→ Definition typischer Merkmale/Kriterien
zur Einordnung von Parallelrechnersystemen

5.1 Motivation

- Klassifikation als Hilfsmittel zur Strukturierung gut geeignet
- erleichtert Übersicht und Vergleich
- insbes. im Bereich Parallelrechner sehr viele verschied. Lösungen
- nur wenige Standards
- z.T. unterschiedl. Begriffe für gleichen Gegenstand
- hohe Dynamik der Entwicklung (auch Ablösung ehemals guter Konzepte)
- Klassifikation nur bzgl. Hardware reicht heute i.allg. nicht mehr, denn Software beeinflusst das Gesamtsystem wesentlich (z.B. via Fähigkeiten des Compilers, des BS, usw.)

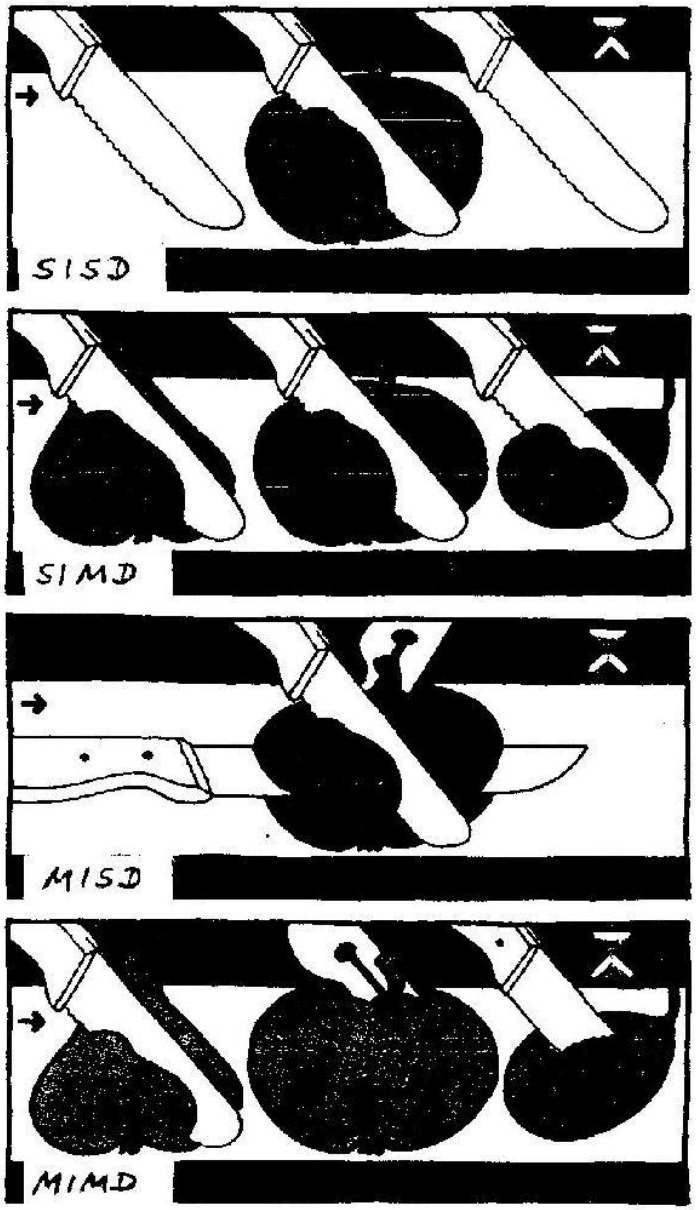
Eine erste Klassifikation könnten übrigens die „Ebenen der Parallelität“ sein (vgl. Kap. 1.6)

5.2 Klassifikation nach FLYNN (1972) (I)

- älteste Taxonomie für Parallelrechner
- einfach, daher (bzgl. Hardware) immer noch oft benutzt
- Rechner werden durch 2 Informationsströme charakterisiert:
  - Befehlsstrom (instructions)
  - Datenstrom (data)
- qualitative Unterscheidung bzgl.
  - einfache Ströme
  - mehrfache Ströme
- quantitative Aspekte unberücksichtigt
- Anordnung von Prozessoren, Speicher, Verbindungsnetz sowie Rolle der Software ebenfalls unberücksichtigt

5.2 Klassifikation nach FLYNN (1972) (II)

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



(Bild-Quelle unbekannt)

5.2 Klassifikation nach FLYNN (1972) (III)

- SISD Single Instruction, Single Data
  - es gibt nur eine Verarbeitungseinheit
  - dieser Prozessor hat Zugriff auf nur *einen Programmspeicher* und nur *einen Datenspeicher*
  - pro Verarbeitungsschritt ist die Abarbeitung *eines Befehls über einem Datenelement* möglich
  - = klassischer Von-Neumann-Rechner (also z.B. klass. PC)
  - rein sequentielle Verarbeitung, keine Parallelität

Bild-Quelle: Rauber/Rünger: Parallele und verteilte Programmierung. Springer, 2000

5.2 Klassifikation nach FLYNN (1972) (IV)

- MISD Multiple Instruction, Single Data
  - es gibt mehrere Verarbeitungseinheiten
  - die Prozessoren haben Zugriff auf jeweils *einen eigenen Programmspeicher*, aber nur auf *einen gemeinsamen Datenspeicher*
  - pro Verarbeitungsschritt ist die Abarbeitung *jeweils eines Befehls durch alle Prozessoren über dem selben Datenelement* möglich
  - Prinzip ist umstritten, praktisch kaum sinnvoll realisierbar!

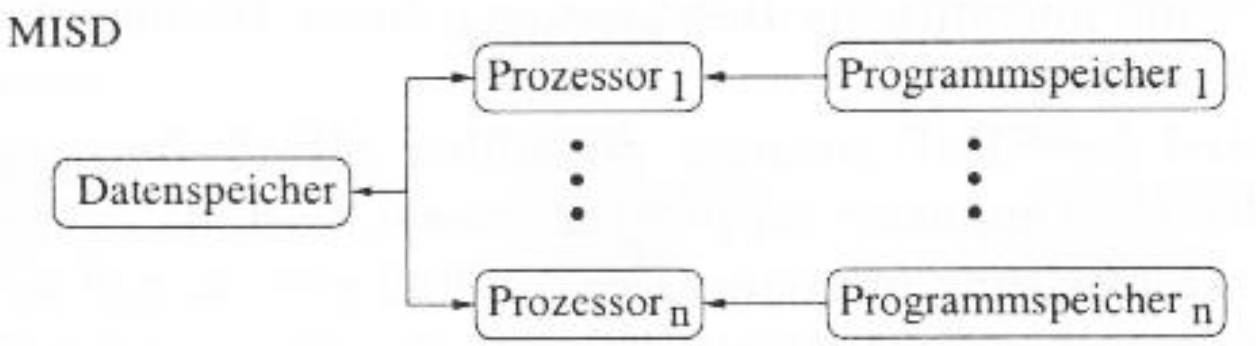


Bild-Quelle: Rauber/Rünger: Parallele und verteilte Programmierung. Springer, 2000

5.2 Klassifikation nach FLYNN (1972) (V)

- SIMD Single Instruction, Multiple Data
  - es gibt mehrere Verarbeitungseinheiten
  - diese Prozessoren haben Zugriff auf nur einen *einzigen Programmspeicher* und auf *einen (gemeinsamen oder verteilten) Datenspeicher*
  - pro Verarbeitungsschritt holt eine Steuereinheit *einen Befehl* aus dem Programmspeicher und übergibt ihn an alle Prozessoren;
  - Alle Prozessoren arbeiten also synchron parallel *denselben Befehl jeweils über einem separaten (eigenen) Datenelement* ab und schreiben das Ergebnis in den Datenspeicher
  - Anwendung z.B. bei Vektoroperationen

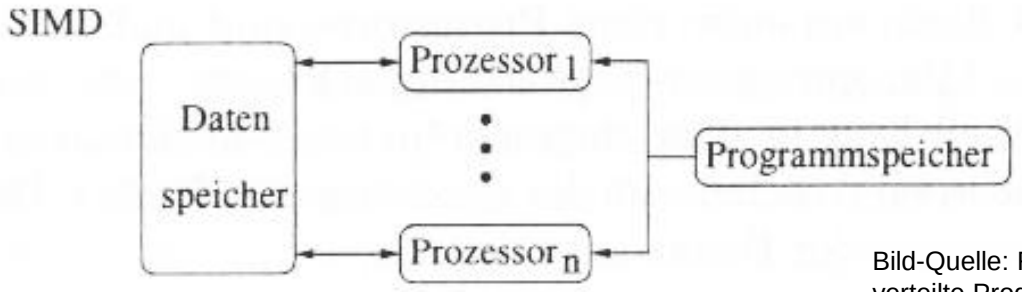


Bild-Quelle: Rauber/Rünger: Parallele und verteilte Programmierung. Springer, 2000

5.2 Klassifikation nach FLYNN (1972) (VI)

- SIMD war lange Zeit das führende Prinzip
  - einfache Programmierung (nur ein Befehlsstrom!),
  - keine Synchronisation auf Programmebene nötig
- aber Berechnungseinheiten sind (teure) Spezialprozessoren
  - Entwicklung geriet in Widerspruch zur Mikroprozessortechnik (billigere Universalprozessoren)
- Prinzip ist heute z.T. in vielen Mikroprozessoren integriert
  - vgl. Intel MMX-Erweiterung, 3DNow!, …
- Architekturbeispiele:
  - erster Parallelrechner Illiac IV (1968),
  - Connection Machine CM-1, CM-2 von Thinking Machines
  - MP-1, MP-2 von MassPar
  - DAP (Distributed Array Processor) von ICL (1981)
  - …

5.2 Klassifikation nach FLYNN (1972) (VII)

- MIMD Multiple Instruction, Multiple Data
  - es gibt mehrere Verarbeitungseinheiten
  - diese Prozessoren haben separaten Zugriff auf *jeweils einen eigenen Programmspeicher* und auf *einen (gemeinsamen oder verteilten) Datenspeicher*
  - pro Verarbeitungsschritt verarbeitet jeder Prozessor *einen Befehl aus seinem lokalen Programmspeicher über einem Datenelement aus dem Datenspeicher*
  - Die Prozessoren können asynchron zueinander arbeiten!

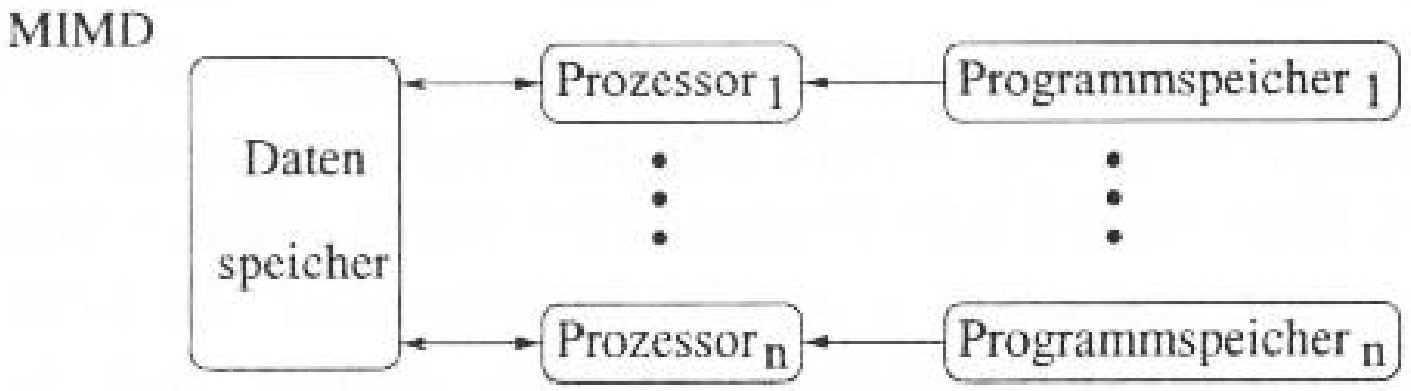


Bild-Quelle:
Rauber/Rünger:
Parallele und
verteilte
Programmierung
Springer, 2000

5.2 Klassifikation nach FLYNN (1972) (VIII)

- MIMD ist das heute am meisten verwendete Prinzip bei Parallelrechnern
- profitiert vor allem von der Mikroprozessorentwicklung, da diese als (billige) Rechenknoten in großer Zahl eingesetzt werden können
- Architekturbeispiele:
  - Intel Paragon
  - KSR-1, KSR-2 Kendall Square Research
  - Cray T3D, T3E
  - SPP-Serie von HP
  - IBM SP2
  - SGI Origin
  - Cluster,…
- wegen Vielfalt der existierenden MIMD-Varianten gibt es hierfür weitere Klassifikationskriterien (später)

5.2 Klassifikation nach FLYNN (1972) (IX)

- Multiple Single Instruction Multiple Data (MSIMD)
  - Rechnerarchitektur, die eine Zwischenstellung einnimmt (passt nicht direkt in das FLYNN-Schema)
  - „MIMD-artige parallele Zusammenschaltung“ mehrerer unabhängiger SIMD-Rechner („Multivektorrechner-System“)
  - Bsp: Earth Simulator (mehrere Jahre Anführer der TOP500!)

5.3 Klassifikation nach dem ECS

- Erlanger Classifcation System ECS (nach Händler, 1977)
- Fokus liegt auf Nebenläufigkeit und Parallelverarbeitung
- Tripel (*k*,*d*,*w*) erlaubt Gruppenzuordnung von Rechnern, wobei:
  - *k* = Anzahl der Kontrolleinheiten
  - *d* = Anzahl der Prozessoren
  - *w* = Anzahl der Bits pro Einheit
  - Bsp.: Parallelrechner mit 64 Prozessoren, jeder Prozessor hat je eine Recheneinheit für Fest- und Gleitkommazahlen, Wortlänge von 32 Bit
    → ECS = (64, 2, 32)
- für Systeme mit Nebenläufigkeit und Pipelining: zusätzl. Tripel (*k'*,*d'*,*w'*)
- es entstehen Tripelpaare (k x k´, d x d´, w x w´)
- Bsp.: Intel Paragon XP/S: 1840 Knoten, jeder Knoten = 2 x Intel i860
  Jeder Prozessor kann zwei Instruktionen gleichzeitig abarbeiten.
  Wortlänge = 64 Bit, dreistufige Pipeline
  → ECS = (1840 x 2; 1 x 2; 64 x 3) (Bsp. Nach Waldschmidt: Parallelrechner. Teubner 1995)

5.4 Klassifikation mittels Kiviat-Graph (I)

- Kiviat-Graph (nach Ferrari, 1978)
  - zur quantitativen Beschreibung von Eigenschaften, wie
    - Prozessorleistung [Bytezugriffe zum Speicher / sec.]
    - Hauptspeicherkapazität und Zugriffszeit
    - Peripheriespeicherkapazität und Zugriffszeit
    - Übertragungsrate der Verbindung zu anderen Rechnern
    - Übertragungsrate der Verbindung zu zusätzl. ext. Geräten
  - berücksichtigt alle wesentl. Hardwarekomponenten (Verarbeitung, Transport, Speicherung)
  - damit Überprüfung der „Ausgewogenheit“ hins. Leistung der einzelnen Komponenten möglich (z.B. Amdahl-Regeln:
    - HS-Kapazität [byte] ≥ Anzahl Befehle/s
    - E/A-Übertragungsrate [bit/s] ≥ Anzahl Befehle/s
  - Rolle der Software unberücksichtigt

5.4 Klassifikation mittels Kiviat-Graph (II)

- Kiviat-Graph (Beispiel)

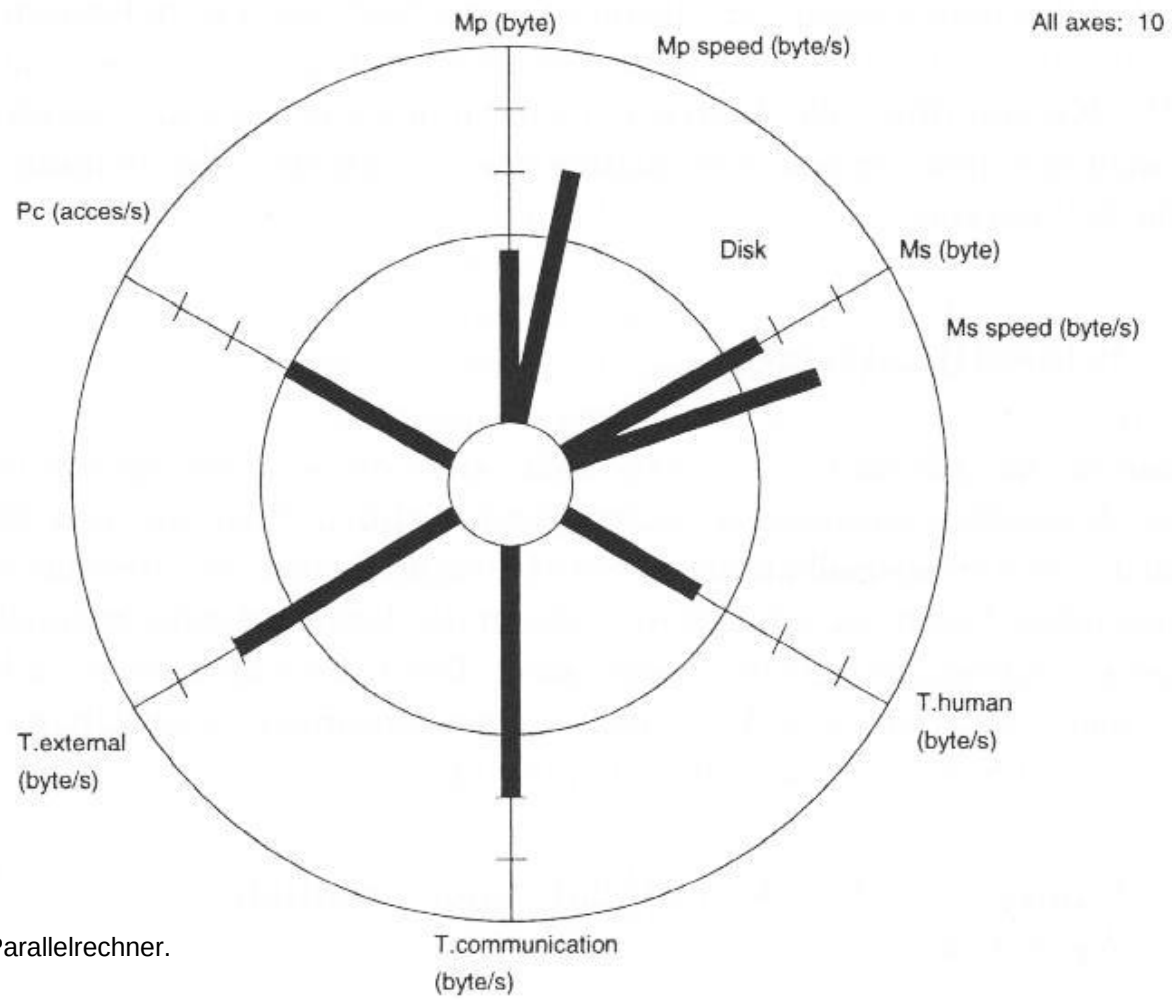


Bild-Quelle: Waldschmidt: Parallelrechner. Teubner 1995

5.5 Klassifikation nach Giloi (I)

1. Taxonomie hins. Operationsprinzipien (1980)

- qualitative Beschreibung von vielen Rechnerarchitekturen
  - Informationsstruktur
  - Steuerstruktur
- keine Quantifizierung von Merkmalen

5.5 Klassifikation nach Giloi (II)

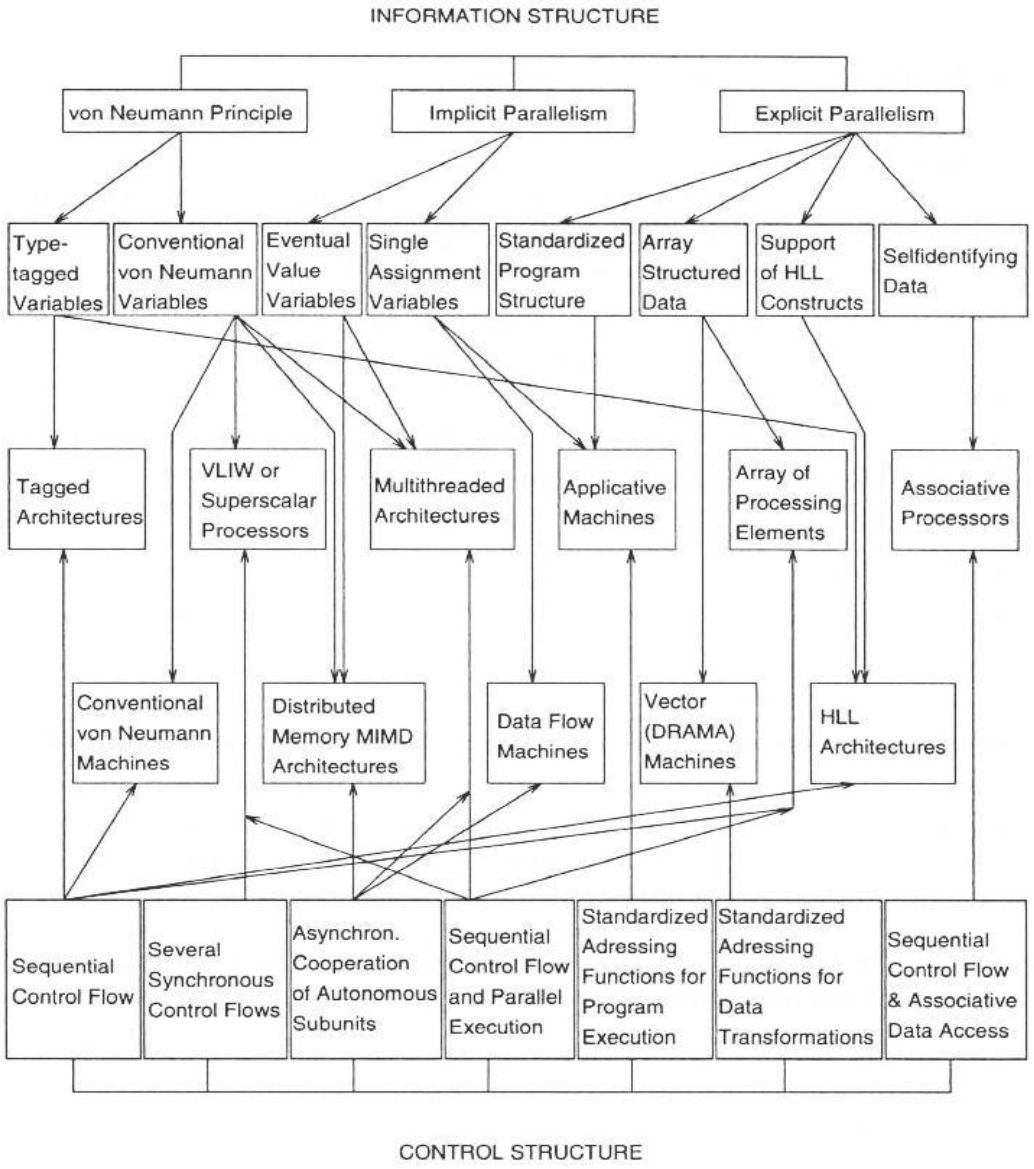


Bild-Quelle: Waldschmidt: Parallelrechner. Teubner 1995

5.5 Klassifikation nach Giloi (III)

2. Taxonomie von Parallelrechnerarchitekturen (1993)

- Versuch, wesentl. Kriterien (Hard- und Software!) zu beschreiben
- Anordnung und Auswahl der Kriterien nicht ganz eindeutig

5.5 Klassifikation nach Giloi (IV)

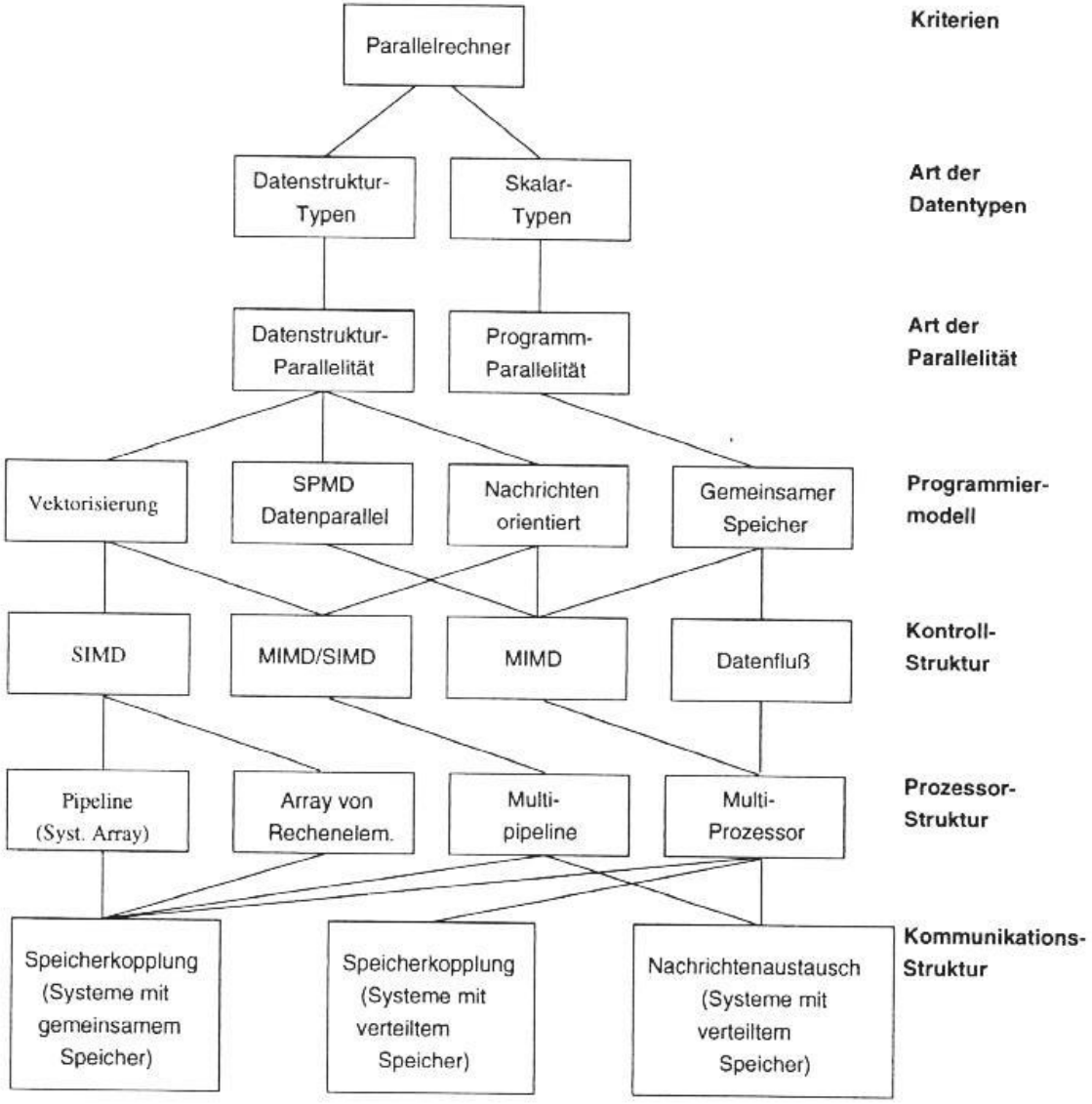


Bild-Quelle: Waldschmidt: Parallelrechner. Teubner 1995

5.6 Klassifikation nach Waldschmidt (I)

- Versuch einer integrierten Klassifikation von Hard- und Software für Parallelrechner (1995)
- Berücksichtigung quantitativer Aspekte (soweit möglich), ansonsten qualitative Bewertung
- „Entwurfsraum“ (keine baumförmige Darstellung möglich, da sich Alternativen nicht zwangsläufig ausschließen)

5.6 Klassifikation nach Waldschmidt (II)

Typen paralleler Algorithmen
Datenparallelität
Funktionsparallelität
Redundante Parallelität
Mischformen

Parallele Programmiersprachen (PS)				
Paradigma	Parallelität	Kommunikation	Sprachtyp	Kommunikationsbibliothek
Imperativ	Implizit	Gemeinsamer Speicher	Erweiterung	Architekturabhängig
Logisch, relational	Explizit	Nachrichtenorientiert	seq. PS	Architekturunabhängig
Objektorientiert,direktiv			Parallele PS	
Funktional,applikativ				

Parallele Betriebsysteme (BS)				
Ausführungsort	Betriebsart	Mehrproz/prog.betrieb	Prozeßmodell/raum	Speicherzugriff
Wirtsrechner	Einprozeßbetrieb	Timesharing	Schwergewichtig	UMA
Parallelrechner	Mehrprozeßbetrieb	Spacesharing	Leichtgewichtig	NUMA
	Mehrprogrammbetrieb		Global	NORMA
			Lokal	

Parallele Hardware: Verarbeitende Elemente		
Ausführungsmodell	Art des Parallelismus	Ebene des Parallelismus
Von Neumann	Nebenläufigkeit	Programm/Prozeß
Nicht von Neumann	Pipelining	Maschinenbefehl/Gruppen/Datenstruktur
	Kombination	Teile von Maschinenbefehlen/Datenwort

Quelle: Waldschmidt: Parallelrechner. Teubner 1995

5.6 Klassifikation nach Waldschmidt (III)

(Fortsetzung der Tabelle)

Parallele Hardware: Verbindungsstrukturen				
Kommunikationssteuerung	Topologie	Verbindungsart	Verbindungsaufbau	Arbeitsweise
Prozessorknoten	Regulär	Leitungsvermittlung	Verteilt	Synchron
Kommunikationswerk	Irregulär	Paketvermittlung	Zentral	Asynchron
Autonomer Kommunikations-	Statisch			Gemischt
Prozessor/Rechner	Dynamisch			
	Einstufig			
	Mehrstufig			
Parallele Hardware: Speicherstruktur				
Parallelisierung				
Funktionale Auftrennung der Adreßräume				
Räumliche Aufteilung auf Basis von Datenzugriffen				
Verteilung des Speichers				
Parallelisierung der Speicher-Binnenstruktur				
Speicherhierarchie				
Parallele Hardware: Peripheriestruktur				
Parallelisierung				
Geräteebene				
Architekturelemente				
Softwareebene				

Quelle: Waldschmidt: Parallelrechner. Teubner 1995