Lecture 4: Systolic Array Architectures

Puneet Gupta

Logistics

- One paper change in Week 3 presentations. Please take a look on Piazza. List is edited.
- Quick comment about roofline lecture:
  - Remember it is a log-log plot. AI is never “0”
- Colab tutorial later today

Roofline Example: GeMM on Nvidia A100

- GeMM kernel (nxn matrices)
  - Flops ~ $2n^3$
  - Memory traffic:
    - Assume large enough on-chip cache
    - Load A, B once. Store C once (FP16 inputs)
    - $3n^2$ memory accesses = $6n^2$ bytes
  - AI = n/3
- Nvidia A100
  - DRAM BW: 1935 GB/s
  - FP16 peak performance: 312TFLOPs
  - Ops/byte = 312/1.935 = 161
- n < 483 → memory bound
- n > 483 → compute bound

```
void gemm(size_t n, const float*A, const float *B,
const float *C) {
size_t i, j, k;
float sum;
for (i = 0; i < n; i++) {
  for (j = 0; j < n; j++) {
  sum = 0.0;
    for (k = 0; k  < n; k++) {
     sum = sum + A[i][k] * B[k][j]; }
    C[i][j] = sum; } } }
```

ML Accelerators Needs

- Improve arithmetic intensity
- Improve reuse from all memories (local and distant)
- Support fast multiply-add (MAC) operations
- (nice to have) support variable precision arithmetic
- Have large DRAM bandwidth
- Flexibility/programmability to choose right dataflow for right NN layers
- Goals:
  - Simple, regular design (keep # unique parts small and regular)
  - High parallelism → high performance
  - Balanced computation and I/O (memory) bandwidth

A ML Accelerator Example: Systolic Array

- Idea: Replace a single processing element (PE) with a regular array of PEs and carefully orchestrate flow of data between the PEs
  - such that they collectively transform a piece of input data before outputting it to memory
- Benefit: Maximizes computation done on a single piece of data element brought from memory
- Pipelining on steroids!
  - Array structure can be multi-dimensional
  - PE connections can be multidirectional
  - PEs can have local memory and execute kernels (rather than a piece of the instruction)

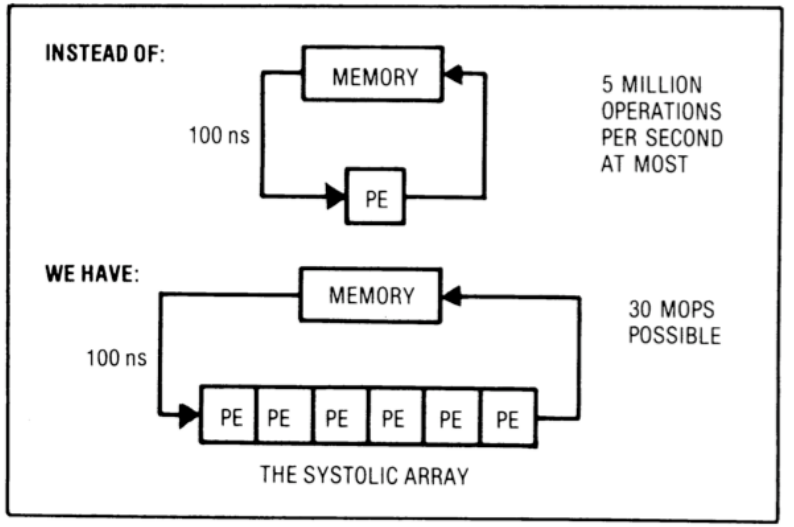


Figure 1. Basic principle of a systolic system.

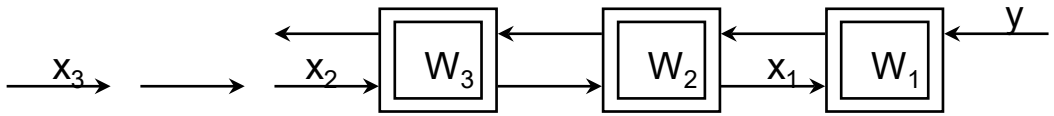
Memory: heart
Data: blood
PEs: cells

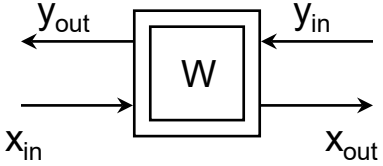
Memory pulses data through Pes
[Kung'82]

6X better arithmetic intensity!

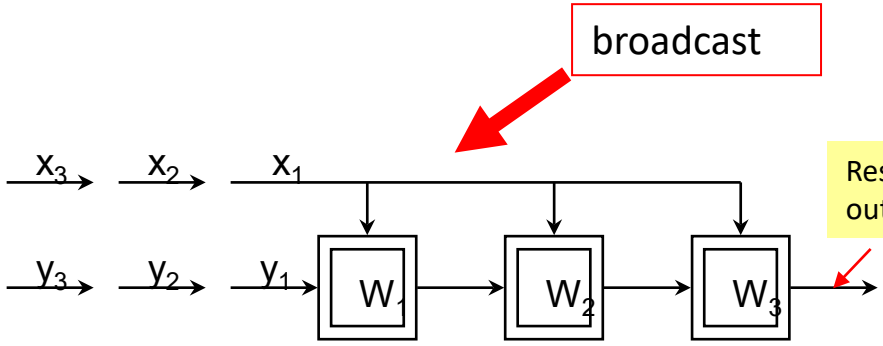
Dot Product in an Example Systolic Array

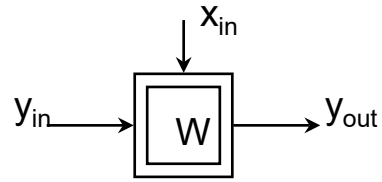
- Weight stationary
  - Option 1
    - Inputs and results move in the opposite direction
    - one-half the cells work
    - t=1: w1x1
    - t=2: w1x1+w2x2,
    - t=3: w1x1+w2x2+w3x3, w1x2
  - Option 2
    - All cells work usefully (in steady state)
    - Need a broadcast → likely more wiring overhead





$y_{out} = y_{in} + W \times x_{in}$
$x_{out} = x_{in}$

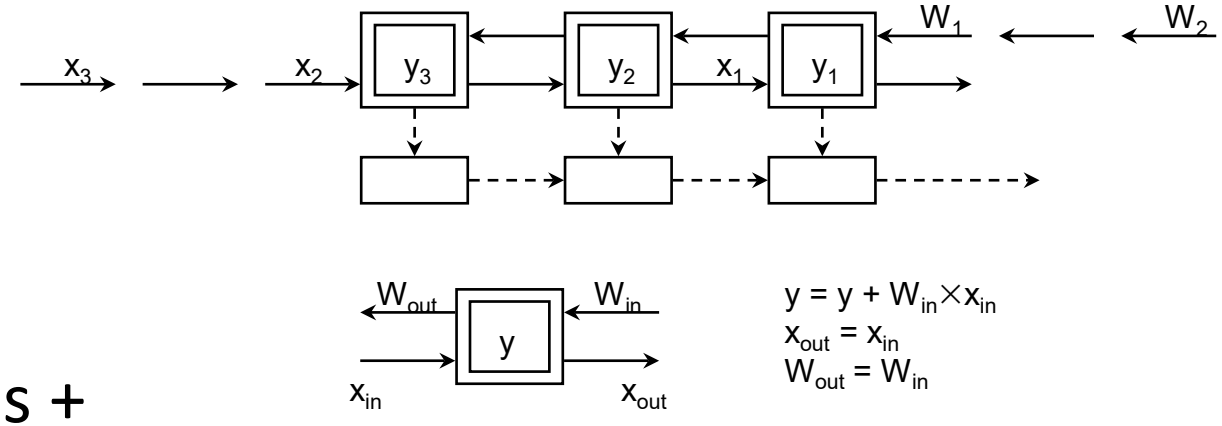




$y_{out} = y_{in} + W \times x_{in}$

Dot Product in an Example Systolic Array

- Output stationary
  - t=1: y1=w1x1
  - t=2: y2=w1x2
  - t=3: y1=w1x1+w2x2, y3=w1x3…
- Many other 1D systolic architectures + dataflows possible with different tradeoffs



Example 2D Systolic Array

$$\begin{bmatrix} c_{00} & c_{01} & c_{02} \\ c_{10} & c_{11} & c_{12} \\ c_{20} & c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix}$$

- Output stationary
- Each PE does a MAC and passes unchanged inputs right and down
- Results can be read out from each PE at the end
- Both ***A*** and ***B*** values are fully reused