64-040 Modul InfB-RS: Rechnerstrukturen

https://tams.informatik.uni-hamburg.de/lectures/2015ws/vorlesung/rs

– Kapitel 11 –

Norman Hendrich

Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

Wintersemester 2015/2016

Kapitel 11

Register-Transfer-Ebene

Register-Transfer-Ebene

Modellierung eines digitalen Systems als Schaltung aus:

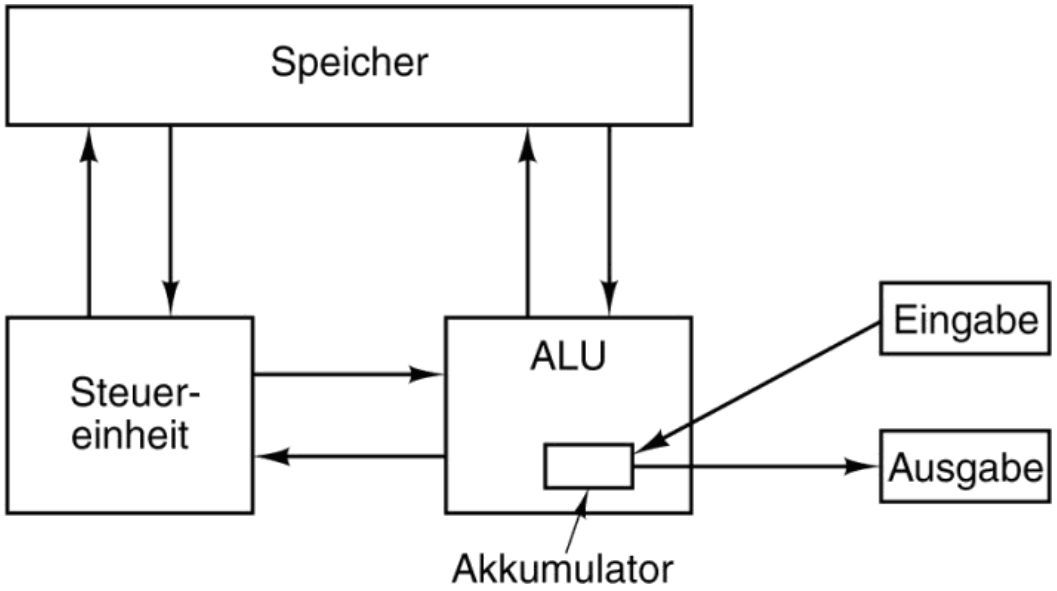
- speichernden Komponenten
  - Registern — Flipflops, Register, Registerbank
  - Speichern — SRAM, DRAM, ROM, PLA
- funktionalen Schaltnetzen
  - Addierer, arithmetische Schaltungen
  - logische Operationen
  - „random-logic“ Schaltnetzen
- Verbindungsleitungen
  - Busse / Leitungsbündel
  - Tri-state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

Rechnerarchitektur: zwei Aspekte

1. Hardwarearchitektur: der strukturelle Aufbau des Rechnersystems
   - Art und Anzahl der Hardware-Betriebsmittel
   - sowie die Verbindungs- / Kommunikationseinrichtungen

2. Operationsprinzip: das funktionelle Verhalten der Architektur
   - Programmierschnittstelle
   - Befehlssatz: Instruction Set Architecture (ISA)
   - Maschinenorganisation: Wie werden Befehle abgearbeitet?
   - folgt im Kapitel „12 Instruction Set Architecture“

Wiederholung: von-Neumann Rechner



[TA14]

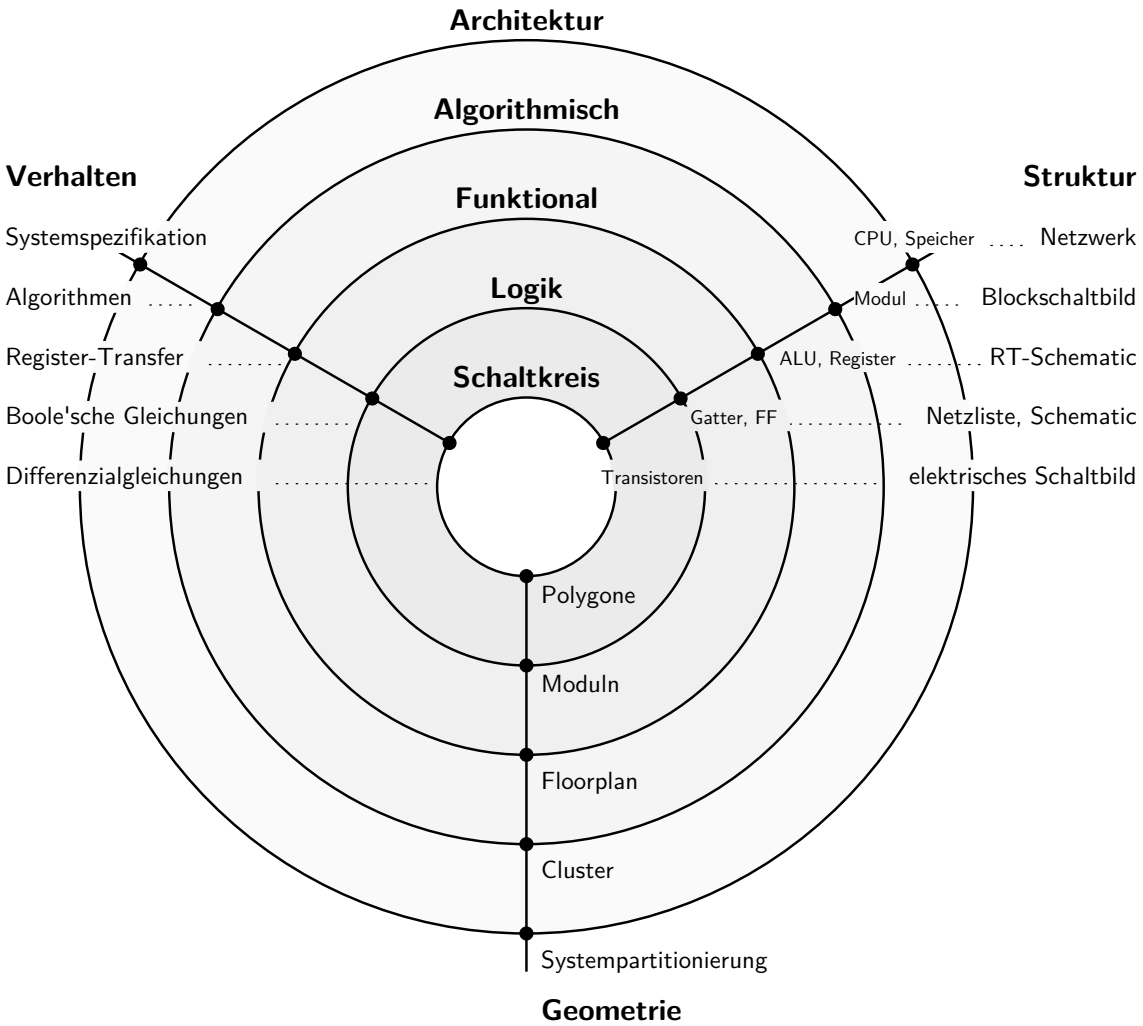
Fünf zentrale Komponenten:

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

Komponenten des von-Neumann Rechners

- Prozessor = Steuerwerk + Operationswerk — CPU
- 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) — typ. 8..64 Register
  - evtl. Register mit Spezialaufgaben
- Speicher — *memory*
  - Hauptspeicher/RAM: — random-access memory
  - Hauptspeicher/ROM: — read-only memory
  - Externspeicher — Festplatten, CD/DVD
- Peripheriegeräte (Eingabe/Ausgabe) — I/O

Exkurs: Y-Diagramm



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

Abstraktionsebenen im Y-Diagramm

drei unterschiedliche Aspekte/Dimensionen:

1. Verhalten — wie?
2. Struktur (logisch) — was?
3. Geometrie (physikalisch) — wo?

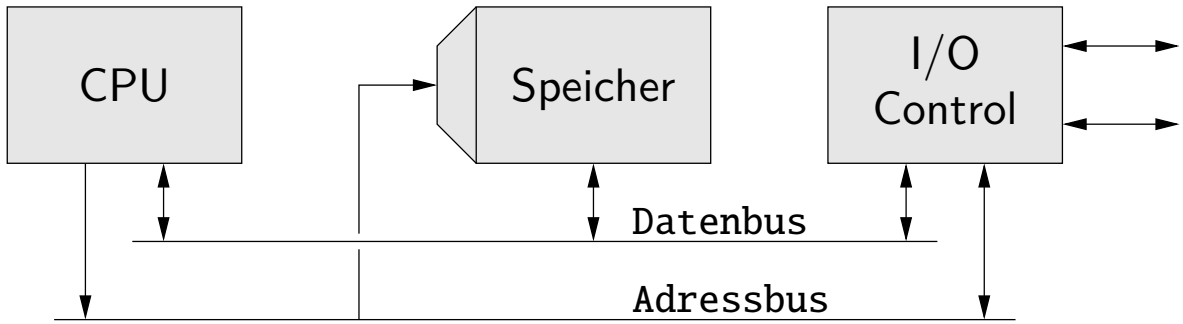
- Start möglichst abstrakt, z.B. als Verhaltensbeschreibung
- Ende des Entwurfsprozesses ist vollständige IC Geometrie für die Halbleiterfertigung (Planarprozess)

- Entwurfsprogramme („EDA“, *Electronic Design Automation*) unterstützen den Entwerfer: setzen Verhalten in Struktur und Struktur in Geometrien um

Abstraktion im Hardware/VLSI-Entwurf

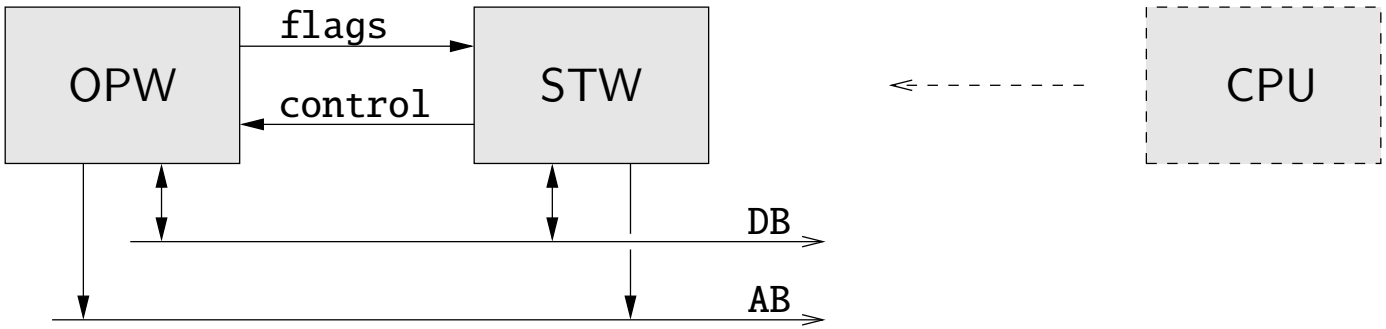
Abstraktionsebenen

- Architekturebene
  - Funktion/Verhalten — Leistungsanforderungen
  - Struktur — Netzwerk
    aus — Prozessoren, Speicher, Busse, Controller...
  - Nachrichten — Programme, Prokolle
  - Geometrie — Systempartitionierung



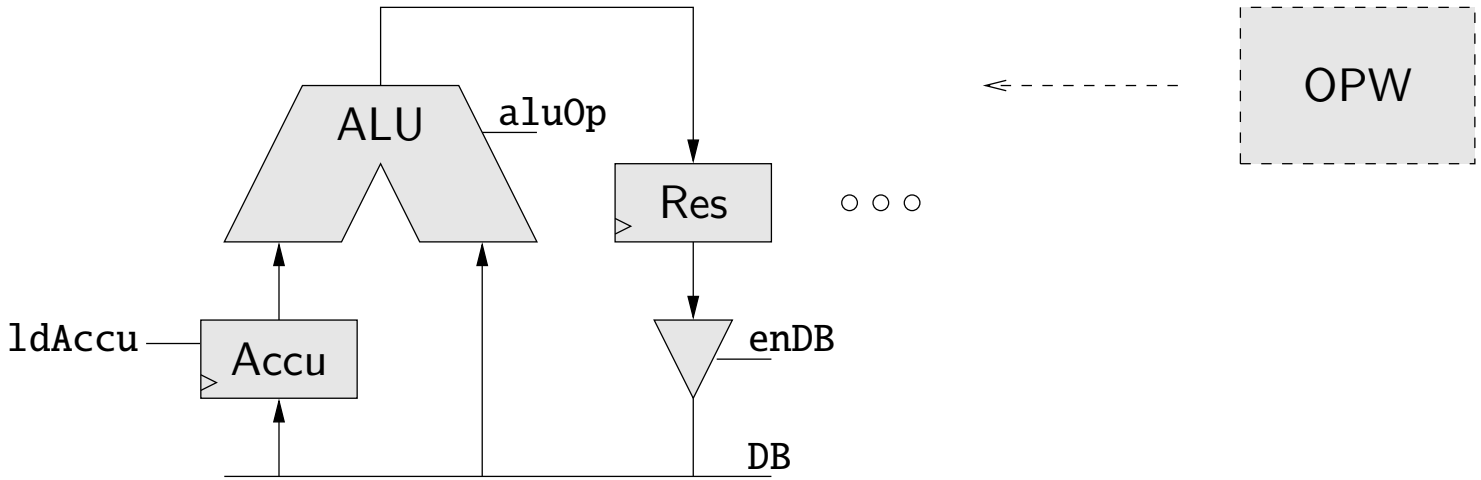
Abstraktion im Hardware/VLSI-Entwurf (cont.)

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



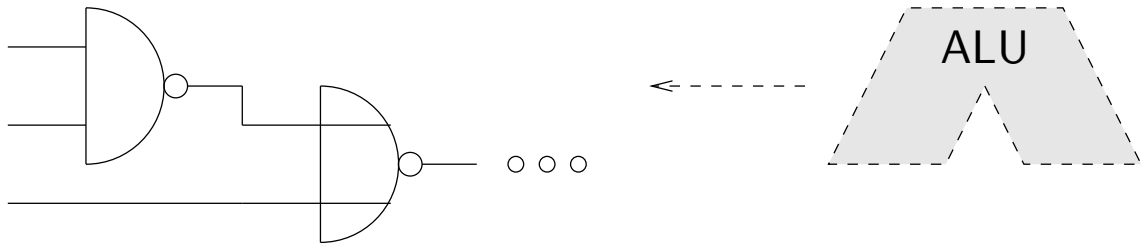
Abstraktion im Hardware/VLSI-Entwurf (cont.)

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



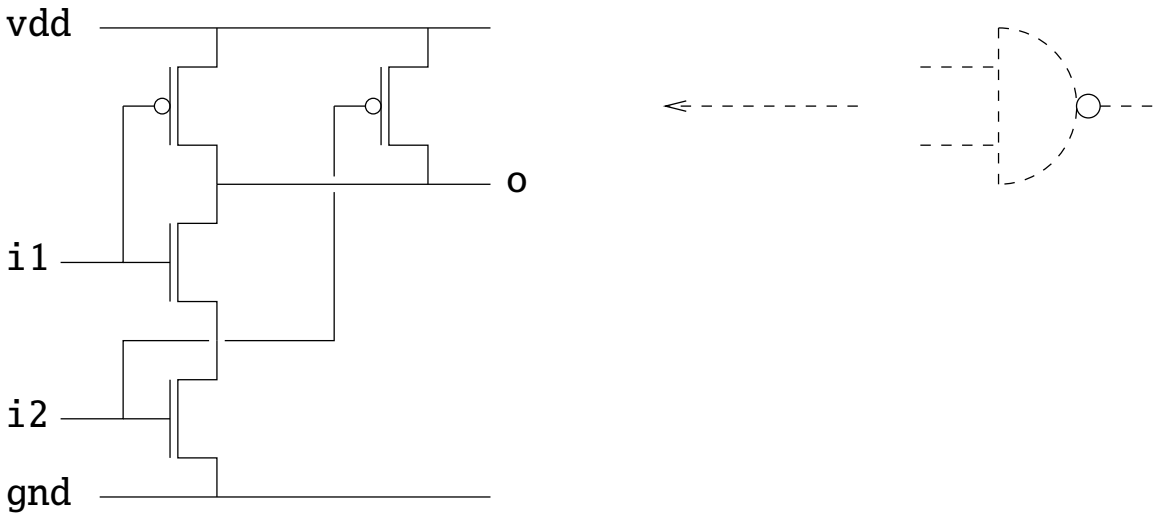
Abstraktion im Hardware/VLSI-Entwurf (cont.)

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



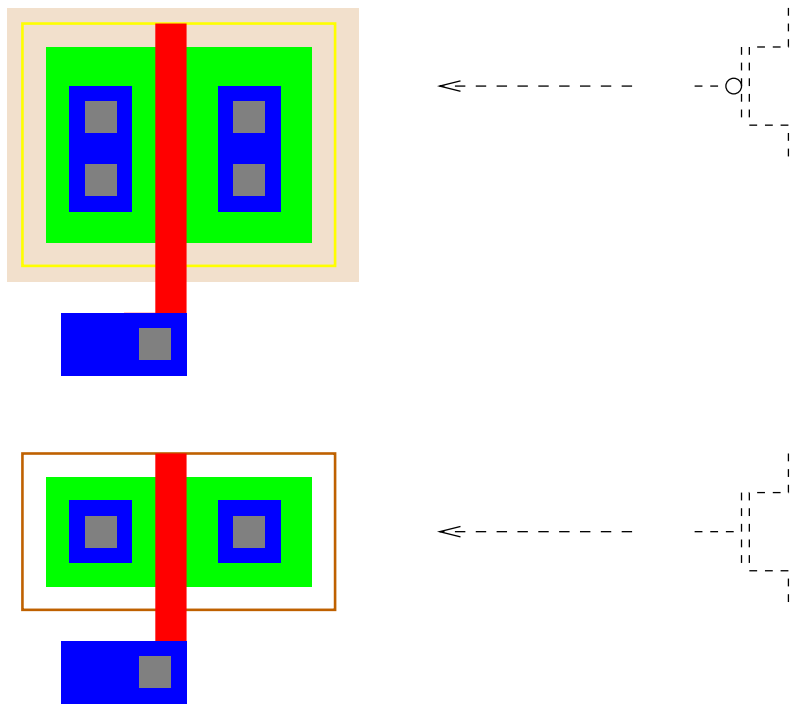
Abstraktion im Hardware/VLSI-Entwurf (cont.)

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



Abstraktion im Hardware/VLSI-Entwurf (cont.)

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



Hauptblockebene/RT-Ebene

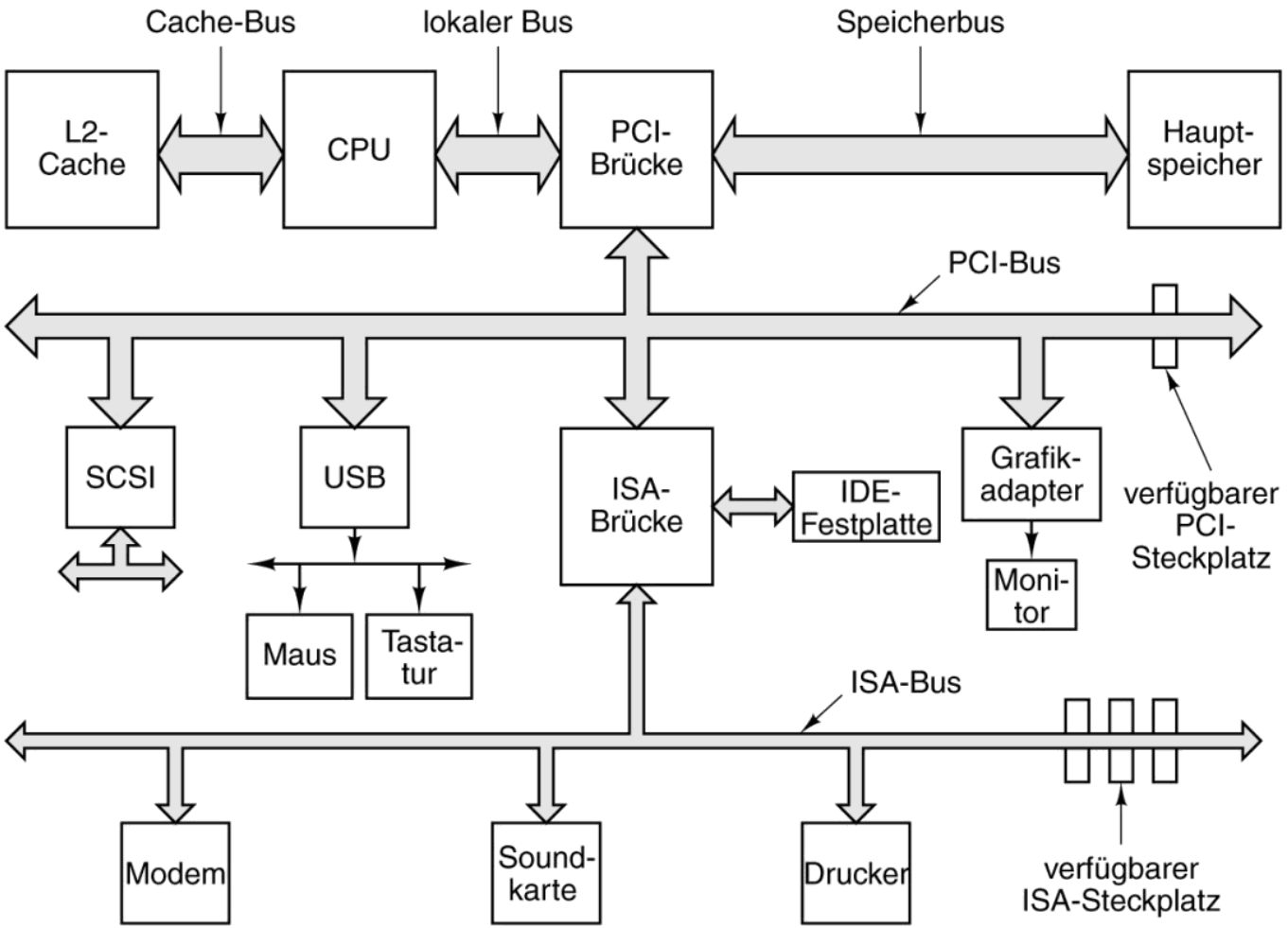
Modellierung eines digitalen Systems als Schaltung aus

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

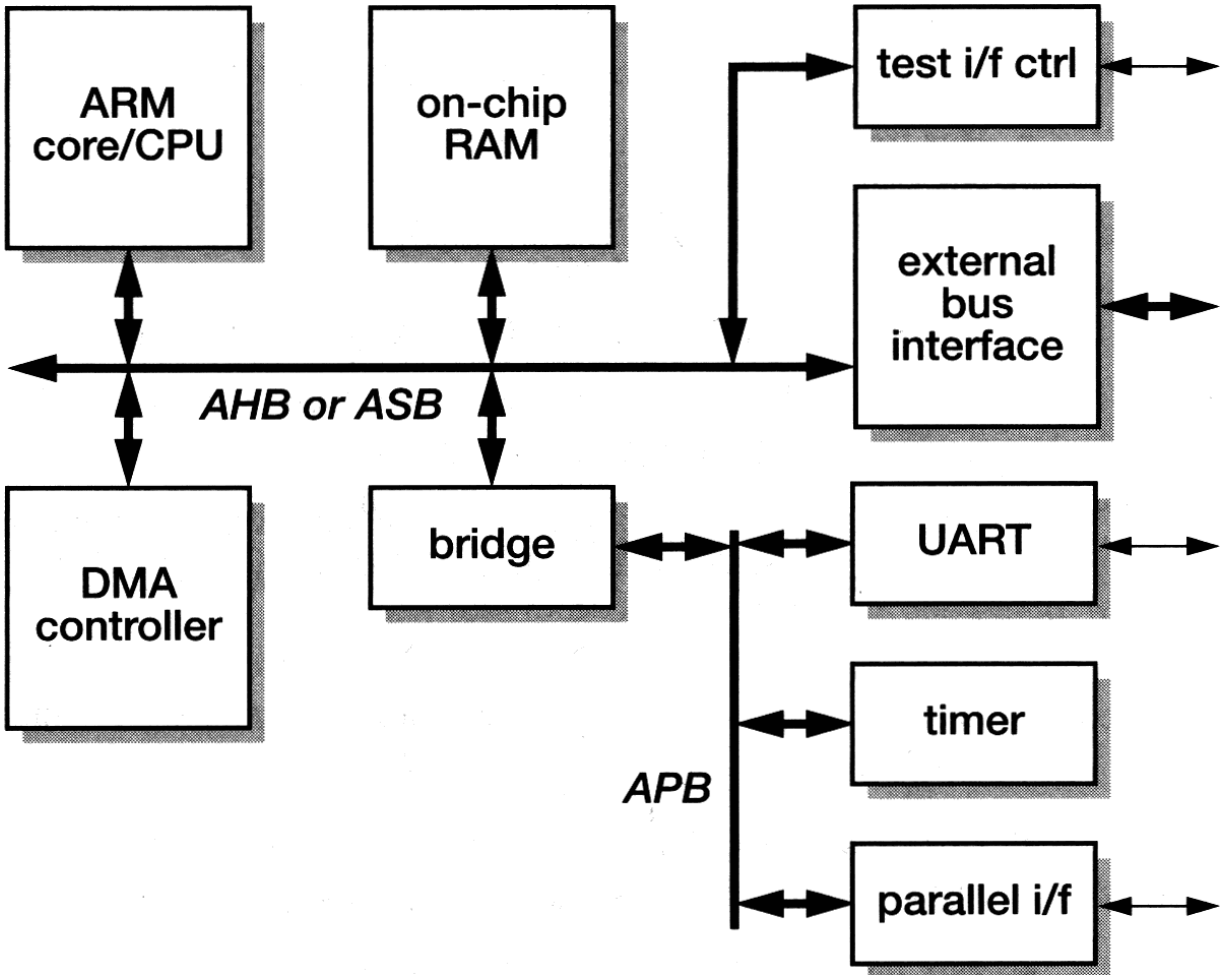
Rechnerarchitektur: Hardwarestruktur

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

PC auf Hauptblockebene

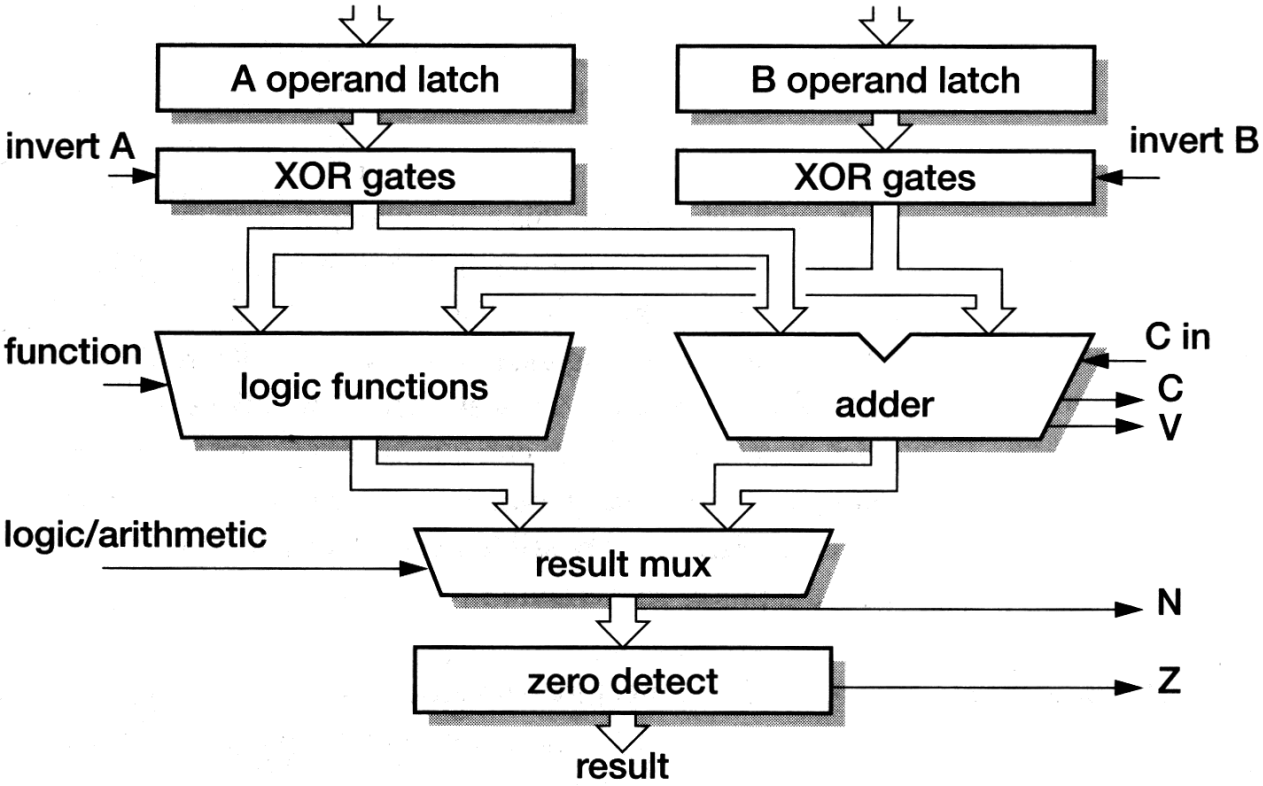


Typisches ARM SoC System



S. Furber: *ARM System-on-Chip Architecture* [Fur00]

RT-Ebene: ALU des ARM 6 Prozessors



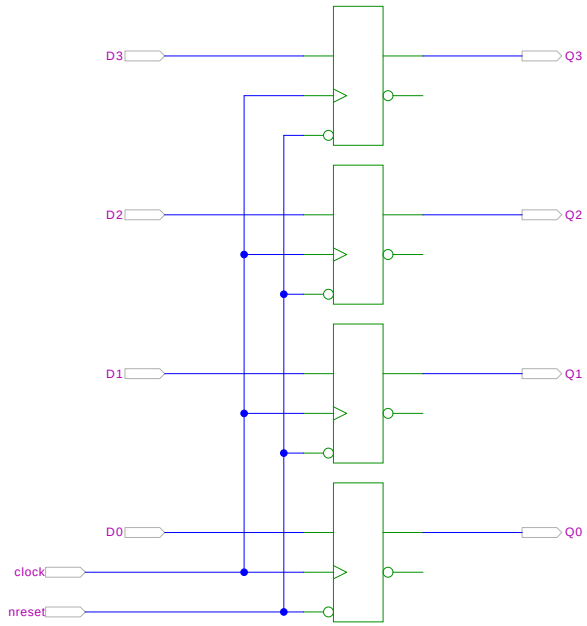
[Fur00]

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

Register

speichert einen n-bit Wert:

- n einzelne Flipflops
- gemeinsamer Takt für alle Flipflops
- ggf. gemeinsame Set/Reset-Leitungen
- alle Varianten, pegel- oder flankengesteuert
- Blockschaltbild wie Flipflops



Speicher

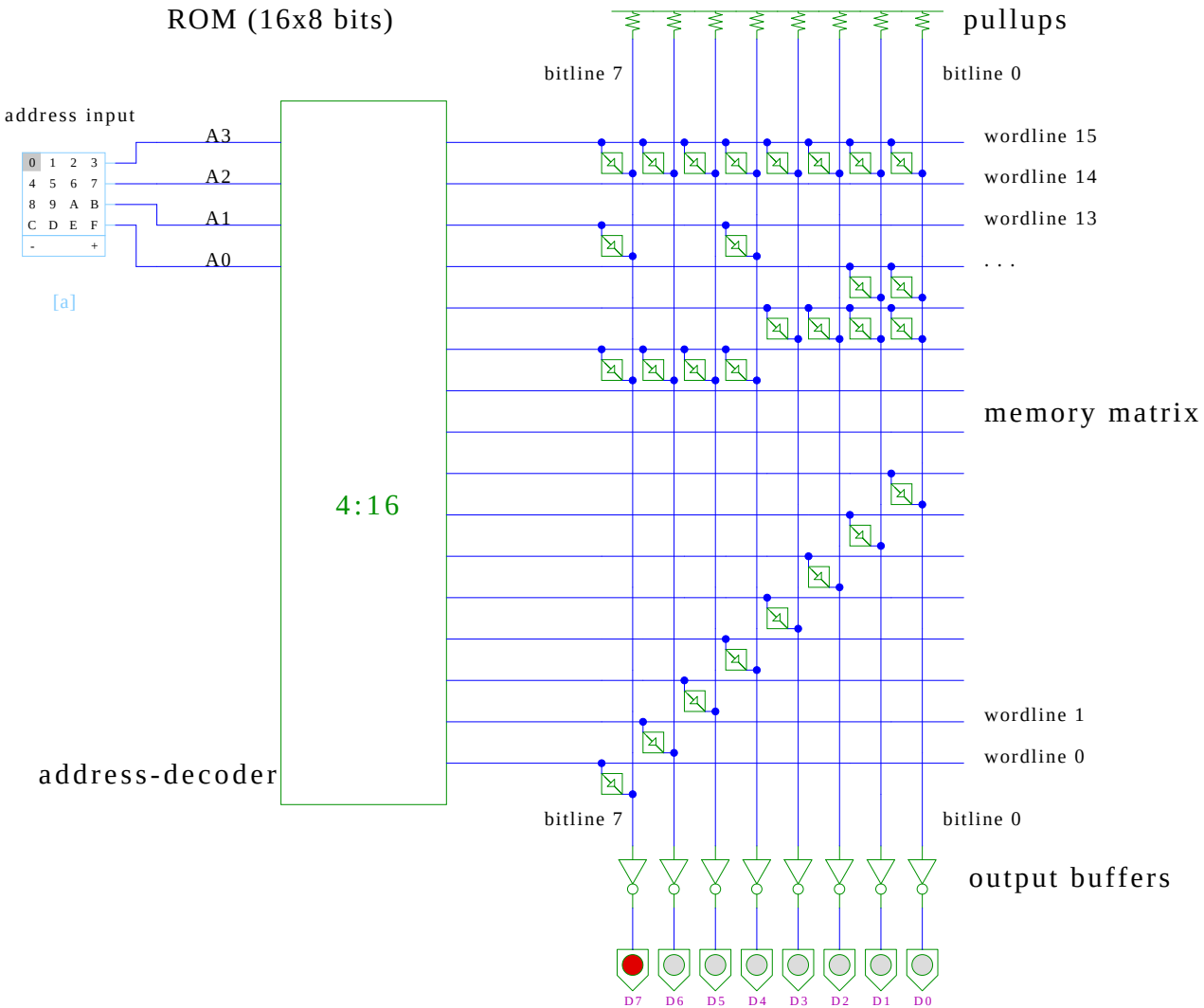
- System zur Speicherung von Information
- als Feld von N Adressen mit je m bit
- typischerweise mit n-bit Adressen und $N = 2^n$
- Kapazität also $2^n \times m$ bits

- 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	Cache
DRAM	Lesen/Schreiben	elektrisch	ja	ja	Hauptspeicher (alt)
SDRAM	Lesen/Schreiben	elektrisch	ja	ja	Hauptspeicher
ROM	nur Lesen	—	ja	nein	Geräte in großen Stückzahlen
PROM	nur Lesen	—	ja	nein	Geräte in kleinen Stückzahlen
EPROM	vorw. Lesen	UV-Licht	ja	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`

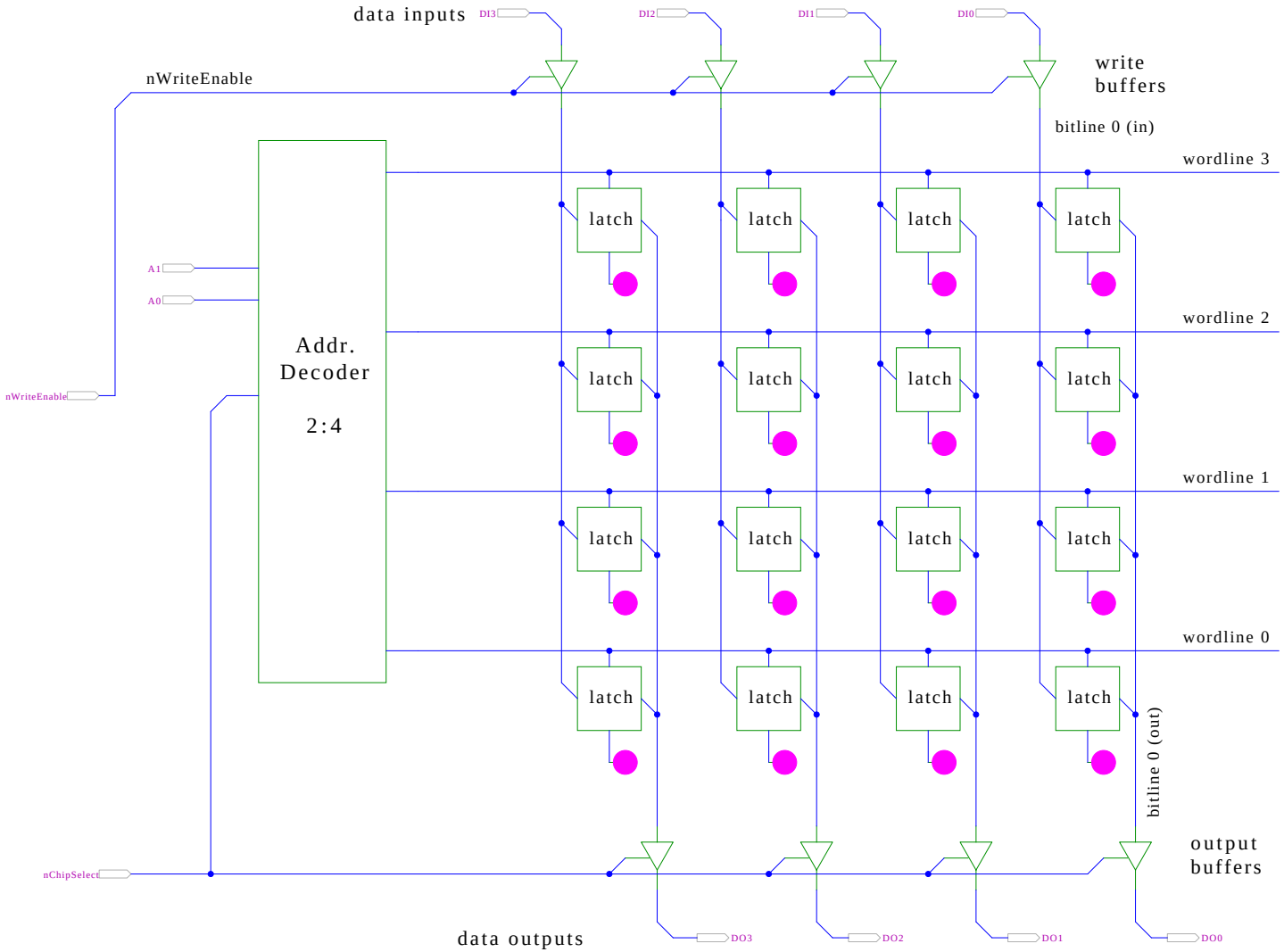
RAM: Random-Access Memory

Speicher, der im Betrieb gelesen und geschrieben werden kann

- Arbeitsspeicher des Rechners
- für Programme und Daten
- keine Abnutzungseffekte

- Aufbau als Matrixstruktur
- n Adressbits, konzeptionell 2^n Wortleitungen
- m Bits pro Wort
- Realisierung der einzelnen Speicherstellen?
  - statisches RAM: 6-Transistor Zelle — SRAM
  - dynamisches RAM: 1-Transistor Zelle — DRAM

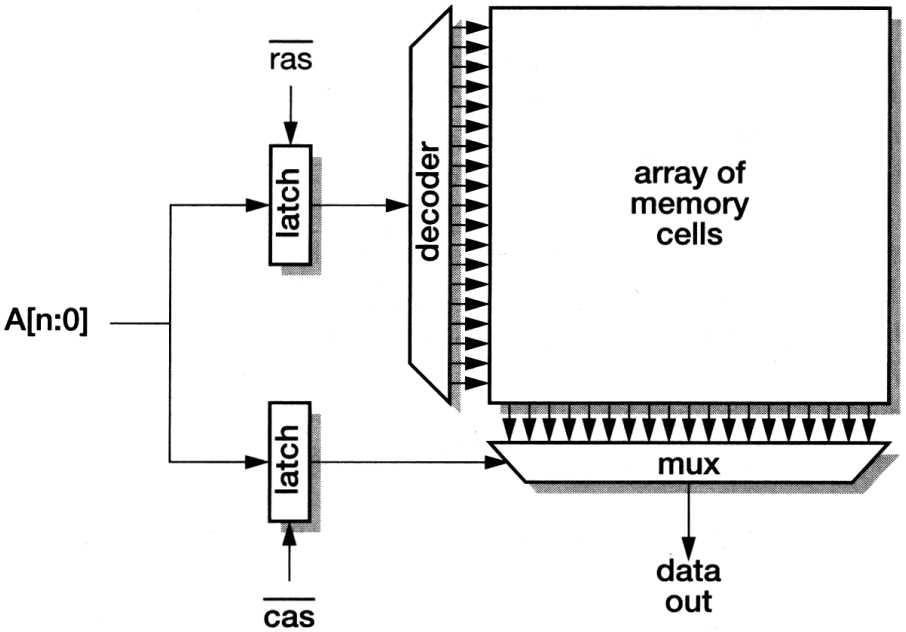
RAM: Blockschaltbild



4×4 bit
2-bit Adresse
4-bit Datenwort

[HenHA] Hades Webdemo:
`40-memories/40-ram/ram`

RAM: RAS/CAS-Adressdecodierung



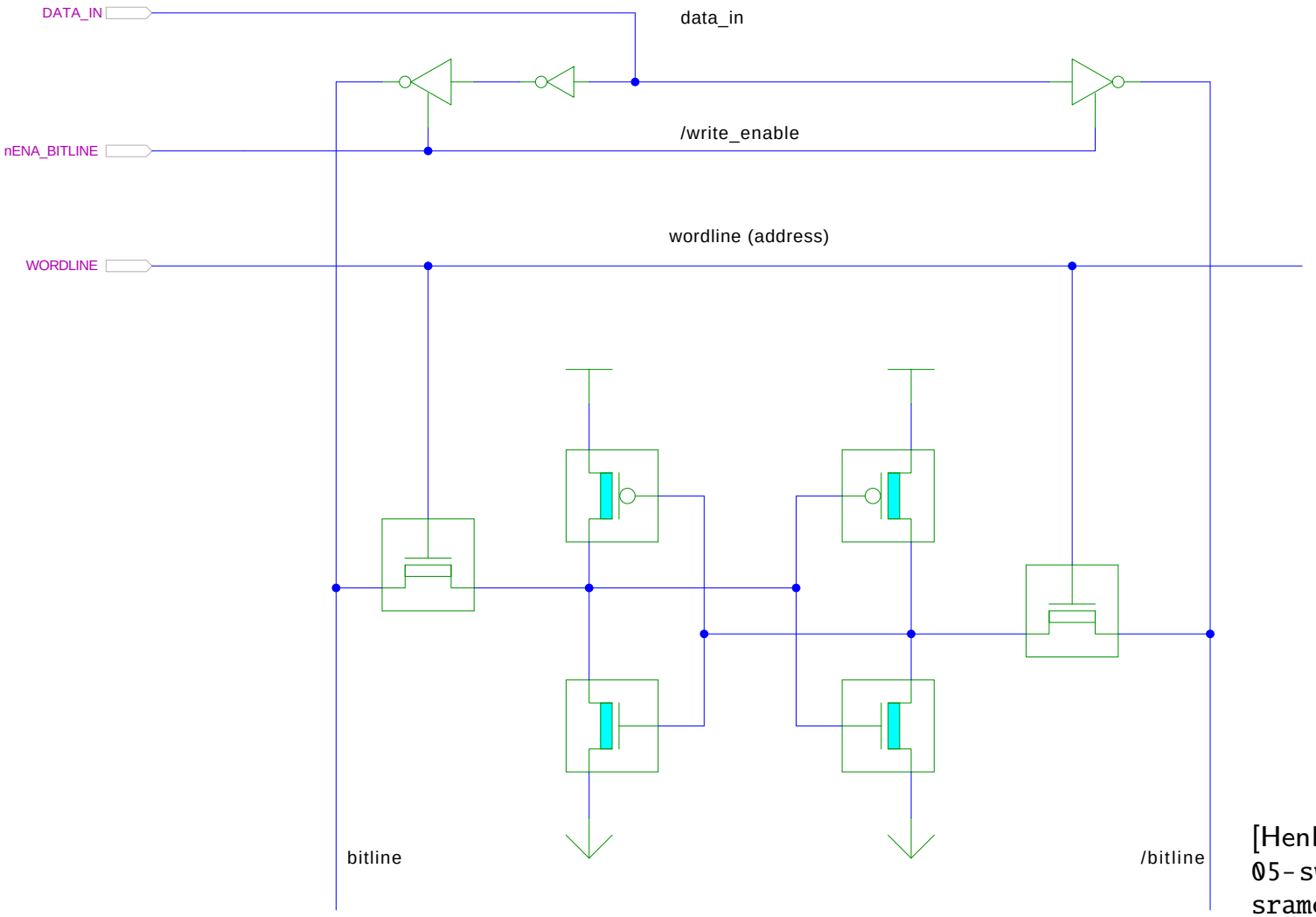
Furber: *ARM SoC Architecture* [Fur00]

- 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

SRAM: statisches RAM

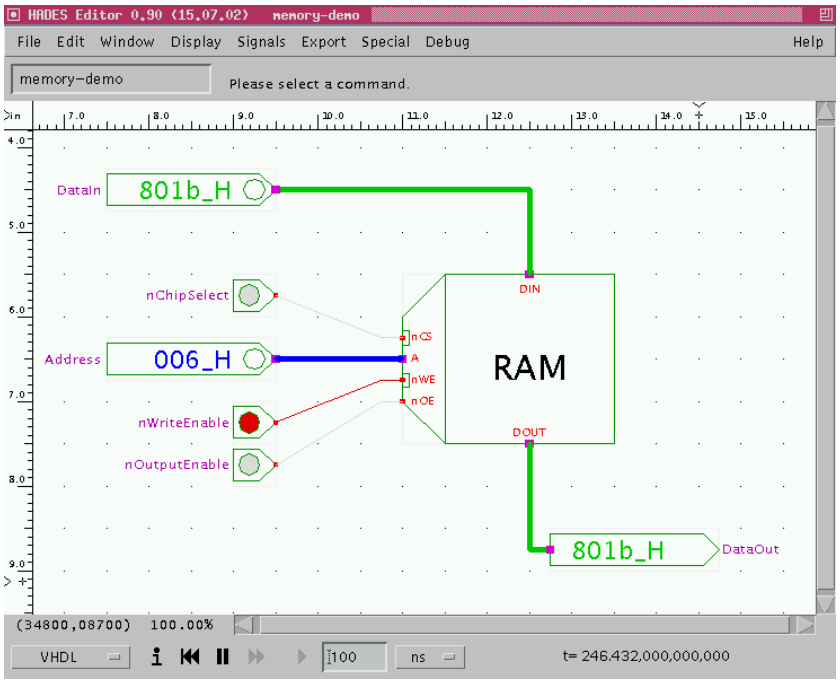
- Inhalt bleibt dauerhaft gespeichert solange Betriebsspannung anliegt
- *sechs-Transistor* Zelle zur Speicherung
  - weniger Platzverbrauch als Latches/Flipflops
  - kompakte Realisierung in CMOS-Technologie (s.u.)
  - zwei rückgekoppelte Inverter zur Speicherung
  - zwei n-Kanal Transistoren zur Anbindung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

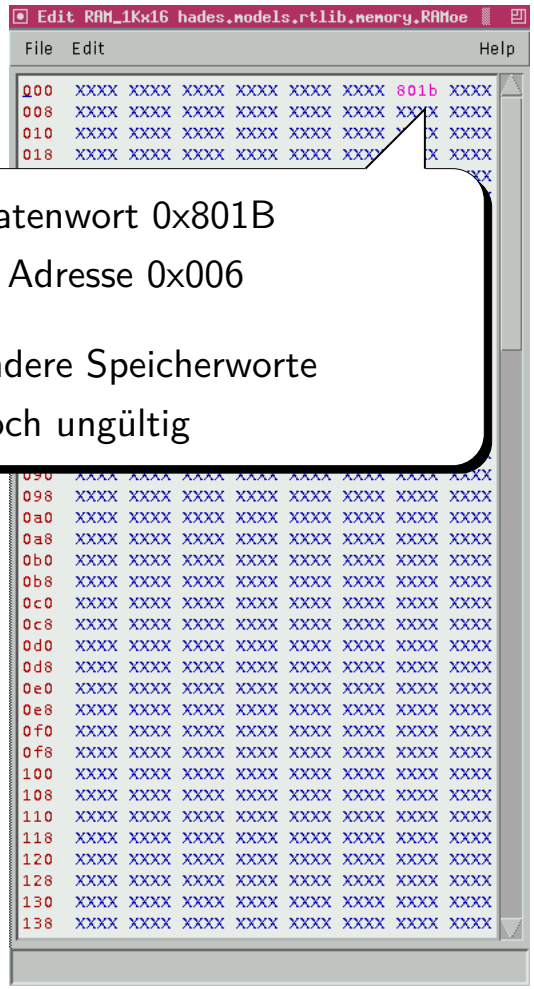
SRAM: Sechs-Transistor Speicherstelle („6T“)



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

SRAM: Hades Demo





Datenwort 0x801B
in Adresse 0x006

andere Speicherworte
noch ungültig

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

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

SRAM: Beispiel IC 6116

- 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`

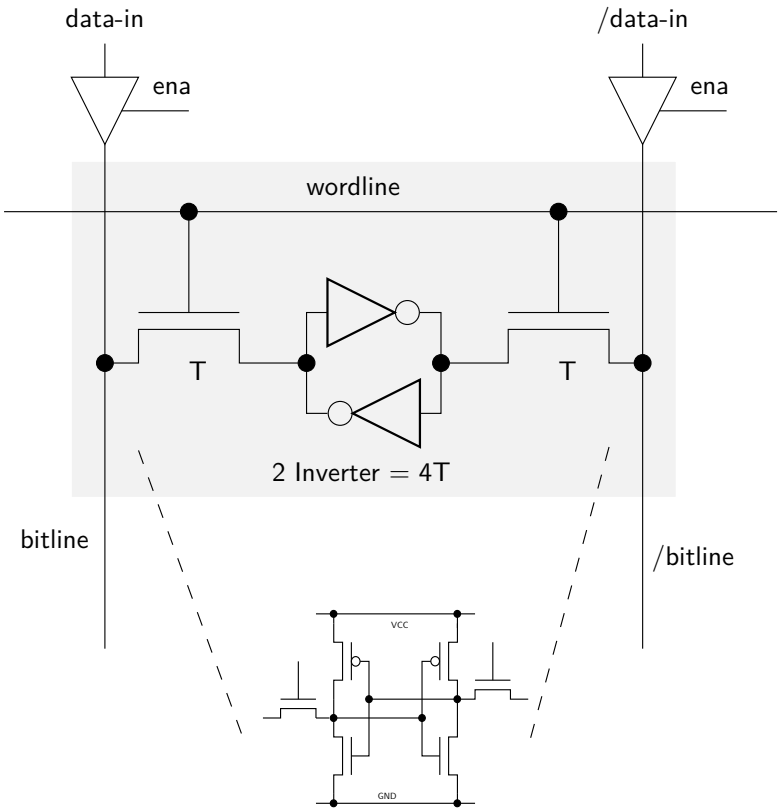
DRAM: dynamisches RAM

- Information wird in winzigen Kondensatoren gespeichert
- pro Bit je ein Transistor und Kondensator

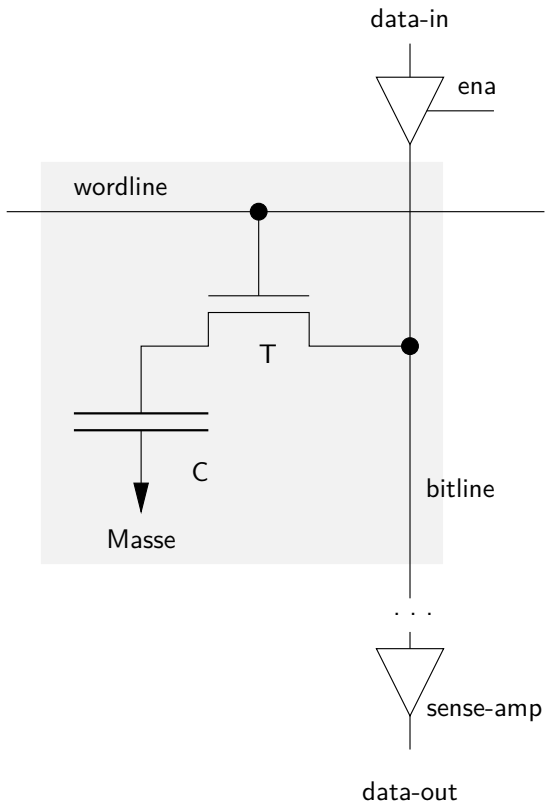
- jeder Lesezugriff entlädt den Kondensator
- *Leseverstärker* zur Messung der Spannung auf der Bitline
  Schwellwertvergleich zur Entscheidung logisch 0/1

– 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

DRAM vs. SRAM



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



- 1 Transistor/bit
- $C = 10\,\mathrm{fF} \approx 200\,000$ Elektronen
- langsam (sense-amp)
- minimale Fläche

DRAM: Stacked- und Trench-Zelle

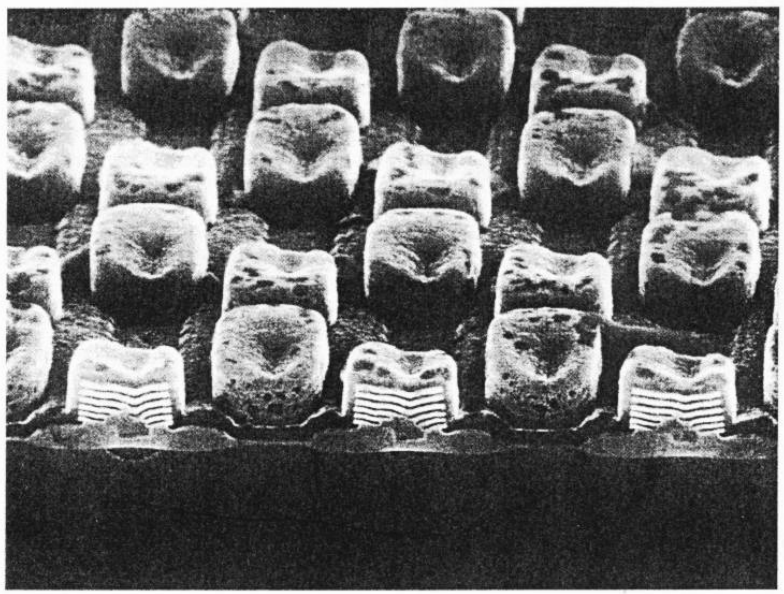
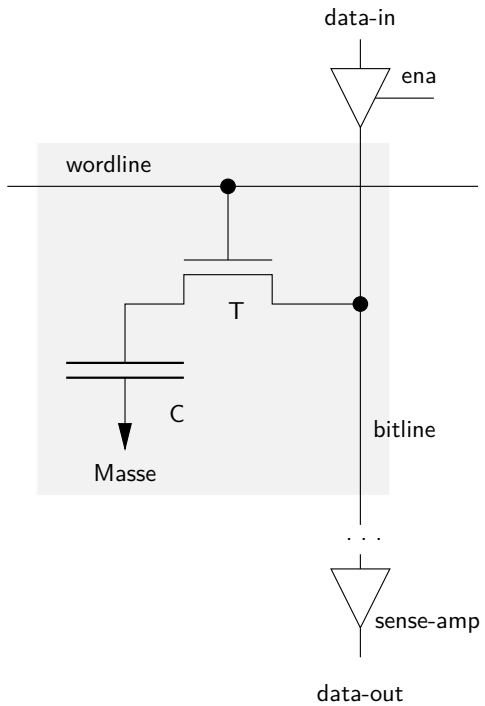


Abb. 7: Prototyp von Speicherzellen (Stapelkondensatoren) für zukünftige Speicherchips wie den Ein-Gigabit-Chip. Da für DRAM-Chips eine minimale Speicherkapazität von 25 fF notwendig ist, bringt es erhebliche Platzvorteile, die Kondensatorelemente vertikal übereinander zu stapeln. Die Dicke der Schichten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

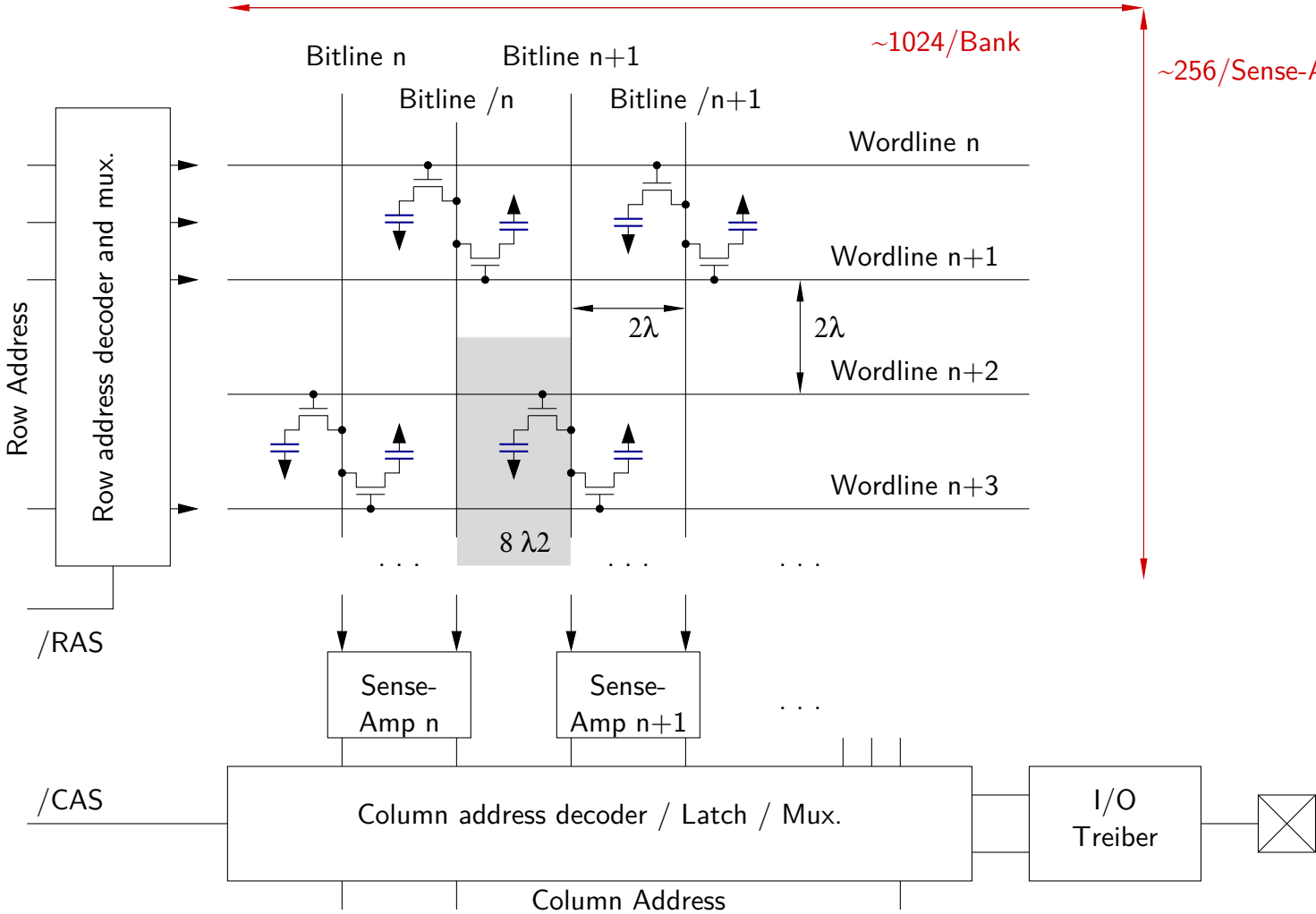
Siemens 1 Gbit DRAM



IBM CMOS-6X embedded DRAM

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

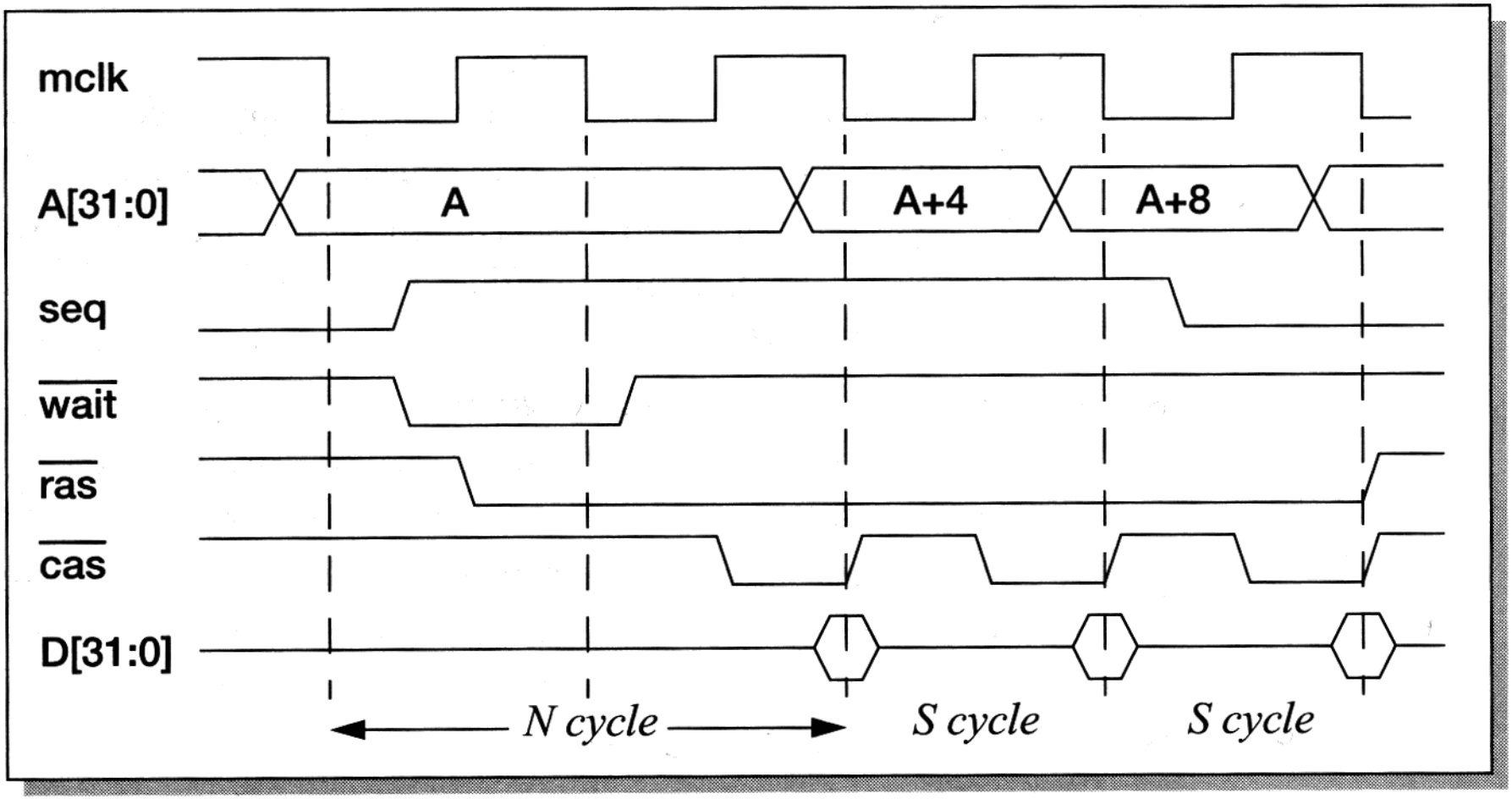
DRAM: Layout


~1024/Bank
~256/Sense-A
Bitline n
Bitline n+1
Bitline /n
Bitline /n+1
Wordline n
Wordline n+1
Wordline n+2
Wordline n+3
2λ
2λ
8 λ2
Row Address
Row address decoder and mux.
/RAS
Sense-
Amp n
Sense-
Amp n+1
/CAS
Column address decoder / Latch / Mux.
I/O
Treiber
Column Address

DRAM: Varianten

- 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, DDR3, DDR4: Varianten mit höherer Taktrate
    aktuell Übertragungsraten bis 25,6 GByte/sec
  - GDDR3...GDDR5 (*Graphics Double Data Rate*)
    bis 5-7 GBit/sec pro Leitung ($\times$ 128-256 Datenleitungen)

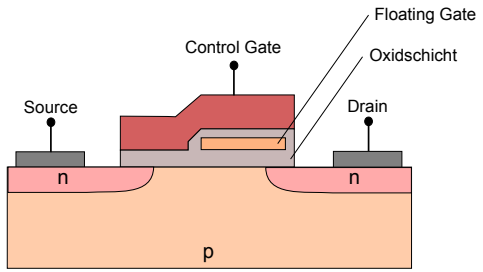
SDRAM: Lesezugriff auf sequenzielle Adressen



[Fur00]

Flash

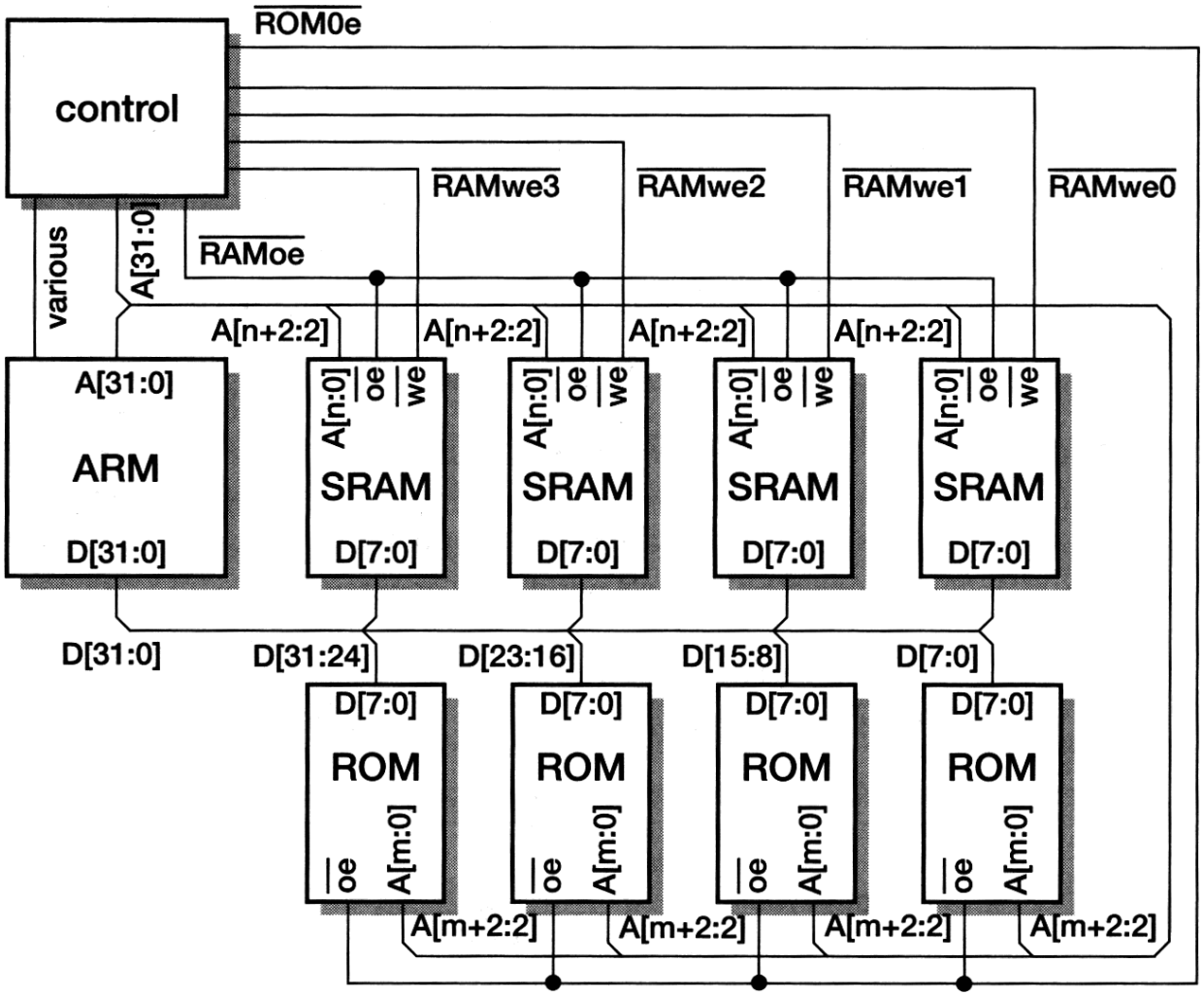
- ähnlich kompakt und kostengünstig wie DRAM
- nichtflüchtig (*non-volatile*): Information bleibt beim Ausschalten erhalten



- 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. . . 100 000 Schreibzugriffe möglich

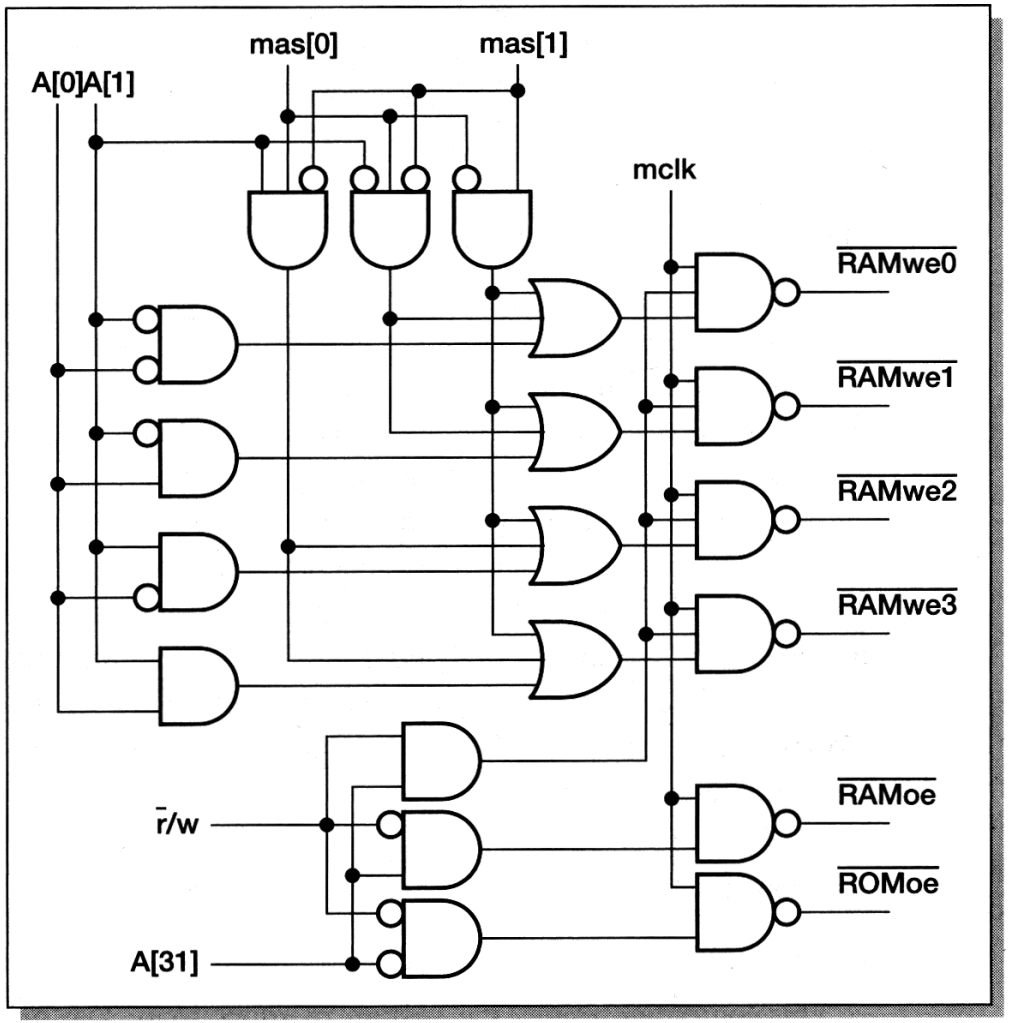
Typisches Speichersystem



32-bit Prozessor
4× 8-bit SRAMs
4× 8-bit ROMs

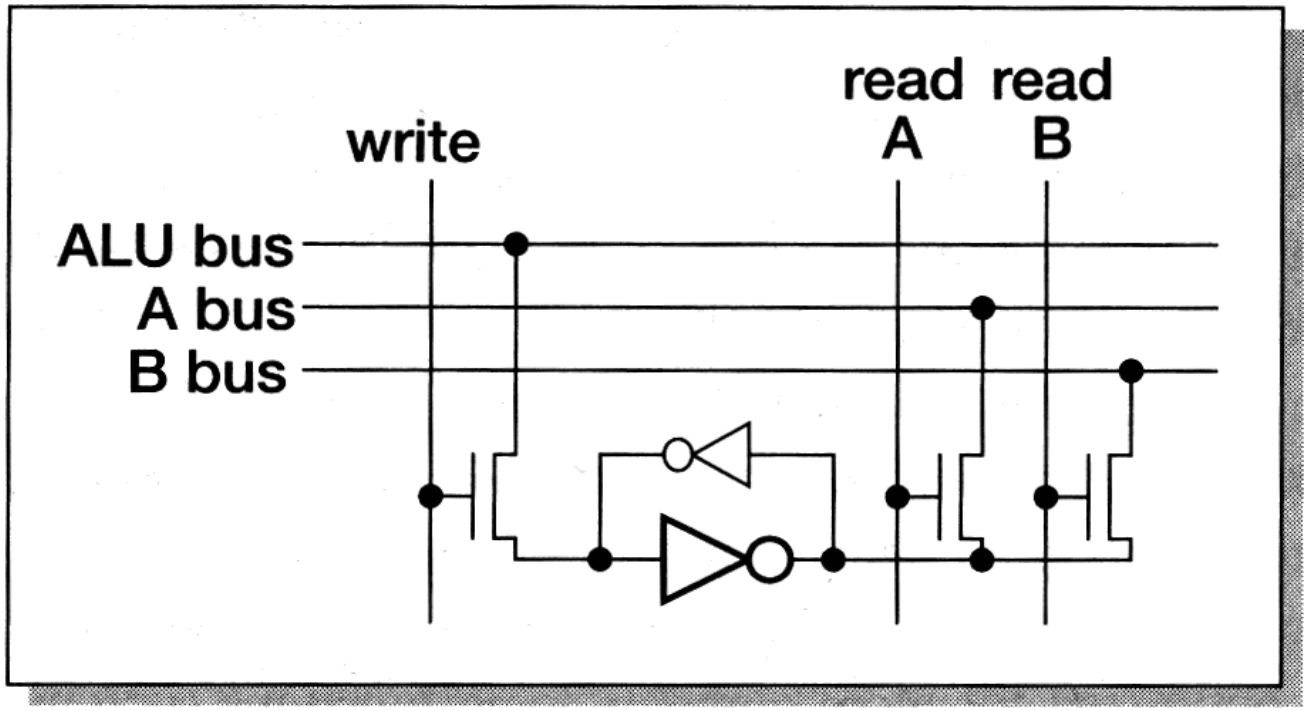
[Fur00]

Typisches Speichersystem: Adressdecodierung



[Fur00]

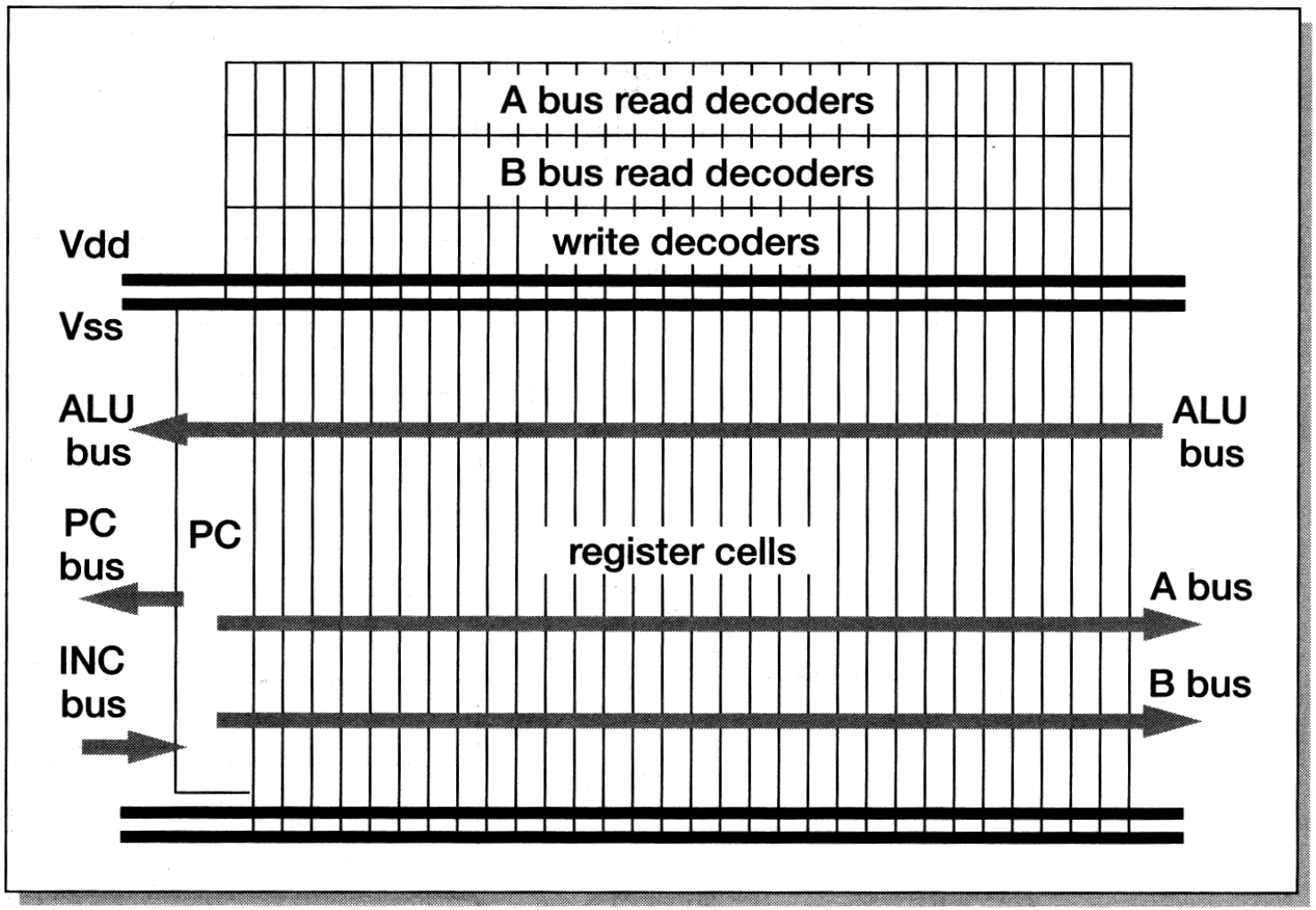
Multi-Port-Registerbank: Zelle



[Fur00]

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

Multi-Port Registerbank: Floorplan/Chiplayout



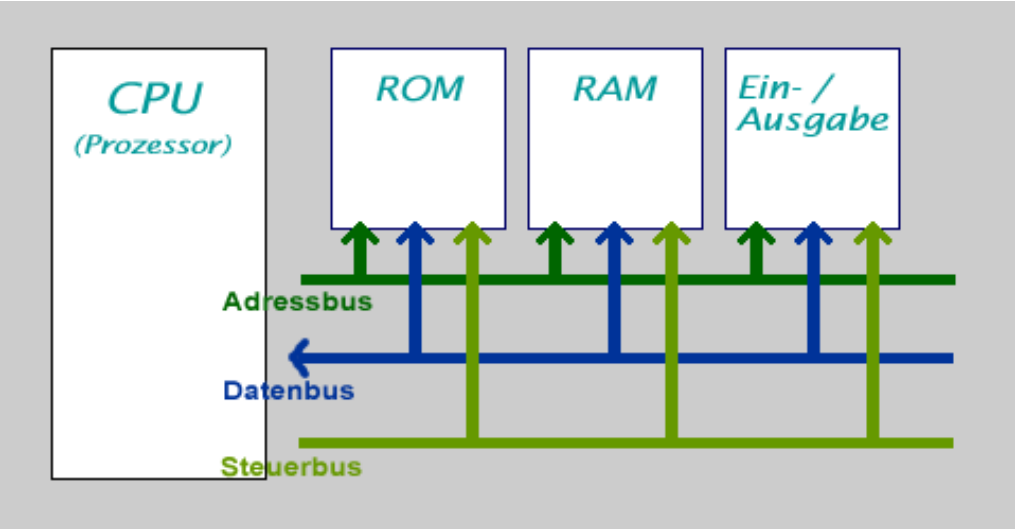
[Fur00]

Bussysteme

- **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

Bussysteme (cont.)

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



Bussysteme (cont.)

- 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^2C, System-Management-Bus...
  - lange Leitungen: RS232, Ethernet...
  - Funkmedium: WLAN, Bluetooth (logische Verbindung)

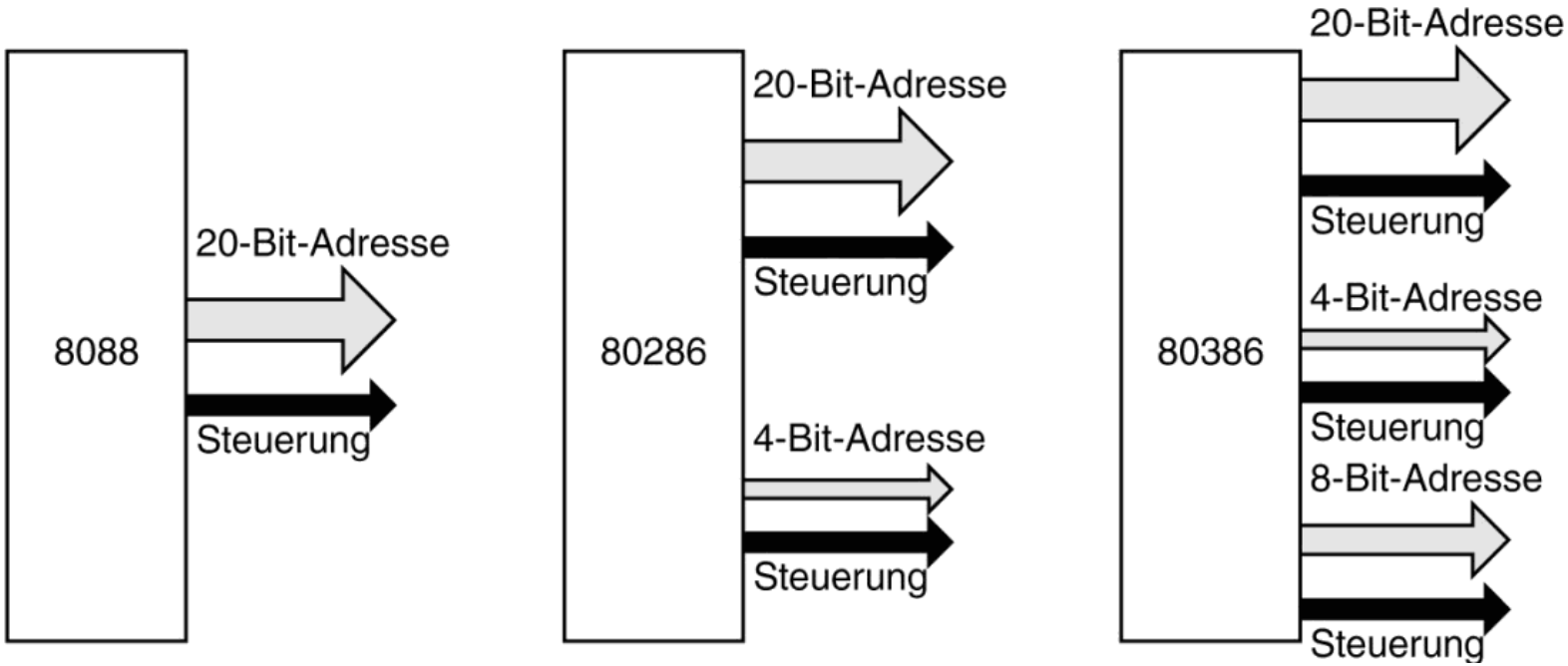
Bus: Mikroprozessorsysteme

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

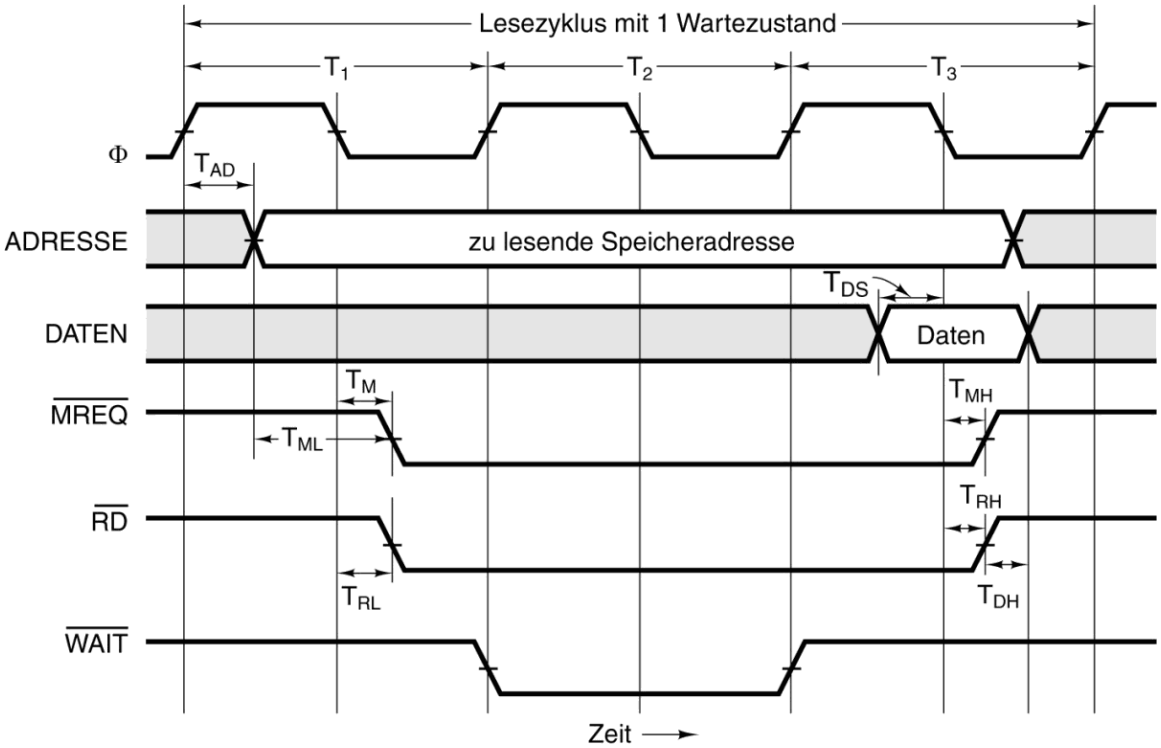
Adressbus: Evolution beim Intel x86



[TA14]

- 20-bit: 1 MiByte Adressraum
  24-bit: 16 MiByte
  32-bit: 4 GiByte
- alle Erweiterungen abwärtskompatibel

Synchroner Bus: Timing



[TA14]

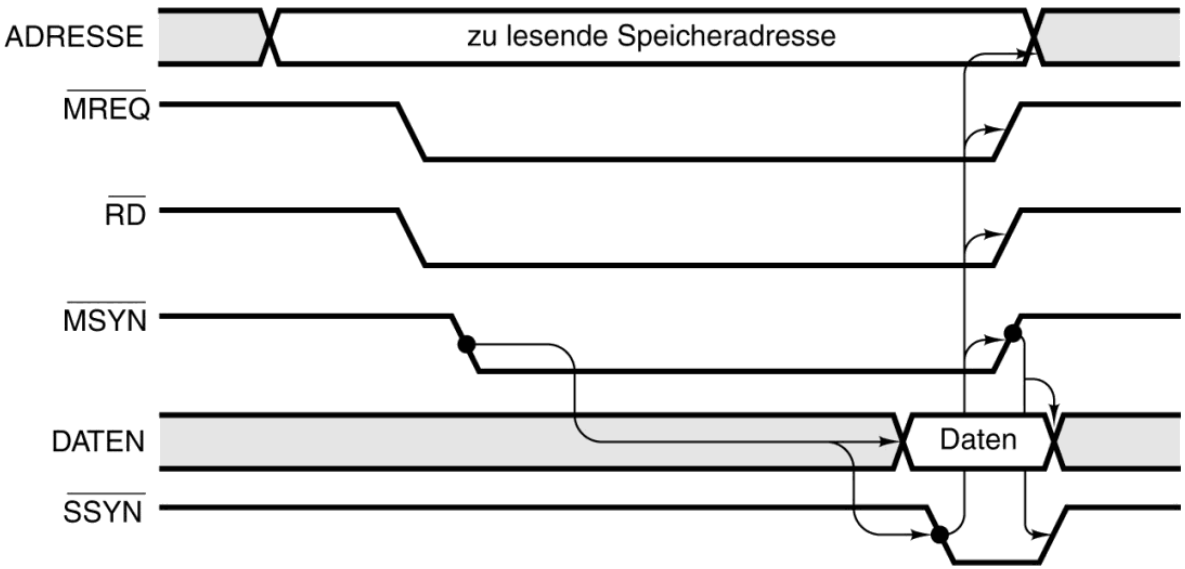
- alle Zeiten über Taktsignal Φ gesteuert
- $\overline{MREQ}$-Signal zur Auswahl Speicher oder I/O-Geräte
- $\overline{RD}$ signalisiert Lesezugriff
- Wartezyklen, solange der Speicher $\overline{WAIT}$ aktiviert

Synchroner Bus: Timing (cont.)

- typische Parameter

Symbol	Parameter	[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 Φ in T_1		3
T_{RL}	RD-Verzögerung nach fallender Flanke von Φ in T_1		3
T_{DS}	*Setup*-Zeit vor fallender Flanke von Φ	2	
T_{MH}	$\overline{MREQ}$-Verzögerung nach fallender Flanke von Φ in T_3		3
T_{RH}	$\overline{RD}$-Verzögerung nach fallender Flanke von Φ 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

Bus Arbitrierung

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

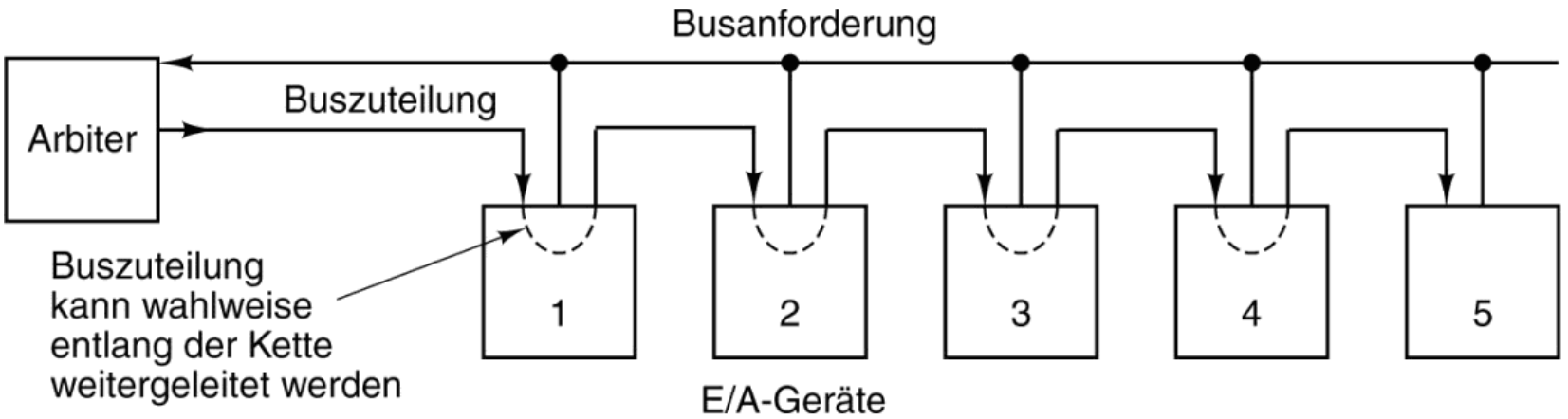
1. zentrale Arbitrierung
   - Arbiter gewährt Bus-Requests
   - Strategien
     - Prioritäten für verschiedene Geräte
     - „round-robin“ Verfahren
     - „Token“-basierte Verfahren
     - usw.

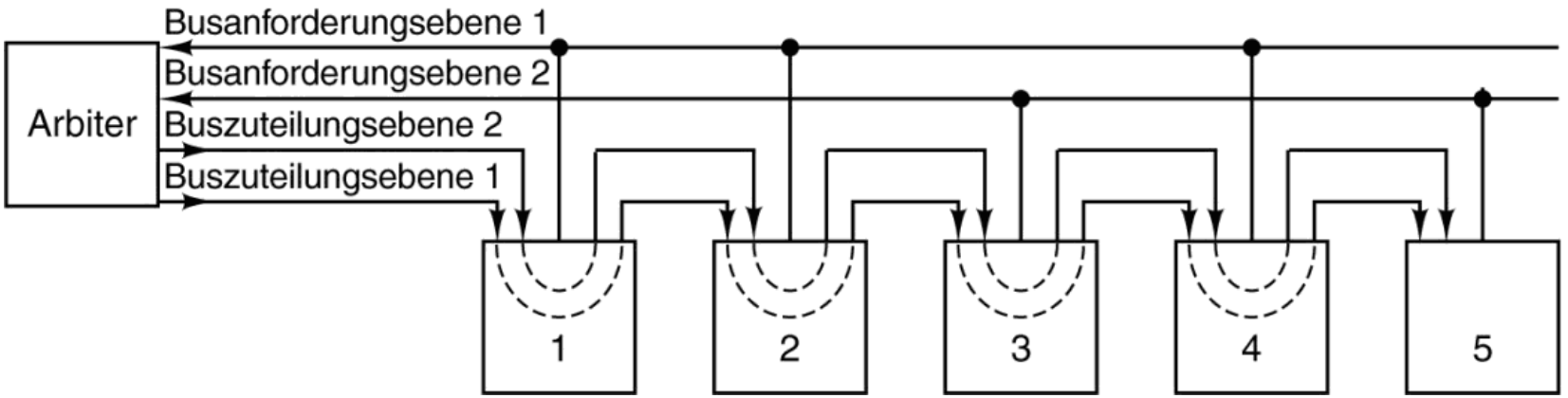
Bus Arbitrierung (cont.)

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

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

Bus Arbitrierung (cont.)





[TA14]

Bus Bandbreite

- Menge an (Nutz-) Daten, die pro Zeiteinheit übertragen werden kann
- zusätzlicher Protokolloverhead ⇒ Brutto- / Netto-Datenrate
RS232	50	Bit/sec	...	460	KBit/sec
I^2C	100	KBit/sec (Std.)	...	3,4	MBit/sec (High Speed)
USB	1,5	MBit/sec (1.x)	...	5	GBit/sec (3.0)
ISA	128	MBit/sec			
PCI	1	GBit/sec (2.0)	...	4,3	GBit/sec (3.0)
AGP	2,1	GBit/sec (1x)	...	16,6	GBit/sec (8x)
PCIe	250	MByte/sec (1.x)	...	1000	MByte/sec (3.0) x1...32
HyperTransport	12,8	GByte/sec (1.0)	...	51,2	GByte/sec (3.1)

- `en.wikipedia.org/wiki/List_of_device_bit_rates`

Beispiel: PCI-Bus

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

- Auto-Konfiguration
  - angeschlossene Geräte werden automatisch erkannt
  - eindeutige Hersteller- und Geräte-Nummern
  - Betriebssystem kann zugehörigen Treiber laden
  - automatische Zuweisung von Adressbereichen und IRQs

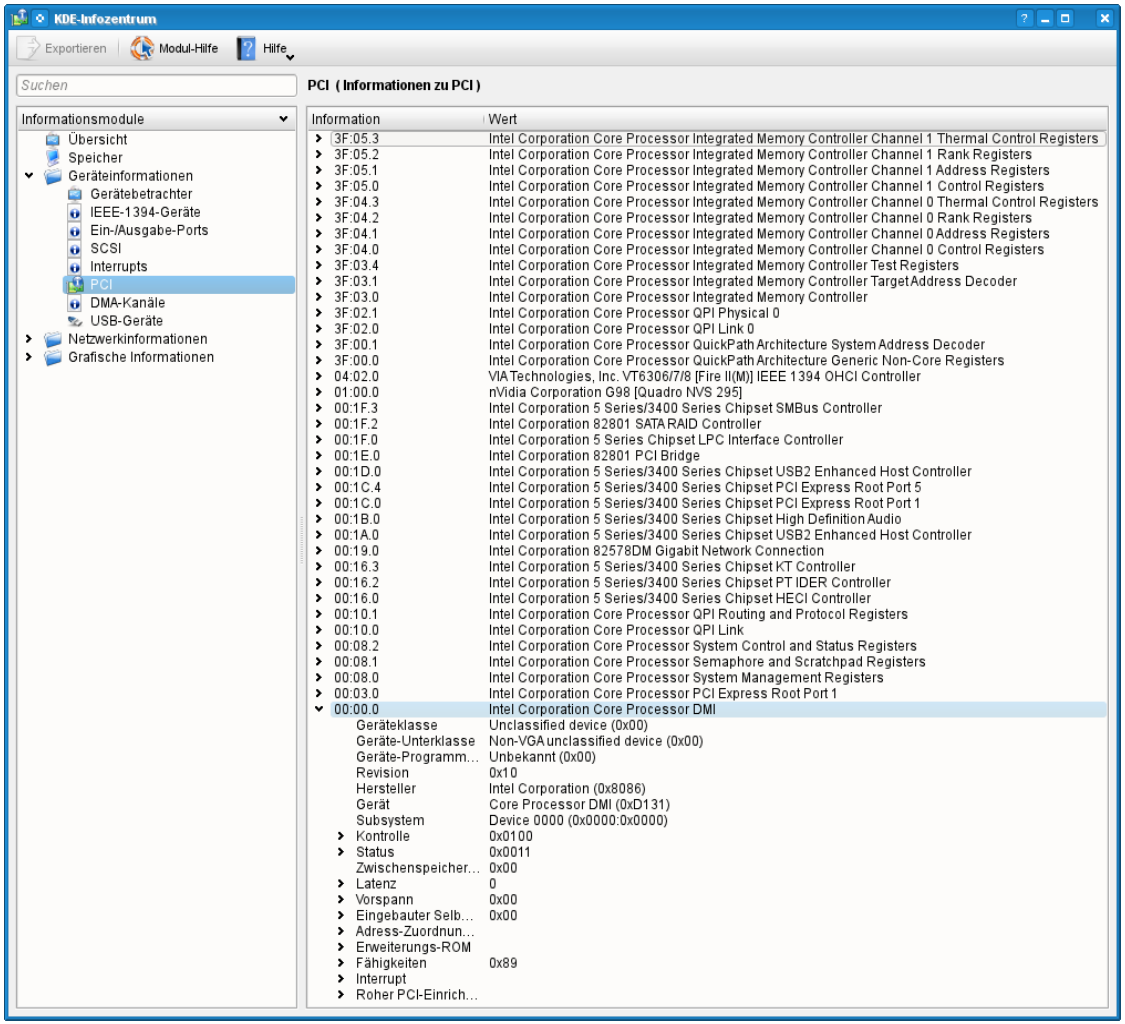
PCI-Bus: Peripheriegeräte

```
[maeder@tams159]~> lspci -v
00:00.0 Host bridge: Intel Corporation Haswell-ULT DRAM Controller (rev 0b)
        Subsystem: Dell Device 05ca
        Flags: bus master, fast devsel, latency 0
        Capabilities: <access denied>

00:02.0 VGA compatible controller: Intel Corporation Haswell-ULT Integrated Graphics
Controller (rev 0b) (prog-if 00 [VGA controller])
        Subsystem: Dell Device 05ca
        Flags: bus master, fast devsel, latency 0, IRQ 63
        Memory at f7800000 (64-bit, non-prefetchable) [size=4M]
        Memory at e0000000 (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
        Kernel modules: i915

00:03.0 Audio device: Intel Corporation Device 0a0c (rev 0b)
        Subsystem: Dell Device 05ca
        Flags: bus master, fast devsel, latency 0, IRQ 65
        Memory at f7e34000 (64-bit, non-prefetchable) [size=16K]
        ...
```

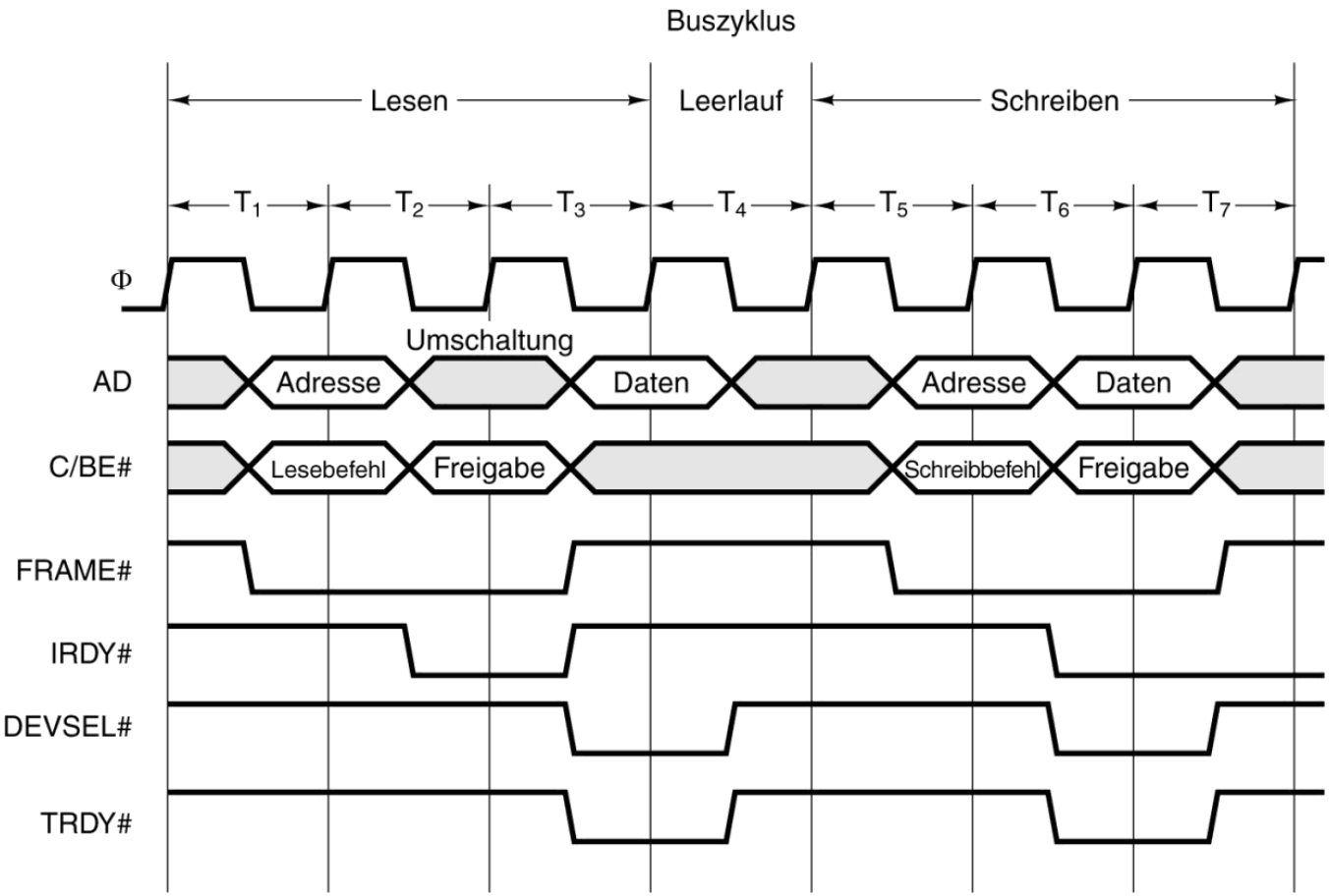
PCI-Bus: Peripheriegeräte (cont.)



PCI-Bus: Leitungen („mindestens“)

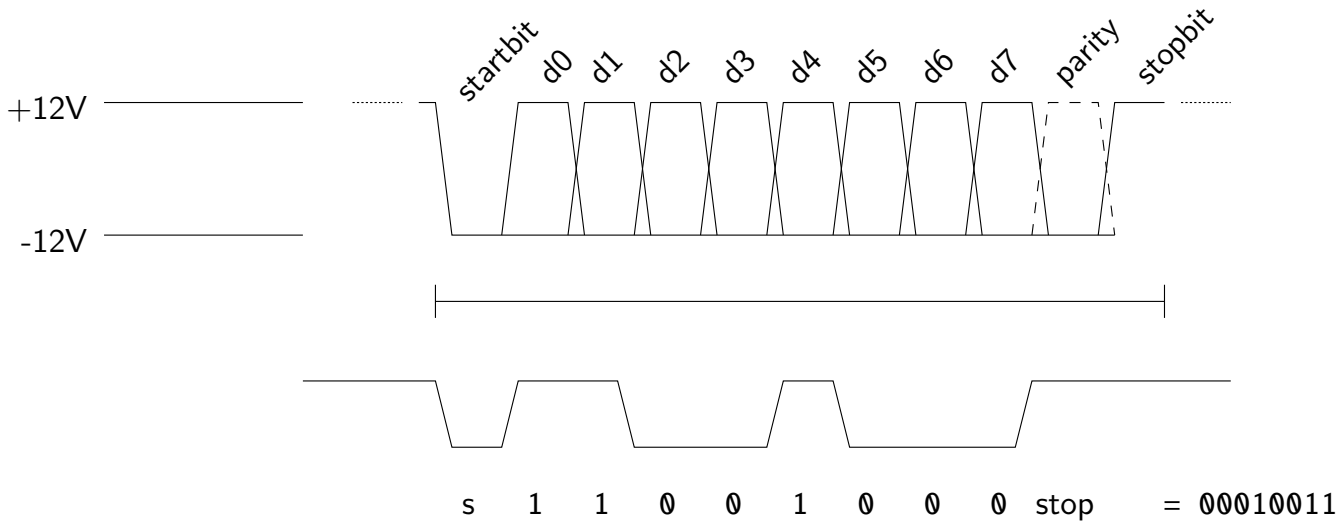
Signal	Leitungen	Master	Slave	Beschreibung
CLK	1			Takt (33 oder 66 MHz)
AD	32	×	×	Gemultiplexte Adress- und Datenleitungen
PAR	1	×		Adress- uder Datenparitätsbit
C/BE	4	×		Busbefehl/Bitmap für Byte Enable (zeigt gültige Datenbytes an)
FRAME#	1	×		Kennzeichnet, dass AD und C/BE aktiviert sind
IRDY#	1	×		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			Adressparätätsfehler 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



[TA14]

RS-232: Serielle Schnittstelle



- Baudrate 300, 600, . . . , 19200, 38400, 115200 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

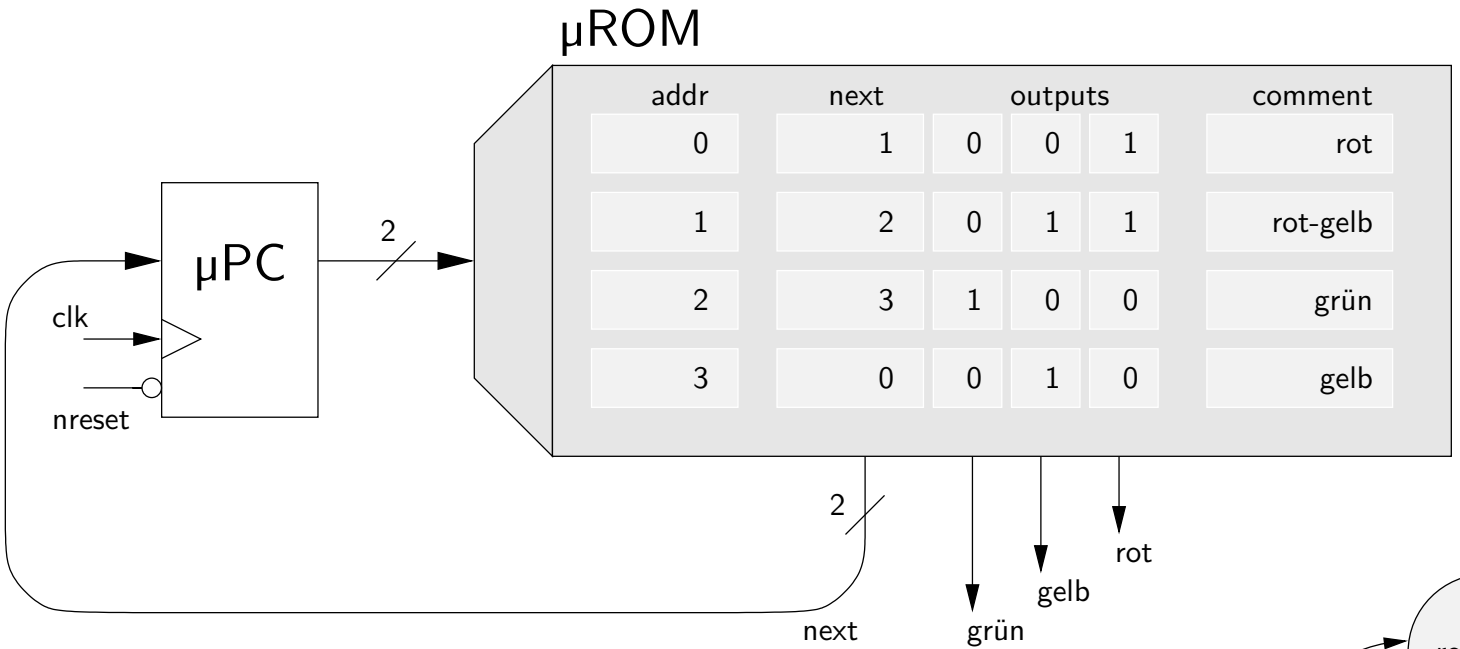
Ablaufsteuerung mit Mikroprogramm

- als Alternative zu direkt entworfenen Schaltwerken

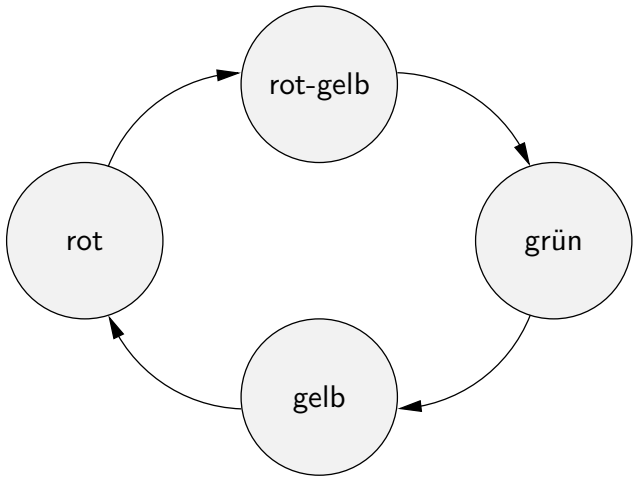
- *Mikroprogrammzähler* μPC: Register für aktuellen Zustand
- μPC adressiert den Mikroprogrammspeicher μROM
- μROM konzeptionell in mehrere Felder eingeteilt
  - die verschiedenen Steuerleitungen
  - ein oder mehrere Felder für Folgezustand
  - ggf. zusätzliche Logik und Multiplexer zur Auswahl unter mehreren Folgezuständen
  - ggf. Verschachtelung und Aufruf von Unterprogrammen: „nanoProgramm“

- siehe „Praktikum Rechnerstrukturen“

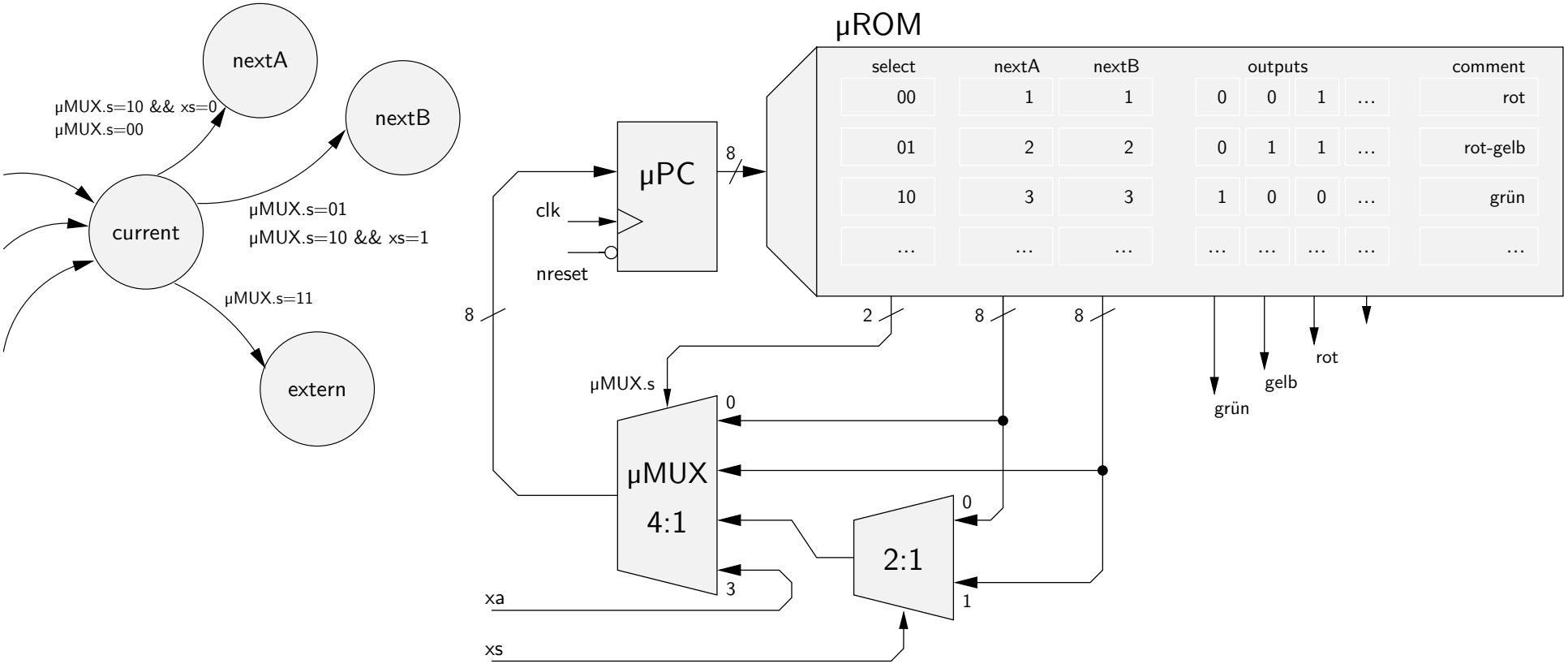
Mikroprogramm: Beispiel Ampel



- μPC adressiert das μROM
- *next*-Ausgang liefert Folgezustand
- andere Ausgänge steuern die Schaltung = die Lampen der Ampel

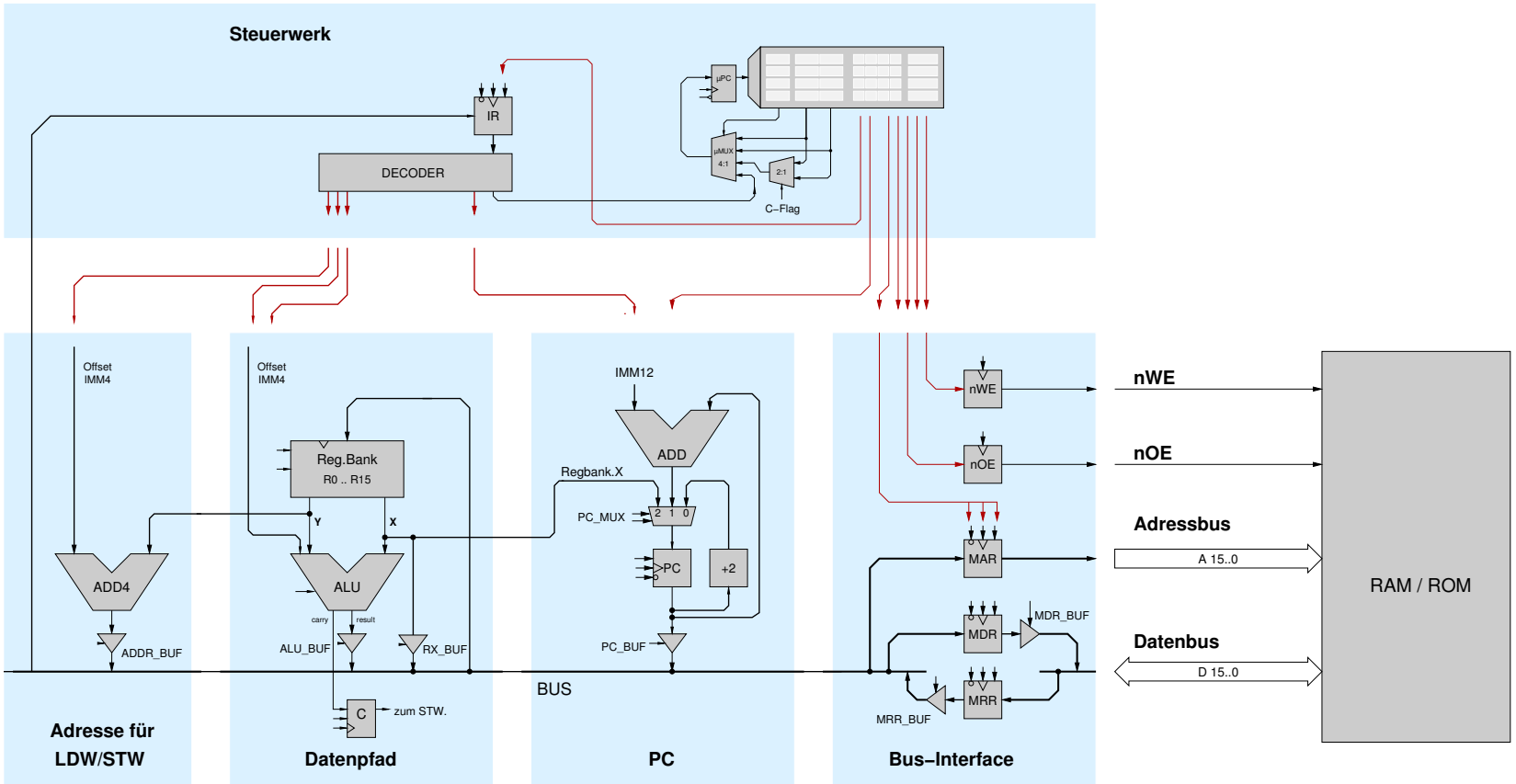


Mikroprogramm: Beispiel zur Auswahl des Folgezustands

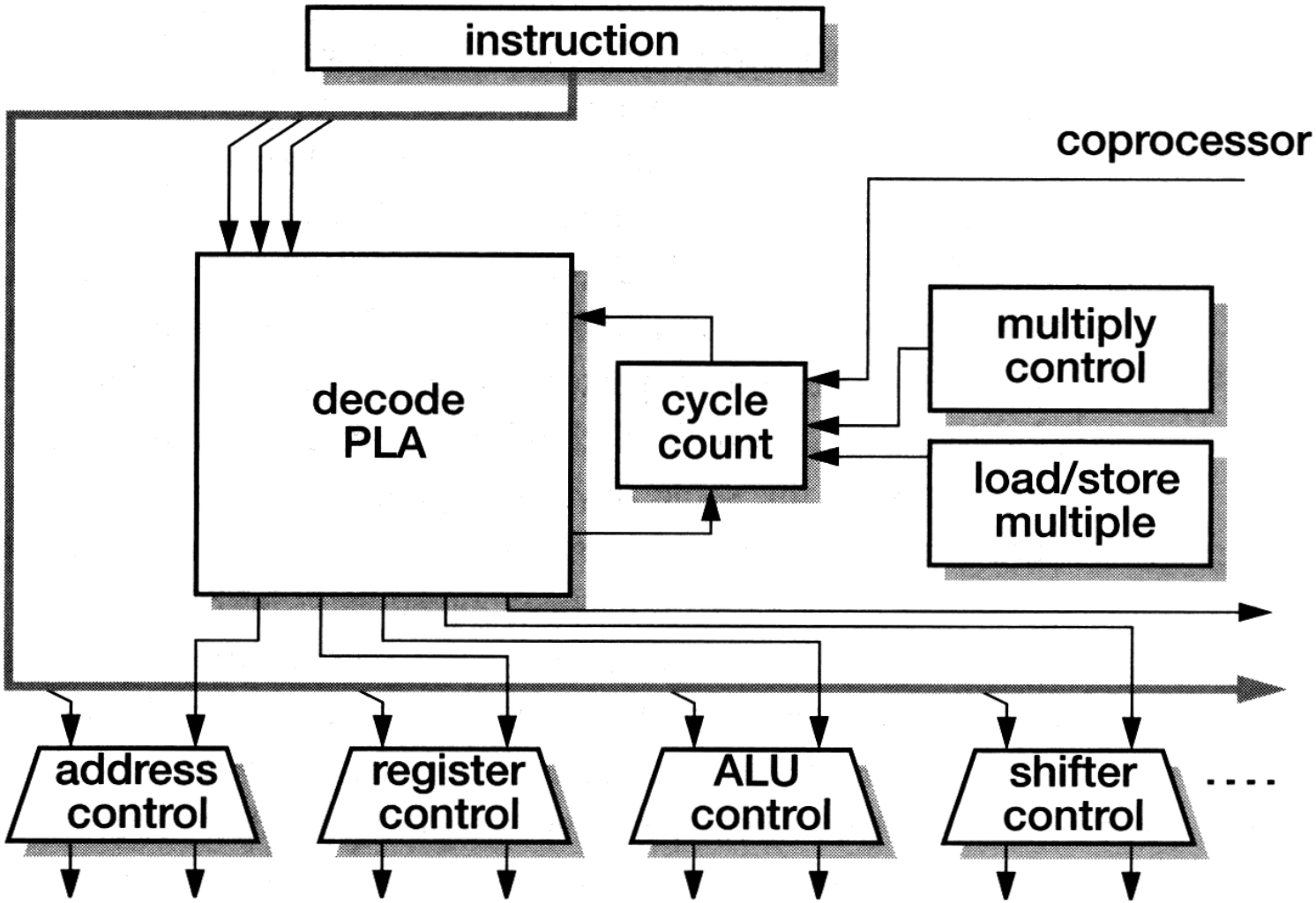


- Multiplexer erlaubt Auswahl des μPC Werts
- $nextA$, $nextB$ aus dem μROM, externer xa Wert
- xs Eingang für bedingte Sprünge

RS-Praktikum: D*CORE Prozessor

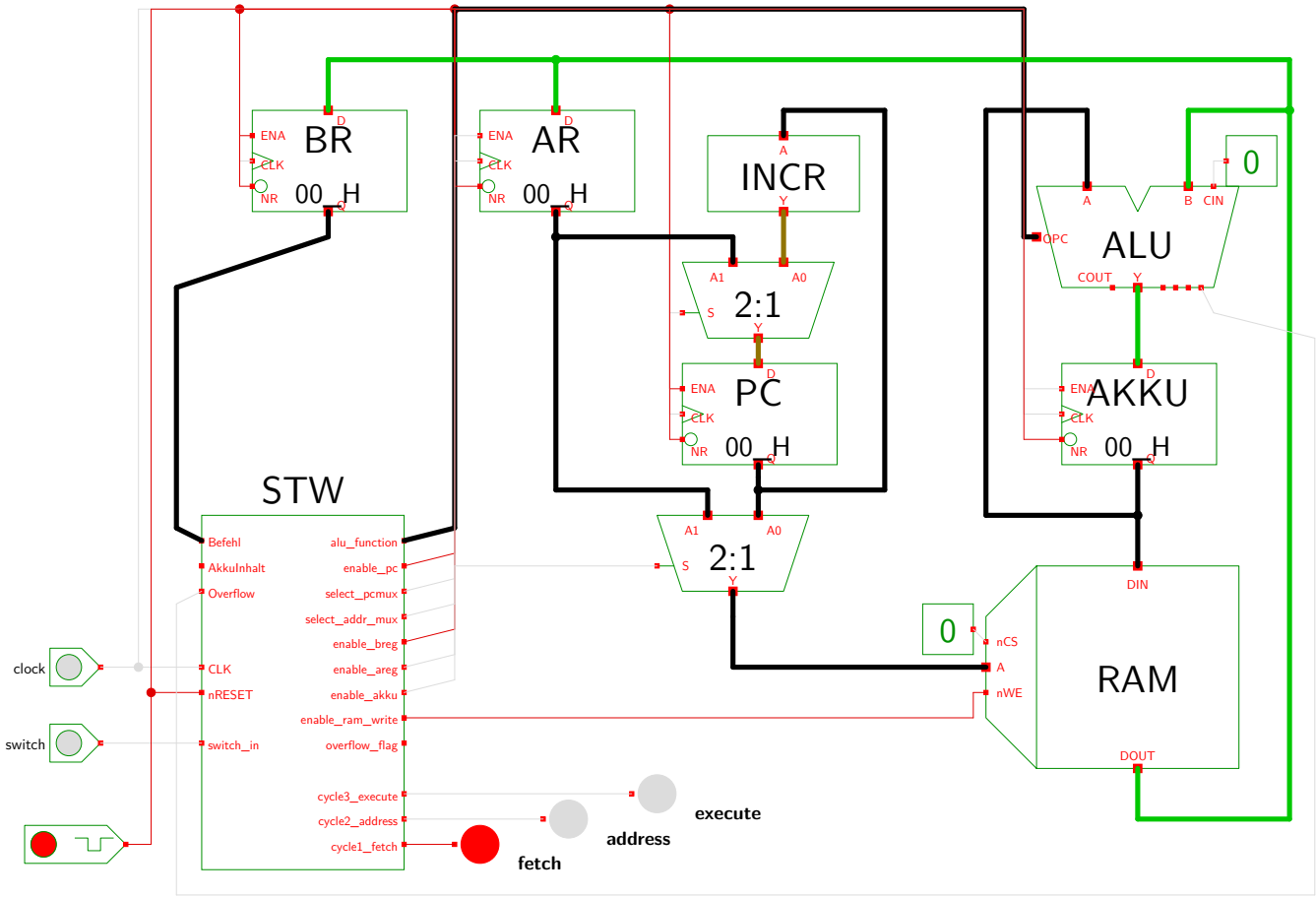


Mikroprogramm: Befehlsdecoder des ARM 7 Prozessors



[Fur00]

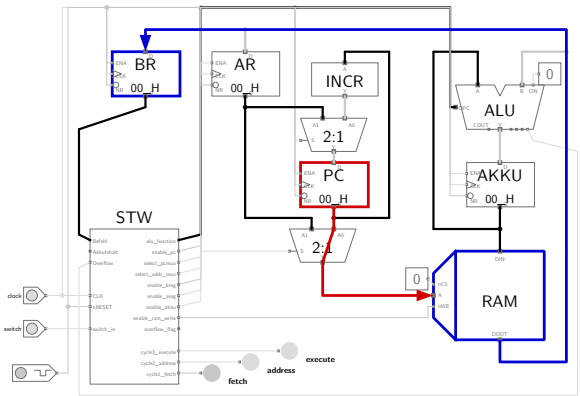
Beispiele: PRIMA: die Primitive Maschine



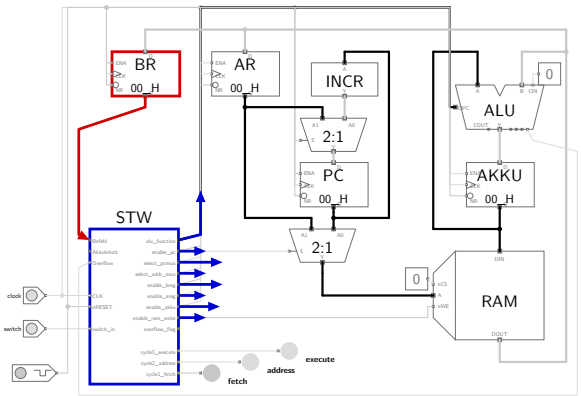
[HenHA] Hades Demo: `50-rtlib/90-prima/prima`

PRIMA: die Zyklen

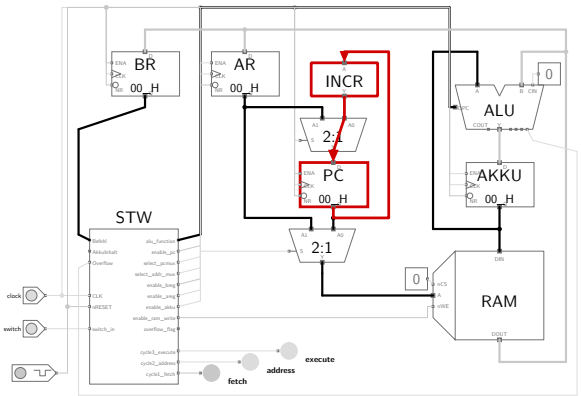
Befehl holen

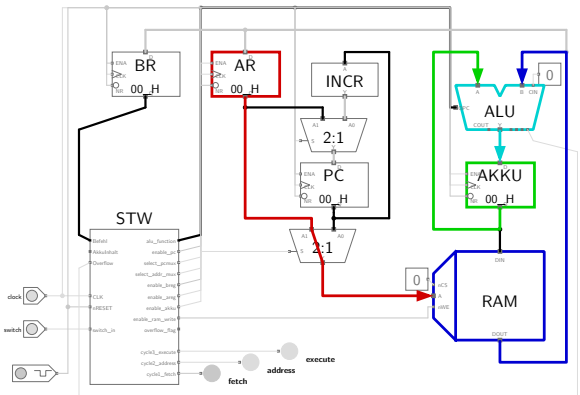


decodieren

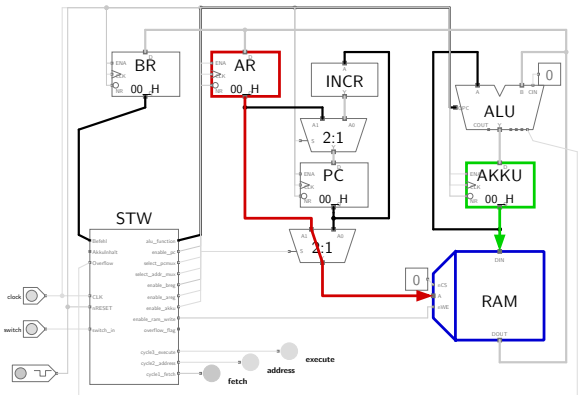


PC inkrementieren

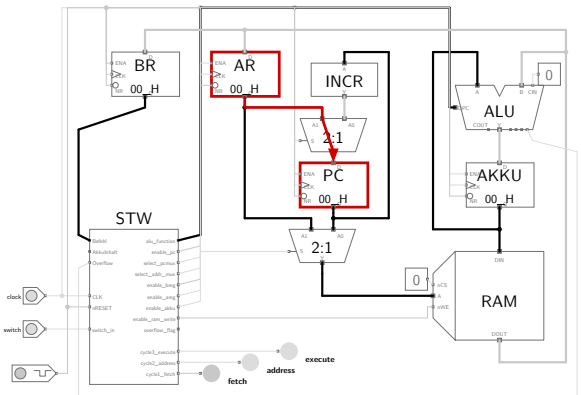




rechnen

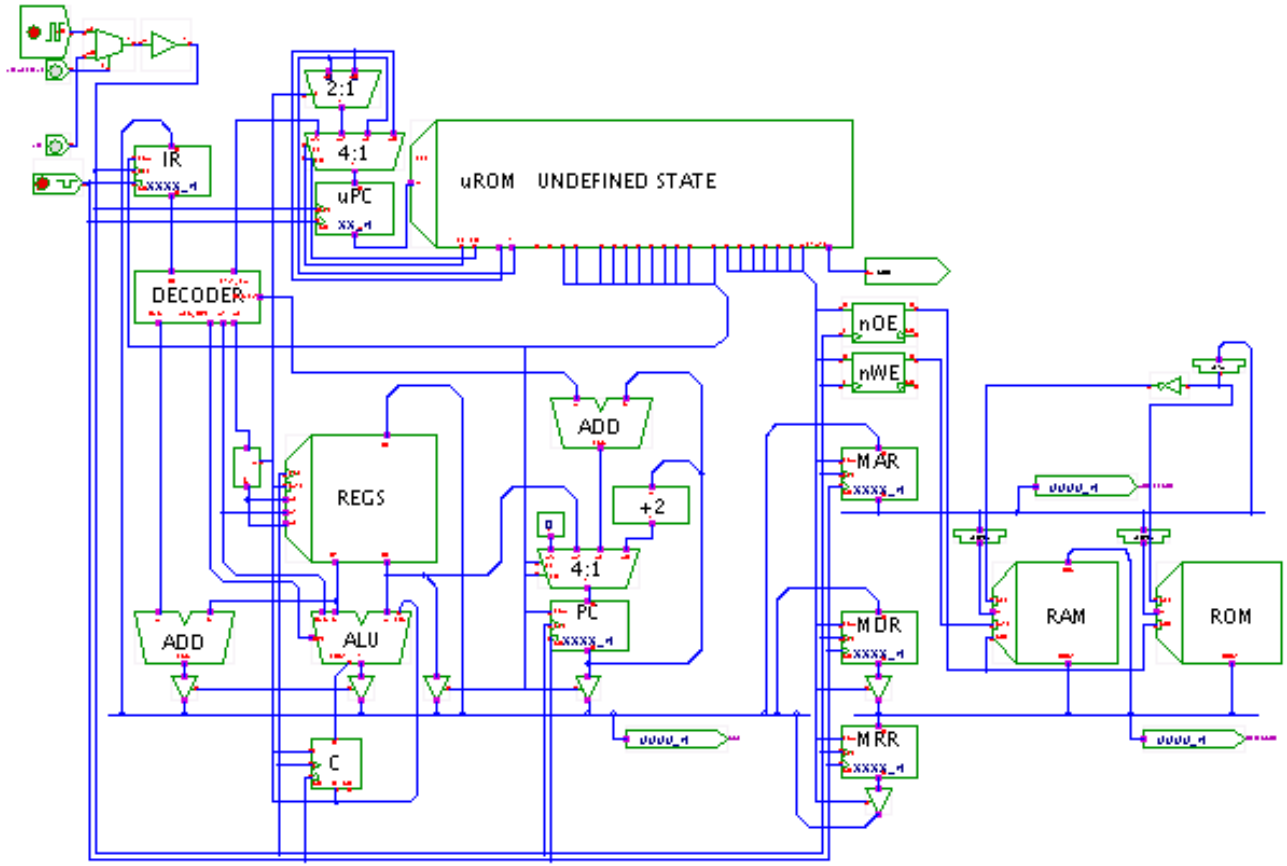


speichern

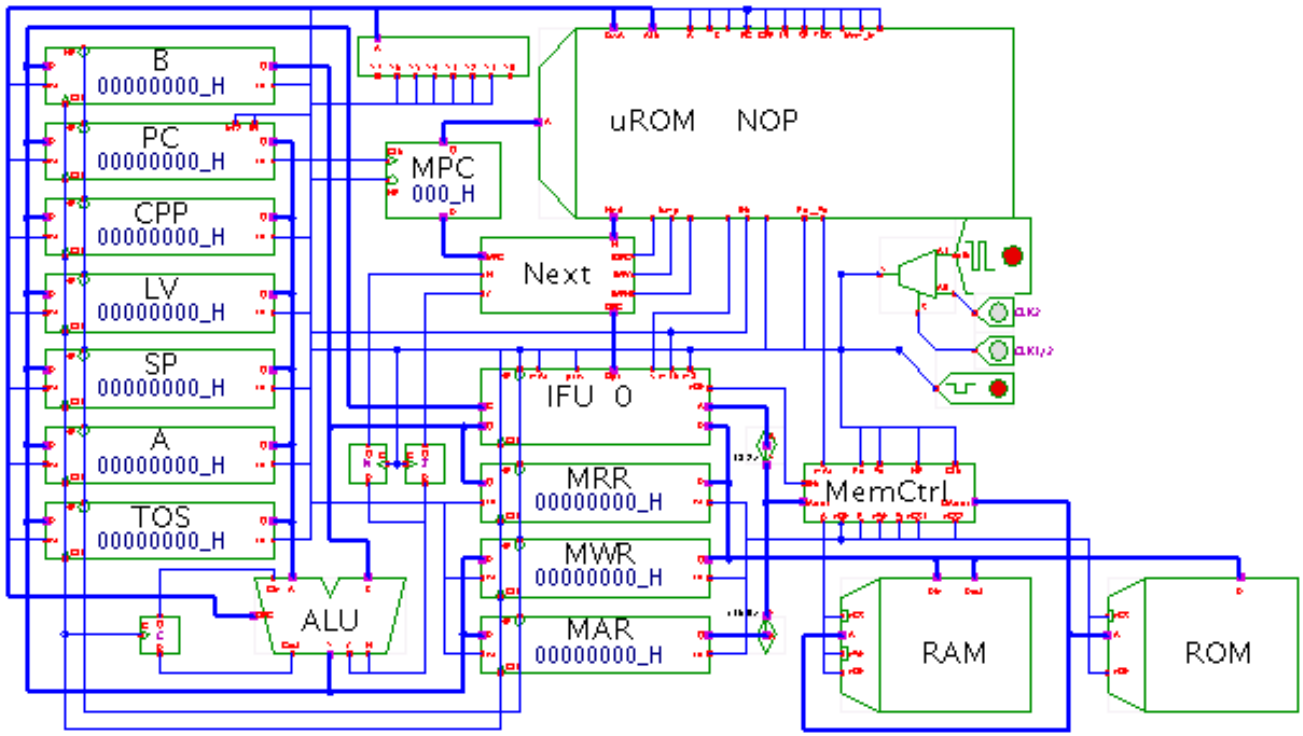


springen

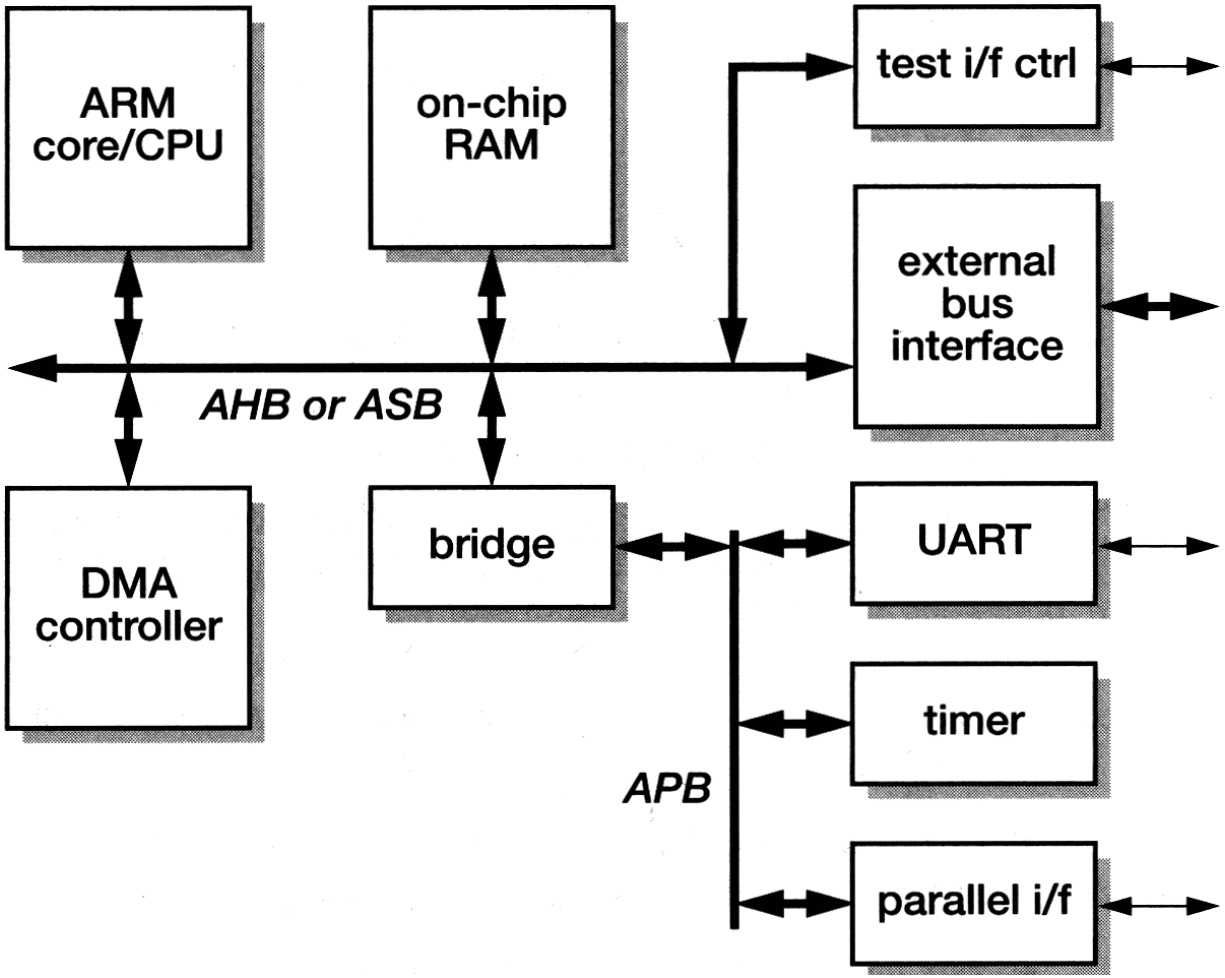
RS-Praktikum: D*CORE Prozessor



MicroJava Prozessor



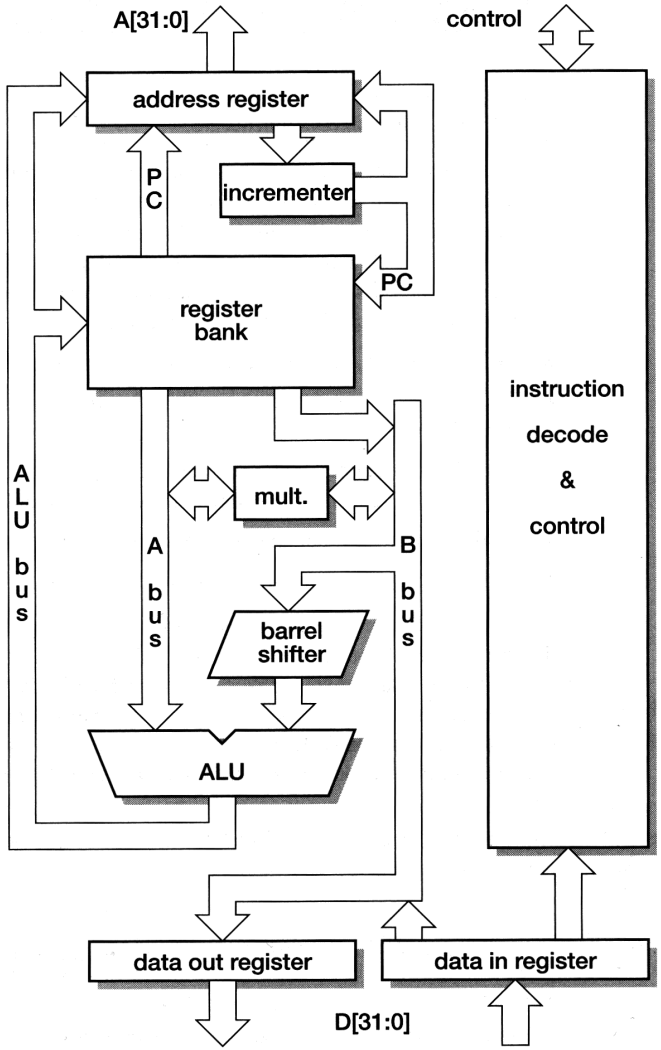
typisches ARM SoC System



S. Furber: *ARM System-on-Chip Architecture* [Fur00]

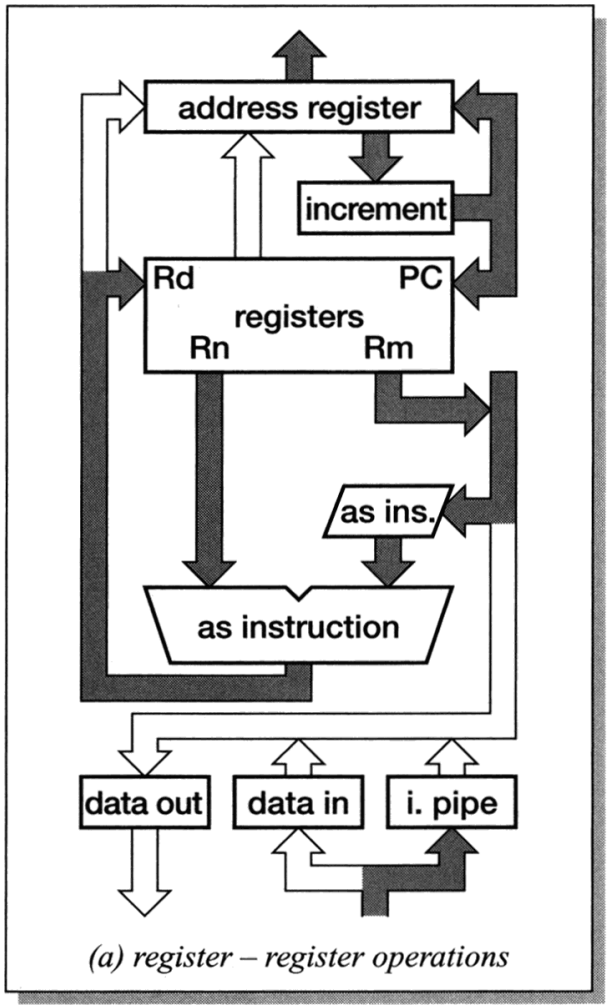
Kompletter Prozessor: ARM 3

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

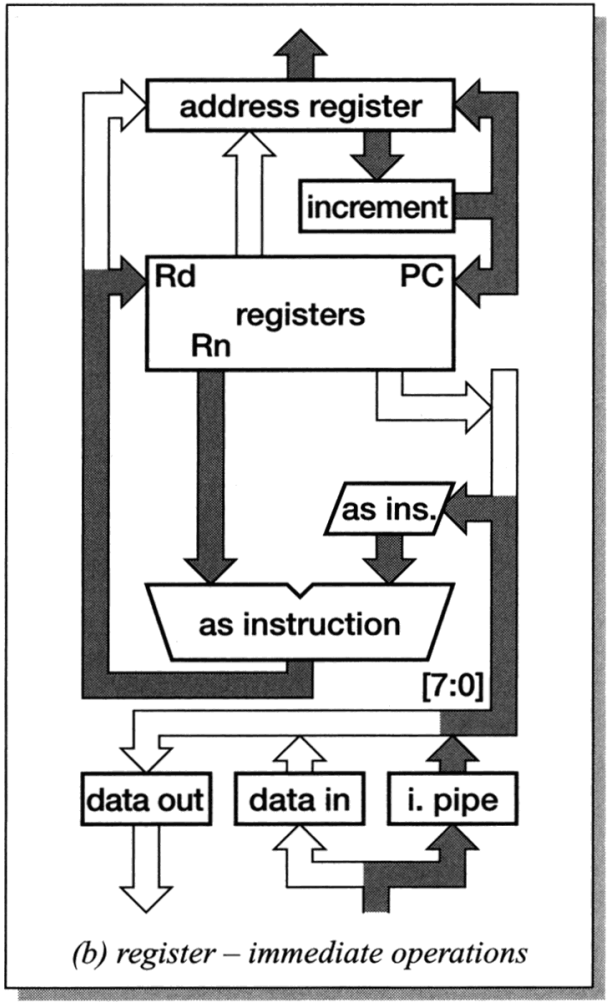


[Fur00]

ARM Datentransfer: Register-Operationen



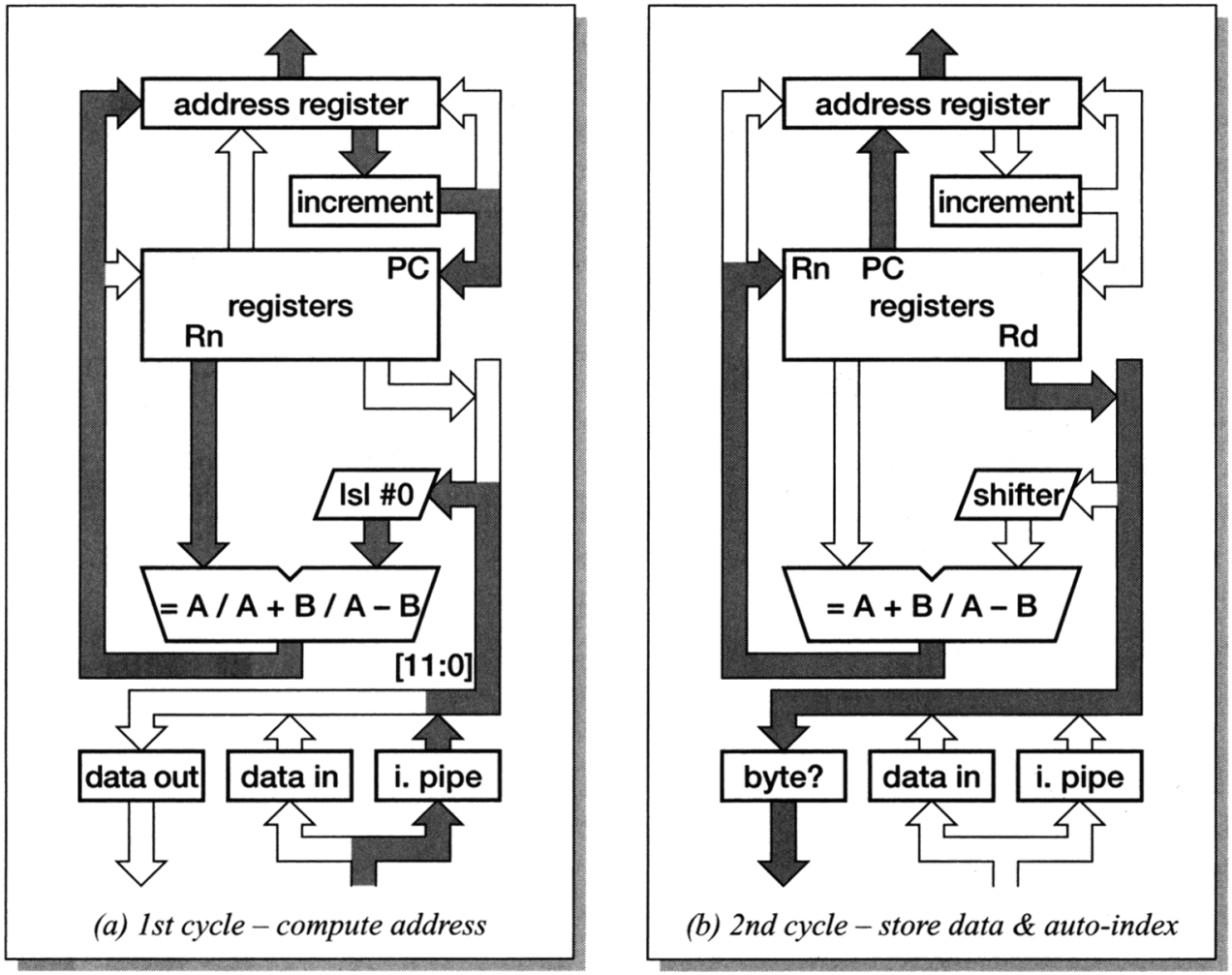
(a) register – register operations



(b) register – immediate operations

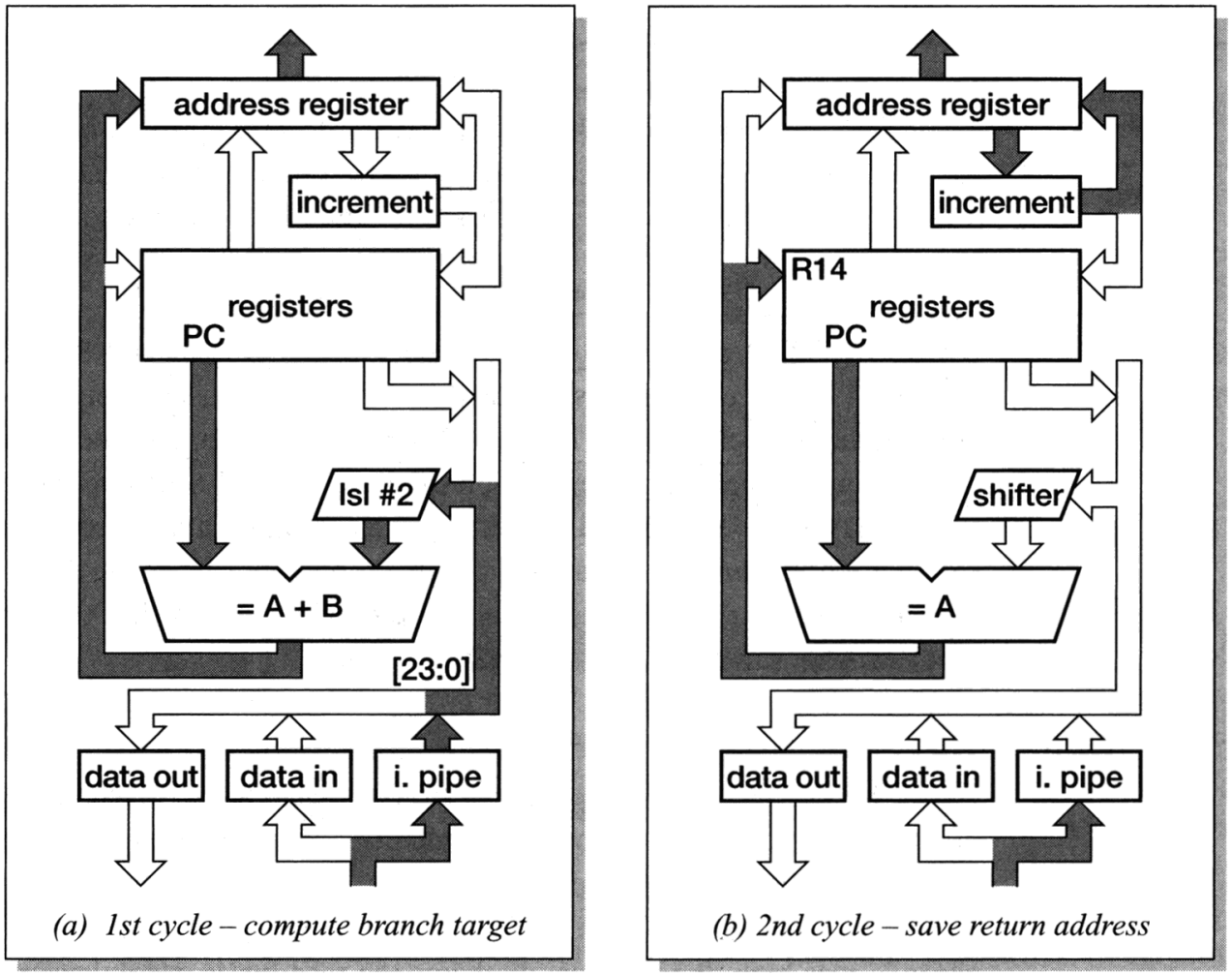
[Fur00]

ARM Datentransfer: Store-Befehl



[Fur00]

ARM Datentransfer: Funktionsaufruf/Sprungbefehl



[Fur00]

Literatur

[BO14] R.E. Bryant, D.R. O'Hallaron:
Computer systems – A programmers perspective.
2nd new intl. ed., Pearson Education Ltd., 2014.
ISBN 978–1–292–02584–1. `csapp.cs.cmu.edu`

[TA14] A.S. Tanenbaum, T. Austin: *Rechnerarchitektur – Von der digitalen Logik zum Parallelrechner*.
6. Auflage, Pearson Deutschland GmbH, 2014.
ISBN 978–3–86894–238–5

Literatur (cont.)

[PH14] D.A. Patterson, J.L. Hennessy: *Computer Organization and Design – The Hardware/Software Interface*. 5th edition, Morgan Kaufmann Publishers Inc., 2014. ISBN 978–0–12–407726–3

[PH11] D.A. Patterson, J.L. Hennessy: *Rechnerorganisation und -entwurf – Die Hardware/Software-Schnittstelle*. 4. Auflage, Oldenbourg, 2011. ISBN 978–3–486–59190–3

[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

Literatur (cont.)

[Fur00] S. Furber: *ARM System-on-Chip Architecture*. 2nd edition, Pearson Education Limited, 2000. ISBN 978–0–201–67519–1

[Mäd11] A. Mäder: *Vorlesung: Rechnerarchitektur und Mikrosystemtechnik*. Universität Hamburg, FB Informatik, 2011, Vorlesungsfolien. `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`