Attention 模块 设计报告

张浩宇
522031910129

一、 模块设计

1.1 设计目标

设计一个模块进行 Self-Attention 计算，输入 Q、K、V 矩阵，得到输出结果，并进行量化。

模块参数设为：Token 数量为 8，Token 特征维度为 4，数据采用 Fix_8_8 即 8 位整数、8 位小数的定点无符号数格式存储和量化。

用 python 设计 golden model 用以验证，并用 verilog 完成硬件实现，要求结果仿真结果正确，代码可综合。

1.2 设计细节

1.2.1 Softmax 简化

标准的 Softmax 计算公式为：

$$\text{softmax}(A[i,j]) = \frac{\exp(A[i,j] - \max(A[:,j]))}{\sum_{i=1}^{N} \exp(A[i,j] - \max(A[:,j]))},$$

实现难度较高，因此采用以下简化的 Softmax 计算方法：

$$\text{softmax}(A[i,j]) = (A[i,j] - \min(A[:,j]))^2$$

1.2.2 数据量化

本设计采用Fix_8_8格式的16位定点数来存储数据，即8位整数、8位小数。但在Attention计算过程中，矩阵乘法和 Softmax 计算会导致数据位宽增大，因此在计算过程中需要将数据量化。

具体而言，在矩阵乘法中，经过乘累加运算，输出的数据为 Fix_18_16，需将其量化为 Fix_8_8；在 Softmax 计算中，经过平方运算，输出的数据为 Fix_16_16，需将其量化为 Fix_8_8。

量化过程包括两种情况。在小数位数不足以表达该数值的小数部分时，本设计采用舍入的方法，即根据超出部分的大小，若小于最小精度的一半，则舍去，否则进一位。在整数部分超出位数，发生溢出时，本设计采用饱和的方法，即近似到最大的正值。

1.3 Golden model 设计

用 python 设计该模块的 golden model，模拟计算中的矩阵乘法、softmax 和量化等过程，以验证硬件实现的正确性。为了编写方便，矩阵操作采用 pytorch 实现。

同时设计了 python 测试脚本，随机生成模块的输入数据，调用 modelsim 进行仿真，并将仿真结构与 golden model 计算结果比较，以验证硬件实现的正确性。

1.4 硬件模块设计

1.4.1 实现原理

Attention 计算可以分为三个步骤，分别是：矩阵乘法 $Q*K^T$、Softmax 计算、矩阵乘法 Score*V。本设计采用流水线结构实现，即每一个步骤作为流水线的一级，形成 3 级流水，3 个周期即可完成全部计算，如图 1。

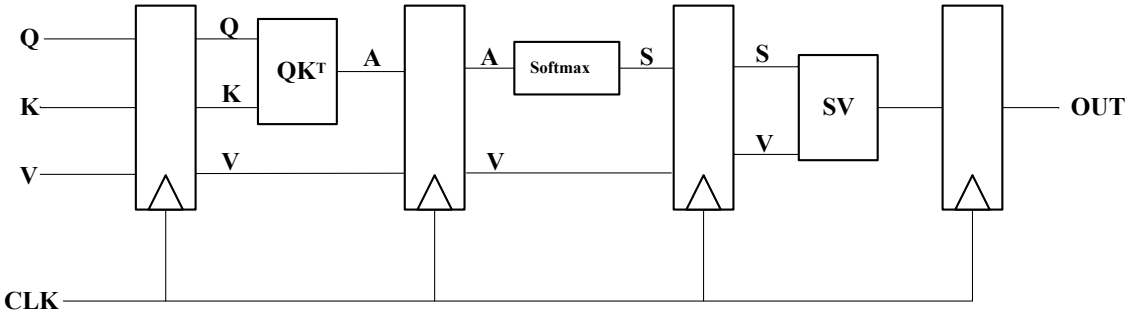


图 1　Attention 计算的流水线结构

1.4.2 模块总览

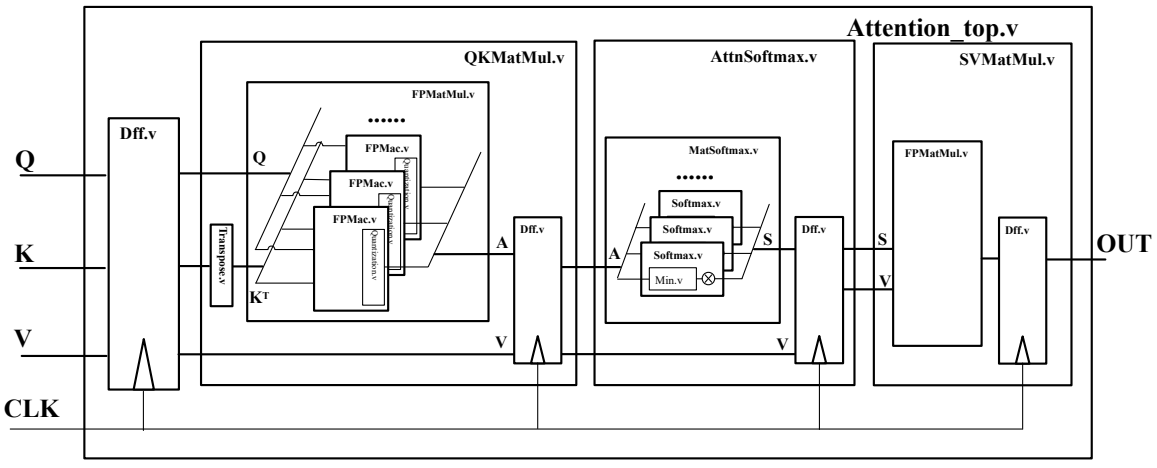


图 2　硬件模块设计

硬件模块设计示意如图 2，顶层模块 Attention_top 输入 Q、K、V 矩阵，和 clk、rst 信号，输出计算结果。在顶层模块中，每一级流水分别作为一级子模块。在第一级子模块中，包括矩阵转

置模块和矩阵乘法模块；在第二级子模块中，包括矩阵 Softmax 模块；在第三级子模块中，包括矩阵乘法模块。在输入和每一级流水结构中，均有寄存器，由 Dff 模块例化生成。

在模块设计中，各个子模块相关接口均采用参数化设计，便于模块复用与扩展。

下面介绍各模块及子模块的设计。

1.4.3 量化模块

量化模块实现对输入数据的量化输出，在矩阵乘法模块和 Softmax 模块中均有使用。该模块首先判断输入输出是否溢出，即高位超出量化位数的部分是否不位 0，若溢出则饱和，否则进行舍入；舍入的判断标准是超出最小精度的部分是否达到最小精度的一半，具体到实现中，通过判断低位超出量化位数部分的最高位是否为 1，若为 0 则舍去超出部分输出，否则进位输出，若进位后发生溢出则饱和输出。

1.4.4 矩阵乘法模块

矩阵乘法实现两个输入矩阵的乘法运算和结果的量化。该模块通过例化一系列带有量化的 MAC 模块实现，每个 MAC 模块的输入是两个输入矩阵的一行与一列，输出对应乘积矩阵的一个元素。

1.4.4.1 MAC（乘累加）模块

MAC 模块实现两个输入向量的乘累加运算，将两个输入向量每个元素相乘后累加，并将结果通过量化模块输出。为了减小 MAC 的关键路径，累加操作通过树状加法器实现。

1.4.5 转置模块

转置模块对输入矩阵进行转置，用于 $Q*K^T$ 的计算中。该模块通过对输入矩阵的元素地址进行变换来实现，即将按行顺序存储的输入矩阵按列读取得到输出转置矩阵。

1.4.6 Softmax 模块

Softmax 模块实现对输入矩阵的 Softmax 计算。该模块通过例化一系列带有量化的行 Softmax 模块，得到每一个元素按行 Softmax 的后的矩阵输出。

1.4.6.1 行 Softmax 模块

行 Softmax 模块实现对输入一行向量的 Softmax 结果。该模块具有子模块 Min，通过树状的逐个对比的方式得到输入向量的最小值，然后对每一个元素计算与最小值之差的平方，并通过量化模块，得到输出的 Softmax 结果。

1.4.7 寄存器模块

异步低电平有效复位的 Dff 模块，用于流水线中各级的寄存器。

1.4.8 Testbench

底层模块的 Testbench，通过读取 golden model 生成的测试数据作为输入，根据测试时给定的测试次数确定仿真周期数，生成对应的时钟，每个周期将将输出结果分别以二进制和十进制浮点数的形式输出到文件中，以供后续对比验证。

二、 设计结果

2.1 计算结果

根据测试脚本的结果，在多次随机输入的测试中，模块输出结果均与 golden model 计算结果一致，验证了硬件实现的正确性。

由于对 Softmax 计算进行了简化，Softmax 的结果并未归一化到 0~1 之间，较大的值经过 Softmax 后会被变得更大，容易导致大量结果溢出饱和，可令输入数据较小即仅使用部分的输入数据位宽来生成输入矩阵，以减少溢出的概率，从而提高测试结果的可读性。

2.2 时序

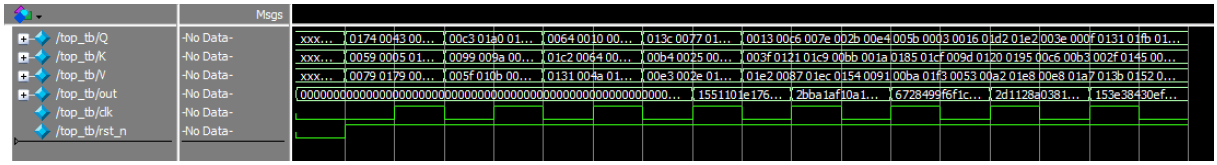
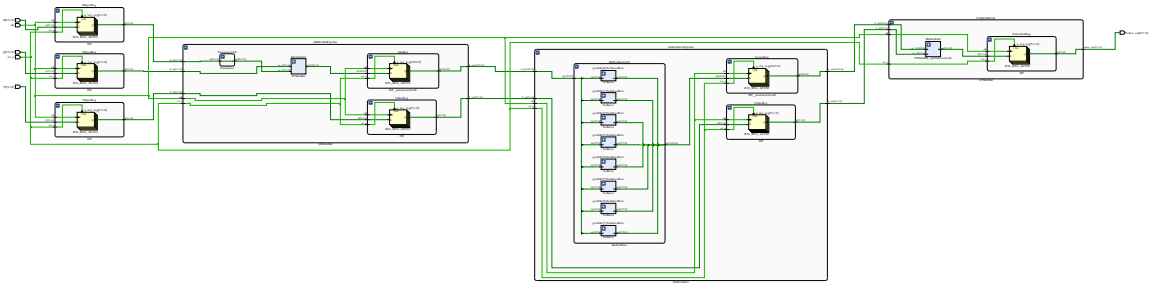


图 3 时序仿真结果

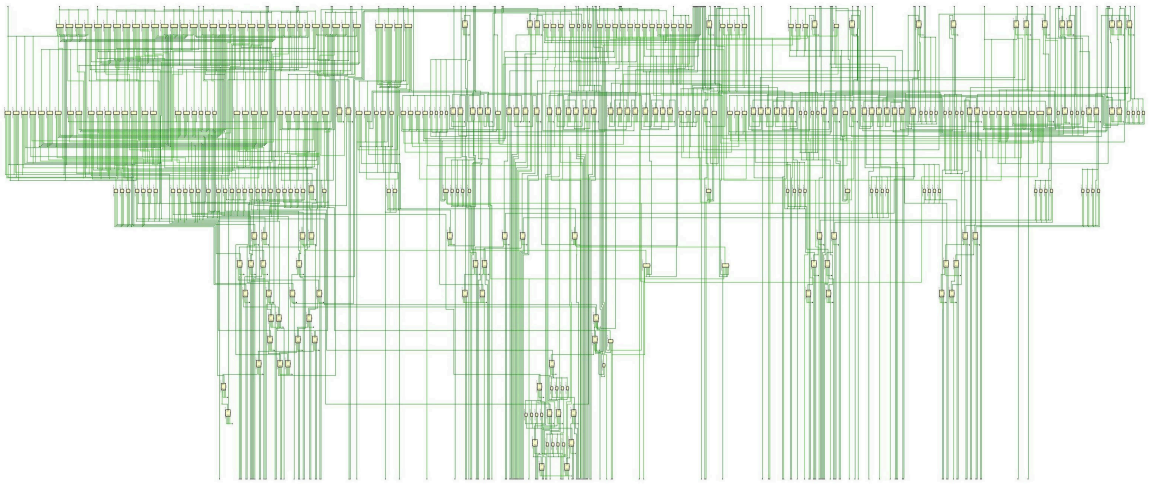
在一次进行五组输入测试的仿真过程中，时序仿真结果如图 3 所示。可以看到，数据输入三个时钟周期后，产生对应的输出结果，从开始输入数据后，经过三个周期的启动延迟，得到第一个输入对应的输出，此后每一个周期输出对应三个周期前输入的结果。可见流水线结构时序正确，符合设计。

2.3 综合

综合结果如图 4 所示。可见模块设计可综合，综合结果符合设计预期。



(a)Analysis 原理图



(b)Synthesis 原理图

图 4　综合结果

三、　讨论

3.1　数据流分析

3.1.1　关键路径

结合模块设计和综合结果

第一级流水包括转置和一个 8*4 矩阵和 4*8 矩阵的乘法，数据的最长路径为转置模块、一个 4 输入的 MAC 模块和量化模块。MAC 路径包括 1 个乘法器，加法器树中的 2 个加法器，如图 5。

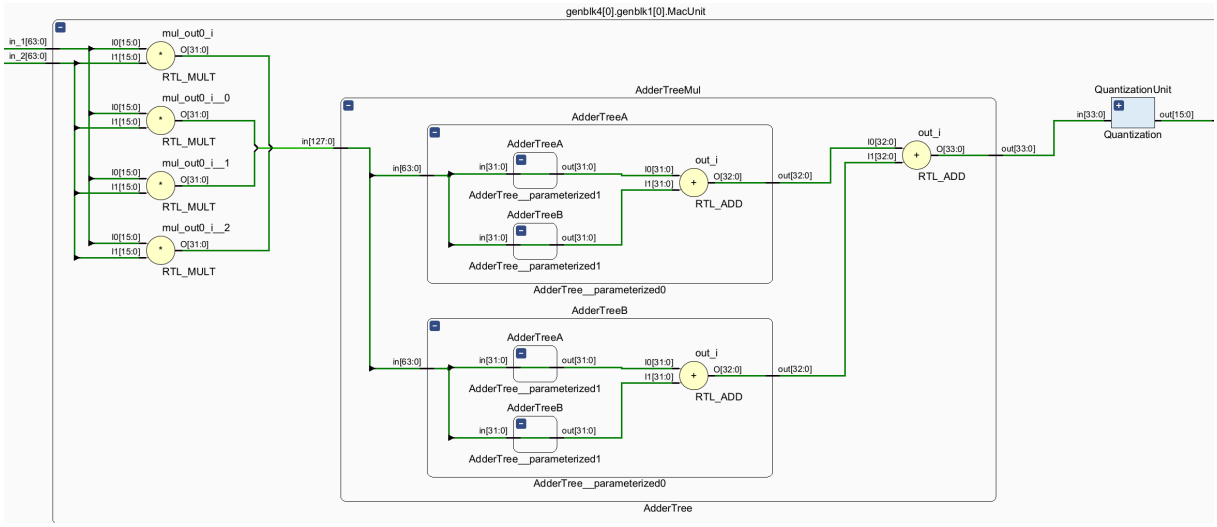


图 5　MAC 模块

第二级流水包括一个 8*8 矩阵的 Softmax 计算，数据的最长路径为一个最小值模块，一个加法（减法）器，一个乘法器和一个量化模块。最小值模块的比较器树包括 3 个比较选择器，如图 6。

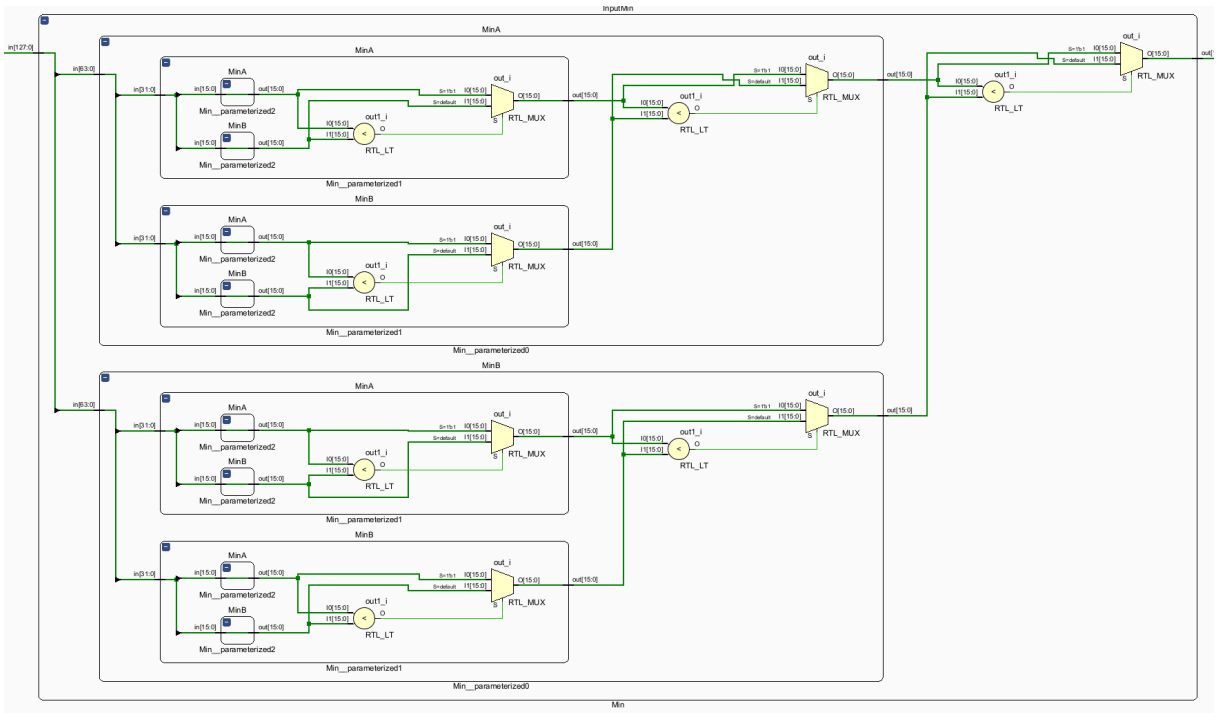


图 6　Min 模块

第三级流水包括一个 8*8 矩阵和 8*4 矩阵的乘法，数据的最长路径位一个 8 输入的 MAC 模块和量化模块。MAC 路径包括 1 个乘法器，加法器树中的 3 个加法器。

量化模块在各级流水线中相同，带来的延迟也相同，因此不影响关键路径。此处转置模块是简单的地址变换，连线直接相连，没有逻辑门带来的延迟。对比 MAC 部分的逻辑门数量，第三

级的延迟要大于第一级。对比第二级和第三级，抛去量化模块和两者都有的一个乘法器，第二级有一个加法器和三个比较选择器，第三级有三个加法器，且比较选择器包含一个 MUX 和一个比较器，第二级的逻辑门延迟应该更大。因此关键路径为第二级流水即 Softmax 操作部分。

若考虑面积开销，则是第一级最大，第三级次之，第二级最小。

3.1.2 延迟与吞吐率

延迟一个时钟周期？ 吞吐率为 8*4*16bit*3 每周期？

3.1.3 折叠设计

如果用折叠的方式实现该模块，可以用一个矩阵乘法器和相关的运算与控制组件和寄存器实现，如图图 7。

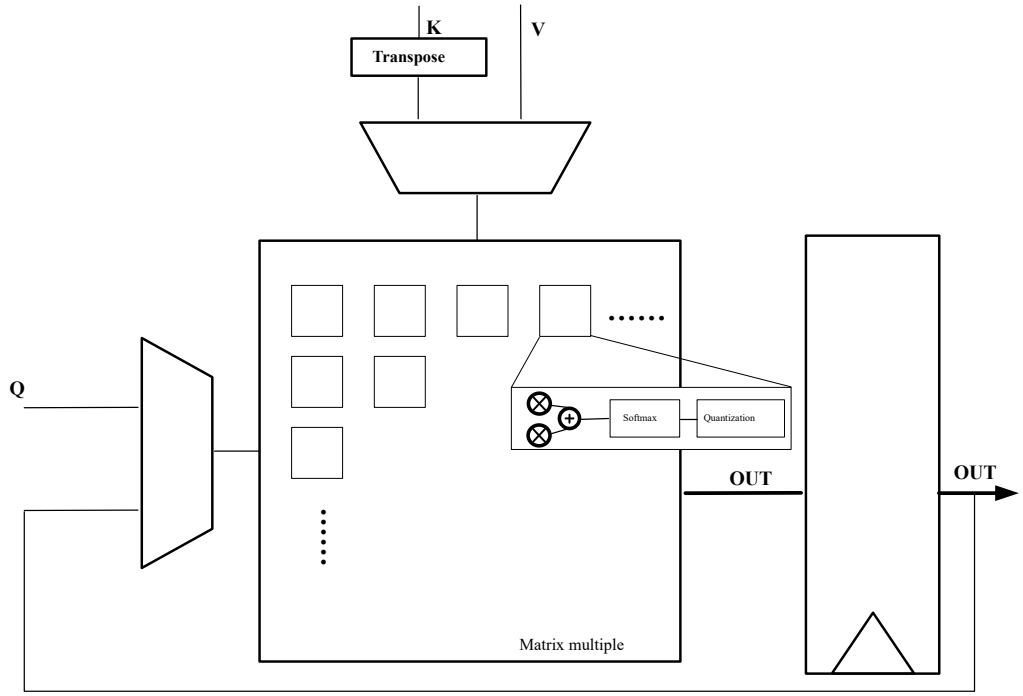


图 7 折叠设计

矩阵乘法器大小 8*8，由 MAC 组成，最多可以计算两个 8*8 矩阵的乘法，用于处理 $Q*K^T$ 和 Score*V 的计算，矩阵乘法每一位的输出带有量化与 Softmax 操作的模块，可选矩阵乘法后是否进行 Softmax 操作。并寄存器用于暂存数据，以及大量选择器用于控制数据流。

矩阵乘法器包含 64 个 8 输入 MAC，每个 MAC 包含 8 个乘法器和 7 个加法器，同时每一个输出的 Softmax 需要 1 个加法器（减法）和一个乘法器，因此总共需要 64*7+64*1=512 个加法器和 64*8+64*1=576 个乘法器。同时还需要 64 个量化模块和 8 个最小值模块。

为了暂存矩阵结果，需要 64*16bit 的寄存器。

3.2 量化中的舍入

python 的 round()函数与硬件四舍五入操作不同，为了让 golden model 与硬件计算尽可能一致，设计了符合硬件上的二进制舍入的函数用于量化。

硬件设计采用的二进制舍入的原理是检查超出量化位宽的最高位的值，若为 1 则进位，否则舍去，即对于超出精度的部分，若其小于最小精度的一半，则舍去，否则进一个最小精度的值。根据这个原理在 python 中对十进制浮点数操作，先得出其超出最小精度的整数倍的值，判读其是否大于最小精度的一半，若大于则进一个最小精度，否则舍去。