HFRWKV：面向 RWKV 的高性能全片上硬件加速器

Shijie Liu 中山大学 中国广州

Zhenghao Zeng 中山大学 中国广州

Han Jiao 中山大学 中国广州

Yihua Huang*
中山大学 中国广州

arXiv:2601.02135v1 [cs.AR] 5 Jan 2026

摘要

RWKV 是一种现代循环神经网络架构，性能接近 Transformer，同时能够以线性内存成本处理长上下文。然而，其顺序计算模式难以有效利用 GPU 并行性，导致计算资源利用率较低。此外，频繁的片外权重访问造成了内存瓶颈。为应对这些挑战，我们提出了 HFRWKV，这是一种专为 RWKV 设计的基于现场可编程门阵列的硬件加速器。在矩阵运算模块中，我们提出了一种新颖的硬件友好型混合精度量化策略，在提升性能的同时保持可接受的准确率。针对指数运算和除法等复杂运算，我们提出了一种结合可复用架构与查找表或分段线性近似的方法，并对算法进行改进，从而有效平衡精度与硬件资源消耗。在此基础上，我们采用了一个全片上计算系统，集成并行矩阵-向量处理阵列和高效的流水线架构。通过计算重排序和分块双缓冲，该系统有效消除了数据传输瓶颈，并提升了整体吞吐量。我们在 Alveo U50 和 U280 平台上实现了 HFRWKV。实验结果表明，与 CPU 相比，吞吐量提升了 63.48×，能效提升了 139.17×。与 GPU 相比，吞吐量提升了 32.33×，能效提升了 171.36×。

关键词

RWKV、混合精度量化、全片上计算系统、硬件加速器

1 引言

近年来，大语言模型（LLMs）已成为现代人工智能的基石，并广泛应用于自然语言理解与生成、科学探索等多个领域 [1]。尽管 Transformer[2, 3]，是大多数最先进大语言模型的主流架构，但其自注意力机制的计算和内存复杂度呈平方增长，严重限制了其在长序列处理和资源受限设备上的应用。

为克服 Transformer架构的性能局限，学术界和工业界都在积极探索计算复杂度更低的新型模型架构[4]。在此背景下，感受加权键值模型（RWKV）[5]作为一种颇具前景的新型算法框架应运而生，它巧妙地结合了 Transformer 的可并行训练能力与循环神经网络的线性推理效率。这种设计使 RWKV 的性能可与主流 Transformer模型相媲美，同时凭借推理阶段的线性复杂度具备显著的效率优势。这种出色的性能与效率平衡引起了广泛关注。然而，在硬件层面实现 RWKV 仍面临诸多挑战。

(1) 在推理过程中，RWKV 以类RNN模式运行，时间步之间存在强烈的顺序依赖，因为第 t 步的状态计算必须等待第 t-1 步的结果。尽管流水线等 GPU并行性技术可以使不同层之间的操作重叠，但每一层中的核心循环状态更新仍然是不可避免的顺序瓶颈。这种不匹配会导致资源利用率不理想，在对延迟敏感的低批量场景中尤为明显，因此需要专用硬件架构。

（2）由于受内存带宽限制的逐元素运算和缺乏专用硬件支持的复杂非线性函数，RWKV无法充分利用针对乘加操作优化的Tensor核心，因此在现代GPU上效率低下，并导致资源严重闲置。

（3）RWKV中的层归一化[6] 虽然能够稳定训练，但由于受带宽限制，成为性能瓶颈。其全局统计量计算需要在GPU上执行多步归约过程：首先将并行计算的局部统计量写入全局DRAM，再由单独的内核进行聚合，最后读回结果。这种密集的内存流量使该操作的速度取决于内存带宽，而非GPU的计算能力。

（4）RWKV包含多个串行计算模块，使其数据流非常适合在FPGA上采用高度定制化的流水线架构。与GPU固定的并行架构相比，FPGA的可重构性能够根据RWKV模型结构设计定制的数据通路，从而最大限度地减少数据移动和计算延迟。然而，学术界和工业界针对如何系统地设计和实现专用于RWKV的基于FPGA的硬件加速，相关研究仍然不足，这表明该领域存在研究空白。

为了解决上述算法与硬件之间的不匹配问题，并充分释放RWKV架构的理论潜力，本文提出了硬件友好型RWKV（HFRWKV），一种全片上硬件加速器系统。我们的贡献概括如下：

（1）我们提出了一种新颖且硬件友好的RWKV混合精度量化策略。针对大型矩阵权重，我们引入了一种创新的 Δ-PoT 量化方法。对于所有激活值和加性权重，我们均采用9位均匀对称量化。

(2)基于提出的 Δ-PoT 量化方案，我们设计了一种可复用的计算架构，将矩阵-向量乘法与逐元素乘法深度融合。该架构以

*通讯作者。电子邮箱：huangyih@mail.sysu.edu.cn。

移位器和加法器为核心，不仅能高效执行 Δ-PoT 量化格式下的乘法，还支持逐元素加法。

（3）为加速复杂函数（例如除法、Sigmoid），我们采用分段线性近似与查找表相结合的方法。该方法以快速、简单的线性片段和查找操作替代计算密集型运算，在保持精度的同时提升速度并降低硬件成本。

（4）我们提出了一种完全集成于片上的架构，整合了矩阵-向量、复杂运算和层归一化模块。该架构在 Alveo U50/U280 平台上实现，其完全流水线化且高度并行的设计实现了高吞吐量和高能效。与 CPU 相比，我们的加速器将吞吐量提高了 63.48×，能效提高了 139.17×。与 GPU 相比，其吞吐量提高了 32.33×，能效提升了 171.36×。

2 背景

2.1 RWKV模型架构

RWKV 将 Transformer 架构可并行训练的优势与循环神经网络的 $O(N)$ 线性推理复杂度特性相结合，在保持高性能的同时，显著提升了处理长序列的效率。如图 1 所示，RWKV架构本质上是线性注意力机制[7]的一种变体。它由一个初始嵌入层、N 个相同的残差RWKV块组成的堆栈，以及一个用于生成词元预测逻辑值的最终输出头层构成。每个 RWKV块依次由时间混合模块和通道混合模块组成，每个模块前都设有各自的层归一化和残差连接。

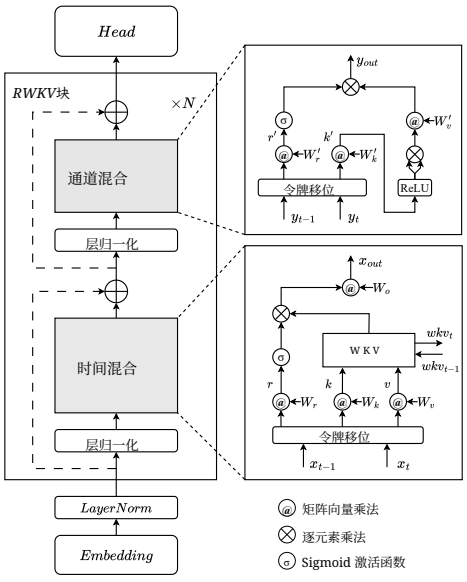

图 1：RWKV计算流程，包括时间混合模块和通道混合模块。

每个 RWKV块中的运算由两项关键计算定义。首先，词元移位机制会在当前时间步 x_t 和前一时间步 x_{t-1}之间进行线性插值，为各模块中的线性投影生成输入向量。词元移位运算可用以下公式表示:

(1).

$$y_t^{(\lambda)} = W^{(\lambda)} \cdot (\mu^{(\lambda)} \odot x_t + (1 - \mu^{(\lambda)}) \odot x_{t-1}) \tag{1}$$

其中，$\lambda \in \{r、k、v\}$ 表示时间混合模块，$\lambda \in \{r'、k'\}$ 表示通道混合模块，而 $\odot$ 表示逐元素乘法。

其次，时间混合模块中的 WKV 计算相当于加权平均，并根据当前 token 的键 k_t、值 v_t、可训练时间衰减向量 w 和奖励向量 u，递归地更新其状态 wkv_t，可用公式 (2) 表示。

$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} \odot v_i + e^{u+k_t} \odot v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}} \tag{2}$$

2.2 大语言模型的量化方案

在大语言模型的研究中，模型量化是提高部署和推理效率的有效技术。均匀量化 [8] 是最基础的量化方法，它将浮点数 x线性映射为 b 位整数 q。均匀量化采用固定的表示分配方式，在分布的密集区域和稀疏区域分配相同的精度；在低比特设置下，这会放大离群值引起的误差，并导致准确率大幅下降。

为了解决均匀量化的缺点，研究人员探索了非均匀量化方法 [9][10][11],，例如对数量化[12][13]。然而，通用的非均匀量化方案在硬件实现方面面临挑战。为解决这些缺点，研究人员探索了使量化级别的分布与权重的统计分布相匹配的非均匀量化方法。在硬件层面，将浮点激活值与任意非均匀量化的权重相乘，通常需要复杂的逻辑电路，这可能会抵消量化带来的部分性能优势 [14]。

为兼顾非均匀性与硬件效率，研究人员提出了二次幂（PoT） [15] 量化。PoT 将连续的浮点权重张量 w 映射到由二次幂值构成的离散集合中。其数学表达式如公式（3）所示:

$$w_q = S \cdot \text{sign}(w) \cdot 2^E \tag{3}$$

其中，w_q 表示量化权重，S 表示浮点缩放因子，sign(w) 表示原始权重 w的符号，E 是表示量化指数的整数，其取值范围由量化位宽 b决定。

尽管 PoT 量化具有很高的效率，但其量化级别仅限于单个二次幂项，导致表示能力受限。为解决这一问题，Li 等人提出了加性二次幂量化（APoT） [16]。在 APoT 引入之后，后续研究（例如“n-hot”量化 [17],）引入了位级稀疏性，并结合

加法与减法 PoT 项来提升表示能力，从而扩展了加法量化的概念，以更好地平衡硬件效率与模型精度之间的权衡。这些进展清晰地表明，研究正朝着开发更灵活、更具硬件感知能力的非均匀量化方法发展。

3 量化策略

为降低计算复杂度和内存开销，并提升硬件效率，本文提出了一种硬件友好的混合精度量化架构。对于参与乘法运算的权重，采用一种新颖的 Δ-PoT 量化，以缓解乘法运算带来的瓶颈。相比之下，对于用于加法等逐元素运算的权重，则采用 9 位均匀对称量化，以确保精度。此外，对所有其他激活值和中间结果应用 9 位均匀定点量化，从而在性能和准确率之间实现最佳平衡。

3.1 矩阵-向量乘法的量化方案

鉴于矩阵-向量乘法构成了 RWKV模型的主要计算瓶颈，且矩阵权重会带来显著的内存带宽压力，本文提出了一种新颖的乘法量化方案，以及一种针对矩阵-向量乘积设计的算法感知架构。受 PoT 量化启发，我们提出了一种改进变体 Δ-PoT，通过采用差分编码格式进行权重存储，增强了 APoT 方案。

为简化起见，在 APoT 量化中，我们只考虑无符号数：每个量化级别都是 n 个 PoT 项之和，如下所示。

$$Q^a(a, kn) = \gamma \times \left\{ \sum_{i=0}^{n-1} p_i \right\} \tag{4}$$
$$\text{where} \quad p_i \in \left\{ 0, \frac{1}{2^i}, \frac{1}{2^{i+n}}, \ldots, \frac{1}{2^{i+(2^k-2)n}} \right\}$$

这里，γ 是一个缩放系数，用于确保 Q^a 中的最大级别为 a。k 表示每个加法项的位宽，n 表示加法项的数量。给定总位宽 b和基准位宽 k，n 的计算结果为 $n = b/k$，从而得到 $2^{kn} = 2^b$ 个总级别。

与 APoT 不同，我们的 Δ-PoT 量化允许每个 p_i 具有不同的 k_i，其量化级别定义如下：

$$Q^a\left(a, \sum k_i\right) = 2\gamma \times \left\{ \sum_{i=0}^{n-1} p_i \right\} \tag{5}$$
$$\text{where} \quad p_i \in \left\{ 0, \frac{p_{i-1}}{2^1}, \frac{p_{i-1}}{2^2}, \ldots, \frac{p_{i-1}}{2^{2^{k_i}-1}} \right\}, p_{-1} = 1$$

这种方法允许任意分配 k_i 值，而不受 $k = b/n$ 的限制，因此在 FPGA 上灵活选择位宽方面具有显著优势，并能在某些场景下实现更高的精度。

令 $p_i = 1/2^{q_i}, \Delta q_i = q_i - q_{i-1}$，可得：

$$p_i = \begin{cases} p_{i-1} \cdot 2^{-\Delta q_i}, & \Delta q_i > 0, \\ 0, & \Delta q_i = 0, \end{cases} \quad \Delta q_i \in \{0, 1, 2, \ldots, 2^{k_i} - 1\} \tag{6}$$

这里，Δq_i 表示为每个项存储的差分编码值，因此我们称之为 Δ-PoT 量化。

我们以 $b = 4$ 和 $k = 2$ 为例，分析 Δ-PoT 相对于 APoT 的优势。在此示例中，APoT 有$p_0 \in \{0, 2_0, 2^{-2}, 2^{-4}\}$、$p_1 \in \{0, 2^{-1}, 2^{-3}, 2^{-5}\}$，而 Δ-PoT 有 $p_0 \in \{0, 2^{-1}, 2^{-2}, 2^{-3}\}$、$p_1 \in \{0, 2^{-1}p_0, 2^{-2}p_0, 2^{-3}p_0\}$。如果我们需要量化 $\gamma \times (2^0 + 2^{-2})$的值，APoT 只能表示为 $\gamma \times (2^0 + 2^{-3})$，而 Δ-PoT 则可以精确表示 $2\gamma \times (2^{-1} + 2^{-3})$。

我们提出的 Δ-PoT 量化方案消除了对乘法器的依赖。它可以通过组合多个移位器和加法器来取代 DSP 单元，为支持桶形移位的平台（如 FPGA）提供了极大的便利。首先，该方案降低了计算复杂度。其次，它以简单的“移位-相加”结构取代复杂的乘法器，从而节省片上资源。第三，它采用最契合硬件特性的形式进行计算，在确保模型性能的同时，通过高精度的非均匀表示降低计算延迟。最后，Δ-PoT 允许以低位宽存储权重，从而节省存储空间和内存带宽；其差分编码机制还扩大了数值表示范围。

与均匀定点量化相比，不同权重可以采用不同的缩放因子，这会进一步加剧带宽压力，并增加算子的复杂性。相比之下，我们的量化方案无需修改算子，且对硬件友好。

3.2 逐元素运算的混合精度量化

除使用 Δ -PoT 量化的矩阵权重外，需要与激活值相乘的权重也采用 Δ-PoT 量化。除此之外，所有参与与激活值进行加法运算的权重均采用 9 位均匀对称量化。这是因为加法运算难以充分利用 Δ-PoT 的结构优势，并且 RWKV模型中的权重在数值上相对更加集中。因此，考虑到准确率要求，简单直接的均匀量化是合适的选择。

为确保应对离群值时的精度，我们对所有激活值采用 9 位均匀量化。对于除法和 Sigmoid 等高精度运算，由于不涉及权重，9 位定点量化已足够。为进一步确保数值稳定性并减小误差，其硬件模块在内部以 16 位精度运行。

4 架构设计

4.1 RWKV 硬件架构

为加速整个 RWKV架构，需要为除法、乘方运算、Sigmoid 激活和层归一化等复杂运算提供专用硬件支持。除法和乘方运算主要集中在 WKV 运算符中，

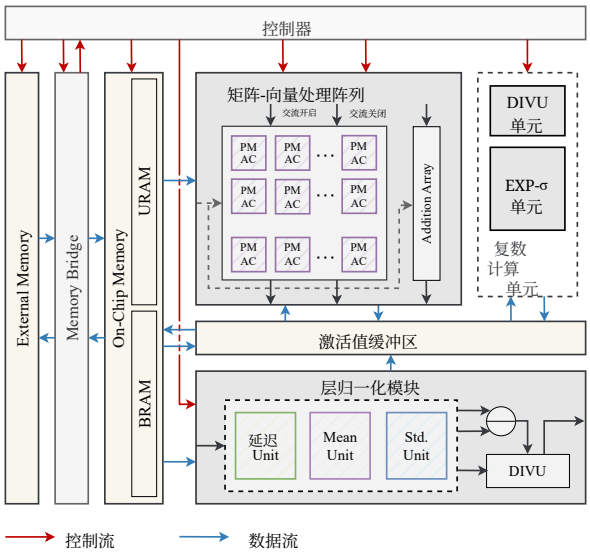

图 2：HFRWKV系统架构

而 Sigmoid 则参与每次 v 项计算。在输入进入时间混合模块和通道混合模块之前，必须执行层归一化操作。为满足这些需求，我们设计了一个专为 RWKV模型打造的全片上计算系统，如图 2 所示。

该架构由外部存储器、存储器桥接器、片上存储器、控制器、激活值缓冲区、矩阵-向量处理阵列、复数计算单元和层归一化模块组成。片外权重通过存储器桥接器从外部存储器批量传输到片上存储器，确保计算时能够高效访问数据。控制器负责协调来自外部存储器的数据传输，并管理片上模块从片上存储器的指定地址获取数据。

为尽量减少片外存储器访问，向量权重和历史值全部存储在片上块随机存取存储器中。由于权重采用混合精度量化，因此先在片外拼接，再传输到片上并解码为相应的位宽。然而，大型矩阵权重（例如，7B模型包含 4096×4096 个元素）超出了片上容量。因此，我们采用基于超大容量RAM的乒乓双缓冲策略，使分块矩阵传输能够与计算重叠进行。这样一来，较小模型（例如0.43B）的权重可以全部存储在超大容量RAM中，而较大模型（例如7B）的权重则可以按块传输，从而有效隐藏存储器延迟并充分利用HBM带宽。

根据 RWKV模型的结构，在主要计算模块之前必须对输入向量应用层归一化。因此，数据流从片上存储器流向层归一化模块。基于层归一化的计算逻辑，该模块包含用于流水线对齐的延迟单元、均值单元和标准差单元，随后执行减法和除法运算（DIVU），以生成归一化输出。

归一化后的向量存储在激活值缓冲区中，以供后续计算使用。该缓冲区临时保留具有时间/空间局部性的数据，包括激活值、中间变量和预取权重。

矩阵-向量乘法和逐元素运算由矩阵-向量处理阵列加速。该可复用该结构支持批量并行计算，并具有三种运行模式：启用 PMAC 单元中的累加器（交流）时，执行矩阵-向量乘法；禁用交流单元时，执行逐元素乘法；通过加法器阵列选择加法。

WKV 运算符中的复杂运算（如乘方运算和除法）由专用复数计算单元处理，其中包括无符号除法单元和共享的自然指数- Sigmoid 单元。这些模块通过资源优化设计，实现 RWKV 中的所有非线性计算。

所有计算模块均采用完全流水线化的高并行架构。细粒度流水线支持批量处理逐元素运算，而粗粒度流水线则使数据传输与计算重叠进行。

4.2 矩阵-向量处理阵列

我们针对使用 Δ-PoT 表示法进行矩阵-向量乘法（MVM）计算，提出了专门的架构优化，并引入了并行矩阵-向量处理阵列。受脉动阵列[18], 的启发，我们发现 MVM 计算同样可以利用时间局部性和空间局部性。关键在于，通过极致的并行性可以最大限度地利用时间局部性：所有涉及当前数据元素的计算都在单个周期内同步完成，从而实现单次读取数据复用。图 3 展示了空间局部性模式。

考虑一个方阵 $W \in \mathbb{R}^{n\times n}$和向量 $\mathbf{v} \in \mathbb{R}^{n}$。如图 3 所示，结果向量 $\mathbf{r(1)}$ 的第一个元素 $= \sum_{j=1}^{n} W_{1j}v_j$。分解这一计算过程可以发现， v_1必须与W的第一列中的每个元素（$W_{i1}\forall i$）相乘。通过重新排列运算，我们在一个周期内并发处理所有 $v_1 \times W_{i1}$计算。该方法实现了 O（n）次运算的高并行度

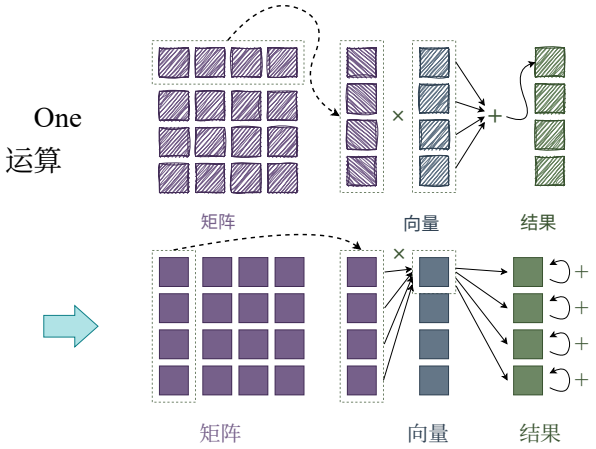

图 3：矩阵向量乘法计算

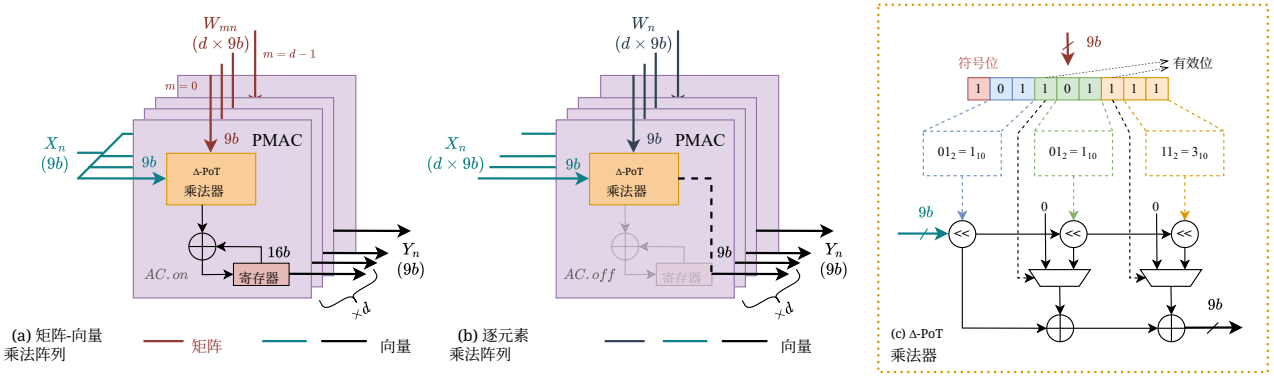

图 4：矩阵-向量处理阵列

每个周期一次，并且每个向量元素仅需读取一次，从而实现最小化数据访问。通过维度感知调度，同样的原理也适用于非方阵。

所提出的架构采用并行矩阵-向量处理阵列，以 Δ-PoT 乘法累加器（PMAC）作为计算单元，并采用三级流水线实现。图 4(a) 展示了累加器（AC）启用时的矩阵-向量乘法模式，每个时钟周期处理矩阵的第 n 列和向量的第 n 个元素。为防止累加期间发生溢出，累加器中加入了 16 位寄存器。对于处理 l 维输入的 d 个并行 PMAC 单元，结果延迟为 $(l+4) \times (l/d)$ 个周期，其中计入了流水线初始化和排空开销。图 4(b) 展示了 AC 停用时的逐元素乘法模式，每个周期处理一个向量元素 n。此配置以 $(l/d+4)$ 个周期的延迟计算逐元素乘积，实现了满流水线利用率。图 4(c) 展示了 Δ-PoT 乘法器的结构。它将乘法器输入（不包括符号位）分解为三个部分，并通过桶形移位器和移位加法累加进行处理。该设计以优化的组合逻辑取代传统的基于数字信号处理器的乘法器，在保持数值精度的同时显著降低了 DSP 资源利用率。所有计算路径均包含图中未明确展示的溢出保护机制。

4.3 无符号除法单元

我们设计中的无符号除法单元（DIVU）如图 5(a) 所示，在运算前需要进行符号位分离以及有符号数到无符号数的转换。除法算法将被除数 X 和除数 Y 分解为归一化形式 $X = 2^{k_1} \cdot x$ 和 $Y = 2^{k_2} \cdot y$，其中 $1 \le x, \ y < 2$。随后，商可推导为：

$$Q = \frac{X}{Y} = \frac{x}{y} \times 2^{(k_1 - k_2)} = \frac{x}{y} \ll (k_1 - k_2) \tag{7}$$

二次幂分量（$k_1 - k_2$）通过前导 1 检测器（LOD）执行减法和移位操作计算得出。两个操作数首先经过 LOD 处理，以确定指数值 k_i，其中 $\mathrm{LOD}(X) = k_1$ 表示 X 中最高有效位“1”的位置。

小数部分除法（x/y）通过二维查找表（2D-LUT）实现。归一化后，x 和 y 中前导“1”之后的最高有效四位（MSB）分别用作行索引和列索引。通过这种 4×4 位索引，查找表的条目数限制在 256 个，从而为小数商提供 8 位精度。

LOD 模块采用分层二分搜索算法。对于 k 位输入，该 $\log_2 k$ 级过程依次检查 8 位、4 位以及更细的片段。在每一级中，根据上半部分是否存在“1”来决定片段选择，如算法 1 所示。与针对 16 位操作数的顺序检测相比，该方法将逻辑深度降低了 58%。

最终商将查找表中的小数部分与指数驱动的移位相结合，在三个流水级中完成除法：(1) 归一化和前导零检测器（LOD）；(2) 小数部分除法；(3) 重组和反归一化。

4.4 指数–Sigmoid 单元

在提出的方法中，可复用的指数–Sigmoid（EXP–σ）单位在统一架构下支持两种不同的运算：以 e 为底的乘方运算和 Sigmoid 激活（图 5(b)）。

算法 1 使用分层二分搜索算法的前导一检测器

```
函数前导零检测器(in[k − 1 : 0])
    p ← 0, w ← k, d ← in
    当 w > 1 时执行
        h ← w/2
        如果 ∨ d[w − 1 : h] = 1 则
            d ← d[w − 1 : h], p ← p + h
        else
            d ← d[h − 1 : 0]
        结束条件判断
        w ← h
    结束循环
    如果 d = 1 则
        返回 p
    else
        返回 −1
    结束 if
结束函数
```

通过选择控制信号 $mode$，同一数据通路可重新用于不同操作，从而减少面积和控制开销。

当 $mode = 0$时，该模块利用一种数学变换，将自然指数运算转换为以2为底的形式[19]，，从而充分利用高效的移位操作：

$$e^X = 2^Y, \quad Y = X \log_2 e, \quad \log_2 e \approx 1.0111_2 \tag{8}$$

随后将乘积 Y 分解为整数部分 u 和小数部分 v。项 2^u 通过位移计算，而 v 的前八位用于索引一个含256个条目的指数查找表，以获得八位精度。乘以常数 $\log_2 e$ 可通过一次加法、一次减法和两次移位操作实现。

当 $mode = 1$时，该模块通过分段线性近似[20], 计算 Sigmoid 函数，其中每个线段的斜率被分解为二进有理数之和，并通过共享的移位加法单元实现。所需系数从 σ-查找表中检索。近似定义如下：

$$f(x) = \begin{cases} 1, & x \geq 5 \\ 0.03125\,x + 0.84375, & 2.375 \leq x < 5 \\ 0.125\,x + 0.625, & 1 \leq x < 2.375 \\ 0.25\,x + 0.5, & 0 \leq x < 1 \\ 1 - f(-x), & x < 0 \end{cases} \tag{9}$$

该方法将非线性计算中繁重的工作卸载到小型、高速存储器中，从而同时实现高吞吐量和低逻辑资源利用率。

在两种模式下，所有定常数乘法——无论是乘以 $\log_2 e$ 还是乘以分段斜率——均由专用的移位加法单位取代。该单位动态选择并组合移位操作数，仅需极少的算术电路，从而最大限度地提高两种模式之间的复用率，并简化整体控制逻辑。

4.5 层归一化模块

在许多以往的研究中，层归一化层的计算任务都交由 CPU[21], 处理，因为 CPU 更擅长原生执行除法和标准差计算等复杂运算。然而，这种方法会产生数据移动延迟，降低整体系统吞吐量，并加剧内存带宽压力。鉴于层归一化计算的大部分工作是向量求和——这类运算正是 FPGA 的强项——我们设计了一个完全片上实现的层归一化模块，以消除片外传输并实现端到端加速。[22]

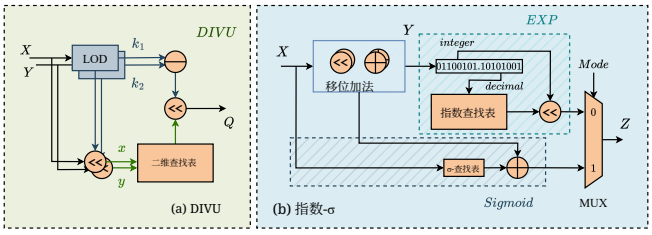

图 5：（a）无符号除法单元（b）指数–Sigmoid 单元

层归一化计算需要均值

$$\mu = \frac{1}{d} \sum_{i=1}^{d} x_i \tag{10}$$

和方差

$$\sigma^2 = \frac{1}{d} \sum_{i=1}^{d} (x_i - \mu)^2 \tag{11}$$

其中，x_i 表示输入向量的第 i 个元素，d 表示其长度。应用恒等式

$$\sigma^2 = \frac{1}{d} \sum_{i=1}^{d} x_i^2 - \left(\frac{1}{d} \sum_{i=1}^{d} x_i \right)^2 \tag{12}$$

我们分别计算两个独立的求和——一个针对 x_i，另一个针对 x_i^2——随后进行减法和平方根运算：

$$\sigma = \sqrt{\sigma^2 + \epsilon} \tag{13}$$

其中，ϵ 是一个较小的常数，用于确保数值稳定性。

为在现场可编程门阵列上加速这些求和运算，我们采用流水线加法树（AT）与累加器（交流）相结合的结构，称为 ATAC 结构。树并行度为 512 时，对 d 个元素的每次求和可在 $\lceil d/512 \rceil + 9$ 个时钟周期内完成（图 6）。输入向量 X 被划分为与 ATAC 并行宽度相匹配的数据块；均值路径和方差路径均包含相同的 ATAC 单元。每个 ATAC 处理完所有数据块后，其输出保持不变。在均值路径中，最终累加和通过优化的移位加法运算除以常数 d。在方差路径中，均值被分发并平方，同时每个 x_i 也被平方，然后减去均值的平方。这两路数据流进入减法—平方根模块，该模块计算方差，加上 ϵ，再求平方根，得到 σ。延迟缓冲器确保同步，使得首个数据块到达归一化阶段时，μ 和 σ 均已有效。最后，通过计算 $(x_i - \mu)/\sigma$ 对 X 的每个数据块进行归一化，并将结果流式输出（参见图 2）。

5 实验

5.1 实验设置

模型与数据集。 为系统评估不同量化策略对RWKV模型性能的影响，我们遵循既定的评估流程，将我们的混合精度量化方案与多种对比量化方法（RTN [8], LogQ [12][13], PoT [15]）应用于RWKV架构。我们通过LAMBADA数据集[23] 上的困惑度（ppl），以及在七个基准数据集上的零样本准确率（acc）来评估模型效果：LAMBADA、HellaSwag [24], Arc-Easy [25], Arc-Challenge [25], SciQ [26], PIQA [27], 和 Winogrande [28]。实验结果汇总于表1。

实现。提出的设计分别部署在赛灵思数据中心加速卡平台 Alveo U50（称为 HFRWKV）和 U280（称为 HFRWKV*），使用 Vivado 2023.2 实现。这两个平台均基于 16 nm UltraScale+ TM 架构，并配备 8 GB HBM2 高带宽内存。Alveo U50 提供的硬件资源包括 872K 个查找表（LUT）、1,743K 个寄存器、5,952 个 DSP 切片、640 个 288 Kb 块RAM（BRAM）、1,344 个 36 Kb 超随机存取存储器（URAM），额定带宽为 201 GB/s。

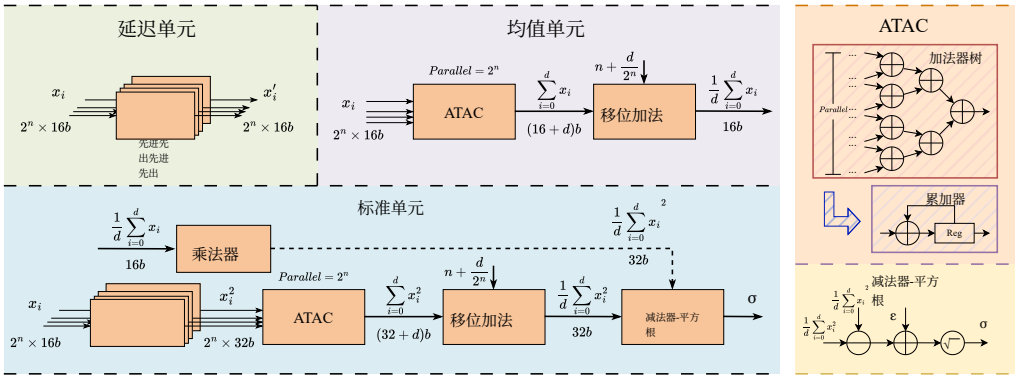

图 6：层归一化模块

表1：不同量化方案和数据集下 RWKV 的困惑度（ppl）和准确率（acc）。

精度	LAMBADA 困惑度 ↓	LAMBADA 准确率 ↑	HellaSwag 准确率 ↑	Arc-Easy 准确率 ↑	Arc-Challenge 准确率 ↑	SciQ 准确率 ↑	PIQA 准确率 ↑	Winogrande 准确率 ↑	平均值 累加器 ↑
FP16	7.18	58.00	42.33	47.0	23.6	82.6	72.2	54.6	54.33
RTN	8.40	52.29	40.33	46.1	22.3	80.4	70.4	52.4	52.03
PoT	14.4	43.14	37.14	45.5	22.2	78.8	69.6	51.6	49.43
LogQ	8.28	52.71	40.11	46.2	22.6	80.2	70.5	52.6	52.13
提出的方法	7.24	56.41	41.33	46.2	22.9	81.4	70.4	53.2	53.12

Alveo U280 提供 1,304K 个查找表、2,607K 个寄存器、9,024 个 DSP 切片、2,016 个 36 Kb 块RAM、960 个 288 Kb 超随机存取存储器，带宽为 460 GB/s。

基准。 据我们所知，目前尚无专门针对 RWKV 架构系列的 FPGA 加速研究。尽管存在类似的 Transformer/RNN加速器[29][30]，但它们在结构和算法上的差异使得性能比较无法作为有意义的基准。因此，我们采用 CPU 和 GPU 平台作为性能基准。CPU 参考系统采用 Intel Core i7-12650H（10纳米工艺，10 核/16 线程）和 16GB DDR4内存。GPU 基准涵盖从中端到高端的多个型号：NVIDIA GeForce RTX 2080Ti（12 纳米，11GB）、RTX 3090（8 纳米，24GB），以及 A100 机器学习优化型HPC卡（7 纳米，40GB）。对于 CPU 和 GPU 的所有测量，我们均使用测试平台的官方 RWKV实现。吞吐量通过反复处理单令牌输入（批大小=1），并对持续的每秒令牌数进行平均来获得，从而提供客观的比较。

5.2 量化消融研究

我们现进行消融实验，以展示不同量化方案对模型性能的影响。该消融实验使用官方rwkv pip包以及ChatRWKV项目中的基准测试脚本，在GPU上完成 [31][32]。如表1所示，所有结果均在不同数据集上与未量化的半精度浮点数基线进行比较。值得注意的是，RTN、PoT和LogQ的结果是通过模拟等效W9A9量化的精度损失得到的。应用传统量化方案会导致性能出现不同程度的下降。PoT量化带来的平均准确率下降最为显著，降至49.43%。虽然RTN和LogQ方案表现更好，但仍无法保持较高的保真度，平均准确率分别为52.03%和52.13%，其在LAMBADA数据集上的困惑度也恶化至超过8.25。

相比之下，我们提出的方案有效缓解了这种性能损失，平均准确率达到 53.12%，明显优于所有其他量化方案。更重要的是，我们的方法在 LAMBADA数据集上的困惑度保持在 7.24，远比其他方案更接近 FP16基线。这表明，我们提出的方案实现了更优的权衡，相比现有技术能够更有效地保留模型准确率。

表2：HFRWKV 和 HFRWKV* 的资源利用率

	HFRWKV	_0 HFRWKV	_1 HFRWKV*	_0 HFRWKV*	_1
平台	Alveo U50		Alveo U280		
支持规模	169M	430M / 1B5 / 3B / 7B	169M	430M / 1B5 / 3B / 7B	
频率	350兆赫兹	350兆赫兹	400兆赫兹	400兆赫兹	
LUT	95718（11%）	137631（16%）	126956（10%）	182372（14%）	
FF	82719（5%）	124350（7%）	102809（4%）	151158（6%）	
DSP	641（11%）	1025（17%）	1025（11%）	1537（17%）	
BRAM	45（3%）	637（47%）	45（2%）	637（32%）	
URAM	96（15%）	128（20%）	192（20%）	256（27%）	

5.3 硬件结果

5.3.1 资源利用率。

为了支持不同规模的模型，同时充分利用高带宽内存，我们在两个Alveo平台上实现了两个系统版本。HFRWKV_0（HFRWKV*_0）用于最小规模的模型，而 HFRWKV_1（HFRWKV*_1）则用于其余模型。HFRWKV部署在 Alveo U50 上，在 350 MHz 片上频率下实现了 99.95% 的带宽利用率。HFRWKV*部署在 Alveo U280 上，在 400 MHz 下达到 99.64% 的带宽利用率。在 HFRWKV_0中，矩阵-向量处理阵列将 l 设为权重向量的维度，批次参数为 $d = 384$；其层归一化模块则将 d 配置为权重向量维度，参数为 $tree\ parallelism = 256$ [_1 _0 _1] 。对于 HFRWKV、HFRWKV*、HFRWKV*，其配置分别为 $d = [512, 768, 1024]$ 和 $tree\ parallelism = [512, 256, 512]$。四种配置均集成了 128 个复制的无符号除法单元和指数–Sigmoid 单元。这种参数设计能够适应不同规模的模型，并最大限度地提高板载资源和带宽利用率。相应的资源利用率指标详见表2。

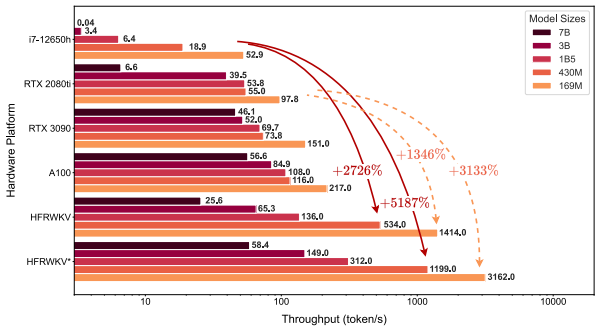

图7：CPU、GPU、HFRWKV 和 HFRWKV* 的吞吐量

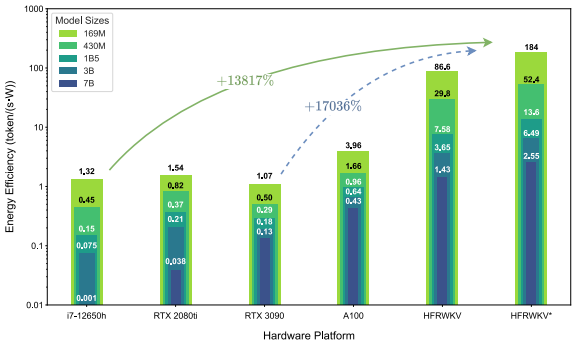

图8：CPU、GPU、HFRWKV 和 HFRWKV* 的能效

5.3.2 性能比较。

加速比。 图 7 总结了 RWKV模型在不同平台上的推理吞吐量测量结果。对于 169M 的模型规模，HFRWKV 的吞吐量达到 CPU 基准的 26.74× 倍，分别达到 RTX 2080Ti、RTX 3090 和 A100 的 14.46× 倍、9.37× 倍和 6.51× 倍。HFRWKV* 则将这些倍数进一步提升至 59.8×、32.33×、20.95× 和 14.55×。

对于最大的 7B 模型规模，HFRWKV 的吞吐量仅为 RTX3090 的 0.55× 倍，以及 A100 的 0.45× 倍。相比之下，HFRWKV* 略微超过 A100，达到其吞吐量的 1.03× 倍，因此在所有模型规模下均优于 A100。

能效。 根据 Vivado功耗分析报告得出的功耗测量结果显示，能效优势显著。如图 8 所示，与 CPU 和 GPU 基准相比，HFRWKV 和 HFRWKV* 的能效都大幅提升，凸显了我们基于FPGA的设计所带来的优势。

6 结论

本文提出了 HFRWKV，这是一种采用混合精度量化、具有全片上计算架构的 RWKV 硬件加速器。我们提出了一种用于乘法运算的新型 Δ-PoT 量化方法，并设计了一种新颖且适用于硬件的混合精度量化策略。针对乘方运算和除法等复杂运算，我们提出了一种结合可复用架构与查找表或分段线性近似的方法。基于这些创新，我们构建了高吞吐量流水线架构，其中集成了并行矩阵-向量处理阵列、复数计算单元和片上层归一化模块。通过计算重排和分块双缓冲技术，我们的架构消除了数据传输瓶颈，实现了完全片上计算。在加速 RWKV-4 时，与 CPU 基准相比，HFRWKV 实现了 63.48× 的加速比和 139.17× 的能效提升；与 GPU 实现相比，则实现了 32.33× 的加速比和 171.36× 的能效提升。

尽管本文聚焦于 RWKV-4，HFRWKV 本质上兼容整个 RWKV 模型系列。未来，我们将扩展 FPGA 加速框架，以支持所有 RWKV 变体。

7 致谢

本研究得到国家自然科学基金（项目编号：62276278）以及广东省基础与应用基础研究基金（项目编号：2022A1515110006 和 2024A1515011259）的资助。

参考文献

[1] Andrea Matarazzo 和 Riccardo Torlone。《大语言模型综述：关于其能力与局限的一些见解》，2025年。[2] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。Attention is all you need。载于 I. Guyon、U. Von Luxburg、S. Bengio、H. Wallach、R. Fergus、S. Vishwanathan 和 R. Garnett 编，神经信息处理系统进展，第30卷。Curran Associates, Inc.，2017年。[3] Susan Zhang、Stephen Roller、Naman Goyal、Mikel Artetxe、Moya Chen、Shuohui Chen、Christopher Dewan、Mona Diab、Xian Li、Xi Victoria Lin 等。OPT：开放式预训练变换器语言模型。*arXiv*预印本 *arXiv:2205.01068*，2022年。[4] Zhuoran Shen、Mingyuan Zhang、Haiyu Zhao、Shuai Yi 和 Hongsheng Li。高效注意力：具有线性复杂度的注意力。载于论文集

IEEE/CVF冬季计算机视觉应用会议，第3531–3539页，2021年1月。[5] Bo Peng、Eric Alcaide、Quentin Anthony、Alon Albalak、Samuel Arcadinho、Stella Biderman、Huanqi Cao、Xin Cheng、Michael Chung、Matteo Grella 等。RWKV：为Transformer时代重新构想循环神经网络。arXiv预印本 arXiv:2305.13048，2023年。[6] Jimmy Lei Ba、Jamie Ryan Kiros 和 Geoffrey E. Hinton。层归一化，2016年。[7] Angelos Katharopoulos、Apoorv Vyas、Nikolaos Pappas 和 François Fleuret。Transformer即循环神经网络：采用线性注意力的快速自回归Transformer。载于国际机器学习会议，第5156–5165页。机器学习研究论文集，2020年。[8] Benoit Jacob、Skirmantas Kligys、Bo Chen、Menglong Zhu、Matthew Tang、Andrew G. Howard、Hartwig Adam 和 Dmitry Kalenichenko。面向高效纯整数运算推理的神经网络量化与训练。CoRR，abs/1712.05877，2017年。[9] Sanghyun Seo 和 Juntae Kim。基于核密度估计的非均匀量化器实现卷积神经网络的高效权重量化。《应用科学》，9(12)，2019年。[10] Dongqing Zhang、Jiaolong Yang、Dongqiangzi Ye 和 Gang Hua。LQ-Nets：用于高精度、紧凑型深度神经网络的学习式量化。CoRR，abs/1807.10029，2018年。[11] Antonio Polino、Razvan Pascanu 和 Dan Alistarh。通过蒸馏和量化进行模型压缩。CoRR，abs/1802.05668，2018年。[12] Jingyong Cai、Masashi Takemoto 和 Hironori Nakajo。深入探讨神经网络模型参数的对数量化。载于国际信息技术进展会议，IAIT '18。计算机协会，2018年。[13] Edward H. Lee、Daisuke Miyashita、Elaina Chai、Boris Murmann 和 S. Simon Wong。LogNet：采用对数计算的节能神经网络。载于IEEE国际声学、语音与信号处理会议（ICASSP），第5900–5904页，2017年。[14] Yang Lin、Tianyu Zhang、Peiqin Sun、Zheng Li 和 Shuchang Zhou。FQ-ViT：无需重新训练的全量化视觉Transformer。CoRR，abs/2111 13824，2021年。 。[15] Aojun Zhou、Anbang Yao、Yiwen Guo、Lin Xu 和 Yurong Chen。增量网络量化：实现低精度权重下的无损卷积神经网络。CoRR，abs/1702.03044，2017年。[16] Yuhang Li、Xin Dong 和 Wei Wang。加性二次幂量化：一种用于神经网络的非均匀离散化方法。CoRR，abs/1909.13144，2019年。[17] Yuiko Sakuma、Hiroshi Sumihiro、Jun Nishikawa、Toshiki Nakamura 和 Ryoji Ikegaya。n-hot：用于二次幂神经网络量化的高效位级稀疏性。CoRR，abs/2103.11704，2021年。[18] Kung。为什么采用脉动阵列架构？《IEEE计算机》，15(1):37–46，1982年。[19] Meiqi Wang、Siyuan Lu、Danyang Zhu、Jun Lin 和 Zhongfeng Wang。深度学习中Softmax函数的高速低复杂度架构。载于IEEE亚太电路与系统会议（APCCAS），第223 226页，2018年 – 。[20] Hesham Amin、K Memy Curtis 和 Barrie R Hayes-Gill。分段线性近似在神经网络非线性函数中的应用。《IEE电路、器件与系统学报》，144(6):313–317，1997年。[21] Zhengang Li、Mengshu Sun、Alec Lu、Haoyu Ma、Geng Yuan、Yanyue Xie、Hao Tang、Yanyu Li、Miriam Leeser、Zhangyang Wang、Xue Lin 和 Zhenman Fang。Auto-ViT-Acc：面向视觉Transformer、采用混合方案量化的面向FPGA的自动加速框架。载于国际现场可编程逻辑与应用会议（FPL），第109–116页，2022年。[22] Kyle Marino、Pengmiao Zhang 和 Viktor K Prasanna。Me-vit：用于视觉Transformer的单次加载的存储器高效FPGA加速器。载于IEEE国际高性能计算、数据与分析会议（HiPC），第213–223页。IEEE，2023年。[23] Denis Paperno、Germán Kruszewski、Angeliki Lazaridou、Quan Ngoc Pham、Raffaella Bernardi、Sandro Pezzelle、Marco Baroni、Gemma Boleda 和 Raquel Fernández。LAMBADA数据集：需要广泛话语语境的词语预测。CoRR，abs/1606.06031，2016年。[24] Rowan Zellers、Ari Holtzman、Yonatan Bisk、Ali Farhadi 和 Yejin Choi。HellaSwag：机器真的能续写你的句子吗？CoRR，abs/1905 07830，2019年。 。[25] Peter Clark、Isaac Cowhey、Oren Etzioni、Tushar Khot、Ashish Sabharwal、Carissa Schoenick 和 Oyvind Tafjord。你认为自己已经解决了问答问题？试试ARC——AI2推理挑战。CoRR，abs/1803.05457，2018年。[26] Johannes Welbl、Nelson F. Liu 和 Matt Gardner。众包科学选择题。CoRR，abs/1707.06209，2017年。[27] Yonatan Bisk、Rowan Zellers、Ronan Le Bras、Jianfeng Gao 和 Yejin Choi。PIQA：在自然语言中推理物理常识。CoRR，abs/1911.11641，2019年。[28] Keisuke Sakaguchi、Ronan Le Bras、Chandra Bhagavatula 和 Yejin Choi。Winogrande：大规模对抗式Winograd模式挑战。《ACM通讯》，第99–106页，2021年8月。[29] Bingbing Li、Santosh Pandey、Haowen Fang、Yanjun Lyv、Ji Li、Jieyang Chen、Mimi Xie、Lipeng Wan、Hang Liu 和 Caiwen Ding。FTRANS：节能

基于现场可编程门阵列的 Transformer 加速。*CoRR*，abs/2007.08563，2020年。[30] Yuntao Han 和 Qiang Liu。HPTA：基于现场可编程门阵列的高性能 Transformer 加速器。收录于国际现场可编程逻辑与应用会议 *(FPL)*，第 27–33 页，2023年。[31] BlinkDL。官方rwkv pip包。https://pypi.org/project/rwkv/。[32] BlinkDL。chatrwkv 基准测试脚本。https://github.com/BlinkDL/ ChatRWKV/blob/main/v2/benchmark.py。