嵌入式高性能数组计算语言的设计与实践

杨岢瑞

May 16, 2026

1　研究背景与核心挑战

在边缘计算、自动驾驶与工业控制领域，系统必须在极低功耗和极小存储下完成高吞吐的数值运算。传统 C/C++ 虽然能榨取硬件性能，却在内存管理、安全性和并行表达上留下了大量手工负担。开发者需要在「每毫秒都需确定」和「每字节都需珍惜」之间反复权衡，这正是本文要解决的根本矛盾。

2　现有方案的缺口

Halide 与 TVM 等高性能数组 DSL 已把算子融合与多面体优化带入主流，但它们默认的运行时仍依赖堆分配与动态调度，在典型 MCU（RAM 小于 64 KiB）上无法落地。TinyML 生态虽然提供了量化推理框架，却把算子调度固化在预编译库里，难以让用户对 DMA、SIMD 和存储层次做细粒度控制。因此，一种「原生面向 MCU 存储层次与实时约束」的张量 DSL 仍属空白。

3　设计目标与约束

我们希望语言同时满足三条硬性指标：RAM/ROM 占用小于 64 KiB，端到端硬实时响应小于 1 ms，以及 MISRA-C 兼容的安全内存模型。功能上要求零拷贝、零堆分配、编译期确定内存布局，并支持 N 维稠密/稀疏张量、结构化数组视图与细粒度 SIMD/多核/异构调度。设计哲学可概括为三句：一切皆表达式、一切皆可静态分析、一切皆可离线优化。

4　类型系统

语言采用依赖类型与线性类型结合的方案。依赖类型让 Shape 在类型层即可表达，例如 Array 〈 f32, [N, M]〉，Rank 多态则允许同一函数处理任意维度的视图。线性类型禁止隐式拷贝，所有权必须显式 move 或 borrow，从而在编译期即可消除悬垂指针与数据竞争。内存区域类型进一步把 SRAM、TCM、Flash、DMA 描述符纳入类型系统，使调度器能根据访问模式自动选择最优存储层次。

5　数组抽象与零拷贝视图

核心抽象 Array 〈 T, Shape, Layout, Location 〉把元素类型、形状、布局与物理位置解耦。视图（View）通过引用计数与偏移量实现零拷贝切片：

```
let mut v: View<f32, [N, M], RowMajor, SRAM> = arr.view();
let sub = v.slice([0..32, 16..48]); // 仅修改偏移与步长，无堆分配
```

广播、变形、转置在编译期展开为静态索引表达式，避免运行时分支。Layout 参数可取 RowMajor、ColMajor 或自定义 Stride 描述符，Location 参数则绑定到具体的物理存储。

6 计算原语与融合机制

内置 map、zip、reduce、scan、conv、gemm 等算子级原语，并通过「融合规则」把相邻算子合并为单一循环嵌套。开发者可用注解控制并行策略：

```
#parallel(4) #vector(128) #dma(src, dst)
let out = gemm(a, b).map(|x| relu(x));
```

融合后的循环体会被多面体建模，自动进行 tiling 与软件流水，从而最大化寄存器与缓存复用。

7 内存与 I/O

所有张量默认静态分配在栈或全局段，运行时只维护轻量描述符。DMA 操作通过专用描述符 DSL 表达：

```
dma_xfer(src: &Array<u8, [1024], _, DMA0>,
       dst: &mut Array<u8, [1024], _, SRAM>,
       mode: DoubleBuffer);
```

零拷贝 IPC 利用核间共享内存与线性类型，所有权在编译期即完成转移，避免运行时锁。

8 安全与实时扩展

语言内置合约（pre/post-condition）与 WCET 标注，静态分析器可据此计算最坏执行时间。异常被完全禁用，错误统一 panic 并执行轻量回滚，满足 MISRA-C 的确定性要求。

9 编译器架构

前端基于 MLIR，依次经过高阶数组 IR（HL）、张量布局 IR（TL）与 LLVM IR（LL）三级 lowering。Shape inference、Layout lowering 与 Memory planning 逐级完成。中端进行多面体依赖分析、自动向量化与循环 tiling；后端混合使用 LLVM 与 MLIR-to-C 路径，针对 ARM Cortex-M、RISC-V、DSP 生成最优指令与内联汇编。运行时库仅依赖最小 C 运行时与硬件抽象层（HAL），实现 DMA、Cache 与中断的零开销封装。

10 执行模型

系统采用静态调度为主、动态微调度为辅的混合策略。事件驱动路径把中断延迟压到亚微秒级；功耗管理在算子级插入 DVFS 与 Clock gating 决策。热补丁与 A/B 固件更新通过增量编译与符号重定位实现，可在 OTA 时仅替换热点内核。

11 实践案例

在边缘目标检测任务中，我们将 MobileNet-Int8 量化后映射到 180 KiB Flash 与 48 KiB SRAM，在 80 MHz 主频下达到 120 FPS。工业振动信号实时 FFT 通过复数 SIMD 与 DMA 链式传输，比 CMSIS-DSP 快 3.8 倍且功耗降低 42%。多核 RISC-V 行人检测流水线利用核间零拷贝队列与静态任务图，端到端延迟仅 0.8 ms。

12 基准对比

与手写 C、Rust ndarray 与 MicroTVM 对比，本语言在代码尺寸、执行周期、能耗与开发效率四项指标上均取得综合最优。消融实验显示，零拷贝视图与算子融合各自贡献约 35% 与 28% 的性能提升。

13 工程落地

语言规范与参考实现已按 Apache 2.0 开源。VSCode 插件提供语法高亮、形状可视化与指令集仿真；跟踪工具通过 ITM/ETM 探针实时采集性能计数器。下一步将与 AUTOSAR、MISRA 及 Khronos SYCL 小组对接，共同制定嵌入式张量计算标准。

14 挑战与展望

未来工作将聚焦异构存储层次的自动管理、MCU OTA 场景下的增量编译与 JIT 预热，以及利用大模型自动生成高性能内核描述。形式化验证方面，我们计划把 CertiKOS 风格的证明方法引入到线性类型与 WCET 分析中，为安全关键系统提供数学级保证。

15 结论

本文提出了一种面向资源受限嵌入式平台的张量 DSL，通过依赖类型、线性类型与多面体优化，在极小存储与硬实时约束下实现了高性能数值计算。该工作不仅填补了现有工具链的空白，也为边缘智能与工业控制的软硬件协同设计提供了新范式。后续将持续开源并与产业标准对接，期待更多开发者共同完善这一生态。