

文章编号:1007-2780(2015)01-0163-07

基于 Wu 反走样的三角形光栅化边缘反走样算法

吴连慧,周建江*,夏伟杰,陈雅雯
(南京航空航天大学 电子信息工程学院,江苏 南京 210016)

摘要:对于光栅化的三角形,其边缘存在明显的锯齿现象,因此需要进行反走样处理。基于 Wu 直线反走样算法的思想,考虑了水平直线外侧直接添加插值点的算法和三角形三边外侧反走样的算法。综合两者的优点,在绘制光栅化的三角形的同时,边缘叠加 Wu 反走样直线,并考虑背景像素灰度值的作用。结果表明,改进的三角形光栅化的边缘反走样算法有效提高了三角形光栅化后边缘的显示效果,该算法计算量小,便于 FPGA 实现,可用于对图形显示质量要求很高的机载显示系统。

关键词:反走样;Wu 算法;光栅化;FPGA
中图分类号:TP391.4 **文献标识码:**A **doi:**10.3788/YJYXS20153001.0163

Edge anti-aliasing algorithm of triangle rasterization based on Wu algorithm

WU Lian-hui, ZHOU Jian-jiang*, XIA Wei-jie, CHEN Ya-wen
(College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract: Anti-aliasing of the triangle rasterization whose edge has ladder phenomenon obviously is proposed in this paper. The method of adding interpolation points outer the horizontal and the method of anti-aliasing line unilateral are discussed, based on Wu anti-aliasing algorithm. The advantages of both methods are considered in order to draw triangle rasterization. Wu anti-aliasing line is overlaid for the edge at the same time. And the gray value of background pixel is under consideration. The result shows that the improved edge anti-aliasing algorithm of triangle rasterization effectively improves the display of the triangle rasterization edge. The principle is simple and it is easy to implement on FPGA. Therefore the improved algorithm can be applied to cockpit display system which needs high display quality urgently.

Key words: anti-aliasing; Wu algorithm; rasterization; FPGA

1 引言

在机载显示系统中,图形显示占据重要地位。传统的机载图形显示大多通过处理器(CPU)和软件方法实现。为了进一步提高图形的实时性,考虑到FPGA具有强大的逻辑资源和丰富的IP核,利用FPGA来代替CPU完成图形处理的大量显示运算任务,实现图形硬件加速。

三角形光栅化^[1-2]是图形光栅化模块中重要组成部分,许多复杂多边形的光栅化都可以在CPU端分解为多个三角形光栅化命令,通过调用三角形光栅化模块来实现。由于三角形光栅化的边缘会出现明显锯齿现象,因此需要对其进行反走样处理。

本文基于水平线扫描转换设计了一种快速的三角形光栅化硬件加速算法。同时利用Wu直线反走样思想,研究三角形边缘反走样,提出了一种计算量小、显示效果良好的边缘反走样算法。并在FPGA上实现了该算法。

2 三角形光栅化硬件加速算法

图形加速系统框图如图1所示,用户接口模块接收上层CPU解析的图形命令,由图形光栅化模块来实现基本图元光栅化,并通过缓存管理模块将其存储到外部存储器中。

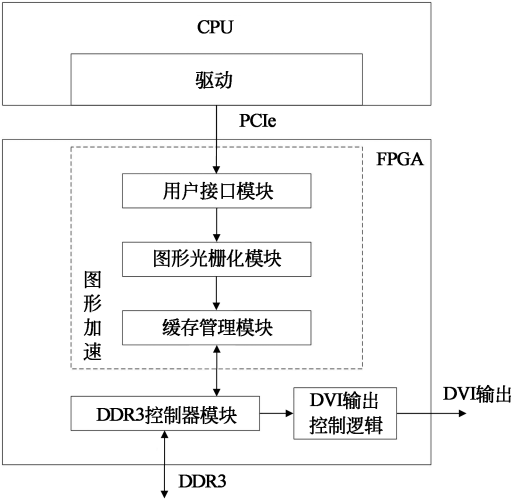


图1 图形加速系统框图

Fig. 1 Diagram of graphics acceleration

光栅化是指把物体的数学描述及其相关的颜色信息转换为屏幕上的像素的过程。三角形光栅化算法思想是,将三角形分解为水平直线顶点信息;再调用水平直线光栅化模块进行逐行扫描转换,从而实现光栅化。难点在于如何将三角形依次分解为水平直线端点。

三角形光栅化示意图如图2所示。步骤如下:

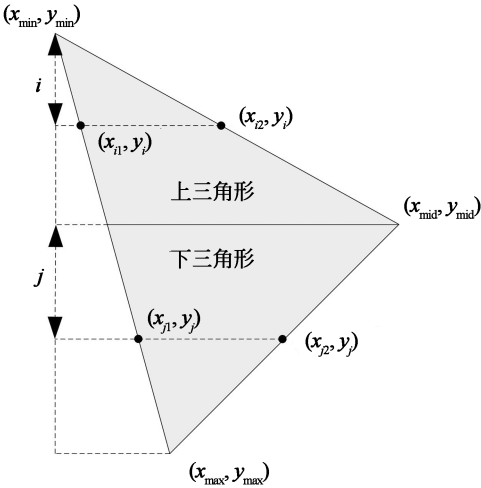


图2 三角形光栅化示意图

Fig. 2 Schematic of triangle rasterization

(1)根据三角形顶点的纵坐标将3个顶点从小到大重新排列,记作 (x_{min}, y_{min}) , (x_{mid}, y_{mid}) , (x_{max}, y_{max}) ,其中 $y_{min} \leq y_{mid} \leq y_{max}$,横坐标之间不存在大小关系。根据 y_{mid} 所在水平直线将普通的三角形分解成上三角形和下三角形。同时计算出3条边的斜率:

$$\begin{cases} S_1 = (x_{max} - x_{min}) / (y_{max} - y_{min}) \\ S_2 = (x_{mid} - x_{min}) / (y_{mid} - y_{min}) \\ S_3 = (x_{mid} - x_{max}) / (y_{mid} - y_{max}) \end{cases} \quad (1)$$

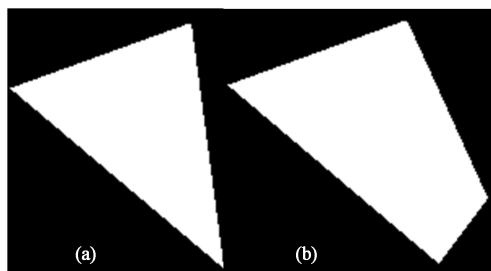
(2)对上三角形进行水平直线端点的分解,其中 i 是从1到 $(y_{mid} - y_{min})$ 依次遍历的整数,则:

$$\begin{cases} x_{i1} = x_{min} + S_1 \times i \\ x_{i2} = x_{min} + S_2 \times i \\ y_i = y_{min} + i, 1 \leq i \leq (y_{mid} - y_{min}) \end{cases} \quad (2)$$

(3)对下三角形进行水平直线端点的分解,其中 j 是从0到 $(y_{max} - y_{mid} - 1)$ 依次遍历的整数,则:

$$\begin{cases} x_{j1} = x_{min} + S_1 \times (y_{mid} - y_{min}) + S_1 \times j \\ x_{j2} = x_{mid} + S_3 \times j \\ y_j = y_{mid} + j, 0 \leq j < (y_{max} - y_{mid}) \end{cases} \quad (3)$$

图 3 为三角形和四边形光栅化 Matlab 仿真图,可以看出,两者的边缘存在明显锯齿状的失真现象,所以需要进行反走样处理。



(a) 三角形光栅化 (b) 四边形光栅化

(a) Triangle rasterization (b) Quadrangle rasterization

图 3 光栅化仿真图

Fig. 3 Simulation of rasterization

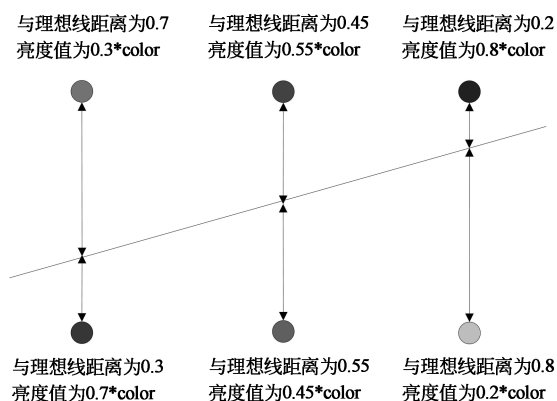
图 4 W_u 直线反走样算法示意图

Fig. 4 Schematic of Wu anti-aliasing algorithm

3.2 水平直线外侧直接添加插值点

由于三角形光栅化是基于水平直线扫描的,考虑每次生成水平直线时,直接利用 Wu 反走样的思想对水平直线端点进行反走样处理。

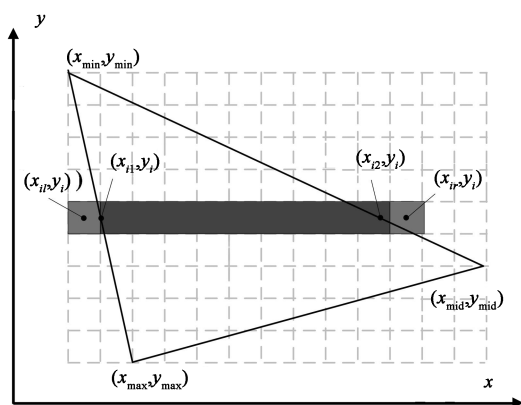


图 5 水平直线外侧添加插值点示意图

Fig. 5 Schematic of adding interpolation points
outer the horizontal line

图5为水平直线外侧直接添加插值点示意图。图中以上三角形水平直线外侧添加插值点为例,水平直线左侧的插值点坐标为 (x_{il}, y_i) ,对应的像素点的亮度值表示为 $G(x_{il}, y_i)$,且插值点坐标 (x_{il}, y_i) 到浮点坐标 (x_{il}, y_i) 的距离表示为 dl_i 。则:

$$dl_i = x_{i1} - |x_{i1}|. \quad (5)$$

$$G(x_{ij}, y_i) = \text{color} \times (1 - dl_i), \quad (6)$$

其中, $\lfloor \cdot \rfloor$ 表示向下取整。水平直线右侧的插值点坐标为 (x_{ir}, y_i) , 对应的像素点的亮度值表示为 $G(x_{ir}, y_i)$, 且插值点坐标 (x_{ir}, y_i) 到浮点坐标

3 基于 W_u 算法的边缘反走样

常见的三角形光栅化反走样算法有超级采样、多级采样和自适应采样等^[3],共同点是通过采样更多的子像素来反走样,在提高图形质量的同时,花费了较大的资源代价和时间消耗。考虑三角形内部像素点完全被三角形像素区域覆盖,不需要反走样,因此只需要对三角形边缘区域进行反走样,而三角形边缘是三条直线,因此考虑可以借鉴直线反走样算法对三角形边缘进行反走样处理。

3.1 Wu 直线反走样算法

Wu 直线反走样算法^[4]是一种基于区域采样原理的简单有效且硬件易实现的直线反走样算法,其算法思想是:沿着长轴方向每前进一个像素单位,在短轴方向上距离理想直线距离最近的两个像素都点亮,两者亮度之和等于待绘制直线的亮度值,且距离理想直线远的亮度低,距离近的亮度高。Wu 反走样算法原理如图 4 所示。

假设绘制颜色的亮度值为 color , 相邻像素坐标距离为 1, 若坐标 (x, y) 在从轴方向上与理想线段之间的距离为 $d (0 \leq d \leq 1)$, 坐标 (x, y) 上像素点的亮度值表示为 $G(x, y)$, 则:

$$G(x, y) = \text{color} \times (1 - d). \quad (4)$$

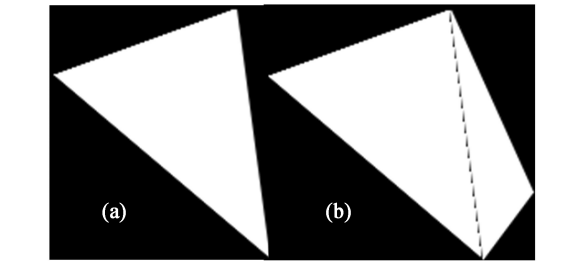
Wu 反走样算法降低了直线光栅化时候的锯齿现象。借鉴这种算法思想对三角形边缘单独处理。

(x_{i2}, y_i) 的距离表示为 dr_i 。则：

$$dr_i = \lfloor x_{i2} + 1 \rfloor - x_{i2}, \tag{7}$$

$$Gr_i = \text{color} \times (1 - dr_i), \tag{8}$$

图 6 为三角形和四边形光栅化时水平直线外侧添加插值点仿真图。从图中可以看出当斜率 k 较大 ($|k| > 1/2$) 时, 图形边缘反走样效果较好; 但图形边缘斜率 k 较小 ($|k| \leq 1/2$) 时, 反走样效果不好; 且用三角形拼接成多边形填充时, 拼接处会有虚化现象。不同斜率显示效果有差异的原因是, Wu 算法是分长轴短轴的, 按照长轴遍历和按照短轴遍历, 效果是不一样的。而三角形分解成的水平直线是统一在水平直线端点两侧添加插值点的, 本身不好区分长短轴, 因此会影响反走样效果。三角形拼接处出现虚化现象是因为拼接处没有判断出是多边形内部的值, 反而按照三角形边缘进行了插值处理, 既浪费了时间又影响显示效果。



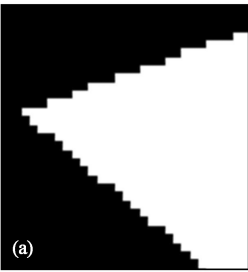
(a) 三角形反走样 (b) 四边形反走样
(a) Anti-aliasing of triangular (b) Anti-aliasing of quadrangular

图 6 水平直线外侧添加插值点仿真图
Fig. 6 Simulation of adding interpolation points outer the horizontal

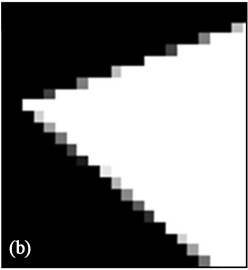
水平直线外侧直接添加插值点的方法比较简单, 但缺点是, 没有考虑长短轴的问题, 当斜率较小时, 效果不好; 且拼接成多边形时, 拼接处会有虚化现象。

3.3 三角形三边外侧反走样

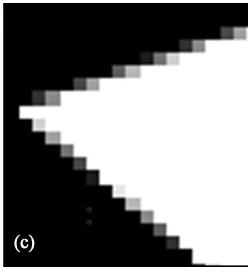
水平直线外侧添加插值点的方法是对每次分解的水平直线端点进行处理, 而不是将三角形 3 条边缘看作 3 条直线单独处理。考虑直接将其当作 3 条直线用 Wu 算法对直线进行单侧的反走样处理。图 7 为不反走样、水平直线外侧添加插值点、三边外侧反走样对比仿真图。从图中可以看出, 当直线与水平线夹角在 $\pm 30^\circ$ 之内时, 该直线外侧反走样插值效果比水平直线外侧直接添加插值点效果好; 其他情况下两者的效果相同。



(a) 不反走样
(a) Rasterization only



(b) 水平直线外侧添加插值点
(b) Adding interpolation points outer the horizontal



(c) 三边外侧反走样
(c) Anti-aliasing line unilateral

图 7 反走样前后仿真对比图

Fig. 7 Comparison of anti-aliasing effect

最终显示效果与图 8 右侧所示效果相同, 三角形三边外侧反走样优点是显示质量好。缺点是, 对于 3 条边本身来说, 已经脱离了三角形的关系, 很难判断哪边是外侧; 对于同一条边, 不同的三角形内外侧也不尽相同; 对于同一条边, 端点的顺序不同, 内外侧是相反的。因此, 对三角形三条边单独进行单侧反走样判断情况过于复杂。

4 改进的边缘反走样算法

分析上述两种方法的优缺点如表 1 所示。综合两者的优点, 提出一种算法能够有较好的显示效果, 同时实现简单且没有虚化现象。

表 1 边缘反走样算法优缺点对比

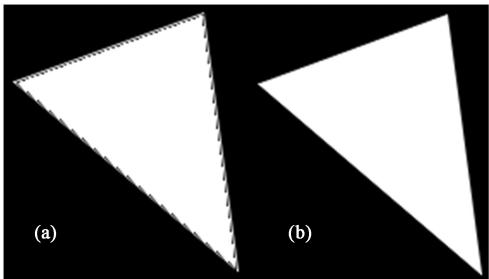
Tab. 1 Advantages and disadvantages of different edge anti-aliasing algorithms

	水平线外侧加插值点	三边外侧反走样
优点	方法简单	显示质量好
缺点	没有考虑长短轴的问题,当斜率较小时,效果不好;且拼接成多边形时会有虚化现象。	对于三条边本身来说,很难判断哪边是外侧,判断情况过于复杂。

4.1 算法思想

4.1.1 边缘叠加 Wu 反走样直线

由于三角形三边外侧反走样效果好,但对于边缘直线本身,内外侧的判断比较复杂,所以考虑直接在三角形光栅化的基础上进行 3 条边的反走样绘制。直线和三角形命令都是由上层 CPU 发送的,但底层 FPGA 并行处理时,先后顺序是不定的。如果先绘制三角形填充再绘制直线,效果如图 8(a)所示;如果先绘制直线,再绘制三角形填充,效果如图 8(b)所示。从图 8(b)看出,此方法效果比水平直线添加插值的方法好,与三边外侧反走样效果相同,但不需要进行内外侧的判断。考虑如何解决叠加顺序不定带来的影响。



(a)先绘制三角形填充 (b)先绘制反走样直线
(a)drawing triangular first (b)drawing anti-aliasing line first

图 8 边缘叠加 Wu 反走样直线仿真图

Fig.8 Simulation of overlaying Wu anti-aliasing line for the edge

4.1.2 插值考虑背景值

为了解决叠加顺序不同带来的显示效果差异,在用单线宽直线反走样对边缘处理时,考虑当前坐标的背景颜色值对插值后颜色值的影响。假设坐标 (x,y) 在从轴方向上与理想线段之间的距离为 $d(0\leq d\leq 1)$,坐标的背景颜色亮度为 $GB(x,y)$,当

前图形颜色值为 G ,若坐标 (x,y) 上像素点的亮度值表示为 $G(x,y)$,则:

$$G(x,y) = G \times (1 - d) + GB(x,y) \times d. \quad (9)$$

先绘制直线,再绘制三角形填充时,在两者重叠的坐标处,将三角形填充颜色值 color 直接写入到当前坐标对应的 DDR3 地址中,覆盖掉之前的插值结果。最终显示效果与图 8(b)相同。

先绘制三角形填充,再绘制直线时,在两者重叠的坐标处,将背景值 color 从 DDR3 对应地址中读回,代入公式进行计算,则

$$G(x,y) = G \times (1 - d) + GB(x,y) \times d = color \times (1 - d) + color \times d = color. \quad (10)$$

从公式可以看出,对于颜色值为 color 时,重叠部分读回背景值也是 color,最终结果还是 color,不会出现虚化现象。最终显示效果也与图 8(b)相同。

当前方法效果好,简单易行,且不反走样三角形和反走样边界直线的顺序是任意的。多边形光栅化时,只需要发送多边形的边缘直线进行反走样,而不需要将分解的三角形各边都进行反走样。

此外,由于三角形光栅化是图形加速模块的子模块,图形加速模块也包括基本的反走样直线,可以调用现有的反走样直线算法,不需要另外编写程序。

4.2 FPGA 仿真和硬件实现

在 Xilinx 的 Kintex-7 FPGA 上对不反走样三角形填充和边缘反走样进行 ModelSim 仿真^[5]和 FPGA 硬件实现,对比边缘反走样前后的三角形光栅化效果。

4.2.1 ModelSim 仿真

三角形填充仿真时,令输入坐标为(280, 100);(300,250);(170,140),对应的 Matlab 仿真效果如图 3(a)所示。每解析出一条水平直线端点信息,就调用水平直线光栅化程序绘制出来。从图 9 可以看出,绘制时,水平直线的长度由短到长,后又由长到短。本文系统输出分辨率为 1 920×1 080,帧速率为 60 Hz,时钟频率为 400 MHz。用新算法绘制该三角形光栅化时,耗时约 33.75 μs。三条边反走样绘制,耗时约 2.16 μs。相对于三角形光栅化消耗的时间,反走样处理占用时间少,且效果改进明显。

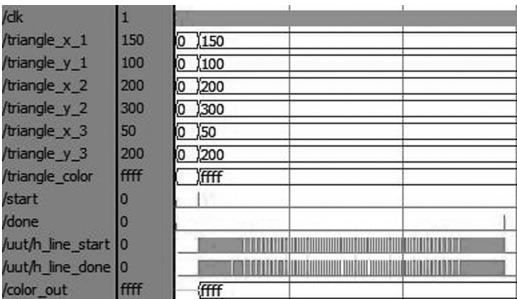
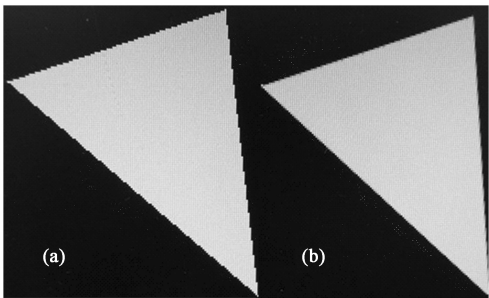


图 9 三角形光栅化 ModelSim 仿真图

Fig. 9 Simulation of triangle rasterization by ModelSim



(a) triangle rasterization (b) edge anti-aliasing

图 10 三角形光栅化及其反走样屏幕显示对比图

Fig. 10 Display comparison of triangle rasterization and its anti-aliasing

4. 2. 2 FPGA 硬件实现

新算法由于计算量小,适合于硬件实现。三角形光栅化及其反走样显示效果如图 10 所示。可以看到反走样后显示效果比不反走样效果好。且用新算法绘制耗时远低于一帧图像的最大处理时间(16.7 ms),满足系统设计要求,故可以用于机载图形显示系统。

5 结 论

介绍了三角形光栅化硬件加速算法,由于其边缘存在明显锯齿状的失真现象,所以需要进行反走样处理。基于 Wu 反走样算法的思想,考虑了水平线外侧加插值点和三边外侧反走样直线的算法。综合两者的优缺点,提出了一种改进的边缘反走样算法。该算法在绘制光栅化的三角形的同时,叠加考虑背景像素作用的 Wu 反走样直线。且用于多边形光栅化时只需要发送多边形的边缘直线进行反走样而不需要将分解的三角形各边都进行反走样。后续研究中可以进一步改进反走样直线的算法^[6-7],从而改进三角形反走样的效果。

经过对比,本文提出的三角形光栅化的边缘反走样算法有效提高了三角形光栅化后边缘的显示效果,计算量小、效果好,且易于 FPGA 等硬件实现。

参 考 文 献:

[1] 黄锐,付宇卓,赵峰. 基于 FPGA 的三角形光栅化模块[J]. 计算机工程, 2008, 34(22): 242-244.
Huang R, Fu Y Z, Zhao F. FPGA-based triangle rasterization module [J]. *Computer Engineering*, 2008, 34(22): 242-244. (in Chinese)

[2] 饶志恒. 图形处理器图形管线研究与实现[D]. 长沙: 湖南大学, 2011.
Rao Z H. Research and implementation on graphics pipeline of graphic processing unit [D]. Changsha: Hunan University, 2011. (in Chinese)

[3] 黄锐. 实时嵌入式图形系统中的三角形光栅化研究与设计[D]. 上海: 上海交通大学, 2008.
Huang R. Study and design of triangle raster in real-time embedded graphics system [D]. Shanghai: Shanghai Jiao-tong University, 2008. (in Chinese)

[4] Wu X. An efficient anti aliasing technique [J]. *Computer Graphics*, 1991, 25(4): 143-152.

[5] 王文华, 何斌, 任建岳. 线阵 CCD 成像系统自校图形设计[J]. 光学精密工程, 2009, 17(8): 2011-2016.
Wang W H, He B, Ren J Y. Design of self-check figures in linear CCD imaging system [J]. *Optics and Precision Engineering*, 2009, 17(8): 2011-2016. (in Chinese)

-
- [6] Xian Z, Xiaobing L. Improved DDA line drawing anti-aliasing algorithm based on embedded graphics system [C]. 2010 *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, IEEE, 2010, 2: V2-497-V2-499.
- [7] Yong J, Lu-ya W, Yue-yue C. The research and implementation of line anti-aliasing algorithm based on coordinate system rotation [C]. 2010 *International Conference on Audio Language and Image Processing (ICALIP)*, IEEE, 2010: 180-184.

作者简介:吴连慧(1989—),女,江苏扬州人,硕士研究生,主要从事信号获取与处理、数字图像处理、FPGA 应用方面的研究。E-mail:wulianhui0208@gmail.com