64-040 Modul IP7: Rechnerstrukturen

http://tams.informatik.uni-hamburg.de/lectures/2011ws/vorlesung/rs

Kapitel 15

Andreas Mäder

Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

Wintersemester 2011/2012

Kapitel 15

Grundkomponenten für Rechensysteme

Aufbau kompletter Rechensysteme

- bisher:
  - Gatter und Schaltnetze
  - Flipflops als einzelne Speicherglieder
  - Schaltwerke zur Ablaufsteuerung
- jetzt zusätzlich:
  - Speicher
  - Busse
  - Register-Transfer Komponenten eines Rechners
  - Ablaufsteuerung (Timing, Mikroprogrammierung)

Wiederholung: von-Neumann-Konzept

- J. Mauchly, J.P. Eckert, J. von-Neumann 1945
- System mit Prozessor, Speicher, Peripheriegeräten

- gemeinsamer Speicher für Programme und Daten
- Programme können wie Daten manipuliert werden
- Daten können als Programm ausgeführt werden

- Befehlszyklus: Befehl holen, dekodieren, ausführen
- enorm flexibel

- **alle** aktuellen Rechner basieren auf diesem Prinzip
- aber vielfältige Architekturvarianten, Befehlssätze, usw.

Wiederholung: von-Neumann Rechner

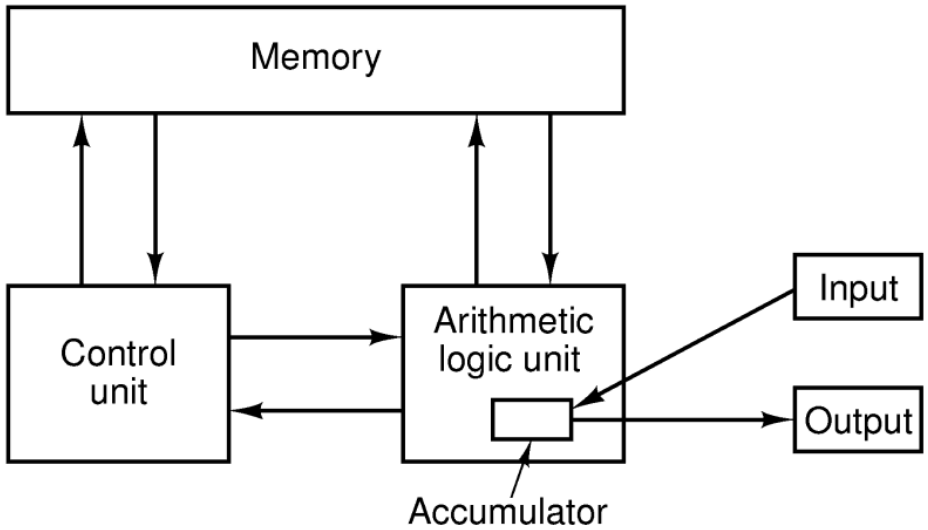


Figure 1-5. The original von Neumann machine.

Fünf zentrale Komponenten:

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

Wiederholung: von-Neumann Rechner (cont.)

- Steuerwerk: zwei zentrale Register
  - Befehlszähler (*program counter PC*)
  - Befehlsregister (*instruction register IR*)
- Operationswerk (Datenpfad, *data-path*)
  - Rechenwerk (*arithmetic-logic unit ALU*)
  - Universalregister (mindestens 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*)

Systemmodellierung

Modellierung eines digitalen Systems als Schaltung aus

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

Speicher

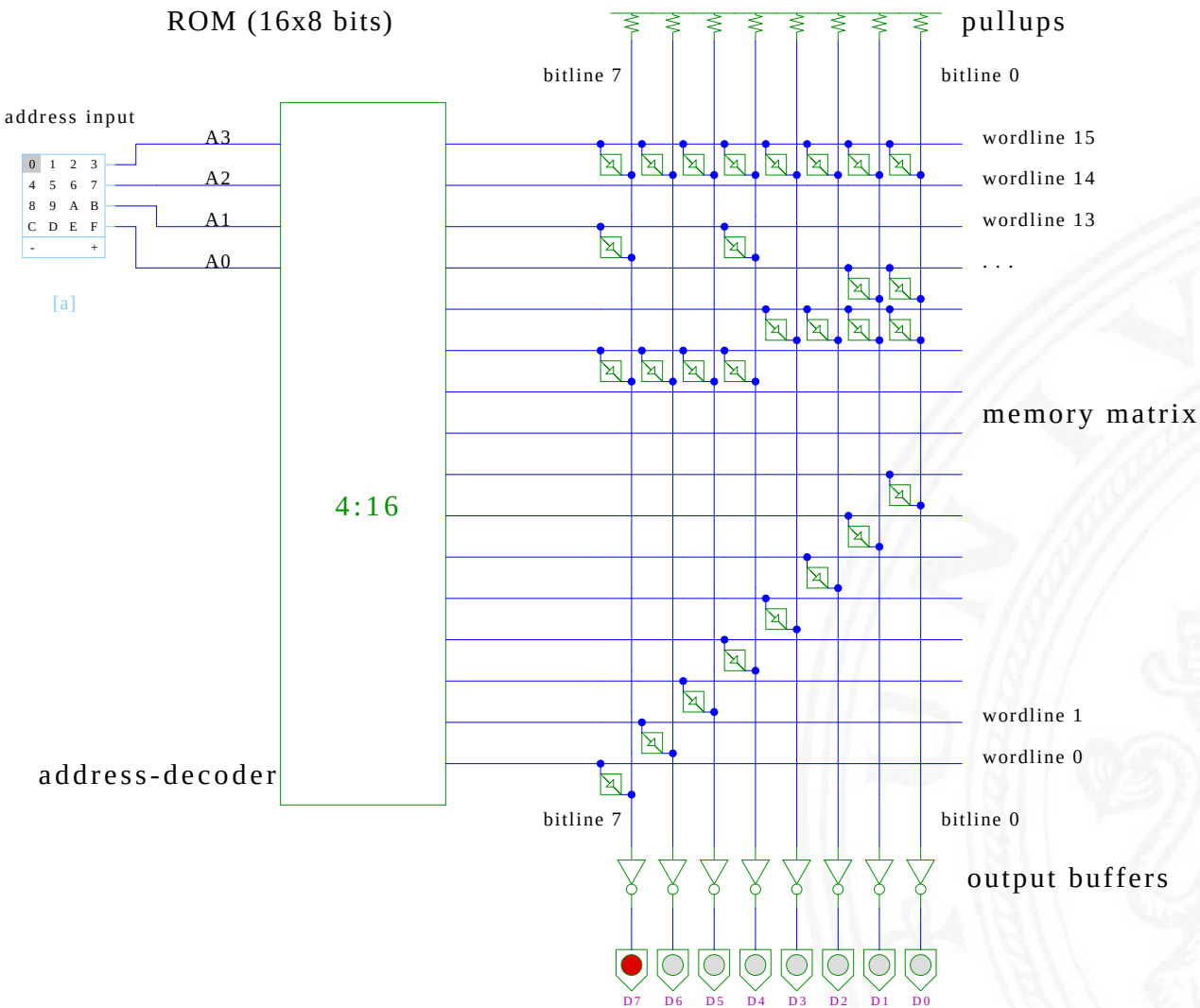
- System zur Speicherung von Information
- als Feld von N Adressen mit je m bit
- typischerweise mit n-bit Adressen und $N = 2^n$
- Kapazität also $2^n \times m$ bits

- 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

Type	Category	Erasure	Byte alterable	Volatile	Typical use
SRAM	Read/write	Electrical	Yes	Yes	Level 2 cache
DRAM	Read/write	Electrical	Yes	Yes	Main memory
ROM	Read-only	Not possible	No	No	Large volume appliances
PROM	Read-only	Not possible	No	No	Small volume equipment
EPROM	Read-mostly	UV light	No	No	Device prototyping
EEPROM	Read-mostly	Electrical	Yes	No	Device prototyping
Flash	Read/write	Electrical	No	No	Film for digital camera

ROM: Read-Only Memory



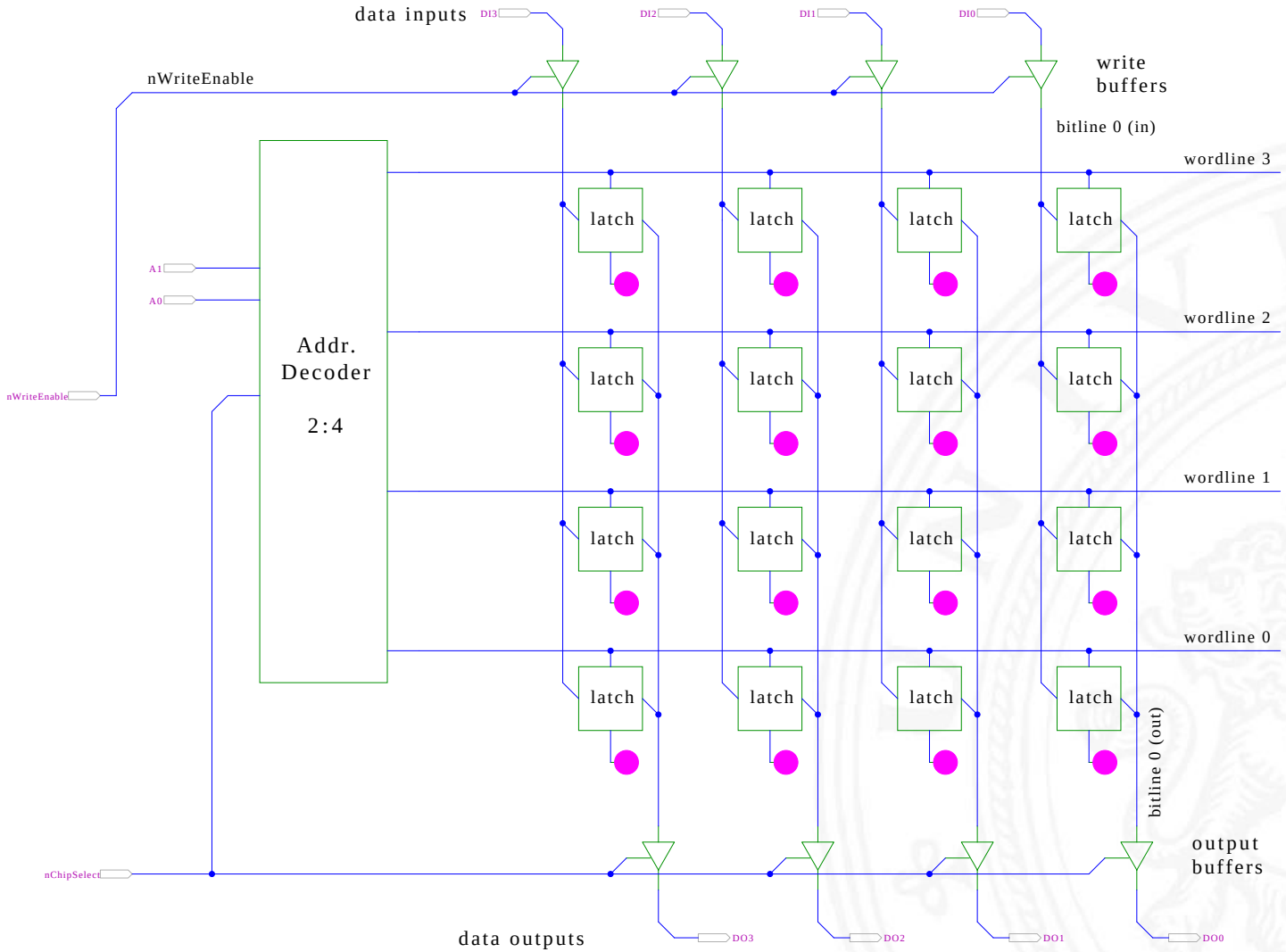
RAM: Random-Access Memory

Speicher, der im Betrieb gelesen und geschrieben werden kann

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

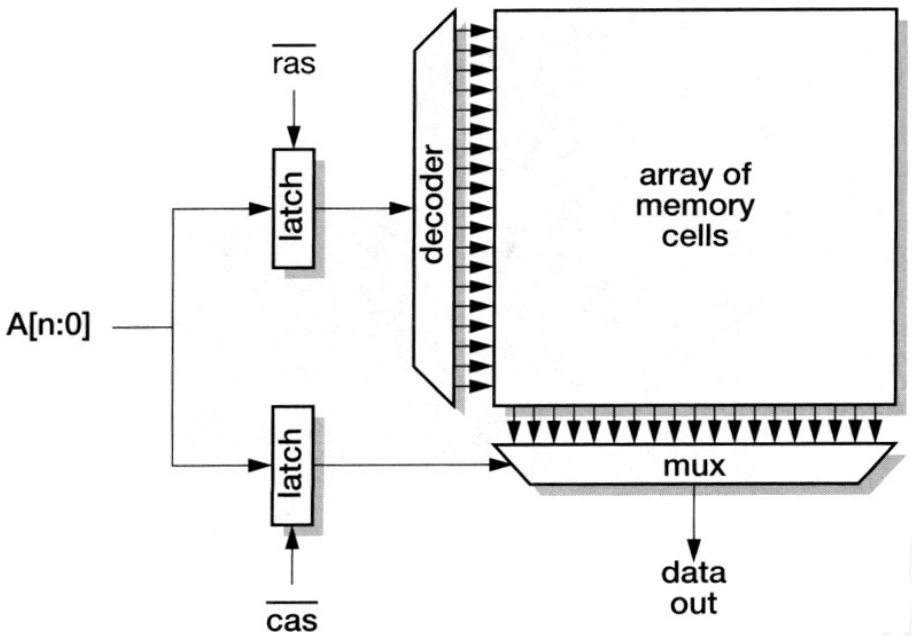
- Aufbau als Matrixstruktur
- n Adressbits, konzeptionell 2^n Wortleitungen
- m Bits pro Wort
- Realisierung der einzelnen Speicherstellen?
  - statisches RAM: 6-Transistor Zelle — SRAM
  - dynamisches RAM: 1-Transistor Zelle — DRAM

RAM: Blockschaltbild



4×4 bit
2-bit Adresse
4-bit Datenwort

RAM: RAS/CAS-Adressdecodierung

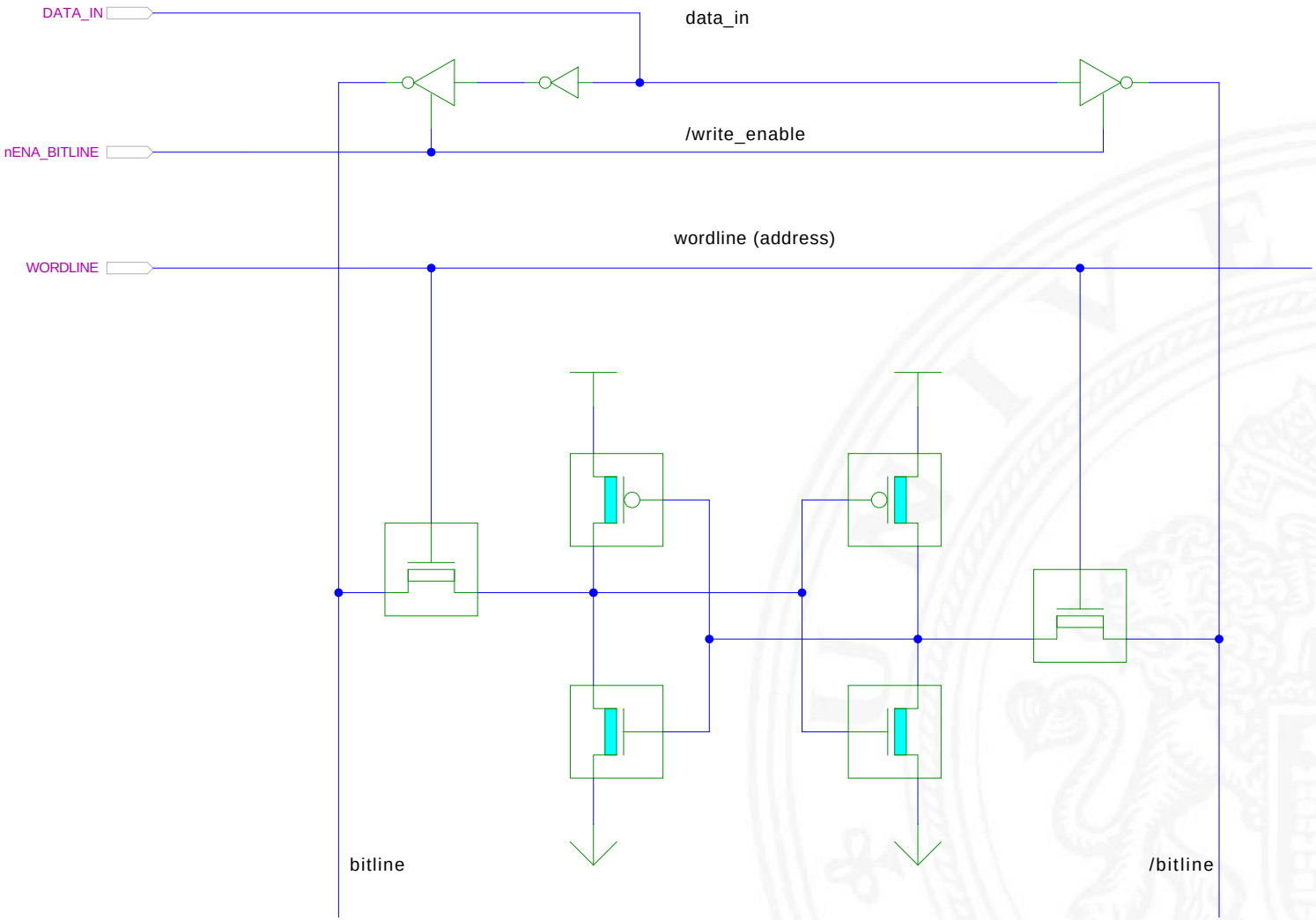


- Aufteilen der Adresse in zwei Hälften
- $\overline{ras}$ „row address strobe“ wählt „Wordline“
  $\overline{cas}$ „column address strobe“ –"– „Bitline“
- je ein $2^{(n/2)}$-bit Decoder/Mux statt ein 2^n-bit Decoder

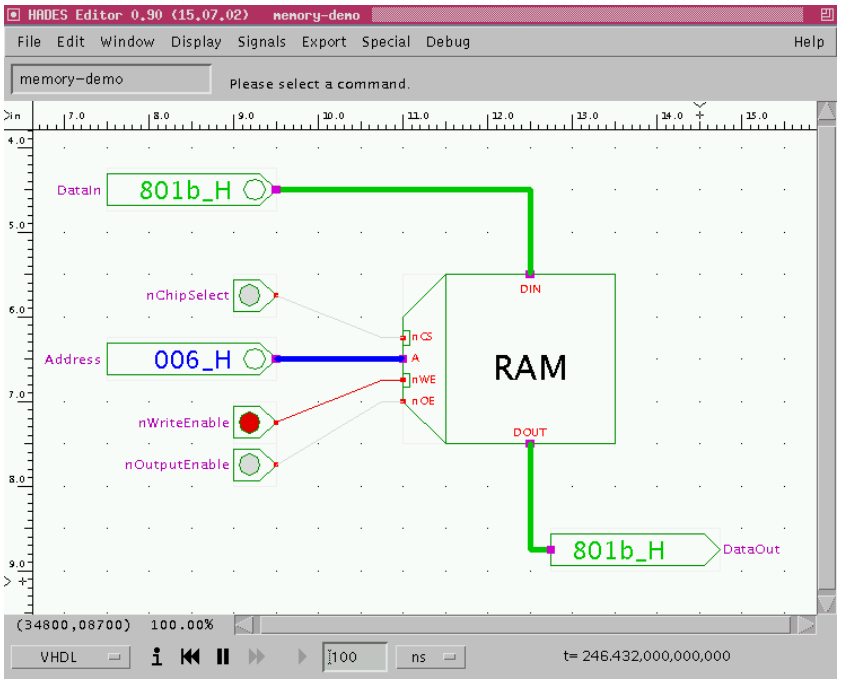
SRAM: statisches RAM

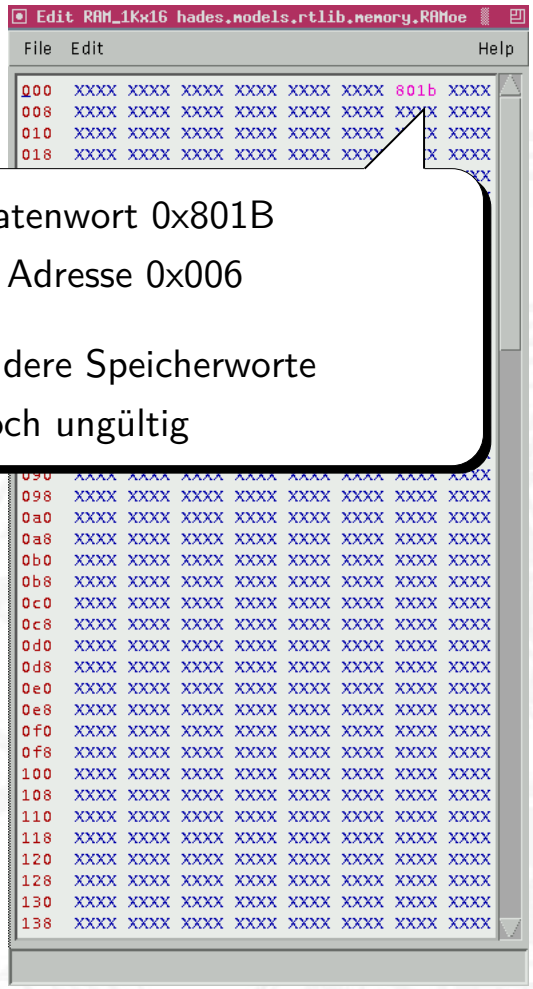
- Inhalt bleibt dauerhaft gespeichert solange Betriebsspannung anliegt
- *sechs-Transistor* Zelle zur Speicherung
  - weniger Platzverbrauch als Latches/Flipflops
  - kompakte Realisierung in CMOS-Technologie (s.u.)
  - zwei rückgekoppelte Inverter zur Speicherung
  - zwei n-Kanal Transistoren zur Anbindung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

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



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)

`tams.informatik.uni-hamburg.de/applets/hades/webdemos/50-rtlib/40-memory/ram.html`

SRAM: Beispiel IC 6116

- integrierte Schaltung, 16 Kbit Kapazität
- Organisation als 2K 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

`tams.informatik.uni-hamburg.de/applets/hades/webdemos/40-memories/40-ram`

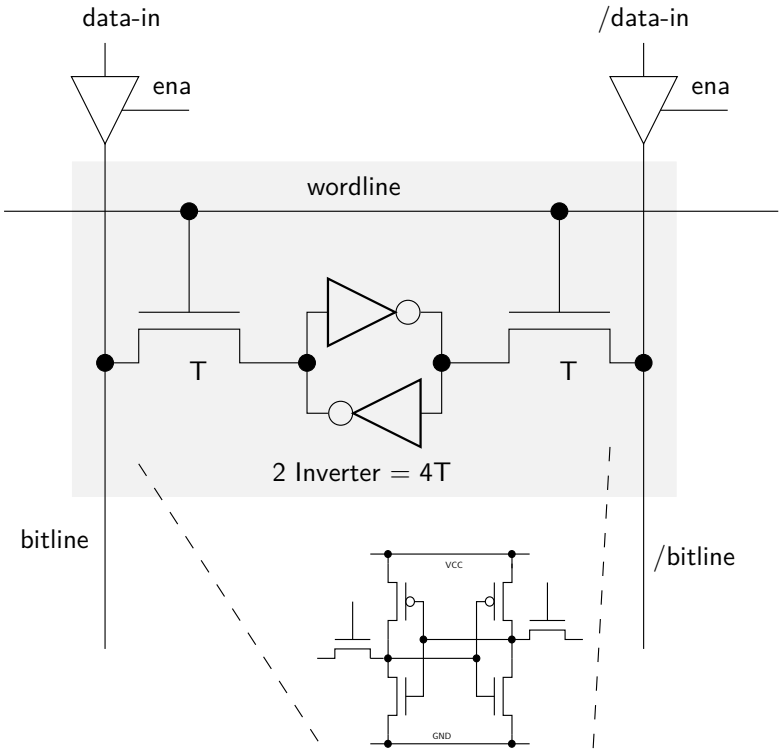
DRAM: dynamisches RAM

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

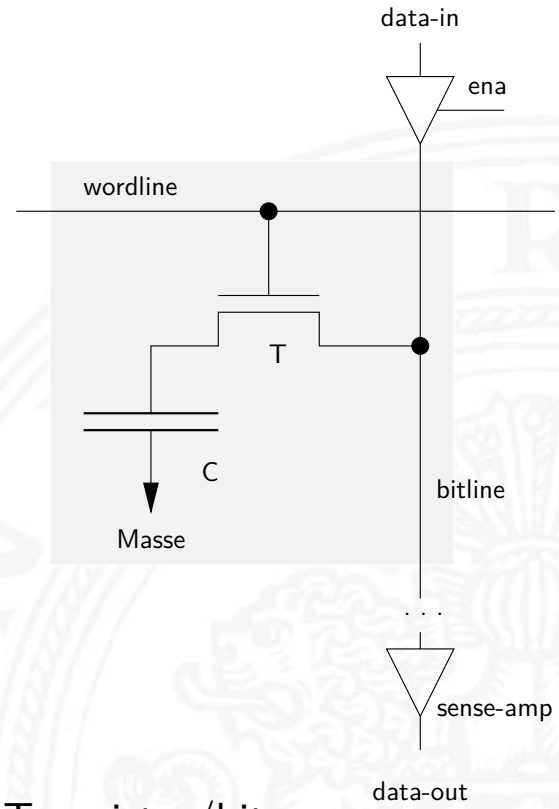
- jeder Lesezugriff entlädt den Kondensator
- *Leseverstärker* zur Messung der Spannung auf der Bitline
  Schwellwertvergleich zur Entscheidung logisch 0/1

– 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 .. 50X DRAM-Fläche



- 1 Transistor/bit
- C=10fF: ~200.000 Elektronen
- langsam (sense-amp)
- minimale Fläche

DRAM: Stacked- und Trench-Zelle

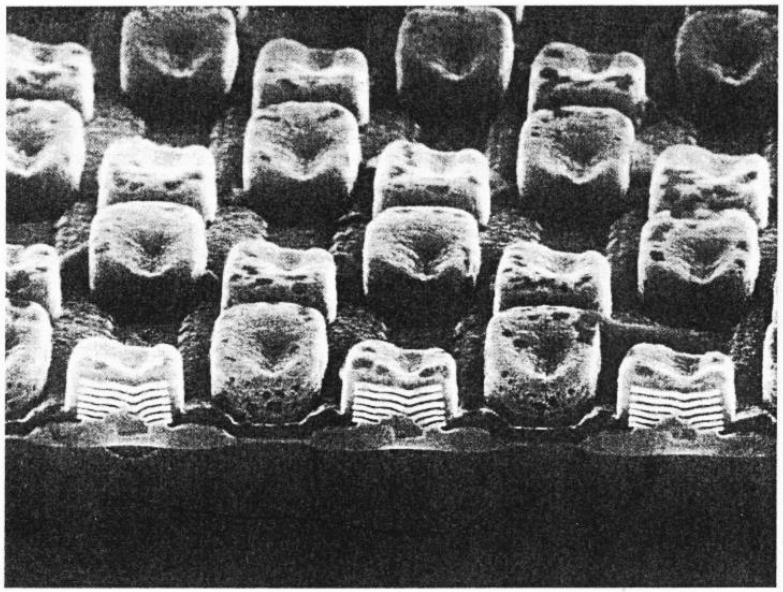
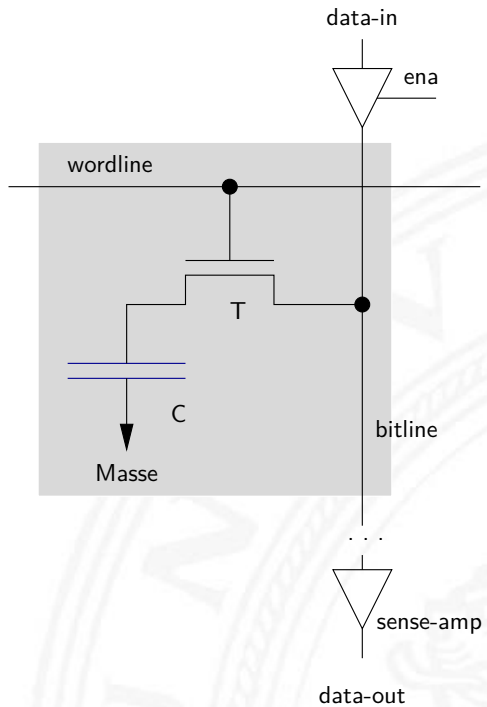
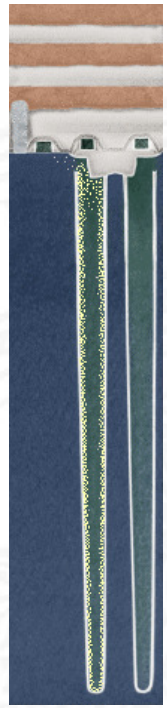


Abb. 7: Prototyp von Speicherzellen (Stapelkondensatoren) für zukünftige Speicherchips wie den Ein-Gigabit-Chip. Da für DRAM-Chips eine minimale Speicherkapazität von 25 fF notwendig ist, bringt es erhebliche Platzvorteile, die Kondensatorelemente vertikal übereinander zu stapeln. Die Dicke der Schichten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

Siemens 1 Gbit DRAM

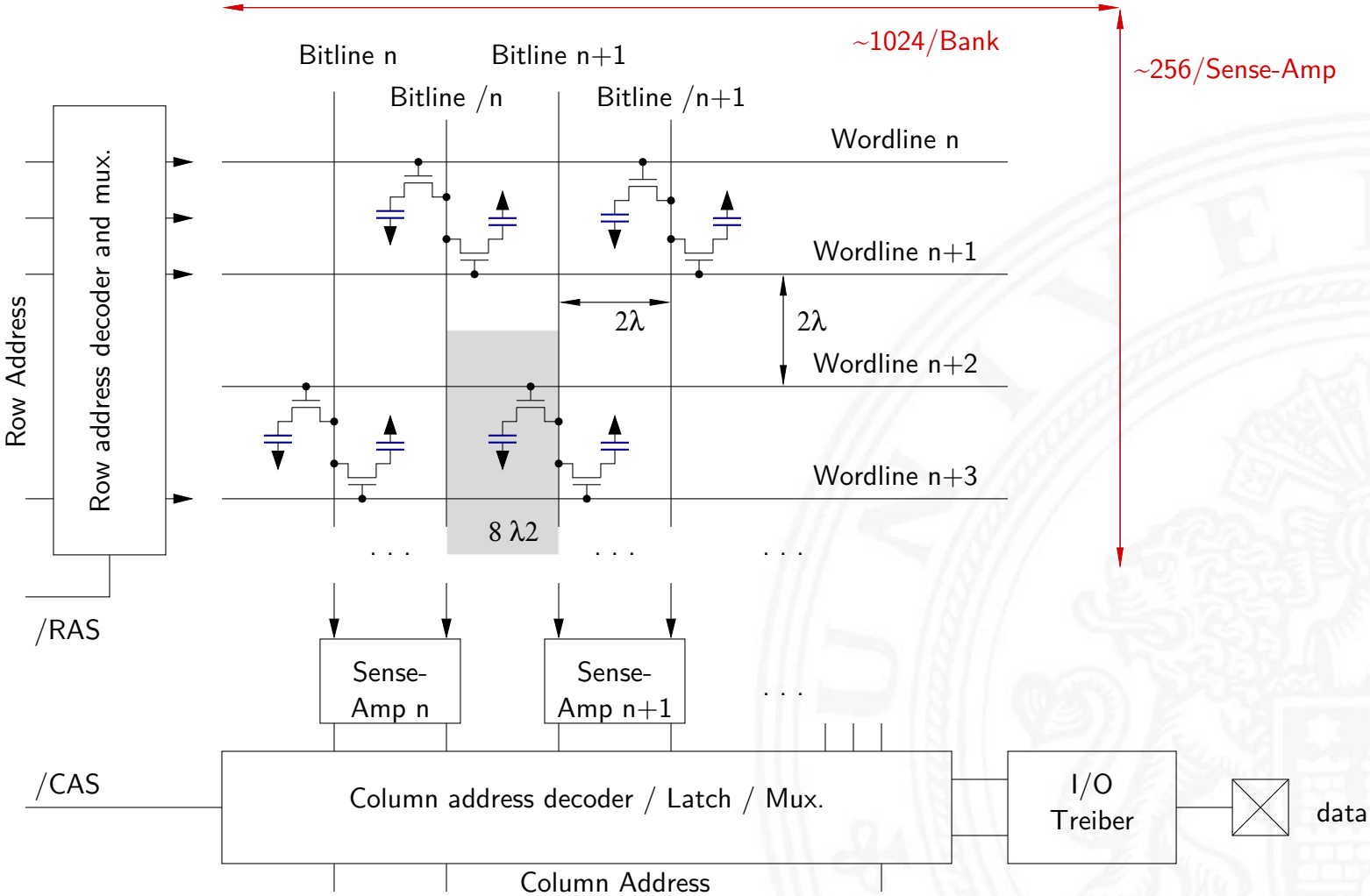




IBM CMOS-6X embedded DRAM

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

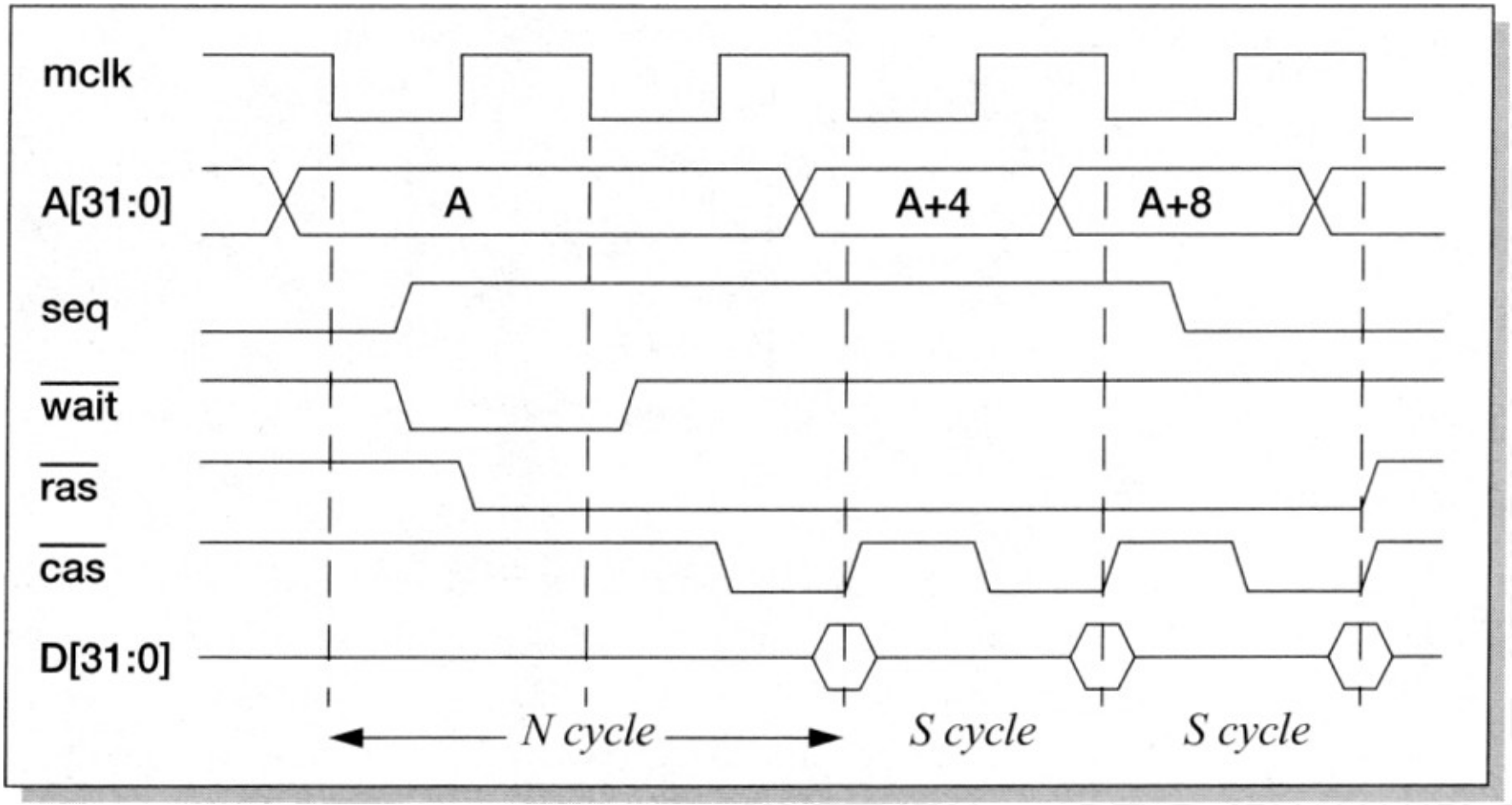
DRAM: Layout


~1024/Bank
~256/Sense-Amp
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.
Column Address
I/O
Treiber
data

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
  - DDR-2, DDR-3: Varianten mit höherer Taktrate
    aktuell Übertragungsraten bis 17 GByte/sec

SDRAM: Lesezugriff auf sequenzielle Adressen



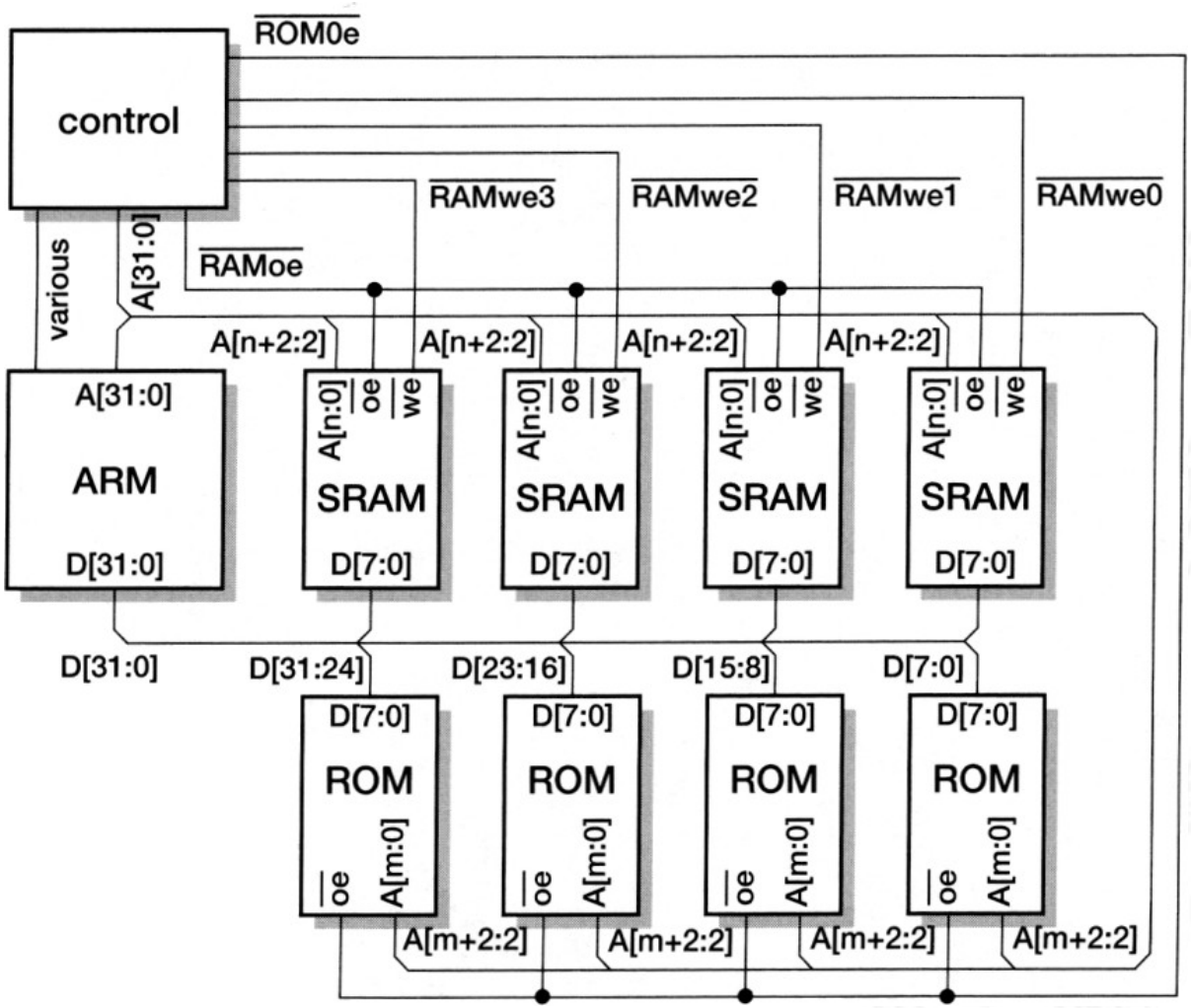
Flash

- 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

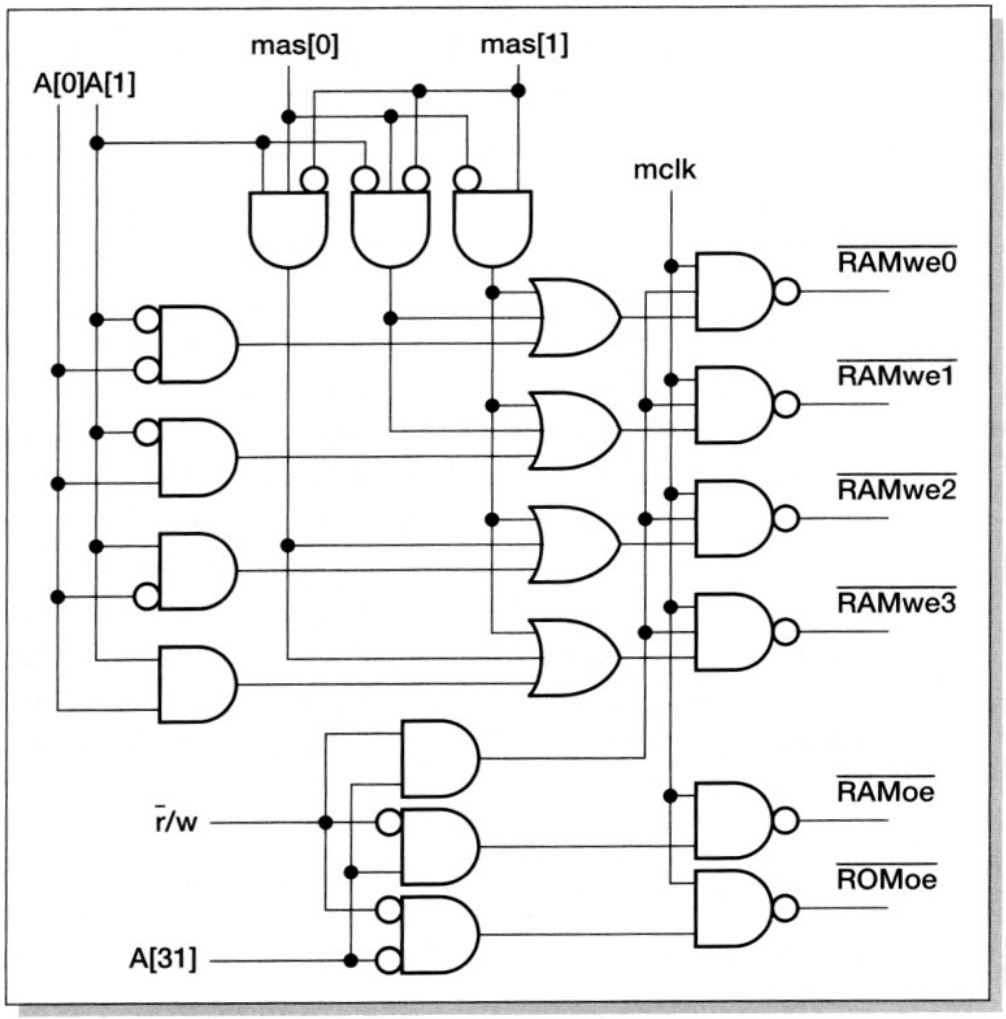
- Auslesen beliebig oft möglich, schnell
- Schreibzugriffe problematisch
  - intern hohe Spannung erforderlich (Gate-Isolierung überwinden)
  - Schreibzugriffe einer „0“ nur blockweise
  - pro Zelle nur einige 10 000. . . 100 000 Schreibzugriffe möglich

Typisches Speichersystem



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

Typisches Speichersystem: Adressdecodierung

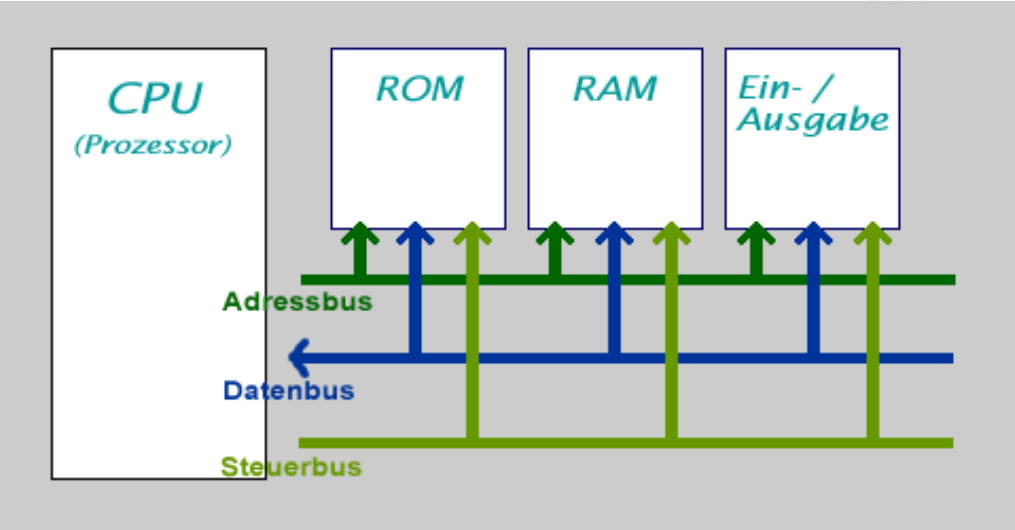


Bussysteme

- **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)

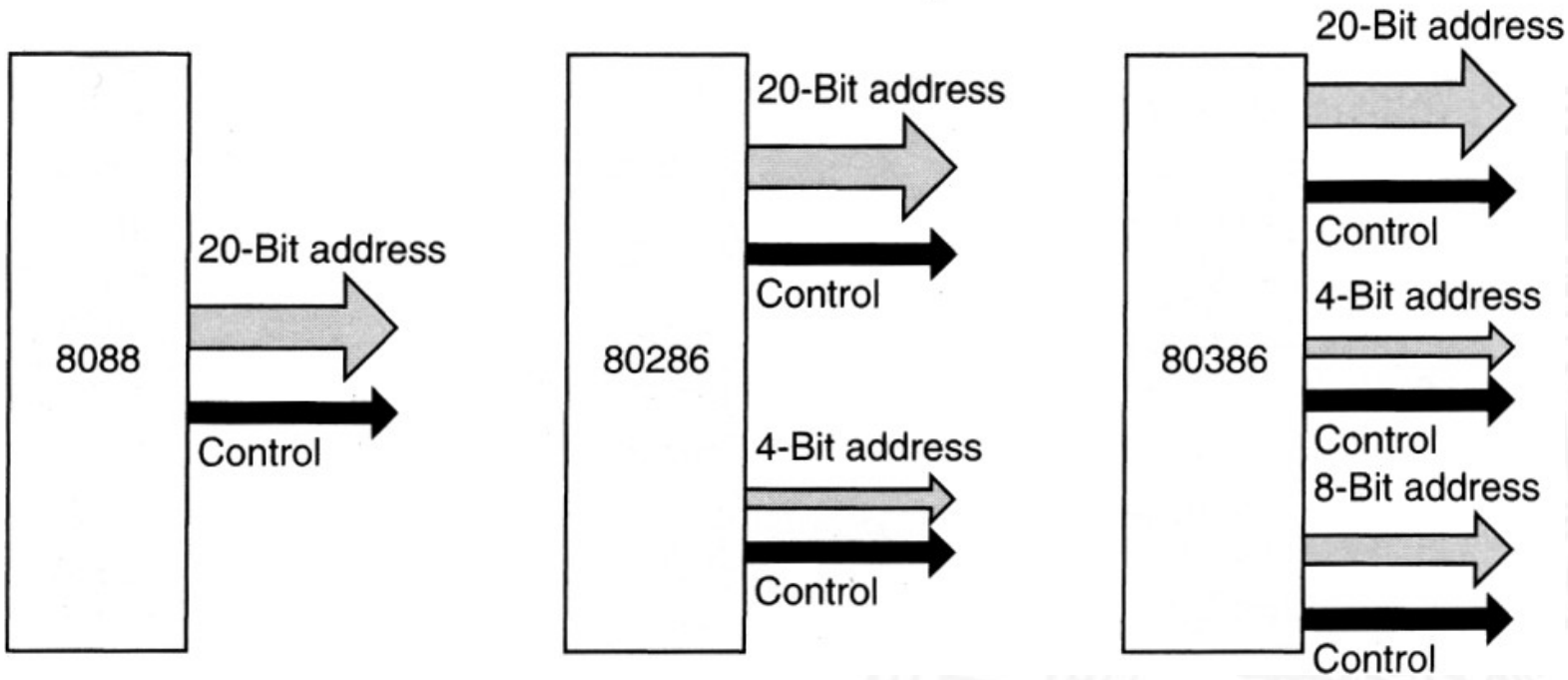
Bus: Mikroprozessorsysteme

typisches n-bit Mikroprozessor-System:

- n Adress-Leitungen, also Adressraum 2^n Bytes — Adressbus
- n Daten-Leitungen — Datenbus

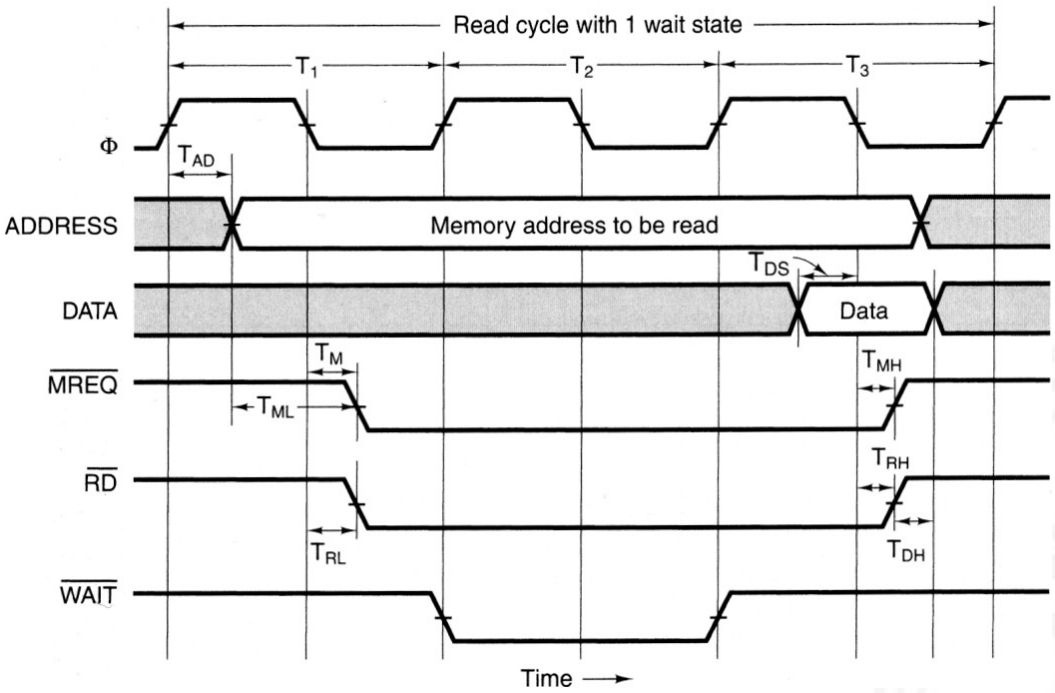
- Steuersignale — Control
  - clock: Taktsignal
  - read/write: Lese-/Schreibzugriff (aus Sicht des Prozessors)
  - wait: Wartezeit/-zyklen für langsame Geräte
  - ...
- um Leitungen zu sparen, teilweise gemeinsam genutzte Leitungen sowohl für Adressen als auch Daten.
  Zusätzliches Steuersignal zur Auswahl Adressen/Daten

Adressbus: Evolution beim Intel x86



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

Synchroner Bus: Timing



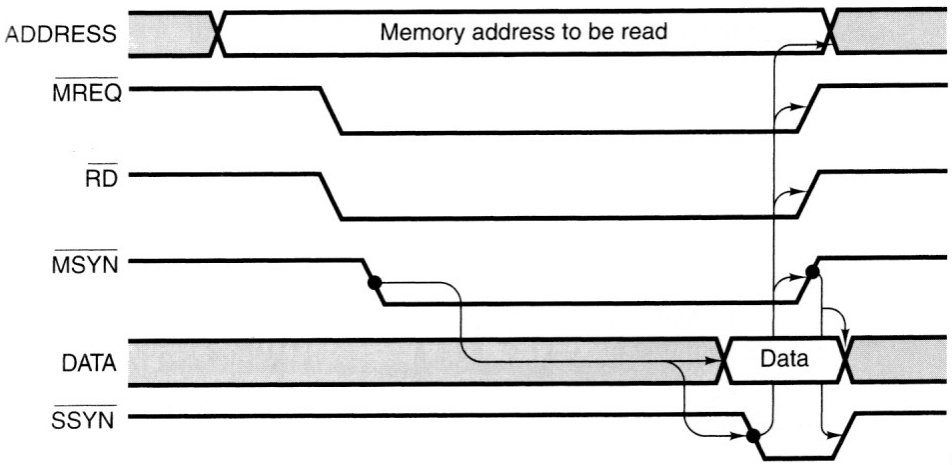
A.S. Tanenbaum, *Structured Computer Organization* 3.4.4

- 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: typische Zeit-Parameter

Symbol	Parameter	Min	Max	Unit
T_{AD}	Address output delay		4	nsec
T_{ML}	Address stable prior to $\overline{MREQ}$	2		nsec
T_M	$\overline{MREQ}$ delay from falling edge of Φ in T_1		3	nsec
T_{RL}	RD delay from falling edge of Φ in T_1		3	nsec
T_{DS}	Data setup time prior to falling edge of Φ	2		nsec
T_{MH}	$\overline{MREQ}$ delay from falling edge of Φ in T_3		3	nsec
T_{RH}	$\overline{RD}$ delay from falling edge of Φ in T_3		3	nsec
T_{DH}	Data hold time from negation of $\overline{RD}$	0		nsec

Asynchroner Bus: Lesezugriff



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

Bus Arbitrierung

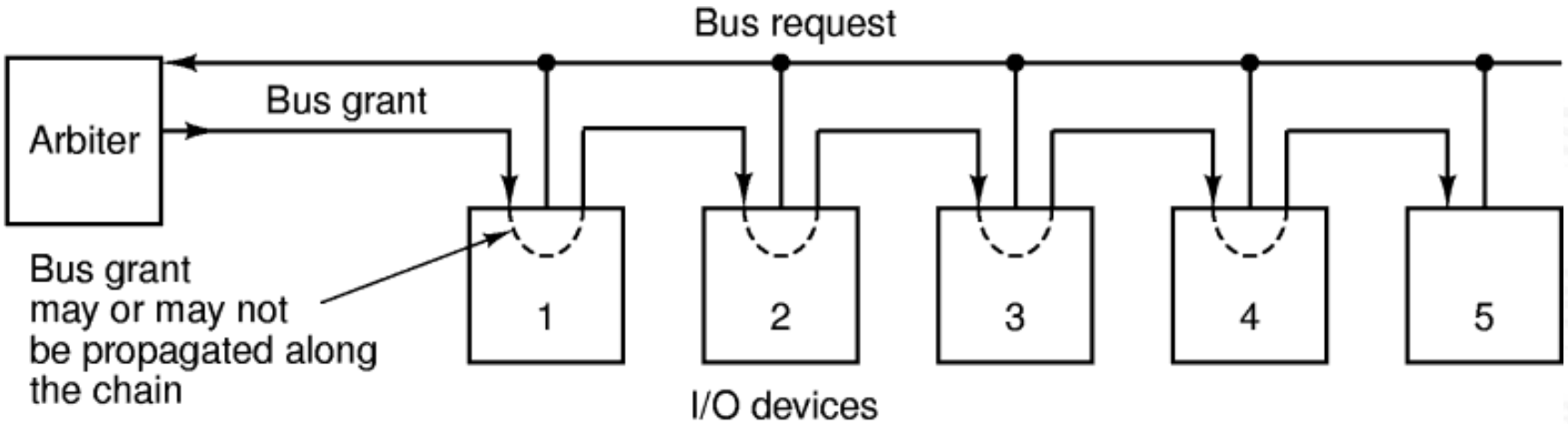
- mehrere Komponenten wollen Übertragung initiieren
  immer nur ein Transfer zur Zeit möglich
- der Zugriff muss serialisiert werden

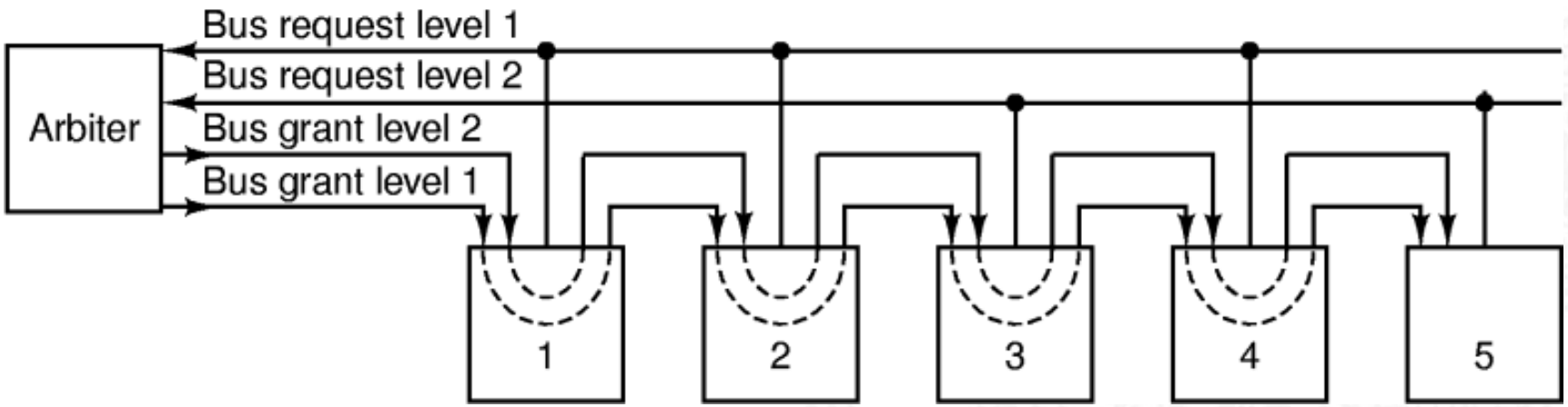
1. zentrale Arbitrierung
   - Arbiter gewährt Bus-Requests
   - Strategien
     - Prioritäten für verschiedene Geräte
     - „round-robin“ Verfahren
     - „Token“-basierte Verfahren
     - usw.

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.)





Bus Bandbreite

- Menge an (Nutz-) Daten, die pro Zeiteinheit übertragen werden kann
- zusätzlicher Protokolloverhead ⇒ Brutto- / Netto-Datenrate
- RS232 50 Bit/sec ... 460 KBit/sec
  I²C 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)

Beispiel: PCI-Bus

Peripherial Component Interconnect (Intel 1991)

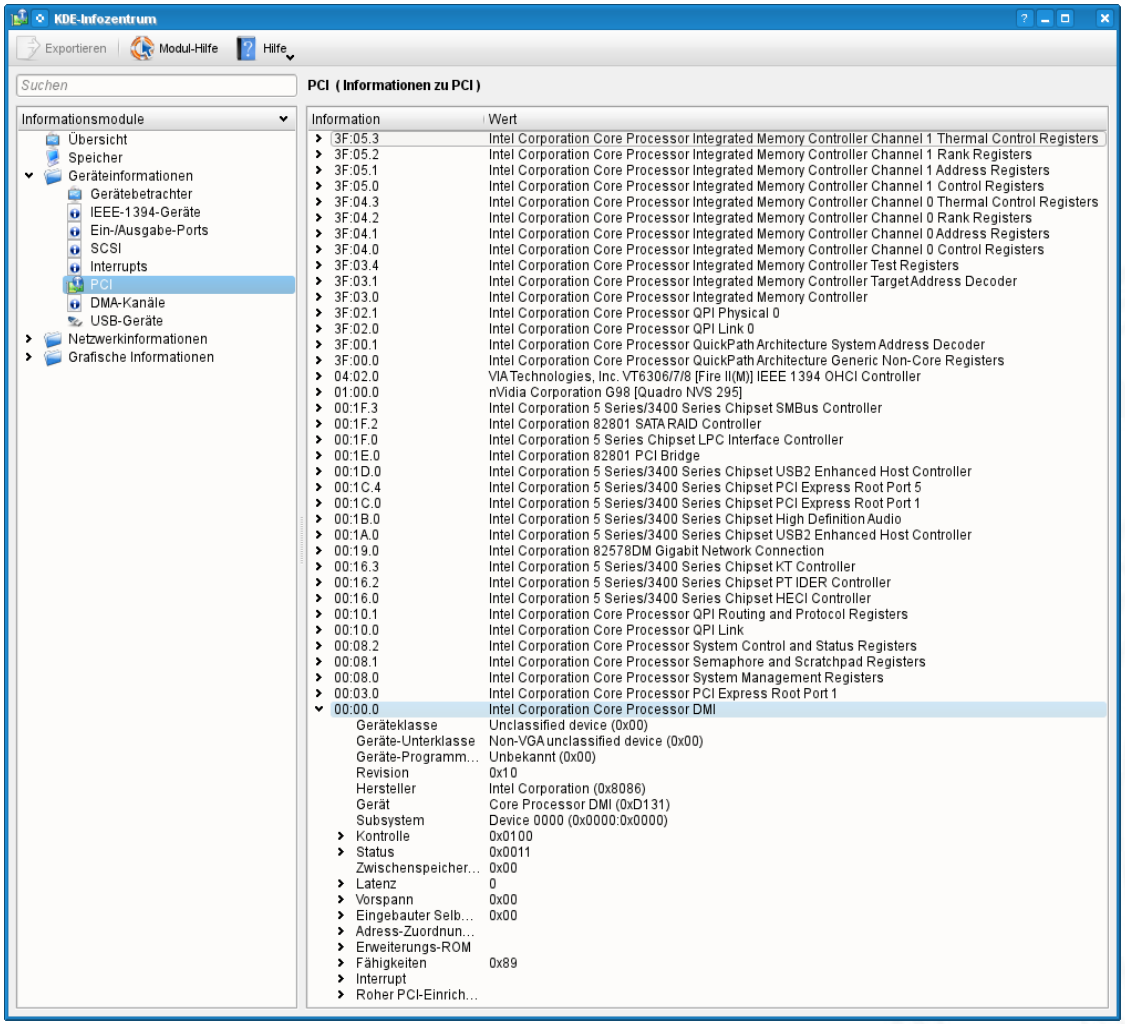
- 33 MHz Takt — optional 64 MHz Takt
- 32-bit Bus-System — optional auch 64-bit
- gemeinsame Adress-/Datenleitungen
- Arbitrierung durch Bus-Master — CPU

- 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

```
tams12> /sbin/lspci
00:00.0 Host bridge: Intel Corporation 82Q963/Q965 Memory Controller Hub (rev 02)
00:01.0 PCI bridge: Intel Corporation 82Q963/Q965 PCI Express Root Port (rev 02)
00:1a.0 USB Controller: Intel Corporation 82801H (ICH8 Family) USB UHCI #4 (rev 02)
00:1a.1 USB Controller: Intel Corporation 82801H (ICH8 Family) USB UHCI #5 (rev 02)
00:1a.7 USB Controller: Intel Corporation 82801H (ICH8 Family) USB2 EHCI #2 (rev 02)
00:1b.0 Audio device: Intel Corporation 82801H (ICH8 Family) HD Audio Controller (rev 02)
00:1c.0 PCI bridge: Intel Corporation 82801H (ICH8 Family) PCI Express Port 1 (rev 02)
00:1c.4 PCI bridge: Intel Corporation 82801H (ICH8 Family) PCI Express Port 5 (rev 02)
00:1d.0 USB Controller: Intel Corporation 82801H (ICH8 Family) USB UHCI #1 (rev 02)
00:1d.1 USB Controller: Intel Corporation 82801H (ICH8 Family) USB UHCI #2 (rev 02)
00:1d.2 USB Controller: Intel Corporation 82801H (ICH8 Family) USB UHCI #3 (rev 02)
00:1d.7 USB Controller: Intel Corporation 82801H (ICH8 Family) USB2 EHCI #1 (rev 02)
00:1e.0 PCI bridge: Intel Corporation 82801 PCI Bridge (rev f2)
00:1f.0 ISA bridge: Intel Corporation 82801HB/HR (ICH8/R) LPC Interface Controller (rev 02)
00:1f.2 IDE interface: Intel Corporation 82801H (ICH8 Family) 4 port SATA IDE Controller (rev 02)
00:1f.3 SMBus: Intel Corporation 82801H (ICH8 Family) SMBus Controller (rev 02)
00:1f.5 IDE interface: Intel Corporation 82801H (ICH8 Family) 2 port SATA IDE Controller (rev 02)
01:00.0 VGA compatible controller: ATI Technologies Inc Unknown device 7183
01:00.1 Display controller: ATI Technologies Inc Unknown device 71a3
03:00.0 Ethernet controller: Broadcom Corporation NetXtreme BCM5754 Gigabit Ethernet PCI Express (rev 02)
```

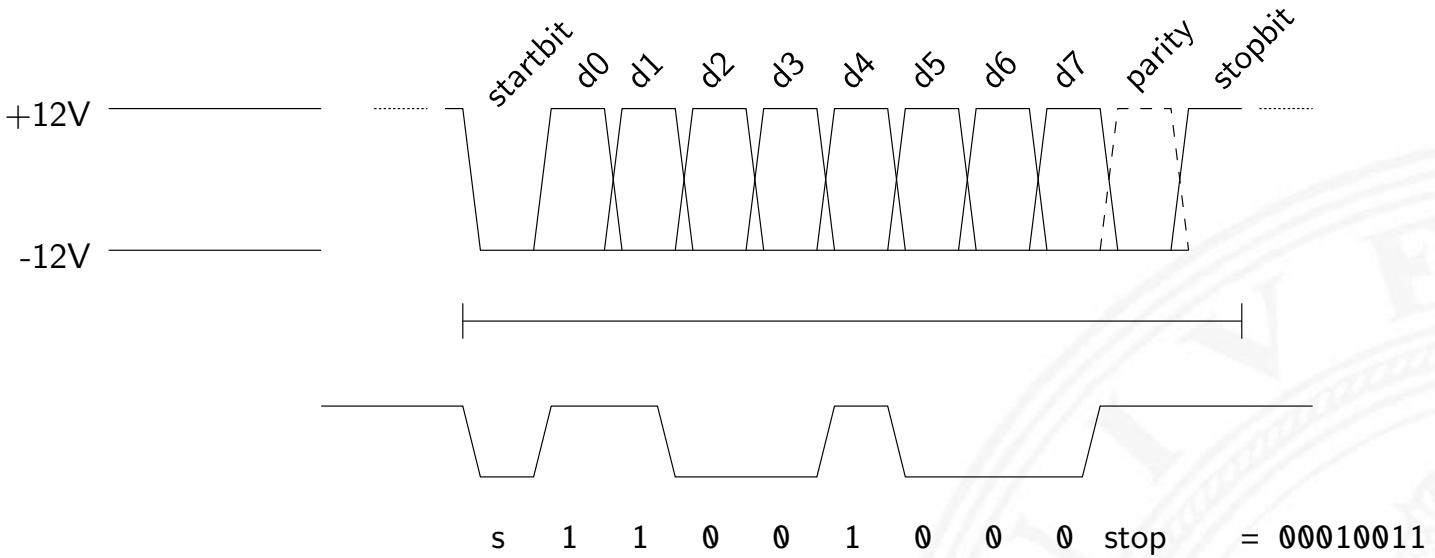
PCI-Bus: Peripheriegeräte (cont.)



PCI-Bus: Leitungen („mandatory“)

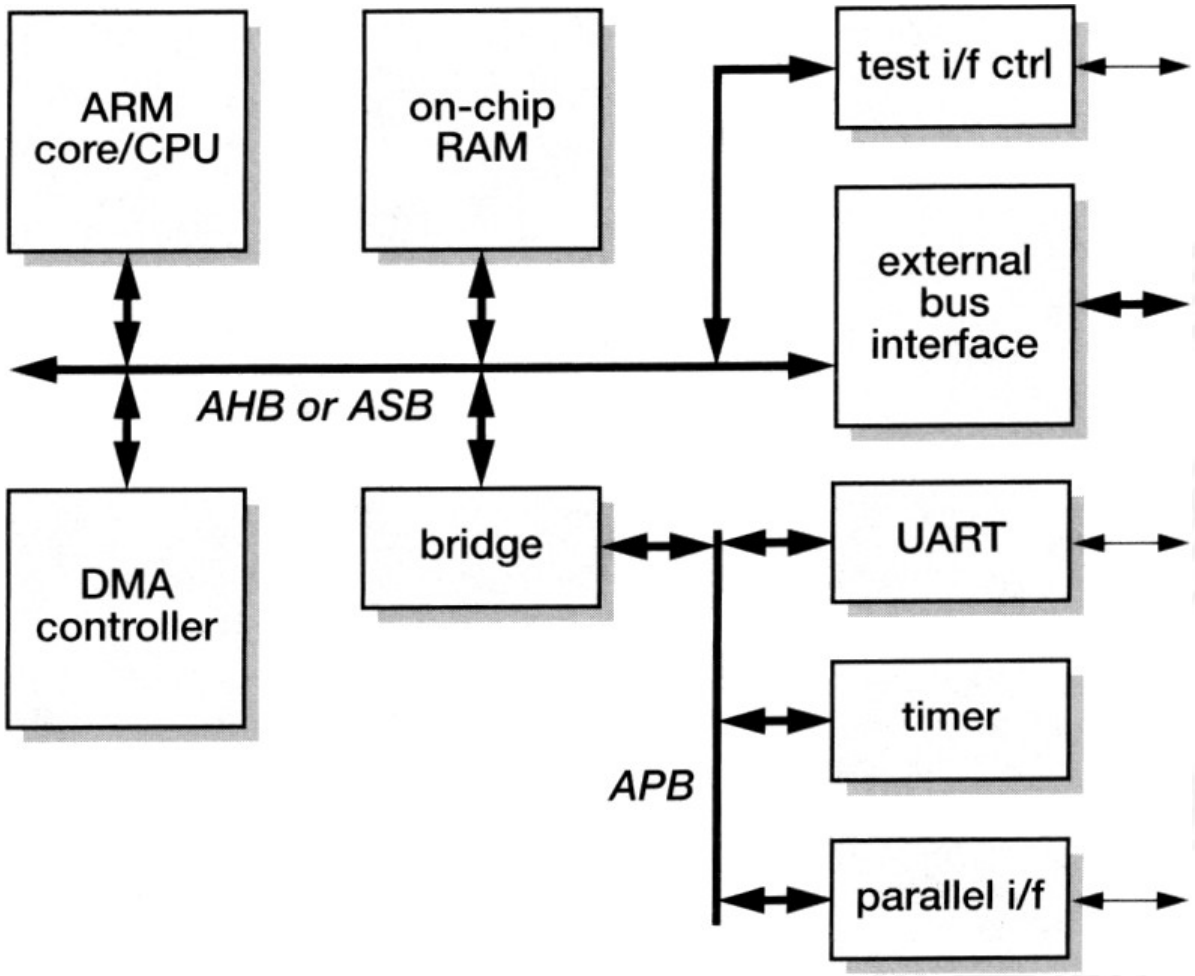
Signal	Lines	Master	Slave	Description
CLK	1			Clock (33 MHz or 66 MHz)
AD	32	×	×	Multiplexed address and data lines
PAR	1	×		Address or data parity bit
C/BE	4	×		Bus command/bit map for bytes enabled
FRAME#	1	×		Indicates that AD and C/BE are asserted
IRDY#	1	×		Read: master will accept; write: data present
IDSEL	1	×		Select configuration space instead of memory
DEVSEL#	1		×	Slave has decoded its address and is listening
TRDY#	1		×	Read: data present; write: slave will accept
STOP#	1		×	Slave wants to stop transaction immediately
PERR#	1			Data parity error detected by receiver
SERR#	1			Address parity error or system error detected
REQ#	1			Bus arbitration: request for bus ownership
GNT#	1			Bus arbitration: grant of bus ownership
RST#	1			Reset the system and all devices

RS-232: Serielle Schnittstelle



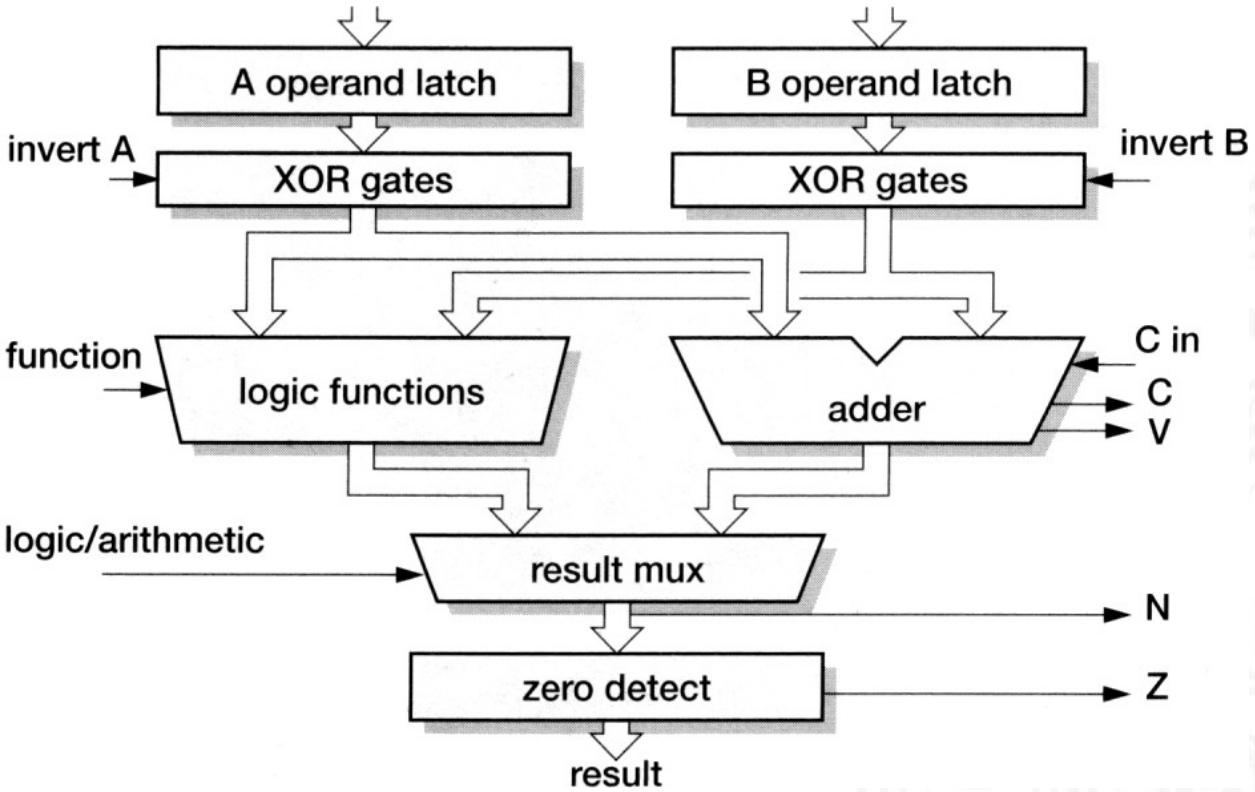
- Baudrate 300, 600, . . . , 19200, 38400, 115200 bits/sec
  Anzahl Datenbits 5, 6, 7, 8
  Anzahl Stopbits 1, 2
  Parität none, odd, even
- minimal drei Leitungen: GND, TX, RX (Masse, Transmit, Receive)
- oft weitere Leitungen für erweitertes Handshake

typisches ARM SoC System



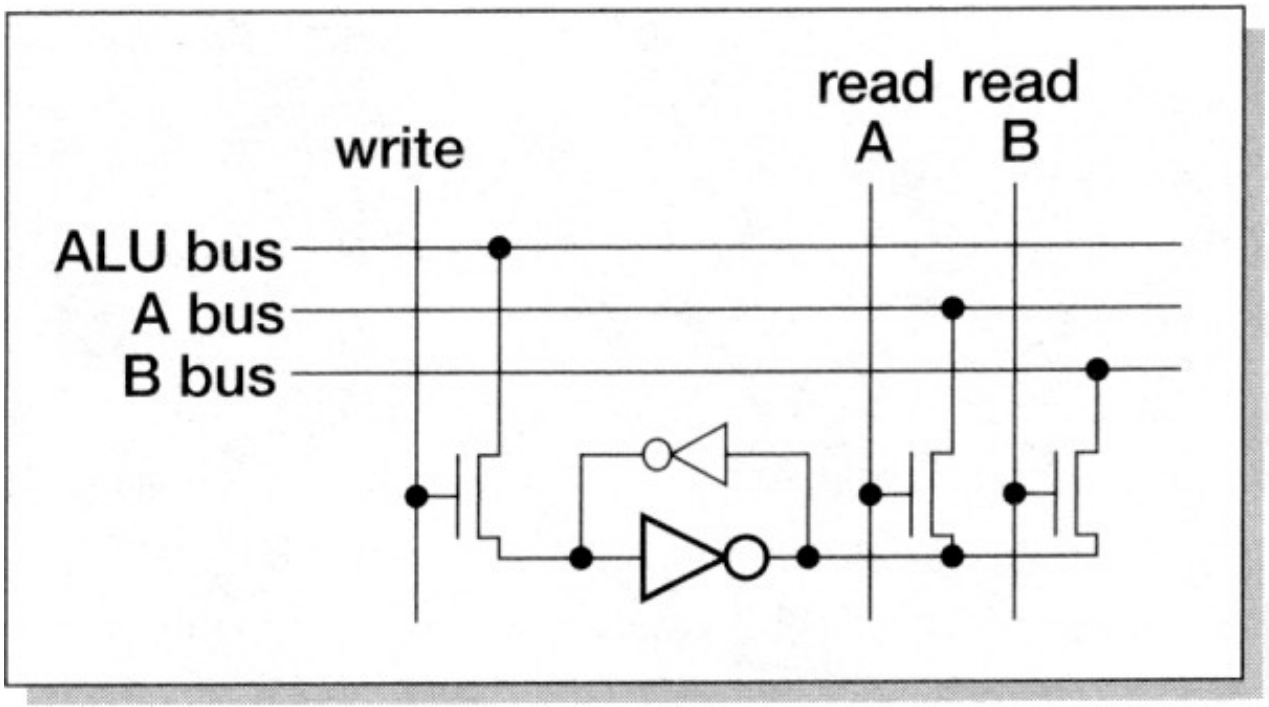
diese und viele folgende Abbildungen: S. Furber, *ARM System-on-Chip Architecture*

RT-Ebene: ALU des ARM-7 Prozessors



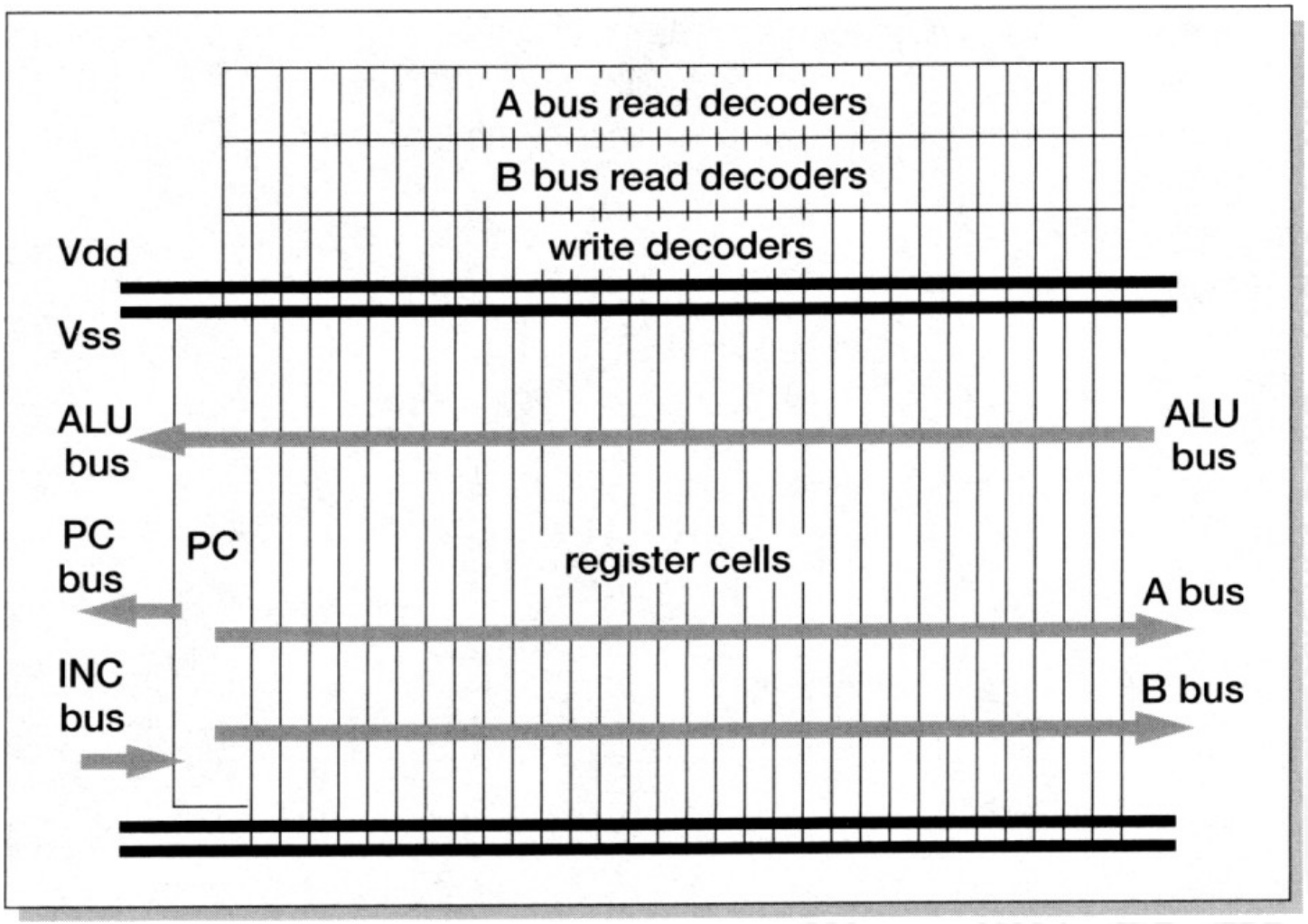
- Register für die Operanden A und B
- Addierer und separater Block für logische Operationen

Multi-Port-Registerbank: Zelle



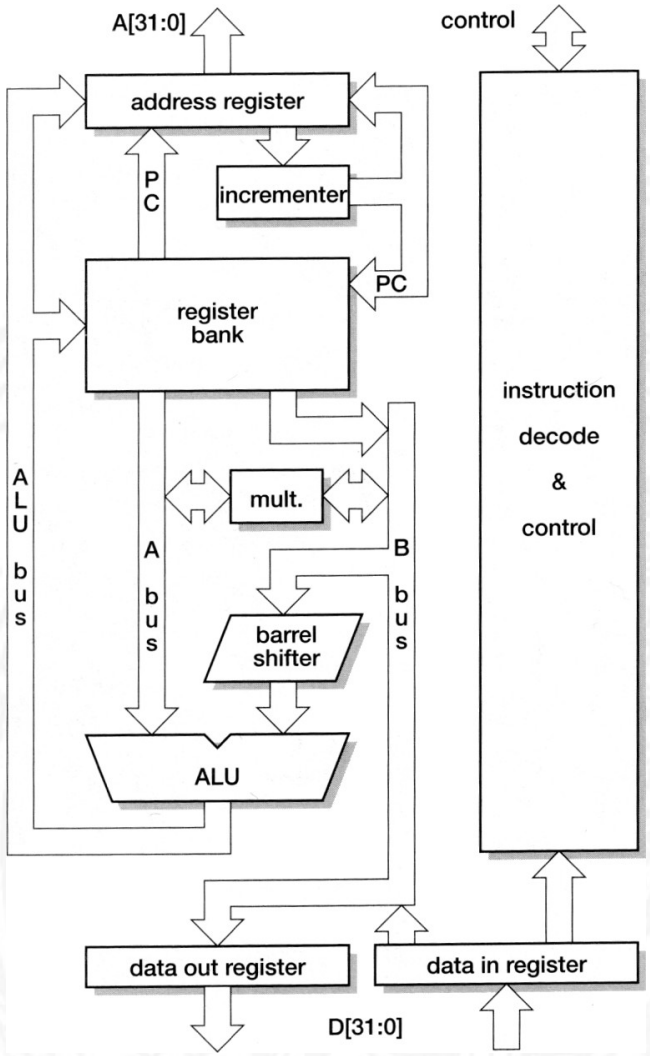
- Prinzip wie 6T-SRAM: rückgekoppelte Inverter
- mehrere (hier zwei) parallele Lese-Ports
- mehrere Schreib-Ports möglich, aber kompliziert

Multi-Port Registerbank: Floorplan/Chiplayout

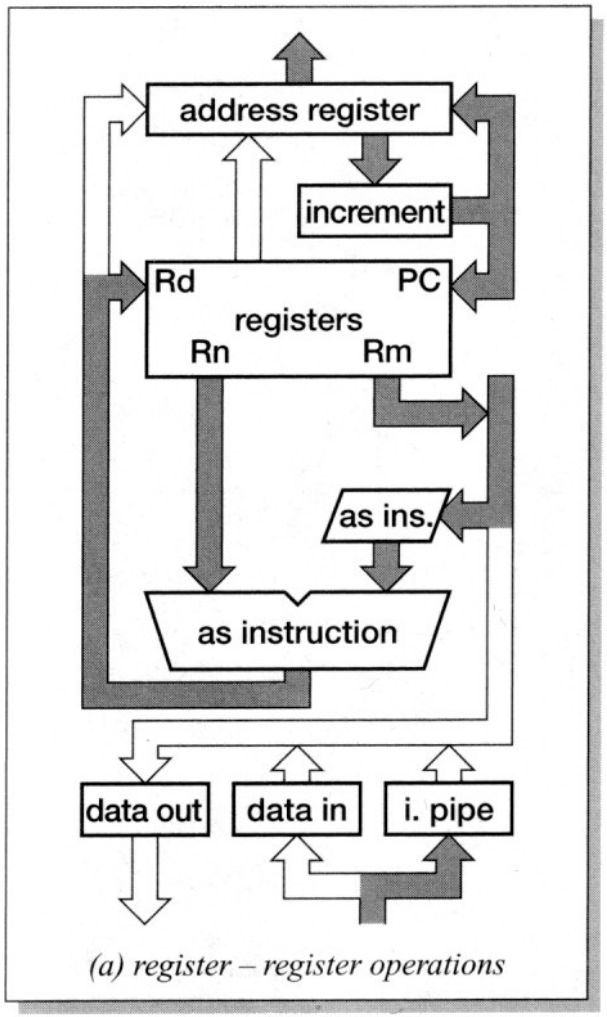


Kompletter Prozessor: ARM-3

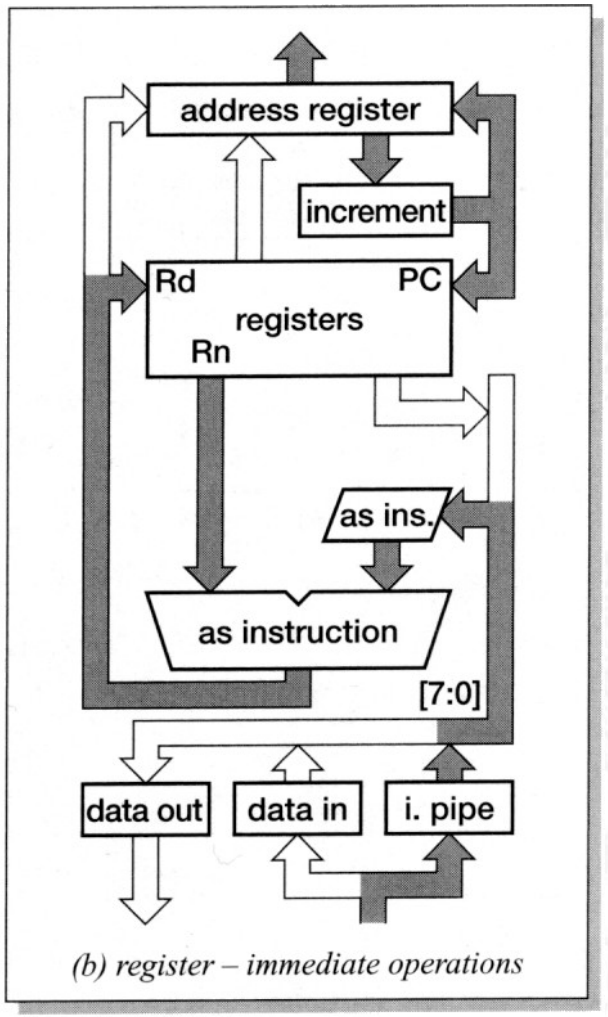
- Registerbank (inkl. Program Counter)
- Inkrementer
- Adress-Register
- ALU, Multiplizierer, Shifter
- Speicherinterface (Data-In / -Out)
- Steuerwerk



ARM-3 Datentransfer: Register-Operationen

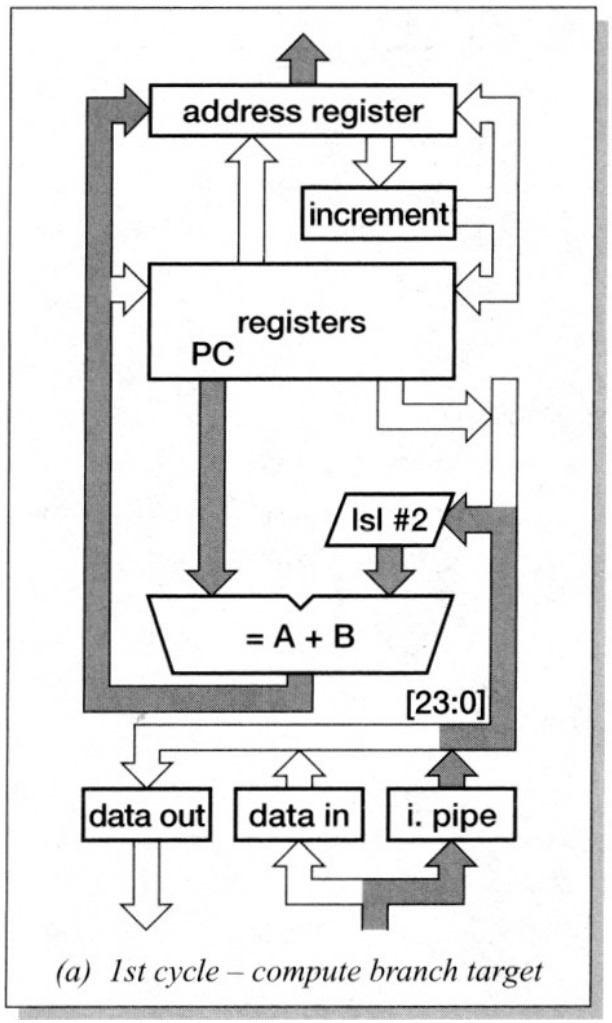


(a) register – register operations

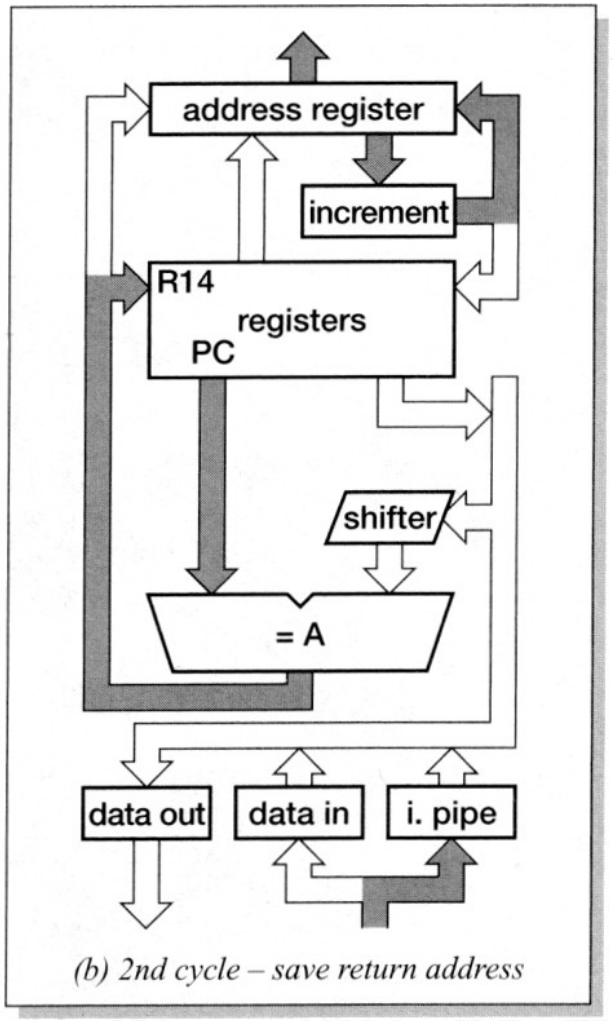


(b) register – immediate operations

ARM-3 Datentransfer: Funktionsaufruf/Sprungbefehl

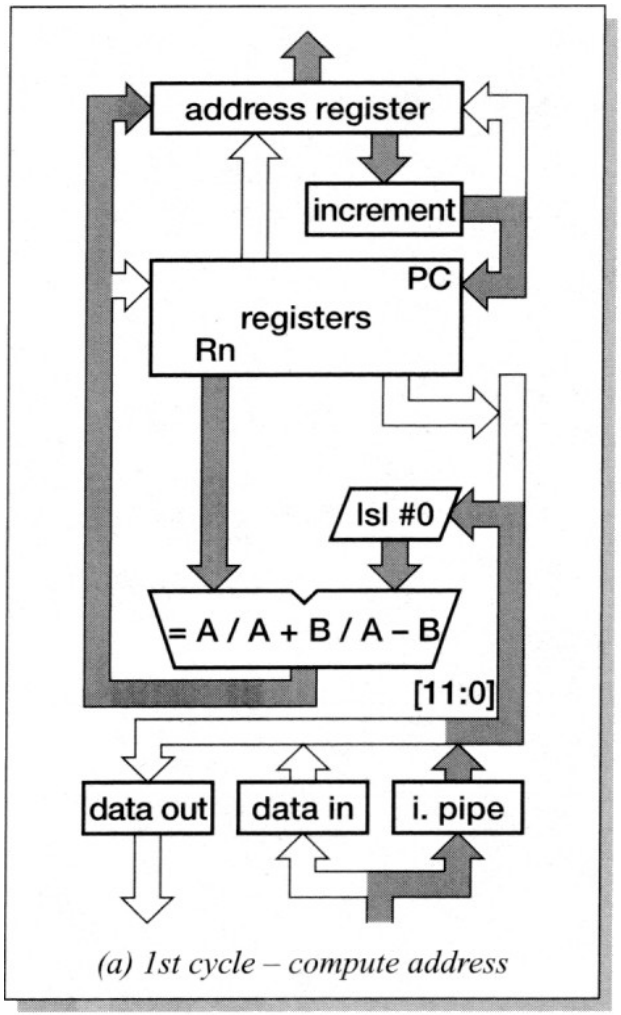


(a) 1st cycle – compute branch target

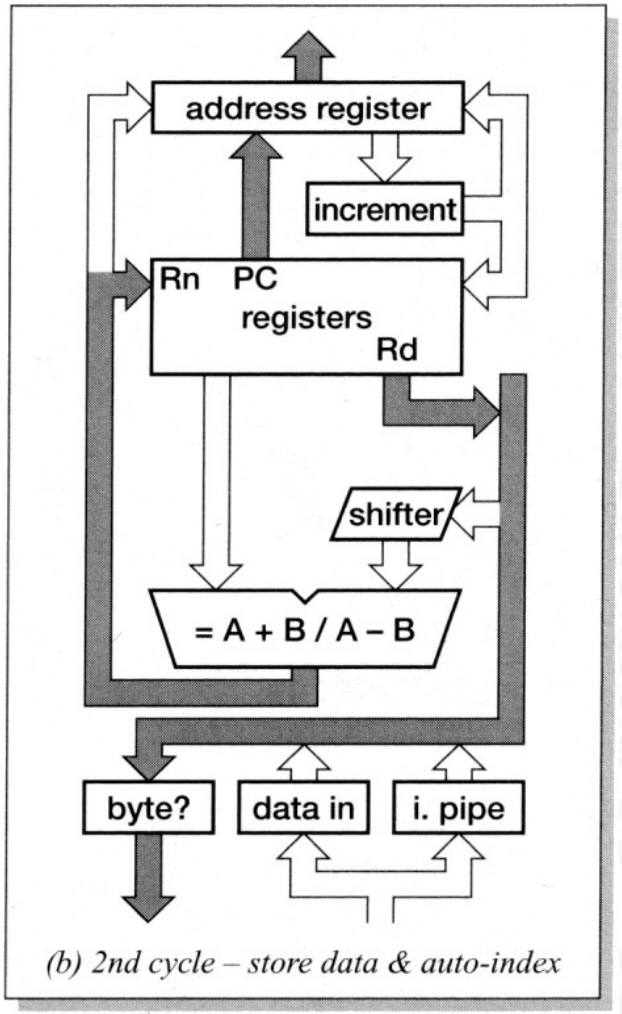


(b) 2nd cycle – save return address

ARM-3 Datentransfer: Store-Befehl



(a) 1st cycle – compute address



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

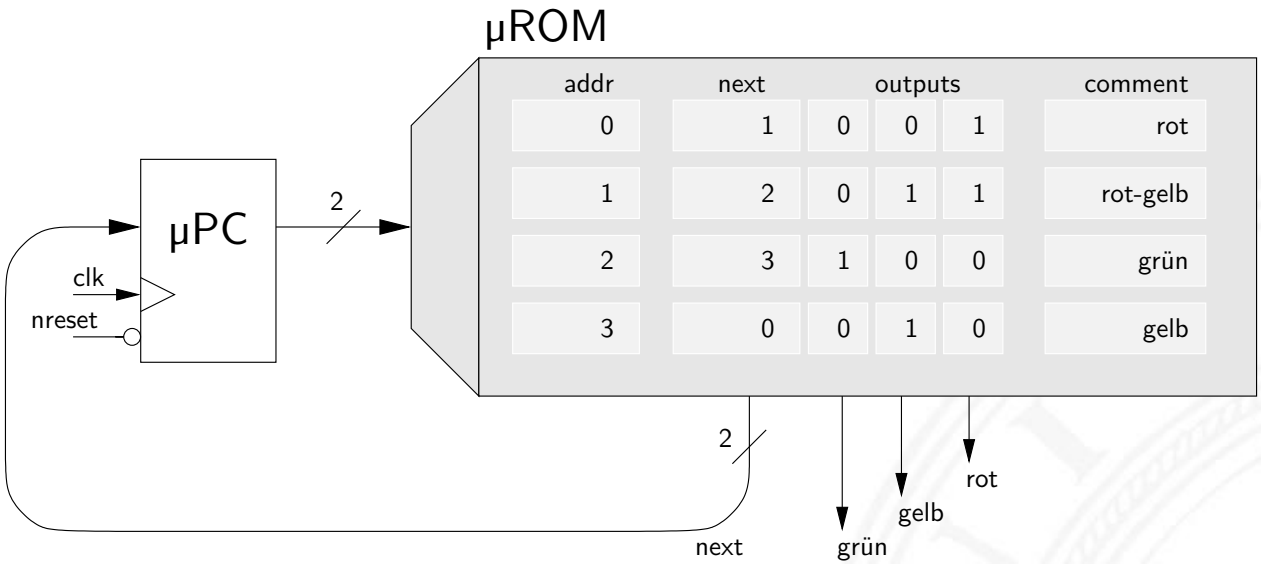
Ablaufsteuerung mit Mikroprogramm

- 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“

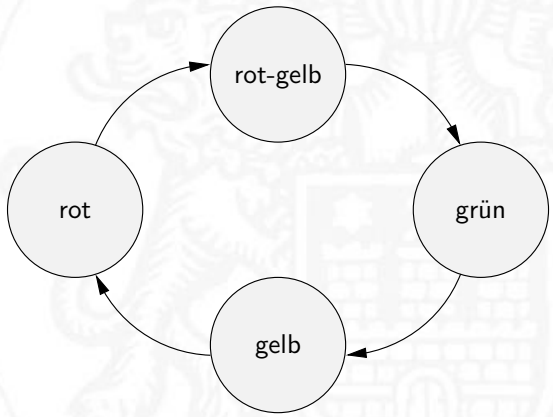
Mikroprogramm: Beispiel Ampel



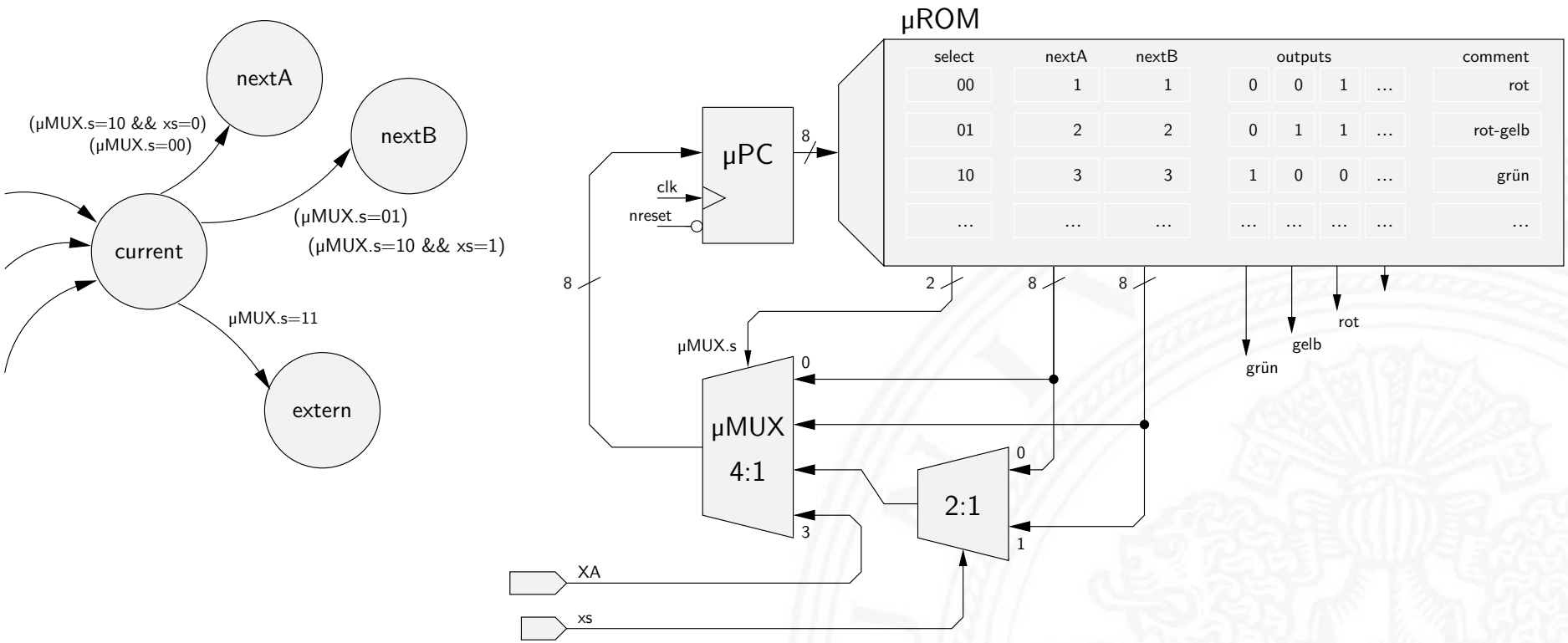
- µPC adressiert das µROM
- "next"-Ausgang liefert den Folgezustand

  (Adresse 0: Wert 1, Adresse 1: Wert 2, usw)
- andere Ausgänge steuern die Schaltung

  (hier die Lampen der Ampel)

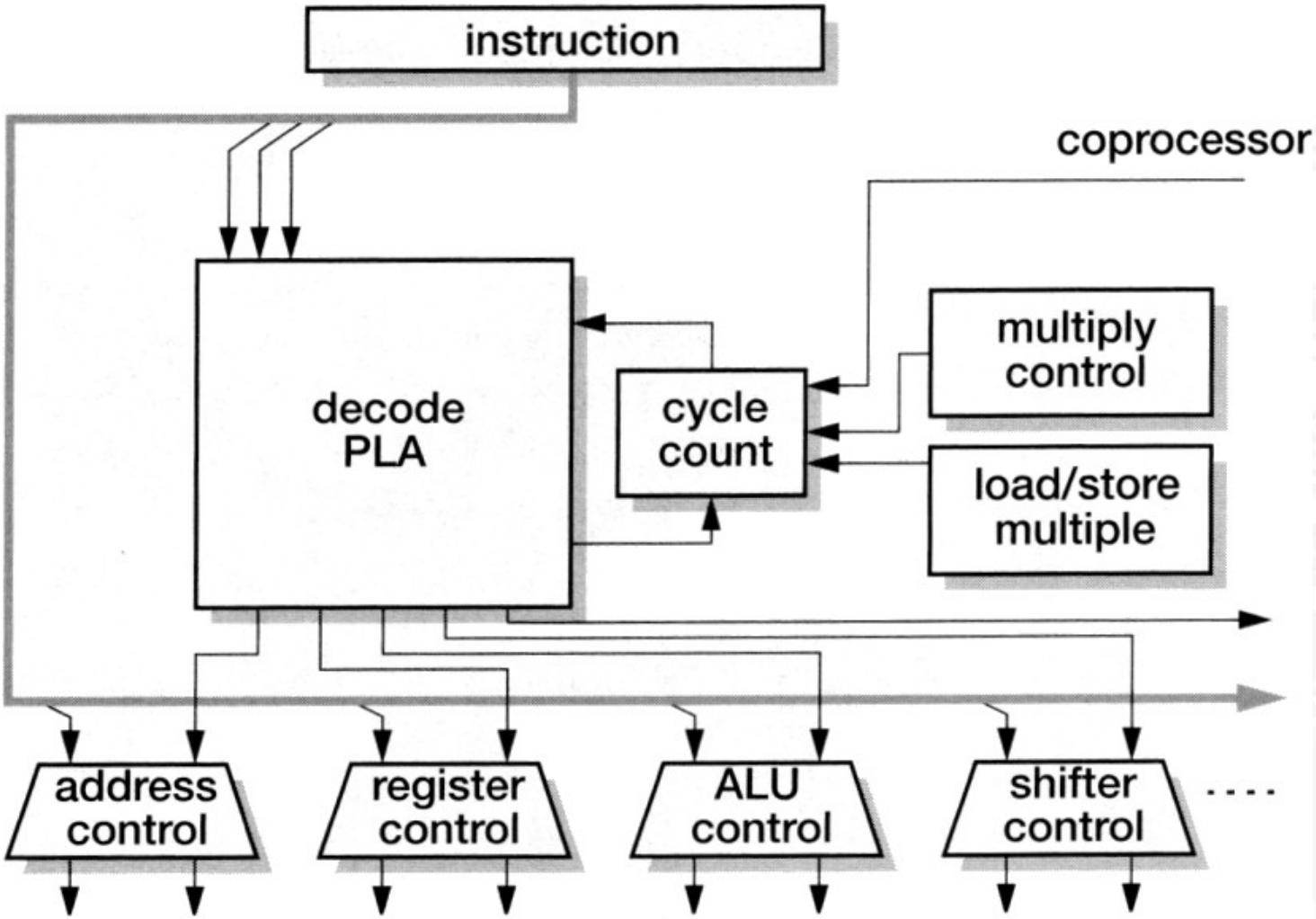


Mikroprogramm: Beispiel zur Auswahl des Folgezustands



- Mulitplexer erlaubt Auswahl des µPC Werts
- "nextA", "nextB" aus dem µROM, externer "XA" Wert
- "xs" Eingang erlaubt bedingte Sprünge

Mikroprogramm: Befehlsdecoder des ARM-7 Prozessors



Literatur: Quellen für die Abbildungen

- Andrew S. Tanenbaum,
  Computerarchitektur: Strukturen, Konzepte, Grundlagen,
  5. Auflage, Pearson Studium, 2006
- Steven Furber,
  ARM System-on-Chip Architecture,
  Addison-Wesley Professional, 2001
- Andreas Mäder,
  Vorlesung: Rechnerarchitektur und Mikrosystemtechnik,
  Universität Hamburg, FB Informatik, 2010
  `tams.informatik.uni-hamburg.de/lectures/2010ws/vorlesung/ram`