智能计算系统

实验第六章 *深度学习处理器运算器设计

张欣

中国科学院计算技术研究所

实验目的

掌握深度学习处理器中运算器的设计原理，能够使用 Verilog 语言实现内积运算器及矩阵运算子单元的设计并进行功能仿真。具体包括：

1) 理解深度学习处理器加速卷积运算的原理，理解本实验和深度学习处理器基本模块间的关系。

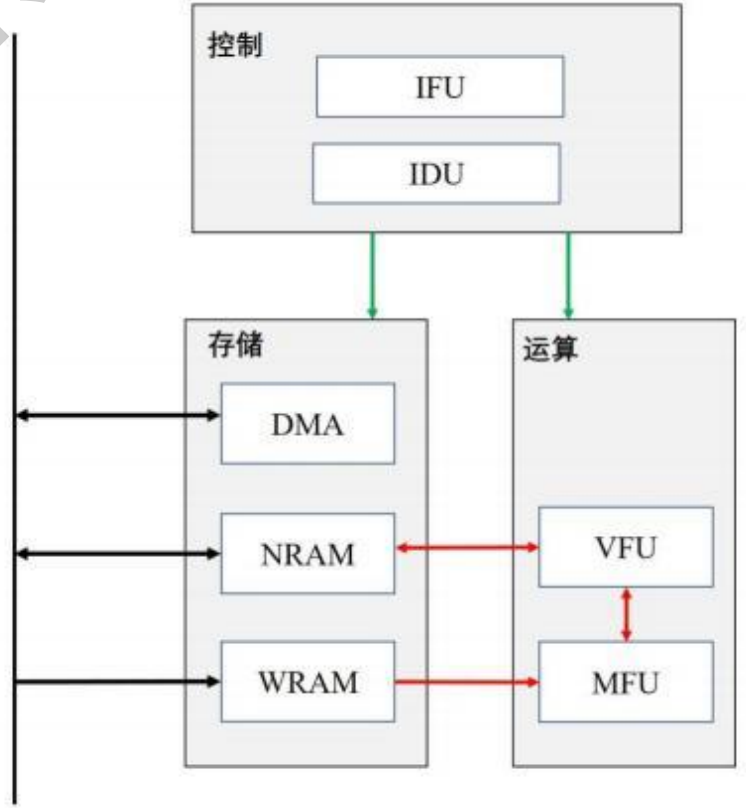
2) 利用 Verilog 语言实现**串行内积运算器**，理解内积运算器的基本组成单元。

3) 在串行内积运算器基础上，利用 Verilog 语言实现**并行内积运算器**，加深对深度学习处理器加速卷积计算原理的理解。

4) 在并行内积运算器基础上，利用 Verilog 语言实现**矩阵运算子单元**，加深对矩阵运算子单元的理解。

实验原理

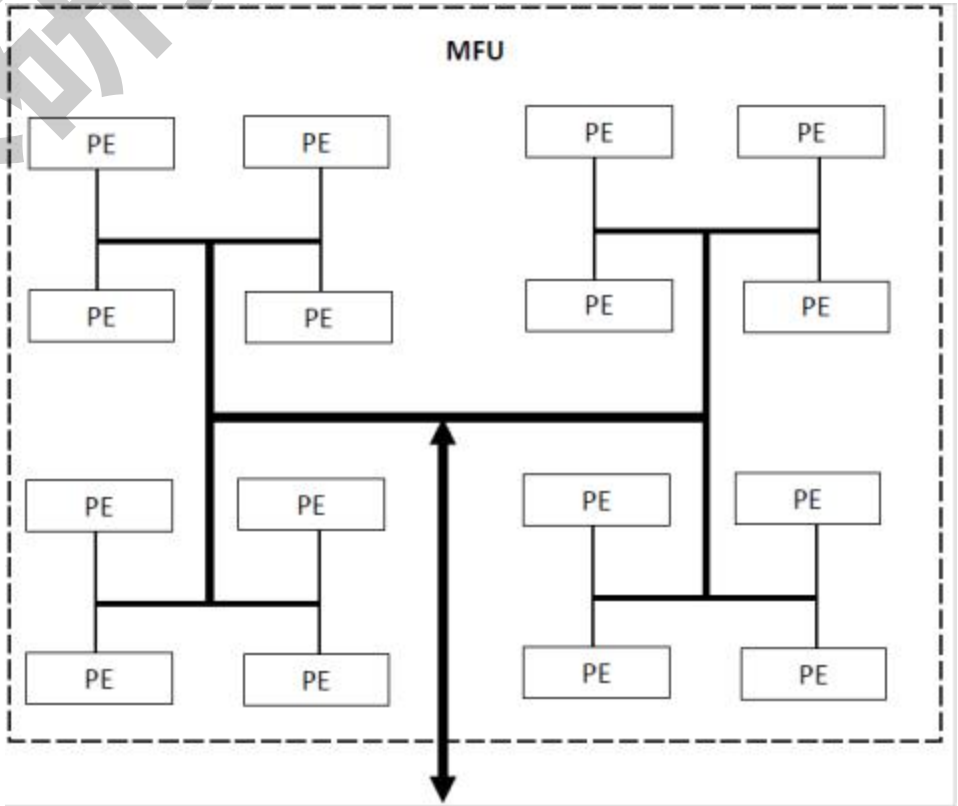
- **控制模块:** 负责协调控制运算模块和存储模块完成深度学习任务
  - 取值单元（Instruction Fetch Unit，IFU)
  - 指令译码单元（Instruction Decode Unit，IDU）
- **存储单元**:
  - 直接内存存取单元（Direct Memory Access，DMA ）
  - 神经元存储单元（Neuron RAM，NRAM）
  - 权重存储单元（Weight RAM，WRAM）
- **运算模块:**
  - 向量运算单元（Vector Function Unit，VFU ）
  - 矩阵运算单元（Matrix Function Unit，MFU）



深度学习处理器总体架构

矩阵运算单元（MFU）

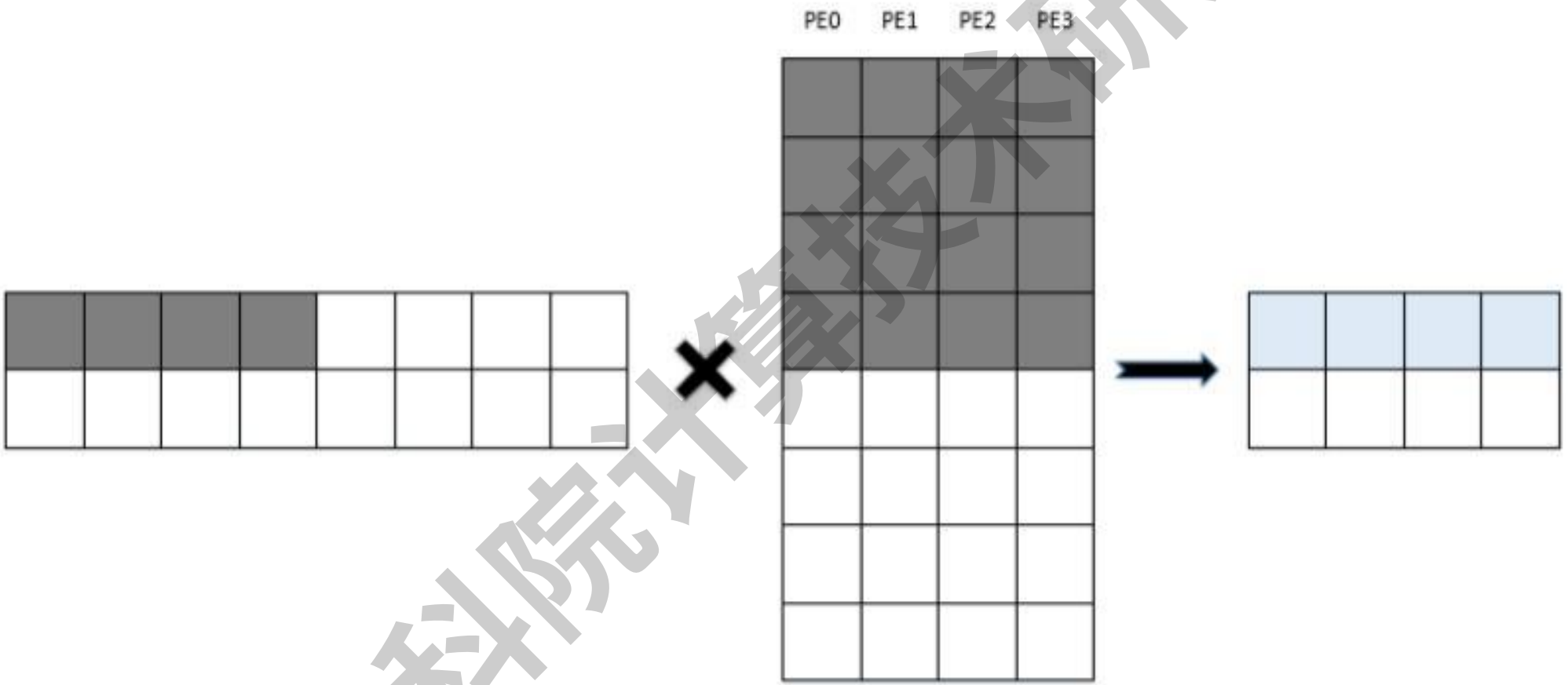
- H树互联
  - 将 M 个矩阵运算子单元（Processing Element，PE）组织为一个完整的矩阵运算单元。
  - 不同 PE 位于 H 树的叶节点。
  - H 树将预处理后的输入神经元和控制信号广播到所有 PE，并收集不同 PE 计算的输出结果返回给 VFU。
- 矩阵运算子单元
  - 向量的内积运算
  - 主要由 N 个乘法器和一个 N 输入的加法树组成



矩阵运算单元结构

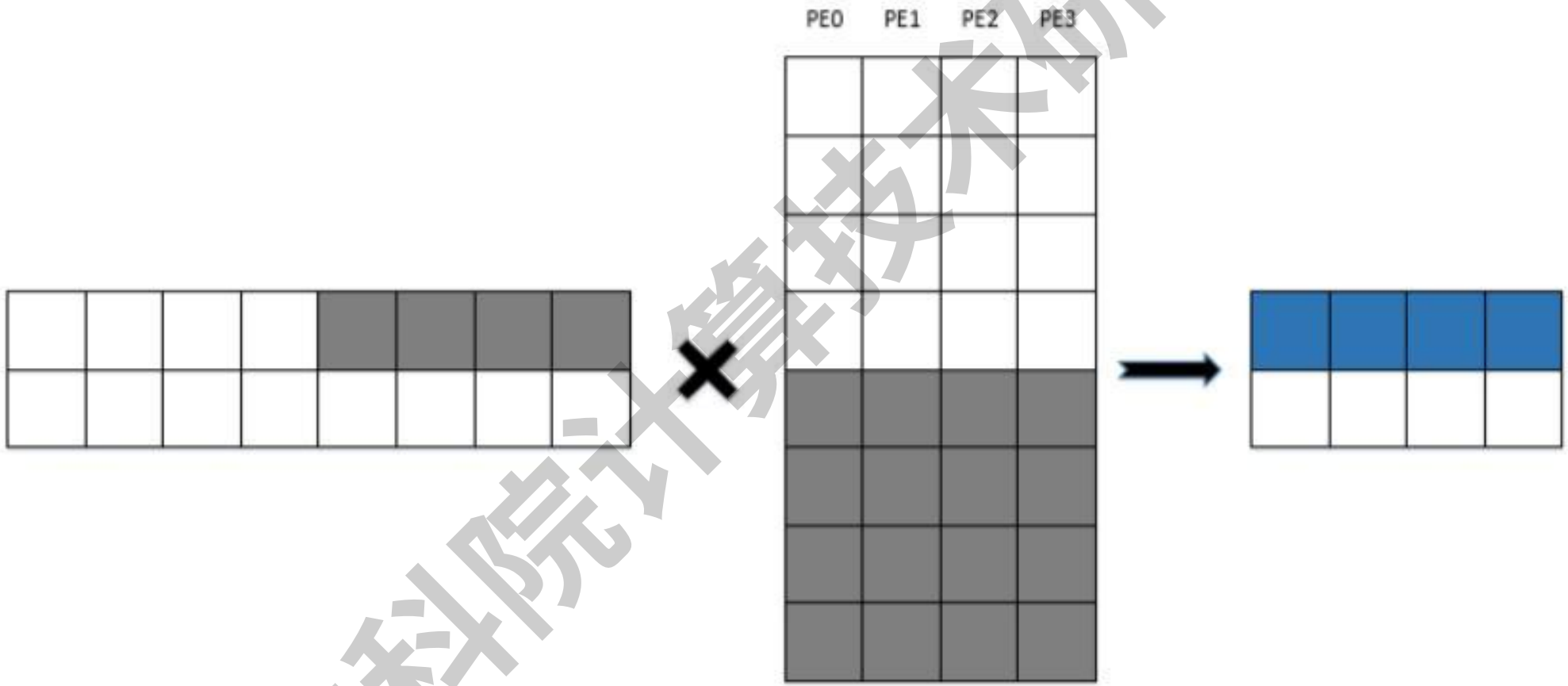
矩阵乘计算过程

假设矩阵运算单元（MFU）包含4个矩阵子单元（PE），矩阵运算单元需4拍完成矩阵运算：



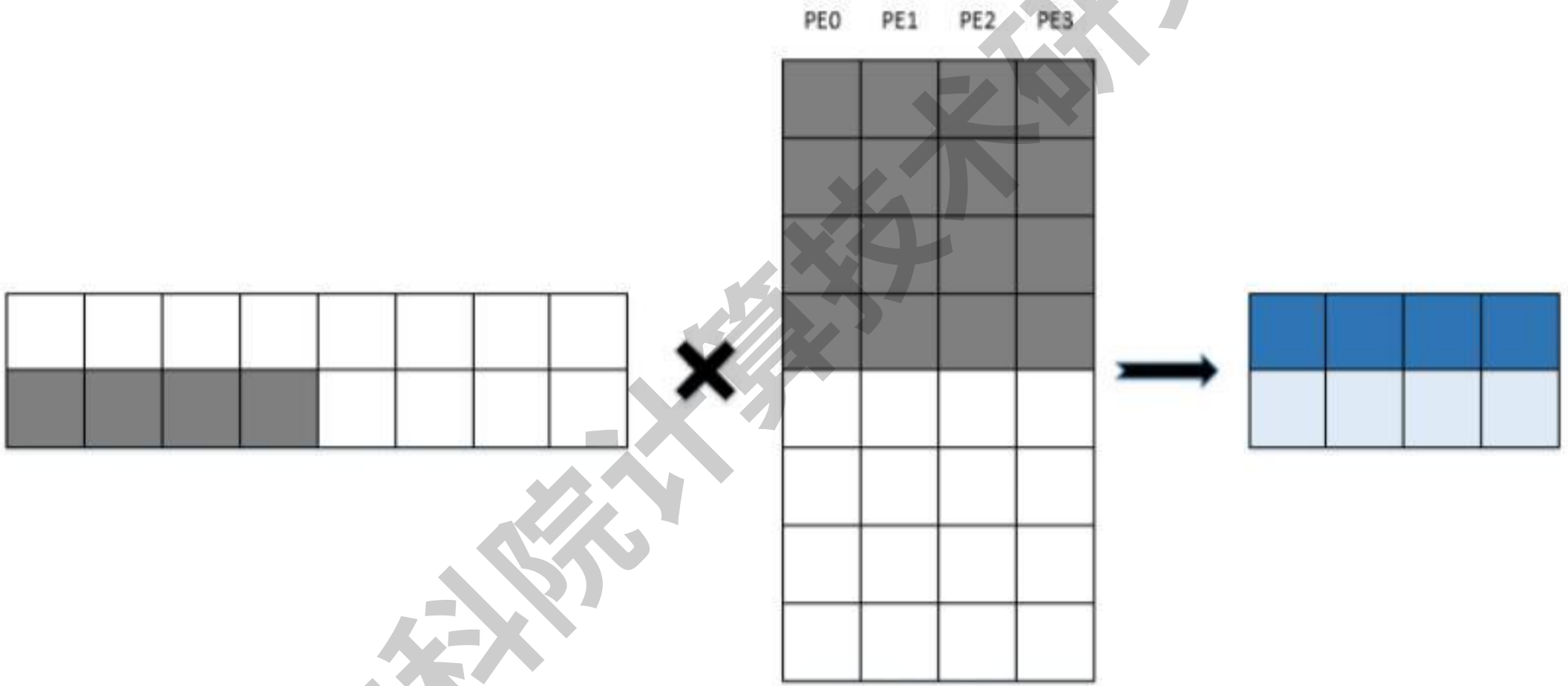
第 1 拍，第一行的前 4 个神经元广播给所有 PE，每个 PE 接收对应列权重的前 4 个元素，分别进行内积运算，得到第一行 4 个输出神经元结果的部分和。

矩阵乘计算过程



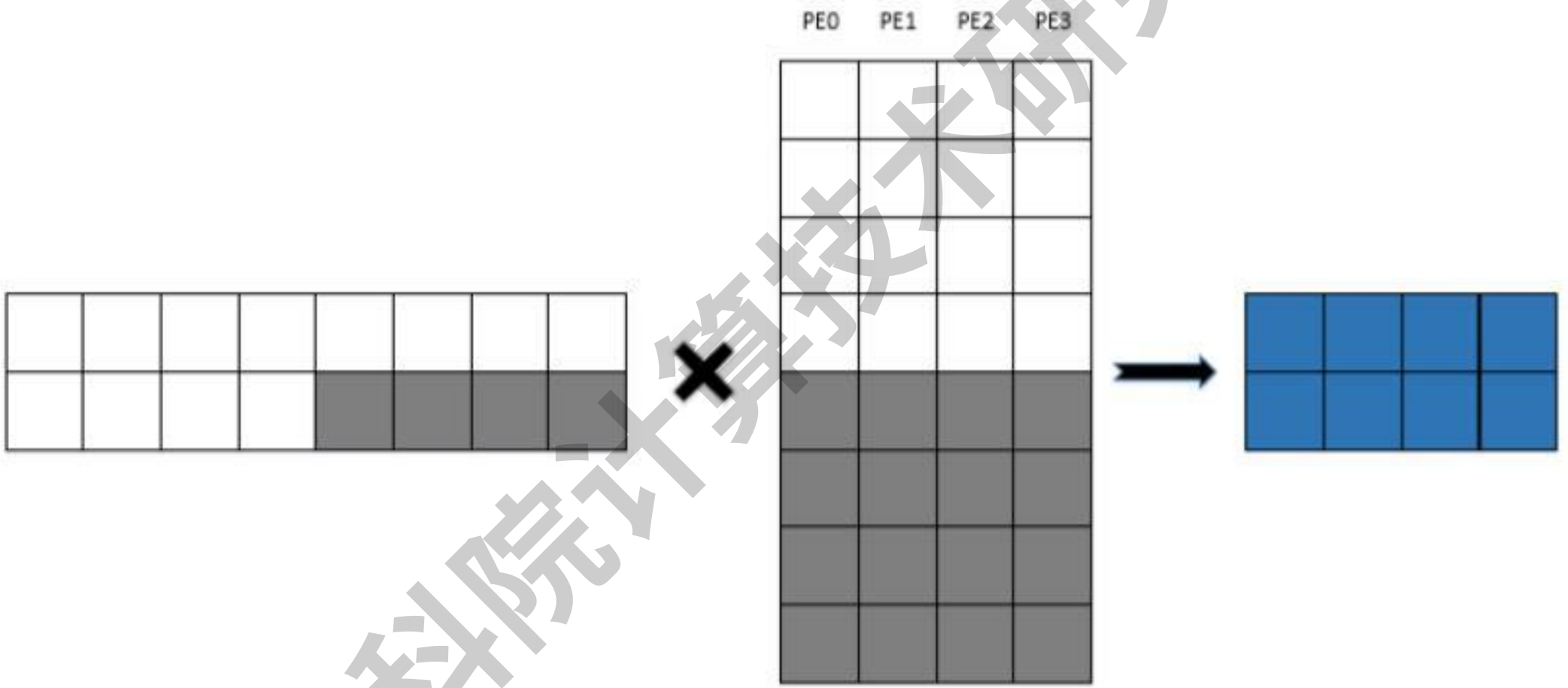
第 2 拍，第一行的后 4 个神经元广播给所有 PE，每个 PE 接收对应列权重的后 4 个元素，分别进行内积运算，然后累加第 1 拍计算的部分和结果，得到第一行 4 个输出神经元。

矩阵乘计算过程



第 3 拍，第二行的前 4 个神经元广播给所有 PE，每个 PE 接收对应列权重的前 4 个元素，分别进行内积运算，得到第二行 4 个输出神经元结果的部分和。

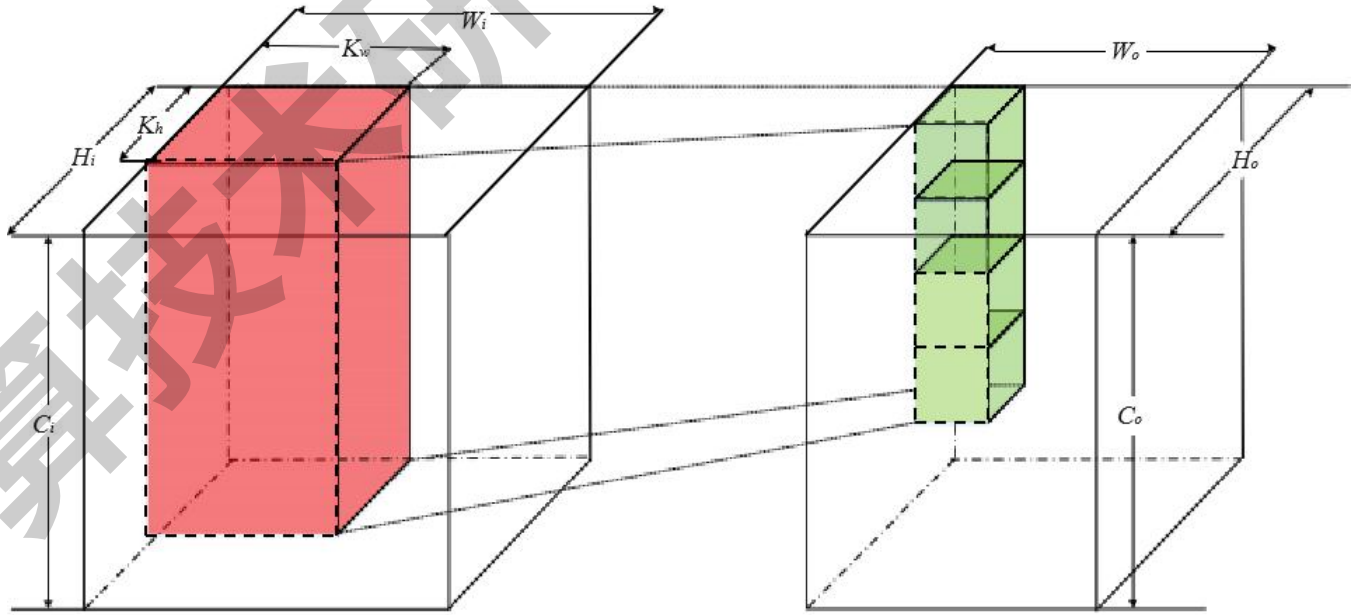
矩阵乘计算过程



第 4 拍，第二行的后 4 个神经元广播给所有 PE，每个 PE 接收对应列权重的后 4 个元素，分别进行内积运算，然后累加第 3 拍计算的部分和结果，得到第二行 4 个输出神经元。

卷积计算过程

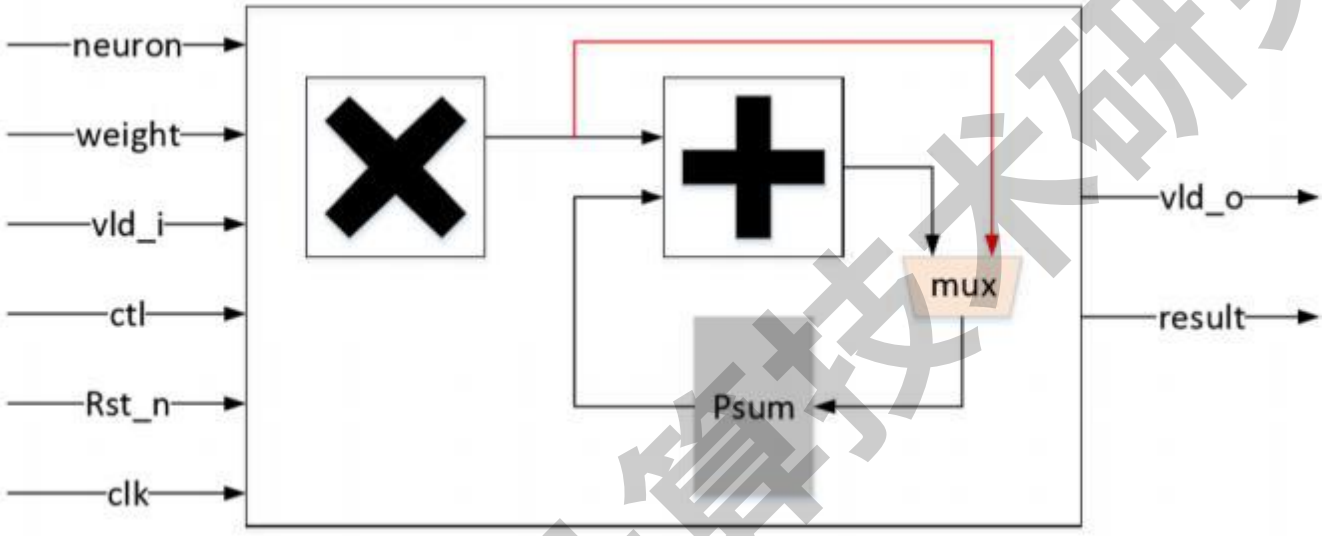
- DLP 计算一组输出特征图中不同输出特征图上相同位置的神经元过程类似于矩阵运算。运算过程可拆分为 4 个步骤：
  - 步骤一：VFU 依次将维度为 $C_i \times K_h \times K_w$ 的数据子块从 NRAM 读出，进行数据预处理后发送给 MFU。
  - 步骤二：MFU 将接收到的输入神经元广播给 n 个 PE 单元。
  - 步骤三：PE 接收 H 树广播的输入神经元，与从 WRAM 读取的权重数据做内积运算，并将内积结果保存至部分和寄存器，或将内积结果累加到部分和寄存器。
  - 步骤四：PE 收到维度为 $C_i \times K_h \times K_w$ 的数据子块的所有数据后将计算的部分和输出。



实验内容

- 矩阵运算子单元（PE）
  - 串行内积运算器设计：串行内积运算器每拍处理一个神经元和一个权重分量的乘累加运算。
  - 并行内积运算器设计：并行内积运算器每拍并行处理多个神经元和权重分量的乘累加运算。该运算器是矩阵运算子单元的基本运算单元。
  - 矩阵运算子单元设计：矩阵运算子单元（PE）可根据控制信号接收神经元数据和权重数据，然后进行内积运算。PE 可以通过在并行内积运算器基础上增加控制逻辑来实现。

串行内积运算器



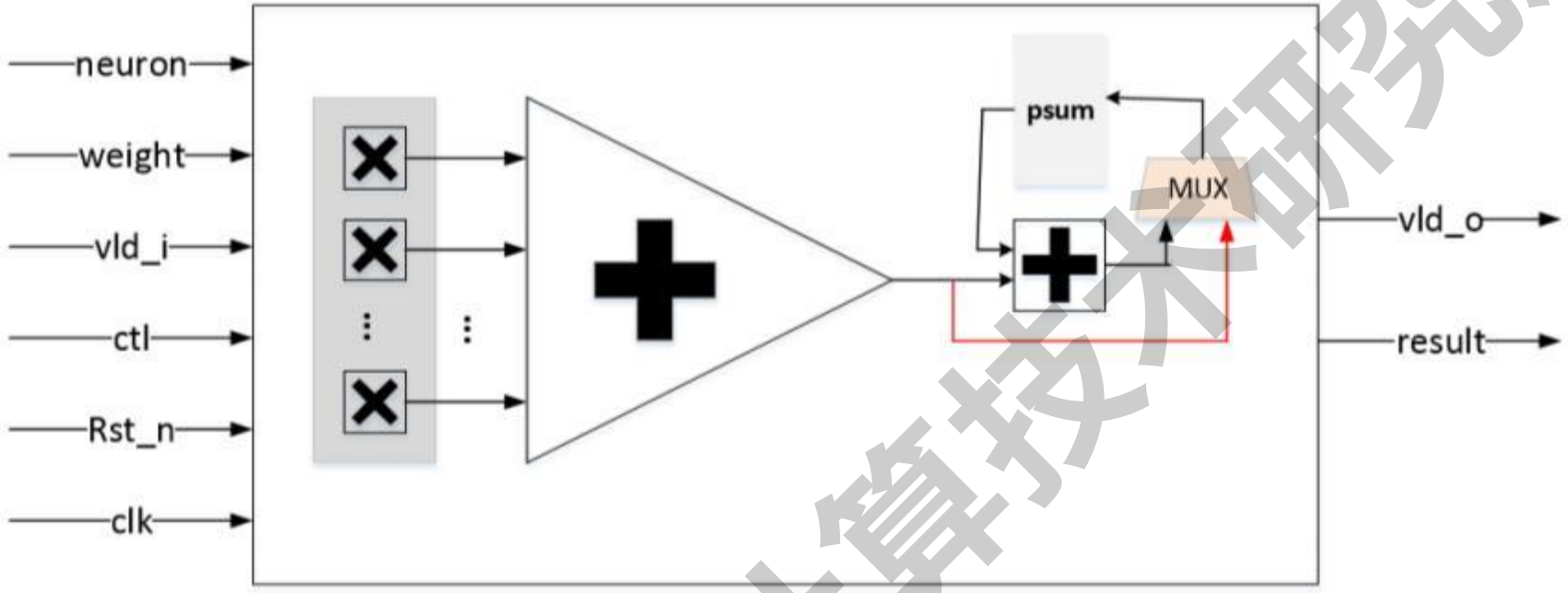
- 串行内积运算器每拍最多接收一个神经元（neuron） 和一个权重分量（weight）进行乘法运算，然后将乘法结果累加到部分和寄存器。
- 串行乘累加
  - 一个乘法器
  - 一个加法器
  - 一个选择器
  - 一个部分和寄存器

串行内积运算器

输入输出接口信号

域	位宽	功能描述
neuron	16	输入 INT16 神经元分量
weight	16	输入 INT16 权值分量
vld_i	1	输入数据和控制信号有效标志，高电平有效
ctl	2	输入控制信号 ctl[0]: 输入神经元/权值数据是一组神经元/权值向量的第一个元素 ctl[1]: 输入神经元/权值数据是一组神经元/权值向量的最后一个元素
rst_n	1	输入复位信号，低电平有效
clk	1	输入时钟信号
result	32	输出内积结果
vld_o	1	输出内积结果有效标志，高电平有效

并行内积运算器



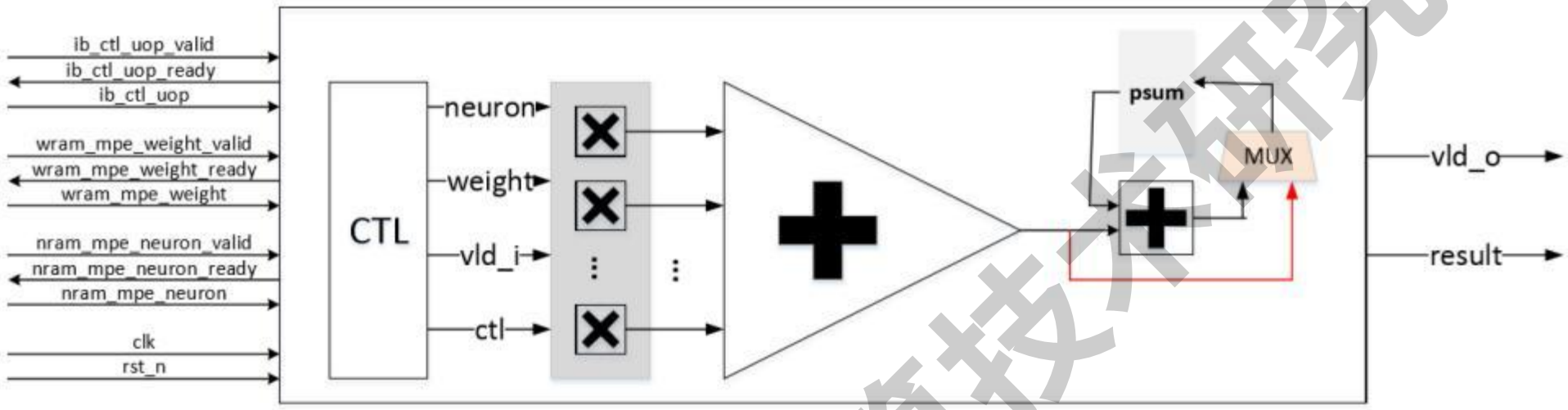
- 不同于串行内积运算器每拍最多接收一个神经元和一个权重分量进行乘法运算，并行内积运算器每拍接收多个神经元和权重分量。
- 并行乘累加
  - 一组乘法器
  - 累加器
  - 一个加法器
  - 一个选择器
  - 一个部分和寄存器

并行内积运算器

输入输出接口信号

域	位宽	功能描述
neuron	512	输入神经元数据，包括 32 个 INT16 分量
weight	512	输入权值数据，包括 32 个 INT16 分量
vld_i	1	输入数据和控制信号有效标志，高电平有效
ctl	2	输入控制信号 ctl[0]: 输入神经元/权值数据是一组神经元/权值向量的第一个元素 ctl[1]: 输入神经元/权值数据是一组神经元/权值向量的最后一个元素
rst_n	1	输入复位信号，低电平有效
clk	1	输入时钟信号
result	32	输出内积结果
vld_o	1	输出内积结果有效标志，高电平有效

矩阵运算子单元



- 矩阵运算子单元根据控制信号来接收 H 树广播的神经元向量和从 WRAM 读取的权重向量数据进行内积运算。由于矩阵运算子单元和 H 树总线、WRAM 相连接，对应神经元、权重、输出结果的接口信号使用 valid-ready 握手机制的总线信号。
  - 并行乘累加
    - 控制单元
    - 一组乘法器
    - 累加器
    - 一个加法器
    - 一个选择器
    - 一个部分和寄存器

矩阵运算子单元

输入输出接口信号

域	位宽	功能描述
ib_ctl_uop_valid	1	输入控制信号有效标志，高电平有效
ib_ctl_uop_ready	1	控制单元可接收控制信号标志，高电平有效
ib_ctl_uop	8	输入控制信号，计算的神经元/权重数据的长度（单位：64 字节）
wram_mpe_weight_valid	1	输入权重有效标志，高电平有效
wram_mpe_weight_ready	1	控制单元可接收权重标志，高电平有效
wram_mpe_weight	512	输入权重数据，包括 32 个 INT16 分量
nram_mpe_neuron_valid	1	输入神经元有效标志，高电平有效
nram_mpe_neuron_ready	1	控制单元可接收神经元标志，高电平有效
nram_mpe_neuron	512	输入神经元数据，包括 32 个 INT16 分量
rst_n	1	输入复位信号，低电平有效
clk	1	输入时钟信号
result	32	输出内积结果
vld_o	1	输出内积结果有效标志，高电平有效

实验环境

- 工具
  - 使用 Verilog 语言实现的模块需要使用 HDL 仿真工具进行编译调试。本节以 Mentor 公司的 ModelSim 仿真工具为例，介绍实验环境和代码文件组织。

代码目录

- 目录src

  编写的矩阵运算子单元verilog源代码

- 目录sim

  顶层文件——tb_top.v

  仿真脚本文件

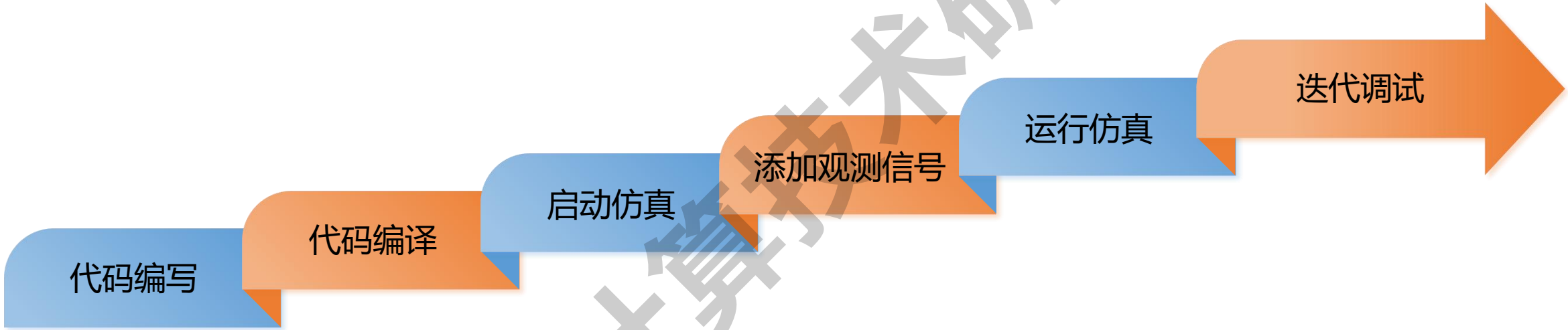
- 目录data

  仿真输入控制信号文件

  仿真输入神经元和权值文件

  仿真输出结果文件

实验步骤



编写仿真顶层文件——top_tb.v

- 生成激励信号
  - 时钟&复位信号
  - 控制信号
  - 输入神经元数据和权值数据
  - 输入输出握手信号
  - 结果比对
- 例化矩阵运算子单元

编写仿真顶层文件——读取输入文件

- $readmemb(“数据文件名”，存储器名);
- $readmemb(“数据文件名”，存储器名，起始地址);
- $readmemb(“数据文件名”，存储器名，起始地址，结束地址);
- $readmemh(“数据文件名”，存储器名);
- $readmemh(“数据文件名”，存储器名，起始地址);
- $readmemh(“数据文件名”，存储器名，起始地址，结束地址);

对于 $readmemb 系统任务，每个数字必须使用二进制表示，对于 $readmemh 系统任务，每个数字必须使用十六进制表示。

```
30 initial
31 begin
32     $readmemb("D:/pe_exp/data/inst", inst);
33     $readmemh("D:/pe_exp/data/neuron", neuron);
34     $readmemh("D:/pe_exp/data/weight", weight);
35 end
```

代码编译

指定需编译的顶层文件和源代码文件

```
1 // 源代码文件
2 path/to/src dir/module_0.v
3 path/to/src dir/module_1.v
4
5 // 顶层测试文件
6 path/to/sim dir/tb_top.v
```

代码编译

- 命令行窗口输入“do path to build/build.do”

```
# 设置仿真环境路径
# TODO
set sim_home Path/to/simulation/Dir

# 在当前目录下创建一个叫做work的目录，在里面存放仿真数据文件
vlib ${sim_home}/work

# 将work目录下的数据文件映射为一个叫做work的仿真库
vmap work ${sim_home}/work

# 编译compile.f文件中指定的代码
vlog -f ${sim_home}/compile.f
```

工具开始执行编译脚本来编译测试顶层文件和源代码

```
# -- Compiling module tb_top
# -- Compiling module fifo
# -- Compiling module mcpu_mult
# -- Compiling module mcpu_acc
# -- Compiling module int45_to_fp_stg1
# -- Compiling module int45_to_fp_stg2
# -- Compiling module int45_to_int16
# -- Compiling module mcpu_cvt
# -- Compiling module mcpu
#
# Top level modules:
#       tb_top
# End time: 20:13:52 on Feb 14,2020, Elapsed time: 0:
# Errors: 0, Warnings: 0
```

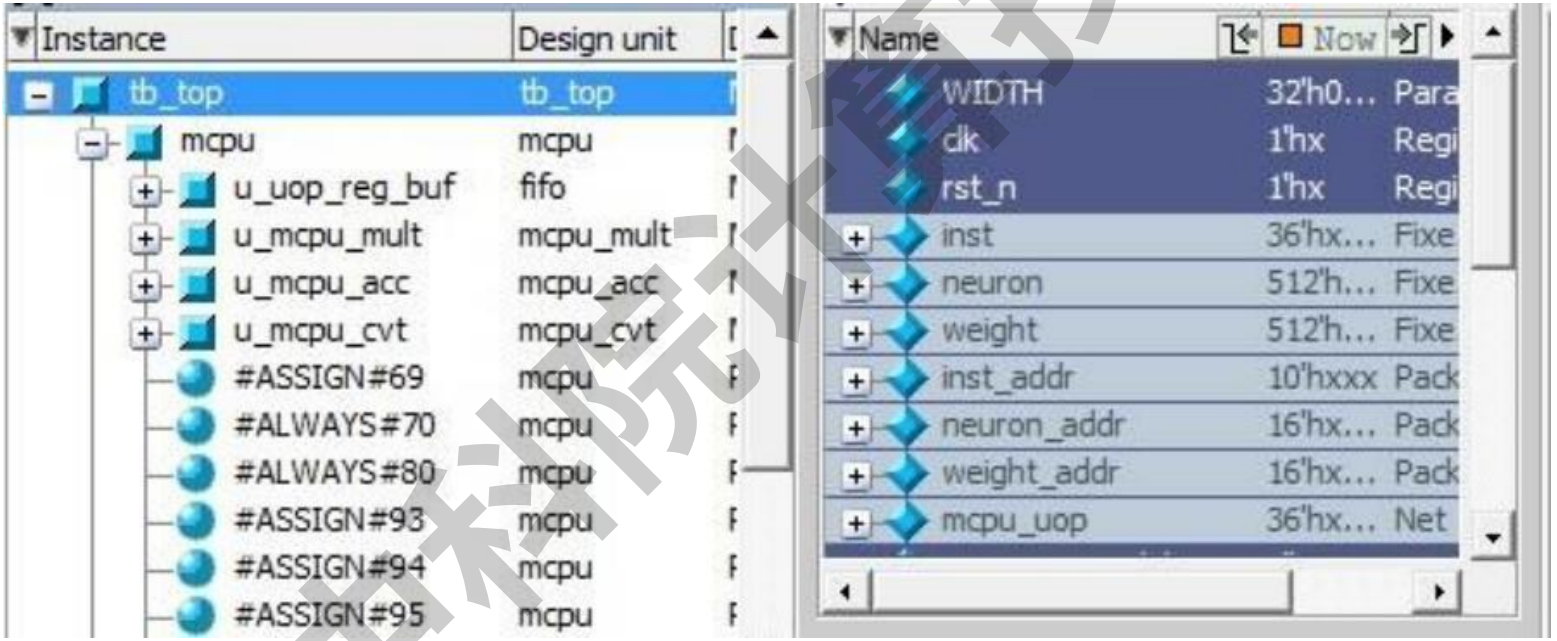
编译日志

启动仿真

- 命令行窗口输入“do path to build/sim_run.do”，运行一下执行脚本启动仿真

```
1 vsim +nowarnTSCALE -lib work -c -novopt tb_top
```

- Sim instance面板
  仿真模块层次和模块信号名称



添加观测信号

- sim instance面板中选择调试模块
  - object面板中选择观察信号
  - 单击右键“Add to -> Wave -> Selected Signals”将选中的信号添加到Wave窗口

运行仿真

- 命令行窗口输入“run -all”命令
- 仿真运行
- 仿真完成后通过波形窗口检查观察信号仿真的波形结果

迭代调试

- 运行“do path to build/build.do”重新编译
- 运行“restart”命令重新启动仿真
- 运行“run -all”命令，执行仿真观察结果
- 仿真完成后，使用“quit -sim”退出仿真

实验评估

- 60 分：完成串行内积运算器
- 80分：完成串行和并行内积运算器
- 100分：完成串行内积运算器、并行内积运算器和矩阵运算子单元

敬请指正！

课程官网：http://novel.ict.ac.cn/aics

MOOC网址：

https://space.bilibili.com/494117284/video