利用verilog语言实现图像的旋转（256\*256bmp）

（代码测试通过）

学校： 鸿蒙大学子虚学院

系别： 乌有系

专业： 梦语天涯专业

班级： x班

姓名：落花秋雨、青蛙不戒

指导教师： 步青云 完成时间：2013.12.16

目 录

[verilog语言描述电路实现bmp文件的翻转 1](#_Toc375000934)

[一、课题复述 1](#_Toc375000935)

[二、问题分析 1](#_Toc375000936)

[三、设计说明 1](#_Toc375000937)

[3.1将bmp文件存入verilog中的存储器mem 1](#_Toc375000938)

[3.2 bmp文件的存储 2](#_Toc375000939)

[3.3引入翻转函数 2](#_Toc375000940)

[3.4 文件头的保护 3](#_Toc375000941)

[四、实验内容 3](#_Toc375000942)

[4.1实现原文件的显示 3](#_Toc375000943)

[4.2实现图像的翻转 4](#_Toc375000944)

[4.3实验结果数据还原为图像 5](#_Toc375000945)

[五、实验心得体会 5](#_Toc375000946)

[六、参考文献 5](#_Toc375000947)

[七、附 录 5](#_Toc375000948)

[7.1bmp文件的存储格式 5](#_Toc375000949)

# verilog语言描述电路实现bmp文件的翻转

## 一、课题复述

用verilog语言描述电路，实现把lena图像（也可以是其他图像）（256\*256像素）顺时针旋转90度。要求分组完成3-4人一组,自由组队，至少三人，不可独立完成，自行合理安排组内每个成员的工作，工作量大致适中。抄袭判不及格。

考试之前打印上交：原代码、仿真结果、设计说明书、工作报告（包括每个人的工作总结）。第17周之后考试之前上交。

原始图像如果不是256\*256\*24bit格式的BMP图片，首先用画图工具裁剪成256\*256的大小\*24bit格式。

具体方法不特别规定，但至少分成两个模块来实现。

## 二、问题分析

我们需要了解BMP位图的存储格式（详见附录）

首先用$fread系统任务将BMP图片的二进制代码读入存储器meme中；由initial子语句产生clk时钟信号; 由于读出的二进制代码的前54个字节是BMP位图的基本代码，因此在对图片进行翻转时前54个字节的位置不变。由于文件格式为256\*256\*24bit文件,所以后面为256\*256个像素，每24位代表一个像素，因为BMP文件的数据是从左到右，从下到上的。也就是说，从文件中最先读到的是图象最下面一行的左边第一个象素，然后是左边第二个象素……接下来是倒数第二行左边第一个象素，左边第二个象素……依次类推 ，最后得到的是最上面一行的最右一个象素。所以要实现图片的顺时针旋转应该把meme中像素逆时针旋转90度。

最后用WinHex软件把得到的lena.txt文件转换成二进制数据另存为.Bmp文件即可得到翻转后的图片。

应用软软件: ModelSim SE 6.1d WinHex

## 三、设计说明

### 3.1将bmp文件存入verilog中的存储器mem

我们将要对256\*256\*24bit bmp图像进行旋转，定义一个存储器mem，其深度为24位，那么存储地址就为0~65553。

|  |  |
| --- | --- |
| mem | 地址 |
| 0x000 | 6 |
| 0x000 | 5 |
| 0x000 | 4 |
| 0x300 | 3 |
| 0x000 | 2 |
| 0xd36 | 1 |
| 0x434 | 0 |

将mem后65536位视为二维数组，存储器中数据可以视为256\*256矩阵。

### 3.2 bmp文件的存储

bmp文件的显示：（以3\*3数组为例见左下图）

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

|  |  |  |
| --- | --- | --- |
| 6 | 7 | 8 |
| 3 | 4 | 5 |
| 0 | 1 | 2 |

根据bmp图像存储格式该文件在存储器中的存储顺序为：（见右上图）

将bmp文件顺时针旋转90。后图像为 左下图

|  |  |  |
| --- | --- | --- |
| 6 | 7 | 8 |
| 3 | 4 | 5 |
| 0 | 1 | 2 |

|  |  |  |
| --- | --- | --- |
| 8 | 5 | 2 |
| 7 | 4 | 1 |
| 6 | 3 | 0 |

而旋转后的图像存储信息为右上图

从图上观察，可发现将图片顺时针旋转90。也就是将存储器中的数据逆时针旋转90。。

### 3.3引入翻转函数

将3\*3数组进行标注

|  |  |  |  |
| --- | --- | --- | --- |
| 地址j i | 0 | 1 | 2 |
| 0 | 0 | 1 | 2 |
| 1 | 3 | 4 | 5 |
| 2 | 6 | 7 | 8 |

mem：012345678

原地址=3\*i+j

旋转后：

|  |  |  |  |
| --- | --- | --- | --- |
| 地址j i | 0 | 1 | 2 |
| 0 | 2 | 5 | 8 |
| 1 | 1 | 4 | 7 |
| 2 | 0 | 3 | 6 |

mem：258147036

新地址=3\*j+(2-i)

同理将此函数应用于256\*256 bmp mem数组

原地址=256\*i+j 即该点所对应的新地址为256\*j+255-i。

新地址=256\*j+255-i。

### 3.4 文件头的保护

将数据的操作进行在地址的后65536位上。将前18位0~17位隔离，进行数据保护。

即mem[0:17] 为文件头数据、mem[18~65553]为bmp像素数据。

原地址为：256\*i+j+18

新地址为：18+256\*j+255-i

## 四、实验内容

我们以下图进行上述实验：（为了便于分辨将边缘部分加了阴影、四周边框为白色）



### 4.1实现原文件的显示

源程序：

module LENA;

parameter size=65554;

reg[23:0] mem[0:size-1],date;

integer file;

integer i,k,j,m,n,p;

initial

begin

i=$fopen("lena.bmp","rb");

j=$fread(mem,i);

$fclose(i);

end

initial

begin

#100;

file=$fopen("lena.txt");

for(p=0;p<=17;p=p+1)

begin

date=mem[p];

$fwrite(file,"%h",date);

end

for(k=0;k<=255;k=k+1)//行

for(m=0;m<=255;m=m+1)//列

begin

n=18+256\*k+m;

date=mem[n];

$fwrite(file,"%h",date);

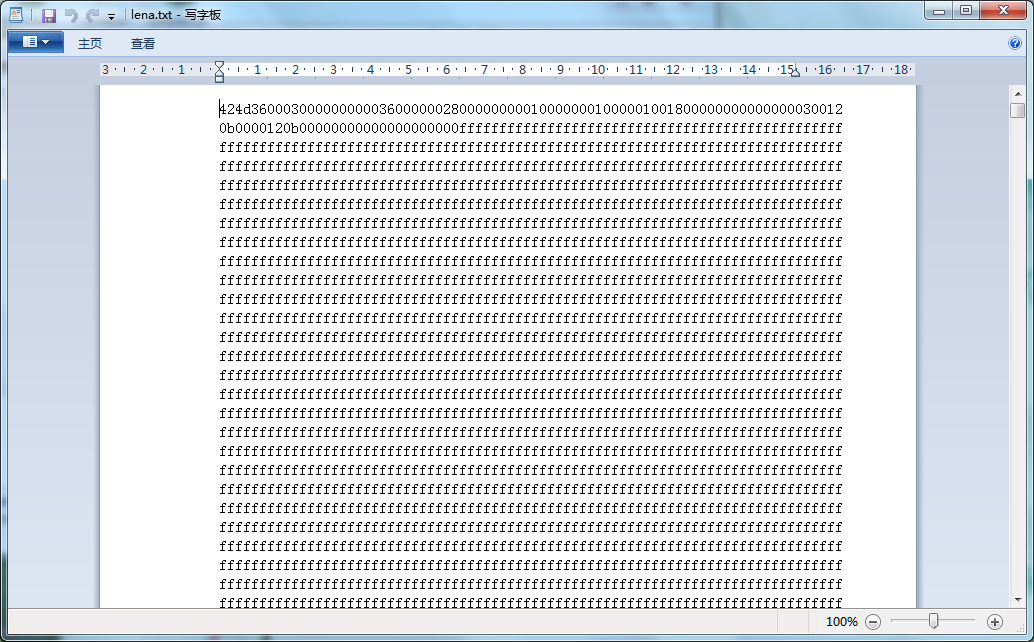
end

$fclose(file);

end

endmodule

结果为：



### 4.2实现图像的翻转

源程序为：

module LENA;

parameter size=65554;

reg[23:0] mem[0:size-1],date;

integer file;

integer i,k,j,m,n,p;

initial

begin

i=$fopen("lena.bmp","rb");

j=$fread(mem,i);

$fclose(i);

end

initial

begin

#100;

file=$fopen("lena.txt");

for(p=0;p<=17;p=p+1)

begin

date=mem[p];

$fwrite(file,"%h",date);

end

for(k=0;k<=255;k=k+1)//行

for(m=0;m<=255;m=m+1)//列

begin

n=18+256\*m+255-k;

date=mem[n];

$fwrite(file,"%h",date);

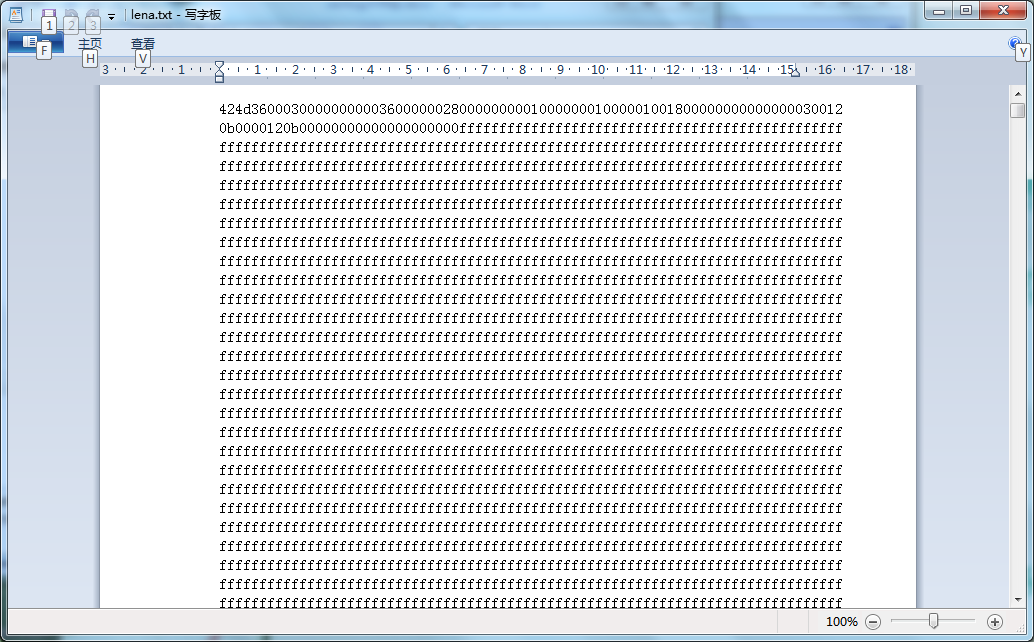
end

$fclose(file);

end

endmodule

结果为：



输出为二进制数据

module LENA;

parameter size=65554;

reg[23:0] mem[0:size-1],date;

integer file;

integer i,k,j,m,n,p;

initial

begin

i=$fopen("lena.bmp","rb");

j=$fread(mem,i);

$fclose(i);

end

initial

begin

#100;

file=$fopen("lena.txt");

for(p=0;p<=17;p=p+1)

begin

date=mem[p];

$fwrite(file,"%b",date);

end

for(k=0;k<=255;k=k+1)//行

for(m=0;m<=255;m=m+1)//列

begin

n=18+256\*m+255-k;

date=mem[n];

$fwrite(file,"%b",date);

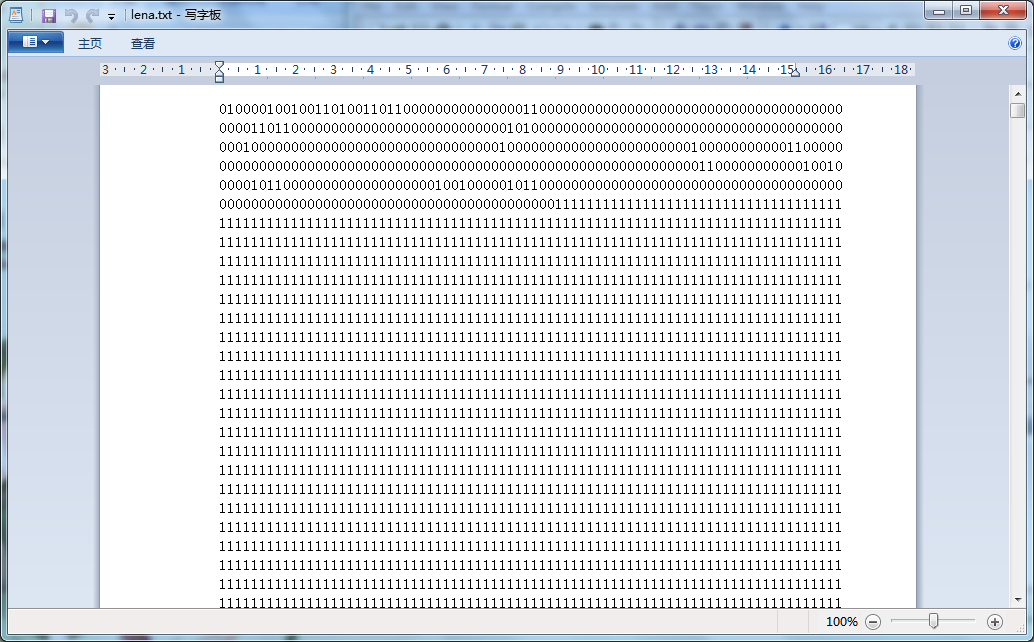
end

$fclose(file);

end

endmodule

数据为：



### 4.3实验结果数据还原为图像



## 五、实验心得体会

青蛙不戒:

经过我们团队的努力完成了大作业，我的收获很多，主要有以下几点：

 1.这次的大作业是次开放性的大作业，通过查阅相关资料结合所学过的知识来设计程序完成对图像顺时针旋转90度。通过这次大作业，对modelsim仿真软件的使用流程有了最基本的了解，使课堂上学的东西得到了很好的实践，在大作业完成之际也体会到了成功的喜悦。

2.对知识的掌握不只是学习课本上的知识，而且应该查阅大量相关的资料。查阅资料是现代科技人员很重要的一项技能，在这次大作业中尤其体现明显。用到的很多知识都不是从课本上学来的，但是要完成大作业必须要查阅大量资料。比如bmp图片的相关知识，只有通过查阅资料才能够掌握从而为自己所用。

3.学以致用的重要性：学的很多知识不运用不能是真正的掌握，这一点在我身上很是有所体现。比如说算图像顺时针转移的算法时，就相当于对一个矩阵中的数据的为之进行适当的转换，但是我的算法出了几次问题，在最后快要成功时功亏一篑，对团队的工作产生了比较大的影响。对此我很惭愧，这就是以前学习的知识不能够很好的应用，对于这一点我应该加以相当的重视

这次的大作业是次很好的实践。通过自己设计，大家都很有收获，不仅完成了作业，而且学到了知识，关键的是自己的自学能力有所加强，所以希望以后还有更多实践的机会，这对于我们的提高大有好处，因为时间是检验真理的唯一标准。这次机会也是我们的一次宝贵经验，对以后的学习工作有很好的借鉴作用。

落花秋雨：

有过这一次经历，让我体会到团队的力量是伟大的！从无到有、化难为易。从一无所知、到最终解决。在其中、我们争过、吵过、烦心过、苦恼过。当最终结果出来的那一刻，一种幸福感，成就感，自豪感油然而生。半个月的努力、到被老师一票否决，从头开始，我们也失望过。在不懈努力下，终于完成了他。这是我们汗水的结晶、也是一次难忘的经历。

## 六、参考文献

Verilog HDL入门：第三版/（美）巴斯克（Bhasker,j）著；夏宇闻，甘伟 译.

——北京：北京航空航天大学出版社， 2008.9

Verilog HDL应用程序设计实例精讲/刘福奇, 刘波编著；

——北京:电子工业出版社,2009

## 七、附 录

### 7.1bmp文件的存储格式

BMP文件可分为四个部分：位图文件头、位图信息头、彩色板、图像数据阵列。

　　一、图像文件头

　　1）1：(这里的数字代表的是"字",即两个字节,下同)图像文件头。424Dh=’BM’，表示是Windows支持的BMP格式。

　　2）2-3：整个文件大小。4690 0000，为00009046h=36934。

　　3）4-5：保留，必须设置为0。

　 4）6-7：从文件开始到位图数据之间的偏移量。4600 0000，为00000046h=70，上面的文件头就是35字=70字节。

　　二、位图信息头

　　5）8-9：位图图信息头长度。

　　6）10-11：位图宽度，以像素为单位。8000 0000，为00000080h=128。

　　7）12-13：位图高度，以像素为单位。9000 0000，为00000090h=144。

　　8）14：位图的位面数，该值总是1。0100，为0001h=1。

　　9）15：每个像素的位数。有1（单色），4（16色），8（256色），16（64K色，高彩色），24（16M色，真彩色），32（4096M色，增强型真彩色）。1000为0010h=16。

　　10）16-17：压缩说明：有0（不压缩），1（RLE 8，8位RLE压缩），2（RLE 4，4位RLE压缩，3（Bitfields，位域存放）。RLE简单地说是采用像素数+像素值的方式进行压缩。T408采用的是位域存放方式，用两个字节表示一个像素，位域分配为r5b6g5。图中0300 0000为00000003h=3。

　　11）18-19：用字节数表示的位图数据的大小，该数必须是4的倍数，数值上等于（≥位图宽度的最小的4的倍数）×位图高度×每个像素位数。0090 0000为00009000h=80×90×2h=36864。

　　12）20-21：用象素/米表示的水平分辨率。A00F 0000为0000 0FA0h=4000。

　　13）22-23：用象素/米表示的垂直分辨率。A00F 0000为0000 0FA0h=4000。

　　14）24-25：位图使用的颜色索引数。设为0的话，则说明使用所有调色板项。

　　15）26-27：对图象显示有重要影响的颜色索引的数目。如果是0，表示都重要。

　　三、彩色板

　　16）28-....(不确定)：彩色板规范。对于调色板中的每个表项，用下述方法来描述RGB的值：

　　1字节用于蓝色分量

　　1字节用于绿色分量

　　1字节用于红色分量

　　1字节用于填充符(设置为0)

　　对于24-位真彩色图像就不使用彩色板，因为位图中的RGB值就代表了每个象素的颜色。

　　如，彩色板为00F8 0000 E007 0000 1F00 0000 0000 0000，其中：

　　00FB 0000为FB00h=1111100000000000（二进制），是红色分量的掩码。

　　E007 0000为 07E0h=0000011111100000（二进制），是绿色分量的掩码。

　　1F00 0000为001Fh=0000000000011111（二进制），是红色分量的掩码。

　　0000 0000总设置为0。

　　将掩码跟像素值进行“与”运算再进行移位操作就可以得到各色分量值。看看掩码，就可以明白事实上在每个像素值的两个字节16位中，按从高到低取5、6、5位分别就是r、g、b分量值。取出分量值后把r、g、b值分别乘以8、4、8就可以补齐第个分量为一个字节，再把这三个字节按rgb组合，放入存储器（同样要反序），就可以转换为24位标准BMP格式了。

对于24bit的bmp格式的图片，没有颜色表项。所以前54个字节表示一些文件的信息。后面的信息就是像素阵列的一些东西，对于24bit的bmp图片，其每一个像素占据3个字节。

对于256×256的位图，其像素阵列占据的字节数是196608（30000H），总共196662（30036H）。