面向大规模 MIMO 检测的矩阵乘法设计

本次实验最多 2 人一组

一、 实验背景

一个基站天线数为B，用户数为U的大规模 MIMO 系统，其输入输出关系可表征如下：

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}$$

其中，$\boldsymbol{x} \in \mathbb{C}^U$为$U$维发射信号向量，其中每个元素取自某调制方案(如QPSK/16-QAM)的归一化星座点集，$\boldsymbol{y} \in \mathbb{C}^B$为$B$维接收信号向量，$\boldsymbol{H} \in \mathbb{C}^{B\times U}$为信道矩阵，本次实验假设$\boldsymbol{H}$中的所有元素满足均值为 0，方差为 1 的独立复高斯分布，$\boldsymbol{n} \in \mathbb{C}^B$为方差为$N_0$的加性复高斯白噪声向量。这里所有矩阵、向量的元素均为复数。

大规模 MIMO 检测是指基站根据接收向量$\boldsymbol{y}$和信道估计结果$\boldsymbol{H}$，求出对发射信号向量$\boldsymbol{x}$的估计值$\hat{\boldsymbol{x}}$。我们当然希望$\hat{\boldsymbol{x}} = \boldsymbol{x}$，然而现实中，由于信道噪声，用户间干扰的存在，基站通常无法得到完全正确的$\hat{\boldsymbol{x}}$，只能通过各种检测算法尽可能降低$\hat{\boldsymbol{x}}$的误符号率。

在本次实验中，我们考虑一种能在大规模 MIMO 系统中实现近最优检测性能的算法——最小均方误差（Minimum Mean Square Error, MMSE）检测算法，其检测判决如下：

$$\hat{\boldsymbol{x}} = (\boldsymbol{H}^H\boldsymbol{H} + N_0\boldsymbol{I})^{-1}\boldsymbol{H}^H\boldsymbol{y}$$

其中$\boldsymbol{H}^H$表示对复数矩阵$\boldsymbol{H}$的共轭转置。对于大规模 MIMO 系统，MMSE 检测算法复杂度主要来源于**求取 Gram 矩阵$\boldsymbol{H}^H\boldsymbol{H}$**以及矩阵求逆运算。这里我们专注于前者的具体实现，即，**设计面向大规模 MIMO 检测的 Gram 矩阵乘法架构。**

二、 实验目标

本次实验将完成 Gram 矩阵求取的运算硬件设计，即计算$\boldsymbol{H}^H\boldsymbol{H}$。假设基站能够获得完美的信道矩阵估计$\boldsymbol{H}$，**请大家设计高吞吐，低延迟，高效率的矩阵乘法架构来求取 Gram 矩阵，并验证实现结果**。

本次实验中，信道矩阵的规模假设为64×8，即$B = 64, U = 8$。

三、 实验内容

3.1 软件部分

使用 MATLAB 或 C/C++实现 Gram 矩阵的求取，以 MATLAB 代码为例，信

道矩阵$\boldsymbol{H}$的生成方式如下：

```
H = sqrt(0.5) .* (randn(B, U) + 1i .* randn(B, U));
```

接着可算出 double 类型的 Gram 矩阵$\boldsymbol{G}_{double} = \boldsymbol{H}^H\boldsymbol{H}$：

```
G = H' * H;
```

本次实验要求输入和输出数据均量化至 16bits 位宽 (实部虚部各 16bits)，小数位宽自定。要求量化后的结果相比浮点结果相对误差小于 10% (越小越好)。相对误差定义如下：

$$\varepsilon = \frac{\left\|\boldsymbol{G}_{double} - \boldsymbol{G}_{fix}\right\|_F}{\left\|\boldsymbol{G}_{double}\right\|_F}$$

其中$\boldsymbol{G}_{fix}$为定点化后的输出 Gram 矩阵，$\|\cdot\|_F$为矩阵的 F 范数，定义为

$$\|\boldsymbol{A}\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n}\left|A_{ij}\right|^2\right)^{\frac{1}{2}}$$

即矩阵元素绝对值的平方和再开平方。

3.2 硬件部分

根据矩阵乘法运算法则完成硬件设计，要求**实现一种基于脉动阵列的架构，完成 Gram 矩阵计算；编写 Verilog 代码并利用 Vivado 或 Design Compiler 完成综合。**在功能验证方面，实验**要求 Verilog 的仿真结果与软件定点化结果逐 bit 一致，即 Verilog 的输出结果应和上述软件部分中得到的$\boldsymbol{G}_{fix}$中每个元素的 32bits 均一致 (尽量生成并验证多组软硬件结果，以保证硬件功能正确)**

这里我们定义架构的吞吐率如下：

$$Throughput = \frac{N}{T} \times f_{clk}$$

其中，N为在正常工作状态下，设计的架构在T周期内能得到的 Gram 矩阵个数，f_{clk}为时钟频率。我们还定义架构的 latency 为从输入有效开始，至开始输出第一个 Gram 矩阵元素信息的时钟周期数。

3.3 评分标准

在软件层面，将根据算法的设计思路与定点化精度进行评估；

在硬件层面，首先保证硬件的功能正确，然后我们将根据以下指标进行评估：

1) 若采用 Vivado 进行综合，FPGA 型号请选择 VC709 evaluation board，要求**时钟频率至少达到 150MHz**，latency 越短越好。另外我们定义硬件效率为

$$Hardware\ Efficiency = \frac{Throughput}{LUTs + FFs + DSP \times 280}$$

硬件效率越高越好；

2) 若采用 Design Compiler，工艺库请采用 0.18um，要求时钟频率至少达到 300MHz，latency 越短越好。定义 ASIC 面积效率如下：

$$Area\ Efficiency = \frac{Throughput}{Gate\ Counts}$$

其中$Gate\ Counts$等于综合后面积除以同工艺下的一个 NAND 门的面积。面积效率越高越好。

3) 加分项：除了脉动阵列架构外，如能额外实现另一种架构 (例如课堂上学习的流水线，折叠等)，则根据设计思路与上述评估标准加分。

四、 报告要求

在报告撰写方面，请将所有的设计亮点在报告中进行强调。报告内容至少包含以下：

a. 软硬件实现Gram矩阵乘法架构的描述；

b. 对于定点量化的考虑；

c. 硬件实现可行方案描述，请给出完整的硬件架构图，数据流图与时序图；

d. 分析硬件关键路径和硬件结构；

e. 主要功能模块的输入输出接口信号描述；

f. 功能仿真的测试向量以及验证结果；

g. 综合结果、关键路径（时序报告第一条路径）、面积或资源开销；

h. 性能整理，请整理图表 (非截图)，包括时钟频率，延时，吞吐，面积或资源等；

i. 有组队的同学请给出分工情况。

请将实验报告、软件代码、硬件代码打包成一个文件夹压缩后提交。

1) 文件夹和压缩包均以以下格式命名：final03_组号。``final03"表示选择的是本实验，组号为两位数字，个位数需要在前面补零，例如组号为3则命名为``final03_03"。
2) 文件夹分为3个部分：实验报告（一个PDF文件），软件代码（一个文件夹），硬件代码（一个文件夹）。
3) 实验报告须为pdf文件，要包括本实验中要求的各点（软件设计思路、硬件设计思路、电路图、验证结果图、硬件综合结果、成员分工等等）。报告开头写出成员名称和学号。
4) 软件代码需要给出必要的注释（输入输出的解释，以及必要的逻辑单元的注释）。
5) 硬件代码需要包括设计部分（电路实现）和验证部分（testbench），代码给

出必要注释（输入输出的解释，以及必要的逻辑单元的注释）。

6) 严禁抄袭。

五、 参考文献

这里给出了大规模 MIMO 检测硬件设计的相关论文，其中均包含了对 Gram 矩阵的运算架构设计 (多数为脉动阵列架构)，谨供大家参考。

[1] B. Yin, M. Wu, G. Wang, C. Dick, J. R. Cavallaro, and C. Studer, “A 3.8 Gb/s large-scale MIMO detector for 3GPP LTE-advanced,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Florence, Italy, May 2014, pp. 3879–3883.

[2] M. Wu, B. Yin, G. Wang, C. Dick, J. R. Cavallaro and C. Studer, "Large-Scale MIMO Detection for 3GPP LTE: Algorithms and FPGA Implementations," in *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 916-929, Oct. 2014.

[3] G. Peng, L. Liu, S. Zhou, Y. Xue, S. Yin, and S. Wei, “Algorithm and architecture of a low-complexity and high-parallelism preprocessing based K-best detector for large-scale MIMO systems,” *IEEE Trans. Signal Process.*, vol. 66, no. 7, pp. 1860–1875, Apr. 2018.

[4] Z. Wu, C. Zhang, Y. Xue, S. Xu, and X. You, “Efficient architecture for soft-output massive MIMO detection with Gauss-Seidel method,” in *Proc. IEEE Int. Symp. Circuits Syst.*, Montreal, QC, Canada, May 2016, pp. 1886–1889.

[5] L. Liu *et al.*, "Energy- and Area-Efficient Recursive-Conjugate-Gradient-Based MMSE Detector for Massive MIMO Systems," in *IEEE Transactions on Signal Processing*, vol. 68, pp. 573-588, 2020.

[6] J. Tu, M. Lou, J. Jiang, D. Shu and G. He, "An Efficient Massive MIMO Detector Based on Second-Order Richardson Iteration: From Algorithm to Flexible Architecture," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 11, pp. 4015-4028, Nov. 2020.

六、 附录

为了便于同学们设计，我们提供了关于 Gram 矩阵与复数乘法的一些性质：

6.1 共轭对称性

注意到 Gram 矩阵$\boldsymbol{G}$满足共轭对称性：

$$\boldsymbol{G}^H = (\boldsymbol{H}^H\boldsymbol{H})^H = \boldsymbol{H}^H\boldsymbol{H} = \boldsymbol{G},$$

即，$\forall 1 \le i \le k \le U$，我们有$G_{ik} = G_{ki}^*$。**因此我们仅需要计算 Gram 矩阵包含主对角线的上三角 (或下三角) 部分。**

对于 Gram 矩阵主对角线上的元素，我们有

$$G_{ii} = \sum_{k=1}^{B} |H_{ki}|^2 = \|\boldsymbol{h}_i\|^2$$

对于 Gram 矩阵上三角部分的元素，设$\forall 1 \le i < k \le U$，我们有

$$G_{ik} = \sum_{j=1}^{B} H_{ji}^* H_{jk}$$

6.2 Winograd 复数乘法器

对于两个复数$a + b\sqrt{-1}$和$c + d\sqrt{-1}$相乘，有

$$\left(a + b\sqrt{-1}\right) * \left(c + d\sqrt{-1}\right) = (ac - bd) + (ad + bc)\sqrt{-1}$$

根据上式，一个复数乘法包含了 4 个实数乘法，2 个实数加法。为了节省硬件开销，可用 Winograd 算法实现复数乘法，通过

$$ac - bd = a(c - d) + d(a - b)$$
$$ad + bc = b(c + d) + d(a - b)$$

从而仅需 3 个实数乘法与 5 个实数加法。即，**Winograd 复数乘法器将复数乘法从 4 乘 2 加转化为 3 乘 5 加。**

另外，为了方便同学们在软件方面的定点化设计，我们推荐 MATLAB 的定点化工具箱 Fixed Point Designer，能够对输入、输出、中间数据进行定点优化设计与误差分析，详细用法见 https://www.mathworks.com/help/fixedpoint/ug/manual-fixed-point-conversion-best-practices.html