GMP

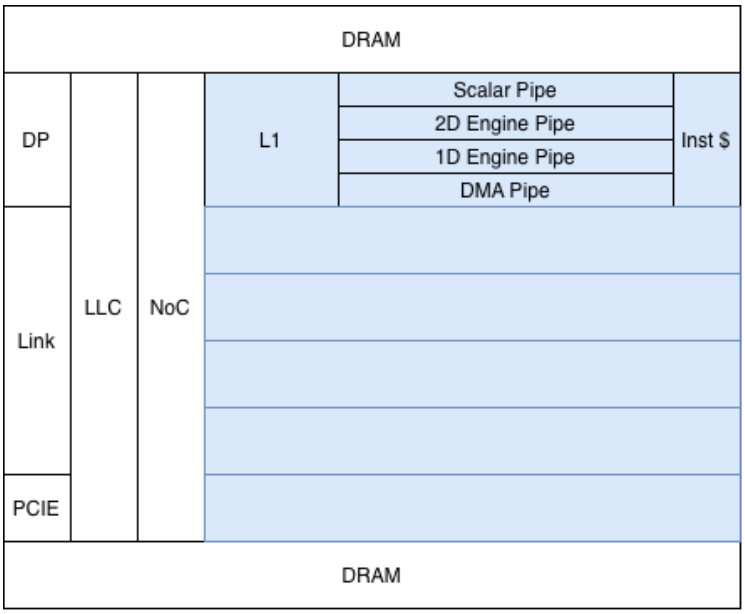
1. 背景
    1. 适应未来的AI计算需求
        1. 存储足够量的权重，但是明显的热点内容访问
        2. 强动态性，大范围、多次的随机动态访问
        3. 节能、低带宽需求，高效率数据流
        4. 低延迟
    2. 软件定义硬件，硬件尽量简单、透明、底层抽象、灵活
2. 目标
    1. 软硬件结合
        1. 大量依靠软件优化，发挥最大的物理效率，同工艺下架构效率达2倍
        2. 算法和硬件协同优化，同模型精度条件下效率达4倍
            1. DRAM或者多机的不确定数据延迟直接整合到算法处理，硬件不做竞争
            2. 硬件采用固定的LUT计算（可能不能等价到矩阵乘法，甚至是乘法本身）
            3. 全模型网络级别优化，利用编译器对整个模型进行搜索优化，生成静态计算图
        3. 指令控制流水线
            1. 指令明确指定指令的调度、**L0 Cache**的使用、依赖关系的建立和解除
    2. 原生支持动态算法：
        1. MoE
        2. 动态算法
    3. 统一的异步通讯管理方案
        1. 整个系统有大量的不同的通讯和同步机制
            1. 流水线内的credit，L1的数据缓冲
            2. 算力核之间的数据交互核同步，NoC的各种协议
            3. L2/L3的复用
            4. 分布式栈：网卡的片上调度，网络的延迟不确定性，通路的复用
        2. outstanding/各种缓存的管理和设计
        3. 异步的调度
        4. launch控制及指令加载加速
    4. 统一的数据流拆分模型
        1. 提供统一的编程模型对数据流进行描述
        2. 硬件加速的数据流动态计算，减少冗余且高成本的除法/模运算的地址计算
            1. 通过自动的预计算和特殊硬件加速
        3. 自动处理的原子操作，以消除写入全局内存时的warp级串行化
        4. 自动进行乒乓双缓冲机制
    5. 权重和临时数据（KV Cache ..）分离的架构
        1. 权重使用固定的通路
        2. 两种数据使用不同的数据流模版进行设计
    6. 算力
        1. **Binary Lut** 方案，减少算力能耗和面积，降低数据搬运量?
        2. **CIM** 专用权重通道，近存架构，存内架构，存内计算?
    7. 1D动态算力架构

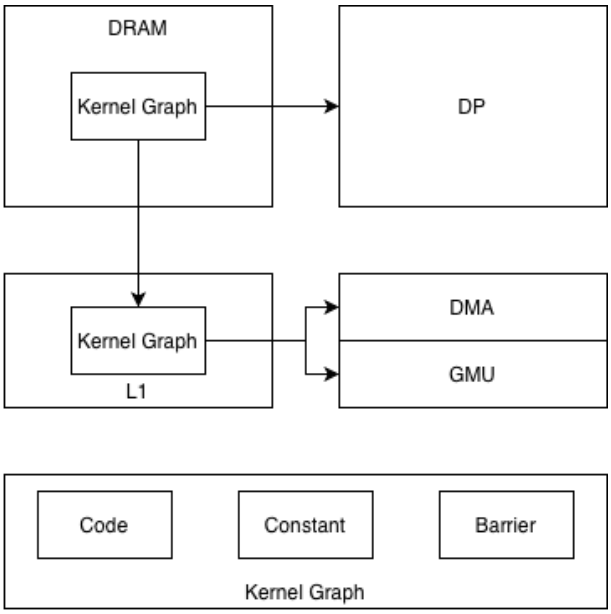
1. 一维计算阵列，动态性体现在一维的长度上，对应到 sequence的长度
2. 二维数据复用，数据通道提供精密的排布和数据复用的调度
3. 编译出向量指令流（LD MUL ST），支持批量动态配置，两块指令流之间流水可以重叠
   ---> DSA?
4. 例化固定数量的 L1读写计数器，用于同步 dma和 Mac，dma和 Mac，自动从 l1 加载指令
5. 问题：
   1. 1D指令表达大算力 2D
      1. VLD VST MLD MST VMUL VMUL_reduce MUL_join 等等 指令
   2. LD/ST的自动 mbarrier
      1. 软件管理 L1的所有 bank，每个 bank的每个地址都记录一个 count，启动初始化的时候初始化 count

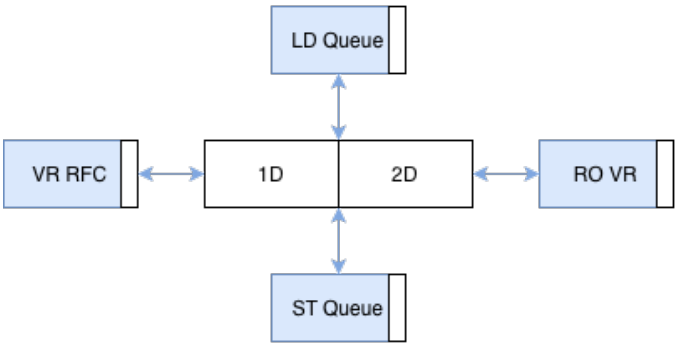
3. 架构考虑
   1. 动态性的表达
   2. 硬件竞争的管理
   3. 灵活性，扩展性，从 edga到集群
   4. 自举，所有单元支持自配置，自启动
   5. 线程内的依赖都是静态的软件调度，软件直接调度流水线，减少硬件的调度
4. 架构方案
   1. 图：编译整个动态计算图，支持 Fork（launch） join sync
   2. 平铺：按照可编程硬件单元进行编程，为每个单元生成一定数量的逻辑线程，支持 sync
   3. … …
5. 规格
6. 指令流
   1. load和 fetch
   2. 基于图的信息，和数据流一样得方式，需要发命令和同步
7. ISA
8. Launch
9. Sync

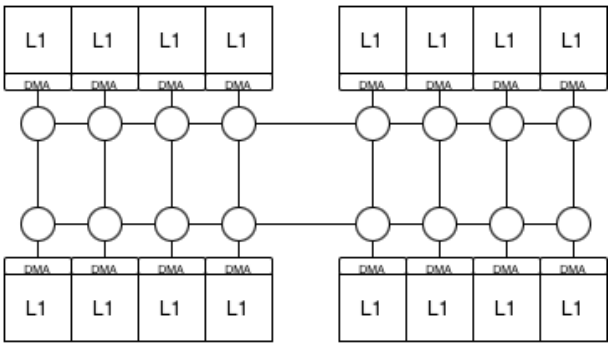
总结

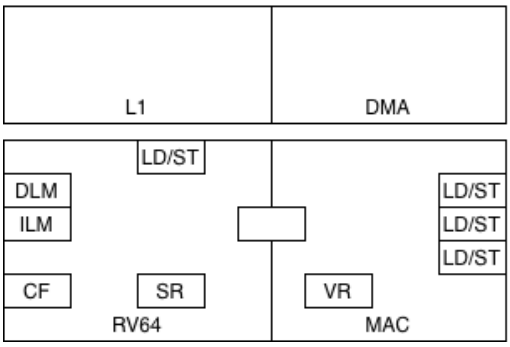
1. 灵活、低延迟的异步/同步机制
2. 简单、透明的硬件拓扑，软件的深度控制











Revision #46
Created 2025-03-16 06:37:22 UTC by Colin
Updated 2026-05-29 11:07:21 UTC by Colin