Politechnika Wrocławska

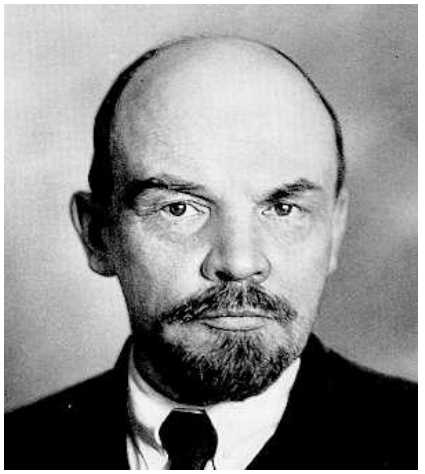
Podstawy Techniki Mikroprocesorowej

wykład 9: SIMD

Dr inż. Jacek Mazurkiewicz
Katedra Informatyki Technicznej
e-mail: Jacek.Mazurkiewicz@pwr.edu.pl

„Kompjuter – eta jest’” i klasyfikacja

- jednostka centralna – procesor
- pamięć operacyjna
- urządzenia wejścia-wyjścia
- magistrale
  - von Neumann (!)



- SISD - Single Instruction Single Data
- SIMD - Single Instruction Multiple Data
- MISD - Multiple Instruction Single Data
- MIMD – Multiple Instruction Multiple Data

SIMD - wprowadzenie

Architektury systoliczne (Kung, Leiserson)

- architektury specjalizowane do implementacji operacji macierzowych, przetwarzania sygnałów i obrazów w czasie rzeczywistym

- zbiór globalnie synchronizowanych procesorów elementarnych połączonych w regularną siatkę

- każdy procesor jest połączony tylko z najbliższymi sąsiadami

- struktura wewnętrzna procesora, w zależności od postawionego przed nim zadania, może być bardzo prosta (sumator, kilka rejestrów przesuwnych) lub dążyć do stopnia ukomplikowania współczesnych mikroprocesorów

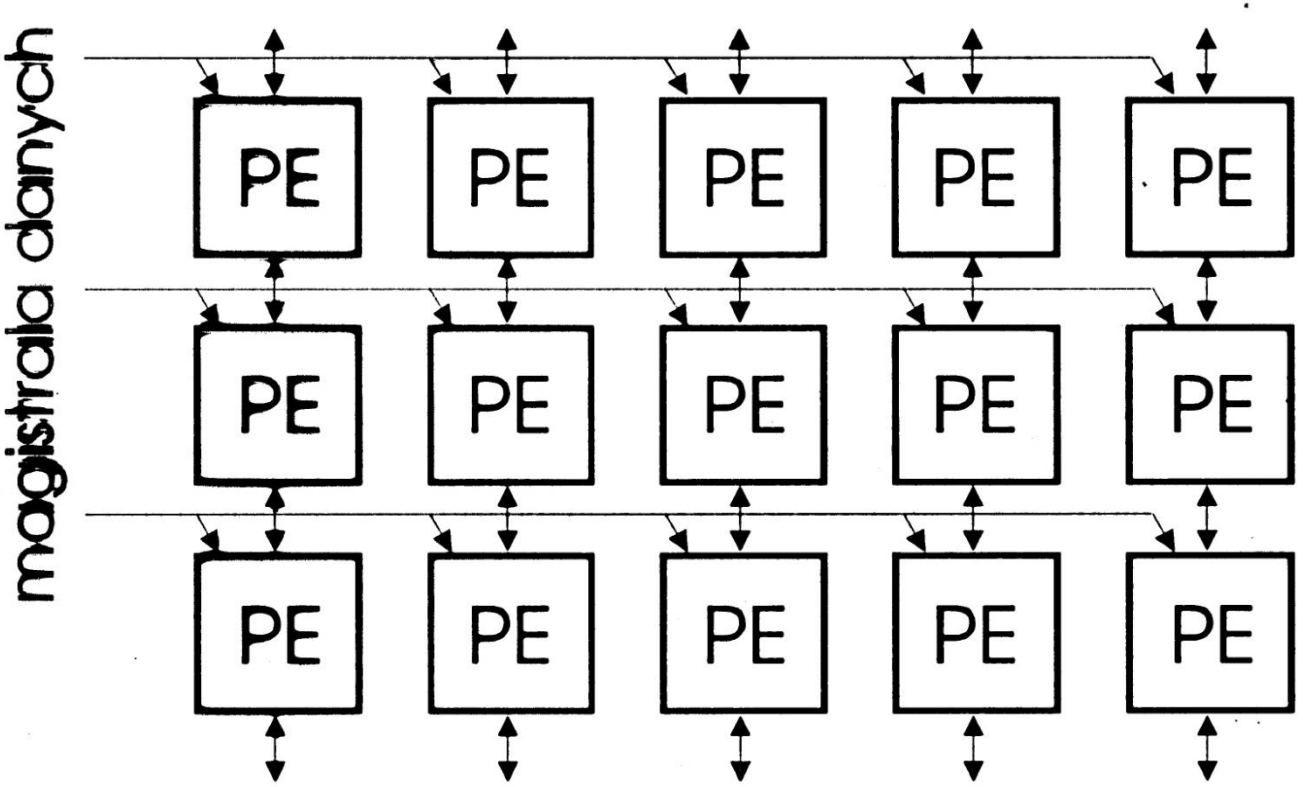
- dane i częściowe wyniki są przesyłane między procesorami również w takt wspólnego sygnału zegarowego

- wyniki przetwarzania danych uzyskiwane są stopniowo

Podstawowe struktury SIMD (1)

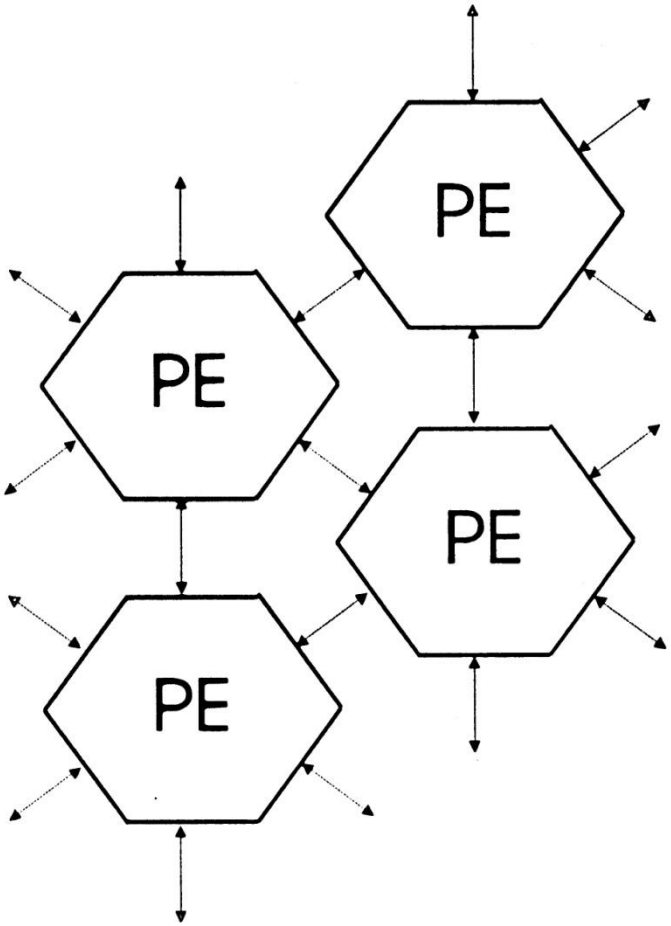
- tablica z częściowym rozpowszechnianiem danych

- Huang, Abraham



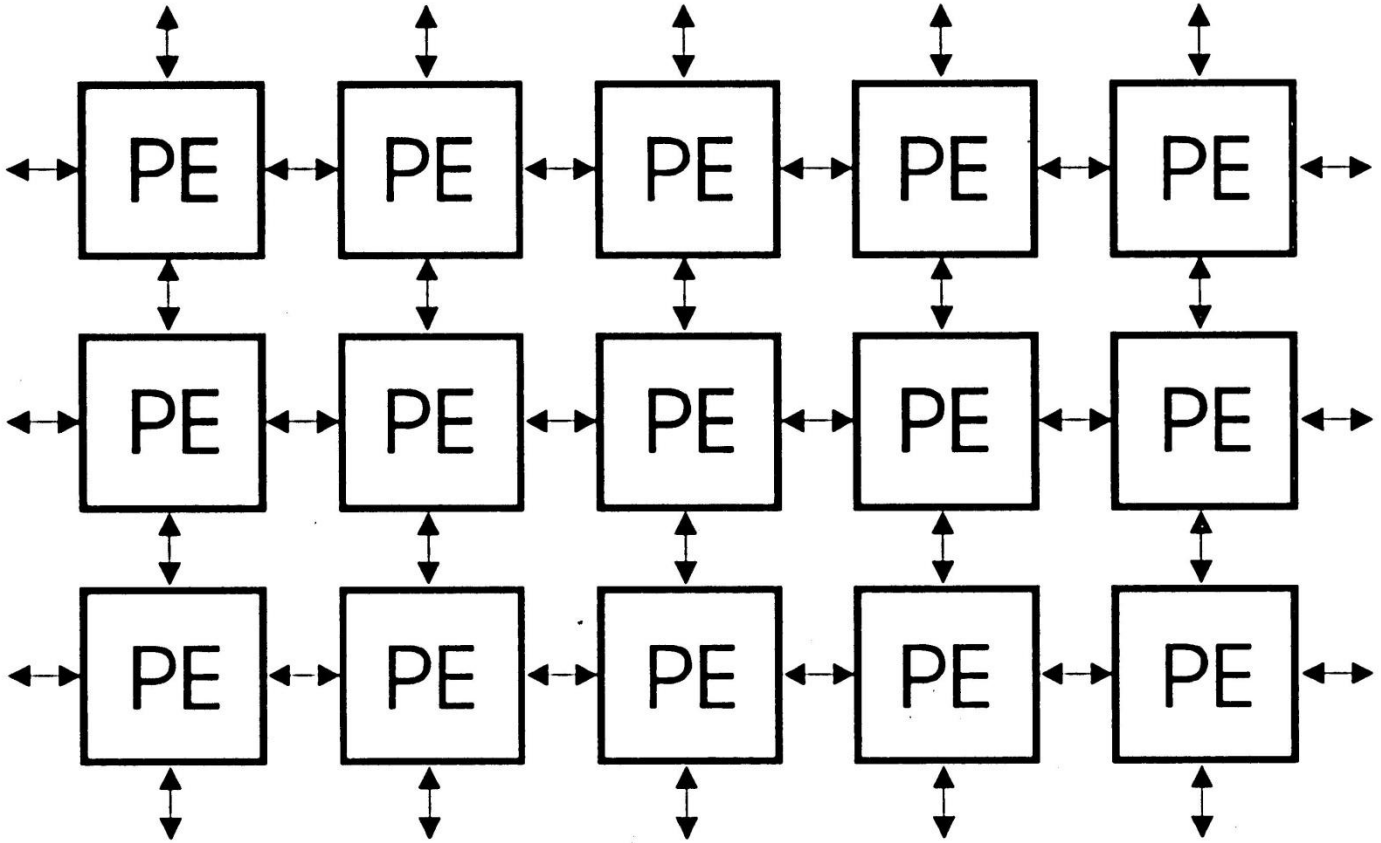
Podstawowe struktury SIMD (2)

- tablica hexagonalna - Kung, Leiserson



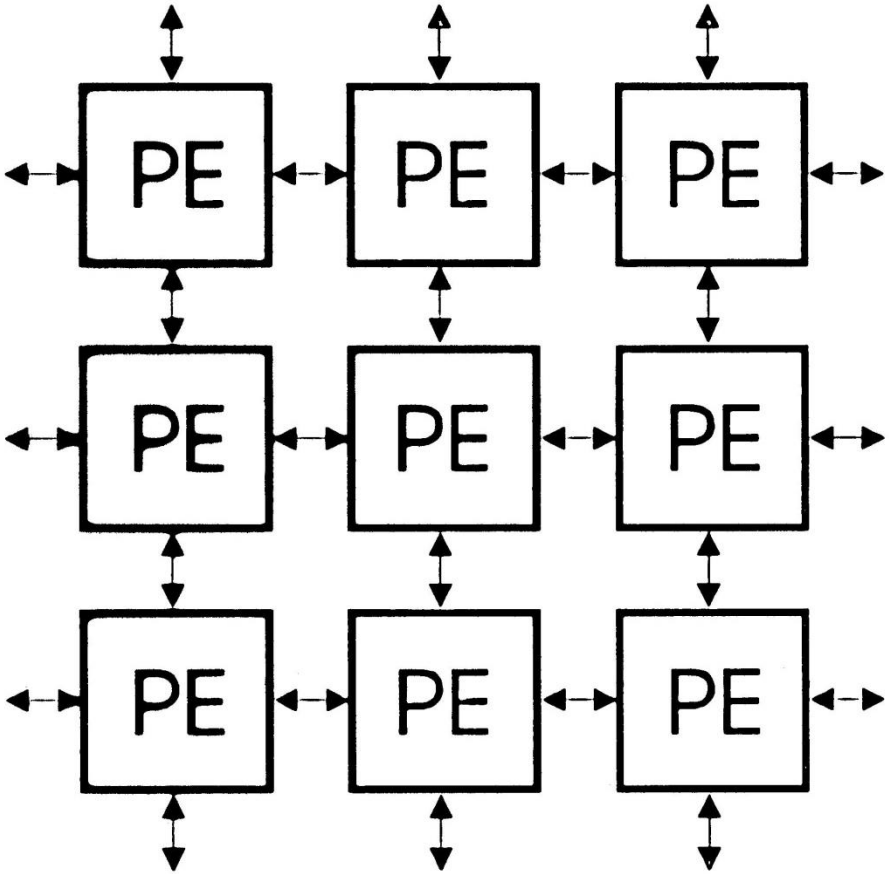
Podstawowe struktury SIMD (3)

- tablica przetwarzania potokowego - Hwang, Cheng



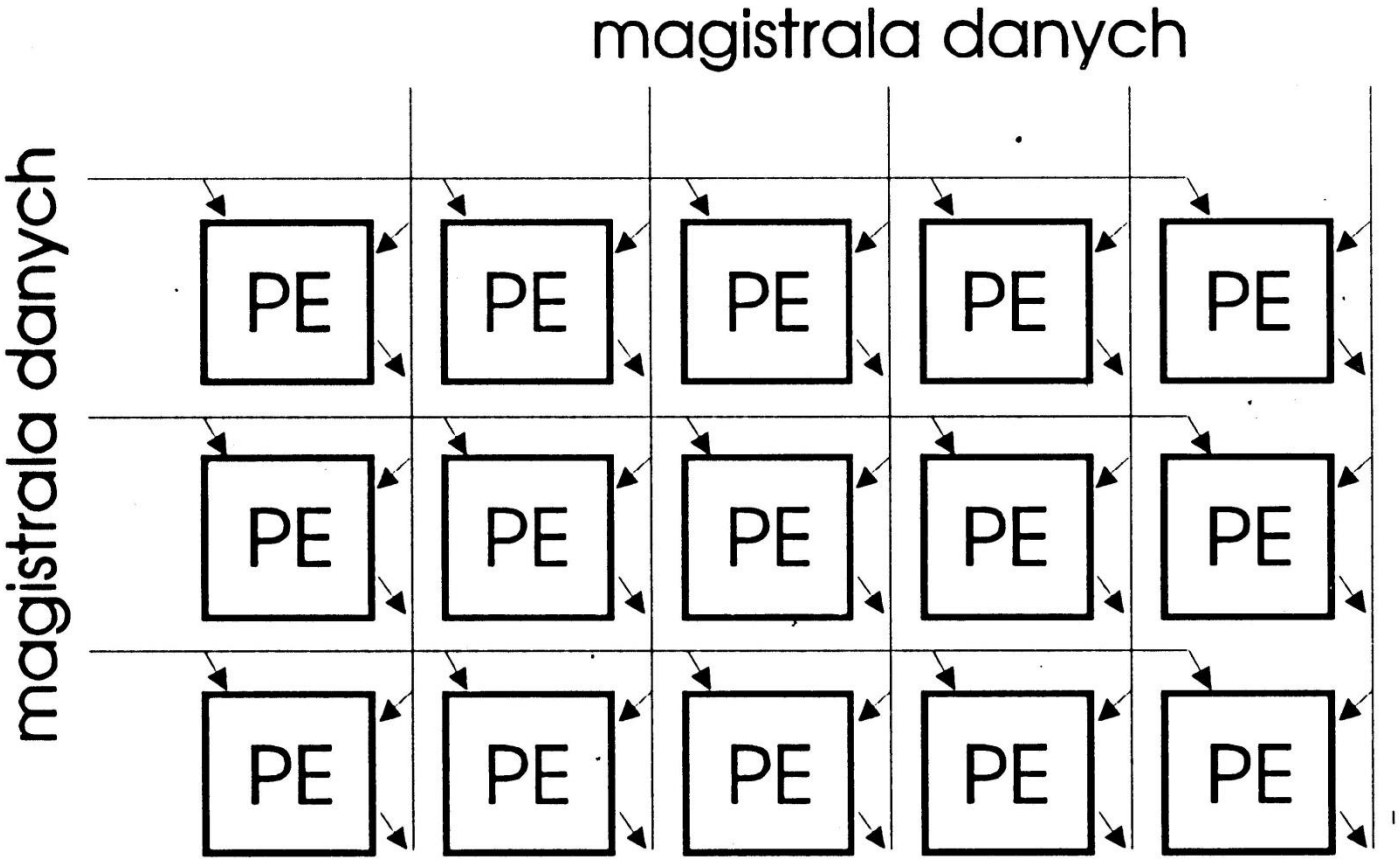
Podstawowe struktury SIMD (4)

- tablica typu wavefront - Kung, Arun



Podstawowe struktury SIMD (5)

- tablica z rozpowszechnianiem danych - Chern, Murata



Efektywność (1)

- tablica hexagonalna - Kung, Leiserson

  - komunikacja między PE: punkt - punkt
  - komunikacja zewnętrzna: tylko procesory brzegowe
  - zadanie mnożenia macierzy o wymiarach $n \times n$:

    liczba PE: $P = (2n-1)^2$

    czas realizacji zadania: $T = 4n-2$ $\qquad$ $T = 5n-3$

  - prosta konstrukcja
  - duża liczba stosowanych PE
  - niski stopień wykorzystania PE - zazwyczaj < 50%

Efektywność (2)

- tablica przetwarzania potokowego - Hwang, Cheng

  - komunikacja między PE: punkt - punkt
  - komunikacja zewnętrzna: tylko procesory brzegowe
  - zadanie mnożenia macierzy o wymiarach $n \times n$:

    liczba PE: $P = n(2n-1)$

    czas realizacji zadania: $T = 4n-2$
  - prosta konstrukcja
  - liczba PE mniejsza niż w tablicy hexagonalnej
  - stopień wykorzystania PE - przynajmniej 50%

Efektywność (3)

- tablica z częściowym rozpowszechnianiem danych

- Huang, Abraham

- komunikacja między PE: punkt - punkt, magistrala
- komunikacja zewnętrzna: poprzez magistralę
- zadanie mnożenia macierzy o wymiarach $n \times n$:
  - liczba PE: $P = n^2$
  - czas realizacji zadania: $T = 2n$ $T = 3n$
- rozbudowane układy sterujące
- stopień wykorzystania PE - ponad 50%

Efektywność (4)

- tablica typu wavefront - Kung, Arun

  - komunikacja między PE: punkt - punkt asynchronicznie
  - komunikacja zewnętrzna: tylko procesory brzegowe
  - zadanie mnożenia macierzy o wymiarach $n \times n$:

    liczba PE: $P = n^2$

    czas realizacji zadania: $T = 3n-2$ $T = 4n-2$
  - dobra skalowalność
  - łatwe przeprogramowywanie
  - dobre parametry FTC
  - stopień wykorzystania PE - zazwyczaj 50%

Efektywność (5)

- tablica z rozpowszechnianiem danych - Chern, Murata

  - komunikacja między PE: magistrale
  - komunikacja zewnętrzna: magistrale
  - zadanie mnożenia macierzy o wymiarach $n \times n$:

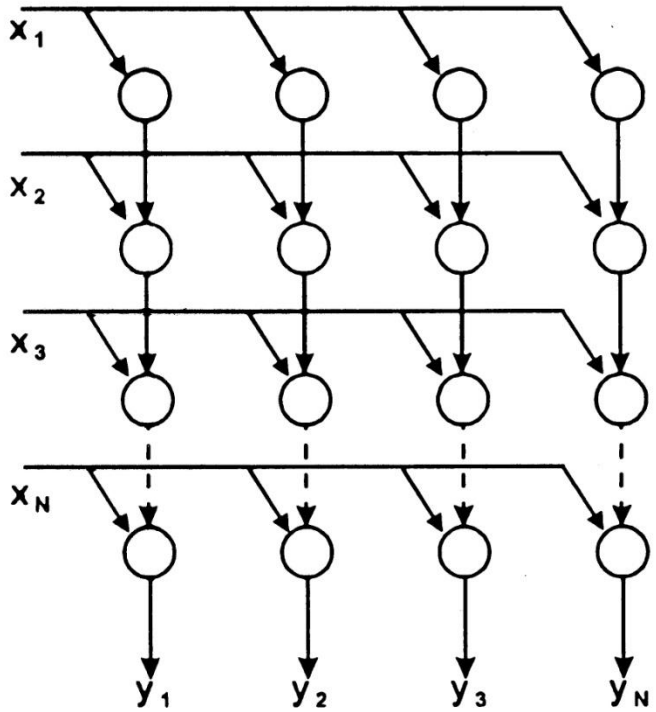
    liczba PE: $P = n^2$

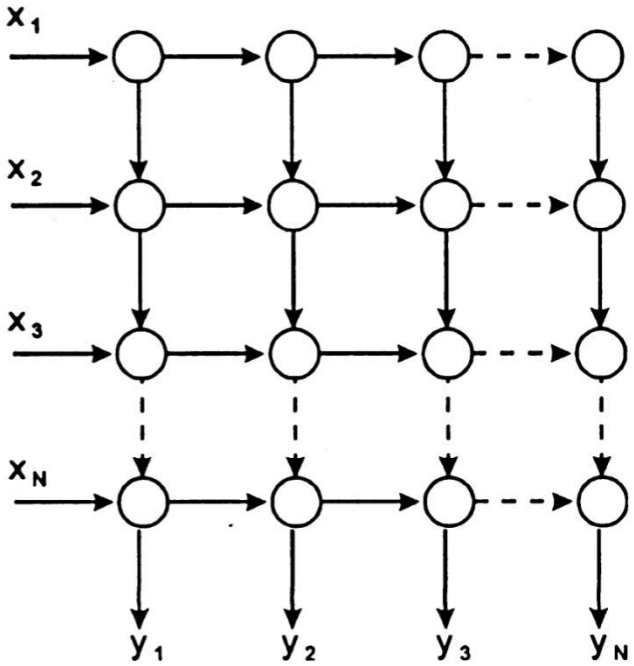
    czas realizacji zadania: $T = n$ $\qquad$ $T = 2n$
  - rozbudowane układy sterujące
  - łatwa implementacja algorytmów
  - wysoki stopień wykorzystania PE

Graf zależności

- zlokalizowany graf zależności

- kompletny graf zależności





Graf przepływu (1)

Założenia o wektorze liniowej projekcji $\vec{d}$ Grafu Zależności na przestrzeń procesorów elementarnych i wektorze następstwa czasowego $\vec{s}$:

$$\vec{s}^{\,T}\vec{e} > 0$$

$\vec{e}$ - wektor reprezentujący dowolny łuk w Grafie Zależności

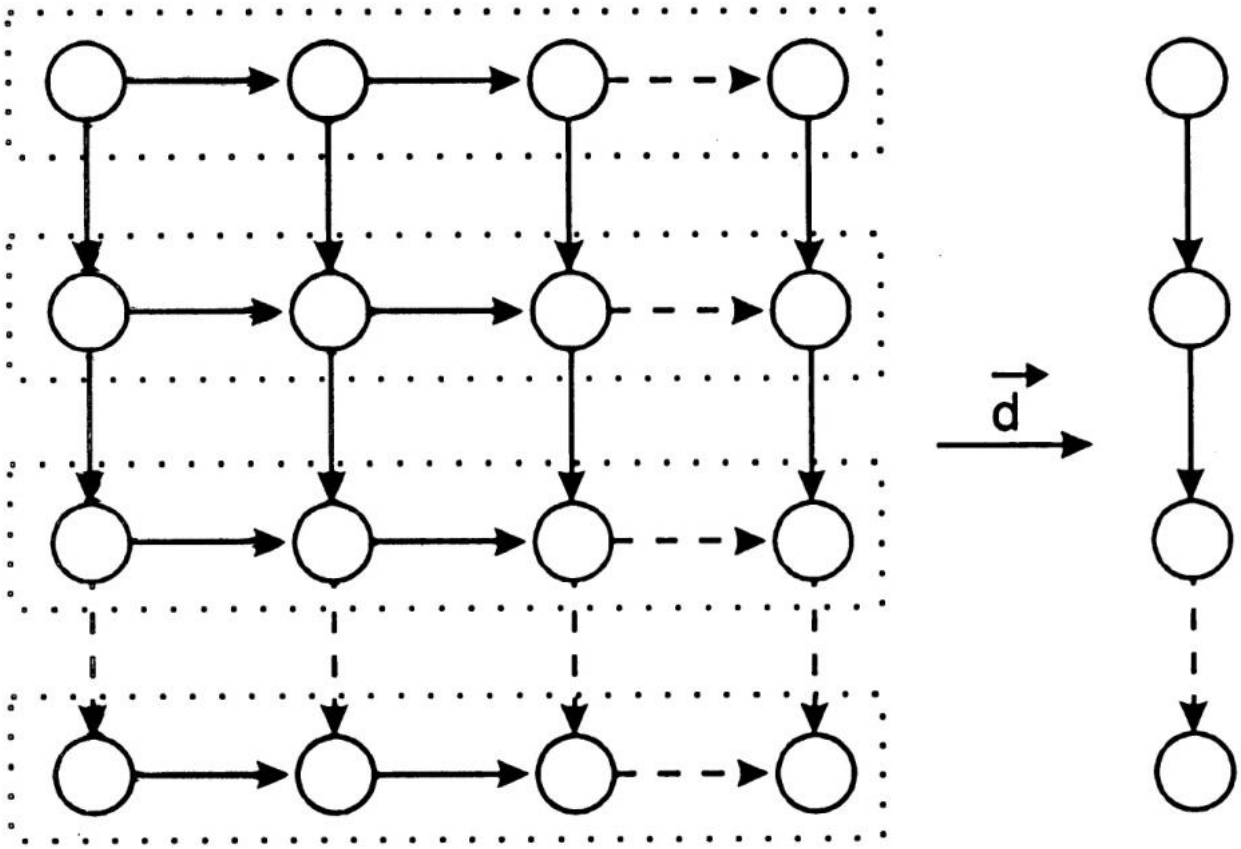
Elementy przetwarzające są oddzielone przynajmniej jednym łukiem

$$\vec{s}^{\,T}\vec{d} > 0$$

Zachowanie następstwa czasowego w Grafie Zależności

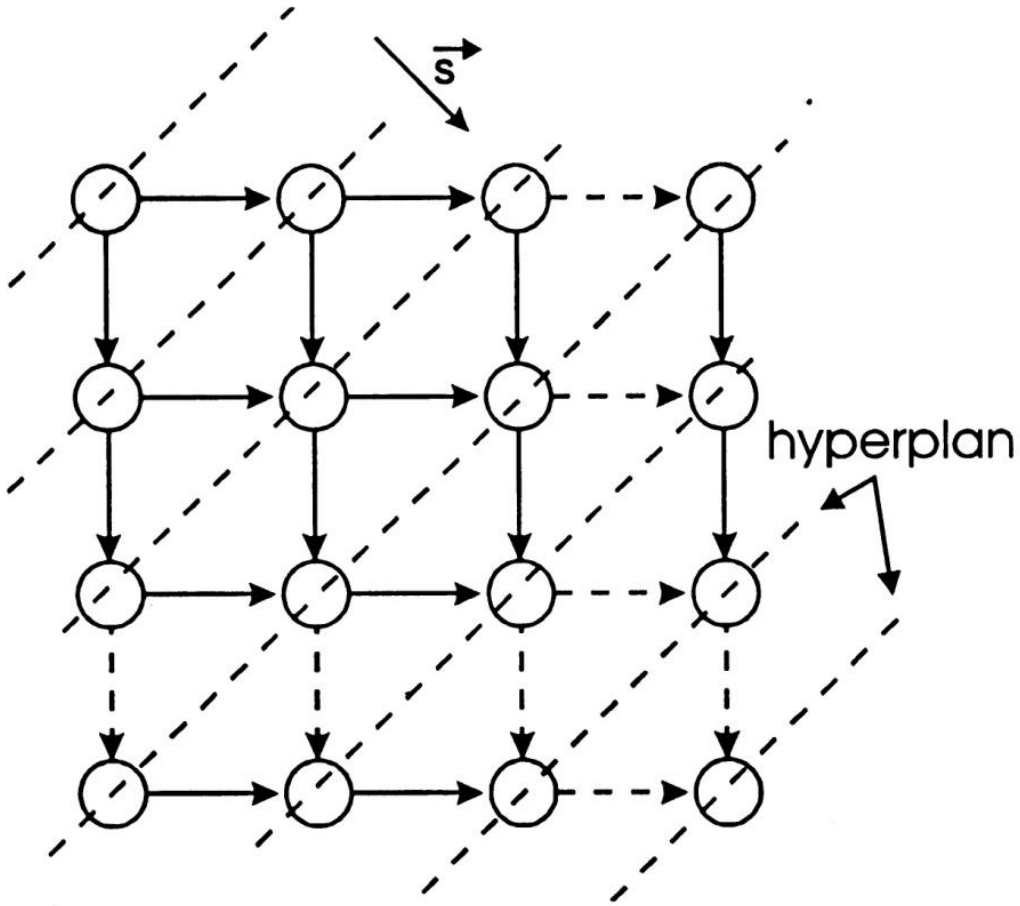
Graf przepływu (2)

- przyporządkowanie operacji do PE



Graf przepływu (3)

- szeregowanie zadań w przestrzeni czasu



Jakość (1)

Computation Time:

This is the time interval between starting the first computation and finishing the last computation of a problem.

Pipelining Period

This is the time interval between two successive computations in a processor.

Block Period:

This is the time interval between the initiation of two successive blocks.

Jakość (2)

Processor Utilization Rate:

$$speed - up = \frac{sequential\ computation\ time}{array\ computation\ time}$$

$$utilization\ rate = \frac{speed - up}{number\ of\ processors}$$

Jakość (3)

Array Processor Size:

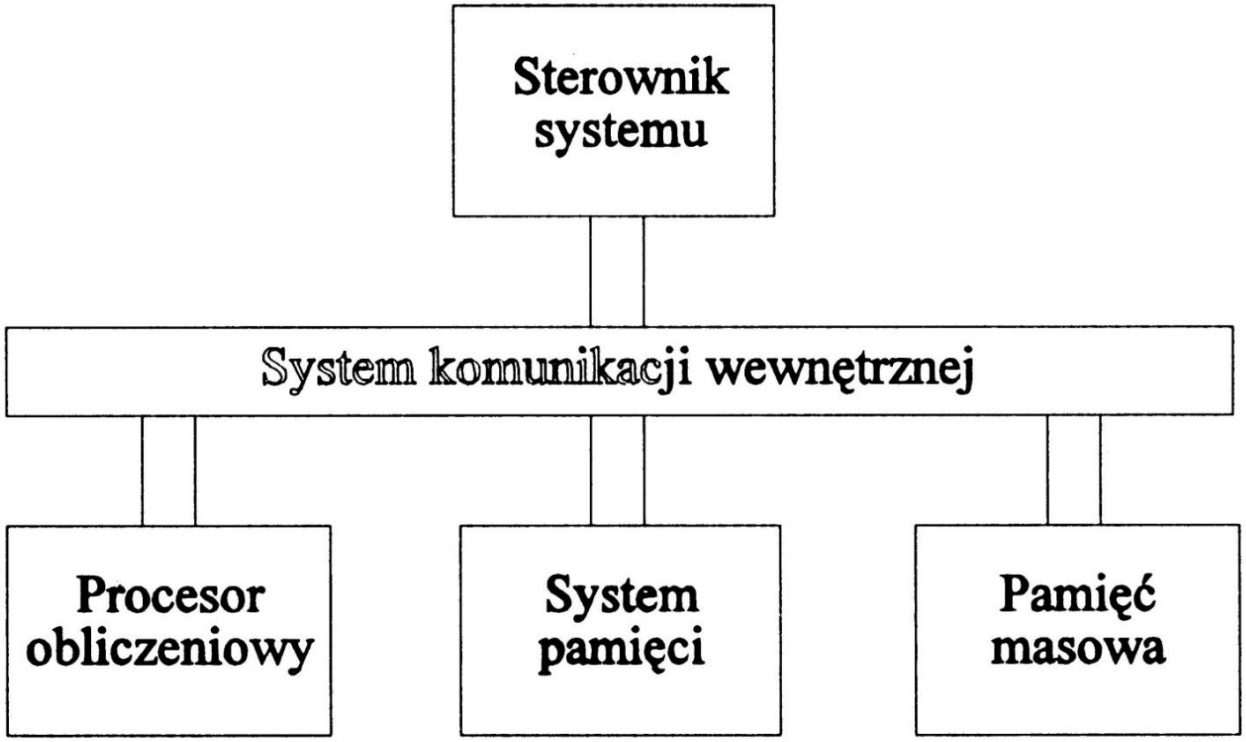
The array size cannot be unlimited. Sometimes, the size of the array is much smaller than the full problem size. In another situation, a one-dimensional array is used in place of an idealistic two-dimensional array. This incurs the so-called partitioning problem.

Local Memory:

Local memory can be used to expand the physical array to a much larger virtual array size.

Komercyjne struktury SIMD (1)

System Saxpy Matrix-1 - Fousler, Schreiber

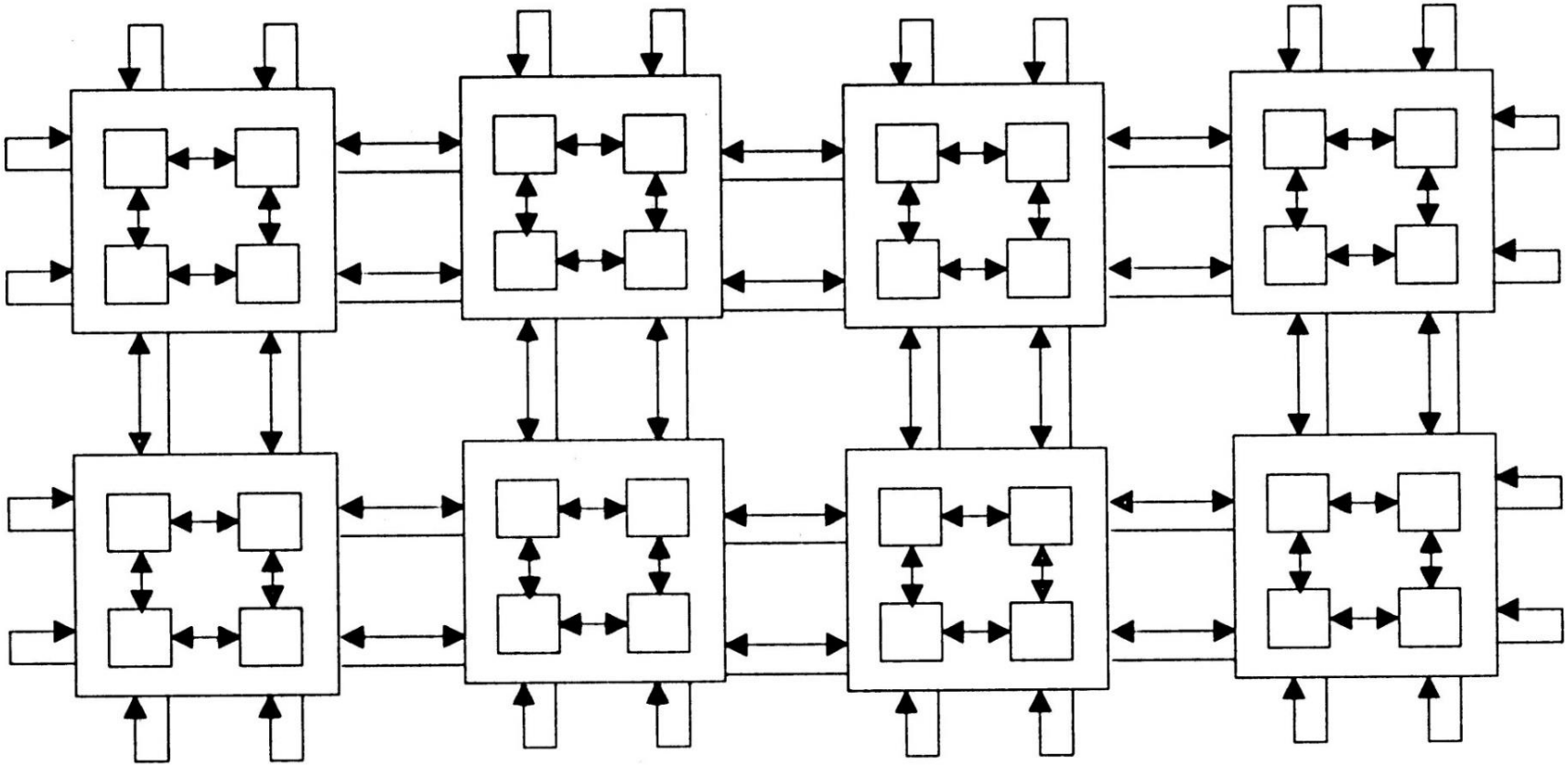


Komercyjne struktury SIMD (2)

- sterownik systemu - komputer, który podejmuje realizację zadania i odpowiednio alokuje zasoby,
- procesor obliczeniowy zbudowany w postaci łańcucha o maksymalnie 32 potokowych, zmiennoprzecinkowych jednostkach z zapewnioną zarówno lokalną (punkt-punkt), jak i globalną komunikacją między nimi,
- system pamięci gromadzący wszytkie dane używane przez procesor opisany uprzednio,
- system masowej pamięci z łączem pozwalającym na szybką wymianę danych z urządzeniami peryferyjnymi,
- system nadzorujący prawidłową współpracę wszystkich elementów.

Komercyjne struktury SIMD (3)

System iWARP - Carnegie Mellon University, INTEL



Komercyjne struktury SIMD (4)

Pojedynczy element:

procesor obliczeniowy i procesor komunikacyjny
4 kanały komunikacyjne przepustowość każdego: 40MB/s
20 MFLOP -32 bitowa precyzja,
10 MFLOP - 64-bitowa precyzja obliczeń

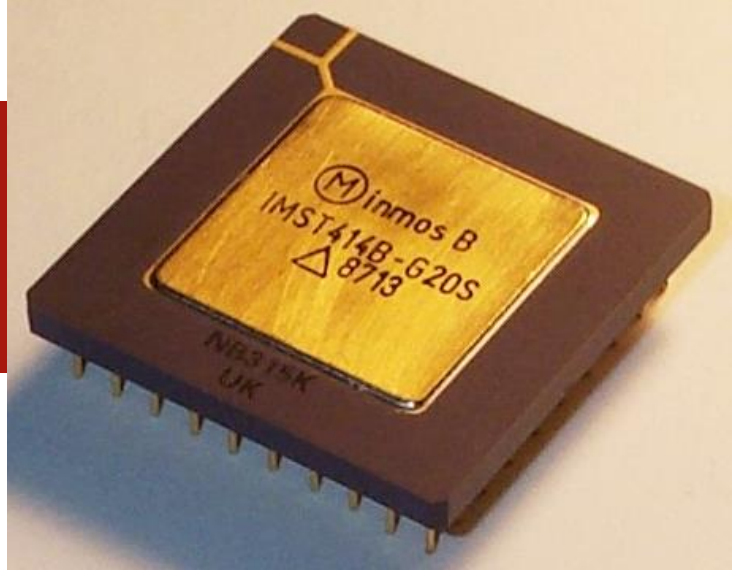
Procesor obliczeniowy:

15 rejestrów - gromadzenie danych,
współdzielona pamięć: 32-bitowe komórki
całkowitoliczbowa 32-bitowa ALU, 20 MIPS
zmiennoprzecinkowa ALU

Procesor komunikacyjny:

właściwa komunikacja systoliczna: punkt - punkt
przekazanie danych z pominięciem
procesora obliczeniowego

Transputery (1)



- transistor + computer
- INMOS – Bristol (UK)
- pracują „stadem”
- komunikacja punkt-punkt
- każdy ma 4 sąsiadów
- 4 kanały: 5, 10, 20 Mb/s
- CPU stały przecinek
- koprocesor
- pamięć lokalna

Transputery (2)

- komputer komunikacyjny
- procesory RISC
- scheduler
- booting
- OCCAM
- C
- 16-bit
- 32-bit
- 64-bit

Mnemonic	Description
J	Jump — add immediate operand to instruction pointer.
LDLP	Load Local Pointer — load a Workspace-relative pointer onto the top of the register stack
PFIX	Prefix — general way to increase lower nibble of following primary instruction
LDNL	Load non-local — load a value offset from address at top of stack
LDC	Load constant — load constant operand onto the top of the register stack
LDNLP	Load Non-local pointer — Load address, offset from top of stack
NFIX	Negative prefix — general way to negate (and possibly increase) lower nibble
LDL	Load Local — load value offset from Workspace
ADC	Add Constant — add constant operand to top of register stack
CALL	Subroutine call — push instruction pointer and jump
CJ	Conditional jump — depending on value at top of register stack
AJW	Adjust workspace — add operand to workspace pointer
EQC	Equals constant — test if top of register stack equals constant operand
STL	Store local - store at constant offset from workspace
STNL	Store non-local - store at address offset from top of stack
OPR	Operate - general way to extend instruction set