基于公共项共享的改进双三次插值算法电路研究

完 海[1],张肖强[1],杨 帆[1],郑辛星[2]
1. 安徽工程大学 电气工程学院(集成电路学院),安徽 芜湖 241000
2. 芜湖职业技术学院 信息与人工智能学院, 安徽 芜湖 241000

摘 要:目的 针对传统双三次插值缩放算法硬件资源消耗大、计算速度相对较慢的问题,提出一种利用公共项共享的改进双三次插值算法硬件电路优化方法。**方法** 该方法涉及构建双三次插值的插值系数计算公式,采用公因式消除法简化公式,目的是提取插值系数计算中的公共成分和中间插值系数;随后,在硬件电路实施过程中,将这些公共成分合并起来,进行综合计算;最终,通过对中间插值系数的表述和共享组件的整合,构建出一个优化的双三次插值电路。**结果** 理论分析表明:乘法器数量从 36 个减少到 20 个,从而降低了硬件资源消耗;所构建的双三次插值电路使用硬件描述语言,并使用 AMD Xilinx 的 Vivado 开发工具进行综合。实验结果表明:优化后的双三次插值电路在基础层面上减少了 8%的 LUT(查找表)、2%的 LUTRAM 和 14%的 DSP(数字信号处理器)资源。**结论** 事实证明:与现有优化技术相比,基于公因子共享的双三次插值算法优化方法能更有效地减少硬件电路资源消耗,同时保持图像缩放质量。

关键词:双三次插值缩放算法;公共项共享;硬件电路优化;插值系数
中图分类号:TP751 **文献标识码:**A doi:10.16055/j.issn.1672-058X.2025.0001.015

Research on Improved Bicubic Interpolation Algorithm Circuit Based on Common Factor Sharing

WAN Hai[1], ZHANG Xiaoqiang[1], YANG Fan[1], ZHENG Xinxing[2]
1. School of Electrical Engineering (School of Integrated Circuits), Anhui Polytechnic University, Anhui Wuhu 241000, China
2. School of Information and Artificial Intelligence, Wuhu Institute of Technology, Anhui Wuhu 241000, China

Abstract: Objective Aiming at the issues of large hardware resource consumption and the relatively slow calculation speed of traditional bicubic interpolation scaling algorithms, this study proposed a method to optimize hardware circuits using an improved bicubic interpolation algorithm based on common factor sharing. **Methods** This method involves constructing interpolation coefficient calculation formulas for bicubic interpolation. The common factor elimination method is employed to simplify the formulas, aiming to extract common components and intermediate interpolation coefficients in

收稿日期:2023-10-18 **修回日期:**2023-12-25 **文章编号:**1672-058X(2025)01-0112-11
基金项目:安徽省自然科学基金面上项目(1908085MF179);安徽省高校自然科学研究重点项目(KJ2019A0983);安徽省教育厅高校优秀科研创新团队项目(2022AH010059);芜湖职业技术学院优秀青年拔尖人才项目;芜湖职业技术学院校级科学研究重点项目(WZYZRZD202302).
作者简介:完海(1997—),男,安徽合肥人,硕士研究生,从事集成电路研究.
通讯作者:张肖强(1981—),男,山东烟台人,博士,副教授,从事电子系统集成、专用集成电路和信息安全芯片研究. Email: zhangxiaoqiang@ahpu.edu.cn.
引用格式:完海,张肖强,杨帆,等. 基于公共项共享的改进双三次插值算法电路研究[J]. 重庆工商大学学报(自然科学版),2025,42(1):112-122.
WAN Hai, ZHANG Xiaoqiang, YANG Fan, et al. Research on improved bicubic interpolation algorithm circuit based on common factor sharing[J]. Journal of Chongqing Technology and Business University (Natural Science Edition), 2025, 42(1): 112-122.

the calculation of interpolation coefficients. Subsequently, in the process of implementing hardware circuits, these common components are merged for comprehensive calculation. Finally, by representing the intermediate interpolation coefficients and integrating shared components, an optimized bicubic interpolation circuit is constructed. **Results** Theoretical analysis shows that the number of multipliers is reduced from 36 to 20, thereby reducing hardware resource consumption. The constructed bicubic interpolation circuit is described using hardware description language and synthesized using AMD Xilinx' s Vivado development tool. Experimental results demonstrate that the optimized bicubic interpolation circuit reduces 8% of the LUTs (lookup tables), 2% of the LUTRAMs, and 14% of the DSP (digital signal processor) resources at the basic level. **Conclusion** The study proves that compared with existing optimization techniques, the optimization method based on common factor sharing for bicubic interpolation algorithms can more effectively reduce hardware circuit resource consumption while maintaining image scaling quality.

Keywords: bicubic interpolation scaling algorithm; common factor sharing; hardware circuit optimization; interpolation coefficient

1　引　言

图像插值是数学图像处理领域的基本操作之一,它在许多实时应用中得到了广泛使用,包括超分辨率[1-2]、图像调整、几何变换以及数字图像相关性的测量。在现有插值算法中,双三次插值算法[3]被认为是在插值质量、计算复杂度和硬件资源消耗之间取得最佳效果的算法。双三次插值算法已成功应用于多个领域,包括图像缩放[4]、图像质量增强[5]以及超分辨率[6]。

在插值算法硬件实现资源优化上,Koljonen 等[7]国外学者研究了双三次插值硬件实现时的精度问题,通过对比不同小数位下的插值精度和速度,选择最优定点数双三次插值硬件实现,以达到提升缩放效率的目的;Dong 等[8]学者首次提出通过使用双三次插值将输入图像提前放大到所需的输出大小,然后,再利用卷积神经网络在图像缩放质量上提升效果,但是卷积神经网络算法在 FPGA 上并不适用,它需要以非常复杂的计算为代价去提升插值的图像质量;Boukhtache 等[9]学者提出一种减少硬件资源消耗的双三次插值算法,在结构上将线性插值与双三次插值结合,在加法器和乘法器数量上有明显改变;Nuno-Maganda 等[10]学者将双三次插值算法分为 3 个模块,第一是生成系数模块,第二为插值计算模块,第三为控制单元。通过插值计算模块,可以计算出 4 个插值点的像素值,最大时钟频率可达 100 MHz,需要 32 个乘法器和 890 个 LBs 单元,以实现对这些插值点的精准计算。

国内在插值算法硬件实现资源优化上的研究也得到了广泛关注。近年来,一些研究人员开展了很多有意义的工作。张弘[11]提出的插值算法可以有效优化资源,通过比较原始图像和目标图像的大小,来确定两者之间的坐标映射规律,从而获取当前帧两行图像的像素数据,并利用双线性插值公式计算出最终结果;严飞等[12]提出一种全新的实时视频流缩放系统,他采用双三次插值算法作为图像处理算法,对算法实现浮点运算进行优化,从而使视频缩放更高效;岳鑫等[13]提出将奇异值分解和双三次插值缩放算法进行改进,采用双三次插值算法将乘法式中左右矩阵进行重新采样,经过重构得到缩放结果图像,提高了图像的压缩率;钟雪艳等[14]提出在 FPGA 中实现双线性插值算法基于 RAM 缓冲的二级循环调度机制,以实现资源共享和并行流水线处理,加快了图像处理速度;王福强[15]提出一种全新的双线性插值与 Bandelet 变换相结合的方案,并将其应用于 FPGA,以构建出一个软核;陈光拓等[16]提出了一种新的方法,即使用小波变换来改善插值效果,并通过 Modelsim 进行模拟;张云山等[17]提出将两个相邻像素点间隔分为 8 个区间,分别计算各个区间的插值系数,然后储存起来,方便在执行算法时使用,该方法简称“查表法”。这种查表方法的缺点是插值计算的精度取决于子区间的数量,需要较大的存储空间,易造成资源浪费。

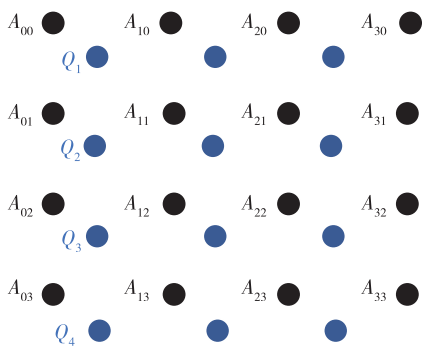
在现有的图像插值技术中,最邻近插值、双线性插值和双三次插值已经成为流行的方法。最邻近插值是最简单、运算最快的算法,它的缺点是导致新图像出现锯齿状或者马赛克状的不均匀分布;双线性插

值算法的优点是在硬件上运算是快速的,有效解决了最邻近插值算法的缺点,确保图像边缘处比较自然,不会出现锯齿现象;与最邻近插值算法和双线性插值相比,双三次插值算法在插值图像质量上能够提供更好的效果,双三次插值算法具有很好的低通滤波器特性,不会损失太多的高频信息,适用于许多安全和监控应用,例如识别车牌、验证身份、遥感等,但算法的复杂度比较高。为了减少硬件资源的消耗,本文研究了一种实时有效的异构双三次插值硬件电路优化方法。该方法在传统的双三次插值算法基础上进行改进,通过重构双三次插值计算公式,采用构建中间插值系数和合并公共项的方式,从而减少硬件实现中乘法器和加法器的使用。与 Maganda. M. A 等学者的电路优化结构相比,所设计方法表现出更少的资源消耗,实现了视频缩放质量和资源利用之间的平衡。

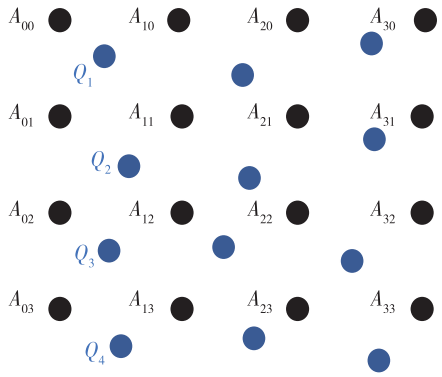
2 改进双三次插值算法的设计

2.1 双三次插值硬件架构的选择

基于双三次插值算法的硬件实现架构有两种,一种是同构架构,如图 1(a)所示,另一种是异构架构,如图 1(b)所示。同构架构指在图像缩放过程中,所有待处理像素点都会选择距离它们最近的 16 个采样点,以进行插值操作,确保采样点在相同距离位置上执行插值。异构架构指在图像缩放过程中,所有待处理像素点都会选择距离它们最近的 16 个采样点,以进行插值操作,确保采样点在不同距离位置上执行插值。在异构体系中,需要对所有的中间插值 Q_1、Q_2、Q_3、Q_4 重新进行插值运算。在同构体系中,Q_1、Q_2、Q_3 的中间插值点可以直接从上一个窗口的结果中获取,由于插值点总是处在同一个位置,因此只有 Q_4 需要重新计算,这种同构结构可以适用于图像缩放。



(a) 同构架构



(b) 异构架构

图 1 不同架构插值

Fig. 1 Interpolation for different architectures

异构架构情况更为复杂,需要更多的硬件资源。为了更好地减少在硬件实现中的资源消耗,使视频缩放应用广泛[18],该研究提出一种实时有效实现异构双三次插值的方法。在实现插值系数计算的过程中,插值基函数提出公共因子,建立中间插值系数,然后在硬件实现时共享插值系数中的公共项。通过构建中间插值系数和公共项合并运算的方法,与常规双三次插值相比,将乘法器的使用从 36 个减少到 20 个,而硬件实现过程中乘法器的使用需要占用大量资源,这大大减少了硬件资源的消耗。

2.2 图像缩放算法电路优化设计

双三次插值算法是在插值质量、硬件资源消耗和实时计算方面所需要的最佳算法。双三次插值算法具体的插值过程如图 2 所示:假设目标像素点周围最邻近的 16 个像素点为 A_{00}—A_{33}。首先,双三次插值在水平方向上进行 4 次一维插值运算处理,然后得到 4 个中间插值像素点 Q_1、Q_2、Q_3、Q_4,最后对 4 个中间插值像素点在垂直方向上再进行一次一维插值运算处理。

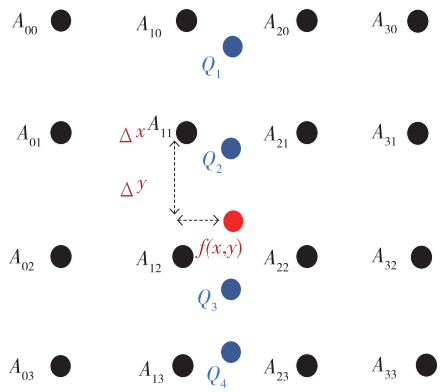


图 2 双三次插值算法像素点

Fig. 2 Pixel points of bicubic interpolation algorithm

如图2所示:在水平方向上先进行4次三次插值处理,得到4个中间插值像素点 Q_0、Q_1、Q_2、Q_3 如式(1)所示,基于所求的插值系数和目标像素点周围16个采样点的像素值,可以得到目标像素点 $f(x,y)$ 的像素值,其中 Δx、Δy 为待测像素点的坐标偏差。当进行插值运算的时候,水平插值运算和垂直插值运算的先后顺序不会影响插值结果。

$$\begin{aligned}Q_1&=A_{00}w_{01}+A_{10}w_{02}+A_{20}w_{03}+A_{30}w_{04}\\Q_2&=A_{01}w_{01}+A_{11}w_{02}+A_{21}w_{03}+A_{31}w_{04}\\Q_3&=A_{02}w_{01}+A_{12}w_{02}+A_{22}w_{03}+A_{32}w_{04}\\Q_4&=A_{03}w_{01}+A_{13}w_{02}+A_{23}w_{03}+A_{33}w_{04}\end{aligned}\tag{1}$$

式(1)中,A_{00}—A_{33} 为原图像的16个像素点,w_{01}—w_{04} 为水平方向上的插值系数。

$$f(x,y)=Q_1\cdot u_{01}+Q_2\cdot u_{02}+Q_3\cdot u_{03}+Q_4\cdot u_{04}\tag{2}$$

式(2)中,$f(x,y)$ 的像素值由中间插值像素点 Q_0、Q_1、Q_2、Q_3 与垂直插值系数 u_{01}、u_{02}、u_{03}、u_{04} 的乘积相加得到。最终得到 $f(x,y)$ 像素值的展开式如式(3)所示:

$$\begin{aligned}f(x,y)=&(A_{00}w_{01}+A_{10}w_{02}+A_{20}w_{03}+A_{30}w_{04})u_{01}+\\&(A_{01}w_{01}+A_{11}w_{02}+A_{21}w_{03}+A_{31}w_{04})u_{02}+\\&(A_{02}w_{01}+A_{12}w_{02}+A_{22}w_{03}+A_{32}w_{04})u_{03}+\\&(A_{03}w_{01}+A_{13}w_{02}+A_{23}w_{03}+A_{33}w_{04})u_{04}\end{aligned}\tag{3}$$

双三次插值基函数 $S(x)$ 表达式如式(4)所示:

$$S(x)=\begin{cases}\dfrac{3}{2}|x|^3-\dfrac{5}{2}|x|^2+1 & 0\leqslant|x|<1\\[2ex]-\dfrac{1}{2}|x|^3+\dfrac{5}{2}|x|^2-4|x|+2 & 1\leqslant|x|<2\\[2ex]0 & 2\leqslant|x|\end{cases}\tag{4}$$

将 $1+\Delta x$、Δx、$1-\Delta x$、$2-\Delta x$ 和 $1+\Delta y$、Δy、$1-\Delta y$、$2-\Delta y$ 分别代入式(4)中,计算出水平插值系数 w_{01}、w_{02}、w_{03}、w_{04} 和垂直插值系数 u_{01}、u_{02}、u_{03}、u_{04}。插值系数表达式如式(5)、式(6)所示:

$$\begin{aligned}w_{01}&=S(1+\Delta x)=-0.5\Delta x(\Delta x-1)^2\omega\\w_{02}&=S(\Delta x)=[1.5\Delta x^2-(1+\Delta x)]\cdot(\Delta x-1)\omega\\w_{03}&=S(1-\Delta x)=-\Delta x\cdot[1.5\Delta x^2-2\Delta x-0.5]\\w_{04}&=S(2-\Delta x)=-0.5\Delta x^2(\Delta x-1)\end{aligned}\tag{5}$$

$$\begin{aligned}u_{01}&=S(1+\Delta y)=-0.5\Delta y(\Delta y-1)^2\\u_{02}&=S(\Delta y)=[1.5\Delta y^2-(1+\Delta y)]\cdot(\Delta y-1)\\u_{03}&=S(1-\Delta y)=-\Delta y\cdot[1.5\Delta y^2-2\Delta y-0.5]\\u_{04}&=S(2-\Delta y)=-0.5\Delta y^2(\Delta y-1)\end{aligned}\tag{6}$$

水平插值系数计算公式和垂直系数计算公式类似。以垂直插值为例,其未优化前的垂直插值系数硬件电路如图3所示。

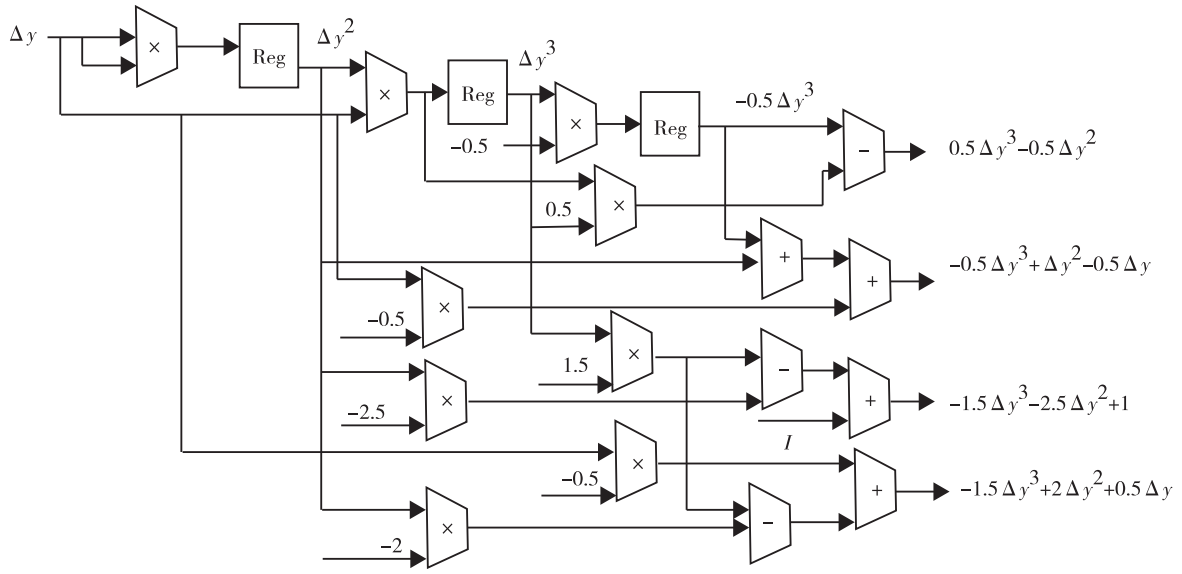


图3　垂直插值系数硬件电路图

Fig. 3　Hardware circuit diagram of vertical interpolation coefficient

对上述双三次插值系数进行公式化简,提取插值系数公式中的公共部分。首先把式(5)和式(6)代入式(3)的展开式中,然后提取公共项后得到式(7):

$$\begin{aligned}f(x,y)=&[t_{00}\cdot\Delta x^3+t_{01}\cdot\Delta x^2+t_{02}\Delta x+t_{03}]\cdot\\&[-0.5\Delta y(\Delta y-1)^2]+[t_{10}\cdot\Delta x^3+t_{11}\cdot\Delta x^2+t_{12}\Delta x+t_{13}]\cdot\\&[1.5\Delta y^2-(1+\Delta y)]\cdot(\Delta y-1)+\end{aligned}$$

$$[t_{20}\cdot\Delta x^3+t_{21}\cdot\Delta x^2+t_{22}\Delta x+t_{23}]\cdot[-\Delta y(1.5\Delta y^2-2\Delta y-0.5)]+$$
$$[t_{30}\cdot\Delta x^3+t_{31}\cdot\Delta x^2+t_{32}\Delta x+t_{33}]\cdot[-0.5\Delta y^2(\Delta y-1)] \quad (7)$$

其中,公共项为 Δx、Δx^2、$\Delta x-1$ 和 Δy、Δy^2、$\Delta y-1$,t_{00}—t_{33} 是插值系数计算简化后提取出的插值中间系数展开式,t_{00}—t_{33} 插值中间系数展开式如表 1 所示。

表 1 插值中间系数展开式

Table 1 Expansion of intermediate coefficients of interpolation

插值中间系数	插值中间系数展开式
t_{00}	$-0.5A_{00}+1.5A_{10}-1.5A_{20}+0.5A_{30}$
t_{01}	$A_{00}-2.5A_{10}+2A_{20}-0.5A_{30}$
t_{02}	$-0.5A_{00}+0.5A_{20}$
t_{03}	A_{10}
t_{10}	$-0.5A_{01}+1.5A_{11}-1.5A_{21}+0.5A_{31}$
t_{11}	$A_{01}-2.5A_{11}+2A_{21}-0.5A_{31}$
t_{12}	$-0.5A_{01}+0.5A_{21}$
t_{13}	A_{11}
t_{20}	$-0.5A_{02}+1.5A_{12}-1.5A_{22}+0.5A_{32}$
t_{21}	$A_{02}-2.5A_{12}+2A_{22}-0.5A_{32}$
t_{22}	$-0.5A_{02}+0.5A_{22}$
t_{23}	A_{12}
t_{30}	$-0.5A_{03}+1.5A_{13}-1.5A_{23}+0.5A_{33}$
t_{31}	$A_{03}-2.5A_{13}+2A_{23}-0.5A_{33}$
t_{32}	$-0.5A_{03}+0.5A_{23}$
t_{33}	A_{13}

把 t_{00}—t_{33} 插值中间系数展开式代入式(7)中,通过因式分解的处理得到式(8)—式(11),计算其相加后的值,即得到目标图像 $f(x,y)$ 像素值。

$$(A_{00}w_{01}+A_{10}w_{02}+A_{20}w_{03}+A_{30}w_{04})u_{01}=$$
$$\{A_{10}+[(-0.5A_{00}+0.5A_{20})+[(A_{00}-2.5A_{10}+5A_{20}-0.5A_{30})+$$
$$(-0.5A_{00}+1.5A_{10}-1.5A_{20}+0.5A_{30})\times\Delta x]\times\Delta x]\times\Delta x\}\times$$
$$[-0.5\cdot\Delta y(\Delta y-1)^2] \quad (8)$$

$$(A_{01}w_{01}+A_{11}w_{02}+A_{21}w_{03}+A_{31}w_{04})u_{02}=$$
$$\{A_{11}+[(-0.5A_{01}+0.5A_{21})+[(A_{01}-2.5A_{11}+5A_{21}-0.5A_{31})+$$
$$(-0.5A_{01}+1.5A_{11}-1.5A_{21}+0.5A_{31})\times\Delta x]\times\Delta x]\times\Delta x\}\times$$
$$[1.5\Delta y^2-(1+\Delta y)]\cdot(\Delta y-1) \quad (9)$$

$$(A_{02}w_{01}+A_{12}w_{02}+A_{22}w_{03}+A_{32}w_{04})u_{03}=$$
$$\{A_{12}+[(-0.5A_{02}+0.5A_{22})+[(A_{02}-2.5A_{12}+5A_{22}-0.5A_{32})+$$
$$(-0.5A_{02}+1.5A_{12}-1.5A_{22}+0.5A_{32})\times\Delta x]\times\Delta x]\times\Delta x\}\times$$
$$[-\Delta y\cdot(1.5\Delta y^2-2\Delta y-0.5)] \quad (10)$$

$$(A_{03}w_{01}+A_{13}w_{02}+A_{23}w_{03}+A_{33}w_{04})u_{04}=$$
$$\{A_{13}+[(-0.5A_{03}+0.5A_{23})+[(A_{03}-2.5A_{13}+5A_{23}-0.5A_{33})+$$
$$(-0.5A_{03}+1.5A_{13}-1.5A_{23}+0.5A_{33})\times\Delta x]\times\Delta x]\times\Delta x\}\times$$
$$[-0.5\Delta y^2(\Delta y-1)] \quad (11)$$

以 t_{00}—t_{03} 插值中间系数为例,因式分解得 $[t_{03}+(t_{02}+(t_{01}+t_{00}\times x\times x)\times x]$,搭建其硬件电路框图如图 4 所示。

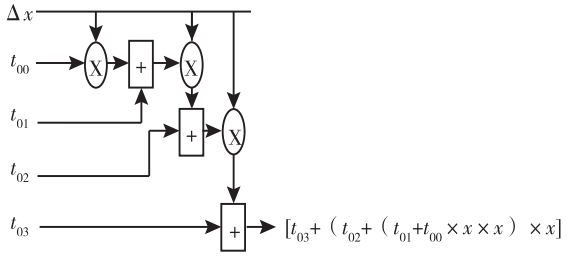


图 4 化简后中间插值系数硬件电路图

Fig. 4 Hardware circuit diagram of intermediate interpolation coefficient after simplification

通过对式(7)插值系数中公共部分的合并运算,计算中间插值系数 t_{00}—t_{33},在算法实现时均为常数的移位或加减,如图 5 所示。硬件电路中没有使用乘法器,因此很大程度上减少了硬件资源的消耗。

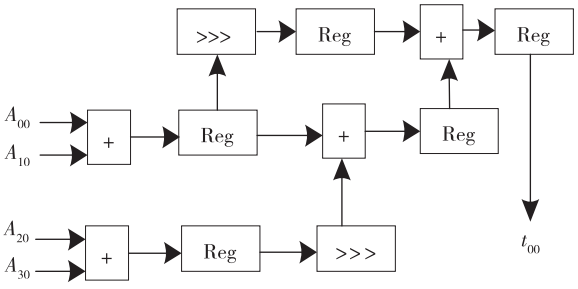


图 5 中间插值系数硬件电路图

Fig. 5 Hardware circuit diagram of intermediate interpolation coefficient

双三次插值系数计算公式进行了公式化简后,得到插值系数中的公共项 $1-\Delta y$,通过采用乘法器复用的架构,实现资源的有效重复利用,优化后的硬件结构图 6 所示。图 6 中的虚线边框标记为可复用的公共项 $1-\Delta y$,通过与化简优化前的硬件电路对比,可以发现乘法器的使用少了 7 个,加法器少了 3 个,理论上减少了硬件资源的消耗。

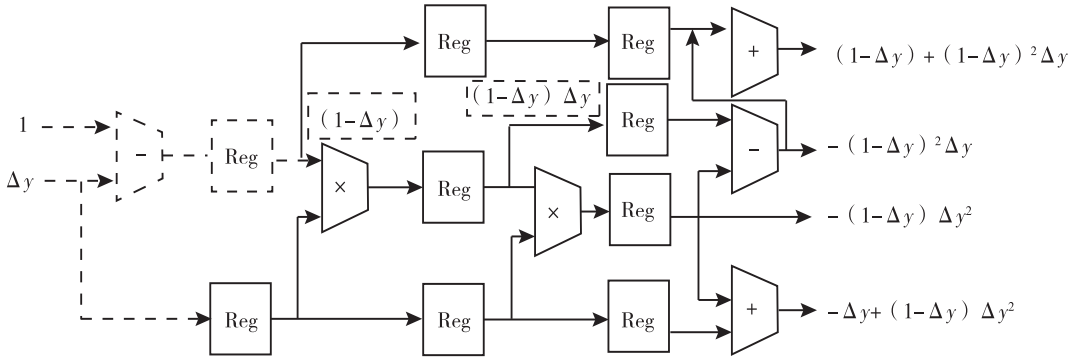


图 6　共享公共项硬件结构图

Fig. 6　Structure diagram of shared common factor hardware

如表 2 所示,对各文献乘法器、加法器数量对比,在电路优化实现之前,文献[19]的双三次插值算法硬件电路使用到的乘法器有 36 个,加法器有 40 个。使用中间插值系数和公共项合并运算方法优化构建的硬件电路,不仅平衡了硬件资源的使用和图像缩放的质量,还将乘法器从 36 个减少到 20 个。

表 2　各文献乘法器和加法器数量对比

Table 2　Comparison of the number of multipliers and adders in various literature

插值算法结构	乘法器数量/个	加法器数量/个
文献[10]	32	31
文献[17]	NA	NA
文献[19]	36	40
本文优化实现	20	39

2.3　图像缩放算法插值模块

基于双三次插值算法的公式结构可以分为 4 个模块,如图 7 所示。首先是双三次插值所需要的 16 个邻域像素点的像素值 A_{00}—A_{33},其次是通过坐标偏差值计算的插值系数,然后是表 1 中所有的 t_{ij} 系数,最后是整个图像缩放算法所需要的乘法器和加法器,最终构成图像缩放算法的硬件实现。

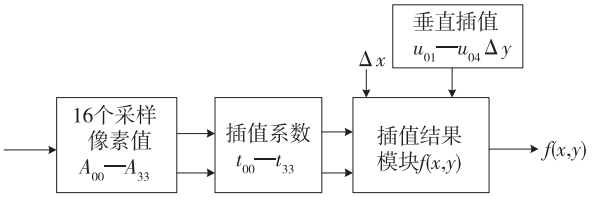


图 7　缩放算法模块硬件结构图

Fig. 7　Hardware structure of scaling algorithm module

将表 1 中 t_{ij} 代入计算后得到的式(8)—式(11)相加就可得到插值后像素点的像素值 $f(x,y)$。缩放算法模块硬件结构图由 4 个部分组成,分别是图像采样的 16 个像素点、简化后的部分插值系数、垂直插值系数、插值结果模块。插值结果计算模块 MUL_SUB 硬件结构如图 8 所示。

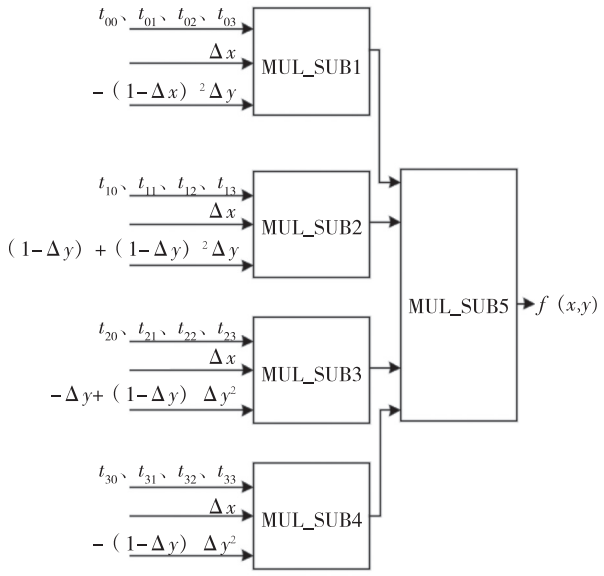


图 8　插值结果模块 MUL_SUB 硬件架构图

Fig. 8　MUL_SUB hardware architecture diagram of interpolation result module

16 个采样的像素值,是由图像传感器采集的像素点数据,像素点数据通过 OV5640 采样后送入计算插值系数模块,表 1 中所有的中间插值系数均是由采集后的 16 个像素点像素值,通过加法或者移位计算得到的。把计算的垂直插值系数 u_{01}、u_{02}、u_{03}、u_{04} 代入,最后得到插值结果 $f(x,y)$。使用并行计算 ,和垂直插值系数 $-0.5\Delta y(\Delta y-1)^2$ 相乘后得[$t_{00}\cdot\Delta x^3+t_{01}\cdot\Delta x^2+t_{02}\cdot$

$\Delta x+t_{03}]\cdot[-0.5\cdot\Delta y(\Delta y-1)^2]$;最后将所有计算结果在 MUL_SUB5 中相加得到目标像素点的值$f(x,y)$。

在对处理时间要求不高时,组合逻辑复用可以减少逻辑资源的使用和降低芯片面积(实际上,CPU 也是一种在不同时间通过指令来复用各个模块的方式)。程序中,在计数值不同时,选择不同的值作为乘法器的输入。程序中至多只使用一个多位宽乘法器,以减少硬件的面积占用。同一个乘法器处理不同的输入值,即在缩放算法模块进行水平插值系数和垂直插值系数计算时,将同类变量合并使用同一个乘法器单元。因为在 FPGA 中,乘法器比加减法器的资源消耗要大,所以乘法器的使用减少,可以有效减少硬件电路的资源消耗。

所提出的一种实时有效实现异构双三次插值的方法在硬件电路实现中通过将插值系数的共享部分合并,构建中间插值系数和公共项的合并运算,从而创建了一个缩放算法模块。因此,与之前的研究文献[10]、文献[17]和文献[19]相比,成功减少了乘法器和加法器的数量。

3 仿真实验与结果分析

3.1 图像缩放算法比较分析

为了验证本文方法的有效性,可以与常规图像缩放算法进行对比,常规缩放算法有最邻近插值算法、双线性插值算法、双三次插值算法。本文通过 3 幅图进行测试,将其进行放大,与源图像进行比较(图 9),通过峰值信噪比(R_{PNSR})和均方误差(R_{MSE})对图像质量进行评估,其表达式如下:

$$R_{PSNR}=10\ln\left(\frac{255^2}{\frac{1}{MN}\sum_{n}^{N-1}\sum_{m}^{M-1}[F(m,n)-G(m,n)]^2}\right) \tag{12}$$

$$R_{MSE}=\frac{1}{MN}\sum_{n}^{N-1}\sum_{m}^{M-1}[F(m,n)-G(m,n)]^2 \tag{13}$$

其中:$F(m,n)$代表原始图像,其尺寸范围介于 0~255,m 和 n 分别表示图像的高度和宽度;而 $G(m,n)$则是通过缩放处理获得的新图像。R_{PSNR} 单位为 dB,它的数值越高,说明插值的精度越高,也就是插值质量越高,R_{MSE} 的值越小,插值算法质量越好。

图 9 测试图像(从左往右依次为:古楼 山丘 山水画)

Fig. 9 Test images (from left to right: ancient building, hill, and landscape painting)

分析不同算法的图像缩放质量$\left(\frac{R_{MSE}}{R_{PSNR}}\right)$,如表 3 所示,本文的方法要优于最邻近插值算法和双线性插值算法,与常规的双三次插值算法接近。

表 3 图像缩放质量比较$\left(\frac{R_{MSE}}{R_{PSNR}}\right)$

Table 3 Comparison of image scaling quality$\left(\frac{R_{MSE}}{R_{PSNR}}\right)$

图像类型	最邻近插值	双线性插值算法	双三次插值算法	本文方法
古楼	33.263/30.673	33.388/29.739	33.440/29.447	33.441/29.440
山丘	35.166/19.790	35.375/18.862	35.419/18.672	35.424/18.650
山水画	33.259/30.700	33.360/29.994	33.378/29.872	33.376/29.880

3.2 图像缩放算法模块实验验证

图像数据处理有两种方式,一种是对图像数据进行定点数转浮点数,另一种是直接使用处理后的浮点数进行计算。虽然定点数计算消耗资源相对较小,运算较快,但是无法在动态范围中满足计算精度的要求,并且定点数无法灵活改变字长格式。该设计实验采用浮点数运算的方式,浮点数计算方式有着动态范围大,基本不存在数值溢出的优势。实验验证对比的是常规

双三次插值算法硬件架构和改进后的双三次插值算法硬件架构以及文献[10]和文献[17]中的硬件资源消耗和图像的缩放质量。

实验验证内容如下:首先,对常规插值系数算法模块进行搭建,使用 Vivado 工具中的 Simulation 仿真工具进行分析,然后通过 Vivado 工具进行综合,实现后得到底层资源报告;其次,对所提出的插值系数模块,基于共享公共因子和中间插值系数的算法硬件架构进行搭建,使用 Vivado 工具中的 Simulation 仿真工具进行分析,然后通过 Vivado 工具进行综合,实现后得到底层资源报告;最后,通过将提出的插值系数模块与常规插值系数模块、文献[10]、文献[19]的硬件电路资源消耗进行对比,得到提出的插值系数模块乘法器从 36 个减少到 20 个,可以有效减少底层资源 LUT、LUTRAM 和 DSP 的资源消耗,分别降低了 8%、2%、14%。本次使用的是 Xilinx 公司 A7 系列 FPGA,其中逻辑资源 LUTs、LUTRAM、FF、DSP 分别为 63 400、19 000、126 800、240。

根据式(7),对常规插值系数算法模块进行硬件电路的构建。根据计算公式,把 $1+\Delta x$、Δx、$1-\Delta x$、$2-\Delta x$ 和 $1+\Delta y$、Δy、$1-\Delta y$、$2-\Delta y$ 分别代入式(4)中,得到水平插值系数 w_{01}、w_{02}、w_{03}、w_{04} 和垂直插值系数 u_{01}、u_{02}、u_{03}、u_{04}。通过 RTL 级行为级仿真,使用 Vivado 工具中的 Simulation 仿真工具,对搭建好的双三次插值算法硬件架构进行波形仿真。

Simulation 仿真结果如图 10 所示:首先,把 640×480 的图像数据转化为固定的定点数;其次,通过硬件描述转换为浮点数 bi[31:0],浮点为单精度浮点,数据位宽指数位为 8 bit,尾数位为 23 bit,符号位为 1 bit;最后,在 20 ns 的时钟周期下得到插值计算后的浮点数数据 m_data_i[31:0]。

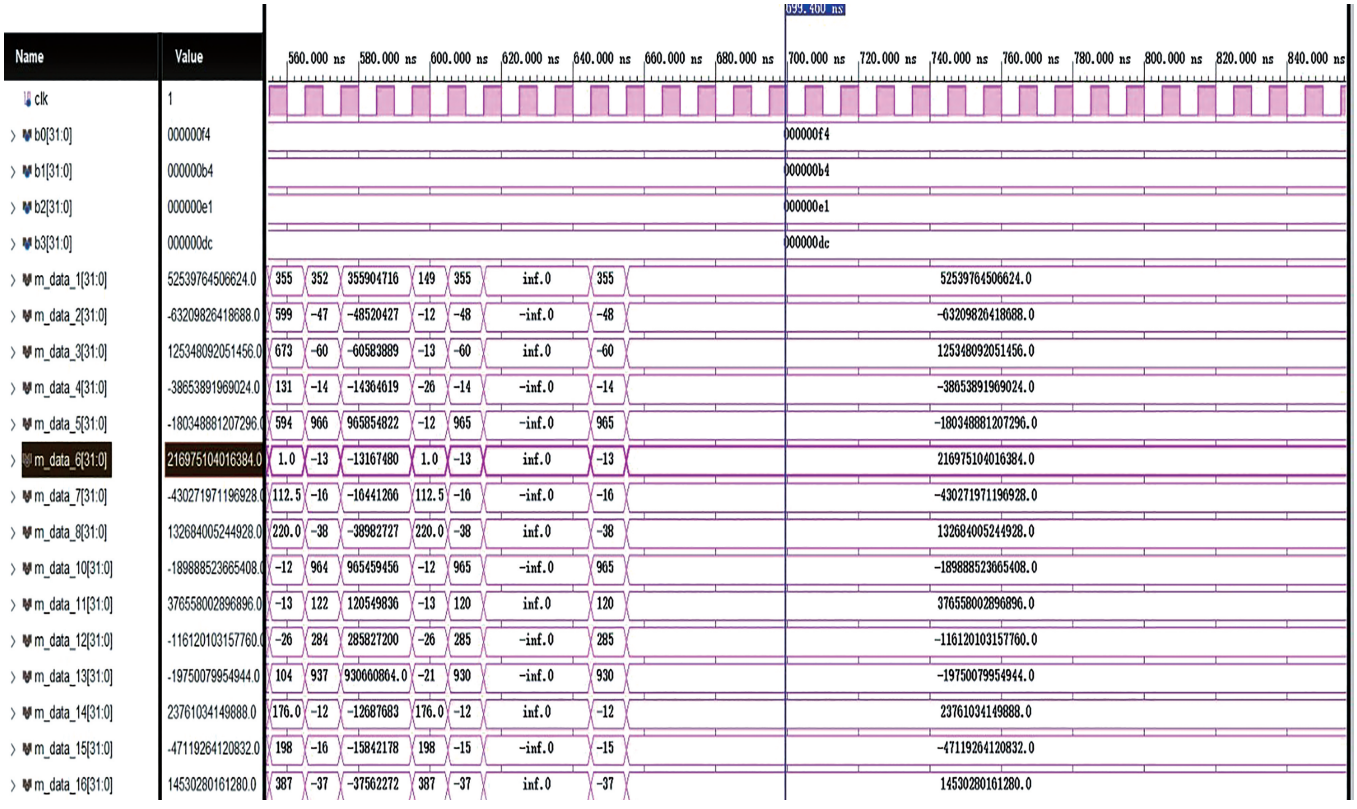


图 10　常规双三次插值算法硬件架构波形仿真

Fig. 10　Hardware architecture waveform simulation of conventional bicubic interpolation algorithm

在基于公共项共享的双三次插值算法插值系数模块硬件电路的验证实验中,对提出的插值系数的计算可以采用共享公共因子和中间插值系数的方法,对算法架构进行搭建。通过 RTL 级行为级仿真结果如图 11 所示。

最后使用 vivado 工具中的 Simulation 仿真工具,对搭建好的改进后的双三次插值算法硬件架构进行波形仿真。仿真结果如图 12 所示。

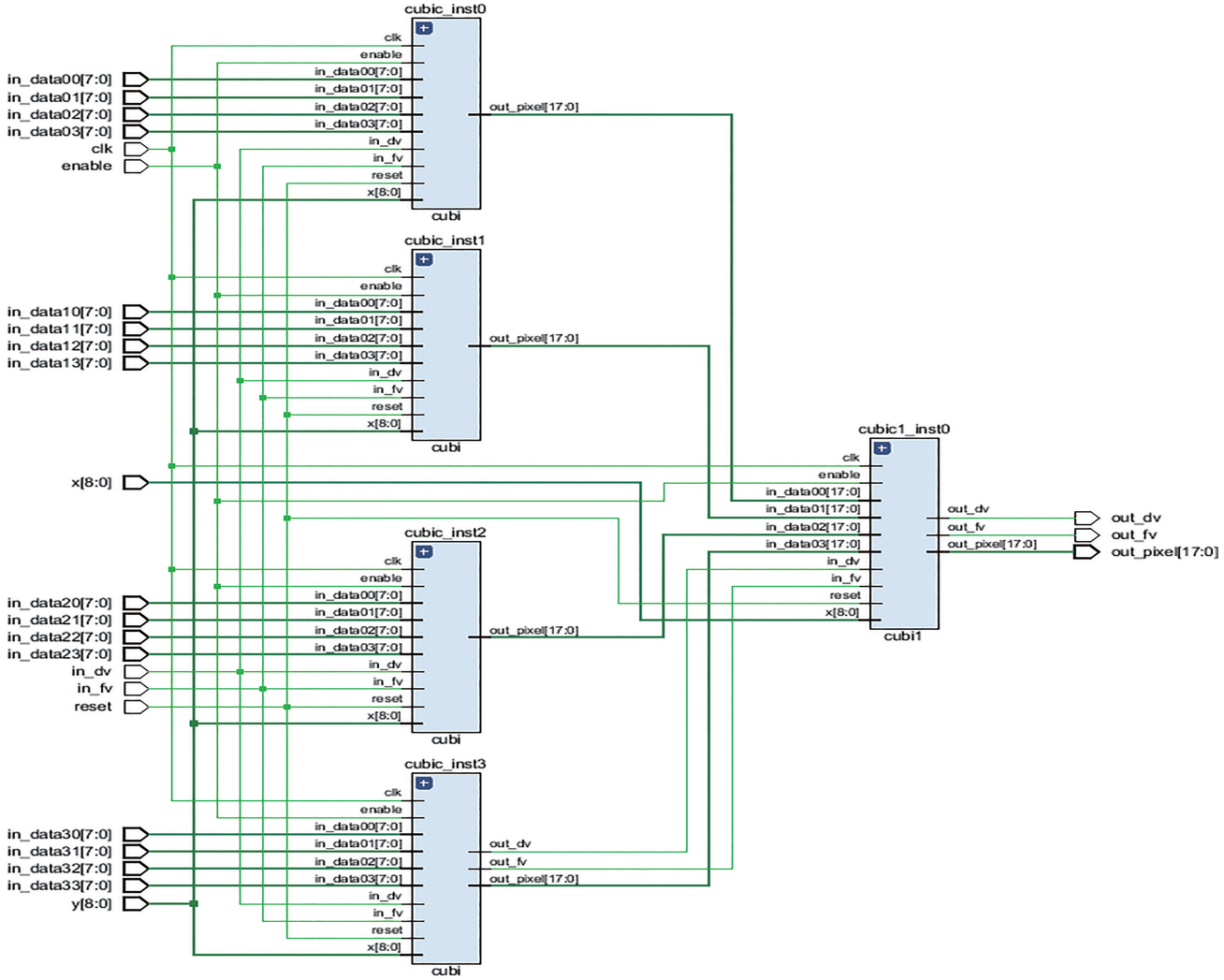


图 11 基于公共项共享的双三次插值算法插值系数模块 RTL 分析原理图

Fig. 11 RTL analysis diagram of interpolation coefficient module of bicubic interpolation algorithm based on shared common items

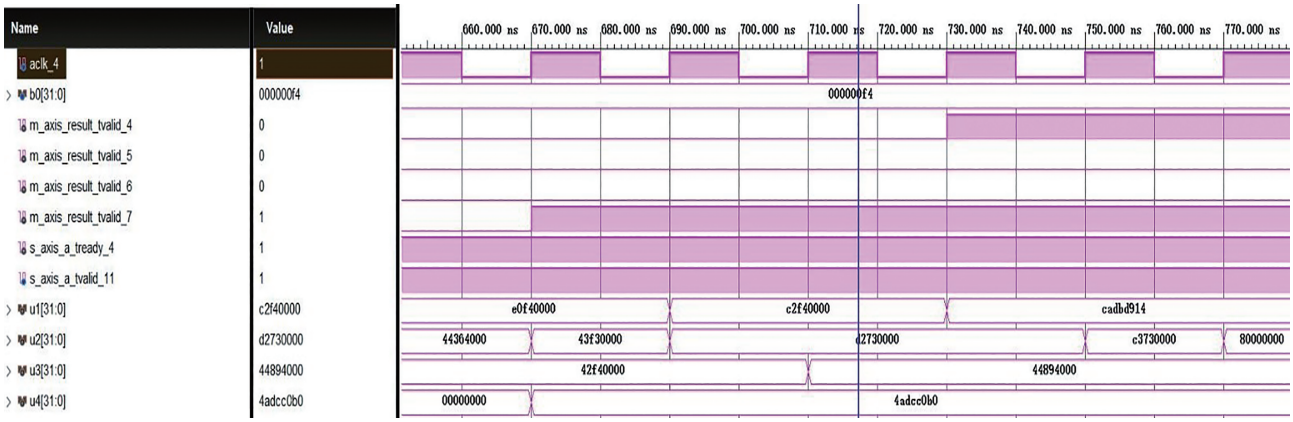


图 12 提出的插值系数模块硬件架构波形仿真

Fig. 12 The waveform simulation of the hardware architecture of the proposed interpolation coefficient module

根据所设计内容,设计实验采用浮点数运算的方式有着动态范围大,基本不存在数值溢出的优势。实验验证对比的是常规双三次插值算法硬件架构和改进后的双三次插值算法硬件架构以及文献[10]和文献[19]。在文献[10]中,实验使用的是英特尔 Virtex-II 型 FPGA,文献[19]中使用的是 Xilinx A7 系列定制芯片。因为所提

出的插值系数模块硬件是对图像做定点数处理,所以在参考范围之内对硬件资源的使用有所差距。

如表 4 所示,使用常规的双三次插值算法硬件架构对于 FPGA 底层资源的消耗大于使用基于公共项共享的双三次插值算法硬件架构,并且在 LUT 的资源消耗上,所提出的算法架构比文献[10]明显要少。通过上述实验对比可以得出:基于公共项共享的双三次插值算法硬件电路优化的方法,在减少硬件电路资源消耗的同时,不会影响图片缩放的质量。

表 4　双三次插值算法硬件架构底层资源消耗

Table 4　Resource consumption of the underlying hardware architecture of bicubic interpolation algorithm

相关文献	乘法器数量	LUTS	LUTRAM	DSP
优化实现	36	8 291	585	65
直接实现	20	3 130	244	32
[10]	32	3 078	2 233	1 543
[19]	36	870	574	NA

3.3　实验结果

为了验证改进的双三次插值算法,使用的开发板为 AMD Xilinx 公司所生产的 Aritx7 系列芯片,开发环境为 Xilinx Vivado。视频缩放系统的实验验证均在该平台实现。平台的主要器件为 OV5640 传感器、HDMI 传输线、HDMI 显示器、联合测试工作组(Joint Test Action Group,JTAG)下载器、电源线。视频图像缩放系统最高工作频为 200 MHz,能够实时处理从高清清晰度 1 280×720 到 1 920×1 080 的视频信号。

在 FPGA 开发板上进行测试,进行完行为级仿真和功能性仿真试验后,经过 Vivado 软件的 JTAG 下载器,可以将 OV5640 摄像头模块安装在“OLED/CAMERA”插座上,并且将 HDMI 电缆的一端与 HDMI_B 插座相连,另一端则与显示器相连,以此来完成上板测试。连接电源线并打开电源开关,接下来下载程序,验证实时显示功能。然后观察实验图像,如图 13 所示。

(a) 视频缩放前(1 280 * 720)

(b) 视频缩放后(1 920 * 1 080)

图 13　视频缩放前后比较

Fig. 13　Comparison before and after video scaling

平台的消耗资源如表 5 所示,系统中图像输出的时钟频率为 200 MHz,搭建的图像缩放系统所占有的资源在可接受范围内。

表 5　视频缩放系统 FPGA 实现资源消耗表

Table 5　Resource consumption table of video scaling system FPGA implementation

资　源	资源利用	可用资源	资源利用率/%
LUT	37 355	63 400	58.92
LUTRAM	13 585	19 000	71.52
FF	22 911	126 800	18.06
DSP	135	240	56.52
IO	171	285	60
MMCM	4	10	40

通过上述设计实验表明:基于双三次插值算法的视频缩放系统基本满足系统设计要求,上板验证了视频从 OV5640 输入经过缩放处理,最终到 HDMI 显示,缩放后的图像比原始图像更加清晰,图像显示正常,没有出现丢帧的现象。

4　结　论

所提出的是一种能够实时有效实现异构双三次插值的方法,重构了双三次插值计算公式,应用公式化简分解了插值基函数,提取出了插值系数中的公共因子,在进行插值系数计算硬件实现过程中把这些公共因子进行合并运算。通过构建中间插值系数和公共项合并运算的方法进行了硬件结构设计。最后上板验证使用的是 Xilinx 公司 A7 系列 FPGA,把乘法器从 36 个减少到 20 个,设计图像缩放算法架构综合后得到底层资源 LUT、LUTRAM 和 DSP 分别减了 8%、2%、14%的资源消耗。基于公共项共享的缩放算法硬件实现方法,在减少硬件电路资源消耗的同时,不会影响图片缩放的质量。

参考文献(References):

[1] HARRIS J L. Diffraction and resolving power[J]. Journal of the Optical Society of America, 1964, 54(7): 931-938.

[2] 郭帅志. 基于最大似然估计的超分辨重建算法研究与FPGA实现[D]. 合肥: 中国科学技术大学, 2019.
GUO Shuai-zhi. Research on algorithms of super resolution reconstruction based on maximum likelihood estimation and FPGA implementation[D]. Hefei: University of Science and Technology of China, 2019.

[3] KEYS R G. Cubic convolution interpolation for digital image processing[J]. IEEE Transactions on Acoustics Speech and Signal Processing, 1981, 29: 1153-1160.

[4] XIANG Z, ZOU X, LIU Z. An high quality image scaling engine for large-scale LCD[C]//Proceedings of the 8th international Conference on Signal Processing. Piscataway: IEEE Press, 2006: 1-5.

[5] HUNG N V, THU HIEN N T, VINH P T, et al. An utilization of edge detection in a modified Bicubic interpolation used for frame enhancement in a camera-based traffic monitoring[C]//Proceedings of the International Conference on Information and Communications. Piscataway: IEEE Press, 2017.

[6] RUANGSANG W, ARAMVITH S. Efficient super-resolution algorithm using overlapping bicubic interpolation[C]//Proceedings of the IEEE 6th Global Conference on Consumer Electronics. Piscataway: IEEE Press, 2017.

[7] KOLJONEN J, BOCHKO V A, LAURONEN S J, et al. Fast fixed-point bicubic interpolation algorithm on FPGA[C]//Proceedings of the IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip. Piscataway: IEEE Press, 2019.

[8] DONG C, LOY C C, HE K, et al. Learnin g a deep convolutional network for image super-resolution[C]//European Conference on Computer Vision. Cham: Springer, 2014: 184-199.

[9] BOUKHTACHE S, BLAYSAT B, GREDIAC M, et al. Alternatives to bicubic interpolation considering FPGA hardware resource consumption[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2021, 29(2): 247-258.

[10] NUNO-MAGANDA M A, ARIAS-ESTRADAM O. Real-time FPGA-based architecture for bicubic interpolation: an application for digital image scaling[C]//Proceedings of the Proceedings of the 2005 International Conference on Reconfigurable Computing and FPGAs. Piscataway: IEEE Press, 2005.

[11] 张弘. 基于FPGA的视频图像处理的研究与实现[D]. 成都: 电子科技大学, 2020.
ZHANG Hong. Research and implementation of video image processing based on FPGA [D]. Chengdu: University of Electronic Science and Technology of China, 2020.

[12] 严飞, 陆宝毅, 刘银萍, 等. 实时视频流缩放系统设计[J]. 液晶与显示, 2019, 34(11): 1124-1130.
YAN Fei, LU Bao-yi, LIU Yin-ping, et al. Design of real-time video stream scaling system[J]. Chinese Journal of Liquid Crystals and Displays, 2019, 34(11): 1124-1130.

[13] 岳鑫, 肖晨. 基于奇异值分解和双三次插值的图像缩放算法改进[J]. 西安邮电大学学报, 2018, 23(4): 72-77.
YUE Xin, XIAO Chen. Improvement of image scaling algorithm based on singular value decomposition and bicubic interpolation[J]. Journal of Xi' an University of Posts and Telecommunications, 2018, 23(4): 72-77.

[14] 钟雪燕, 夏前亮, 陈智军. 基于FPGA的图像超分辨率的硬件化实现[J]. 现代电子技术, 2017, 40(17): 44-46, 50.
ZHONG Xue-yan, XIA Qian-liang, CHEN Zhi-jun. FPGA-based hardware implementation of image super-resolution[J]. Modern Electronics Technique, 2017, 40(17): 44-46, 50.

[15] 王福强. 基于FPGA超分辨率图像放大算法研究及实现[D]. 四川绵阳: 西南科技大学, 2012.
WANG Fu-qiang. Research and implementation of super-resolution image enlargement algorithm based on FPGA [D]. Sichuan, Mianyang: Southwest University of Science and Technology, 2012.

[16] 陈光拓, 孙洋, 杜雨洺, 等. 基于小波的超分辨率算法研究及FPGA实现[J]. 成都信息工程大学学报, 2017, 32(6): 601-604.
CHEN Guang-tuo, SUN Yang, DU Yu-ming, et al. Research and FPGA implementation of super-resolution algorithm based on wavelet transform[J]. Journal of Chengdu University of Information Technology, 2017, 32(6): 601-604.

[17] ZHANG Y, LI Y, ZHEN J, et al. The hardware realization of the bicubic interpolation enlargement algorithm based on FPGA[C]//Proceedings of the Third International Symposium on Information Processing. Piscataway: IEEE Press, 2010: 277-281.

[18] KUMAR A K, PATNAIK S, JEEVARATNAM N. On-chip memory for image processing applications based on FPGA[C]//Proceedings of the International Conference on Signal Processing, Communication, Power and Embedded System. Piscataway: IEEE Press, 2016.

[19] LIU D Q, ZHOU G Q, ZHOU X, et al. Fpga-based on-board cubic convolution interpolation for spaceborne georeferencing[J]. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2020, XLII-3/W10: 349-356.

责任编辑:李翠薇