设计方案

算法步骤

1. 计算图像梯度

```
Image = imread('f6-2.jpg');      %读取待处理图像
Image = rgb2gray(uint8(Image));%图像灰度化
fx=[1 4 1;0 0 0; -1 -4 -1];      %x方向梯度算子
fy=[1 0 -1; 4 0 -4; 1 0 -1];     %y方向梯度算子
fx=fx.*(1/6);                    %x方向梯度算子归一化
Ix=filter2(fx,Image)             %计算x方向梯度Ix
fy=fy.*(1/6);                    %y方向梯度算子归一化
Iy=filter2(fy,Image)             %计算y方向梯度Iy
Ix2 = Ix.^2;                     %计算Ix的平方
Iy2 = Iy.^2;                     %计算Iy的平方

Ixy = Ix.*Iy;                    %计算Ix和Iy的乘积
```

2. 对Ix2、Iy2、Ixy分别进行B样条滤波分别得到A、B、C

```
mh=[1 4 1; 4 16 4; 1 4 1].*(1/36);%B样条滤波算子
h=mh.*(1/36);                     %B样条滤波算子归一化
A = filter2(h,Ix2);               %对Ix2进行B样条滤波
```

```
B = filter2(h,Iy2);                    %对Iy2进行B样条滤波
C = filter2(h,Ixy);                    %对Ixy进行B样条滤波
```

3. 计算CRF和CRF的最大值CRFmax，CRF为矩阵、CRFmax为该矩阵中的最大值，每帧结束更新依次CRFmax

```
            M = [A(i,j) C(i,j);
                 C(i,j) B(i,j)];

          CRF(i,j) = det(M)-t*(trace(M))^2;
```

说明：

$$M=\begin{bmatrix} A(i,j) & C(i,j) \\ C(i,j) & B(i,j) \end{bmatrix}$$

$$CRF(i,j)=\det(M)-t*(trace(M))\hat{}2$$

$$\det M = AB - C^2$$

$$traceM = A + B$$

$$CRF = (AB - C^2) - t(A+B)^2$$

4. 根据相似度方法初筛角点

即判断3X3窗口的中心像素值和周围像素值的差异是否在阈值范围内，如在则认为中心的和周围像素点相似，统计相似点的个数，只有相似点个数在1~6之间的才认为是候选角点。

```
[nrow,ncol]=size(Image);
Corner = zeros(nrow,ncol); %矩阵Corner用来保存候选角点位置,初值全零，值为1的点是角点
t=20;%参数t:点(i,j)八邻域的“相似度”参数，只有中心点与邻域其他八个点的像素值之差在（-t,+t）之间，才确认它们为相似点，相似点不在候选角点之列
boundary=1;%由窗口大小决定，等于窗口大小/2，如3X3的窗口，boundary=1；5X5的窗口，boundary=2
for i=boundary+1:nrow-boundary
    for j=boundary+1:ncol-boundary
          nlike=0; %相似点个数
          if Image(i-1,j-1)>Image(i,j)-t && Image(i-1,j-1)<Image(i,j)+t
              nlike=nlike+1;
          end
          if Image(i-1,j)>Image(i,j)-t  && Image(i-1,j)<Image(i,j)+t
              nlike=nlike+1;
          end
          if Image(i-1,j+1)>Image(i,j)-t && Image(i-1,j+1)<Image(i,j)+t
              nlike=nlike+1;
          end
          if Image(i,j-1)>Image(i,j)-t  && Image(i,j-1)<Image(i,j)+t
              nlike=nlike+1;
```

```
            end
            if Image(i,j+1)>Image(i,j)-t &&
Image(i,j+1)&& Image(i,j+1)<Image(i,j)+t
                nlike=nlike+1;
            end
            if Image(i+1,j-1)>Image(i,j)-t &&
Image(i+1,j-1)<Image(i,j)+t
                nlike=nlike+1;
            end
            if Image(i+1,j)>Image(i,j)-t  &&
Image(i+1,j)<Image(i,j)+t
                nlike=nlike+1;
            end
            if Image(i+1,j+1)>Image(i,j)-t &&
Image(i+1,j+1)<Image(i,j)+t
                nlike=nlike+1;
            end
            if nlike>=1 && nlike<=6
             Corner(i,j)=1;
            end
        end
    end
```

5. 角点判断，并得到角点的坐标和总数量

判断一个坐标为(i, j)的点是为角点需要满足3个条件：

1. (i, j)是步骤4中初筛剩下的点；

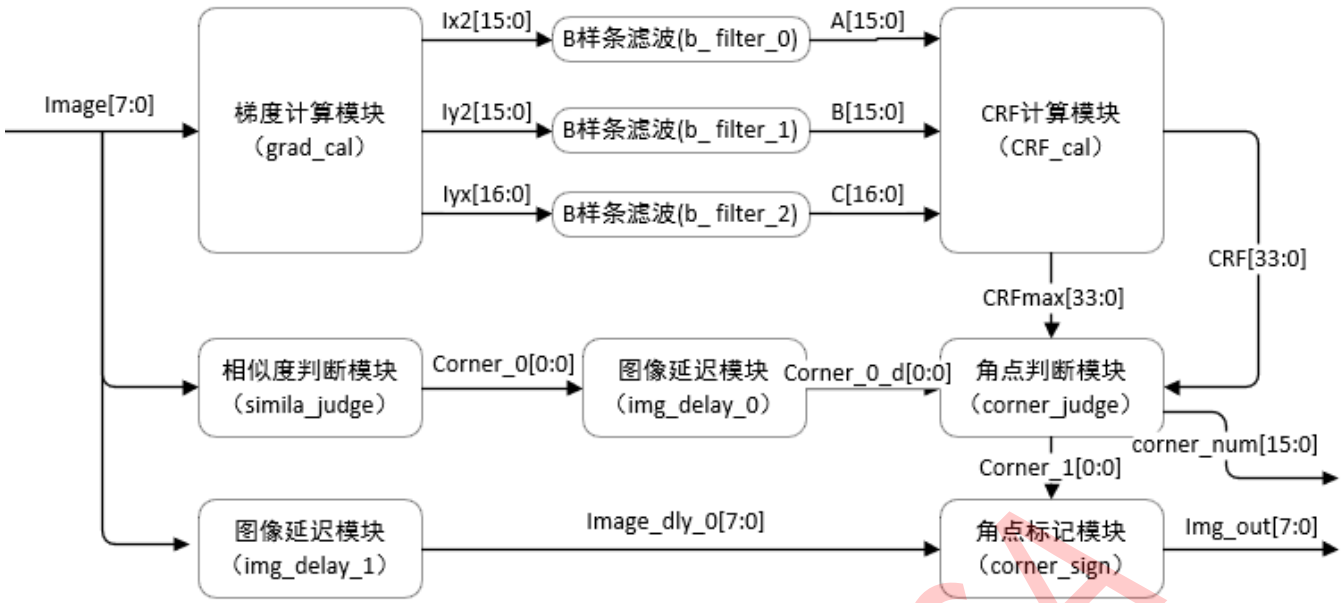
2. CFR(i, j)是3x3窗口邻域内的最大值，也就是局部极大值；

3. CFR(i,j) > t*CRFmax，t是提前设置好的系数，本设计使用0.015625即1/64；

```
 count = 0;          % 用来记录角点的个数
 t=0.015625;  %1/64
% 下面通过一个3*3的窗口来判断当前位置是否为角点
for i = boundary+1:nrow-boundary
for j = boundary+1:ncol-boundary
             if CRF(i,j) > t*CRFmax && CRF(i,j)
>CRF(i-1,j-1) ......
                && CRF(i,j) > CRF(i-1,j) && CRF(i,j)
> CRF(i-1,j+1) ......
                && CRF(i,j) > CRF(i,j-1) && CRF(i,j)
> CRF(i,j+1) ......
                && CRF(i,j) > CRF(i+1,j-1) &&
CRF(i,j) > CRF(i+1,j).....
                && CRF(i,j) > CRF(i+1,j+1)
             Corner(i,j) = 1;
             count=count+1;%这个是角点，count加1
             else % 如果当前位置（i,j）不是角点，则在
Corner(i,j)中删除对该候选角点的记录
                  Corner(i,j) = 0;
              end
end
end
```

6. 标记角点位置

方案框图



梯度计算模块（grad_cal）用于计算横纵方向的梯度，横坐标方向的梯度计算算子为：

fx=[1 0 -1; 4 0 -4; 1 0 -1]，纵坐标方向的梯度计算算子为：fy=[1 4 1;0 0 0; -1 -4 -1]。Ix、Iy分别为原图像经过fx、fy算子卷积得到的结果，也即梯度矩阵。输出数据为Ix2、Iy2、Ixy，它们和Ix、Iy的关系为：Ix2 = Ix*Ix、Iy2=Iy*Iy、Ixy=Ix*Iy。

B样条滤波模块(b_ filter_0)是一个3x3的2D滤波模块,该模块滤波算子为mh=[1 4 1; 4 16 4; 1 4 1]。

CRF计算模块（CRF_cal）计算CRF矩阵，即求每个CRF(i,j),同时计算每帧CRF的最大值CRFmax。

相似度判断模块（simila_judge）功能是对角点进行初筛，利用每个像素点和其周围8邻域点的相似程度决定该点是否为角点候选点，相似度由像和8邻域点的灰度差值决定，差值绝对值小于阈值20则认为相似，相似点的个数在1~4范围内时该点才被判断为候选角点，否则直接认为不是角点。

图像延迟模块的作用主要是对齐图像数据，实际上也是对齐行场同步信号如vsync和hsync，包含两项工作，一个是进行行缓存，以匹配其他窗口滤波模块带来的图像边界问题，该边界问题表现为图像滤波后整体上移一行；另一个是匹配其他窗口滤波模块带来的数据延迟问题，以梯度计算

模块为例，计算梯度需要多个时钟打拍以保证时序性能，所以图像延迟模块还有适配这种延迟，而不单是适配图像上移的情况。

角点判断模块（corner_judge）的功能是判断像素点是否为角点，判断的依据是同时满足以下3个条件： 第一个条件是坐标(i,j)是相似度判断模块（simila_judge）初筛剩下的点；第二个条件是CFR(i,j)是3x3窗口邻域内的最大值，也就是局部极大值；第三个条件是CFR(i,j) > t*CRFmax，t是提前设置好的系数，本设计使用0.015625即1/64；

角点标记模块（corner_sign）功能是在角点坐标上显示不同的颜色以突显各个角点位置。

子模块设计

梯度计算模块

功能说明

计算图像横、纵方向的梯度的平方以及横、纵方向的梯度的乘积：定义Ix、Iy分别为原图像在横、纵坐标方向上的梯度值，其中Ix是原图像经过滤波窗口为fx=[1 4 1;0 0 0; -1 -4 -1]的窗口滤波的结果、Iy是原图像经过滤波窗口为fy=[1 0 -1; 4 0 -4; 1 0 -1]的窗口滤波的结果。最终输出的结果是Ix2、Iy2、Ixy，它们和Ix、Iy的关系为：Ix2 = Ix*Ix、Iy2=Iy*Iy、Ixy=Ix*Iy。

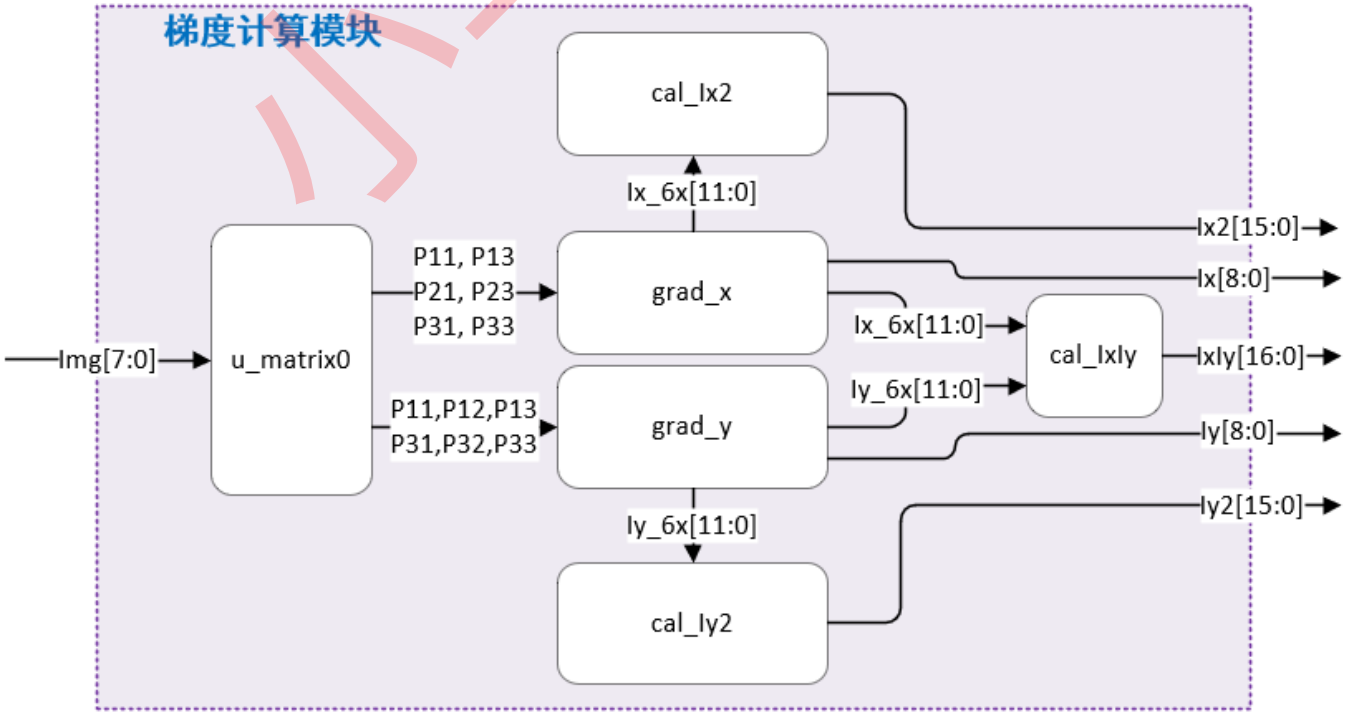
接口定义

信号	方向
clk	input
rst_n	input
pre_vsync	input
pre_hsync	input

pre_valid	input
pre_data[7:0]	input
post_vsync	output
post_hsync	output
post_valid	output
post_Ix2[15:0]	output
post_Iy2[15:0]	output
post_Ixy[16:0]	output

方案框图

如下图所示：



u_matrix0是3x3行缓存模块，该模块可以提取出窗口数据，其中Pij代表窗口中第i行第j列的像素的灰度值，u_matrix0模块会导致图像整体

上移一行，这个问题会导致边界的一些小问题，这里我们可以不关心边界问题，但是图像上移需要关注，实际上每进行一次窗口滤波（使用到u_matrix0），结果都会有一行的上移，所以后续的模块设计时需要将图像上移一行的问题考虑进去，设法使不同的滤波图像之间或于原图之间数据对齐。

grad_x模块和grad_y模块分别计算x和y方向的梯度，输出的结果以放大6倍的形式表示即Ix_6x和Iy_6x，后续模块计算时再考虑归一化，这样可以最大程度保持精度。Ix_6x和Iy_6x均是二进制补码形式的有符号数。

cal_Ix2和cal_Iy2模块分别计算Ix和Iy的平方，同时进行归一化后输出结果，Ix2和Iy2都是16位的无符号数，因为Ix实际的位宽为9位，也即多一位符号位，所以其绝对值的范围只需要8位即可表示，所以Ix2最多只需要16位即可表示，Iy也是同理。

cal_IxIy模块计算Ix和Iy的乘积，结果以17位的有符号数表示，二进制补码形式。

B样条滤波模块

功能说明

B样条滤波模块是一个3X3窗口滤波模块，滤波系数fb=[1 4 1; 4 16 4; 1 4 1]，该模块输入为17bits二进制补码形式有符号，输出亦为17bits二进制补码形式有符号。

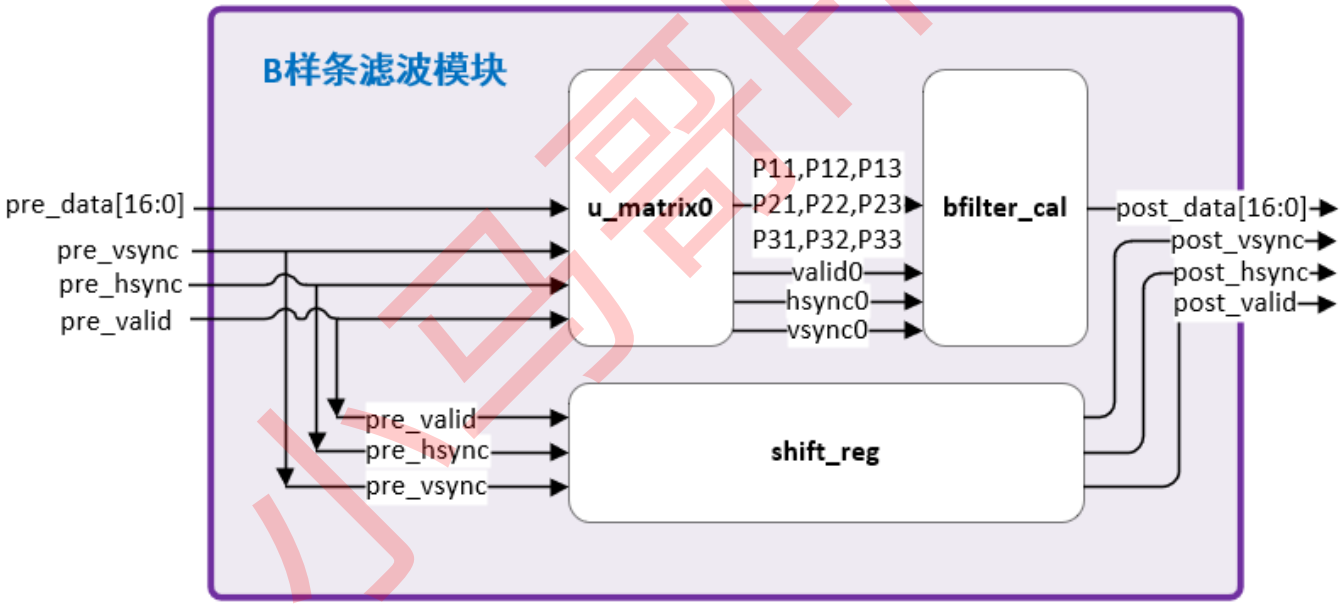
接口定义

信号	方向
clk	input
rst_n	input
pre_vsync	input
pre_hsync	input

pre_valid	input
pre_data[16:0]	input
post_vsync	output
post_hsync	output
post_valid	output
post_data[16:0]	output

方案框图

如下图所示：



u_matrix0是3x3行缓存模块，该模块可以提取出窗口数据，其中Pij代表窗口中第i行第j列的像素的灰度值，u_matrix0模块会导致图像整体上移一行，这个问题会导致边界的一些小问题，这里我们可以暂时不关心边界问题，但是图像上移需要清楚，以免仿真时产生困惑，实际上每进行一次窗口滤波（使用到u_matrix0），结果都会有一行的下移。所以系统设计时需要将图像下移一行的问题考虑进去，设法使不同的滤波图像之间或于原图之间数据对齐。

bfilter_cal模块计算3x3滤波结果，输出的结果是二进制补码形式的有符号数，位宽和输入相同，均为17bits。

shift_reg模块对同步信号进行打拍延迟，以和运算后的结果数据保持对齐，实际上延迟的拍数和bfilter_cal模块卷积运算的数据延迟时钟数一致。

CRF计算模块

功能说明

CRF计算模块实现计算CRF功能，并且在一帧最后（VSYNC下降沿）输出CRF的最大值CRFmax。CRF的计算公式为：

$$CRF = \left(AB - C^2\right) - t(A+B)^2$$

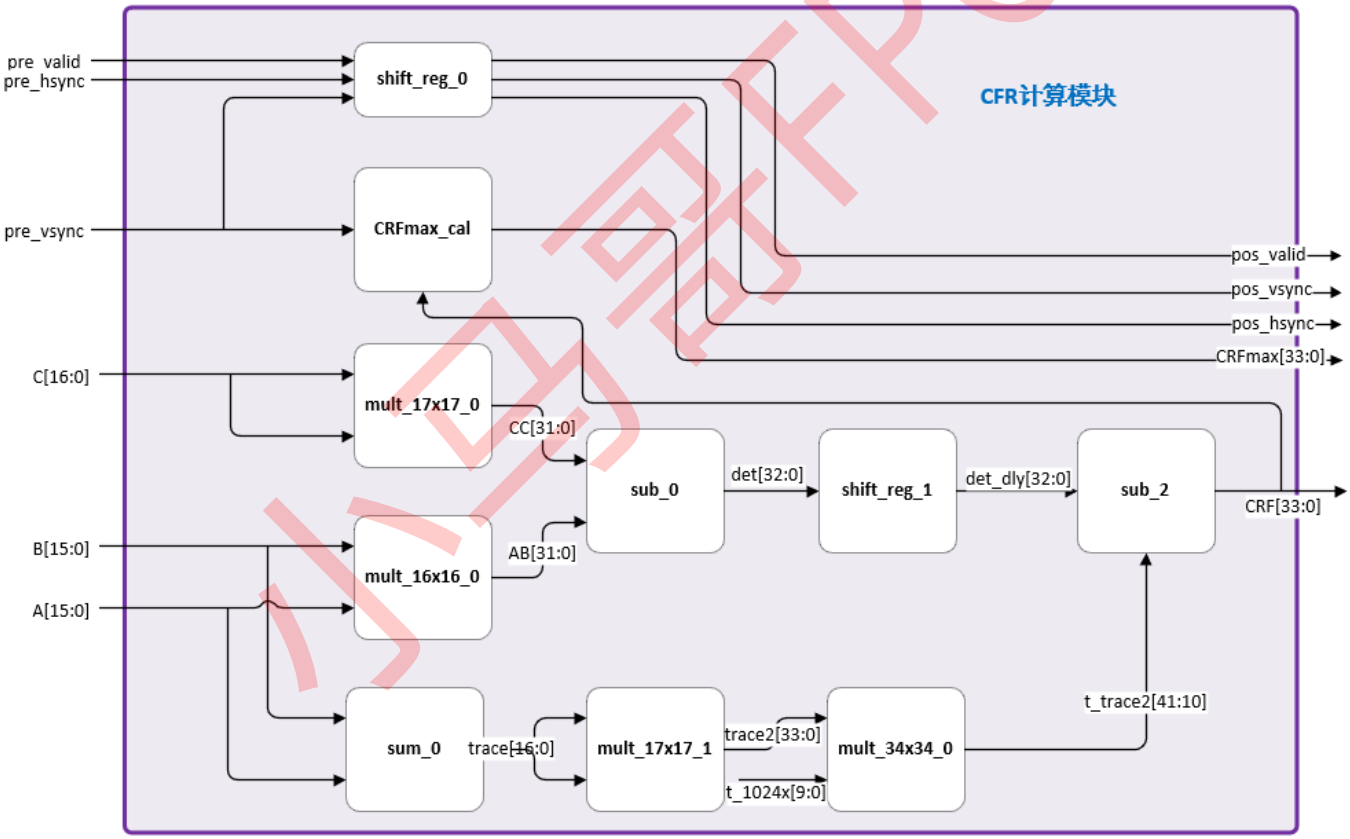
其中A、B、C均为输入数据，C为17bits有符号数，二进制补码形式，A、B为16bits无符号数。t为系数参数，默认值为0.05，经过1024倍放大量化后的值为51。

接口定义

信号	方向
clk	input
rst_n	input
pre_vsync	input
pre_hsync	input
pre_valid	input
A[15:0]	input
B[15:0]	input
C[16:0]	input

post_vsync	output
post_hsync	output
post_valid	output
post_CRF[33:0]	output
post_CRFmax[33:0]	output

方案框图



C[16:0]是二进制补码形式的有符号数输入，mult_17x17_0是17bits的有符号数乘法器，其输出应当是34btis，但由于这模块是求C的平方，所以实际结果CC[31:0]是取低32bits且是无符号数。

A[15:0]和B[15:0]是无符号数，两者经过mult_16x16_0模块后得到32位的无符号输出AB[31:0]，A和B经过sum_0模块后得到17bits的无符号数输出trace[16:0]，trace[16:0]经过mult_17x17_1和自身做乘法得到trace2[33:0]，trace2[33:0]和常数t_1024x(默认51)经过mult_34x34_1

做乘法后，结果应该是44bits，但是由于t取值必定小于0.25（实际远小于）所以该结果最多只需要42bits即可表示，再截尾低10bits进行归一化得到32bits的无符号输出t_trace2[41:10]。

AB[31:0]和CC[31:0]经过sub_0做减法后得到33bits的二进制补码形式的有符号数det[32:0]，det[32:0]经过shift_reg_1模块延迟几个时钟后得同样是有符号数的det_dly[32:0]，这里shift_reg_1的作用是对齐数据。

det_dly[32:0]和t_trace2[41:10]经过sub_2减法运算后得到34bits的有符号数输出CRF[33:0]。

CRFmax_cal模块根据pre_vsync信号的下降沿更新每帧的CRF最大值并输出，CRFmax[33:0]是二进制补码形式的有符号数。在比较大小时，若两个操作数都是signed类型的，则即使位宽不一致也是可以直接用比较运算符比较得到正确结果的，但只要有一个操作数是无符号数，则自动生成的比较器就是无符号数比较器，这样就不一定得到正确的结果了。

shit_reg_0则对同步信号进行延迟操作以补偿数据运算带来的延迟，使得同步信号和数据对齐。

相似度判断模块

功能说明

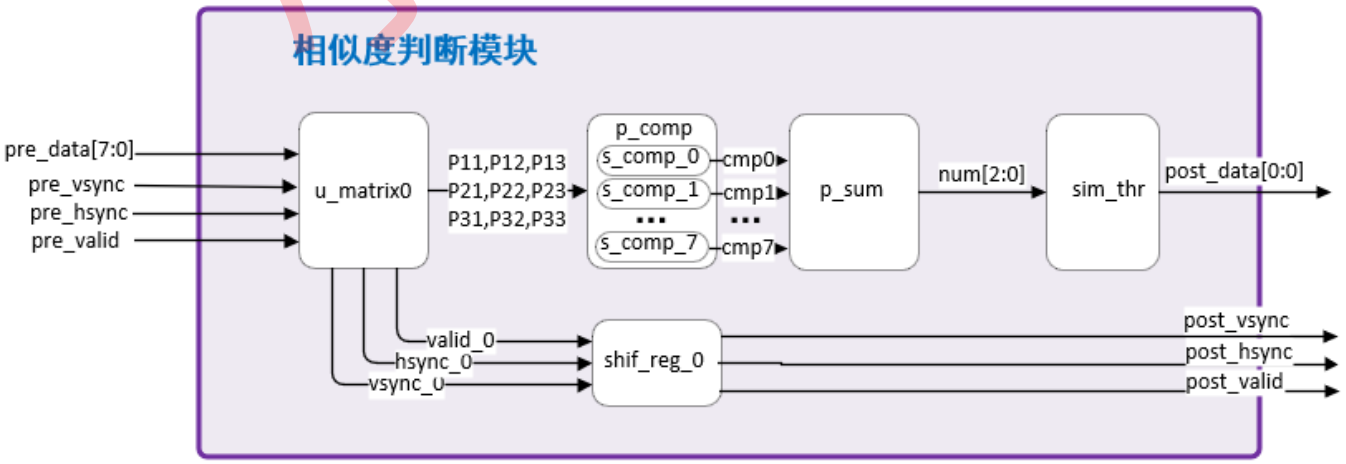
相似度判断模块（simila_judge）功能是对角点进行初筛，利用每个像素点和其周围8邻域点的相似程度决定该点是否为角点候选点，相似度由像和8邻域点的灰度差值决定，差值绝对值小于阈值20则认为相似，相似点的个数在1~4范围内时该点才被判断为候选角点，否则直接认为不是角点。

接口定义

信号	方向

信号	方向
clk	input
rst_n	input
pre_vsync	input
pre_hsync	input
pre_valid	input
pre_data[7:0]	input
post_vsync	output
post_hsync	output
post_valid	output
post_data[0:0]	output

方案框图



u_matrix0是3x3行缓存模块，该模块可以提取出窗口数据，其中Pij代表窗口中第i行第j列的像素的灰度值，u_matrix0模块会导致图像整体上移一行，这个问题会导致边界的一些小问题，这里我们可以不关心边界问题，但是图像上移的情况需要知道，以免仿真时产生困惑，实际上每进行一次窗口滤波（使用到u_matrix0），结果都会有一行的下移。

p_comp模块是8路并行比较模块，把P22和其他8个数据同时进行比较，得到8个结果cmp0~7输出。

p_sum模块功能是并行加法器，其计算cmp0~7这8个1bit数据的和并输出结果num[2:0]。

sim_thr模块功能是判断num[2:0]的值是否在预定的范围中（默认：1≤num≤4），在范围中输出1否则输出0，也即输出post_data[0:0]。

shift_reg_0模块对同步信号进行延迟，以补偿p_comp、p_sum、sim_thr模块运算产生的数据延迟，以保证同步信号和数据的对齐。

角点判断模块

功能说明

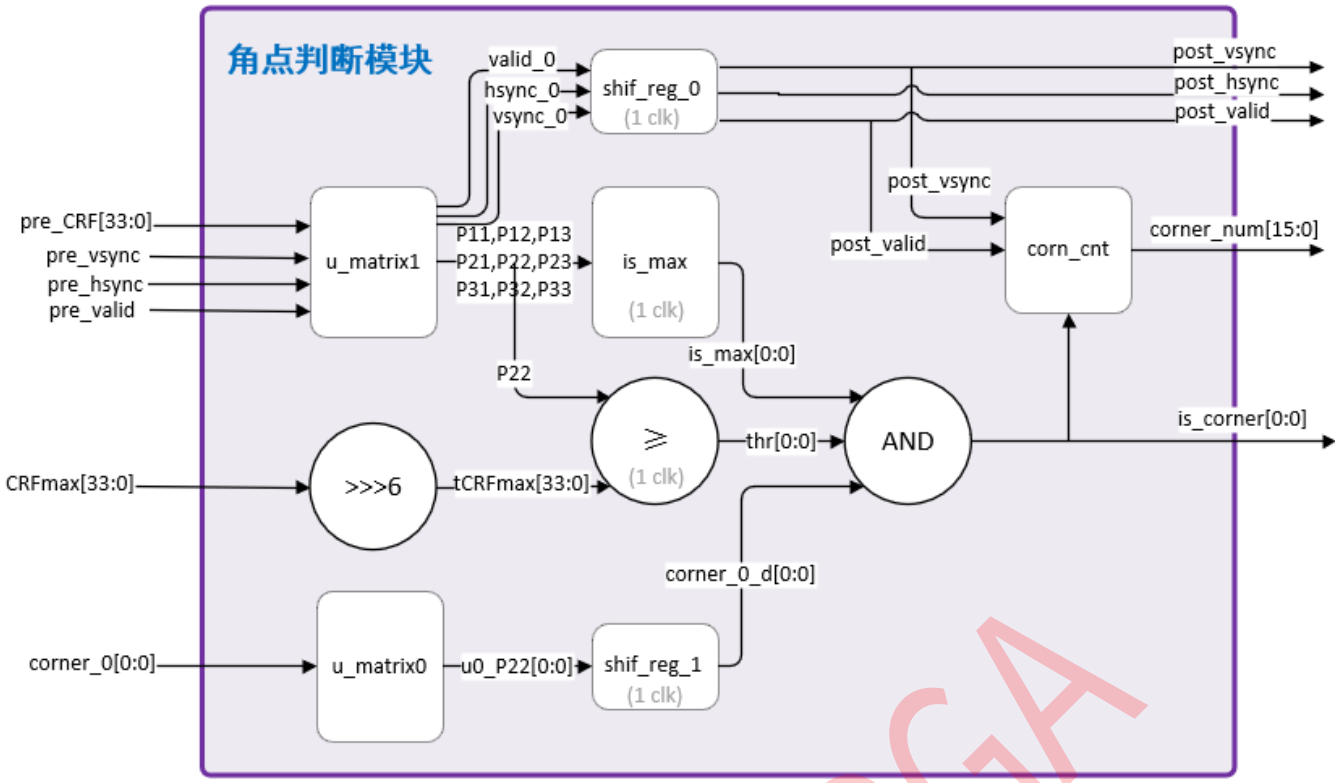
角点判断模块（corner_judge）的功能是判断像素点是否为角点，判断的依据是同时满足以下3个条件： 第一个条件是坐标(i,j)是相似度判断模块（similar_judge）初筛剩下的点即输入corner_0[0:0]为1；第二个条件是CFR(i,j)是3x3窗口邻域内的最大值，也就是局部极大值；第三个条件是CFR(i,j) > t*CRFmax，t是提前设置好的系数，本设计使用0.015625即1/64;

接口定义

信号	方向
clk	input
rst_n	input

rst_n	input
pre_vsync	input
pre_hsync	input
pre_valid	input
pre_CRF[33:0]	input
corner_0[0:0]	input
CRFmax[33:0]	input
post_vsync	output
post_hsync	output
post_valid	output
corner_num[15:0]	output
is_corner[0:0]	output

方案框图



u_matrix1模块的功能是提取pre_CRF[33:0]的3x3窗口数据，以便于判断中心点CRF是否是局部极大值；

is_max模块则是判断中心点CRF是否是3x3窗口内最大值，若是则输出的is_max[0:0]为1，否则为0；

CRFmax[33:0]在一帧期间都保持不变，将其算术右移6位得到的结果和中心点CRF比较，即图中的P22和tCRFmax比较，若P22≥tCRFmax则thr输出1否则输出0；

u_matrix0模块的功能是对corner_0进行1行延迟以适配CRF的窗口提取带来的延迟，保证corner_0和CRF一一对应；

shift_reg_0和shift_reg_1都是对延迟一个时钟，目的是补偿is_max运算带来的1个时钟的数据延迟，保证数据对齐；

corn_cnt是角点数量统计模块，统计每帧is_corner为1的像素点个数，corner_num根据post_vsync信号的下降沿更新数据，每帧更新一次。