1、使用vivado simulator 在摸写内存存在bug，一帧数据处理结果在写入txt文件中时，总是末尾少写一些数据。

同样的代码在 ModelSim Simulator仿真则没有问题可以正常完整写完。

2、

24bit 彩色 bmp格式文件转换txt，数据拍的格式如下，仿真读写的时候需要注意

像素

R 1 2 3 4 5 ... 1280

G 1 2 3 4 5 ... 1280

B 1 2 3 4 5 ... 1280

R 1281 1282 1283 1284 1285 ... 2560

G 1281 1282 1283 1284 1285 ... 2560

B 1281 1282 1283 1284 1285 ... 2560

3、

原转换公式与实际在RTL电路中计算的公式存在一些误差

**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;**

为了方便在RTL电路中运算将公式乘以256后取整后运算，然后再取高8位的结果。

例如 0.2568 \* 256 后的结果为65.7408 为了避免使用浮点数，减少使用资源，取结果为66，

其他涉及浮点数均取整。

**Y = (66\*****R + 129\*G + 25\*B + 4096 )>>8;**

**Cb= (-38\*R - 74\*G + 112\*B + 32768)>>8;**

**Cr = (112\*R - 94\*G - 18\*B + 32768 )>>8;**

同理

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

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

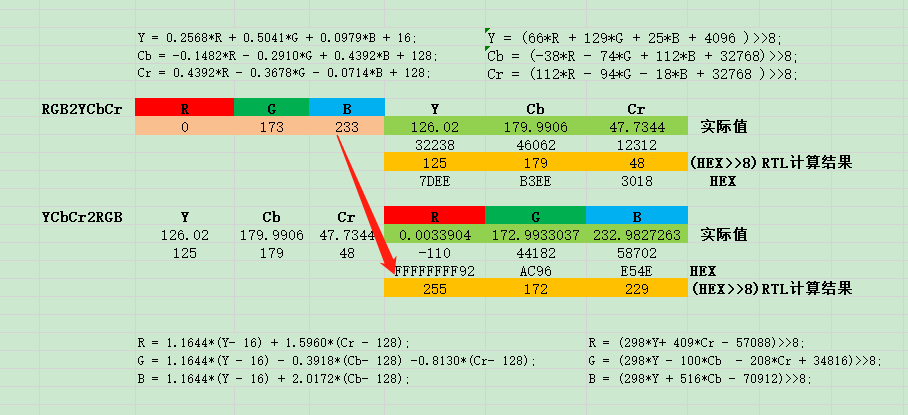
**B = 1.1644\*(Y - 16) + 2.0172\*(Cb- 128);**

**R = (298\*Y+ 409\*Cr - 56992)>>8;**

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

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

这样处理减少了资源的使用，同时需要知道带来的影响，颜色的准确度会稍微有点偏差，以及在转换计算的过程出现不准的情况。例如当RGB红色分量的值为零，通过RGB2YCbCr和YCbCr2RGB转换后红色分量的结果可能是个负值，负值的高8位为FF。具体表现如下图中情况。



原图



经过RGB2YCbCr 以及 YCbCr2RGB模块转换后的结果。



因此在实际使用过程中要求比较高，还需要对算法进行优化。