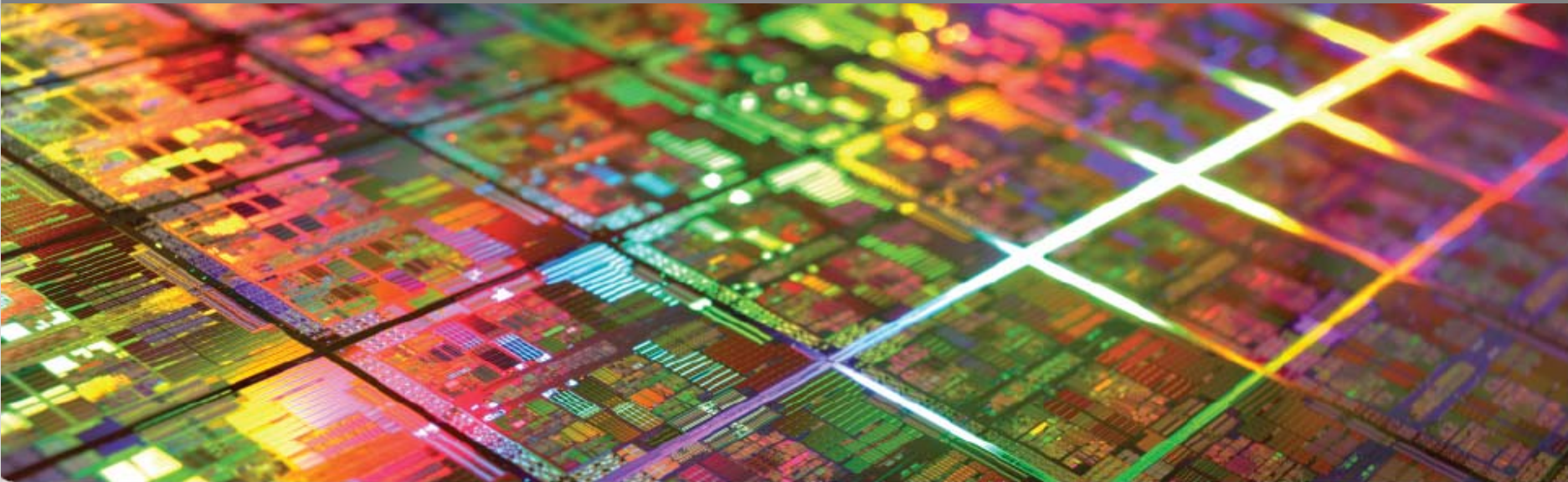


Rechnerstrukturen

Vorlesung im Sommersemester 2010

Prof. Dr. Wolfgang Karl

Fakultät für Informatik – Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

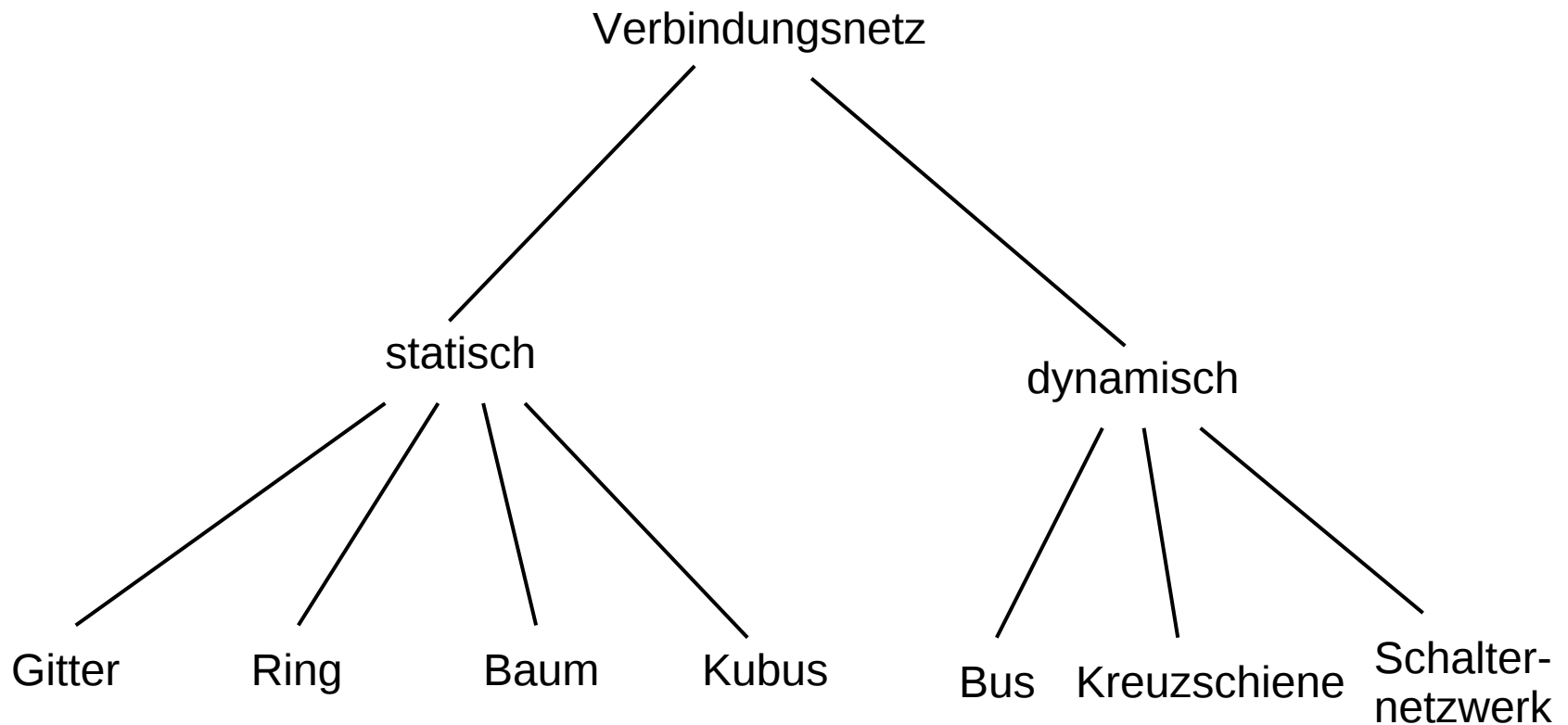


Vorlesung Rechnerstrukturen

- **Kapitel 3: Multiprozessoren – Parallelismus auf Prozess-/Blockebene**
- 3.5 Verbindungsstrukturen

Verbindungsstrukturen

■ Topologie: Klassifizierung



Verbindungsstrukturen

- **Topologie:**
- Statische Verbindungsnetze
 - Nach Aufbau des Verbindungsnetzes bleiben die Verbindungen fest
 - Gute Leistung für Probleme mit vorhersagbaren Kommunikationsmustern zwischen benachbarten Knoten

Verbindungsstrukturen

■ Topologie:

■ Statische Verbindungsnetze

■ Vollständige Verbindung

- Jeder Knoten ist mit jedem anderen Knoten verbunden
- Höchste Leistungsfähigkeit
 - Arbeitet für alle Anwendungen mit allen Arten von Kommunikationsmustern effizient
- Aber: nicht praktikabel in Parallelrechnern
 - Netzwerkkosten steigen quadratisch mit der Anzahl der Prozessoren

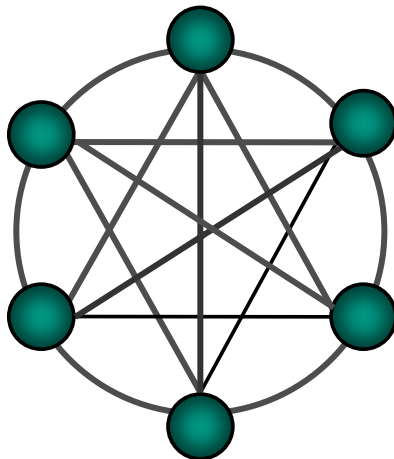
Verbindungsstrukturen

■ Topologie:

■ Statische Verbindungsnetze

■ Vollständige Verbindung

- Jeder Knoten ist mit jedem anderen Knoten verbunden
- Höchste Leistungsfähigkeit
 - Arbeitet für alle Anwendungen mit allen Arten von Kommunikationsmustern effizient
- Aber: nicht praktikabel in Parallelrechnern
 - Netzwerkkosten steigen quadratisch mit der Anzahl der Prozessoren



Verbindungsstrukturen

- **Topologie:**
- Statische Verbindungsnetze
- Gitterstrukturen
 - 1-dimensionales Gitter (lineares Feld, Kette)
 - Verbindet N Knoten mit $(N-1)$ Verbindungen
 - Endknoten haben den Grad 1, Zwischenknoten den Grad 2 und sind mit benachbarten Knoten verbunden
 - Diameter r ist $N-1$
 - Disjunkte Bereiche des linearen Netzwerkes können gleichzeitig genutzt werden, aber es sind mehrere Schritte notwendig, um eine Nachricht zwischen zwei nicht benachbarte Knoten zu verschicken



Verbindungsstrukturen

■ Topologie:

■ Statische Verbindungsnetze

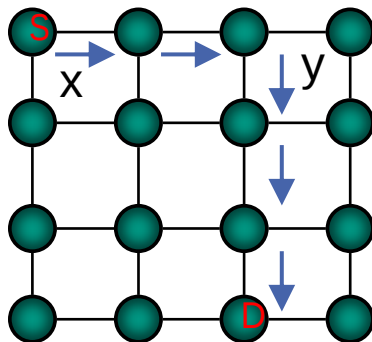
■ Gitterstrukturen

■ k-dimensionales Gitter mit N Knoten

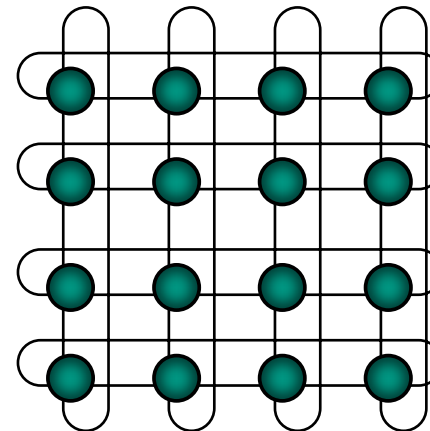
- Innere Knoten haben den Grad $2k$, wobei die $2k$ benachbarten Knoten miteinander verbunden sind

- In einem k-dimensionalen Netzwerk mit $\sqrt[k]{N}$ Knoten in jeder Dimension beträgt der Diameter $k(\sqrt[k]{N} - 1)$

2-dim. Gitter



2-dim. Torus



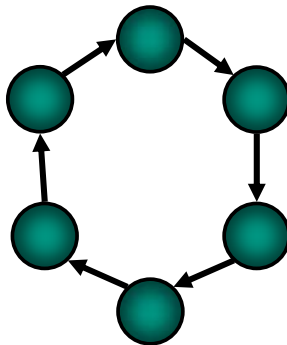
Verbindungsstrukturen

■ Topologie:

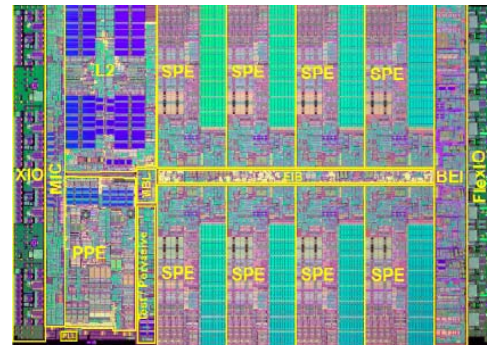
■ Statische Verbindungsnetze

■ Ring

- Erhält man, wenn man die Endknoten eines linearen Feldes miteinander verbindet
- Unidirektionaler Ring mit N Knoten
 - Nachrichten werden in einer Richtung vom Quellknoten zum Zielknoten verschickt
 - Diameter r ist $N-1$
 - Bei Ausfall einer Verbindung bricht die Kommunikation zusammen



IBM Cell Broadband Engine:



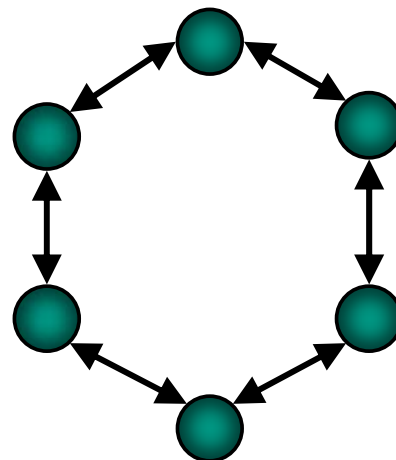
Verbindungsstrukturen

■ Topologie:

■ Statische Verbindungsnetze

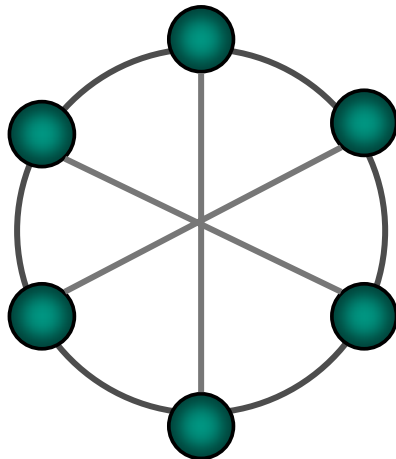
■ Bidirektionaler Ring mit N Knoten

- symmetrisches Netzwerk
- Der längste Pfad, den eine Nachricht nehmen muss, ist nicht länger als $N/2$
- Bei Ausfall einer Verbindung bricht die Kommunikation noch nicht zusammen, während zwei Ausfälle von Verbindungen das Netzwerk in zwei disjunkte Teilnetzwerke aufteilen

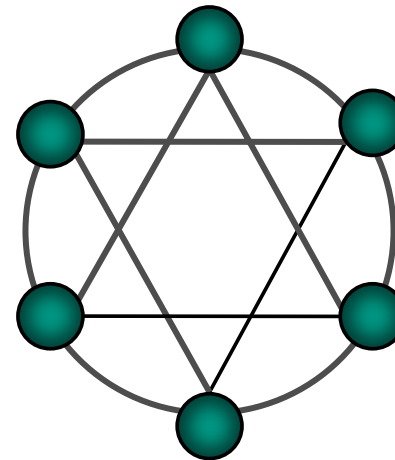


Verbindungsstrukturen

- **Topologie:**
- Statische Verbindungsnetze
- Chordaler Ring
 - Hinzufügen redundanter Verbindungen
 - erhöht Fehlertoleranzeigenschaft des Verbindungsnetzwerkes
 - Höherer Knotengrad und kleinerer Diameter gegenüber Ring



Chordaler Ring mit Knotengrad 3



Chordaler Ring mit Knotengrad 4

Verbindungsstrukturen

■ Topologie:

■ Statische Verbindungsnetze

■ Baumstrukturen

■ Binärer Baum mit m-Ebenen:

■ Auf Ebene m: $N=2^m-1$ Knoten

■ Diameter: $2(m-1)$

■ Adressierung der Knoten:

■ Die Knotennummer auf Ebene m besteht aus m Bits

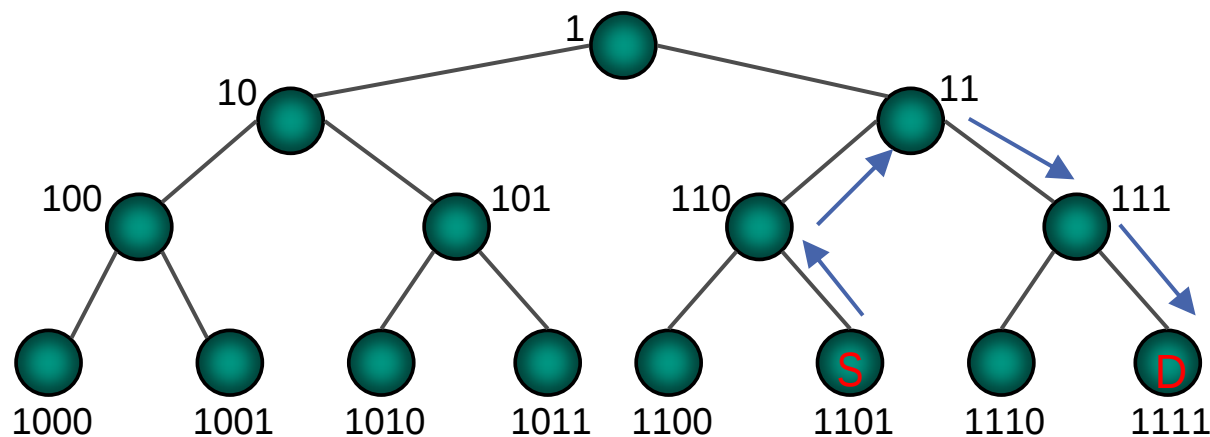
■ Der Wurzelknoten hat die Nummer 1

■ Die Nummer des linken Kindknotens erhält man durch Hinzufügen einer 0 an die niederwertige Stelle der Adresse des Elternknoten

■ Die Nummer des rechten Kindknotens erhält man durch Hinzufügen einer 1 an die niederwertige Stelle der Adresse des Elternknoten

Verbindungsstrukturen

- **Topologie:**
- Statische Verbindungsnetze
- Baumstrukturen: Routing
 - Finde gemeinsamen Elternknoten P von S und D
 - Gehe von S nach P und von P nach D



Verbindungsstrukturen

■ Topologie:

■ Statische Verbindungsnetze

■ Baumstrukturen: Routing

- Die Binärdarstellung der Adresse eines Quellknotens S auf Ebene i sei $S_i S_{i-1} \dots S_1$ und die der Adresse des Zielknotens D auf Ebene j sei $D_j D_{j-1} \dots D_1$
- Finde die gemeinsamen höchstwertigen Bits von S und D , so dass die Adresse des Elternknotens P gleich $D_j D_{j-1} \dots D_x = S_i S_{i-1} \dots S_{(i-j+x)}$ ist
- Steige von S $(i-j+x)$ Ebenen auf nach P
- for $k=x-1$ step 1 until 0
 - {steige nach links ab, falls $D_x=0$
 - steige nach rechts ab, falls $D_x=1$ }

Verbindungsstrukturen

■ Topologie:

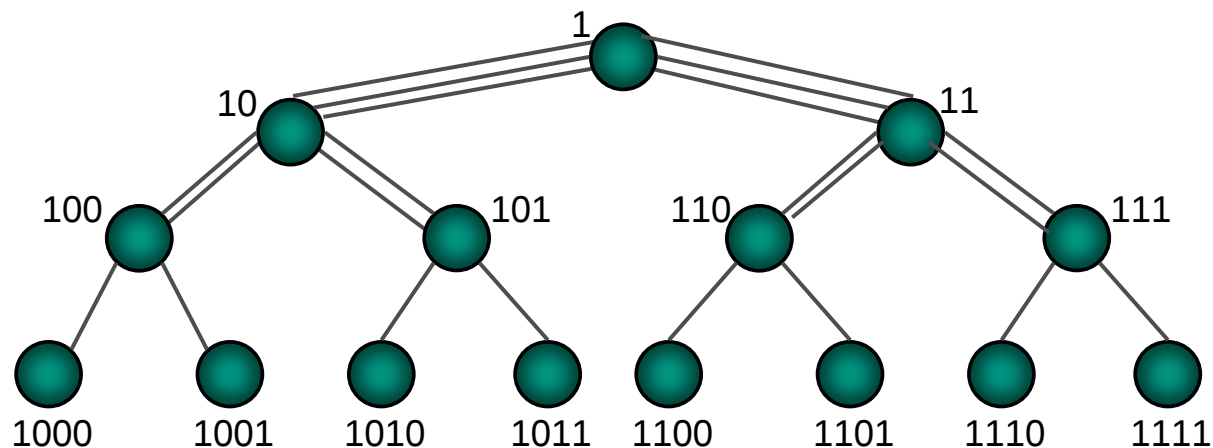
■ Statische Verbindungsnetze

■ Baumstrukturen

■ Fat Tree:

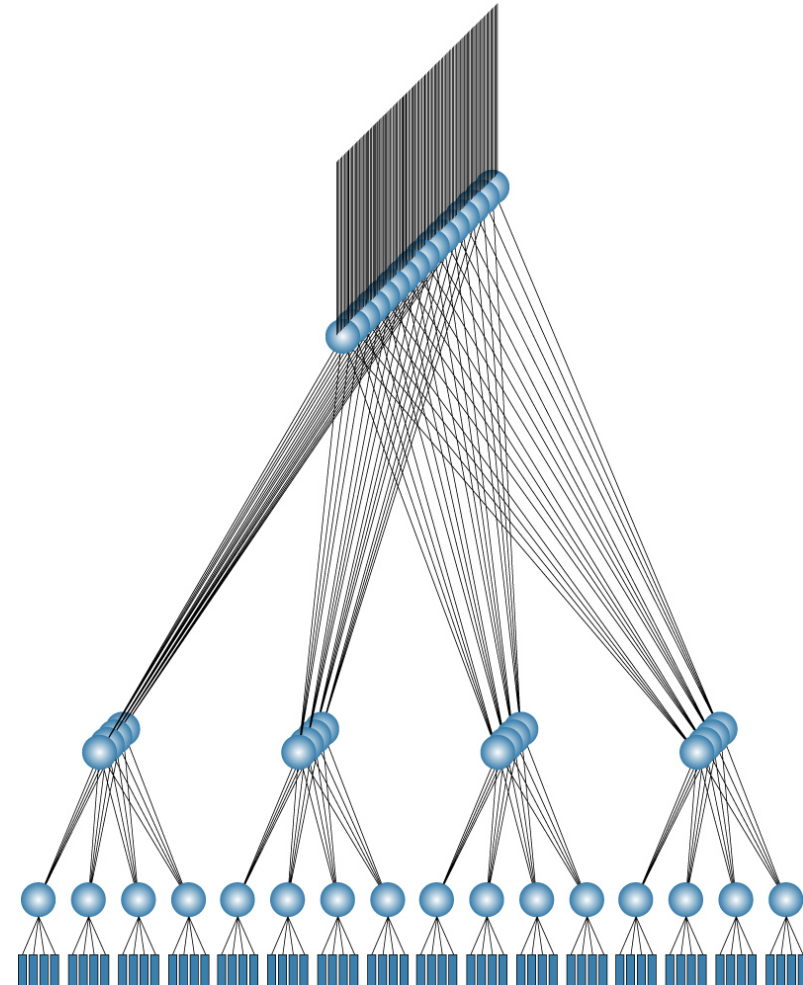
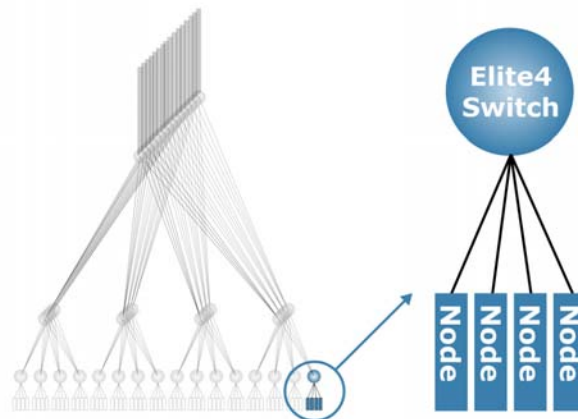
■ Lösung des Blockierungsproblems in Richtung Wurzel

■ Kommunikationskanäle werden größer, je näher man sich der Wurzel nähert



Verbindungsstrukturen

- **Topologie:**
- Statische Verbindungsnetze
- Beispiel: Quadrics QSnet

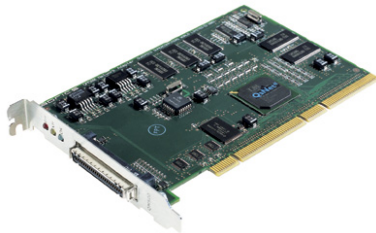


Quelle: D. Addison, J. Beecroft, D. Hewson, M. McLaren (Quadrics Ltd.),
Fabrizio Petrini (LANL): A network for Supercomputing Applications. Hot Chips,
August 2003, <http://www.c3.lanl.gov/~fabrizio/quadrics.html>

Verbindungsstrukturen

- **Topologie:**
- Statische Verbindungsnetze
- Beispiel: Quadrics QSnet

Elan 4 network interface card:



Elite 4 Switch Component:



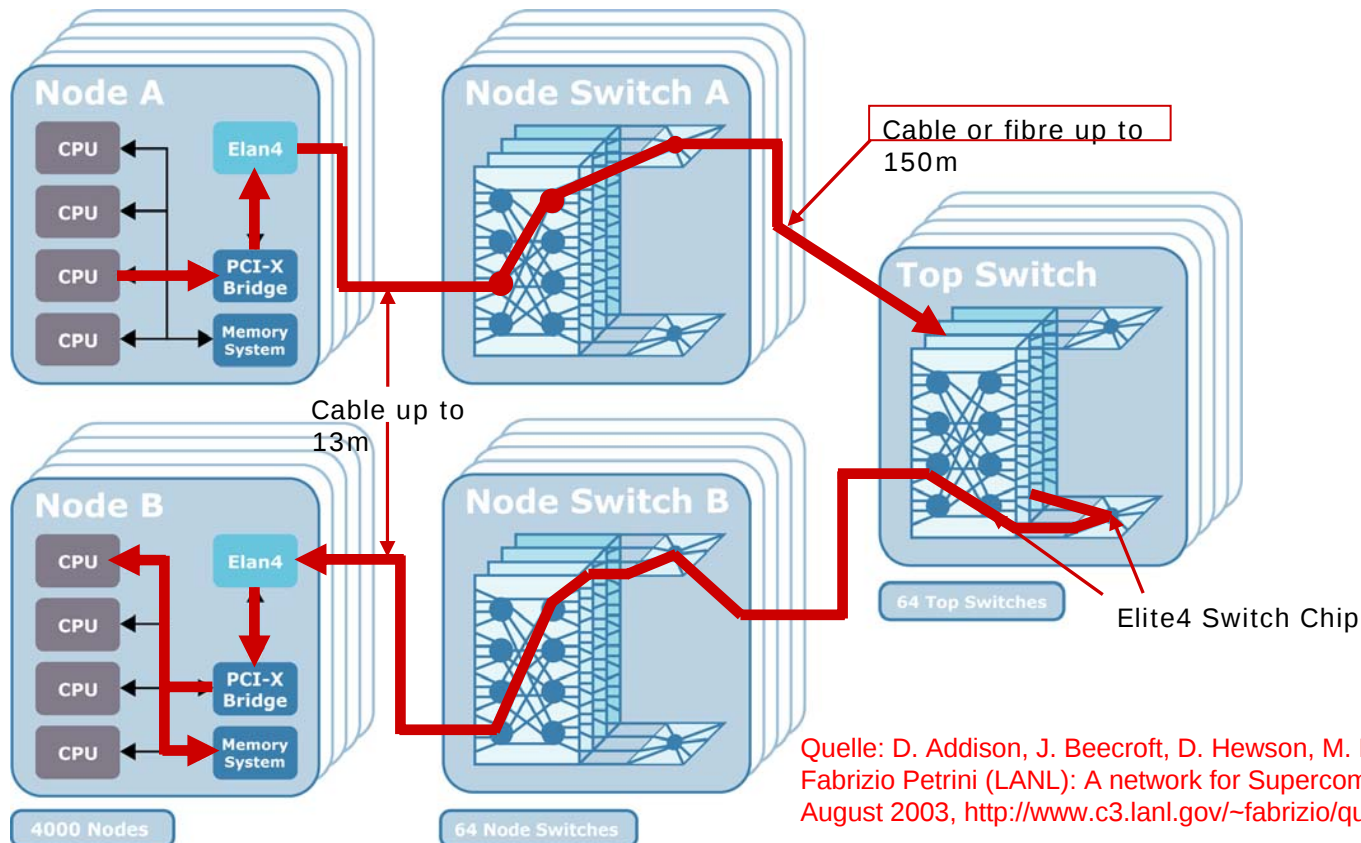
QsNet II Switch:



Quelle: D. Addison, J. Beecroft, D. Hewson, M. McLaren (Quadrics Ltd.),
Fabrizio Petrini (LANL): A network for Supercomputing Applications. Hot Chips,
August 2003, <http://www.c3.lanl.gov/~fabrizio/quadrics.html>

Verbindungsstrukturen

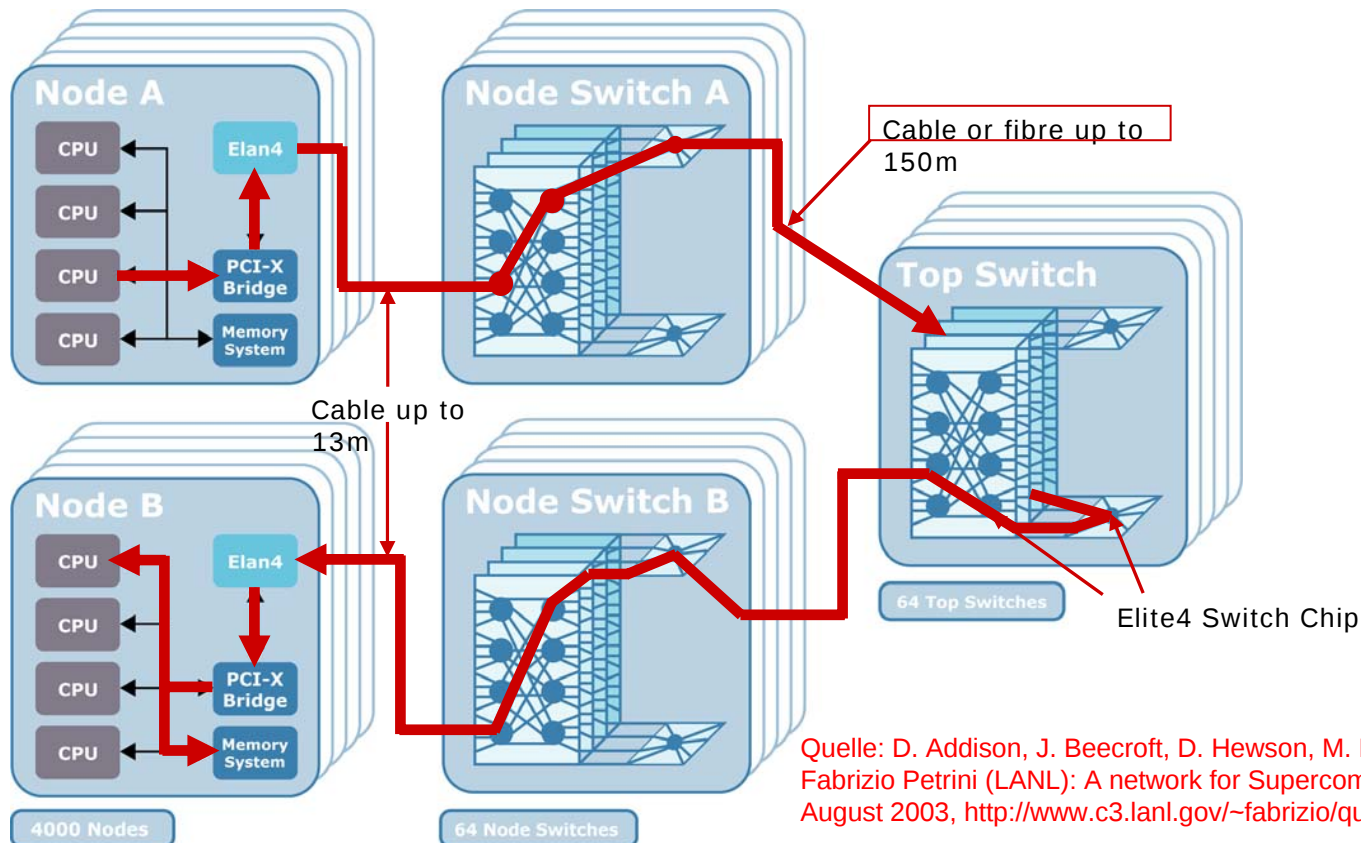
- **Topologie:**
- Statische Verbindungsnetze
- Beispiel: Quadrics QSnet



Quelle: D. Addison, J. Beecroft, D. Hewson, M. McLaren (Quadrics Ltd.), Fabrizio Petrini (LANL): A network for Supercomputing Applications. Hot Chips, August 2003, <http://www.c3.lanl.gov/~fabrizio/quadrics.html>

Verbindungsstrukturen

- **Topologie:**
- Statische Verbindungsnetze
- Beispiel: Quadrics QSnet



Quelle: D. Addison, J. Beecroft, D. Hewson, M. McLaren (Quadrics Ltd.), Fabrizio Petrini (LANL): A network for Supercomputing Applications. Hot Chips, August 2003, <http://www.c3.lanl.gov/~fabrizio/quadrics.html>

Verbindungsstrukturen

■ Topologie:

■ Statische Verbindungsnetze

■ K-ärer n-Kubus (Cubes, Würfel)

- Allgemeine Form eines Kubus-Verbindungsnetzwerkes
- Ringe, Gitter, oder Hyperkubi sind topologisch isomorph zu einer Familie von K-ären n-Kubus Netzwerken
 - n ist die Dimension
 - Radius K ist die Anzahl der Knoten, die einen Zyklus in einer Dimension bilden
- Enthält $N=K^n$ Knoten
- Die Knoten werden über eine n-stellige binäre Radius K Zahl der Form $a_0, a_1, \dots, a_{n-1}$ adressiert
 - Jede Stelle $0 \leq a_i < K$ stellt die Position des Knotens in der entsprechenden i-ten Dimension dar, mit $0 \leq i \leq n-1$
 - Ein Nachbarknoten in der i-ten Dimension zu einem Knoten mit Adresse $a_0, a_1, \dots, a_{n-1}$ kann erreicht werden mit $a_0, a_1, \dots, a_{(i \pm 1)} \bmod K \dots a_{n-1}$.
- Knotengrad ist $2n$ und der Diameter ist

$$n \left\lceil \frac{k}{2} \right\rceil$$

Verbindungsstrukturen

■ Topologie:

■ Statische Verbindungsnetze

■ Hyperkubus (Hypercubes)

■ Verallgemeinerter Würfel:

- die $N = 2^n$ Prozessoren sind Ecken eines n -dimensionalen Würfels, wobei die Verbindungen dann die Kanten des Würfels darstellen.

■ Komplexität ist $(N \cdot \log_2 N)/2$.

■ Diameter beträgt $\log_2 N$.

■ Lange Zeit häufigste Verbindungsstruktur bei den nachrichtengekoppelten Multiprozessoren, aber:

■ Skalierbarkeit:

- Jede Erweiterung benötigt mindestens die Verdopplung der Prozessorenanzahl.
- Aus räumlichen Anordnungsgründen begrenzt.

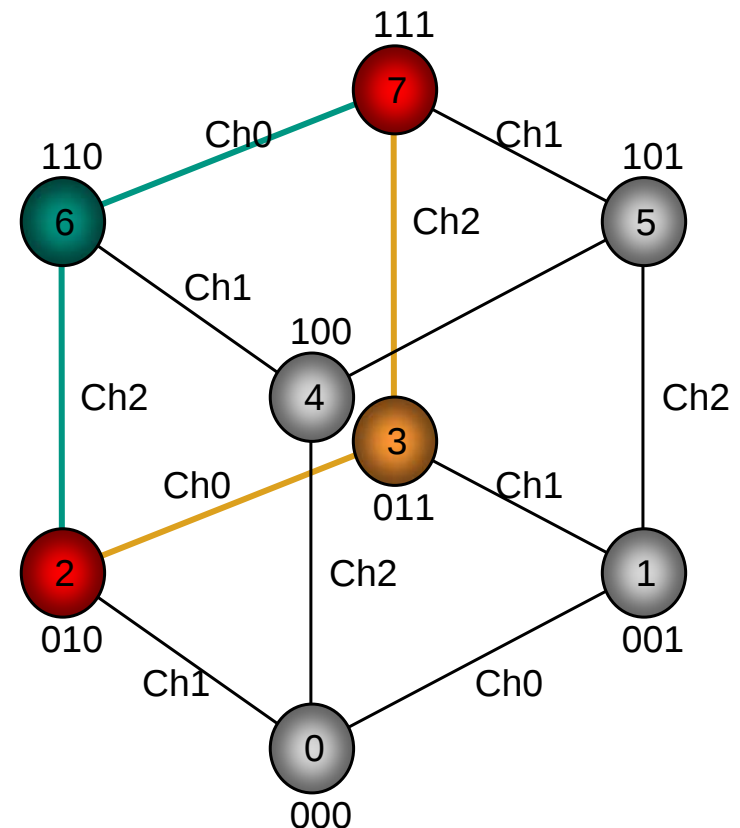
Verbindungsstrukturen

- **Topologie:**
- Statische Verbindungsnetze
- Hyperkubus
 - e-Cube Routing
 - Die Knotennummern werden als Binärzahlen geschrieben, dadurch unterscheiden sich benachbarte Knoten in genau einer Stelle, die zudem die Richtung der Verbindung angeben kann (Hamming Distanz)
 - Eine einfache Wegewahl:
die Bits in Start- und Zieladresse werden mittels einer XOR-Verbindung verknüpft und das Resultat bestimmt die möglichen Wege.

Verbindungsstrukturen

- **Topologie:**
- Statische Verbindungsnetze
- Hyperkubus
- Beispiel:

- $A = (010)$ und $B = (111)$
- $W = A \text{ XOR } B = 101$
- $(010) \rightarrow (011) \rightarrow (111)$,
- $(010) \rightarrow (110) \rightarrow (111)$



Verbindungsstrukturen

■ Topologie:

■ Dynamische Verbindungsnetzwerke:

- Geeignet für Anwendungen mit variablen und nicht regulären Kommunikationsmustern

■ Bus:

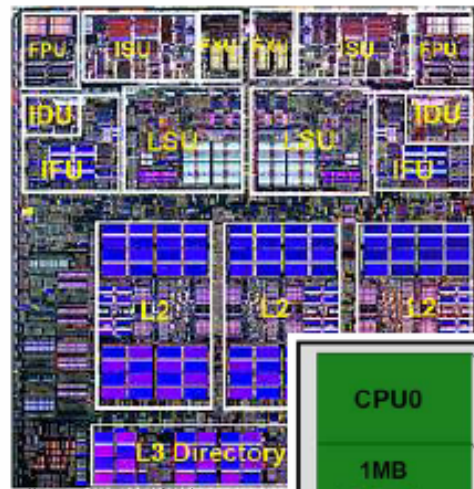
- Wird von den am Bus angeschlossenen Prozessoren gemeinsam benützt
- Ein Datentransport zu einem Zeitpunkt
- Nachricht von einer Quelle zu jedem Ziel in einem Schritt
- Busbandbreite = $w * f$
 - w : Anzahl der Datenleitungen (Busbreite)
 - F : Frequenz
 - Bestimmt maximale Anzahl der Prozessoren, d. h. die Bandbreite muss mit dem Produkt der Anzahl der Prozessoren und ihrer Geschwindigkeit abgestimmt werden

Verbindungsstrukturen

- **Topologie:**
- Dynamische Verbindungsnetzwerke:
- Bus:
 - Reduzierung des Busverkehrs
 - Verwendung von Cache-Speichern mit Cache-Kohärenz-Protokollen
 - Verwendung von sog. Split-Phase Busprotokollen
 - Das Protokoll gibt den Bus nach der Übertragung einer Speichereferenzanforderung wieder frei
 - Wenn der Speicher bereit ist, das Datum zu liefern, fordert dieser den Bus an und schickt die Daten als Antwort
 - Ermöglicht, dass andere Prozessoren in der Zwischenzeit den Bus anfordern können, vorausgesetzt, dass ein verschränkter Speicher vorliegt oder Pipelining möglich ist

Verbindungsstrukturen

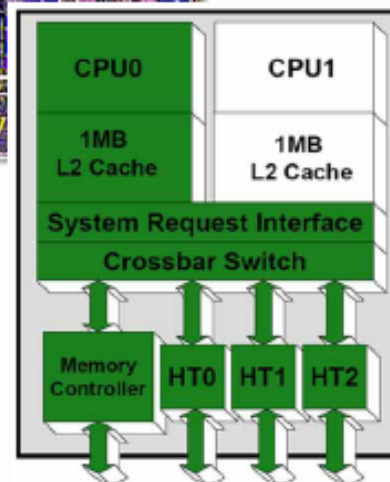
- **Topologie:**
- Dynamische Verbindungsnetzwerke:
- Bus / Kreuzschienenverteiler:



Quelle: S. Amarasinghe, MIT

IBM Power5

AMD Opteron

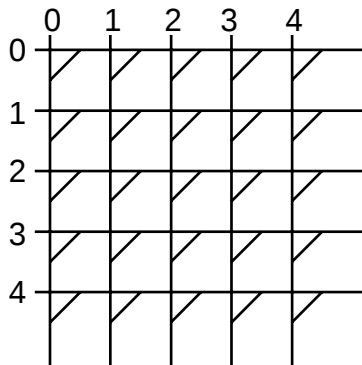


Verbindungsstrukturen

- **Topologie:**
- Dynamische Verbindungsnetzwerke:
- Kreuzschienenverteiler (Crossbar)
 - Hardware-Einrichtung, die so geschaltet werden kann, dass in einer Menge von Prozessoren alle möglichen disjunkten Paare von Prozessoren gleichzeitig und blockierungsfrei miteinander kommunizieren können.
 - In Abhängigkeit vom Zustand der Schaltelemente im Kreuzschienenverteiler können dann je zwei beliebige Elemente aus den verschiedenen Mengen miteinander kommunizieren.
 - Alle $N!$ Permutationen sind möglich
 - An den Kreuzungspunkten sitzen Schaltelemente: hoher Hardware-Aufwand
 - Kosten: N^2 Schaltelemente (bei N Knoten pro Dimension)
 - Ein Schaltelement entspricht einem Paar von Quelle und Ziel, so dass die Darstellung einer Permutation als eine Liste solcher Paare direkt zu der korrekten Schaltung der Schalterelemente führt.

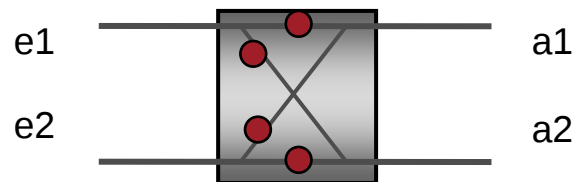
Verbindungsstrukturen

- **Topologie:**
- Dynamische Verbindungsnetzwerke:
- Kreuzschienenverteiler (Crossbar)
 - Vollständig vernetztes Verbindungswerk mit allen möglichen Permutationen der N Einheiten, die über das Netzwerk verbunden werden



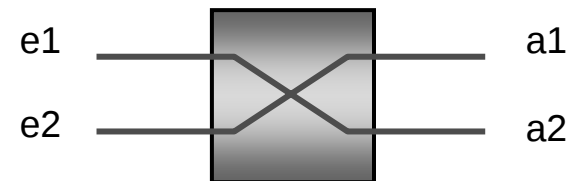
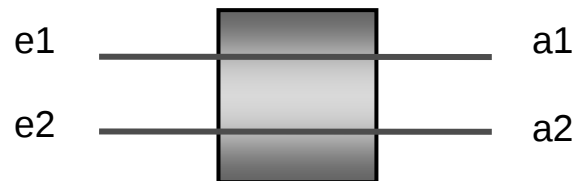
Verbindungsstrukturen

- **Topologie:**
- Dynamische Verbindungsnetzwerke:
- Schalterelemente (2x2 Kreuzschienenverteiler)
 - bestehen aus Zweierschaltern mit zwei Eingängen und zwei Ausgängen, die entweder durchschalten oder die Ein- und Ausgänge überkreuzen können



Durchschalten

● Kontakt, der geöffnet oder geschlossen werden kann



Vertauschen

Verbindungsstrukturen

- **Topologie:**
- Dynamische Verbindungsnetzwerke:
- Mehrstufige Verbindungsnetzwerke (Schalternetzwerke, Permutationsnetzwerke)
 - Kompromiss zwischen der niedrigeren Leistungsfähigkeit von Bussen und hohem Hardware-Aufwand von Kreuzschienenverteilern
 - Oft 2 x 2 Kreuzschienenverteiler (Schalterelement) als Grundelement

Verbindungsstrukturen

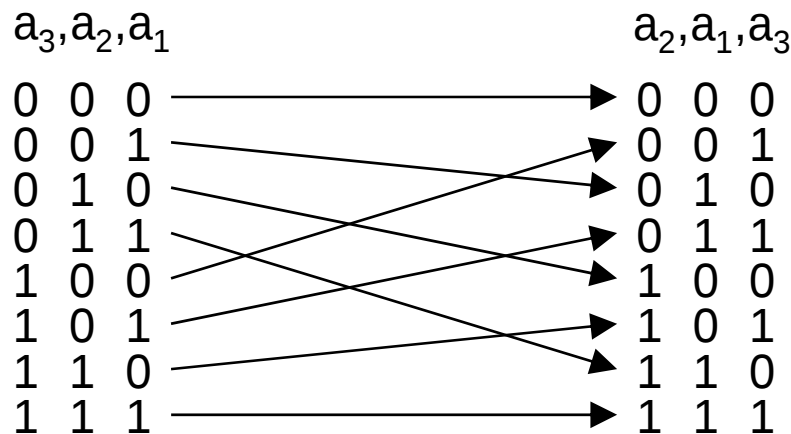
- **Topologie:**
- Dynamische Verbindungsnetzwerke:
- Permutationsnetze
 - p Eingänge des Netzes können gleichzeitig auf p Ausgänge geschaltet werden und somit wird eine Permutation der Eingänge erzeugt.
- Einstufige Permutationsnetze
 - enthalten eine einzelne Spalte von Zweierschaltern,
- Mehrstufige Permutationsnetze
 - enthalten mehrere solcher Spalten
 - Spalten: Stufen des Permutationsnetzwerkes

Verbindungsstrukturen

- **Topologie:**
- Dynamische Verbindungsnetzwerke:
- Permutationsnetze
- reguläre Permutationsnetzwerke:
 - p Eingänge, p Ausgänge und k Stufen mit jeweils $p/2$ Zweierschaltern, wobei die Zahl p normalerweise eine Zweierpotenz ist.
- Irreguläre Permutationsnetzwerke
 - weisen gegenüber der vollen regulären Struktur Lücken auf

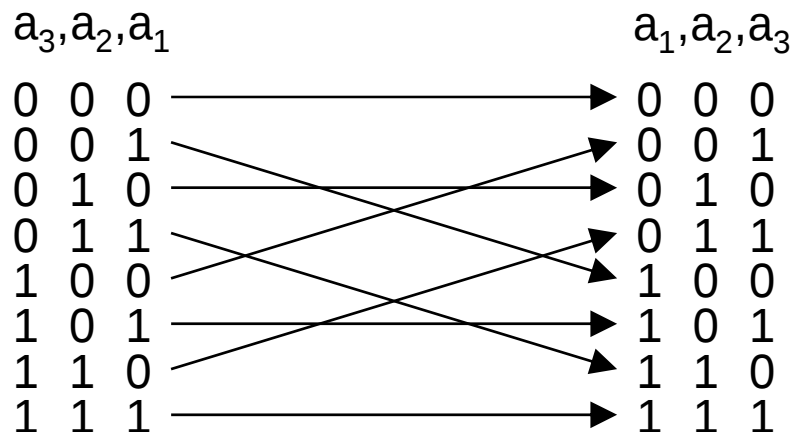
Verbindungsstrukturen

- **Topologie:**
- Dynamische Verbindungsnetzwerke:
- Permutationen
 - Mischpermutation M (Perfect Shuffle):
 - $M(a_n, a_{n-1}, \dots, a_2, a_1) = (a_{n-1}, \dots, a_2, a_1, a_n)$



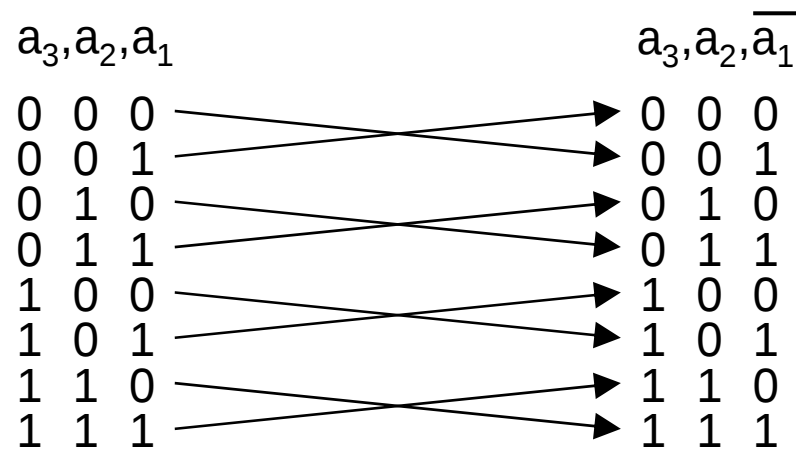
Verbindungsstrukturen

- **Topologie:**
- Dynamische Verbindungsnetzwerke:
- Permutationen
 - Kreuzpermutation K (Butterfly):
 - $K(a_n, a_{n-1}, \dots, a_2, a_1) = (a_1, a_{n-1}, \dots, a_2, a_n)$



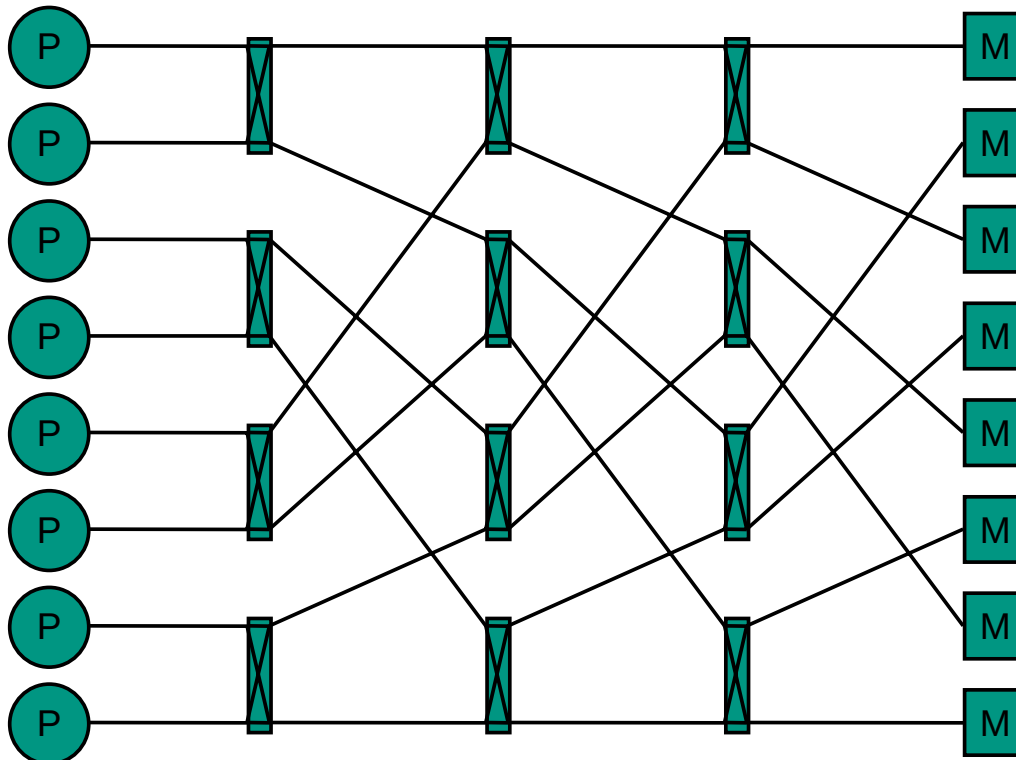
Verbindungsstrukturen

- **Topologie:**
- Dynamische Verbindungsnetzwerke:
- Permutationen
 - Tauschpermutation T (Butterfly):
 - Negation des niedrigwertigen Bits
 - $T(a_n, a_{n-1}, \dots, a_2, a_1) = (a_n, a_{n-1}, \dots, a_2, \overline{a_1})$



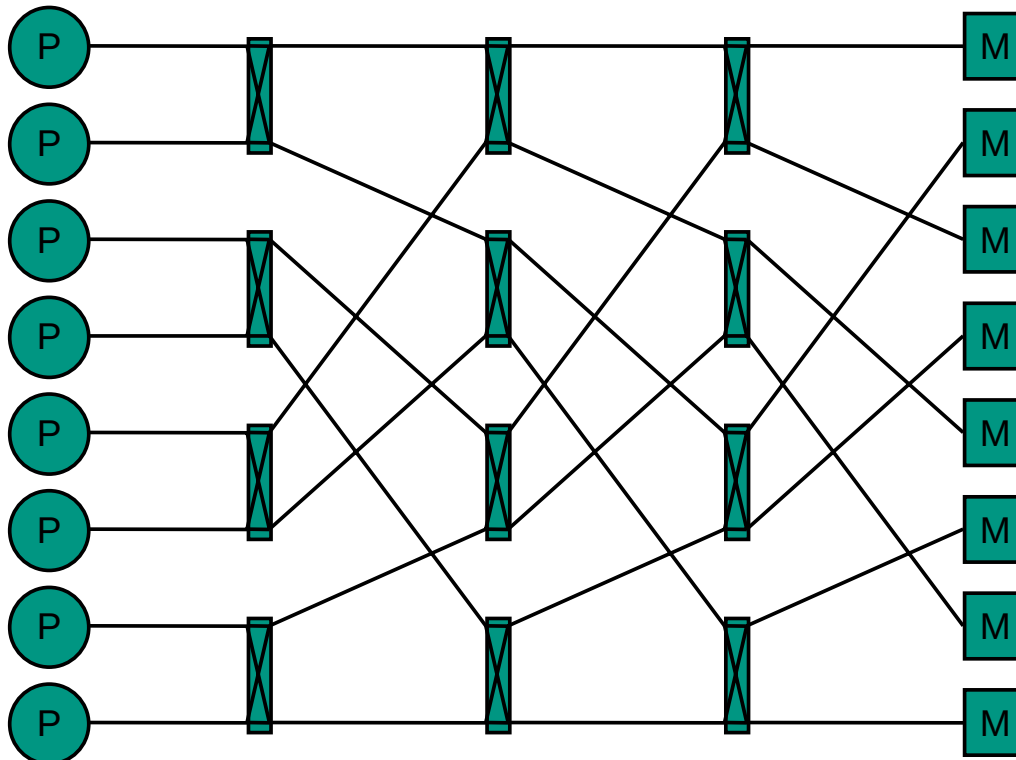
Verbindungsstrukturen

- **Topologie:**
- Dynamische Verbindungsnetzwerke:
- Speichergekoppeltes Omega-Netzwerk mit 8 Eingängen und 8 Ausgängen



Verbindungsstrukturen

- **Topologie:**
- Dynamische Verbindungsnetzwerke:
- Speichergekoppeltes Omega-Netzwerk mit 8 Eingängen und 8 Ausgängen



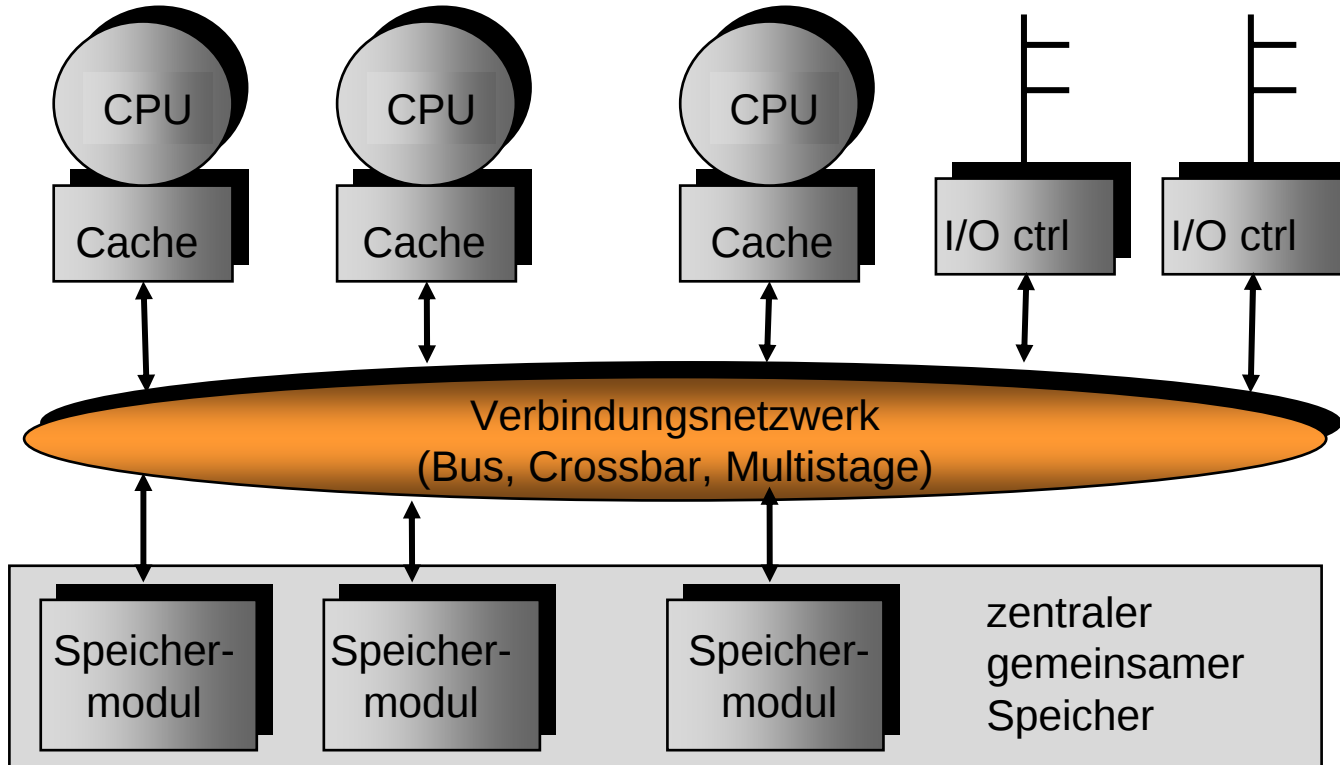
Vorlesung Rechnerstrukturen

- **Kapitel 3: Multiprozessoren – Parallelismus auf Prozess-/Blockebene**
- 3.6 Multiprozessoren mit gemeinsamem Speicher

Multiprozessorsysteme

■ Multiprozessor mit gemeinsamem Speicher

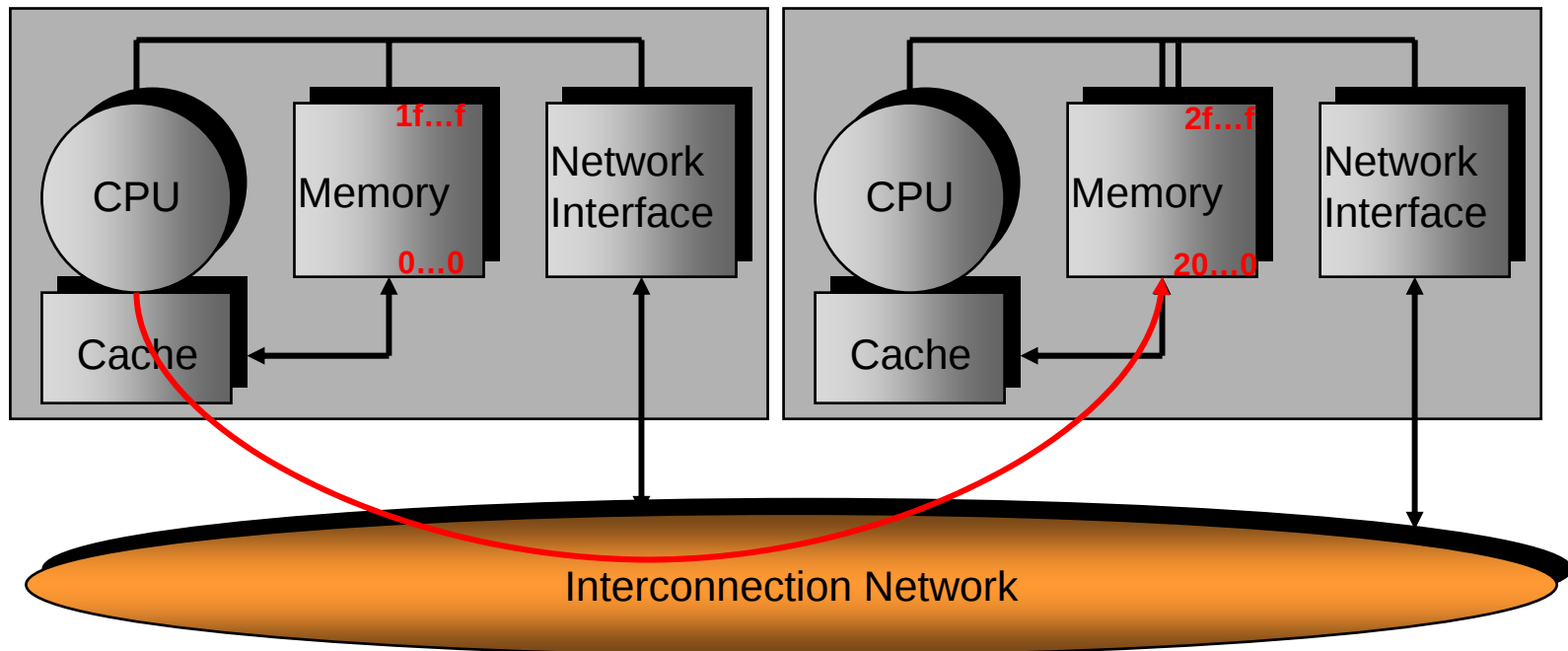
- UMA: Uniform Memory Access
- Beispiel: symmetrischer Multiprozessor (SMP)
 - Gleichberechtigter Zugriff der Prozessoren auf die Betriebsmittel



Multiprozessorsysteme

■ Multiprozessor mit verteiltem gemeinsamen Speicher

- NUMA: Non-Uniform Memory Access
- Globaler Adressraum: Zugriff auf entfernten Speicher



Multiprozessorsysteme

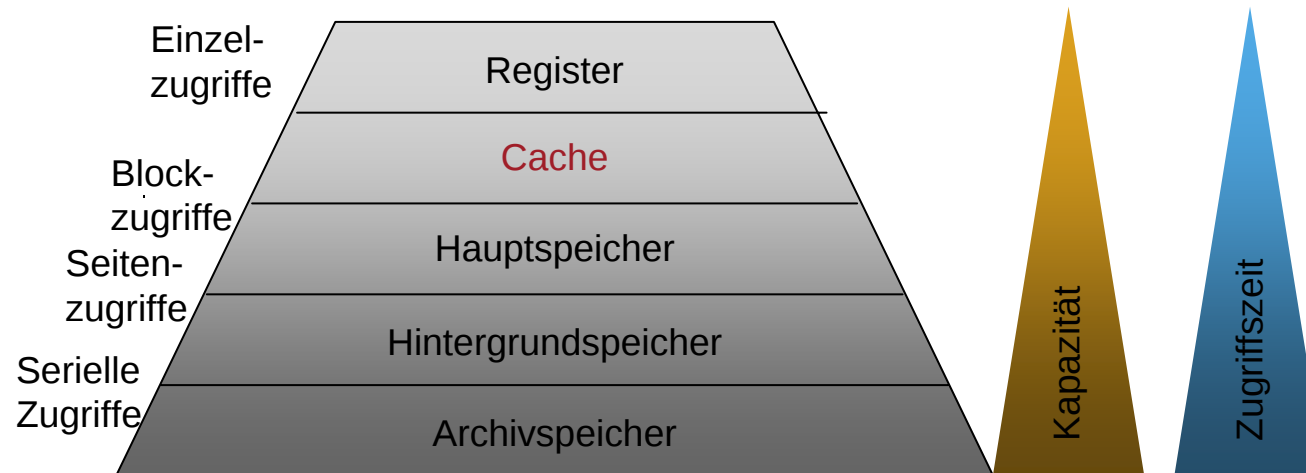
■ Gültigkeitsproblem

- wenn diese Prozessoren jeweils unabhängig voneinander auf Speicherwörter des Hauptspeichers zugreifen können.
 - Mehrere Kopien des gleichen Speicherwortes müssen miteinander in Einklang gebracht werden.
-
- Eine Cache-Speicherverwaltung heißt cache-kohärent, wenn ein Lesezugriff immer den Wert des zeitlich letzten Schreibzugriffs auf das entsprechende Speicherwort liefert.

Wiederholung: Cache-Speicher

■ Speicherhierarchie

- Ausnützen der Lokalitätseigenschaft von Programmen
- Kompromiss zwischen Preis und Leistungsfähigkeit
- Speicherkomponenten mit unterschiedlichen Geschwindigkeiten und Kapazitäten



Wiederholung: Cache-Speicher

- **Definitionen und Eigenschaften**
- **Cache-Speicher**
 - Pufferspeicher mit schnellem Zugriff
 - Wichtigste Anwendung:
 - Pufferspeicher zwischen Hauptspeicher und Prozessor
 - Stellt die während einer Programmausführung jeweils aktuellen Hauptspeichereinhalte für Prozessorzugriffe als Kopien möglichst schnell zur Verfügung.
 - Weiteres Beispiel:
 - Translation-Lookaside Buffer (TLB)

Wiederholung: Cache-Speicher

■ Definitionen und Eigenschaften

- Cache-Speicher-Steuerung /Cache-Controller
 - Sorgt dafür, dass der Cache-Speicher in der Regel das Datum enthält, auf das der Prozessor als nächstes zugreift.
- Besondere Strategien für das
 - Laden
 - Aktualisieren und Adressieren des Inhalts
- Speicherung von Befehlen / Daten
 - Gemeinsam in einem Cache oder
 - Getrennt jeweils in einem Befehls- und Daten-Cache
 - Harvard-Architektur
- Cache-Hierarchie
 - Cache-Speicher auf mehreren Ebenen
- Inklusionseigenschaft
 - Inhalt des auf höchster Stufe stehenden Cache-Speichers auf in den Cache-Speichern auf niedrigerer Ebene:
 - $(L1\$) < (L2\$) < \dots$

Wiederholung: Cache-Speicher

■ Definitionen und Eigenschaften

- Blockrahmen, Zeile (Block-Frame, Cache-Line)
 - Datenteil
 - Folge von Speicherwörtern im Cache-Speicher
 - Blocklänge in Bytes bestimmt die Länge der Zeile
 - Anzahl der Speicherplätze in einem Blockrahmen
- Adressetikett (Adress-Tag):
 - Verbunden mit Blockrahmen
 - Enthält den gemeinsamen Adressteil der in einer Zeile gespeicherten Datenkopien.
- Statusbits
 - Valid-Bit
 - Gültigkeitsbit zeigt an, ob Cache-Zeile gültige Kopien enthält
 - Dirty-Bit
 - Zeigt für Daten-Caches an, ob die Daten in der Cache-Zeile verändert worden sind

Wiederholung: Cache-Speicher

■ Arbeitsweise

- Cache-Steuerung prüft bei Speicherzugriffen des Mikroprozessors, ob
 - der zur Speicheradresse gehörende Hauptspeichereintrag als Kopie im Cache steht (Bedingung 1) und
 - dieser Cache-Eintrag durch das Valid-Bit als gültig gekennzeichnet ist (Bedingung 2).
- Treffer (Cache-Hit):
 - Beide Bedingungen sind erfüllt
- Fehlzugriff (Cache-Miss):
 - Eine der beiden Bedingungen ist nicht erfüllt

Wiederholung: Cache-Speicher

■ Arbeitsweise

■ Cache-Fehlzugriff

■ Aktionen bei Lesezugriffen (read-miss)

- Lesen des Datums aus den niedrigeren Ebenen oder dem Hauptspeicher und Laden des Cache-Speichers
- Kennzeichnen der Cache-Eintrages als gültig (Setzen des Valid-Bits)
- Speichern der Adressinformation im Adressteil des Cache-Speichers

■ Aktionen bei Schreibzugriffen (write-miss):

- Aktualisierungsstrategie bestimmt, ob
- der entsprechende Block in Cache geladen und dann mit dem zu schreibenden Datum aktualisiert wird, oder ob
- nur der Hauptspeicher aktualisiert wird und der Cache unverändert bleibt.

■ Cache-Treffer

- Zugriff erfolgt auf Cache

Wiederholung: Cache-Speicher

- **Cache-Organisation**
- Wohin wird ein Block im Cache geladen?
 - Direkte Abbildung

Hauptspeicher

Block B_j , $j = 0, 1, \dots, n-1$

Kapazität: $n * b = 2^{s+w}$ Wörter

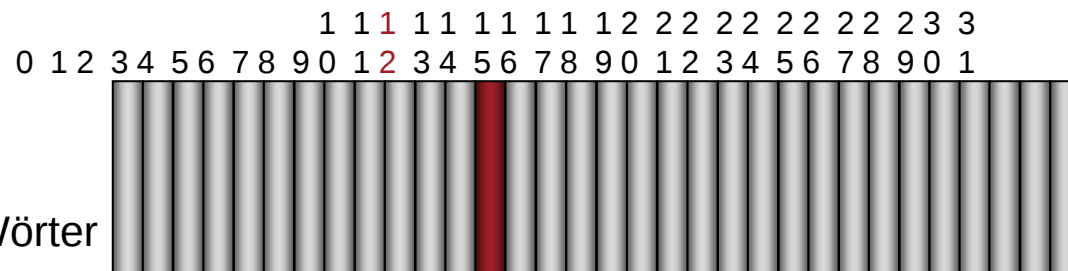
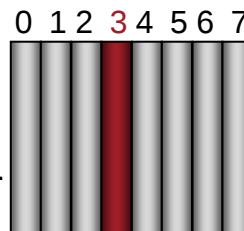


Abbildung von $\{B_j\}$ nach $\{Z_i\}$

Cache

Zeile Z_i , $i = 0, 1, \dots, m-1$

Kapazität: $m * b = 2^{r+w}$ Wörter



Legende:

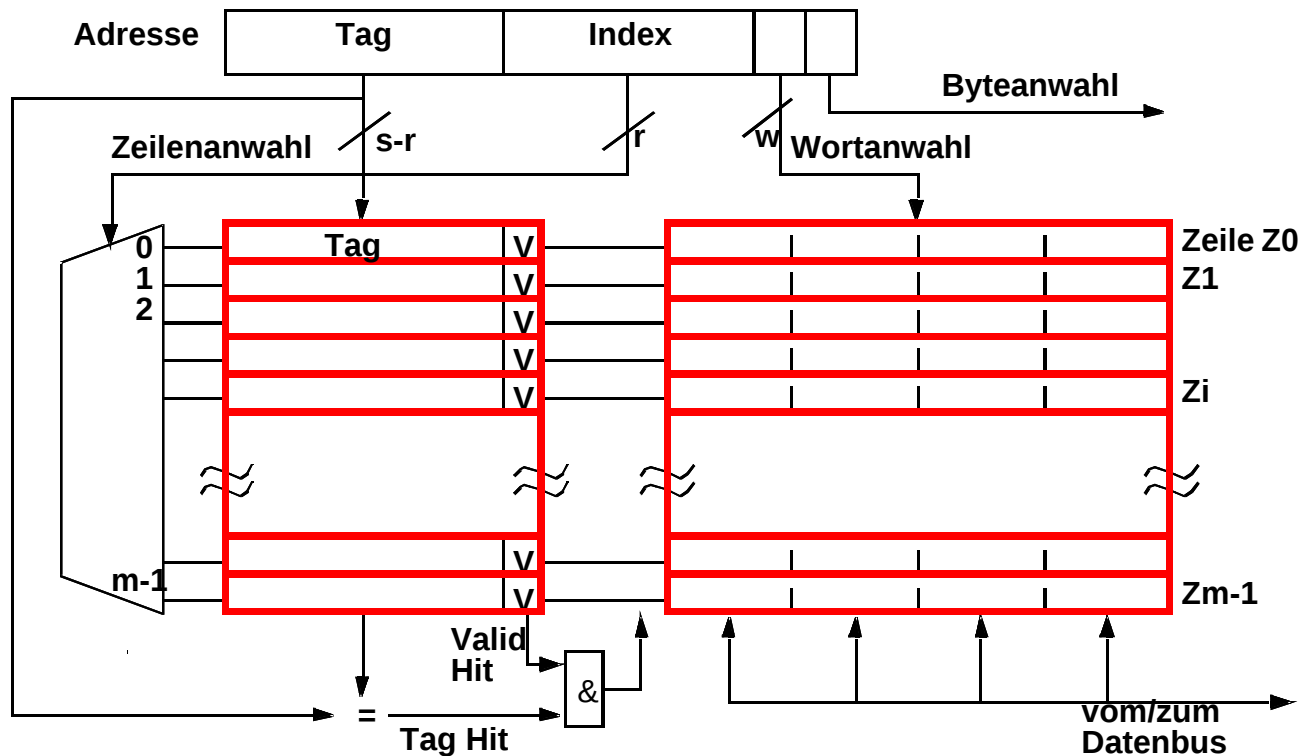
$n \gg m$, $n = 2^s$, $m = 2^r$

Jeder Block enthält b Wörter mit $b = 2^w$

Wiederholung: Cache-Speicher

Cache-Organisation

Direct-Mapped Cache



Wiederholung: Cache-Speicher

■ Cache-Organisation

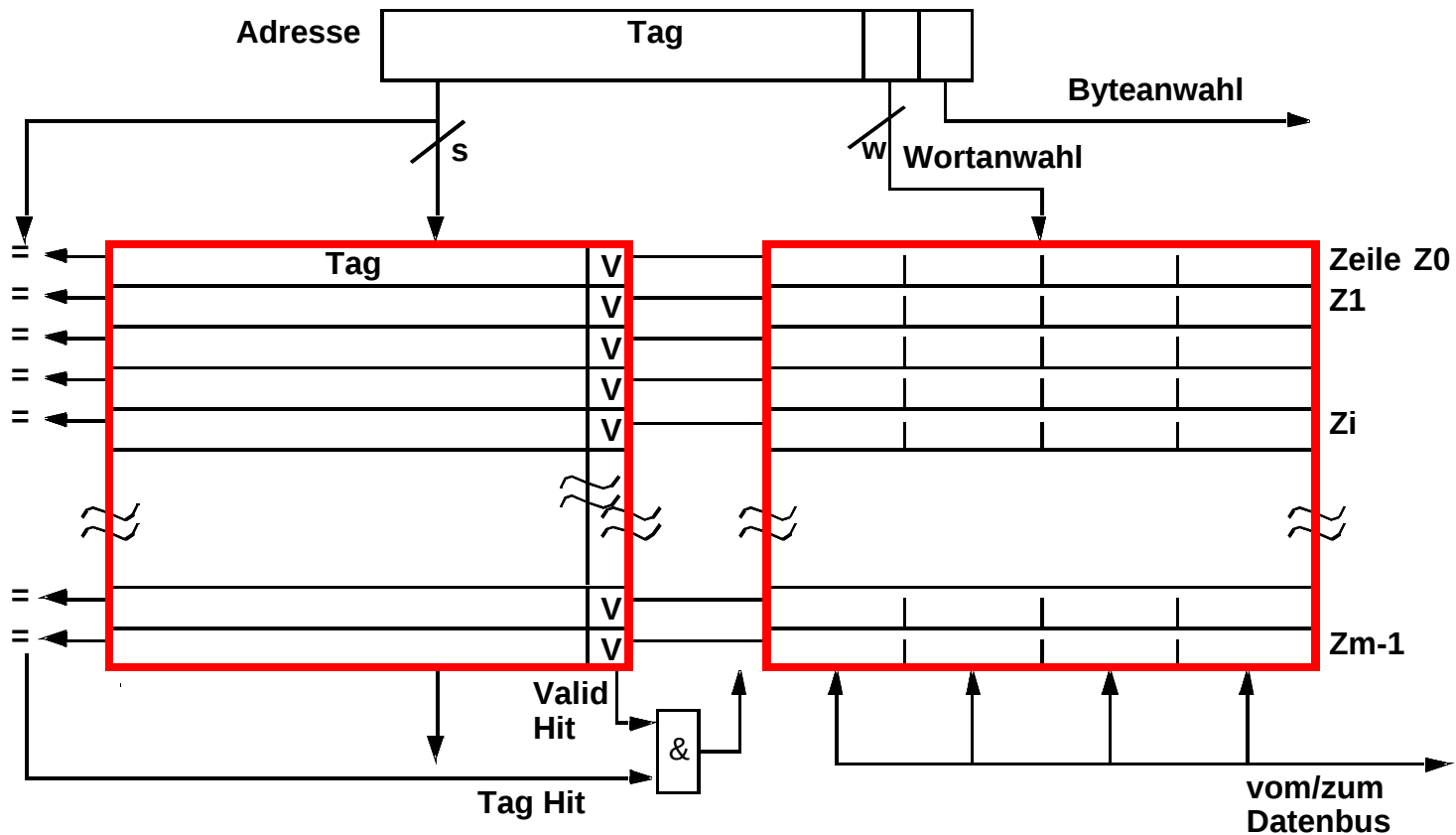
■ Vollassoziativer Speicher

- jeder Block des Hauptspeichers kann auf jede Zeile des Cache-Speichers abgebildet werden (Flexibilität)
- Ersetzungsstrategie gibt vor, welche Zeile beim Laden ersetzt werden soll (z.B. Least-Recently-Used)
- hoher Hardware-Aufwand (Anzahl Vergleiche = Anzahl Zeilen)

Wiederholung: Cache-Speicher

Cache-Organisation

Vollassoziativer Speicher



Wiederholung: Cache-Speicher

■ Cache-Organisation

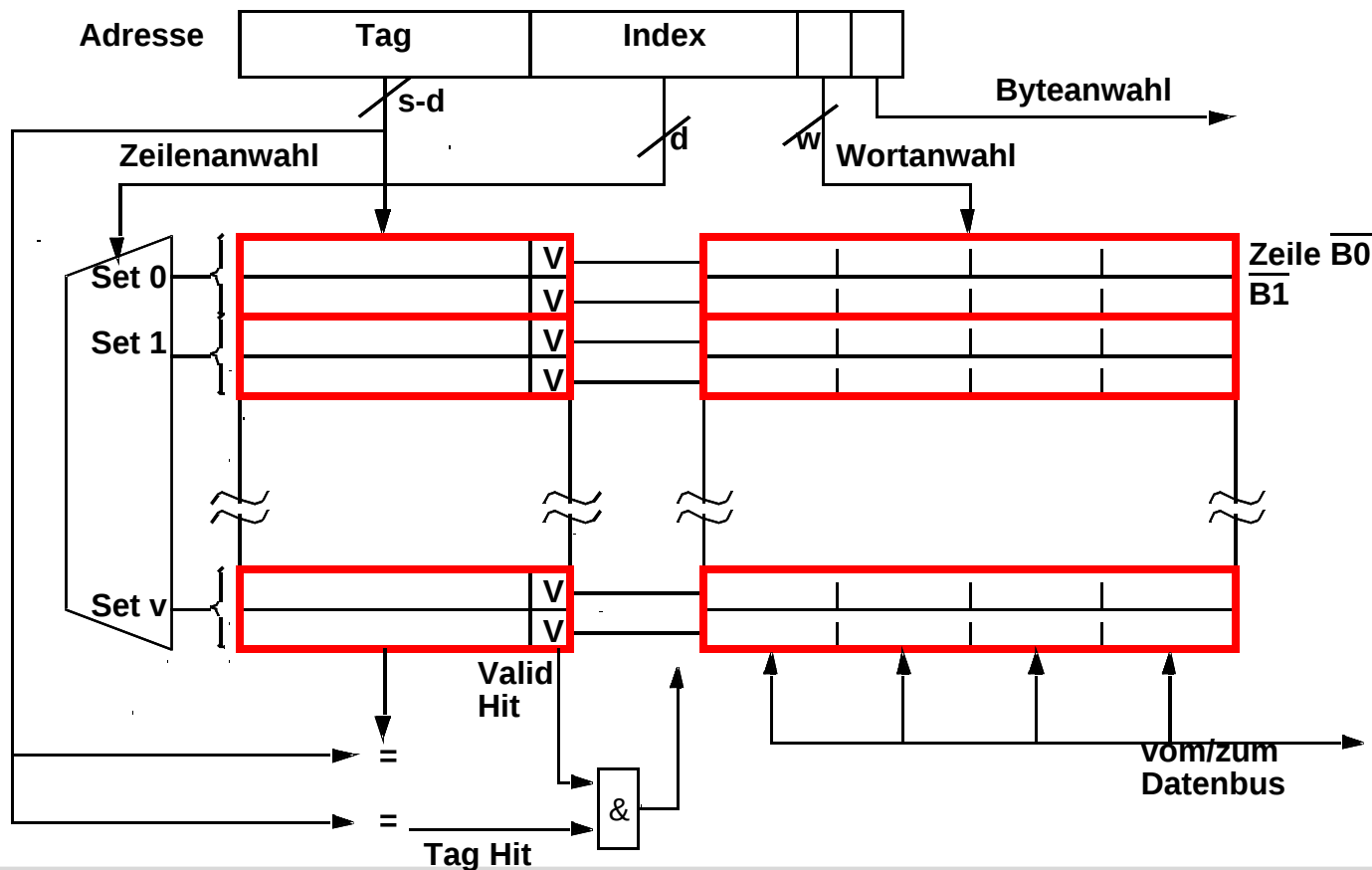
■ k-fach satzassoziativer Speicher

- Kompromiss zwischen Direct-Mapped- und vollassoziativem Cache;
 - Zusammenfassen von k Zeilen zu einer Menge (set)
 - Aufteilen der m Cache-Zeilen in $v = m/k$ Mengen zu je k Zeilen;
 - jede Menge wird über eine d Bit breite Nummer identifiziert: $2^d = v$;
 - ein Block B_j kann in eine der verfügbaren Zeilen Z_f in einer Menge S_i abgebildet werden:
-
- $B_j \rightarrow Z_f \in S_i$, mit $j(\text{mod } v) = i$.

Wiederholung: Cache-Speicher

Cache-Organisation

k-fach satzassoziativer Speicher



Wiederholung: Cache-Speicher

■ Aktualisierungsstrategie

■ Befehls-Caches:

- Prozessor führt nur Lesezugriffe durch
- Im Cache befindet sich immer die identische Kopie des Hauptspeichers

■ Daten-Cache:

- Prozessor führt auch Schreibzugriffe durch.
- Im Hauptspeicher können sich veraltete Daten befinden
- Aktualisierungsstrategie
 - Bei Schreibzugriff (write hit)
 - Aktualisieren des Cache-Speichers oder des Hauptspeichers oder beide (write hit):
 - Lesezugriffe dürfen nicht auf inzwischen veraltete Daten gehen;

Wiederholung: Cache-Speicher

■ Aktualisierungsstrategie für Caches mit je einem Valid- und einem Dirty-Bit

Cache-Zugriff	Write-Through No-Write Alloc.	Write-Through Write-Alloc.	Copy-Back
Read-Hit	Cache-Datum --> CPU	Cache-Datum --> CPU	Cache-Datum --> CPU
Read-Miss	HS-Block, Tag --> Cache HS-Datum --> CPU 1 --> V	HS-Block, Tag --> Cache HS-Datum --> CPU 1 --> V	Cache-Zeile --> HS HS-Block, Tag --> Cache HS-Datum --> CPU 1 --> V, 0 --> D
Write-Hit	CPU-Datum --> Cache, HS	CPU-Datum --> Cache, HS	CPU-Datum --> Cache 1 --> D
Write-Miss	CPU-Datum --> HS	HS-Block, Tag --> Cache, 1 --> V CPU-Datum --> Cache, HS	Cache-Zeile --> HS HS-Block, Tag --> Cache 1 --> V CPU-Datum --> Cache 1 --> D