中国科学院大学
《计算机组成原理(研讨课)》实验报告

姓名 薛翼舟 学号 2023K8009929044 专业 计算机科学与技术

实验项目编号 5.3 实验名称 dnn 卷积算法和池化

注 1:	撰写此 Word 格式实验报告后以 PDF 格式保存 SERVE CloudIDE 的 `/home/serve-ide/cod-lab/reports` 目录下(注意: reports 全部小写)。文件命名规则: `prjN.pdf`, 其中 `prj` 和后缀名 `pdf` 为小写, `N` 为 1 至 4 的阿拉伯数字。例如: `prj1.pdf`。PDF 文件大小应控制在 5MB 以内。此外, 实验项目 5 包含多个选做内容, 每个选做实验应提交各自的实验报告文件, 文件命名规则: `prj5-projectname.pdf`, 其中 "-" 为英文标点符号的短横线。文件命名举例: `prj5-dma.pdf`。具体要求详见实验项目 5 讲义。
注 2:	使用 `git add` 及 `git commit` 命令将实验报告 PDF 文件添加到本地仓库 master 分支, 并通过 `git push` 推送到实验课 SERVE GitLab 远程仓库 master 分支(具体命令详见实验报告)。
注 3:	实验报告模板下列条目仅供参考, 可包含但不限定如下内容。实验报告中无需重复描述讲义中的实验流程。

一、 逻辑电路结构与仿真波形的截图及说明(比如 Verilog HDL 关键代码段 {包含注释} 及其对应的逻辑电路结构图 {自行画图, 推荐用 PPT 画逻辑结构框图后保存为 PDF, 再插入到 LaTeX. 中}、相应信号的仿真波形和信号变化的说明等)

本次实验分为两个部分, 分别是硬件部分对于乘法指令的添加 (这一部分很显然, 不多赘述), 另一部分是对 C 语言中的 dnn 卷积算法和池化的实现, 将分别介绍这两部分的实现

1.1 硬件部分

硬件部分的工作就是乘法指令的添加, 这一部分是比较平凡的, 无非就是在 ID 阶段增加对乘法指令的译码, 另外在 EX 阶段增加对乘法指令的处理, 值得注意的就是位数的选取, 各个部分的代码如下

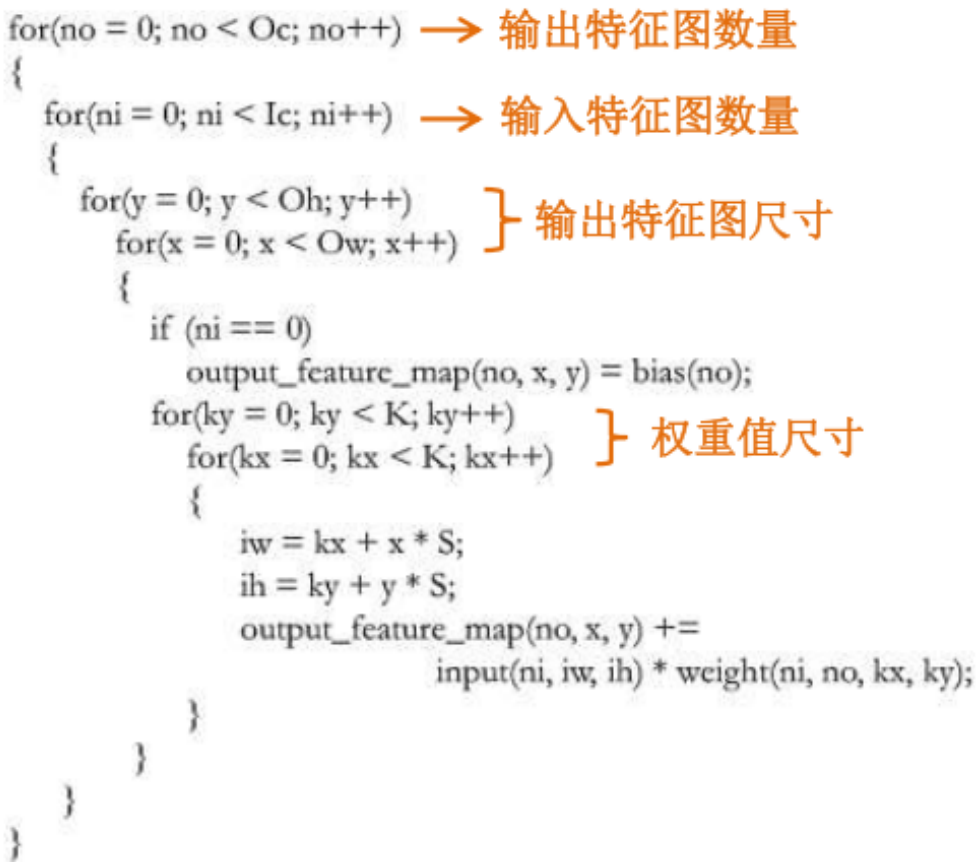
```
//ID阶段
wire mul = (opcode==`R_type && func7==7'b0000001 && func==3'b000);

//EX阶段
wire [63:0] mul_result;

assign mul_result = {64{mul}} & (ALUop_A * ALUop_B);

assign  Result     =     (opcode==`JAL ||opcode==`JALR)?          PC_EX + 4:
                         (opcode==`AUIPC)?                        PC_EX + U_imm:
                         (opcode==`LUI)?                          U_imm:
```

```
                        (mul)?                                  mul_result[31:0]:
                        (AluShi_sel)?                           Shifter_result:
                                                                ALU_result;
```

这一部分所见即所得, 不多赘述

1.2 C 语言部分

这部分代码的实现是本次实验的重点, 主要就是对于卷积算法的编写和池化代码的编写, 主要就是对于讲义上的伪代码进行具体的 c 语言实现, 此外还要考虑对于边界 padding 的考虑, 以下是讲义上的伪代码



接下来是我对 dnn 算法的具体实现, 具体定义的中间变量如下

```
unsigned input_size = mul(input_fm_h,input_fm_w);
        //输入的大小
        unsigned filter_size = 1 + mul(weight_size.d2, weight_size.d3);
        //每组kernel的大小, 每个kernel的第一个元素是bias
        //每个kernel都有一个输出

        unsigned num_out;               //输出个数
        unsigned num_in;                //输入个数
```

```
9         unsigned bias;
10
11        short   input_address;
12        short   filter_outer_address;
13
14        //一些即将用到的的线性坐标
15        short   filter_line_addr;
16        short   input_line_addr;
17
18        //x*S, y*S
19        short   x_stride;
20        short   y_stride;
21
22        //用于标记输出相对于基址的偏移
23        short   position = 0;
24
25        //输入的位置
26        short   iw;
27        short   ih;
28
29        //用于暂存结果，用int来防止溢出
30        int     value_temp;
31        short   value_true;
```

具体代码的实现如下

```
for(num_out = 0; num_out < conv_size.d1; num_out++){
    filter_outer_address = mul(num_out,filter_size);    //调整到对应的filter
    bias = weight[filter_outer_address];
    //当前输出所在的位置
    for(int y = 0; y < conv_out_h ; y++){
        y_stride = mul(y,stride);
        for(int x = 0; x < conv_out_w; x++){
            value_temp = 0;
            x_stride = mul(x,stride);
            for(num_in = 0; num_in < rd_size.d1; num_in++){
                input_address = mul(num_in,input_size);
                for(int ky = 0; ky < weight_size.d2; ky++){
                    ih = ky + y_stride - pad;   //找到对应的输入的位置，放在外面进行可以减少乘法的计算次数，  要考虑pad
                    if(ih < 0 || ih >= input_fm_h){
                        continue; //考虑pad的情况，pad在最周围一圈，放在之前来减少乘法的计算次数
                    }
                    filter_line_addr = mul(ky,weight_size.d3);//注意是宽度
                    input_line_addr = mul(ih,input_fm_w);
                    for(int kx = 0; kx < weight_size.d3; kx++){
                        iw = kx + x_stride - pad;
                        if(iw < 0 || iw >= input_fm_w){
                            continue;
                        }
                        //weight里的i用于跳过bias
                        value_temp = value_temp + mul(in[input_address + input_line_addr + iw], weight[1 + filter_outer_address + filter_line_addr + kx]);
                    }//kx
                }//ky
            }//in
            value_true = (value_temp >> FRAC_BIT);
            out[position] = (short)(bias + value_true);
            position++; //计算之后的值
            //out[position] = (short)(bias + value_temp >> FRAC_BIT);
        }//x
    }//y
}//no
```

主要就是利用了线性存储的行优先存储的特征, 因为是行优先, 所以对于 (i,j) 位置的元素, 其线性坐标为

$$\text{line\_addr} = i \times \text{width} + j \tag{1}$$

于是我们可以根据而借此来索引输入和 filter 中的元素, 另外需要注意的是, 由于 filter 的第一个元素是 bias, 所以在计算时需要将其单独处理, 另外值得一提的是在这次实验中我们考虑了边界 padding 的情况, 具体实现就是根据当前的坐标计算出目前的 ih 和 iw, 并且根据 ih 和 iw 的值是否超出输入的范围来判断, 另外我将对于行的判断放在了倒数第二个循环, 来尽可能的减少循环次数

1.3 池化

池化的代码的思路和卷积部分基本上类似, 有关 padding 的部分是完全相同的, 代码如下

```
for(num_out = 0; num_out < conv_size.d1; num_out++){
    input_address = mul(num_out, input_size);
    for(short y = 0; y < pool_out_h; y++){
        y_stride = mul(y, stride);
        for(short x = 0; x < pool_out_w; x++){
            x_stride = mul(x,stride);
            short max = 0x8000;//初始化最大值, 来实现池化的功能
            for(ky = 0; ky < pool_size; ky++){
                ih = ky + y_stride - pad;
                if(ih < 0 || ih >= input_fm_h){
                    continue; //考虑pad的情况, pad在最周围一圈, 放在之前来减少乘法的计算次数
                }
                input_line_addr = mul(ih, input_fm_w);
                for(kx = 0; kx < pool_size; kx++){
                    iw = kx + x_stride - pad;
                    if(iw < 0 || iw >= input_fm_w){
                        continue; //考虑pad的情况, pad在最周围一圈, 放在之前来减少乘法的计算次数
                    }
                    value = out[input_address + input_line_addr + iw];
                    max = (max > value)? max: value;
                }//kx
            }//ky
            out[position] = max;
            position++;
        }//x
    }//y
}//no
```

依然是利用线性存储的特点来对数据进行索引, 与卷积不同的地方在于, 在每次设置的过程中, 维护一个 max 值来存储每一次的最大值, 另外需要注意的是, 在池化的过程中, 需要对输入的大小进行判断, 如果输入的大小小于 pool_size, 则直接将输入作为输出这个 max 在每一个元素被赋值前初始化位最小的 short 型, 并之后每次与当前的值进行比较, 取最大值, 这样就可以得到池化的结果

二、 实验过程中遇到的问题、对问题的思考过程及解决方法

本次实验比较简单, 难点在于对于池化和卷积这两个算法及其具体实现的理解, 另外就是对于 padding 的处理, 由于 padding 的处理需要考虑到边界的情况, 所以需要对输入的坐标进行判断, 如何判断才能正确的处理边界情况, 怎样的判断才能尽可能地减少时间开销也是本次实验需要考虑的

另外我在本次实验中遇到了一个问题就是数据越界, 这是因为我一开始在设置时候用了一个 int 来存储结果, 导致最终出现了问题

另外还有一个问题是一个低级错误, 我在 ex 阶段的 result 把 mul_result 中的按位与写成了逻辑与.....

三、 实验所耗时间

在课后，你花费了大约____6____ 小时完成此次实验。