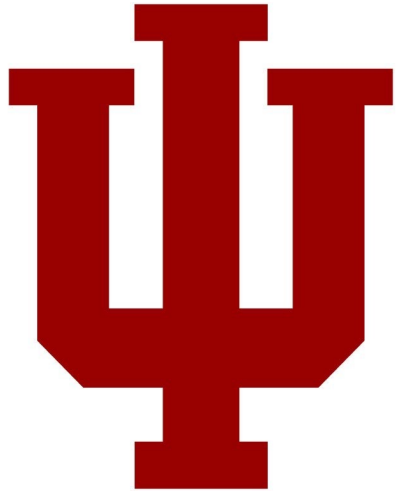
18: Parallelism I

ENGR 315: Hardware/Software CoDesign

Andrew Lukefahr

Indiana University

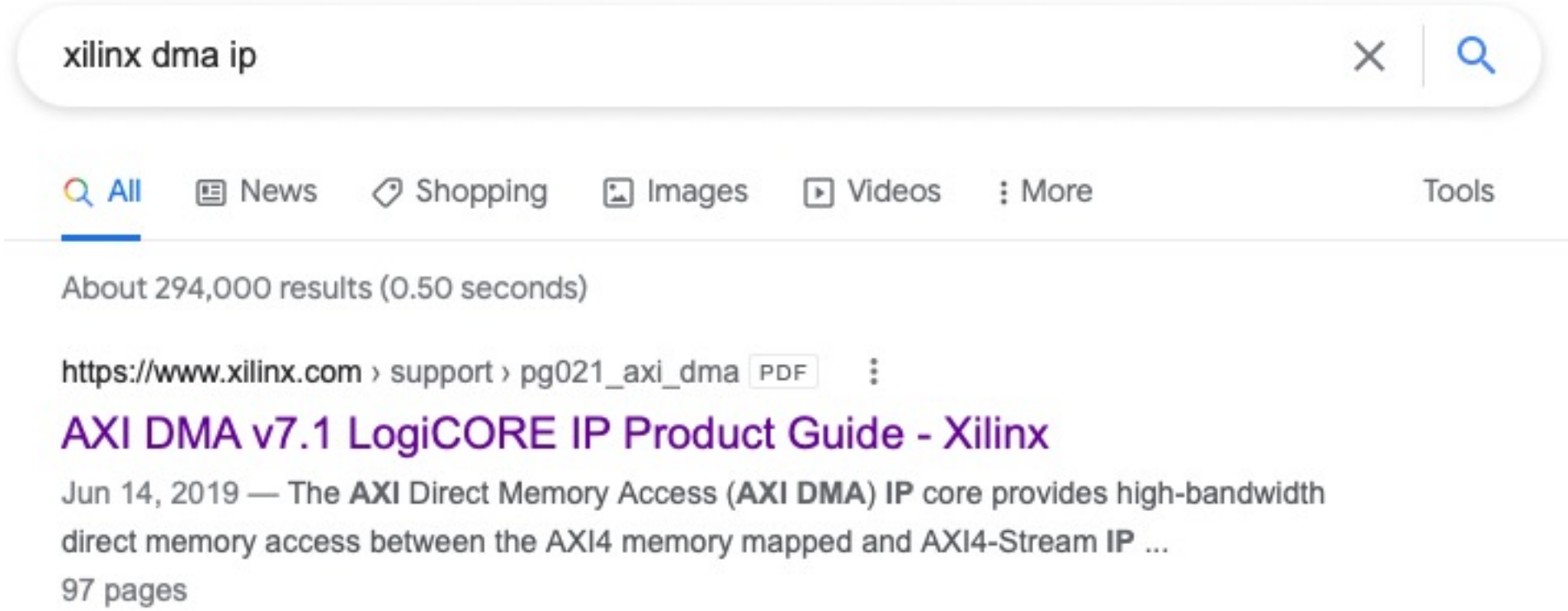


Some material taken from:
https://github.com/trekhleb/homemade-machine-learning/tree/master/homemade/neural_network
http://cs231n.github.io/neural-networks-1/

Announcements

- P7: Due _____.
- Review: Wednesday, Nov 1st
- Exam: Monday, Nov 7th
- P8: Coming Soon.

P7 – DMA from C



https://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v7_1/pg021_axi_dma.pdf

P7 – DMA from C

Programming Sequence

Direct Register Mode (Simple DMA)

P7 – DMA from C

1. Start the MM2S channel running by setting the run/stop bit to 1 (MM2S_DMACR.RS = 1). The halted bit (DMASR.Halted) should deassert indicating the MM2S channel is running.
2. Skip
3. Write a valid source address to the MM2S_SA register.
4. Write the number of bytes to transfer in the MM2S_LENGTH register.

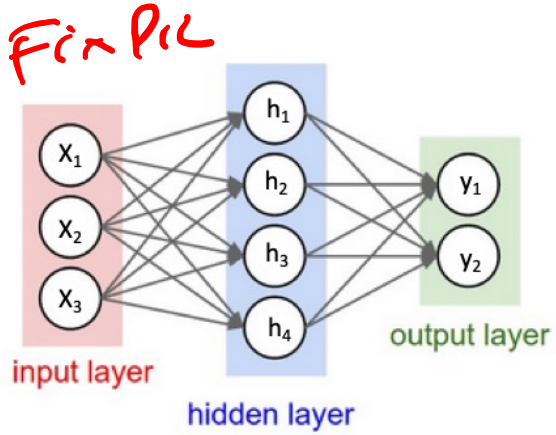
   The MM2S_LENGTH register must be written last.

5. Wait until MM2S_DMASR.Idle==1 for completion

P8+ are part of the “Design Challenge”

- Goal: Accelerate reference neural network
- Harder, more open-ended projects

P8+ focus on Dot Product




```
# forward-pass of a 3-layer neural network:
f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (use sigmoid)
x = np.random.randn(3, 1) # random input vector of three numbers (3x1)
h1 = f(np.dot(W1, x) + b1) # calculate first hidden layer activations (4x1)
h2 = f(np.dot(W2, h1) + b2) # calculate second hidden layer activations (4x1)
out = np.dot(W3, h2) + b3 # output neuron (1x1)
```

Matrix Multiplication (Dot Product)

inputs

$$\begin{bmatrix} 0.1 & 0.2 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} =$$

weights

$$= \begin{bmatrix} (0.1 \times 1 + 0.2 \times 4) & (0.1 \times 2 + 0.2 \times 5) & (0.1 \times 3 + 0.2 \times 6) \end{bmatrix}$$

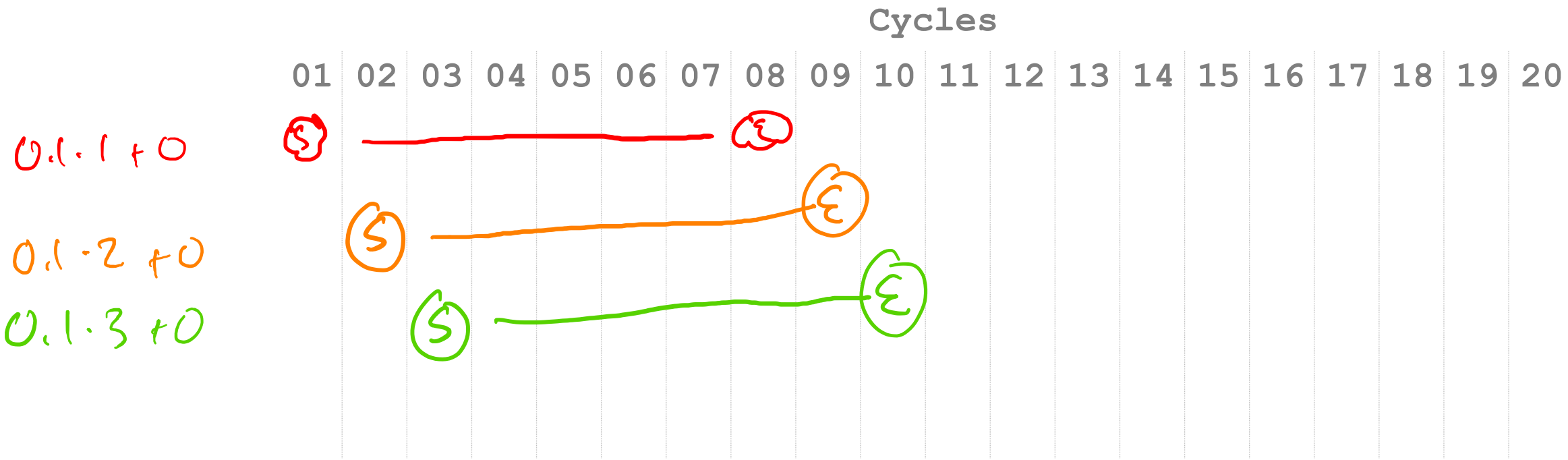
(Answer)

$$= \begin{bmatrix} 0.9 & 1.2 & 1.5 \end{bmatrix}$$

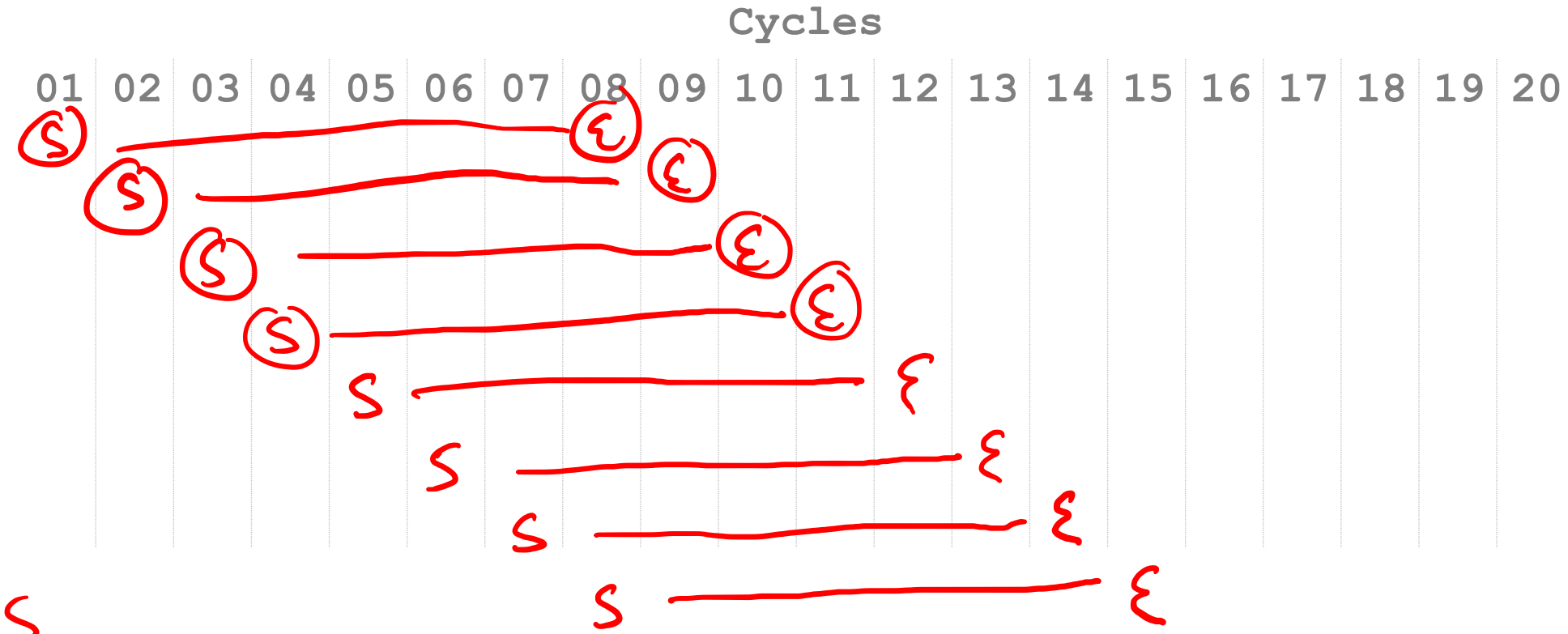
Floating-Point math takes 8 cycles.

- Floating-Point is complicated.
- FMAC: a * b + c
- 8 cycles of complicated.
- How do we work around an 8 cycle latency?
- Pipelining!

FMAC Pipelining



FMAC Pipelining



individual:
8 cycles

Average:
8 mults / 15 cycles

~2 cycles/mult.

~1/2 mult / cycle

Latency vs. Throughput

- **Latency**: How long does an individual operation take to complete?

individual mult: 8 cycles

- **Throughput**: How many operations can you complete per second (or per cycle)?

Average output per cycles:

~ 1 mult / cycle

Multiply-Accumulate Dot Computations

$$[0.1 \quad 0.2] \times \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} = [0.9 \quad 1.2 \quad 1.5]$$

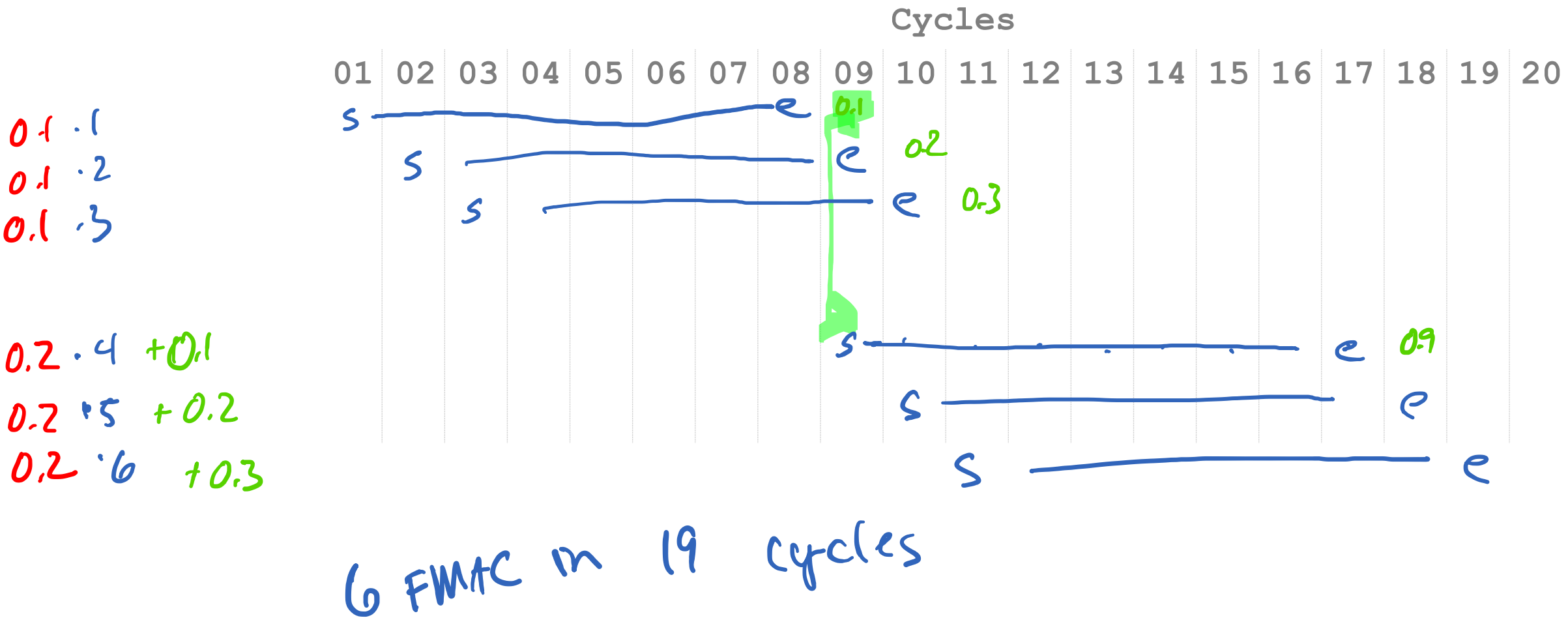
$$= [0 \quad 0 \quad 0]$$

$0.1 \cdot 1 + 0$ $\quad$ $0.1 \cdot 2 + 0$ $\quad$ $0.1 \cdot 3 + 0 = [0.1 \quad 0.2 \quad 0.3]$

$0.2 \cdot 4 + 0.1$ $\quad$ $0.2 \cdot 5 + 0.2$ $\quad$ $0.2 \cdot 6 + 0.3 = [0.9 \quad 1.2 \quad 1.5]$

Dependencies

$$\begin{bmatrix} 0.1 & 0.2 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$



Dependencies on Pipelined FMAC

- Solution: Stall at the end of a row.
- “Drain” the pipeline.

- Restart new row
- “Refill” the pipeline

How many cycles *should* this take?

$$\begin{bmatrix} 0.1 & 0.2 & 0.3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} 3.8 & 4.4 & 5 & 5.6 \end{bmatrix}$$

38

26/27

infinite HW

8

32

24

Worst case HW

12-8 = 96

40 cycles / column

= 160 cycles

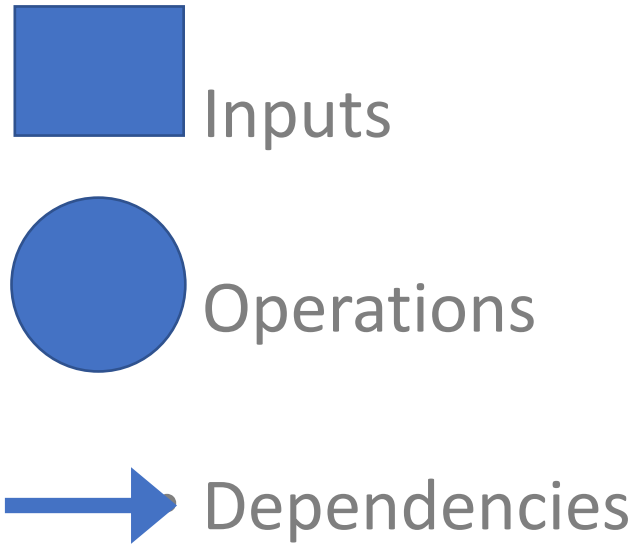
Data flow graph

- A data-flow graph is **a collection of arcs and nodes in which the nodes are either places where variables are assigned or used**, and the arcs show the relationship between the places where a variable is assigned and where the assigned value is subsequently used.

[ref]

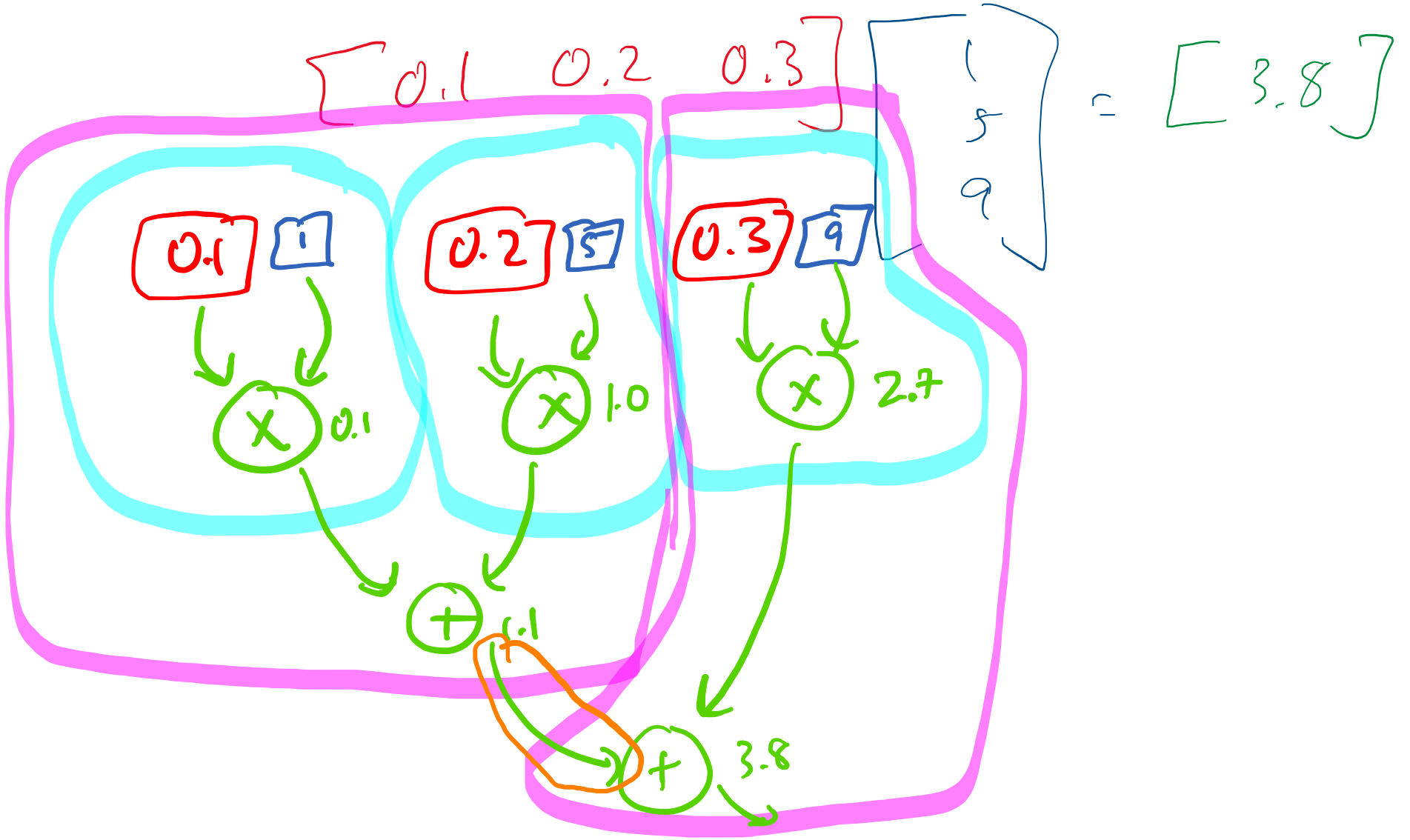
Data Flow Graph

- Illustrates the dependencies between inputs and operations to produce an output



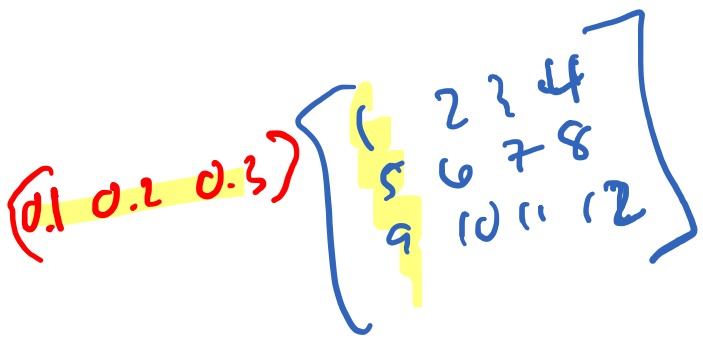
Data Flow Graph

* NO Pipelining
* NO MAC (A*B or A+B)



Finding Parallelism

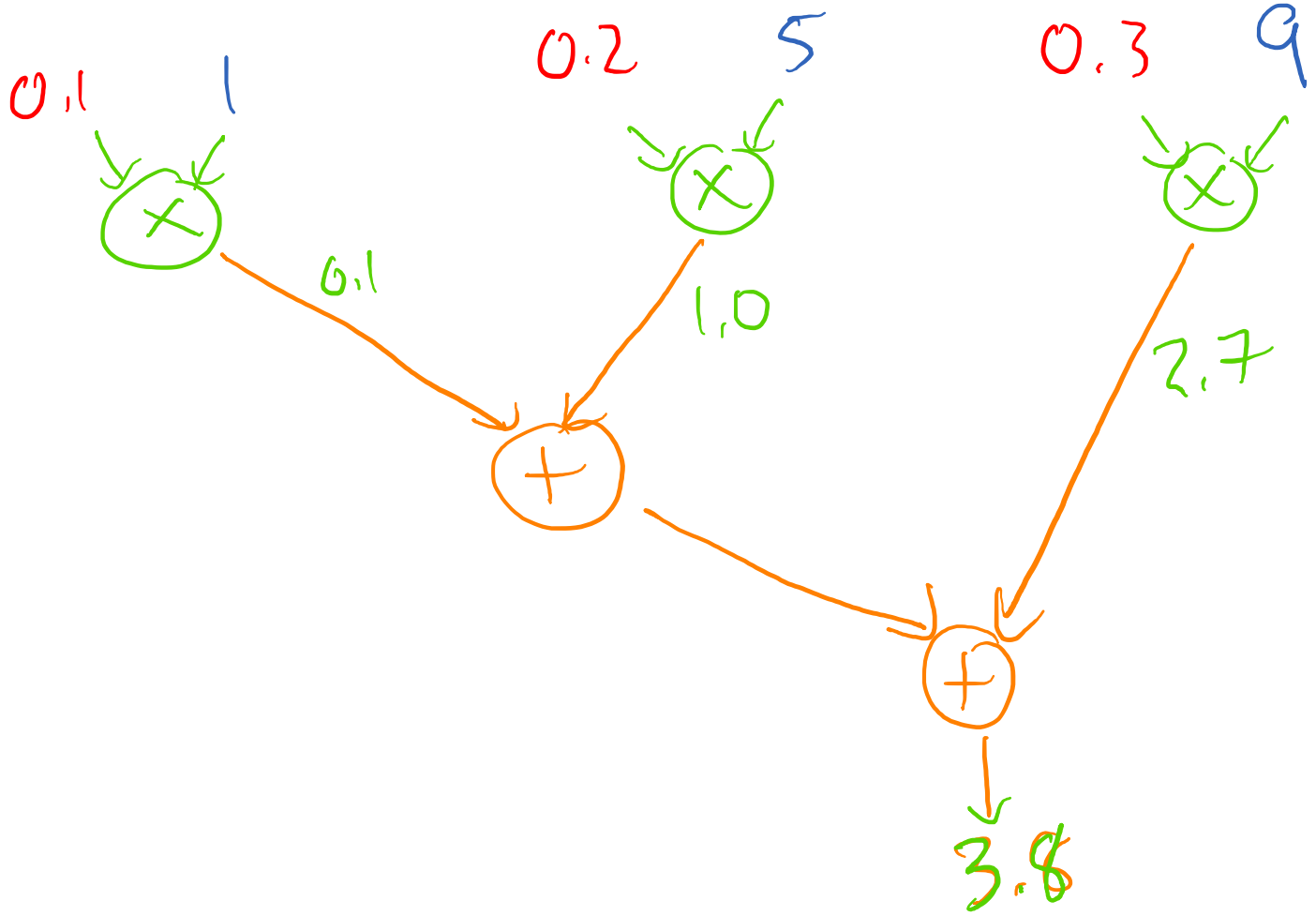
- Find some computation that doesn't depend on other computation's results.
- No arrows between blocks.
- Shared Inputs are OK.



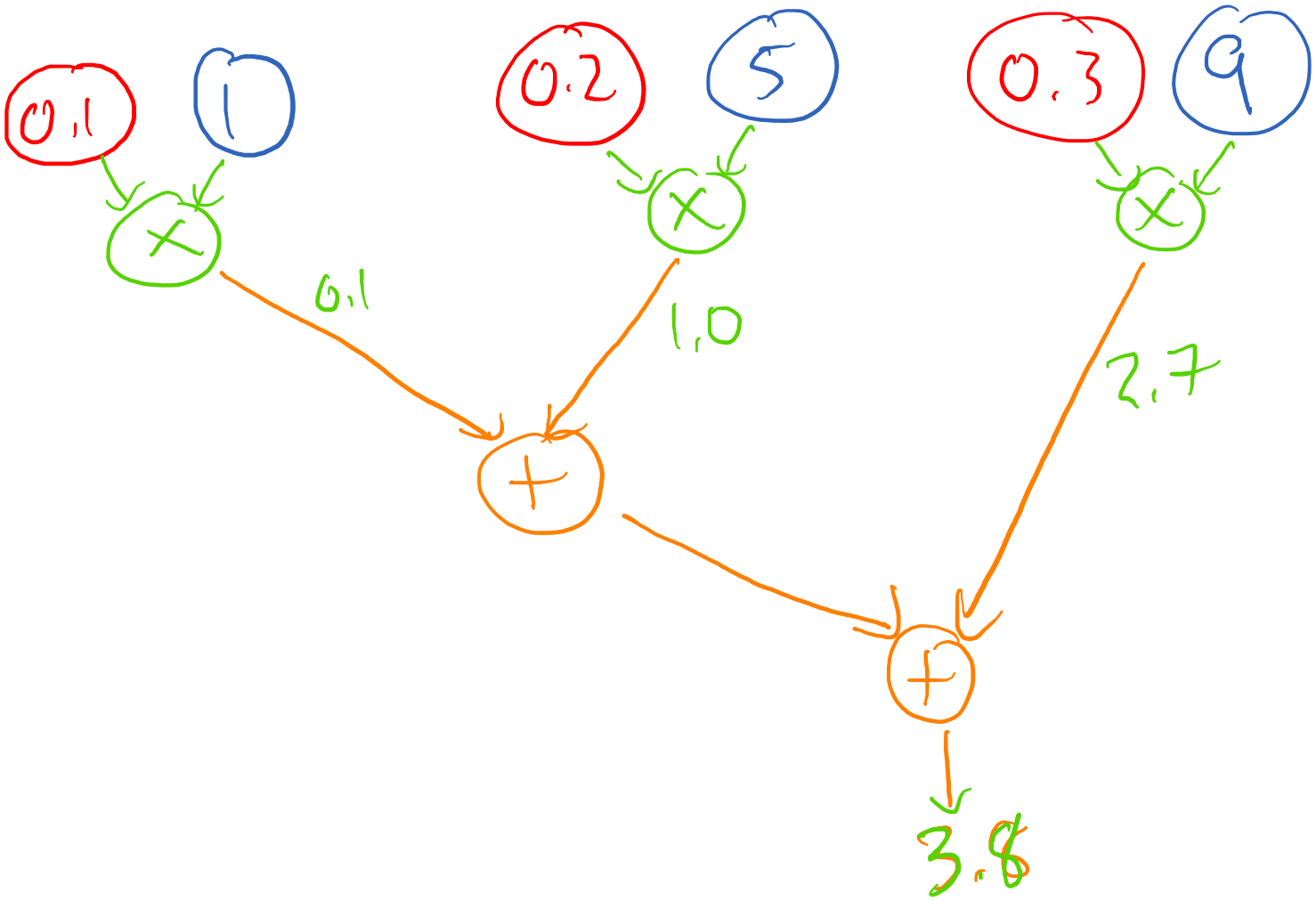
24 cycles

96 cycles w/ 3 mults

$$\begin{bmatrix} 0.1 & 0.2 & 0.3 \end{bmatrix} \begin{bmatrix} 1 \\ 5 \\ 9 \end{bmatrix} = \begin{bmatrix} 3.8 \end{bmatrix}$$

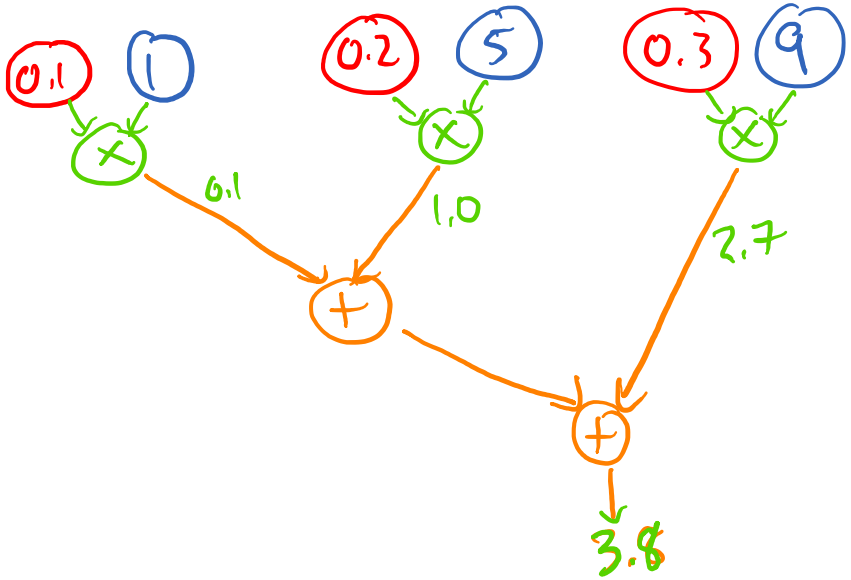


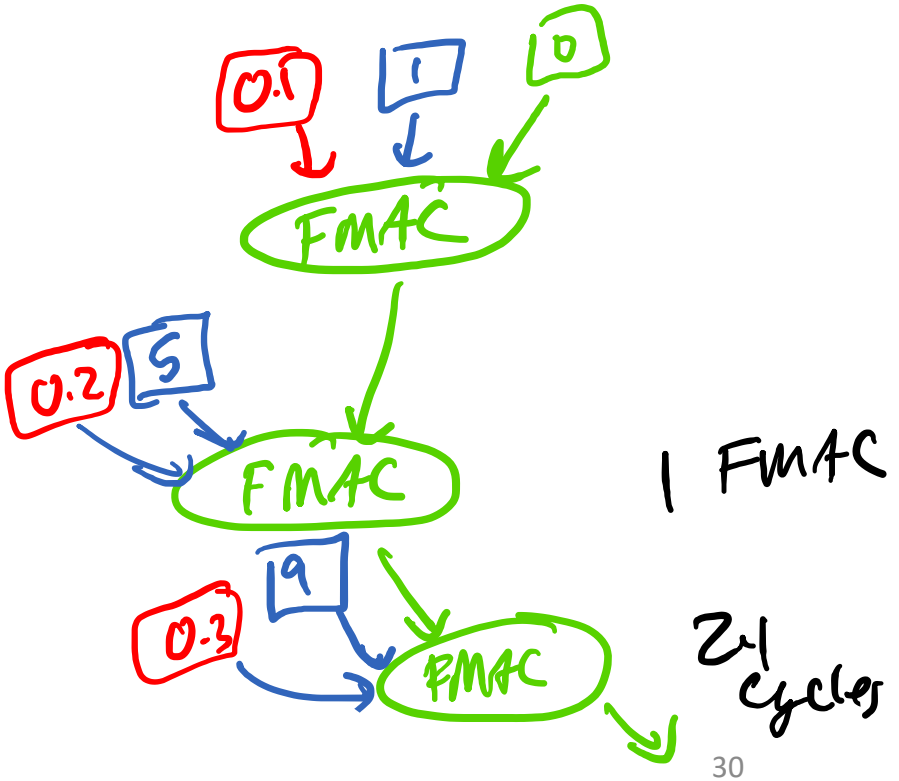
How many cycles *should* this take?



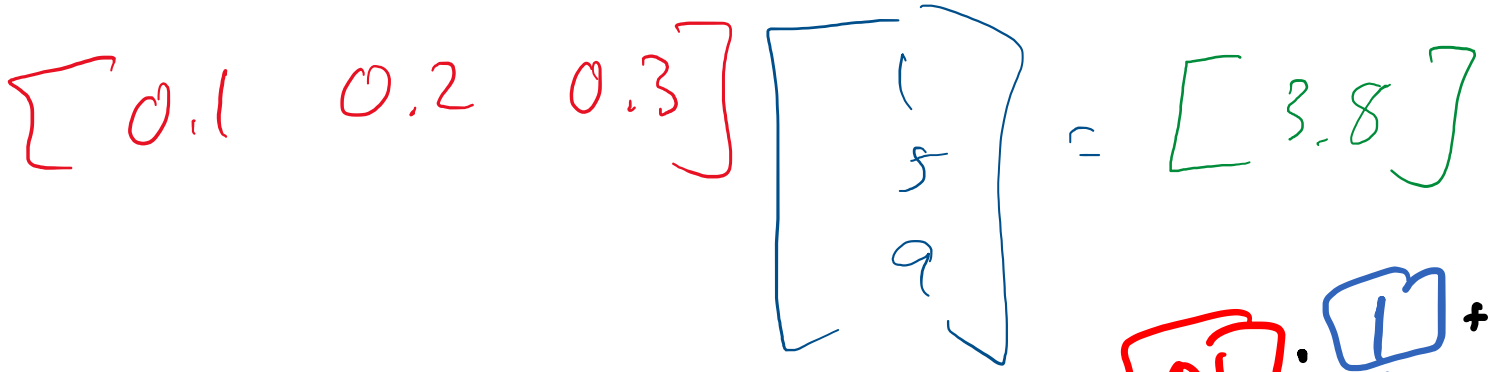
How many cycles *should* this take?

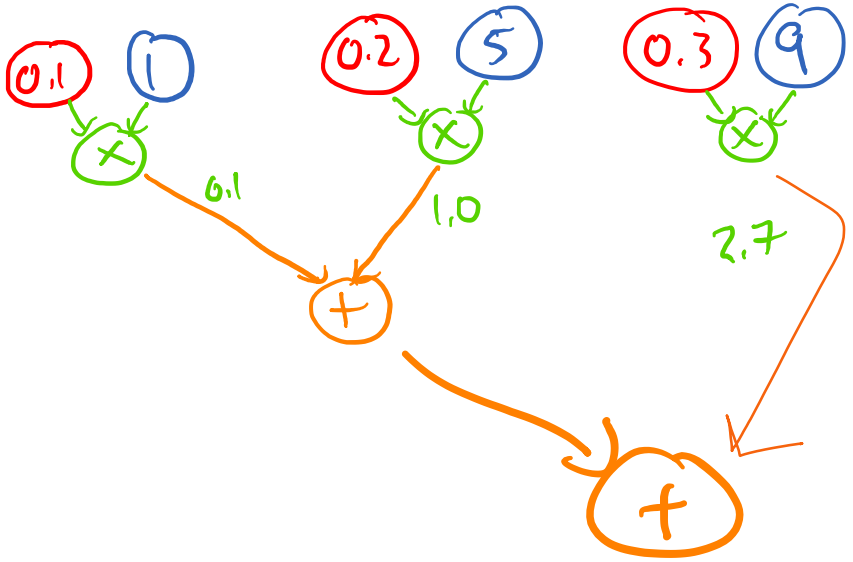
$$\begin{bmatrix} 0.1 & 0.2 & 0.3 \end{bmatrix} \begin{bmatrix} 1 \\ 5 \\ 9 \end{bmatrix} = \begin{bmatrix} 3.8 \end{bmatrix}$$

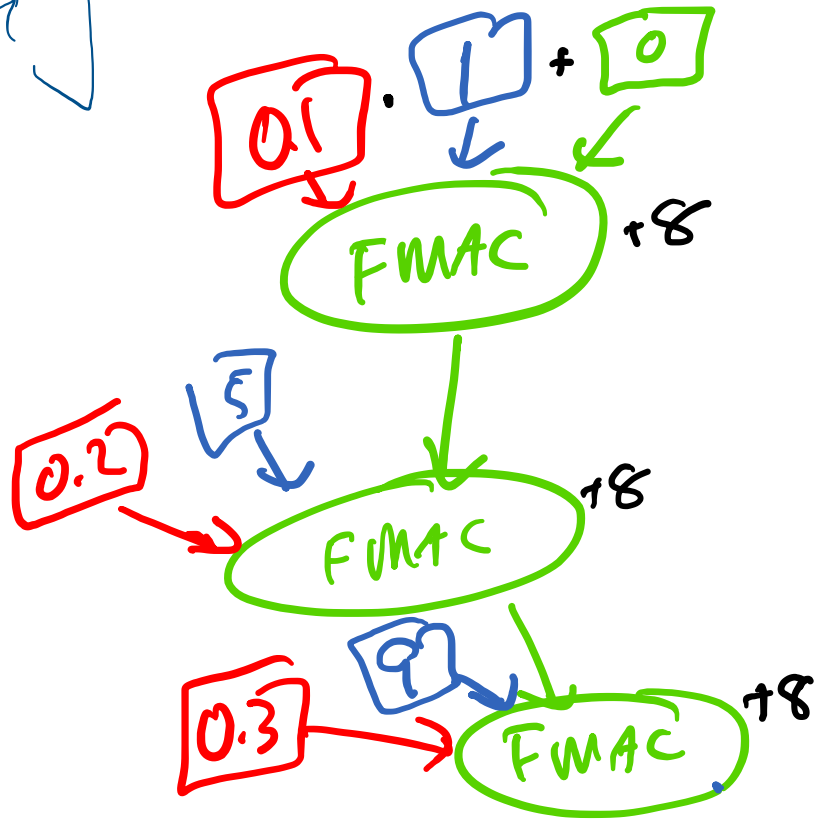




How many cycles *should* this take?

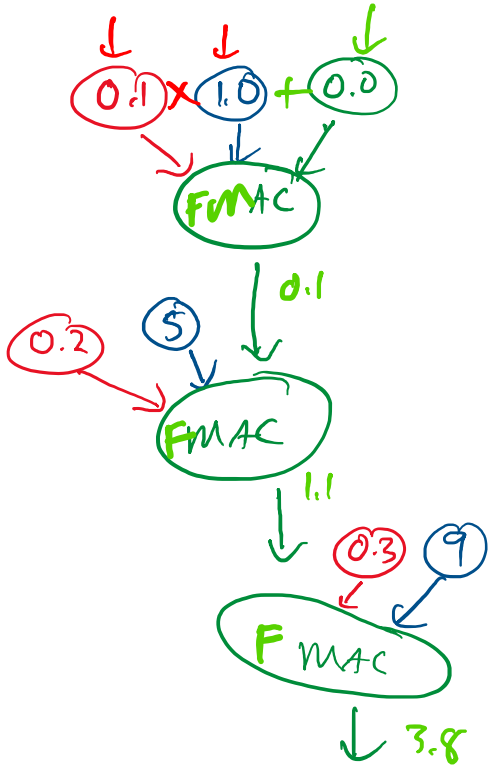


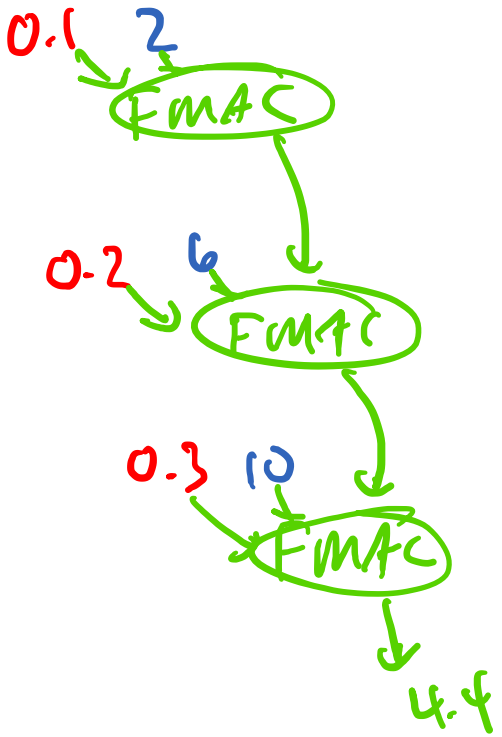




More Data Flow

$$\begin{bmatrix} 0.1 & 0.2 & 0.3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} 3.8 & 4.4 & 5 & 5.6 \end{bmatrix}$$



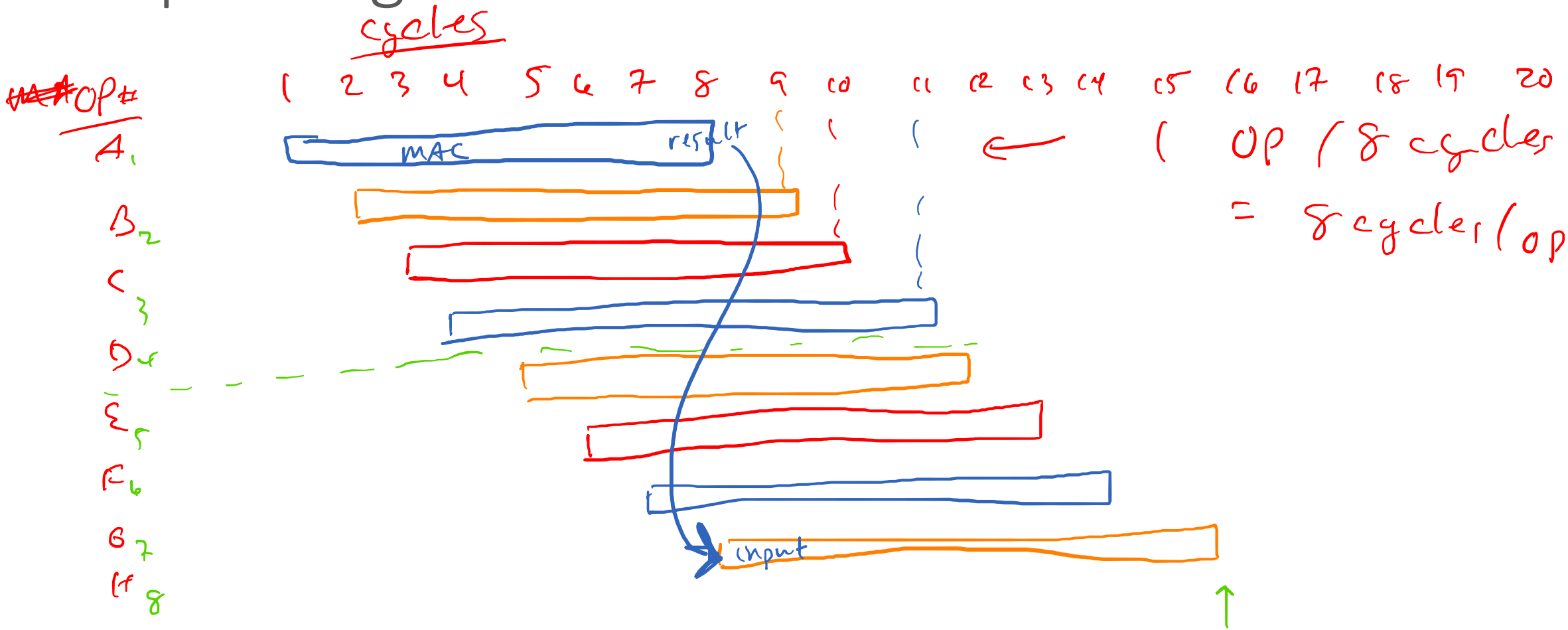


Pipelining

- FMAC takes 8 cycles for 1 value
- But can accept a new value every cycle.

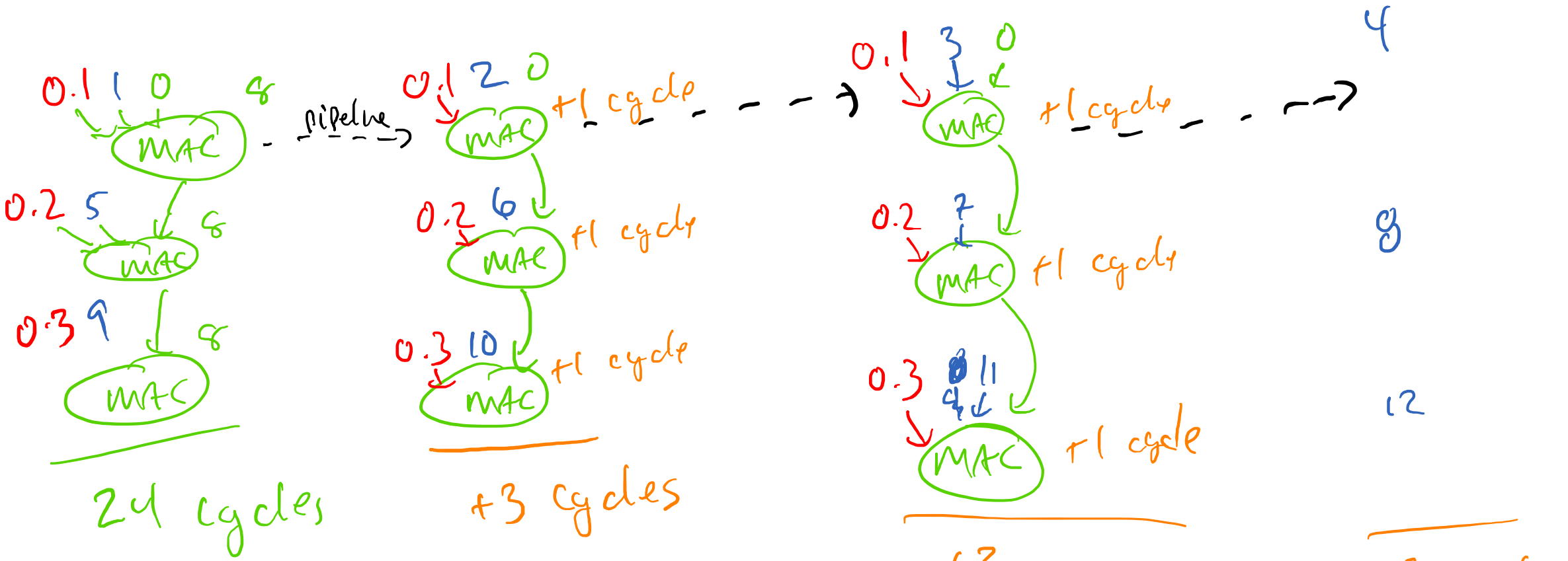
- Latency: 8 cycles / value
- Throughput: 1 value / cycle

Pipelining


cycles
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
OP#
A1
B2
C3
D4
E5
F6
G7
H8
MAC
result
input
1 OP / 8 cycles
= 8 cycles / OP
8 ops / 16 cycles
= 2 cycles / OP
16 cycles
8 cycles each

$$\begin{bmatrix} 0.1 & 0.2 & 0.3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} 3.8 & 4.4 & 5 & 5.6 \end{bmatrix}$$

$$\begin{bmatrix} 0.1 & 0.2 & 0.3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} 3.8 & 4.4 & 5 & 5.6 \end{bmatrix}$$



$24 + 3 + 3 + 3 = 33$ cycles

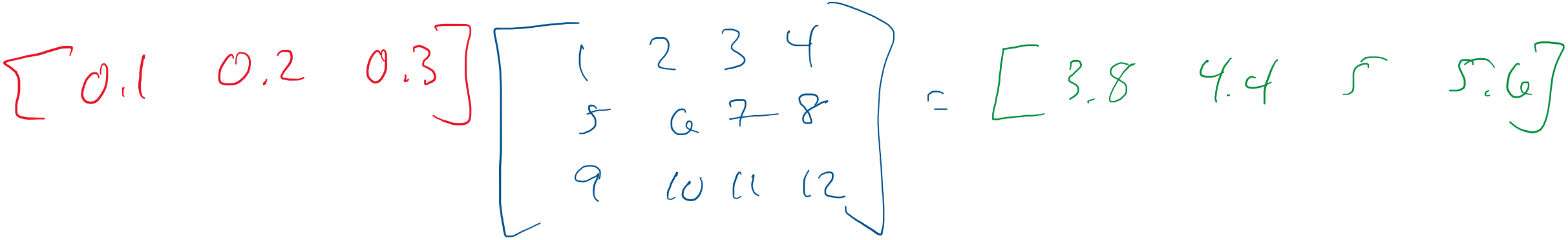
$$\begin{bmatrix} 0.1 & 0.2 & 0.3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} 3.8 & 4.4 & 5 & 5.6 \end{bmatrix}$$

Fix me.

33 vs 27 cycles?

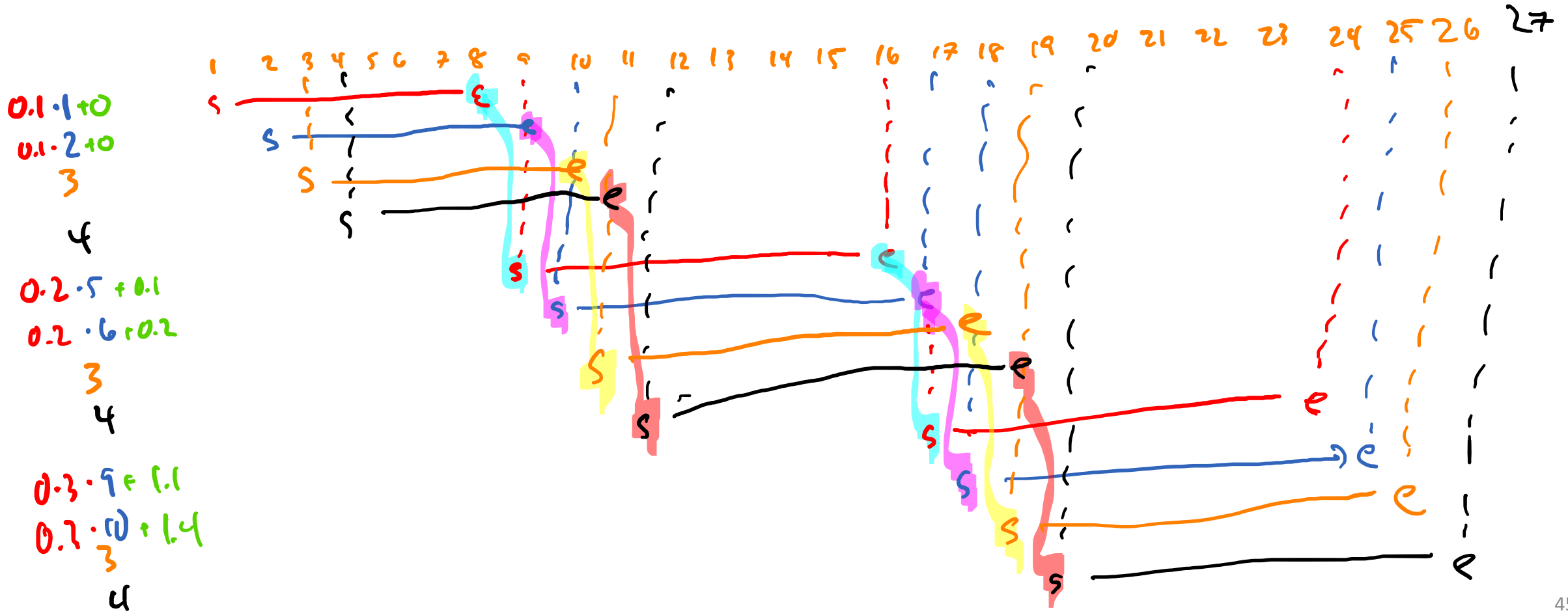
$$\begin{bmatrix} 0.1 & 0.2 & 0.3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} 3.8 & 4.4 & 5 & 5.6 \end{bmatrix}$$

	One Output	All 4 Outputs
Multiply & Add separate:	40	160
MAC	24 (x1)	96
Pipelined MAC	24 (+3)	27
Maximum Parallel:	24	24



	One Output	All 4 Outputs
Multiply & Add separate:	40 cycles	160 cycles
MAC	24 cycles	96 cycles
Pipelined MAC	24 cycles (+1 for extra rows)	33 cycles
Maximum Parallel:	24 cycles	24 cycles

Next Time: Divide and Conquer



$$\begin{bmatrix} 0.1 & 0.2 & 0.3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} 3.8 & 4.4 & 5 & 5.6 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$$

$$0.1 \cdot \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \end{bmatrix}$$

$$0.2 \cdot \begin{bmatrix} 5 & 6 & 7 & 8 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \end{bmatrix} = \begin{bmatrix} 1.1 & 1.4 & 1.7 & 2.0 \end{bmatrix}$$

$$0.3 \cdot \begin{bmatrix} 9 & 10 & 11 & 12 \end{bmatrix} + \begin{bmatrix} 1.1 & 1.4 & 1.7 & 2.0 \end{bmatrix} = \begin{bmatrix} 3.8 & 4.4 & 5 & 5.6 \end{bmatrix}$$