Rechnerstrukturen

Vorlesung im Sommersemester 2006

Prof. Dr. Wolfgang Karl

Universität Karlsruhe (TH)

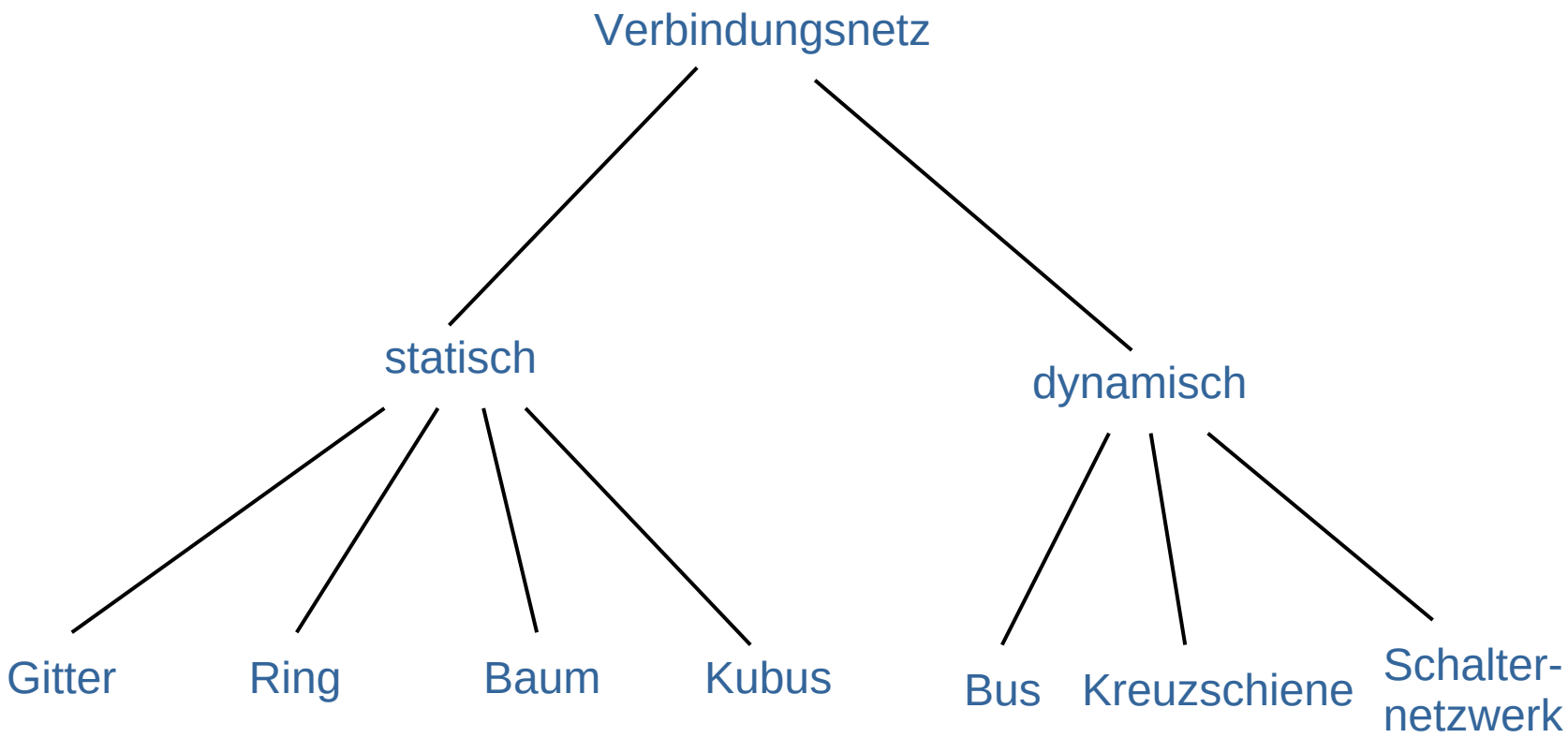
Fakultät für Informatik

Institut für Technische Informatik

- **Kapitel 3: Multiprozessoren – Parallelismus auf Prozess/Thread-Ebene**

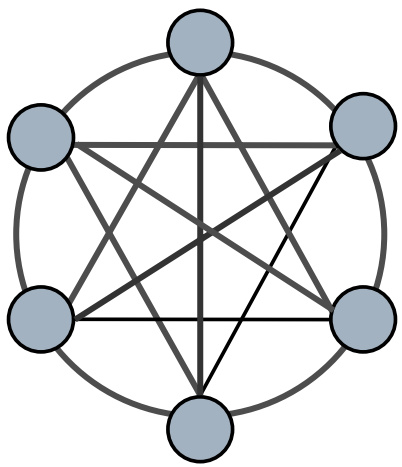
3.4: Verbindungsstrukturen

- Topologie: Klassifizierung



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

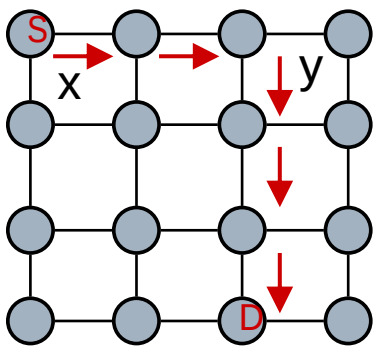
- Statische Verbindungsnetze
  - Vollständige Verbindung
    - Jeder Knoten ist mit jedem anderen Knoten verbunden
    - Höchste Leistungsfähigkeit
      - Arbeitet für alle Anwendungen mit allen Arten von Kommunikationsmustern effizient
    - Aber: nicht praktikabel in Parallelrechnern
      - Netzwerkkosten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



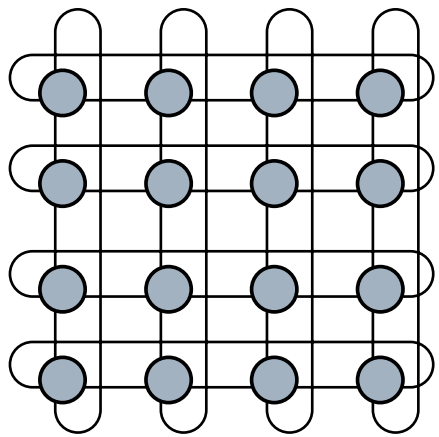
- Statische Verbindungsnetze
  - Gitterstrukturen
    - 1-dimensionales Gitter (lineares Feld, Kette)
      - Verbindet N Knoten mit (N-1) Verbindungen
      - Endknoten haben den Grad 1, Zwischenknoten den Grad 2 und sind mit benachbarten Knoten verbunden
      - Diameter r ist N-1
      - Disjunkte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

- 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 $k(\sqrt[k]{N}-1)$

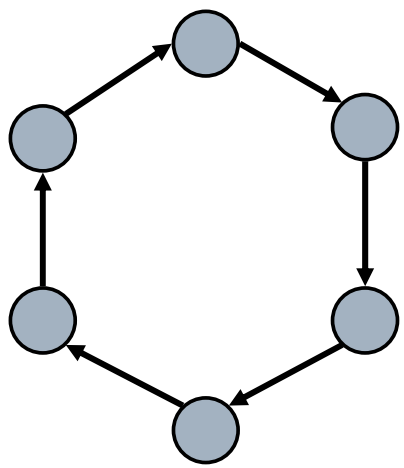
2-dim. Gitter



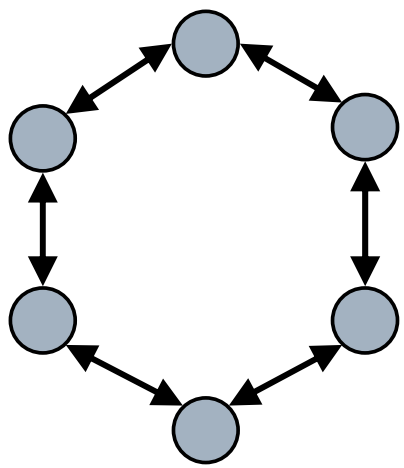
2-dim. Torus



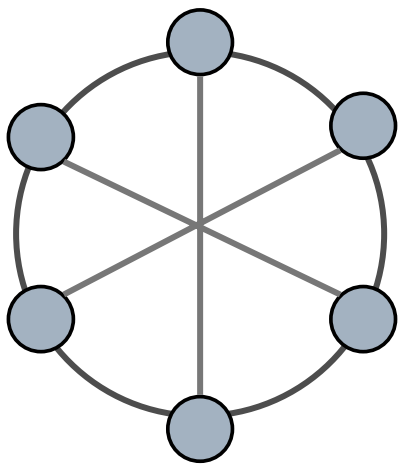
- Statische Verbindungsnetze
  - Ring
    - Erhält man, wenn man die Endknoten eines linearen Feldes miteinander verbindet
    - Unidirektionaler Ring mit N Knoten
      - Nachrichten werden in einer Richtung vom Quellknoten zum Zielknoten verschickt
      - Diameter r ist N-1
      - Bei Ausfall einer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



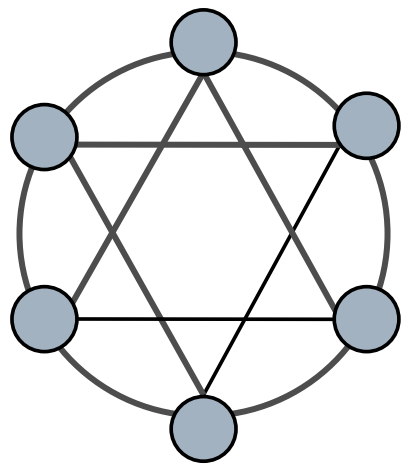
- Statische Verbindungsnetze
  - Ring
    - Bidirektionaler Ring mit N Knoten
      - symmetrisches Netzwerk
      - Der längste Pfad, den eine Nachricht nehmen muss, ist nicht länger als N/2
      - Bei Ausfall einer Verbindung bricht die Kommunikation noch nicht zusammen, während zwei Ausfälle von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



- Statische Verbindungsnetze
  - Ring
    - Chordialer Ring
      - Hinzufügen redundanter Verbindungen
        - » erhöht Fehlertoleranzeigenschaft des Verbindungsnetzwerkes
        - » Höherer Knotengrad und kleinerer Diameter gegenüber Ring

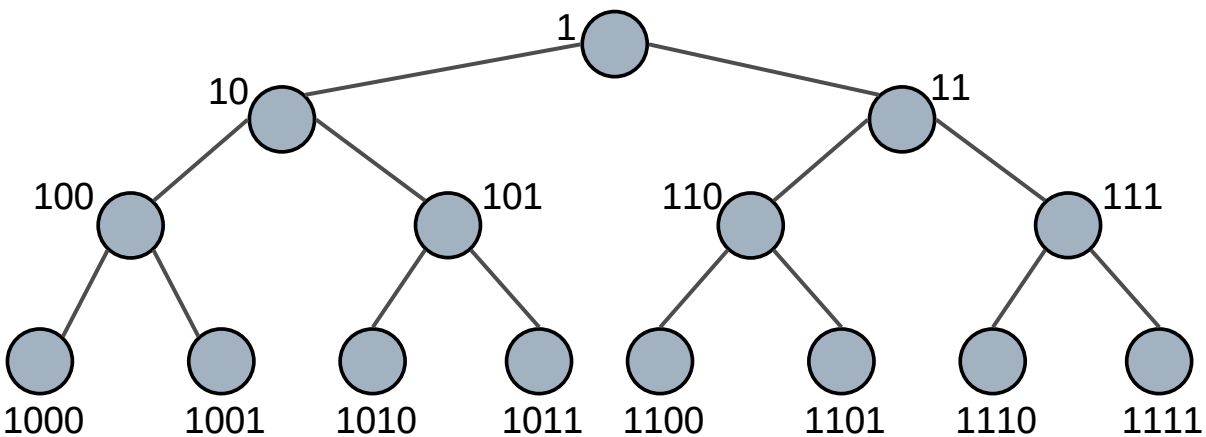


Chordaler Ring mit Knotengrad 3

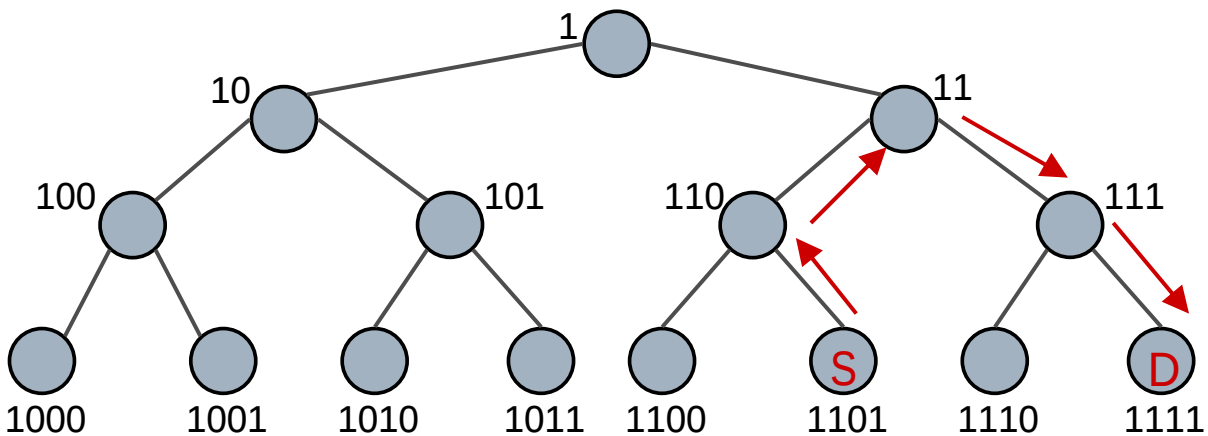


Chordaler Ring mit Knotengrad 4

- 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

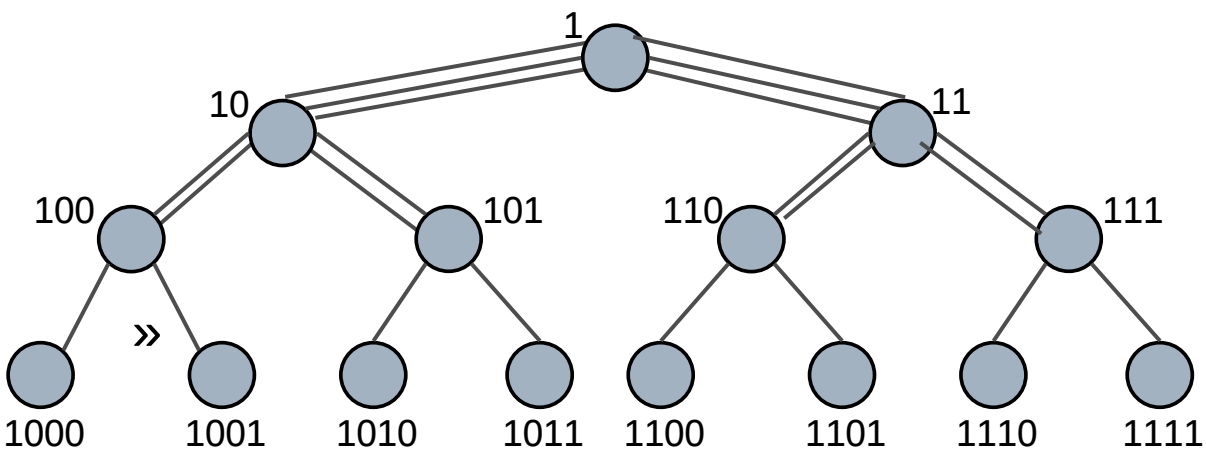


- Statische Verbindungsnetze
  - Baumstrukturen
    - Routing:
    - Finde gemeinsamen Elternknoten P von S und P
    - Gehe von S nach P und von P nach D



- Statische Verbindungsnetze
  - Baumstrukturen
    - Routing:
      - **Die Binärdarstellung der Adresse eines Quellknotens S auf Ebene i sei $S_iS_{i-1} \ldots S_1$ und die der Adresse des Zielknotens D auf Ebene j sei $D_jD_{j-1}\ldots D_1$**
      - **Finde die gemeinsamen höchstwertigen Bits von S und D, so dass die Adresse des Elternknotens P gleich $D_jD_{j-1}\ldots D_x = S_iS_{i-1} \ldots S_{(i-j+x)}$ 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}

- Statische Verbindungsnetze
  - Baumstrukturen
    - Fat Tree:
      - Lösung des Blockierungsproblems in Richtung Wurzel
      - Kommunikationskanäle werden größer, je näher man sich der Wurzel nähert

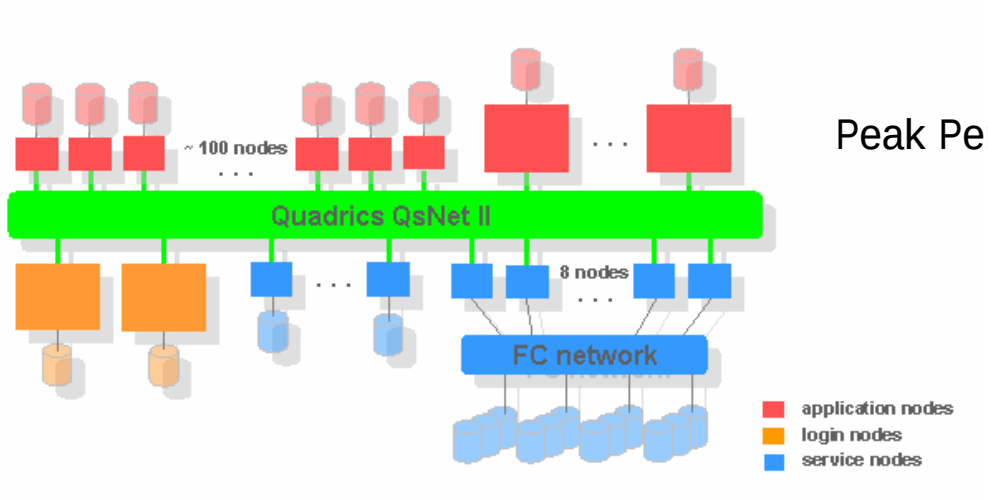


- Statische Verbindungsnetze
  - Baumstrukturen
    - Dynamic Fat Tree:
      - Interne Knoten sind Schalter
      - Beispiel: Quadrics QSnet
        - » Los Alamos Lab (LANL): ASCI Q

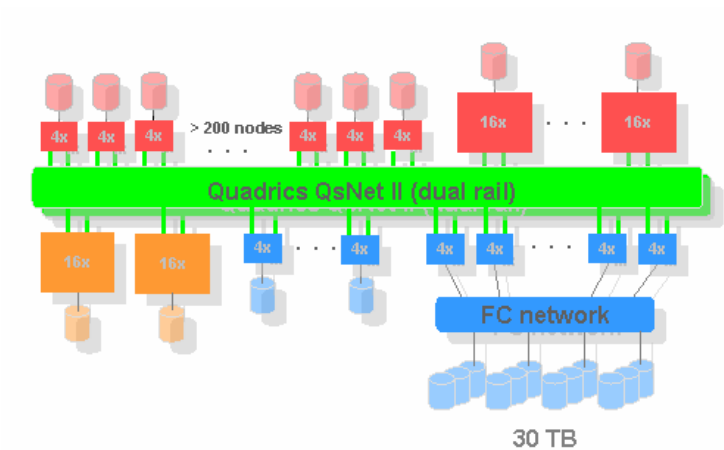
Images Courtesy of LANL, LLNL,PNNL, PSC, CEA
Quelle: http://www.c3.lanl.gov/~fabrizio/quadrics.html

- Statische Verbindungsnetze
  - Los Alamos Lab (LANL): ASCI Q AlphaServer SC45, 1.25 GHz
    - 8125 Prozessoren Alpha 1250 MHz
    - Verbindungsnetzwerk: Quadrics
    - Linpack: 13880 GFLOPS (TOP500: 12, 06/2005)

- Statische Verbindungsnetze
  - Baumstrukturen
    - Fat Tree:
      - Beispiel: Quadrics QSnet
        - » Universität Karlsruhe (TH), Landeshöchstleitungsrechner, HP XC 6000

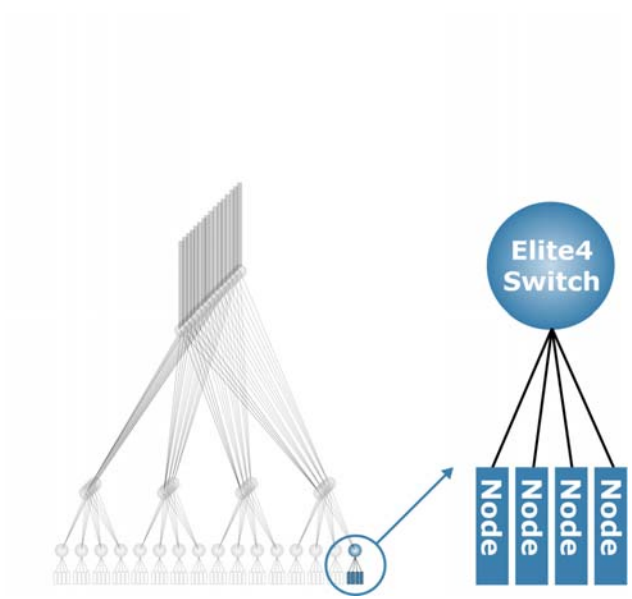
- Statische Verbindungsnetze
  - Universität Karlsruhe (TH), Landeshöchstleitungsrechner,
    - HP XC 6000: Konfiguration
      - 108 Knoten mit jeweils 2 Intel Itanium2 Prozessoren (1,5 GHz) ,12 GB Hauptspeicher pro Knoten und 146 GB lokalem Plattenplatz,
      - 12 Knoten mit jeweils 8 Intel Itanium2 Prozessoren (1,6 GHz), 64 GB Hauptspeicher pro Knoten und ca. 500 GB lokalem Plattenplatz,
      - 8 2-Wege Fileserver-Knoten basierend auf Xeon Prozessoren mit angeschlossenen Platten in der Größe von 10 TB und
      - Quadrics QsNet II Interconnect
      - http://www.rz.uni-karlsruhe.de/ssc/hpxc.php
      - 

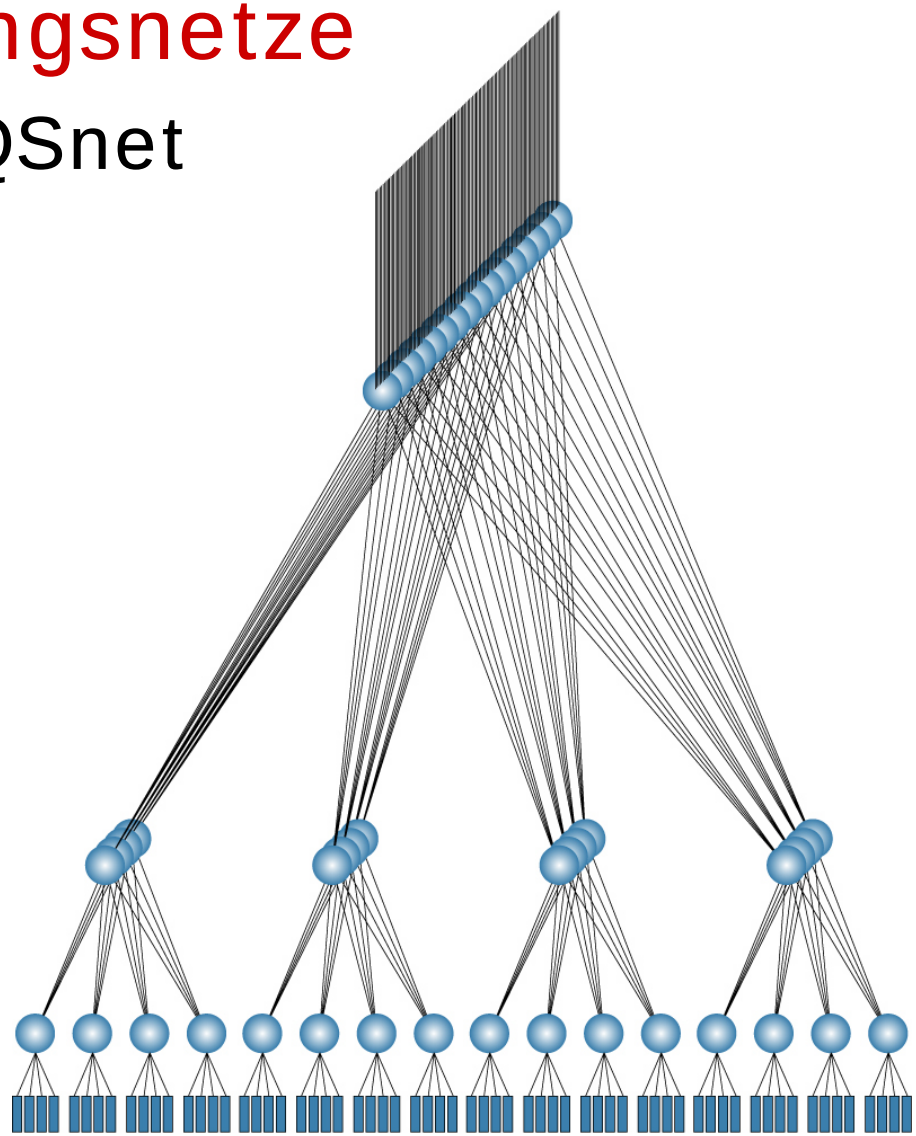
Peak Performance: 2,2 TFLOPS

- Statische Verbindungsnetze
  - Universität Karlsruhe (TH), Landeshöchstleitungsrechner, HP XC 6000
    - Im 1. Quartal 2006 wird das Produktionssystem um 218 4-Wege Knoten erweitert mit
      - 2 "dual core" Intel Itanium2 Prozessoren,
      - 24 GB Hauptspeicher pro Knoten,
      - 146 GB lokalem Plattenplatz,
      - dual rail Quadrics QSNet II Interconnect und
      - 30 TB globaler Plattenplatz.
  - 

Peak Performance: 11 TFLOPS

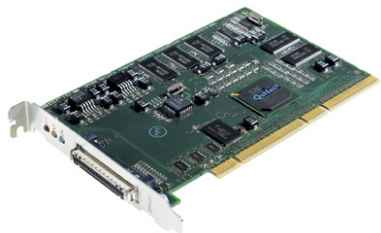
- Statische Verbindungsnetze
  - Beispiel: Quadrics QSnet





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

- Statische Verbindungsnetze
  - Beispiel: Quadrics QSnet
    - Elan 4 network interface card:

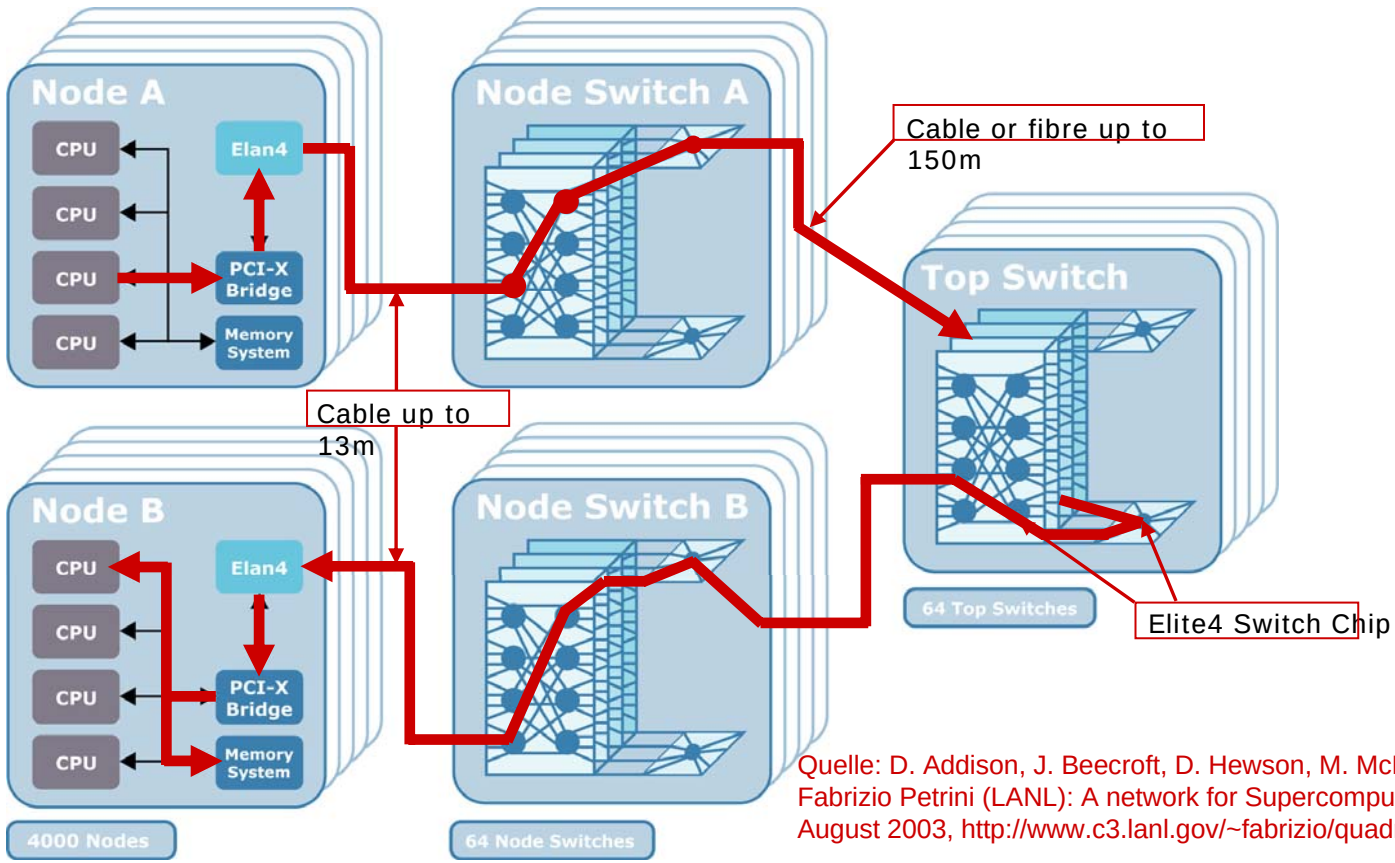


    - Elite 4 Switch Component:

    - QsNet II Switch:

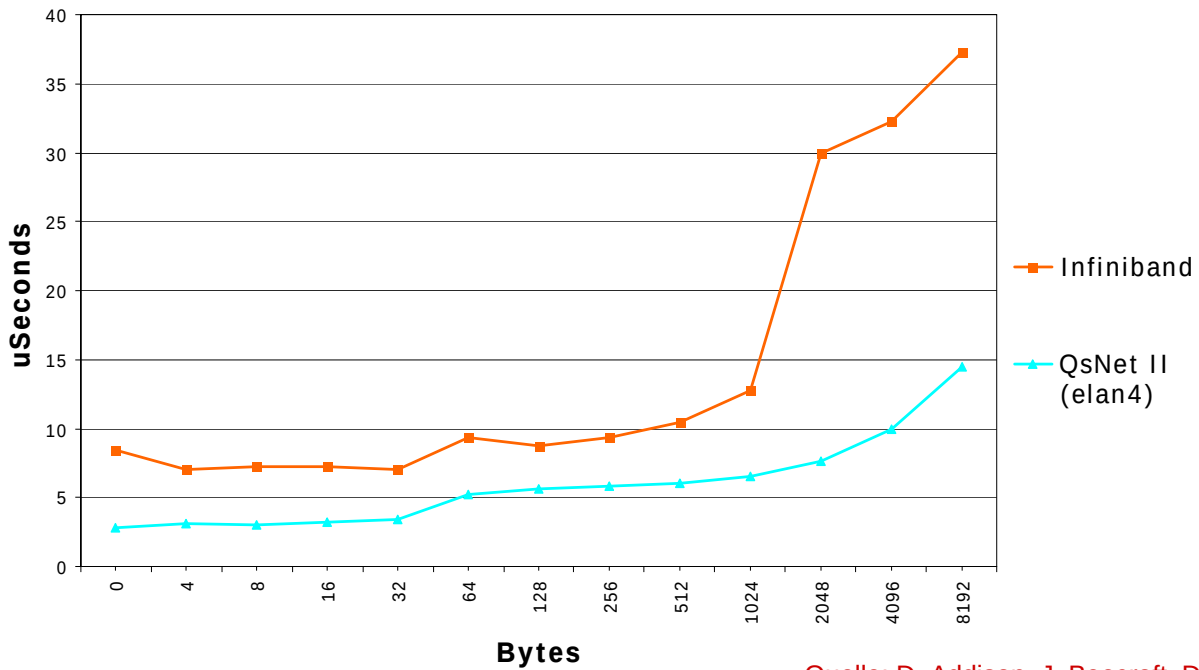
Quelle: D. Addison, J. Beecroft, D. Hewson, M. McLaren (Quadrics Ltd.), Fabrizio Petrini (LANL): A network for Supercomputing Applications. Hot Chips, August 2003, http://www.c3.lanl.gov/~fabrizio/quadrics.html

- Statische Verbindungsnetze
  - Beispiel: Quadrics QSnet



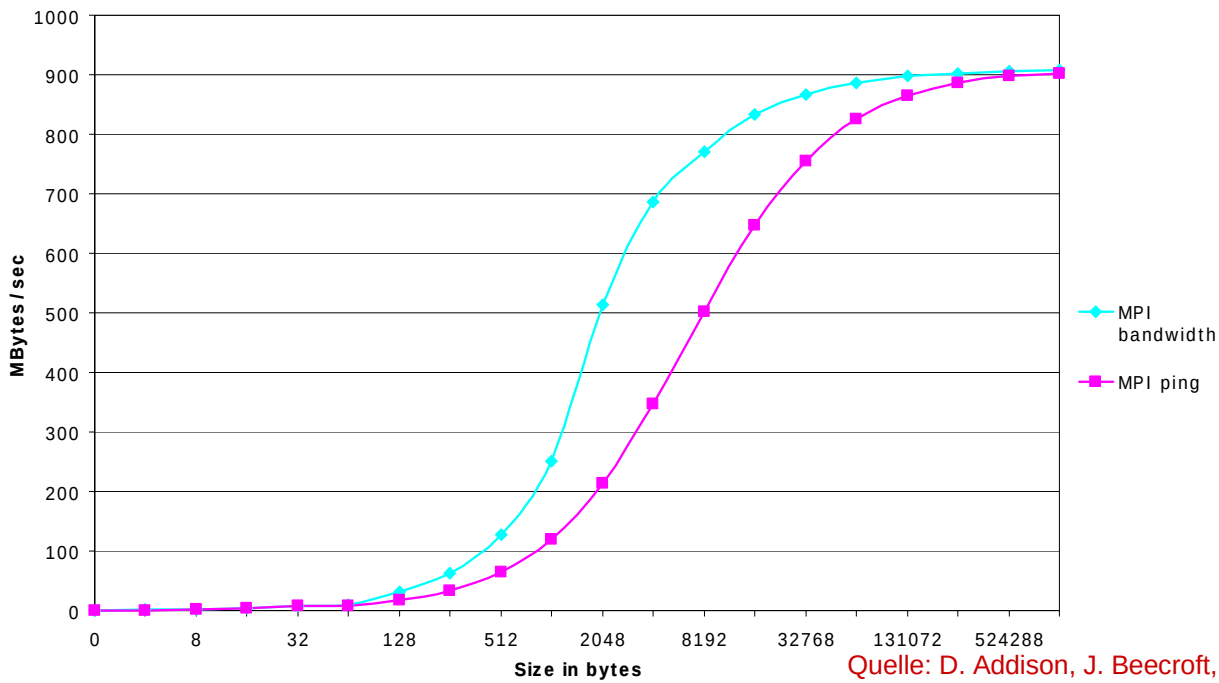
Quelle: D. Addison, J. Beecroft, D. Hewson, M. McLaren (Quadrics Ltd.), Fabrizio Petrini (LANL): A network for Supercomputing Applications. Hot Chips, August 2003, http://www.c3.lanl.gov/~fabrizio/quadrics.html

- Statische Verbindungsnetze
  - Beispiel: Quadrics QSnet:
    - MPI short message latency



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

- Statische Verbindungsnetze
  - Beispiel: Quadrics QSnet:
    - MPI Bandwidth



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

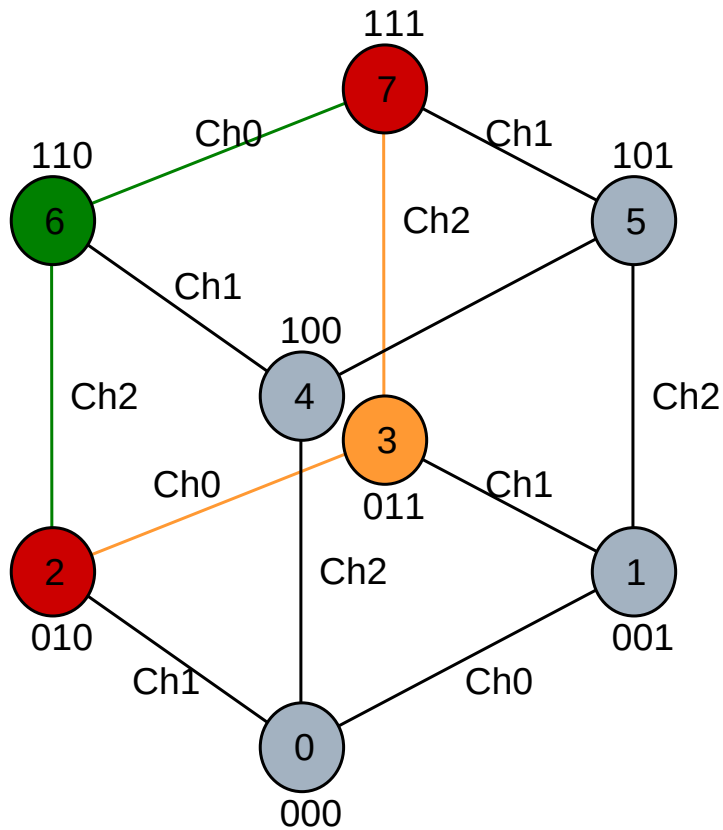
- Statische Verbindungsnetze
  - K-ärer n-Kubus (Cubes, Würfel)
    - Allgemeine Form eines Kubus-Verbindungsnetzwerkes
    - Ringe, Gitter, oder Hyperkubi sind topologisch isomorph zu einer Familie von K-ären n-Kubus Netzwerken
      - n ist die Dimension
      - Radius K ist die Anzahl der Knoten, 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 $a_0,a_1,\ldots,a_{n-1}$ 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 $a_0,a_1,\ldots,a_{n-1}$ kann erreicht werden mit $a_0,a_1,\ldots,a_{(i \pm 1)} \bmod k \ldots a_{n-1}$.
    - Knotengrad is 2n und der Diameter ist $n\left\lfloor \frac{k}{2} \right\rfloor$

- Statische Verbindungsnetzwerke:
  - 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.

- Statische Verbindungsnetzwerke:
  - 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.

- Statische Verbindungsnetzwerke:
  - Hyperkubus
    - e-Cube Routing Algorithmus:
      - „messages are routed in increasingly higher dimensions of channels until the destination is reached“
      - Dimension eines Kanals = Bitposition von (Knoten# XOR Knoten#)

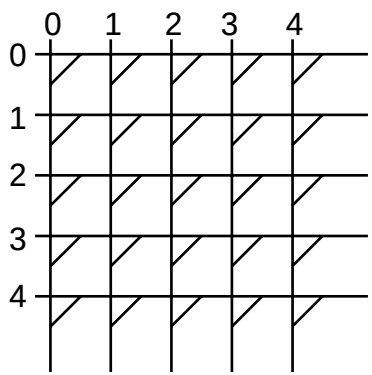
- Statische Verbindungsnetzwerke:
  - Hyperkubus: e-cube Routing
  - Beispiel:
    - A = (010) und B = (111)
    - W=A XOR B = 101
    - (010) → (011) → (111),
    - (010) → (110) → (111)



- 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

- 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

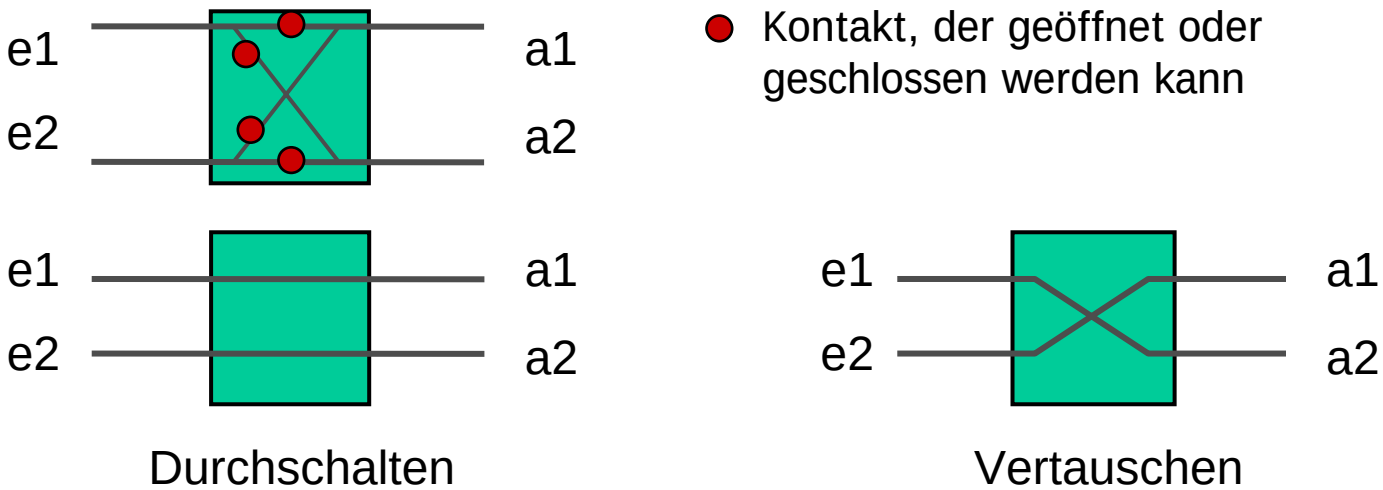
- Dynamische Verbindungsnetzwerke:
  - Kreuzschienenverteiler (Crossbar)
    - Vollständig vernetztes Verbindungswerk mit allen möglichen Permutationen der N Einheiten, die über das Netzwerk verbunden werden



- 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.

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

➔ Schalternetzwerke



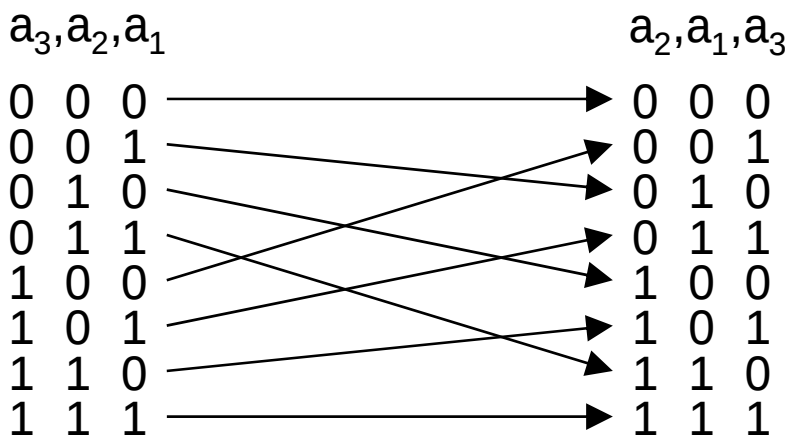
- Dynamische Verbindungsnetzwerke:
  - Mehrstufige Verbindungsnetzwerke (Schalternetzwerke, Permutationsnetzwerke)
    - Kompromiss zwischen der niedrigeren Leistungsfähigkeit von Bussen und hohem Hardware-Aufwand von Kreuzschienenverteilern
    - Oft 2 x 2 Kreuzschienenverteiler (Schalterelement) als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

- 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

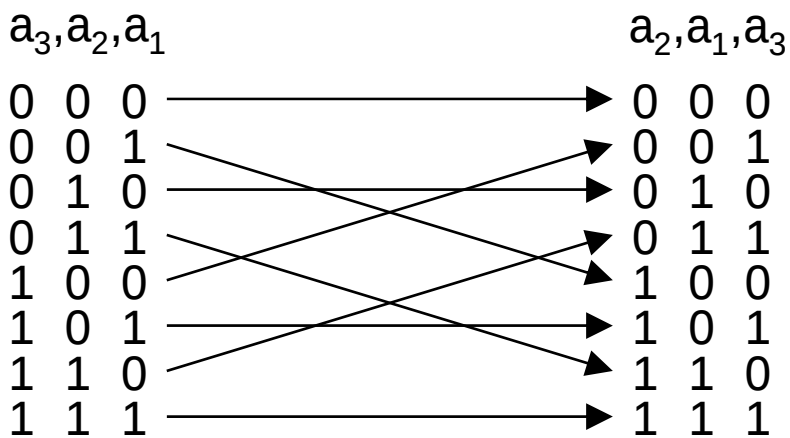
- Dynamische Verbindungsnetzwerke:
  - Permutationsnetze
    - reguläre Permutationsnetzwerke:
      - p Eingänge, p Ausgänge und k Stufen mit jeweils p/2 Zweierschaltern, wobei die Zahl p normalerweise eine Zweierpotenz ist.
    - Irreguläre Permutationsnetzwerke
      - weisen gegenüber der vollen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

- Dynamische Verbindungsnetzwerke:
  - Permutationen
    - eineindeutige (bijektive) Zuordnungen von Eingängen zu Ausgängen
    - Man stellt die Eingänge als binären Zahlenwert dar, d.h., man nummeriert die Eingänge beginnend mit 0 bis zum (2n-1)-ten Eingang durch.
    - Auf diese Weise ordnet man also jedem Eingang eine Art Adresse zu.
    - Die Permutation lässt sich nun durch eine Bitmanipulation dieser Adresse darstellen, so dass am Ausgang neue Bitmuster entstehen.

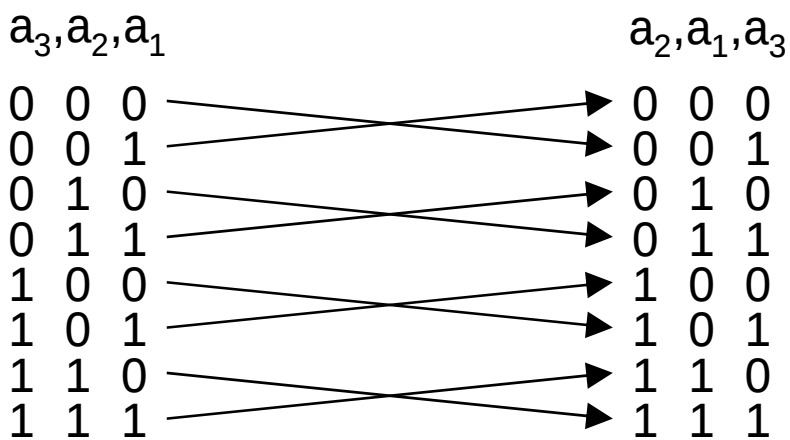
- Dynamische Verbindungsnetzwerke:
  - Permutationen
    - Mischpermutation M (Perfect Shuffle):
      - $M(a_n,a_{n-1},...a_2,a_1)=(a_{n-1},...a_2,a_1,a_n)$



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

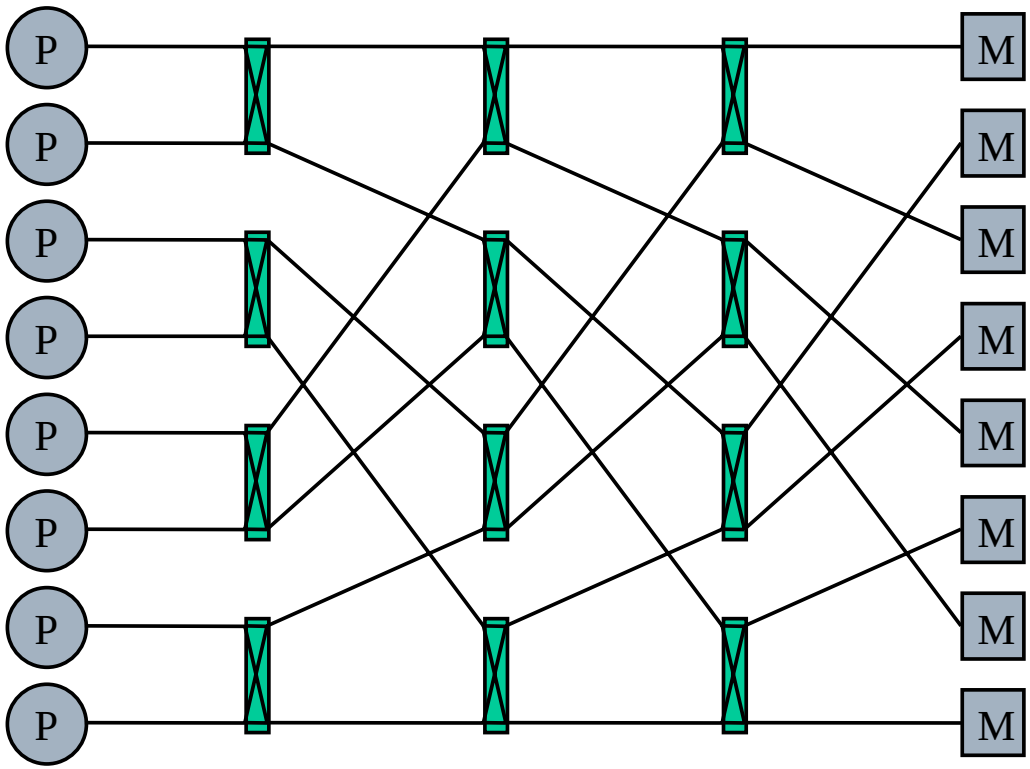


- Dynamische Verbindungsnetzwerke:
  - Permutationen
    - Tauschpermutation T (Butterfly):
      - Negation des niedrigwertigen Bits
      - $T(a_n,a_{n-1},...a_2,a_1)=(a_n,a_{n-1}...a_2,a_1)$

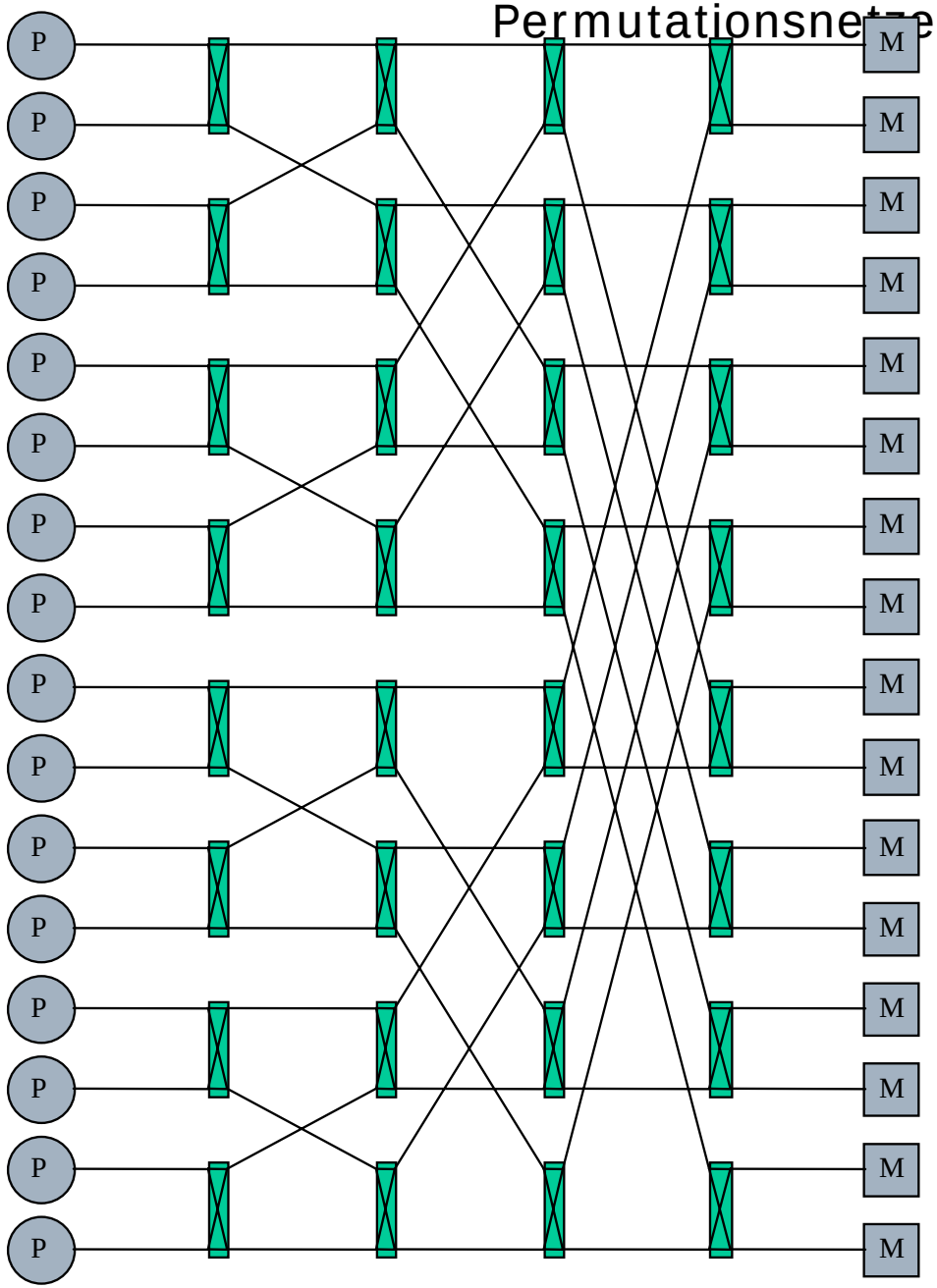


- Dynamische Verbindungsnetzwerke:
  - Omega-Netzwerk
    - Das Netzwerk für p=2n Ein-/Ausgänge umfasst n = ld p Stufen von Zweierschaltern, die untereinander jeweils nach dem Grundmuster der Mischpermutation verknüpft sind.
    - Gesamtzahl der Zweierschalter in einem Omega-Netzwerk mit p = 2n Ein-/Ausgängen beträgt
      (p/2) * ld p
    - Nicht blockierungsfrei

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



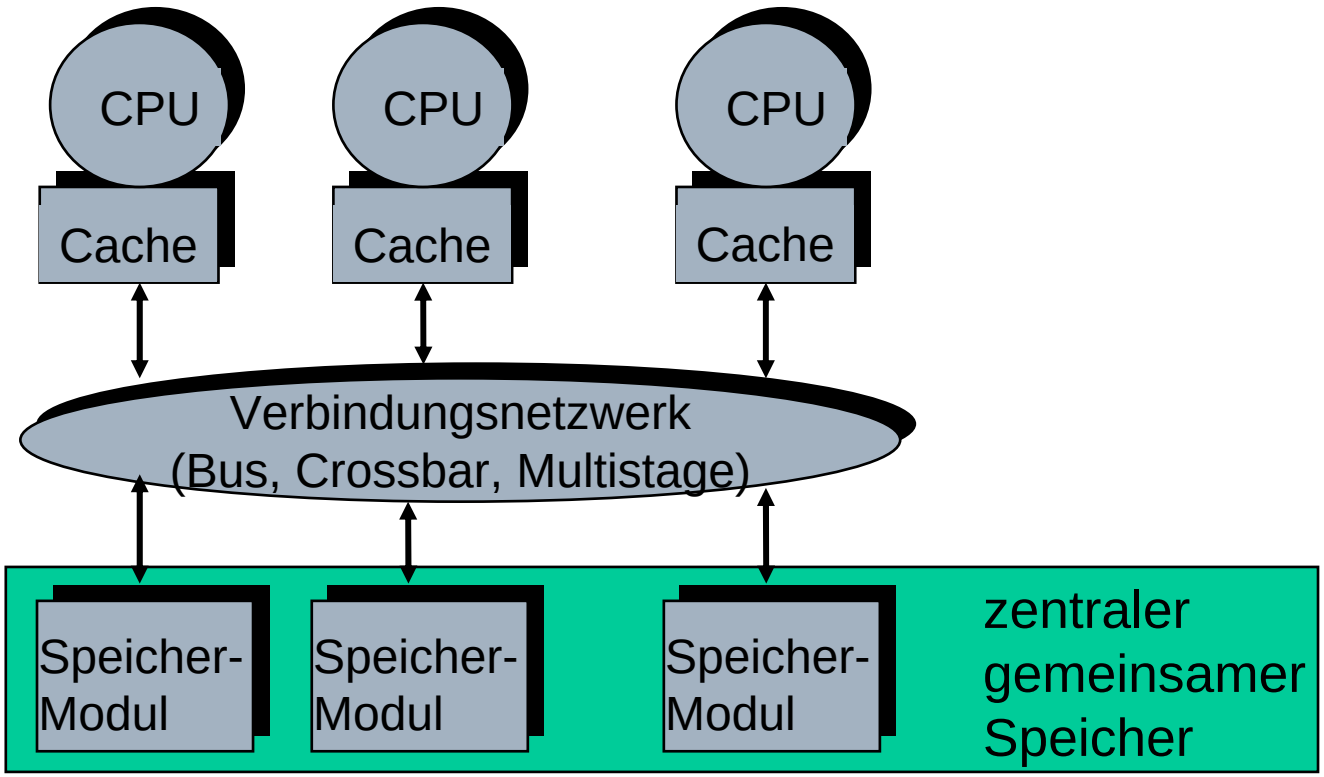
Speichergekoppeltes Switching-Banyan-Netzwerk mit 16 Ein- und 16 Ausgängen



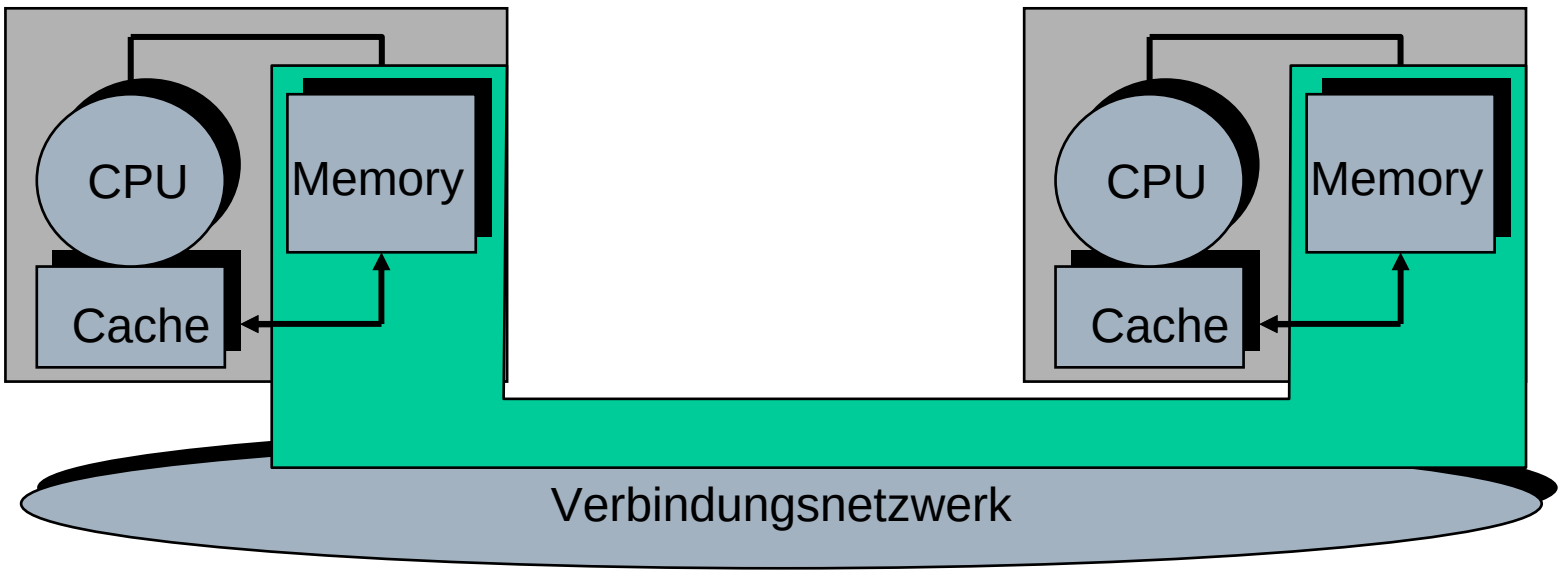
- **Kapitel 3: Multiprozessoren – Parallelismus auf Prozess/Thread-Ebene**

3.4: Multiprozessoren mit gemeinsamem Speicher

- Multiprozessoren mit gemeinsamem Speicher

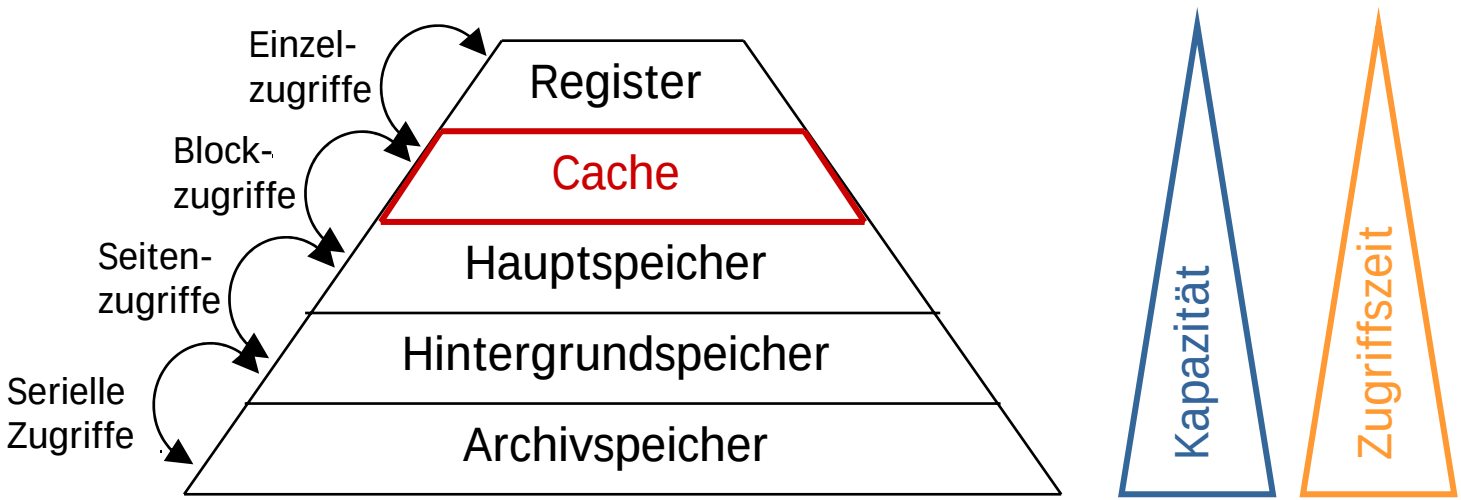


- Multiprozessoren mit verteiltem gemeinsamem Speicher



- 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.

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



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

- Definitionen und Eigenschaften
  - Cache-Speicher-Verwaltung
    - Sorgt dafür, dass der Cache-Speicher in der Regel das Datum enthält, auf das der Prozessor als nächstes zugreift.
  - Cache-Controller
    - kopiert automatisch die Daten in den Cache, auf die der Prozessor zugreift.

Wiederholung: Cache-Speicher

- Eigenschaften
  - Geringere Kapazität im Vergleich zum Hauptspeicher
  - Besondere Strategien für das
    - Laden
    - Aktualisieren und
    - Adressieren des Inhalts
  - Speicherung von Befehlen
    - Gemeinsam in einem Cache oder
    - Getrennt jeweils in einem Befehls- und Daten-Cache
      - Harvard-Architektur

Wiederholung: Cache-Speicher

- Eigenschaften
  - Blockrahmen (Block-Frame, Cache-Line)
    - Reihe von Speicherplätzen im Cache-Speicher
  - Blocklänge
    - Anzahl der Speicherplätze in einem Blockrahmen
  - Adressetikett (Adress-Tag):
    - Verbunden mit Blockrahmen
    - Enthält den gemeinsamen Adressteil der in einem Block gespeicherten Datenkopien.
  - Statusbits
  - Datenteil

- Arbeitsweise
  - Cache-Steuerung prüft bei Speicherzugriffen des Mikroprozessors, ob
    - der zur Speicheradresse gehörende Hauptspeicherinhalt als Kopie im Cache steht (Bedingung 1) und
    - dieser Cache-Eintrag durch das Valid-Bit als gültig gekennzeichnet ist (Bedingung 2).
    - Prüfung führt zu Cache-Treffer oder zu Fehlzugriff.

- Arbeitsweise
  - Cache-Fehlzugriff (cache-miss):
    - eine oder beide Bedingungen sind nicht erfüllt!
    - Aktionen bei Lesezugriffen (read-miss):
      - Lesen des Datums aus 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

- Arbeitsweise
  - 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.

- Arbeitsweise
  - Cache-Treffer (Cache hit, read-hit, write-hit)
    - Bedingungen 1 und 2 sind erfüllt!
    - Zugriff erfolgt auf Cache-Speicher;

- Wohin wird ein Block im Cache geladen?
  - Cache-Organisation

Hauptspeicher

Block B_j $j = 0, 1, ..., (n-1)$

Kapazität: $n * b = 2^{s+w}$ Wörter

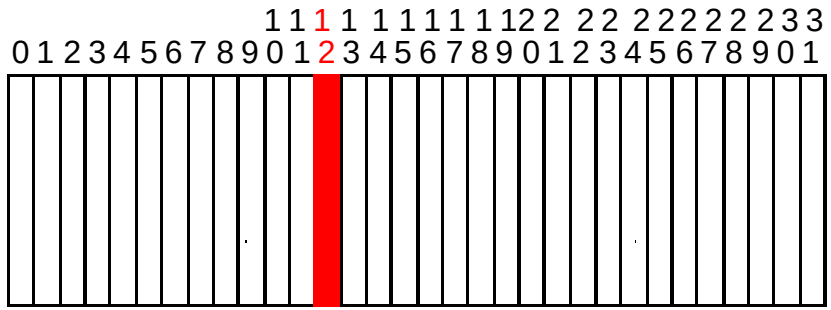


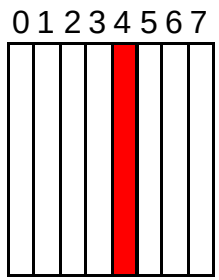
Abbildung von $\{B_j\}$ nach $\{Z_i\}$

Cache

Blockrahmen Z_i (Cache-Zeile)

$i = 0, 1, ...(m-1)$

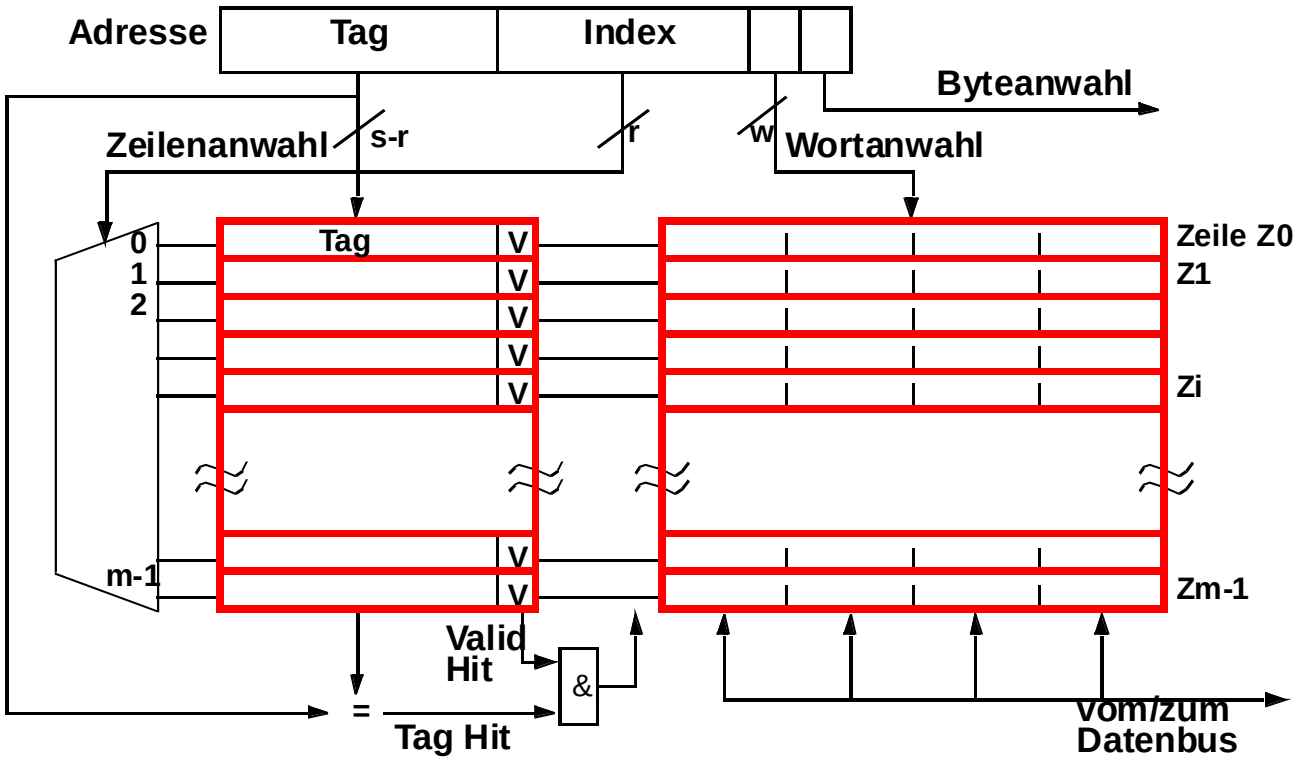
Kapazität: $m * b = 2^{r+w}$ Wörter



$n >> m$, $n = 2^s$, $m = 2^r$

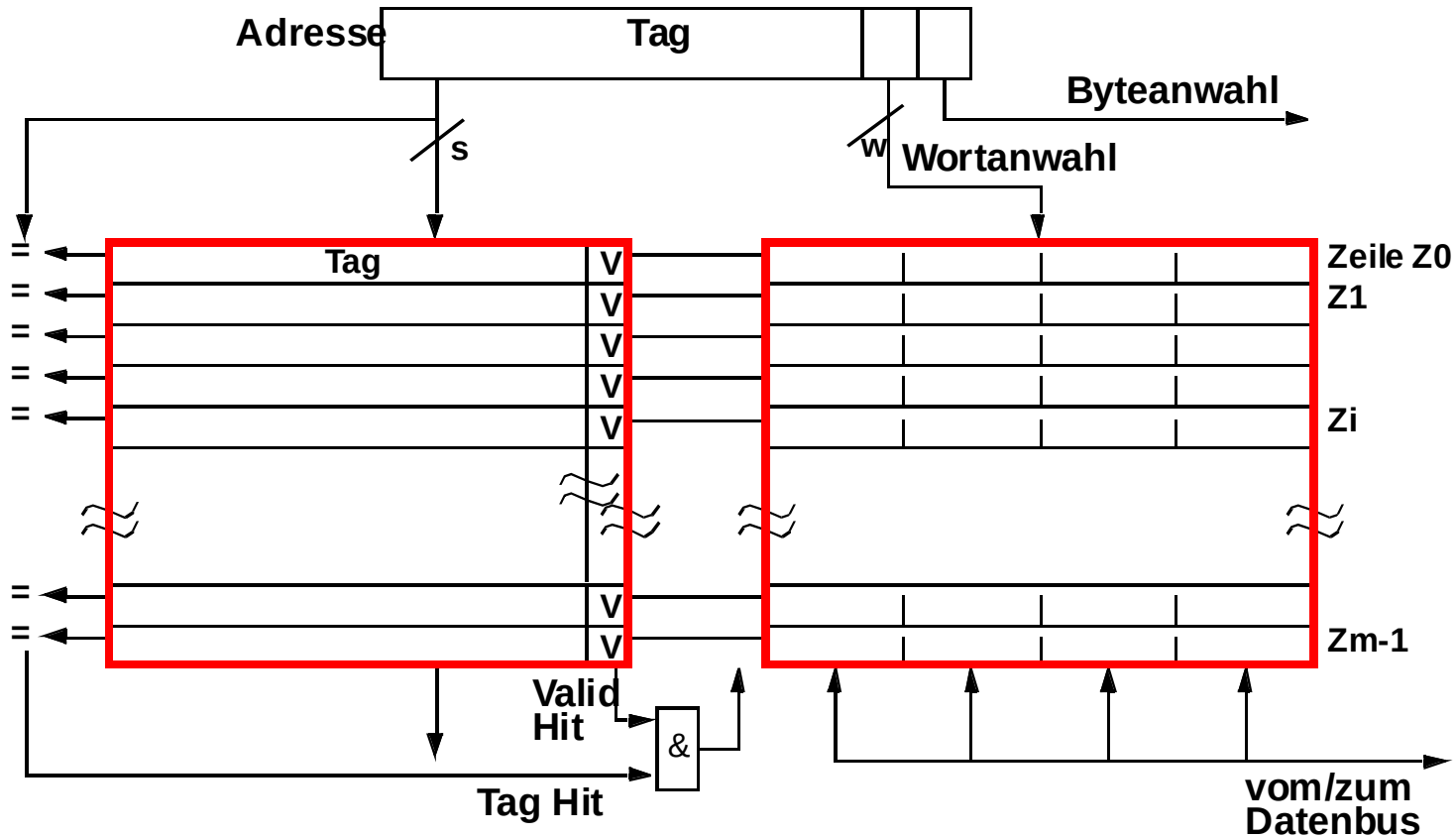
jeder Block enthält b Wörter mit $b = 2^w$

- Cache-Organisation
  - Direct-Mapped Cache



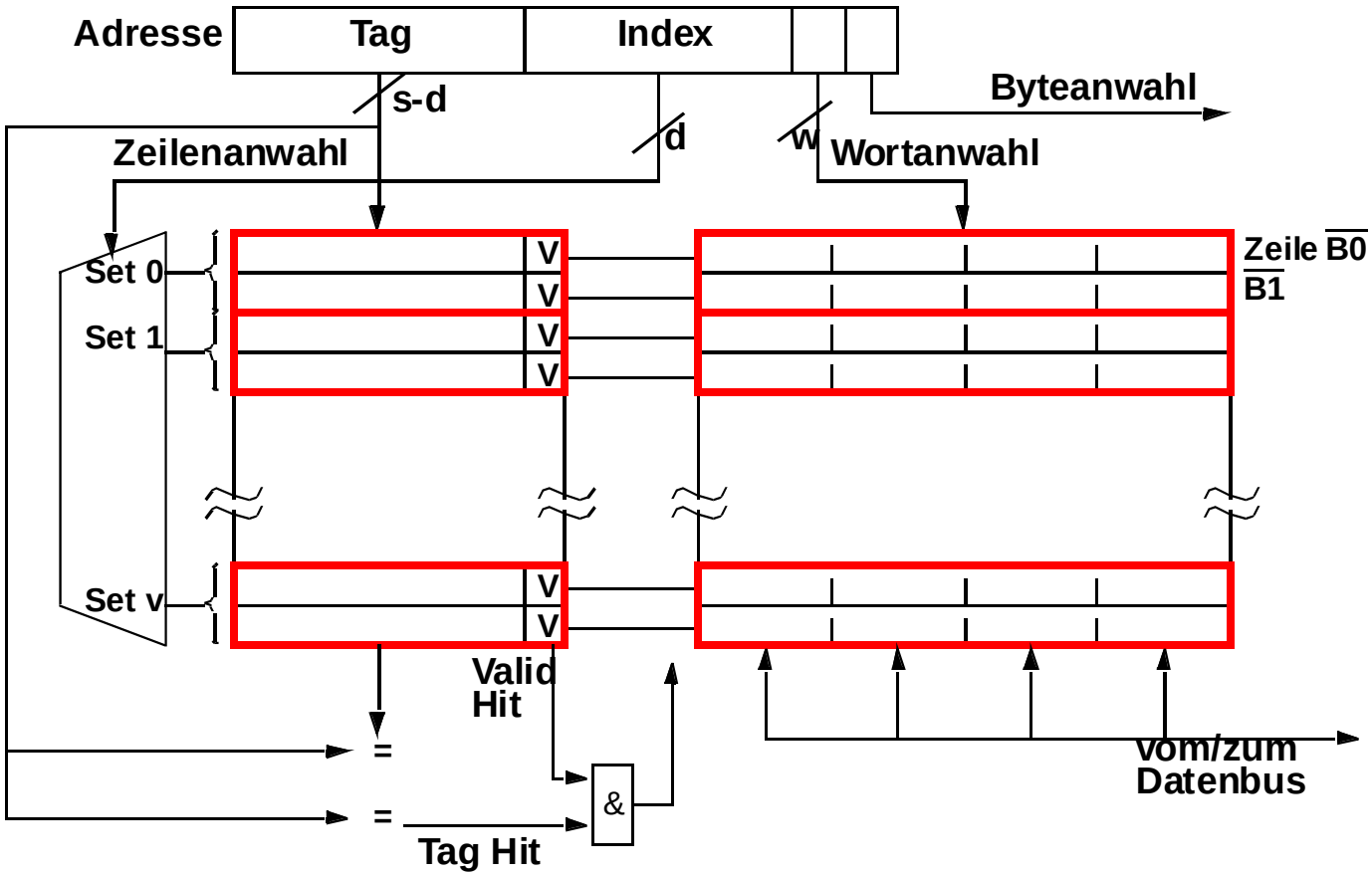
- Cache-Organisation
  - Vollassoziativer Speicher
    - jeder Block des Hauptspeichers kann auf jede Zeile des Cache-Speichers abgebildet werden (Flexibilität)
    - Ersetzungsstrategie gibt vor, welche Zeile beim Laden ersetzt werden soll (z.B. Least-Recently-Used)
    - hoher Hardware-Aufwand (Anzahl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

- Cache-Organisation
  - Vollassoziativer Cache



- 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:
    - $B_j \dashrightarrow Z_f \in S_i$, mit $j \pmod v = i$.

- Cache-Organisation
  - 2-fach mengenassoziativer Cache



- Aktualisierungsstrategie und Cache-Kohärenz
  - Kohärenz
    - Korrektes Voranschreiten des Systemzustandes durch ein abgestimmtes Zusammenwirken der Einzelzustände.
  - Cache-Kohärenz:
    - Prozessor muss auf aktuelle Inhalte im Cache wie im Hauptspeicher zugreifen können;

Wiederholung: Cache-Speicher

- Aktualisierungsstrategie und Cache-Kohärenz
  - Cache-Kohärenz:
    - Befehls-Caches:
      - Prozessor führt nur Lesezugriffe durch.
    - Daten-Cache:
      - Prozessor führt auch Schreibzugriffe durch.
  - Aktualisierungsstrategie
    - Aktualisieren des Cache-Speichers oder des Hauptspeichers oder beide (write hit):
    - Lesezugriffe dürfen nicht auf inzwischen veraltete Daten gehen;
  - Ein System ist konsistent, wenn alle Kopien eines Datenwortes im, Hauptspeicher und den verschiedenen Cache-Speichern identisch ist.

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