

# Systolic Architecture (Systolic Array)

<https://www.cs.auckland.ac.nz/~jmor159/363/html/systolic.html>

# Systolic arrays

- arrays of processors
  - which are connected to a small number of nearest neighbours in a mesh-like topology
  - Processors perform a sequence of operations on data that flows between them
    - operations will be the same in each processor, with each processor performing an operation (or small number of operations) on a data item
    - then passing it on to its neighbour. Like MISD machines, systolic arrays compute in "lock-step" with each processor undertaking alternate compute | communicate phases. Actually it is Single Function, Multiple Data, Merged Result(s).

# Matrix Multiplication Example

Provided by: [bob63](#)

Learn more at: <http://www.cs.ucf.edu>

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

Conventional Method:  $N^3$

For I = 1 to N

For J = 1 to N

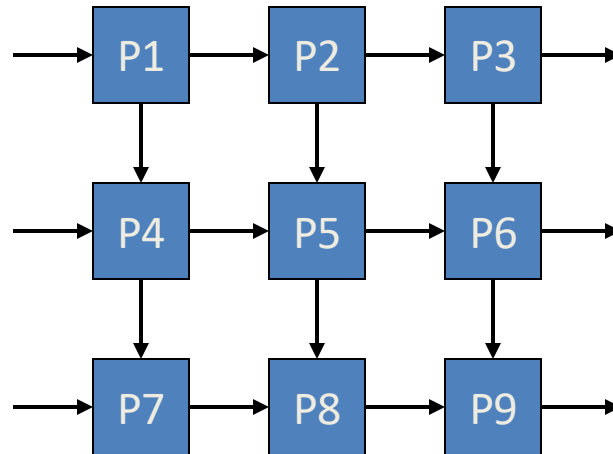
For K = 1 to N

$C[I,J] = C[I,J] + A[J,K] * B[K,J];$


# Systolic Method


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

To run in  $N$  time we need  $N \times N$  processing units, in this case we need 9.

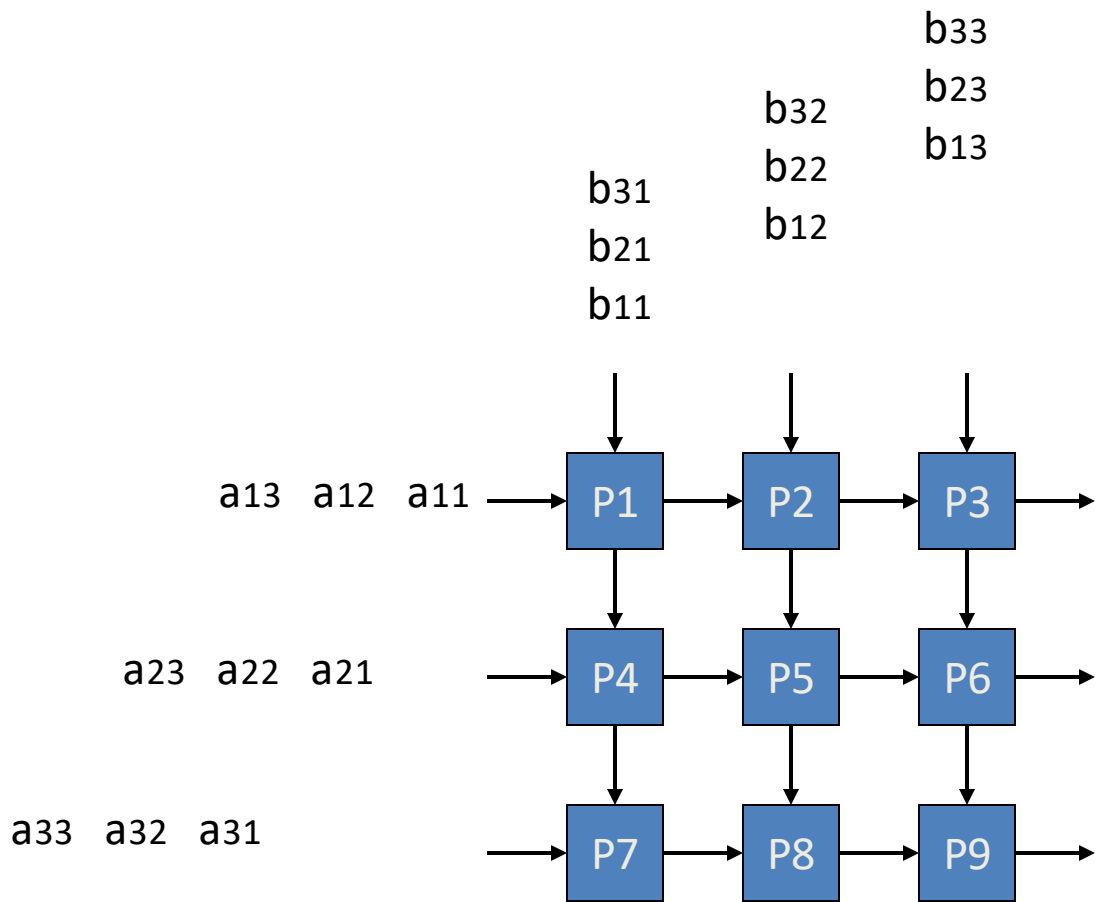


We need to modify the input data, like so:

Flip columns 1 & 3             $\begin{matrix} a_{13} & a_{12} & a_{11} \\ a_{23} & a_{22} & a_{21} \\ a_{33} & a_{32} & a_{31} \end{matrix}$

Flip rows 1 & 3             $\begin{matrix} b_{31} & b_{32} & b_{33} \\ b_{21} & b_{22} & b_{23} \\ b_{11} & b_{12} & b_{13} \end{matrix}$

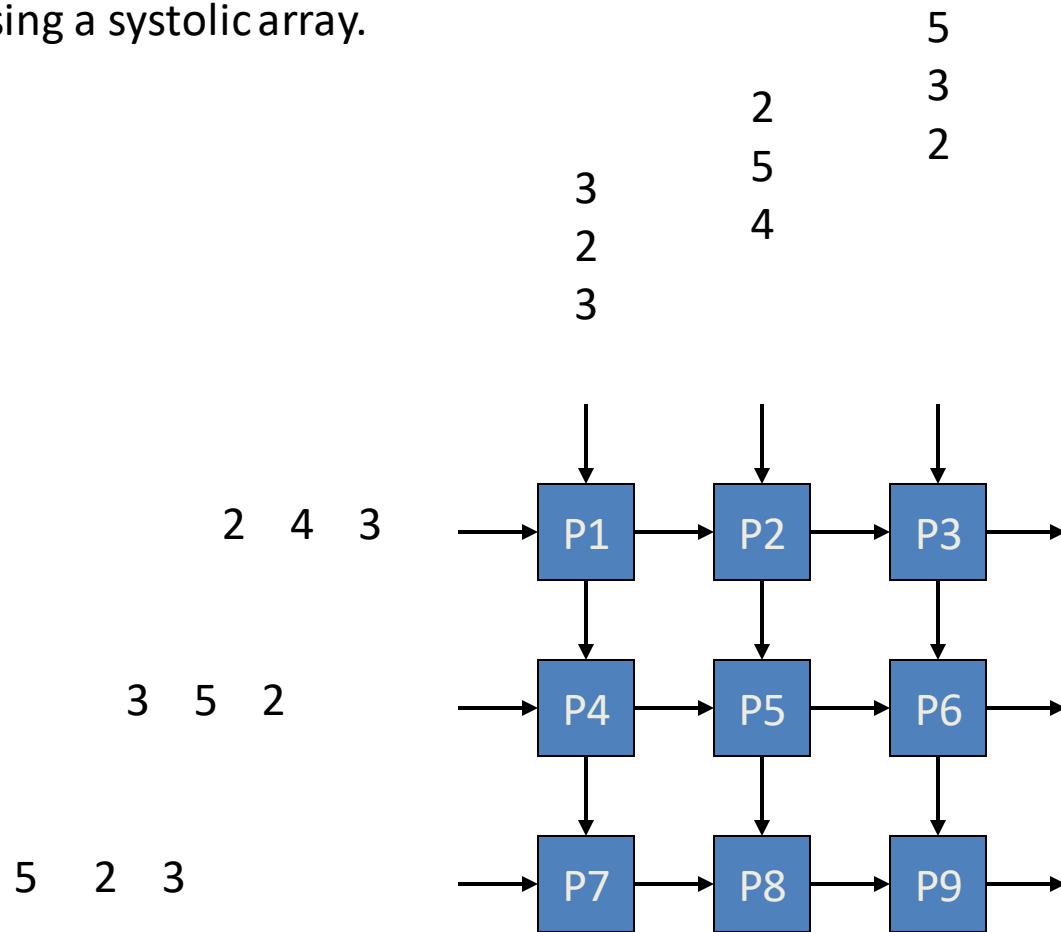
and finally stagger the data sets for input.



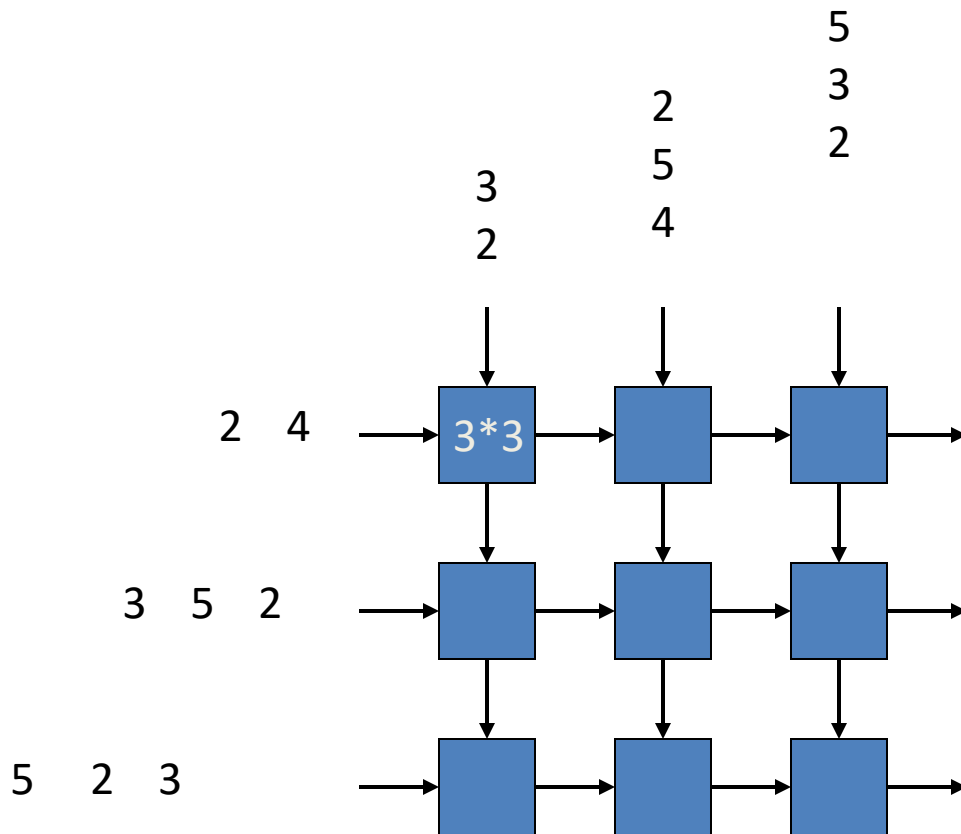
At every tick of the global system clock data is passed to each processor from two different directions, then it is multiplied and the result is saved in a register.

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

Lets try this using a systolicarray.



Clock tick: 1

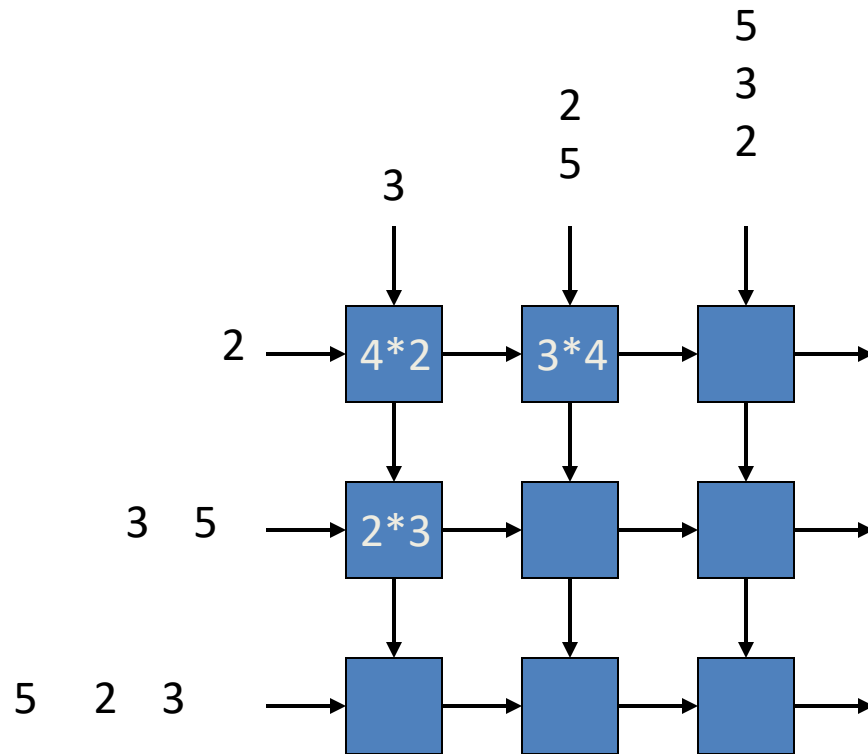


P1                  P2                  P3                  P4                  P5                  P6                  P7                  P8                  P9

9	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---



Clock tick: 2



P1

P2

P3

P4

P5

P6

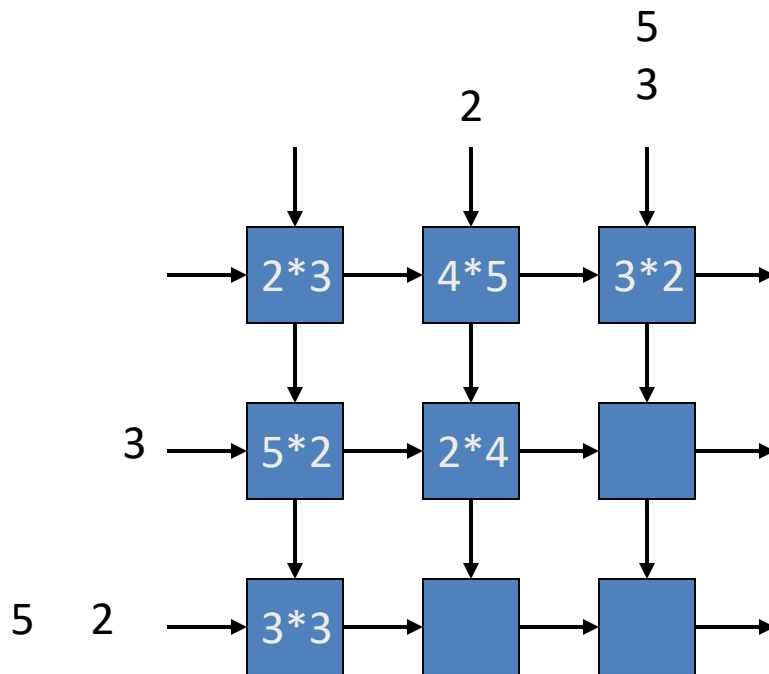
P7

P8

P9

17	12	0	6	0	0	0	0	0
----	----	---	---	---	---	---	---	---

Clock tick: 3



P1

P2

P3

P4

P5

P6

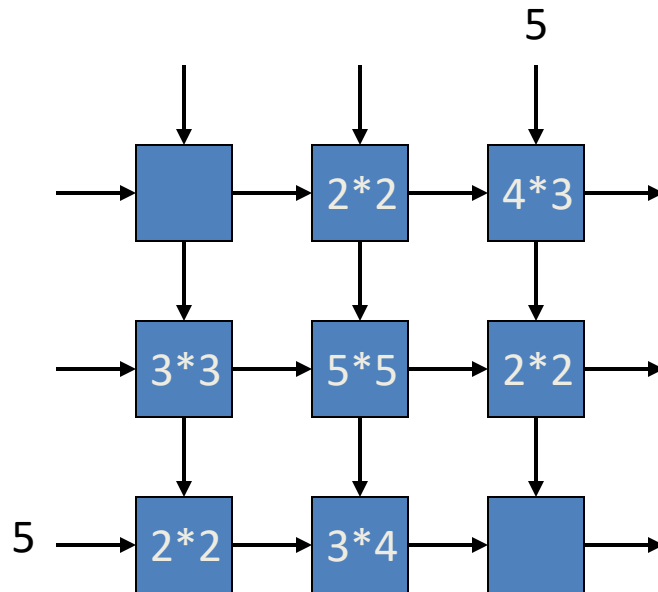
P7

P8

P9

23	32	6	16	8	0	9	0	0
----	----	---	----	---	---	---	---	---

Clock tick: 4



P1

P2

P3

P4

P5

P6

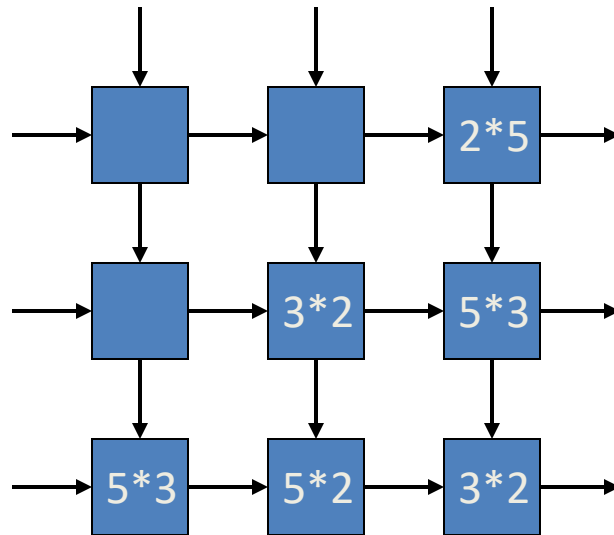
P7

P8

P9

23	36	18	25	33	4	13	12	0
----	----	----	----	----	---	----	----	---

Clock tick: 5



P1

P2

P3

P4

P5

P6

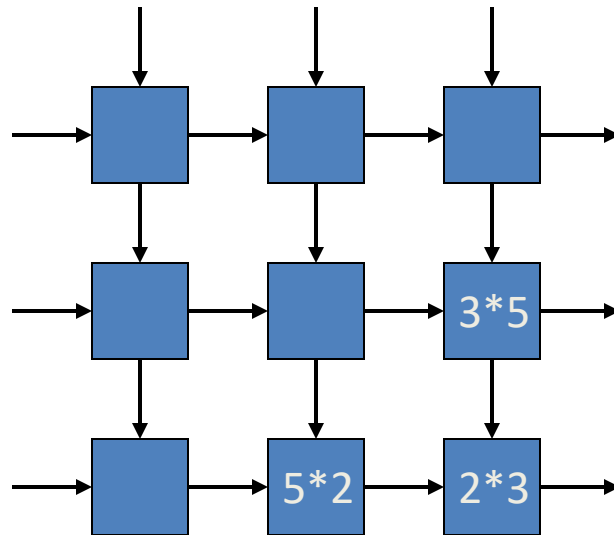
P7

P8

P9

23	36	28	25	39	19	28	22	6
----	----	----	----	----	----	----	----	---

Clock tick: 6



P1

P2

P3

P4

P5

P6

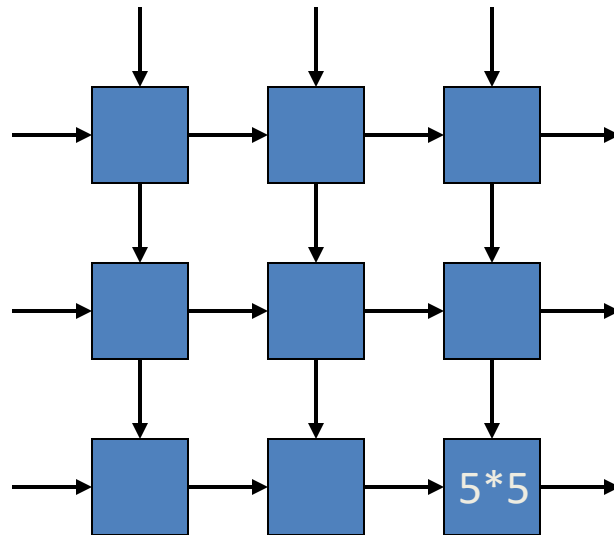
P7

P8

P9

23	36	28	25	39	34	28	32	12
----	----	----	----	----	----	----	----	----

Clock tick: 7



P1

P2

P3

P4

P5

P6

P7

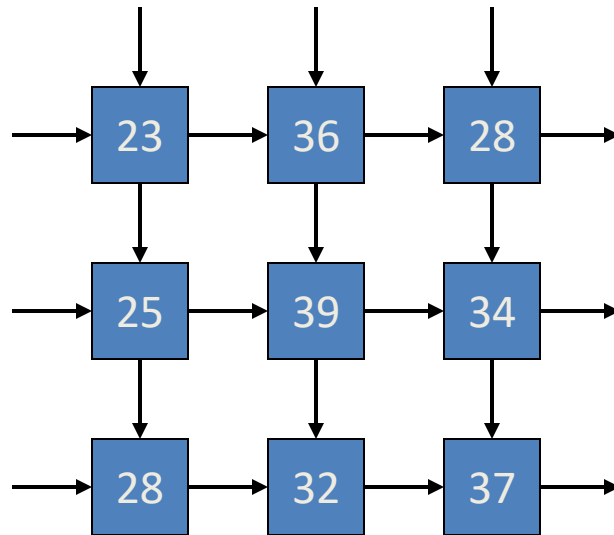
P8

P9

23	36	28	25	39	34	28	32	37
----	----	----	----	----	----	----	----	----

Same answer! In  $2n + 1$  time, can we do better?

The answer is yes, there is an optimization.



P1

P2

P3

P4

P5

P6

P7

P8

P9

23	36	28	25	39	34	28	32	37
----	----	----	----	----	----	----	----	----

# Cannon's Technique

- No processor is idle.
- Instead of feeding the data, it is cycled.
- Data is staggered, but slightly different than before.
- Not including the loading which can be done in parallel for a time of 1, this technique is effectively  $N$ .



# Other Applications

- Signal processing
- Image processing
- Solutions of differential equations
- Graph algorithms
- Biological sequence comparison
- Other computationally intensive tasks

# Samba: Systolic Accelerator for Molecular Biological Applications

This systolic array contains 128 processors shared into 32 full custom VLSI chips. One chip houses 4 processors, and one processor performs 10 millions matrix cells per second.



# Why Systolic?

- Extremely fast.
- Easily scalable architecture.
- Can do many tasks single processor machines cannot attain.
- Turns some exponential problems into linear or polynomial time.

# Why Not Systolic?

- Expensive.
- Not needed on most applications, they are a highly specialized processor type.
- Difficult to implement and build.

# Summary

- What a systolic array is.
- Step through of matrix multiplication.
- Cannon's optimization.
- Other applications including samba.
- Positives and negatives of systolic arrays.

# References

- “Systolic Algorithms & Architectures” by Patrice Quinton and Yves Robert, 1991 Prentice Hall International
- “The New Turing Omnibus” by A.K. Dewdney, New York
- <http://www.irisa.fr/symbiose/people/lavenier/Samba/>
- <http://www.cs.hmc.edu/courses/mostRecent/cs156/html08/slides08.pdf>
- <http://www.ntu.edu.sg/home/ecrwan/Phdthesi.htm>







[https://en.wikipedia.org/wiki/Systolic\\_array](https://en.wikipedia.org/wiki/Systolic_array)

- **systolic array** is a homogeneous [network](#) of tightly coupled [data processing units](#) (DPUs) called cells or [nodes](#). Each node or DPU independently computes a partial result as a function of the data received from its upstream neighbors, stores the result within itself and passes it downstream. Systolic arrays were invented by [Richard P. Brent](#) and [H. T. Kung](#), who developed them to compute [greatest common divisors](#) of integers and polynomials.<sup>[1][\[not in citation given\]](#)</sup> They are sometimes classified as multiple-instruction single-data ([MISD](#)) architectures under [Flynn's taxonomy](#), but this classification is questionable because a strong argument can be made to distinguish systolic arrays from any of Flynn's four categories: [SISD](#), [SIMD](#), [MISD](#), [MIMD](#), as discussed later in this article. Actually it is Single Function, Multiple Data, Merged Result(s).
- The parallel input [data](#) flows through a network of hard-wired [processor](#) nodes, which combine, process, [merge](#) or [sort](#) the input data into a derived result. Because the [wave](#)-like propagation of data through a systolic array resembles the [pulse](#) of the human circulatory system, the name *systolic* was coined from medical terminology. The name is derived from [systole](#) as an analogy to the regular pumping of blood by the heart.
- Applications<sup>[\[edit\]](#)</sup>
- Systolic arrays are often hard-wired for specific operations, such as "multiply and accumulate", to perform massively [parallel](#) integration, [convolution](#), [correlation](#), [matrix multiplication](#) or data sorting tasks. They are also used for [dynamic programming](#) algorithms, used in DNA and protein [sequence analysis](#).
- systolic array are triggered by the arrival of new data and always process the data in exactly the same way. The actual processing within each node may be hard wired or block [microcoded](#), in which case the common node personality can be block programmable.
- The systolic array paradigm with data-streams driven by data [counters](#), is the counterpart of the Von Neumann architecture with instruction-stream driven by a program counter. Because a systolic array usually sends and receives multiple data streams, and multiple data counters are needed to generate these data streams, it supports [data parallelism](#).