MYTHIC

Analog Compute-in-Memory at Mythic

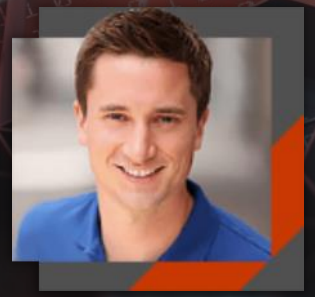
Dave Fick, CTO / Founder

dave.fick@mythic-ai.com

MYTHIC

Mythic, Inc.

- AI startup founded in 2012 focused on power-efficient AI inference processing
- Unique analog compute-in-memory (CIM) architecture using flash memory
- 120+ employees in Austin, TX and Redwood City, CA
- $86M in venture funding:
  - Softbank, DFJ, Lux, Valor Equity Partners, Lockheed Martin, Micron, and others



Mike Henry-
Redwood City, CA

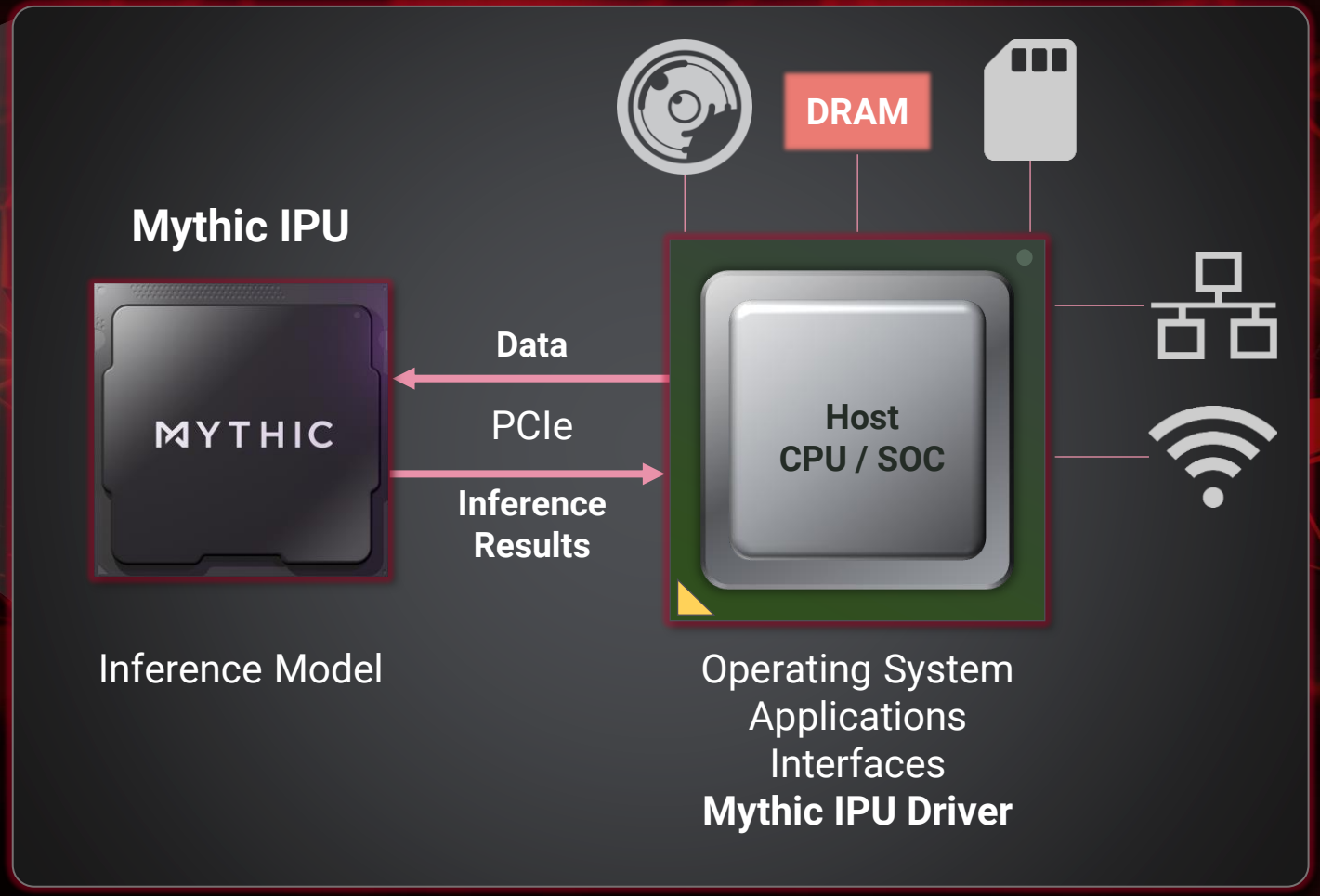
FOUNDER, CEO

Dave Fick-
Austin, TX

FOUNDER, CTO

Mythic IPU is a PCI Express Accelerator

High performance for constrained environments



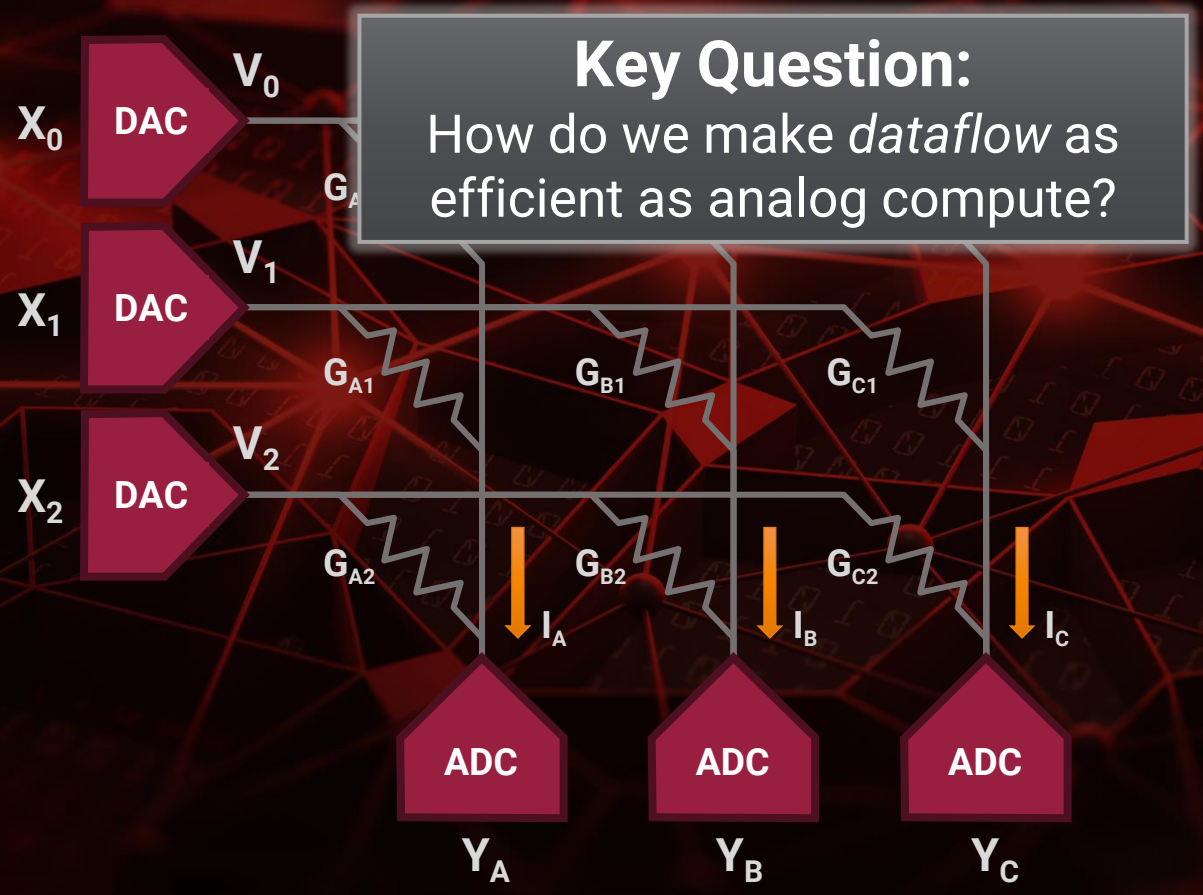
Analog Compute Gives Us Efficient Matrix Multiplication

Eliminating weight movement and using analog computation provides >10x **overall** efficiency improvement vs digital systems

Flash transistors can be modeled as **variable conductances** representing the weights

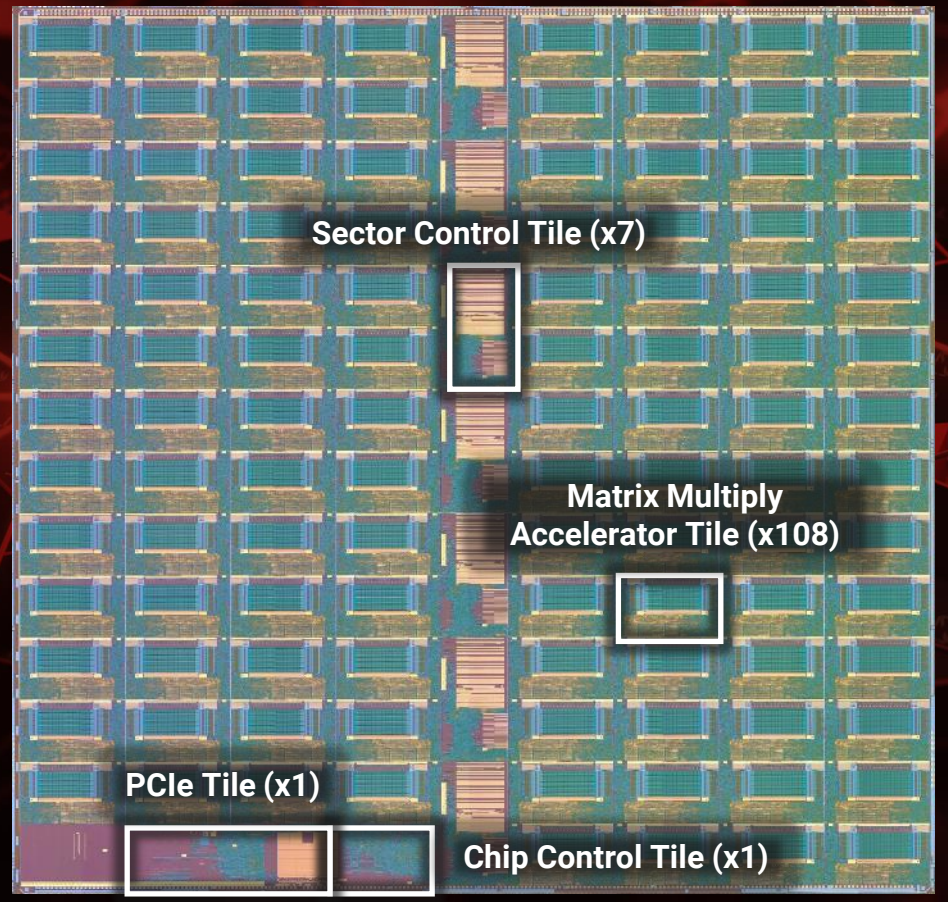
The I=G*V current equation will achieve the math we need:

- Inputs (X) = DAC Inputs
- Weights (G) = Flash transistors
- Outputs (Y) = ADC Outputs

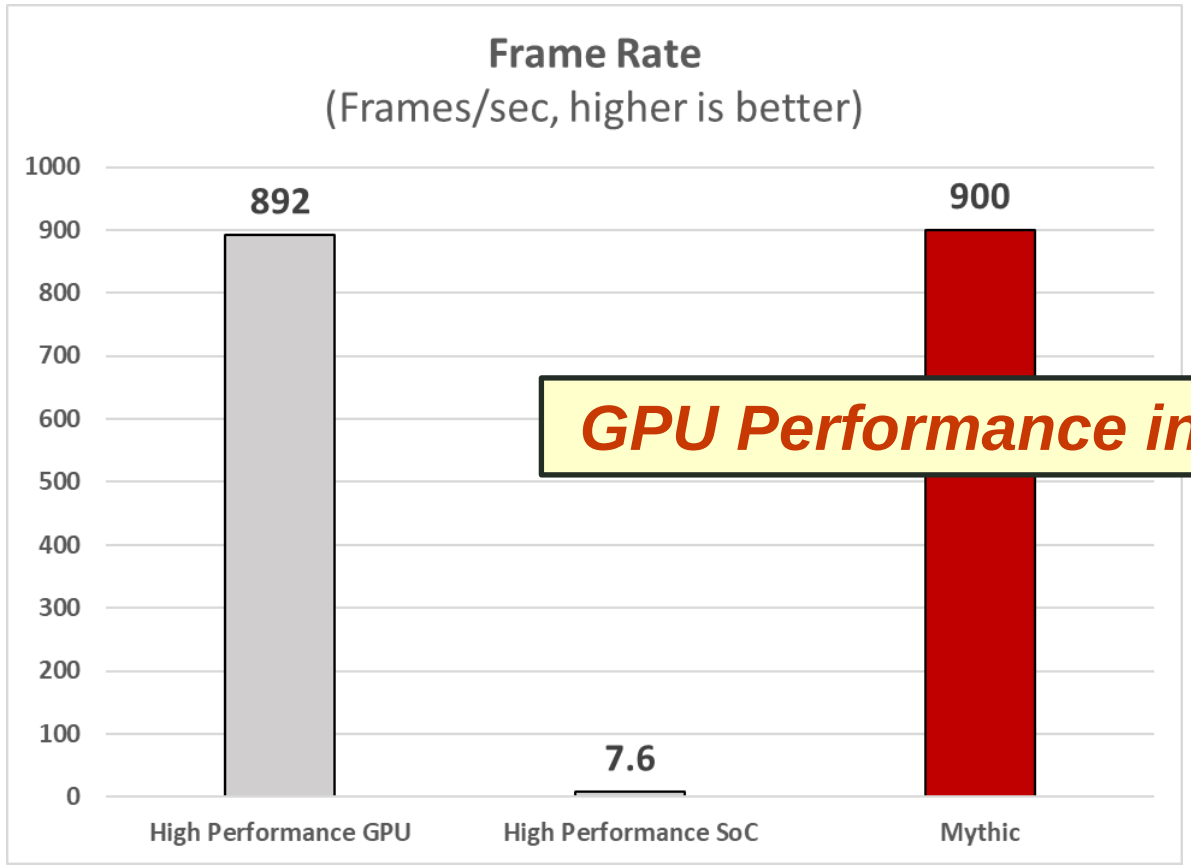


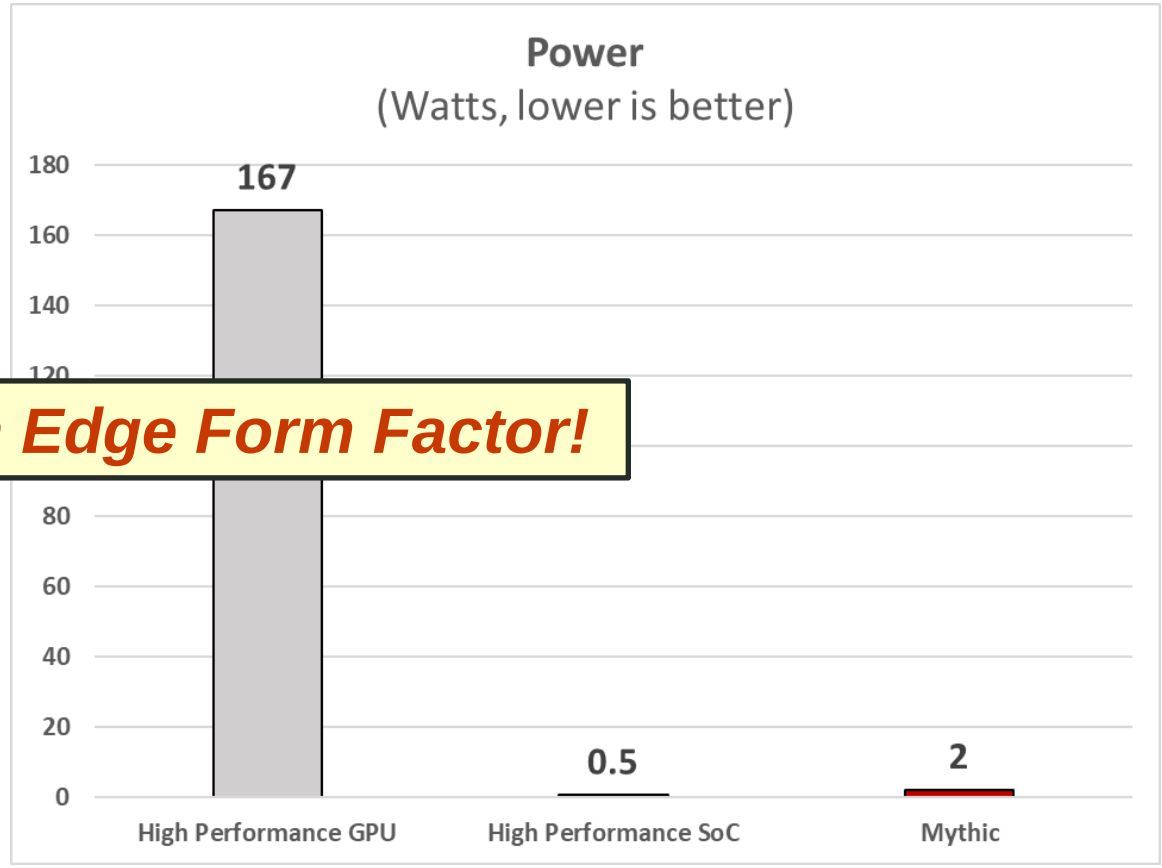
Mythic IPU Overview

- ☑ Low Latency
  - Runs batch size = 1, single frame latency
- ☑ 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



Example Application: ResNet-50

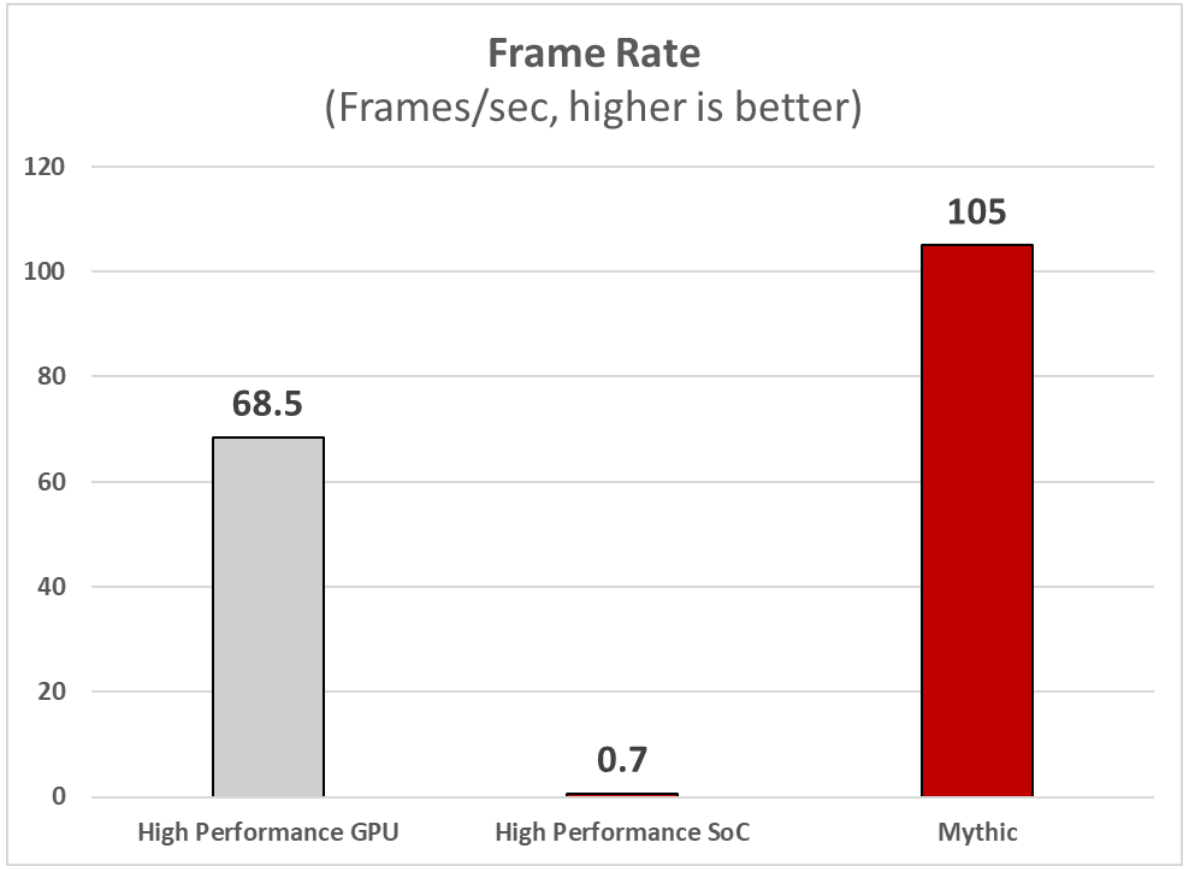


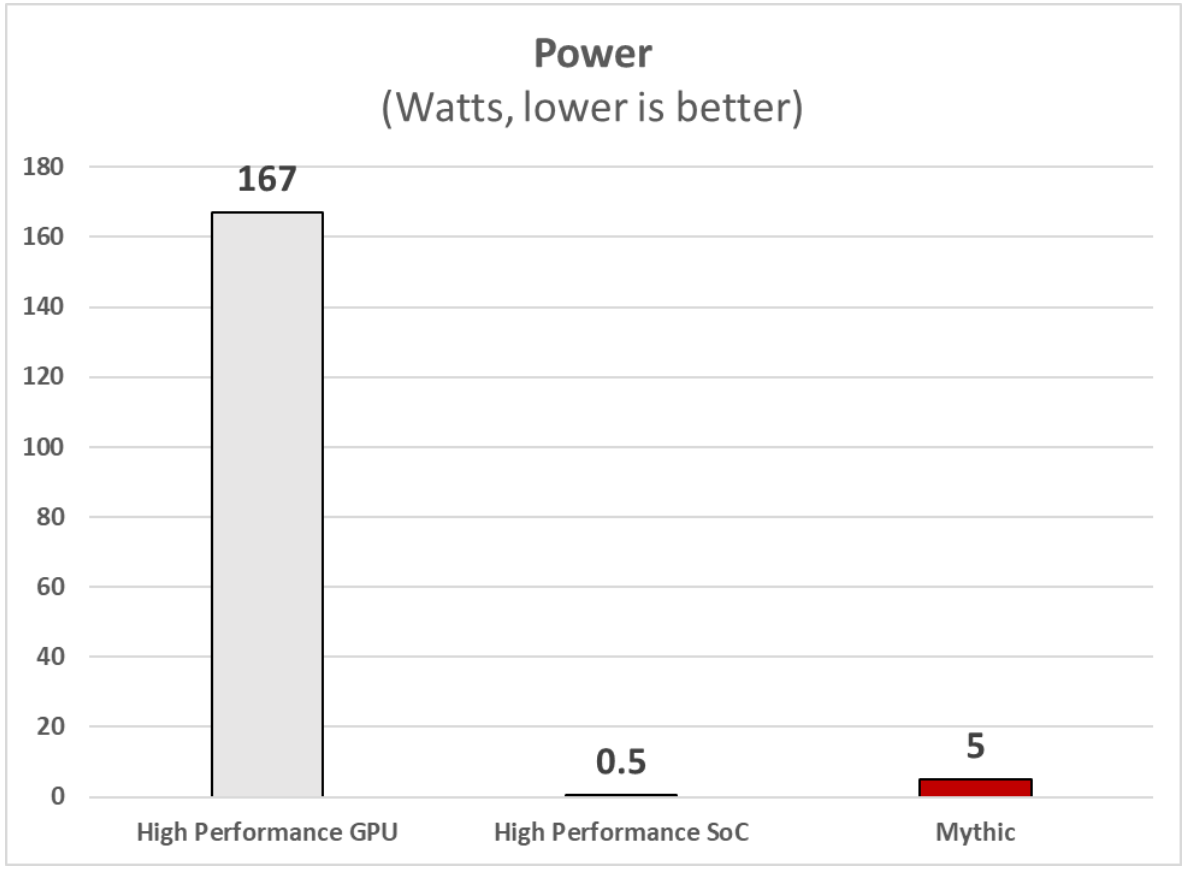


GPU Performance in an Edge Form Factor!

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

Example Application: OpenPose





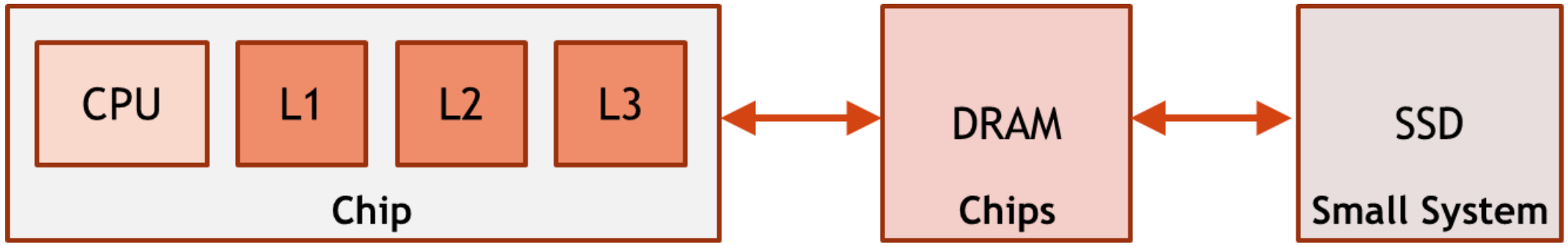
Running at 656x368 resolution. Mythic estimated, GPU/SoC measured

Outline

- Compute-in-Memory Overview
- Analog Compute Overview
- Mythic GPS
- Mythic Neural Networks

When Should I use Compute-in-Memory?

Compute-In-Memory



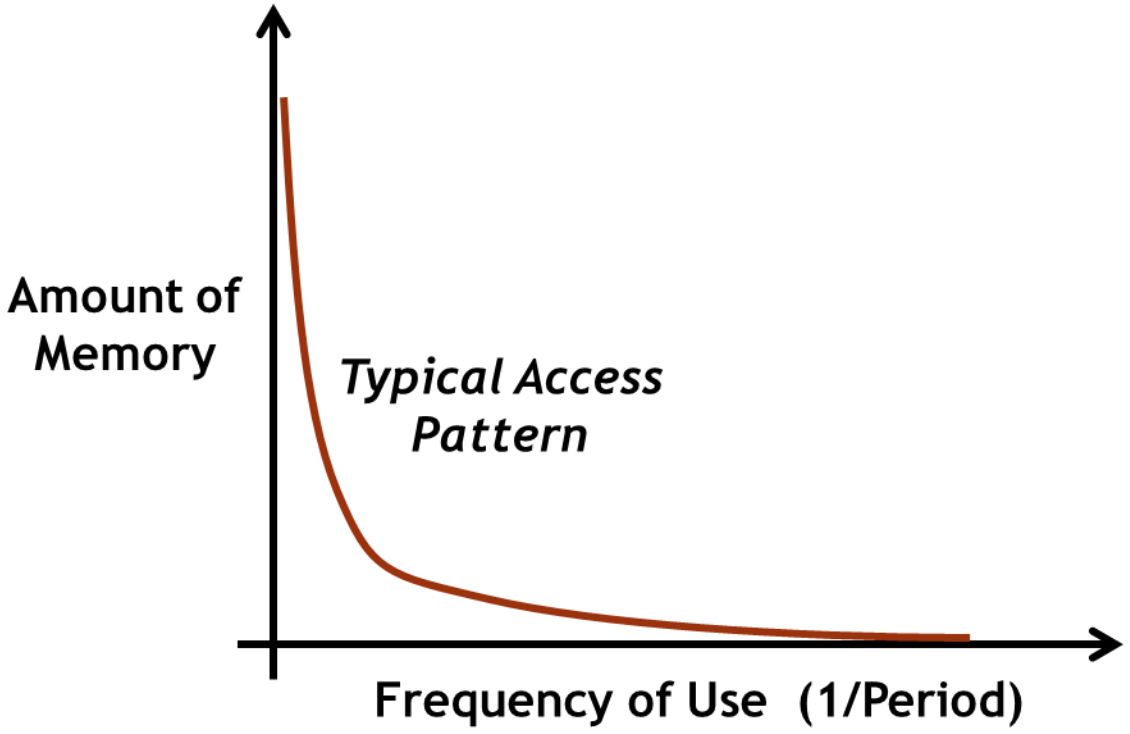
- "Compute-in-Memory" is **relative!**
- Typically the CPU works on the L1 Cache

- CIM could mean…
  - Compute at L2/L3
  - Compute in DRAM chips
  - Compute in the SSD
  - Compute in an accelerator that contains memory

Memory Systems are Built for Data Access Patterns

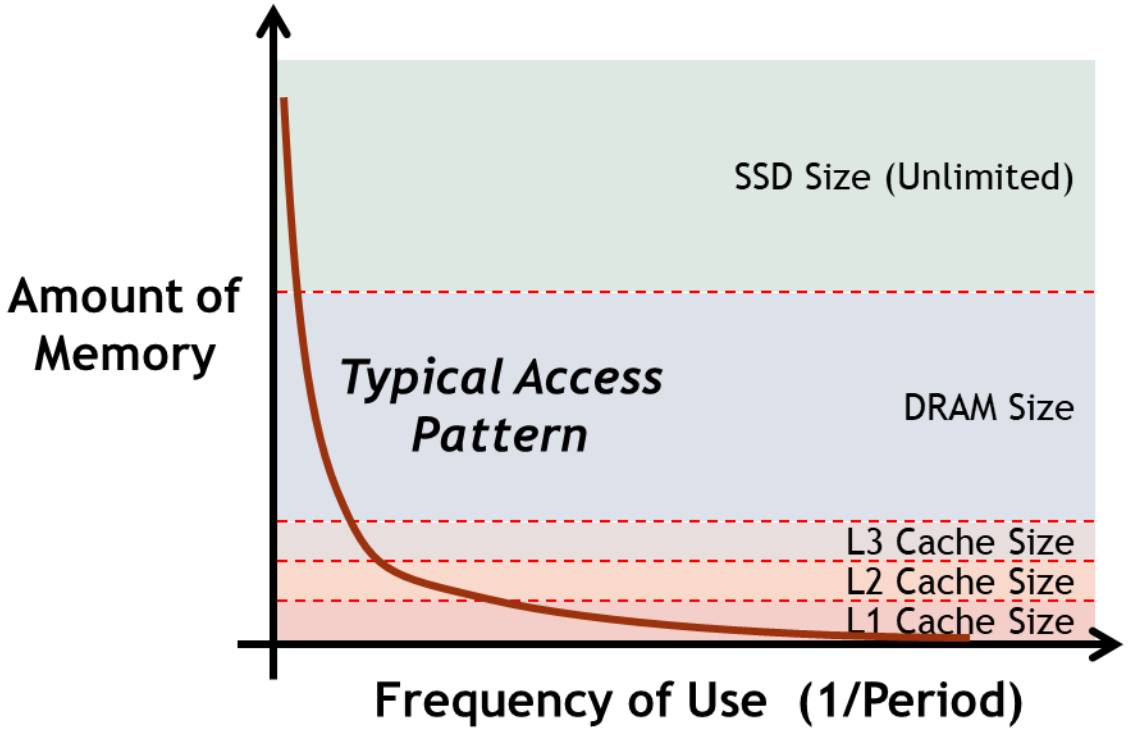
- The existing memory structure has built-in assumptions:
  - Temporal Locality
    - If at one point a particular memory location is referenced, then it is likely that the same location will be referenced again in the near future.
  - Spatial Locality
    - If a particular storage location is referenced at a particular time, then it is likely that nearby memory locations will be referenced in the near future.
  - Probability, not Certainty
    - We do not know exactly what data will be needed next

Data Access Patterns: Analysis Via “Working Set”



Working set: the amount of memory needed by the application over a period of time

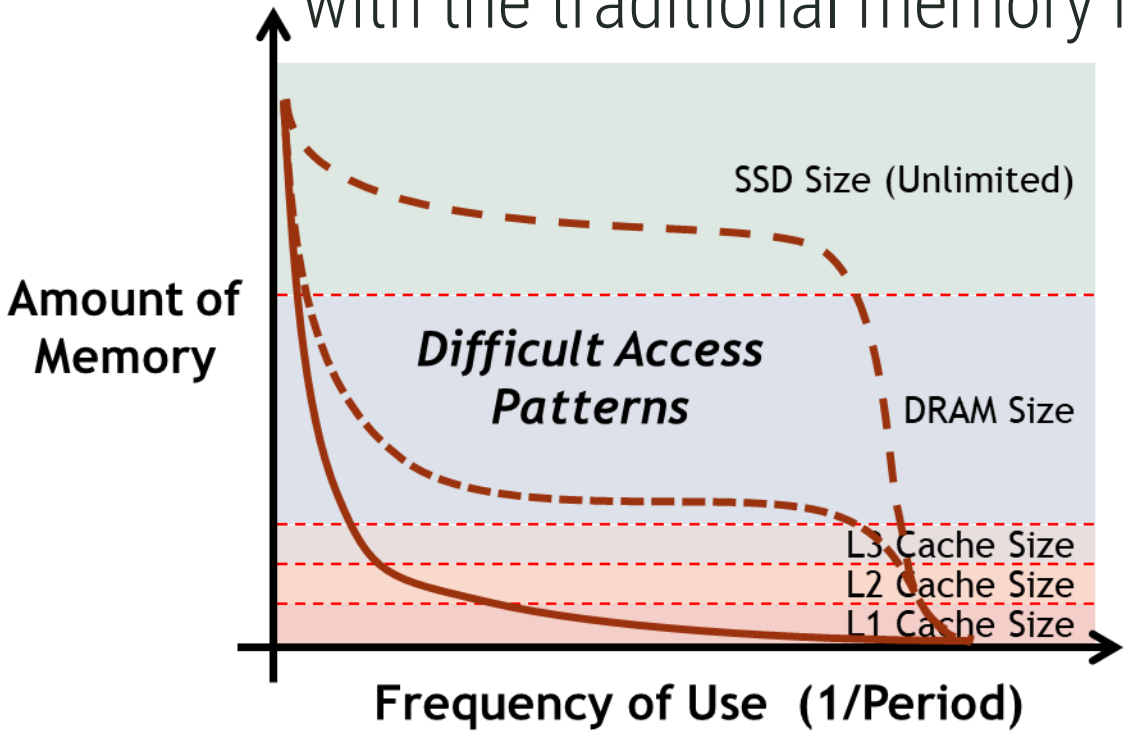
Data Access Patterns: Analysis Via “Working Set”



The cache system captures larger and larger working sets. The mostly infrequently used data is stored in system storage (SSD)

Compute-in-Memory for Difficult Access Patterns

Some applications have access patterns that do not “play nice” with the traditional memory hierarchy



Compute-in-Memory systems can target these applications!

Other Reasons for Compute-in-Memory

- Deterministic Data Patterns
  - In some cases, we know the exact data access pattern to be performed, so hierarchical memory systems do not provide a benefit and are inefficient.
- ASIC Capabilities Further Minimize Data Movement
  - In other cases, we can build in ASIC capabilities that take advantage of known data patterns.
- Analog Computation
  - In extreme cases, analog computation can be added to achieve most minimal data movement.

Compute-in-Memory is Not Always the Answer

- General Purpose Computing
  - You need an application to take advantage of.
- Low Application Importance
  - If this application is not >90% of the system time or power, then you will not be able to get a 10x improvement.
- Small Working Sets
  - Compute-in-memory often requires relatively large working sets to make sense.
  - Applications that fit in L1 cache are hard to improve.

What is Analog Compute?

What Does “Analog Compute” Mean?

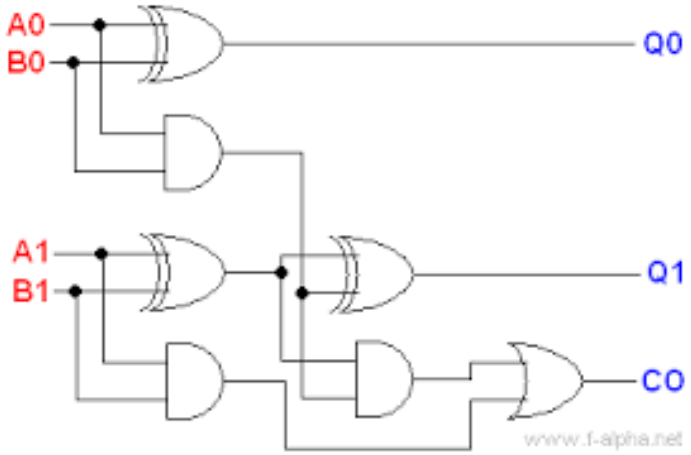
Definition of “Analog Computer”: a type of computer that uses the continuously changeable aspects of physical phenomena such as *electrical,* mechanical, or hydraulic quantities to model the problem being solved. In contrast, **digital computers** represent varying quantities symbolically.

What Does “Analog Compute” Mean?

Problem: 2+2

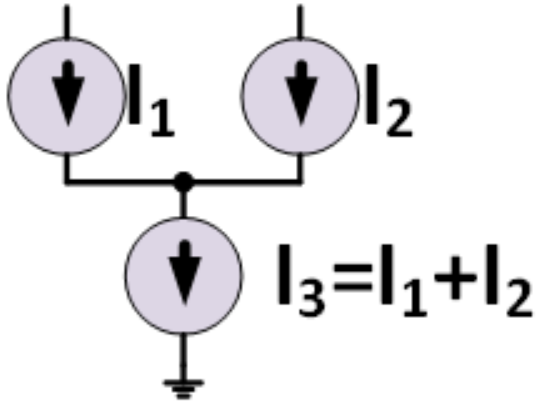
Digital Solution

0b10 + 0b10 = 0b100



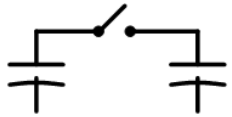
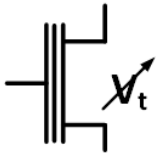
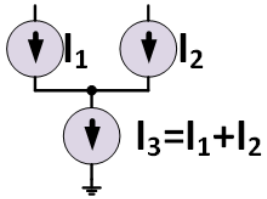
A0 = 0
A1 = 1
B0 = 0
B1 = 1

Analog Solution



I1 = 2uA
I2 = 2uA
I3 = 4uA

Analog Compute Options

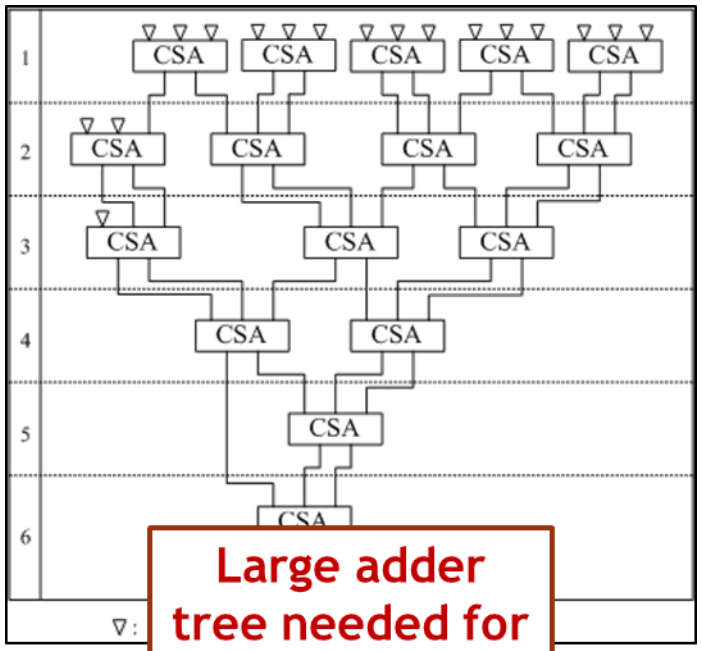
	Charge	Conductance	Current
			
Example Uses	•Multiply	•Absolute-value-of-difference	•Summation
Advantages	•Energy efficient •Result is a voltage	•Energy efficient •Area efficient	•Can use many inputs •Process tolerant
Disadvantages	•Area •Variation	•Special device (flash) •Variation	•Lower energy efficiency

Why and *When* is Analog Compute Useful?

1. Large problems
2. Noise tolerant algorithms

Digital Compute:

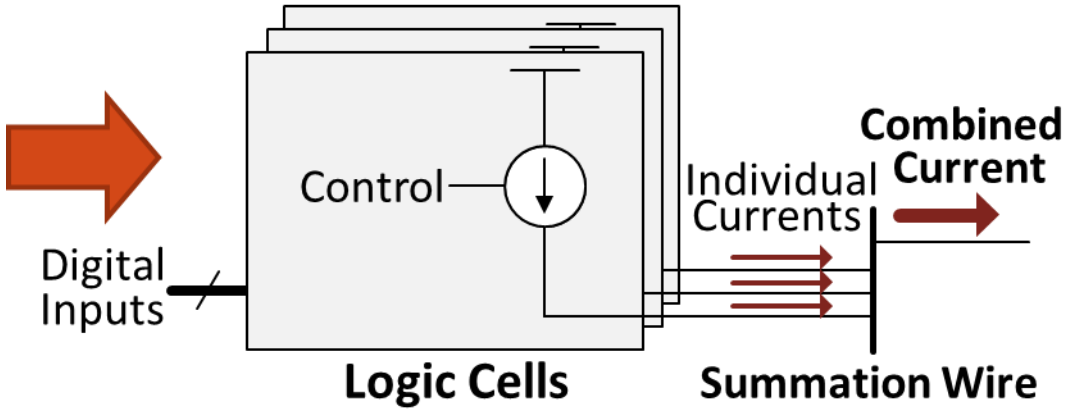
8168 full adders, 15 stage tree



Large adder tree needed for many addends

Analog Compute:

Current-mode summation, Single summation wire



Single Wire Summation Regardless of how many addends

What is Analog Compute *bad* at?

- Downsides of Analog Computation
  - Noise! → compute using *changing* signals introduces noise
  - Flexibility
  - Data corruption (coupling, routing loss)

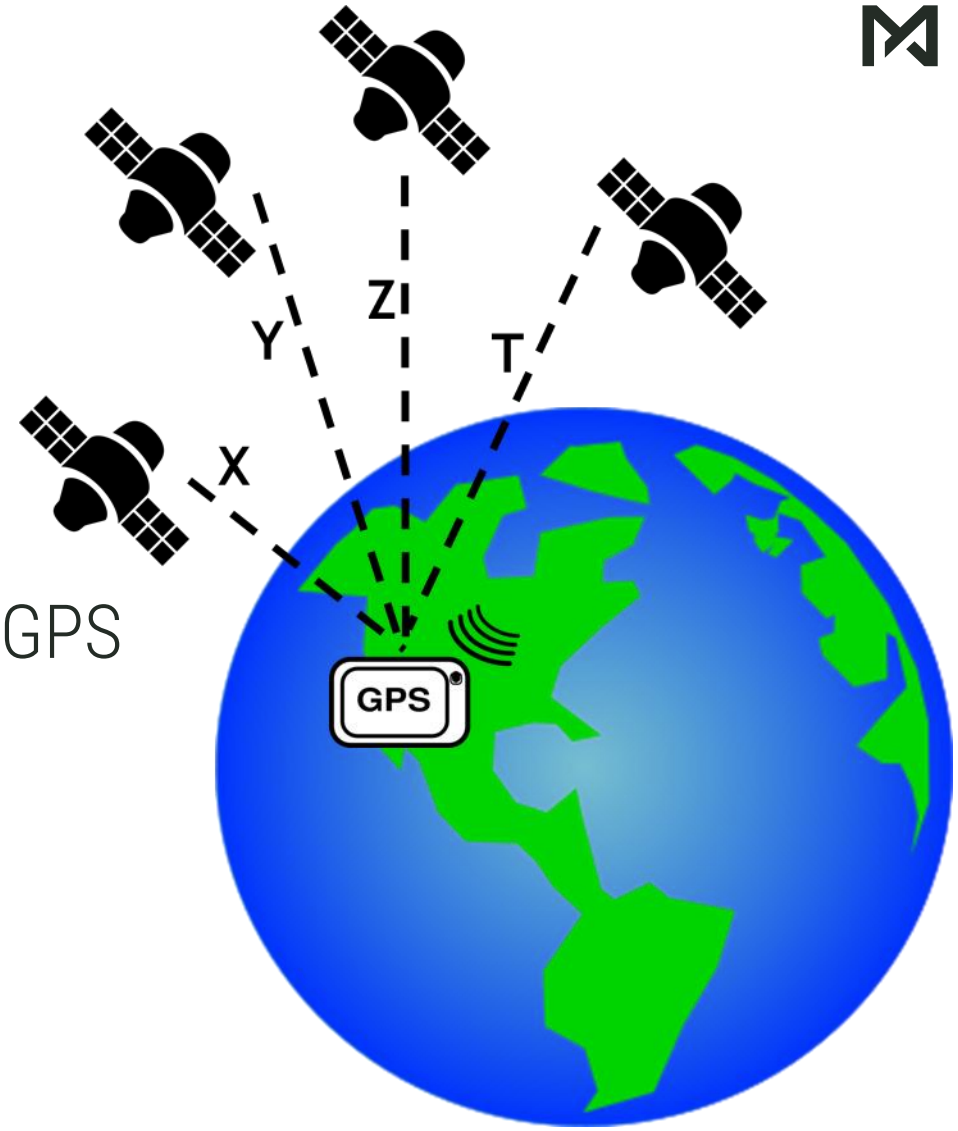
Use analog where analog is best
Digital where digital is best

Analog compute is not a silver bullet!

Mythic’s First Experience With Compute-in-Memory: GPS

Mythic Circa 2012 (aka Isocline)

Working on an SBIR for GPS

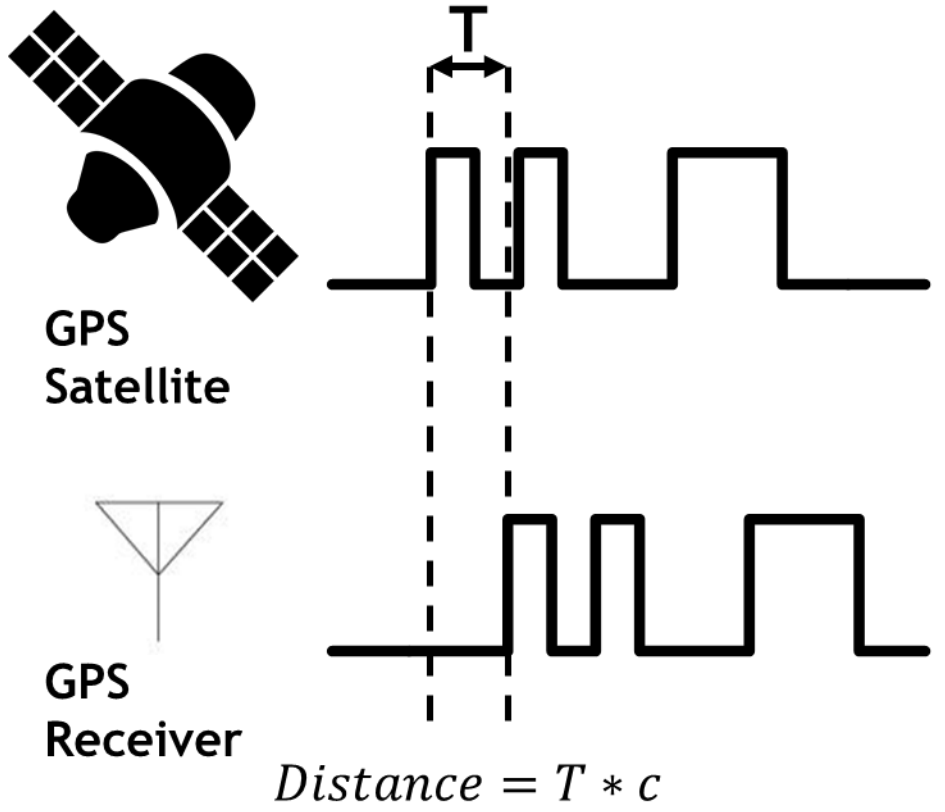


- Mike Henry received an SBIR to do low power GPS on chip
  - Subthreshold digital was the initial proposal

** All GPS images courtesy of Skylar's ISSCC presentation

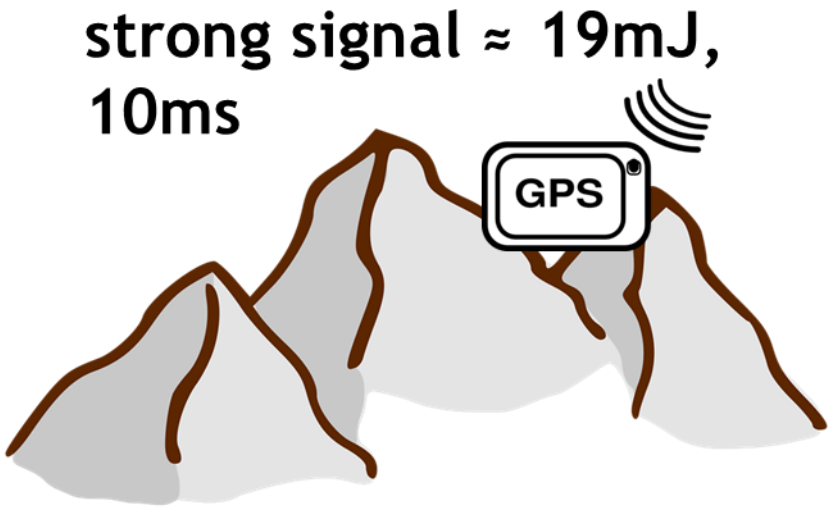
GPS Acquisition is a Large Search…

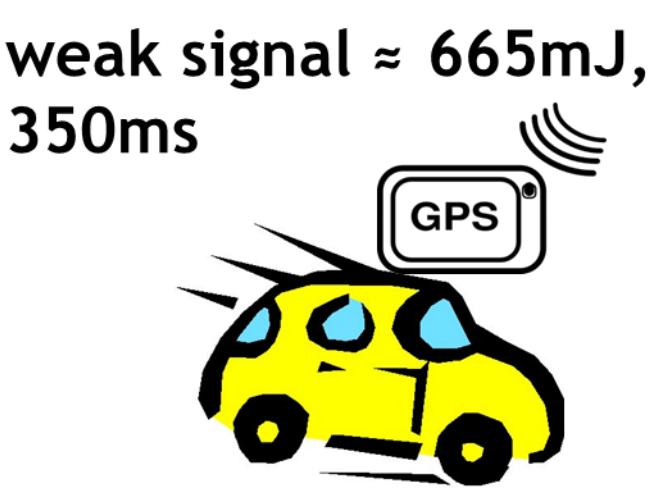
- Time-domain search
- Timing offset (T) is calculated
  - Done through correlation
    - 1000's of 2-bit vector multiply
    - 1000's of results to accumulate
- Pattern is very long
  - Reject noise
  - Increase gain



…and Energy/Time Intensive

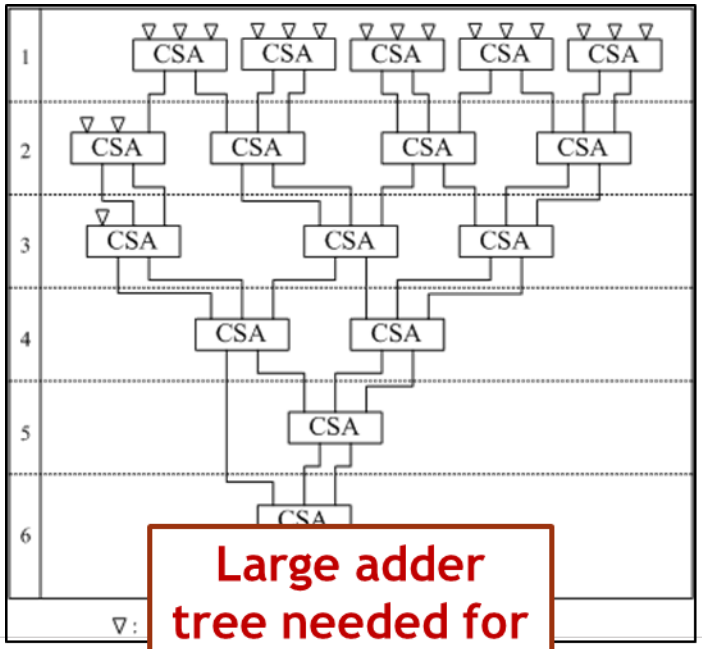
- Requires many operations
  - (10-350)×10^9 for civilian
  - 35×10^{12} for military
- Energy intensive
  - 19-665 mJ per satellite
- Time consuming
  - 10-350 ms per satellite



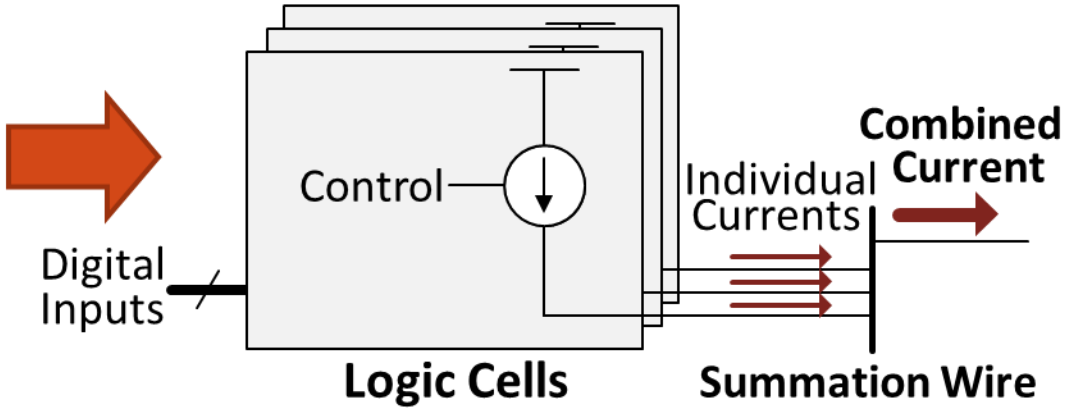


But Can Be Done with a Single Wire in Analog

- Digital:
  - 8168 full adders
  - 15 stage tree
- Analog:
  - Current-mode summation

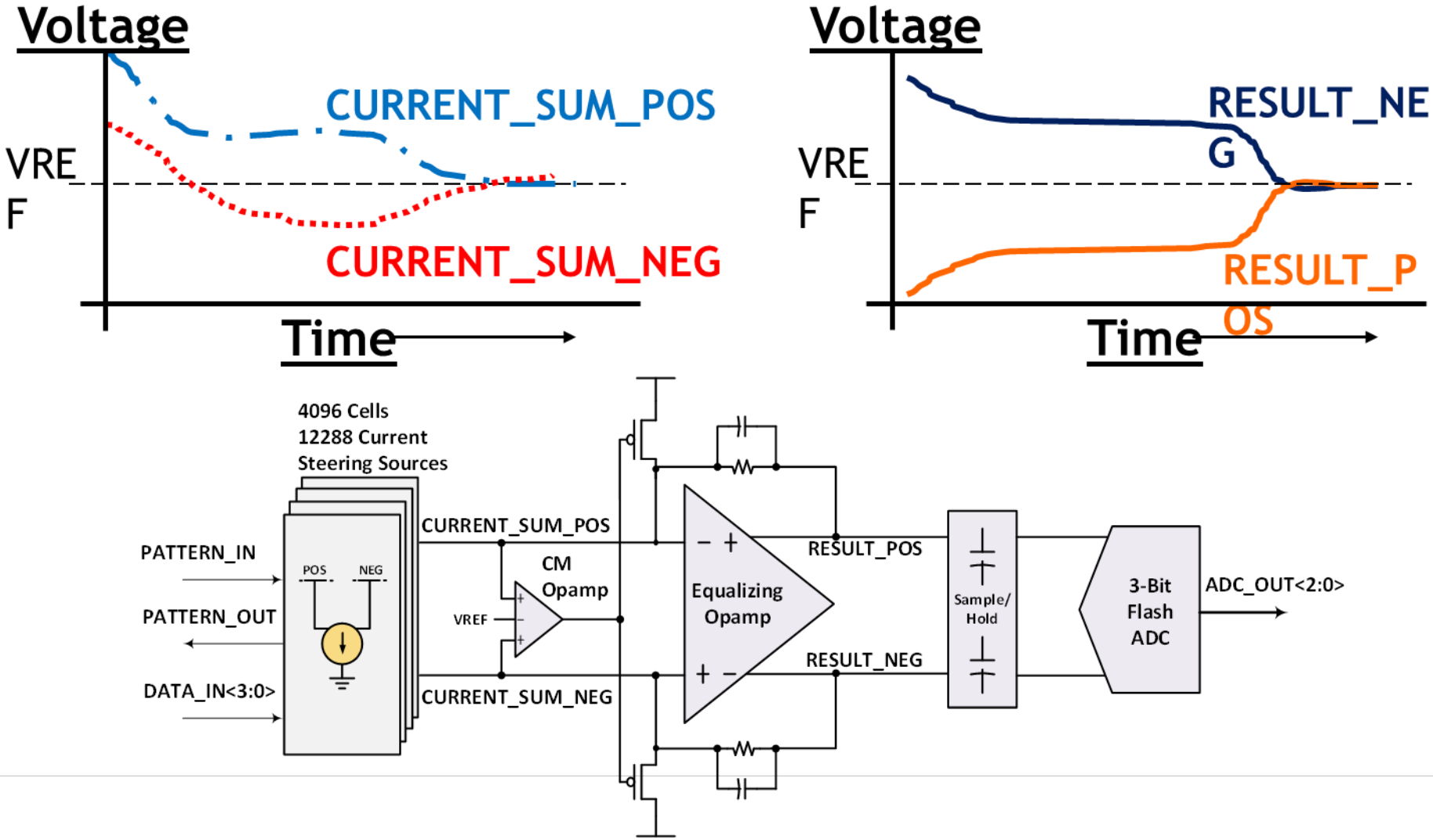


Large adder tree needed for many addends

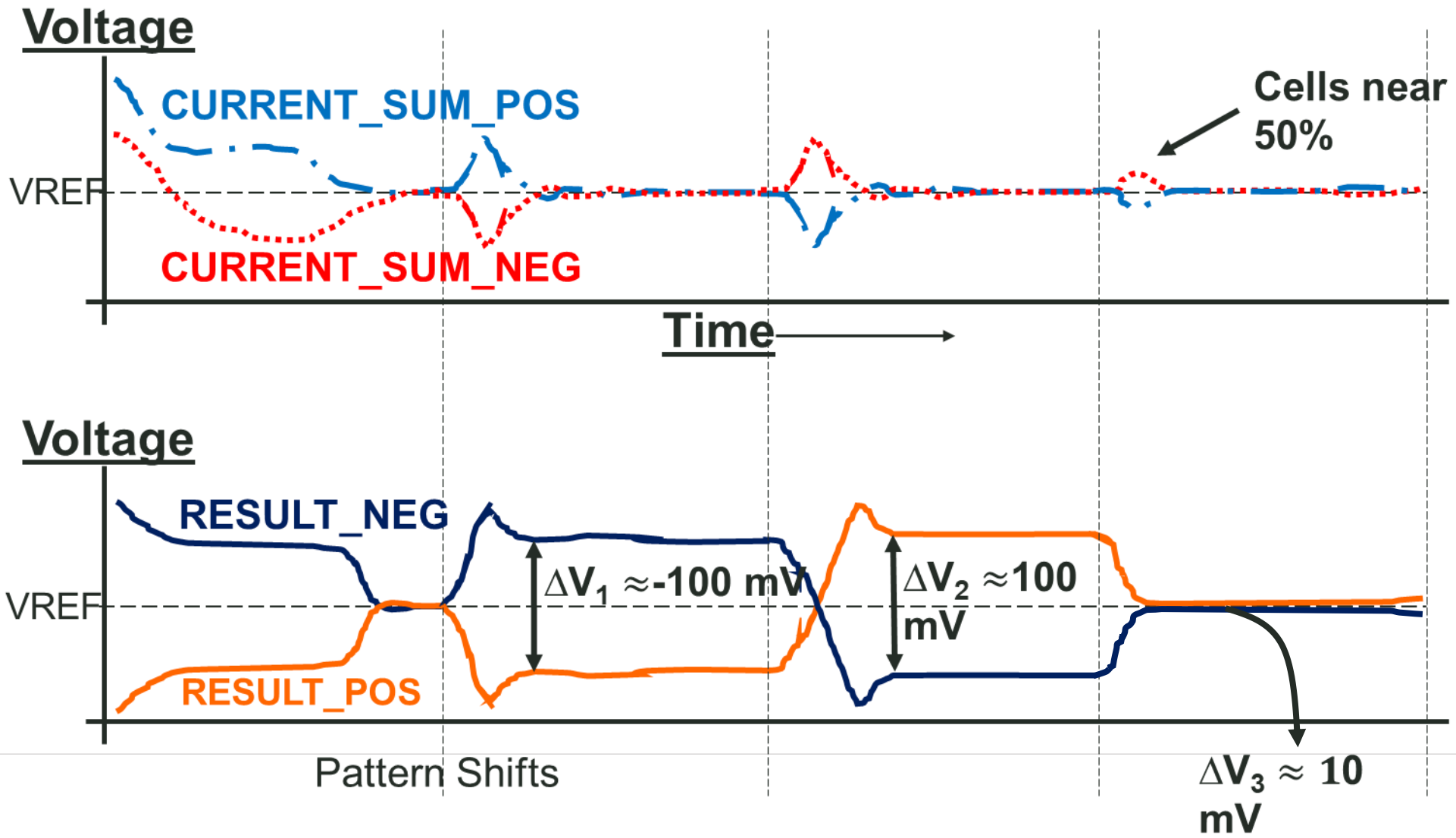


Single Wire Summation Regardless of how many addends

Example Calculation: Initialization

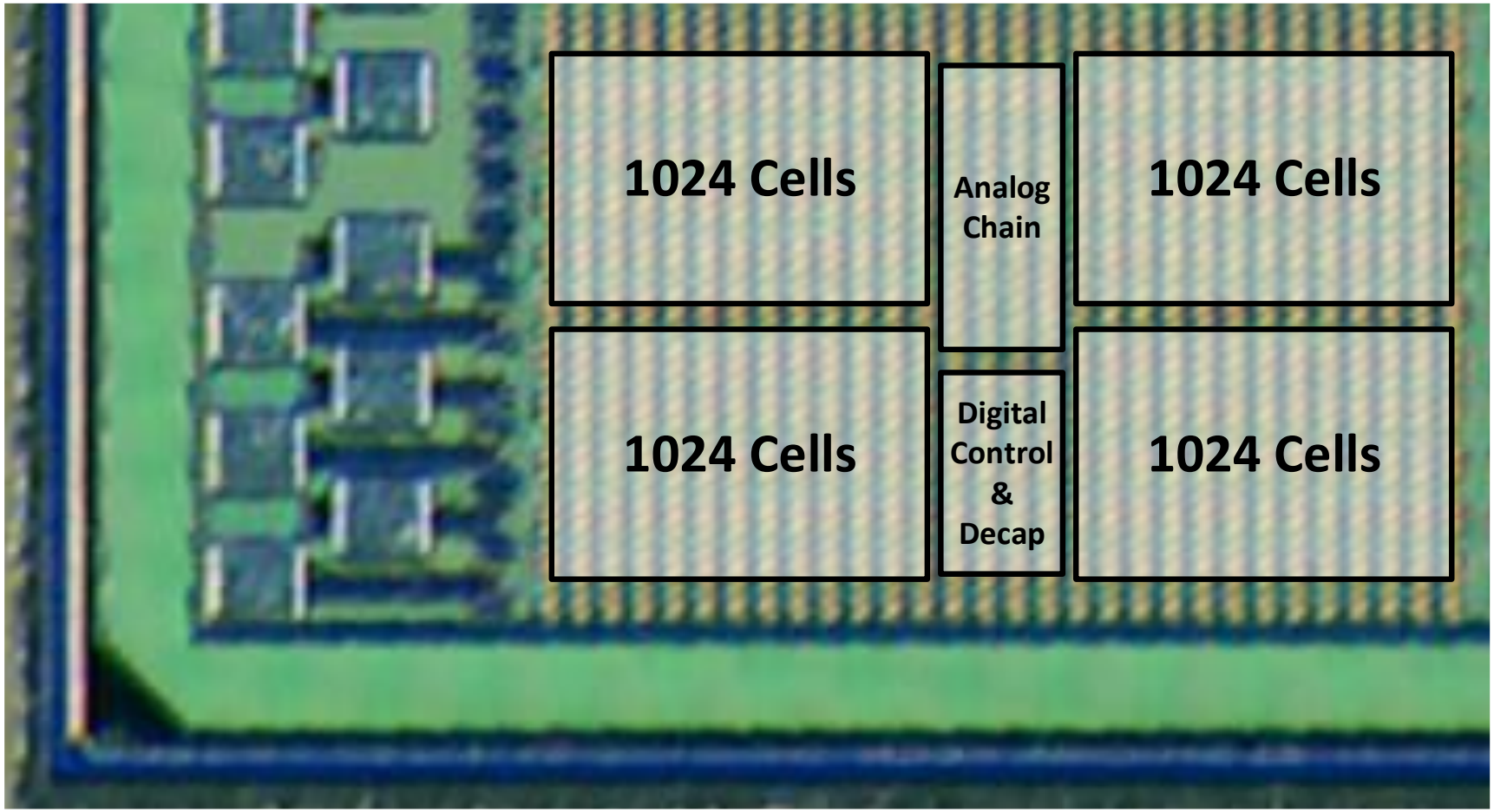


Example Calculation

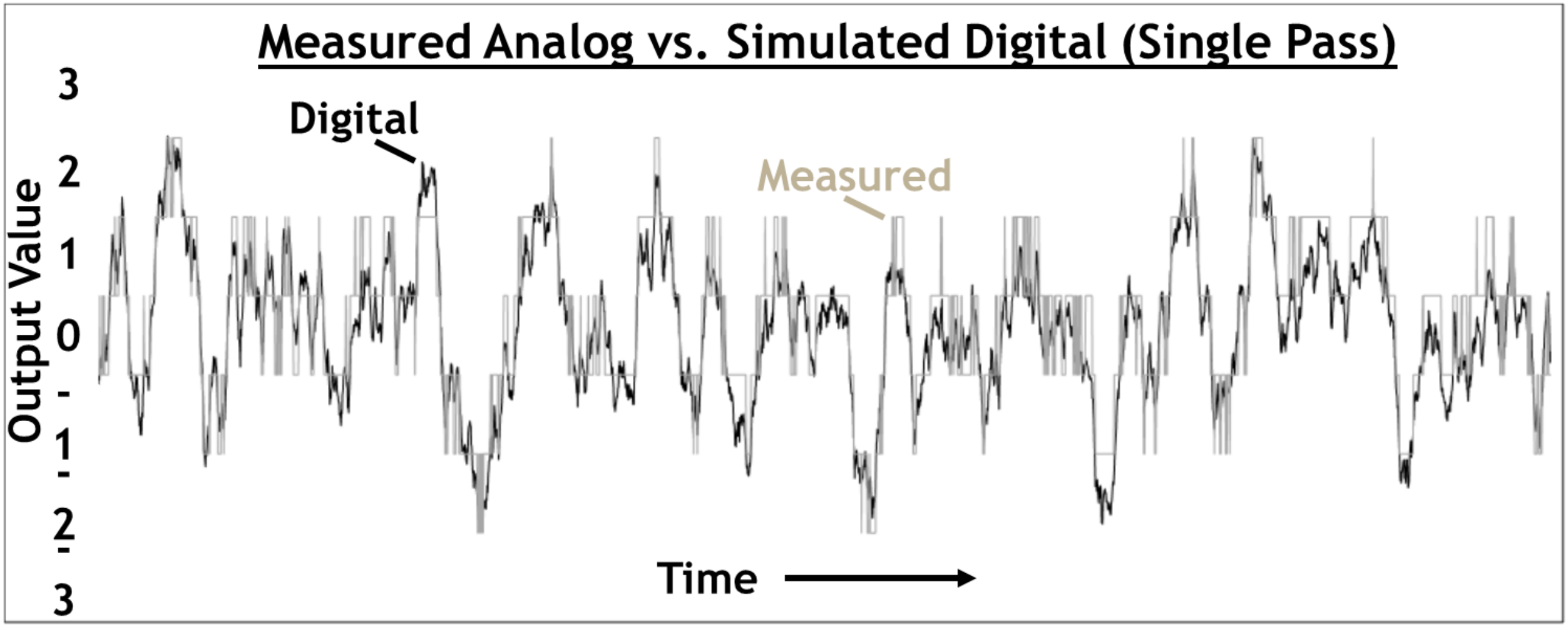


GPS Die Photo

- TSMC65LP
- 0.325mm^2

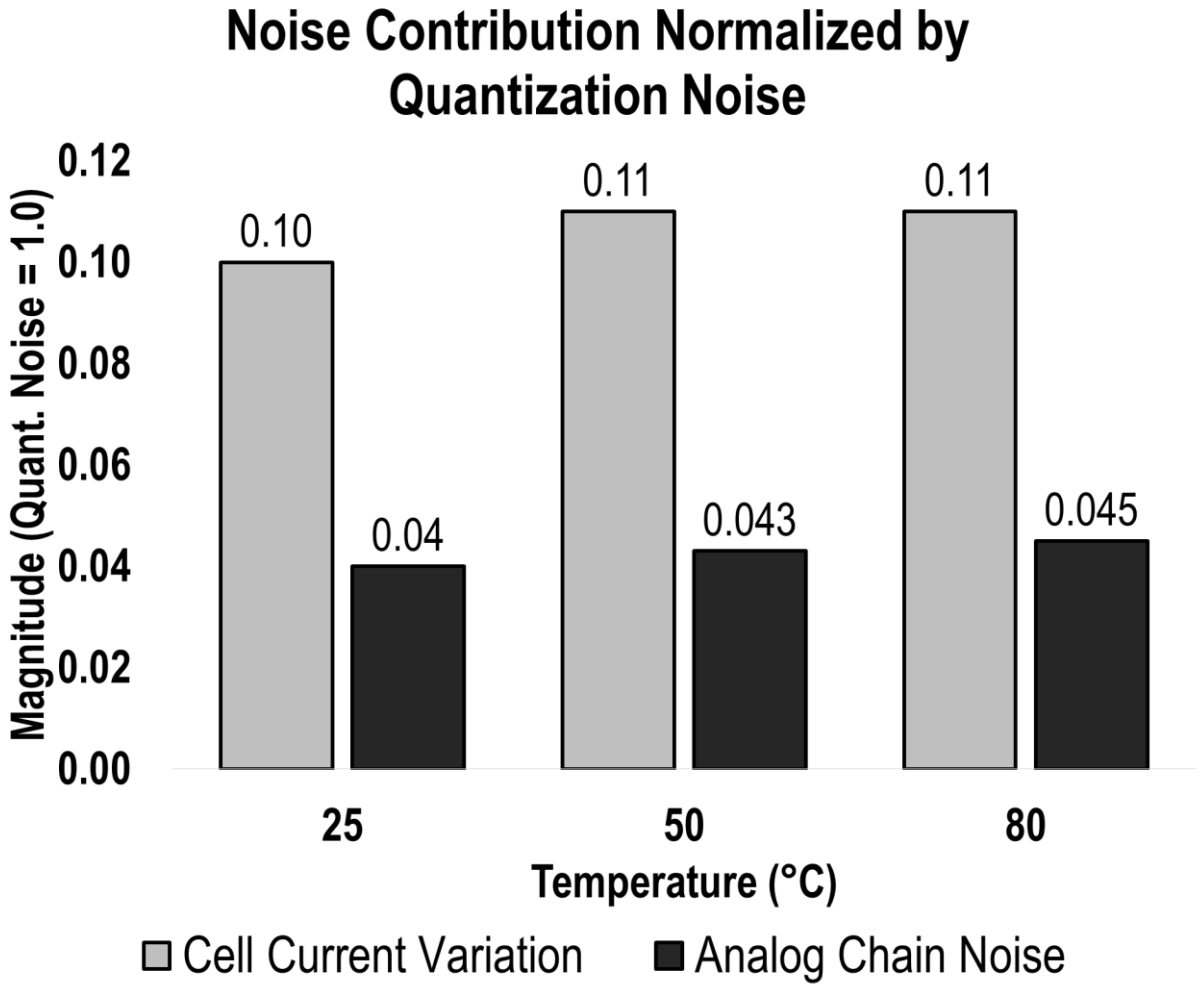


Results: Implementation vs. Ideal



Analog Compute Noise is Less than Quantization

- Signal has inherent noise
  - RF front-end
- Analog compute noise:
  - 10x lower than quantization noise
  - Analog noise is dominated by current source variation



Results: Comparison

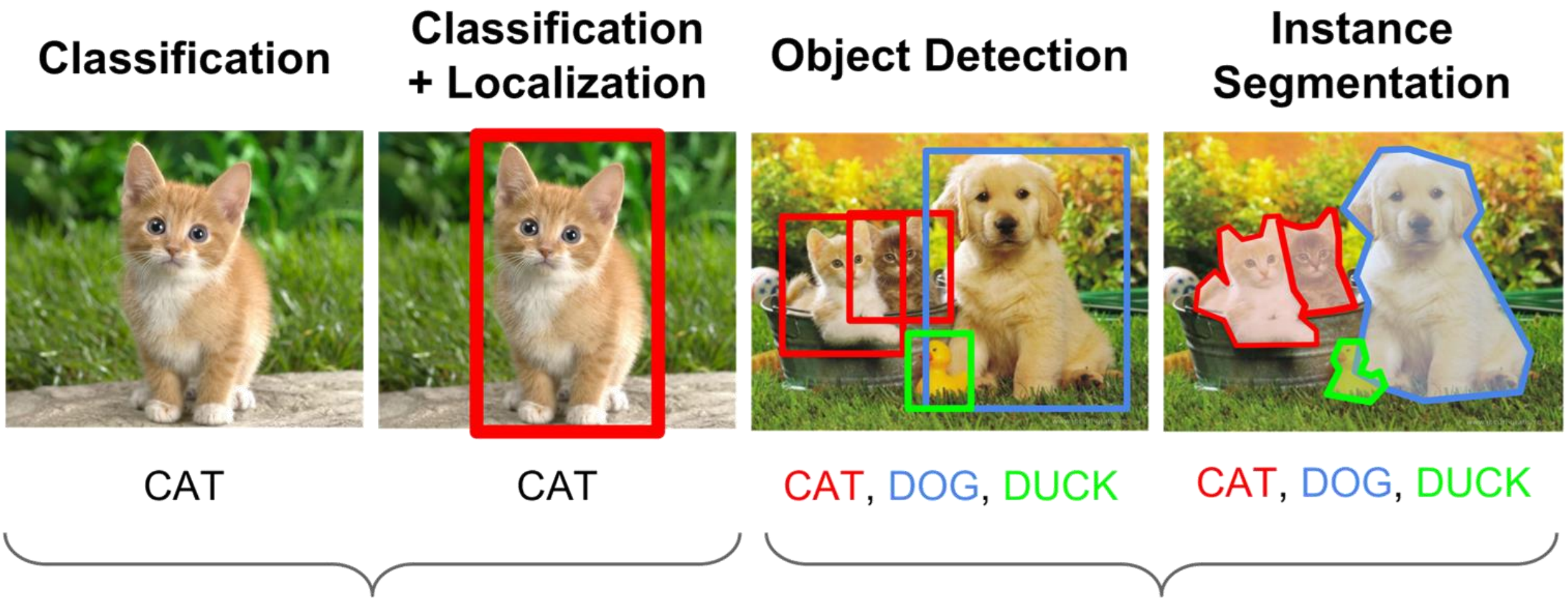
- 340-27,000x performance increase
- 67x energy efficiency increase
- Scalable for application

	This Work	MITRE	JSSC'05	ISCAS'11
Technology	65 nm	180 nm	350 nm	130 nm
V_{DD} (V)	1.20 (Analog) 1.15 (Digital)	1.8	2.0	1.0
Clock Frequency (MHz)	170	20.46	8	0.2
Power (mW)	18.9*	1,900	2	0.0004
TOPS	0.70*	1.05	2.05E-3	2.56E-5
TOPS/W	36.8	0.55	1.05	64.0
TOPS/mm^2	2.154*	0.0119	0.0038	0.000197
Vector Length	4,096*	51,150	256	128
Quantization	2-bit	2-bit	Analog	Analog
Area (mm^2)	0.325*	88.0	0.54	0.13
Topology	Digital storage/ switched current	All digital	Analog storage/ switched current	Analog storage/ switched capacitor

*The design could be tiled to proportionally scale these metrics.

Application Analysis: Neural Networks

Neural Networks = *Intuition*



DNNs are Largely Multiply-Accumulate

Primary DNN Calculation is Input Vector * Weight Matrix = Output Vector

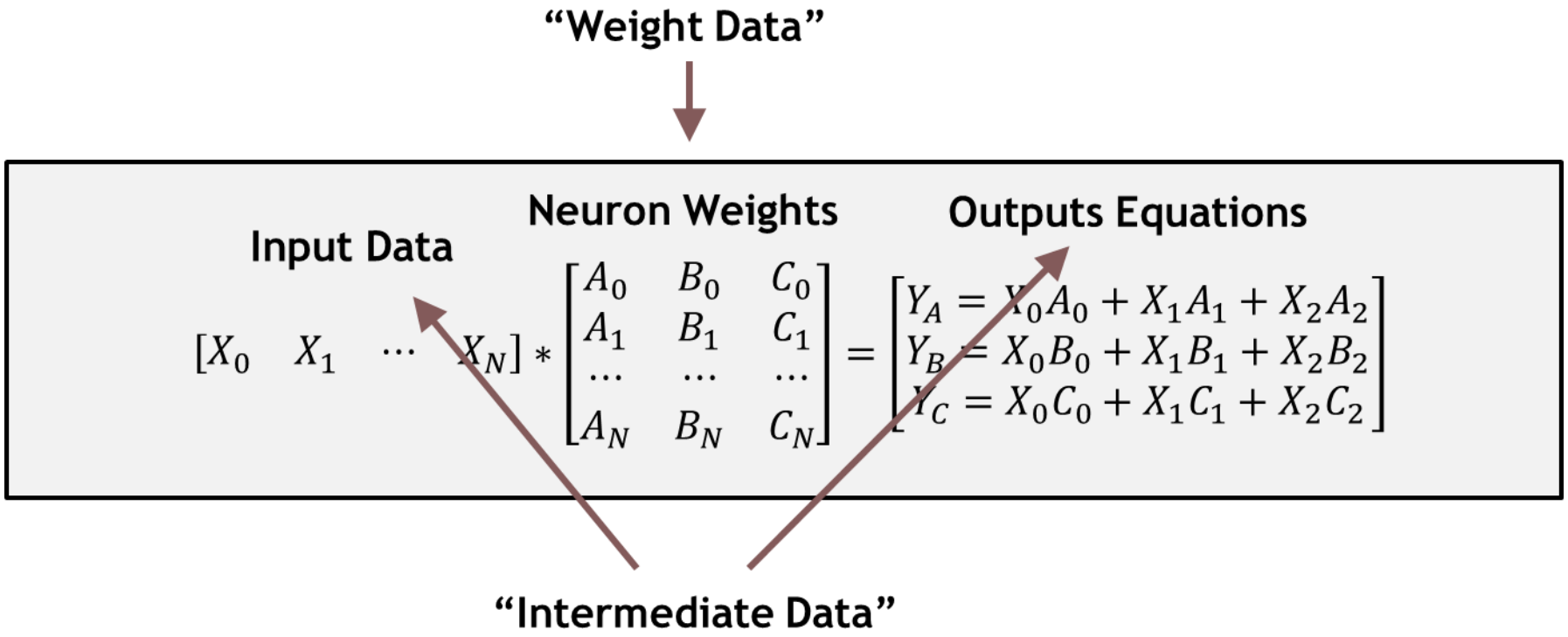
Input Data **Neuron Weights** **Outputs Equations**

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

Key Operation: Multiply-Accumulate, or "MAC"

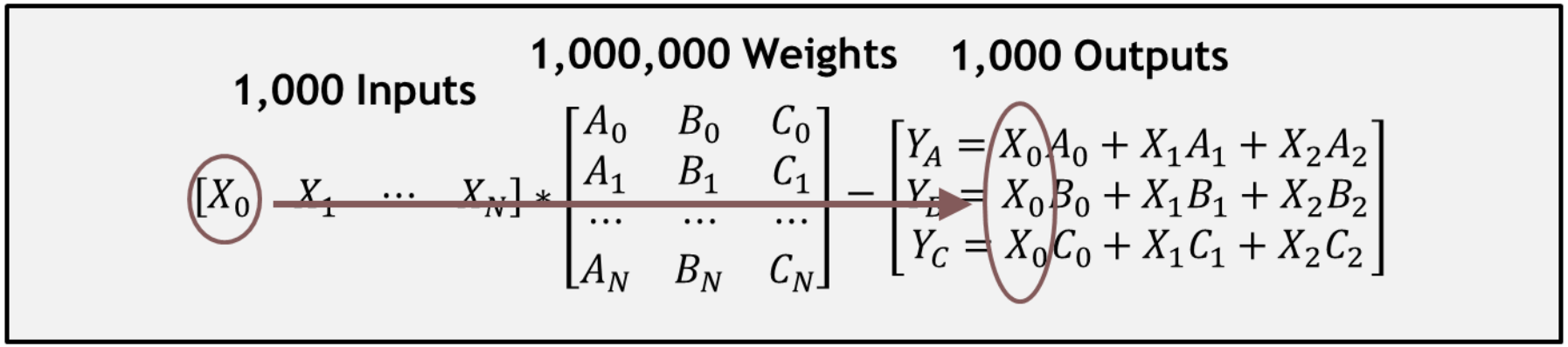
Figure of Merit: How many picojoules to execute a MAC?

Memory Access Includes Weight Data and Intermediate Data



For a 1000 input, 1000 neuron matrix…

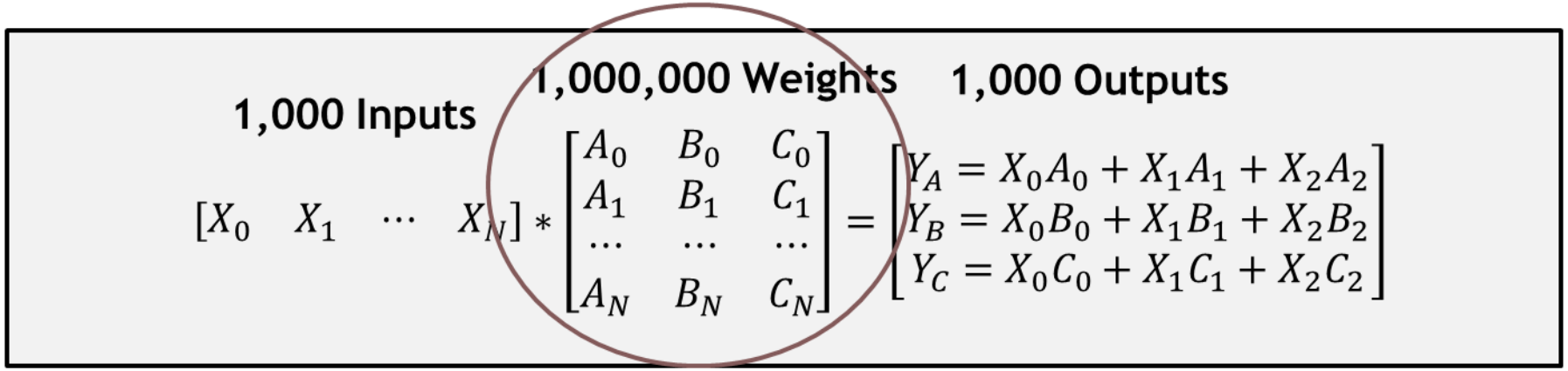
Intermediate Data Accesses are Naturally Amortized



Intermediate data accesses are amortized **64-1024x** since they are used in many MAC operations

For a 1000 input, 1000 neuron matrix…

Weight Data Accesses are Not Amortized



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

Network	Weights	MACs	...@ 30 FPS
AlexNet[1]	61 M	725 M	22 B
ResNet-18[1]	11 M	1.8 B	54 B
ResNet-50[1]	23 M	3.5 B	105 B
VGG-19[1]	144 M	22 B	660 B
OpenPose[2]	46 M	180 B	5400 B

Very hard to fit this in an Edge solution

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

[1]: 224x224 resolution
[2]: 656x368 resolution

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

	Enterprise With DRAM	Enterprise No-DRAM	Edge With DRAM	Edge No-DRAM
SRAM	<50 MB	100+ MB	< 5 MB	< 5 MB
DRAM	8+ GB	-	4-8 GB	-
Power	70+ W	70+ W	3-5 W	1-3 W
Sparsity	Light	Light	Moderate	Heavy
Precision	32f / 16f / 8i	32f / 16f / 8i	8i	1-8i
Accuracy	Great	Great	Moderate	Poor
Performance	High	High	Very Low	Very Low
Efficiency	25 pJ/MAC	2 pJ/MAC	10 pJ/MAC	5 pJ/MAC

Mythic is Fundamentally Different

	Enterprise With DRAM	Enterprise No-DRAM	Edge With DRAM	Edge No-DRAM	Mythic NVM
SRAM	<50 MB	100+ MB	< 5 MB	< 5 MB	< 5 MB
DRAM	8+ GB	-	4-8 GB	-	-
Power	70+ W	70+ W	3-5 W	1-3 W	1-5 W
Sparsity	Light	Light	Moderate	Heavy	None
Precision	32f / 16f / 8i	32f / 16f / 8i	8i	1-8i	1-8i
Accuracy	Great	Great	Moderate	Poor	Great
Performance	High	High	Very Low	Very Low	High
Efficiency	25 pJ/MAC	2 pJ/MAC	10 pJ/MAC	5 pJ/MAC	0.5 pJ/MAC

Mythic is Fundamentally Different

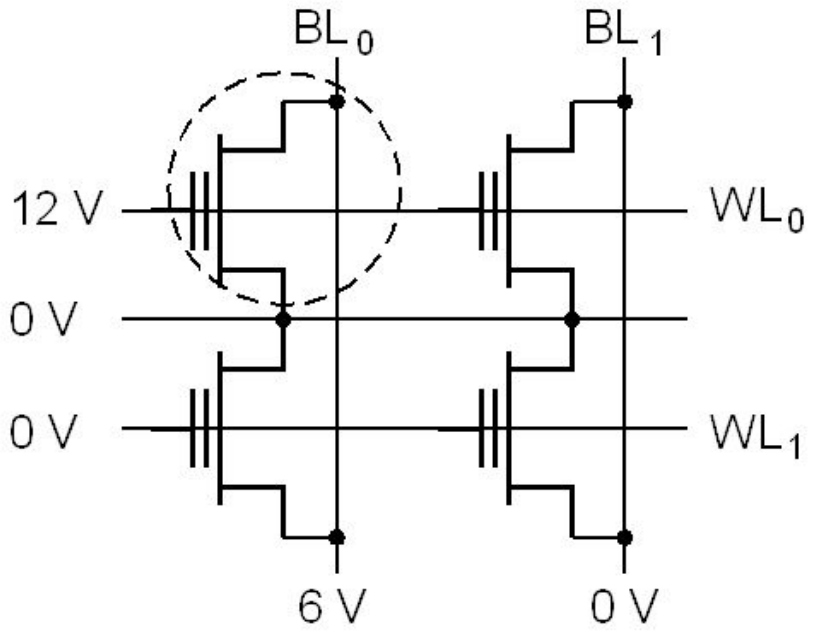
	Enterprise With DRAM	Enterprise No-DRAM	Edge With DRAM	Edge No-DRAM	Mythic NVM
SRAM	<50 MB	100+ MB	< 5 MB	< 5 MB	< 5 MB
DRAM	8+ GB	-	4-8 GB	-	-
Power	70+ W	70+ W	3-5 W	1-3 W	1-5 W
Sparsity	Light				None
Precision	32f / 16f / 8i	8i			1-8i
Accuracy	Great	Great	Moderate	Poor	Great
Performance	High	High	Very Low	Very Low	High
Efficiency	25 pJ/MAC	2 pJ/MAC	10 pJ/MAC	5 pJ/MAC	0.5 pJ/MAC

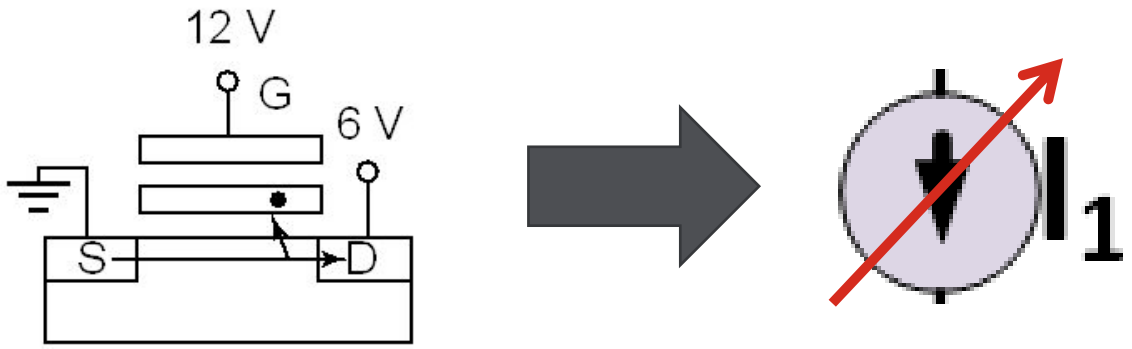
Also, Mythic does this in a 40nm process, compared to 7/10/16nm

Analog Compute-in-Memory Using Flash Transistors

What are Flash Transistors?

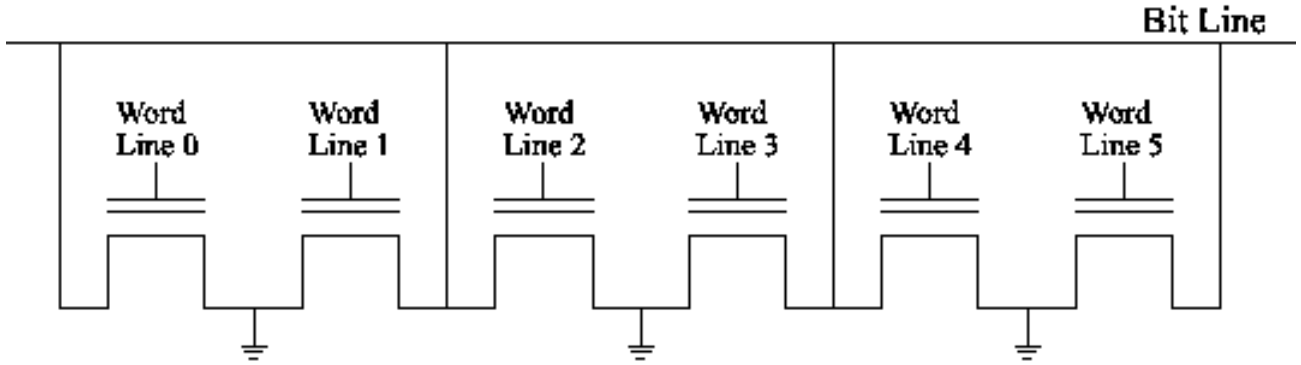
- Transistor with an extra “floating” gate
- Floating gate traps electrons → changes the threshold voltage of the device
- With constant VG,VS,VD – flash cell operates as a programmable current source





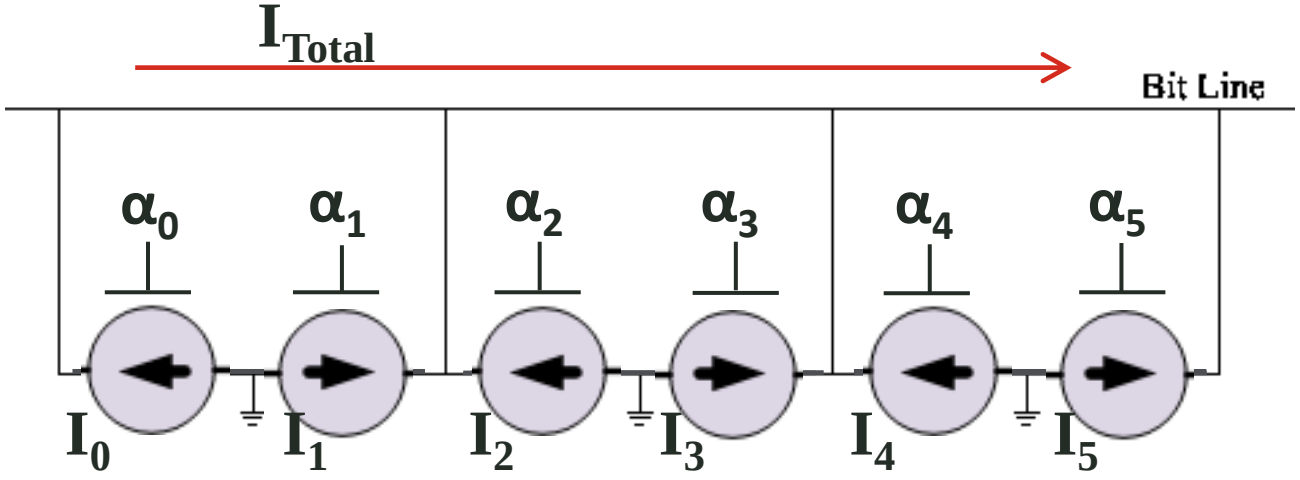
Flash Cells as Programmable Current Sources

$$I_n = \alpha_n \times I'_n = \alpha_n \times (V_{GS} - V_{th_n})$$
$$I_{Total} = I_0 + I_1 + I_2 + I_3 + I_4 + I_5$$



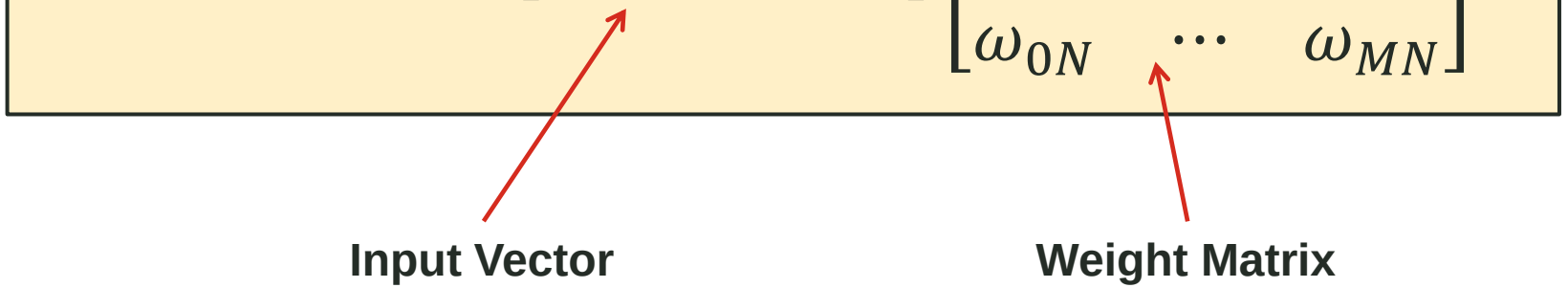
- Each flash cell acts as a *gated* current source (multiplication)
- Flash cells on the same bit line sub current (accumulation)

Multiply-accumulate function!



Mapping NN Weights to Flash Current

$$z = A \times W = [\alpha_0 \quad \cdots \quad \alpha_N] \begin{bmatrix} \omega_{00} & \cdots & \omega_{M0} \\ \vdots & \ddots & \vdots \\ \omega_{0N} & \cdots & \omega_{MN} \end{bmatrix}$$



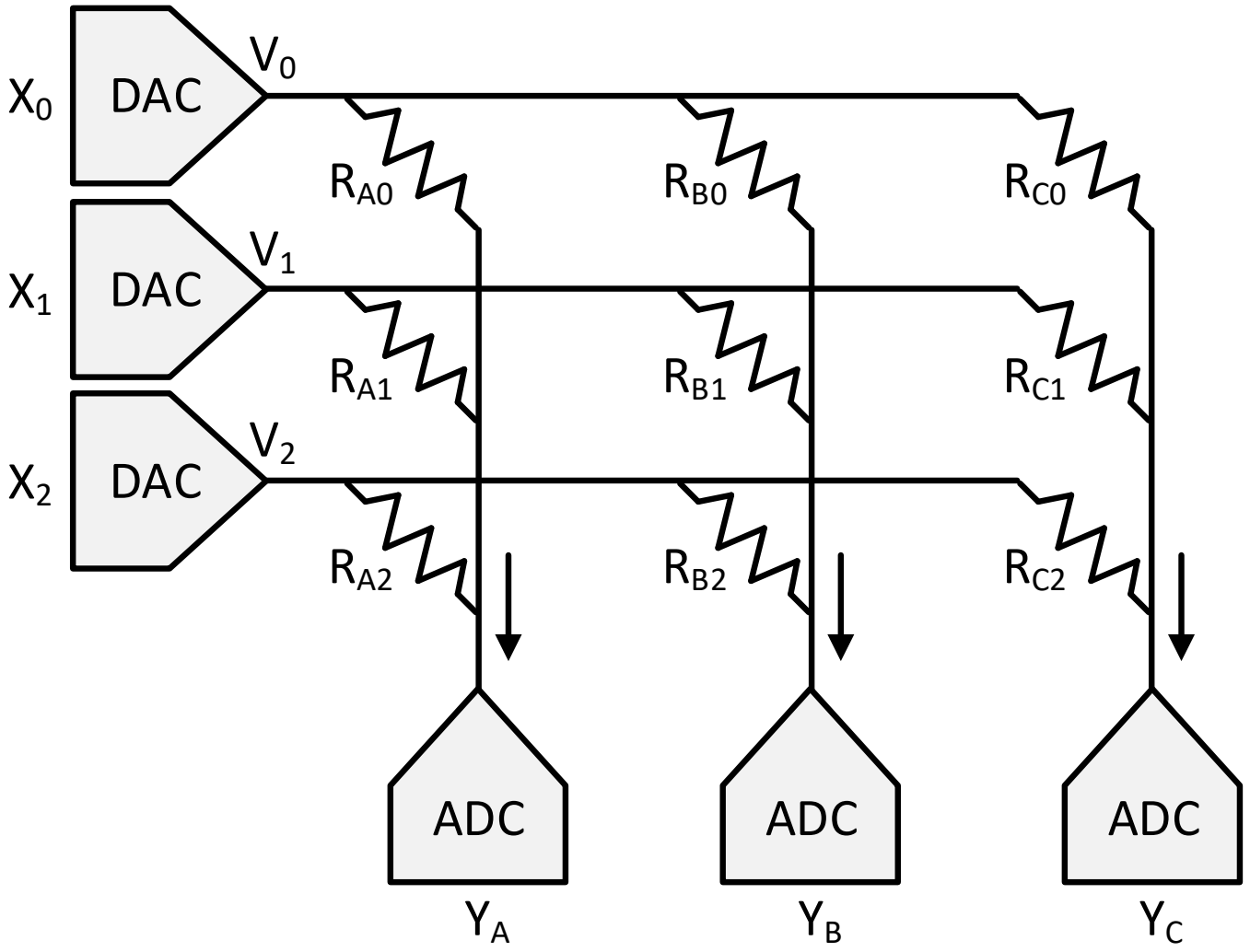
Neural Networks via Flash + Analog Compute

Flash transistors can be modeled as **variable resistors** representing the weight

The V=IR current equation will achieve the math we need:

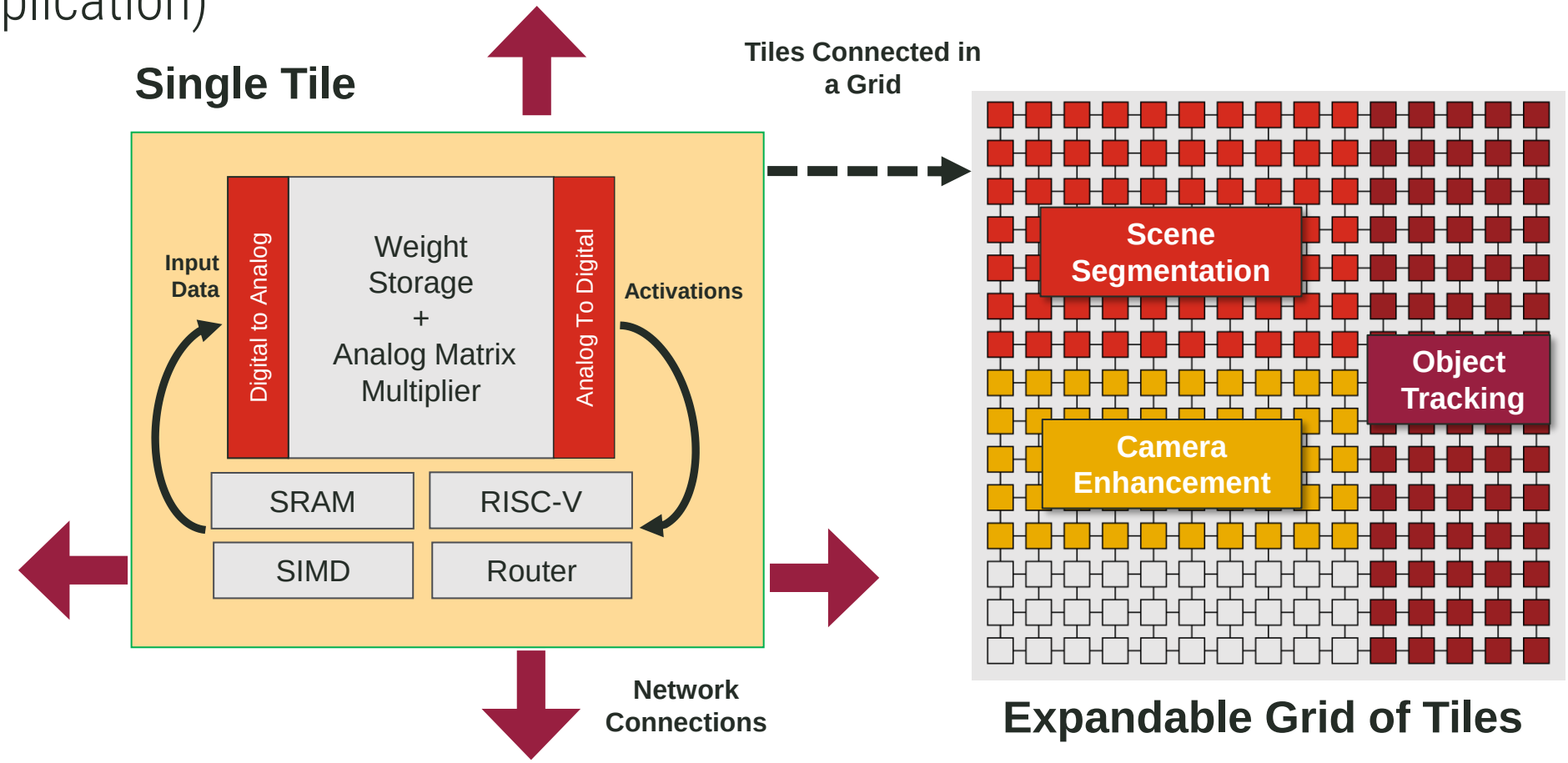
- Inputs (X) = DAC
- Weights (R) = Flash transistors
- Outputs (Y) = ADC Outputs

The ADCs convert current to digital codes, and provide the non-linearity needed for DNN

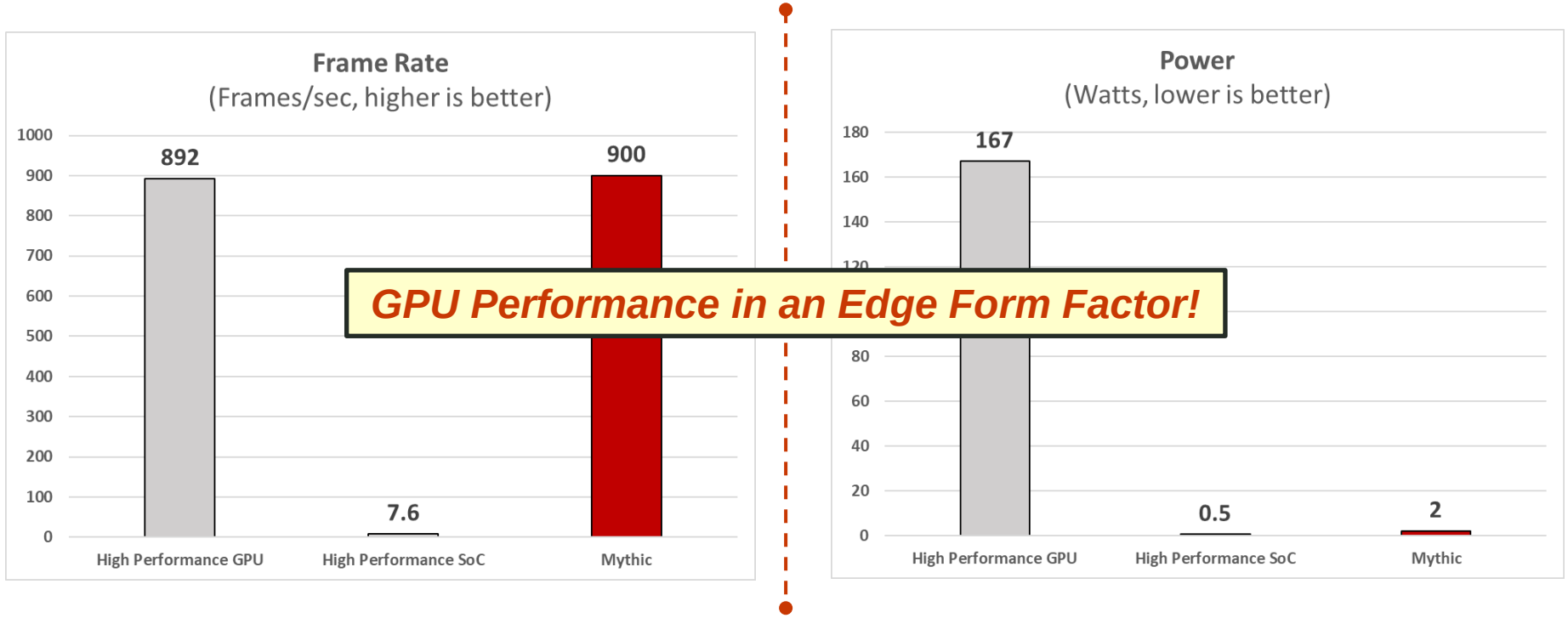


Mythic Neural Network Digital Architecture

Our difference: Mixed-Signal Compute (Ultra-dense storage + matrix multiplication)

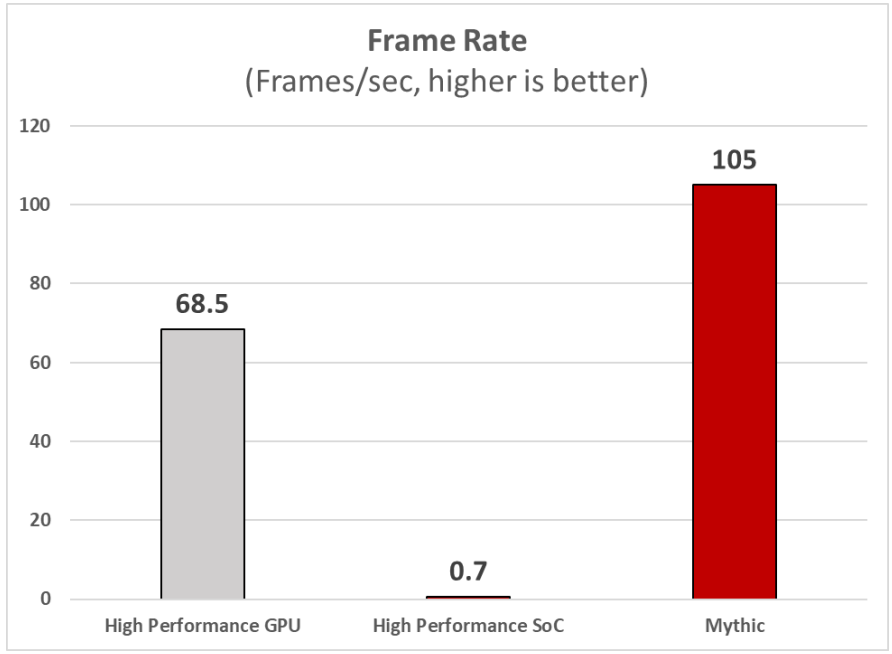


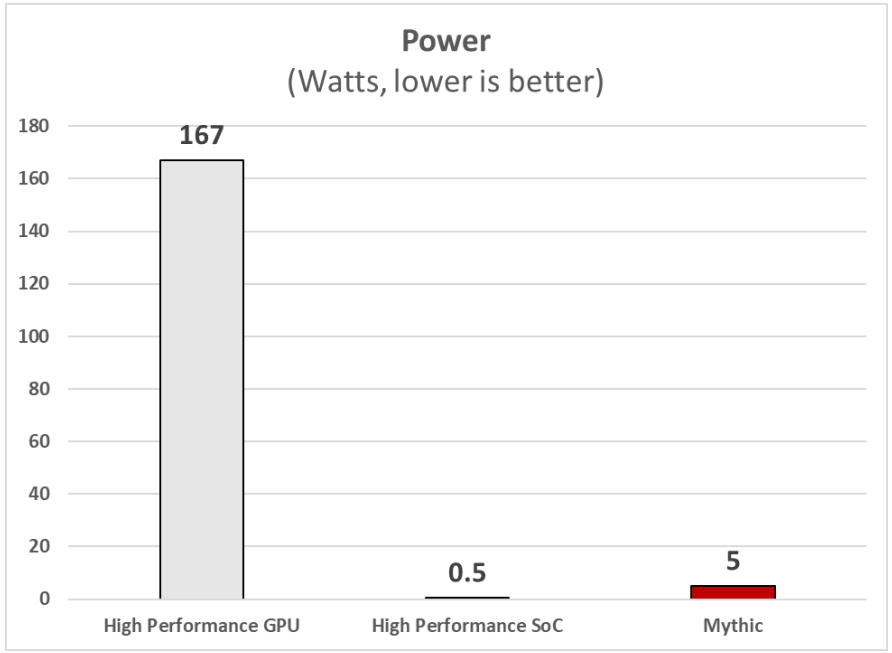
Example Application: ResNet-50



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

Example Application: OpenPose

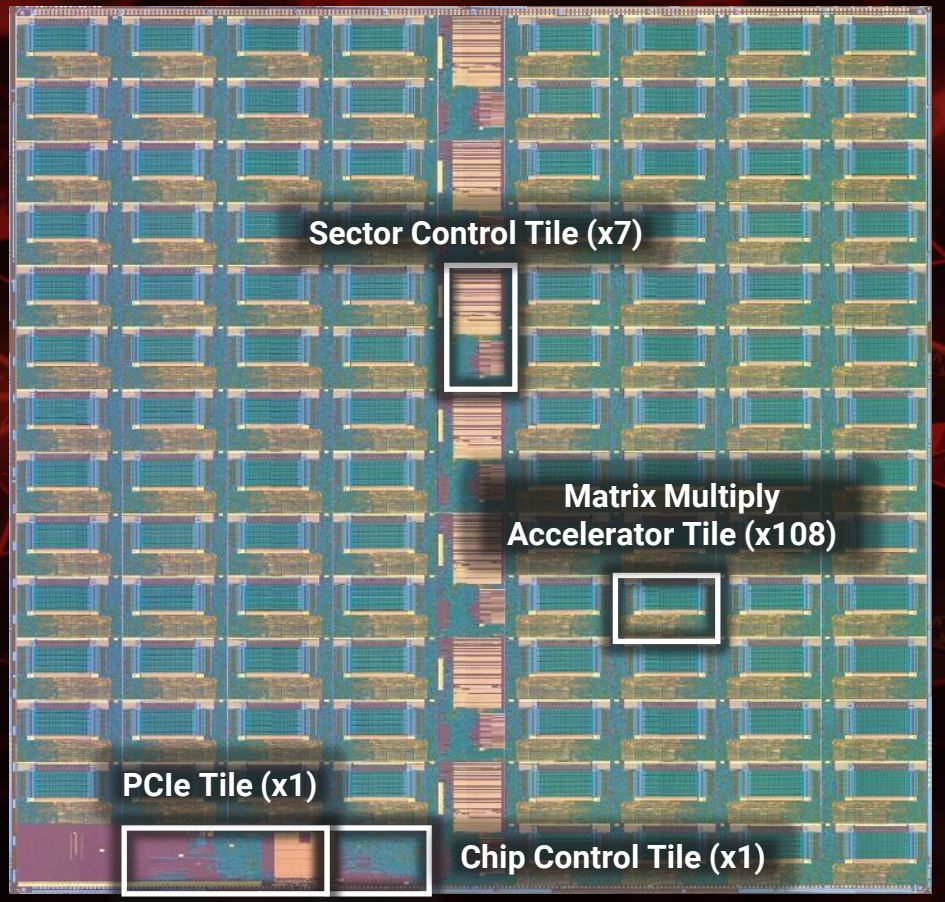




Running at 656x368 resolution. Mythic estimated, GPU/SoC measured

Mythic IPU Overview

- ☑ Low Latency
  - Runs batch size = 1, single frame latency
- ☑ 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



Conclusion

What is Possible with Compute-in-Memory?

- >10x improvement in energy efficiency
- >10x improvement in performance

- Application specific benefits
  - Not every algorithm can benefit from CiM!
  - Some benefit more than others

Compute-in-Memory Considerations

- What does the working set look like?
  - Is it “wide”?
  - Is it “large”?
- How important is this algorithm to our system?
  - Does it use up to 90% of something?
- How predictable are our data patterns?
  - Can we reduce data movement somehow?