yuv444，yuv422，yuv420

在 YUV 色彩空间中，

每一个颜色有一个亮度信号 Y，

和两个色度信号 U 和 V。

亮度信号是强度的感觉，它和色度信号断开，这样的话强度就可以在不影响颜色的情况下改变。YUV，分为三个分量，

“Y”表示明亮度（Luminance 或 Luma），也就是灰度值；

而“U”和“V” 表示的则是色度（Chrominance 或 Chroma），作用是描述影像色彩及饱

和度，用于指定像素的颜色。如果没有 uv 信息就会显示黑白。人眼的视觉特点是对亮度更铭感，对位置、色彩相对来说不铭感。

在视频编码系统中为了**降低带宽**，可以保存**更多的亮度信息(luma)**，保存**较少的色差信息(chroma)**。

通常对 yuv444，yuv422，yuv420 的解释是后面三个数字分别对应前面三个字母。

拿 yuv422 来说，

y 对应 4，表示四个图形像素中，每个都有亮度值；

u 对应 2，表示四个图形像素中，Cb 只占用两个像素；

v 对应 2，表示四个图形像素中，Cr 占用两个像素。

对于 yuv422 模式，这样解释是没有问题。但是对于 yuv420 解释就不对了，

不能说四个图形像素中，Cr 占用 0 个像素吧？

表示, 是 J：a：b 模式，实心黑色圆圈表示包含色度像素(Cb+Cr），空心圆圈表示不包含色度像素。对于 J:a:b 模式，主要是围绕参考块的概念定义的，这个参考块是一个 J x 2 的矩形，J 通常是 4。

这样，此参考块就是宽度有4个像素、高度有 2 个像素的矩形。a 表示参考块的第一行包含的色度像素样本数，b 表示在参考块的第二行

包含的色度像素样本数。

YUV 使用 RGB 的信息，但它

从全彩色图像中产生一个黑白图像（Y 信号），

然后提取出三个主要的颜色变成两个额外的信号来描述颜色。

把这三个信号组合回来就可以产生一个全彩色图像。

Y通道描述明亮度信号，值的范围介于亮和暗之间。明亮度是黑白电视可以看到的信号。

U (Cb)和V(Cr)通道从红(U)和蓝(V)中提取亮度值来减少颜色信息量。

这些值可以从新组合来决定红，绿和蓝的混合信号。

YCbCr 与 RGB 的相互转换:

Y = 0.2568\*R + 0.5041\*G + 0.0979\*B + 16;

Cb = -0.1482\*R - 0.2910\*G + 0.4392\*B + 128;

Cr = 0.4392\*R - 0.3678\*G - 0.0714\*B + 128;

RGB2YUV

VS\_i

HS\_i

DE\_i

RGB[23:0]

VS\_o

HS\_o

DE\_o

YUV[23:0]

仿真模型

RGB2YUV

VS\_i

HS\_i

DE\_i

RGB[23:0]

VS\_o

HS\_o

DE\_o

MEM

xx.txt

0latency

MEM

post\_process\_dat\_y.txt

MEM

post\_process\_dat\_u.txt

MEM

post\_process\_dat\_v.txt

Y[7:0]

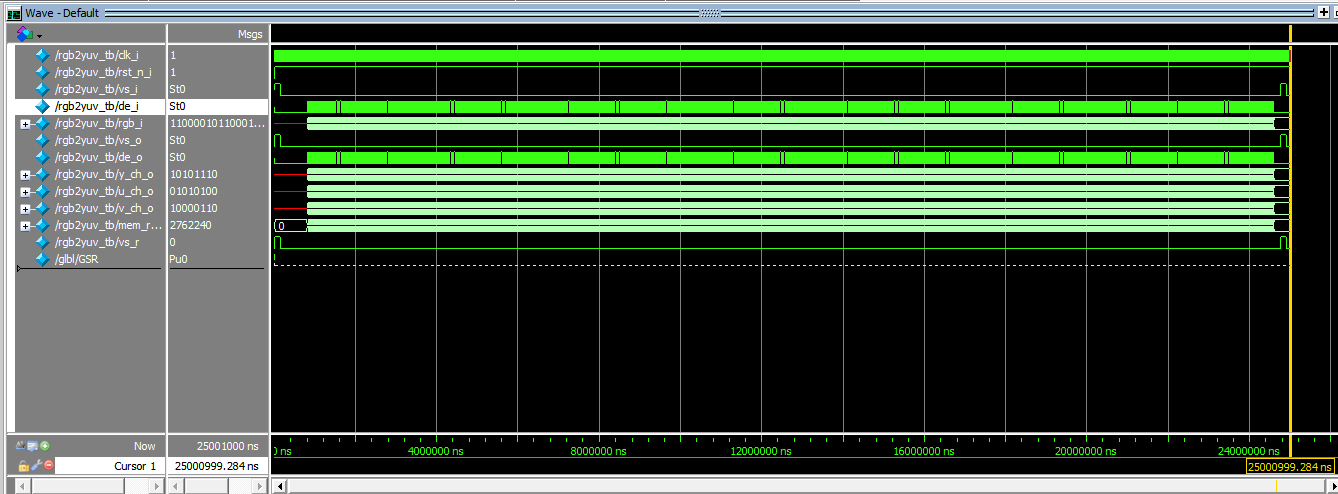
U[7:0]

V[7:0]

Y[23:0]

VGA\_Timing\_gen

仿真结果



输入文件：1.txt（1280\*760 24bit的一张彩色图片）

输出文件：post\_process\_dat\_y.txt，post\_process\_dat\_u.txt，post\_process\_dat\_v.txt

R = 1.1644\*(Y- 16) + 1.5960\*(Cr - 128);

G = 1.1644\*(Y - 16) - 0.3918\*(Cb- 128) -0.8130\*(Cr- 128);

B = 1.1644\*(Y - 16) + 2.0172\*(Cb- 128);

R = (298\*Y+ 409\*Cr - 59684)>>8;

G = (298\*Y - 100\*Cb - 208\*Cr + 34816)>>8;

B = (298\*Y + 516\*Cb - 70912)>>8;

RGB2YUV

VS\_i

HS\_i

DE\_i

YUV[23:0]

VS\_o

HS\_o

DE\_o

RGB[23:0]

仿真模型

RGB2YUV

VS\_i

HS\_i

V[7:0]

DE\_i

Y[7:0]

VS\_o

HS\_o

DE\_o

RGB[23:0]

MEM(FIFO)

0latency

MEM(FIFO)

0latency

MEM

post\_process

\_dat\_v.txt

U[7:0]

MEM(FIFO)

0latency

MEM(FIFO)

0latency

MEM

post\_process

\_dat\_b.txt

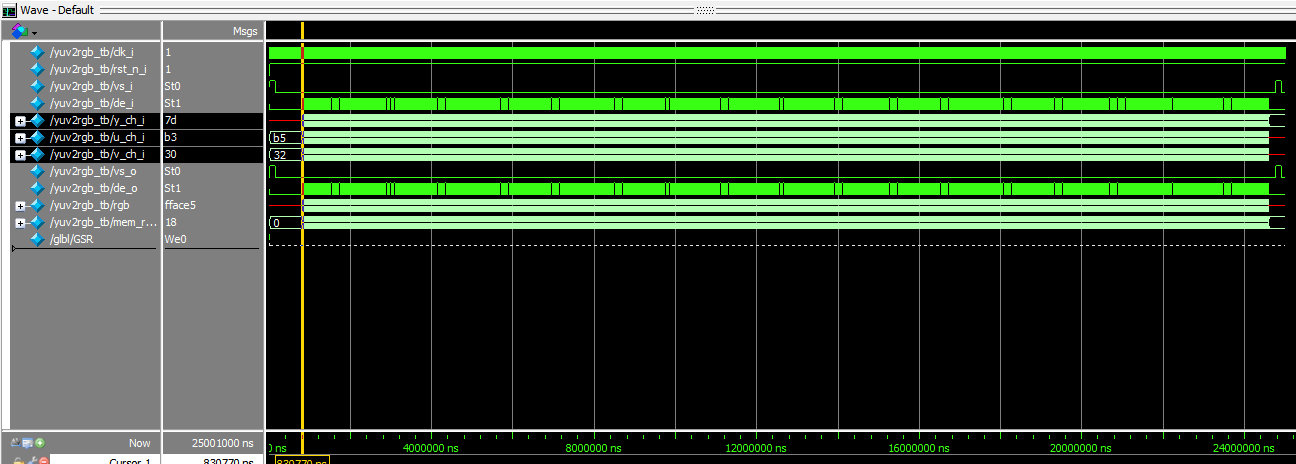
R[7:0]

G[7:0]

B[7:0]

VGA\_Timing\_gen

仿真结果



输入文件：post\_process\_dat\_y.txt，post\_process\_dat\_u.txt，post\_process\_dat\_v.txt

输出文件：post\_process\_dat\_r.txt，post\_process\_dat\_g.txt，post\_process\_dat\_b.txt

Matlab：

**将bmp格式图片转换为txt文件**

function img\_size = readimg\_to\_txt(input\_bmp, output\_txt);

% bmp to Verilog memory init txt file

% retuen size

img = imread(input\_bmp);

figure, imshow(img);

fid = fopen(output\_txt, 'w');

fprintf(fid, '@00\r\n');

[rows ,cols] = size(img);

for m = 1:rows

for n = 1:cols

if (mod(n,1280) == 0)

fprintf(fid, '%2x\r\n', img(m, n));

else

fprintf(fid, '%2x ', img(m, n));

end

end

fprintf(fid, '\r\n');

end

fclose(fid);

**将转换后结果再转换为Bmp格式的图片**

function img = GetImgFromTxt3(ImgTxt\_File\_Name\_r, ImgTxt\_File\_Name\_g,ImgTxt\_File\_Name\_b,Col, Row, OutImgFile\_Name);

fid = fopen(ImgTxt\_File\_Name\_r, 'r');

img = fscanf(fid,'%x',[Col Row]);

img = img';

image1\_in\_r = uint8(zeros(Row,Col));

for row = 1:Row

for col = 1:Col

image1\_in\_r(row, col) = img(row, col);

end

end

fid = fopen(ImgTxt\_File\_Name\_g, 'r');

img = fscanf(fid,'%x',[Col Row]);

img = img';

image1\_in\_g = uint8(zeros(Row,Col));

for row = 1:Row

for col = 1:Col

image1\_in\_g(row, col) = img(row, col);

end

end

fid = fopen(ImgTxt\_File\_Name\_b, 'r');

img = fscanf(fid,'%x',[Col Row]);

img = img';

image1\_in\_b = uint8(zeros(Row,Col));

for row = 1:Row

for col = 1:Col

image1\_in\_b(row, col) = img(row, col);

end

end

image1\_out\_rgb = cat(3,image1\_in\_r,image1\_in\_g,image1\_in\_b);

imshow(image1\_out\_rgb),title('The rgb image');

imwrite(image1\_out\_rgb, OutImgFile\_Name, 'bmp');

fclose(fid);