面向 RISC-V 向量处理器大语言模型推理的原生块量化矩阵乘法与双视图访问下的显式 K/V 驻留

Weiying Wang、Yuanyang Xiang 和 Zhiwei Zhang，中国科学院自动化研究所，中国科学院大学人工智能学院，中国北京 100049。电子邮箱：{wangweiying2021,xiangyuanyang2020,zhiwei.zhang}@ia.ac.cn

arXiv:2611.31937v1 [cs.AR] 24 Nov 2026

摘要——大语言模型推理中的一些关键执行状态会跨多条指令持续存在，难以在以单条操作为中心的 RVV 语义中表达和利用。我们根据数据状态的生命周期定义专用化边界，并在保留 RVV 可编程性的同时提出量化块流 (QBS) 和注意力/KV 上下文 (AKV)。QBS 将原生块格式处理、块遍历、整数计算和有序 FP32 累加整合为统一的执行流程；AKV 则维持本地 K/V 驻留，并提供与计算方向相匹配的操作数视图，同时将注意力运算留在 RVV 上。周期精确的寄存器传输级 (RTL) 评估表明，对于投影 *Compute* 和矩阵乘法，QBS 分别实现了 35.68× 和 51.34× 的几何平均加速比；对于解码注意力，AKV 相较于分块 RVV 实现了 1.45× 的几何平均加速比。与基线相比，完整设计的总面积增加了 12.65%；在所评估的投影工作负载上，单位面积吞吐量和能效分别提升了 46.63× 和 19.78×。进一步评估表明，该设计适用于多种块量化格式，以及采用不同注意力组织方式的模型。

索引词—RISC-V 向量扩展、大语言模型推理、块量化、量化矩阵乘法、K/V 复用

I. 引言

大语言模型（LLM）广泛应用于语言理解、内容生成和代码处理 [1]–[3], ，对推理效率提出了越来越高的要求。Transformer 推理会反复执行线性投影和注意力操作 [4]; 随着模型规模和上下文长度持续增长，权重访问、注意力计算和中间数据移动的成本日益显著，使推理效率同时取决于计算吞吐量和数据供给效率。RISC-V 向量扩展（RVV）提供与向量长度无关的计算和内存操作，能够在不同硬件实现和张量形状之间提供统一的可编程执行基础 [5]。然而，运行时确定的数据布局、复用关系和状态生命周期仍主要由软件通过底层向量操作进行管理，因此硬件难以直接利用跨指令持续存在的数据组织和复用模式。基于这一观察，我们在保留 RVV 可编程性的同时，定义根据大语言模型工作负载结构划分的专用化边界，并优化相应的关键执行阶段。

llama.cpp 等推理运行时采用块量化，以减少模型权重存储和带宽需求 [6], ，但紧凑表示并不能直接转化为高效的向量执行。除低比特值外，块量化权重还包含缩放因子、子组元数据和特定格式的校正规则，因此 RVV 内核必须显式执行格式解析、整数计算、校正、浮点更新和块遍历。已有工作在不同执行边界上优化量化投影，包括矩阵级执行、块缩放点积以及原生量化块处理 [7]–[11]; 然而，原生块量化投影仍需要将格式解析、块遍历和输出累加与矩阵级复用协调起来。注意力带来了不同的数据供给问题：K/V 数据在运行时生成，并由后续查询反复复用，而分组查询注意力（GQA）和预填充进一步扩大了其复用范围 [12]。尽管 FlashAttention 通过分块和在线 Softmax 减少了外部数据移动 [13], ，但与计算方向相匹配的本地 K/V 驻留组织和操作数供给，仍需要在向量处理器上由软件进行显式编排。

尽管这两个瓶颈源于不同的执行路径，但它们具有一个共同特性：关键数据状态的生命周期超出了单条 RVV 指令所能表达的范围。如图 1 所示，量化投影必须在连续的 K 块之间保留并更新输出累加状态，而注意力则必须在不同查询之间复用 K/V，并根据计算形成不同的操作数视图方向。基于这一观察，我们提出量化块流（QBS）和注意力/*KV* 上下文 （AKV）。QBS 定义了一个统一的执行边界，覆盖从原生量化块到完整结果分块的整个过程，将格式处理、块遍历以及跨 K 块的有序累加纳入同一执行流程；而 AKV 则将专门化限定在本地 K/V 驻留和操作数供给上，提供与计算相匹配的操作数视图。注意力运算及其他超出支持范围的操作

继续在标准 RVV 上执行。

本工作的主要贡献如下：

- 跨指令状态生命周期的专用化边界与架构接口。针对量化投影中原生量化块之间的有序累加，以及注意力查询之间的 K/V 复用，分别定义了从原生量化块到已完成结果分块、以及从驻留 K/V 到操作数视图的两类架构接口。
- RISC-V 向量处理器中的集成与运行时映射。QBS 和 AKV 复用现有的依赖管理、地址转换与内存保护，以及向量寄存器文件结果传递路径。运行时根据量化格式、布局、形状和数据类型选择专用化执行，而超出所支持接口约束的操作则继续在标准 RVV 上执行。
- 基于真实模型工作负载的寄存器传输级性能评估与功能验证。对于投影计算和矩阵乘法，QBS 分别实现了 35.68× 倍和 51.34× 倍的几何平均加速比；对于解码注意力，AKV 相较于 GQA 感知分块式 RVV 实现了 1.45× 倍的几何平均加速比。验证涵盖多种量化格式，以及采用不同注意力配置的模型。

II. 工作负载结构与相关工作

A. 量化投影与 K/V 复用

线性投影计算 $Y = AW^{\mathsf{T}}$，其中 $A \in \mathbb{R}^{M\times K}$是激活矩阵，$W \in \mathbb{R}^{N\times K}$ 是权重矩阵。单令牌解码通常对应于 $M = 1$，而预填充包含多行激活。块量化沿 K 维度对权重进行划分，形成原生块，因此每个输出元素 $Y[m, n]$ 都会沿 K由多个原生块依次更新。具体而言，一个 Q4 K 原生块占用 144 字节，表示 256 个权重，并划分为八个包含 32 个元素的子组。与 Q8 K 激活配对时，一个块内的整数计算可通过点积和校正项 [6]概括

$$T = \sum_g s_g \sum_{i\in\mathcal{I}_g} q_i^a q_i^w, \qquad U = \sum_g \mu_g \sum_{i\in\mathcal{I}_g} q_i^a. \tag{1}$$

其中，q_i^a 和 q_i^w 分别是激活值和权重的整数值，而 s_g 和 μ_g 编码子组缩放因子与最小值信息。令 y 表示由前序原生块累积得到的输出。当前块随后执行格式定义的 FP32 更新：

$$\begin{aligned} z &= \mathrm{fma}_{32}(R_{32}(d_w d_a), R_{32}(T), y)\,, \\ y' &= \mathrm{fma}_{32}(-R_{32}(d_{\min} d_a), R_{32}(U), z)\,. \end{aligned} \tag{2}$$

在此，d_a、d_w和 $d_{\min}$ 分别表示激活缩放因子、权重缩放因子和最小值缩放因子，而 R_{32} 表示规定的转换或舍入为 FP32。因此，原生块格式规定的不仅是数据编码，还包括子组组织方式、缩放和校正规则，以及块级更新顺序；除了低精度点积之外，其处理还涉及格式解析、特定于格式的校正和 FP32 更新。矩阵结构进一步从两个方向展现了复用：一个激活块可以沿 N为多个输出通道提供服务，而一个权重块可以沿 M为多行激活提供服务。在本文的其余部分，配置表示原生块的编码和数值规则，布局表示其物理放置和遍历顺序，而形状表示一次执行所覆盖的 M、N 和 K 范围。

对于注意力，令 $Q \in \mathbb{R}^{M_q\times D}$和 K，$V \in \mathbb{R}^{T\times D}$。其计算为

$$P = \mathrm{softmax}\left(\frac{QK^{\mathsf{T}}}{\sqrt{D}} + \mathcal{M}\right), \qquad O = PV, \tag{3}$$

其中，D 为头维度，T为当前可见的 K/V 令牌数量，$\mathcal{M}$为注意力掩码。在 GQA 中，多个查询头共享同一个 K/V 头 [12]，；而预填充允许多个查询位置复用相同的 K/V 数据。图 1(b) 展示了这种复用及相应的操作数访问方向。在本工作采用的向量化调度下，注意力分数计算固定特征维度 d，并沿令牌维度读取 $K[:, d]$；而对 V进行加权累加时固定令牌 t，并沿特征维度读取 $V[t,:]$。因此，K 列和 V 行是由向量化调度形成的操作数视图，而不是 K/V 固有的存储形式。

B. 既有工作中的专用化边界

既有关于量化投影和注意力的研究，围绕计算粒度、数据表示、执行映射和存储管理开展了专用化探索 [14]–[16]。对于量化投影，MX、VMXDOTP 和 SpacemiT IME 分别在矩阵、微缩放点积和分块层面扩展了低精度计算，而 F-BFQ 和 DECA 则将原生格式处理或近核权重扩展与专用矩阵执行相结合 [7]–[11]。对于注意力，FlashAttention 及其后续工作主要优化分块、在线 Softmax 和向量映射 [13], [17]–[20]，；而 KIVI、PagedAttention 和 PLENA 则通过 K/V 表示、缓存管理以及配备专用执行支持的片上数据组织来降低开销 [21]–[23]。

相反，本工作根据跨指令状态生命周期这一共同属性来定义专用化边界。QBS 将从原生块访问到完整结果块提交的整个路径纳入一次专用化执行，在 K 个块上共同完成格式解析与校正、块遍历和有序累加；AKV 则将专用化限制在本地 K/V 驻留和操作数视图供给，而注意力运算仍由标准 RVV 执行。这两种机制分别利用跨 K 个块的输出累加状态和跨查询的 K/V 状态，在保留 RVV 可编程性的同时，发挥跨指令状态生命周期和复用关系的优势。

III. 架构接口与运行时映射

QBS 和 AKV 定义了各自对软件可见的接口。表 I 总结了主要操作，而表 II

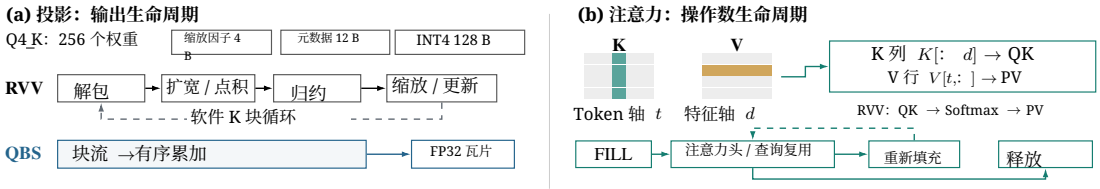


图 1. 跨指令　　生命周期专门化。QBS 会在多个块之间保留输出状态；AKV 会在多个查询之间保留 K/V，并提供操作数视图。

表 I 所提架构的自定义向量指令。

指令	功能
vqbinfo	报告支持的配置、布局和形状
vqbexec	将原生块执行到 $M \times N$ FP32 瓦片；支持激活快照复用
vakvinfo	报告支持的上下文和视图参数
vakvfill	建立或更新 K/V 上下文（填充/重新填充）
vakvload.v	将选定的 Q/K/V 行视图加载到向量寄存器文件
vakvkcol.v	将 K 列或 Panel4 视图加载到向量寄存器文件
vakvrelease	释放当前 K/V 上下文

表 II 架构执行限制。

QBS 模式	M	N	K 块
标准	1–4	≤ 32	≤ 256
Wide	5–8	≤ 16	≤ 256
快照	1	≤ 32	≤ 16 (Q8 K)
AKV 类型	D	Q 行数	令牌
F16 K/V	64/96/128	1–8	≤ 64

给出了单次执行的范围。

A. QBS：原生量化块执行接口

对于 $Y = AW^{\mathsf{T}}$，一条 QBS 命令会处理 M 激活行和 N 个输出，沿着 K 维度遍历指定数量的原生量化块，并保留输出累加状态，直到归约完成。QBS 描述符指定权重地址、配置、布局、 N以及 K 块数量；命令还会指定激活地址、 M和目标寄存器。当M 或 N 超出命令范围时，运行时会沿相应维度生成多个输出分块；当 K 块数量超出限制时，则使用分割 K，每个 K 分段由一条 QBS 命令独立累加并舍入，然后由软件合并部分结果，因此合并结果不要求与未分割执行的结果逐位一致。

当多个输出分块共享相同的量化激活值时，软件可以通过填充、复用和释放显式管理激活快照。对于满足快照约束的 Q8 K 激活，填充会保存当前命令读取的激活数据，复用会使快照可供后续命令使用，而释放则会在最后一次成功使用后将其释放。快照仅存储已量化的数据，而 F32 到 Q8 K 量化仍由软件负责；复用由软件显式指定而不是根据源地址推断，并且只改变跨命令数据供给路径，不会改变 QBS 的分块级计算或舍入规则。

B. AKV：*K/V* 上下文与视图接口

与 QBS 在一条命令中维护输出累加状态不同，AKV 通过显式的注意力上下文维护可供多条指令复用的 K/V 驻留，其生命周期由 FILL、REFILL 和 RELEASE 管理。FILL 建立新的上下文并加载第一个 K/V 分块；REFILL 保留现有查询数据和上下文配置，仅替换当前驻留的 K/V；RELEASE 则在复用完成后释放上下文。建立上下文后，软件通过视图操作获取后续 RVV 计算所需的操作数：行视图返回所选查询行、K 行或 V 行中连续的特征向量；列视图固定一个特征维度，并构成沿令牌方向排列的数据；Panel4 则在单次请求中构成对应于四个相邻特征维度的列视图。将这些操作数写入向量寄存器文件后，QK、Softmax 和 PV 仍使用标准 RVV 指令执行；因此，AKV 改变的只是 K/V 驻留和操作数供给，而不接管注意力运算。当头维度、K/V 序列长度或共享相同 K/V 的查询数量超出单次执行的处理范围时，运行时分别沿特征维度进行分段调度、沿令牌维度将 K/V 序列划分为连续分块，或对查询进行分组；跨 K/V 分块的 Softmax 状态以及跨分段和查询组的调度仍由软件控制。

C. 运行时组合与回退

QBS 和 AKV 分别面向量化投影和注意力计算中的 K/V 操作数供给，并与标准 RVV 协同构成 Transformer 执行路径。满足接口约束的 Q/K/V 投影、输出投影和前馈矩阵乘法可映射到 QBS；K/V 缓存写入完成且满足 AKV 接口约束后，AKV 会提供本地驻留数据及后续注意力计算所需的视图，其余操作则继续在 RVV 上执行。派发前，运行时根据量化格式、数据布局、张量形状和数据类型等约束选择专用路径或 RVV 路径；若不满足接口约束，则直接使用现有的 RVV 实现。

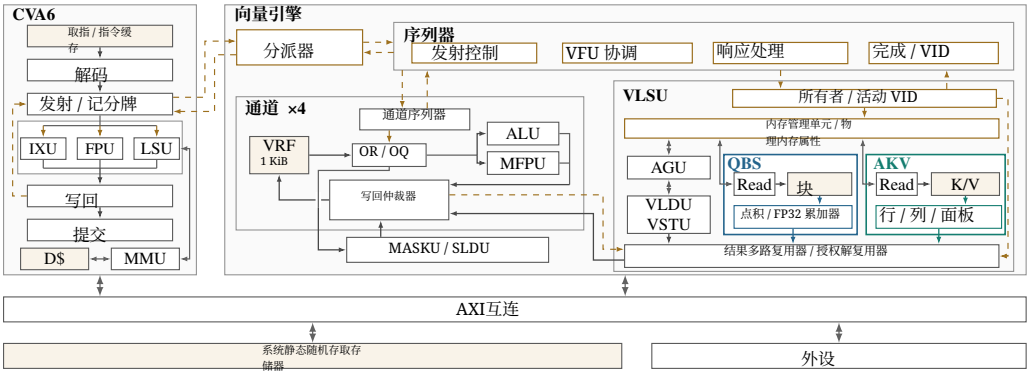


图 2. QBS 和 AKV 集成到 RISC-V 向量执行路径中。

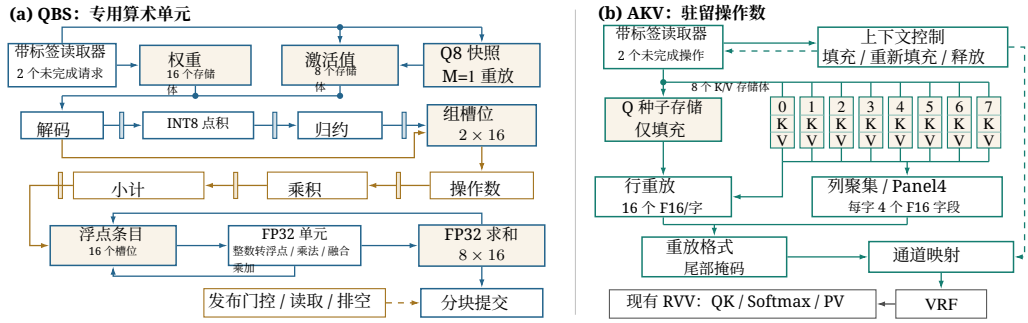


图 3. QBS 原生块执行与 AKV 驻留操作数数据通路。QBS 执行有序的原生块累加，而 AKV 为现有 RVV 路径提供驻留的 K/V 视图。

IV. 微体系结构

A. 共享执行、依赖关系与状态可见性

图 2 展示了 QBS 和 AKV 的整体微体系结构，以及它们与现有向量执行路径的共享集成方式。QBS、AKV 和普通向量内存操作共享 VLSU 地址转换路径、物理内存属性检查、AXI 接口和 Lane结果路径；序列器则继续使用现有指令标识符来跟踪目标寄存器依赖关系和完成状态。QBS命令和 AKV上下文操作在执行前，会等待较早的向量操作完成，并等待相关的普通内存事务排空；它们通过互斥仲裁使用共享的 VLSU 资源，确保读取响应、故障信息和返回结果始终与当前执行请求相关联。

QBS结果累加状态仅属于当前命令，与跨命令复用的激活快照相互独立；AKV键值状态可以跨多个视图指令持续存在，而每条视图指令都保留自身的目标寄存器、依赖关系和完成状态。两种机制都只有在执行成功后才会更新架构可见状态：QBS会在相关内存访问和计算完成后提交最终结果，而AKV仅在无故障地完成填充后才会将上下文标记为有效。在复用期间，QBS激活快照和AKV上下文不会重新访问相应的源地址，也不会自动跟踪源内存的变化，因此当源数据发生变化或复用结束时，软件必须重建或释放相应状态。

B. QBS 原生块执行与有序累加

如图 3(a) 所示，QBS数据通路依次执行原生块取指、格式解码、整数计算、FP32累加和结果提交。带标签读取路径可适配不同的原生块大小和权重布局；返回的数据按字节对齐，并在进入由相应配置选定的解码路径之前重新组装为完整的原生量化块。模型加载期间可以转换权重布局；例如，四行块交错布局（R4）会交错来自四个输出行、位于相同 K 位置的原生块，同时不改变其内部块编码，从而使权重读取顺序与多输出计算相匹配。解码后的数据进入四个并行的整数计算集群，每个集群包含八个有符号 INT8×INT8 乘法器，可并行执行 32 次整数乘法。对于 $M = 1$，2，4，各集群分别处理一行、两行或四行激活值；针对 $M = 5$–8 的 Wide模式复用同一权重瓦片，并分两轮处理激活行。整数点积结果在对应子组内归约，随后根据当前配置进行校正和缩放，形成 FP32块级更新。不同配置的主要区别在于格式解码和块级更新规则，而整数点积、归约和FP32累加资源则为各配置共用。

块级更新进入FP32 累加路径后，由标签跟踪进行中的更新，并防止对同一累加状态产生冲突。不同输出的更新可以交错进行，而同一输出的更新始终保持原生 K-块顺序。当前实现提供 128 个命令局部 FP32累加状态，使整个 $M \times N$ 结果瓦片在沿 K 归约期间始终保留在 QBS 中，避免每个原生块处理后都重复进行 VRF 回写；归约完成后，最终的 FP32 结果将写入目标向量寄存器组。

C. AKV 分银行存储与多视图操作数供给

如图 3(b) 所示，AKV 根据令牌索引将驻留 K/V 分块分布到八个单端口存储体中，其中令牌 t 映射到存储体 t mod8，K 和 V 占据不同的逻辑区域，同时保留外部行优先布局。此组织方式可直接支持按令牌读取 K/V 行，并按特征维度生成 K 列视图。对于 K 列访问，硬件固定一个特征维度；每轮中，八个存储体读取当前 8 令牌组对应的存储字，并提取相同位置的 F16 元素，因此包含T_{valid} 个有效令牌的列只需 $\lceil T_{\text{valid}}/8 \rceil$个存储体读取轮次。Panel4 还会在同一读取轮次中提取对应于四个相邻特征维度的 F16 字段，并生成四个列视图，无需增加 SRAM 读取端口。K/V 填充、行读取和列视图生成通过互斥仲裁共用相同的单端口存储体；有效位会屏蔽尾部块中的无效令牌。

V. 评估

A. 评估方法与基线

我们采用经过功能验证的 Ara 实现 [24]（已修正已知实现问题）作为基线。所有实验均使用四通道配置，VLEN=1024、ELEN=64，以及 128 位 AXI 接口。算子级测试使用周期精确 RTL 仿真，并采用从真实模型执行中采集的张量。配对实验使用相同的输入和数值验收标准，数据布局则根据相应的内核进行组织。我们报告执行周期、共享向量端口上的 AXI 响应字节数，以及已退役向量指令数，并使用内部硬件计数器分析主要执行约束。

对于投影，基线采用针对 Ara 优化的特定格式多输出 RVV 内核。*Compute* 包括动态激活量化和矩阵计算，而矩阵乘法指矩阵计算阶段；静态权重预打包在模型加载期间执行。对于注意力，我们比较 Direct RVV、GQA 感知分块式 RVV 和 AKV。Tiled RVV 已采用软件分块，并在多个查询之间复用 K/V，因此将其与 AKV 比较，可以衡量本地 K/V 驻留和操作数视图供给带来的增量收益。模型级功能运行用于验证不同模型和注意力配置下的运行时分派与回退。PPA 评估采用相同的架构参数和实现约束，以量化所提设计的实现成本和效率。

B. 量化投影性能与复用

图 4(a) 比较了多种代表性块量化格式下的投影性能。QBS 在 Compute 和矩阵乘法上分别实现了 35.68× 和 51.34× 的几何平均加速比，表明将格式处理、整数计算和结果更新组织为原生块执行，可显著降低量化投影开销。图 4(c) 从数据移动和指令执行两方面进一步印证了这一结果：所有受评估格式返回的 AXI 字节数均少于 RVV 基线，而已退役向量指令数则降至基线的仅 0.42%–0.56%。因此，性能提升不仅源于更高效的计算路径，也得益于数据移动减少和显式 RVV 指令开销降低。

图 5(a)进一步揭示了与格式相关的执行行为。Q3 K–Q6 K 主要受点积执行支配，其中 Q5 K 和 Q6 K 中的权重供给停顿更为明显。相比之下，Q8 0 更频繁的结果更新使 FP32 更新开销更加突出。图 5(b) 和图 6(a) 分别考察了激活值复用和权重复用。复用量化激活可将三个相同投影的总周期数降低 47.6%，而 Wide-M 在激活行之间共享权重，并在相同工作量下将输入负载减少 33.2%。前者避免了重复的激活量化，后者则减少了冗余的权重读取，从而进一步降低了激活准备和权重供给带来的投影开销。

C. 注意力性能与 *K/V* 供给

图 4(b)比较了解码期间的注意力延迟。相较于已经利用感知 GQA 的分块并在查询之间复用 K/V 的 Tiled RVV，AKV 实现了 1.45× 几何平均加速比。随着 KV 长度从 16 增加到 256，加速比从 1.22× 提升至 1.60×。由于软件基线已经显式利用了 K/V 共享，剩余的增益表明，K/V 的组织与传输仍构成显著的数据供给开销，并且随着上下文增长，其影响变得更加明显。

图 5(c) 进一步分解了这一增益。在 K/V 有效载荷保持不变的情况下，双查询共享与 Panel4 相结合，使列视图命令和本地重放分别减少了 81.7% 和 30.4%，同时使执行周期减少了 14.5%。因此，性能提升主要源于对驻留 K/V 的组织和供给更加高效，而非有效数据量或注意力计算本身有所减少。图6(b) 从存储访问层面解释了这一现象：Panel4 利用单个 256 位存储体字中已有的相邻 F16 字段，在保持单端口存储体组织不变的同时，将四个相邻列视图所需的存储体访问轮次从 32 次降至 8 次。因此，AKV 将专用化限定在 K/V 驻留和操作数视图供给上，而 QK、Softmax 和 PV 仍在现有 RVV 数据通路上执行。

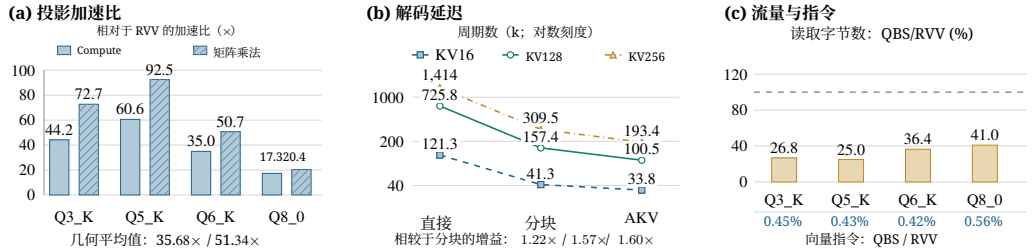


图4. 投影、解码和数据移动结果。(a) 对于 $M = 1$、$N = 256$，相较于 RVV 的投影加速比，其中 $K = 1536$适用于 Q3 K–Q6 K，$K = 896$ 适用于 Q8 0。(b) $K = 896$ 的解码延迟、GQA6 和两个 K/V 头。(c) QBS/RVV 的读取字节比和向量指令比。

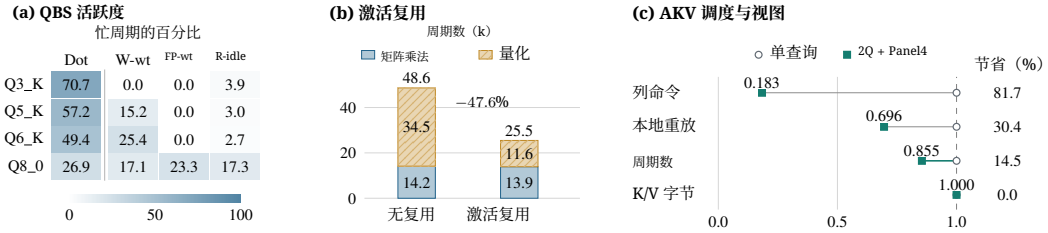


图 5。瓶颈与复用效应。（a）QBS 忙周期活动（W-wt/FP-wt/R-idle：权重等待/FP 条目等待/读响应空闲）。（b）三个 Q4 K 投影中的激活复用情况。（c）AKV 调度与视图供给计对 $M = 15$ 预填充，在 K/V 负载相同的情况下，以单查询 AKV 为基准进行归一化。

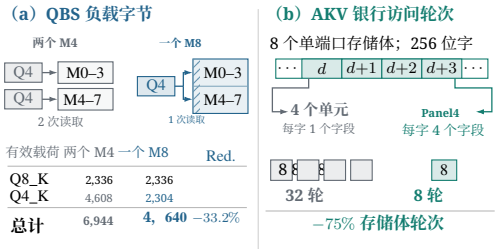


图6. QBS 和 AKV 中机制层面的复用：(a) Wide-M 权重有效载荷复用；(b) Panel4 存储体访问减少。数值表示数据供给工作量，而非执行延迟。

表III 功耗、性能和面积及效率对比。

指标	基准 Ara	本工作	相对变化
工艺技术	台积电 28 nm HPC+		–
频率 (GHz)	1.0		–
通道 / VLEN / 数据长度	4 / 1024 / 256		–
性能 (每秒十亿次运算)	0.615	32.29	51.34×
面积（mm^2）	3.195	3.599	+12.65%
面积效率（每秒十亿次运算/mm^2）	0.192	8.97	46.63×
功耗（mW）	177.0	471.7	+166.49%
执行能量（pJ/次运算）	312.8	16.91	−94.59%
能效（每秒十亿次运算/W）	3.55	70.18	19.78×
能量延迟积（归一化）	1.00	0.0010	−99.90%

注： 两种设计均采用台积电 28 nm HPC+（TT/0.9 V/25℃）工艺，工作频率为 1 GHz，配备四通道、VLEN=1024 和数据长度=256。

D. 模型适用性与运行时验证

本节通过跨配置 RTL测试和模型级功能运行评估设计的适用性。除主要评估中使用的 D128/GQA配置外，RefactD64/MQA32 [25] 和 Gemma 3 D256/GQA4 [26] 复用现有的 K/V 驻留和视图供给结构，相较于 Direct RVV 分别实现 2.04× 和 2.55× 倍加速。模型级功能运行涵盖 Qwen2.5 [27],、 Qwen3 [28],、 Refact 和 Gemma 3；所有案例均执行原生QBS命令，并验证 AKV运行时调度及回退功能，且全部通过功能检查。这些结果表明，无需更改核心硬件结构，即可通过软件调度适配不同的模型配置，从而在保持与现有执行路径兼容的同时支持多种模型配置。

E. 实现成本与效率

表III比较了在相同工艺技术、处理器配置和实现约束下，基线 Ara 与所提出设计的功耗、性能和面积（PPA）及效率结果。在 1GHz 工作频率下，总面积增加从 3.195mm^2 增至 3.599mm^2，，对应 12.65% 的面积开销；而整体性能达到基线的 51.34×，单位面积吞吐量提升 46.63×。相较于性能增益，额外计算、状态存储和数据供给逻辑带来的面积开销仍然有限，这表明额外硬件资源能够有效转化为处理吞吐量。所提出设计的平均功耗为基线的 2.67×，而能效从 3.55 提升至 70.18 每秒十亿次运算/W，对应 19.78× 的增益。性能提升幅度显著高于功耗增幅，这表明降低格式处理、显式向量执行和操作数供给的开销，能够有效摊销额外的硬件功耗成本，从而显著提升能效。

VI. 结论

本文提出 QBS 和 AKV，以利用大语言模型推理过程中跨指令持续存在的状态与复用关系。

QBS 将原生块格式处理、块遍历和有序 FP32 累加整合在一起；AKV 则通过本地 K/V 驻留和多视图供给，降低操作数组织开销，同时保留现有的 RVV 算术路径。寄存器传输级评估表明，这两种机制降低了量化投影和 K/V 供给开销，同时提升了单位面积吞吐量和能效。在多种量化格式和模型特定的注意力组织上进行验证后，结果进一步表明，该设计能够适应多样的大语言模型推理配置，同时保留统一的硬件结构和 RVV 可编程性。

参考文献

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell 等，“语言模型是少样本学习器”，载于《神经信息处理系统进展》，第 33 卷，2020 年，第 1877–1901 页。[2] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale 等，“Llama 2：开放基础模型与微调聊天模型”，2023 年。 [在线]。可访问：https://arxiv.org/abs/2307 09288 。B. Roziere, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, R. Sauvestre, T. Remez 等，“Code Llama：面向代码的开放基础模型”，2023 年。 [在线]。可访问：https://arxiv.org/abs/2308 12950 。A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser 和 I. Polosukhin，“注意力就是你所需要的一切”，载于《神经信息处理系统进展》，第 30 卷，2017 年。[5] RISC-V International，《RISC-V“V”向量扩展》，1.0 版，RISC-V International，2021 年。 [在线]。可访问：https://docs.riscv.org/reference/isa/extensions/vector/ attachments/riscv-v-spec.pdf [6] G. Gerganov 及贡献者，“llama.cpp：使用 C/C++ 实现大语言模型推理”，https://github.com/ggml-org/llama.cpp，2026 年，访问日期：2026-09-19。[7] M. Perotti, Y. Zhang, M. A. Cavalcante, E. Mustafa 和 L. Benini，“MX：增强 RISC-V 向量指令集架构，实现超低开销、高能效矩阵乘法”，载于 2024 年欧洲设计、自动化与测试会议暨展览会（DATE），2024 年，第 1–6 页。[8] M. Wipfli, G. Islamoglu, N. K. Purayil, A. Garofalo 和 L. Benini，“VMXDOTP：用于高效微缩放（MX）格式加速的 RISC-V 向量指令集架构扩展”，载于 2026 年欧洲设计、自动化与测试会议（DATE），2026 年，第 1–7 页。[9] SpacemiT，《RISC-V IME 集规范》，https://github.com/spacemit-com/riscv-ime-extension-spec，2026 年，1.0 版规范；访问日期：2026-09-19。[10] J. Haris 和 J. Cano，“F-BFQ：面向大语言模型的灵活块浮点量化加速器”，载于《应对大语言模型与图神经网络计算需求的新方法》（LG-ARC），与 ISCA 同期举办，日本东京，2025 年 6 月。 [在线]。可访问：https: //arxiv.org/abs/2510 13401 。G. Gerogiannis, S. Eyerman, E. Georganas, W. Heirman 和 J. Torrellas，“DECA：基于三维屋顶线模型的近核大语言模型解压缩加速器”，载于第 58 届 IEEE/ACM 国际微体系结构研讨会（MICRO），2025 年，第 184–200 页。[12] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebron 和 S. Sanghai，“GQA：基于多头检查点训练广义多查询变换器模型”，载于 2023 年自然语言处理经验方法会议论文集（EMNLP）。计算语言学协会，2023 年，第 4895–4901 页。[13] T. Dao, D. Y. Fu, S. Ermon, A. Rudra 和 C. R´e，“FlashAttention：具备输入输出感知能力的快速、内存高效精确注意力”，载于《神经信息处理系统进展》，第 35 卷，2022 年，第 16 344–16 359 页。[14] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth 和 S. Han，“SmoothQuant：面向大语言模型的准确高效训练后量化”，载于第 40 届国际机器学习会议论文集，系列：机器学习研究论文集，第 202 卷。PMLR，2023 年，第 38 087–38 099 页。 [在线]。可访问：https://proceedings.mlr.press/v202/xiao23c.html [15] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan 和 S. Han，“AWQ：用于端侧大语言模型压缩与加速的激活感知权重量化”，载于《机器学习与系统论文集》，第 6 卷，2024 年。[在线]。可访问：https://proceedings.mlsys.org/paper files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html [16] C. Hooper, S. Kim, H. Mohammadzadeh, M. W. Mahoney, Y. S. Shao, K. Keutzer 和 A. Gholami，“KVQuant：利用键值缓存量化实现千万级上下文长度的大语言模型推理”，载于《神经信息处理系统进展》，第 37 卷，2024 年。[17] T. Dao，“FlashAttention-2：通过更优的并行性和工作划分实现更快的注意力”，载于第十二届国际学习表征会议（ICLR），2024 年。 [在线]。可访问：https://openreview.net/forum?id=mZn2Xyh9Ec [18] J. Shah, G. Bikshandi, Y. Zhang, V. Thakkar, P. Ramani 和 T. Dao，“FlashAttention-3：具备异步性和低精度特性的快速、准确注意力”，载于《神经信息处理系统进展》，第 37 卷，2024 年。

[19] V. Titopoulos, K. Alexandridis 和 G. Dimitrakopoulos，“在 RISC-V 向量处理器上实现向量化 FlashAttention，并采用低成本指数运算”，《超级计算杂志》，第 82 卷，第 4 期，第 189 页，2026 年。[20] S. R. Gupta, N. Papadopoulou 和 M. Peric`as，“面向可扩展向量架构的 FlashAttention”，2026 年。 [在线]。可访问：https: //arxiv.org/abs/2608 18656 。Z. Liu, J. Yuan, H. Jin, S. Zhong, Z. Xu, V. Braverman, B. Chen 和 X. Hu，“KIVI：一种无需调优的键值缓存非对称 2 位量化方法”，载于第 *41* 届国际机器学习会议论文集，系列：机器学习研究论文集，第 235 卷。PMLR，2024 年，第 32 332–32 344 页。 [在线]。可访问：https://proceedings.mlr.press/v235/liu24bz.html [22] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang 和 I. Stoica，“借助 PagedAttention 实现大语言模型服务的高效内存管理”，载于第 *29* 届操作系统原理研讨会论文集（*SOSP*）。ACM，2023 年，第 611–626 页。[23] H. Wu, C. Xiao, J. Nie, X. Guo, B. Lou, J. T. H. Wong, Z. Mo, C. Zhang, P. Forys, C. Ai, T. Adeniran, W. Luk, H. Fan, J. Cheng, T. M. Jones, R. Antonova, R. Mullins 和 A. Zhao，“应对内存墙：长上下文智能体大语言模型推理的优化路径”，载于第 *53* 届 *IEEE* 年度国际计算机体系结构研讨会（*ISCA*）论文集。IEEE，2026 年，第 100–115 页。[24] M. Perotti, M. Cavalcante, R. Andri, L. Cavigelli 和 L. Benini，“Ara2：探索采用高效且符合 RVV 1.0 标准的开源处理器实现单核和多核向量处理”，《*IEEE* 汇刊：计算机》，第 73 卷，第 7 期，第 1822–1836 页，2024 年。[25] Refact.ai，“Refact-1.6B-fim”，Hugging Face 模型仓库，2023 年，访问日期：2026-09-21。 [在线]。可访问：https://huggingface.co/ refactai/Refact-1 6B-fim [26] Gemma 团队，“Gemma 3 技术报告”，2025 年。 [在线]。可访问：https://arxiv.org/abs/2503 19786 。A. Yang 等，“Qwen . 技术报告”，2024 年。[27] [在线]。可访问：https://arxiv.org/abs/2 5 2412 15115 。——，“Qwen 技术报告”，2025 年。 [在线]。可访问：https: //arxiv.org/abs/2505.09388 [28] 3