Halbleitertechnologie von A bis Z

Digitaltechnik

Aufbau eines Prozessors

Einordnung

Ein Prozessor (CPU, Central Processing Unit) ist das „Gehirn" eines Computers – aber anders, als es dieses Bild vermuten lässt, denkt er nicht selbst, sondern arbeitet stur eine Liste von Befehlen ab, die im Speicher liegt. Diese Liste heißt Programm, jeder einzelne Eintrag ist ein Maschinenbefehl (z. B. „addiere zwei Zahlen" oder „lade einen Wert aus dem Speicher"). Ein Speicherbaustein wie SRAM oder DRAM kann Daten nur halten und bei Bedarf herausgeben – der Prozessor dagegen verarbeitet diese Daten aktiv: Er liest Befehle, versteht sie und führt sie aus, einen nach dem anderen, milliardenfach pro Sekunde. Alles, was ein Computer „tut" – von einer einfachen Addition bis zum Rendern eines Videos – ist letztlich eine sehr lange Kette solcher einfachen Einzelschritte.

Grundarchitektur

Damit ein Prozessor überhaupt an Befehle und Daten herankommt, braucht er einen Weg zum Speicher. Die meisten heutigen Prozessoren folgen der *Von-Neumann-Architektur*: Programm (die Befehle) und Daten (die Werte, mit denen gerechnet wird) liegen im selben Speicher und werden über denselben Übertragungsweg geladen. Das vereinfacht den Entwurf erheblich, erzeugt aber eine Engstelle: Befehl und Daten können nicht gleichzeitig über diesen einen Weg transportiert werden, der Prozessor muss abwechselnd warten (der sogenannte Von-Neumann-Flaschenhals). Die *Harvard-Architektur* trennt deshalb Programmspeicher und Datenspeicher physisch voneinander und erlaubt echten Parallelzugriff auf beide gleichzeitig. Ein Kompromiss, den viele moderne Prozessoren nutzen: Sie sind nach außen von-Neumann-artig aufgebaut, besitzen intern aber getrennte Caches für Befehle und Daten, um von den Vorteilen beider Ansätze zu profitieren.

Kernkomponenten

Um zu verstehen, wie ein Befehl tatsächlich ausgeführt wird, hilft ein Blick auf drei zentrale Bausteine:

Das *Steuerwerk* (Control Unit) ist so etwas wie der Dirigent des Prozessors. Es liest den nächsten Befehl aus dem Speicher, erkennt, um welche Art von Operation es sich handelt, und erzeugt daraufhin die passenden elektrischen Steuersignale, die alle anderen Bausteine im richtigen Moment aktivieren – ohne selbst zu rechnen.

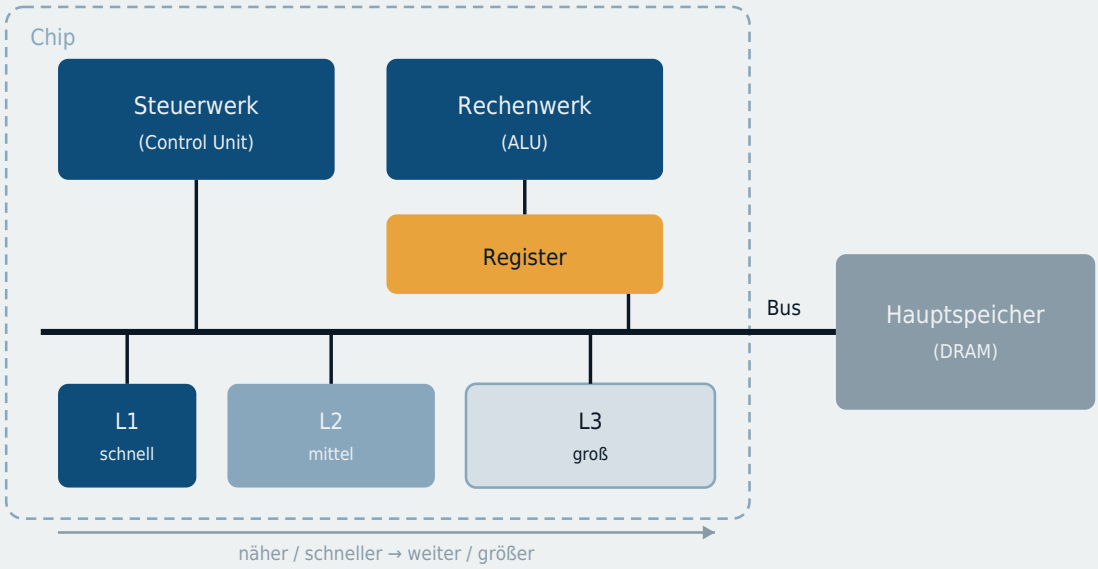
Das *Rechenwerk* (ALU, Arithmetic Logic Unit) ist der Baustein, der tatsächlich rechnet: Addition, Subtraktion, Vergleiche, logische Verknüpfungen (UND, ODER, usw.). Es bekommt vom Steuerwerk gesagt, welche Operation auszuführen ist, und von wo die Eingabewerte kommen.

Register sind winzige, extrem schnelle Speicherzellen direkt im Prozessor – vergleichbar mit einem Notizzettel, auf dem die ALU ihre Zwischenergebnisse ablegt, während sie rechnet. Es gibt nur eine Handvoll davon (typischerweise 16–32), weil sie viel Chipfläche pro Bit benötigen; dafür sind sie um Größenordnungen schneller als jeder andere Speicher im System.

Damit diese Bausteine überhaupt Daten austauschen können, braucht es einen *Bus* – eine Sammlung elektrischer Leitungen, über die Adressen (wo liegt ein Wert?), Daten (welcher Wert?) und Steuersignale (was soll damit passieren?) zwischen Steuerwerk, Rechenwerk, Registern, Cache und Speicher hin- und hertransportiert werden. Man kann sich den Bus wie eine gemeinsam genutzte Straße vorstellen, auf der zu jedem Zeitpunkt nur eine bestimmte Menge an „Verkehr" (Bits) gleichzeitig fließen kann – seine Breite (z. B. 32 oder 64 Bit) bestimmt, wie viele Daten pro Takt übertragen werden können.

Ein einzelner Befehl durchläuft dabei immer drei Phasen: *Fetch* (das Steuerwerk holt den nächsten Befehl aus dem Speicher über den Bus), *Decode* (der Befehl wird interpretiert, das Steuerwerk erkennt die auszuführende Operation) und *Execute* (die ALU führt die Operation aus, das Ergebnis landet in einem Register oder wird zurück in den Speicher geschrieben).

Aufbau eines Prozessors – Blockdiagramm



Bus verbindet alle Einheiten; Cache überbrückt die Lücke zwischen Registern und Hauptspeicher

Speicherhierarchie

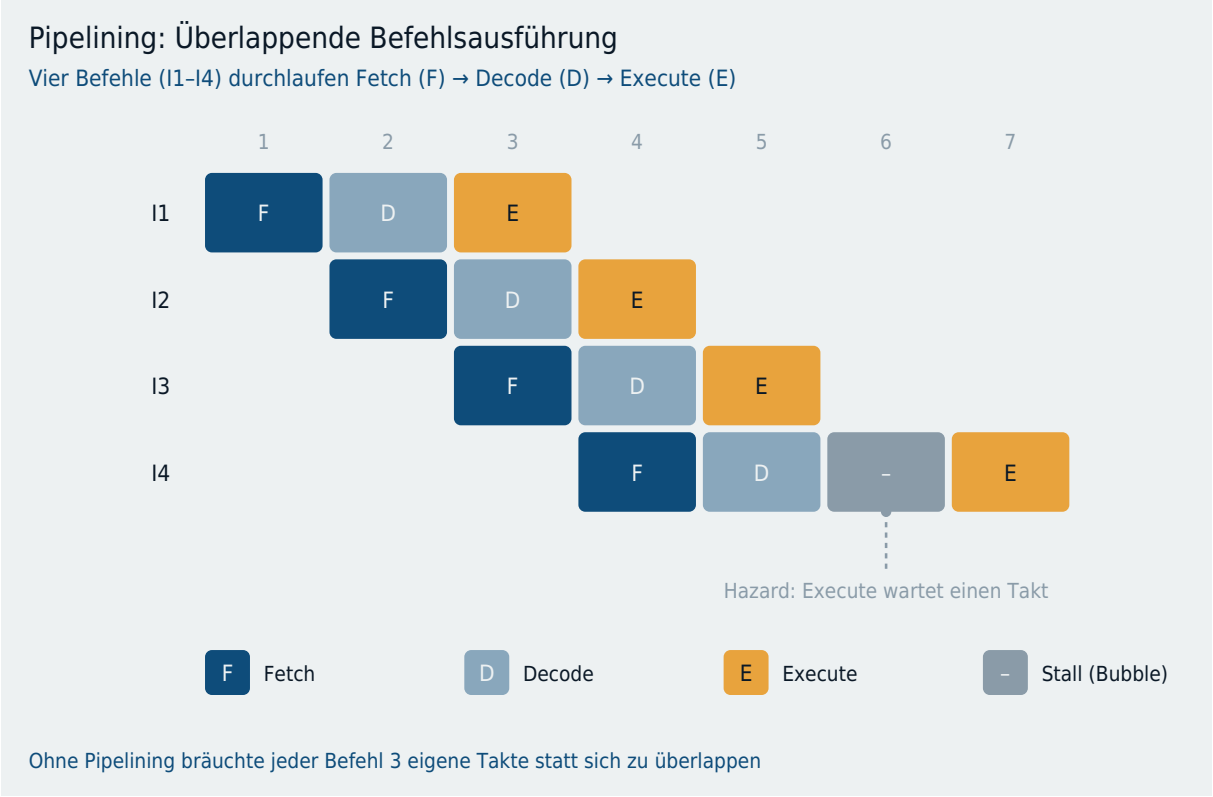
Register sind extrem schnell, aber es gibt nur wenige von ihnen – für alles, was gerade nicht in Bearbeitung ist, braucht der Prozessor größeren Speicher. Hier setzt der *Cache* an: ein kleiner, aber sehr schneller Zwischenspeicher, der häufig benötigte Daten und Befehle nahe am Rechenwerk vorhält, damit der Prozessor nicht bei jedem Zugriff den langsamen Hauptspeicher bemühen muss. Cache ist meist in mehreren Stufen gestaffelt: L1 (Level 1) sitzt direkt am Rechenwerk, ist am schnellsten, aber auch am kleinsten (oft nur 32–64 KB); L2 ist größer, aber etwas langsamer; L3 ist nochmals größer, langsamer, und wird häufig von mehreren Rechenkernen gemeinsam genutzt. Der Hauptspeicher (DRAM, auf separaten Speicherchips außerhalb des Prozessors) ist um Größenordnungen größer als jeder Cache, aber auch deutlich langsamer, weil die Signale einen viel längeren physischen Weg zurücklegen müssen.

Dass diese Hierarchie überhaupt funktioniert, liegt am Prinzip der *Lokalität*: Programme greifen typischerweise wiederholt auf dieselben Speicherbereiche zu (zeitliche Lokalität) und auf Adressen, die nahe beieinanderliegen (räumliche Lokalität) – zum Beispiel bei einer Schleife, die immer wieder dieselben paar Variablen verwendet. Der Cache nutzt das aus, indem er kürzlich verwendete Daten „vorhält" und so die meisten Zugriffe abfängt, bevor sie den langsamen Hauptspeicher erreichen müssen.

Takt, Pipelining und Parallelität

Ein Taktgeber gibt dem Prozessor seinen Rhythmus vor: Die Taktfrequenz (z. B. 3 GHz = 3 Milliarden Takte pro Sekunde) bestimmt, wie oft pro Sekunde eine neue Verarbeitungsstufe angestoßen wird. Höhere Taktfrequenzen bedeuten mehr Verarbeitungsschritte pro Sekunde, erzeugen aber auch mehr Verlustleistung (Wärme) und stoßen an physikalische Grenzen, weil Signale in der verfügbaren Zeit nicht beliebig schnell durch die Schaltung laufen können.

Um trotzdem mehr Leistung herauszuholen, nutzen moderne Prozessoren *Pipelining*: Statt einen Befehl komplett von Fetch bis Execute abzuarbeiten, bevor der nächste beginnt, wird die Ausführung in mehrere Stufen zerlegt, die wie an einem Fließband gleichzeitig, aber versetzt an unterschiedlichen Befehlen arbeiten. Während ein Befehl gerade ausgeführt wird (Execute), wird bereits der nächste decodiert (Decode) und ein dritter aus dem Speicher geholt (Fetch) – alle im selben Takt, aber in verschiedenen Stufen. Das erhöht den Durchsatz spürbar, ohne dass ein einzelner Befehl schneller würde. Probleme entstehen, wenn ein Befehl vom Ergebnis eines vorherigen abhängt, das noch nicht fertig ist (sogenannte *Hazards*) – dann muss die Pipeline kurz pausieren oder umsortieren, was zusätzliche Steuerlogik erfordert.



Weil einzelne Rechenkerne kaum noch schneller getaktet werden können, ohne unpraktikabel viel Wärme zu erzeugen, setzen moderne Prozessoren stattdessen auf *Multicore-Design*: Statt eines einzelnen, immer schnelleren Kerns integrieren sie mehrere vollständige, aber etwas einfachere Rechenkerne auf einem Die, die unabhängige Aufgaben parallel abarbeiten können.

Vom Die zum Chip

Im fertigen Layout sind Steuerwerk, Rechenwerk, Register und Cache als klar abgegrenzte, wiederholt genutzte Blöcke auf dem Die angeordnet – ähnlich wie Zimmer in einem Gebäudeplan. Der Cache nimmt dabei oft einen erstaunlich großen Flächenanteil ein, weil jede einzelne SRAM-Speicherzelle aus sechs Transistoren besteht und damit vergleichsweise viel Fläche benötigt. E/A-Bereiche (I/O-Pads), über die der Chip mit der Außenwelt – Speicher, Stromversorgung, andere Bausteine – verbunden ist, liegen am Rand des Dies, damit die Bonddrähte oder Lötkontakte (Bumps) zu den Gehäuseanschlüssen kurz bleiben. Wie diese Blöcke im Detail geplant und angeordnet werden, zeigt das Kapitel zu Schaltungslayouts.