Homework 4: Systolic Array

> ***DDL: 2023.12.29 23:59.*** *请将作业提交到**服务器指定位置***
>
> *本次作业包含 3 个问题，请独立完成*

Problem 1 & 2

针对 GEMM $Y(i,j) \mathrel{+}= A(i,k) \times B(k,j)$

请分别依照以下两个问题中的 STT，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下发文件给出的模板中）

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，如果沿着 y 轴正方向传递，输入的 PE 为 PE(0,0)，PE(1,0)，... 以此类推；如果沿着 x 轴正方向传递，输入的 PE 为 PE(0,0), PE(0,1), ... 以此类推。Problem 1 中 Stationary Tensor 沿 y 轴正方向传递，Problem 2 中 Stationary Tensor 沿 x 轴正方向传递。**

除了代码之外，需要提供一份简单的报告，报告内容包括：

- 三个张量各自的 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

无法实现代码时，助教会根据报告酌情给分。

Problem 1 STT	Problem 2 STT
$T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

Problem 3

本问题难度相对较高，考虑每个信号都被 Ready/Valid 信号控制，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：

- `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，包含两项：

- `data: dtype`，数据
- `pos: UInt(2.W)`，位置，这里表示 Y 坐标 （$0 \sim 3$）

注意 Vec 中每个元素被单独的 ready/valid 信号所控制。数据 systolic 传递的方式与 FIFO 相同。

张量 C 的 stationary 控制方式这里作出如下假定：

- 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 时才输出 ready 且进行运算。张量 A 和 B 的数据需要在路过的每一各 PE 中都进行一次运算，才能继续往后传递。

同学们可以在报告中简单描述自己 PE 和 PE Array 的设计。

提示

- 请在所提供代码模板的 `//TODO` 位置实现你的代码，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 /root/homework/5 中，并同时上传报告）

要求文件结构

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

其他要求：

- 请在 report 中详细写好自己的姓名学号。
- 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

我们会使用 Test Bench 测试你的代码，对于每一个问题：

- 当代码正确时，得满分，助教仅检查报告和代码的一致性。
- 当代码不正确时，根据代码和报告中的设计酌情给分，最多能获得一半分数。