1

Kurzüberblick Multicore-Prozessoren

Die Entwicklung der Mikroprozessoren hat in den letzten Jahrzehnten durch verschiedene technologische Innovationen immer leistungsstärkere Prozessoren hervorgebracht. Multicore-Prozessoren stellen einen weiteren Meilenstein in der Entwicklung dar.

1.1 Entwicklung der Mikroprozessoren

Prozessorchips sind intern aus Transistoren aufgebaut, deren Anzahl ein ungefähres Maß für die Komplexität und Leistungsfähigkeit des Prozessors ist. Das auf empirischen Beobachtungen beruhende **Gesetz von Moore** besagt, dass die Anzahl der Transistoren eines Prozessorchips sich alle 18 bis 24 Monate verdoppelt. Diese Beobachtung wurde 1965 von Gordon Moore zum ersten Mal gemacht und gilt nun seit über 40 Jahren. Ein typischer Prozessorchip aus dem Jahr 2007 besteht aus ca. 200-400 Millionen Transistoren: beispielsweise besteht ein Intel Core 2 Duo Prozessor

aus ca. 291 Millionen Transistoren, ein IBM Cell-Prozessor aus ca. 250 Millionen Transistoren.

Die Erhöhung der Transistoranzahl ging in der Vergangenheit mit einer Erhöhung der Taktrate einher. Dies steigerte die Verarbeitungsgeschwindigkeit der Prozessoren und die Taktrate eines Prozessors wurde oftmals als alleiniges Merkmal für dessen Leistungsfähigkeit wahrgenommen. Gemeinsam führte die Steigerung der Taktrate und der Transistoranzahl zu einer durchschnittlichen jährlichen Leistungssteigerung der Prozessoren von ca. 55% (bei Integer-Operationen) bzw. 75% (bei Floating-Point-Operationen), was durch entsprechende Benchmark-Programme gemessen wurde, siehe [32] und `www.spec.org` für eine Beschreibung der oft verwendeten SPEC-Benchmarks. Eine Erhöhung der Taktrate im bisherigen Umfang ist jedoch für die Zukunft nicht zu erwarten. Dies ist darin begründet, dass mit einer Erhöhung der Taktrate auch die Leistungsaufnahme, also der Energieverbrauch des Prozessors, ansteigt, wobei ein Großteil des verbrauchten Stroms in Wärme umgewandelt wird und über Lüfter abgeführt werden muss. Das Gesetz von Moore scheint aber bis auf weiteres seine Gültigkeit zu behalten.

Die steigende Anzahl verfügbarer Transistoren wurde in der Vergangenheit für eine Vielzahl weiterer architektonischer Verbesserungen genutzt, die die Leistungsfähigkeit der Prozessoren erheblich gesteigert hat. Dazu gehören u.a.

- die Erweiterung der internen Wortbreite auf 64 Bits,
- die Verwendung interner Pipelineverarbeitung für die ressourcenoptimierte Ausführung aufeinanderfolgender Maschinenbefehle,
- die Verwendung mehrerer Funktionseinheiten, mit denen voneinander unabhängige Maschinenbefehle parallel zueinander abgearbeitet werden können und

- die Vergrößerung der prozessorlokalen Cachespeicher.

Wesentliche Aspekte der Leistungssteigerung sind also die Erhöhung der Taktrate und der interne Einsatz paralleler Abarbeitung von Instruktionen, z.B. durch das Duplizieren von Funktionseinheiten. Die Grenzen beider Entwicklungen sind jedoch abzusehen: Ein weiteres Duplizieren von Funktionseinheiten und Pipelinestufen ist zwar möglich, bringt aber wegen vorhandener Abhängigkeiten zwischen den Instruktionen kaum eine weitere Leistungssteigerung. Gegen eine weitere Erhöhung der prozessoreigenen Taktrate sprechen mehrere Gründe [36]:

- Ein Problem liegt darin, dass die Speicherzugriffsgeschwindigkeit nicht im gleichen Umfang wie die Prozessorgeschwindigkeit zunimmt, was zu einer Erhöhung der Zyklenanzahl pro Speicherzugriff führt. So brauchte z.B. um 1990 ein Intel i486 für einen Zugriff auf den Hauptspeicher zwischen 6 und 8 Maschinenzyklen, während 2006 ein Intel Pentium Prozessor über 220 Zyklen benötigt. Die Speicherzugriffszeiten stellen daher einen kritischen limitierenden Faktor für eine weitere Leistungssteigerung dar.
- Zum Zweiten wird die Erhöhung der Transistoranzahl durch eine erhöhte Packungsdichte erreicht, mit der aber auch eine gesteigerte Wärmeentwicklung pro Flächeneinheit verbunden ist. Diese wird zunehmend zum Problem, da die notwendige Kühlung entsprechend aufwendiger wird.
- Zum Dritten wächst mit der Anzahl der Transistoren auch die prozessorinterne Leitungslänge für den Signaltransport, so dass die Signallaufzeit eine wichtige Rolle spielt. Dies sieht man an folgender Berechnung: Ein mit 3GHz getakteter Prozessor hat eine Zykluszeit von 0.33 ns $= 0.33 \cdot 10^{-9}$ sec. In dieser Zeit kann ein Signal eine

Entfernung von $0.33 \cdot 10^{-9}$s$\cdot 0.3 \cdot 10^{9}$m/s $\approx$ 0.1m zurücklegen, wenn wir die Lichtgeschwindigkeit im Vakuum als Signalgeschwindigkeit ansetzen. Je nach Übergangsmedium ist die Signalgeschwindigkeit sogar deutlich niedriger. Damit können die Signale in einem Takt nur eine relativ geringe Entfernung zurücklegen, so dass der Layout-Entwurf der Prozessorchips entsprechend gestaltet werden muss.

Um eine weitere Leistungssteigerung der Prozessoren im bisherigen Umfang zu erreichen, setzen die Prozessorhersteller auf eine explizite Parallelverarbeitung innerhalb eines Prozessors, indem mehrere logische Prozessoren von einem physikalischen Prozessor simuliert werden oder mehrere vollständige, voneinander nahezu unabhängige Prozessorkerne auf einen Prozessorchip platziert werden. Der Einsatz expliziter Parallelverarbeitung innerhalb eines Prozessors hat weitreichende Konsequenzen für die Programmierung: soll ein Programm von der verfügbaren Leistung des Multicore-Prozessors profitieren, so muss es die verfügbaren Prozessorkerne entsprechend steuern und effizient ausnutzen. Dazu werden Techniken der parallelen Programmierung eingesetzt. Da die Prozessorkerne eines Prozessorchips ein gemeinsames Speichersystem nutzen, sind Programmieransätze für gemeinsamen Adressraum geeignet.

1.2 Parallelität auf Prozessorebene

Explizite Parallelität auf Prozessorebene wird durch eine entsprechende Architekturorganisation des Prozessorchips erreicht.

Eine Möglichkeit ist die oben erwähnte Platzierung mehrerer Prozessorkerne mit jeweils unabhängigen Ausführungseinheiten auf einem Prozessorchip, was als **Multicore-**

Prozessor bezeichnet wird. Ein anderer Ansatz besteht darin, mehrere Kontrollflüsse dadurch gleichzeitig auf einem Prozessor oder Prozessorkern auszuführen, dass der Prozessor je nach Bedarf per Hardware zwischen den Kontrollflüssen umschaltet. Dies wird als **simultanes Multithreading** (SMT) oder **Hyperthreading** (HT) bezeichnet [43].

Bei dieser Art der Parallelität werden die Kontrollflüsse oft als **Threads** bezeichnet. Dieser Begriff und die Unterschiede zu Prozessen werden in den folgenden Abschnitten näher erläutert; zunächst reicht es aus, einen Thread als Kontrollfluss anzusehen, der parallel zu anderen Threads desselben Programms ausgeführt werden kann.

Simultanes Multithreading (SMT)

Simultanes Multithreading basiert auf dem Duplizieren des Prozessorbereiches zur Ablage des Prozessorzustandes auf der Chipfläche des Prozessors. Zum Prozessorzustand gehören die Benutzer- und Kontrollregister sowie der Interrupt-Controller mit seinen zugehörigen Registern. Damit verhält sich der physikalische Prozessor aus der Sicht des Betriebssystems und des Benutzerprogramms wie zwei **logische Prozessoren**, denen Prozesse oder Threads zur Ausführung zugeordnet werden können. Diese können von einem oder mehreren Anwendungsprogrammen stammen.

Jeder logische Prozessor legt seinen Prozessorzustand in einem separaten Prozessorbereich ab, so dass beim Wechsel zu einem anderen Thread kein aufwendiges Zwischenspeichern des Prozessorzustandes im Speichersystem erforderlich ist. Die logischen Prozessoren teilen sich fast alle Ressourcen des physikalischen Prozessors wie Caches, Funktions- und Kontrolleinheiten oder Bussystem. Die Realisierung der SMT-Technologie erfordert daher nur eine ge-

ringfügige Vergrößerung der Chipfläche. Für zwei logische Prozessoren wächst z.B. für einen Intel Xeon Prozessor die erforderliche Chipfläche um weniger als 5% [44, 67]. Die gemeinsamen Ressourcen des Prozessorchips werden den logischen Prozessoren reihum zugeteilt, so dass die logischen Prozessoren simultan zur Verfügung stehen. Treten bei einem logischen Prozessor Wartezeiten auf, können die Ausführungs-Ressourcen dem anderen logischen Prozessor zugeordnet werden, so dass aus der Sicht des physikalischen Prozessors eine verbesserte Nutzung der Ressourcen gewährleistet ist.

Untersuchungen zeigen, dass die verbesserte Nutzung der Ressourcen durch zwei logische Prozessoren je nach Anwendungsprogramm Laufzeitverbesserungen zwischen 15% und 30% bewirken kann [44]. Da alle Ressourcen des Chips von den logischen Prozessoren geteilt werden, ist beim Einsatz von wesentlich mehr als zwei logischen Prozessoren für die meisten Einsatzgebiete keine weitere signifikante Laufzeitverbesserung zu erwarten. Der Einsatz simultanen Multithreadings wird daher voraussichtlich auf wenige logische Prozessoren beschränkt bleiben. Zum Erreichen einer Leistungsverbesserung durch den Einsatz der SMT-Technologie ist es erforderlich, dass das Betriebssystem in der Lage ist, die logischen Prozessoren anzusteuern. Aus Sicht eines Anwendungsprogramms ist es erforderlich, dass für jeden logischen Prozessor ein separater Thread zur Ausführung bereitsteht, d.h. für die Implementierung des Programms müssen Techniken der parallelen Programmierung eingesetzt werden.

Multicore-Prozessoren

Neue Prozessorarchitekturen mit mehreren Prozessorkernen auf einem Prozessorchip werden schon seit vielen Jah-

ren als die vielversprechendste Technik zur weiteren Leistungssteigerung angesehen. Die Idee besteht darin, anstatt eines Prozessorchips mit einer sehr komplexen internen Organisation mehrere Prozessorkerne mit einfacherer Organisation auf dem Prozessorchip zu integrieren. Dies hat den zusätzlichen Vorteil, dass der Stromverbrauch des Prozessorchips dadurch reduziert werden kann, dass vorübergehend ungenutzte Prozessorkerne abgeschaltet werden [27].

Bei Multicore-Prozessoren werden mehrere Prozessorkerne auf einem Prozessorchip integriert. Jeder Prozessorkern stellt für das Betriebssystem einen separaten logischen Prozessor mit separaten Ausführungsressourcen dar, die getrennt angesteuert werden müssen. Das Betriebssystem kann damit verschiedene Anwendungsprogramme parallel zueinander zur Ausführung bringen. So können z.B. mehrere Hintergrundanwendungen wie Viruserkennung, Verschlüsselung und Kompression parallel zu Anwendungsprogrammen des Nutzers ausgeführt werden [58]. Es ist aber mit Techniken der parallelen Programmierung auch möglich, ein rechenzeitintensives Anwendungsprogramm (etwa aus dem Bereich der Computerspiele, der Bildverarbeitung oder naturwissenschaftlicher Simulationsprogramme) auf mehreren Prozessorkernen parallel abzuarbeiten, so dass die Berechnungszeit im Vergleich zu einer Ausführung auf einem Prozessorkern reduziert werden kann.

Damit können auch Standardprogramme wie Textverarbeitungsprogramme oder Computerspiele zusätzliche, im Hintergrund ablaufende Funktionalitäten zur Verfügung stellen, die parallel zu den Haupt-Funktionalitäten auf einem separaten Prozessorkern durchgeführt werden und somit für den Nutzer nicht zu wahrnehmbaren Verzögerungen führen. Für die Koordination der innerhalb einer Anwendung ablaufenden unterschiedlichen Funktionalitäten

müssen Techniken der parallelen Programmierung eingesetzt werden.

1.3 Architektur von Multicore-Prozessoren

Für die Realisierung von Multicore-Prozessoren gibt es verschiedene Implementierungsvarianten, die sich in der Anzahl der Prozessorkerne, der Größe und Anordnung der Caches, den Zugriffmöglichkeiten der Prozessorkerne auf die Caches und dem Einsatz von heterogenen Komponenten unterscheiden [37]. Dabei werden zur Zeit drei unterschiedliche Architekturmodelle eingesetzt, von denen auch Mischformen auftreten können.

Hierarchisches Design

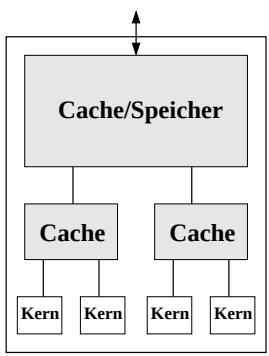

Abbildung 1.1. Hierarchisches Design.

Bei einem **hierarchischen Design** teilen sich mehrere Prozessorkerne mehrere Caches, die in einer baumartigen Konfiguration angeordnet sind, wobei die Größe der Caches von den Blättern zur Wurzel steigt. Die Wurzel repräsentiert die Verbindung zum Hauptspeicher. So kann z.B. jeder Prozessorkern einen separaten L1-Cache haben, sich aber mit anderen Prozessorkernen einen L2-Cache teilen. Alle Prozessorkerne können auf den gemeinsamen externen Hauptspeicher zugreifen, was eine

dreistufige Hierarchie ergibt. Dieses Konzept kann auf mehrere Stufen erweitert werden und ist in Abbildung 1.1 für drei Stufen veranschaulicht. Zusätzliche Untersysteme können die Caches einer Stufe miteinander verbinden. Ein hierarchisches Design wird typischerweise für Server-Konfigurationen verwendet.

Ein Beispiel für ein hierarchisches Design ist der IBM Power5 Prozessor, der zwei 64-Bit superskalare Prozessorkerne enthält, von denen jeder zwei logische Prozessoren durch Einsatz von SMT simuliert. Jeder Prozessorkern hat einen separaten L1-Cache (für Daten und Programme getrennt) und teilt sich mit dem anderen Prozessorkern einen L2-Cache (1.8 MB) sowie eine Schnittstelle zu einem externen 36 MB L3-Cache. Andere Prozessoren mit hierarchischem Design sind die Intel Core 2 Prozessoren und die Sun UltraSPARC T1 (Niagara) Prozessoren.

Pipeline-Design

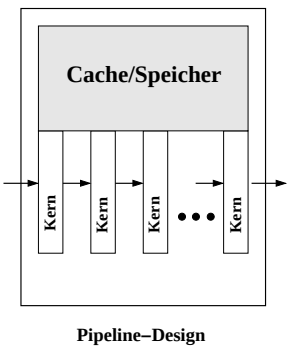

Abbildung 1.2. Pipeline-Design.

Bei einem **Pipeline-Design** werden die Daten durch mehrere Prozessorkerne schrittweise weiterverarbeitet, bis sie vom letzten Prozessorkern im Speichersystem abgelegt werden, vgl. Abbildung 1.2. Hochspezialisierte Netzwerk-Prozessoren und Grafikchips arbeiten oft nach diesem Prinzip. Ein Beispiel sind die X10 und X11 Prozessoren von Xelerator zur Verarbeitung von Netzwerkpaketen in Hochleistungs-Routern. Der Xelerator X10q,

eine Variante des X10, enthält z.B. 200 separate Prozessorkerne, die in einer logischen linearen Pipeline miteinander verbunden sind. Die Pakete werden dem Prozessor über mehrere Netzwerkschnittstellen zugeführt und dann durch die Prozessorkerne schrittweise verarbeitet, wobei jeder Prozessorkern einen Schritt ausführt. Die X11 Netzwerkprozessoren haben bis zu 800 Pipeline-Prozessorkerne.

Netzwerkbasiertes Design

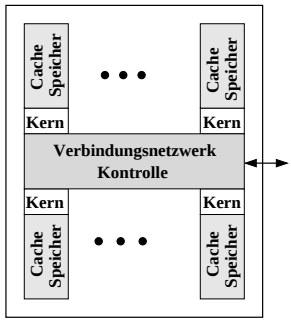

Abbildung 1.3. Netzwerkbasiertes Design.

Bei einem **netzwerkbasierten Design** sind die Prozessorkerne und ihre lokalen Caches oder Speicher über ein Verbindungsnetzwerk mit den anderen Prozessorkernen des Chips verbunden, so dass der gesamte Datentransfer zwischen den Prozessorkernen über das Verbindungsnetzwerk läuft, vgl. Abbildung 1.3. Der Einsatz eines prozessorinternen Netzwerkes ist insbesondere dann sinnvoll, wenn eine Vielzahl von Prozessorkernen verwendet werden soll. Ein netzwerkorientiertes Design wird z.B. für den Intel Teraflop-Prozessor verwendet, der im Rahmen der Intel Tera-Scale-Initiative entwickelt wurde, vgl. unten, und in dem 80 Prozessorkerne eingesetzt werden.

Weitere Entwicklungen

Das Potential der Multicore-Prozessoren wurde von vielen Hardwareherstellern wie Intel und AMD erkannt und

seit 2005 bieten viele Hardwarehersteller Prozessoren mit zwei oder mehr Kernen an. Ab Ende 2006 liefert Intel Quadcore-Prozessoren und ab 2008 wird mit der Auslieferung von Octcore-Prozessoren gerechnet. IBM bietet mit der Cell-Architektur einen Prozessor mit acht spezialisierten Prozessorkernen, vgl. Abschnitt 1.4. Der seit Dezember 2005 ausgelieferte UltraSPARC T1 Niagara Prozessor von Sun hat bis zu acht Prozessorkerne, von denen jeder durch den Einsatz von simultanem Multithreading, das von Sun als CoolThreads-Technologie bezeichnet wird, vier Threads simultan verarbeiten kann. Damit kann ein UltraSPARC T1 bis zu 32 Threads simultan ausführen. Das für 2008 angekündigte Nachfolgemodell des Niagara-Prozessors (ROCK) soll bis zu 16 Prozessorkerne enthalten.

Intel Tera-Scale-Initiative

Intel untersucht im Rahmen des *Tera-scale Computing Program*s die Herausforderungen bei der Herstellung und Programmierung von Prozessoren mit Dutzenden von Prozessorkernen [27]. Diese Initiative beinhaltet auch die Entwicklung eines Teraflop-Prozessors, der 80 Prozessorkerne enthält, die als 8×10-Gitter organisiert sind. Jeder Prozessorkern kann Floating-Point-Operationen verarbeiten und enthält neben einem lokalen Cachespeicher auch einen Router zur Realisierung des Datentransfers zwischen den Prozessorkernen und dem Hauptspeicher. Zusätzlich kann ein solcher Prozessor spezialisierte Prozessorkerne für die Verarbeitung von Videodaten, graphischen Berechnungen und zur Verschlüsselung von Daten enthalten. Je nach Einsatzgebiet kann die Anzahl der spezialisierten Prozessorkerne variiert werden.

Ein wesentlicher Bestandteil eines Prozessors mit einer Vielzahl von Prozessorkernen ist ein effizientes Verbin-

dungsnetzwerk auf dem Prozessorchip, das an eine variable Anzahl von Prozessorkernen angepasst werden kann, den Ausfall einzelner Prozessorkerne toleriert und bei Bedarf das Abschalten einzelner Prozessorkerne erlaubt, falls diese für die aktuelle Anwendung nicht benötigt werden. Ein solches Abschalten ist insbesondere zur Reduktion des Stromverbrauchs sinnvoll.

Für eine effiziente Nutzung der Prozessorkerne ist entscheidend, dass die zu verarbeitenden Daten schnell zu den Prozessorkernen transportiert werden können, so dass diese nicht auf die Bereitstellung der Daten warten müssen. Dazu sind ein leistungsfähiges Speichersystem und I/O-System erforderlich. Das Speichersystem setzt private L1-Caches ein, auf die nur von jeweils einem Prozessorkern zugegriffen werden kann, sowie gemeinsame, evtl. aus mehreren Stufen bestehende L2-Caches ein, die Daten verschiedener Prozessorkerne enthalten. Für einen Prozessorchip mit Dutzenden von Prozessorkernen muss voraussichtlich eine weitere Stufe im Speichersystem eingesetzt werden [27]. Das I/O-System muss in der Lage sein, Hunderte von Gigabytes pro Sekunde zum Prozessorchip zu transportieren. Hier arbeitet z.B. Intel an der Entwicklung geeigneter Systeme.

Tabelle 1.1 gibt einen Überblick über aktuelle Multicore-Prozessoren. Zu bemerken ist dabei, dass der Sun UltraSPARC T1-Prozessor im Gegensatz zu den drei anderen Prozessoren kaum Unterstützung für Floating-Point-Berechnungen bietet und somit überwiegend für den Einsatz im Serverbereich, wie Web-Server oder Applikations-Server, geeignet ist. Für eine detailliertere Behandlung der Architektur von Multicore-Prozessoren und weiterer Beispiele verweisen wir auf [10, 28].

Tabelle 1.1. Überblick über verschiedene Multicore-Prozessoren, vgl. auch [28].

Prozessor	Intel Core 2 Duo	IBM Power 5	AMD Opteron	Sun T1
Prozessorkerne	2	2	2	8
Instruktionen pro Zyklus	4	4	3	1
SMT	nein	ja	nein	ja
L1-Cache I/D in KB per core	32/32	64/32	64/64	16/8
L2-Cache	4 MB shared	1.9 MB shared	1 MB per core	3 MB shared
Taktrate (GHz)	2.66	1.9	2.4	1.2
Transistoren	291 Mio	276 Mio	233 Mio	300 Mio
Stromverbrauch	65 W	125 W	110 W	79 W

1.4 Beispiele

Im Folgenden wird die Architektur von Multicore-Prozessoren anhand zweier Beispiele verdeutlicht: der Intel Core 2 Duo-Architektur und dem IBM Cell-Prozessor.

Intel Core 2 Prozessor

Intel Core 2 bezeichnet eine Familie von Intel-Prozessoren mit ähnlicher Architektur. Die Intel Core-Architektur basiert auf einer Weiterentwicklung der Pentium M Prozessoren, die viele Jahre im Notebookbereich eingesetzt wurden. Die neue Architektur löst die bei den Pentium 4 Prozessoren noch eingesetzte NetBurst-Architektur ab. Signifikante Merkmale der neuen Architektur sind:

- eine drastische Verkürzung der internen Pipelines (maximal 14 Stufen anstatt maximal 31 Stufen bei der NetBurst-Architektur), damit verbunden
- eine Reduktion der Taktrate und damit verbunden auch
- eine deutliche Reduktion des Stromverbrauchs: die Reduktion des Stromverbrauches wird auch durch eine Power-Management-Einheit unterstützt, die das zeitweise Ausschalten ungenutzter Prozessorteile ermöglicht [48] und
- die Unterstützung neuer Streaming-Erweiterungen (Streaming SIMD Extensions, *SSE*).

Intel Core 2 Prozessoren werden zur Zeit (August 2007) als Core 2 Duo bzw. Core 2 Quad Prozessoren mit 2 bzw. 4 unabhängigen Prozessorkernen in 65nm-Technologie gefertigt. Im Folgenden wird der Core 2 Duo Prozessor kurz beschrieben [24]. Die Core 2 Quad Prozessoren haben einen ähnlichen Aufbau, enthalten aber 4 statt 2 Prozessorkerne.

Da die Core 2 Prozessoren auf der Technik des Pentium M Prozessors basieren, unterstützen sie kein Hyperthreading. Die allgemeine Struktur der Core 2 Duo Architektur ist in Abb. 1.4 wiedergegeben. Der Prozessorchip enthält zwei unabhängige Prozessorkerne, von denen jeder separate L1-Caches anspricht; diese sind für Instruktionen (32K) und Daten (32K) getrennt realisiert. Der L2-Cache (4 MB) ist dagegen nicht exklusiv und wird von beiden Prozessorkernen gemeinsam für Instruktionen und Daten genutzt. Alle Zugriffe von den Prozessorkernen und vom externen Bus auf den L2-Cache werden von einem L2-Controller behandelt. Für die Sicherstellung der Kohärenz der auf den verschiedenen Stufen der Speicherhierarchie abgelegten Daten wird ein MESI-Protokoll (Modified, Exclusive, Shared, Invalid) verwendet, vgl. [17, 47, 59] für eine detaillierte Erklärung. Alle Daten- und I/O-Anfragen zum oder vom externen Bus

(Front Side Bus) werden über einen Bus-Controller gesteuert.

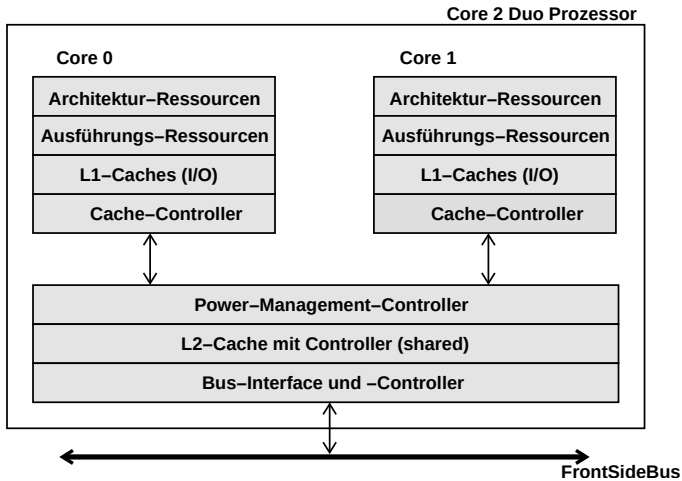

Abbildung 1.4. Überblick Core 2 Duo Architektur.

Ein wichtiges Element ist die Kontrolleinheit für den Stromverbrauch des Prozessors (Power Management Controller) [48], die den Stromverbrauch des Prozessorchips durch Reduktion der Taktrate der Prozessorenkerne oder durch Abschalten (von Teilen) des L2-Caches reduzieren kann.

Jeder Prozessorkern führt einen separaten Strom von Instruktionen aus, der sowohl Berechnungs- als auch Speicherzugriffsinstruktionen (load/store) enthalten kann. Dabei kann jeder der Prozessorkerne bis zu vier Instruktionen gleichzeitig verarbeiten. Die Prozessorkerne enthalten separate Funktionseinheiten für das Laden bzw. Speichern von

Daten, die Ausführung von Integeroperationen (durch eine ALU, arithmetic logic unit), Floating-Point-Operationen sowie SSE-Operationen. Instruktionen können aber nur dann parallel zueinander ausgeführt werden, wenn keine Abhängigkeiten zwischen ihnen bestehen. Für die Steuerung der Ausführung werden komplexe Schedulingverfahren eingesetzt, die auch eine Umordnung von Instruktionen (*out-of-order execution*) erlauben, um eine möglichst gute Ausnutzung der Funktionseinheiten zu verwirklichen [28].

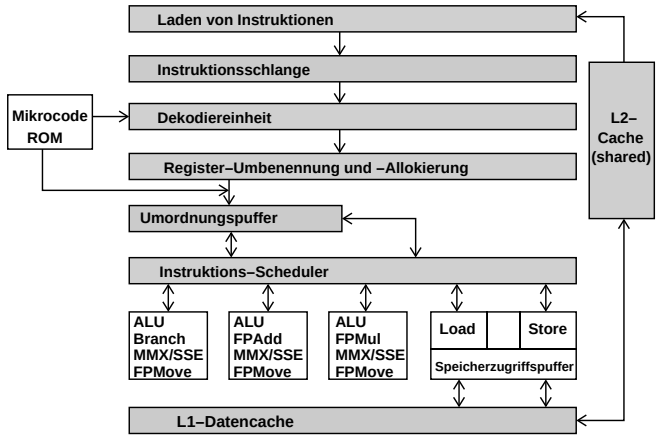

Abbildung 1.5. Instruktionsverarbeitung und Speicherorganisation eines Prozessorkerns des Intel Core 2 Prozessors.

Abbildung 1.5 veranschaulicht die Organisation der Abarbeitung von Instruktionen durch einen der Prozessorkerne [20]. Jeder der Prozessorkerne lädt $x86$-Instruktionen in eine Instruktionsschlange, auf die die Dekodiereinheit zugreift und die Instruktionen in Mikroinstruktionen zer-

legt. Für komplexere $x86$-Instruktionen werden die zugehörigen Mikroinstruktionen über einen ROM-Speicher geladen. Die Mikroinstruktionen werden vom Instruktions-Scheduler freien Funktionseinheiten zugeordnet, wobei die Instruktionen in einer gegenüber dem ursprünglichen Programmcode geänderten Reihenfolge abgesetzt werden können. Alle Speicherzugriffsoperationen werden über den L1-Datencache abgesetzt, der Integer-und Floating-Point-Daten enthält.

Für Ende 2007 bzw. Anfang 2008 sollen Intel Core 2-Prozessoren mit verbesserter Core-Architektur eingeführt werden (Codename Penryn). Voraussichtlich für Ende 2008 ist eine neue Generation von Intel-Prozessoren geplant, die auf einer neuen Architektur basiert (Codename Nehalem). Diese neuen Prozessoren sollen neben mehreren Prozessorkernen (zu Beginn acht) auch einen Graphikkern und einen Speichercontroller auf einem Prozessorchip integrieren. Die neuen Prozessoren sollen auch wieder die SMT-Technik (simultanes Multithreading) unterstützen, so dass auf jedem Prozessorkern gleichzeitig zwei Threads ausgeführt werden können. Diese Technik wurde teilweise für Pentium 4 Prozessoren verwendet, nicht jedoch für die Core 2 Duo und Quad Prozessoren.

IBM Cell-Prozessor

Der **Cell-Prozessor** wurde von IBM in Zusammenarbeit mit Sony und Toshiba entwickelt. Der Prozessor wird u.a. von Sony in der Spielekonsole PlayStation 3 eingesetzt, siehe [39, 34] für ausführlichere Informationen. Der Cell-Prozessor enthält ein *Power Processing Element* (PPE) und 8 *Single-Instruction Multiple-Datastream* (SIMD) Prozessoren. Das PPE ist ein konventioneller 64-Bit-Mikroprozessor auf der Basis der Power-Architektur von IBM mit relativ

einfachem Design: der Prozessor kann pro Takt zwei Instruktionen absetzen und simultan zwei unabhängige Threads ausführen. Die einfache Struktur hat den Vorteil, dass trotz hoher Taktrate eine geringe Leistungsaufnahme resultiert. Für den gesamten Prozessor ist bei einer Taktrate von 3.2 GHz nur eine Leistungsaufnahme von 60-80 Watt erforderlich.

Auf der Chipfläche des Cell-Prozessors sind neben dem PPE acht SIMD-Prozessoren integriert, die als SPE (*Synergetic Processing Element*) bezeichnet werden. Jedes SPE stellt einen unabhängigen Vektorprozessor mit einem 256KB großen lokalem SRAM-Speicher dar, der als *Local Store* (LS) bezeichnet wird. Das Laden von Daten in den LS und das Zurückspeichern von Resultaten aus dem LS in den Hauptspeicher muss per Software erfolgen.

Jedes SPE enthält 128 128-Bit-Register, in denen die Operanden von Instruktionen abgelegt werden können. Da auf die Daten in den Registern sehr schnell zugegriffen werden kann, reduziert die große Registeranzahl die Notwendigkeit von Zugriffen auf den LS und führt damit zu einer geringen mittleren Speicherzugriffszeit. Jedes SPE hat vier Floating-Point-Einheiten (32 Bit) und vier Integer-Einheiten. Zählt man eine Multiply-Add-Instruktion als zwei Operationen, kann jedes SPE bei 3.2 GHz Taktrate pro Sekunde über 25 Milliarden Floating-Point-Operationen (25.6 GFlops) und über 25 Milliarden Integer-Operationen (25.6 Gops) ausführen. Da ein Cell-Prozessor acht SPE enthält, führt dies zu einer maximalen Performance von über 200 GFlops, wobei die Leistung des PPE noch nicht berücksichtigt wurde. Eine solche Leistung kann allerdings nur bei guter Ausnutzung der LS-Speicher und effizienter Zuordnung von Instruktionen an Funktionseinheiten der SPE erreicht werden. Zu beachten ist auch, dass sich diese Angabe auf 32-Bit Floating-Point-Zahlen bezieht. Der Cell-

Prozessor kann durch Zusammenlegen von Funktionseinheiten auch 64-Bit Floating-Point-Zahlen verarbeiten, dies resultiert aber in einer wesentlich geringeren maximalen Performance. Zur Vereinfachung der Steuerung der SPEs und zur Vereinfachung des Schedulings verwenden die SPEs intern keine SMT-Technik.

Die zentrale Verbindungseinheit des Cell-Prozessors ist ein Bussystem, der sogenannte Element Interconnect Bus (EIB). Dieser besteht aus vier unidirektionalen Ringverbindungen, die eine Wortbreite von 16 Bytes haben und mit der halben Taktrate des Prozessors arbeiten. Zwei der Ringe werden in entgegengesetzter Richtung zu den anderen beiden Ringe betrieben, so dass die maximale Latenz im schlechtesten Fall durch einen halben Ringdurchlauf bestimmt wird. Für den Transport von Daten zwischen benachbarten Ringelementen können maximal drei Transferoperationen simultan durchgeführt werden, für den Zyklus des Prozessors ergibt dies $16 \cdot 3/2 = 24$ Bytes pro Zyklus. Für die vier Ringverbindungen ergibt dies eine maximale Transferrate von 96 Bytes pro Zyklus, woraus bei einer Taktrate von 3.2 GHz eine maximale Transferrate von über 300 GBytes/Sekunde resultiert. Abbildung 1.6 zeigt einen schematischen Aufbau des Cell-Prozessors mit den bisher beschriebenen Elementen sowie dem Speichersystem (Memory Interface Controller, MIC) und dem I/O-System (Bus Interface Controller, BIC). Das Speichersystem unterstützt die XDR-Schnittstelle von Rambus. Das I/O-System unterstützt das Rambus RRAC (Redwood Rambus Access Cell) Protokoll.

Zum Erreichen einer guten Leistung ist es wichtig, die SPEs des Cell-Prozessors effizient zu nutzen. Dies kann für spezialisierte Programme, wie z.B. Videospiele, durch direkte Verwendung von SPE-Assembleranweisungen erreicht werden. Da dies für die meisten Anwendungsprogramme

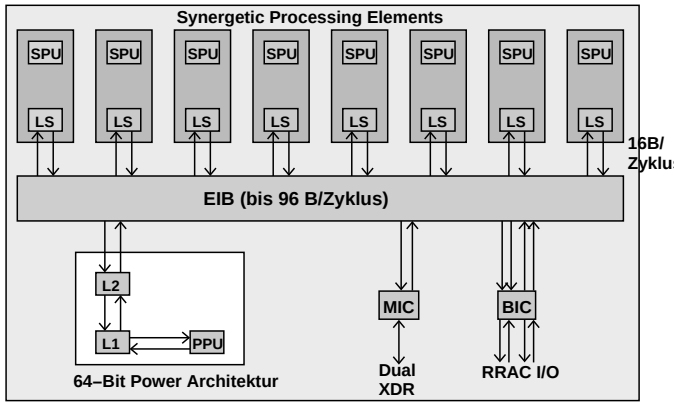

Abbildung 1.6. Schematischer Aufbau des Cell-Prozessors.

zu aufwendig ist, werden für das Erreichen einer guten Gesamtleistung eine effektive Compilerunterstützung sowie die Verwendung spezialisierter Programmbibliotheken z.B. zur Verwaltung von Taskschlangen wichtig sein.