TESLA

DOJO

The Microarchitecture of Tesla's Exa-Scale Computer

Core 核心

Emil Talpes, Douglas Williams, Debjit Das Sarma

AI 芯片 - NPU详解

Talk Overview

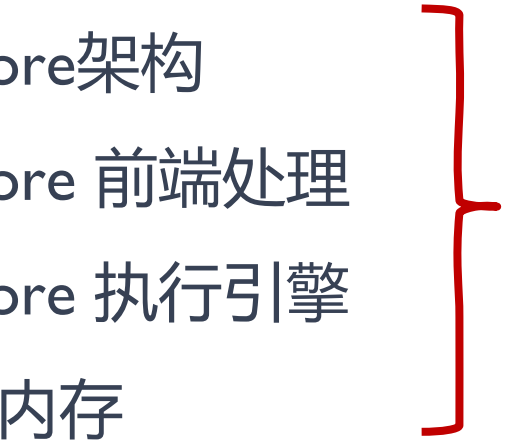
1. AI 计算体系
   - 深度学习计算模式
   - 计算体系与矩阵运算
2. AI 芯片基础
   - 通用处理器 CPU
   - 从数据看 CPU 计算
   - 通用图形处理器 GPU
   - AI专用处理器 NPU/TPU
   - 计算体系架构的黄金10年

1. **华为昇腾 NPU**
   - 达芬奇架构
   - 昇腾AI处理器
2. **谷歌 TPU**
   - TPU 核心脉动阵列
   - TPU 系列架构
3. **特斯拉 DOJO**
   - DOJO 架构
4. **国内外其他AI芯片**
   - AI芯片的思考

Talk Overview

1. 基本内容

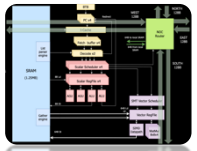
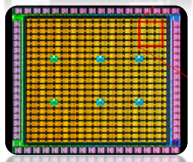
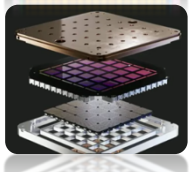
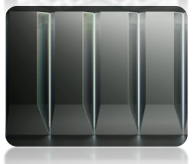
- DOJO 整体架构
- DOJO Core架构
- DOJO Core 前端处理
- DOJO Core 执行引擎
- SRAM 与内存
- 内核与物理实现
- 问题与思考



DOJO
整体结构回顾

DOJO 架构设计哲学

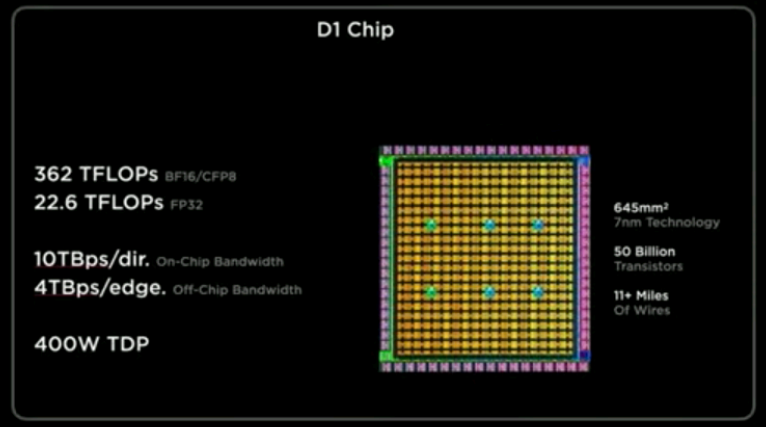
- DOJO 采用**存算一体架构（“存内计算”或者“近存计算”）**，单个可扩展计算平面、全局寻址快速存储器和统一的高带宽+低延迟。

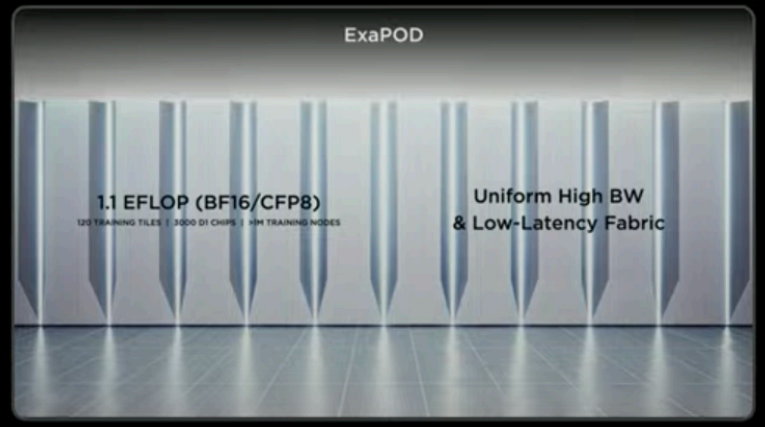





分层	名称	片上SRAM	算力	备注
内核	DOJO Core	1.25MB	1.024 TFLOPS	单个计算核心，64bit，4个8x8x4矩阵计算核心，2GHz主频
芯片	DOJO D1	440MB	362 TFLOPS	单芯片，354核心数，654mm2
格点	DOJO Tile	11GB	9050 TFLOPS	单个训练模组，每5x5个芯片组成一个训练模组
集群	ExaPOD	1320GB	1.1 EFLOPS	训练集群，每12个训练模组组成一个机柜，每10个机柜组成一个ExaPOD，一共3000个D1芯片

DOJO 超级计算机系统

- 可以把一个 DOJO Core 当做一台具有 CPU 专用内存和 I/O 接口的PC，每个 Core 都可以执行独立任务，不依赖于共享 Cache 或 Register File。
- D1 作为超标量（Super Scalar）内核，这意味着它在内核支持指令级并行，使用更多线程来驱动更多指令通过 DOJO Core，多线程执行意味着每时钟周期执行更多任务。

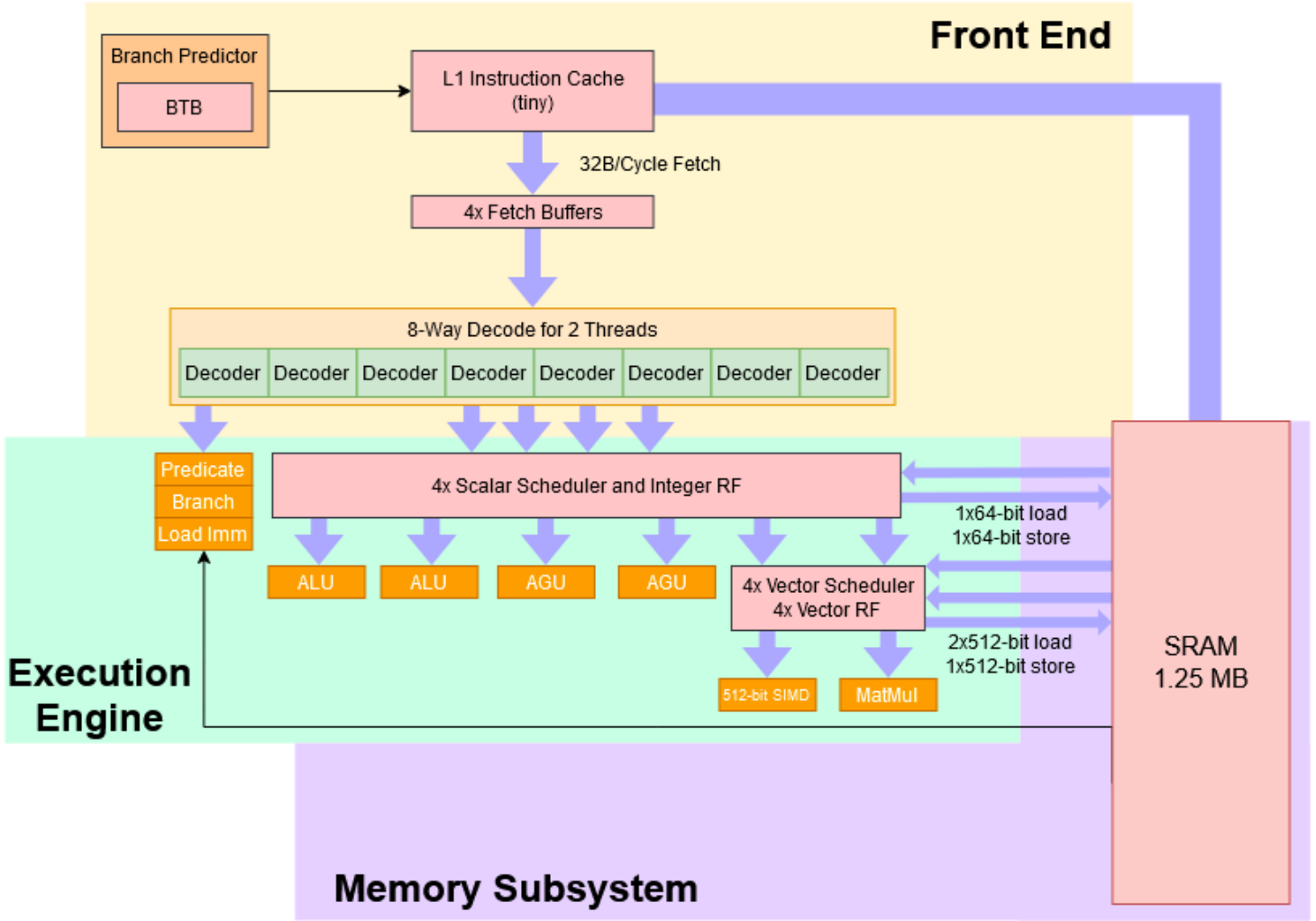




DOJO Core
整体结构

DOJO Core 主要参数

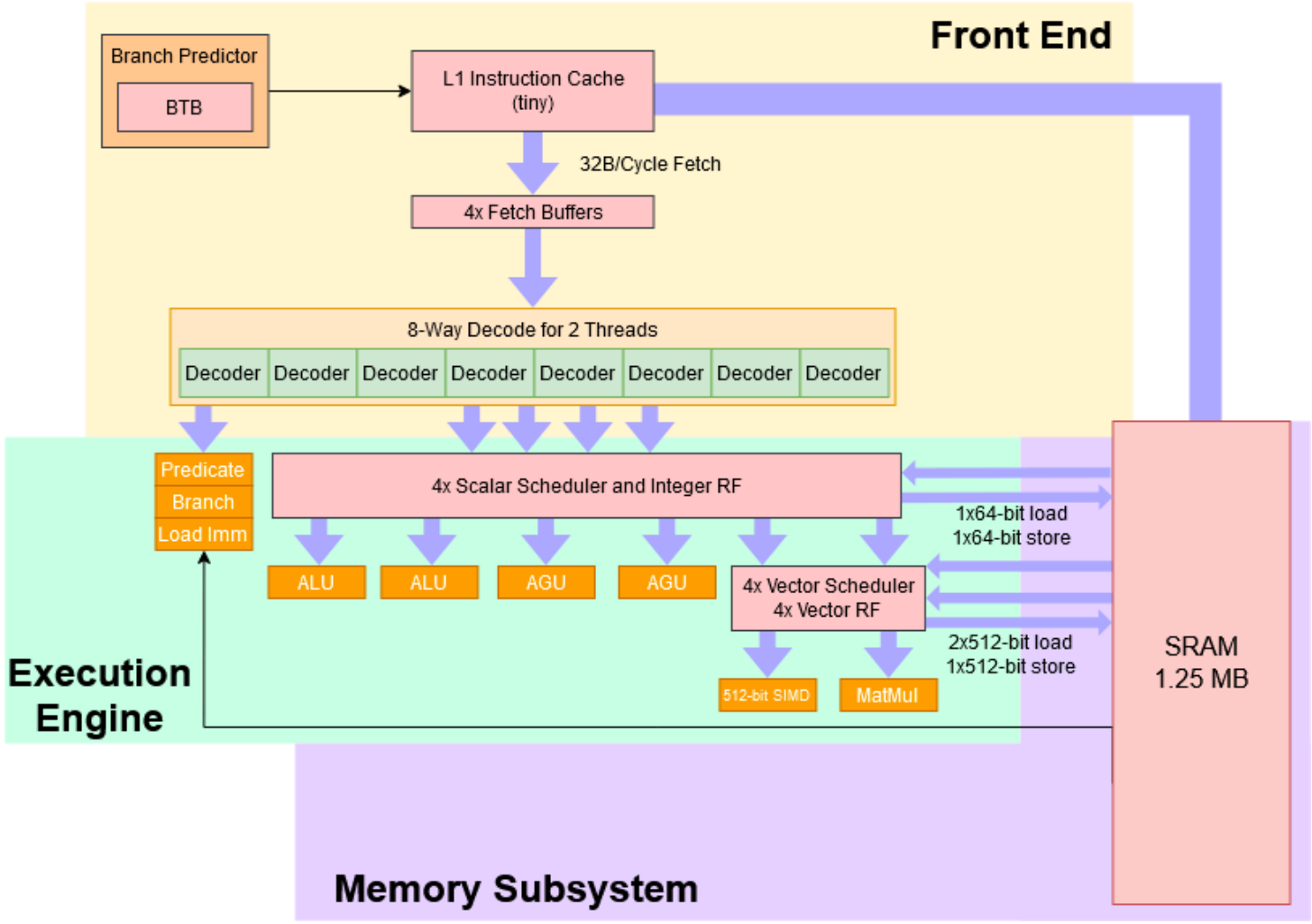
数据格式	FP16/FP32 BFP16/CFP16/CFP8
位宽bit width	64b
SMT	4路
MM单元	4路 8x8
主频	2GHz
取指	32B
SIMD	64B
SRAM Size	1.25MB
SRAM Rate	read: 400 GB/s write: 270 GB/s



DOJO Core 的结构和特点

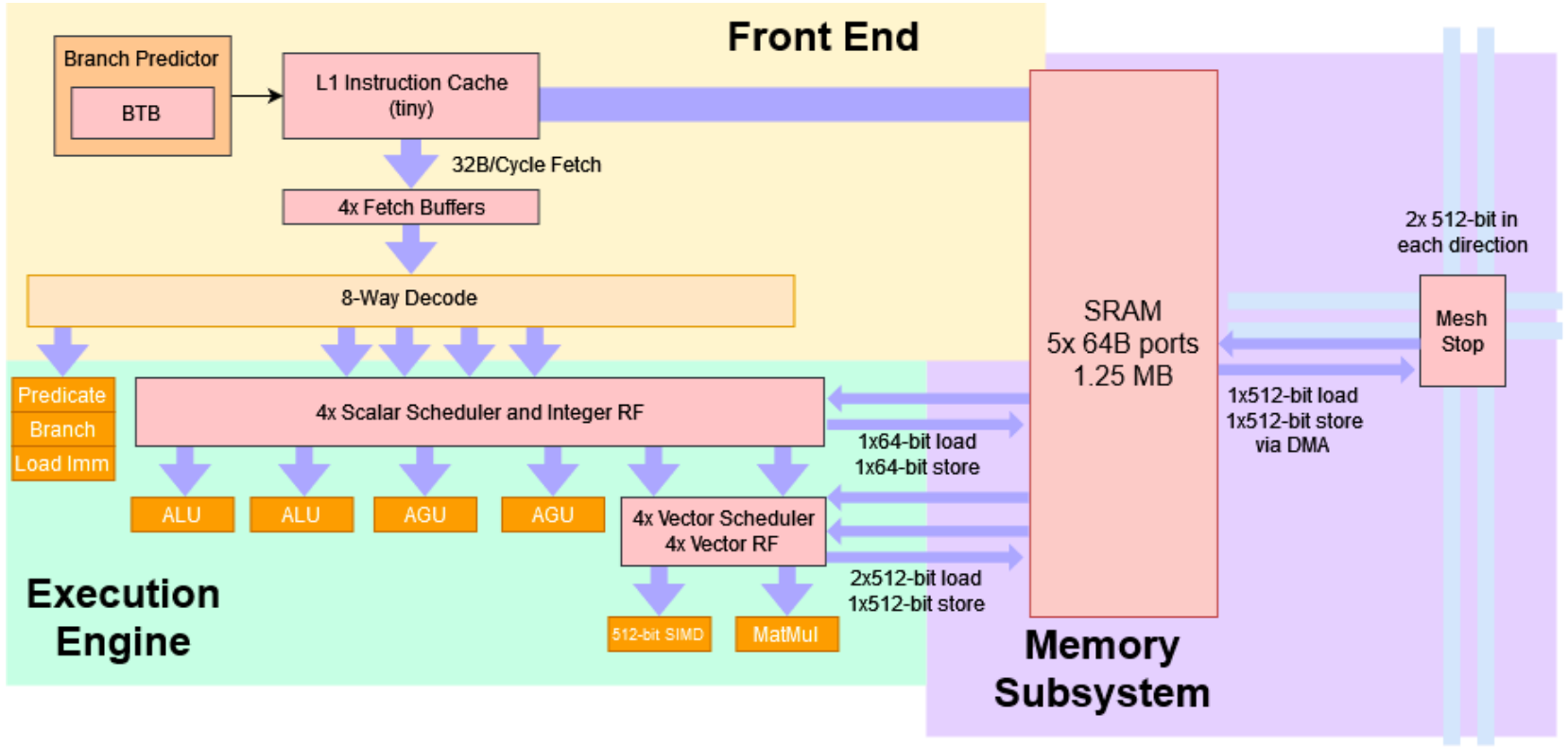
- 每个 DOJO 核心是带有向量计算/矩阵计算能力的处理器，具有完整的取指、译码、执行部件；

- D1 指令集类似于 RISC-V，频率 2 GHz，4 组 8x8 矩阵乘法计算单元，具有一组自定义向量指令 AI 计算。



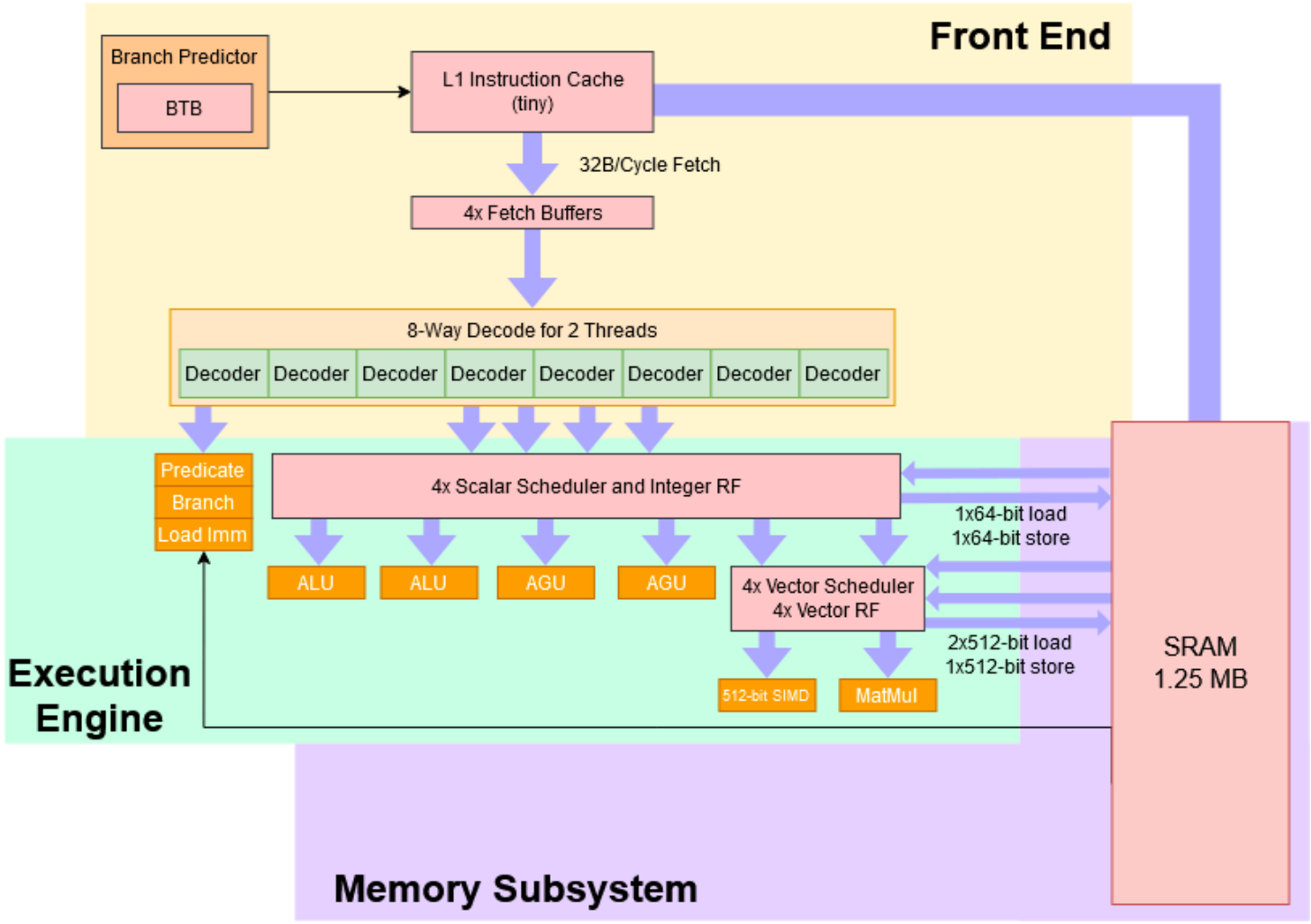
DOJO Core 的结构和特点

- DOJO Core 由前端、执行引擎、SRAM 和 NoC 路由4部分组成，比CPU和GPU的控制部件都更少，具有类似CPU的AGU和类似GPU Tensor core 的矩阵计算单元。



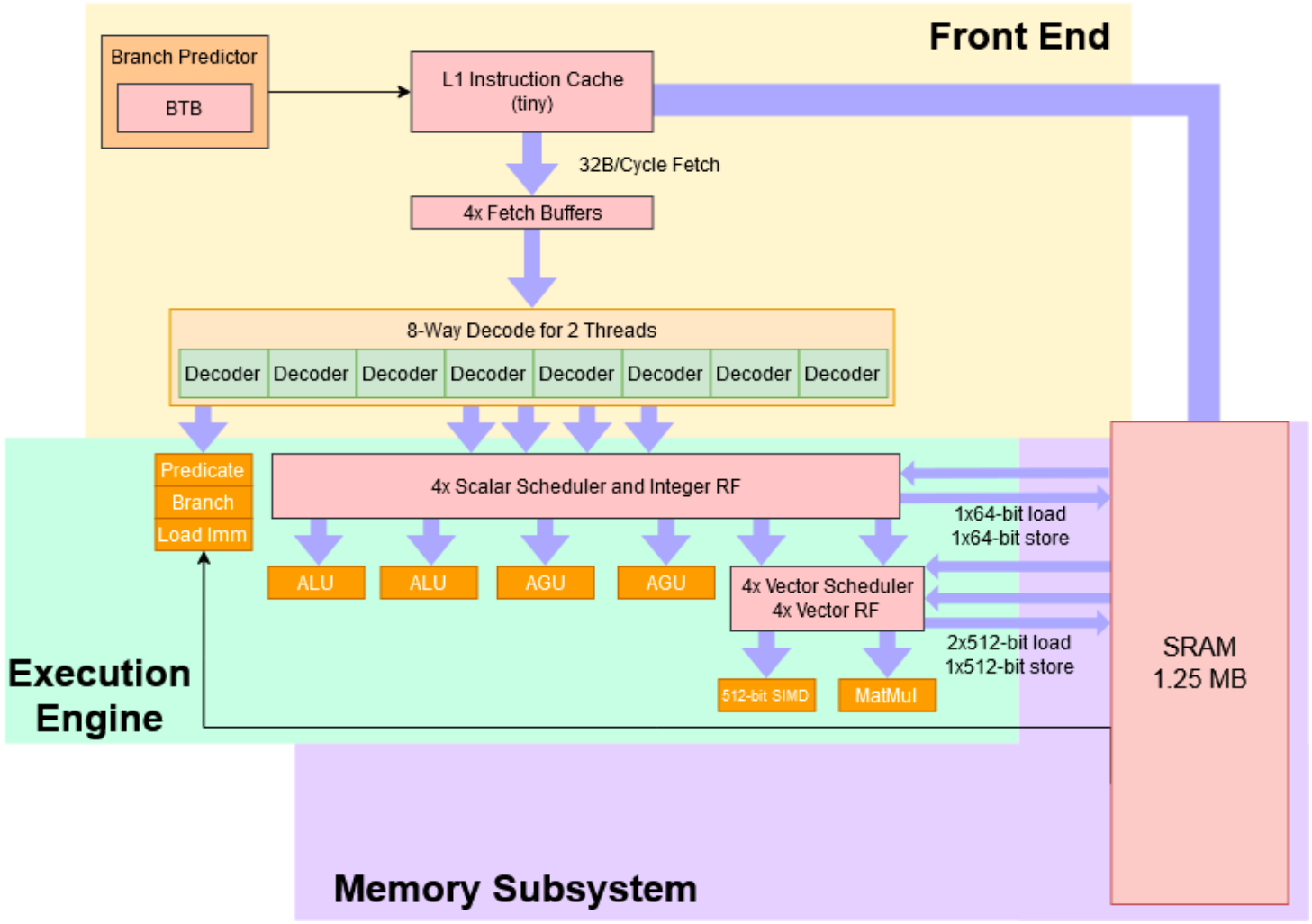
DOJO Core 的结构和特点

- 取消了 Rename 等改善执行部件利用率的组件，同时也不支持虚拟内存。能顾有效减少控制部分占用的面积，把芯片上更多的面积划分给计算执行单元。
- 每个DOJO Core 提供 1.024 TFLOPS 算力，几乎所有的算力都由矩阵计算单元提供。因而矩阵计算单元和 SRAM 共同决定了 D1 处理器的计算能效比。



DOJO Core 的结构和特点

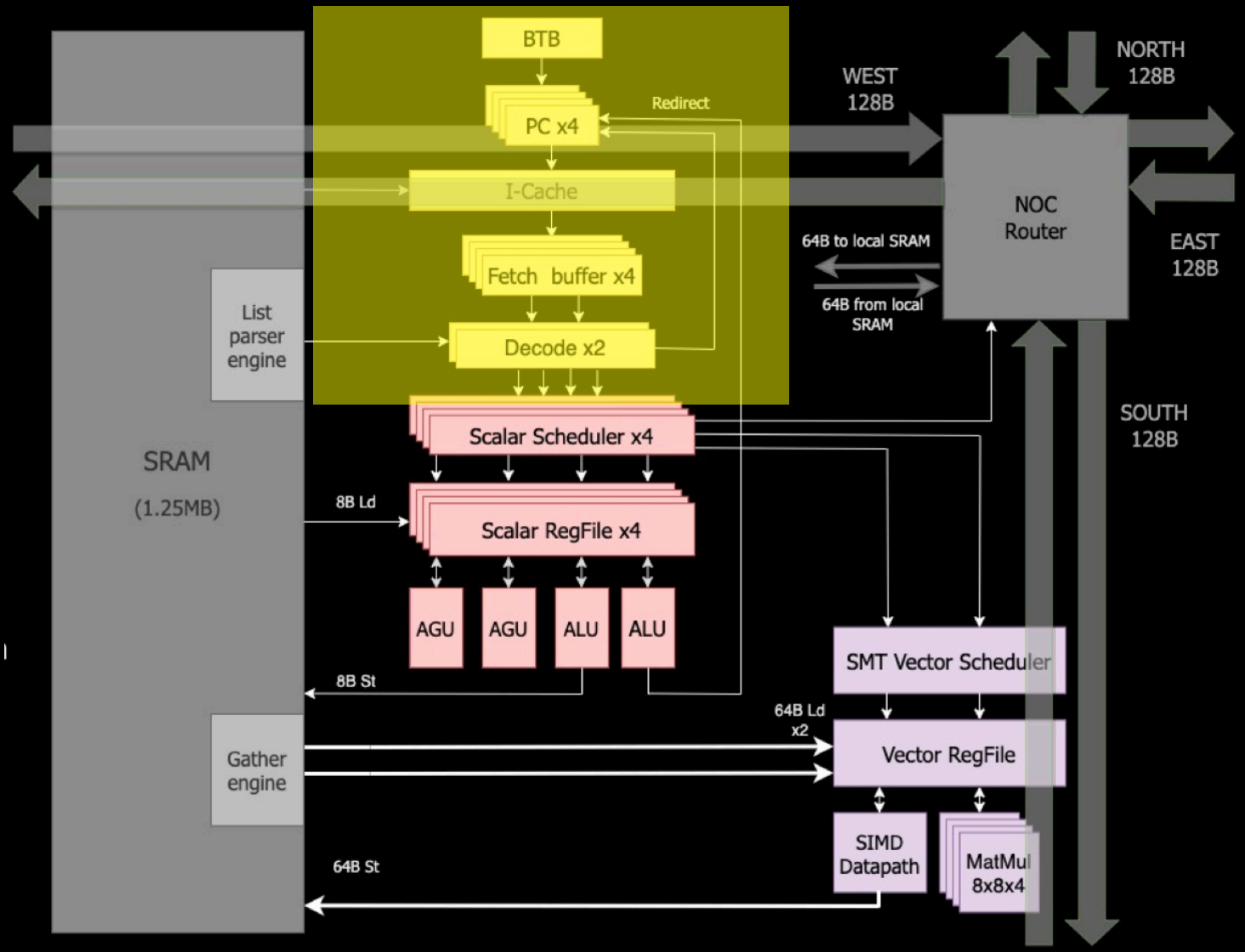
- DOJO Core 针对 Vector 计算进行优化，不过依赖于单独的主机处理器进行工作分配。
- DOJO 上运行的代码不能直接访问系统内存。相反，应用程序预计主要在一小部分本地 SRAM 中工作。
- 本地 SRAM 由软件（编译器）进行管理，不能用作缓存。如果需要来自主存储器的数据，则必须使用 DMA 操作将其引入。



DOJO Core
前端处理

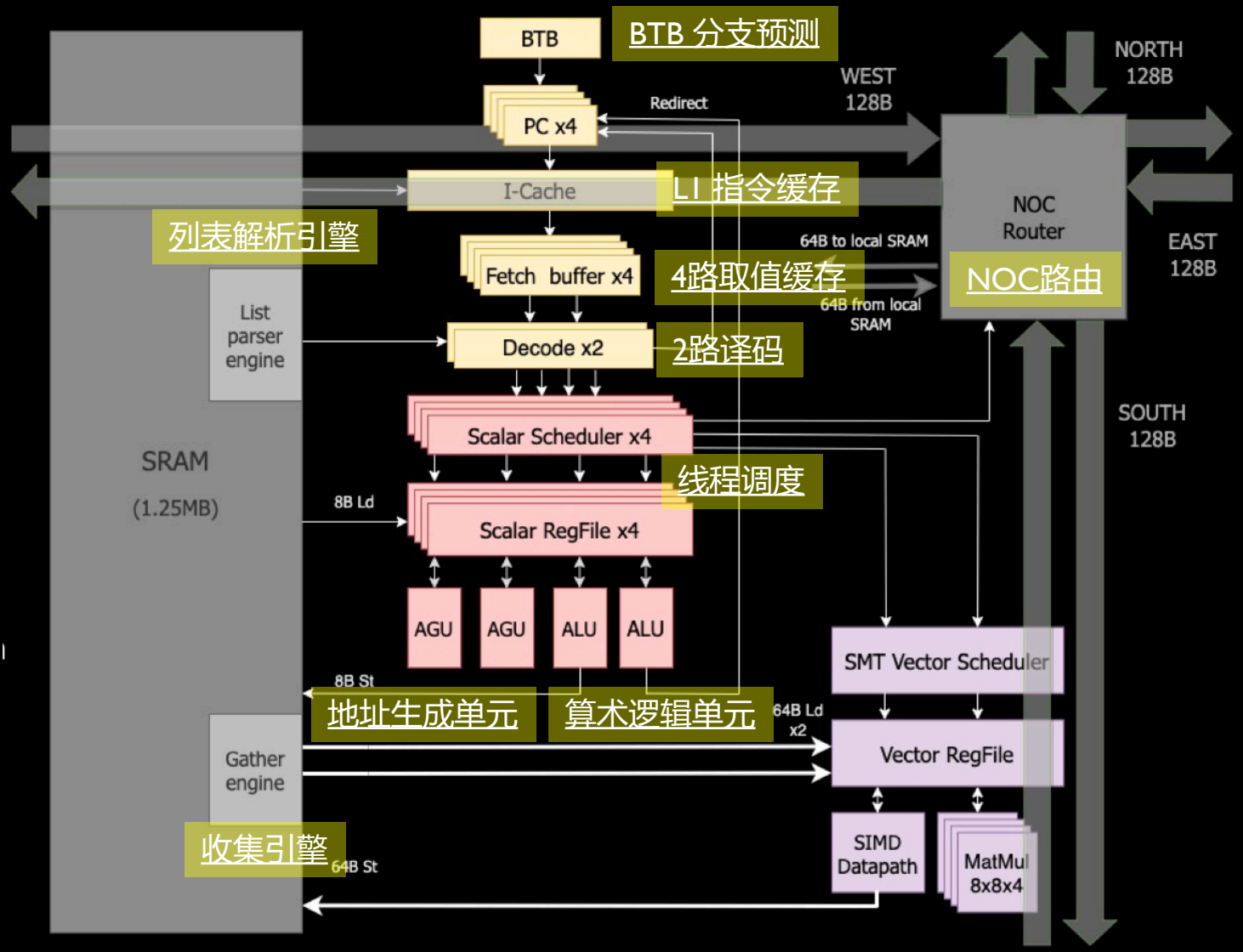
前端模块

- 64 位处理器，具有 32B 取指窗口（ fetch window ），最多可容纳 8 条指令；8路解码器每个周期可以处理两个线程。从提取缓冲区中，Core 中解码器每个周期可以处理两个线程的8条指令。
- DOJO 将解码器分成两个，并选择两个线程在每个循环中为其提供数据，会减少所采用分支的解码吞吐量损失。在解码时，某些指令（branches, predicated operations等）可以在前端执行并从pipeline中删除。



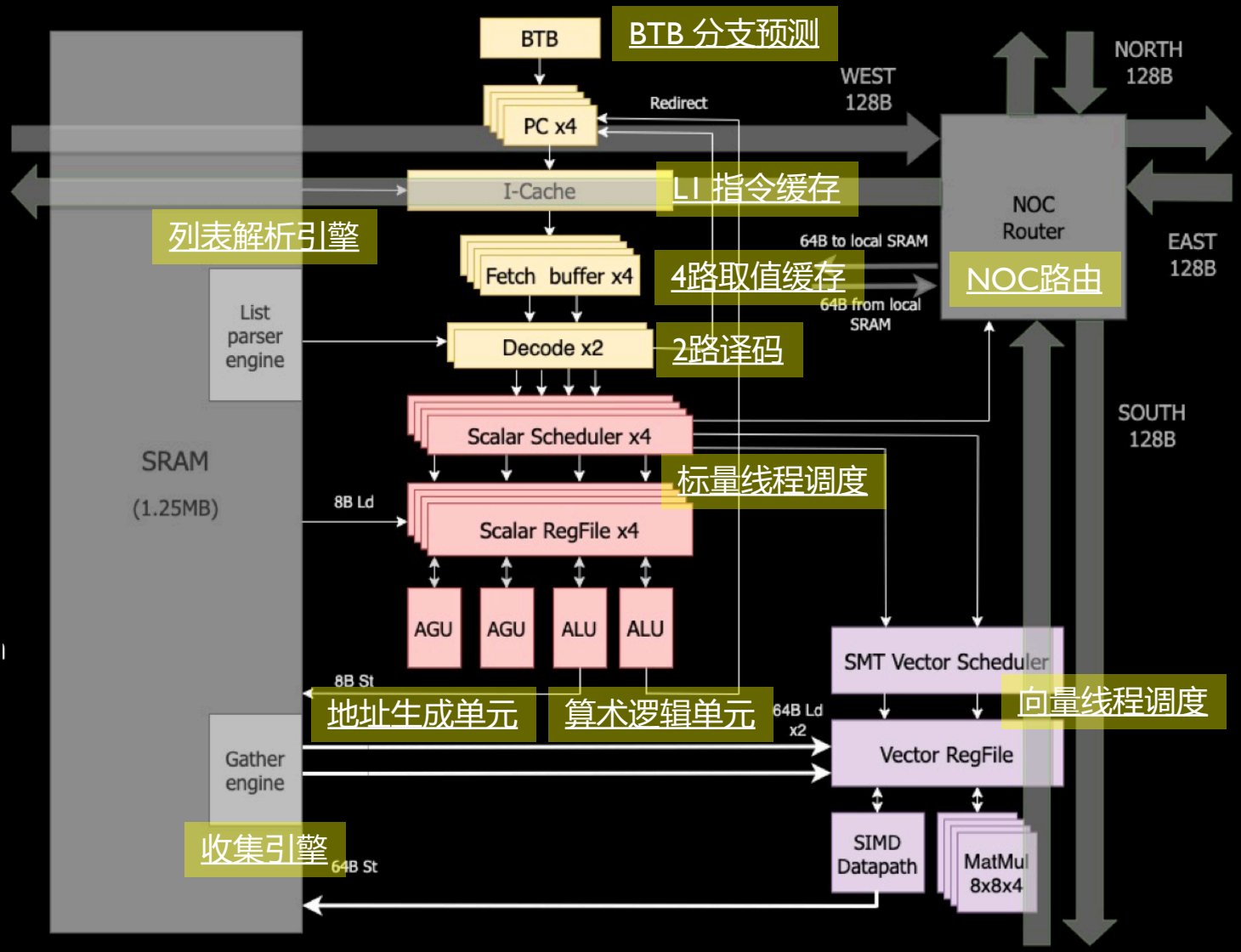
前端模块

- **分支预测**：DOJO Core 核心没有 SIMT 堆栈核心来进行多线程分支任务的分配，通过 BTB（分支目标缓冲区）可以进行简单分支预测来提升性能。
- BTB 将分支成功的分支指令地址和分支目标地址都放到一个缓冲区中保存起来，缓冲区以分支指令地址作为标识。通过预测分支的路径和缓存分支使用的信息来减少流水线处理器中分支的性能损失。



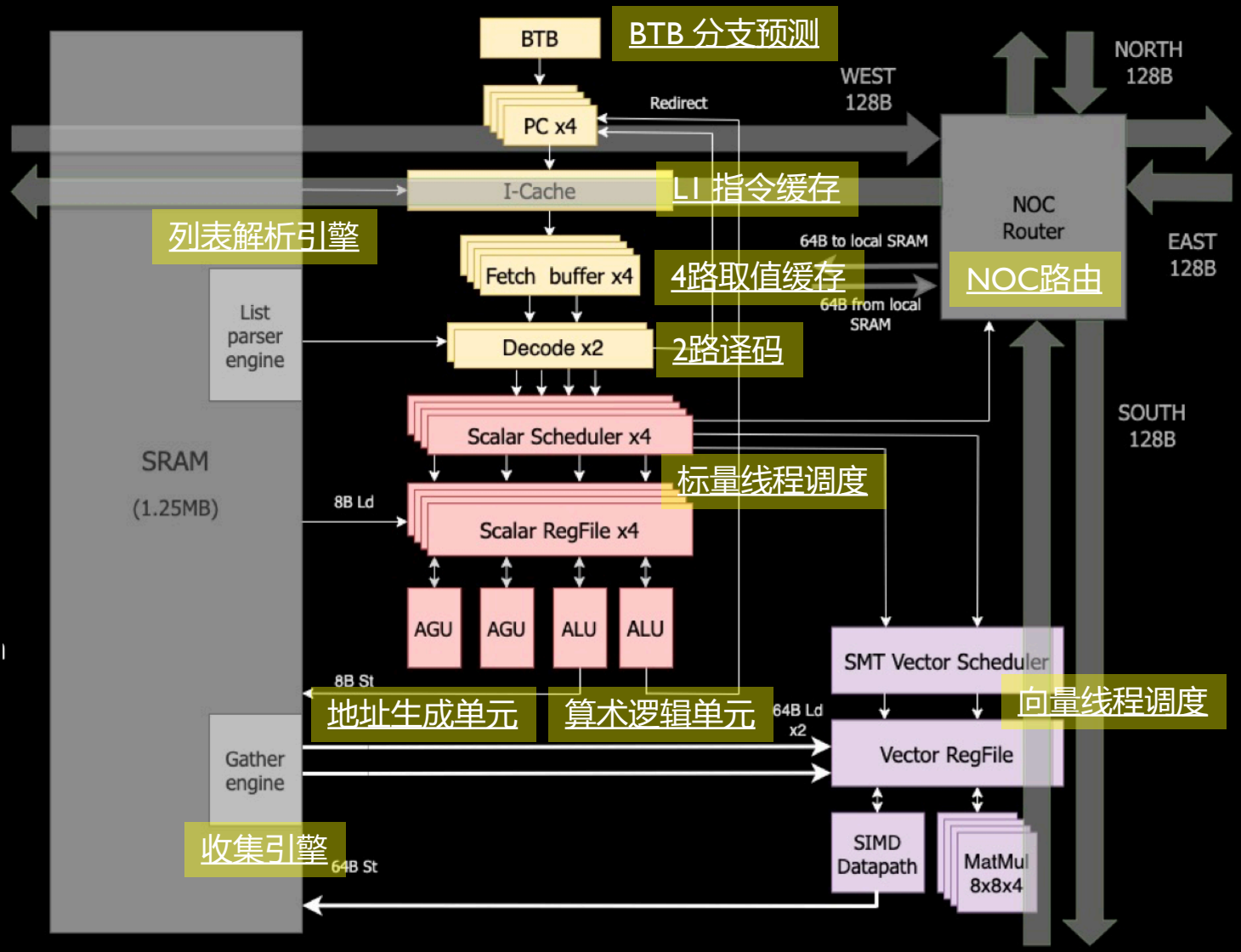
前端模块

- **指令缓存**：L1指令缓存直接与 Core 中 SRAM 相连获取计算指令。
- **取指**：每个 Core 具有 32 B 取指窗口，最多可容纳 8 条指令。
- **译码**：从取指缓冲获取指令并译码，并根据每条指令要求分配必要的执行资源。
- **线程调度**：分为向量和标量的调度器（Scheduler）和寄存器堆（Register File）。



前端模块

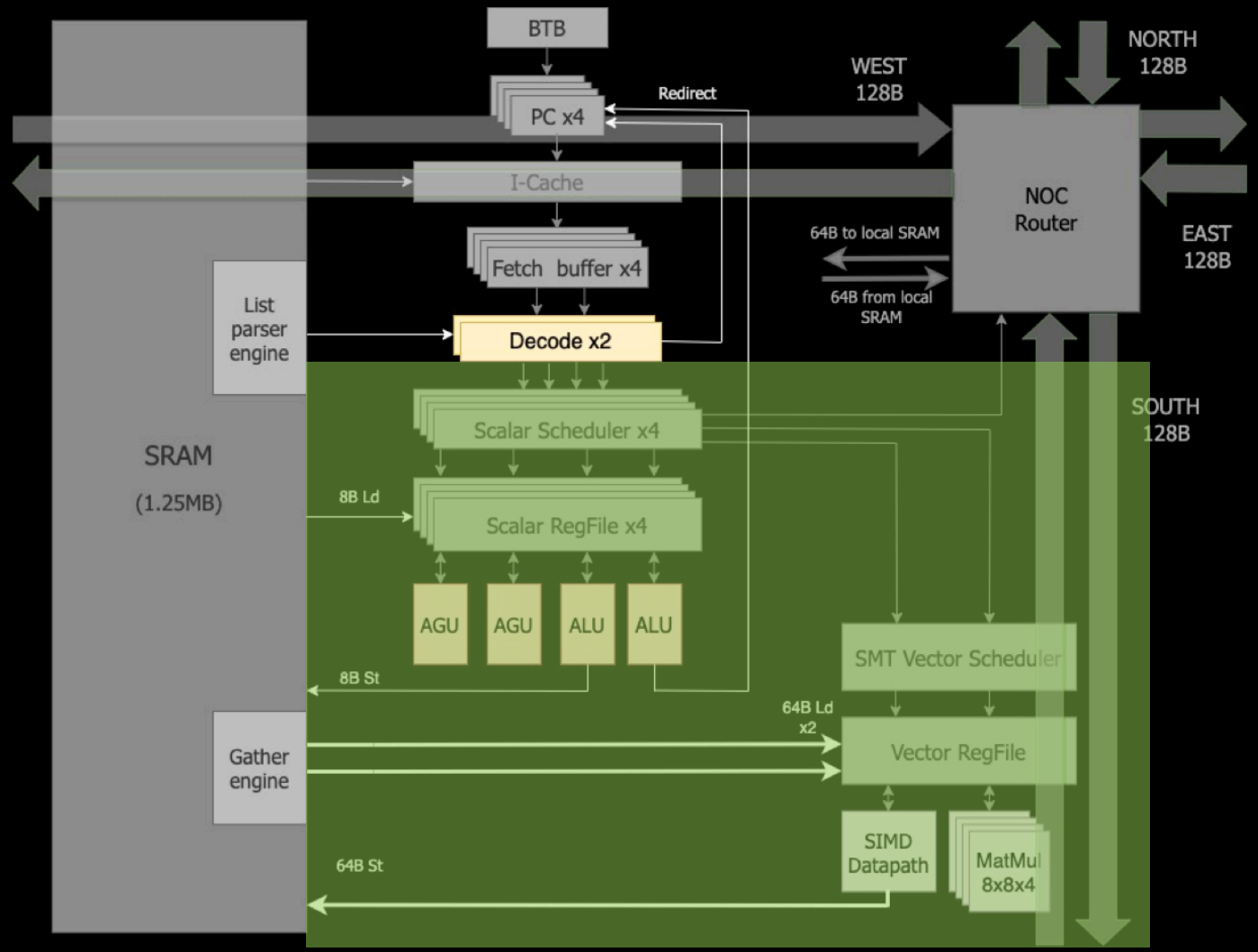
- 一旦分支预测器生成了下一条指令提取指针，Dojo 每个周期从指令缓存中提取 32 个字节到每个线程的提取缓冲区中（per-thread fetch buffers）。
- 指令缓存有助于降低本地 SRAM 的指令带宽压力，确保数据端可以访问 SRAM 时尽可能少地争用。
- 如果将新的代码加载到本地 SRAM 中，则必须在分支到该新代码之前刷新指令缓存。



DOJO 執行引擎

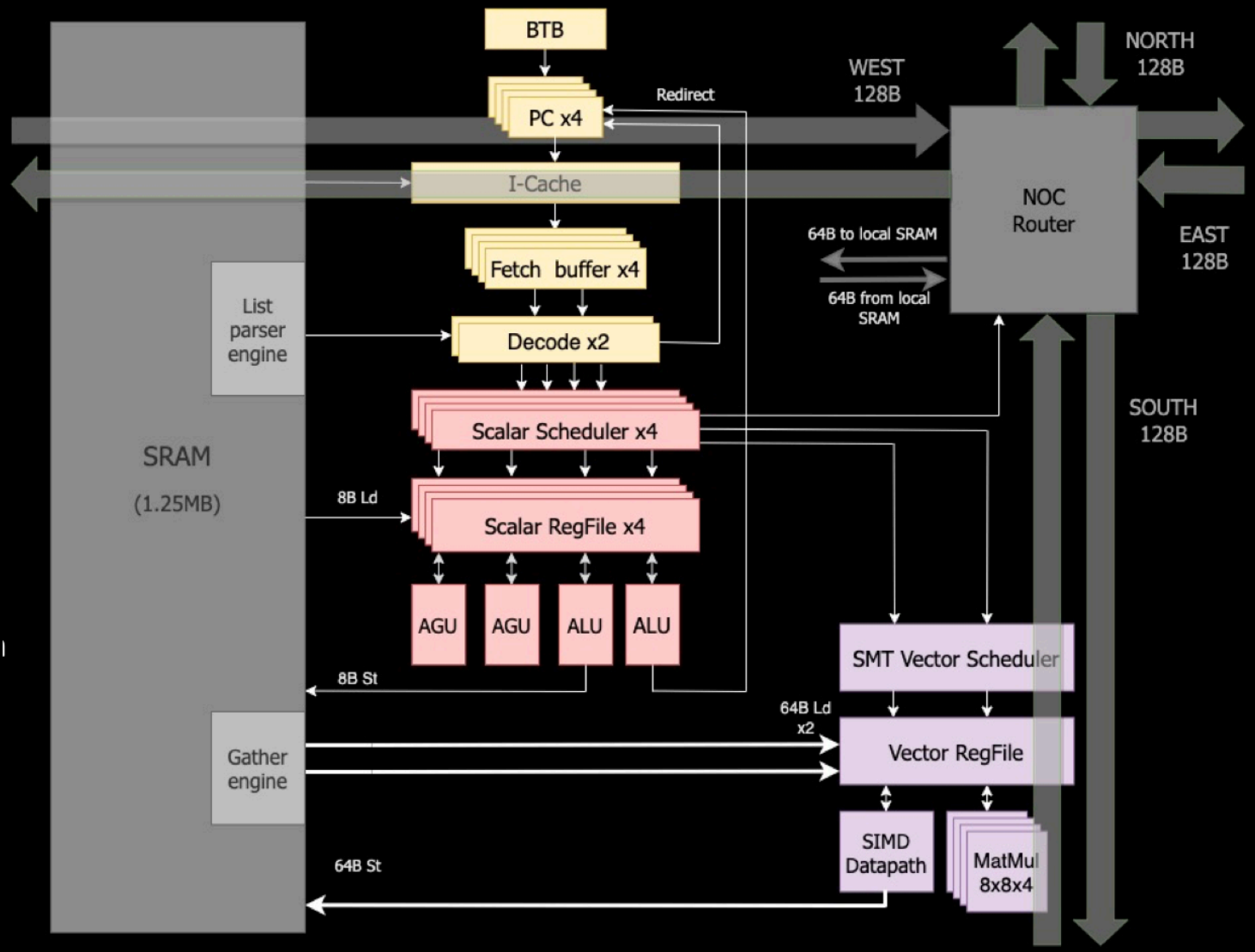
执行引擎

- 4-wide scalar scheduler, 4-way SMT
  - 2 integer ALUs
  - 2 address units
  - Register file replicated per thread
- 2-wide vector scheduler, 4-way SMT
  - 64B wide SIMD unit
  - 8x8x4 matrix multiplication units
- SMT support focuses on single threaded application
  - No virtual memory, limited protection mechanisms, SW-managed sharing of resources
  - Typical application uses 1 or 2 compute threads and 1-2 communication threads



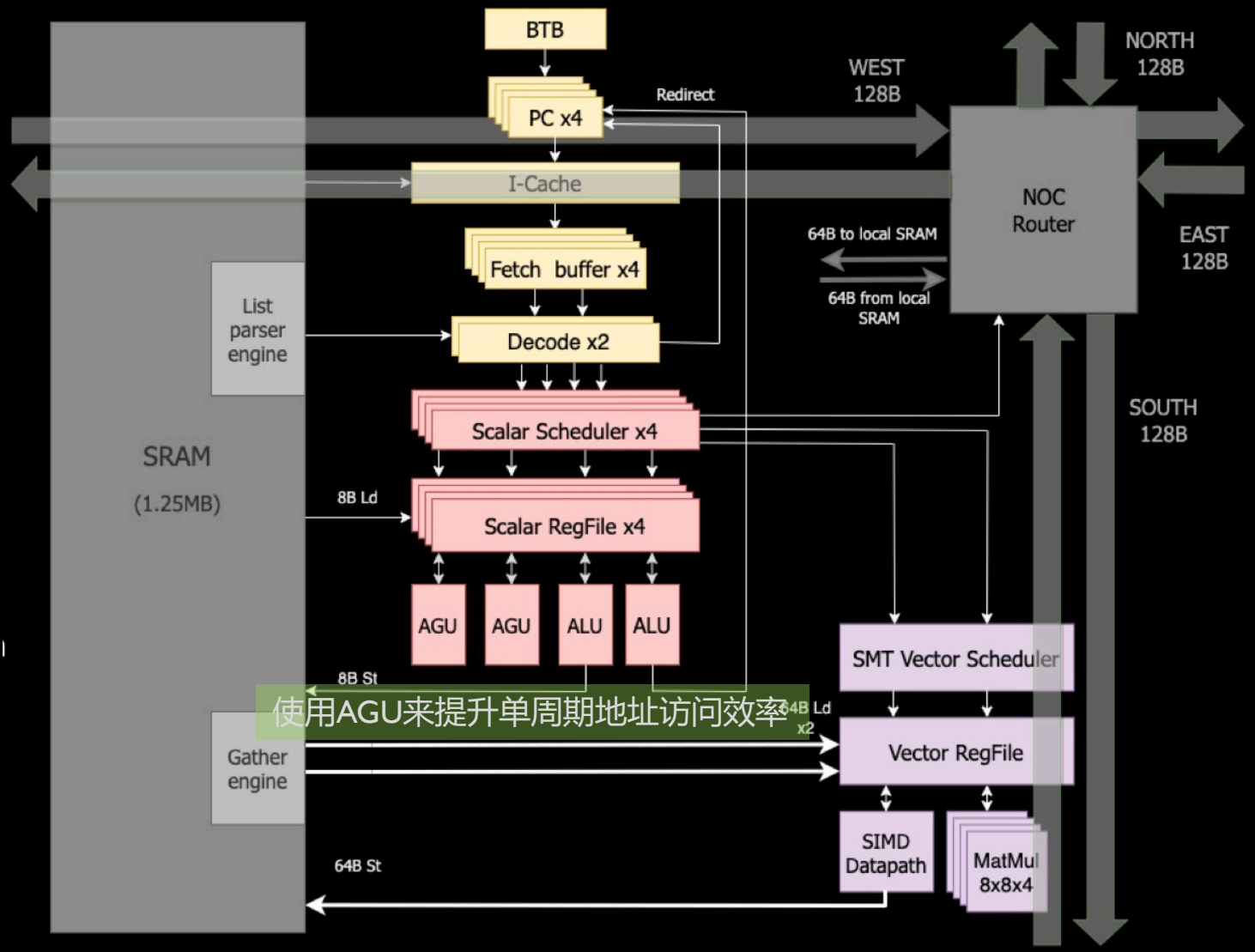
执行引擎

- **执行单元**：2 路 ALU 和 2路 AGU，以及针对向量/矩阵计算的 512 位 SIMD 和矩阵计算单元，其中矩阵计算单元是 D1 芯片的**算力主体**。
- **ALU 和 AGU**：负责矩阵计算外少量逻辑计算。AGU **地址生成单元**，用于生成操作SRAM所需的地址和访问其他核心的地址。通过由与 CPU 其余部分并行运行地址计算。
- **SIMD**：负责激活等特殊功能计算和数据的累加。
- **矩阵计算单元**：DOJO 主要算力，负责二维矩阵计算，实现Conv、Transformer等计算。



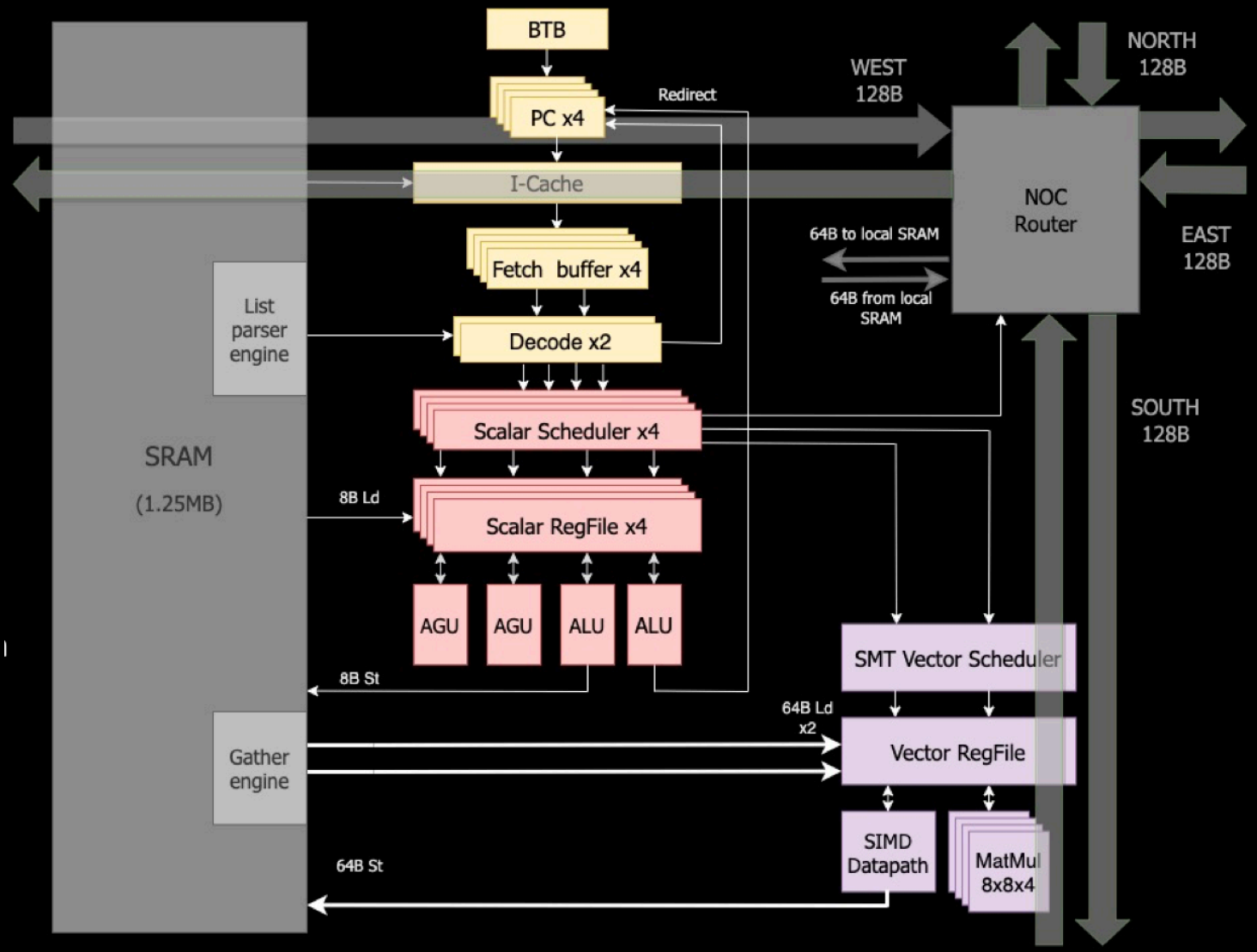
执行引擎

- DOJO 向量和矩阵执行，放置在标量执行引擎之后，并且有两个执行 Pipeline 。一个 Pipeline 可以执行 512 bit 向量计算，另一个 Pipeline 执行 8x8x4 矩阵乘。
- 前端 feeds into 一个四宽标量调度器（four-wide scalar schedule），该调度器具有四路 SMT，2 个整数单元、2 个地址单元和 1 个用于线程的寄存器文件。还有一个带有 4 路 SMT 的向量调度器，数据收集引擎送入一个 64 B 宽 SIMD 单元或四个 8x8x4 矩阵乘法单元。



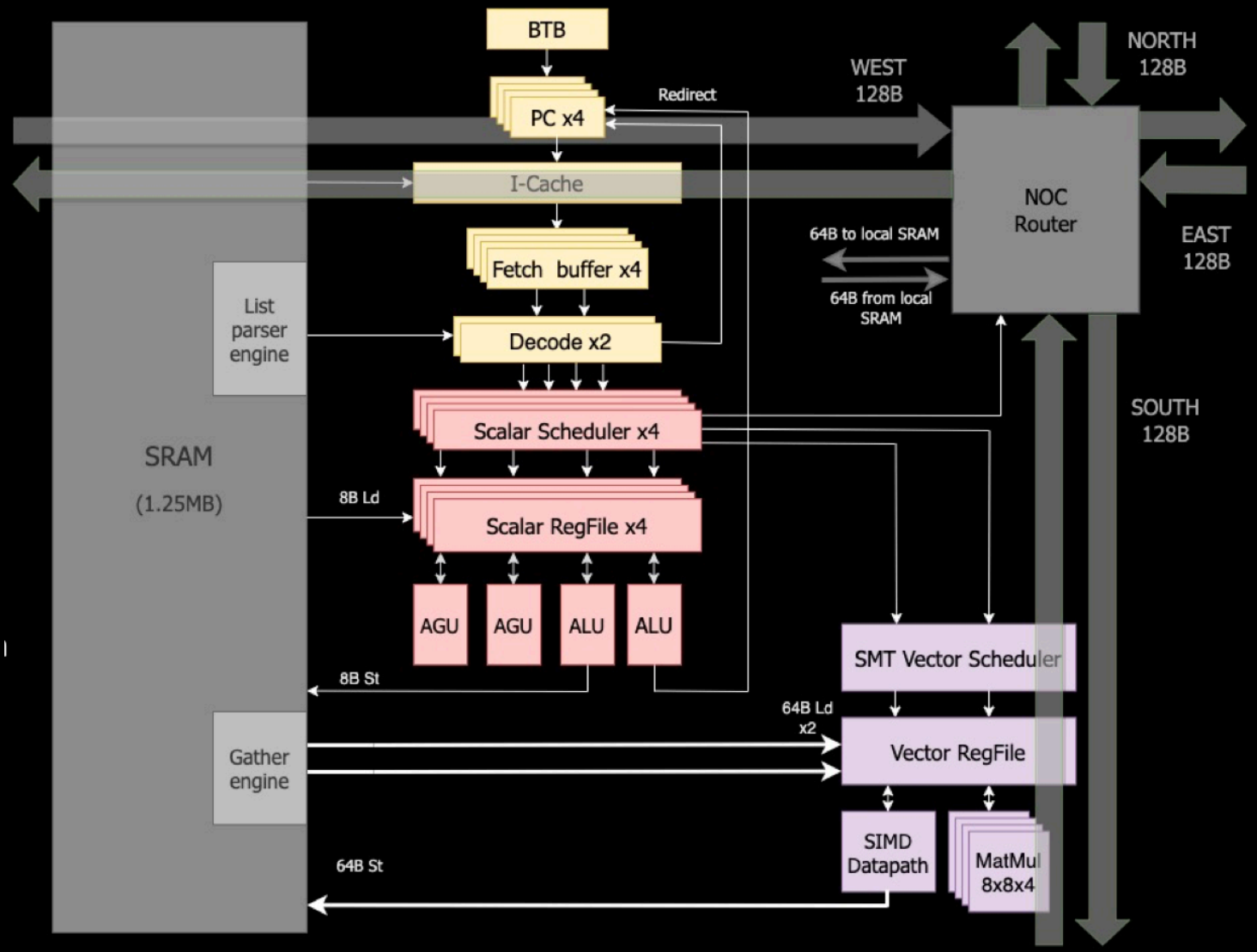
执行引擎

- DOJO Core 架构算力增强的核心是矩阵计算单元。矩阵计算单元与核心 SRAM 的数据交互构成了主要的内核数据搬运功耗。



执行引擎

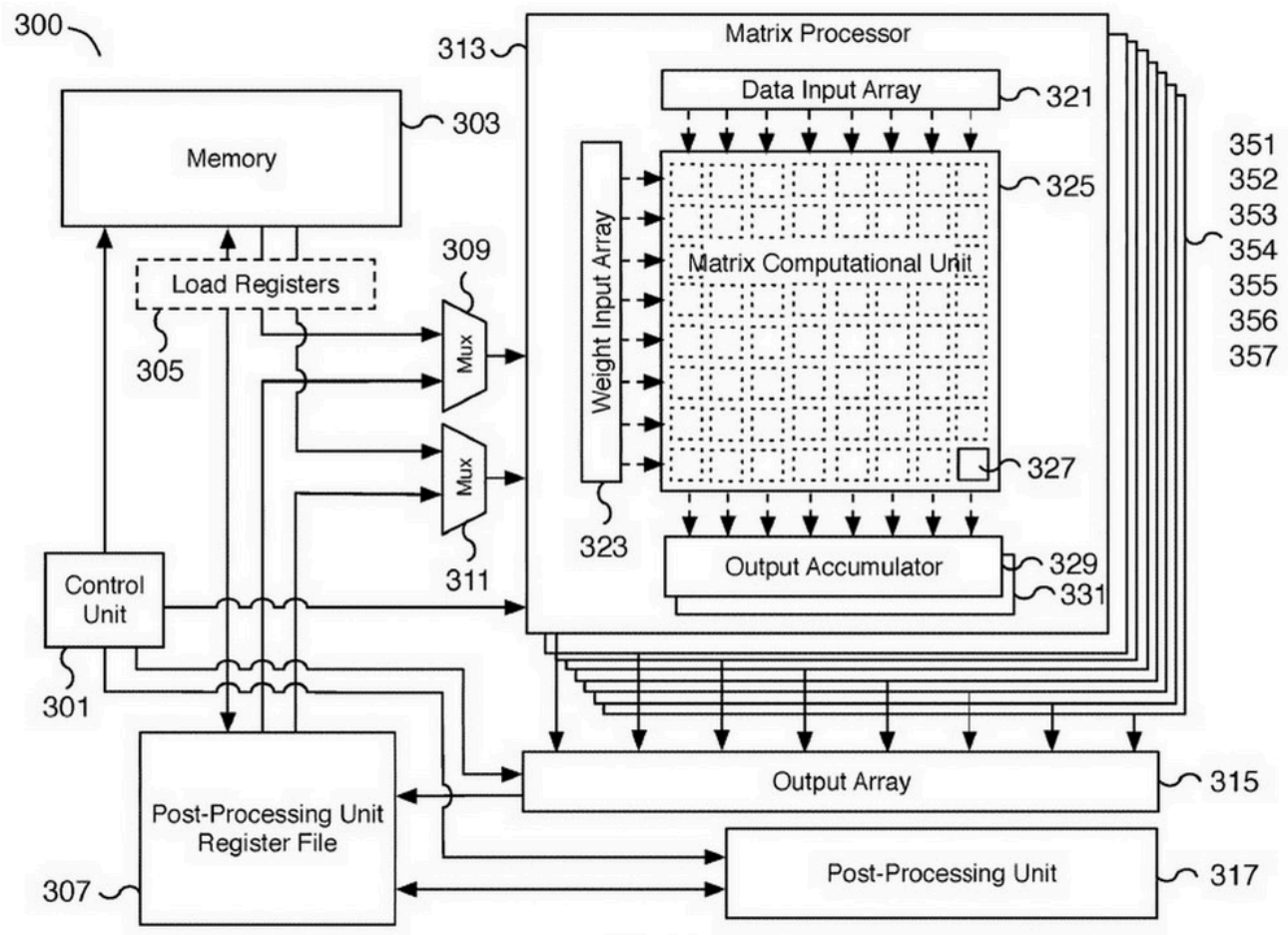
- 内核的 4 路 SMT 功能更多的是让单个应用程序公开显式并行性，而不是提高多任务处理性能。例如，一个线程可以执行向量计算，而另一个线程将数据从系统内存异步加载到 SRAM（通过 DMA）。



矩阵计算单元

https://patents.google.com/patent/US20190235866A1/en
https://patents.google.com/patent/US20200348909A1/en
https://patents.google.com/patent/US20200349216A1/en

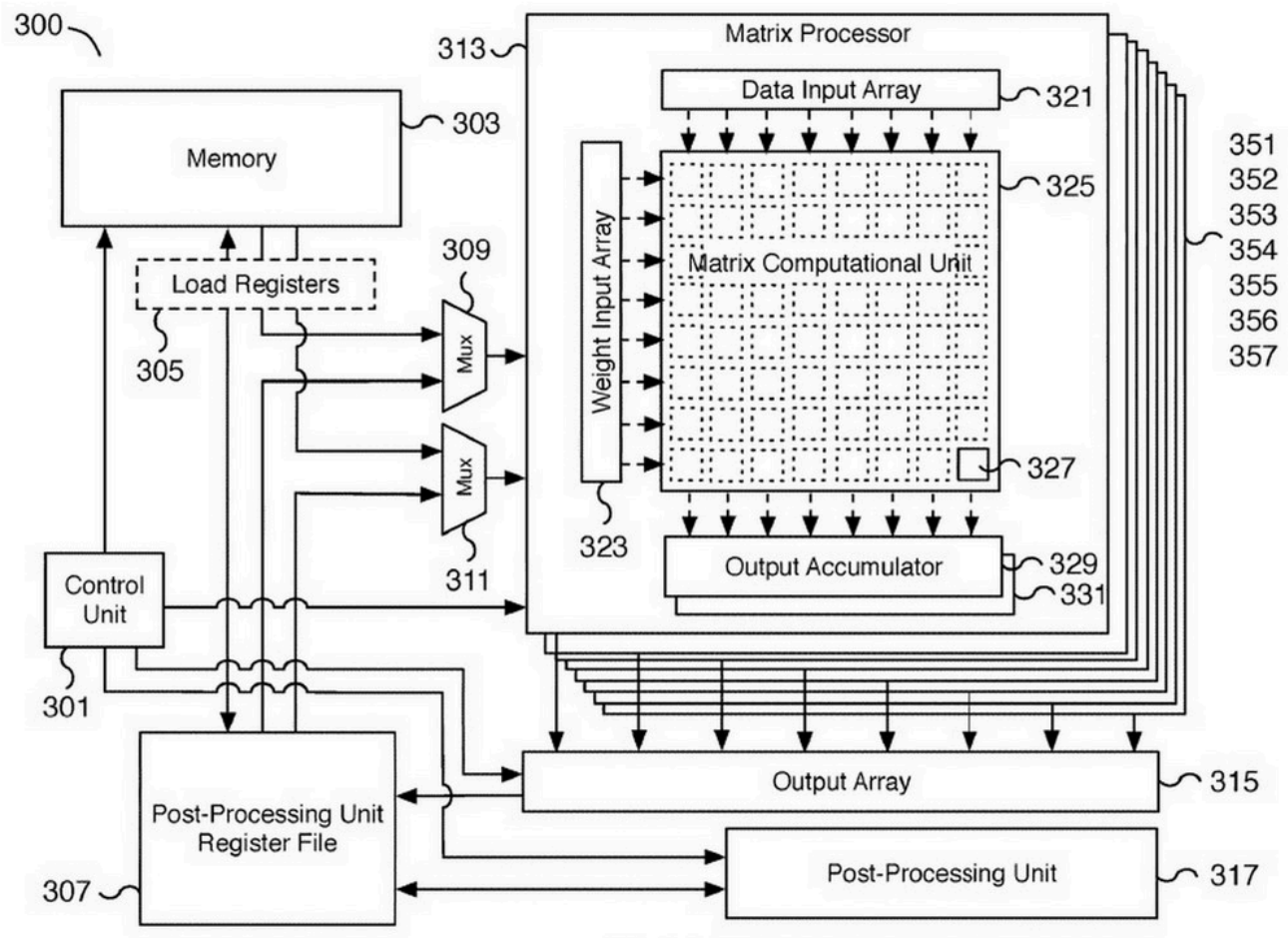
- **8x8 矩阵乘法单元：**输入为数据输入阵列和权重输入阵列，计算矩阵乘法后直接在输出进行累加。每个DOJO Core 包括4路 8x8 矩阵乘法单元。
- **设计结构和数据读写：**精简了RISC-V缓存结构，节约缓存面积并减少延迟。每个DOJO Core 1.25MB SRAM 可以为 SIMD 和矩阵计算单元提供 2x512 bit的读和512 bit 写带宽，以及面向整数寄存器堆的 64 bit 读写能力。计算的主要数据流是从 SRAM 到 SIMD 和矩阵乘法单元。



矩阵计算单元

https://patents.google.com/patent/US20190235866A1/en
https://patents.google.com/patent/US20200348909A1/en
https://patents.google.com/patent/US20200349216A1/en

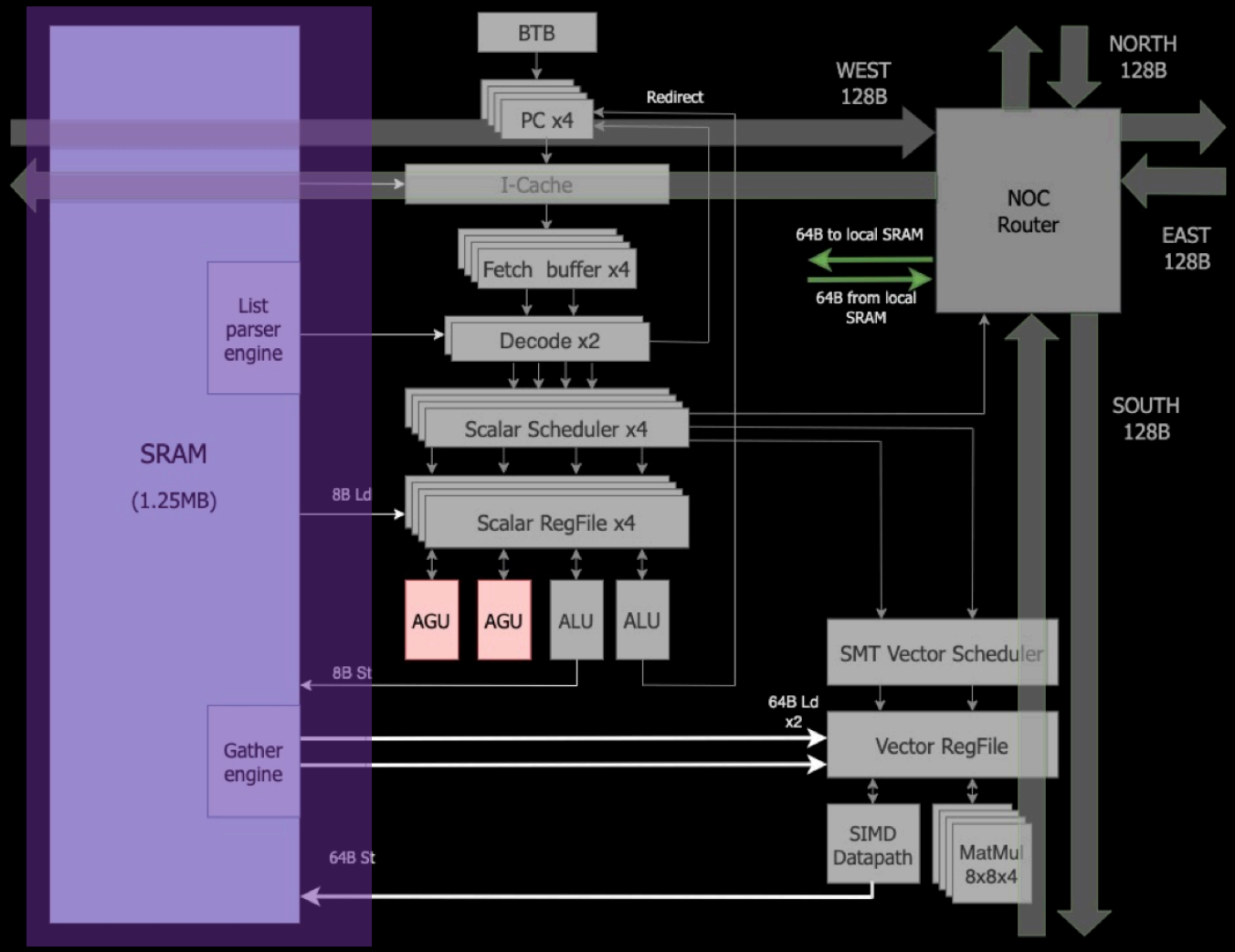
- 通过多路选择器从 SRAM 中加载权重和数据到输入阵列。输入数据与权重在矩阵单元（Matrix computation Unit）中进行乘法计算。
- 结果输出到输出累加器中进行累加，计算时可以通过矩阵划分拼接的方式进行超过8x8的矩阵计算。
- 累加后输出传入后处理器寄存器堆进行缓存，随后进行后处理（激活、池化、Padding等）。
- 整个计算流程由控制单元（Control unit）直接控制。



SRAM 与内存

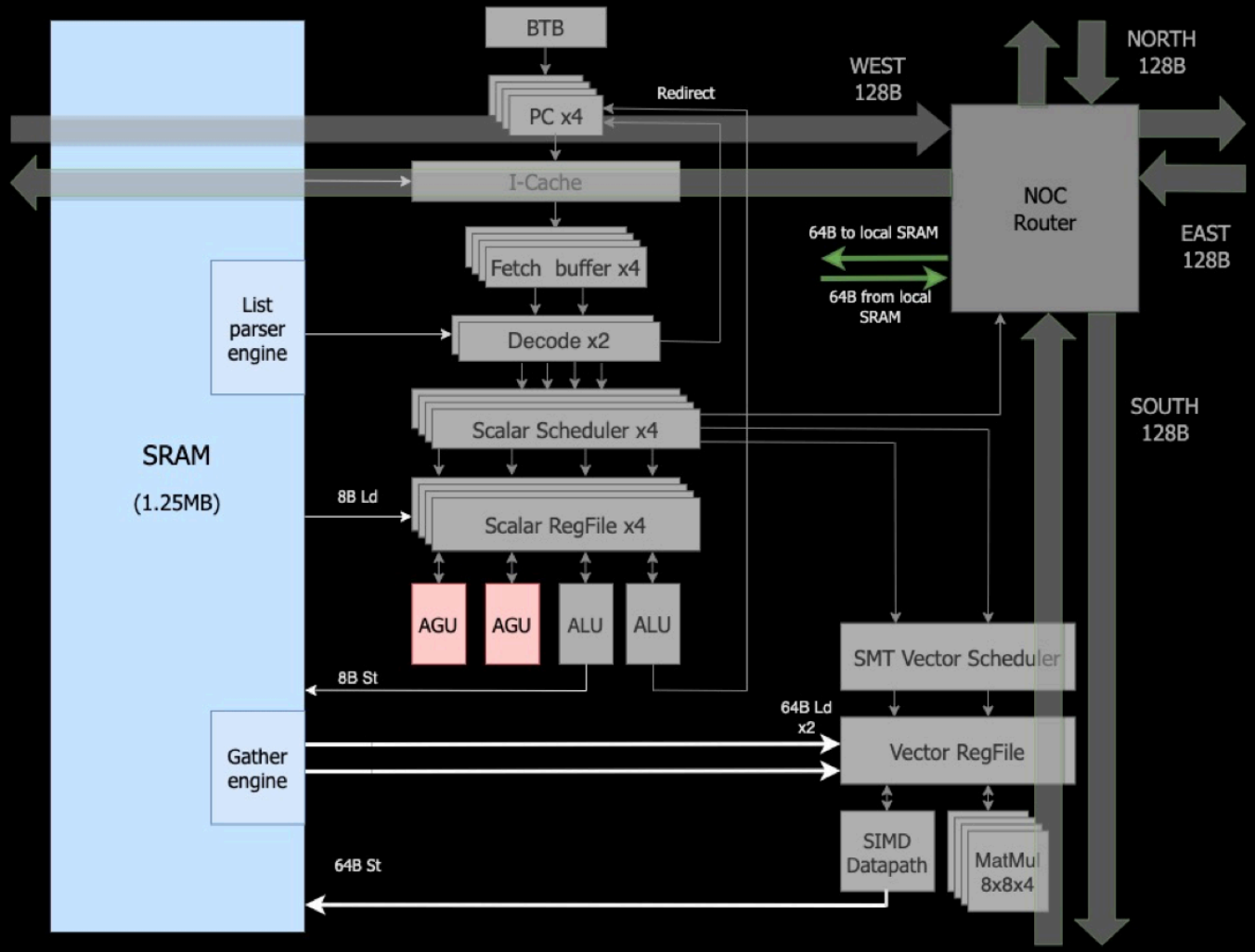
SRAM

- 每个节点 1.25MB SRAM，将内存与计算共置，以最大限度地减少数据传输（Memory Bound）。
- D1 上运行的代码不能直接访问系统内存，应用程序主要在本地 SRAM 中工作；如果需要来自主存（DDR 或 HBM）的数据，须使用 DMA 操作进行读入。
- 通过列表解析引擎可以将复杂的不同数据类型的传输序列进行打包，提升传输效率。



SRAM

- 为了保持低延迟，SRAM 设计为非缓存，跳过一级缓存节省芯片面积和功耗，不需要与数据一起存储的标记和状态位。SRAM 前面也没有 L1D 缓存。
- 为了进一步减少延迟、面积和核心复杂性，DOJO 没有虚拟内存支持。因此，没有 TLB 或页面遍历机制（page walk mechanisms）。
- DOJO 只用 21 个地址位寻址 SRAM，简化 DOJO 的 AGU 和寻址总线，以避免在其前面实现单独的 L1 数据缓存。



NOC Router

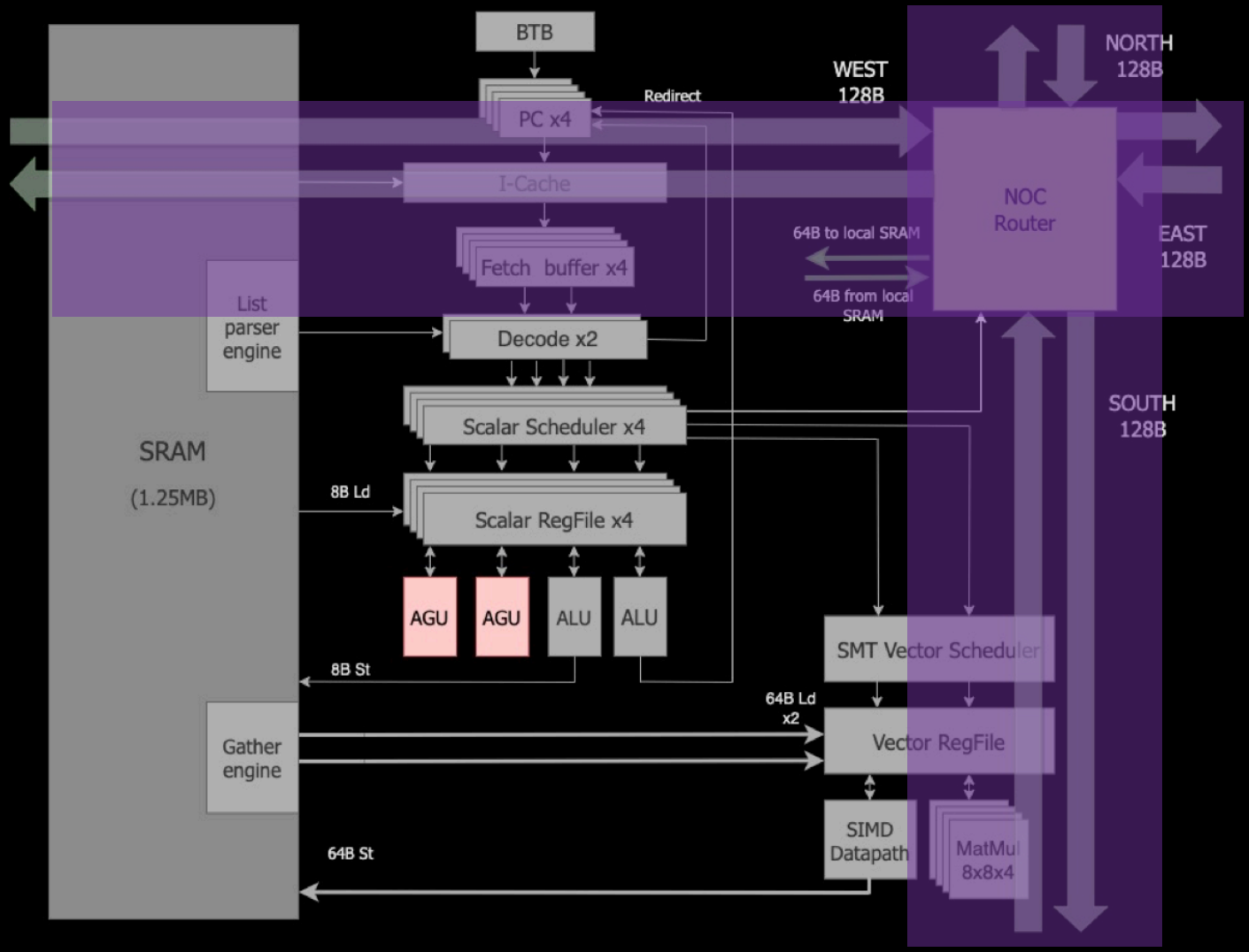
- 每个节点都连接到一个2D网格

2D mesh spanning all processing nodes
- Eight packets per cycle across the node boundary

Each node has independent network connection
- Direct SRAM connection, one read and one write packet per cycle
- Single cycle per hop in every direction

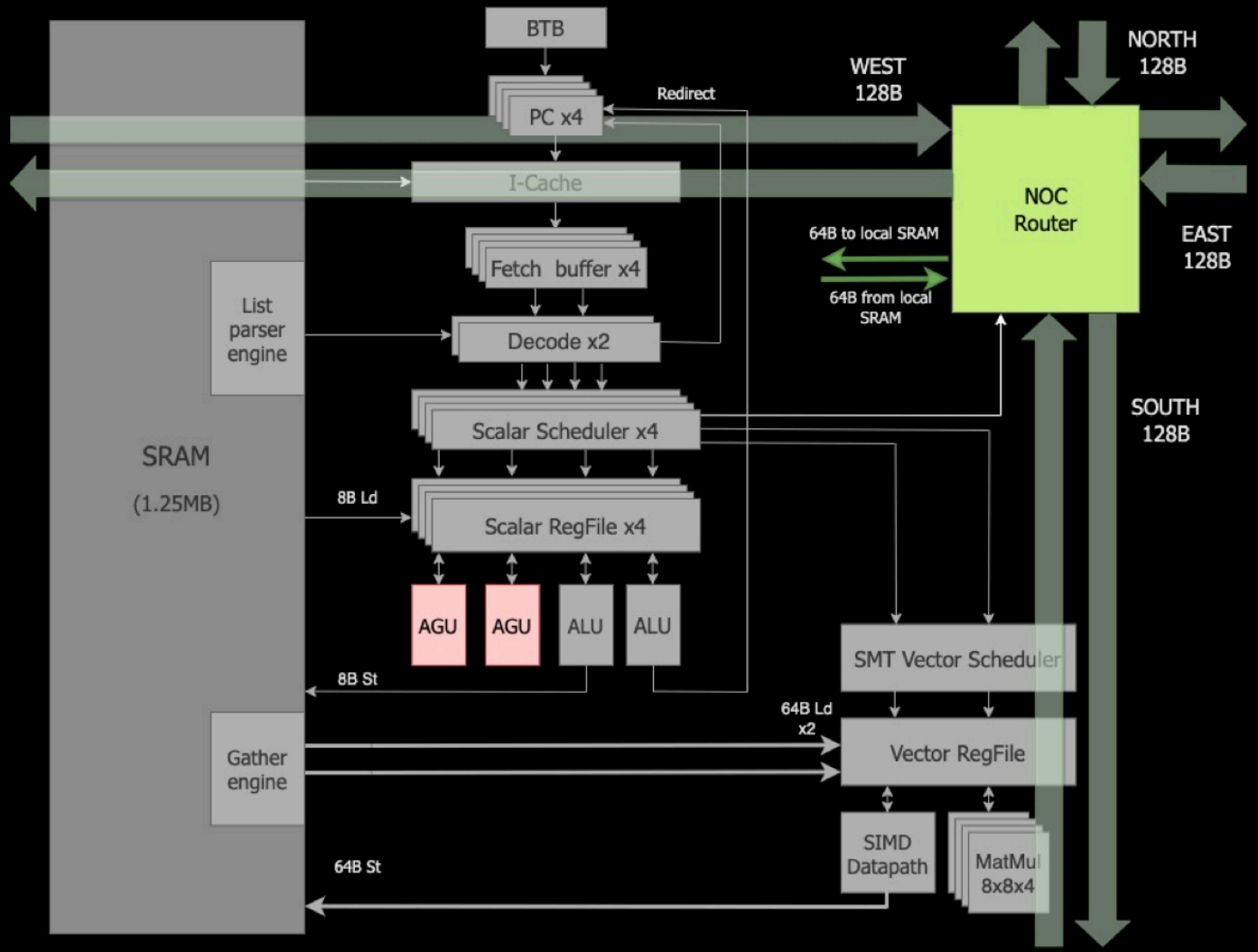
Block level DMA operations for data push and pull

Seamless connection to neighboring nodes



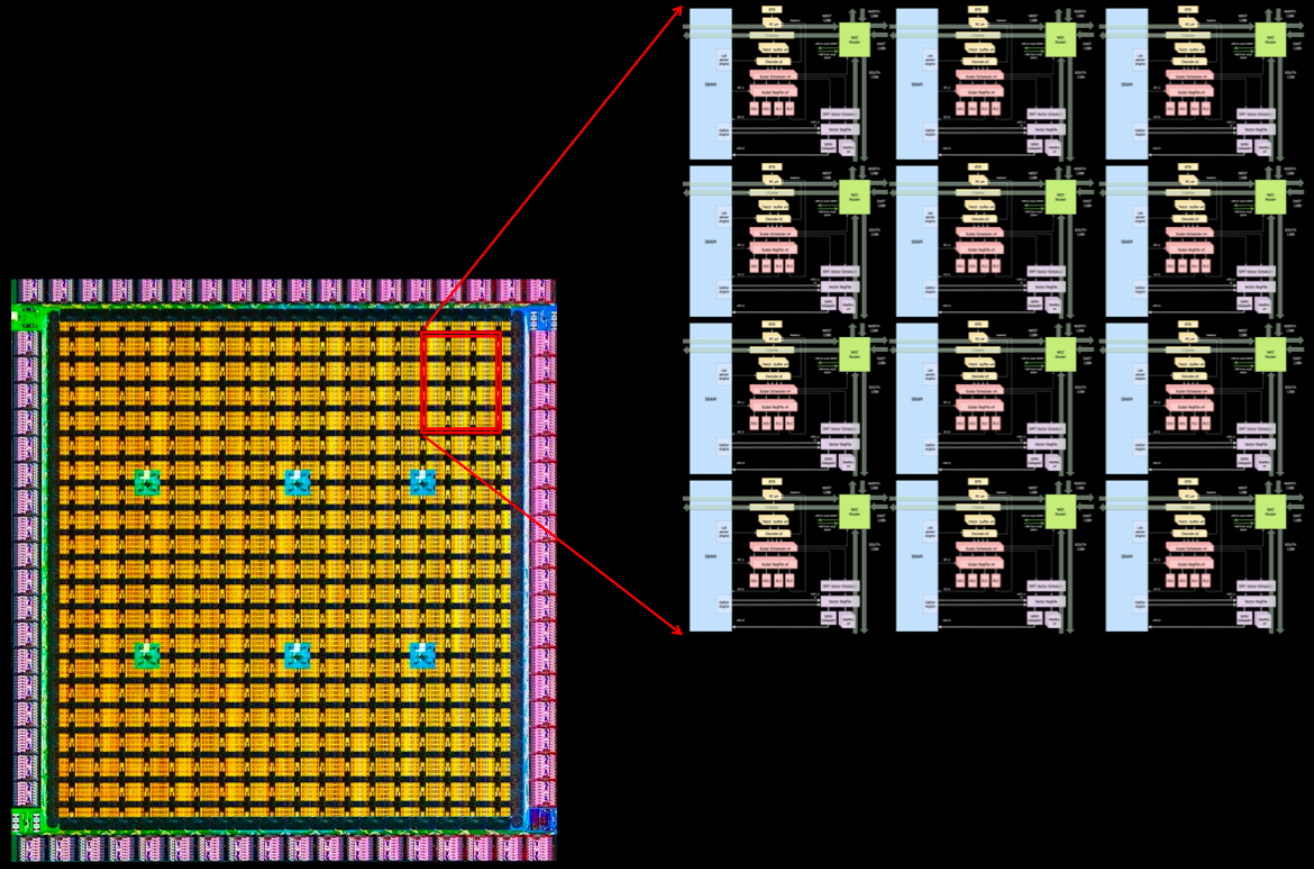
计算单元与 SRAM/NoC 数据交互

- SRAM 具有大读写带宽，以 400 GB/s 速度加载并以 270 GB/s 速度写入。DOJO Core 内部指令集具有专用的网络传输指令，通过 NoC 路由，可以直接将数据移入或移出其他 Core SRAM 存储器。
- 嵌入在该 SRAM 中的是一个列表解析器引擎（ list parser engine ），该引擎馈入解码器对和一个收集引擎（ gather engin e ），馈入向量寄存器文件，可以将信息发送到其他节点或从其他节点获取信息。



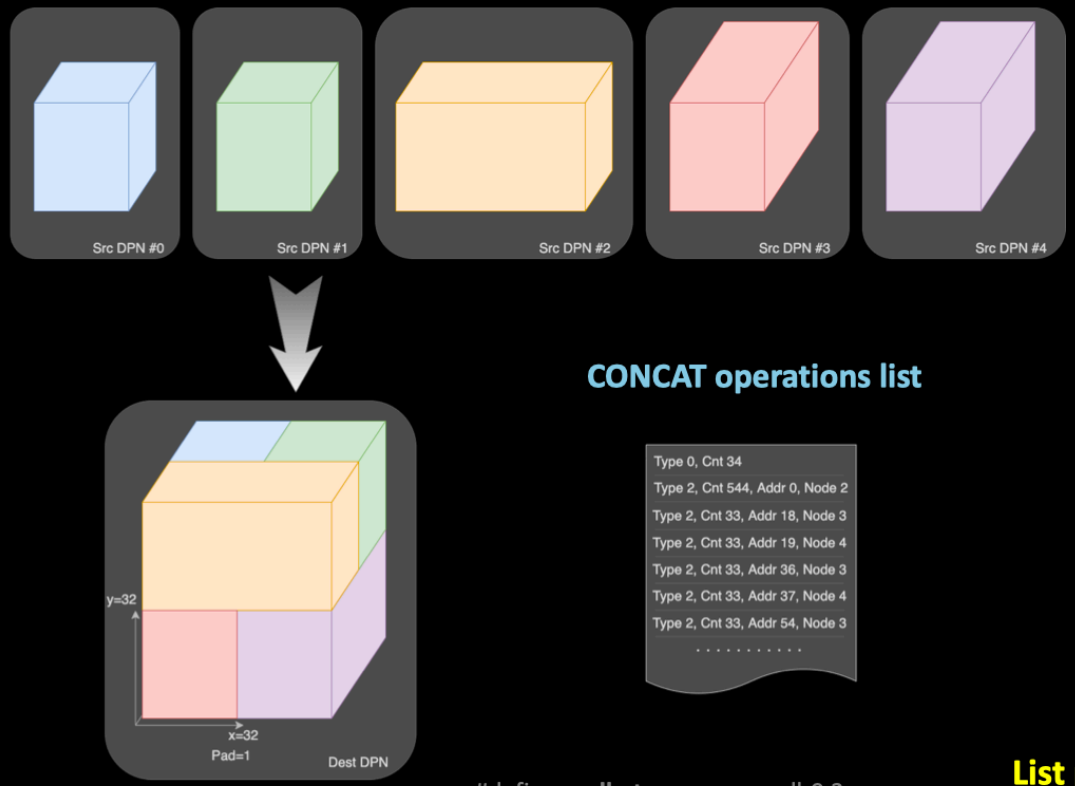
计算单元与 SRAM/NoC 数据交互

- DOJO D1 不会在 DDR 和 PCIe 控制器上花费空间。大部分裸片都被大量 Dojo 核心占据，除了外部设计用于与相邻裸片接口的定制 IO 连接器。
- DOJO 不会尝试跨内核保持缓存一致性，并且不会将任何 SRAM 用于窥探过滤。
-
-



列表解析

- 一种封装不同数据位的方法，以便可以在系统中的 DOJO Core 之间高效传输。



CONCAT operations list


```
#define   db_type        db0:3
#define   db_cnt         db3:12
#define   db_offset      db15:9
#define   db_local_addr  db15:17
#define   db_node_addr   db32:32
#define   db_total_cnt   db3:21
```

```
parse_record DmaDB
movi32 x1, x0, db_local_addr, 6.          ; bring in a src address
movi32 x1, x1, db_node_addr, 24
vxor r31, r31, r31                        ; create a zero register
loop
    loop db_cnt
        db_type==0 : st [x2!+64], r31     ; padding
        db_type!=0  : ldr [x2!+64], [x1!+64], s7   ; pull request
    loop_end
    next_record
    db_type==0 : loop_end
    mov x3, db_offset
    add x1, x1, x3
    db_type==1 : loop_end
    db_type==3 : loop_break all_done      ; exit
    movi32 x1, x0, db_local_addr, 6       ; bring in new src addr
    movi32 x1, x1, db_node_addr, 24
loop_end

all_done:
swait s7, db_total_cnt                    ; wait for data to arrive
```

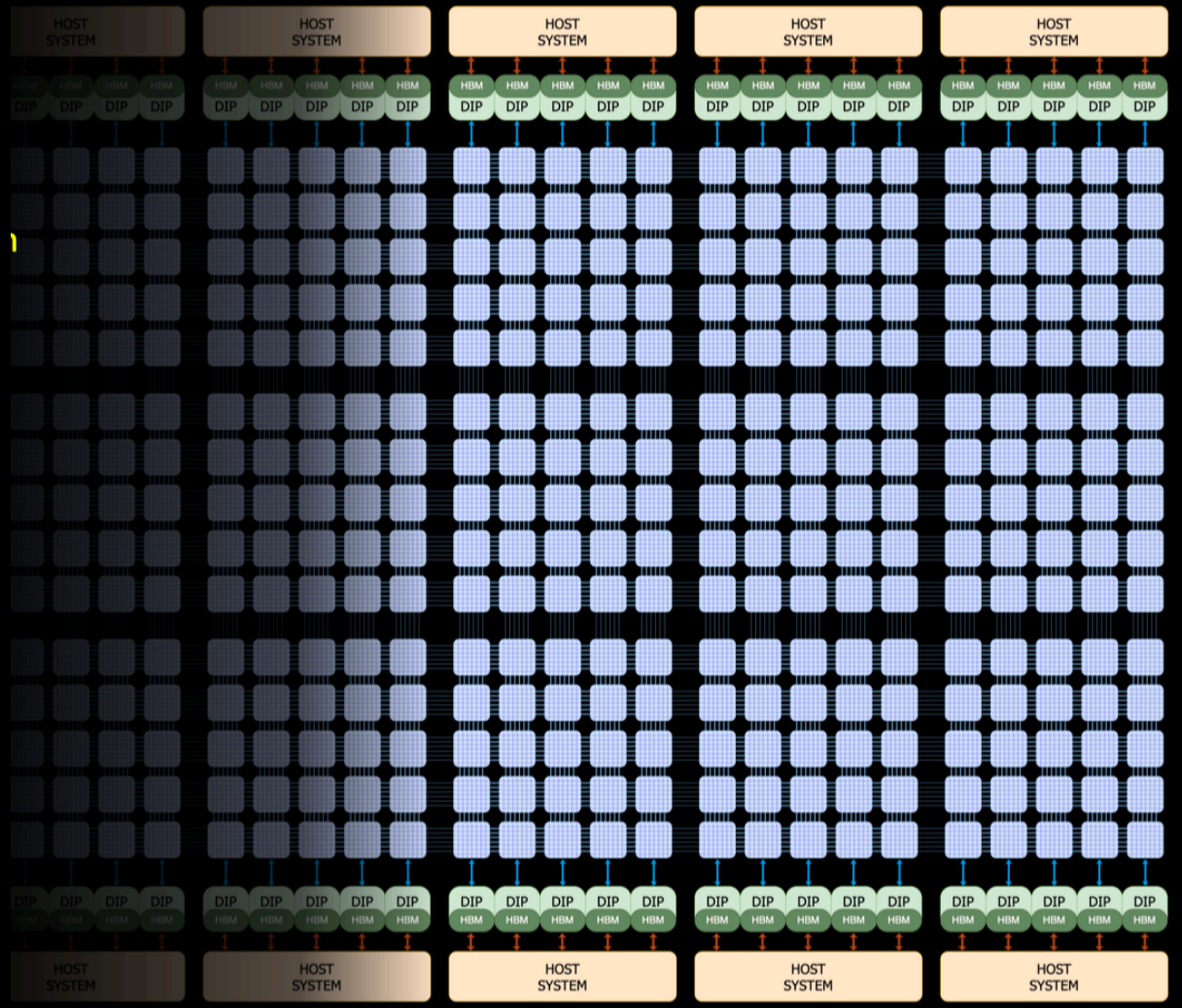
List parser allows efficient packaging of complex transfer sequences

Most instructions execute in the front-end

Sequence can run asynchronously on its own thread

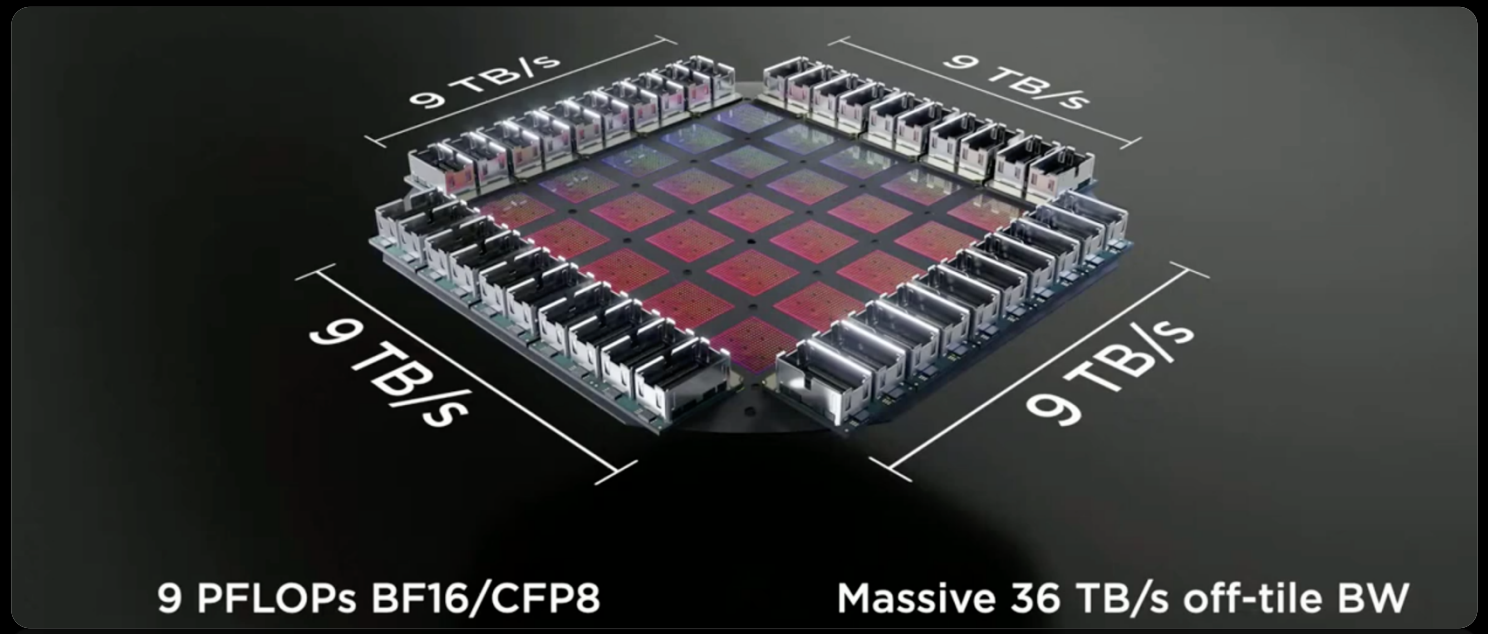
系统内存存取

- 系统内存中，DOJO 芯片并不直接连接到内存。相反，其连接到配备 HBM DIP 上。这些接口处理器还负责与主机系统通信。
- 系统内存可以跨 Tile 边界从每个芯片边缘传输 900 GB/s，这意味着可以以 4.5 TB/s 的链路带宽访问接口处理器及其 HBM。



系统内存存取

- 单个 DOJO 芯片不能自给自足，它没有 DDR 或 PCIe 控制器。因此在 die 边缘周围有 IO 接口，可以让 die 与相邻的 die 进行通信，延迟约为 100 ns。
- 理论上，最小的功能性 DOJO 部署将涉及一个 DOJO D1、一个接口处理器卡和一个主机系统。但特斯拉将 DOJO die 部署在每个包含 25 个 die 模块中；DOJO D1 die 专门设计用于超级计算机的构建块。



指令与数据格式

DOJO 指令集

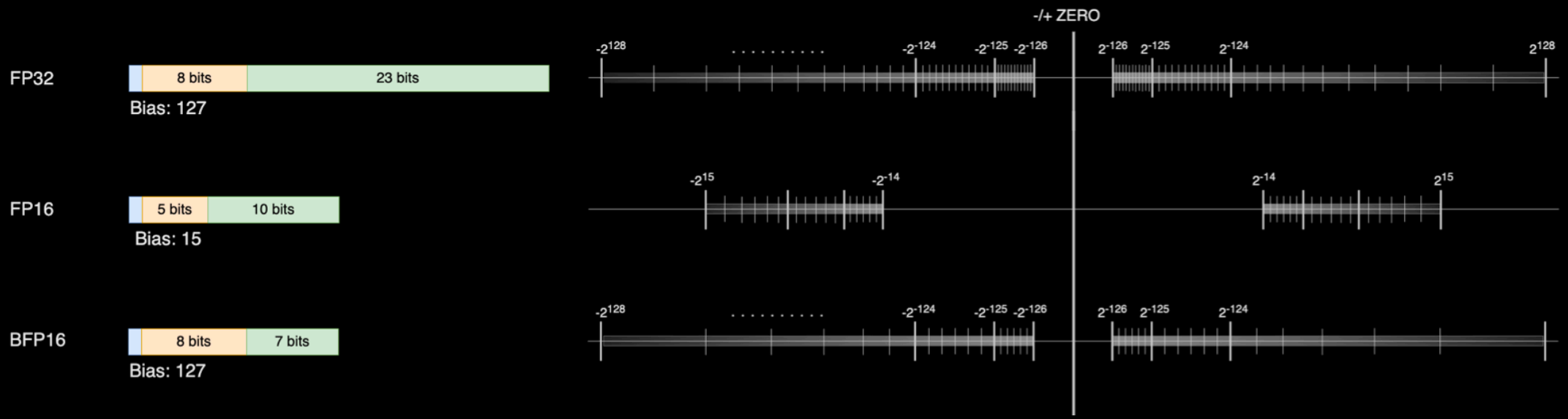
D1指令集支持 64 位标量指令和 64字节 SIMD 指令，包括网络传输与同步原语和机器学习/深度学习相关的专用原语（例如8x8矩阵计算）。并支持信号量（semaphore）和屏障约束（ barrier constraints），这是使内存操作符合指令不仅在 D1 内核中运行，而且在 D1 内核的集合中运行：

1. 至于特定于 ML 的指令，DOJO 定义了包括 shuffle、transpose 、 padding、convert 等数学操作的指令，以及随机舍入（ stochastic rounding ）相关指令；晶体管中，核心还进行随机舍入（ stochastic rounding ），可以进行隐式 2D 填充（implicit 2D padding），即通常通过在一条数据的两侧添加零来调整张量来完成。
2. 在网络数据传输和同步原语方面，支持从本地存储（SRAM）到远程存储传输数据的指令原语（Primitives），以及信号量（Semaphore）和屏障约束（ Barrier constraints）。这可以使D1支持多线程，其存储操作指令可以在多个 D1 内核中运行。

Inst Type	Unique opcodes	Variants
Front-end	12	21
Scalar	74	243
Vector	142	1095

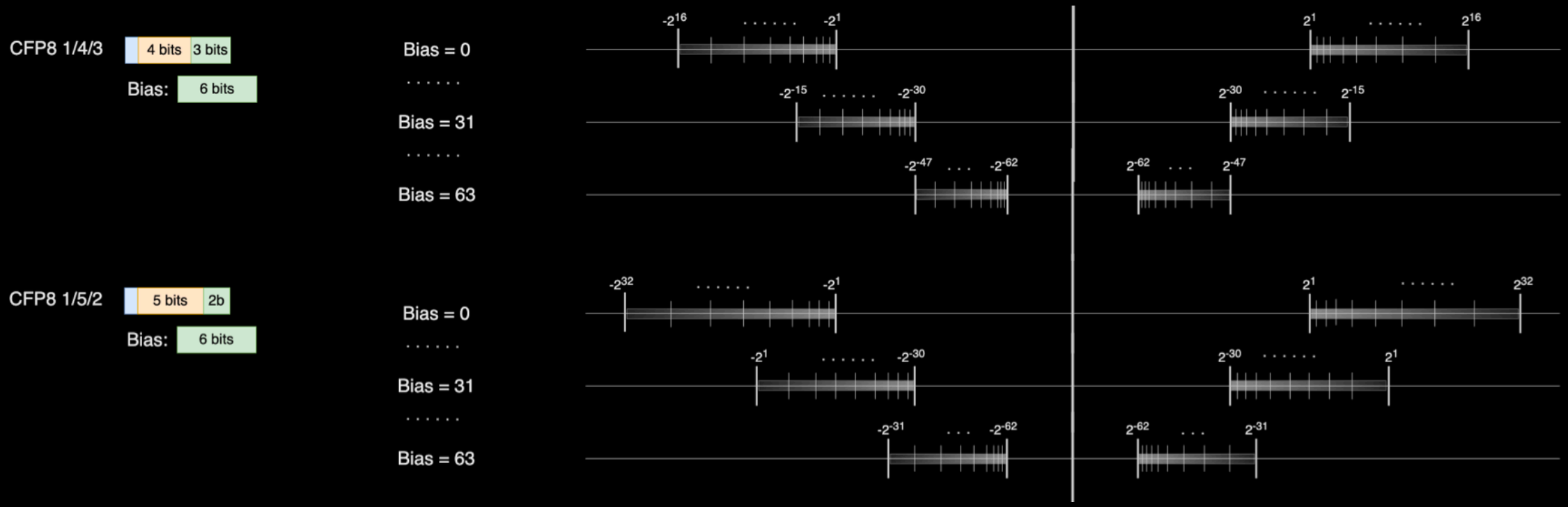
数据格式

- D1 内核支持多种数据格式。标量单元支持 8、16、32 或 64 位的整数，而向量单元及其关联的矩阵单元支持多种数据格式，具有精度和数值范围的混合，其中不少是动态的可由 Dojo 编译器组合。



数据格式

- Tesla 不仅提出了用于较低精度和更高吞吐量矢量处理的 8 位 FP8 格式，而且还提出了一组可配置的 8 位和 16 位格式，DOJO 编译器可以在尾数的精度附近滑动指数位，以涵盖更广泛的范围和精度。在任何给定时间，最多可以使用 16 种不同的矢量格式，但每个 64 B 数据包必须属于同一类型。



Reference 引用&参考

1. https://www.youtube.com/watch?v=uE2f7kiRhmw
2. https://www.youtube.com/watch?v=QurtwJdb5Ew
3. https://www.youtube.com/watch?v=DSw3IwsgNnc
4. https://chipsandcheese.com/2022/09/01/hot-chips-34-teslas-dojo-microarchitecture/
5. https://en.wikipedia.org/wiki/Tesla_Dojo
6. https://www.qbitai.com/2022/08/37209.html
7. https://zhidx.com/p/347884.html
8. https://www.cnblogs.com/wujianming-110117/p/17115152.html
9. https://mp.weixin.qq.com/s/xGytSXTW7-CL-O1V7y7QRw
10. https://mp.weixin.qq.com/s/uBL4x4MJz1GiCf2a0gnnUQ
11. https://mp.weixin.qq.com/s/N_sMmndpyiq0_qbdwq3MuA
12. https://www.51cto.com/article/717372.html
13. https://mp.weixin.qq.com/s/vmNsRVwmi3Azo-bPDanc2g

BUILDING A BETTER CONNECTED WORLD

THANK YOU