Cerebras Architecture Deep Dive:
First Look Inside the HW/SW Co-Design for Deep Learning

Cerebras Systems

Sean Lie
Co-founder & Chief Hardware Architect
Cerebras Systems

Building and deploying a new class of computer system
Designed for the purpose of accelerating AI and changing the future of AI work

Founded in 2016

400+ Engineers in 14 Countries

Engineering Offices
Silicon Valley | San Diego
Toronto | Bangalore

Customers
North America | Asia | Europe
Select Cerebras Customers

Customers: Large Enterprise, HPC; Military and IC

- GlaxoSmithKline, TotalEnergies, AstraZeneca, Bayer, Genentech, Tokyo Electron Devices...
- ANL, LLNL, NETL, PSC, NCSA, EPPC, Leibniz Supercomputing Centre...
- Security, e.g. DARPA, USAF, ARL
Exponential Growth of Neural Networks

Over 1000x increase
In just 2 years

Tomorrow, multi-trillion parameter models
The ML Demand Challenge

We need **order of magnitude** improvements in 3 dimension:
1. Core architecture
2. Scale-up
3. Scale-out

But is it possible?

Requires specialized co-designed architecture for neural networks
Core Architecture

Scale-up
Scale-out

Accelerating dynamic sparse compute
Core Design

Efficient small core design
- 228µm x 170µm core area
- TSMC N7

Balanced logic and memory
- 50:50 logic to SRAM area ratio
- 110,000 logic standard cells
- 48kB high density SRAM memory

Power efficient design point
- 1.1GHz clock frequency
- 30mW peak power
Traditional memory bandwidth is low
- Central shared memory is off-chip DRAM with ~100-cycle access
- Insignificant amount of local SRAM
- Requires high data reuse and caching to be efficient

Distributed memory has full bandwidth
- All memory is SRAM with single-cycle access
- All memory is fully distributed with cores
- Full datapath performance without caching
- Unstructured sparsity processing at full performance
- High capacity from wafer-scale integration
Efficient Local Memory

Full memory performance with efficient SRAM banking
- 48kB total memory per PE
- 8 banks, 6kB per bank, 32b wide each, single port
- Full datapath bandwidth: 2 full reads + 1 full write per cycle

Software-managed cache for ultra-low power
- 256B local cache for low power
- Used for high frequency access data such as accumulators

Distributed memory architecture gives WSE-2 unprecedented memory bandwidth
200x normalized memory BW vs. GPU

*GPU normalized comparison is WSE-2 memory BW (400TB/s) in equivalent area against Nvidia A100 (2TB/s)
Memory Performance at All BLAS Levels
Enabling fine-grained unstructured sparsity

Massive bandwidth jump!

<table>
<thead>
<tr>
<th>0.005 Byte/FLOP</th>
<th>1 Byte/FLOP</th>
<th>2 Byte/FLOP</th>
<th>3 Byte/FLOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix-Matrix GEMM</td>
<td>Matrix-Vector GEMV</td>
<td>Vector-Vector DOT</td>
<td>Vector-Scalar AXPY</td>
</tr>
<tr>
<td>$C \leftarrow \alpha AB + \beta C$</td>
<td>$y \leftarrow \alpha Ax + \beta y$</td>
<td>$\alpha \leftarrow (x, y)$</td>
<td>$y \leftarrow \alpha x + y$</td>
</tr>
<tr>
<td>$C$ += $\begin{bmatrix} A &amp; B \end{bmatrix}$</td>
<td>$y$ += $\begin{bmatrix} A &amp; x \end{bmatrix}$</td>
<td>$\alpha$ += $\begin{bmatrix} y &amp; x \end{bmatrix}$</td>
<td>$y$ += $\begin{bmatrix} \alpha &amp; x \end{bmatrix}$</td>
</tr>
</tbody>
</table>

Sparse GEMM is one AXPY per non-zero weight
Core Datapath

Programmable execution for changing NN architectures

• Flexible **general ops** for control processing
  • Ops: arithmetic, logical, load/store, compare, branch
• Independent instructions for each core
• 48kB memory for data and instructions
• 16 general purpose registers (GPRs)
• Compact 6-stage pipeline

High performance flexible tensor processing

• Optimized **tensor ops** for high perf data processing
• Fine-grained 64b datapath with 4x FP16 FMACs
• Tensors as first-class operands to each instruction
  • e.g. \( \text{fmac} [\text{fsum}] = [\text{fsum}], [\text{fwd_wgt}], \text{r_in} \)
    
    3D 3D 2D scalar
  • 44 data structure registers (DSRs) to describe tensor operands
    • Descriptor for up to 4D tensors, FIFOs, and fabric tensors
    • Specifies tensor address, length, dimensions, stride
Core Dataflow Scheduling

Native sparsity acceleration with dataflow scheduling

- Data and control transmitted on fabric
- Triggers lookup and execution of handler instructions
- Lookup based on input fabric color or control information
- Native unstructured sparsity harvesting by filtering out zeros

Micro-threading to drive higher utilization

- 8 simultaneous tensor operations supported in hardware
- Interleaving tensor iterations cycle-by-cycle
- Scheduling based on input/output tensor availability and priority

Fine-grained dynamic compute core enables unprecedented compute performance

10x sparse utilization vs. GPU

*GPU utilization comparison is estimate against 10x unstructured sparse compute on Nvidia A100
Core Architecture

Scale-up
Scale-out

Amplifying Moore’s Law
Cerebras Wafer-Scale Engine (WSE-2)

The Largest Chip in the World

850,000 cores optimized for sparse linear algebra
46,225 mm$^2$ silicon
2.6 trillion transistors
40 gigabytes of on-chip memory
20 PByte/s memory bandwidth
220 Pbit/s fabric bandwidth
7nm process technology

56x larger than largest GPU
From Small Core to Massive Wafer

- Core: 228um x 170um
- Die: 17mm x 30mm
  - 66 x 154 Cores
  - 10,156 Cores
- WSE-2: 215mm x 215mm
  - 12 x 7 Die
  - 84 Die
High Bandwidth Low Latency Fabric

Efficient high performance
- 2D mesh topology with low overheads
- 5-port router to 4 neighbors and core
- 32b/cycle bidirectional data transfer
  - Individual packages are 32b
  - Payload carries data (16b) and index (16b)
- Single cycle latency between cores
  - Flow controlled with low buffering
- 24 configurable static routing (colors)
  - Each color has dedicated buffering, is non-blocking
  - All colors are time-multiplexed onto same physical link
- Hardware broadcast/multicast
Uniform Fabric Across Entire Wafer

**Designed to scale beyond individual die**
- Bridge <1mm across scribe lines between die
- Source synchronous parallel interface
- Redundancy with training and auto-correction state machine

**Uniform bandwidth across entire wafer**
- The entire wafer is a single chip all with *on-chip* bandwidth
- Full bandwidth within die and between die
- Wafer integration enables ultra short inter-chip links
## Unprecedented Fabric Performance and Power

<table>
<thead>
<tr>
<th></th>
<th>Area</th>
<th>Bandwidth</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mm²</td>
<td>TB/s</td>
<td>GB/s/mm²</td>
</tr>
<tr>
<td>GPU Estimate</td>
<td>826</td>
<td>0.6</td>
<td>0.7</td>
</tr>
<tr>
<td>WSE-2 Sub-fabric</td>
<td>826</td>
<td>4.3</td>
<td>5.2</td>
</tr>
<tr>
<td>Ratio</td>
<td>7x</td>
<td>7x</td>
<td>66x</td>
</tr>
</tbody>
</table>

Wafer-scale fabric architecture gives WSE-2 unprecedented fabric bandwidth and power

7x normalized fabric bandwidth vs. GPU

*GPU estimate uses 7nm PCIe 5.0 serdes with Nvidia A100 NVLink bandwidth
*WSE-2 sub-fabric uses subset of wafer area equivalent to Nvidia A100
Cluster-scale compute in a single chip

- Train the largest neural networks (e.g. GPT-3)
- On a single chip without partitioning

Built for extreme-scale neural networks

- *Weight Streaming* execution decouples memory from compute
- Weights stored externally off-wafer in MemoryX
- Weights streamed onto wafer to compute layer
- Execute one layer at a time
- Gradients streamed out of wafer
- Weight update occurs in MemoryX

All Model Sizes on a Single Chip
Mapping Neural Networks to WSE-2

The full wafer is the MatMul array for even the largest matrices

• Activations from batch are spread across the entire wafer
• Local memory in each core stores a chunk of the tensor
High bandwidth on-wafer fabric enables efficient global communication of data and control

- Each fabric packet carries data and/or control
  - Data: weight (16b) + index (16b)
  - Control: command
- Packets are broadcast to all cores to trigger work
  - Data packets trigger FMA computation
  - Commands trigger other computations and synchronization
Dataflow scheduling enables fully unstructured sparse MatMul with low overhead

- Executed as a series of AXPY operations per row
- Row of non-zero weights broadcast over columns of cores
- Each individual weight triggers FMACs
- No compute for zero weights, not streamed in at all
- No memory used for weights, not even stored temporarily
GEMM with Sparse Input

Multiply
- Each weight triggers FMACs with local column of activations
Partial Sum Reduce

- `PSUM/FSUM` commands broadcast to start partial sum reduction
- Partial sums reduced over rows of cores in a ring
- `FSUM` command directs final sums to the correct column
- Reductions are overlapped with the next set of weights

```
Weights[i][k:k+n]
act[j:j+m][k:k+n]
SRAM
```

```
PSUM PSUM PSUM FSUM PSUM PSUM PSUM PSUM
```
All Model Sizes at Extreme Performance on a Single Chip

Architecture enables efficient wafer-scale computation

- Full bandwidth memory to datapath
  - AXPY operations for sparsity acceleration
- Dataflow scheduling
  - Unstructured sparsity acceleration by skipping zero weights
  - Massive model support by never storing weight matrix
- High bandwidth wafer-scale fabric
  - Global weight broadcast and reduction across wafer

No matrix blocking or partitioning required

Up to 100k x 100k MatMul
Run models of all sizes in a single device with

75 PFLOPS FP16 Sparse
7.5 PFLOPS FP16 Dense

*Sparse performance with 10x sparsity acceleration
Core Architecture
☑️ Scale-up

Scale-out

Inherently Scalable Clustering
Challenges to Scaling on GPU Clusters

Hybrid parallelism on traditional devices

Data Parallel
- Multiple samples at a time
- Parameter memory limits

Pipelined Model Parallel
- Multiple layers at a time
- Communication overhead
- $N^2$ activation memory

Tensor Model Parallel
- Multiple splits at a time
- Communication overhead
- Complex partitioning

Distribution complexity scales dramatically with cluster size
Complexity in Practice on GPU Clusters

Traditional scaling complexity

- Extreme-scale models on GPU requires all forms of parallelism simultaneously
- Tensor model parallel limited to within single server
- Pipelined model parallel makes up most of parallelism for largest model, but it’s the most complex
- Resulting in complexity and often poor scaling

Cerebras scaling simplicity

- Execution on single device without partition
- Data parallel only scaling to multiple devices

*Parallelism breakdown derived from the model publication: NV Megatron V1, MS T-NLG, Eleuther GPT-NeoX, OpenAI GPT-3, PanGu-alpha, NV Megatron V2, MS MT-NLG
Near-Linear Data Parallel Only Scaling

**Specialized interconnect for scale-out**
- Data parallel distribution through SwarmX interconnect
- Weights are **broadcast** to all CS-2s
- Gradients are **reduced** on way back

**Multi-system scaling with the same execution as single system**
- Same system architecture
- Same network execution flow
- Same software user interface
Meeting the Grand Challenge

- Process
- Core Architecture
- Scale-out
The Grand ML Demand Challenge

Is it possible?
Enabling All to Train Largest Models Ever

Specialized architecture with **order of magnitude** improvements in all 3 dimensions:

1. Core architecture
2. Scale-up
3. Scale-out

There’s no end in sight

- Models continue to grow exponentially
- Few companies have access to largest models today
- Cerebras architecture makes running largest models fast and easy
  - Largest models on a single device
  - Data parallel only scale-out
  - Native unstructured sparsity acceleration

Making the largest models available to everyone
Thank you