Analog Computation in Flash Memory for Datacenter-scale AI Inference in a Small Chip

Dave Fick, CTO/Founder
Mike Henry, CEO/ Founder
About Mythic

- Focused on high-performance Edge AI
  - Full stack co-design: device physics to new algorithms and applications

- Founded in 2012 by Mike Henry and Dave Fick
  - While working with the Michigan Integrated Circuits Lab (MICL)

- Raised $55M from top-tier investors: DFJ, SoftBank, Lux, DCVC
  - Offices in Redwood City & Austin
  - 60+ employees
DNNs are Largely Multiply-Accumulate

Primary DNN Calculation is  \( \text{Input Vector} \times \text{Weight Matrix} = \text{Output Vector} \)

Input Data  \[
\begin{bmatrix}
X_0 & X_1 & \cdots & X_N
\end{bmatrix}
\]

Neuron Weights  \[
\begin{bmatrix}
A_0 & B_0 & C_0 \\
A_1 & B_1 & C_1 \\
\vdots & \vdots & \vdots \\
A_N & B_N & C_N
\end{bmatrix}
\]

Outputs Equations  \[
\begin{align*}
Y_A &= X_0A_0 + X_1A_1 + X_2A_2 \\
Y_B &= X_0B_0 + X_1B_1 + X_2B_2 \\
Y_C &= X_0C_0 + X_1C_1 + X_2C_2
\end{align*}
\]

Key Operation: Multiply-Accumulate, or “MAC”

Figure of Merit: How many picojoules to execute a MAC?
Memory Access Includes Weight Data and Intermediate Data

Input Data

\[ [X_0 \ X_1 \ \cdots \ X_N] \]

Neuron Weights

\[
\begin{bmatrix}
  A_0 & B_0 & C_0 \\
  A_1 & B_1 & C_1 \\
  \vdots & \vdots & \vdots \\
  A_N & B_N & C_N \\
\end{bmatrix}
\]

Outputs Equations

\[
\begin{align*}
  Y_A &= X_0A_0 + X_1A_1 + X_2A_2 \\
  Y_B &= X_0B_0 + X_1B_1 + X_2B_2 \\
  Y_C &= X_0C_0 + X_1C_1 + X_2C_2
\end{align*}
\]
Intermediate Data Accesses are Naturally Amortized

For a 1000 input, 1000 neuron matrix....

Intermediate data accesses are amortized 64-1024x since they are used in many MAC operations
Weight Data Accesses are Not Amortized

For a 1000 input, 1000 neuron matrix….

\[
\begin{bmatrix}
X_0 & X_1 & \cdots & X_N
\end{bmatrix} \times
\begin{bmatrix}
A_0 & B_0 & C_0 \\
A_1 & B_1 & C_1 \\
\vdots & \vdots & \vdots \\
A_N & B_N & C_N
\end{bmatrix} =
\begin{bmatrix}
Y_A = X_0A_0 + X_1A_1 + X_2A_2 \\
Y_B = X_0B_0 + X_1B_1 + X_2B_2 \\
Y_C = X_0C_0 + X_1C_1 + X_2C_2
\end{bmatrix}
\]

Weight data could need to be stored in DRAM, and it does not have the same amortization as the intermediate data.
DNN Processing is All About Weight Memory

- 10+M parameters to store
- 20+B memory accesses
- How do we achieve…
  - High Energy Efficiency
  - High Performance
  - “Edge” Power Budget (e.g., 5W)

<table>
<thead>
<tr>
<th>Network</th>
<th>Weights</th>
<th>MACs</th>
<th>…@ 30 FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet¹</td>
<td>61 M</td>
<td>725 M</td>
<td>22 B</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>11 M</td>
<td>1.8 B</td>
<td>54 B</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>23 M</td>
<td>3.5 B</td>
<td>105 B</td>
</tr>
<tr>
<td>VGG-19¹</td>
<td>144 M</td>
<td>22 B</td>
<td>660 B</td>
</tr>
<tr>
<td>OpenPose²</td>
<td>46 M</td>
<td>180 B</td>
<td>5400 B</td>
</tr>
</tbody>
</table>

Very hard to fit this in an Edge solution

¹: 224x224 resolution
²: 656x368 resolution
Common Techniques for Reducing Weight Energy Consumption

Weight Re-use

- **Focus on CNN**
  - Re-use weights for multiple windows
  - Can build specialized structures
  - *Not all problems map to CNN well*

- **Focus on Large Batch**
  - Re-use weights for multiple inputs
  - *Edge is often batch=1*
  - *Increases latency*

Weight Reduction

- **Shrink the Model**
  - Use a smaller network that can fit on-chip (e.g., SqueezeNet)
  - *Possibly reduced capability*

- **Compress the Model**
  - Use sparsity to eliminate up to 99% of the parameters
  - Use literal compression
  - *Possibly reduced capability*

- **Reduce Weight Precision**
  - 32b Floating Point => 2-8b Integer
  - *Possibly reduced capability*
Key Question: Use DRAM or Not?

Benefits of DRAM
- Can fit arbitrarily large models
- Not as much SRAM needed on chip

Drawbacks of DRAM
- Huge energy cost for reading weights
- Limited bandwidth getting to weight data
- Variable energy efficiency & performance depending on application
## Common NN Accelerator Design Points

<table>
<thead>
<tr>
<th></th>
<th>Enterprise With DRAM</th>
<th>Enterprise No-DRAM</th>
<th>Edge With DRAM</th>
<th>Edge No-DRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SRAM</strong></td>
<td>&lt;50 MB</td>
<td>100+ MB</td>
<td>&lt; 5 MB</td>
<td>&lt; 5 MB</td>
</tr>
<tr>
<td><strong>DRAM</strong></td>
<td>8+ GB</td>
<td>-</td>
<td>4-8 GB</td>
<td>-</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>70+ W</td>
<td>70+ W</td>
<td>3-5 W</td>
<td>1-3 W</td>
</tr>
<tr>
<td><strong>Sparsity</strong></td>
<td>Light</td>
<td>Light</td>
<td>Moderate</td>
<td>Heavy</td>
</tr>
<tr>
<td><strong>Precision</strong></td>
<td>32f / 16f / 8i</td>
<td>32f / 16f / 8i</td>
<td>8i</td>
<td>1-8i</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td>Great</td>
<td>Great</td>
<td>Moderate</td>
<td>Poor</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>High</td>
<td>High</td>
<td>Very Low</td>
<td>Very Low</td>
</tr>
</tbody>
</table>
## Mythic is Fundamentally Different

<table>
<thead>
<tr>
<th></th>
<th>Enterprise With DRAM</th>
<th>Enterprise No-DRAM</th>
<th>Edge With DRAM</th>
<th>Edge No-DRAM</th>
<th>Mythic NVM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SRAM</strong></td>
<td>&lt;50 MB</td>
<td>100+ MB</td>
<td>&lt; 5 MB</td>
<td>&lt; 5 MB</td>
<td>&lt; 5 MB</td>
</tr>
<tr>
<td><strong>DRAM</strong></td>
<td>8+ GB</td>
<td>-</td>
<td>4-8 GB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>70+ W</td>
<td>70+ W</td>
<td>3-5 W</td>
<td>1-3 W</td>
<td>1-5 W</td>
</tr>
<tr>
<td><strong>Sparsity</strong></td>
<td>Light</td>
<td>Light</td>
<td>Moderate</td>
<td>Heavy</td>
<td>None</td>
</tr>
<tr>
<td><strong>Precision</strong></td>
<td>32f / 16f / 8i</td>
<td>32f / 16f / 8i</td>
<td>8i</td>
<td>1-8i</td>
<td>1-8i</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td>Great</td>
<td>Great</td>
<td>Moderate</td>
<td>Poor</td>
<td>Great</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>High</td>
<td>High</td>
<td>Very Low</td>
<td>Very Low</td>
<td>High</td>
</tr>
</tbody>
</table>
## Mythic is Fundamentally Different

<table>
<thead>
<tr>
<th></th>
<th>Enterprise With DRAM</th>
<th>Enterprise No-DRAM</th>
<th>Edge With DRAM</th>
<th>Edge No-DRAM</th>
<th>Mythic NVM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SRAM</strong></td>
<td>&lt;50 MB</td>
<td>100+ MB</td>
<td>&lt; 5 MB</td>
<td>&lt; 5 MB</td>
<td>&lt; 5 MB</td>
</tr>
<tr>
<td><strong>DRAM</strong></td>
<td>8+ GB</td>
<td>-</td>
<td>4-8 GB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>70+ W</td>
<td>70+ W</td>
<td>3-5 W</td>
<td>1-3 W</td>
<td>1-5 W</td>
</tr>
<tr>
<td><strong>Sparsity</strong></td>
<td>Light</td>
<td>Light</td>
<td>Moderate</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><strong>Precision</strong></td>
<td>32f / 16f / 8i</td>
<td>32f / 16f / 8i</td>
<td>8i</td>
<td>1-8i</td>
<td>1-8i</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td>Great</td>
<td>Great</td>
<td>Moderate</td>
<td>Poor</td>
<td>Great</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>High</td>
<td>High</td>
<td>Very Low</td>
<td>Very Low</td>
<td>High</td>
</tr>
</tbody>
</table>

Also, Mythic does this in a 40nm process, compared to 7/10/16nm.
Mythic’s New Architecture Merges Enterprise and Edge

- Mythic introduces the **Matrix Multiplying Memory**
  - Never read weights

- This effectively makes weight memory access **energy-free** (only pay for MAC)

- And eliminates the need for...
  - Batch > 1
  - CNN Focus
  - Sparsity or Compression
  - Nerfed DNN Models

*Made possible with Mixed-Signal Computing on embedded flash*
# Revisiting Matrix Multiply

Primary DNN Calculation is \( \text{Input Vector} \times \text{Weight Matrix} = \text{Output Vector} \)

<table>
<thead>
<tr>
<th>Input Data</th>
<th>Neuron Weights</th>
<th>Outputs Equations</th>
</tr>
</thead>
</table>
| \([X_0, X_1, \ldots, X_N]\) | \[
\begin{bmatrix}
A_0 & B_0 & C_0 \\
A_1 & B_1 & C_1 \\
\vdots & \vdots & \vdots \\
A_N & B_N & C_N \\
\end{bmatrix}
\] | \[
\begin{bmatrix}
Y_A = X_0A_0 + X_1A_1 + X_2A_2 \\
Y_B = X_0B_0 + X_1B_1 + X_2B_2 \\
Y_C = X_0C_0 + X_1C_1 + X_2C_2 \\
\end{bmatrix}
\] |

Flash Transistors
Flash transistors can be modeled as variable resistors representing the weight.

The $V=IR$ current equation will achieve the math we need:
- Inputs ($X$) = DAC
- Weights ($R$) = Flash transistors
- Outputs ($Y$) = ADC Outputs

The ADCs convert current to digital codes, and provide the non-linearity needed for DNN.
DACs & ADCs Give Us a Flexible Architecture

We have a **digital** top-level architecture:

- Interconnect
- Intermediate data storage
- Programmability (XLA/ONNX => Mythic IPU)
To Simplify we use Digital Approximation

To improve time-to-market, we have left the Input DAC as a future endeavor

We achieve the same result through digital approximation

Silver lining: we have future improvements available
Mythic Mixed-Signal Computing

Single Tile

- Input Data
- Weight Storage + Analog Matrix Multiplier
- Analog To Digital

Components:
- SRAM
- RISC-V
- SIMD
- Router

Made possible with Mixed-Signal Computing on embedded flash
Mythic Mixed-Signal Computing

**Single Tile**
- Input Data
- Weight Storage + Analog Matrix Multiplier
- Digital to Analog
- Analog To Digital
- SRAM
- RISC-V
- SIMD
- Router

**Tiles Connected in a Grid**

**Example DNN Mapping (Post-Silicon)**
- Scene Segmentation
- Camera Enhancement
- Object Tracking

**Expandable Grid of Tiles**
System Overview

Initial Product
- 50M weight capacity
- PCIe 2.1 x4
- Basic Control Processor

Envisioned Customizations (Gen 1)
- Up to 250M weight capacity
- PCIe 2.1 x16
- USB 3.0/2.0
- Direct Audio/Video Interfaces
- Enhanced Control Processor (e.g., ARM)

Intelligence Processing Unit (IPU)
Mythic is a PCIe Accelerator

Inference Model
(specified via TensorFlow, Caffe2, or others)

Operating System
Applications Interfaces
Mythic IPU Driver
We Also Support Multiple IPUs

Mythic IPUs

Inference Model
(specified via TensorFlow, Caffe2, or others)

Data

PCIe

Inference Results

Host SoC

Operating System
Applications Interfaces
Mythic IPU Driver

DRAM
We Account For All Energy Consumed

- Numbers are for a typical application, e.g. ResNet-50
  - Batch size = 1
  - We are relatively application-agnostic (especially compared to DRAM-based systems)
- 8b analog compute accounts for about half of our energy
  - We can also run lower precision
  - Control, storage, and PCIe accounts for the other half

Energy (pJ/MAC)
Total = 0.5
Example Application: ResNet-50

Running at 224x224 resolution. Mythic estimated, GPU/SoC measured

GPU Performance in an Edge Form Factor!
Example Application: OpenPose

Running at 656x368 resolution. Mythic estimated, GPU/SoC measured
Timeline

- Alpha release of software tools and profiler: Late 2018
- External samples: Mid 2019
  - PCIe Dev Boards (1, 4 IPUs)
- Volume shipments: Late 2019
  - BGA Chips
  - PCIe Cards (1, 4, 16 IPUs)
Mythic IPU Overview

- **Low Latency**
  - Runs batch size = 1
  - E.g., single frame delay

- **High Performance**
  - 10’s of TMAC/s

- **High Efficiency**
  - 0.5 pJ/MAC aka 500mW / TMAC

- **Hyper-Scalable**
  - Ultra low power to high performance

- **Easy to use**
  - Topology agnostic (CNN/DNN/RNN)
  - TensorFlow/Caffe2/etc supported

Made possible with Mixed-Signal Computing on embedded flash
Thank you for listening!

Questions?