**测试向量**

为了有效考察FPGA硬件加速模块在bf16数据类型下对七种核心激活函数（Softmax, LayerNorm, RMSNorm, SiLU, GELU, Elementwise Add, Elementwise Multiply）的浮点计算精度和特殊值处理能力，需要设计一个全面且分层的测试策略。该策略应覆盖功能正确性、精度边界、特殊值场景、边界值场景以及实际数据分布模拟。

**一、 测试策略概述**

**目标：**

1. **精度验证：** 确保FPGA实现的bf16浮点运算结果与PyTorch bf16参考结果高度一致，满足评分标准中的 。
2. **特殊值处理：** 验证硬件模块能够正确处理bf16浮点数中的特殊值（+0.0, -0.0, +Inf, -Inf, NaN, 次正规数）。
3. **边界条件：** 测试输入数据在bf16表示范围的边界（最大/最小正规数、次正规数）时的行为。
4. **功能完备性：** 确保所有七种激活函数在各种有效输入下都能正确工作。
5. **批次处理：** 验证 $N \times D$ 维度张量（$N=64, D=768$）的正确处理，包括并行性和数据流。

**bf16 数据类型特性：**  
 bf16（bfloat16）具有1个符号位、8个指数位和7个尾数位。其范围与float32相似，但精度较低。在测试中需特别关注其舍入行为（通常为最近偶数舍入）、次正规数的处理（是否刷新为零或精确计算）以及特殊值（Inf, NaN）的传播规则。

**二、 测试向量分类与设计**

测试向量将分为以下几类，以确保全面覆盖：

**A. 基础功能正确性测试**

**目的：** 验证每个激活函数的核心计算逻辑是否按其数学定义正确实现。  
**通用原则：** 使用简单、可预测的输入，其输出结果易于手动或通过简单计算验证。

* **Softmax:**
* **全零输入：** [0.0, 0.0, ..., 0.0]。输出应为 [1/D, ..., 1/D]。
* **单元素为1，其余为0的行：** 例如 [0.0, 1.0, 0.0, ...]。验证指数归一化及“减最大值”防溢出逻辑。
* **所有元素相同的行：** 例如 [2.5, 2.5, ..., 2.5]。输出应为 [1/D, ..., 1/D]。
* **输入值差异极大的情况：** 例如 [0.0, 0.0, 10.0, 0.0] 或 [100.0, 101.0, 102.0]。考验 exp 计算和减最大值操作的稳定性，确保输出和为1。
* **LayerNorm/RMSNorm:**
* **均值为0、方差为1的输入：** 例如 [1.0, -1.0, 1.0, -1.0, ...]。
* **所有元素相同的行：** 例如 [5.0, 5.0, ..., 5.0]。验证均值/平方和计算精度，此时方差/均方根应接近0或为0，考验 epsilon 对除以零的保护作用。
* **零均值/均方根接近零的输入：** 例如 [0.0, 0.0, ..., 0.0]。
* **SiLU:**
* **输入为0：** SiLU(0) = 0。
* **输入为1.0：** SiLU(1.0) = 1.0 \* sigmoid(1.0) ≈ 0.731。
* **输入为-5.0：** SiLU(-5.0) = -5.0 \* sigmoid(-5.0) ≈ -0.0335。
* **GELU:**
* **输入为0：** GELU(0) = 0。
* **输入为1.0：** GELU(1.0) ≈ 0.841。
* **输入为-3.0：** GELU(-3.0) ≈ -0.004。
* **Elementwise Add/Mul:**
* **Add：** 输入1为全1，输入2为全2。结果应为全3。
* **Mul：** 输入1为全1，输入2为全2。结果应为全2。
* **Mul：** 输入之一为全0。结果应为全0。

**B. 精度敏感区域测试**

**目的：** 验证在函数行为变化剧烈、容易引入精度损失的区域，硬件模块仍能保持高精度。

* **已知解析解的输入：** 对于某些函数，在特定输入下有精确的数学结果（例如，SiLU在x=0时输出0，LayerNorm在所有输入相等时输出0）。使用这些点验证精度。
* **敏感区域密集采样：**
* **Softmax：** 输入值接近但略有不同，例如 [1.000, 1.001, 1.002]。
* **LayerNorm/RMSNorm：** 方差/均方根非常小但非零，例如输入值集中在一个很小的范围内。
* **SiLU/GELU：** 输入接近0的区域（例如 [-0.1, 0.1] 范围内密集采样），因为这些函数的导数在0附近变化较大。
* **Add/Mul：** 大数加小数（例如 1000.0 + 0.001），正负数相加结果接近零（例如 1.00001 + (-1.0)），两个很小的数相乘（可能导致下溢）。
* **随机输入 (大规模精度验证)：**
* 生成大量在不同数值范围（例如 [-10.0, 10.0], [-100.0, 100.0], [-1.0, 1.0]）均匀分布的随机bf16数作为输入。
* 与PyTorch bf16参考结果进行比较，统计误差分布，确保满足 $10^{-3}$ 的无损精度阈值要求。

**C. 特殊值与边界值测试**

**目的：** 确保硬件模块能正确处理bf16浮点数的各种特殊情况和表示极限，遵循IEEE 754（或bf16等效）标准。

* **特殊值输入：**
* +Inf (正无穷大)
* -Inf (负无穷大)
* NaN (非数字，包括静默NaN和信号NaN，验证其传播行为)
* +0.0 和 -0.0 (正零和负零)
* 次正规数 (Subnormal/Denormalized Numbers)：例如 1e-40 转换为bf16后的值。
* **边界值输入：**
* 最大正数 (BF16\_MAX)
* 最小正正规数 (BF16\_MIN\_NORMAL)
* 最大负数 (-BF16\_MAX)
* 最小负正规数 (-BF16\_MIN\_NORMAL)
* 次正规数的最大值和最小值。
* **混合输入：** 创建包含正常数、特殊值和边界值混合的张量，以测试运算单元在复杂情况下的行为。
* **按函数细化特殊值处理：**
* **Softmax：** 包含 +Inf（一个或多个）、-Inf、NaN 的输入。例如 [0.0, Inf, 0.0], [Inf, Inf, 0.0], [NaN, 1.0, 2.0]。
* **LayerNorm/RMSNorm：** 包含 Inf、NaN 的输入，验证均值/方差/均方根计算中 NaN 的传播。
* **SiLU/GELU：** 包含 Inf、-Inf、NaN 的输入。
* **Add/Mul：**
* Inf + (-Inf) (应为 NaN)
* 0 \* Inf (应为 NaN)
* x + NaN = NaN, x \* NaN = NaN
* 两个非常大的数相加/相乘（验证溢出）。
* 两个非常小的数相乘（验证下溢或次正规数）。

**D. 实际分布模拟测试**

**目的：** 模拟大语言模型推理中可能出现的实际输入数据分布，验证硬件在实际工作负载下的表现。

* **真实数据采样：** 如果可能，从实际的大语言模型推理过程中捕获中间激活值作为测试输入。
* **统计分布模拟：** 生成符合特定统计分布（如正态分布、均匀分布、截断正态分布等）的随机数，调整其均值和方差以模拟不同Transformer层（如注意力输出、FFN输出）的激活输出特征。

根据您提出的测试策略和 N=64, D=768 的固定输入张量尺寸，我们可以设计一个单一的 64 x 768 的 bf16 输入张量（记为 X\_test\_tensor），其中每一行 X\_test\_tensor[i, :] 代表一个特定的测试用例或测试场景。对于像 Elementwise Add 和 Elementwise Multiply 这样的双输入操作，X\_test\_tensor 将作为第一个操作数，而测试脚本将负责生成相应的第二个操作数 Y。

以下是 X\_test\_tensor 的详细填充策略，旨在覆盖功能正确性、精度边界、特殊值场景和实际分布模拟：

**X\_test\_tensor (64 x 768, bf16) 填充策略**

我们将 X\_test\_tensor 的 64 行分为以下几类，并详细说明每行的数据内容。

1. 通用 BF16 特性与常见值测试 (12 行)

这类测试旨在验证浮点运算单元对基本数值和特殊值的处理能力，适用于所有激活函数。

* **行 0: 全零**
* 内容：所有元素均为 +0.0。
* 目的：验证零输入下的行为。
* **行 1: 全一**
* 内容：所有元素均为 1.0。
* 目的：验证单位输入下的行为。
* **行 2: 全负一**
* 内容：所有元素均为 -1.0。
* 目的：验证负单位输入下的行为。
* **行 3: 全 BF16\_MAX**
* 内容：所有元素均为 bf16 最大正数。
* 目的：验证最大值输入下的行为，考察溢出边界。
* **行 4: 全 -BF16\_MAX**
* 内容：所有元素均为 bf16 最大负数。
* 目的：验证最小值输入下的行为，考察溢出边界。
* **行 5: 全 BF16\_MIN\_NORMAL**
* 内容：所有元素均为 bf16 最小正规数。
* 目的：验证小正规数输入下的行为。
* **行 6: 全 -BF16\_MIN\_NORMAL**
* 内容：所有元素均为 bf16 最小负正规数。
* 目的：验证小负正规数输入下的行为。
* **行 7: 全正次正规数**
* 内容：所有元素均为一个较大的正次正规数（例如，float32 的 1e-30 转换为 bf16 后）。
* 目的：验证次正规数处理。
* **行 8: 全负次正规数**
* 内容：所有元素均为一个较大的负次正规数。
* 目的：验证次正规数处理。
* **行 9: 混合正常值**
* 内容：由 [1.0, -1.0, 0.5, -0.5, 2.0, -2.0, 0.1, -0.1, 10.0, -10.0] 等正常值组成的重复模式。
* 目的：验证混合值的通用计算。
* **行 10: 混合特殊值**
* 内容：由 [+Inf, -Inf, NaN, +0.0, -0.0, 1.0, -1.0] 等特殊值和正常值组成的重复模式。
* 目的：验证特殊值的传播和交互。
* **行 11: 交替 +0.0 和 -0.0**
* 内容：偶数索引为 +0.0，奇数索引为 -0.0。
* 目的：验证带符号零的处理。

1. 随机分布输入测试 (8 行)

这类测试模拟实际数据分布，并验证硬件在不同数值范围内的精度和稳定性。

* **行 12: 均匀随机 [-10.0, 10.0]**
* 内容：在 [-10.0, 10.0] 范围内均匀分布的随机数。
* 目的：常用激活函数输入范围。
* **行 13: 均匀随机 [-1000.0, 1000.0]**
* 内容：在 [-1000.0, 1000.0] 范围内均匀分布的随机数。
* 目的：较大数值范围的测试。
* **行 14: 均匀随机 [-0.1, 0.1]**
* 内容：在 [-0.1, 0.1] 范围内均匀分布的随机数。
* 目的：对 SiLU 和 GELU 在零点附近精度敏感。
* **行 15: 均匀随机 [0.0, 1.0]**
* 内容：在 [0.0, 1.0] 范围内均匀分布的随机数。
* 目的：全正数输入测试。
* **行 16: 正态分布 (均值0, 标准差5)**
* 内容：服从标准正态分布的随机数，乘以 5。
* 目的：模拟常见神经网络权重或激活的分布。
* **行 17: 随机值注入 Inf/NaN**
* 内容：大部分为 [-5.0, 5.0] 范围内的随机数，但在几个特定位置注入 +Inf 和 NaN。
* 目的：验证随机数据流中特殊值的处理。
* **行 18: 随机值注入次正规数**
* 内容：大部分为 [-5.0, 5.0] 范围内的随机数，但在几个特定位置注入非常小的数（转换为 bf16 后为次正规数）。
* 目的：验证随机数据流中次正规数的处理。
* **行 19: 随机大正数**
* 内容：在 [0.0, BF16\_MAX/2] 范围内均匀分布的随机数。
* 目的：验证大数值下的精度和稳定性。

1. 激活函数特定模式测试 (34 行)

这类测试针对每个激活函数的数学特性和潜在的数值挑战进行设计。

* **Softmax (6 行)**
* **行 20: 大正值递增序列**
* 内容：例如 [100.0, 100.1, 100.2, ..., 100.0 + D \* 0.1]。
* 目的：验证 exp() 计算的稳定性，特别是 max\_k 归一化防溢出机制。
* **行 21: 大负值递减序列**
* 内容：例如 [-100.0, -100.1, -100.2, ..., -100.0 - D \* 0.1]。
* 目的：验证 exp() 下溢至零的行为。
* **行 22: 单个元素远大于其他**
* 内容：中间某个元素为 10.0，其余为 0.0。
* 目的：验证输出集中在优势元素位置。
* **行 23: 单个 +Inf 元素**
* 内容：中间某个元素为 +Inf，其余为 0.0。
* 目的：验证 +Inf 导致输出为 1.0。
* **行 24: 两个 +Inf 元素**
* 内容：两个元素为 +Inf，其余为 0.0。
* 目的：验证多个 +Inf 导致输出均匀分布。
* **行 25: 单个 NaN 元素**
* 内容：中间某个元素为 NaN，其余为 0.0。
* 目的：验证 NaN 传播。
* **LayerNorm (6 行)**
* **行 26: 零均值、非零方差**
* 内容：例如 [1.0, -1.0, 1.0, -1.0, ...]。
* 目的：验证均值和方差计算。
* **行 27: 值接近常数 (小方差)**
* 内容：例如 [5.000, 5.001, 5.002, ...]。
* 目的：验证 epsilon 对除以零的保护作用，以及小方差下的精度。
* **行 28: 包含 Inf/NaN**
* 内容：随机数中注入 +Inf 和 NaN。
* 目的：验证均值和方差计算中特殊值的传播。
* **行 29: 所有值均相差 epsilon**
* 内容：例如 [0.0, EPSILON\_LN, 2\*EPSILON\_LN, ...]。
* 目的：精确测试 epsilon 边界效应。
* **行 30: 大数值**
* 内容：在 [0.0, BF16\_MAX/10] 范围内随机数。
* 目的：验证大数值下的数值稳定性。
* **行 31: 小数值 (非次正规)**
* 内容：在 [BF16\_MIN\_NORMAL, BF16\_MIN\_NORMAL\*10] 范围内随机数。
* 目的：验证小数值下的数值稳定性。
* **RMSNorm (6 行)**
* **行 32: 零均值、非零均方根**
* 内容：例如 [1.0, -1.0, 1.0, -1.0, ...]。
* 目的：验证均方根计算。
* **行 33: 值接近零 (小均方根)**
* 内容：例如 [0.000, 0.001, 0.002, ...]。
* 目的：验证 epsilon 对除以零的保护作用，以及小均方根下的精度。
* **行 34: 包含 Inf/NaN**
* 内容：随机数中注入 +Inf 和 NaN。
* 目的：验证均方根计算中特殊值的传播。
* **行 35: 所有值均相差 epsilon (从零开始)**
* 内容：例如 [0.0, EPSILON\_RMS, 2\*EPSILON\_RMS, ...]。
* 目的：精确测试 epsilon 边界效应。
* **行 36: 大数值**
* 内容：在 [0.0, BF16\_MAX/10] 范围内随机数。
* 目的：验证大数值下的数值稳定性。
* **行 37: 小数值 (非次正规)**
* 内容：在 [BF16\_MIN\_NORMAL, BF16\_MIN\_NORMAL\*10] 范围内随机数。
* 目的：验证小数值下的数值稳定性。
* **SiLU (4 行)**
* **行 38: 零点附近密集采样**
* 内容：例如 [-5.0, ..., 0.0, ..., 5.0] 的线性序列。
* 目的：验证 sigmoid 近似在零点附近的精度和行为。
* **行 39: 大正值**
* 内容：例如 [10.0, ..., 100.0] 的线性序列。
* 目的：验证 SiLU(x) 趋近于 x。
* **行 40: 大负值**
* 内容：例如 [-10.0, ..., -100.0] 的线性序列。
* 目的：验证 SiLU(x) 趋近于 0。
* **行 41: 包含 Inf/-Inf/NaN**
* 内容：随机数中注入 +Inf, -Inf, NaN。
* 目的：验证 sigmoid 和乘法操作对特殊值的处理。
* **GELU (4 行)**
* **行 42: 零点附近密集采样**
* 内容：例如 [-5.0, ..., 0.0, ..., 5.0] 的线性序列。
* 目的：验证 erf 近似在零点附近的精度和行为。
* **行 43: 大正值**
* 内容：例如 [10.0, ..., 100.0] 的线性序列。
* 目的：验证 GELU(x) 趋近于 x。
* **行 44: 大负值**
* 内容：例如 [-10.0, ..., -100.0] 的线性序列。
* 目的：验证 GELU(x) 趋近于 0。
* **行 45: 包含 Inf/-Inf/NaN**
* 内容：随机数中注入 +Inf, -Inf, NaN。
* 目的：验证 erf 近似和乘法操作对特殊值的处理。
* **Elementwise Add/Multiply (8 行 - 作为操作数 X)**
* 这些行作为 Elementwise Add 和 Elementwise Multiply 的第一个输入 X。测试脚本将生成相应的第二个输入 Y 来形成完整的测试用例。
* **行 46: 随机正数**
* 内容：在 [0.0, 10.0] 范围内均匀分布的随机数。
* 目的：通用加法/乘法测试。
* **行 47: 随机负数**
* 内容：在 [-10.0, 0.0] 范围内均匀分布的随机数。
* 目的：通用加法/乘法测试。
* **行 48: 大数值**
* 内容：在 [BF16\_MAX/2, BF16\_MAX] 范围内均匀分布的随机数。
* 目的：与 Y 配合测试溢出。
* **行 49: 小数值**
* 内容：在 [BF16\_MIN\_NORMAL, BF16\_MIN\_NORMAL\*10] 范围内均匀分布的随机数。
* 目的：与 Y 配合测试下溢。
* **行 50: 包含 +Inf**
* 内容：随机数中注入 +Inf。
* 目的：与 Y 配合测试 Inf + (-Inf) 或 0 \* Inf 等。
* **行 51: 包含 -Inf**
* 内容：随机数中注入 -Inf。
* 目的：与 Y 配合测试 Inf + (-Inf) 等。
* **行 52: 包含 NaN**
* 内容：随机数中注入 NaN。
* 目的：与 Y 配合测试 NaN 传播。
* **行 53: 包含 0.0**
* 内容：随机数中注入 0.0。
* 目的：与 Y 配合测试 0 \* Inf 等。

1. 剩余行 (10 行)

* **行 54-63: 更多随机或复杂模式**
* 内容：可以使用不同分布（如对数正态分布）、不同数值范围的随机数填充，或重复一些现有测试用例但使用不同的随机种子，以增加测试的广度。也可以设计一些“病态”输入，例如交替的极大值和极小值，以测试流水线和资源复用中的潜在问题。

**测试脚本的职责**

在实际测试过程中，您的Python测试脚本将：

1. **生成 X\_test\_tensor：** 按照上述策略填充 64 x 768 的 bf16 张量。
2. **生成 Y\_test\_tensor (针对 Add/Mul)：** 为 Elementwise Add 和 Elementwise Multiply 操作生成一个与 X\_test\_tensor 结构类似的 Y\_test\_tensor。Y\_test\_tensor 的每一行应与 X\_test\_tensor 中对应的行配合，以形成有意义的测试对（例如，X 为大正数，Y 为大正数测试溢出；X 为 +Inf，Y 为 -Inf 测试 NaN 结果）。
3. **生成黄金参考输出：** 使用 PyTorch 的 bfloat16 数据类型，对 X\_test\_tensor 的每一行（以及 Y\_test\_tensor 的对应行）应用相应的激活函数，计算出黄金参考输出 Y\_ref。
4. **数据传输：** 将 X\_test\_tensor (以及 Y\_test\_tensor 的相关行) 的 bf16 位模式打包成 AXI Memory-Mapped 接口所需的格式，并传输到 FPGA。
5. **FPGA 结果读取：** 从 FPGA 读取计算结果 Y\_fpga。
6. **结果比较：** 将 Y\_fpga 转换为 bfloat16 浮点数，并与 Y\_ref 进行比较，计算相对 L2 误差 epsilon\_f，并根据评分标准评估精度得分 A\_f。同时，对特殊值进行逐元素精确比较。

通过这种方式，可以在一个固定大小的输入张量中，全面而有效地测试 FPGA 硬件加速模块的浮点计算精度和特殊值处理能力。