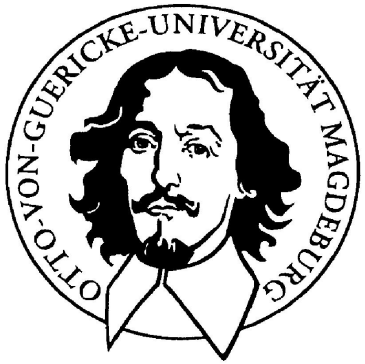
Rechnersysteme



Parallele Rechnerarchitekturen

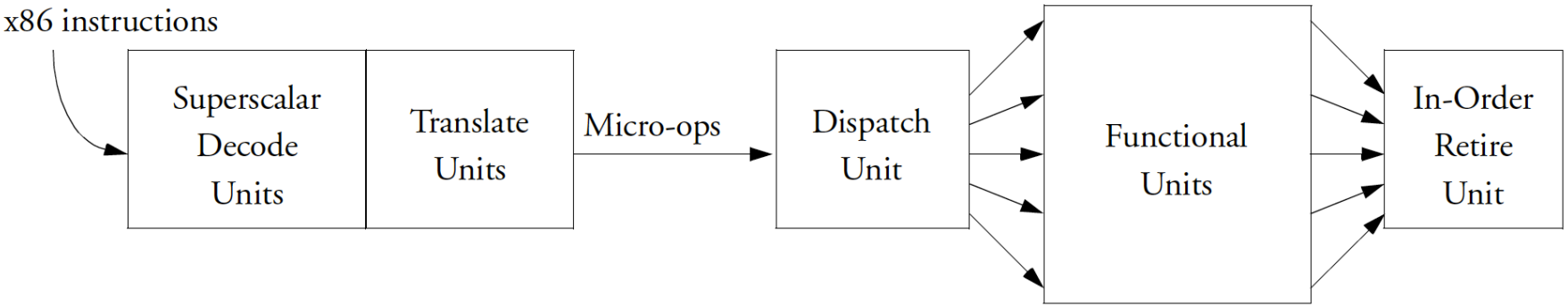
Ein kurzer Überblick

Otto-von-Guericke-Universität Magdeburg

Ebene der Parallelität	Mechanismen		Beispiele:
Instruktion	sequentiell Pipelining	parallel out-of-order VLIW	Frühe RISC. Crusoe, TriMedia
Thread	mehrere Kontexte		Xeon, Pentium 4
Recheneinheiten	Array-Rechner, Vektor-Rechner		Illiac IV, Cray-I
Funktion	Coprozessore Multiprozessoren		Grafik, FPU, MMU, Krypto, Netzwerk, Medien.
CPU	mehrere Kerne, 1 Speicher Mehrere Kerne mit priv. Speicher homogen, heterogen		SMP: Dual/Quad Core,Hochleist-ungsrechner
Kompletter Rechner	mehrere Rechner, Netzwerkverbindung		Verteilte Systeme, NFS, Cluster, Cloud, Grid

Parallelität auf Instruktionsebene

Superskalare Pipeline

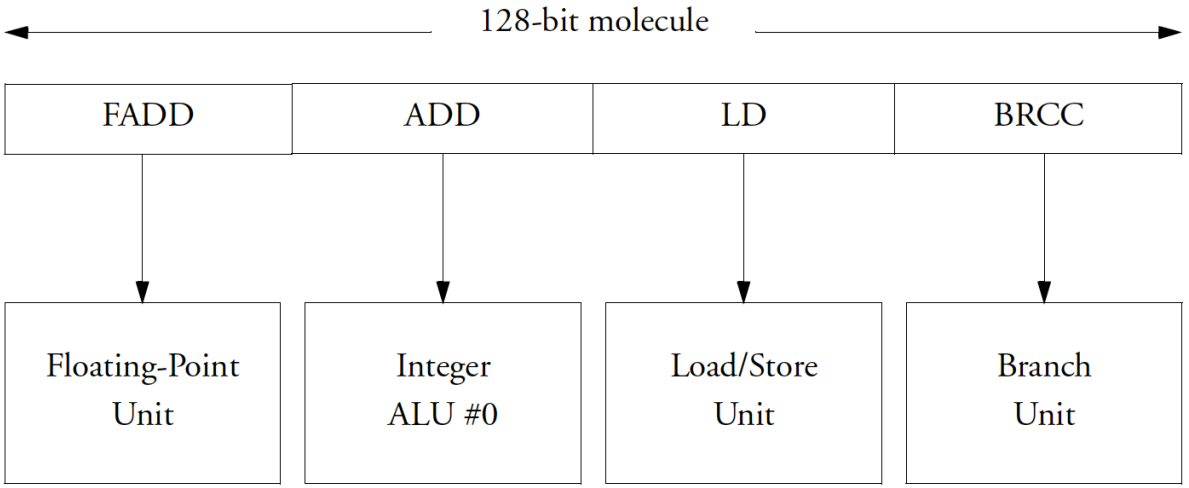


mehrere Pipelines

Parallelität auf Instruktionsebene

Very Long Instruction Word Computer

Transmeta Crusoe



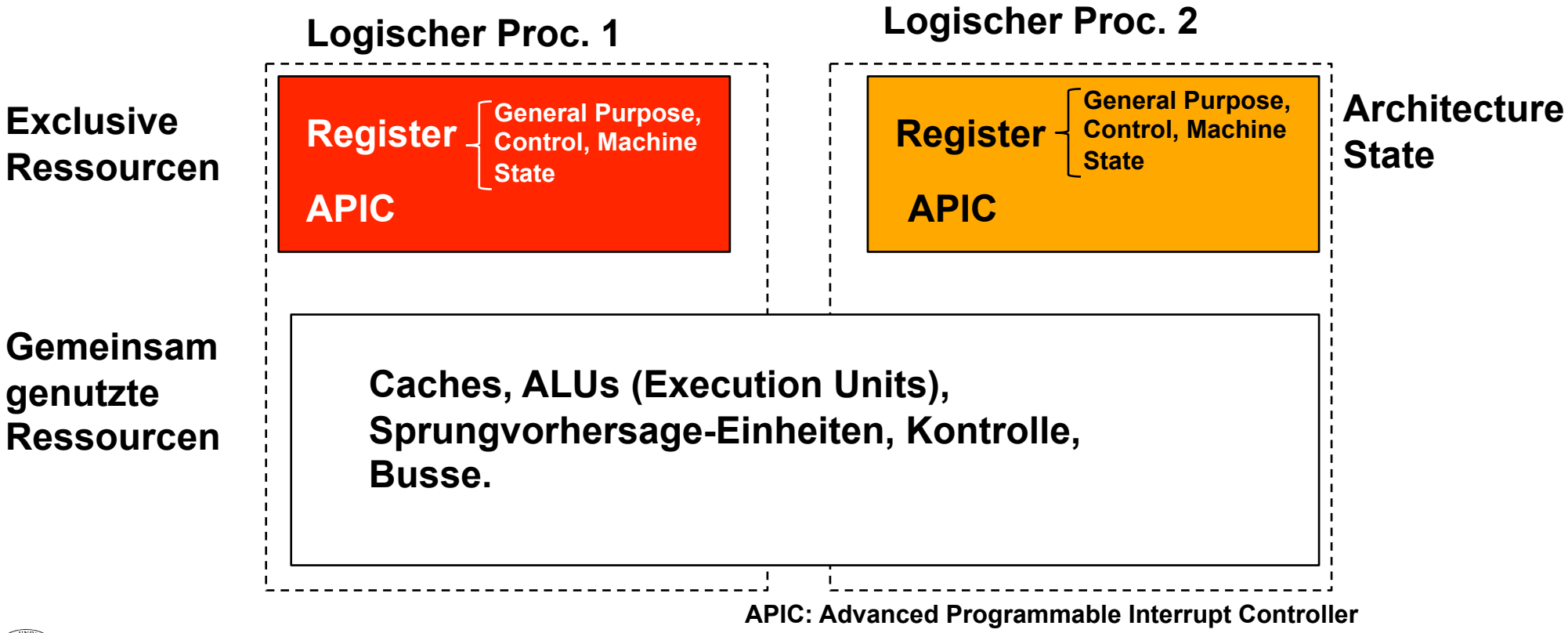
Ein Mulekül kann bis zu 4 Atome (Instruktionen) enthalten

Beispiele: Ithanium, TriMedia (Philips)

Parallelität auf Thread-Ebene

Motivation: Pipeline-Verzögerungen wie Speicherzugriffe, Cache-Fehler, falsche Sprungvorhersage, und Datenabhängigkeiten

Idee: Mehrere logische Prozessoren



Parallelität auf Thread-Ebene

Objectives:

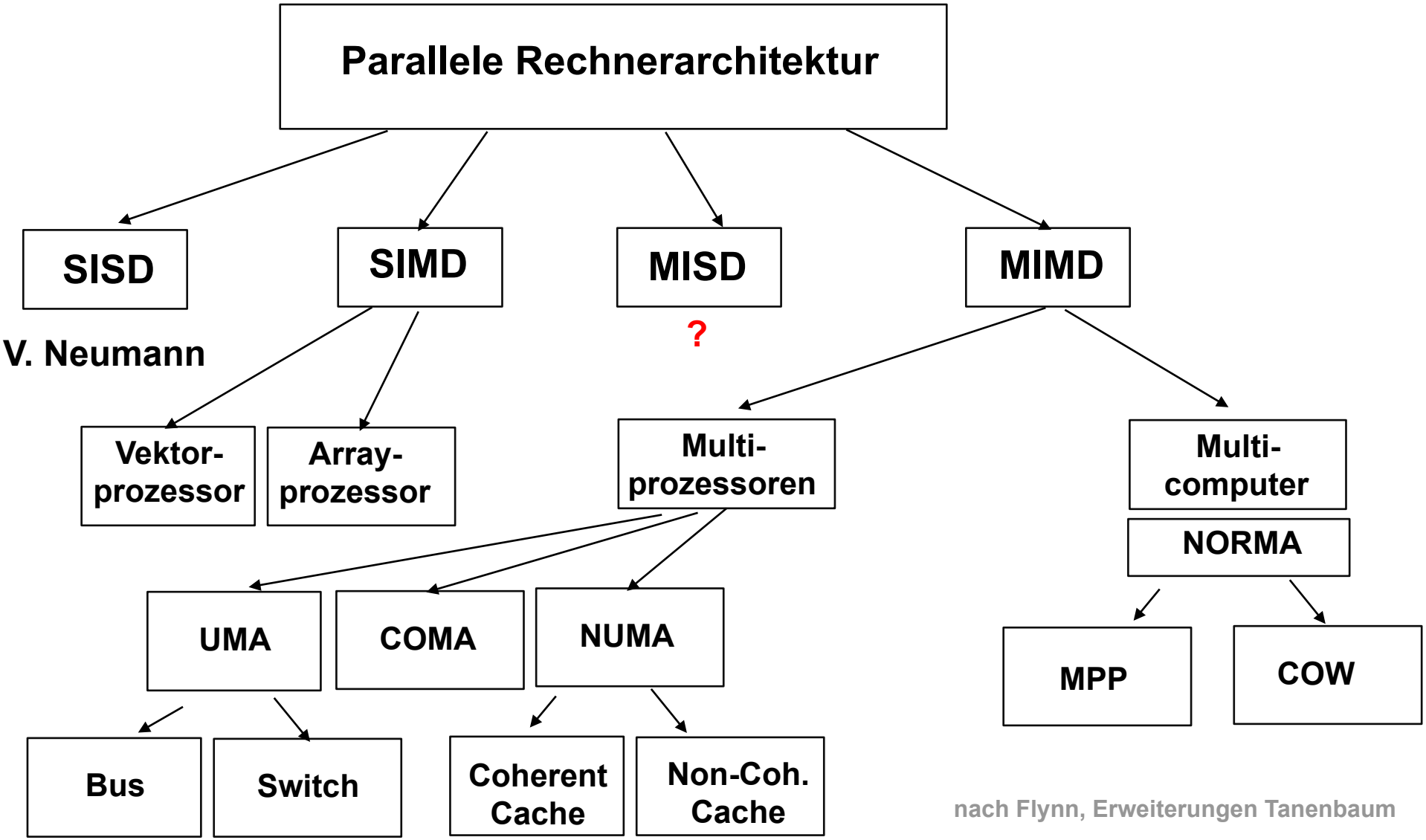
Minimize the die area cost of implementing Hyper-Threading Technology.
→ less than 5% of the total die area.

Ensure that when one logical processor is stalled the other logical processor could continue to make forward progress.
→ Independent forward progress was ensured by managing buffering queues such that no logical processor can use all the entries.

A processor running only one active software thread to run at the same speed on a processor with Hyper-Threading Technology as on a processor without this capability.
→ allocate as much resources as possible to the active logic processor (recombing partitioned resources).

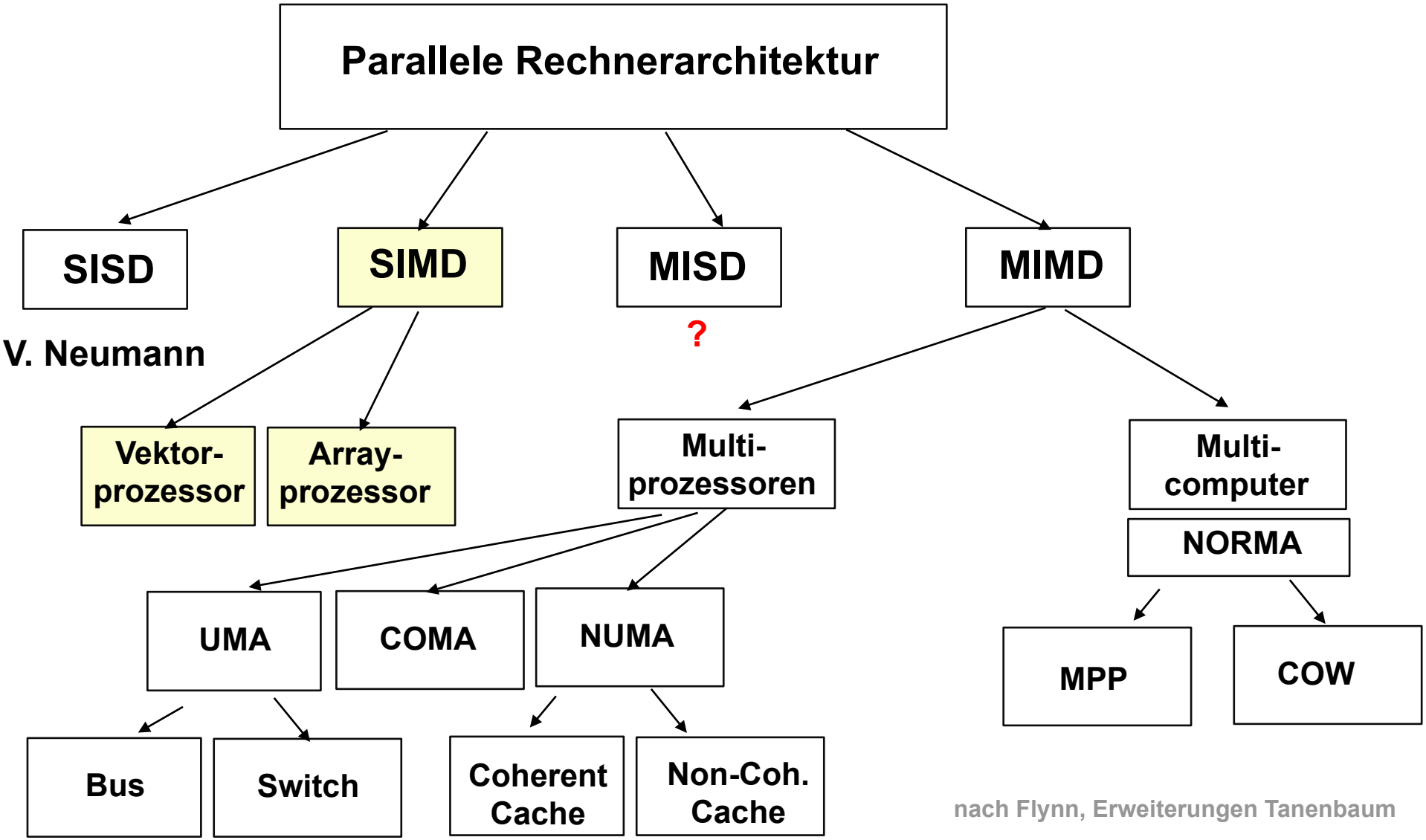
Deborah T. Marr et al. Desktop Products Group, Intel Corp.
Hyper-Threading Technology Architecture
and Microarchitecture

Taxonomie*



nach Flynn, Erweiterungen Tanenbaum

Taxonomie*



nach Flynn, Erweiterungen Tanenbaum

Vektor-Rechner

Mathematische Darstellung:

$$Y = a \cdot X + Y$$

Multipliziere Vektor X mit Skalar a und addiere zu Vektor Y

C-Programm:

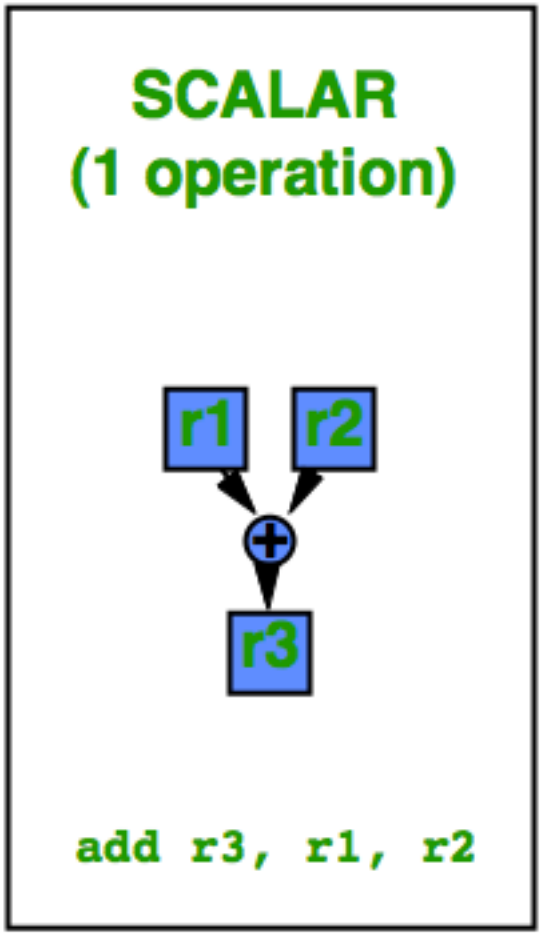
```
for (i=0; i <= 63; i++)
  Y[i] = a * X[i] + Y[i];
```

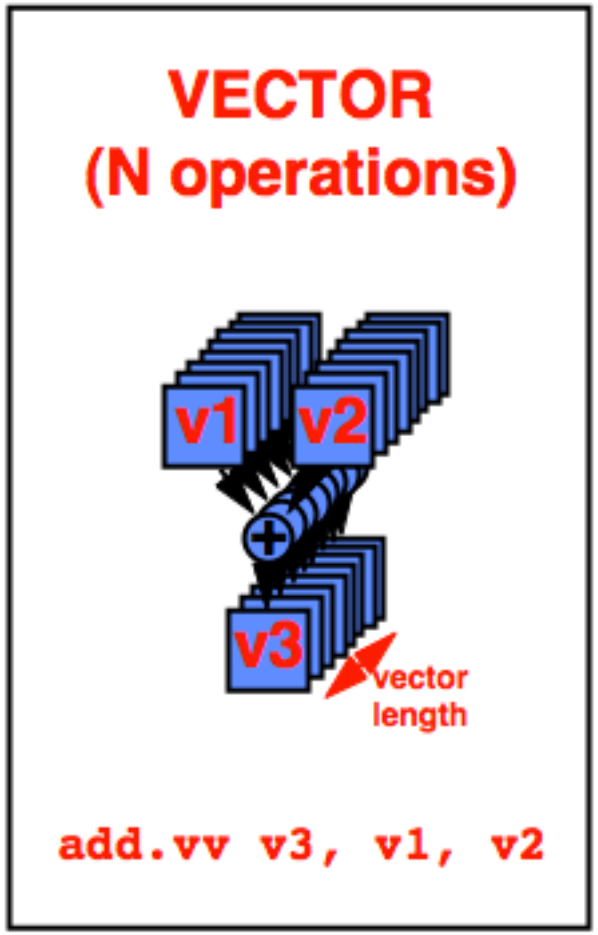
MIPS Assembler (Vektoren enthalten 64 Element a 8 Bytes (Doppelworte)):

```
        L.D     F0, a           ; Skalar a laden
        DADDIU  R4, Rx, #512    ; letzte Adresse 512/8 = 64 → R4
Loop:   L.D     F2, 0(Rx)       ; X(i) laden
        MUL.D   F2, F2, F0      ; a * X(i)
        L.D     F4, 0(Ry)       ; Y(i) laden
        ADD.D   F4, F4, F2      ; a * X(i) + Y(i)
        S.D     0(Ry), F4       ; Y(i) speichern
        DADDIU  Rx, Rx, #8      ; Index (i) von X inkrementieren
        DADDIU  Ry, Ry, #8      ; Index (i) von Y inkrementieren
        DSUBU   R20, R4, Rx     ; Rand berechnen
        BNEZ    R20, Loop       ; wenn 0, dann fertig
```

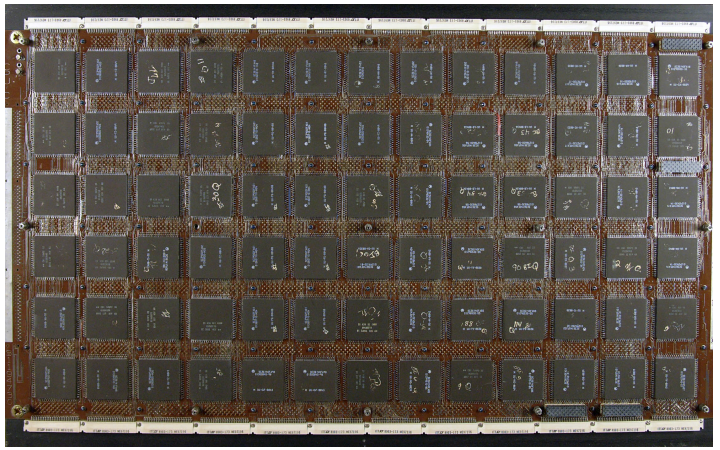
Insgesamt: 64 Durchläufe ·9 Befehle + 2 Init-Befehle = 578 Befehle

Vektor-Rechner





Lecture 6: Vector Processing
Professor David A. Patterson
Computer Science 252
Spring 1998



Gray-I Prozessor Platine (Wickipedia)

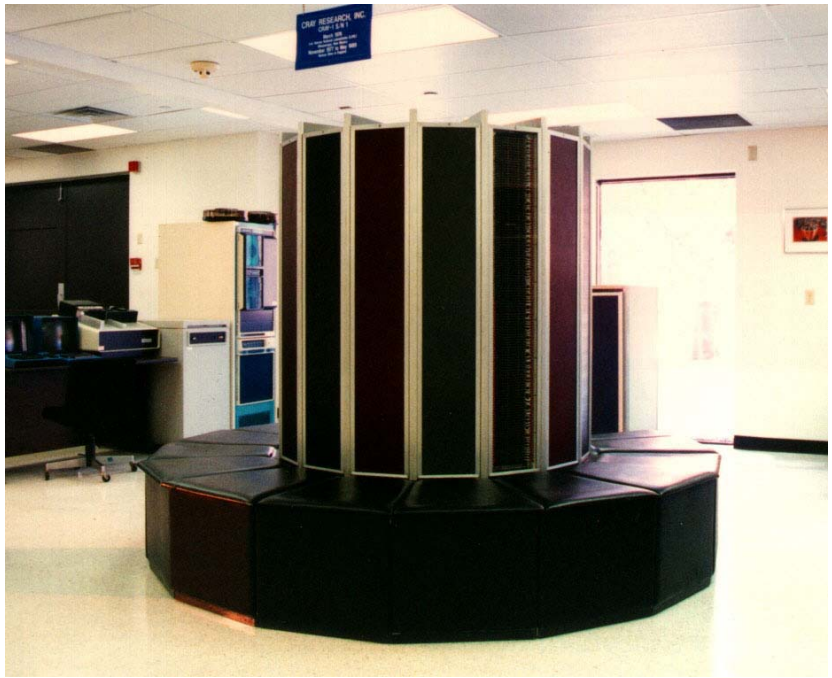
Vektor-Rechner

```
L.D   F0, a              ; Skalar a laden
LV    V1, Rx             ; Vector X laden
MULVS.D V2, V1, F0       ; Vector-Skalar Multiplikation
LV V3, Ry                ; Vector Y laden
ADDV.D   V4, V2, V3      ; Vektor Addition
SV Ry, V4                ; Resultat speichern
```

Insgesamt: 6 Befehle

Gewinn: 578/6= 96 !

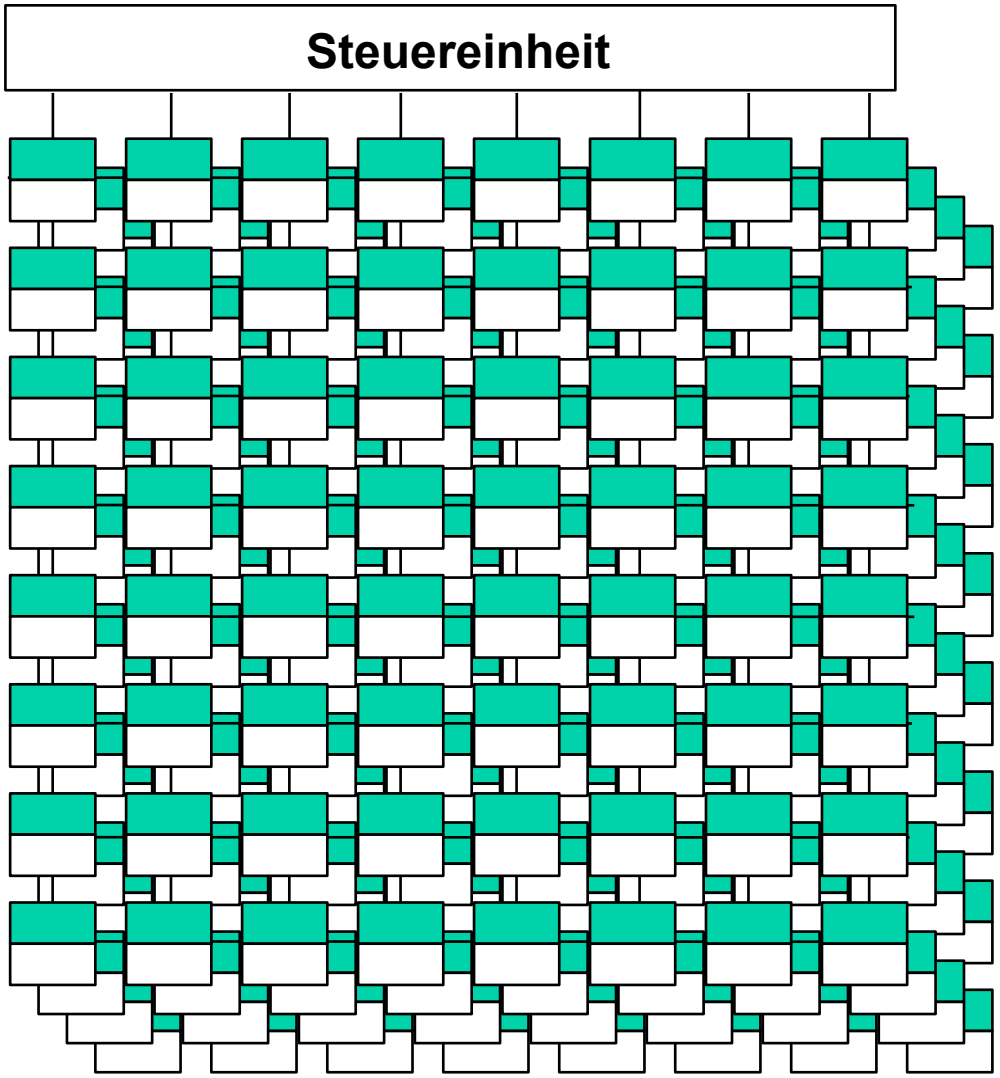
Ausserdem:
Kein Aufwand für Schleifenabfrage und Sprünge → keine Pipelinehindernisse!



Heisser Ofen: Gray-1

Feldrechner (Array-Rechner)

ILLIAC IV



Feld-Rechner

ILLIAC IV

Matrix-Multiplikation

Based on slides by: Jonathan Break, http://www.cs.ucf.edu/courses/cot4810/fall04/presentations/Systolic_Arrays.ppt

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

```
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];
```

Beispiel:

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

$$\begin{bmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{bmatrix} * \begin{bmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{bmatrix} = \begin{bmatrix} P1 & P2 & P3 \\ P4 & P5 & P6 \\ P7 & P8 & P9 \end{bmatrix}$$

$P1 = a_{11} * b_{11} + a_{12} * b_{21} + a_{13} * b_{31} \rightarrow 3*3 + 4*2 + 2*3 = 23$

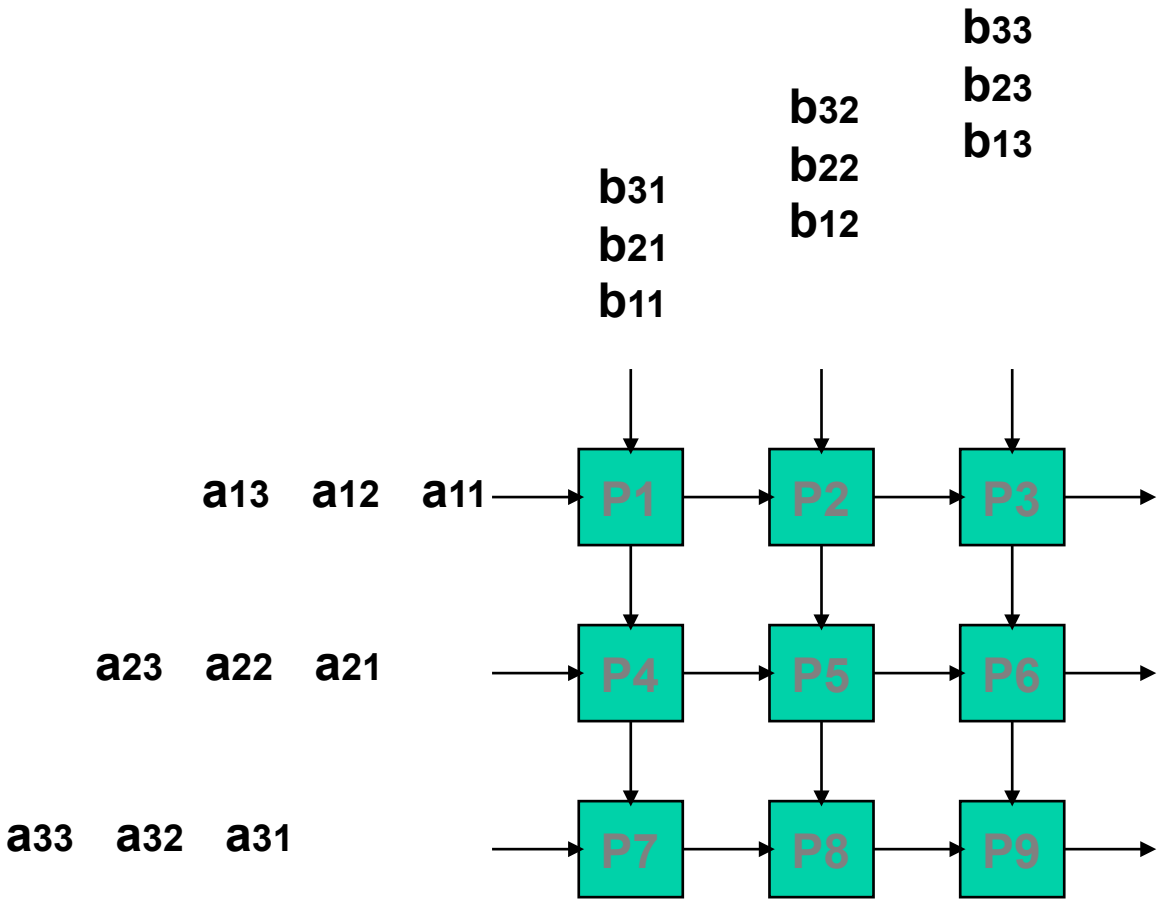
$P2 = a_{11} * b_{12} + a_{12} * b_{22} + a_{13} * b_{32} \rightarrow 3*4 + 4*5 + 2*2 = 36$

$P3 = a_{11} * b_{13} + a_{12} * b_{23} + a_{13} * b_{33} \rightarrow 3*2 + 4*3 + 2*5 = 38$

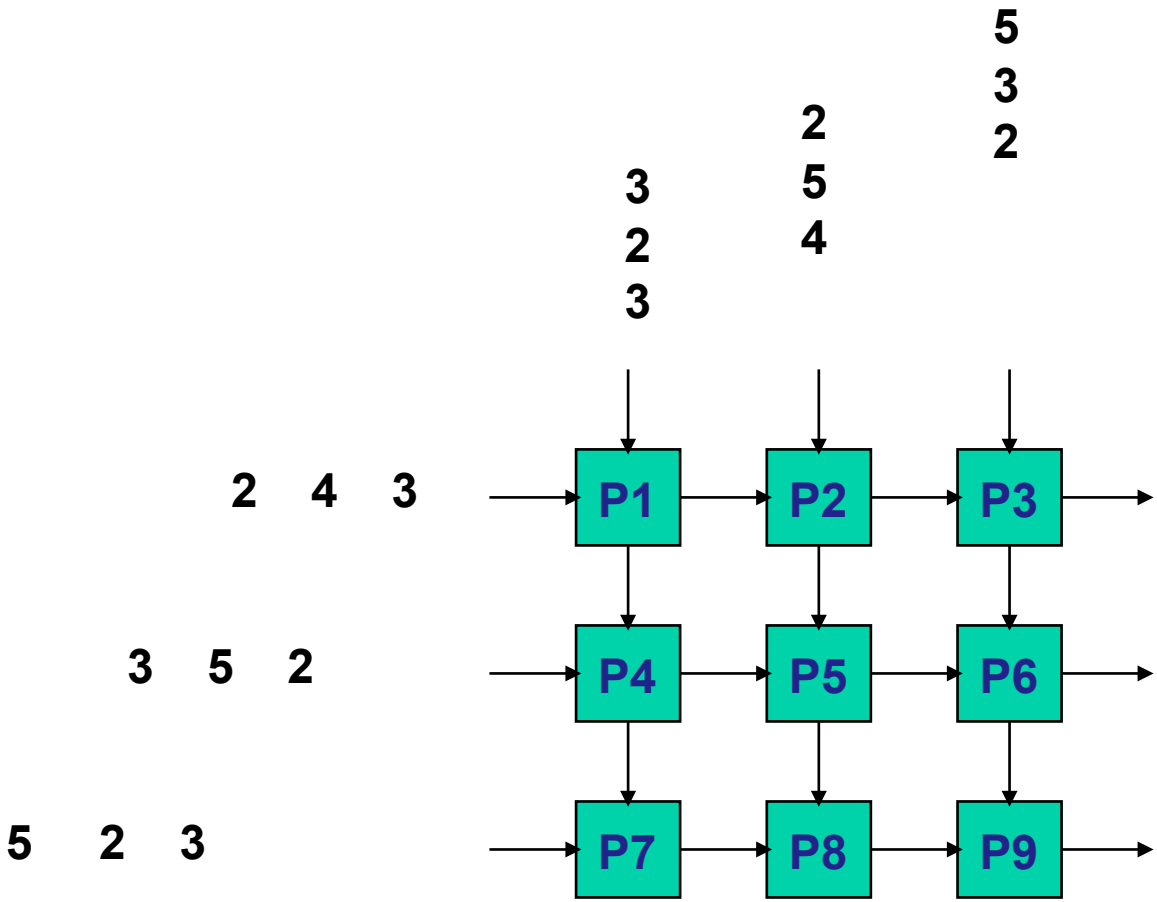
.....

.....

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

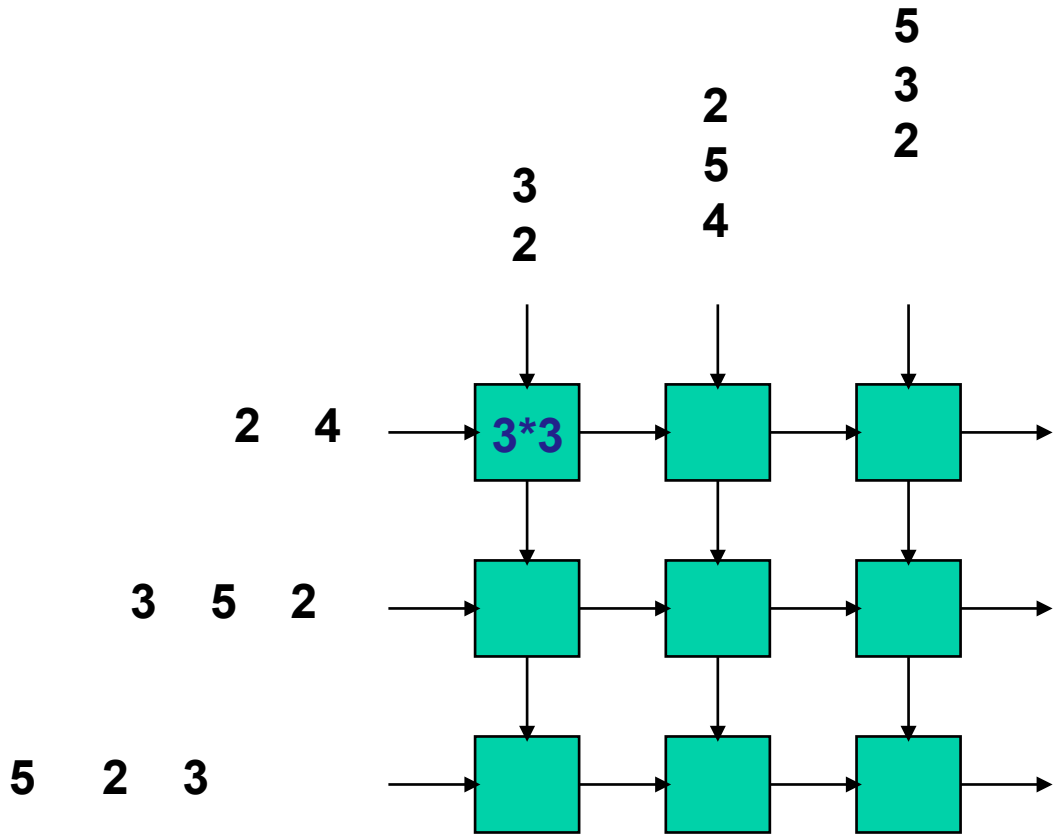


Bei jedem Takt der Systemuhr werden Daten an jedes Prozessorelement aus unterschiedlichen Richtungen angelegt. Sie werden miteinander multipliziert und das Ergebnis wird in einem Register gespeichert.



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

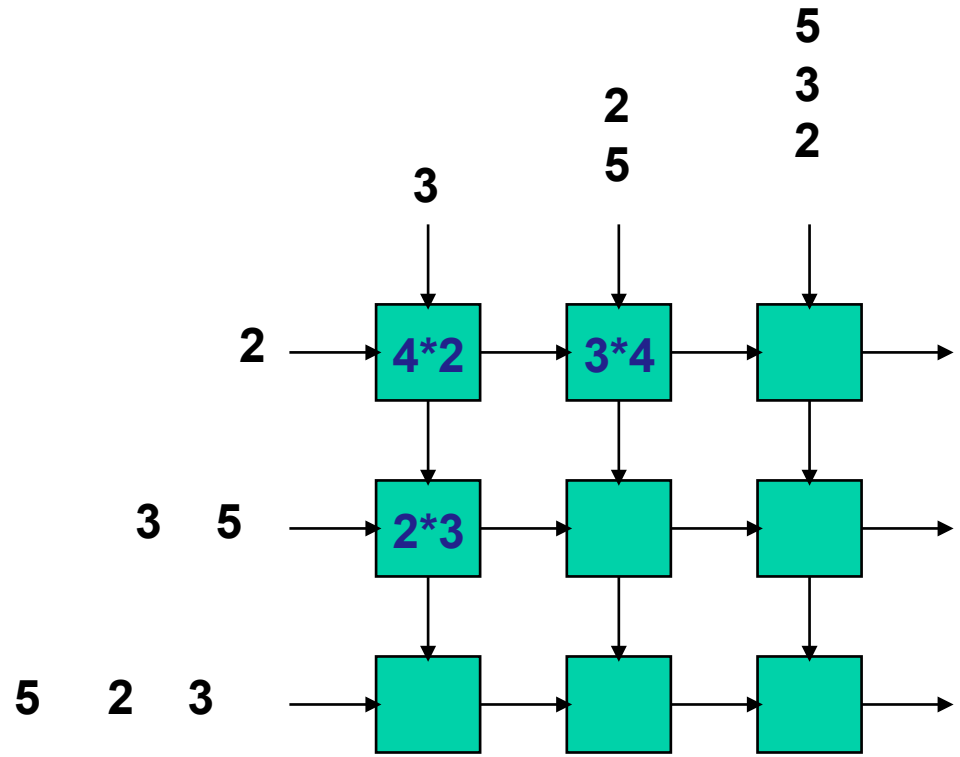
$$\begin{bmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{bmatrix} * \begin{bmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{bmatrix} = \begin{bmatrix} 23 & 36 & 28 \\ 25 & 39 & 34 \\ 28 & 32 & 37 \end{bmatrix}$$



Takt 1

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

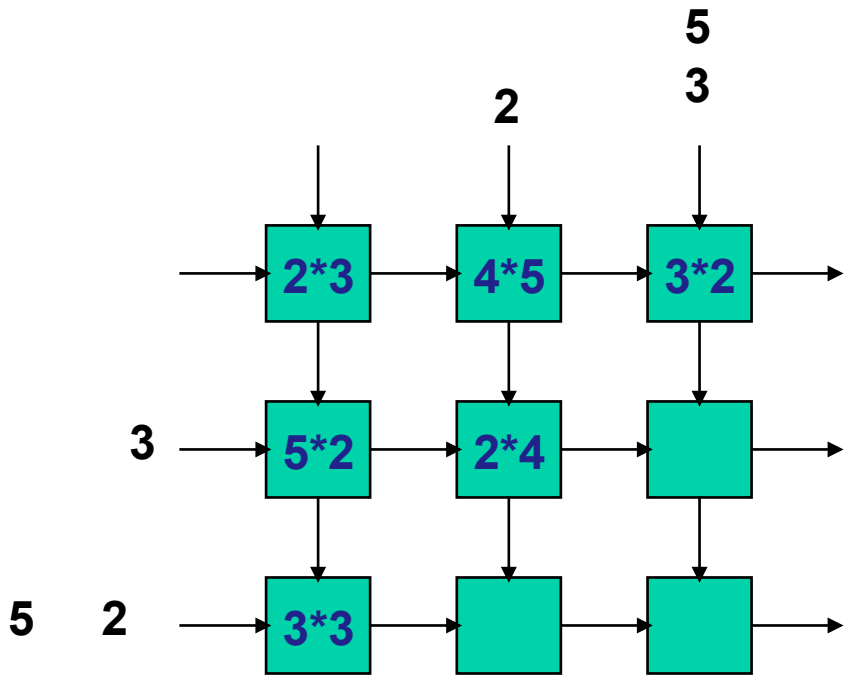
$$\begin{bmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{bmatrix} * \begin{bmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{bmatrix} = \begin{bmatrix} 23 & 36 & 28 \\ 25 & 39 & 34 \\ 28 & 32 & 37 \end{bmatrix}$$



Takt 2

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

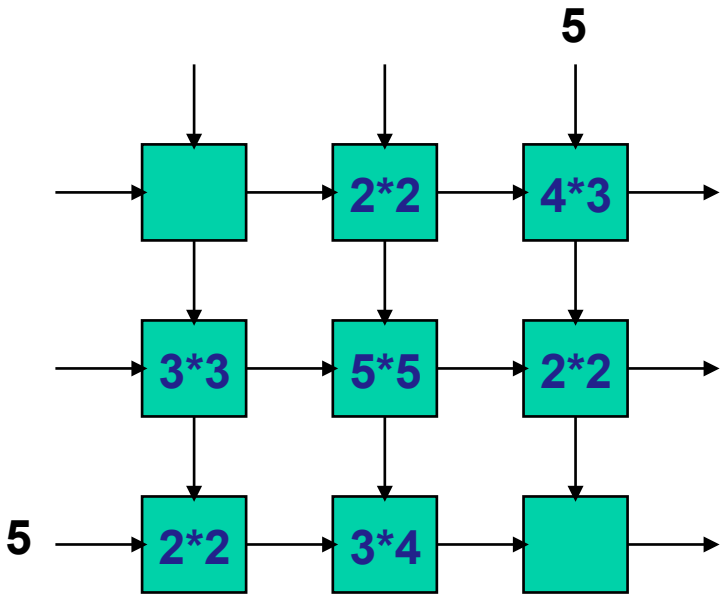
$$\begin{bmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{bmatrix} * \begin{bmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{bmatrix} = \begin{bmatrix} 23 & 36 & 28 \\ 25 & 39 & 34 \\ 28 & 32 & 37 \end{bmatrix}$$



Takt 3

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

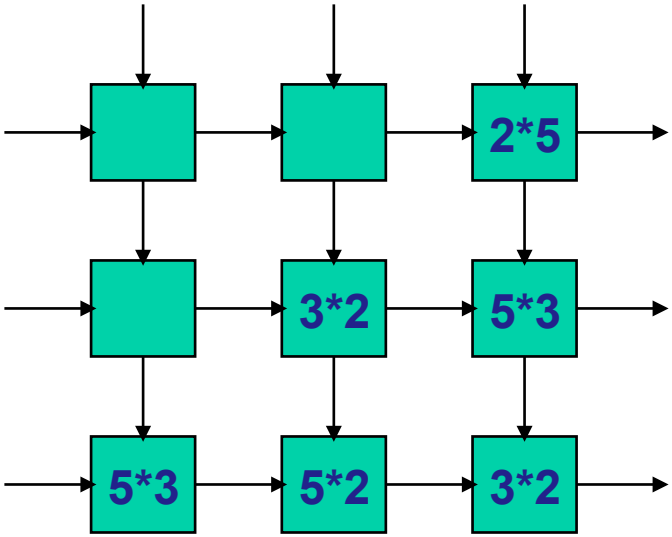
$$\begin{bmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{bmatrix} * \begin{bmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{bmatrix} = \begin{bmatrix} 23 & 36 & 28 \\ 25 & 39 & 34 \\ 28 & 32 & 37 \end{bmatrix}$$



Takt 4

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

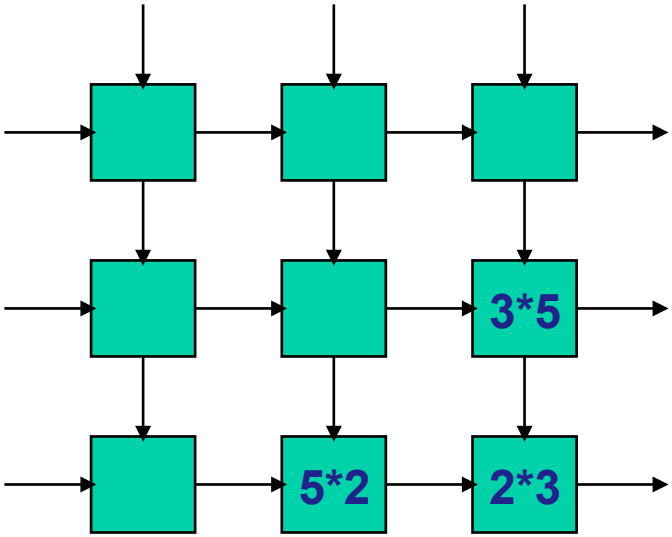
$$\begin{bmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{bmatrix} * \begin{bmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{bmatrix} = \begin{bmatrix} 23 & 36 & 28 \\ 25 & 39 & 34 \\ 28 & 32 & 37 \end{bmatrix}$$



Takt 5

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

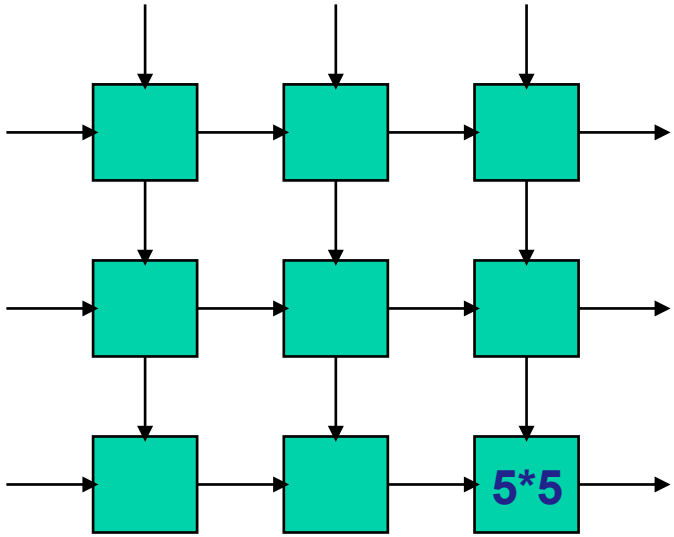
$$\begin{bmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{bmatrix} * \begin{bmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{bmatrix} = \begin{bmatrix} 23 & 36 & 28 \\ 25 & 39 & 34 \\ 28 & 32 & 37 \end{bmatrix}$$



Takt 6

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

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

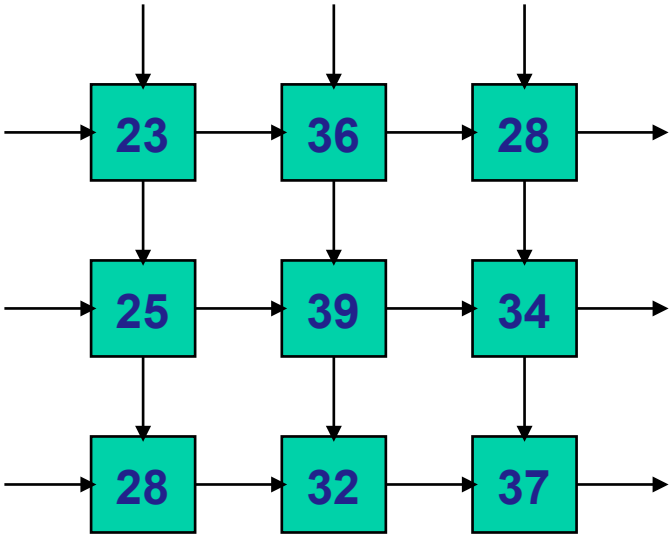


Takt 7

P1	P2	P3	P4	P5	P6	P7	P8	P9
23	36	28	25	39	34	28	32	37

Same answer! In 2n + 1 time

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



Takt 8

P1	P2	P3	P4	P5	P6	P7	P8	P9
23	36	28	25	39	34	28	32	37

ILLIAC IV

The move[1] slowed development, and the machine was not completed until 1972. By this time the original $8 million estimated from the first design in 1966 had risen to $31 million, while the performance had dropped even further, from 1 GFLOPS to 250 MFLOPS to perhaps 100 MFLOPS with peaks of 150.

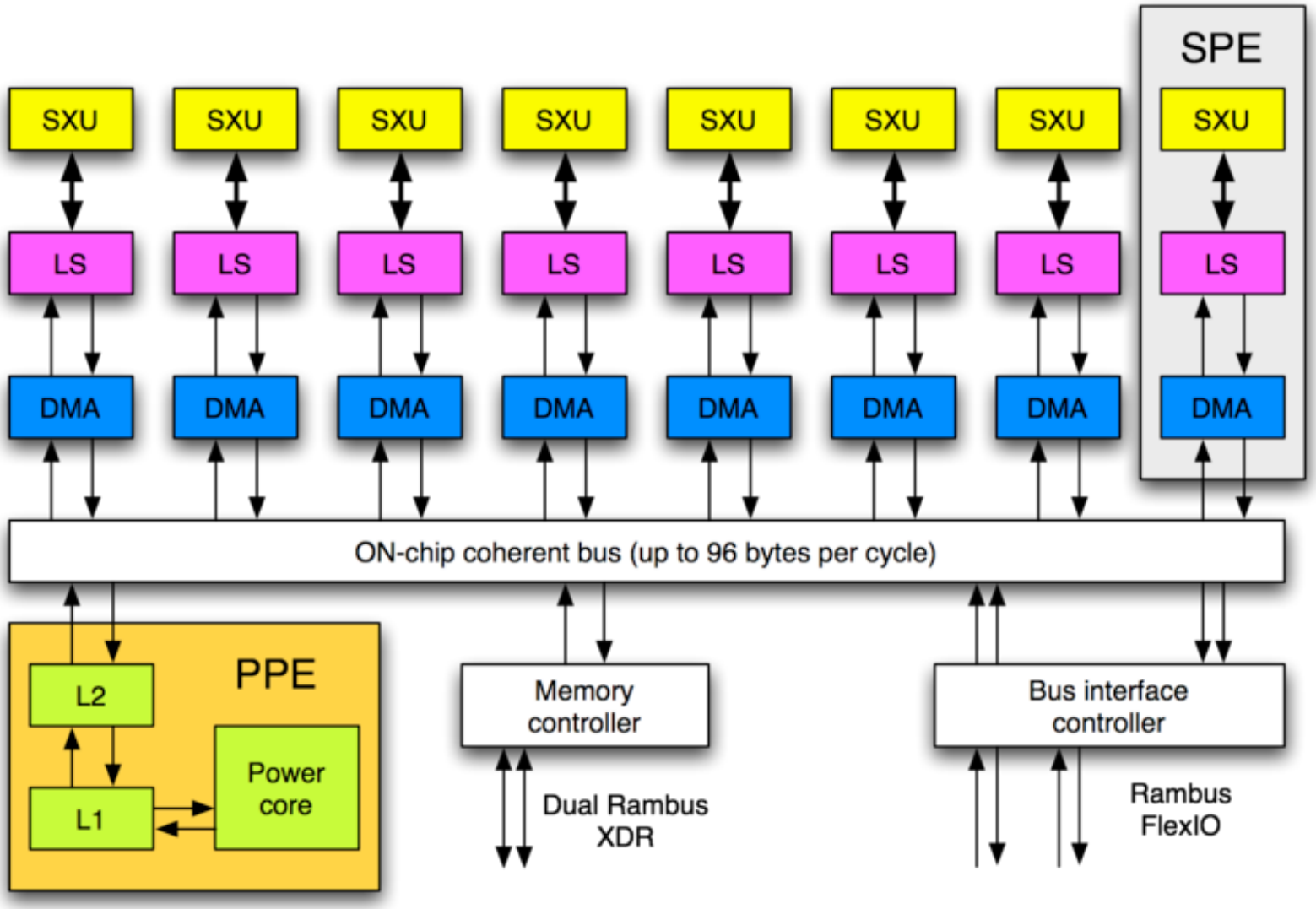
When the ILLIAC was finally turned on in 1972 it was found to be barely operable, failing continually. Efforts to correct the reliability allowed it to run its first complete program in 1974, and go into full operation in 1975. Even "full operation" was somewhat limited; the machine was operated only Monday to Friday and had up to 40 hours of planned maintenance a week.

The first full application was run on the machine in 1976, the same year the Cray-1 was released with roughly the same performance.

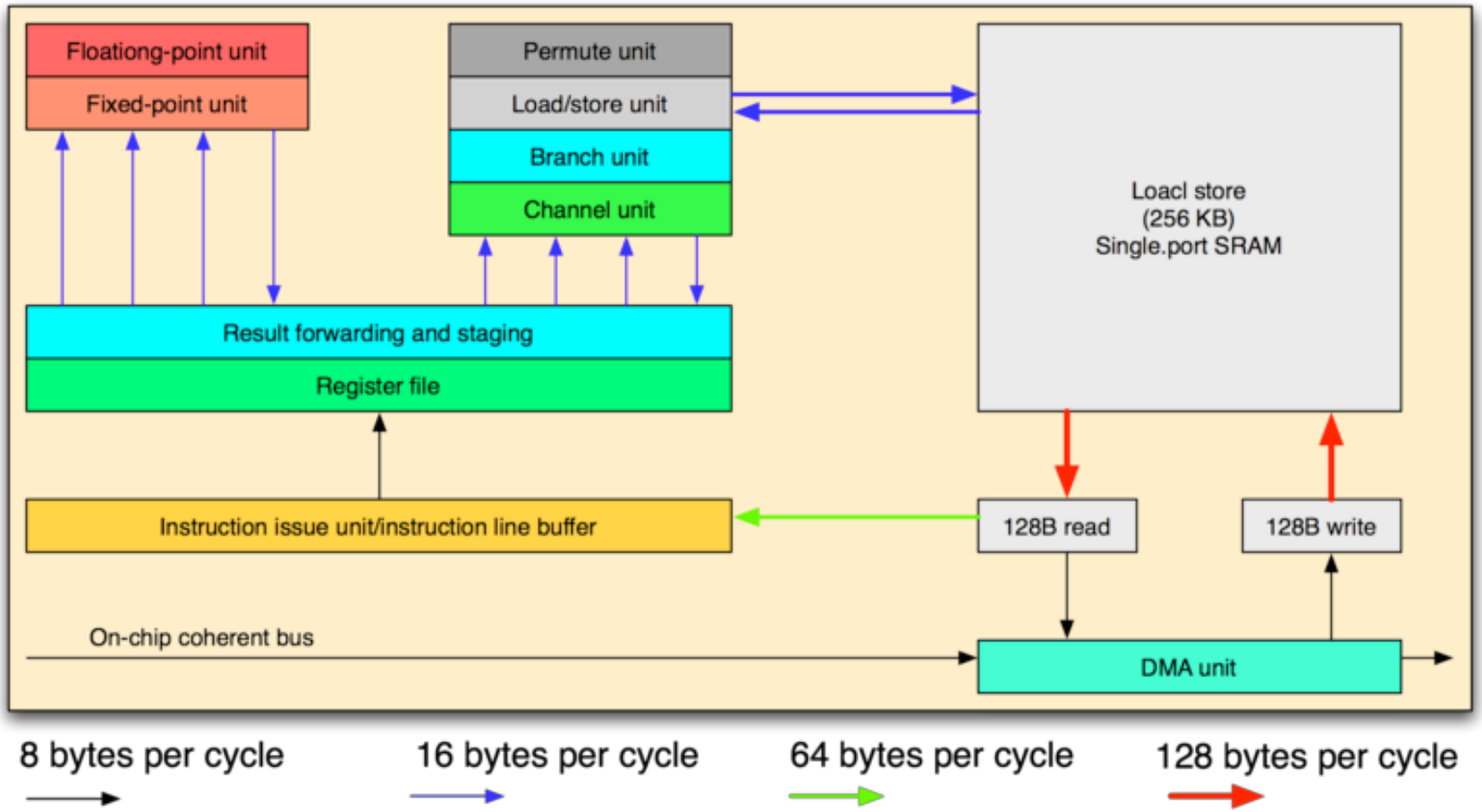
http://en.wikipedia.org/wiki/ILLIAC_IV

[1] an einen sichereren Platz als die Universität

Die CELL Architektur



CELL "Synergistic Processing Element (SPE)

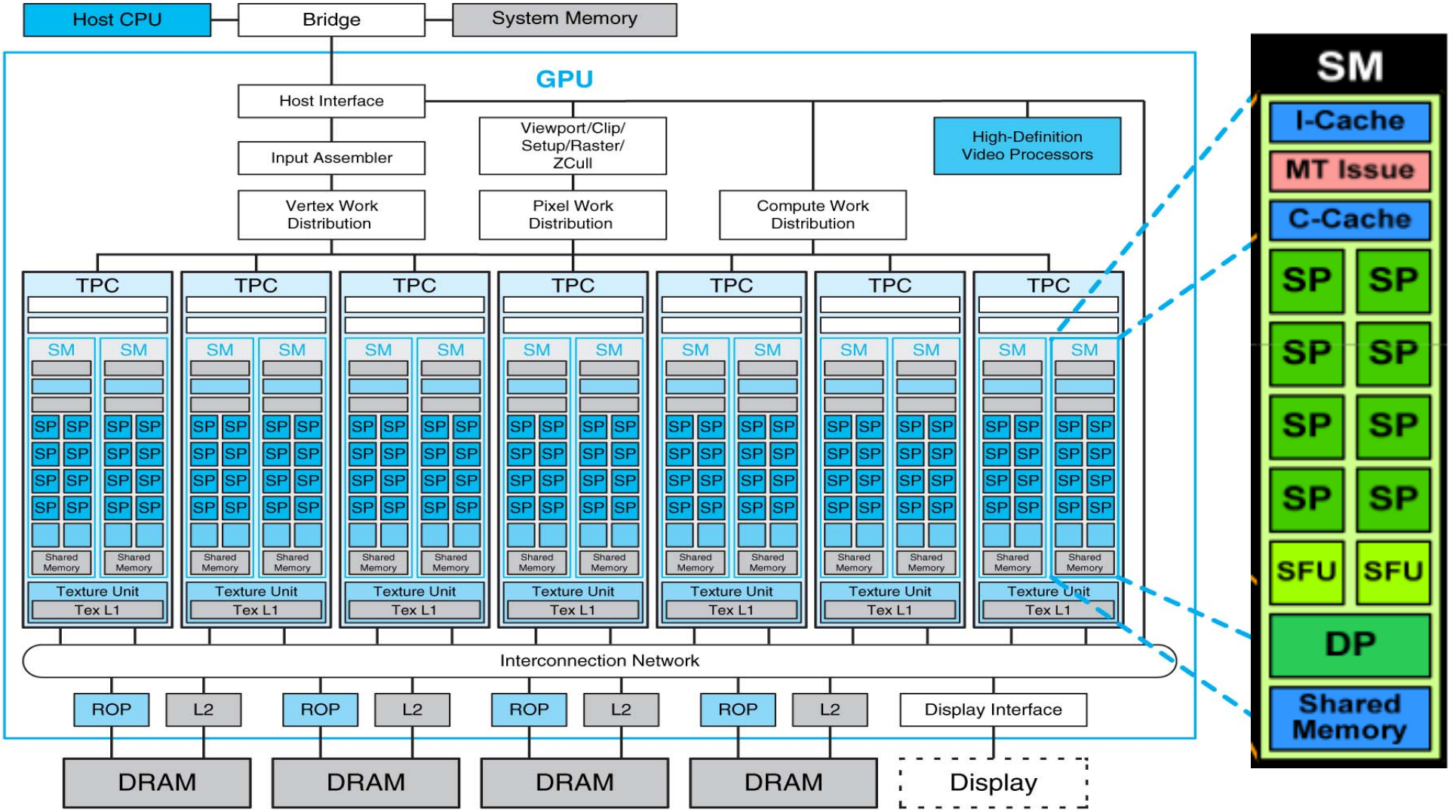


CELL im Vergleich mit GPUs

Hersteller	Prozessor	Anzahl der Kerne	Anzahl der SIMD-Einheiten	FMUL +FADD Performance in Flop/s	Takt in GHz	Spitzenleistung in GFlop/s	BLAS/ SGEMM in GFlop/s	Verlustleistung in Watt	Ausführung
IBM	Cell BE[1]	8	4	2	3,2	204,8	201	80	Prozessor
Nvidia	8800Ultra (G80)	128	1	2	1,512	387,1	[2]	>170	Karte
Nvidia	8800GTX (G80)	128	1	2	1,350	345,6	105[3]	120–170	Karte
Nvidia	GT200b	240	1	n/a	1,476	1062,7	[2]	180–240	Karte
ATI	HD2900 XT (R600)	320	5	2	0,742	474,9	[2]	150–200	Karte
ATI	1900XTX (R580)	48	4	2	0,65	249,6	120	130–170[4]	Karte
ATI	RV770	800	5	n/a	0,750	1200	[2]	80–160	Karte

http://de.wikipedia.org/wiki/Cell_Prozessor

GPU Architecture (TESLA)



Computer organization and design: the hardware/software interface von David A. Patterson,John L. Hennessy

GPU Architecture (TESLA)

SM has 8 SP (Streaming Processors)
Thread Processors:

- **IEEE 754 32-bit floating point**
- **32-bit and 64-bit integer**
- **16K 32-bit registers**
- **SM has 2 SFU Special Function Units**
- **SM has DP Double Precision Unit**
  - **IEEE 754 64-bit floating point**
- **Fused multiply-add•Multithreaded Instruction Unit**
- **1024 threads, hardware multithreaded**
- **32 SIMT warps of 32 threads**
- **Independent thread execution**
  - **Hardware thread scheduling**
- **16KB Shared Memory**
- **Concurrent threads share data**
- **Low latency load/store**

Typ	Rechenleistung	Speicherbus-Datenrate
ATI Radeon HD 5870	2720,0 GFlops	153,6 GByte/s
NVIDIA GeForce GTX 295	1788,5 GFlops	223,8 GByte/s Intel
Core i7-970	94,0 GFlops	4,8 GByte/s
Intel Pentium 4 mit SSE3, 3,6 GHz	14,4 GFlops	5,0 GByte/s

CUDA: Compute Unified Device Architecture, erlaubt die Programmierung von GPUs als universelle "Number Cruncher".

In the year 2000 the world's fastest supercomputer, a cluster of linked machines costing $110 million, operated at slightly more than seven teraflops.

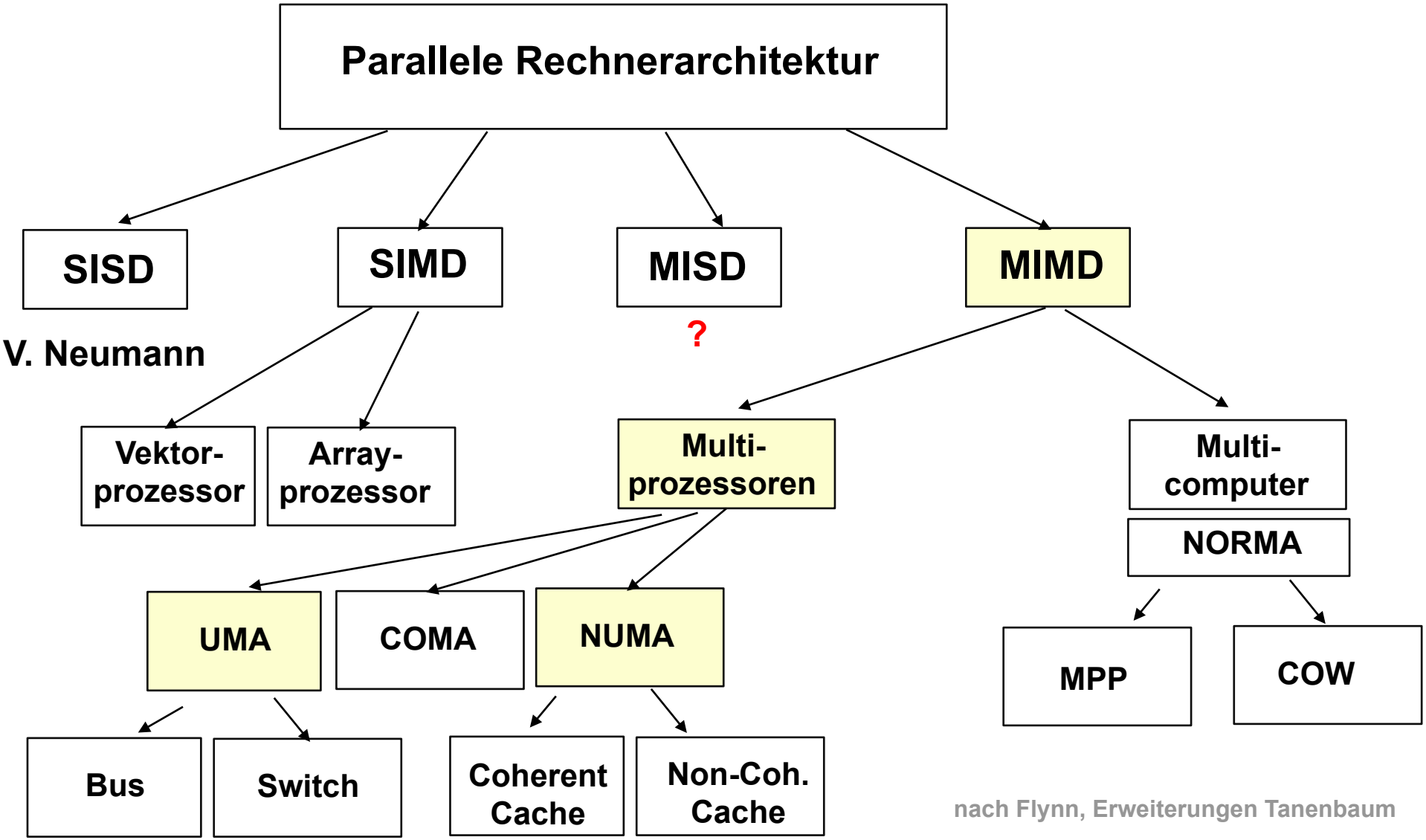
http://www.gtri.gatech.edu/casestudy/Teraflop-Troubles-Power-Graphics-Processing-Units-GPUs-Password-Security-System

GPU is a brilliant tool for password cracking
Not just for pretty pictures

Read more:.....

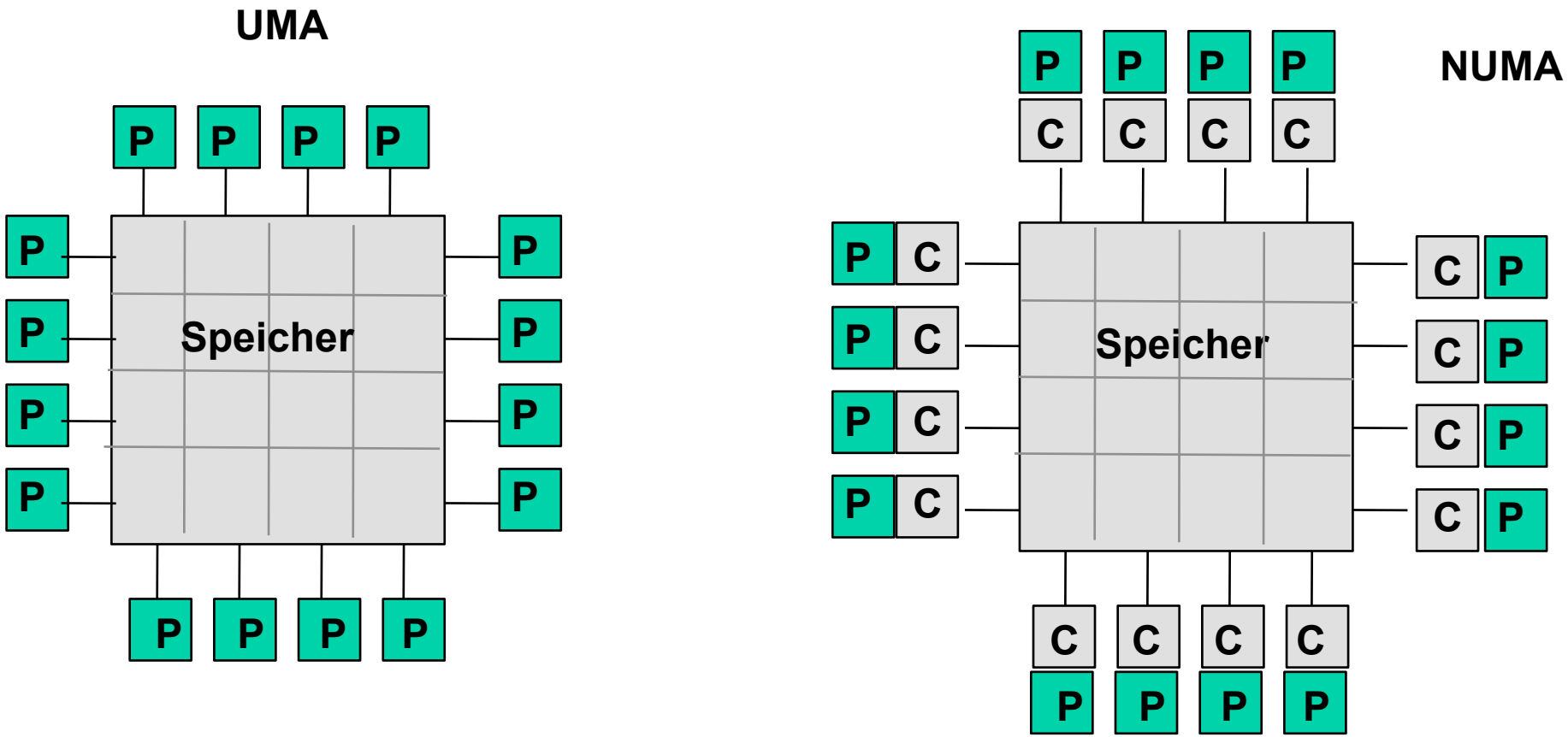
Dim\Time	CUDA	CPU
64x64	0.417465 ms	18.0876 ms
128x128	0.41691 ms	18.3007 ms
256x256	2.146367 ms	145.6302 ms
512x512	8.093004 ms	1494.7275 ms
768x768	25.97624 ms	4866.3246 ms
1024x1024	52.42811 ms	66097.1688 ms
2048x2048	407.648 ms	Didn't finish
4096x4096	3.1 seconds	Didn't finish

Taxonomie*



nach Flynn, Erweiterungen Tanenbaum

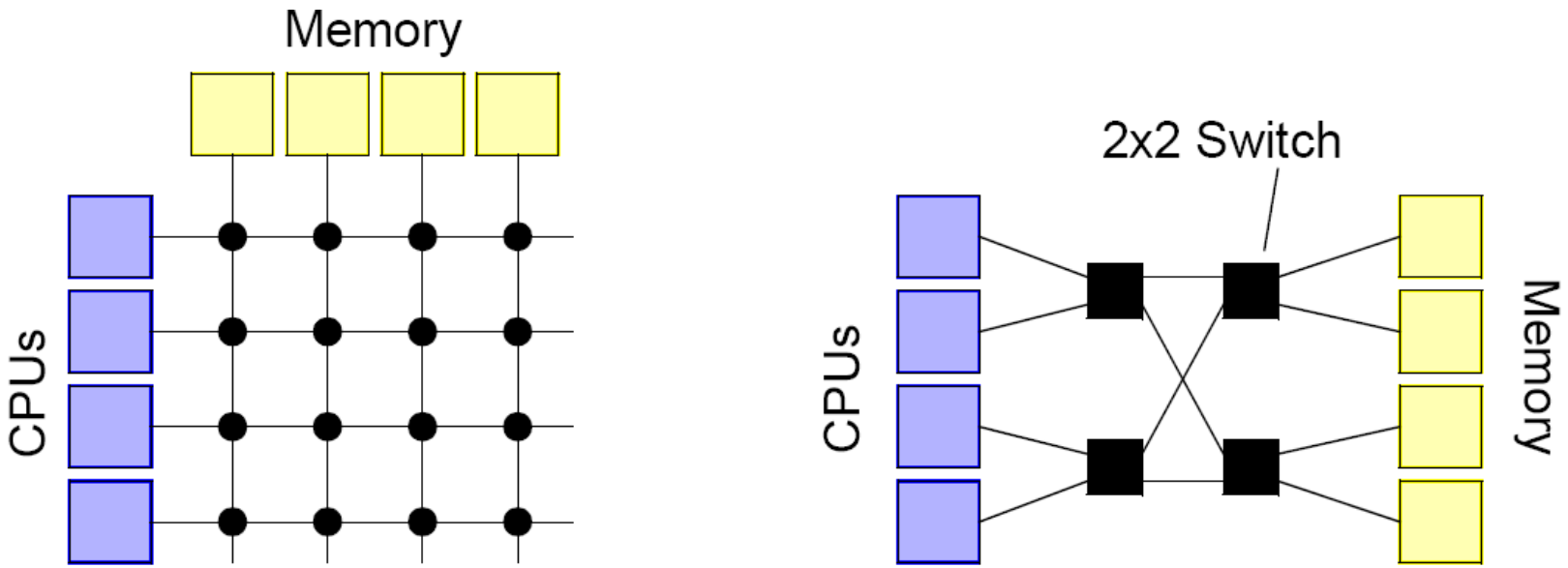
Multicomputer mit gemeinsamem Speicher



Wie sieht das Verbindungsnetzwerk zwischen Prozessoren und Speicher aus?

UMA Multi-Processor Systeme

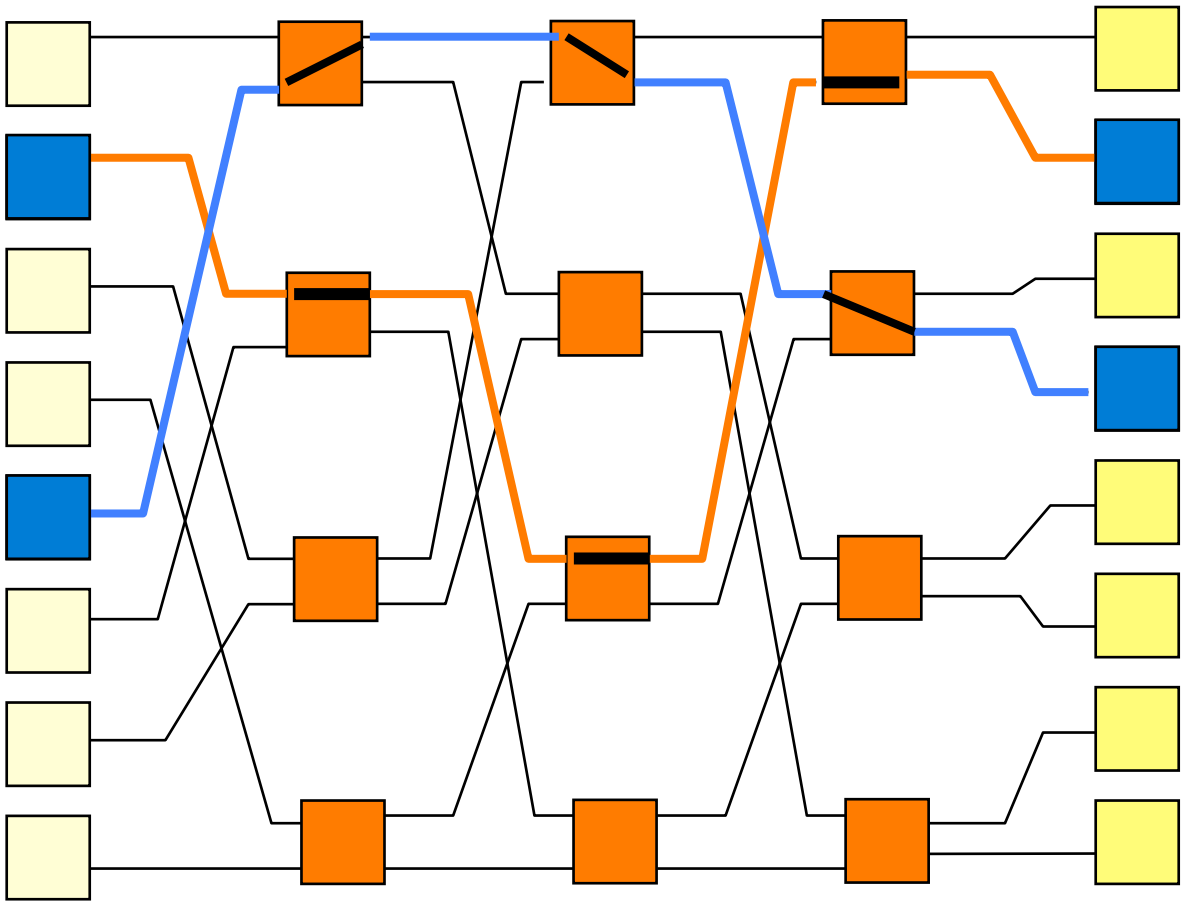
Verbindungs-orientierte Multi-Prozessorsysteme.
Realisierung: Spezielle Schaltnetzwerke (Kreuzschienenverteiler, Omega Netzwerk, Banyan trees,..)
Problem: Komplexität des Verbindungsnetzwerks.



Ein Omega Netzwerk

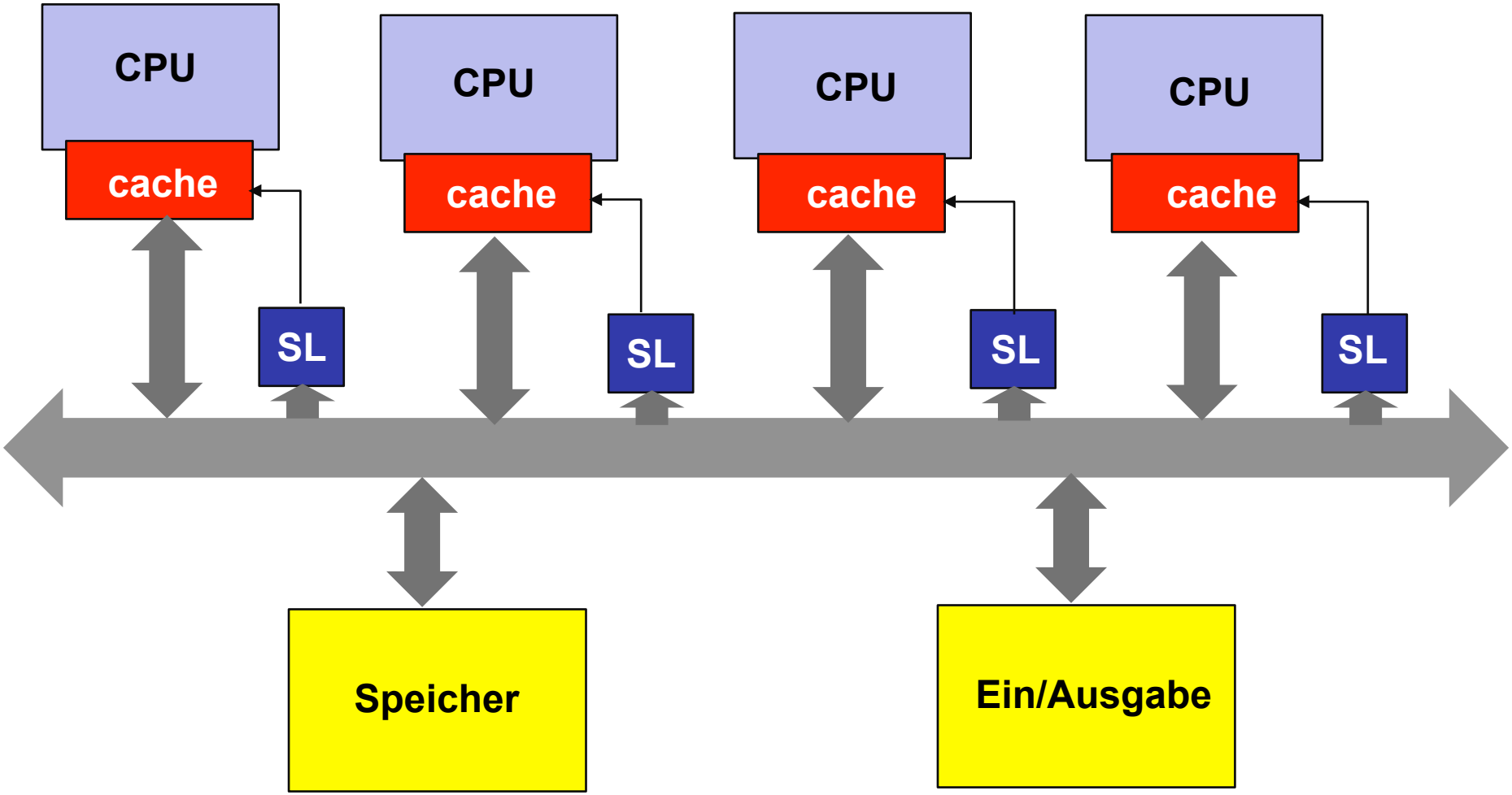
log2 n stages

2^k = N inputs



NUMA (wenige Kerne): Symmetric Multi Processing

Cache Kohärenz durch "Schnüffel-Logik" (Snooping Caches)

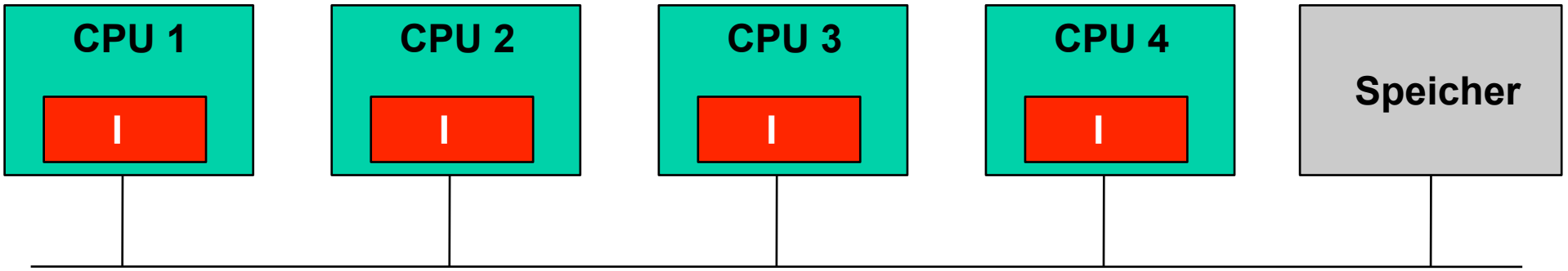


MESI Cache Coherency Protocol

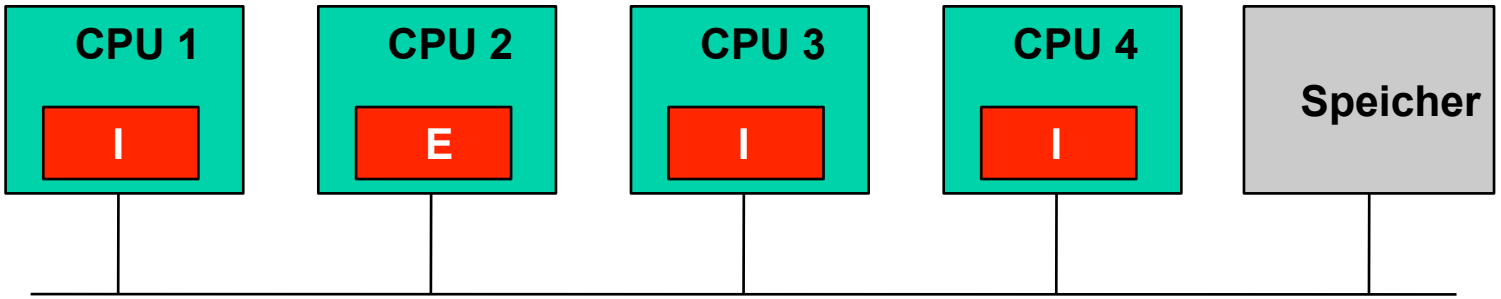
Modified **E**xclusive **S**hared **I**nvalid

Ein Cache-Eintrag kann die folgenden Zustände annehmen:

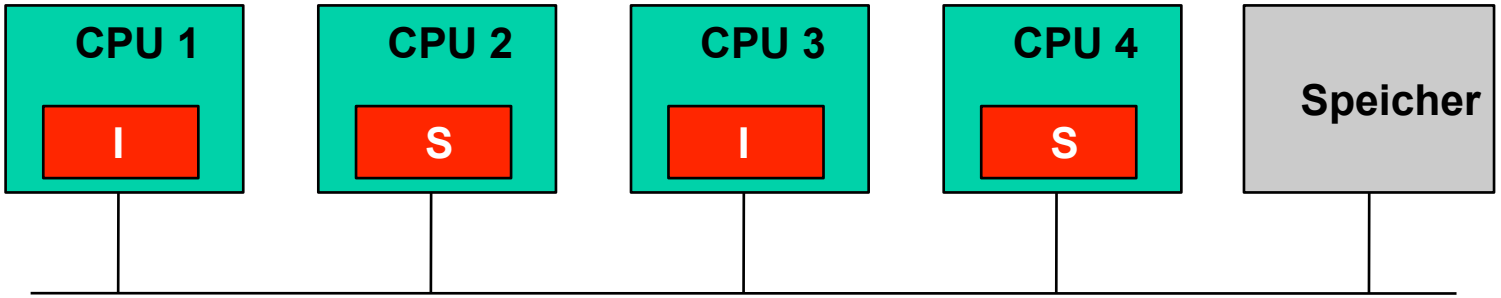
- **Ungültig (I)**
- **Shared (S): Dieser Block kann sich in mehreren Caches befinden.**
- **Exclusive (E): Dieser Block befindet sich in keinem anderen Cache, im (Haupt-) Speicher ist eine aktuelle Kopie.**
- **Modifiziert (M): Der Eintrag ist gültig, der (Haupt-) Speicher ist ungültig, d.h. es ist eine veraltete Kopie dort vorhanden.**



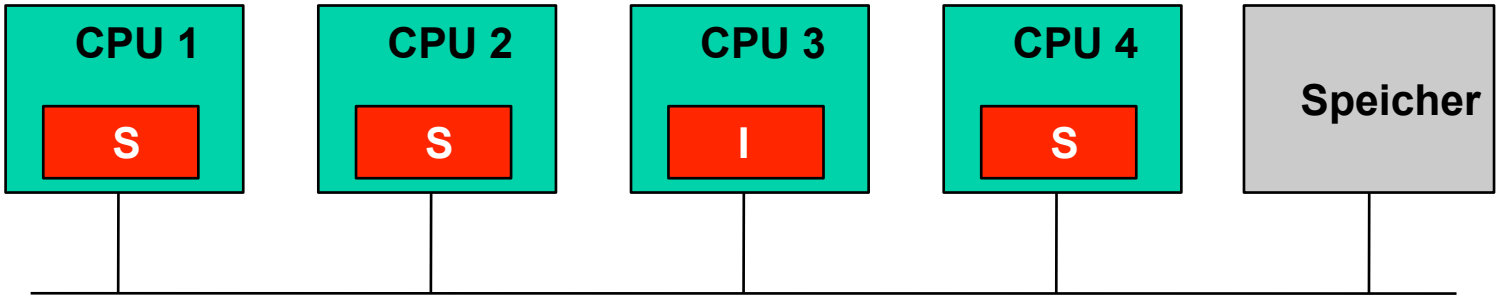
MESI Cache Coherency Protocol



CPU 2 liest Block x aus dem Speicher

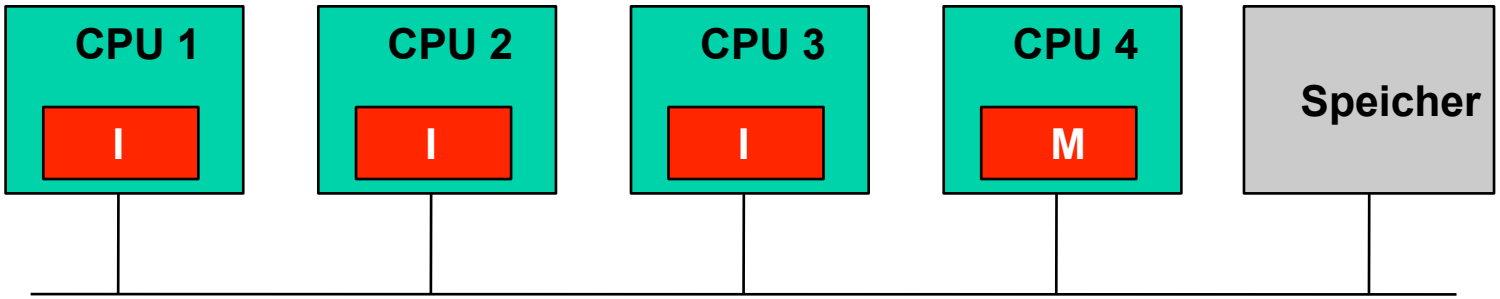


CPU 4 liest Block x aus dem Speicher. CPU 2 hat das "gesehen", beide setzen Status auf "S".

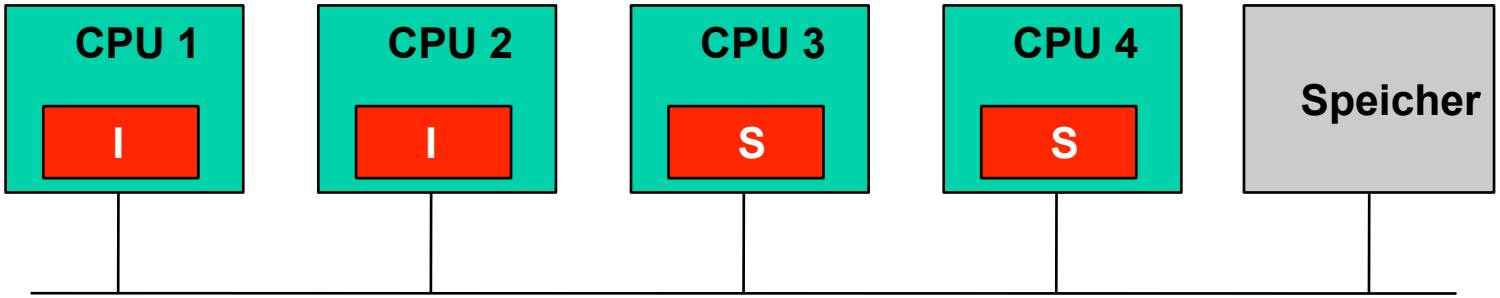


CPU 1 liest Block x aus dem Speicher. Status auf "S".

MESI Cache Coherency Protocol



CPU 4 beschreibt einen Eintrag von Block x. Sie legt ein "Invalidate" Signal auf den Bus, so dass alle anderen ihren Cache Block ungültig kennzeichen.

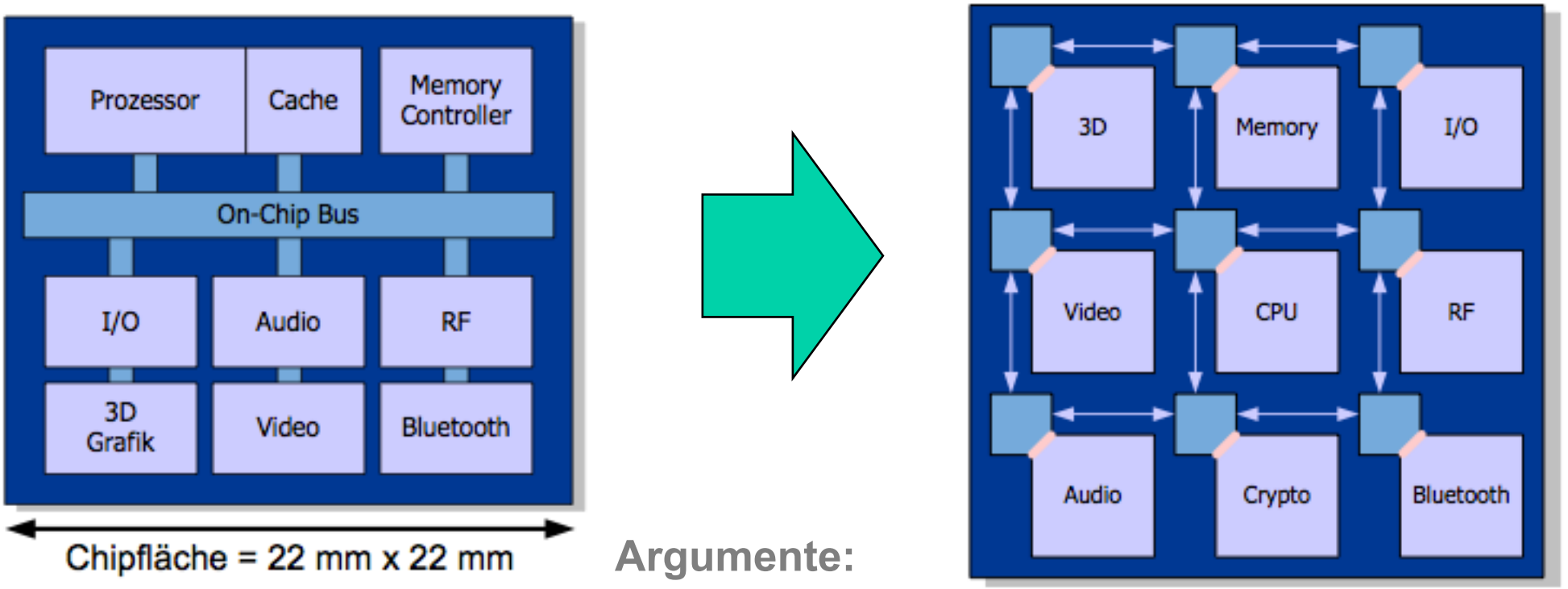


CPU 3 liest einen Eintrag von Block x. Da CPU 4 die Adresse sieht, legt sie ein Wartesignal auf den Bus und schreibt ihren modifizierten Block zurück. Dann kann CPU 3 lesen. Sowohl CPU 3 wie auch 4 markieren den Eintrag mit "S".

- Lesen kann nebenläufig lokal erfolgen
- Schreiben kann lokal erfolgen bis eine andere CPU diesen Eintrag liest.
- Durch Snooping werden Konflikte automatisch erkannt.

Systems on Chip (SoC) und Networks on Chip (NoC)

Vom Systembus **zum** **Netzwerk**

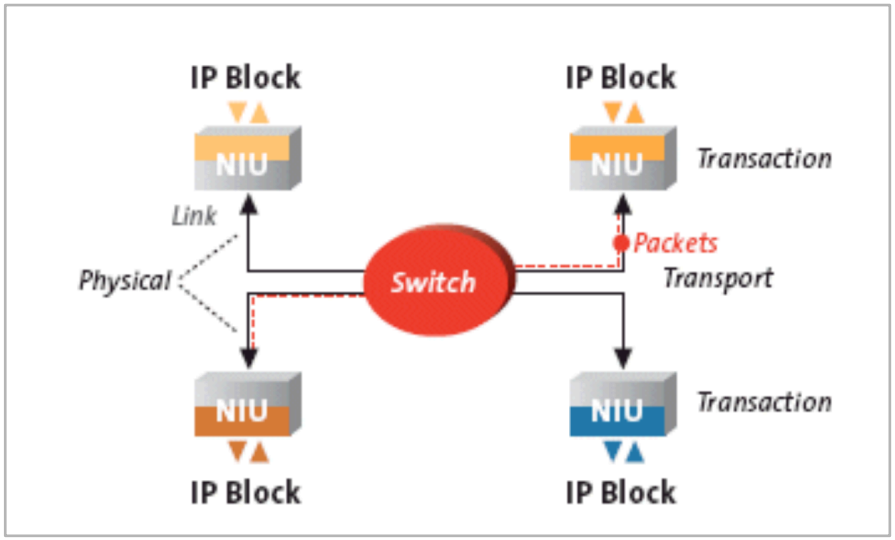


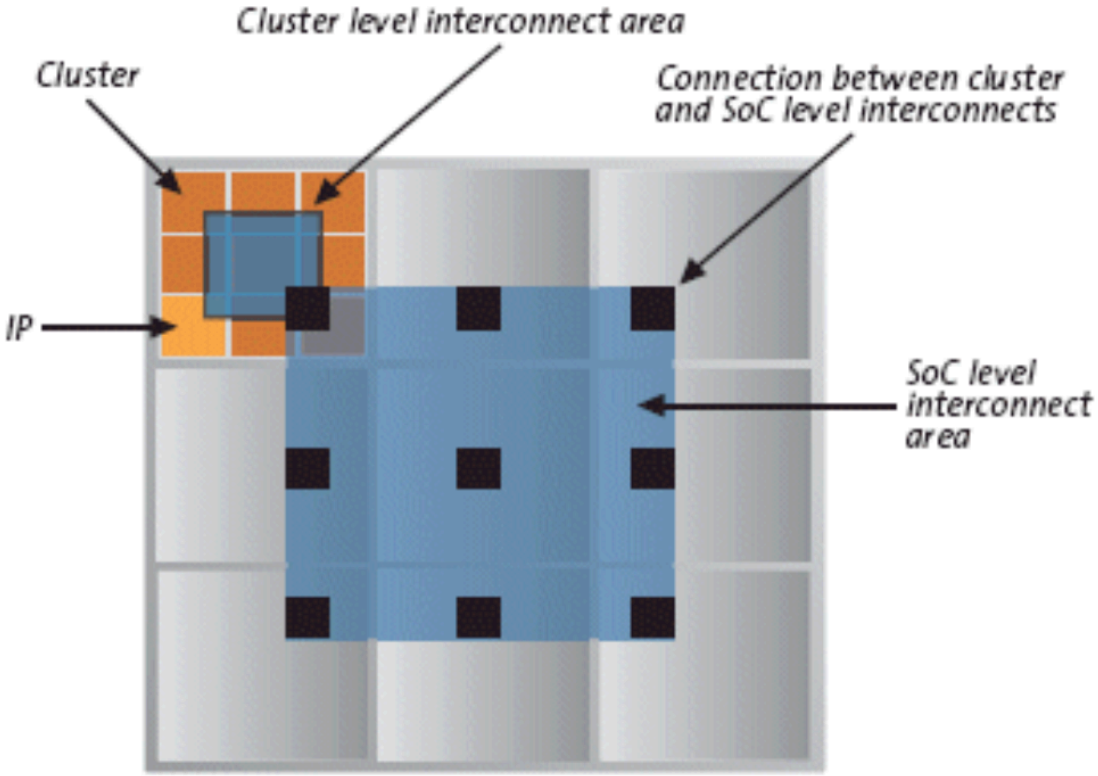
Argumente:

- Synchronität, Taktverteilung
- Leistungsverbrauch
- Getrennte Komponentenentwicklung

Abbildungen aus: Claas Cornelius,Universität Rostock, GRK-Workshop, November 2005

Systems on Chip (SoC) und Networks on Chip (NoC)

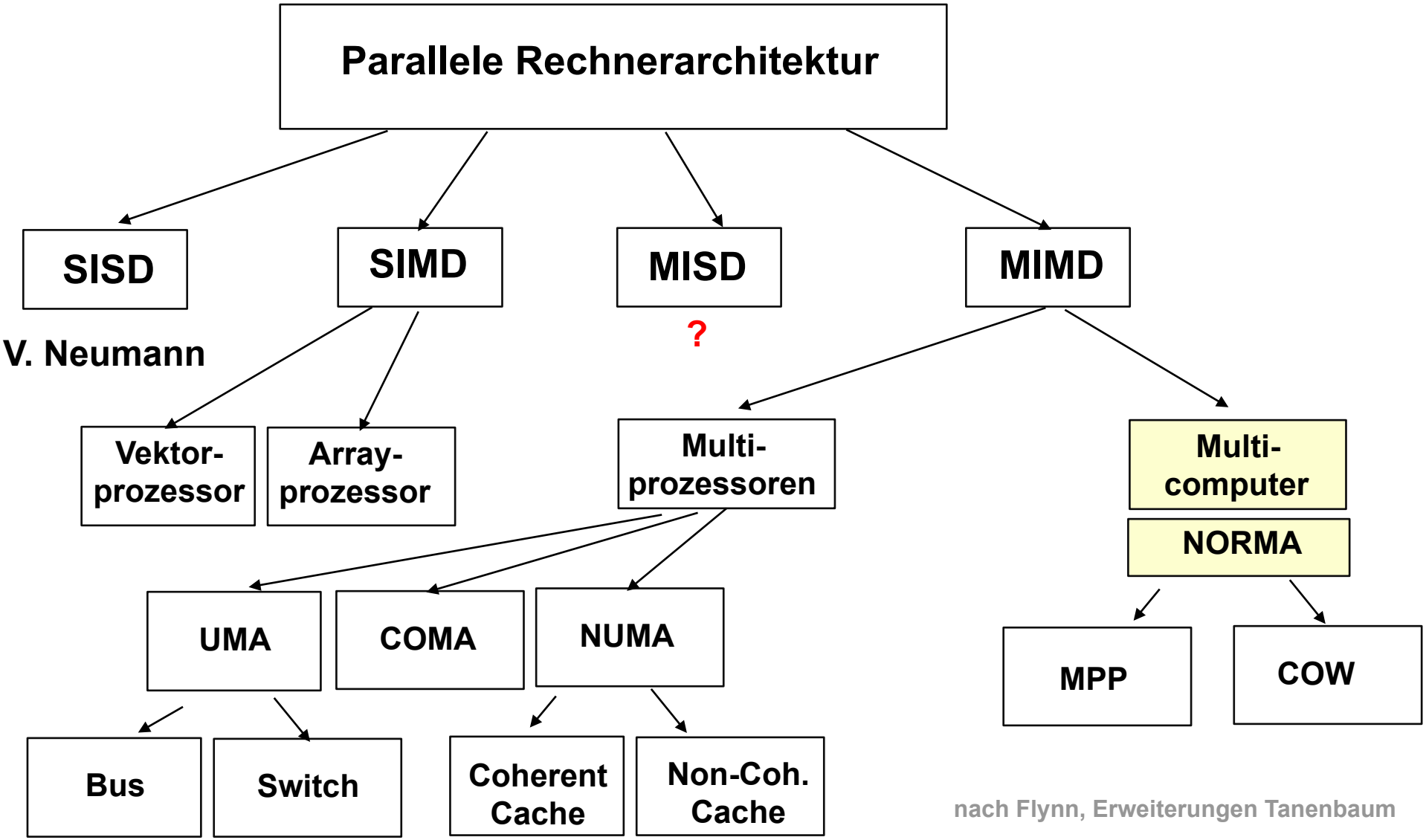




- **Mehrere Ebenen der Venetzung: Transaction, Transport (Packet), Physical (Wires)**
- **Auf SoC-Ebene Paket-orientiert**
- **Leistungsfähigkeit von Verbindungen kann nach Bedarf skaliert werden**

Abbildungen aus: D&R Industry Articles, A comparison of Network-on-Chip and Busses,2009, by Arteris (www.arteris.com)

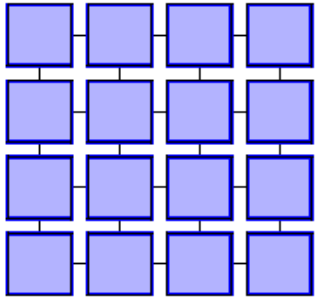
Taxonomie*



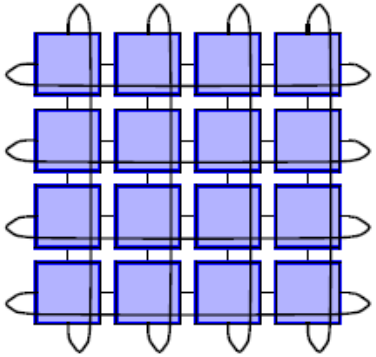
nach Flynn, Erweiterungen Tanenbaum

Multi-Prozessor Systeme mit lokalen Verbindungen

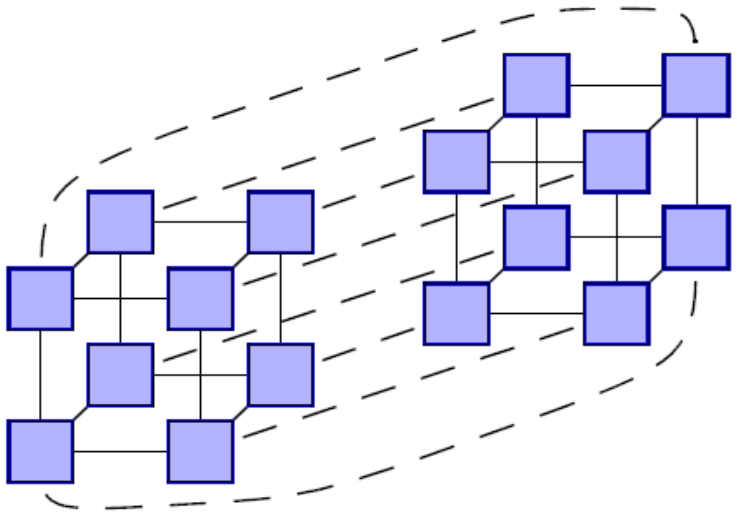
Grid



Torus



Hypercube



	dimensions	max. distance
Grid	2	2n (6 bei 4x4 Knoten)
Torus	3	n/2 + m/2 (4 bei 4x4 Knoten)
Cube	3	3
Hyperc.	4	3

Ausblick

Inhaltliche Ausrichtung

Einführung und Grundlagen:
Rechnerarchitektur:

- ***Einfacher Modellrechner***
- ***Interpreter und Mikroprogrammierung***
- ***ISA und Rechnerfamilien***
- ***Ein einfacher (Mikro-) Prozessor***
- ***Unterbrechungsverarbeitung***
- ***Adressierungsoptionen und alternative***
- ***Instruktionssätze (68K, MIPS, JVM)***
- ***Assemblerprogrammierung***
- ***Prozessornahe Programmiertechniken***

Erhöhung der Rechenleistung:

- ***Entwurfsphilosophie der RISC-Prozessoren***
- ***Pipelines***

Speicherorganisation

- ***Adressumsetzung und Caches***
- ***Metriken zur Leistungsabschätzung***
- ***Parallelrechner***