Systolic Arrays & Their Applications

By: Jonathan Break

Overview

- What it is
- N-body problem
- Matrix multiplication
- Cannon’s method
- Other applications
- Conclusion

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.

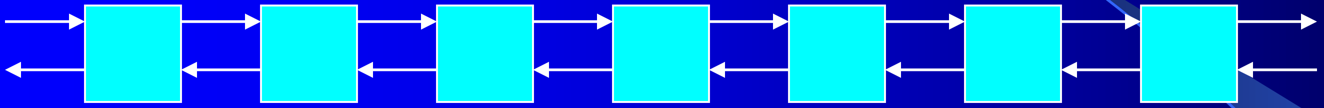
What Is a Systolic Array?

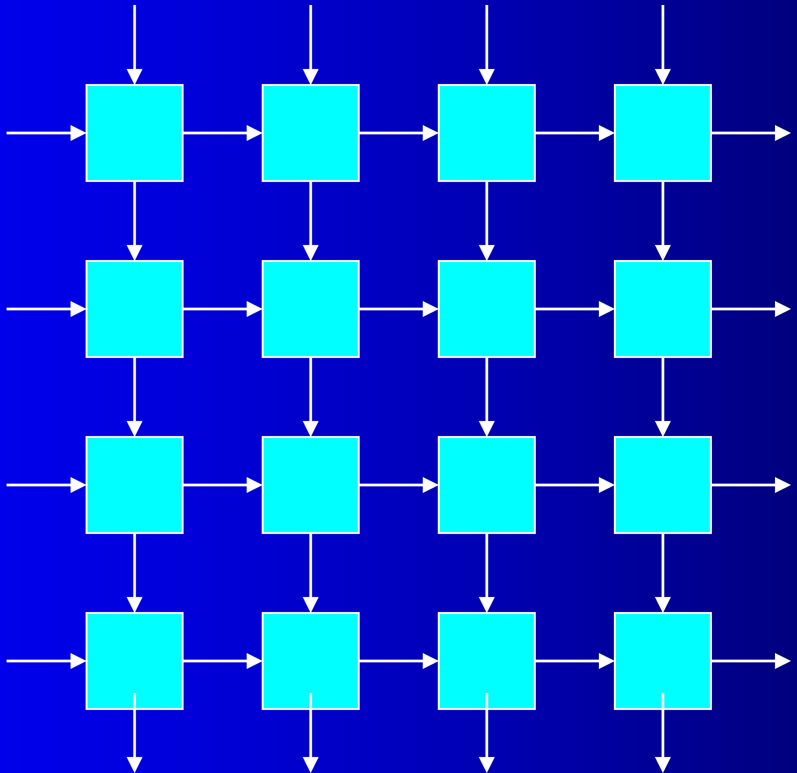
- A specialized form of parallel computing.
- Multiple processors connected by short wires.
- Unlike many forms of parallelism which lose speed through their connection.
- Cells(processors), compute data and store it independently of each other.

Systolic Unit(cell)

- Each unit is an independent processor.
- Every processor has some registers and an ALU.
- The cells share information with their neighbors, after performing the needed operations on the data.

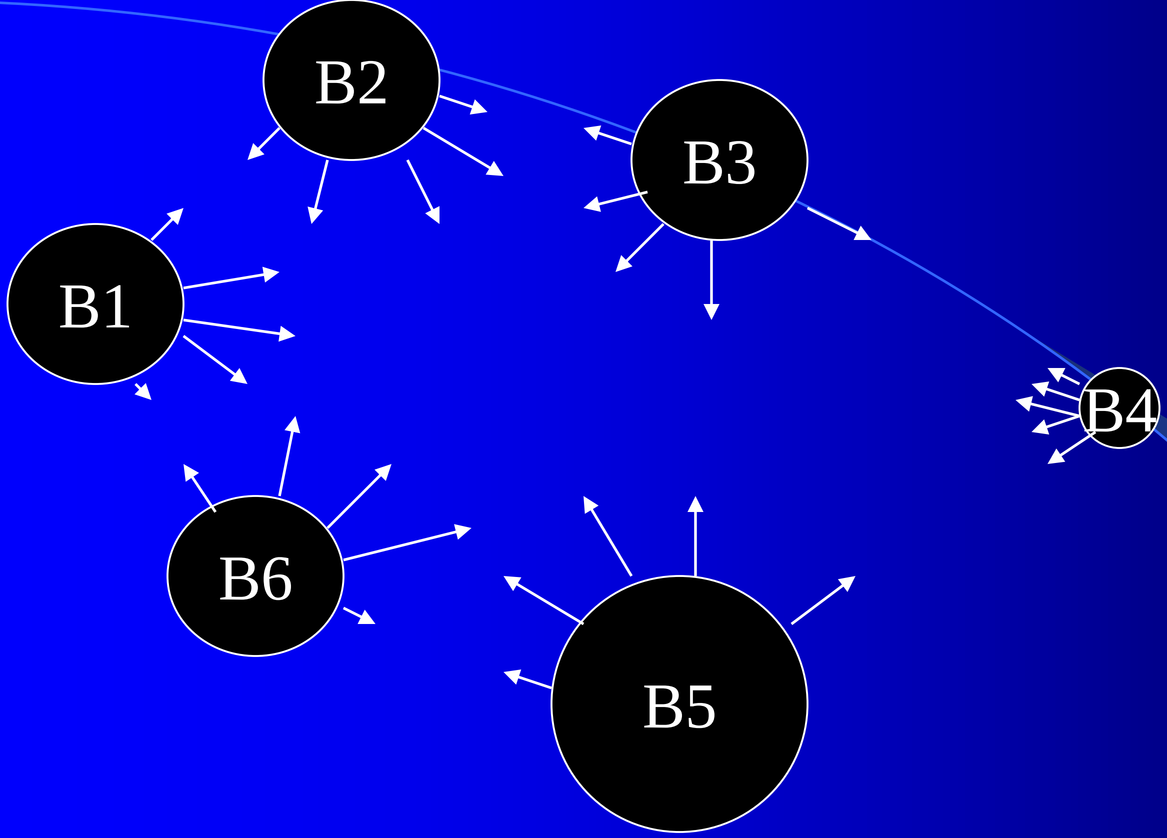
Some simple examples of systolic array models.



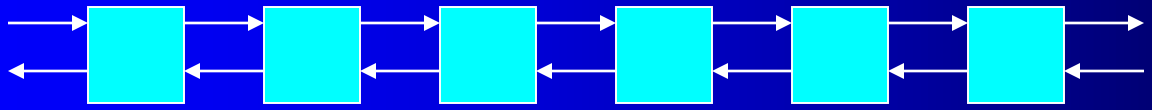


N-body Problem

- Conventional method: N^2
- Systolic method: N
- We will need one processing element for each body, lets call them planets.

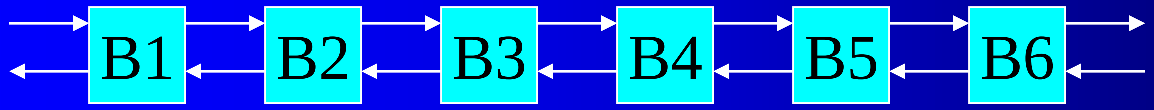


Six planets in orbit with different masses, and different forces acting on each other, so are systolic array will look like this.

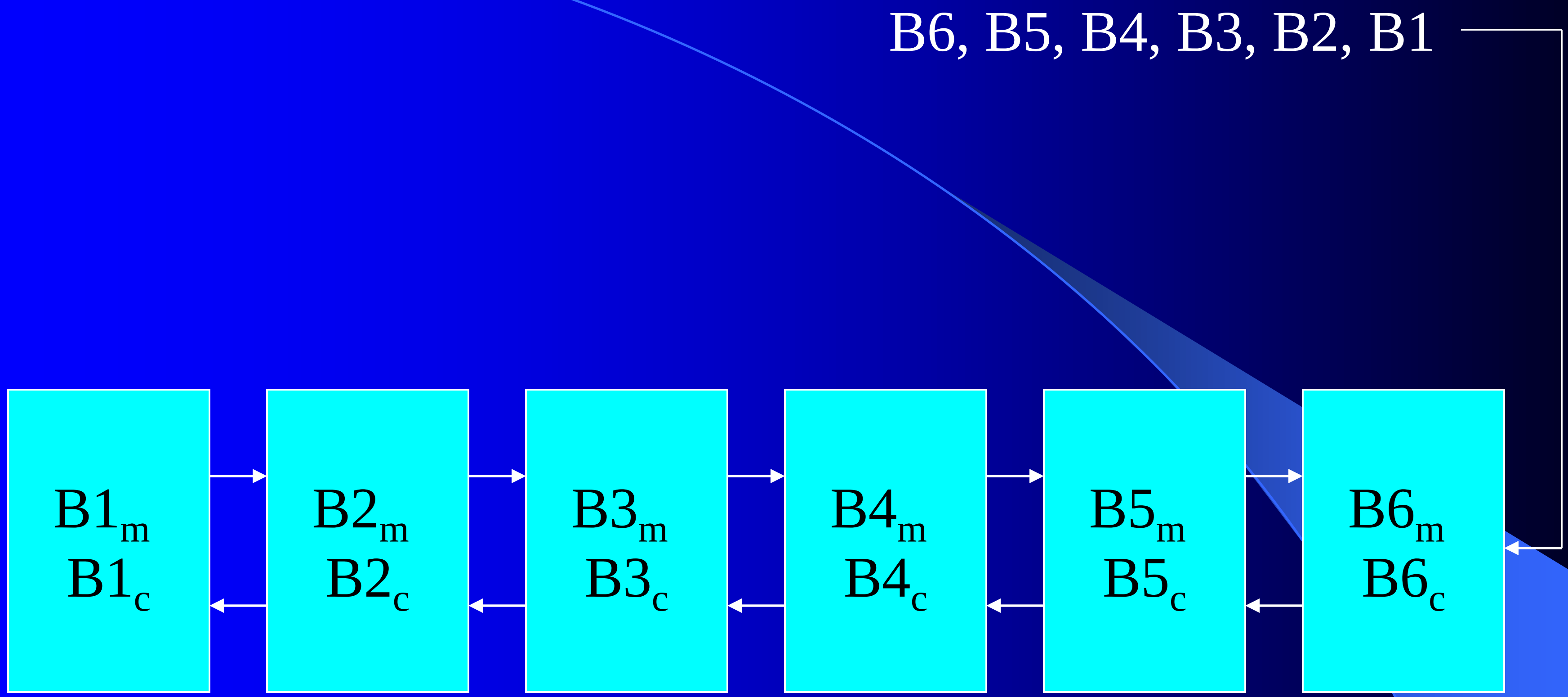


Each processor will hold the newtonian force formula: $F_{ij} = km_i m_j/d^2_{ij}$, k being the gravitational constant.

Then load the array with values.



Now that the array has the mass and the coordinates of Each body we can begin are computation.



B6, B5, B4, B3, B2, B1

$B1_m$ $B1_c$

$B2_m$ $B2_c$

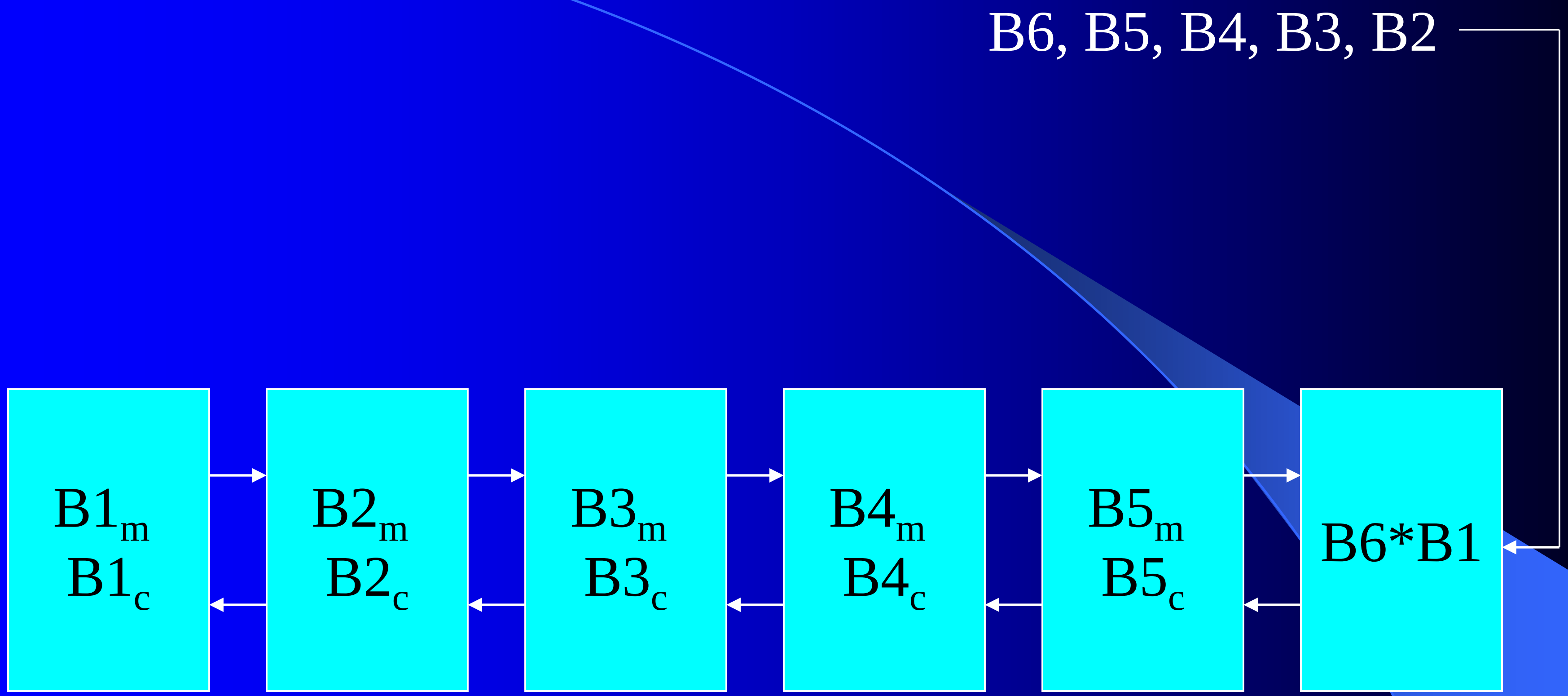
$B3_m$ $B3_c$

$B4_m$ $B4_c$

$B5_m$ $B5_c$

$B6_m$ $B6_c$

$$F_{ij} = k m_i m_j / d^2_{ij}$$



B6, B5, B4, B3, B2

$B1_m$ $B1_c$

$B2_m$ $B2_c$

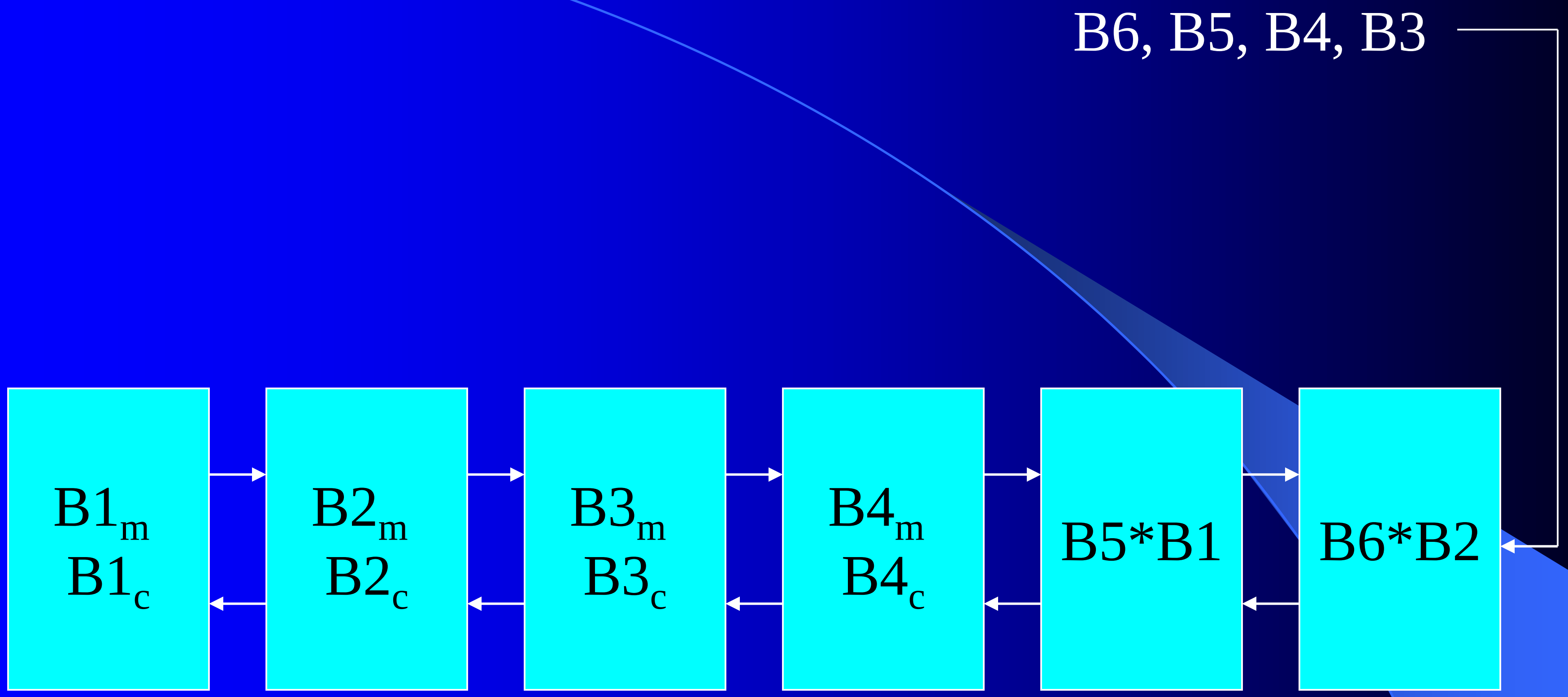
$B3_m$ $B3_c$

$B4_m$ $B4_c$

$B5_m$ $B5_c$

B6*B1

$$F_{ij} = km_i m_j/d^2_{ij}$$



B6, B5, B4, B3

$B1_m$ $B1_c$

$B2_m$ $B2_c$

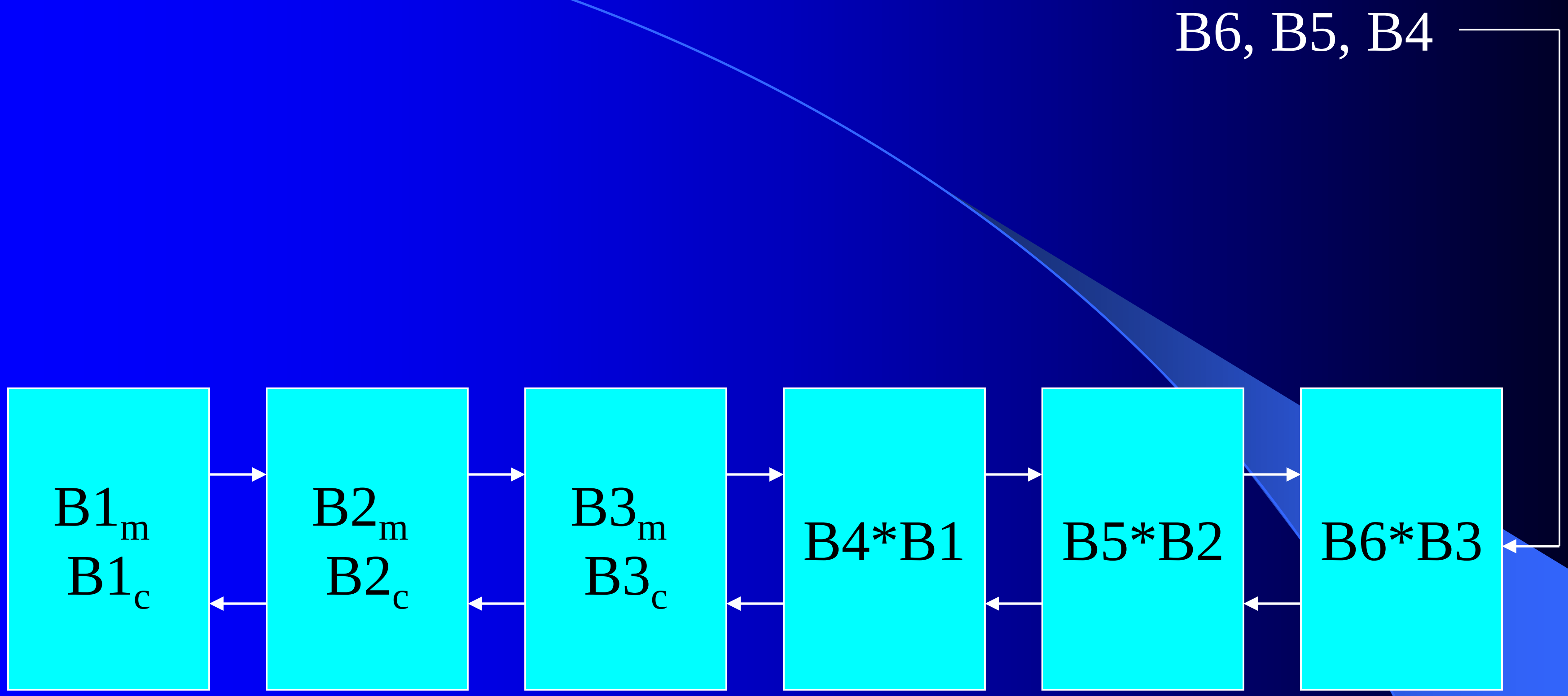
$B3_m$ $B3_c$

$B4_m$ $B4_c$

B5*B1

B6*B2

$$F_{ij} = k m_i m_j / d^2_{ij}$$



B6, B5, B4

$B1_m$ $B1_c$

$B2_m$ $B2_c$

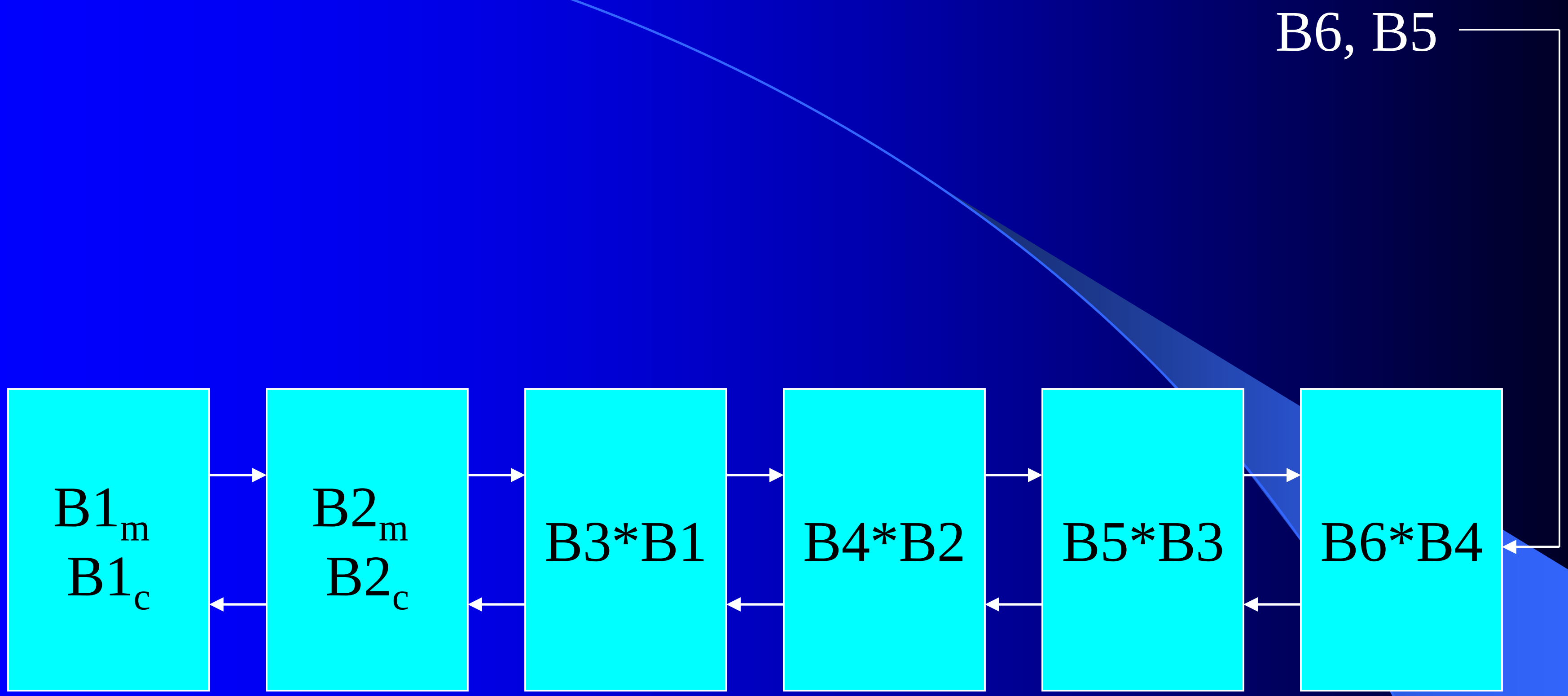
$B3_m$ $B3_c$

B4*B1

B5*B2

B6*B3

$$F_{ij} = km_i m_j / d^2_{ij}$$



B6, B5

$B1_m$ $B1_c$

$B2_m$ $B2_c$

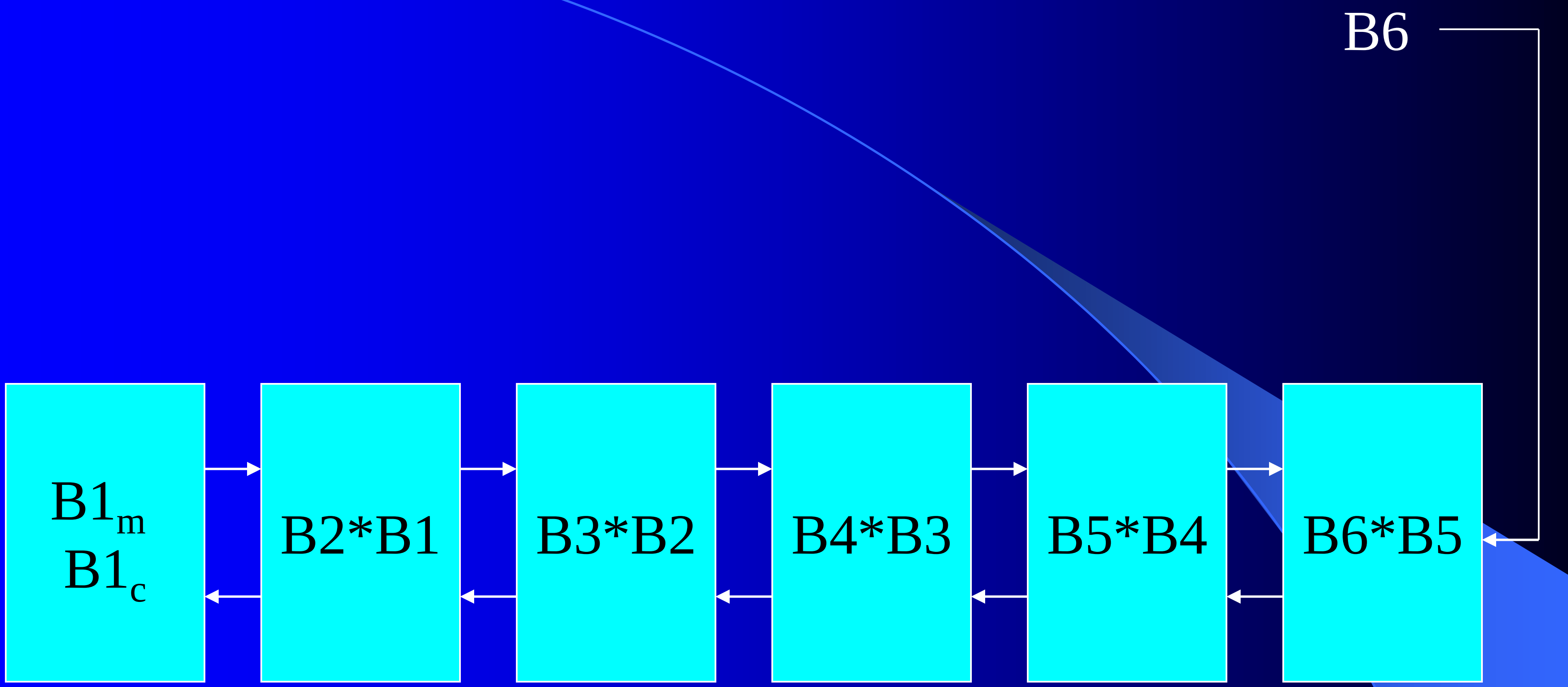
B3*B1

B4*B2

B5*B3

B6*B4

$$F_{ij} = km_i m_j/d^2_{ij}$$



B6

$B1_m$
$B1_c$

B2*B1

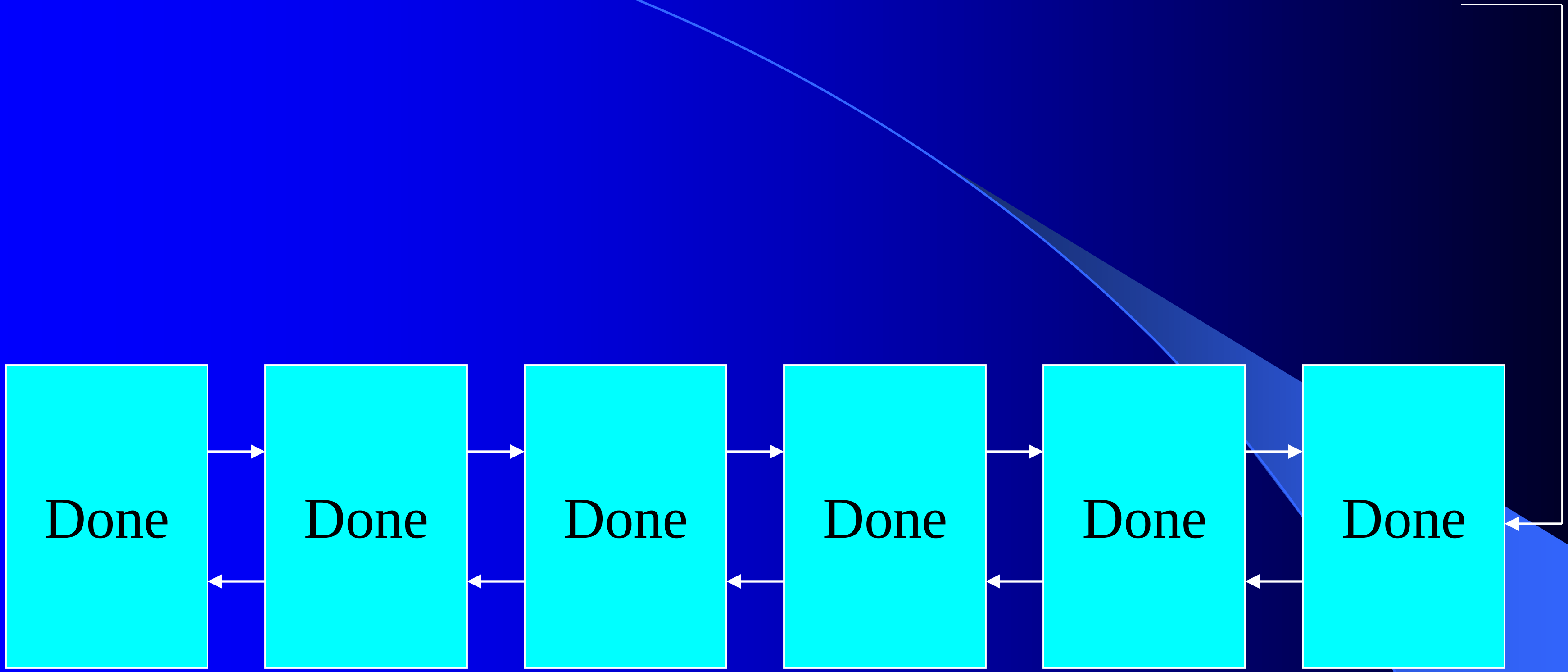
B3*B2

B4*B3

B5*B4

B6*B5

$$F_{ij} = km_im_j/d^2_{ij}$$



Done

Done

Done

Done

Done

Done

$$F_{ij} = km_i m_j / d^2_{ij}$$

Matrix Multiplication

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

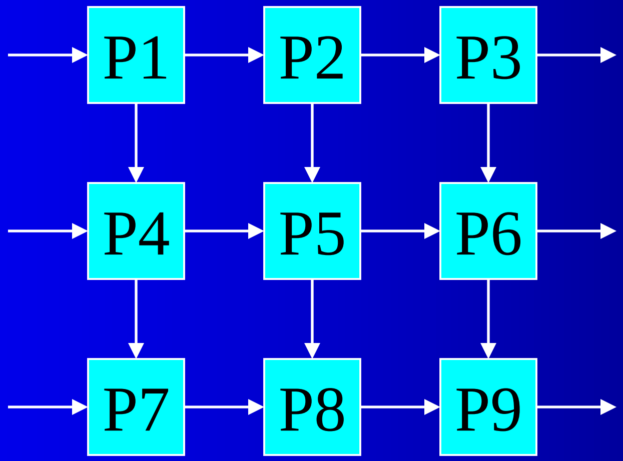
Conventional Method: N^3

```
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) 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:

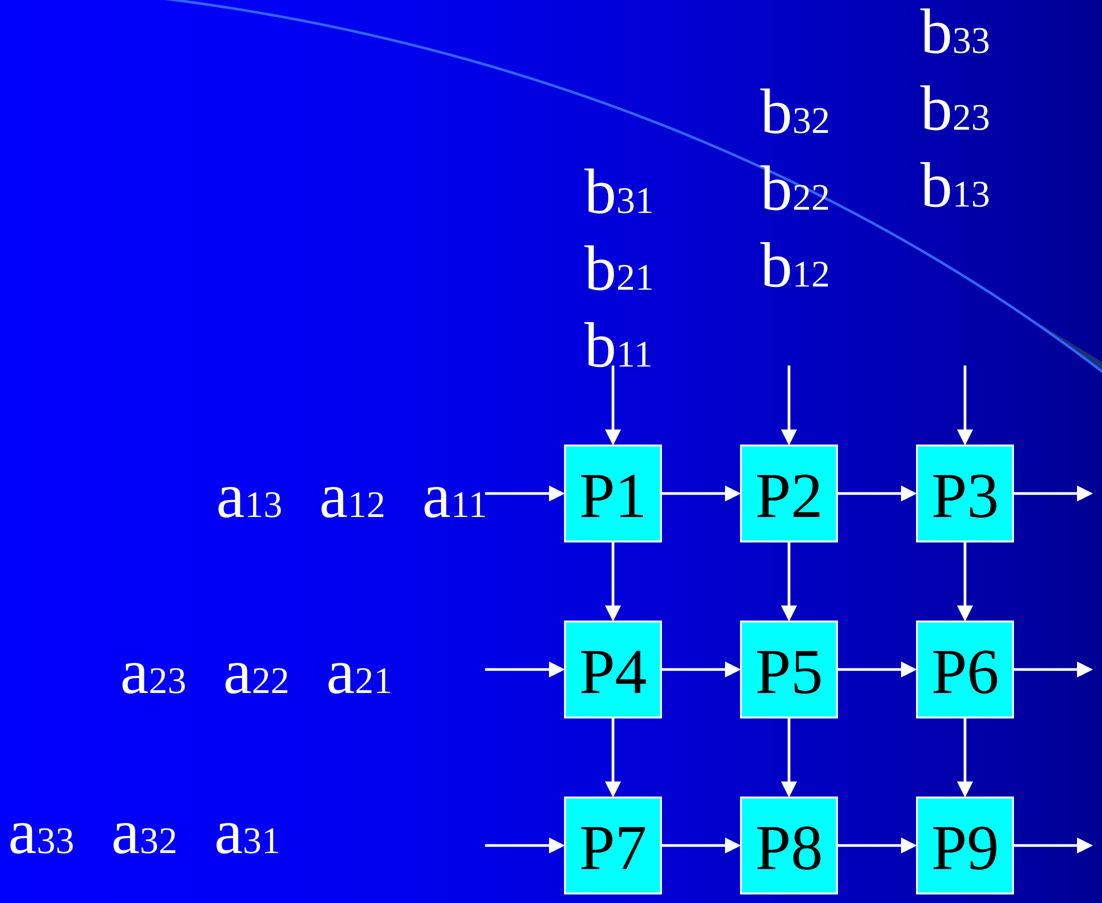
Flip columns 1 & 3 →

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

Flip rows 1 & 3 →

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

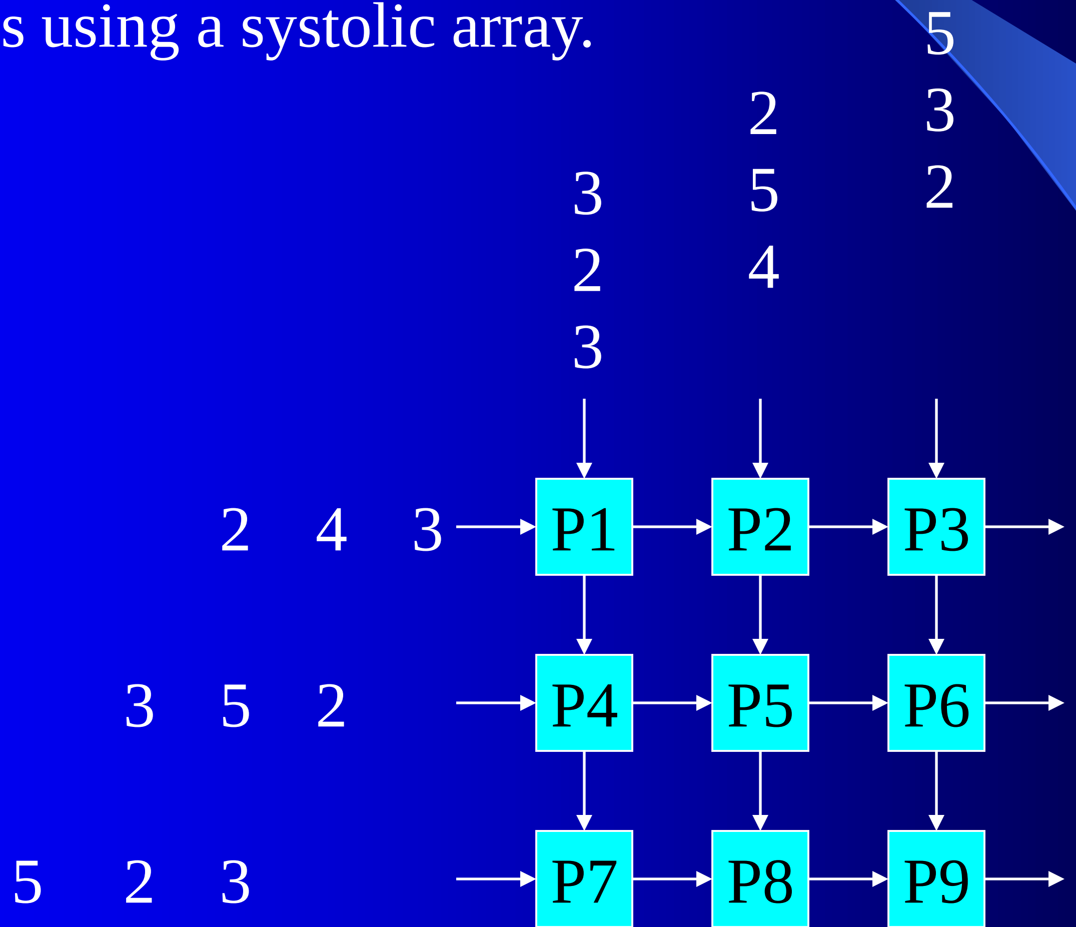
and finally stagger the data sets for input.


b_{33}
b_{32} b_{23}
b_{31} b_{22} b_{13}
b_{21} b_{12}
b_{11}
a_{13} a_{12} a_{11}
P1
P2
P3
a_{23} a_{22} a_{21}
P4
P5
P6
a_{33} a_{32} a_{31}
P7
P8
P9

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.



5
3
2

2
5
4

3
2
3

2 4 3

3 5 2

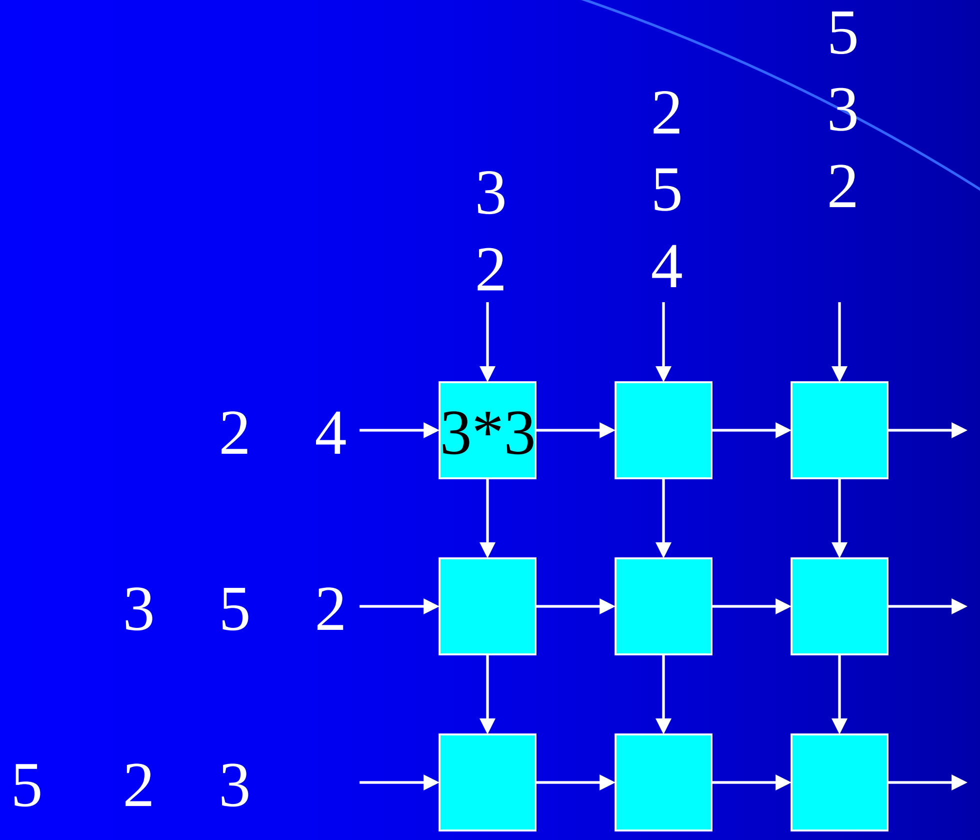
5 2 3

P1 P2 P3

P4 P5 P6

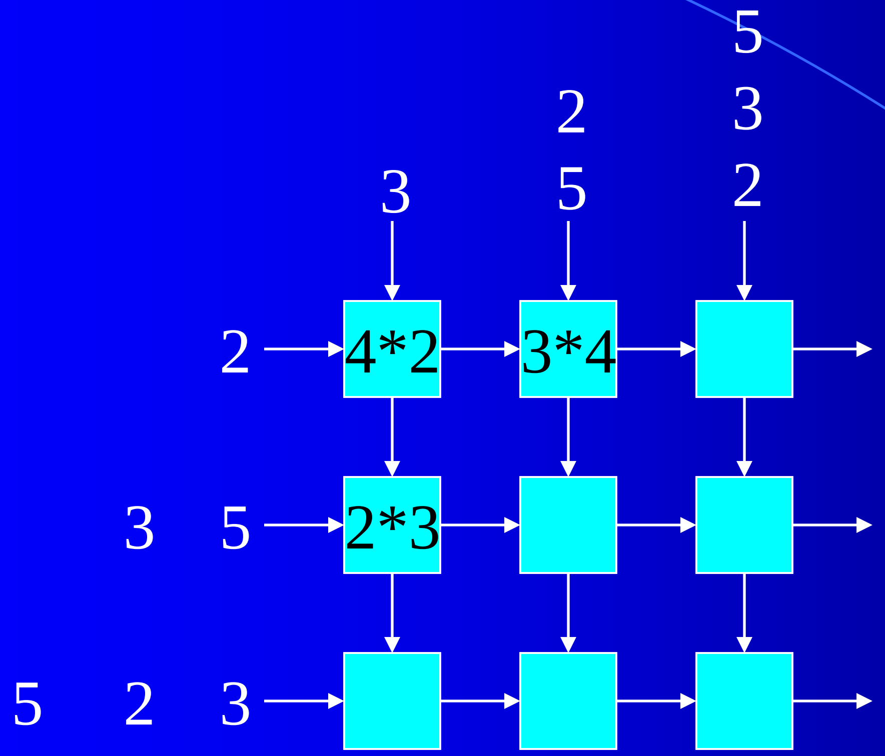
P7 P8 P9

Clock tick: 1



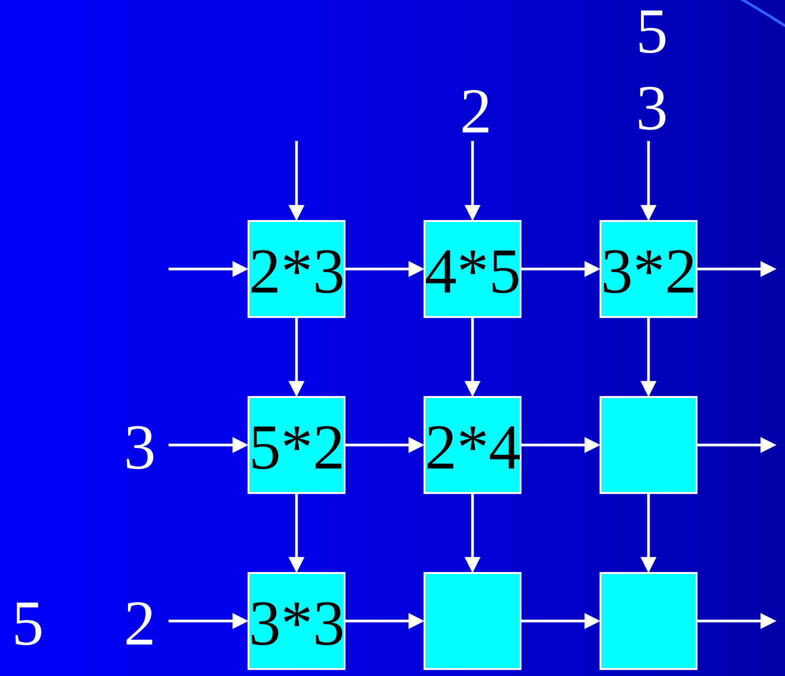
P1	P2	P3	P4	P5	P6	P7	P8	P9
9	0	0	0	0	0	0	0	0

Clock tick: 2



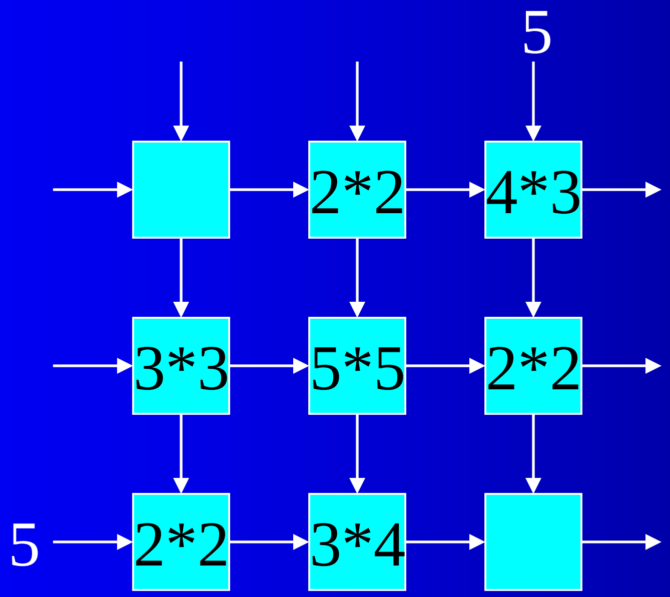
P1	P2	P3	P4	P5	P6	P7	P8	P9
17	12	0	6	0	0	0	0	0

Clock tick: 3



P1	P2	P3	P4	P5	P6	P7	P8	P9
23	32	6	16	8	0	9	0	0

Clock tick: 4



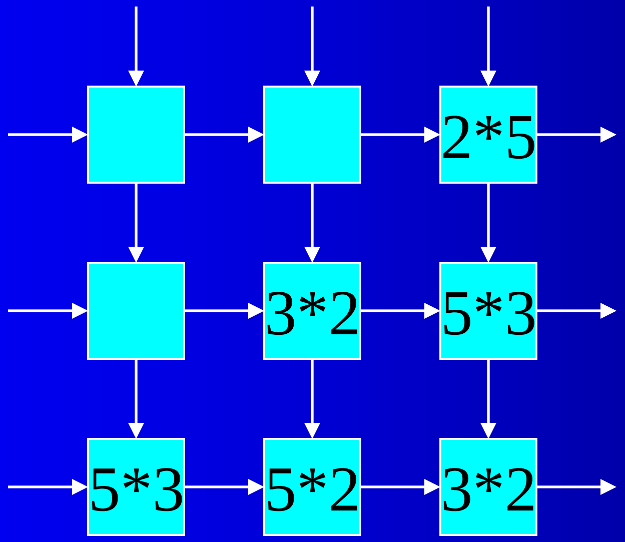
5

	2*2	4*3
3*3	5*5	2*2
2*2	3*4	

5

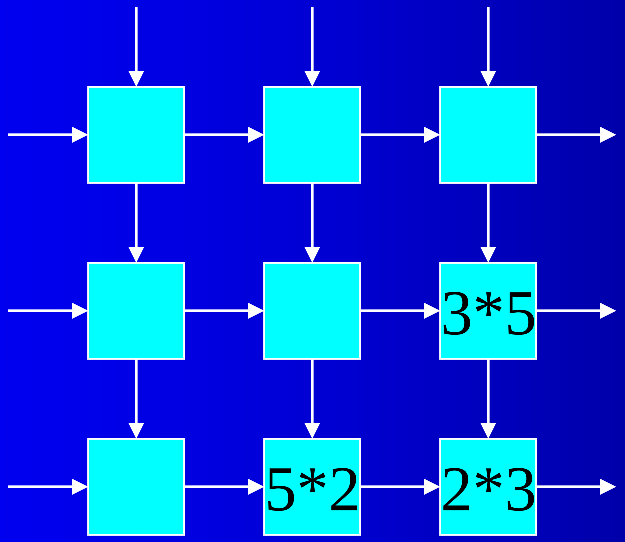
P1	P2	P3	P4	P5	P6	P7	P8	P9
23	36	18	25	33	4	13	12	0

Clock tick: 5



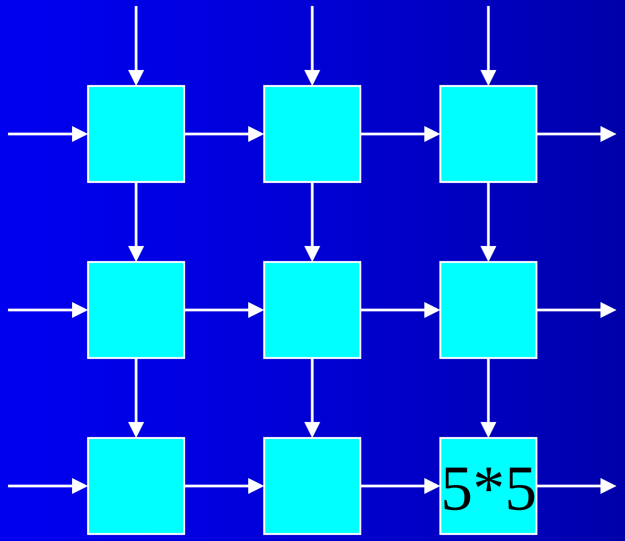
P1	P2	P3	P4	P5	P6	P7	P8	P9
23	36	28	25	39	19	28	22	6

Clock tick: 6



P1	P2	P3	P4	P5	P6	P7	P8	P9
23	36	28	25	39	34	28	32	12

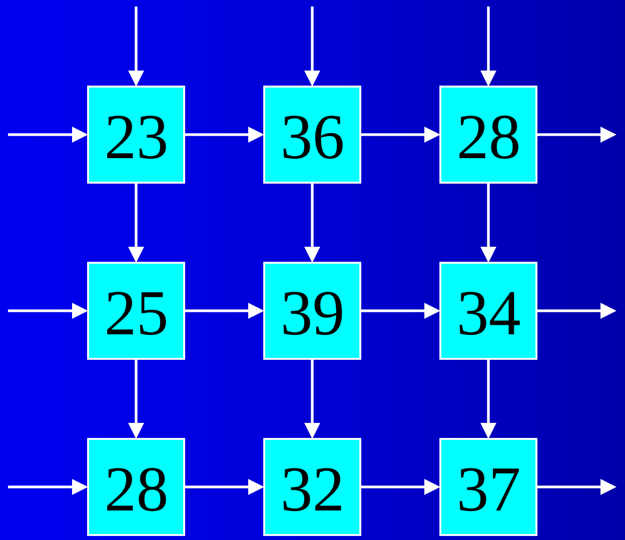
Clock tick: 7



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.



23	36	28
25	39	34
28	32	37

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