第二届

全国大学生集成电路创新创业大赛

NCICC

项目设计报告

参赛题目：基于 PGT180H 的人工智能识别系统

队伍编码：257

团队名称：天枢

目 录

(1) 系统设计概述

(1.1) 设计目的

根据比赛要求，将基于紫光同创的 Titan 系列 FPGA——PGT180H，设计一个用于智能识别某类物体的数字系统，采用的算法是当下十分火热的深度学习技术——卷积神经网络。

(1.2) 设计要求

1. 识别采用 FPGA 实现深度学习算法;
2. 识别对象可包含但不限于人脸或虹膜、车牌等；
3. 识别系统应支持动态识别，如动态识别人脸，具备基于可见光环境下能实时获得人脸信息是否匹配等功能
4. 识别速率不低于 15FPS，识别准确率不低于 95%。

(1.3) 设计概述

经过我们的协商和讨论以及查阅相关资料之后，我们发现人工智能识别下的人脸识别、车牌识别等等已经较为成熟，基于实用性以及创新性，我们确定了本系统的主要任务是根据输入的人脸图片，经由系统处理后，识别该图片中人脸做出的表情，并将分类结果输出，即表情识别。

我们选择的数据集来自日本女性面部表情（JAFFE）数据库，该数据库包含由10 名日本女性模特组成的 7 种面部表情（6 个基本面部表情+ 1 个中性??）的 213 幅图像。每个图像被 60 个日语科目评为 6 个情感形容词。数据库由 Michael Lyons，Miyuki Kamachi 和 Jiro Gyoba 计划和组装。这些照片是在九州大学心理学系拍摄的。

其部分图片如图 1 所示。

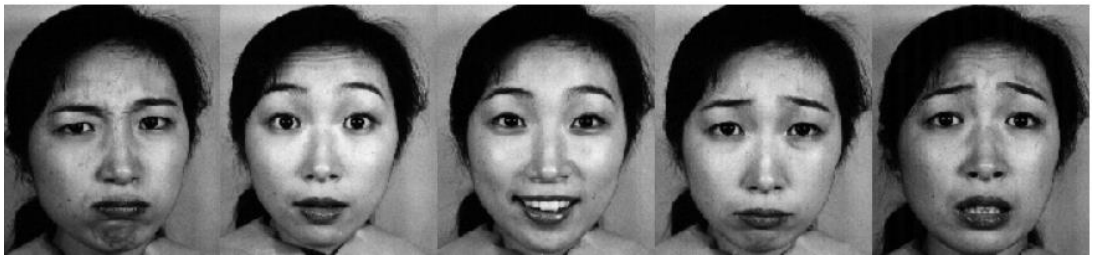

图 1 **表情图片实例**

数据集的图片是64 × 64 像素的 8bit(256 灰度等级)图像，送入模型之前对图片重新调整大小(resize)为32 × 32的大小以减小模型训练负担，并对灰度做归一化(normalize)处理，使得灰度范围在[−1,1]之内。

算法上，我们选用了目前在计算机视觉领域常用的一种卷积神经网络结构——“VGG-net”。VGG-Net 由牛津大学的视觉几何组（Visual Geometry Group）提出，是 ILSVRC-2014 中定位任务第一名和分类任务第二名。其突出贡献在于证明使用很小的卷积核（3x3），增加网络深度可以有效提升模型的效果，而且 VGG-Net 对其他数据集具有很好的泛化能力。

我们对 VGG-net 的结构做了些许改动，以在我们所选择的数据集上表现良好的同时将模型尽量压缩，最后形成的网络结构如表 1 所示。

表 1 卷积神经网络结构

Number	Layer	Context
1	卷积层	3×3, **输入特征数:1, 输出特征数:64**

2	采样层	2×2
3	卷积层	3×3, **输入特征数:64, 输出特征数:64**
4	采样层	2×2
5	卷积层	3×3, **输入特征数:64, 输出特征数:128**
6	采样层	2×2
7	卷积层	3×3, **输入特征数:128, 输出特征数:256**
8	全连接层	**神经元数:7**

我们使用了当前最流行的深度学习框架 TensorFlow 来搭建和训练此模型。卷积层卷积后不加激活函数，步长为 1；采样层步长为 2。全连接层输出接激活函数 $ReLU(x) = max\{0, x\}$，两层卷积层之间加入 dropout 层来避免过拟合。

经过上述配置，训练结果为：模型在训练集（总共 200 张图片）上准确率为 80%，在测试集（13 张图片）上，错误率为 15%，准确率随步长变化如图 2 所示。

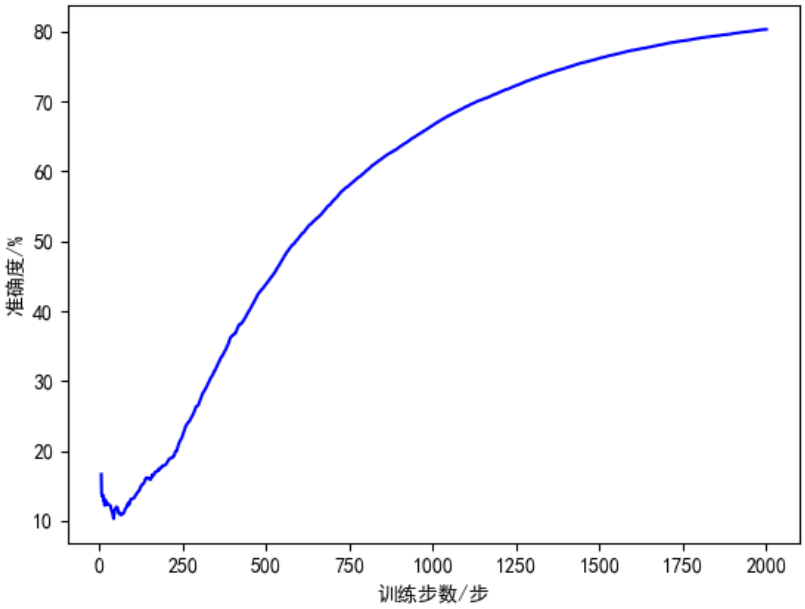

图 2 训练时准确率变化

(2) 系统架构设计

(2.1) 顶层结构

设计的智能识别系统的顶层模块结构如图 3 所示。支持的具体流程是首先在计算机上训练好算法模型，在运算时将模型的数据加载到 RAM 上。而图片则在计算机上准备好，经由串口将图像数据传送至 FPGA 开发板的高速串口上，并做格式转换和处理后送至输入缓冲中，准备输入。

输入缓冲确定输入有效后，计算模块读入图像数据和神经网络数据，依次处理，内部各个计算单元以及 RAM 由控制模块监视处理，比如计算数据的地址，控制计算部件的运行等等，故而控制模块是控制系统运行的核心部分。

接下来将细述各个模块的设计思路。

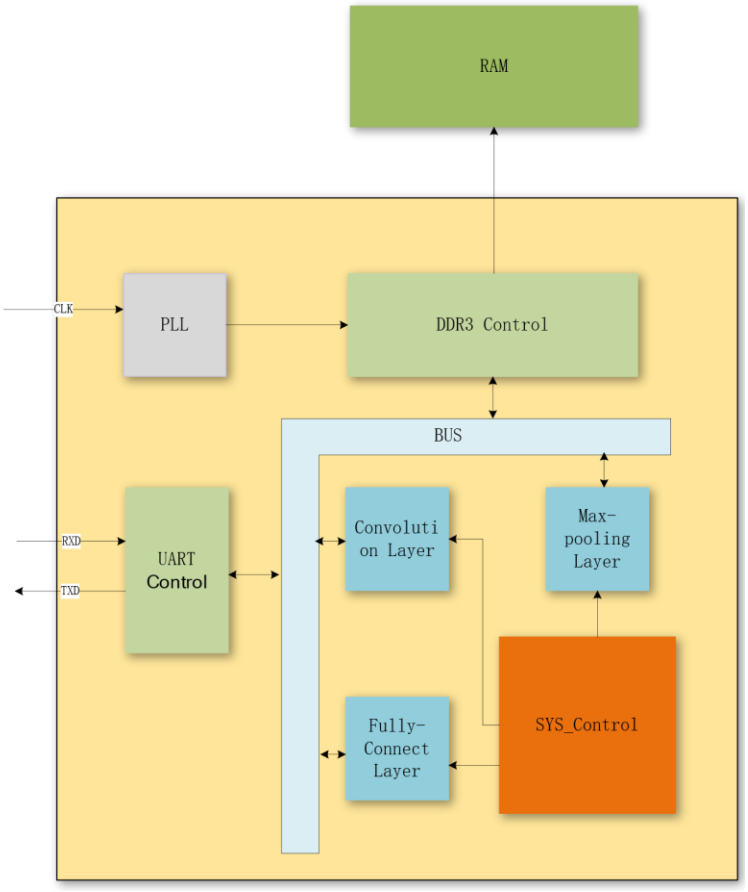

图 3 顶层结构设计

(2.2) 计算模块

根据我们的 VGG-net 网络结构，我们的计算模块主要分为以下四个分运算模块，分别是：

1) 卷积层模块
2) 采样层模块
3) 全连接层模块
4) 激活函数模块

(2.2.1) 卷积层模块 conv

(2.2.1.1) 模块结构设计

根据图像卷积运算的性质，我们发现其有大量的可以并行计算的特性，可以在 FPGA 上得到充分的优化。一般而言，并行优化卷积运算的方式有两种：一是粗粒度优化，二是细粒度优化。细粒度优化指的是单个卷积核上，一个卷积窗口（3x3）中所有数据的乘累加之间的并行计算，与此同时，不同卷积窗口之间同样可以并行，好处是可以利用数据重用从而减小 RAM 访问带宽，但需要较复杂的控制逻辑；粗粒度优化指的是不同输入特征图上，同一个位置可以并行计算，与此同时，对不同的卷积核，同样可以并行计算该位置的输出，好处是控制逻辑简单，代价是存储器带宽要求较大。

为充分利用 FPGA 片上资源，我们拟采用粗粒度并行计算加上卷积窗口内部并行计算的方式处理卷积运算。以一个大小为2×2，输入特征数为 2，有两个卷积核的最小卷积运算作为 Testbench。

卷积层最小顶层模块结构如图 4 所示，其端口定义如表 2 所示。

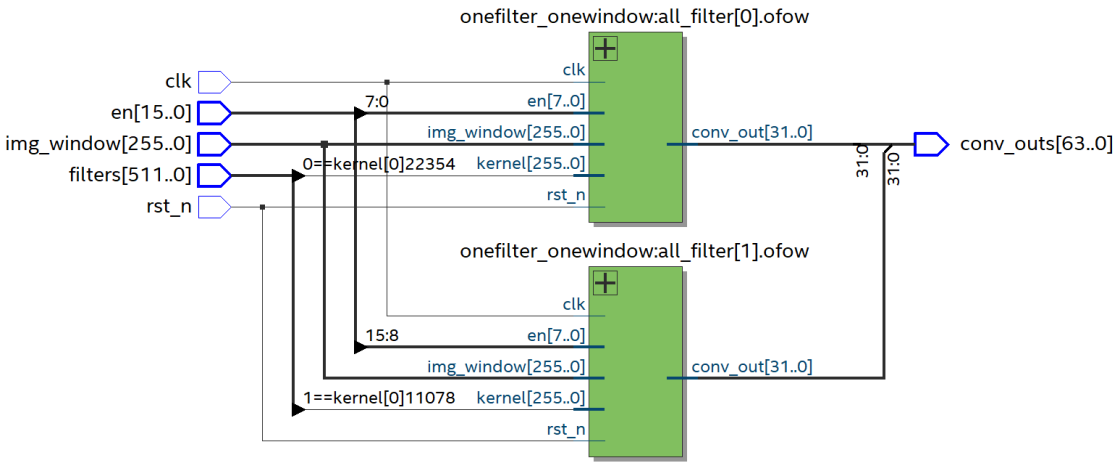

图 4 卷积层的最小顶层结构

表 2 卷积层模块端口列表

端口定义	端口说明	方向	位宽
clk	时钟信号端口	Input	1
en	使能信号端口	Input	1
img_window	采集到的图像信号的一个卷积窗口的输入	Input	Window_size*channel_size*32
filters	卷积和输入端口	Input	Filter_size*window_size*channel_size*32
rst_n	复位信号端口，低电平有效	Input	1
conv_outs	对一个卷积窗口和所有卷积核做卷积运算输出结果	Output	32

图 5 给出了卷积层子模块 onefilter_onewindow 内部电路结构，其中两个卷积核是并行计算的，端口情况如表 3 所示。我们选择的数据格式是 32 位浮点数，即每一个像素、权值是 32 位的数据，这里端口 img_window 是 256 位的输入端，代表输入的图像含有 8 个像素（$2\times2\times2\times32=256$），端口 filters 是 512（$256\times2=512$）位输入端，代表了两个与输入图像相同大小的卷积核。

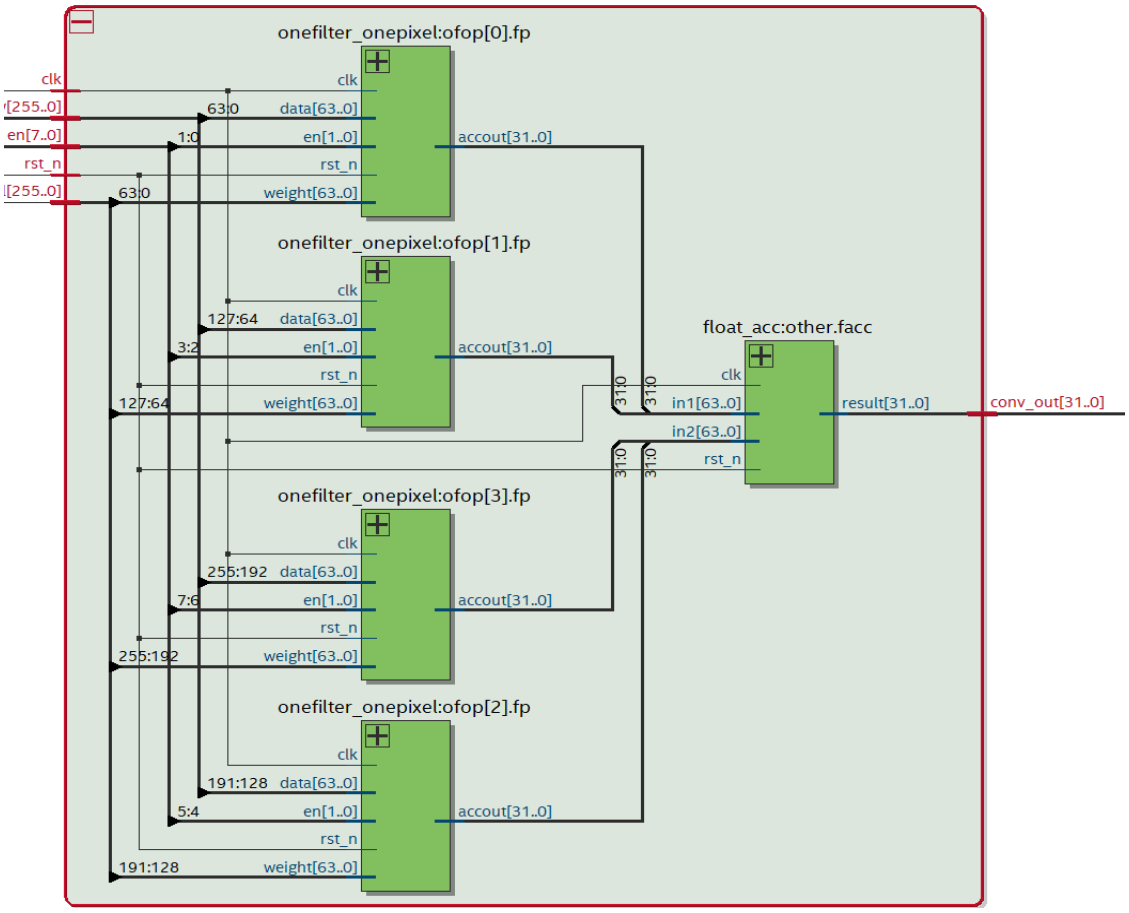

图 5 卷积层子模块 onefilter_onewindow 内部电路结构

表 3 onefilter_onewindow 端口列表

端口定义	端口说明	方向	位宽
clk	时钟信号输入端口	Input	1
data	采集到的图像信号的一个卷积窗口的	Input	Window_size* channel_size*

	输入		32
Kernel	卷积核输入	Input	Window_size* channel_size* 32
en	使能信号输入端口	Input	1
rst_n	复位信号输入端口	Input	1
conv_out	对一个卷积窗口和一个卷积核做一次卷积所得结果输出	Output	32

表 3 中，window_size 参数代表卷积窗口的大小，channel_size 参数代表被卷积的图片的总通道（或特征）数。

Onefilter_onewindow 模块同时计算一个卷积窗口（2×2 大小的像素区域）与一个卷积核的卷积。在这个最小例子中，一个窗口有 4 个像素，而每个像素包含 2 个特征（彩色图像通常是 RGB 三个特征或灰度图像一个特征），卷积核同样大小。在卷积粗粒度优化算法中，每个像素的计算是独立的，因此该模块又可拆分为 4 个子模块，最后将所有的子模块输出做累加得到卷积和。每一个子模块命名为 onefilter_onepixel，其内部结构如图 6 所示，模块端口如表 4 所示。

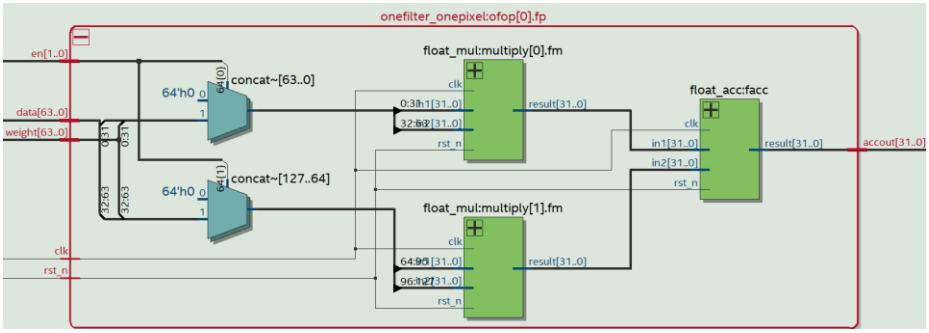

图 6 onefilter_onepixel 模块内部结构

表 4 同一位置所有特征通道并行计算端口列表

端口定义	端口说明	方向	位宽
en	使能信号输入端口	Input	1
data	像素点输入端口	Input	Channel_size*32
weight	权值输入端口	Input	Channel_size*32
clk	时钟信号输入端口	Input	1
rst_n	复位信号输入端口	Input	1
account	一个像素点所有特征的计算结果输出	Output	32

图 6 中，输入像素点和输入的权值都是 64 位宽，代表同时有 2 个特征进入模块参与运算。模块内部包含有 2 个乘法器和一个加法器，乘法器之前两个选择器作为使能信号的控制器。

(2.2.1.2) 功能验证

我们在 modelsim 中使用了 3x3，8 通道，8 个卷积核的 testbench，验证结果如下图：

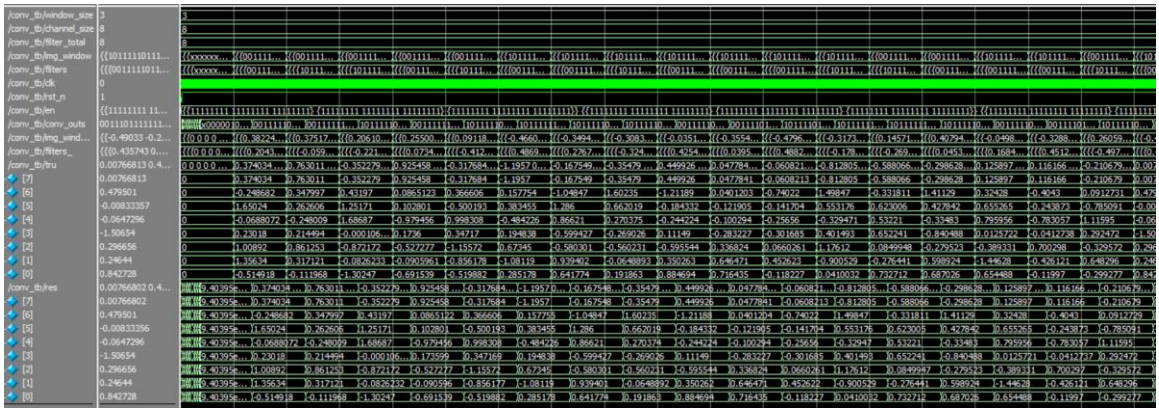

图 7 卷积层仿真截图

图 7 中，输入信号 img_window 代表输入的图像数据，输入信号 filters 代表输入的卷积核。Clk 和 rst_n 代表时钟和复位信号，en 代表输入有效信号，输出信号 conv_outs 代表卷积结果。下方的 img_window_和 filters_分别是相应输入数据的浮点数版本。数据 tru 代表卷积结果的真实值，与输入数据输入的同时输出，而 res 则是 conv_outs 的浮点数版本。

每个输入信号都保持 1000 个仿真时间单位。经过大概 300 个时间单位左右后，模块输出计算结果。从图中我们可以看到，tru 信号组与 res 信号组做对比，二者对应位置的信号是相同的，表明了模块的功能是正确的。

(2.2.2) 采样层模块 pooling

(2.2.2.1) 模块设计

采样层将紧跟在卷积层输出之后，对同一个输出一个2×2窗口每个特征图进行最大采样，取出窗口 4 个像素点在某个特征图上最大值。由于步长为 2，故而卷积窗口之间没有重叠部分。 该模块包含三个比较器，如图 8 所示。

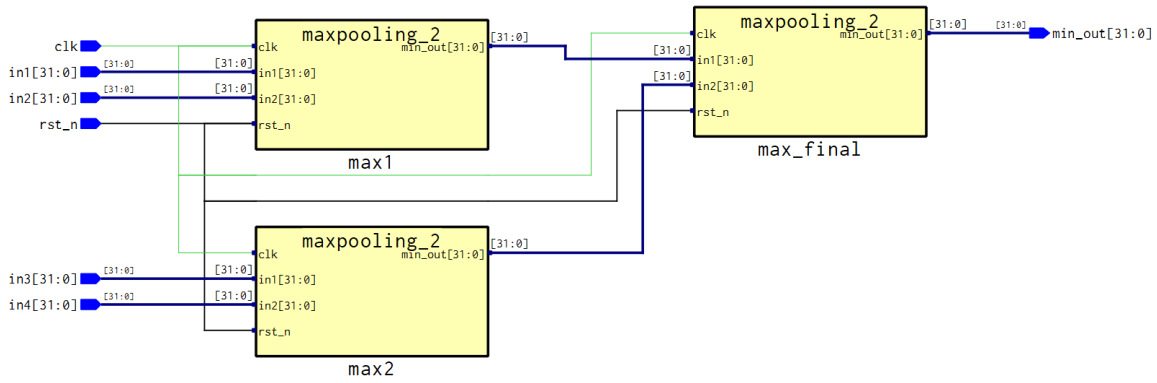

图 8 采样层模块电路结构

表 5 maxpooling 层端口列表

端口定义	端口说明	方向	位宽
Clk	时钟	Input	1
Rst_n	低电平复位端	Input	1
In1，in2，in3，in4	输入数据	Input	32
Max_out	输出数据	Output	32

两两比较取大者，再比较一次即得到输出。

(2.2.2.2) 功能验证

仿真截图如图所示：

图 9 仿真结果

采样层 ing 模块实现功能为输入四个 32 位标准格式浮点数，输出其中最大值。图为该模块的 modelsim 仿真截图，可以看出所得结果正确，即功能成功实现。在输入新的数据之后第五个时钟周期以后可得到结果，由此可得可以每五个时钟周期经行一次数据的采集。以上仿真结果图中，第 1 行信号为时钟信号输入，2—5 行信号为输入四个不同浮点数，共输入三组信号，第 6 行信号为复位信号，第 7 行为输出信号，即四个数中最大值。

(2.2.3) 全连接层模块 fully-connect

(2.2.3.1) 模块设计

这里做的实际上是矩阵乘加法计算，预先存储的权值矩阵与输入的向量做矩阵乘法，并与偏置向量做加法。最后输出至激活函数模块。因此，该模块包含两个子模块，一个是矩阵乘法模块 matmul，另一个是矩阵加法模块 matadd。

这里，我们假设输入的向量是一个 10 维向量，每个分量是一个 32 位浮点数，输入到 320 位宽的端口 data；权值矩阵是一个 10x4 的矩阵，输入到 1280 位宽的端口 weight；偏置向量是 4 维向量，输入到 128 位宽的端口 bias。

模块大致结构如图：

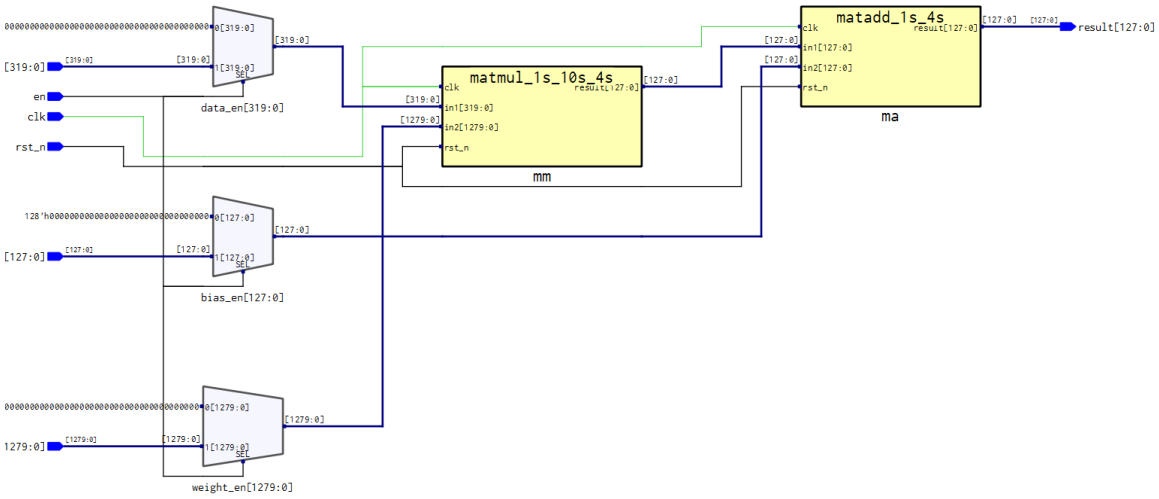

图 10 结构框图

表 6 权值矩阵与偏置向量计算模块端口示意表

端口定义	端口说明	方向	位宽
Data	data 数据输入端口	Input	Batch_size*feature_size*32
en	使能信号输入端口	Input	1
clk	时钟信号输入端口	Input	1
rst_n	复位信号输入端口	Input	1
Bias	偏置值输入端口	Input	Bias_size*32
Weight	权值输入端口	Input	Feature_size*bias_size*32
result	模块计算结果输出端口	Output	Batch_size*bias*size*32

其中子模块 matmul 的内部结构实现如下：

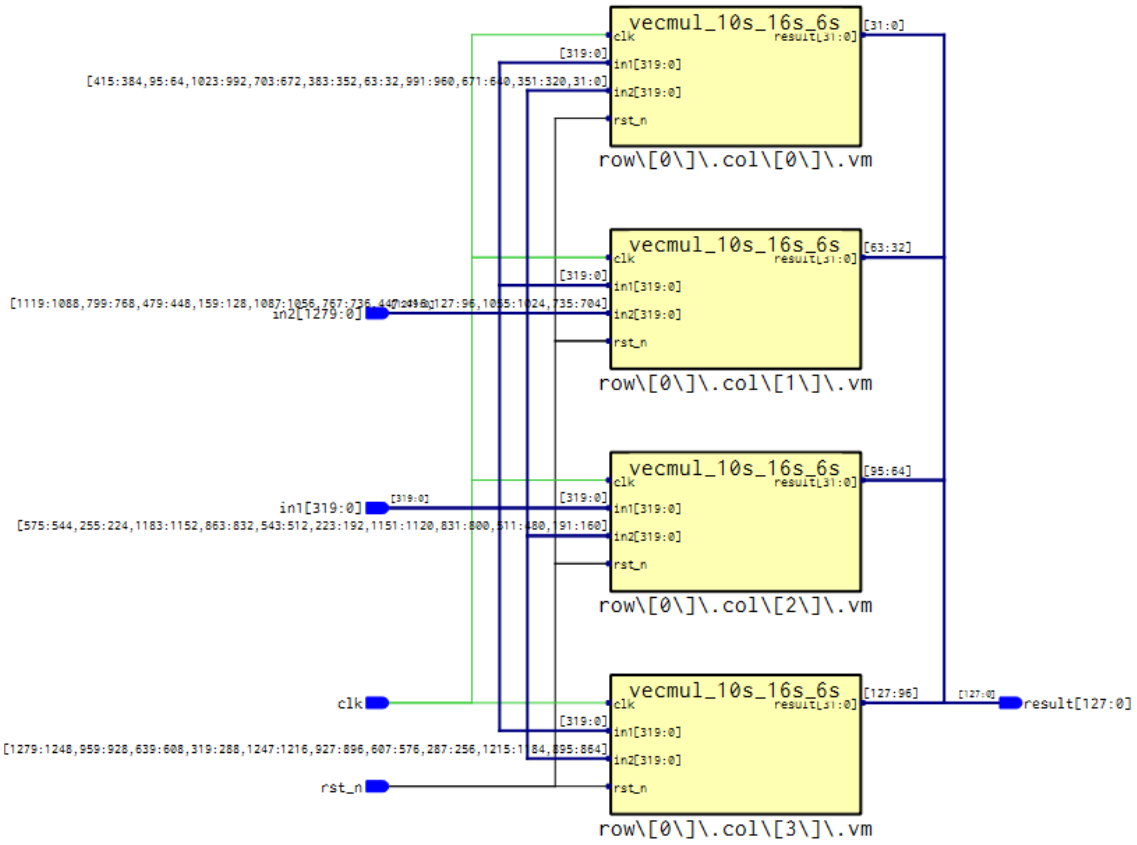

图 11 矩阵乘法模块 matmul 内部结构

在这个例子中，Matmul 模块将矩阵乘法分解为 4 个计算向量内积的子模块 vecmul，每个子模块输出（result）32 位浮点数，最后将其合并为一个 128 位信号做为输出矩阵（向量）。端口 in1 和 in2 分别是两个矩阵的输入端，clk 和 rst_n 分别是上升沿有效时钟信号和低电平有效复位信号端。图 12 中的子模块 vecmul 内部结构如下：

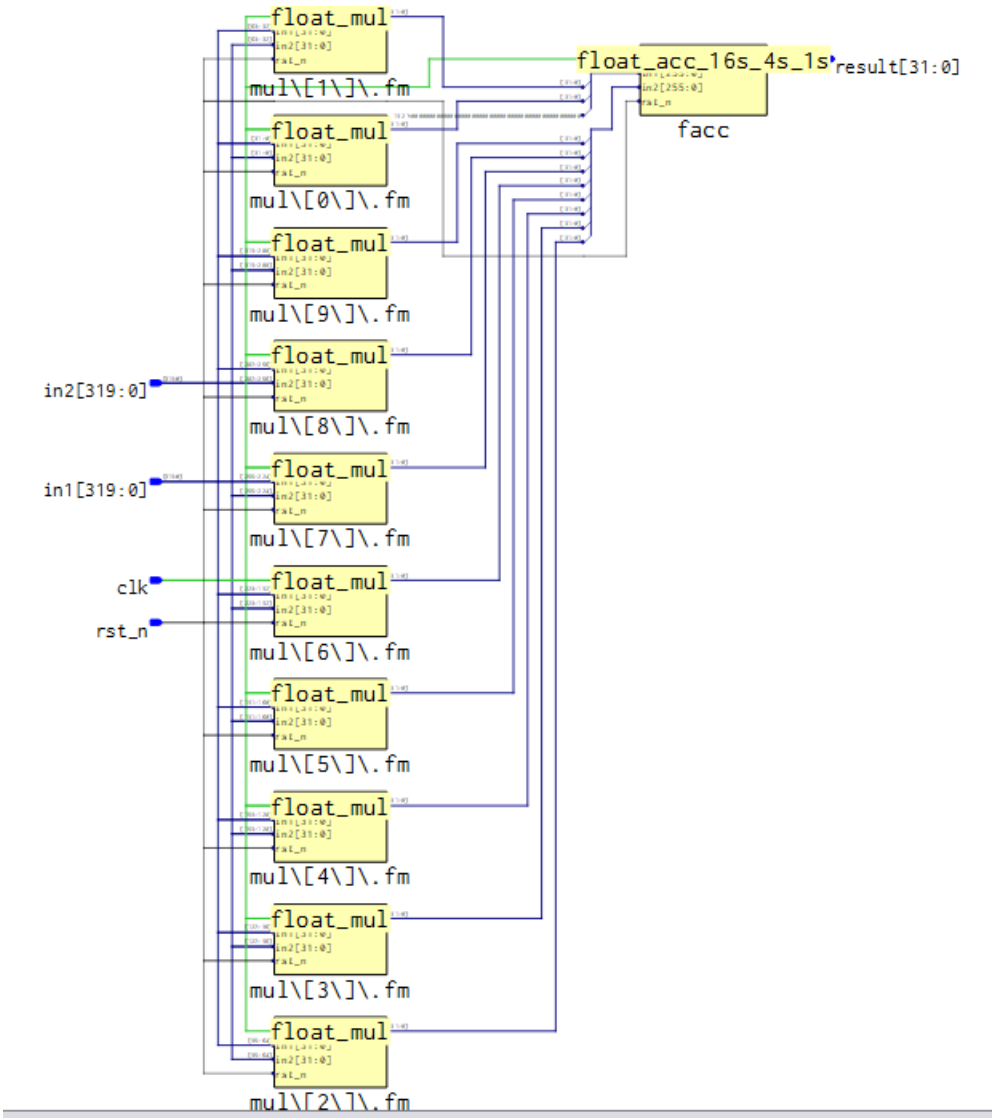

图 12 向量内积模块 vecmul

两个输入向量 in1 和 in2 的每个分量都送入浮点数乘法模块 float_mul 中，输出连到累加模块 float_acc 中，计算得到输出结果。

矩阵加法模块 matadd 结构大致如图 14，对 4 维向量的输入，将每个相同位置的元素相加得到输出向量，故而内部使用了 4 个加法器模块 float_add，输出再合并拼接为 128 位的向量 输出。

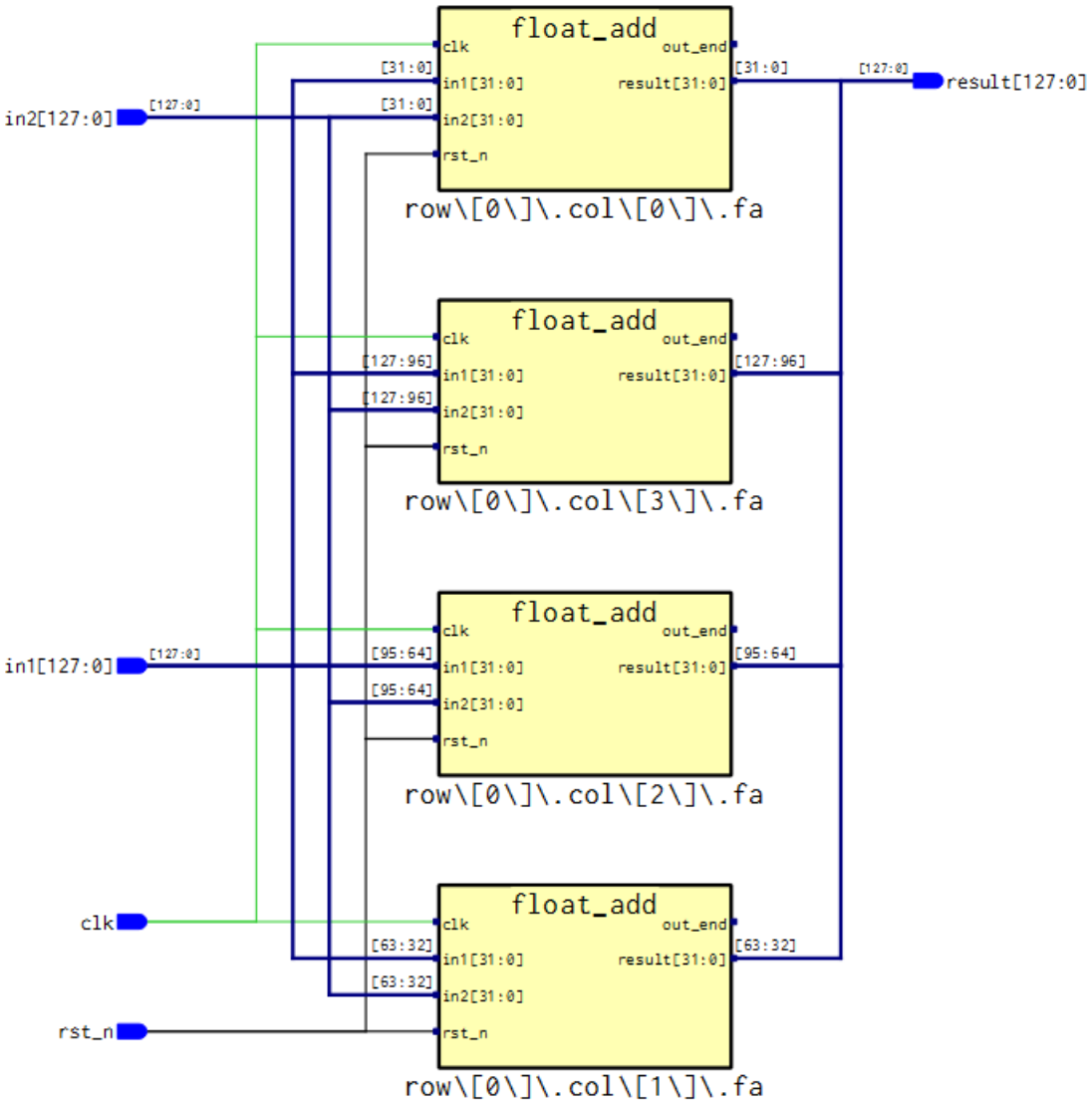

图 13 矩阵加法模块 matadd 结构图

(2.2.4) 激活函数模块 acti_fn

(2.2.4.1) 模块设计

激活函数使用的是常见的 ReLU 函数，表达式$f(x) = max(0, x)$。

我们设置 x 为一个 3 维向量，总共 96 位宽信号输入，在模块内部分成三部分分别计算。

具体实现如下：

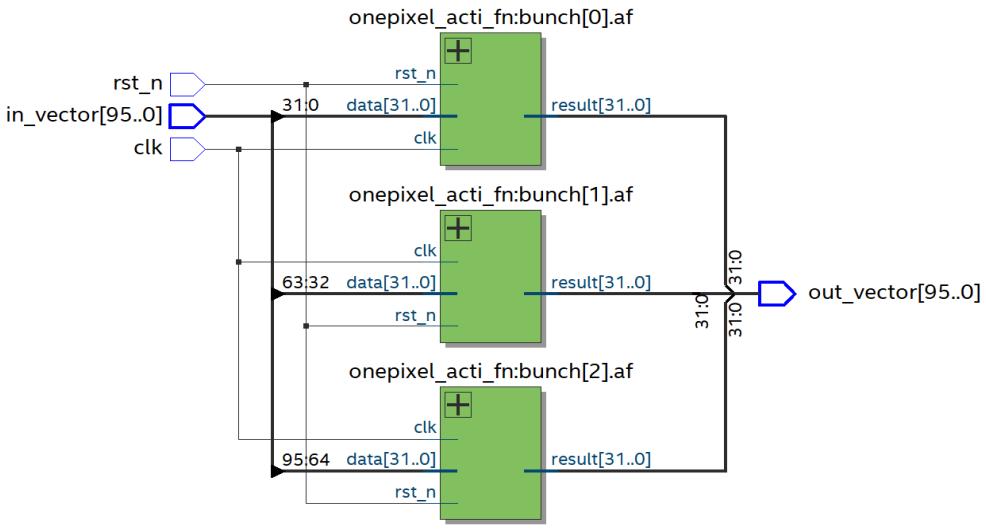

图 14 元素输入向量的激活函数模块示例

表 7 激活函数模块端口示意表

端口定义	端口说明	方向	位宽
rst_n	复位信号输入端口	Input	1
in_vector	输入向量端口	Input	Ndim*32
clk	时钟信号输入端口	Input	1
out_vector	输出向量端口	Output	Ndim*32

(2.2.4.2) 功能验证

正如上图，输入的测试向量是一个 3 元素向量，输入$(1.5, -1.5, 0.05)$：

仿真输出如图：

图 15 仿真结果

从图中我们可以看到，向量的第一个元素（00111111110000000000000000000000）和第三个元素（00111101010011001100110011001101）原样输出，而第二个元素（10111111110000000000000000000000）小于零而归零。满足了激活函数的要求。

(2.3) 其他模块

其余各个模块主要负责：1）与总线的桥接 2）控制计算模块的运作 3）与外部计算机交换数据。

(2.3.1) 总控模块

总控模块主要负责各个运算模块随着计算的进行的开启和关闭。其运转过程可有一个状态机如图 16 做描述。

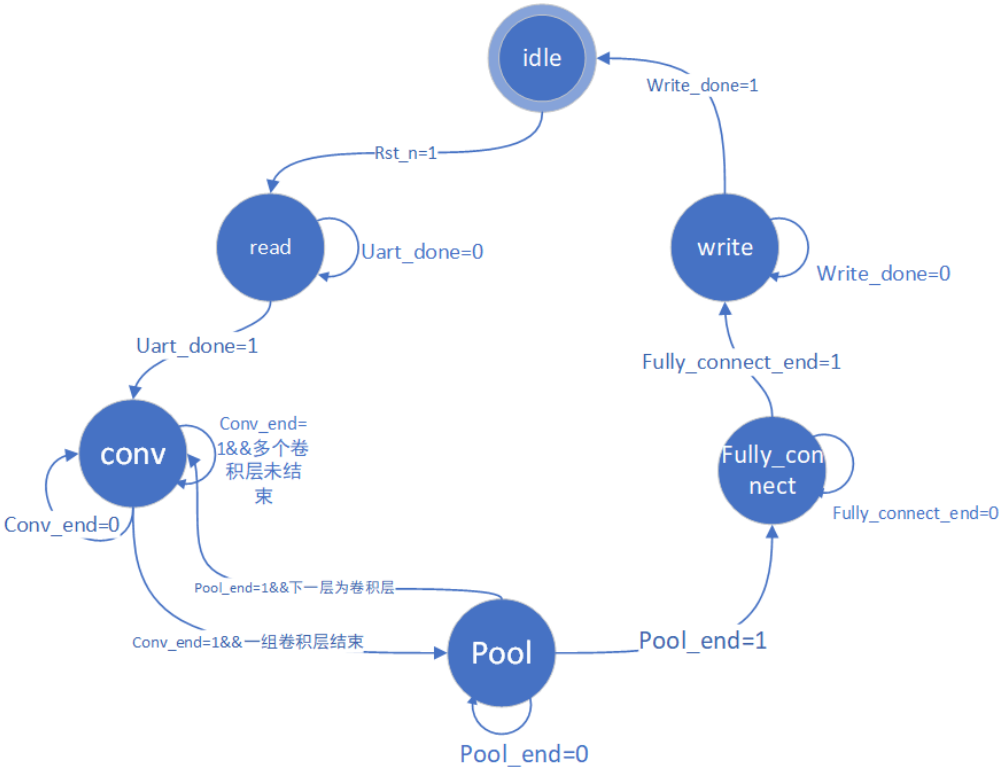

图 16 系统控制状态机

端口定义	端口说明	方向	位宽
Clk	时钟	Input	1
Rst_n	复位	Input	1
System_end	系统停止信号	Output	1
Conv_en	卷积层模块启动信号	Output	1
Conv_done	卷积运算结束信号	Input	1
Conv_link_read	卷积模块总线读开关	Output	1
Conv_link_write	卷积模块总线写开关	Output	1
Conv_init_addr	卷积层对应卷积核组首地址	Output	28
Conv_init_addr_en	地址使能	Output	1
Pool_en	采样启动信号	output	1
Pool_done	采样结束信号	Input	1
Pool_link_write	采样模块总线写开关	Output	1
Pool_link_read	采样模块总线读开关	Output	1
Fc_en	全连接层启动	Output	1

Fc_done	全连接层计算结束信号	Input	1
Fc_link_write	全连接层总线写开关	Output	1
Fc_link_read	全连接层总线读开关	Output	1
Uart_en	串口启动	Output	1
Uart_done	串口传输结束信号	Input	1
Uart_link_read	串口总线读开关	Output	1
Uart_link_write	串口总线写开关	Output	1

(2.3.2) 卷积层控制接口

卷积层计算模块及其控制模块封装为一个卷积单元，通过地址总线和数据总线与内存交换数据，并受到系统状态控制器的控制。其外部接口模型如下：

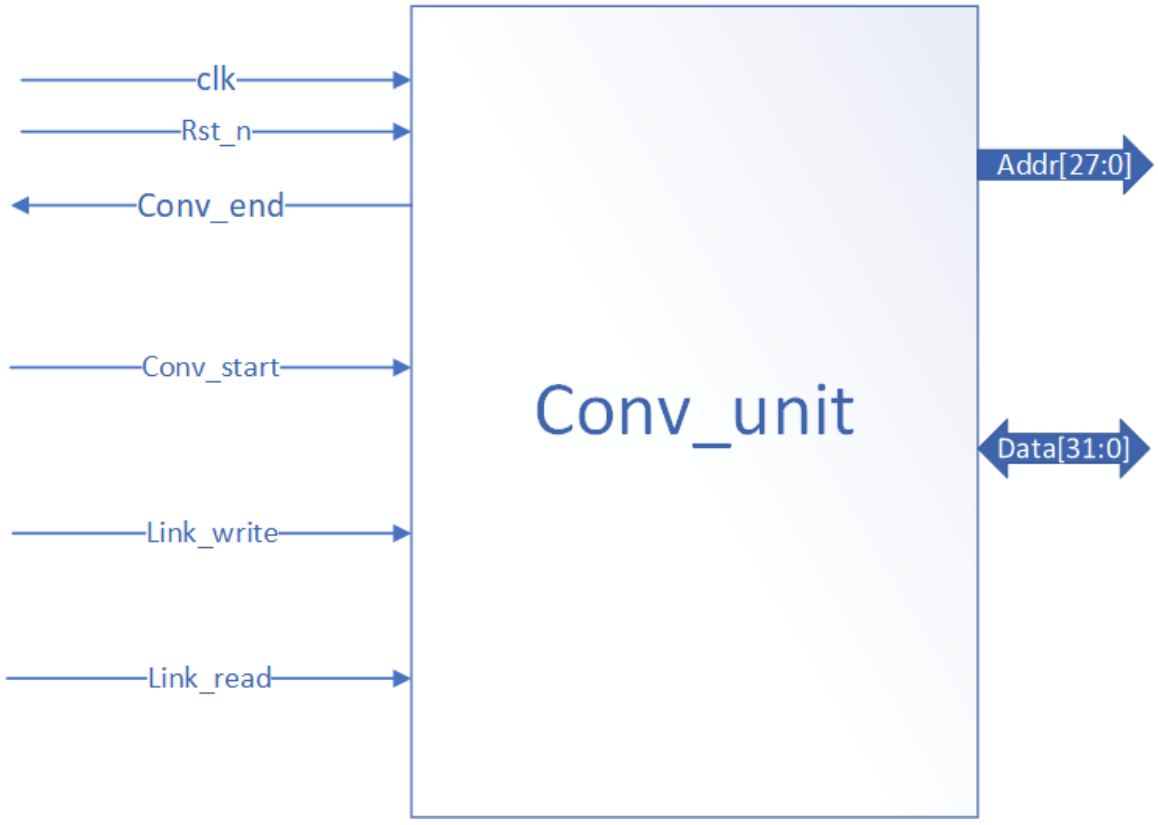

图 17 卷积单元外部接口模型

其中，clk 与 rst_n 信号有系统提供，conv_start 由状态控制器提供，指明卷积运算的开启，conv_end 在卷积运算结束后提供，指明卷积运算的结束。Link 信号控制总线的开关。

其内部模块连接图（由综合工具生成的 RTL 结构图）如图 17 所示。

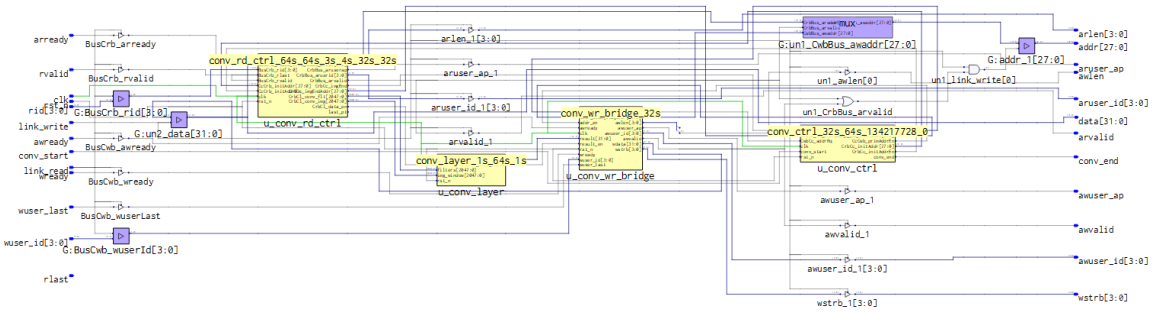

图 18 内部链接图

其中各个模块详细说明见下方。

(2.3.2.1) 卷积层读控制模块

卷积层读取控制负责从内存中读取需要的数据，并把数据放入计算模块中。该模块实现了自动计算卷积窗口位置，自动计算当前数据在内存中的地址，并检测当前输入图片的计算的状态，由此控制卷积层的运行。

其外部接口如下所示：

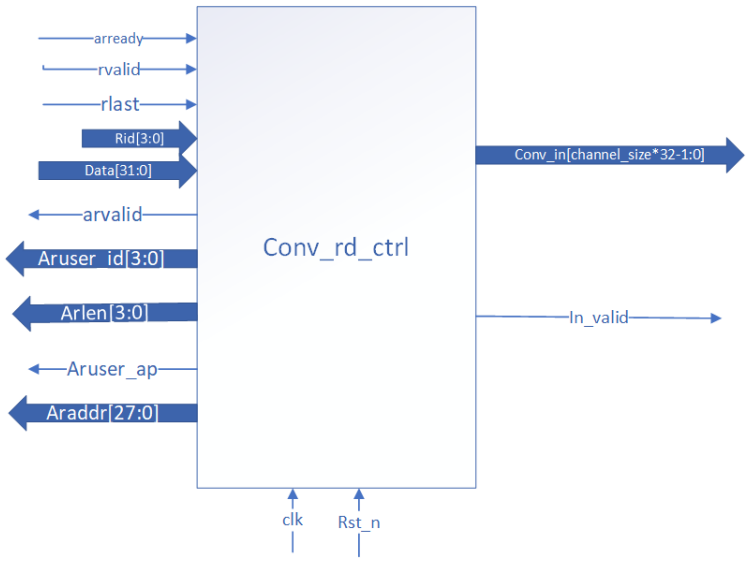

图 19 conv_rd_ctrl 接口示意

表 8　conv_rd_ctrl 端口列表

端口定义	端口说明	方向	位宽
Aready	写入读地址完成	Input	1
Rvalid	读数据有效	Input	1
Rlast	数据突发传输结束	Input	1
Rid	读数据批次的 tag	Input	4
Data	总线输入数据	Input	32

Arvalid	写入读地址有效	Output	1
Aruser_id	标记要读取的数据批次的 tag	Output	4
Arlen	突发传输数据的长度	Output	4
Aruser_ap	自动预充电信号，高有效	Output	1
Araddr	数据地址	Output	28
Conv_in	输出的包含多通道的数据	Output	Channel_size*32
In_valid	输出数据有效	Output	1
Clk	时钟信号	Input	1
Rst_n	低电平有效复位信号	Input	1

(2.3.2.2) 卷积层写控制模块

卷积层写控制模块负责将计算结果写入内存，以供下一次计算使用。

其外围接口模型如下图所示：

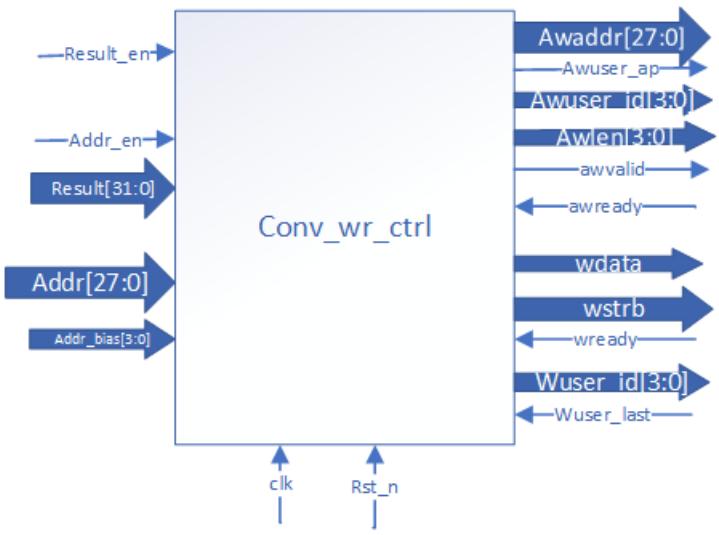

图 20 conv_wr_ctrl 模块接口

表 9 conv_wr_ctrl 模块端口列表

端口定义	端口说明	方向	位宽
Clk	时钟	Input	1
Rst_n	复位	Input	1
Result	卷积计算结果	Input	32
Addr	卷积窗口首元素的内存地址，也是卷积结果存放地址	Input	28
Addr_bias	地址偏移量，表达该结果是第 addr_bias 个 filter 卷积得到	Input	6
Result_en	结果输入有效	Input	1

Addr_en	地址输入有效	Input	1
Awaddr	输出至总线的读地址	Output	28
Awuser_ap	预充电	Output	1
Awuser_id	写数据 id	Output	4
Awlen	写数据突发传输长度	Output	4
Awvalid	输出地址有效	Output	1
Awready	读地址完成	Input	1
Wdata	输出数据	Output	32
Wuser_id	写数据选通位	Input	4
Wuser_last	突发传输最后一位数据写结束标志	Input	1
Wready	写数据开始	Input	1

(2.3.2.3) 读写调度模块

读写调度模块主要用于控制整个卷积运算的运算流程，包括控制读模块何时读取数据以及写模块何时写数据至内存，最后可以停止和启动整个卷积模块。

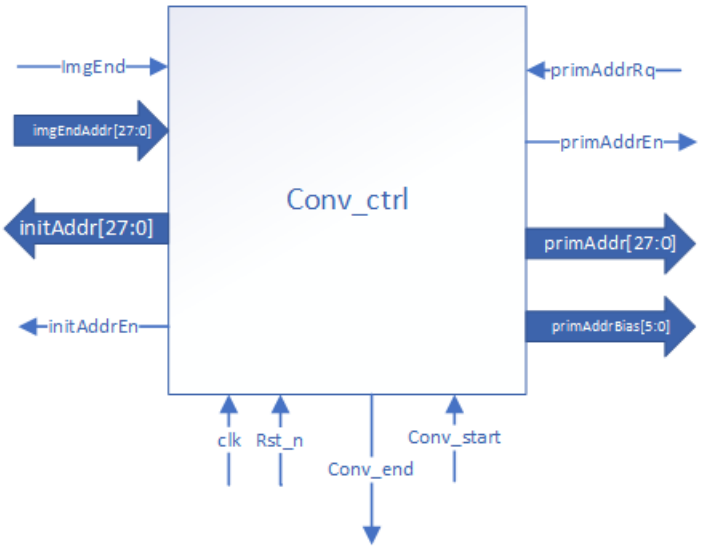

图 21 读写调度模块

表 10 conv_ctrl 端口列表

端口定义	端口说明	方向	位宽
ImgEnd	卷积运算至最后一点的标志信号	Input	1
ImgEndAddr	最后一点像素所在地址	Input	28
initAddr	卷积运算所需数据的初始地址	Output	28
InitAddrEn	地址有效	Output	1
Conv_start	卷积运算开始信号	Input	1
Conv_end	卷积运算结束信号	Output	1
PrimAddrRq	卷积窗口首像素点地址获取请求	Input	1
Primaddr		Output	28

PrimAddrEn	地址有效	Output	1
PrimAddrBias	地址偏置	Output	6

(2.3.3) 采样层控制接口

采样层的控制逻辑与卷积层的控制逻辑极其类似，只是在某些细节上有一些差别，因此二者的控制接口基本相同。

(2.3.3.1) 采样层读控制模块

表 11 pool_rd_ctrl 端口列表

端口定义	端口说明	方向	位宽
Aready	写入读地址完成	Input	1
Rvalid	读数据有效	Input	1
Rlast	数据突发传输结束	Input	1
Rid	读数据批次的 tag	Input	4
Data	总线输入数据	Input	32
Arvalid	写入读地址有效	Output	1
Aruser_id	标记要读取的数据批次的 tag	Output	4
Arlen	突发传输数据的长度	Output	4
Aruser_ap	自动预充电信号，高有效	Output	1
Araddr	数据地址	Output	28

In1， in2，in3，in4	采样层所需的 4 个相邻位置的多通道数据	Output	Channel_size*32
In_valid	输出数据有效	Output	1
Clk	时钟信号	Input	1
Rst_n	低电平有效复位信号	Input	1

(2.3.3.2) 采样层写控制模块

表 12 pool_wr_ctrl 端口列表

端口定义	端口说明	方向	位宽
Clk	时钟	Input	1
Rst_n	复位	Input	1
Result	最大采样输出结果	Input	32
Addr	采样层窗口首元素的内存地址，也是采样结果存放地址	Input	28
Result_en	结果输入有效	Input	1
Addr_en	地址输入有效	Input	1
Awaddr	输出至总线的读地址	Output	28
Awuser_ap	预充电	Output	1

Awuser_id	写数据 id	Output	4
Awlen	写数据突发传输长度	Output	4
Awvalid	输出地址有效	Output	1
Awready	读地址完成	Input	1
Wdata	输出数据	Output	32
Wuser_id	写数据选通位	Input	4
Wuser_last	突发传输最后一位数据写结束标志	Input	1
Wready	写数据开始	Input	1

(2.3.3.3) 采样层读写调度模块

端口定义	端口说明	方向	位宽
ImgEnd	采样结束（比较至最后一个窗口）	Input	1
ImgEndAddr	最后一点像素所在地址	Input	28
initAddr	采样层所需数据初始地址（28’b0）	Output	28
InitAddrEn	地址有效	Output	1
pool_start	采样开始信号	Input	1

pool_end	采样结束信号	Output	1
PrimAddrRq	卷积窗口首像素点地址获取请求	Input	1
Primaddr		Output	28
PrimAddrEn	地址有效	Output	1
PrimAddrBias	地址偏置（恒为0）	Output	6

(2.3.4) Uart 数据输入输出模块

负责与外界计算机交互，接受计算机输入的串行信号，转换为 32 位宽的数据写入总线，并将从总线获得的数据转换后转送至计算机。

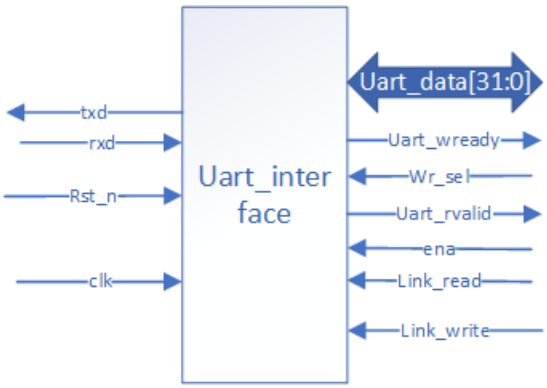

图 22 uart 模块定义

表 13 uart 接口端口列表

端口定义	端口说明	方向	位宽
Txd	串行输出	Output	1
Rxd	串行输入	Input	1

Rst_n	低电平复位信号	Input	1
Clk	时钟信号	Input	1
Uart_rvalid	串口信号读有效，高电平有效	Output	1
data	数据读写端	Inout	32
Link_write	总线写开关，高电平有效	Input	1
Ena	串口控制使能，高电平有效	Input	1
Link_read	总线读开关，高电平有效	Input	1
Wr_sel	读写选择，=0 为写，=1 为读	Input	1
Uart_wready	串口信号写有效，高电平有效	Output	1

该模块的读时序如图 22 所示。

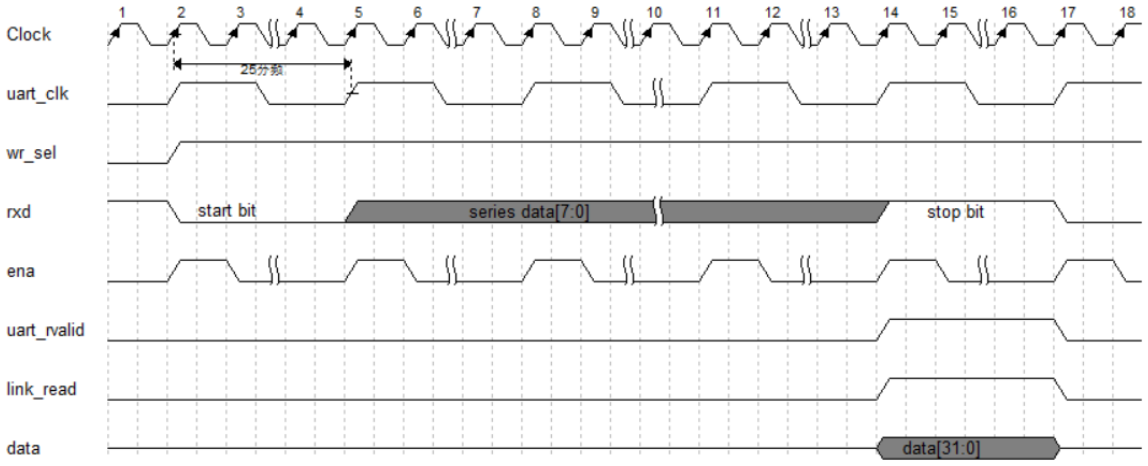

图 23 uart 读时序图

由于该串口通信使用的芯片 CP2104 的波特率最高时 2Mbps，因此要针对其速率对时钟做相应的分频。

端口 wr_sel 拉高后选择读模式，读取 rxd 输入的信号，帧格式如图，以一位启动位开始并以一位停止位结束，数据帧为 8 位连续信号。接受 4 次后，输出端口 uart_valid 拉高表明一个 32 位数据读出完成，link_read 拉高使得数据可以输出至总线。

其写时序如图 23 。

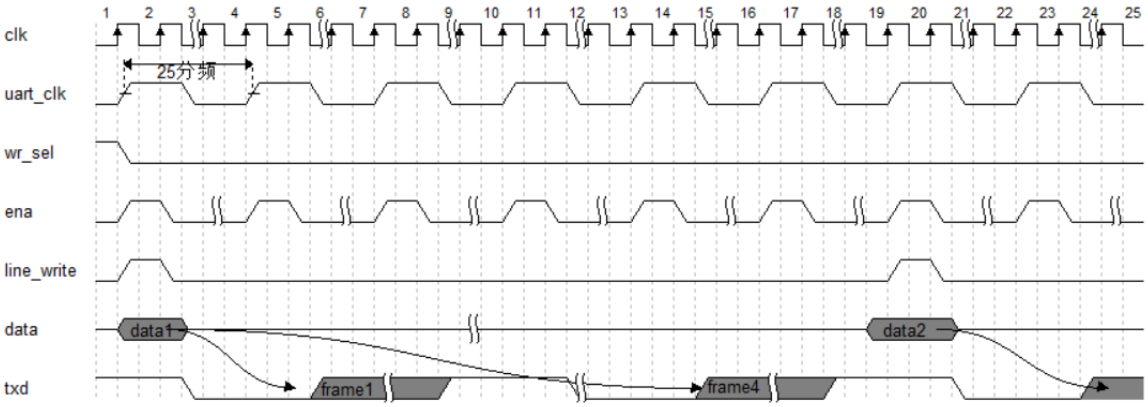

图 24 uart 写时序

端口 wr_sel 拉低后，选择写模式，数据写到总线上同时拉高 link_write 使得数据可以写进数据寄存器中，完成并转串功能并打包成合法的数据帧送到输出端 txd，送完 4 帧后输出端 uart_wready 拉高。

仿真截图如下：

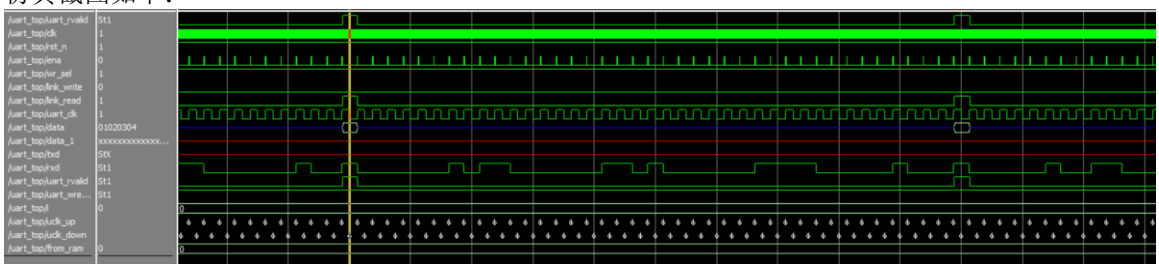

图 25 uart 读数据仿真截图

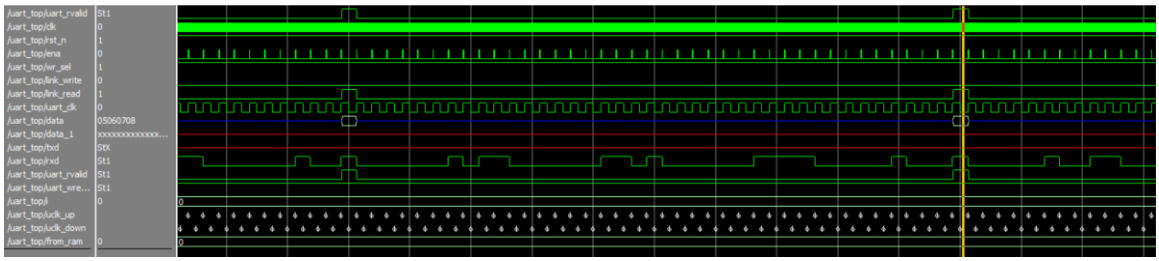

图 26 uart 读数据仿真截图 2

图 24 和图 25 中，wr_sel=1，选择读模式，link_write=0，关闭写数据通道。link_read=1，打开读数据通道后，串口数据读入，第一次读入的数据为 01020304（见图 24 左侧端口列表 data 值），第二次读入的数据为 05060708 。

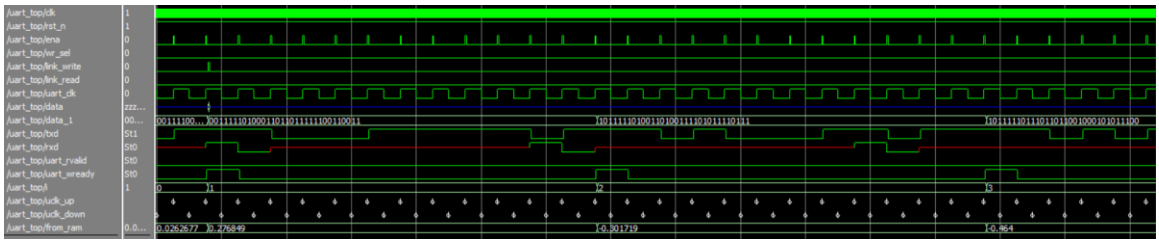

图 27 uart 写数据仿真截图

图 26 中，wr_sel=0，选择写模式，link_read=0，关闭读数据通道。Link_write=1，打开写数据通道后，串口数据转换为 32 位数据后送入总线。

(3) 项目总结

(3.1) 项目特点

(3.1.1) 计算模块的高度并行性

众所周知，深度卷积神经网络的训练与推断过程是一项计算密集型任务。目前绝大多数深度学习研究者都将神经网络模型的训练部署在多台 GPU 上，而模型的商业化应用也大多部署在分布式服务器上。但即便通过种种措施，其本质上依旧是串行结构，只不过要么是部分并行（利用 GPU 做加速矩阵计算），要么是分解计算任务（采用分布式计算），或兼有之但成本过于昂贵，普通团队或企业难以承担。

本项目充分利用了卷积神经网络本身计算的并行性——其层与层之间相互独立无数据反馈，各个卷积核独立参与运算，图片各通道之间独立计算等特点，将原本需要高性能 GPU 承担的超大型矩阵运算分解为各个并行的计算单元，使得原本在 GPU 上计算所耗费的时间大大缩短，让计算的实时性有了充分的保证。

本项目在 FPGA 上充分支持了卷积神经网络的三种并行性：各卷积核之间的并行计算，图片与卷积核的各个通道卷积运算的并行性以及同一个卷积窗口各个像素之间并行性，完成了可综合 RTL 级代码并通过了仿真（见 2.2.1.2 节）。

除此以外，我们同样实现了采样层与全连接层的并行计算，并且针对其特殊的计算结构做了并行优化（见 2.2.2 节、2.2.3 节），完成了 RTL 代码和仿真。

(3.1.2) 采用流水线结构优化吞吐量

我们利用了 FPGA 片上寄存器资源丰富的特点，在每个计算层级之间加上寄存器以缓存结果，以流水线结构处理整个计算过程，极大提高了模型的吞吐量，理论上可以一个时钟得到一个结果，具有很大的潜力。同时还降低了对时钟频率的要求，可以降低 计算功耗，也提高了计算的稳定性与健壮性。

(3.1.3) 可配置神经网络超参数

为了应对深度神经网络模型的多样性，本项目在代码层面支持了对模型结构的调整。只需调整参数来例化计算模块，便可以方便的生成所需模型的结构，非常灵

活。本项目中可变的参数包括了卷积层的个数，卷积核的个数，卷积核的大小，图片和卷积核的通道数，全连接层的神经元的个数，采样层的大小。以上皆可在实例化模块的时候调整参数即可得到，而无需将代码重构，因此可以提高模块的复用性。

(3.2) 项目展望

为了满足模块的并行性要求以及相应的流水线结构，需要较多的逻辑资源和寄存器资源，这也限制了太深的神经网络的部署，因此在更强大的、逻辑资源更丰富的 FPGA 上我们的项目会有更好的表现。

本项目也要求更大的 I/O 带宽和存储器带宽，以适应项目的大量并行计算。

同时，我们也希望有针对神经网络加速的新的计算架构出现来改进我们现有的系统结构，使得其具备优异的性能同时节省面积和功耗。

(4) 结束语

时光荏苒，我们在大三上学期跟随我们的指导老师江先阳教授昨业余科研，在上学期末得知了此次全国大学生集成电路创新创业大赛即将举办的消息，于是立刻报名参加，并选择了智能识别这个我们深感兴趣的课题。

如今半年过去，整个参与比赛的过程中，我们学习到了整个集成电路系统设计的设计方法和完整流程，走了不少弯路，也获得了不少经验，更重要的是锻炼了我们坚持不懈的意志力和一颗不惧挑战的决心。这些宝贵财富的获得离不开我们的导师江先阳教授的悉心教导，没有导师的指导，我们没有机会参与也没有能力和信心将比赛进行下去。当然还有我们团队的通力合作，我们才能完成一个庞大的系统开发任务。

最后，也感谢母校武汉大学给予我们研究的平台，以及大赛官方以及紫光同创给予的支持，感谢评委们的细心和公平的审核。谢谢！