Software-defined Hardware with Groq’s Tensor Streaming Processor

A hardware-software approach
HELLO ATPESC

Andrew Ling, Ph.D.
Sr. Director Software
MY GOAL TODAY

Introduce you to Groq’s approach to SW defined hardware in the era beyond CPUs
Think Fast: A Tensor Streaming Processor (TSP) for Accelerating Deep Learning Workloads

Dennis Abt Groq Inc.
Matt Boyd Groq Inc.
Roberto DeCesco Groq Inc.
Jennifer Hwang Groq Inc.
Elyas Mehrabzadeh Groq Inc.
Andrew Ling Groq Inc.
Salil ParMar Groq Inc.
John Thompson Groq Inc.
Peter Lillians Groq Inc.
Thomas Schrimmers Groq Inc.
Jonathan Ross Groq Inc.

ABSTRACT

We describe a novel software-defined architecture for a large-scale interconnection network of tensor streaming processors (TSPs) that incorporates the communication and synchronization primitives of a network-on-chip substrate for global communication. This scalable communication fabric provides a hardware backend for large-scale systems based on a software-defined Dragongeny topology, ultimately enabling a scalable tensor processing system with extensibility to support a variety of workloads, both training and inference. We introduce a flexible tensor streaming framework that simplifies global memory allocation, as implemented in logic and hardware. The TSP architecture provides the necessary infrastructure to enable global-scale communication as a commodity, which is central to the development of novel deep learning systems.

CCS CONCEPTS

• Computing systems organization → Interconnection architectures

1 INTRODUCTION

Heterogenous, high-performance computing (HPC) systems were broadly categorized as either GPU or CPU systems. These systems excelled in applications because of communication latency and bandwidth. More recently, deep learning systems have emerged, which are often based on deep neural networks to identify patterns and solve complex problems (e.g., image recognition). These systems are comprised of specialized hardware and software components that are tailored to efficiently process and analyze large-scale datasets. The interconnection networks in these systems are crucial for enabling effective communication between the various components and for scaling the system to handle large datasets.

A Software-defined Tensor Streaming Multiprocessor for Large-scale Machine Learning

Dennis Abt Groq Inc.
Matt Boyd Groq Inc.
Roberto DeCesco Groq Inc.
Jennifer Hwang Groq Inc.
Elyas Mehrabzadeh Groq Inc.
Andrew Ling Groq Inc.
Salil ParMar Groq Inc.
John Thompson Groq Inc.
Peter Lillians Groq Inc.
Thomas Schrimmers Groq Inc.
Jonathan Ross Groq Inc.

ABSTRACT

We describe a novel software-defined architecture for a large-scale interconnection network of tensor streaming processors (TSPs) that incorporates the communication and synchronization primitives of a network-on-chip substrate for global communication. This scalable communication fabric provides a hardware backend for large-scale systems based on a software-defined Dragongeny topology, ultimately enabling a scalable tensor processing system with extensibility to support a variety of workloads, both training and inference. We introduce a flexible tensor streaming framework that simplifies global memory allocation, as implemented in logic and hardware. The TSP architecture provides the necessary infrastructure to enable global-scale communication as a commodity, which is central to the development of novel deep learning systems.

CCS CONCEPTS

• Computing systems organization → Interconnection architectures

1 INTRODUCTION

Heterogenous, high-performance computing (HPC) systems were broadly categorized as either GPU or CPU systems. These systems excelled in applications because of communication latency and bandwidth. More recently, deep learning systems have emerged, which are often based on deep neural networks to identify patterns and solve complex problems (e.g., image recognition). These systems are comprised of specialized hardware and software components that are tailored to efficiently process and analyze large-scale datasets. The interconnection networks in these systems are crucial for enabling effective communication between the various components and for scaling the system to handle large datasets.

The interconnection network is the backbone of a deep learning system, providing the necessary infrastructure to enable global-scale communication as a commodity. Traditional interconnection networks were designed to support specific applications and were limited in their programmability. In contrast, software-defined networking (SDN) allows for the dynamic allocation of resources and the efficient use of network bandwidth, which is essential for the scalability and performance of deep learning systems.

The key components of a software-defined tensor streaming multiprocessor (TSP) are described in this document. The architecture is designed to support both training and inference workloads, allowing for the efficient execution of deep learning models. The TSP architecture also features a flexible tensor streaming framework that simplifies global memory allocation, providing a hardware backend for large-scale systems based on a software-defined Dragongeny topology. This scalable communication fabric enables the development of novel deep learning systems that can handle large datasets efficiently.

CCS CONCEPTS

• Computing systems organization → Interconnection architectures

1 INTRODUCTION

Heterogenous, high-performance computing (HPC) systems were broadly categorized as either GPU or CPU systems. These systems excelled in applications because of communication latency and bandwidth. More recently, deep learning systems have emerged, which are often based on deep neural networks to identify patterns and solve complex problems (e.g., image recognition). These systems are comprised of specialized hardware and software components that are tailored to efficiently process and analyze large-scale datasets. The interconnection networks in these systems are crucial for enabling effective communication between the various components and for scaling the system to handle large datasets.

The interconnection network is the backbone of a deep learning system, providing the necessary infrastructure to enable global-scale communication as a commodity. Traditional interconnection networks were designed to support specific applications and were limited in their programmability. In contrast, software-defined networking (SDN) allows for the dynamic allocation of resources and the efficient use of network bandwidth, which is essential for the scalability and performance of deep learning systems.

The key components of a software-defined tensor streaming multiprocessor (TSP) are described in this document. The architecture is designed to support both training and inference workloads, allowing for the efficient execution of deep learning models. The TSP architecture also features a flexible tensor streaming framework that simplifies global memory allocation, providing a hardware backend for large-scale systems based on a software-defined Dragongeny topology. This scalable communication fabric enables the development of novel deep learning systems that can handle large datasets efficiently.
Groq in a Nutshell

**Founded**
2016

**Flagship Product**
Groq Tensor Streaming Processor Chip (TSP) and Software for use in AI, Machine and Deep Learning applications

**Target Market(s)**
AI and Machine Learning Hyperscalers, Government, HPC clusters, Autonomous vehicles, high performance edge appliances

**Employees**
~250

**Founder/CEO**
Jonathan Ross

WE ARE HIRING

ML Compilers and Tools
DNN Library Developers
Host-IO Developers
ASIC/HW Designers
Supporting Workloads of Today

- What would you build given the data flow nature of Machine Learning Workloads and HPC?
  - “Nodes” in the computational graph represent **operators** and “edges” are the **operands** and **results**.

- Groq re-examined the hardware-software contract
- Created hardware that is much more predictable and streaming based
- **Gave more control to software!**
Have we been here before? RISC \[\rightarrow\] VLIW?

With RISC, CPU architecture hides a lot of complex control logic from software

- Caching, prefetching, out-of-order execution, branch prediction
- Area cost of this control hardware reduces available area for raw compute

**Very Long Instruction Word (VLIW) to the rescue?**

The next natural step in the trend of shifting work and control from hardware to software

- Execution ordering, data prefetching, branch resolution
- Compiler responsible for exploiting instruction-level parallelism (ILP) available in a program

**Itanium...**

Compiler struggled to achieve good ILP due to limited view of dynamic hardware behaviour

- Cache misses, dynamic mem dependencies, dynamic branches
The end of CPU hegemony

Growing demand in dataflow dominated compute

Slowing of Moore’s Law and Dennard Scaling

CPU “abstraction” no longer the only foundation for developing software

Dawn of a new golden age in computing

Hennessy & Patterson: “A New Golden Age for Computer Architecture”

Lattner: “A New Golden Age of Compilers”

Karpathy: “Software 2.0” → A New Golden Age for Algorithms

In the past, hardware (CPUs) has defined our software

The hardware-software abstraction contract has been reopened!

Opportunity now to achieve “software-defined hardware”
Simplifying Compute

GroqChip™

Simplicity enables Compute Performance

Nvidia GTX 1070

Complexity leads to more Compute Costs
GroqChip Scalable Architecture

220MB SRAM
80 TB/s of Stream Bandwidth on-chip
Massive concurrency

Dense MatMul
720 TeraOP/s (1 TeraOP / s / mm²)

Vector Units

480GBps Chip-2-Chip Links

Dataflow

Instruction Control
Designing for determinism...

- Design choices along the way need to accommodate the “design for determinism” design philosophy

- Hardware must **enable** the compiler and runtime interfaces to reason about program execution
  - Memory consistency model must be well understood - disallowing memory references from being reordered
  - No “reactive components” like arbiters, crossbars, replay mechanisms, caches, etc
  - Software must have access to the **architectural-visible machine state** in order to intercept the data (operands) with the instruction that will execute on them.

- Compiler “knows” the exact location of every tensor on-chip

- In this way, the compiler is **orchestrating** the arrival of operands and the instructions which use them. The **producer-consumer stream programming model** allows a set of “streaming register files” to track the state of each tensor flowing through the chip.
Avoiding Complexity at the Chip Level

Conventional CPUs add features and add complexity:

- **Speculative execution** and out-of-order retirement
  - to improve instr level parallelism - **increases tail latency**

- **Implicit data flow** through cache memory hierarchies introduce complexity and non-determinism
  - (e.g. DRAM → L3 → L2 → L1 → GPRs) to hide DRAM access latency & pressure - **not energy or silicon efficient**

The TSP simplifies data flow through Stream Programming:

- A large, single-level scratchpad SRAM - **fixed, deterministic latency**

- **Explicitly** allocate tensors in space and time unlocking massive memory concurrency, and compute flexibility along multiple dimensions:

  [device, hemisphere, memory slice, bank, address offset]

Requires dynamic profiling to understand execution time and throughput characteristics of deep learning models

Predictable Performance at Scale
Functionally Sliced Microarchitecture

Reorganizing the multicore mesh
Functionally Sliced Microarchitecture

Reorganizing the multicore mesh
Functionally Sliced Microarchitecture

Reorganizing the multicore mesh

Reorganize a conventional manycore 2D mesh

MEM: on-chip SRAM
VXM: vector unit
MXM: matrix unit
SXM: data reshapess

Tensor operands and results flow on “streams” horizontally

Instructions flow vertically executed in a SIMD manner

© 2022 Groq, Inc. | Software-defined Tensor Streaming Multiprocessor

© 2022 Groq, Inc. | Software-defined Tensor Streaming Multiprocessor
MXM - Matrix Multiply Engines

<table>
<thead>
<tr>
<th>Numeric Mode</th>
<th>Max Size</th>
<th>Supported Density</th>
<th>Result Tensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>int8</td>
<td>[1, 320] x [320, 320]</td>
<td>Two per MXM</td>
<td>int32</td>
</tr>
<tr>
<td>float16</td>
<td>[1, 320] x [160, 320]</td>
<td>One per MXM</td>
<td>float32</td>
</tr>
</tbody>
</table>

320B x 320B dot product
Loads 320B x16 in 20 cycles
20 cycle execution
Fully pipelined, N

Int8 & float16
Full precision expansion
32-bit accumulate

Used Independently or together
## VXM and Complex, Customized Functions

<table>
<thead>
<tr>
<th>MXM</th>
<th>SXM</th>
<th>MEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MatMul</td>
<td>Dist</td>
<td>VXM</td>
</tr>
<tr>
<td>MEM</td>
<td>SXM</td>
<td>MXM</td>
</tr>
</tbody>
</table>

**Dataflow begins with memory Read onto Stream Tensor**

Many concurrent streams are supported in programming model.

VXM provides a flexible and programmable fabric for Compute.

Compute occurs on data locality of passing Stream Tensor.

MEM bandwidth supports high concurrency.
# SXM - Switch eXecution Module

<table>
<thead>
<tr>
<th>MXM</th>
<th>SXM</th>
<th>MEM</th>
<th>VXM</th>
<th>MEM</th>
<th>SXM</th>
<th>MXM</th>
</tr>
</thead>
</table>

- **Swiss army knife for data manipulation & Intra-vector byte operations**
- **Distributor**: 4 per hemisphere perform unto mapping of input + mask to output stream within a 16 byte superlane
- **Transposer**: 2 per hemisphere perform intra-superlane transpose over 16 lanes for 20 superlanes
- **Permuter/Shift**: arbitrary mapping of input + mask, shuffling between 320B vector elements - used for data transforms like pads/reshapes
- **Shift, Rotate, Distribute, Permute, Transpose, Transport to SuperLanes**
Streaming Registers

Contrasts to CPU / Register file
- Memory semantics have an address and a direction of data flow

Streaming registers are like a tensor assembly line flowing eastward and westward.

Each stream holds a vector of **320 bytes** (i.e. each element in the vector is 1-byte of data)
- **Multiple streams** are used to represent larger multi-byte data types (eg. FP32).
Rank-2 Tensor Dataflow

Rank-2 tensors, with inner dimension of 320 elements,
Rank-2 Tensor Dataflow

Rank-2 tensors, with inner dimension of 320 elements,
Streaming Registers

Larger data types are formed by “interleaving” 320-element vectors on multiple streams

fp32 datatype
An ISA That Empowers Software

Explicit time and space of instruction execution exposed by ISA to the compiler

Each Functional Unit (FU) type provides its own low-level instruction set

Number of FUs of each type and relative positions on chip are exposed to software

Compiler can choose to leverage multiple FUs for more concurrency or more pipelining

<table>
<thead>
<tr>
<th>Function</th>
<th>Instruction Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICU</td>
<td>ICU</td>
</tr>
<tr>
<td></td>
<td>ICU</td>
</tr>
<tr>
<td>MEM</td>
<td>MEM</td>
</tr>
<tr>
<td>VXM</td>
<td>VXM</td>
</tr>
<tr>
<td>MXM</td>
<td>MXM</td>
</tr>
<tr>
<td>SXM</td>
<td>SXM</td>
</tr>
<tr>
<td>C2C</td>
<td>C2C</td>
</tr>
</tbody>
</table>

Instruction Set

- Low level
- 320-vector ops
- Explicit resource selection
- Explicit scheduling

Instruction Set

- Low level
- 320-vector ops
- Explicit resource selection
- Explicit scheduling
Power of Data Orchestration

Given to Groq Compiler
Compilation
Traditional HPC Compiler Flow

A: The following graphic shows how cuDNN relates to other software in the stack.

Figure 11. Software stack with cuDNN.

NVIDIA CUDNN DOCUMENTATION
https://docs.nvidia.com/deeplearning/cudnn/developer-guide/index.html
“The most dangerous phrase in the language is “We’ve always done it this way.”

REAR ADMIRAL GRACE HOPPER
Pioneering Computer Scientist
1906–1992
Software-defined hardware relies on several interfaces:

- Static-Dynamic interface - compile-time versus runtime
- Hardware-software interface - exposing the architectural visible-state

Kernel-less approach to HPC Compilation
GEMM - General Matrix Multiplication

- Vector-Matrix multiplication and Matrix-Matrix multiplication are the workhorse for many ML and HPC workloads
- On-chip memory bandwidth determines how quickly we can ramp up ALUs for vector and matrix operations
- Consistent performance across a range of tensor sizes - less “hardware fitting”
- State-of-the-art (SOTA) results across a range of models and applications
  - CNNs
  - RNNs, LSTMs
  - NLP, BERT
  - Dense linear algebra V*M and M*M
GroqFamily

GroqChip™
GroqCard™
GroqNode™
GroqRack™
Scale-out organization

- Topology objectives
  - Low network diameter
  - Direct network
  - Hierarchical packaging-aware topology
- System packaging hierarchy
- Chip-to-chip (C2C) links and flow control
Low-diameter Network

Minimize the number of hops in the network

- The total observed latency and variance increases with the number of hops in the network

- Dragonfly is a *hierarchical* topology minimizes the number of hops taken
  - Local group topology
  - All-to-all global topology

- Exploits packaging locality

- Local topology 2x speedup

- Scalable from 1 TSP to thousands

**GLOBAL topology**

**LOCAL topology**

TSPs within the node with all-to-all connections

\[
25 \text{GBps} \times 4 \text{lanes} \times 2 \text{ dir} \times 28 \text{ links} = 700 \text{ GB/s}
\]

\[
25 \text{GBps} \times 4 \text{lanes} \times 2 \text{ dir} \times 4 \times 36 \text{ links} = 3.6 \text{ TB/s}
\]
Chip-to-chip (C2C) links and flow control

- Simplified communication model provides **logically shared access to global SRAM** which is physically distributed among the TSPs
- C2C links directly connect TSPs
- Comparing conventional RDMA with Groq C2C communication
Cholesky Factorization

- Cholesky decomposition is an important technique for solving large-scale system of linear equations.
- Compute is $N^3/3$ for a SPD matrix of size $N$.
- Block-cyclic distributions of 320 rows across chips.
Multichip Compilation

Available soon in next Quarter

Inter Op Partitioning

- Model graph split into multiple graphs
- Intermediate activations become inputs to successive graphs
- *Reduces weights on chip*

Intra Op Partitioning

- Tile the face of the input
- Operate on a “fractional” portion of the input at any given time
- *Reduces peak activation / input size on the chip*
Multichip Inter Op Partitioning

Device 0

Device 1

Device 2

Device 3

partition 0

partition 1

partition 2

partition 3

C2C
Multichip Pipelining

- Invocation is composed of compute on each of the devices followed by C2C to next subgraph's device
  - Pipeline parallel execution

Latency = 4 x (Compute + IO time)

Throughput = Clock freq / (Compute + IO time)
Inter Op Partitioning Results Today

Linear Scaling Achieved

**Scaling BERT encoders**

Scaling to 6, 24, 48, and 96 BERT encoders on 1, 4, 8, and 16 TSPs respectively
Results
Natural Language Processing with BERT

Seq Len=128

- Spread the layers across TSPs in each node for multi-TSP model parallelism
- Mini-batch (data parallelism) across nodes for throughput
- Goal: Accelerate BERT minimizing latency and maximizing throughput
- Minimize latency variance to provide predictable throughput

**T4, Current SOTA (A100) [21] and this work BERT-base latency (128 sequence length).**

<table>
<thead>
<tr>
<th></th>
<th>T4 (µs)</th>
<th>Current SOTA A100 (µs)</th>
<th>This work (µs)</th>
<th>Speedup (This work vs SOTA)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average</strong></td>
<td>1330</td>
<td>630</td>
<td>128.9</td>
<td>4.8×</td>
</tr>
<tr>
<td><strong>95th Percentile</strong></td>
<td>1550</td>
<td>780</td>
<td>129.1</td>
<td>6×</td>
</tr>
<tr>
<td><strong>99th Percentile</strong></td>
<td>1570</td>
<td>790</td>
<td>129.5</td>
<td>6.1×</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chip</th>
<th>Die Area (mm²)</th>
<th>Tech Process (nm)</th>
<th>Transistor count (B)</th>
<th>TDP (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVIDIA T4</td>
<td>545</td>
<td>12</td>
<td>13.6</td>
<td>70</td>
</tr>
<tr>
<td>NVIDIA A100</td>
<td>826</td>
<td>7</td>
<td>54.2</td>
<td>400</td>
</tr>
<tr>
<td>Groq TSP</td>
<td>725</td>
<td>14</td>
<td>26.8</td>
<td>275</td>
</tr>
</tbody>
</table>
GROQ™ COMPILER

Achieving SOTA Results

LSTM-512 TOPs vs Batch Size

- Intel NX int8
- V100 int8
- T4 int8
- GroqChip int8

Nvidia results from publicly available data on nvidia.com
LSTM results from Intel sponsored paper, FPT 2020, available on intel.com
Groq Exponentially Supporting More Workloads

What we’ve enabled thus far...

Legend:
- **COMPUTER VISION**
- **TEXT-TO-SPEECH**
- **NATURAL LANGUAGE PROCESSING**
- **HIGH PERFORMANCE COMPUTING**

<table>
<thead>
<tr>
<th>Q2 2021</th>
<th>Q3 2021</th>
<th>Q4 2021</th>
<th>Q1 2022</th>
<th>Q2 2022</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toy RNN</td>
<td>Toy MLP</td>
<td>Toy LSTM</td>
<td>Q8BERT</td>
<td>LSTM set</td>
</tr>
<tr>
<td>FFT</td>
<td>DistilBERT</td>
<td>LSTM</td>
<td>Yolo</td>
<td>MobileBERT</td>
</tr>
<tr>
<td>Yolo</td>
<td>MobileNet</td>
<td>T5</td>
<td>WaveGlow</td>
<td>Random Forest</td>
</tr>
<tr>
<td>T5</td>
<td>SqueezeNet</td>
<td>DETR</td>
<td>WaveGlow</td>
<td>GNN</td>
</tr>
<tr>
<td>LSTM</td>
<td>Bregman</td>
<td>GPT-Neo</td>
<td>WebGPT</td>
<td>T5 encoder</td>
</tr>
<tr>
<td>RN-50/101</td>
<td>VoVNet</td>
<td>EfficientNet</td>
<td>MT5</td>
<td>splinter</td>
</tr>
<tr>
<td>Bregman</td>
<td>MobileNet</td>
<td>Electra</td>
<td>MobileBERT</td>
<td>realm</td>
</tr>
<tr>
<td>LSTM</td>
<td>T5</td>
<td>BERT-Large</td>
<td>T5 encoder</td>
<td>SVM</td>
</tr>
<tr>
<td>Yolo</td>
<td>WaveGlow</td>
<td>BERT-64</td>
<td>MT5</td>
<td>Bidir LSTM</td>
</tr>
<tr>
<td>LSTM</td>
<td>WaveGlow</td>
<td>EfficientDet</td>
<td>MT5 encoder</td>
<td>STAC</td>
</tr>
<tr>
<td>LSTM</td>
<td>small-XLNet</td>
<td>MobNet2</td>
<td>EffientDet</td>
<td>LSTM A/B/C</td>
</tr>
<tr>
<td>Yolo</td>
<td>Bregman</td>
<td>WaveGlow</td>
<td>WaveGlowsmall-XLNet</td>
<td>CLIP</td>
</tr>
<tr>
<td>LSTM</td>
<td>MobileNet</td>
<td>MobileNet</td>
<td>WaveGlow</td>
<td>WebGPT</td>
</tr>
</tbody>
</table>

Under planning, subject to change
Summary and Takeaways

- Low-latency and high-throughput are necessary...
- Delivering **predictable** and **repeatable** performance is critical for many user-facing applications
  - Batch-1 inference is important for responsiveness and delivering quality-of-service (QoS) that is impossible to do with more traditional microarchitectures using crossbars, cache hierarchies, etc.

- **Determinism** enables **software-defined hardware** and entails a design philosophy that spans both hardware and software
  - ISA is **not** about abstraction of hardware details, but about **exerting control of underlying hardware**
    - 144 independent instruction control units (ICUs) of the TSP
  - Expose the architecturally-visible state (GPRs, SRAM, instruction buffers, etc)
  - Software-based replay and exception handling

- Extending the single-chip TSP determinism to the multiprocessor using **software scheduled networking** to explicitly schedule tensors on the network links

- **Synchronous communication** model allows for large-scale machine learning
We’re Hiring!

Join us...