64-040 Modul IP7: Rechnerstrukturen

http://tams.informatik.uni-hamburg.de/lectures/2012ws/vorlesung/rs

– Kapitel 13 –

Andreas Mäder

Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

Wintersemester 2012/2013

Kapitel 13

Grundkomponenten für Rechensysteme

- Motivation
- Speicherbausteine
- Busse
- Beispielsystem: ARM
- Mikroprogrammierung
- Literatur

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)

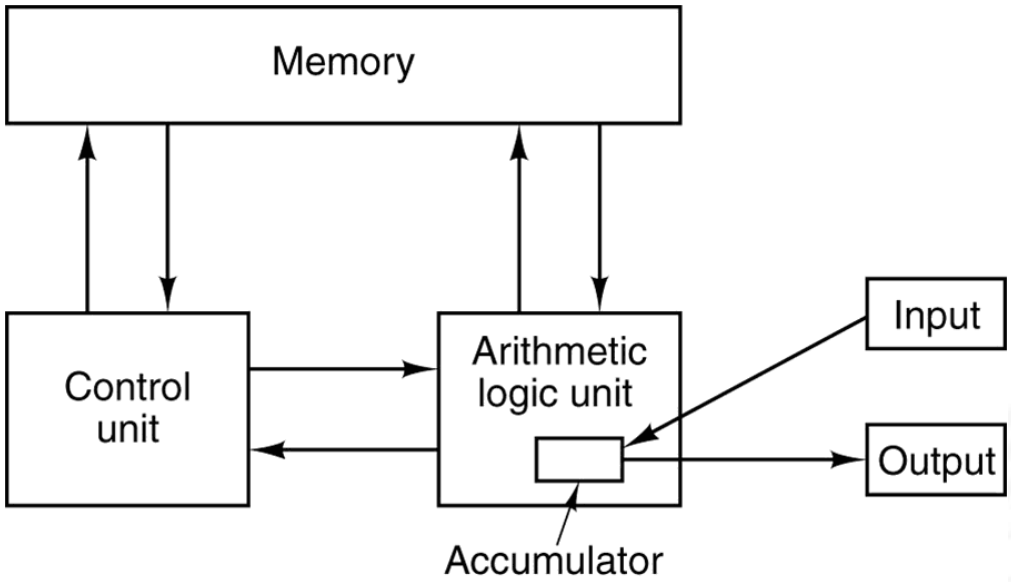
Wiederholung: von-Neumann Konzept

- J. Mauchly, J.P. Eckert, J. von-Neumann 1945
- 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

$\Rightarrow$ enorm flexibel

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

Wiederholung: von-Neumann Rechner



[Tan09]

Fünf zentrale Komponenten:

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

Wiederholung: von-Neumann Rechner (cont.)

- 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*)

Systemmodellierung

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

$\Rightarrow$ Register-Transfer Modell

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

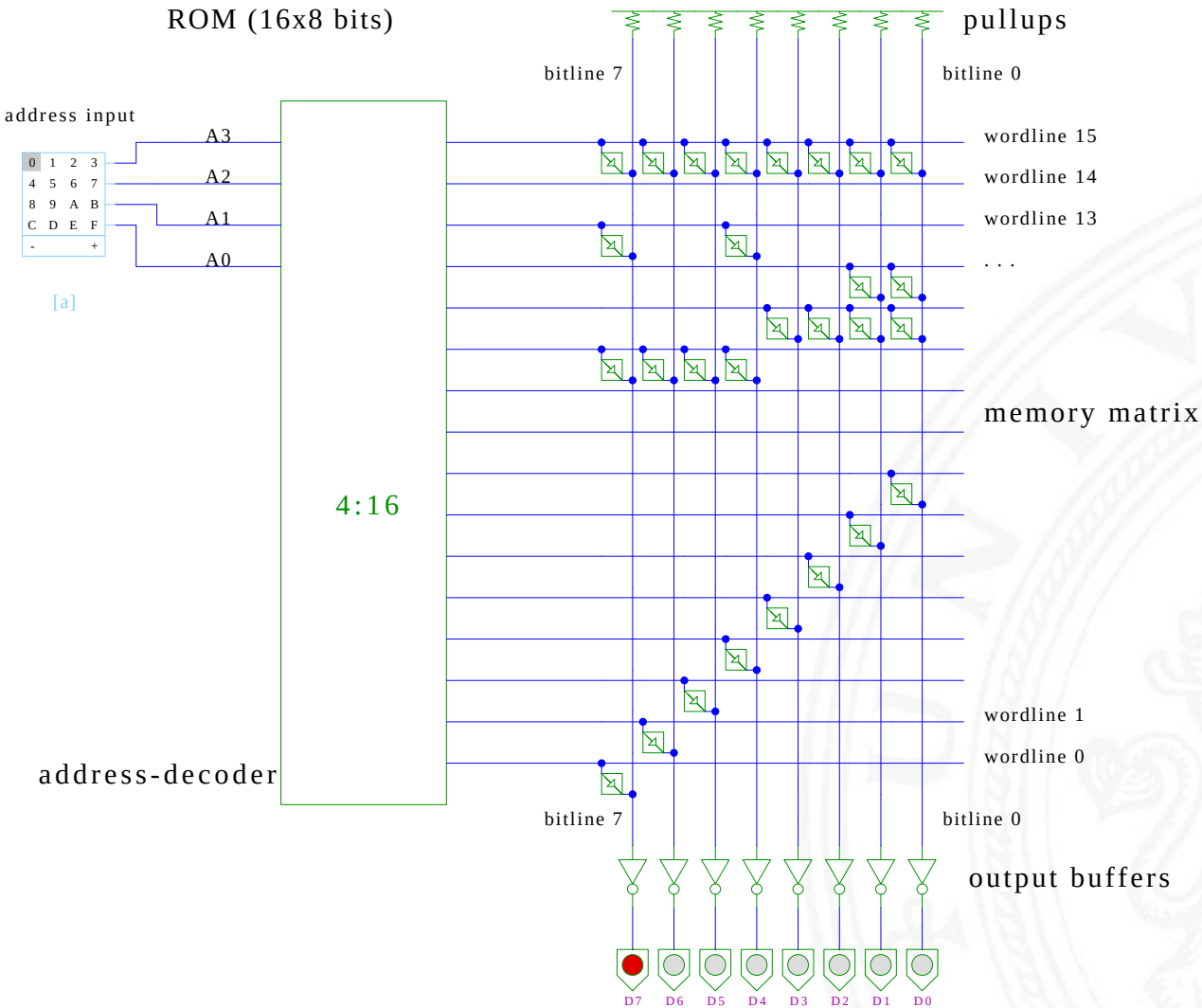
- Klassifikation:
  - Speicherkapazität
  - Schreibzugriffe möglich?
  - Schreibzugriffe auf einzelne bits/Bytes oder nur Blöcke?
  - Information flüchtig oder dauerhaft gespeichert?
  - Zugriffszeiten beim Lesen und Schreiben
  - Technologie

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 (old)
SDRAM	Read/write	Electrical	Yes	Yes	Main memory (new)
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

[Tan09]

ROM: Read-Only Memory



[Hen] Hades Webdemo:
`40-memories/20-rom/rom`

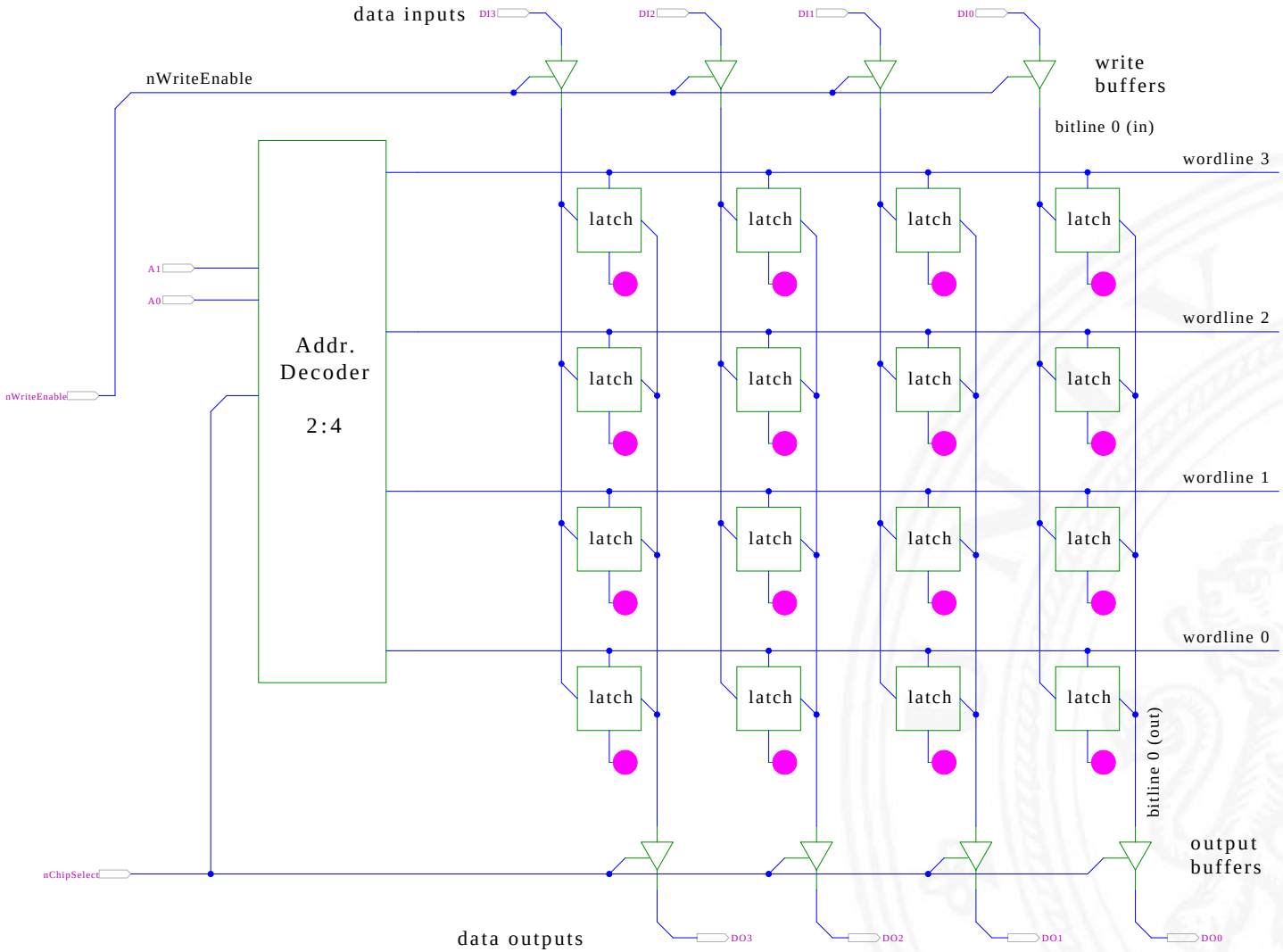
RAM: Random-Access Memory

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

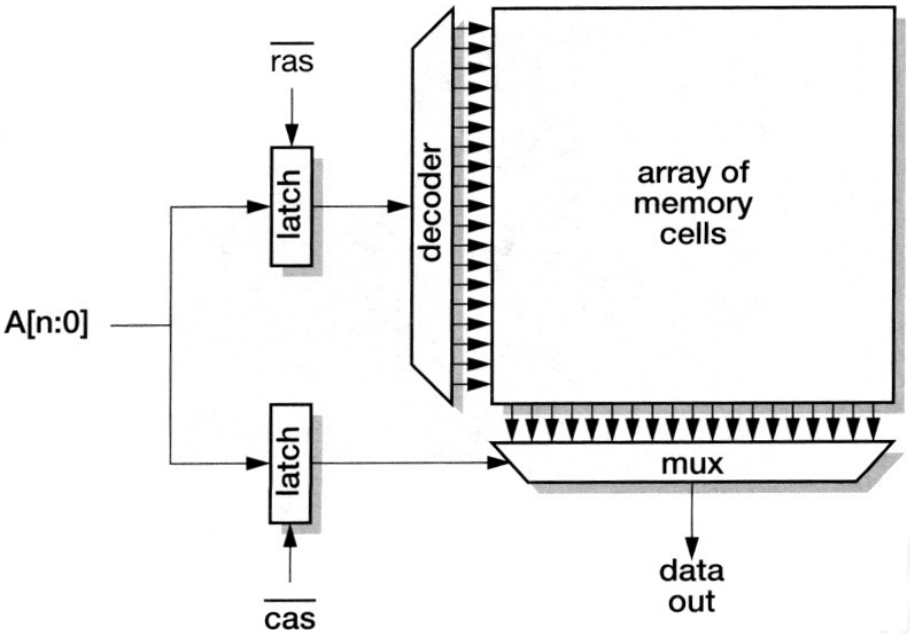
RAM: Blockschaltbild



4×4 bit
2-bit Adresse
4-bit Datenwort

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

RAM: RAS/CAS-Adressdecodierung



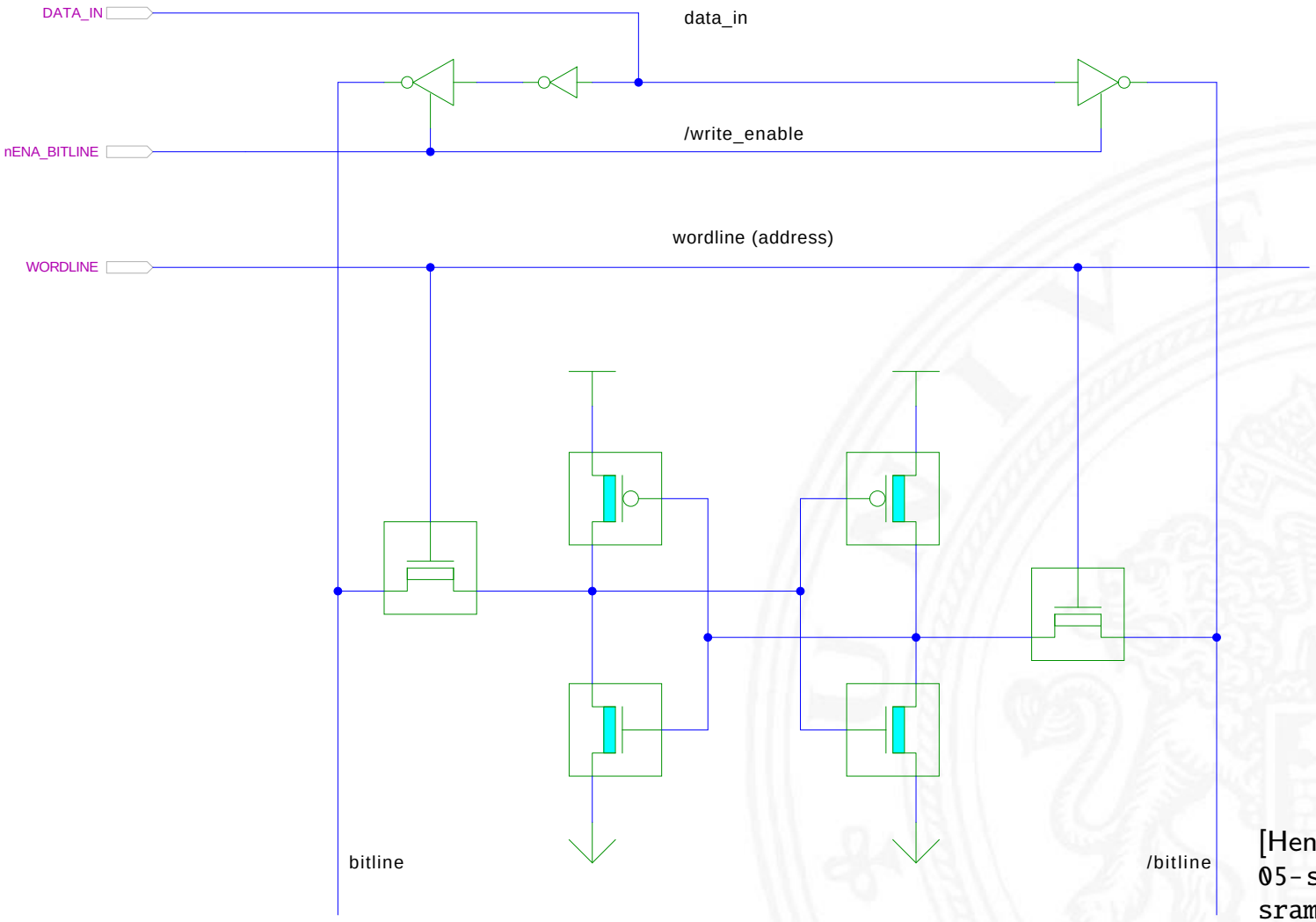
Furber: *ARM SoC Architecture* [Fur01]

- 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

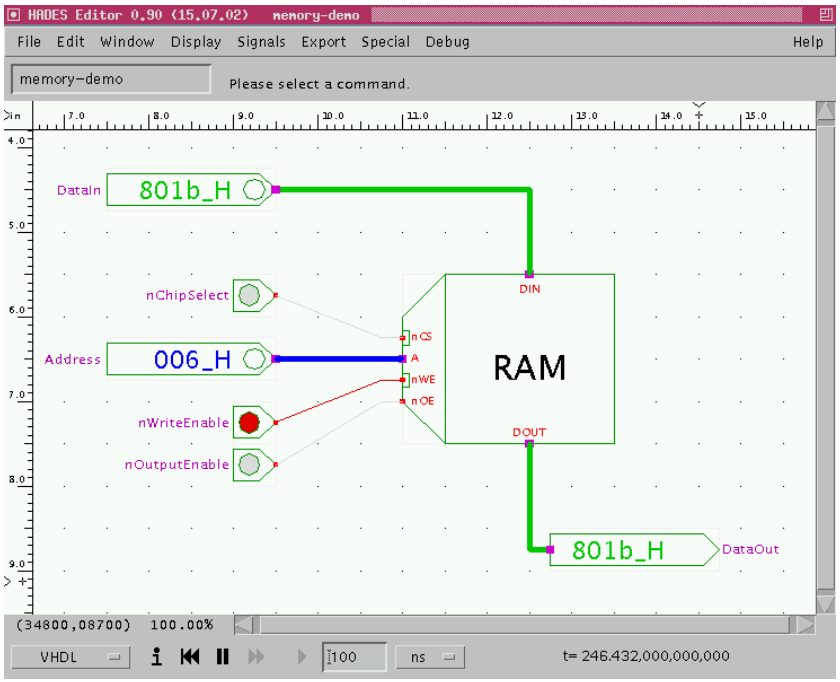
- Inhalt bleibt dauerhaft gespeichert solange Betriebsspannung anliegt
- *sechs-Transistor* Zelle zur Speicherung
  - weniger Platzverbrauch als Latches/Flipflops
  - kompakte Realisierung in CMOS-Technologie (s.u.)
  - zwei rückgekoppelte Inverter zur Speicherung
  - zwei n-Kanal Transistoren zur Anbindung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

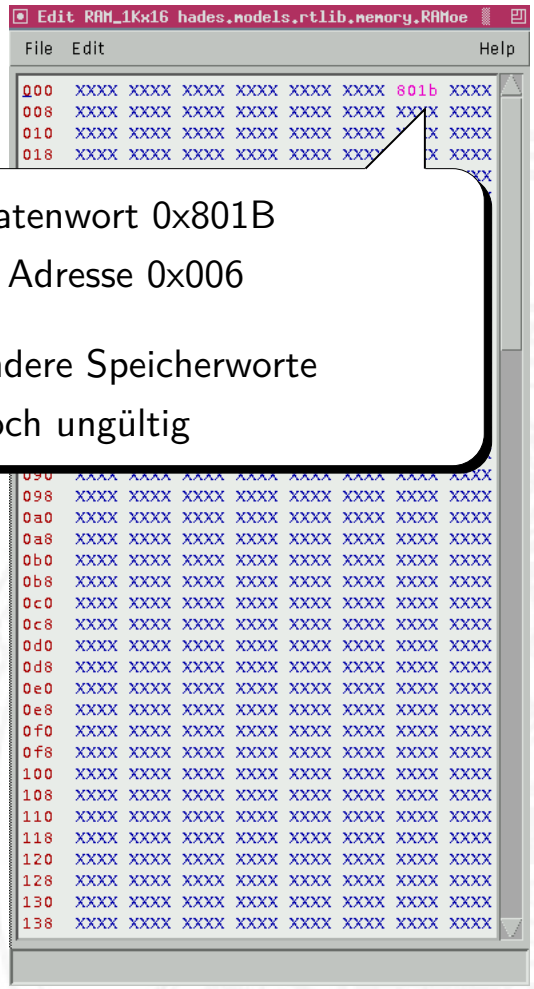
SRAM: Sechs-Transistor Speicherstelle („6T“)



[Hen] 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*)

[Hen] Hades Webdemo: `50-rtlib/40-memory/ram`

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 [Hen]
  - Hades Webdemo: `40-memories/40-ram/demo-6116`
  - Hades Webdemo: `40-memories/40-ram/two-6116`

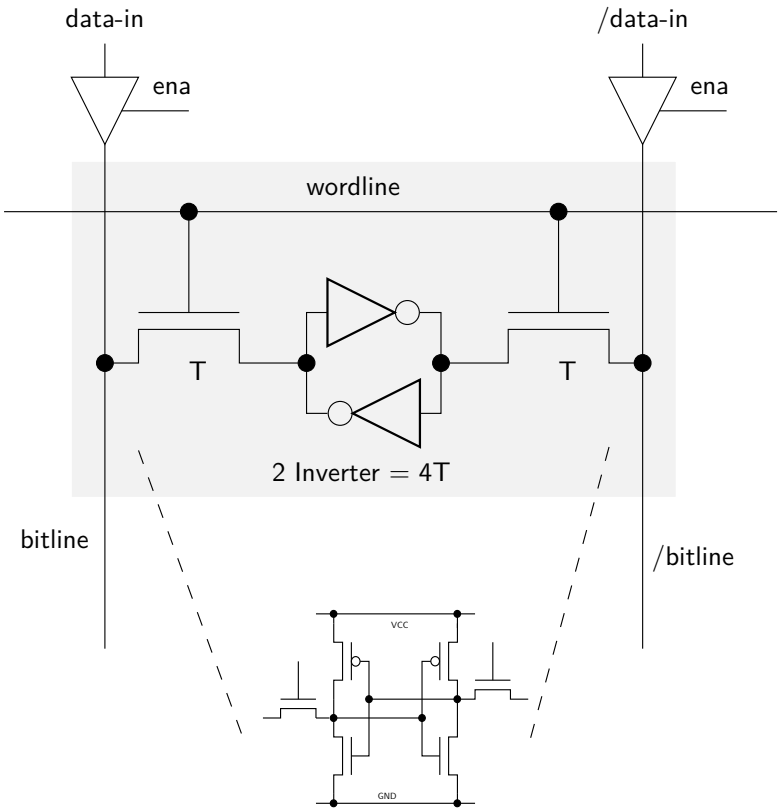
DRAM: dynamisches RAM

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

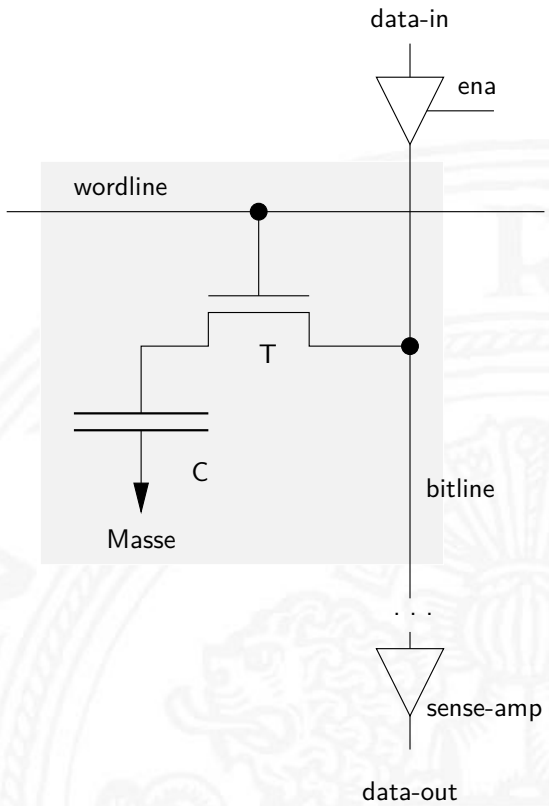
- jeder Lesezugriff entlädt den Kondensator
- *Leseverstärker* zur Messung der Spannung auf der Bitline
  Schwellwertvergleich zur Entscheidung logisch 0/1

– 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
- $C = 10\,\text{fF} \approx 200\,000$ Elektronen
- langsam (sense-amp)
- minimale Fläche

DRAM: Stacked- und Trench-Zelle

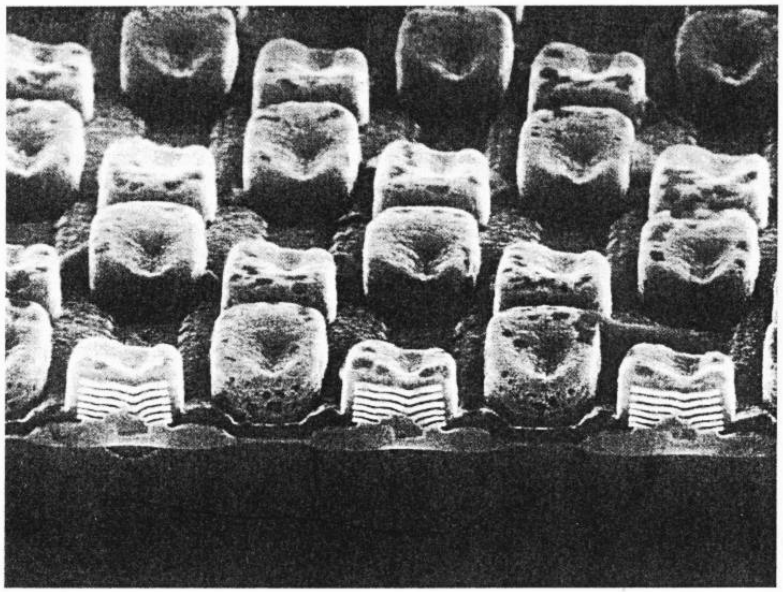
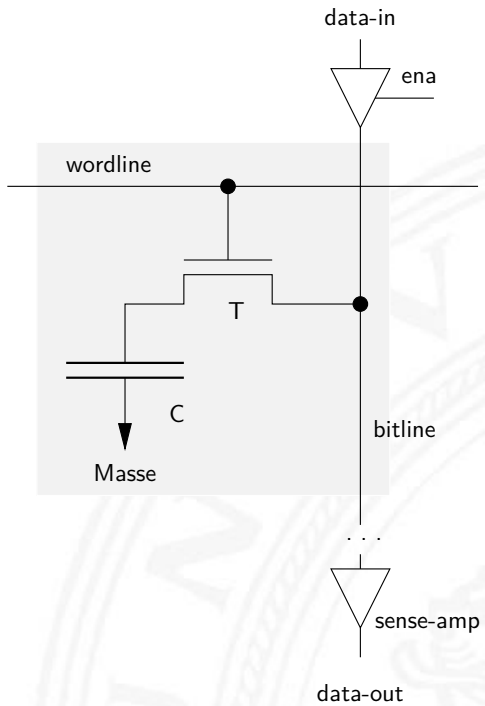
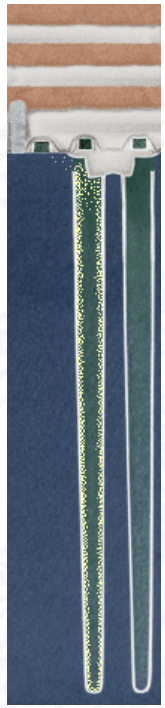


Abb. 7: Prototyp von Speicherzellen (Stapelkondensatoren) für zukünftige Speicherchips wie den Ein-Gigabit-Chip. Da für DRAM-Chips eine minimale Speicherkapazität von 25 fF notwendig ist, bringt es erhebliche Platzvorteile, die Kondensatorelemente vertikal übereinander zu stapeln. Die Dicke der Schichten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

Siemens 1 Gbit DRAM

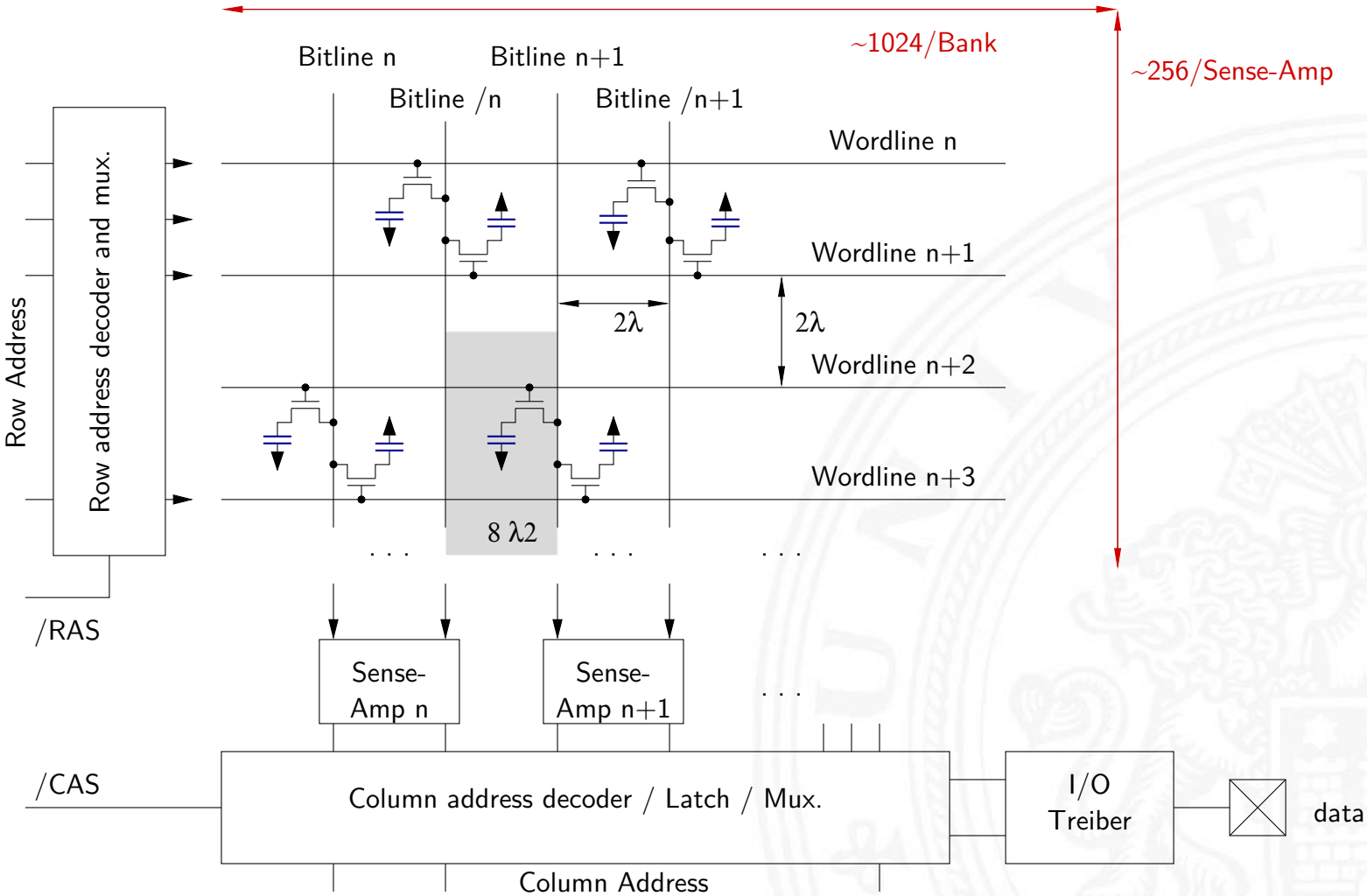




IBM CMOS-6X embedded DRAM

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

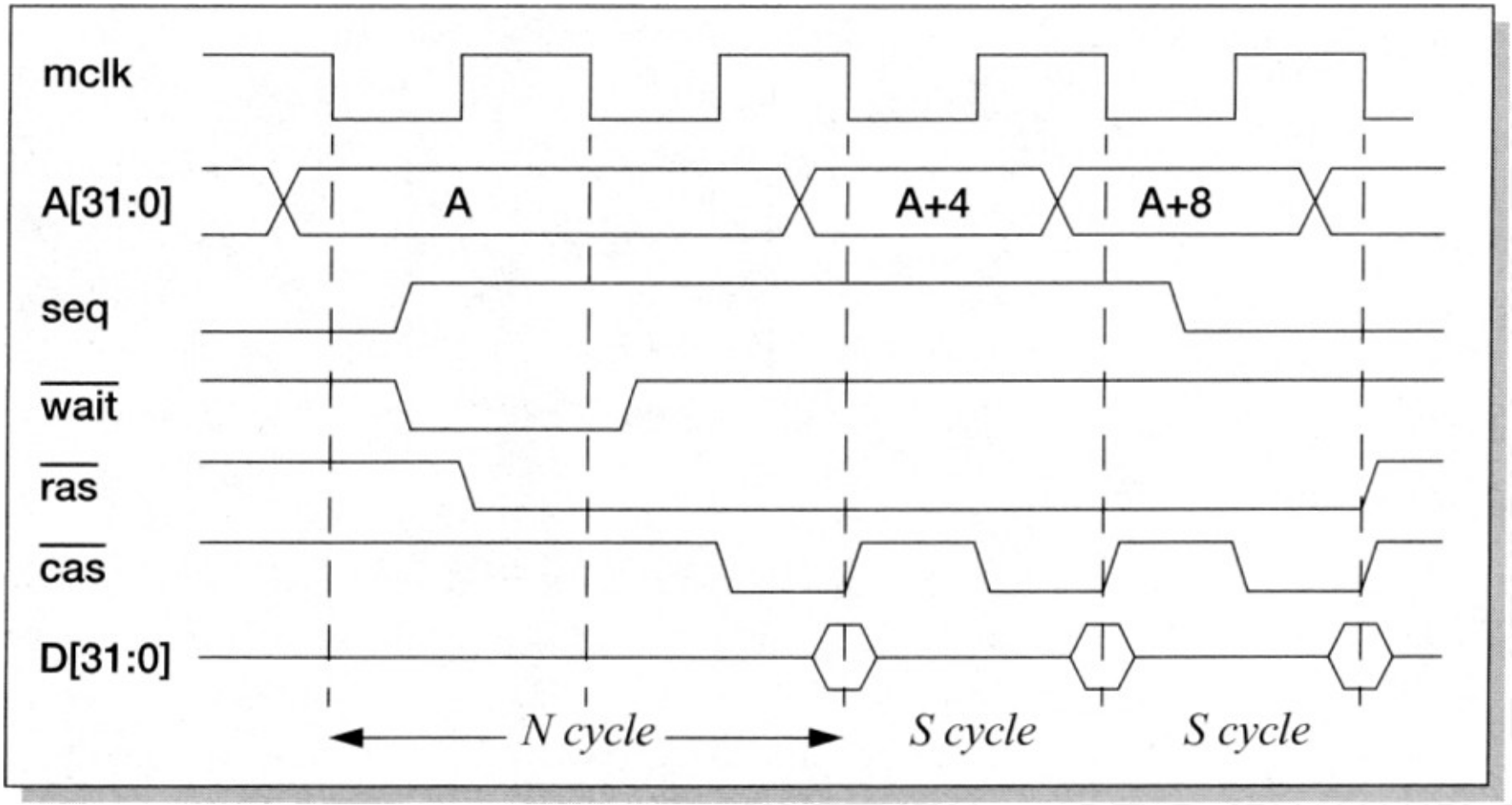
DRAM: Layout


~1024/Bank
~256/Sense-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
  - DDR2, DDR3, DDR4: Varianten mit höherer Taktrate
    aktuell Übertragungsraten bis 25,6 GByte/sec
  - GDDR3...GDDR5 (*Graphics Double Data Rate*)
    bis 48 GByte/sec

SDRAM: Lesezugriff auf sequenzielle Adressen



[Fur01]

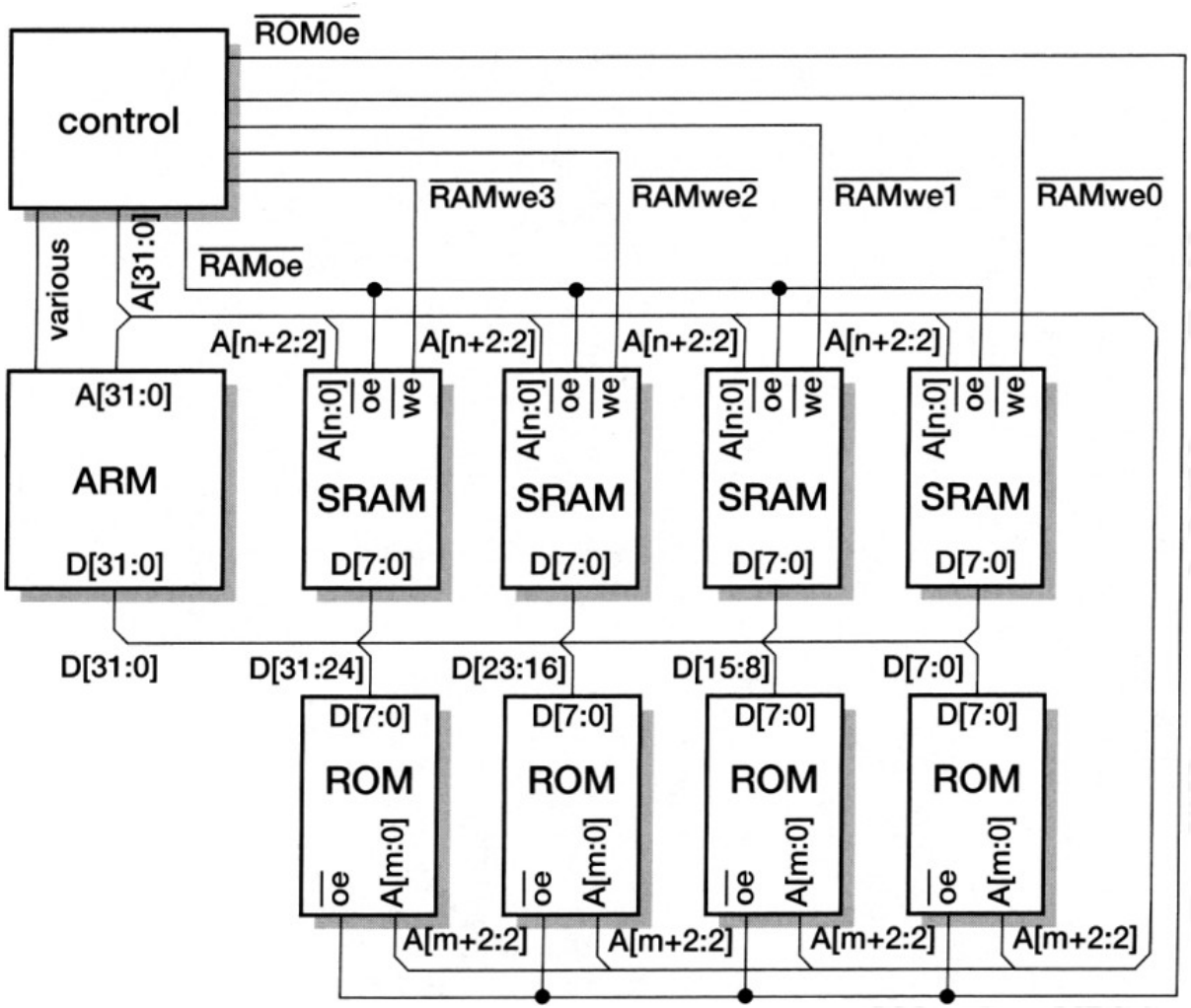
Flash

- 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

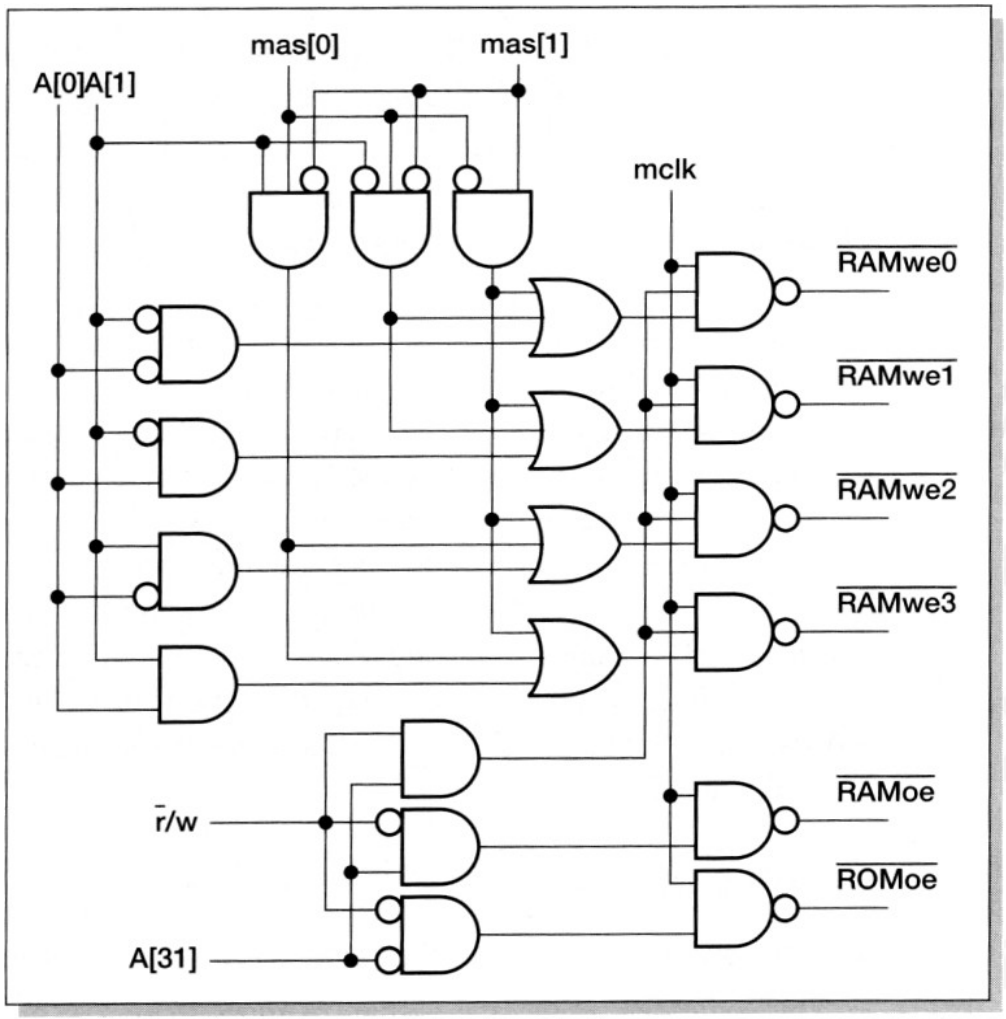
Typisches Speichersystem



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

[Fur01]

Typisches Speichersystem: Adressdecodierung



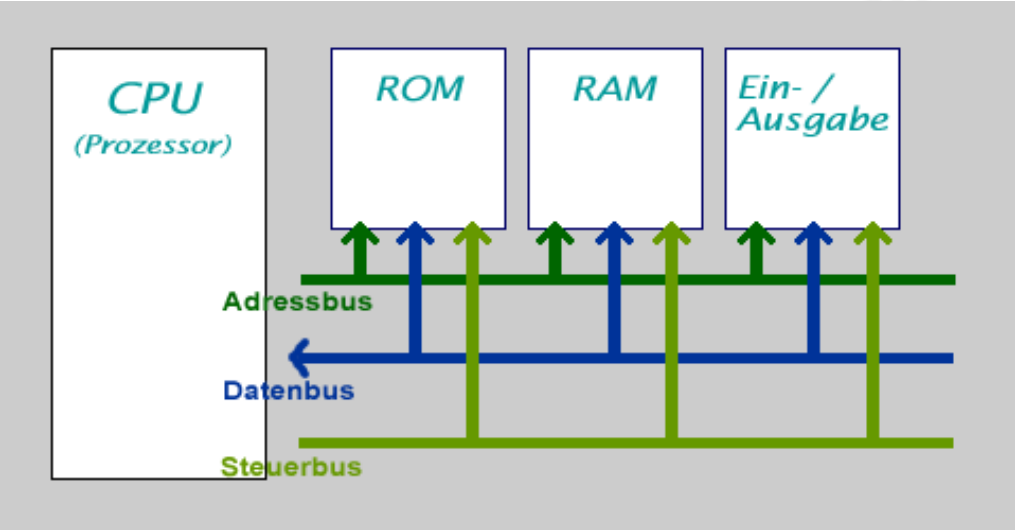
[Fur01]

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)

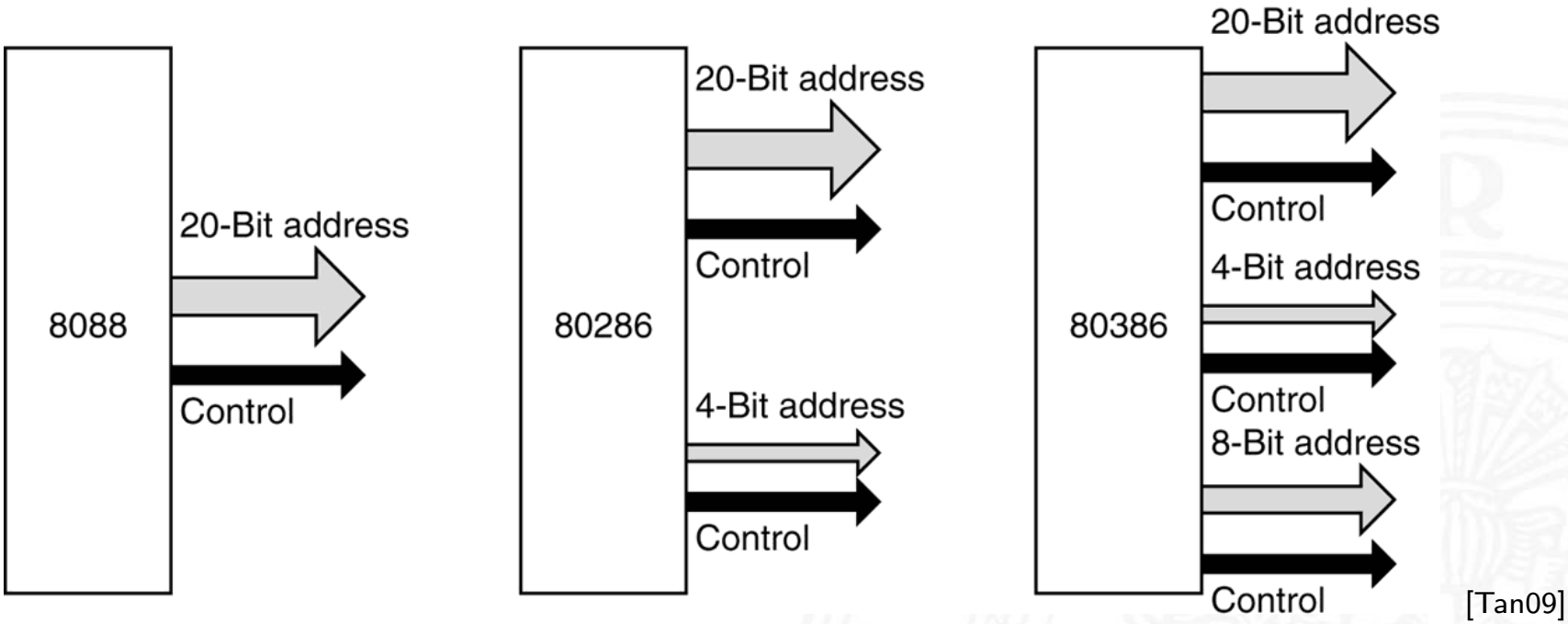
Bus: Mikroprozessorsysteme

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

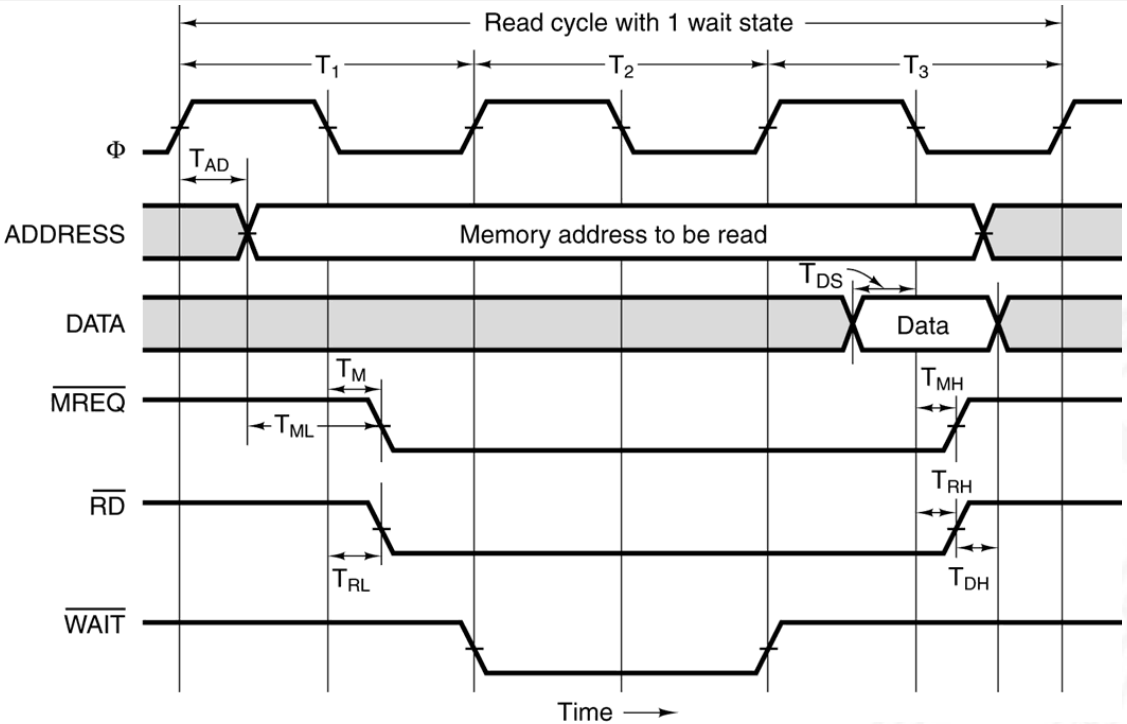
Adressbus: Evolution beim Intel x86



[Tan09]

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

Synchroner Bus: Timing



[Tan09] A.S. Tanenbaum:
Structured Computer Organization

- 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

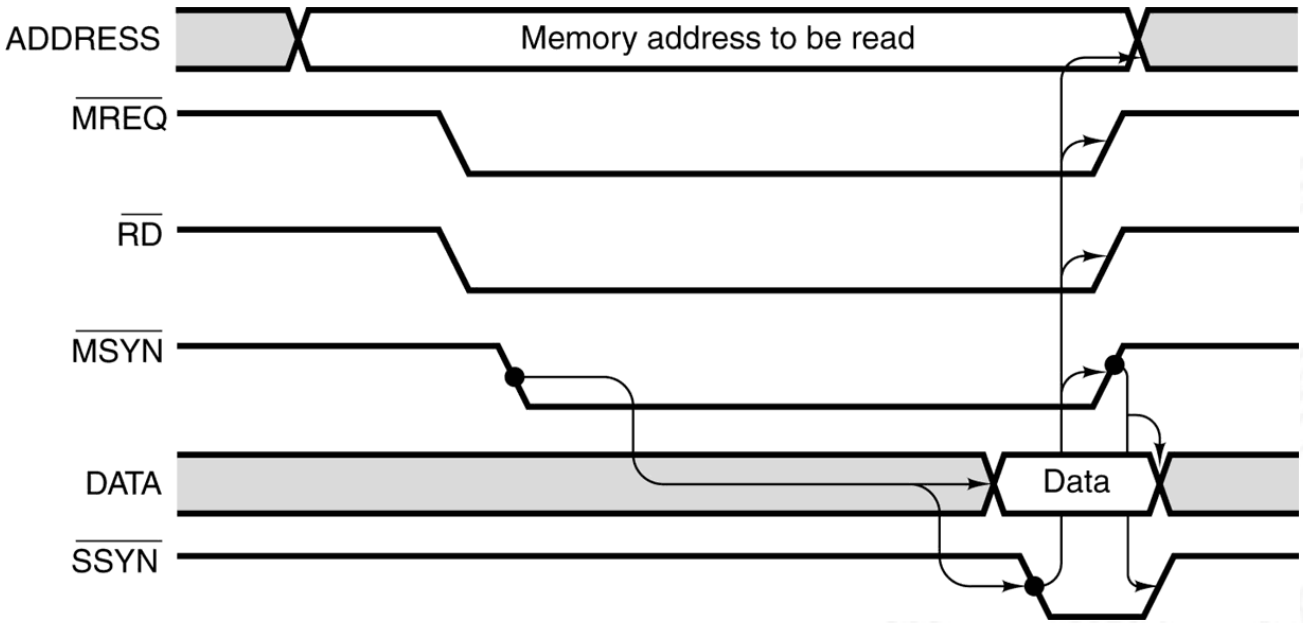
Synchroner Bus: Timing (cont.)

- typische Parameter

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

[Tan09]

Asynchroner Bus: Lesezugriff



[Tan09]

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

Bus Arbitrierung

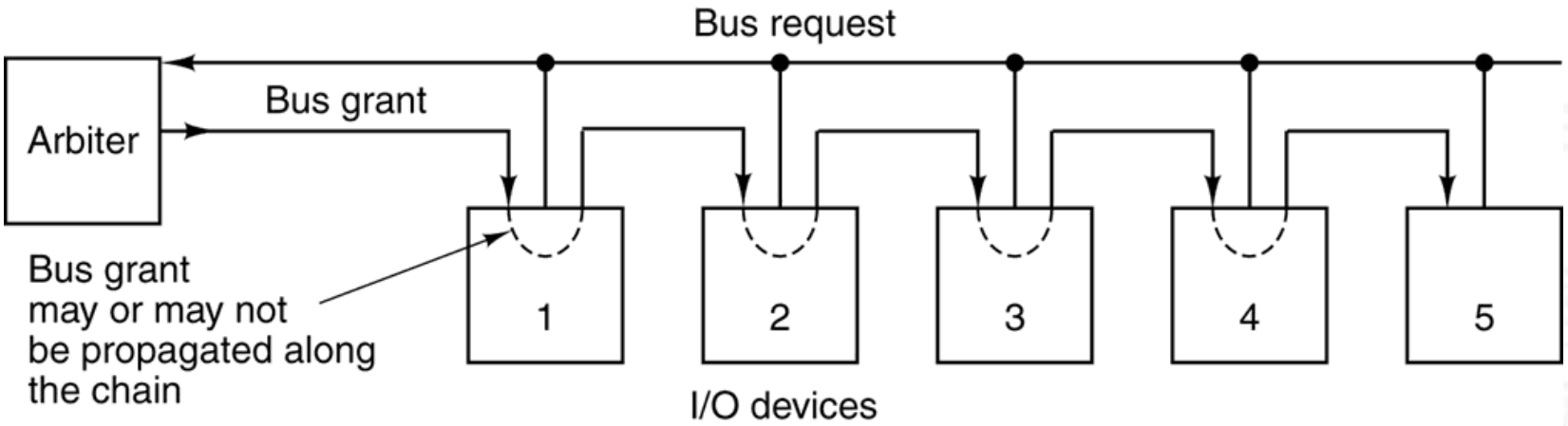
- mehrere Komponenten wollen Übertragung initiieren
  immer nur ein Transfer zur Zeit möglich
- der Zugriff muss serialisiert werden

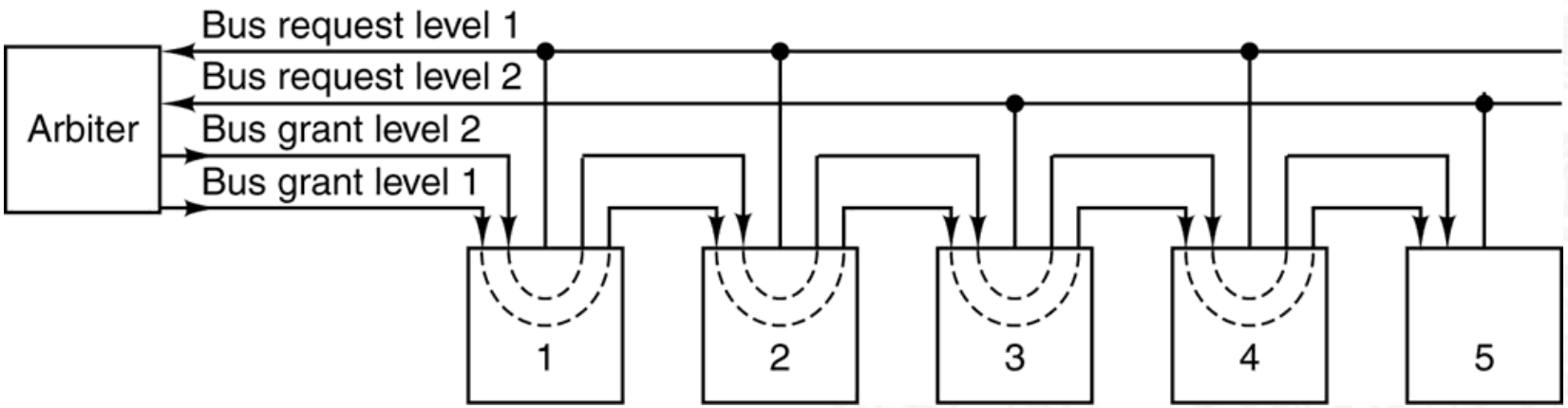
1. zentrale Arbitrierung
   - Arbiter gewährt Bus-Requests
   - Strategien
     - Prioritäten für verschiedene Geräte
     - „round-robin“ Verfahren
     - „Token“-basierte Verfahren
     - usw.

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.)





[Tan09]

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_bandwidths`

Beispiel: PCI-Bus

Peripheral Component Interconnect (Intel 1991)

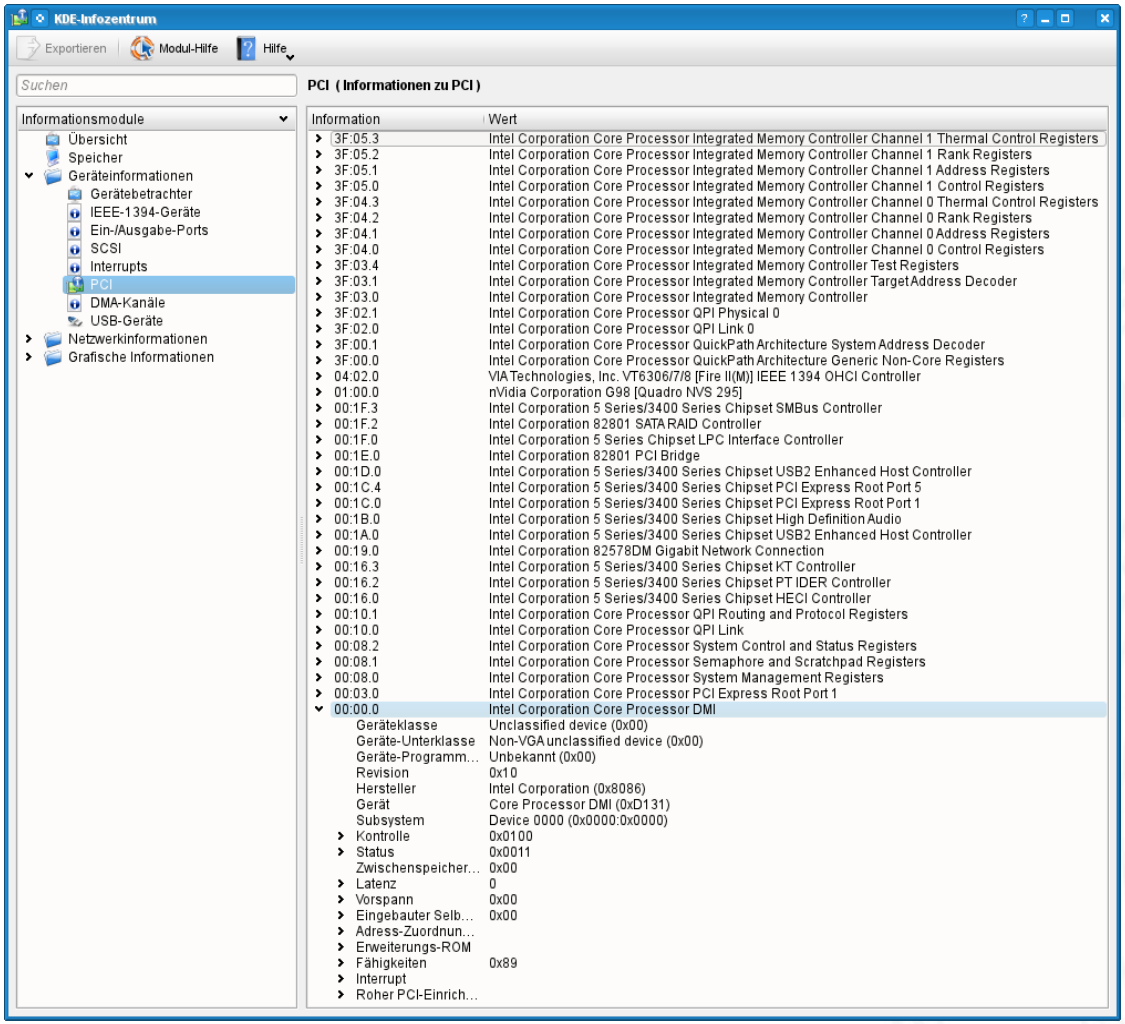
- 33 MHz Takt — optional 64 MHz Takt
- 32-bit Bus-System — optional auch 64-bit
- gemeinsame Adress-/Datenleitungen
- Arbitrierung durch Bus-Master — CPU

- 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

```
[maeder@tams110]~> lspci
00:00.0 Host bridge: Intel Corporation Ivy Bridge DRAM Controller (rev 09)
00:01.0 PCI bridge: Intel Corporation Ivy Bridge PCI Express Root Port (rev 09)
00:14.0 USB controller: Intel Corporation Panther Point USB xHCI Host Controller (rev 04)
00:16.0 Communication controller: Intel Corporation Panther Point MEI Controller #1 (rev...
00:19.0 Ethernet controller: Intel Corporation 82579LM Gigabit Network Connection (rev 04)
00:1a.0 USB controller: Intel Corporation Panther Point USB Enhanced Host Controller #2 ...
00:1b.0 Audio device: Intel Corporation Panther Point High Definition Audio Controller  ...
00:1d.0 USB controller: Intel Corporation Panther Point USB Enhanced Host Controller #1 ...
00:1e.0 PCI bridge: Intel Corporation 82801 PCI Bridge (rev a4)
00:1f.0 ISA bridge: Intel Corporation Panther Point LPC Controller (rev 04)
00:1f.2 SATA controller: Intel Corporation Panther Point 6 port SATA Controller [AHCI mode]
00:1f.3 SMBus: Intel Corporation Panther Point SMBus Controller (rev 04)
01:00.0 VGA compatible controller: NVIDIA Corporation GF108 [Quadro 600] (rev a1)
01:00.1 Audio device: NVIDIA Corporation GF108 High Definition Audio Controller (rev a1)
02:02.0 FireWire (IEEE 1394): VIA Technologies, Inc. VT6306/7/8 [Fire II(M)] IEEE 1394  ...
```

PCI-Bus: Peripheriegeräte (cont.)

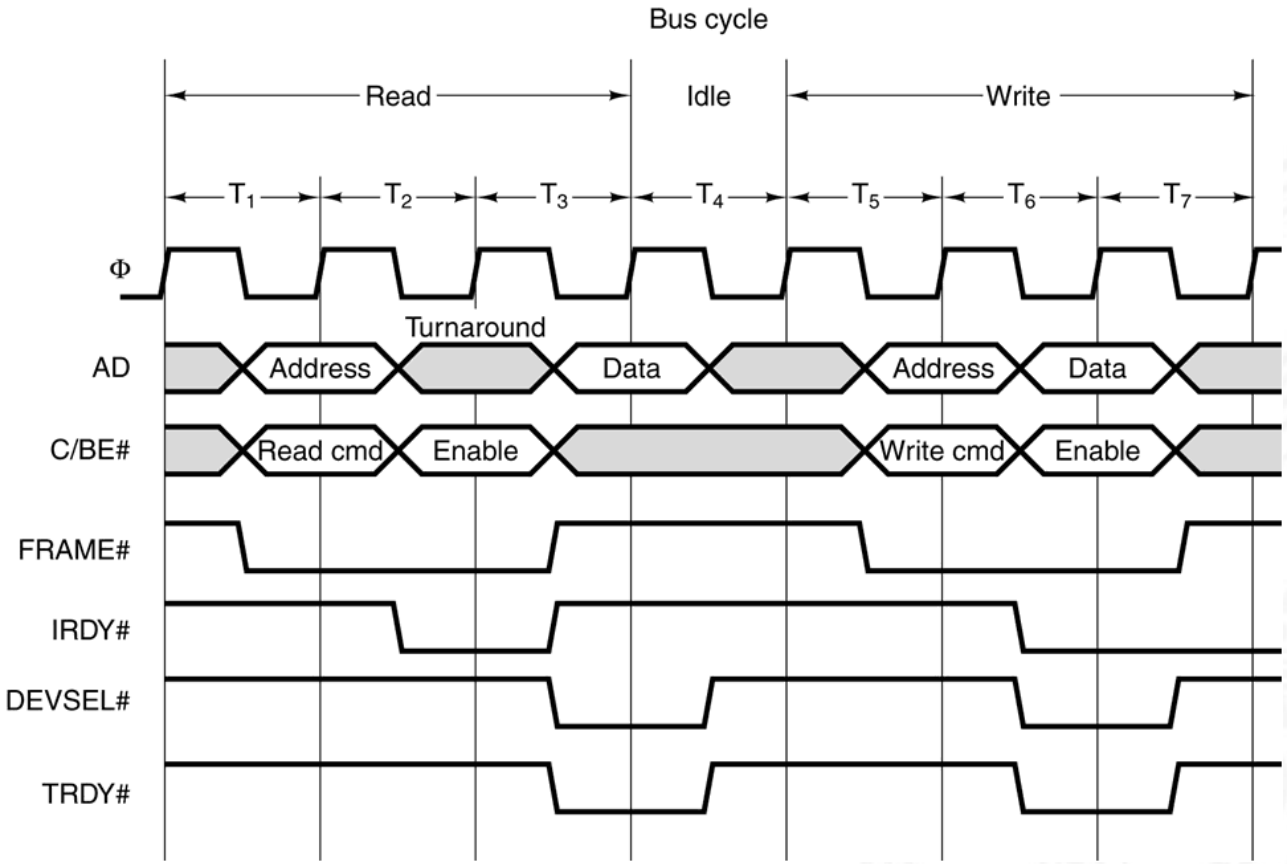


PCI-Bus: Leitungen („mindestens“)

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

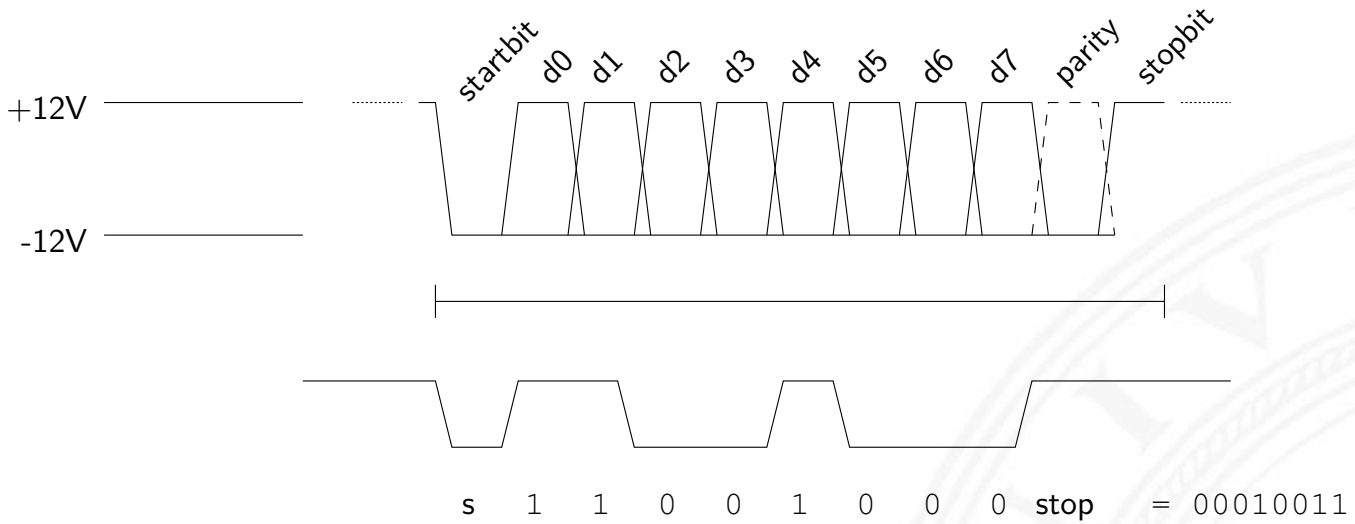
[Tan09]

PCI-Bus: Transaktionen



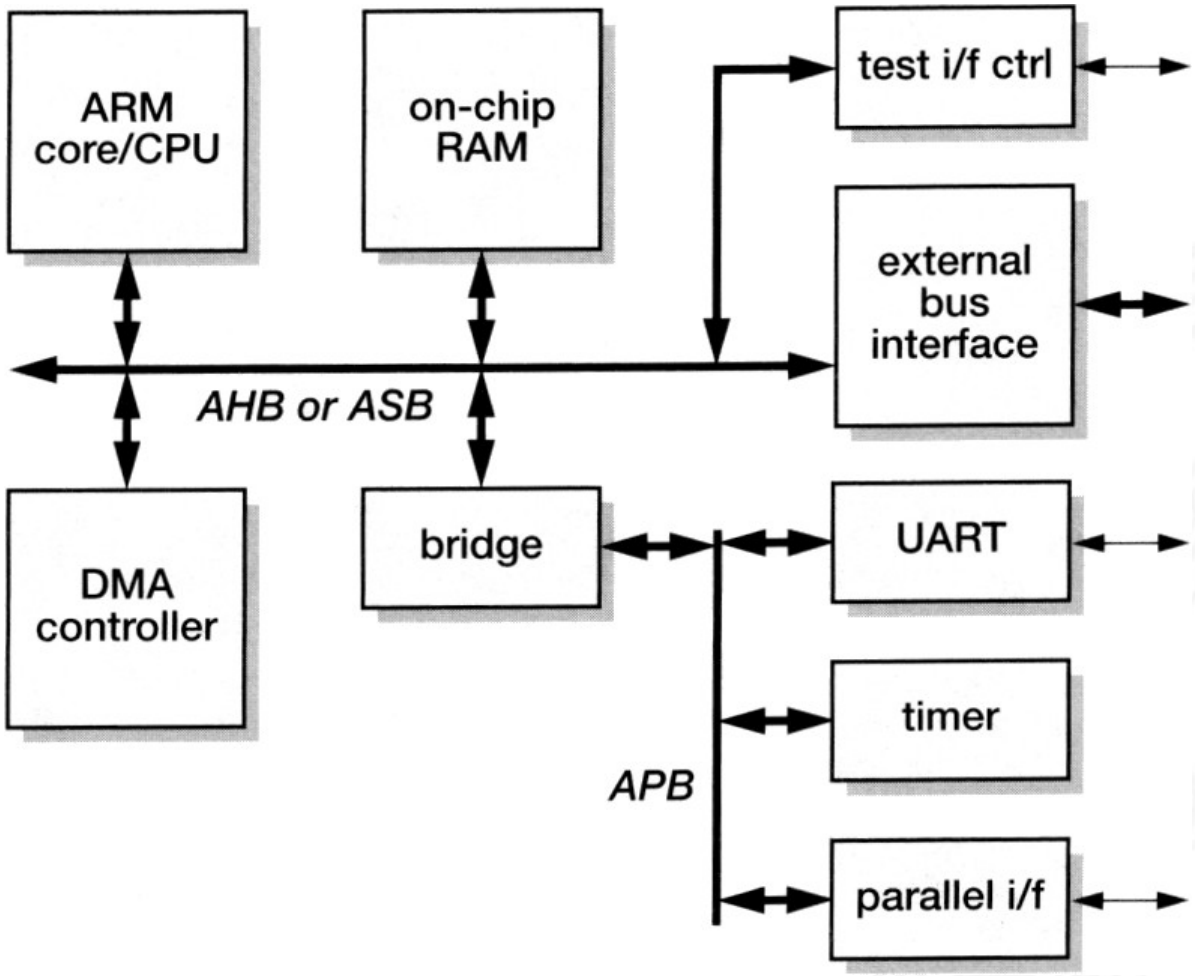
[Tan09]

RS-232: Serielle Schnittstelle



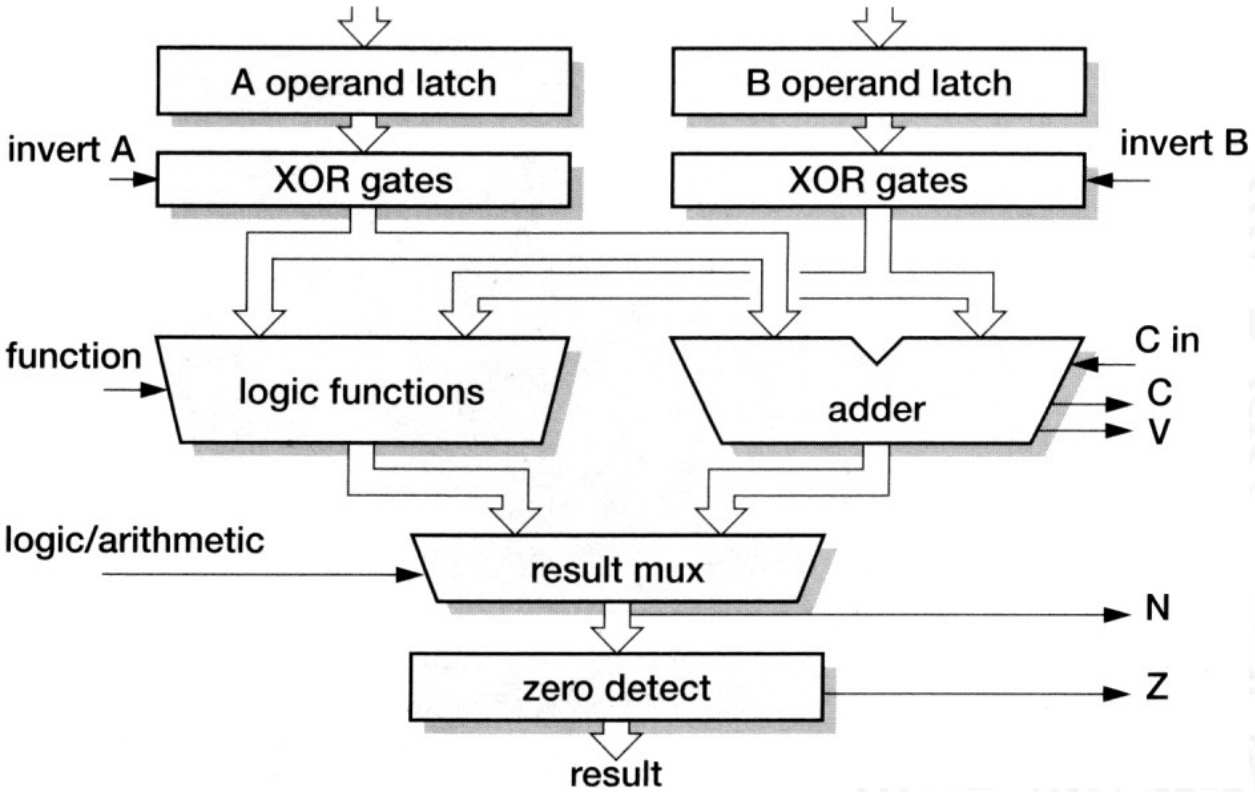
- Baudrate 300, 600, . . . , 19200, 38400, 115200 bits/sec
  Anzahl Datenbits 5, 6, 7, 8
  Anzahl Stopbits 1, 2
  Parität none, odd, even
- minimal drei Leitungen: GND, TX, RX (Masse, Transmit, Receive)
- oft weitere Leitungen für erweitertes Handshake

typisches ARM SoC System



S. Furber: *ARM System-on-Chip Architecture* [Fur01]

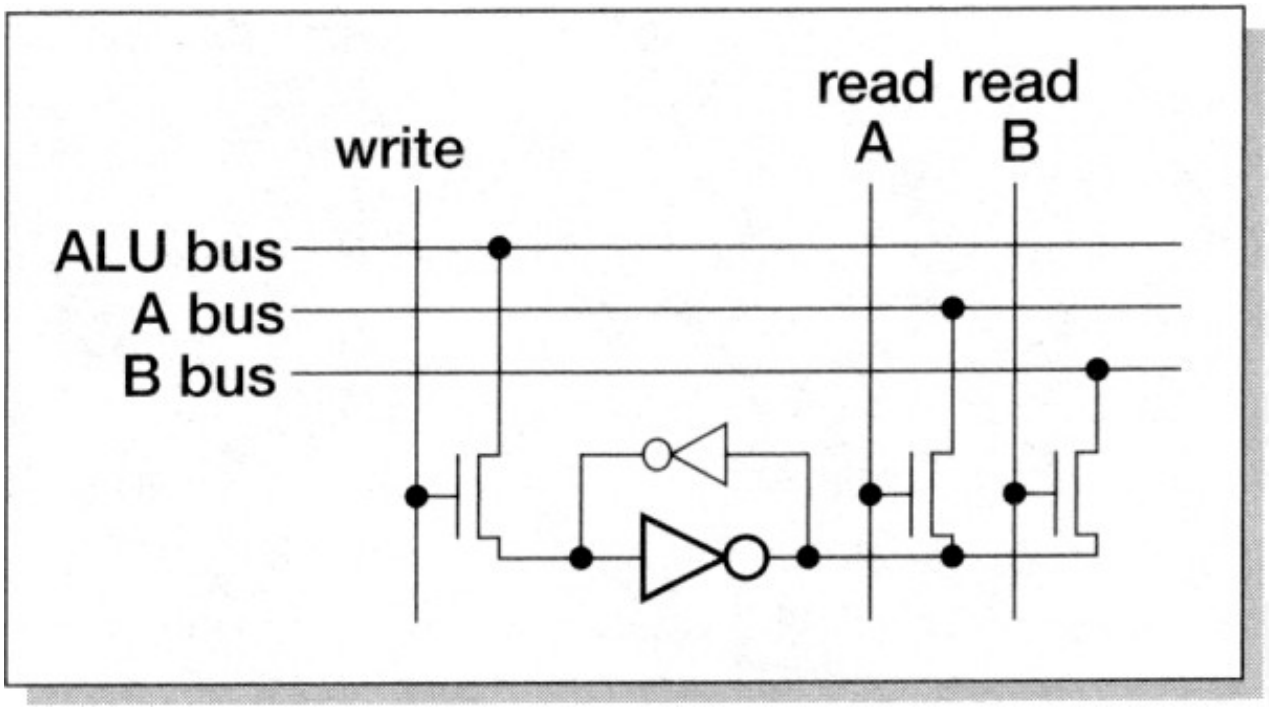
RT-Ebene: ALU des ARM6 Prozessors



[Fur01]

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

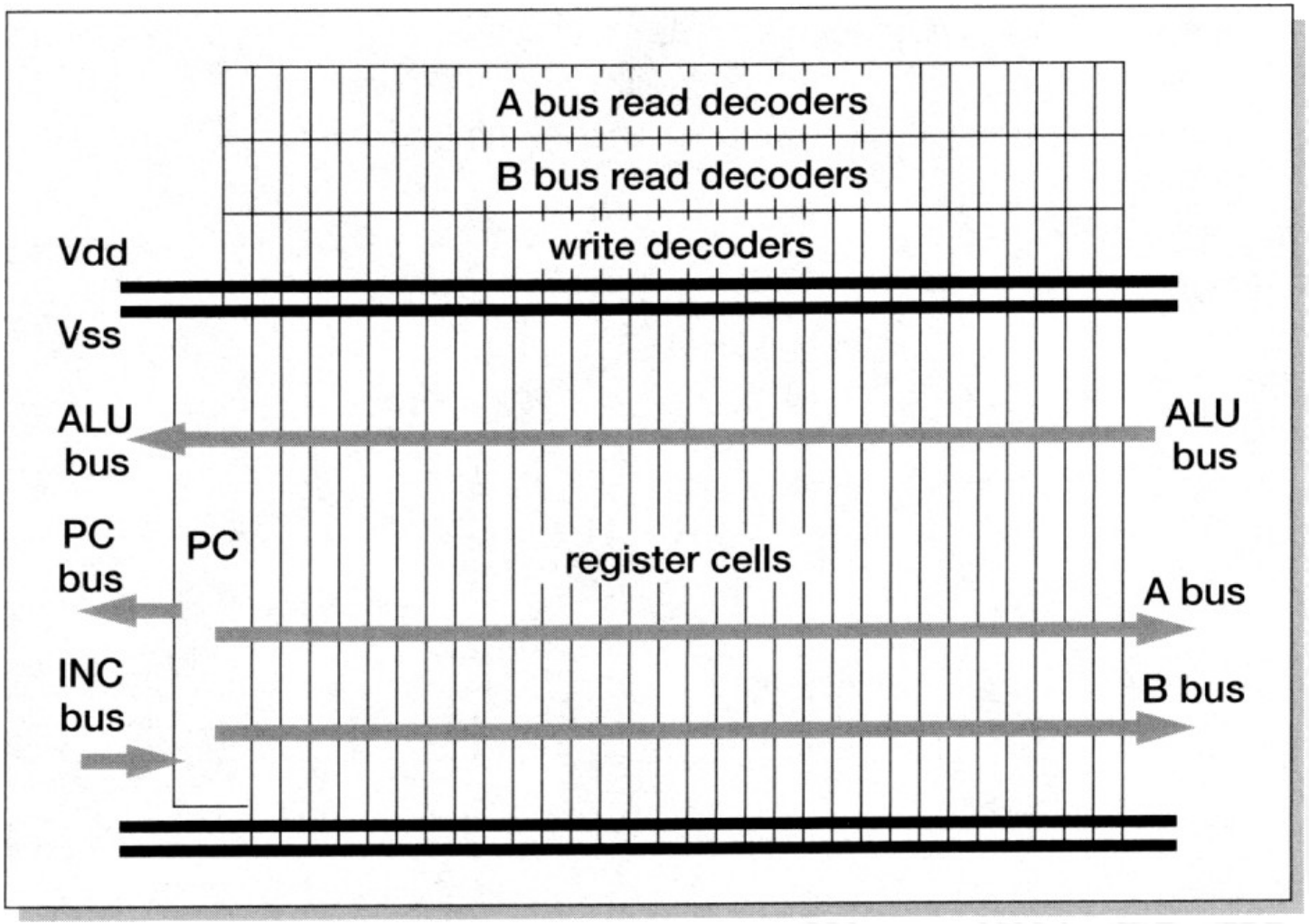
Multi-Port-Registerbank: Zelle



[Fur01]

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

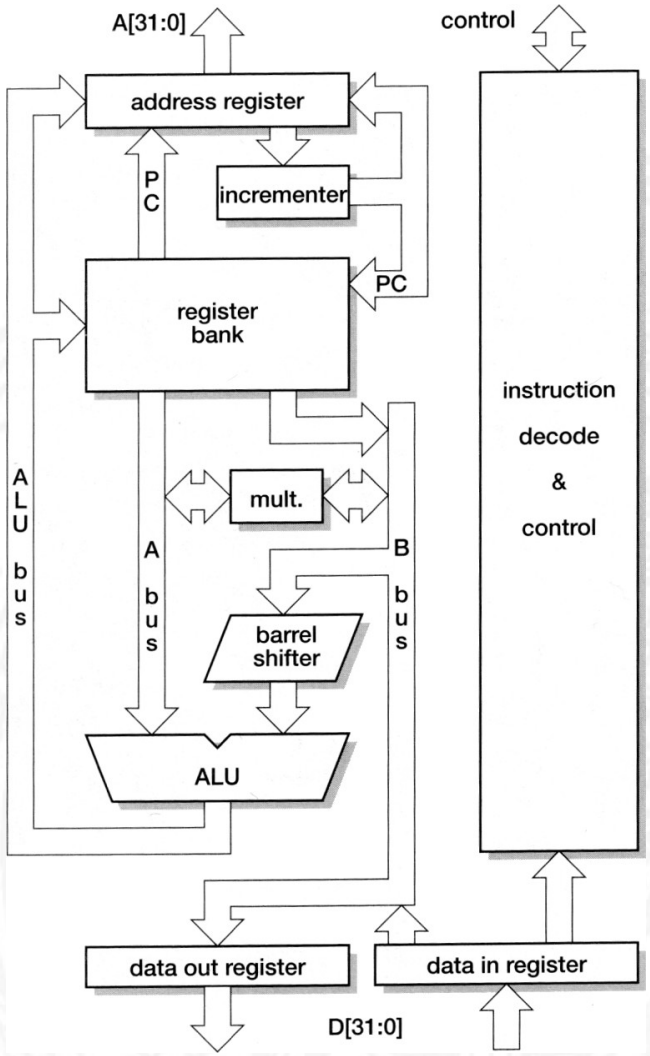
Multi-Port Registerbank: Floorplan/Chiplayout



[Fur01]

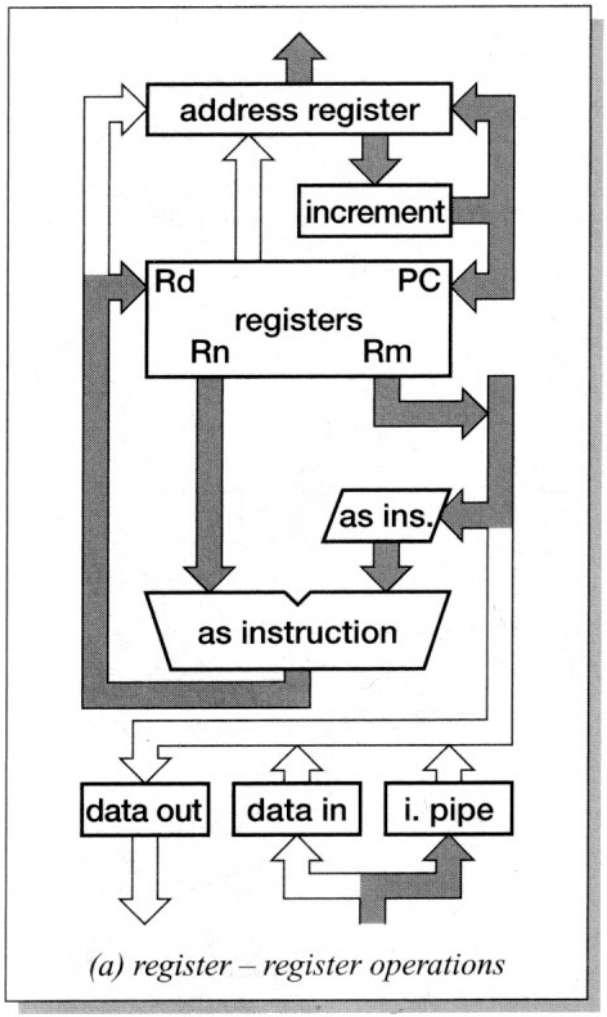
Kompletter Prozessor: ARM

- Registerbank (inkl. Program Counter)
- Inkrementer
- Adress-Register
- ALU, Multiplizierer, Shifter
- Speicherinterface (Data-In / -Out)
- Steuerwerk
- bis ARM7: 3-stufige Pipeline
  fetch, *decode*, *execute*

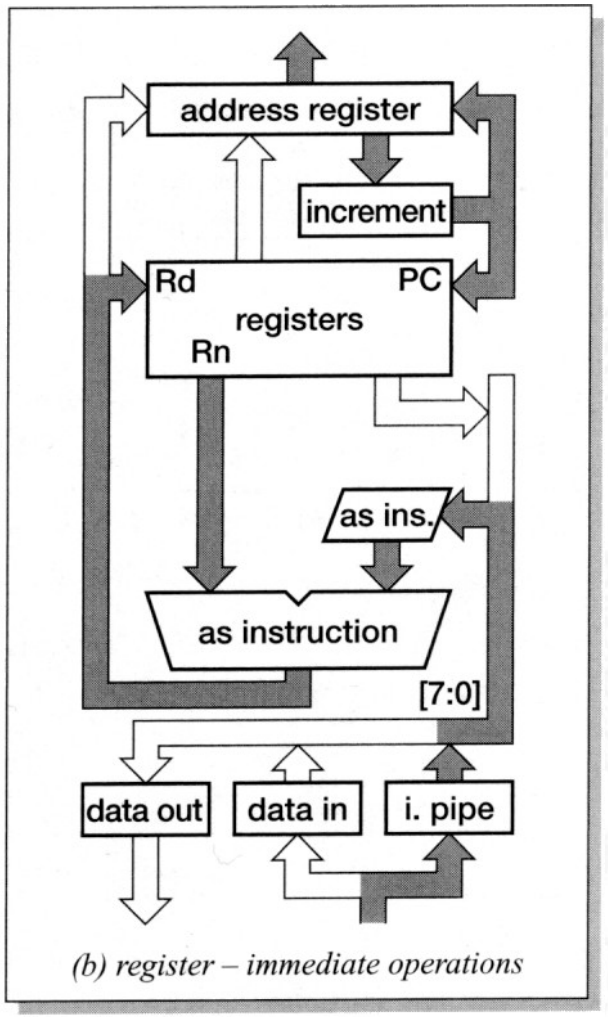


[Fur01]

ARM Datentransfer: Register-Operationen



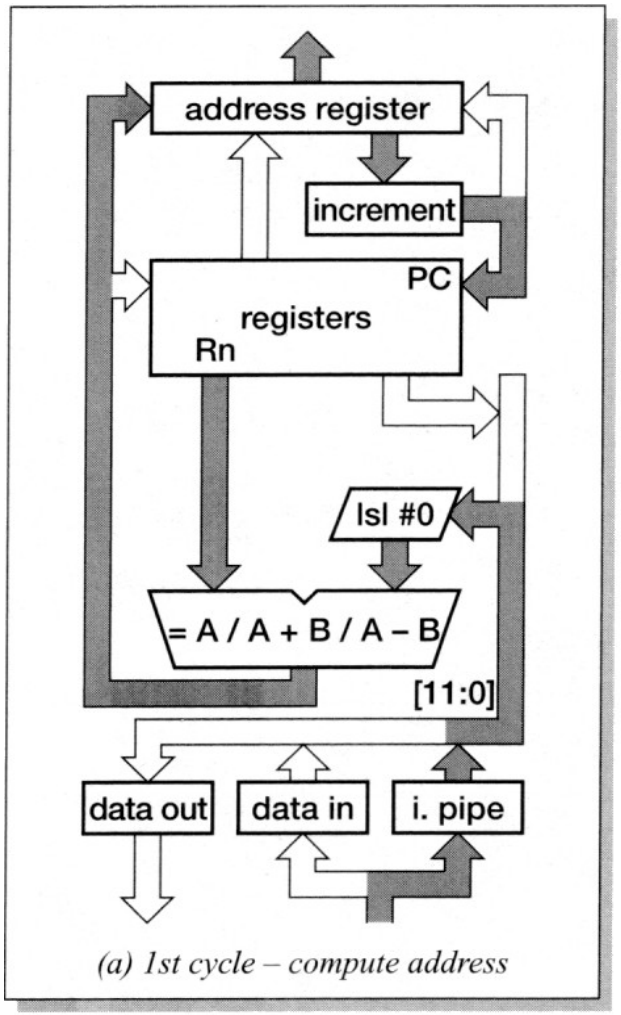
(a) register – register operations



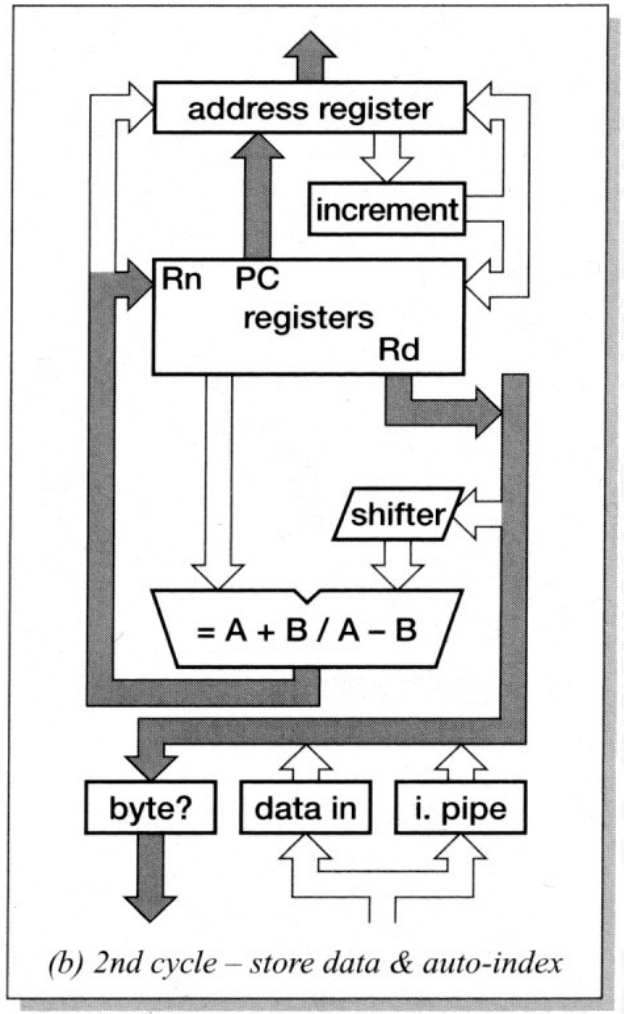
(b) register – immediate operations

[Fur01]

ARM Datentransfer: Store-Befehl



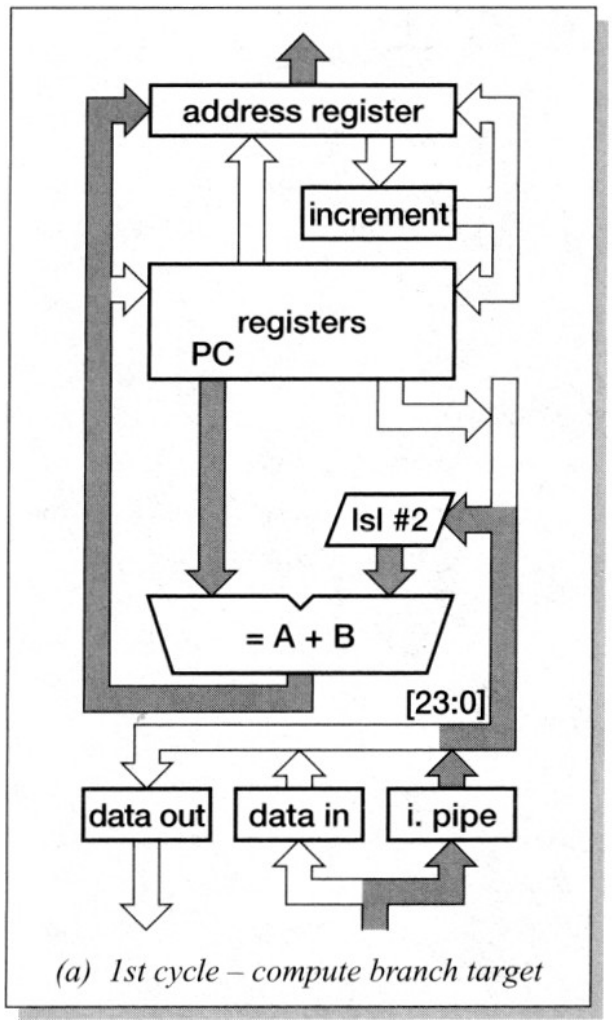
(a) 1st cycle – compute address



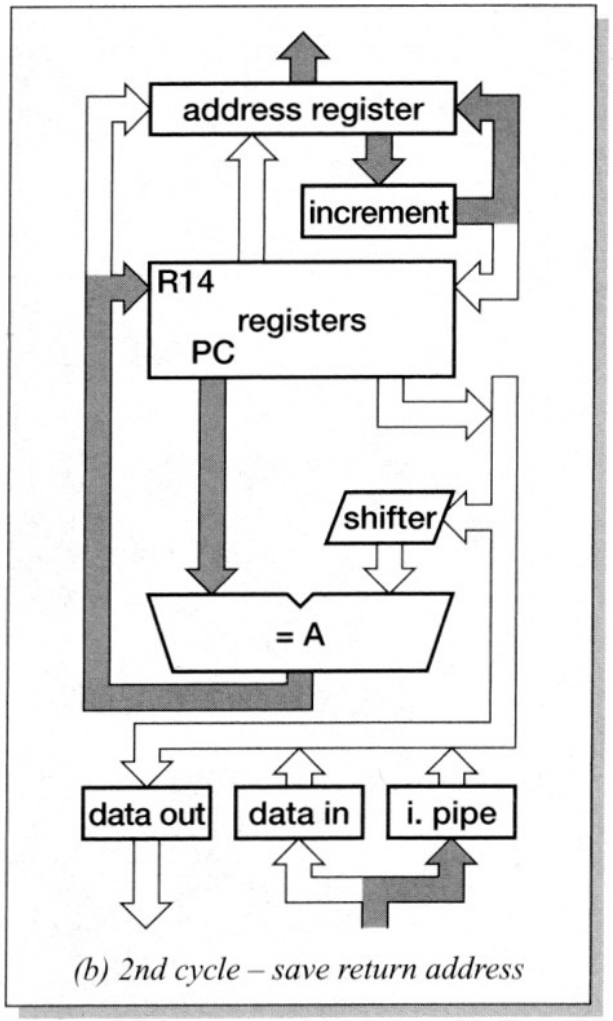
(b) 2nd cycle – store data & auto-index

[Fur01]

ARM Datentransfer: Funktionsaufruf/Sprungbefehl



(a) 1st cycle – compute branch target



(b) 2nd cycle – save return address

[Fur01]

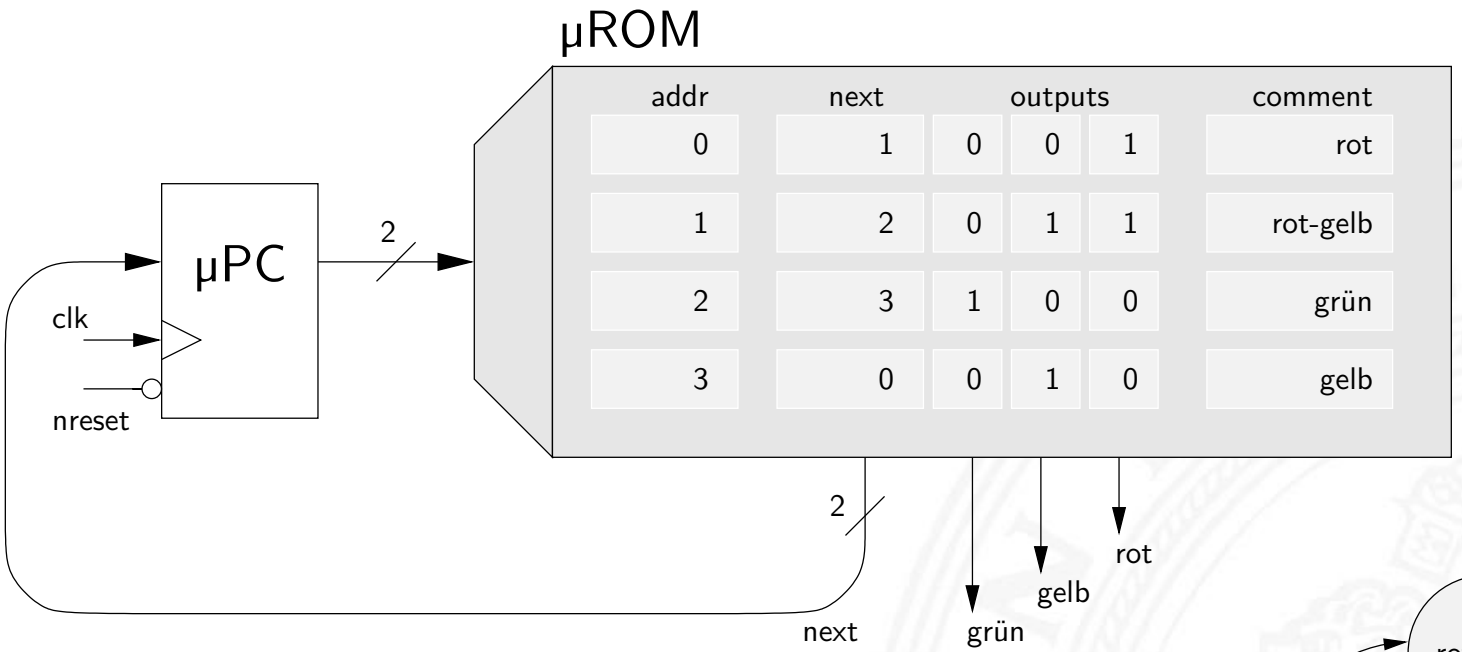
Ablaufsteuerung mit Mikroprogramm

- als Alternative zu direkt entworfenen Schaltwerken

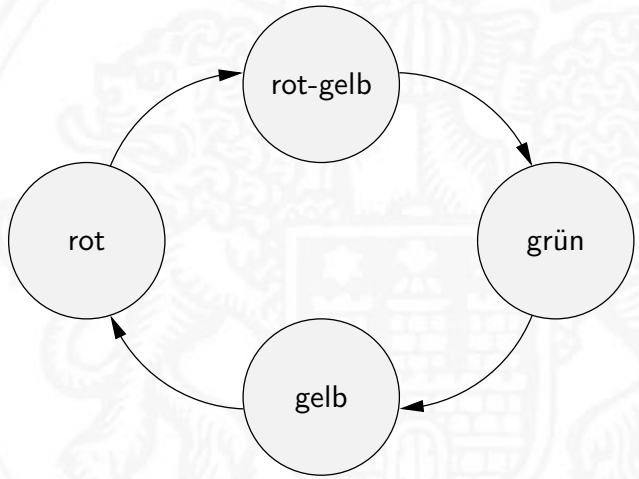
- *Mikroprogrammzähler* μPC: Register für aktuellen Zustand
- μPC adressiert den Mikroprogrammspeicher μROM
- μROM konzeptionell in mehrere Felder eingeteilt
  - die verschiedenen Steuerleitungen
  - ein oder mehrere Felder für Folgezustand
  - ggf. zusätzliche Logik und Multiplexer zur Auswahl unter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

- siehe „Praktikum Rechnerstrukturen“

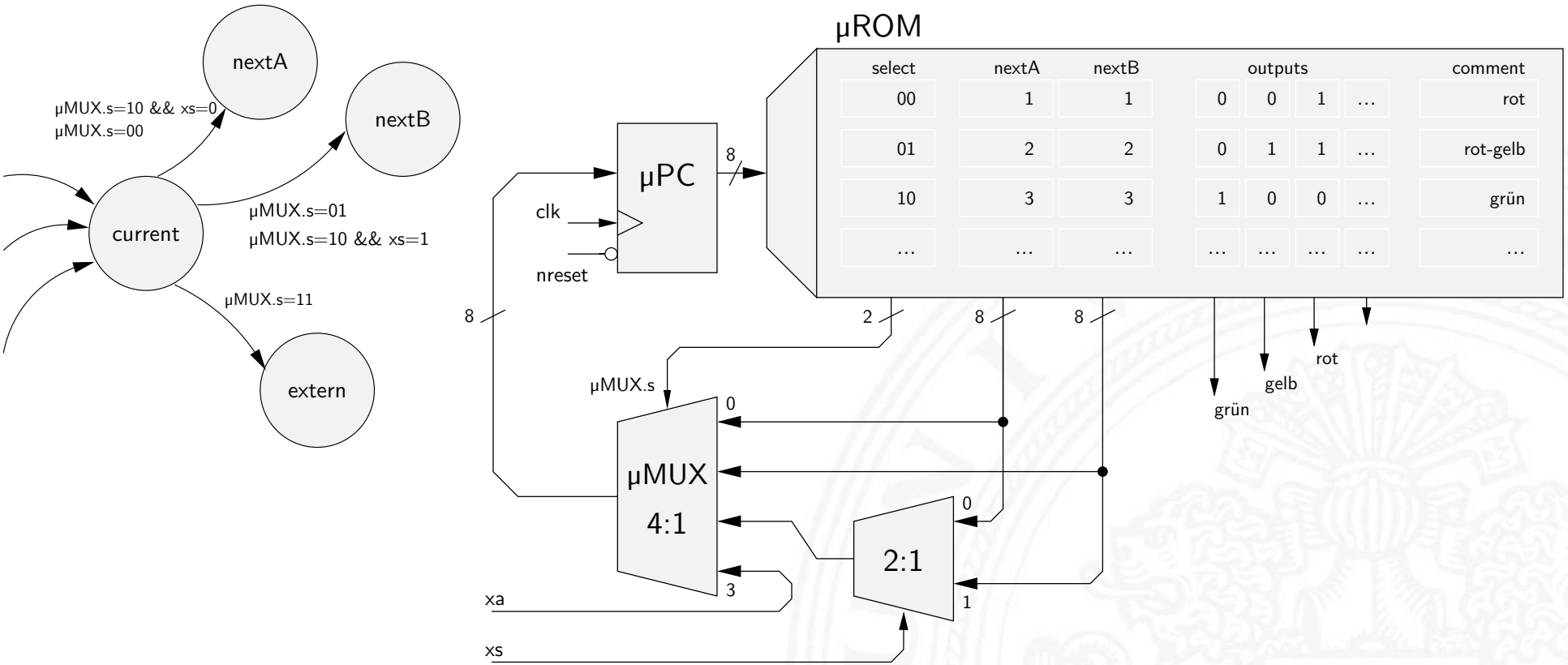
Mikroprogramm: Beispiel Ampel



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

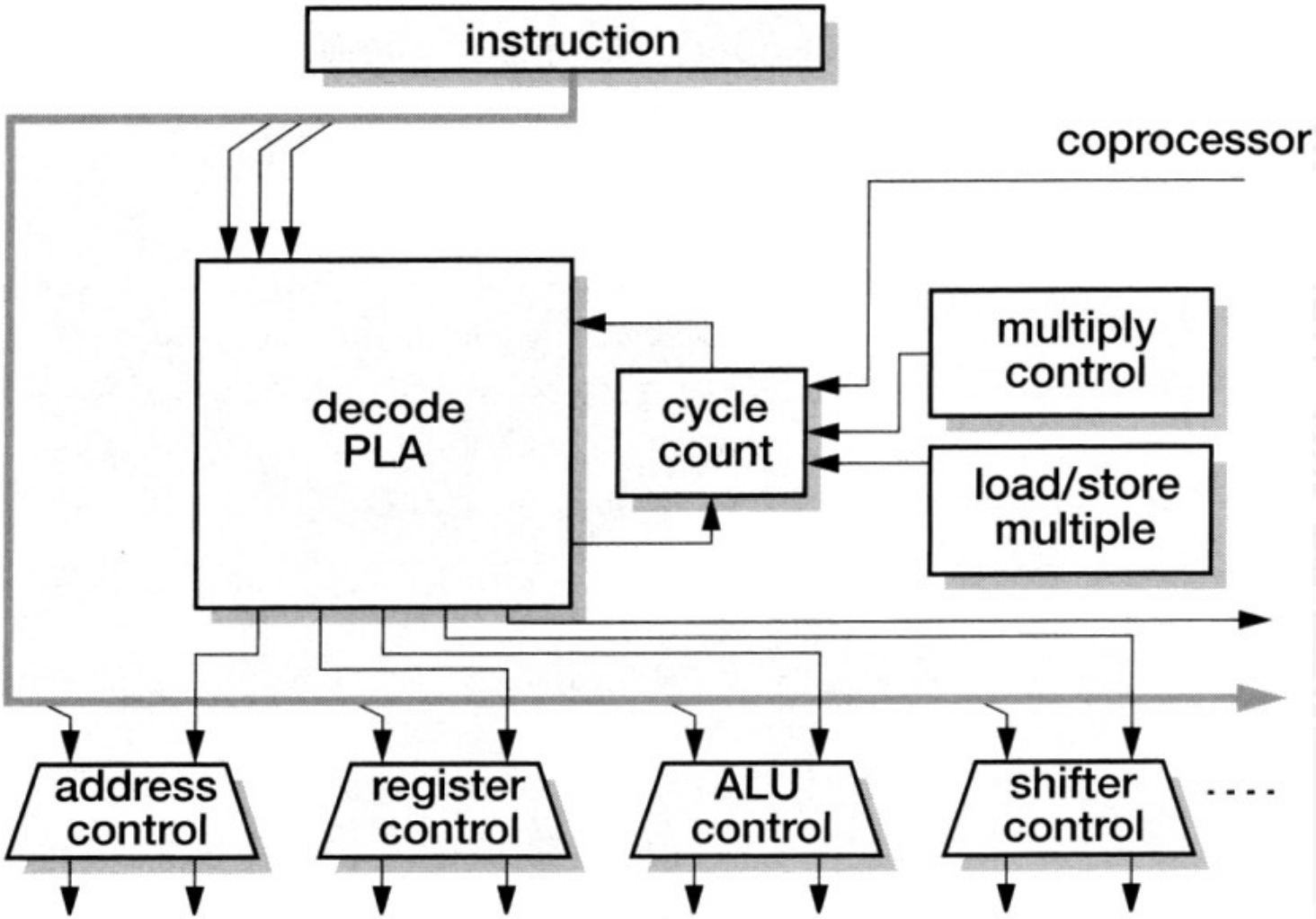


Mikroprogramm: Beispiel zur Auswahl des Folgezustands



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

Mikroprogramm: Befehlsdecoder des ARM Prozessors



[Fur01]

Literatur

[Tan06] A.S. Tanenbaum:
Computerarchitektur – Strukturen, Konzepte, Grundlagen.
5. Auflage, Pearson Studium, 2006. ISBN 3–8273–7151–1

[Tan09] A.S. Tanenbaum: *Structured Computer Organization*.
5th rev. edition, Pearson International, 2009.
ISBN 0–13–509405–4

[Fur01] S. Furber: *ARM System-on-Chip Architecture*.
2nd edition. Addison-Wesley Professional, 2001.
ISBN 978–0–201–67519–1

Literatur (cont.)

[Mäd10] A. Mäder:
Vorlesung: Rechnerarchitektur und Mikrosystemtechnik.
Universität Hamburg, FB Informatik, 2010, Vorlesungsfolien.
tams.informatik.uni-hamburg.de/lectures/2010ws/vorlesung/ram

[Hen] N. Hendrich: *HADES — HAmburg DEsign System*.
Universität Hamburg, FB Informatik, Lehrmaterial.
tams.informatik.uni-hamburg.de/applets/hades