

练习：使用 HLS 生成 RTL 并达到更好的加速效果

目的：

- (1) 熟悉 Xilinx HLS 工具的使用，了解其流程。
- (2) 如何修改 C 代码结构，或者设置不同的 pragma，达到最佳的加速效果。

学到的内容：

该实验做完后，你将学到以下内容：

1. 以矩阵乘法为例，介绍了如何在 RTL 中进行硬件流水线设计。
2. 如何借助 xilinx HLS 工具完成高层次综合流程。
3. 如何在 Vitis HLS 中选择不同的指令，并了解它们对设计的影响。
4. 如何在 Vitis HLS 中进行 C-RTL 协同仿真，并完成实现。

矩阵乘算法简介：

在数学中，矩阵乘法或矩阵乘积是一种由两个矩阵生成矩阵的二进制运算。该定义的动机是线性方程和向量上的线性变换，它们的应用数学、物理和工程中有许多应用。更详细地说，如果 A 是 $n \times m$ 矩阵， B 是 $m \times p$ 矩阵，则它们的矩阵乘积 AB 是 $n \times p$ 矩阵，其中 a 行上的 m 个条目与 B 列下的 m 个条目的乘积并求和以产生 AB 的条目。当两个线性变换由矩阵表示时，矩阵乘积表示两个变换的组合。图 1 显示了矩阵乘法的运算。

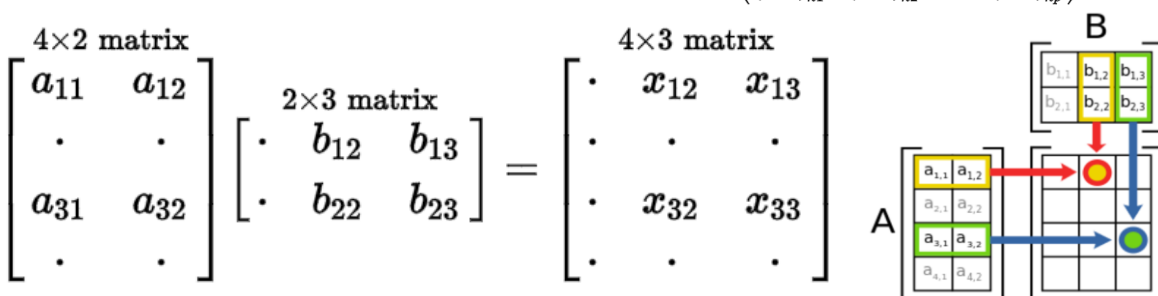
$$A = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1p} \\ B_{21} & B_{22} & \cdots & B_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & \cdots & B_{mp} \end{pmatrix} \quad AB = \begin{pmatrix} (AB)_{11} & (AB)_{12} & \cdots & (AB)_{1p} \\ (AB)_{21} & (AB)_{22} & \cdots & (AB)_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ (AB)_{n1} & (AB)_{n2} & \cdots & (AB)_{np} \end{pmatrix}$$


Figure 1

计算矩阵乘积既是许多数值算法的核心运算，又可能耗时，使其成为数值计算中研究最深入的问题之一。在机器学习领域，矩阵乘法也被广泛使用，例如在递归神经网络（RNN）中。

LAB:

在这个实验室练习中，你将熟悉 Xilinx Vitis HLS。包括三个 C 文件：（1）matmult.c、（2）tb_matmult.c 和（3）matmult.h。其中，文件 matmult.c 包含一个 C 函数，该函数从

软件角度实现矩阵乘法。文件 `tb_matmult.c` 包含调用 C 函数并打印其结果的主文件，而文件 `matmult.h` 是由用户定义的数据类型、数据宽度和函数声明组成的头文件。

需要具体做什么、展示什么：

- (i) 用 HLS 工具实现 RTL 硬件描述的输出. 展示生成的 Verilog 文件和 VHDL 文件，同时展示具体的 Latency 值、所用的资源数量.
- (ii) 探索至少 2 种加速的方式，比如通过不同的 `pragma`，或者通过修改原始 C 代码的编写风格. 展示加速后的 Latency 值和资源数量，并简要分析这样为何能得到更好加速效果的原因. (Tips: 循环优化? 接口优化?)

开放性题目（advanced）：

1. 你认为通过 `pragma` 对 C 代码进行小的更改很容易吗？通过二者生成的 Verilog/VHDL 相比，你能看到其在 RTL 上的差异，以及对设计性能的影响吗？？

需课后提交的实验报告（约 5 页）：

简单的 word 实验报告，内容含上面提到的内容。比如，生成的 Verilog/VHDL 文件截图，资源使用情况报告截图；怎样加速的，基于什么考虑，加速后的 Latency 和资源怎么变化等。开放性题目的解答也一并放入报告即可。

A note: 如果你有兴趣和额外的精力，可以通过阅读 Vivado HLS 教程（UG871）或用户指南（UG902）来了解如何 HLS 更多的使用技巧。这些文档在公网可以搜到。