Technische Universität Braunschweig

Informatik für Ingenieure – VL 12

Übersicht

Letztes Mal:

- Mikrorechner
- Adressierung
- Typische Befehle

Heute:

- Typische Befehle
- Instruction Set Architecture
- Systembus

Teile des heutigen Vortrags basiert auf der Vorlesungen von
Prof. H Michalik (TU Braunschweig)
Prof. B Parhami (UC Santa Barbara)
Introduction to SoC Design (ARM University)
Rapid Embedded System Design (ARM University)

Weitere Typische Befehle

Logische Befehle

- Zu den elementarsten Manipulationen von Binärfolgen zählen logische Operationen. Allgemein üblich sind die vier folgenden Anweisungen:
  - AND : logisches UND
  - OR: logisches ODER
  - XOR : exklusives ODER
  - NOT : Negation
- Die jeweils korrespondierenden Stellen der Operanden werden gemäß der spezifizierten Operation miteinander verknüpft.

Arithmetische Befehle

Arithmetische Anweisungen für ganzzahlige Operanden

- Für die Manipulation ganzzahliger Operanden gibt es Additions-, Subtraktions-, Multiplikations- und Divisionsanweisungen. Ihre Realisierung kann unterschiedlich erfolgen.

Arithmetische Anweisungen für Gleitkomma-Operanden:

- Für Gleitkomma-Operanden gibt es wiederum die vier Operationen Addition, Subtraktion, Multiplikation und Division. Bei Mikrorechnern sind derartige Anweisungen allerdings meist nicht im Mikroprogrammcode vorhanden, sondern werden mit den vorhandenen Befehlen für ganzzahlige Operanden per Emulation durch ein Programm ausgeführt.

Beispiel für eine Gleitkomma Addition:

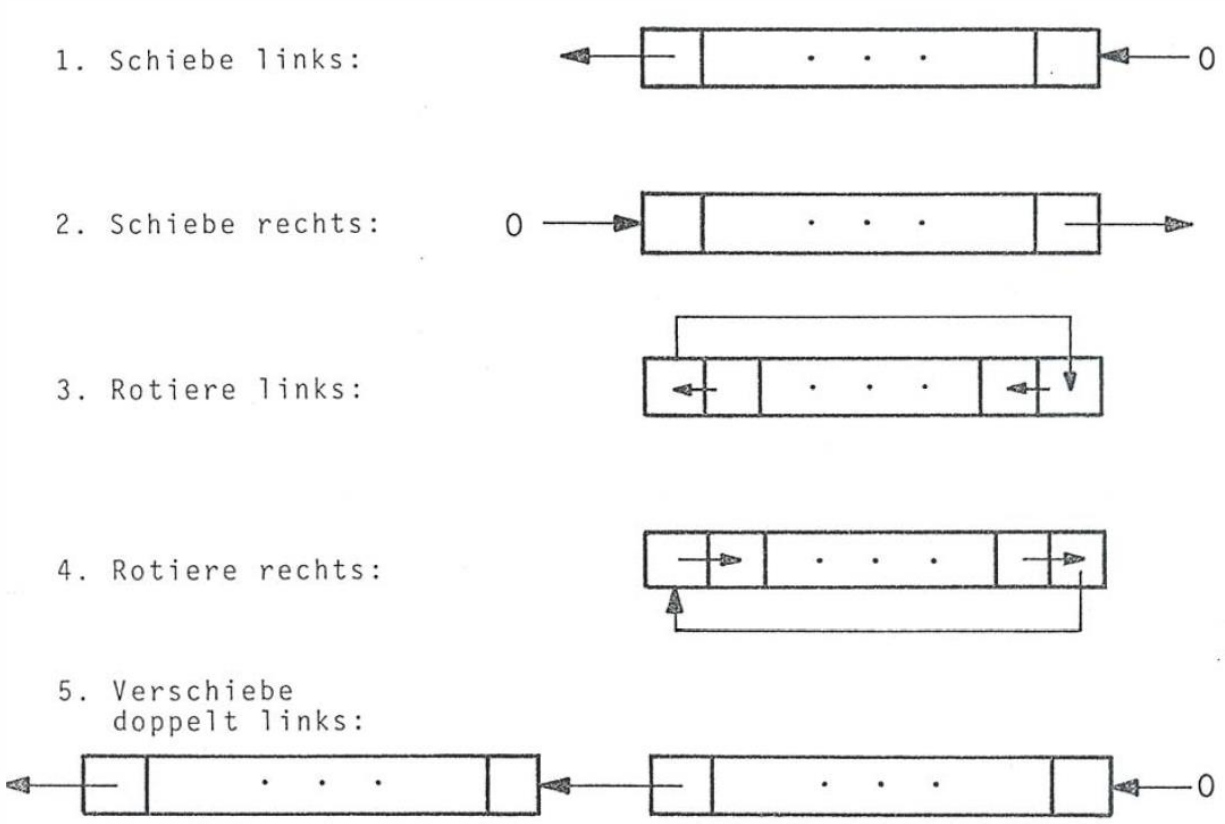
1. Angleichung der Exponenten:
    - Der kleinere Exponent wird an den größeren durch Rechtsverschiebung angeglichen. Falls die Differenz zwischen beiden größer ist als die Mantissenlänge, werden alle signifikanten Stellen heraus geschoben, so dass die Summe gleich der größeren Zahl ist.
2. Addition der Mantissen:
    - Die Mantissen werden als ganze Zahlen interpretiert und entsprechend addiert.
3. Abschlussbehandlung:
    - Abhängig von den Vorzeichen der beiden Operanden werden im letzten Schritt Korrekturen am Ergebnis vorgenommen (Bereichsüberlauf etc.).

Schiebebefehle

- Die meisten Rechner verfügen über zwei Arten von Schiebeoperationen (Shift):
  - logische: Hierbei wird der Inhalt eines Registers oder einer Speicherzelle als Bitfolge aufgefasst
  - arithmetische: Hierbei wird der Inhalt eines Registers als Binärzahl in 2-Komplement-Darstellung aufgefasst
- Die Verschiebungen erfolgen je nach Rechner entweder grundsätzlich um eine Stelle oder aber um beliebig viele Stellen.

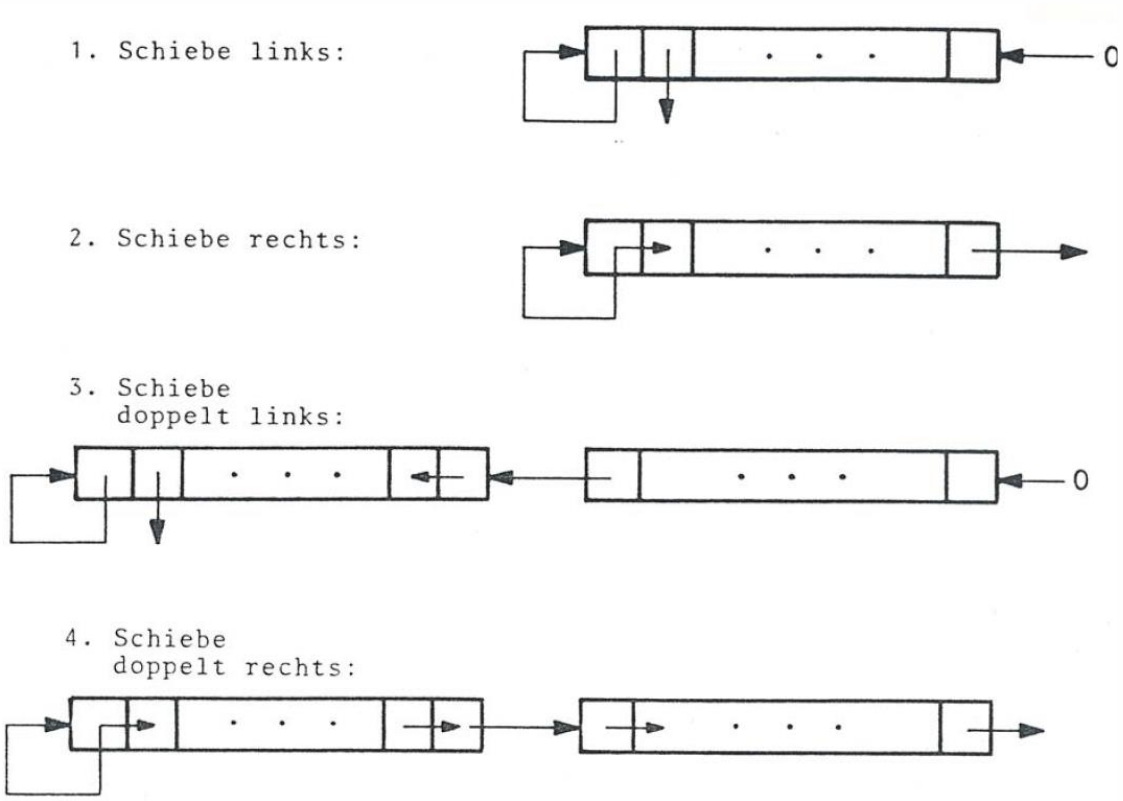
Logische Verschiebungen

- Die Verschiebungen erfolgen je nach Rechner entweder grundsätzlich um eine Stelle oder aber um beliebig viele Stellen.



Arithmetische Verschiebungen

- Bei beiden Schiebeoperationen bleibt das Vorzeichenbit erhalten.
- Die Verschiebung um eine Stelle nach links entspricht demnach einer Multiplikation mit 2,
- Die Verschiebung nach rechts entspricht einer (ganzzahligen) Division durch 2.



Vergleichsbefehle

- Vergleichsbefehle ermöglichen ein Setzen von Anzeigen, ohne dass das Ergebnis des Vergleichs wie bei anderen Anweisungen explizit abgespeichert werden muss. Sie haben damit große Bedeutung für die weiter unten beschriebenen Sprungbefehle.
- Vergleichsbefehle kann man ganz allgemein so charakterisieren, dass sie auf Hauptspeicherzellen und/oder Registern oder Teilen davon (durch entsprechende Masken ausgewählt) operieren und deren Inhalt im Sinne von Bitfolgen, ganzen Zahlen o.ä. miteinander vergleichen, d.h. entweder eine logische Verknüpfung oder eine Subtraktion durchführen.

Befehle zur Steuerung des Programmablaufs

- Maschinenanweisungen werden sequenziell ausgeführt, basierend auf ihrer Anordnung im Hauptspeicher.
- Es gibt spezielle Anweisungen, um diese Sequenz zu unterbrechen und die Interpretation zu einer anderen Anweisung zu verschieben. Sprungbefehle ermöglichen die Steuerung des Programmablaufs.
- Es gibt unbedingte und bedingte Sprünge auf Maschinensprachebene.
- Bei bedingten Sprüngen hängt die Veränderung des Programmzählers vom Zustand verschiedener Statusanzeigen oder explizit angegebener Vergleichsoperationen ab. Zum Beispiel:
    - Springe, falls Ergebnis gleich null
    - Springe, falls Ergebnis kleiner null
    - Springe, falls erster Operand kleiner als zweiter
    - Springe, falls Anzeige mit Maske xxx übereinstimmt, etc.

Unterprogrammsprünge

Unterprogrammsprünge ermöglichen die Steuerung des Programmablaufs und erfordern eine Rückkehr zur Sprungstelle.

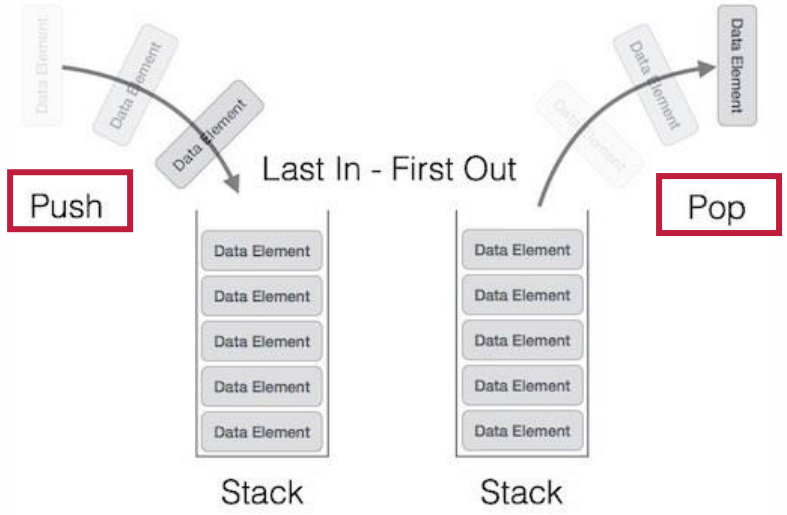
Es gibt verschiedene Möglichkeiten, die Rückkehradresse zu speichern:

a) Stapelspeicher für geschachtelte Unterprogrammaufrufe.

b) Register für die Rücksprungadresse, erfordert Aufbewahrung aller alten Adressen bei geschachtelten Aufrufen.

c) Reservierte Speicherzelle am Anfang des Unterprogramms.

Rückkehr erfolgt entweder durch unbedingten Sprung oder spezielle Anweisung.



Stapelspeicher

Unterbrechungsanforderungen

Unterbrechungen, auch Interrupts genannt, sind implizite Unterprogrammaufrufe, die automatisch bei bestimmten Bedingungen auftreten können.

Es gibt drei Arten von Unterbrechungsursachen:

a) Interne Unterbrechungsursachen, z. B. Programmfehler wie Division durch Null, Bereichsüberschreitung oder undefinierter Operationscode.

b) Externe Unterbrechungsursachen, die von Ein-/Ausgabegeräten asynchron zum laufenden Programm durch Erzeugung eines Unterbrechungssignals verursacht werden.

c) Programmierte Unterbrechungen, bei denen spezielle Maschinenanweisungen verwendet werden, um Betriebssystemfunktionen von Benutzerprogrammen aus aufzurufen.

Programmierte Unterbrechungen ermöglichen eine Trennung von Betriebssystem und Benutzersoftware, da der Aufrufer nicht die genaue Anfangsadresse der aufzurufenden Prozedur kennen muss.

Behandlung von Unterbrechungen

Bei allen genannten Unterbrechungen müssen in einem Prozessor prinzipiell folgende Schritte realisierbar sein:

1. Der Prozessor wird durch ein Unterbrechungssignal veranlasst, nach Abarbeitung der gerade interpretierten Anweisung nicht zur sequentiell folgenden überzugehen.
2. Die Adresse dieser Anweisung, die im Befehlszähler steht, muss an einer vereinbarten Stelle im Hauptspeicher bzw. auf einem Stack (falls ein solcher vorhanden ist) gespeichert werden, um eine spätere Fortsetzung des unterbrochenen Programmes zu ermöglichen.
3. Die Anfangsadresse der Unterbrechungsbearbeitungsroutine muss an einer fest vereinbarten Stelle im Hauptspeicher verfügbar sein.
4. Für diese Routine müssen Hinweise über die Unterbrechungsursache im Hauptspeicher oder in speziellen Registern zur Verfügung stehen.
5. Es sollte möglich sein, die Annahme von Unterbrechungssignalen für eine gewisse Zeit zu unterbinden (maskieren).

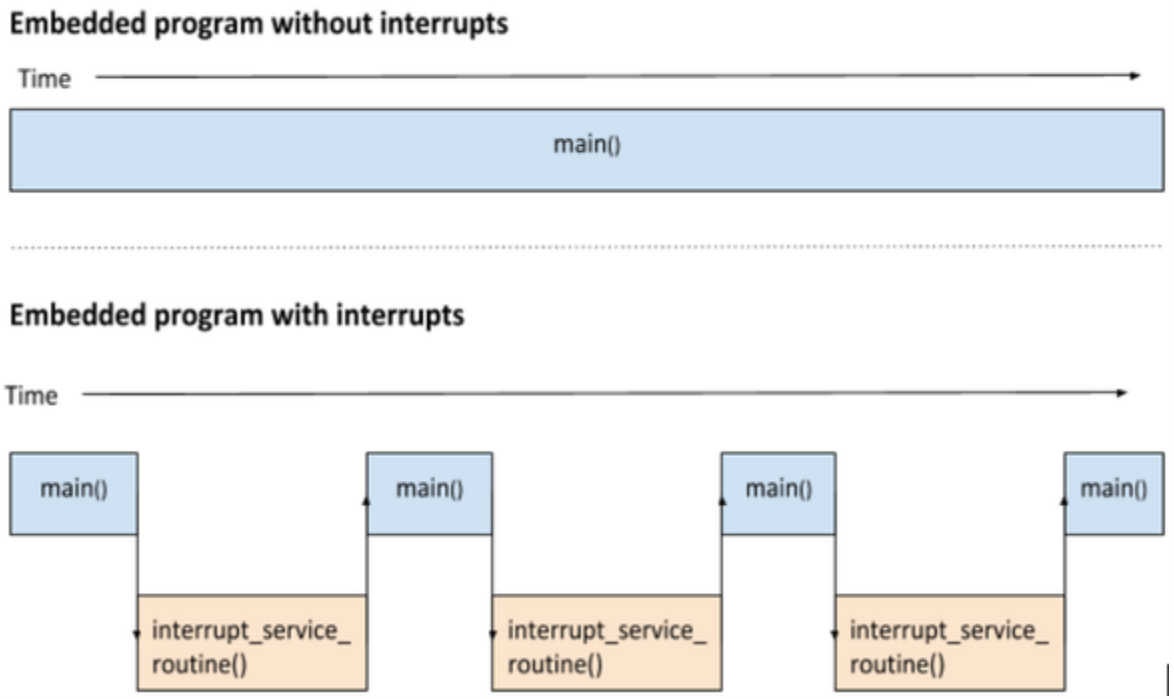
Unterbrechungsanforderungen

Überlappende Unterbrechungen erfordern die Befolgung des gleichen Verfahrens.

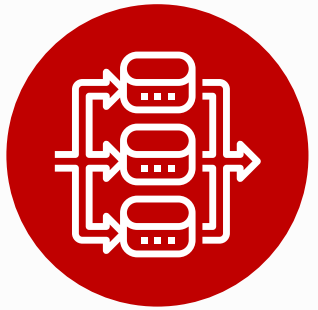
Ein Prioritätsschema regelt die Annahme solcher Signale.

Jedes Unterbrechungssignal wird mit einer Priorität versehen.

Nur Unterbrechungssignale mit einer höheren Priorität als die aktuell bearbeitete Unterbrechungsursache haben eine Auswirkung.



Instruction Set Architecture



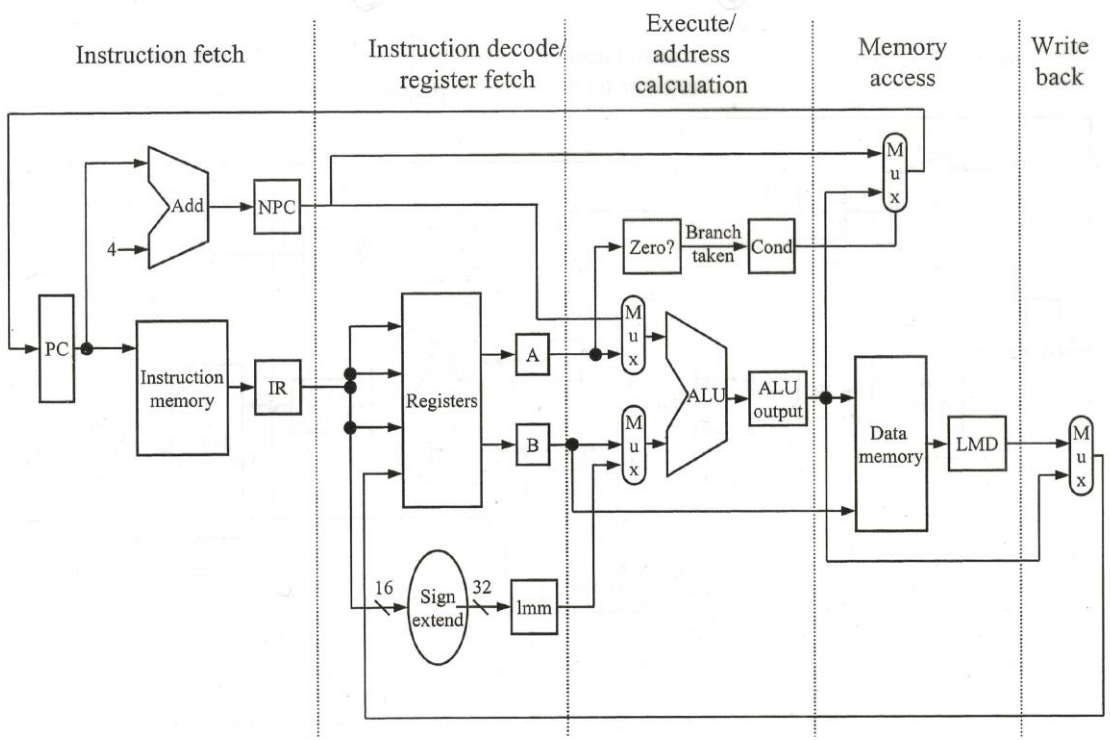
RISC vs. CISC

Die Entwicklung der Mikroprozessoren ist bis zur Mitte der 80er Jahre gekennzeichnet durch Maschinen mit immer komplexeren microcodierten Instruktionen (Complex Instruction Set Computer, CISC)

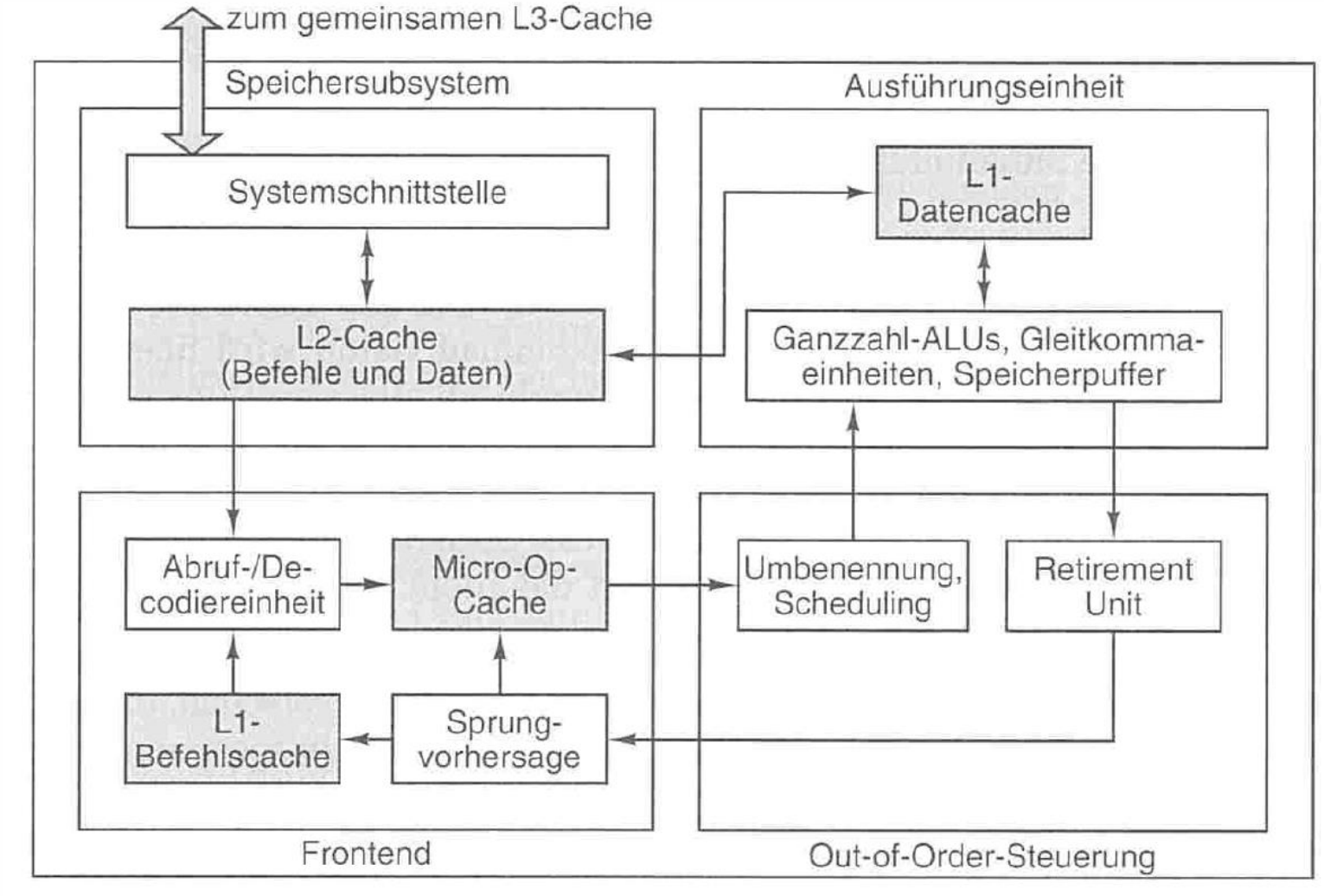
Die einfachen RISC (Reduced Instruction Set Computer) Maschinen setzen dagegen auf schnelle Verarbeitung von wenigen, direkt verdrahtbaren Instruktionen und benötigen zur optimierten Nutzung auch optimierende Compiler

RISC-Beispiel DLX Prozessor

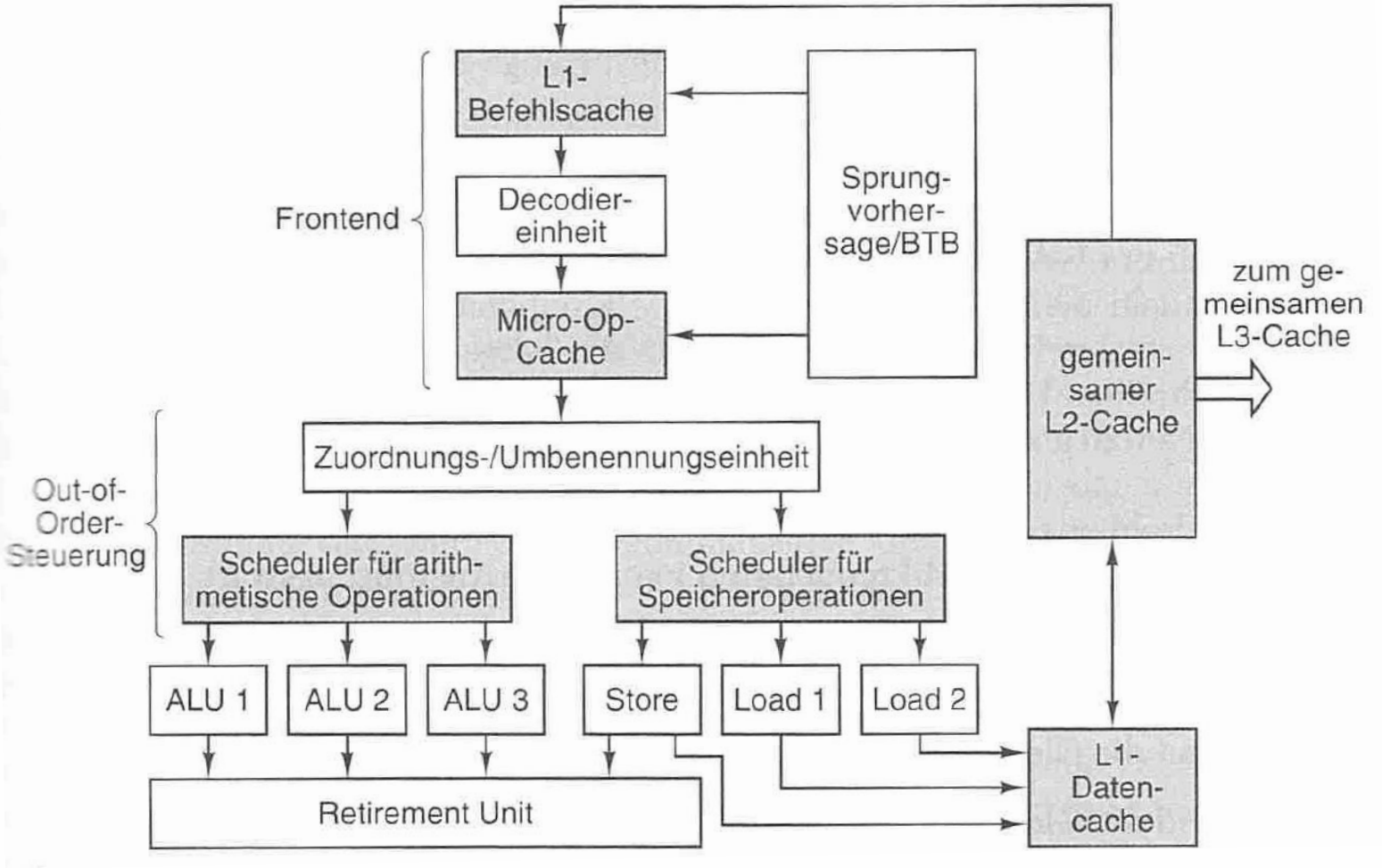
Das folgende Bild zeigt den vereinfachten Datenpfad der CPU (ohne Steuerwerk und für einfache load/store und registerbasierte Instruktionen. Er besteht aus den Hauptkomponenten Instruktionsspeicher, Registerbank, ALU und Datenspeicher. Dieser Aufbau ermöglicht eine Befehlsabarbeitung in mehreren Stufen, die in Form einer Befehls-Pipeline parallelisiert erfolgt.



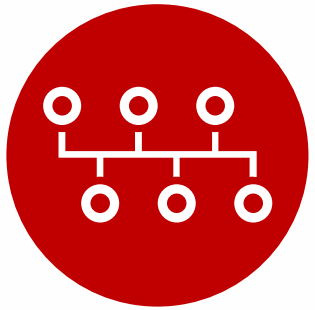
CISC Beispiel: Intel Core i7 Blockdiagramm



CISC Beispiel: Intel Core i7 Datenpfad



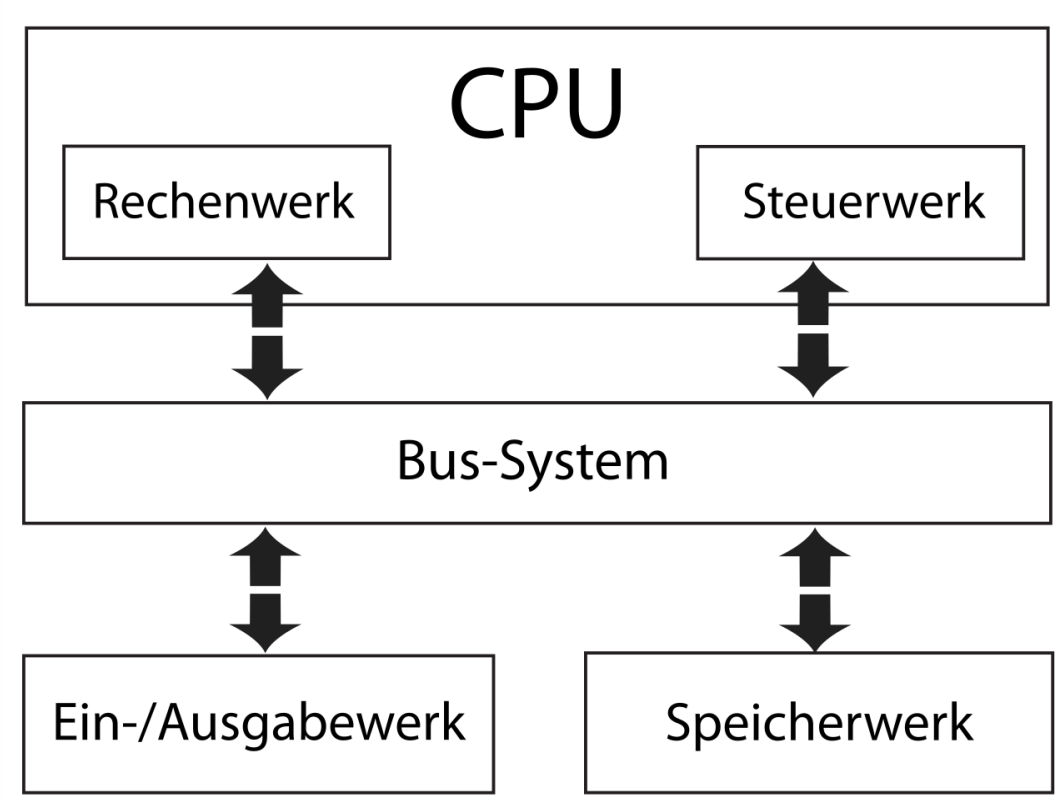
Systembus



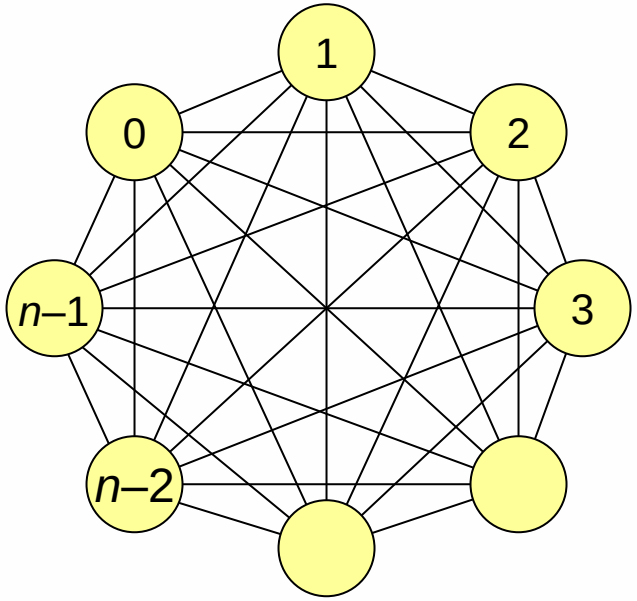
Begriffsbestimmung

Der **Systembus** – ein (quasi) standardisiertes Leitungssystem.

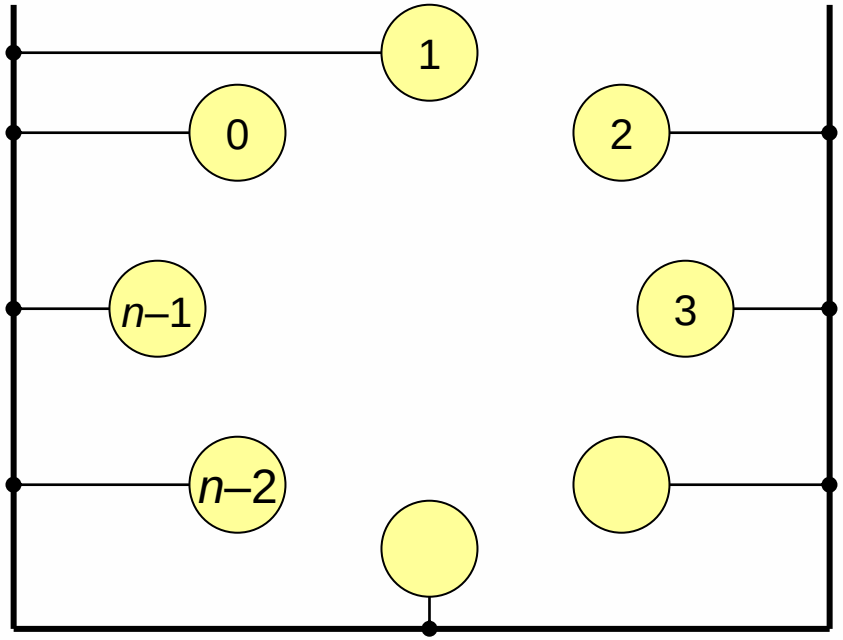
Der **Systembus** dient als Verbindungssystem für Einzelkomponenten innerhalb und außerhalb von Rechner-Systemen und/oder für die Verbindung kompletter Systeme.



Busse und ihre Vorteile



Punkt-zu-Punkt-Verbindungen zwischen n Einheiten erfordern n(n - 1) Kanäle, oder n(n - 1)/2 bidirektionale Verbindungen, d. h. $O(n^2)$ Verbindungen



Die Busverbindung erfordert nur einen Eingangs- und einen Ausgangsanschluss pro Einheit oder O(n) Verbindungen in allen

Kommunikation - Allgemeines

Der Datenaustausch kann erfolgen:

- Seriell oder parallel
- Synchron oder asynchron
- Monodirektional oder bidirektional
- Bidirektionale Kommunikation kann vollduplex oder halbduplex sein
- Kann einem Standard- oder benutzerdefinierten Protokoll folgen

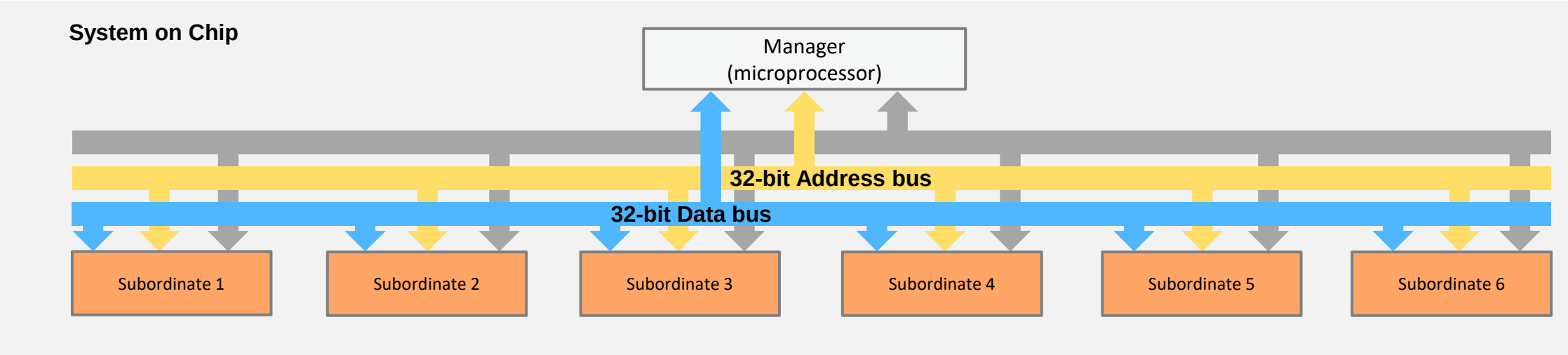
Busbetrieb im Allgemeinen

Ein Bus besteht in der Regel aus drei Arten von Signalleitungen:

Der Datenbus wird für den Austausch von Dateninformationen verwendet.

Der Adressbus dient der Auswahl eines der Peripheriegeräte (oder eines Registers eines Peripheriegeräts).

Steuersignale werden zur Synchronisierung und Identifizierung von Transaktionen verwendet, z. B. Bereitschafts-, Schreib-/Lese- und Übertragungsmodus-Signale.



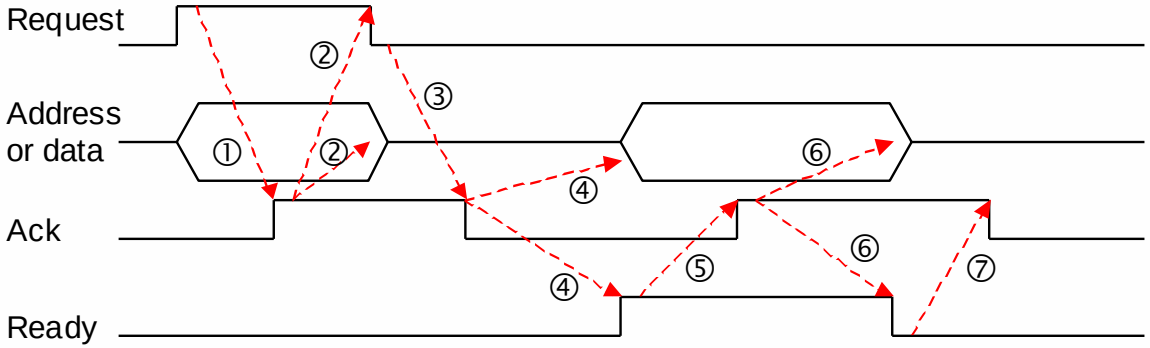
Synchronisationstechniken

Die Datenübertragung zwischen CPU und einer Schnittstelle, z.B. zum Speicher oder zur Eingabe/Ausgabe (I/O), muss synchronisiert werden. (typisch: langsames I/O oder langsamer externer Speicher)

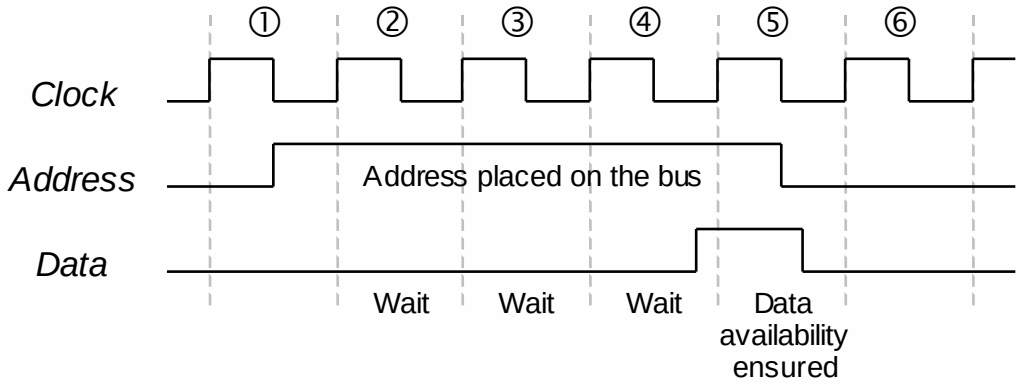
Die Synchronisation erfolgt

a) durch Steuersignale (Control Bus) oder/und in die Daten integrierte Steuerinformation

b) implizit durch auf einen Synchronisationstakt bezogene Vereinbarungen



Handshaking auf einem asynchronen Bus für eine Eingabeoperation (z. B. Lesen aus dem Speicher).



Synchroner Bus mit Geräten mit fester Latenzzeit.

Buszuteilung (Arbitration)

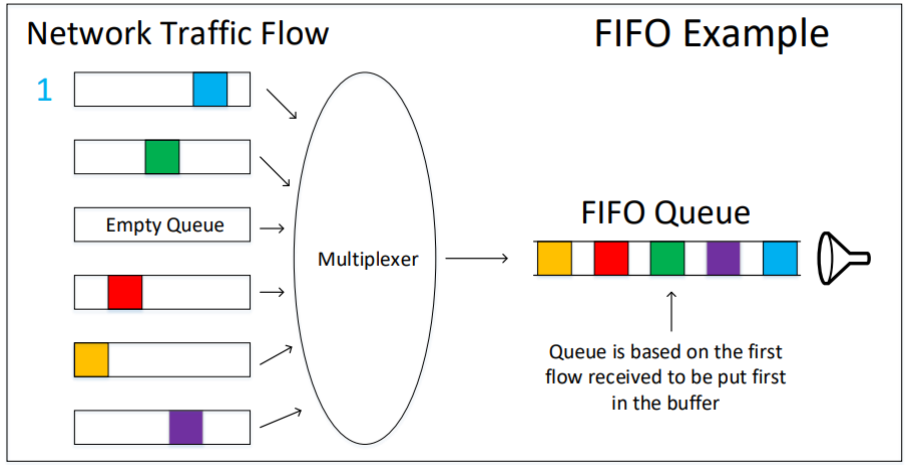
Die Ressource Bus muss bei üblicherweise mehrfach vorhandenen gleichberechtigten Nutzern (Busmaster) zugeteilt werden

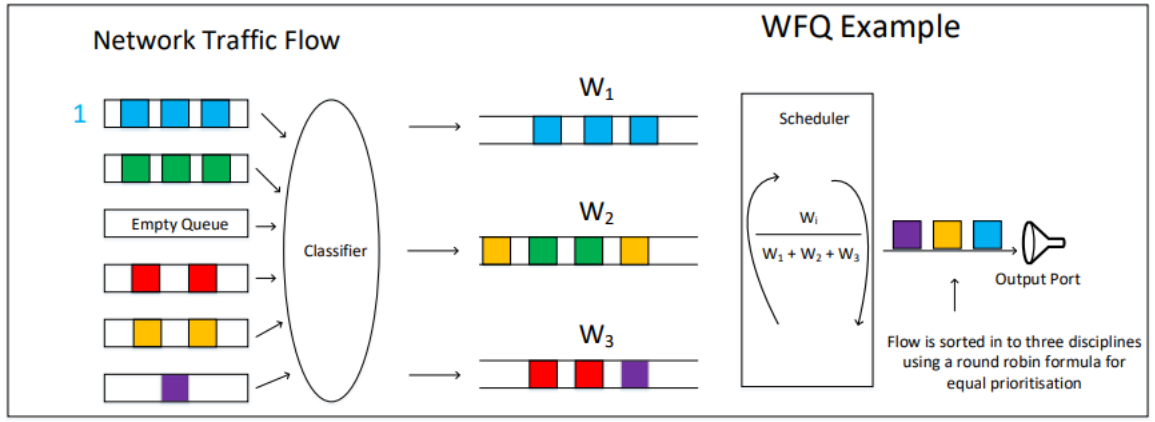
Die Zuteilung kann **zentral** durch einen Busarbiter erfolgen oder **dezentral** auf die Einheiten verteilt werden.

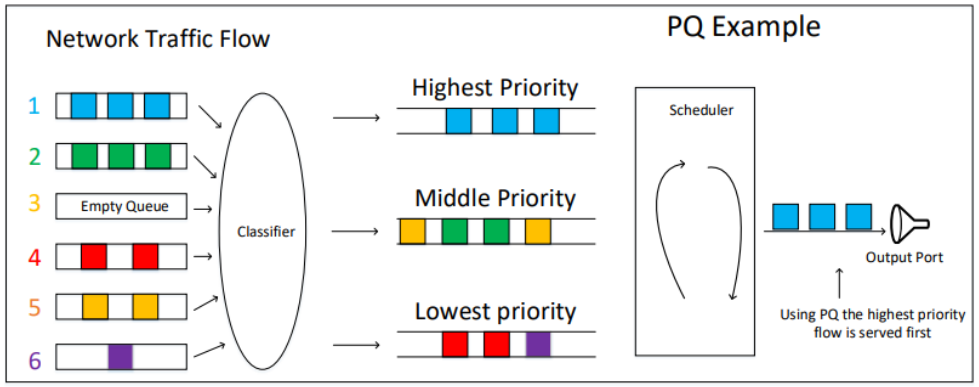
Die Zuteilung kann nach verschiedenen Verfahren erfolgen, z.B:

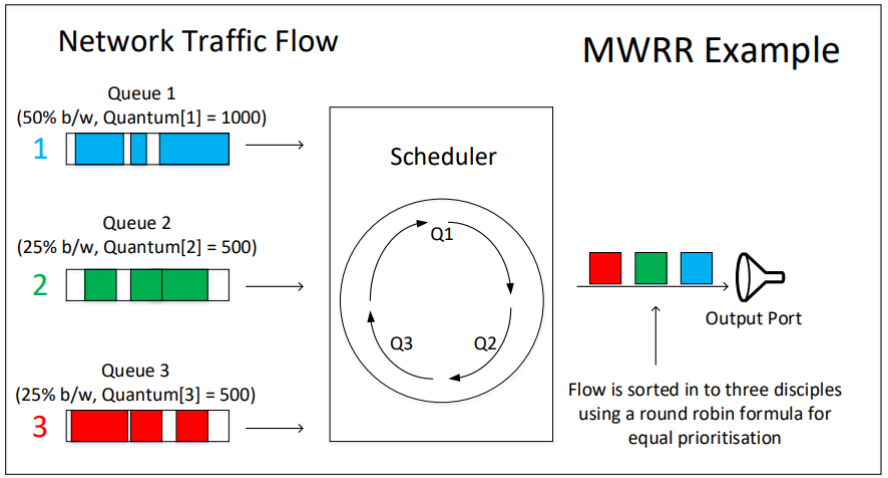
- First come, first serve (Anmeldeverfahren)
- Round Robin (Zeitschlitzverfahren)
- Priority-based (Prioritätverfahren)
- Polling (Abfrage durch den Master)
- Token passing (Weitergabeverfahren)

Einige einfache Busarbiter







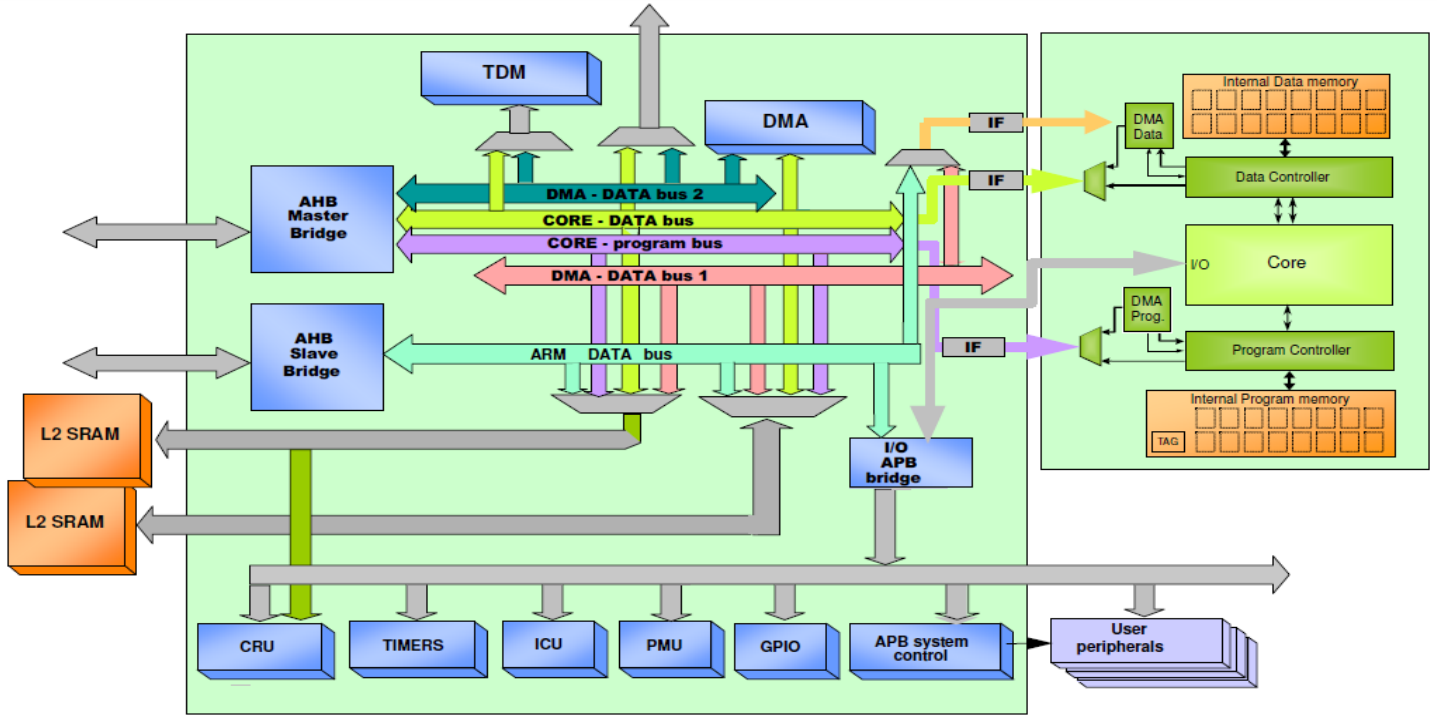


Source: Williams-Kinnaird, Callum. (2019). The Impact of Advanced QoS Mechanisms on the Performance of Converged Networks. 10.13140/RG.2.2.10803.78883.

Onchip-Kommunikationsstandards

Warum brauchen wir Kommunikationsstandards?

- Modularer Entwurfsansatz
- Ermöglicht die Wiederverwendung von Designs
- Erleichtert die IP-Integration in ein SoC-Design



Picture source: http://www.ecs.soton.ac.uk/ (SoC Advance design Technique)

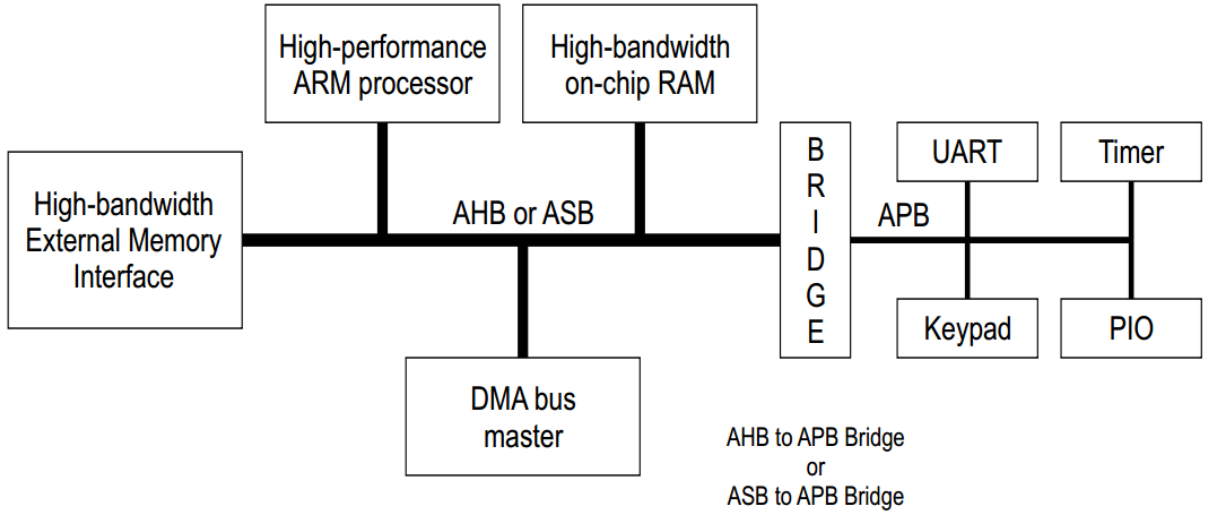
AMBA: Advanced microcontroller bus architecture

Der Schnittstellenstandard von ARM ermöglicht die Wiederverwendung von IP-Blöcken.

Erleichtert die Entwicklung von Multiprozessor-Designs mit einer großen Anzahl von Controllern und Peripheriegeräten auf Anhieb.

Unterstützt verschiedene Busbreiten 64/128/256/512/1024 bit

Weit verbreitet in modernen tragbaren mobilen Geräten, wie Tablets und Smartphones



AMBA AHB:

- Hohe Leistung
- Pipeline-Betrieb
- Mehrere Bus-Master
- Burst-Übertragungen

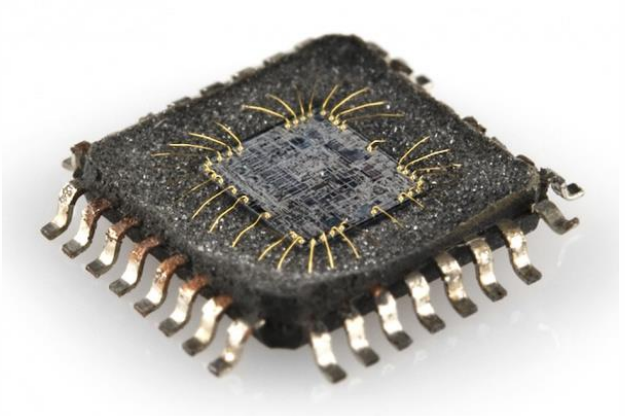
AMBA APB:

- Niedriger Stromverbrauch
- Latched Adresse und Steuerung
- Einfache Schnittstelle

Off-Chip-Kommunikation

Serielle Busse übertragen Daten Bit für Bit in einer sequentiellen Weise.

Beispiele sind UART, SPI, I2C, USB, Ethernet, PCI Express usw.



Parallele Busse übertragen mehrere Bits gleichzeitig

Beispiele: PATA, SCSI, IEEE 1284, GPIO

Kompromisse bei der Offchip-Kommunikation

Kosten und Gewicht

Die serielle Kommunikation ist kostengünstiger und leichter als die parallele Kommunikation, da weniger Kabel und kleinere Stecker benötigt werden.

Die serielle Kommunikation ist zuverlässiger.

Bei der parallelen Kommunikation kann es zu größeren Taktabweichungen und zum Übersprechen zwischen den verschiedenen Leitungen kommen.

Höhere Taktrate

Aufgrund der höheren Zuverlässigkeit kann die serielle Kommunikation mit einer höheren Frequenz getaktet werden als die parallele Kommunikation, wodurch sich der Durchsatz erhöht.

Die Konvertierung zwischen seriellen und parallelen Daten kann zusätzlichen Overhead verursachen.

PCI Express

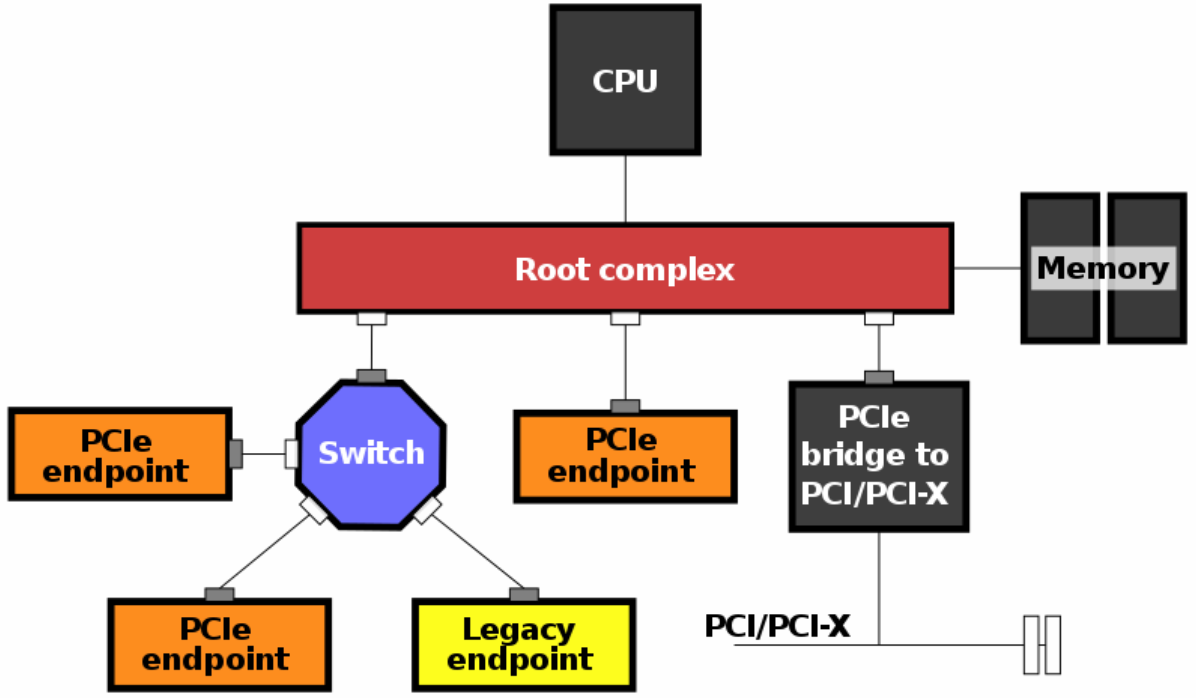
PCIe (Peripheral Component Interconnect Express) ist ein serieller Hochgeschwindigkeitsbus, die Weiterentwicklung des älteren PCI-Busses.

Paketbasierten Netzwerk mit Switches

Punkt zu Punkt Verbindung mit PCI sehr leitungsaufwändig

Kompatibilität zu PCI nur in den höheren Protokollebenen notwendig

Serielle Verbindung über differentielles Leitungspaar: 1 Lane = 4 Leitungen (2*Hin- und 2*Rückkanal)



Serial Peripheral Interface (SPI) Bus

Erfunden von Motorola

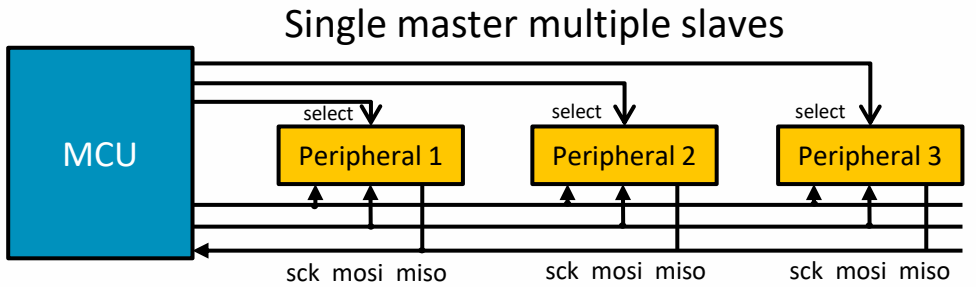
Synchrone Kommunikation, Taktleitung erforderlich

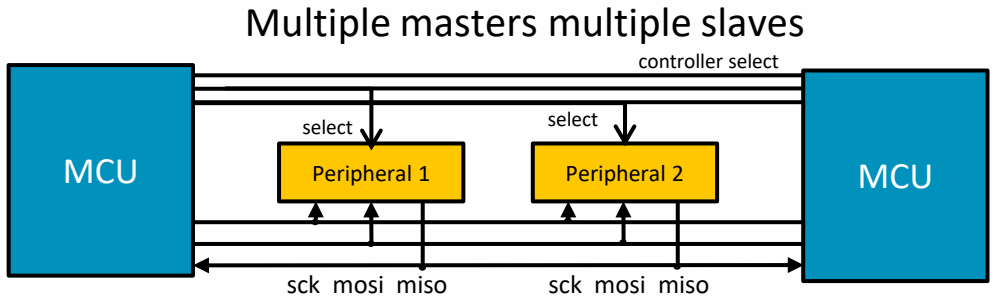
Getrennte Sende- und Empfangsdrähte

Serieller Vier-Draht-Bus (CLK, MOSI, MISO, CS)

Geräte kommunizieren im Master/Slave-Modus

- Master initiiert Datenübertragungen
- Mehrere Slaves werden einzeln über die Chip-Select-Leitungen (CS) aktiviert





Inter-integrated Circuit (I2C) Bus

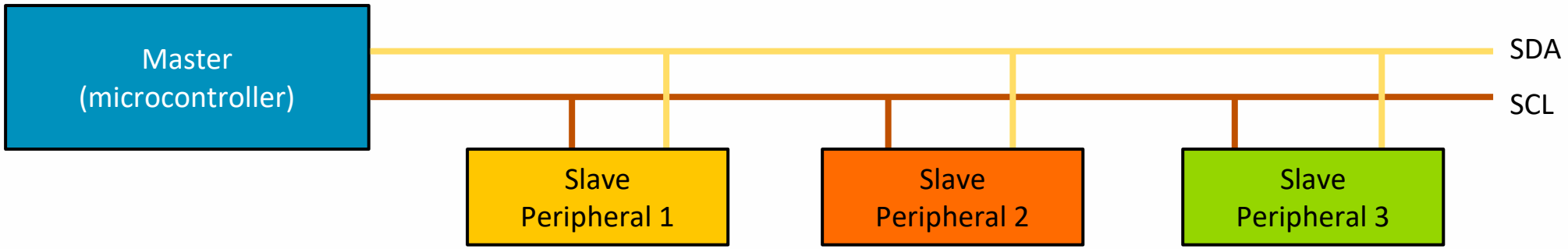
Serieller Computerbus mit mehreren Controllern

Erfunden von der Halbleiterabteilung von Philips (heute: NXP Semiconductors)

Kommuniziert mit Peripheriegeräten niedriger Geschwindigkeit

Zwei Signalleitungen

- SCL: Serieller Takt
- SDA: Serielle Daten



I²C Bus Connections

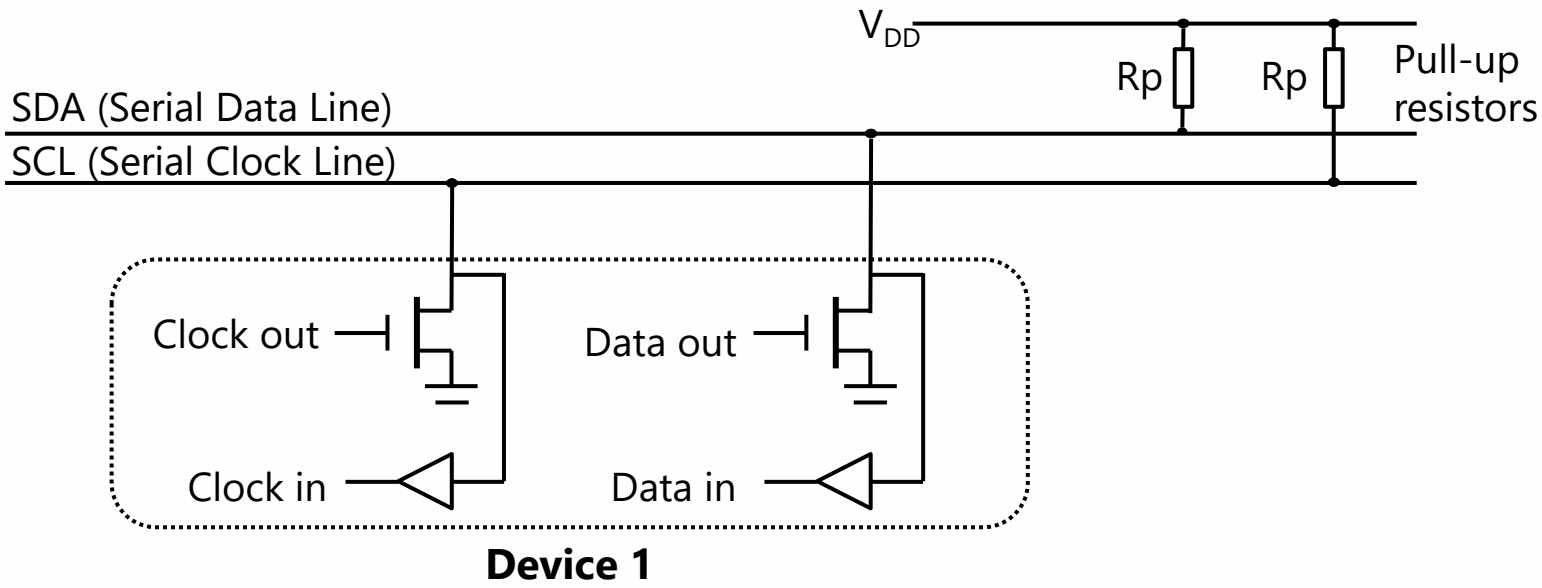
Der Bus wird in der Regel vom Master gesteuert; die Slaves reagieren, wenn sie angesprochen werden.

Widerstände ziehen die Leitungen nach oben zu VDD

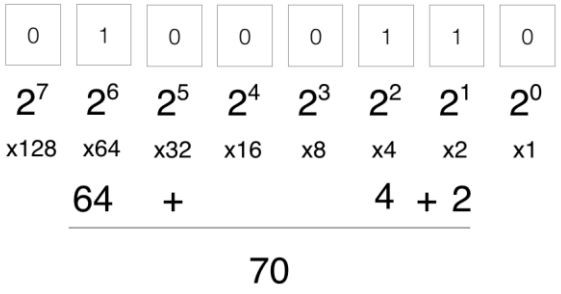
Open-Drain-Transistoren ziehen die Leitungen nach unten zur Masse

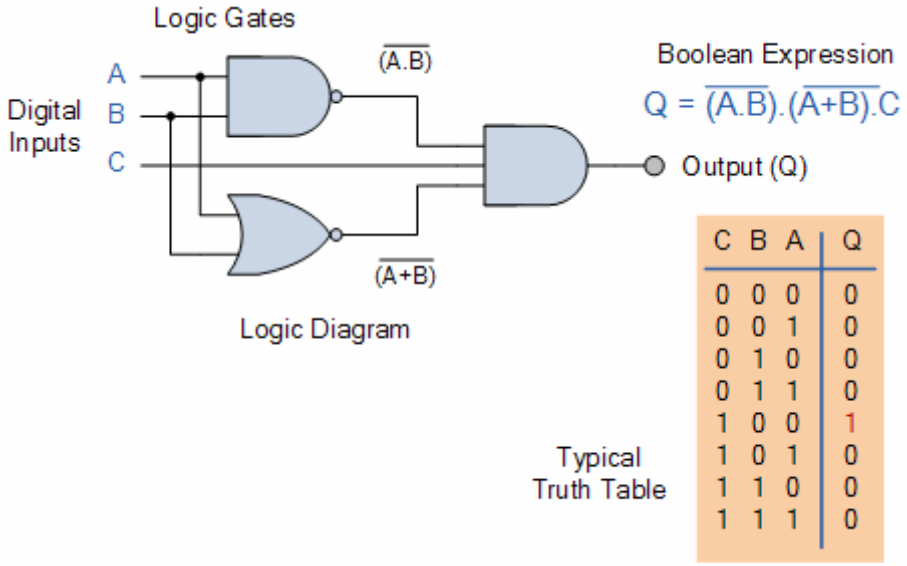
Controller erzeugt SCL-Taktsignal

Kann bis zu 400 kHz, 1 MHz oder mehr betragen



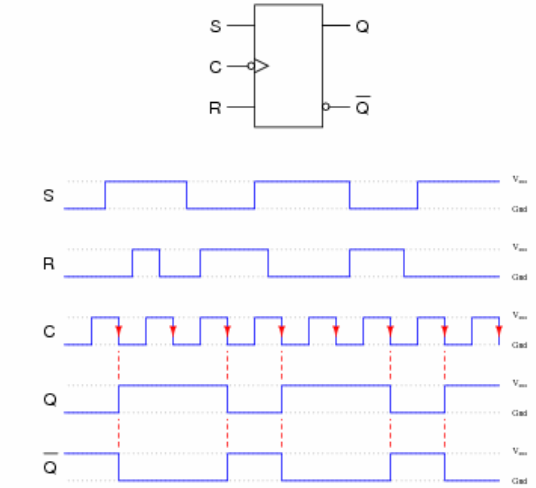
Inhalt



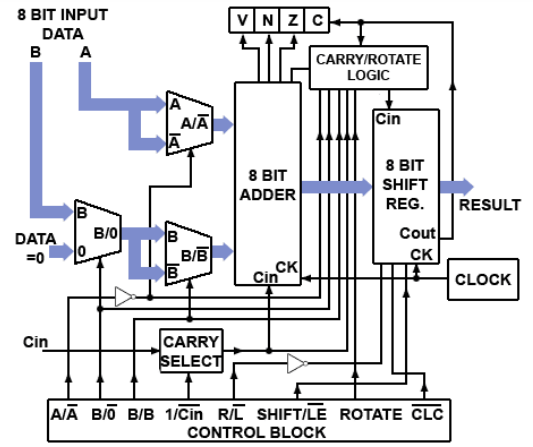


Inhalt

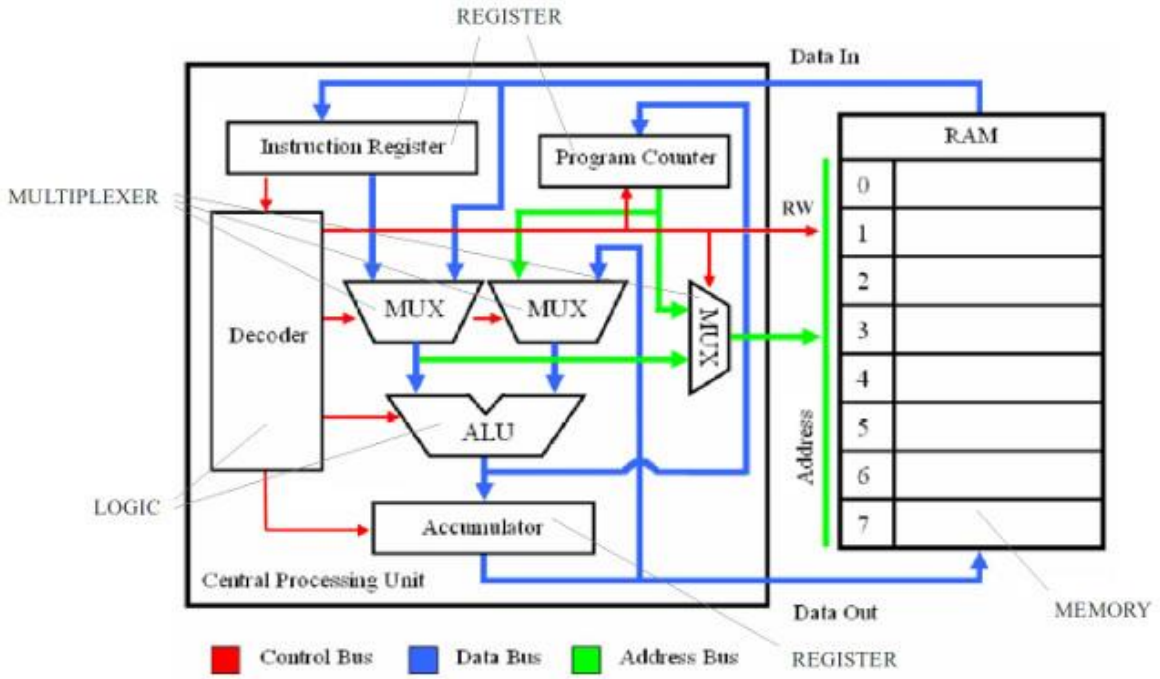
3. Entwurf von sequentieller Logik
   - Definition
   - Grundtypen
   - Elementare sequentielle Schaltungen
     - Flip-Flops
     - Zähler
   - Synthese von synchroner sequentieller Logik



4. Beispielimplementierung Rechenwerk (ALU)



Inhalt



Vielen Dank