In-Datacenter Performance Analysis of A Tensor Processing Unit

DONG-HYUN HWANG

Preview of Highlights

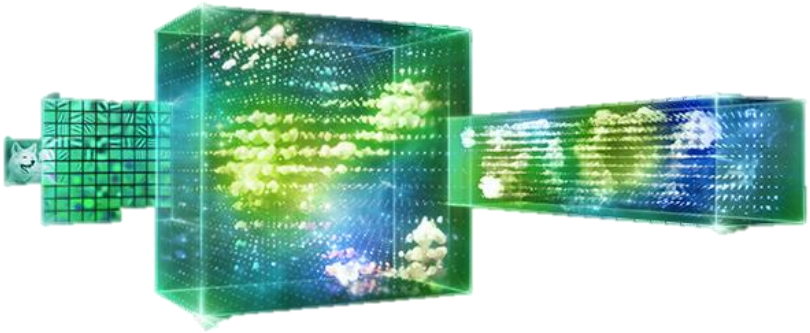
- Inference apps usually emphasize response-time over throughput since they are often user-facing.
- Due to latency limits, the K80 GPU is underutilized for inference, and is just a little faster than a Haswell CPU.
- Despite having a much smaller and lower power chip, the TPU has 25 times as many MACs and 3.5 times as much on-chip memory as the K80 GPU.
- The TPU is about 15X - 30X faster at inference than the K80 GPU and the Haswell CPU.

Preview of Highlights(cont.)

- Four of the six NN apps are memory-bandwidth limited on the TPU; if the TPU were revised to have the same memory system as the K80 GPU, it would be about 30X – 50X faster than the GPU and CPU.
- The performance/Watt of the TPU is 30X – 80X that of contemporary products; the revised TPU with K80 memory would be 70X – 200X better.
- While most architects have been accelerating CNNs, they represent just 5% of our datacenter workload

Introduction

- The synergy between the large data sets in the cloud and the numerous computers that power it has enabled a renaissance in machine learning.
- Deep neural networks(DNN) have reduced word error rate in speech recognition, image recognition and beat a human champion at Go.



AlphaGo

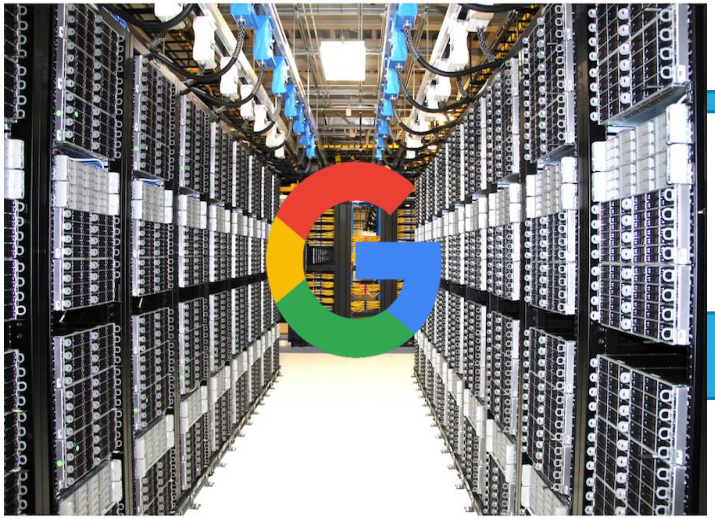
Introduction

- The “deep” part of DNN comes from going beyond a few layers and virtually all training today is in floating point, which is one reason GPUs have been so popular.
- Quantization transforms FP numbers into integers (often 8-bit) which are usually good enough for prediction (inference).
- 8-bit integer multiplies can be 6X less energy and area than 16-bit FP multiplies.
- Integer addition can be 13X less energy and 38X less area than 16-bit FP additions.
- TPU in this paper doesn’t support training.
  - TPU2 supports it.

NNs Are Popular Today

- Multi-Layer Perceptrons (MLP): fully connected NN
- Convolutional Neural Networks (CNN): Each ensuing layer is a set of nonlinear functions of weighted sums of spatially nearby subsets of outputs from the prior layer.
- Recurrent Neural Networks (RNN): Each subsequent layer is a collection of nonlinear functions of weighted sums of outputs and the previous state. The most popular RNN is Long Short-Term Memory (LSTM). The art of the LSTM is in deciding what to forget and what to pass on as state to the next layer. The weights are reused across time step

Main Purpose

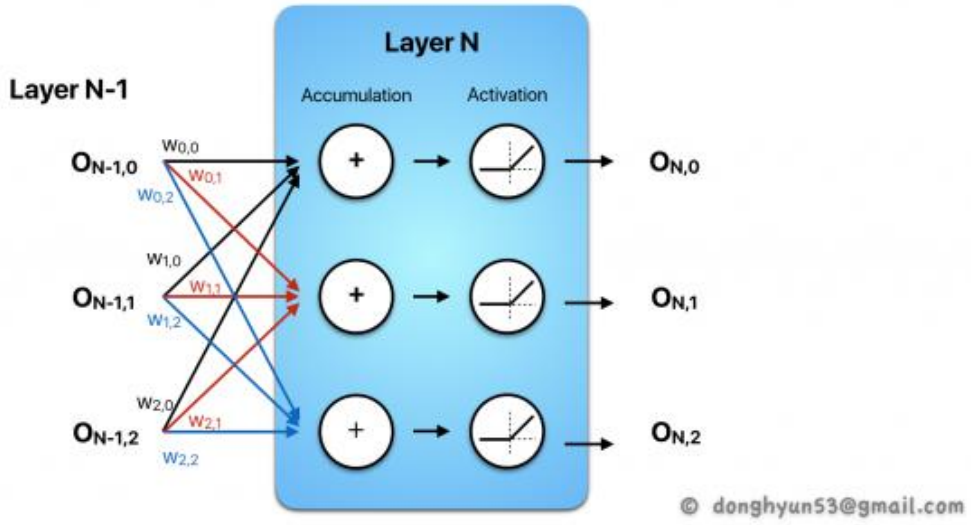


Training

Prediction

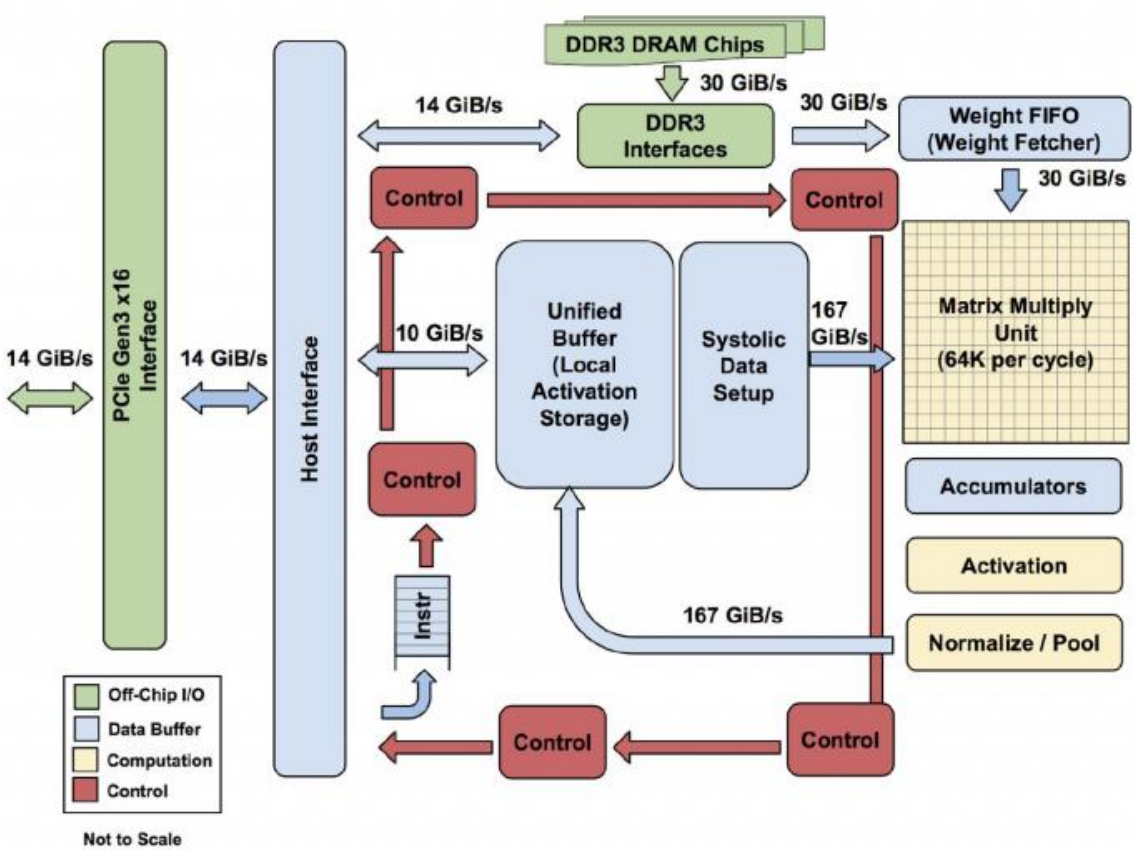
In the google data center, computation of prediction is more than computation of training.

Prediction



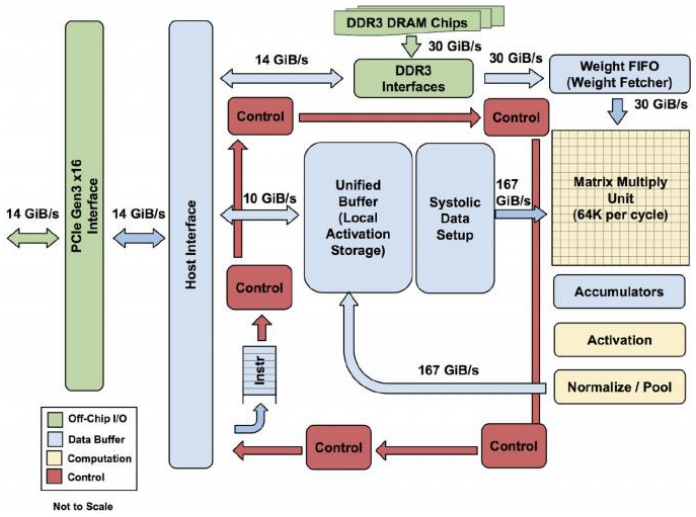
$$max\left(0, \begin{bmatrix} O_{N-1,0} & O_{N-1,1} & O_{N-1,2} \end{bmatrix} \bullet \begin{bmatrix} W_{0,0} & W_{0,1} & W_{0,2} \\ W_{1,0} & W_{1,1} & W_{1,2} \\ W_{2,0} & W_{2,1} & W_{2,2} \end{bmatrix}\right) = \begin{bmatrix} O_{N,0} & O_{N,1} & O_{N,2} \end{bmatrix}$$

Block Diagram



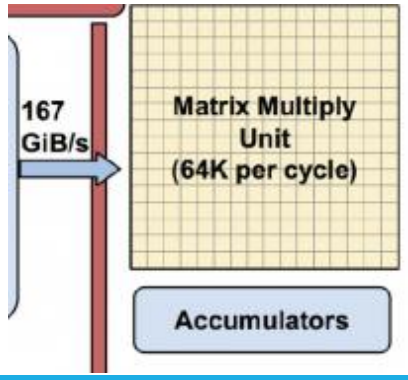
Architecture

- To reduce the delay of deployment, the TPU was designed to be a coprocessor on the PCIe I/O bus.
- The TPU instructions are sent from the host over the PCIe Gen3 x16 bus into an instruction buffer.
- Internal blocks are connected together by 256-byte-wide paths.



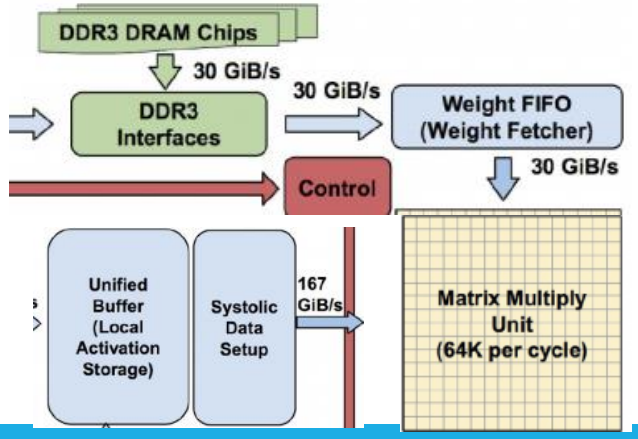
MMU

- Matrix Multiply Unit(MPU) is a heart of TPU.
  - It contains 256x256 MACs(Multiply ACcumulate unit).
    - Performing 8-bit multiply and adds on signed or unsigned integers.
      - Output is 16-bit data.
  - 16-bit products are collected in the 4MiB of 32-bit accumulators.
  - The 4MiB represents 4096 node.
    - Each node has 256-element of 32-bit accumulators.
  - The matrix unit produces one 256-element partial sum per clock cycle
  - The matrix unit holds one 64KiB tile of weights plus one for double buffering. (To hide the 256 cycles it takes to shit a tile in)
    - Single weight is 8-Bit.



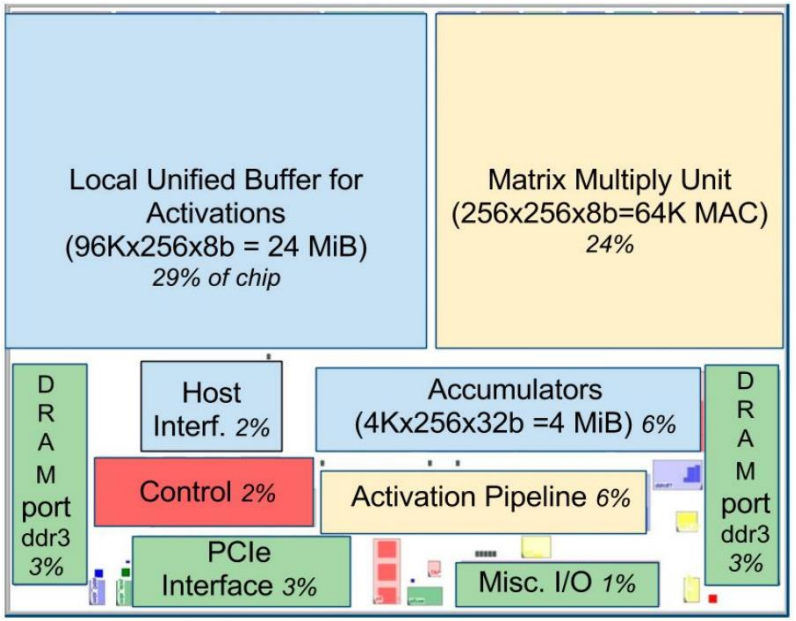
Memory and Buffer

- The weights for the matrix unit are staged through an on-chip Weight FIFO that reads from an off-chip 8 GiB DRAM called Weight Memory.
- The weight FIFO is four tiles deep.
  - The intermediate results are held in the 24 MiB on-chip Unified Buffer, which can serve as inputs to the Matrix Unit.



Architecture (cont.)

- Unified Buffer is almost a third (29%) of the die and the Matrix Multiply Unit is a quarter (24%).
- The 24 MiB size was picked in part to match the pitch of the Matrix Unit on the die.



Architecture (cont.)

- As instructions are sent over the relatively slow PCIe bus, TPU instructions follow the CISC tradition, including a repeat field.
- The average clock cycles per instruction (CPI) of these CISC instructions is typically 10 to 20.

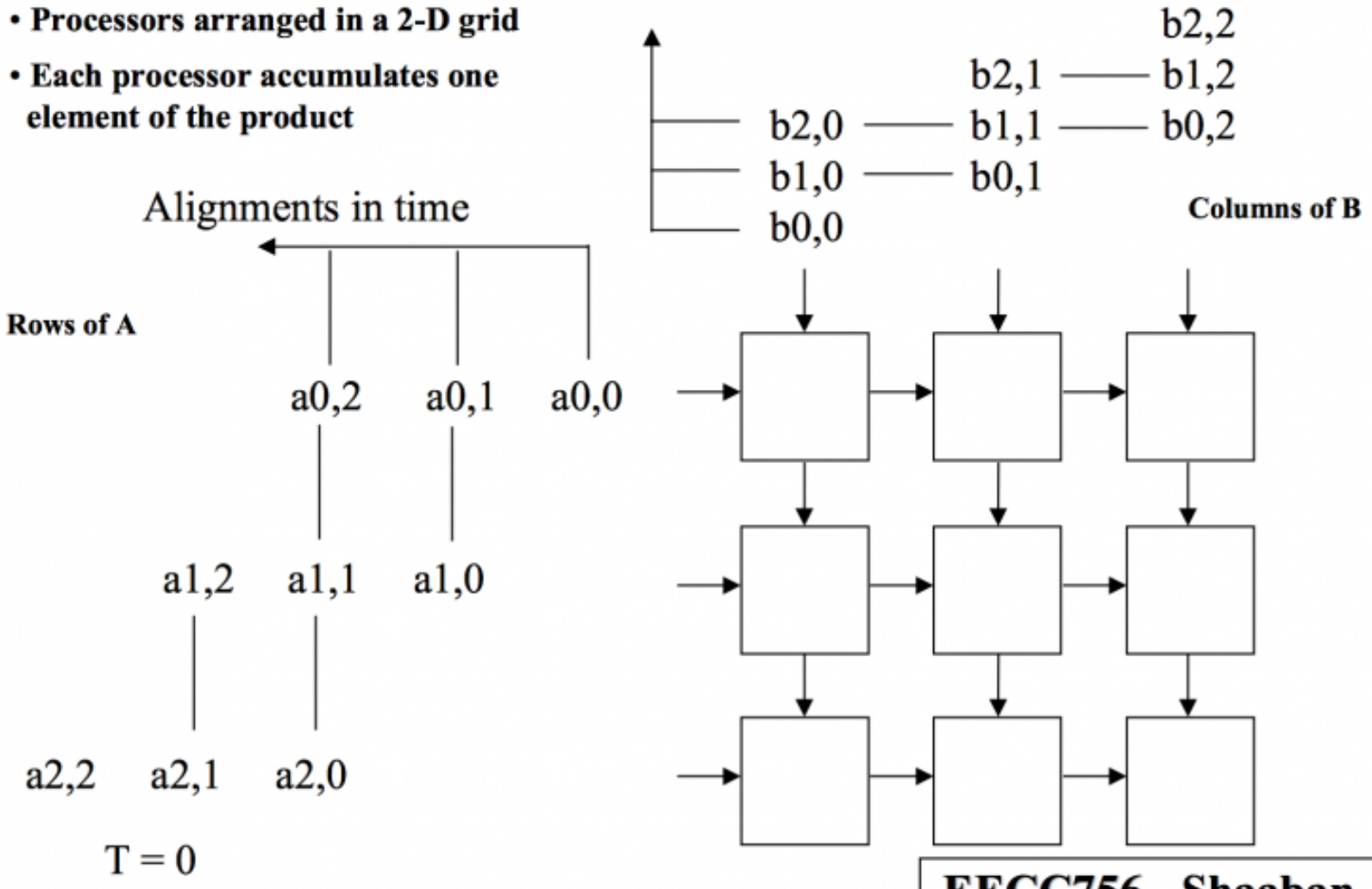
TPU Execution Stage

- CISC MatrixMultiply Instruction in TPU. (12bit)

- The philosophy of the TPU microarchitecture is to keep the matrix unit busy, where each instruction executes in a separate stage.
- It uses a 4-stage pipeline for these CISC instructions, where each instruction executes in a separate stage.

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

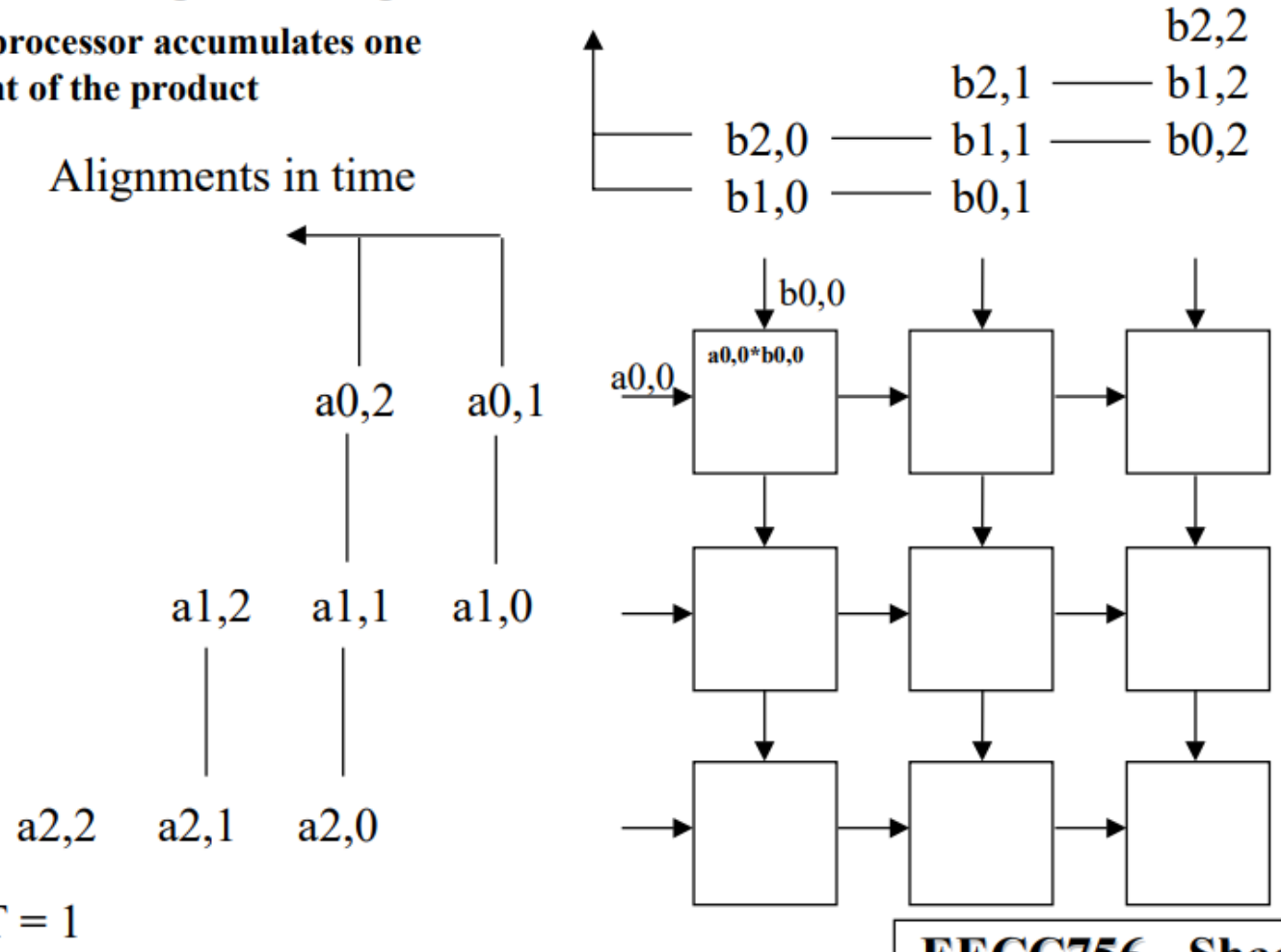
- **Processors arranged in a 2-D grid**
- **Each processor accumulates one element of the product**



Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

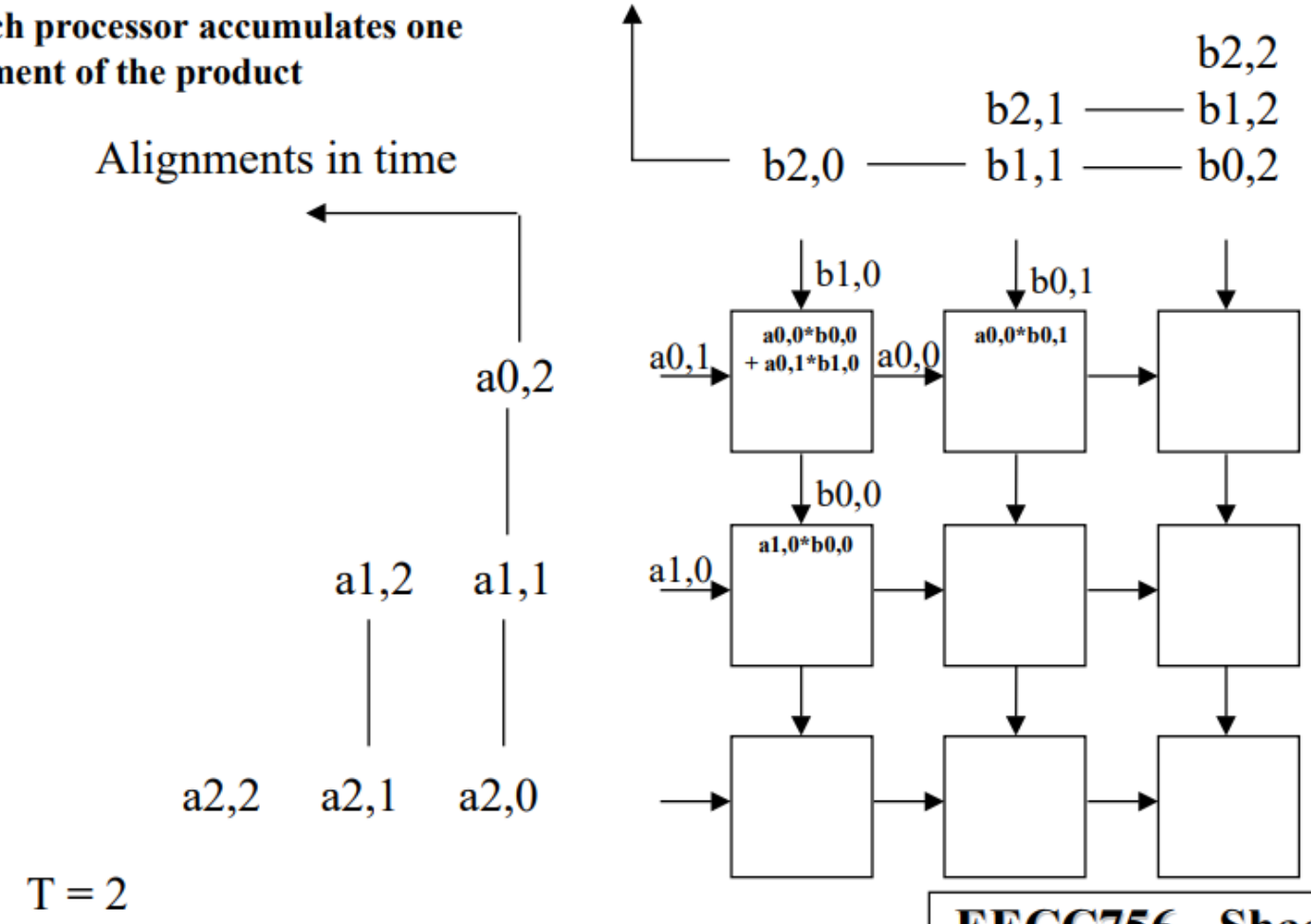
- **Processors arranged in a 2-D grid**
- **Each processor accumulates one element of the product**



Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

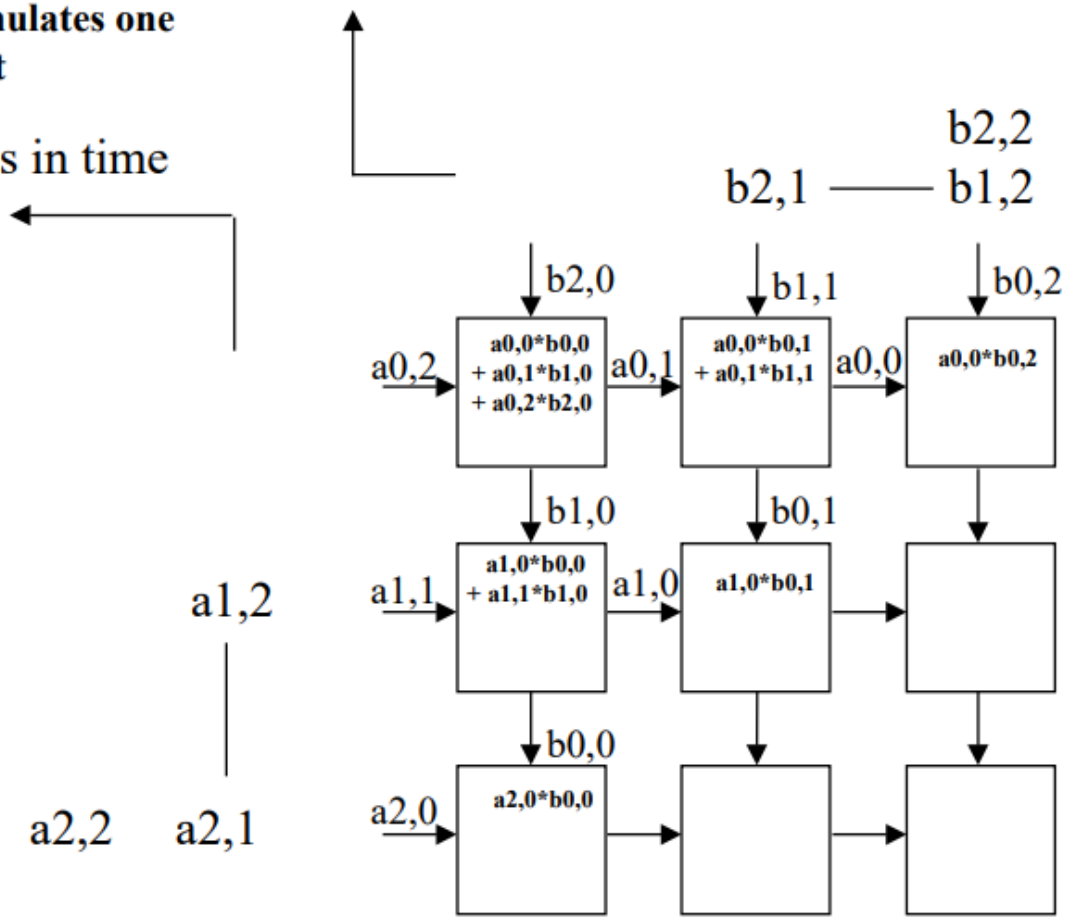
- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product



Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

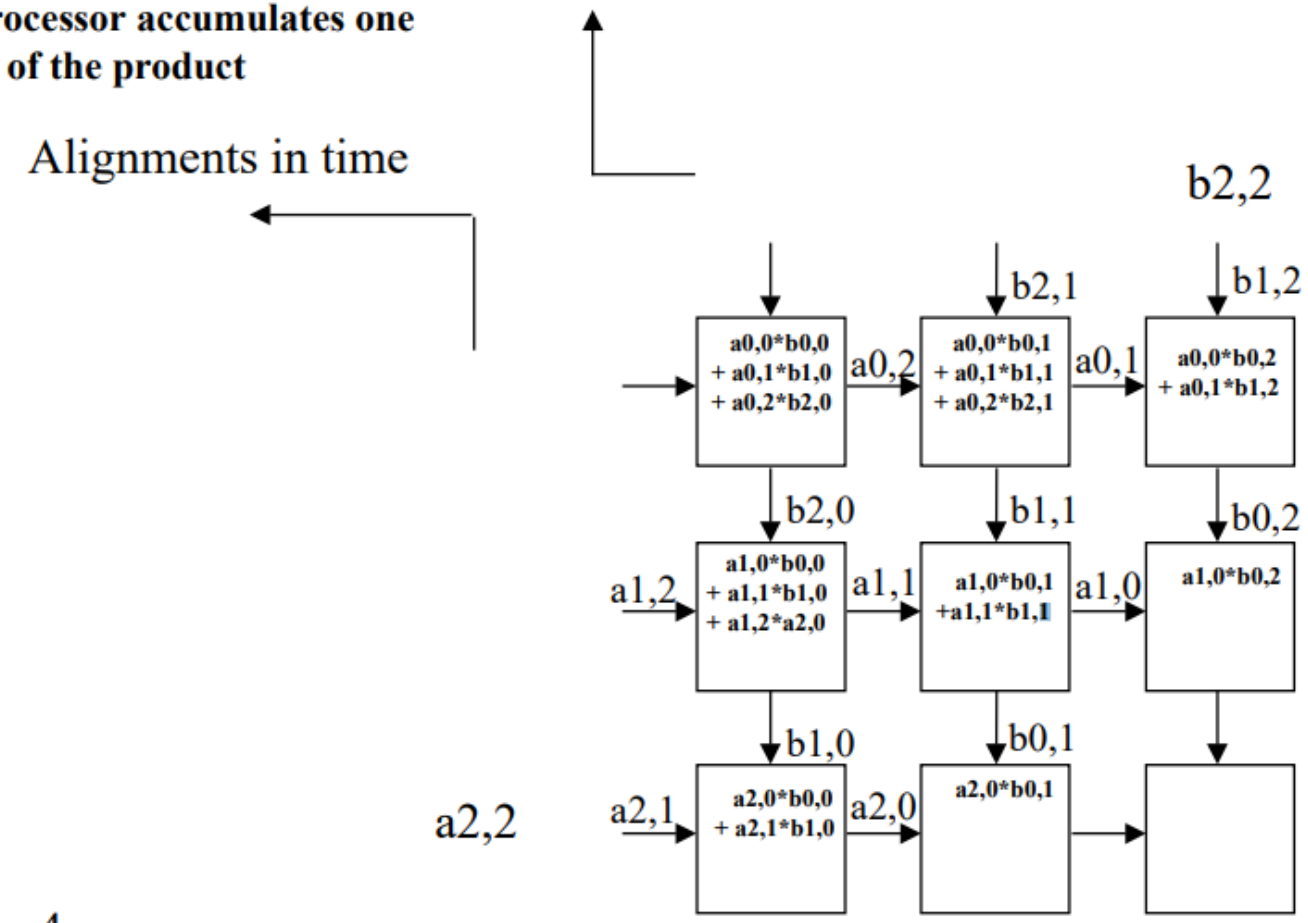
- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product



Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- **Processors arranged in a 2-D grid**
- **Each processor accumulates one element of the product**

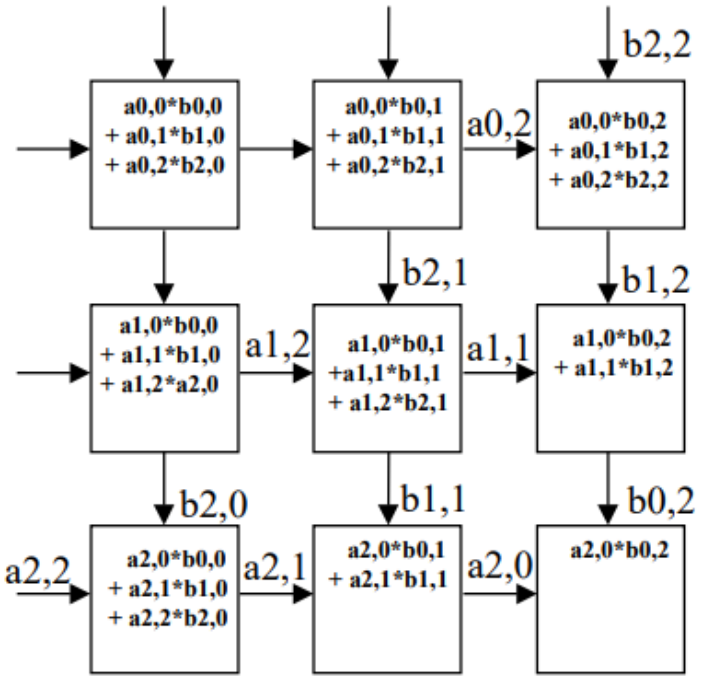


Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

Alignments in time



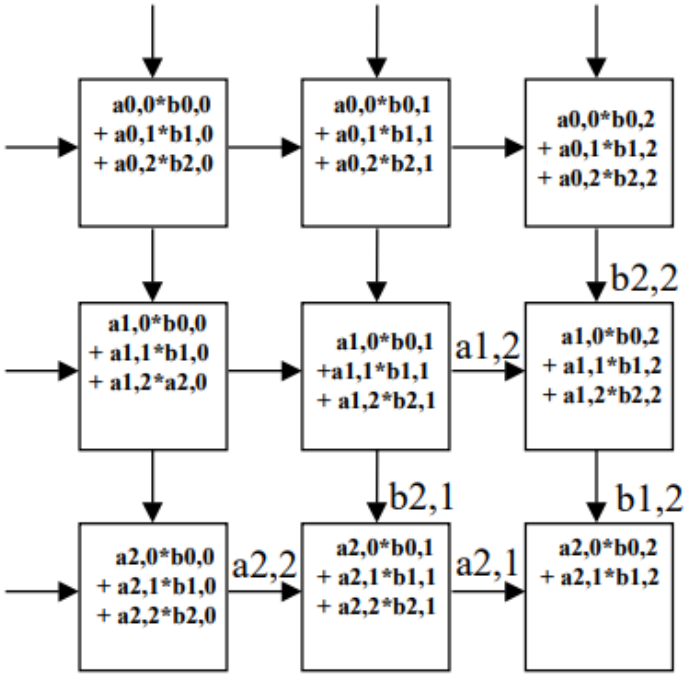
T = 5

Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- **Processors arranged in a 2-D grid**
- **Each processor accumulates one element of the product**

Alignments in time

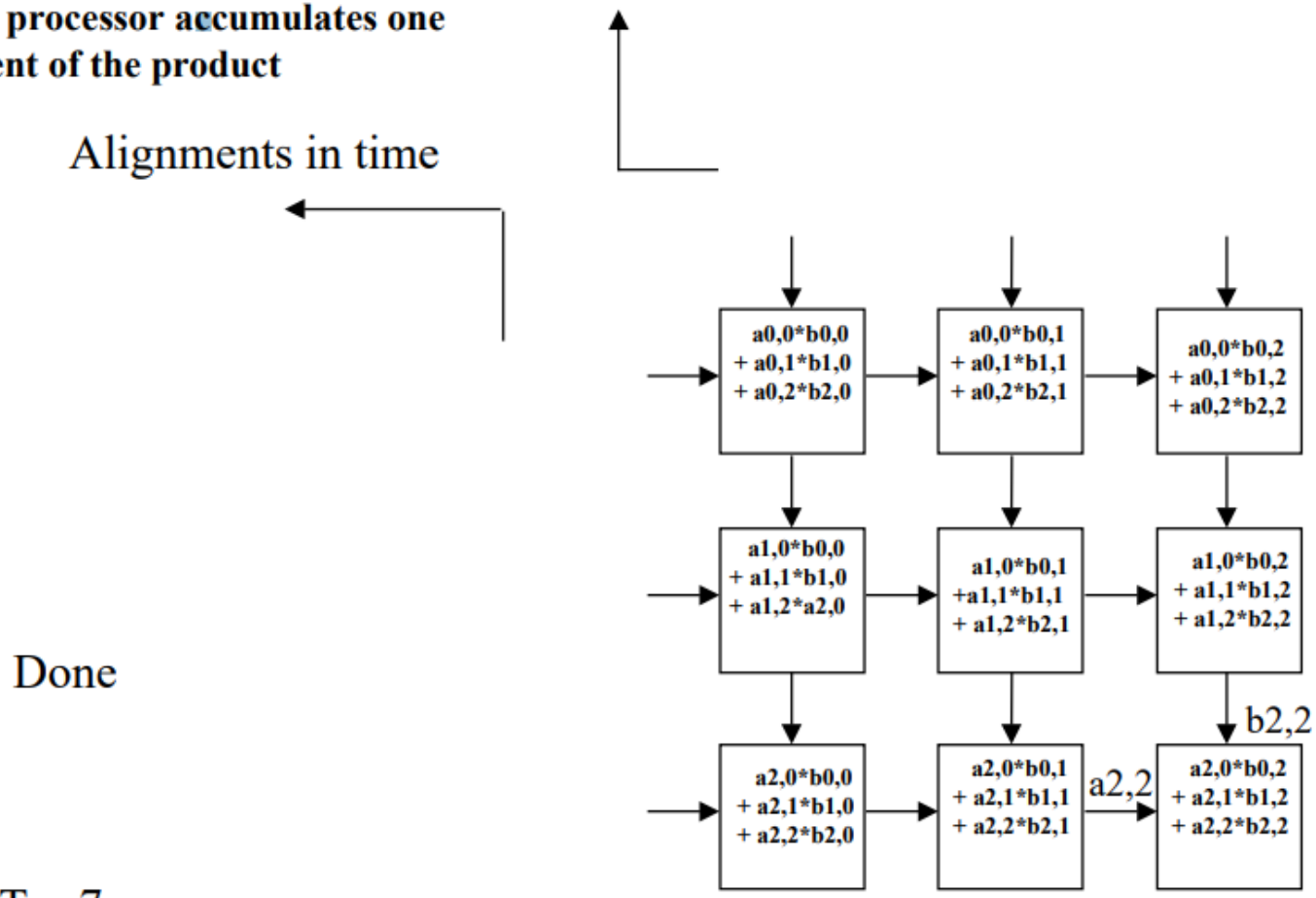


T = 6

Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

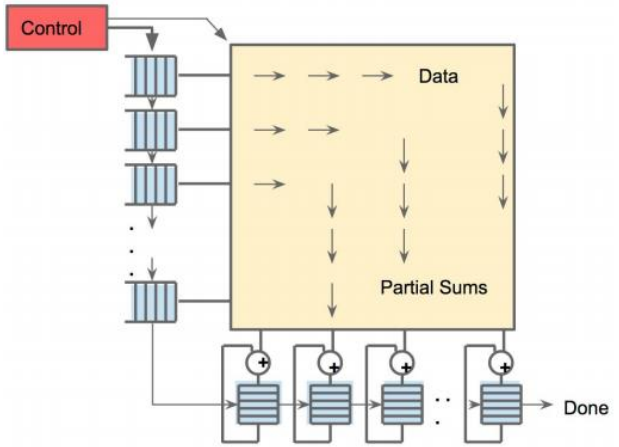
- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product



Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array

- As reading a large SRAM uses much more power than arithmetic, the matrix unit uses systolic execution to save energy by reducing reads and writes of the Unified Buffer.
- The data flows in from the left, and the weights are loaded from the top.
- A given 256-element multiply-accumulate operation moves through the matrix as a diagonal wavefront.



Software

- The TPU software stack had to be compatible with those developed for CPUs and GPUs so that applications could be ported quickly to the TPU.
- The portion of the application run on the TPU is typically written in TensorFlow and is compiled into an API that can run on GPUs or TPUs.
- In TensorFlow 1.3, operations and bindings for the cloud TPU are included.

Key Instruction Set

- It has about a dozen instructions overall, but these five are the key ones.
  - Read_Host_Memory reads data from the CPU host memory into the Unified Buffer.
  - Read_Weights reads weights from Weight Memory into the Weight FIFO as input to the Matrix Unit.
  - MatrixMultiply/Convolve do matrix multiply or convolution operation and save results to accumulator.
  - Activate do activate function operation such as RELU, Sigmoid.
  - Write_Host_Memory writes data from the Unified Buffer into the CPU host memory.

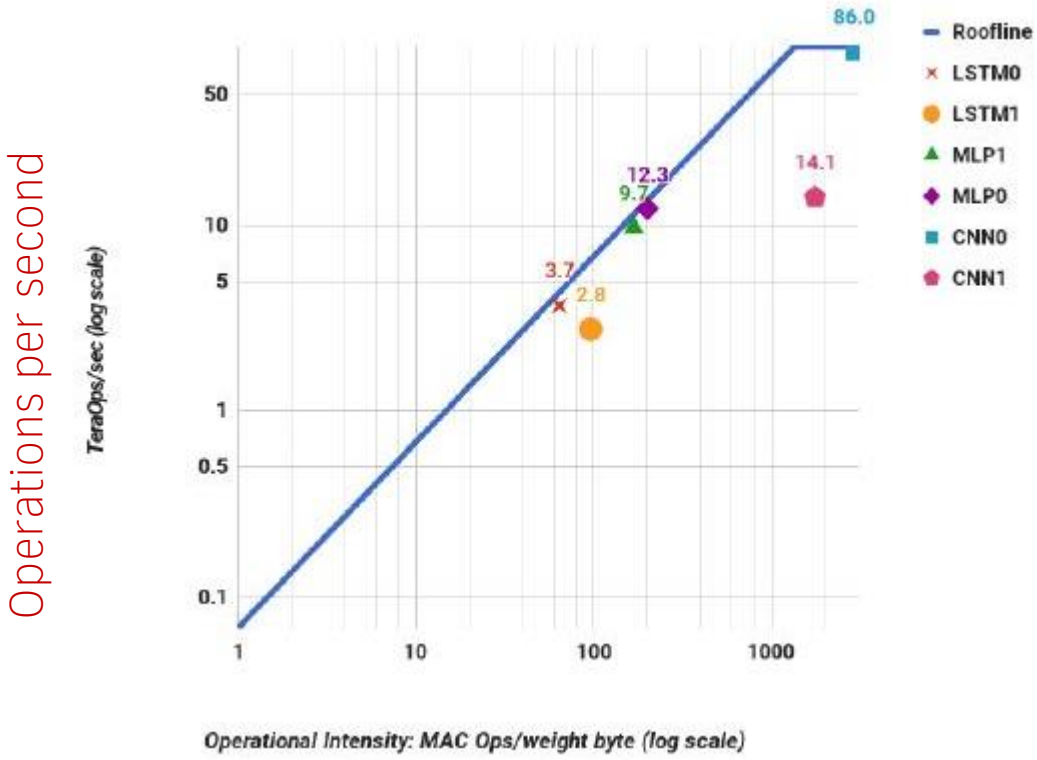
NN Structure and H/W

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	

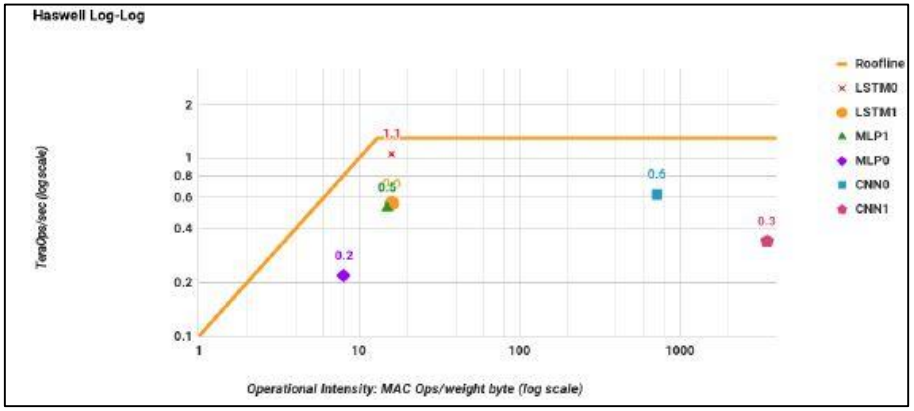
Model	*Die*										*Benchmarked Servers*				
	mm²	*nm*	*MHz*	*TDP*	*Measured*		*TOPS/s*		*GB/s*	*On-Chip Memory*	*Dies*	*DRAM Size*	*TDP*	*Measured*	
					Idle	*Busy*	8b	FP						*Idle*	*Busy*
Haswell E5-2699 v3	662	22	2300	145W	41W	145W	2.6	1.3	51	51 MiB	2	256 GiB	504W	159W	455W
NVIDIA K80 (2 dies/card)	561	28	560	150W	25W	98W	--	2.8	160	8 MiB	8	256 GiB (host) + 12 GiB x 8	1838W	357W	991W
TPU	NA*	28	700	75W	28W	40W	92	--	34	28 MiB	4	256 GiB (host) + 8 GiB x 4	861W	290W	384W

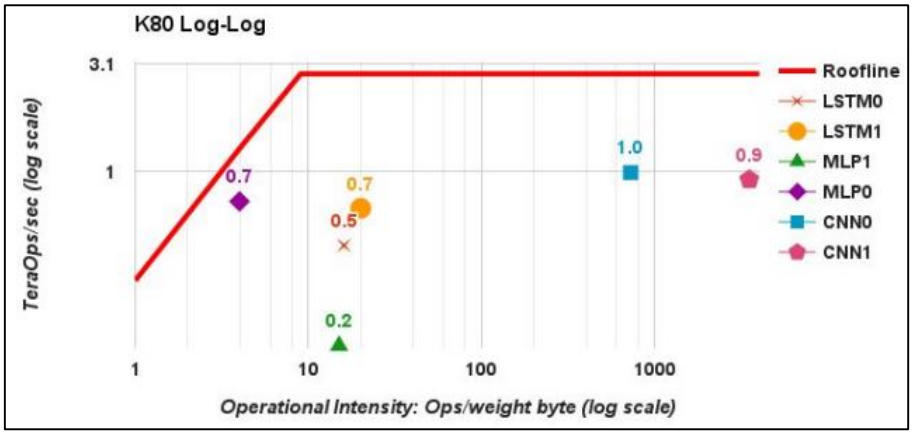
Performance

- To illustrate the performances, they used **Roofline Performance model**.



Performance (cont.)





- The six NN applications are generally further below their ceilings than was the TPU and reason is **the response time**.
- Many of these NN applications are parts of end-user-facing services.
- Small increases in response time cause customers to use a service less.

Performance (cont.)

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%

Table 4. 99th% response time and per die throughput (IPS) for MLP0 as batch size varies for MLP0. The longest allowable latency is 7 ms. For the GPU and TPU, the maximum MLP0 throughput is limited by the host server overhead. Larger batch sizes increase throughput, but as the text explains, their longer response times exceed the limit, so CPUs and GPUs must use less-efficient, smaller batch sizes (16 vs. 200). They run 2.3X – 2.7X slower than if response time was unbound, but the for the more deterministic TPU, the slowdown from the 99th% response-time limit is just 1.2X.

- CPU and GPU performance in restrict response time(7ms) is almost 40% of their full performance.
- TPU's performance in restrict response time is 80% of TPU's full performance.

Performance (cont.)

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

Table 3. Factors limiting TPU performance of the NN workload based on hardware performance counters. Rows 1, 4, 5, and 6 total 100% and are based on measurements of activity of the matrix unit. Rows 2 and 3 further break down the fraction of 64K weights in the matrix unit that hold useful weights on active cycles. Our counters cannot exactly explain the time when the matrix unit is idle in row 6; rows 7 and 8 show counters for two possible reasons, including RAW pipeline hazards and PCIe input stalls. Row 9 (TOPS) is based on measurements of production code while the other rows are based on performance-counter measurements, so they are not perfectly consistent. Host server overhead is excluded here. The MLPs and LSTMs are memory-bandwidth limited but CNNs are not. CNN1 results are explained in the text.

The Time TPU Interacts with CPU

MLP0	*MLP1*	*LSTM0*	*LSTM1*	*CNN0*	*CNN1*
21%	76%	11%	20%	51%	14%

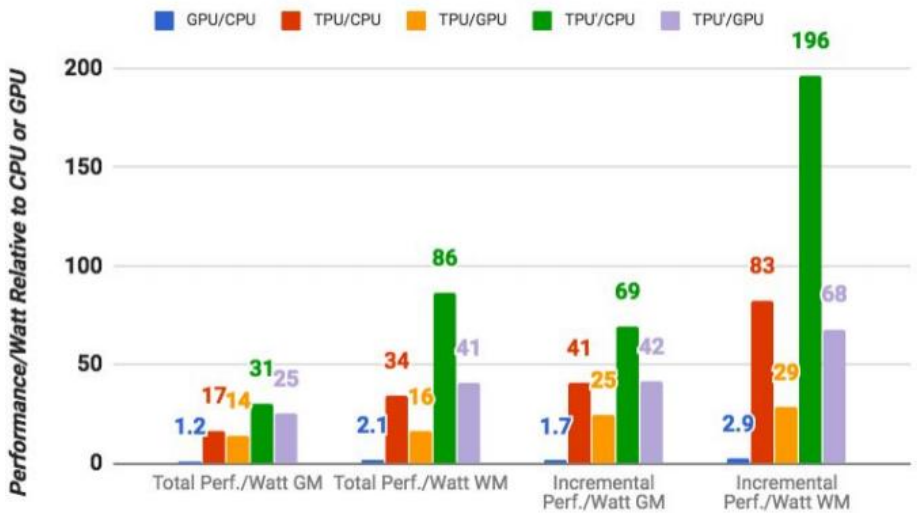
Table 5. Time for host CPU to interact with the TPU expressed as percent of TPU execution time (from TPU performance counters). This fraction is the time the CPU and TPU are communicating over the PCIe bus, *not* including the time the CPU is doing a portion of the application but not interacting with the TPU. As the text explains, it's hard for the TPU to measure if the CPU is idle or working on the app.

Performance (cont.)

Type	*MLP0*	*MLP1*	*LSTM0*	*LSTM1*	*CNN0*	*CNN1*	*GM*	*WM*
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

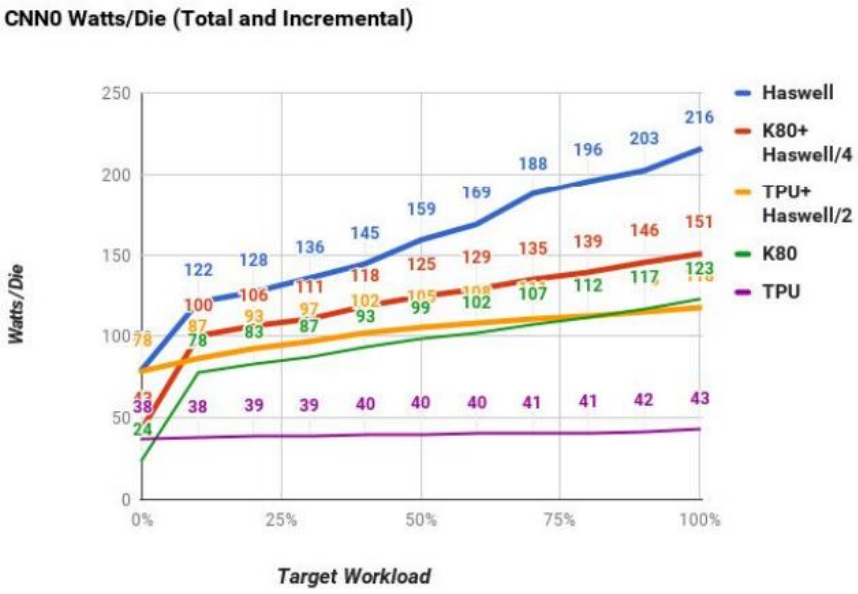
- Relative inference(prediction) performance per die including the host server overhead for the two accelerators versus the CPU.
- GM and WM are geometric and weighted mean.
- TPU is 14.5 times faster than CPU and is 13.2 times faster than GPU.

Performance/Watt



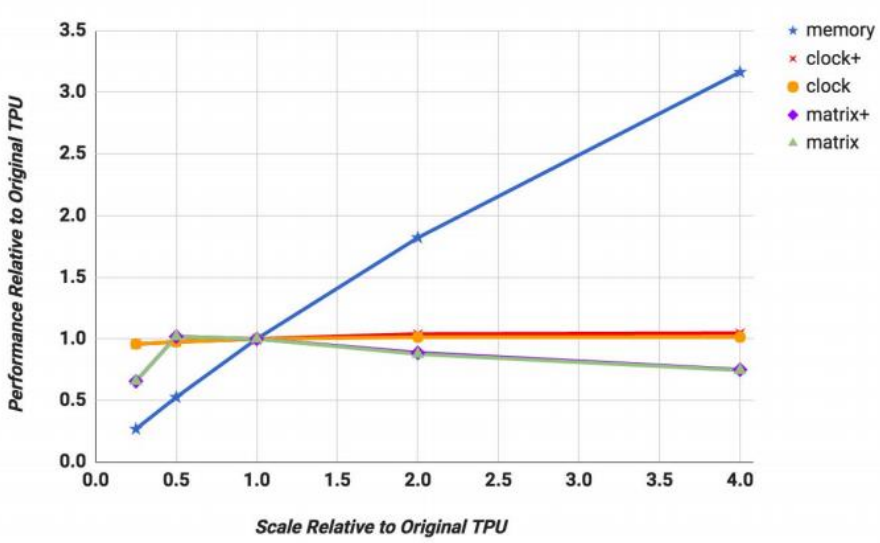
- TPU' is an improved TPU.
- Total Perf./Watt includes CPU and GPU power consumption for server computing
- Incremental Perf./Watt doesn't includes CPU and GPU power consumption for server computing.
- In incremental case, TPU' Perf./Watt by GM is 17, 34times better than CPU and is 14, 16 times better than GPU.

Watts/die for CNN0



- The Total GPU and TPU power are the red and orange lines and their Incremental power are the green and purple lines.
- A server has 2 CPUs and 8 GPUs or 4 TPUs.
- TPU's energy proportionality is most bad.
- However, TPU consumes only around 40Watt.

Performance by element scale



- Weighted mean TPU performance as metrics scale from 0.25x to 4x: memory bandwidth, clock rate + accumulators, clock rate, matrix unit dimension + accumulators, and matrix unit dimension.
- Memory bandwidth is most important element for performance.
- Because most NN Benchmark applications cause performance bottlenecks due to memory bounds.

Usage of Unified Buffer

MLP0	*MLP1*	*LSTM0*	*LSTM1*	*CNN0*	*CNN1*
11.0	2.3	4.8	4.5	1.5	13.9

- A 14 MiB Unified Buffer is sufficient for these apps.

Discussion

- In this paper, they organize the authors' ideas by dividing various topics (author thoughts) into Fallacy (false thinking) and Pitfall (danger).

Discussion(cont.)

- Fallacy: NN inference applications in datacenters value throughput as much as response time.
- Answer: Developers had strong response-time demands. Even the developers of one application in 2014 that cared about response time (LSTM1) said the limit was 10 ms in 2014, but shrank it to 7 ms when they actually ported it to the TPU.
- Fallacy: The K80 GPU architecture is a good match to NN inference.
- Answer: K80 is only a little faster at inference than Haswell and much slower than the TPU. Throughput-oriented architectural approach, it may be more challenging for GPUs to meet the strict latency limits

Discussion(cont.)

- Pitfall: Architects have neglected important NN tasks
- Answer: 15% of the papers at ISCA 2016 were on hardware accelerators for NN. All nine papers looked at CNNs, and only two mentioned other NNs. CNN is only about 5% of our datacenter NN workload.
- Pitfall: For NN hardware, Inferences Per Second (IPS) is an inaccurate summary performance metric
- Answer: Results show that IPS is a poor overall performance summary for NN hardware. For example, the TPU runs the 4-layer MLP1 at 360,000 IPS but the 89-layer CNN1 at only 4,700 IPS, so TPU IPS vary by 75X!. To compare NN machines better, we need a benchmark suite written at a high-level to port it to the wide variety of NN architectures.

Discussion(cont.)

- Fallacy: The K80 GPU results would be much better if Boost mode were enabled.
- Answer: Setting aside the negative impact of K80 Boost mode on TCO (Section 3), we measured it on LSTM1. Boost mode increased the clock rate by a factor of up to 1.6—from 560 to 875 MHz—which increased performance by 1.4X, but it also raised power by 1.3X. The net gain in performance/Watt is 1.1X, and thus for LSTM1, boost mode would have a minor impact on our energy-speed analysis.

Discussion(cont.)

- Fallacy: CPU and GPU results would be comparable to the TPU if we used them more efficiently or compared to newer versions.
- Answer: In 8-bit results for just one DNN on the CPU, the benefit was ~3.5X. It was less confusing (and less space) to present all CPU results in floating point, rather than having one exception, with its own roofline. DNNs had similar speedup, performance/Watt ratio would drop from 41-83X to 12-24X.
- Pitfall: Performance counters added as an afterthought for NN hardware.
- Answer: The TPU has 106 performance counters, and if anything we would like a few more. The raison d'etre for NN accelerators is performance, and it is way too early in their evolution to have good intuition about what is going on.

Discussion(cont.)

- Fallacy: After two years of software tuning, the only path left to increase TPU performance is hardware upgrades.
- Answer: The performance of CNN1 on the TPU could improve if developers and compiler writers did more work to match CNN1 to the TPU hardware.

Related Work

- Fathom: reference workloads for modern deep learning methods
- The Handbook of Brain Theory and Neural Networks
- Training Neural Networks with Spert-II. Chapter 11 in Parallel Architectures for Artificial Networks: Paradigms and Implementations
- Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks
- High Performance Hardware for Machine Learning
- A VLSI architecture for highperformance, low-cost, on-chip learning
- Special-purpose digital hardware for neural networks: An architectural survey
- If I could only design one circuit…: technical perspective
- Toward accelerating deep learning at scale using specialized hardware in the datacenter
- A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services
- Convolution engine: balancing efficiency & flexibility in specialized computing
- Design of a 1st Generation Neurocomputer

Conclusion

- Despite living on an I/O bus and having relatively little memory bandwidth that limits utilization of the TPU. four of the six NN applications are memory-bound.
- The TPU leverages the order-of-magnitude reduction in energy and area of 8-bit integer systolic matrix multipliers over 32-bit floating-point datapaths of a K80 GPU to pack 25 times as many MACs and 3.5 times the on-chip memory (28 MiB vs. 8 MiB) while using less than half the power of the K80 in a relatively small die.
- Inference applications have serious response-time bounds because they are often part of user facing applications, thus NN architectures need to perform well when coping with 99th-percentile latency deadlines.

Conclusion(cont.)

- The TPU die leverages its advantage in MACs and on-chip memory to run short programs written using the domain specific TensorFlow framework 15 times as fast as the K80 GPU die, resulting in a performance/Watt advantage of 29 times, which is correlated with performance/total cost of ownership. Compared to the Haswell CPU die, the corresponding ratios are 29 and 83.
- The TPU succeeded because of
  - the large—but not too large—matrix multiply unit
  - reduce dependence on host CPU
  - a single-threaded, deterministic execution model that proved to be a good match to 99th-percentile response time limits
  - enough flexibility to match the NNs of 2017 as well as of 2013
  - the omission of general-purpose features that enabled a small and low power die despite the larger data path and memory
  - the use of 8-bit integers by the quantized applications
  - applications were written using TensorFlow, which made it easy to port them to the TPU at high-performance rather than them having to be rewritten to run well on the very different TPU hardware.