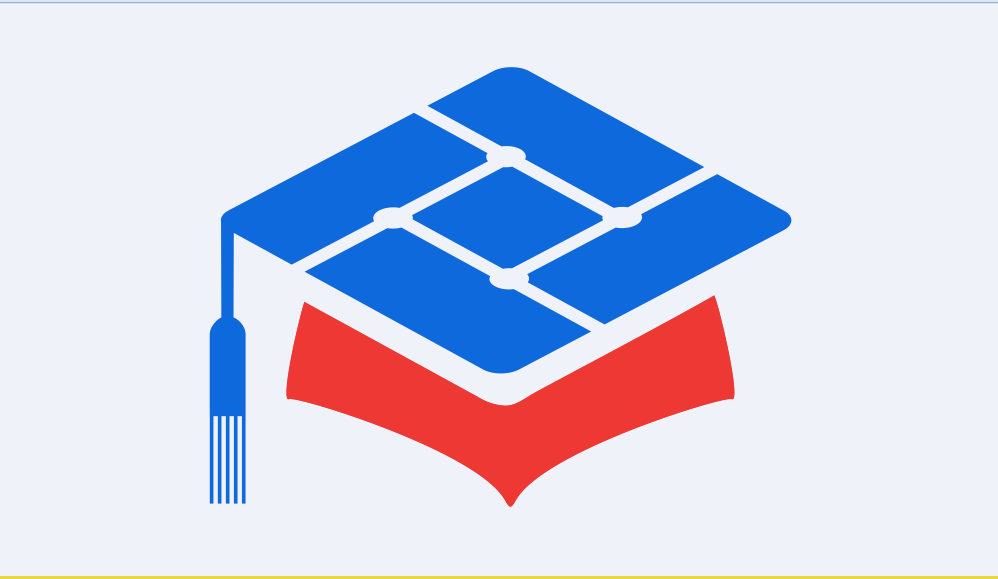
**全国大学生集成电路创新创业大赛**

**CICIEC**

****

**Arm 片上系统设计挑战赛设计报告**

参赛题目： Arm 片上系统设计挑战赛

队伍号： MSC83479

团队名称： 猎鹰队

目录

[1. 系统方案的设计和介绍 1](#_Toc14510744)

[1.1 系统方案的选择 1](#_Toc14510745)

[1.2 软硬件介绍 1](#_Toc14510746)

[2. 神经网络的原理及实现 3](#_Toc14510747)

[2.1. 神经网络的数学模型 3](#_Toc14510748)

[2.2. 反向传播算法 4](#_Toc14510749)

[2.3. 卷积神经网络 5](#_Toc14510750)

[2.4. 数据量化方法 6](#_Toc14510751)

[2.5. 算法验证效果 7](#_Toc14510752)

[3. 硬件介绍 10](#_Toc14510753)

[3.1. SoC总体架构介绍 10](#_Toc14510754)

[3.2. Memory Map简介 11](#_Toc14510755)

[3.3. Bus Matrix 12](#_Toc14510756)

[3.4. OLED 13](#_Toc14510757)

[3.5. UART 13](#_Toc14510758)

[3.6 摄像头驱动模块 14](#_Toc14510759)

[3.7. 神经网络加速器 15](#_Toc14510760)

[3.7.1 卷积层加速器 16](#_Toc14510761)

[3.7.2 全连接层加速器 17](#_Toc14510762)

[4. 软件介绍 19](#_Toc14510763)

[4.1.图像定位与切割 19](#_Toc14510764)

[4.1.1 数字定位 19](#_Toc14510765)

[4.1.2. 数字切割 20](#_Toc14510766)

[4.2. 流程控制 22](#_Toc14510767)

[5. 总结 23](#_Toc14510768)

[参考文献 24](#_Toc14510769)

# 系统方案的设计和介绍

## 1.1 系统方案的选择

本次课题的难点有两个：第一，在中低端FPGA上实现车牌识别，中低端的FPGA资源有限；第二，识别车牌的准确率要求高、速度要求快，并且算法的准确率往往跟算法的复杂程度呈正相关。

基于上述两点考虑，本次实验使用的硬件平台为TERASIC CYCLONE V GX STARTER KIT。该开发平台上的FPGA具有77K的可编程逻辑，150块DSP资源，4884K bits的存储资源。基本能够满足本次实验对于硬件的需求。

车牌识别包含车牌定位、数字分割和识别3部分。本次比赛场景简单，定位分割基于色彩即可完成。在数字识别方案的选择上，常用有两种方案——KNN（K类最近邻聚类），CNN（卷积神经网络）。对于KNN算法，系统的准确率跟聚类对比的库的大小呈正相关，但是要达到题目所要求的精度，KNN的库要非常的大，这带来的计算复杂程度的提升和存储资源的消耗是巨大的。使用CNN算法进行识别的好处有CNN算法不仅抗干扰能力强，并且面向硬件的优化方案成熟可靠（具体的优化方案在第二章介绍），识别的准确度能够达到题目的要求。

综合考虑算法复杂度、加速效果和资源消耗问题，本系统车牌图片采集、定位、车牌数字分割以及其他一些调度任务由软件完成。CNN算法采用硬件加速器实现。

由于本次比赛不仅比拼识别的准确率，还比拼识别的速度。本次设计采用全硬件的加速器的形式来实现CNN算法。这种形式的好处就是能够有效减少总线访问内存的时间，达到更快的速度。

## 1.2 软硬件介绍

我们使用了如下的硬件：

* TERASIC CYCLONE V GX STARTER KIT
* OV5640 CAMERA
* 128\*64单色OLED驱动板
* 自制FPGA GPIO扩展PCB板
* Nano DAP Debugger

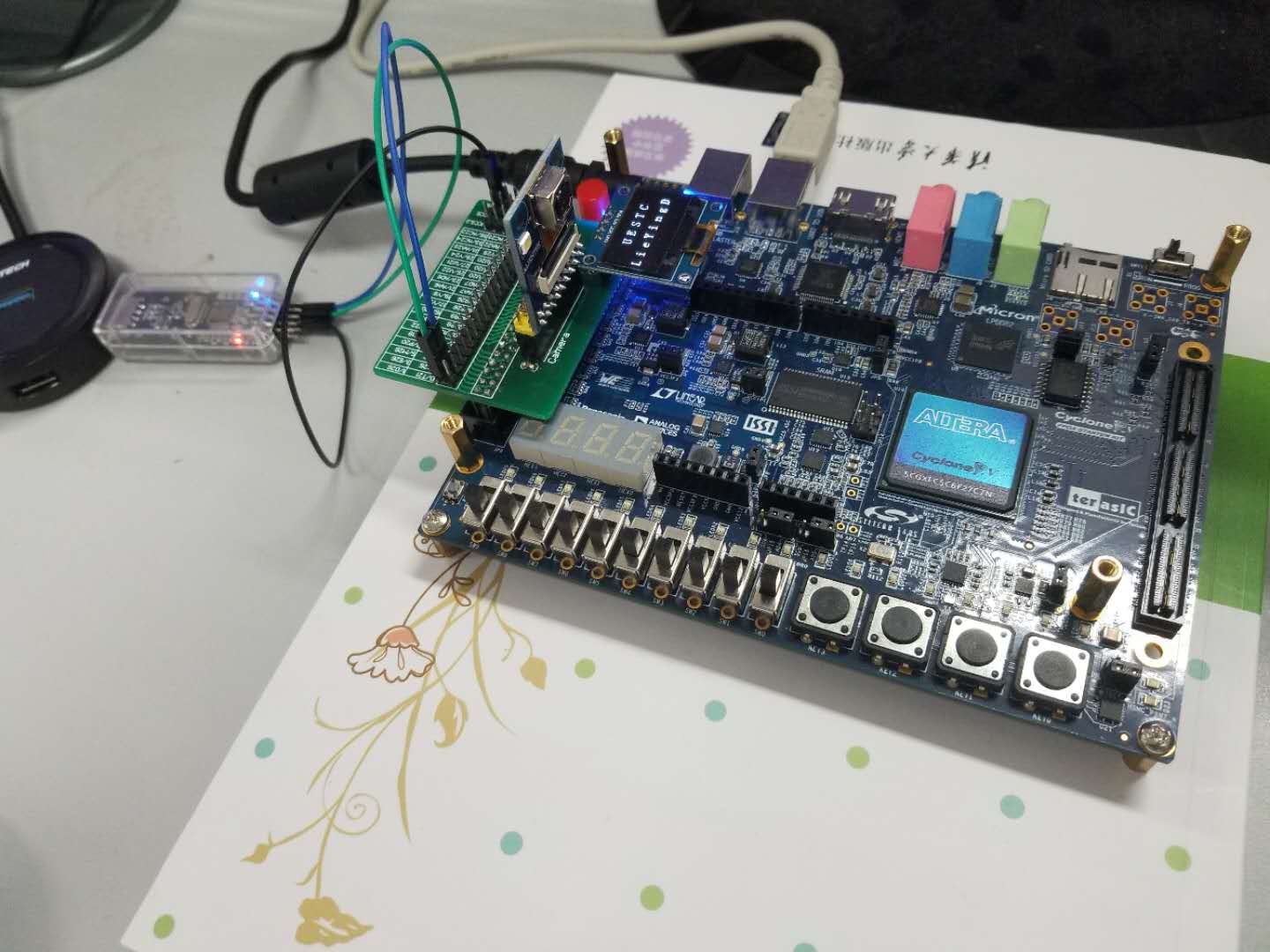
硬件平台如图1-1所示。

图1-1 硬件平台

我们使用了如下的软件：

* QUARTUS PRIME STANDARD EDITION 18.0.0
* KEIL V5.06
* MODELSIM – INTEL FPGA STARTER EDITION 10.6B
* TensorFlow

# 神经网络的原理及实现

## 神经网络的数学模型

构成神经网络的基本单元是神经元，神经元的数学模型如图2-1。

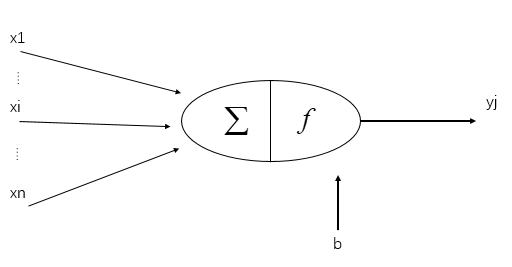


图2-1 神经元模型示意图

公式 (2-1) 描述了神经元的输入、权重和输出之间的数学关系，其中f表示激活函数，激活函数能够给神经网络加入非线性因素，解决线性模型不能解决的问题；而偏置b的引入则使神经网络模型能在任意位置划分平面。

 (2-1)

对于以上给定的神经元模型，权重、偏置和激活函数一定，那么该模型就能表示成一个特定的函数映射。与以上模型不同的是单层感知器的权重可变，这样就可以通过不断的改变权重来进行学习，以获得更优的模型。单层感知器的输入层只提供输入数据，不对数据进行处理，而输出层根据神经元模型对输入数据和权重进行计算并获得输出。公式（2-2）到 (2-5) 描述了单层感知器的学习算法，其中 t 表示真实输出，表示学习率，和表示权重和偏置的变化量。单层感知器从真实输出t和预测输出y的误差中计算权重和偏置的变化，然后更新这两者。

 (2-2)

 (2-3)

 (2-4)

 (2-5)

## 反向传播算法

单层感知器只能够处理线性划分的问题，对于非线性划分的问题可以通过在单层感知器的输入层和输出层之间添加一些隐藏层来解决。隐藏层的节点不存在期望输出，可以通过反向传播算法进行训练。一个单隐层感知器的结构如图2-2所示，其中输入层有d个神经元x1…xd, 隐层有q个神经元b1…bq, 输出层有n个神经元y1….yn; 输入层与隐层之间的权重用v表示，隐层与输出层之间的权重用w表示；隐层与输出层之间的偏置分别用和来表示，激活函数分别用f和g来表示。

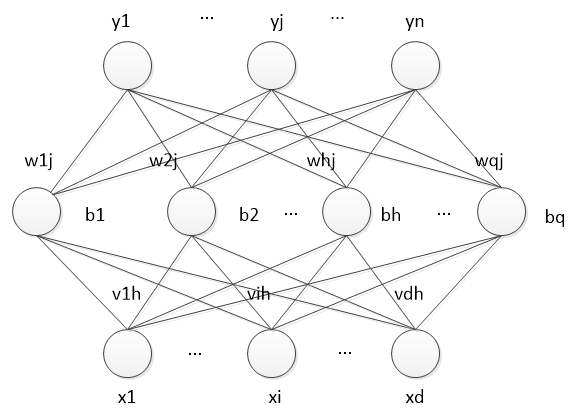


图2-2 单隐层感知器

假设对于输出层的神经元的输入为, 输出为，对于隐层的神经元的输入为,输出为，那么其值分别由公式(2-6)到(2-9)计算可得。

 （2-6）

 （2-7）

 （2-8）

 （2-9）

假设该多层感知器的损失函数为L，那么可以通过反向传播算法和链式梯度法来计算L对各个参数的梯度，并以此梯度和学习率来计算各个参数变化量，从而更新参数，达到学习的目的。以隐藏层和输出层之间的权重为例，其变化量可以由公式（2-10）计算得出，然后在公式(2-11)来更新该参数。

 （2-10）

 （2-11）

## 卷积神经网络

神经网络的层数越多，越能拟合更加复杂的函数。但是网络层数的增加会带来一些问题，比如参数膨胀。由于神经网络两层之间的神经元是全连接的，那么层数增多，连接的参数个数也会跟着膨胀。巨量的参数会使神经网络训练起来特别困难，它会减缓训练的速度，同时也更容易陷入局部最优解。卷积神经网络具有稀疏交互、参数共享和等变表示优良特性，很擅长于图像分类识识。基于此，本次比赛搭建了如图2-3所示的卷积神经网络。该网络总体的架构是2个卷积层，卷积核大小为2\*2，两个池化层，池化窗口为2\*2，两个全连接层，最后一层全连接层保留34个神经元，分别对应车牌数字0到9和24个字母(除去I和O)，softmax层到对应这34个数字和字母的概率分布情况。

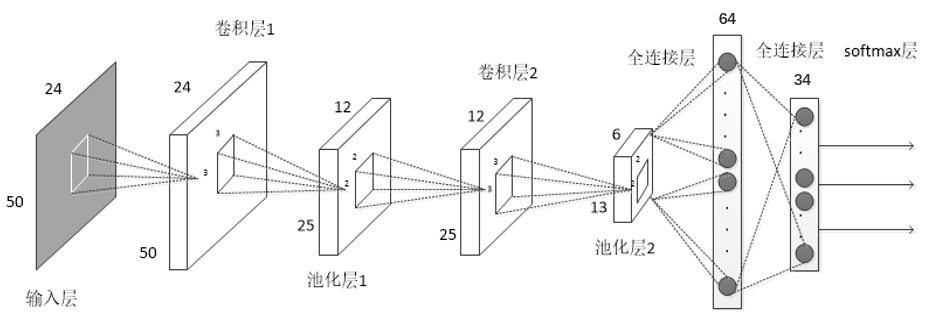


图2-3 卷积神经网络架构

如图2-3所示，与全连接神经网络不同的是，卷积神经网络的权重又叫卷积核，并引入两种特别的层：卷积层和池化层。

卷积层主要利用卷积核对图像进行特征提取，不同卷积核能够提取图像的不同特征。如图2-4所示，假设图像大小是5\*5,卷积核大小是3\*3,为了使卷积之后的特征图大小不变，可以在图像的周围补0，这样卷积之后仍然得到5\*5的特征图。卷积核在补0后的7\*7的图像上进行滑动卷积，步幅假设为1，首先以为中心的9个像素值和卷积核对应项目相乘后把积累加起来得到C0,然后卷积核向右滑动一个像素，继续按同样的方法计算C1。以此类推，直到计算出C24为止，卷积结束。

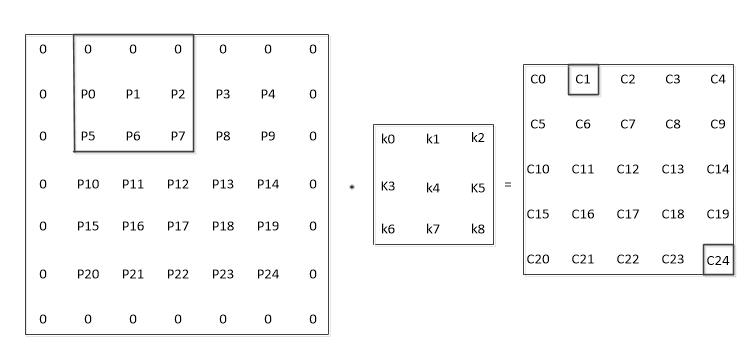


图2-4 图像卷积示意图

池化层的目的主要是把卷积后得到的特征图进行压缩，一方面可以进一步减少参数的数量，另一方面可以提取更加抽象的特征。池化层与卷积层不同的是池化层没有可以训练的参数。一般常用的池化计算方式有两种，最大池化和平均池化。如图2-5所示，最大池化把每个池化窗口中最大的数作为池化的结果。

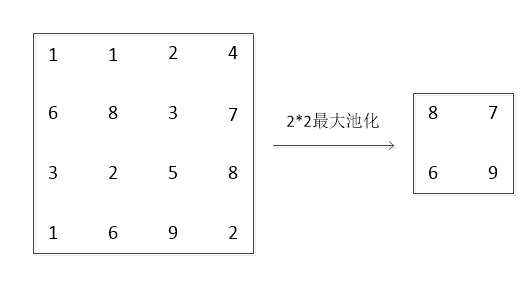


图2-5 池化方式说明

## 数据量化方法

计算机中小数通常有两种表示方法，定点表示法和浮点表示法。由于定点数表示法中，小数点的位置约定好之后不能再改变，因此表示数据的动态范围很小，无法同时表示相差很大的数据。浮点表示法虽然精度高，动态范围大，但是其表示方法复杂，需要专用的浮点运算单元来实现两个浮点数的计算，硬件资源消耗大。而且神经网络的容错能力很强，对单点计算的精度要求不高。而定点数计算消耗的资源相对较小，而且定点数的位数越少，存储和计算需要消耗的资源越少。因此，本文采用8位有符号定点数的方式来量化数据。由于定点数动态范围小，而神经网络中有大量的乘累加操作，因此很容易造成加速器的计算结果超出了该定点数的表示范围的情况，这样会很大程度上影响神经网络模型的准确度。为了解决以上问题，本次比赛我们引入动态机制。对于两个向量操作数的计算，每个向量中的元素的小数点位置是一样的，但是这两个向量的小数点位置可以不一样。计算结束后，将结果裁剪到8位，而这个裁剪过程也是可以动态地配置，这样就可以实现动态地设置计算结果的小数点位置。

在实际的量化操作中，小数点位置可以根据被量化数据的范围来确定。我们用BP表示小数点的位置，规定BP为从小数点到最后一位之间的二进制位数。对于8位有符号定点数来讲，BP的范围是0 ~ 7，如果BP=0则说明定点数为纯整数，如果BP=7则说明定点数为纯小数。比如在量化神经网络某一层的权重时，发现这一层的权重在-0.9到0.9之间，那么小数点在可以确定在有符号为之后，即BP=7;如果发现某一层的权重在-3到4之间，那么可以确定BP=4，即用3位来表示整数部分，4位表示小数部分。

同样，计算结果的量化也要根据其大致动态范围来确定。对于计算结果的裁剪可以通过将结果右移取其低8位的方式来实现。在不对结果裁剪的情况下，如果已知两个操作数的小数点位置和计算类型，那么可以推断出结果的小数点位置为BPy，那么可以由公式(2-12)推断点积的结果小数点位置BPx.y。如果裁剪后要求小数点的位置为BPclip，那么右移的位数Rbits可由式(2-13)得出。

 （2-12）

 （2-13）

## 算法验证效果

在算法验证时，从网上下载标准的车牌数据图片，再将数据集按比例缩放成摄像头拍摄的图片对应的分辨率，处理后的图片作为训练集和验证集。利用tensorflow框架搭建好神经网络后，再将摄像头拍到的车牌定位分割后得到的车牌数字作为测试集进行算法验证，验证结果如图2-6，经过460次迭代，在测试集上识别的准确率达到100%。

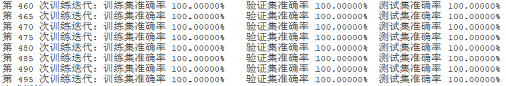


图2-6 算法验证结果

此时，再利用训练得到的网络识别测试集中每一张图片，并将推断过程中每一步中间计算结果以及网络参数的数值按照8位动态定点数量化。经过对比，发现测试0~9这10个数字，中间计算结果小数点位置大致相同，选取小数位数最小的那张图片作为参考对数据进行量化，可以兼容小数位数较多的情况。最终，各个变量的小数点位数如表2-1所示。

表2-1 卷积神经网络各变量量化小数点位数

|  |  |  |  |
| --- | --- | --- | --- |
| 变量 | 小数点位数 | 变量 | 小数点位数 |
| x | 6 | h\_relu2 | 2 |
| w\_conv1 | 7 | h\_pool2 | 2 |
| b\_conv1 | 7 | w\_fc1 | 7 |
| h\_conv1 | 4 | b\_fc1 | 7 |
| h\_sum1 | 4 | h\_mat1 | 1 |
| h\_relu1 | 4 | h\_fc1\_sum | 1 |
| h\_pool1 | 4 | h\_fc1\_relu | 1 |
| w\_conv2 | 7 | w\_fc2 | 7 |
| b\_conv2 | 7 | b\_fc2 | 7 |
| h\_conv2 | 2 | h\_mat2 | 0 |
| h\_sum2 | 2 | y | 0 |

其中，x表示神经网络的输入，w\_conv1和b\_conv1分别表示卷积层1的权重和偏置; h\_conv1、h\_sum1和h\_relu1分别表示卷积层1的卷积结果，加上偏置结果和激活之后结果; h\_pool1表示池化层1最大池化后的结果。同样，w\_conv2和b\_conv2分别表示卷积层2的权重和偏置; h\_conv2、h\_sum2和h\_relu2分别表示卷积层2的卷积结果、加上偏置的结果和激活之后的结果; h\_pool2表示池化层2最大池化后的结果。w\_fc1和b\_fc1表示全连接层1的权重和偏置；h\_mat1、h\_fc1\_sum和h\_fc1\_relu分别表示全连接层1乘累加之后的结果、加上偏置的结果和激活后的结果。w\_fc2和b\_fc2分别表示全连接层2的权重和偏置；h\_mat2表示全连接层2乘累加之后的结果，而y则是加上偏置之后的结果，也是该卷积神经网络最终的计算结果。

固定网络参数及数据的小数点位数后，用量化后的神经网络识别测试集，测得测试集的准确率依然是100%。

# 硬件介绍

## SoC总体架构介绍

我们设计的SoC由如下几个主要模块构成，其中除去CM3核源代码外，其余部分均是自己编写实现。

* Cortex-M3 Core
* Bus Matrix
* Single Channel DMA
* ITCM(Store new program code from KEIL for debugging)
* ROM(Store the initial program code)
* DTCM(Stack and Heap)
* UART
* GPIO(LED and Switch)
* OLED
* CAMERA
* ACC
* Timer

所有外设通过AHB3lite总线接入Bus Matrix实现与核的通信，SoC总体结构图如图3-1所示。

在最初设计Bus Matrix时，预留了两个加速器接口，分别为Master和Slave接口，现在加速器被设计为总线Slave，因此预留了Master接口将不会被使用到。

Cortex-M3总共使用了7个IRQ中断：

* 四个为按键中断，由FPGA开发板上PUSH BUTTON经过消抖与上升沿检测得到；
* 一个为加速器中断，加速器计算完成后将会通过此中断告知核；
* 一个为DMA中断，我们设计的Bus Matrix为静态仲裁，因此需要用WFI指令配合DMA，使核主动交出总线控制权，因此DMA工作完成后需要用中断唤醒核；
* 一个为UART中断，UART接收模块没有缓冲区，因此使用中断。

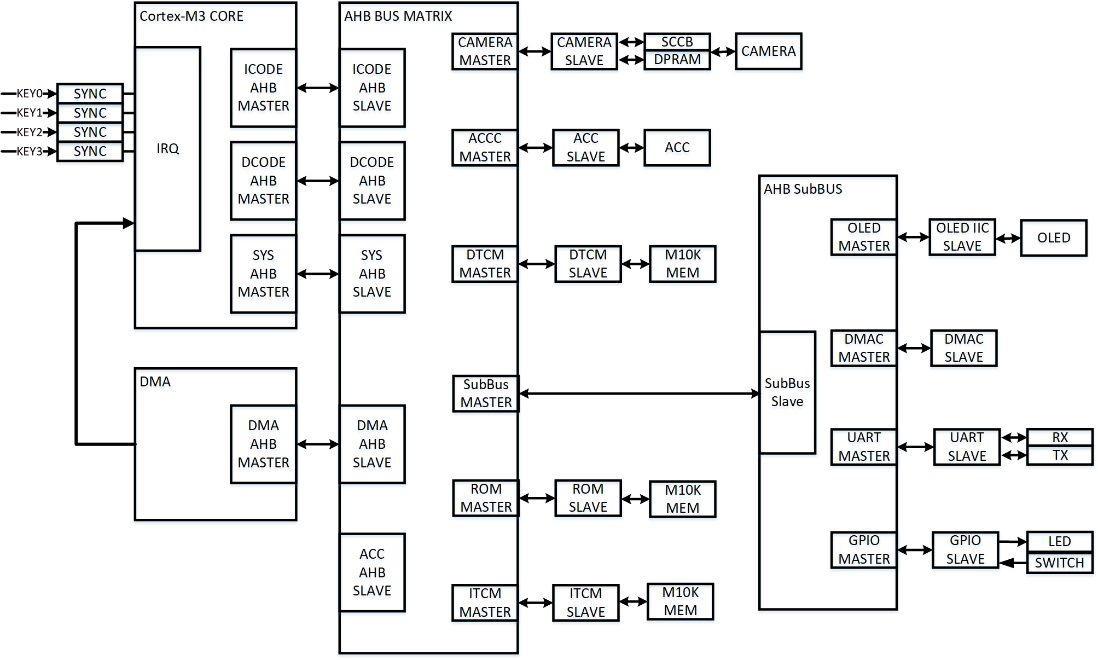


图3-1 SoC总体架构

## Memory Map简介

本设计的地址空间分配如图3-2所示。

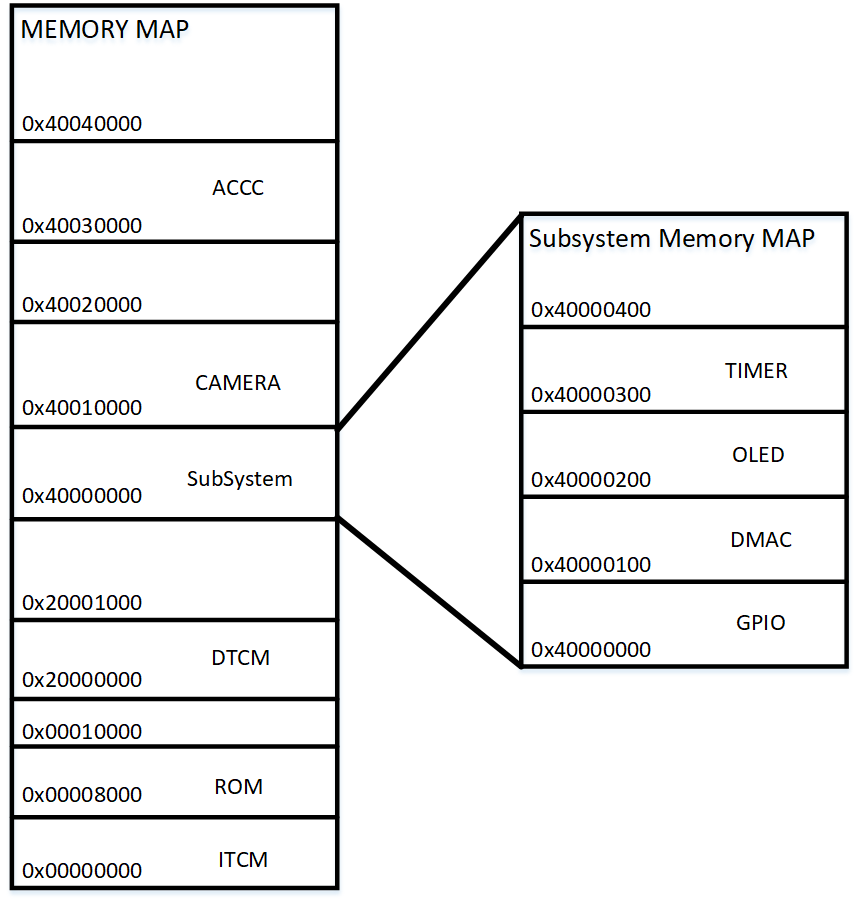


图3-2 Memory Map

## Bus Matrix

Bus Matrix结构如图3-3所示，其总共由三级组成：输入级、译码级、仲裁输出级。

总共有5个Slave(From Master)接口：ICODE、DCODE、SYS、DMA、ACC，其中由于加速器被设计为总线Slave模块，因此ACC Master已经被放弃。

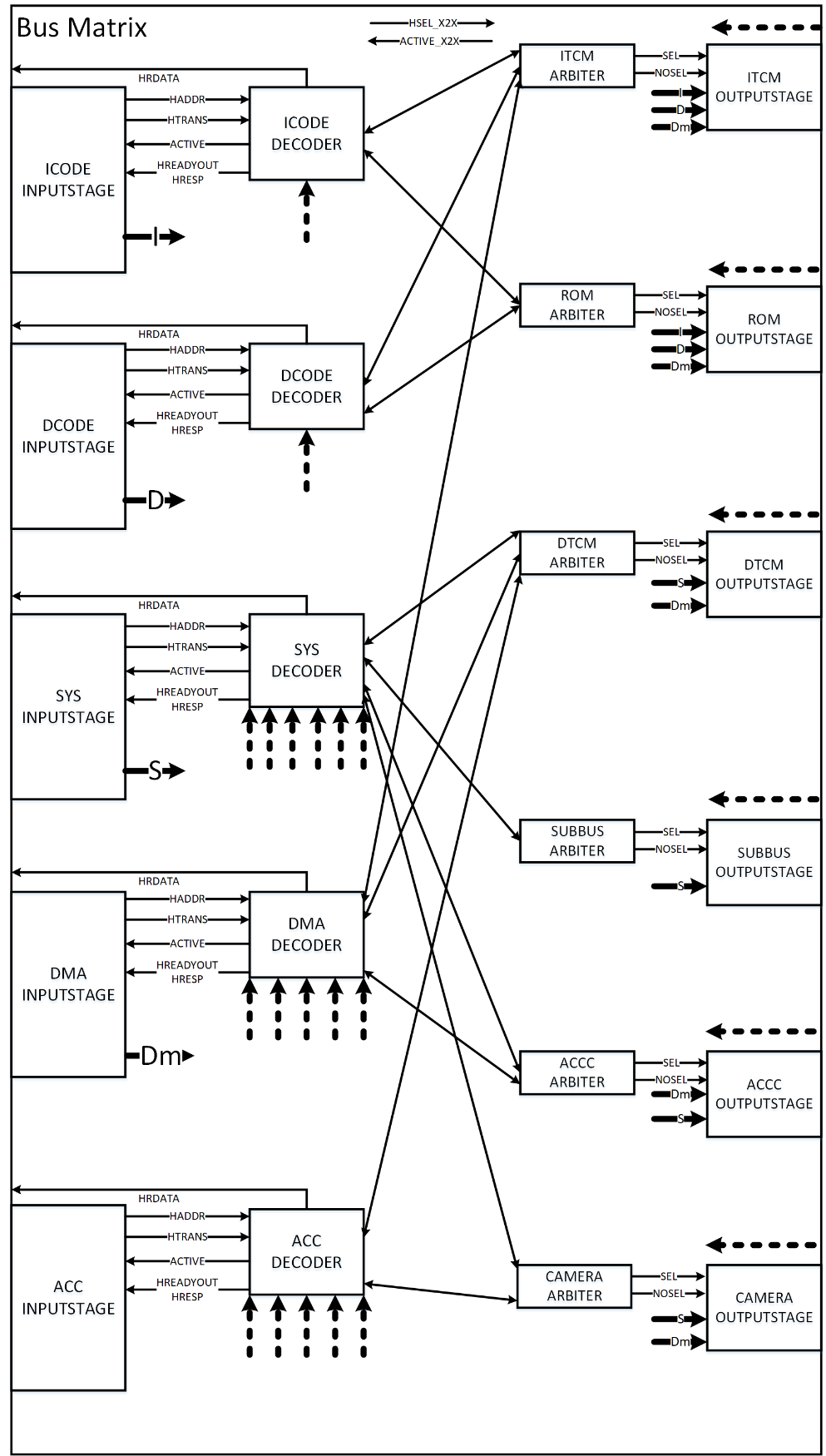
总共有9个Master(To Slave)接口，分别连接每个外设。

图3-3 Bus Matrix结构

## OLED

OLED显示屏为IIC接口，因此直接使用GPIO模拟IIC时序实现OLED配置。核通过写两个寄存器的最低位分别实现IIC总线的SCLK与SDIN控制，如下图3-4所示。

图3-4 OLED控制寄存器

## UART

UART外设主要由三部分组成：

* UART RX：用于接收数据，数据接收完成后向总线输入接收到的数据值并向IRQ中断产生一个时钟周期的脉冲。
* UART TX：用于发送数据，内部包含有一个缓冲器（FIFO）用以缓冲总线传来的数据，并通过总线提供FIFO满状态的状态寄存器，CPU需要根据此寄存器判断是否可写。

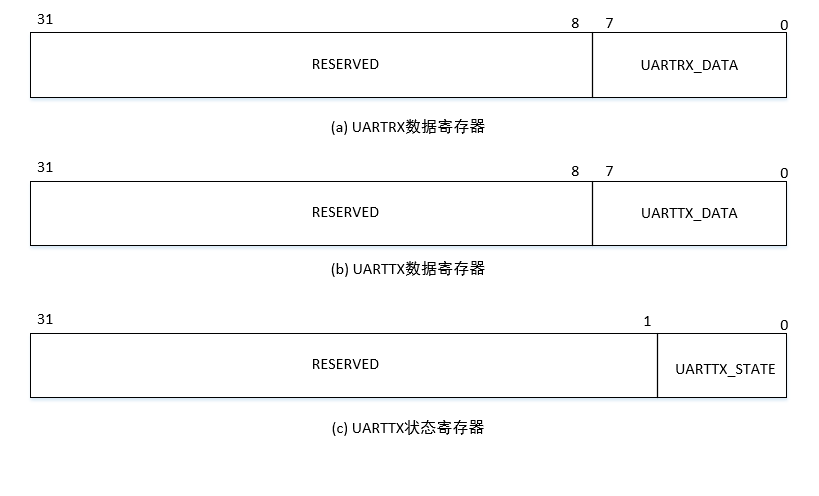
UART对应着总线上三个寄存器，及三个word的地址空间，三个寄存器格式如图3-5所示。

图3-5 UART寄存器格式

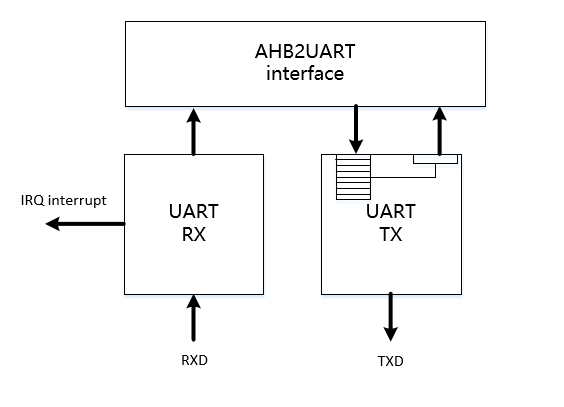
UART结构如图3-6所示。

图3-6 UART模块结构图

## 3.6 摄像头驱动模块

OV5640的驱动接口，主要分为两部分：第一部分为SCCB（Serial Camera Control Bus）总线控制接口，用于配置摄像头的寄存器，使摄像头工作在想要的状态；第二部分为，摄像头数据接口，数据接口传回来摄像头拍摄的图片数据。

由于本次实验的内容为对车牌进行拍照，为了最求更快的运算速度。在保证准确率不降低的情况下，我们将摄像头配置成QVGA输出模式，以减小单张图片的数据量。同时，将数据的格式配置成数字系统中易于处理的RGB565格式。尽管如此，图片的数据仍然会占用大量的片内存储资源。本次实验的一大难点就是FPGA片上资源有限。在图片的存储上花费大量的存储资源是不值得的，因为我们接下来所使用的神经网络结构将会使用大量的权重数据。因此，本次实验中，我们在摄像头数据的前级加入了一个数据预处理单元。

由于在环境情况不复杂的情景下，在RGB565颜色空间中，车牌的蓝底白字两种颜色中R通道和G通道的数值相差往往较大，且存在的干扰少，因此我们可以直接使用比较器对图片的像素点进行二值化处理。这能够大大降低存储图片所需要的存储资源的消耗，使我们避免了使用外部内存，大大提高了系统的运行速度。

如图下图所示，是我们摄像头前级驱动模式的设计结构图，我们的摄像头前级驱动模块主要分为4个部分：SCCB总线驱动，摄像头数据接口，数据预处理单元，摄像头数据存储RAM。系统初始化时，处理器通过SCCB总线配置摄像头。当需要读取数据时，处理器通过总线发送控制信号，数据被预处理单元读取，并将处理好的数据写入到摄像头存储RAM中。



图3-7摄像头驱动模块设计原理图

## 神经网络加速器

在第2节介绍，本设计的神经网络分有2个卷积层（卷积核为33，8通道的卷积层），2个池化层，2个全连接层和1个softmax层。在硬件设计时，简化softmax层，避免设计复杂的指数运算模块，而采用直接比较最后一层神经元的大小为待测图片分类。另一方面，考虑到神经网络两个卷积层规模相同，在设计硬件时可以尽量复用，以节约硬件资源。最后，采用采用8位动态定点数对网络参数和中间计算结果进行量化，不仅解决了FPGA中资源有限，浮点运算速度慢的问题，而且保证了计算的精度。

如图3-8所示的加速器与总结连接关系，加速器的结构包含权重内存W\_RAM和数据内存D\_RAM、一层卷积层加速器、8个通道的RAM作为中间数据的存储、以及一层全连接层加速器。运算开始使，软件将二值化的摄像头图片数据搬到数据内存D\_RAM中，再通过总线给出Ready握手信号，在计算完成过后，加速器输出Valid信号，处理器就通过总线读取0到9这10个数字的可能值。

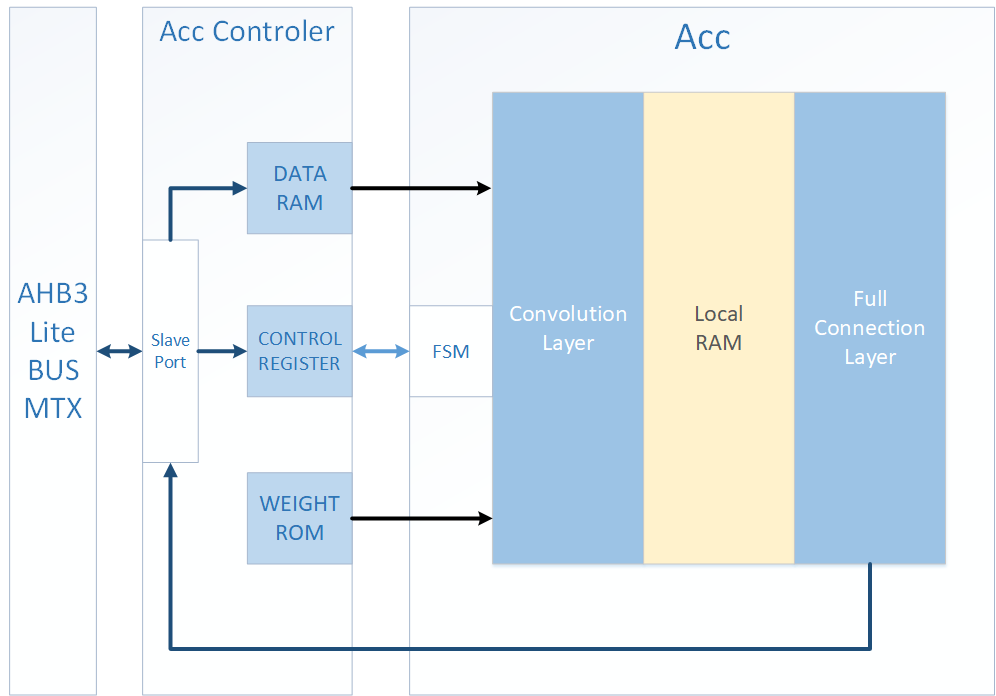
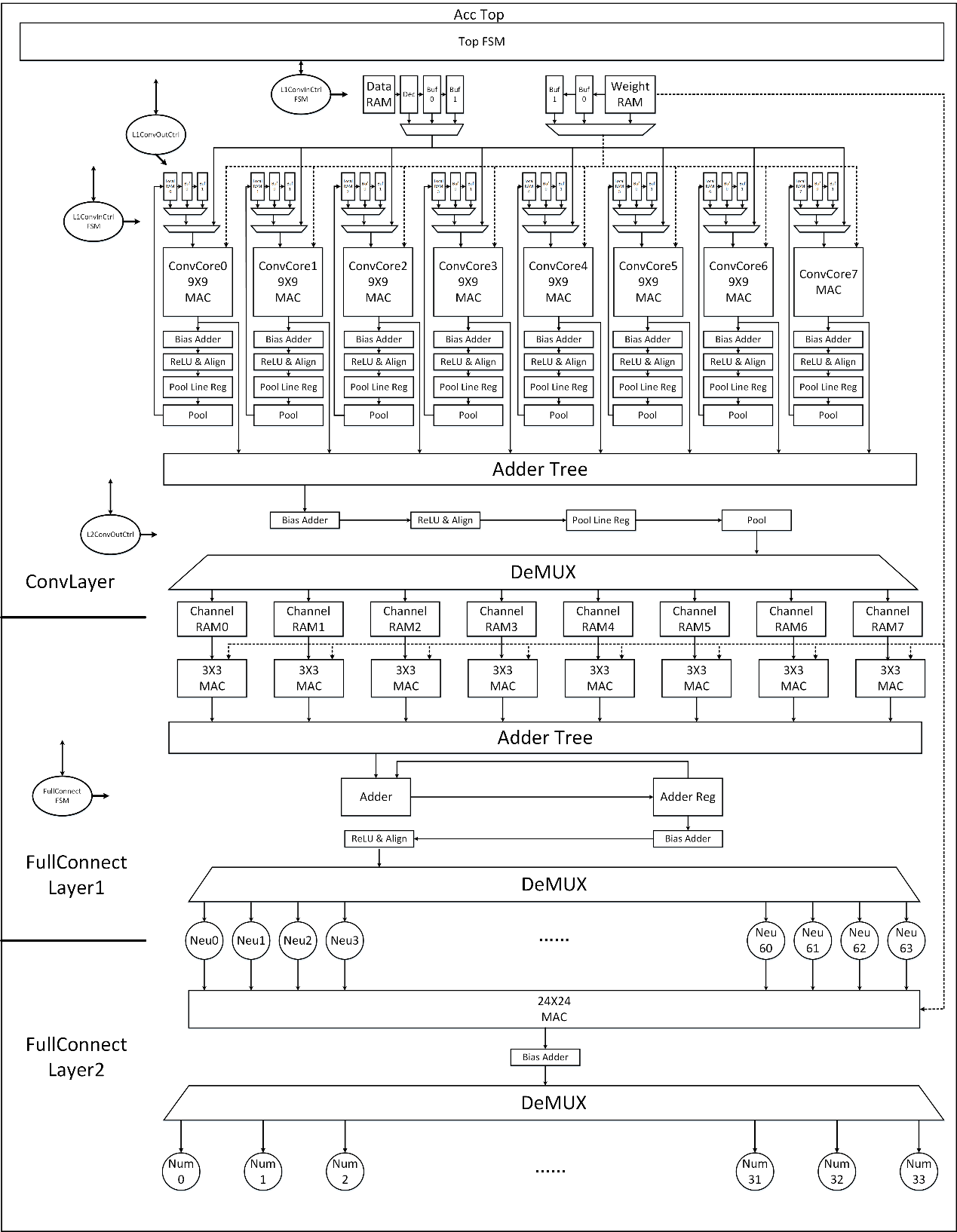
 神经网络加速器顶层结构如下图3-9。

图3-8加速器结构图

图3-9 神经网络加速器顶层结构

### 3.7.1 卷积层加速器

由上面神经网络介绍可以知道，我们本次神经网络加速器包含了一个卷积层加速器和一个全连接层加速器。卷积层加速器完成了两层卷积层的运算。

卷积层加速器包含了卷积数据选择单元、激活函数运算单元、数据对齐运算单元、池化运算单元以及8组组合逻辑运算电路构成。每组运算单元包含9个8位定点乘法器和9个定点加法器，可执行33。

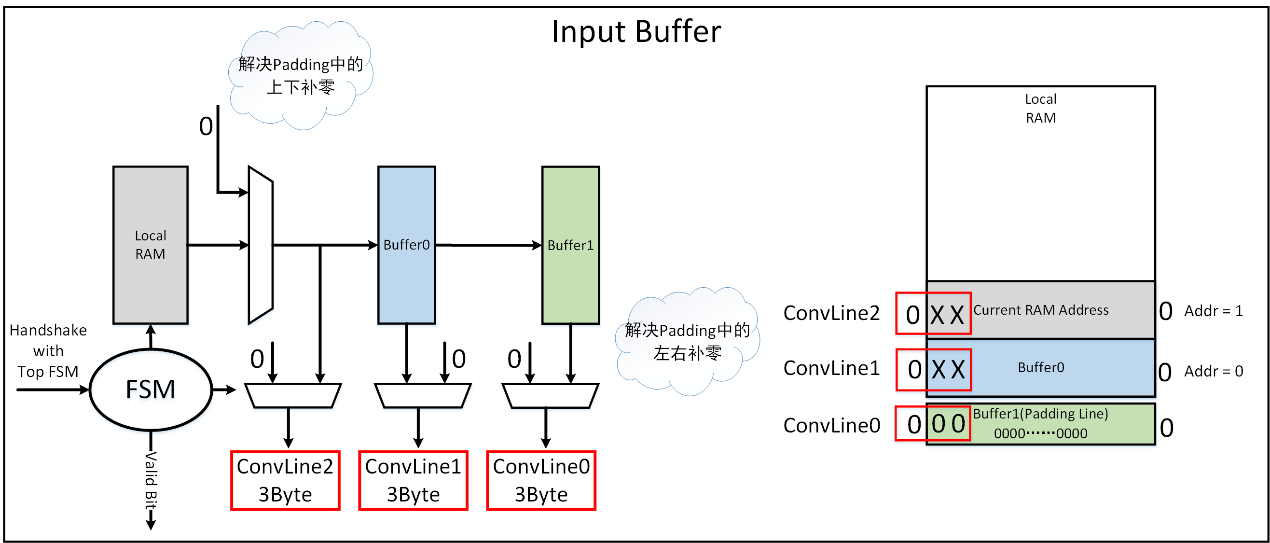
卷积层数据选择器如3-10所示，是由多路复用器构成，每次读取数据图片的三行数据存在寄存器中（第一层卷积层二值化的图片数据一行24个byte，第二层卷积的图片数据为每行12byte），通过多路复用器选择三行中固定位置的数据输入到运算单元中。

图3-10 Input Buffer

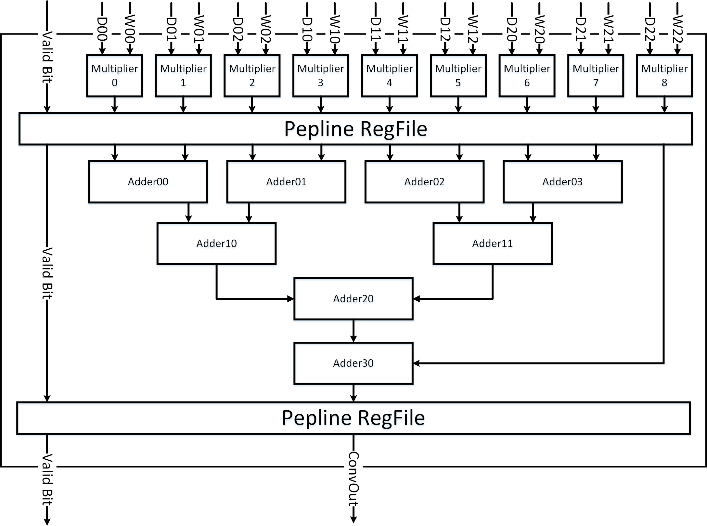
如图3-10所示的卷积核为两级流水线，同时流水线寄存器堆有一个单独的valid bit随着数据一起流动，表示数据的有效性。在后面的池化寄存器控制以及输出控制部分，均有valid bit作为使能信号。

图3-10 两级流水线卷积核

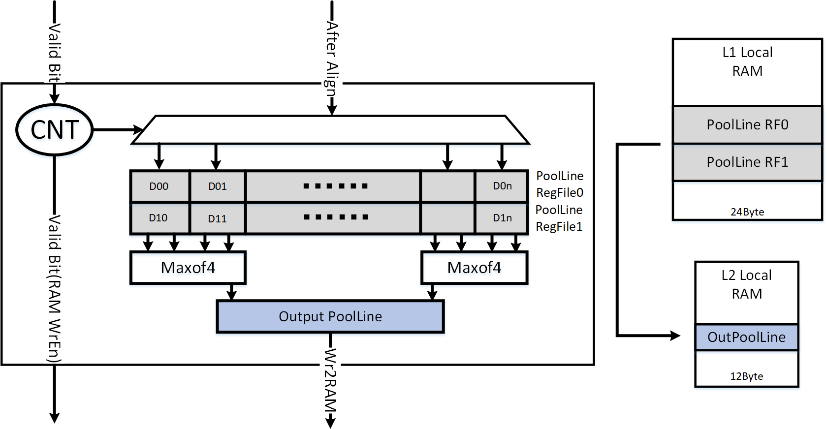
每个运算单元的处理结果经过Relu模块进行激活，在通过Align模块将数据截取成8位定点，并且保证数据的小数点位数与下一层定点数的小数点位数相匹配，这是因为动态定点数的加法中要去两个被加数的小数点位数一致才能保证加法的正确。在进行一系列处理过后，对结果进行池化运算，池化运算不作为一次单独的访存模块，而是在卷积后立刻池化，如图3-11所示，池化寄存器将卷积完成后的数据一次写入两行寄存器中，写满后直接组合逻辑得到最大值并拼接为一行作为下一级RAM的输入。

图3-11 池化寄存器

最后，截取过后的数据存储到通道RAM中，通道RAM的数据可以通过多路复用器的选择，作为第二层卷积层运算的输入数据。第二层卷积运算，由于在软件算法上使用相同的的卷积核，相同的激活函数，相同的池化函数。所以我们将第一层的结果通过多路复用器再次连接到卷积层加速中，实现了运算单元的复用，节省了资源。

数据对齐运算单元是本次设计的一个重要创新点。由于本次神经网络运算采用8位动态定点数进行运算，在进行乘法和加法过后，8位的动态定点数的小数位会发生变化。在下一层网络的运算过程中，由于两个定点数小数位数的不同，两个数值相加的结果会发生错误。因此，在一层网络的结果计算完成过后，我们还需要将计算结果进行截取并且使定点数的格式与下一层权重数据的格式对齐。

### 3.7.2 全连接层加速器

由于FPGA的资源有限，我们在卷积层使用了72个乘累加并行运算，留给全连接层的资源不多。因此对于全连接层，由于其在软件层面对全连接层进行了简化，我们仅仅使用24个乘累加单元并行运算。

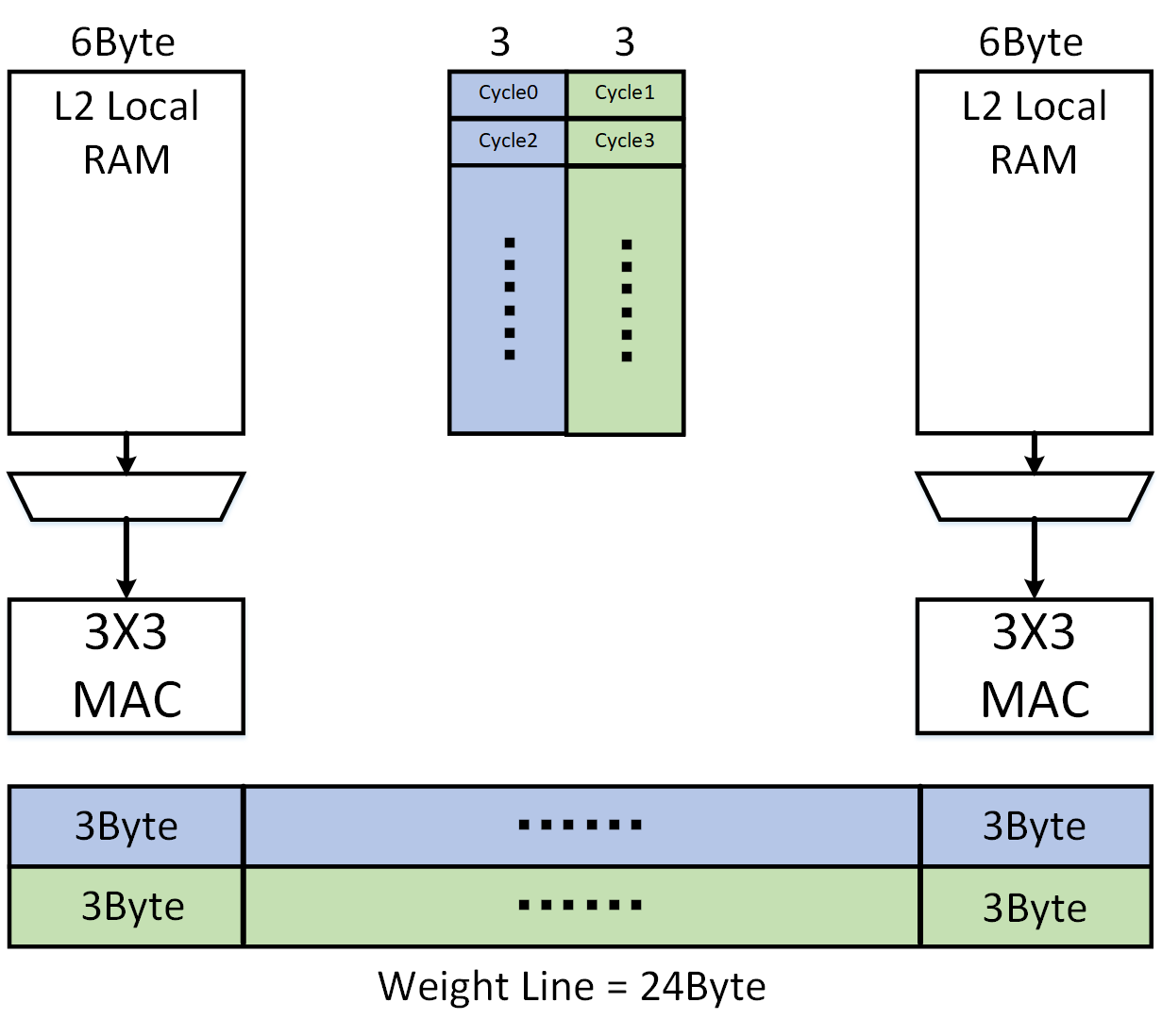
如图3-12所示为全连接层加速器权重匹配。其设计思路跟卷积层加速器类似，运算单元为24个定点乘法器和24个定点加法器。第一层全连接层的运算，从权重RAM中读取权重值，再从上一层16个通道的数据RAM读取数据并且选择输出运算。由于权重ROM每行为24Byte，而8通道的全连接输入RAM，每个RAM每行为6Byte，所以总共一行需要48Byte的权重数据。因此一行计算被拆分为两个周期，第一个周期计算每个通道的低3Byte，第二个周期计算每个通道的高3Byte。

图 3-12 全连接加速器与权重宽度匹配模式

第一层运算的结果将会得到16个8位定点数，再将16个数分批输入到乘累加结构中，得到10个数字的对应的可能性值。最后，拉高valid信号，总线读取10个寄存器的值，就能识别出图片的数字。

# 软件介绍

Cortex-M3软件主要用于流程控制与图像切割定位。

## 4.1.图像定位与切割

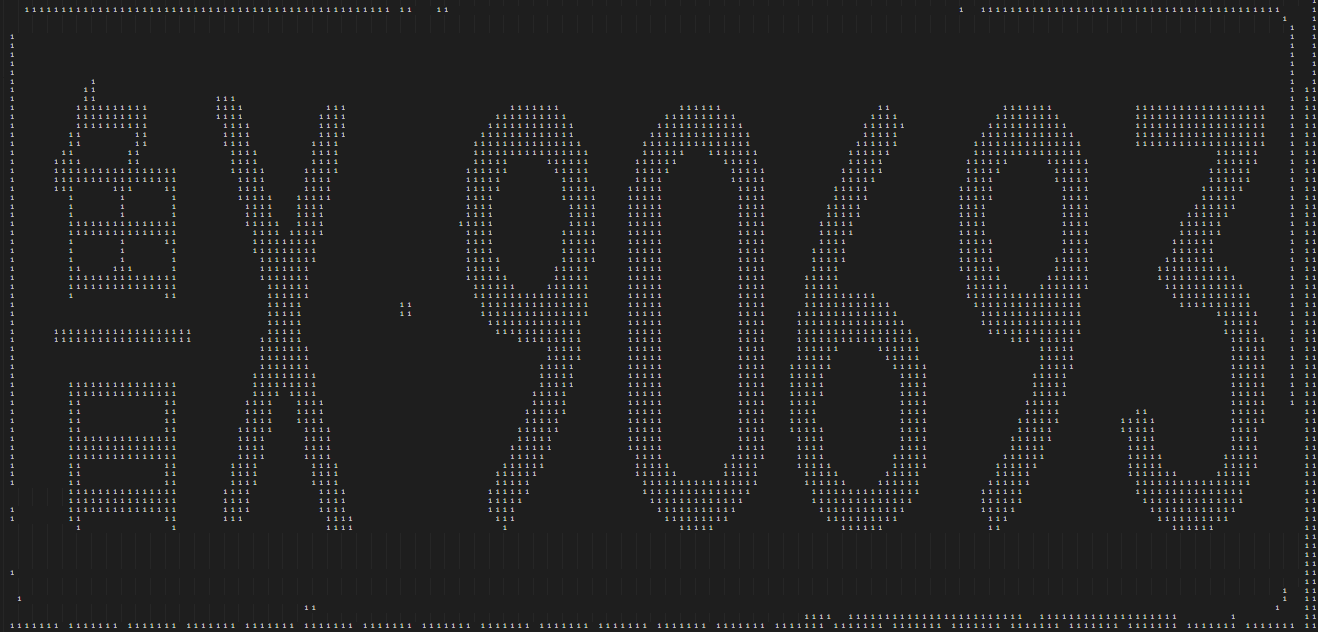
在上一章CAMERA一节中介绍了摄像头数据硬件采样与二值化方法，摄像头输出数据二值化后保存在RAM中，通过UART将图像数据发送至PC后打印出的部分车牌如图4-1所示。

图4-1 QVGA分辨率下二值化车牌图像

### 4.1.1 数字定位

数字定位步骤采用软件行列扫描，通过读取摄像头RAM数据，计算每个数字的横纵坐标，实现定位。

* 从上往下行扫描，计数1的个数，若小于50%，则认为到达车牌顶部蓝色边界，记录TOP坐标
* 从下往上行扫描，计数1的个数，若小于50%，则认为到达车牌底部蓝色边界，记录DOWN坐标
* 从左往右列扫描，计数1的个数，若不全为1，则认为到达车牌左侧蓝色边界，记录LEFT坐标
* 从右往左列扫描，计数1的个数，若不全为1，则认为到达车牌右侧蓝色边界，记录RIGHT坐标
* 从上往下行扫描，若全为0，则认为到达车牌内部白线内的蓝色部分，更新TOP坐标
* 从下往上行扫描，若全为0，则认为到达车牌内部白线内的蓝色部分，更新DOWN坐标
* 从上往下行扫描，若不全为0，则认为到达车牌内数字顶部，更新TOP坐标
* 从下往下行扫描，若不全为0，则认为到达车牌内数字底部，更新DOWN坐标
* 根据绝对值比例切除左侧汉字与英文字母部分，跟新LEFT坐标

### 4.1.2. 数字切割

完成数字总体定位后，开始每个数字的定位，计算每个数字的起始横坐标，从左往后列扫描：

* 若不全为0，到达数字左边界，记录数字起始横坐标
* 若全为0，到达数字右边界，记录数字结束横坐标，计数器加加

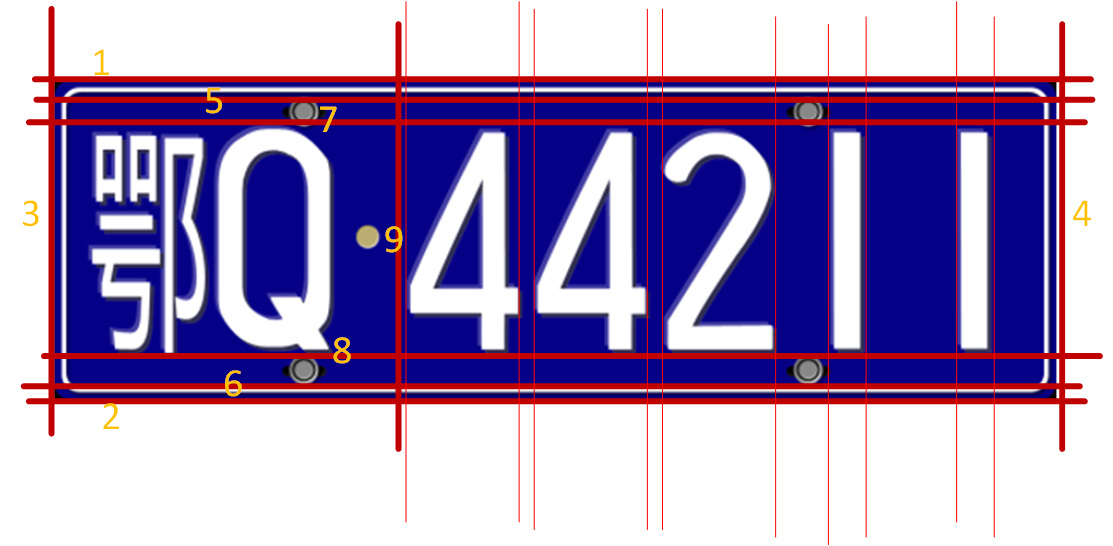
下图4-2展示了定位步骤。

图4-2 车牌数字定位

完成定位后，通过UART将部分切割以及数字坐标传输到PC上打印如图4-3所示。

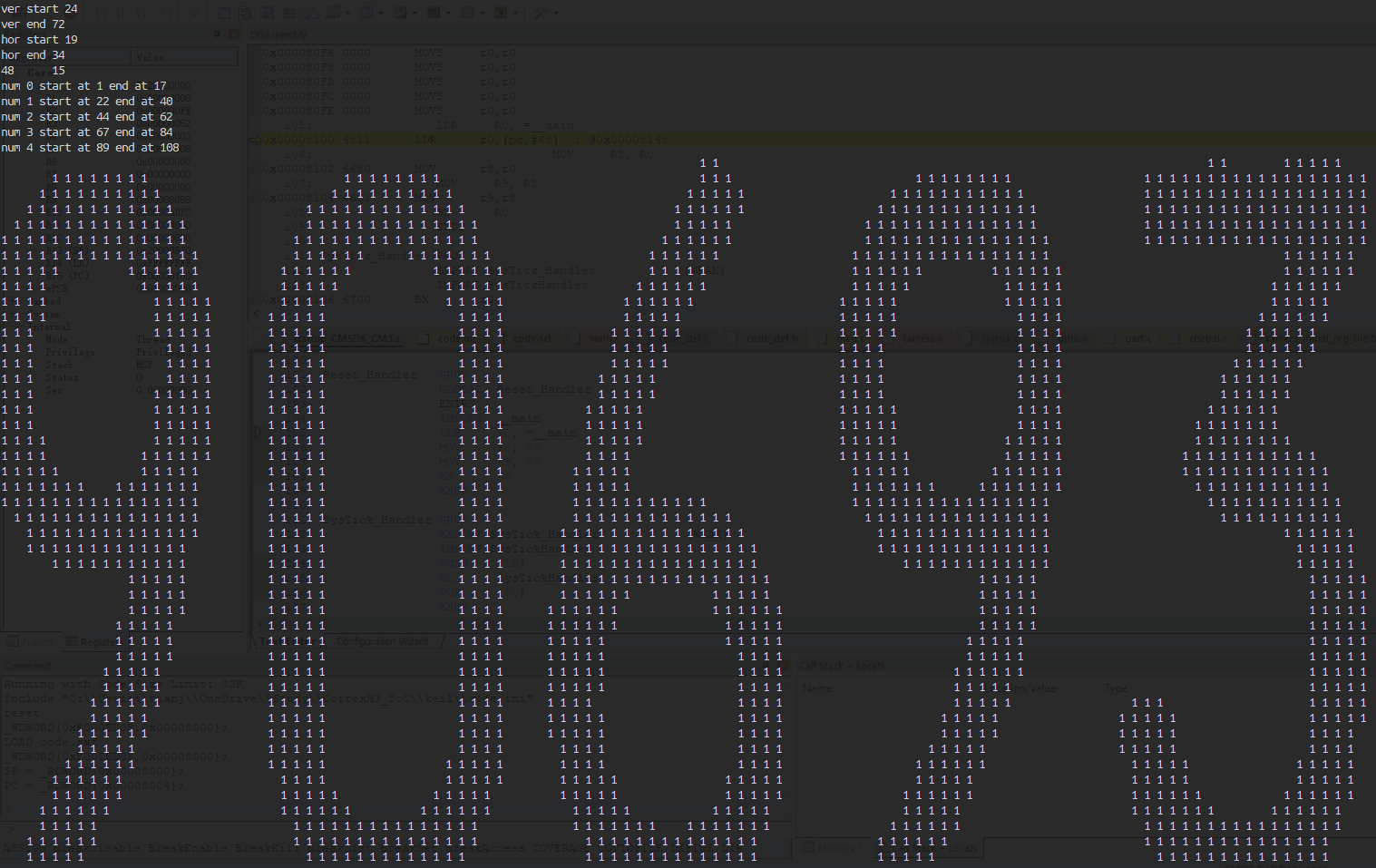


图4-3 数字定位

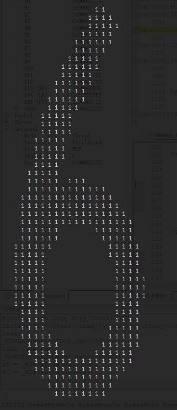
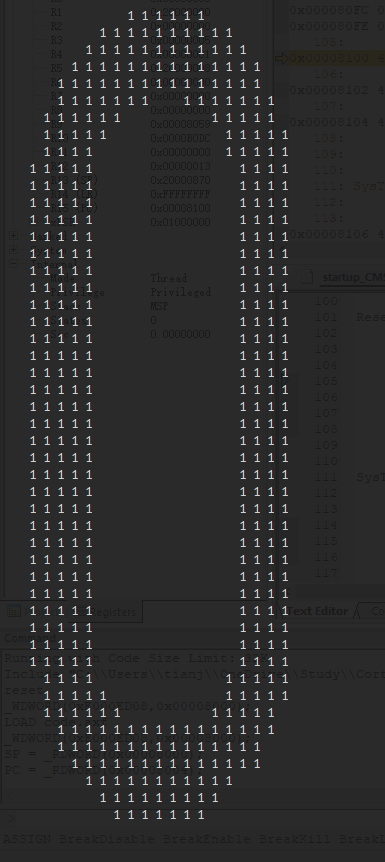
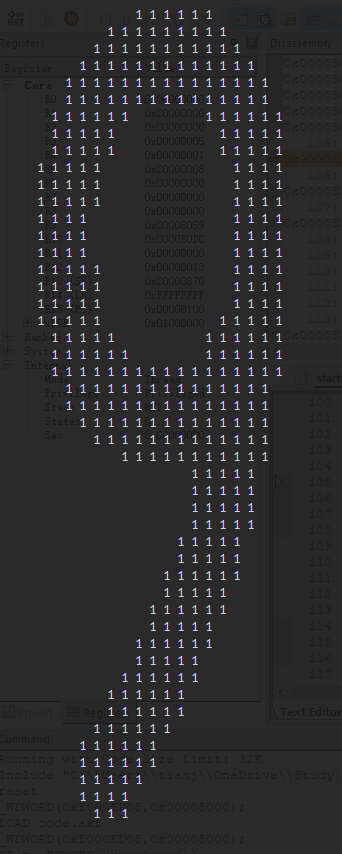
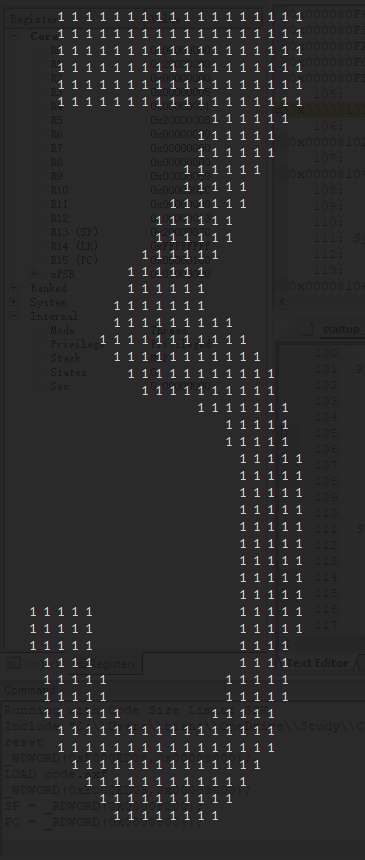
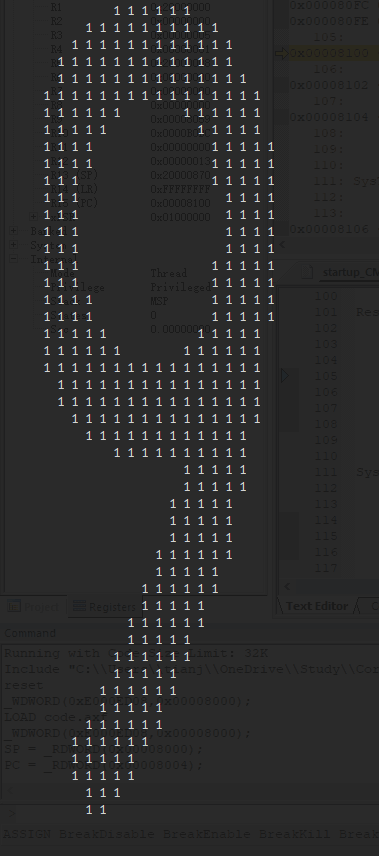
在得到每个数字的坐标后，开始向主存中依次搬运数字，并且搬运的同时实现数字切割对齐，使其居中保存为50\*3Byte，作为神经网络的输入数据。将拼接对齐后的数字通过UART传输到PC中打印，如图4-4 所示。

图4-4 切割定位出的数字

## 流程控制

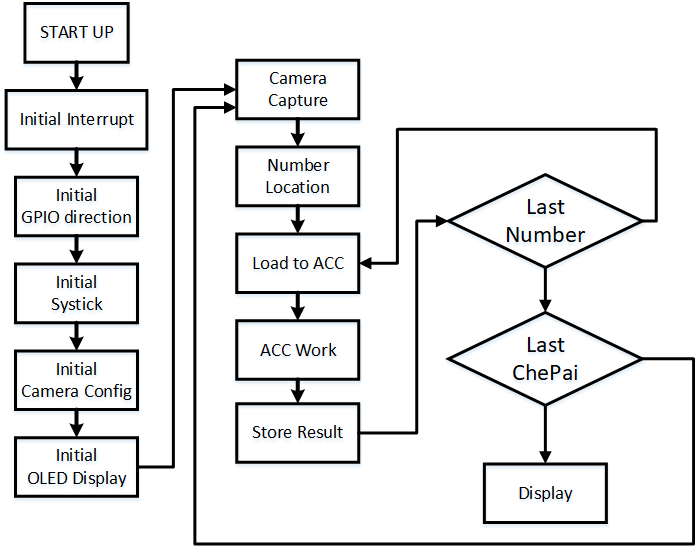
SoC的工作流程如下图4-5所示。

图4-5 软件流程

软件流程大致分为5个步骤：

1. 初始化系统。对系统的摄像头。OLED等外设进行初始化；
2. 按下按键开始存储摄像头图片；
3. 软件算法对图像进行分割、定位；
4. 将图片的数据加载到加速器中，并存储结果；
5. 最后一个车牌识别完成后，按下按键即可循环显示识别到的车牌。

# 总结

本次设计的系统，我们基本完成了题目所要求的功能。我们最大化FPGA内部的资源，本次识别的CNN算法中，采用8位动态定点数能够让我们的加速器达到更大的并行度，我们全连接层加速器的并行度为24，卷积层加速器的并行度达到了72。经过我们的实测，我们采用CNN加速器的形式对车牌识别的算法进行加速，我们识别20张车牌花费时间的1.9s，并且准确率达到了100%。

# 参考文献

[1] 伊恩.古德费洛.深度学习[M].北京:人民邮电大学出版社,2017:201:210

[2] AMBA3 AHB Lite Protocol Specification [J]