基于 Catapult C Synthesis 的图像校正算法设计

李 杨
(辽宁广播电视大学 沈阳 110034)

摘 要：为解决遥感相机在运动过程中的抖动造成的图像位置偏移问题，提出了一种实时图像校正算法。由于在FPGA中采用HDL进行算法设计难度大、开发周期长，故设计中采用了C语言进行算法设计，然后借助Calypto公司的Catapult C Synthesis工具将抽象的C设计转换成硬件RTL代码。在Catapult C Synthesis中对设计的算法进行了C/C++、RLT协同仿真测试，并在Xilinx XC5VLX110T型FPGA上进行了验证。仿真测试及硬件验证结果表明，采用Catapult C Synthesis设计的算法在时序、性能方面均满足设计要求，能够对偏移的图像进行实时校正。

关键词：实时图像校正；FPGA；Catapult C Synthesis；RTL
中图分类号：TN393.1 **文献标识码**：B **国家标准学科分类代码**：520.3040

Design of image correction algorithm based on Catapult C Synthesis

Li Yang
(Liaoning Radio and TV University, Shenyang 110034, China)

Abstract: The jitter of remote sensor during exercise will cause the image position shift. To settle this problem, a real-time image correction algorithm is proposed. However, the design of algorithm by using HDL has great difficulty and it leads to long development cycle. Therefore, the design of algorithm is achieved by adopting C language. Then the abstract C algorithm is translated to RLT code by means of Calypto's Catapult C Synthesis tools. Furthermore, a co-simulation test of C/C++ and RLT is carried out in Catapult C Synthesis tools. It's also verified in the platform of XC5VLX110T. The results of Simulation test and hardware verification show that the algorithm designed by applying Catapult C Synthesis tools can meet the design requirements both in timing and performance. And it can correct the shifted image in real-time.

Keywords: real-time image correction; FPGA; Catapult C Synthesis; RLT

1 引 言

随着遥感技术的不断发展，对时间分辨率以及空间分辨率的要求越来越高，这直接导致了数传系统的带宽不断增大[1]。受制于现有技术手段，数传系统带宽的提高十分有限。因此，需要对图像数据进行实时处理，减小数传系统的压力[2]。

FPGA在流水线设计、并行处理方面的优势使得其在图像处理方面具有明显的优势[3-4]。然而在过去的数十年里，主要是基于C语言的硬件编程算法研究较为成熟，而采用硬件编程语言Verilog或VHDL进行算法设计的研究并不多[5]。为解决遥感相机在运动过程中的抖动造成的偏移问题，本文结合FPGA在图像算法处理方面的优势以及C语言进行算法设计的简易性，采用了Calypto公司的高级综合工具Catapult C Synthesis进行了图像校正算法的设计，将C编写的算法程序转换成RTL代码，在减小设计的难度同时，以最短的设计周期解决图像偏移问题。

2 Catapult C Synthesis 算法设计流程

传统算法设计与Catapult C算法设计流程对比如图1所示。从图1中可以看出，传统设计流程中的微架构设计、RTL代码编写以及面积和时序的优化工作均可由Catapult C Synthesis工具自动完成。在传统设计流程中，由于浮点运算的RTL编码较为复杂，通常将浮点数据转换成定点数据进行处理。而最新版的Catapult C Synthesis工具加入了对浮点数据的支持，在数据精度要求较高的场合，可以直接使用浮点数据，而不需要为了处理方便而对数据的精度做出妥协。

收稿日期：2015-12

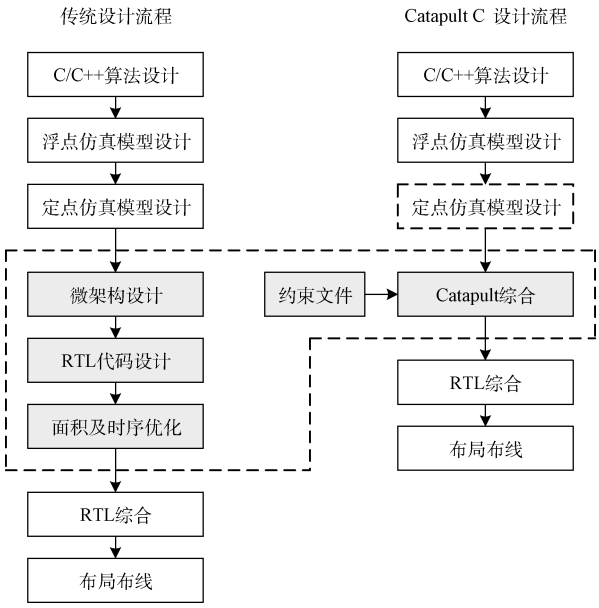

图 1　两种设计流程对比

Catapult 综合部分的具体流程包括:1)算法 C 源程序设计;2)算法分析及约束设置;3)C/C++、RTL 协同仿真;4)综合生成门级电路;5)功耗及性能分析。若功耗及性能分析满足要求,设计流程结束,否则修改约束设置,重复上述流程[6]。

2　图像校正算法设计

2.1　图像校正算法原理

遥感相机运动过程中的抖动造成的图像偏移现象如图 2 所示。实线所示的图像为正常图像,虚线所示的两幅图像在 x 和 y 方向均产生了偏移,即此时相机获取到的图像并不是实际需要获取位置的图像。本文介绍的图像校正算法就是通过多帧图像数据求取出偏移量 Δx 和 Δy,然后把偏移的图像校准回原位置。

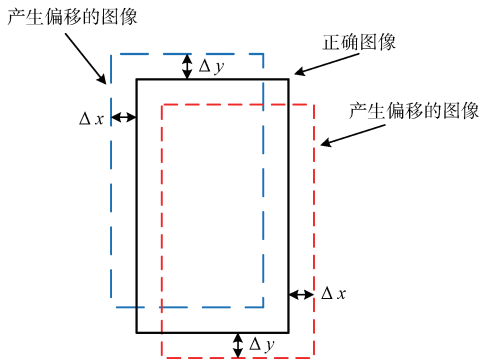

图 2　图像产生位置偏移示意

算法的基本思想是:首先对输入的图像数据进行均值滤波,平滑噪声[7-8];然后对去噪后的前后帧图像数据分别在 x 和 y 方向求取方差,如果图像没有偏移,方差值为 0,否则,图像偏移越大,方差值越大;最后将求取的 x 和 y 方向的方差分别进行多项式拟合,求取出两条曲线,计算曲线的极值点,极值点的坐标即为 x 和 y 方向的位移(理论上,图像未发生偏移时,极值点坐标为 0),然后将偏移的图像向相反的方向移动 Δx 和 Δy,即得到校正后的图像。故本算法的关键是求取出 Δx 和 Δy。

2.2　图像校正算法的实现

一帧图像由 1 280×1 024 个像素组成。图像的输入数据流如图 3 所示,每个像素数据维持 8 个时钟周期,数据标志位维持 1 个时钟周期。因此,要实现实时图像校正,对每个像素数据的所有处理操作必须在 8 个时钟周期内完成。

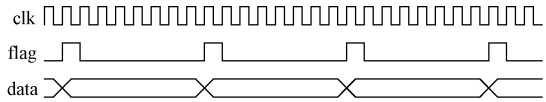

图 3　输入数据流

输出数据流如图 4 所示。从第 2 帧开始,每一帧的结尾给出 x 和 y 方向的计算结果。图中的 cnt 是帧计数变量(添加 cnt 便于输出时序理解),非实际输出信号。

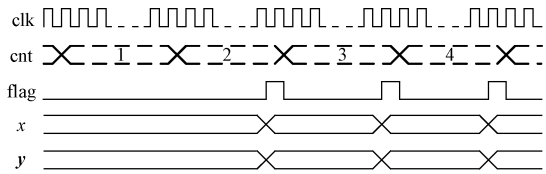

图 4　输出数据流

2.2.1　均值滤波算法的实现

均值滤波的原理基于邻域平均法,将原图像中的一个像素与它邻域的 $M-1$ 个像素的灰度值求和,然后取平均值代替原图像中像素的灰度值[9]。邻域平均的数学表达式如式(1)所示[10]:

$$g(x,y)=\frac{1}{M}\sum_{f\in S}f(x,y) \tag{1}$$

式中:$g(x,y)$ 为处理后的像素灰度值,$f(x,y)$ 为待处理像素点及该点邻域像素点的灰度值,S 为模板,M 为模板中像素的个数。

设计中采用 5x5 的模板。在 Catapult C 中创建一个二维数组 $A[5][5]$用于存储模板数据。由于图像数据是按行顺序输入的,只有当第 5 行第 5 个数据到来时,才能开始进行第一次均值滤波计算,故需要开辟 4 个长度为 1 276 的一维数组 store1～store4(一行图像数据为 1 280 个像素,其中每行有 4 个输入数据直接存储到模板数组 A 中),用于保存前 4 行的数据。采用模板窗口滑动的方式进行计算,每次参与计算的 5×5 模板中只有 1 列数据发生改变,即 $A[4][0]$～$A[4][4]$从 store1～store4 和输入数据流中读取 5 个新的数据,然后数组 A 按列进行平移,得到新的模板数

据。该算法的设计中，需要在 Catapult C 中对 for 循环进行约束设置，设置为 unroll 或 pipeline 方式均可在 8 个时钟周期内给出计算结果，为节省资源选择了 pipeline 方式。

2.2.2　方差的实现

方差计算过程如图 5 所示。上一帧的每一行数据都要和当前帧的对应行以及上下各两行数据做方差运算，分别记作 y_{-2}、y_{-1}、y_0、y_1、y_2，然后把所有行对应的这 5 个计算结果分别对应相加，得到最终的方差。同理 x 方向做类似的计算，分别得到 x_{-2}、x_{-1}、x_0、x_1、x_2。

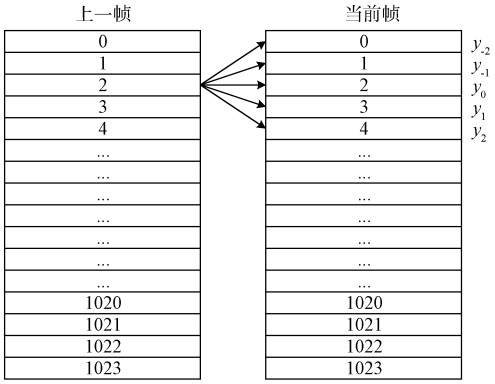

图 5　方差计算示意

为节省资源，在设计中未对所有图像帧数据进行缓存，而是采用了乒乓操作的思想，只存储两帧图像数据。在 Catapult C 中开辟两个二维数组 store1A[1024][1276]和 store1B[1024][1276]，奇数帧写 store1A，读 store1B；偶数帧写 store1B，读 store1A。方差计算过程中第 0 行不能计算 y_{-2} 和 y_{-1}，第 1 行不能计算 y_{-2}，第 1022 行不能计算 y_2，第 1023 行不能计算 y_1 和 y_2，所以在对所有行对应的这 5 个参数对应相加后，只有 y_0 是 1024 行累加计算的结果，其他 4 个参数需要进行加权处理。该算法最终输出的 x 和 y 方向的 5 个计算值分别用于后续的多项式拟合。

在 x 方向计算时，当前帧的一个像素需要分别和上一帧对应位置的像素以及左右各两个像素进行计算。上一帧的这 5 个像素存储在同一个数组 store1A 或 store1B 中，对应到 RTL 中，即这 5 个像素位于同一个 RAM 中。由于在一个时钟周期只能读写 RAM 中的一个数据，因此，在 for 循环中做 x 方向的计算时，不能将 Catapult 的 for 循环约束设置为 unroll 模式，只能设置为 pipeline 方式，这样在 5 个时钟周期内可以完成 x 方向的计算，仍然不超过每个像素的 8 个维持周期。

2.2.3　多项式拟合算法的实现

该算法将方差计算后的$(-2, y_{-2})$、$(-1, y_{-1})$、$(0, y_0)$、$(1, y_1)$、$(2, y_2)$5 个点进行多项式拟合，得到一条曲线。然后对曲线求极值，得到极值点对应的坐标。理论上，如果图像没有偏移，极值点坐标应该为 0，如果极值点坐标不为 0，该坐标即为图像偏移的距离，把图像向相反的方向移动相同的距离，即得到校正后的图像。

设计中根据式(3)～(7)拟合出了形如式(2)所示的 4 次多项式[11]。

$$y = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4 \tag{2}$$

$$\boldsymbol{B} = \begin{bmatrix} 1 & x_1 & \cdots & x_1^4 \\ 1 & x_2 & \cdots & x_2^4 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_5 & \cdots & x_5^4 \end{bmatrix} \tag{3}$$

$$\boldsymbol{C} = [c_0 \quad c_1 \quad c_2 \quad c_3 \quad c_4]^{\mathrm{T}} \tag{4}$$

$$\boldsymbol{Y} = [y_{-2} \quad y_{-1} \quad y_0 \quad y_1 \quad y_2] \tag{5}$$

$$\boldsymbol{T} = (\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B})^{-1}\boldsymbol{B} \tag{6}$$

$$\boldsymbol{C} = \boldsymbol{T} \cdot \boldsymbol{Y} \tag{7}$$

在 $x_1 \sim x_4$ 已知的情况下(分别对应－2～2)，$\boldsymbol{B}$ 和 $\boldsymbol{T}$ 均为常量，可以通过 MATLAB 计算出具体值，然后在设计中建立两个常量存储 B 和 T。故在 Catapult C 中只需要编写式(7)的计算程序即可。式(7)为两个矩阵相乘，在 C 中可以通过 for 循环实现，实现方法较为简单，此处不再赘述。

由于相机在运动过程中的抖动通常不能过大，否则该相机的设计是不合理的。因此，极值点应该出现在零点附近。根据实际情况，以 0 为中心，以 0.001 为步进量，在 0 左右各取 100 个点代入多项式求值，然后比较 201 个值的大小，得到最小值的坐标。

结合方差计算结果以及式(3)～(7)，在 MATLAB 中对上述思想进行了仿真验证，得到如图 6 所示的多项式拟合曲线。极值点出现在－0.002 位置，即 y 方向向上偏移了 0.002，需要将原图像向下移动 0.002。x 方向计算过程与 y 方向相同，此处不再赘述。

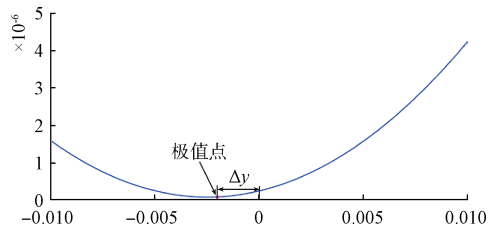

图 6　y 方向多项式拟合曲线

3　仿真验证

Catapult C Synthesis 提供了仿真验证功能，它可以将 C/C++编写的 Testbench 直接转换成 RTL Testbench。同时拥有良好的第三方程序接口，可以直接调用 QuestaSim 进行仿真，在输出信号中给出了 ERR 信号，用于指示 RTL 代码仿真结果与 C/C++代码仿真结果是否一致。

本设计的仿真结果如图 7 所示。输入信号为 pxiel_in_rsc_z、dataflag_rsc_z、clk、rst，其中_rsc_z 为 Catapult C 自

动添加的后缀,clk 和 rst 是 Catapult C 自动添加的信号。输出信号为 y_rsc_z、x_rsc_z、flag_rsc_z。同时程序还给出了 x_ERR＃、y_ERR＃信号,用于指示 C/C＋＋代码与 RTL 代码的仿真结果是否一致。除了设计中的这些输入输出信号外,Catapult C 还给出了一些后缀为_lz 的信号,如 pxiel_in_rsc_lz,该信号用于指示信号是否有效。从图中可以看出,从第二帧开始每帧的结尾给出 x_rsc_z 和 y_rsc_z的计算结果。在第一个 x_rsc_z 和 y_rsc_z 有效数据出现前,x－ERR＃和 y－ERR＃每个时钟周期加 1,当计算出第一个 x_rsc_z 和 y_rsc_z 后,x－ERR＃和 y－ERR＃保持原值,不再累加,说明此时,C/C＋＋代码的仿真结果与 RTL 代码仿真结果一致。

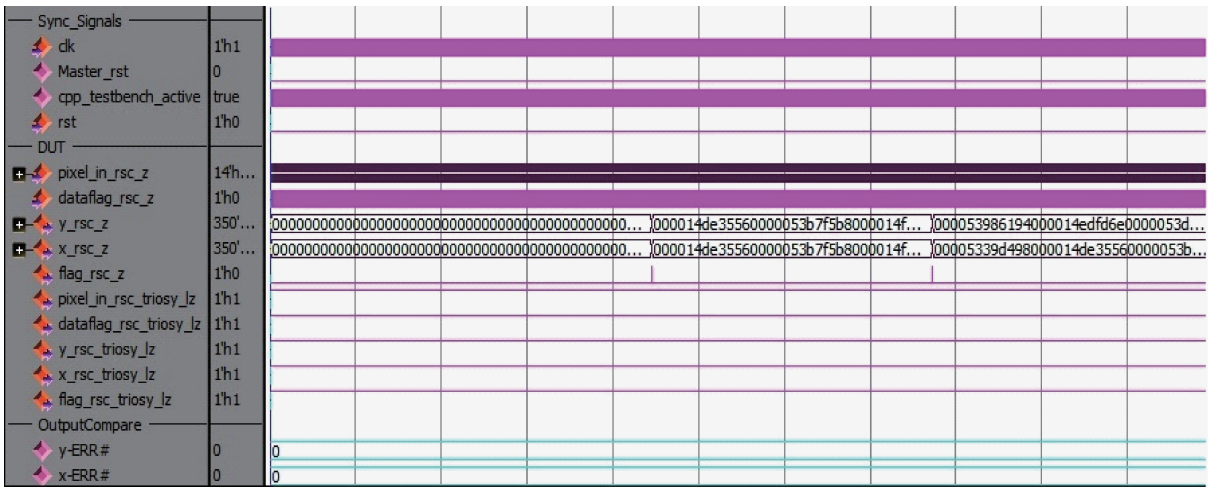

图 7　算法仿真结果

除了仿真测试外,设计的算法程序在遥感相机的视频处理平台上进行了验证测试,测试方法为:在 FPGA 内模拟生成两帧图像数据,前后两帧图像对应像素数据大小相差 1,然后将前后帧数据发送到图像校正算法模块,求出 x 和 y 方向偏移。实际测试结果与仿真结果相同。

4　结　　论

采用 Catapult C Synthesis 在 FPGA 上进行图像校正算法设计,可以直接将 C 语言编写的代码转换成 RTL 代码,并且可以通过添加约束由 Catapult C 自动对面积和时序进行优化,简化了设计流程,降低了设计的难度,缩短了设计周期。本文基于 Catapult C Synthesis 设计的实时图像校正算法,仿真结果正确,性能分析良好,并在硬件平台上进行了验证测试。

参考文献

[1]　李涛. 航天技术发展背景下的嵌入式图像处理技术[J]. 中国新技术新产品,2015(10):6.

[2]　李洋,禹卫东,胡骁,等. 基于 FPGA 的千兆以太网数传系统设计[J]. 电子测量技术,2015,38(10):72.

[3]　杨悦梅,冯冬芹. 基于 FPGA 的高速图像处理技术[J]. 舰船科学技术,2015(4):207.

[4]　冯胜民,陈娟花,曹占山. 基于 FPGA 和 TDC-GP2 的时钟测量系统设计[J]. 国外电子测量技术,2015,34(1):63-64.

[5]　BAILEY D G. Design for Embedded Image Processing on FPGAs[M]. Singapore: John Wiley & Sons (Asia) Pte Ltd,2014: VII-VIII.

[6]　秦贵军. 基于 Catapult C 的汽车轮胎力估计的 FPGA 实现[D]. 长春:吉林大学,2013:40-42.

[7]　刘春香,李宁,石俊霞. TDICCD 相元非均匀性对图像压缩的影响[J]. 电子测量与仪器学报,2015,29(3):375-376.

[8]　秦莉,董丽丽,许文海. CCD 图像灰度与照度的转换标定方法[J]. 仪器仪表学报,2015,36(3):640.

[9]　付石,潘晓璐,李一民. 一种均值滤波的优化算法[J]. 信息技术,2012(3):133-134.

[10]　GONZALEZ R C, WOODS R E. Digital Image Processing[M]. Prentice Hall, 2012:183.

[11]　常锦才,赵龙,杨倩丽. 基于正交多项式的数据拟合方法[J]. 河北理工大学学报,2011,4(33):79-81.

作者简介

李杨,工程硕士,现辽宁广播电视大学讲师,主要研究方向为测试与测量技术等。
E-mail:393205347@163.com