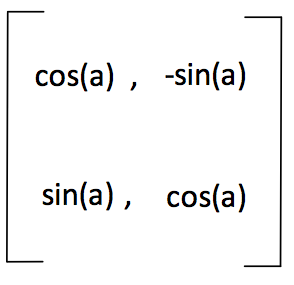
# 图像旋转方案

## 一，图像旋转原理

图像旋转的本质利用的是向量的旋转。

矩阵乘法的实质是进行线性变换，因此对一个向量进行旋转操作也可以通过矩阵和向量相乘的方式进行。

假设有二维向量v = [x ; y]，其中x,y是原图的像素点的横轴和纵轴坐标。若要进行逆时针旋转角度a。则旋转矩阵R为



旋转后的向量 Ro = R \* v。 Ro =[Xo,Yo]; Ro 是输出图像的坐标值。

在正式处理过程中可以这么表示，原像素位置记为p，中心点记为c，旋转后像素位置记为pp。则有(pp - c) = R\*(p - c)，即

pp = R\*(p-c) + c

## 输入输出图像坐标的方案选择

方案1，第一次实现代码的思路是正向的思路，也就是把原图进行向量的旋转，找到旋转后的向量的位置，然后将原图的像素值赋值过去即可。





利用MATLAB进行该算法的验证，可以看到有这么几个瑕疵，首先是图像的大小和原图一样导致边缘被裁剪了，其次是图像中会出现很多噪声很多杂点，出现杂点的原因是从原图旋转后的像素位置在原图可能找不到，解决方法是用逆向思维，从目标图片反向旋转到原图进行像素查找。

方案2，

## 旋转参数的提取和表示

在该设计中，要求图像拥有0到360的任意角度的旋转，坐标变换需要角度的正弦和余弦值。

1. 该正弦，余弦通过MATLAB计算得到，并预先储存到FPGA的片上储存空间中，在进行坐标变换时，读取对应角度的正弦，余弦值，进行坐标变换。由于计算得到的正弦和余弦值为浮点数，而FPGA擅长于进行整数运算。故要进行浮点数到整数的转换，具体的实现方法是，将计算得到的浮点正弦，余弦值乘上 256 后再取整，计算得到的结果于原结果相比被扩大了256倍，而在数字电路中，除法操作可以用移位来进行。结果右移8位即等效于除于256 。
2. 坐标变换需要0到360度的正弦，余弦值。一般的方案时直接储存 0到360的正弦，余弦值，需要时直接读取即可，但这种方法过于浪费储存资源。根据数学上的原理，正弦，余弦函数是周期变换的，在一个周期内有多个重复的函数。只需要储存四分之一周期的正弦函数值，再根据角度关系，确定其他四分之三周期角度的正弦，余弦值以及符号值。
3. 正负数的表示方法。在数字电路中，无符号数计算较为简单，电路设计也比较简单。但在坐标变换中，有符号位负的参数。在工程设计中，采用储存各个参数的绝对值，并另外设置一个符号标志位来标志参数的正负。用1表示参数是正数，用0表示参数是负数。在坐标变换计算中，根据标识位的值（0,1）来确定进行进行加法计算还是减法计算。

4，将坐标变换计算模块封装为一个子模块，输入输出图像的坐标和旋转角度后，即可计算出对应的输入图像对应的像素的坐标。然后读取该坐标的像素值，写入到旋转重建的图像对应的坐标位置即可。