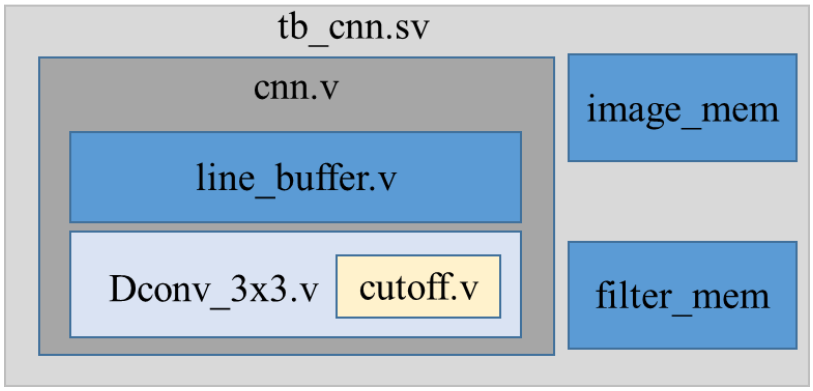
人工智能芯片导论实验 —— 卷积运算量化

实验描述

在此实验中，您将根据要求编写 Verilog HDL 代码和 python 代码，针对 3x3 卷积得到输出特征图的量化结果，包括饱和、舍入量化的操作。



实验背景

1、被处理图像大小为 66*66 的像素阵列 image，每个像素用 8bits 有符号数表示，整张图片数据量为 66*66 个 8bits 有符号数据，数据存储于外部的存储器中。数据的存储格数如下，共 4356 个 8bits 有符号数据连续存储，起始地址为 0，数据阵列的最外一圈其实都是 0，是为了计算方便而填充的。

Image[0]	Image[1]	Image[2]	…	…	…	…	Image[65]
Image[66]	Image[67]	Image[68]	…	…	…	…	Image[131]
Image[132]	Image[133]	Image[134]	…	…	…	…	Image[197]
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	Image[4223]
…	…	…	…	…	…	Image[4288]	Image[4289]
…	…	…	…	…	Image[4353]	Image[4354]	Image[4355]

2、对图像数据进行卷积运算的卷积核 Filter 大小为 3*3Bytes，也可以说卷积核为一个 3*3 的 8bits 有符号数据矩阵。数据存储于外部存储器中。数据的存储格式如下，

Filter[0]	Filter[1]	Filter[2]
Filter[3]	Filter[4]	Filter[5]
Filter[6]	Filter[7]	Filter[8]

3、针对图像上的每 3*3 的像素点，与给定的 3*3 的卷积核做一次卷积，卷积的结果作为输出对应图像像素的一个点，这样，完整的输出是一个 64*64 的矩阵共 4096 个元素，矩阵的每个元素是 8bits 有符号数据。

例如，针对输入图像最左上角的 3*3 的像素，我们需要完成：

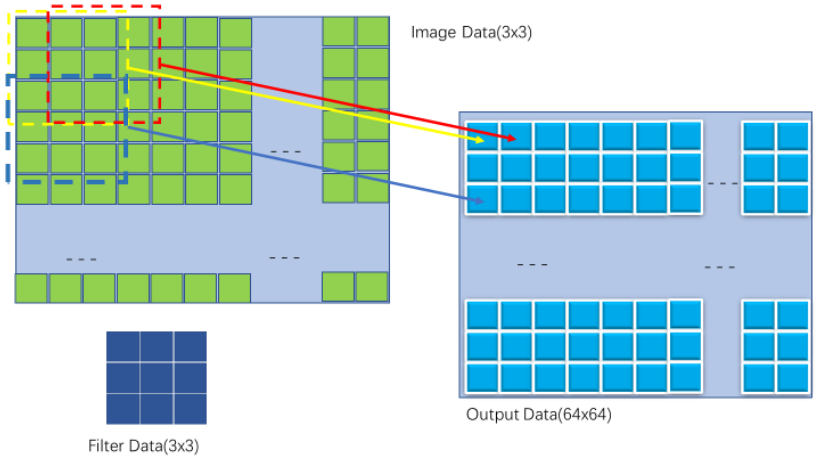
a. Output[0] = Image[0] *Filter[0]+ Image[1]*Filter[1]+ Image[2]*Filter[2]+
Image[66] *Filter[3]+ Image[67]*Filter[4]+ Image[68]*Filter[5]+
Image[132] *Filter[6]+ Image[133]*Filter[7]+ Image[134]*Filter[8]

b. Output[1] = Image[1] *Filter[0]+ Image[2]*Filter[1]+ Image[3]*Filter[2]+
Image[67] *Filter[3]+ Image[68]*Filter[4]+ Image[69]*Filter[5]+
Image[133] *Filter[6]+ Image[134]*Filter[7]+ Image[135]*Filter[8]

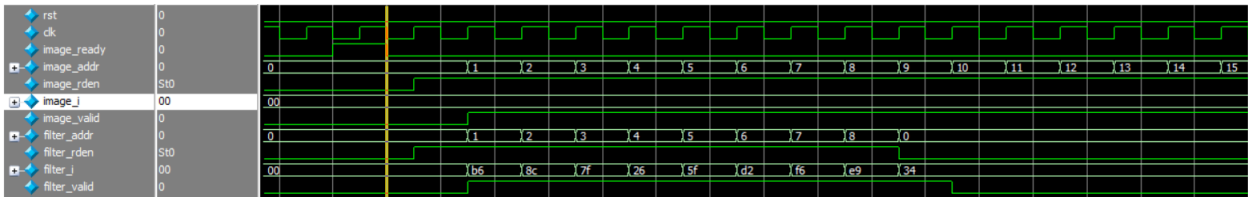
c. 以此类推，到最后一个：
Output[4095] = Image[4221] *Filter[0]+ Image[4222]*Filter[1]+ Image[4223]*Filter[2]+
Image[4287] *Filter[3]+ Image[4288]*Filter[4]+ Image[4289]*Filter[5]+
Image[4353] *Filter[6]+ Image[4354]*Filter[7]+ Image[4355]*Filter[8]

d. 以此类推，计算出全部输出的 64*64 个点的卷积值，并将其输出。

e. 卷积计算的示意图如下：



4、提供的 testbench 文件(tb_cnn.sv)可以自动将 image 数据文件(image_data.txt)和 Filter 数据文件(weight_data.txt)读入，image 数据和 Filter 数据的外部存储接口时序一致，如下图所示，



实验要求

1、提供的 Dconv_3x3_PE 模块可以完成 3x3 卷积运算，但欠缺一个模块完成量化，即饱和与舍四进五操作，其中 cutoff 模块的声明为

```
module cutoff#(
    parameter input_width       = 20,
```

```
  parameter output_width      = 8,
  parameter radix_point_right  = 8
  )(
  input wire  clk,       // clock signal
  input wire  rst_n,     // reset signal
  input wire  [input_width  - 1 : 0] data_in,
  output reg  [output_width - 1 : 0] data_out
  );
```

其中 radix_point_right 标注了小数点位置，请在 cutoff.v 文件中完善代码，不妨假定输入特征图为 Fix_8_5，权重为 Fix_4_4，输出特征图为 Fix_8_1。

2、利用 HDL 仿真软件运行 tb_cnn.sv，可以得到 3x3 卷积计算及量化后的结果，输出于 cnn_output.txt 文件，试利用 python 编写 golden model，即 3x3 卷积计算及量化结果，并与硬件输出结果相比较，验证结果正确性。

3、 附加思考题(可选)：

(1) 若同时在量化模块(cutoff.v)完成激活函数 ReLU 操作，代码会有什么不同。

(2) 对于单引擎架构，不同网络层在同一硬件上复用，而不同网络因为数据分布不同，小数点位置会发生变化，此时 cutoff.v 的声明变成

```
module cutoff#(
  parameter input_width       = 20,
  parameter output_width      = 8
  )(
  input wire  clk,       // clock signal
  input wire  rst_n,     // reset signal
  input wire  [5          - 1 : 0] radix_point_right,
  input wire  [input_width  - 1 : 0] data_in,
  output reg  [output_width - 1 : 0] data_out
  );
```

试分析现有硬件代码(cnn.v / cutoff.v)能否支持不同小数点位置发生变化，如不支持，如何修改。

(3) 阅读 Dconv_3x3_PE.v 分析卷积计算数据流，请指出关键路径与数据复用方式。

(4) python 的 round() 函数与硬件舍四进五操作不同，试分析如何解决

需要提交的材料：

1. 硬件 VerilogHDL 量化代码 cutoff.v
2. 软件量化代码
3. 实验报告：代码逻辑、结果、附加题讨论(可选)

评分标准：满分 100 分

1、Python 代码 (50 分) + Verilog 代码(50 分)
2、量化结果计算正确，VerilogHDL 代码可综合
3、代码风格，实验报告与分析
4、附加题讨论(可选)(10 分)
5、严禁抄袭，违者零分