北京郵電大學
Beijing University of Posts and Telecommunications

2016140402-ZQ

硕士研究生学位论文阶段报告

学　　号: 2016140402

姓　　名: 宋子恒

学　　院: 计算机学院

专业(领域): 计算机技术

研究方向: 物联网与嵌入式系统

导师姓名:

北京邮电大学

2018 年 11 月 24 日

论文题目	基于异构计算架构的嵌入式交通标志牌识别系统的设计与实现		
论文类型	应用研究	选题来源	其他
开题日期	2017-11-29	是否开题题目	是
论文开始日期	2017-11-29	报告日期	2018-11-28
报告地点	教三 912	报告时间	上午 8:00

研究内容简介

1、研究背景和意义：

(一)选题背景

人工智能概念提出已经 60 余年随着物联网大数据的发展，越来越多的人工智能应用逐步实现，自动驾驶技术是目前最引人注目的人工智能技术集成应用之一。

自动驾驶中，很重要的一个问题就是自动识别道路边的交通标志。这个部分需要很高的实时性才能够保证行车安全。由于自动驾驶对于交通标志检测实时性的高要求，单纯的 CPU 硬件架构已经很难满足系统高实时性的要求。随着半导体技术的不断发展，近年来有一些新的硬件开始广泛的应用在图像处理和人工智能领域，为计算密集的系统加速，这其中就包括 FPGA 和 GPU。

FPGA(Field－Programmable Gate Array) 即现场可编程门阵列。随着 FPGA 芯片设计集成度不断提高，越来越复杂的功能可以在 FPGA 上得以实现。由于 FPGA 的可并行性，许多在传统 CPU 上只能串行执行的任务和算法可以充分利用 FPGA 的并行性来重新设计和实现。大大的提高了算法的执行速度。而且 FPGA 10W~30W 的功耗相比传统的 CPU 和 GPU 100W~200W 的功耗而言有着更低的功耗，更适合于对于功耗要求较高的嵌入式系统中。

OpenCV 是一个基于 BSD 许可（开源）发行的跨平台计算机视觉库，可以运行在 Linux、Windows、Android 和 Mac OS 操作系统上。它轻量级而且高效——由一系列 C 函数和少量 C++ 类构成，同时提供了 Python、Ruby、MATLAB 等语言的接口，实现了图像处理和计算机视觉方面的很多通用算法。

在此背景下，本研究将设计并实现一套基于 CPU+FPGA 的异构计算的硬件的嵌入式交通标志识别系统。该系统配合卷积神经网络及 OpenCV 的图像处理识别算法，将这些算法的运算密集部分使用 FPGA 进行加速。实时的从摄像头采集到的车前方的图像中提取出相应的交通标志牌上的信息，并将这些信息反馈给汽车的控制系统，使汽车自动做出相应的反应动作。同时该系统能够满足嵌入式场景下对于系统低功耗的要求。

2、国内外研究现状：

当前已经有的关于交通标识识别算法的研究。一般的路标检测是基于颜色阈值或者形状分割定位路标的具体位置。使用形状检测来分割交通标志时，摄像头拍摄到的交通标志会因为拍摄角度的问题导致图片中交通标志的形状发生改变导致识别率偏低且运算量大。由于交通标志的颜色较统一，故有很多论文使用 RGB 颜色空间作为识别标准，但由于 RGB 颜色空间受光照影响明显，会降低识别准确性。后来又有人提出了基于 HSV 颜色空间的交通标志识别方法，很好的克服了光照强度对交通标志识别的影响。

对于交通标志内容的识别方法常用的方法有模板匹配法。即计算待识别的图片和一个或几个特定模板图片的相似度完成分类。但是这种方法对于待识别图片的角度很敏感，现实环

境中路标的角度是多变的，所以该方法并不合适。机器学习和深度学习的发展为这一问题带来了新的机会，很多文献中使用卷积神经网络或者 SVM 算法来完成图片内容的识别。卷积神经网络是近年发展起来的，并引起广泛重视的一种高效识别方法。20 世纪 60 年代，Hubel 和 Wiesel 在研究猫脑皮层中用于局部敏感和方向选择的神经元时发现其独特的网络结构可以有效地降低反馈神经网络的复杂性，继而提出了卷积神经网络（Convolutional Neural Networks-简称 CNN）。现在，CNN 已经成为众多科学领域的研究热点之一，特别是在模式分类领域，由于该网络避免了对图像的复杂前期预处理，可以直接输入原始图像，因而得到了更为广泛的应用。

进行交通标志牌的识别是计算密集型任务，如果是从高帧率的视频中去识别，具有很高的实时性要求，只在传统的单纯 CPU 硬件架构上运行速度很慢，难以达到实际使用的目的。有少部分学者使用 FPGA 来对交通标志牌识别系统的部分关键节点进行硬件加速，但只限于在前期的图像预处理上，后期也只是使用简单的模板匹配作为识别算法，未使用神经网络作为交通标志的识别算法，准确性和鲁棒性较差。如果将神经网络以及图像前期预处理都放入 FPGA 中，可以使得原来需要数个 CPU 周期完成的计算任务能够通过 FPGA 在很少的时钟周期内就可以完成。同时在 FPGA 上进行合理的硬件结构设计,提升 FPGA 的资源使用率和吞吐量。

（一）研究内容

研究内容是基于异构硬件加速的交通标志牌检测系统的设计与实现。实现一套能够将汽车行驶过程中前方的视野中的路标检测识别出来并通知汽车控制系统和司机的系统。致力于在整个的系统设计中，体现软硬协同的设计思路，即在 FPGA 部分实现图像预处理中关于色域转换，降噪，膨胀，腐蚀，神经网络中的卷积池化等适合于并行和流水化的操作以来提高整体系统关键点的速度和吞吐量。在 CPU 部分，通过软件来完成硬件数据流管理管理，通知用户，状态切换，结果输出等一些流程控制上的工作。将 FPGA 的高速性和软件在流程控制上的灵活性有机的结合起来。

该系统系统整体的硬件结构如下

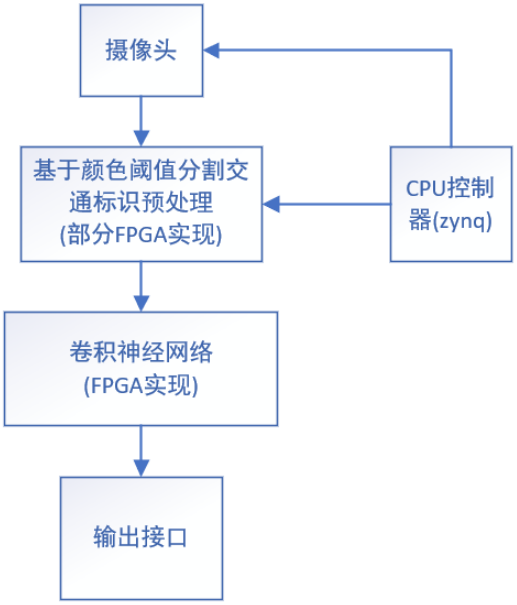


图 1.1 系统总体结构

研究点 1 卷积神经网络的 FPGA 加速

研究目标：**在 FPGA 实现卷积神经网络并获得较好的加速效果**

具体研究内容：

CNN 是著名的深度学习架构，从人工神经网络扩展而来，它已经大量用于不同应用，包括视频监控，移动机器人视觉，数据中心的图像搜索引擎等。受生物视觉神经行为的启发，CNN 用多层神经元相连处理数据，在图像识别中可获得很高准确率。

一个典型 CNN 由两部分组成：特征提取器 + 分类器。特征提取器用于过滤输入图像，产生表示图像不同特征的特征图。这些特征可能包括拐角，线，圆弧等，对位置和形变不敏感。特征提取器的输出是包含这些特征的低维向量。该向量送入分类器（通常基于传统的人工神经网络）分类器的目的是决定输入属于某个类别的可能性。一个典型 CNN 包括多个计算层，例如，特征提取器可能包括几个卷积层和可选的下采样层。卷积层收到 N 个特征图作为输入，每个输入特征图被一个 K ＊ K 的核卷积，产生一个输出特征图的一个像素。滑动窗的

间隔为 S，一般小于 K。总共产生 M 个输出特征图用于下一卷积层。计算神经网络中的一个卷积层其实质是一个多层嵌套循环的乘法和加法运算。由于 CNN 的这种特殊计算模式，通用处理器实现 CNN 并不高效，所以很难满足性能需求。而 FPGA 由于其可并行计算的特性，基于 FPGA 的加速器由于其更好的性能，高能效，快速开发周期以及可重配置能力吸引了越来越多研究者的注意。

FPGA 对卷积神经网络的加速主要来自于循环展开和设计流水线两方面。循环展开是对于卷积神经网络中存在的大量嵌套循环，在某些层级进行展开，使用 FPGA 的资源并行计算。比如一个 5X5 的二维卷积 可以在 1~3 个时钟周期内同时使用 FPGA 的 25 个乘法器进行运算。这样就实现了原来在 CPU 中需要 25 个乘法周期内完成的工作在一个时钟周期内完成。虽然 FPGA 的主频只有 50MHz~200MHz 只有通用 CPU 的 1/10。但只要进行合理的并行结构设计，就可以将大量的计算并行化。

本系统设计中采用的计算该层卷积的计算部分采用并行向量乘法和加法树相结合的结构，可以在 6 个时钟周期内完成 5*5 的二维卷积运算。同时该部件构成流水线，大量数据连续计算的情况的平均计算一个 5*5 的二维卷积运算近似于一个时钟周期。后面工作内容中会对该部分详细描述。

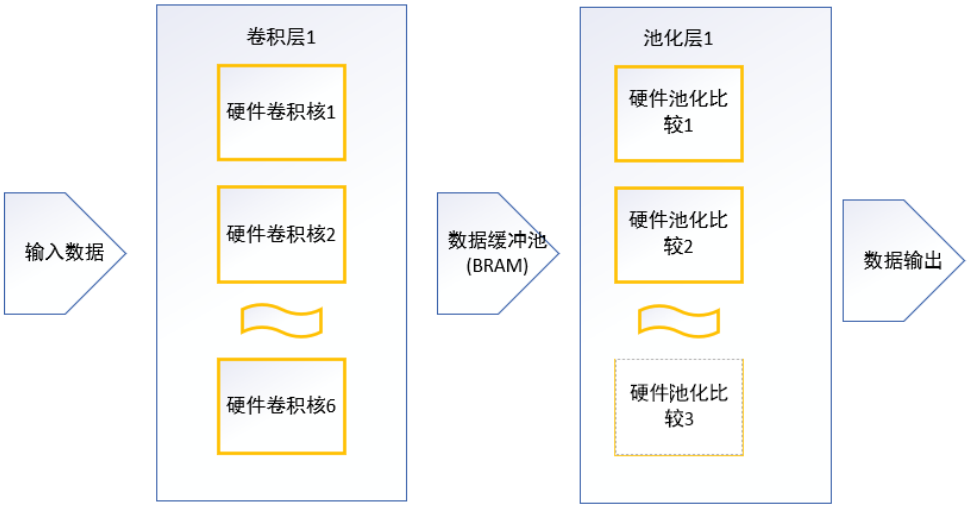


图 1.2：卷积神经网络每层的并行展开和整体的流水线硬件设计

研究点 2 图像预处理在 FPGA 加速

研究目标：**在 FPGA 上可以实现实时的图像预处理**

具体研究内容：

在本系统中，首先需要使用颜色阈值分割的方法从摄像头采集到的图片中选定大致的交通标志所在的位置，其中涉及到大量的色域变换，比较，过滤以及膨胀腐蚀运算操作。虽然对于一个像素的运算简单，但是对于视频流中的大量需要实时处理的图片，CPU 无法达实时的处理要求。 而用 FPGA 做图像处理最关键的一点优势就是：FPGA 能进行实时流水线运算，能

达到最高的实时性。因此在一些对实时性要求非常高的应用领域，做图像处理基本就只能用FPGA。例如在一些分选设备中图像处理基本上用的都是FPGA，因为在其中相机从看到物料图像到给出执行指令之间的延时大概只有几毫秒，这就要求图像处理必须很快且延时固定，只有FPGA进行的实时流水线运算才能满足这一要求。

FPGA进行图像处理的优势在于进行的实时流水线运算和DSP，GPU等进行的图像处理运算不同。DSP，GPU，CPU对图像的处理基本是以帧为单位的，从相机采集的图像数据会先存在内存中，然后GPU会读取内存中的图像数据进行处理。假如采集图像的帧率是30帧，那么DSP，GPU要是能在1/30秒内完成一帧图像的处理，那基本上就能算是实时处理。

FPGA对图像进行实时流水线运算是以行为单位的。FPGA可以直接和图像传感器芯片连接获得图像数据流，如果是RAW格式的则还可以进行差值以获得RGB图像数据。FPGA能进行实时流水线处理的关键是它可以用其内部的Block Ram缓存若干行的图像数据。Block Ram类似于CPU里面的Cache，但Cache不可控制的，但Block Ram是完全可控的，可以用它实现各种灵活的运算处理。这样FPGA通过缓存若干行图像数据就可以对图像进行实时处理，少量数据数据就这样一边流过就一边处理，不需要送入DDR缓存了之后再读出来处理,大大减小了开销。

本系统预处理的过程中滤波、取边缘、膨胀腐蚀等算法一大类用3x3到NxN的算子进行的和卷积神经网络中最前面的卷积层运算是类似的，所以也可以借鉴卷积神经网络在FPGA上应用的例子。FPGA进行的这种算子法处理是并行流水线算法，其延时是固定,可以根据时钟周期来直接计算FPGA对图像进行预处理的速度。在密集运算中，消耗大量时间的操作往往不是运算本身，而是把数据在内存之间的搬移。GPU，CPU在进行运算时需要不停地在内存中读取和写入数据，这样内存带宽往往成了运算速度的瓶颈，数据读写过程中的功耗占的比重也不会小。FPGA则可以通过并行很多计算硬件的方法把要做的运算都展开，然后数据从中流过，完成一个阶段的运算之后就直接流入第二个阶段，不需要把一个计算阶段完成后的数据再送回内存中，再读出来交给下一个阶段的运算。如此可以节省很多时间。

研究点3 软硬件协同设计

研究目标：在zynq上实现软件硬件对数据的协同处理

具体研究内容：

在上一个研究点中预处理部分有算法在FPGA上难以实现，所以需要将这部分算法的执行放到CPU中使用软件计算，这其中涉及Zynq中PL（FPGA）部分和PS（CPU）的数据交互问题。为了加快数据交互处理的能力，可以借助Xilinx的AXI总线， 该总线频率高，相对较外部DDR Ram交互更快，不易使数据传输成为整个处理流水线中的瓶颈。

(二)关键技术

本系统需要解决的关键问题包括算法上的和系统实现上的问题。

1. 算法方面的关键技术：

（1）使用Lenet-5 卷积神经网络作为最终识别的。

（2）图像预处理中需要是用那些处理步骤来获得更好的图片分割效果，使用的算子的大小以及阈值应该如何确定。

2. 系统设计层面的关键技术：

(1) 流水线设计以及计算资源的分配。基于FPGA现有的资源,结合确定好的神经网络结构设计关键的部件,使用多大的并行程度来保证系统能够在规定的时间内处理完数据并给出结果，以及合理的根据不同部分的运算密度来分配FPGA资源，提高FPGA整体的资源使用率，

并保数据处理速度在整个卷积神经网络中的不同部分都相对均衡,防止出现流水线中某些层处理负担过重而其他层却空转这种资源分配不均衡的情况。

（3） 设计好的卷积计算单元。在尽可能少的时钟周期内完成卷积的计算并且能够保证计算卷积的部件是具有流水线结构的，能够更充分的利用时间和硬件资源。

（4）如果受限于单个 FPGA 芯片的资源问题， 是否应该考虑使用多块 FPGA 级联来构成一个系统，不同层级系统之间的通信速率应该使用什么样的借口来保证。

（5）神经网络的参数中浮点数的处理。FPGA 本身并不合适使用大量的浮点数运算，这会导致消耗掉大量的逻辑资源。所以需要考虑使用定点数运算。需要根据 FPGA 现有的资源的使用情况。具体的需要确定定点数的位数，以及不同的位数的精度损失对整个系统最终造成的影响。

(6)使 CPU 能够协调好大量数据量在系统中的流动。对于预处理算法中稍微复杂不方便使用 FPGA 完成的部分应该如何协调至 CPU 进行处理，如何减少从 FPGA 至 CPU 切换过程中的数据交换量，保证整个系统的流水线不发生拥堵。

(7)设计好系统的预处理程度。如果预处理过少，会导致目标图像识别区域范围过大，把很多本来不相关的物体过多的纳入范围。如果预处理过多，又有可能导致原本的物体的特征被消除的过多，导致目标区域丢失。其次在于如何选定目标区域的特征， 比如识别路标时，如果以路标的颜色作为首要的特征，那么应该使用何种颜色空间来进行判别，阈值应该如何设定，其他的形状等特征是否可以使用， 都需要在后续的研究中通过实验的结果来确定。

(3) 如何能尽量减少大量数据在 CPU 中通过的时间提升系统性能。根据目前的研究来看，1080*1920 的一帧图片大约 3MB 大小， 比如使用 AXI 总线 64bit 宽度在 FPGA 时钟 50MHz 的情况下传输只需要 8ms 所以目前考虑使用流水线结构使用 AXI 中线来将各个级别的处理模块串联起来对每一帧图片进行分析，一方面是的系统的吞吐率得以提升不会出现数据积压在输入端导致丢帧的状况，另一方面使用标准的 AXI 总线可以利用一些既有的 IP 核来实现部分功能，能够缩短设计验证周期。另一方面使用 AXI 接口对模块本身标准化也有利于增强这些模块的可重用性。

(4)应该如何选择摄像头的分辨率，分辨率过高会导致内部总线负担过重和数据丢失的现象，如果分辨率过低又会导致目标失去关键的特征而无法辨认，所以应该在两者之间找到一个平衡，这个问题可以通过先在 PC 上的算法验证来得到一个准确率较高的方案。

论文计划及安排

时间	研究内容	实际完成内容
2017.12-2018.01	Verilog 及 FPGA	熟悉了使用 Verilog 的 FPGA 的开发以及使用 Vivado 调试和仿真
2018.02-2018.04	基于颜色阈值的交通标志分割	实现了从视频中提取和分割交通标志的处理程序
2018.05-2018.07	深度学习以及交通标志识别	使用 Keras 搭建了 Lenet-5 卷积神经网络
2018.08-2018.10	使用 Verilog 编写并仿真神经网络以及图像预处理部分	完成了神经网络的设计仿真和综合图像预处理部分的部分算法的 FPGA 仿真完成
2018.11-2019.1	完成所有图像预处理部分并对系统进行综合测试并撰写	待完成
2019.2-2019.3	论文初稿攥写	待完成
2019.04-2019.06	论文终稿攥写	待完成

论文进度

时间	研究内容	预期效果
2017.12-2018.01	Verilog 及 FPGA	可以完成基本的
2018.02-2018.04	基于颜色阈值的交通标志分割	可以从视频中截取交通标志所在的区域出来供后续神经网络处理
2018.05-2018.07	深度学习以及交通标志识别	训练一个可以分类交通标志的卷积神经网络
2018.08-2018.10	使用 Verilog 编写并仿真神经网络 以及图像预处理部分	完成毕业论文 70%以上的内容并完
2018.11-2019.1	对系统进行综合测试	完成所有研究工作
2019.2-2019.3	撰写论文	完成所有材料及代码的集成

论文进展情况

根据前期开题报告中的计划，论文进展顺利。并取得了一定的成果

一、在理论研究方面：

1.使用了使用 Keras 深度学习框架搭建 lenet-5 的卷积神经网络,为了硬件上的实现方便，将激活函数全部替换为 relu。 在 GTSRB 交通标志数据集上实现了 97%的分类准确度。

2.对于基于颜色阈值的交通标志分割中，通过统计不同标志在 HSV 颜色空间上的分部情况实现了 95%的召回率。

3.在使用定点数精度方面，通过软件模拟实验获得如下结果

二、在系统实现方面：

1.使用 Verilog 完成了 lenet-5 在 FPGA 上的实现，完成了仿真,综合，写入 FPGA。处理一帧图片用时（包括读写缓存）7332 个时钟周期 其中各层读写缓存总时间 3040 个时钟周期，其中各层计算延迟(不包括写缓存)如下

层次	处理所需时钟周期数
卷积层 1	968
池化层 1	1350
卷积层 2	733
池化层 2	400
全连接层 1	487
全连接层 2	175
全连接层 3	179

表 2.1 卷积神经网络各层处理耗时

2. 完成了图像预处理部分在 FPGA 上的一些预处理模块 并且完成了仿真,综合。

工作成果

现阶段的主要工作成果包含两个，如下

(1)软件算法的实现与仿真，包括使用 keras 在 PC 上实现了用于交通标志识别的 lenet-5 、基于颜色的交通标志的分割算法、以及定点数不同位数对模型结果精度的仿真程序。

(2)lenet-5 网络在 FPGA 上的实现与仿真。

下面依次叙述以上完成的两个研究大点。

一、软件算法的实现与仿真

1.基于颜色的交通标志分割。

系统的第一步即要定位交通标志处于图像中的什么位置，所以应当先将交通标志从一张图中提取裁剪出来在送入到后面的神经网络中进行识别。提取的方法一方面要准确另一方面要尽可能的简单便于之后的硬件实现。所以最终考虑使用基于颜色阈值的方法来进行提取。颜色是交通标志特有的特征，而且相对标准，同一类别的交通标志颜色分部的方差相对较小。考虑到 RGB 颜色在色彩空间上并非连续，不太适合只通过一个颜色区间来过滤出有效的内容，故考虑通过色域转换将颜色变换到 HSV(色调，饱和度，明度)颜色空间进行过滤。使用该算法的流程如下。

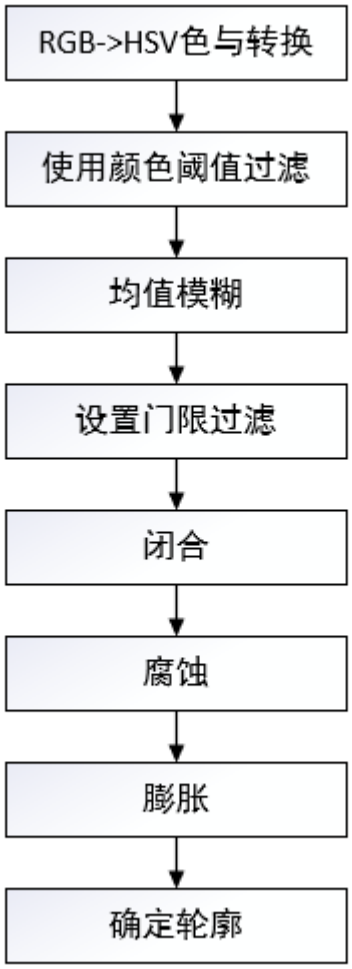


图 2.2 基于颜色阈值的过滤算法

该算法目的是粗选交通标志所在的位置，所以召回率对准确率更为重要，最终通过设置较宽的过滤门限，可以使召回率达到 95%。

最终根据数据统计规律选定的颜色阈值如下：

	H(0~180)	S(0~255)	V(0~255)
Blue	[97.5,117.5]	[64, 255]	[38, 255]
Red	[0,10]U[170,180]	[26, 255]	[38, 255]
Yellow	[12, 32]	[69, 255]	[38, 255]

表 2.2：颜色阈值表

下面以从图像中分割出红色的交通标志为例：

图 2.3.1 原始图像

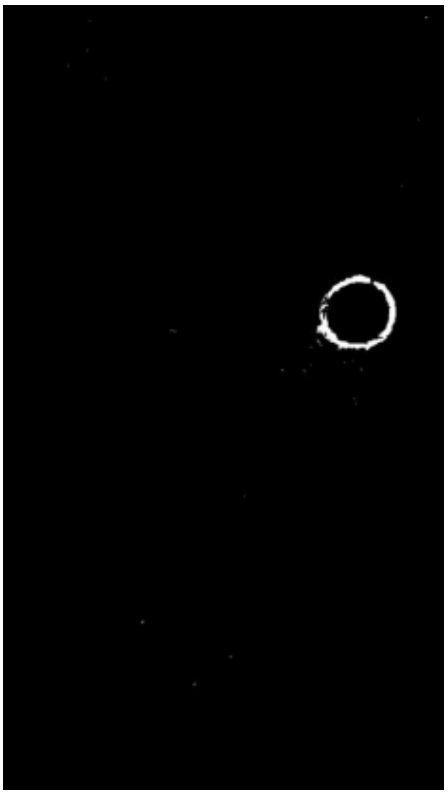


图 2.3.2 使用红色阈值过滤后的图像

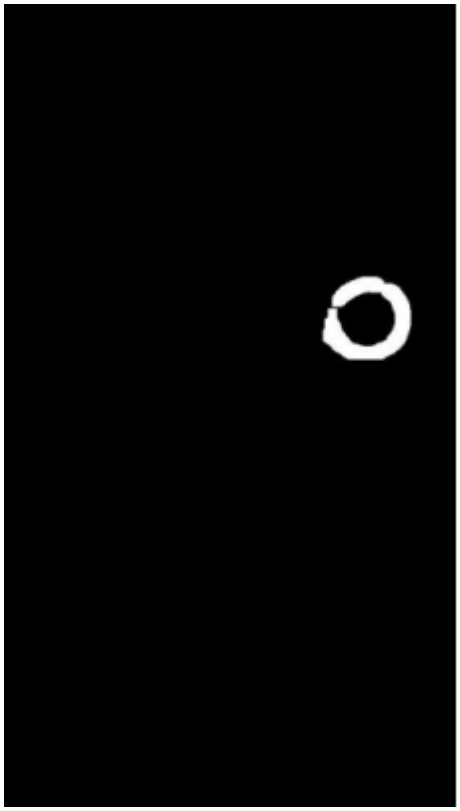


图 2.3.3 腐蚀膨胀后的图像

图 2.3.4 在原图圈出轮廓

2 搭建交通标志识别的卷积神经网络

使用 keras 搭建 lenet-5。 将上一步分割出来的交通标志上的具体内容识别出来需要使用 lenet-5。考虑到 FPGA 资源有限，对于 sigmoid 这类函数并不好实现 所以考虑使用 relu 作为整个网络的所有激活函数，该函数在 FPGA 内部使用比较器即可实现。模型的结构如下

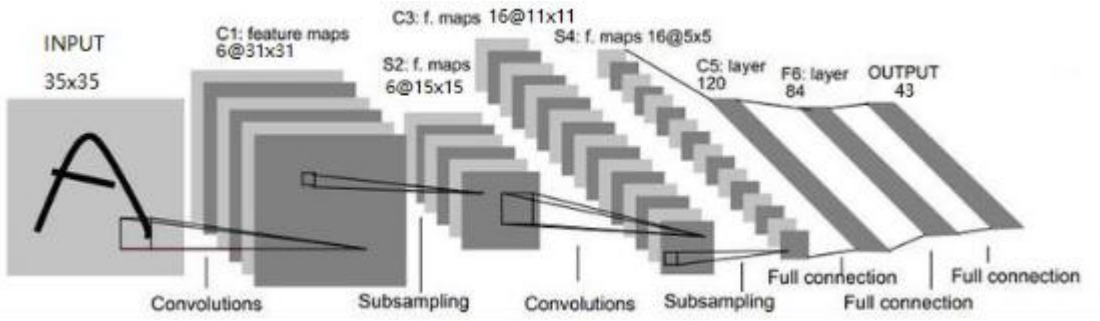


图 2.4 修改后的 lenet-5 结构

3 模拟定点数运算对整个模型的误差。

由于 fpga 不方便使用大量的浮点数运算，而且模型中的参数值也比较小,所以考虑使用定点数来替代原有的 PC 端模型的浮点数,使用 keras 训练好的额模型中的参数 使用 numpy 重写 lenet-5，然后自定义定点数计算类使其模拟定点数计算的过程，可以对这个类设定不同的位宽和精度。使用该类替换 numpy 重写的 lenet-5 中的 numpy 数组中的浮点数。最后根据模型的输出来判断不同定点数精度对模型准确率的影响。最终根据资源使用和对模型准确度的影响，折中后选定 8 位整数 8 位小数的定点数 最小精度 1/256。不同位宽的定点数对模型准确度的影响如下。

不同位宽定点数	模型结果准确度	乘法消耗 DSP 硬核数量
keras 模型基准 (float32)	97.5%	NA
16 位整数 16 位小数	97.0%	4
8 位整数 16 位小数	96.4%	2
8 位整数 8 位小数	96.0%	1

表 2.3： 不同精度浮点数对模型的影响

二、硬件系统的设计与实现

1. lenet-5 网络在 FPGA 上的设计与实现

本部分是本系统的重点

Lenet-5 中的每层的内部硬件结构类似，都是由数据缓冲区，地址生成部件，卷积（全连接层是内积）运算部件构成。

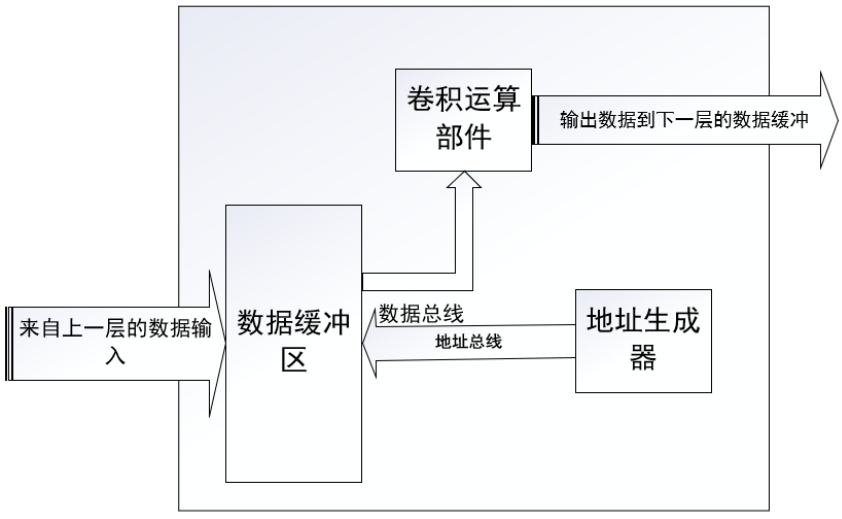


图 2.5： 卷积神经网络每层的大致结构

（1）卷积运算部件

对于卷积网络中大量的乘法和加法运算，使用 Xilinx 的乘法 IP 核以及加法 IP 核。其中乘法 IP 核使用了 FPGA 的 DSP 资源。考虑到整个网络中的卷积核大小都是 5*5 所以构建了如下的求卷积部件：乘法部分使用了一个 25 口的并行乘法元件 可以在一个时钟周期内计算出两个长度为 25 的向量对应位置的积的结果。后面的 26 口加法树 专门用来计算乘法之后 25 个乘积的和 为了减少计算周期 整个加法器被组织为二叉树的结构。这样做完卷积之后的 26 个数（包括了偏置参数 bias）在 5 个时钟周期后给出他们的和的结果。加法树和向量乘法器组合在一起构成了一个流水线。每个周期都可以向里面输入不同的值让其计算卷积。该部件仍然可以在后续的全连接层中计算 两个向量的内积时使用。

计算卷积部件的硬件结构如下：

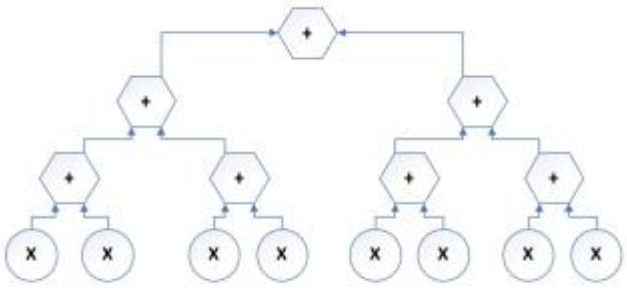


图 2.6：卷积计算部件结构

如上图该部件每一层的运算都是并行的，数据从该部件的底部进入，每个时钟周期向上流动一层。对于 5*5 的二维卷积运算 乘法器消耗一个时钟周期，加法器树消耗 5 个时钟周期，总共计算一个卷积只需要 6 个时钟周期。而且该具有流水线结构，每一个时钟周期就可以向部件里面输入一组卷积数组，所以该部件的数据吞吐量和时钟频率一致。

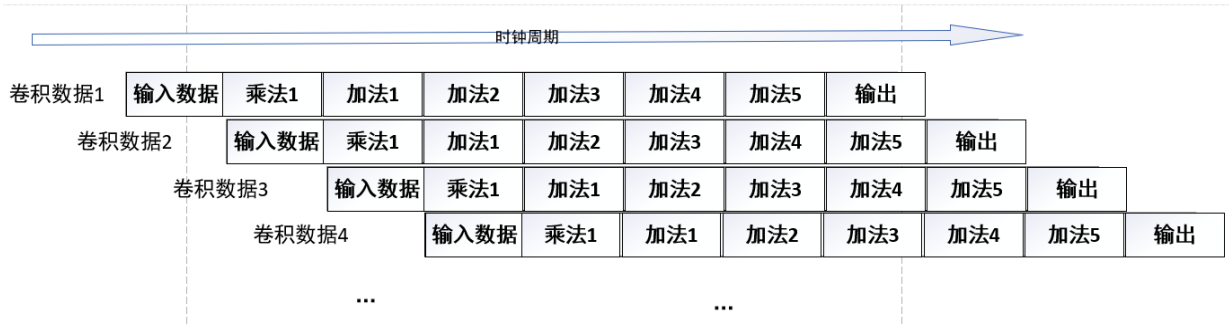


图 2.6.1：卷积计算流水线过程

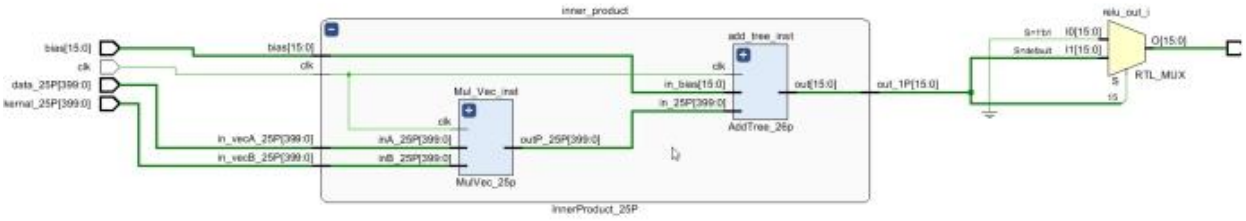


图 2.7：卷积部件 RTL 图

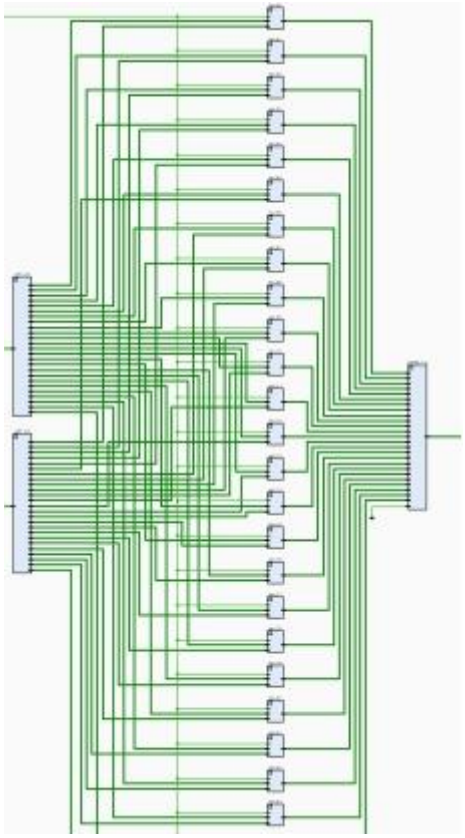


图 2.8：并行乘法部件

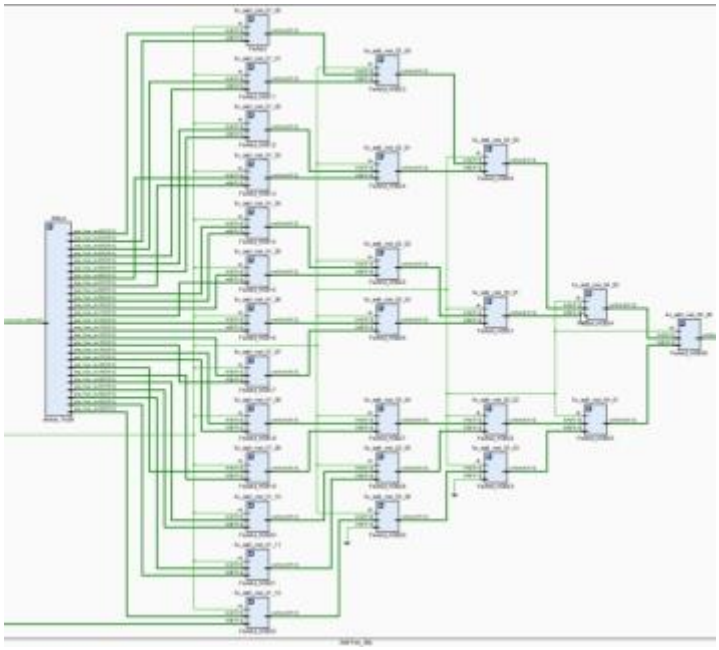


图 2.9：加法树部件 RTL 图

(2)数据缓冲部件

由于数据传输实际占用了整个卷积神经网络计算将近一半的时间，所以需要高效的的缓冲区作为层与层之间的数据交互。该部件需要保证可以被多个端口同时随机读取。而且在读取的同时允许上一层将数据写入该缓冲区 作为下一次读取的数据,且当前读取数据的操作不受影响。基于这样的需求，本系统设计了一个拥有两个缓冲区的缓冲部件，两个缓冲区会被交替的读写。记两个缓冲区分别为 A B 读写流程如下

周期 1 前一层写入 A 当前层读取 B

周期 2 前一层写入 B 当前层读取 A

后面依次交替，如此设计的流水线可以将数据在层和层之间传递的时间减少一半(3000 个时钟周期)。该缓冲部件内部有状态机来控制内部的读写顺序，内部的两个缓冲区对外部的操作者是透明的。

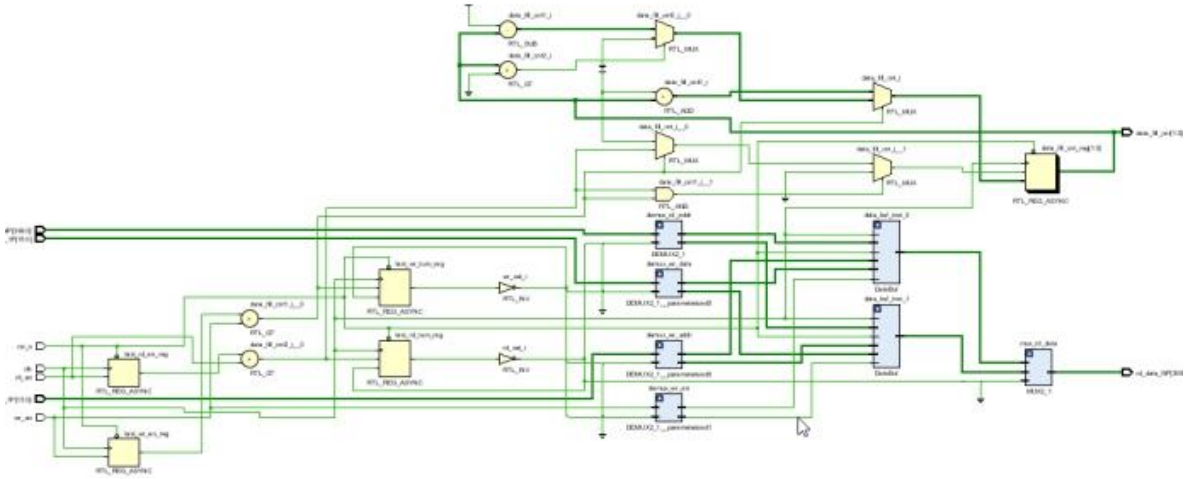


图 2.10 数据缓冲和部件 RTL 图

（3）地址生成器

对于每一层，都需要获取地址后从数据缓冲区里去处卷积或者池化所需的数据。

所以需要地址生成逻辑。不同层的地址生成逻辑不同。对于卷积层和池化层而言，在二维地址空间中需要先生成二维地址区域的顶点，之后以这个顶点在生成其他的点的地址。最后还要将二维的地址通过映射逻辑映射为一维地址才能从数据缓冲区中读取。

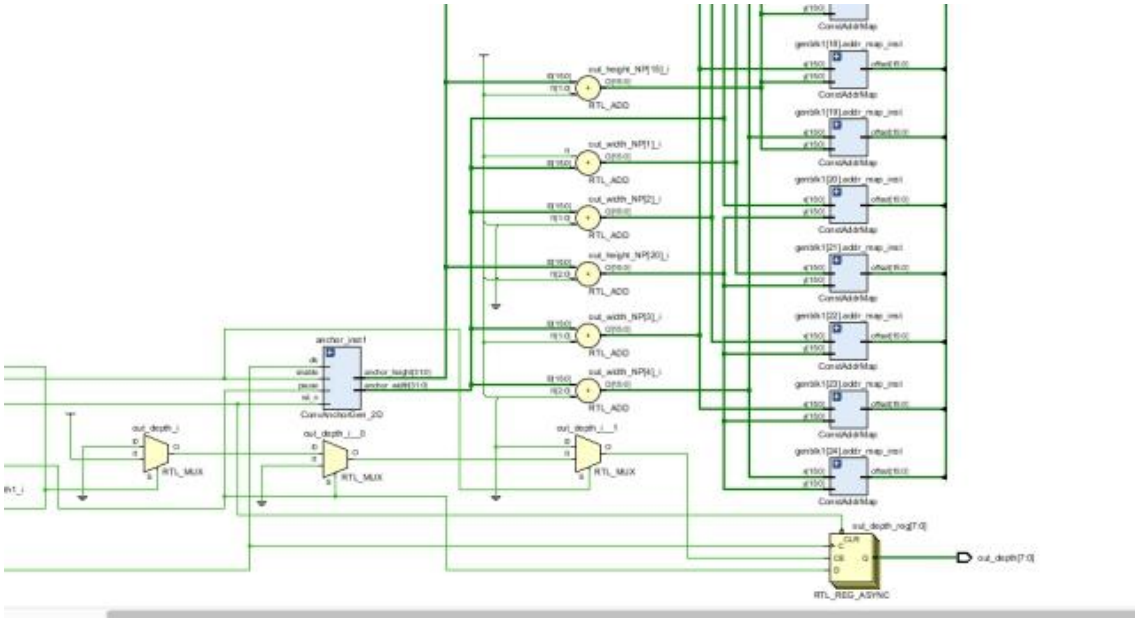


图 2.11：地址生成器整体逻辑

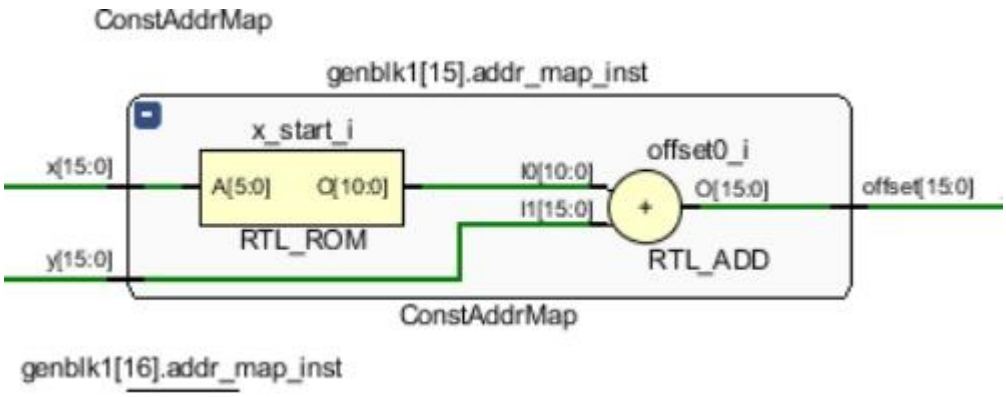


图 2.12 二维地址映射为一维地址逻辑（使用了 ROM 做为乘法器）

(4) 卷积层硬件设计

以第一层卷积层为例：第一层卷积层的输入时一副 35x35 的图像，有 6 个 5x5 的卷积核

所以根据这个特征 给第一层设计了 6 个硬件卷积核，即在同一时刻，6 个卷积核在同时运算。这 6 个卷积核运算卷积的这个周期，地址生成器会生成下一次卷积所需要的 25 个点的地址，在下一个周期输出缓冲区会根据这 25 个地址输出对应二维图像的数据供卷积核使用。同样第二层卷积层有 16 个卷积核，所以设置了 16 个硬件卷积核，同一时刻有 16 个卷积核在同时运算。这样的设计将卷积运算中在卷积核一层的循环以及在每个卷积中的循环全部展开，大大提高了运算速度。同理，池化层也具有类似的结构。

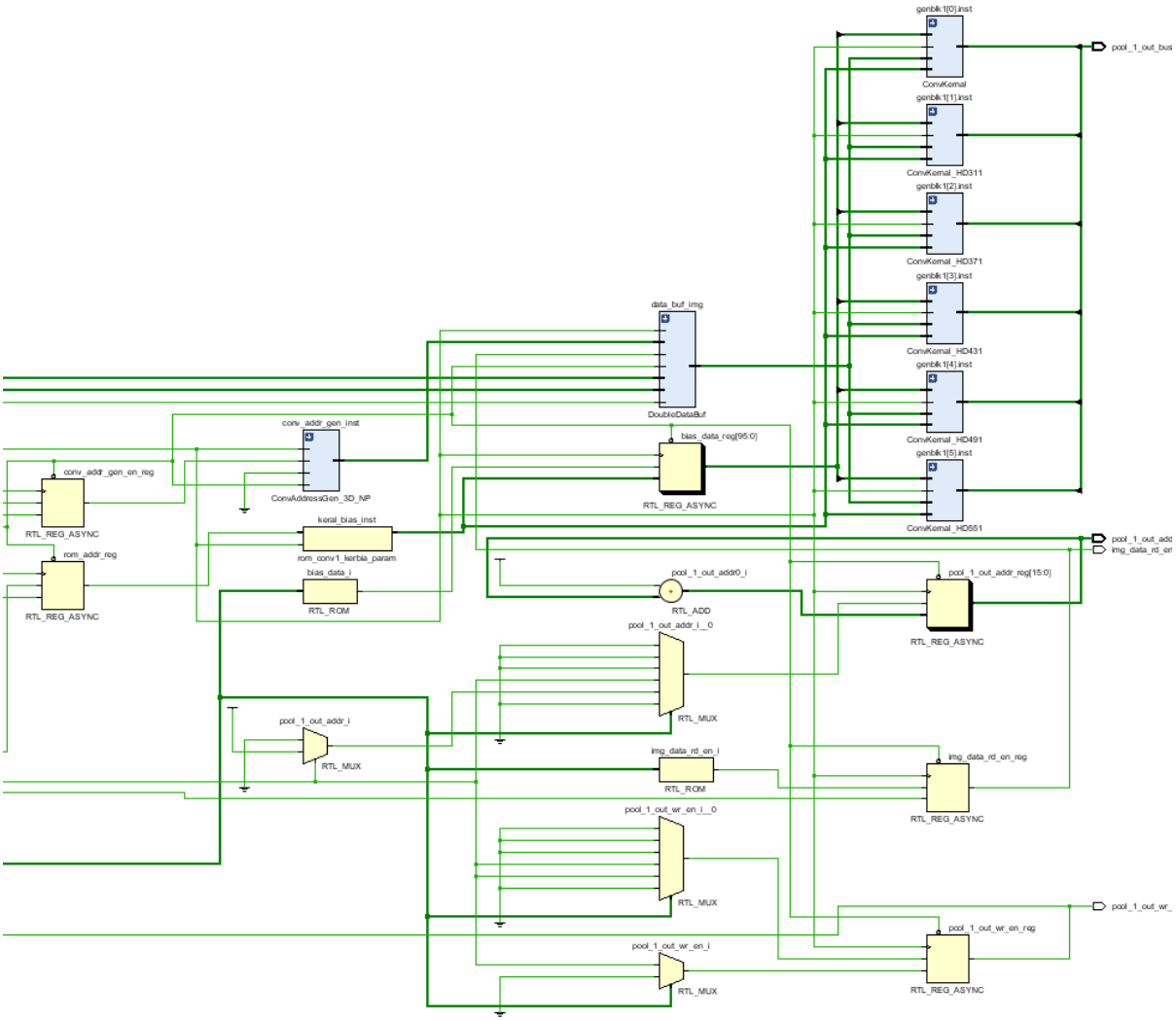
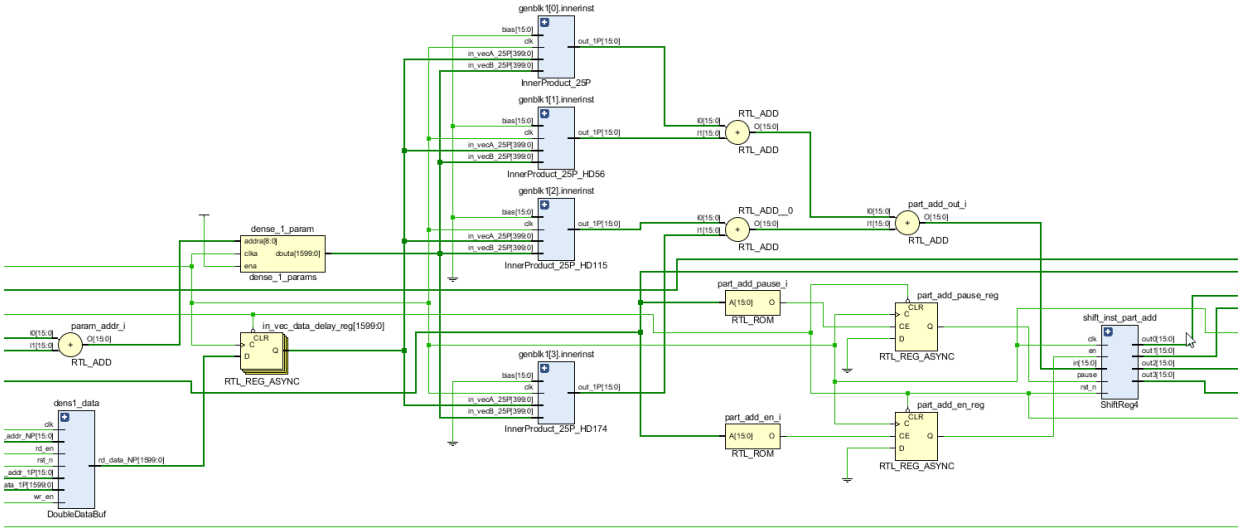


图 2.13 卷积层 1 RTL 结构图

(5) 全连接层硬件设计

本质是矩阵相乘，以全连接层 1 为例 输入是一个长度 400 的向量,输出为长度 120 的向量,那么网络拥有的参数矩阵尺寸就是 120x400。计算过程中会遇到两个长度为 400 的向量的内积的情况。由于 FPGA 资源的限制，该层分配到了 4 个卷积计算单元 每个长度 25.所以一

个时钟周期只能计算两个长度为 4x25 的向量的内积。所以计算产生一行矩阵相乘的结果需要在四个时钟周期向卷积计算单元输入数据。前面 3 个周期计算出来的部分结果需要等待



最后一个周期的计算结果，所以考虑使用移位寄存器来存储前三个周期的结果，在第四个周期将 4 个部分的结果求和然后经过加偏置和 relu 函数输入到下一层的缓冲区中。

图 2.14 全连接层 1 RTL 结构图

(6) 整体系统设计中的负载均衡

卷积神经网络中不同层的计算次数相差很多，但是 FPGA 的资源是有限的 总共可以部署 29 个卷积计算部件。根据各层的计算需求 按照下表进行分配

层名称	乘法次数	分配卷积计算单元数	分配资源后的计算时间（时钟周期）
卷积层 1	144150	6	968
卷积层 2	290400	16	733
全连接层 1	48000	4	487
全连接层 2	10080	2	175
全连接层 3	3612	1	179

表 2.4：卷积计算部件资源分配情况

保证各部分的计算时间相差不多，不会造成流水线过于堵塞形成系统的瓶颈。

(7) 系统仿真与综合

最终系统完成整合，使用全局状态机在顶层模块进行协调控制，仿真与综合下载到 FPGA 开

发板上。选取部分仿真结果如下：

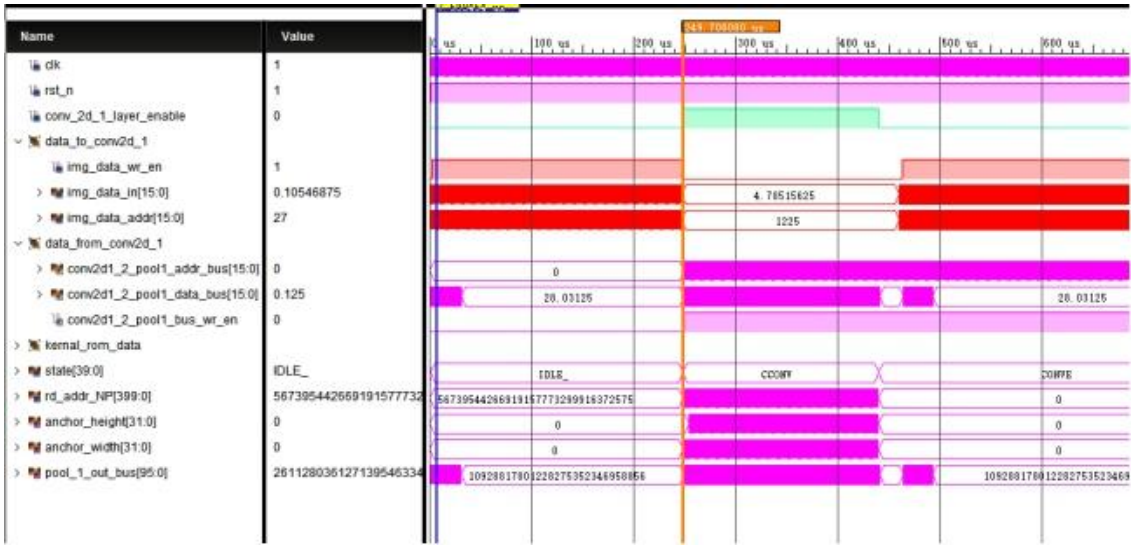


图 2.13：卷积层 1 的仿真结果

(6) 系统性能分析与测试

系统性能分析：

层次	处理所需时钟周期数
卷积层 1	968
池化层 1	1350
卷积层 2	733
池化层 2	400
全连接层 1	487
全连接层 2	175
全连接层 3	179

表 2.5：各层处理图片所需时间

从上面的表中可以看出 一张图片通过所需时间最长的地方在池化层 1 用时 1350 个时钟周期。所整个流水线而言，每隔 1350 个时钟周期就可以处理一张图片。FPGA 的时钟频率为 100MHz。可以计算出吞吐量：

$$\frac{1}{\frac{1}{100M} * 1350} \approx 74074$$ 张 / 秒

使用同样的神经网络在不同平台上处理 10000 张图片的的耗时与该系统作为对比,获得如下效果

测试平台	运行时间(不考虑)	功耗
树莓派 3B	43s	约 5W
台式机 酷睿 i5CPU	6s	90W
FPGA	约 0.15s	约 5W

表 2.6：本系统中 FPGA 和其他平台性能功耗的对比

三、部分预处理算法在 FPGA 上的实现与仿真

该部分目前在 FPGA 上实现并仿真了二值化，阈值过滤，色域转换，模糊等相对简单算法。后续会整合到系统中。

计划及进度安排

后期的工作主要是继续完成全部的图像预处理部分，完成软硬结合的设计，以及最终整个系统的综合测试与联调。

内容	时间	工作量估计
完成预处理部分在 FPGA 上的全部工作	2018.12-2019.1	可完成，已经完成理论研究，缺少代码实现
系统集成测试联调与论文初稿撰写	2019.2-2019.3	可完成，只是工作量的问题
论文终稿	2019.4-2019.6	可完成，只是工作量的问题

问题及整改方案：

(1) 目前预处理部分的轮廓圈定算法较为复杂需要在 CPU 上实现， 需要整合软硬件资源一起完成且不能让 CPU 的处理能力以及 CPU 的计算速度成为整个系统的瓶颈。

(2) 其次摄像头需要使用 USB 和系统进行连接，此过程中需要处理好软件部分嵌入式操作系统的问题，需要搞清楚摄像头的具体协议，如何快速的传输视频数据，是否可直接不通过 CPU 传输到 FPGA 中进行处理， 或者使用 DMA 部件来完成传输。

(3) 对 Zynq 的软件和硬件联合开发流程不是十分的熟悉， zynq 上运行 linux 系统还需要进一步的研究才能完成。

参考文献

[1] Dong Wang, Ke Xu, Diankun Jiang. PipeCNN: An OpenCL-Based Open-Source FPGA Accelerator for Convolution Neural Networks [J] IEEE 2017

[2] Chao Huang. A Layerbased Structured Design of CNN on FPGA[A]. IEEE Beijing Section、Fudan University.Proceedings of 2017 IEEE 12th International Conference on ASIC[C].IEEE Beijing Section、Fudan University:IEEE BEIJING SECTION(跨国电气电子工程师学会北京分会),2017:4.

[3] Yuchen Yao. FPGA-based Convolution Neural Network for Traffic Sign Recognition[A]. IEEE Beijing Section、Fudan University.Proceedings of 2017 IEEE 12th International Conference on ASIC[C].IEEE Beijing Section、Fudan University:IEEE BEIJING SECTION(跨国电气电子工程师学会北京分会),2017:4.

[4] Utku Aydonat, Shane O'Connell, Davor Capalija, Andrew C. Ling, and Gordon R. Chiu. Deep Learning Accelerator on Arria 10. [C] 2018

[5] Kamel Abdelouahab1, Maxime Pelcat1, Jocelyn Sérot,François Berry. Accelerating CNN inference on FPGAs A Survey. [J] 2018

[7] 孙凡. 卷积神经网络加速器的实现与优化[D].中国科学技术大学,2018.

[8] 刘志成,祝永新,汪辉,田犁,封松林.基于 FPGA 的卷积神经网络并行加速结构设计[J].微电子学与计算机,2018,35(10):80-84.

[9] 李泽坤. 基于 FPGA 的卷积神经网络系统的设计与实现[D]. 哈尔滨:哈尔滨工业大学 2017

[10] 王小雪. 基于 FPGA 的卷积神经网络手写数字识别系统的实现[D]. 北京:北京理工大学 2016

[11] 吴晋. 基于 FPGA 的目标检测算法加速与实现[D].北京交通大学,2018.

[12] 周鑫. 全连接神经网络在 FPGA 上的实现与优化[D].中国科学技术大学,2018.

[13] 张榜,来金梅.一种基于 FPGA 的卷积神经网络加速器的设计与实现[J].复旦学报(自然科学版),20
242.

[14] Chao Huang. A Layerbased Structured Design of CNN on FPGA[A]. IEEE Beijing Section、
University.Proceedings of 2017 IEEE 12th International Conference on ASIC[C].IEEE Be
Fudan University:IEEE BEIJING SECTION(跨国电气电子工程师学会北京分会),2017:4.

[15] 余奇. 基于 FPGA 的深度学习加速器设计与实现[D]. 合肥：中国科学技术大学，2016.

[16] 张培. 复杂背景下交通标志的颜色分割[D]. 武汉：武汉理工大学，2012.

[17] 方睿，刘加贺，薛志辉，杨广文.卷积神经网络的 FPGA 并行加速方案设计[J]. 计算机工程与应用，2015, 51（8）：32-36.

[18] 陆佳华，潘祖龙，彭竞宇.嵌入式系统软硬件协同设计实战指南[M].机械工业出版.社.2015.

[19] Lei Gong, Chao Wang, Xi Li, Huaping Chen, Xuehai Zhou. Work-in-Progress: A Power-Efficient and High Performance FPGA Accelerator for Convolutional Neural

Networks [J]. CODES/ISSS ’ 17 Companion, October 15 - 20, 2017.

[20] 李嘉辉,蔡述庭,陈学松,熊晓明.基于 FPGA 的卷积神经网络的实现[J].自动化与信息工程, 2018,39(01):32-37.

[21] 姜典坤. 基于异构处理器的深度卷积神经网络加速系统设计与实现[D].北京交通大学,2018.

[22] 鲍贤亮. 一种高性能 CNN 专用卷积加速器的设计与实现[D].南京大学,2018.

评审小组

姓 名	职 称	职务	工 作 单 位

导师评语

导师：

日期：　　年　　月　　日

阶段报告小组意见：

负责人：

日期：　　年　　月　　日

学院意见：

负责人：

日期：　　年　　月　　日　（签章）