



# 64-040 Modul InfB-RSB

## Rechnerstrukturen und Betriebssysteme

[https://tams.informatik.uni-hamburg.de/  
lectures/2020ws/vorlesung/rsb](https://tams.informatik.uni-hamburg.de/lectures/2020ws/vorlesung/rsb)

– Kapitel 11 –

Andreas Mäder



Universität Hamburg  
Fakultät für Mathematik, Informatik und Naturwissenschaften  
Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

Wintersemester 2020/2021



## Rechnerarchitektur I

Motivation

von-Neumann Rechner

Beschreibungsebenen

Software

HW Abstraktionsebenen

Hardwarestruktur

Speicherbausteine

Busse

Mikroprogrammierung

Beispielsystem: ARM

Wie rechnet ein Rechner?

Literatur



## Definitionen

1. *The term architecture is used here to describe the attributes of a system as seen by the programmer, i.e., the conceptual structure and functional behaviour, as distinct from the organization and data flow and control, the logical and the physical implementation. [Amdahl, Blaauw, Brooks]*
2. *The study of computer architecture is the study of the organization and interconnection of components of computer systems. Computer architects construct computers from basic building blocks such as memories, arithmetic units and buses.*

# Was ist Rechnerarchitektur? (cont.)

*From these building blocks the computer architect can construct anyone of a number of different types of computers, ranging from the smallest hand-held pocket-calculator to the largest ultra-fast super computer. The functional behaviour of the components of one computer are similar to that of any other computer, whether it be ultra-small or ultra-fast.*

*By this we mean that a memory performs the storage function, an adder does addition, and an input/output interface passes data from a processor to the outside world, regardless of the nature of the computer in which they are embedded. The major differences between computers lie in the way of the modules are connected together, and the way the computer system is controlled by the programs. In short, computer architecture is the discipline devoted to the design of highly specific and individual computers from a collection of common building blocks. [Stone]*

## 1. Operationsprinzip:

das funktionelle Verhalten der Architektur

Befehlssatz

- = Programmierschnittstelle
- = ISA – **I**nstruction **S**et **A**rchitecture  
Befehlssatzarchitektur
- = Maschinenorganisation: *Wie werden Befehle abgearbeitet?*

→ folgt ab Kapitel 12 *Instruction Set Architecture*

## 2. Hardwarearchitektur:

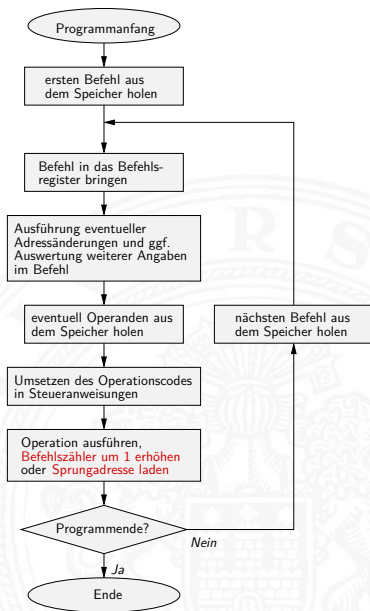
der strukturelle Aufbau des Rechnersystems

Mikroarchitektur

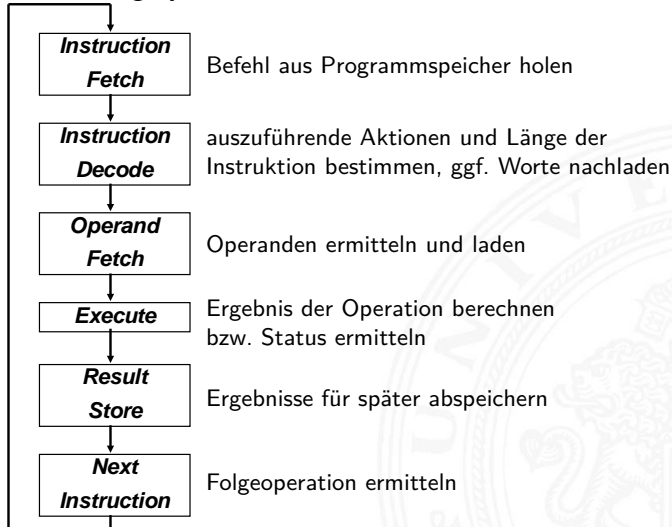
- = Art und Anzahl der Hardware-Betriebsmittel +  
die Verbindungs- / Kommunikationseinrichtungen
- = (technische) Implementierung

- ▶ J. Mauchly, J.P. Eckert, J. von-Neumann 1945
  - ▶ Abstrakte Maschine mit minimalem Hardwareaufwand
    - ▶ System mit Prozessor, Speicher, Peripheriegeräten
    - ▶ die Struktur ist unabhängig von dem Problem, das Problem wird durch austauschbaren Speicherinhalt (Programm) beschrieben
  - ▶ gemeinsamer Speicher für Programme und Daten
    - ▶ fortlaufend adressiert
    - ▶ Programme können wie Daten manipuliert werden
    - ▶ Daten können als Programm ausgeführt werden
  - ▶ Befehlszyklus: Befehl holen, decodieren, ausführen
- ⇒ enorm flexibel
- ▶ **alle** aktuellen Rechner basieren auf diesem Prinzip
  - ▶ aber vielfältige Architekturvarianten, Befehlssätze usw.

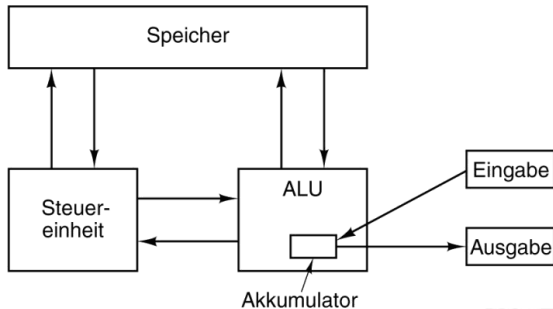
- ▶ Programm als Sequenz elementarer Anweisungen (Befehle)
- ▶ als Bitvektoren im Speicher codiert
- ▶ Interpretation (Operanden, Befehle und Adressen) ergibt sich aus dem Kontext (der Adresse)
- ▶ zeitsequenzielle Ausführung der Instruktionen



## ► Ausführungszyklus







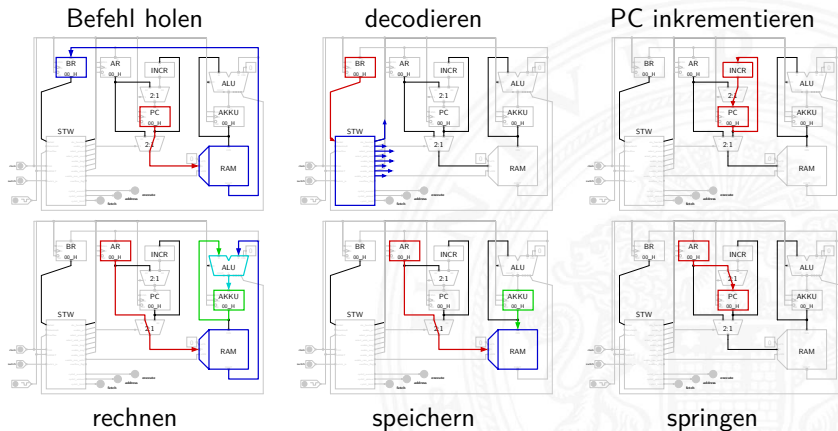
[TA14]

Fünf zentrale Komponenten:

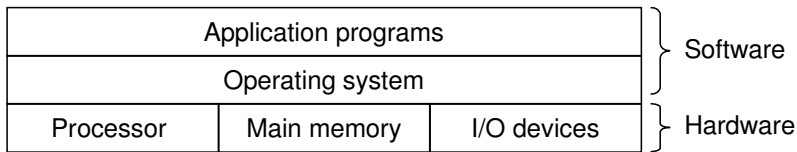
- ▶ Prozessor mit **Steuerwerk** und **Rechenwerk** (ALU, Register)
- ▶ **Speicher**, gemeinsam genutzt für Programme und Daten
- ▶ **Eingabe-** und **Ausgabewerke**
- ▶ verbunden durch Bussystem

- ▶ Prozessor (CPU) = Steuerwerk + Operationswerk
- ▶ Steuerwerk: zwei zentrale Register
  - ▶ Befehlszähler (*program counter PC*)
  - ▶ Befehlsregister (*instruction register IR*)
- ▶ Operationswerk (Datenpfad, *data-path*)
  - ▶ Rechenwerk (*arithmetic-logic unit ALU*)
  - ▶ Universalregister (mind. 1 *Akkumulator*, typisch 8...64 Register)
  - ▶ evtl. Register mit Spezialaufgaben
- ▶ Speicher (*memory*)
  - ▶ Hauptspeicher/RAM: *random-access memory*
  - ▶ Hauptspeicher/ROM: *read-only memory* zum Booten
  - ▶ Externspeicher: Festplatten, CD/DVD, Magnetbänder
- ▶ Peripheriegeräte (Eingabe/Ausgabe, *I/O*)

- ▶ Verschaltung der Hardwarekomponenten für alle mögl. Datentransfers
- ▶ abhängig vom Befehl werden nur bestimmte Pfade aktiv
- ▶ Ausführungszyklus

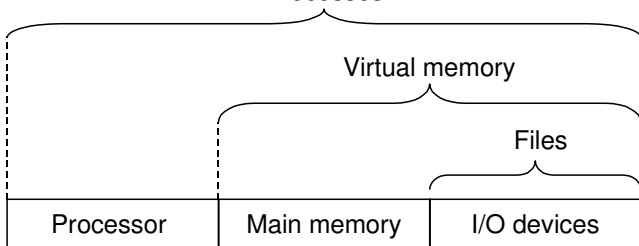


- ▶ Schichten-Ansicht: Software – Hardware

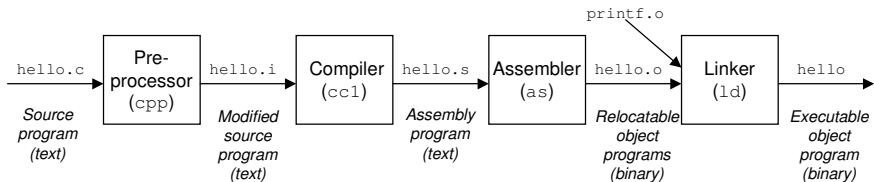


[BO15]

- ▶ Abstraktionen durch Betriebssystem  
Processes



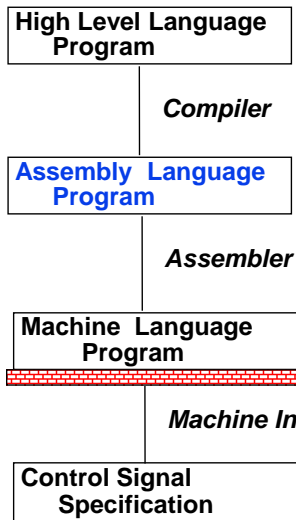
[BO15]



[BO15]

- ▶ verschiedene Repräsentationen des Programms
  - ▶ Hochsprache
  - ▶ Assembler
  - ▶ Maschinensprache
- ▶ Ausführung der Maschinensprache
  - ▶ von-Neumann Zyklus: Befehl holen, decodieren, ausführen
  - ▶ reale oder virtuelle Maschine

# Das Compilierungssystem (cont.)



```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```

```
lw $15, 0($2)  
lw $16, 4($2)  
sw $16, 0($2)  
sw $15, 4($2)
```

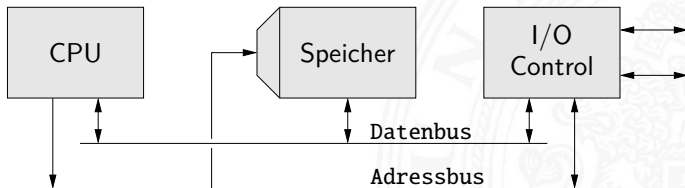
```
0000 1001 1100 0110 1010 1111 0101 1000  
1010 1111 0101 1000 0000 1001 1100 0110  
1100 0110 1010 1111 0101 1000 0000 1001  
0101 1000 0000 1001 1100 0110 1010 1111
```

```
ALUOP[0:3] <= InstReg[9:11] & MASK
```

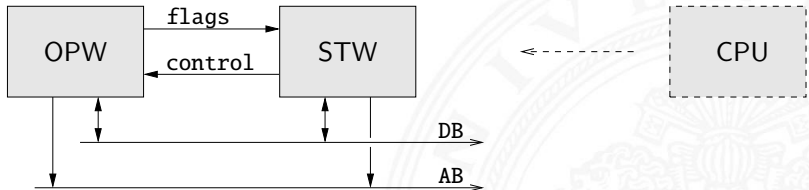
[PH16]

## Hardware Abstraktionsebenen

- keine einheitliche Bezeichnung in der Literatur
- ▶ **Architekturebene**
  - ▶ Funktion/Verhalten Leistungsanforderungen
  - ▶ Struktur Netzwerk  
aus Prozessoren, Speicher, Busse, Controller ...
  - ▶ Nachrichten Programme, Prokollé
  - ▶ Geometrie Systempartitionierung

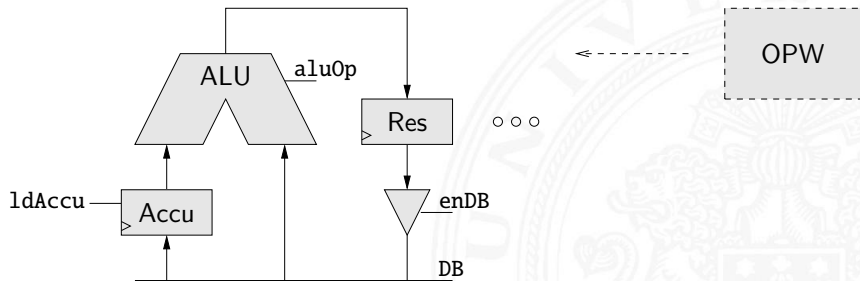


- ▶ Hauptblockebene (Algorithmenebene, funktionale Ebene)
  - ▶ Funktion/Verhalten Algorithmen, formale Funktionsmodelle
  - ▶ Struktur Blockschaltbild  
aus Hardwaremodule, Busse ...
  - ▶ Nachrichten Protokolle
  - ▶ Geometrie Cluster

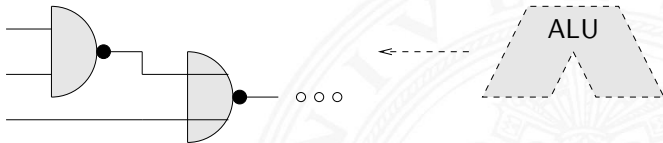




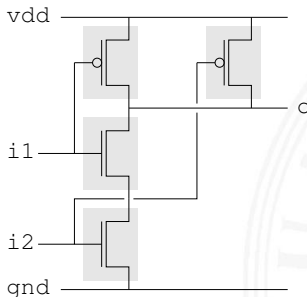
- ▶ Register-Transfer Ebene
  - ▶ Funktion/Verhalten Daten- und Kontrollfluss, Automaten ...
  - ▶ Struktur RT-Diagramm
    - aus Register, Multiplexer, ALUs ...
  - ▶ Nachrichten Zahlencodierungen, Binärworte ...
  - ▶ Geometrie Floorplan



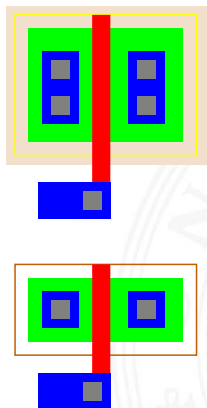
- ▶ Logikebene (Schaltwerkebene)
  - ▶ Funktion/Verhalten Boole'sche Gleichungen
  - ▶ Struktur Gatternetzliste, Schematic  
aus Gatter, Flipflops, Latches ...
  - ▶ Nachrichten Bit
  - ▶ Geometrie Moduln

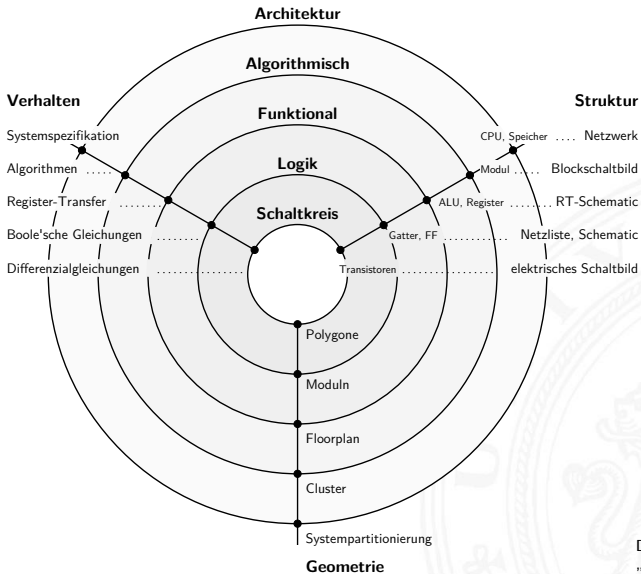


- ▶ elektrische Ebene (Schaltkreisebene)
  - ▶ Funktion/Verhalten Differentialgleichungen
  - ▶ Struktur elektrisches Schaltbild  
aus Transistoren, Kondensatoren ...
  - ▶ Nachrichten Ströme, Spannungen
  - ▶ Geometrie Polygone, Layout → physikalische Ebene



- ▶ physikalische Ebene (geometrische Ebene)
  - ▶ Funktion/Verhalten partielle DGL
  - ▶ Struktur Dotierungsprofile





D. Gajski, R. Kuhn 1983:  
„New VLSI Tools“ [GK83]



drei unterschiedliche Aspekte/Dimensionen:

1 Verhalten

2 Struktur (logisch)

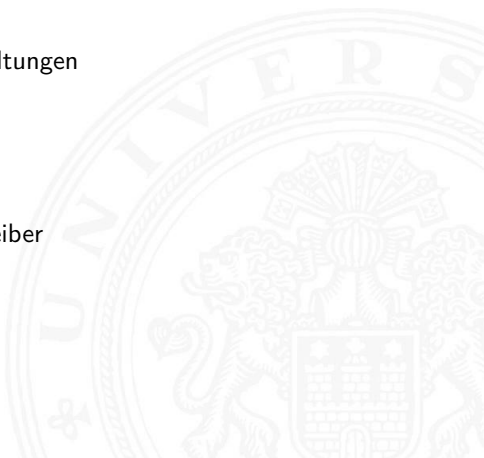
3 Geometrie (physikalisch)

- ▶ Start möglichst abstrakt, z.B. als Verhaltensbeschreibung
- ▶ Ende des Entwurfsprozesses ist vollständige IC Geometrie für die Halbleiterfertigung (Planarprozess)
- ▶ Entwurfsprogramme („EDA“, *Electronic Design Automation*) unterstützen den Entwerfer: setzen Verhalten in Struktur und Struktur in Geometrien um



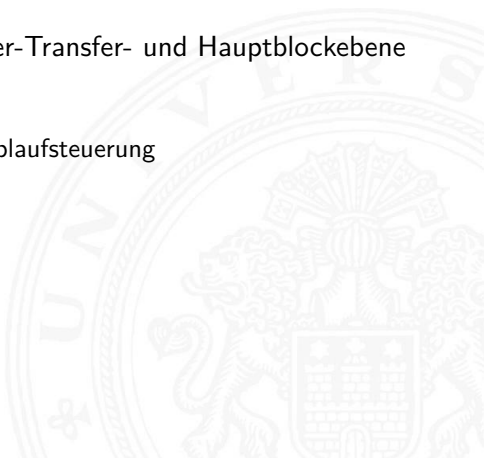
## Modellierung eines digitalen Systems als Schaltung aus

- ▶ speichernden Komponenten
  - ▶ Registern Flipflops, Register, Registerbank
  - ▶ Speichern SRAM, DRAM, ROM, PLA
- ▶ funktionalen Schaltnetzen
  - ▶ Addierer, arithmetische Schaltungen
  - ▶ logische Operationen
  - ▶ „random-logic“ Schaltnetzen
- ▶ Verbindungsleitungen
  - ▶ Busse / Leitungsbündel
  - ▶ Multiplexer und Tri-state Treiber





- ▶ bisher:
  - ▶ Gatter und Schaltnetze
  - ▶ Flipflops als einzelne Speicherglieder
  - ▶ Schaltwerke zur Ablaufsteuerung
  
- ▶ weitere Komponenten: Register-Transfer- und Hauptblockebene
  - ▶ Speicher
  - ▶ Busse, Bustiming
  - ▶ Mikroprogrammierung zur Ablaufsteuerung



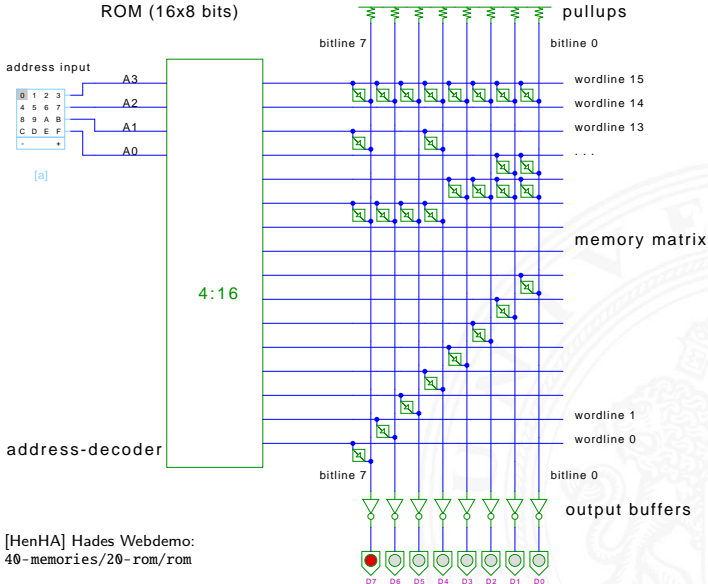


- ▶ System zur Speicherung von Information
- ▶ als Feld von  $N$  Adressen mit je  $m$ -bit Speicherworten
- ▶ typischerweise mit  $n$ -bit Adressen und  $N = 2^n$
- ▶ Kapazität also  $2^n \cdot m$  Bits
  
- ▶ Klassifikation
  - ▶ Speicherkapazität?
  - ▶ Schreibzugriffe möglich?
  - ▶ Schreibzugriffe auf einzelne Bits/Bytes oder nur Blöcke?
  - ▶ Information flüchtig oder dauerhaft gespeichert?
  - ▶ Zugriffszeiten beim Lesen und Schreiben
  - ▶ Technologie

# Speicherbausteine: Varianten

Typ	Kategorie	Löschen	byte-adressierbar	flüchtig	Typische Anwendung
SRAM	Lesen/Schreiben	elektrisch	ja	ja	Level-2 Cache
DRAM	Lesen/Schreiben	elektrisch	ja	ja	Hauptspeicher (alt)
SDRAM	Lesen/Schreiben	elektrisch	ja	ja	Hauptspeicher
ROM	nur Lesen	—	nein	nein	Geräte in großen Stückzahlen
PROM	nur Lesen	—	nein	nein	Geräte in kleinen Stückzahlen
EPROM	vorw. Lesen	UV-Licht	nein	nein	Prototypen
EEPROM	vorw. Lesen	elektrisch	ja	nein	Prototypen
Flash	Lesen/Schreiben	elektrisch	nein	nein	Speicherkarten, Mobile Geräte, SSDs

# ROM: Read-Only Memory



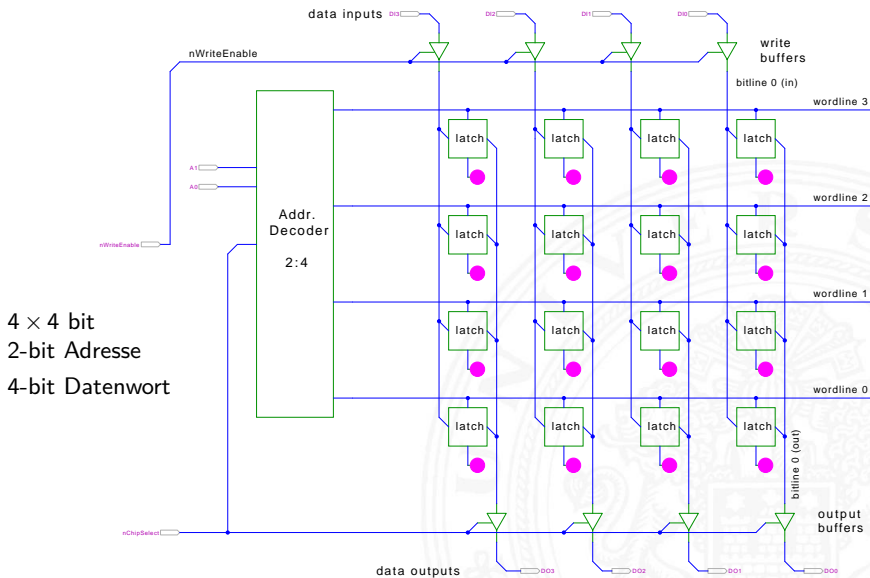
[HenHA] Hades Webdemo:  
40-memories/20-rom/rom

Speicher, der im Betrieb gelesen und geschrieben werden kann

- ▶ Arbeitsspeicher des Rechners
- ▶ für Programme und Daten
- ▶ keine Abnutzungseffekte
- ▶ benötigt Spannungsversorgung zum Speichern
  
- ▶ Aufbau als Matrixstruktur
- ▶  $n$  Adressbits, konzeptionell  $2^n$  Wortleitungen
- ▶  $m$  Bits pro Wort
- ▶ Realisierung der einzelnen Speicherstellen?
  - ▶ statisches RAM: 6-Transistor Zelle
  - ▶ dynamisches RAM: 1-Transistor Zelle

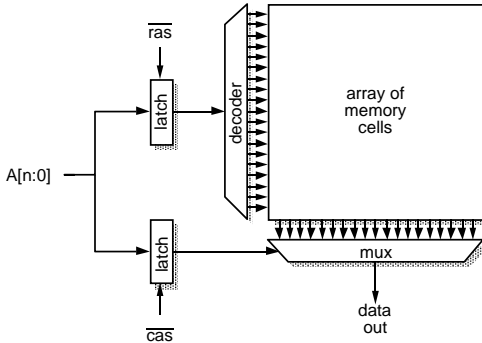
SRAM  
DRAM

# RAM: Blockschaltbild



4 × 4 bit  
2-bit Adresse  
4-bit Datenwort

# RAM: RAS/CAS-Adresdecodierung



Furber: *ARM SoC Architecture* [Fur00]

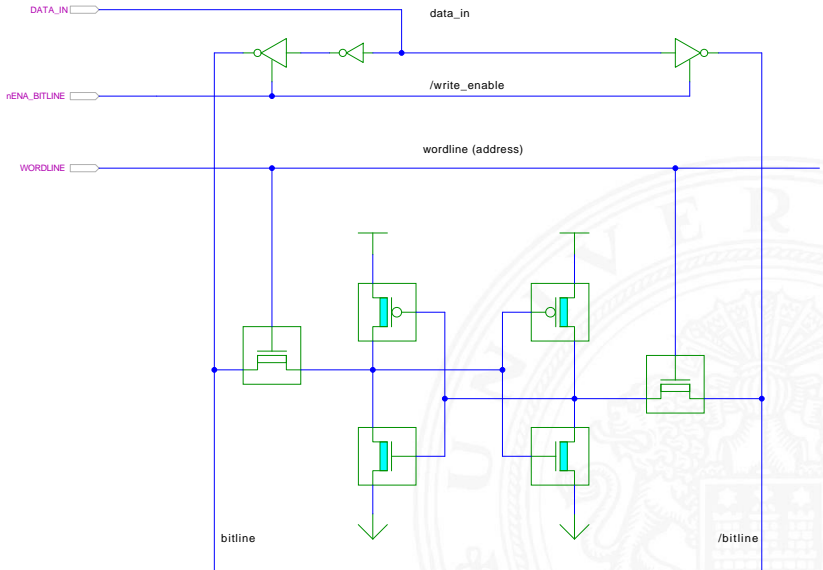
- ▶ Aufteilen der Adresse in zwei Hälften
- ▶  $\overline{ras}$  „row address strobe“ wählt „Wordline“
- ▶  $\overline{cas}$  „column address strobe“ – „Bitline“
- ▶ je ein  $2^{(n/2)}$ -bit Decoder/Mux statt ein  $2^n$ -bit Decoder

- ▶ Inhalt bleibt gespeichert solange Betriebsspannung anliegt
- ▶ *sechs-Transistor* Zelle zur Speicherung
  - ▶ weniger Platzverbrauch als Latches/Flipflops
  - ▶ kompakte Realisierung in CMOS-Technologie (s.u.)
  - ▶ zwei rückgekoppelte Inverter zur Speicherung
  - ▶ zwei n-Kanal Transistoren zur Anbindung an die Bitlines
- ▶ schneller Zugriff: Einsatz für Caches
- ▶ deutlich höherer Platzbedarf als DRAMs

# SRAM: Sechs-Transistor Speicherstelle („6T“)

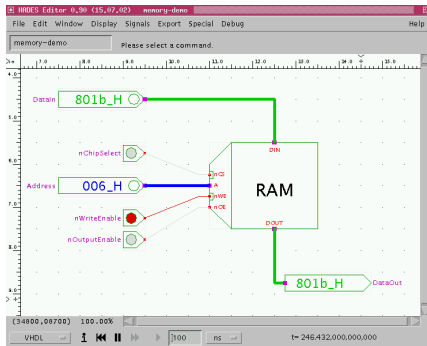
11.4.1 Rechnerarchitektur I - Hardwarestruktur - Speicherbausteine

64-040 Rechnerstrukturen und Betriebssysteme



[HenHA] Hades Demo: 05-switched/40-cmos/sramcell





- ▶ nur aktiv wenn  $nCS = 0$  (chip select)
- ▶ Schreiben wenn  $nWE = 0$  (write enable)
- ▶ Ausgabe wenn  $nOE = 0$  (output enable)

The screenshot shows a memory dump window titled 'Edit RAM\_1Kx16\_hades.models.rtl.lib.memory\_RAM0e'. The dump displays hexadecimal values for memory addresses from 000 to 138. A callout bubble points to the value '801b' at address 000. The text inside the bubble reads: 'Datenwort 0x801B in Adresse 0x006' and 'andere Speicherworte noch ungültig'. The memory dump shows 'XXXX' for most addresses, indicating uninitialized memory.

```
000 XXXX XXXX XXXX XXXX XXXX XXXX 801b XXXX
008 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
010 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
018 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
...
098 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
0A0 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
0A8 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
0B0 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
0B8 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
0C0 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
0C8 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
0D0 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
0D8 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
0E0 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
0E8 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
0F0 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
0F8 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
100 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
108 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
110 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
118 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
120 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
128 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
130 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
138 XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

[HenHA] Hades Demo: 50-rtlib/40-memory/ram

- ▶ integrierte Schaltung, 16 Ki bit Kapazität
- ▶ Organisation als 2 Ki Worte mit je 8-bit
  
- ▶ 11 Adresseingänge (A10...A0)
- ▶ 8 Anschlüsse für gemeinsamen Daten-Eingang/-Ausgang
- ▶ 3 Steuersignale
  - ▶  $\overline{CS}$  chip-select: Speicher nur aktiv wenn  $\overline{CS} = 0$
  - ▶  $\overline{WE}$  write-enable: Daten an gewählte Adresse schreiben
  - ▶  $\overline{OE}$  output-enable: Inhalt des Speichers ausgeben
  
- ▶ interaktive Hades-Demo zum Ausprobieren [HenHA]
  - ▶ Hades Demo: `40-memories/40-ram/demo-6116`
  - ▶ Hades Demo: `40-memories/40-ram/two-6116`

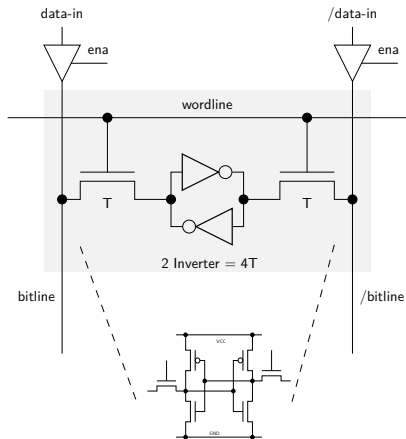


- ▶ Information wird in winzigen Kondensatoren gespeichert
- ▶ pro Bit je ein Transistor und Kondensator
  
- ▶ jeder Lesezugriff entlädt den Kondensator
- ▶ *Leseverstärker* zur Messung der Spannung auf der Bitline  
Schwellwertvergleich zur Entscheidung logisch 0/1
  
- Information muss anschließend neu geschrieben werden
- auch ohne Lese- oder Schreibzugriff ist regelmäßiger *Refresh* notwendig, wegen Selbstentladung (Millisekunden)
- 10 × langsamer als SRAM
- + DRAM für hohe Kapazität optimiert, minimaler Platzbedarf

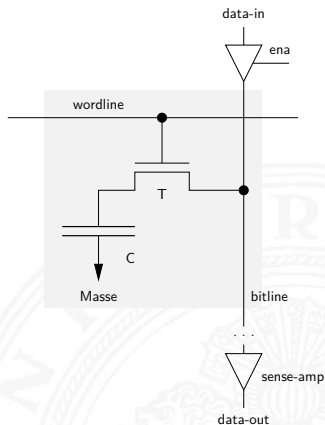
# SRAM vs. DRAM

11.4.1 Rechnerarchitektur I - Hardwarestruktur - Speicherbausteine

64-040 Rechnerstrukturen und Betriebssysteme



- ▶ 6 Transistoren/bit
- ▶ statisch (kein refresh)
- ▶ schnell
- ▶ 10...50 × DRAM Fläche



- ▶ 1 Transistor/bit
- ▶  $C = 5 \text{ fF} \approx 47\,000$  Elektronen
- ▶ langsam (sense-amp)
- ▶ minimale Fläche

# DRAM: Stacked- und Trench-Zelle

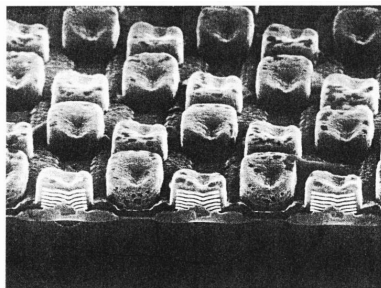
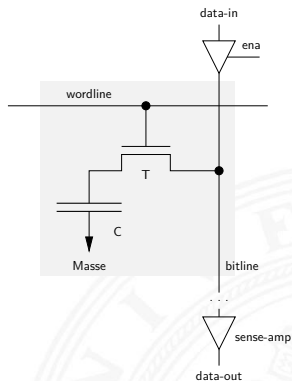


Abb. 7: Prototyp von Speicherzellen (Stapelkondensatoren) für zukünftige Speicherchips wie den Ein-Gigabit-Chip. Da für DRAM-Chips eine minimale Speicherkapazität von 25 fF notwendig ist, bringt es erhebliche Platzvorteile, die Kondensatorelemente vertikal übereinander zu stapeln. Die Dicke der Schichten beträgt etwa 50 nm. (Foto: Siemens)

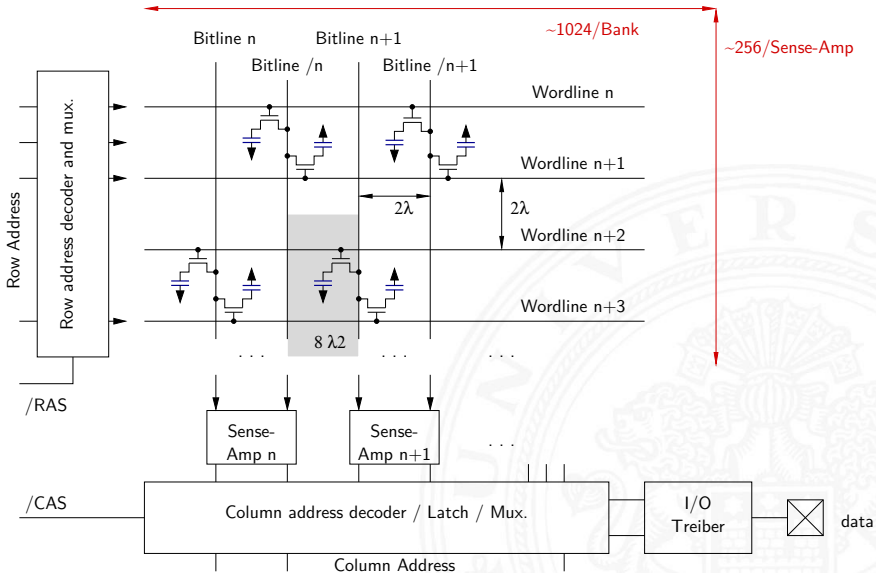


Siemens 1 Gbit DRAM

IBM CMOS-6X embedded DRAM

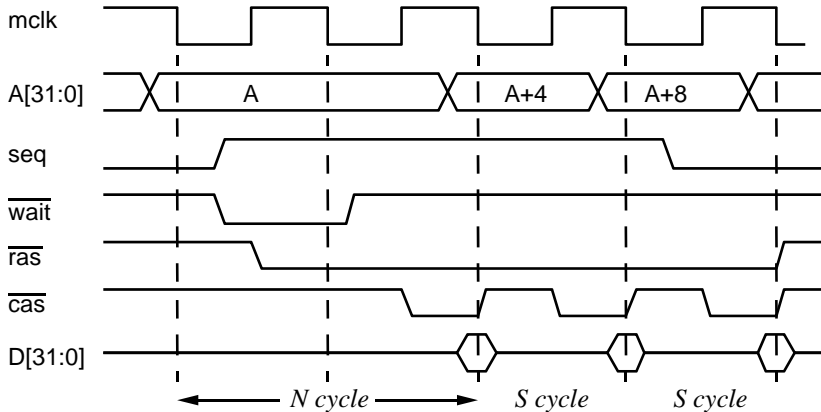
- ▶ zwei Bauformen: „stacked“ und „trench“
- ▶ Kondensatoren
  - ▶ möglichst kleine Fläche
  - ▶ Kapazität gerade ausreichend

# DRAM: Layout



- ▶ veraltete Varianten
  - ▶ FPM: *fast-page mode*
  - ▶ EDO: *extended data-out*
  - ▶ ...
  
- ▶ heute gebräuchlich
  - ▶ SDRAM: Ansteuerung synchron zu Taktsignal
  - ▶ DDR-SDRAM: *double-data rate* Ansteuerung wie SDRAM  
Daten werden mit steigender und fallender Taktflanke übertragen
  - ▶ DDR2...DDR5: Varianten mit höherer Taktrate  
aktuelle Übertragungsraten bis 51,2 GByte/sec pro Speicherkanal
  - ▶ GDDR3...GDDR6 (*Graphics Double Data Rate*)  
derzeit bis 128 GByte/sec
  - ▶ HBM...HBM3 (*High Bandwidth Memory*)  
derzeit bis 512 GByte/sec
  - ▶ HMC...HMC2 (*Hybrid Memory Cube*)  
derzeit bis 480 GByte/sec

# SDRAM: Lesezugriff auf sequenzielle Adressen

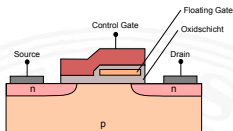


[Fur00]



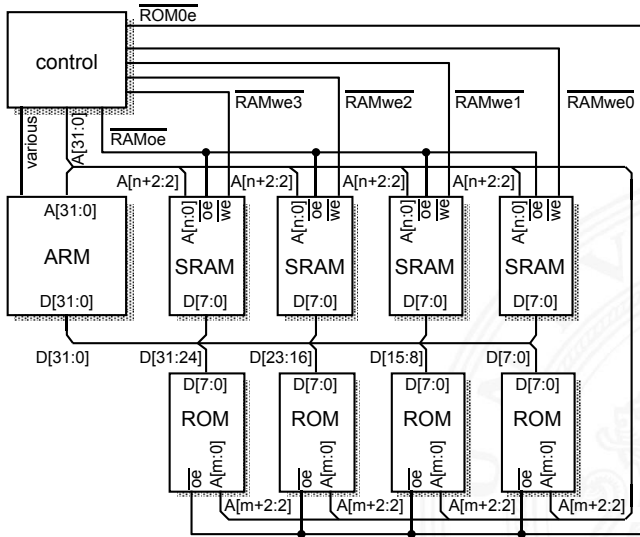
- ▶ ähnlich kompakt und kostengünstig wie DRAM
- ▶ nichtflüchtig (*non-volatile*): Information bleibt beim Ausschalten erhalten

- ▶ spezielle *floating-gate* Transistoren
  - ▶ das *floating-gate* ist komplett nach außen isoliert
  - ▶ einmal gespeicherte Elektronen sitzen dort fest



- ▶ Auslesen beliebig oft möglich, schnell
- ▶ Schreibzugriffe problematisch
  - ▶ intern hohe Spannung erforderlich (Gate-Isolierung überwinden)
  - ▶ Schreibzugriffe einer „0“ nur blockweise
  - ▶ pro Zelle nur einige 10 000 ... 100 M Schreibzugriffe möglich

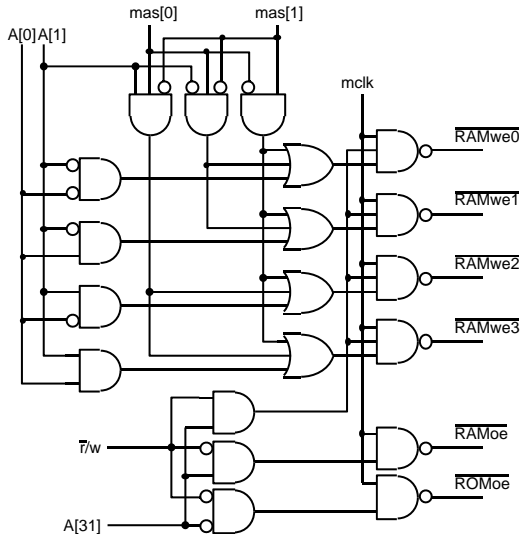
# Typisches Speichersystem



32-bit ARM Proz.  
4 × 8-bit SRAMs  
4 × 8-bit ROMs

[Fur00]

# Typisches Speichersystem: Adresdecodierung



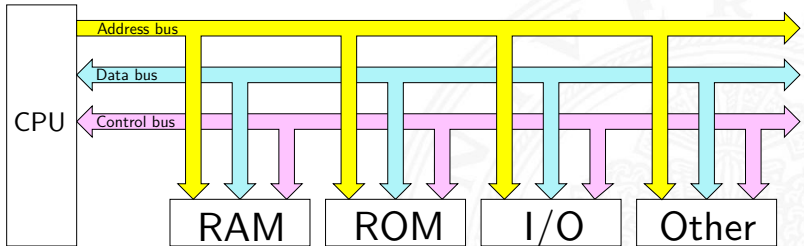
[Fur00]



- ▶ **Bus:** elektrische (und logische) Verbindung
  - ▶ mehrere Geräte
  - ▶ mehrere Blöcke innerhalb einer Schaltung
- ▶ Bündel aus Daten- und Steuersignalen
- ▶ mehrere Quellen (und mehrere Senken [lesende Zugriffe])
  - ▶ spezielle elektrische Realisierung:  
Tri-State-Treiber oder Open-Drain
- ▶ Bus-Arbitrierung: wer darf, wann, wie lange senden?
  - ▶ Master-Slave
  - ▶ gleichberechtigte Knoten, Arbitrierungsprotokolle
- ▶ synchron: mit globalem Taktsignal vom „Master“-Knoten  
asynchron: Wechsel von Steuersignalen löst Ereignisse aus

## ► typische Aufgaben

- Kernkomponenten (CPU, Speicher ...) miteinander verbinden
- Verbindungen zu den Peripherie-Bausteinen
- Verbindungen zu Systemmonitor-Komponenten
- Verbindungen zwischen I/O-Controllern und -Geräten
- ...

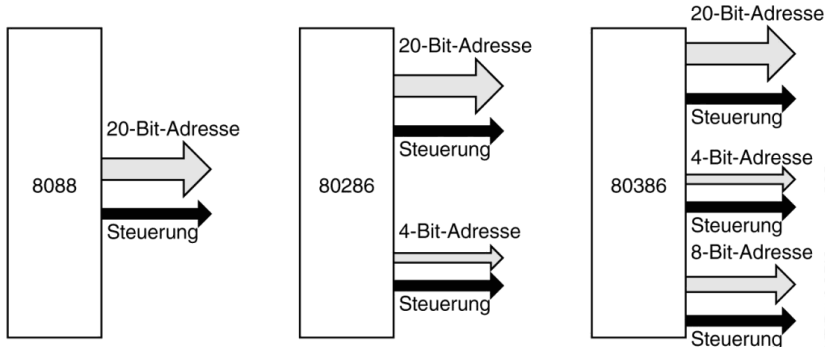


- ▶ viele unterschiedliche Typen, standardisiert mit sehr unterschiedlichen Anforderungen
  - ▶ High-Performance
  - ▶ einfaches Protokoll, billige Komponenten
  - ▶ Multi-Master-Fähigkeit, zentrale oder dezentrale Arbitrierung
  - ▶ Echtzeitfähigkeit, Daten-Streaming
  - ▶ wenig Leitungen bis zu Zweidraht-Bussen:  
I<sup>2</sup>C, SPI, System-Management-Bus ...
  - ▶ lange Leitungen: EIA-485, RS-232, Ethernet ...
  - ▷ Funkmedium: WLAN, Bluetooth (logische Verbindung)

typisches  $n$ -bit Mikroprozessor-System:

- ▶  $n$  Adress-Leitungen, also Adressraum  $2^n$  Bytes      Adressbus
- ▶  $n$  Daten-Leitungen      Datenbus
  
- ▶ Steuersignale      Control
  - ▶ clock: Taktsignal
  - ▶ read/write: Lese-/Schreibzugriff (aus Sicht des Prozessors)
  - ▶ wait: Wartezeit/-zyklen für langsame Geräte
  - ▶ ...
  
- ▶ um Leitungen zu sparen, teilweise gemeinsam genutzte Leitungen sowohl für Adressen als auch Daten.  
Zusätzliches Steuersignal zur Auswahl Adressen/Daten

# Adressbus: Evolution beim Intel x86

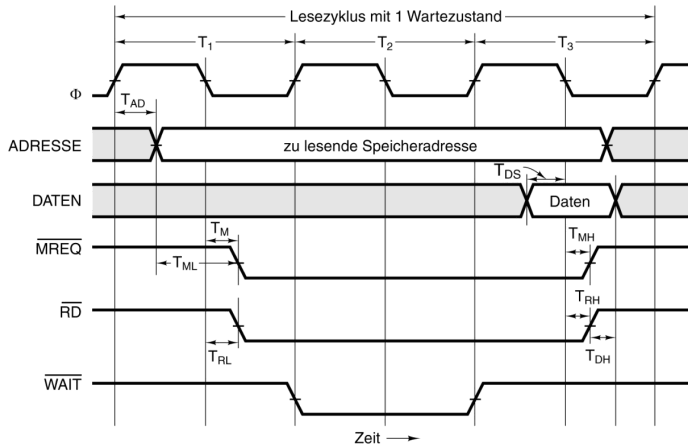


[TA14]

- ▶ 20-bit: 1 MiByte Adressraum
- 24-bit: 16 MiByte
- 32-bit: 4 GiByte
- ▶ alle Erweiterungen abwärtskompatibel
- ▶ 64-bit Architekturen: 48-, 56-, 64-bit Adressraum



# Synchroner Bus: Timing



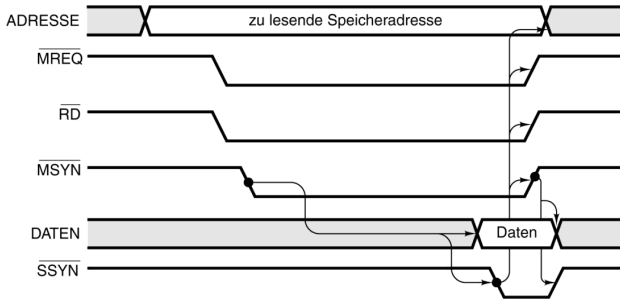
[TA14]

- ▶ alle Zeiten über Taktsignal  $\Phi$  gesteuert
- ▶  $\overline{MREQ}$ -Signal zur Auswahl Speicher oder I/O-Geräte
- ▶  $\overline{RD}$  signalisiert Lesezugriff
- ▶ Wartezyklen, solange der Speicher  $\overline{WAIT}$  aktiviert

► typische Parameter

Symbol	[ns]	Min	Max
$T_{AD}$ Adressausgabeverzögerung			4
$T_{ML}$ Adresse ist vor $\overline{MREQ}$ stabil		2	
$T_M$ $\overline{MREQ}$ -Verzögerung nach fallender Flanke von $\Phi$ in $T_1$			3
$T_{RL}$ $RD$ -Verzögerung nach fallender Flanke von $\Phi$ in $T_1$			3
$T_{DS}$ Setup-Zeit vor fallender Flanke von $\Phi$		2	
$T_{MH}$ $\overline{MREQ}$ -Verzögerung nach fallender Flanke von $\Phi$ in $T_3$			3
$T_{RH}$ $\overline{RD}$ -Verzögerung nach fallender Flanke von $\Phi$ in $T_3$			3
$T_{DH}$ Hold-Zeit nach der Deaktivierung von $\overline{RD}$		0	

# Asynchroner Bus: Lesezugriff



[TA14]

- ▶ Steuersignale  $\overline{MSYN}$ : Master fertig  
 $\overline{SSYN}$ : Slave fertig
- ▶ flexibler für Geräte mit stark unterschiedlichen Zugriffszeiten

- ▶ mehrere Komponenten wollen Übertragung initiieren  
immer nur ein Transfer zur Zeit möglich
- ▶ der Zugriff muss serialisiert werden

## 1. zentrale Arbitrierung

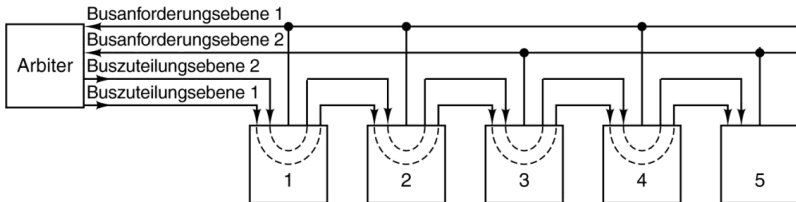
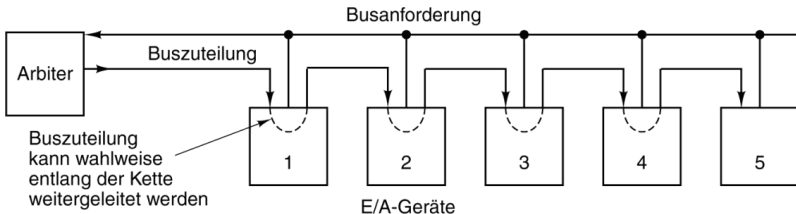
- ▶ Arbitrer gewährt Bus-Requests
- ▶ Strategien
  - ▶ Prioritäten für verschiedene Geräte
  - ▶ „round-robin“ Verfahren
  - ▶ „Token“-basierte Verfahren
  - ▶ usw.

## 2. dezentrale Arbitrierung

- ▶ protokollbasiert
- ▶ Beispiel
  - ▶ Komponenten sehen ob Bus frei ist
  - ▶ beginnen zu senden
  - ▶ Kollisionserkennung: gesendete Daten lesen
  - ▶ ggf. Übertragung abbrechen
  - ▶ „später“ erneut versuchen

# Bus Arbitrierung (cont.)

- ▶ I/O-Geräte oft höher priorisiert als die CPU
  - ▶ I/O-Zugriffe müssen schnell/sofort behandelt werden
  - ▶ Benutzerprogramm kann warten



[TA14]

- ▶ Menge an (Nutz-) Daten, die pro Zeiteinheit übertragen werden kann
- ▶ zusätzlicher Protokolloverhead  $\Rightarrow$  Brutto- / Netto-Datenrate

▶ RS-232	50 bit/sec	...	460 Kbit/sec
I <sup>2</sup> C	100 Kbit/sec (Std.)	...	3,4 Mbit/sec (High Speed)
USB	1,5 Mbit/sec (1.x)	...	40 Gbit/sec (4.0)
ISA	128 Mbit/sec	...	
PCI	1 Gbit/sec (2.0)	...	4,3 Gbit/sec (3.0)
AGP	2,1 Gbit/sec (1x)	...	34,1 Gbit/sec (8x 64-bit)
PCI Express	2,5 Gbit/sec (1.x)	...	1024 Gbit/sec (6.0) x16
HyperTransport	25,6 Gbit/sec (1.0)	...	409,6 Gbit/sec (3.1)
NVLink	640,0 Gbit/sec (1.0)	...	1,2 Tbit/sec (2.0)
Infinity Fabric		...	4,1 Tbit/sec

- ▶ [en.wikipedia.org/wiki/List\\_of\\_interface\\_bit\\_rates](https://en.wikipedia.org/wiki/List_of_interface_bit_rates)

## Peripheral Component Interconnect (Intel 1991)

- ▶ 33 MHz Takt optional 66 MHz Takt
- ▶ 32-bit Bus-System optional auch 64-bit
- ▶ gemeinsame Adress-/Datenleitungen
- ▶ Arbitrierung durch Bus-Master CPU
  
- ▶ Abwärtskompatibilität
  - ▶ PCI-Bus als logische Verbindung (SW-Layer) zu Komponenten
  - ▶ technisch: PCIe
  
- ▶ Auto-Konfiguration
  - ▶ angeschlossene Geräte werden automatisch erkannt
  - ▶ eindeutige Hersteller- und Geräte-Nummern
  - ▶ Betriebssystem kann zugehörigen Treiber laden
  - ▶ automatische Zuweisung von Adressbereichen und IRQs

```
[maeder@tams165]~>lspci -v
00:00.0 Host bridge: Intel Corporation Sky Lake Host Bridge/DRAM Registers (rev 08)
  Subsystem: Dell Device 06dc
  Flags: bus master, fast devsel, latency 0
  Capabilities: <access denied>

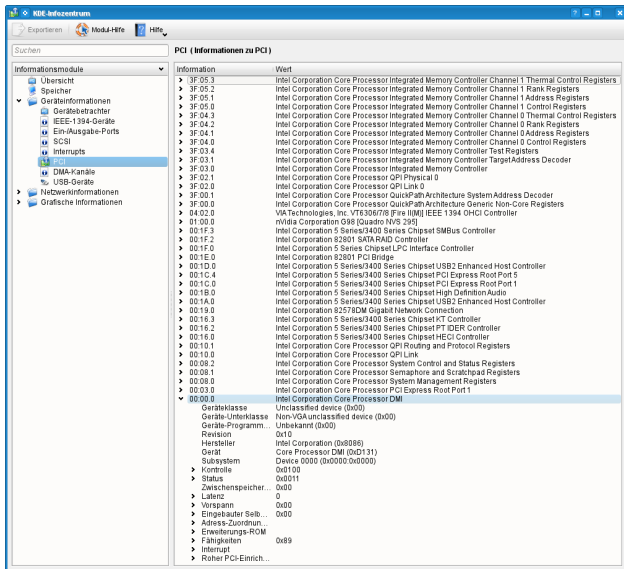
00:02.0 VGA compatible controller: Intel Corporation Sky Lake Integrated Graphics (rev 07)
  Subsystem: Dell Device 06dc
  Flags: bus master, fast devsel, latency 0, IRQ 134
  Memory at e0000000 (64-bit, non-prefetchable) [size=16M]
  Memory at d0000000 (64-bit, prefetchable) [size=256M]
  I/O ports at f000 [size=64]
  Expansion ROM at <unassigned> [disabled]
  Capabilities: <access denied>
  Kernel driver in use: i915_bpo

00:04.0 Signal processing controller: Intel Corporation Device 1903 (rev 08)
  Subsystem: Dell Device 06dc
  Flags: bus master, fast devsel, latency 0, IRQ 16
  Memory at e1340000 (64-bit, non-prefetchable) [size=32K]
  Capabilities: <access denied>
  Kernel driver in use: proc_thermal

00:14.0 USB controller: Intel Corporation Device 9d2f (rev 21) (prog-if 30 [XHCI])
  Subsystem: Dell Device 06dc
  Flags: bus master, medium devsel, latency 0, IRQ 125
  Memory at e1330000 (64-bit, non-prefetchable) [size=64K]
  ...
```



# PCI-Bus: Peripheriegeräte (cont.)



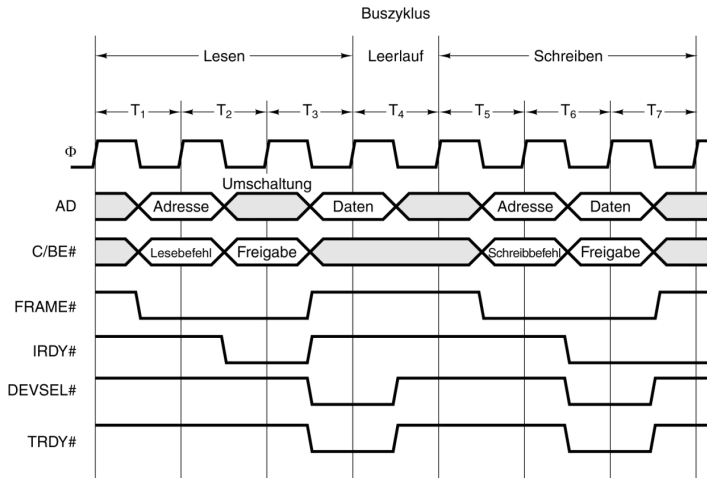
The screenshot shows the KDE InfoCenter window with the 'PCI (Informationen zu PCI)' section selected. The left sidebar shows a tree view of system information, with 'PCI' highlighted under 'Geräteinformationen'. The main pane displays a table of PCI device information.

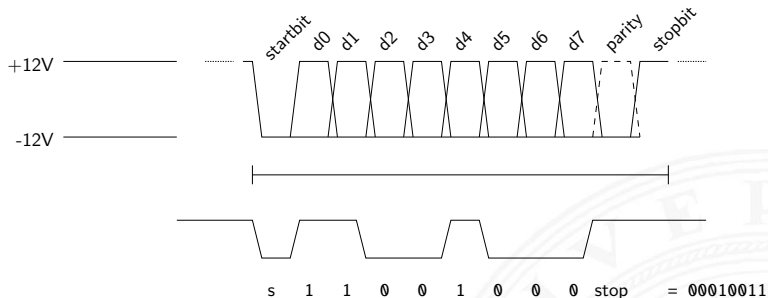
Information	Wert
3F.05.3	Intel Corporation Core Processor Integrated Memory Controller Channel 1 Thermal Control Registers
3F.05.2	Intel Corporation Core Processor Integrated Memory Controller Channel 1 Rank Registers
3F.05.1	Intel Corporation Core Processor Integrated Memory Controller Channel 1 Address Registers
3F.05.0	Intel Corporation Core Processor Integrated Memory Controller Channel 1 Control Registers
3F.04.3	Intel Corporation Core Processor Integrated Memory Controller Channel 0 Thermal Control Registers
3F.04.2	Intel Corporation Core Processor Integrated Memory Controller Channel 0 Rank Registers
3F.04.1	Intel Corporation Core Processor Integrated Memory Controller Channel 0 Address Registers
3F.04.0	Intel Corporation Core Processor Integrated Memory Controller Channel 0 Control Registers
3F.03.4	Intel Corporation Core Processor Integrated Memory Controller Test Registers
3F.03.1	Intel Corporation Core Processor Integrated Memory Controller TargetAddress Decoder
3F.03.0	Intel Corporation Core Processor Integrated Memory Controller
3F.02.1	Intel Corporation Core Processor QPI Physical 0
3F.02.0	Intel Corporation Core Processor QPI Link 0
3F.00.1	Intel Corporation Core Processor QuickPath Architecture System Address Decoder
3F.00.0	Intel Corporation Core Processor QuickPath Architecture Generic Non-Core Registers
04.02.0	VIA Technologies, Inc. VT6306/7/8 [Fire II(M)] IEEE 1394 OHCI Controller
01.00.0	nVidia Corporation G98 [Quadro NV5 295]
00.1F.3	Intel Corporation 5 Series/3400 Series Chipset SMBus Controller
00.1F.2	Intel Corporation 82801 SATA RAID Controller
00.1F.0	Intel Corporation 5 Series Chipset LPC Interface Controller
00.1E.0	Intel Corporation 82801 PCI Bridge
00.1D.0	Intel Corporation 5 Series/3400 Series Chipset USB2 Enhanced Host Controller
00.1C.4	Intel Corporation 5 Series/3400 Series Chipset PCI Express Root Port 5
00.1C.0	Intel Corporation 5 Series/3400 Series Chipset PCI Express Root Port 1
00.1B.0	Intel Corporation 5 Series/3400 Series Chipset High Definition Audio
00.1A.0	Intel Corporation 5 Series/3400 Series Chipset USB2 Enhanced Host Controller
00.19.0	Intel Corporation 82576DM Gigabit Network Connection
00.16.3	Intel Corporation 5 Series/3400 Series Chipset KT Controller
00.16.2	Intel Corporation 5 Series/3400 Series Chipset PT IDER Controller
00.16.0	Intel Corporation 5 Series/3400 Series Chipset HECI Controller
00.10.1	Intel Corporation Core Processor QPI Routing and Protocol Registers
00.10.0	Intel Corporation Core Processor QPI Link
00.08.2	Intel Corporation Core Processor System Control and Status Registers
00.08.1	Intel Corporation Core Processor Semaphore and Scratchpad Registers
00.08.0	Intel Corporation Core Processor System Management Registers
00.03.0	Intel Corporation Core Processor PCI Express Root Port 1
00.00.0	Intel Corporation Core Processor DMI
Geräteklasse	Unclassified device (0x00)
Geräte-Unterklasse	Non-VGA unclassified device (0x00)
Geräte-Programm...	Unbekannt (0x00)
Revision	0x10
Hersteller	Intel Corporation (0x8086)
Gerät	Core Processor DMI (0xD131)
Subsystem	Device 0000 (0x0000 0x0000)
Kontrolle	0x0100
Status	0x0011
Zwischenspeicher...	0x00
Latenz	0
Vorspann	0x00
Eingehalter Selb...	0x00
Adress-Zuordnun...	
Erweiterungs-ROM	
Fähigkeiten	0xB9
Interrupt	
Roher PCI-Einrich...	

# PCI-Bus: Leitungen („mindestens“)

Signal	Leitungen	Master	Slave	Beschreibung
CLK	1			Takt (33 oder 66 MHz)
AD	32	×	×	Gemultiplexte Adress- und Datenleitungen
PAR	1	×		Adress- oder Datenparitätsbit
C/BE	4	×		Busbefehl/Bitmap für Byte Enable (zeigt gültige Datenbytes an)
FRAME#	1	×		Kennzeichnet, dass AD und C/BE aktiviert sind
IRDY#	1	×		Lesen: Master wird akzeptieren Schreiben: Daten liegen an
IDSEL	1	×		Wählt Konfigurationsraum statt Speicher
DEVSEL#	1		×	Slave hat seine Adresse decodiert und ist in Bereitschaft
TRDY#	1		×	Lesen: Daten liegen an Schreiben: Slave wird akzeptieren
STOP#	1		×	Slave möchte Transaktion sofort abbrechen
PERR#	1			Empfänger hat Datenparitätsfehler erkannt
SERR#	1			Adressparitätsfehler oder Systemfehler erkannt
REQ#	1			Bus-Arbitration: Anforderung des Busses
GNT#	1			—" Zuteilung des Busses
RST#	1			Setzt das System und alle Geräte zurück

# PCI-Bus: Transaktionen

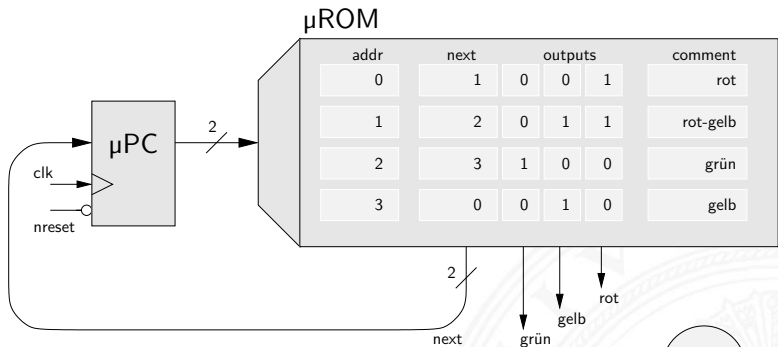




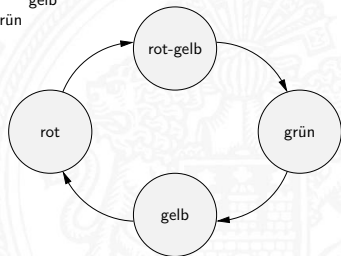
- ▶ Baudrate 300, 600, ..., 19 200, 38 400, 115 200 bits/sec
- ▶ Anzahl Datenbits 5, 6, 7, 8
- ▶ Anzahl Stopbits 1, 2
- ▶ Parität none, odd, even
- ▶ minimal drei Leitungen: GND, TX, RX (Masse, Transmit, Receive)
- ▶ oft weitere Leitungen für erweitertes Handshake

- ▶ als Alternative zu direkt entworfenen Schaltwerken
- ▶ *Mikroprogrammzähler  $\mu PC$* : Register für aktuellen Zustand
- ▶  $\mu PC$  adressiert den Mikroprogrammspeicher  $\mu ROM$
- ▶  $\mu ROM$  konzeptionell in mehrere Felder eingeteilt
  - ▶ die verschiedenen Steuerleitungen
  - ▶ ein oder mehrere Felder für Folgezustand
  - ▶ ggf. zusätzliche Logik und Multiplexer zur Auswahl unter mehreren Folgezuständen
  - ▶ ggf. Verschachtelung und Aufruf von Unterprogrammen:  
„nanoProgramm“
- ▶ siehe „Praktikum Rechnerstrukturen und Betriebssysteme“

# Mikroprogramm: Beispiel Ampel



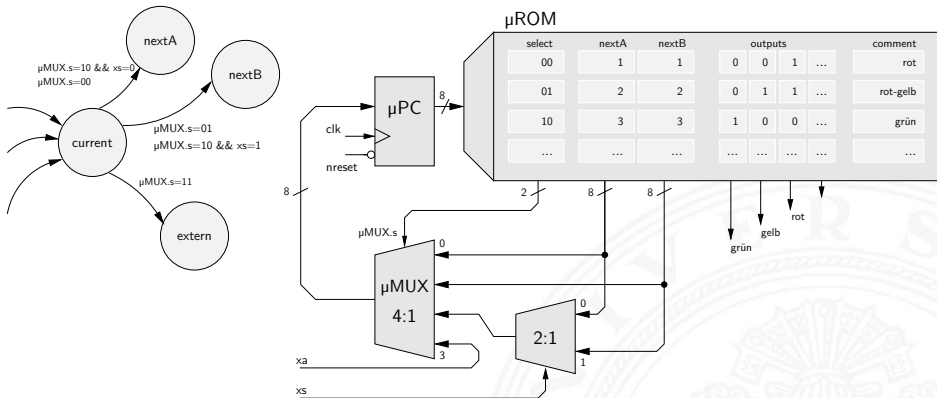
- ▶  $\mu PC$  adressiert das  $\mu ROM$
- ▶ *next*-Ausgang liefert Folgezustand
- ▶ andere Ausgänge steuern die Schaltung = die Lampen der Ampel



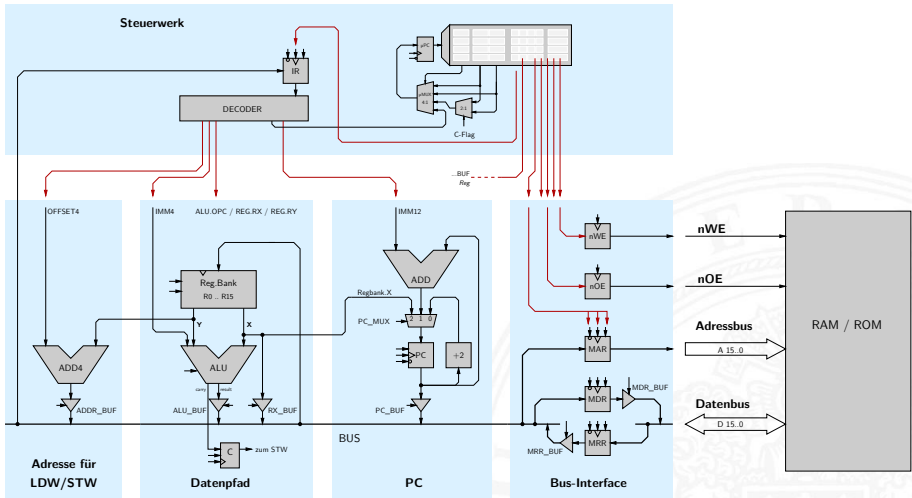
# Mikroprogramm: Beispiel zur Auswahl des Folgezustands

11.4.3 Rechnerarchitektur I - Hardwarestruktur - Mikroprogrammierung

64-040 Rechnerstrukturen und Betriebssysteme



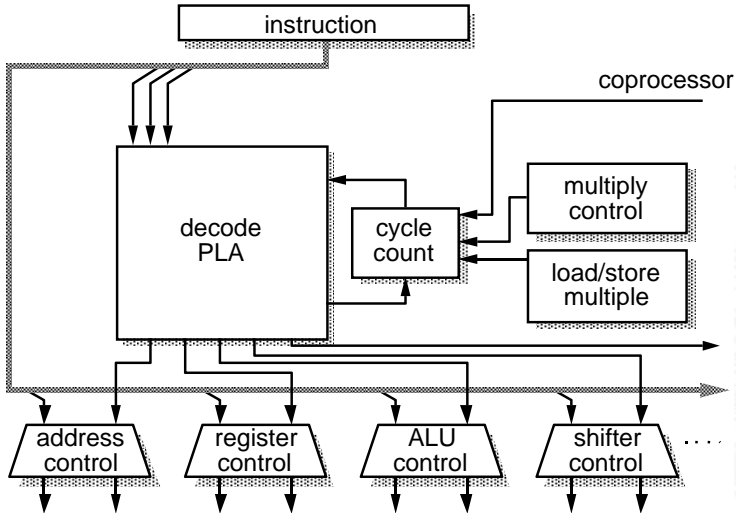
- ▶ Multiplexer erlaubt Auswahl des  $\mu PC$  Werts
- ▶  $nextA$ ,  $nextB$  aus dem  $\mu ROM$ , externer  $xa$  Wert
- ▶  $xs$  Eingang für bedingte Sprünge



- ▶ aktuelle Architekturen: weniger Mikroprogrammierung
- ⇒ Pipelining (folgt in 14 Rechnerarchitektur II – Pipelining)

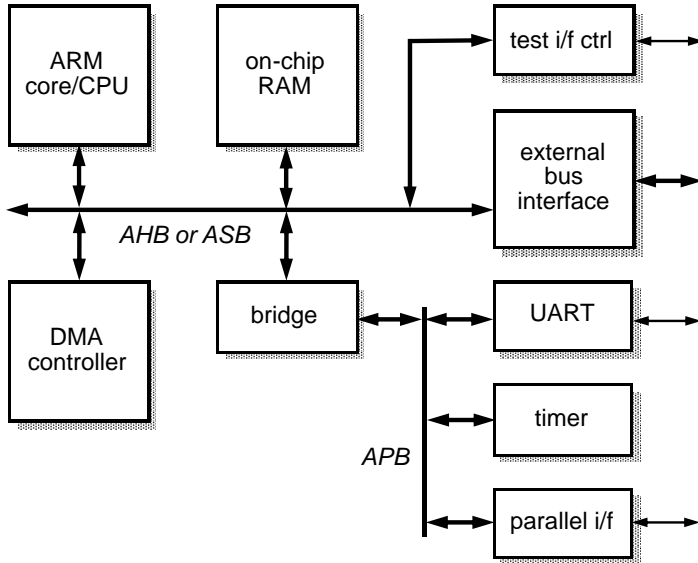


# Mikroprogramm: Befehlsdecoder des ARM 7 Prozessors

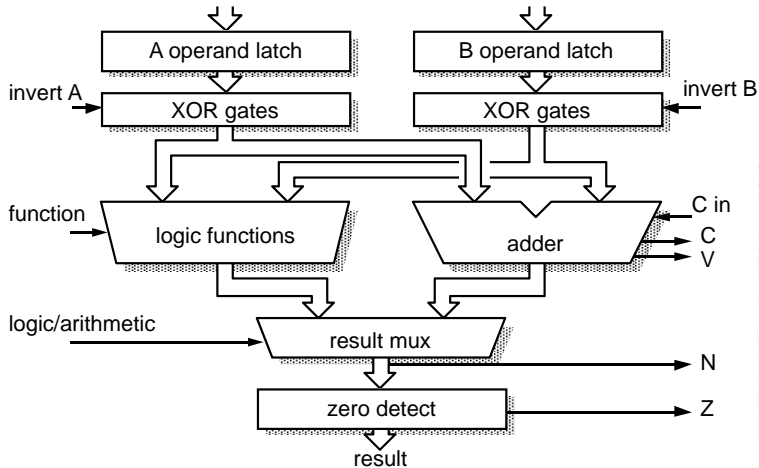


[Fur00]

# typisches ARM SoC System



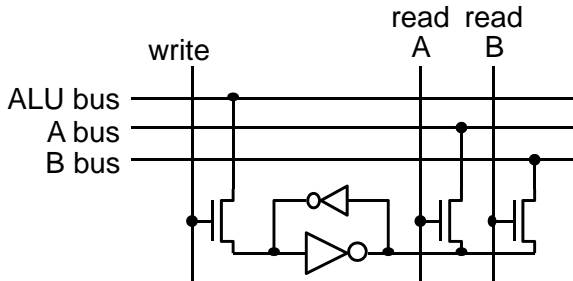
# RT-Ebene: ALU des ARM 6 Prozessors



[Fur00]

- ▶ Register für die Operanden A und B
- ▶ Addierer und separater Block für logische Operationen

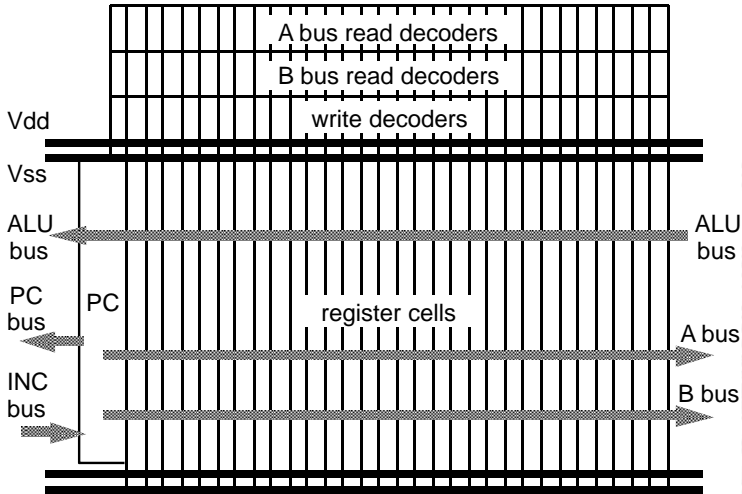
# Multi-Port-Registerbank: Zelle



[Fur00]

- ▶ Prinzip wie 6T-SRAM: rückgekoppelte Inverter
- ▶ mehrere (hier zwei) parallele Lese-Ports
- ▶ mehrere Schreib-Ports möglich, aber kompliziert

# Multi-Port Registerbank: Floorplan/Chiplayout

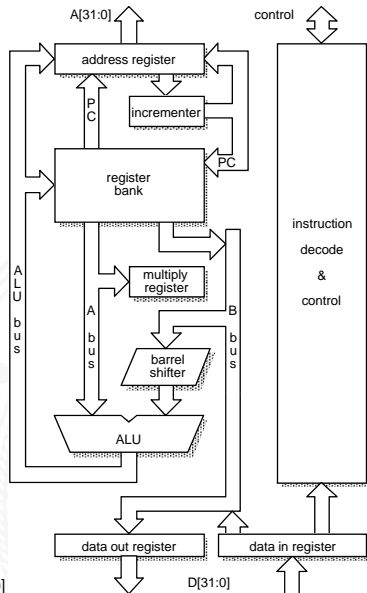


# Kompletter Prozessor: ARM 3

11.4.4 Rechnerarchitektur I - Hardwarestruktur - Beispielsystem: ARM

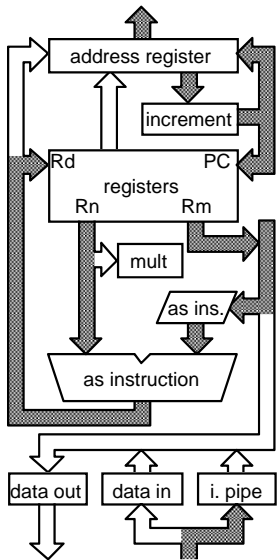
64-040 Rechnerstrukturen und Betriebssysteme

- ▶ Registerbank (inkl. Program Counter)
- ▶ Inkrementer
- ▶ Adress-Register
  
- ▶ ALU, Multiplizierer, Shifter
  
- ▶ Speicherinterface (Data-In / -Out)
  
- ▶ Steuerwerk
- ▶ bis ARM 7: 3-stufige Pipeline  
*fetch, decode, execute*

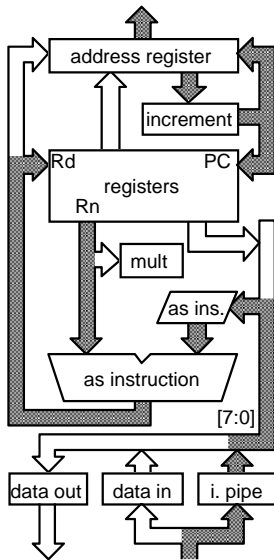


[Fur00]

# ARM Datentransfer: Register-Operationen



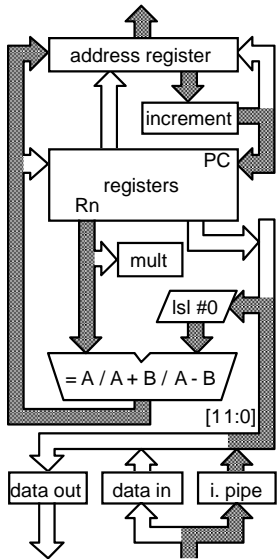
(a) register – register operations



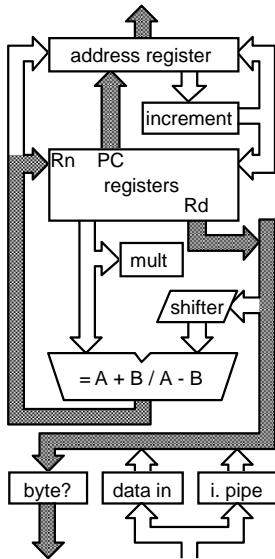
(b) register – immediate operations

[Fur00]

# ARM Datentransfer: Store-Befehl



(a) 1st cycle - compute address



(b) 2nd cycle - store & auto-index

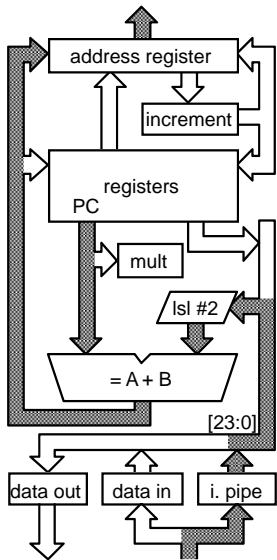
[Fur00]



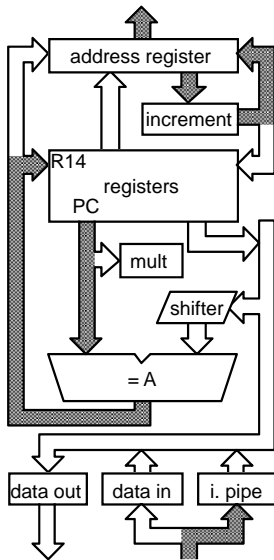
# ARM Datentransfer: Funktionsaufruf/Sprungbefehl

11.4.4 Rechnerarchitektur I - Hardwarestruktur - Beispielsystem: ARM

64-040 Rechnerstrukturen und Betriebssysteme



(a) 1st cycle - compute branch target

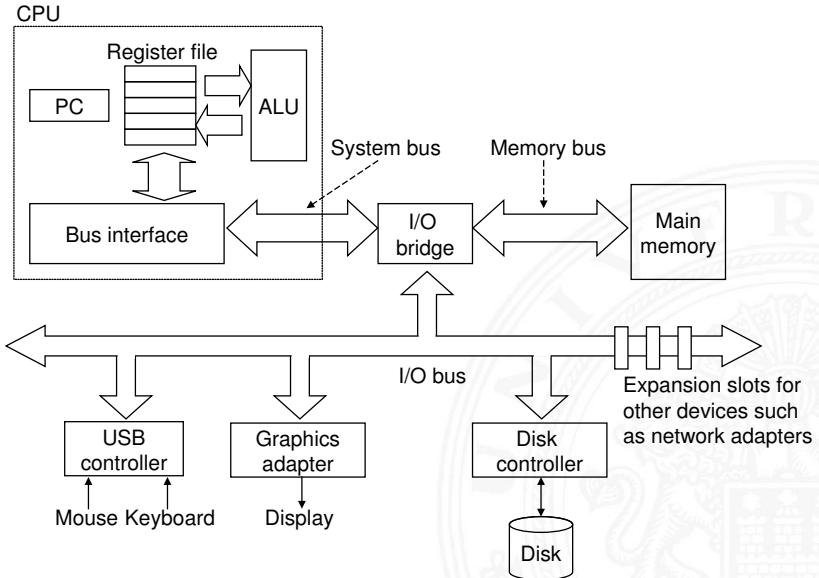


(b) 2nd cycle - save return address

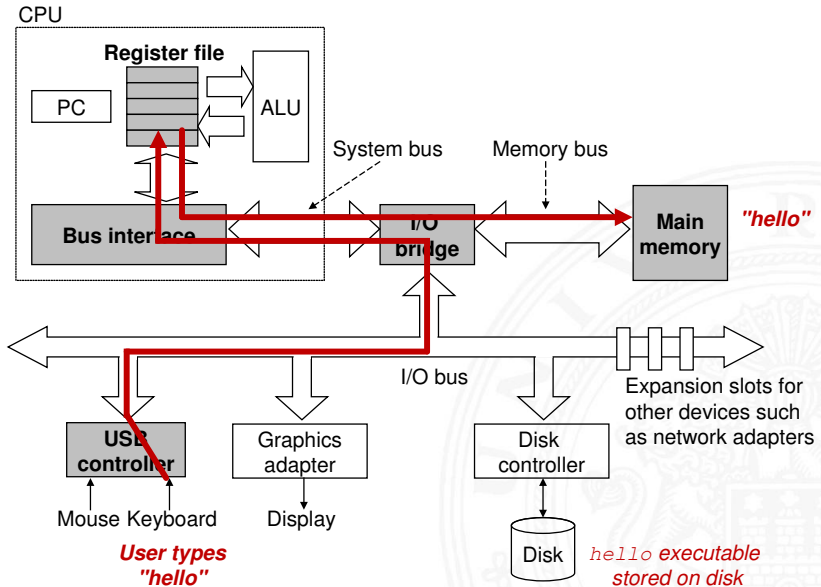
[Fur00]

- ▶ „Choreografie“ der Funktionseinheiten?
- ▶ Wie kommuniziert man mit Rechnern?
- ▶ Was passiert beim Einschalten des Rechners?
  
- ▶ Erweiterungen des von-Neumann Konzepts
  - ▶ parallele, statt sequenzieller Befehlsabarbeitung  
⇒ *Pipelining*
  - ▶ mehrere Ausführungseinheiten  
⇒ *superskalare Prozessoren, Mehrkern-Architekturen*
  - ▶ dynamisch veränderte Abarbeitungsreihenfolge  
⇒ „*out-of-order execution*“
  - ▶ getrennte Daten- und Instruktionsspeicher  
⇒ *Harvard-Architektur*
  - ▶ *Speicherhierarchie*, Caches etc.
  
- siehe Kapitel 14 *Rechnerarchitektur II*

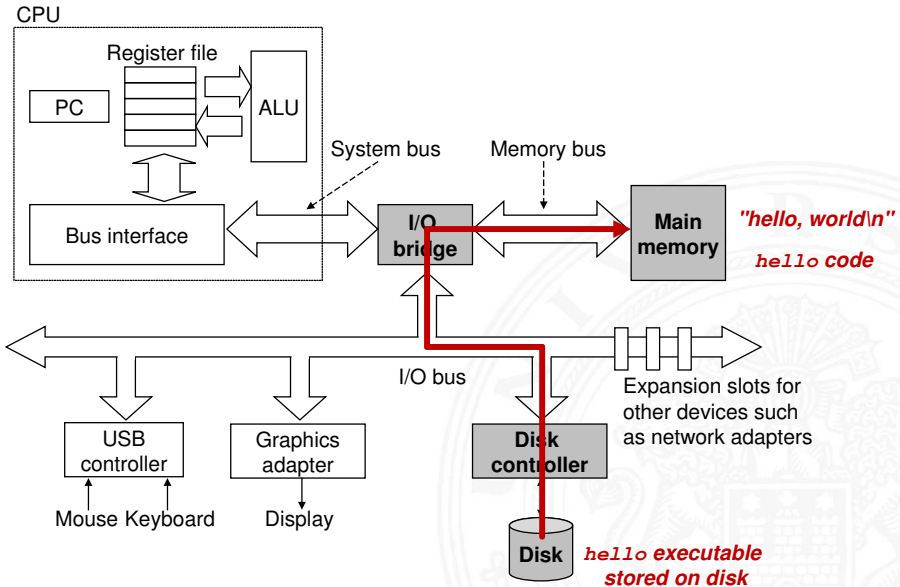
# Hardwareorganisation eines typischen Systems



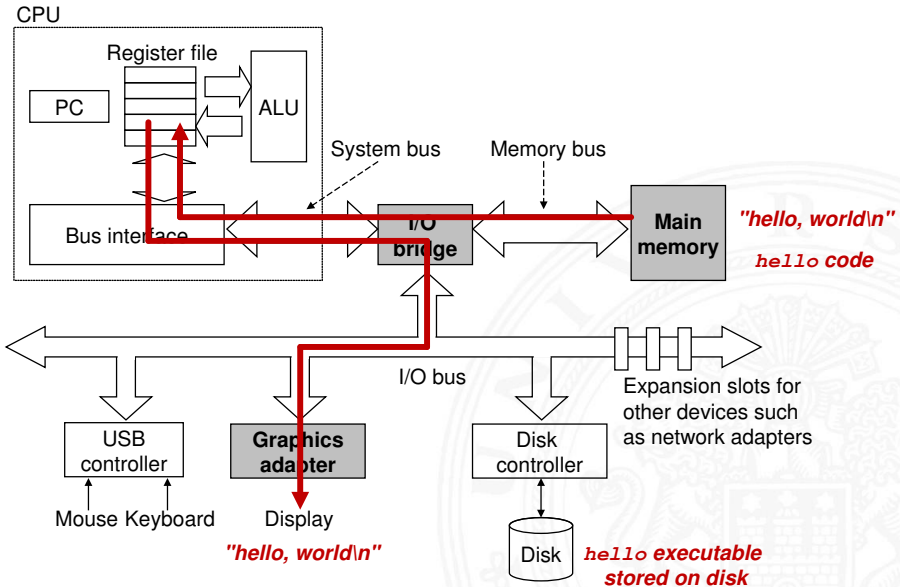
# Programmausführung: 1. Benutzereingabe



# Programmausführung: 2. Programm laden



# Programmausführung: 3. Programmlauf





# Boot-Prozess

## Was passiert beim Einschalten des Rechners?

- ▶ Chipsatz erzeugt Reset-Signale für alle ICs
- ▶ Reset für die zentralen Prozessor-Register (PC ...)
- ▶ PC wird auf Startwert initialisiert (z.B. 0xFFFF FFEF)
- ▶ Befehlszyklus wird gestartet
  
- ▶ Prozessor greift auf die Startadresse zu  
dort liegt ein ROM mit dem Boot-Programm
- ▶ Initialisierung und Selbsttest des Prozessors
- ▶ Löschen und Initialisieren der Caches
- ▶ Konfiguration des Chipsatzes
- ▶ Erkennung und Initialisierung von I/O-Komponenten
  
- ▶ Laden des Betriebssystems

- [BO15] R.E. Bryant, D.R. O'Hallaron:  
*Computer systems – A programmers perspective.*  
3rd global ed., Pearson Education Ltd., 2015.  
ISBN 978-1-292-10176-7. [csapp.cs.cmu.edu](http://csapp.cs.cmu.edu)
- [TA14] A.S. Tanenbaum, T. Austin: *Rechnerarchitektur – Von der digitalen Logik zum Parallelrechner.*  
6. Auflage, Pearson Deutschland GmbH, 2014.  
ISBN 978-3-8689-4238-5
- [Fur00] S. Furber: *ARM System-on-Chip Architecture.*  
2nd edition, Pearson Education Limited, 2000.  
ISBN 978-0-201-67519-1
- [GK83] D.D. Gajski, R.H. Kuhn: *Guest Editors' Introduction: New VLSI Tools.* in: *IEEE Computer* 16 (1983), December, Nr. 12, S. 11-14. ISSN 0018-9162



- [PH20] D.A. Patterson, J.L. Hennessy: *Computer Organization and Design – The Hardware Software Interface – MIPS Edition*. 6th edition, Morgan Kaufmann Publishers Inc., 2020. ISBN 978-0-12-820109-1
- [PH16] D.A. Patterson, J.L. Hennessy: *Rechnerorganisation und Rechnerentwurf – Die Hardware/Software-Schnittstelle*. 5. Auflage, Oldenbourg, 2016. ISBN 978-3-11-044605-0
- [Mäd11] A. Mäder: *Vorlesung: Rechnerarchitektur und Mikrosystemtechnik*. Universität Hamburg, FB Informatik, 2011, Vorlesungsfolien. [tams.informatik.uni-hamburg.de/lectures/2011ws/vorlesung/ram](http://tams.informatik.uni-hamburg.de/lectures/2011ws/vorlesung/ram)
- [HenHA] N. Hendrich: *HADES — HAMBURG DEsign System*. Universität Hamburg, FB Informatik, Lehrmaterial. [tams.informatik.uni-hamburg.de/applets/hades/webdemos](http://tams.informatik.uni-hamburg.de/applets/hades/webdemos)