

64-040 Modul IP7: Rechnerstrukturen

9. Register-Transfer Ebene, Integrierte-Schaltungen

Norman Hendrich & Jianwei Zhang

Universität Hamburg
 MIN Fakultät, Department Informatik
 Vogt-Kölln-Str. 30, D-22527 Hamburg
 {hendrich,zhang}@informatik.uni-hamburg.de

WS 2010/2011



Inhalt

Register-Transfer Ebene

Speicherbausteine

Register-Transfer Ebene

Halbleitertechnologie

CMOS-Schaltungen

Programmierbare Logikbausteine

Entwurf Integrierter Schaltungen

Literatur



Motivation: Aufbau kompletter Rechensysteme

- ▶ bisher:
 - ▶ Gatter und Schaltnetze
 - ▶ Flipflops als einzelne Speicherglieder
 - ▶ Schaltwerke zur Ablaufsteuerung

- ▶ jetzt zusätzlich:
 - ▶ Speicher
 - ▶ Register-Transfer-Komponenten eines Rechners
 - ▶ Ablaufsteuerung (Timing, Mikroprogrammierung)
 - ▶ Halbleitertechnologie
 - ▶ CMOS-Schaltungen
 - ▶ Entwurf integrierter Schaltungen

Speicher

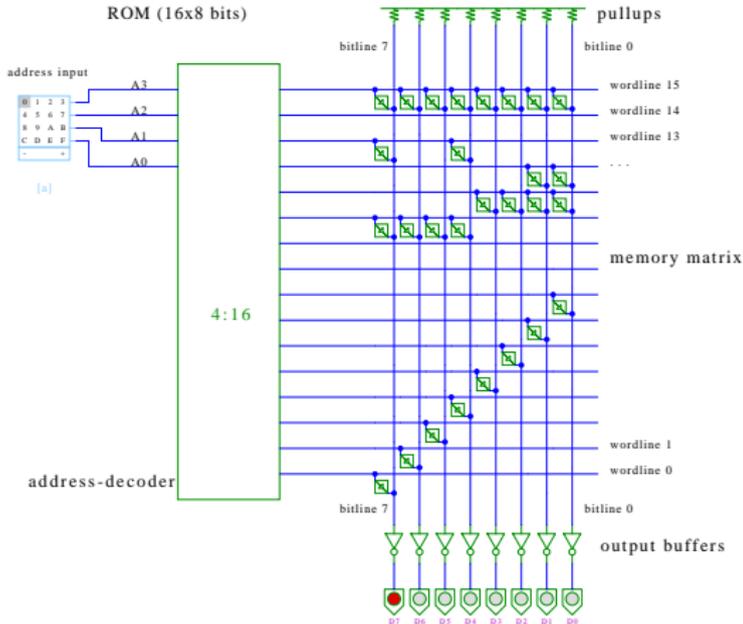
- ▶ System zur Speicherung von Information
- ▶ als Feld von N Adressen mit je m Bit
- ▶ typischerweise mit n -bit Adressen und $N = 2^n$
- ▶ Kapazität also $2^n \times m$ Bits

- ▶ 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

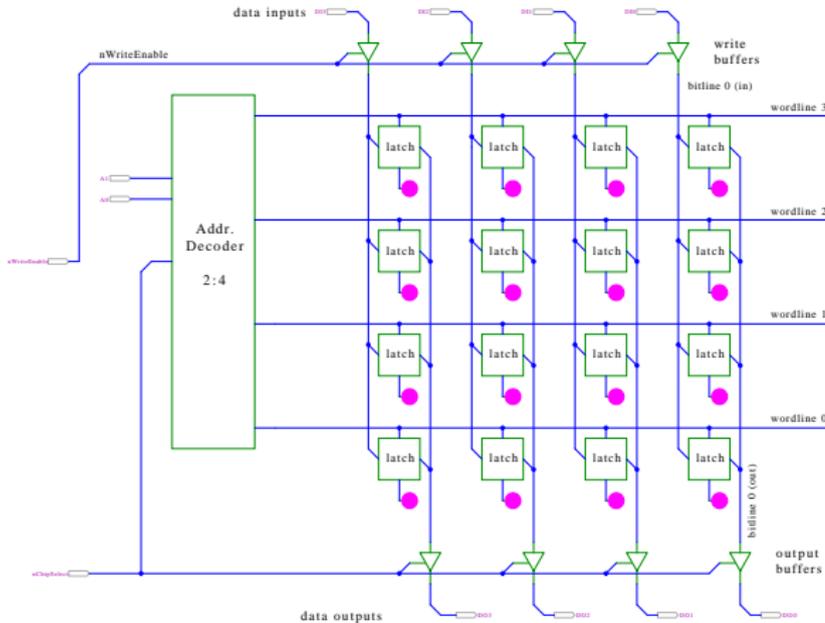
Speicher, der im Betrieb gelesen und geschrieben werden kann

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

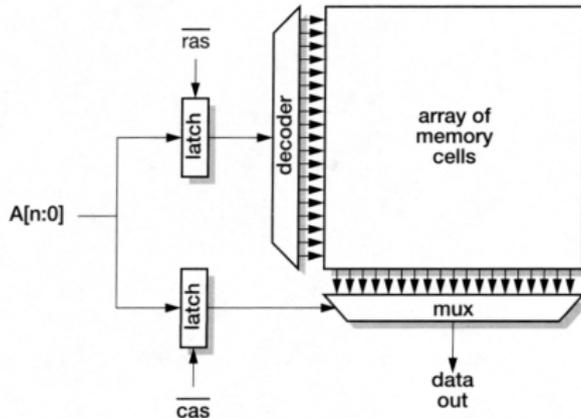
- ▶ Aufbau als Matrixstruktur
- ▶ n Adressbits, konzeptionell 2^n Wortleitungen
- ▶ m Bits pro Wort
- ▶ Realisierung der einzelnen Speicherstellen?
 - ▶ statisches RAM: 6-Transistor Zelle (SRAM)
 - ▶ dynamisches RAM: 1-Transistor Zelle (DRAM)

RAM: Blockschaltbild

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



RAM: RAS/CAS-Adressdekodierung



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

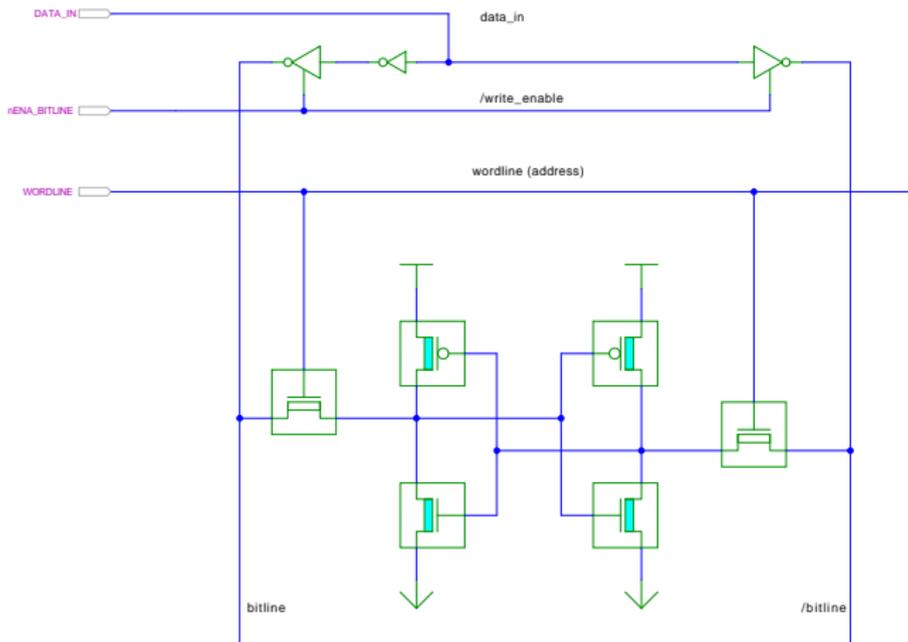
SRAM: statisches RAM

- ▶ Inhalt bleibt dauerhaft gespeichert
- ▶ solange Betriebsspannung anliegt

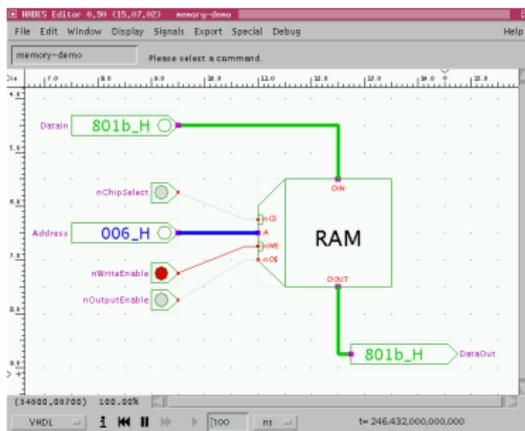
- ▶ *sechs-Transistor* Zelle zur Speicherung
 - ▶ weniger Platzverbrauch als Latches/Flipflops
 - ▶ kompakte Realisierung in CMOS-Technologie (s.u.)
 - ▶ zwei rückgekoppelte Inverter zur Speicherung
 - ▶ zwei n-Transistoren zur Anbindung an die Bitlines

- ▶ schneller Zugriff: Einsatz für Caches
- ▶ deutlich höherer Platzbedarf als DRAMs

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



SRAM: Hades Demo



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



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 werden an gewählte Adresse geschrieben
 - ▶ \overline{OE} : output-enable: Inhalt des Speichers wird ausgegeben

- ▶ interaktive Hades-Demo zum Ausprobieren

DRAM: dynamisches RAM

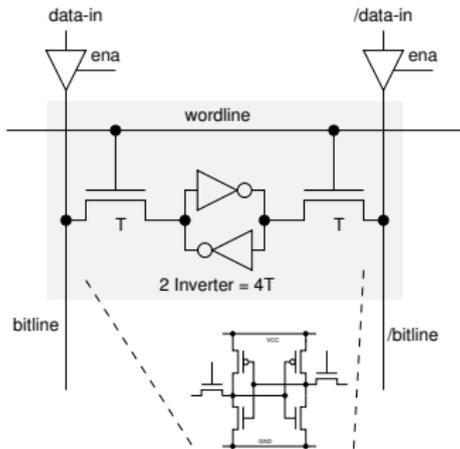
- ▶ Information wird in winzigen Kondensatoren gespeichert
- ▶ pro Bit je ein Transistor und Kondensator

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

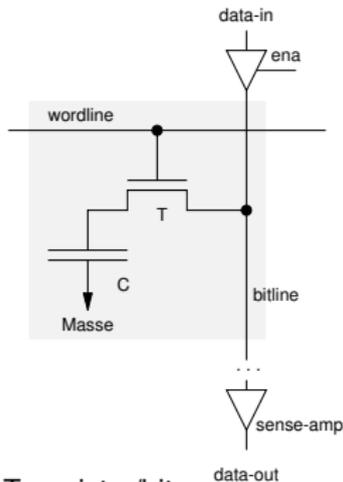
- ▶ Information muss anschließend neu geschrieben werden
- ▶ auch ohne Lese- oder Schreibzugriff ist regelmässiger *Refresh* notwendig (Millisekunden)

- ▶ DRAM wird für hohe Kapazität optimiert
- ▶ minimaler Platzbedarf, aber ca. 10X langsamer als SRAM

DRAM: vs SRAM



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



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

DRAM: Stacked- und Trench-Zelle

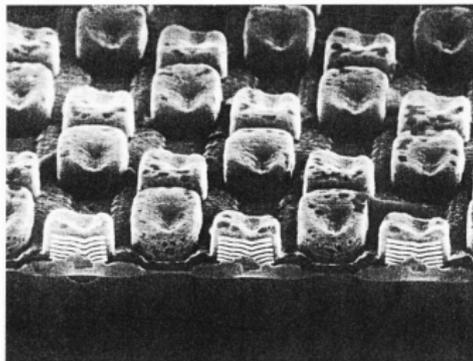
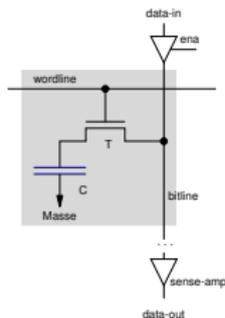
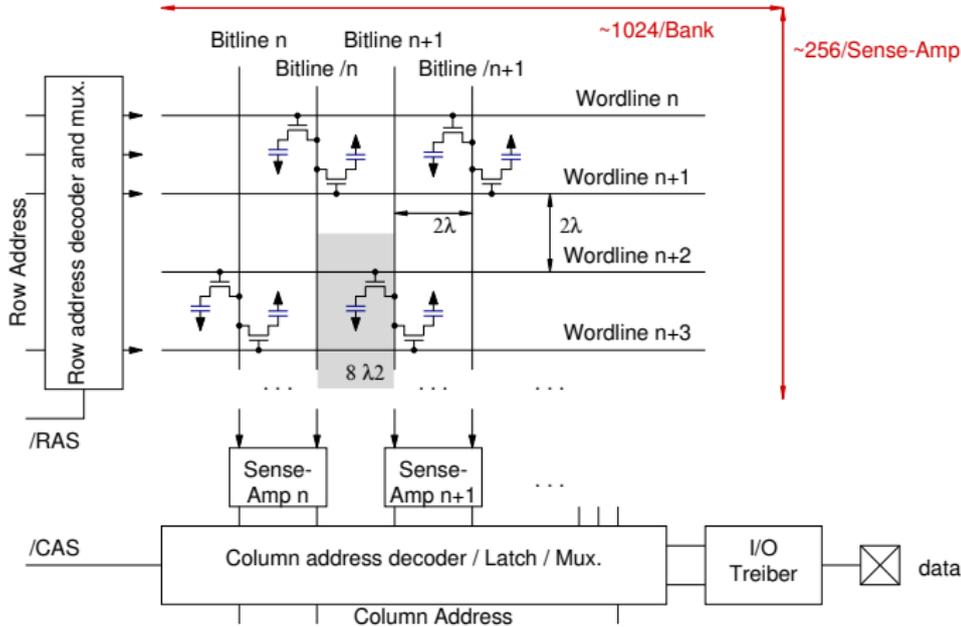


Abb. 7: Prototyp von Speicherzellen (Stapelkondensatoren) für zukünftige Speicherchips wie den Ein-Gigabit-Chip. Da für DRAM-Chips eine minimale Speicherkapazität von 25 fF notwendig ist, bringt es erhebliche Platzvorteile, die Kondensatorelemente vertikal übereinander zu stapeln. Die Dicke der Schichten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



- ▶ möglichst kleine Fläche der Kondensatoren, Kapazität gerade ausreichend
- ▶ zwei Bauformen: „stacked“ und „trench“
 (Siemens 1 Gbit DRAM) (IBM CMOS-6X embedded DRAM)
- ▶ jeweils $C \approx 10\text{fF}$: etwa 200.000 Elektronen

DRAM: Layout

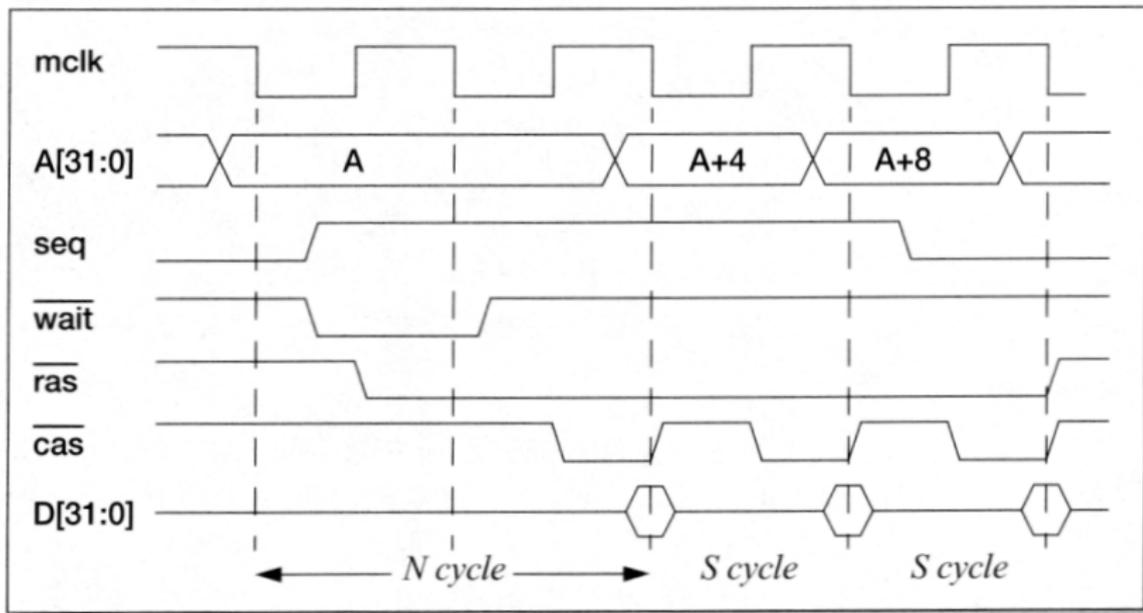


DRAM: Varianten

- ▶ 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, aber Daten werden mit steigender und fallender Taktflanke übertragen
 - ▶ DDR-2, DDR-3: Varianten von DDR mit höherer Taktrate
 - ▶ aktuell Übertragungsraten bis ca. 6 GByte/sec

SDRAM: Lesezugriff auf sequentielle Adressen

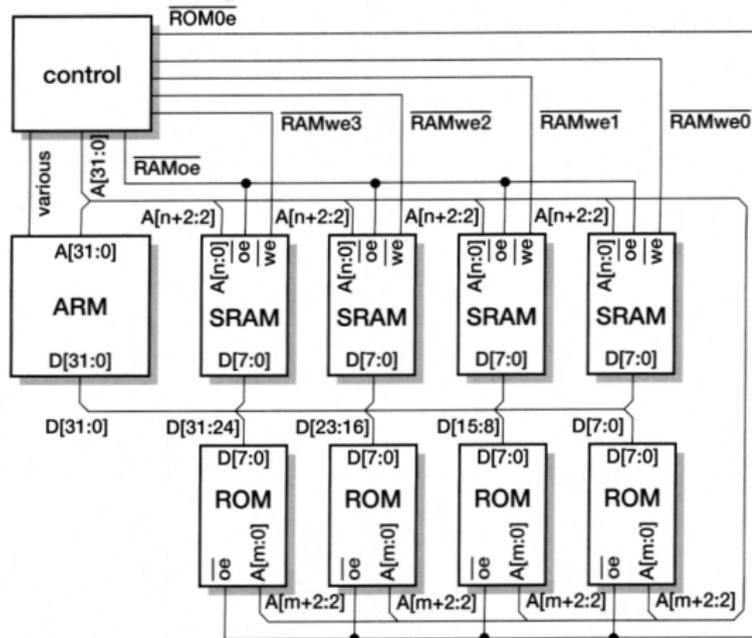


Flash

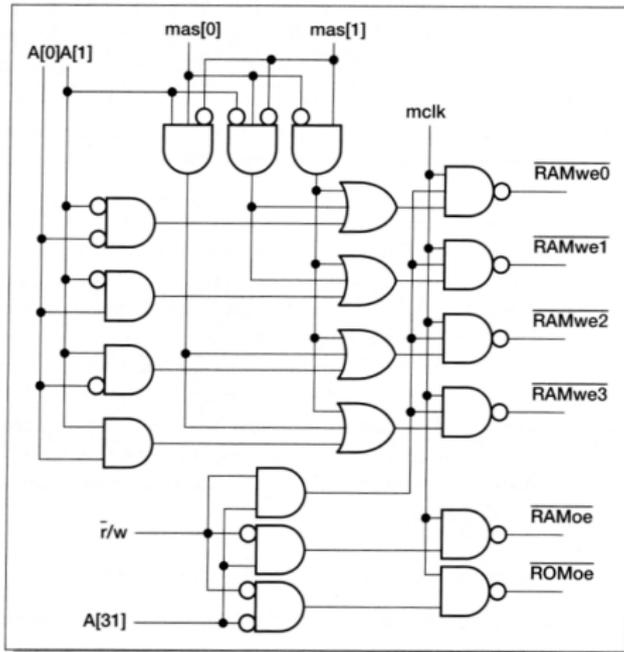
- ▶ ähnlich kompakt und kostengünstig wie DRAM
- ▶ non-volatile: Information bleibt beim Ausschalten erhalten
- ▶ spezielle *floating-gate* Transistoren
 - ▶ das *floating-gate* ist komplett nach außen isoliert
 - ▶ einmal gespeicherte Elektronen sitzen dort fest
- ▶ Auslesen beliebig oft möglich, schnell
- ▶ Schreibzugriffe problematisch
 - ▶ intern hohe Spannung zum Überwinden der Isolierung des *floating-gate* erforderlich
 - ▶ Schreibzugriffe einer „0“ nur blockweise
 - ▶ pro Zelle nur einige 10.000 .. 100.000 Schreibzugriffe möglich

Typisches Speichersystem

32-bit Prozessor, je 4 8-bit SRAMs und ROMs



Typisches Speichersystem: Adressdekodierung



Inhalt

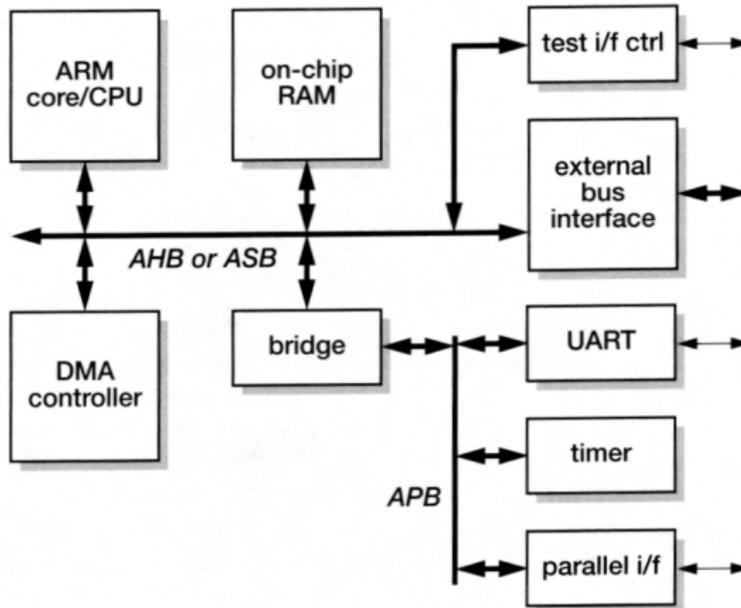
- ▶ Speicher
- ▶ Register-Transfer-Ebene
- ▶ Halbleitertechnologie
- ▶ CMOS-Schaltungen
- ▶ Entwurf Integrierter Schaltungen

Register-Transfer-Ebene

Modellierung eines digitalen Systems als Schaltung aus

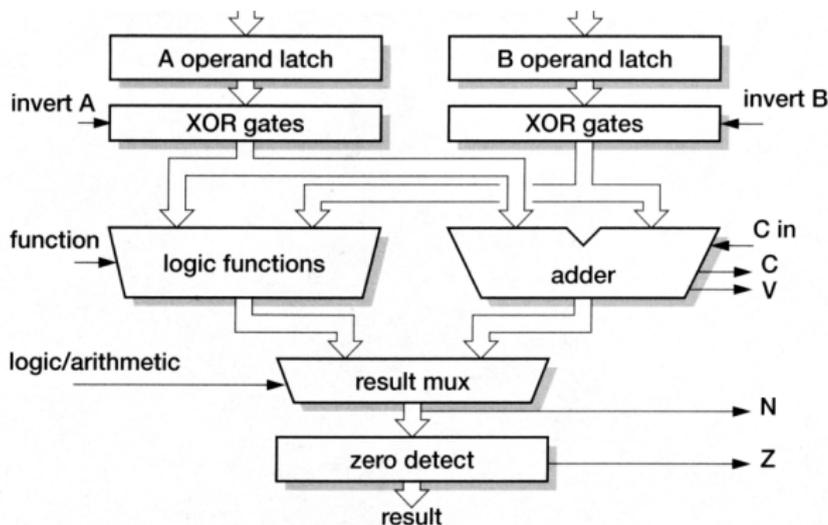
- ▶ Speichergliedern:
 - ▶ Registern (Flipflops, Register, Registerbank)
 - ▶ Speichern (SRAM, DRAM, ROM, PLA)
- ▶ Rechenwerken:
 - ▶ Addierer, arithmetische Schaltungen
 - ▶ logische Operationen
 - ▶ „random-logic“ Schaltnetzen
- ▶ Verbindungsleitungen:
 - ▶ Busse / Leitungsbündel
 - ▶ Multiplexer und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

Hauptblockebene: typisches ARM SoC System



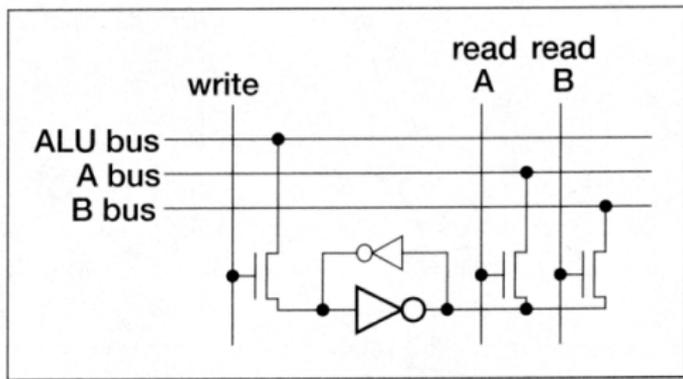
diese und viele folgende Abbildungen: (Furber, *ARM Soc Architecture*)

RT-Ebene: ALU des ARM-7 Prozessors



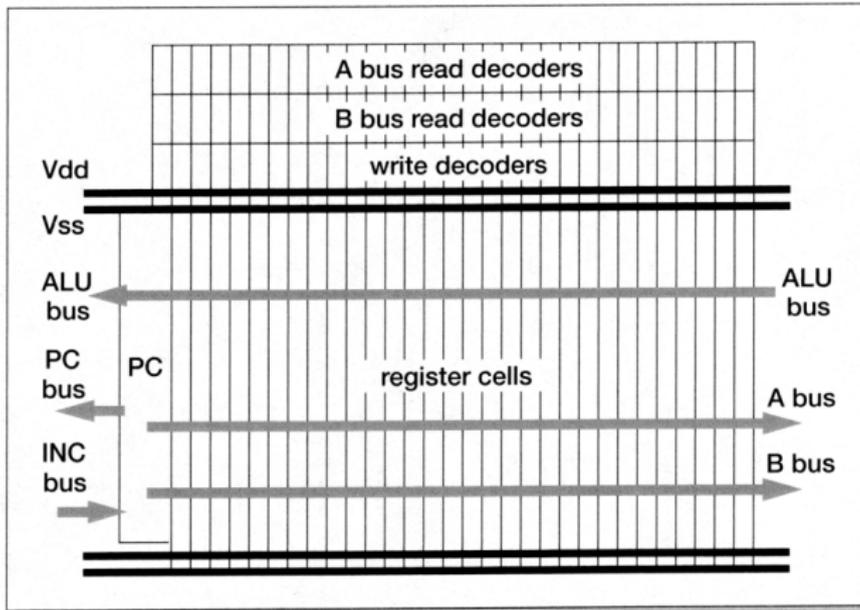
- ▶ Register für die Operanden A und B
- ▶ Addierer und separater Block für logische Operationen

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

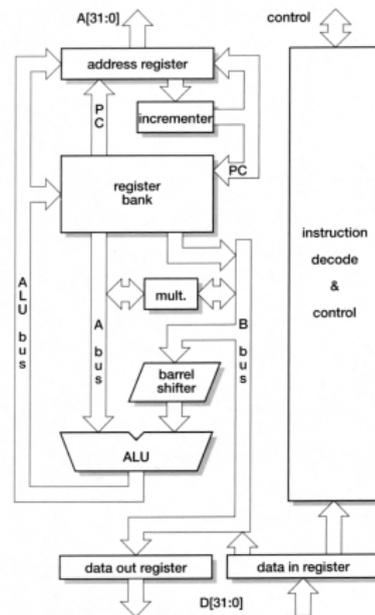


Kompletter Prozessor: ARM-3

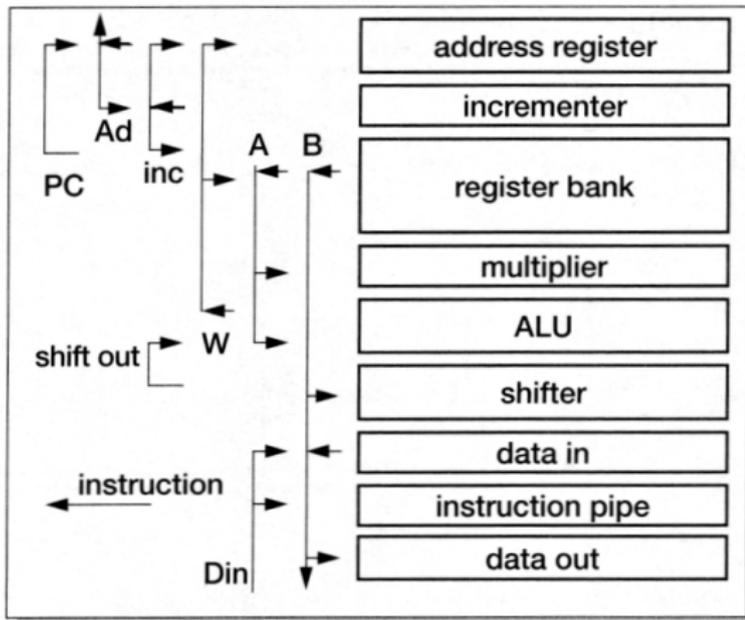
- ▶ Registerbank (inkl. Program Counter)
- ▶ Incrementer
- ▶ Adress-Register

- ▶ ALU, Multiplizierer, Shifter
- ▶ Speicherinterface (Data-In, Data-Out)

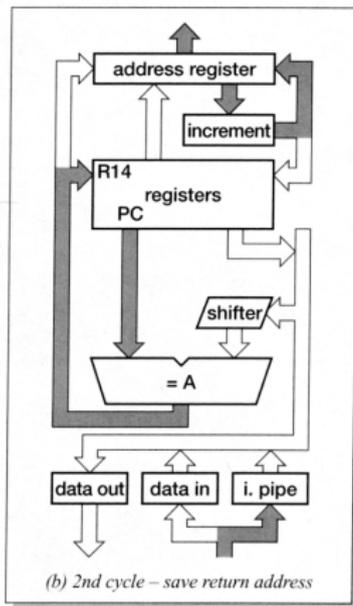
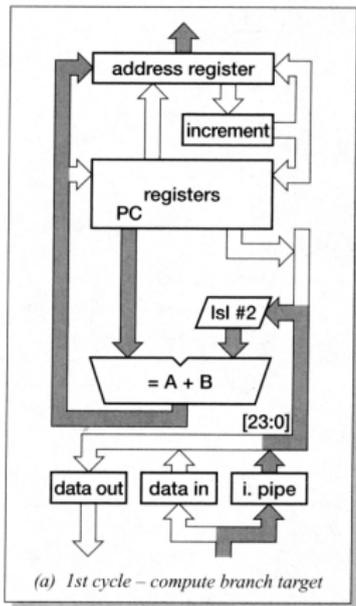
- ▶ Steuerwerk



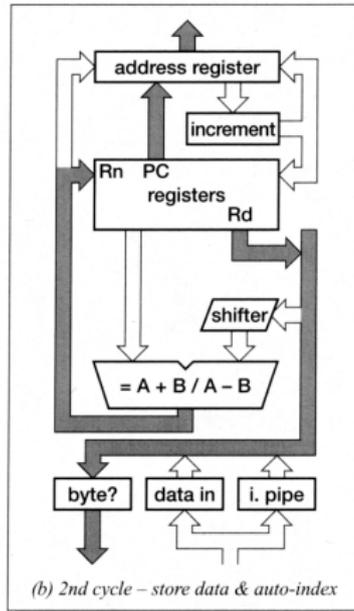
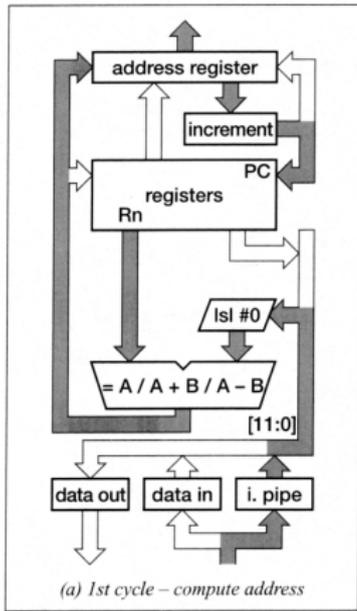
Floorplan ARM-3 Prozessor



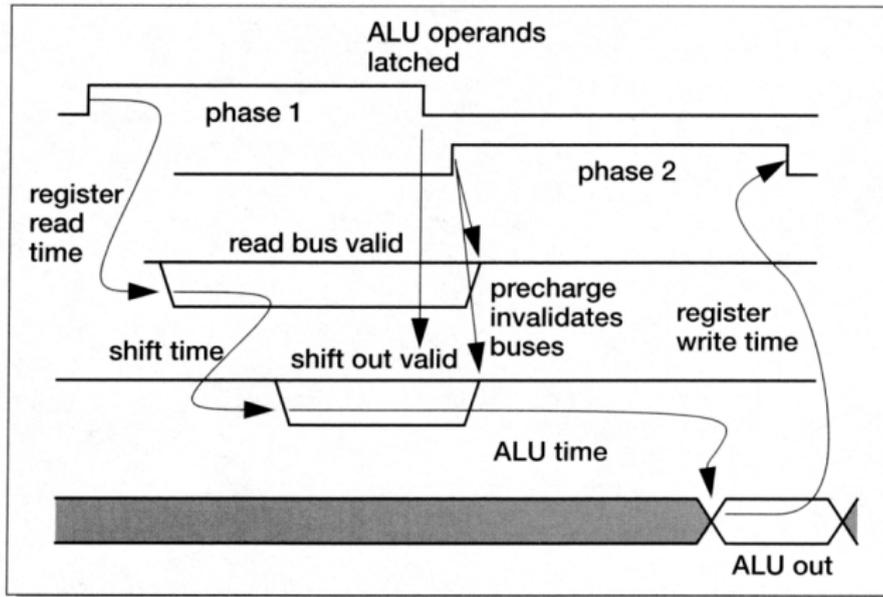
ARM-3 Datentransfers: Funktionsaufruf/Sprungbefehl



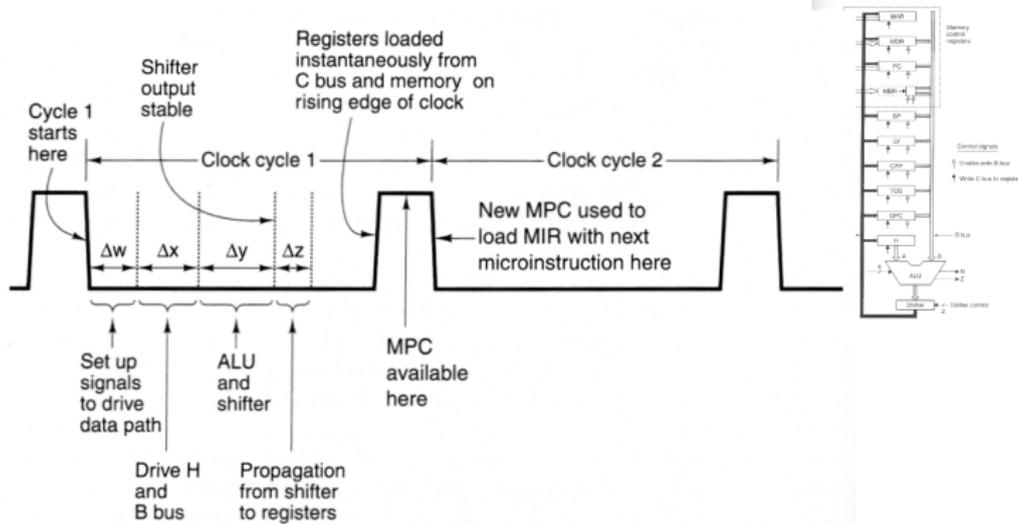
ARM-3 Datentransfers: Store-Befehl



ARM-3 Datentransfers: Timing



IJVM Java-Prozessor: Timing

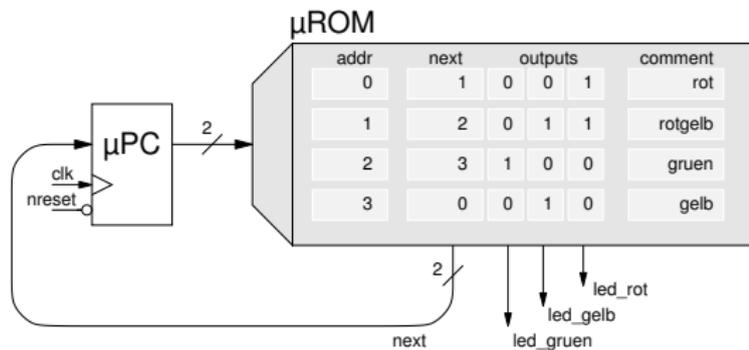


- ▶ Datenfluss: Register → BUS → ALU → Shifter → Bus → Register
- ▶ Details: Tanenbaum, *Structured Computer Organization*, 4.1.1

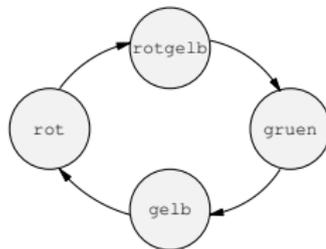
Ablaufsteuerung mit Mikroprogramm

- ▶ 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

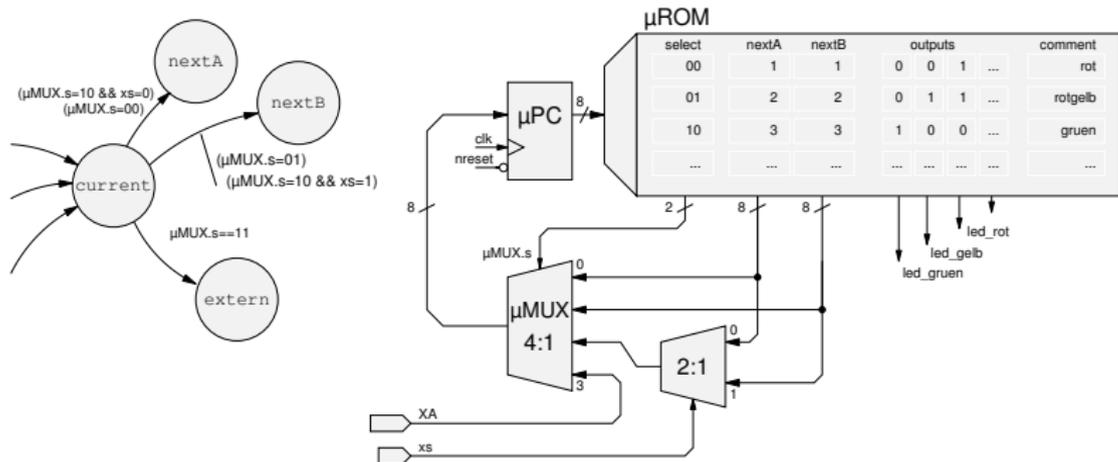
Mikroprogramm: Beispiel Ampel



- μ PC adressiert das μ ROM
- "next"-Ausgang liefert den Folgezustand (Adresse 0: Wert 1, Adresse 1: Wert 2, usw)
- andere Ausgänge steuern die Schaltung (hier die Lampen der Ampel)

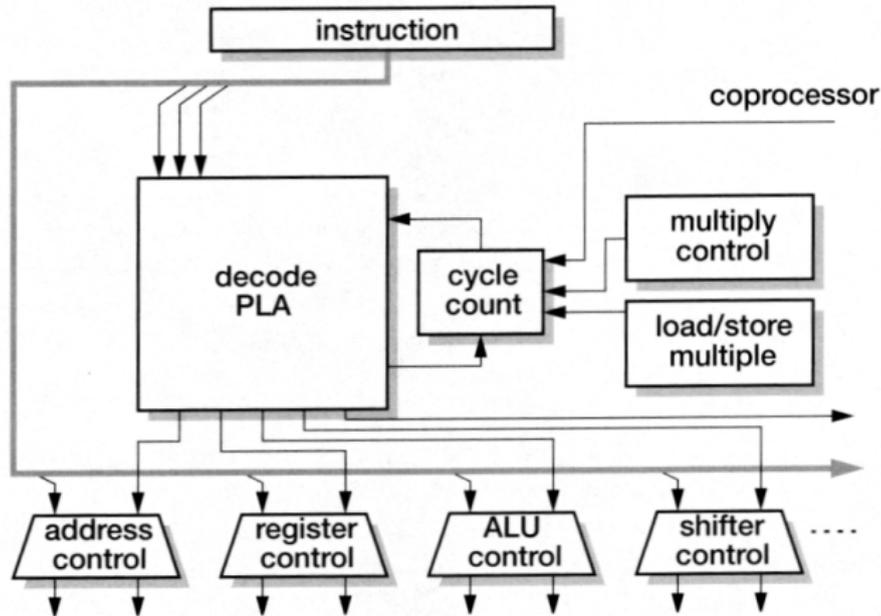


Mikroprogramm: Beispiel zur Auswahl des Folgezustands



- Multiplexer erlaubt Auswahl des μ PC Werts
- "nextA", "nextB" aus dem μ ROM, externer "XA" Wert
- "xs" Eingang erlaubt bedingte Sprünge

Mikrogramm: Befehlsdekoder des ARM-7 Prozessors



Bus: elektrische und logische Verbindung

- ▶ zwischen **mehreren** Geräten
- ▶ oder mehreren Blöcken innerhalb einer Schaltung

- ▶ Bündel aus Daten- und Steuersignalen
 - ▶ elektrische Realisierung: Tri-State-Treiber oder Open-Drain

- ▶ Bus-Arbitrierung: wer darf wann wie-lange senden?
- ▶ Master-Slave oder gleichberechtigte Knoten
- ▶ synchron: mit globalem Taktsignal vom „Master“-Knoten
- ▶ asynchron: Wechsel von Steuersignalen löst Ereignisse aus

Bus: Mikroprozessorsysteme

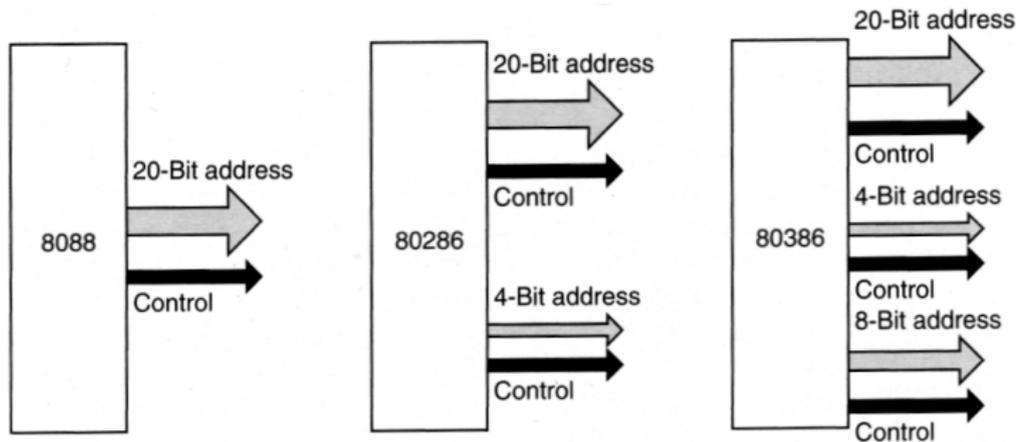
typisches n -bit Mikroprozessor-System:

- ▶ 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
 - ▶ ...

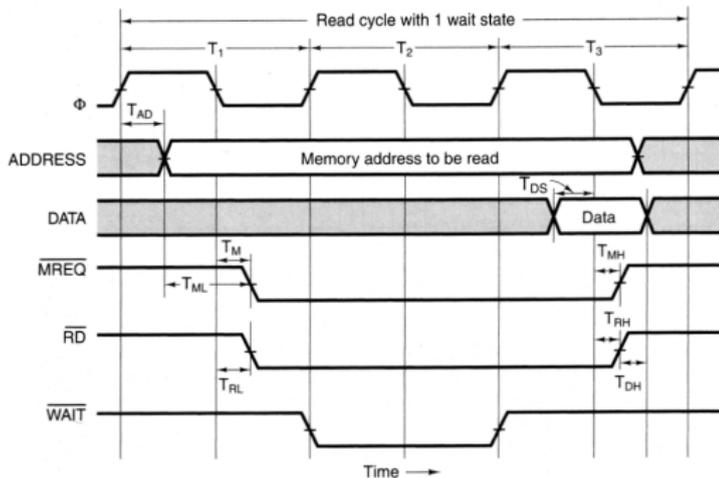
- ▶ um Leitungen zu sparen, teilweise gemeinsam genutzte Leitungen sowohl für Adressen als auch Daten. Zusätzliches Steuersignal zur Auswahl Adressen/Daten

Adressbus: Evolution beim Intel x86



- ▶ 20-bit: 1 MByte Adressraum, 24-bit: 16 MByte, 32-bit: 4 GByte
- ▶ alle Erweiterungen abwärtskompatibel

Synchroner Bus: Timing

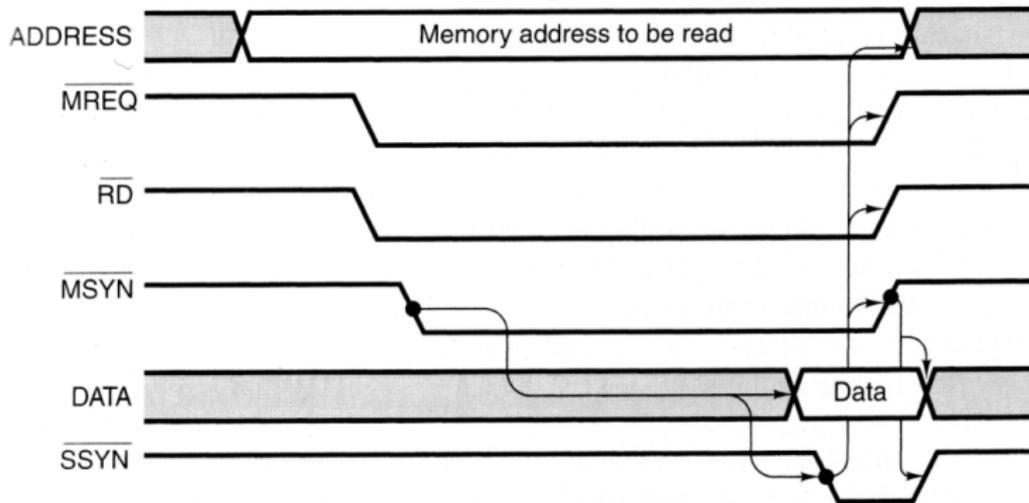


- ▶ alle Zeiten über Taktsignal Φ gesteuert
- ▶ $\overline{\text{MREQ}}$ -Signal zur Auswahl Speicher oder I/O-Geräte
- ▶ $\overline{\text{RD}}$ signalisiert Lesezugriff
- ▶ Wartezyklen, solange der Speicher $\overline{\text{WAIT}}$ aktiviert

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

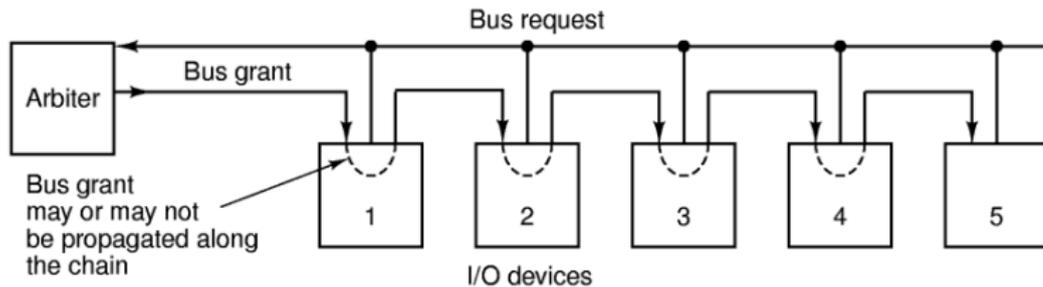


- ▶ Steuersignale: \overline{MSYN} : Master fertig, \overline{SSYN} : Slave fertig
- ▶ flexibler für Geräte mit stark unterschiedlichen Zugriffszeiten

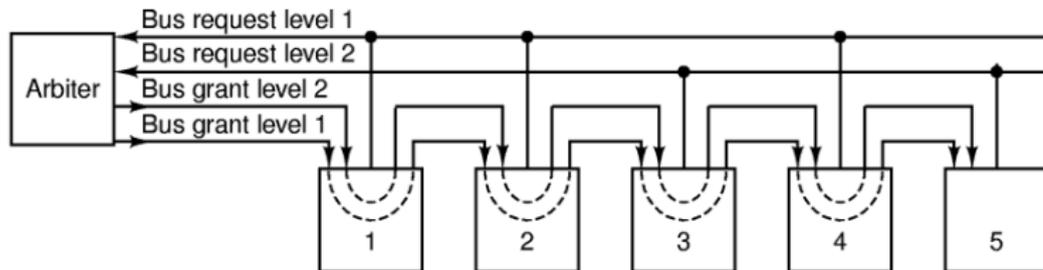
Bus-Arbitrierung

- ▶ immer nur ein Transfer zur Zeit möglich
- ▶ mehrere sendende Geräte müssen aufeinander warten
- ▶ diverse Strategien:
 - ▶ Prioritäten für verschiedene Geräte
 - ▶ „round-robin“ Verfahren
 - ▶ „Token“-basierte Verfahren
 - ▶ usw.
- ▶ I/O-Geräte oft höher priorisiert als die CPU
 - ▶ I/O-Zugriffe müssen schnell/sofort behandelt werden
 - ▶ Benutzerprogramm kann warten

Bus: Arbitrierung



(a)



(b)

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 (Linux)

```
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: Konfiguration einiger Geräte (Linux)

```
tams12> /sbin/lspci -v
00:00.0 Host bridge: Intel Corporation 82Q963/Q965 Memory Controller Hub (rev 02)
  Flags: bus master, fast devsel, latency 0

00:1a.0 USB Controller: Intel Corporation 82801H (ICH8 Family) USB UHCI #4 (rev 02) (prog-if 00 [UHCI])
  Flags: bus master, medium devsel, latency 0, IRQ 169
  I/O ports at ff20 [size=32]

00:1f.2 IDE interface: Intel Corporation 82801H (ICH8 Family) 4 port SATA IDE Controller (rev 02)
  (prog-if 8f [Master SecP Sec0 PriP Pri0])
  Flags: bus master, 66MHz, medium devsel, latency 0, IRQ 209
  I/O ports at fe00 [size=8]
  I/O ports at fe10 [size=4]
  I/O ports at fe20 [size=8]
  I/O ports at fe30 [size=4]
  I/O ports at fec0 [size=16]

01:00.0 VGA compatible controller: ATI Technologies Inc Unknown device 7183 (prog-if 00 [VGA])
  Flags: bus master, fast devsel, latency 0, IRQ 169
  Memory at c0000000 (64-bit, prefetchable) [size=256M]
  Memory at dfde0000 (64-bit, non-prefetchable) [size=64K]
  I/O ports at dc00 [size=256]
  Expansion ROM at dfe00000 [disabled] [size=128K]

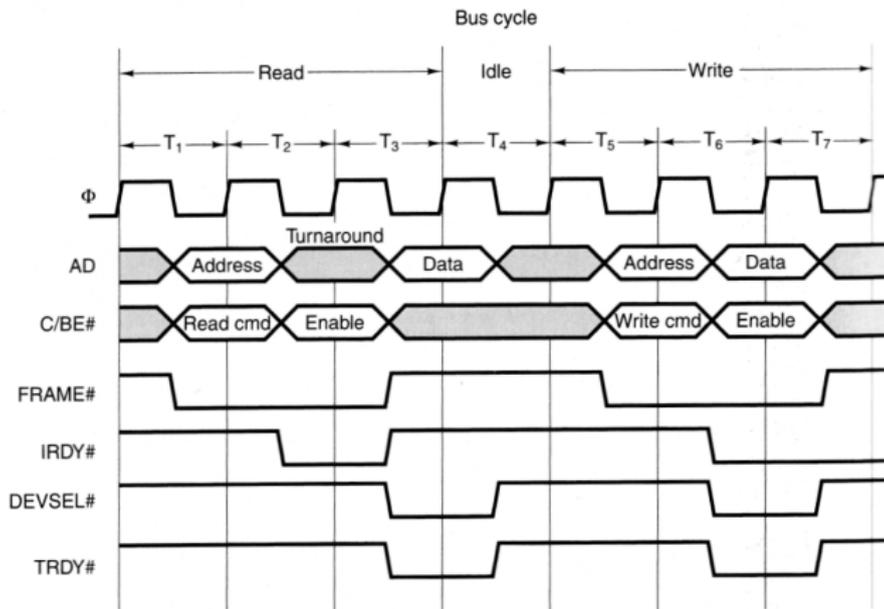
...
```

PCI-Bus: Leitungen („mandatory,,)

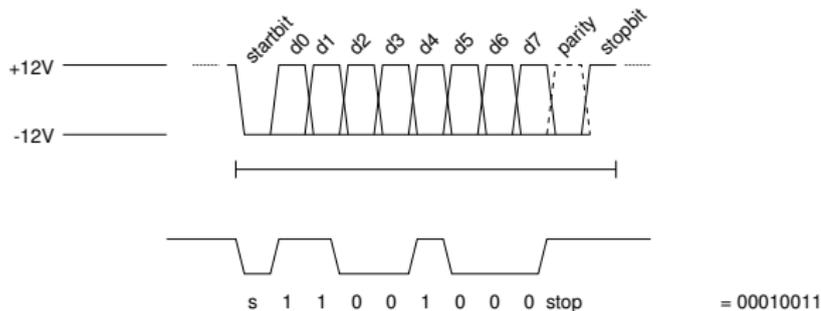
Signal	Lines	Master	Slave	Description
CLK	1			Clock (33 MHz or 66 MHz)
AD	32	×	×	Multiplexed address and data lines
PAR	1	×		Address or data parity bit
C/BE	4	×		Bus command/bit map for bytes enabled
FRAME#	1	×		Indicates that AD and C/BE are asserted
IRDY#	1	×		Read: master will accept; write: data 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



PCI-Bus: Transaktionen



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

Inhalt

- ▶ Speicher
- ▶ Register-Transfer-Ebene
- ▶ Halbleitertechnologie
- ▶ CMOS-Schaltungen
- ▶ Entwurf Integrierter Schaltungen

Erinnerung

Das **Konzept** des Digitalrechners (von-Neumann Prinzip) ist völlig unabhängig von der Technologie:

- ▶ mechanische Rechenmaschinen
- ▶ pneumatische oder hydraulische Maschinen
- ▶ Relais, Vakuumröhren, diskrete Transistoren
- ▶ molekulare Schaltungen
- ▶ usw.

Aber:

- ▶ nur hochintegrierte Halbleiterschaltungen („VLSI“) erlauben die billige Massenfertigung mit Milliarden von Komponenten
- ▶ **Halbleiter** und **Planarprozess** sind essentielle Basistechnologien

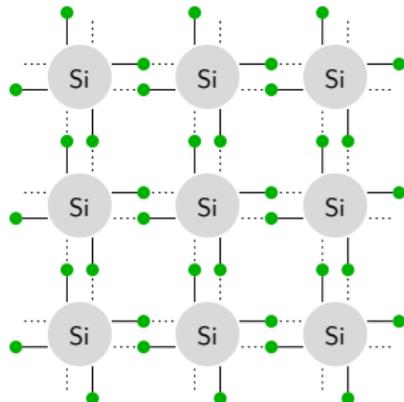
Halbleiter

Halbleiter stehen zwischen *Leitern* (z.B.: Metalle) und *Isolatoren*.

- ▶ bei Raumtemperatur quasi nicht-leitend
- ▶ Leitfähigkeit steigt mit der Temperatur \Rightarrow Heißeiter
- ▶ physikalische Erklärung über Bändermodell
 siehe <http://de.wikipedia.org/wiki/Halbleiter>

Kristallstruktur aus 4-wertigen Atomen

- ▶ elementare Halbleiter: Ge, Si
- ▶ Verbindungshalbleiter: GaAs, InSb



Leitung im undotierten Kristall

- ▶ Paarentstehung: Elektronen lösen sich aus Gitterverband
 Paar aus Elektron und „Loch“ entsteht
- ▶ Rekombination: Elektronen und Löcher verbinden sich
 quasistatischer Prozess
- ▶ Eigenleitungsdichte n_i : temperatur- und materialabhängig

$$\text{Si} : 1,2 \cdot 10^{10} \text{ cm}^{-3}$$

$$\text{Ge} : 2,5 \cdot 10^{13} \text{ cm}^{-3}$$

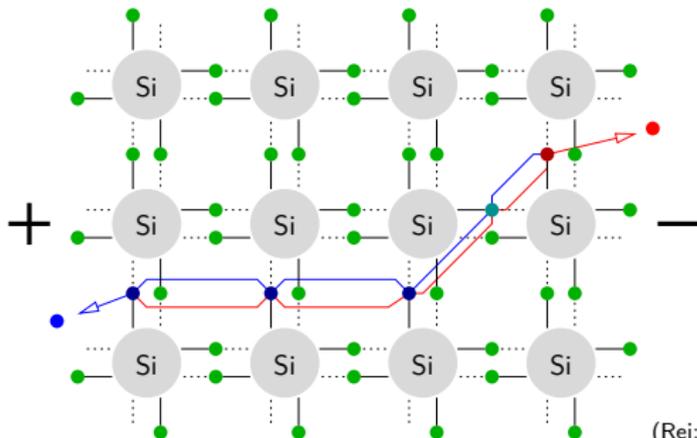
$$\text{GaAs} : 1,8 \cdot 10^6 \text{ cm}^{-3}$$

$$\text{bei } 300^\circ \text{K} \approx 20^\circ \text{C}$$

Atomdichte

$$\text{Si} : 5 \cdot 10^{22} \text{ cm}^{-3}$$

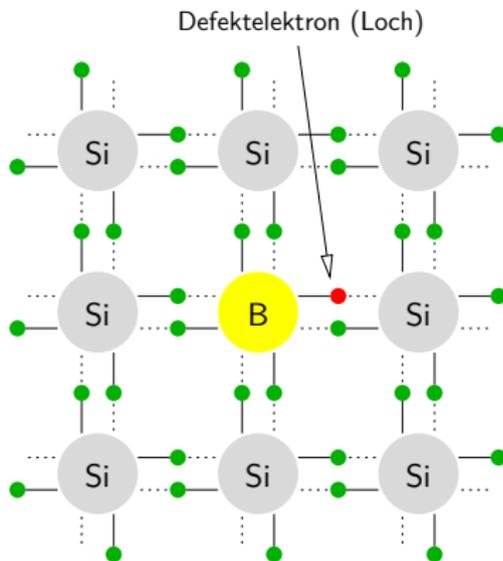
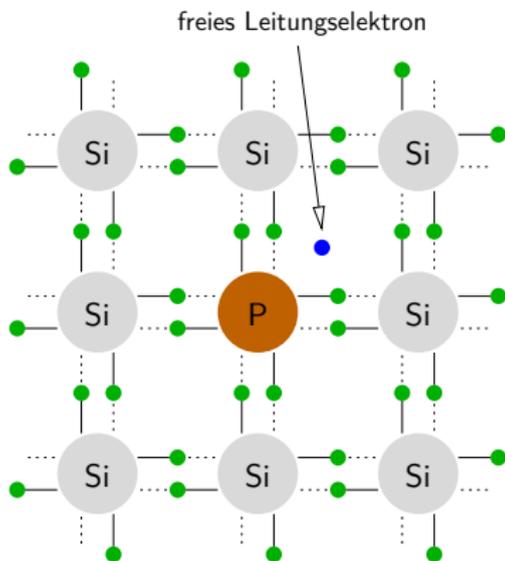
- ▶ es gilt: $n_i^2 = n_n \cdot n_p$



(Rei:98)

Dotierung mit Fremdatomen

Ein kleiner Teil der vierwertigen Atome wird durch fünf- oder dreiwertige Atome ersetzt.



Dotierung mit Fremdatomen (cont.)

- ▶ Donatoren, Elektronenspender: Phosphor, Arsen, Antimon
- ▶ Akzeptoren: Bor, Aluminium, Gallium, Indium

Dotierungsdichten	Stärke	Fremdatome [cm^{-3}]
	schwach n^-, p^-	$10^{15} \dots 10^{16}$
	mittel n, p	$10^{16} \dots 10^{19}$
	stark n^+, p^+	$10^{19} \dots$

- ▶ Beweglichkeit μ : materialspezifische Größe

$T = 300^\circ K$		Si	Ge	GaAs	[$cm^2/(Vs)$]
Elektronen μ_n	1500	3900	8500		
Löcher μ_p	450	1500	400		

- ▶ Leitfähigkeit: ergibt sich aus Material, Beweglichkeit und Ladungsträgerdichte(n)

$$K = e(n_n \mu_n + n_p \mu_p)$$

Dotierung mit Fremdatomen (cont.)

- ▶ selbst bei hoher Dotierung ist die Leitfähigkeit um Größenordnungen geringer als bei Metallen
 - Si 1 freier Ladungsträger pro 500 Atome ($10^{19}/5 \cdot 10^{22}$)
 - Metall mindestens 1 Ladungsträger pro Atom
- ▶ Majoritätsträger: Ladungsträger in Überzahl (i.d.R. Dotierung)
 Minoritätsträger: Ladungsträger in Unterzahl

$$n_i^2 = n_n \cdot n_p$$

Halbleitertechnologie

Übersicht in: <http://de.wikipedia.org/wiki/Silicium>



Rohsilizium

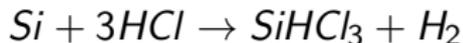
- ▶ Siliziumoxid (SiO_2): Sand, Kies...
ca. 20% der Erdkruste
- ▶ Herstellung im Lichtbogenofen: Siliziumoxid + Koks
 $SiO_2 + 2C \rightarrow Si + 2CO$
- ▶ amorphe Struktur, polykristallin
- ▶ noch ca. 2% Verunreinigungen (Fe, Al...)



Solarsilizium

Ziel: Fremdatome aus dem Silizium entfernen

1. Chemische Bindung des Siliziums



Reaktion mit Salzsäure erzeugt

SiHCl_3 Trichlorsilan

SiCl_4 Siliziumchlorid (10%)

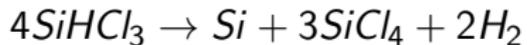
SiH_2Cl_2 div. andere Chlorsilane/Silane

$\text{FeCl}_2, \text{AlCl}_3$ div. Metallchloride

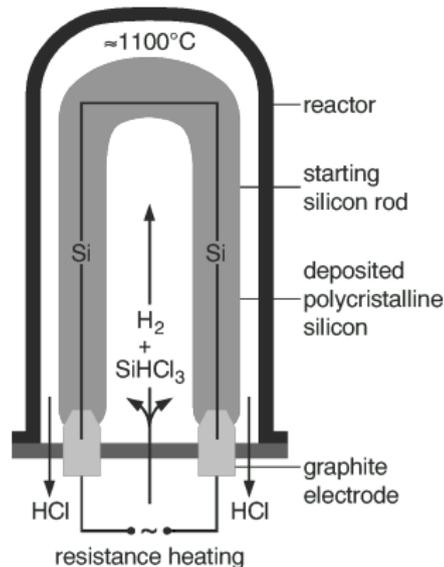
2. Verschiedene Kondensations- und Destillationschritte trennen Fremdverbindungen ab, hochreines Trichlorsilan entsteht
 < 1ppm Verunreinigungen

Solarsilizium (cont.)

3. CVD (Chemical Vapour Deposition) zur Abscheidung des Trichlorsilans zu elementarem Silizium



- ⇒ polykristallines Silizium
 < 0,1ppm Verunreinigungen



Siliziumeinkristall

Weitere Ziele

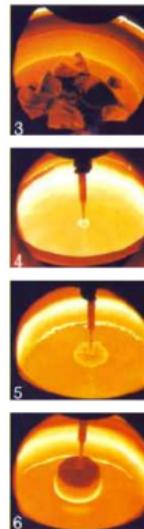
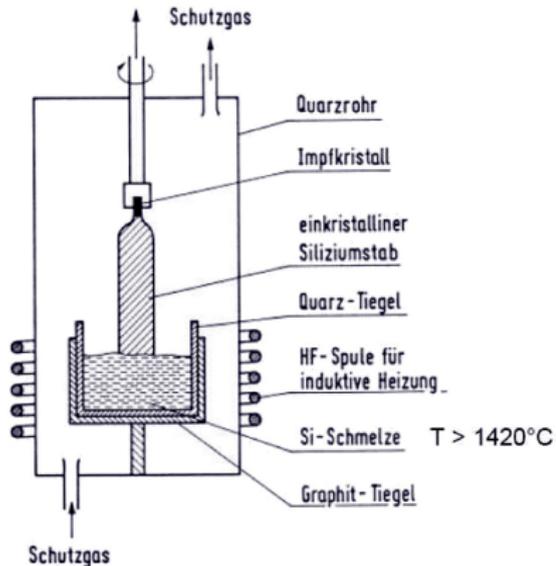
- ▶ Einkristalline Struktur erzeugen
- ▶ Reinheit für Halbleiterherstellung erhöhen
<, \ll 1ppb
- ▶ ggf. Dotierung durch Fremdatome einbringen

Es gibt dazu mehrere technische Verfahren, bei denen das polykristalline Silizium geschmolzen wird und sich monokristallin an einen Impfkristall anlagert.



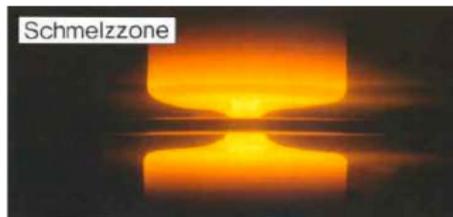
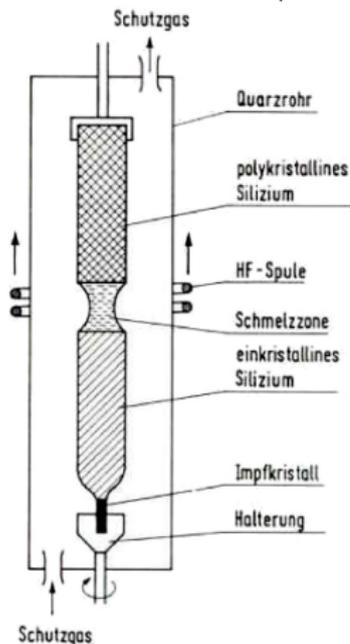
Siliziumeinkristall (cont.)

Czochralski-Verfahren (Tiegelziehverfahren)



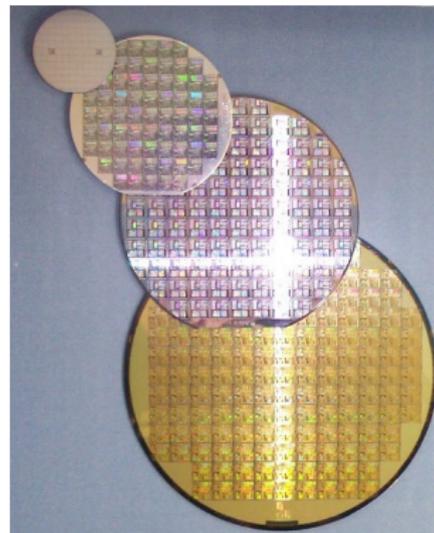
Siliziumeinkristall (cont.)

Zonenschmelz- / Zonenziehverfahren



Wafer

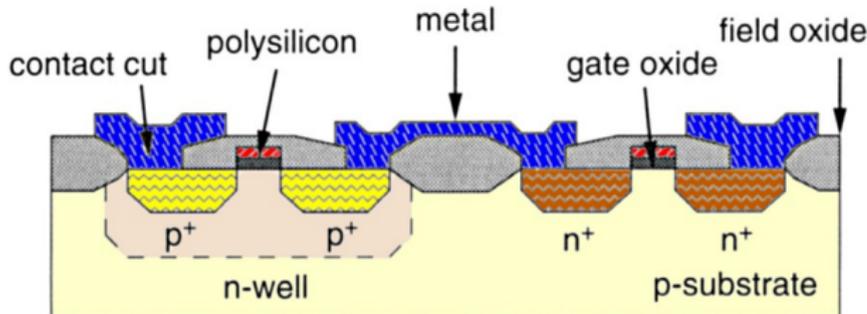
- ▶ weitere Bearbeitungsschritte:
zersägen, schleifen, läppen, ätzen, polieren
- ▶ Durchmesser bis 30 cm
2012: 45 cm (ITRS 07)
- Dicke $< 1\text{mm}$
- Rauhigkeit $\approx \text{nm}$
- ▶ Markieren: Kerben, Lasercodes. . .
früher „flats“



Technologien

Technologien zur Erstellung von Halbleiterstrukturen

- ▶ Epitaxie: Aufwachsen von Schichten
- ▶ Oxidation von Siliziumoberflächen: SiO_2 als Isolator
- ▶ Strukturierung durch Lithografie
- ▶ Dotierung des Kristalls durch Ionenimplantation oder Diffusion
- ▶ Ätzprozesse: Abtragen von Schichten

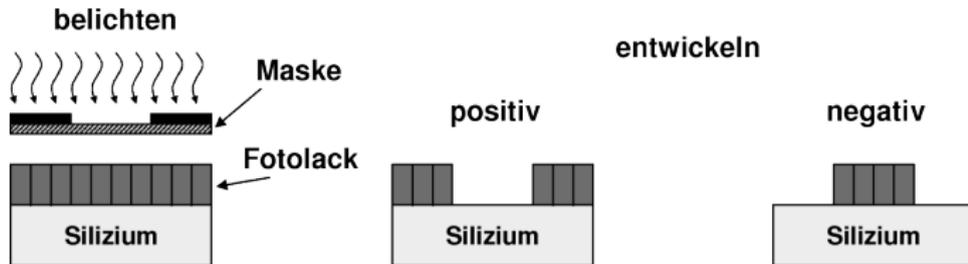
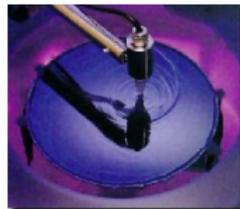


Lithografie

Übertragung von Strukturen durch einen Belichtungsprozess

1. Lack Auftragen (Aufschleudern)

- ▶ Positivlacke: hohe Auflösung \Rightarrow MOS
- ▶ Negativlacke: robust, thermisch stabil



Lithografie (cont.)

2. „Belichten“

- ▶ Maskenverfahren: 1:1 Belichtung, Step-Verfahren
UV-Lichtquelle
- ▶ Struktur direkt schreiben: Elektronen- / Ionenstrahl
- ▶ andere Verfahren: Röntgenstrahl- / EUV-Lithografie

3. Entwickeln, Härten, Lack entfernen

- ▶ je nach Lack verschiedene chemische Reaktionsschritte
- ▶ Härtung durch Temperatur

... weitere Schritte des **Planarprozess**

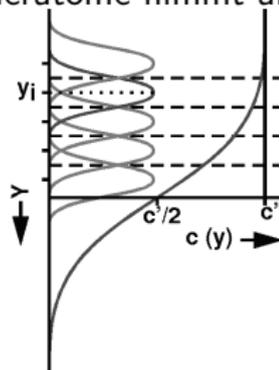
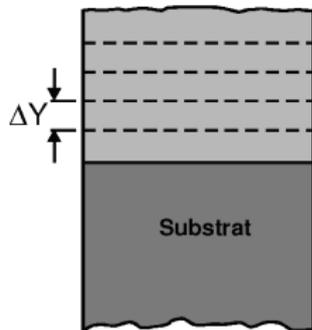
Dotierung

Fremdatome in den Siliziumkristall einbringen

► Diffusion

- Diffusionsofen ähnlich CVS-Reaktor
- gaußförmiges Dotierungsprofil

Konzentration der Dotieratome nimmt ab

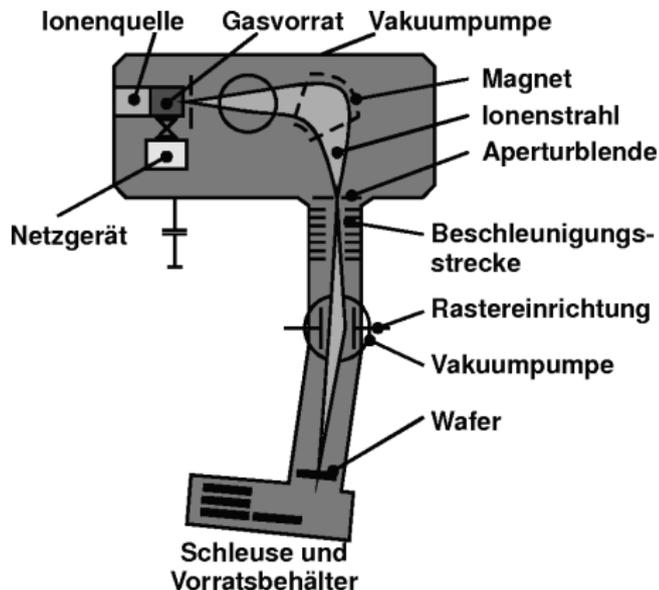


Dotierung (cont.)

- ▶ Ionenimplantation
 - ▶ „Beschuss“ mit Ionen
 - ▶ Beschleunigung der Ionen im elektrischen Feld
 - ▶ Über die Energie der Ionen kann die Eindringtiefe sehr genau eingestellt werden
 - ▶ „Temperung“ notwendig: Erhitzen des Einkristalls zur Neuorganisation des Kristallgitters

Dotierung (cont.)

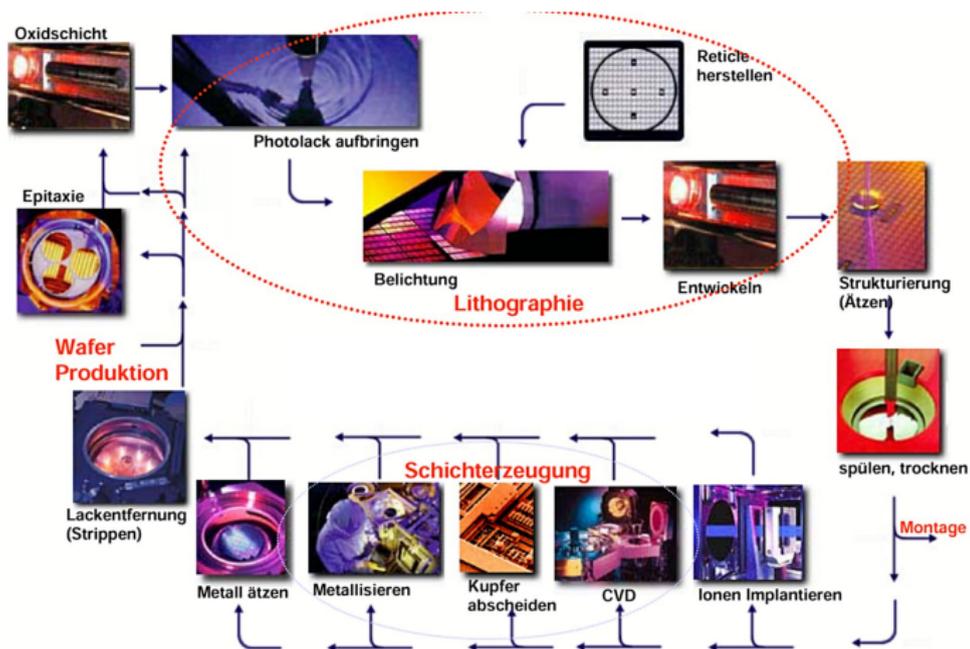
Ionenimplantation



Planarprozess

- ▶ Der zentrale Ablauf bei der Herstellung von Mikroelektronik
- ▶ Ermöglicht die gleichzeitige Fertigung aller Komponenten auf dem Wafer
- ▶ Schritte
 1. Vorbereiten / Beschichten des Wafers:
Oxidation, CVD, Aufdampfen, Sputtern...
 2. Strukturieren durch Lithografie
 3. Übertragen der Strukturen durch Ätzprozesse
 4. Modifikation des Materials: Dotierung, Oxidation
 5. Vorbereitung für die nächsten Prozessschritte...

Planarprozess: Schema



Inhalt

- ▶ Speicher
- ▶ Register-Transfer-Ebene
- ▶ Halbleitertechnologie
- ▶ CMOS-Schaltungen
- ▶ Entwurf Integrierter Schaltungen

MOS-Transistor

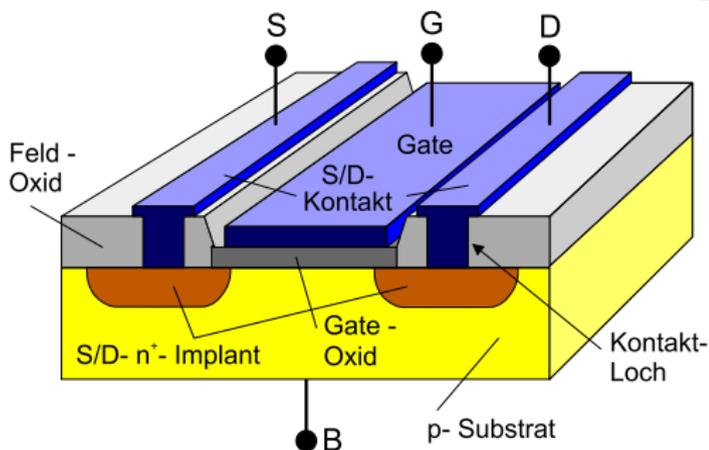
- ▶ MOS: Metal On Silicon
FET : Feldeffekttransistor
 - ▶ <http://olli.informatik.uni-oldenburg.de/weTEiS/weteis/tutorium.htm>
 - ▶ <http://de.wikipedia.org/wiki/Feldeffekttransistor>
 - ▶ <http://de.wikipedia.org/wiki/MOSFET>

Literatur: Weste&Eshragian, Tietze&Schenk, usw.

- ▶ unipolarer Transistor: nur eine Art von Ladungsträgern, die Majoritätsträger, ist am Stromfluss beteiligt

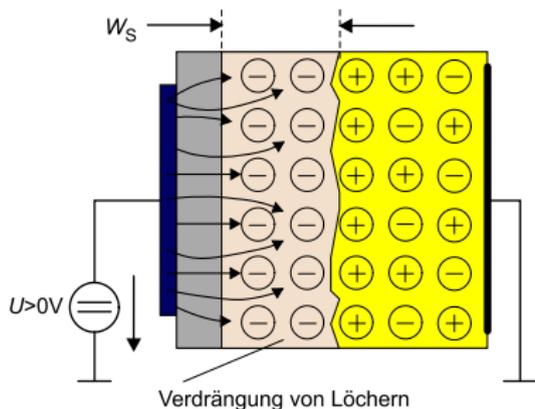
MOS-Transistor (cont.)

- ▶ Anschlüsse: **Source** Quelle der Ladungsträger
- Gate** steuert den Stromfluss
- Drain** Senke der Ladungsträger
- Bulk** siehe „Herstellungstechnik“

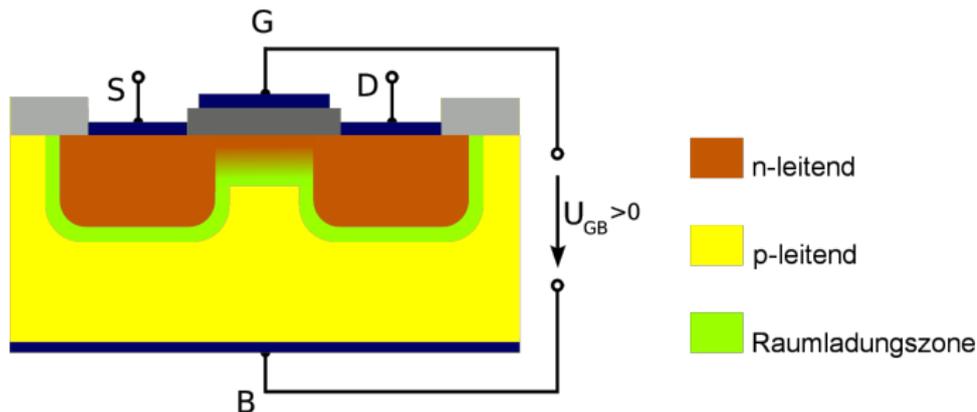


MOS-Transistor (cont.)

- ▶ Funktionsweise: die Ladung des Gates erzeugt ein elektrisches Feld. Durch Inversion werden Ladungsträger unterhalb des Gates verdrängt und ein leitender Kanal zwischen Source und Drain entsteht



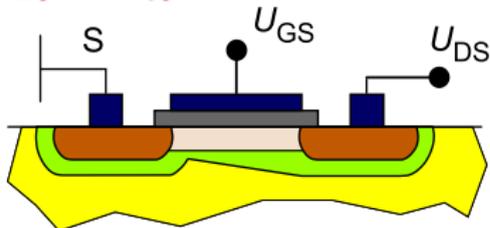
MOS-Transistor (cont.)



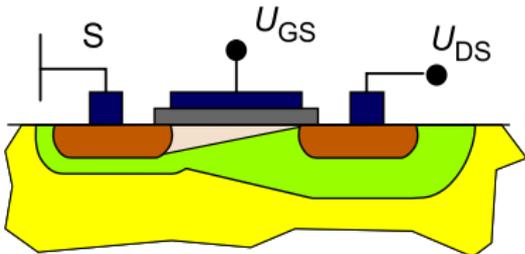
- ▶ Schwellspannung U_P : abhängig von der Dotierungsdichte, den Parametern des MOS-Kondensators (Dicke und Material der Gate-Isolationsschicht)...
 U_P möglichst klein: 0,3...0,8 V früher: deutlich mehr

MOS-Transistor (cont.)

- ▶ $U_{DS} \ll U_{GS} - U_P$ normaler Betrieb (Triodenbereich)

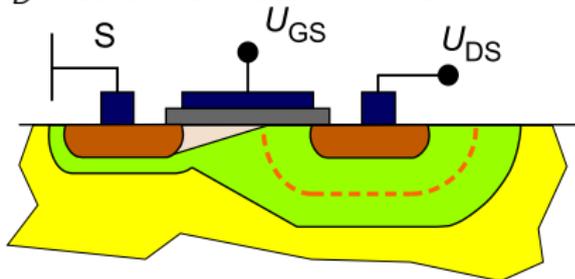


- ▶ $U_{DS} = U_{GS} - U_P$ Kanalabschnürung
 Spannungsabfall zwischen S und D durch den Kanalwiderstand



MOS-Transistor (cont.)

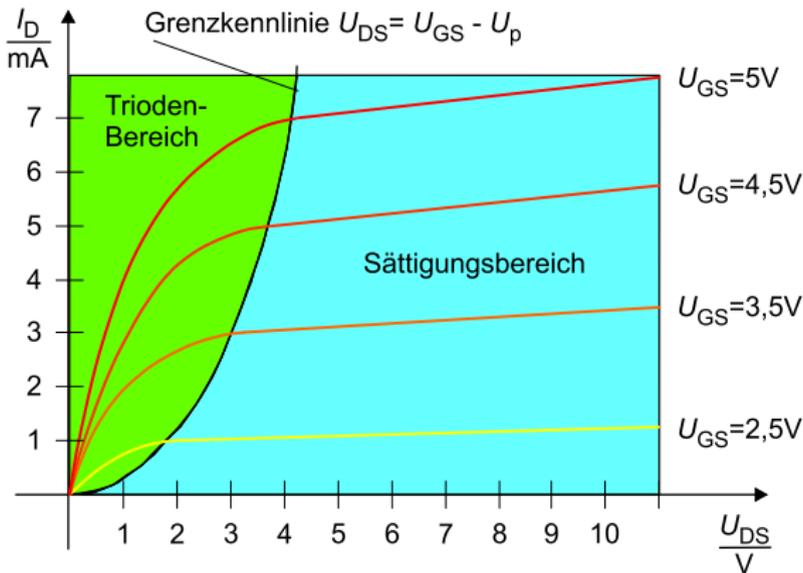
- ▶ $U_{DS} > U_{GS} - U_P$ Kanalverkürzung (Sättigungsbereich)
 Der Kanal wird weiter verkürzt, die Spannung U_{DS} bewirkt ein virtuell größeres Drain durch Inversion.
 I_D wächst nur noch minimal.



- ⇒ kurze Kanäle aktueller Submikronprozesse können allein durch hohe Spannungen U_{DS} leitend werden (Durchgreifbetrieb)
- ⇒ einer der Gründe für sinkende Versorgungsspannungen

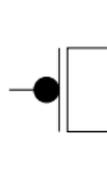
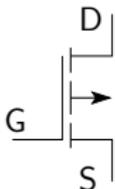
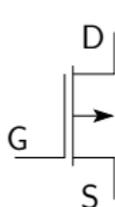
MOS-Transistor (cont.)

► Kennlinienfeld

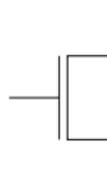
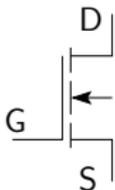
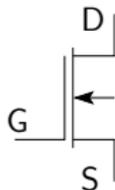


MOS-Transistor: Schaltsymbole

P-Kanal



N-Kanal



selbstleitend

selbstsperrend

Verarmungs-

Anreicherungstyp

Inhalt

- ▶ Speicher
- ▶ Register-Transfer-Ebene
- ▶ Halbleitertechnologie
- ▶ **CMOS-Schaltungen**
- ▶ Programmierbare Logikbausteine
- ▶ Entwurf Integrierter Schaltungen



CMOS-Technologie

Complementary Metal-Oxide Semiconductor: die derzeit dominierende Technologie für alle hochintegrierten Schaltungen:

- ▶ Schaltungsprinzip nutzt n-Kanal und p-Kanal Transistoren
- ▶ alle elementaren Gatter verfügbar
- ▶ effiziente Realisierung von *Komplexgattern*
- ▶ *Transmission-Gate* als elektrischer Schalter
- ▶ effiziente Realisierung von Flipflops und Speichern

- ▶ sehr hohe Integrationsdichte möglich
- ▶ sehr schnelle Schaltgeschwindigkeit der Gatter
- ▶ sehr geringer Stromverbrauch pro Gatter möglich
- ▶ Integration von digitalen und analogen Komponenten

CMOS: Überblick

- ▶ Schaltungsprinzip
- ▶ Inverter und nicht-invertierender Verstärker
- ▶ NAND, NAND3, NOR (und AND, OR)
- ▶ XOR

- ▶ Komplexgatter
- ▶ Transmission-Gate

- ▶ Beispiele für Flipflops
- ▶ SRAM

CMOS: Schaltungsprinzip von „static CMOS“

- ▶ Transistoren werden als Schalter betrachtet
- ▶ zwei zueinander **komplementäre** Zweige der Schaltung
- ▶ nur n-Kanal Transistoren zwischen Masse und Ausgang y
- ▶ nur p-Kanal Transistoren zwischen VCC und Ausgang y

- ▶ der p-Kanal Zweig ist komplementär („dualer Graph“) zum n-Kanal Zweig: jede Reihenschaltung von Elementen wird durch eine Parallelschaltung ersetzt (und umgekehrt)

- ▶ immer ein direkt leitender Pfad von entweder VCC (1) oder Masse (0) zum Ausgang
- ▶ niemals ein direkt leitender Pfad von VCC nach Masse
- ▶ kein statischer Stromverbrauch im Gatter

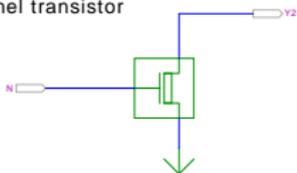
CMOS: „static“ CMOS-Gatter

- ▶ Schaltungen: *negierte monotone boole'sche Funktionen*
- ▶ Beliebiger schaltalgebraischer Ausdruck *ohne Negation*: \vee, \wedge
- ▶ Negation des gesamten Ausdrucks: Ausgang *immer* negiert
- ▶ je Eingang: ein Paar p-/n-Kanal Transistoren
- ▶ Dualitätsprinzip: n- und p-Teil des Gatters

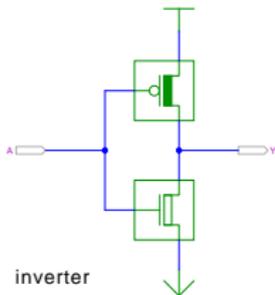
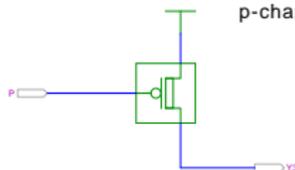
n-Teil	p-Teil	Logik, ohne Negation
seriell	\Leftrightarrow parallel	$\equiv \wedge$ / und
parallel	\Leftrightarrow seriell	$\equiv \vee$ / oder
- ▶ Konstruktion
 1. n-Teil aus Ausdruck ableiten
 2. p-Teil dual dazu entwickeln

CMOS: n- und p-Kanal Transistor, Inverter, Verstärker

n-channel transistor

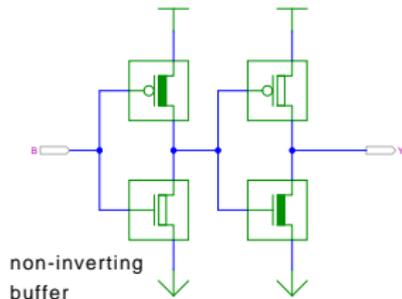


p-channel transistor



inverter

links: $Y = \neg A$


 non-inverting
 buffer

rechts: $Y = \neg \neg A = A$

CMOS: Inverter

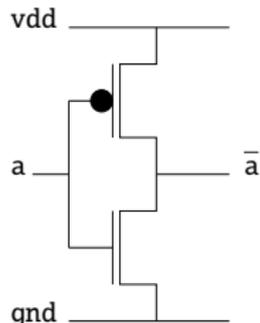
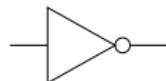
Funktionsweise

- ▶ selbstsperrende p- und n-Kanal Transistoren
- ▶ komplementär beschaltet
- ▶ Ausgang: Pfad über p-Transistoren zu V_{dd}
 –"– n-Transistoren zu Gnd
- ▶ genau *einer* der Pfade leitet

- ▶ Eingang T_{pP} T_{nN} Ausgang

$a = 0 \rightarrow$ leitet / sperrt \rightarrow über T_P mit V_{dd} verbunden = 1

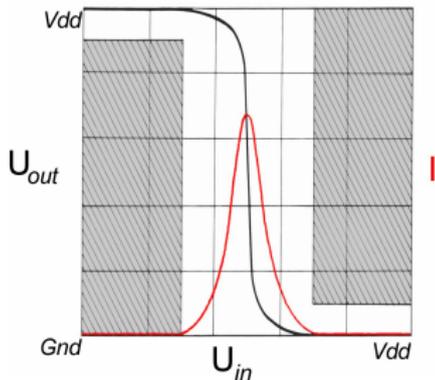
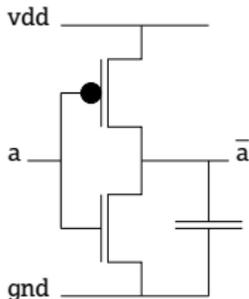
$a = 1 \rightarrow$ sperrt / leitet \rightarrow über T_N mit Gnd verbunden = 0



CMOS: Leistungsaufnahme

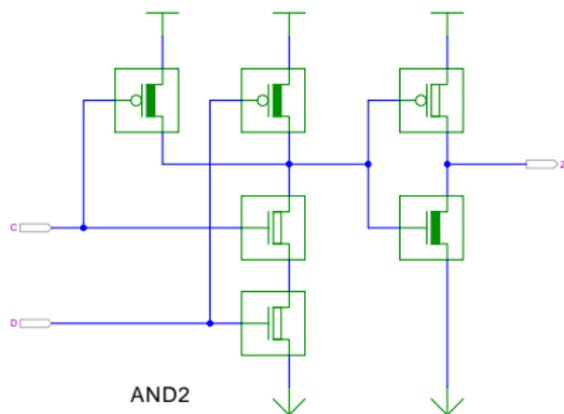
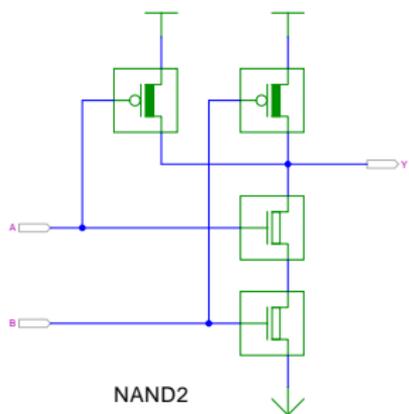
Leistungsaufnahme

1. $U_{in} = 0$, bzw. V_{dd} : Sperrstrom, nur μA
 \Rightarrow niedrige statische Leistungsaufnahme
2. Querstrom beim Umschalten:
 kurzfristig leiten beide Transistoren
 \Rightarrow Forderung nach steilen Flanken
3. Kapazitive Last: Fanout-Gates
 Energie auf Gate(s): $W = \frac{1}{2} C_T V_{dd}^2$
 Verlustleistung_(0/1/0): $P = C_T V_{dd}^2 \cdot f$



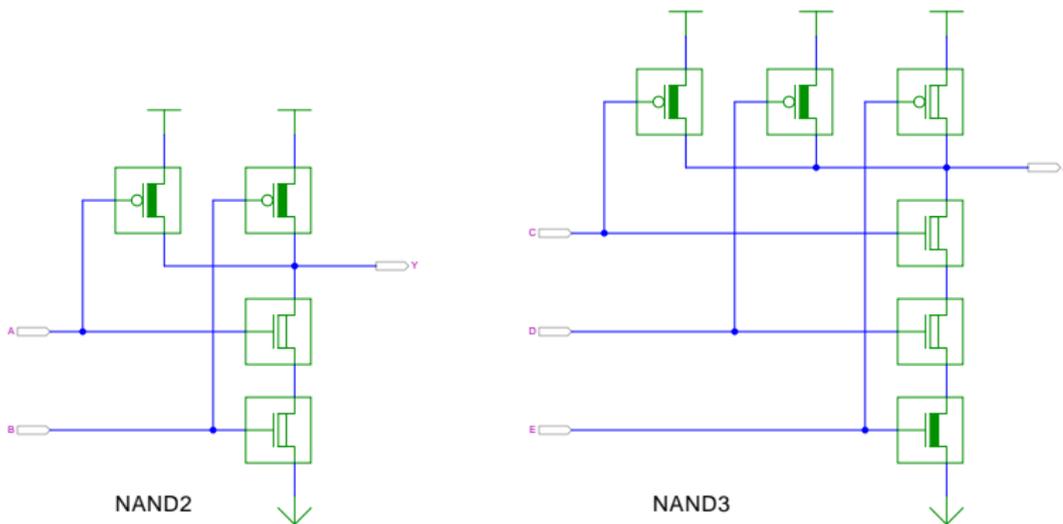
Transfercharakteristik

CMOS: NAND- und AND-Gatter



- ▶ NAND: n-Transistoren in Reihe, p-Transistoren parallel
- ▶ AND: Kaskade aus NAND und Inverter

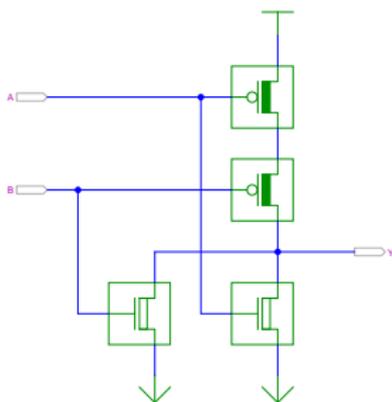
CMOS: NAND-Gatter mit drei Eingängen



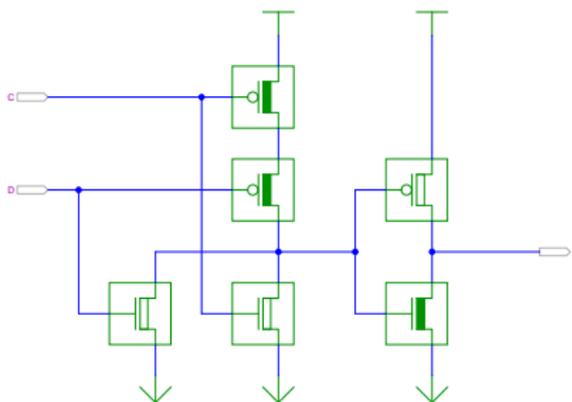
- ▶ n-Transistoren in Reihe, p-Transistoren parallel
- ▶ normalerweise max. 4 Transistoren in Reihe (Spannungsabfall)

CMOS: NOR- und OR-Gatter

NOR2



OR2



- ▶ Struktur komplementär zum NAND/AND
- ▶ n-Transistoren parallel, p-Transistoren in Reihe
- ▶ p-Transistoren schalten träge: etwas langsamer als NAND

CMOS-Technologie: Demos

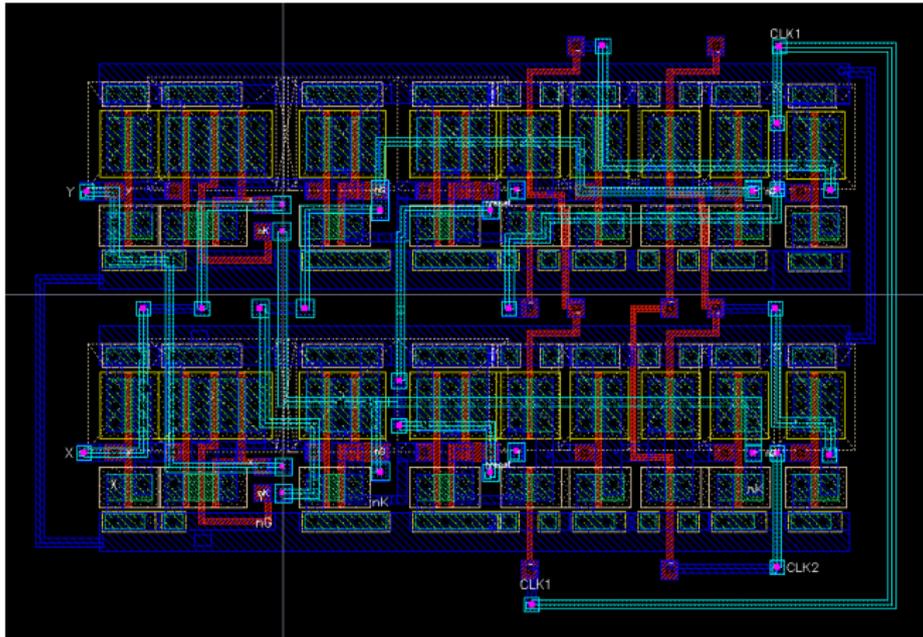
CMOS 3-input NAND gate demo: $Y = \overline{(A \wedge B \wedge C)}$
 [Click near inputs to toggle input voltages:]

C	B	A	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Colors: 1 [+5V] 0 [0V] [Z (floating)] [Short-circuit]

- ▶ Interaktive Demonstration der CMOS-Grundgatter (Java):
<http://tams-www.informatik.uni-hamburg.de/applets/cmos/>

CMOS: Beispiel-Layout



CMOS: Komplex-Gatter

Verallgemeinerung des Prinzips von NAND und NOR

- ▶ beliebige Parallel- und Serienschaltung der n-Transistoren
- ▶ komplementäre Seriell- und Parallelschaltung der p-Transistoren
- ▶ typischerweise max. 4 Transistoren in Reihe

- ▶ viele invertierende logische Funktionen effizient realisierbar
- ▶ Schaltungslayout automatisch synthetisierbar
- ▶ zwei gängige Varianten:
 - ▶ AOI-Gatter („AND-OR-invert“)
 - ▶ OAI-Gatter („OR-AND-invert“)

CMOS-Komplexgatter

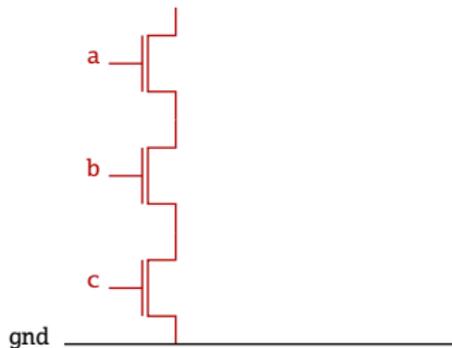
Beispiel: $\overline{(a \wedge b \wedge c) \vee d \vee (e \wedge f)}$

„AOI321-Gatter“, AND-OR-INVERT Struktur

- ▶ UND-Verknüpfung von (a,b,c)
 - ▶ UND-Verknüpfung von (e,f)
 - ▶ OR-Verknüpfung der drei Terme
 - ▶ anschließend Invertierung
-
- ▶ direkte Realisierung hätte $(6+2)+(0)+(4+2)+4$ Transistoren

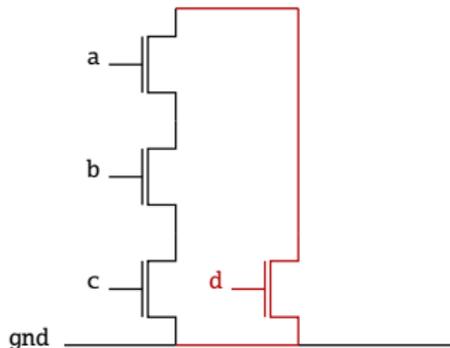
CMOS-Komplexgatter

Beispiel: $\overline{(a \wedge b \wedge c) \vee d \vee (e \wedge f)}$



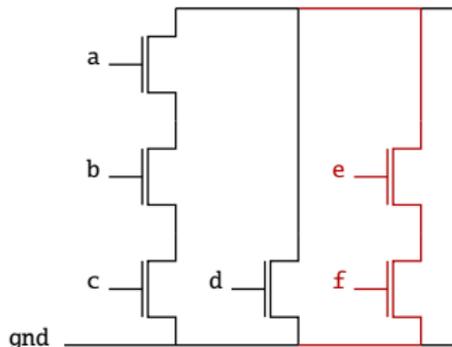
CMOS-Komplexgatter

Beispiel: $\overline{(a \wedge b \wedge c) \vee d \vee (e \wedge f)}$



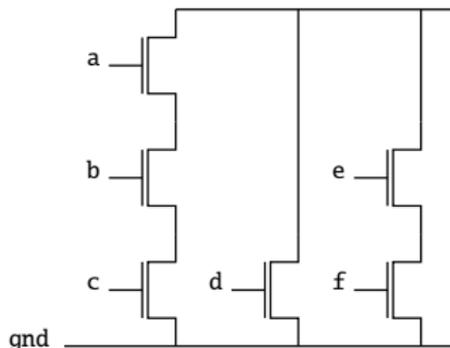
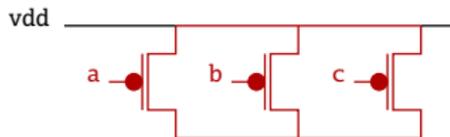
CMOS-Komplexgatter

Beispiel: $\overline{(a \wedge b \wedge c) \vee d \vee (e \wedge f)}$



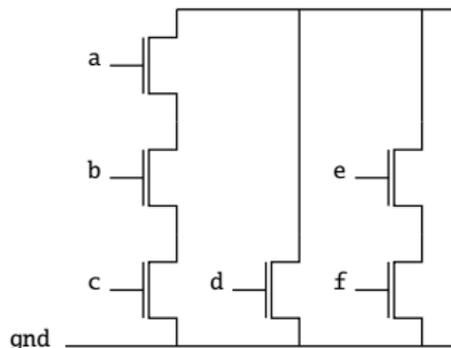
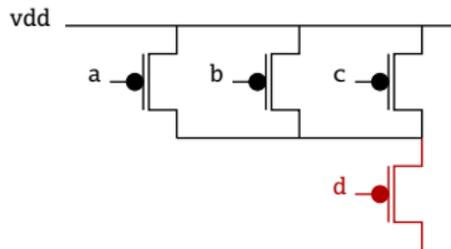
CMOS-Komplexgatter

Beispiel: $\overline{(a \wedge b \wedge c)} \vee d \vee (e \wedge f)$



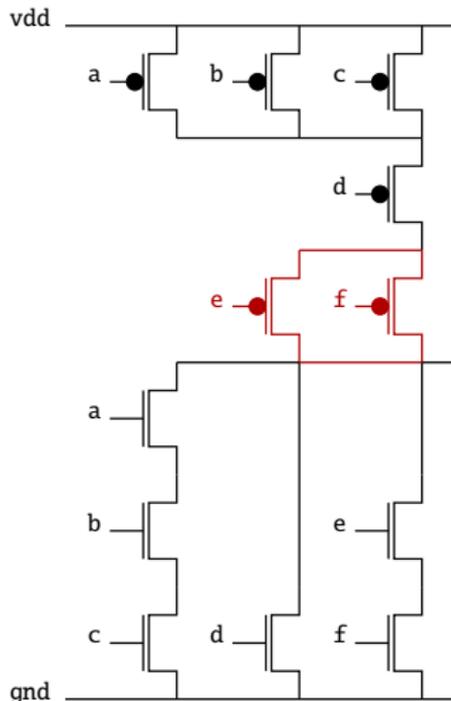
CMOS-Komplexgatter

Beispiel: $\overline{(a \wedge b \wedge c) \vee d \vee (e \wedge f)}$



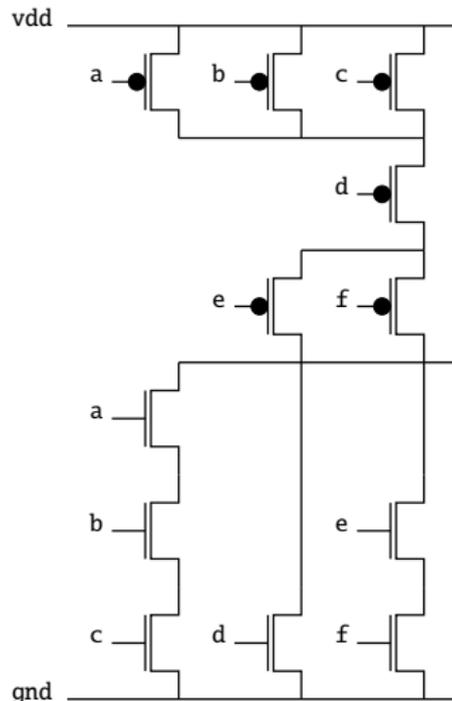
CMOS-Komplexgatter

Beispiel: $\overline{(a \wedge b \wedge c)} \vee d \vee (e \wedge f)$



CMOS-Komplexgatter

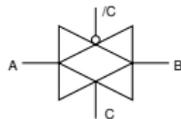
Beispiel: $\overline{(a \wedge b \wedge c) \vee d \vee (e \wedge f)}$



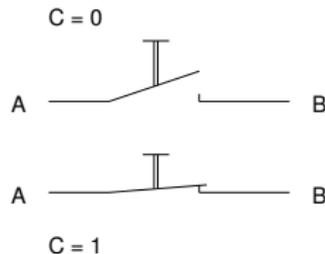
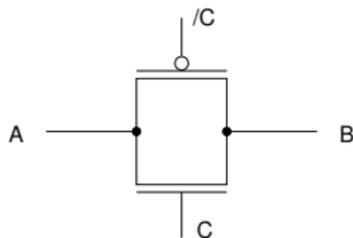
CMOS: Transmission-Gate

Transmissions-Gatter (*transmission gate, t-gate*)

- ▶ Paar aus je einem n- und p-Kanal MOS-Transistor
- ▶ symmetrische Anordnung
- ▶ Ansteuerung der beiden Gate-Elektroden mit invertierter Polarität
- ▶ entweder beide Transistoren leiten, oder beide sperren
- ▶ Funktion entspricht **elektrisch gesteuertem Schalter**
- ▶ effiziente Realisierung vieler Schaltungen

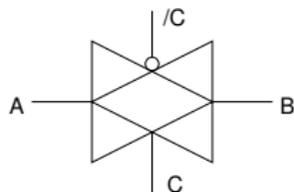


CMOS: Transmission-Gate

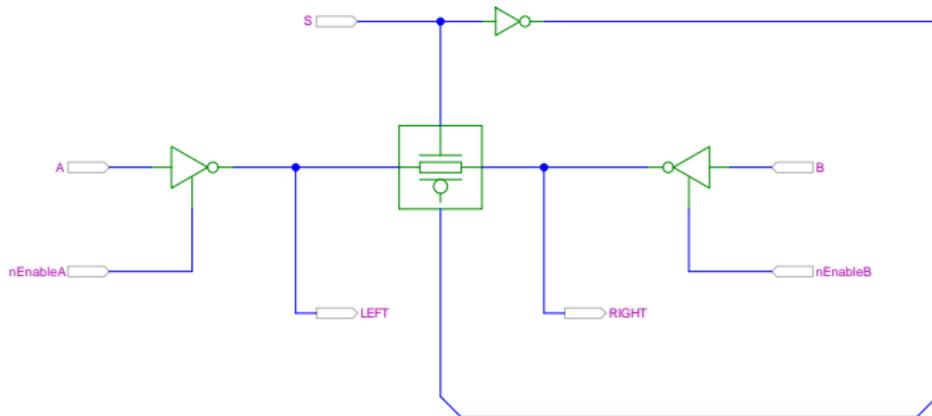


elektrisch gesteuerter Schalter:

- ▶ $C = 0$: keine Verbindung von A nach B
- ▶ $C = 1$: leitende Verbindung von A nach B
- ▶ symmetrisch in beide Richtungen



CMOS: Transmission-Gate Demo



tgate demonstration

- ▶ Werte an *A* und *B* anlegen, Treiber mit enable-Signalen aktivieren
- ▶ Gatter mit *S* ein- oder ausschalten

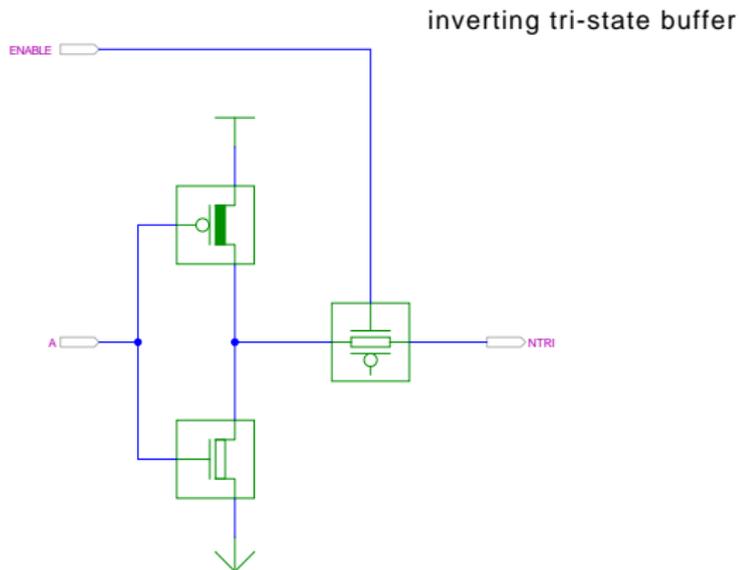
tams.informatik.uni-hamburg.de/applets/hades/webdemos/05-switched/40-cmos/tgate.html

CMOS: Transmission-Gate Tristate-Treiber

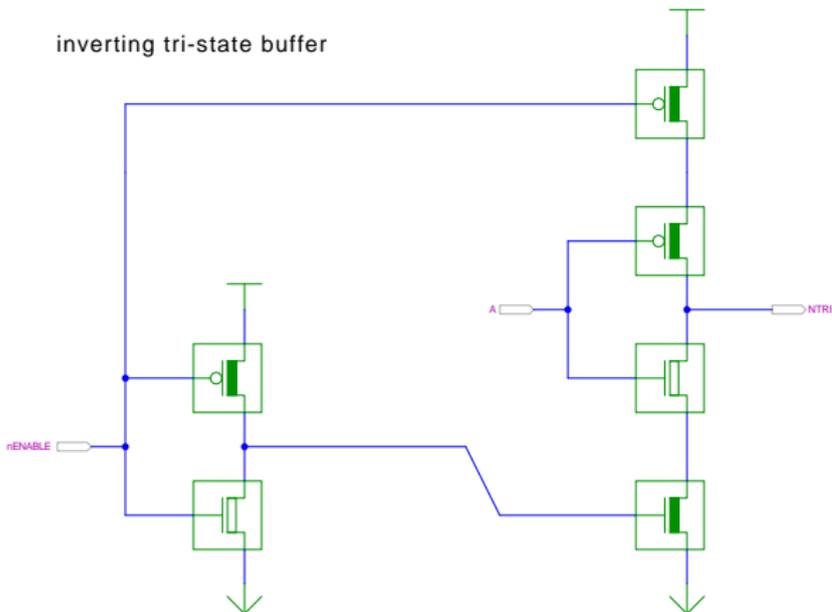
- ▶ $e = 1$:
Ausgang aktiv: $y = \bar{a}$

- ▶ $e = 0$:
hochohmig: $y = Z$

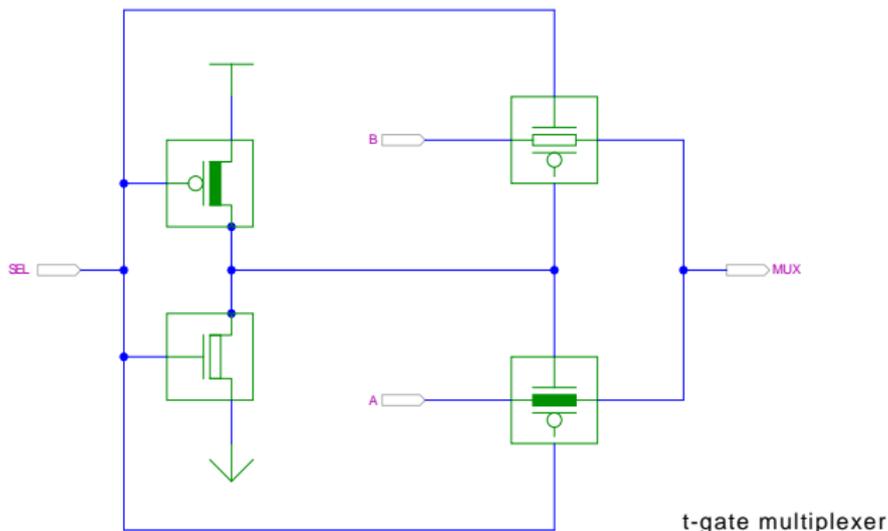
e	a	y
0	0	Z
0	1	Z
1	0	1
1	1	0



CMOS: Tristate-Treiber (Variante)

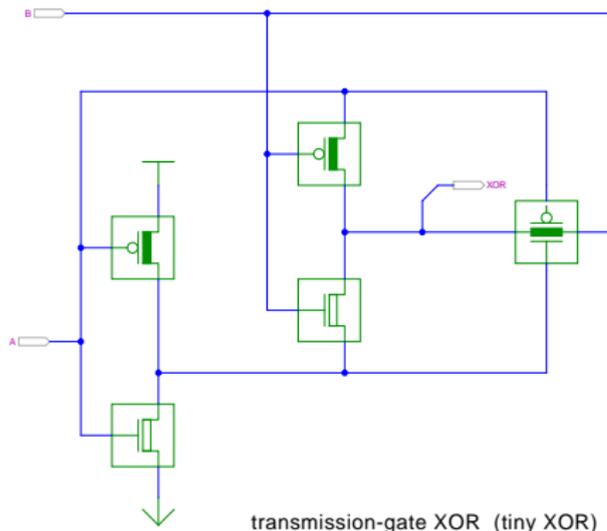


CMOS: Transmission-Gate Multiplexer



- ▶ kompakte Realisierung (4 bzw. 6 Transistoren)
- ▶ Eingänge *a* und *b* nicht verstärkt: nur begrenzt kaskadierbar

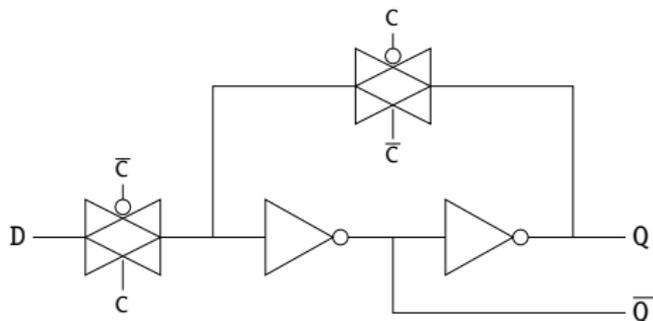
CMOS: Transmission-Gate XOR-Gatter



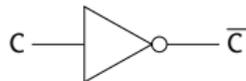
- ▶ kompakte Realisierung des XOR (nur 6 Transistoren)
- ▶ Eingang *b* nicht verstärkt: nur begrenzt kaskadierbar

CMOS: D-Latch (quasi-statisch)

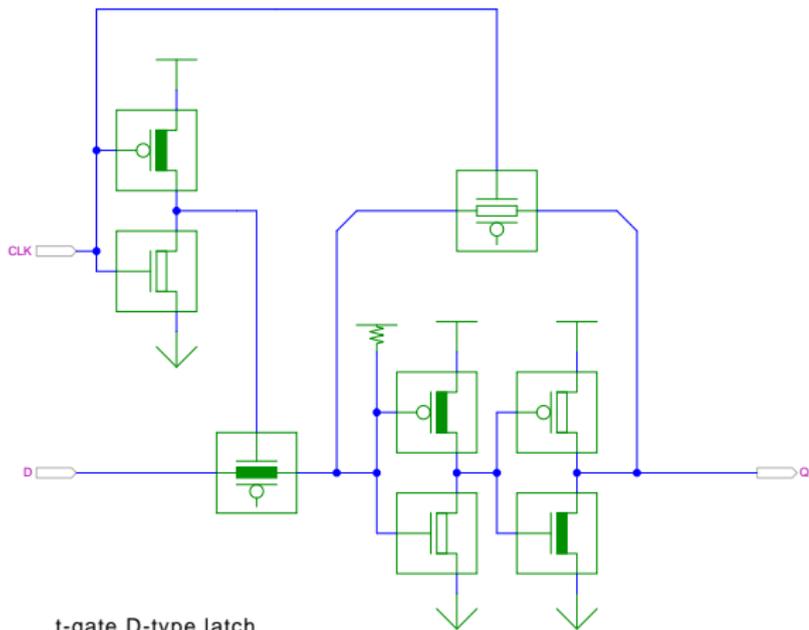
- ▶ Dateneingang D , Takteingang C
- ▶ Transmission-Gates als Schalter:
 - $C = 1$: Transparent Eingang über die Inverter zum Ausgang
 - $C = 0$: Speicherung Rückkopplungspfad aktiv



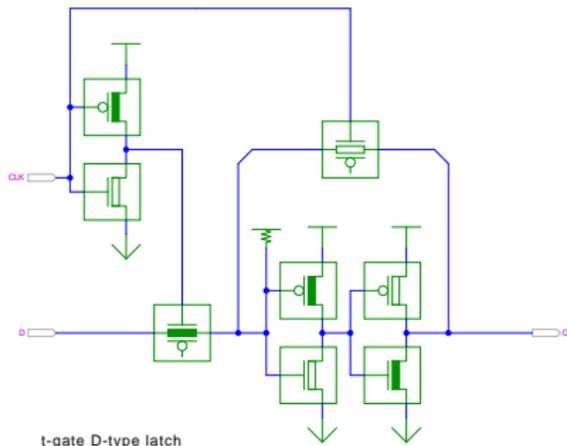
Latch, high-aktiv



CMOS: D-Latch (Demo)

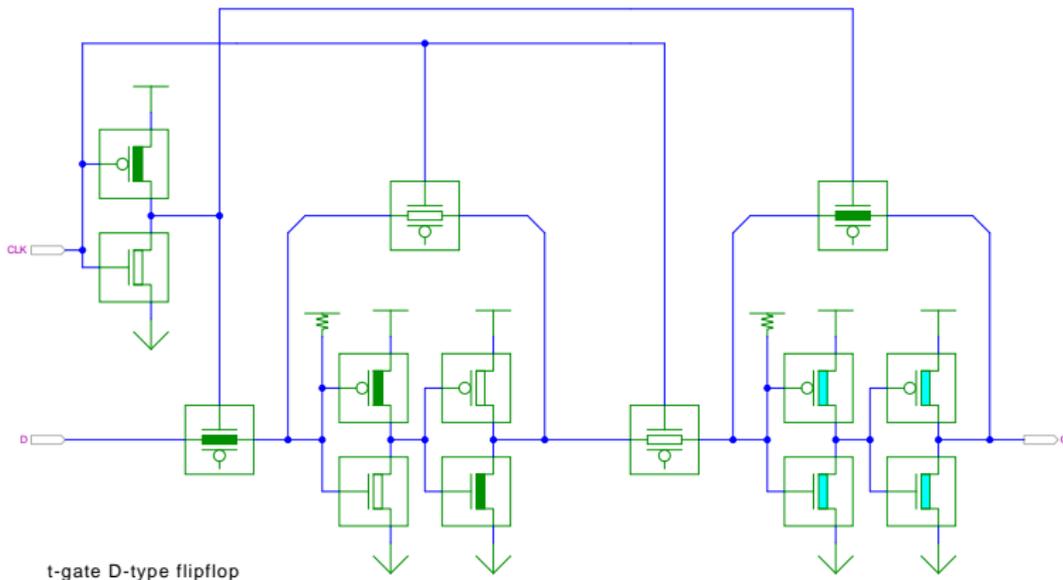


CMOS: Funktion des T-Gate D-Latch



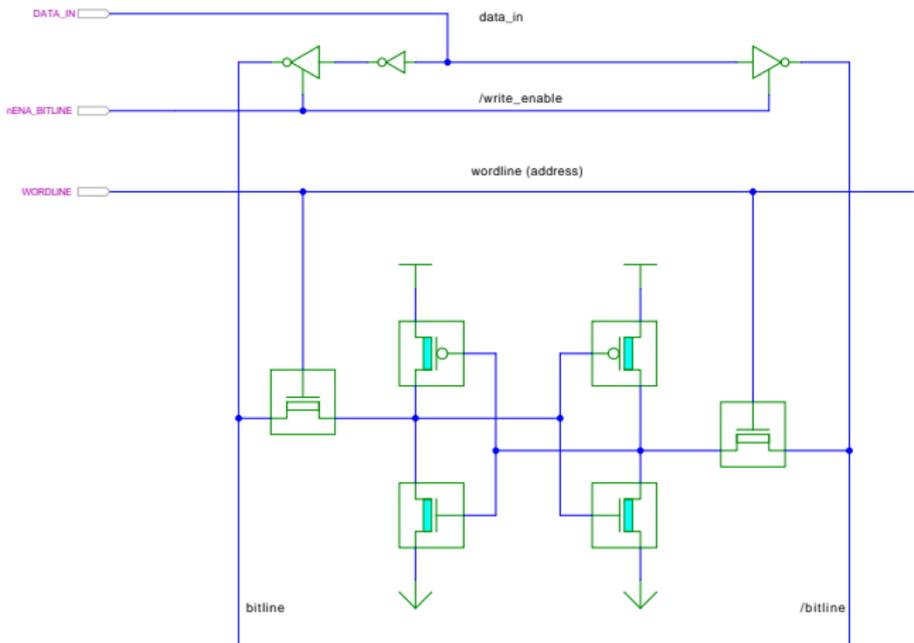
- ▶ Gegentakt-Ansteuerung der beiden T-Gates
- ▶ Takteingang C , Inverter erzeugt zusätzlich \overline{C}
- ▶ vorderes T-Gate aktiv: direkter Pfad von D nach Q
- ▶ Rückkopplungsschleife: Q zweimal invertiert: also gespeichert

CMOS: Transmission-Gate D-Flipflop



- Flankensteuerung via Master-Slave Prinzip

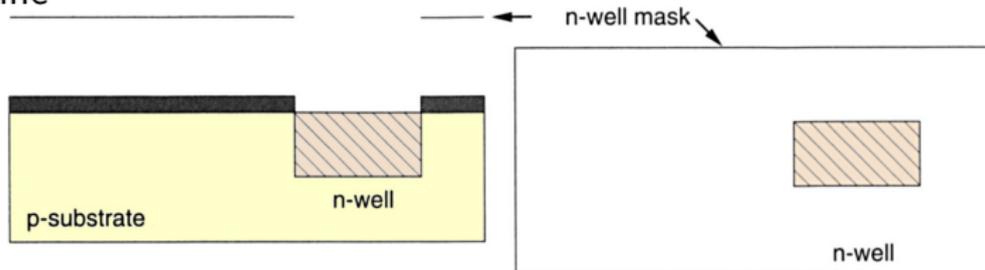
CMOS: Sechs-Transistor Speicherstelle („6T“)



CMOS-Herstellungsprozess

Ein n-Wannen Prozesses

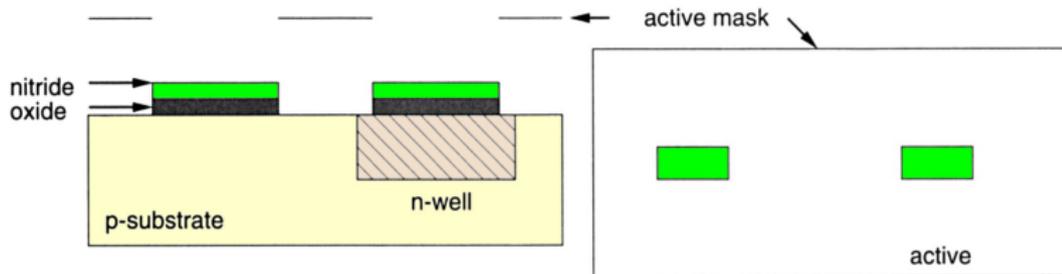
1. Ausgangsmaterial: p-dotiertes Substrat
2. n-Wanne



- ▶ Dotierung für p-Kanal Transistoren
- ▶ Herstellung: Ionenimplantation oder Diffusion

CMOS-Herstellungsprozess (cont.)

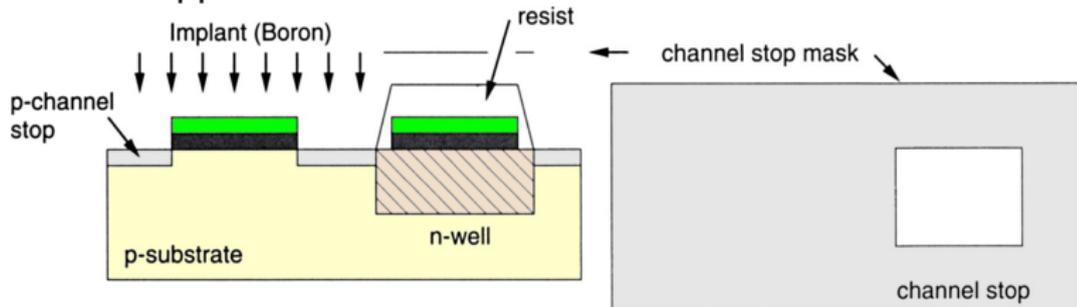
3. „aktive“ Fläche / Dünnoxid



- ▶ Spätere Gates und $p^+/-n^+$ -Gebiete
- ▶ Herstellung: Epitaxie SiO_2 und Abdeckung mit Si_3N_4

CMOS-Herstellungsprozess (cont.)

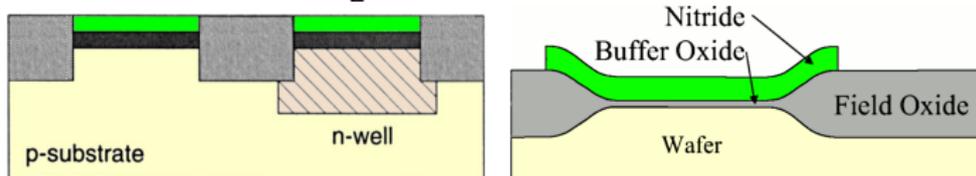
4. p-Kanalstopp



- ▶ Begrenzt n-Kanal Transistoren
 - ▶ p-Wannen Maske, bzw. \neg n-Wanne
 - ▶ Maskiert durch Resist und Si_3N_4
 - ▶ Substratbereiche in denen keine n-Transistoren sind
 - ▶ Herstellung: p^+ -Implant (Bor)
- ▶ n-Kanalstopp aktueller Prozesse: analog dazu

CMOS-Herstellungsprozess (cont.)

5. Resist entfernen
6. Feldoxid aufwachsen – SiO_2



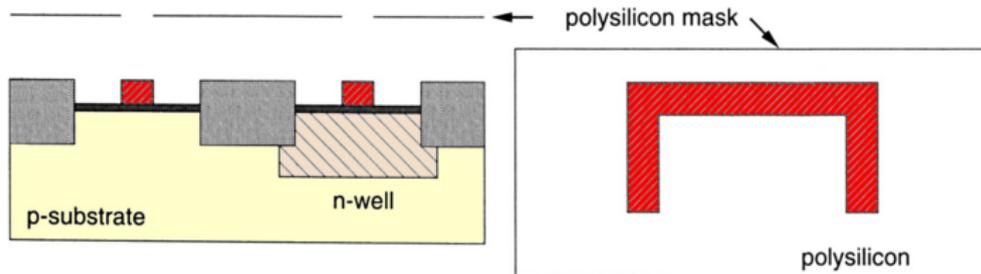
- ▶ LOCOS: **L**ocal **O**xidation of **S**ilicon
- ▶ Maskiert durch Si_3N_4
- ▶ Wächst auch lateral unter Si_3N_4/SiO_2 (aktive) Bereiche
 engl. *bird's beak*
- ▶ Der aktive Bereich wird kleiner als vorher maskiert
- ▶ Herstellung: Epitaxie und Oxidation
- ▶ Problem: nicht plane Oberfläche

CMOS-Herstellungsprozess (cont.)

7. Si_3N_4 entfernen, Gateoxid bleibt SiO_2
8. Transistor Schwellspannungen „justieren“
 - ▶ Meist wird das Polysilizium zusätzlich n^+ dotiert
Grund: bessere Leitfähigkeit
 - ▶ Problem: $U_D(T_N) \approx 0,5 \dots 0,7 V$
 $U_D(T_P) \approx -1,5 \dots -2,0 V$
 - ▶ Maske: n-Wanne, bzw. p-Wanne
 - ▶ Herstellung: Epitaxie einer leicht negativ geladenen Schicht an der Substratoberfläche

CMOS-Herstellungsprozess (cont.)

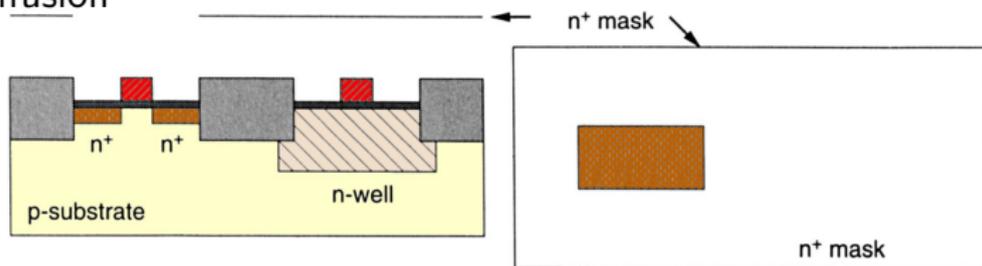
9. Polysilizium Gate



- ▶ Herstellung: Epitaxie von Polysilizium, Ätzen nach Planarprozess

CMOS-Herstellungsprozess (cont.)

10. n^+ -Diffusion

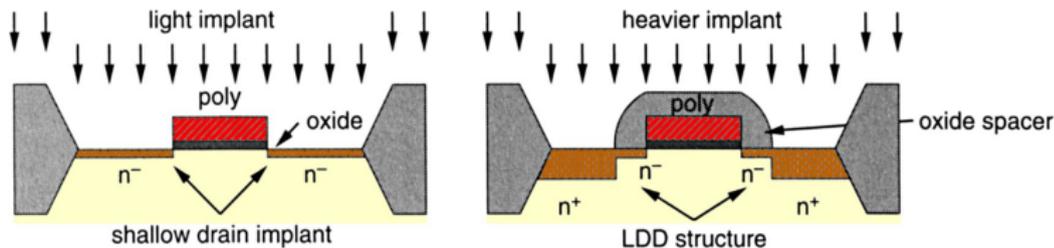


- ▶ Erzeugt Source und Drain der n-Kanal Transistoren
- ▶ Maskiert durch aktiven Bereich, n^+ -Maske und Polysilizium
 ⇒ Selbstjustierung
- ▶ Dotiert auch das Polysilizium Gate leicht (s.o.)
- ▶ Herstellung: Ionenimplantation, durchdringt Gateoxid

CMOS-Herstellungsprozess (cont.)

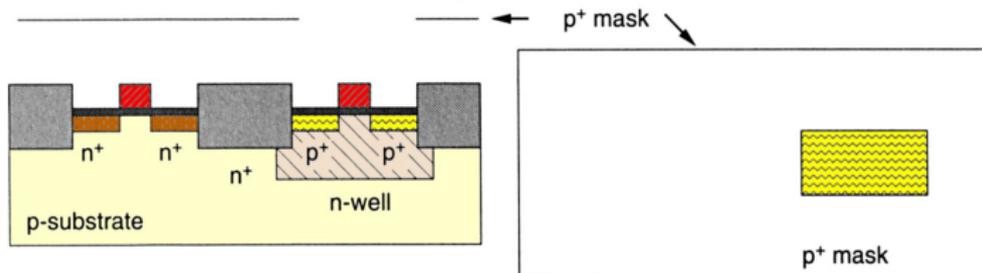
Zusätzliche Schritte bei der Source/Drain Herstellung

- ▶ Problem „Hot-Carrier“ Effekte (schnelle Ladungsträger): Stoßionisation, Gateoxid wird durchdrungen. . .
- ▶ Lösung: z.B. LDD (**L**ightly **D**oped **D**rain)
 - a. „flaches“ n-LDD Implant
 - b. zusätzliches SiO_2 über Gate aufbringen (*spacer*)
 - c. „normales“ n^+ -Implant
 - d. Spacer SiO_2 entfernen



CMOS-Herstellungsprozess (cont.)

11. p^+ -Diffusion



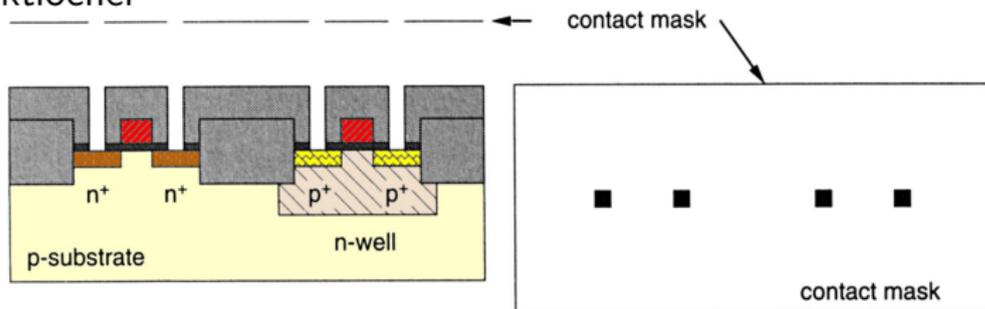
- ▶ Erzeugt Source und Drain der p -Kanal Transistoren
- ▶ Maskiert durch aktiven Bereich, p^+ -Maske und Polysilizium
 ⇒ Selbstjustierung
- ▶ teilweise implizite p^+ -Maske = \neg n^+ -Maske
- ▶ wenig schnelle Ladungsträger (Löcher), meist keine LDD-Schritte
- ▶ Herstellung: Ionenimplantation, durchdringt Gateoxid

CMOS-Herstellungsprozess (cont.)

12. SiO_2 aufbringen, Feldoxid

- ▶ Strukturen isolieren
- ▶ Herstellung: Epitaxie

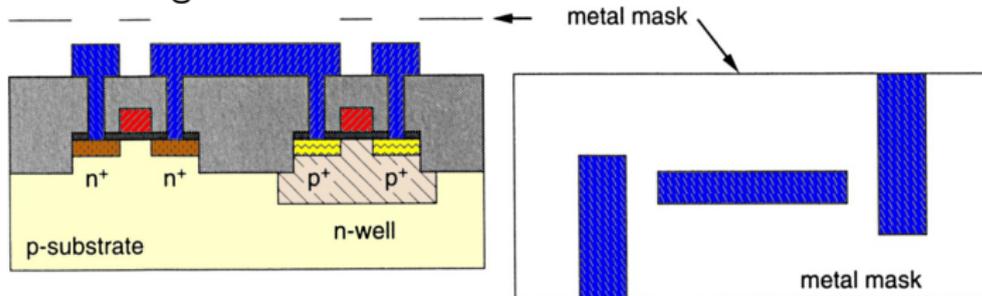
13. Kontaktlöcher



- ▶ Verbindet (spätere) Metallisierung mit Polysilizium oder Diffusion
- ▶ Anschlüsse der Transistoren: Gate, Source, Drain
- ▶ Herstellung: Ätzprozess

CMOS-Herstellungsprozess (cont.)

14. Metallverbindung



- ▶ Erzeugt Anschlüsse im Bereich der Kontaktlöcher
- ▶ Herstellung: Metall aufdampfen, Ätzen nach Planarprozess

15. weitere Metalllagen

- ▶ Weitere Metallisierungen, bis zu $7 \times$ Metall
- ▶ Schritte: 12. bis 14. wiederholen



CMOS-Herstellungsprozess (cont.)

16. Passivierung

- ▶ Chipoberfläche abdecken, Plasmanitridschicht

17. Pad-Kontakte öffnen

Inhalt

- ▶ Speicher
- ▶ Register-Transfer-Ebene
- ▶ Halbleitertechnologie
- ▶ CMOS-Schaltungen
- ▶ Programmierbare Logikbausteine
- ▶ Entwurf Integrierter Schaltungen

Programmierbare Logikbausteine

Kompromiss zwischen fest aufgebauter Hardware und Software-basierten Lösungen auf Computern

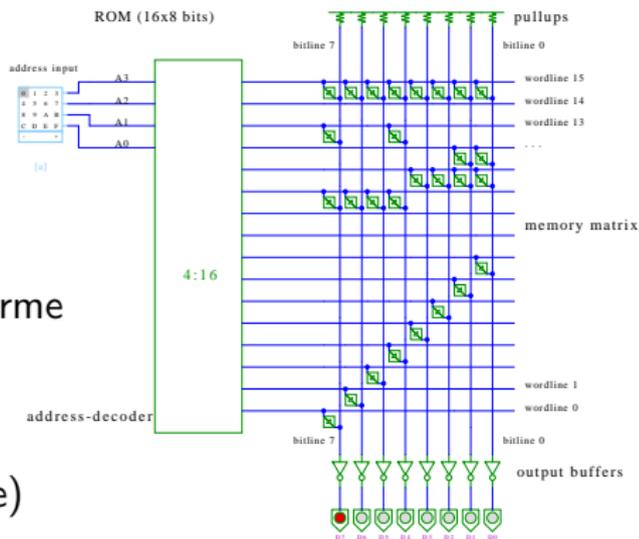
- ▶ Realisierung anwendungsspezifischer Funktionen und Systeme
 - ▶ gute bis sehr gute Performance
 - ▶ hoher Entwurfsaufwand
 - ▶ vom Anwender (evtl. mehrfach) programmierbar

- ▶ Klassifikation nach Struktur und Komplexität
 - ▶ PROM Programmable Read-Only Memory
 - ▶ PAL Programmable Array Logic
 - ▶ GAL Generic Array Logic
 - ▶ PLA Programmable Logic Array
 - ▶ CPLD Complex Programmable Logic Device
 - ▶ FPGA Field-Programmable Gate Array
 - ▶ ...

PROM: Programmable Read-Only Memory

- ▶ UND-ODER Struktur
- ▶ festes UND-Array
voll auskodiert
 2^n Terme
- ▶ programmierbare ODER-Terme

- ▶ auch: „LUT“ (look-up table)
- ▶ Hades Beispiel: $n = 4$, 16×8 bit



PAL: Programmable Array Logic

- ▶ disjunktive Form: UND-ODER Struktur
- ▶ UND-Ausgänge fest an die ODER-Eingänge angeschlossen
- ▶ Eingänge direkt und invertiert in die UND-Terme geführt
- ▶ Verknüpfungen der Eingänge zu den UND-Termen programmierbar

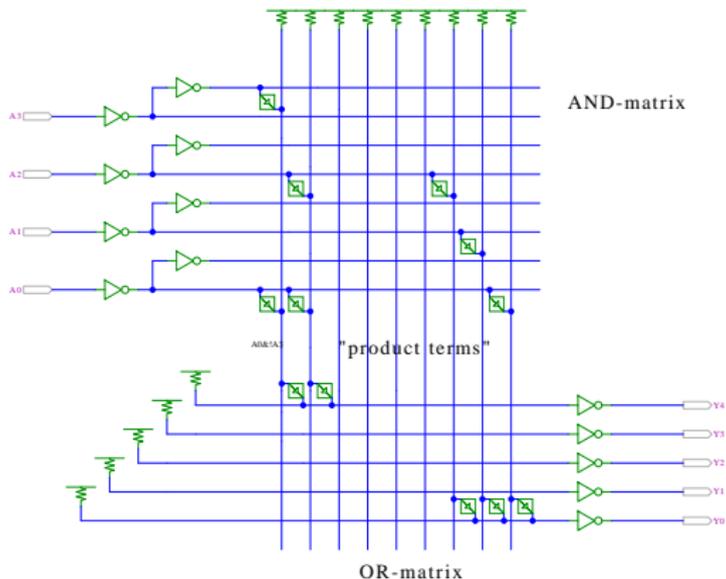
- ▶ heute durch GAL ersetzt (s.u.)

PLA: Programmable Logic Array

- ▶ disjunktive Form: logische UND-ODER Struktur
- ▶ Eingänge direkt und invertiert in die UND-Terme geführt
- ▶ Verknüpfungen Eingänge UND-Terme
- ▶ Verknüpfungen UND-Ausgänge zu ODER-Eingängen programmierbar

- ▶ in NMOS-Technologie sehr platzsparend realisierbar
- ▶ und zwar als NOR-NOR Matrix (de-Morgan Regel!)
- ▶ aber statischer Stromverbrauch
- ▶ in CMOS-Technologie kaum noch verwendet

PLA: Programmable Logic Array



(Hades tams.informatik.uni-hamburg.de/applets/hades/webdemos/42-programmable/10-pla/pla.html)

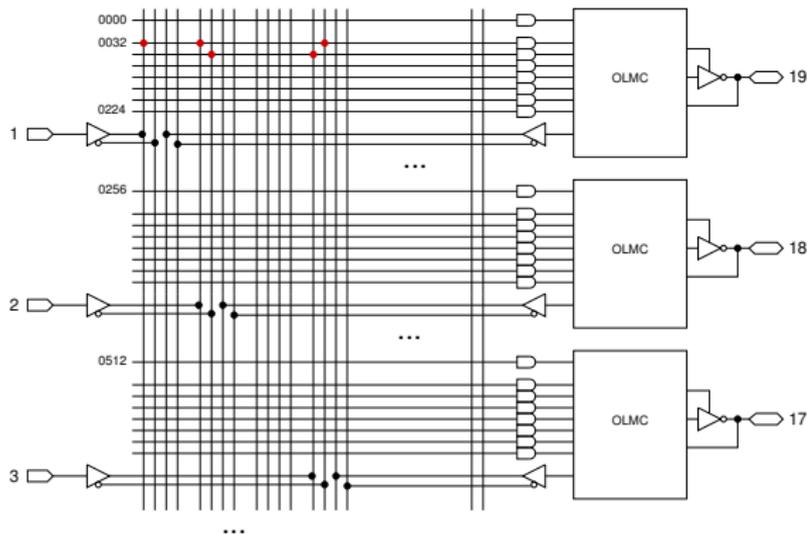
GAL: Generic Array Logic

- ▶ disjunktive UND-ODER Struktur
- ▶ externe Eingänge und Ausgangswerte direkt/invertiert
- ▶ „Fuses“ verbinden Eingangswerte mit den AND-Termen

- ▶ programmierbare Ausgabezellen (OLMC) mit je einem D-Flipflop
- ▶ Output-Enable über AND-OR Matrix steuerbar
- ▶ drei Optionen
 - ▶ synchron/kombinatorisch (Flipflop nutzen oder umgehen)
 - ▶ Polarität des Eingangs (D oder \bar{D} speichern)
 - ▶ Polarität des Ausgangs (Q oder \bar{Q} ausgeben)

- ▶ Beispiel: GAL16V8 mit 8 Ausgabezellen, je 7+1 OR-Terme pro Ausgabezelle, 32 Eingänge pro Term.

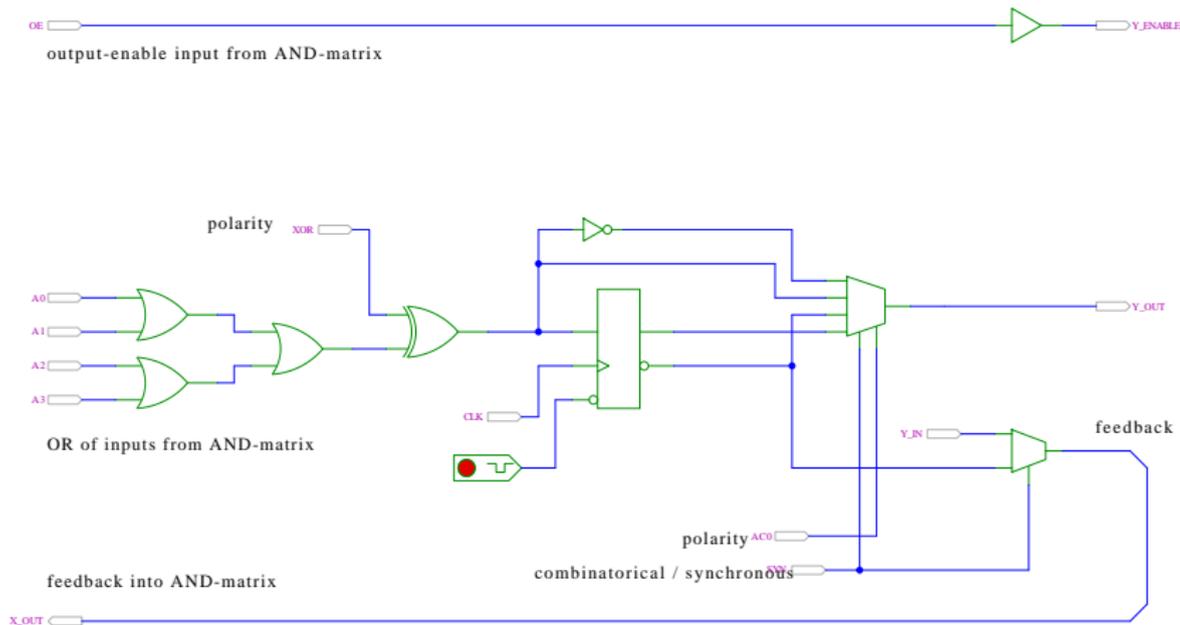
GAL: Blockschaltbild (Ausschnitt)



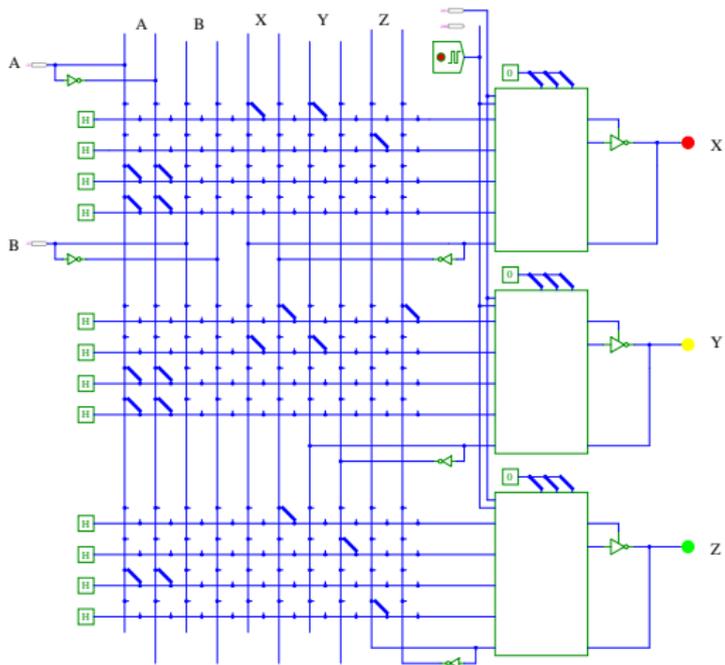
- ▶ programmierbare Sicherungen durchnummeriert
- ▶ kompakte Darstellung der UND-Terme: je eine Zeile
- ▶ Beispiel zweiter Term (ab 0032): $y = 1 \vee 2 \vee \bar{3}$

GAL: Ausgabezelle mit Flipflop

OLMC: Output-Logic-Macrocell



GAL: Beispiel Ampel

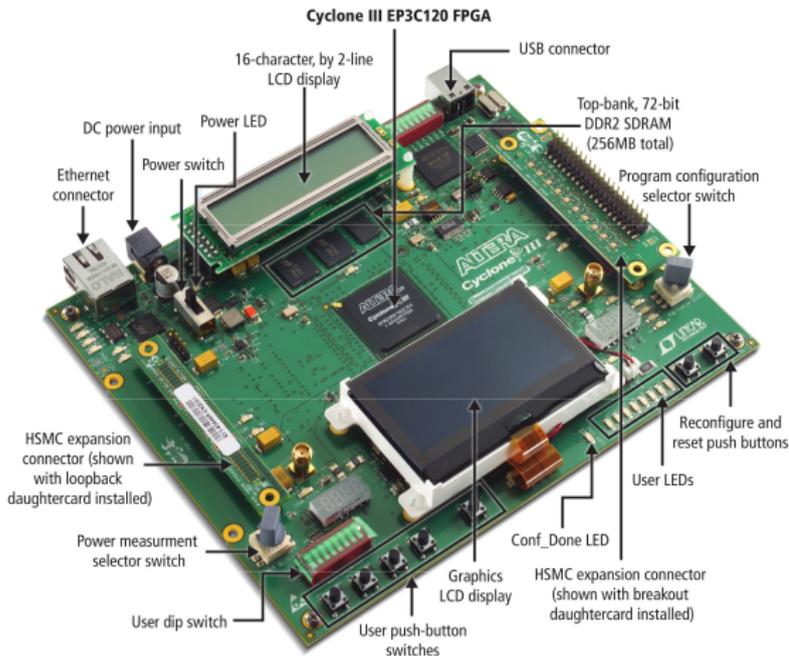


FPGA: Field-Programmable Gate-Array

Sammelbegriff für „große“ anwender-programmierbare Schaltungen

- ▶ Matrix von kleineren programmierbaren Zellen
- ▶ z.B. 64x1 SRAM als Lookup für Funktionen mit 4 Variablen
- ▶ Multiplexer-Netzwerk als programmierbare Verbindung
- ▶ zusätzliche „Makrozellen“:
 - ▶ Multiplizierer
 - ▶ schnelle serielle Kommunikation
 - ▶ eingebettete Multiplizierer
- ▶ Komplexität derzeit bis über 400 000 Gatteräquivalente
- ▶ Xilinx, Altera, weitere

FPGA: Altera Prototypenboard



Inhalt

- ▶ Speicher
- ▶ Register-Transfer-Ebene
- ▶ Halbleitertechnologie
- ▶ CMOS-Schaltungen
- ▶ Programmierbare Logikbausteine
- ▶ Entwurf Integrierter Schaltungen

Entwurf Integrierter Schaltungen

besonders anspruchsvoller Bereich der Informatik:

- ▶ Halbleiterfertigung benötigt vorab sämtliche Geometriedaten
- ▶ spätere Änderungen eines Chips nicht möglich
- ▶ Durchlauf aller Fertigungsschritte dauert Wochen bis Monate
- ▶ Entwürfe müssen komplett fehlerfrei sein

- ▶ spezielle Hardware-/System-Beschreibungssprachen
- ▶ Simulation des Gesamtsystems
- ▶ Analyse des Zeitverhaltens
- ▶ ggf. Emulation/Prototyping mit FPGAs

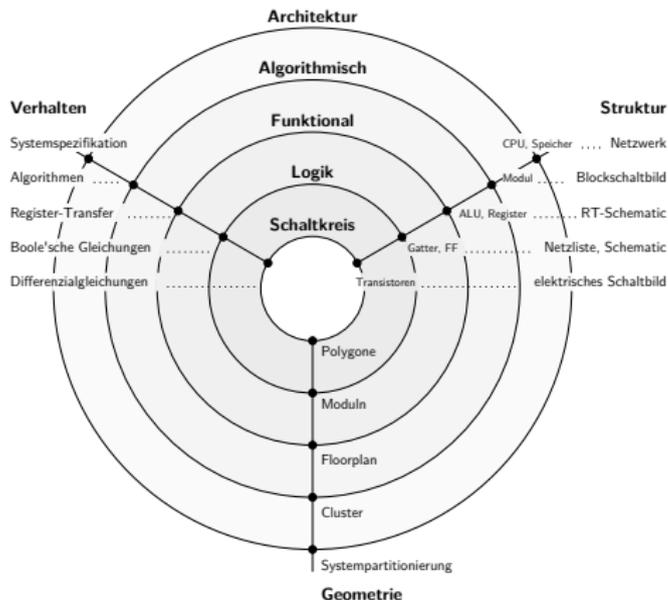
- ▶ Kombination von Hardware- oder Softwarerealisierung von Teilfunktionen, sog. **HW/SW-Codesign**

Entwurfsablauf

Wasserfallmodell

- ▶ Lastenheft
- ▶ Verhaltensmodell (Software)
- ▶ Aufteilung in HW- und SW-Komponenten
- ▶ funktionale Simulation/Emulation und Test
- ▶ Synthese oder manueller Entwurf der HW, Floorplan
- ▶ Generieren der „Netzliste“ (logische Struktur)
- ▶ Simulation mit Überprüfung der Gatter-/Leitungslaufzeiten
- ▶ Generieren und Optimierung des Layouts („Tapeout“)

Y-Diagramm



(Gajski, 1988)

Y-Diagramm: Deutung

- ▶ drei unterschiedliche Aspekte/Dimensionen:
 - 1 Verhalten
 - 2 Struktur (logisch)
 - 3 Geometrie (physikalisch)

- ▶ Ende des Entwurfsprozesses ist die vollständige Geometrie des Chips (ganz innen)
- ▶ benötigt für die Halbleiterfertigung (Planarprozess)

- ▶ Start möglichst abstrakt, z.B. als Verhaltensbeschreibung
- ▶ Entwurfsprogramme („EDA“, *electronic design automation*) unterstützen den Entwerfer: setzen Verhalten in Struktur und Struktur in Geometrien um

Entwurfstile

Was ist die „beste“ Realisierung einer gewünschten Funktionalität?

- ▶ mehrere konkurrierende Kriterien
 - ▶ Performance, Chipfläche, Stromverbrauch
 - ▶ Stückkosten vs. Entwurfsaufwand und Entwurfskosten
 - ▶ Zeitbedarf bis zur ersten Auslieferung und ggf. für Designänderungen
 - ▶ Schutz von Intellectual-Property
 - ▶ ...
- ▶ vier gängige Varianten:
 - ▶ full-custom Schaltungen
 - ▶ semi-custom Bausteine: Standardzellen, Gate-Arrays
 - ▶ anwender-programmierbare Bausteine: FPGA, PAL/GAL, ROM
 - ▶ als Software auf von-Neumann Rechner: RAM, ROM



Full-custom („Vollkunden-Entwurf“)

- ▶ vollständiger Entwurf der gesamten Geometrie eines Chips
- ▶ jeder Transistor einzeln „massgeschneidert“ und platziert
- ▶ vorgegeben sind lediglich die Entwurfsregeln (*design-rules*) des jeweiligen Herstellungsprozesses (Strukturbreite, Mindestabstände, usw.)
- ▶ gegebenenfalls Verwendung von Teilschaltungen/Makros des Herstellers
- ▶ minimale Chipfläche, beste Performance, kleinster Stromverbrauch
- ▶ geringste Stückkosten bei der Produktion
- ▶ aber höchste Entwurfs- und Maskenkosten
- ▶ erste Prototypen erst nach Durchlaufen aller Maskenschritte

Semi-custom: Standardzell-Entwurf

- ▶ Entwurf der Schaltung mit vorhandenen Grundkomponenten:
 - ▶ Basisbibliothek mit Gattern und Flipflops
 - ▶ teilweise (konfigurierbare) ALUs, Multiplizierer
 - ▶ teilweise (konfigurierbare) Speicher
- ▶ Entwurfsregeln sind der Bibliothek berücksichtigt
- ▶ Platzierung der Komponenten und Verdrahtung
- ▶ kleine Chipfläche, gute Performance, niedriger Stromverbrauch
- ▶ geringe Stückkosten
- ▶ hohe Maskenkosten (alle Masken erforderlich)
- ▶ erste Prototypen erst nach Durchlaufen aller Maskenschritte

- ▶ nur bei Massenprodukten wirtschaftlich, ab ca. 100.000 Stück
- ▶ z.B. Speicherbausteine (SRAM, DRAM), gängige Prozessoren

Semi-custom: Gate-Arrays

- ▶ Schaltung mit Gattern/Transistoren an festen Positionen
- ▶ Entwurf durch Verdrahten der vorhandenen Transistoren
- ▶ überzählige Transistoren werden nicht angeschlossen

- ▶ mittlere Chipfläche, mittlere Performance, mittlerer Stromverbrauch
- ▶ mittlere Stückkosten
- ▶ mittlere Maskenkosten (nur Verdrahtung kundenspezifisch)
- ▶ Prototypen schnell verfügbar (nur Verdrahtung)

- ▶ ab mittleren Stückzahlen wirtschaftlich, ab. ca. 1 000 Stück

FPGA: Field-Programmable Gate-Arrays

- ▶ Hunderte/Tausende von konfigurierbaren Funktionsblöcken
- ▶ Verschaltung dieser Blöcke vom Anwender programmierbar

- ▶ Entwurfsprogramme setzen Beschreibung des Anwenders auf die Hardware-Blöcke und deren Verschaltung um
- ▶ derzeit bis ca. 1M Gatter-Äquivalente möglich
- ▶ Taktfrequenzen bis ca. 100 MHz
- ▶ zwei dominierende Hersteller: Xilinx, Altera

- ▶ nicht benutzte Blöcke liegen brach
- ▶ Schaltung kann in Minuten neu programmiert/verbessert werden

FPGA: selbstgemacht: Projekt 64-189

- ▶ Ideen für einen Mikrochip? Zum Beispiel für Bildverarbeitung, 3D-Algorithmen, Parallelverarbeitung, usw.
- ▶ Hereinschnuppern: Projekt 64-189 *Entwurf eines Mikrorechners*
- ▶ eigenen Prozessor mit Befehlssatz etc. entwerfen und auf FPGA realisieren
- ▶ Demo-Boards von Altera und Xilinx und Entwurfssoftware sind bei uns am Department verfügbar
- ▶ interessante Entwürfe können sofort umgesetzt und getestet werden
- ▶ einfach bei TAMS oder TIS vorbeischauchen



Literatur: Quellen für die Abbildungen

- ▶ Andrew Tanenbaum,
Structured Computer Organization, 5th. edition, Pearson
Prentice Hall, 2006
- ▶ Steven Furber,
ARM System-on-Chip Architecture,
Addison-Wesley, 2000
- ▶ Andreas Mäder,
Vorlesung Rechnerstrukturen und Mikrosysteme (RAM),
Universität Hamburg, FB Informatik, 2008
- ▶ Norbert Reifschneider: *CAE-gestützte IC-Entwurfsmethoden*.
Prentice Hall; München, 1998

Literatur: Vertiefung

- ▶ Reiner Hartenstein, *Standort Deutschland: Wozu noch Mikro-Chips*, IT-Press Verlag, 1994 (vergriffen)
- ▶ N.E.H. Weste & K. Eshragian, *Principles of CMOS VLSI Design — A Systems Perspective*, Addison-Wesley Publishing, 1993
- ▶ G. Nicolescu and P.J. Mosterman (ed.), *Model-Based Design for Embedded Systems*, CRC Press, 2010
- ▶ C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1982
- ▶ Giovanni de Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994