In-Datacenter Performance Analysis of a Tensor Processing Unit

Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon

Google, Inc., Mountain View, CA USA

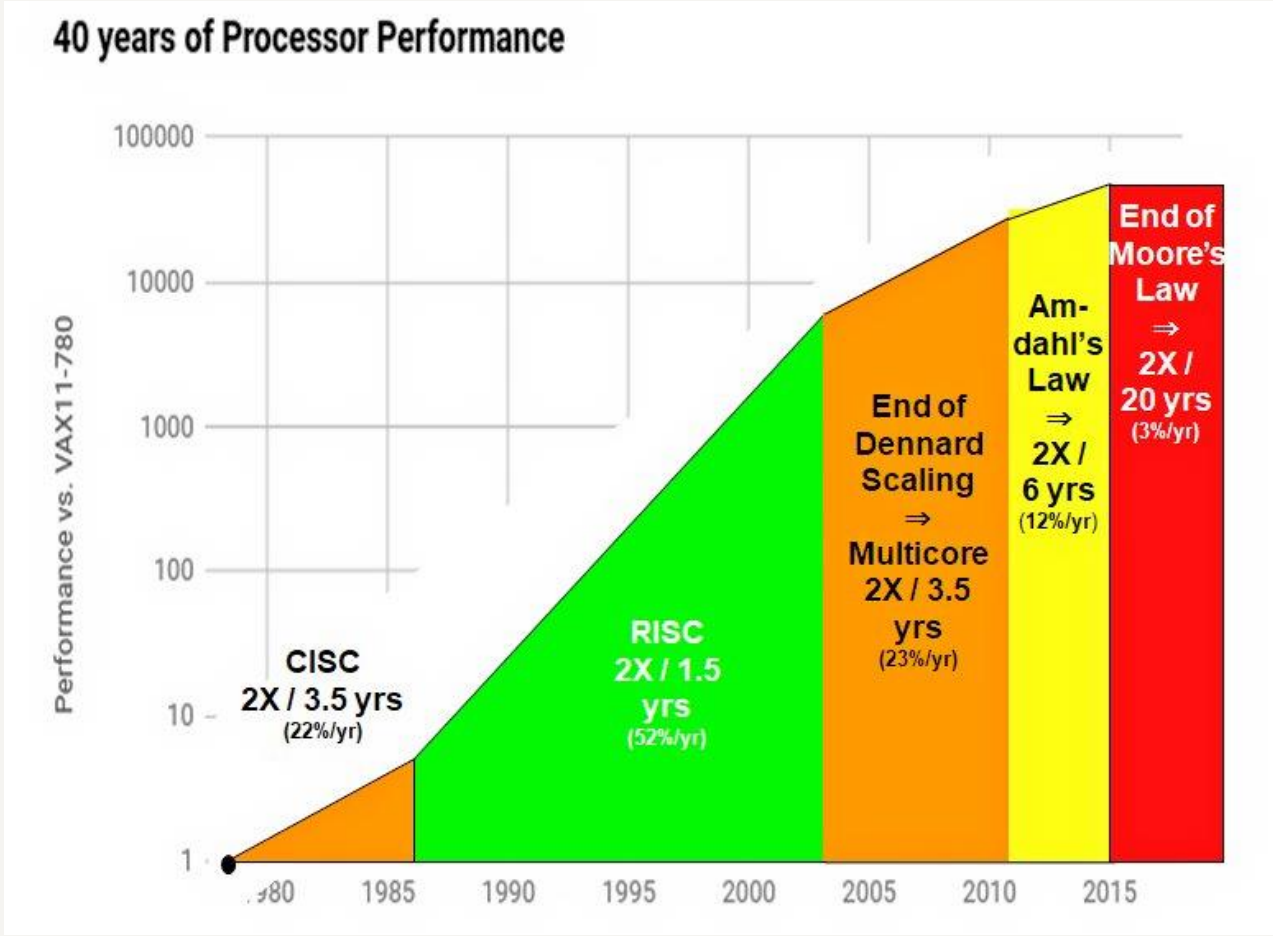
jouppi@google.com

Presenter:
Sushil Raj Regmi
University of North Texas

Evolution of Processors

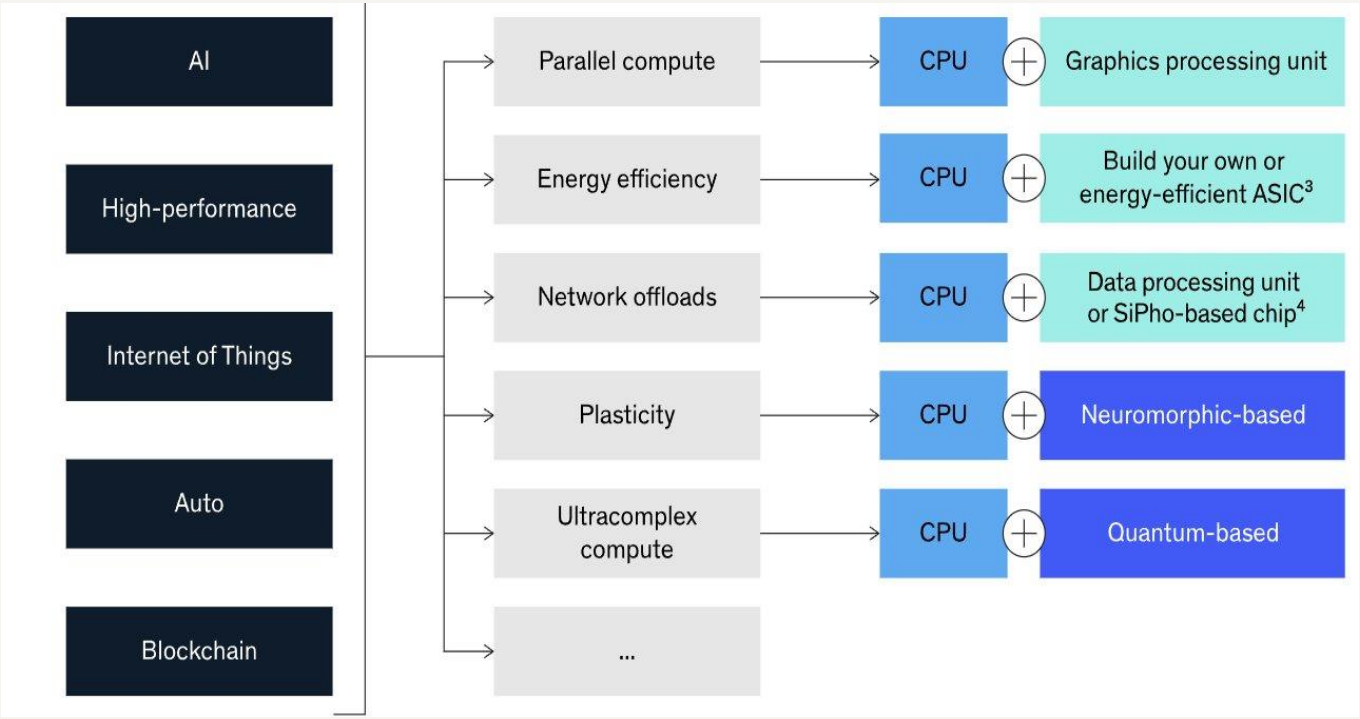
- Processor's performance improvement per year decreased to 3 % after 2015

Any Room for Improvement ?



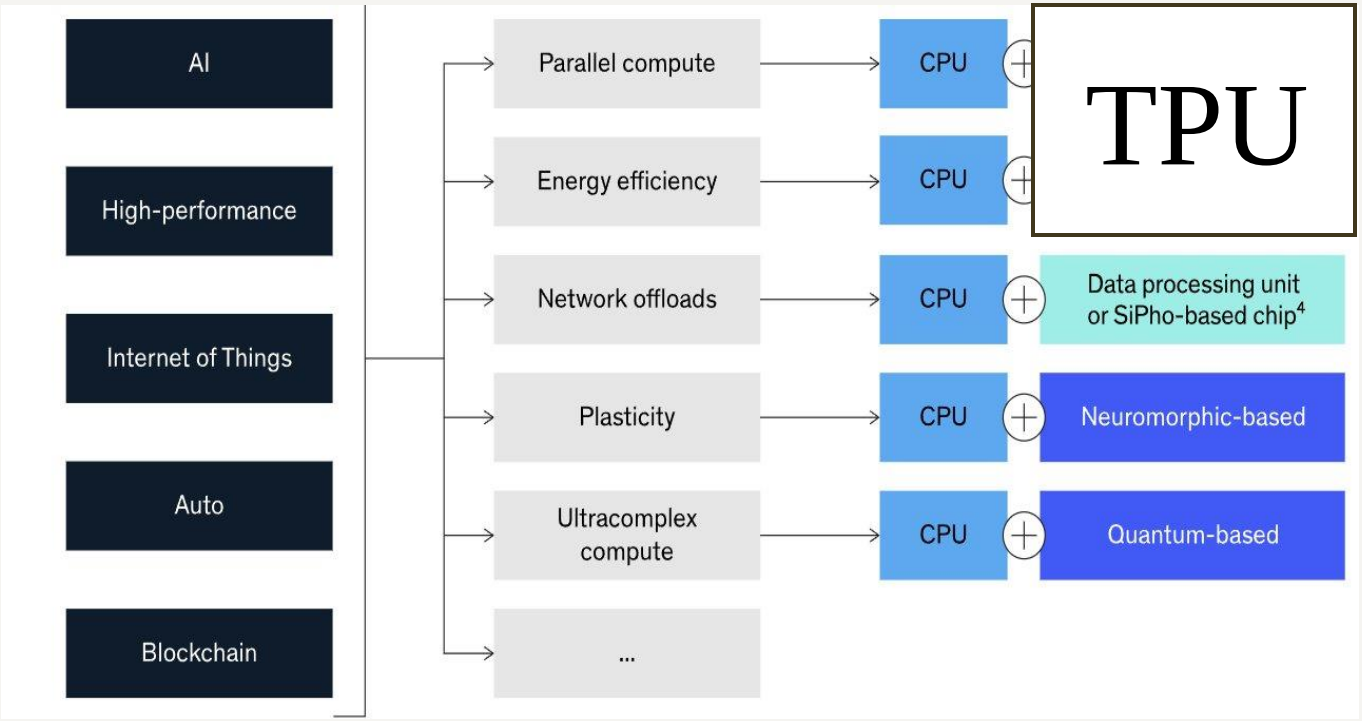
Introduction to Domain Specific Architectures (DSA)

- Specific application domain
- Use dedicated memories
- Use parallelism
- Use simplest data types
- Use domain specific languages



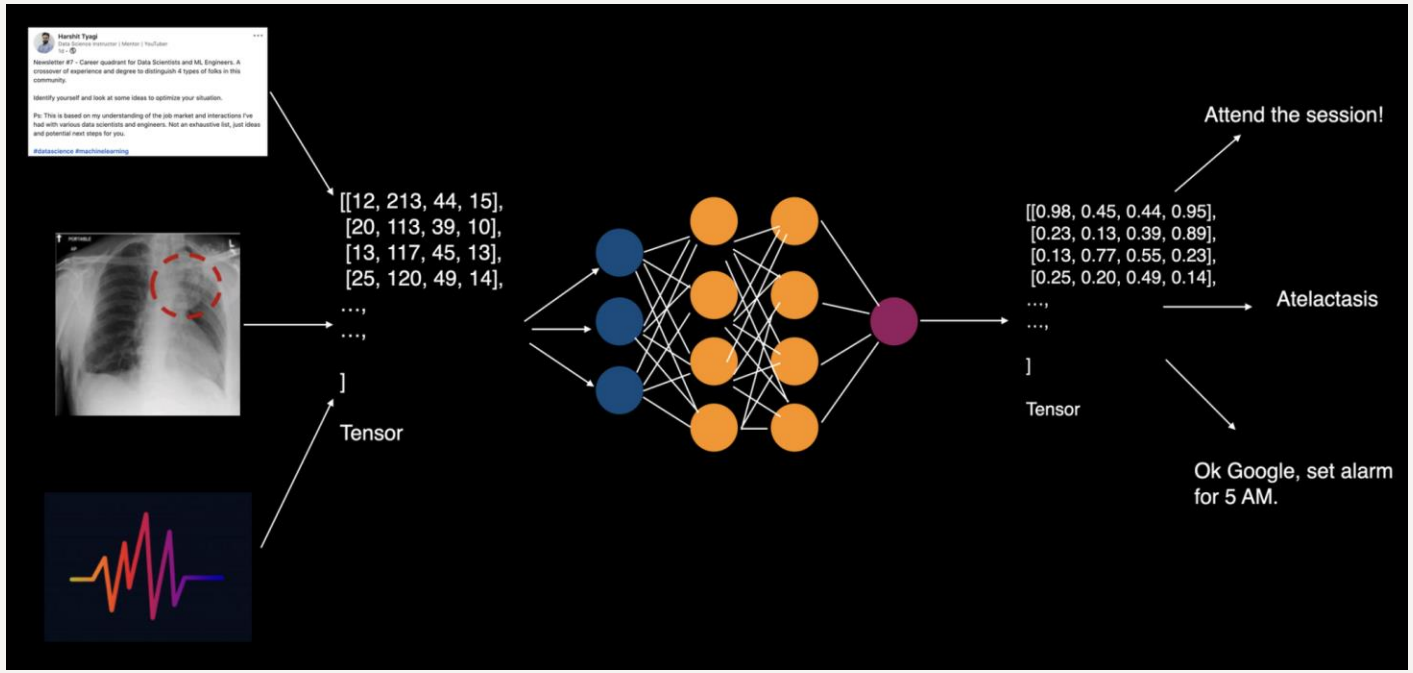
Introduction to Domain Specific Architectures (DSA)

- Google introduced Tensor Processing Units (TPU) in 2015
- Matrix processors specialized for neural network workloads



Why we need TPU ?

- Fundamental operation in deep learning is matrix multiplication (tensor operations)
- General purpose processors (CPUs) and GPUs are not optimized for tensor operation
- High power consumption and lower efficiency for ML tasks
- Need for a hardware that is power efficient, scalable and cost-effective



Popular Neural Networks (NNs)

- Multilayer Perceptron (MLP)
- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)

Name	*LOC*	*Layers*					*Nonlinear function*	*Weights*	*TPU Ops / Weight Byte*	*TPU Batch Size*	*% of Deployed TPUs in July 2016*
		FC	*Conv*	*Vector*	*Pool*	*Total*					
MLP0	100	5				5	ReLU	20M	200	200	61%
MLP1	1000	4				4	ReLU	5M	168	168	
LSTM0	1000	24		34		58	sigmoid, tanh	52M	64	64	29%
LSTM1	1500	37		19		56	sigmoid, tanh	34M	96	96	
CNN0	1000		16			16	ReLU	8M	2888	8	5%
CNN1	1000	4	72		13	89	ReLU	100M	1750	32	

= 95 %

TPU Origin

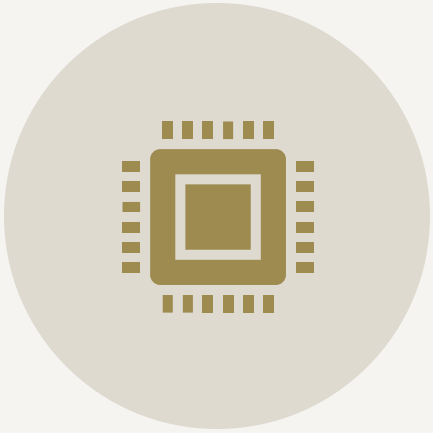
Projection

If people use voice search 3 minutes per day, computation demand needs to be doubled

Goal

Improve cost-performance by 10x over GPUs.

Result



TPU was designed, verified, built, and deployed in Google data centers in just 15 months

TPU Architecture and Implementation

- Coprocessor on the PCIe I/O bus in existing servers
- Host server sends the TPU instruction
- Goal was to run whole inference models in the TPU to reduce interactions with host CPU.

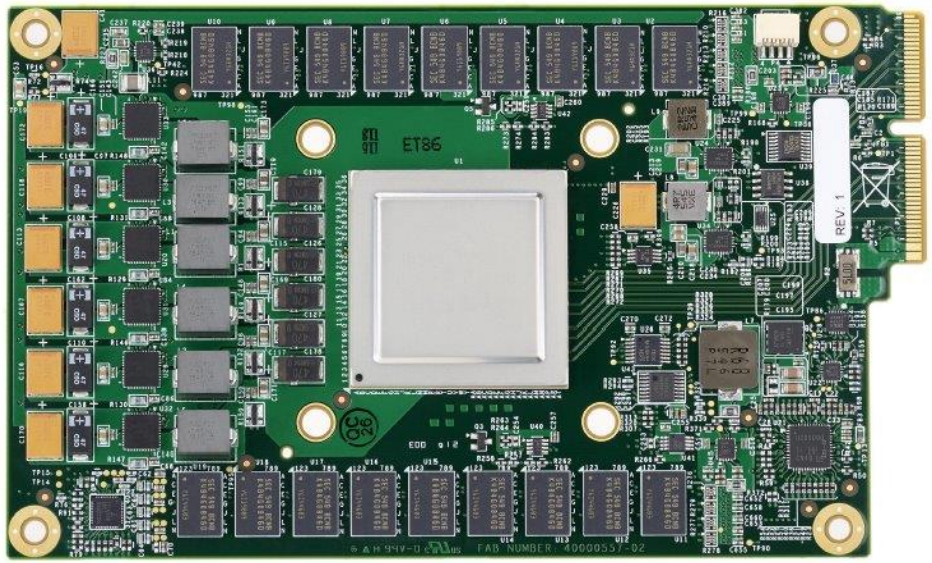


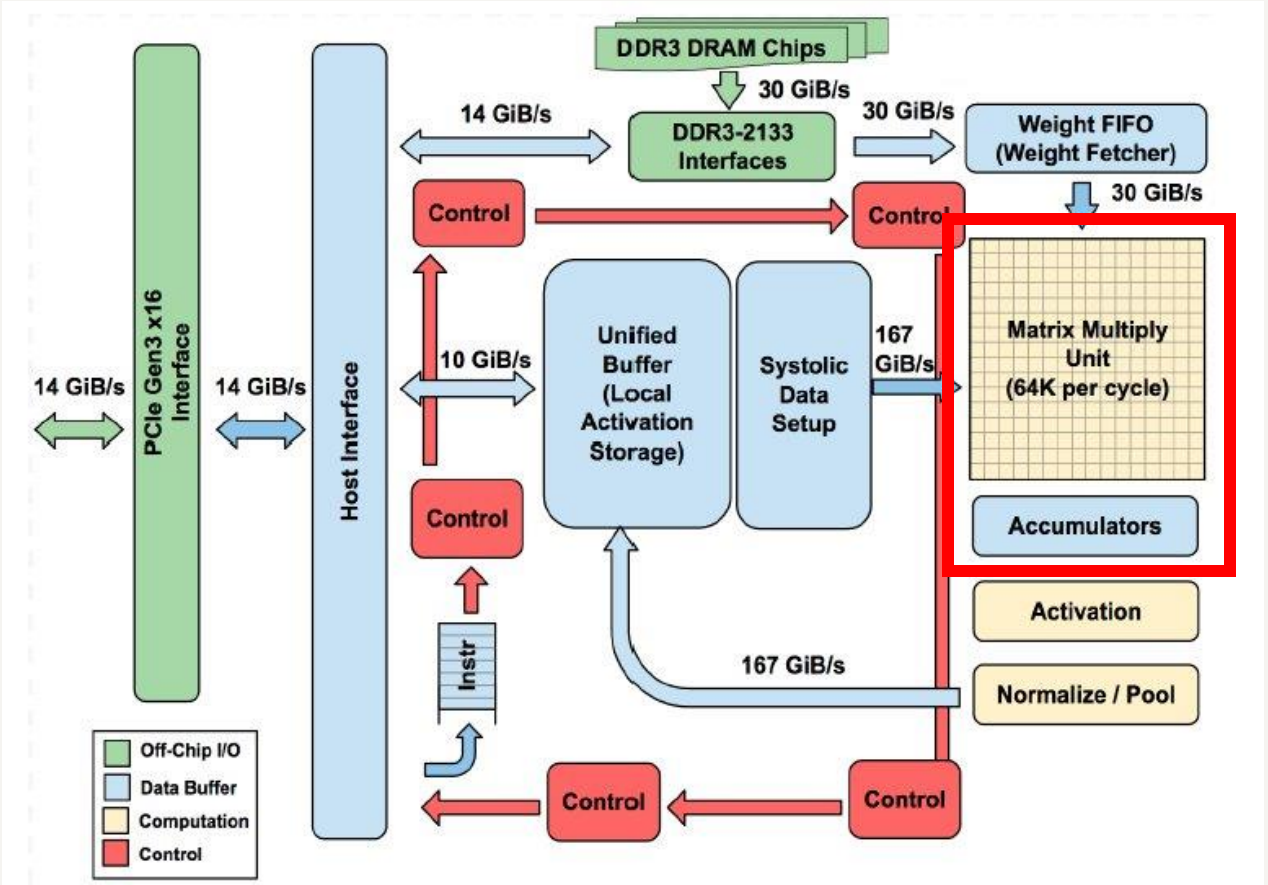
Figure 3. TPU Printed Circuit Board. It can be inserted into the slot for a SATA disk in a server.

TPU Architecture and Implementation

TPU High Level Architecture

- **Matrix Multiply Unit**
  - It contains 256x256 MACs(Multiply Accumulate Unit)
  - Performs 8-bit multiply-and-adds on integers
  - Holds one 64 KiB tile of weights plus one for double buffering

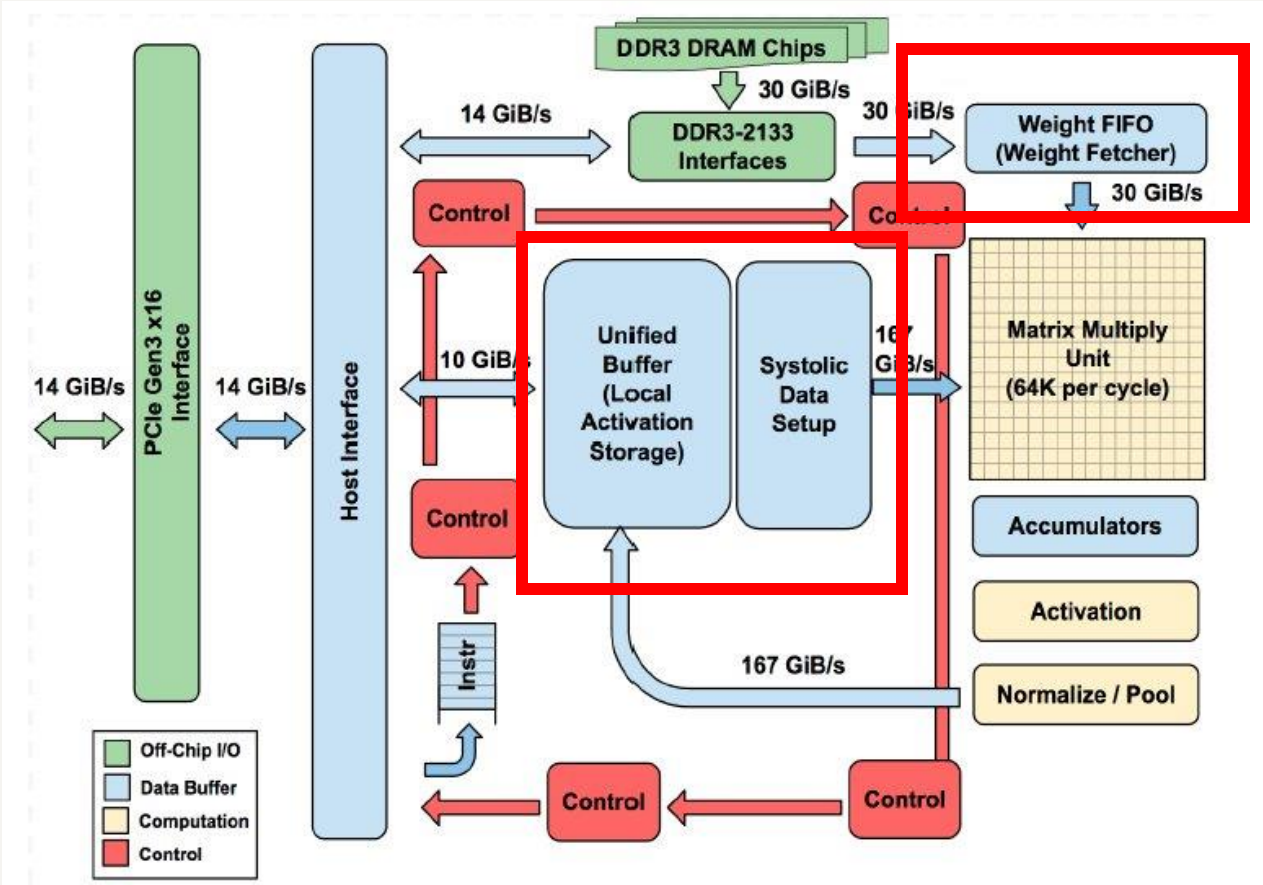
- **Accumulators (4 MiB)**
  - It stores 16-bit products from MMU
  - 32 bit
  - Divided into 4096 nodes each containing 256-element of 32-bit accumulators



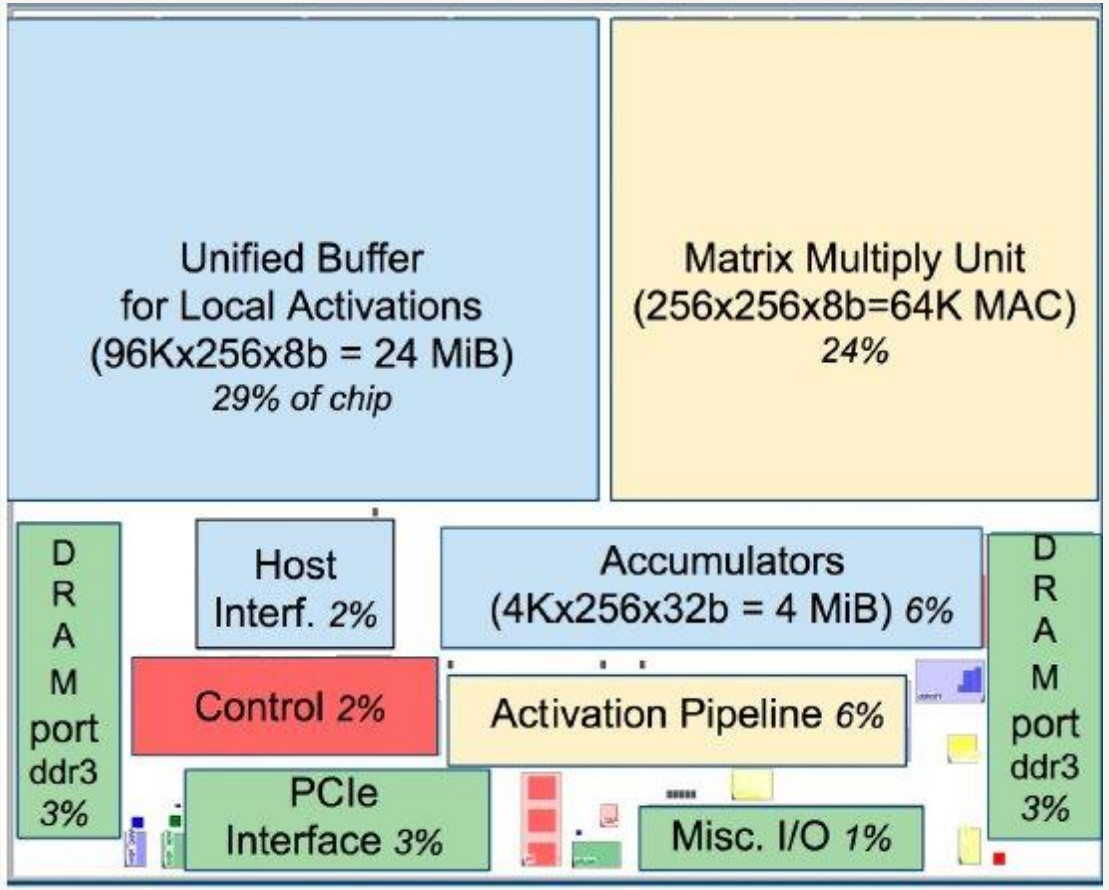
TPU Architecture and Implementation

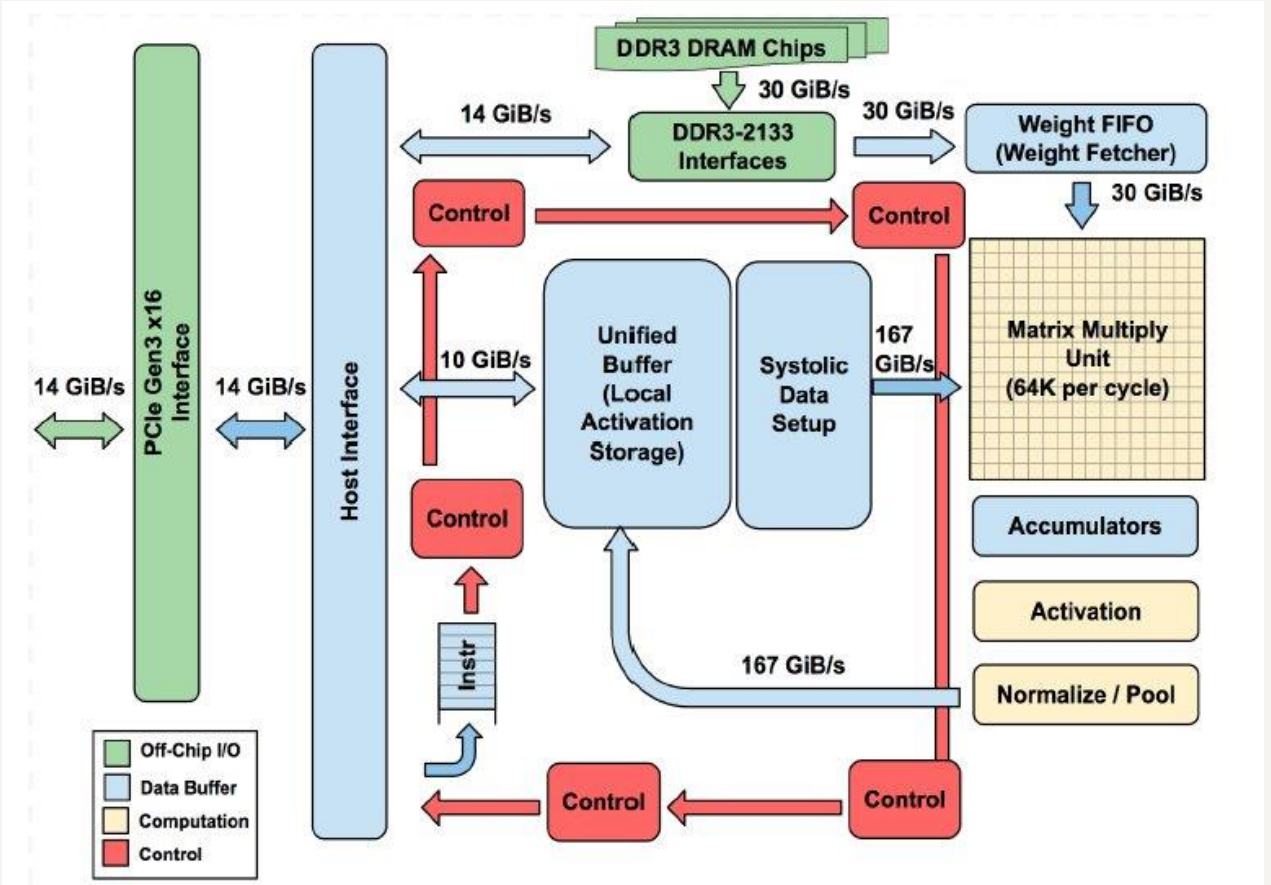
TPU High Level Architecture

- **Memory and Buffer**
  - Weights for the matrix unit are staged through on-chip **Weight FIFO** that reads from an off-chip 8 GiB weight memory
  - Intermediate results are held in the 24 MiB on chip **Unified Buffer**



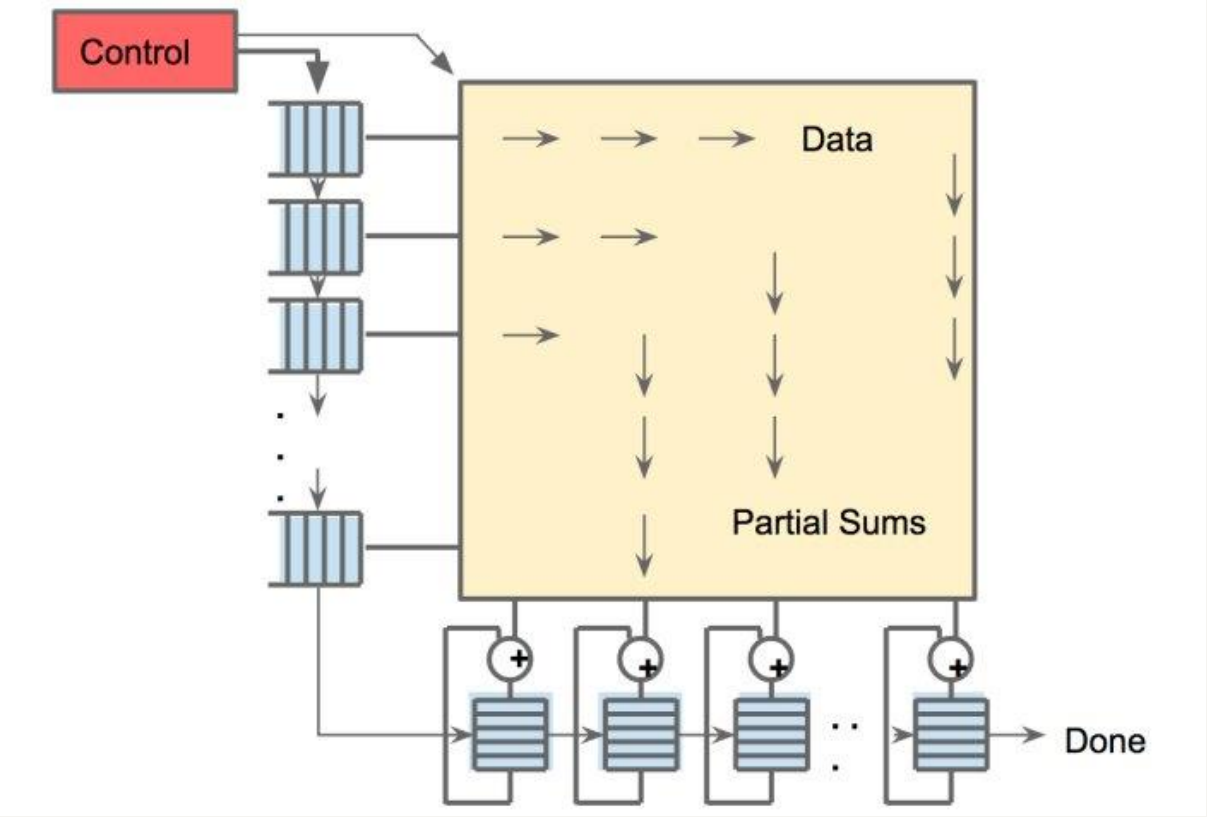
Floor Plan of TPU Die



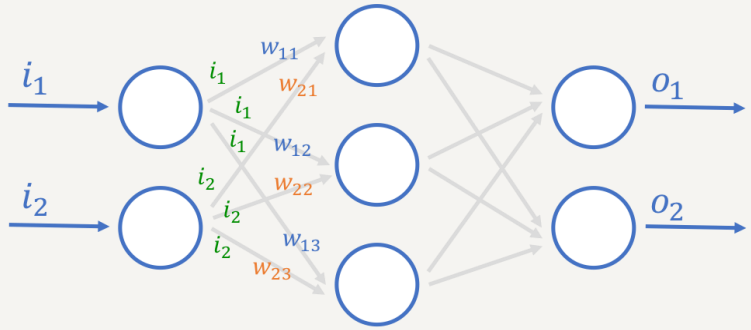


TPU Systolic Array

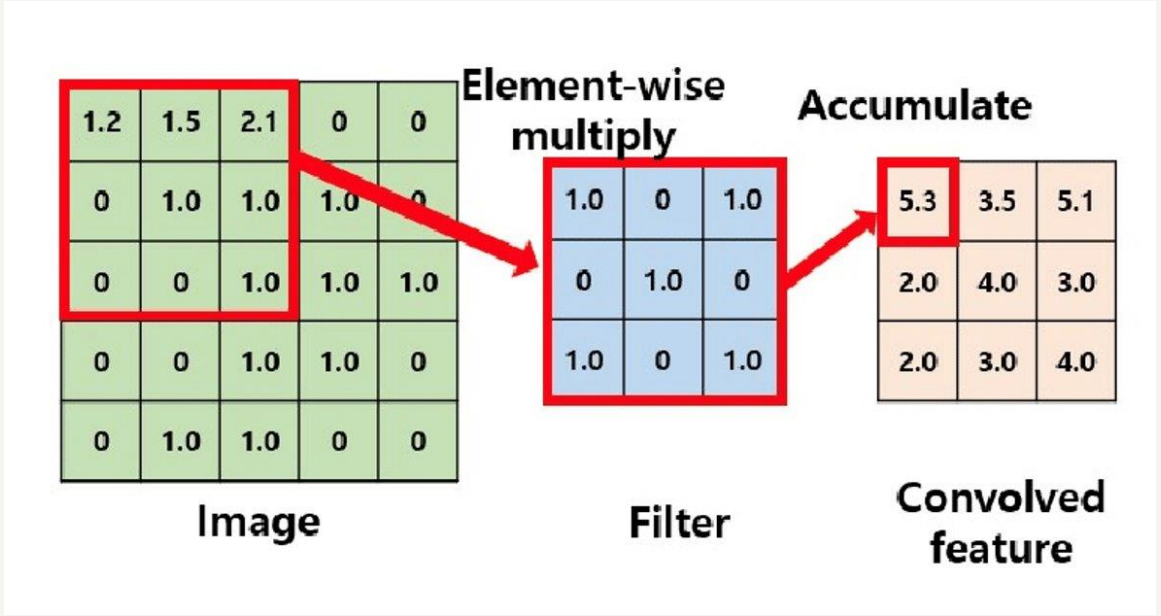
- Weights are loaded from the top and the input data flows into the array from the left
- Weights are preloaded in the systolic array



Matrix Multiplication Implementation



$$\begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \cdot \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} (w_{11} \times i_1) + (w_{21} \times i_2) \\ (w_{12} \times i_1) + (w_{22} \times i_2) \\ (w_{13} \times i_1) + (w_{23} \times i_2) \end{bmatrix}$$



Multilayer Perceptron (MLP)

Convolutional Neural Network

Basic operation is matrix multiplication

Matrix Multiplication Implementation

Let's take an example of multiplying two matrices: A (input) and B (weight)

a11	a12	a13
a21	a22	a23
a31	a32	a33

X

b11	b12	b13
b21	b22	b23
b31	b32	b33

=

c11	c12	c13
c21	c22	c23
c31	c32	c33

c11 = a11*b11 + a12*b21 + a13*b31

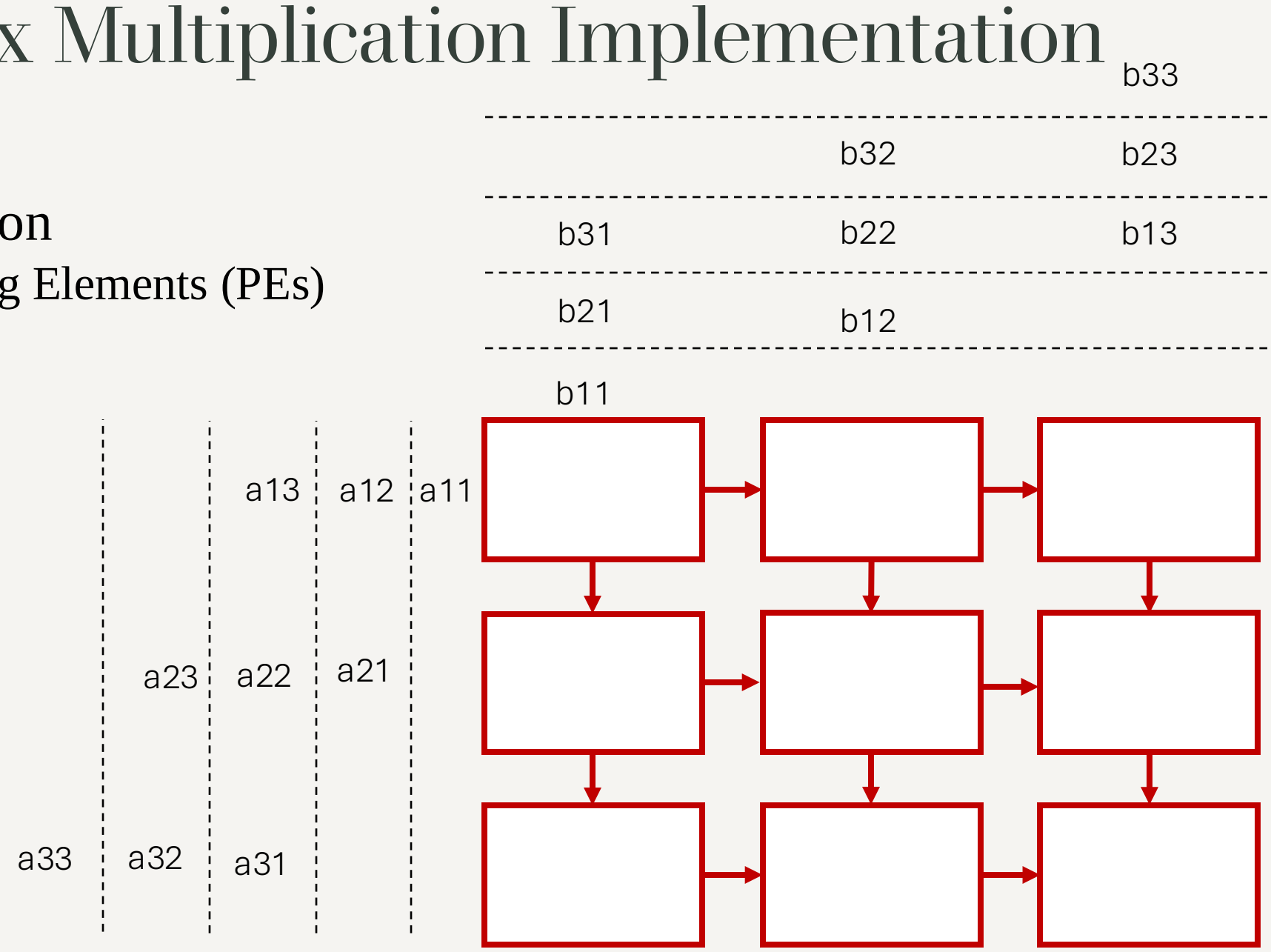
Using brute force approach:

Total operations needed to obtain C : 45 (add and multiply)

Matrix Multiplication Implementation

Systolic Execution

- 3 x 3 Processing Elements (PEs)

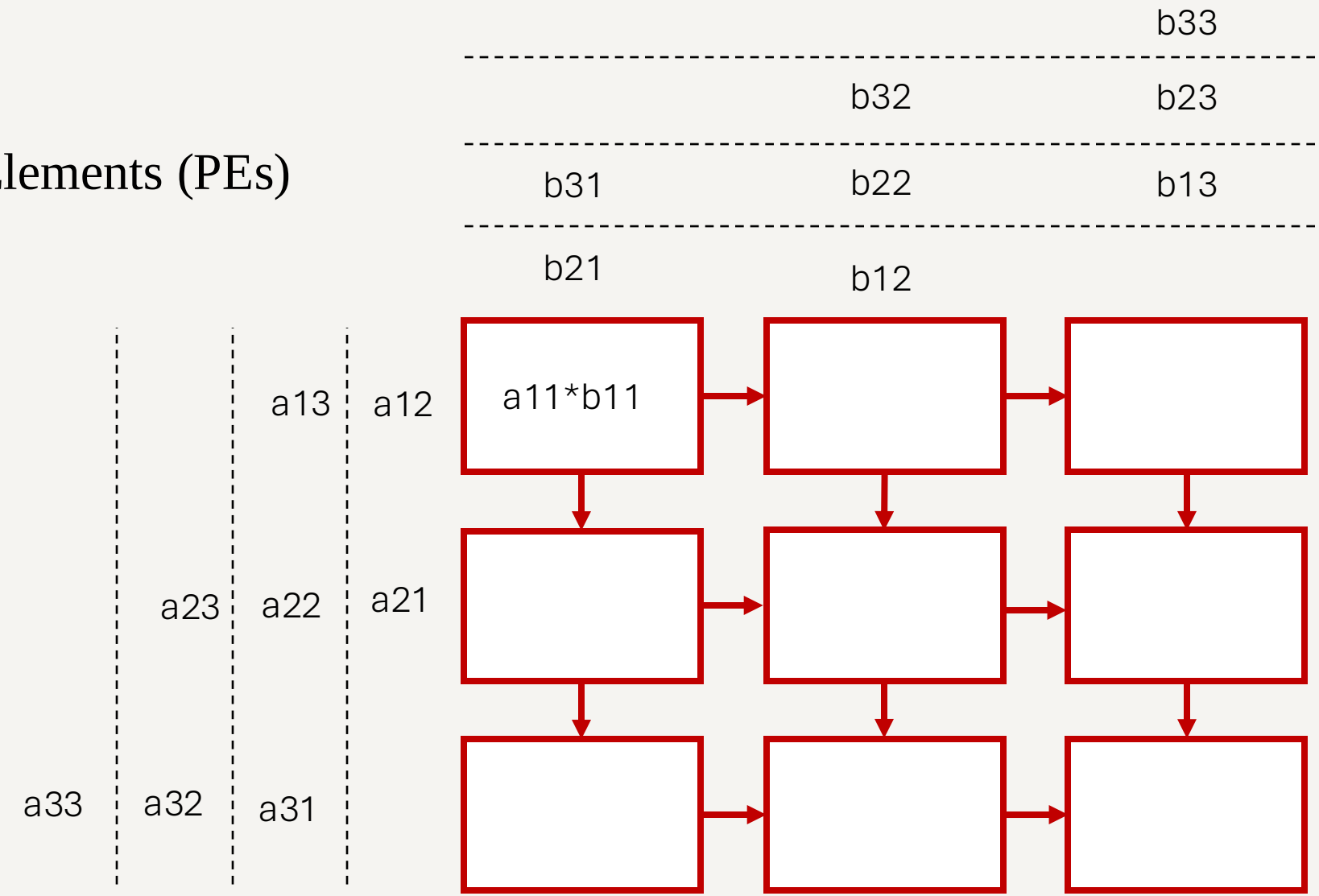


Matrix Multiplication Implementation

Systolic Execution

- 3 x 3 Processing Elements (PEs)

Cycle 1

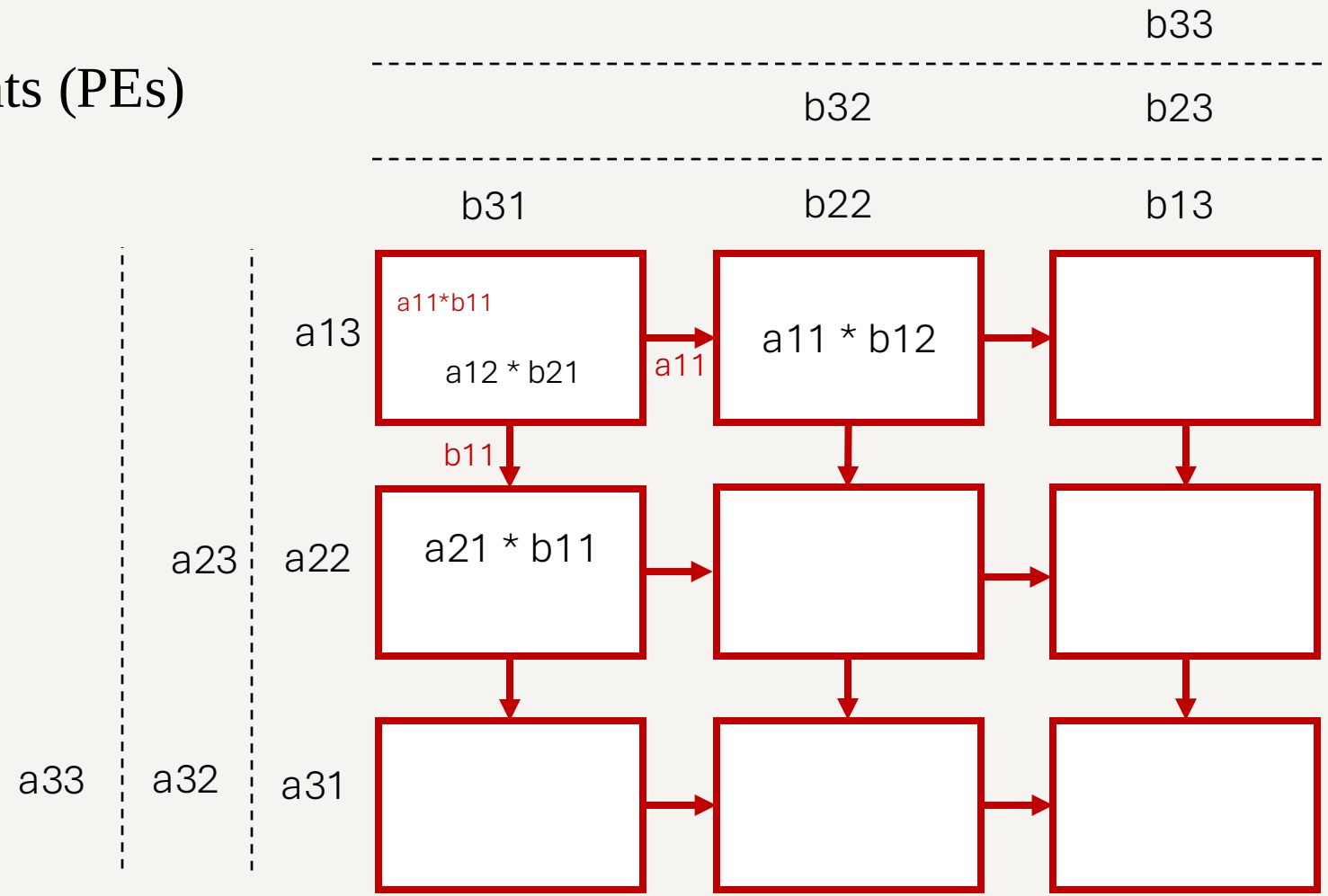


Matrix Multiplication Implementation

Systolic Execution

- 3 x 3 Processing Elements (PEs)

Cycle 2

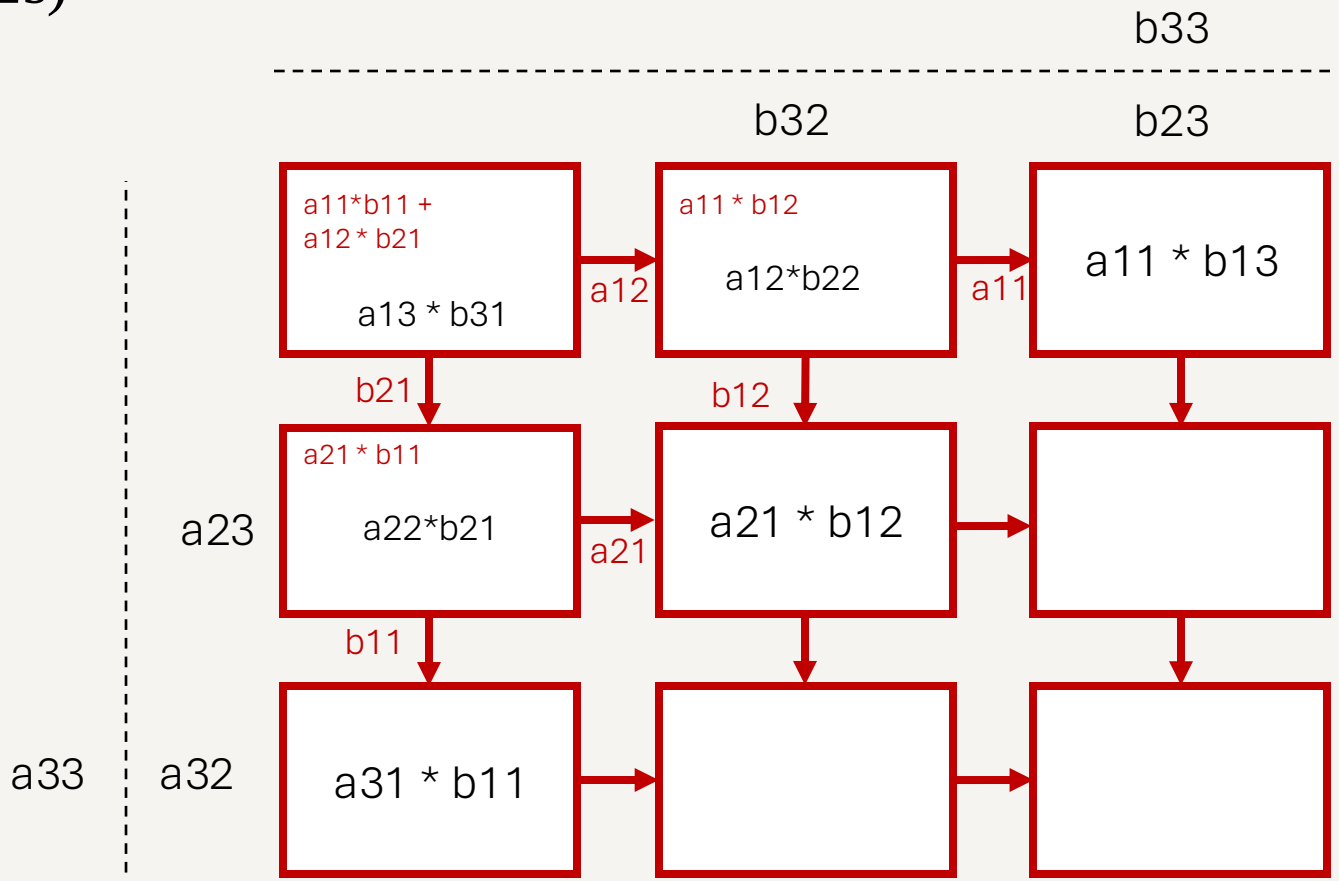


Matrix Multiplication Implementation

Systolic Execution

- 3 x 3 Processing Elements (PEs)

Cycle 3



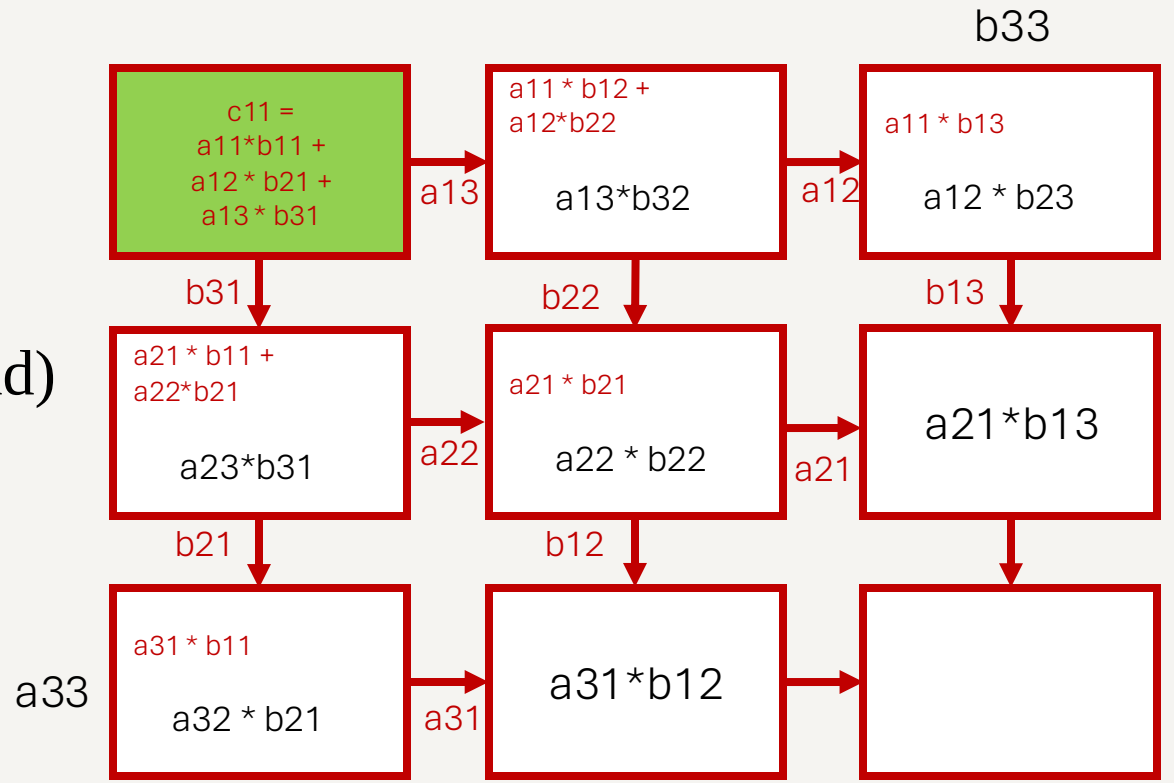
Matrix Multiplication Implementation

Systolic Execution

- 3 x 3 Processing Elements (PEs)

Cycle 4

Total operation = 14 (multiply and add)



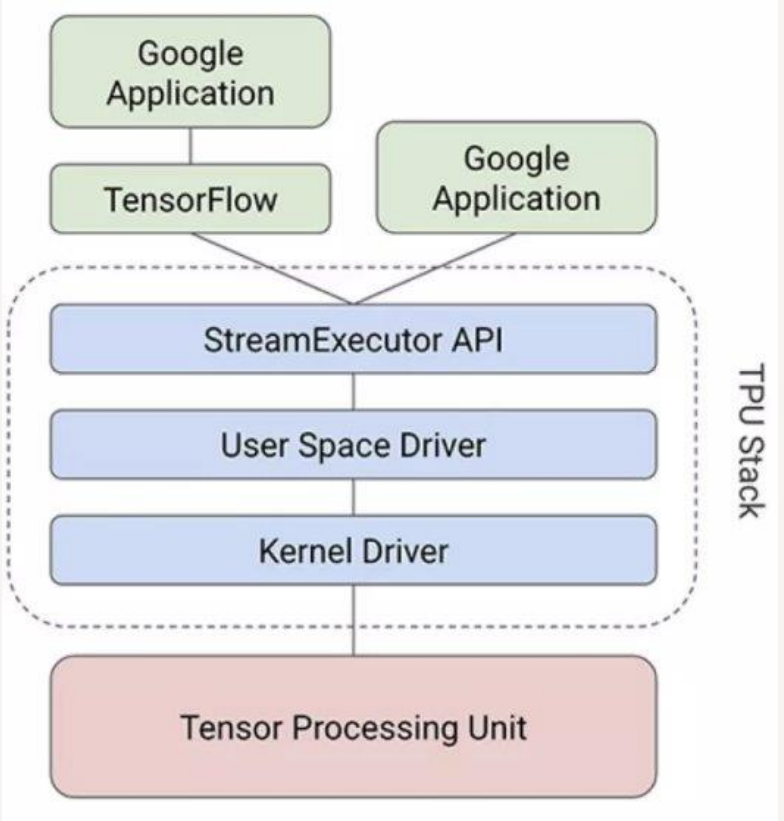
TPU Instructions

Follows CISC tradition; average CPI is typically 10 to 20

Instruction	Function
Read_Host_Memory	reads data from CPU memory into Unified Buffer
Read_Weights	reads weights from Weight Memory into Weight FIFO
MatrixMultiply/Convolve	multiply with data and weights
Activate	apply activation functions
Write_Host_Memory	writes data from the Unified Buffer into the CPU host memory.

Software Stack

- Divided into User Space driver and a Kernel driver
- Kernel driver handles memory management and interrupts
- User Space:
  - Sets up and controls TPU execution
  - Reformats data into TPU order
  - Translates API calls into TPU instructions



Experiment

Experiment Set-up

CPU, GPU and TPU platforms

Model	Die										Benchmarked Servers				
	mm^2	nm	MHz	TDP	Measured		TOPS/s		GB/s	On-Chip Memory	Dies	DRAM Size	TDP	Measured	
					Idle	Busy	8b	FP						Idle	Busy
Haswell E5-2699 v3	662	22	2300	145W	41W	145W	2.6	1.3	51	51 MiB	2	256 GiB	504W	159W	455W
NVIDIA K80 (2 dies/card)	561	28	560	150W	25W	98W	--	2.8	160	8 MiB	8	256 GiB (host) + 12 GiB x 8	1838W	357W	991W
TPU	<331*	28	700	75W	28W	40W	92	--	34	28 MiB	4	256 GiB (host) + 8 GiB x 4	861W	290W	384W

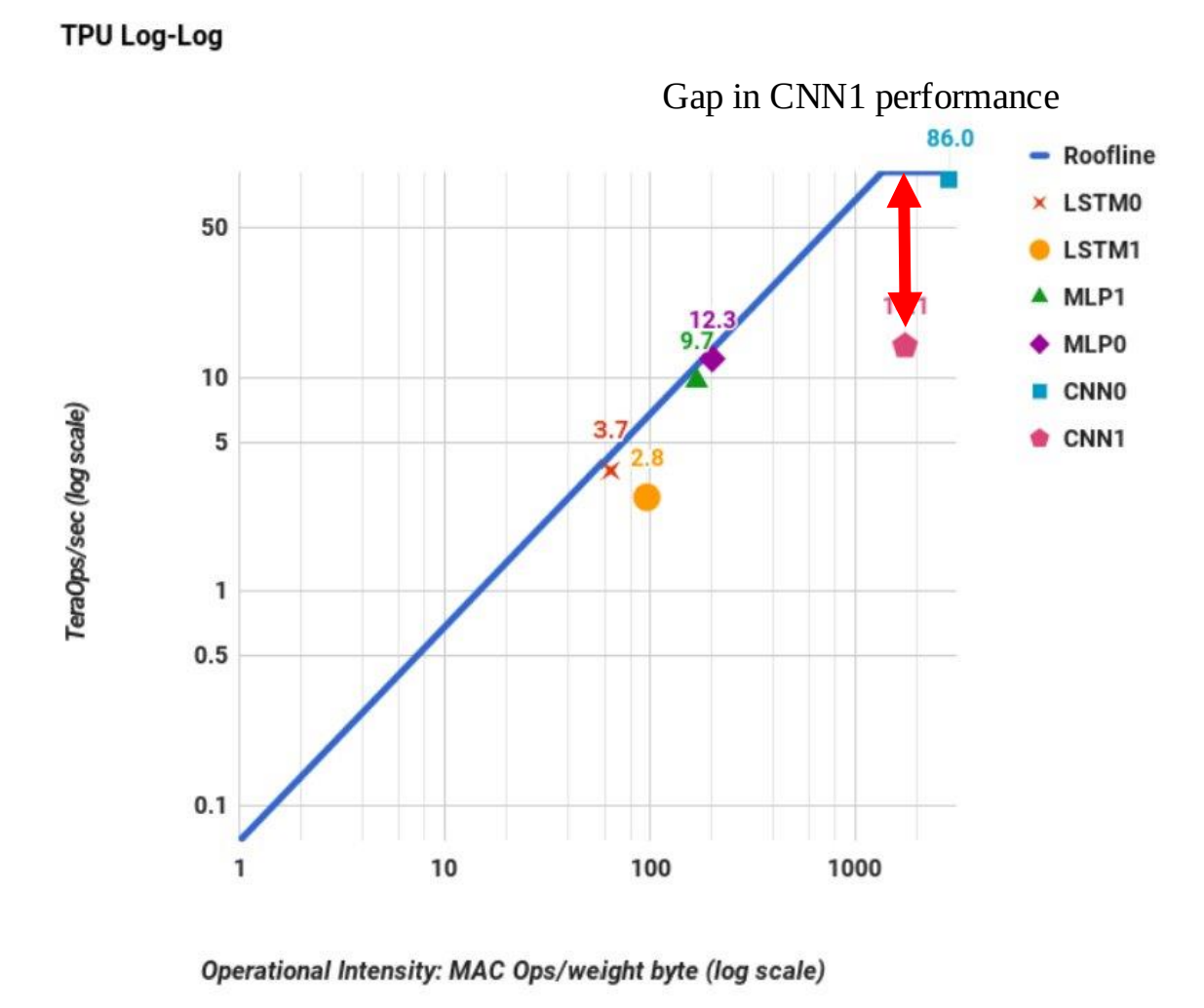
Six NN applications

Name	LOC	Layers					Nonlinear function	Weights	TPU Ops / Weight Byte	TPU Batch Size	% of Deployed TPUs in July 2016
		FC	Conv	Vector	Pool	Total					
MLP0	100	5				5	ReLU	20M	200	200	61%
MLP1	1000	4				4	ReLU	5M	168	168	
LSTM0	1000	24		34		58	sigmoid, tanh	52M	64	64	29%
LSTM1	1500	37		19		56	sigmoid, tanh	34M	96	96	
CNN0	1000		16			16	ReLU	8M	2888	8	5%
CNN1	1000	4	72		13	89	ReLU	100M	1750	32	

Experiment: Roofline Performance Model

- To figure out the performance bottlenecks (computation limited or memory bandwidth limited)
- Gap shows the potential improvement

- MLPS and LSTMs are memory bound while CNNs are compute bound.



Experiment: Roofline Performance Model

TPU Operations

Activity of the matrix unit based on hardware performance counters

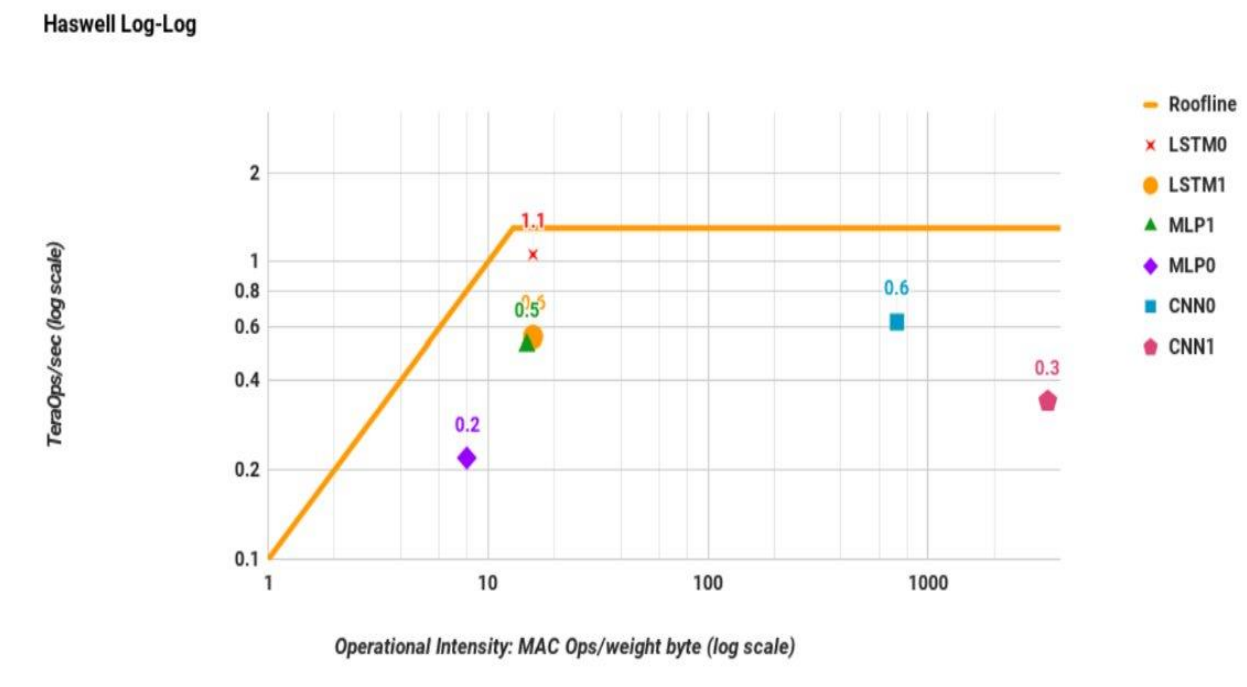
Application	*MLP0*	*MLP1*	*LSTM0*	*LSTM1*	*CNN0*	*CNN1*	*Mean*	*Row*
Array active cycles	12.7%	10.6%	8.2%	10.5%	78.2%	46.2%	28%	1
Useful MACs in 64K matrix (% peak)	12.5%	9.4%	8.2%	6.3%	78.2%	22.5%	23%	2
Unused MACs	0.3%	1.2%	0.0%	4.2%	0.0%	23.7%	5%	3
Weight stall cycles	53.9%	44.2%	58.1%	62.1%	0.0%	28.1%	43%	4
Weight shift cycles	15.9%	13.4%	15.8%	17.1%	0.0%	7.0%	12%	5
Non-matrix cycles	17.5%	31.9%	17.9%	10.3%	21.8%	18.7%	20%	6
RAW stalls	3.3%	8.4%	14.6%	10.6%	3.5%	22.8%	11%	7
Input data stalls	6.1%	8.8%	5.1%	2.4%	3.4%	0.6%	4%	8
TeraOps/sec (92 Peak)	12.3	9.7	3.7	2.8	86.0	14.1	21.4	9

For CNN1

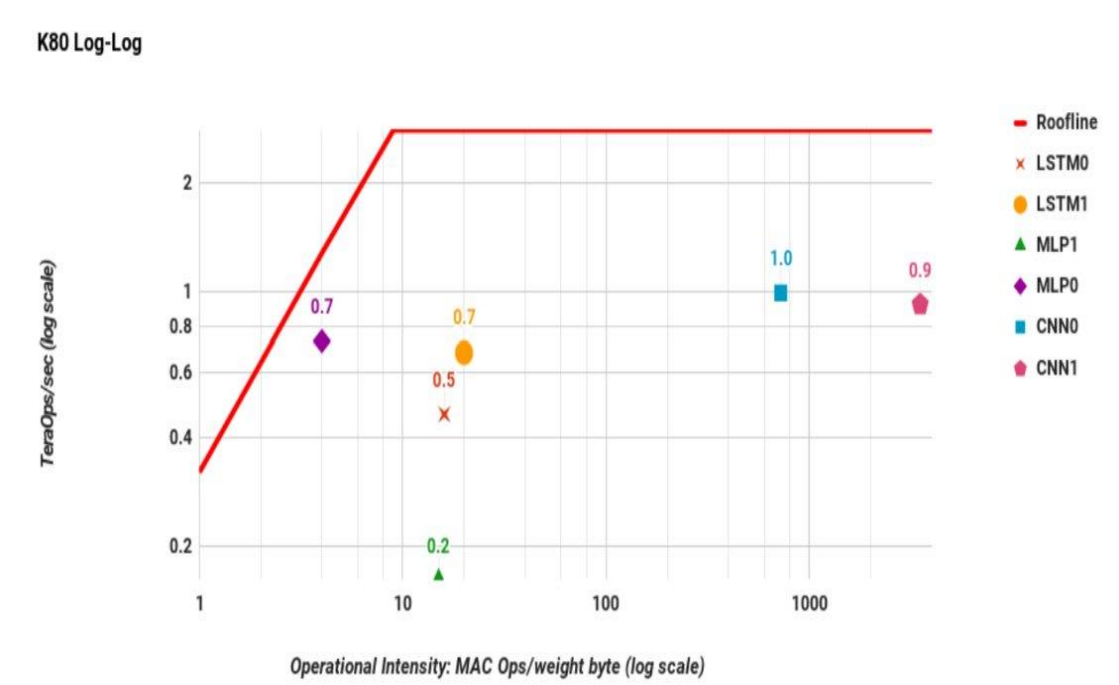
- TPU spends less than half of its cycles performing matrix operation
- Only half of MACs hold useful weights

Experiment: Roofline Performance Model

For Haswell CPU



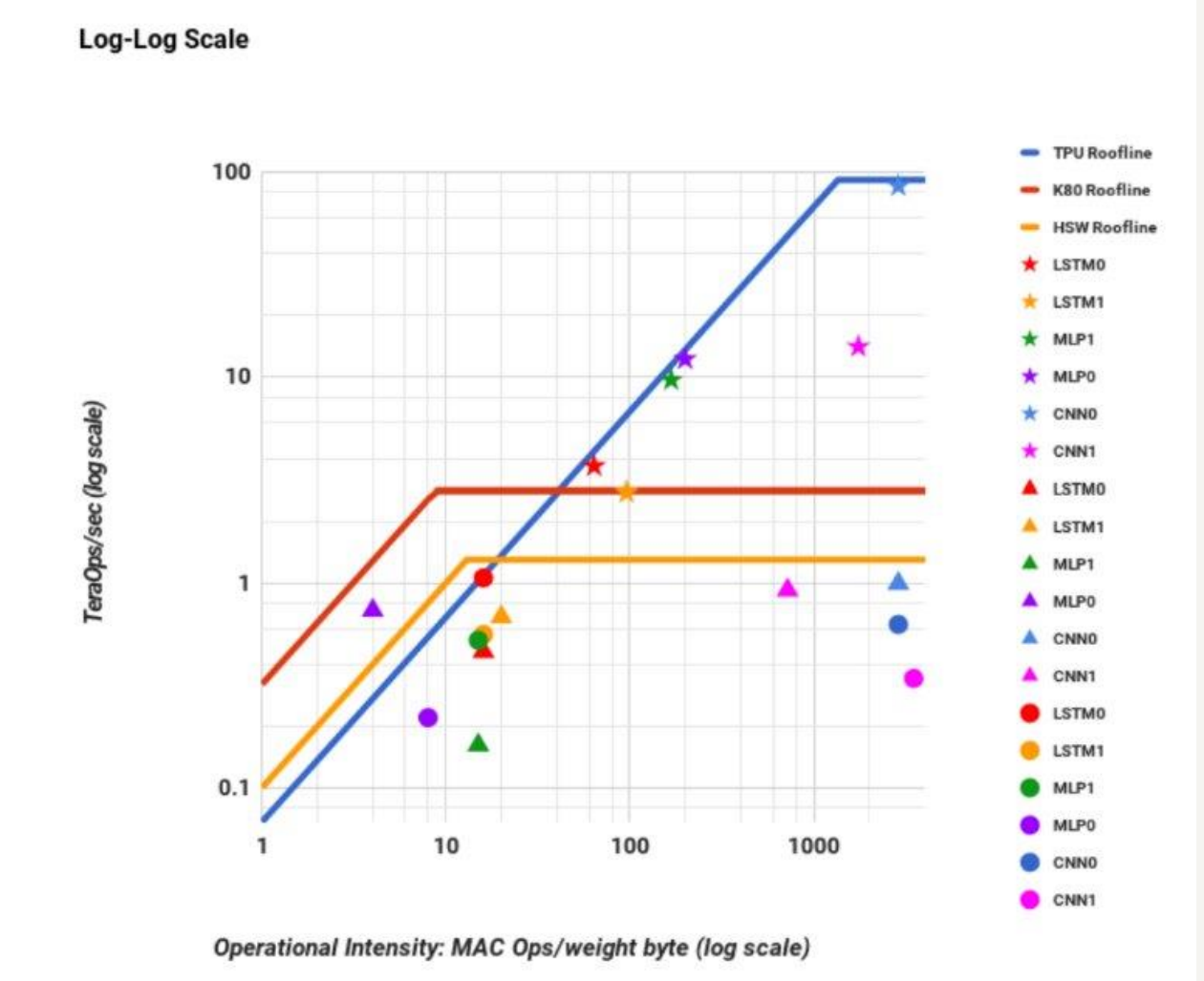
For K80 GPU



Experiment: Roofline Performance Model

Why so far below the rooflines?

- Response time
- Small increase in response time cause customer to use service less
- Inference prefers latency over throughput



Experiment: Response Time

Type	*Batch*	*99th% Response*	*Inf/s (IPS)*	*% Max IPS*
CPU	16	7.2 ms	5,482	42%
CPU	64	21.3 ms	13,194	100%
GPU	16	6.7 ms	13,461	37%
GPU	64	8.3 ms	36,465	100%
TPU	200	7.0 ms	225,000	80%
TPU	250	10.0 ms	280,000	100%

- Maximum allowable latency is 7 ms.
- If response time is bounded to 7 ms, CPUs and GPUs run 2.3x to 2.7x slower.
- But for TPU, the slowdown is just 1.2x.

Experiment: Inference performance comparison

Performance improvement of GPUs and TPUs over CPU

Type	*DNN*		*LSTM*		*CNN*		*GM*	*WM*
	0	*1*	*0*	*1*	*0*	*1*		
GPU	2.5	0.3	0.4	1.2	1.6	2.7	1.1	1.9
TPU	41.0	18.5	3.5	1.2	40.3	71.0	14.5	29.2
Ratio	16.7	60.0	8.0	1.0	25.4	26.3	13.2	15.3

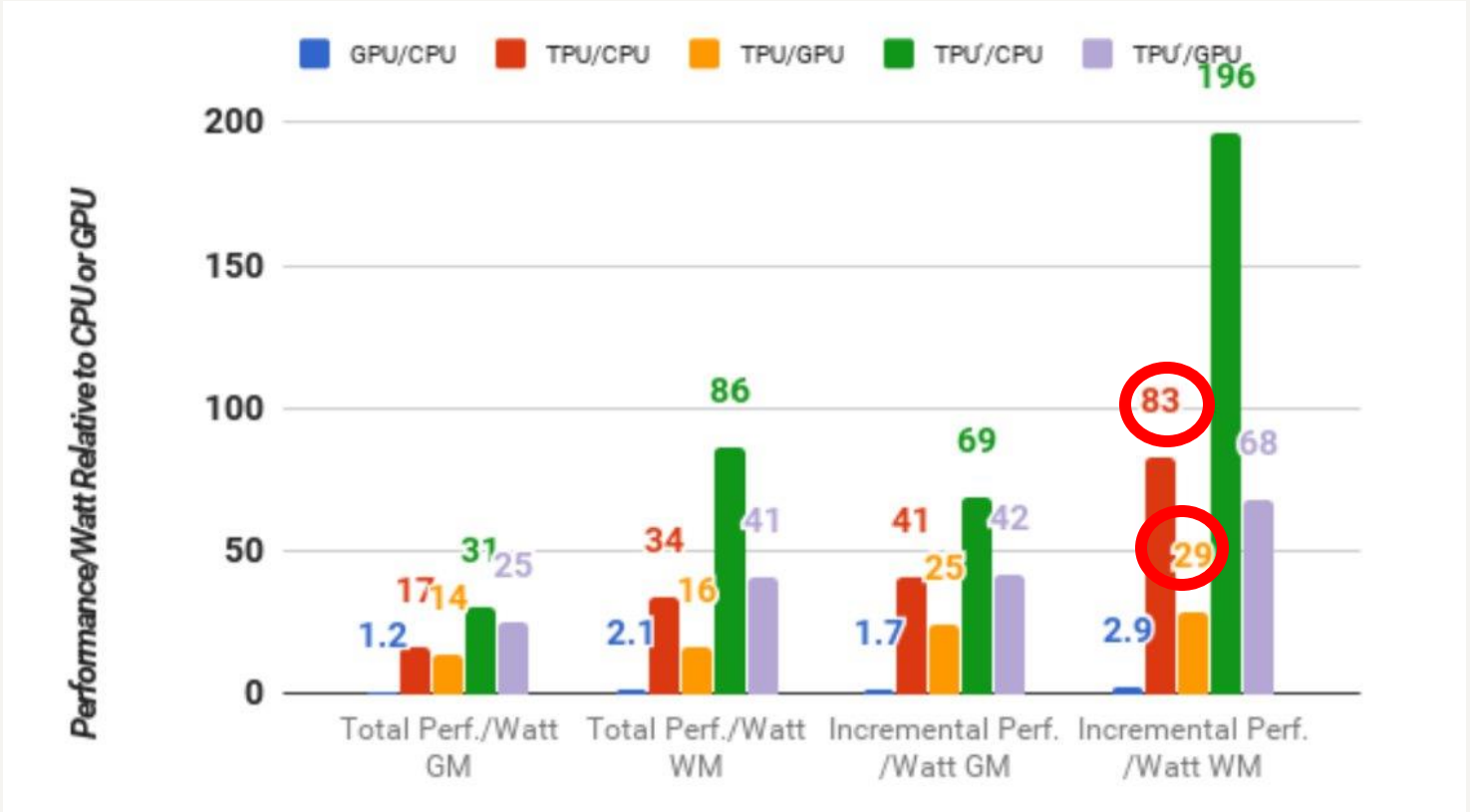
- GPU is 1.9 times faster than CPU
- TPU is 29.2 times faster than CPU
- TPU is 15.3 times faster than GPU

Experiment: Performance/Watt

To evaluate the efficiency of processors

- 83x incremental Perf./watt of Haswell CPU
- 29x incremental Perf./watt of K80 GPU

TPU' is an improved TPU that uses GDDR5 memory



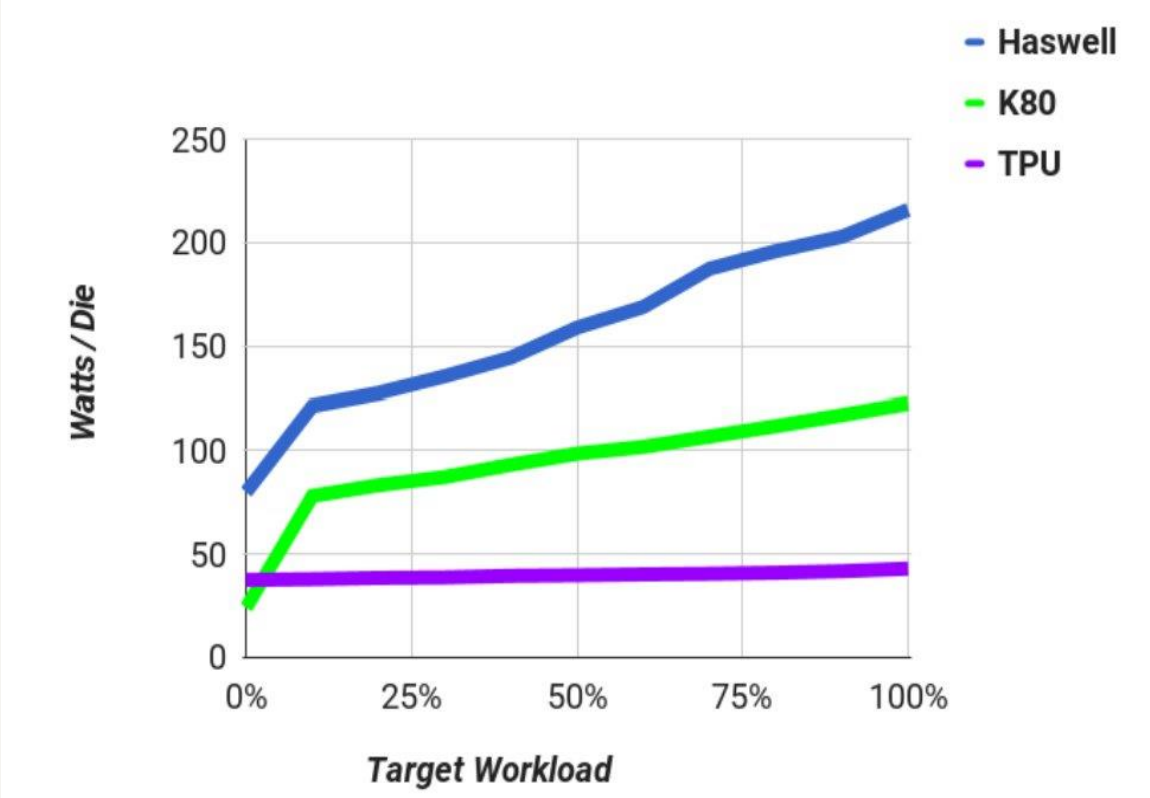
- Total: includes power consumed by host CPU when calculating P/W of GPUs and CPUs
- Incremental : subtract host CPU server power

Experiment: Energy Proportionality

Servers should consume power proportional to the amount of work performed.

- TPU has poor energy proportionality; At 10% workload, it uses 88% of the power it uses at 100%

- For Haswell = 56%
- For K80 = 66%



Contributions

- Introduces Tensor Processing Unit (TPU), an ASIC designed to accelerate machine learning inference tasks
- TPU is 15x to 30x faster than contemporary GPUs and CPUs for inference tasks
- TPU achieves 30X to 80X better performance/watts compared to GPUs and CPUs
- Demonstrates the advantage of hardware tailored for neural networks, introducing Domain Specific Architectures for AI task.
- Provides data-driven analysis from Google’s production-scale inference workloads.

Limitations

- Designed only for inference workloads, not for training deep learning models
- Evaluation only conducted for Google workloads which limits the generalization over other applications
- Lacks flexibility for non-neural network tasks in comparison to GPUs

Recent Trends in TPUs

	TPUv1	TPUv2	TPUv3	TPUv4[15][17]	TPUv5e[18]	TPUv5p[19] [20]	Trillium[21]
Date introduced	2015	2017	2018	2021	2023	2023	2024
Process node	28 nm	16 nm	16 nm	7 nm	Unstated	Unstated	
Die size (mm^2)	331	< 625	< 700	< 400	300-350	Unstated	
On-chip memory (MiB)	28	32	32	32	48	112	
Clock speed (MHz)	700	700	940	1050	Unstated	1750	
Memory	8 GiB DDR3	16 GiB HBM	32 GiB HBM	32 GiB HBM	16 GB HBM	95 GB HBM	32 GB ?
Memory bandwidth	34 GB/s	600 GB/s	900 GB/s	1200 GB/s	819 GB/s	2765 GB/s	~1.6 TB/s ?
TDP (W)	75	280	220	170	Not Listed	Not Listed	
TOPS (Tera Operations Per Second)	23	45	123	275	197 (bf16) 393 (int8)	459 (bf16) 918 (int8)	
TOPS/W	0.31	0.16	0.56	1.62	Not Listed	Not Listed	

https://en.wikipedia.org/wiki/Tensor_Processing_Unit

THANK YOU