

AI Accelerator Terminology Cheat Sheet

Shared roles, different architectures. Read each column within its stated scope.

Concept & meaning	NVIDIA GPUs Hopper / CUDA	AMD Instinct GPUs CDNA 3 / HIP	Google TPU v6e / Pallas	AWS Trainium Trainium2 / NKI
01 Execution	How work is expressed and scheduled			
Execution Model How a program expresses parallel work; the execution models differ.	SIMT A kernel runs as many threads. 1	SIMT HIP threads map to wavefronts. 6	Vector/tile program A sequential instruction stream with asynchronous units. 10	Tensor program Sequential semantics; independent engine work may overlap. 14
Work Grid The index space used to divide an operation into program instances.	Grid of thread blocks blockIdx + threadIdx 1	Grid of thread blocks blockIdx + threadIdx 6	Pallas grid Generally iterated sequentially on a core. 10	SPMD launch grid nl.program_id identifies an instance. 20
Cooperation Scope A scope for sharing temporary data within local work.	Thread block Threads share block shared memory. 1	Thread block / work-group Threads share LDS. 6 7	Kernel program Tile operations use VMEM/SMEM buffers. 11	Kernel instance Engine operations use SBUF/PSUM tiles. 15 14
Execution Group The hardware grouping below a program; these are different abstractions.	Warp 32 threads. 1	Wavefront 64 work-items on CDNA 3; HIP calls it a warp. 7 6	Vector lanes / sublanes Elements of a VREG, not GPU threads. 10	Engine-wide tile operation No CUDA-style warp in the NKI model. 14 19
Matrix Operation An instruction or kernel API that requests matrix multiply-accumulate work.	mma / wgmma Warp / asynchronous warp-group operations. 3	MFMA Matrix fused multiply-add instructions. 7	lax.dot / lax.dot_general Pallas lowers matrix operations to MXUs. 10	nisa.nc_matmul Stationary and moving operand tiles. 17
02 Hardware	Physical compute blocks and engines			
Compute Block A repeated block containing arithmetic and local storage; sizes and capabilities differ.	SM Streaming Multiprocessor. 2	CU Compute Unit; CUs are grouped in XCDs. 5	TensorCore One per v6e chip; includes matrix, vector and scalar units. 9	NeuronCore-v3 Eight physical cores per Trainium2 chip. 13
Matrix Engine Specialized hardware for dense matrix arithmetic.	Tensor Core Inside an SM. 21	Matrix Core Inside a CU. 5	MXU Matrix-multiply unit; two per v6e TensorCore. 9	Tensor Engine Systolic matrix engine in a NeuronCore. 19
General Arithmetic Hardware for work around matrix multiplication, such as elementwise operations and control.	CUDA cores / SM pipelines Thread arithmetic and other instructions. 21	VALU + SALU Vector work and wave-uniform scalar work. 7	Vector unit + scalar unit Array arithmetic and scalar control. 9 11	Vector, Scalar, GPSIMD The Scalar Engine also operates across many elements. 19
03 Memory	Where values live and how they move			
Device Memory Large accelerator-attached storage for tensors.	Global memory / HBM For the H100 example. 1 2	Global memory / HBM3 For the MI300 example. 5	HBM Staged into on-chip memory for computation. 9 11	HBM Explicit transfers to and from SBUF. 15
Operand Scratchpad Software-managed on-chip storage for reusable input and intermediate tiles.	Shared memory Block-scoped; Hopper also supports cluster access. 1 2	LDS Local Data Share; work-group allocation. 7	VMEM / SMEM Vector SRAM / scalar SRAM. 11	SBUF State Buffer shared by the core's compute engines. 15
Register Storage Storage closest to arithmetic; ownership and allocation differ.	Registers Per-thread architectural values. 1	VGPRs / SGPRs Per-lane vector / wave-uniform scalar values. 7	VREGs / SREGs Vector / scalar registers. 11	Scalar VirtualRegister NKI tensor values live in SBUF/PSUM. 14 15
Matrix Accumulators Where partial matrix results are kept between multiply-accumulate steps.	Register accumulators For Hopper mma / wgmma. 3	VGPRs / AccVGPRs Destination depends on the MFMA form. 7	FP32 vector values For floating-point Pallas dot; VREGs and VMEM scratch. 10 11	PSUM Partial Sum Buffer; dedicated matrix-result memory. 15 17
Tensor Layout How logical tensor elements map to the storage consumed by an instruction.	Register fragments / shared descriptors MMA operands use instruction-specific register or shared-memory layouts. 3	MFMA operand layout Instruction-specific lane/VGPR mapping. 7	VREG lanes / sublanes Tiling and padding follow vector dimensions. 10	Partition + free dimensions SBUF/PSUM use partitioned 2D storage. 16
Bulk Data Movement A mechanism to stage large tiles while arithmetic can proceed.	TMA Tensor Memory Accelerator; global to/from shared copies. 2	Global-to-LDS loads ISA loads can target LDS; completion needs a wait. 7	DMA HBM to/from VMEM/SMEM copies. 11	DMA engines HBM to/from SBUF; independent of compute engines. 13
Cache or Explicit Buffer Whether fast-memory contents are filled transparently or by program-visible transfers.	L1 / L2 caches Shared memory is explicitly managed. 2	L1 / L2 caches L2 is shared by CUs within an XCD. 5	Explicit VMEM/SMEM staging Pallas exposes SRAM buffers and DMA. 11	Software-managed hierarchy SBUF/PSUM are not transparent hardware caches. 15
Private Temporaries Storage for values local to execution; "local" does not imply on-chip SRAM.	Local memory Thread-private device-memory space, including spills. 1 4	Scratch memory Private per-work-item storage in memory. 7	VREG spills to VMEM Program scratch is not per-thread local memory. 10	SBUF intermediates Kernel-local tensor storage in the NKI model. 15
04 Coordination	Ordering work and exchanging data			
Local Ordering Ensure producers finish before dependent local consumers proceed.	__syncthreads() A converged block barrier; not a grid barrier. 1	__syncthreads() HIP block barrier. ISA s_barrier alone does not complete memory operations. 8 7	Program dependencies Sequential core program; asynchronous work needs completion tracking. 10 11	Program dependencies NKI preserves sequential semantics when scheduling engines. 14
Transfer Completion Wait for an in-flight data transfer before reading or reusing its buffers.	cuda::barrier / pipeline Track asynchronous shared-memory copies. 2	s_waitcnt ISA counters track relevant outstanding memory operations. 7	DMA semaphore + wait Copy descriptors track completion. 11 12	Compiler dependencies NKI schedules required ordering between DMA and consumers. 14
Chip-to-Chip Exchange Dedicated links carry data between accelerator chips; topology and scope vary.	NVLink GPU-to-GPU links; separate from the SM hierarchy. 2	Infinity Fabric links Connect MI300X GPUs within an eight-GPU node. 5	ICI Inter-chip interconnect; Pallas exposes remote DMA. 9 12	NeuronLink-v3 Trainium2 device-to-device communication. 13
05 Performance	Precision, utilization and data access			
Matrix Input Precision Example arithmetic formats; a tensor's storage type alone does not select the math.	FP8 / FP16 / BF16 / TF32 Examples of Hopper Tensor Core modes. 3	FP8 / FP16 / BF16 / INT8 Examples of CDNA 3 Matrix Core modes. 5 7	BF16 / INT8 Documented v6e matrix compute examples. 9	FP8 / FP16 / BF16 / TF32 Examples of Trainium2 Tensor Engine modes. 13
Accumulation Precision The running sum can be wider than the multiply inputs; mode and instruction matter.	FP32 accumulation Available for BF16/FP16 matrix operations. 3	MFMA_F32 forms BF16/FP16 inputs can accumulate into FP32. 7	FP32 floating-point dot results FP32 inputs may be rounded to BF16 by default. 10	FP32 PSUM NKI matmul writes FP32 partial results. 17
Latency Hiding Overlap useful work with data waits. Occupancy counts resident state, not utilization.	Warp occupancy + pipelines Resident warps and asynchronous tile staging. 2 4	Wave occupancy + pipelining Resident waves and staged LDS data. 8 7	Software pipelining Overlap DMA with a core's computation. 11	Engine/DMA overlap Pipeline tiles; inspect each engine's utilization. 18
Memory Access Efficiency Match data access patterns to the memory system to reduce wasted transfers.	Coalescing / bank conflicts Warp global accesses / shared-memory banks. 4	Coalescing / LDS banking Wave accesses / shared-memory banks. 8	Block shape / padding Tile layout must fit the vector/matrix units. 10	DMA efficiency / partition layout Block transfers and reuse SBUF tiles. 16 18
Branching and Selection Conditional work can waste parallel arithmetic; control and selection differ by model.	Warp divergence / predication Different thread paths can reduce efficiency. 1 4	Wave divergence / EXEC mask Inactive lanes do not contribute useful work. 7 8	Scalar control / vector select No GPU thread-warp divergence model. 10	Compiled scalar loops Python branches are resolved during specialization. 14

Notes
Same Role, Different Unit
An SM, CU, Google TensorCore and NeuronCore have different sizes and capabilities. Google TensorCore names a compute block; NVIDIA Tensor Core names a matrix engine.

Scratchpads Have Different Scopes
CUDA shared memory and AMD LDS support thread cooperation. TPU VMEM and Trainium SBUF serve tile programs and engines. These are role comparisons, not interchangeable memory models.

Precision Has Three Parts
Distinguish operand storage, multiply precision and accumulation. FP32 output does not guarantee full FP32 multiplication. Format support and throughput vary by instruction and generation.

Scope: NVIDIA Hopper / CUDA 13.0.2; AMD Instinct MI300 / CDNA 3 / HIP; Google TPU v6e / Pallas; AWS Trainium2 / NeuronCore-v3 / Neuron 2.33.0. Examples are not exhaustive feature lists.