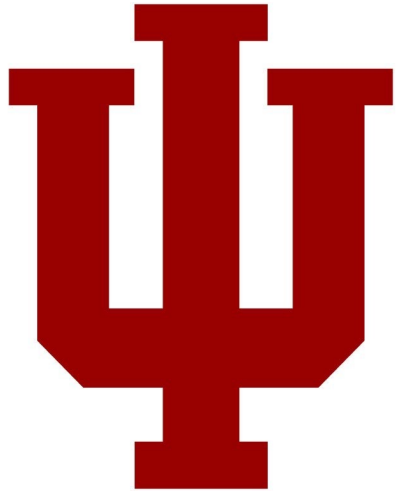
20: Parallelism III

ENGR 315: Hardware/Software CoDesign

Andrew Lukefahr

Indiana University

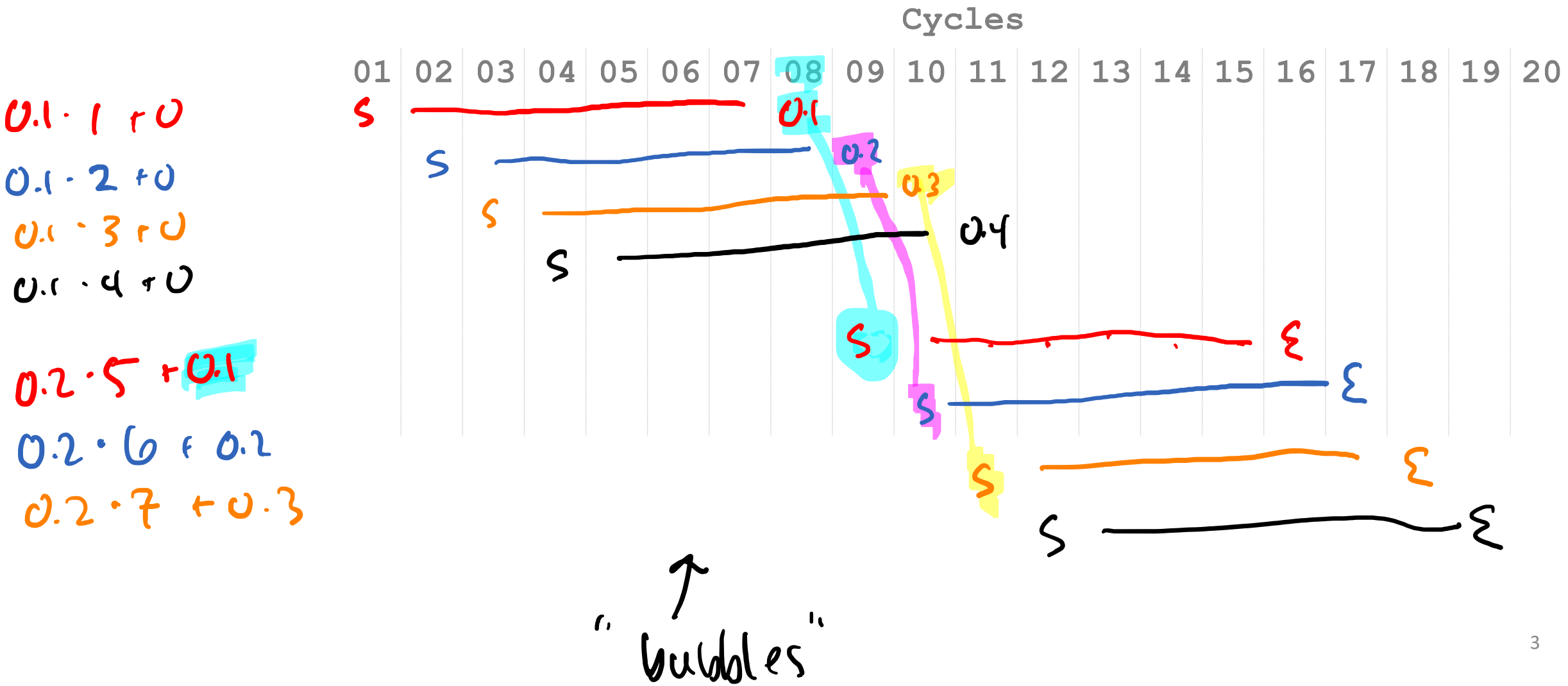


Announcements

- P8: Due Friday
- P9: Parallelism
- Report: Coming up

```
Input                                Weights                          Output
[[0.1 0.2 0.3]]          .           [1. 2. 3. 4.]            =       [3.8000002 4.4       5.        5.6000004]
                                     [5. 6. 7. 8.]
                                     [ 9. 10. 11. 12.]
```

Dependencies



Verilog Parameters

```
module adder #(parameter BITS = 2) (
       input [BITS-1:0] a,
      input [BITS-1:0] b,
      output [BITS-1:0 c) );
      assign c = a + b;
endmodule
```

```
// 2-bit adder
adder add2 (a2, b2, c2);
//another 2-bit adder
adder #(BITS=2) add2b (a2b,b2b,c2b);
//a 3-bit adder
adder #(BITS=3) add3 (a3, b3, c3);

Adder #(BITS=32) add32 (a32, b32,
c32);
```

data-dependence graph
(data-flow graph)

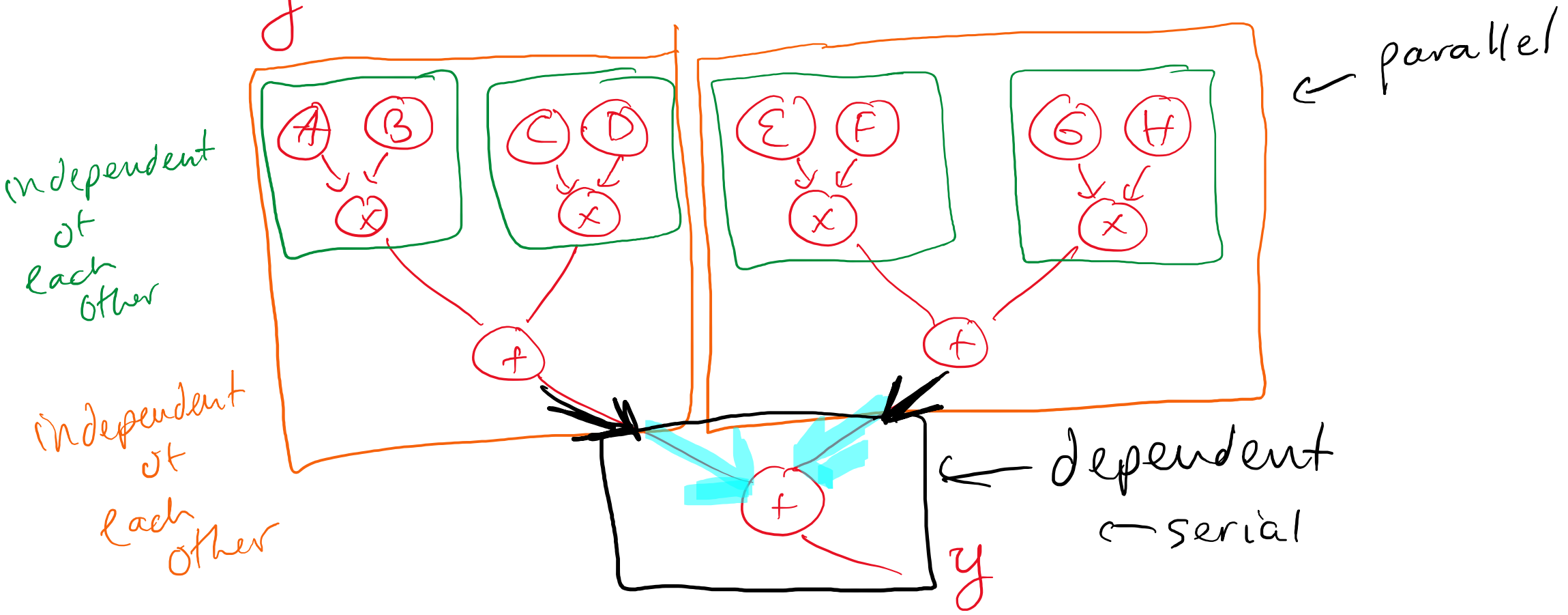
$$Z = C(A+B) - D \cdot E$$

data-dependence graph
(data-flow graph)

$$y = A \cdot B + C \cdot D + E \cdot F + G \cdot H$$

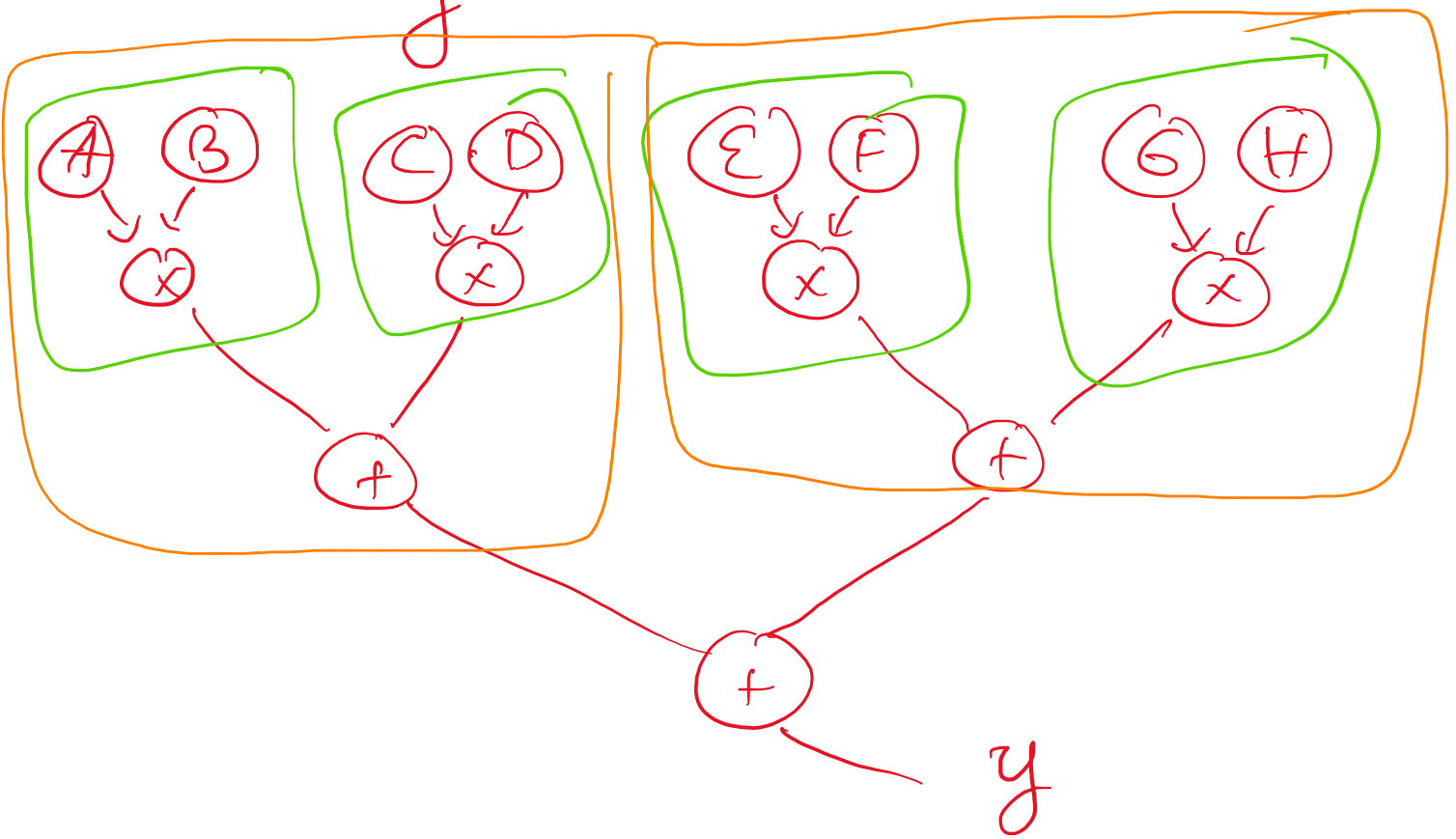
data-dependence graph
(data-flow graph)

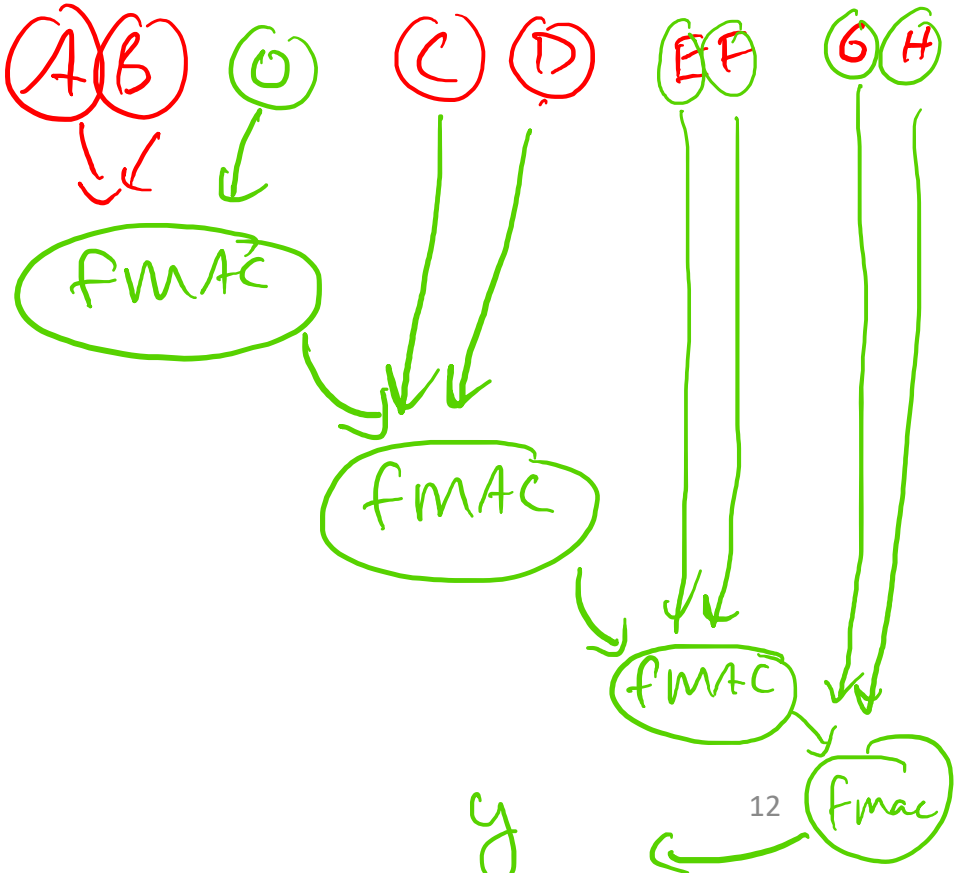
$y = A \cdot B + C \cdot D + E \cdot F + G \cdot H$



data-dependence graph (data-flow graph)

$y = A \cdot B + C \cdot D + E \cdot F + G \cdot H$

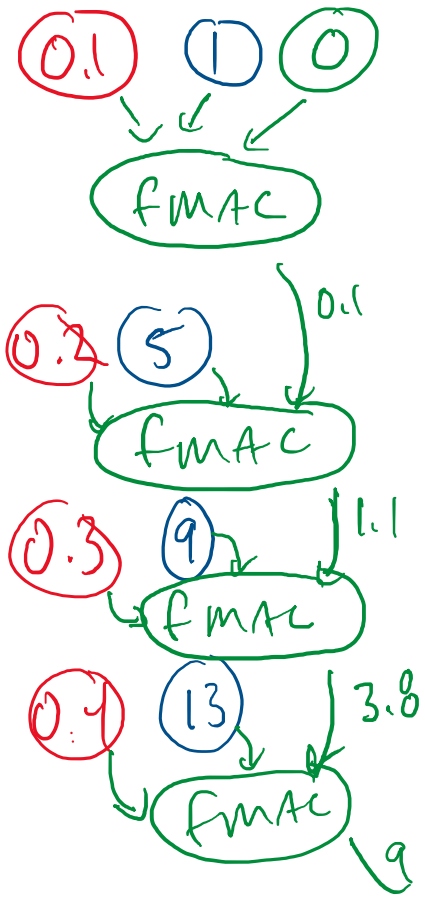


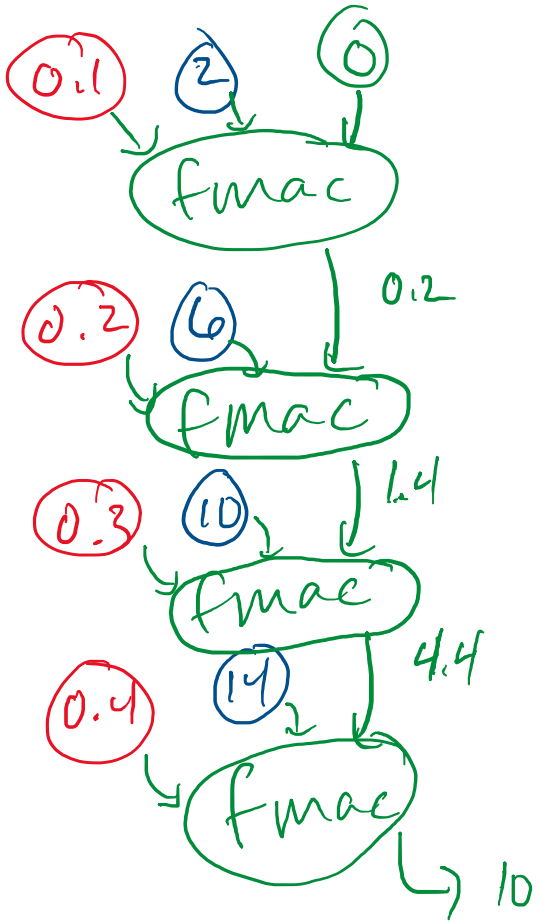


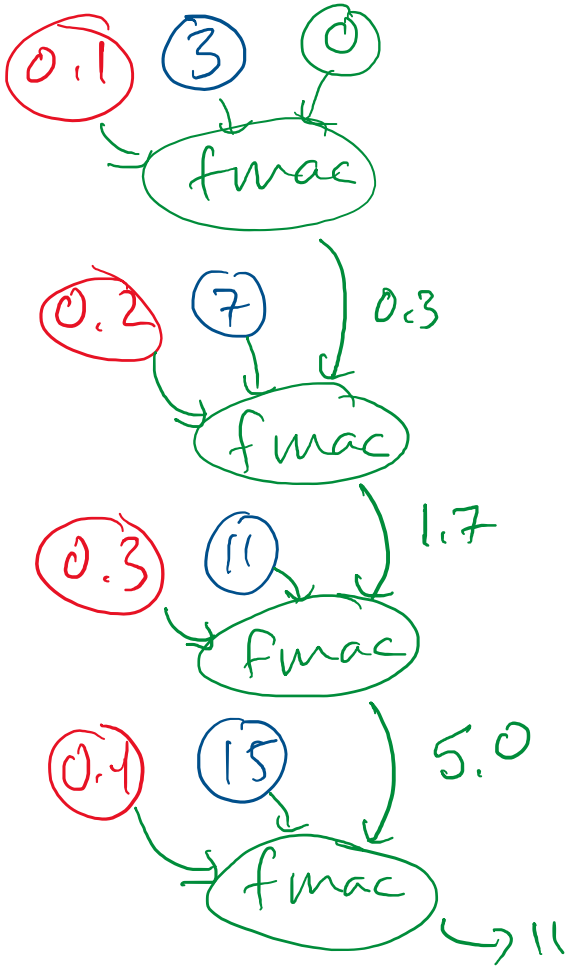
$$y = A \cdot B + C \cdot D + E \cdot F + G \cdot H$$

$$\begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} = \begin{bmatrix} 9 & 10 & 11 & 12 \end{bmatrix}$$

↑ outputs are independent







4th output not shown

$$\begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} = \begin{bmatrix} 9 & 10 & 11 & 12 \end{bmatrix}$$

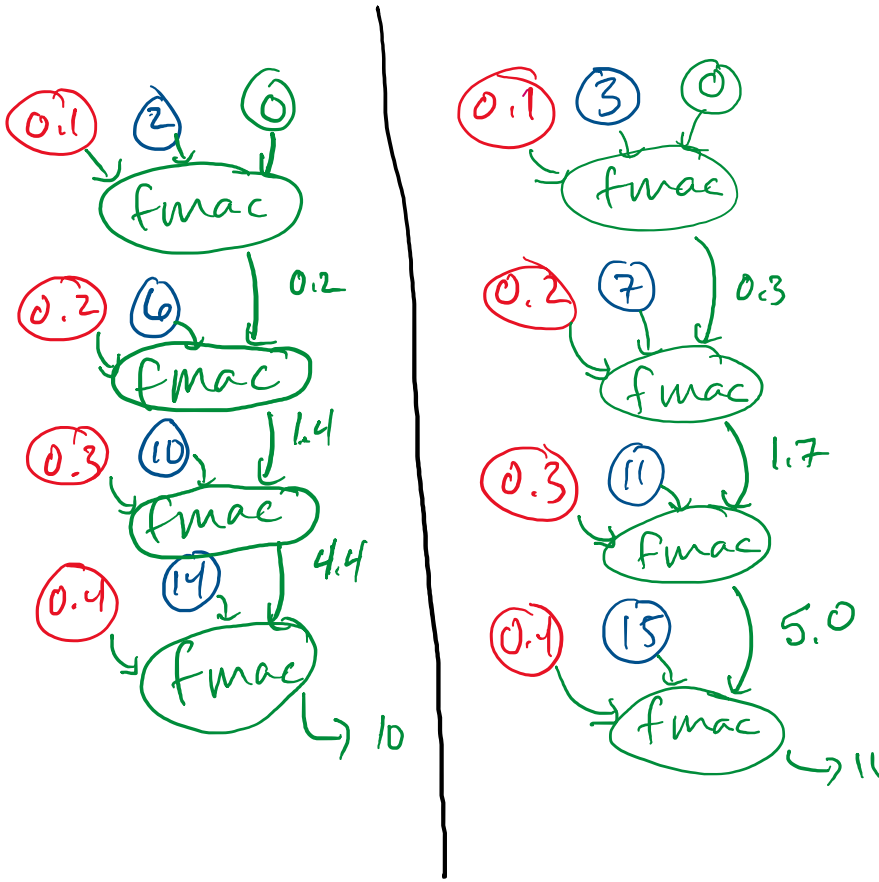
$$\begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} = \begin{bmatrix} 9 & 10 & 11 & 12 \end{bmatrix}$$

$$\begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} = \begin{bmatrix} 9 & 10 & 11 & 12 \end{bmatrix}$$

$$\begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 5 & 6 \\ 9 & 10 \\ 13 & 14 \end{bmatrix} = \begin{bmatrix} 9 & 10 \end{bmatrix}$$

$$\begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \end{bmatrix} \begin{bmatrix} 3 & 4 \\ 7 & 8 \\ 11 & 12 \\ 15 & 16 \end{bmatrix} = \begin{bmatrix} 11 & 12 \end{bmatrix}$$

$$\begin{bmatrix}0.1 & 0.2 & 0.3 & 0.4\end{bmatrix} \cdot \left[\begin{array}{cc|cc} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{array}\right] = \begin{bmatrix} 9 & 10 & 11 & 12 \end{bmatrix}$$



$$\begin{bmatrix}0.1 & 0.2 & 0.3 & 0.4\end{bmatrix} \begin{bmatrix} 1 & 2 \\ 5 & 6 \\ 9 & 10 \\ 13 & 14 \end{bmatrix} = \begin{bmatrix} 9 & 10 \end{bmatrix}$$

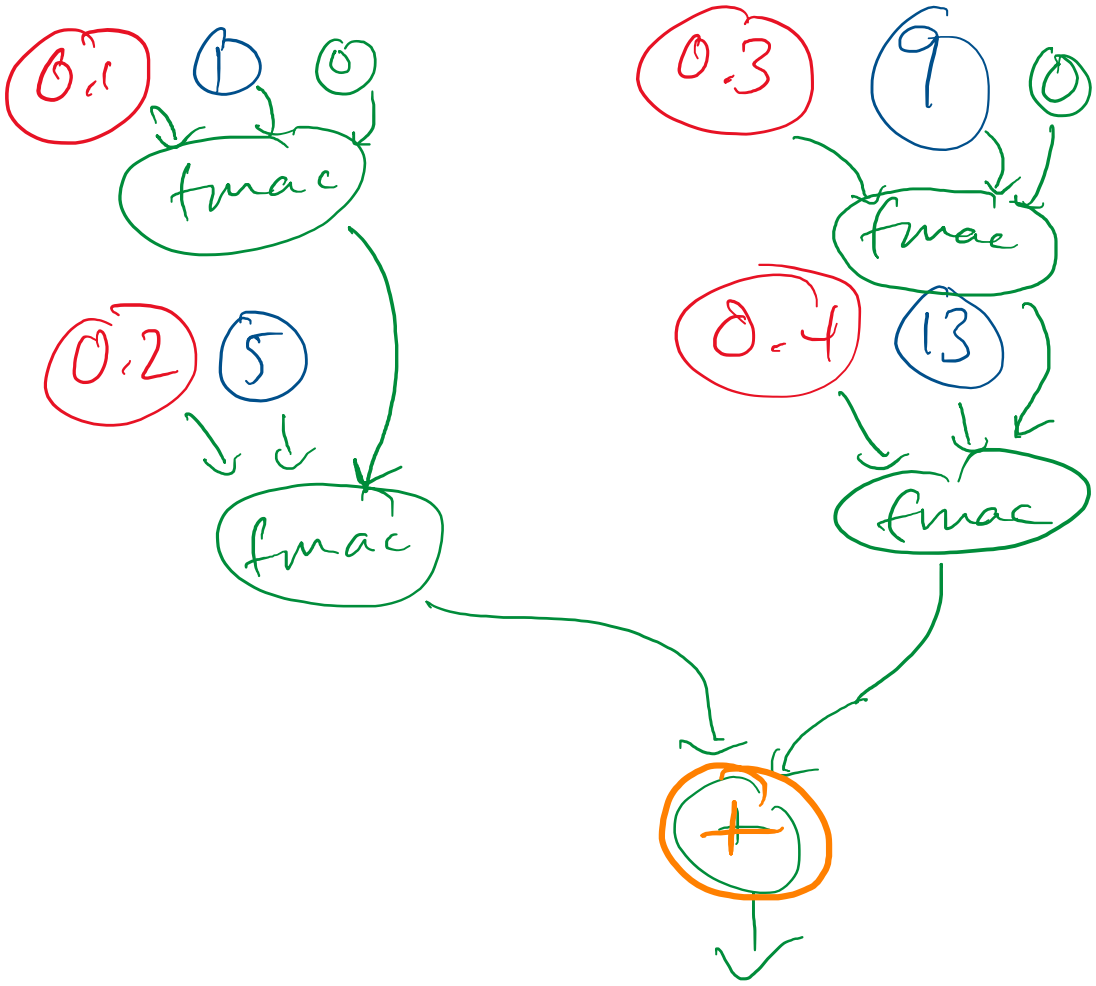
$$\begin{bmatrix}0.1 & 0.2 & 0.3 & 0.4\end{bmatrix} \begin{bmatrix} 3 & 4 \\ 7 & 8 \\ 11 & 12 \\ 15 & 16 \end{bmatrix} = \begin{bmatrix} 11 & 12 \end{bmatrix}$$

$$[0.1 \quad 0.2 \quad 0.3 \quad 0.4] \cdot \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} = [9 \quad 10 \quad 11 \quad 12]$$

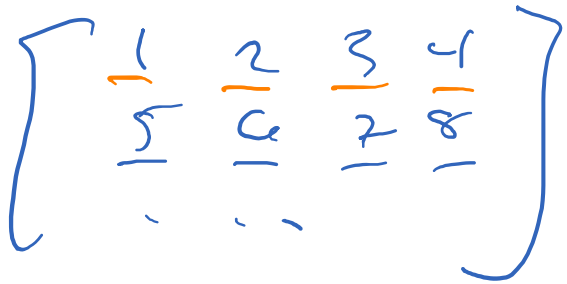
$$[0.1 \quad 0.2] \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

$$+ [0.3 \quad 0.4] \begin{bmatrix} 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

$$[9 \quad 10 \quad 11 \quad 12]$$



Can we parallelize Dot?




```
# how its done in dot.sv
def pydot(inputs,weights):
    inputs = inputs[0] # remove outer nesting
    outs = np.zeros(weights.shape[1], dtype=np.float32)
    for i in range(weights.shape[0]): # input length
        for j in range(weights.shape[1]): # output length
            outs[j] = outs[j] + weights[i][j] * inputs[i]
    return outs
```

0.1 0.2 0.3 0.4

```
weights = np.array( [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]], dtype=np.float32)
inputs = np.array([[0.1,0.2,0.3,0.4]], dtype=np.float32)
outputs = np.dot(inputs, weights)
```

```
Input                    Weights                        Output
[0.1 0.2 0.3 0.4]    .   [1. 2. 3. 4.]          =   [ 9. 10. 11. 12.]
                         [5. 6. 7. 8.]
                         [ 9. 10. 11. 12.]
                         [13. 14. 15. 16.]
```

```
outputs1 = np.dot([[0.1,0.2,0.3, 0.4]], [[1,2],[5,6],[9,10],[13,14]])
outputs2 = np.dot([[0.1,0.2,0.3, 0.4]], [[3,4],[7,8],[11,12],[15,16]])
outputs = np.concatenate((outputs1, outputs2), axis=1)
```

```
[[ 9. 10.]]
[[11. 12.]]
[[ 9. 10. 11. 12.]]
```

```
weights = np.array( [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]], dtype=np.float32)
inputs = np.array([[0.1,0.2,0.3,0.4]], dtype=np.float32)
outputs = np.dot(inputs, weights)
```

```
Input                    Weights                    Output
[0.1 0.2 0.3 0.4]    .   [1. 2. 3. 4.]          =   [ 9. 10. 11. 12.]
                         [5. 6. 7. 8.]
                         [ 9. 10. 11. 12.]
                         [13. 14. 15. 16.]
```

```
outputs1 = np.dot([[0.1,0.2,0.3, 0.4]], [[1,2],[5,6],[9,10],[13,14]])
outputs2 = np.dot([[0.1,0.2,0.3, 0.4]], [[3,4],[7,8],[11,12],[15,16]])
outputs = np.concatenate((outputs1, outputs2), axis=1)
```

```
[[ 9. 10.]]
[[11. 12.]]
[[ 9. 10. 11. 12.]]
```

```
outputs1 = np.dot([[0.1, 0.3]], [[1,2,3,4],[9,10,11,12]])
outputs2 = np.dot([[0.2, 0.4]], [[5,6,7,8],[13,14,15,16]])
outputs = outputs1 + outputs2
```

```
[[2.8 3.2 3.6 4. ]]
[[6.2 6.8 7.4 8. ]]
[[ 9. 10. 11. 12.]]
```

Can we parallelize Dot?

27

Can we parallelize Dot?

29

21: Hardware Acceleration V

Engr 315: Hardware / Software Codesign

Andrew Lukefahr
Indiana University