第六届

全国大学生集成电路创新创业大赛

报告类型：技术文档

参赛杯赛：紫光同创杯

作品名称：水果识别系统

队伍编号：CICC4953

团队名称：湘云川菜队

目录

背景.....1

1 功能与应用景.....2

1.1 功能场景.....2

1.2 应用场景.....2

2 模块内容及算法介绍.....2

2.1 物体识别系统架构.....3

2.2 图像处理模块各自模块算法（ISP）.....4

2.2.1RGB2YCBCR.....4

2.2.2 中值滤波.....5

2.2.3 颜色识别.....6

2.2.4 二值化.....7

2.2.5 腐蚀膨胀.....8

2.2.6 hvcount.....8

2.2.7projection.....8

2.2.8 种类识别.....8

2.2.9 数量识别.....10

2.2.10 display.....10

3 子模块接口列表与接口时序.....10

3.1 RGB2YCBCR.....10

3.1.1 接口列表. 11
3.1.2 接口时序. 11
3.2 中值滤波. 11
3.2.1 接口列表. 11
3.2.2 接口时序. 12
3.3 二值化. 12
3.3.1 接口列表. 12
3.3.2 接口时序. 12
3.4 膨胀. 12
3.4.1 接口列表. 13
3.4.2 接口时序. 13
3.5 腐蚀. 13
3.5.1 接口列表. 13
3.5.2 接口时序. 14
3.6HVCOUNT. 14
3.6.1 接口列表. 14
3.6.2 接口时序. 14
3.7 PROJECTION. 14
3.7.1 接口列表. 14
3.7.2 接口时序. 15
3.8 颜色识别. 15
3.8.1 接口列表. 16

3.8.2 接口时序. 16
3.9 数量识别. 16
3.9.1 接口列表. 16
3.9.2 接口时序. 17
3.10 种类识别. 17
3.10.1 接口列表. 17
3.10.2 接口时序. 18
3.11 DISPLAY. 18
3.11.1 接口列表. 18
3.11.2 接口时序. 19
4 水果识别系统硬件验证. 20
4.1 测试效果展示. 20
4.2 水果识别结果. 21

背景：物品识别的背景分析（水果识别）

物体识别是计算机视觉领域中的一项基础研究，它的任务是识别出图像中有什么物体，并报告出这个物体在图像表示的场景中的位置和方向。

在 80 年代以来已有的水果识别系统中，应用得较多的包括水果的伤痕及缺陷检测，水果等级的自动识别与划分等。这类分类系统大多要求将采摘的水果置于严格限定的状态之中，比如一类水果分类系统利用气味传感器件采集数据，然后以此作为特征对水果进行分类或者质量评估，分类准确率可以达到 90%。但是这类系统的一个明显缺陷是必须将采摘的水果置于相对隔离的空间中，需要采用特殊的气味传感器件来采集数据,并且容易受到外部环境，例如流通的空气等因素影响。但是由于水果类作物的图像具有显著的分辨特征，很容易进行图像特征采集，因此利用图像信息对水果分类和评级得到更为广泛的关注，可以在无损状态下进行快速而准确的自动分类。在水果识别学术研究方面，从 80 年代开始，国内外就有学者研究水果的计算机自动识别，传统的水果识别仅仅基于水果的轮廓曲线特征实现，近年来出现了一些基于模式识别算法的较新方法，比如基于神经网络的水果识别系统，并利用了多种光谱下的水果图像来进行识别。这些技术相比早期方法有了很大进步，但大部分的研究都集中于分类器的设计，忽略了系统前端的图像信号采集和有效特征提取。例如经网络系统对于输入的水果图像直接进行训练，没有考虑样本中存在的噪音。神经网络中最优参数的选择也只能根据经验。在图像采集方面虽然加入多光谱图像，但是成本较高，无法得到推广。在一个模式识别系统中，除了分类器的选择和训练，图像特征的筛选、采集环境、信号预处理、最佳训练样本选择都对系统性能有着重要影响，因此，为了获得更好的性能，必须基于用户的实际需求，从一个系统、整体的观点，来构建一个更加有效的水果识别系统。

1 功能与应用场景

1.1 功能：

能准确实现下列功能识别：水果的颜色，水果的种类，水果的数量。

1.2 应用场景：

借鉴国际发展历史，水果智能识别技术必将越来越多的进入行业领域，因此，作为水果分级、分类识别中核心的计算机图像处理技术，可使分类检测实现全自动化、摆脱传统人工操作，可以为行业发展提供广阔空间，利用新兴的先进技术手段和方法可以大大提高我国水果产后的处理水平,提高生产效率,提升我国水果国际竞争力。水果图像分类还可以广泛应用于商业销售环节，比如自动称重，自动计价，提供便捷的人机交互方式，减轻人工负担等。

2 模块内容及算法介绍

2.1 物体识别系统架构:

物体识别系统架构如图 1 所示

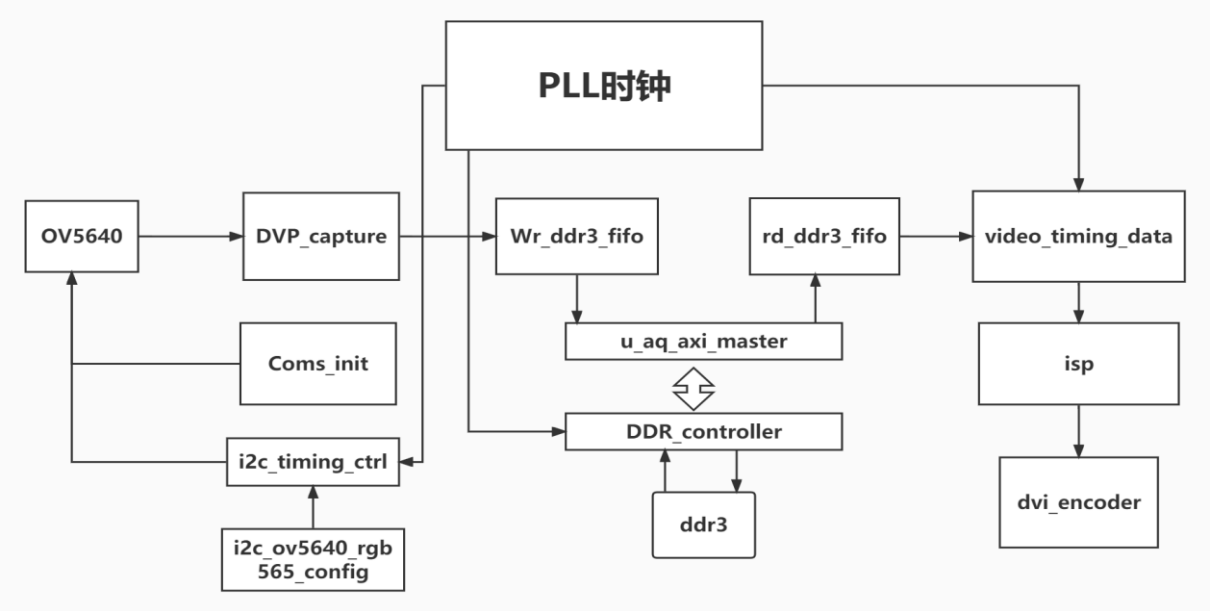


图 1 物体识别系统

PLL 时钟模块：时钟模块通过调用 PLL IP 核实现，共输出 5 个时钟，频率分别为 100M 时钟、400M 时钟、50M 时钟、65Mhz 时钟和 325M 时钟（HDMI 像素时钟的 5 倍频）。其中 pll 产生了 50M 时钟、100M 时钟和 400M 时钟，pll_hdmi 产生了 65Mhz 时钟和 325M 时钟。100Mhz 时钟作为 ddr 控制模块的驱动时钟，

400M 时钟用来输出给外部 ddr 芯片使用，50Mhz 时钟作为 I2C 驱动模块的驱动时钟，65Mhz 时钟和 325M 时钟（HDMI 像素时钟的 5 倍频）负责驱动 HDMI 顶层模块。

I2C 驱动模块（i2c_ctrl）：I2C 驱动模块负责驱动 OV5640SCCB 接口总线。

I2C 配置模块（i2c_ov5640_rgb565_config）：I2C 配置模块的驱动时钟是由 I2C 驱动模块输出的时钟提供的，这样方便了 I2C 驱动模块和 I2C 配置模块之间的数据交互。该模块寄存需要配置的寄存器地址、数据以及控制初始化的开始与结束，同时该模块输出 OV5640 的寄存器地址和数据以及控制 I2C 驱动模块开始执行的控制信号，直接连接到 I2C 驱动模块的用户接口，从而完成对 OV5640 传感器的初始化。

OV5640 模块：OV5640 摄像头模组采用美国 OmniVision(豪威)CMOS 芯片图像传感器 OV5640，支持自动对焦 的功能。OV5640 芯片支持 DVP 和 MIPI 接口, OV5640 摄像头模组通过 DVP 接口和 FPGA 连接实现图像的传输。

DVP_capture：OV5640 提供了一个 DVP 接口用来输出采集到的图像数据流，本文提供了一个将 DVP 接口的图像数据接收并转换为 RGB565 图像格式的控制逻辑，使用该控制逻辑，可以非常方便的将摄像头输出的图像数据以 RGB565 像素格式写入到 RAM 或 FIFO 中这是一个非常好用的 OV5640 摄像头数据流接口，不仅能够接收数据流，还能够实时输出每个像素的 X、Y 坐标，以供某些对图像像素位置有需求的应用中使用。

Wr_ddr3_fifo:将通过 DVP_capture 传输转换的数据流图像数据以 rgb565 像素写入 FIFO。

Rd_ddr3_fifo：读出从 DDR 中储存的数据。

U_aq_axi_master:通过外部 master 端口读取 DDR 数据。

dvi_encoder 模块：模块中包含两个模块 encode 和 serdes_4b_10to1 模块，实现 RGB 格式的图像转 化成 TMDS 差分输出，来驱动 HDMI 显示。

(1). encode: 红，绿，蓝的 8 位视频数据及时钟编码成 10 位的 TMDS 视频数据。

(2). serdes_4b_10to1:方法是通过两个 OSERDESE2 的串联把 10 比特的并行数据转换成串行发送出去。

DDR3_controller：DDR3 读写控制器模块负责驱动 DDR3 片外存储器，缓存

图像传感器输出的图像数据。

I2C 驱动模块（i2c_timing_ctrl）：I2C 驱动模块负责驱动 OV5640SCCB 接口总线。

Cmos_init：摄像头采集模块在像素时钟的驱动下将传感器输出的场同步信号、行同步信号以及 8 位数据转换成写使能信号和 16 位写数据信号，完成对 OV5640 传感器图像的采集。

Video_timing_data:该模块完成视频时序到 FIFO 读取的信号的转换，主要原理就是把视频时序中的“DE”做为 FIFO 的读信号，但是读出的数据会有延时，所以做了相应的对齐处理。

图像处理模块（ISP）：对采集后的图像数据进行处理。

2.2 图像处理模块各子模块算法（ISP）：

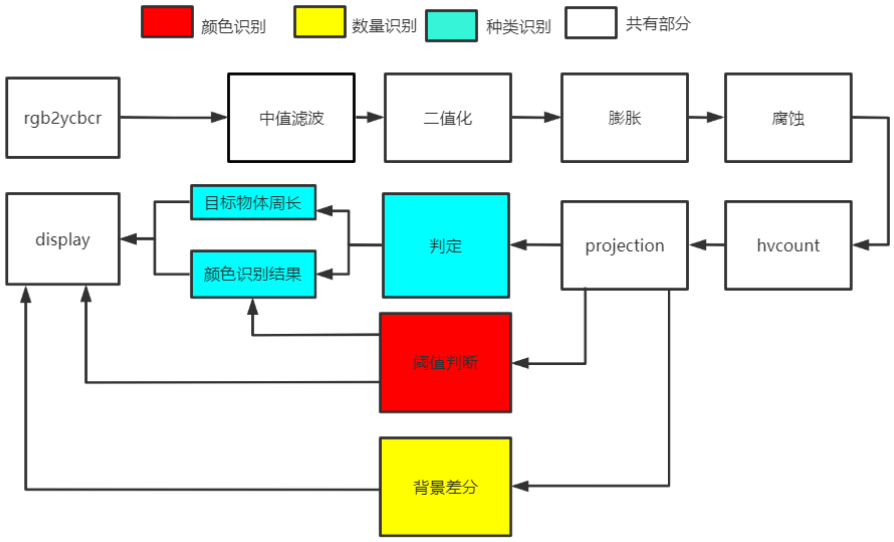


图 2 图像处理模块框图

2.2.1RGB565TOYCBCR

由于我们使用的颜色识别算法需要使用 YCBCR 的阈值进行颜色识别判定，所以我们的操作对象是水果的 YCBCR 图，摄像头 sensor 输出的是 rgb565 格式的，不符合颜色识别算法的前提条件。为了满足这一前提，需要对视频数据流进行实时的格式转换。

RGB 格式可以通过以下公式转化为 YCBCR 格式

$$Y = 0.257 * R + 0.564 * G + 0.098 * B + 16$$

$$CB = -0.148 * R - 0.291 * G + 0.439 * B + 128$$

$$CR = 0.439 * R - 0.368 * G - 0.071 * B + 128$$

式子中 R、G、B 分别为每个像素点红色、绿色、蓝色的量化值,考虑到 FPGA 并不擅长小数运算和乘法运算，同时为了化简设计难度，根据二进制数的性质进行移位加变换操作，替换乘除法并最后得到下面的公式。

$$Y = ((77 * R + 150 * G + 29 * B) >> 8)$$

$$CB = ((-43 * R - 85 * G + 128 * B) >> 8)$$

$$CR = ((128 * R - 107 * G - 21 * B) >> 8)$$

该模块采用三级流水线设计，耗时约为 46.2ns (摄像头的输出时钟为 65Mhz)。其中第一级流水用于计算式中的乘法，第二级流水用于计算式中的加法，第三级流水用于将 Y, CB, CR 这三个分量相加，采用寄存器输出。

2.2.2 高斯滤波算法

作用：高斯滤波是一种线性平滑滤波，适用于消除噪声，去除噪点，广泛应用于图像处理的减噪过程。

优异性：高斯滤波后图像被平滑的程度取决于标准差。它的输出是领域像素的加权平均，同时离中心越近的像素权重越高。因此，相对于均值滤波（mean filter）它的平滑效果更柔和，而且边缘保留的也更好。

具体操作：用一个模板（或称卷积、掩模）扫描图像中的每一个像素，用模板确定的邻域内像素的加权平均灰度值去替代模板中心像素点的值，在高斯滤波中，会将中心点的权重值加大，远离中心点的权重值减小，在此基础上计算邻域内各个像素值不同权重的和。

示例操作过程如下图：

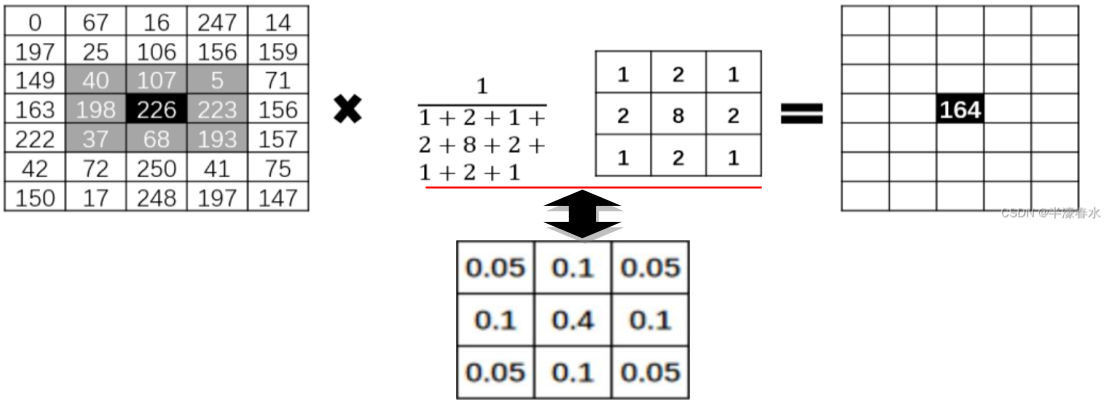


图 3 高斯滤波流程图

针对第 4 行第 3 列位置上的像素值为 226 的像素点进行高斯滤波处理，计算方式为：

=(40×0.05+107×0.1+5×0.05)+(198×0.1+226×0.4+223×0.1)

+(37×0.05+68×0.1+193×0.05)

=164

2.2.3 颜色识别

在预处理算法，获得水果 YCBCR 图像后，我们颜色识别的核心是通过颜色阈值比较的方法对赛题中八种水果的四种颜色进行识别，当水果的 YCBCR 值处于某个颜色范围时即给该颜色信号赋 1，否者为 0，通过这一过程即可完成水果的颜色识别，表 1 为我们通过 python 仿真得出的水果 YCBCR 值，部分代码如下：

```
assign en0=i_ycbcr[23:16]>=Y_TL_R&&i_ycbcr[23:16]<=Y_TH_R;
assign en1=i_ycbcr[15:8]>=CB_TL_R&&i_ycbcr[15:8]<=CB_TH_R;
assign en2=i_ycbcr[7:0]>=CR_TL_R&&i_ycbcr[7:0]<=CR_TH_R;
assign en0_1=i_ycbcr[23:16]>=Y_TL_B&&i_ycbcr[23:16]<=Y_TH_B;
assign en1_1=i_ycbcr[15:8]>=CB_TL_B&&i_ycbcr[15:8]<=CB_TH_B;
assign en2_1=i_ycbcr[7:0]>=CR_TL_B&&i_ycbcr[7:0]<=CR_TH_B;
assign en0_2=i_ycbcr[23:16]>=Y_TL_Y&&i_ycbcr[23:16]<=Y_TH_Y;
assign en1_2=i_ycbcr[15:8]>=CB_TL_Y&&i_ycbcr[15:8]<=CB_TH_Y;
assign en2_2=i_ycbcr[7:0]>=CR_TL_Y&&i_ycbcr[7:0]<=CR_TH_Y;
assign en0_3=i_ycbcr[23:16]>=Y_TL_BR&&i_ycbcr[23:16]<=Y_TH_BR;
assign en1_3=i_ycbcr[15:8]>=CB_TL_BR&&i_ycbcr[15:8]<=CB_TH_BR;
assign en2_3=i_ycbcr[7:0]>=CR_TL_BR&&i_ycbcr[7:0]<=CR_TH_BR;
always@(posedge pixelclk or negedgereset_n)begin
if(!reset_n)begin
red_en<=0;
yellow_en<=0;
black_en<=0;
brown_en<=0;
end
else begin
if(en0==1'b1&&en1==1'b1&&en2==1'b1)begin
red_en<=1;
end
else if(en0_1==1'b1&&en1_1==1'b1&&en2_1==1'b1)begin
black_en<=1;
```

```
end
else if(en0_2==1'b1&&en1_2==1'b1&&en2_2==1'b1)begin
yellow_en<=1;
end
else if(en0_3==1'b1&&en1_3==1'b1&&en2_3==1'b1)begin
brown_en<=1;
end
end
end
```

颜色	Y	CB	CR
红色	30~50	120~130	140~150
青色	70~110	80~100	100~115
紫色	20~30	120~130	120~130
黄色	85~110	70~90	120~140
乳白色	35~50	105~119	120~130
橙色	30~50	100~115	140~160
绿色	40~60	100~118	90~115

表 1 水果 YCBCR 值

颜色	水果			
红色	红苹果	火龙果	红橙子	红山竹
黄色	香蕉	黄梨	黄芒果	
青色	青苹果	青芒果		
棕色	猕猴桃			
紫色	葡萄			
栗色	无花果			
绿色	梨			
乳白色	雪梨			

2.2.4 二值化算法

我们使用一个固定的阈值对摄像头采集的图像数据进行二值化分割，经过我们大量测试，当阈值设为 80 时二值化效果最好，如图 4 所示，二值化效果图，当图像数据大于阈值时，给图像数据赋 0，反之赋值为 1。

图 4 二值化效果图

2.2.5 腐蚀与膨胀算法

为了修复二值图像中二值化图形中的坑坑洼洼，填补小裂缝，使其目标特征更加完备，我们设计了闭运算形态学滤波模块，即先膨胀后腐蚀的算法为实现闭运算，需要有至少连续三行的图像数据，为此要行缓存和列延迟。我们算法中行缓存采用 RAM 实现，延迟由 D 触发器实现，如图 5 所示.

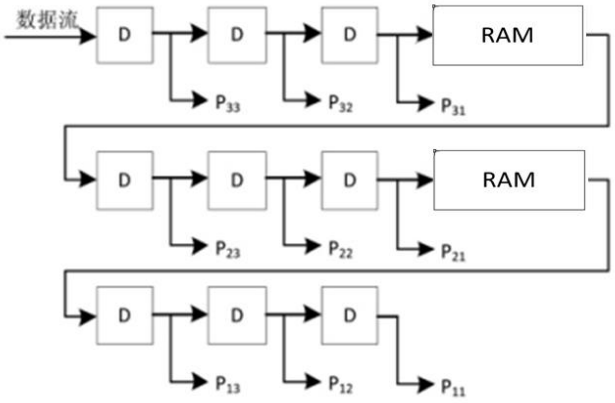


图 5 闭运算的行缓存与延迟

腐蚀运算为：$erode=P11\&P12\&P13\&P21\&P22\&P23\&P31\&P32\&P33$

即 9 个像素点值均为 1 时，腐蚀运算后的输出值为 1。

膨胀运算为： $dilate = P_{11}||P_{12}||P_{13}||P_{21}||P_{22}||P_{23}||P_{31}||P_{32}||P_{33}$

即 9 个像素点值有一个为 1 时，膨胀运算后的输出值为 1。

2. 2. 6hvcount

此为像素行列坐标生成模块，算法基本设计思想是使用 2 个计数器分别计算列坐标和行坐标，当列坐标计数器计数至最大列坐标时，行坐标值增 1

2. 2. 7projection

此模块为水平垂直投影算法，经过预处理后的图像已变成前后分明的二值化

图，图像数据只有 0 和 1，所以我们设计算法，水平方向投影即将图像数组进行列求和；垂直方向投影即将图像数组进行行求和；循环各行各列，依次判断每一列，每一行的像素值总和是否为一，当在水平方向投影时，列像素和由 0 变为 1 则判定为图像的左边界，列像素和由 1 变为 0 则判定为图像的右边界，垂直方向投影同理得到图像上下边界。

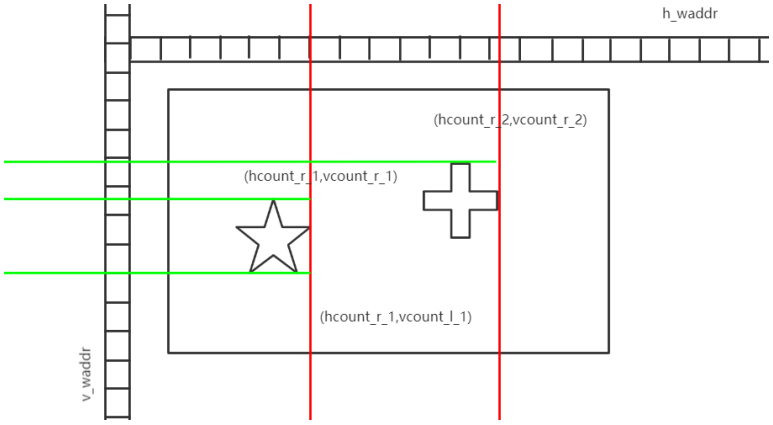
屏幕分割：

以精确的坐标为后续的种类识别和数量识别做铺垫。

屏幕分割法是在获取了边界坐标的基础上通过不同图像的边界坐标进行分割，当图像上存在两个目标时，以该图像的第一个目标右边界和第二个图像的右边界进行分割，其余的上下左右边界正常显示，这样就实现了两个屏幕的分割，多个目标的分割也同上所述。

创新点：能够避免因为外界环境的变化，使坐标的获取出现混叠或不断跳变的情况，对图像的坐标获取更准确。

如图示例：红色线代表屏幕分割线



2.2.8 种类识别

水果有颜色特征和形状特征之分，其中形状特征包括面积，周长等。我们采用了颜色结合面积与周长比的方式实现了种类识别，颜色的判断在前面的颜色识别模块已得出，对于面积和周长比我们将采用 SOBEL 边缘检测算法实现周长的统计，采用行列坐标乘积实现面积的计算，最终使用除法器对面积和周长做比值计算，根据不同的比值实现不同水果形状上的识别。

周长计算：由于图像并不是一维层面的长方形，正方形等图形，因此需采用边缘检测的方法来实现周长的计算，而在众多的边缘检测方法中，SOBEL 边缘检测对图像的效果最好。我们将灰度化处理之后的图像通过行缓存得到图像的 3X3 矩阵，将该矩阵的像素点与 SOBEL 算子进行卷积，求绝对值得到边界像素点，如果出现边界像素点则把该点值赋为‘1’，否则赋为‘0’，对每一个边界像素点进行计数累加，得到的累加值就是该图像的周长。

面积计算：由于外界环境的影响和图像本身的面积计算量庞大，资源消耗较大，因此我们通过计算水果由水平垂直得出的各对应边界坐标差来计算，若此时只出现一个图像，则我们通过该图像的四个坐标差可以得到该图像长和宽，对长和宽做乘积运算，便可以得到该图像的面积。

比值计算：采用可以计算商和余数的除法器，对面积和周长作比，得到不同的商和余数。

创新点：不同的水果图像可以得到不同的商和余数，我们通过这两个值的大小和颜色识别，实现水果的种类识别。

部分代码如下：

表 2 各水果边界坐标差

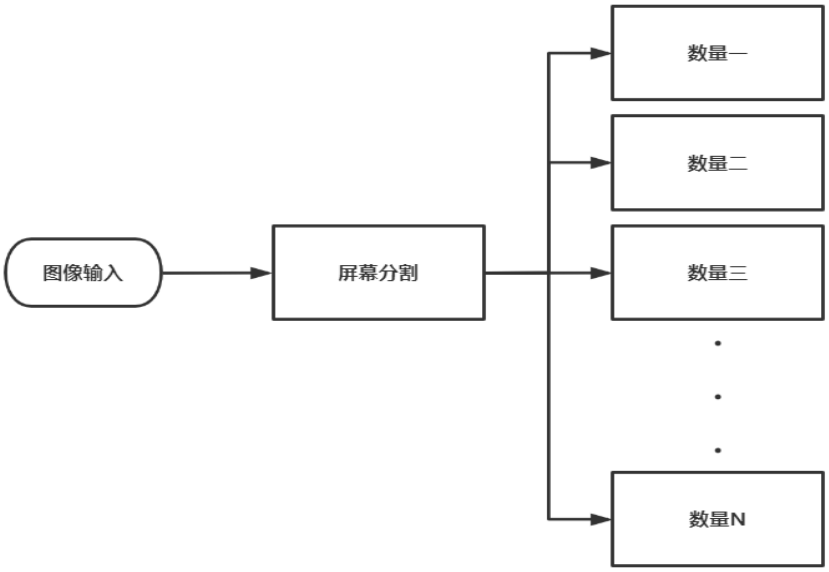
水果	左右边界坐标差	上下边界坐标差
芒果	213	144
苹果	282	286
香蕉	509	154
梨子	418	300
葡萄	482	183
橙子	204	215
火龙果	413	407
猕猴桃	357	219

2.2.9 数量识别

采用分屏算法对屏幕中的图像进行多次分割。

我们之前通过分割算法已经得到了一个图像的数据，要实现多数目的识别，我们将水平垂直投影得到的每一个图像的右边界为分割依据，进行多次分割，便识别出了多个水果。

如图所示：



为了简化计算量，我们设计的算法是通过物体左右边界和上下边界的坐标差大小来判断画面中是否存在物体并进行计数，经过测试当左右边界和上下边界的坐标差值都大于 50 则判定物体存在并进行计数，部分代码如下：

```
always@(posedge pclk)begin
cnt_r1<=hcount_r1-hcount_l1;
cnt_r2<=vcount_r1-vcount_l1;
cnt_r3<=hcount_r2-hcount_l2;
cnt_r4<=vcount_r2-vcount_l2;
cnt_r5<=hcount_r3-hcount_l3;
cnt_r6<=vcount_r3-vcount_l3;

if(cnt_r1>50&&cnt_r2>50)begin
o_num<=1;
if(cnt_r3>50&&cnt_r4>50)begin
o_num<=2;
if(cnt_r5>50&&cnt_r6>50)begin
o_num<=3;end
else
o_num<=o_num;
end
else
o_num<=o_num;
end
else
o_num<=o_num;
```

end

2.2.10display

此模块我们设计的算法是通过种类识别模块，颜色识别模块，数量识别模块的输出判定信号，来进行具体的识别结果字符输出，并且通过画框操作对每个水果进行位置确定。

3 子模块接口列表与接口时序

3.1RGB2YCBCR

3.1.1 接口列表

RGB2YCBCR			
input		output	
pixelclk	输入时钟信号	O_rgb	输出 rbg 图像数据
Rst_n	复位清零信号	O_ycbcr	输出 ycbcr 图像数据
I_rgb	输入图像数据	O_hsync	输出行同步信号
I_hsync	输入行同步信号	O_vsync	输出场同步信号
I_vsync	输入场同步信号	O_de	图像使能信号
I_de	图像使能信号		

3.1.2 接口时序

此模块使用的算法使用了三级流水，所以输出信号 O_hsync，O_vsync，O_de 相较于输入信号 i_hsync，i_vsync，i_de 延时了三个时钟，当 o_hsync，o_vsync 信号同时为 1 时开始输出图像数据。

3.2 中值滤波

3.2.1 接口列表

中值滤波

input		output	
clk	输入时钟信号	O_rgb	输出处理后图像数据
rst_n	复位清零信号	Pos_frame_vsync	输出场同步信号
pe_img_y	输入灰度图像数据	Pos_frame_href	输出行同步信号
Pe_frame_href	输入行同步信号	Pos_frame_clken	图像使能信号
Pe_frame_vsync	输入场同步信号		
Pe_frame_clken	图像使能信号		

3.2.2 接口时序

此模块中使用两个周期进行图像行缓存，使用三个周期进行图像数据中值计算，所以输出信号 Pos_frame_vsync，Pos_frame_href，Pos_frame_clken 相较于输入信号 Pe_frame_vsync，Pe_frame_href，Pe_frame_clken 延时了五个时钟，当 Pe_frame_href，Pe_frame_vsync 信号同时为 1 开始输出图像数据。

3.3 二值化

3.3.1 接口列表

二值化			
input		output	
pixelclk	输入时钟信号	O_rgb	输出处理后图像数据
Reset_n	复位清零信号	O_hsync	输出行同步信号
I_ycbcr	输入 ycbcr 图像数据	O_vsync	输出场同步信号
I_rgb	输入 rgb 图像数据	O_de	图像使能信号
I_hsync	输入行同步信号		

I_vsync	输入场同步信号		
I_de	图像使能信号		

3.3.2 接口时序

此模块仅使用一个时钟进行阈值判断，所以输出信号 o_hsync,，o_vsync，o_de 相较于输入 i_hsync，i_vsync，i_de 信号延时了一个时钟，当 o_hsync，o_vsync 信号同时为 1 时开始输出图像数据。

3.4 膨胀

3.4.1 接口列表

膨胀			
input		output	
Clk	输入时钟信号	Dilation_frame_vsync	输出场同步信号
Rst_n	复位清零信号	Dilation_frame_href	输出行同步信号
Per_frame_vsync	输入场同步信号	Dilation_frame_clken	图像使能信号
Per_frame_href	输入行同步信号	Dilation_img_Bit	输出处理后图像数据
Per_frame_clken	图像使能信号	o_rgb	输出 rgb 图像数据
I_rgb	输入 rgb 图像数据		
Per_img_Bit	输入处理后图像数据		

3.4.2 接口时序

此模块使用两个周期进行图像行缓存，使用两个周期进行膨胀计算，所以输出信号 Dilation_frame_vsync，Dilation_frame_href，Dilation_frame_clken 相较于输入信号 per_frame_vsync，per_frame_href，per_frame_clken 延时了四个时钟，当 Dilation_frame_href,Dilation_frame_vsync 信号同时为 1 时开始输出图像数据

3.5 腐蚀

3.5.1 接口列表

腐蚀			
input		output	
pclk	输入时钟信号	Erosion_out_bit	输入时钟信号
rst	复位清零信号	Erosion_out_hs	复位清零信号
Erosion_in_bit	输入处理后图像数据	Erosion_out_vs	输入处理后图像数据
Erosion_in_hs	输入行同步信号	Erosion_out_de	输入行同步信号
Erosion_in_vs	输入场同步信号	O_rgb	输入场同步信号
Erosion_in_de	图像使能信号		
I_rgb	输入 rgb 图像数据		

3.5.2 接口时序

此模块使用两个周期进行图像行缓存，使用两个周期进行膨胀计算，所以输出信号 Erosion_out_vs，Erosion_out_hs，Erosion_out_de 相较于输入信号 Erosion_in_vs，Erosion_in_hs，Erosion_in_de 延时了四个时钟，当 Erosion_in_hs，Erosion_in_vs 信号同时为 1 时开始输出图像数据

3.6hvcount

3.6.1 接口列表

hvcount			
input		output	
Pixelclk	输入时钟信号	Vcount	输出列坐标
Reset_n	复位清零信号	Hcount	输出行坐标
I_data	输入 rgb 图像数据	O_data	输出 rbg 数据
I_binary	输入处理后图像数据	O_bianry	输出
I_hsync	输入行同步信号	O_hsync	输出行同步信号
I_vsync	输入场同步信号	O_vsync	输出场同步信号

I_de	图像使能信号	O_de	图像使能信号

3.6.2 接口时序

此模块使用一个周期进行行列计数器累加，所以输出信号 o_hsync,，o_vsync，o_de 相较于输入 i_hsync，i_vsync，i_de 信号延时了一个时钟，当 o_hsync，o_vsync 信号同时为 1 时开始输出像素行列坐标

3.7projection

3.7.1 接口列表

projection			
input		output	
Clk	输入时钟信号	Hcount_l1	第一个物体左边界
Rst_n	复位清零信号	Hcount_r1	第一个物体的右边界
I_binary	输入图像数据	Vcount_l1	第一个物体的上边界
I_hs	输入行同步信号	Vcount_r1	第一个物体的下边界
I_vs	输入场同步信号	Hcount_l2	第二个物体左边界
I_de	图像使能信号	Hcount_r2	第二个物体的右边界
I_hcount	输入行坐标	Vcount_l2	第二个物体的上边界
I_vcount	输入列坐标	Vcount_r2	第二个物体的下边界
		Hcount_l3	第三个物体左边界
		Hcount_r3	第三个物体的右边界
		Vcount_l3	第三个物体的上

			边界
		Vcount_r3	第三个物体的下边界

3.7.2 接口时序

此模块并未有输出行场同步信号接入下一模块

3.8 颜色识别

3.8.1 接口列表

颜色识别			
input		output	
clk	输入时钟信号	O_hsync	输出行同步信号
Rst_n	复位清零信号	O_vsync	输出场同步信号
I_rgb	输入 rgb 信号	O_de	图像使能信号
I_hsync	输入行同步信号	Red_en	红色判定信号
I_vsync	输入场同步信号	Yellow_en	黄色判定信号
I_de	图像使能信号	Black_en	黑色判定信号
	输入时钟信号	Blown_en	棕色判定信号

3.8.2 接口时序

此模块仅使用一个周期进行颜色识别，所以输出信号 o_hsync,，o_vsync，o_de 相较于输入 i_hsync，i_vsync，i_de 信号延时了一个时钟，当 o_hsync，o_vsync 信号同时为 1 时开始输出颜色判定信号。

3.9 数量识别

3.9.1 接口列表

数量识别	
input	output

Clk	输入时钟信号	O_hsync	输出行同步信号
Rst_n	复位清零信号	O_vsync	输出场同步信号
I_binary	输入图像信号	O_de	图像使能信号
I_hsync	输入行同步信号	O_num	物体数量
I_vsync	输入场同步信号		
I_de	图像使能信号		
I_Hcount_l1	第一个物体左边界		
I_Hcount_r1	第一个物体的右边界		
I_Hcount_l2	第二个物体的右边界		
I_Hcount_r2	第二个物体的上边界		
I_Hcount_l3	第三个物体左边界		
I_Hcount_r3	第三个物体的右边界		

3.9.2 接口时序

模块使用一个时钟进行边界坐标差计算，使用一个时钟数量识别，所以输出信号 o_hsync,，o_vsync，o_de 相较于输入 i_hsync，i_vsync，i_de 信号延时了两个时钟，当 o_hsync，o_vsync 信号同时为 1 时开始输出数量识别结果

3.10 种类识别

3.10.1 接口列表

种类识别	
input	output

Clk	输入时钟信号	O_hsync	输出行同步信号
Rst_n	复位清零信号	O_vsync	输出场同步信号
I_binary	输入图像信号	O_de	图像使能信号
I_hsync	输入行同步信号	en_1	芒果判定信号
I_length	输入场同步信号	en_2	橙子判定信号
i_de	图像使能信号	en_3	苹果判定信号
I_Hcount_l1	第一个物体左边界	en_4	香蕉判定信号
I_Hcount_r1	第一个物体的右边界	en_5	葡萄判定信号
I_vcount_l1	第一个物体的上边界	en_6	猕猴桃判定信号
I_vcount_r1	第一个物体的下边界	en_7	火龙果判定信号
I_Hcount_l2	第二个物体左边界	en_8	梨子判定信号
I_Hcount_r2	第二个物体的右边界		
I_vcount_l2	第二个物体的上边界		
I_vcount_r2	第二个物体的下边界		

3.10.2 接口时序

此模块使用一个周期进行边界坐标差计算，一个周期进行种类识别，，所以输出信号 o_hsync,，o_vsync，o_de 相较于输入 i_hsync，i_vsync，i_de 信号延时了两个时钟，当 o_hsync，o_vsync 信号同时为 1 时开始输出水果判定信号。

3.11display

3.11.1 接口列表

display

input		output	
Pixeclk	输入时钟信号	O_rgb	输出图像信号
Reset_n	复位清零信号	O_hsync	输出行同步信号
Red_en	红色判定信号	O_vsync	输出场同步信号
Yellow_en	黄色判定信号	O_de	图像使能信号
Black_en	黑色判定信号		
Brown_en	棕色判定信号		
I_rgb	输入图像信号		
I_hsync	输入行同步信号		
I_vsync	输入场同步信号		
I_de	图像使能信号		
Hcount	输出列坐标		
Vcount	输出行坐标		
Hcount_l1	第一个物体左边界		
Hcount_r1	第一个物体的右边界		
Vcount_l1	第一个物体的上边界		
Vcount_r1	第一个物体的下边界		
Hcount_l2	第二个物体左边界		
Hcount_r2	第二个物体的右边界		
en_1	芒果判定信号		
en_2	橙子判定信号		

en_3	苹果判定信号		
en_4	香蕉判定信号		
en_5	葡萄判定信号		
en_6	猕猴桃判定信号		
en_7	火龙果判定信号		
en_8	梨子判定信号		

3.11.2 接口时序

此模块使用一个周期进行画框操作，使用一个周期进行字符显示操作，所以输出信号 o_hsync,，o_vsync，o_de 相较于输入 i_hsync，i_vsync，i_de 信号延时了两个时钟，当 o_hsync，o_vsync 信号同时为 1 时开始输出图像数据

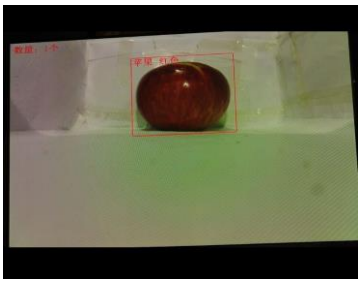
4 水果识别系统硬件验证

4.1 测试效果展示

系统由展示台、PGL22G 开发板、OV5640 摄像头、HDMI 显示屏及各类水果组成。

测试效果图如下：

苹果：

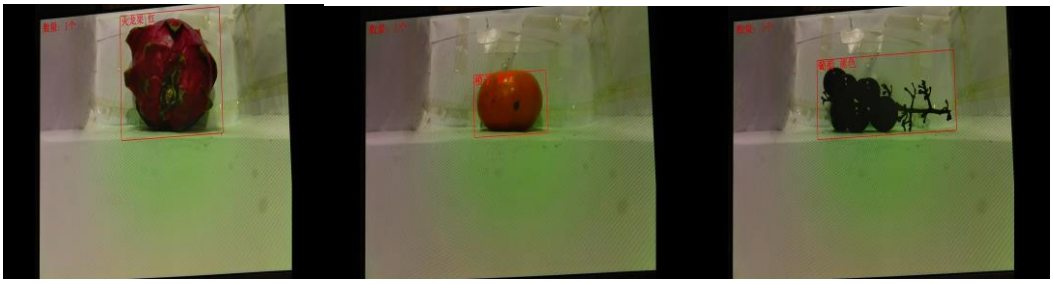


苹果，芒果：

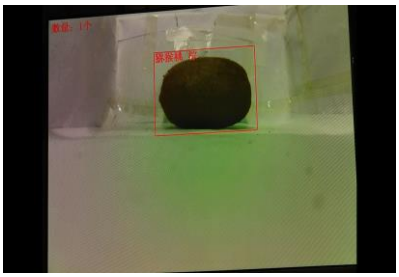
葡萄 ：

橙子 ：

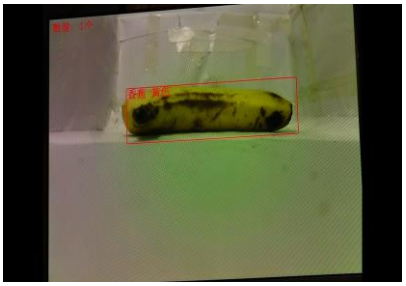
火龙果：



猕猴桃：



香蕉：



4.2 水果识别结果

目前我们的系统能够实现五个以下数量水果的颜色和种类识别并且识别的效果速度都较好，如表 3 所示。具体识别结果见设计数据文档

表 3 水果识别结果

水果种类	识别速度	识别率
苹果	0.37ms	98.00%
香蕉	0.37ms	99.00%
葡萄	0.37ms	98.50%
火龙果	0.37ms	98.00%
梨子	0.37ms	98.00%
芒果	0.37ms	98.00%
猕猴桃	0.37ms	98.50%
橙子	0.37ms	99.00%