减到了 2^3 ，表 q 给出了每个 ROM 的内容：

Table 1
Content of the ROM i ($0 \leq i \leq 2$)

$b_{0,m}$	$b_{1,m}$	$b_{2,m}$	The content of the ROM i
0	0	0	0
0	0	1	A_{i2}
0	1	0	A_{i1}
0	1	1	$A_{i1} + A_{i2}$
1	0	0	A_{i0}
1	0	1	$A_{i0} + A_{i2}$
1	1	0	$A_{i0} + A_{i1}$
1	1	1	$A_{i0} + A_{i1} + A_{i2}$

3.3 建议的结构

由于我们的目标是实现一个可以执行两个不同色彩空间转换（R'G'B'与 Y'CrCb），所以需要六个 ROM（每个转换需要三个），图表 2 和 3 给展示了建议的核的引脚和内部结构：

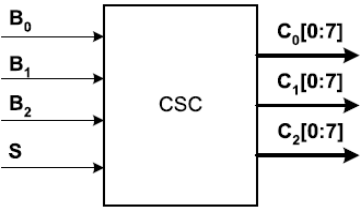


Fig. 2. Symbol of the CSC core.

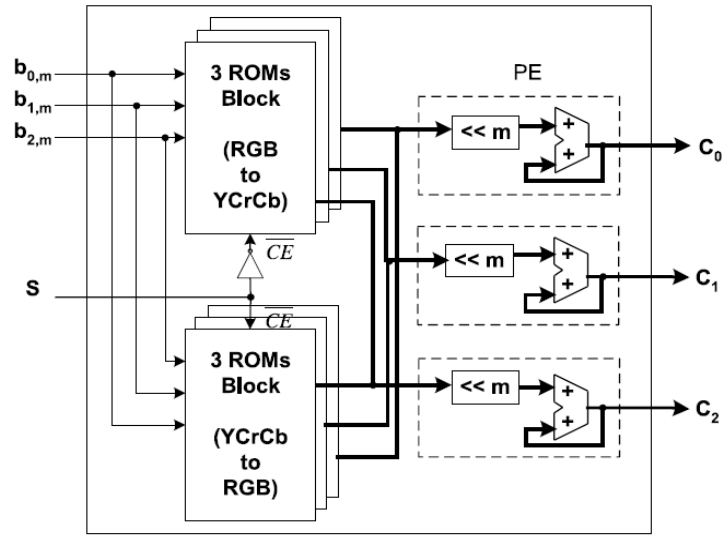


Fig. 3. Serial CSC based DA architecture.

引脚的解释在表 2 中：

Table 2
Pins description

Name	Dir	Description
B_0	I	First input colour space component
B_1	I	Second input colour space component
B_2	I	Third input colour space component
C_0	O	First output colour space component
C_1	O	Second output colour space component
C_2	O	Third output colour space component
S	I	Colour space conversion type selection

建议的结构有三个独立的运算单元（PEs）和两个存储器块构成。每个 PE 包括一个并行累加器（ACC）和一个右向移位寄存器，每个存储器块由三个大小为 2^3 的 ROM 构成（见图表 4）。ROM 的内容是不同且依赖于系数矩阵 A 的，A 的系数则依赖于转换类型。

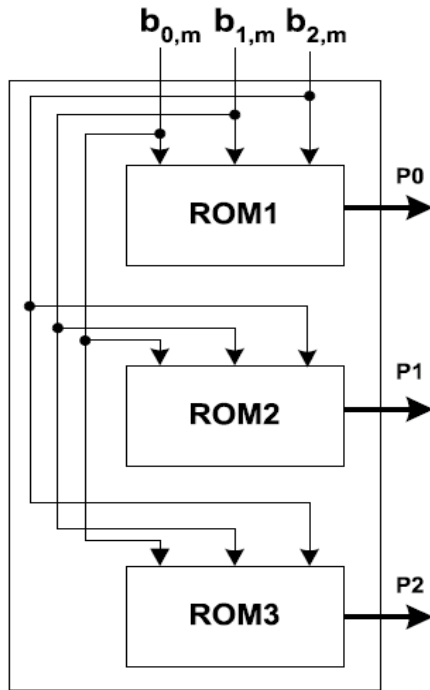


Fig. 4. Memory block structure.

我们的结构是可扩展的，这十分有价值，例如，如果我们想执行 n 转换，我们只需要添加每一次 $3 \times n$ 个 ROM，保证能够存储转换技术矩阵即可，而 PEs 则是不用更改的。在被建议的结构中，一个 $N \times M$ 的图像在设置输入为 $R'G'B$ 后，每八个周期便可以转换完毕，这是一个由新的像素构成的集合（为了逆变换的 $Y'CrCb$ ）见图表

5。

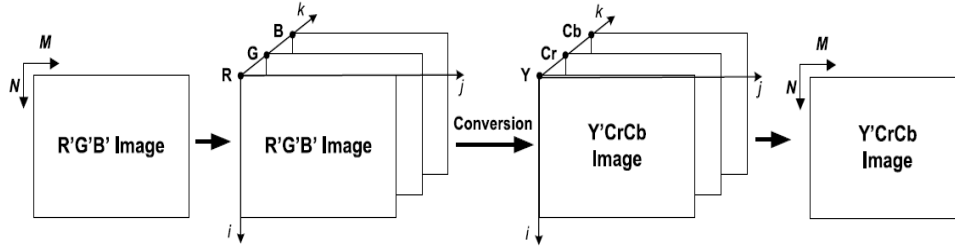


Fig. 5. R'G'B' to Y'CrCb conversion.

4. 基于并行处理方法的结构

4.1 数学背景

考虑一个 $N \times M$ 的图像（N:图像高度，M:图像宽度）。

使用 b_{ijk} 来表示每一个像素（ $0 \leq i \leq N-1, 0 \leq j \leq M-1, 0 \leq k \leq 2$ ），如下：

$$\begin{cases} b_{ij0} = R_{ij} / \text{the red component of the pixel in row } i \\ \text{and column } j \\ b_{ij1} = G_{ij} / \text{the green component of the pixel in row } i \\ \text{and column } j \\ b_{ij2} = B_{ij} / \text{the blue component of the pixel in row } i \\ \text{and column } j \end{cases} \quad (16)$$

图像可以用一下数学公式被转换：

$$\begin{pmatrix} \begin{pmatrix} c_{000} \\ c_{001} \\ c_{002} \end{pmatrix} & \cdots & \begin{pmatrix} c_{0(M-1)0} \\ c_{0(M-1)1} \\ c_{0(M-1)2} \end{pmatrix} \\ \begin{pmatrix} c_{100} \\ c_{101} \\ c_{102} \end{pmatrix} & \cdots & \begin{pmatrix} c_{1(M-1)0} \\ c_{1(M-1)1} \\ c_{1(M-1)2} \end{pmatrix} \\ \vdots & \cdots & \vdots \\ \begin{pmatrix} c_{(N-1)00} \\ c_{(N-1)01} \\ c_{(N-1)02} \end{pmatrix} & \cdots & \begin{pmatrix} c_{(N-1)(M-1)0} \\ c_{(N-1)(M-1)1} \\ c_{(N-1)(M-1)2} \end{pmatrix} \end{pmatrix} \\
= \begin{pmatrix} A_{00} & A_{01} & A_{02} & A_{03} \\ A_{10} & A_{11} & A_{12} & A_{13} \\ A_{20} & A_{21} & A_{22} & A_{23} \end{pmatrix} \\
\otimes \begin{pmatrix} \begin{pmatrix} b_{000} \\ b_{001} \\ b_{002} \\ 1 \end{pmatrix} & \cdots & \begin{pmatrix} b_{0(M-1)0} \\ b_{0(M-1)1} \\ b_{0(M-1)2} \\ 1 \end{pmatrix} \\ \begin{pmatrix} b_{100} \\ b_{101} \\ b_{102} \\ 1 \end{pmatrix} & \cdots & \begin{pmatrix} b_{1(M-1)0} \\ b_{1(M-1)1} \\ b_{1(M-1)2} \\ 1 \end{pmatrix} \\ \vdots & \cdots & \vdots \\ \begin{pmatrix} b_{(N-1)00} \\ b_{(N-1)01} \\ b_{(N-1)02} \\ 1 \end{pmatrix} & \cdots & \begin{pmatrix} b_{(N-1)(M-1)0} \\ b_{(N-1)(M-1)1} \\ b_{(N-1)(M-1)2} \\ 1 \end{pmatrix} \end{pmatrix}
\end{pmatrix}$$

公式中的符号 \widehat{x} 被做了以下定义：

$$\begin{pmatrix} c_{ij0} \\ c_{ij1} \\ c_{ij2} \end{pmatrix}$$

这是以下乘积的结果：

$$\begin{pmatrix} A_{00} & A_{01} & A_{02} & A_{03} \\ A_{10} & A_{11} & A_{12} & A_{13} \\ A_{20} & A_{21} & A_{22} & A_{23} \end{pmatrix} \begin{pmatrix} b_{ij0} \\ b_{ij1} \\ b_{ij2} \\ 1 \end{pmatrix},$$

其中 c_{ijk} 表示输出图像的色彩空间构成，同时：

$$A = \begin{pmatrix} A_{00} & A_{01} & A_{02} & A_{03} \\ A_{10} & A_{11} & A_{12} & A_{13} \\ A_{20} & A_{21} & a_{22} & A_{23} \end{pmatrix}$$

这表示了等式(1)和(2)的常数矩阵之一。

c_{ijk} 元素（输出图像的色彩空间构成）能够被下面的等式所计算：

$$c_{ijk} = \sum_{m=0}^3 A_{km} b_{ijm} \quad (0 \leq i \leq N-1, 0 \leq j \leq M-1, \\ 0 \leq k \leq 2) \quad (18)$$

其中 $\{A_{km}\}$'s 是 Lbits 的常数，而 $\{b_{ijm}\}$'s 被写在一个无符号二进制数表达式，如式中所见：

$$b_{ijm} = \sum_{l=0}^{W-1} b_{ijm,l} 2^l \quad (0 \leq i \leq N-1, 0 \leq j \leq M-1, \\ 0 \leq m \leq 2) \quad (19)$$

在章节 4.1 中使用同样的过程，等式(16)能够被重写为：

$$c_{ijk} = \sum_{l=0}^7 Z_l^* 2^l + A_{k3} \quad (20)$$

其中：

$$Z_l^* = \sum_{m=0}^2 A_{km} b_{ijm,l} \quad (21)$$

$$\begin{aligned}
& \begin{matrix} m=0 & m=1 & m=1 \\ (A_{k0} \times b_{ij0,0} + & A_{k1} \times b_{ij1,0} + & A_{k2} \times b_{ij2,0}) \times 2^0 + & l=0 \\ (A_{k0} \times b_{ij0,1} + & A_{k1} \times b_{ij1,1} + & A_{k2} \times b_{ij2,1}) \times 2^1 + & l=1 \\ c_{ijk} = (A_{k0} \times b_{ij0,2} + & A_{k1} \times b_{ij1,2} + & A_{k2} \times b_{ij2,2}) \times 2^2 + & l=2 \\ \vdots & \vdots & \vdots & \vdots \\ (A_{k0} \times b_{ij0,7} + & A_{k1} \times b_{ij1,7} + & A_{k2} \times b_{ij2,7} + & \times 2^7 + & l=7 \end{matrix} \quad (22) \\
& A_{k3} \\
& PP_{ijk0} + PP_{ijk1} + PP_{ijk2} + PP_{ijk3} + \\
& = PP_{ijk4} + PP_{ijk5} + PP_{ijk6} + PP_{ijk7} + \\
& A_{k3}
\end{aligned}$$

和第一种建议的结构一样，ROM 的内容不同且依赖于系数矩阵 A，A 矩阵依赖于转换类型。

4.2 建议的结构

等式 (18) 能够部署到建议的结构中，就像是图表 6 所表现的一样：

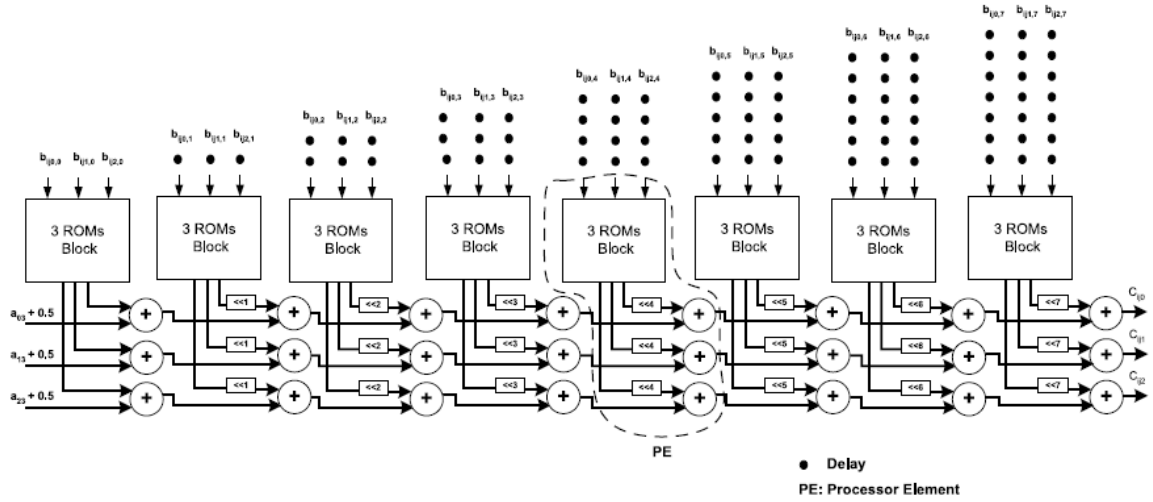


Fig. 6. Proposed parallel architecture based on DA principles clock cycles, where (3×4) is the constant matrix A size.

这个结构由八个完全一样的 PEns ($0 \leq n \leq 7$) 构成。每个 PEn 都包括三个并行有符号加法器，三个 n 位右移寄存器和一个 ROM 块，就像是图标 4 所述的结构那样。值得注意的是，这个结构拥有一个 W 的延迟和速率为 1 的吞吐量。整幅图像的转换能够在 (延迟 + (NM) 吞吐量) = 8 + (NM) 个时钟周期完成。在如图表 7 的标准算法下，转换可以在 (3x4xNxM) 个时钟周期完成，(3x4) 是静态矩阵 A 的大小。

```

for i ← 1 to L do           // scanning image rows
  for j ← 1 to M do         // scanning image columns
    for k ← 1 to 3 do        // scanning the three RGB value of a pixel
      for k ← 1 to 3 do      // scanning columns of the constant conversion matrix

         $c_{ijk} += a_{km} \times b_{ijm}$ 

      end for
    end for
  end for
end for

```

Fig. 7. Pseudo code for the standard algorithm.

图表 8 表明了建议结构的功能分析图。

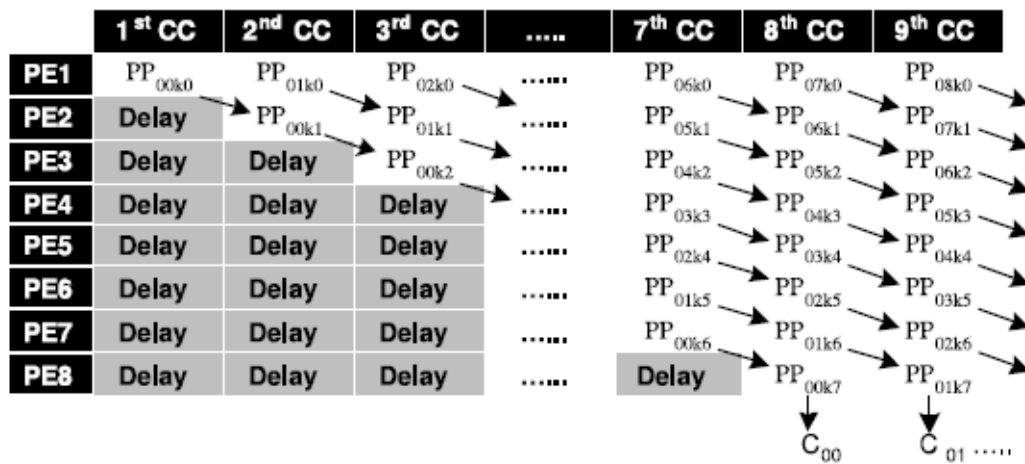


Fig. 8. Functional analysis diagram.

5. 硬件实现

两种建议的结构都是基于 DA 技术的，并且已经在 Celoxica RC1000-PP FPGA 开发板上被实现和验证了。

RC1000-PP 开发板惯用配备了 Virtex-E2000 FPGA 芯片（bg560 封装，速度等级为 6）的 PCI 标准总线卡。表 3 和表 4 顺序给出了一些用于 R'G'B'到 Y'CrCb 的转换或者从 Y'CrCb 到 R'G'B'的转换的 ROM 的内容：

Table 3
The content of the ROMs (R'G'B' to Y'CrCb)

$R'_m/R'_{ij0,l}$	$G'_m/G'_{ij1,l}$	$B'_m/B'_{ij2,l}$	ROM1	ROM2	ROM3
0	0	0	0	0	0
0	0	1	0.098	-0.071	0.439
0	1	0	0.504	-0.368	-0.291
0	1	1	0.602	-0.439	0.148
1	0	0	0.257	0.439	-0.148
1	0	1	0.355	0.368	0.291
1	1	0	0.761	0.071	-0.439
1	1	1	0.859	0	0

Table 4
The content of the ROMs (Y'CrCb to R'G'B')

$Y'_m/Y'_{ij0,l}$	$Cr_m/Cr_{ij1,l}$	$Cb_m/Cb_{ij2,l}$	ROM1	ROM2	ROM3
0	0	0	0	0	0
0	0	1	0	-0.392	0
0	1	0	1.596	-0.813	1.596
0	1	1	1.596	-1.025	1.596
1	0	0	1.164	1.164	1.164
1	0	1	1.164	0.772	1.164
1	1	0	2.76	0.351	2.76
1	1	1	2.76	-0.041	2.76

第二种被建议的结构能够使用以下方法去做反向变换（从 Y'CrCb 到 R'G'B'）：

- 复制用于第一种结构的、使用同样的实现方法的 ROM（带有一个可以让用户选择适当的方式的选择器）。
- 预先设置 ROM 的，这取决于所期望的转换。

预处理的部分乘积被存储在使用 13 位定点表示法（8 位数据位和 5 位功能位）的 ROM 内，改结构中使用了 13 位运算。打磨输出的输出软件 **hetwo** 结构被提出为一个使用 8 位数据并且对输出结果进行舍入。如果余数十进制大于等于 0.5 则舍入，结果将会加 1。这意味着这个条件的验证从属于另一种运算方法。一个更加有效的舍入方法是给结果加上 0.5，并且截断十进制值。这个技术已经被应用于我们所建议的结构中。每个 PE 的 ACC（用于串行结构）和每第一个加法器（用于并行结构）的初始值被设置为（Ai3+0.5）， $0 \leq i \leq 2$ 。MACs 和并行带符号加法器已经被实现在了 Xilinx CoreGen 单元，它们包含着许多有效的设计，同时还可以节省的编程时间[22]。移位寄存器和 ROM 出初始化已经由 VHDL 实现。所有的设计部件都已经被 Handel-C 连接到了一起。Handel-C 是一个一个支持并行、拥有灵活数据大小、可以被以高级语言的形式编写，并且可以直接描述 FPGA 的类 C 语言[23]。

6. 结果和分析

为了与现有的基于 FPGA 的 CSC 进行更加客观公正的比较，我们引入了 XCV50E-8 FPGA 设备作为载体。表 5 为本文提出的结构与其他方法[4-6]在占用空间与运行速度上的比较。通过多方位应用考察，与现有应用相比，本文提出的方法在 RGB-YCrCb 转换的表现上有着显著的提升[4-6]，具体表现在占用空间与最大运行时钟频率上，而表中第二个结构在转换速率上的表现大大超出现有方法。

需要指出的是，用来比较的现有方法与本文提出方法采用了相同的输入位数与输出位数。此外，我们还进行了基于软件的测试，该测试使用 2.0 GHz 英特尔奔腾 4 处理器，1 GB SDR RAM，在 Borland C++ V6.0 版本上进行。

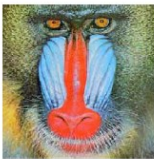





表 6 为文中第二种 DA 架构下硬件/软件实现的比较，以 RMS 误差 - 利用两种方法中数据的差值：

$$\left(\text{RMS}_{\text{Error}} = \sqrt{\frac{1}{NM} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (I_{\text{soft}}(i,j) - I_{\text{hard}}(i,j))^2} \right)$$

以及运算时间为参考标准。表 6 显示了针对两幅不同图像（狒狒 (512 x 512) 以及青椒 (256 x 256)）的测试结果

果。结果显示，FPGA 下的图像转换速度更快，且误差更小（通过两种方法的差值求得）。

Table 6
Software/hardware implementations for RGB to YCrCb CSC comparisons

Original image	Software implementation	Hardware implementation	RMS error	Computation time (ms)	
				Software	Hardware
			Y 0.487 Cr 0.630 Cb 0.461	126	1.2
			Y 0.684 Cr 0.830 Cb 0.396	43	0.28

7. 结论

在 RGB 空间，每个像素都有一组 RGB 值的情况下进行图像处理并非最有效率的方法。为了加快处理速度，很多直播、视频以及图像标准采用了辉度与色差信号，如 YCrCb，这使得格式转换成为必要步骤。RGB-YCrCb 转换需要极大的运算量，而本文展示了一个新颖、可扩展、高效的基于 DA 原理的架构。实际运行结果也证实了该架构的高效性。本文还从占用空间以及最大运行频率的角度评估了该方法的表现，实测可知，与现存其他系统相比，本文提出的系统能够利用更小的空间、进行更高频的运算。

引用

[1] B. Payette, Color space converter: R0G0B0 to Y0CrCb, Xilinx Aplication Note, XAPP637, V1.0, September, 2002.

[2] C. Poynton, A Technical Introductioin to Digital Video, Wiley, New York, 1996.

[3] R.C. Gonzalez, R.E. Woods, Digital Image Processing, second ed., Prentice-Hall, Englewood Cliffs, NJ, 2002.

[4] Data sheet (www.amphion.com), Color Space Converters, Amphion semiconductor Ltd, DS6400 V1.1, April, 2002.

[5] Application Note (www.cast-inc.com), CSC Color Space Converter, CAST Inc., April, 2002.

[6] Datasheet (www.alma-tech.com), High Performance Color Space Converter, ALMA Technologies, May 2002.

[7] M. Sima, S. Vassiliadis, S.D. Cotozana, J.T.J. van Eijndhoven, Color space conversion for MPEG decoding on FPGA-augmented TriMedia processor The 14th IEEE International Conference on Applicationspecific Systems, Architectures, and Processors (ASAP'03), Netherlands, June (2003) pp. 250–259.

[8] F. Bensaali, A. Amira, Design and efficient FPGA implementation of an RGB to YCrCb color space converter using distributed arithmetic, Proceedings of the International Conference on Field Programmable Logic (FPL), Lecture Notes in Computer Science, to be published by Springer, August, 2004.

[9] A. Amira, A Custom Coprocessor for Matrix Algorithm, PhD thesis, Queen’s University of Belfast, 2001.

[10] F. Bensaali, A. Amira, I.S. Uzun, A. Ahmedsaid, An FPGA implementation of 3D affine transformations The 10th IEEE International Conference on Electronics, Circuits and Systems (ICECS'03), Sharjah, UAE, December (2003).

[11] F. Bensaali, A. Amira, I.S. Uzun, A. Ahmedsaid, Efficient implementation of large parallel matrix product for DOTs The International Conference on Computer, Communication and Control Technologies (CCCT'03), Florida, USA, July (2003).

[12] Datasheet, (www.celoxica.com), RC1000 Reconfigurable hardware development platform, Celocixa Ltd, 2001.

- [13] URL:www.xilinx.com.
- [14] A. Albiol, L. Torres, E.J. Delp, An unsupervised color image segmentation algorithm for face detection applications Proceedings of the International Conference on Image Processing, October vol. 2 (2001) pp. 681–684.
- [15] P. Kuchi, P. Gabbur, P.S. Bhat, S. David, Human face detection and tracking using skin color modelling and connected component operators, The IETE Journal of Research, Special issue on Visual Media Processing May (2002).
- [16] M. Bartkowiak, Optimisations of color transformation for real time video decoding Digital Signal Processing for Multimedia Communications and Services, EURASIP ECMCS 2001, Budapest, September (2001).
- [17] J.L. Mitchell, W.B. Pennebaker, MPEG Video Compression Standard, Chapman and Hall, London, 1996.
- [18] J. Bracamonte, P. Stadelmann, M. Ansorge, F. Pellandini, A multiplierless implementation scheme for the JPEG image coding algorithm IEEE Nordic Signal Processing Symposium, Kolmarden, Sweden, June 13–15 (2000).
- [19] A. Amira, An FPGA based parameterisable system for discrete Hartley transforms implementation Proceedings of the International Conference on Image Processing (ICIP), Barcelona, Spain, September (2003).
- [20] H. Ohlsson, L. Wanhammer, Maximally fast numerically equivalent state-space recursive digital filters using distributed arithmetic Proceedings of the IEEE Symposium in Nordic Signal Processing (NORSIG2000), Kolmarden, Sweden, June (2000) pp. 295–298.
- [21] O. Gustafsson, L. Wanhammar, Implementation of a digital beamformer in an FPGA using distributed arithmetic Proceedings of the IEEE Symposium in Nordic Signal Processing (NORSIG2000) Kolmarden, Sweden, June (2000) pp. 295–298.
- [22] Application Note, Xilinx CoreGen, Handel-C, AN 58 v1.0, 2001.
- [23] Manual, (www.celoxica.com) Handel-C Language Reference Manual, Celocixa Ltd, 2003.