UC San Diego

Think Fast: A Tensor Streaming Processor (TSP) for Accelerating Deep Learning Workloads

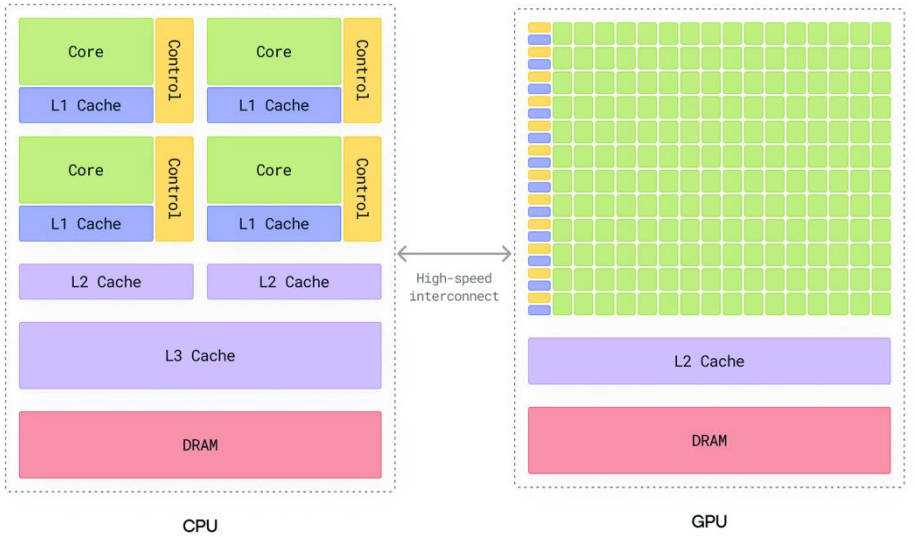
Dennis Abts, Jonathan Ross, Jonathan Sparling, Mark Wong-VanHaren, Max Baker, Tom Hawkins, Andrew Bell, John Thompson, Temesghen Kahsai, Garrin Kimmell, Jennifer Hwang, Rebekah Leslie-Hurd, Michael Bye, E.R. Creswick, Matthew Boyd, Mahitha Venigalla, Evan Laforge, Jon Purdy, Purushotham Kamath, Dinesh Maheshwari, Michael Beidler, Geert Rosseel, Omar Ahmad, Gleb Gagarin, Richard Czekalski, Ashay Rane, Sahil Parmar, Jeff Werner, Jim Sproch, Adrian Macias, Brian Kurtz

Groq, Inc. · Mountain View, California - 2022

Presented by: Shoumik Panandikar

The Cost of Generality

- CPUs & GPUs built for unpredictable, general workloads
- This demands expensive hardware:
  - Caches → hide unpredictable memory latency
  - Branch predictors → handle arbitrary control flow
  - Arbitration logic → manage resource contention
- DNNs are regular and predictable so we're paying for complexity we don't need



DNNs Are Predictable by Nature

- Same operations, same order, every inference
- Tensor shapes known at compile time
- No branches, no dynamic dispatch
- Same op applied to thousands of elements independently
- Weights are fixed during inference

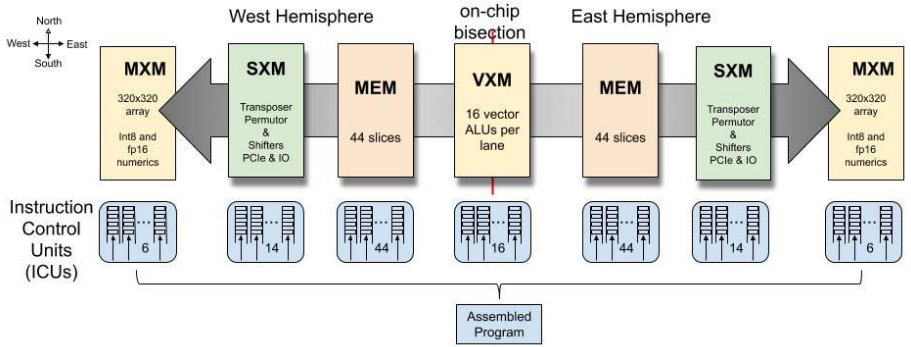
If the hardware knows what's coming, why prepare for anything else?

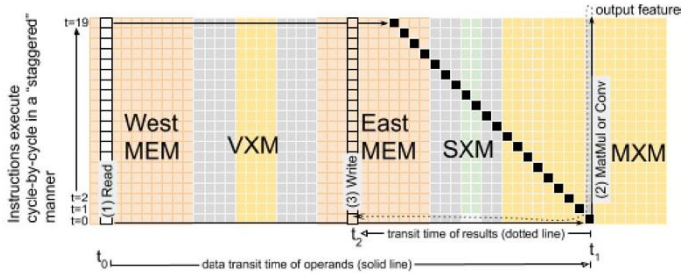
Rethinking the Hardware-Software Contract

- **Compiler-driven execution** — all scheduling done at compile time, not runtime
- **No dynamic hardware** — no caches, no arbiters, no branch predictors
- **Deterministic performance** — latency predictable to the exact clock cycle
- **5× compute density** — 30K ops/transistor vs GPU's 6.2K
- **ICU = only 3% of chip area** — transistors saved on control go directly to compute

Functional Slicing

- Chip reorganized into vertical columns by function — not identical cores
- Each column does ONE thing: MEM, VXM, MXM, or SXM
- Instructions flow North ↑, data flows East → West ←





Five Functional Slices

ICU

Instruction Control Unit

- Fetches & dispatches instructions
- Sync + Notify for barrier sync
- NOP for cycle-precise scheduling
- Only 3% of chip area!

MEM

Memory Slices

- 88 slices × 2.5 MiB = 220 MB total
- Reads/writes 32 bytes/lane/cycle
- Gather/Scatter for indirect addressing
- 55 TiB/s SRAM bandwidth

MXM

Matrix Execution Module

- Four 320×320 MACC arrays
- Int8 and fp16 supported
- Loads 409,600 weights in < 40 cycles
- Workhorse of conv2D + MatMul

VXM

Vector Execution Module

- 4×4 mesh of ALUs per lane
- Point-wise ops: ReLU, add, mul
- Batch norm, quantization, activations
- 5,120 total vector ALUs on chip

SXM

Switch Execution Module

- Transpose, permute, rotate data
- North/South lane shifters
- Inter-lane movement in Y-dimension
- Essential for tensor reshape ops

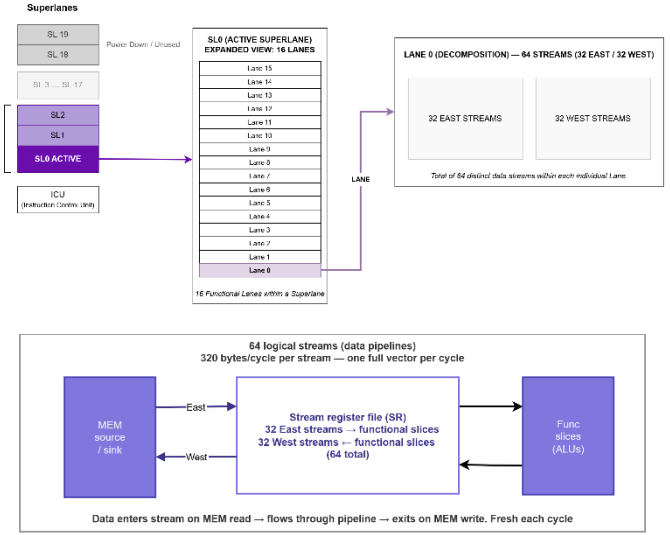
Streams & the Programming Model

Superlanes

- 20 superlanes × 16 lanes = 320 elements per vector
- Each superlane is the minimum unit of computation
- Unused superlanes powered down

The Programming Model

- Producer-consumer: each slice consumes from one stream, produces to another
- 64 logical streams total → 32 flowing East, 32 flowing West
- Compiler tracks every stream's position and timing cycle-by-cycle



ResNet50 Inference Flow on TSP

←→ Streams flow East/West across chip — no intermediate writeback between chained steps

Compiler Scheduling
All 144 instruction queues scheduled cycle-accurately. Operands meet instructions at precise tile + time.

MXM → VXM Chaining
MXM int32 output streams directly into VXM — no MEM writeback between steps, saving ~2 cycles/layer.

Bank Interleaving
Layer N inputs read from MEM while layer N-1 outputs are still being written — no pipeline bubble.

What the compiler does

2D Scheduling:

- Assigns every instruction a precise cycle AND spatial location.
- Tracks all 144 instruction queues simultaneously.

Stream Assignment:

- Decides which stream ID carries which data, which direction (East/West), and when it will arrive at each functional slice.

Memory Layout Body:

- Allocates tensors across 88 MEM slices to maximize concurrency.
- Interleaves banks so reads/writes don't conflict.

NOP Insertion

- Inserts NOP instructions as precise cycle-level delays to ensure instructions and data meet at the correct tile.

Results

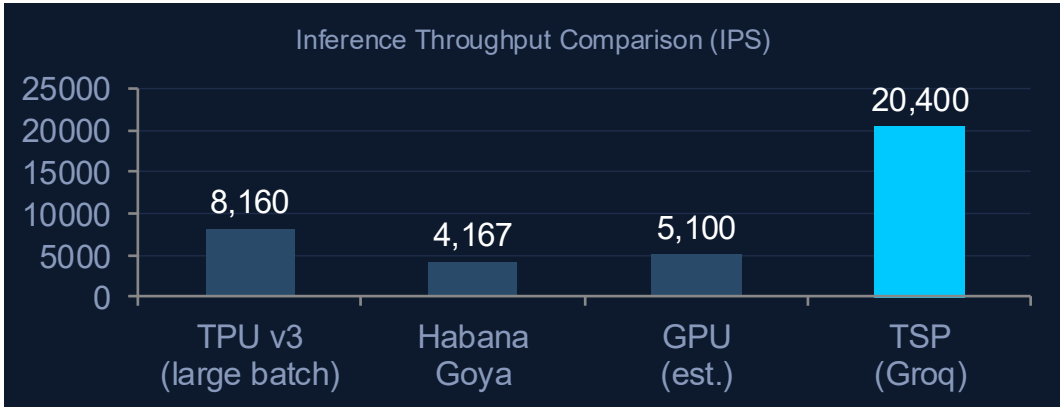
UC San Diego

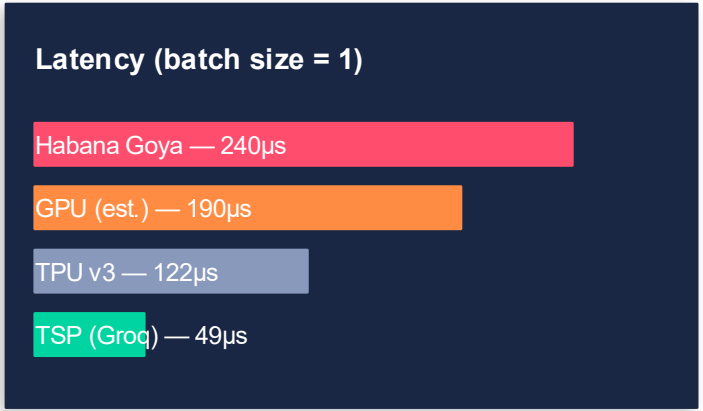
20,400
IPS
Images per second
(batch size = 1)

49 µs
Latency per image
(single query)

4–5×
Faster than Google TPU v3
and Habana Goya

820
TOps/s
Peak performance
@ 1 GHz





Limitations & Shortfalls

- **Inference only** — designed and optimised exclusively for inference
- **Fixed shapes** — variable-length inputs require full recompilation
- **Sparse computation** — all MACCs fire even on zero elements
- **No control flow** — one fixed execution path, decided at compile time
- **No custom kernels** — new ops need compiler + ISA support
- **Compiler dependency** — every new model needs slow recompilation

TSP wins at inference on standard models.
That's the design, not an oversight.

From ResNet50 to Large Language Models

What Carries Over:

- **Matrix multiply is still the bottleneck:** attention (QKV), feed-forward layers, embeddings are all dense matmuls, which is exactly what MXM was built for
- **Weights still static during inference:** a deployed LLM has fixed parameters, same as ResNet50
- **Data parallelism still abundant:** each token processed independently across attention heads
- **Determinism still valuable:** chat applications need consistent response latency

What Gets Harder:

- **Variable sequence length:** unlike images (always 224×224), prompts vary from 10 to 100,000 tokens → TSP must recompile or pad aggressively
- **KV cache is dynamic:** attention requires storing past token states that grow with conversation length → hard to pre-schedule
- **Model size:** Llama 70B has 70 billion parameters which far exceeds 220MB on-chip SRAM → requires multi-chip with C2C links
- **Autoregressive generation:** each token depends on all previous tokens → sequential dependency that limits parallelism

From 2020 Paper to Today

UC San Diego

The Architecture Held Up

- TSP rebranded → **LPU (Language Processing Unit)** as LLMs took over
- Same architecture, same determinism, now running transformer models
- 300 tokens/sec on Llama 70B → **10× faster than NVIDIA H100**
- The 2020 paper's bet on determinism paid off exactly where it mattered most

Commercial Timeline

- **2020** — ISCA paper published
- **2024** — GroqCloud launches: 1.9M+ developers, Meta Llama API partnership
- **Feb 2025** — $1.5B Saudi Arabia infrastructure deal
- **Sept 2025** — $6.9B valuation (Series E)
- **Dec 2025** — NVIDIA licenses TSP architecture for **$20 billion***

*https://groq.com/newsroom/groq-and-nvidia-enter-non-exclusive-inference-technology-licensing-agreement-to-accelerate-ai-inference-at-global-scale

UC San Diego

Thank you!