Introduction

- Moore's Law enabled:
  - Deep memory hierarchy
  - Wide SIMD units
  - Deep pipelines
  - Branch prediction
  - Out-of-order execution
  - Speculative prefetching
  - Multithreading
  - Multiprocessing

Whats the big deal ?

Figure 1. The historical virtuous cycle of universal processers (a) is turning into a fragmentation cycle (b).

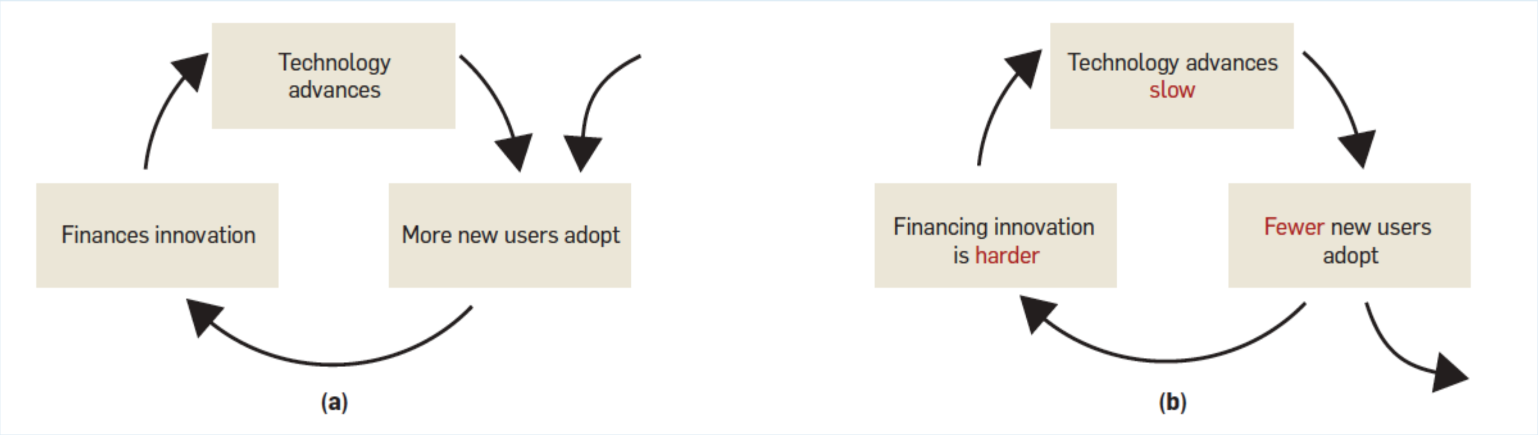
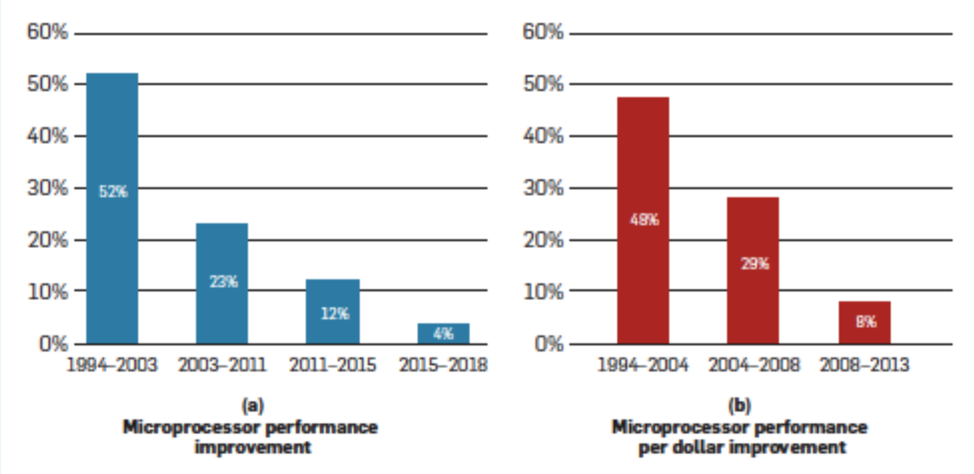
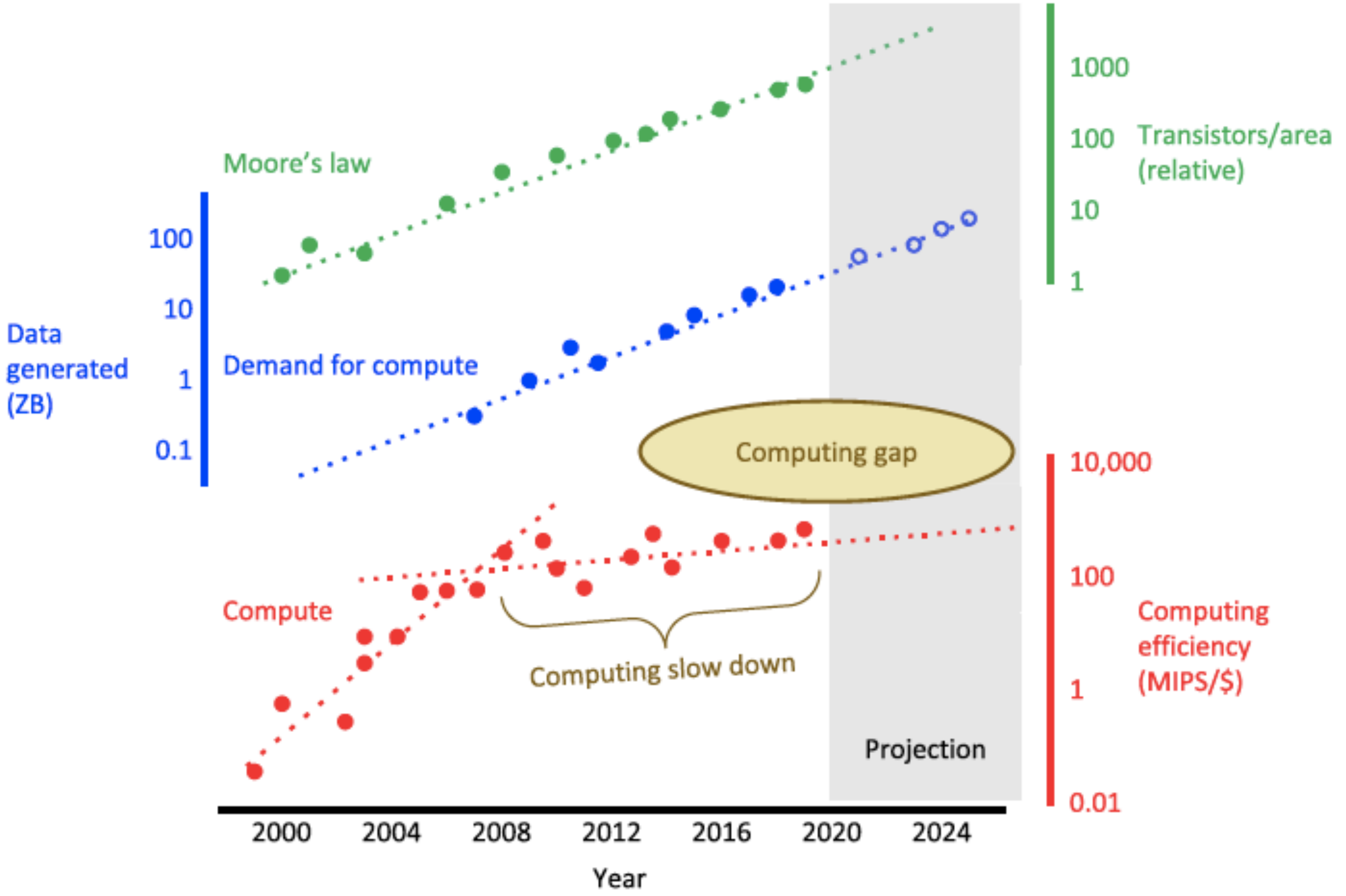


Figure 2. Rate of improvement in microprocessors, as measured by (a) Annual performance improvement on the SPECint benchmark,[7appx] and (b) Annual quality-adjusted price decline.[1appx]



N. Thompson, S. Spanuth, Decline of Computers as a General Purpose Technology, Comm of the ACM March 2021

Whats the big deal ?

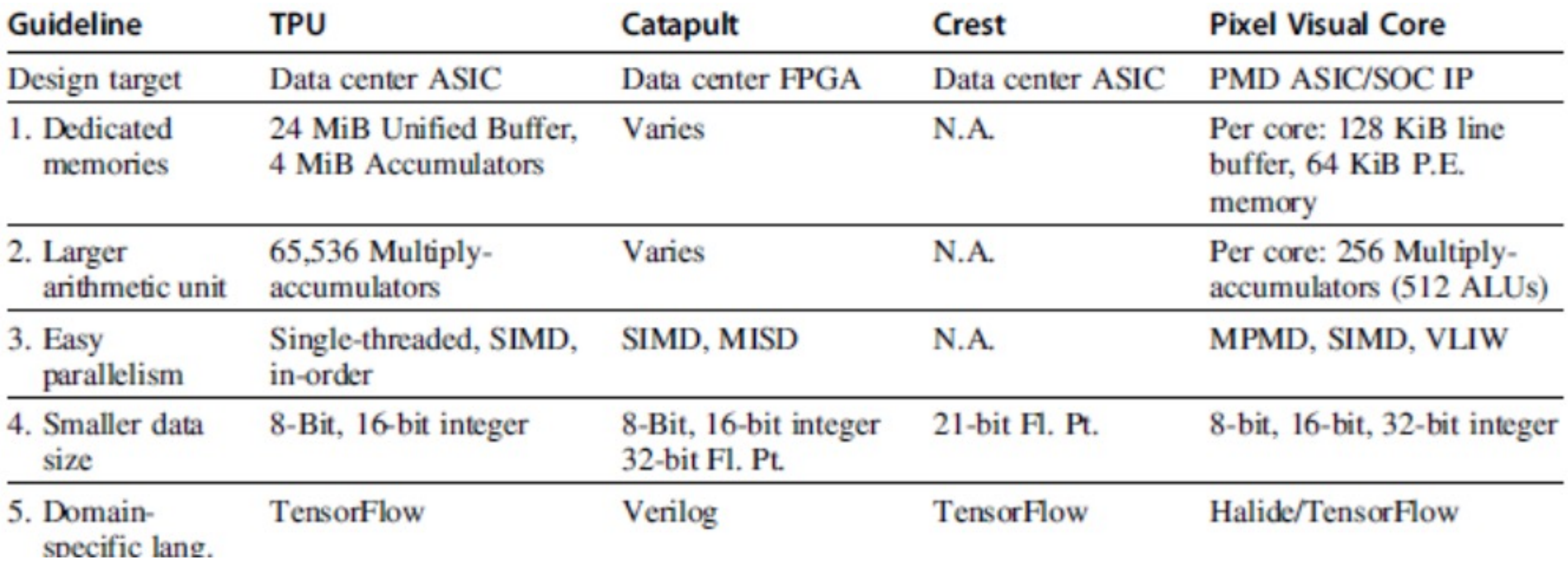


DSA Philosophy 101…..

- Use dedicated memories to minimize data movement
- Invest resources into more arithmetic units or bigger memories
- Use the easiest form of parallelism that matches the domain
- Reduce data size and type to the simplest needed for the domain
- Use a domain-specific programming language

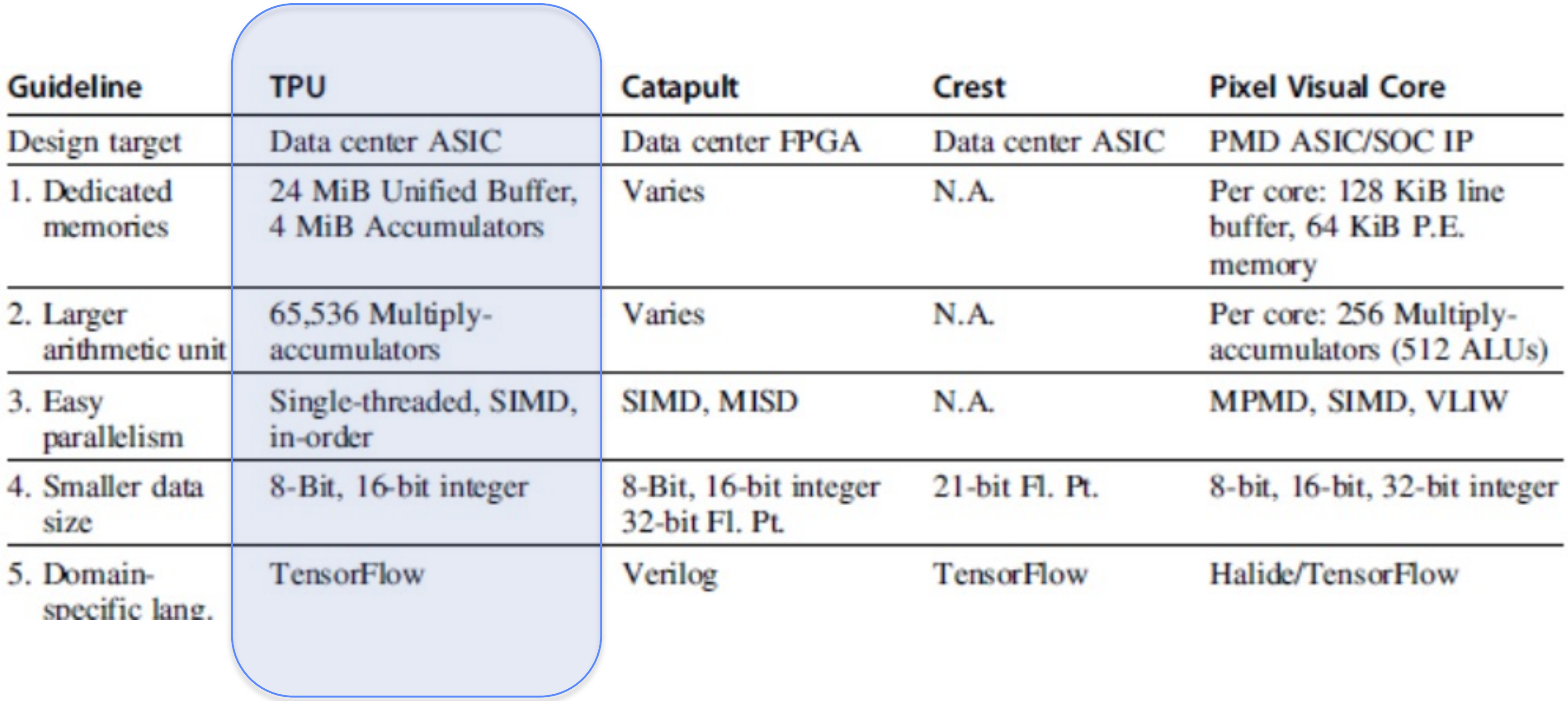
Slides adapted from Comp. Arch A Quantitative Approach Hennessy and Patterson

Examples from Textbook



Guideline	TPU	Catapult	Crest	Pixel Visual Core
Design target	Data center ASIC	Data center FPGA	Data center ASIC	PMD ASIC/SOC IP
1. Dedicated memories	24 MiB Unified Buffer, 4 MiB Accumulators	Varies	N.A.	Per core: 128 KiB line buffer, 64 KiB P.E. memory
2. Larger arithmetic unit	65,536 Multiply-accumulators	Varies	N.A.	Per core: 256 Multiply-accumulators (512 ALUs)
3. Easy parallelism	Single-threaded, SIMD, in-order	SIMD, MISD	N.A.	MPMD, SIMD, VLIW
4. Smaller data size	8-Bit, 16-bit integer	8-Bit, 16-bit integer 32-bit Fl. Pt.	21-bit Fl. Pt.	8-bit, 16-bit, 32-bit integer
5. Domain-specific lang.	TensorFlow	Verilog	TensorFlow	Halide/TensorFlow

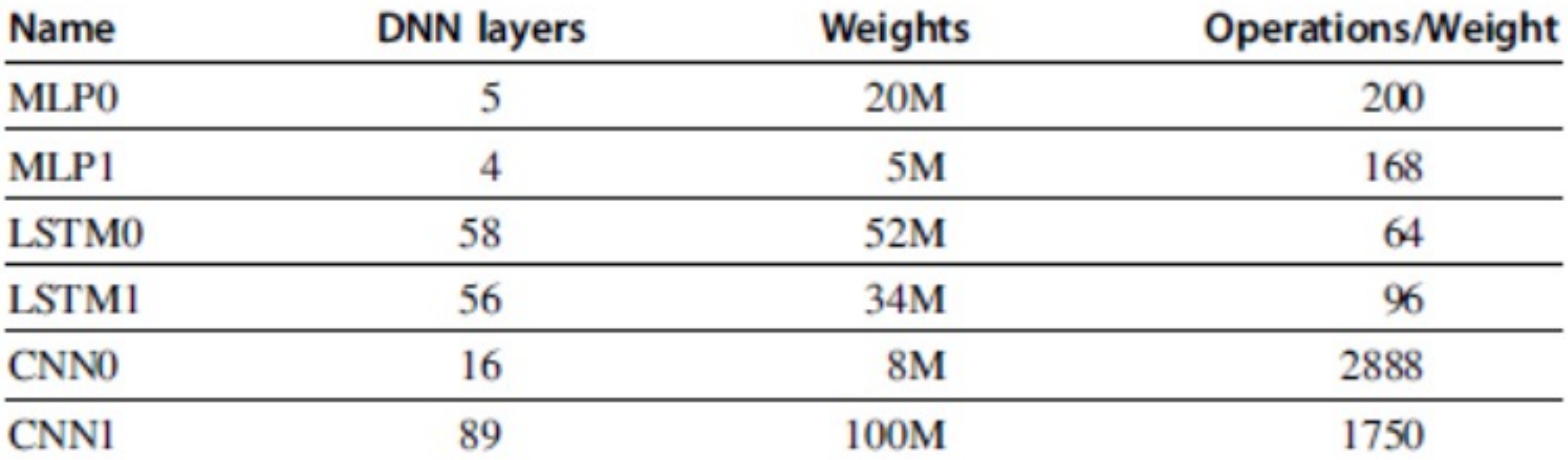
Examples from Textbook



Guideline	TPU	Catapult	Crest	Pixel Visual Core
Design target	Data center ASIC	Data center FPGA	Data center ASIC	PMD ASIC/SOC IP
1. Dedicated memories	24 MiB Unified Buffer, 4 MiB Accumulators	Varies	N.A.	Per core: 128 KiB line buffer, 64 KiB P.E. memory
2. Larger arithmetic unit	65,536 Multiply-accumulators	Varies	N.A.	Per core: 256 Multiply-accumulators (512 ALUs)
3. Easy parallelism	Single-threaded, SIMD, in-order	SIMD, MISD	N.A.	MPMD, SIMD, VLIW
4. Smaller data size	8-Bit, 16-bit integer	8-Bit, 16-bit integer 32-bit Fl. Pt.	21-bit Fl. Pt.	8-bit, 16-bit, 32-bit integer
5. Domain-specific lang.	TensorFlow	Verilog	TensorFlow	Halide/TensorFlow

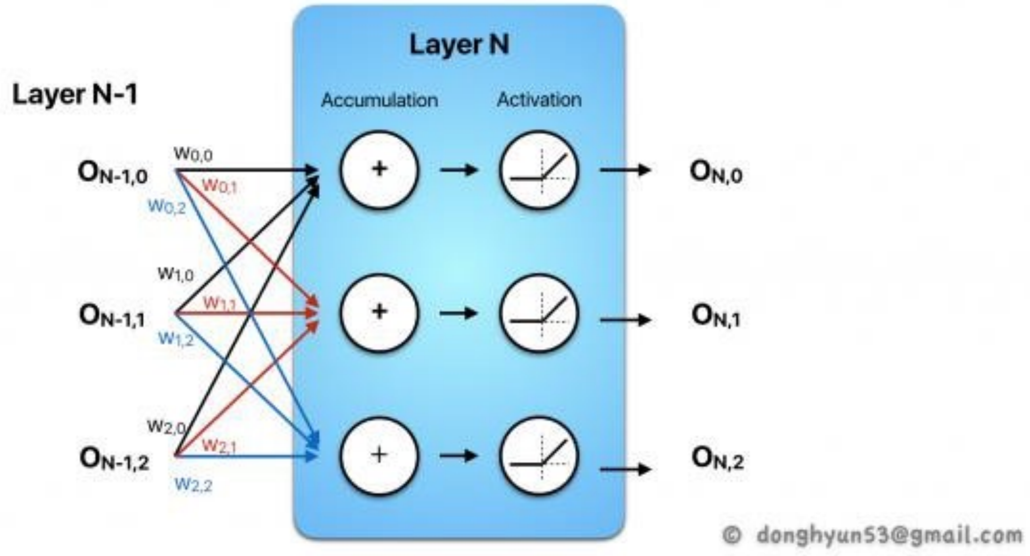
DNN’s: Todays Driving Application

- Inpired by neuron of the brain
- Computes non-linear “activiation” function of the weighted sum of input values
- Neurons arranged in layers



Name	DNN layers	Weights	Operations/Weight
MLP0	5	20M	200
MLP1	4	5M	168
LSTM0	58	52M	64
LSTM1	56	34M	96
CNN0	16	8M	2888
CNN1	89	100M	1750

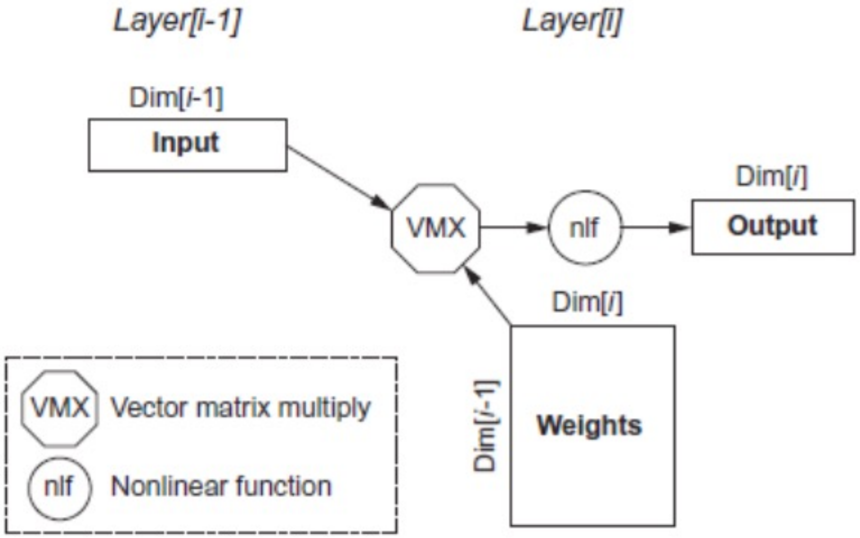
Multi-Layer Perceptrons



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

Multi-Layer Perceptrons

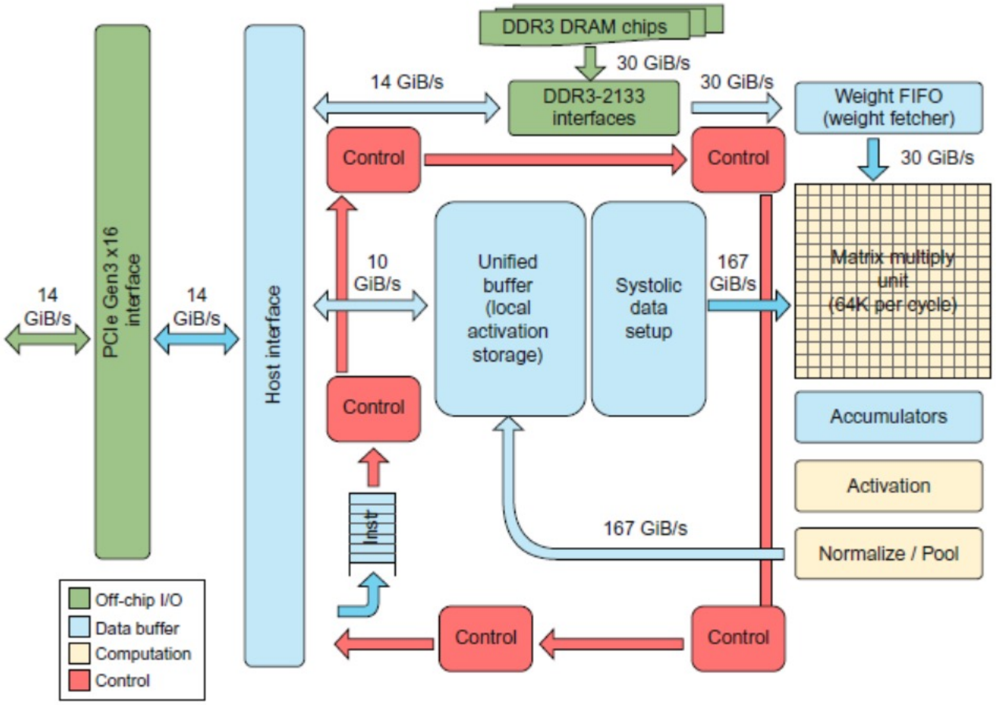
- Parameters:
  - Dim[i]: number of neurons
  - Dim[i-1]: dimension of input vector
  - Number of weights: Dim[i-1] x Dim[i]
  - Operations: 2 x Dim[i-1] x Dim[i]
  - Operations/weight: 2



Slides adapted from Comp. Arch A Quantitative Approach Hennessy and Patterson

Tensor Processing Unit

- Google's DNN ASIC
- 256 x 256 8-bit matrix multiply unit
- Large software-managed scratchpad
- Coprocessor on the PCIe bus



DDR3 DRAM chips

30 GiB/s

14 GiB/s

DDR3-2133 interfaces

30 GiB/s

Weight FIFO (weight fetcher)

30 GiB/s

Control

Control

14 GiB/s

PCIe Gen3 x16 interface

14 GiB/s

Host interface

10 GiB/s

Unified buffer (local activation storage)

Systolic data setup

167 GiB/s

Matrix multiply unit (64K per cycle)

Control

Instr

Accumulators

Activation

167 GiB/s

Normalize / Pool

Control

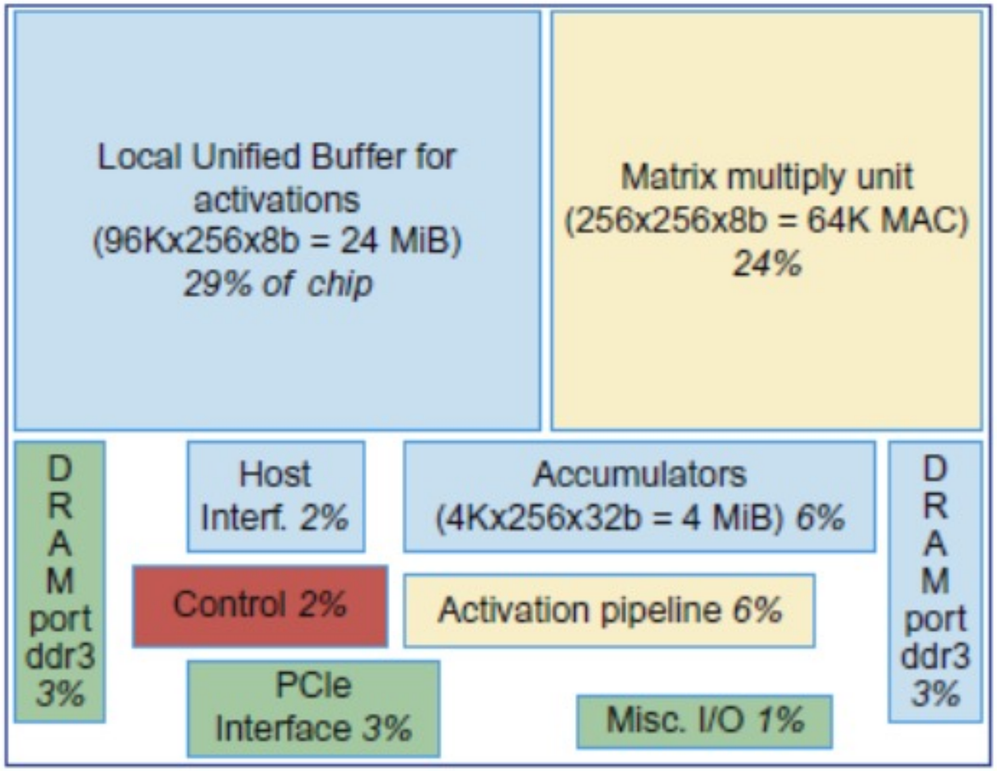
Control

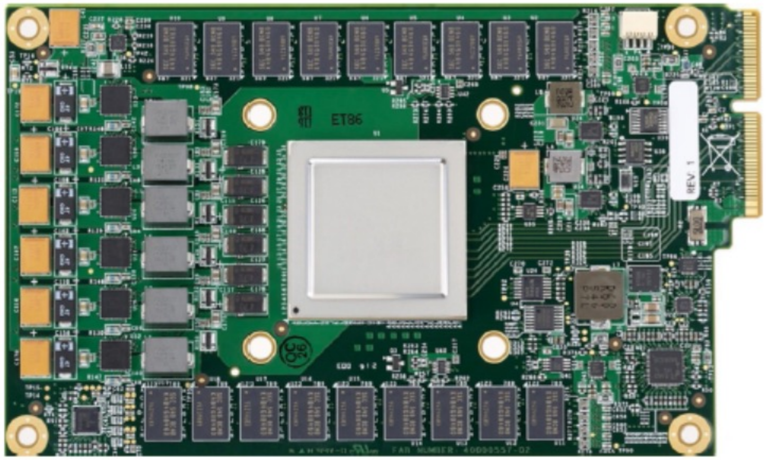
- 🟩 Off-chip I/O
- 🟦 Data buffer
- 🟨 Computation
- 🟥 Control

TPU ISA

- Read_Host_Memory
  - Reads memory from the CPU memory into the unified buffer
- Read_Weights
  - Reads weights from the Weight Memory into the Weight FIFO as input to the Matrix Unit
- MatrixMatrixMultiply/Convolve
  - Perform a matrix-matrix multiply, a vector-matrix multiply, an element-wise matrix multiply, an element-wise vector multiply, or a convolution from the Unified Buffer into the accumulators
  - takes a variable-sized B*256 input, multiplies it by a 256x256 constant input, and produces a B*256 output, taking B pipelined cycles to complete
- Activate
  - Computes activation function
- Write_Host_Memory
  - Writes data from unified buffer into host memory

TPU ISA



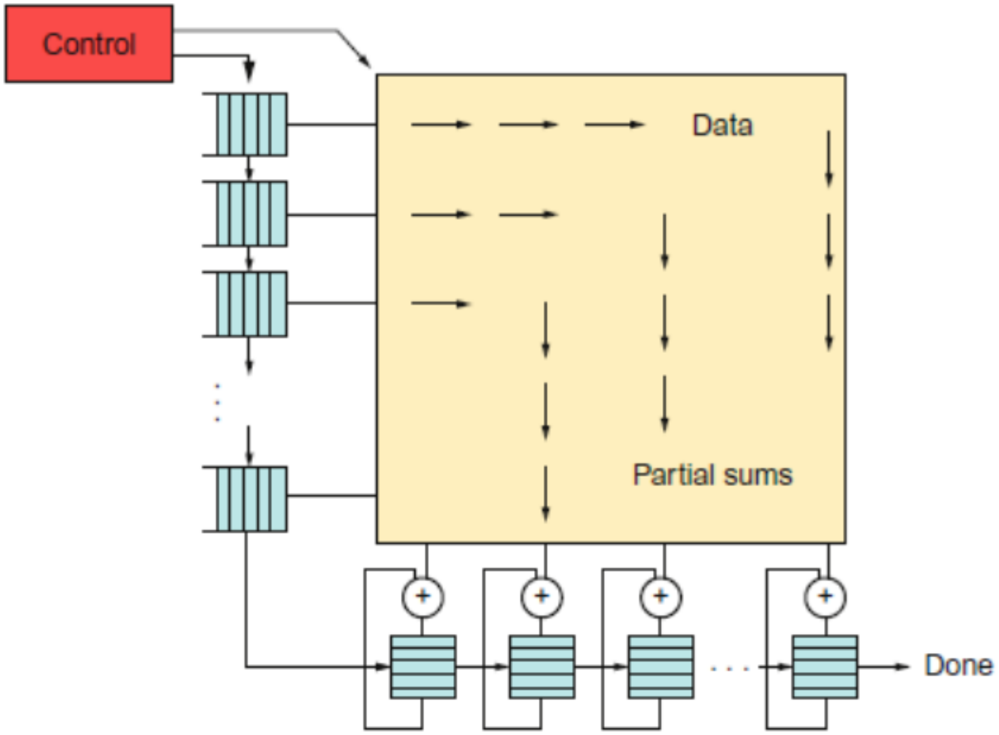


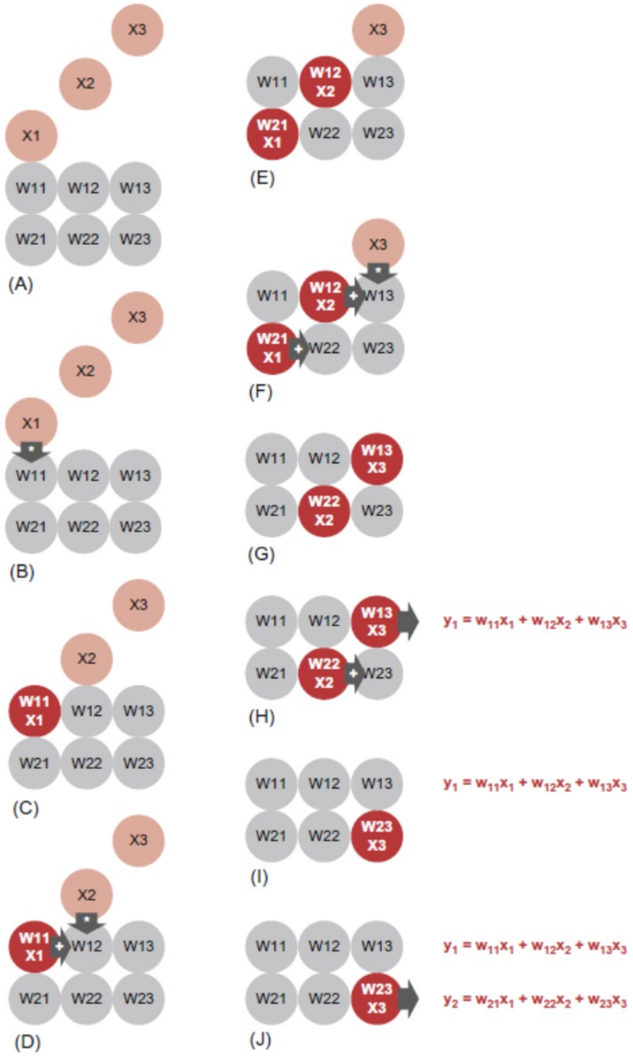
Slides adapted from Comp. Arch A Quantitative Approach Hennessy and Patterson

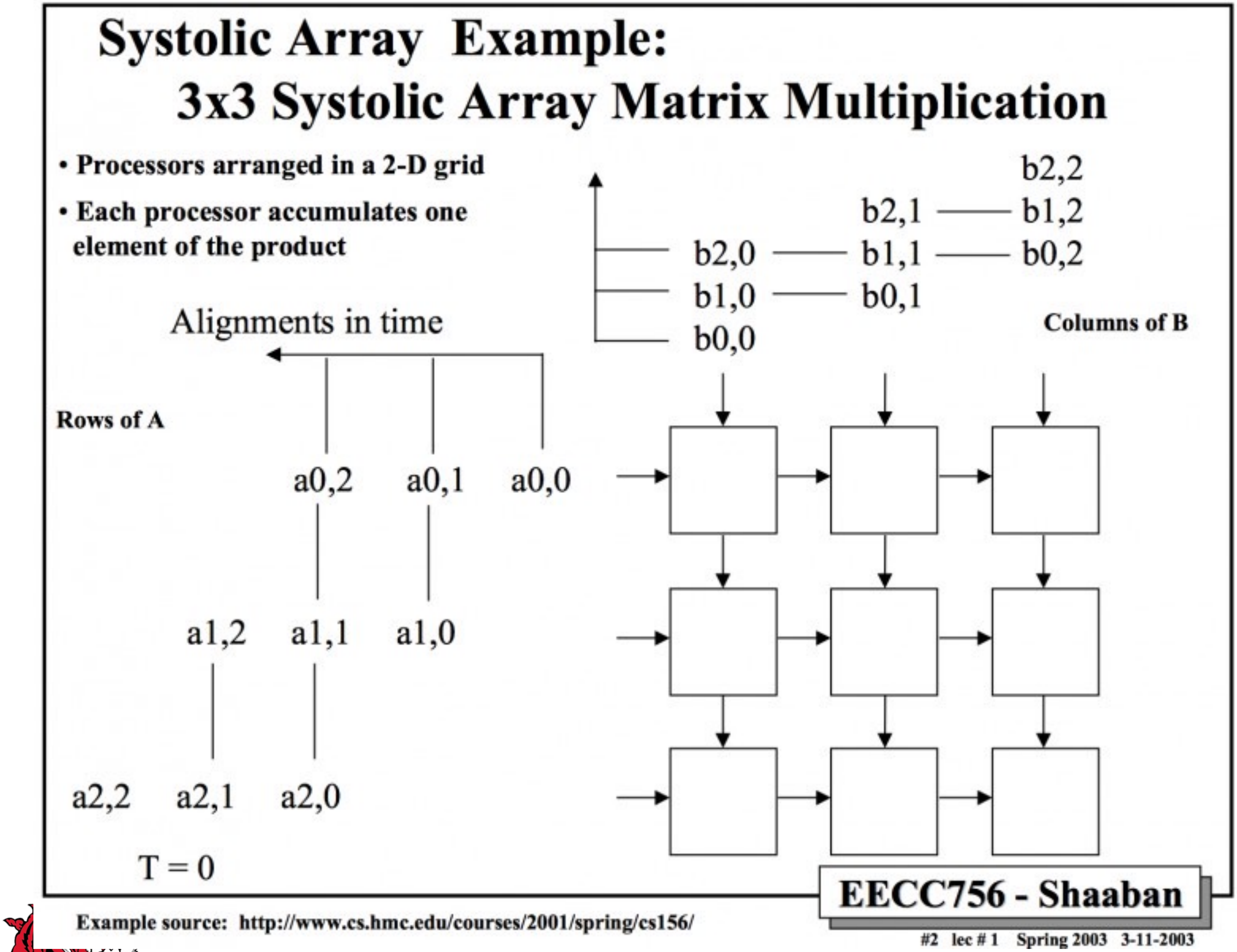
TPU ISA

- 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.

Structured as a Systolic Array




X1
X2
X3
W11
W12
W13
W21
W22
W23
(A)
(B)
(C)
(D)
(E)
(F)
(G)
(H)
(I)
(J)
$y_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3$
$y_2 = w_{21}x_1 + w_{22}x_2 + w_{23}x_3$



Systolic Array Example: 3x3 Systolic Array Matrix Multiplication

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

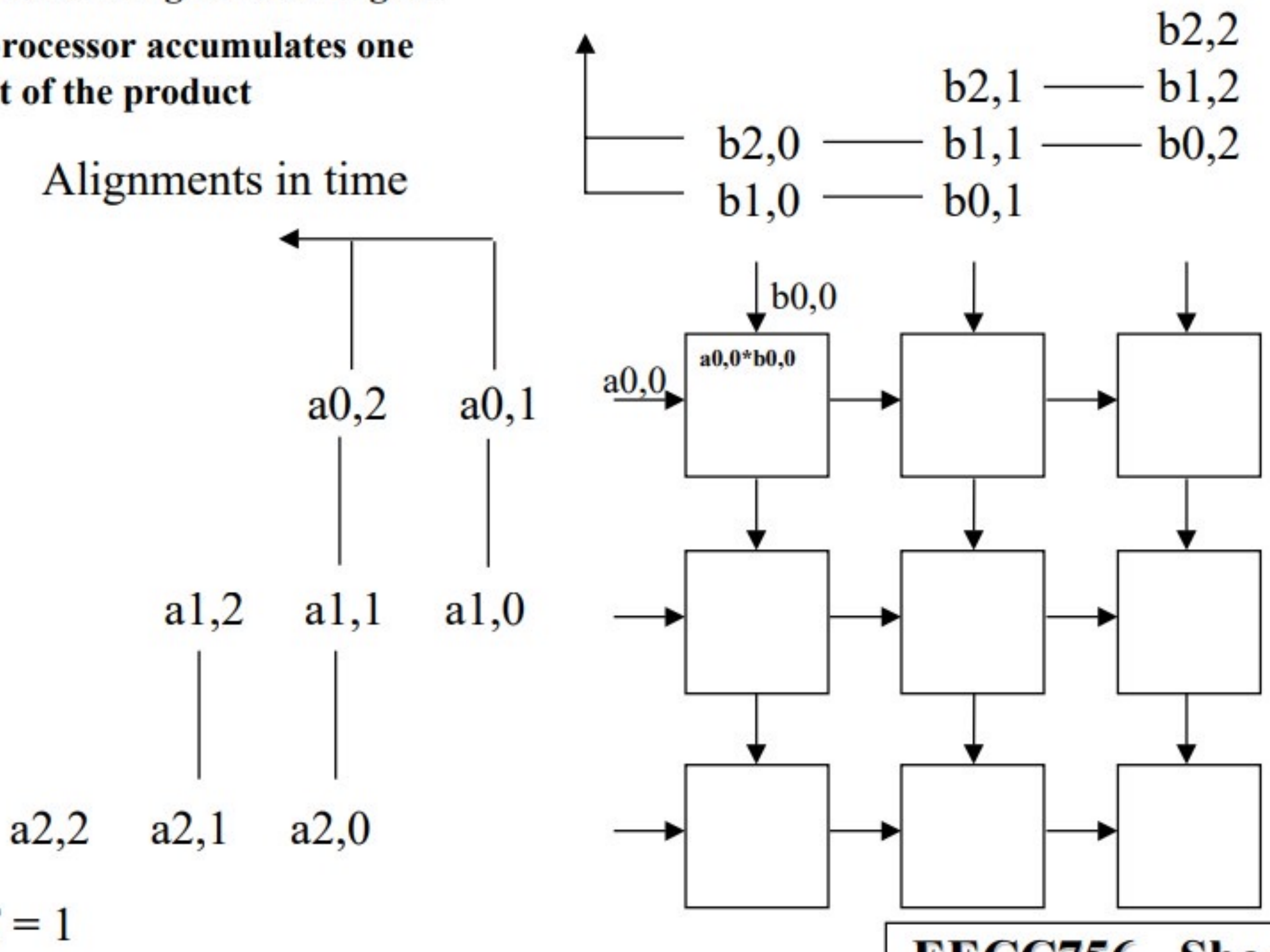
EECC756 - Shaaban

#2 lec # 1 Spring 2003 3-11-2003

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

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

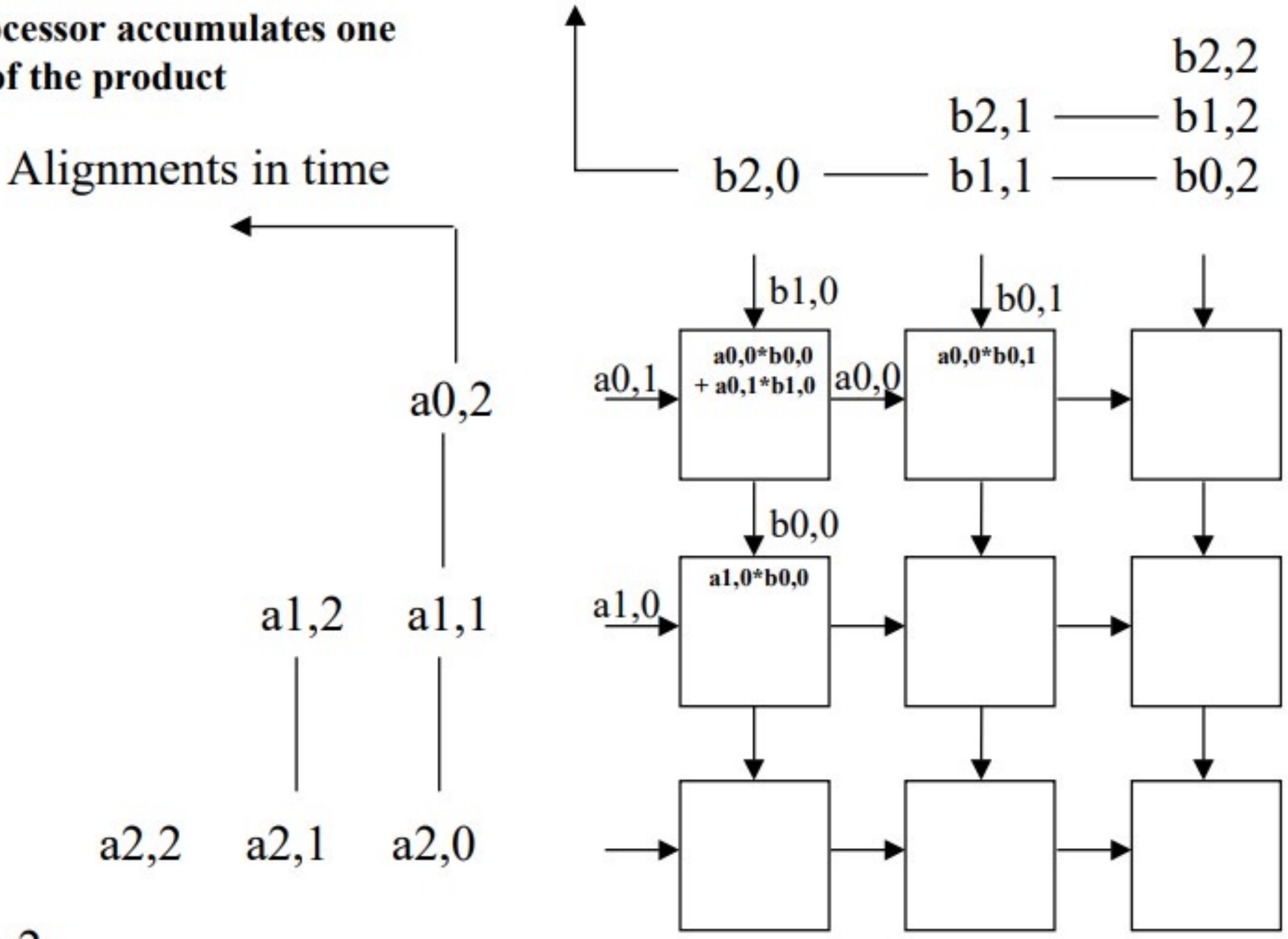
- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product



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

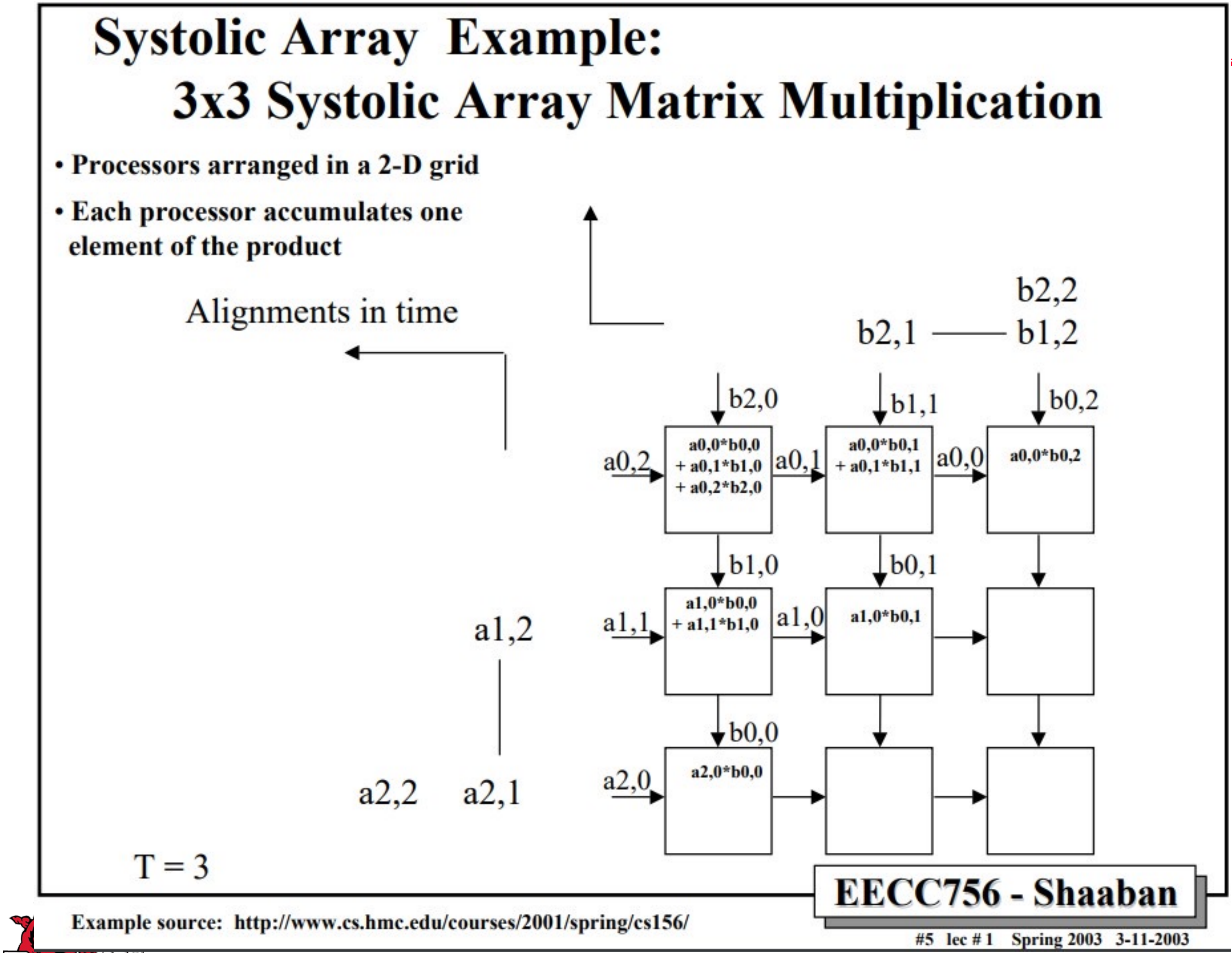
Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

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



T = 2

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



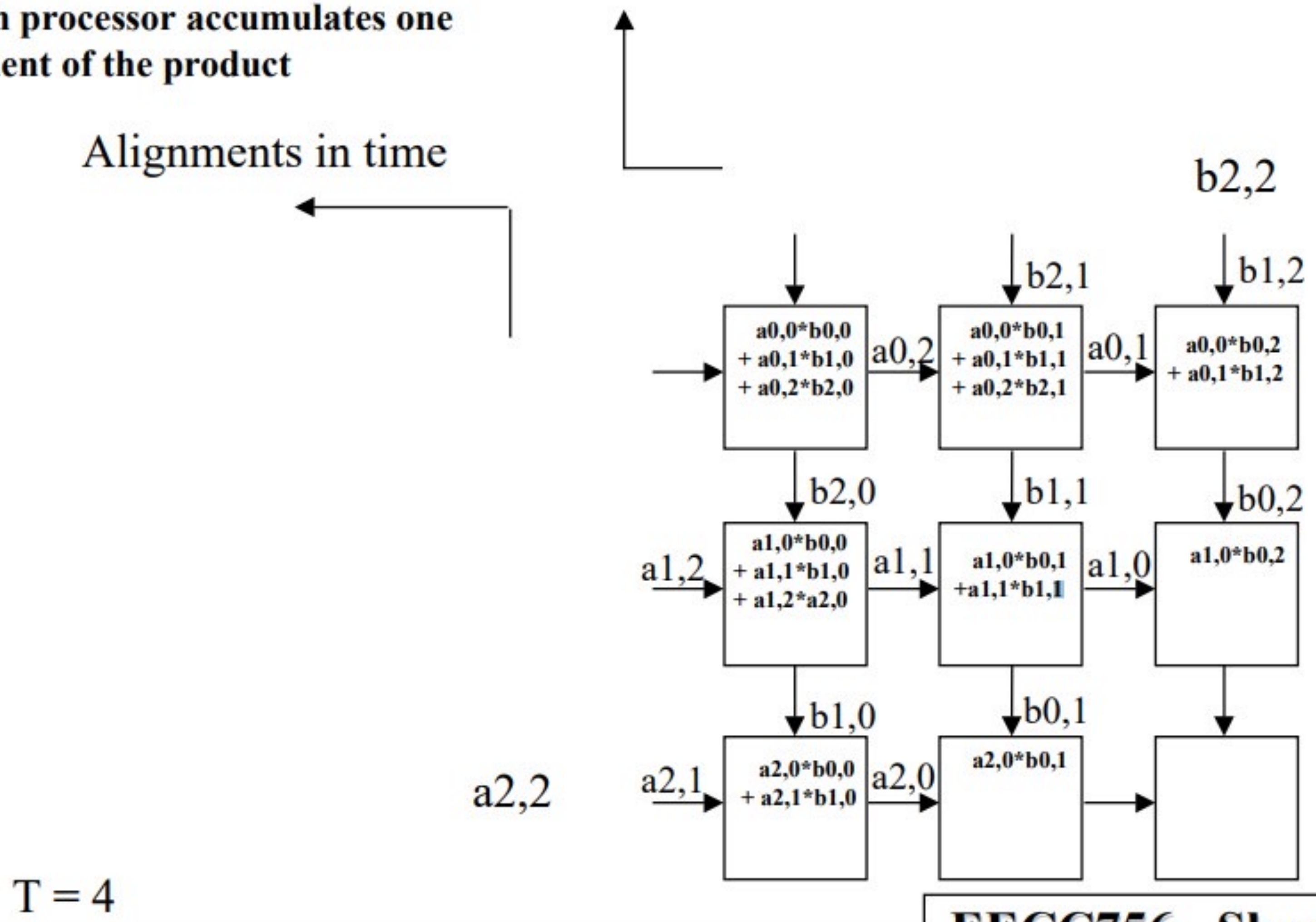
Systolic Array Example: 3x3 Systolic Array Matrix Multiplication

- 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

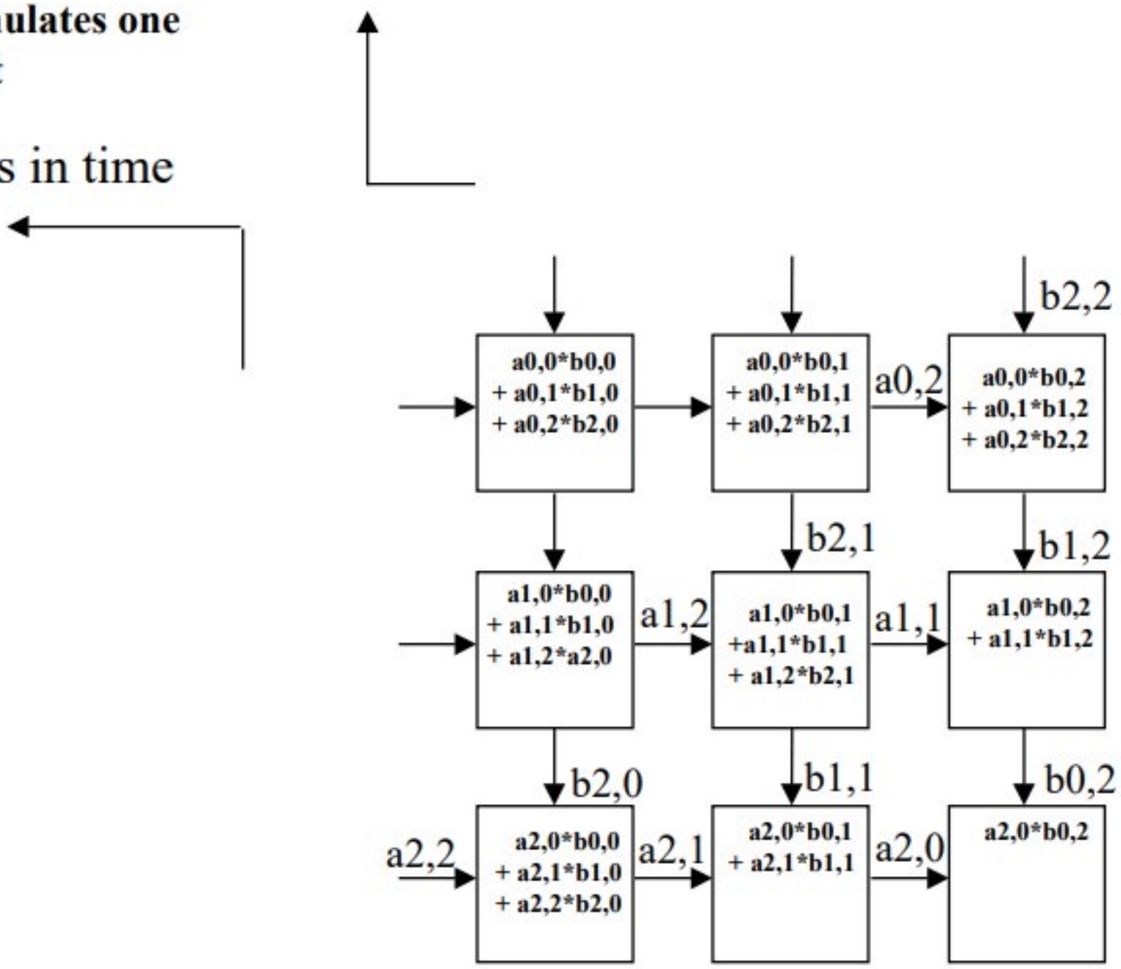


Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

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



a0,0*b0,0 + a0,1*b1,0 + a0,2*b2,0

a0,0*b0,1 + a0,1*b1,1 + a0,2*b2,1

a0,0*b0,2 + a0,1*b1,2 + a0,2*b2,2

a1,0*b0,0 + a1,1*b1,0 + a1,2*a2,0

a1,0*b0,1 +a1,1*b1,1 + a1,2*b2,1

a1,0*b0,2 + a1,1*b1,2

a2,0*b0,0 + a2,1*b1,0 + a2,2*b2,0

a2,0*b0,1 + a2,1*b1,1

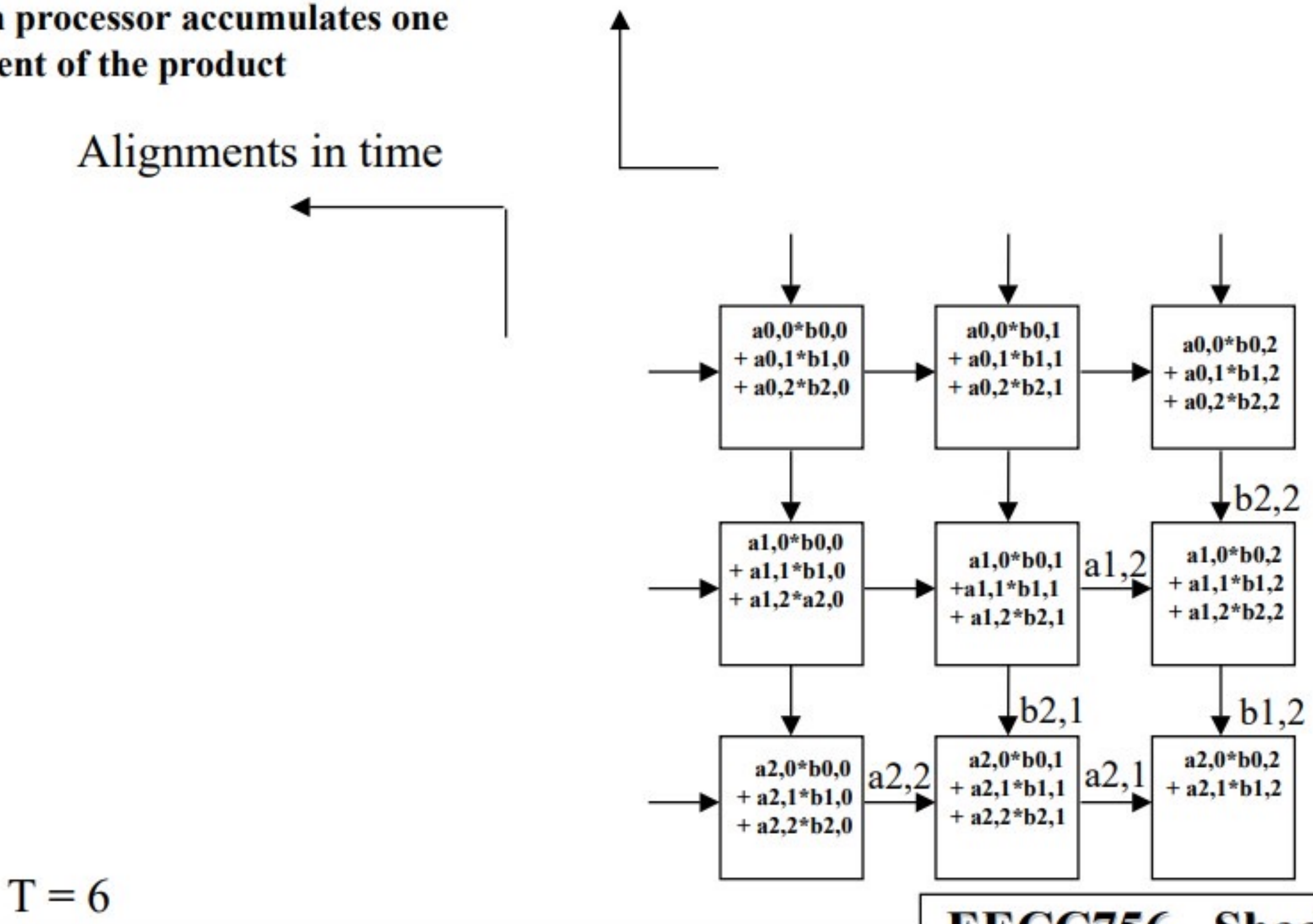
a2,0*b0,2

T = 5

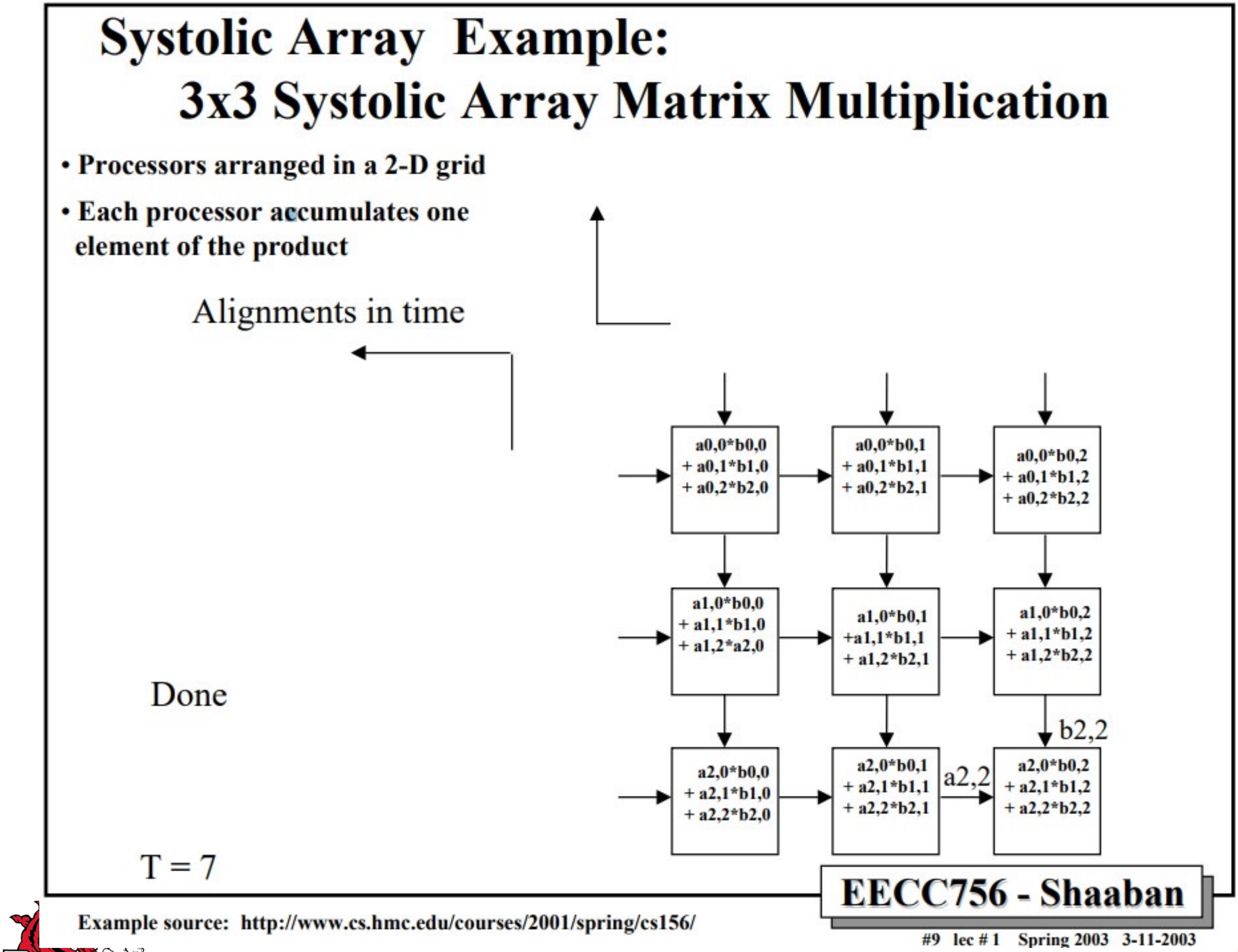
Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

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



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

a0,0*b0,0 + a0,1*b1,0 + a0,2*b2,0	a0,0*b0,1 + a0,1*b1,1 + a0,2*b2,1	a0,0*b0,2 + a0,1*b1,2 + a0,2*b2,2
a1,0*b0,0 + a1,1*b1,0 + a1,2*a2,0	a1,0*b0,1 +a1,1*b1,1 + a1,2*b2,1	a1,0*b0,2 + a1,1*b1,2 + a1,2*b2,2
a2,0*b0,0 + a2,1*b1,0 + a2,2*b2,0	a2,0*b0,1 + a2,1*b1,1 + a2,2*b2,1	a2,0*b0,2 + a2,1*b1,2 + a2,2*b2,2

Done

T = 7

EECC756 - Shaaban

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

#9 lec # 1 Spring 2003 3-11-2003

The TPU and the Guidelines

- Use dedicated memories
  - 24 MiB dedicated buffer, 4 MiB accumulator buffers
- Invest resources in arithmetic units and dedicated memories
  - 60% of the memory and 250X the arithmetic units of a server-class CPU
- Use the easiest form of parallelism that matches the domain
  - Exploits 2D SIMD parallelism
- Reduce the data size and type needed for the domain
  - Primarily uses 8-bit integers
- Use a domain-specific programming language
  - Uses TensorFlow