Systolic Arrays & Their Applications

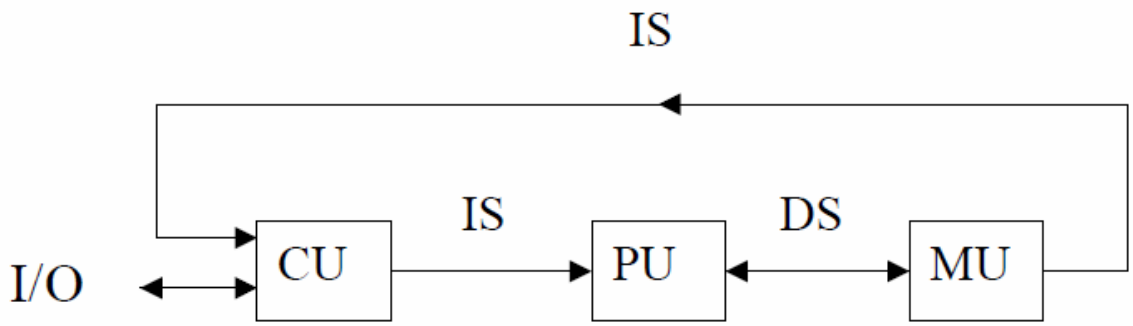
Dr. Shubhajit Roy Chowdhury,

Centre for VLSI and Embedded Systems Technology,

IIIT Hyderabad, India

Email: src.vlsi@iiit.ac.in

Flynn's Classification of Computer Architectures



(a) SISD Uniprocessor Architecture

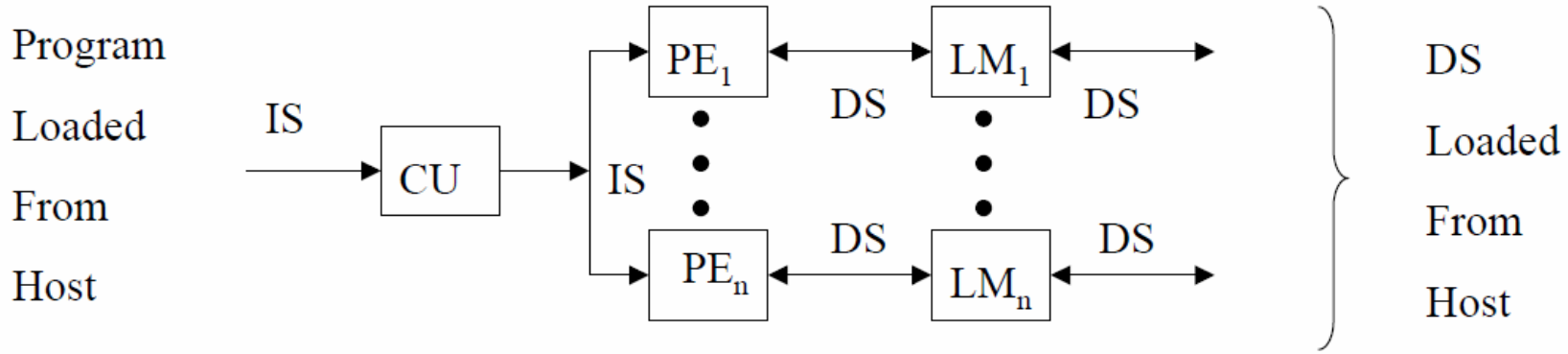
Captions:

CU - Control Unit ; PU – Processing Unit

MU – Memory Unit ; IS – Instruction Stream

DS – Date Stream

Flynn's Classification of Computer Architectures



(b) SIMD Architecture (with Distributed Memory)

Captions:

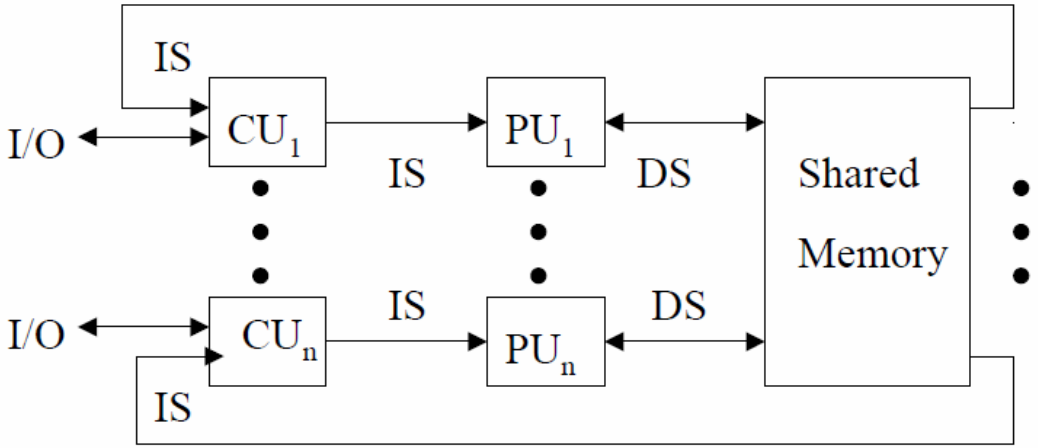
CU - Control Unit ; PU - Processing Unit

MU - Memory Unit ; IS - Instruction Stream

DS - Date Stream ; PE – Processing Element

LM – Local Memory

Flynn's Classification of Computer Architectures



(c) MIMD Architecture (with Shared Memory)

Captions:

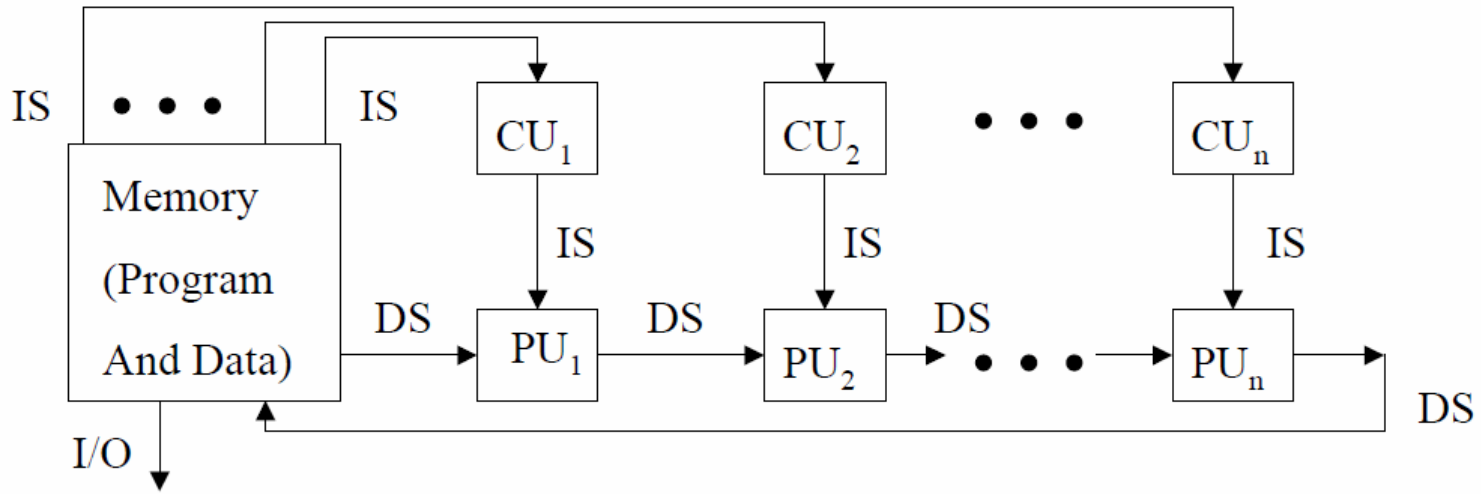
CU - Control Unit ; PU - Processing Unit

MU - Memory Unit ; IS - Instruction Stream

DS - Date Stream ; PE – Processing Element

LM – Local Memory

Flynn's Classification of Computer Architectures



(d) MISD Architecture (the Systolic Array)

Captions:

CU - Control Unit ; PU - Processing Unit

MU - Memory Unit ; IS - Instruction Stream

DS - Date Stream ; PE – Processing Element

LM – Local Memory

What Is a Systolic Array?

A **systolic array** is an arrangement of processors in an array where data flows synchronously across the array between neighbors, usually with different data flowing in different directions

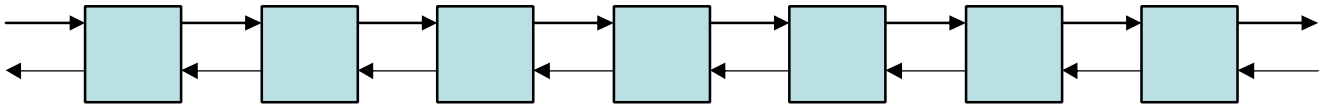
Each processor at each step takes in data from one or more neighbors (e.g. North and West), processes it and, in the next step, outputs results in the opposite direction (South and East).

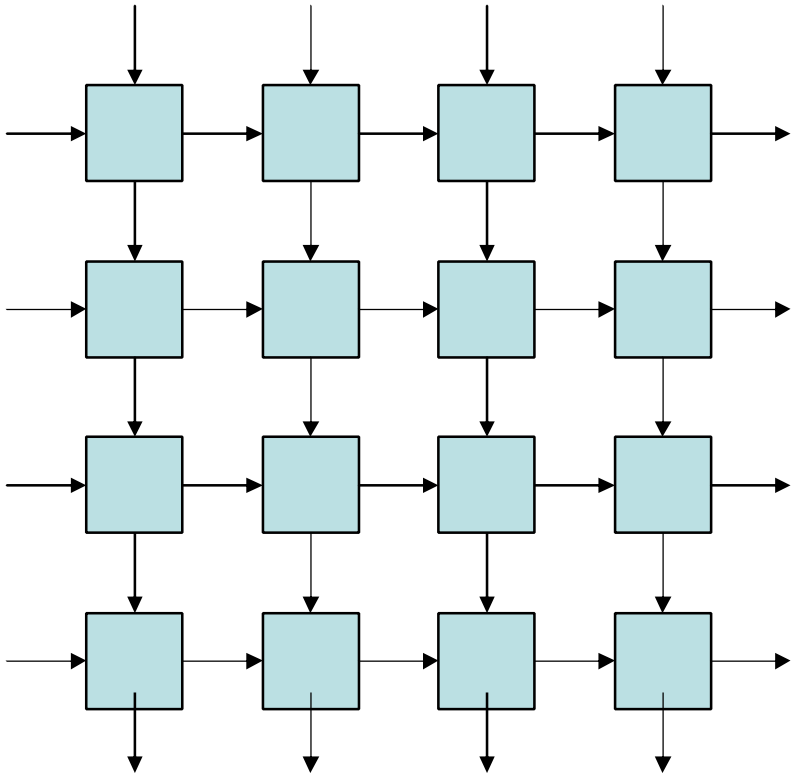
H. T. Kung and Charles Leiserson were the first to publish a paper on systolic arrays in 1978, and coined the name.

What Is a Systolic Array?

- A specialized form of parallel computing.
- Multiple processors connected by short wires.
- Unlike many forms of parallelism which lose speed through their connection.
- Cells(processors), compute data and store it independently of each other.
  - Each unit is an independent processor.

Some simple examples of systolic array models.





Matrix Multiplication

$$\begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix} \quad * \quad \begin{matrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{matrix} \quad = \quad \begin{matrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{matrix}$$

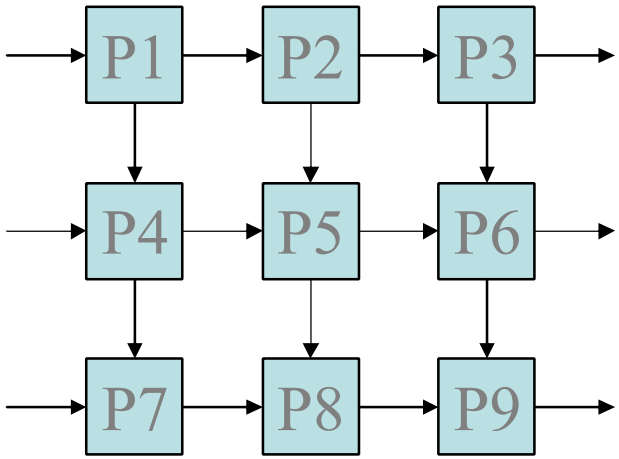
Conventional Method: N^3

```
For I = 1 to N
    For J = 1 to N
        For K = 1 to N
            C[I,J] = C[I,J] + A[J,K] * B[K,J];
```

Systolic Method

This will run in $O(n^3)$ time!

To run in N time we need N x N processing units, in this case we need 9.



We need to modify the input data, like so:

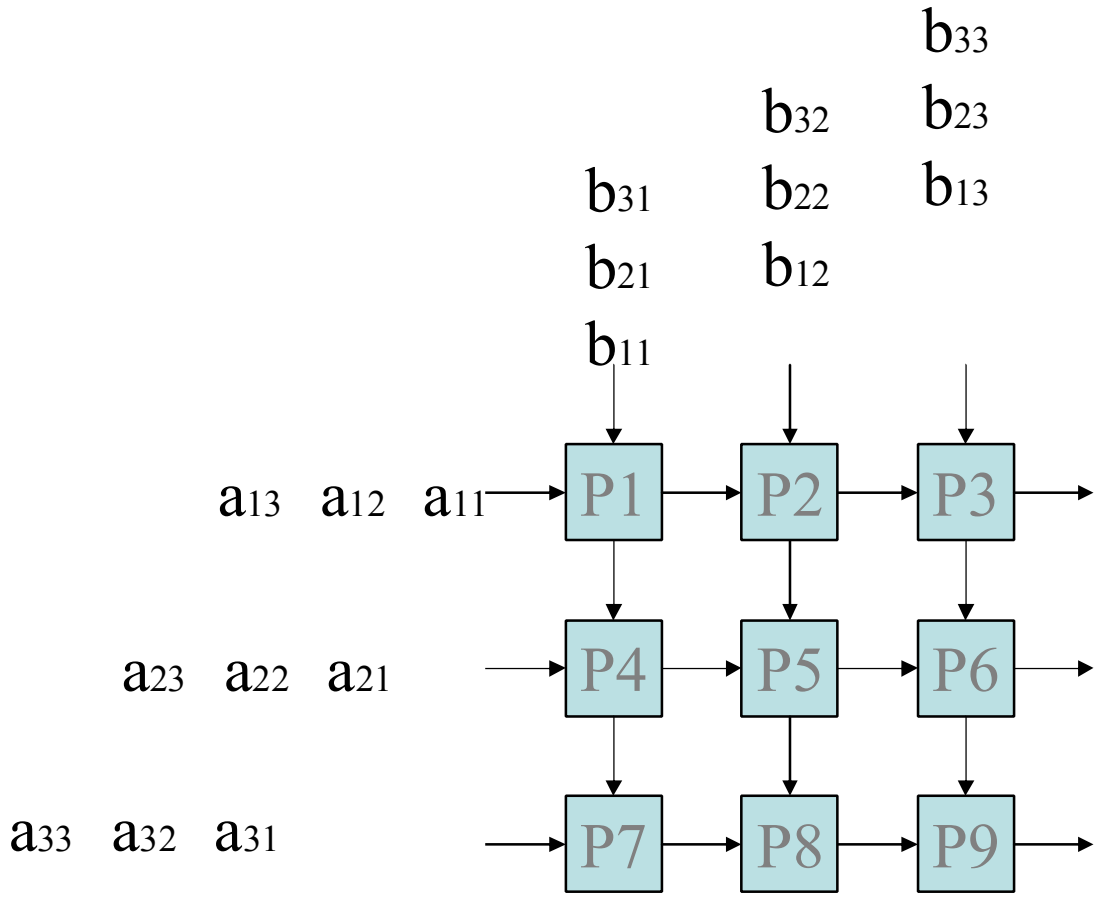
Flip columns 1 & 3 $\longrightarrow$

$$\begin{matrix} a_{13} & a_{12} & a_{11} \\ a_{23} & a_{22} & a_{21} \\ a_{33} & a_{32} & a_{31} \end{matrix}$$

Flip rows 1 & 3 $\longrightarrow$

$$\begin{matrix} b_{31} & b_{32} & b_{33} \\ b_{21} & b_{22} & b_{23} \\ b_{11} & b_{12} & b_{13} \end{matrix}$$

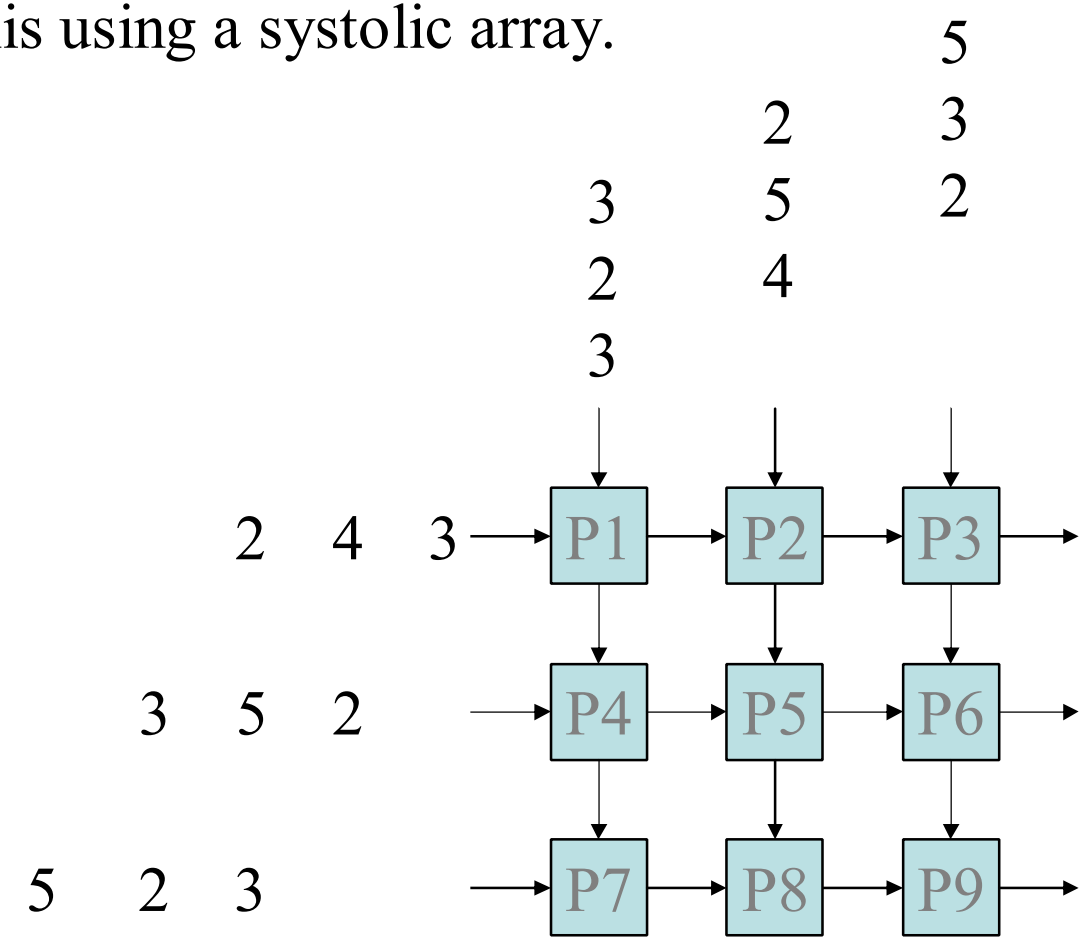
and finally stagger the data sets for input.



At every tick of the global system clock data is passed to each processor from two different directions, then it is multiplied and the result is saved in a register.

$$\begin{matrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{matrix} \quad * \quad \begin{matrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{matrix} \quad = \quad \begin{matrix} 23 & 36 & 28 \\ 25 & 39 & 34 \\ 28 & 32 & 37 \end{matrix}$$

Lets try this using a systolic array.



Systolic Array for Convolution

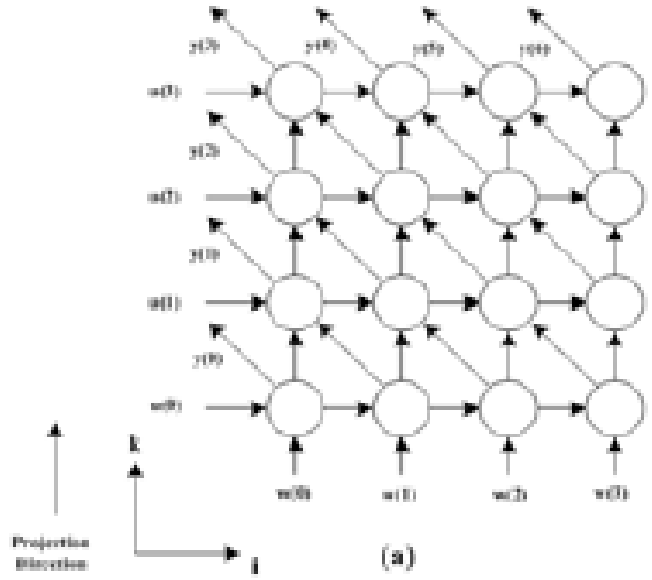
The problem of convolution is defined as follows:

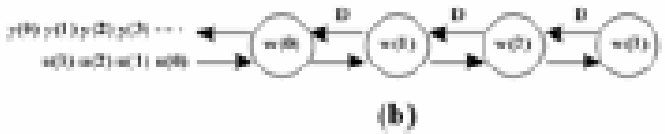
Given two sequences u(i) and w(i), i=0,1,2,….,N-1,

The convolution of the two sequences is:

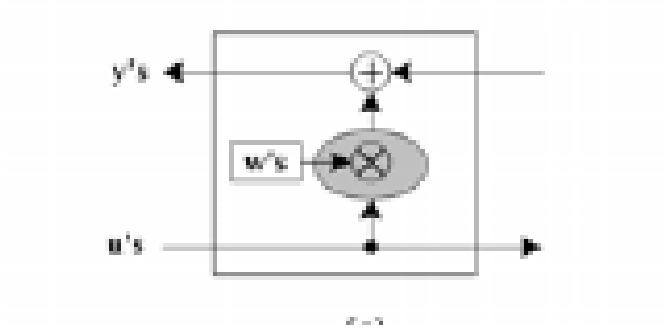
$$y(i) = \sum_{k=0}^{N-1} u(k)w(i-k)$$

Dependence graph for convolution





Systolic Array Cell



General Observations about Systolic Array

- May be visualized as parallel processing architecture
- Same data flowing through multiple computational elements
- Suitable for real time signal processing

Thank You