用 Simulink 实现基于 FPGA 的像素流视频图像边缘检测算法①

王水鱼, 王 欣

(西安理工大学 自动化与信息工程学, 西安 710048)

摘 要: 基于 FPGA 实现高清、大容量的视频处理的算法具有一定的复杂性, 为了更好的用 Verilog HDL 描述图像处理算法, 采用了一种在 Simulink 中搭建视频图像处理模型, 利用 MathWorks 最新推出的 Vision HDL Toolbox 进行帧到像素流的转化, 然后再对图像的实现边缘检测, 最后把该算法自动生成 Verilog HDL 代码的方法,通过利用 Simulink 和 ModelSim 进行联合仿真,验证了这种方法的可行性, 即可以快速的生成更加准确 HDL 代码, 提高了用 HDL 描述图像处理算法的速度.

关键词: FPGA; Simulink; Verilog HDL; ModelSim; HDL

Realization of Edge Detection Algorithm Based on FPGA by Simulink

WANG Shui-Yu, WANG Xin

(College of Automatic and Information, Xi'an University of Technology, Xi'an 710048, China)

Abstract: The algorithm based on FPGA to achieve high definition and large capacity video processing has certain complexity. In order to use the Verilog HDL to describe the image processing algorithms better, a set of video image processing model in Simulink is used. MathWorks company's latest Vision HDL Toolbox is used in the conversion from frame to pixel stream, and then image edge is detected. Finally, the feasibility of the method of this algorithm automatically generating Verilog HDL is verified by using Simulink and ModelSim co-simulation. That could quickly generate more accurate HDL code, improving the speed of description of the image processing algorithm using HDL.

Key words: FPGA; Simulink; Verilog HDL; ModelSim; HDL

随着微电子技术的发展, 现在的视频越来越高清, 并且视频图像的尺寸也逐渐增大. 因而大多采用大规模的集成电路或者专用的芯片来实现视频图像的处理. 现场可编程门阵列(FPGA)是现在运用非常广泛的可编程逻辑器件,它不同于 DSP 及其它处理器, 它在设计上可以实现硬件的并行和流水线技术, 并且具很强的灵活性, 可以根据需要实现重新配置, 具有很好地灵活性和适用性[1]. 为了更有效的利用 FPGA 的并行性, 所以视频图像边缘检测采用像素流的处理方法.

对 FPGA 的设计采用硬件描述语言(HDL)实现, 随着视频数据量的增多即算法实现的复杂性, 所以为了快速实现图像处理算法, 则采用基于 Simulink 搭建图像处理模块, 并运用 Simulink 中的 HDL Coder 工具箱自动产生不同于以往的 IP Core 的可读、可跟踪的 HDL 代码, 从而实现视频的快速处理. 由于自动 HDL 代码生成流程比手工编码快, 工程师得以把节省下来的时间投入到更好地方案设计中. 比起手动的设计, 这种方法使工程师能够以更快的速度生成质量更佳的 FPGA 原型. 但是, Simulink 的中大部分的处理都是对大的阵列进行处理, 即对帧的图像处理, 而 FPGA 对图像的处理是基于像素流的处理, 所以在进行图像处理之前把帧数据转换成像素流数据需要人为的编写转换代码, 但是这种方法即麻烦又不可靠. 为了更好更快捷实现 HDL 代码的自生成, 本文先采用 Simulink 的 Vision HDL Toolbox 工具箱把帧数据自动转化成像素流数据作为数据处理的输入量, 然后再在 Simulink

① 收稿时间:2015-08-18;收到修改稿时间:2015-10-08

中设计一个针对产生 HDL 代码的子系统,从而在该子系统中设计基于像素流的边缘检测模块, 在子系统搭建好后就可以利用 Simulink 的 HDL Code 工具自动产生边缘检测算法的 Verilog HDL 代码. 最后通过 Simulink 与 ModelSim 进行联合仿真以来验证这种方法的可行性及利用 QuartusII 验证代码的正确性.

1 边缘检测算法

1.1 图像边缘

图像的边缘是指其周围像素灰度急剧变化的那些像素的集合, 它是图像最基本的特征. 边缘主要存在于目标、背景和区域之间, 因此, 它是图像分割所依赖的最重要的依据, 也是目标区域识别和区域形状提取等图像处理技术的基础. 由于边缘检测十分重要, 因此成为机器视觉研究领域最活跃的课题之一[2].

边缘检测的基本思想是先检测图像中的边缘点, 在按照某种策略将边缘点连接成轮廓, 从而构成分割区域. 由于边缘是索要提取的目标和背景的分界线, 提取出边缘才能将目标个背景区分开, 因此边缘检测对于数字图像非常重要.

1.2 边缘检测算子

基于图像像素灰度值的连续性研究, 通常用图像中某个像素邻域内像素梯度来确定图像的边缘, 梯度对应一阶导数, 梯度算子为一阶导数算子. 对于一个灰度图像函数 $f(x,y)$, 其梯度可表示成一个向量. $\nabla f(x,y)=[Gx/Gy]$, 其中

$$Gx=\frac{\partial f}{\partial x},\quad Gy=\frac{\partial f}{\partial y} \tag{1}$$

从向量分析可知, 梯度向量坐标指向坐标 (x,y) 的灰度值的最大变化率的地方. 向量的幅度和方向角分别为

$$|(\nabla f)|=\sqrt{Gx+Gy}$$

$$\theta=\arctan[Gy/Gx]$$

梯度幅值的计算通常采用计算量更小的绝对值来近似: $|\nabla f|=|G|+|G|$

以上各式的偏导数需要对每个像素的位置计算, 在实际中常用小区域模板进行卷积来近似计算. 对 Gx 和 Gy 各用一个模板, 将两个模板组合起来就构成一个梯度算子, 根据模板的元素值和大小的不同构造不同的算子, 常见的有 Roberts、Sobel、Prewitt、Canny 边缘检测算子等[3]. 对比各个算子后, 本文采用 Sobel 边缘检测, Sobel 算子基于图像的梯度值进行边缘检测, 具有一定的降噪能力, 相较于其他边缘检测算子, Sobel算子运算比较简单, 是最常用的边缘检测算法[4]. 由于本设计采用的是 Simulink 自带的边缘检测模块, 所以不必对 Sobel 算法做过多的研究.

2 Simulink与ModelSim简介

2.1 Simulink

Simulink 是 MATLAB 最重要的组件之一, 是在 MATLAB 中基于帧的视频图形处理的系统仿真软件, 是实现动态系统建模、仿真和分析的集成环境. 在该环境中, 无需大量编写程序代码, 设计的流程在 Simulink 中以图形形式描述算法[5]. Simulink 已被广泛应用于控制理论和数字信号处理的复杂仿真和设计. 同时各种时变系统, 包括通讯、控制、信号处理、视频处理和图像处理系统, Simulink 提供了交互式图形化环境和可定制模块库来对其进行设计、仿真、执行和测试. Simulink 中的 Computer Vision System Toolbox(机器视觉工具箱)工具可以非常方便地进行视频图像的帧处理.

2.2 ModelSim

ModelSim 是 Model Technology(Mentor Graphics 的子公司)HDL 硬件描述语言的仿真工具, 使用该软件可实现对设计的 Verilog HDL、VHDL 或者是两种语言混合的程序进行仿真[6]. ModelSim 是业界最优秀的 HDL 语言仿真器, 具有快速的仿真性能和最先进的调试能力, 支持众多的 FPGA 和 ASIC 厂家库, 是做 FPGA、ASIC 设计的 RTL 级和门级电路仿真的首选. 同是它具有即丰富又易用的图形用户界面, 提供最友好的调试环境, 为加快调试提供强有力的手段[7].

3 边缘检测的实现

3.1 在 Simulink 中实现 Sobel 边缘检测

数字视频是数字多媒体的重要组成部分, 数字视频是数字图像在时间轴上的扩展, 可以将视频的每一帧视为一幅静止的图像. 由此可见, 视频序列图像由一帧一帧具有相互关联的图像构成, 对视频图像的分析处理可以转化成对每一帧的处理[8]. 随着现在视频处理的分辨率越高及视频图像的尺寸越大, 从而对视频处理的难度也随之变高, 同时在 FPGA 上用 Verilog HDL 进行视频图形处理的描述更为复杂.

在这里, 我们采用 Simulink 中的 Comupter Vision System Toolbox(机器视觉工具箱)以及 MathWorks 公司最新推出的 Vision HDL Toolbox 工具箱来搭建本系统设计模型. Comupter Vision System Toolbox 进行图像处理是基于帧的, 但是 FPGA 用 Verilog HDL 描述的处理算法是基于像素的, 因此我们采用 Vision HDL Toolbox 中的 Frame to Pixels 模块进行帧到像素流的转换, 该模块把全帧的视频格式转换成像素流 pixel 和控制总线 ctrl, 用该模块来产生针对 HDL 代码产生的子系统的输入, 需要注意的是该模块并不会产生 HDL 代码. 输出的控制总线 ctrl 与串行像素数据有关, 该总线包含 5 个布尔信号来表示像素的有效性和其在一帧中的位置, 即行的起始像素信号 hStart、行结束像素信号 hEnd、场中第一行的第一个像素信号 vStart、场中最后一行的最后一个像素信号 vEnd 、任何有效像素信号 valid. 这五个信号都是由该模块根据一帧中的像素自行控制, 不需要人为设置, 从而使得自动转换成像素流数据更为方便和准确[9].

接下来在一个子系统中建立实现基于像素流的边缘检测, 命名该子系统为 HDL Algorithm, 在 HDL Algorithm 中建立一个边缘检测模块 Edge Detector, 并在该模块的参数设置框中选择 Method 栏为 Sobel, 阈值为 13. Edge Detector 的输入为经 Frame to Pixels 模块转换后的像素流数据和控制总线信号, 输出则为经边缘检测处理后的像素和控制总线信号. HDL Algorithm 子系统内部的 Edge Detector 模块如图 1 所示.

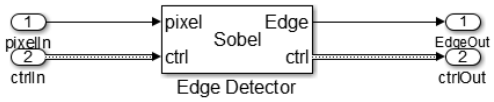

图 1 基于像素流的边缘检测模块

为了验证基于像素流的边缘检测效果与基于帧的处理效果的一致性, 所以搭建的系统包含基于帧和基于像素流的处理, 以便进行对比. 基于帧的边缘检测的阈值也必须和基于像素流处理的阈值一样, 即 13, 最后通过显示模块进行对比. 系统级模型如图 2 所示[10].

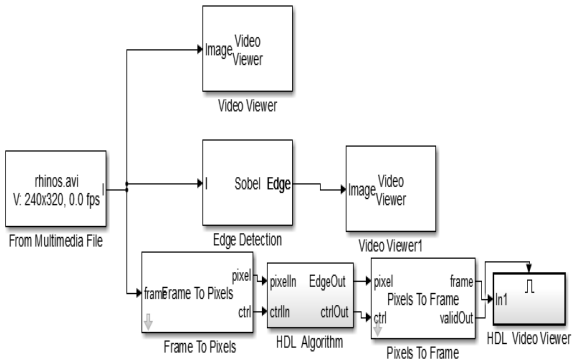

图 2 系统级模型

3.2 结果分析

本设计采用的视频源为 RGB 格式视频, 所以需把 Full Multimedia File 模块的输出图像参数设为灰度图像后进行处理. 最终运行的结果分别如图 3、图 4、图 5 所示.

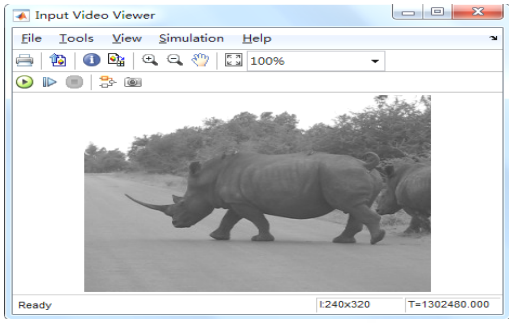

图 3 灰度化后的视频源

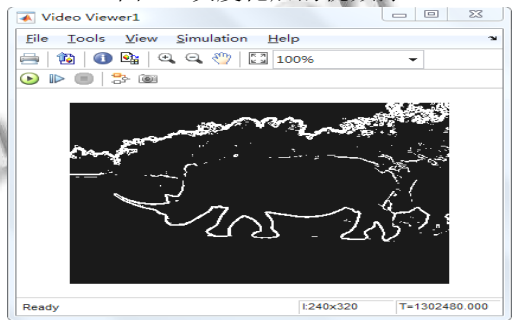

图 4 基于帧的边缘检测

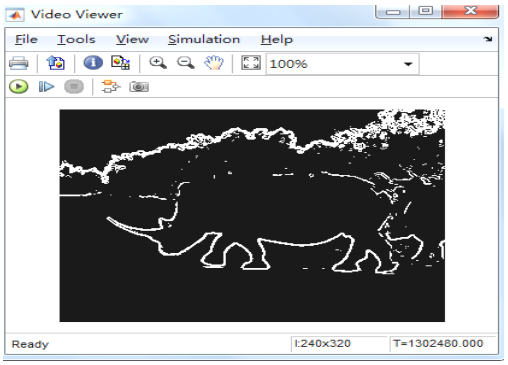

图 5 基于像素流的边缘检测

由以上的运行结果可知, 图 3 为待处理的灰度视频图形, 图 4 为基于帧的边缘检测, 图 5 为基于像素流的边缘检测. 从图 4 和图 5 可以验证把帧转换成像素流进行处理的可行性,
经对比两者的效果完全一致. 从而可以证明在 Simulink 中基于帧的边缘检测与经帧转换成像素流进行边缘检测的效果相同.

3.3 产生 Verilog HDL 代码

根据 Simulink 的用法只能对子模块产生 HDL 代码, 所以必须在建立子系统的子系统中搭建边缘检测模块, 即建立子系统 HDL Algorithm, 并在其内部添加边缘检测模块. 运行成功后将 HDL Algorithm 模块通过设置 HDL Workflow Advisor 从而生成可读的、可追踪的 Verilog HDL 代码. 如图 6 所示.

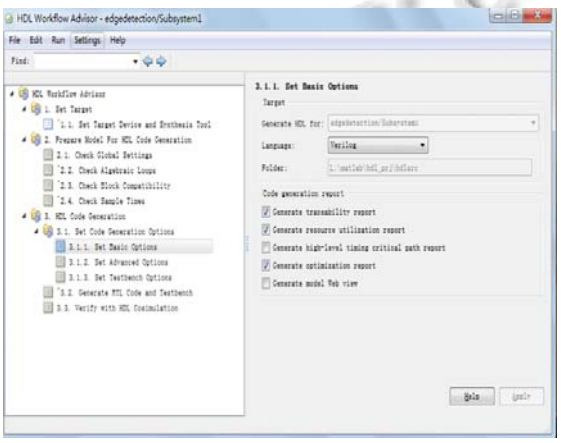

图 6　HDL Workflow Advisor 中设置代码生成

4　仿真验证

由于之前已经验证基于帧的边沿检测与经过帧转换成像素流的边缘检测效果一样, 所以重新搭建一个基于像素流的边缘检测模型, 然后把边沿检测模块自动产生 Verilog HDL 代码. 为了验证自动产生 Verilog HDL 代码与搭建的模型是否一致, 则需要采用 Simulink 与 ModelSim 联合仿真进行 HDL 验证. 在 Simulink 中搭建系统级测试平台, 用系统级平台作为测试的激励, ModelSim 运用 Verilog HDL 代码, 输入为系统级输入, 输出为从 Modelsim 拿出与系统级模型作对比, 从而来验证产生的 HDL 代码与模型的一致性. 系统级测试平台如图 7 所示.

ModelSim 的仿真结果如图 8 所示. 根据 ModelSim 的仿真波形可以观察到: 当 ctrlIn_valid、ctrl_hStart 和 ctrl_vStart 为高电平时, 开始输入经帧转换成像素流的像素, 当 ctrl_hEend 和 ctrl_hVend 为高电平是分别表示一行像素输入完毕和一帧像素输入完毕. 同时, 经边缘检测处理后的边沿输出和控制总线 ctrl 的 5 个信号也完全符合时序. 从而可以证明自动产生 Verilog HDL 代码与搭建的模型是完全一致.

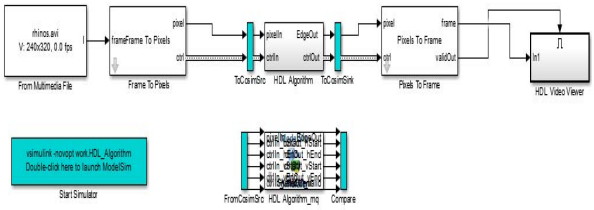

图 7　ModelSim 与 Simulink 联合仿真平台

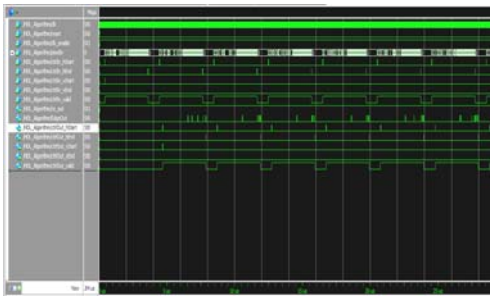

图 8　ModelSim 仿真波形图

最后自动生成的边缘检测模块的 Verilog HDL 代码在 QuartusII 上编译成功,编译报告如图 9 所示. 以及最后生产生成的 RTL 图如图 10 所示.

Flow Summary	
Flow Status	Successful - Sun Aug 16 16:29:59 2015
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Full Version
Revision Name	HDL_Algorithm
Top-level Entity Name	HDL_Algorithm
Family	Cyclone III
Total logic elements	1,167 / 5,136 (23 %)
Total combinational functions	798 / 5,136 (16 %)
Dedicated logic registers	917 / 5,136 (18 %)
Total registers	917
Total pins	23 / 183 (13 %)
Total virtual pins	0
Total memory bits	37,004 / 423,936 (9 %)
Embedded Multiplier 9-bit elements	4 / 46 (9 %)
Total PLLs	0 / 2 (0 %)
Device	EP3C5F256C6
Timing Models	Final

图 9　QuartusII 编译报告

从编译报告可已看出自动生成的边缘检测算法的 Verilog HDL 代码没有错误, 以及从生成的 RTL 图的输入输出口可以看出在时钟和 5 个控制信号的作用下输入 8 位的像素流数据, 经边缘检测处理后输出边缘和 5 个输出控制信号. 并且可以证明通过 Frame to

Pixels 模块的确将帧数据转换成了像素流数据作为边缘检测的输入.

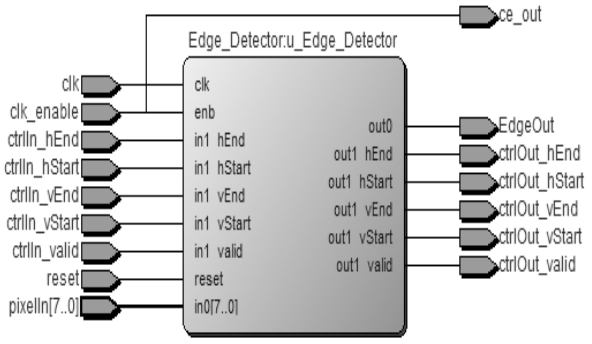

图 10 边缘检测算法的 RTL 图

5 结语

本文在 Simulink 上搭建系统级模型, 通过验证基于帧的边缘检测与通过 Vision HDL Toolbox 工具箱把帧转换成像素流后进行的边缘检测的效果一样后, 令基于像素流的边缘检测算法自动生成可读的、可追踪的 Verilog HDL 代码. 最终利用 Simulink 和 ModelSim 联合仿真来验证产生的 Verilog HDL 代码与算法模块的一致性, 以及利用 QuartusII 验证算法代码的正确性. 采用这种方法可以快速生成视频图像处理算法的 HDL, 与传统人为编写图像帧转换成像素流的代码和图像处理代码相比, 这种方法更快、更准确. 在如今的大容量、高清视频处理中更能体现这一方法的价值.

参考文献

1 侯法柱.基于 FPGA 的图像采集与处理系统设计[硕士学位论文].长沙:湖南大学,2010.
2 MATLAB 技术联盟 张岩. MATLAB 图像处理超级学习手册,北京:人民邮电出版社, 2014:269−271.
3 熊晓薇.基于 FPGA 的视频图像处理的研究与实现[硕士学位论文].石家庄:河北大学, 2014.
4 杨光耀.基于 FPGA 的图像采集及处理系统设计[硕士学位论文].呼和浩特:内蒙古大学, 2014.
5 GBailey D.原魁,何文浩,肖晗译.基于 FPGA 的嵌入式图像处理系统设计.北京:电子工业出版社,2013:70−71.
6 华清远见嵌入式培训中心.FPGA 应用开发入门与典型实例.北京:人民邮电出版社,2008:243−244.
7 葛亚明,彭永丰,薛冰等.零基础学 FPGA 基于 Altera FPGA 器件&Verilog HDL 语言.北京:机械工业出版社,2010:176−177.
8 赵小川,何灏,缪远诚等.MATLAB 数字图像处理实战.北京:机械工业出版社,2013:148−149.
9 Frame To Pixels User's Guide. MathWorks Inc, 2015.
10 Edge Detection and Image Overlay User's Guide. MathWorks Inc, 2015.