实验 4-2 LeNet 卷积模块 (选做)

一、实验背景及整体概述

LeNet 网络是一种经典的神经网络，应用于手写数字识别时可获得优异的识别精度。本次实验所用网络结构为简化版的 LeNet 网络，具体如图 1 所示，整个网络包含卷积层 convolution，池化层 subsampling 和全连接层 full connection。

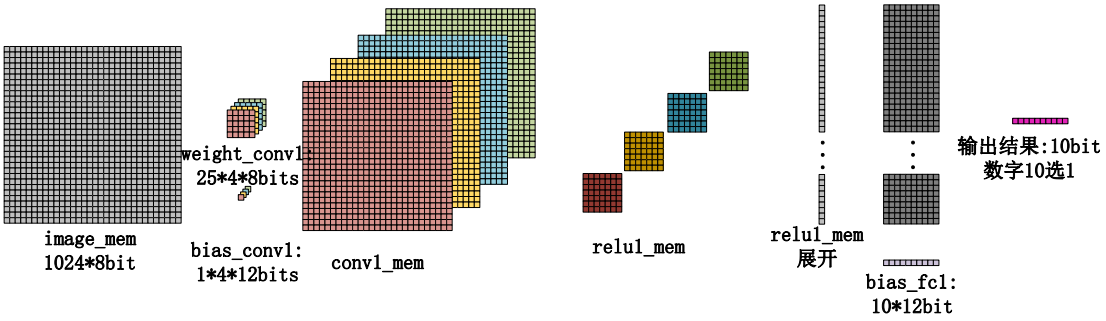

图 1 LeNet 网络结构：从左到右依次为卷积层、池化层、全连接层

上述结构在硬件加速器中实现的结构框图如图 2 所示。其中控制器控制加速器中的各个计算模块（卷积、池化、全连接），计算中所有涉及到的数据均存储在 BRAM (Block Random Access Memory，是 FPGA 中特有的内存结构)。当执行运算时，会首先从存储器中读取输入数据，执行计算，最后将计算结果写回存储器中。

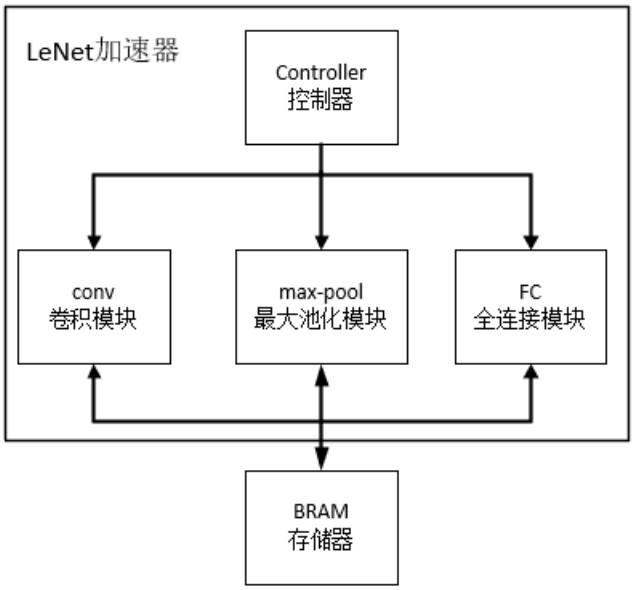

图 2 LeNet 加速器整体结构

在 LeNet 中，卷积层的计算量占比最大，因此是电路设计的重点。卷积层的计算方式如

图 3 所示，卷积核与输入特征图的一个滑动窗做乘累加得到输出特征图的一个点。以图中所示计算为例，该卷积计算的卷积核大小为 3*3，在第一次计算时，执行计算为$1\times2+2\times0+3\times1+0\times0+1\times1+2\times2+3\times1+0\times0+1\times2=15$; 在第二次计算时，卷积窗口向右滑动，同理完成计算；第三次计算时，由于输入特征图的前三行已经扫完，卷积核下移，执行计算（为方便说明，在图中仅采用了 4*4 的输入特征图，在实际中特征图的尺寸会更大）；以此类推，卷积核会滑过整张输入特征图，得到输出结果。

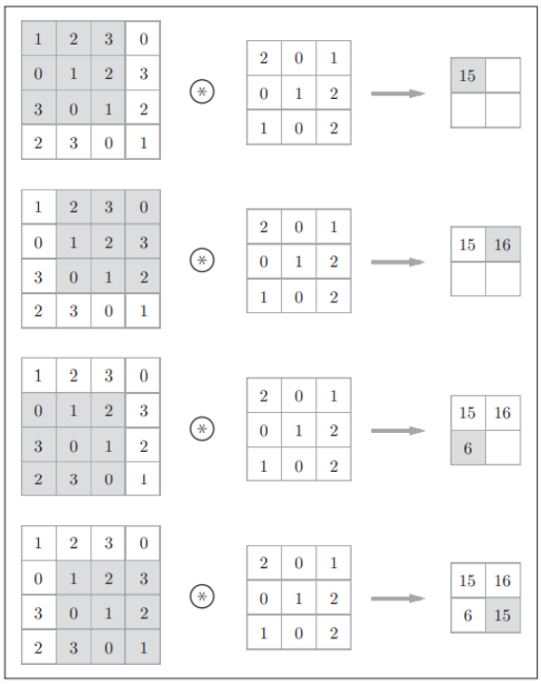

图 3 卷积计算的过程：图中的卷积核大小为 3*3，输入特征图大小为 4*4，计算完成后得到的输出特征图为 2*2

不同于 CPU 串行计算（一个周期完成一条指令）的方式，神经网络专用加速器会大量利用并行计算。如图 4 所示，将图 2 的 conv 模块展开看，模块内部有大量的乘累加模块以完成并行计算，如果有 N 个乘累加模块并行，每周期将完成 N 条计算，相比于串行计算将大大加速。

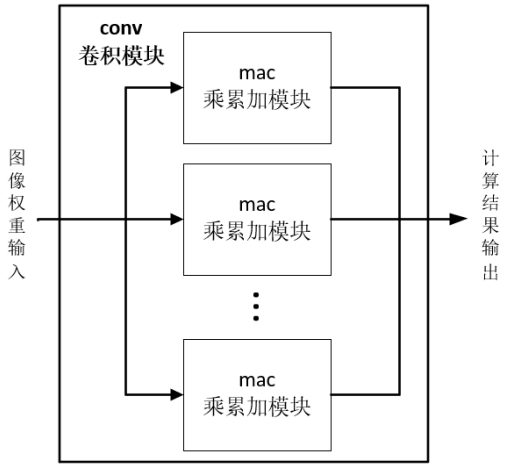

图 4 卷积模块中的并行计算

二、实验电路

1、乘累加器 MAC 模块的设计（需自行填写）

本次实验将完成上述系统中的乘累加模块，该模块在整体系统中的层次结构如图 4 所示。

该实验电路的管脚定义如表 1 所示，具体时序如下所述：

实验电路中的所有输入数据均在 clk 的上升沿到来；当复位信号有效或 en 信号无效时，输出结果均为 0，内部寄存器清零；当复位信号无效且 en 信号有效时，模块开始正常工作。

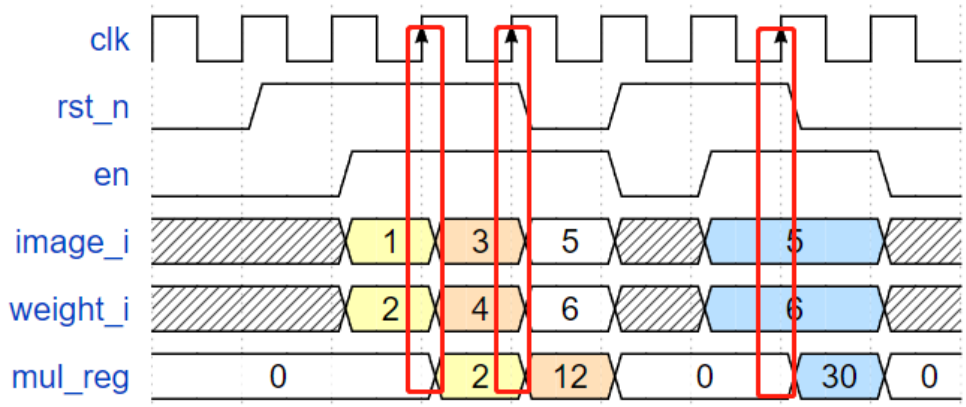

图 7 复位信号、en 信号功能解释波形图（十进制）

为了更好展示以上功能，可以参考图 7 波形，只有在箭头所指的三个 clk 上升沿，复位信号无效且 en 信号有效，输出 mul_reg 在 clk 上升沿后输出有效值。两个条件不满足任何一个时，输出 mul_reg 变为 0。注意：mul_reg 在该图中定义为 image_i*weight_i，该波形也只是解释波形逻辑，并非结果波形图。

first_data 为脉冲信号，当检测到该信号时，开始执行计算，具体计算为：将输入的 image_i 信号与 weight_i 信号相乘，结果暂存；每个时钟周期都会有新的 image_i 信号与 weight_i 信号输入至模块中，此后的每个周期都将对这两者进行乘法，并与之前结果进行累加；该计算持续执行，直至检测到 last_data 信号，完成计算，在 clk 上升沿后将结果输出。

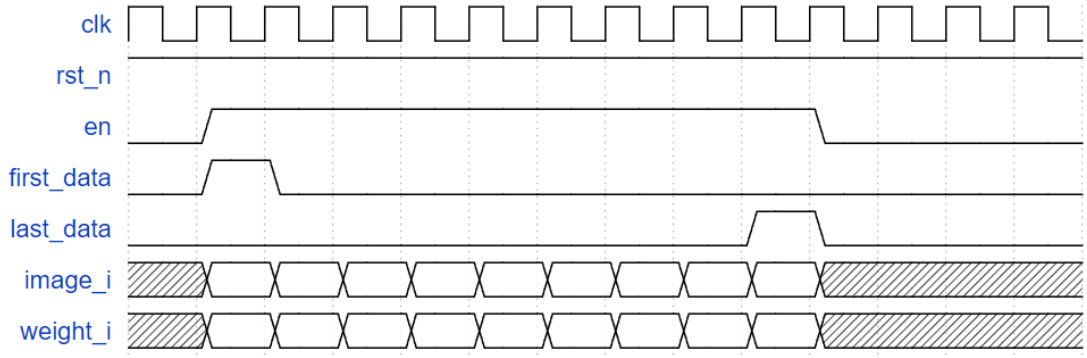

图 8 一种输入 case 的波形图

为了更好展示以上功能，降低实验难度，避免同学走弯路，本实验提供了一种输入 case 的波形图，如图 8 所示。该图可以解释为 weight 和 image 输入 9 个数，控制信号的时序波形也在图 8 中展示，first_data 标志位跟随第一个数据到来，last_data 标志位跟随最后一个数

据到来，last_data之后一个周期，输出结果。同学们可以按照此波形设计mac单元，在testbench中设计更多case（如产生随机数量、随机大小的输入数据），证明设计的正确性。本次实验电路实际应用场景为LeNet5网络，卷积核大小为5*5，累加需要的周期为25。

表 1. 实验电路管脚定义

管脚名	I/O	备注
clk	I	系统时钟，100MHz
rst_n	I	复位信号，低电平有效
en	I	使能信号，在 en 为高时模块工作，为低时不工作
first_data	I	开始标志位，与第一组数据一起到来
last_data	I	结束标志位，与最后一组数据一起到来
image_i	I	8 位有符号数，输入特征图数据
weight_i	I	8 位有符号数，权重数据
q_en	O	输出使能信号，为高时表示输出有效
q	O	16 位有符号数，输出数据（实际应用场景下累加结果不会溢出）

补充说明：

1. 有符号数和无符号数最重要的区别就是如何对运算结果的符号位进行扩展，无符号数是添 0，有符号数是添加符号位。$signed(a)是一个 function，用于运算中符号位扩展。如果事先没有定义 a,b,c 是有符号数，对于 c = a + b 的有符号运算，可以使用 c = $signed(a) + $signed(b)进行转换计算，减法、乘法等其他运算同理。如果已定义该数为有符号数，可以不进行转换。

2. 该 LeNet 网络中另有控制模块控制 first_data，last_data 何时到来，输入特征图和权重数据分别是什么，本实验是完成计算模块，计算模块只需要按上述的接口定义完成。

2、四通道并行卷积设计（代码中已完成）

定制电路相较于通用电路的一大优势是并行度更高，而神经网络正是计算密度高，可大规模并行计算的应用。本次实验还需要同学们完成多输出通道的并行计算，即图 4 中的 conv 模块。

卷积计算的多输出通道指的是多个卷积核对同一个输入特征图的操作，本质上就是基础实验中 mac 模块的复制多份，各个 mac 模块同时计算各自的输出通道，这些 mac 模块输入的特征图数据是相同的，但权重数据是不同的，同时 conv 模块还应将偏置数据 bias 加到输出结果中，且 bias 的计算不增加额外周期。本次实验输出通道的并行度为 4，即 4 个输出通道同时计算。

完成此实验，需要添加两个模块，一个模块用于添加 bias，另一个模块用于完成并行操作。也可以在一个模块中完成所有操作。

第一模块 conv_in1_acc 中，需要例化 mac 模块，然后将输出结果加上有符号变量 bias 得到一个新的有符号数 acc_q。一个卷积模块输入变量宽度和输出变量宽度通常是一样的，但是我们计算完成后的 acc_q 是一个 18 比特数，需要进行截位，在这里我们规定截取[17:10]。

第二个模块 conv_in1 中，需要完成变量转换和并行操作，4 并行操作本质上就是例化 4 个 conv_in1_acc 模块。

conv_in1 模块管脚定义如下：

管脚名	I/O	备注
clk	I	系统时钟，100MHz
rst_n	I	复位信号，低电平有效
aa_en	I	使能信号，在 aa_en 为高时模块工作，为低时不工作
aa_first_data	I	开始标志位，与第一组数据一起到来
aa_last_data	I	结束标志位，与最后一组数据一起到来
image	I	8 位有符号数，输入特征图数据
weight	I	8*4 位有符号数，权重数据，每 8 位为一个通道的权重
bias	I	12*4 位有符号数，偏置数据，每 8 位为一个通道的偏置
q_en	O	输出使能信号，为高时表示输出有效
q	O	8*4 位有符号数，输出数据，每 8 位为一个通道的输出，截取 mac 模块输出的高八位

本实验 Lenet 有 4*25 个权重，权重 4 路 8 位共 32 位一起进行存放。卷积输出 4 路 28*28，有 4 个 bias 对应 4 路，每一路的输出结果都添加同一个 bias，bias4 路 12 位共 48 位一起存放。变量转化相对较难，要把输入 weight 由[31:0]转化为 4 个位宽为 8 比特的 weight_in 变量，参考图 9 波形图。具体的实现方式可以通过定义数组 weight_in[0:3]，或者定义 weight_in0、weight_in1、weight_in2、weight_in3 都行；同理对 bias 权重也要转化;对输出变量 q[31:0]，需要组合 4 个乘累加模块的 8 比特输出变量 acc_q。

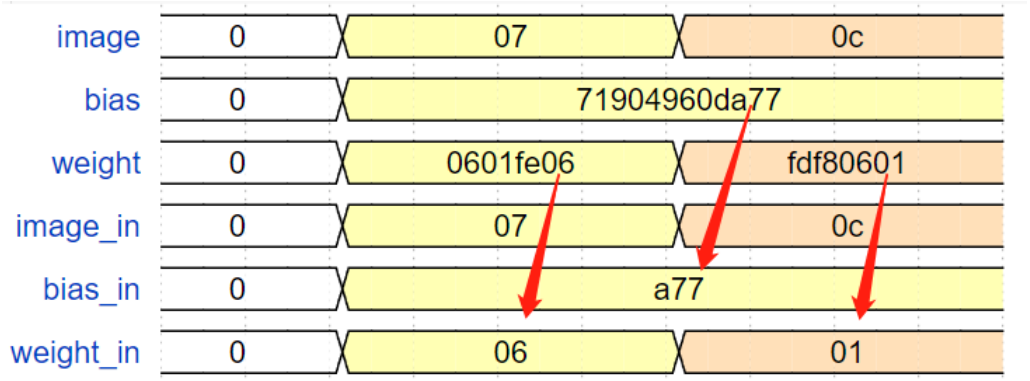

图 9 变量转换波形图

aa_first_data、aa_last_data、aa_en 三个控制变量与 mac 模块中的 first_data、last_data、en 信号功能一样。但**不能用** aa_first_data、aa_last_data、aa_en 直接控制 mac 单元，如图 8 所示，en 与 image_i 是同一个周期到达，再观察图 10 示意波形图，aa_en 与 image 之间相差一个时钟周期，该模块数据 image、bias 与 weight 晚一个周期到达，所以需要将三个控制信号延迟一个周期变成 en、first_data、last_data，与 image 等变量**对齐**后，才能控制 mac 单元。同学们编写 testbench 时，也要将控制信号与数据变量相差一个周期，否则你的设计与验证可能是自洽的，但无法嵌入我们整体 Lenet 网络。

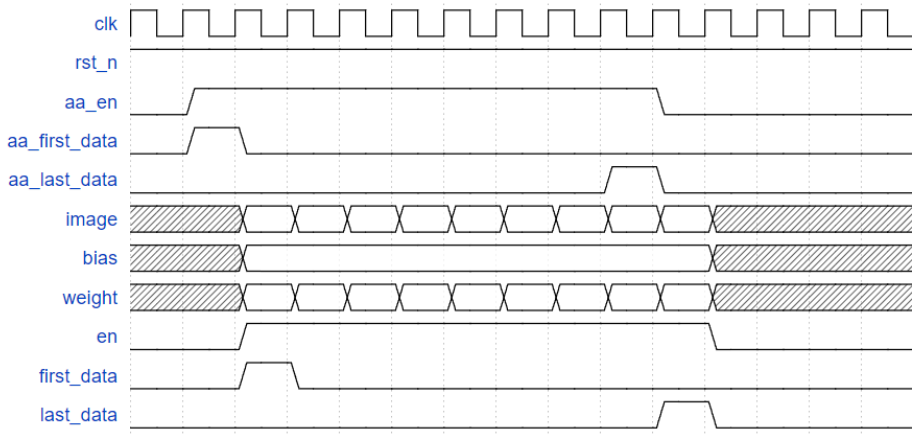

图 10 一种 case 的输入输出波形图