Homework 5 问题描述

本次作业包括3个问题。

Problem 1 & 2

针对GEMM $Y(i,j)+=A(i,k)\times B(k,j)$

请分别依照以下两个问题中的STT，依照TensorLib的方法，使用Chisel完成PEArray模块。

其中，模块名都为PEArray，但分别写在两个不同的文件（PEArray1，PEArray2）中，且package名不同。（package定义部分已经包含在给出的模板中）

PE Array的大小为4×4

参数：`dtype` 数据类型

输入：

- `a_in: Vec(4, dtype)`， Tensor A的输入
- `b_in: Vec(4, dtype)`，Tensor B的输入
- `c_in: Vec(4, dtype)`， Tensor C的部分和
- `stationaryCtrl: Bool`， 当stationaryCtrl为1.B时，从对应的输入通道载入数据，否则保持stationary。有且仅有一个Tensor的reuse类型是stationary，这个控制信号用于控制其数据载入（和移出，如果是输出张量）。

输出：`c_out: Vec(4, dtype)`， Tensor C的输出

其中，输入输出Vec的方向与STT映射之后的方向一致，例如：假设Tensor A沿着x轴正方向systolic，那么a_in(0)从PE(0,0)进入，a_in(1)从PE(0,1)进入，以此类推。对于Stationary的，一律沿着y轴正方向传递，因此输入的PE为PE(0,0)，PE(1,0)……以此类推。

除了代码之外，还可以提交一份可选的简单报告。这份报告的内容包括：

- 3个张量各自的Reuse类型和方向（包括计算过程）
- PE内部结构，以及PE Array设计的大致描述。

这份报告用于无法实现代码时酌情给分。

Problem 1 STT

$$T=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Problem 2 STT

$$T=\begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}$$

Problem 3

这是一个难度较高的问题，考虑每一个信号都被Ready/Valid所控制的情况，实现为PEArray3。
由于不同位置的数据可能在不同的时间Valid，因此STT的调度在这个情况下会失效。在这个问题中，我们直接要求PEArray3的数据传输方式是：张量A沿着X正方向systolic传播，张量B沿着Y正方向systolic传播，张量C是stationary的，其输入数据沿着Y正方向传入，输出数据同样沿着该方向传出（每次传递的距离和周期都为1）。只需要专注于PE Array的部分，可以不用管输入的数据在矩阵乘法中的具体含义是什么。

输入输出改为

输入：

- `a_in: Vec(4, DeqIO(dtype))`
- `b_in: Vec(4, DeqIO(dtype))`
- `c_in: Vec(4, DeqIO(PData(dtype)))`

输出：`c_out: Vec(4, EnqIO(PData(dtype)))`

其中，`PData`是一个Bundle，包括两项：

- `data: dtype`，数据
- `pos: UInt(2.W)`，位置，这里是y坐标(0~3)

注意Vec中每一个元素被单独的信号所控制。数据systolic传递的方式与队列相同。

张量C的stationary的控制方式这里作出这样的假定：

- 当`c_in`有一个valid的数据传入时，将其沿着Y正方向传递。
- 当其`pos`域与PE的y坐标相同时，在下一个周期及之后将其`data`用于计算中，直到被下一个替换。
- 当一个数据被替换时，将其沿着Y正方向传递出去，到`c_out`输出。（提示：输入和输出应是两条不同的通道）
- 不存在初始数据，即reset后所有C都默认valid=0
- 只有在A和B全部停止输入至少8周期后，c_in才会有输入。之后当每一个c_out都得到了4个输出之后，才会继续输入A和B。
- 测试时，只检测结果是否正确，`c_out`的`pos`顺序可以是乱序的。测试时，c_out最多等待8个ready的周期，超过了则认为出现了错误（例如某一个c_out在第1,3,5,7,9,11,13,17个周期ready=1，如果第17个周期结束时，仍然没有总共收到4个valid的输出，那么就认为PE Array内部出现了问题）。

在计算单元$c' = a * b + c$中，当所有输入a, b, c同时valid时且输出ready才进行计算。张量A和张量B的数据，需要在路过的每一个PE中都进行一次计算，才能继续往后传递。

这个问题的报告可以简单写一下自己是怎么设计PE与PE array的。

提示

- 请在提供的代码模板的`//TODO`位置实现，不要更改模块名和package名。
- 使用`mill PEArray.test`来运行测试
- 仔细观察提供的test bench以确定寄存器的数量使得延迟正好与测试样例一样
- Reuse分析的正确性可以通过观察test bench验证
- 结果在输出前必须储存在寄存器中，否则测试时会报错`chiseltest.ThreadOrderDependentException`
- 中间结果都用dtype储存即可，溢出不需要处理。
- stationary的输入保证在其他操作之前，输出保证在其他操作之后。
- 张量的大小可以大于4x4，请思考这个情况下stationary会怎样，`c_in`所起的作用。
- 寄存器的使用有一定的容忍范围：在PEArray1和PEArray2中，`c_in`的输入可以比`a_in`和`b_in`的迟0~5个周期，测试时会逐个尝试，只要有一个延迟可以得出正确结果，就算对。

- 自动测试无法确定你是否是用的Systolic Array来解决这些问题的。如果你使用了别的方法来解决这些问题，由于不符合题目要求，会适当扣分。

提交方式

代码题需要提交代码和报告。请按下述文件结构上传至服务器中。（简单来说，把下发的PEArray文件夹里面的PEArray1/2/3.scala改成自己的结果，然后复制到~\homework\5中，并同时上传报告）

要求文件结构：

```
+ ~\homework\5
+ --- report.doc/docx/pdf/md
+ --- + PEArray
+ --- + --- + src
+ --- + --- + --- PEArray1.scala
+ --- + --- + --- PEArray2.scala
+ --- + --- + --- PEArray3.scala
```

其他要求：

- 请在report中写好自己详细姓名学号。
- 如果没有做出某一道代码题，也请直接提交代码模板。

评分标准

提供部分Test Bench验证代码的正确性，具体方法同课上练习。

对于每一个问题：

- 当代码正确时：不看报告。
- 当代码不正确时，看设计酌情给分，总共最多能得一半的分数。如果代码本身已经得了超过一半的分数，报告不再得分。