9

Classification of Parallel Computer Architectures

SIMD

Criteria

- Organization of the Control and Data Flow
- Address Space Organization
- Use of Physical Memory

Why Parallel Computing?

Although speed of computers is steadily increasing, new applications require even higher speed.

Performance of serial computers is beginning to saturate.

A natural way to solve the problem is the usage of an ensemble of processors

⇒ parallel computers

Main Purposes for Parallel Computing

A parallel computer is simply a collection of processors, interconnected in a certain fashion to allow the *coordination of their activities* and the *exchange of data*

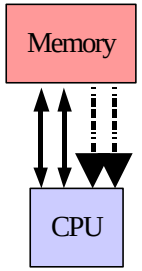
Parallel computers **need parallel algorithms, i.e.** algorithms suitable for implementation on parallel computers

Faster solutions

Solution of large size problems

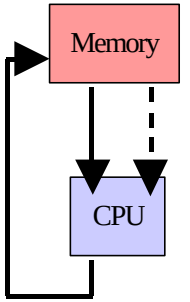
Flynn's Classes of Computer Architectures

- SI: Single Instruction Stream
- MI: Multiple Instruction Stream
- SD: Single Data Stream
- MD: Multiple Data Stream



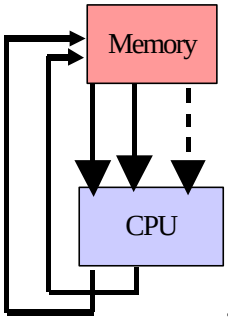
Flynn's Classes of Computer Architectures

- SISD: Single Instruction Stream-Single Data Stream
  - Monoprocessor
- Princeton (Von Neumann) -Architecture:
  - common instruction and data stream
- Harward - Architecture
  - separate instruction and data streams



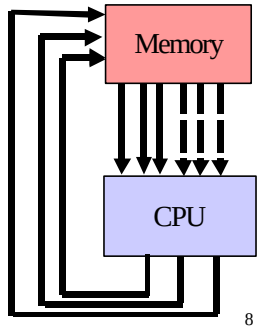
Flynn's Classes of Computer Architectures

- SIMD: Single Instruction Stream-Multiple Data Stream
  - array processors, vector computers



Flynn's Classes of Computer Architectures

- MIMD: Multiple Instruction Stream-Multiple Data Stream



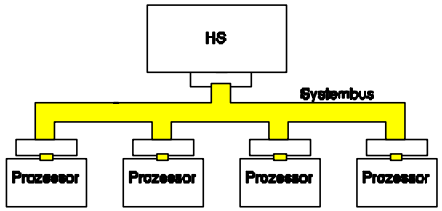
Degree of Parallelism

Moderate Parallel Computers:
SMP (Symmetric Multiprocessor)

Massively Parallel Processor: MPP

- Hugh number of processing nodes
- Hugh switching network
- Single computing resource - for a single job

Symmetric Multiprocessor

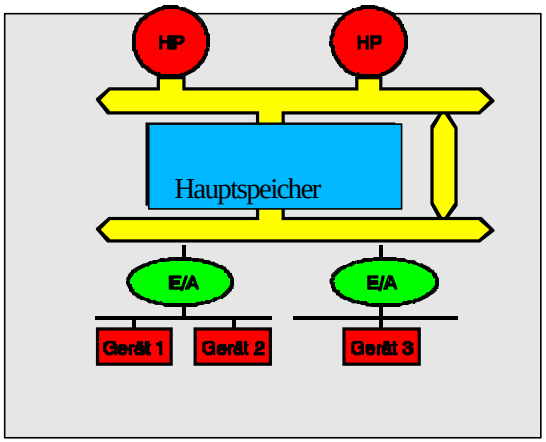


In einem (symmetrischen) Multiprozessorsystem können alle Prozessoren auf einen gemeinsamen Hauptspeicher zugreifen.

Dadurch ist es ihnen möglich, Daten auszutauschen.

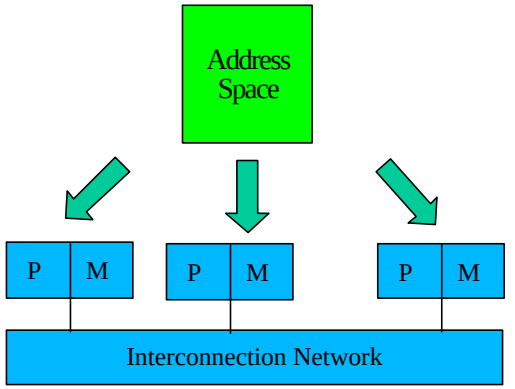
Den Prozessoren steht aber nur ein gemeinsamer Datenweg zum Hauptspeicher zur Verfügung, z.B. ein **gemeinsamer Bus**.

Non-symmetric Multiprocessor



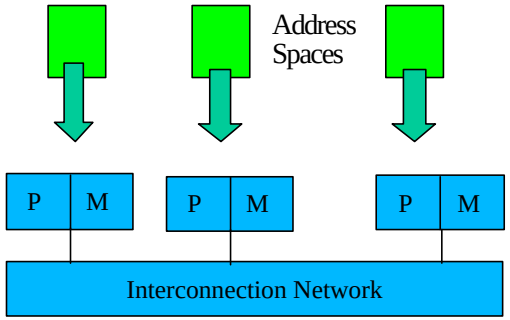
Haupt- und Ein/Ausgabe-Prozessoren

Global Address Space
Distributed Physical Memory



Verbindungsnetz, interconnection network, interconnect

Local Address Spaces



distributed and private physical memory

Parallel Architectures

- Control / Data Flow: centralized or distributed (Flynn)
- Address Space: local or global
- Physical Memory: private or shared
- Type of Interconnection Network

Classification of Parallel Architectures

Fife Dimensions

Control:	centralized or distributed
Data Flow:	sequential or parallel
Address Space:	local or global
Physical Memory:	central or distributed
Interconnect:	static or dynamic

SISD: von Neumann

Fife Dimensions

Control:	centralized
Data Flow:	sequential
Address Space:	global
Physical Memory:	central
Interconnect:	static

Parallel Architectures

SIMD: Vector Computers
Array Processors

Control: centralized

Data Flow: parallel

Address Space: global

Physical Memory: central or distributed

Interconnect: static

Vector Computers

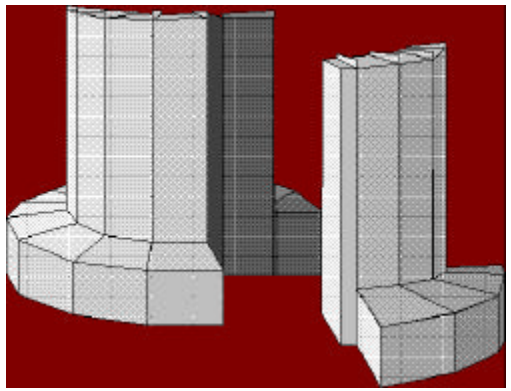
Cray-1
Cray-2
Cray C-90

NEC

Fujitsu

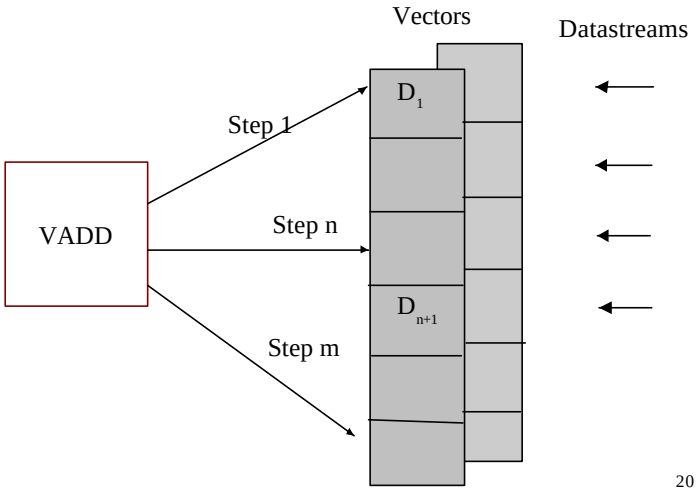
They all have different architectures.

Vector Computer
(vector processor)

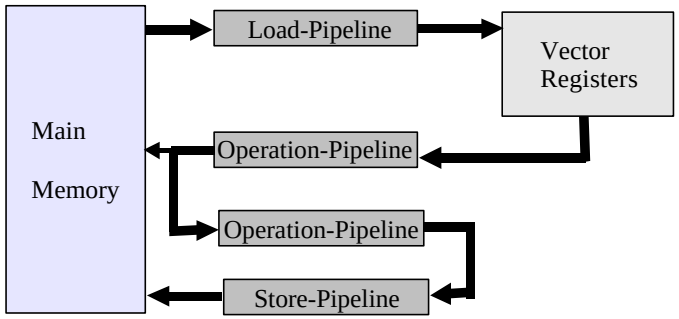


Powerful CPUs
Large Memory
Several Vector Pipelines

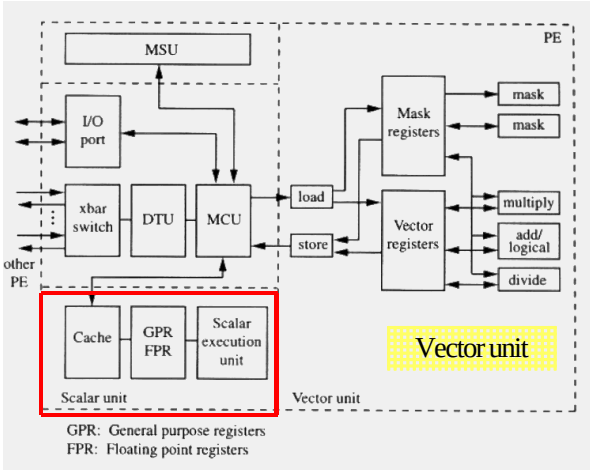
Example: Vector Operation



Vector Pipelining



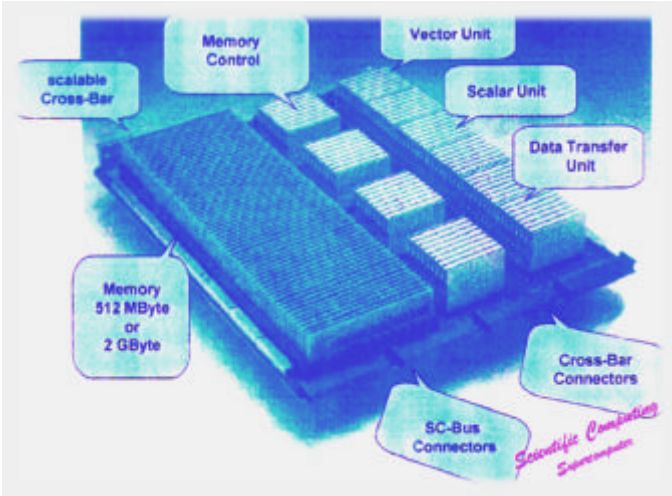
Vector Processor



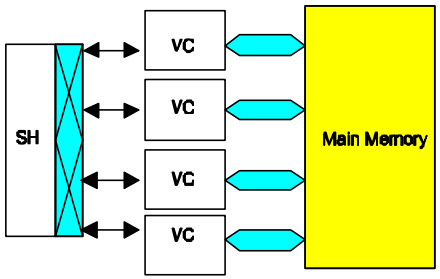
MSU: memory
MCU: memory control unit
DTU: data transfer unit

← arithmetic pipelines

Hardware Example



Example: MSIMD-Vector Computer



VC Vector Computer
SH Synchronisation Hardware

Global Address Space

Global or Shared Address Space:
Processors interact by modifying data in a shared address space.

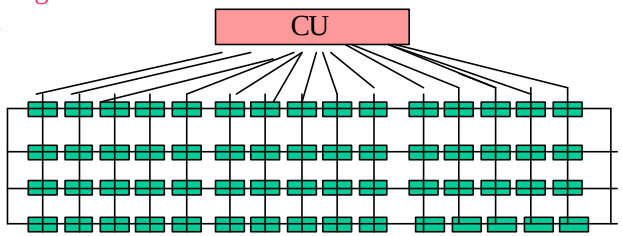
Hardware support exists for coordinating read and write accesses by the processors.

Performance of Vector Computers MSIMD

Name	Vector registers	Elements per vector register	Elements computed per clock cycle	Number of functional units	Processor clock rate	Maximum number of processors	Maximum memory size/system
Cray J90	8	64	1	4	100 MHz	32	8,192 MB
Cray T90	8	128	2	8	455 MHz	32	8,192 MB
Fujitsu VPP300	8–256	64–2048	8	4	140 MHz	16	32,768 MB
NEC SX-4 single node	8 + 8192 scratchpad	256 + variable up to 8K	8	16	125 MHz	32	8,192 MB

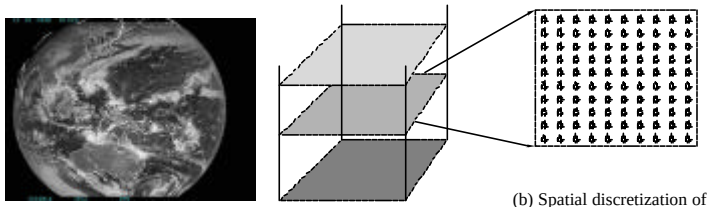
Array Processors

Central Program Control



Massively parallel:
Very many processing units PU
Large memory
Powerful I/O

Simulating Ocean Currents



(a) Cross sections

(b) Spatial discretization of a cross section

- Model as two-dimensional grids
  - Discretize in space and time
  - finer spatial and temporal resolution => greater accuracy
- Many different computations per time step
  - » set up and solve equations
  - Concurrency across and within grid computations
- Static and regular

Array Processors

Centralized control Þ single control unit

Single-Instruction Stream - Multiple-Data Straem, (SIMD)

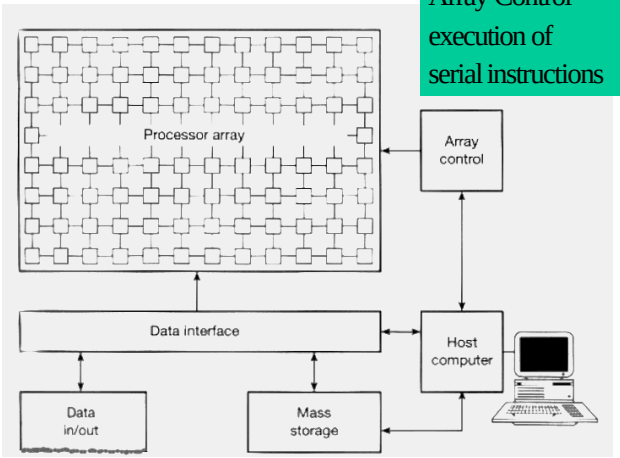
The control unit dispatches instructions to each processing unit.

Processing units can be selectively switched out.

Primarily for data parallel programs

Overall Architecture of a Processor Array

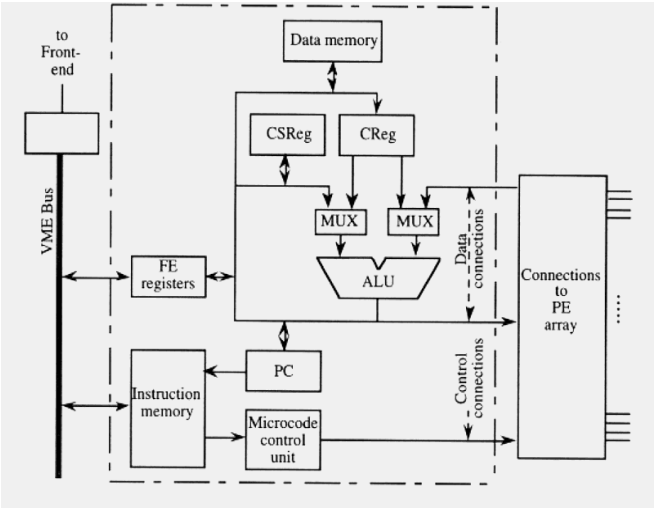
Array Control execution of serial instructions



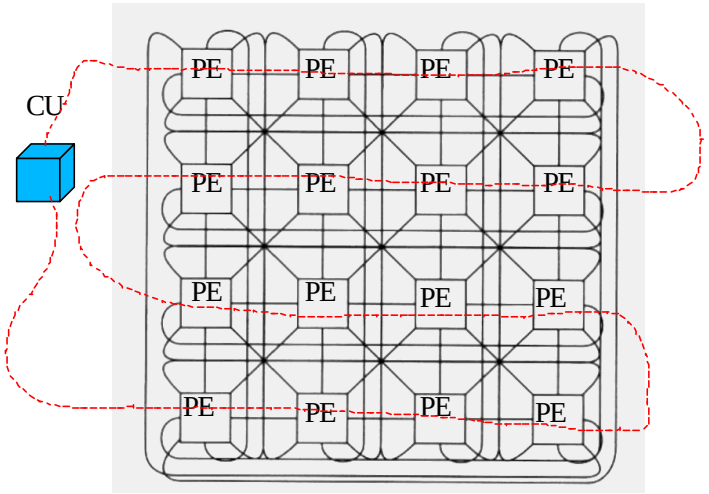
SIMD: Array Processors

Institution	Name	Maximum no. of proc.	Bits/ proc.	Proc. clock rate (MHz)	Number of FPUs	Maximum memory size/system (MB)	Communi-cations BW/system (MB/sec)	Year
U. Illinois	Illiac IV	64	64	5	64	0.125	2,560	1972
ICL	DAP	4,096	1	5	0	2	2,560	1980
Goodyear	MPP	16,384	1	10	0	2	20,480	1982
Thinking Machines	CM-2	65,536	1	7	2048 (optional)	512	16,384	1987
Maspar	MP-1216	16,384	4	25	0	256 or 1024	23,000	1989

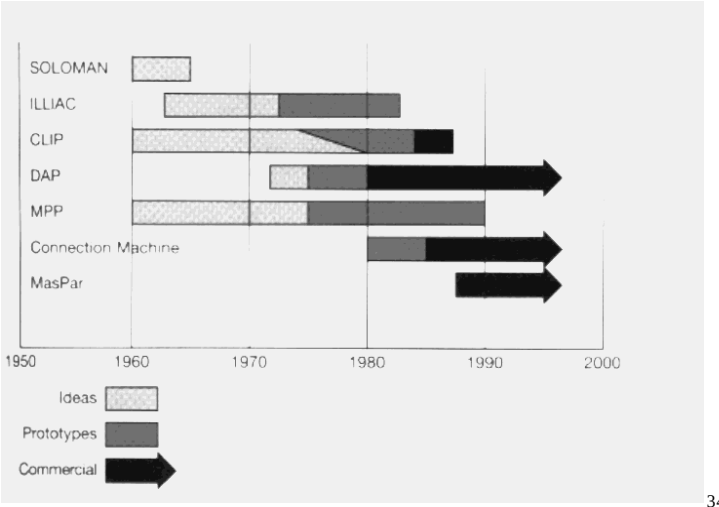
Example: MasPar Control Unit



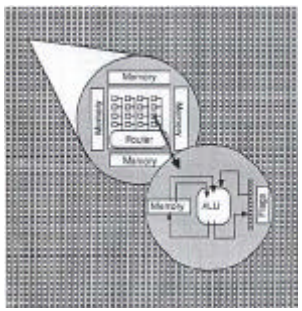
MasPar: Interconnection Networks

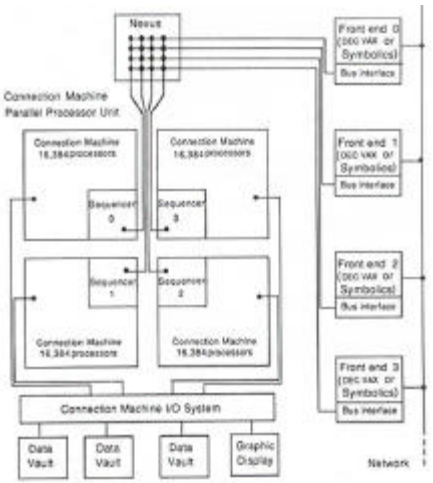


Some SIMD-Array Processors



Connection Machine

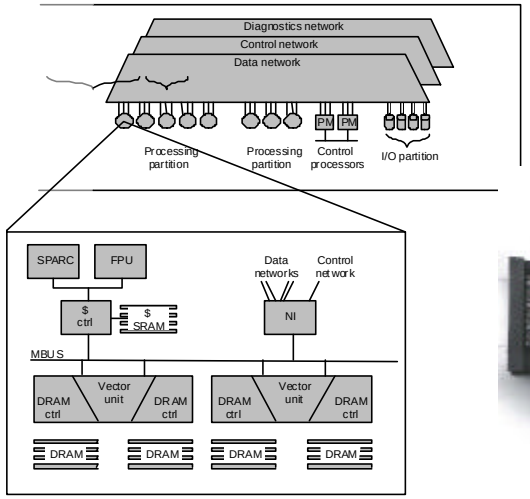




CM-5

- Repackaged SparcStation
  - 4 per board
- Fat-Tree network
- Control network for global synchronization

CM-5 Machine Organization



Evolution

- SIMD popular when cost savings of centralized sequencer high
  - 60s when CPU was a cabinet
  - Replaced by vectors in mid-70s
    - More flexible w.r.t. Memory layout and easier to manage
  - Revived in mid-80s when 32-bit datapath slices just fit on chip
- Simple, regular applications have good locality

- Programming model
  - needs fast global synchronization
  - structured global address space