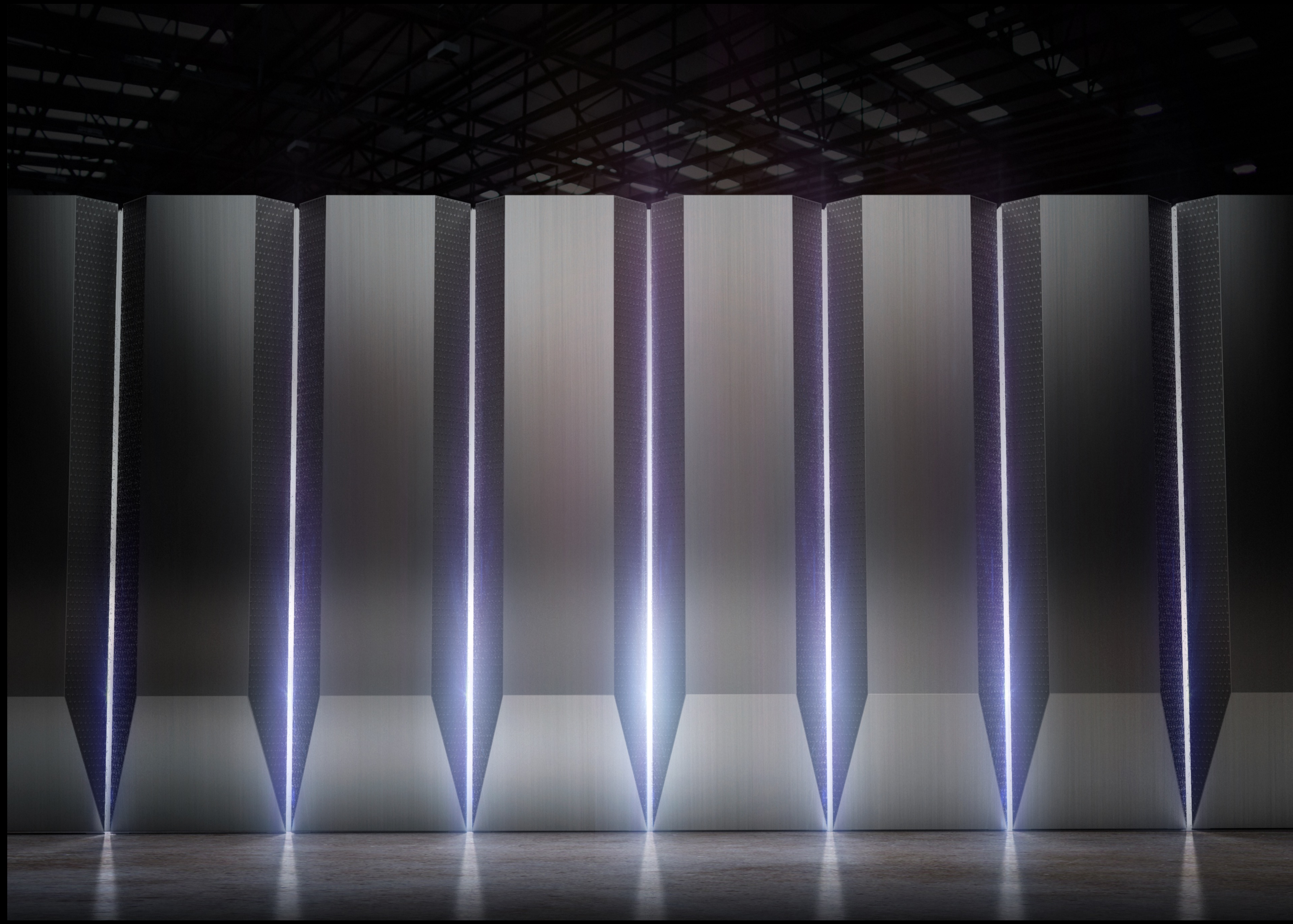
DOJO

The Microarchitecture of Tesla’s Exa-Scale Computer

Emil Talpes, Douglas Williams, Debjit Das Sarma

What is DOJO?



Tesla's in-house supercomputer for Machine Learning

Highly scalable and fully flexible distributed system

- Optimized for Neural Network training workloads
- General-purpose system capable of adapting to new algorithms and applications

Built from grounds up with large systems in mind

- Not evolved from existing small systems

Anatomy of a distributed system

Distributed systems are built as hierarchies of nesting boxes

- CPU -> Die -> Module -> Board -> Rack -> Cabinet -> System
- Integration gets looser as we move outward – lower bandwidth, higher latencies

System is described by three models

- Compute – architecture of the inner box
- Communication – how data moves between boxes
- Synchronization – how events get ordered across the entire system

This talk describes our way of filling these boxes

Microarchitecture of the DOJO node

High throughput, general purpose CPU

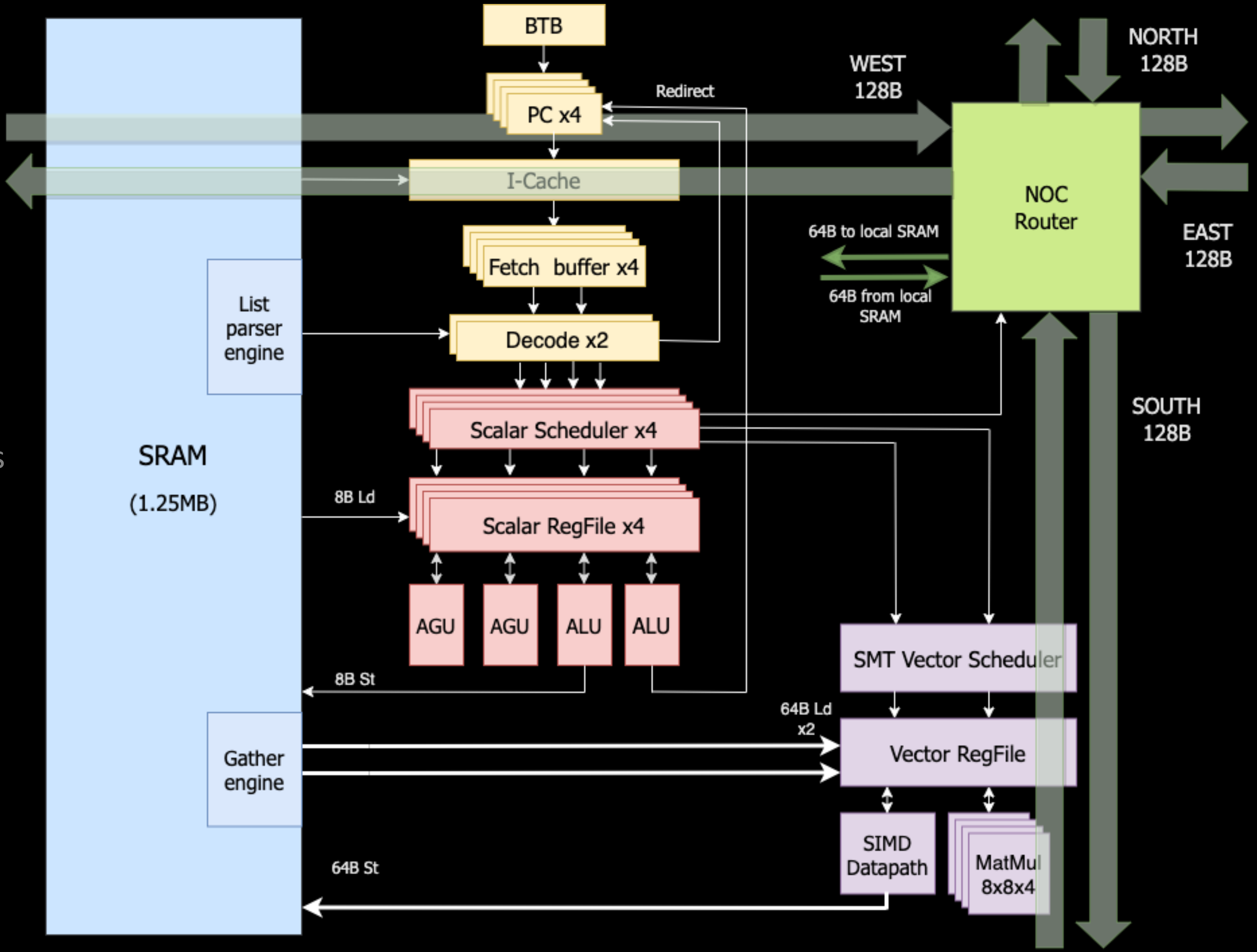
DOJO nodes are full-fledged computers

- Dedicated CPU, local memory, communication interface

Superscalar, multi-threaded organization

- Optimized for high-throughput math applications rather than control heavy code

Custom ISA optimized for ML kernels



Processing pipeline

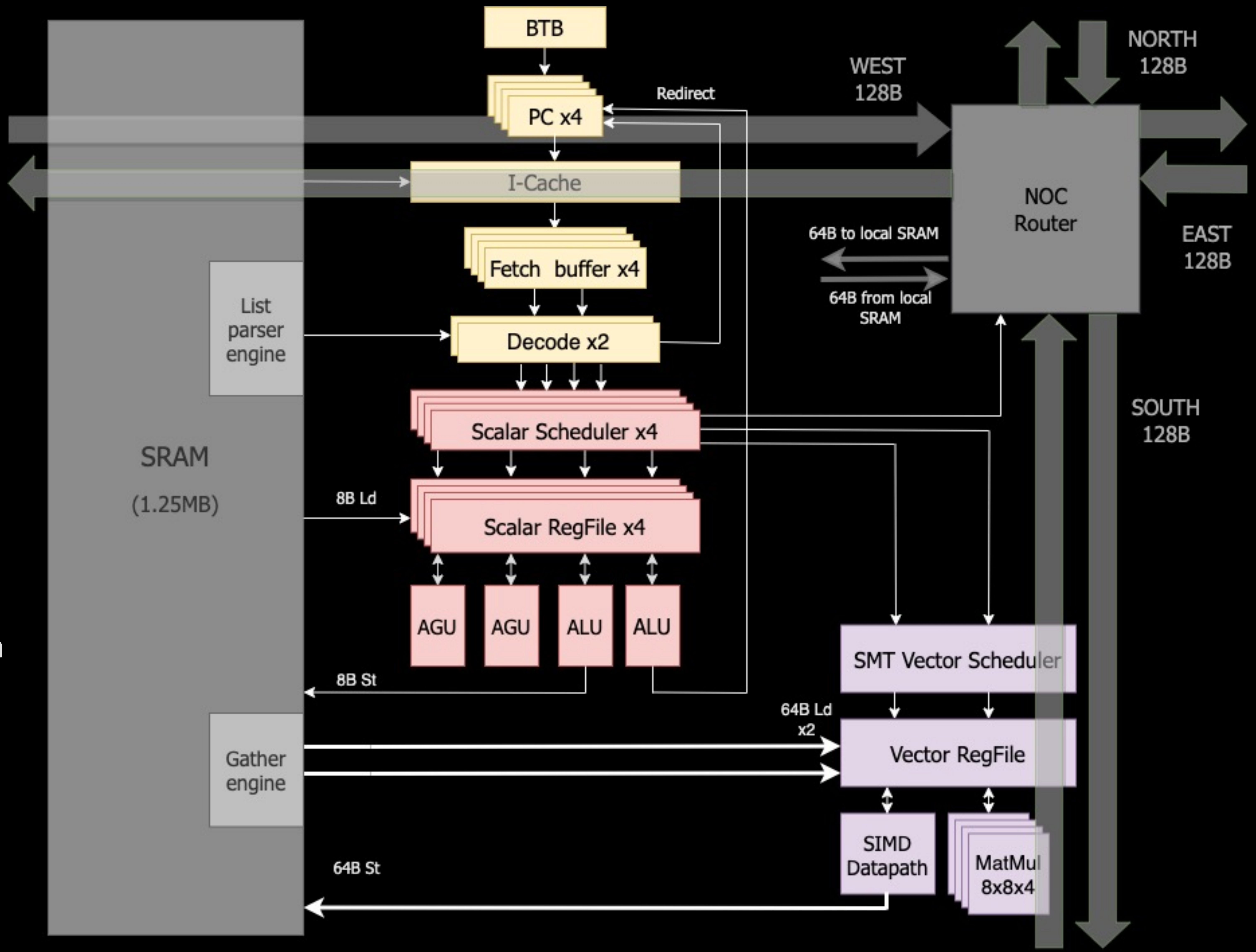
32B fetch window holding up to 8 instructions

8-wide decode handling 2 threads per cycle

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



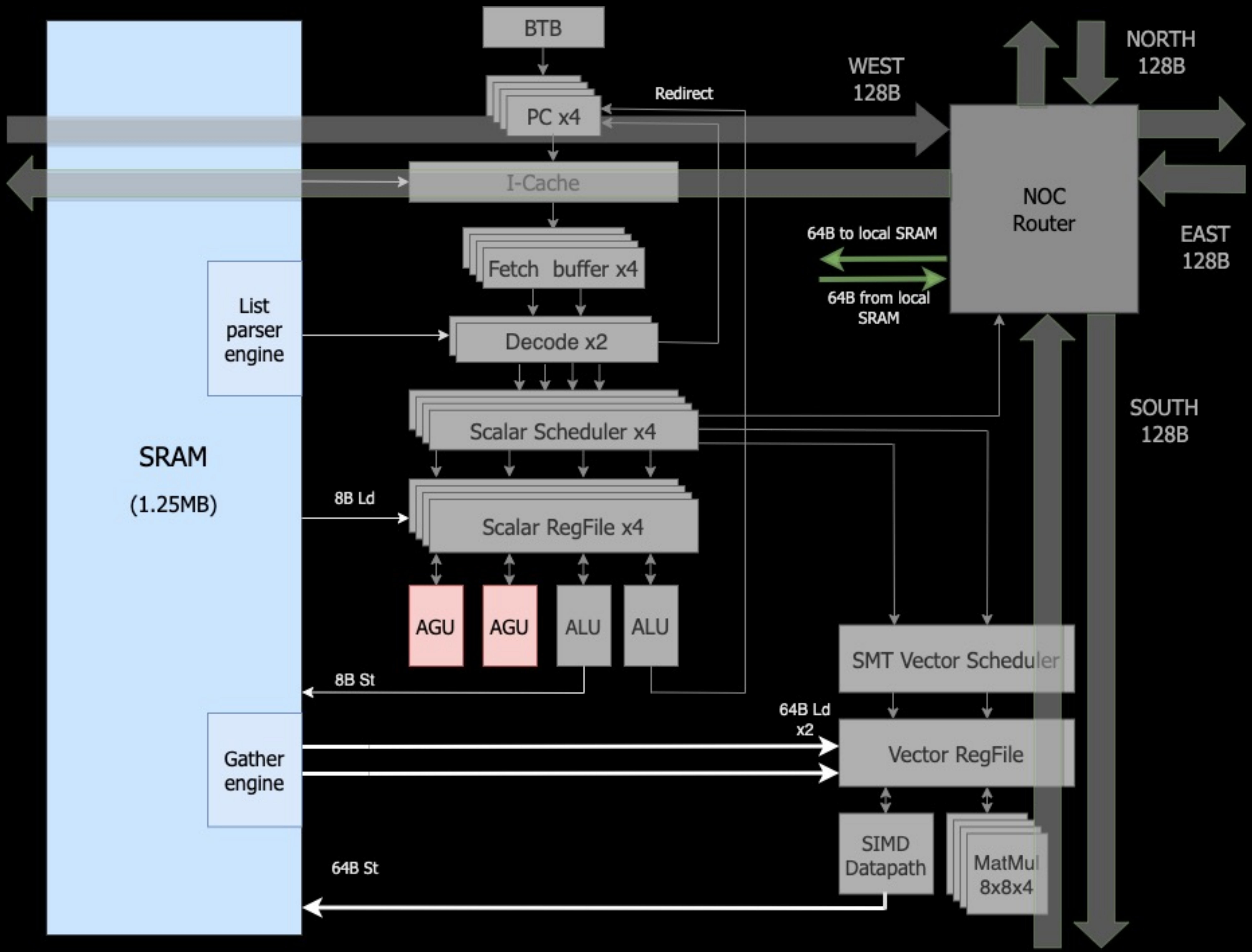
Node memory

1.25MB SRAM per node
- 400 GBps load, 270 GBps store

Gather engine
- 8B and 16B granularity

Load, store, load+execute from local memory
- Explicit transfer instructions for remote memory access

List parsing



Network interface

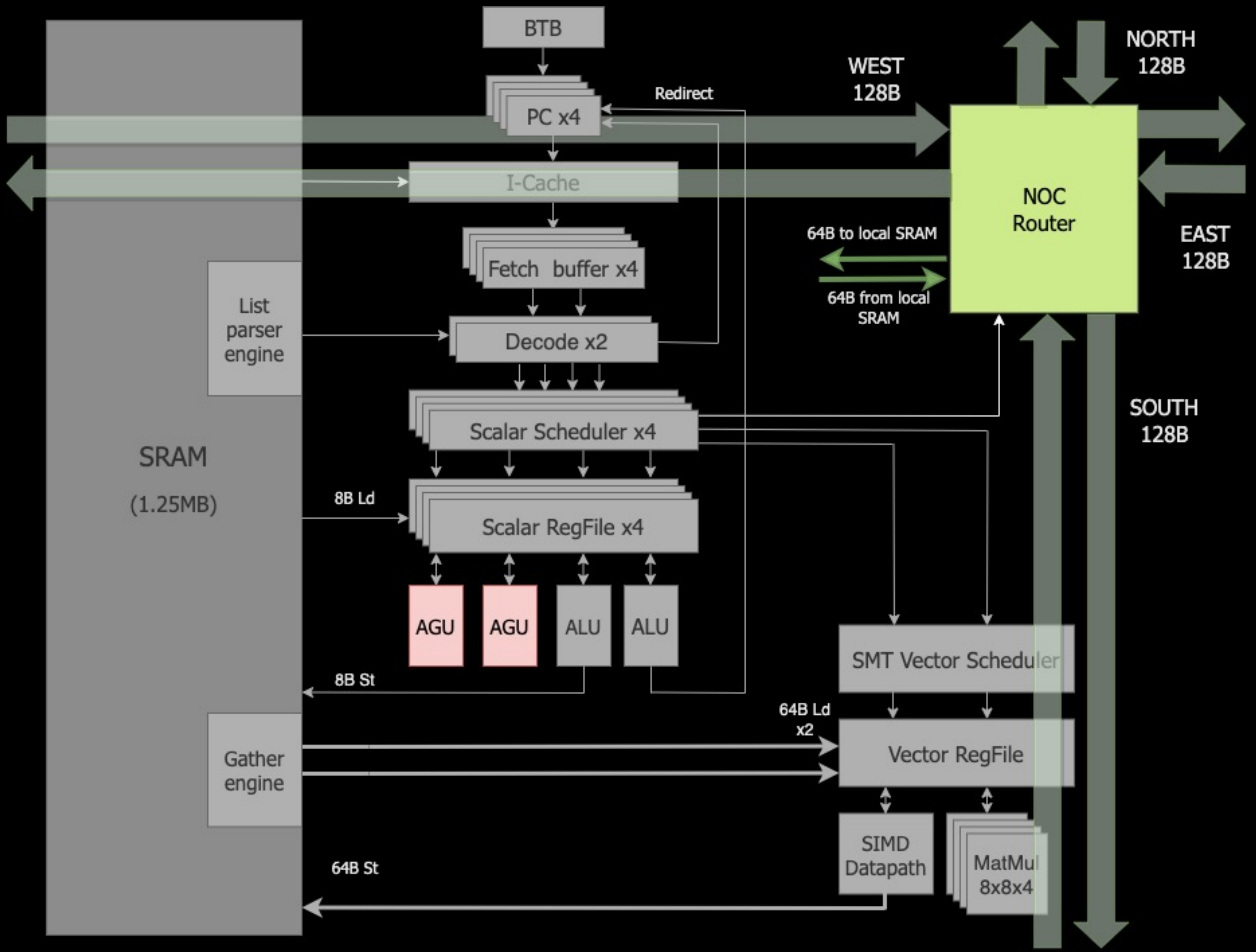
TESLA

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



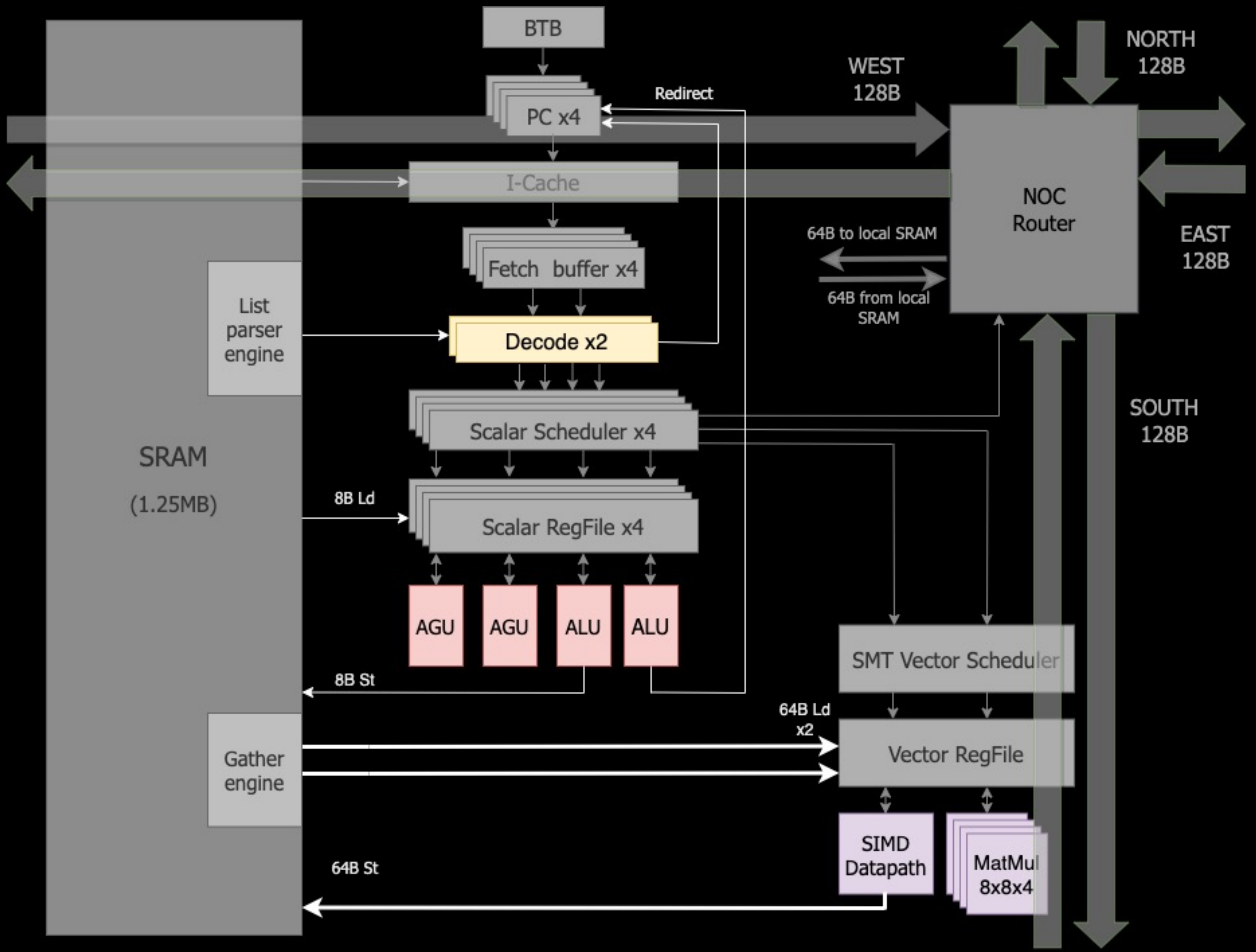
Datapath

Pipeline width reduces progressively
- 8-way in Decode
- 4-way in the Scalar engine
- 2-way in the Vector engine

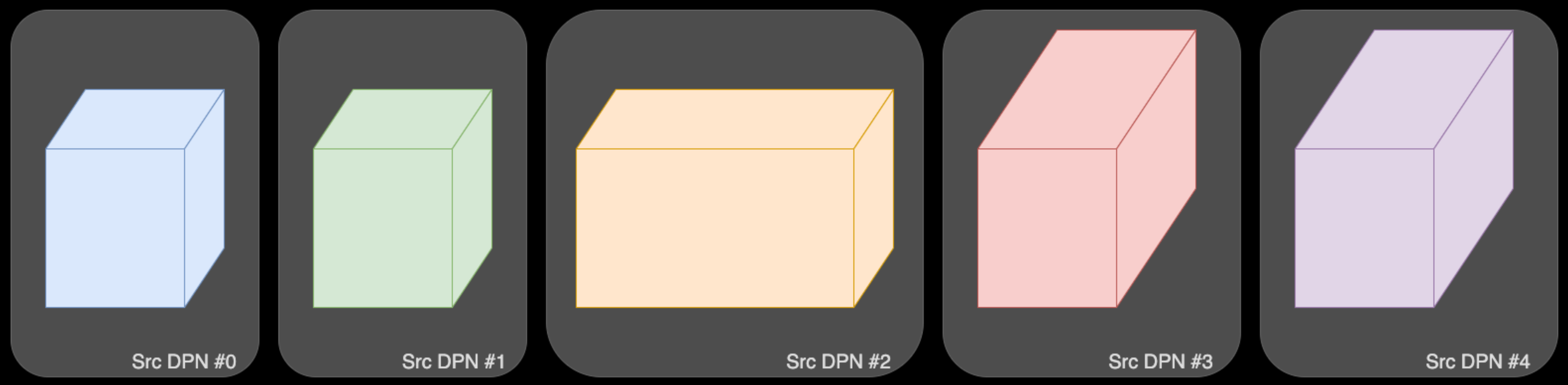
Simple primitives can execute early in Decode
- Looping, list parsing
- Predication

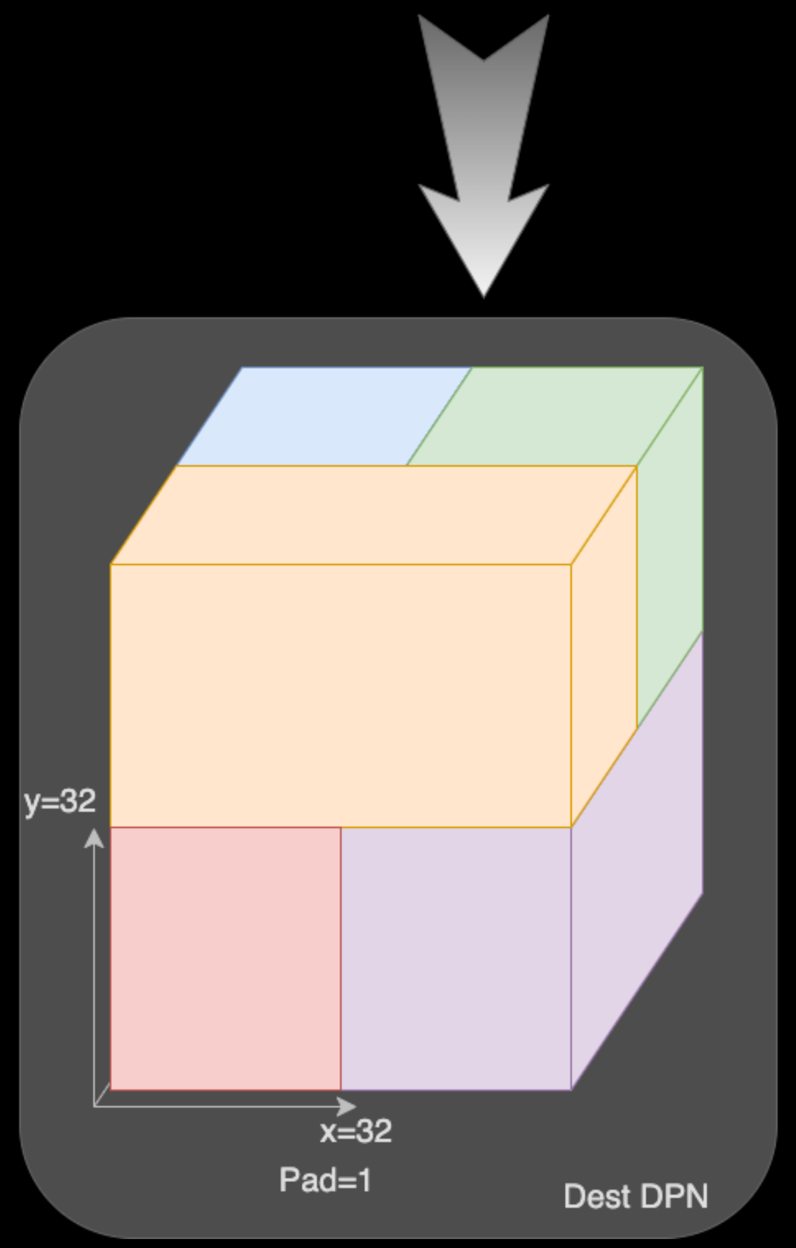
Scalar instructions
- Regular integer code, address generation
- Network synchronization primitives

Vector datapath
- 8x8 matrix multiplication instructions
- 64B SIMD pipeline
- Special ML formats (CFP8, storage CFP16)
- Special ML instructions (e.g., stochastic rounding, etc.)



List parsing





CONCAT operations list

Type 0, Cnt 34
Type 2, Cnt 544, Addr 0, Node 2
Type 2, Cnt 33, Addr 18, Node 3
Type 2, Cnt 33, Addr 19, Node 4
Type 2, Cnt 33, Addr 36, Node 3
Type 2, Cnt 33, Addr 37, Node 4
Type 2, Cnt 33, Addr 54, Node 3
. . . .

```
#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

DOJO Instruction Set

Fully featured, general-purpose instruction set

- 64b scalar instructions
- 64B wide SIMD instructions
  - Load+execute encodings to reduce pressure on architectural registers
  - Masked execution

Network transfer and synchronization primitives

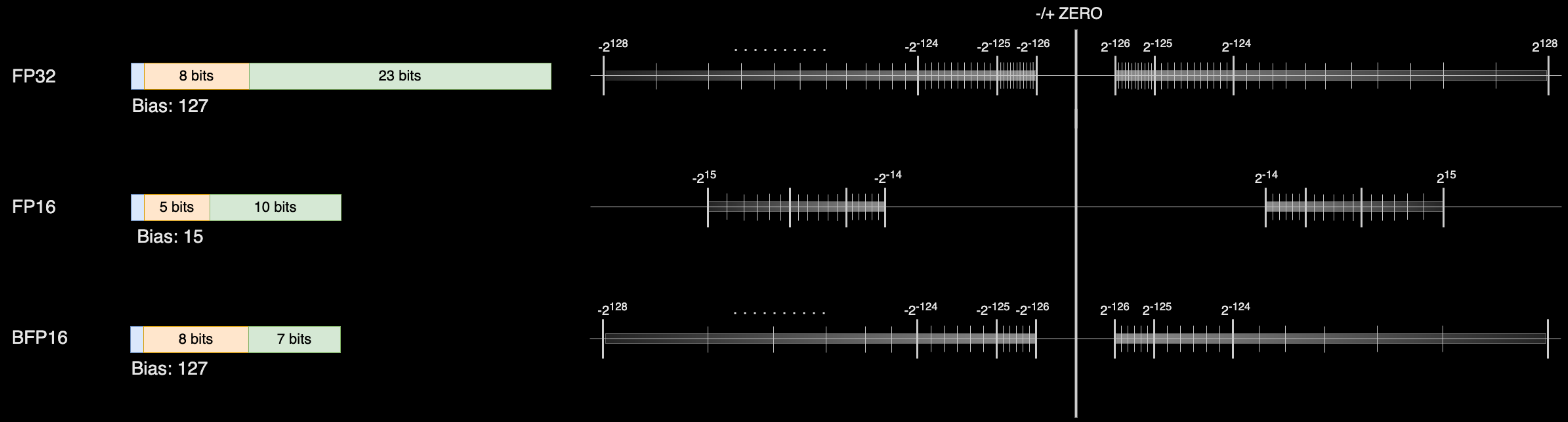
- Local to remote memory transfers
- Semaphore and barrier support

ML specific primitives

- 8x8 Matrix multiplication engine
  - Inline support for loading and gathering operands, transposing or expanding compressed operands
- Special set of shuffle, transpose and convert instructions
- Stochastic rounding
- Implicit 2D padding

Inst Type	Unique opcodes	Variants
Front-end	12	21
Scalar	74	143
Vector	142	1095

DOJO arithmetic formats

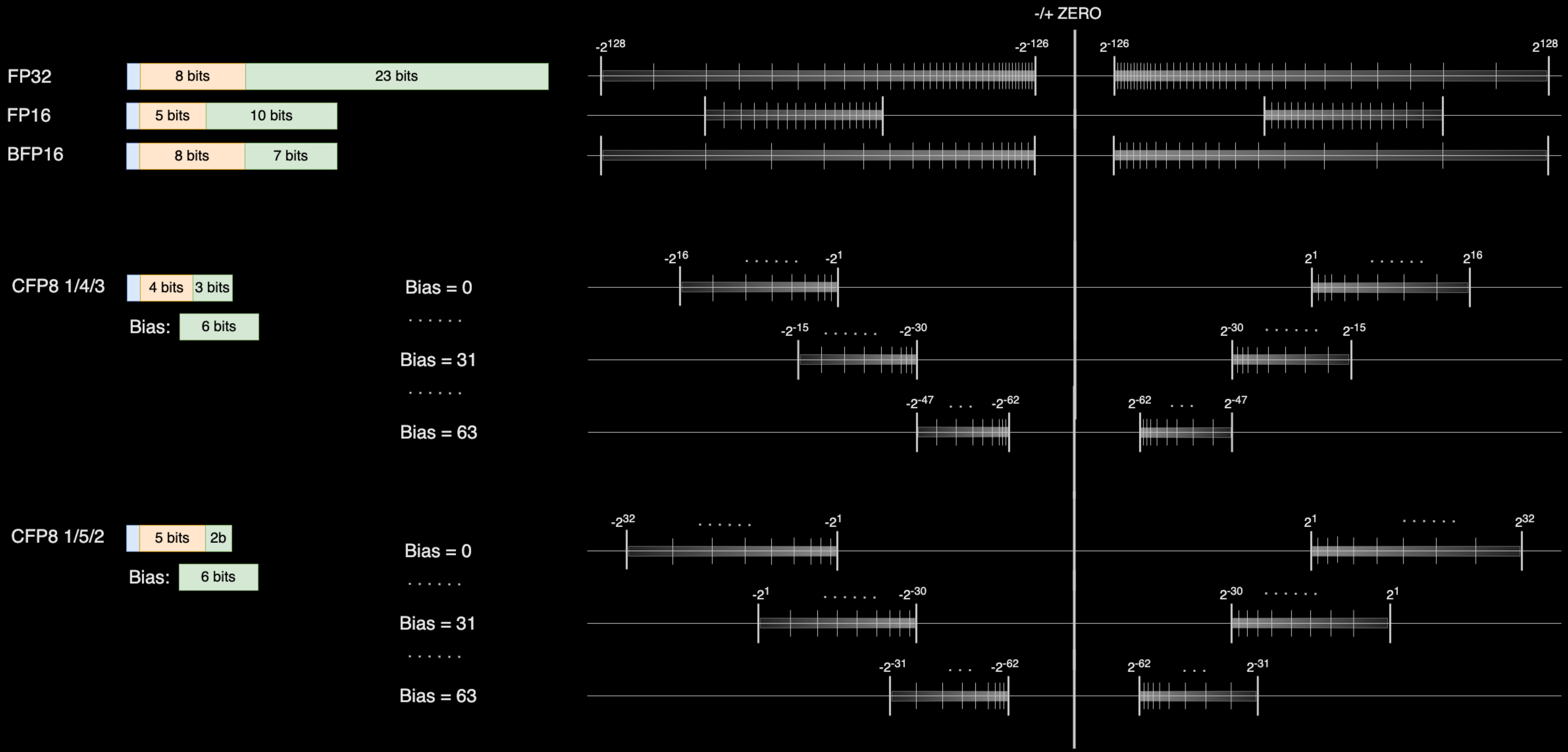


FP32 has more range and precision than the application requires

16b FP formats allow higher representation density, with some usage restrictions

- **IEEE FP16 does not have enough range to cover all layers**
- **BFP has more range, but sparser coverage density**

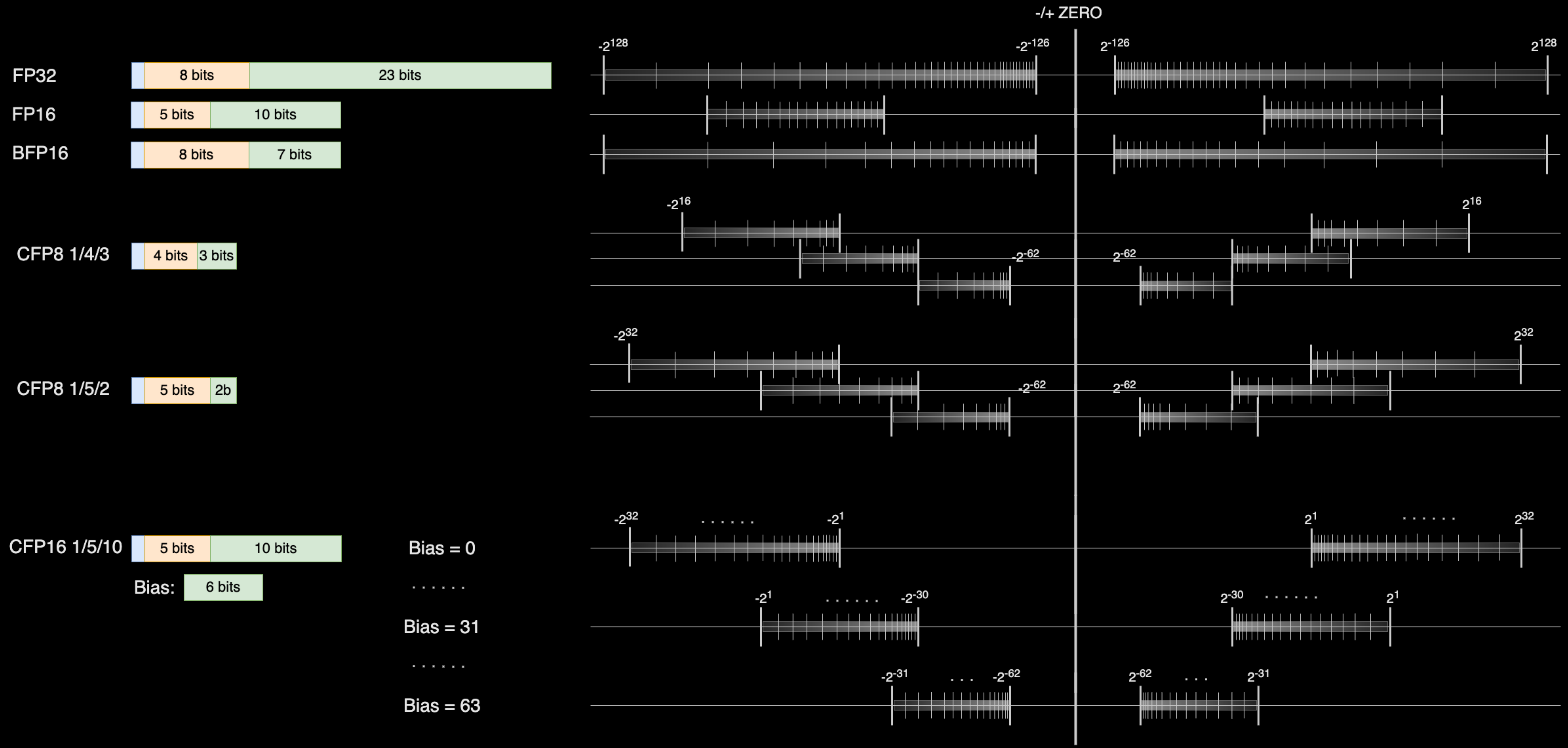
DOJO arithmetic formats



CFP8 has even higher total representation range

Configurable bias allows the application to shift the representation range based on local needs

DOJO arithmetic formats



CFP16 can be used when higher precision is required (e.g., gradients)

DOJO implements FP32, BFP16, CFP8 and CFP16

Up to 16 data types can be used at any time, with each 64B packet sharing a single type

First integration box - D1 Die

TSMC 7nm, 645mm^2

Physically and logically arranged as a 2D array
- 354 DOJO processing nodes on die

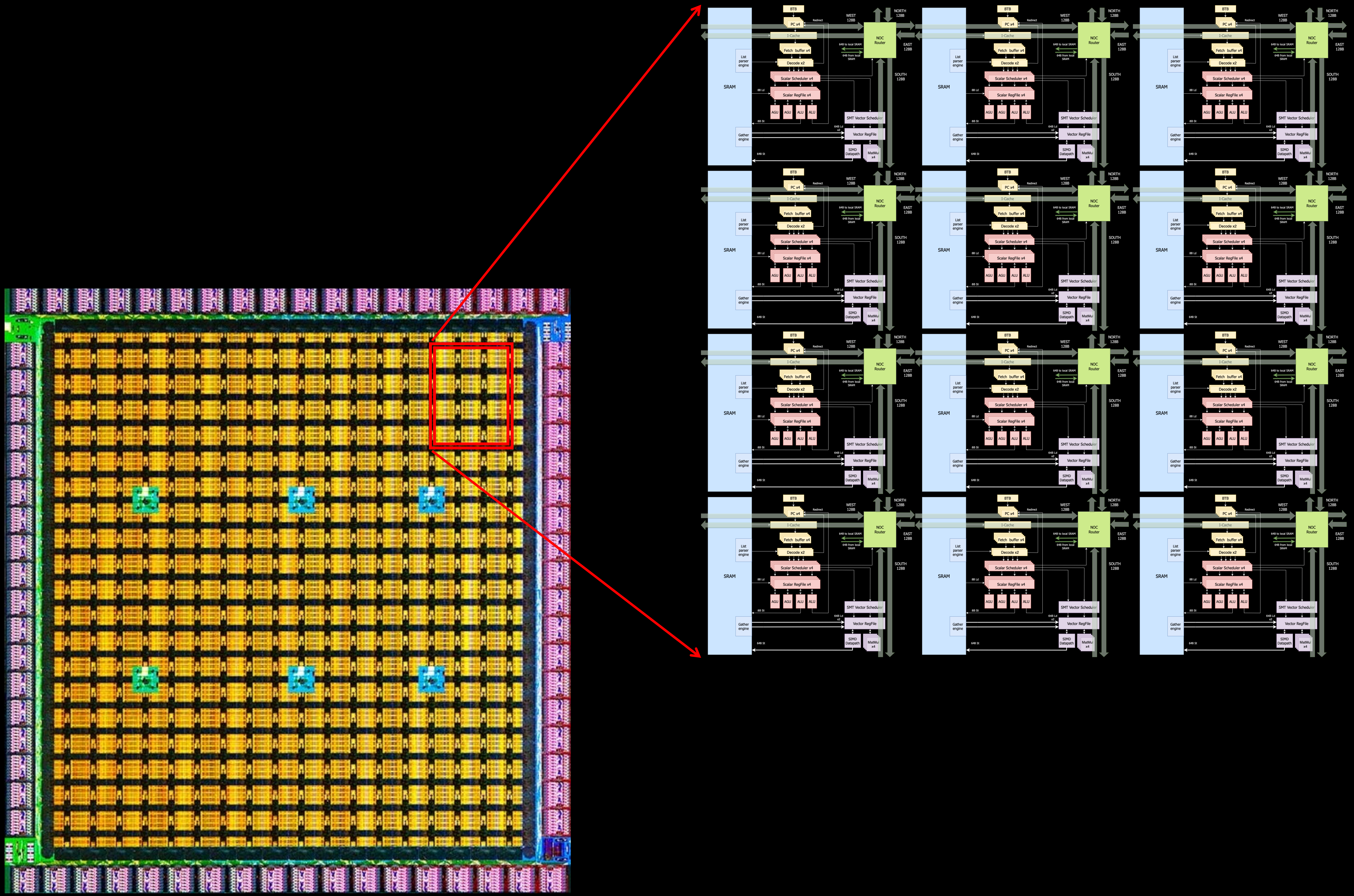
Extremely modular design

362 TFlops BF16/CFP8, 22 TFlops FP32 @2GHz

440 MB SRAM

Custom low power serdes channels on all edges
- 576 bidirectional channels
- 2 TB bandwidth on each edge

Seamless connection to neighboring dies



TESLA

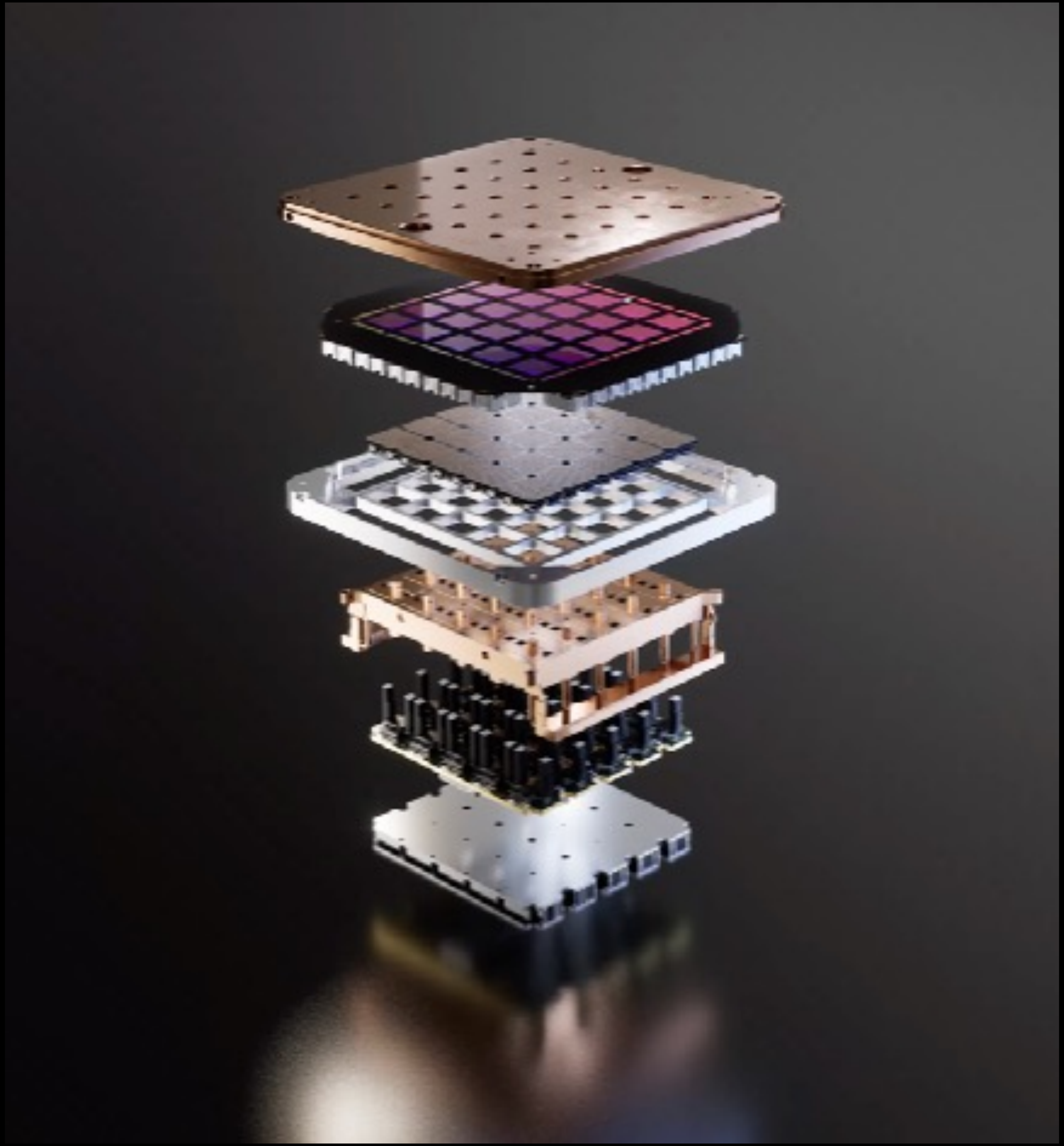
Second integration box – Dojo Training Tile

5x5 array of known good D1 chips
- 4.5TB/s off-tile bandwidth per edge
  - Half of in-tile bandwidth

Fully integrated module
- Electrical + thermal + mechanical
- 15kW of power delivery

Custom power delivery
- Horizontal data communication plane
- Vertical power delivery and cooling
- 15kW per module

Custom high-density connectors
- Seamless connection to neighboring training tiles



DOJO System Topology

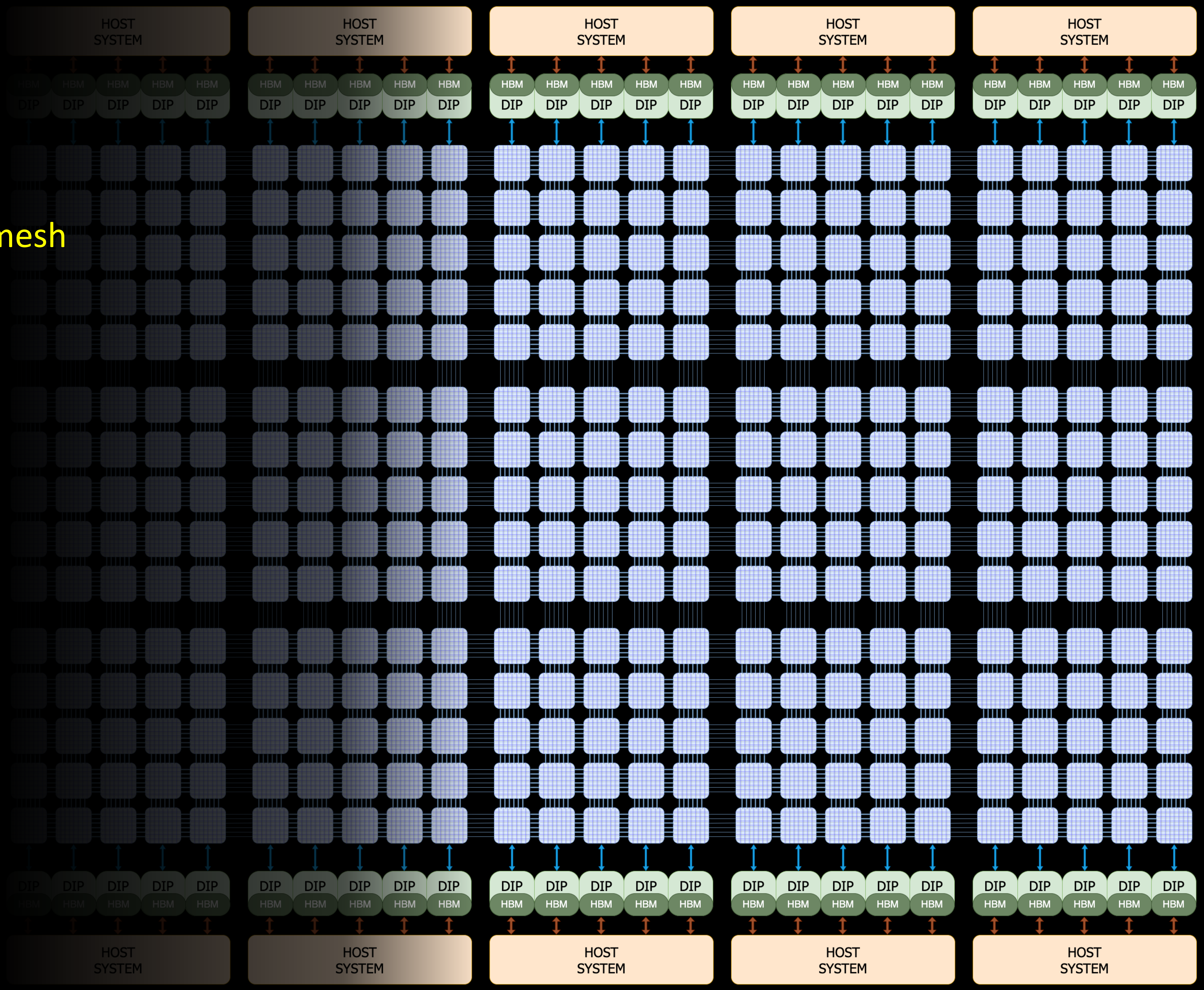
Full system is a plane of DOJO training tiles organized as a 2D mesh

DOJO interface processors

- Located on the edges of the mesh
- Provide connectivity to the outside world
- Provide shared memory support

Shared DRAM vs. private SRAM

- Each training tile
  - 11GB of private SRAM, highly distributed
  - 160GB of shared DRAM, lower bandwidth but more contiguous

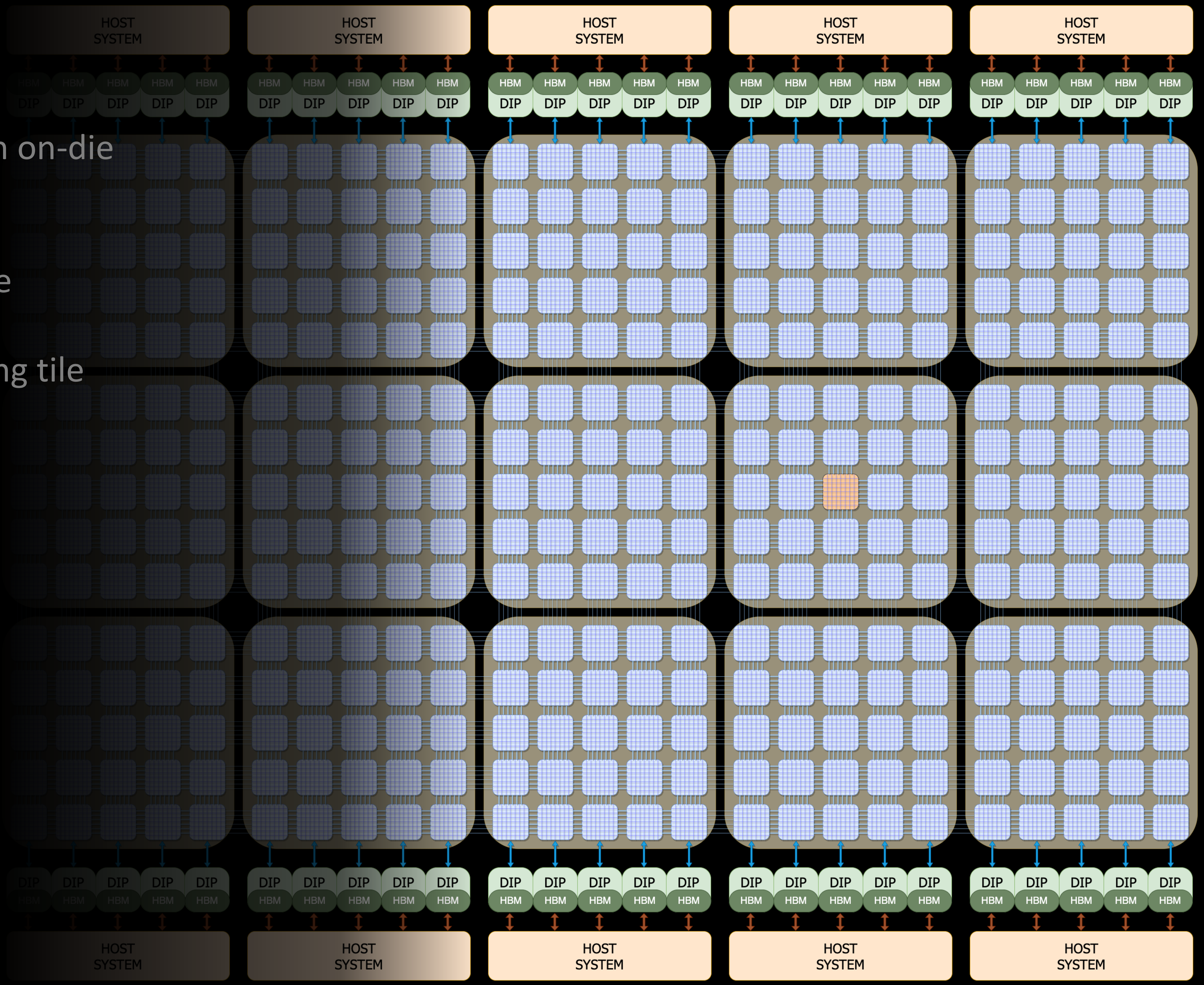


TESLA

Communication mechanisms

Logical 2D mesh connecting all processing nodes

- 256GBps bidirectional bandwidth on every row and column on-die
  - Single cycle per hop
  - 5TB cross-section
- 2TB bandwidth on every D1 die edge within the training tile
  - 100ns die-to-die latency
- 900GBps bandwidth on every D1 die edge out of the training tile

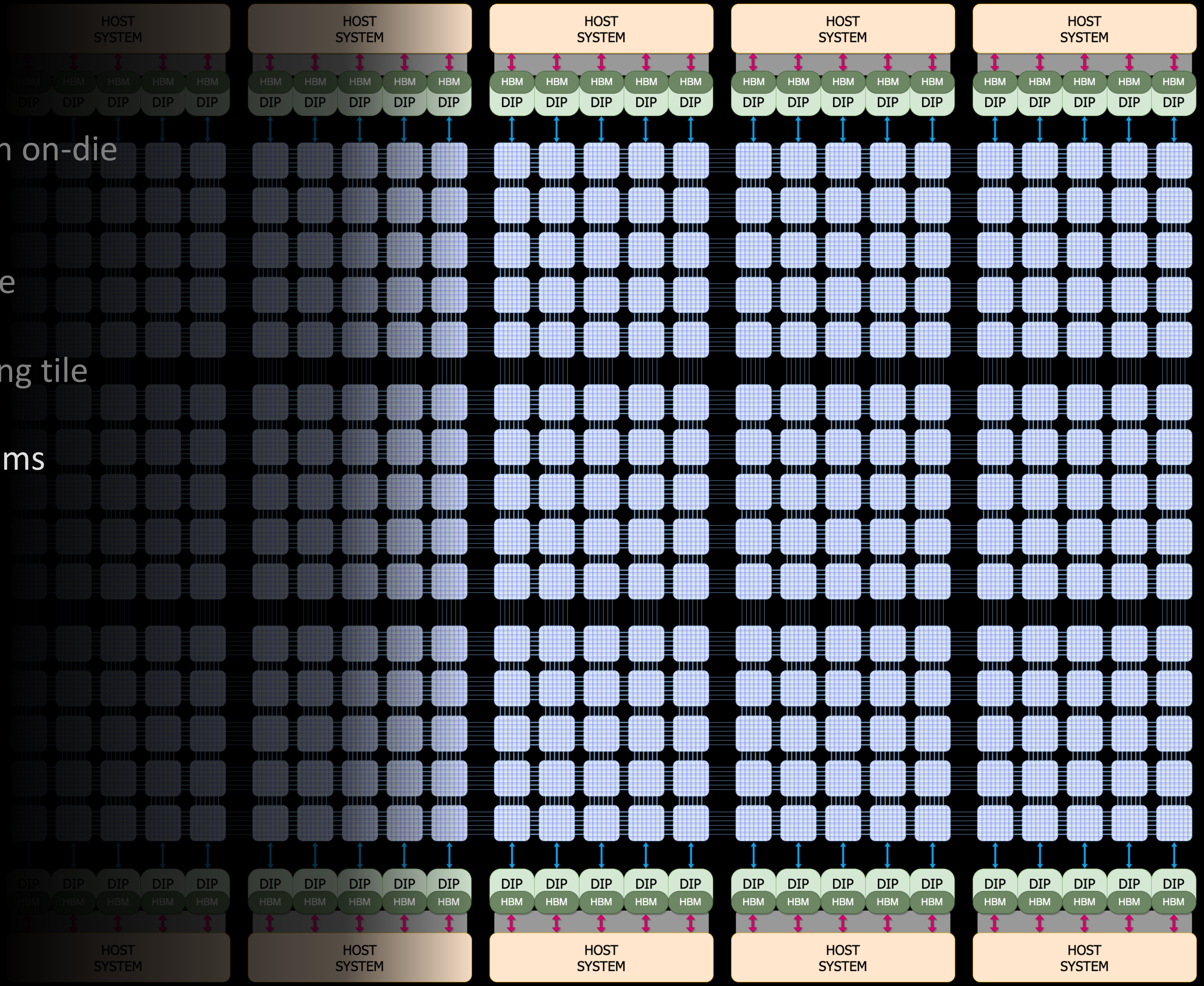


TESLA

Communication mechanisms

Logical 2D mesh connecting all processing nodes
- 256GBps bidirectional bandwidth on every row and column on-die
  - Single cycle per hop
  - 5TB cross-section
- 2TB bandwidth on every D1 die edge within the training tile
  - 100ns die-to-die latency
- 900GBps bandwidth on every D1 die edge out of the training tile

Edge communication - PCIe links between DIPs and host systems
- 32GBps bandwidth per DIP, 160GBps per host



Communication mechanisms

Logical 2D mesh connecting all processing nodes

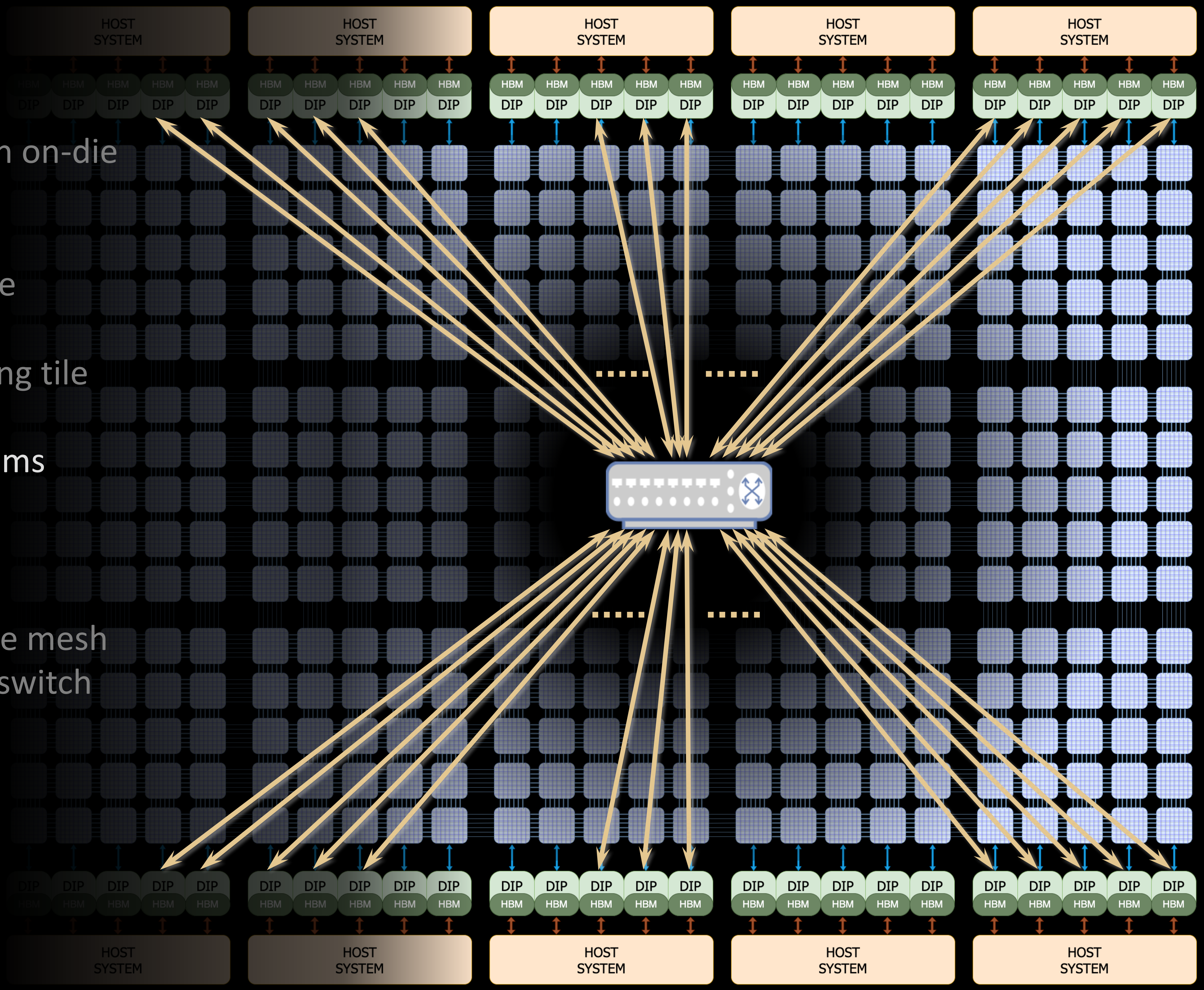
- 256GBps bidirectional bandwidth on every row and column on-die
  - Single cycle per hop
  - 5TB cross-section
- 2TB bandwidth on every D1 die edge within the training tile
  - 100ns die-to-die latency
- 900GBps bandwidth on every D1 die edge out of the training tile

Edge communication - PCIe links between DIPs and host systems

- 32GBps bandwidth per DIP, 160GBps per host

Global communication – Z-plane links between DIPs

- Very long global communication routes are expensive in the mesh
- Tesla Transport Protocol links all DIPs through an ethernet switch
- 32GBps bandwidth per DIP



Communication mechanisms

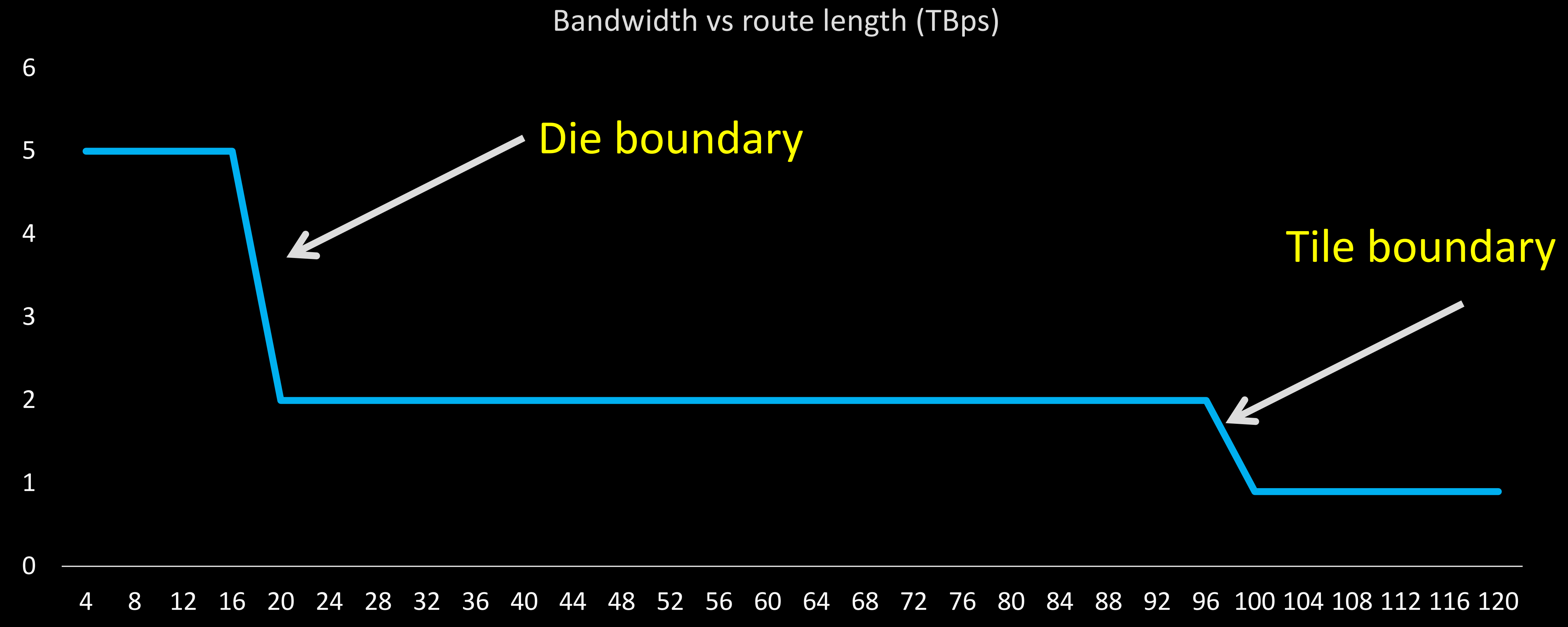
Why treat long range communication differently?

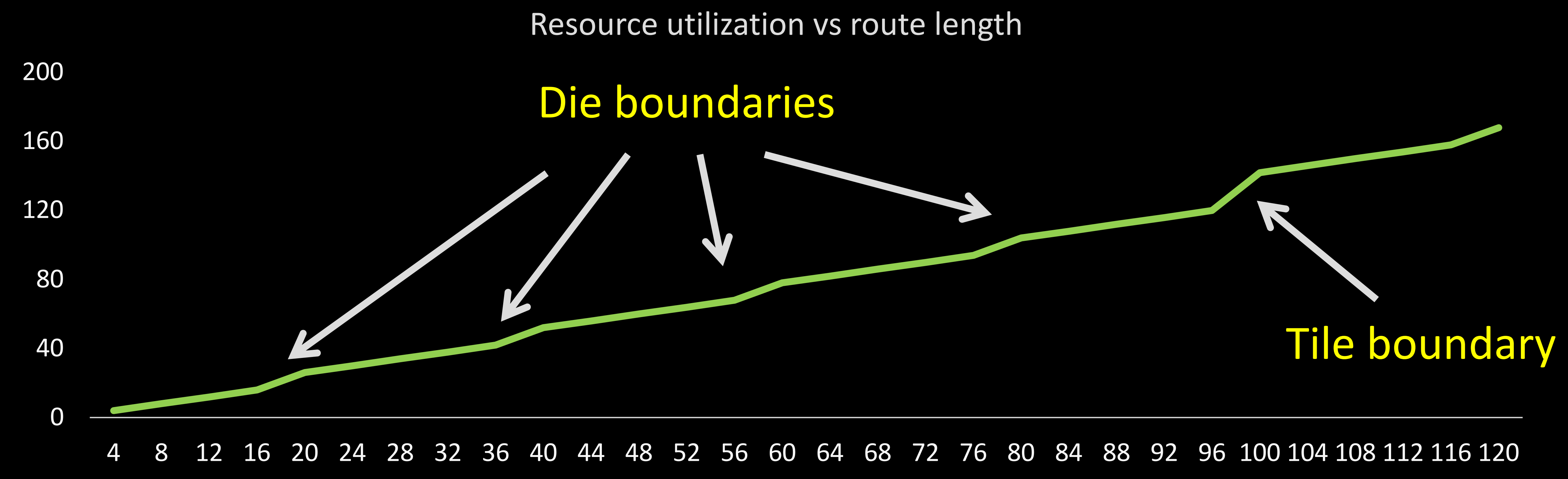
DOJO network has less bandwidth for long routes
- Long routes span multiple integration boundaries

Long routes consume a lot more of the system resources
- Packets need to traverse multiple local communication links

Strong incentive software to keep most communication local
- Amount of data transferred should go down quickly with distance

Restrict global traffic to synchronization and All Reduce primitives
- Data parallel instances should be kept relatively small





TESLA

Bulk Communication

Any processing node can access data stored anywhere in the system

- Can issue PUSH or PULL requests for each packet
- Can use bulk transfer requests to initiate larger transfers

SRAM to DRAM

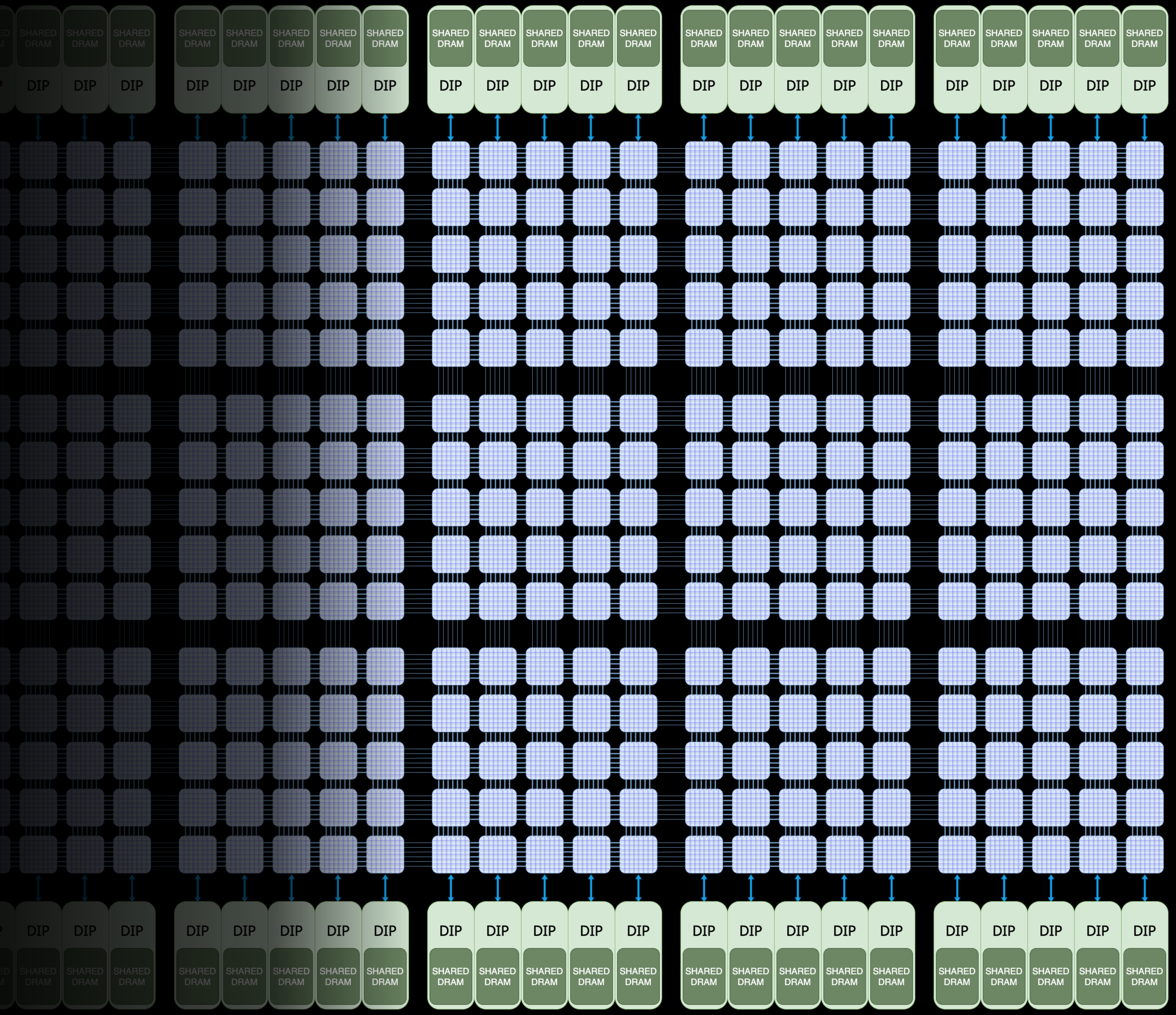
- DIPs accept DMA commands to copy data to and from DRAM
  - Contiguous or strided transfers up to 3 dimensions

SRAM to SRAM

- PUSH Broadcast
  - Enabled within sets of nodes within the same D1 die
- PUSH / PULL memory region
  - Copy a contiguous region from a single source to a single destination

List-based operations

- Allow complicated data gather and scatter patterns
- Dedicated instructions sequences or threads utilizing the list parsing engines



TESLA

DOJO System Network

Target hardware simplicity!

Flat addressing scheme exposes system topology to software

D1 address format:

Die Address Y	Die Address X	Node Addr. X	Node Addr. Y	Local address

DIP address format:

Die Address Y	Die Address X	Local address

TESLA

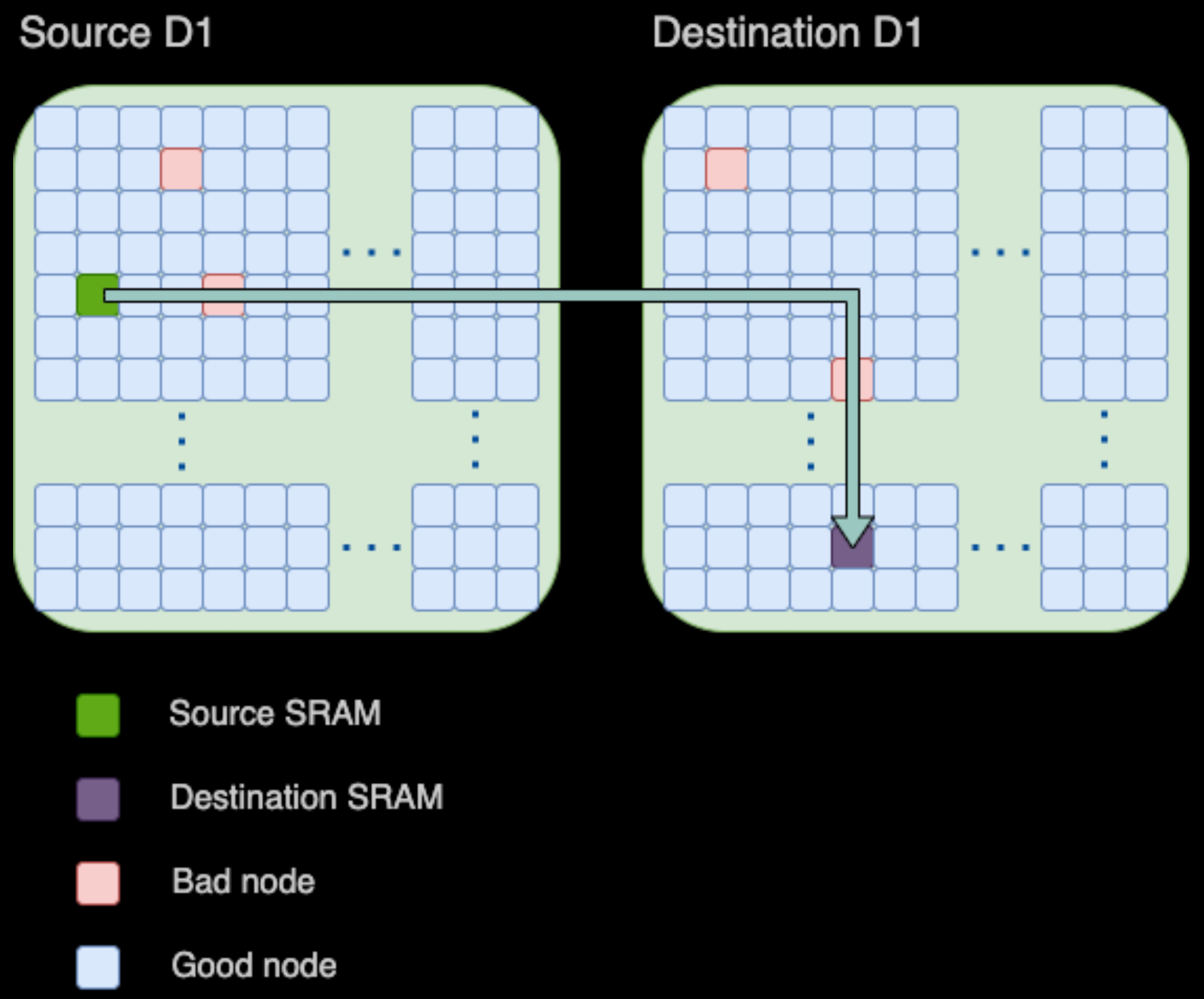
DOJO System Network

Target hardware simplicity!

Flat addressing scheme exposes system topology to software

Routing scheme

- Simple 2D routing within the destination D1 die
  - All routers must work on a functional die
  - Dead processing nodes are avoided by SW



TESLA

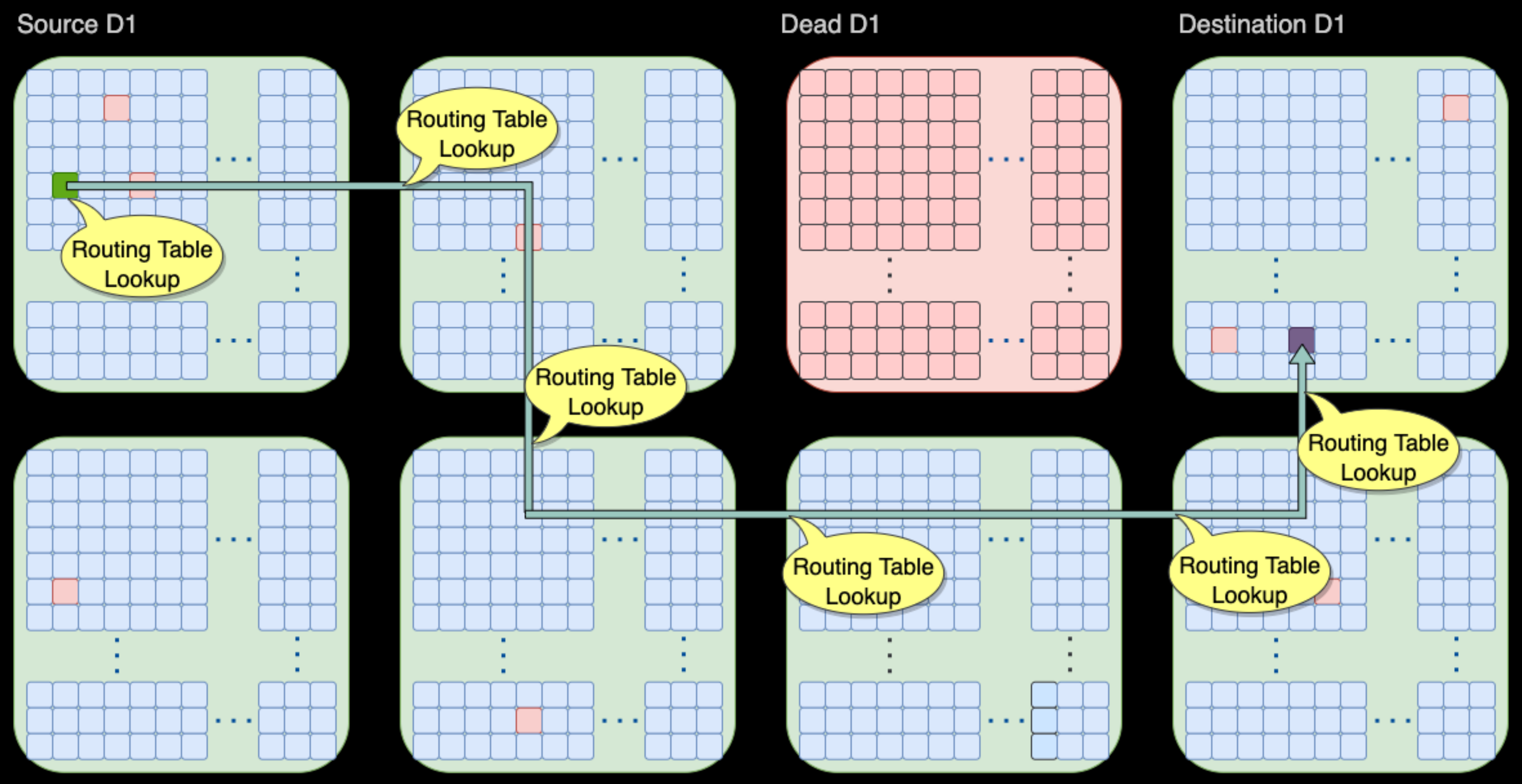
DOJO System Network

Target hardware simplicity!

Flat addressing scheme exposes system topology to software

Routing scheme

- Simple 2D routing within the destination D1 die
  - All routers must work on a functional die
  - Dead processing nodes are avoided by SW
- Each D1 die has a programable routing table
  - Can route packets towards the TTP-based Z dimension
  - Allows routing around dead D1 dies



TESLA

DOJO System Network

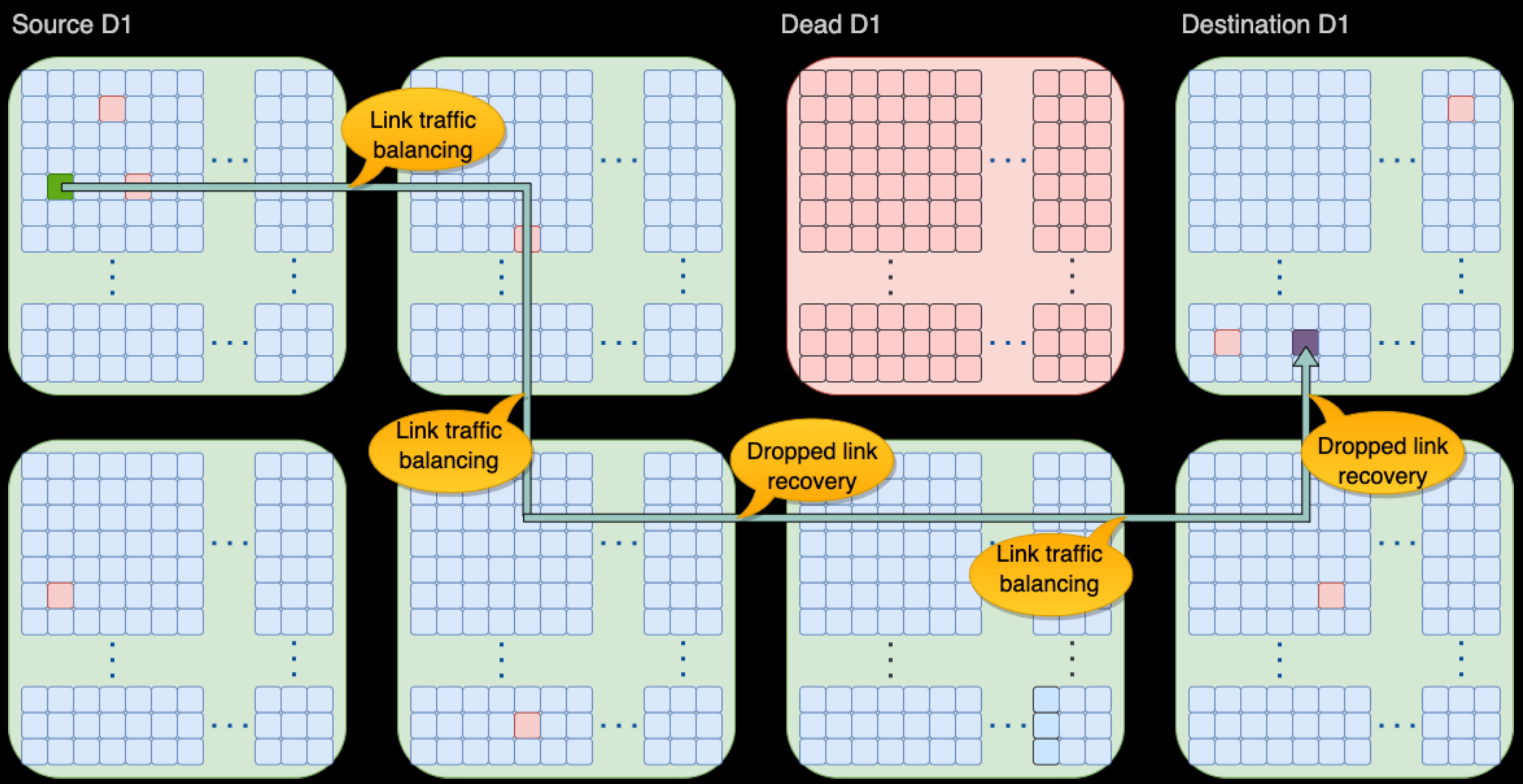
Target hardware simplicity!

Flat addressing scheme exposes system topology to software

Routing scheme
- Simple 2D routing within the destination D1 die
  - All routers must work on a functional die
  - Dead processing nodes are avoided by SW
- Each D1 die has a programable routing table
  - Can route packets towards the TTP-based Z dimension
  - Allows routing around dead D1 dies

Packet ordering
- DOJO network does not guarantee end-to-end traffic ordering

Counting
- Arriving packets must be counted at destination before use



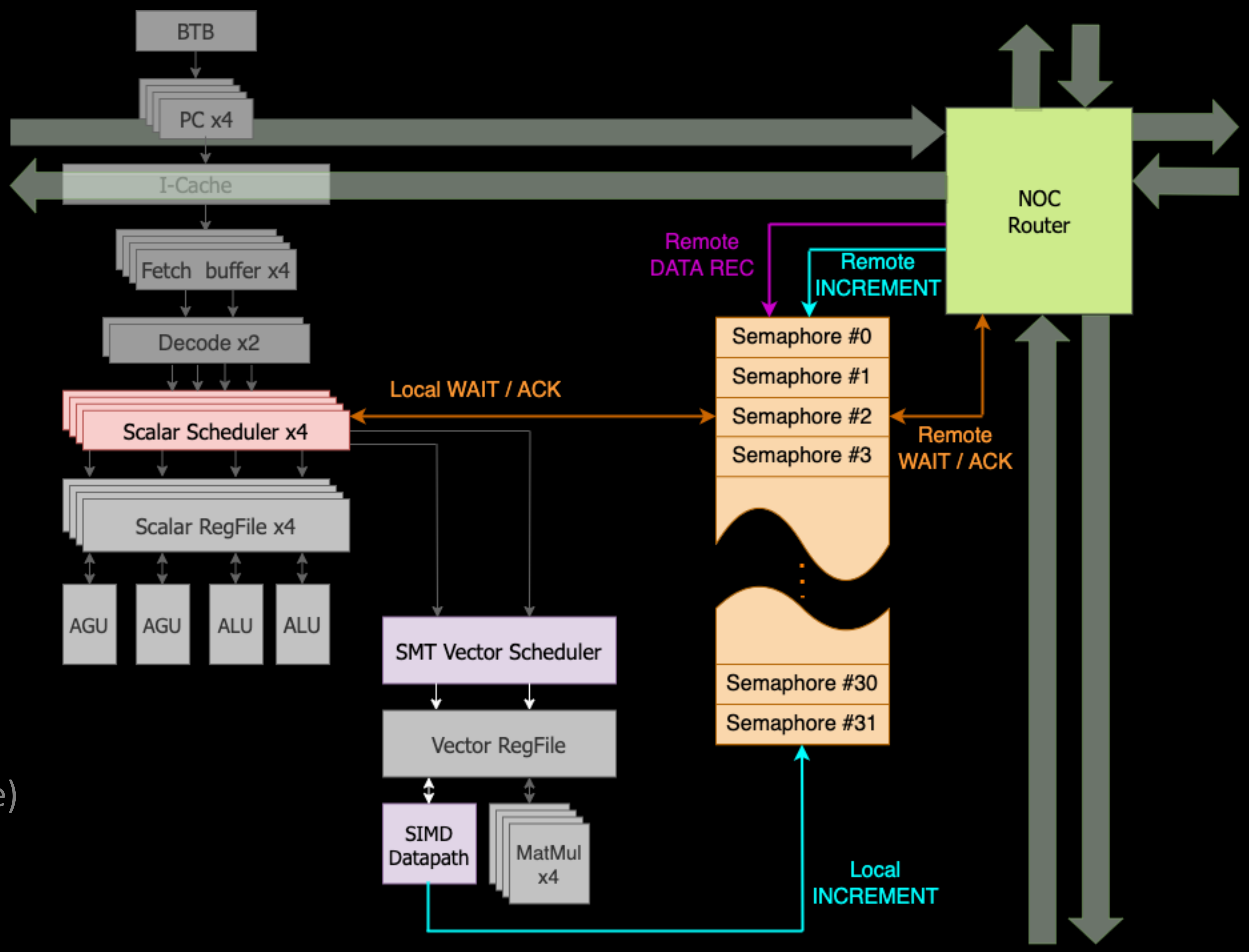
DOJO System Synchronization

Counting semaphores are hardware managed event trackers
- HW ensures update atomicity and starvation avoidance

Event generation
- Execute S(emaphore)SET on one of the local threads
- Execute R(emote)S(emaphore)SET on a remote thread
- Receive data packet from the network

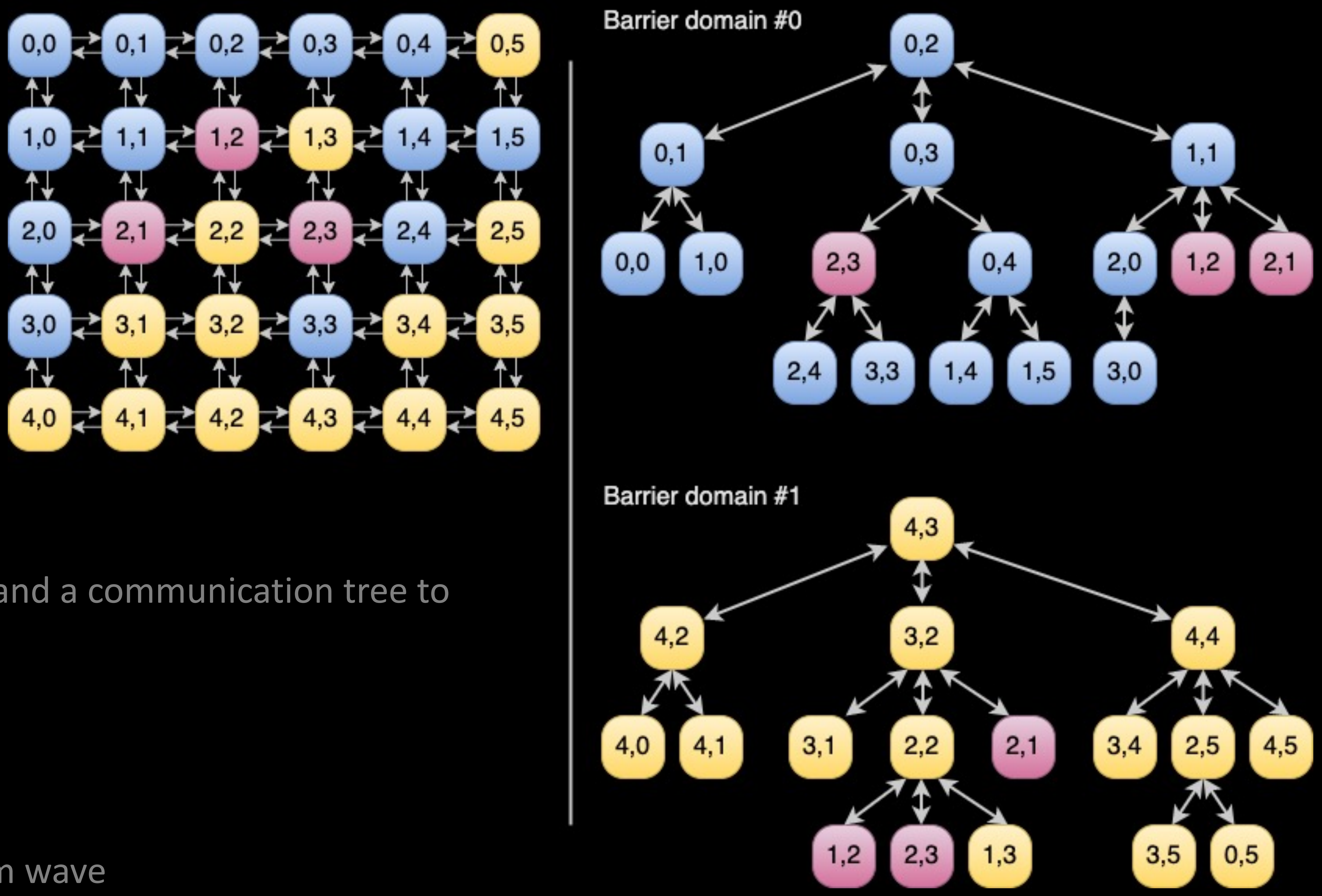
Event monitoring
- Execute S(emaphore)WAIT on one of the local threads
- Execute R(emote)S(emaphore)WAIT on a remote thread
- Waiting threads are put to sleep until condition is met

Typical use models
- Mutex
- Producer-consumer thread synchronization (local and remote)
- Completion for network transfers



TESLA

DOJO System Synchronization



Software defined barrier trees

- Compiler defines sets of nodes for each barrier domain and a communication tree to cover each set
- Barrier domains can span any number of nodes
- Nodes can be assigned to multiple barrier domains

Software signals reaching and checking the barrier

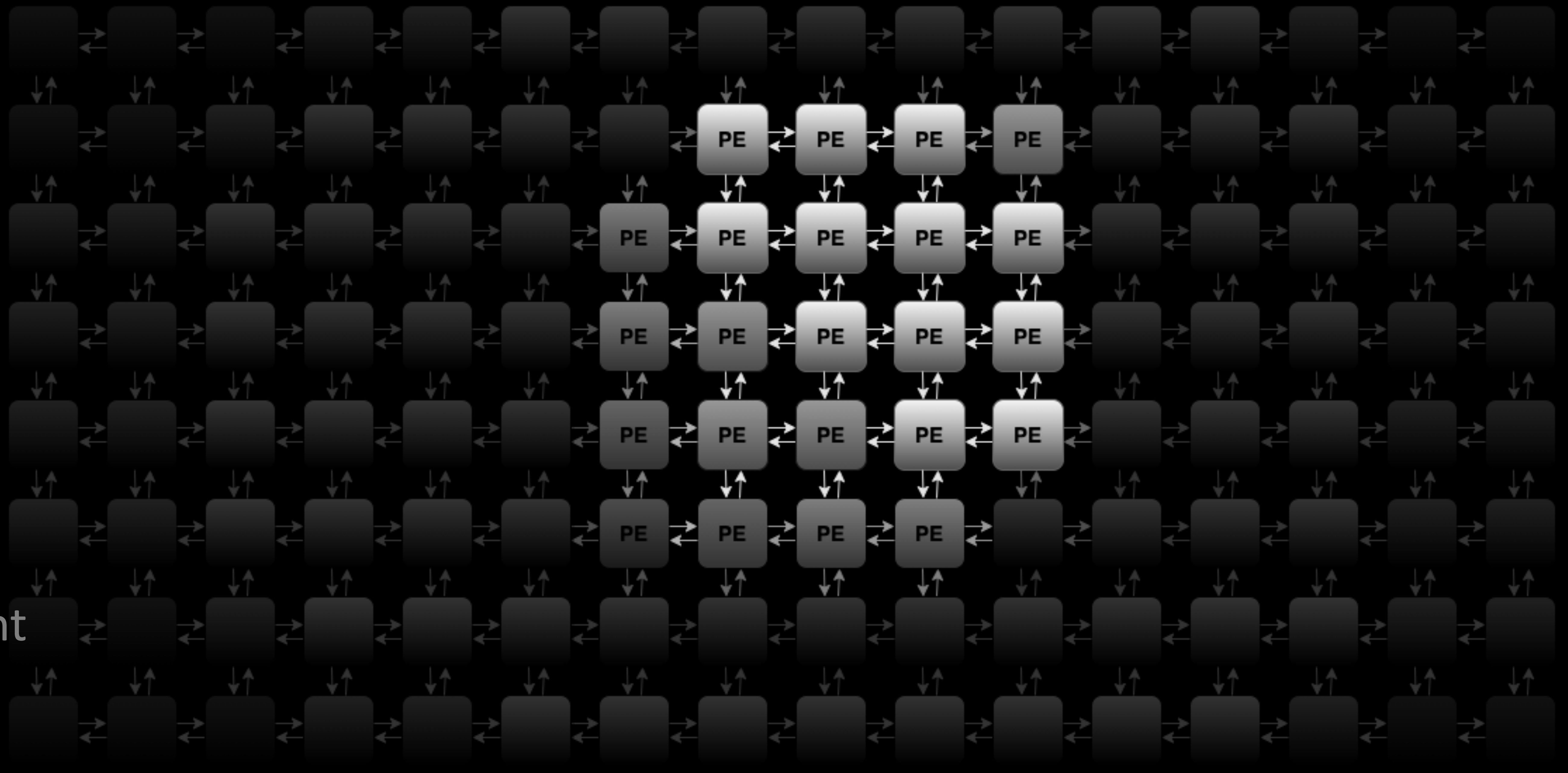
- All nodes execute B(arrier)ARM to complete an upstream wave
- Root node triggers a downstream wave
- All nodes execute B(arrier)CHECK to wait until the downstream wave reaches them

Summary

DOJO is a large-scale distributed system
- One exa-pod has more than 1,000,000 CPUs

DOJO component modules
- At the edge between general-purpose and application-optimized hardware

Defining characteristic is extreme scalability
- De-emphasize poorly scaling mechanisms like virtual memory, coherency, global data structures
- DOJO relies less on local storage and more on fast data movement
- Order of magnitude higher interconnect bandwidth than typical distributed systems



Integration and utilization of these components is the subject of the next presentation!