

第六届

全国大学生集成电路创新创业大赛

报告类型\*： 设计方案

参赛杯赛\*： 紫光同创杯

作品名称\*： 基于PGL22G的MobileNet-YOLO水果识别系统

队伍编号\*： CICC1084

团队名称\*： 点灯稳过

目录

[1 项目背景 1](#_Toc104819047)

[2 系统功能介绍 1](#_Toc104819048)

[3 系统架构 2](#_Toc104819049)

[3.1 IO设备设计 2](#_Toc104819050)

[3.2 CNN加速电路接口时序 4](#_Toc104819051)

[3.3 KNN实时颜色识别电路 5](#_Toc104819052)

[3.4 图像缓存 5](#_Toc104819053)

[4 算法设计 6](#_Toc104819054)

[4.1 改进的YOLO-MobileNet算法 6](#_Toc104819055)

[4.2 基于KNN的水果颜色识别算法 9](#_Toc104819056)

[5 子模块设计 11](#_Toc104819057)

[5.1 硬件加速模块 11](#_Toc104819058)

[5.1.1 CBL结构 11](#_Toc104819059)

5.1.1.1 Relu层 [.......................................](#_5.1.1.1__Relu层) 11

5.1.1.2 卷积层 [........................................](#_5.1.1.2_卷积层) 13

[5.1.2 线性层 20](#_Toc104819060)

[5.1.3 上采样层 20](#_Toc104819061)

[5.1.4 最大池化层 22](#_Toc104819062)

[5.2 颜色识别模块 28](#_Toc104819063)

[5.2.1 KNN结构 28](#_Toc104819064)

[5.3 前处理与后处理模块 33](#_Toc104819065)

[6 设计参数 34](#_Toc104819066)

# 项目背景

随着我国经济水平与消费水平的增长，人民对于美好生活的需要也日益增长，这其中就包括了对于各种水果的需求，因此也大大增加了物流分装过程中的水果筛选需求。但我国目前的水果筛选与分装仍停留在人工和半机械阶段，且易受到环境、视力、疲劳的影响，效率较低。因此，一个高识别率与高效率的水果识别系统是十分必要的。

而当今市场中的水果识别系统，大多是基于传统的CV、ML算法，通过识别色块、获取轮廓、分割前后景并对图片进行适当调整，从而提取图片特征，匹配与之最接近模型，识别目标水果。这种方法易受背景、水果图片特征混淆干扰，准确度低，不适用于大量水果的识别与商业化的推广。

基于这些问题，为了提高识别率、加快识别速率，本项目针对算法和FPGA进行创新，基于紫光同创PGL22G平台进行CNN水果识别算法的FPGA部署，并将其应用于适应复杂光线变化的物流分装中的水果筛选。

# 系统功能介绍

本水果识别系统实现了1024\*768p识别视频输入和串口数据输出的全流程，包括水果数量识别、水果颜色识别、水果类别识别三个功能：

在水果数量识别方面，系统能够识别10个以下水果的数量，并通过串口输出识别结果数字。

在水果颜色识别方面，系统能够识别水果的颜色，并根据结果指定串口输出RED、ORANGE、YELLOW、GREEN、PURPLE、WHITE、BROWN、MAROON九种颜色单词。

在水果类别识别方面，系统能够识别苹果、香蕉、葡萄、火龙果、梨子、芒果、猕猴桃、橙子、山竹、无花果十种水果，并通过串口输出识别结果APPLE、BANANA、GRAPE、PITAYA、PEAR、MANGO、KIWI、ORANGE、STEEN、FIG十种水果类别单词。

接下来举例系统对于葡萄的识别过程：

1. 将葡萄放置在开发板摄像头前；
2. 打开设备电源，连接串口，上位机将收到空数据；
3. 首先按下板载的KEY4按钮，对应LED4亮起，权重逐次从SD卡加载到设备，识别开始；
4. 按下板载的KEY2或KEY3按钮可切换识别模式，分别有颜色识别模式、水果种类识别模式和水果数量识别模式；
5. 模型训练时将一串葡萄的水果数量设置为1，因此得到结果：颜色为紫色（purple）、种类为葡萄（grape）、数量为1；

具体操作可参考附件中的演示视频

# 系统架构

本项目基于紫光同创PGL22G开发板设计，全部使用开发板板载资源实现。图像数据通过OV5640摄像头输入，优先缓存到片上资源实现的RAM中，超过容量输入的图片数据会被缓存到片外DDR等待读取。数据首先经一个并行化接口，从串行的RAM输出数据转换成宽位并行的帧数据输入到CNN加速电路和KNN实时颜色识别电路，一套按钮状态机处理板上控制按钮按下情况并对数据流进行分配。KNN电路输出一个颜色识别码到串口模块，串口模块会对颜色识别码进行译码处理并输出具体的颜色；CNN电路输出一个种类识别码到串口模块，同时输出一个识别框坐标序列到计数器模块，串口模块直接译码出种类发送到上位机；计数器模块记录了一批次输入图像的识别状态，并对识别框序列进行计数，从而得到每帧图像上的水果数量。系统还会通过一个数据选择器输出摄像头实时输出或经过CNN电路检测目标后的图像数据，由一个HDMI控制器进行输出。系统架构框图如图3.1所示。

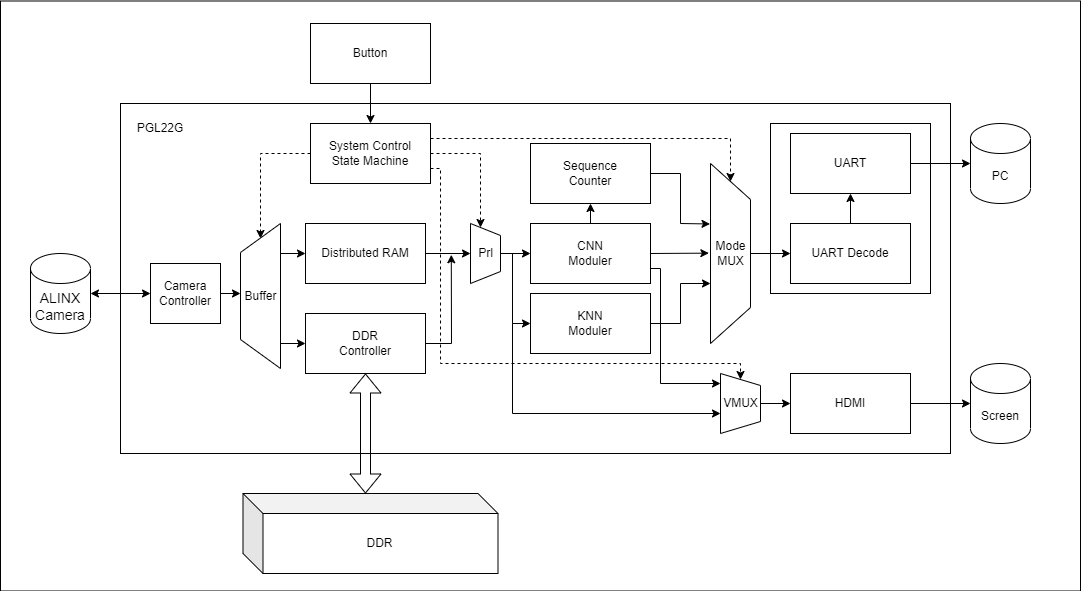


图3.1

## IO设备设计

设备搭载了摄像头模块、MIG-AXI转换模块、SD卡控制模块、按键处理模块、HDMI输出模块、串口输出模块用于数据的输入输出。

摄像头模块采用典型的DVP摄像头总线驱动电路和SCCB总线驱动电路，摄像头数据输入后会通过一个摄像头控制器异步FIFO（DCMI\_FIFO）与主体程序耦合，同时避免了跨时钟域数据传输问题。

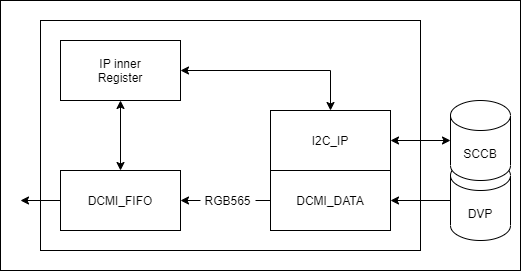


图3.1.1

MIG-AXI转换模块使用紫光同创提供的MIG IP实现，接口时序与官方数据手册一致。

按键处理模块主要由按键消抖子模块和按键状态机子模块组成，按下开发板上按钮后，按键消抖子模块滤除按键抖动，输出一个高电平脉冲，按键状态机采集脉冲并调整当前系统运行模式。设备内部所有数据流多路选择器完全由这个状态机控制。除了既定的三种识别模式外，还设置了低功耗运行的无输出模式，该模式下CNN、KNN模块都不会启动，系统只会采集摄像头数据并通过HDMI接口输出。

HDMI输出模块负责将原始视频数据或来自CNN模块的处理后数据转换成HDMI时序，采用了来自黑金ALINX的HDMI实现。

SD卡控制模块用于读取保存在SD卡上的模型权重信息。进入识别模式后，CNN模块中每一层算子电路会在需要工作时读取SD上的权重信息，根据对应权重进行运算。SD卡控制模块包含SDIO子模块、权重识别子模块、识别码解析子模块。CNN模块中每一层都被分配了一个8位的识别码，识别码解析子模块会对发出申请的算子ID进行译码，权重识别子模块找到其需要的权重，再向算子发送允许信号，算子方可进行计算。

串口输出模块由串口时序发生模块、串口FIFO、串口译码模块组成。串口译码模块根据当前系统工作模式，接收来自序列计数器、CNN模块、KNN模块的水果种类、颜色或数量指示数据，译码成可读的ASCII码后通过串口发送到上位机。

## CNN加速电路接口时序

CNN加速电路模块和外围模块连接中采用了简单快速的valid-ready握手协议，内部采用了易于实现的高电平触发信号协议，可以高速实现对INT8整型的运算。

CNN模块内部，每一层算子完成运算后都会发送一个时钟周期的高电平到下一层算子，下一层算子在接收到高电平后会从上一层算子中接收输出的数据，等待来自SD卡的权重输入，直到权重输入完成后再运算，上一层的输出数据在此期间将保持直到下一层的忙信号拉高。大部分算子都属于CBL算子，包含实现卷积运算的时序电路和实现ReLU运算的组合逻辑电路，因此延迟最低可以达到一个机器周期（即一层卷积电路对一副图片进行运算的时间）。在时序通路中稍加优化即可达到较好的时序效果，同时能够实现流水线操作。

大部分卷积层都需要从片外SD卡读取权重。当其收到上一层电路发送的高电平触发时，会优先发送请求信号到SD卡模块，等待权重加载。SD卡模块会成批加载权重到CNN电路以提高速率。权重加载完毕后，该层向上一层发送忙信号。

CNN加速电路内部接口中使用到的端口如表3.2.1所示

表3.2.1 CNN模块内部接口时序表

|  |  |
| --- | --- |
| 端口名 | 功能 |
| work\_en 计算使能 | 接收上一层的计算完成触发信号 |
| work\_fin 计算完成 | 发送到下一层的计算完成触发信号 |
| busy 忙信号 | 标志本层正在进行运算的触发信号 |

CNN模块与外围模块使用宽位并行接口连接，使用简单的valid-ready握手协议进行通信。从并行数据转换器输出的图像信号，首先转换为RGB565格式，按行同步输入CNN模块的前处理子模块，完成图像缩放，变换成YOLOv4能够处理的416\*416像素RGB888数据。完成转换后，前处理子模块向CNN控制器子模块发送ready信号，CNN控制器子模块根据前一帧图像的运算情况选择抛弃下一帧图像或将下一帧图像输入CNN加速器，若决定输入图像则会输出valid信号，并完成数据读取。

外围模块之间的数据通路也是用valid-ready握手协议，当模块空闲时，会向要读取数据的模块发送ready信号，若此时数据发送模块valid信号拉高，表示数据有效，后一模块将数据读取到内部进行处理。

表3.2.2 CNN模块外部接口时序表

|  |  |
| --- | --- |
| 端口名 | 功能 |
| valid 允许输出信号 | 拉高代表数据可信，能够读取 |
| ready 准备完成信号 | 拉高代表可以输入数据 |

## KNN实时颜色识别电路

KNN电路内部中使用了valid-ready时序，外部使用VGA时序。在KNN模块信号输入端，前处理模块会通过VGA时序读取来自并行数据转换器的图像信号，完成RGB-HSV转换和直方图统计，输出数据通过上述valid-ready协议传输到KNN电路。

详细内容将在KNN子模块中介绍。

## 图像缓存

摄像头数据输入后会根据当前运行模式和数据处理情况被缓存到片上DRAM或片外DDR，该过程由按键状态机输出的模式控制码和FIFO状态控制，如表3.4.1所示。

表3.4.1 图像缓存选择

|  |  |  |
| --- | --- | --- |
| 系统工作模式 | FIFO状态 | 缓存目标 |
| 任意 | 满 | DDR |
| 水果颜色识别 | 空 | DRAM |
| 水果种类识别、数量识别 | 空 | DDR |

# 算法设计

团队对现有的三种主流目标识别算法进行分析，发现传统上基于图像的机器视觉算法存在准确度低、难以适应复杂光照环境的缺点；基于最新研究的脉冲神经网络（SNN）或视觉Transformer（ViT）的神经网络算法存在实现难度过高、片上资源消耗巨大的缺点；而近些年来占据视觉神经网络主流的卷积神经网络很好地平衡了二者。三者的特点列举如下：

|  |  |  |
| --- | --- | --- |
| 算法 | 主要优点 | 主要缺点 |
| 传统CV算法 | 实现简单，易于部署 | 抗干扰能力差 |
| 卷积神经网络 | 泛化能力强，可适应更多环境 | 片上资源消耗大，实现难度大，需要额外采集数据集 |
| ViT、SNN等 | 准确度高、前景广阔 | 片上资源消耗大，实现难度极高，需要额外采集数据集 |

综上，项目选择在FPGA上移植适合的卷积神经网络模型作为主体算法，但是对于颜色识别这样传统算法就可以很好完成的任务；考虑到神经网络算法的训练过程中需要使用大量人工标注的数据集，如果将颜色识别功能一并作为神经网络的分类输出，无疑会导致数据集需求量倍增，因此团队将颜色识别功能独立出来，在主神经网络电路旁挂一个K最近邻网络（KNN）电路，使用传统的机器学习算法来达到更好的效果和更小的工作量，充分利用片上资源。

## 改进的YOLO-MobileNet算法

项目设计过程中，团队首先基于PyTorch，使用Python高级语言在已有实现上针对FPGA环境和项目应用改进了YOLOv3算法，主要改进点包括：

1. 将Backbone部分从ResNet-53替换为MobileNetv2，大幅减少资源消耗；
2. 将YOLOv4的YoloBody部分移植到YOLOv3，并将主体中所有使用了传统3x3卷积核的CBL块改为使用1x1卷积核的inverse CBL块；
3. 将每个特征图输出的五层卷积都改为四层，减少一定资源消耗；
4. 将上采样阶段多余卷积层删除，只留下基本的上采样层；
5. 借鉴YOLOv4的masaic数据增强、KNN预规划选择框等技巧，并使用基于低阶矩的自适应估计的Adam算法进行训练。

经过上述改进，本项目中的算法（称为Yolo-MobileNet）最终在PC测试平台（Nvidia RTX 2060）上达到约27帧的识别速率；官方的YOLOv3-tiny实现能够达到相同速率，但准确度大不如本模型；YOLOv4-tiny实现的准确度与本模型相差约10%；YOLOv4-n实现则只能达到约20帧的速率。同时模型所需的网络总大小减少到7710.43MB，相比原版YOLOv4-tiny减少了2000MB左右。网络权重大小增加了约20MB，但所需的网络层数大大减少了，有利于FPGA端部署。

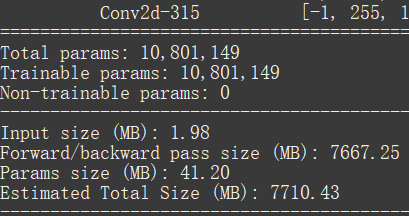


图4.1.1 改进版YOLOv3

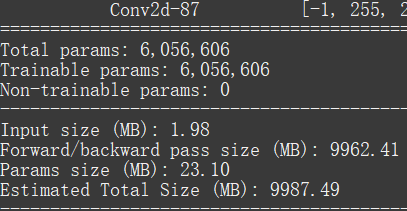


图4.1.2 原版YOLOv4-tiny

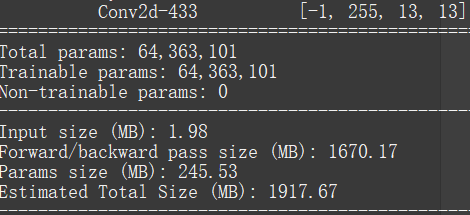


图4.1.3 原版YOLOv4-n

项目算法的整体优化思路即舍弃部分实时性和参数大小，换取更高的算法效率，从而让FPGA上的电路实现更节省片上资源，达到相对更高的硬件效率；同时尽量避免使用不同大小的卷积层，从而让硬件电路可以在相对简单的时序条件下形成全流水，加快图像处理速度。

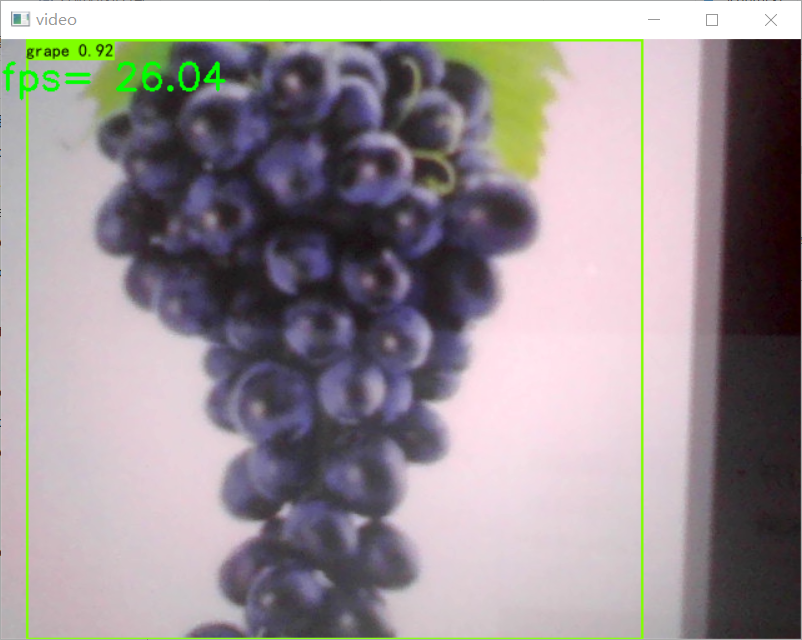


图4.1.4 算法实测效果（摄像头前放置水果图片进行测试）

模型训练后得到的参数经过ONNX INT8量化，转换为便于FPGA运算的整数，权重通过文件的形式保存到SD卡上，FPGA实时读取加载SD卡上的权重，根据运算流程，分配到到各层算法电路。图片在处理过程中先被分散为数个帧，多个通道的卷积核会对其并行处理。量化过程中也实现了卷积层和BN层合并，进一步简化了网络实现。

## 基于KNN的水果颜色识别算法

项目的FPGA实现中还与YOLO电路一并实现了一套KNN水果颜色识别算法。算法参数由上位机预训练结果提供，由于权重较小，将其转换成coe格式，使用FPGA的片上存储资源进行存储，在系统启动后自动加载。

整体算法数据流如图4.2.1所示，摄像头数据会被优先存入DRM，如果存放不及时，数据会被缓存到外部DDR中。片上存储的数据会优先转换成并行模式分发给预处理模块和KNN模块，由一个按钮控制状态机负责系统识别状态切换。

进入颜色识别模式下，KNN模块的参数被加载，首先进行RGB到HSV颜色空间的转换，并对图片像素信息逐行进行直方图统计，直方图统计结果经KNN电路运算后可以输出最接近的颜色。

在水果种类识别和数量识别模式下，主要任务由YOLO神经网络模块完成，图像帧会受输出情况控制，从而实现算法流水处理，YOLO神经网络输出的数据分为水果种类信息和水果数量信息，种类信息会被直接输出到串口和HDMI接口，数量信息由YOLO输出预测框的序列组成，会被送入一个计数模块，对YOLO输出预测框的数量计数后得到当前帧图片内的水果数量。

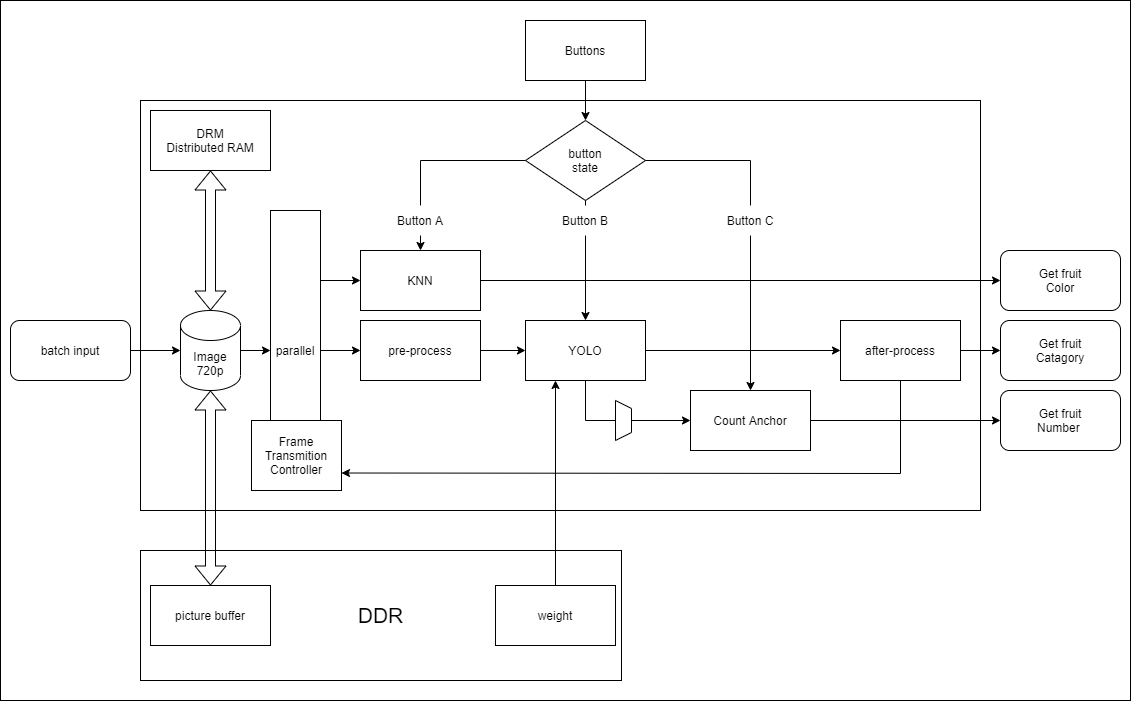


图4.2.1 算法数据流图

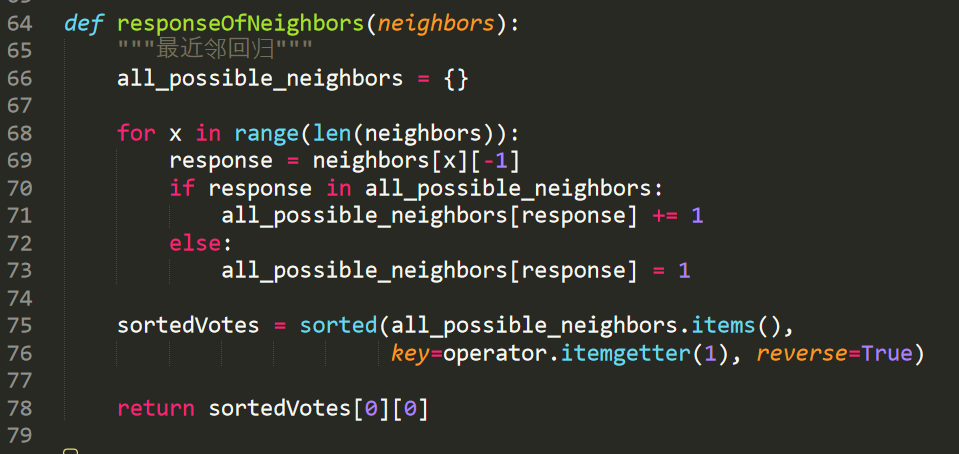


图4.2.2 关键代码



图4.2.3 算法测试效果

# 子模块设计

## 硬件加速模块

### CBL结构

#### 5.1.1.1 Relu层

本层的作用是实现Relu线性运算，即实现分段函数：

在设计中，此模块由relu\_top、relu两个模块组成，接下来将对这两个模块的主要功能进行描述。

relu\_top模块作为relu模块的顶层模块，对relu模块进行例化调用。其主要功能对上一级模块输出的数据进行预处理，同时对从relu模块输出的数据进行后处理，最终输出给下一级模块。

relu\_top模块的输入输出端口如图5.1.1.1。其中en\_in是Relu层接收上一级工作完成后发来的工作使能信号，en\_out是Relu层在结束工作时给输出给下一级的工作使能信号。in\_data是上一级输出数据的并行输入端口，result是输出给下一级的并行输出端口。relu\_top模块的预处理功能就是将并行输入的数据在时序的控制下分块传给relu模块，同样将relu模块输出的数据整合成并行数据进行输出，具体实现将使用数据选择器。relu\_top模块中对relu模块的例化如图5.1.1.2,indataArray和resultArray是输入和输出数据的储存缓冲区。

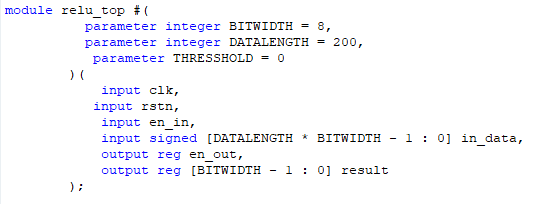


图5.1.1.1

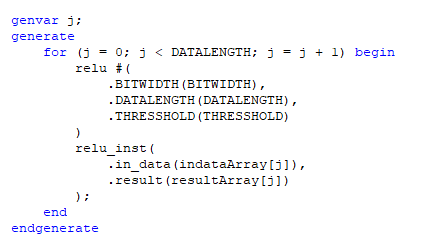


图5.1.1.2

relu模块代码如图5.1.1.3所示，在设计采用数据选择器电路对Relu功能进行实现。

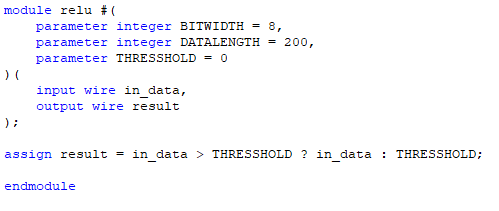


图5.1.1.3

#### 5.1.1.2 卷积层

本层的作用是完成卷积运算。

卷积层由四个模块组成，其分别是:conv\_top、conv\_buffer、conv\_search、conv\_cal组成，同时还例化了pds提供的IP核Multiply-Accumulator作为卷积计算核心。conv\_top作为卷积层的顶层模块，例化conv\_buffer、conv\_search、conv\_cal三个子模块，并完成三个模块间的信号调度。conv\_buffer模块是卷积层的储存缓冲区。conv\_search模块是conv\_buffer储存缓冲区数据的检索模块。conv\_cal模块是卷积层的计算模块，对输入数据完成卷积并且输出相应的数据。接下来将对此四个模块具体功能进行描述。

conv\_top模块主要完成三个子模块的互联并进行信号调度，其输入输出端口如图5.1.1.4所示，img带卷积数据，weight则是权重数据，bias是偏置数据，conv\_en是卷积层接收上一级工作完成后发来的工作使能信号，conv\_fin是卷积层在结束工作时给输出给下一级的工作使能信号。

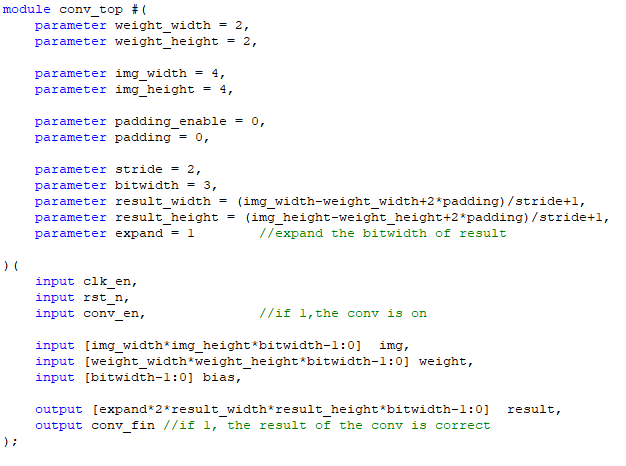


图5.1.1.4

conv\_buffer模块的主要功能完成对权重数据的静态缓存以及对待卷积数据的动态缓存，其输入输出端口如图5.1.1.5所示，anchor\_l、anchor\_c是定位坐标，用于定位待卷积数据的输入起始位置;buf\_l、buf\_c是缓冲区数据检索地址;img\_cal、wei\_cal则是输出卷积计算所需的数据。

待卷积数据的动态缓冲区的构建如图5.1.1.6所示，权重数据的静态缓冲区的构建如图5.1.1.7所示。设计中搭建了数据选择器电路来完成数据的更新与定向提取，动态缓冲区的设计则可以有效的减少片内占用资源，该缓冲区数据的定位则依赖于anchor\_l、anchor\_c两组输入信号。

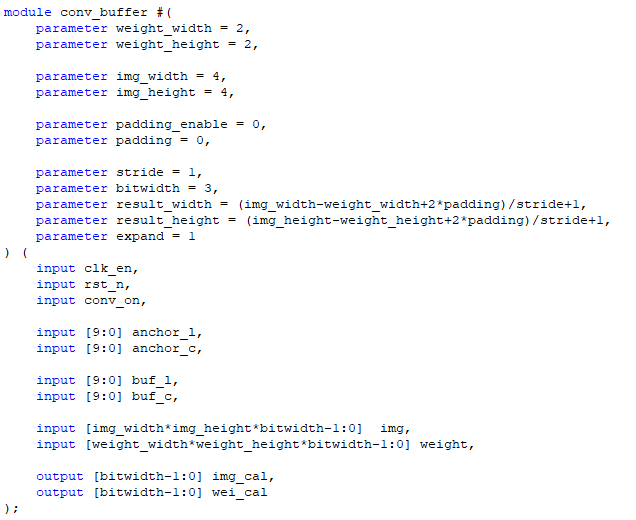


图5.1.1.5

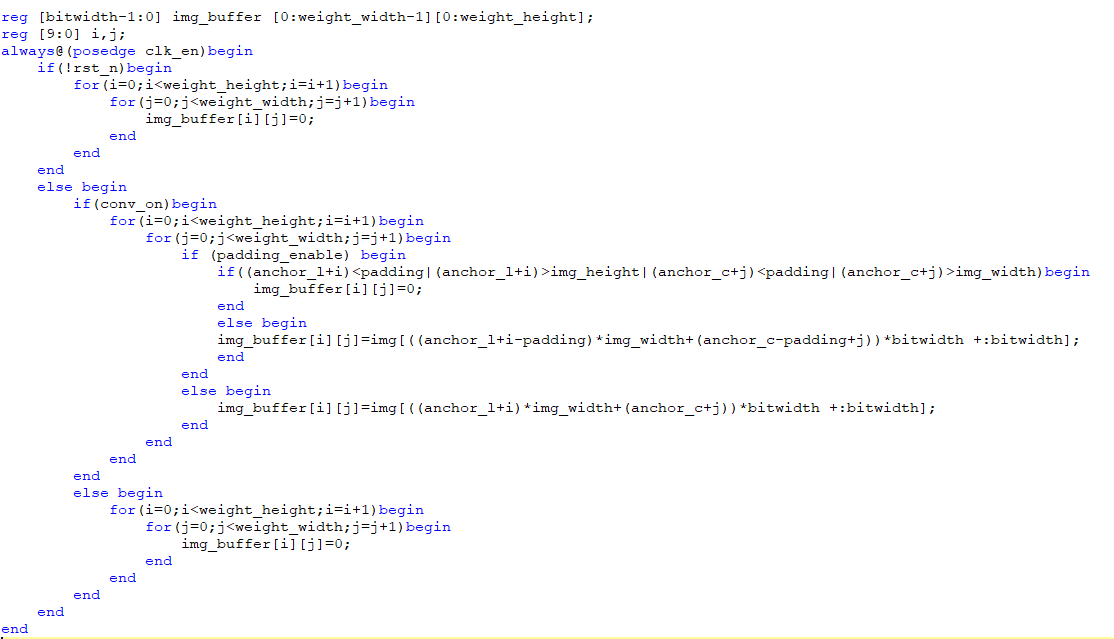


图5.1.1.6



图5.1.1.7

conv\_search模块是完成对所需数据的检索，并输出相应的检索坐标，其输入输出端口如图5.1.1.8所示，此模块的输出信号基本都是conv\_buffer模块的输入信号，用于完成所需数据的定位、检索与提取。

本模块主要搭建了一系列的加法时序电路来更新动态缓冲区输入数据的起始坐标更新，完成对缓冲区中存有的待卷积数据以及权重数据的遍历，更新结果缓冲区的储存坐标。为了让各数据的更新有序进行并且提高运行效率，在此模块中设计了三级流水时序，此流水时序将在一个缓冲区遍历完成前三个时钟周期触发，流水顺序分别是定位坐标的更改-缓冲区的更新以及遍历坐标的更新-结果缓冲区储存坐标的更新。如图5.1.1.9展示缓冲区数据的遍历代码，其它坐标的更新大同小异。

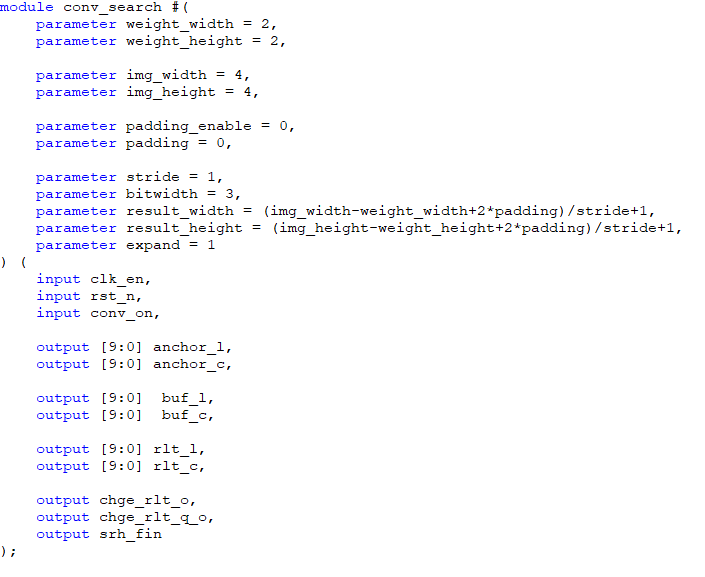


图5.1.1.8



图5.1.1.9

conv\_cal模块的功能是对从缓冲区提取出的待卷积数据以及权重数据进行卷积运算，并且加上偏置，将最终结果输入到结果缓冲区中，完成结果的串转并行输出。其输入与输出端口如图5.1.1.10所示，result即是最终结果的输出端口。

卷积的计算使用了IP核Multiply-Accumulator，提高了卷积运算在PGL22G芯片上的适应性。当一组卷积完成之后，Multiply-Accumulator中累加的值将会输出到结果缓冲区中。结果缓冲区数据会在卷积结束之后将数据转为并行数据，在添加偏置后通过result输出口输出。IP核的例化如图5.1.1.11,偏置的添加与输出如图5.1.1.12，其中rdata即是结果缓冲区。

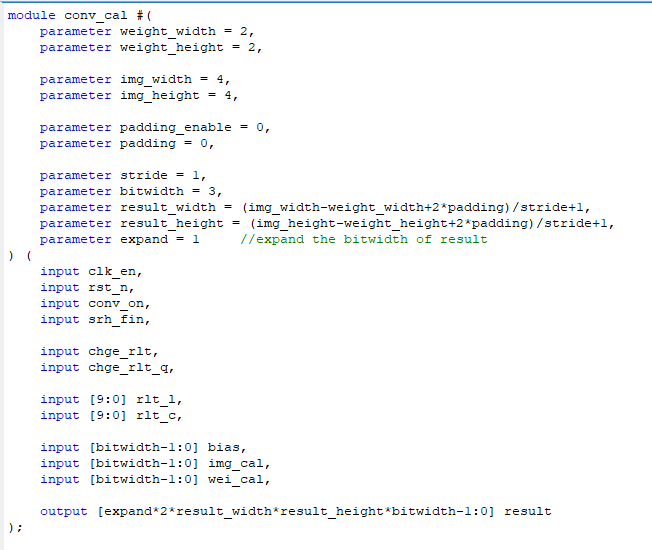


图5.1.1.10

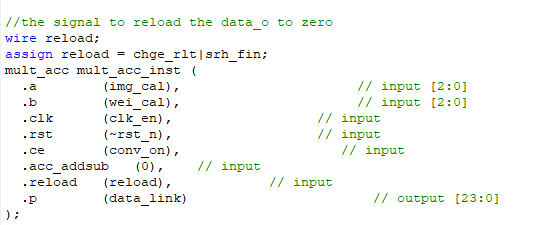


图5.1.1.11

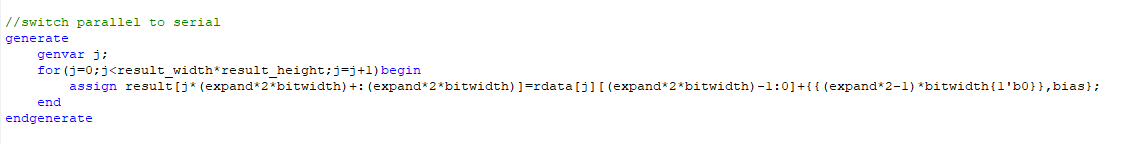


图5.1.1.12

卷积层仿真结果如图5.1.1.13所示，设定步长为1，无padding,其中:img待卷积数据矩阵为:

weight权重数据矩阵为:

预计结果矩阵为：

经对比仿真数据无误：

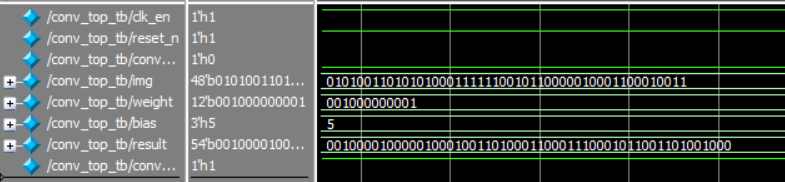


图5.1.1.13

### 线性层

本层的作用是完成神经网络中的线性运算。

线性层只由一个linear模块构成，输入输出端口如图5.1.2.1，in\_features是输入数据,bias为偏置数据,outfeatures为输出数据，flag是线性层接收上一级工作完成后发来的工作使能信号，en\_out是线性层在结束工作时给输出给下一级的工作使能信号。其中例化了一个IP核Simple Multiplier作为乘法计算核心。为了提高运行效率，本模块中设计了两层流水电路，其流水顺序为输入数据和权重相乘-中间结果累加得到最终结果。如图5.1.2.2展示了累加电路设计。

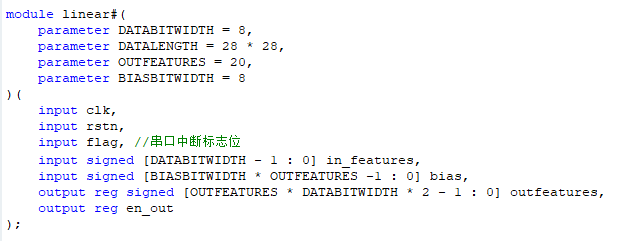


图5.1.2.1

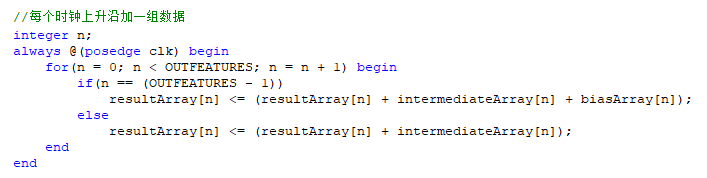


图5.1.2.2

### 上采样层

本层的作用主要是完成数据的上采样中的最近邻功能完成对数据的处理。

上采样层由u\_sampling\_top与u\_sampling两个模块组成，u\_sampling\_top作为u\_sampling的顶层模块完成各种信号的连接，u\_sampling则进行上采样处理。

u\_sampling\_top模块的代码如图5.1.3.1所示，主要功能是例化u\_sampling模块与完成信号的连接。

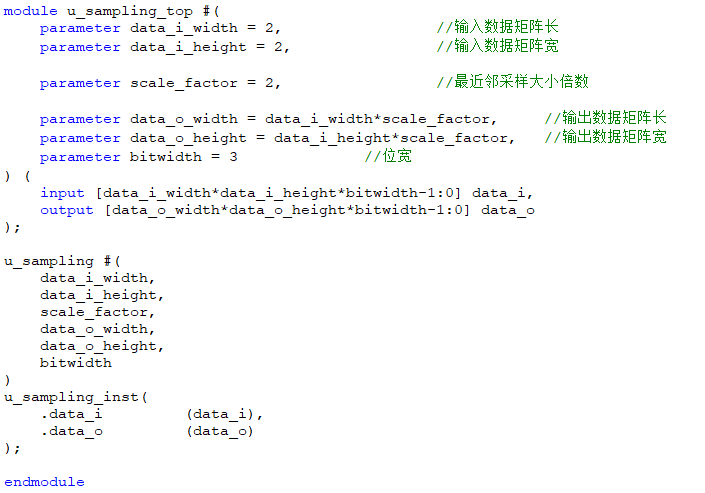


图5.1.3.1

u\_sampling模块功能是完成上采样的最近邻处理，代码如图5.1.3.2，实现方法是搭建逻辑接口连线电路，把相对应的输入输出口连接即可。



图5.1.3.2

仿真结果如图5.1.3.3，经观察可发现已成功完成上采样操作。



图5.1.3.3

### 最大池化层

本层的作用是对数据进行最大池化处理。

最大池化层由maxpool\_top、maxpool\_pick、maxpool\_compare、maxpool\_poolturn四个模块组成。maxpool\_top的功能是进行各子模块之间信号的调度。maxpool\_pick的功能是开辟缓冲区暂时储存待处理信号，并且根据遍历坐标输出对应的数据。maxpool\_compare的功能是将从缓冲区提取出来的信号和临时值进行比较，确定一组数据中的最大值。

maxpool\_top的功能是完成各子模块信号之间的调度，输入输出端口如图5.1.4.1，data\_i是处理数据输入,data\_o是处理完成的数据输出, work\_en是最大池化层接收上一级工作完成后发来的工作使能信号，work\_fin是最大池化层在结束工作时给输出给下一级的工作使能信号。

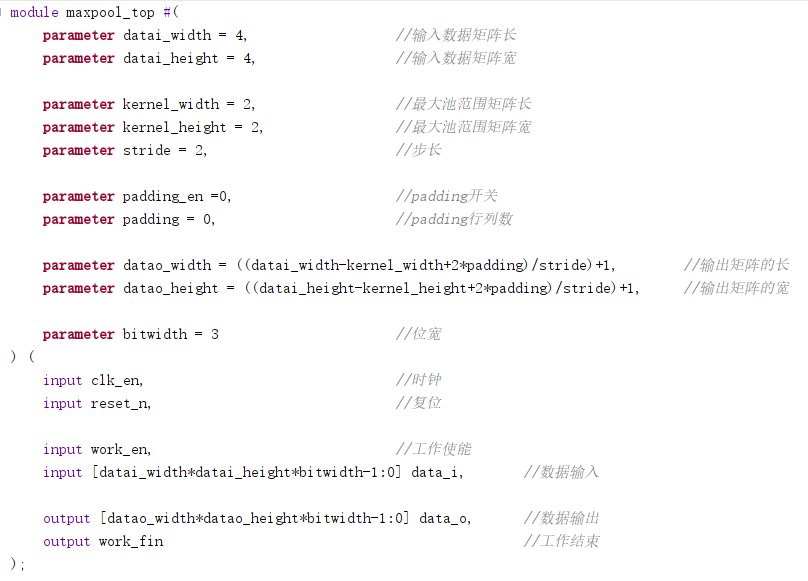


图5.1.4.1

maxpool\_poolturn模块的功能是对maxpool\_pick中的数据缓冲区进行遍历，输入输出端口如图5.1.4.2，data\_l和data\_c是遍历坐标输入端口。

在本模块中如图5.1.4.3开辟了一块逻辑电路用于存放输入数据，另外设计了如图5.1.4.4的数据选择器用于挑选数据输出。



图5.1.4.2

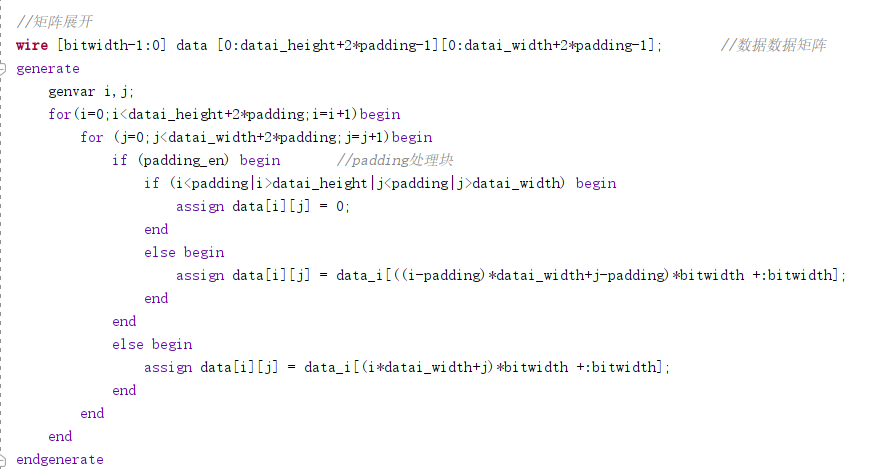


图5.1.4.3

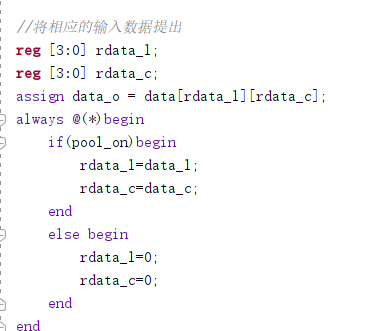


图5.1.4.4

maxpool\_poolturn模块的功能是对输入数据缓冲区与输出数据缓冲区两个缓冲区的存储位置进行遍历，输入与输出端口如图5.1.4.5所示，其中resu\_l和resu\_c是输出数据缓冲区遍历的一组坐标。

本模块设计了时序加法电路，为顺利完成遍历，代码中设计了两层流水，其在一组遍历完成前两拍触发，流水顺序为最大池处理区域更新-最大池数据遍历坐标回归左上角与输出数据缓冲区存储坐标更新。如图5.1.4.6展示输入数据缓冲区坐标遍历部分代码，其它遍历代码大同小异。

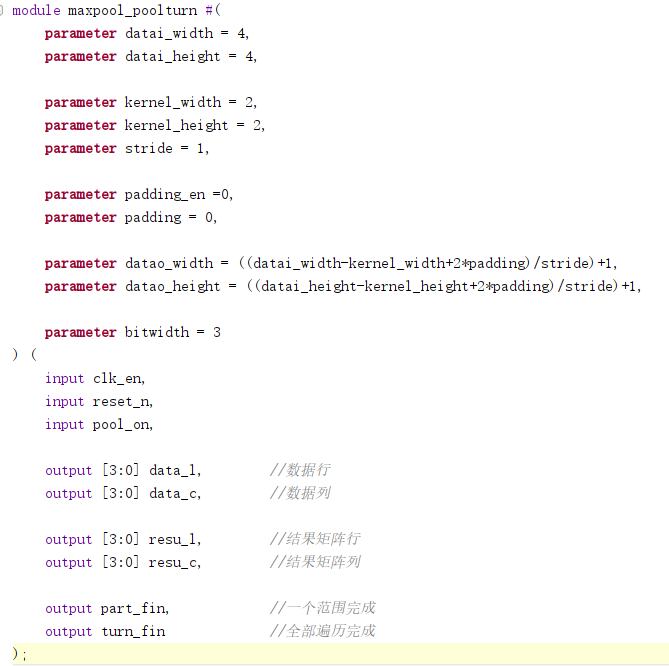


图5.1.4.5

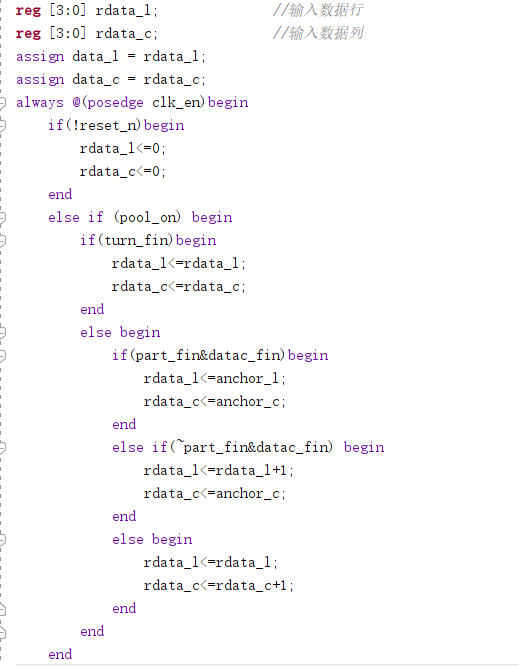


图5.1.4.6

maxpool\_compare的功能是完成数据的比较，并找出最大池区域内的最大值存到输出数据缓冲区内，最后将缓冲区内的数据转为并行数据进行输出。其输入输出端口如图5.1.4.7所示。

此模块搭建了比较器电路进行数据的比较，采用的方法为使用前一个比较出来的最大值和接下来的值作比较，若前值大，则保留；若后值大，则替换前值。在设计中，比较出来的值间接值将直接存入输出数据缓冲区，替换也直接和输出数据缓冲区进行替换，以提高运行效率。如图5.1.4.8展示了比较电路设计的部分代码。

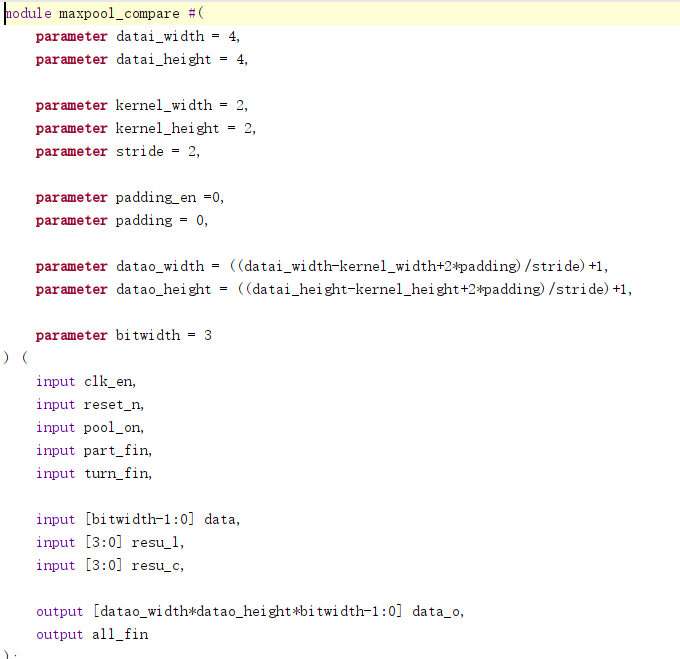


图5.1.4.7

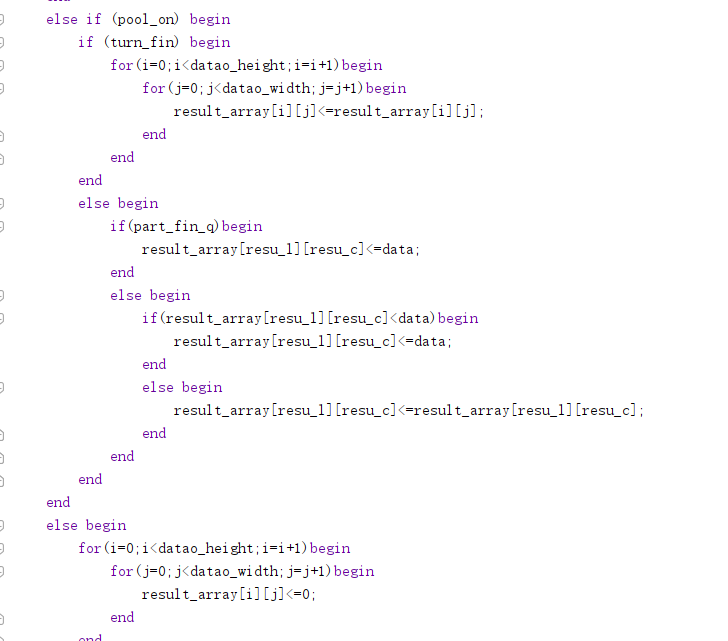


图5.1.4.8

仿真结果如图5.1.4.9所示，设定步长为2，padding为1，其中：

输入数据矩阵为：

输出数据矩阵应为：

经比较仿真结果无误。

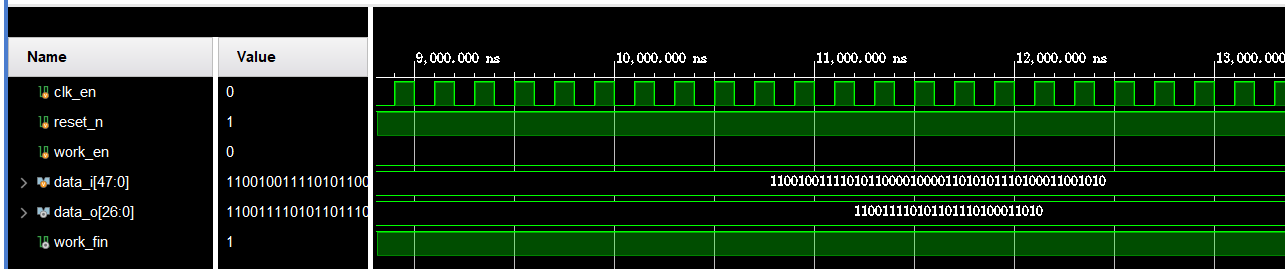


图5.1.4.9

## 颜色识别模块

### KNN结构

KNN结构由五个模块组成，其分别是knn\_top、knn\_img、knn\_dic、knn\_distance、knn\_result。knn\_top模块作为顶层模块，负责调度各子模块的信号。knn\_img模块对缓冲区图像数据进行遍历。knn\_dic模块对缓冲区训练集数据进行遍历。

knn\_distance模块是欧氏距离计算模块。knn\_result模块完成距离比较并输出相应的颜色识别结果。

knn\_top作为颜色识别模块的顶层模块，其中例化了knn\_img、knn\_dic、knn\_distance、knn\_result四个子模块，其输入输出端口如图5.2.1.1所示。

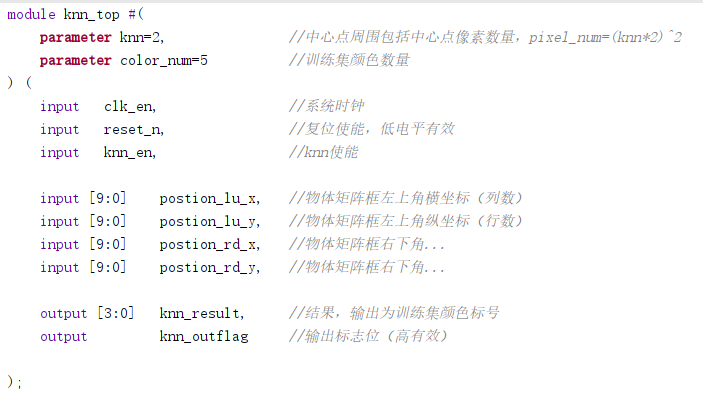


图5.2.1.1

knn\_img模块为图像缓冲区数据遍历模块，其输入输出端口如图5.2.1.2，此模块首先选定knn划定区域的左上角点，然后依次遍历所有knn划定区域内的像素点，输出为遍历坐标。在电路设计中，采用时序递增电路完成此功能。部分遍历代码如图5.2.1.3展示。

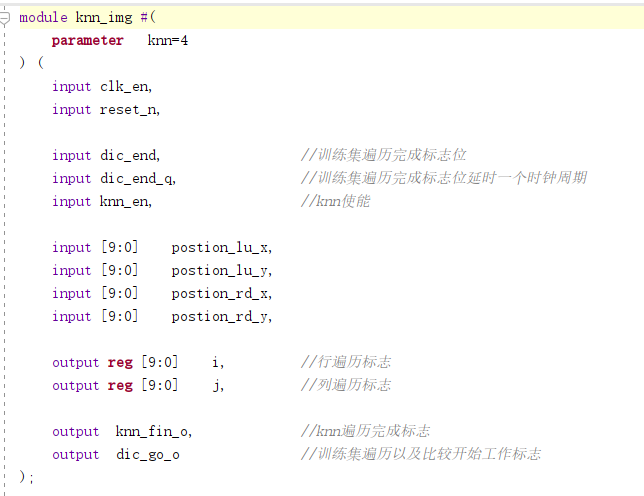


图5.2.1.2



图5.2.1.3

knn\_dic模块的功能是完成缓冲区训练集数据的遍历，输入输出端口如图5.2.1.4。knn\_dic模块与knn\_img的遍历存在一定的流水关系，knn\_img需等待训练集遍历完成才会更换一下个像素坐标，即在这个过程中此像素数据会和每一个训练集数据完成欧氏距离计算。在电路设计中依然采用时序递增电路，部分代码如图5.2.1.5所示。

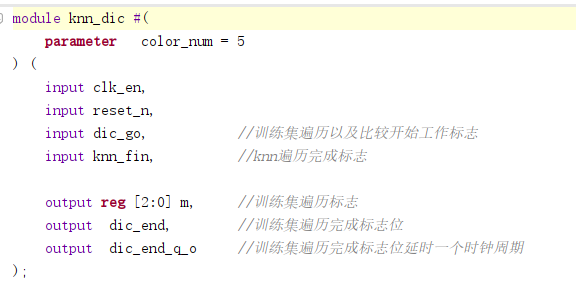


图5.2.1.4



图5.2.1.5

knn\_distance的功能是计算欧式距离，代码如图5.2.1.6，电路设计上采用逻辑运算电路。

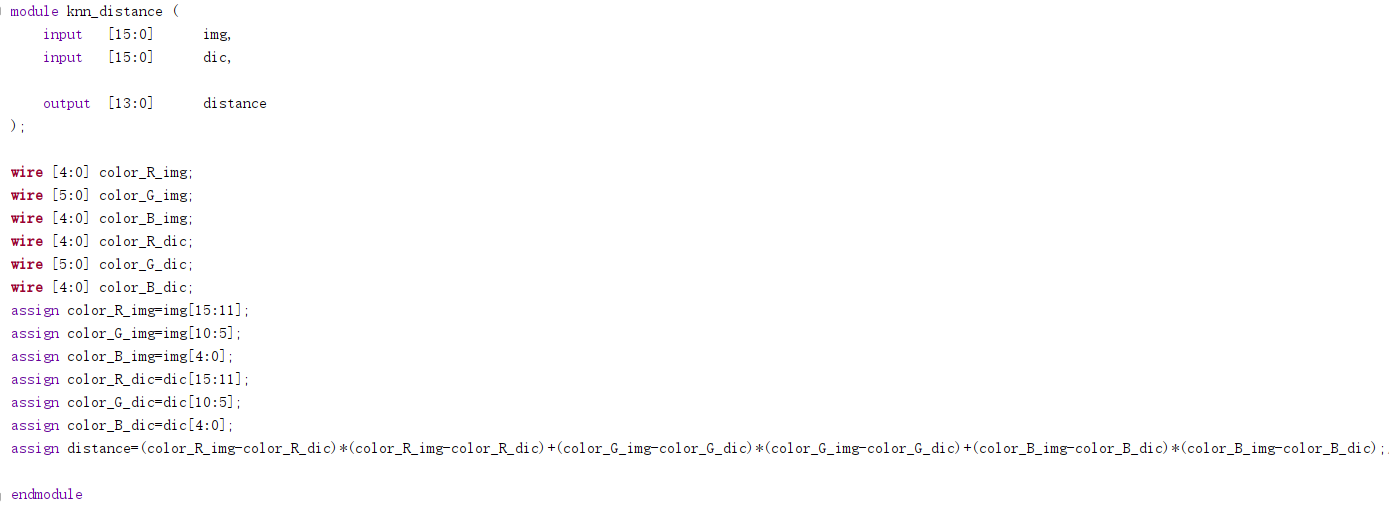


图5.2.1.6

knn\_result的功能是完成各像素点与各训练集间距离的比较，筛选出距离最小的训练集，并做直方图统计，最后根据直方图峰值确定该knn块所对应的训练集颜色种类，输入输出端口如图5.2.1.7。距离最小值筛选部分代码如图5.2.1.8，搭建比较电路完成此项功能。而直方图统计代码如图5.2.1.9，在设计上采用选择电路。直方图峰值筛找电路同距离最小值筛选电路搭建比较电路。

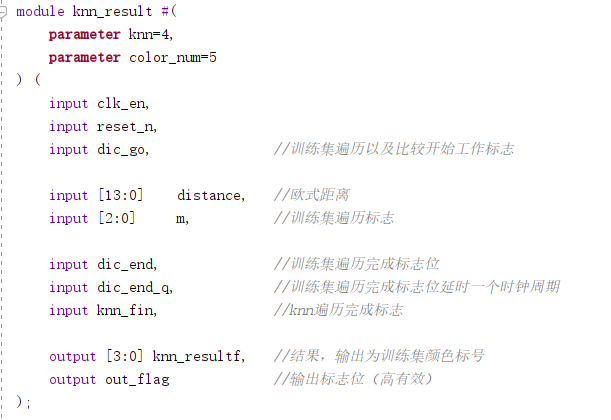


图5.2.1.7

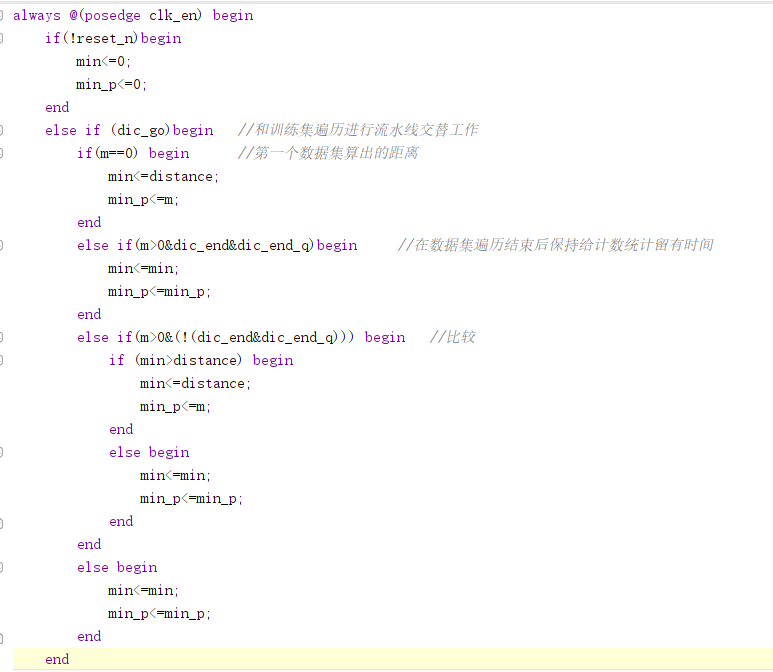


图5.2.1.8

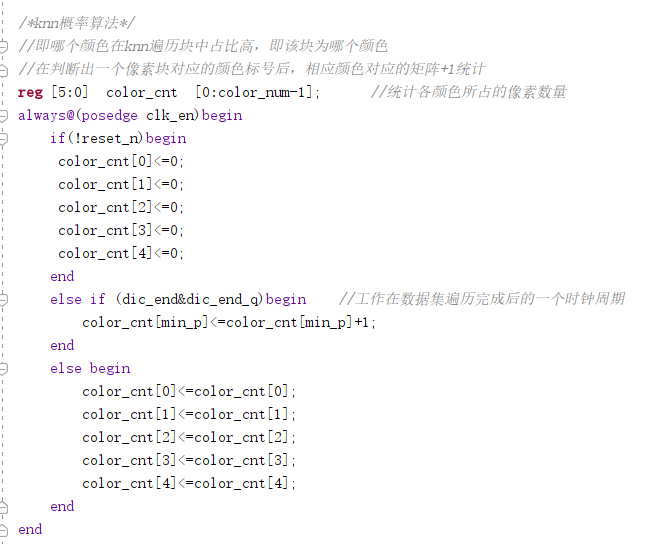


图5.2.1.9

仿真结果如图5.2.1.10，其中：

图像RGB模拟矩阵为：

训练集数据为：

0：7800h（深红色）

1:：FFE0h（黄色）

2：001Fh（蓝色）

3：0000h（黑色）

4：FFFFh（白色）

观察仿真结果，knn\_result输出为1，即黄色训练集代表的信号，颜色识别正确。

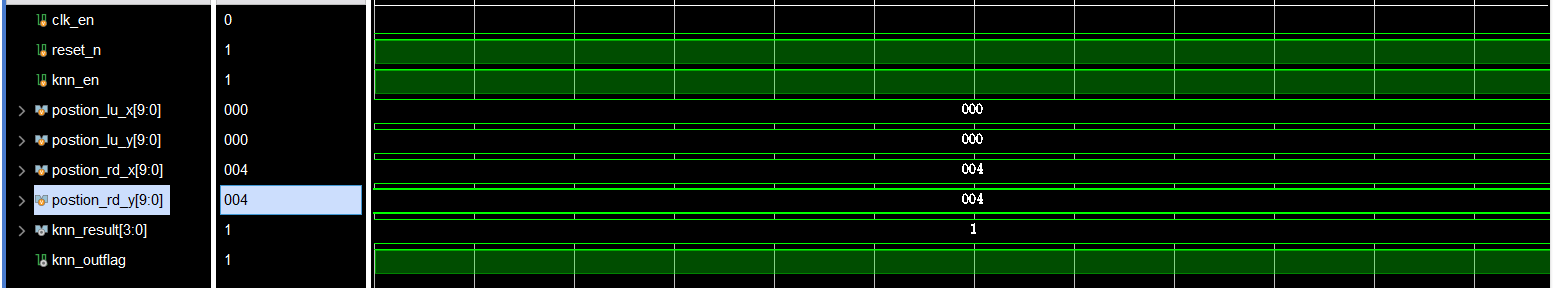


图5.2.1.10

## 前处理与后处理模块

CNN和KNN加速电路与前处理模块独立，前处理模块用于进行图片缩放、色彩空间变换等工作。后处理模块主要用于水果种类、数量、颜色识别码的生成和VGA时序匹配。

CNN电路的前处理模块使用均匀下采样算法实现将1024\*768像素的RGB565图像转换为416\*416的RGB888图像，并行输出三个通道数据给后续的CNN电路处理。KNN电路的前处理模块将输入图像从RGB颜色空间转换为HSV颜色空间，进行直方图统计后输出每个通道对应的像素数；KNN模块接收对应像素值，进行运算后输出可能的颜色值。

CNN电路的后处理模块根据CNN的输出坐标对预测框进行标记，对应像素点被填充为红色，输出识别后的图像；，模块还会根据置信度从高到低排序后根据最高置信度对水果种类识别码编码。KNN电路的后处理模块根据KNN的输出颜色置信度从高到低排序后输出水果暗色识别码。

# 设计参数

使用到的板上资源：

按钮x4、LEDx4、HDMI接口、UART接口、DVP摄像头接口、SPI FLASH

片上资源空载使用量：

LUTs：15.63%

Registers：5.62%

DRM18K：8.33%

片上资源带权重使用量：

LUTs：84.11%

Registers：75.38%

DRM18K：30.12%

功耗：约5W

权重大小：41.3MB

空载速率：约30帧

识别速率：约10帧