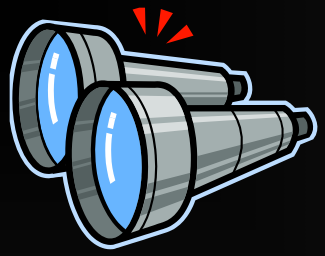
Digitales Design

Motivation
Grundlagen



© A. Steininger / TU Wien



Überblick

- Einordnung des Fachgebietes
- Ziele der LVA, Überblick
- Begriffsdefinitionen
- Kombinatorische Grundfunktionen
- Sequentielle Grundfunktionen
- Finite State Machines
- Boolsche Algebra

© A. Steininger / TU Wien

Einsatz digitaler Elektronik

- Multimedia & Spiele
  - Digitale Kamera, MP3, CD, DVD, Animation & Spezialeffekte...
- Telekommunikation
  - Handy/Telefonie, Internet, Zahlungsverkehr,...
- Automation & Steuerung
  - Leitsysteme, Stellwerke, ESP, „by-wire"-Funktionen, Haushalt...



© A. Steininger / TU Wien

Gesellschaftl. Auswirkungen

- 40 Jahre Microcontroller:
  - 15.11.1971: erster frei programmierbarer Microcontroller auf dem Markt (Intel 4004)

- Der Fortschritt, den wir in den nächsten 40 Jahren erzielen werden, wird den, den wir in den letzten 10000 Jahren erzielt haben, bei weitem übertreffen.

  [Justin Rattner, CTO Intel, 2011]

© A. Steininger / TU Wien

Verbreitung von Elektronik

„Last year, more transistors were produced – and at a lower cost – than grains of rice"

[Semiconductor Industry Association,
Annual Report 2005
www.sia-online.org]



Das sind etwa 10^{18}, oder 150 Millionen pro Mensch

…oder etwa 100 pro Ameise

© A. Steininger / TU Wien

Wie arbeitet ein Computer?

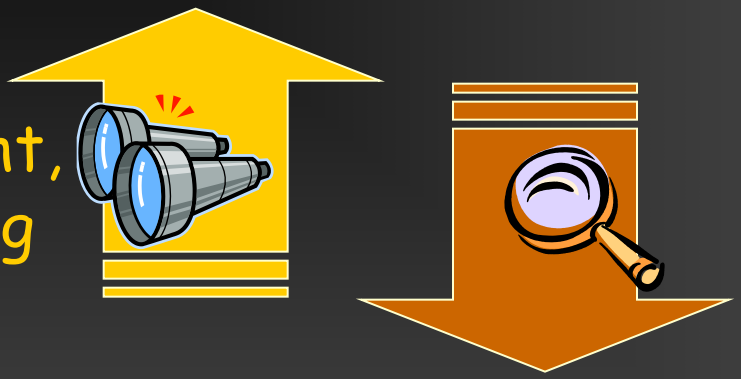
- Algorithmen — Regler, Signalverarbeitung...
- Hochsprache — Datentyp, Zuweisung, Programmfluss,...
- Assembler — Register-File, Befehl, Sprung...
- Registertransfer — Register, Datenpfad, ALU, ...
- Gatter — logische Pegel, Boolsche Verknüpfung
- Schalter — Schalter offen/geschlossen, Steuerung
- Transistoren — Schwellspannung, Ausgangsstrom,...
- Physik — elektromagn. Feld, Thermodynamik,...

© A. Steininger / TU Wien

Überblick oder Detail?

hohe Abstraktionsebene

mehr Übersicht, Funktionsbezug



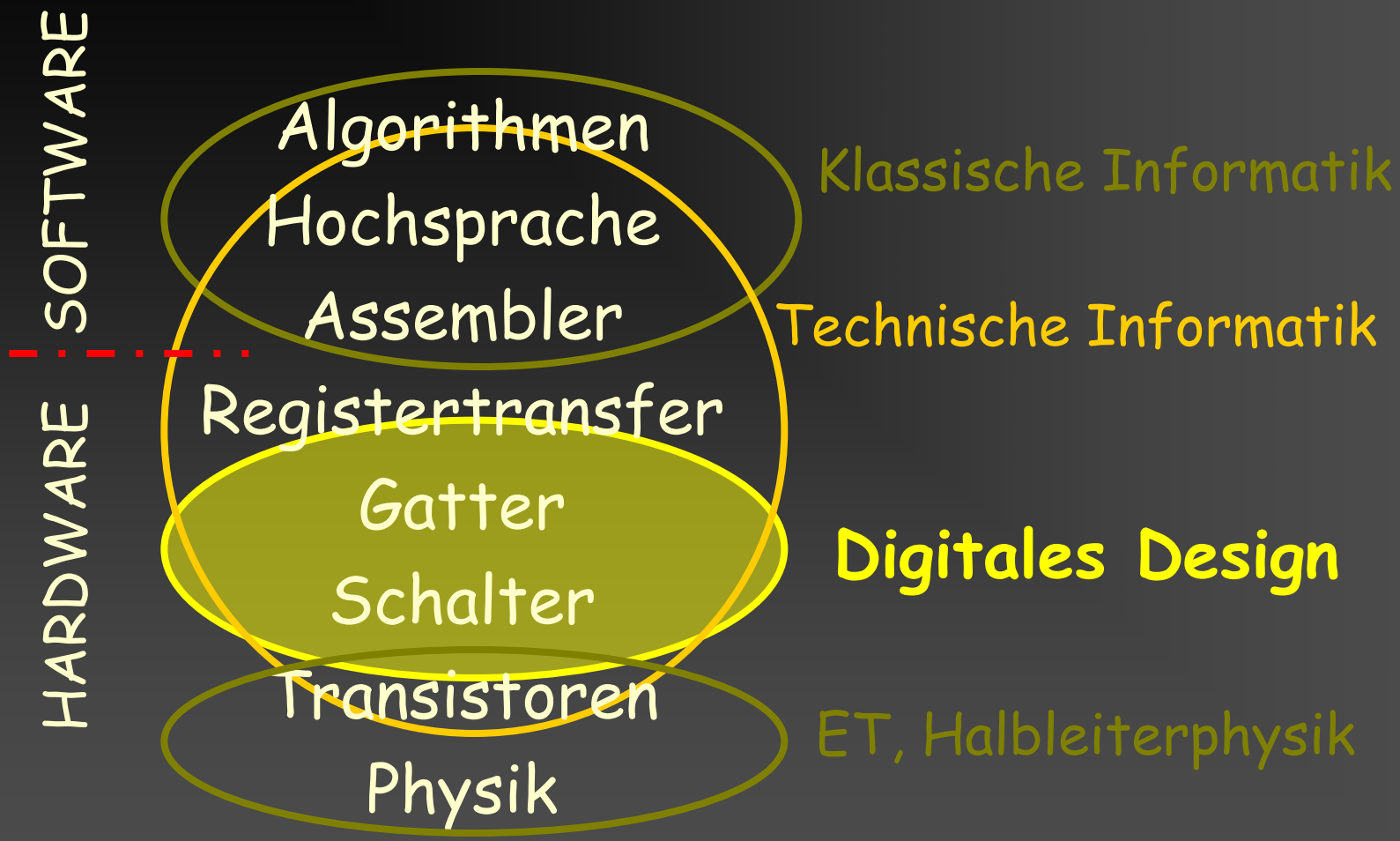
mehr Details, phys. Effekte, Geometrie

niedrige Abstraktionsebene

Wie produktiv wäre ein Programmierer, müsste er sich stets über den Verlauf des elektrischen Feldes im Kanal jedes MOS-Transistors in der Ziel-Hardware Gedanken machen?

© A. Steininger / TU Wien

Betrachtung in dieser VO



© A. Steininger / TU Wien

Ziele der LVA (VO + LU)



- Verständnis für Aufbau, Funktionieren und mögliches Fehlverhalten digitaler Hardware
- Erkennen der Grenzen der Hardware bzw. der beim Entwurf unterstellten Modelle
- Umgang mit zeitgemäßen Design-Methoden für digitale Hardware
- bestmögliche Ausnutzung der technologischen Möglichkeiten
- Abschätzen von Implementierungsaufwand

...als Basis z.B. für HW/SW-Codesign

© A. Steininger / TU Wien

Ziele der LVA im Detail



Die Absolventen dieser LVA können...

- den Entwurfs- und den Fertigungsprozess eines CMOS ASIC beschreiben, deren Schritte begründen und deren Herausforderungen nennen,
- die Abstraktion des Feldeffekt-Transistors als Schalter richtig verwenden und damit die grundlegende Funktion einfacher logischer Gatter erklären
- grundlegende Funktionselemente der digitalen Logik in Aufbau und Funktion beschreiben und richtig einsetzen

© A. Steininger / TU Wien

Ziele der LVA im Detail



Die Absolventen dieser LVA können...

- für eine gegebene Problemstellung eine (kombinatorische oder einfache sequenzielle) digitale Schaltung entwerfen und deren Implementierungsmöglichkeiten vergleichen,
- die Grenzen der beim digitalen Schaltungsentwurf verwendeten Modelle und Abstraktionen in der Anwendung erkennen und berücksichtigen,
- Entwurfsentscheidungen richtig treffen und deren Auswirkungen einschätzen,

© A. Steininger / TU Wien

Ziele der LVA im Detail



Die Absolventen dieser LVA können...

- Defekte und Funktionsfehler in digitalen integrierten Schaltungen in Verbindung mit den Schritten im Entwurfs- und Fertigungsprozess bringen, sowie deren wichtigste Einflussgrößen qualitativ und, wo möglich, auch quantitativ beschreiben.

© A. Steininger / TU Wien

Themen der Vorlesung

(1) Grundlagen

(2) ASIC-Fertigung

(3) CMOS-Logik

(4) Design-Flow

(5) Speicher-technologien

(6) ASIC-Zieltechnologien

(7) (VHDL)

(8) Datenblatt-Angaben

(9) Synchrones Design und seine Grenzen

(10) Defekte

(11) Test

(12) Logikanalysator

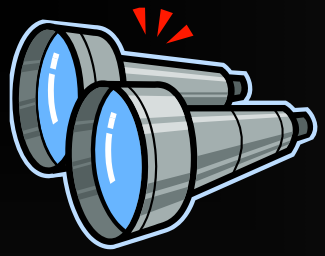
© A. Steininger / TU Wien

Allgemeine Erkenntnisse



- Laufzeit von Information
- Metastabilität
- Temperatureffekte
- Umgang mit parallelen Prozessen
- Modellierung von Alterungseffekten
- hierarchisches Design & Abstraktion
- Fehlermodellierung & Testprinzipien
- ...

© A. Steininger / TU Wien



Überblick

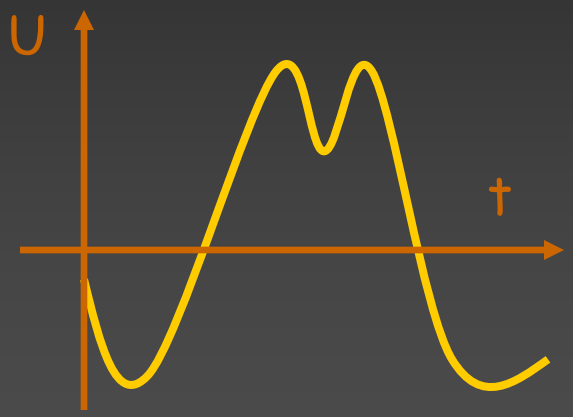
- Einordnung des Fachgebietes
- Ziele der LVA, Überblick
- **Begriffsdefinitionen**
- Kombinatorische Grundfunktionen
- Sequentielle Grundfunktionen
- Finite State Machines
- Boolsche Algebra

© A. Steininger / TU Wien

Analog versus digital

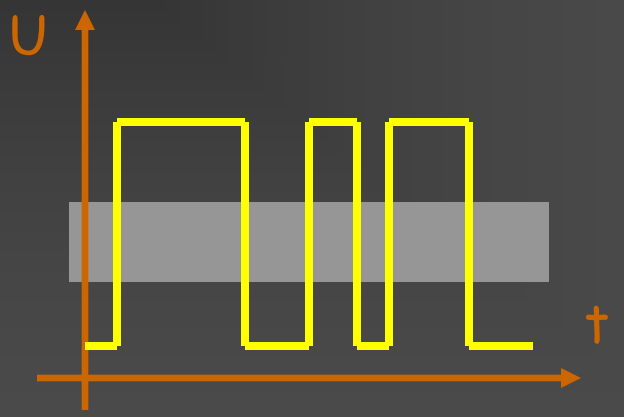
Analoge Signale

- kontinuierliche Betrachtung
- relevant ist exakter Spannungspegel



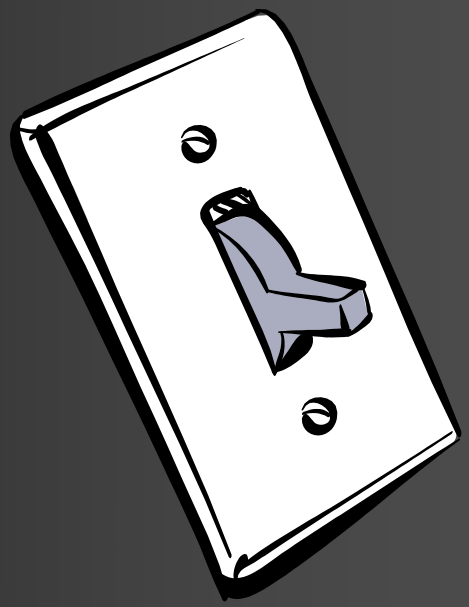
Digitale Signale

- „diskrete" Betrachtung im Wertebereich
- relevant ist nur Unterscheidung (HI/LO)



© A. Steininger / TU Wien

Warum digital ?



- Einfache Speicherung
- Robustheit gegen Störungen
- Reproduzierbarkeit
- Programmierbarkeit
- Mächtige Methoden zur Verarbeitung
- Einfache Schaltungstechnik/Technologie
- Rasanter Fortschritt
- Einfaches Design (?)

© A. Steininger / TU Wien

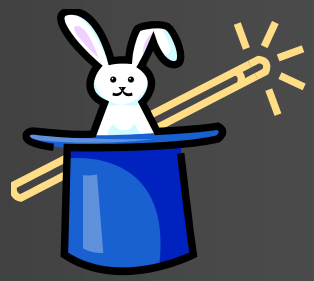
Ist die Welt „digital“?



- Reale physikalische Größen (Temperatur, Motordrehzahl,…) sind fast immer analog.
- Spannungsverläufe digitaler Signale sind analog.
- Digitale Bauelemente (Transistoren) arbeiten eigentlich analog.
- Aber: Die digitale Betrachtung (Diskretisierung) erlaubt einen höheren Abstraktionsgrad.
- Diese Abstraktion impliziert aber Idealisierungen und Annahmen (eingeschwungen, störungsfrei,…).

© A. Steininger / TU Wien

Die Kunst des Digital Design



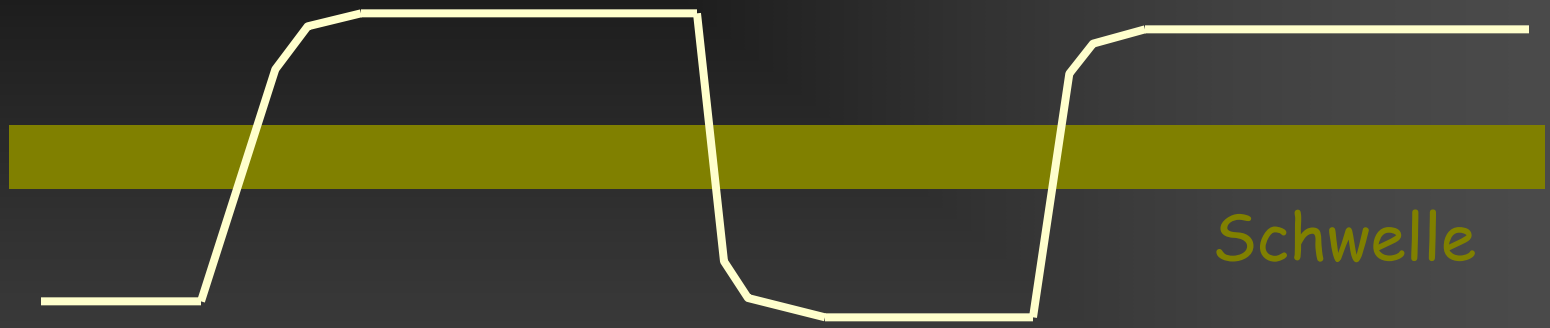
...besteht darin,

- ein Design so auszulegen, dass die unterstellten Idealisierungen möglichst gut erfüllt sind;
- zu wissen, welche nicht-idealen Effekte wichtig sind;
- die Standard-Funktionen zu verstehen und effizient einzusetzen;
- durch Kenntnis von Design- und Fertigungsprozess die Gesamtkosten des *Produktes* minimal zu halten.

© A. Steininger / TU Wien

Logikpegel

„1“: analoger Signalpegel oberhalb der Schwelle



„0“: analoger Signalpegel unterhalb der Schwelle

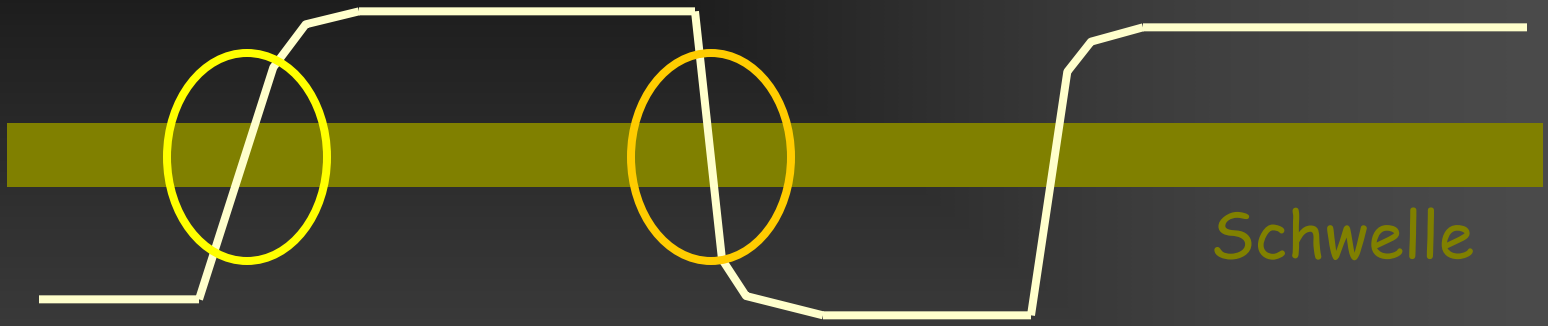
„Schwelle“ ist in Realität ein Bereich (siehe später).
Bei neg. Logik Zuordnung umgekehrt (nicht weiter betrachtet).

© A. Steininger / TU Wien

Flanken



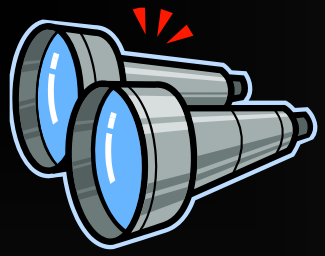
Steigend: Übergang von „0“ auf „1“



Fallend: Übergang von „1“ auf „0“

Aktive Flanke: steigende oder fallende Flanke wird (willkürlich) als Referenzpunkt z.B. für Datenübernahme festgelegt.

© A. Steininger / TU Wien



Überblick

- Einordnung des Fachgebietes
- Ziele der LVA, Überblick
- Begriffsdefinitionen
- **Kombinatorische Grundfunktionen**
- Sequentielle Grundfunktionen
- Finite State Machines
- Boolsche Algebra

© A. Steininger / TU Wien

Verknüpfung von 2 Variablen

e_1	e_2	$e_1 \wedge e_2$
0	0	0
0	1	0
1	0	0
1	1	1

AND

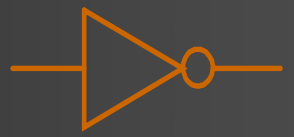
© A. Steininger / TU Wien

Verknüpfung von 2 Variablen

e_1	e_2	const	$e_1 \wedge e_2$	$e_1 \wedge \neg e_2$	e_1	$\neg e_1 \wedge e_2$	e_2	$\neg(e_1 = e_2)$	$e_1 \vee e_2$	$\neg(e_1 \vee e_2)$	$e_1 = e_2$	$\neg e_2$	$e_1 \vee \neg e_2$	$\neg e_1$	$\neg e_1 \vee e_2$	$\neg(e_1 \wedge e_2)$	const
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
			AND		BUF		BUF	XOR	OR	NOR	XNOR	INV		INV		NAND	

© A. Steininger / TU Wien

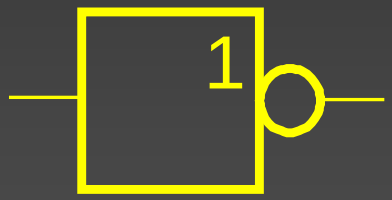
Der Inverter (NOT)



- Boolesche Funktion: $a = \neg e$
- Wahrheitstabelle:

e	a
0	1
1	0

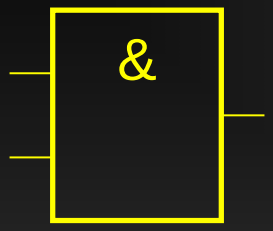
- Schaltsymbol:
  (lt. IEC 617-12)



© A. Steininger / TU Wien

Konjunktion und Disjunktion

AND

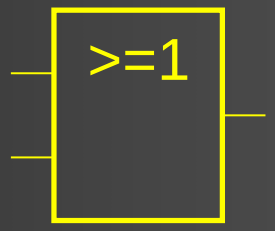


$a = e_1 \wedge e_2$

e_1	e_2	a
0	0	0
0	1	0
1	0	0
1	1	1



OR



$a = e_1 \vee e_2$

e_1	e_2	a
0	0	0
0	1	1
1	0	1
1	1	1



© A. Steininger / TU Wien

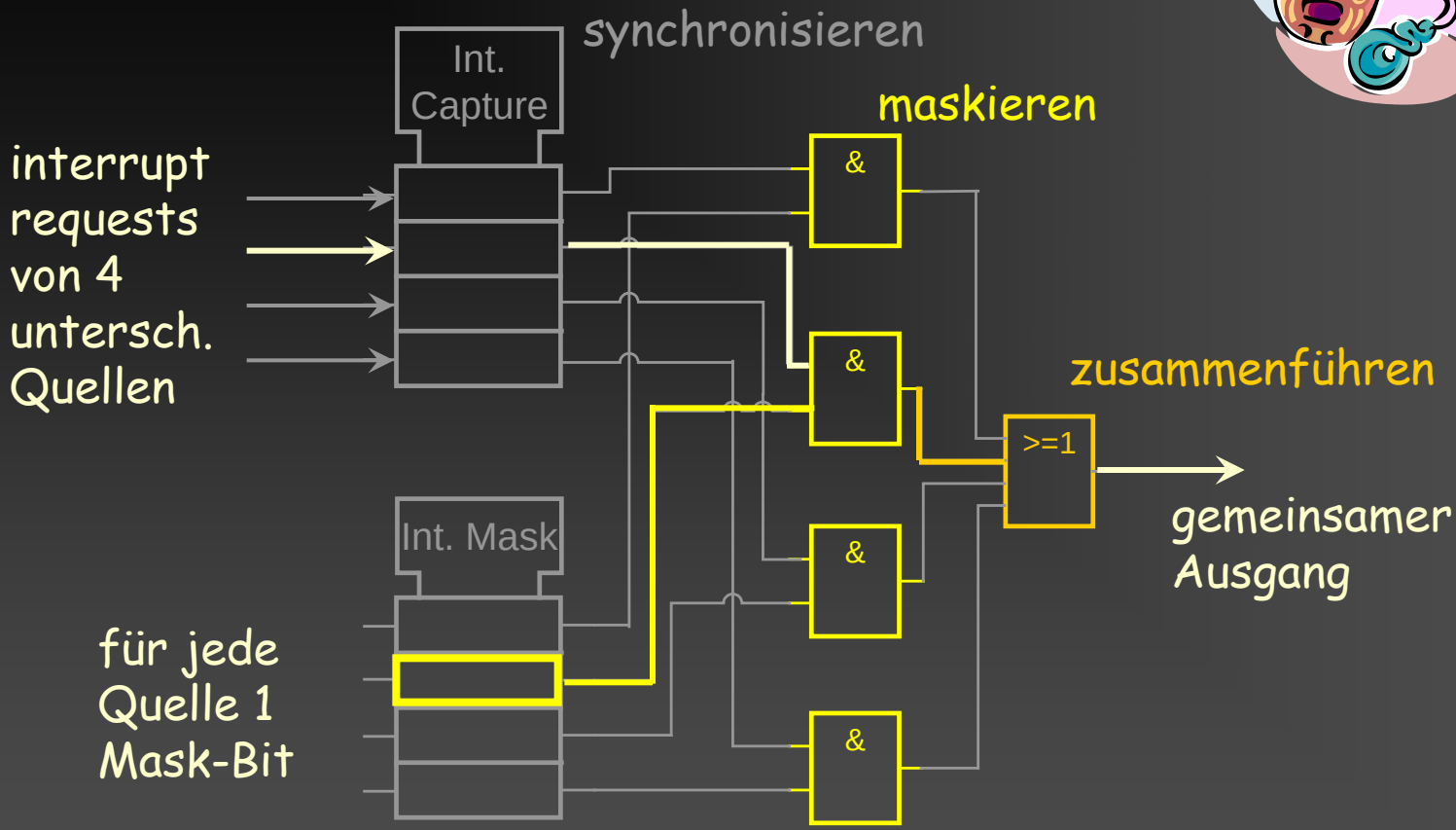
AND/OR: typische Anwendung

- Maskieren:
  - AND: 0 an einem Eingang erzwingt 0 am Ausgang; „sperrt" also die anderen Eingänge.
  - OR: 1 an einem Eingang erzwingt 1 am Ausgang; „sperrt" also die anderen Eingänge.
- Zusammenführen mehrerer Quellen:
  - OR: Sobald einer der Eingänge auf 1 geht, geht auch der Ausgang auf 1.
  - AND: Sobald einer der Eingänge auf 0 geht, geht auch der Ausgang auf 0.

© A. Steininger / TU Wien

Beispiel: Interrupt-Mask

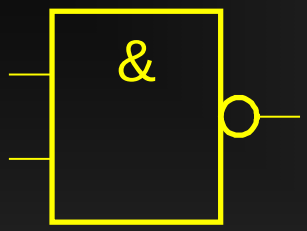




© A. Steininger / TU Wien

Invertierte Funktionen

NAND

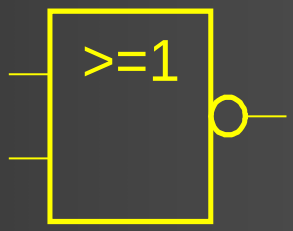


$a = \neg (e_1 \wedge e_2)$

e_1	e_2	a
0	0	1
0	1	1
1	0	1
1	1	0



NOR



$a = \neg (e_1 \vee e_2)$

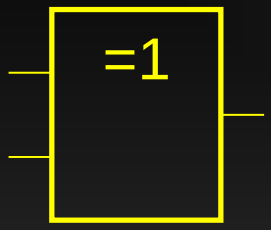
e_1	e_2	a
0	0	1
0	1	0
1	0	0
1	1	0



© A. Steininger / TU Wien

Antivalenz und Äquivalenz

XOR

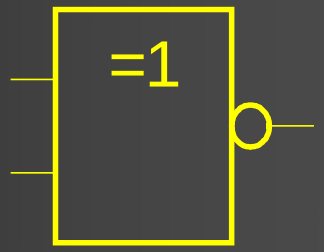


$a = (e_1 \vee e_2) \wedge \neg(e_1 \wedge e_2)$

e_1	e_2	a
0	0	0
0	1	1
1	0	1
1	1	0



XNOR



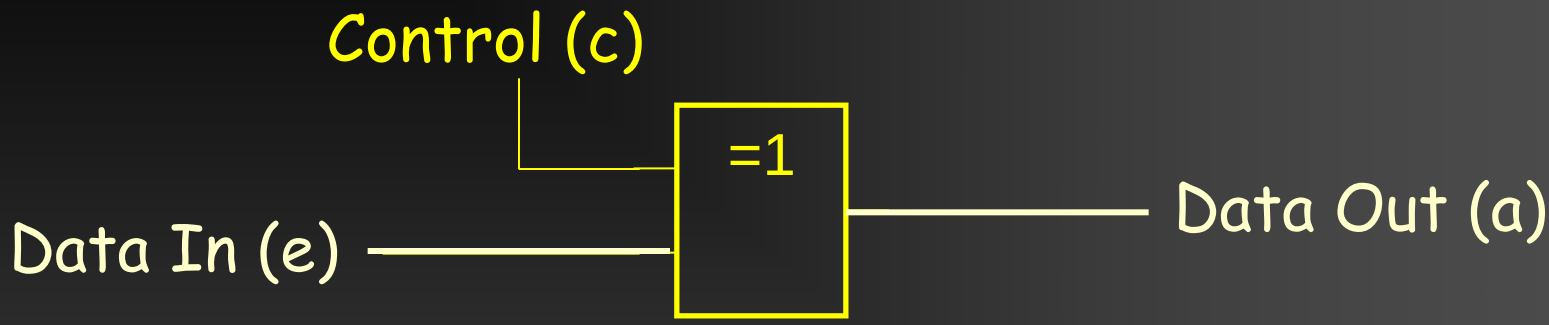
$a = \neg(e_1 \vee e_2) \vee (e_1 \wedge e_2)$

e_1	e_2	a
0	0	1
0	1	0
1	0	0
1	1	1



© A. Steininger / TU Wien

Anw-Bsp.: Gesteuerte Inversion



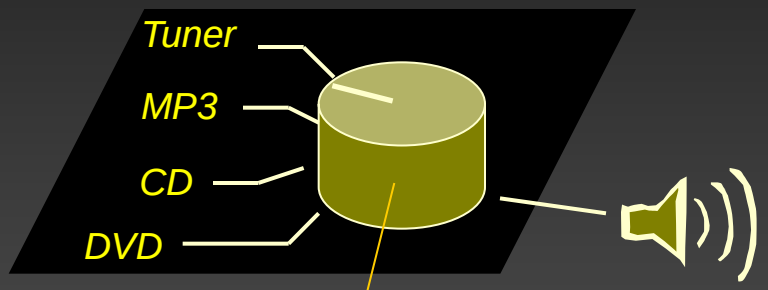
	c	e	a
normal	0	0	0
	0	1	1
invertieren	1	0	1
	1	1	0

© A. Steininger / TU Wien

Multiplexer: Funktion

„Wahlschalter“:

- Von 2^n Eingängen wird genau einer auf den Ausgang abgebildet.



- Die Auswahl erfolgt über *n* Steuersignale.

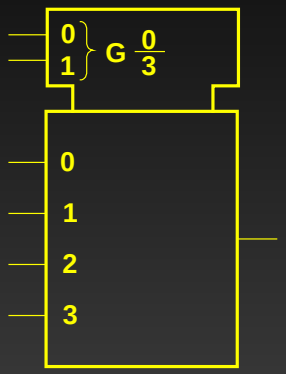
Beispiel 4:1 MUX

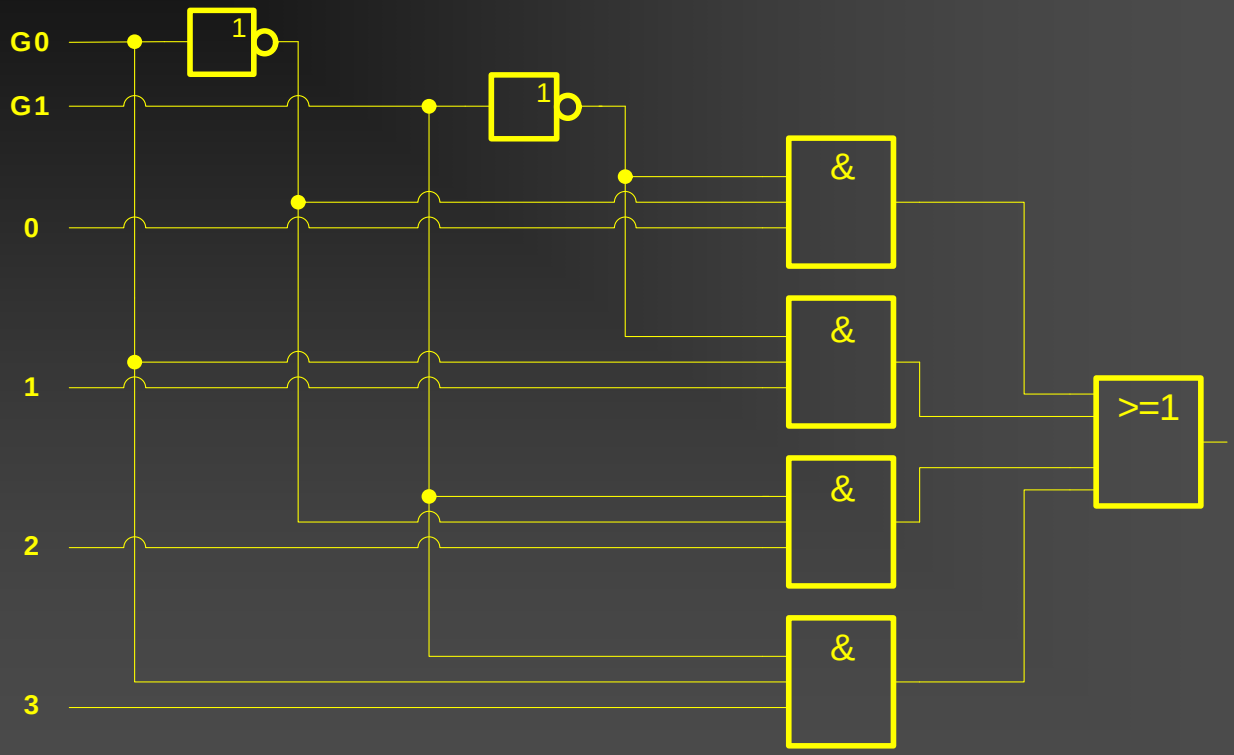
e_1	e_2	e_3	e_4	s_1	s_2	a
0	x	x	x	0	0	0
1	x	x	x	0	0	1
x	0	x	x	0	1	0
x	1	x	x	0	1	1
x	x	0	x	1	0	0
x	x	1	x	1	0	1
x	x	x	0	1	1	0
x	x	x	1	1	1	1

© A. Steininger / TU Wien

Multiplexer: Prinzipschaltung

- Schaltsymbol und Prinzip eines 4:1 Mux





© A. Steininger / TU Wien

Multiplexer für mehrere Bit

- Ein *k* :1 Multiplexer kann aus *k* Quellen mit je einem Bit gezielt eine selektieren.
- Zum Umschalten zwischen Quellen mit *b* Bit breiten Datenpfaden betreibt man *b* solche Multiplexer parallel:
  - jeder Mux wählt zwischen Bits mit gleichem Index [0…*b*-1], jeweils von allen *k* Quellen
  - die Steuersignale sind für alle Mux gleich

© A. Steininger / TU Wien

k-aus-n Threshold-Gate

„Schwellwertschalter“:

- Wenn *k* oder mehr der *n* Eingänge auf 1 sind, geht der Ausgang auf 1.
- Spezialfall AND: *n*-aus-*n* Threshold-Gate
- Spezialfall OR: 1-aus-*n* Threshold-Gate

Beispiel 2-aus-3 THG

e_1	e_2	e_3	a
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

© A. Steininger / TU Wien

Parity

Der Ausgang stellt sich so ein, dass die Anzahl der 1en an Eingang und Ausgang zusammen

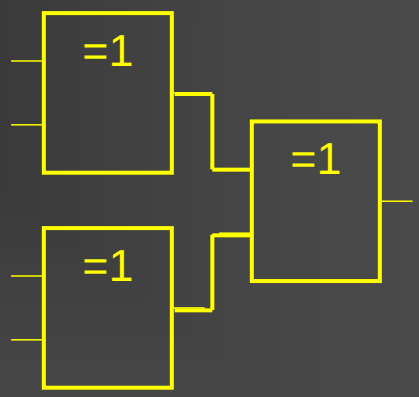
- gerade (even parity) bzw.
- ungerade (odd parity) ist.

In diesem Zusammenhang kann

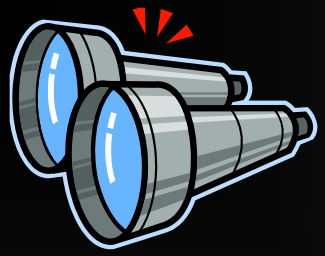
- XOR als elementares Element für even Parity und
- XNOR für odd Parity

angesehen werden

Beispiel:
even parity
4-bit



© A. Steininger / TU Wien



Überblick

- Einordnung des Fachgebietes
- Ziele der LVA, Überblick
- Begriffsdefinitionen
- Kombinatorische Grundfunktionen
- **Sequentielle Grundfunktionen**
- Finite State Machines
- Boolsche Algebra

© A. Steininger / TU Wien

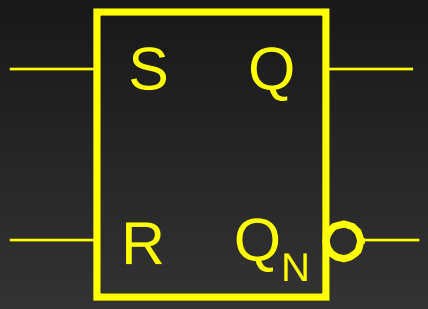
Sequentielle Logik

- bisher: „Kombinatorische Logik“
  - Ausgang a allein durch Eingänge e_i bestimmt,
  - unabhängig von der Vorgeschichte
  - keine Speicherelemente oder Rückkopplungen
- Gegensatz: „Sequentielle Logik“
  - enthält Speicherelemente s_i („Gedächtnis“), explizit oder in Form von Rückkopplungen
  - hat daher einen inneren Zustand $S = (s_1, s_2 \ldots s_n)$, der durch die Vorgeschichte bestimmt ist
  - Ausgang a hängt von den Eingängen e_i und dem aktuellen inneren Zustand S ab

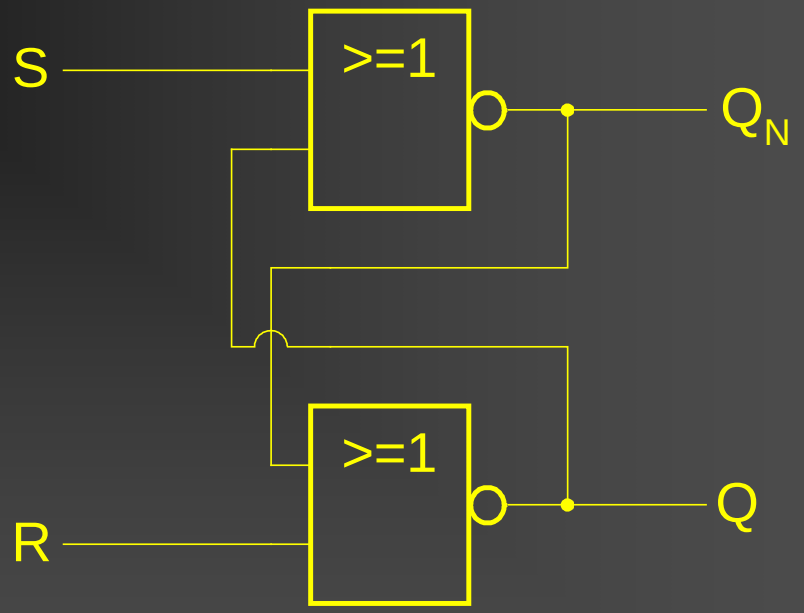
© A. Steininger / TU Wien

SR-Latch: Schaltung

Schaltsymbol:



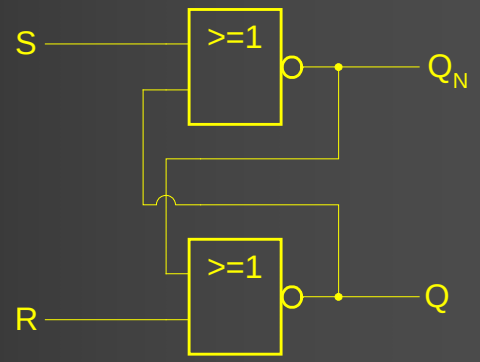
Prinzipschaltung:



S … Set-Eingang
R … Reset-Eingang
Q … Ausgang
Q_N … „invertierter" Ausgang

© A. Steininger / TU Wien

SR-Latch: Funktion



S	R	Q	Q_N	
0	0	last Q	last Q_N	Halten (hold)
0	1	0	1	Löschen (reset)
1	0	1	0	Setzen (set)
1	1	0	0	Set & Reset

„Setzen": Puls an S setzt Q auf 1

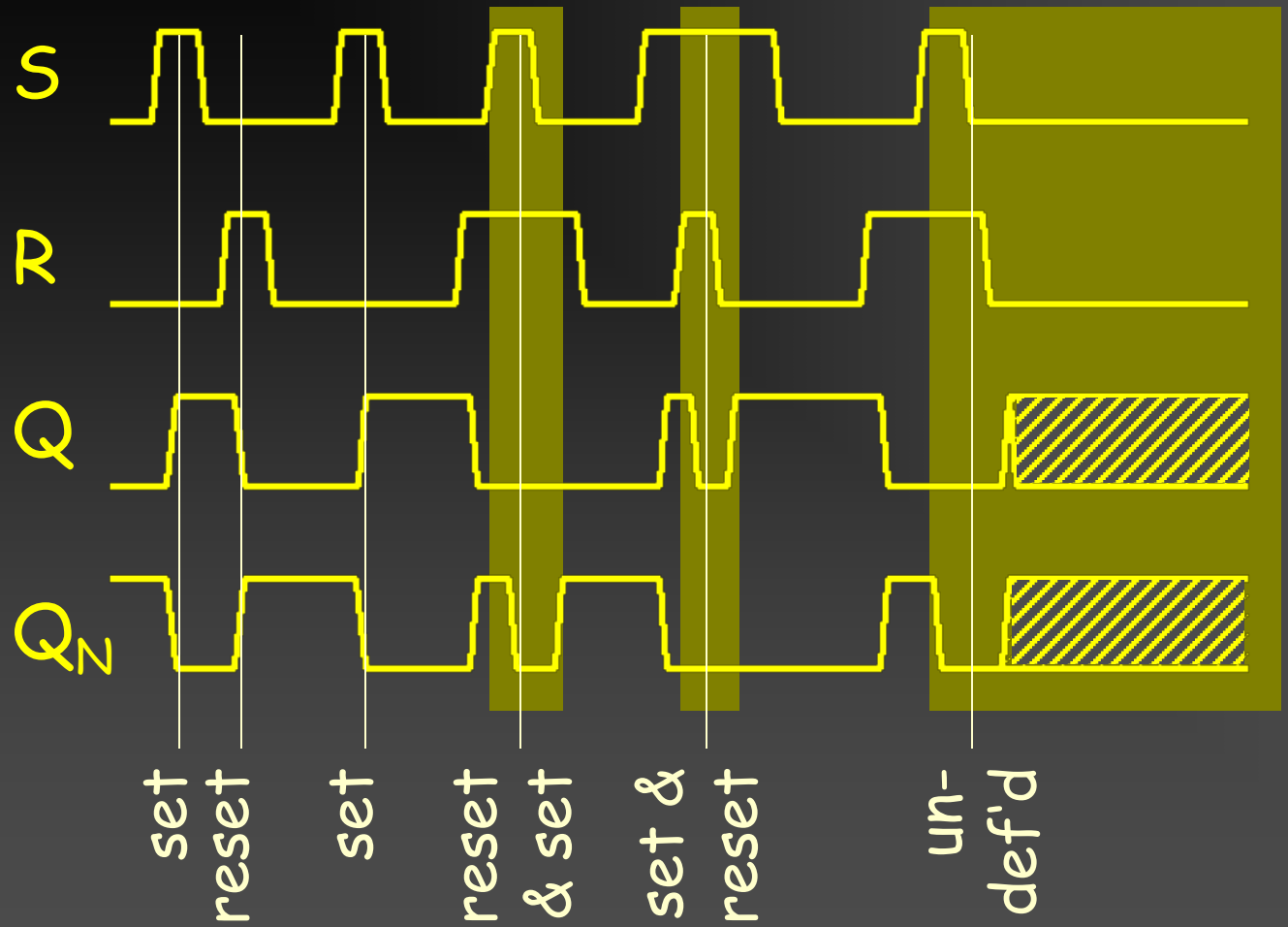
„Löschen": Puls an R setzt Q auf 0

„Halten": S und R inaktiv ⇨ Q hält letzten Zustand

Problematisch: S und R gleichzeitig aktiv ➡ $Q = Q_N$!

© A. Steininger / TU Wien

SR-Latch: Timing



© A. Steininger / TU Wien

SR-Latch mit Enable: Funktion

S	R	EN	Q	Q_N	
0	0	1	last Q	last Q_N	Halten (hold)
0	1	1	0	1	Löschen (reset)
1	0	1	1	0	Setzen (set)
1	1	1	0	0	Set & Reset
x	x	0	last Q	last Q_N	Halten (hold)

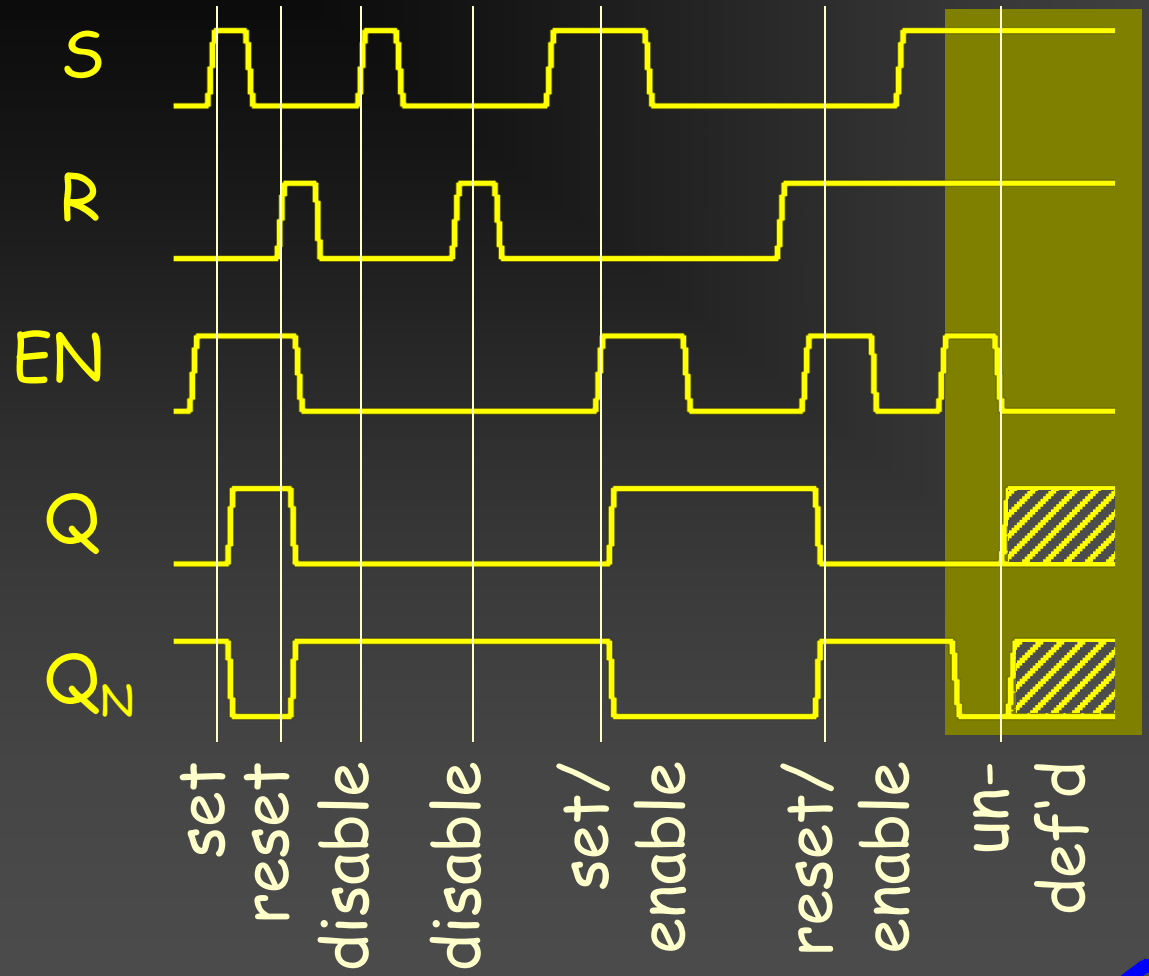
Setzen und Löschen über R und S wie zuvor

Halten: mit S und R inaktiv wie zuvor UND

in jedem Fall solange Enable EN = 0

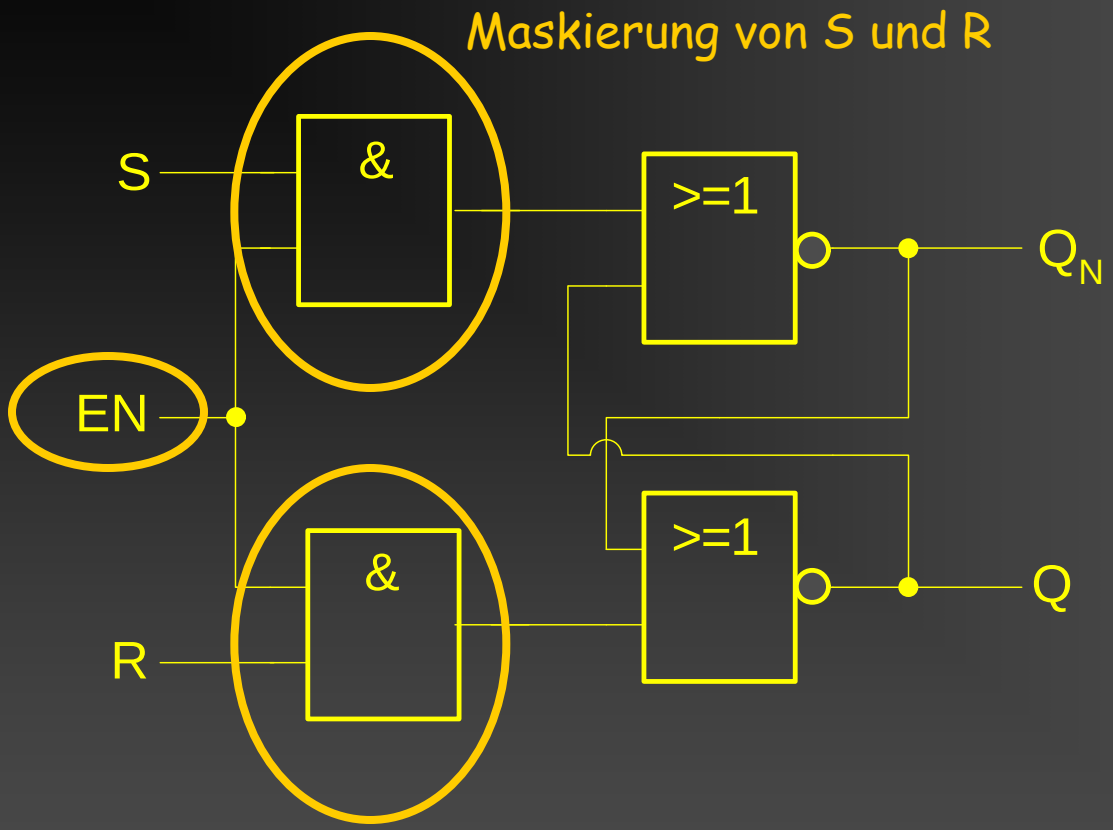
© A. Steininger / TU Wien

SR-Latch mit Enable: Timing



© A. Steininger / TU Wien

SR-Latch mit Enable: Prinzip

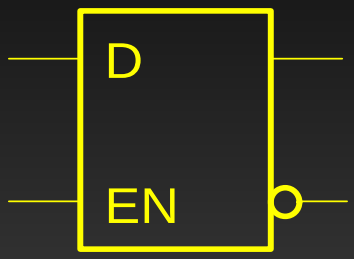


Verbesserungspotential: R = ¬S

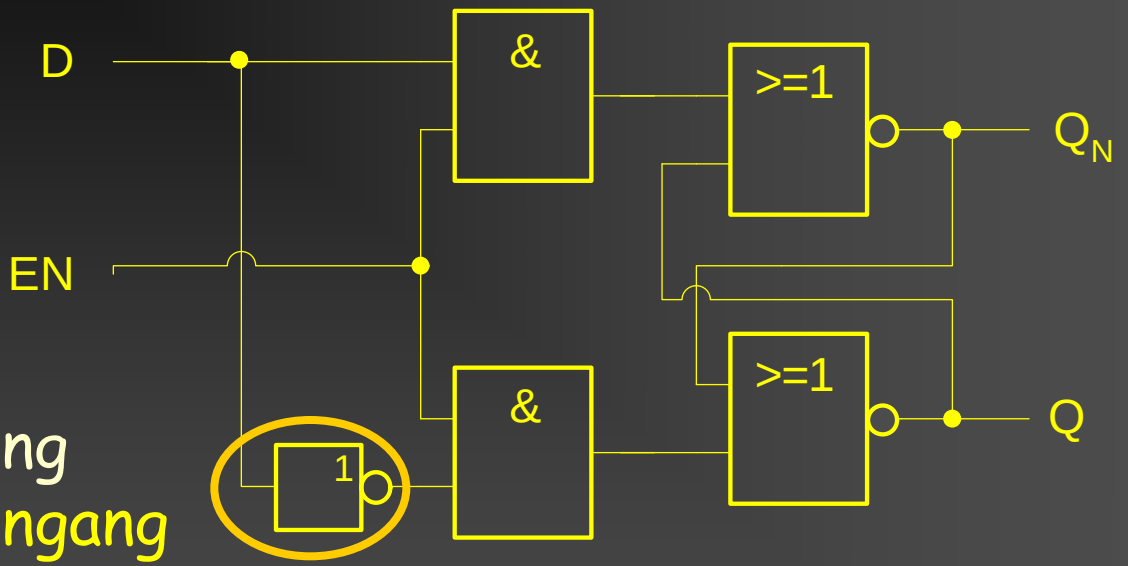
© A. Steininger / TU Wien

D-Latch: Schaltung

Schaltsymbol:



Prinzipschaltung:



D … Data-Eingang
EN … Enable-Eingang
Q … Ausgang
Q_N … invertierter Ausgang

© A. Steininger / TU Wien

D-Latch: Funktion

EN	D	Q	Q_N	
0	x	last Q	last Q_N	Halten (hold)
1	0	0	1	Löschen (reset)
1	1	1	0	Setzen (set)

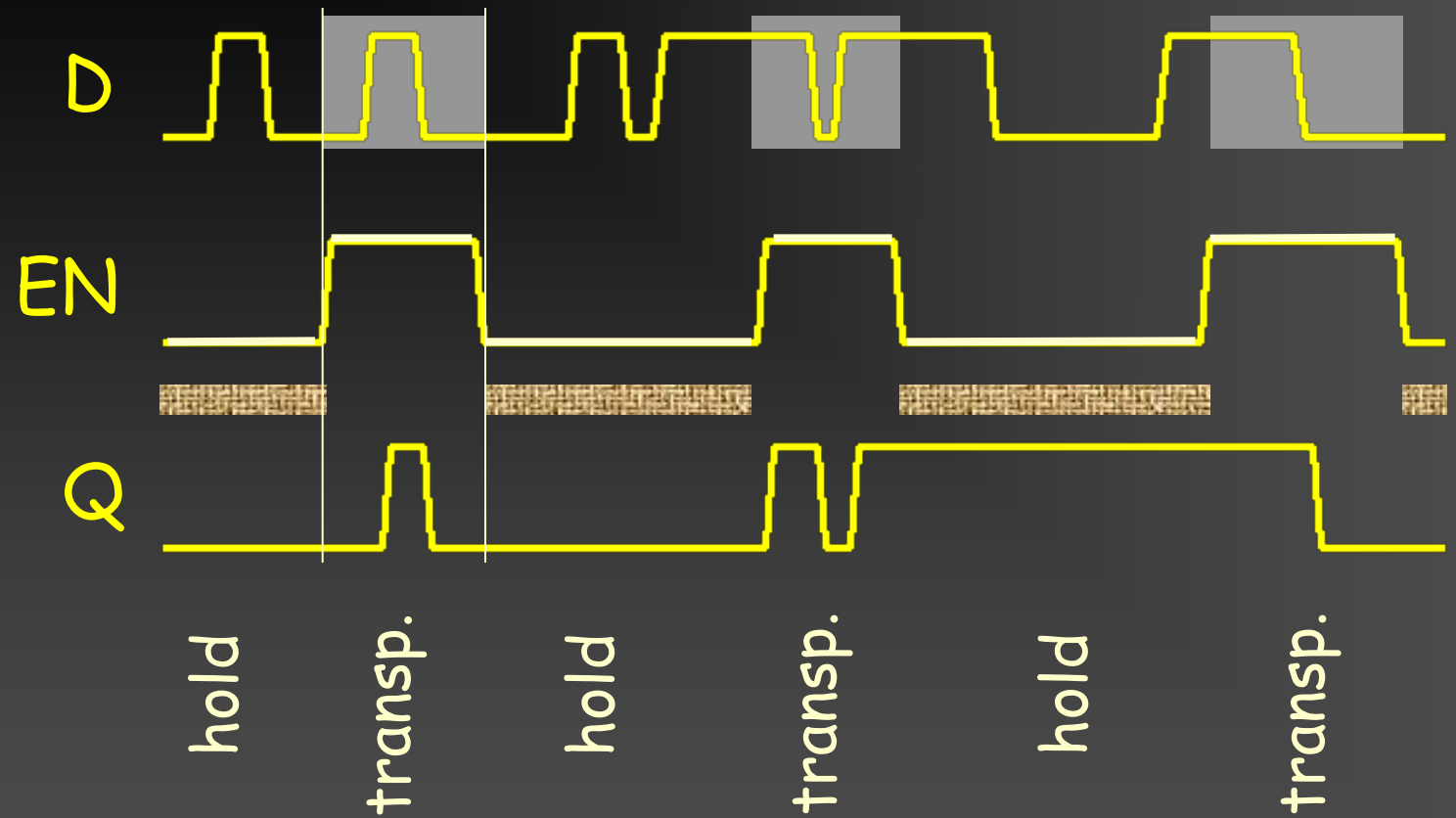
„Transparent": Bei EN = 1 wird D direkt auf Q abgebildet: Q = D

„Halten": Bei EN = 0 hält Q seinen letzten Zustand

Gleichzeitiges Set und Reset ist nicht mehr möglich, es gilt stets $Q_N = \neg Q$

© A. Steininger / TU Wien

D-Latch: Timing



© A. Steininger / TU Wien

Latch versus Flip-Flop



Latch

- Pegel der Eingangssignale allein bewirkt Zustandsänderung
- Asynchrones Verhalten
  Es gibt kein Taktsignal
- Für synchrone Designs nicht empfehlenswert



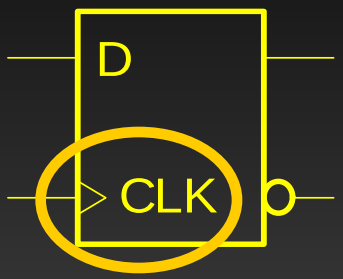
Flip-Flop

- Pegel der Eingangssignale bewirkt erst bei aktiver Taktflanke Zustandsänderung
- Synchrones Verhalten
- Flip-Flop ist das zentrale Element in synchronen Designs

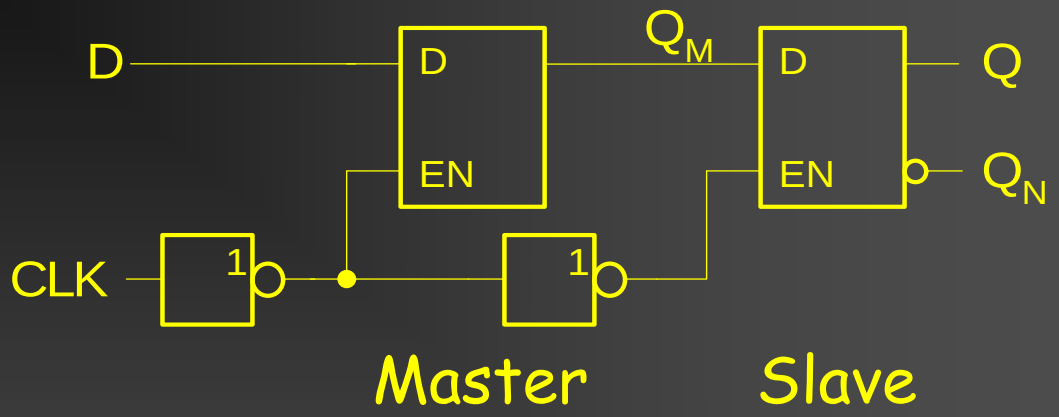
© A. Steininger / TU Wien

D-Flip-Flop: Schaltung

Schaltsymbol:



Prinzipschaltung:



D … Data-Eingang
CLK … Takteingang (clock)
Q … Ausgang
Q_N … invertierter Ausgang

© A. Steininger / TU Wien

D-Flip-Flop: Funktion

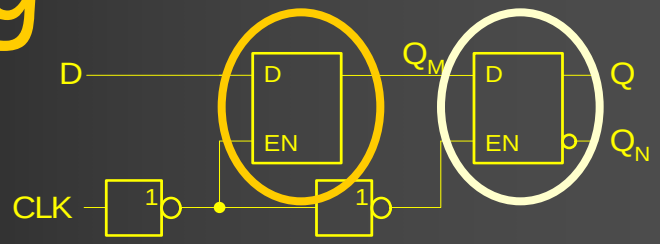
D	CLK	Q	Q_N	
0	↑	0	1	Übernehmen (capture)
1	↑	1	0	Übernehmen (capture)
x	0	last Q	last Q_N	Halten (hold)
x	1	last Q	last Q_N	Halten (hold)

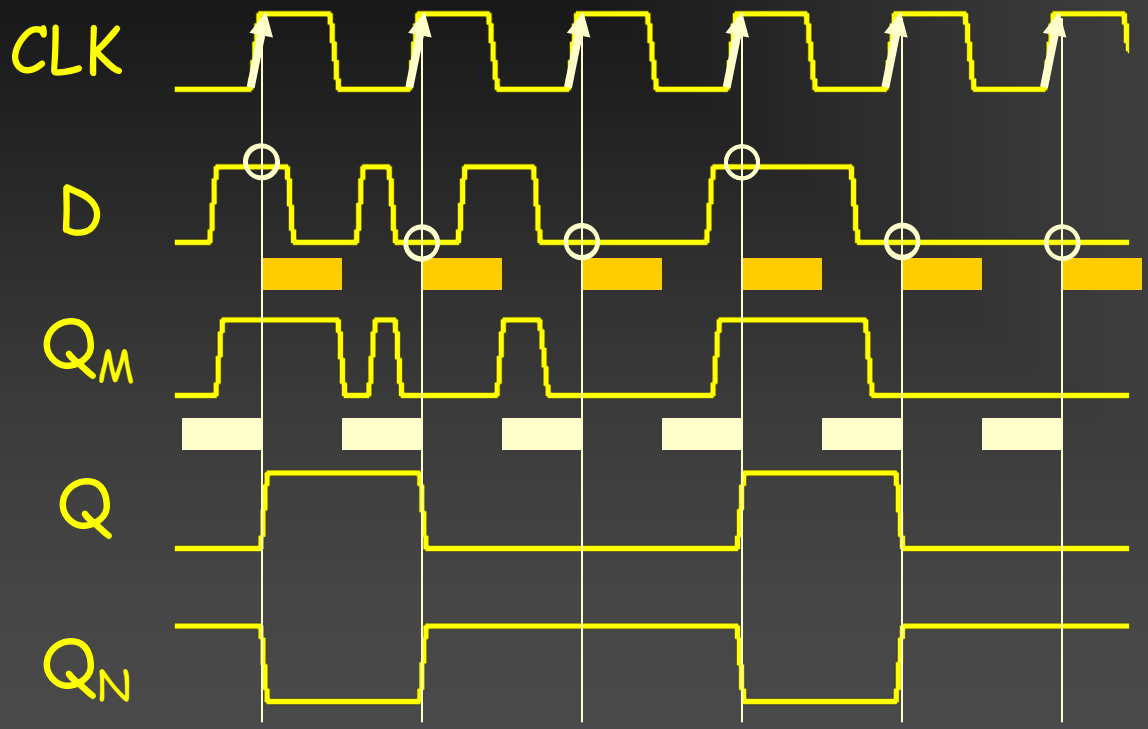
„Übernehmen“: Bei der aktiven Taktflanke (↑) wird D direkt auf Q abgebildet: Q = D

„Halten“: Bis zur nächsten aktiven Flanke hält Q seinen letzten Zustand: Q = last Q

© A. Steininger / TU Wien

D-Flip-Flop: Timing

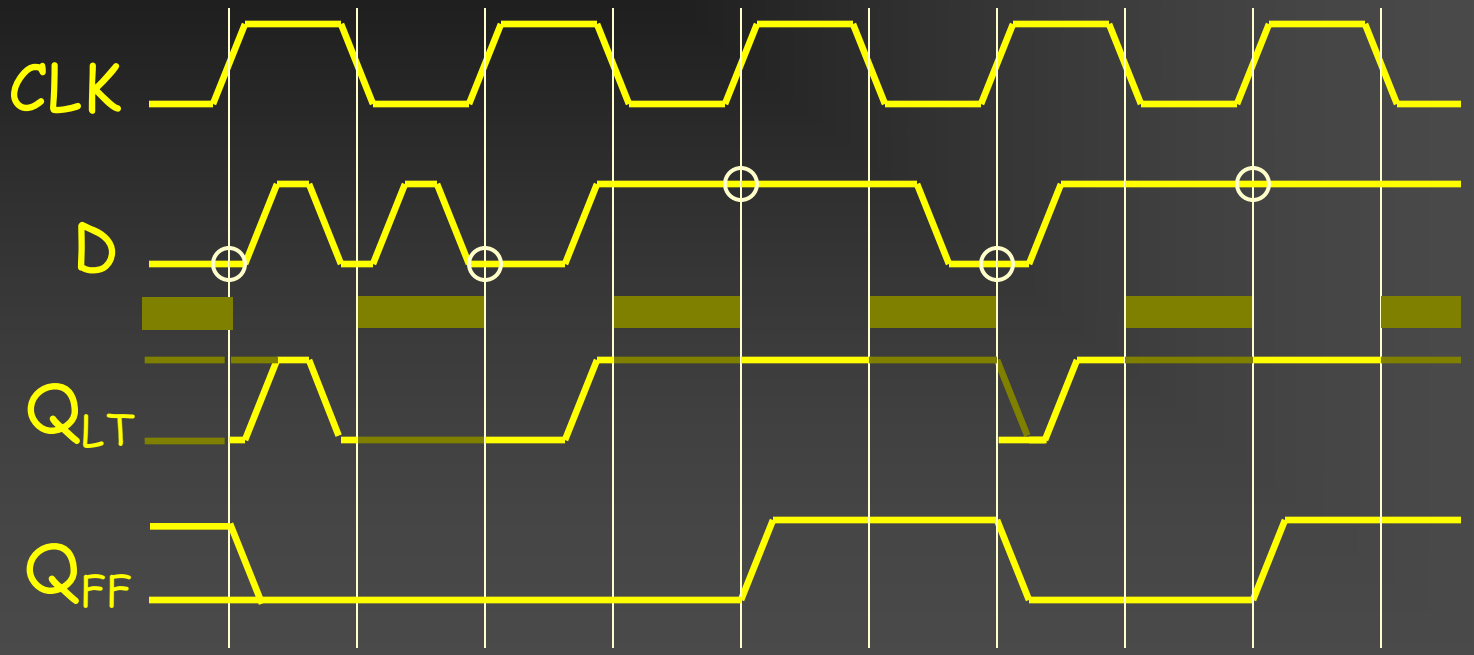




© A. Steininger / TU Wien

Latch vs. FF: ein Beispiel

Vergleiche für das gegebene Eingangssignal die Ausgangssignale von D-Latch und D-Flip-Flop

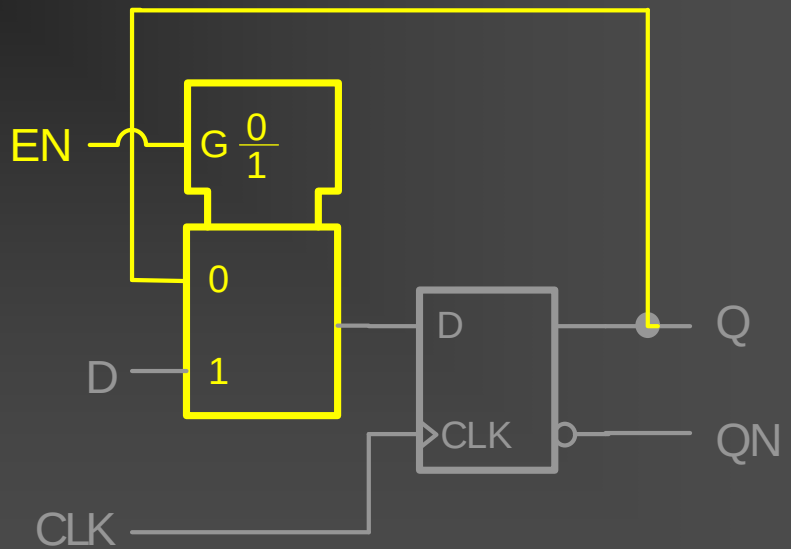


© A. Steininger / TU Wien

Clock Enable



- Oft ist die Übernahme von D nach Q nicht mit jeder aktiven Taktflanke erwünscht
- Steuerung mittels Clock-Enable (CE oder EN)
- Diese Funktion wird im Datenpfad realisiert, nicht durch Maskieren des Taktes !



© A. Steininger / TU Wien

Preset und Clear beim D-FF

Preset (Set):

erzwingt Q = 1

Clear (Reset):

erzwingt Q = 0

jeweils unabh. von D und CLK, d.h. asynchron

- für Initialisierung (Reset) sehr wichtig, sonst vermeiden.
- Realisierung durch Einfügen entsprechender Gatter im Datenpfad und in den Speicherschleifen

© A. Steininger / TU Wien

Funktion von Preset und Clear

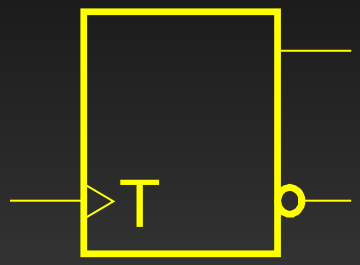
D	CLK	PRE	CLR	Q	Q_N	
0	↑	1	1	0	1	
1	↑	1	1	1	0	
x	0	1	1	last Q	last Q_N	
x	1	1	1	last Q	last Q_N	
x	x	0	1	1	0	**preset**
x	x	1	0	0	1	**clear**

Bei gleichzeitiger Aktivierung von PRE und CLR ergeben sich gleichartige Probleme wie beim SR-Latch

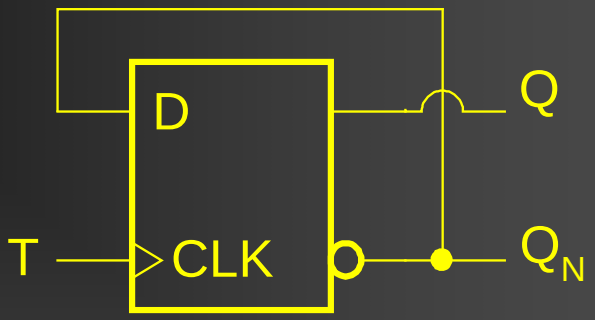
© A. Steininger / TU Wien

Toggle-Flip-Flop: Schaltung

Schaltsymbol:



Prinzipschaltung:



T … Takteingang (clock)
Q … Ausgang
Q_N … invertierter Ausgang

Als Eingang D wird invertierter Ausgang Q_N verwendet: Mit jedem Takt wird Ausgang invertiert

© A. Steininger / TU Wien

Toggle-Flip-Flop: Funktion

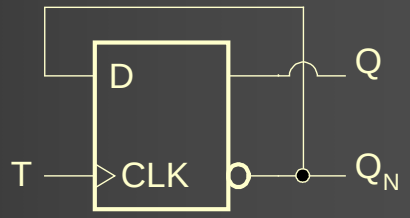
T	Q	Q_N	
↑	last Q_N	last Q	**Ausg. invertieren (toggle)**
0	last Q	last Q_N	Halten (hold)
1	last Q	last Q_N	Halten (hold)

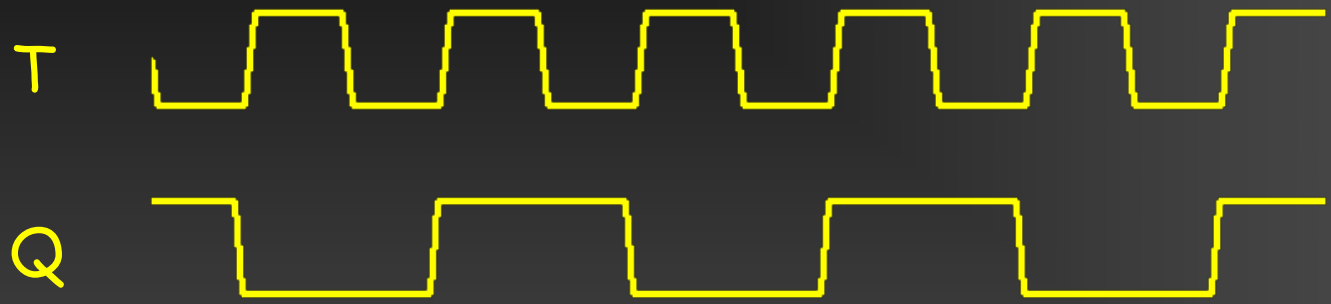
„Toggle“: Bei der aktiven Flanke (↑) von T wird der Ausgang invertiert: Q = ¬ (last Q)

„Halten“: Bis zur nächsten aktiven Flanke hält Q seinen letzten Zustand: Q = last Q

© A. Steininger / TU Wien

Toggle-Flip-Flop: Timing



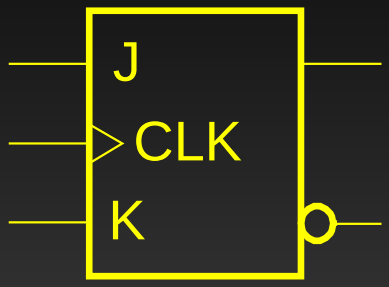


- Das T-Flip-Flop halbiert die Frequenz des Eingangssignals an T

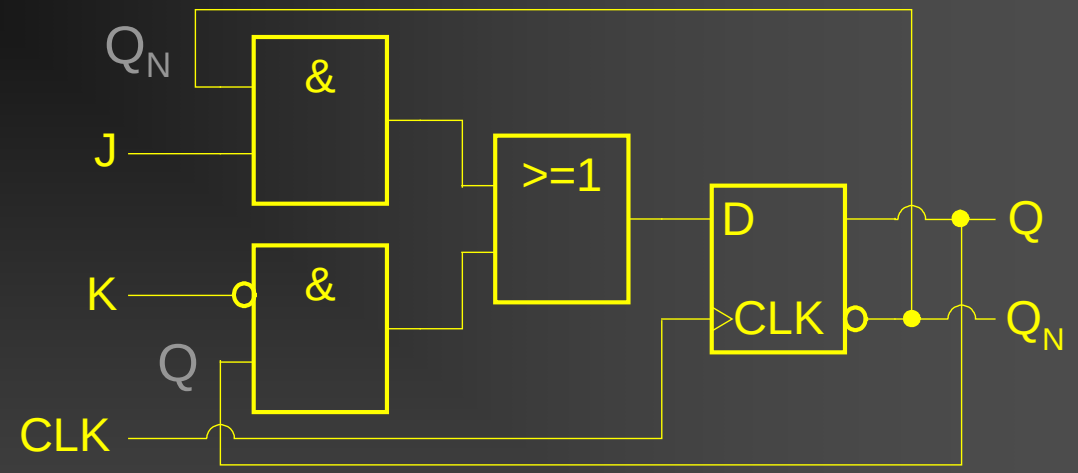
© A. Steininger / TU Wien

JK-Flip-Flop: Schaltung

Schaltsymbol:



Prinzipschaltung:



J ... Set-Eingang
K ... Clear-Eingang
CLK ... Takteingang (clock)
Q ... Ausgang
Q_N ... invertierter Ausgang

© A. Steininger / TU Wien

JK-Flip-Flop: Funktion

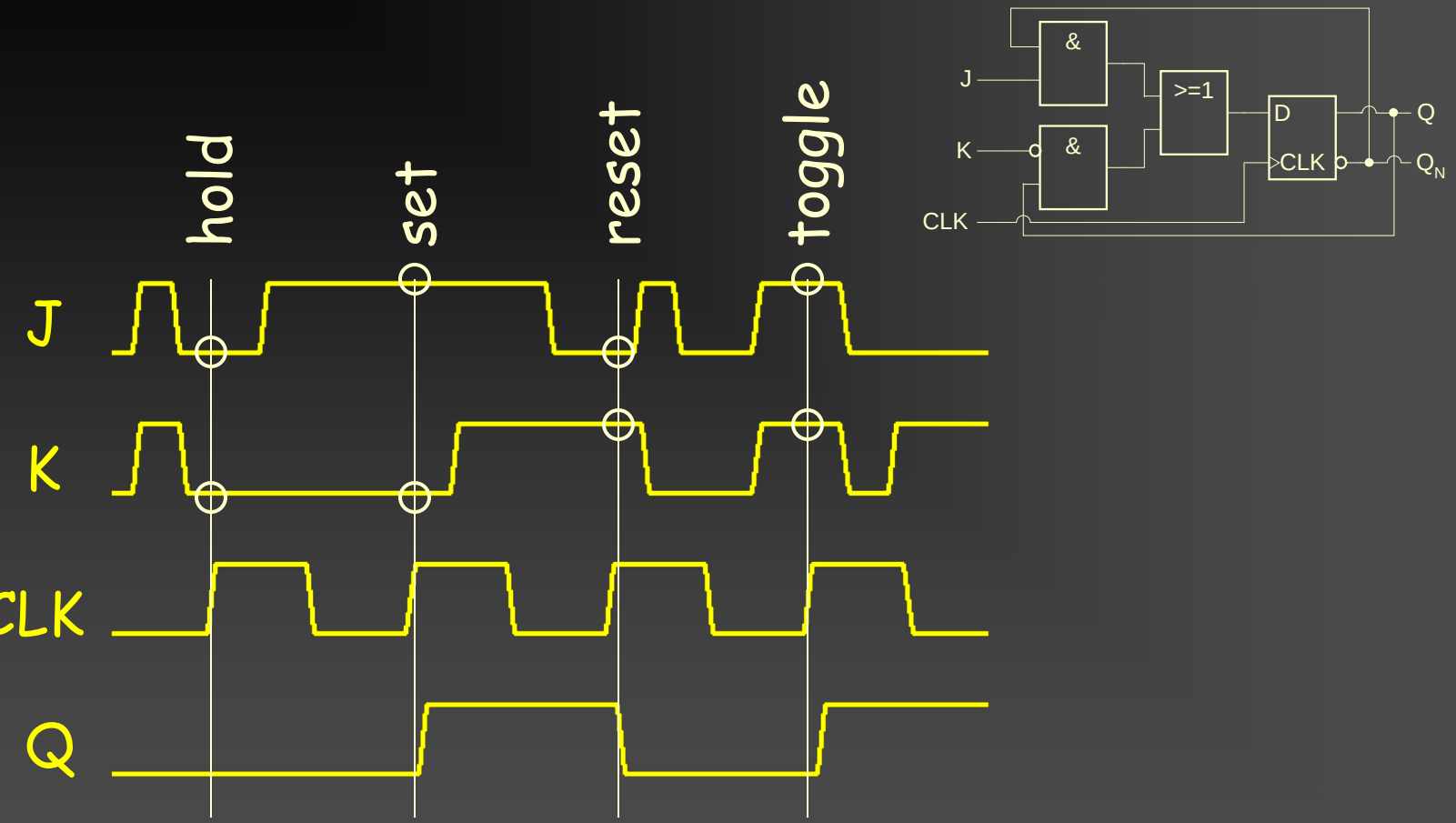
J	K	CLK	Q	Q_N	
0	0	↑	last Q	last Q_N	hold
0	1	↑	0	1	sync. clear
1	0	↑	1	0	sync. set
1	1	↑	last Q_N	last Q	toggle
x	x	0	last Q	last Q_N	hold
x	x	1	last Q	last Q_N	hold

„sync Set/clear“: Mittels J und K erfolgt ein synchrones Set bzw. Clear

„toggle“: Gleichzeitiges Set & Clear führt zum Toggeln, (als definierte Funktion, keine Instabilität)

© A. Steininger / TU Wien

JK-Flip-Flop: Timing



© A. Steininger / TU Wien

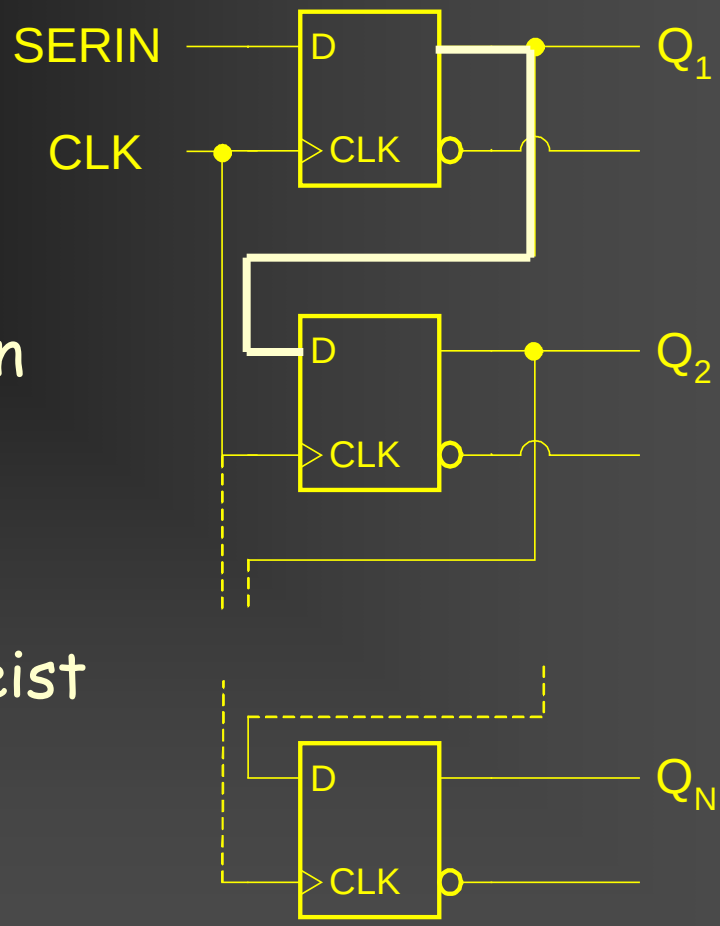
Anwendungen sequent. Logik

- Register
  - „Vektor“ von D-FFs mit gemeinsamer Steuerung
- Schieberegister, Seriell/Parallel-Konv.
- Pseudozufallsgenerator, CRC-Generator
- Finite State Machines
- ...

© A. Steininger / TU Wien

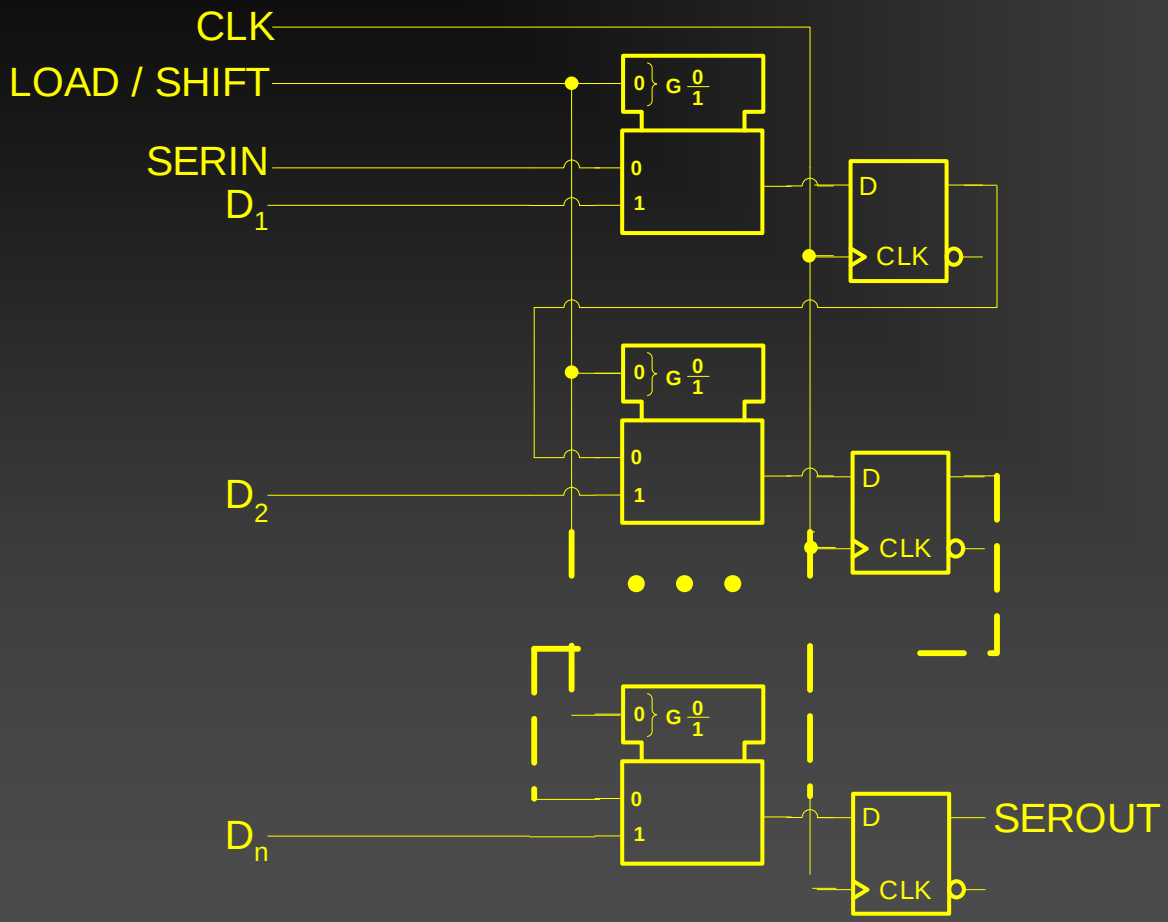
Seriell/Parallel-Konverter

- Kette von D-FFs als Schieberegister
- Mit jeder aktiven Taktflanke wird Q des vorigen FF als D vom nächsten übernommen
- Am D des 1.FF (SERIN) wird „neues Bit“ eingespeist
- Sequenz der Bits ist an $Q_1 \ldots Q_N$ sichtbar.



© A. Steininger / TU Wien

Parallel/Seriell-Konverter (1)



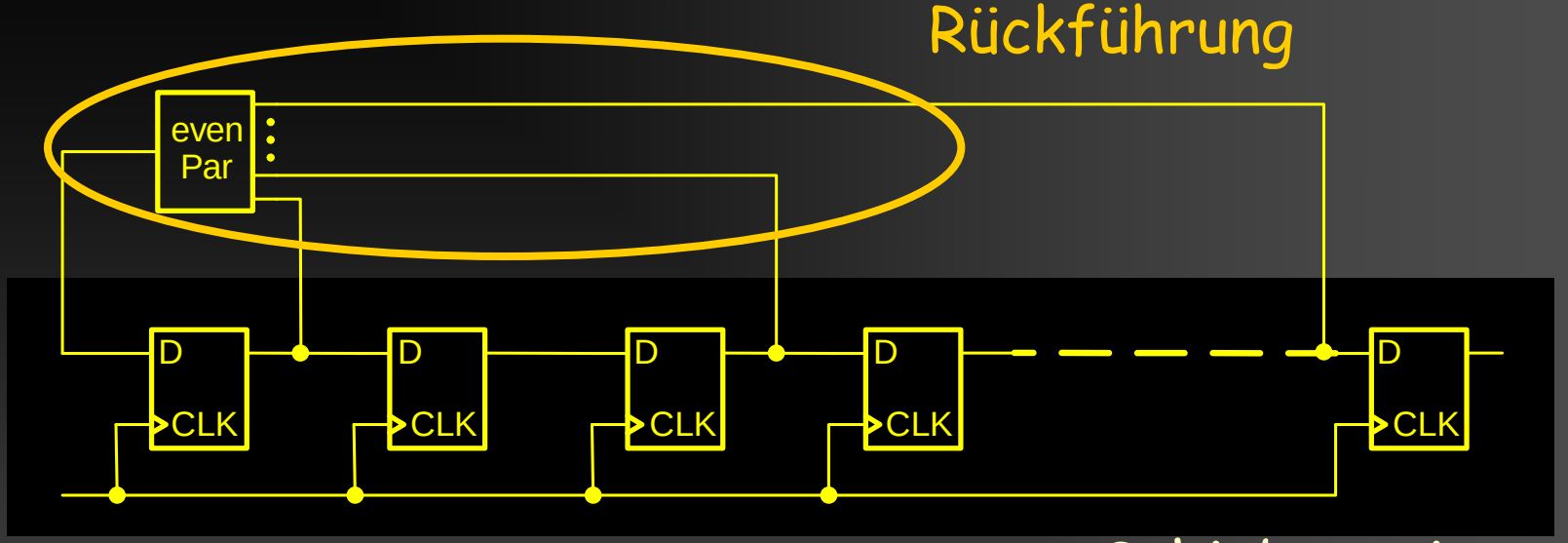
© A. Steininger / TU Wien

Parallel/Seriell-Konverter (2)

- Kette von D-FFs als Schieberegister
- Wenn LOAD aktiv ist, wird Datenwort 1D...ND synchron übernommen: LOAD schaltet MUX so dass Eingangswort an die FFs gelangt
- Schiebefunktion wie zuvor: Wenn LOAD nicht aktiv, legt MUX Q des vorigen D-FF als D an den nächsten
- Am Q des letzten FF (N) erscheinen sequentiell die Bits des geladenen Datenworts

© A. Steininger / TU Wien

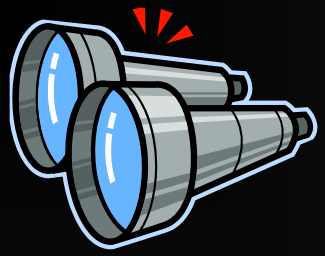
Linear Feedback Shift Register



„Rückgekoppeltes Schieberegister"

Anwendungen: Pseudozufallsgenerator,
CRC-Berechnung, ...

© A. Steininger / TU Wien



Überblick

- Einordnung des Fachgebietes
- Ziele der LVA, Überblick
- Begriffsdefinitionen
- Kombinatorische Grundfunktionen
- Sequentielle Grundfunktionen
- **Finite State Machines**
- Boolsche Algebra

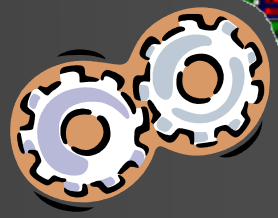
© A. Steininger / TU Wien

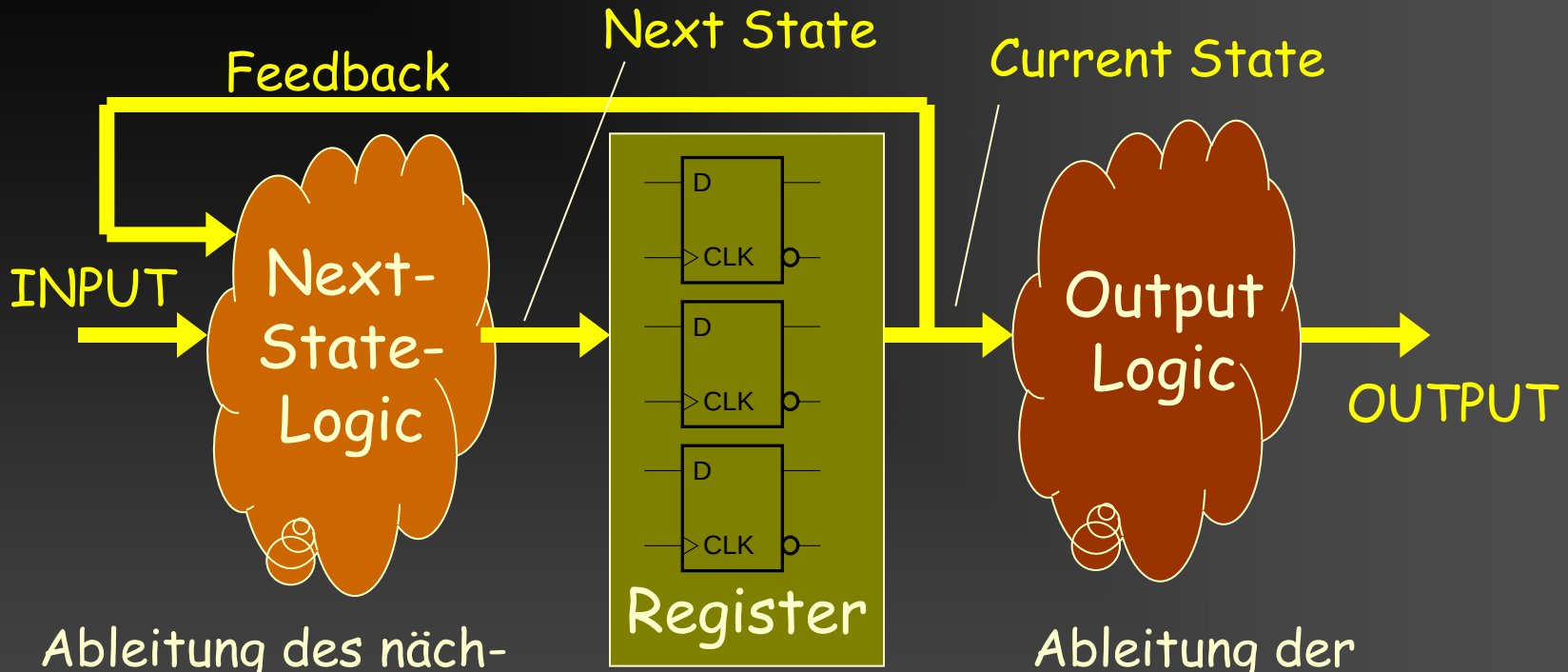
Prinzip einer State-Machine

- Die State Machine (FSM…Finite State Machine) durchläuft eine Sequenz von Zuständen.
- Zustandswechsel erfolgen ausschließlich synchron.
- Die Zustandswechsel erfolgen in Abhängigkeit vom aktuellen Zustand und von den Eingängen.
- Die Ausgangssignale werden durch den aktuellen Zustand gesteuert (Moore-State-Machine).
- Bei der Mealy-State-Machine gibt es zusätzlich eine (asynchrone) Kopplung zwischen Eingängen und Ausgängen.

© A. Steininger / TU Wien

Moore-State Machine





Ableitung des nächsten Zustands aus aktuellem Zustand und Eingängen

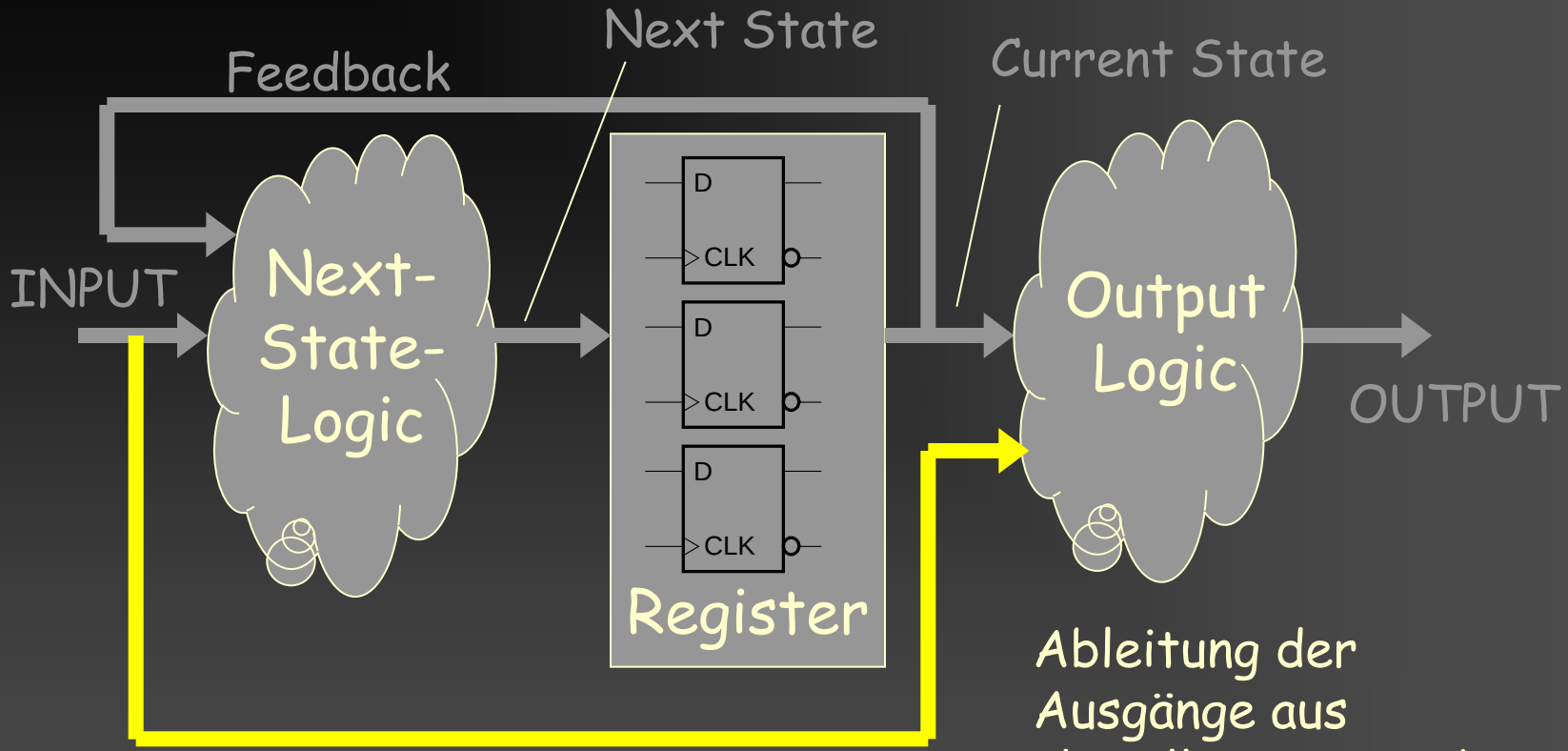
synchroner Zustandswechsel

Ableitung der Ausgänge aus aktuellem Zustand

© A. Steininger / TU Wien

Mealy-State Machine

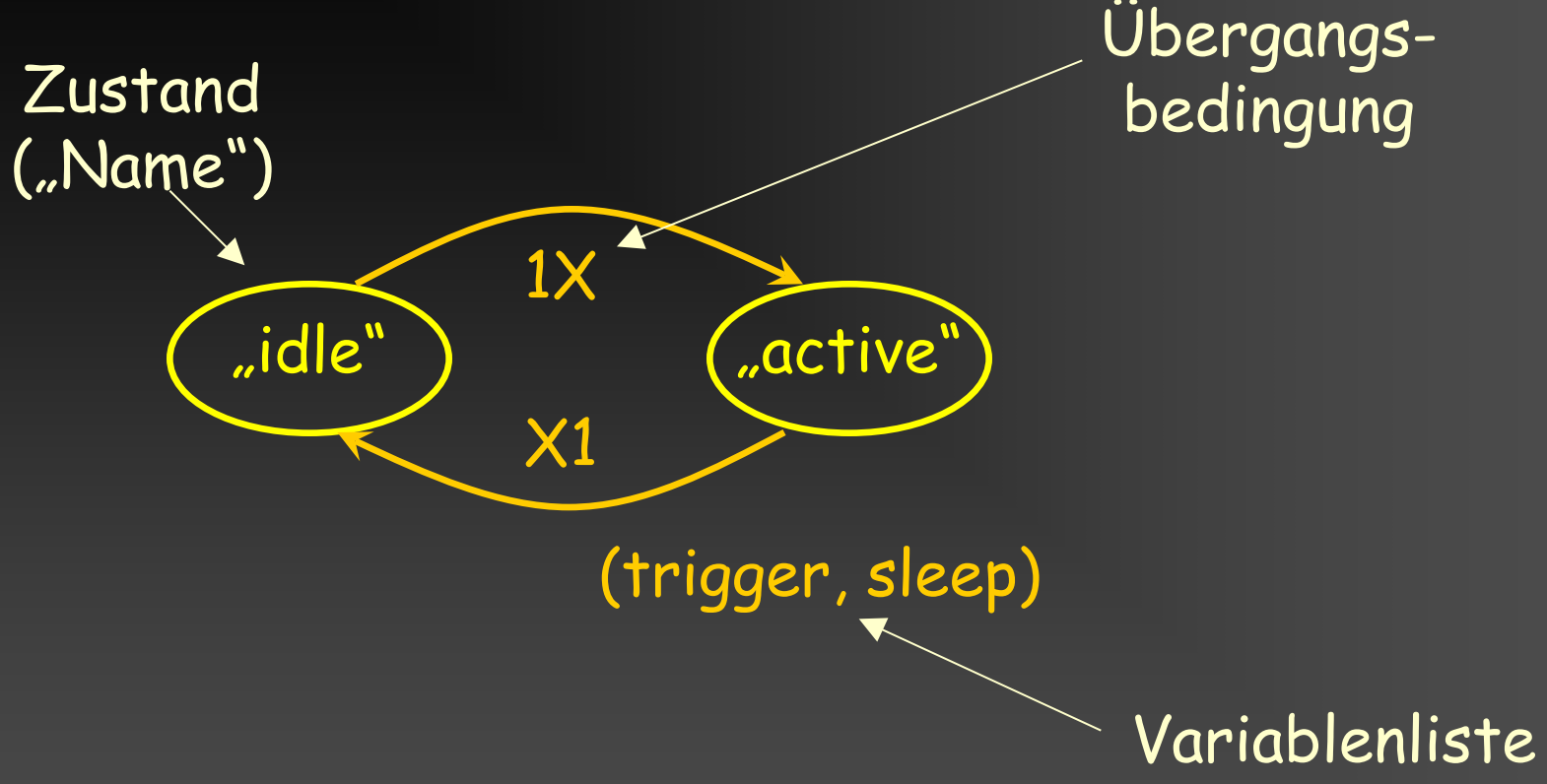




Ableitung der Ausgänge aus aktuellem Zustand und Eingängen

© A. Steininger / TU Wien

State Machine-Notation



© A. Steininger / TU Wien

State Machine: Ausgabewert

Zustand („Name") + Ausgabewert

Übergangs-bedingung + Ausgabewert

günstig bei Mealy



(trigger, sleep)

Variablenliste

Zustand	Ausgabe
idle	0
active	1

üblich bei Moore (HW-Impl.)

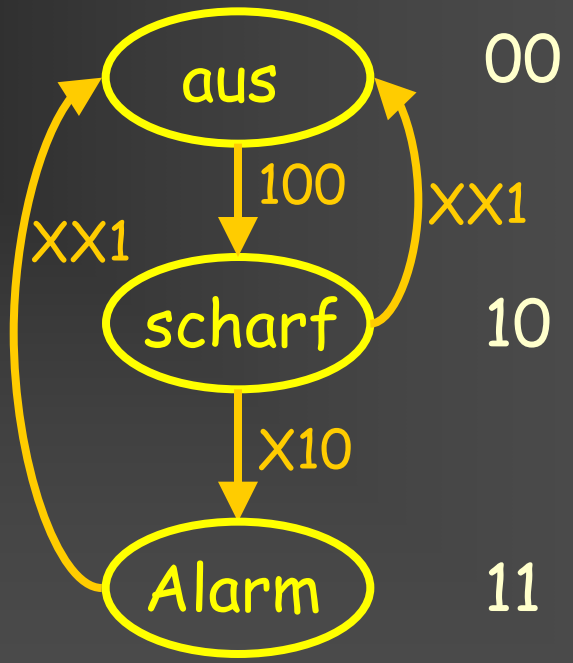
© A. Steininger / TU Wien

Beispiel: Alarmanlage

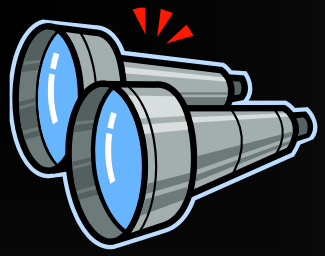
Eingänge: Aktivierungstaster, Türkontakt, Codeschalter

Ausgänge: Bereitschafts-LED, Sirene

Zust.	Ausg.	Überg. beding.	Folge-zust.
aus	00	100	scharf
scharf	10	XX1	aus
		X10	Alarm
Alarm	11	XX1	aus



© A. Steininger / TU Wien



Überblick

- Einordnung des Fachgebietes
- Ziele der LVA, Überblick
- Begriffsdefinitionen
- Kombinatorische Grundfunktionen
- Sequentielle Grundfunktionen
- Finite State Machines
- **Boolsche Algebra**

© A. Steininger / TU Wien

Boolsche Algebra: Axiome

Axiom = nicht herleitbare grundlegende Annahme

$X = 0$ wenn $X \neq 1$

wenn $X = 0$ dann $\neg X = 1$

$0 \wedge 0 = 0$

$1 \wedge 1 = 1$

$0 \wedge 1 = 1 \wedge 0 = 0$

$X = 1$ wenn $X \neq 0$

wenn $X = 1$ dann $\neg X = 0$

$1 \vee 1 = 1$

$0 \vee 0 = 0$

$1 \vee 0 = 0 \vee 1 = 1$

© A. Steininger / TU Wien

Theoreme mit 1 Variablen

Identität:

$$X \vee 0 = X \qquad X \wedge 1 = X$$

Null-Element:

$$X \vee 1 = 1 \qquad X \wedge 0 = 0$$

Idempotenz:

$$X \vee X \vee X \ldots = X \qquad X \wedge X \wedge X \ldots = X$$

Komplement:

$$X \vee (\neg X) = 1 \qquad X \wedge (\neg X) = 0$$

Involution: $\neg(\neg X) = X$

© A. Steininger / TU Wien

Theoreme mit 2 Variablen

Kommutativität:

$$X \vee Y = Y \vee X \qquad X \wedge Y = Y \wedge X$$

Überdeckung:

$$X \vee (X \wedge Y) = X \qquad X \wedge (X \vee Y) = X$$

Kombination:

$$(X \wedge Y) \vee (X \wedge \neg Y) = X \qquad (X \vee Y) \wedge (X \vee \neg Y) = X$$

© A. Steininger / TU Wien

Theoreme mit 3 Variablen

Assoziativität:

$(X \vee Y) \vee Z = X \vee (Y \vee Z)$ $\quad (X \wedge Y) \wedge Z = X \wedge (Y \wedge Z)$

Distributivität:

$$(X \wedge Y) \vee (X \wedge Z) = X \wedge (Y \vee Z) \qquad (X \vee Y) \wedge (X \vee Z) = X \vee (Y \wedge Z)$$

Konsens:

$$(X \wedge Y) \vee (\neg X \wedge Z) \vee (Y \wedge Z) = (X \wedge Y) \vee (\neg X \wedge Z) \qquad (X \vee Y) \wedge (\neg X \vee Z) \wedge (Y \vee Z) = (X \vee Y) \wedge (\neg X \vee Z)$$

© A. Steininger / TU Wien

Das Theorem von De Morgan

$$\neg F(X_1, X_2, X_3, \ldots X_n, \wedge, \vee)$$
$$=$$
$$F(\neg X_1, \neg X_2, \neg X_3, \ldots \neg X_n, \vee, \wedge)$$

Beispiel:

$$\neg\{(a \wedge b) \vee (b \wedge c)\} =$$
$$\neg(a \wedge b) \wedge \neg(b \wedge c) =$$
$$(\neg a \vee \neg b) \wedge (\neg b \vee \neg c)$$

© A. Steininger / TU Wien

Erweiterungstheorem (Shannon)

$$F(X_1,X_2,X_3,\ldots X_n)$$
$$=$$
$$X_1 \wedge [F(1,X_2,X_3,\ldots X_n)]$$
$$\vee$$
$$\neg X_1 \wedge [F(0,X_2,X_3,\ldots X_n)]$$

Beispiel:

$$(a \wedge b) \vee (b \wedge c) =$$
$$\{b \wedge [(a \wedge 1) \vee (1 \wedge c)]\} \quad \vee \quad \{\neg b \wedge [(a \wedge 0) \vee (0 \wedge c)]\} =$$
$$b \wedge (a \vee c)$$

© A. Steininger / TU Wien

Minterm

- logische UND-Verknüpfung von Variablen (negiert oder nicht negiert)
- keine Variable kommt öfter als einmal vor (durch Umformungen stets erreichbar)
- oder auch eine einzelne nicht weiter verknüpfte Variable (negiert oder nicht)
- Beispiele: $\neg Y$
  $X \wedge Y \wedge Z$
  $X \wedge \neg Y \wedge Z$

© A. Steininger / TU Wien

Maxterm

- logische ODER-Verknüpfung von Variablen (negiert oder nicht negiert),
- keine Variable kommt öfter als einmal vor (durch Umformungen stets erreichbar)
- oder auch eine einzelne nicht weiter verknüpfte Variable (negiert oder nicht)
- Beispiele: $\neg Y$
  $X \vee Y \vee Z$
  $X \vee \neg Y \vee Z$

© A. Steininger / TU Wien

Zeilen der Wahrheitstabelle

A	B	C	F	Minterm	Maxterm
0	0	0	0	$\neg A \wedge \neg B \wedge \neg C$	$A \vee B \vee C$
0	0	1	0	$\neg A \wedge \neg B \wedge C$	$A \vee B \vee \neg C$
0	1	0	1	$\neg A \wedge B \wedge \neg C$	$A \vee \neg B \vee C$
0	1	1	1	$\neg A \wedge B \wedge C$	$A \vee \neg B \vee \neg C$
1	0	0	0	$A \wedge \neg B \wedge \neg C$	$\neg A \vee B \vee C$
1	0	1	1	$A \wedge \neg B \wedge C$	$\neg A \vee B \vee \neg C$
1	1	0	0	$A \wedge B \wedge \neg C$	$\neg A \vee \neg B \vee C$
1	1	1	1	$A \wedge B \wedge C$	$\neg A \vee \neg B \vee \neg C$

© A. Steininger / TU Wien

Disjunktive Normalform

Ableitung einer algebraischen Darstellung einer beliebigen logischen Funktion aus deren Wahrheitstabelle:

Alle Minterme, für die F = 1 werden ODER-verknüpft

Beispiel von voriger Tabelle:

$$F = (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge C)$$

© A. Steininger / TU Wien

Konjunktive Normalform

Ableitung einer algebraischen Darstellung einer beliebigen logischen Funktion aus deren Wahrheitstabelle:

Alle Maxterme, für die F = 0 werden UND-verknüpft

Beispiel von voriger Tabelle:

$$F = (A \vee B \vee C) \wedge (A \vee B \vee \neg C) \wedge (\neg A \vee B \vee C) \wedge (\neg A \vee \neg B \vee C)$$

© A. Steininger / TU Wien

Karnaugh-Veitch-Diagramm

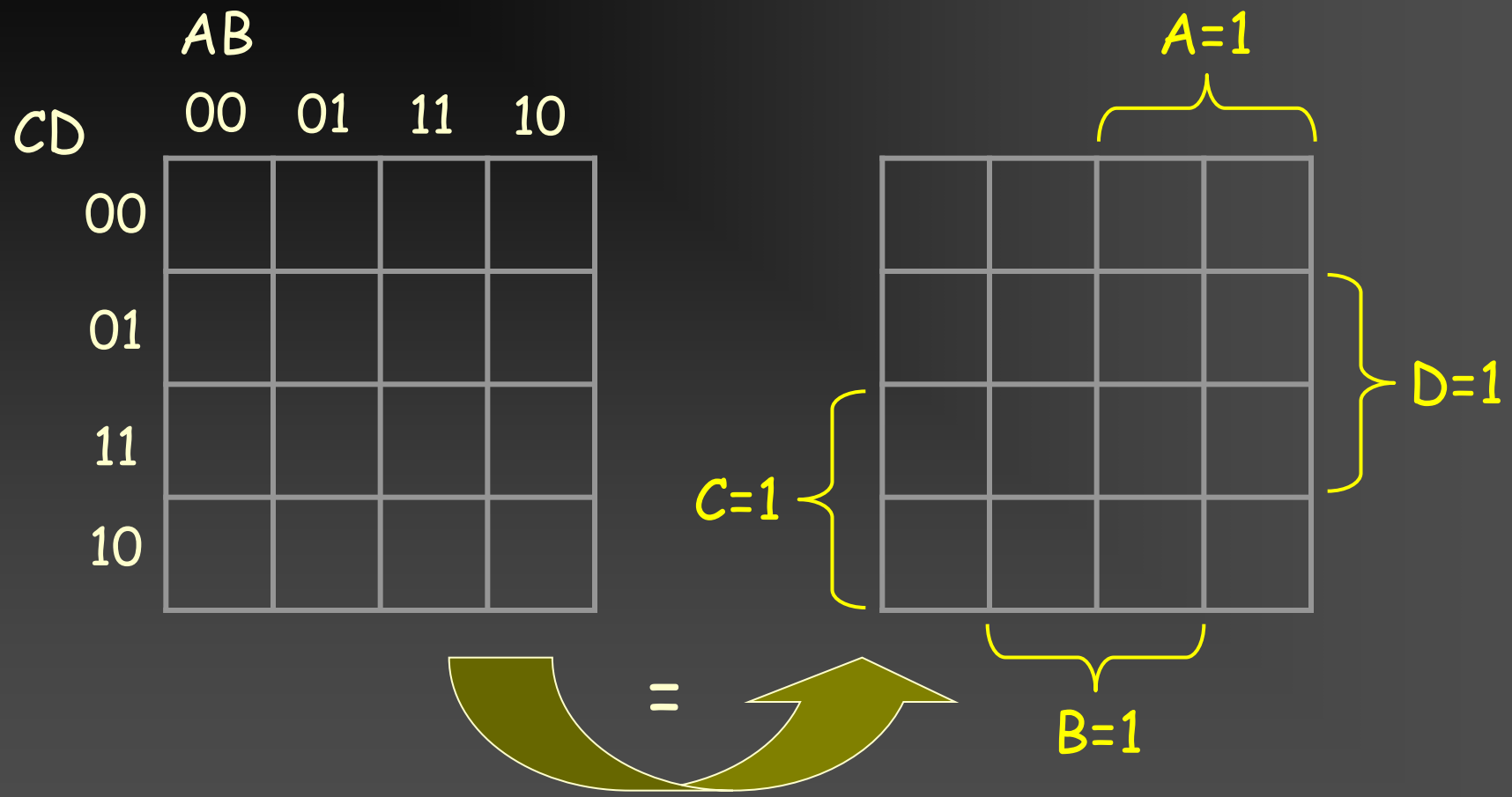
„KV-Diagramm"

- andere Darstellung der Wahrheitstabelle
- Zwei benachbarte Positionen unterscheiden sich genau um 1 Bit
- Grundlage für Optimierungen

F	AB 00	01	11	10
CD 00	0	0	1	1
01	0	0	1	1
11	1	1	0	1
10	0	0	1	1

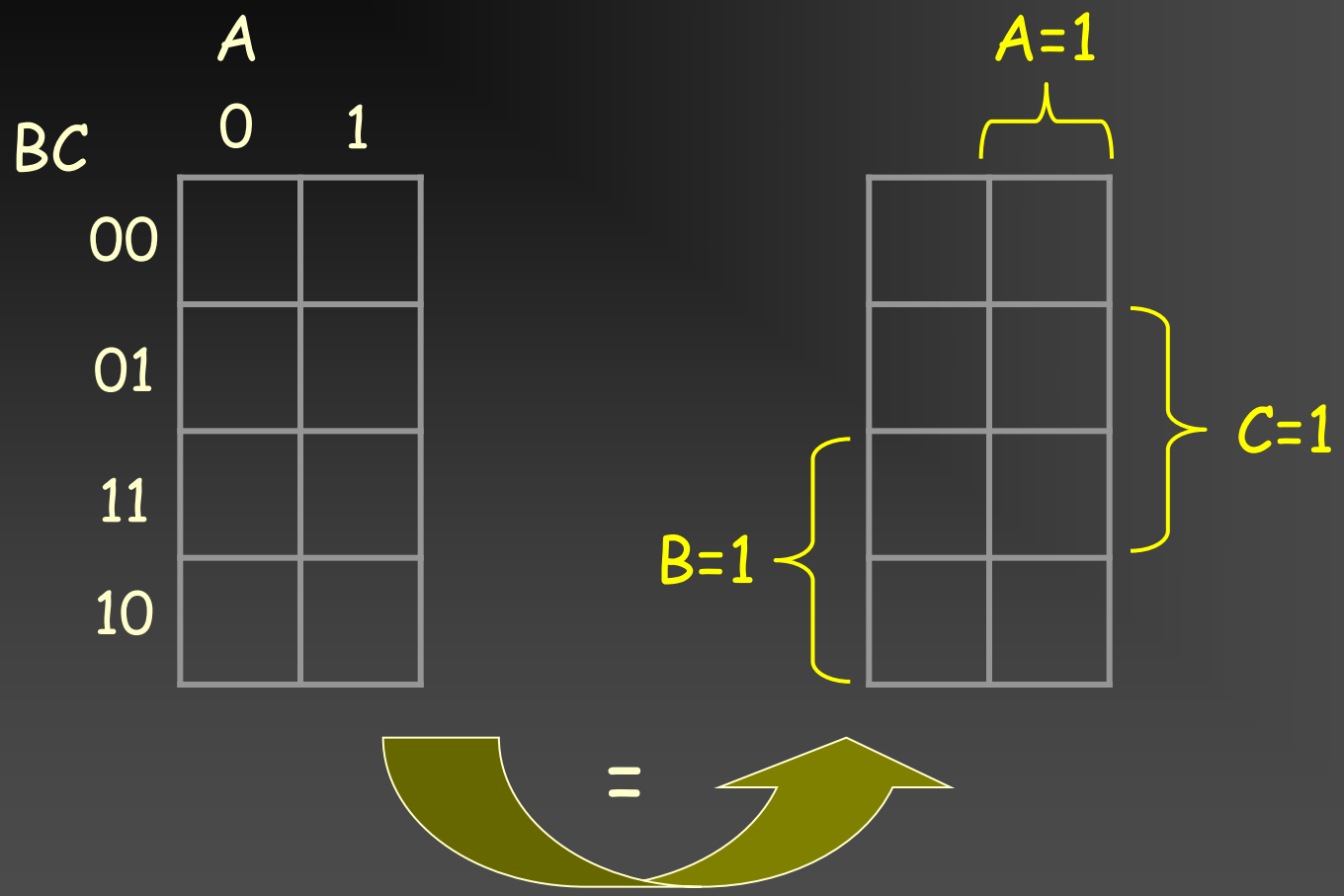
© A. Steininger / TU Wien

KV-Diagramm: Darstellungen



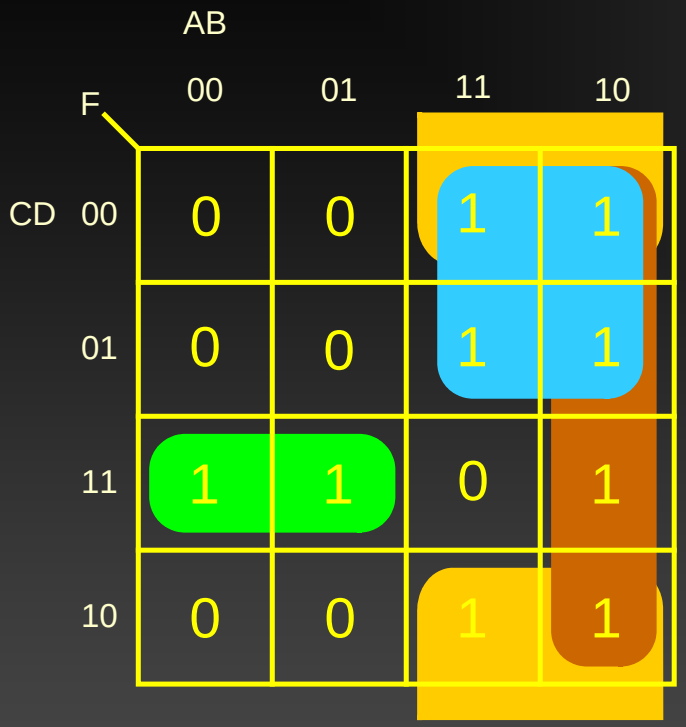
© A. Steininger / TU Wien

KV-Diagramm für 3 Variable



© A. Steininger / TU Wien

KV-Diagramm: optimale DNF

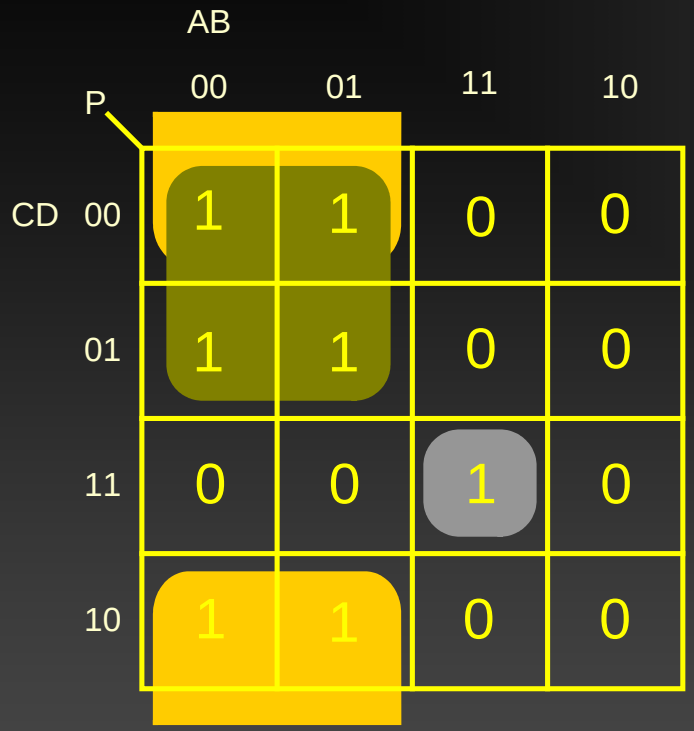


- Überdecken aller 1-Fenster mit möglichst wenigen möglichst großen Rechtecken
- Mehrfachüberdeckung ist erlaubt
- Jedes Rechteck beschreibt einen Minterm

$$F = (A \wedge \neg B) \vee (A \wedge \neg C) \vee (A \wedge \neg D) \vee (\neg A \wedge C \wedge D)$$

© A. Steininger / TU Wien

KV-Diagramm: Ableitung KNF



- Vertauschen aller 1 mit 0 (Inversion)
- Ermittlung der optimalen DNF wie zuvor
- Umwandlung in KNF mittles Theoremen (DeMorgan)

$$\neg F = (A \wedge B \wedge C \wedge D) \vee (\neg A \wedge \neg D) \vee (\neg A \wedge \neg C)$$

$$F = (\neg A \vee \neg B \vee \neg C \vee \neg D) \wedge (A \vee D) \wedge (A \vee C)$$

© A. Steininger / TU Wien



Zusammenfassung (1)

- Der Einsatz digitaler Elektronik in sogenannten „Embedded Systems" prägt unsere Gesellschaft.
- Digitale Elektronik bietet gegenüber der analogen Vorteile wie einfache Speicherbarkeit, bessere Störsicherheit und höheren Abstraktionsgrad.
- Die Kunst des digitalen Design besteht darin, die Grenzen digitaler Logik zu verstehen, die immanenten Idealisierungen zu bedenken und durch Kenntnis des gesamten Entwicklungsprozesses die Gesamtkosten eines Design zu minimieren.

© A. Steininger / TU Wien



Zusammenfassung (2)

- Kombinatorische Logik funktioniert unabhängig von der Vorgeschichte. Sequentielle Logik hat im Gegensatz dazu ein Gedächtnis (Rückkopplung oder Speicher) und daher einen inneren Zustand.
- Kombinatorische Grundfunktionen sind Inverter, AND, OR, NAND, NOR, XOR und XNOR.
- Aus diesen Funktionen lassen sich komplexere Funktionen mit mehr Eingängen wie z.B. ein Multiplexer oder ein Threshold Gate realisieren.

© A. Steininger / TU Wien



Zusammenfassung (3)

- Sequentielle Grundfunktionen sind das Latch und das Flip-Flop. Es gibt jeweils mehrere Varianten.
- Beim D-Latch steuert ein „Enable"-Signal, ob die aktuellen Daten durchgelassen oder die alten Daten gehalten werden.
- Das D-Flip-Flop übernimmt mit der aktiven Taktflanke neue Daten und hält diese bis zur nächsten.
- Zusatzfunktionen beim Flip-Flop sind „Preset" und „Clear" sowie „Clock Enable".

© A. Steininger / TU Wien



Zusammenfassung (4)

- Eine State-Machine durchläuft synchron eine Sequenz von Zuständen.
- Die Abfolge der Zustände wird durch die Eingangssignale gesteuert.
- Bei der Moore-State Machine hängen die Ausgänge nur vom Zustand ab, bei der Mealy-State Machine werden sie zusätzlich auch (asynchron) von den Eingängen beeinflusst.
- Die Darstellung einer State-Machine erfolgt im Zustandsgraph und im Zustandsdiagramm.

© A. Steininger / TU Wien



Zusammenfassung (5)

- Die Boolsche Algebra umfasst mehrere Theoreme, wie z.B. jene von De Morgan und Shannon.
- Jede Boolsche Funktion ist durch ihre Wahrheitstabelle eindeutig beschrieben.
- Eine grafische Darstellung der Wahrheitstabelle ist im Karnaugh-Veitch-Diagramm möglich.
- Aus Wahrheitstabelle bzw. KV-Diagramm lassen sich Minterme bzw. Maxterme ableiten und durch deren Verknüpfung schließlich die disjunktive bzw. die konjunktive Normalform.

© A. Steininger / TU Wien

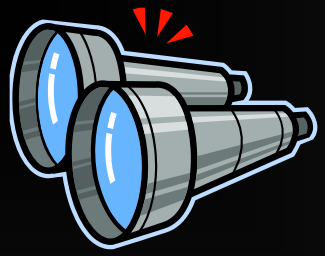
ASIC-Fertigung





Vom Sand zum
hochkomplexen Chip

© A. Steininger / TU Wien



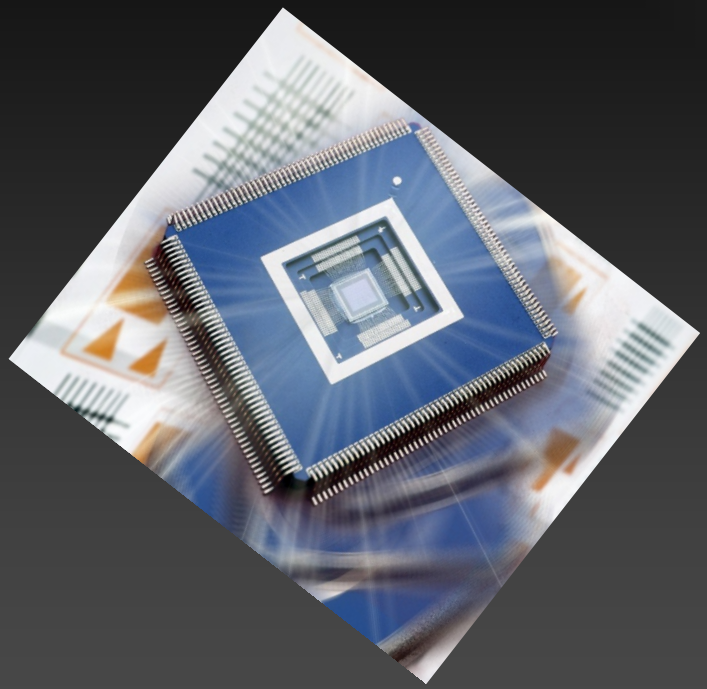
Überblick

- Motivation für ICs
- Entwicklungstrends in der IC-Technologie
- Aufbau & Fertigung eines Chips
- Moderne Formen von Chips (MCM, SOC)
- Grenzen der Technologie

© A. Steininger / TU Wien

Warum „Integrated Circuit“?

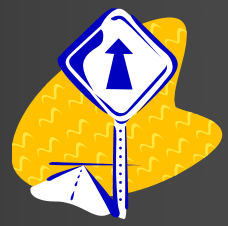
Schaltung wird



- kleiner
- schneller
- leistungsfähiger
- stromsparender
- störsicherer
- billiger
- schwerer kopierbar ...

© A. Steininger / TU Wien

Technology Roadmap



- „International Technology Roadmap for Semiconductors (ITRS)“ www.itrs.net
- Herausgeber:

SEMATECH
Accelerating the next technology revolution.

- Konsortium der weltweit wichtigsten Halbleiterhersteller
- Projektion der Technologie-entwicklung für die nächsten 15 Jahre

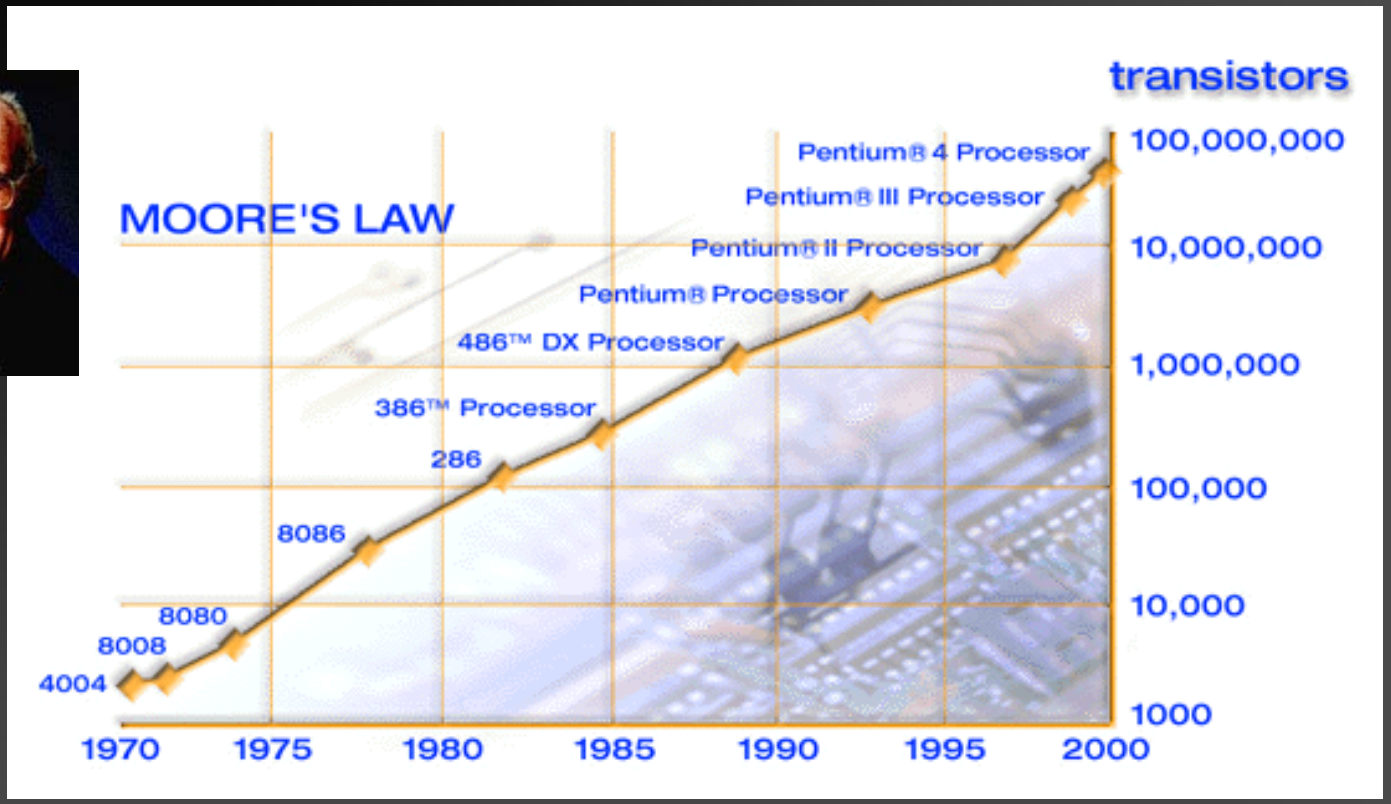
SEMATECH consortia members make up 50 percent of the worldwide chip market

AMD, HEWLETT PACKARD, IBM, Infineon, intel, Micron, NEC, NXP, Panasonic, qimonda, RENESAS, SAMSUNG, SPANSION, tsmc, TEXAS INSTRUMENTS

© A. Steininger / TU Wien

Das Moore'sche Gesetz

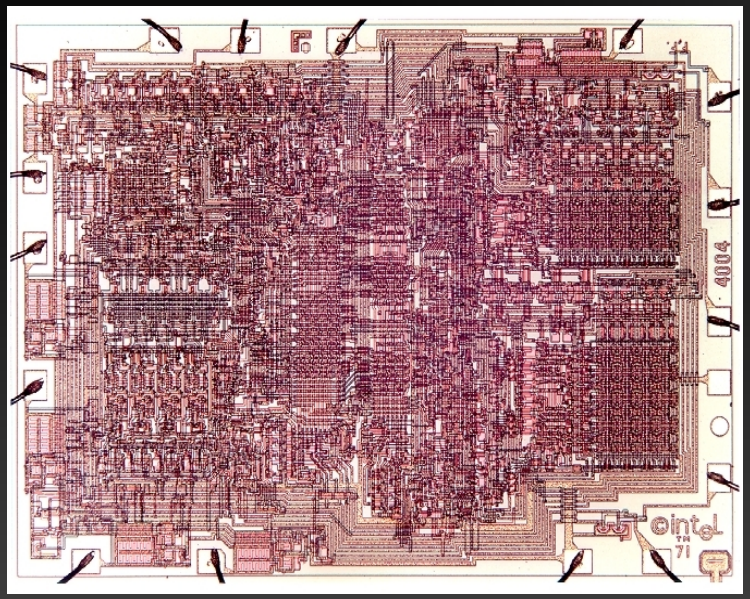


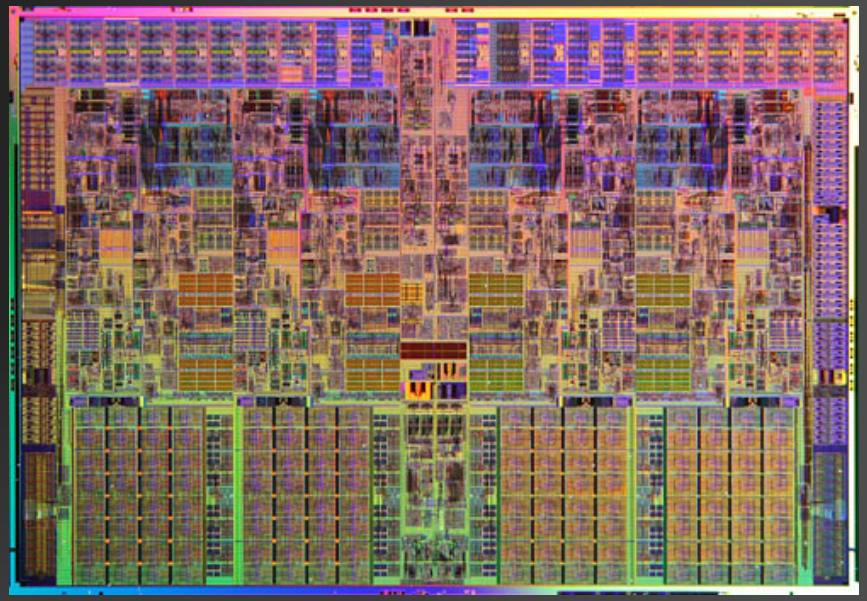


„Die Komplexität verdoppelt sich alle 1,5 Jahre"

© A. Steininger / TU Wien

Komplexitätszuwachs





Intel 4004 (1971)

- 2300 Transistoren
- 12 mm² / 10µm
- 108 kHz

Intel i7 (2009)

- 731 Millionen Transistoren
- 263 mm² / 0,045µm
- 3,3 GHz

© A. Steininger / TU Wien

Komplexitätsmaße

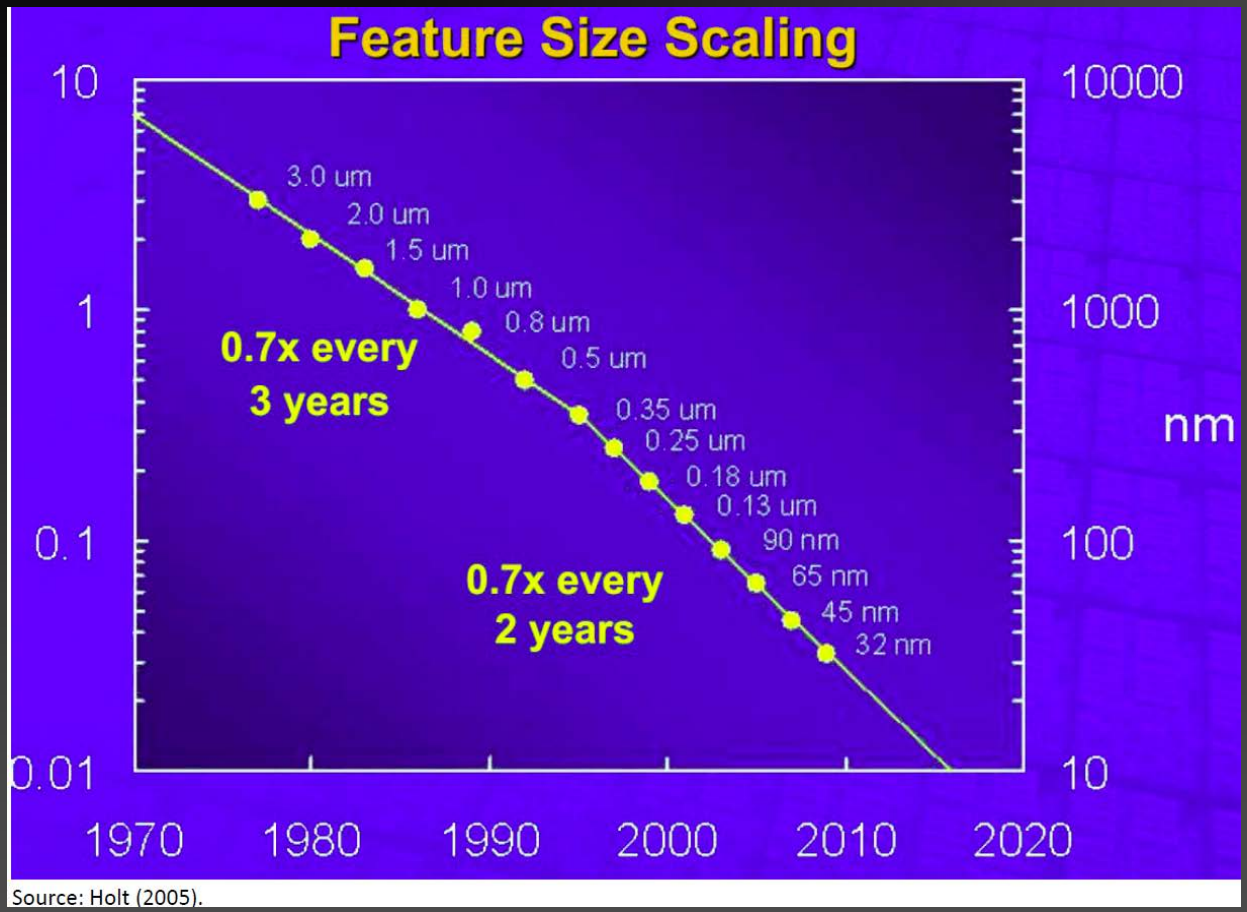


- „Transistoren“
  1970: ca. 10...100
  2017: ca. 20 Milliarden
- „Gate Count“
  Anzahl der äquivalenten 2-Input NAND-Gatter
- Prozeßtechnologie („x nm“)
  Länge des kleinsten Transistors
  „Feature Size“ λ ist die Hälfte davon

© A. Steininger / TU Wien

Kleiner...





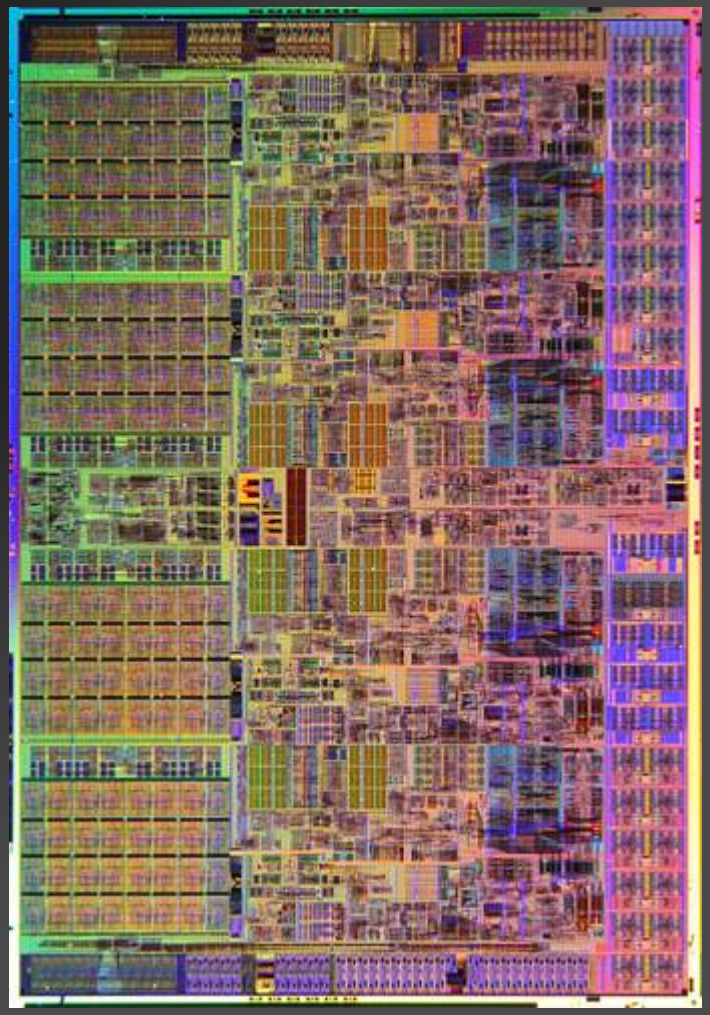
Source: Holt (2005).

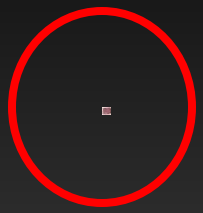
quadratischer Einfluss auf Fläche:
0.5x /3 Jahre

© A. Steininger / TU Wien

Miniaturisierung: ein Beispiel

Core i7





Verhältnis Fläche

1: 1.000.000

4004

Verhältnis #Trans

1: 317826

© A. Steininger / TU Wien

Ein anderes Beispiel…



"Mailüfterl" 1958

8000 "Transistoren"

20km Schaltdraht

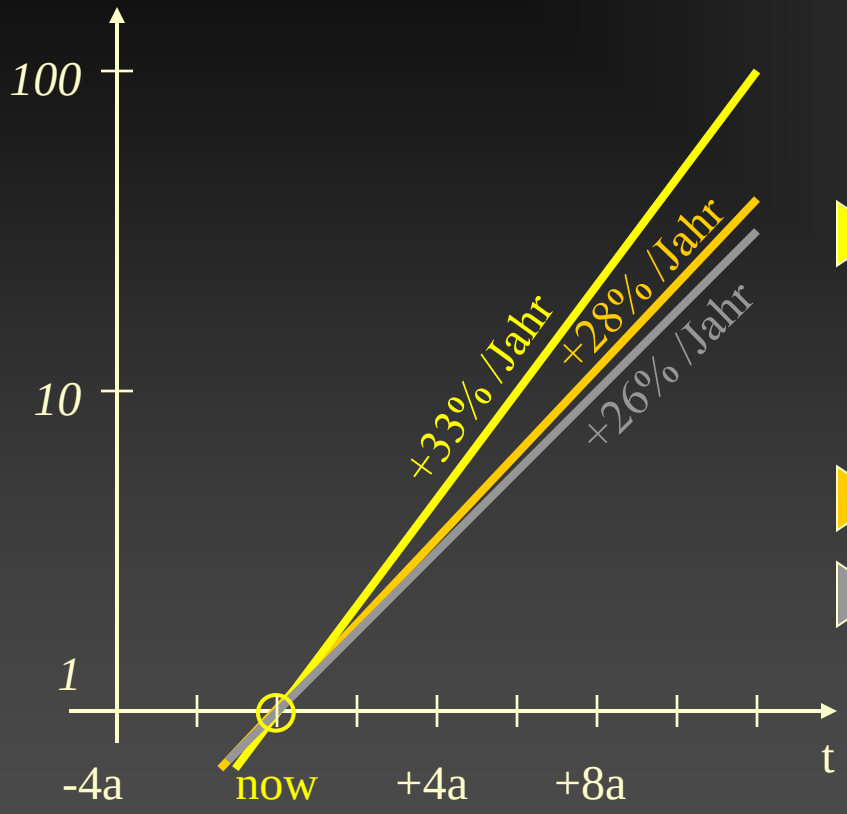
[By Florian Staudacher CC-BY-3.0 via Wikimedia Commons.]

© A. Steininger / TU Wien

Leistungsfähiger...



SEMATECH
Accelerating the next technology revolution.



Speicherdichte

- DRAM: ≈10Gbit/cm²

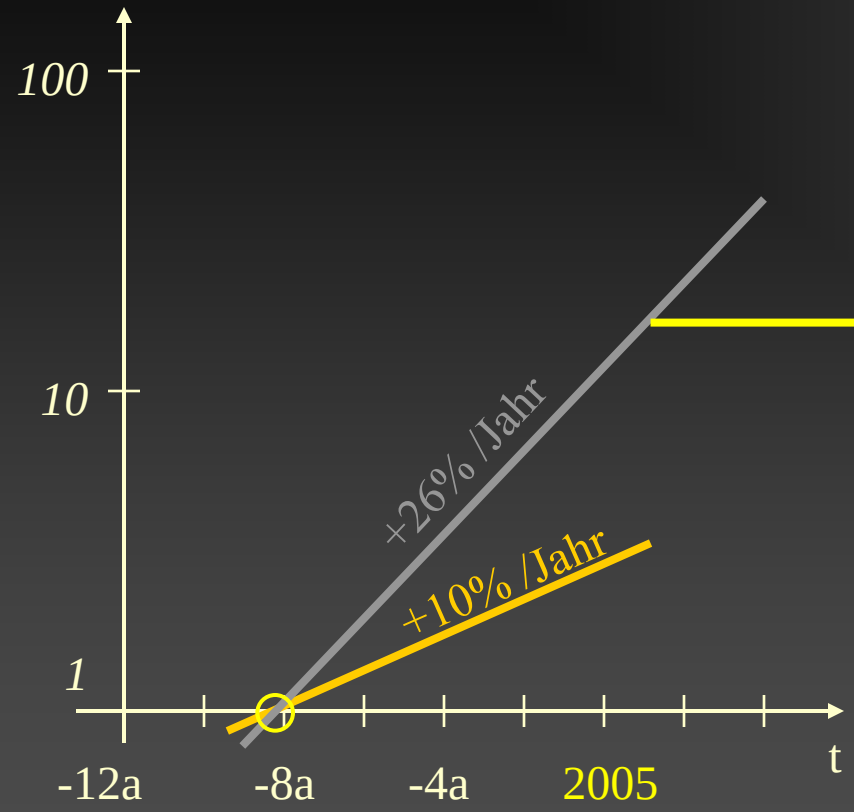
Logikdichte

- SRAM: ≈1GTrans/cm²
- Logik: ≈0,1GTrans/cm²

© A. Steininger / TU Wien

Schneller...



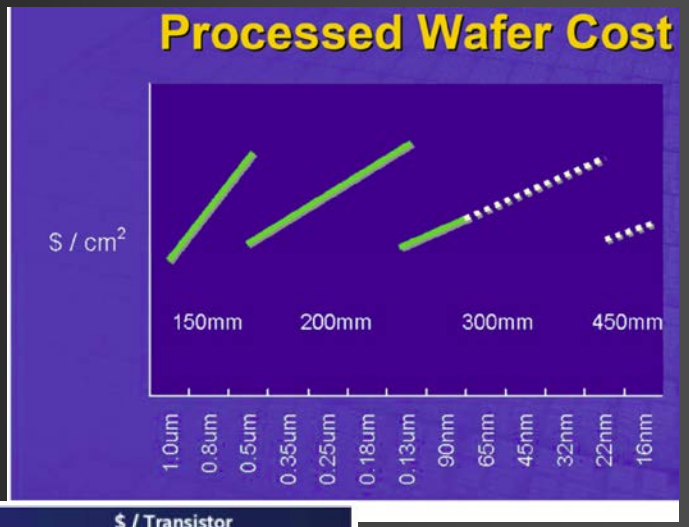


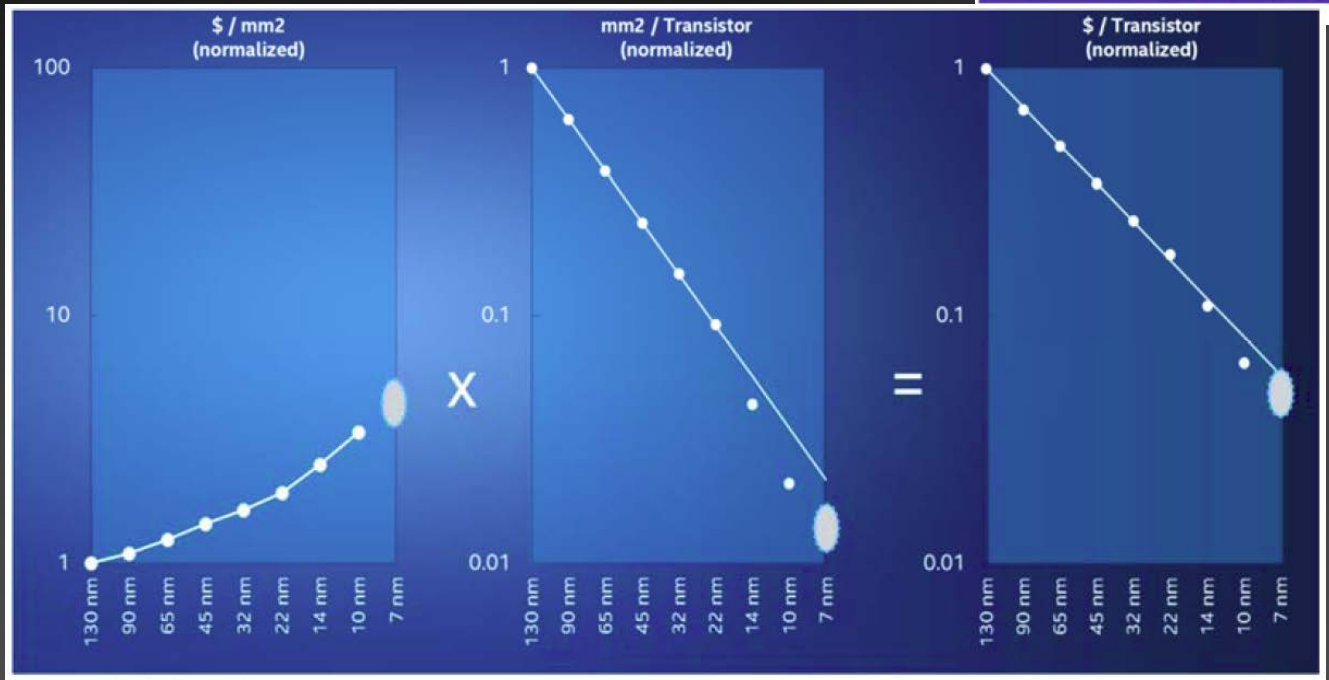
- Taktfrequenz
  ≈5GHz
- DRAM-Zugriff

© A. Steininger / TU Wien

Billiger…

Kosten pro Transistorfertigung fallen mit ca 30% pro Jahr



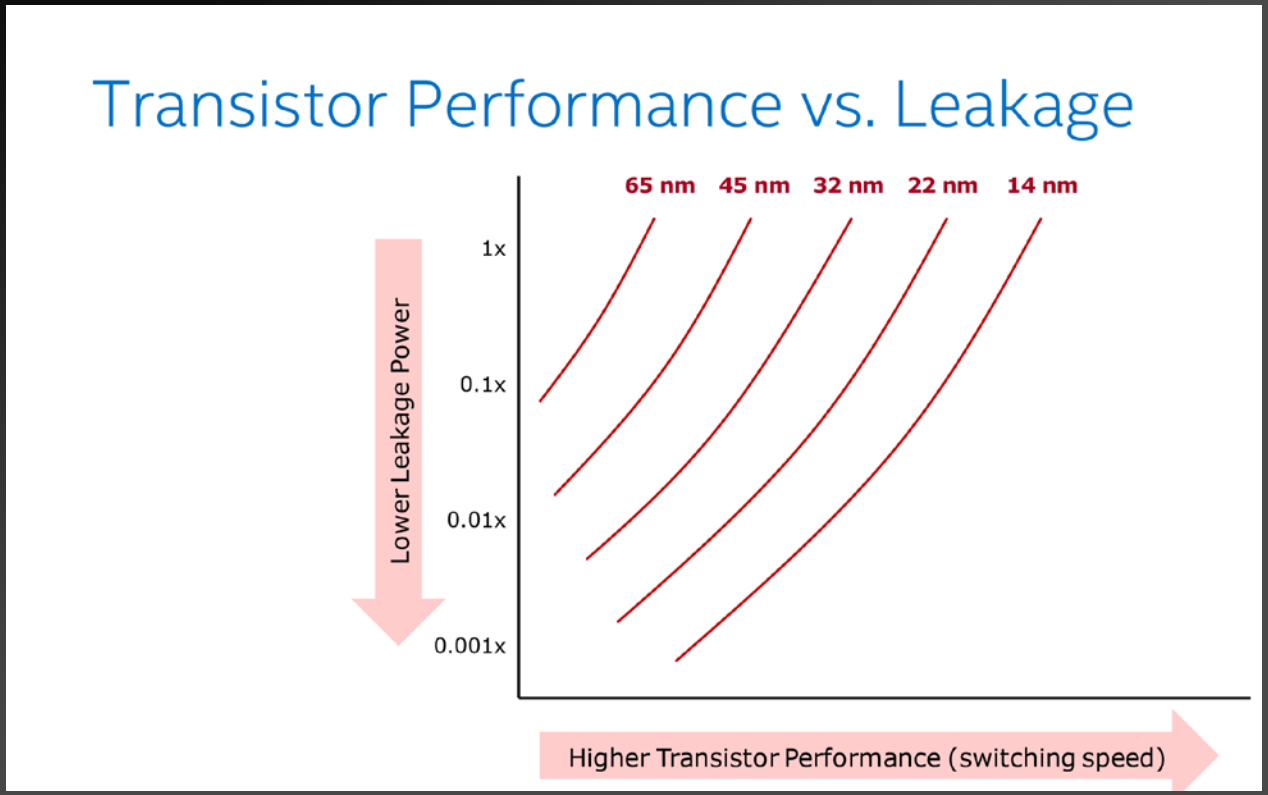


Source: Holt (2015).

Problem: Testkosten!

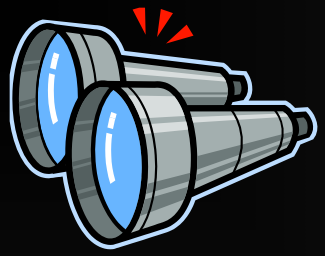
© A. Steininger / TU Wien

Energieeffizienter…



source: Intel

© A. Steininger / TU Wien



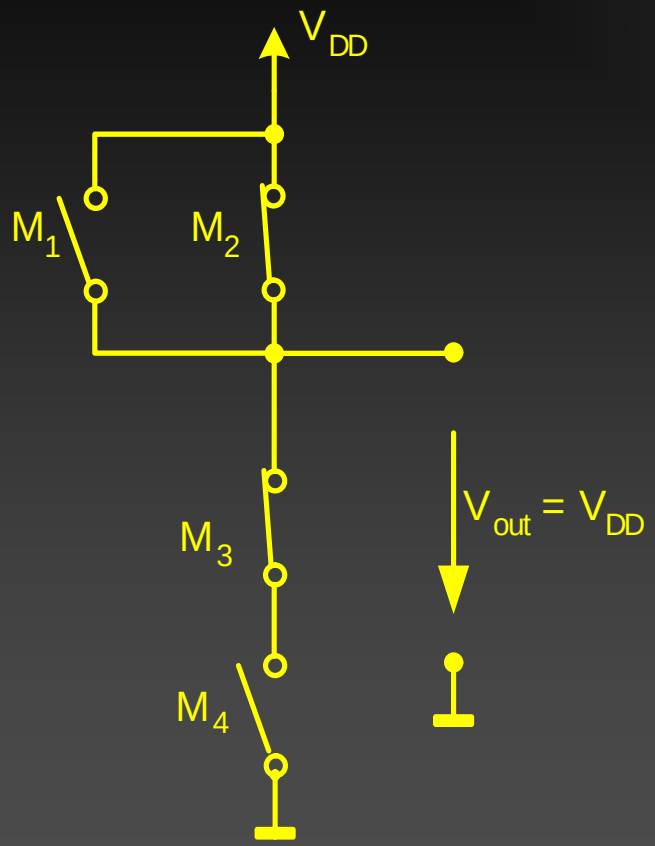
Überblick

- Motivation für ICs
- Entwicklungstrends in der IC-Technologie
- **Aufbau & Fertigung eines Chips**
- Moderne Formen von Chips (MCM, SOC)
- Grenzen der Technologie

© A. Steininger / TU Wien

Aufbau digitaler Logik

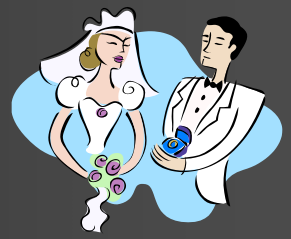




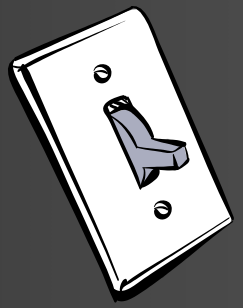
- feste Verbindungen (Verdrahtung)
- Schaltbare Verbindungen
- Isolation

© A. Steininger / TU Wien

Bestandteile eines Chips



- Feste Verbindungen:
  - Polykristallines Silizium („Poly-Si")
  - Aluminium bzw. Kupfer



- Schaltbare Verbindungen
  - Dotiertes Silizium



- Isolation
  - Silizium-Dioxid (SiO_2)

© A. Steininger / TU Wien

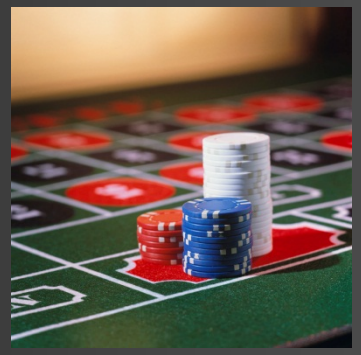
Rohmaterial für einen Chip

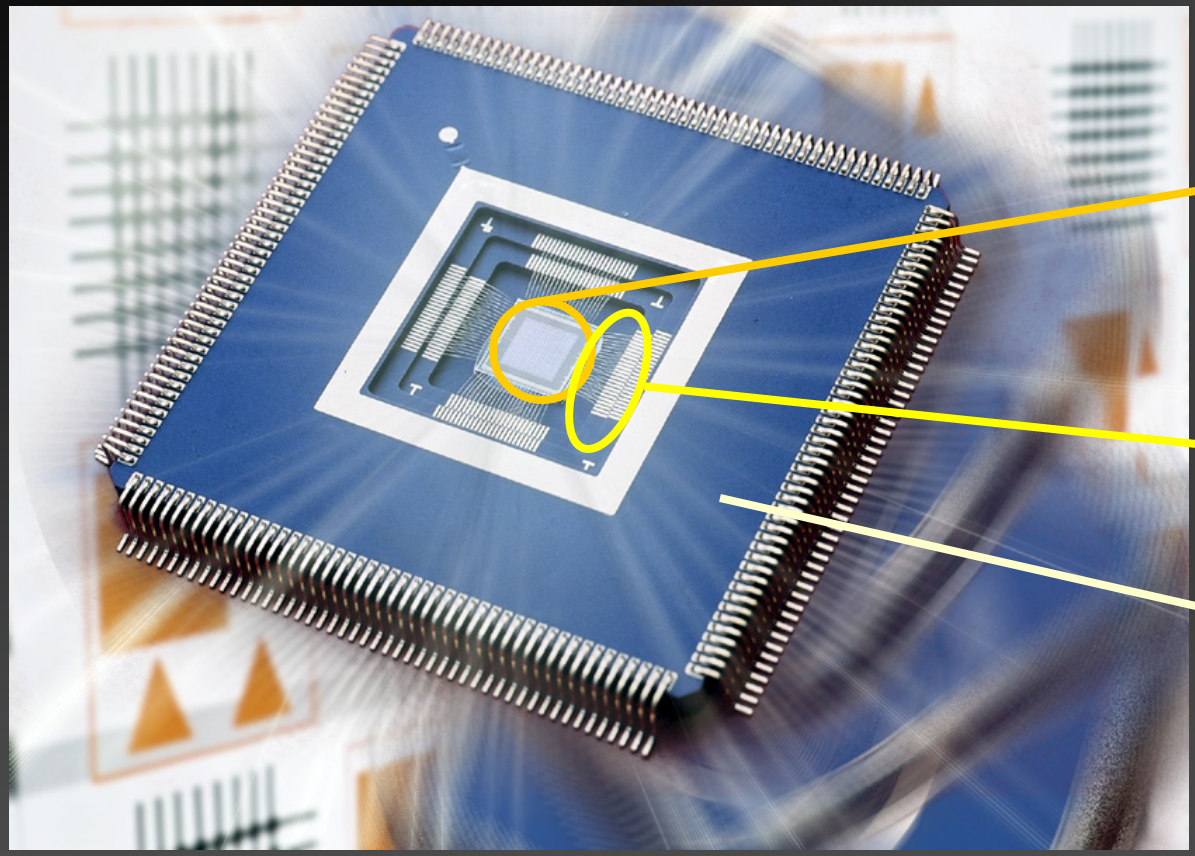


The Desert by John O'Nolan CC-BY-2.0, http://flic.kr/p/aEJ8Rk

© A. Steininger / TU Wien

Ein fertiger Chip



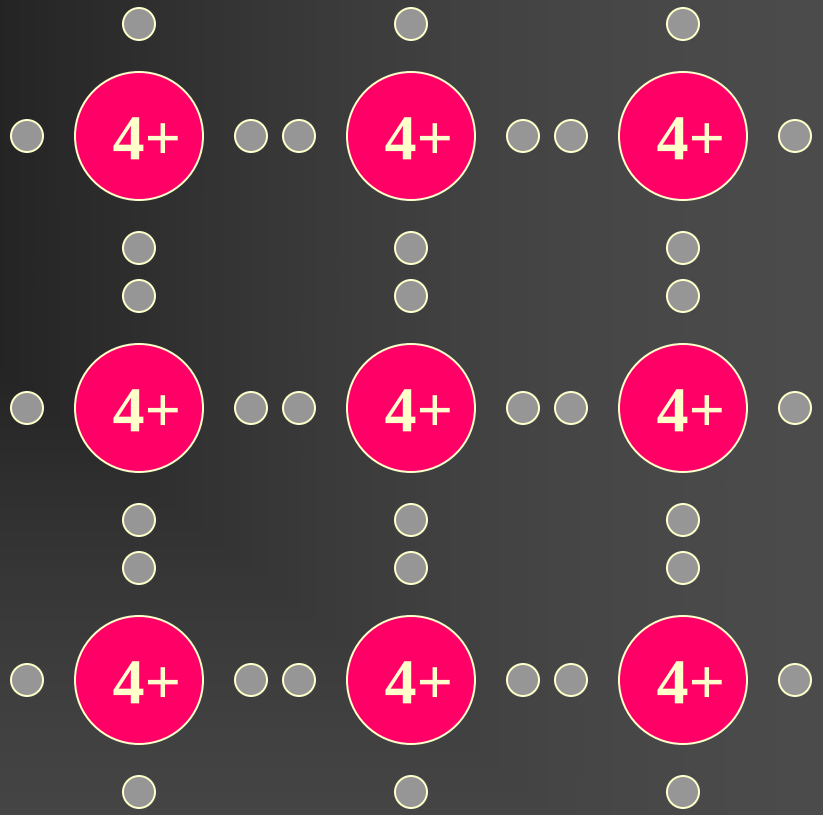


© A. Steininger / TU Wien

Silizium

- diamantähnliche Kristallstruktur
- Leitfähigkeit
  - stark temperaturabh.
  - bei 20°C schlecht
  - bei 0K Isolator

  → „Halbleiter“
- Vorkommen
  - Erdkruste (27,8 %)
  - Sand, Quarz, Kiesel,...

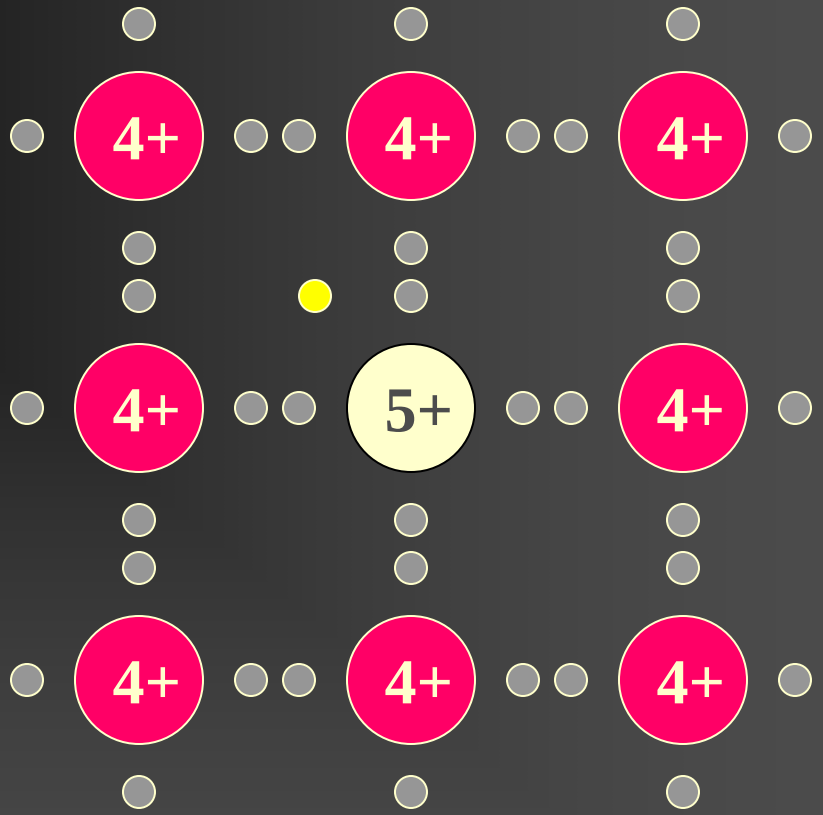


© A. Steininger / TU Wien

n-Dotierung

- Es werden vereinzelt 5-wertige Atome in das Si-Kristallgitter eingepflanzt.
- Ein Elektron ist nicht am Gitter beteiligt

  ➡ „n-Silizium“
- Ladung des Kerns hält Elektron fest, aber externes E-Feld kann es losreißen.

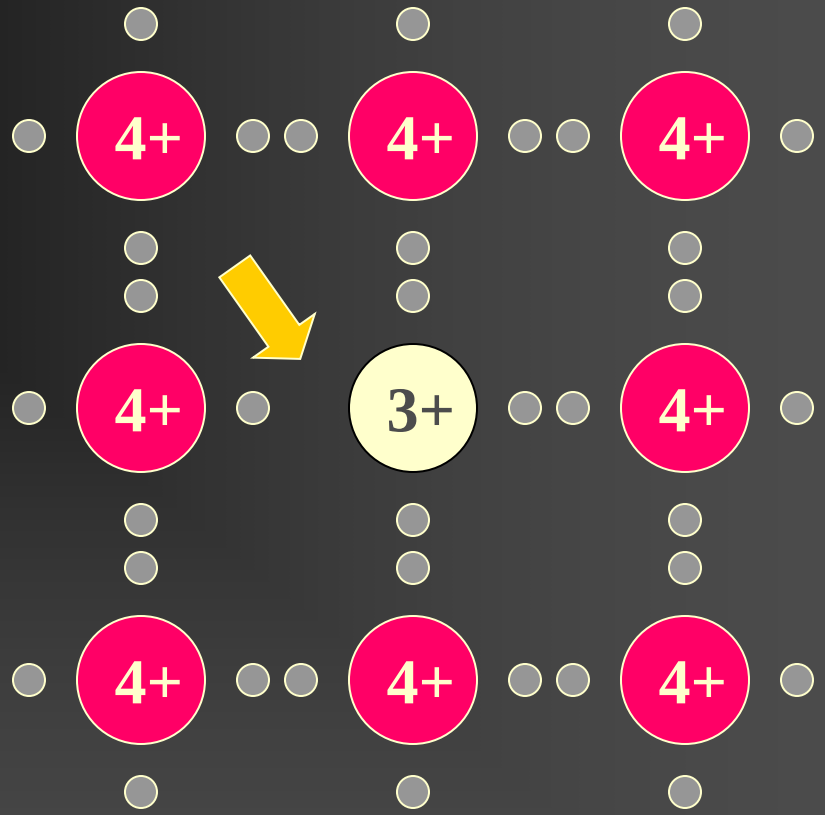


© A. Steininger / TU Wien

p-Dotierung

- Es werden vereinzelt 3-wertige Atome in das Si-Kristallgitter eingepflanzt.
- Ein Elektron fehlt im Gitter => „Loch“

  ➡ „p-Silizium“
- Loch kann sich im Si bewegen, aber schlechter als Elektron.

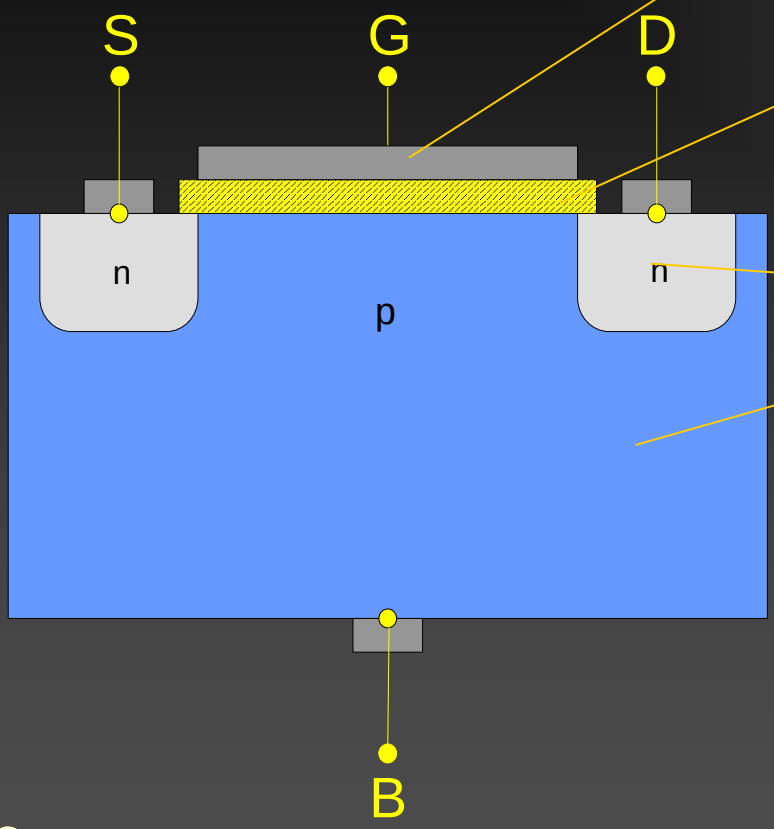


© A. Steininger / TU Wien

Der MOS-Transistor



Beispiel: n-Kanal-MOS-FET



Poly-Silizium (Kontakte)

Silizium-Dioxid (Isolator)

n-dotiertes Si

p-dotiertes Si („Substrat")

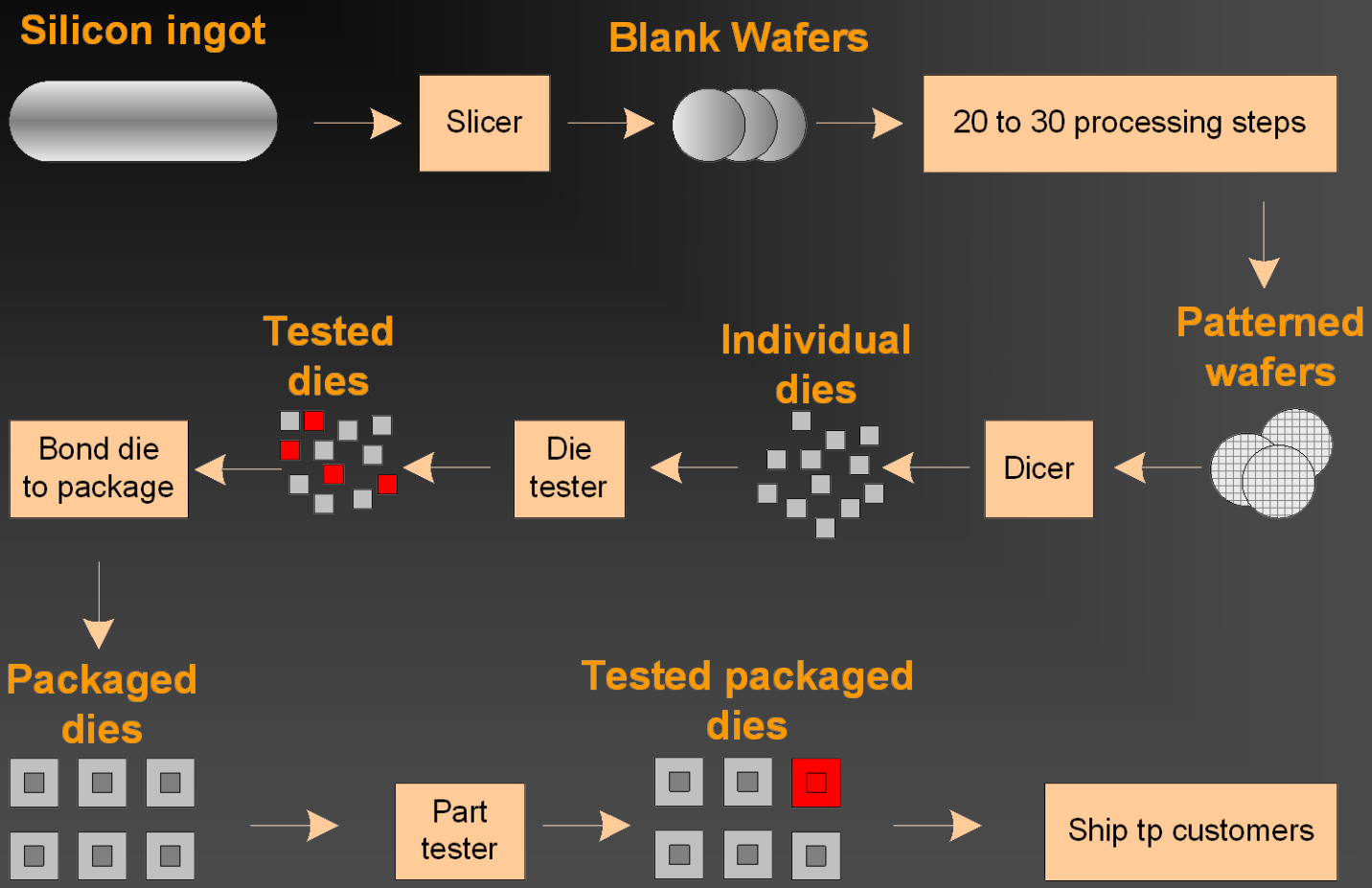
Metall *(früher)*

Oxid

Semi-conduc-tor (Si)

© A. Steininger / TU Wien

Chipfertigung im Überblick

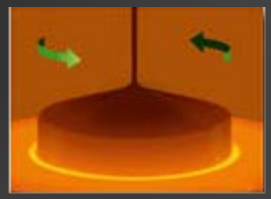


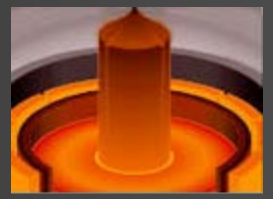
© A. Steininger / TU Wien

Vom Sand zum Wafer



- SiO_2 hoher Reinheit (99%)
  (Felsquarz, Seesand)
  div. Mahl- und Schmelzprozesse
- polykristallines Reinst-Si (10^{-9} = 99,9999999%)
  Tiegelziehen
  nach Czochralski
  (bei über 1400° C)





- Si-Einkristall mit definierter Orientierung
- Schneiden mit Diamantsäge



© A. Steininger / TU Wien

Silicon Ingot



- Durchmesser 30-40cm
- Länge 2m
- Gewicht > 100kg
- Reinheit > 10^{-9}

[Oleg Alexandrov CC-BY-SA-3.0 via Wikimedia Commons]

© A. Steininger / TU Wien

Vom Wafer zum Chip

- Oxidation: Si ⇨ SiO_2 Isolator
- Dotierung: Si ⇨ n-Si bzw. p-Si Schalter
- Anlagerung Cu, Al Verbindung

- Photolithographie

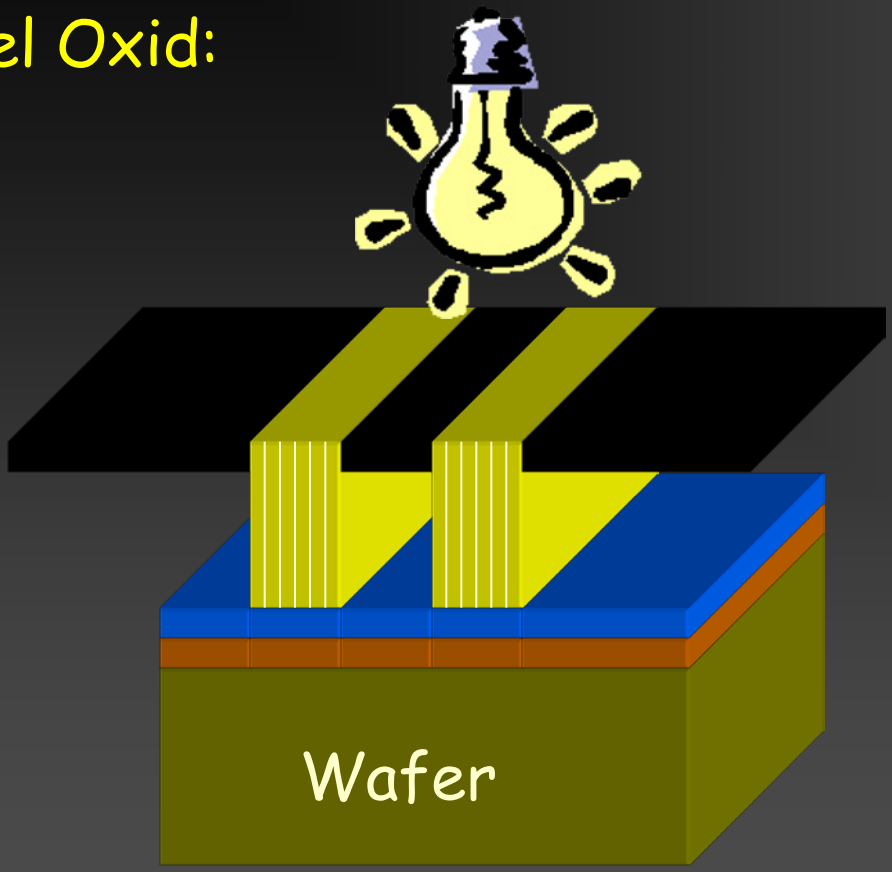
  zur gezielten Beschränkung der Prozesse
  ⇨ Abbildung der gewünschten Strukturen
- Ätzen

  Zum selektiven Entfernen von Oxid bzw. Metall

© A. Steininger / TU Wien

Ablauf der Photolithographie

Beispiel Oxid:

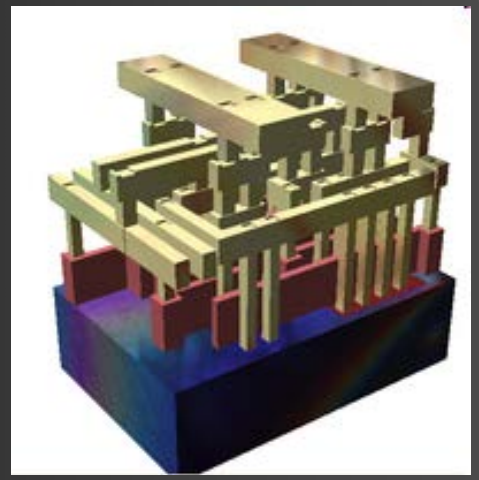


Belichten (UV)

Maske
Entwickeln
Fotolack
Oxidation
Ätzen
Reinigen

© A. Steininger / TU Wien

Aufbau eines Die



- Interconnect:

  für die Verbindungen stehen weitere 5...12 Layers zur Verfügung (Metallisierung)

- Zellen:

  die Transistoren / Zellen benötigen 5...10 Layers (Diffusionsprozesse auf dem Silizium)

  für jeden Layer gibt es eine „Maske"

© A. Steininger / TU Wien

Zweck der Masken

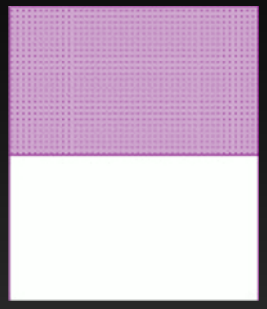


- Zur Erreichung der gewünschten Strukturen müssen die Prozess-Schritte gezielt auf kleine Bereiche angewandt werden.
- Diese strukturelle Information ist auf Masken gespeichert (vgl. Schablone).
- Photolack wird auf die gesamte Oberfläche aufgebracht, danach über die Maske belichtet.
- In den belichteten Bereichen lässt sich der Lack abwaschen, in den nicht belichteten verbleibt er als Abdeckung.

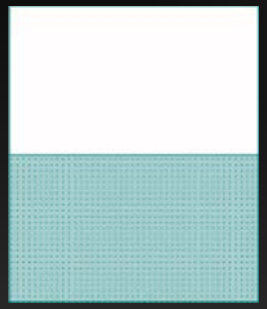
© A. Steininger / TU Wien

Beispiel für Masken

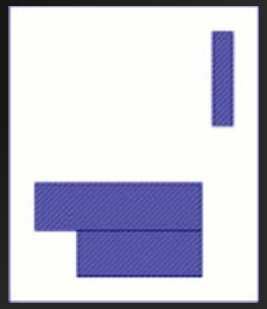




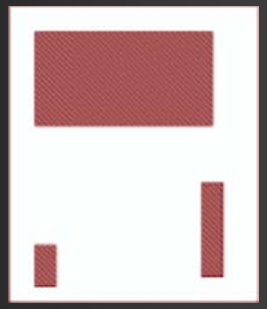
n well



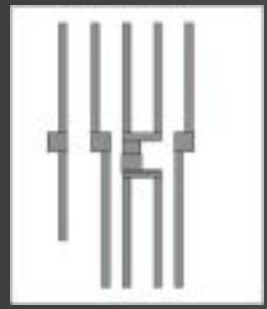
p well



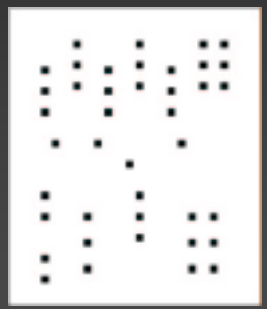
n diff



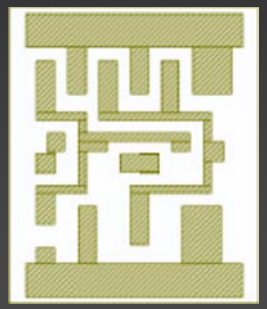
p diff



poly



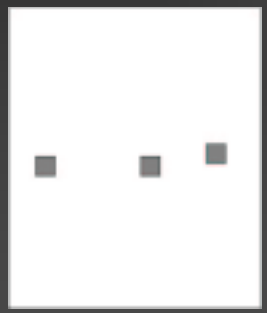
contact



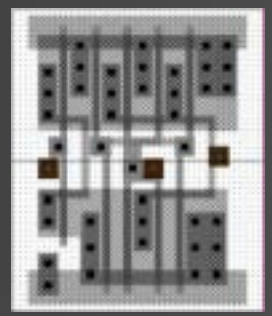
metal 1



via



metal 2



© A. Steininger / TU Wien

Feature Size λ

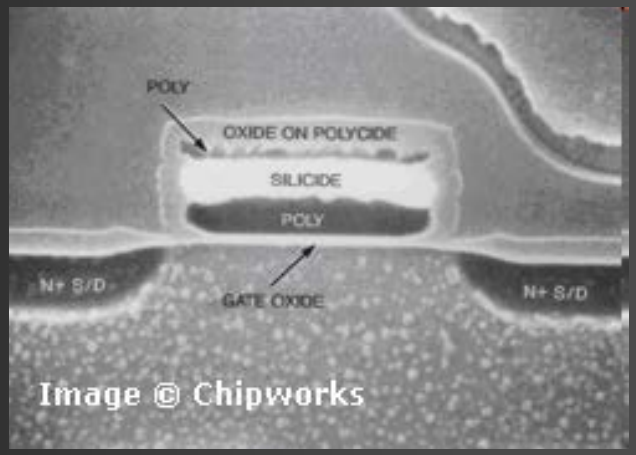


- kleinste darstellbare Strukturgröße
  - ➡ Maß für den Technologiefortschritt
- Angegeben wird meist die Kanallänge 2λ
- Diese liegt derzeit (2018) bei 0.014μm

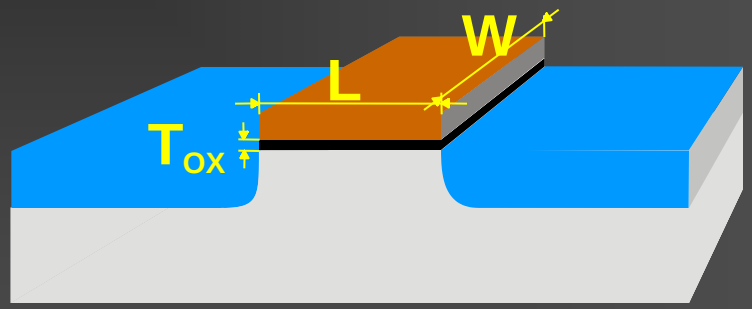
  Vergleiche: menschl. Haar: 100μm
  Staubkorn: 50μm
- Mit der Feature Size
  - sinkt die Fläche des Transistors ($\propto \lambda^2$)
  - steigt die Geschwindigkeit des Transistors ($\propto \lambda$)
  - sinkt der Leistungsverbrauch des Transistors

© A. Steininger / TU Wien

Typische Prozessparameter

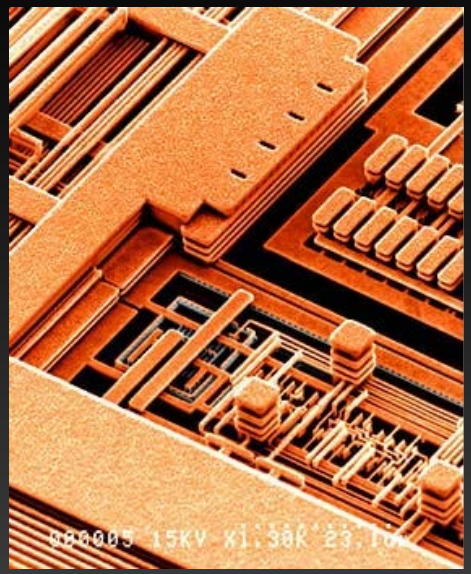


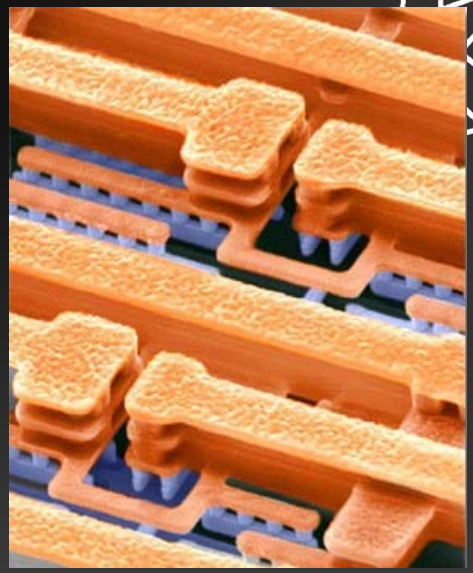
- Kanal: Länge L: 2λ
  Breite W: 3λ
- Oxid: Dicke T_{ox} < 1nm
  (Atomdurchmesser Si ≈0.25nm !)
- Diffusionsbereich:
  Länge 3λ
- Metall-Verbindungen:
  Breite 3λ
  Abstand 3λ

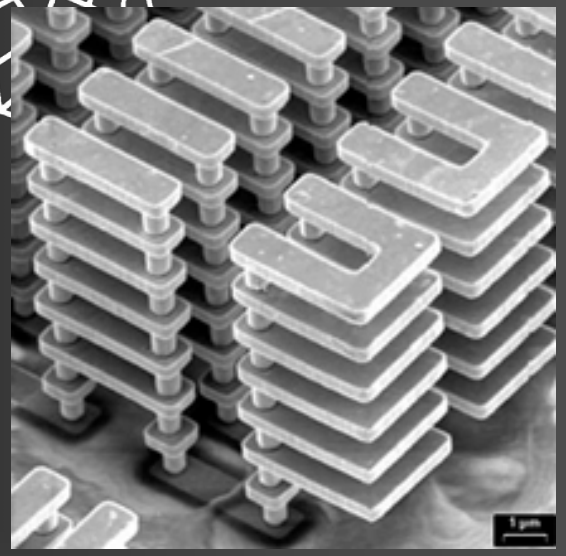


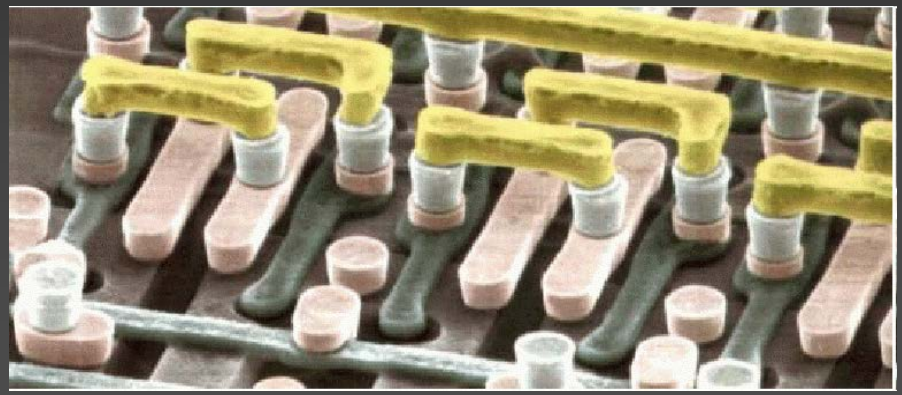
© A. Steininger / TU Wien

Interconnect







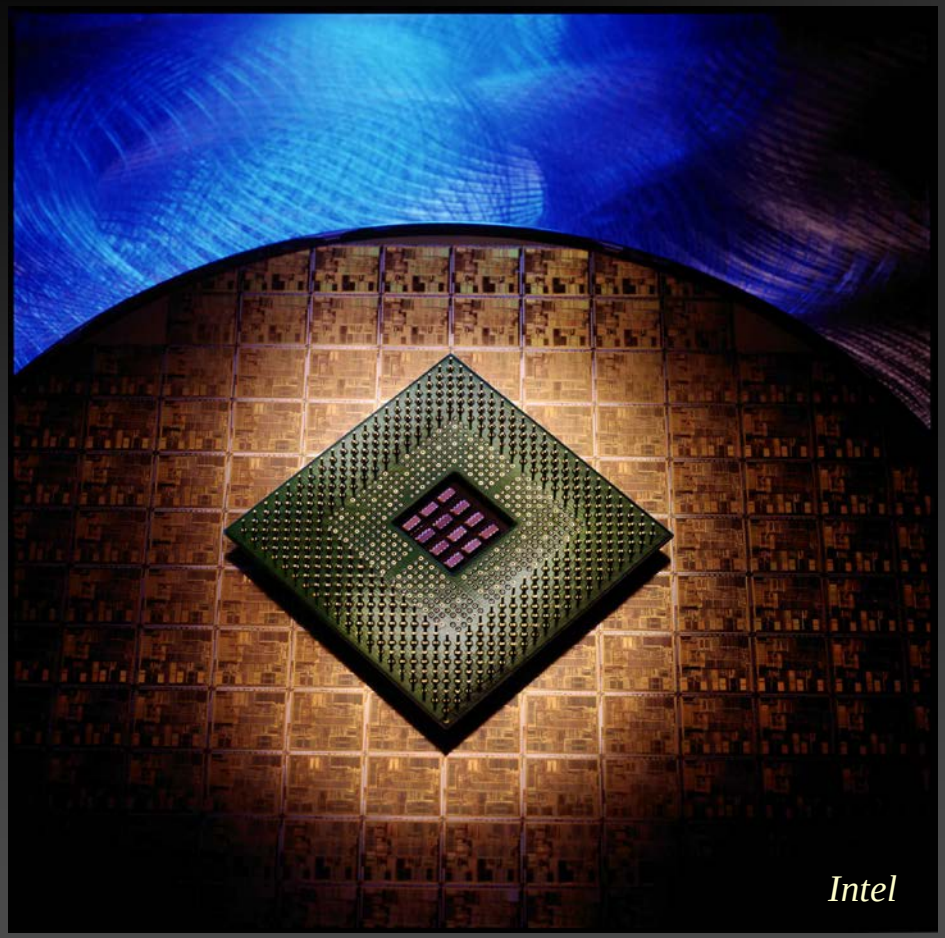


VLSILab Politechnico Torino

Mehrere Lagen aus Cu-"Leiterbahnen" (früher Al) verbinden die Transistoren, dazwischen jeweils Isolation & Durchkontaktierungen („Vias")

© A. Steininger / TU Wien

Der bearbeitete Wafer





∅ = 30 cm („Pizza")

Dicke ≈ 0,5mm

© A. Steininger / TU Wien

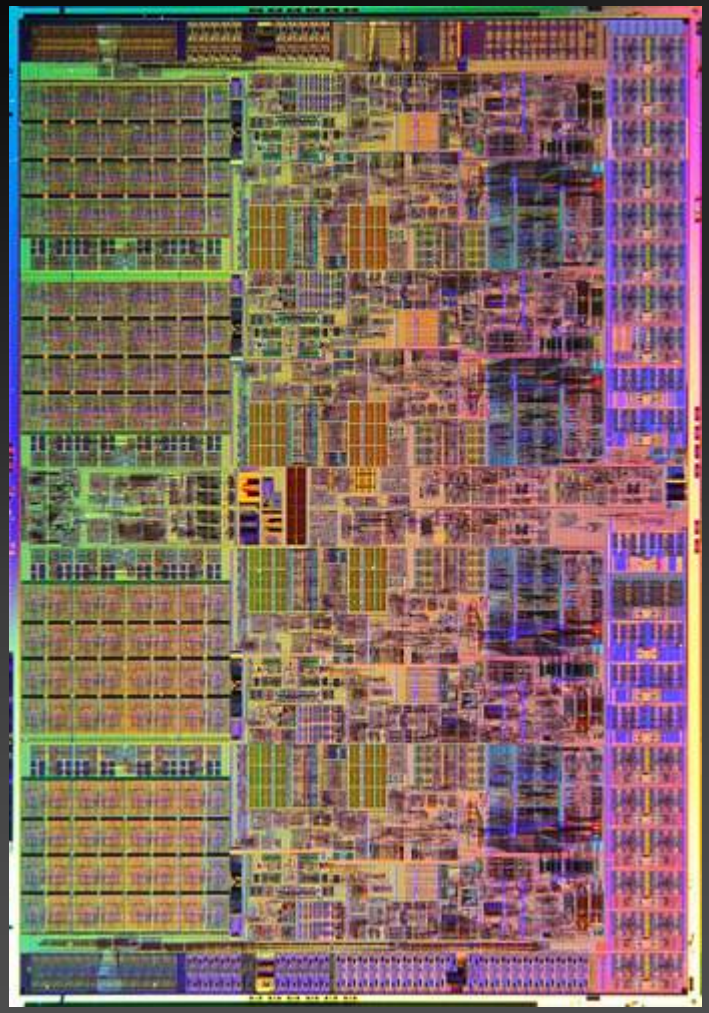
Der Wafertest

- Jeder Chip wird mit Prüfspitzen kontaktiert und getestet
- Defekte Chips werden mit Farbe markiert



© A. Steininger / TU Wien

Der fertige Die



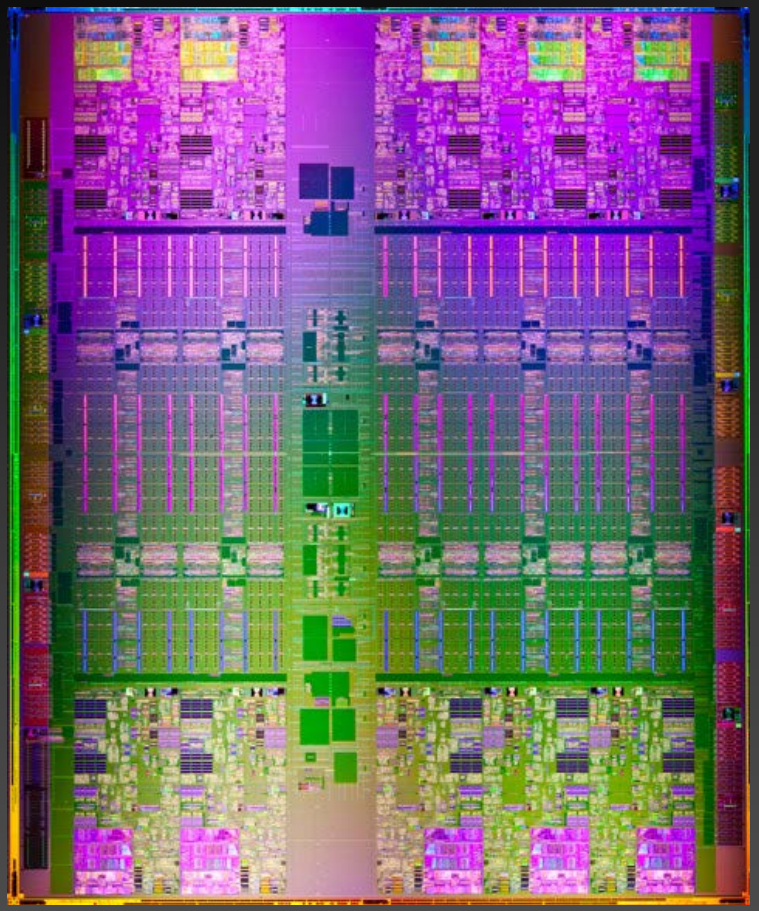
Intel Bloomfield (Core i7-920):

- 1,9 Milliarden Transistoren
- 3,3 GHz
- 0,032 µm

(Stand 2009)

© A. Steininger / TU Wien

Stand 2011

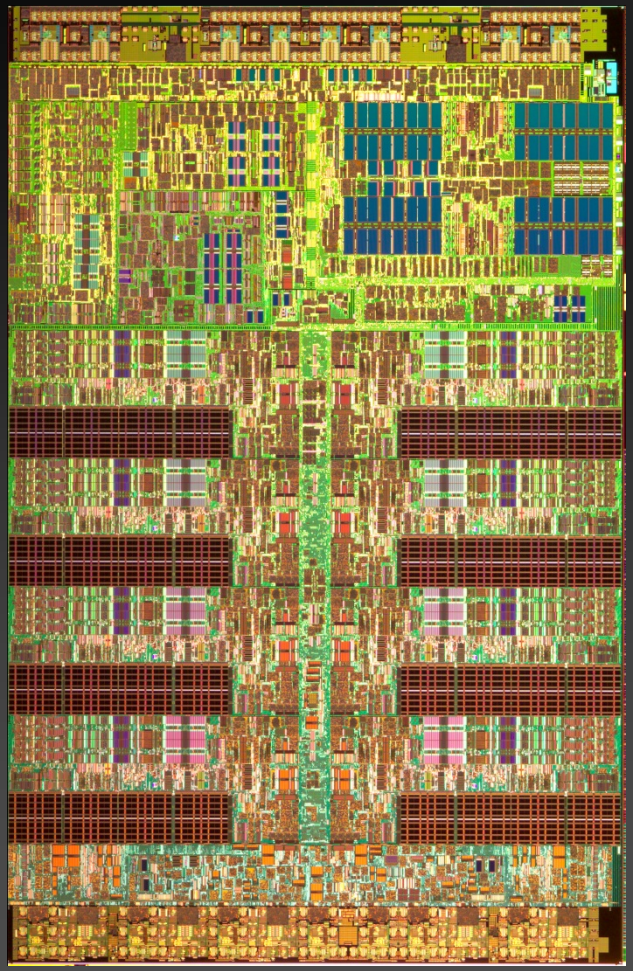


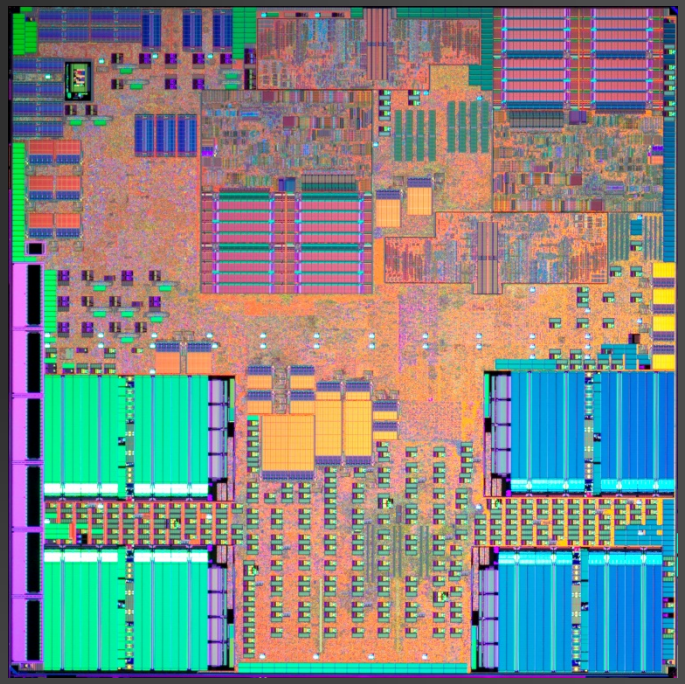
Intel Xeon E7

- 2.6 Mrd. Transistoren
- 32nm-Technologie
- 3.4 GHz
- 6, 8, 10 Prozessorkerne
- 130 Watt

© A. Steininger / TU Wien

Weitere Beispiele



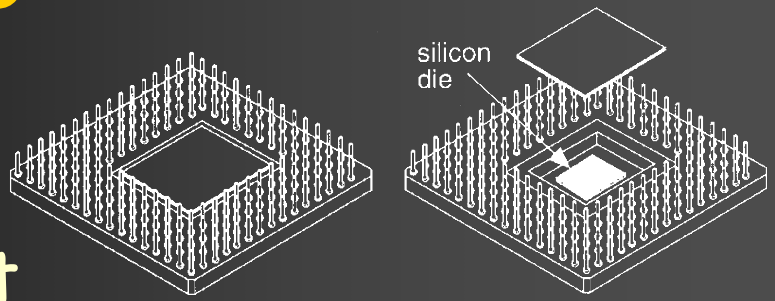


Blue Gene [IBM]

Cell Processor [IBM]

© A. Steininger / TU Wien

Packaging: Prinzip



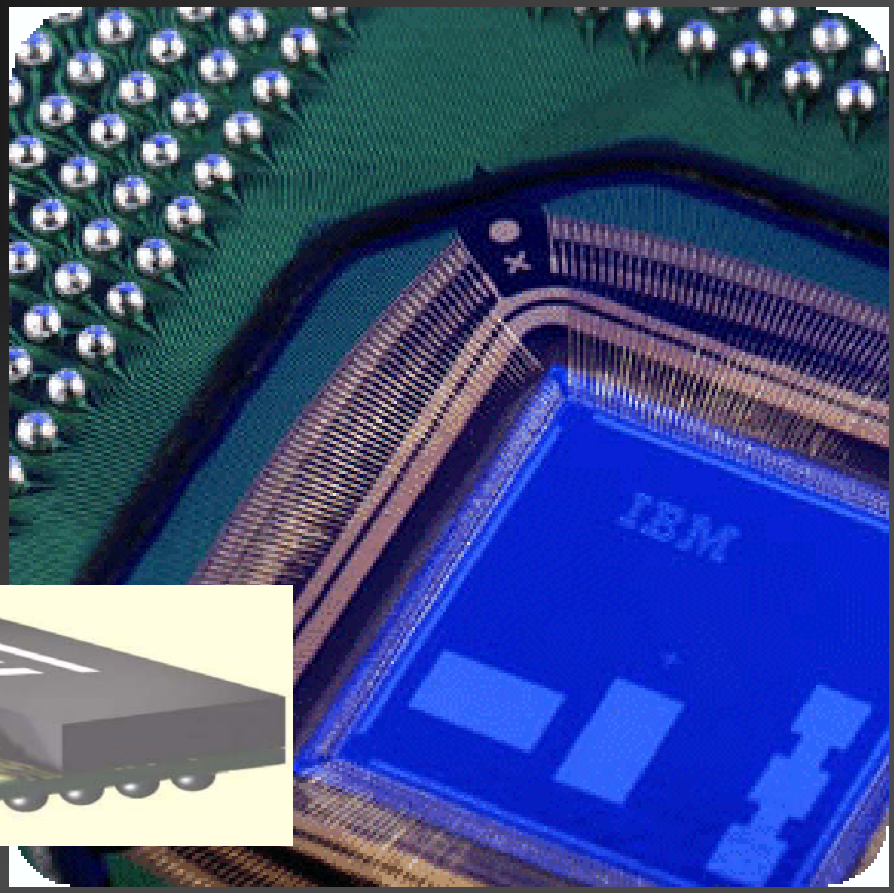
- Der fertige Die wird mit Passivierung überzogen
- Der Chip kommt in ein Gehäuse (Package): mechanische Befestigung (die attach) und Kontaktierung der Anschlüsse (bonding)
  - Schutz
  - Standard-Kontaktierung auf der Leiterplatte
  - Wärmeabfuhr
- „Flip-Chip“: ohne Package auf Leiterplatte

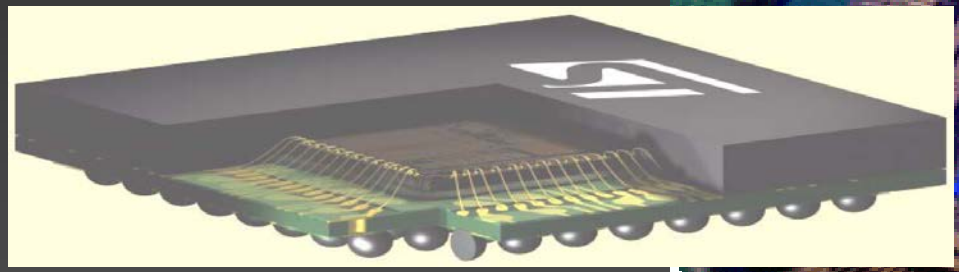
© A. Steininger / TU Wien

Packaging: Beispiele



Itanium



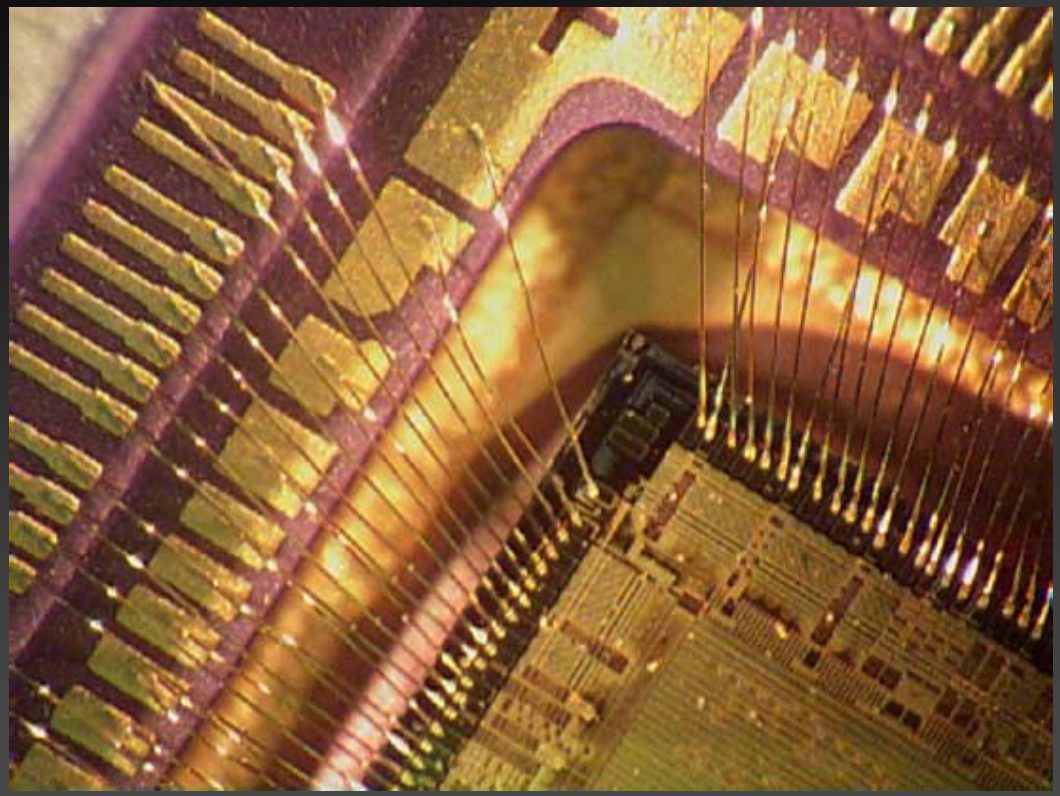


[C.Cognetti, ST Micro]

© A. Steininger / TU Wien

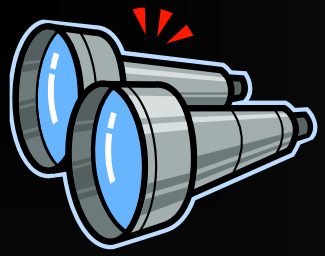
Bonding





Dünne Drähte verbinden die Kontakte auf dem Die mit den Package-Pins

© A. Steininger / TU Wien

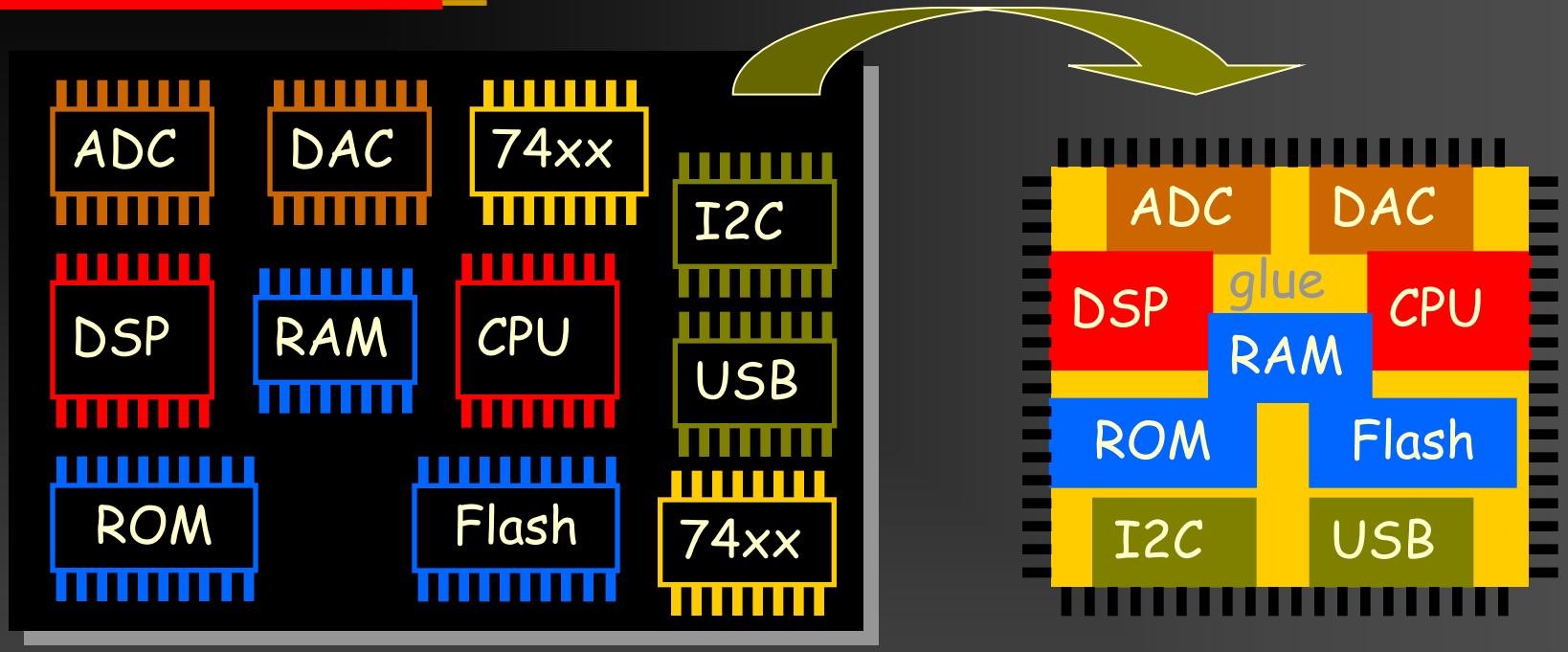


Überblick

- Motivation für ICs
- Entwicklungstrends in der IC-Technologie
- Aufbau & Fertigung eines Chips
- **Moderne Formen von Chips (MCM, SOC)**
- Grenzen der Technologie

© A. Steininger / TU Wien

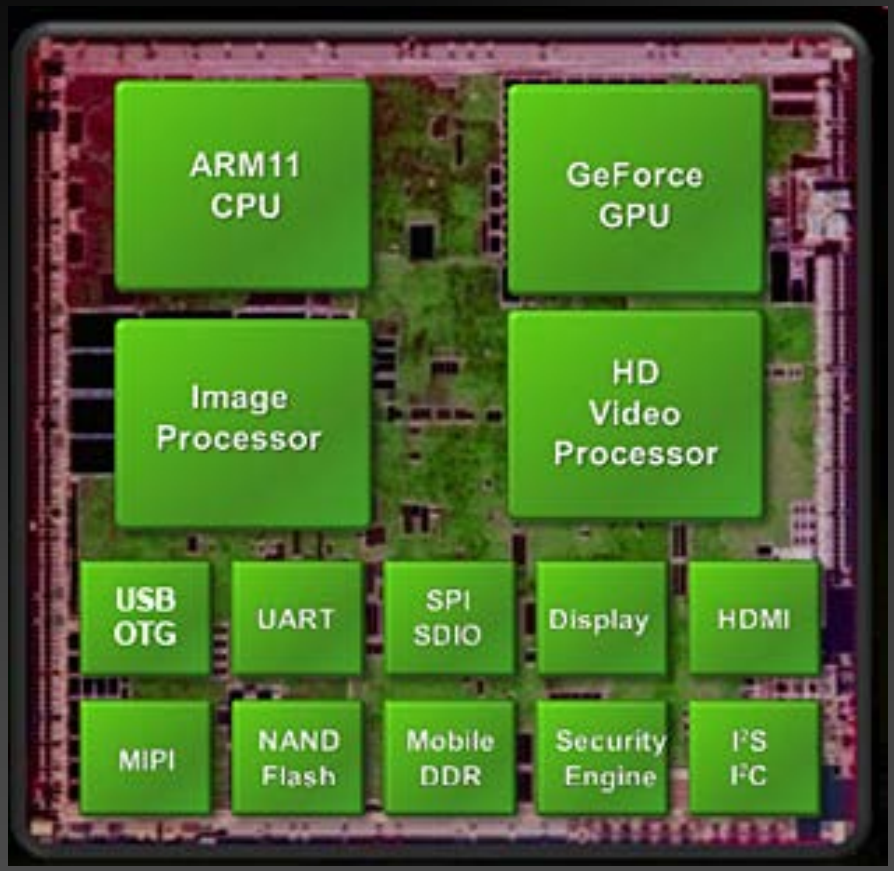
System on a chip (SOC)



- alle für die Anwendung benötigten Funktionen werden auf einem Chip (Die) untergebracht
  - ➡ weitere Platzersparnis

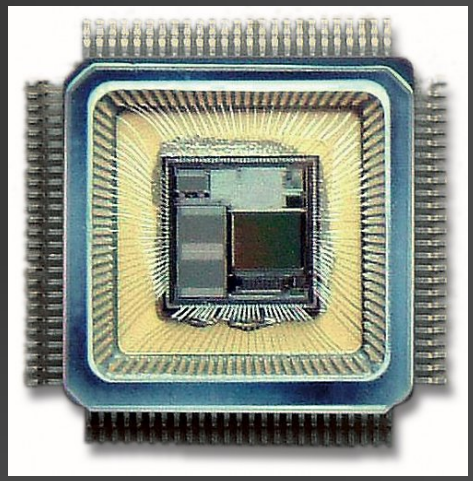
© A. Steininger / TU Wien

SoC: Beispiele



[Nvidia Tegra]

Vision system on chip
(incl. Bildsensor)



[Neuricam]

© A. Steininger / TU Wien

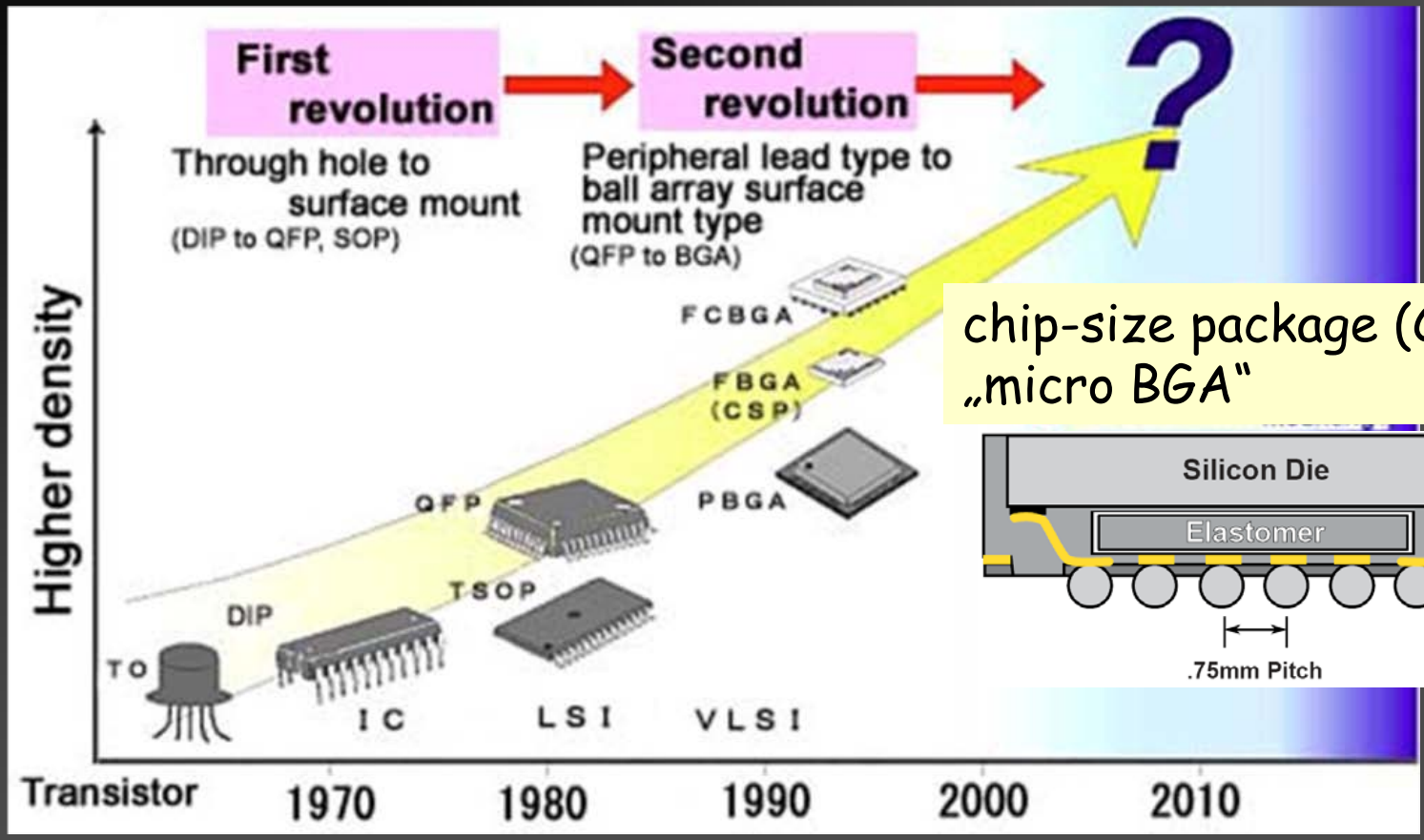
SoC: Pros und Cons

- Vorteile:
  - hohe Performance (kurze Leitungen)
  - billige Produktion
  - geringe Baugröße
- Probleme
  - hohe Komplexität für Design & Fertigung
  - sub-optimale Technologie (Logik, Speicher, analog auf einem Die)
  - manches ist nicht integrierbar (div. Sensoren, passive Komp.)

© A. Steininger / TU Wien

Packaging: Trends

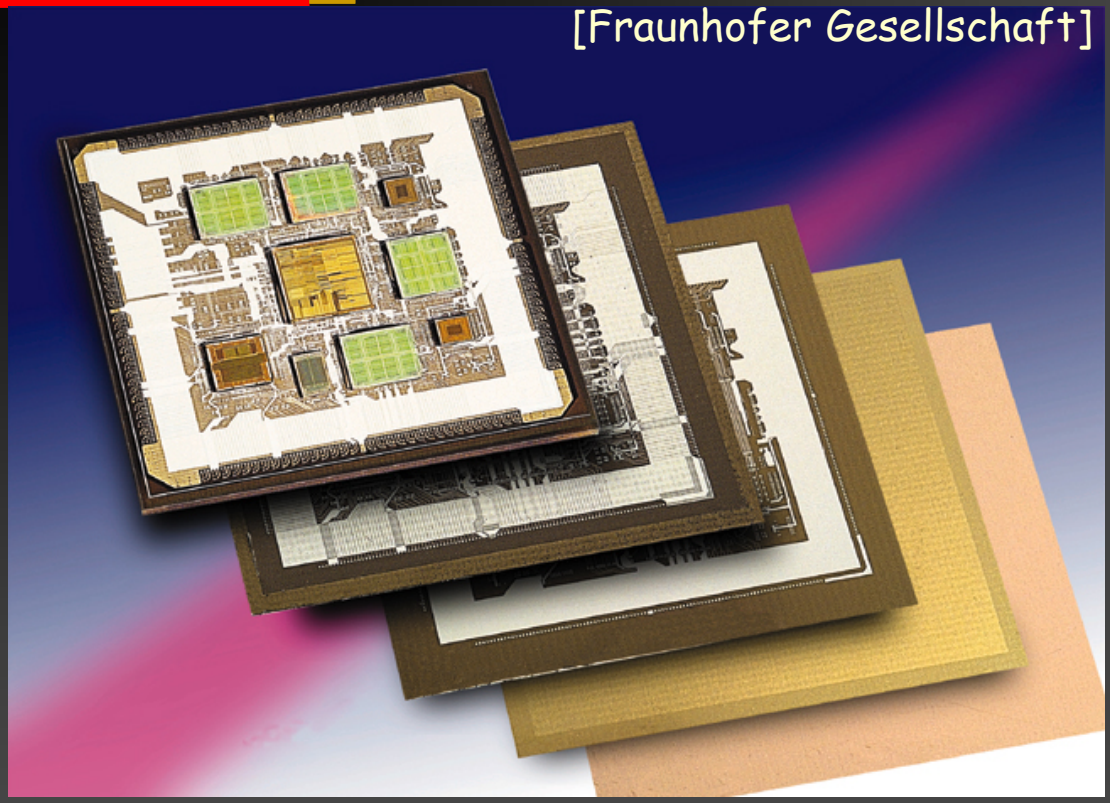




chip-size package (CSP)
„micro BGA"

© A. Steininger / TU Wien

Multichip-Module (MCM)

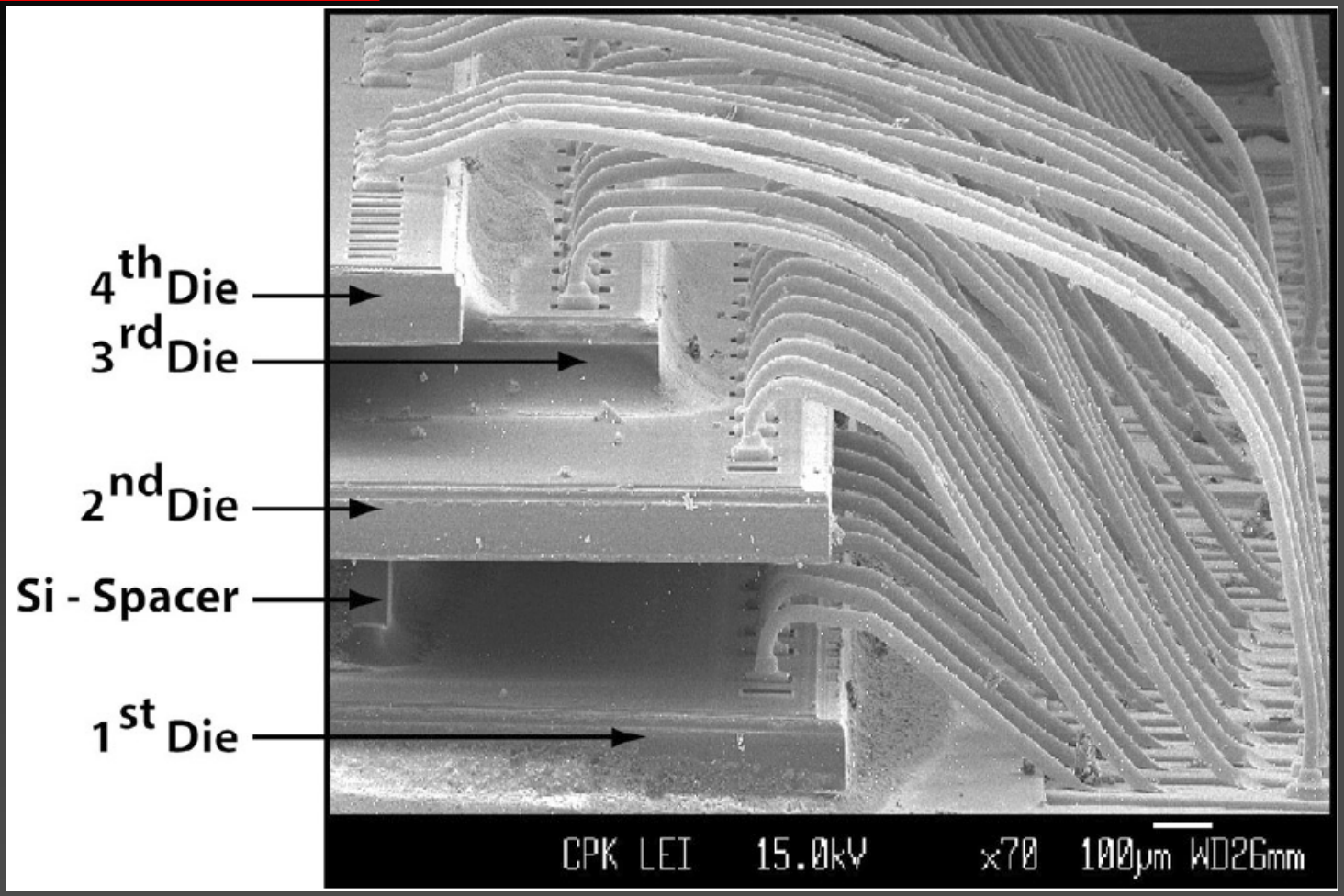


mehrere Dies in einem Gehäuse
=> „System in Package (SiP)“

kleiner
billiger

© A. Steininger / TU Wien

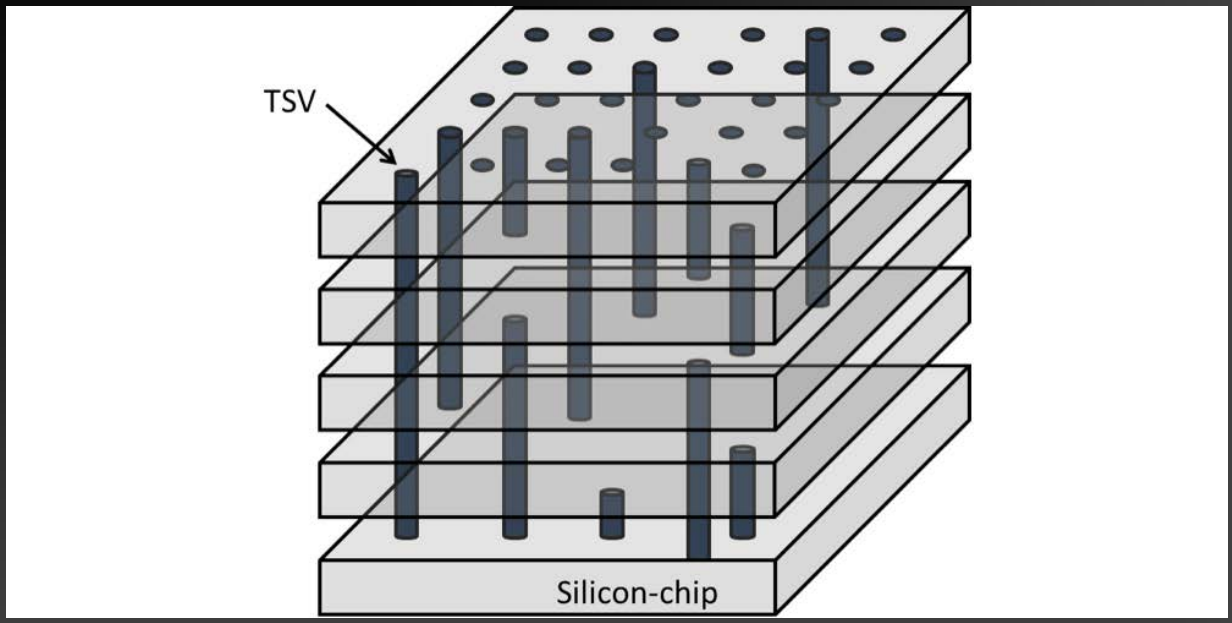
Stacked Die Package



[E.J.Vardaman, TechSearch Inc.]

© A. Steininger / TU Wien

Through Silicon Via (TSV)



[http://www.nims.go.jp]

Mehrere Dies übereinander, mit "Durchkontaktierungen"

- Vorteile: kürzere Wege, kleinere Dies
- Probleme: Wärmeabfuhr, Test, Passgenauigkeit,…

© A. Steininger / TU Wien

„More than Moore"

- „More Moore":
  - Miniaturisierung der Halbleiterstrukturen erlaubt weiterhin Anstieg von Komplexität und Integrationsdichte: System on Chip (SoC)
- „More than Moore":
  - Fortschritte in der Packaging-Technologie ermöglichen eine Zunahme von Komplexität und Integrationsdichte weit über Moore's Law hinaus: System in Package (SiP), 3D Packaging

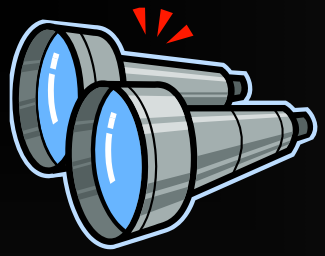
© A. Steininger / TU Wien

Test



- selbst bei perfekt fehlerfreiem Design gibt es noch Fehlerquellen
  - im Wafer-Material
  - bei der Lithographie
  - in den Prozessen
  - beim Packaging
  - ...
- Der Chip muss unbedingt getestet werden (Details siehe später)

© A. Steininger / TU Wien

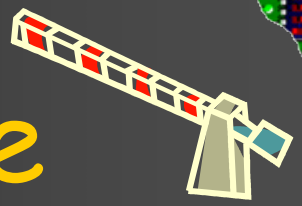


Überblick

- Motivation für ICs
- Entwicklungstrends in der IC-Technologie
- Aufbau & Fertigung eines Chips
- Moderne Formen von Chips (MCM, SOC)
- **Grenzen der Technologie**

© A. Steininger / TU Wien

Grenzen der Technologie



für die weitere Miniaturisierung sind viele Grenzen abzusehen:

- physikalische Grundgesetze
- materialbedingte Grenzen
- strukturbedingte Grenzen
- fertigungsbedingte Grenzen
- wirtschaftliche Grenzen
- Grenzen des Interconnect

bisher wurden Grenzen stets überwunden ...

© A. Steininger / TU Wien

Grenzen der Miniaturisierung

- kleinste Ladungseinheit ist das Elektron

  $e = -1{,}602 \cdot 10^{-19}\ C$

- Isolator bricht bei hoher Feldstärke durch

  Feldstärke = Spannung/Dicke = V_{DD}/T_{ox}

  (Maß für Kraft auf Ladungsträger im Isolator)

- Größe der Atome

  Ø Si-Atom ≈ 0.25nm, T_{ox} < 10 Atomlagen

- „Tunnelströme" durch dünne Isolatoren

  steigen exponentiell an: - bei dünnerem Gate-Oxid

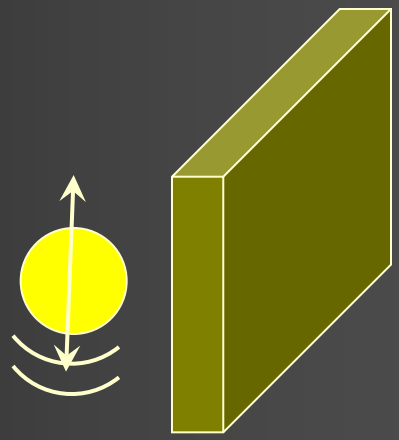
  - bei kürzerem Kanal

© A. Steininger / TU Wien

Der Tunneleffekt



- klassische Mechanik:
  - Elektron kann Potential-barriere nur dann überwinden, wenn Energie groß genug ist.
- Quantenmechanik:
  - Es gibt Wahrscheinlichkeit dafür dass auch Elektron mit zu wenig Energie durch Barriere „hindurchtunnelt".



© A. Steininger / TU Wien

Miniaturis.: weitere Grenzen

- Wellenlänge des Lichts
  sichtbarer Bereich: 750...400nm, UV 400... ca. 150nm
- Molekülgröße des Photolacks
- „Statistik" der Dotation gilt nicht mehr
  nur mehr <100 n- / p-Atome je Diffusionsbereich
- „Bändermodell" gilt nicht mehr
- Die thermische Leitfähigkeit ist begrenzt
  bei gleicher Leistung lokale Überhitzung am Transistor
- Investitionskosten
  machen das Risiko für neue Technologie bald untragbar

© A. Steininger / TU Wien

Chip Fab – ein Beispiel

- Beispiel 20nm Technologie
  - Fab kostet 5 Mrd $
  - Durchsatz: 40000 wafers/Monat
  - Kosten/Wafer: 1000…2000$
  - Amortisation nach frühestens 5…10 Jahren

[Quelle: A. Strojwas, Keynote DDECS 2011]

© A. Steininger / TU Wien

Fertigungskosten

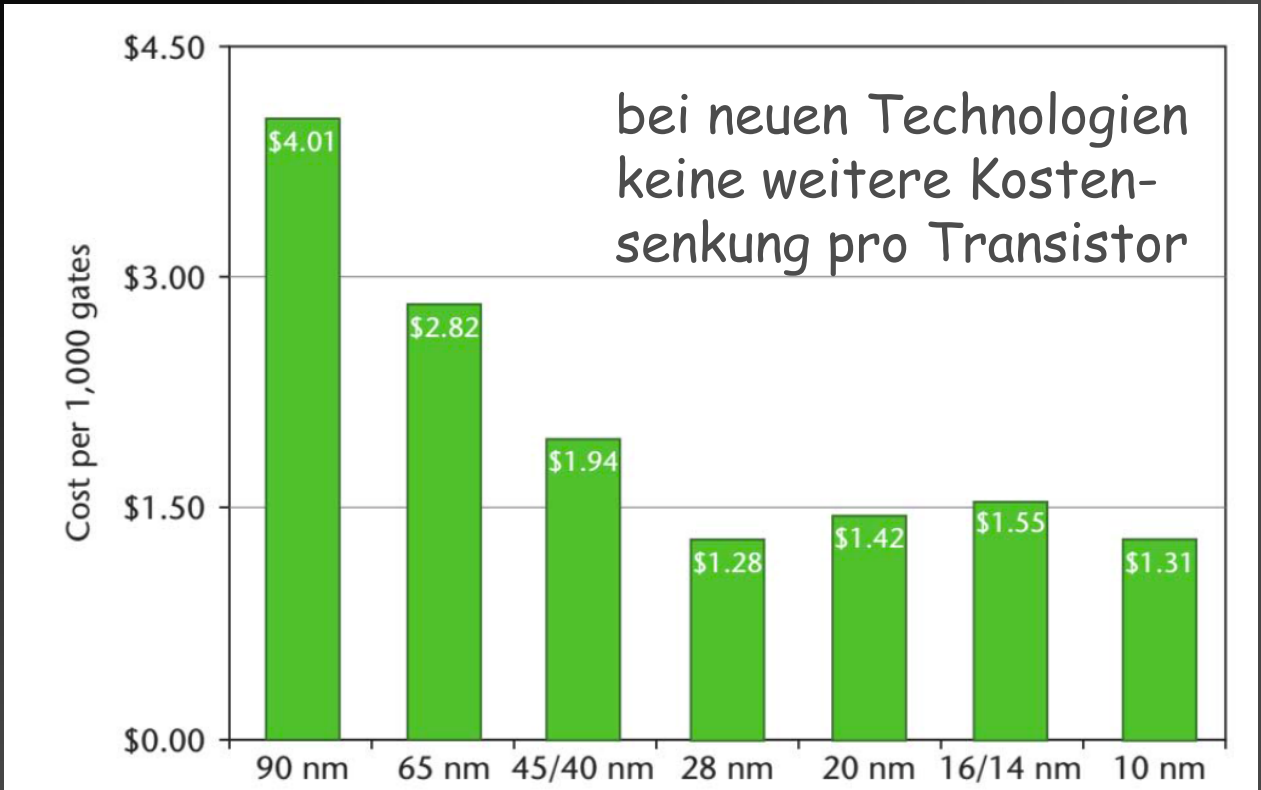


Figure 6. Cost per logic gate, with projection for 10nm technology node

Source: Jones (2015)

© A. Steininger / TU Wien

Der Transistor der Zukunft

- „Strained Silicon“
  - „gestrecktes“ Kristallgitter vermindert Kollisionen => schneller, höhere Ströme
- Gate aus Metall statt Poly-Si +
- „High-k“-Dielektrikum statt SiO_2 +
- Gate beidseitig vom Kanal (3D Struktur)
  - Bessere Steuerbarkeit eines kurzen Kanals
- „Silicon on Insulator“ (SOI)
  - Substrat wird zuerst mit SiO_2 „überzogen“ => Weniger Kapazitäten, weniger Leckströme

© A. Steininger / TU Wien

Silicon on Insulator (SOI)

- Dicke des Wafers (ca. 0.5mm)
  - dient nur seiner mechanischen Festigkeit
  - Transistorstrukturen nutzen nur Oberfläche
- darunter verbleibendes Si stört zunehmend:
  - parasitäre Kapazitäten
  - Leckströme
  - Anfälligkeit für Störungen
- Abhilfe: Isolator als Träger
  - (z.B. Saphir, SiO_2),
  - nur dünne Si-Schicht an Oberfläche

© A. Steininger / TU Wien

3D-Struktur: FINFET

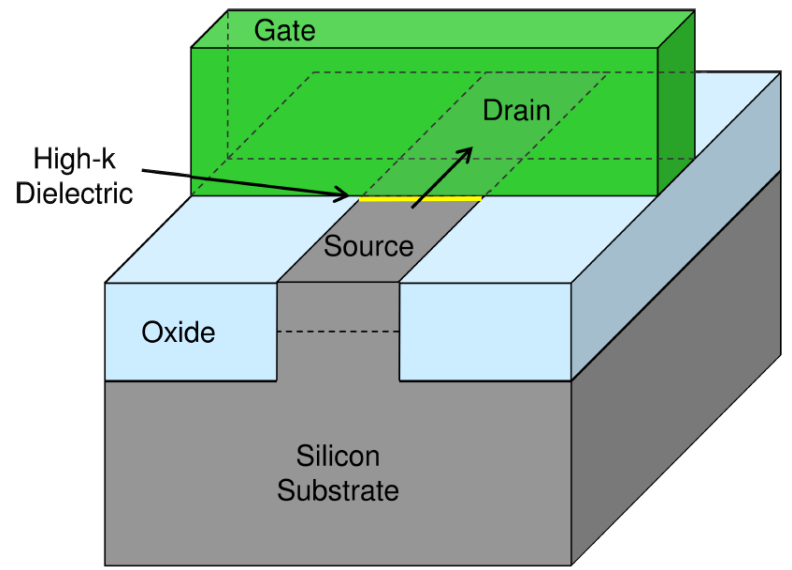


Abbildung: Traditionalles FET-Transistordesign © Intel

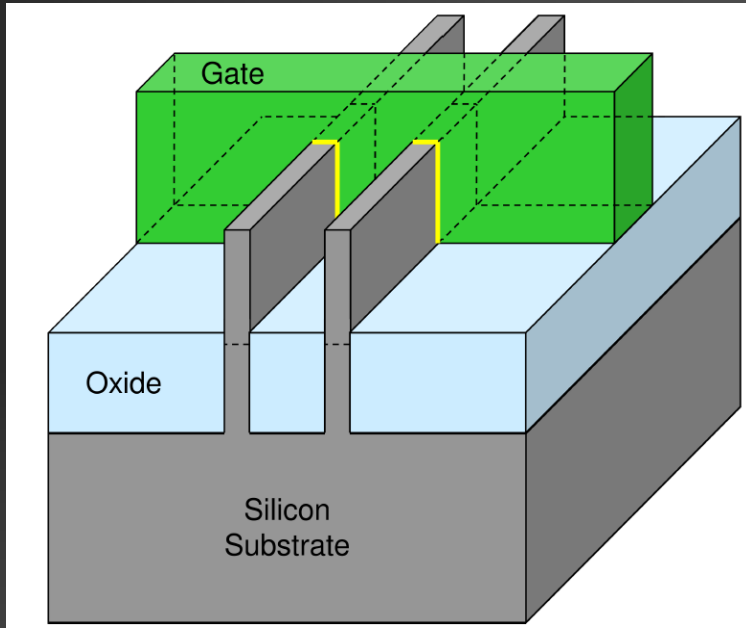


Abbildung: 14 nm FinFET-Transistordesign © Intel

© A. Steininger / TU Wien

Mögliche Alternativen

- Quantencomputer
  Elektronenspin als Informationsträger
- „Nanotubes“
  zylindrische Röhrchen aus Kohlenstoff;
  erlauben Aufbau von Transistoren
- Molekular-Elektronik
  auf Basis von Benzolringen
  Transistor und Speicher

© A. Steininger / TU Wien

Grenzen der Geschwindigkeit



- Wellenausbreitung
  - Information kann sich niemals schneller als mit Lichtgeschwindigkeit ausbreiten.
- Ladevorgänge
  - Das Laden von Kapazitäten mit begrenztem Strom beansprucht Zeit.
- Bewegung der Ladungsträger
  - Bewegung/Diffusion von Ladungsträgern im Halbleiter erfolgt nur mit begrenzter Geschwindigkeit. (Sättigungswert bei Si typ. 0,1 mm/ns)

© A. Steininger / TU Wien

Wellenausbreitung

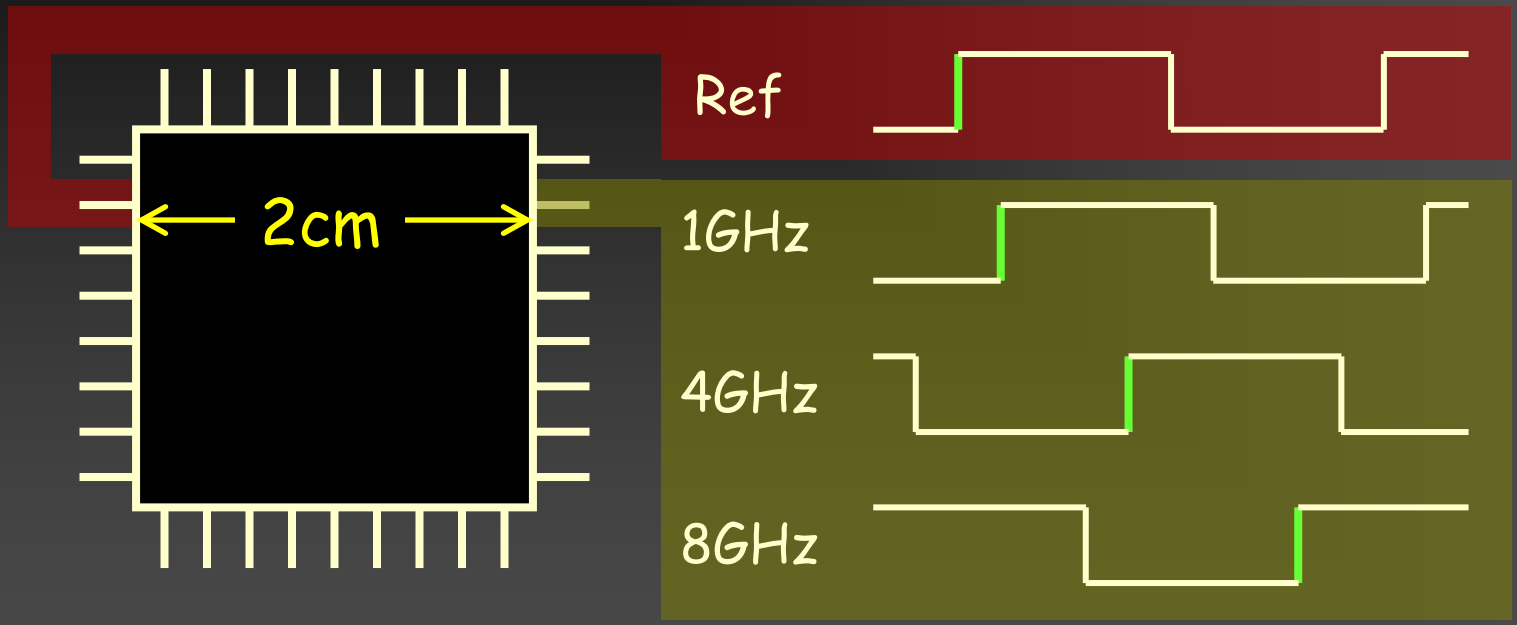


- Elektrische Signale und die damit verbundenen elektromagnetischen Wellen breiten sich mit Lichtgeschwindigkeit aus.
- Die Lichtgeschwindigkeit beträgt im Vakuum 3×10^8 m/s , das sind 30cm/ns.
- Im Medium ist die Lichtgeschwindigkeit (abh. von Materialeigenschaften) stets geringer als im Vakuum. Auf einer Leitung beträgt sie typisch 2/3 der Vakuumlichtgeschwindigkeit , also 20cm/ns.
- Während der gesamten Periodendauer eines 10GHz-Taktes (0,1ns) legt ein elektrisches Signal auf einer Leitung also etwa 2cm zurück.

© A. Steininger / TU Wien

„Isochronic" Regions ?

Phasenverschiebung beim Queren eines Die



Grenzen der Geschwindigkeit

- ✓ Wellenausbreitung
  - Information kann sich niemals schneller als mit Lichtgeschwindigkeit ausbreiten.
- Ladevorgänge
  - Das Laden von Kapazitäten mit begrenztem Strom beansprucht Zeit.
- Bewegung der Ladungsträger
  - Bewegung/Diffusion von Ladungsträgern im Halbleiter erfolgt nur mit begrenzter Geschwindigkeit. (Sättigungswert bei Si typ. 0,1 mm/ns)

© A. Steininger / TU Wien

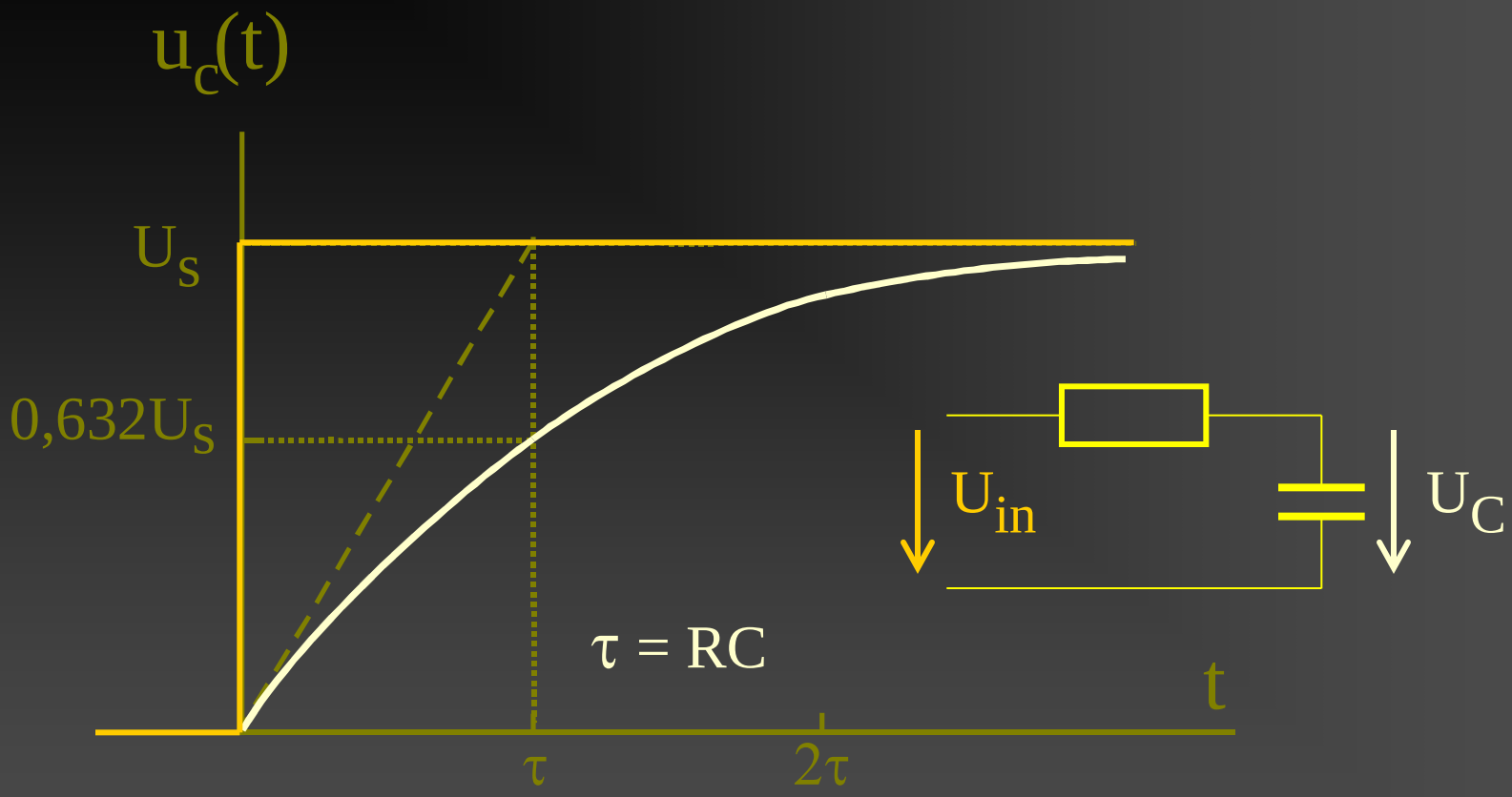
Ladevorgang am Kondensator

- Legt man an eine Anordnung aus Widerstand R und Kondensator C einen Spannungssprung, verändert sich die Spannung am Kondensator niemals sprunghaft, sondern folgt einer definierten Ladekurve.
- Die „Trägheit" der Ladekurve ist bestimmt durch die Zeitkonstante τ, die sich aus dem Produkt aus R und C ergibt.
- Große Werte von R und C führen zu einem langsamen Ladevorgang und folglich zu einer größeren Signalverzögerung.

© A. Steininger / TU Wien

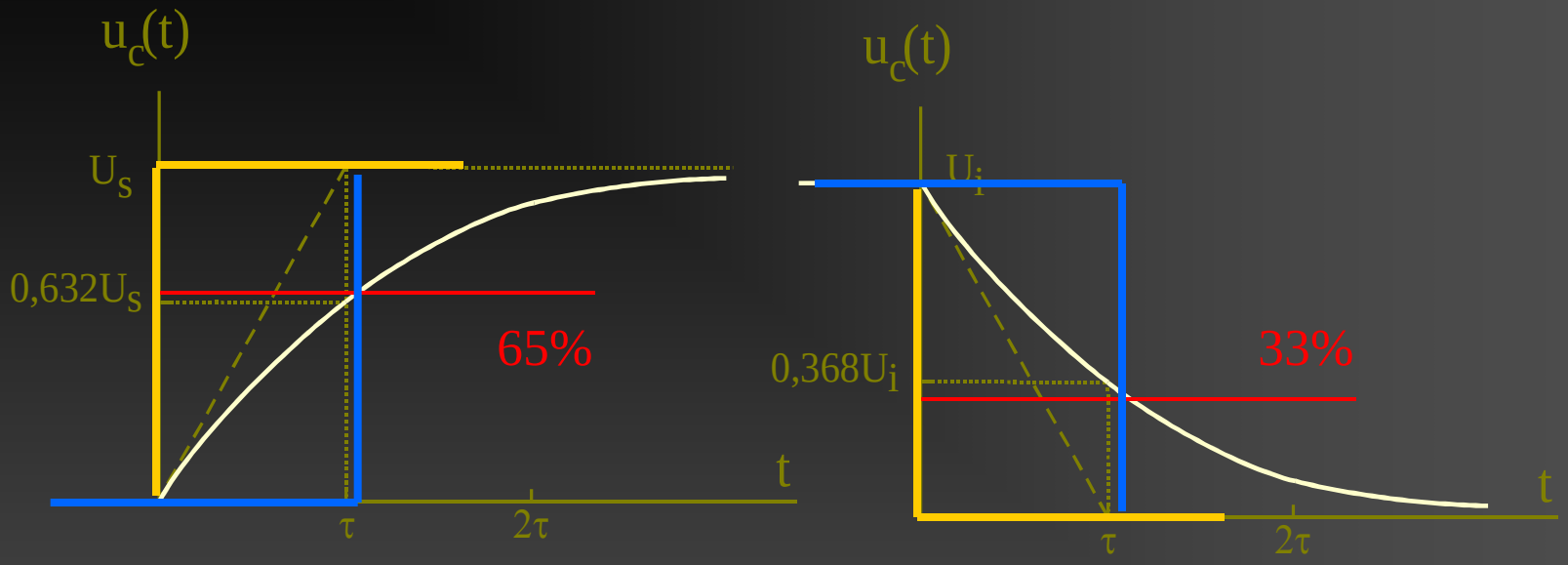
Die RC-Ladekurve





© A. Steininger / TU Wien

Verzögerung durch RC-Konstante

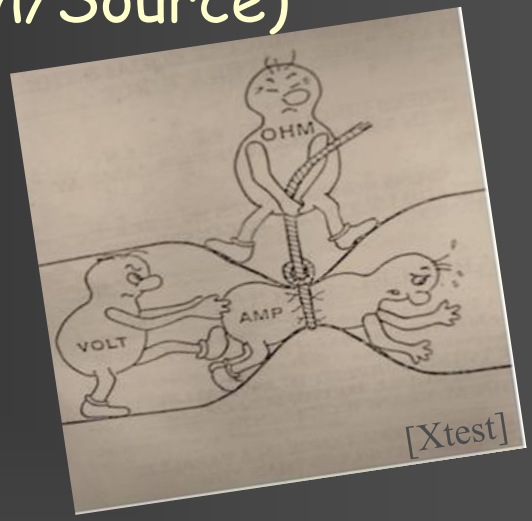


Unter der Annahme von Schaltschwellen bei 65% (1) und 33% (0) beträgt die Verzögerung fast genau eine Zeitkonstante

© A. Steininger / TU Wien

Woher kommen R und C ?

- Widerstand (R)
  - des „eingeschalteten“ FET (Drain/Source)
  - der Verbindungen (Al, Cu)
  - der Durchkontaktierungen
  - der Programmierelemente



- Kapazität (C)

  leitende Teile haben gegeneinander eine Kapazität
  - Strukturen eines bzw. unterschiedlicher FETs
  - Leitungen

© A. Steininger / TU Wien

Grenzen der Geschwindigkeit

- ✓ Wellenausbreitu
  - Information k sich niemals schneller als mit Lichtgesch igkeit au eiten.
- ✓ Ladevorgänge
  - Das Laden von k itäten mit begrenztem Strom bear nt Zeit.
- Bewegung der Ladungs er
  - Bewegung/Diffusio adungsträgern im Halbleiter erfolgt n begrenzter Geschwindigkeit. (Sättig t bei Si typ. 0,1 mm/ns)

Naturgesetz

unvermeidbar

unvermeidbar

© A. Steininger / TU Wien

Grenzen der Komplexität

- Aufwand für Verifikation & Test
  formale Methoden, Design for Test, Built-in Self-Test
- „Design Crisis"
  Produktivität wächst nicht gemäß Moore's Law, Teamgröße begrenzt => Design-Reuse, IP-Module, ASIP, bessere Tools
- Ausbeute (yield)
  on-chip repair
- Anzahl der Pins pro Gehäuse
  System on chip
- Leistungsverbrauch
  Power-Management

© A. Steininger / TU Wien

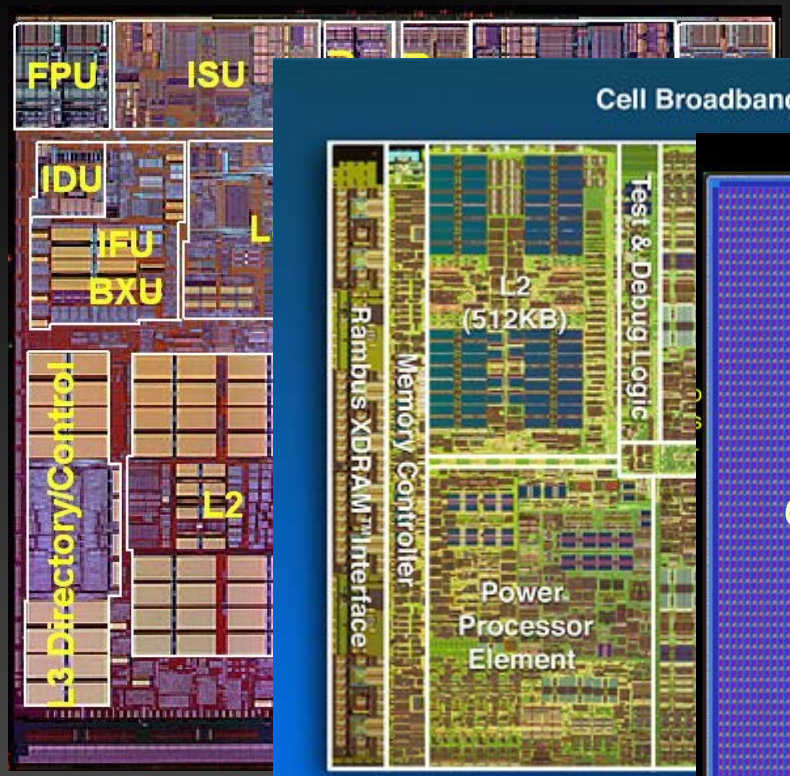
Multiprocessor Chips



- Multiprocessing längst üblich im High-end Bereich
- Vervielfachung eines Designs löst Design Crisis
- Performance-Steigerung bei gleichbleibender Taktrate (!)
- 2007: 4 Cores schon im Consumer-Bereich
- Verdopplung alle 1.5 Jahre (Moore) ?
- Optimierung von HW hin zur SW verschoben
- Nutzung dieses Potentials ist gewaltige Herausforderung für Informatiker!

© A. Steininger / TU Wien

Multicore: Beispiele



IBM POWER

IBM Cell (8-core)

IBM True North (Synapse)

© A. Steininger / TU Wien



Zusammenfassung (1)

- Die technologische Entwicklung im Bereich der ASICs ist extrem dynamisch. Der bekannteste Indikator dafür ist das Moore'sche Gesetz: Die Komplexität (Anzahl von Transistoren in einem Design) verdoppelt sich alle 1,5 Jahre.
- Siliziumdioxid ist ein Isolator, polykristallines Silizium ein Leiter, und mittels Dotierung lassen sich mit Silizium auch Schalter (Transistoren) realisieren. Damit ist Silizium der ideale Ausgangsstoff für digitale Logik.

© A. Steininger / TU Wien



Zusammenfassung (2)

- Ausgehend vom Rohstoff Quarz wird über komplexe Fertigungsschritte ein Chip gefertigt:
  - Schmelzvorgänge
  - Dotierung
  - Oxidation
  - Metallisierung
- Mittels Photolithographie werden dabei die gewünschten Strukturen hergestellt. Diese sind über Masken definiert.

© A. Steininger / TU Wien



Zusammenfassung (3)

- Der charakteristische Parameter einer Technologie ist die Feature-Size λ.
- Die einzelnen Transistor-Strukturen werden über den metallischen Interconnect verbunden.
- Der fertige Die wird getestet und in ein Gehäuse gepackt.
- Aktuelle Trends bei der ASIC-Fertigung sind Silicon on Insulator, Multichip-Module, Thorugh-Silicon Via und System on a chip.

© A. Steininger / TU Wien



Zusammenfassung (4)

- Eine Reihe technologischer Grenzen scheint das weitere Wachstum der Entwicklung zu begrenzen. Bisher wurden solche Grenzen jedoch stets überwunden – nicht zuletzt aufgrund der immensen Forschungsaufwände in diesem Bereich.

© A. Steininger / TU Wien



Zusammenfassung (5)

- Die Geschwindigkeit elektronischer Schaltungen ist durch die Geschwindigkeit der Wellen-ausbreitung (ca. 20cm/ns im Medium), durch Ladevorgänge sowie durch die Ladungsträger-geschwindigkeit begrenzt.
- Um Ladevorgänge zu beschleunigen, müssen Widerstände und Kapazitäten klein gehalten werden, damit die Zeitkonstante $\tau = RC$ klein bleibt.

© A. Steininger / TU Wien

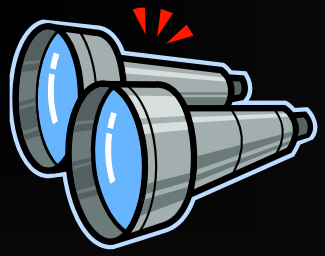
Aufbau logischer Gatter

Vom Feldeffekt-Transistor
zum Supercomputer





© A. Steininger / TU Wien



Überblick

- Was ist CMOS ?
- Feldeffekt-Transistor & CMOS-Prozess
- kombinatorische Logikzellen
- sequenzielle Logikzellen
- weitere Logikfamilien

© A. Steininger / TU Wien

Der Feldeffekt-Transistor

... hat 3 Anschlüsse: Gate, Drain, Source

... funktioniert bei richtiger Auslegung
wie ein Schalter

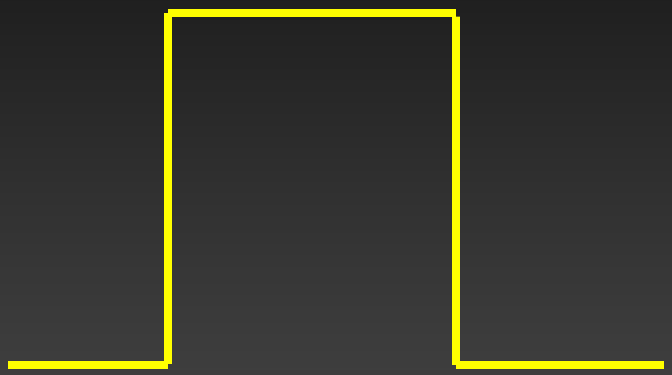
... ist aber bei genauerer Betrachtung
eigentlich ein analoges Bauelement

- analoge Zustandsübergänge (Schaltflanken)
- begrenzte Schaltzeiten
- Einschwingen und Überschwingen, etc.

© A. Steininger / TU Wien

Schaltvorgang: Ideal & Realität

- Idealisierung
- Realität

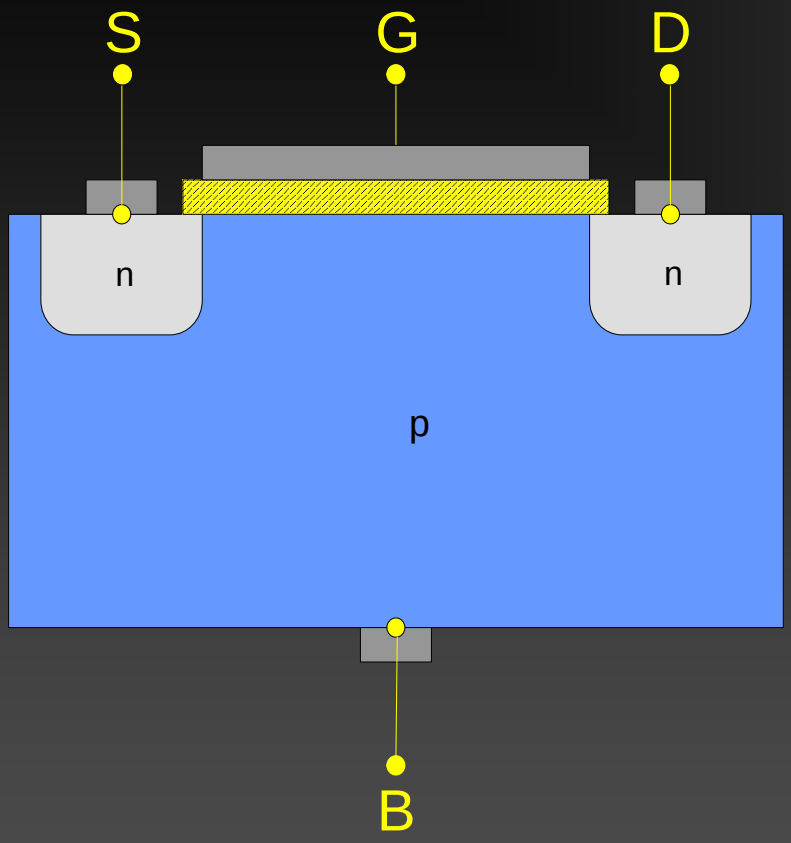


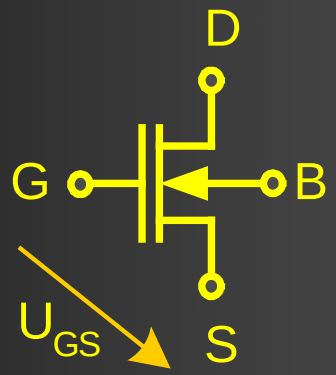




© A. Steininger / TU Wien

n-Kanal Enhancement MOSFET





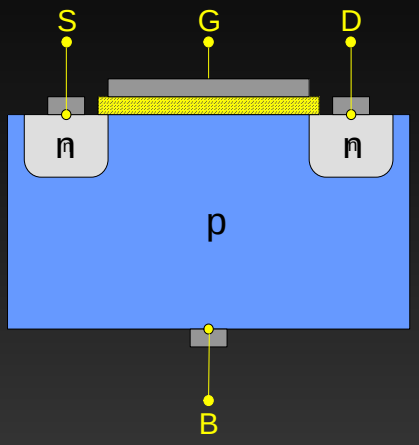


bei $U_{GS} = 0$
kein Stromfluß
=> "selbstsperrend".

bei $U_{GS} > U_{th} > 0$
Stromfluß von D nach S
(U_{th} ... Schwellspannung)

© A. Steininger / TU Wien

Was passiert im FET?



pn-Übergang: Gleichgewicht der Kräfte auf Elektron

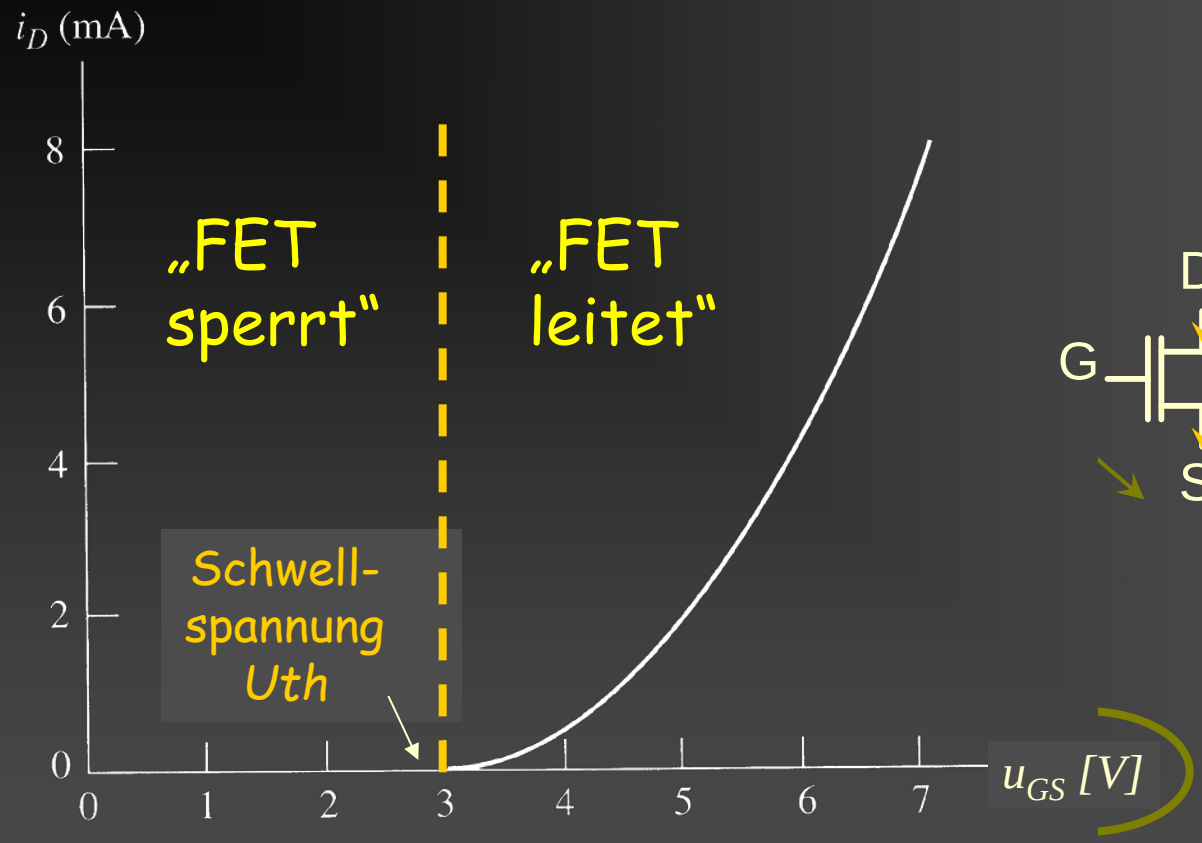
- elektr. Kraft (zum Kern)
- Gitterkraft (zum Loch)

(thermodyn. Vorgänge, stark temperaturabh.!)

- U_{GS} bewirkt E-Feld (= zusätzl. elektr. Kraft auf Elektronen) und verschiebt dadurch Gleichgewicht.
- Bei $U_{GS} = U_{th}$ sind die Löcher im p-Si gefüllt; Elektronen können den Kanal zwischen D und S passieren.

© A. Steininger / TU Wien

n-Kanal FET: Eingangskennlinie



© A. Steininger / TU Wien

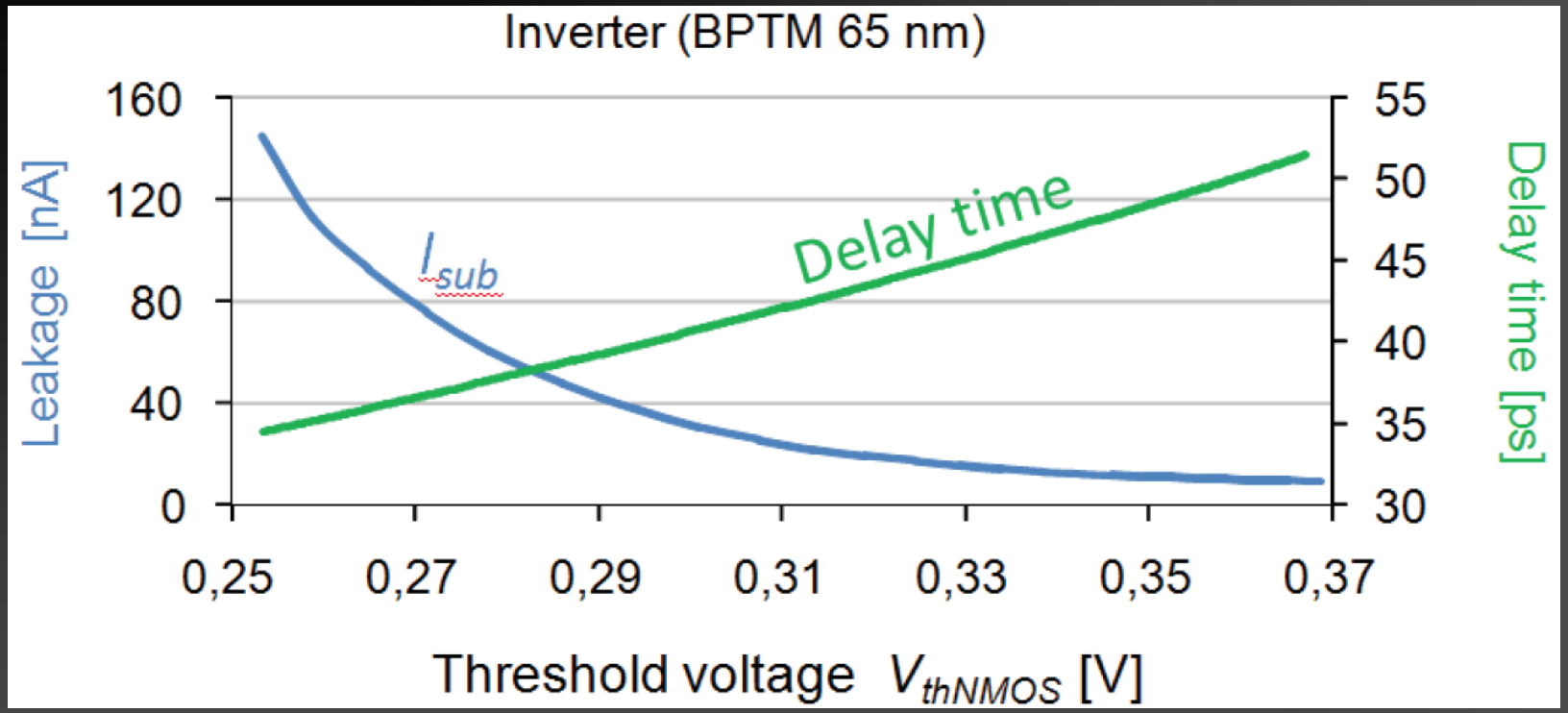
Dimensionierung



- Schwellspannung U_{th}
  - ✓ Grenzwert der Spannung zwischen „Schalter geschlossen“ und „Schalter offen“
  - ➢ Einstellbar über Dotierung

© A. Steininger / TU Wien

Einfluss der Schwellspannung



Trade-off zwischen Verlustleistung und Geschwindigkeit

© A. Steininger / TU Wien

Dimensionierung

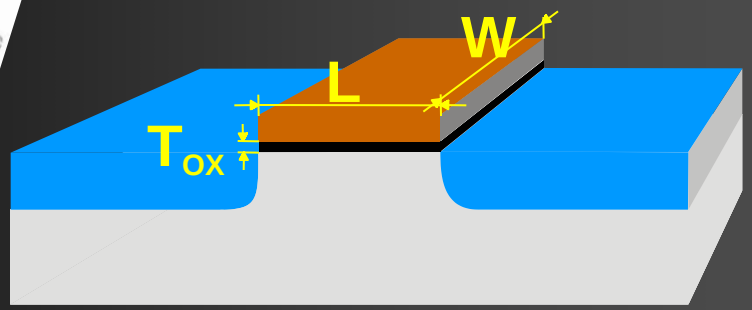


- Schwellspannung U_{th}
  - ✓ Grenzwert der Spannung zwischen „Schalter geschlossen" und „Schalter offen"
  - ➢ Einstellbar über Dotierung
- Ausgangsstrom I_{DSS}
  - ✓ Maximaler Strom, den der FET bei „Schalter geschlossen" führen kann
  - ➢ Einstellbar über Verhältnis von Kanallänge L zu Kanalbreite W: $I_{DSS} \propto W/L$ („Formfaktor")

© A. Steininger / TU Wien

Formfaktor

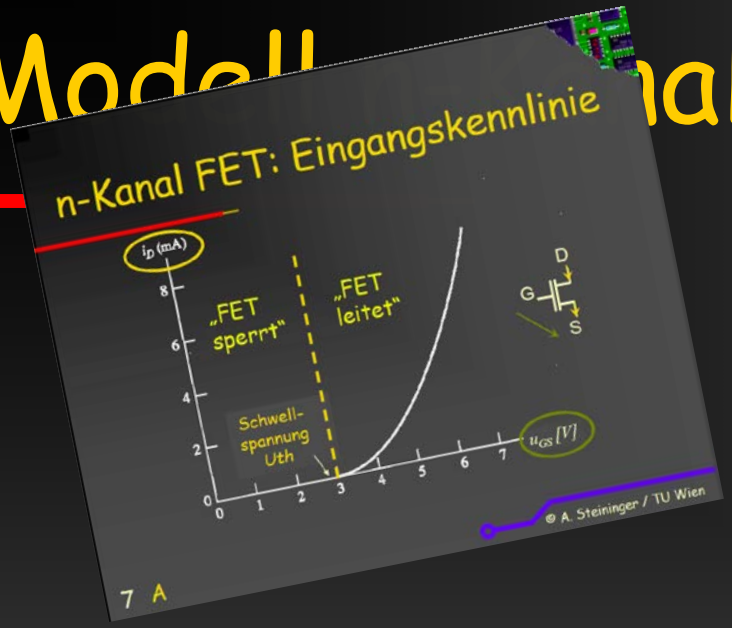


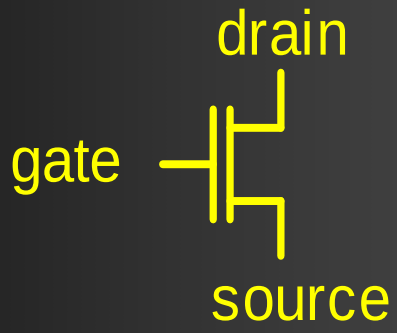


- Quotient aus Kanalbreite W und Kanallänge L
- bestimmt Sättigungsstrom des Schalters
- Erlaubt Einstellen der Treiberstärke
  - höhere Treiberstärke als X1 (X2, X4, X8) (X1 entspricht dem einfachen Inverter)
  - Angleich p-Kanal / n-Kanal (Mobilität d. Löcher schlechter => ca. Faktor 2)
  - Optimierung nach Performance / Fläche

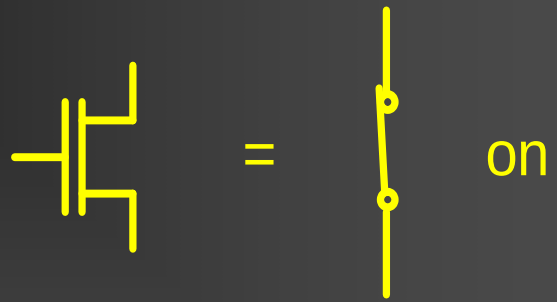
© A. Steininger / TU Wien

Modell ...nal FET

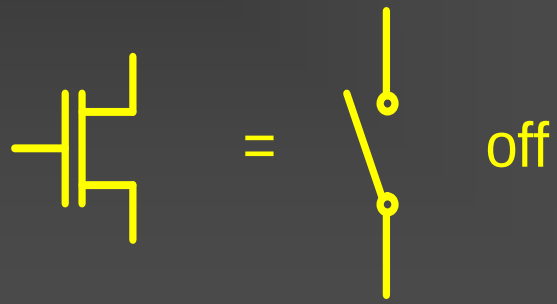




- bei logisch 1 ist der Schalter geschlossen



- bei logisch 0 ist der Schalter offen



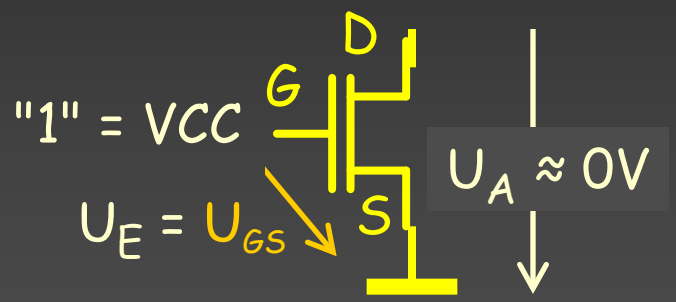
© A. Steininger / TU Wien

FET-Grundschaltung 1

„Sourceschaltung“

VCC

R



Gleichungssystem:

$U_A = VCC - R \cdot I_D$

$I_D = K\,[2(U_{GS}-U_{th})U_{DS}-U_{DS}^2]$

$U_{DS} = U_A$

$U_{GS} = U_E$

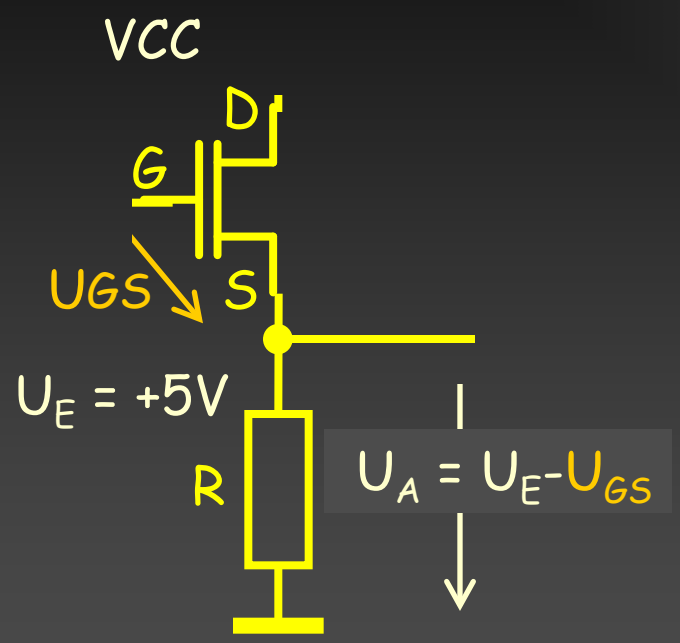
Steuerspannung U_{GS} nur durch Eingangsspannung bestimmt

Spannung wird invertiert

© A. Steininger / TU Wien

FET-Grundschaltung 2

„Sourcefolger“



Gleichungssystem:

$U_A = R \cdot I_D$

$I_D = K (U_{GS} - U_{th})^2$

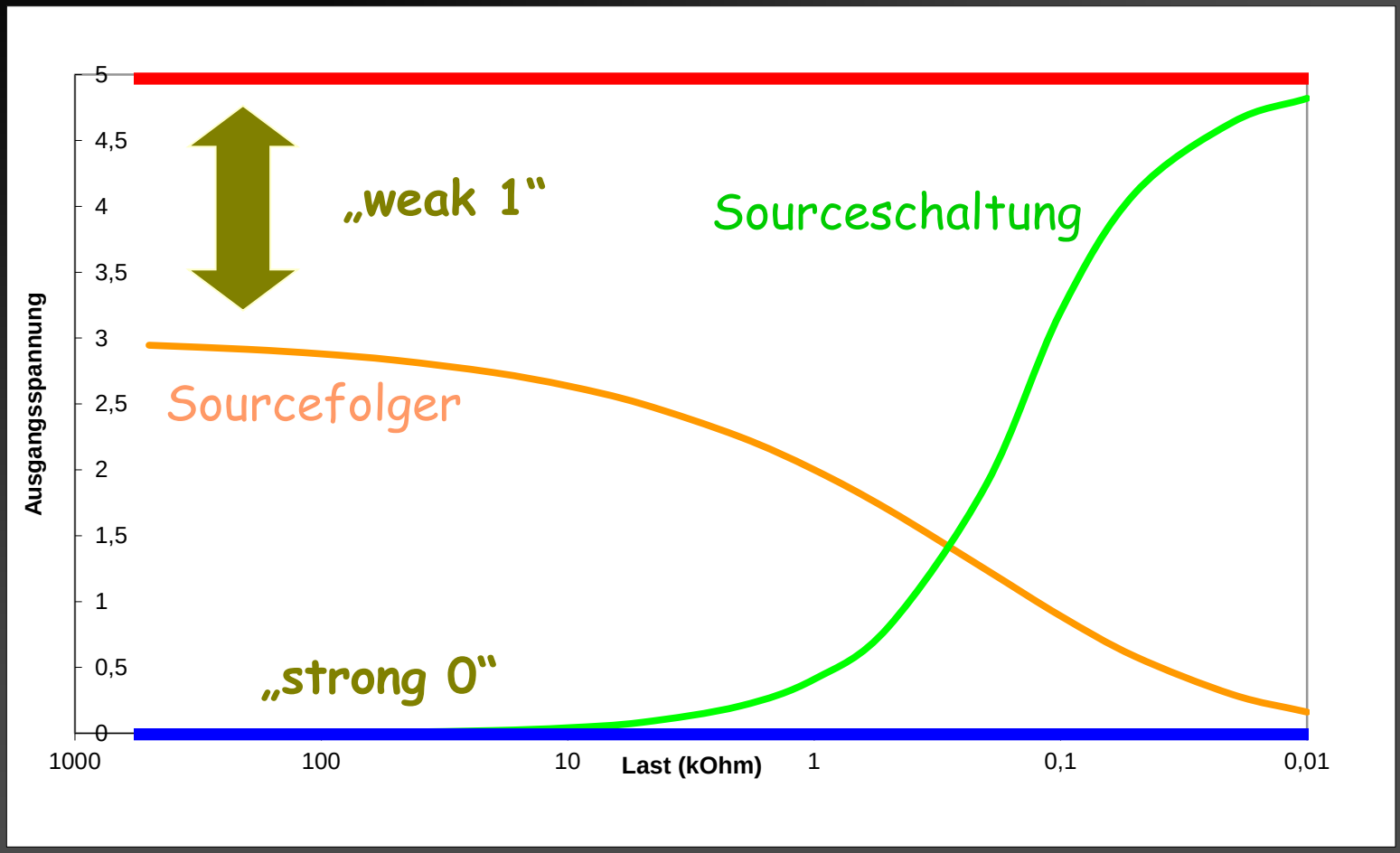
$U_{GS} = U_E - U_A$

Ausgangsspannung U_A vermindert verfügbare Steuerspannung U_{GS} !

Ausgangsspannung U_A ist stets kleiner als Eingangsspannung

© A. Steininger / TU Wien

Vergleich der Schaltungen

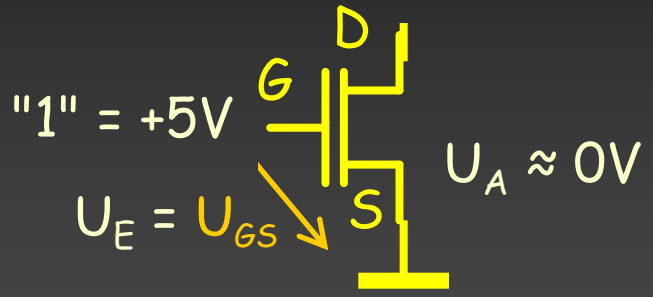


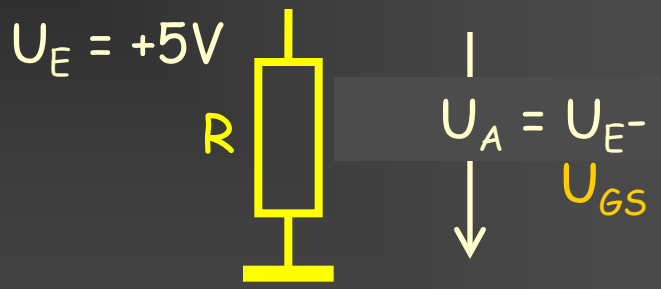
© A. Steininger / TU Wien

Starke und schwache Pegel

+5V

R





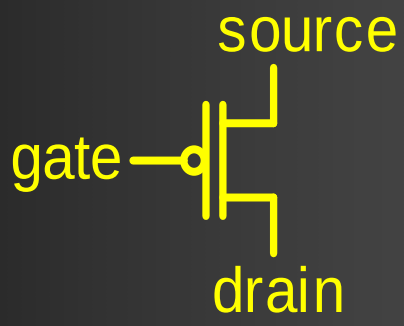
U_{GS} ist nur durch U_E bestimmt, unabh. von U_A

U_{GS} sinkt wenn U_A steigt => FET-Schalter öffnet! „Schalter" abh. v. Ausgang

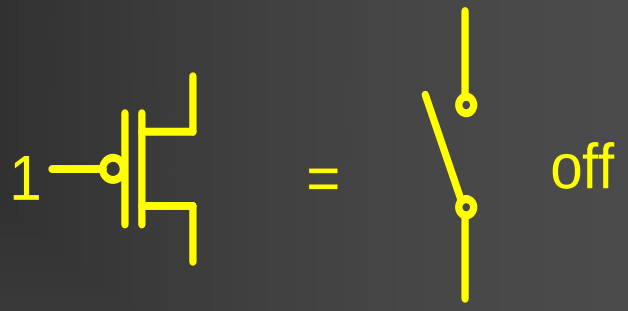
© A. Steininger / TU Wien

Modell p-Kanal FET

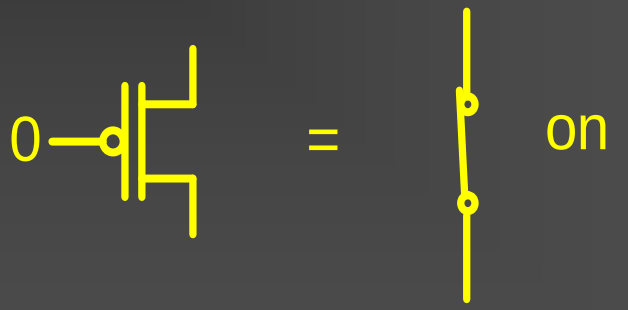
umgekehrt wie bei n-Kanal FET !



- bei logisch 1 ist der Schalter offen



- bei logisch 0 ist der Schalter geschlossen



© A. Steininger / TU Wien

Vorteil „komplementärer“ FETs

- n-Kanal FET kann
  - logisch „0“ aktiv treiben (strong 0),
  - logisch „1“ nur *sehr schwach* (weak 1)
- p-Kanal FET kann
  - logisch „1“ aktiv treiben (strong 1),
  - logisch „0“ nur *sehr schwach* (weak 0)
- Durch Kombination kann man *beide logischen Pegel aktiv* treiben

© A. Steininger / TU Wien

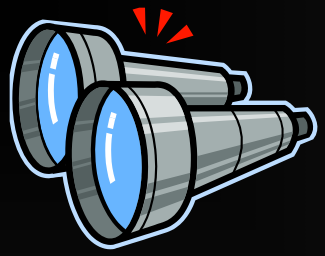
CMOS-Logik

- n-Kanal FET und p-Kanal-FET nennt man zueinander *„komplementär"*.
- es werden **MOS**FET-Transistoren verwendet (**M**etall/**O**xid/**S**emiconductor)

„**C**omplementary **MOS**FET" **CMOS**

- **CMOS treibt beide Logikpegel aktiv.**
- In CMOS lassen sich logische Funktionen besonders effizient implementieren.

© A. Steininger / TU Wien



Überblick

- Was ist CMOS ?
- Feldeffekt-Transistor & CMOS-Prozess
- **kombinatorische Logikzellen**
- sequenzielle Logikzellen
- weitere Logikfamilien

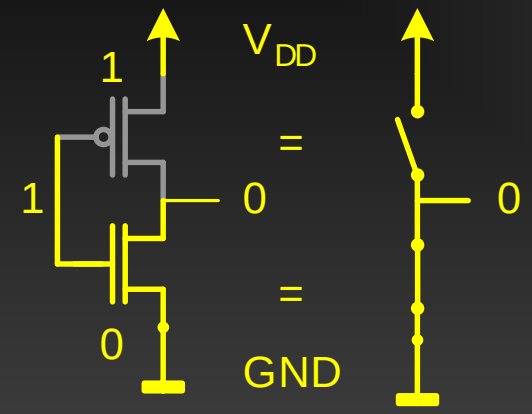
© A. Steininger / TU Wien

Der CMOS-Inverter: Funktion

„1" am Eingang:

p-FET offen

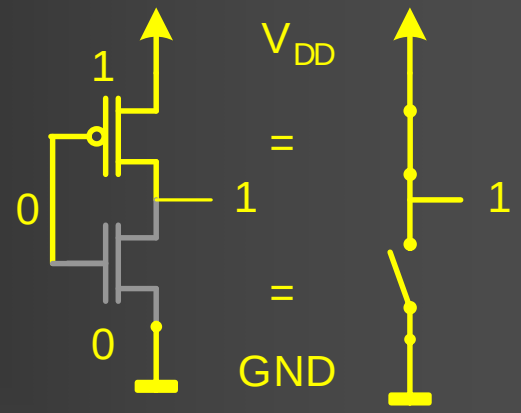
n-FET geschl.

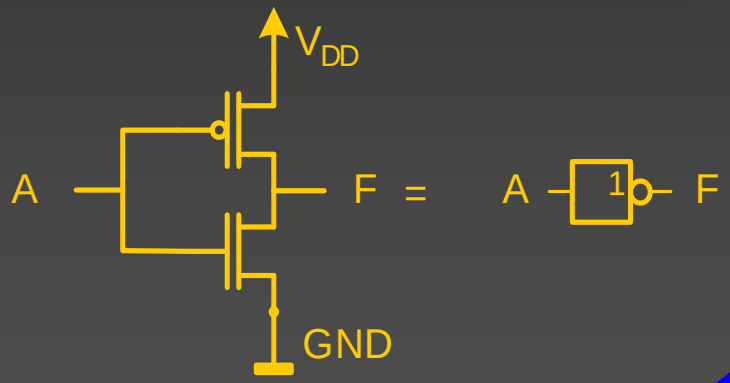


„0" am Eingang:

p-FET geschl.

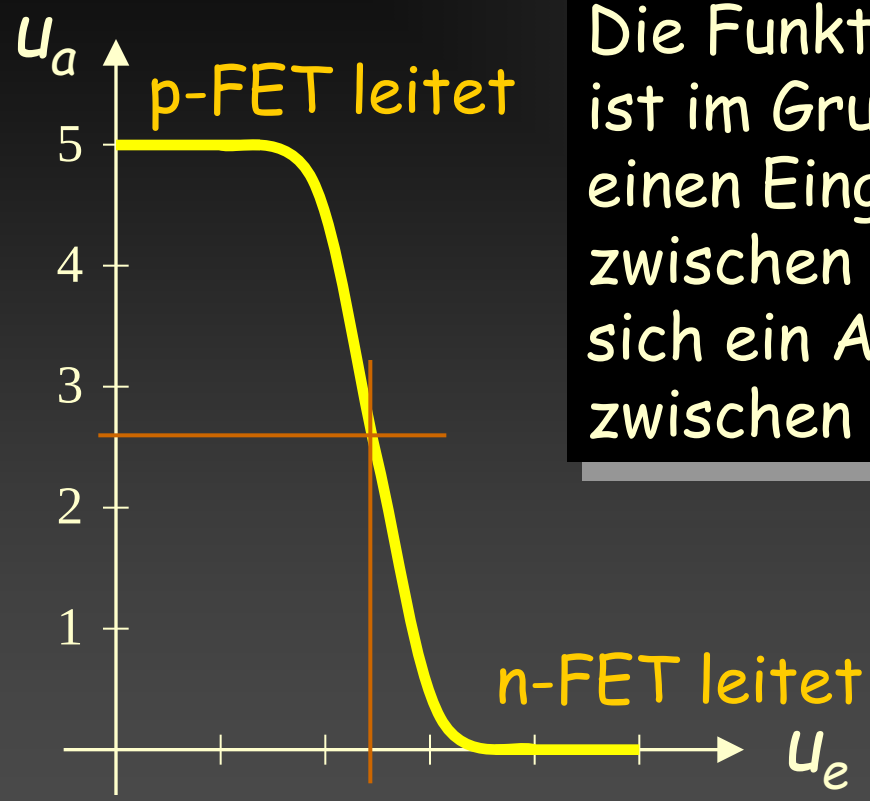
n-FET offen





© A. Steininger / TU Wien

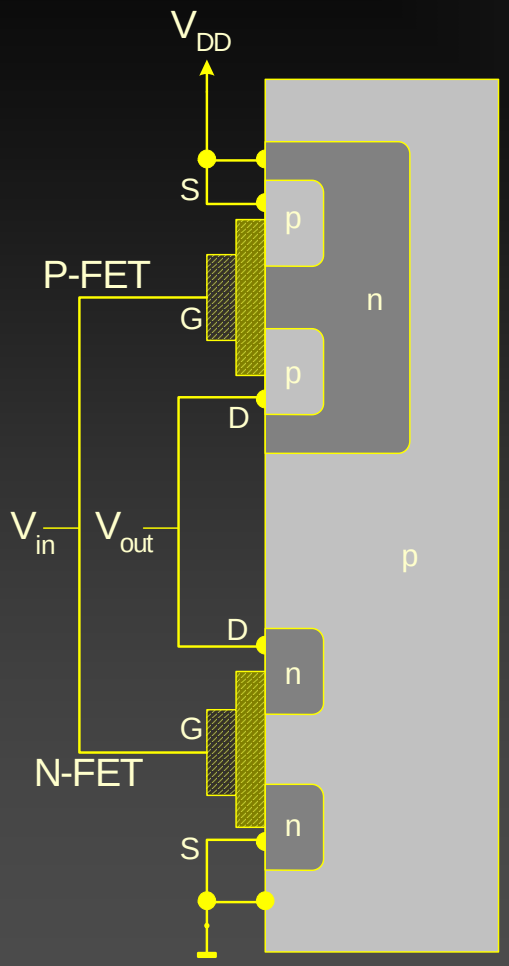
CMOS-Inverter: Kennlinie

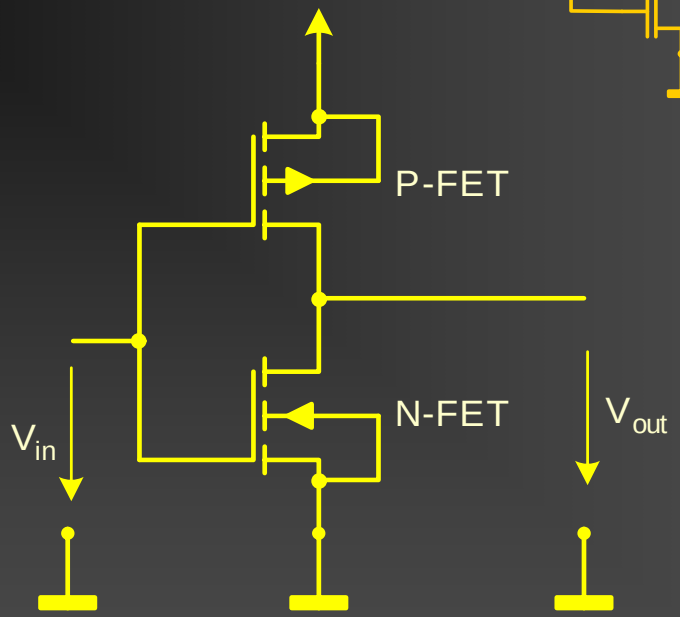


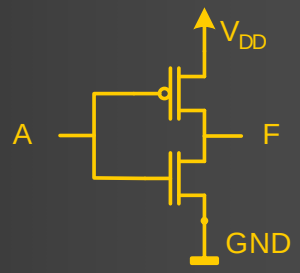
Die Funktion des Inverters ist im Grunde analog: Für einen Eingangspegel zwischen HI und LO kann sich ein Ausgangspegel zwischen LO und HI ergeben

© A. Steininger / TU Wien

CMOS-Inverter: Technologie

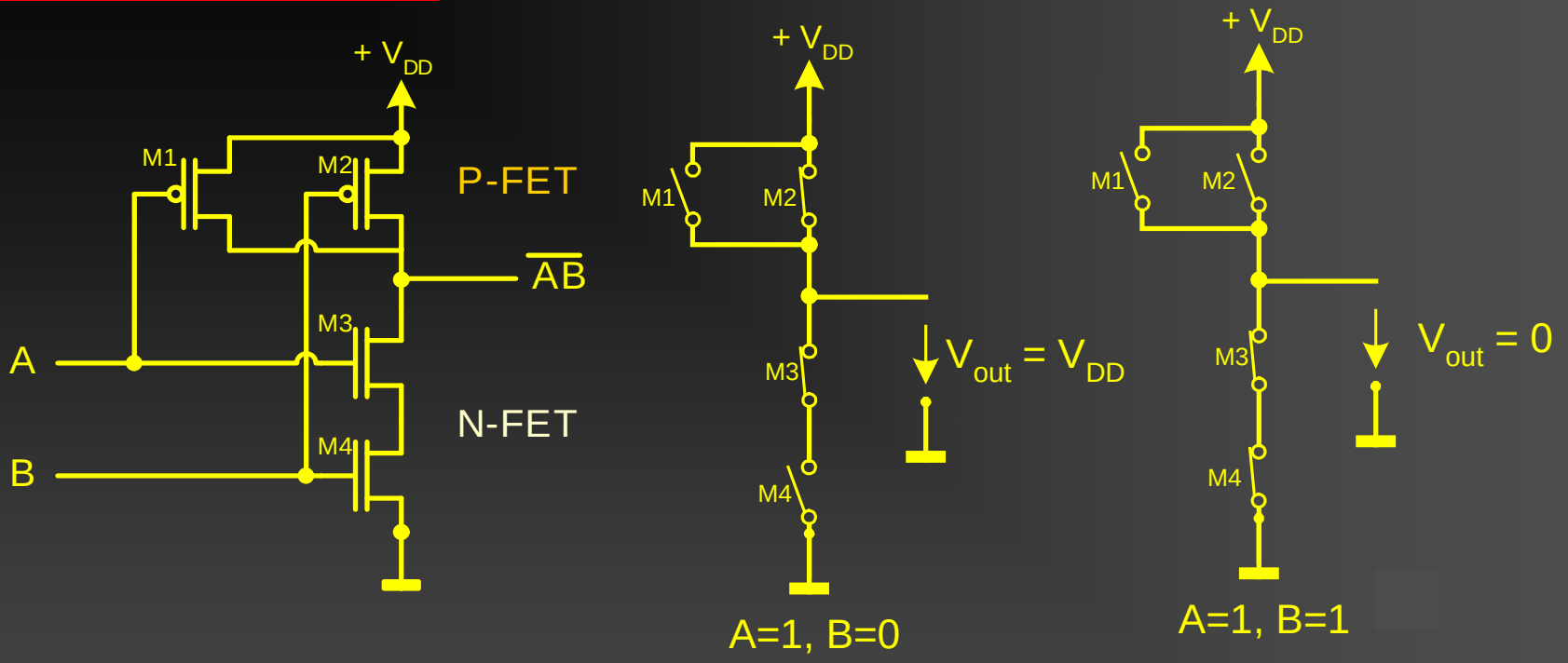






© A. Steininger / TU Wien

Aufbau eines CMOS-NAND2

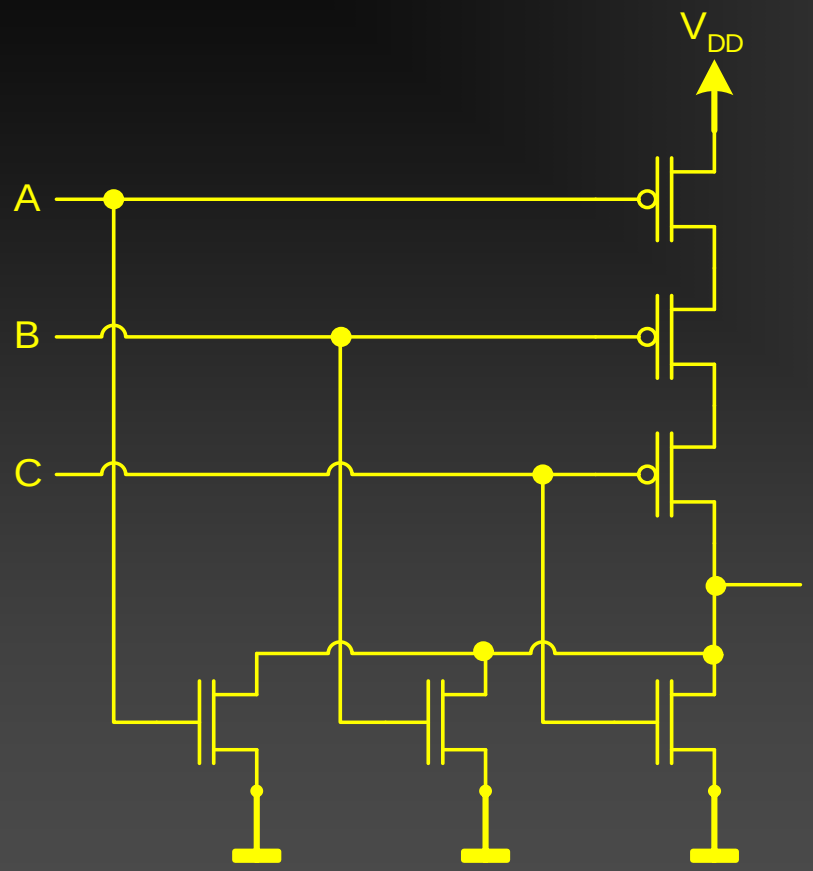


p-FETs parallel ⇨ „0" an A oder B für Y = „1"

n-FETs in Serie ⇨ „1" an A und B für Y = „0"

© A. Steininger / TU Wien

Aufbau eines CMOS-NOR3



p-FETs in Serie
„0" an A, B und C
für Y = „1"

n-FETs parallel
„1" an A, B oder C
für Y = „0"

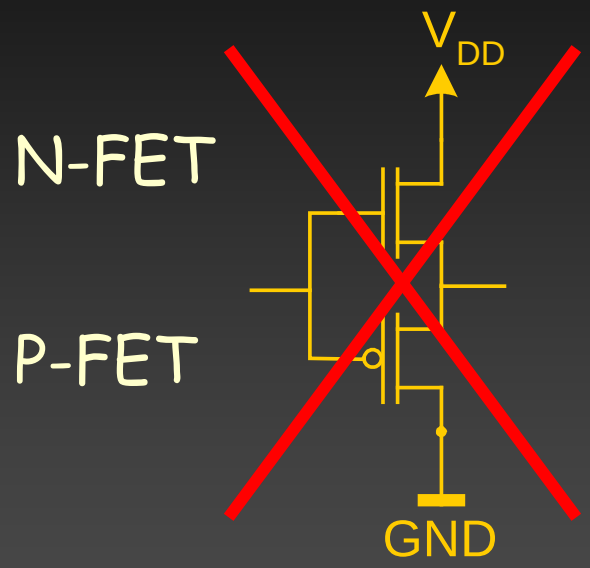
© A. Steininger / TU Wien

CMOS-Buffer



- Falsch

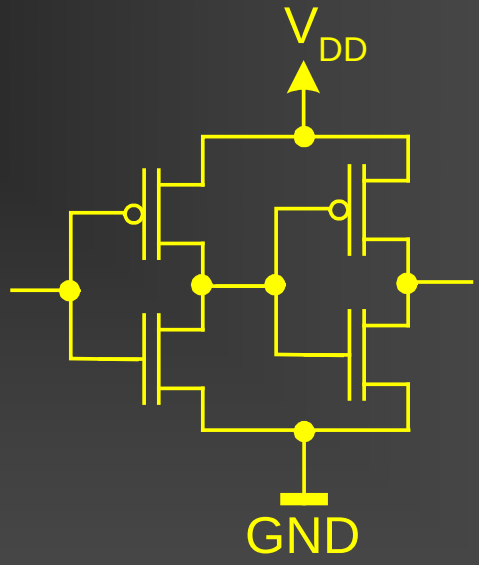
N & P-FET vertauscht



Nur schwache Pegel !

- Richtig

2 Inverter in Serie

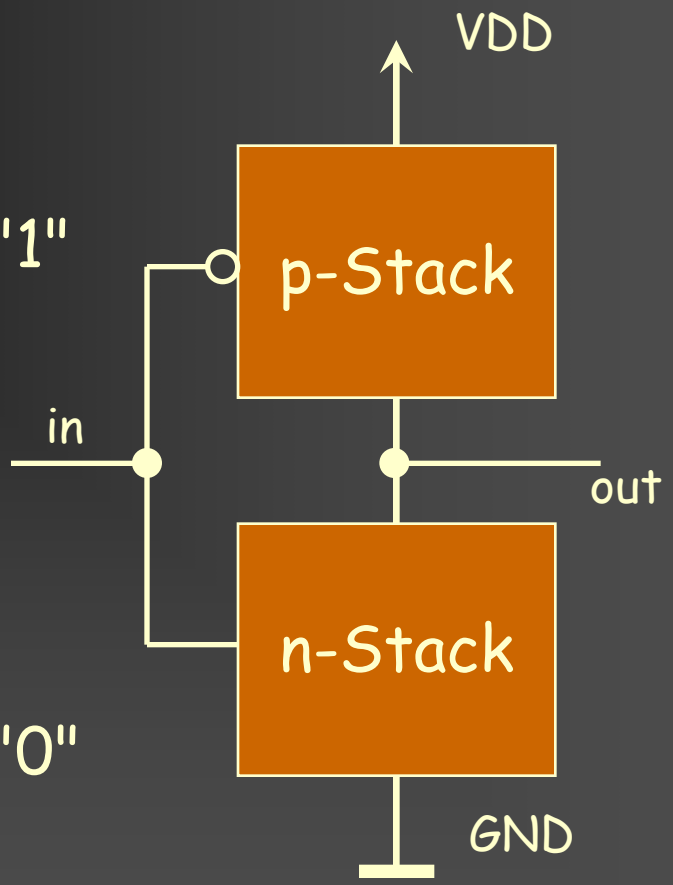


2-stufige Schaltung !

© A. Steininger / TU Wien

CMOS-Gatter: allg. Aufbau

- Der p-Stack
  - wird aus p-FETs gebildet
  - schaltet den Ausgang auf "1"
- Der n-Stack
  - wird aus n-FETs gebildet
  - schaltet den Ausgang auf "0"

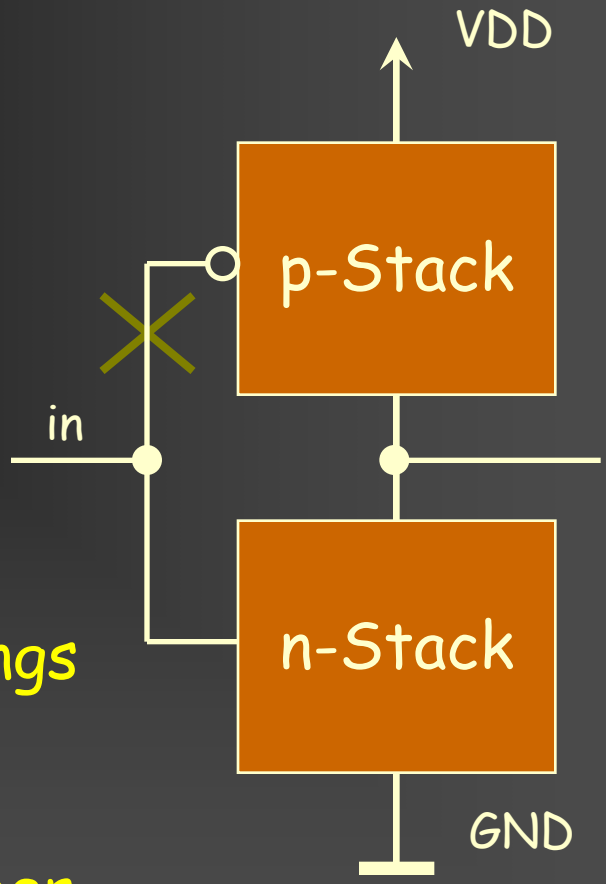


© A. Steininger / TU Wien

Tri-State-Ausgang



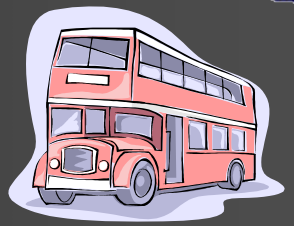
p-Stk	n-Stk	Ausgang
ein	aus	1
aus	ein	0
aus	aus	Tri-state
ein	ein	Kurzschluß



erlaubt Abschalten des Ausgangs über einen Steuereingang „output enable (OE)".
Vorteil: erlaubt mehrere Treiber an einem Bus

© A. Steininger / TU Wien

Tri-State Bus: Probleme



- Bus-Contention:

  auf einer Leitung ist zu einem Zeitpunkt mehr als ein Treiber aktiv => hohe Ströme, Pegel undefiniert

- Floating Bus:

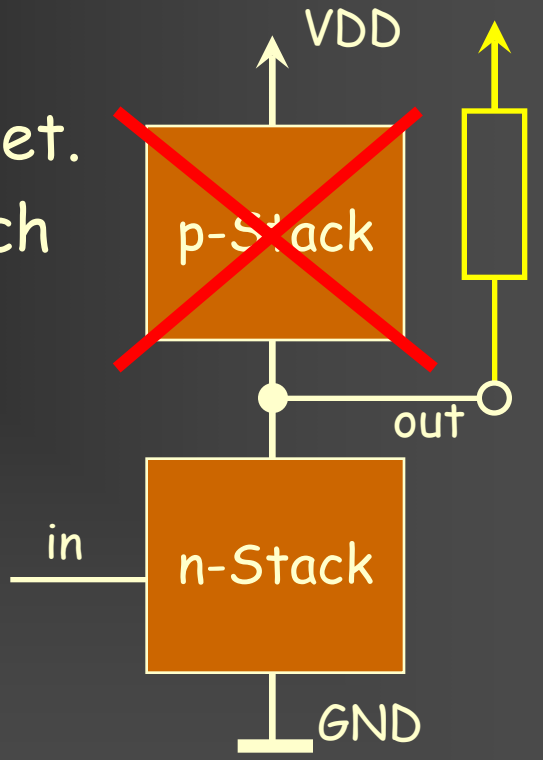
  auf einer Leitung ist kein Treiber aktiv
  => Pegel undefiniert

- Bus-Keeper (bus friendly Logic): FF aus antiparallelen Invertern hält den letzten Zustand, kann aber leicht „overruled" werden (schwache Treiberstärke)

© A. Steininger / TU Wien

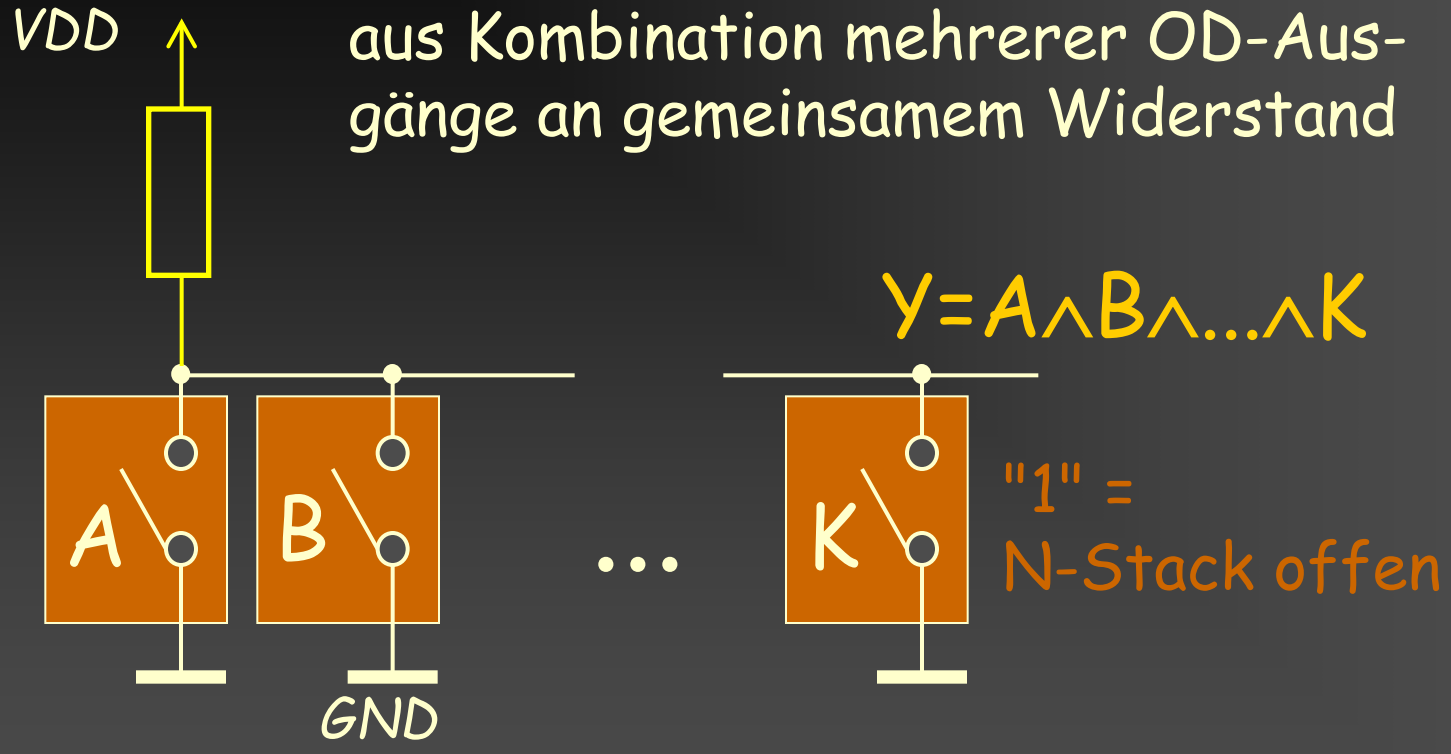
Open-Drain Ausgang (OD)

- Der (aktive) p-Stack wird weggelassen. An seiner Stelle wird extern ein Widerstand verwendet.
- Ausgang "0" wird weiterhin durch den n-Stack erzwungen. Es sind auch größere Ströme zulässig.
- Ausgang "1" wird bei offenem n-Stack durch den Widerstand (schwach) hergestellt: Bei größeren Strömen bricht die Spannung ein.



© A. Steininger / TU Wien

Prinzip des „Wired AND"



aus Kombination mehrerer OD-Ausgänge an gemeinsamem Widerstand

$Y = A \wedge B \wedge \ldots \wedge K$

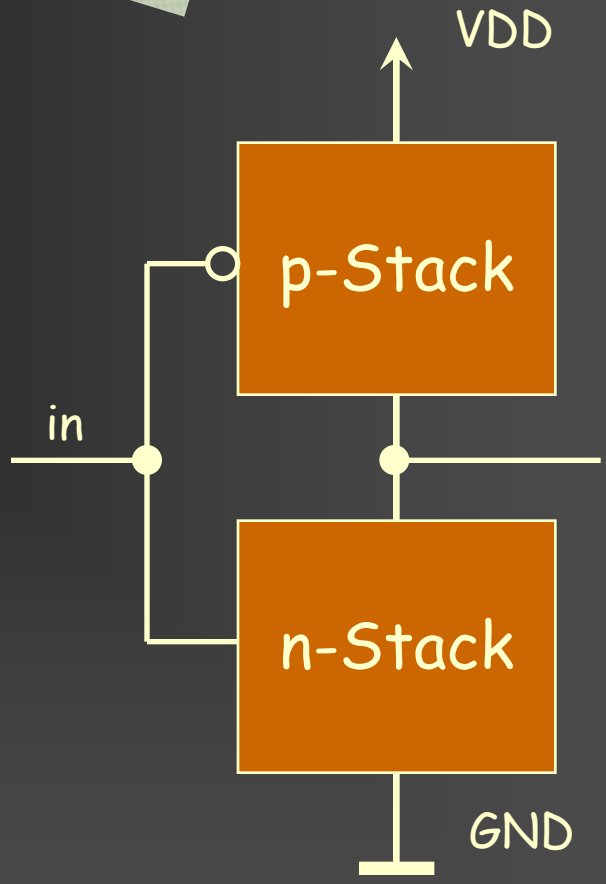
"1" = N-Stack offen

© A. Steininger / TU Wien

Aktiver Ausgang



p-Stk	n-Stk	Ausgang
ein	aus	1
aus	ein	0
aus	aus	Tri-state
ein	ein	Kurzschluß



© A. Steininger / TU Wien

n-Stack: Aufbau



- Ein AND-Term wird durch Serienschaltung von FETs bzw. Blöcken realisiert, ein OR-Term durch Parallelschaltung.
- Durch geeignete Kombination lassen sich beliebige Boolsche Verknüpfungen realisieren, allerdings mit folgenden Einschränkungen:
  - Da der n-Stack genau dann durchschalten soll, wenn die Zielfunktion "0" ist, läßt sich nur eine AND/OR Verknüpfung mit Inversion am Schluß realisieren.
  - Da die n-FETs jeweils bei "1" am Eingang durchschalten, kann man also nicht mit invertierten Eingängen arbeiten.

© A. Steininger / TU Wien

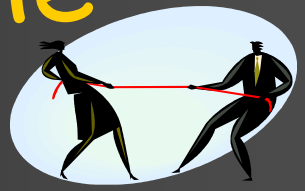
p-Stack: Aufbau



- Ein AND-Term wird auch hier wieder durch Serienschaltung von FETs bzw. Blöcken realisiert, ein OR-Term durch Parallelschaltung.
- Für die Zielfunktion gelten folgende Einschränkungen:
  - Da der p-Stack genau dann durchschalten soll, wenn die Zielfunktion "1" ist, darf die Zielfunktion keine Inversion am Schluß haben.
  - Da die p-FETs jeweils bei "0" am Eingang durchschalten, kann man also nur mit invertierten Eingängen arbeiten.

© A. Steininger / TU Wien

Lösung der Widersprüche



- n-Stack
  - Inversion am Ende
  - nicht-invertierte Eingänge

De Morgan

$$\neg F(X1, X2, X3, \ldots Xn, \wedge, \vee)$$
$$=$$
$$F(\neg X1, \neg X2, \neg X3, \ldots \neg Xn, \vee, \wedge)$$

- p-Stack
  - keine Inversion am Ende
  - nur invertierte Eingänge

© A. Steininger / TU Wien

Entwurfsregeln im Überblick

- Durch Kombination aus Serien- und Parallelschaltung lassen sich auch komplexere Funktionen als NAND und NOR realisieren:

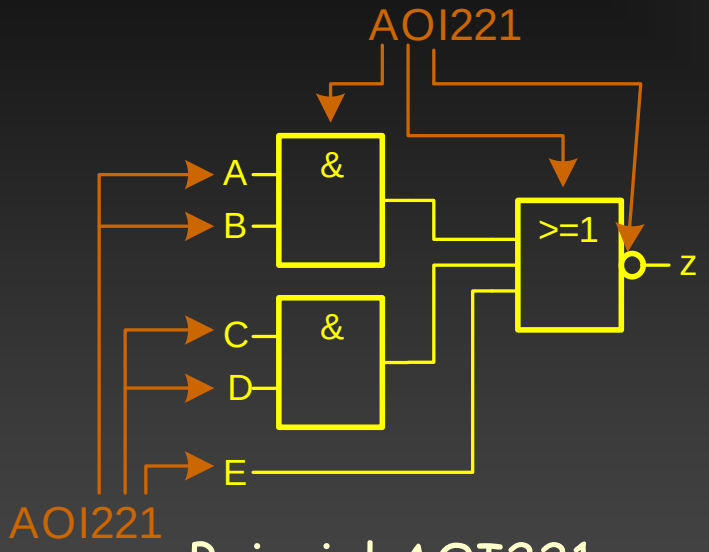
  AND-OR-Invert bzw. OR-AND-Invert,

- In jedem Fall mit Inversion am Ausgang (wenn nötig extra Inverter nachschalten).
- In jedem Fall nicht invertierte Eingänge (wenn nötig extra Inverter vorschalten).
- In jedem Fall p-Stack oben und n-Stack unten.
- In jedem Fall p-Stack dual zu n-Stack.

© A. Steininger / TU Wien

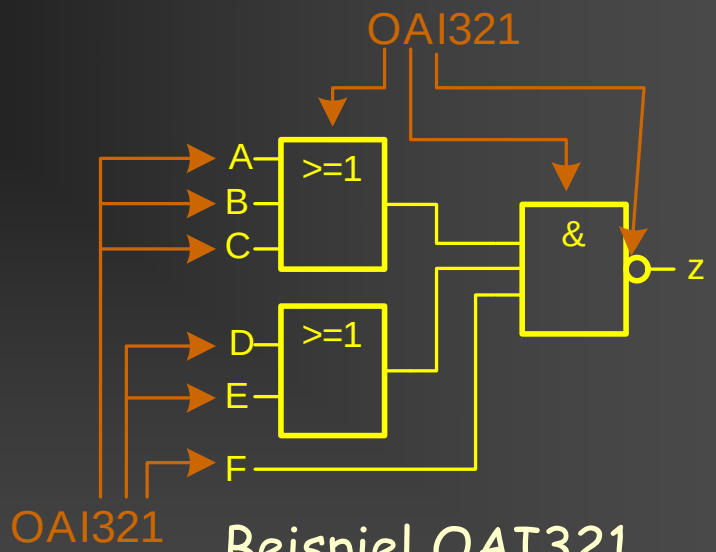
AOI und OAI: Terminologie

AND-OR-Invert



Beispiel AOI221

OR-AND-Invert



Beispiel OAI321

AOI- und OAI-Zellen sind sehr effizient durch Serien/Parallelschaltung von FETs realisierbar

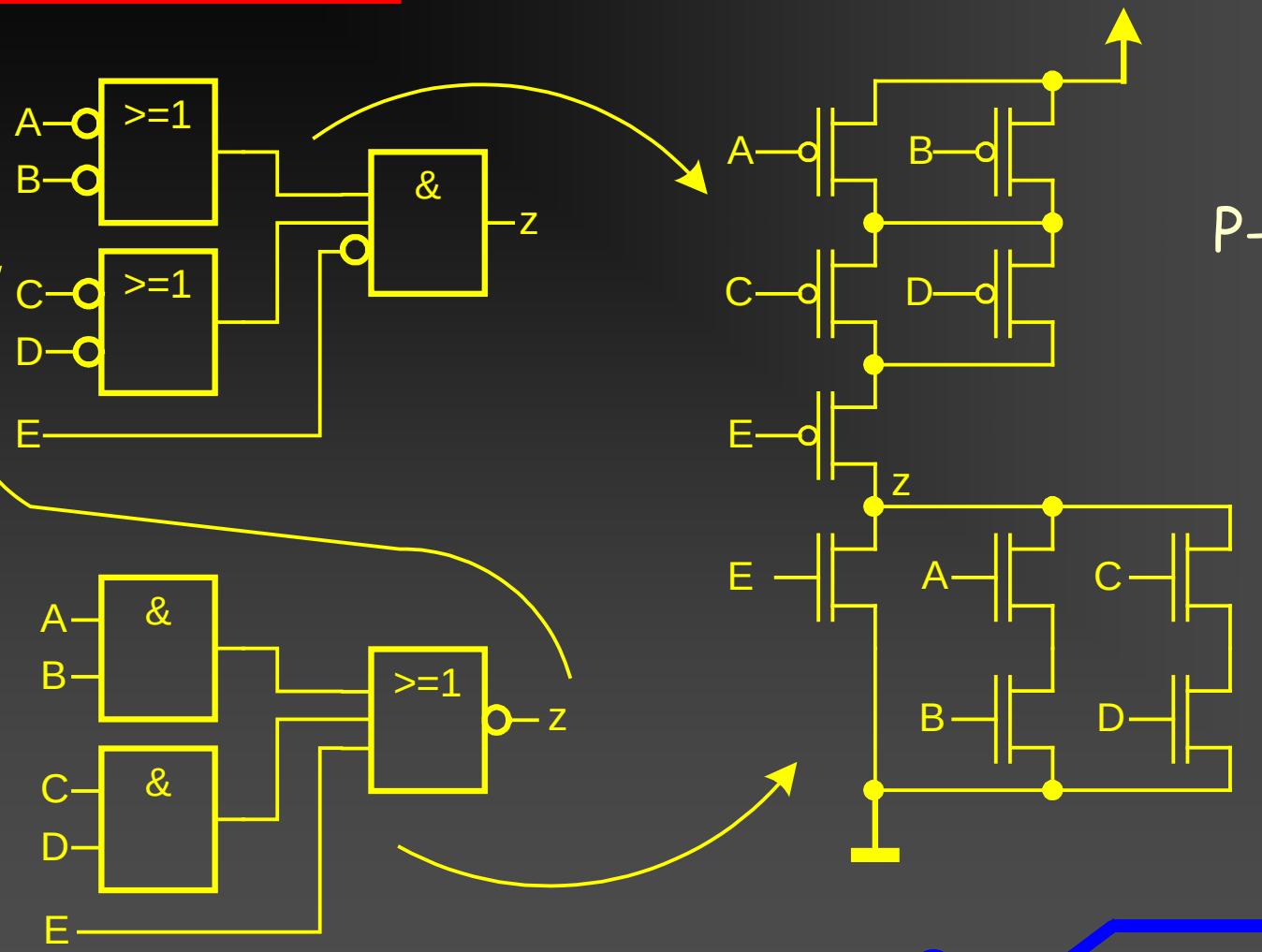
© A. Steininger / TU Wien

Entwurfsregeln für AOI & OAI

1. Gleichung G entsprechend der Funktion aufstellen (Inversion am Schluß!)
   - ➡ n-Stack (strong „0“)
     Inversion am Ausgang erfolgt automatisch
2. Inversion zu Eingängen transformieren: (De Morgan) => Gleichung G*
   - ➡ p-Stack (strong „1“)
     Inversion an d. Eingängen erfolgt automatisch wegen p-Kanal-FET
3. In beiden Fällen gilt: AND = Serienschaltung
   OR = Parallelschaltung

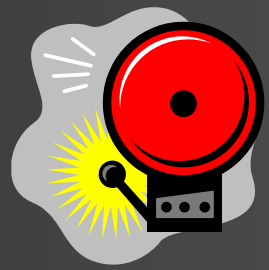
© A. Steininger / TU Wien

Entwurfsbeispiel AOI221



© A. Steininger / TU Wien

Rechenbeispiel



- Alarmanlage:
  - 1 Innenkreis mit 1 Bewegungsmelder B, aktivierbar über Schalter S1
  - 1 Außenkreis mit 2 Türkontakten K1 und K2, aktivierbar über Schalter S2
  - Alle Schalter und Kontakte low-aktiv
  - Auslösung d. Sirene über Signal AL (high-aktiv)
- Gesucht: Realisierung als AOI oder OAI

© A. Steininger / TU Wien

Umformungen

$AL = (\neg S1 \wedge \neg B) \vee (\neg S2 \wedge (\neg K1 \vee \neg K2))$

- AOI:

$AL = (\neg S1 \wedge \neg B) \vee (\neg S2 \wedge \neg K1) \vee (\neg S2 \wedge \neg K2)$

  - nicht invertierend => Inverter am Ausg.
  - invertierte Eingänge => Inverter an allen Eing.

- OAI:

$\neg AL = (S1 \vee B) \wedge (S2 \vee K1) \wedge (S2 \vee K2)$

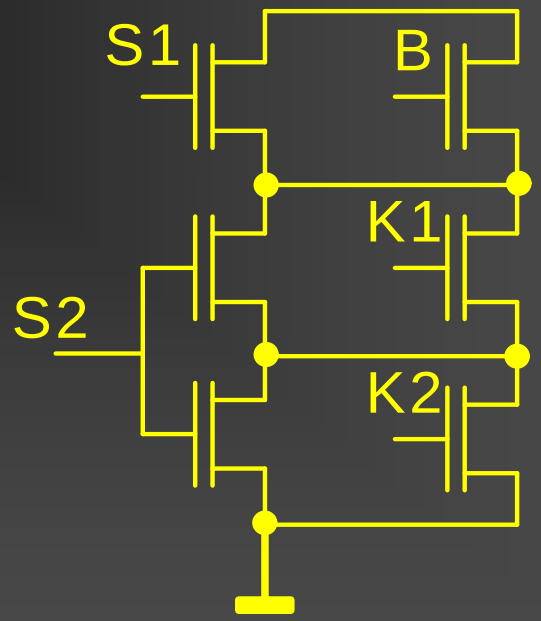
  - Inversionen bereits an den richtigen Stellen
    => viel günstiger zu realisieren

© A. Steininger / TU Wien

Alarmanlage als OAI: n-Stack

$\neg AL = (S1 \vee B) \wedge (S2 \vee K1) \wedge (S2 \vee K2)$

- n-Stack:
  - S1 parallel B
  - S2 parallel K1
  - S2 parallel K2
  - alle Parallel-elemente in Serie



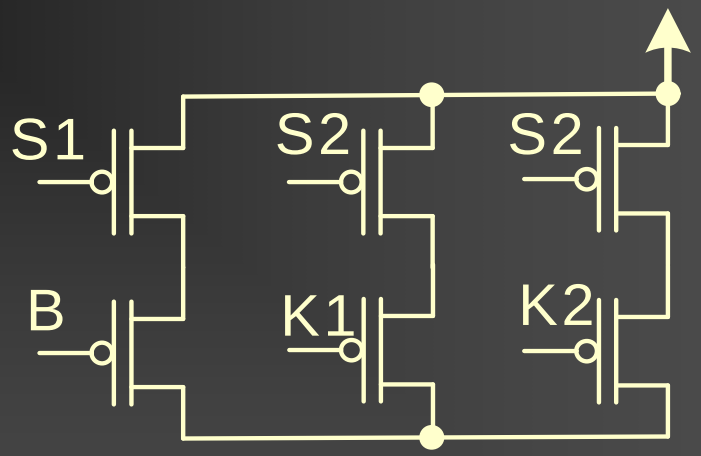
© A. Steininger / TU Wien

Alarmanlage als OAI: p-Stack

$$\neg AL = (S1 \vee B) \wedge (S2 \vee K1) \wedge (S2 \vee K2)$$

$$AL = (\neg S1 \wedge \neg B) \vee (\neg S2 \wedge \neg K1) \vee (\neg S2 \wedge \neg K2)$$

- p-Stack:
  - S1 in Serie mit B
  - S2 in Serie mit K1
  - S2 in Serie mit K2
  - Alle Serien-elemente parallel



© A. Steininger / TU Wien

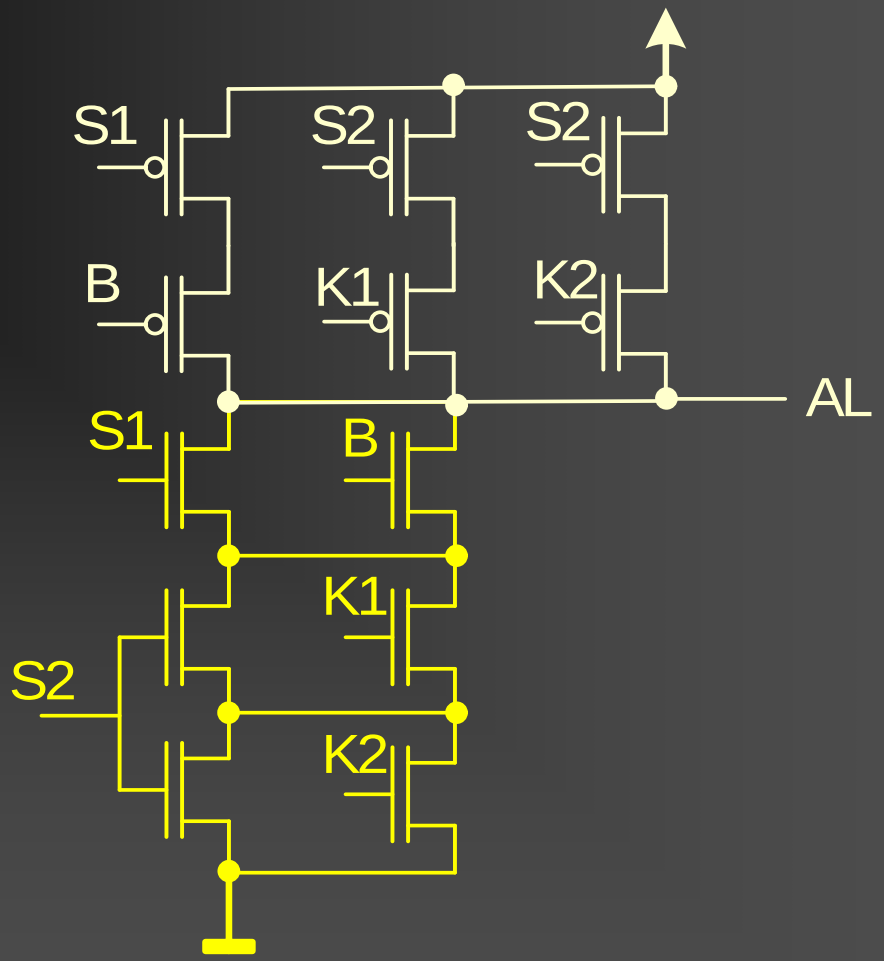
Alarmanlage als OAI: Lösung

- p-Stack:

S1 + B, S2 + K1, S2 + K2, alle parallel

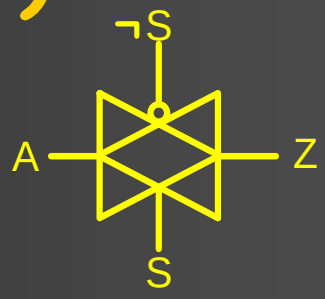
- n-Stack:

S1 par B, S2 par K1, S2 par K2, alle in Serie

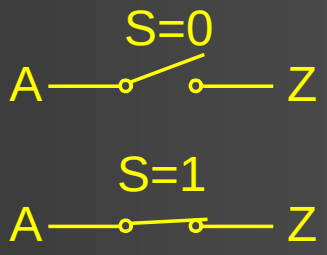


© A. Steininger / TU Wien

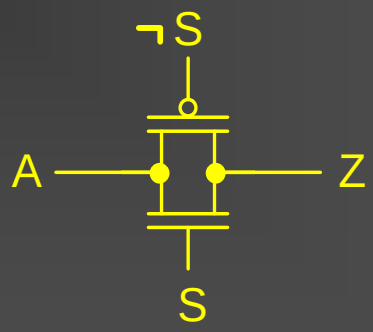
Transmission-Gate (TG)



- **Funktion:**
  Schaltbare Verbindung zwischen zwei Leitungen („offen" = echte Trennung, keine Maskierung)

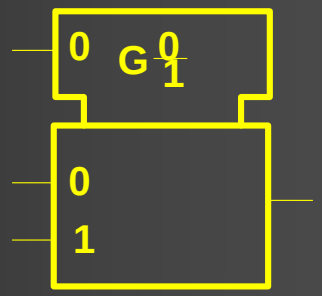


- **Realisierung:**
  n-Kanal FET und p-Kanal FET parallel
  (strong ´1´ und strong ´0´!)



© A. Steininger / TU Wien

Multiplexer (Mux)

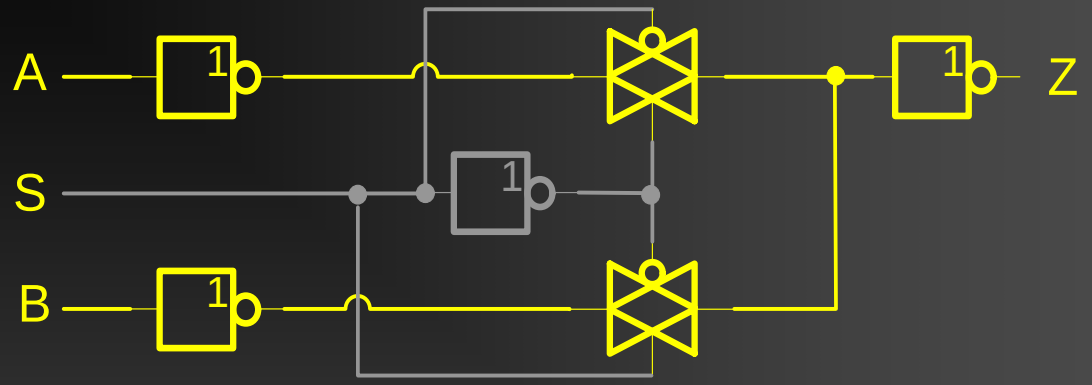


- Funktion:
  Von mehreren (*n*) Eingangssignalen wird über einen Steuereingang eines ausgewählt und an den Ausgang durchgeschaltet.
- Realisierung:
  - eines von *n* Transmission Gates wird aktiviert
  - Kombinatorische Verknüpfung:
    2:1 Mux als OAI22 + Inverter
    4:1 Mux als OAI3333 + Inverter

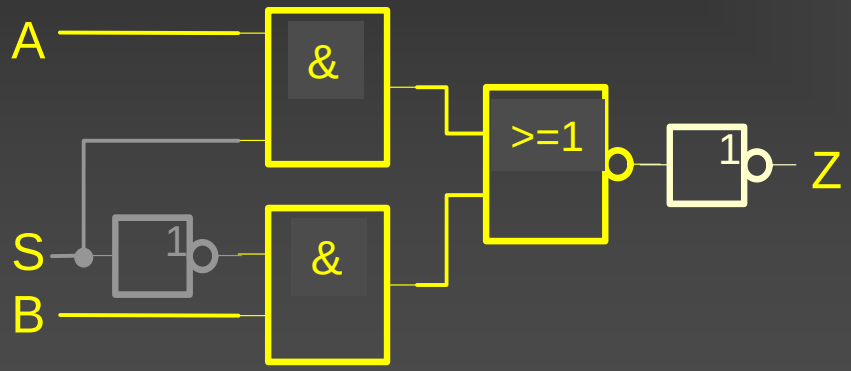
© A. Steininger / TU Wien

Multiplexer-Realisierungen

- TG
  (3GE)



- AOI
  (3GE)



© A. Steininger / TU Wien

Exklusiv-ODER (XOR)

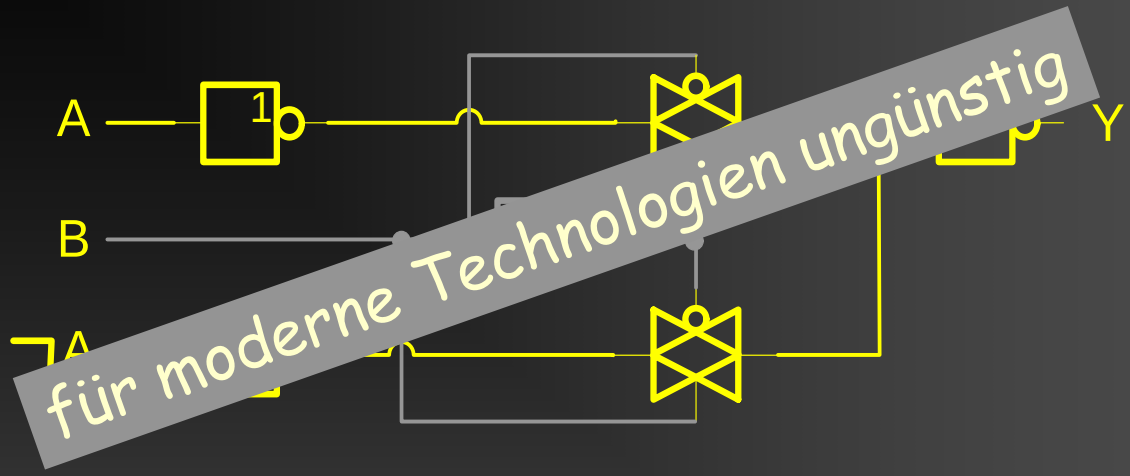
- Funktion:
  logische Verknüpfung $Y = (A \wedge \neg B) \vee (\neg A \wedge B)$

- Realisierung:
  - Multiplexer:
    B am Select-Eingang wählt zwischen A und $\neg A$
  - Kombinatorische Verknüpfung: AOI21 + NOR
    $\neg Y = (A \wedge B) \vee (\neg(A \vee B))$

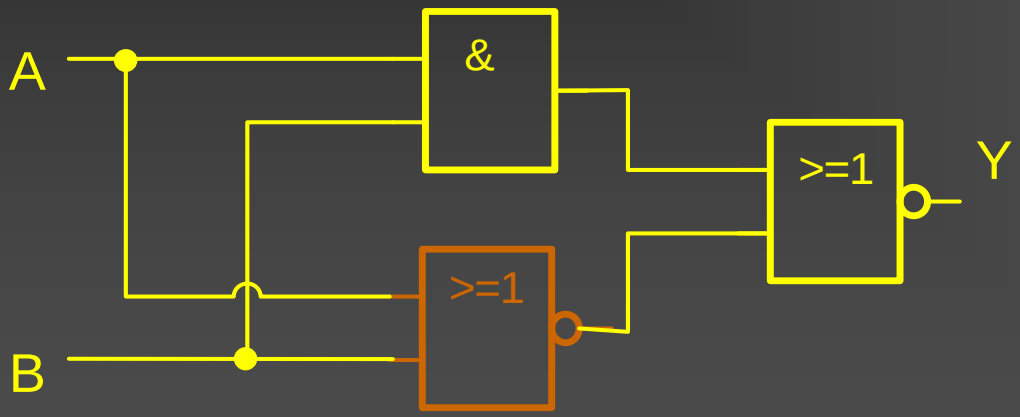
© A. Steininger / TU Wien

XOR-Realisierungen

- TG
  (3GE)

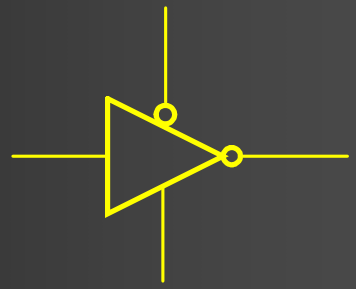


- AOI
  (2.5GE)



© A. Steininger / TU Wien

Getakteter Inverter



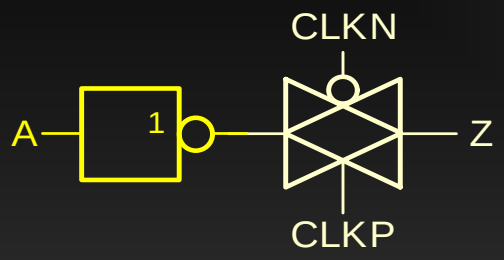
- Funktion:

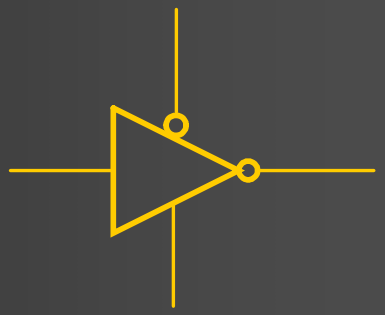
  Wie Transmission Gate, aber
  - Signal wird invertiert
  - Takt als Steuersignal (S)
- Realisierung:

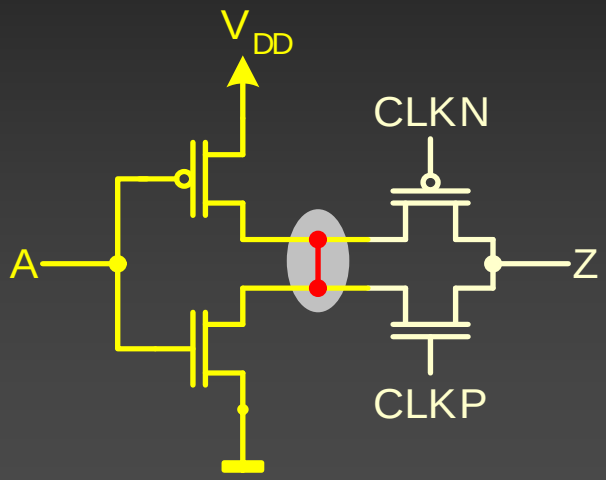
  Serienschaltung Inverter + Transmission Gate
  Dabei läßt sich eine Verbindung einsparen (siehe nächste Folie)
- Anwendung: bei Latch und Flip-Flop

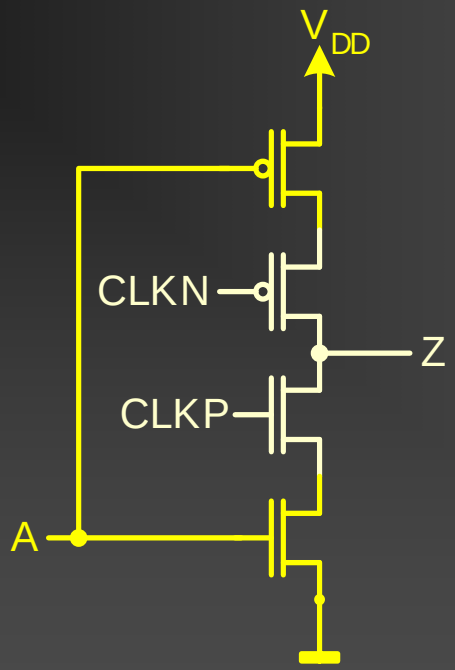
© A. Steininger / TU Wien

Getakteter Inv.: Realisierung

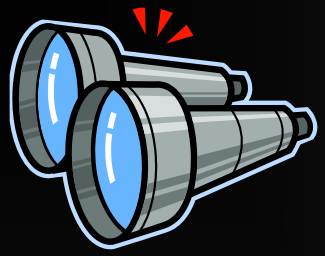








© A. Steininger / TU Wien



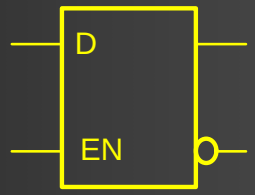
Überblick

- Was ist CMOS ?
- Feldeffekt-Transistor & CMOS-Prozess
- kombinatorische Logikzellen
- **sequenzielle Logikzellen**
- weitere Logikfamilien

© A. Steininger / TU Wien

Latch





- **Funktion:** *(positive enable)*

  Eingänge D (Data) und EN (Enable), Ausgang Q
  - Transparent: D wird direkt auf Q abgebildet
  - Hold: letzter Zustand von Q wird eingefroren

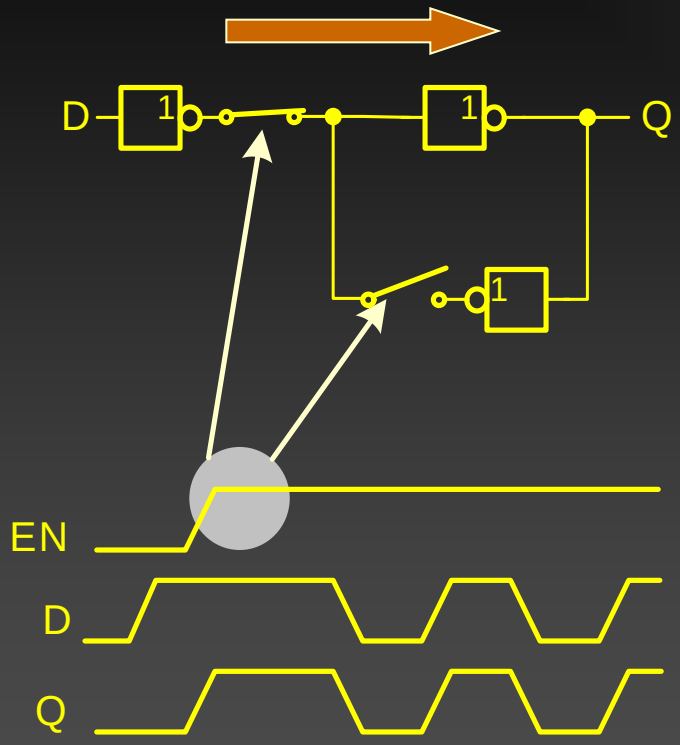
- **Realisierung:**

  Ausgang Q wird entweder von D angesteuert (transparent) oder von sich selbst (Rückkopplung). Umschaltung mittels Multiplexer aus zwei TGs

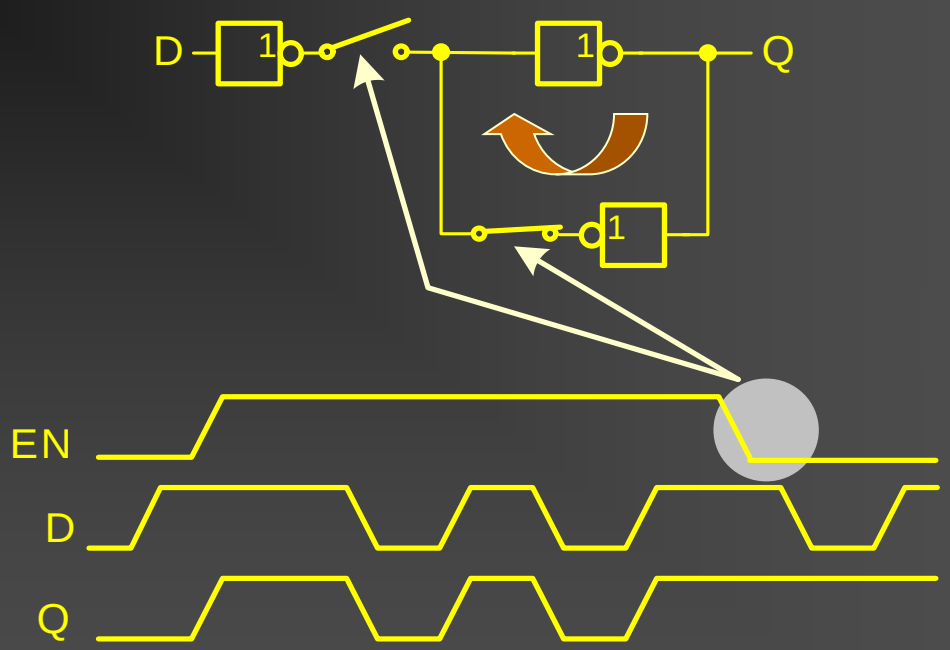
© A. Steininger / TU Wien

Funktionsmodell eines Latch

transparent



hold



© A. Steininger / TU Wien

Wiederholung

Grenzen der Geschwindigkeit



- Wellenausbreitung
  - Information kann sich niemals schneller als mit Lichtgeschwindigkeit ausbreiten. (ca. 20cm/ns)
- Ladevorgänge
  - Das Laden von Kapazitäten mit begrenztem Strom beansprucht Zeit. ($\tau = RC$)
- Bewegung der Ladungsträger
  - Bewegung/Diffusion von Ladungsträgern im Halbleiter erfolgt nur mit begrenzter Geschwindigkeit. (Sättigungswert bei Si typ. 0,1 mm/ns)

© A. Steininger / TU Wien

Setup- und Hold-Time



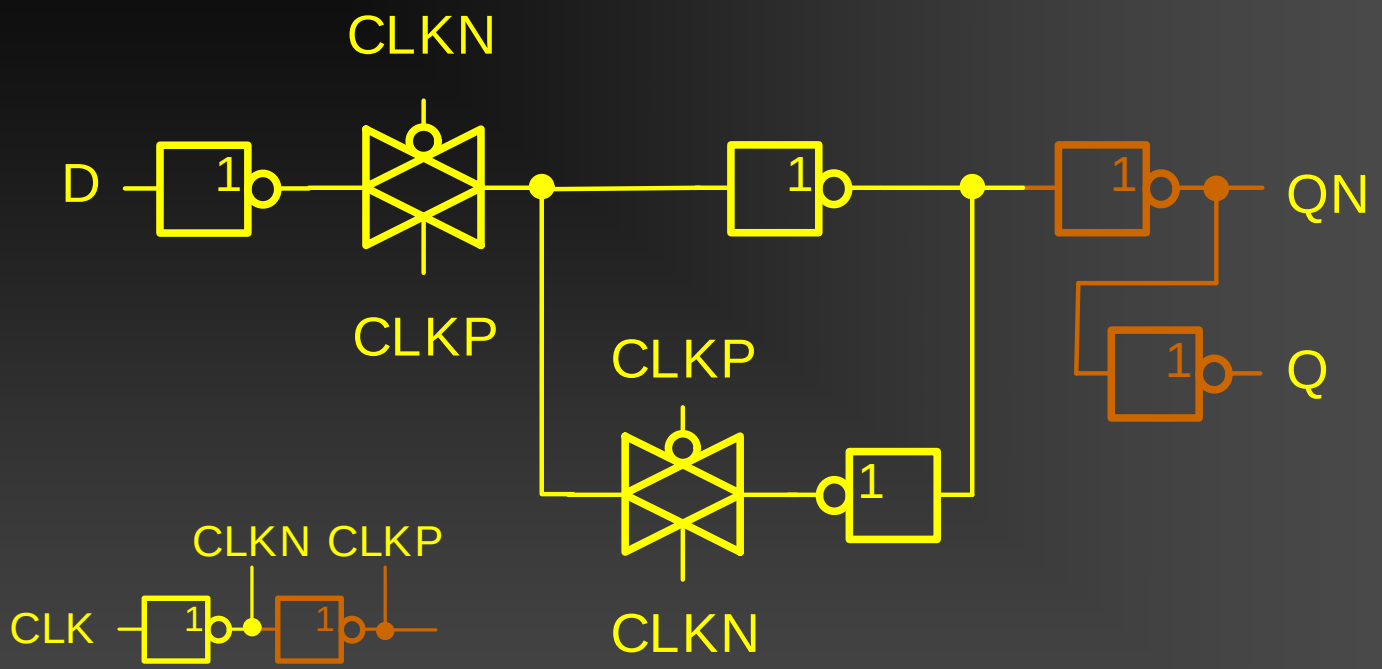
- Eine Änderung am Eingang muss die Rückkopplungsschleife vollständig durchlaufen
  UND
- Die TGs müssen umgeschalten werden

Diese Vorgänge brauchen **Zeit:**
„Decision Window" (= SetupTime + Hold-Time)

Innerhalb dieses „Decision Window" dürfen keine Flanken am Eingang auftreten (Metastabilität!)

© A. Steininger / TU Wien

Realisierung eines Latch

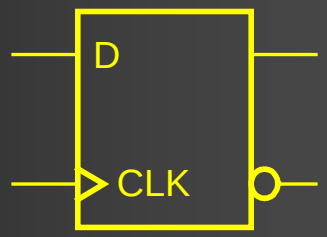


Aufwand: 7 Inv. + 2 TGs = 18 Trans = 4.5 GE

© A. Steininger / TU Wien

D-Flip-Flop





- Funktion:

  Eingänge D (Data) und CLK (Clock), Ausgang Q

  Der Zustand von D wird jeweils mit der **aktiven Flanke** auf Q übernommen und eingefroren.

- Realisierung:

  zwei Latches in Master/Slave-Schaltung

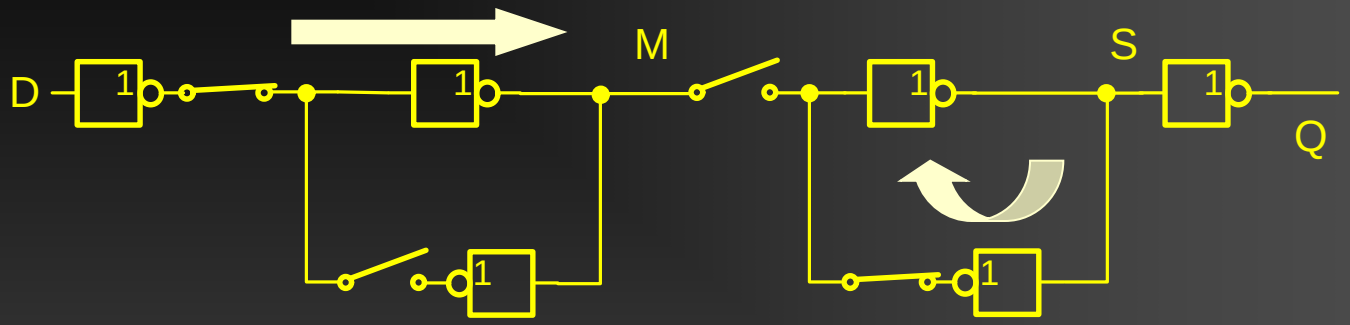
© A. Steininger / TU Wien

Funktionsmodell eines Flip-Flop

CLK = 0

load master

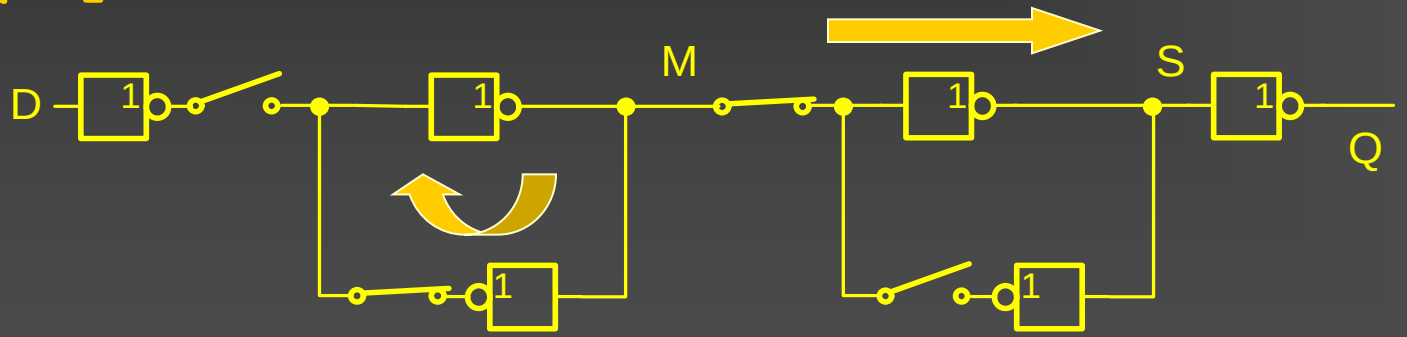
hold slave



CLK = 1

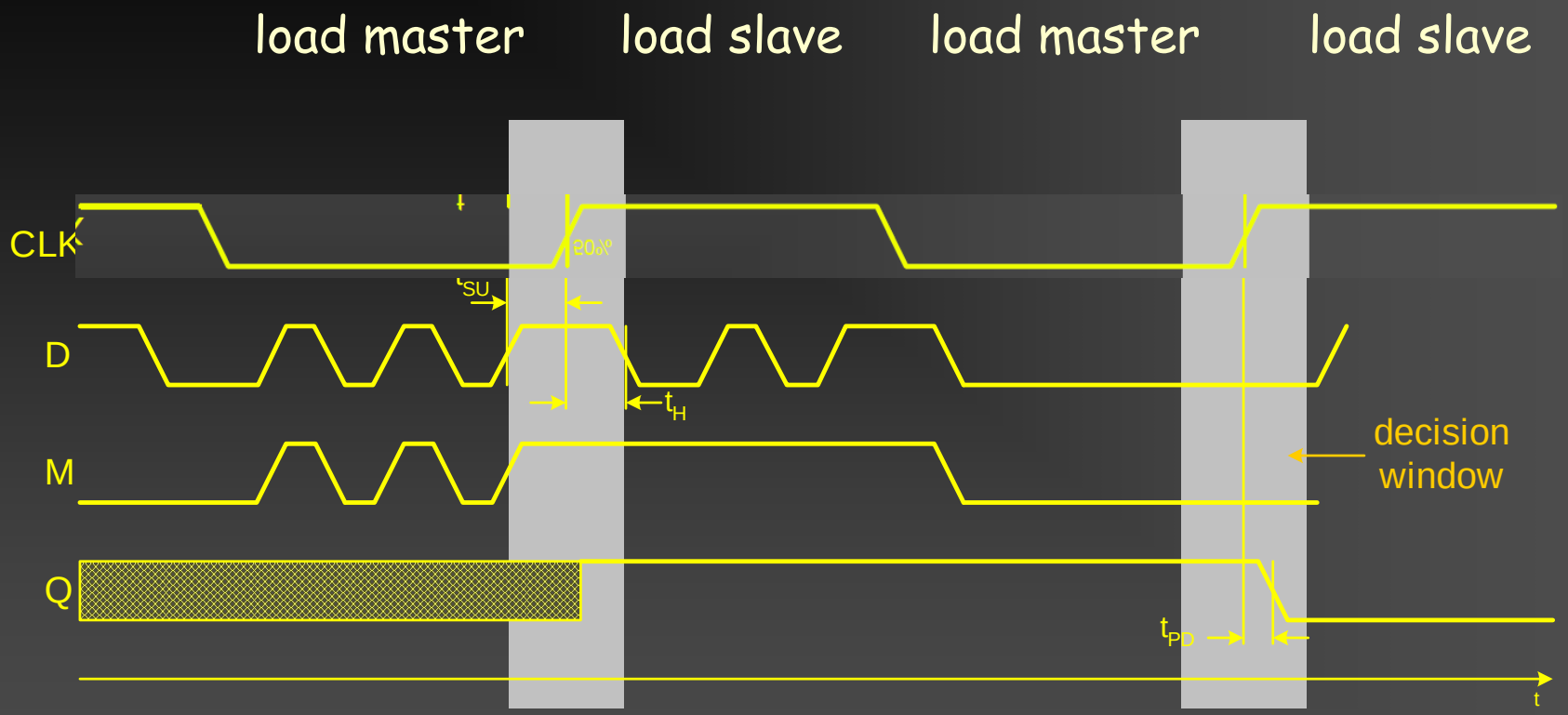
hold master

load slave



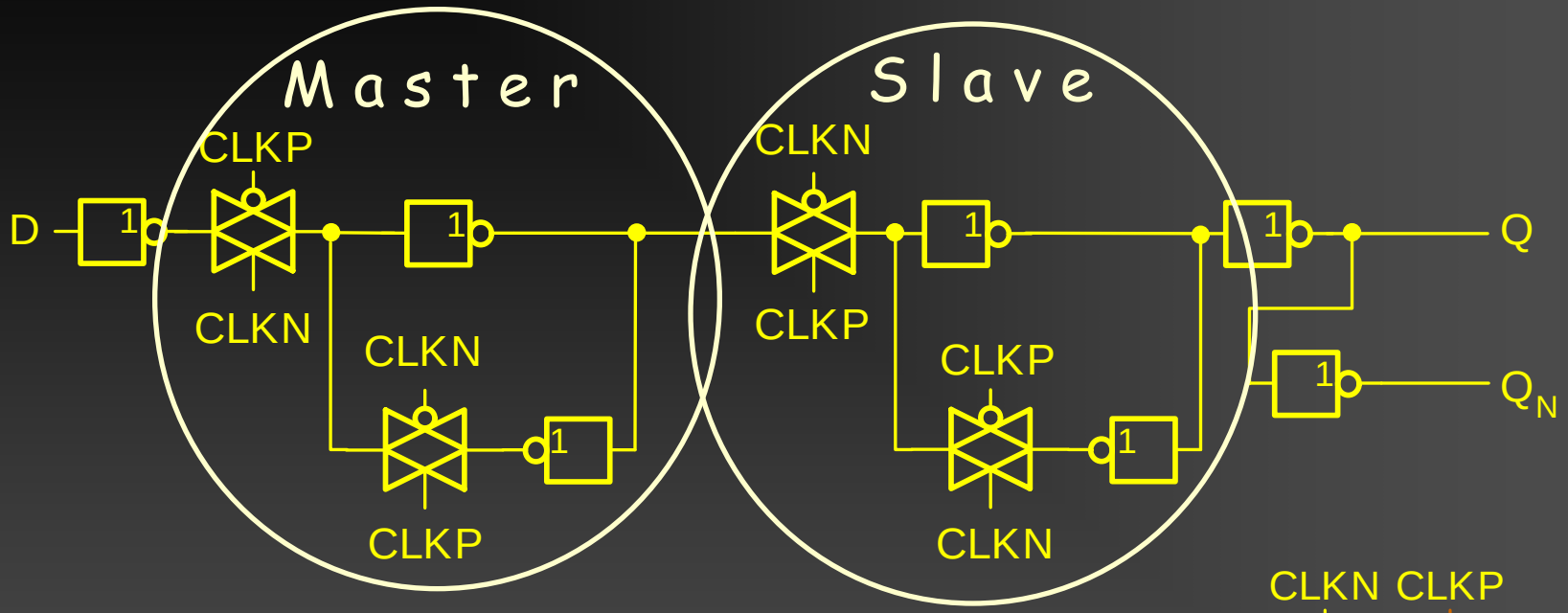
© A. Steininger / TU Wien

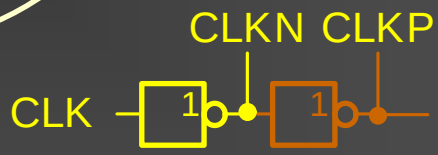
Decision Window beim Flip-Flop



© A. Steininger / TU Wien

Realisierung eines Flip-Flop





- Je 1 Latch für Master und Slave
- Taktversorgung gemeinsam
- Eingangs- und Ausgangsbuffer gemeinsam

© A. Steininger / TU Wien

Flip-Flop: Schaltungsaufwand

2 Latches entsprechen 36 Trans. = 9 GE, aber durch folgende Einsparungen

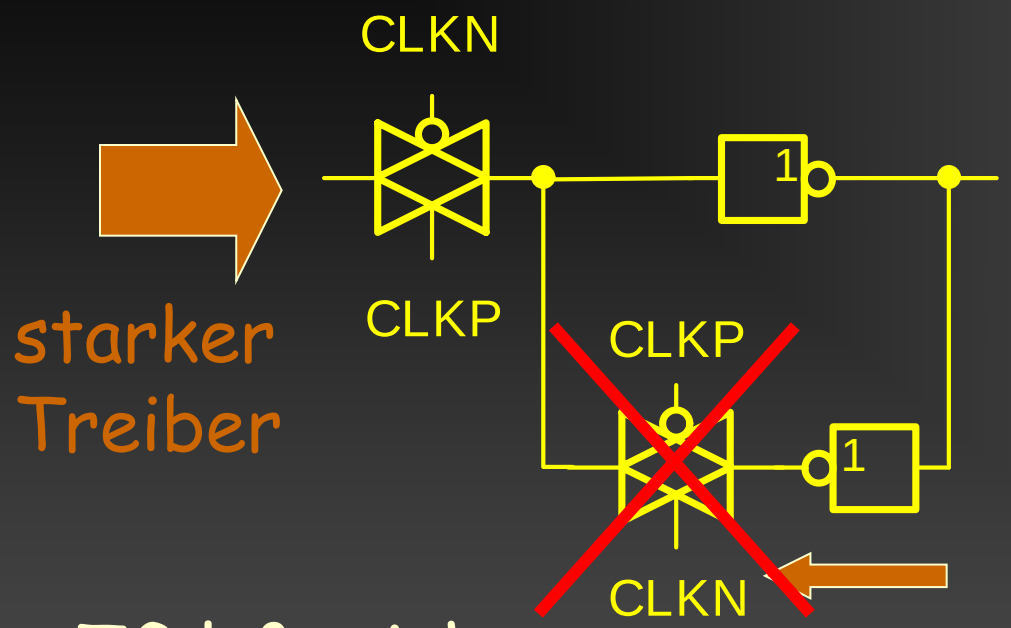
- Taktversorgung nur einmal (2 Inv.)
- Bufferung am Ausgang nur einmal (2 Inv.)
- Buffer am D-Eingang des Slave entfällt (1 Inv.)

ergibt sich ein Aufwand von

9 Inv. + 4 TGs = 26 Trans. = 6.5 GE

© A. Steininger / TU Wien

Weiteres Einsparpotential



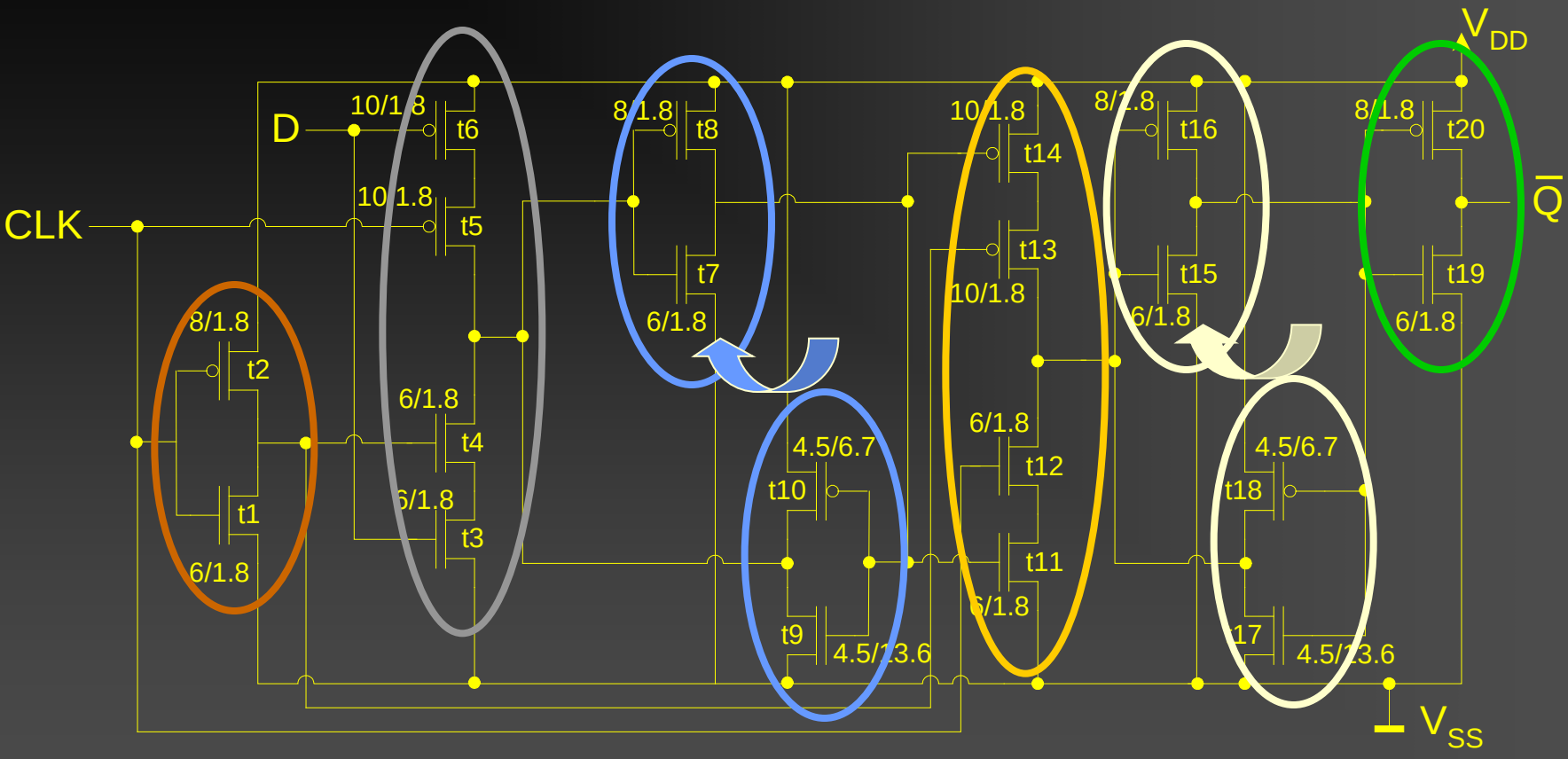
Treiberstärke einstellbar über W/L ☺

TG läßt sich einsparen

schwacher Treiber

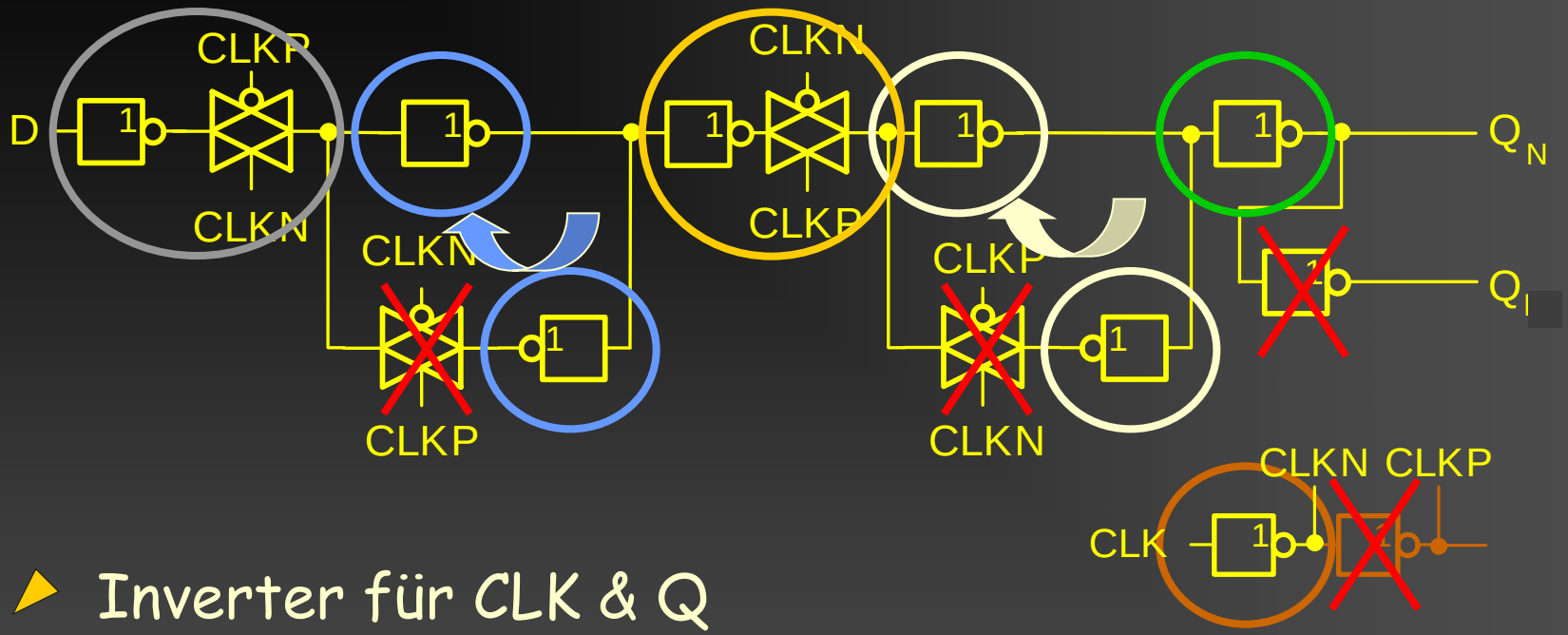
© A. Steininger / TU Wien

Flip-Flop: Implementierung



© A. Steininger / TU Wien

Realisierung eines Flip-Flop



- Inverter für CLK & Q
- 2 Speicherschleifen: TGs eingespart (Treiberstärke)
- TGs jeweils am Eingang
- Buffer eingespart (definierte Verhältnisse)

© A. Steininger / TU Wien

Setup/Hold bei anderen FFs ?

- Bei allen Typen von Flip-Flops und Latches gibt es die Setup/Hold-Problematik
  (wenn auch in unterschiedlicher Ausprägung)
- Beim SR-Latch kann z.B. kann es zu Metastabilität kommen durch
  - einen zu kurzer Puls an S bzw. R, oder
  - die "gleichzeitige" (= zu rasch aufeinander-folgende) Deaktivierung von S und R
- Es gibt kein Patentrezept gegen Metastabilität.

© A. Steininger / TU Wien

Register

Ein Register ist ein Array von Flip-Flops.

Ein 16-bit Register ist also

- ein Array aus 16 D-Flip-Flops
- mit gemeinsamem Takt
- mit gemeinsamem Clear, Enable, etc.
- Ein- und Ausgänge sind typischerweise zu „Bussen" zusammengefasst (Daten, Adressen)

© A. Steininger / TU Wien

Realisierung eines Speichers

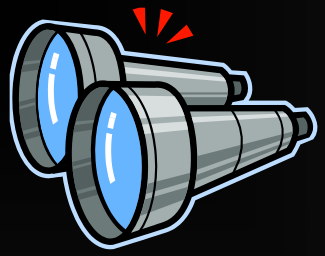
- Flip-Flops: ca. 20 Transistoren/Bit
- SRAM (siehe später): 6 Transistoren/Bit
- DRAM (siehe später): 1 Transistor/Bit (+1 Kondensator)

Realisierung größerer Speicher…

- mittels Flip-Flops ist extrem ineffizient.
- unbedingt mittels RAM-Blöcken aus der Library realisieren.

!

© A. Steininger / TU Wien



Überblick

- Was ist CMOS ?
- Feldeffekt-Transistor & CMOS-Prozess
- kombinatorische Logikzellen
- sequentielle Logikzellen
- **weitere Logikfamilien**

© A. Steininger / TU Wien

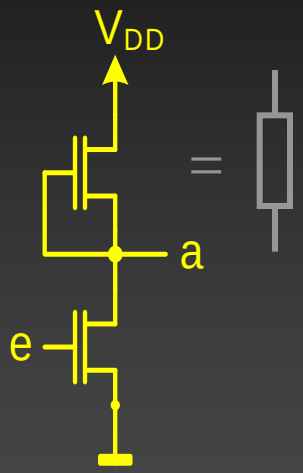
Weitere MOS-Logikfamilien

NMOS

nur n-Kanal FETs

Nachteil:

„weak" 1

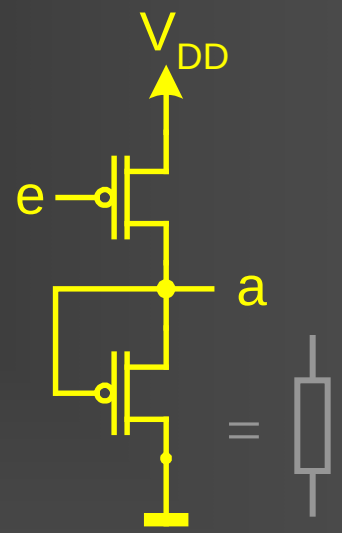


PMOS

nur p-Kanal-FETs

Nachteil:

„weak" 0



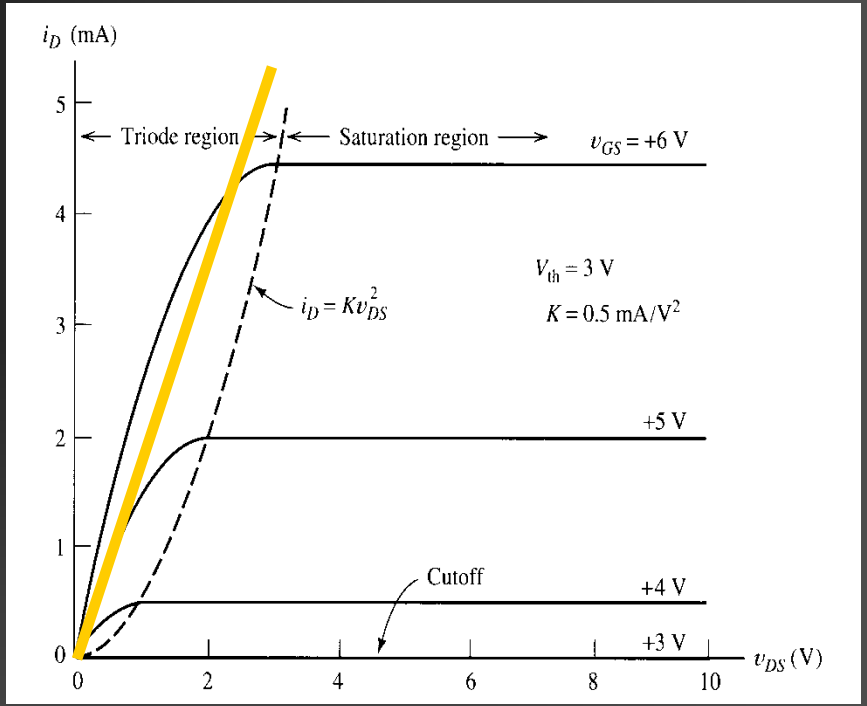
Vorteil: Fertigung billiger (weniger Masken)

Nachteil: statischer Stromverbrauch

© A. Steininger / TU Wien

Der FET als Widerstand

- Integration von Widerständen ist schwierig (Kohle, Metallfilm?)
- Bei geeigneter Auslegung verhält sich FET in guter Näherung wie Widerstand
- Widerstand einstellbar über Formfaktor W/L



➡ Realisierung v. Widerständen dch FETs

© A. Steininger / TU Wien

Bipolare Logikfamilien: TTL

TTL (Transistor-Transistor-Logic)

- Prinzip: Logische Verknüpfungen über Dioden-Netzwerke bzw. Transistor-Schalter
- sehr ähnlich wie bei CMOS, aber mit Bipolar-Transistoren statt FETs
- verbraucht im Ruhezustand mehr Energie
- weitgehend kompatibel zu CMOS
- weitgehend von CMOS abgelöst
- legendäre 74xx-Serie (Sylvania 1963)

© A. Steininger / TU Wien

Bipolare Logikfamilien: ECL

ECL (Emitter-Coupled Logic)

- Prinzip: Umschalten von Strompfaden in Differenzverstärkern (mit Bipolartransistoren)
- Wenig Spannungshub, keine Sättigung, daher extrem schnell
- Extrem hoher Leistungsverbrauch
- Weit verbreitete 10K und 100K-Familien
- Nicht kompatibel zu CMOS und TTL
- Anwendung: Glasfaser-Interface, ATM

© A. Steininger / TU Wien

Bi-CMOS Logik

... BIpolar und CMOS gemischt

- Schaltung hauptsächlich in CMOS realisiert, aber
- Bipolar-Transistoren sind für höhere Ströme geeignet, daher für die Ausgangsstufen (Treiber) verwendet, z.T für direkte Ansteuerung von kleineren Motoren o.ä.
- Mischen der Technologien macht Fertigung komplizierter und daher teurer

© A. Steininger / TU Wien



Zusammenfassung (1)

- Grundelement der digitalen Logik ist der Enhancement-FET, wobei bei CMOS der n-Kanal-Typ und der p-Kanal komplementär zum Einsatz kommen.
- Die wichtigsten Parameter des FET sind Schwellspannung und Ausgangsstrom (bzw. Formfaktor)
- Im Idealfall verhält sich ein FET wie ein Schalter: der n-Kanal-FET schließt bei 1 am Steuereingang, der p-Kanal-FET bei 0.
- Die Idealisierung als Schalter funktioniert nur unter geeigneten Randbedingungen. Bei genauerer Betrachtung (im Zeit oder Amplitudenbereich) verhält sich der FET wie ein analoges Bauelement.

© A. Steininger / TU Wien



Zusammenfassung (2)

- Der Inverter ist die Grundstruktur aller Logik-funktionen. Er lässt sich technologisch einfach implementieren.
- Ersetzt man die beiden Einzeltransistoren durch einen sog. n-Stack bzw. p-Stack, so lassen sich bei geeigneter Abstimmung allgemeine logische Funktionen wie AOI und OAI implementieren, sowie als Sonderfälle auch NAND und NOR.
- Nicht invertierende Funktionen können in CMOS nicht einstufig realisiert werden.
- Weitere typische Elemente sind Transmission Gate, Multiplexer und getakteter Inverter.

© A. Steininger / TU Wien



Zusammenfassung (3)

- Mittels getakteter Inverter kann ein Latch realisiert werden, durch Master/Slave Kombination zweier Latches ein Flip-Flop.
- Aufgrund der Einschwingzeit der Datenpfade (und insbesondere der Speicherschleife) darf innerhalb des „Decision-Window" (Summe aus Setup- und Hold-Time) keine Änderung der Daten erfolgen, sonst kann Metastabilität auftreten.
- Die Realisierung eines ganzen Speichers mittels Flip-Flop oder Latch ist sehr ungünstig, effizienter sind hier SRAM oder DRAM.

© A. Steininger / TU Wien



Zusammenfassung (4)

- Neben dem komplementären Ausgang gibt es den Tri-State Ausgang sowie den Open Drain Ausgang.
- Die CMOS-Technologie ist derzeit am weitesten verbreitet, in besonderen Anwendungen findet man jedoch auch bipolare Logikfamilien wie TTL oder ECL, oder auch Bi-CMOS (für hohe Treiberleistung).

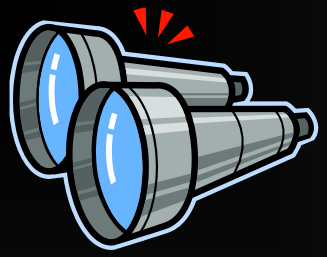
© A. Steininger / TU Wien

Der Design-Flow eines ASIC

Von der Idee zum funktionierenden Produkt



© A. Steininger / TU Wien



Überblick

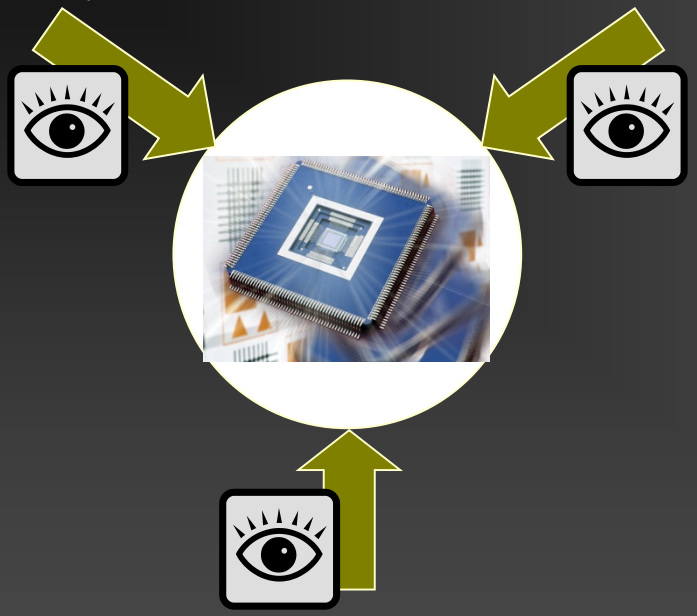
- Das Y-Diagramm
- Design-Schritte
- Synthese und PPR als Optimierungen
- Spezifische Probleme des Routing
- Verifikations-Schritte
- Simulation
- Statische Timing-Analyse

© A. Steininger / TU Wien

Verschiedene Sichtweisen

Verhalten:
Was tut der Chip?

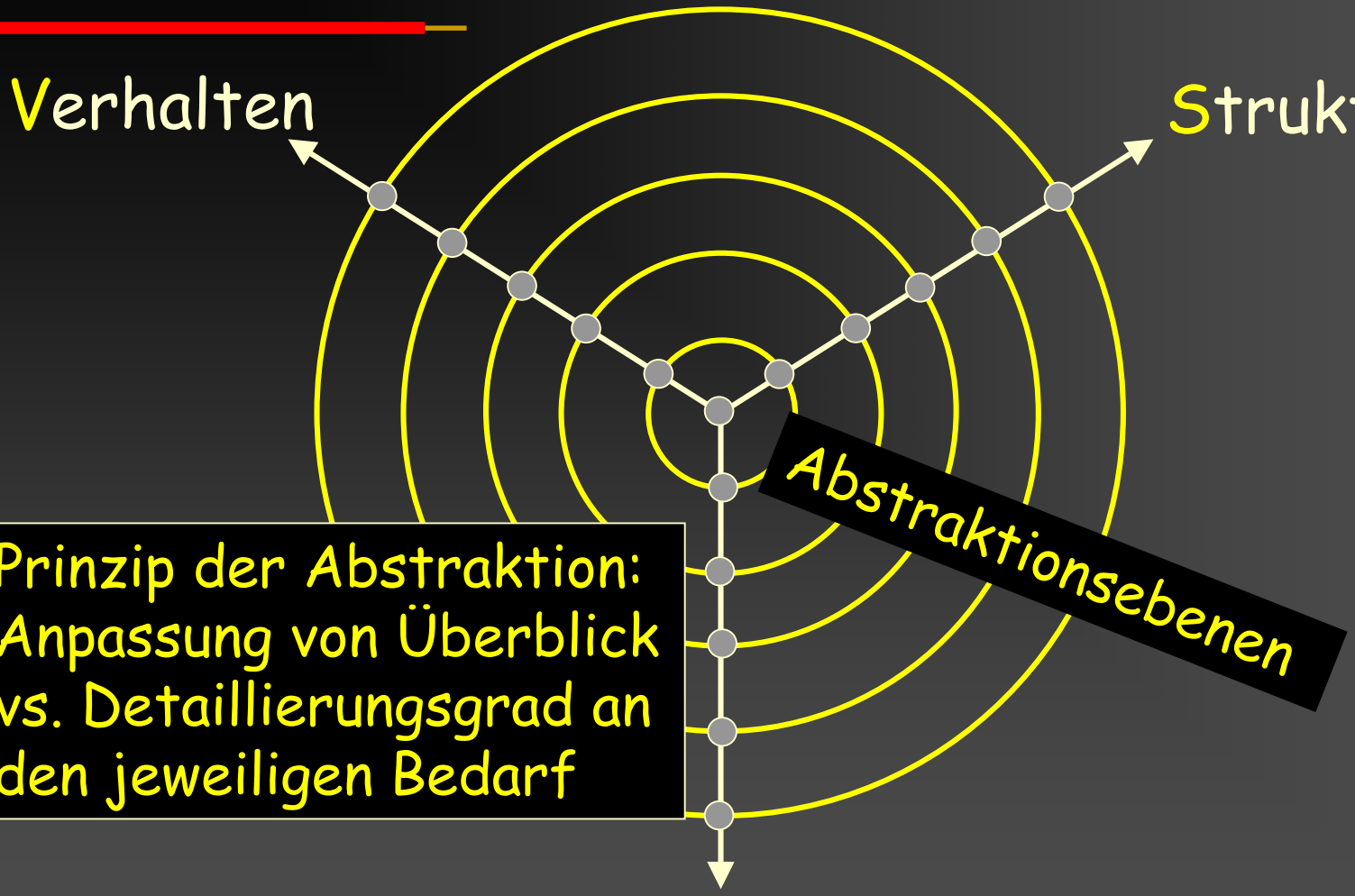
Struktur:
Welche Blöcke umfasst er?



Geometrie:
Wie ist er aufgebaut?

© A. Steininger / TU Wien

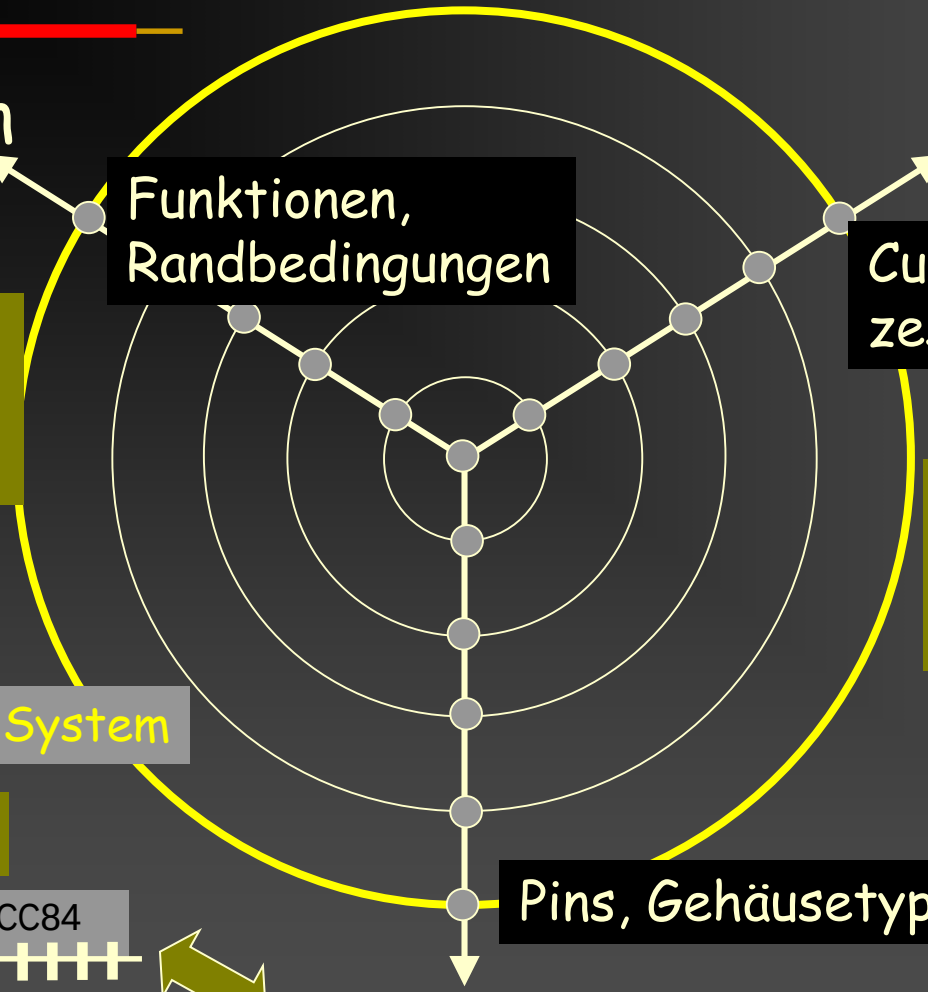
Das Y-Diagramm von Gajski



Prinzip der Abstraktion:
Anpassung von Überblick vs. Detaillierungsgrad an den jeweiligen Bedarf

© A. Steininger / TU Wien

Y-Diagramm: Systemebene



Verhalten

Struktur

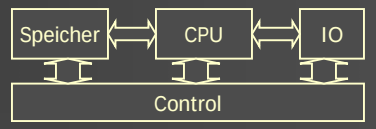
Funktionen, Randbedingungen

Custom-HW, Pro-zessor,Speicher, ...

(funktionale) System-Spezifikation

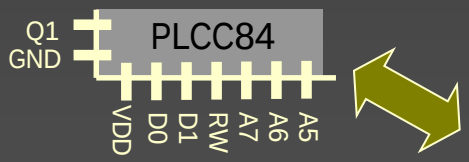
Inputs : Keyboard
Output: Display
Funktion: Umrechnung,...

Funktions-schaltbild, Partitioning



System

Package

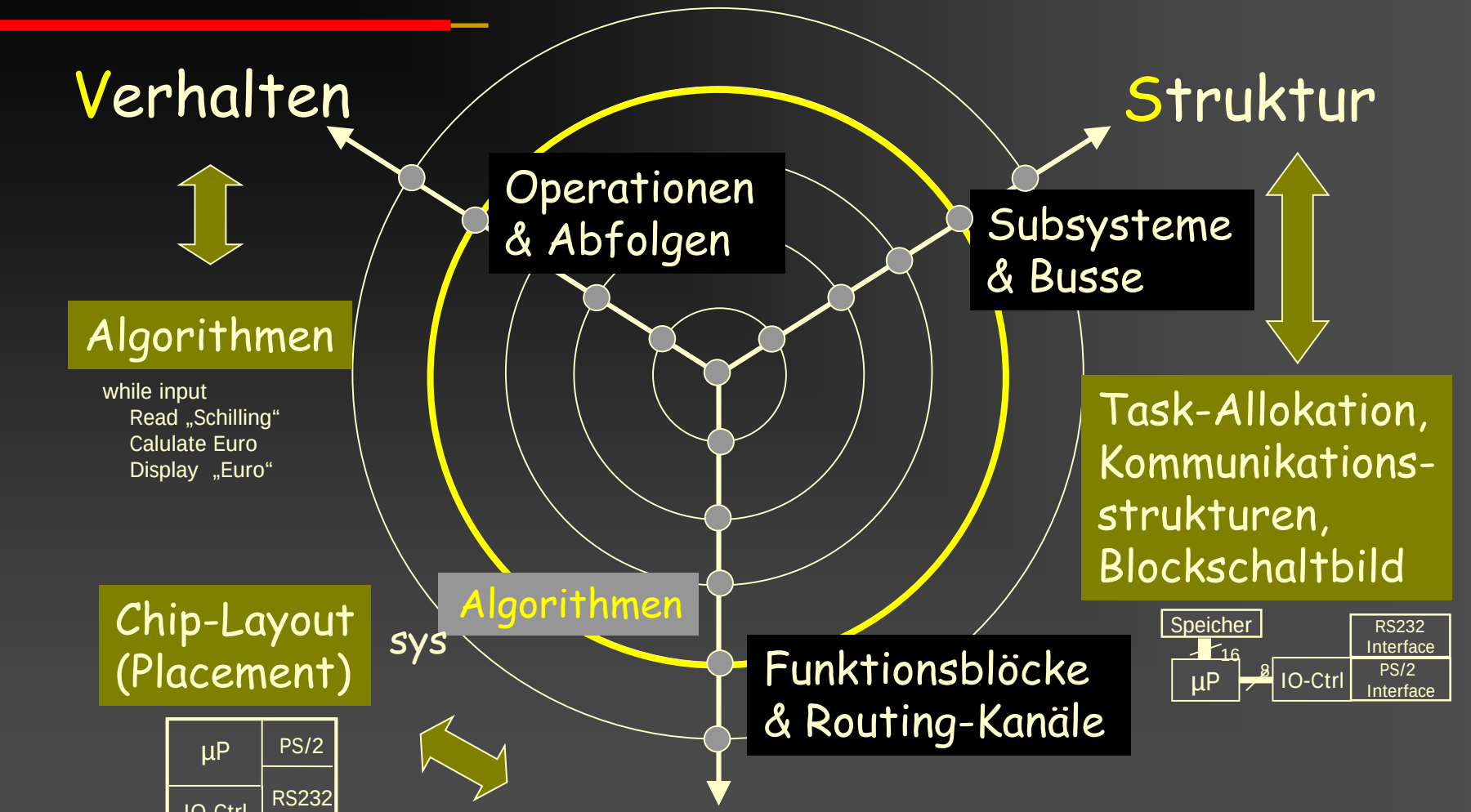


Pins, Gehäusetyp

Geometrie

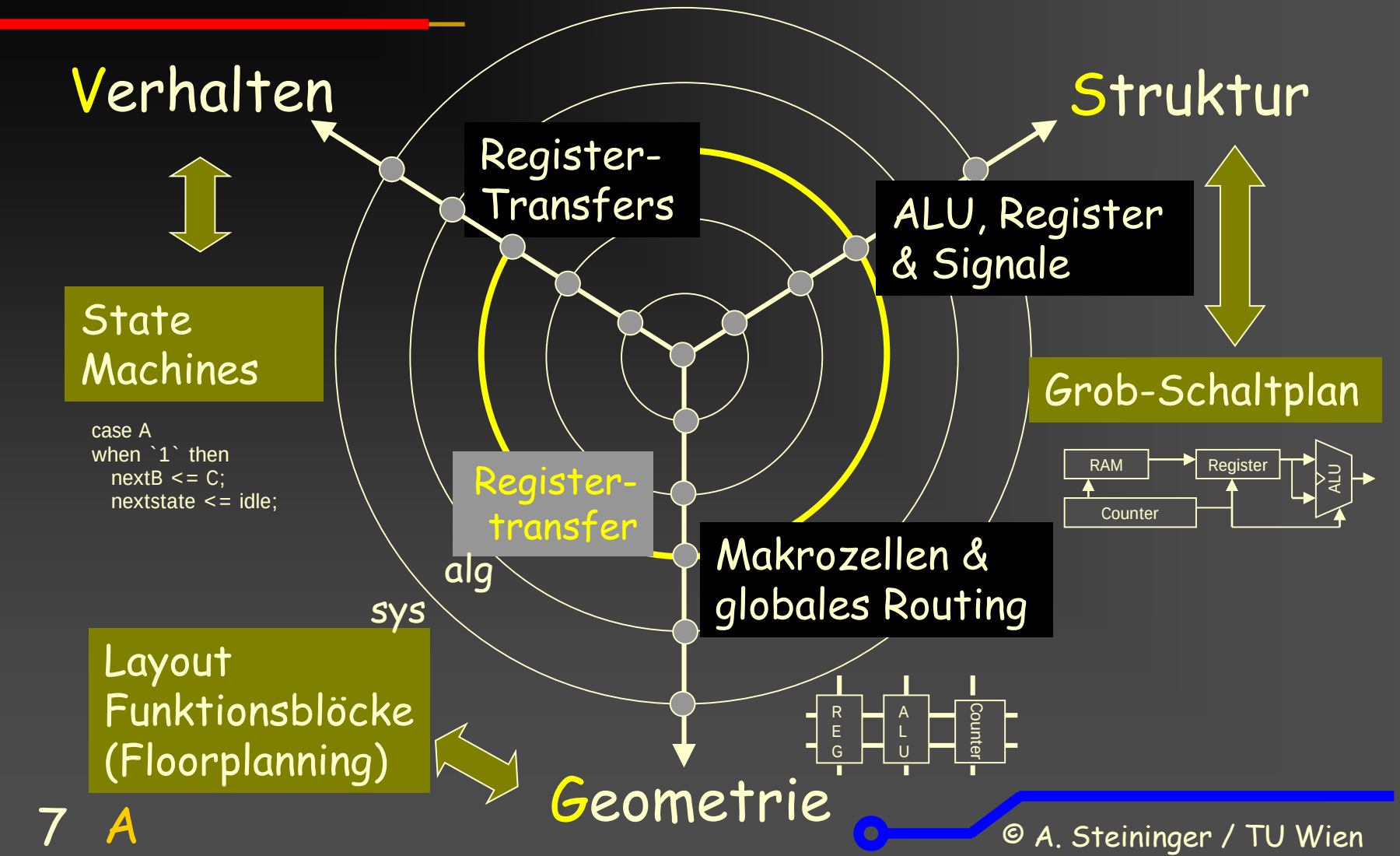
© A. Steininger / TU Wien

Y-Diagramm: Algorithm. Ebene



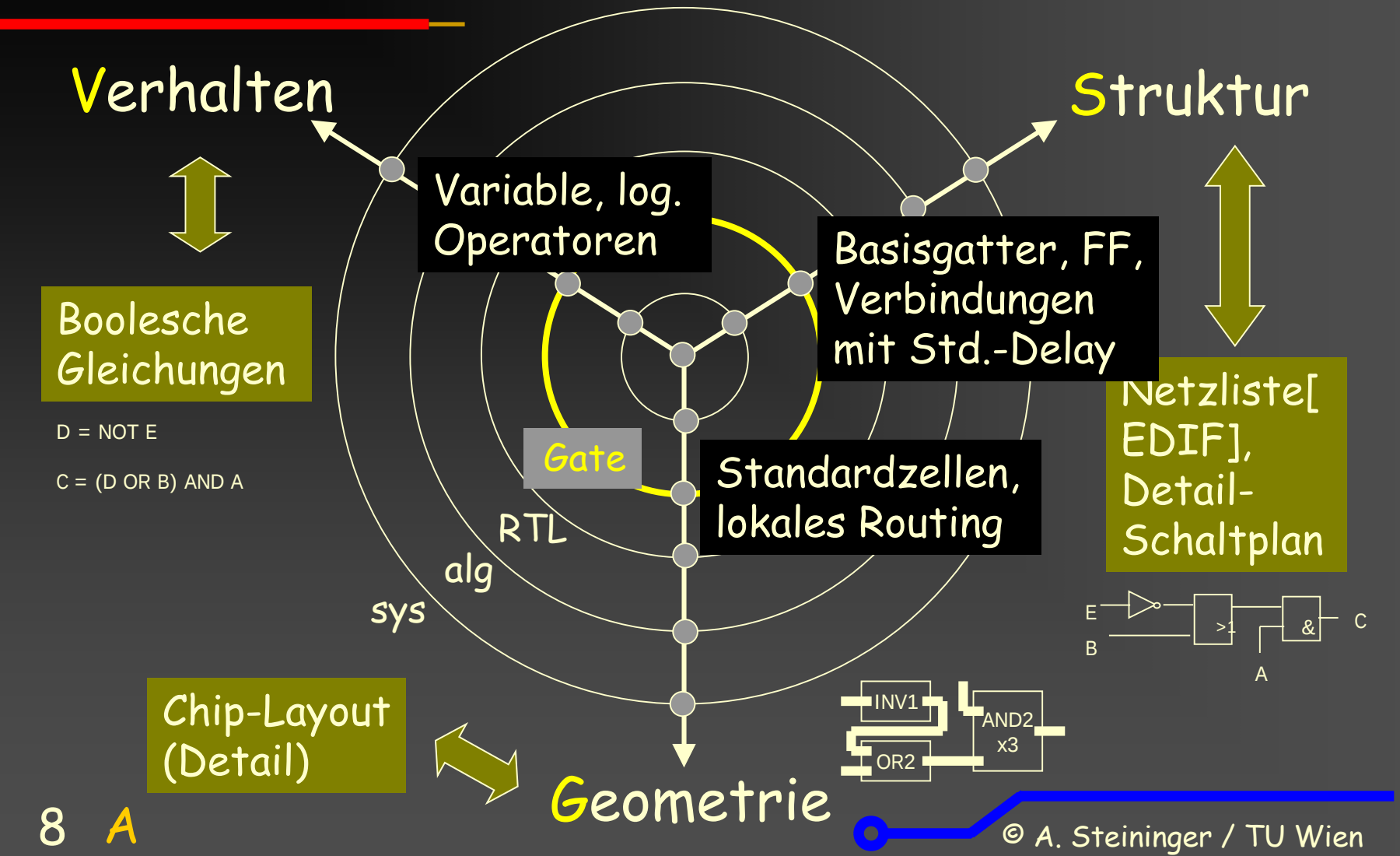
© A. Steininger / TU Wien

Y-Diagramm: RTL Ebene



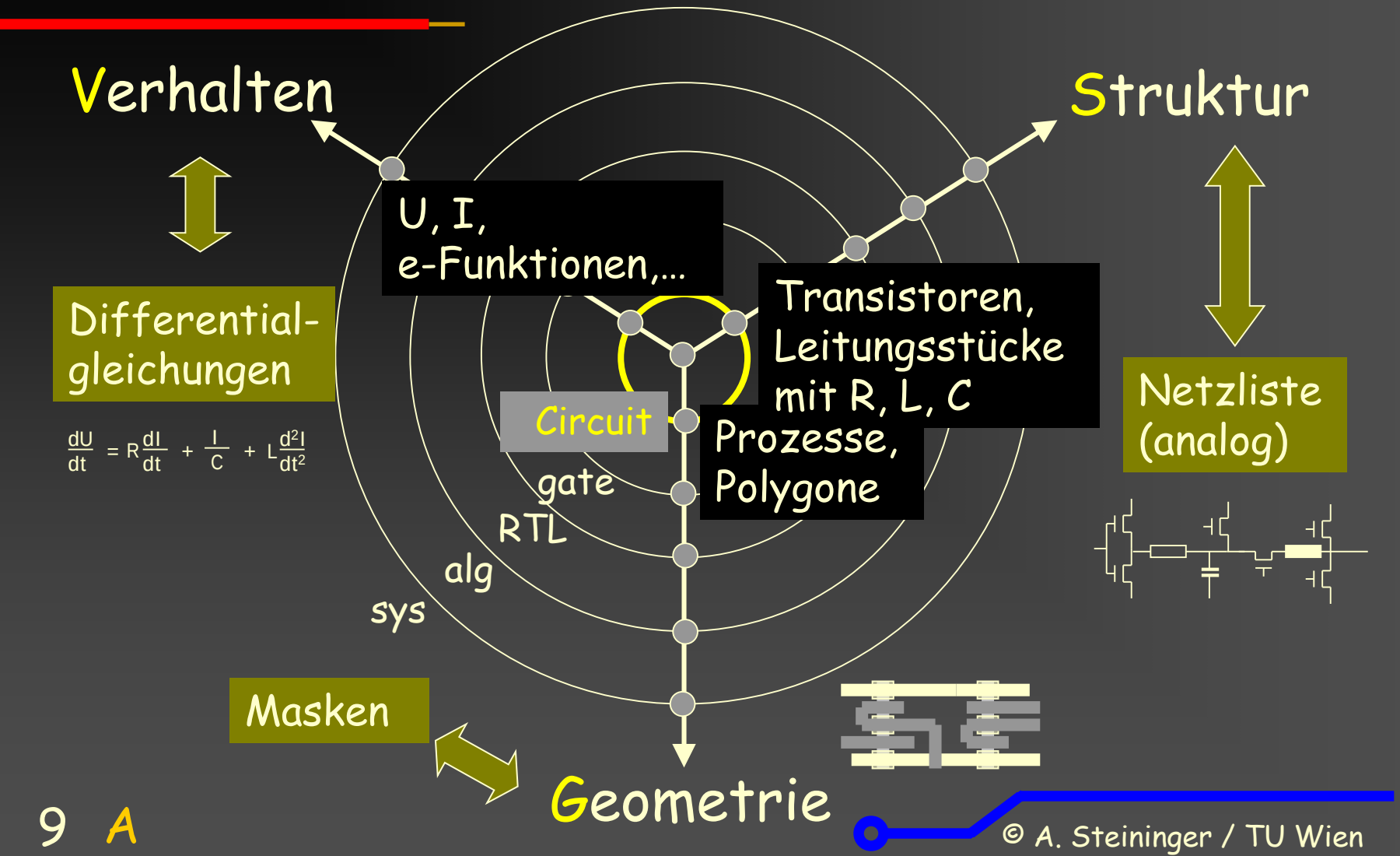
© A. Steininger / TU Wien

Y-Diagramm: Logikebene



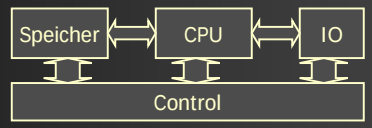
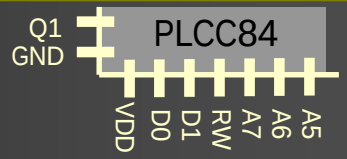
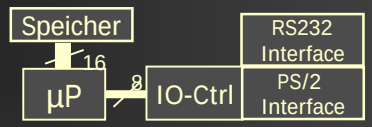
© A. Steininger / TU Wien

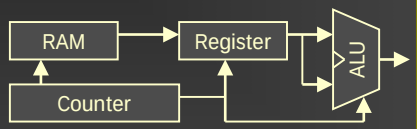
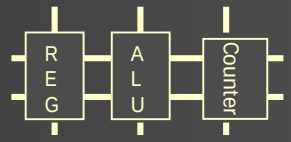
Y-Diagramm: Schaltkreisebene

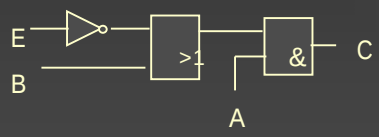
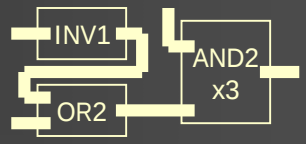
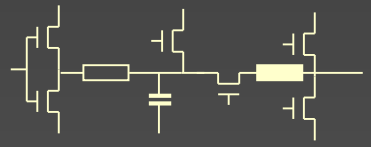


© A. Steininger / TU Wien

„Y-Tabelle"

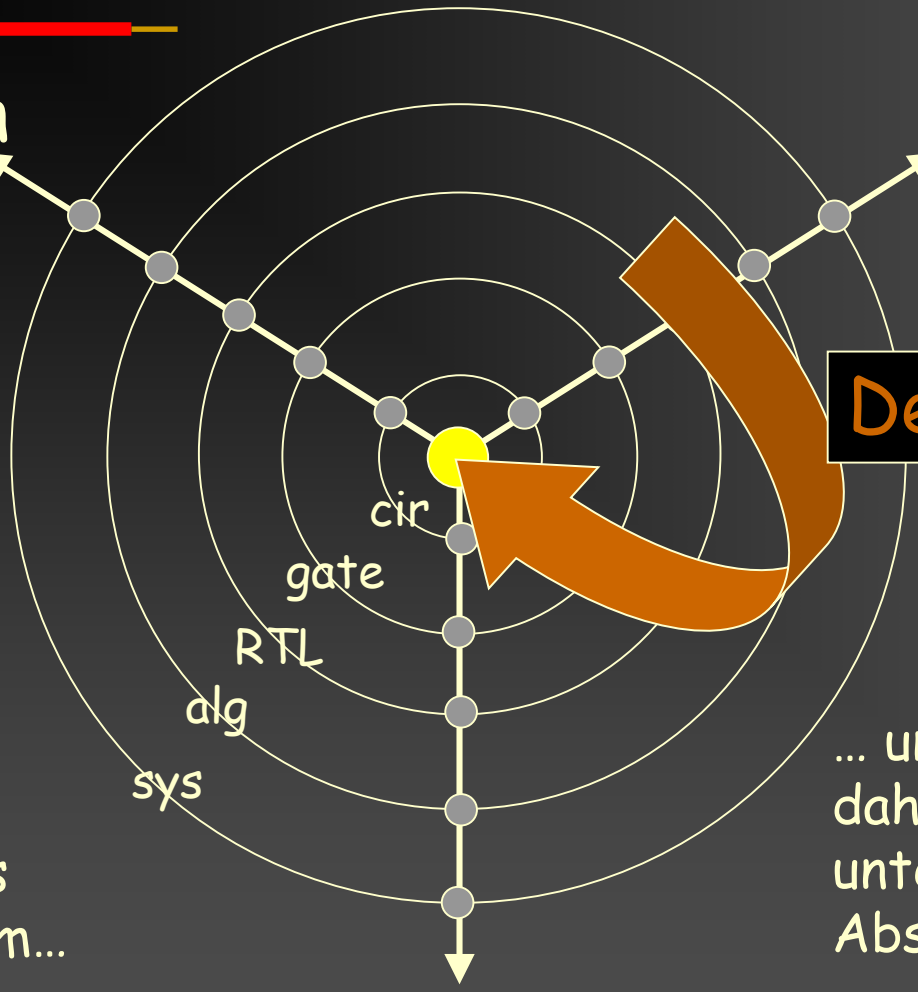
Ebene	Verhalten	Struktur	Geometrie
System	Inputs : Keyboard Output: Display Funktion: Umrechnung,...		
Algorithmisch	while input Read „Schilling" Calulate Euro Display „Euro"		

Registertransfer (RTL)	case A
	when `1` then
	nextB <= C;
	nextstate <= idle;
			

Logik	D = NOT E C = (D OR B) AND A		
Schaltkreis	$\frac{dU}{dt} = R\frac{dI}{dt} + \frac{I}{C} + L\frac{d^2I}{dt^2}$		

© A. Steininger / TU Wien

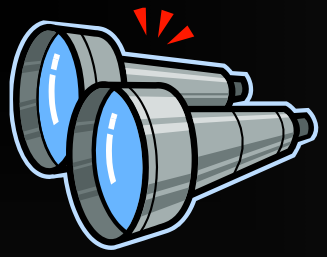
Y-Diagramm: Konvergenz



alle drei Sichtweisen beschreiben letztlich das selbe System…

… und konvergieren daher auf der untersten Abstraktionsebene

© A. Steininger / TU Wien

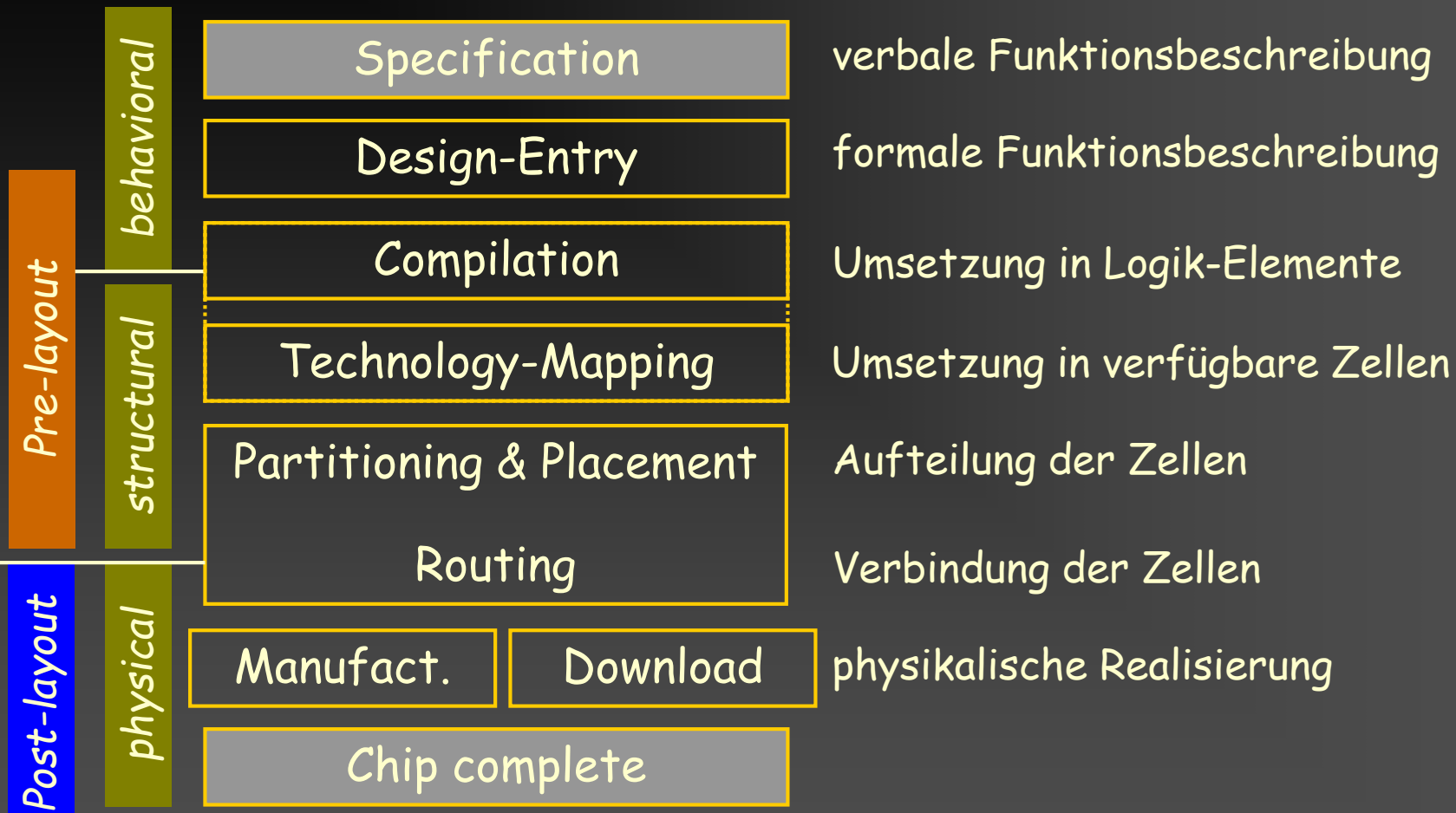


Überblick

- Das Y-Diagramm
- **Design-Schritte**
- Synthese und PPR als Optimierungen
- Spezifische Probleme des Routing
- Verifikations-Schritte
- Simulation
- Statische Timing-Analyse

© A. Steininger / TU Wien

Design-Flow im Überblick



© A. Steininger / TU Wien

Verifikationsschritte



Pre-layout

Post-layout

behavioral

structural

physical

Specification

Design-Entry

Compilation

Technology-Mapping

Partitioning & Placement

Routing

Manufact.

Download

Chip complete

Validation

Behavioral Simulation

Functional Simulation

Prelayout-
Gate-Level-Simulation

Postlayout-
Gate-Level-Simulation

Test

© A. Steininger / TU Wien

Spezifikation

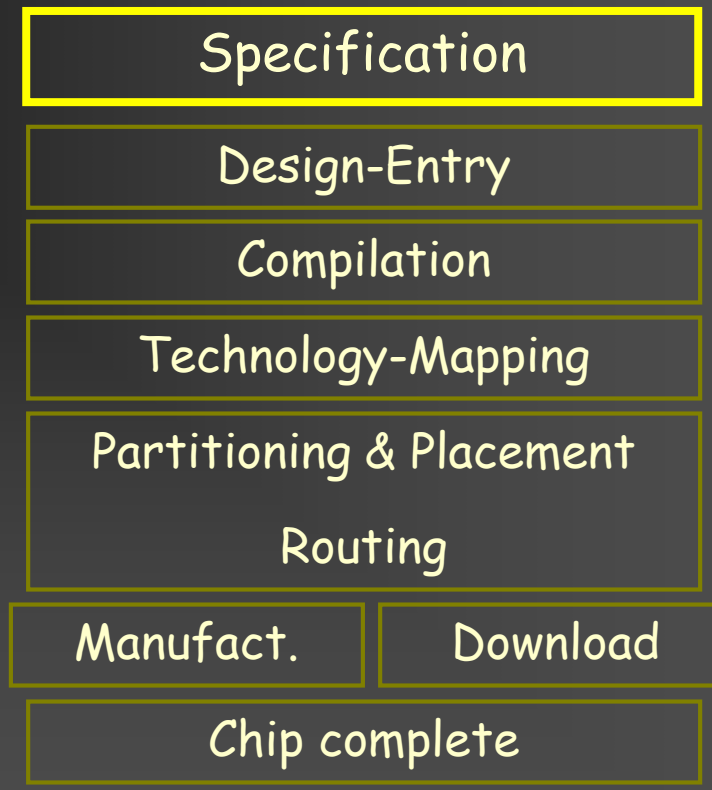
- Zweck:

  Exakte Formulierung der vom Produkt gewünschten Funktion und der entsprechenden Betriebsbedingungen.

  Meist nicht in formaler Darstellung sondern verbal bzw. mit Skizzen.

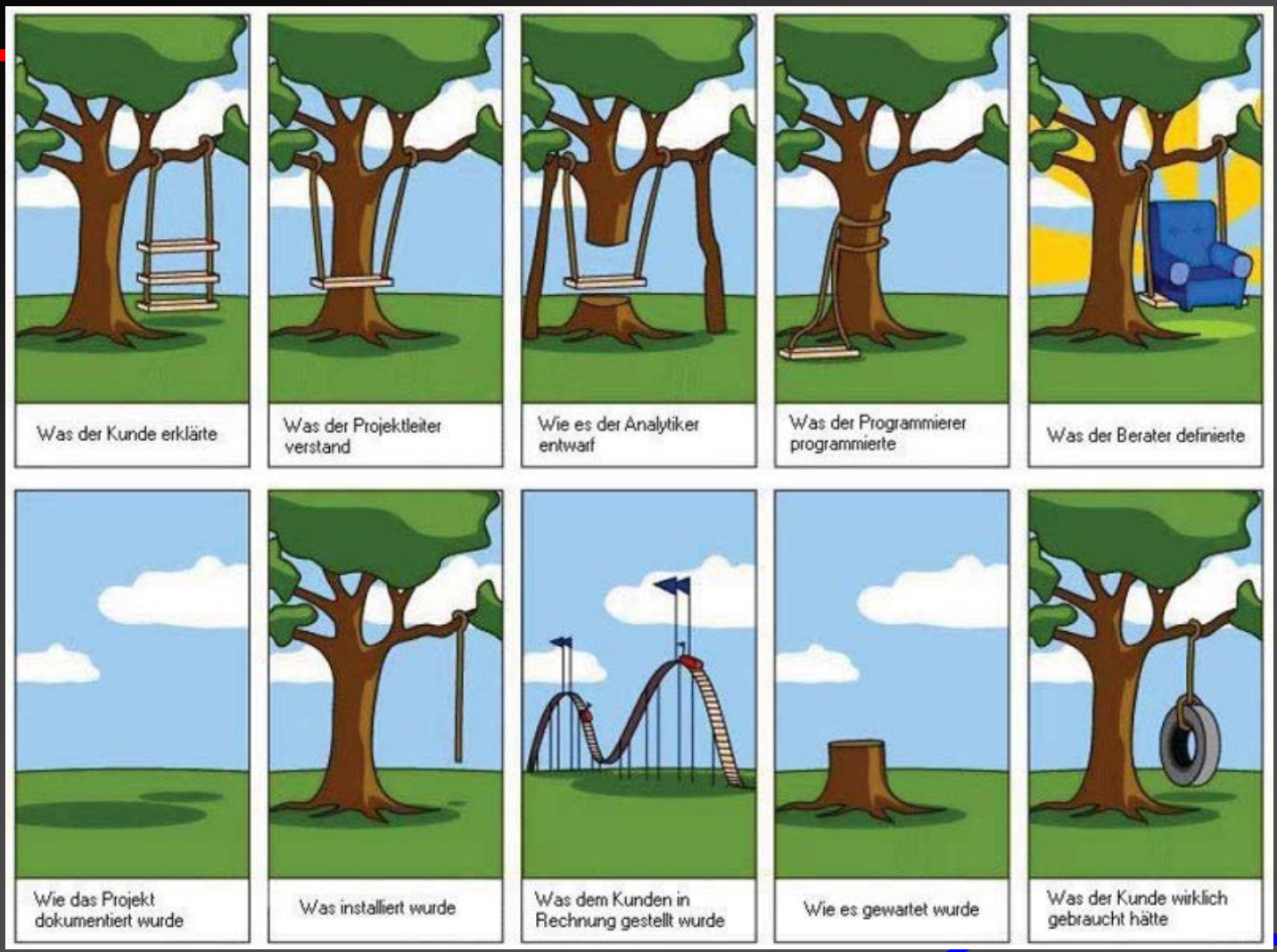


Idee



© A. Steininger / TU Wien

Die Realität ....



[ZID TU Wien]

© A. Steininger / TU Wien

Beispieldesign: Spezifikation

Synchronisierschaltung für BCD-Eingang

- Eingangsvektor data_d[3:0] (4 Bit BCD-Wert)
- Ausgangsvektor digit_L_d[3:0]
- Synchronisation auf positive Flanke von clk
- reset_board setzt den Ausgang synchron auf „0000“

Hier fehlen noch Angaben:

Temperaturbereich, Versorgungsspannung, zulässige Größe, Preis, Zuverlässigkeit, Geschwindigkeit, ….

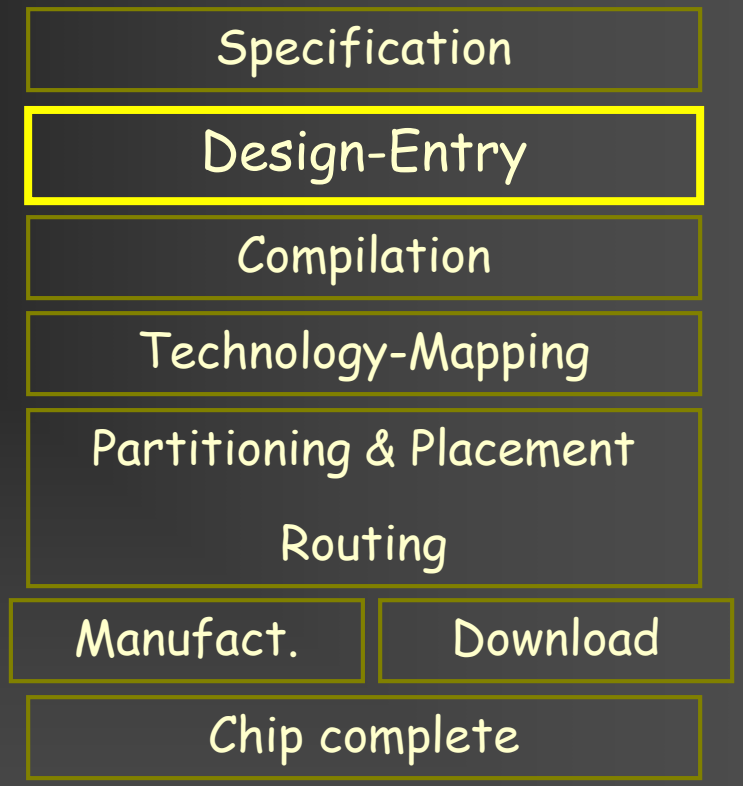
© A. Steininger / TU Wien

Design-Entry



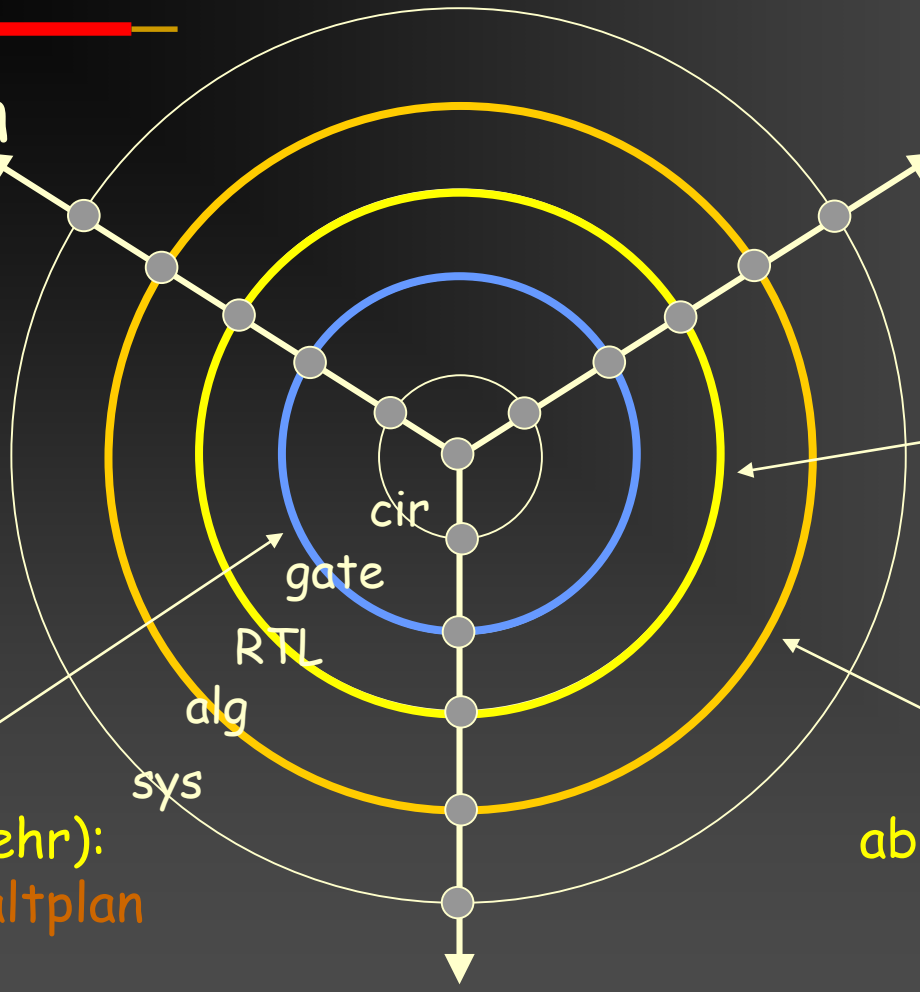
- Zweck:

  Umsetzung der Spezifikation in eine Form,
  - die vom Computer erfasst werden kann
  - die als Basis für Simulation und Änderungen dient
  - die als Dokumentation geeignet ist



© A. Steininger / TU Wien

Abstraktion v. Design Entry



State of the Art: VHDL

Können wir, wollen wir aber nicht (mehr): (Gatter-)Schaltplan

Wollen wir, können wir aber (noch) nicht: System-C

© A. Steininger / TU Wien

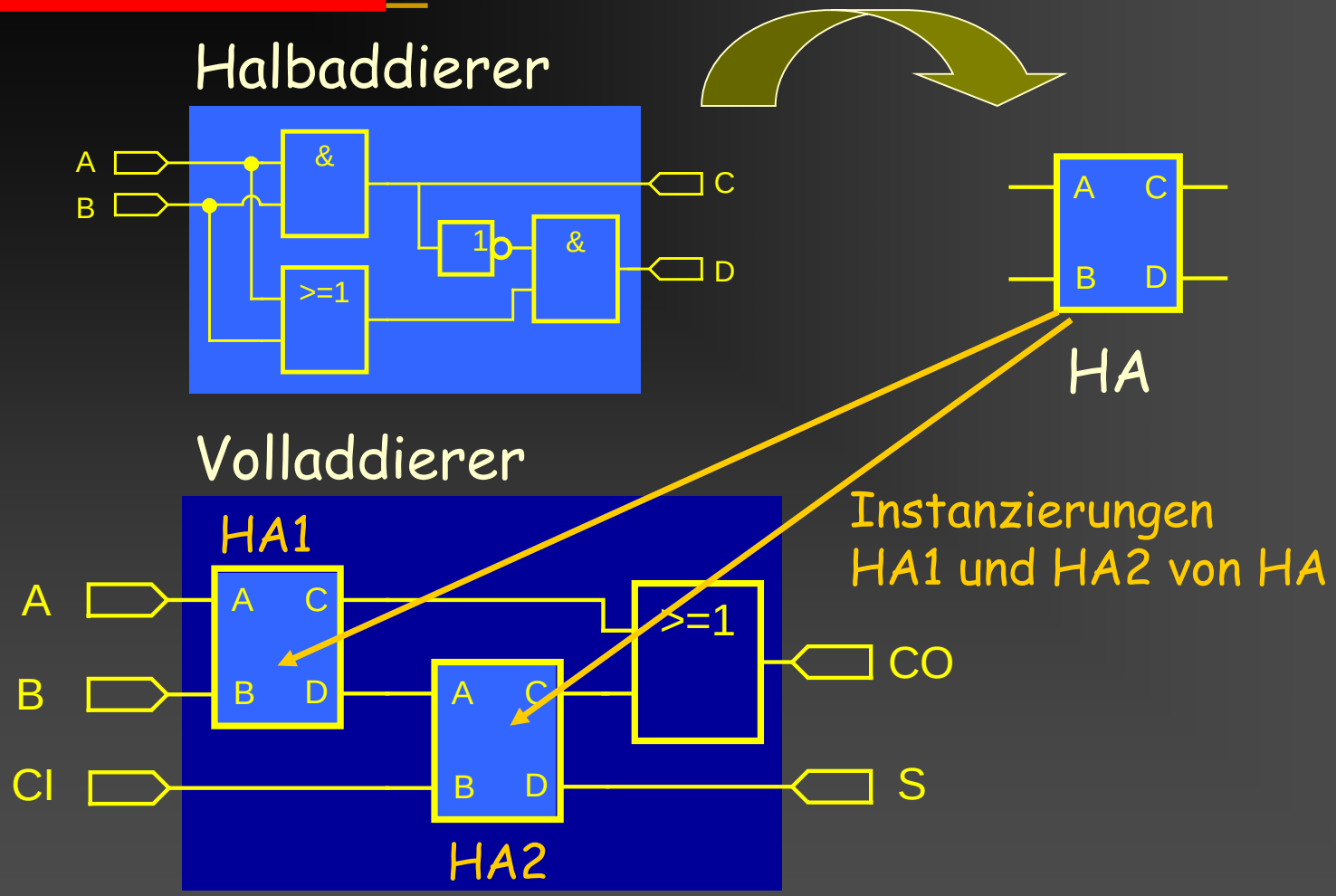
Design-Entry: Möglichkeiten

	grafisch	*textuell*
high-level	State-Chart (Zustandsgraph)	VHDL, Verilog, System C
low-level	Schematic Entry (Schaltplan)	ABEL, CUPL PALASM

- Kriterien:
  - Unterstützung der menschlichen Intuition
  - Effizienz der Darstellung
  - Weiterverarbeitbarkeit durch Computer

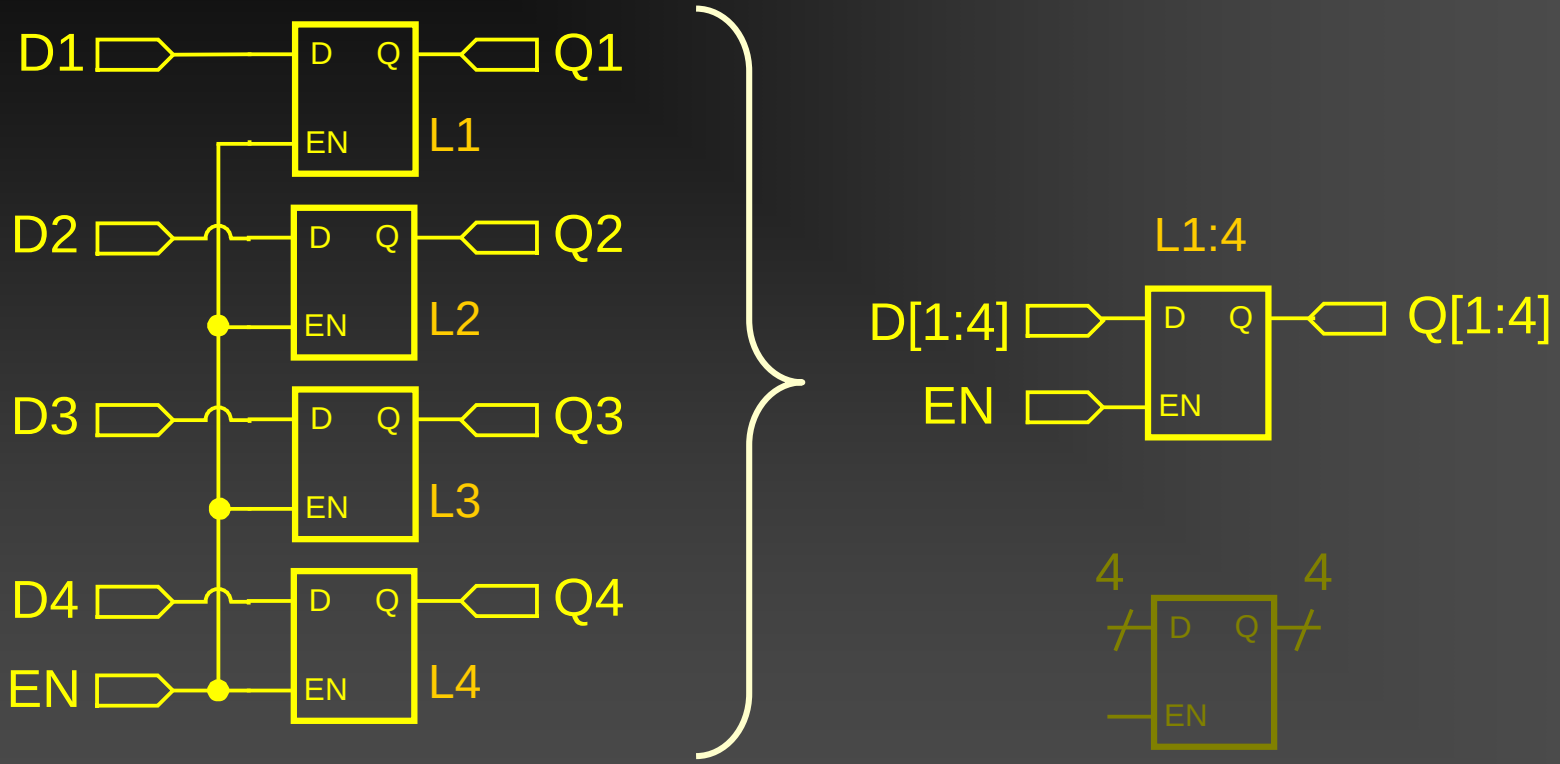
© A. Steininger / TU Wien

Design-Hierarchie



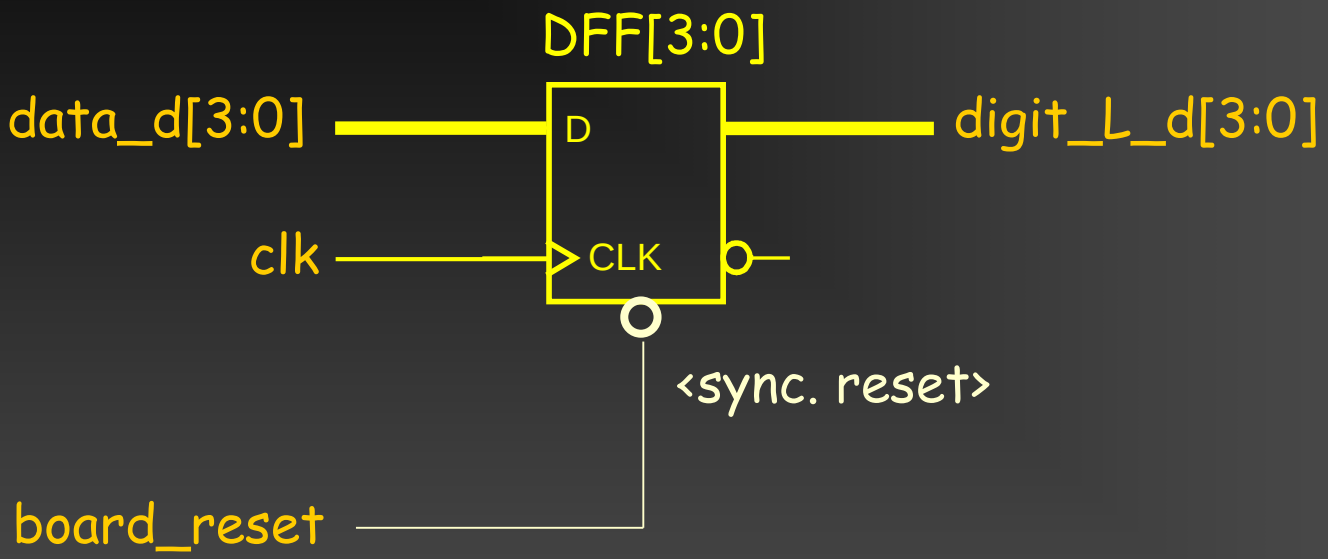
© A. Steininger / TU Wien

„Vectored Instance"



© A. Steininger / TU Wien

Bsp.-design: Schematic Entry



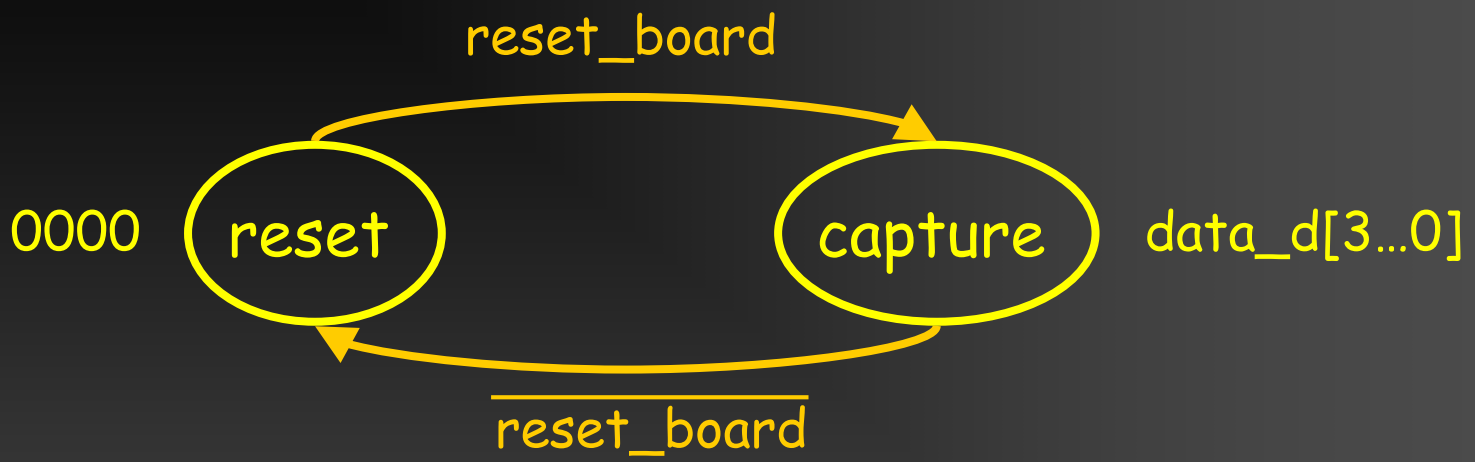
© A. Steininger / TU Wien

Beispieldesign: Log/iC

```
  <*Identification ... >
*Declarations
  X-Var = 5;
  Y-Var = 4;
*X-Names
  reset_board;
  data_d[0..3];
*Y-Name
  digit_L_d[0..3];
*Boolean Equations
  digit_L_d[0..3]:=data_d[0..3] & reset_board;
    <*PAL ...   *Pins ... >
*Special Functions
  digit_L_d[0..3].REG = YES;
*END
```


© A. Steininger / TU Wien

Beispieldesign: State-Chart



Zustand	Ausgabe digit_L_d[3…0]
reset	0000
capture	data_d[3…0]

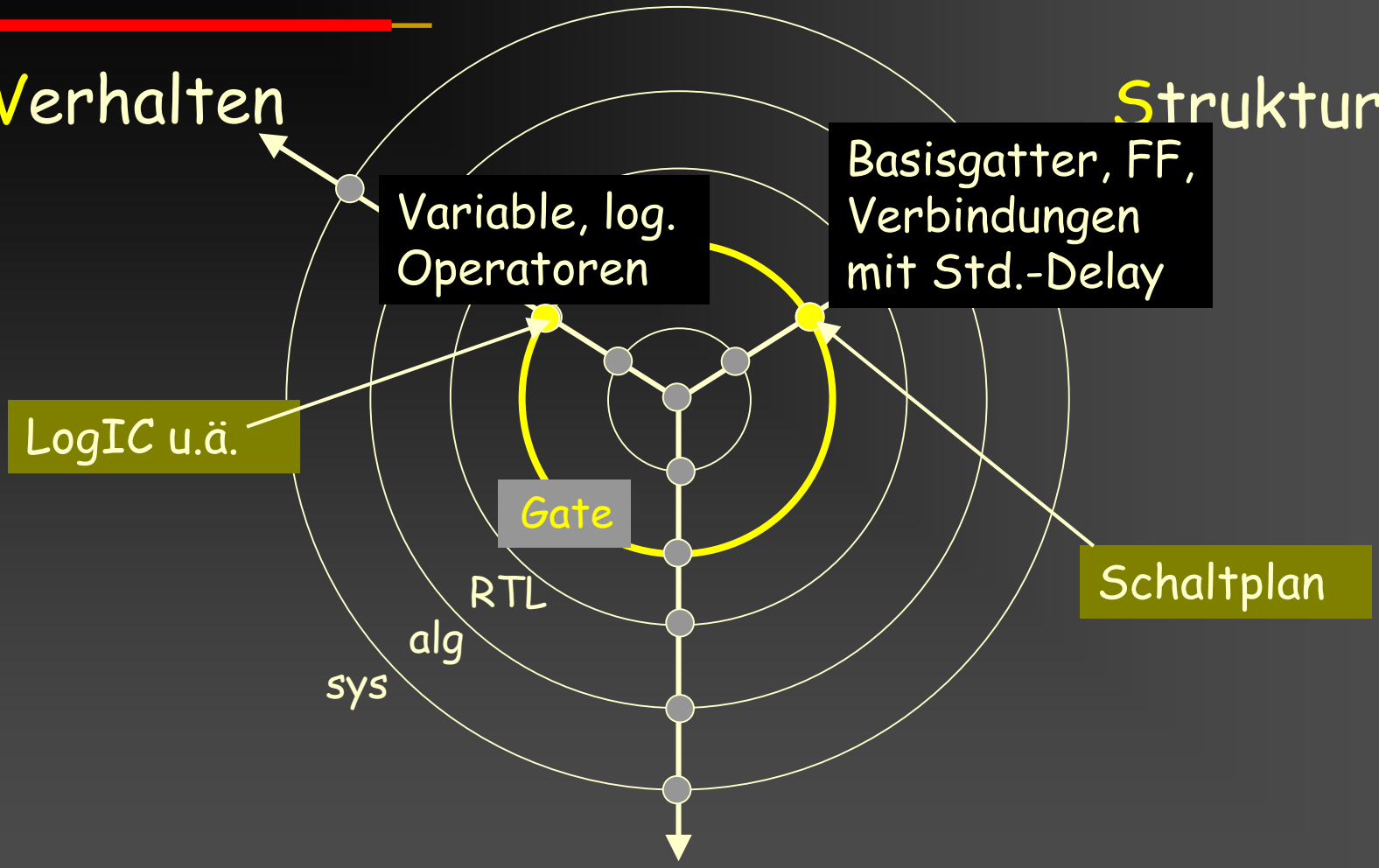
© A. Steininger / TU Wien

Beispieldesign: VHDL-Code

```
architecture behaviour of digit is
  begin
    SYNC_DIGIT_P : process (clk)
    begin
      if clk'event and clk = '1' then
        if reset_board = '0' then
          digit_L_d <= "0000";
        else
          digit_L_d <= data_d;
        end if;
      end if;
    end process SYNC_DIGIT_P;
end behaviour;
```

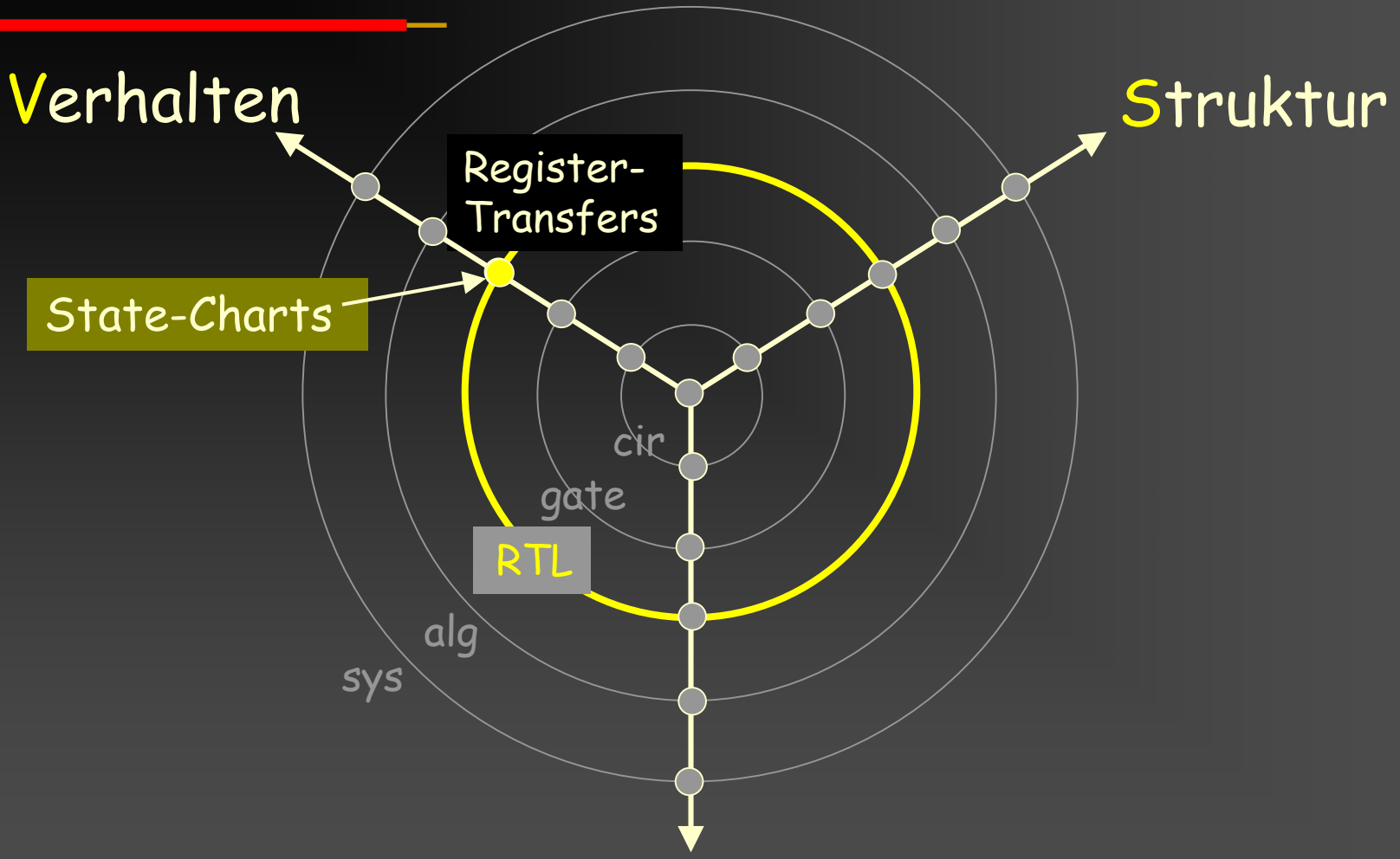

© A. Steininger / TU Wien

Design-Entry im Vergleich (1)



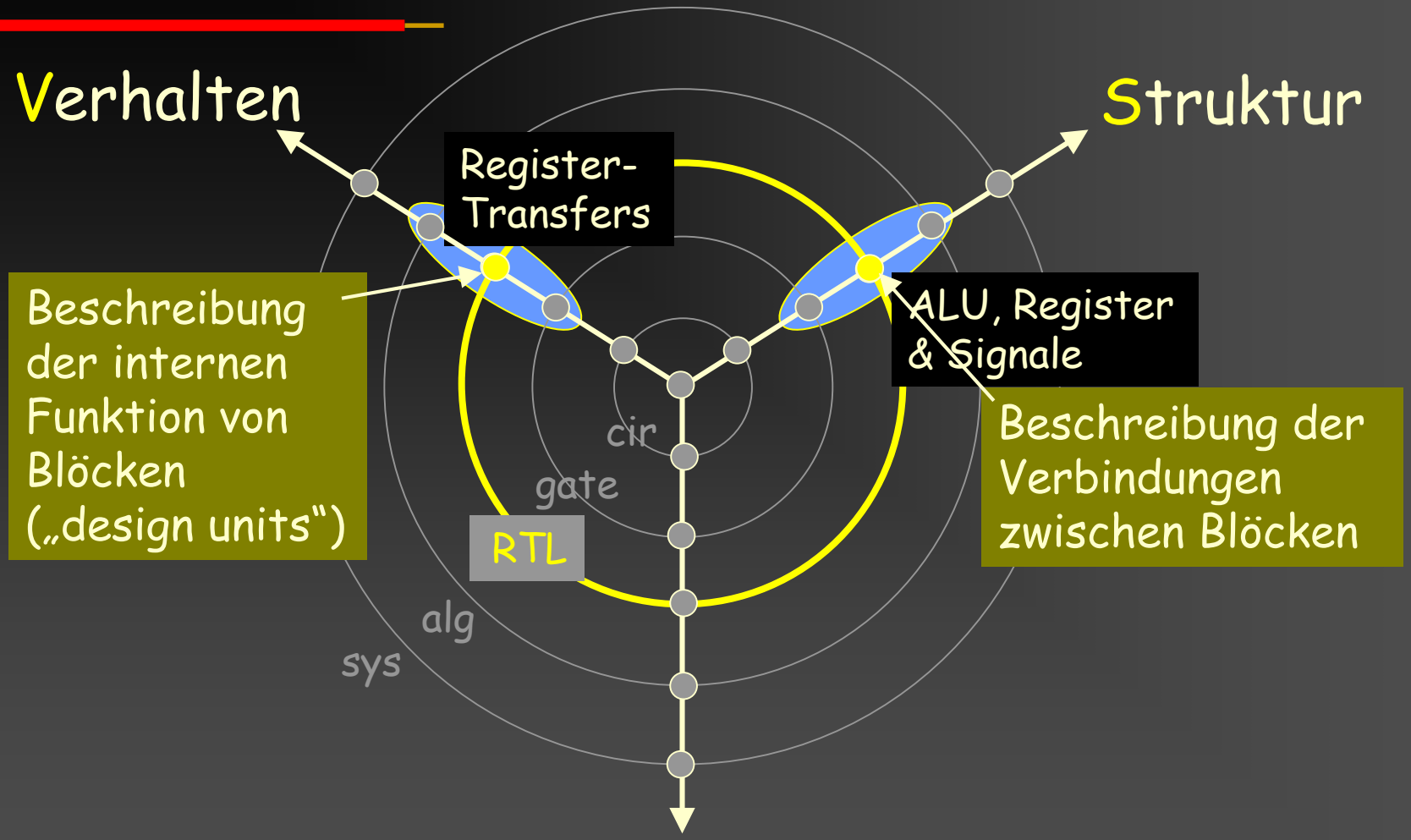
© A. Steininger / TU Wien

Design-Entry im Vergleich (2)



© A. Steininger / TU Wien

VHDL-Entry im Y-Diagramm



© A. Steininger / TU Wien

Compilation (Logic Optimization)

- Zweck:

  Umsetzung der

  verhaltensbasierten Beschreibung

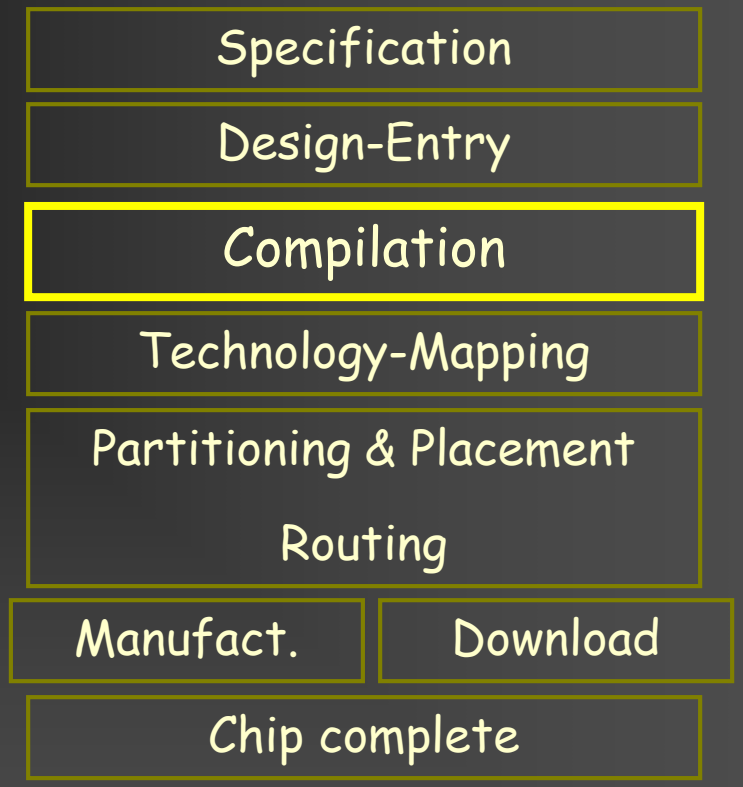
  (z.B. HDL Code)

  in eine

  strukturelle Darstellung

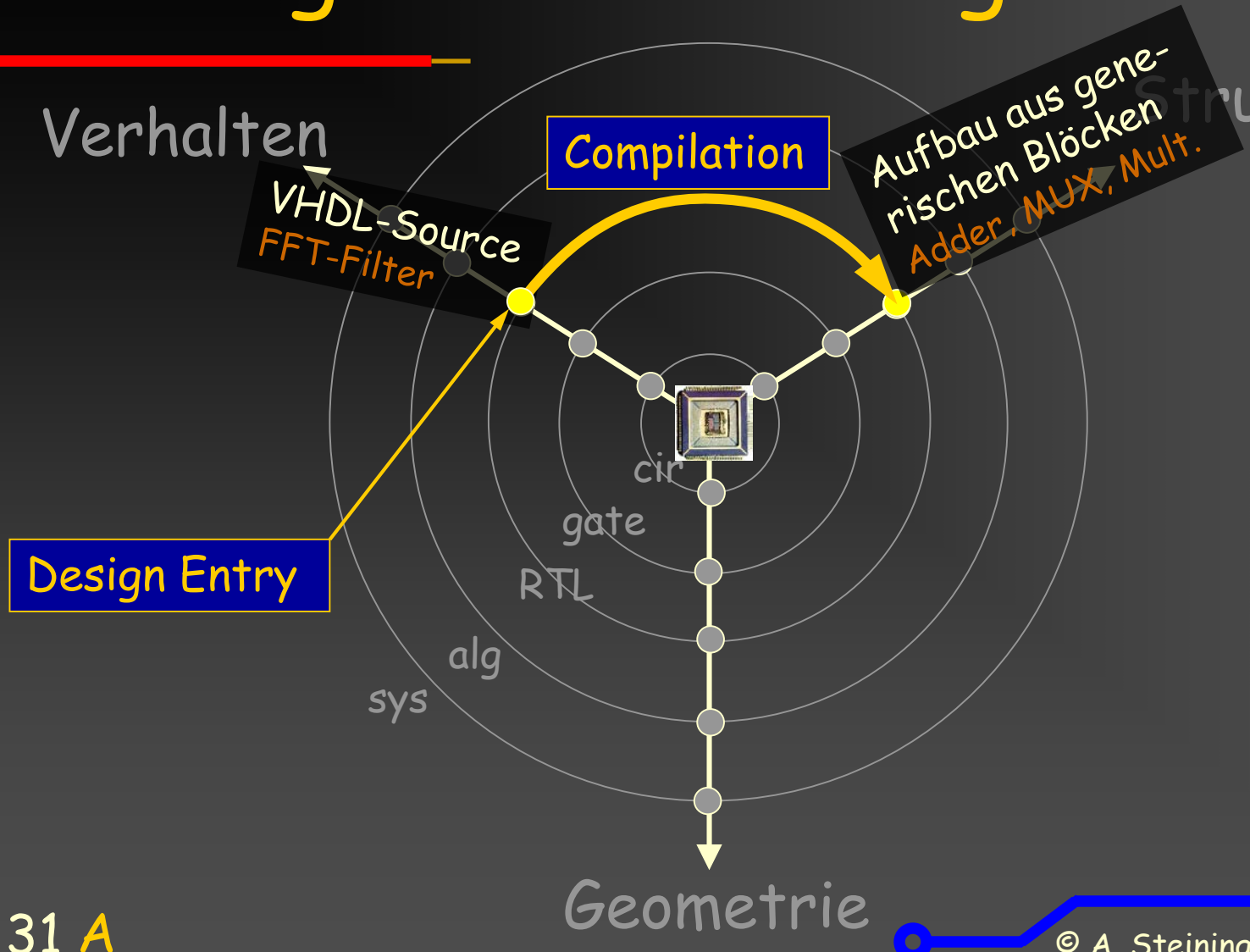
  (z.B. EDIF-Netzliste)





© A. Steininger / TU Wien

Y-Diagr.: FPGA Design Flow



© A. Steininger / TU Wien

Compilation: Schritte

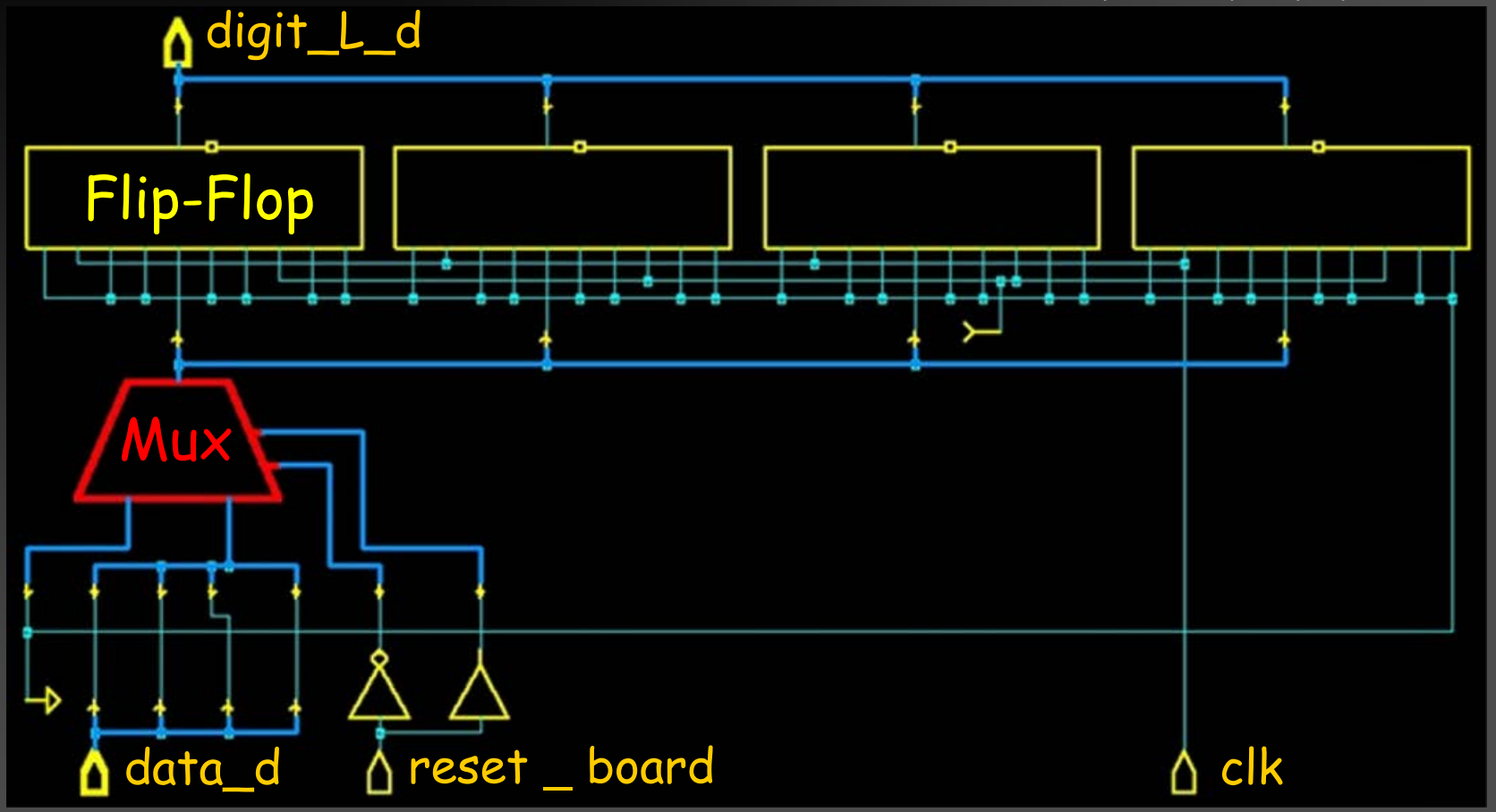


- Analyse des Designs/HDL-Codes (vgl. Parsing), syntaktische Prüfung
- Elaboration: Umwandlung in eine Datenstruktur
- Transformation in ein Netzwerk aus generischen logischen Zellen (AND, INV, ...)
- Minimierung der Logik (vgl. KV-Diagramm)
- Timing-Analyse basierend auf „unit-delay"

© A. Steininger / TU Wien

Ergebnis der Compilation

Beispiel: Synopsys



© A. Steininger / TU Wien

Vergleich mit VHDL-Code

```
architecture behaviour of digit is
  begin
    SYNC_DIGIT_P : process (clk)
    begin
      if clk'event and clk = '1' then
        if reset_board = '0' then
          digit_L_d <= "0000";
        else
          digit_L_d <= data_d;
        end if;
      end if;
    end process SYNC_DIGIT_P;
end behaviour;
```

Flip-Flops

Mux

© A. Steininger / TU Wien

Compiler: SW vs. HW

Software

- Compilertechnik ist extrem ausgereift
- Hochsprache ist für Programmierung konzipiert worden
- alle Hochsprachen-konstrukte sind in Assemblercode darstellbar

Hardware

- Compiler sind wenig ausgereift
- VHDL ist konzipiert für Dokumentation
  Verilog für Simulation
- Nur wenige HDL-Konstrukte sind auch implementierbar

© A. Steininger / TU Wien

Technology Mapping

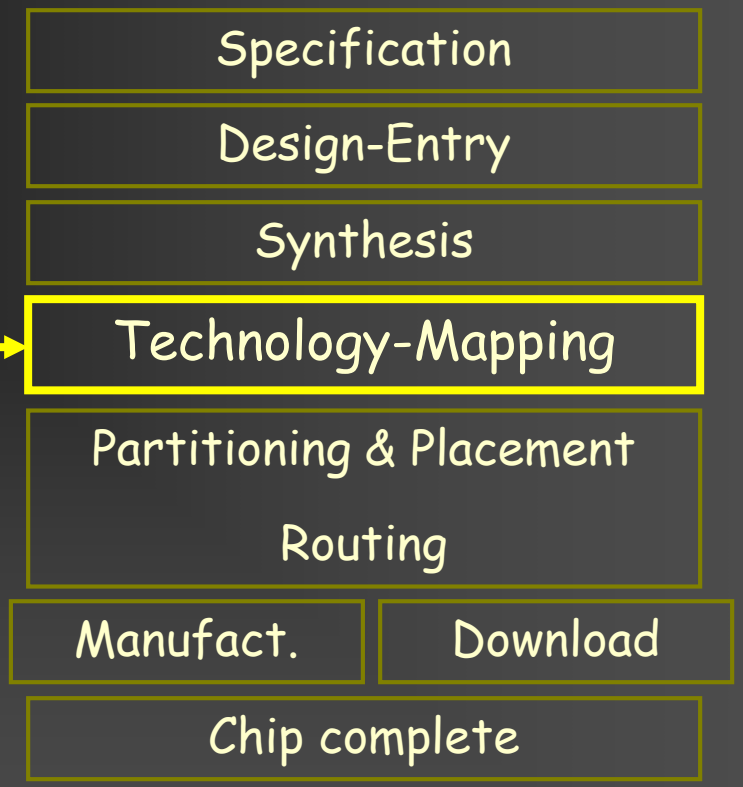
- **Zweck:**

  Abbildung der generischen Gatter auf die Logikelemente der Zieltechnologie

- **Besonderheit:**

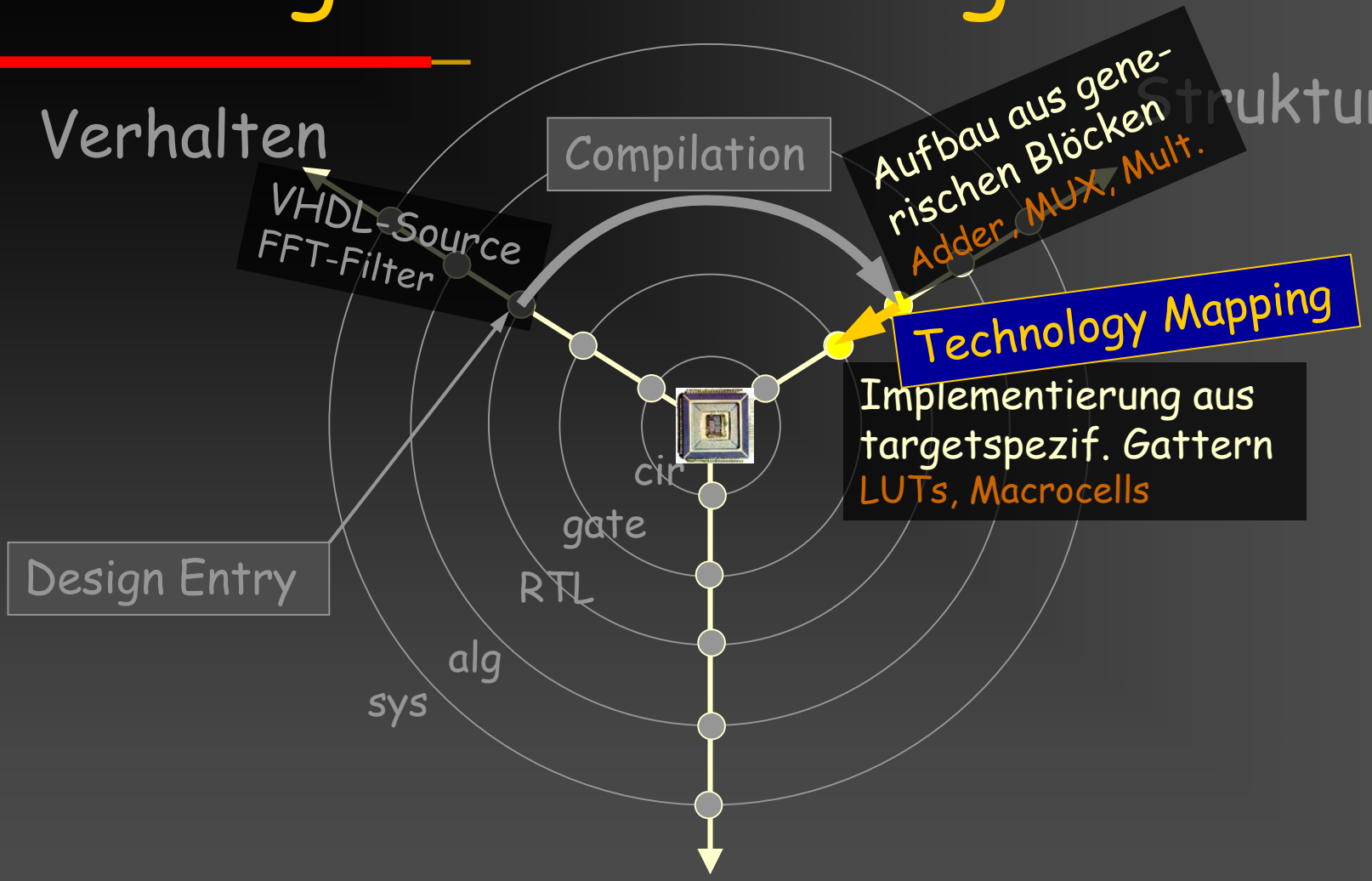
  In diesem Schritt wird Zieltechnologie festgelegt





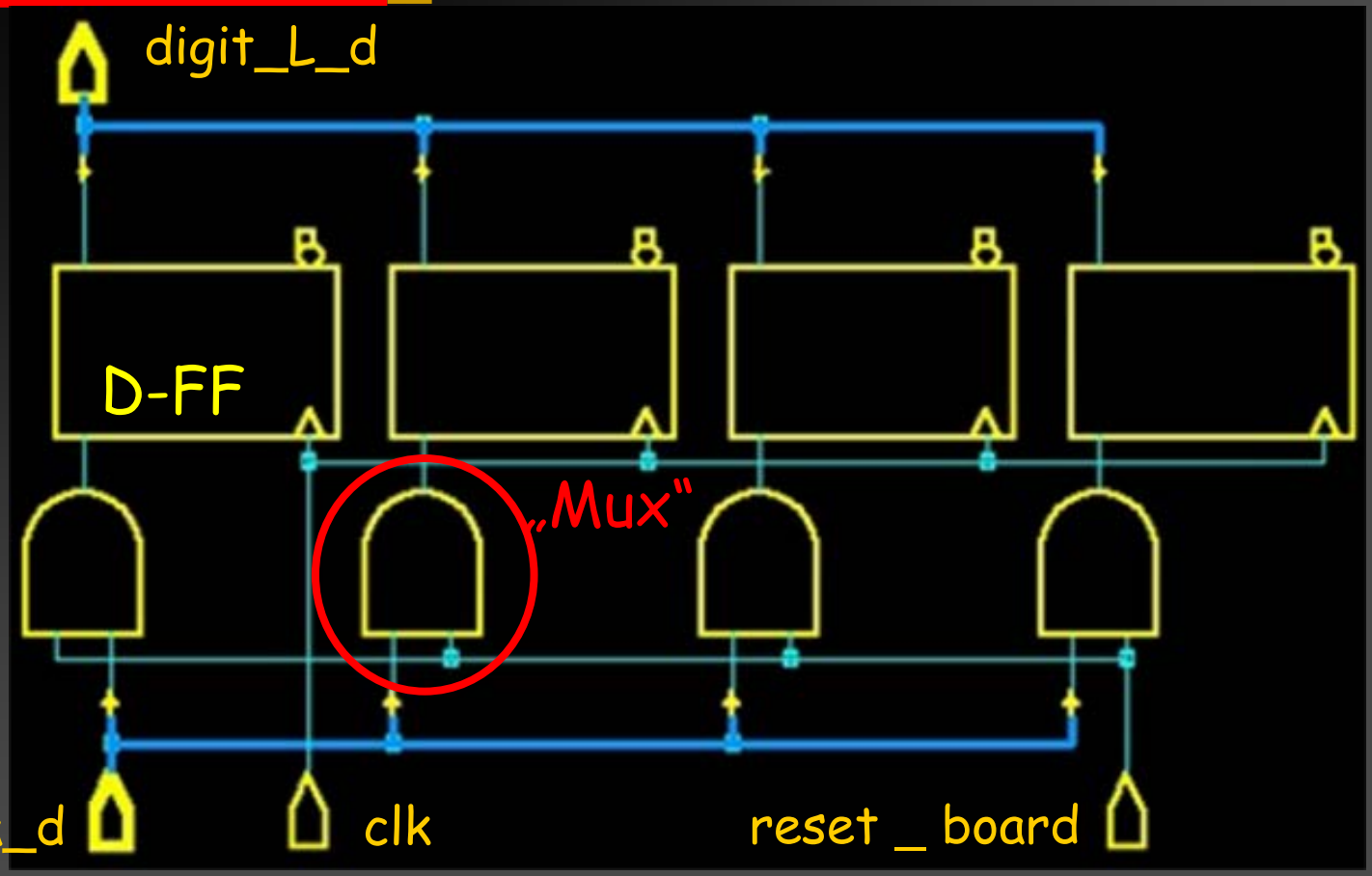
© A. Steininger / TU Wien

Y-Diagr.: FPGA Design Flow



© A. Steininger / TU Wien

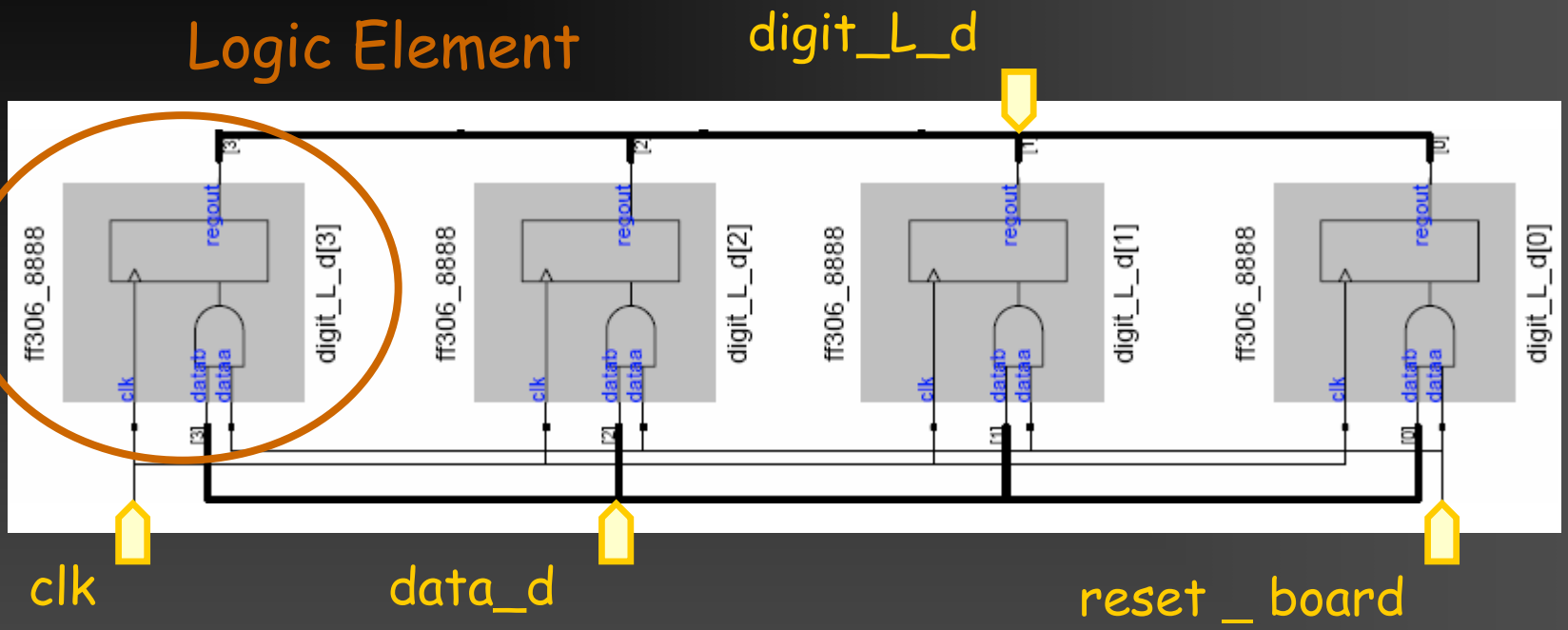
Technology Mapping: ASIC



Beispiel: Synopsys für Zieltechnologie Standard-Lib.

© A. Steininger / TU Wien

Technology Mapping: FPGA



Beispiel: Synplify für Zieltechnologie Altera Stratix

© A. Steininger / TU Wien

Synthese

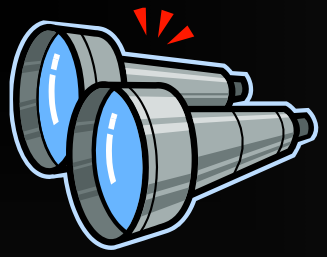


- Compilation
  + Optimierung
- Technology Mapping
  + Optimierung

Synthesis

Synthese ist also der Prozeß der Abbildung der beim Design-Entry angegebenen Beschreibung auf Elemente der Target-Library.

© A. Steininger / TU Wien



Überblick

- Das Y-Diagramm
- Design-Schritte
- **Synthese und PPR als Optimierungen**
- Spezifische Probleme des Routing
- Verifikations-Schritte
- Simulation
- Statische Timing-Analyse

© A. Steininger / TU Wien

Prinzip einer Optimierung

Mittels eines Algorithmus soll

eine Kostenfunktion minimiert oder
eine Nutzenfunktion maximiert werden

und zwar unter

Einhaltung von Randbedingungen

Als Voraussetzung müssen daher

- Kosten / Nutzen meßbar und
- alle Randbedingungen bekannt sein

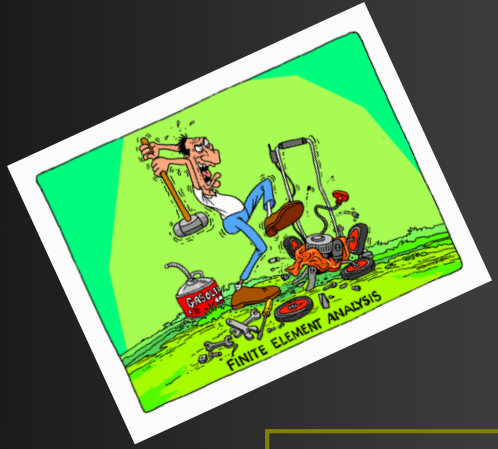
© A. Steininger / TU Wien

Synthese als Optimierung

- mögliche Kosten- bzw. Nutzenfunktionen (Optimierungskriterium OK)
  - maximum speed
  - minimum area
  - minimum effort
- typische Randbedingungen (RB)
  - Timing-Vorgaben für diverse Pfade
  - Power-Consumption
  - Sperren von Optimierungen

© A. Steininger / TU Wien

Partitioning

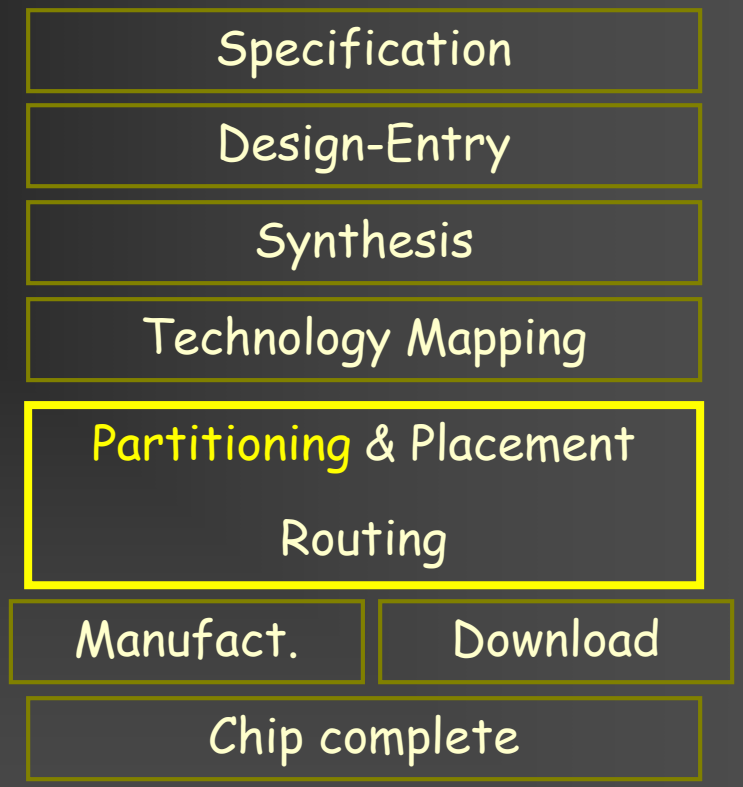


- Zweck:

  optimale Aufteilung des Designs auf mehrere ASICs (falls nötig)

  OK: minimale Anzahl von Verbindungen

  RB: Größe, Speed, Verlustleistung, am Chip verfügbare Technologie,...



© A. Steininger / TU Wien

Partitioning — Strategien



- Konstruktiv
  Startzelle — jede weitere dazuprobieren, beste belassen — weitere dazuprobieren, ... bis ASIC voll ist
- Iterativ
  "Seed"-Konfiguration — Zellen einzeln zwischen ASICs vertauschen, nur im Fall einer Verbesserung belassen
- Simulated Annealing
  wie „iterativ", aber manchmal auch ohne Verbesserung belassen, => Überwinden lokaler Minima
- ...

$$P_{belassen} = \exp\left(-\frac{\Delta E}{T}\right)$$

ΔE → Verschlechterung

T → Temperatur

© A. Steininger / TU Wien

Placement

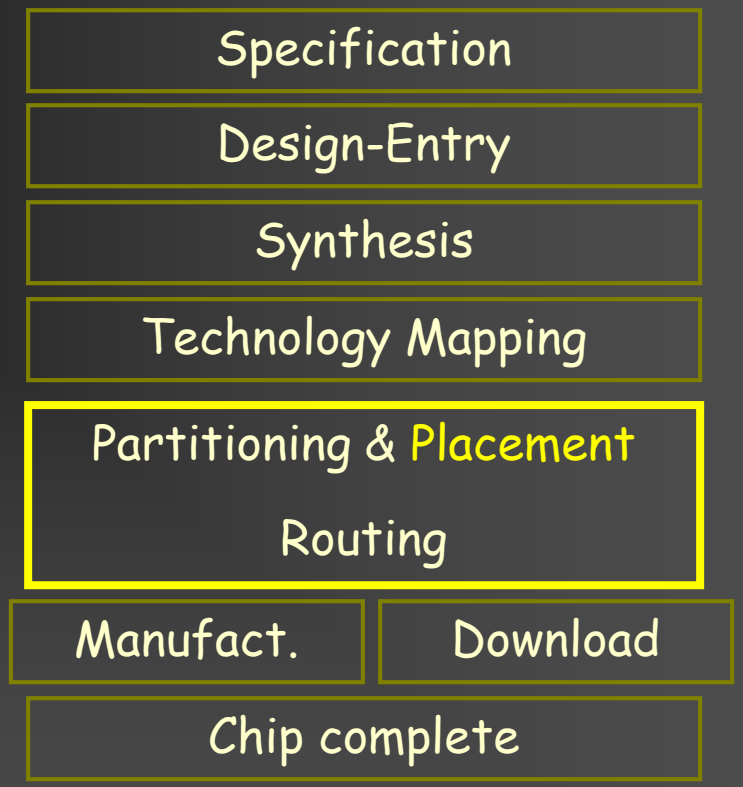


- Zweck:

  optimale Verteilung der Logikelemente und Routing-Kanäle auf dem ASIC

  OK: Minimaler Inter-connect-Delay

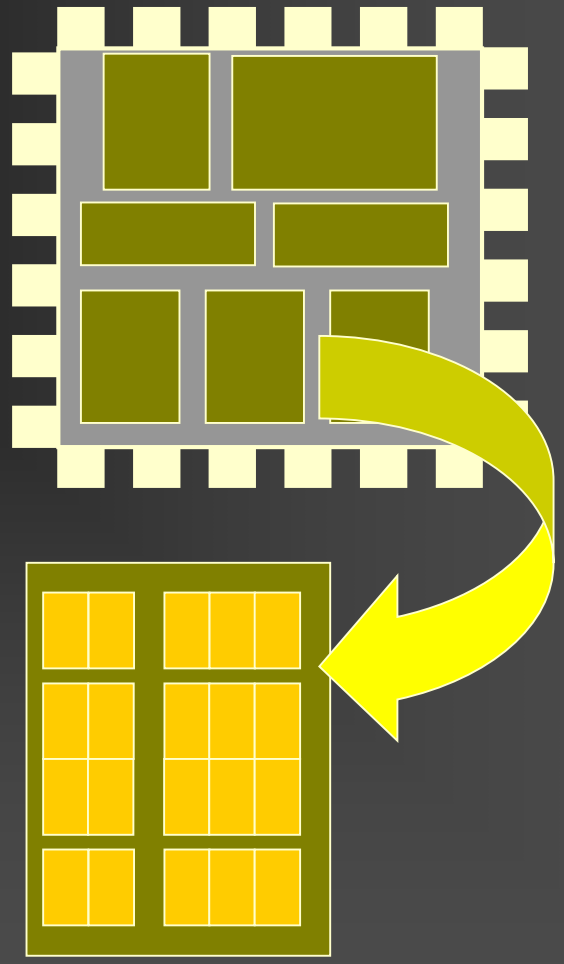
  RB: Größe, Pinbelegung, „Constraints" des Designers, ...



© A. Steininger / TU Wien

Placement: Schritte

- **Floorplanning**
  Anordnen von Funktionsblöcken des Designs im ASIC
  (vgl. Räume im Gebäude)
- **Placement**
  (im engeren Sinn)
  Anordnen der Logik-Elemente innerhalb eines Funktionsblockes
  (vgl. Einrichtung der Räume)



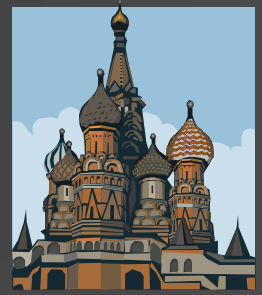
© A. Steininger / TU Wien

Placement: Das Grundproblem

- Optimierung erfordert Abschätzen des Interconnect-Delay noch VOR dem Routing

  Vergleich: Wie lange dauert eine Reise von China nach Russland?
  - Von wo in China?
  - Nach wohin in Rußland?
  - Welche Straßen gibt es?





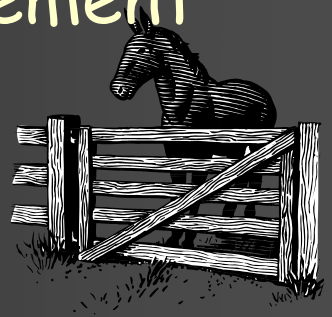
- Lösung: Heuristik (Erfahrungen & Statistiken aus bestehenden Designs)

© A. Steininger / TU Wien

Komponenten des Delay



- Gate Delay
  - ✓ Durchlaufzeit durch ein Logikelement
  - ✓ kaum abhängig vom Routing
  - ✓ relativ gut vorhersagbar



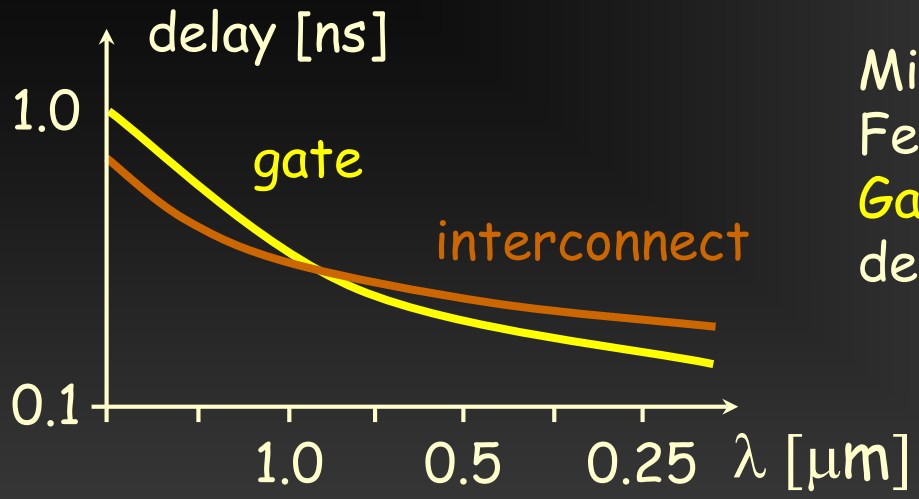
- Interconnect Delay
  - ✓ Signallaufzeit auf den Leitungen
  - ☹ stark abhängig vom Routing
  - ☹ schlecht vorhersagbar



© A. Steininger / TU Wien

Trends beim Delay

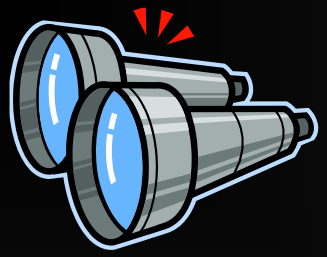




Mit Verkleinerung der Feature-Size sinkt der Gate Delay rascher als der Interconnect Delay

Bei den heute üblichen Technologien überwiegt der Interconnect Delay klar. Eine realistische Vorhersage des Timings ist daher erst nach dem Routing möglich, Optimierungen vor dem Routing werden immer schwieriger.

© A. Steininger / TU Wien



Überblick

- Das Y-Diagramm
- Design-Schritte
- Synthese und PPR als Optimierungen
- **Spezifische Probleme des Routing**
- Verifikations-Schritte
- Simulation
- Statische Timing-Analyse

© A. Steininger / TU Wien

Routing

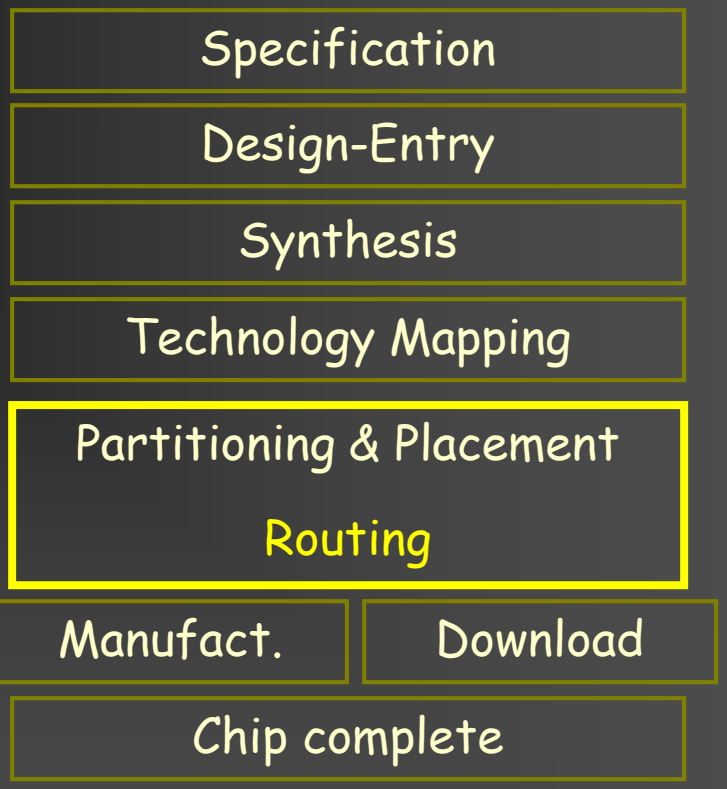


- Zweck:

  Herstellen aller nötigen Verbindungen auf dem ASIC

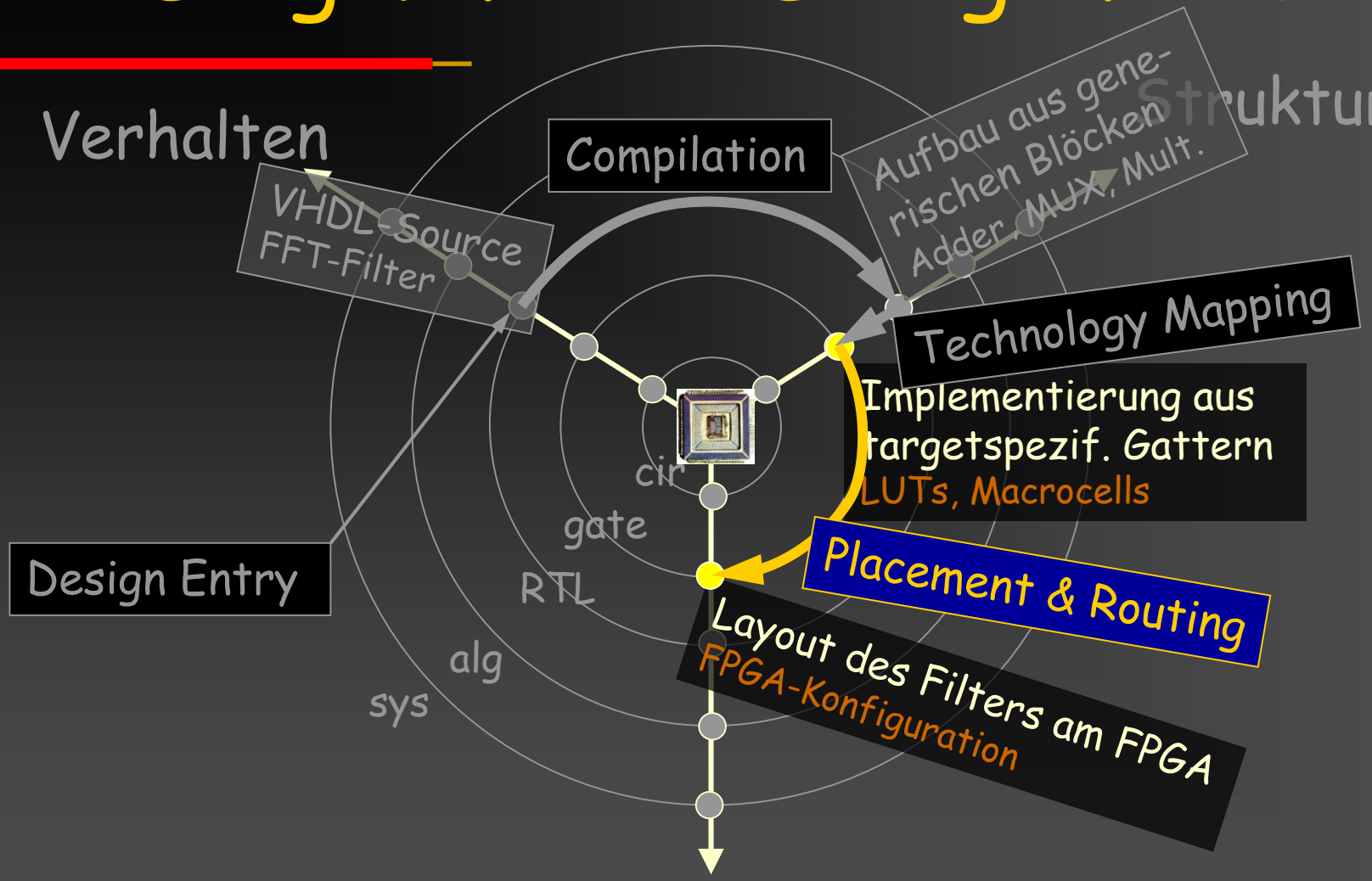
  OK: Minimale Inter-connect-Länge, minimaler krit. Pfad, Minimum an Vias

  RB: Verfügbarkeit von Leitungen/Kanälen, Constraints, ...



© A. Steininger / TU Wien

Y-Diagr.: FPGA Design Flow



© A. Steininger / TU Wien

Umfang des Interconnect



- Stand 2007 [ITRS'06]
  - Der Interconnect in einem Chip umfasst typisch etwa 1,5km Leitungen pro cm^2.
  - Prognose: Anstieg von 15% pro Jahr
  - Es gibt 11 Metallisierungsebenen
  - „half pitch"
    = Abstand der Leiterbahnen
    = Breite der Leiterbahnen = 65nm

© A. Steininger / TU Wien

Routing des Clock

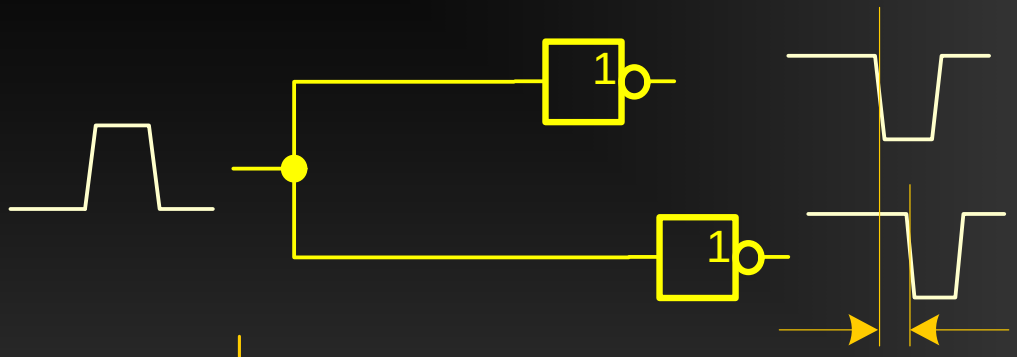


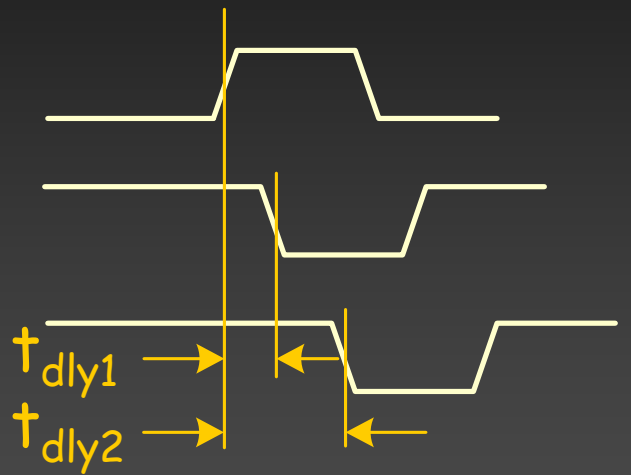
- Mimimaler Delay
  - Starke Treiber (größere Fläche)
  - Treiber parallel (geringere Last)
  - Kurze Verbindungen
- Minimaler Skew
  - Symmetrische Netze („Tree", „Spine")
  - „gematchte" gleich belastete Treiber

➡ Vorgegebene Clock-Netze im FPGA

© A. Steininger / TU Wien

Was ist Skew ?





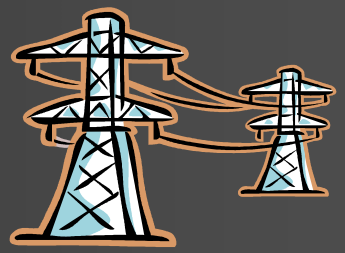
$t_{skew} = |t_{dly1} - t_{dly2}|$

Skew ist der Unterschied im Signal-Delay

- entlang unterschiedl. Äste einer verzweigten Leitung
- entlang eines Busses
- durch gleichartige Gatter

© A. Steininger / TU Wien

Routing der Versorgung



- Stromdichte

  Stromdichte J = Strom/Querschnitt [A/cm^2]

  Zu hohe Stromdichte führt zu inakzeptabler Defektrate im Betrieb durch Abwandern des Materials („Elektromigration")

  Typ. Wert: $J \approx 1mA/(\mu m)^2$

- „Fat metal Rules"

  zu dicke Metallflächen (Leitungen) neigen zum Ablösen während des Packaging

© A. Steininger / TU Wien

Stromdichte – ein Vergleich

Gegeben: Standard-Installationsdraht mit Querschnitt 2,5mm^2.

Gesucht: Welcher Strom muss durch diesen Draht fließen damit sich die gleiche Stromdichte ergibt wie in der Versorgungsleitung eines ASIC ? (1mA /(μm)^2)

Lösung: Querschnitt = 2,5mm^2 = 2,5*10^6 (μm)^2

Strom = Querschnitt * Stromdichte

Es fließt ein Strom von 2,5*10^6 mA = 2,5kA

Das entspricht mehr als 0,5MW bei 230V



© A. Steininger / TU Wien

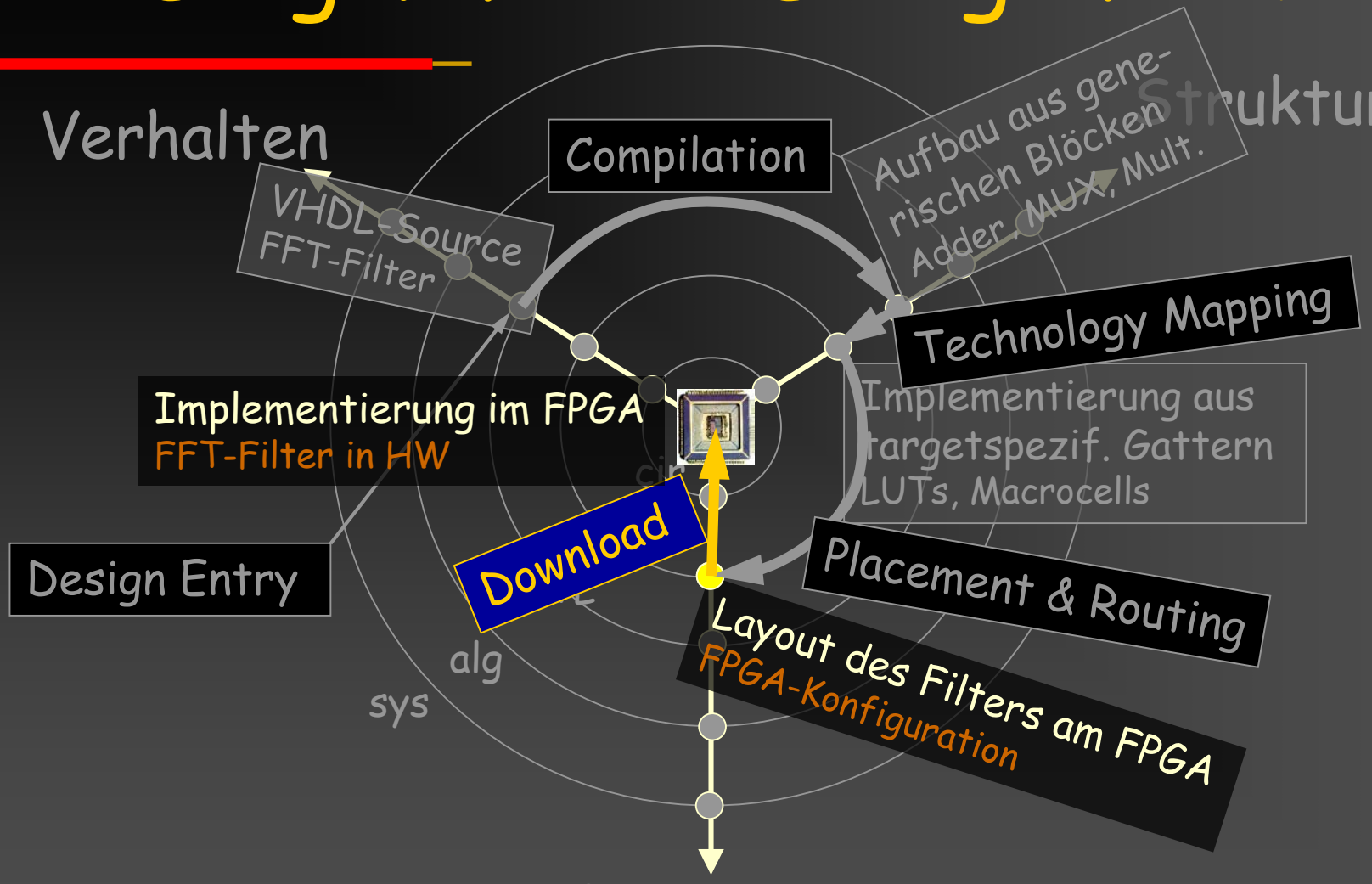
Ergebnisse nach dem Routing

- Position aller benötigten Logikelemente steht fest (Placement)
- Layout aller Logikelemente ist bekannt (Library)
- Alle Verbindungen sind gelegt (Routing)

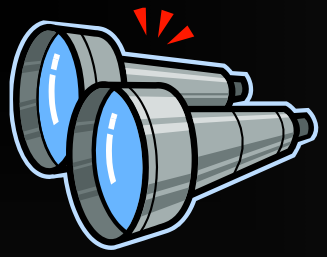
- ✓ Alle Masken können erstellt werden („Tape-out").
- ✓ Alle realen Delays können genau (≈5%) ermittelt und dem Simulator mitgeteilt werden („Back-Annotation")

© A. Steininger / TU Wien

Y-Diagr.: FPGA Design Flow



© A. Steininger / TU Wien



Überblick

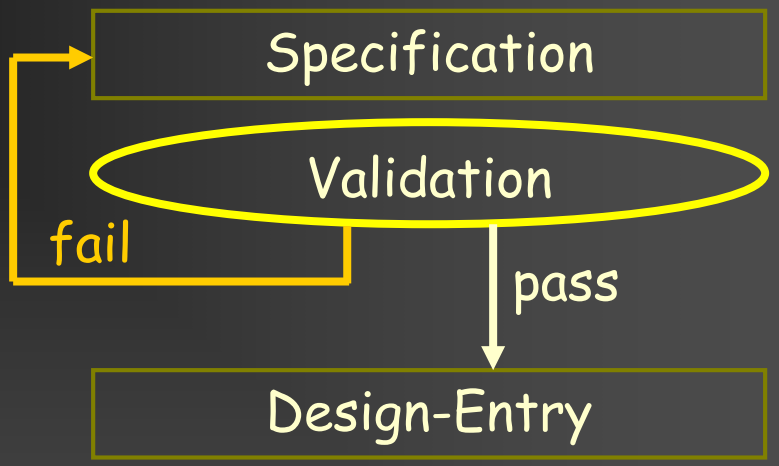
- Das Y-Diagramm
- Design-Schritte
- Synthese und PPR als Optimierungen
- Spezifische Probleme des Routing
- **Verifikations-Schritte**
- Simulation
- Statische Timing-Analyse

© A. Steininger / TU Wien

Validation

- Zweck:

  überprüfen der Spezifikation
  - Beschreibt sie die angestrebte Funk-tionalität adäquat ?
  - Ist sie umsetzbar ?
  - Enthält sie Widersprüche ?



© A. Steininger / TU Wien

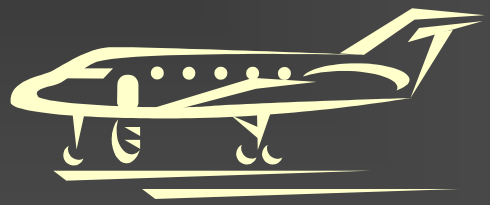
Simulation – Wozu?



Debugging eines physikal. Prototypen wäre

- zu kompliziert/unmöglich (Zugänglichkeit im Chip?)
- zu teuer (Fertigungskosten für Prototyp)
- zu spät (time-to-market!)

Das ist bei Software (leider) oft anders…



(siehe Clip „If SW-Programmers had to build planes")

© A. Steininger / TU Wien

Wert eines Prototypen



- Erfolgserlebnis
- gute Basis für ausführliche Tests in Echtzeit (= schneller als Simulation)
- Indiz für das Funktionieren des Designs
- ausreichend für den „Eigenbedarf“
- NICHT AUSREICHEND für ein industrielles Produkt (worst case bezügl. Temperaturbereich, Prozessvariationen?)

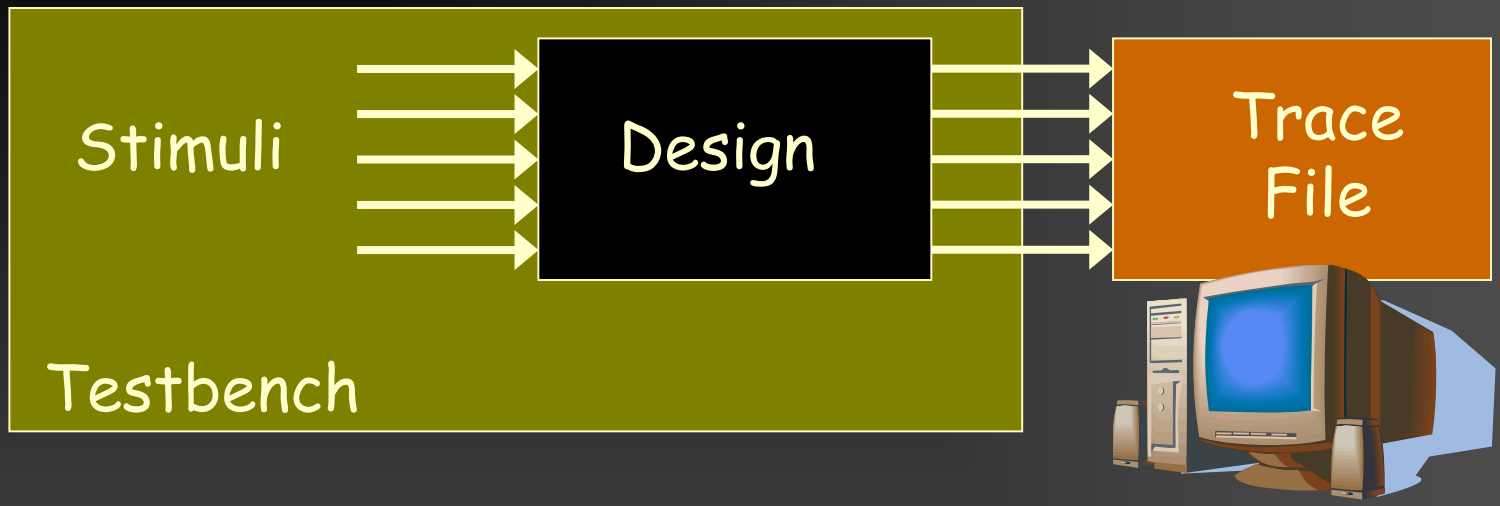
© A. Steininger / TU Wien

Ablauf einer Simulation

- Anlegen von Stimuli an das Design
  - Meist wird dazu ein eigenes (virtuelles) Design - eine „Testbench" – erstellt.
  - Eine Testbench für alle Simulationsebenen.
- Überprüfen der Reaktion
  - Entspricht das Verhalten der Spezifikation ?
  - Trace-File von voriger Simulation als Referenz
- War die Simulation vollständig ?
  - Ein positives Ergebnis bezieht sich nur auf die simulierten Funktionen !

© A. Steininger / TU Wien

Simulation & Testbench



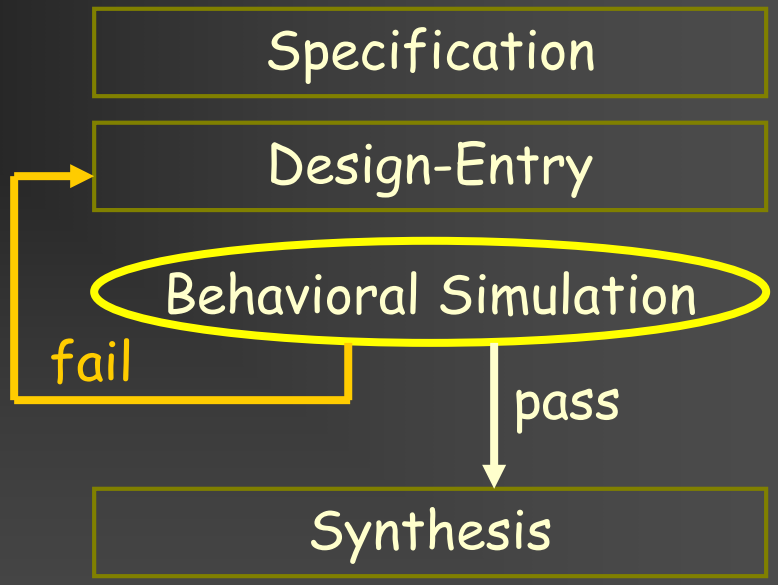
Design wird als Modul in größeres Design (Testbench) eingebettet und erhält so Stimuli für die Eingänge bei der Simulation

© A. Steininger / TU Wien

Behavioral Simulation



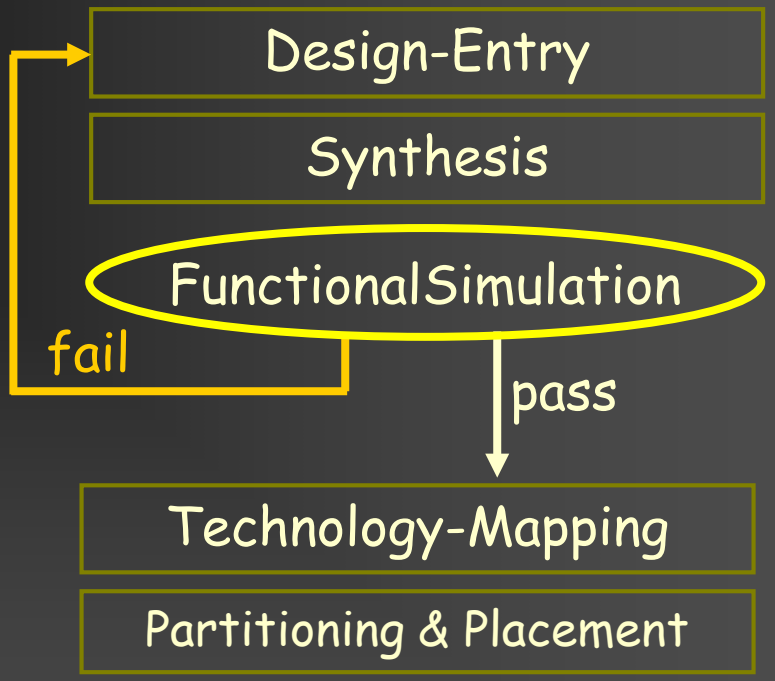
- Zweck:
  - Wurde Spezifikation richtig umgesetzt ?
  - Funktioniert die Testbench ?
- Besonderheit:
  - berücksichtigt keinerlei Timing
  - berücksichtigt nicht HW-Realisierbarkeit



© A. Steininger / TU Wien

Functional Simulation

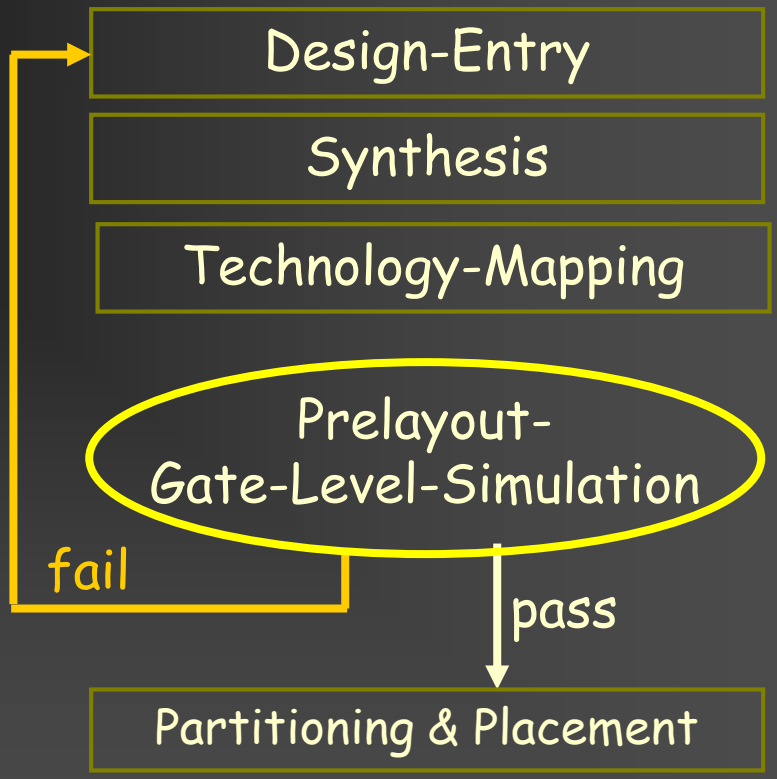
- Zweck:
  - Wurde der Code richtig interpretiert?
- Besonderheit:
  - Design besteht aus generischen Gattern mit „unit-Delay“



© A. Steininger / TU Wien

Prelayout-Gate-Level Simul.

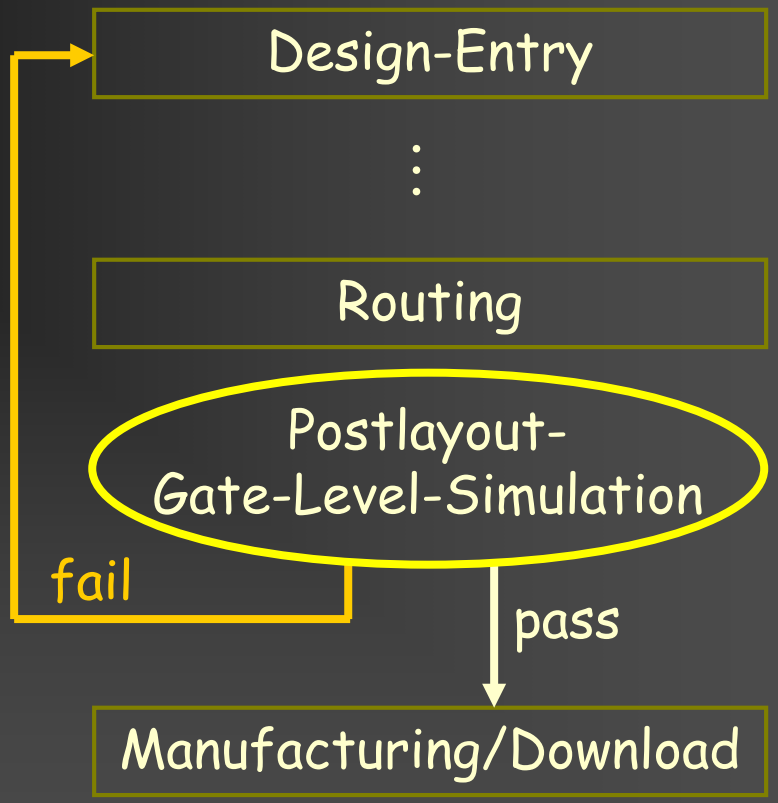
- Zweck:
  - Gab es Fehler beim Technology-Mapping?
- Besonderheit:
  - Design besteht aus Gattern der Zieltechnologie
  - Gatterdelay real, Routing-Delay grob geschätzt



© A. Steininger / TU Wien

Postlayout-Gate-Level Simul.

- Zweck:
  - Stimmt das Timing?
- Besonderheit:
  - Die genauen Timing-Werte von der Back-Annotation stehen zur Verfügung



© A. Steininger / TU Wien

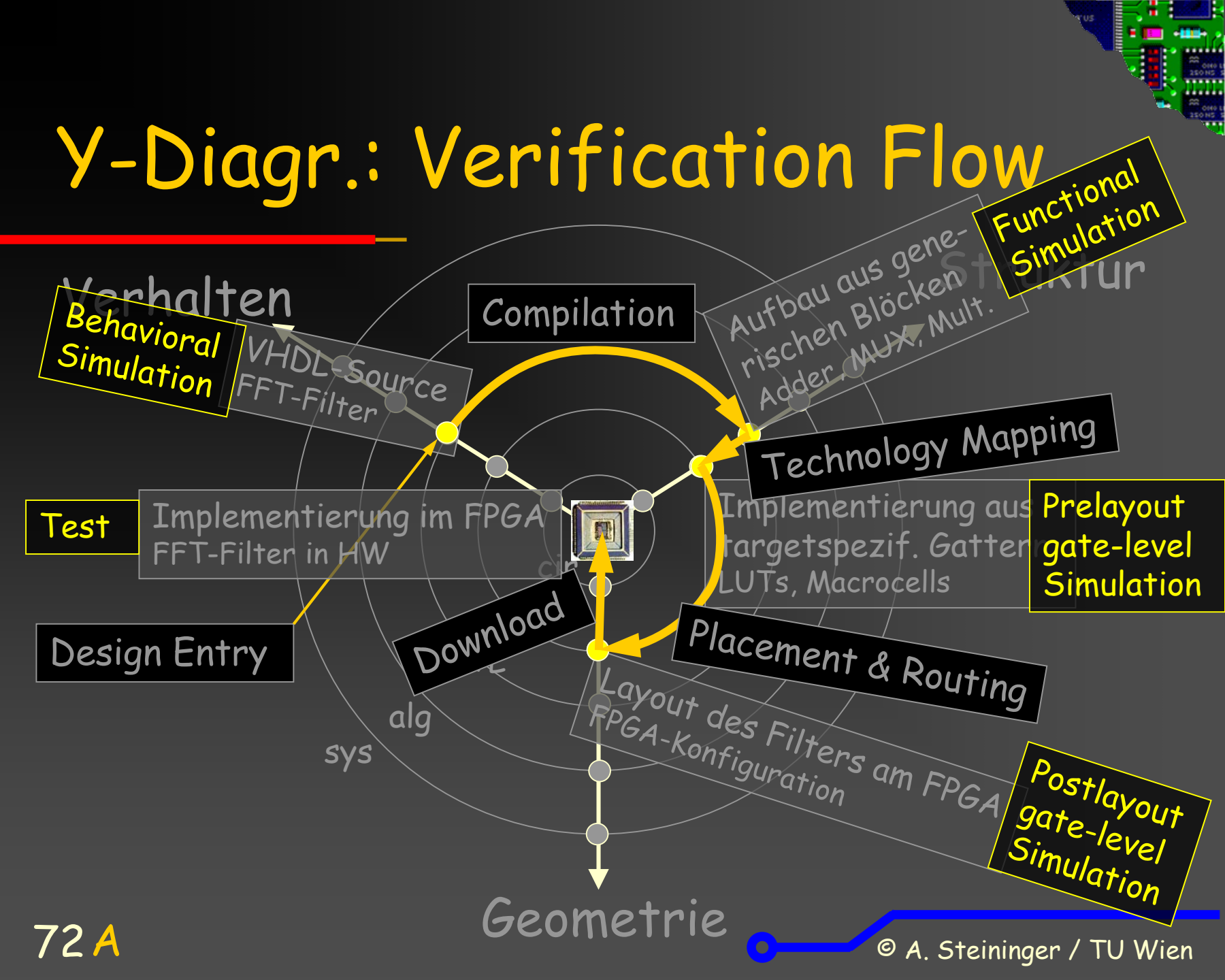
Simulation auf allen Ebenen?

Je feinstufiger die Simulationsebenen, desto

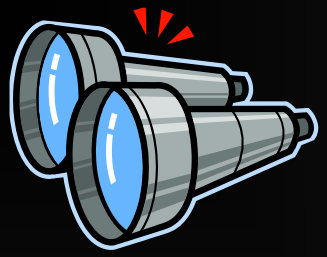
- klarer sind Fehler zuordenbar
- weniger Zeit (für Synthese, P&R etc.) wird bis zur Erkennung des Fehlers aufgewendet
- geringer ist die Gefahr einer gegenseitigen Maskierung zweier Fehler

© A. Steininger / TU Wien

Y-Diagr.: Verification Flow



© A. Steininger / TU Wien



Überblick

- Das Y-Diagramm
- Design-Schritte
- Synthese und PPR als Optimierungen
- Spezifische Probleme des Routing
- Verifikations-Schritte
- **Simulation**
- Statische Timing-Analyse

© A. Steininger / TU Wien

Terminologie zur Simulation

- **Mixed-level Simulation**

  Funktionsblöcke werden auf unterschiedlichen Abstraktionsebenen simuliert

- **Mixed mode Simulator**

  Simulation erfolgt gemischt analog/digital

- **Sign-off Simulation**

  typisch postlayout Gate-Level-Simulation bei Übergabe zur Fertigung, als Vorlage für Funktionstest des Chips nach der Fertigung

© A. Steininger / TU Wien

Logikpegel bei der Simulation

Was passiert, wenn auf einer Leitung gleichzeitig zwei widersprüchliche Pegel auftreten:

- zwei Treiber gleichzeitig aktiv (Fehler)
- ein Treiber zieht Leitung trotz Pull-up auf ´0´
- Treiber "overruled" den Bus-Keeper
- ...

die beiden logischen Pegel ´1´ und´0´ allein bieten dem Simulator zu wenig Möglichkeiten, solche Situationen aufzulösen

© A. Steininger / TU Wien

9-wertige Logik (IEEE Std 1164-1993)

0	strong low	*Treiberausgang, definiert*
1	strong high	*"*
L	weak low	*Pull-down*
H	weak high	*Pull-up*
X	strong unknown	*Treiberausgang, undef´d.*
W	weak unknown	*bus-keeper, uninitialisiert*
Z	high impedance	*Treiberausgang, tri-state*
-	don't care	*Pegel bedeutungslos*
U	uninitialized	*FF-Ausgang, uninitialisiert*

© A. Steininger / TU Wien

Signal Resolution Table



AND	U	X	0	1	Z	W	L	H	-
U	U	U	0	U	U	U	0	U	U
X	U	X	0	X	X	X	0	X	X
0	0	0	0	0	0	0	0	0	0
1	U	X	0	1	X	X	0	1	X
Z	U	X	0	X	X	X	0	X	X
W	U	X	0	X	X	X	0	X	X
L	0	0	0	0	0	0	0	0	0
H	U	X	0	1	X	X	0	1	X
-	U	X	0	X	X	X	0	X	X

© A. Steininger / TU Wien

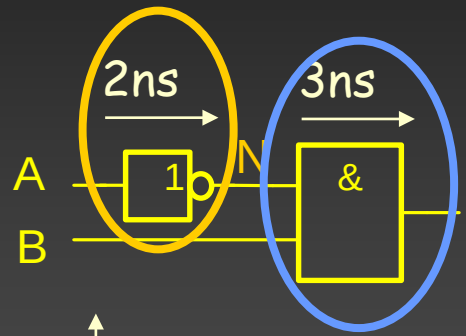
Ereignisgesteuerte Simulation

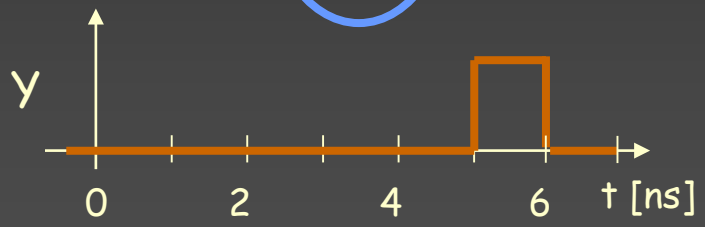
- Ereignis (event): Pegeländerung an einem Knoten
- Ereignisliste enthält alle (aktuell bekannten) zukünftigen Ereignisse mit Zeitpunkt, Knoten und neuem Pegel
- aktuelle Zeit schreitet fort, bis sie mit Zeitpunkt des nächsten Ereignisses übereinstimmt
- es folgt Auswertung der Konsequenzen des Events: Aktualisierung der Eingänge, Eintragen weiterer Events (z.B. Aktualisieren der Ausgänge nach einem Delay)
- Iterationen für die Auswertung in "delta-time"
- danach Fortschreiten der aktuellen Zeit

© A. Steininger / TU Wien

Beispiel für eine Simulation

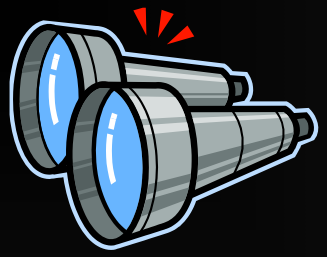






t	event	A	B	N	Y
0-	init	1	0	0	0
0	A ↓	0	0	0	0
0+Δ	B ↑	0	1	0	0
2	A ↑	1	1	0	0
2+Δ	N ↑	1	1	1	0
3	B ↓	1	0	1	0
3+Δ	Y ↓				0
4	N ↓	1	0	0	0
5	Y ↑	1	0	0	1
6	Y ↓	1	0	0	0

© A. Steininger / TU Wien



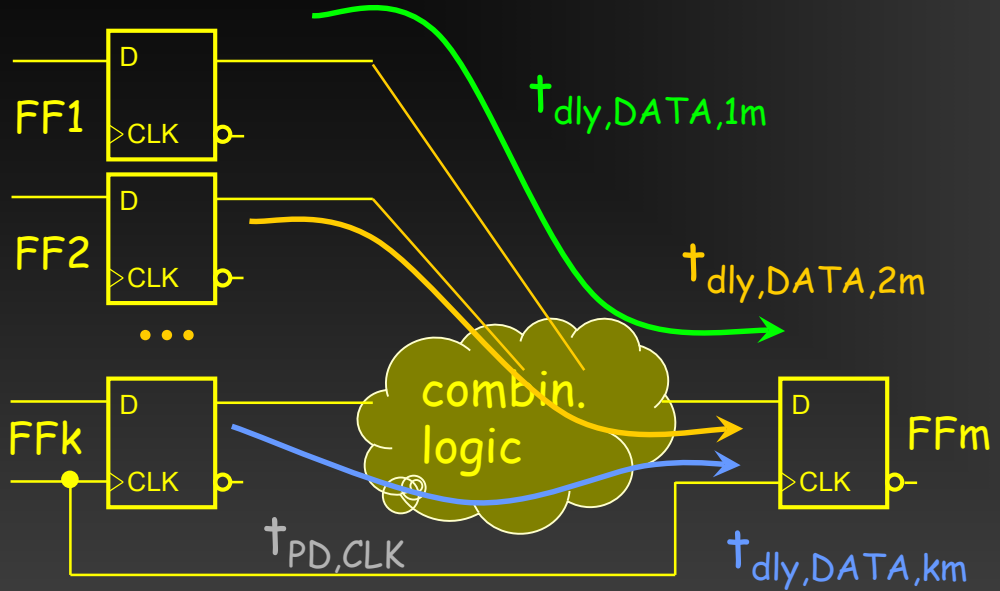
Überblick

- Das Y-Diagramm
- Design-Schritte
- Synthese und PPR als Optimierungen
- Spezifische Probleme des Routing
- Verifikations-Schritte
- Simulation
- **Statische Timing-Analyse**

© A. Steininger / TU Wien

Statische Timinganalyse





Delays im Daten- und Clock-Pfad werden für jedes Flip-Flop systematisch analysiert

$$1/f_{CLK,max} = \max (t_{dly,DATA,ij}) + t_{SU} - \min (t_{dlyD,CLK,i})$$

- ✓ Überprüfung vorgegebener Constraints
- ✓ Ermittlung der maximal zulässigen Taktfrequenz

© A. Steininger / TU Wien

Stat. Analyse vs. Simulation

Statische Analyse

... findet *sicher* den kritischen Pfad

... liefert *keine* Information über die zugehörigen Eingangsvektoren

... findet möglicherweise einen *ungültigen* Pfad

Simulation

... findet den kritischen Pfad *nicht sicher* (Eingangsvektoren?)

... liefert *automatisch* Information über die zugehörigen Eingangsvektoren

... findet sicher einen *gültigen* Pfad

© A. Steininger / TU Wien

Library-Databook /1



… charakterisiert die verfügbaren Basiszellen; z.B.:

Eingangskapazität [pF] (Beispiel: Inverter)

Typ (Treiberstärke)	inv1	inv2	inv4	inv8	inv12
flächenoptimiert	0.034	0.067	0.133	0.265	0.397
Performance-optimiert	0.145	0.292	0.584	1.169	1.753

Beiträge zur Kapazität: - Gate des p-Kanal-FET
- Gate des n-Kanal-FET
- internes Routing

© A. Steininger / TU Wien

Library-Databook /2

vorige Folie

Beispiel: 2-to-1 MUX

$$t_{delay} = t_{\text{int}} + C_{load} \cdot t_{ext}$$

from input	to output	intrinsic [ns]	extrinsic [ns/pF]
D0↓	Z↓	1.42	2.10
D0↑	Z↑	1.23	3.66
D1↓	Z↓	1.42	2.10
D1↑	Z↑	1.23	3.66
SD↓/ ↑	Z↓	1.42	2.10
SD ↓/ ↑	Z↑	1.09	3.66

© A. Steininger / TU Wien

Derating Factors



	4.50 V	4.75 V	5.00 V	5.25 V	5.50 V
-40°	0.77	0.73	0.68	0.64	0.61
0°	1.00	0.93	0.87	0.82	0.78
25°	1.14	1.07	1.00	0.94	0.90
85°	1.50	1.40	1.33	1.26	1.20
100°	1.60	1.49	1.41	1.34	1.28
125°	1.76	1.65	1.56	1.47	1.41

Die Angaben im Datenblatt sind der worst case innerhalb eines spezifizierten Bereiches.
Grundsätzlich gilt: Hohe Temperatur und niedrige Versorgung verlangsamen den Chip.

© A. Steininger / TU Wien

Formale Verifikation: Prinzip

- bei komplexen Designs wird Coverage von Test bzw. Simulation zum Problem
- Lösung: formale, vollständige Überprüfung
- überprüft werden meist
  - Model Checking: Überprüfen von bekannten Bedingungen/Eigenschaften (Deadlocks etc.)
  - Equivalence checking: Überprüfen der Übereinstimmung zweier Modelle (z.B. des Designs auf verschiedenen Abstraktionsebenen)
- erfordert Darstellung des Designs als Modell

© A. Steininger / TU Wien

Formale Verifikation: Tools

- Model checking
  - vollautomatisch
  - kommerzielle Tools
  - benötigt spezif. Parametrierung
  - liefert pass/fail Entscheidung (evtl. „Gegenbeispiel“)
  - gut für nach-trägliche Prüfung
- formale Beweise
  - manuell
  - mathemat. Methoden
  - Parameter als Variable zulässig
  - liefert Bedingungen in Form v. Ungleichungen
  - gut als Grundlage für Design-Entscheidungen

© A. Steininger / TU Wien

Benötigte Design-Tools



- Design-Entry
- Logic Compiler (technologieunabhängig)
- Simulation (div. Levels)
- PPR (incl. Technology-Mapping)

...sind grundsätzlich unabhängige Funktionen,

⇨ Tools verschiedener Hersteller kombinierbar, aber

☹ Unterschiedliche Zielrichtungen & Stärken

☹ Kompatibilität ?

☹ Back-Annotation ?

© A. Steininger / TU Wien



Zusammenfassung (1)

- Der Design-Flow eines ASIC bzw. FPGA umfasst die folgenden Schritte:
  - Spezifikation
  - Design-Entry
  - Compliation
  - Technology-Mapping
  - (Partitioning)
  - Floorplanning, Placement & Routing (PPR)
  - Fertigung bzw. Download

© A. Steininger / TU Wien



Zusammenfassung (2)

- Der Design-Prozess ist höchst komplex und daher fehleranfällig. An vielen Stellen ist daher eine Verifikation nötig, und im Fehlerfall müssen die einzelne Design-Schritte wiederholt werden. Das Design ist also ein iterativer Prozess.
- Die Verifikation umfasst folgende Schritte:
  - Validation (Prüfen der Spezifikation),
  - Simulation (Prüfen der virtuellen Implementierung) und
  - Test (Prüfen des physikalischen Designs)

© A. Steininger / TU Wien



Zusammenfassung (3)

- Das Y-Diagramm erlaubt eine Veranschaulichung des Design-Prozesses. Es umfasst die 3 Achsen Verhalten, Struktur und Geometrie. Durch konzentrische Kreise werden die Abstraktions-ebenen dargestellt.
- Der Design-Prozess beginnt auf einer hohen Abstraktionsebene (typ. RTL) auf der Verhaltens- (und/oder Struktur-) achse. Mittels Tool-Support gelangt man über Struktur und die Geometrie-achse zu niedrigeren Abstraktionsebenen und schließlich ins Zentrum des Diagramms.

© A. Steininger / TU Wien



Zusammenfassung (4)

- In HDLs lässt sich vieles einfach beschreiben, eine Abbildung auf HW erweist sich jedoch bei der Synthese oft als zu aufwendig oder unmöglich.
- Durch einen Optimierungsprozeß wird eine Kostenfunktion minimiert bzw. eine Nutzenfunktion maximiert, jeweils unter Einhaltung gegebener Randbedingungen.
- Partitioning, Placement und Routing sind solche Optimierungsprozesse. Vielfach werden hier aufgrund der Komplexität heuristische Methoden den geschlossenen Lösungen vorgezogen.

© A. Steininger / TU Wien



Zusammenfassung (5)

- Für Partitioning und insbesondere Placement und Routing ist eine Abschätzung der Signallaufzeiten essenziell. Diese erweist sich jedoch aufgrund der Dominanz des Interconnect-Delay für neuere Technologien als zunehmend schwieriger.
- Simulation sollte auf möglichst vielen Ebenen durchgeführt werden, um Fehler rasch und eindeutig identifizieren zu können.
- Die Signal-Resolution Table gibt Aufschluss darüber, welcher Ausgangspegel aus dem Zusammenwirken mehrerer Eingangspegel an einem bestimmten Logikelement entsteht.

© A. Steininger / TU Wien



Zusammenfassung (6)

- Bei der ereignisgesteuerten Simulation werden die Ereignisse nach einer Liste chronologisch abgearbeitet, neue Folge-Ereignisse werden in der Liste ergänzt. Gleichzeitigkeit wird durch die „Delta-Time" berücksichtigt.
- Die statische Timinganalyse sucht systematisch das Design nach den langsamsten Datenpfaden ab.
- Das Timing ist in Libraries definiert. Variationen in der Temperatur oder der Versorgungsspannung werden durch Derating-Factors berücksichtigt.

© A. Steininger / TU Wien



Zusammenfassung (7)

- Die formale Verifikation erlaubt eine lückenlose Überprüfung des Designs nach bestimmten Kriterien. Voraussetzung ist aber das Vorliegen eines entsprechenden Modells.
- Als Werkzeuge werden einerseits Modelchecker und andererseits formale Beweise verwendet.

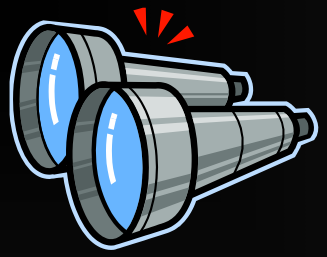
© A. Steininger / TU Wien

Speichertechnologien

Über das richtige Ablegen
von Bits & Bytes



© A. Steininger / TU Wien



Überblick

- Funktionsprinzipien von Speichern
- Klassifikation von Speichern
- ROM, SRAM, DRAM, MRAM:
  Aufbau, Varianten, Anwendung, Timing
- Multiport-RAM und FIFO
- Error Detection & Correction

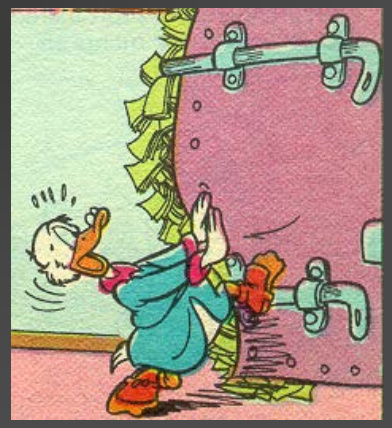
© A. Steininger / TU Wien

Digitaler Speicher: Prinzip

Für jedes Bit gibt es einen Informationsträger mit zwei klar unterscheidbaren

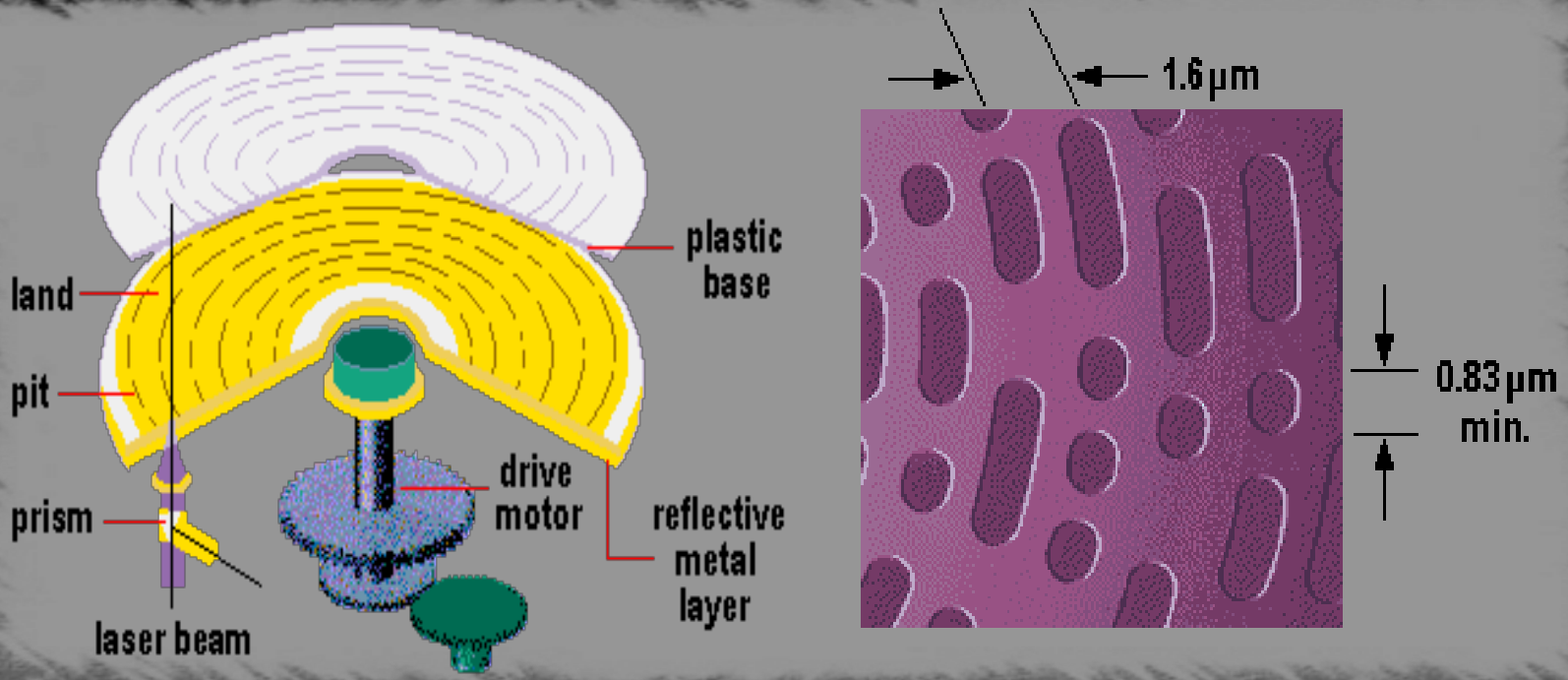
- Spannungspegeln / Ladungszuständen
- Magnetisierungsrichtungen
- Optischen Reflexionseigenschaften
- Topologien
- ...

Die strukturierte Anordnung solcher Informationsträger bildet einen Speicher.



© A. Steininger / TU Wien

Beispiel: Optischer Speicher



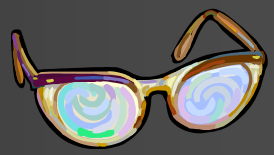
„Pit" (Vertiefung) reflektiert schlecht, „Land" gut

© A. Steininger / TU Wien

Lesen und Schreiben



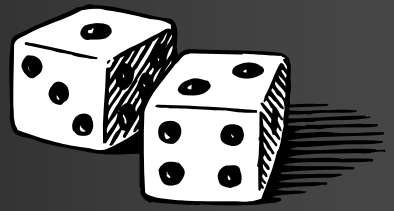
- Das „Programmieren"/„Schreiben"
  - … ist im Betrieb möglich
    => Read/write memory
  - … ist im Betreib nicht mehr möglich
    => Read-only Memory (ROM)
    Daten sind non-volatile & schreibgeschützt



- Das Auslesen der Speicherinhalte ist im Betrieb immer möglich
  - macht ein Write-only Memory (WOM) Sinn?

© A. Steininger / TU Wien

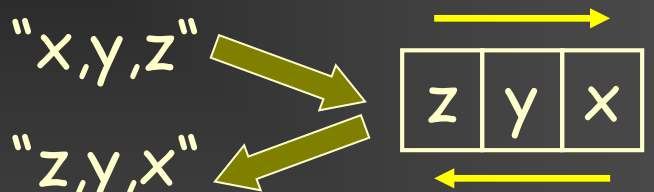
Random Access



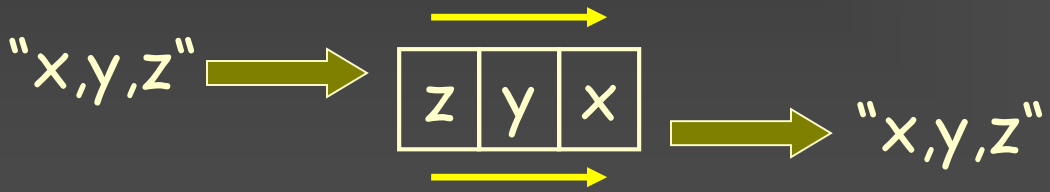
...bedeutet wahlfreien Zugriff auf jeden Inhalt (Adresse) bei gleicher Zugriffszeit

Gegenbeispiel: sequentieller Zugriff

- FILO (Stack):



- FIFO:



© A. Steininger / TU Wien

Non-Volatile Memory (NVM)

...bezeichnet „nicht-flüchtigen" Speicher: Speicherinhalte bleiben nach Abschalten der Spannungsversorgung erhalten

- Beispiele: ROM, Harddisk, CDROM,

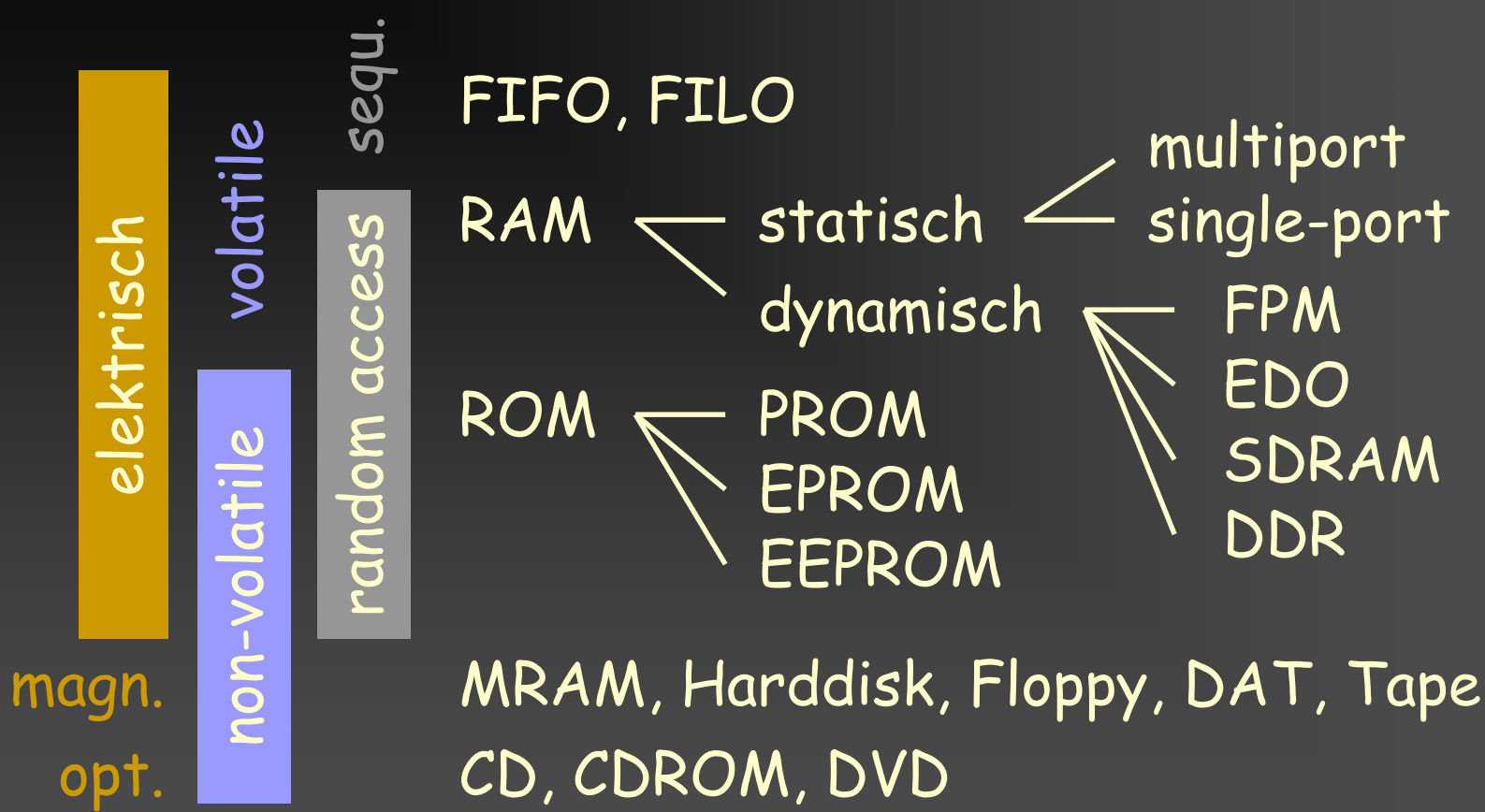
Gegensatz:
volatile Memory („flüchtiger" Speicher)

- Beispiele: SRAM, DRAM

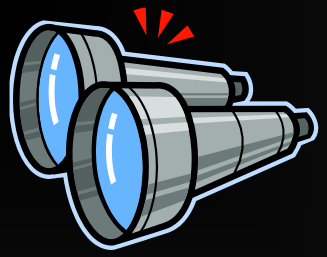


© A. Steininger / TU Wien

Speichertypen: Überblick



© A. Steininger / TU Wien



Überblick

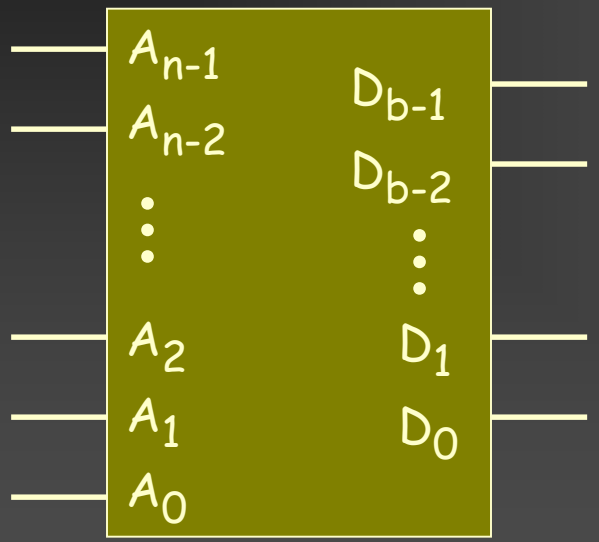
- Funktionsprinzipien von Speichern
- Klassifikation von Speichern
- **ROM**, SRAM, DRAM, MRAM:
  Aufbau, Varianten, Anwendung, Timing
- Multiport-RAM und FIFO
- Error Detection & Correction

© A. Steininger / TU Wien

Read-only Memory: Funktion

Ein Eingangsmuster („Adresse“) wird durch eine kombinatorische Funktion auf ein Ausgangsmuster („Daten“) abgebildet

2^n x b ROM :



Adresse

Daten-wort

© A. Steininger / TU Wien

Read-only Memory: Anwendung

- Befehlscode für Prozessor
- Beliebige kombinatorische Funktion
  - Wahrheitstabelle
- Tabelle für
  - Berechnung (Multiplikation)
  - Linearisierung
  - Transformation

© A. Steininger / TU Wien

EPROM als Befehlsspeicher

Address		Data	
...	...	...	...
0x08	00001000	00010100	add r1,r2,r5
0x09	00001001	01001001	cmp r2,r0,r4
0x0A	00001010	11000101	beq r4,r7,r3
0x0B	00001011	11100010	or r6,r7,r5
0x0C	00001100	00100011	addi r1,0x2,r2
0x0D	00001101	11110010	cmp r1,r0,r3
...	...	...	...

EPROM

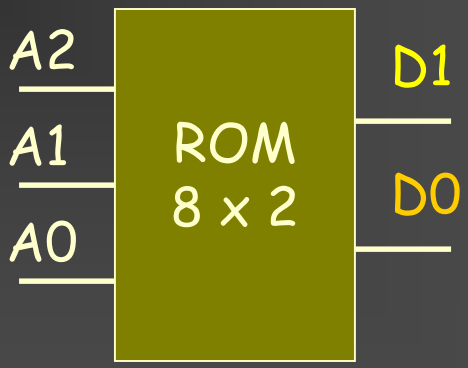
© A. Steininger / TU Wien

Wahrheitstabelle mit ROM

Adresse			Datenbits	
A2	A1	A0	D1	D0
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

$$D0 = \neg A2 \wedge \neg A1 \wedge A0$$

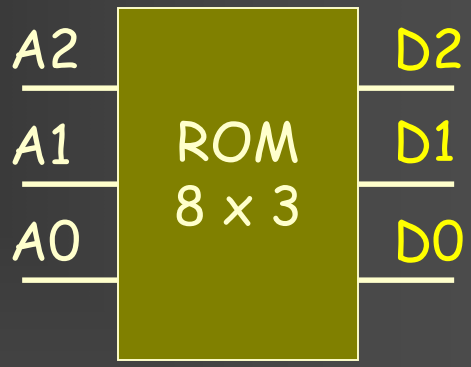
$$D1 = \neg(A2 \vee A1)$$



© A. Steininger / TU Wien

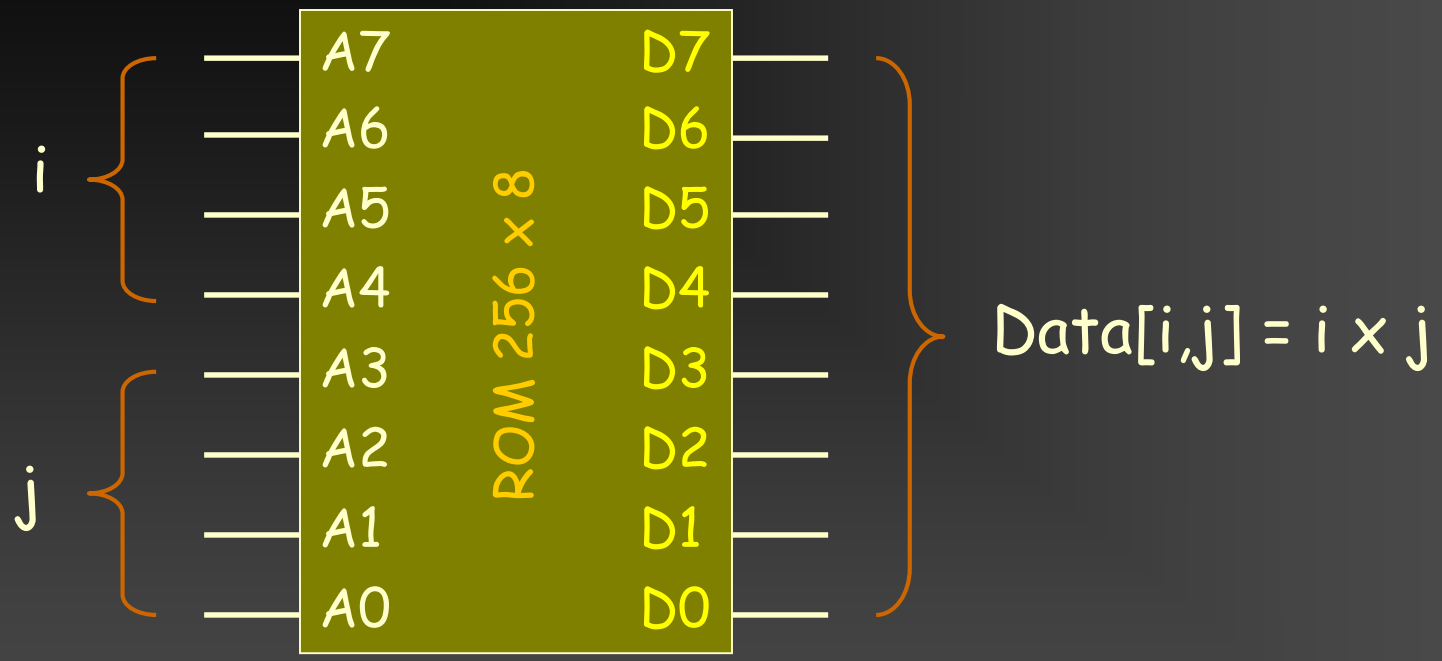
Gray-Decoder mittels ROM

Binärcode			Gray-Code		
A2	A1	A0	D2	D1	D0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0



© A. Steininger / TU Wien

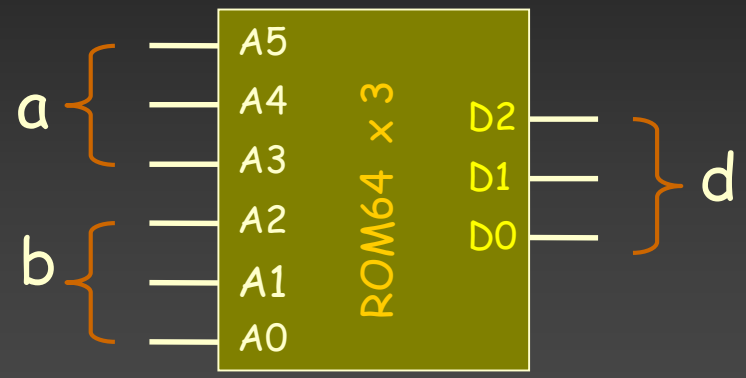
Multiplikation mittels ROM



© A. Steininger / TU Wien

Beispiel zum ROM

- Entwerfen Sie eine Schaltung, die das Maximum zweier positiver 3-bit Binärzahlen ausgibt



a \ b	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	1	2	3	4	5	6	7
2	2	2	2	3	4	5	6	7
3	3	3	3	3	4	5	6	7
4	4	4	4	4	4	5	6	7
5	5	5	5	5	5	5	6	7
6	6	6	6	6	6	6	6	7
7	7	7	7	7	7	7	7	7

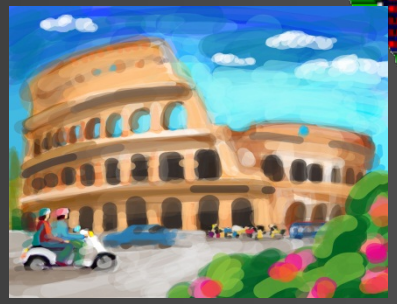
© A. Steininger / TU Wien

Read-only Memory: Typen

- **ROM** („mask-ROM")
  Prog. bei der Fertigung, nicht löschbar
- **PROM** (programmable ROM)
  Prog. beim Anwender („one-time programmable" OTP), nicht löschbar
- **EPROM** („UV-EPROM", erasable PROM)
  wie PROM, durch UV-Licht löschbar
- **EEPROM** (electrically erasable PROM)
  wie PROM, elektrisch („in circuit") löschbar

© A. Steininger / TU Wien

Prinzip eines ROM (1)

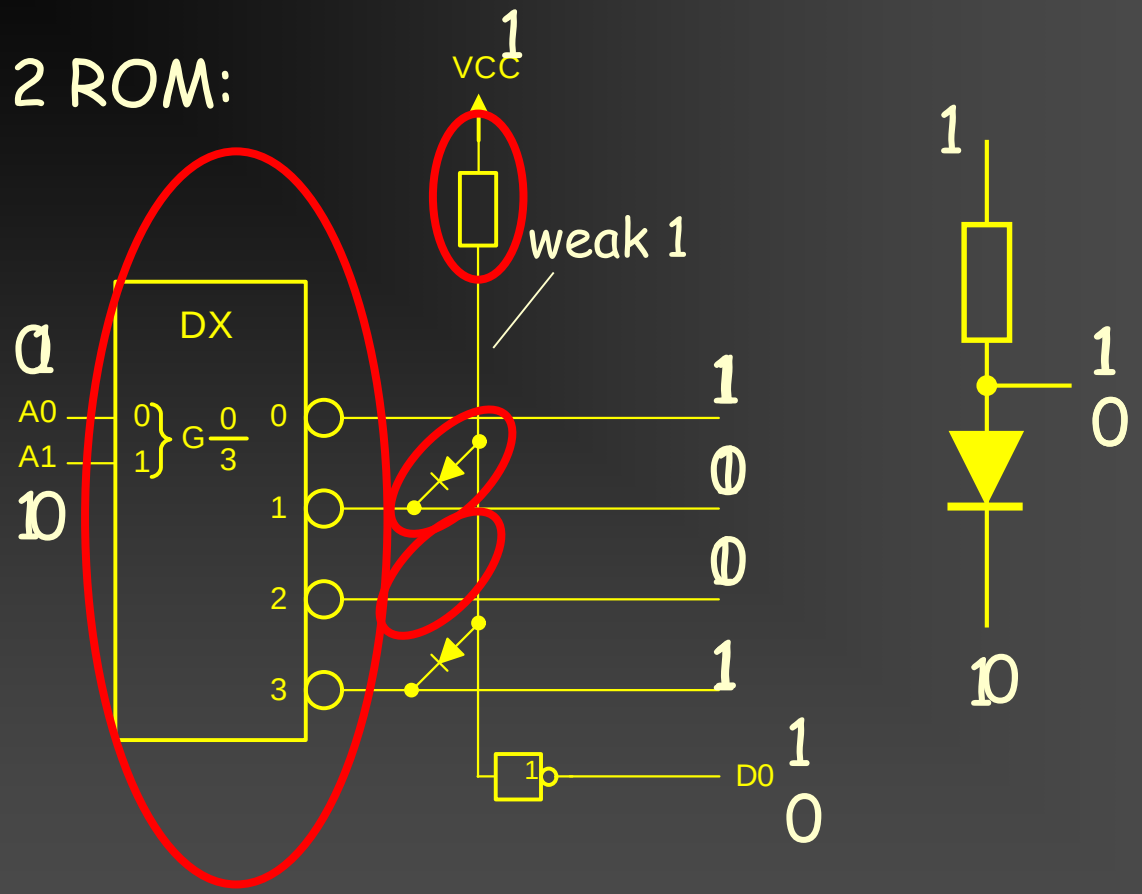


- Decoder wandelt Adresse von binär auf 1-aus-n um. Die Ausgänge des Decoders sind invertiert.
- Eine „Zeile" Zy ist also auf 0, alle anderen auf 1.
- Die Bits des Datenwortes sind in Spalten angeordnet. Jede Spalte wird über einen Pull-up-Widerstand (schwach) auf 1 gezogen
- Liegt im Kreuzungspunkt der Spalte Sx mit der selektierten Zeile Zy eine Diode, so wird das Datenbit (Bit x, Adresse y) auf 0 gezogen, anderenfalls bleibt es auf 1 (Pull-up).
- Die Programmierung erfolgt durch Setzen / Entfernen der Dioden in die Kreuzungspunkte.

© A. Steininger / TU Wien

Prinzip eines ROM (2)

Beispiel 4 x 2 ROM:



© A. Steininger / TU Wien

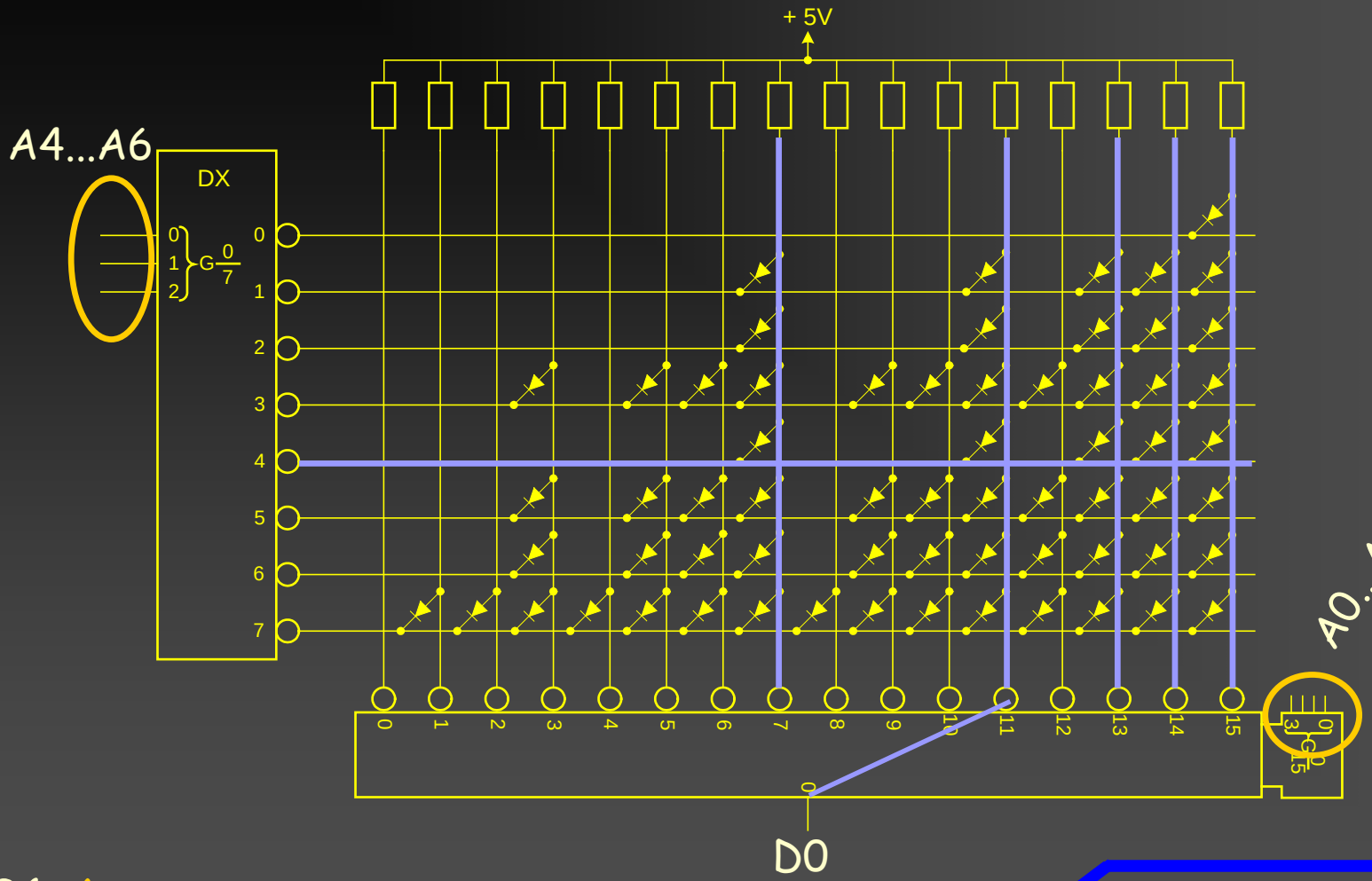
Probleme mit dem Decoder

- Ein Speicher mit 2^n Bit benötigt einen 1-aus-2^n Decoder, d.h. 2^n AND Gates mit je n Eingängen.
- Für ein 1M x 1 ROM (20-zu-1M-Decoder) sind dies 1048576 AND-Gatter mit je 20 Eingängen.

- Bei der zweidimensionalen Decodierung benötigt man im Idealfall je einen Decoder und einen Multiplexer für $n/2$ bit.
- Für das 1M x 1 ROM sind dies ein 10-zu-1k-Decoder und ein 1k-zu-1-Mux , das sind zusammen 1024 AND-Gatter mit je 10 oder 11 Eingängen
  – eine Ersparnis um den Faktor 2000 !

© A. Steininger / TU Wien

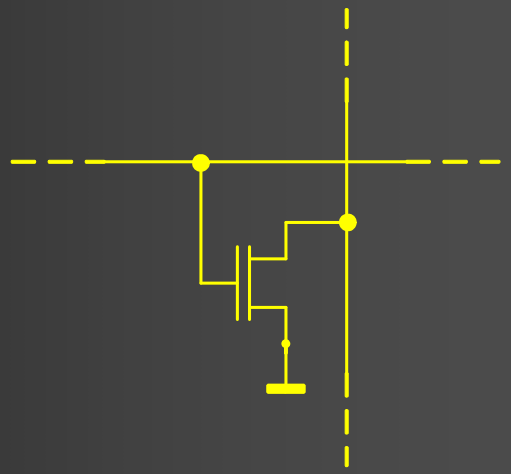
Zweidimensionale Decodierung



© A. Steininger / TU Wien

Mask-ROM: Implementierung

- Die Dioden in den Kreuzungspunkten (in der Praxis verwendet man Transistoren) werden bei der Fertigung entsprechend den vom Anwender spezifizierten Dateninhalten angeschlossen (Maske für die Metallisierung => Name)



- Jede Änderung erfordert neue Chipfertigung (!), => nur für Massenprodukte wirtschaftlich
- Datenspeicherung ist äußerst robust und störfest

© A. Steininger / TU Wien

OTP: Implementierung

- Bei der Fertigung werden alle Dioden (Transistoren) angeschlossen.
- Programmierung erfolgt beim Anwender: Durch eine Programmierspannung (z.B. 12V) werden bei unerwünschten Verbindungen „Sicherungen" gezielt durchgebrannt, nur die erwünschten verbleiben
- nicht änderbar (=> „OTP"), billig, robust, rasch programmierbar (≈10μs/Byte)

© A. Steininger / TU Wien

UV-EPROM: Implementierung

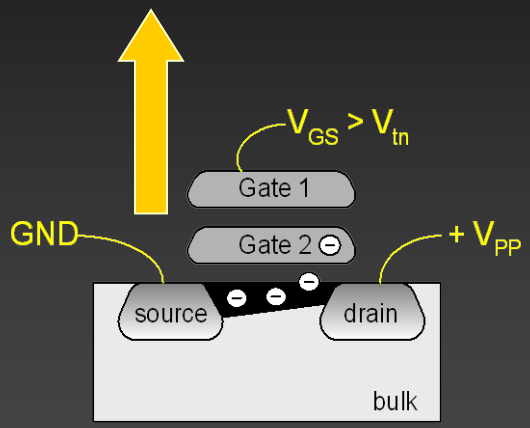
- Bei der Fertigung werden in allen Kreuzungspunkten Transistoren mit „Floating Gate“ eingefügt.
- Programmierung erfolgt beim Anwender: Kreuzungspunkte werden gezielt verbunden
- löschbar mit UV (nur pauschal, ≈15 Min.), störanfällig (Daten halten 10 Jahre, UV-empfindlich, nicht radiation hard,...), rasch programmierbar (≈10μs/Byte).

© A. Steininger / TU Wien

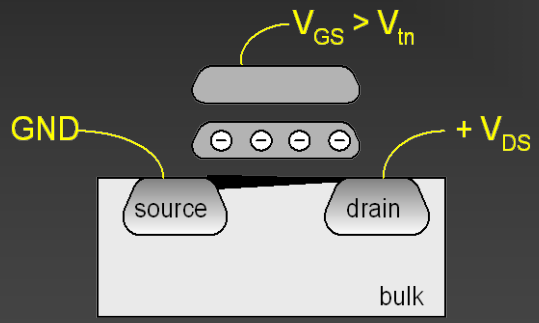
Floating Gate



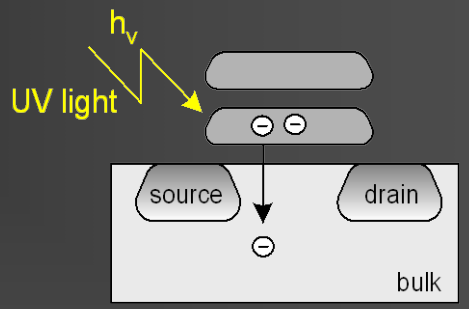
elektrisches Feld zieht Elektronen ins Floating Gate
=> Steuerspannung zum Schließen des Schalters wird so hoch, dass kein Schließen mehr möglich.



Programmierung



Auslesen

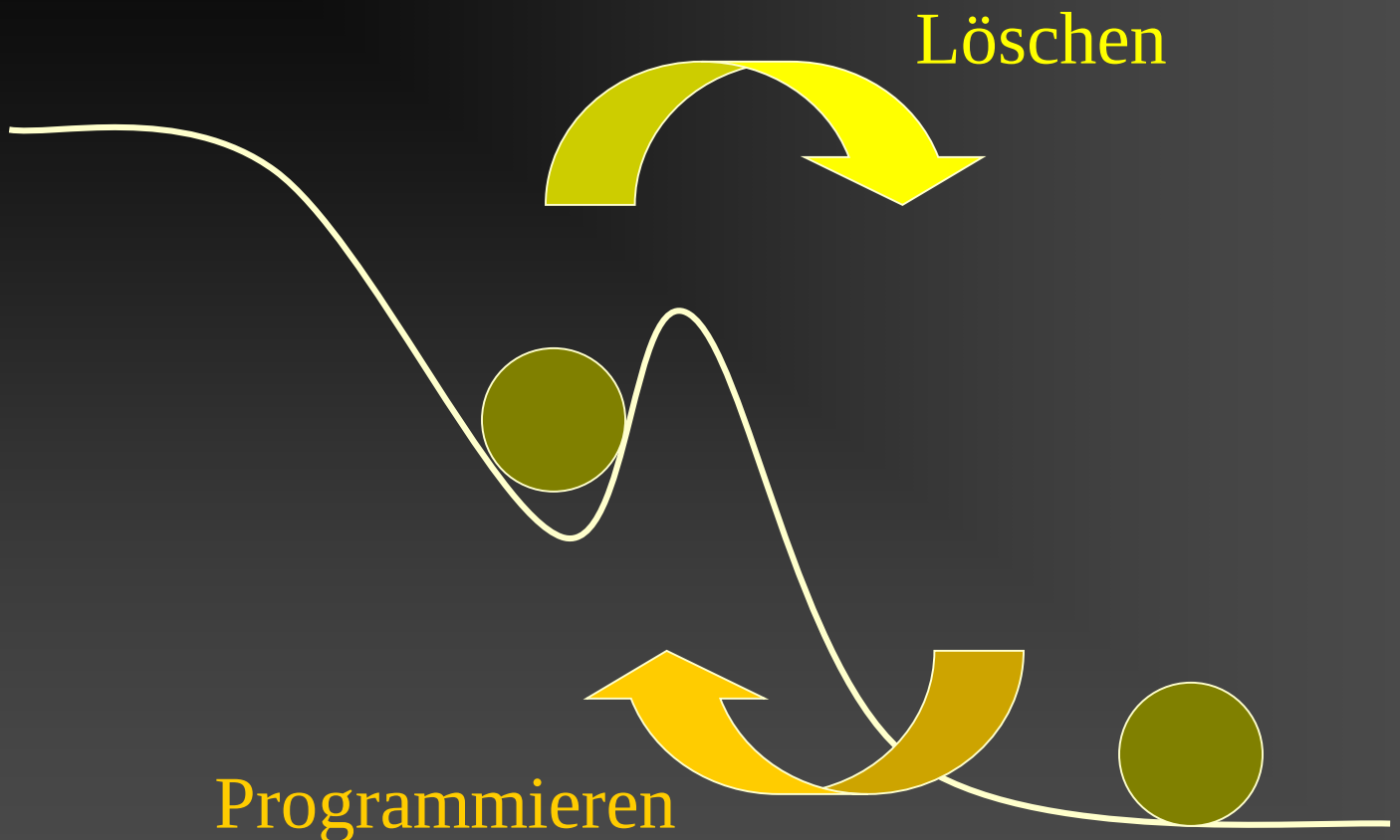


Löschen

© A. Steininger / TU Wien

Energie im EPROM



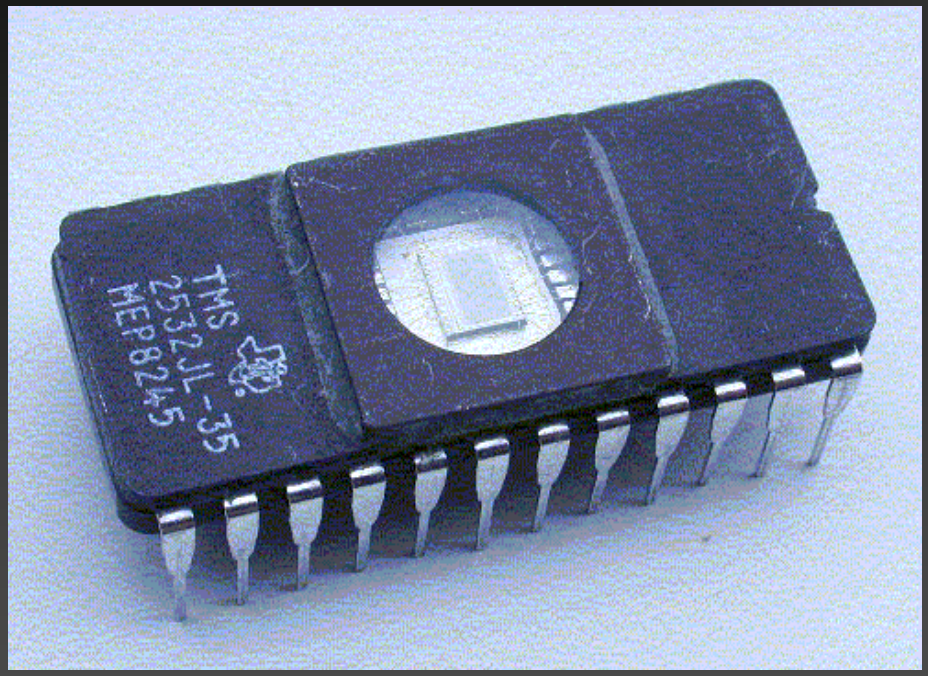


© A. Steininger / TU Wien

UV-EPROM: Package



Package mit Fenster



... lässt beim Löschen das UV-Licht an den Chip.

... sollte im Betrieb zugeklebt werden

... macht das Bauteil teurer

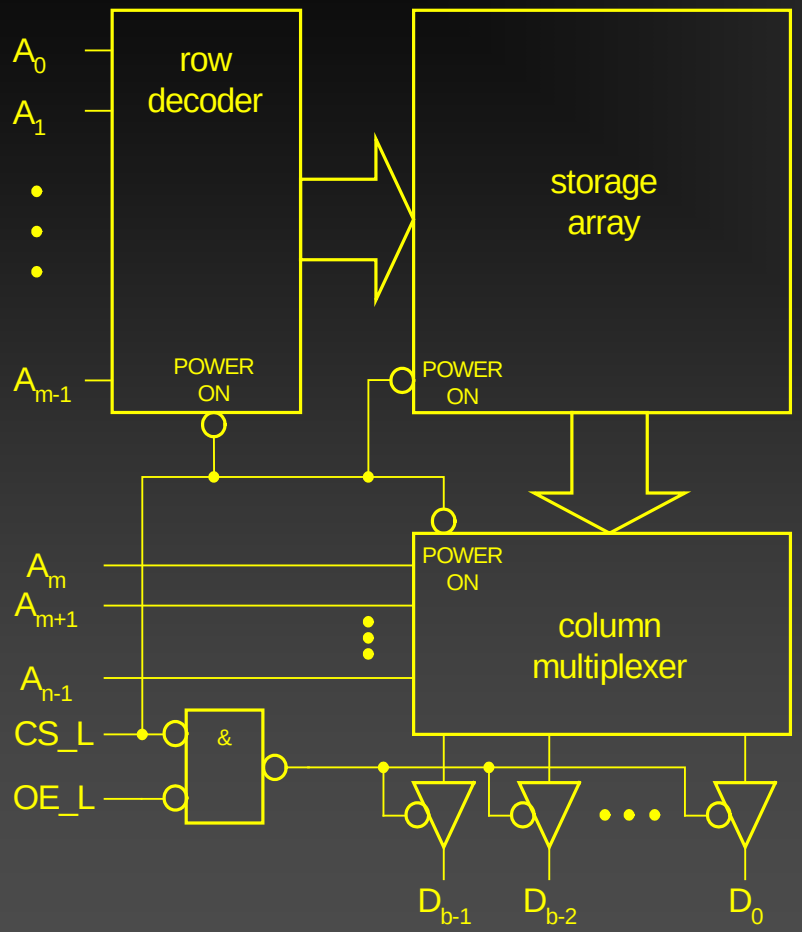
© A. Steininger / TU Wien

EEPROM: Implementierung

- Fertigung & Programmierung wie beim UV-EPROM
- Löschbar durch elektr. Feld (Spannung), Zellen einzeln löschbar
  - „Flash Memory“: nur pauschal löschbar
- Unterschiede zum RAM:
  non-volatile, Schreiben dauert viel länger
- rasch programmierbar (≈10μs/Byte), störanfällig, nur ≈10000 Programmier-zyklen je Zelle.

© A. Steininger / TU Wien

Steuersignale eines ROM



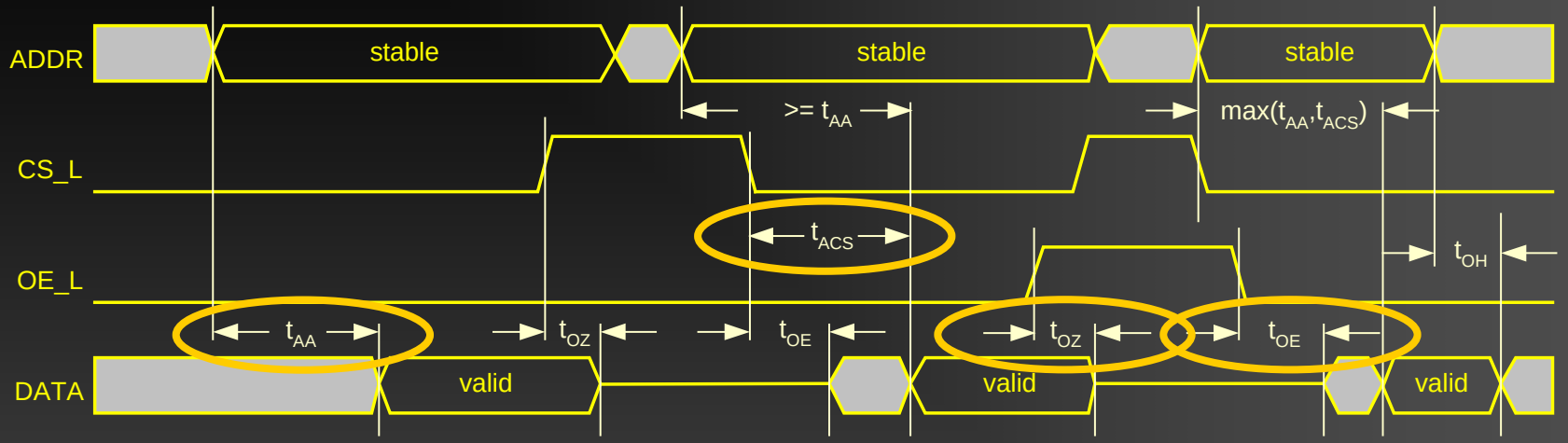
Steuersignale:

- Adressen $A_0 \ldots A_{n-1}$
  Wahl der Speicherzelle
- Chip Select CS
  Aktivierung des Chip (low-aktiv)
- Output Enable OE
  Aktivierung der Ausgangstreiber (low-aktiv)

© A. Steininger / TU Wien

ROM - Timing





Daten werden gültig (1) t_{AA} nach Anlegen der gültigen Adresse
(2) t_{ACS} nach Aktivierung des CS
(3) t_{OE} nach Aktivierung des OE

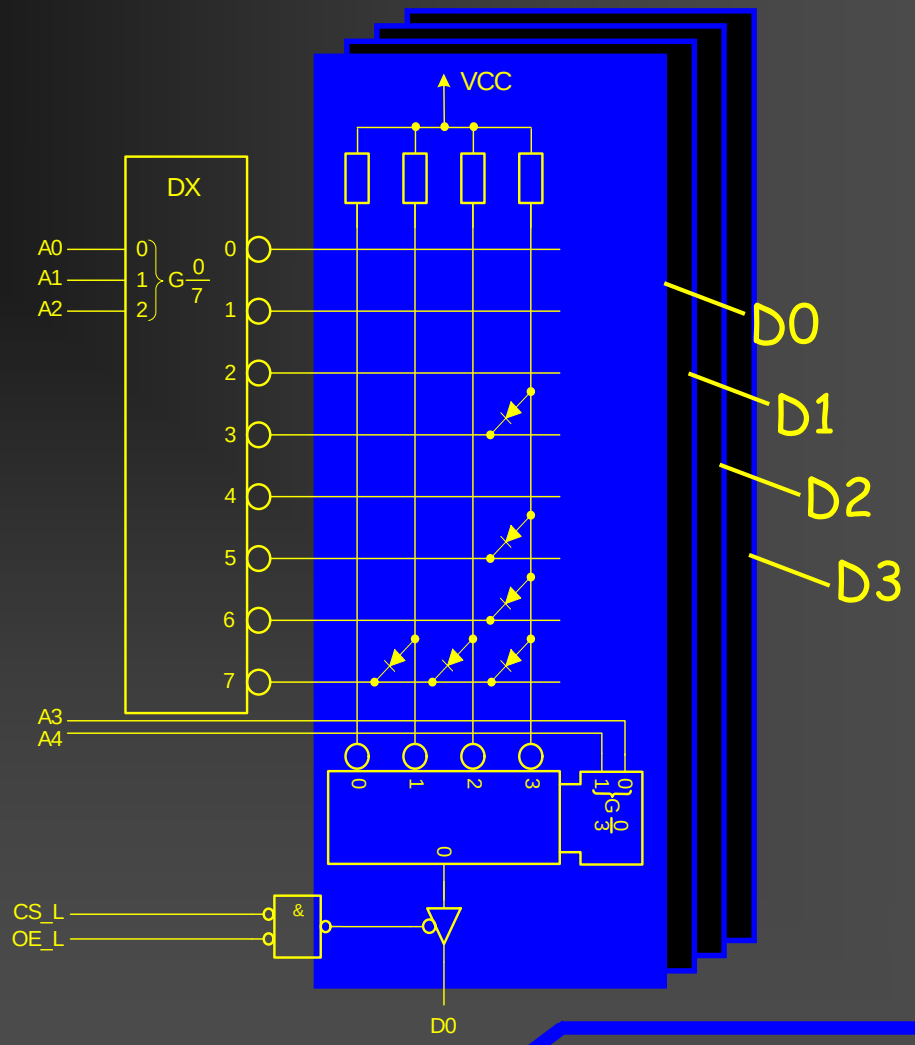
Ausgang ist tri-state t_{OZ} nach Deaktivierung von CS oder OE

© A. Steininger / TU Wien

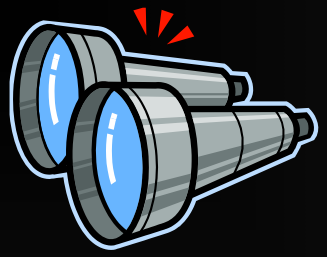
Beispiel zur Decodierung

Skizzieren Sie die interne Struktur eines 32 x 4 bit ROM.

- 2-dimensionale Decodierung (8 x 4)
- Je Datenbit eine solche Einheit (Decoder wird z.T. gemeinsam verwendet)



© A. Steininger / TU Wien

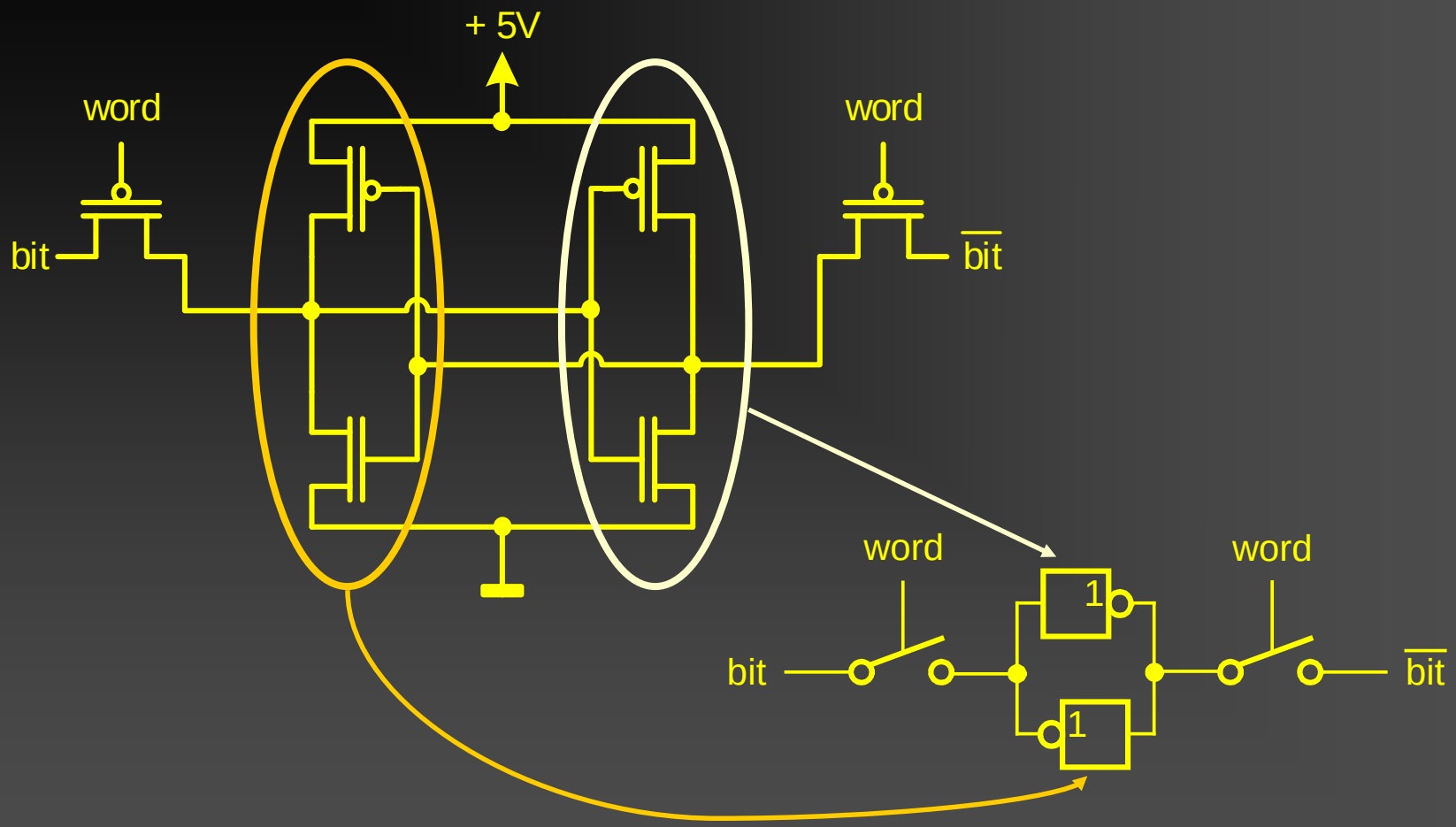


Überblick

- Funktionsprinzipien von Speichern
- Klassifikation von Speichern
- ROM, **SRAM**, DRAM, MRAM:
  Aufbau, Varianten, Anwendung, Timing
- Multiport-RAM und FIFO
- Error Detection & Correction

© A. Steininger / TU Wien

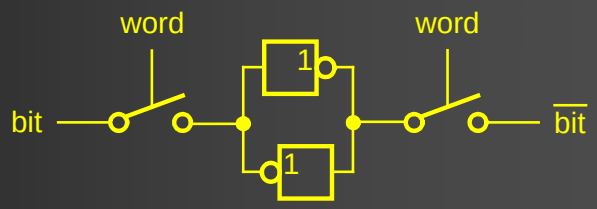
Aufbau eines Static RAM



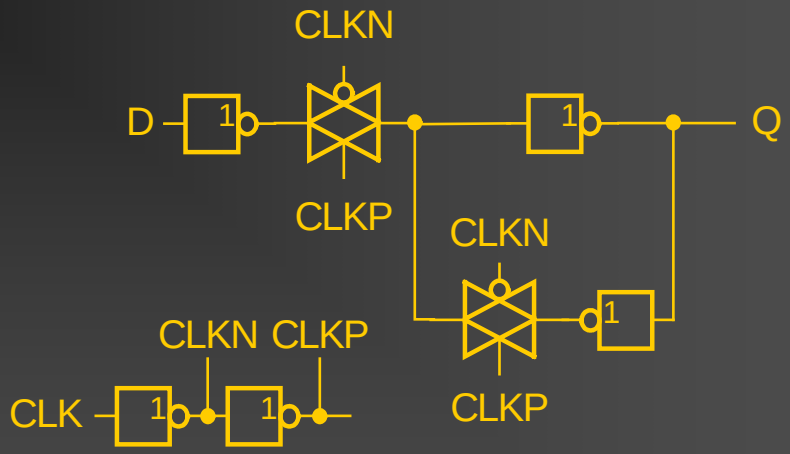
© A. Steininger / TU Wien

SRAM versus Latch

- SRAM-Cell entspricht Storage Loop des Latch



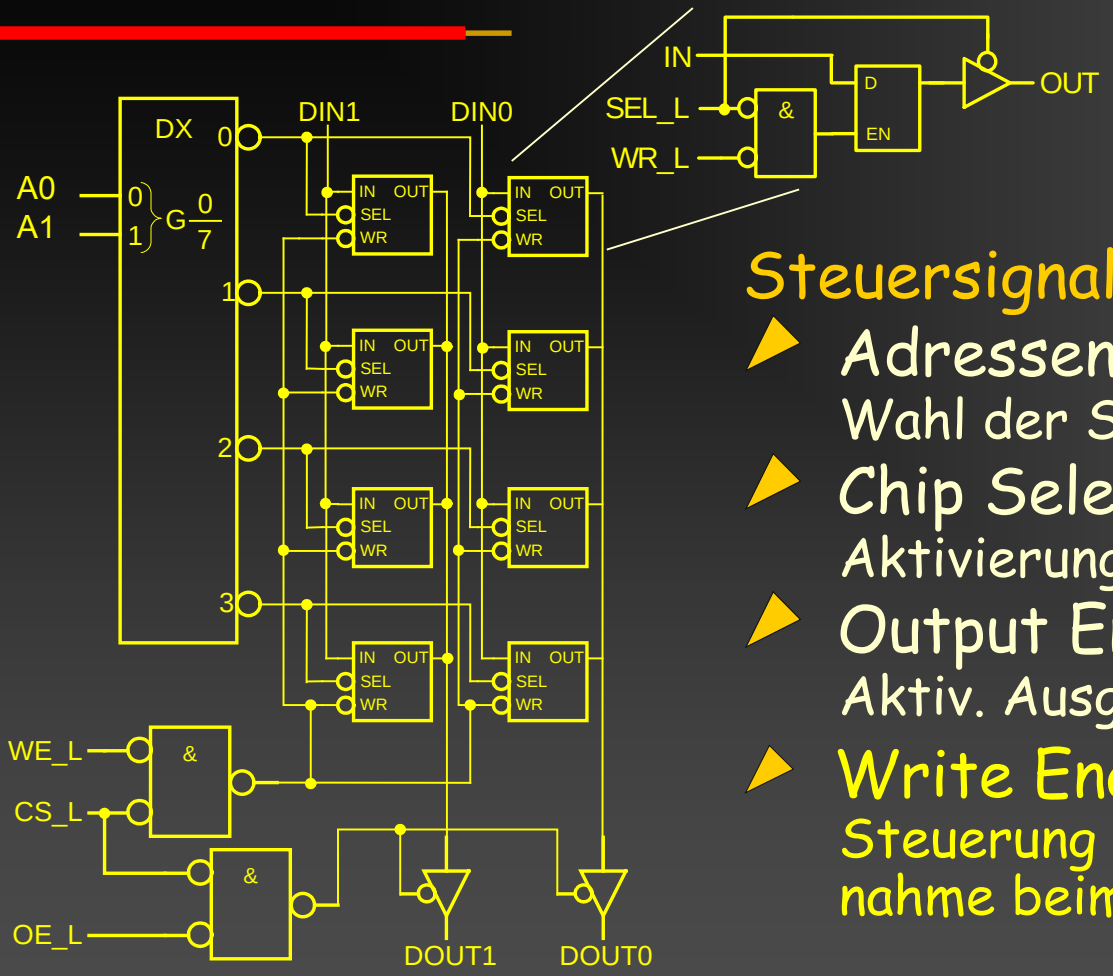
- Einbettung in die reguläre RAM-Struktur
- gemeinsame Verwendung von Steuerlogik + Ein- u. Ausgangsbuffern für alle Zellen



➡ Speicherzellen viel einfacher gehalten

© A. Steininger / TU Wien

Steuersignale eines SRAM



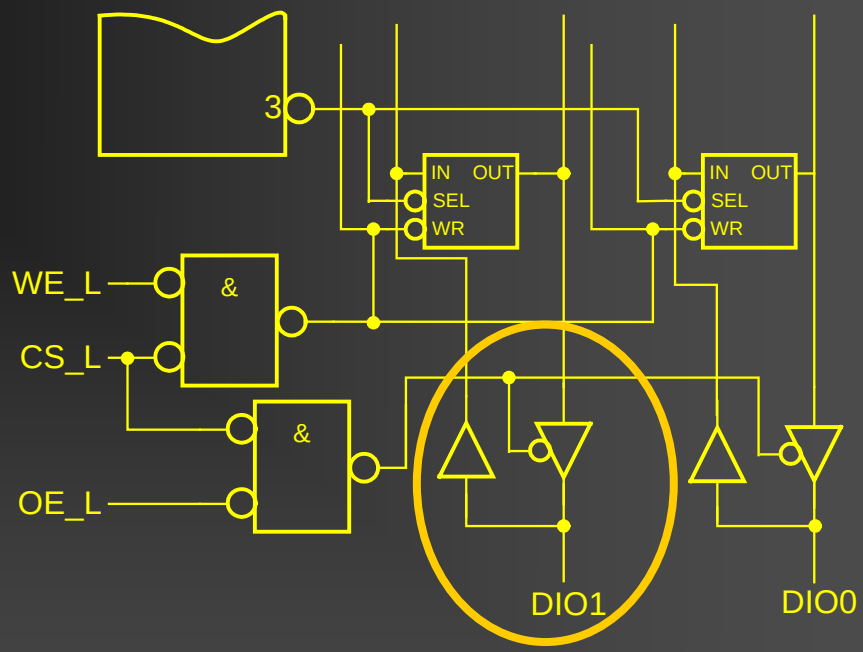
Steuersignale:

- Adressen $A_0 \ldots A_{n-1}$
  Wahl der Speicherzelle
- Chip Select CS
  Aktivierung des Chip
- Output Enable OE
  Aktiv. Ausgangstreiber
- Write Enable WE
  Steuerung der Datenüber-nahme beim WR (low-aktiv)

© A. Steininger / TU Wien

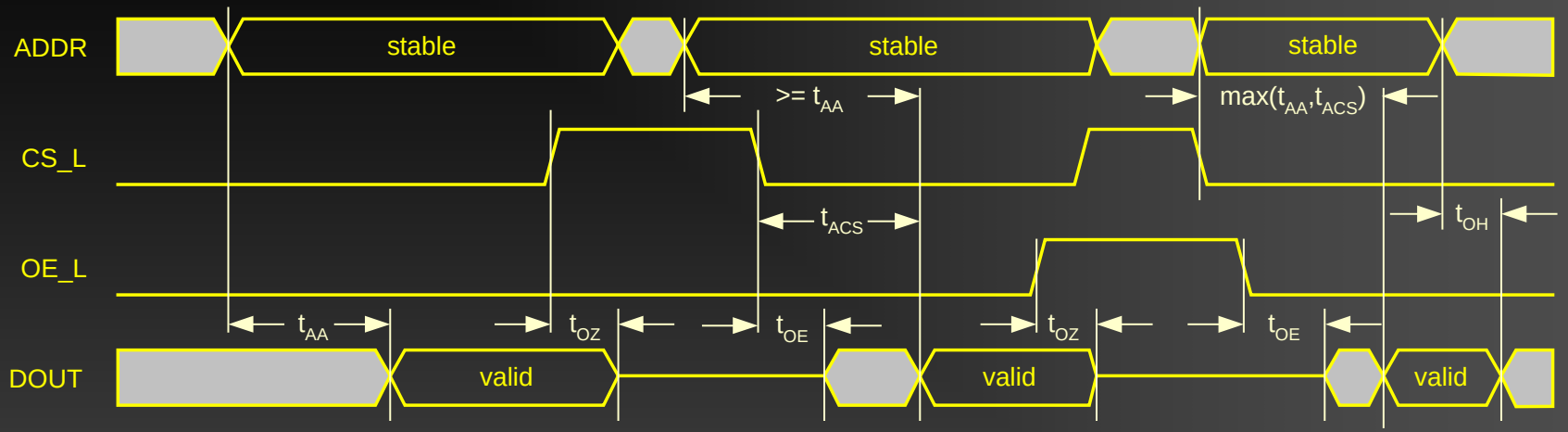
Bidirektionaler Datenbus

- Eingangs- (WR) und Ausgangsdaten (RD) werden über die selben Pins geführt.
  - Einsparung an Pins
  - RD und WR müssen so koordiniert werden, dass je Leitung immer nur ein Treiber aktiv ist



© A. Steininger / TU Wien

SRAM - Read-Timing

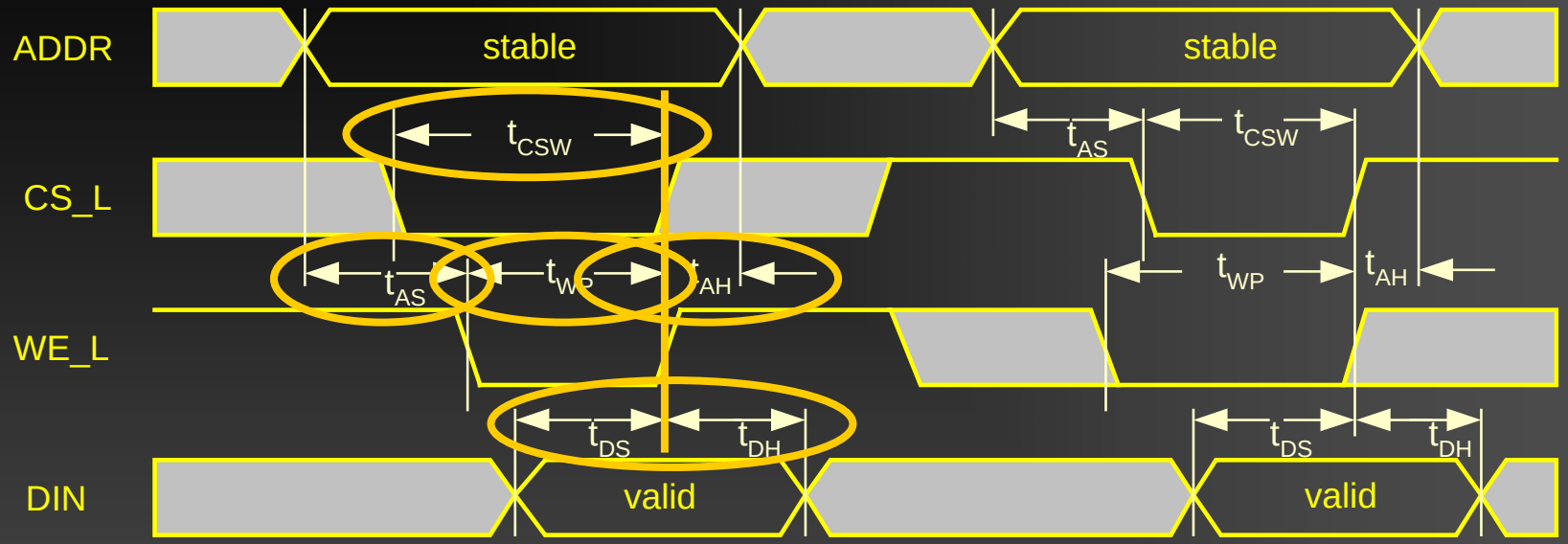


Daten werden gültig (1) t_{AA} nach Anlegen der gültigen Adresse
(2) t_{ACS} nach Aktivierung des CS
(3) t_{OE} nach Aktivierung des OE

Ausgang ist tri-state t_{OZ} nach Deaktivierung von CS oder OE

© A. Steininger / TU Wien

SRAM - Write-Timing



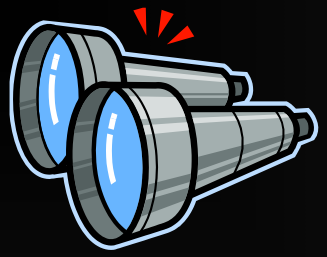
- Daten werden mit steigender Flanke von WE oder CS übernommen
- CS muss mindestens t_{CSW} lang aktiv sein, WE mindestens t_{WP}
- t_{AS} davor bis t_{AH} danach muss die Adresse stabil sein
- t_{DS} davor bis t_{DH} danach müssen die Daten stabil sein (Setup/Hold)

© A. Steininger / TU Wien

Synchrones SRAM

- alle Aktivitäten laufen synchron zu einem Takt
- Eingangssignale werden synchron in Register übernommen
- Read-Daten stehen entweder asynchron (SSRAM mit „flow-through outputs") oder synchron (SSRAM mit „pipelined outputs") zur Verfügung.
- Zugriffe werden intern über eine Pipeline abgewickelt.
- Im „Burst Mode" kann Adresse intern automatisch inkrementiert werden => schneller

© A. Steininger / TU Wien

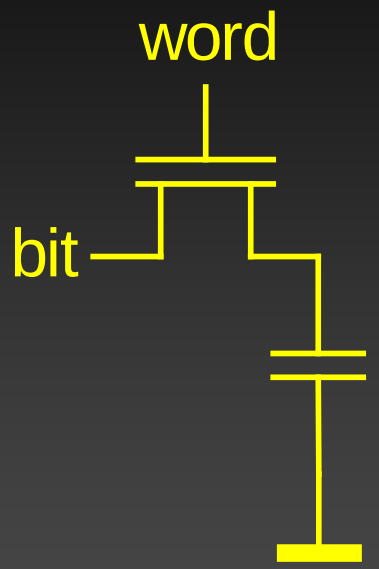


Überblick

- Funktionsprinzipien von Speichern
- Klassifikation von Speichern
- ROM, SRAM, **DRAM**, **MRAM**:
  Aufbau, Varianten, Anwendung, Timing
- Multiport-RAM und FIFO
- Error Detection & Correction

© A. Steininger / TU Wien

Aufbau eines Dynamic RAM



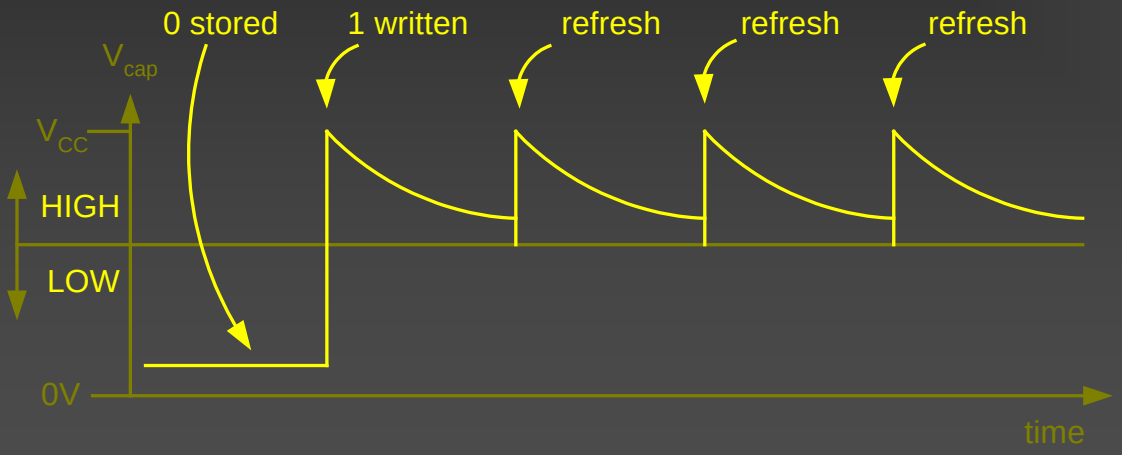
- Bit wird als Spannung in Kondensator abgelegt (HI/LO)
- Solange FET-Schalter offen, bleibt Ladung im C erhalten
- RD: Auslesen der Spannung (FET geschlossen) & Auswertung in „Sense-Amplifier". RD konsumiert Ladung im C => write-back nötig
- WR: Laden/Entladen des C (FET geschlossen)

© A. Steininger / TU Wien

Refresh

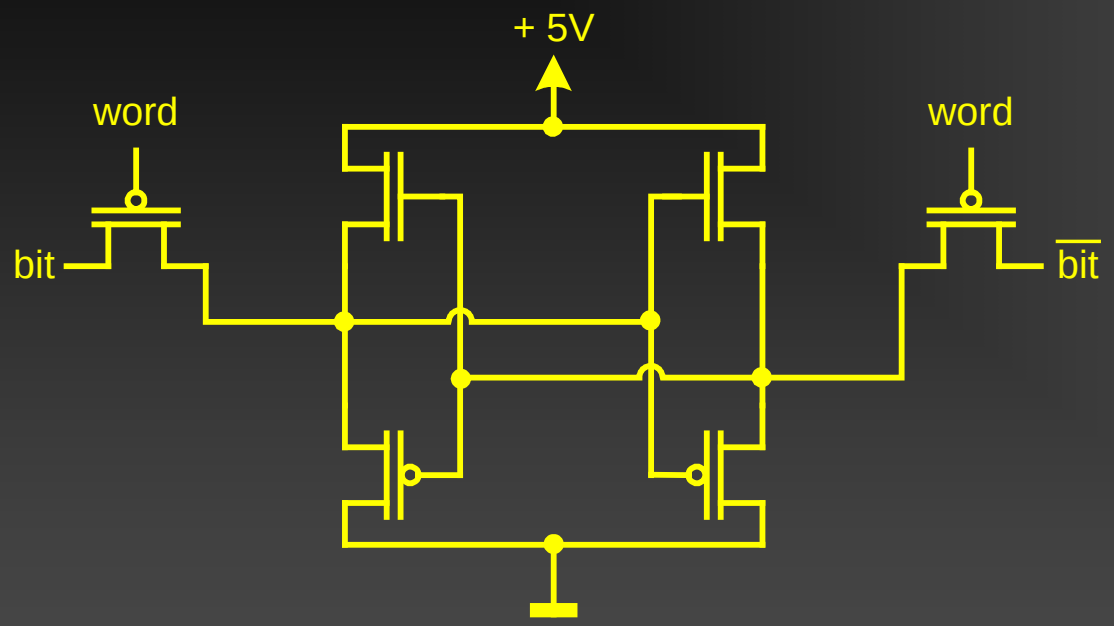


- Aufgrund unvermeidlicher Leckströme entlädt sich der Speicherkondensator (ca. 30fF) der DRAM-Zelle.
- Bevor das Bit „kippt" muss es daher abgefragt und die Ladung im Kondensator entsprechend aufgefrischt werden (typ. alle 1...100ms).
- Dies erfolgt für eine ganze Zeile auf einmal.

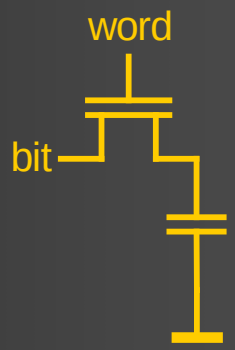


© A. Steininger / TU Wien

Größenvergleich SRAM/DRAM



6 Transistoren



1 Transistor
+ 1 Kondensator

© A. Steininger / TU Wien

SRAM versus DRAM

SRAM

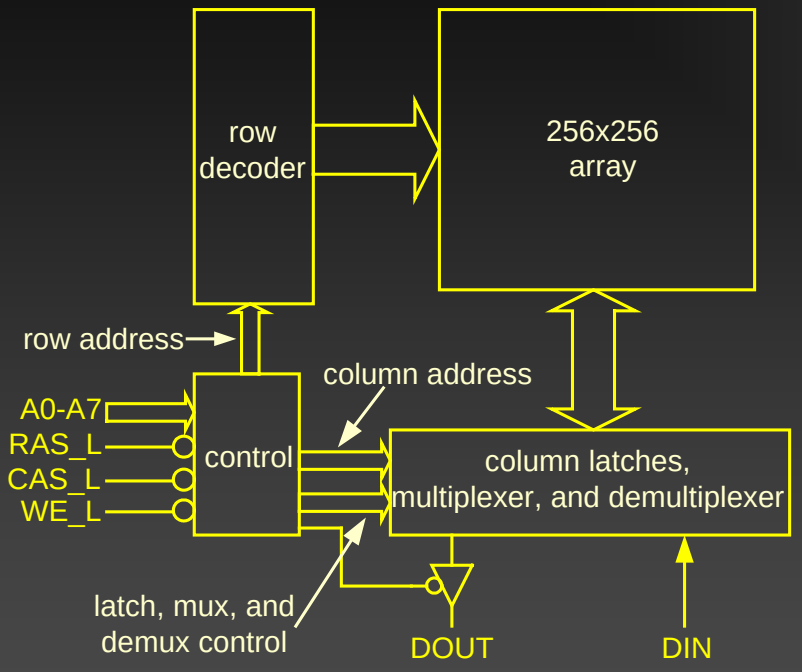
- teurer
- weniger Bits/Chip
- höherer Leistungs-verbrauch
- schneller
- kein Refresh nötig
- robuster
- Anwendung typ. als Cache bzw. für kleine „embedded" Speicher

DRAM

- billiger
- hohe Speicherdichte (Technologietreiber)
- leistungsärmer
- komplizierter Zugriff
- Refresh erforderlich
- störanfälliger
- Anwendung typ. als Hauptspeicher

© A. Steininger / TU Wien

Steuersignale eines DRAM

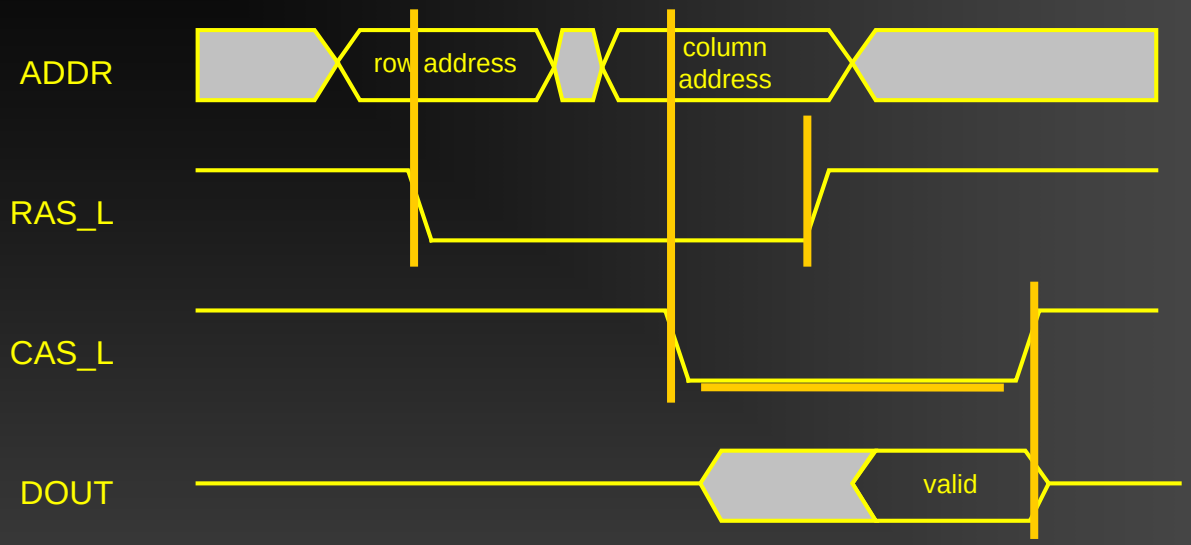


Steuersignale:

- Adressen $A_0 \ldots A_{n-1}$
  nur halbe Anzahl !
- Write Enable WE
  Auswahl: RD oder WR
- Row Addr. Strobe RAS
  Übernahme Zeilenadr.
- Col. Addr. Strobe CAS
  Übernahme Spaltenadr.

© A. Steininger / TU Wien

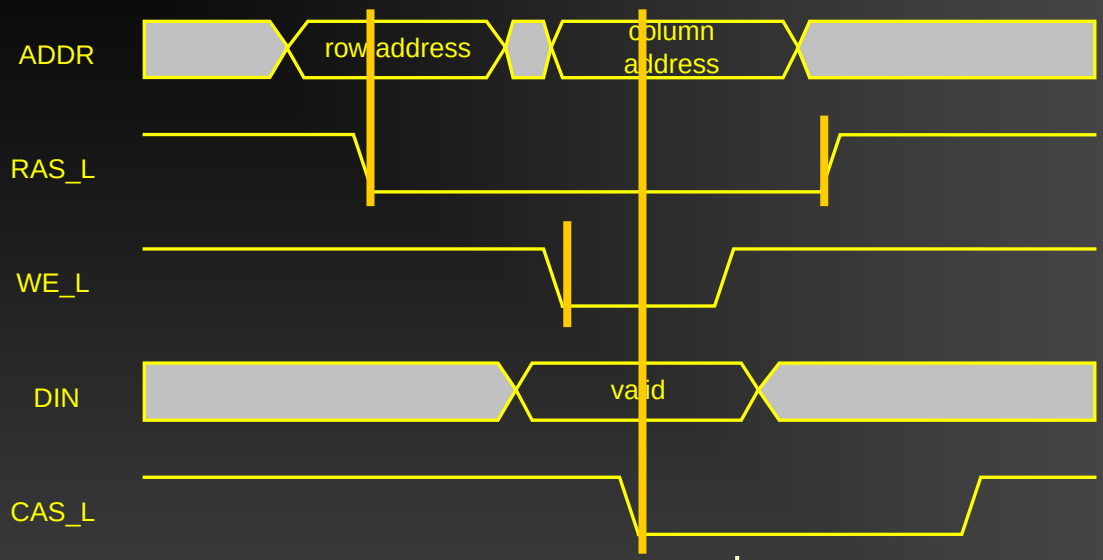
DRAM - Read-Timing



- Anlegen der Row-Address, in ein Latch übernehmen mit RAS ↓
- Anlegen der Column-Adress, in ein Latch übern. mit CAS ↓
- CAS = 0 aktiviert Ausgangstreiber => Daten gültig
- Mit RAS ↑ Rückschreiben der Daten (Refresh) der ganzen Row
- Mit CAS = 1 geht Ausgangstreiber wieder in tri-state

© A. Steininger / TU Wien

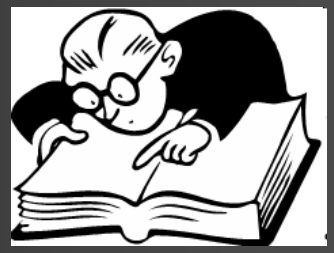
DRAM - Write-Timing



- Latchen der Row-Address mit RAS ↓
- Anlegen der Daten, einleiten des WR mit WE ↓
- Latchen der Column-Adress mit CAS ↓
- Mit RAS ↑ Rückschreiben der Daten (Refresh) der ganzen Row, bei selektierter Column den neuen Datenwert
- Unabh. von CAS bleibt Ausgangstreiber immer in tri-state

© A. Steininger / TU Wien

DRAM – Page Mode



- Erster Zugriff läuft ganz normal ab.
- Danach bleibt jedoch RAS ständig aktiviert, die Row-Address bleibt im Latch gespeichert.
- Bei unmittelbar anschließenden Zugriffen auf die gleiche Row („Page"), sind nur noch CAS-Pulse nötig, um die neuen Column-Addresses zu übernehmen.
- Das ermöglicht einen (etwa 2-fach) schnelleren Zugriff auf Daten innerhalb der gleichen Row

© A. Steininger / TU Wien

Extended Data Out (EDO)

- Die Ausgangstreiber werden nicht von CAS sondern von eigenem Signal Output Enable (OE) gesteuert
- Das erlaubt bessere Überlappung der Zugriffe: Latchen der nächsten Column-Address kann bereits erfolgen, noch während die Daten der vorigen Column gelesen werden

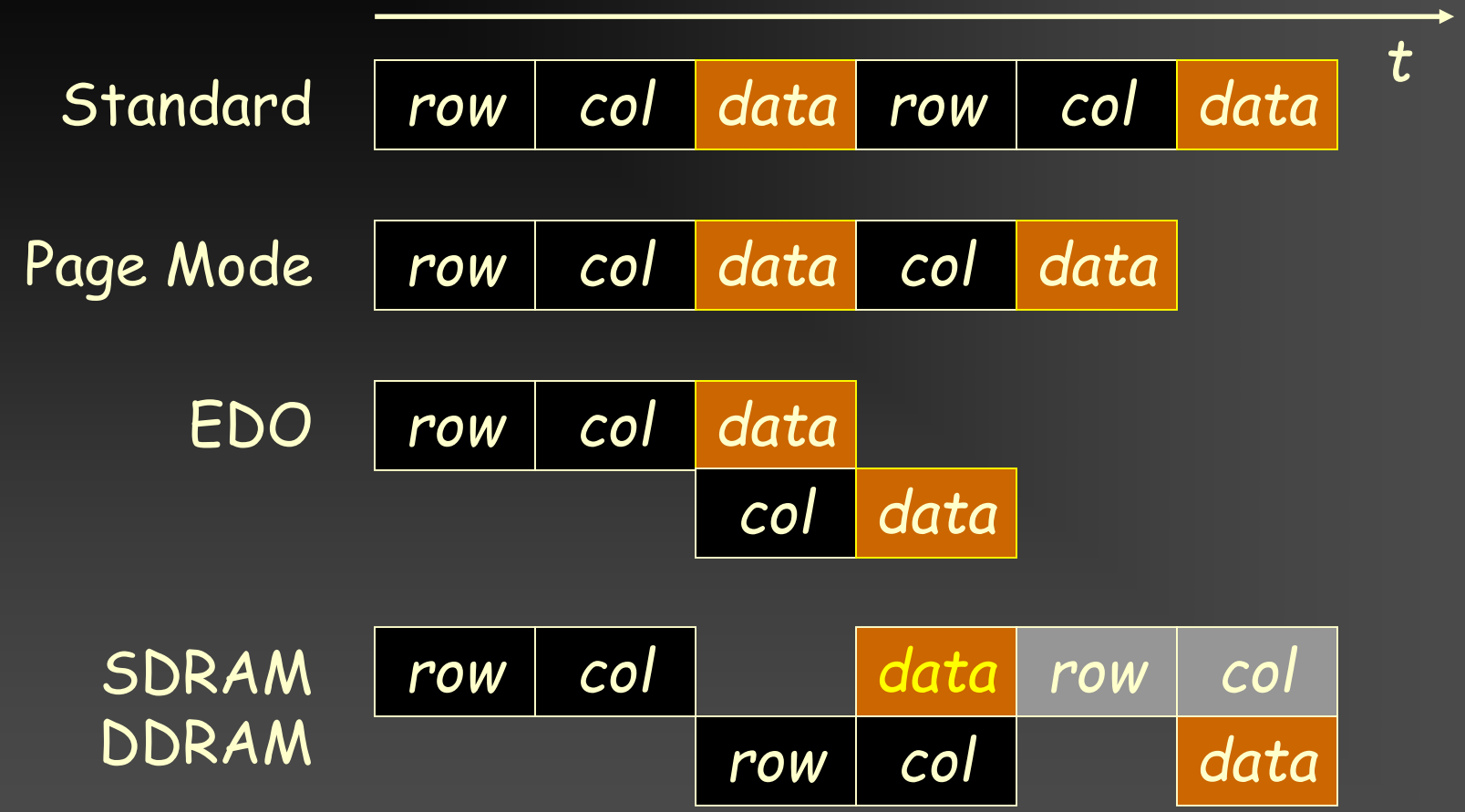
© A. Steininger / TU Wien

Synchrones DRAM & DDRAM

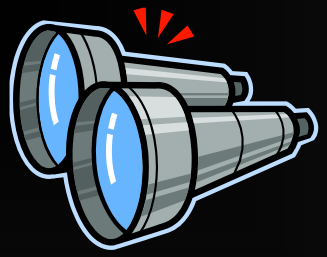
- Das Timing des DRAM läuft synchron zu einem Takt (dzt. 133...400MHz) ab.
- Das spart Probleme mit der Synchronisation in schnellen Designs
- RAM-intern sind mehrere Taktschritte für einen Zugriff nötig (=> „Wartezeit" nach außen)
- Durch interne Organisation in Bänken können jedoch mehrere Zugriffe überlappend ablaufen
- Beim DDRAM werden zusätzlich noch beide Taktflanken (↑ und ↓) verwendet.

© A. Steininger / TU Wien

Zugriffsarten (schematisiert)



© A. Steininger / TU Wien



Überblick

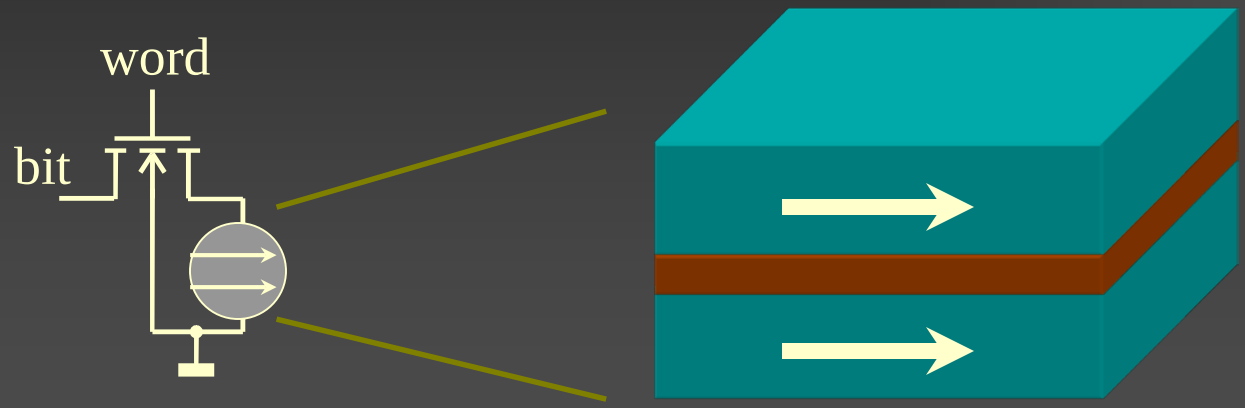
- Funktionsprinzipien von Speichern
- Klassifikation von Speichern
- ROM, SRAM, **DRAM**, **MRAM: Aufbau**, Varianten, **Anwendung**, Timing
- Multiport-RAM und FIFO
- Error Detection & Correction

© A. Steininger / TU Wien

Aufbau des SDT-MRAM



- Spin-Dependent Tunneling (SDT)
  - Zelle = Sandwich aus 2 magnetischen Schichten mit extrem dünnem Isolator dazwischen (nm).
  - Isolator ist so dünn, dass „Tunnelstrom" hindurchfließt (quantenmechan. Effekt).



© A. Steininger / TU Wien

Prinzip des SDT-MRAM

- Schreiben:
  - Magnetische Polarisation der Schichten wird durch Stromrichtung festgelegt.
- Lesen:
  - Je nach magnet. Ausrichtung der beiden Schichten zueinander ergeben sich unterschiedlich große Tunnelströme



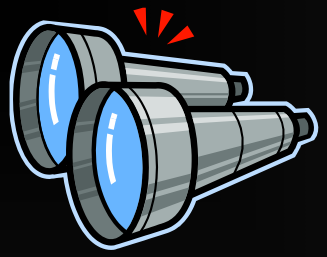
≈ magnetfeldabhängiger Widerstand „magnetoresistives Verhalten".

© A. Steininger / TU Wien

MRAM - Features

- random access
- non-volatile (magnetische Speicherung)
- >10^{10} WR Zyklen
- etwa so schnell wie DRAM
- 40% kleiner als DRAM
- stromsparend (Energie nur für WR & RD)
- Marktreife schon länger erwartet

© A. Steininger / TU Wien



Überblick

- Funktionsprinzipien von Speichern
- Klassifikation von Speichern
- ROM, SRAM, DRAM:
  Aufbau, Varianten, Anwendung, Timing
- **Multiport-RAM und FIFO**
- Error Detection & Correction

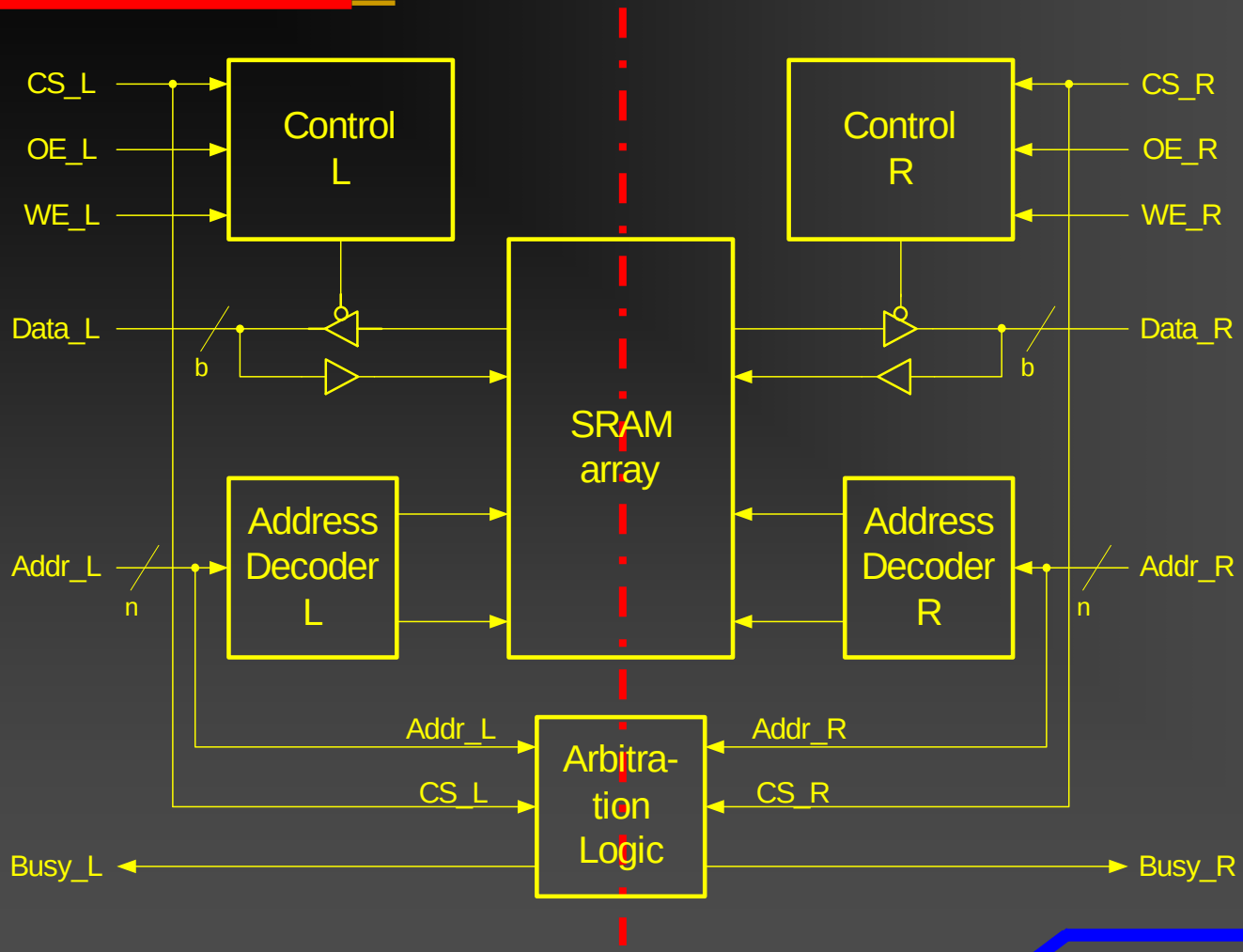
© A. Steininger / TU Wien

Multiport Memory - Prinzip

- Beispiel Dual-Port RAM
  - Gemeinsames RAM-Array, aber
  - Doppelte Steuerlogik (links / rechts) für CS, R/W, OE
  - Doppelter Adressdecoder
  - Doppelte Interfaces (Pins)
  - Zusätzliches Busy-Signal

© A. Steininger / TU Wien

Multiport-Memory - Aufbau



© A. Steininger / TU Wien

Semaphor-Bit

- Erlaubt Koordination des Zugriffs auf gemeinsame Ressourcen (z.B. Speicher)
- Durch Setzen des Bits kann Zugriff auf Ressource angefordert werden (Request), gleichzeitig werden konkurrierende Zugriffe blockiert, d.h. Konkurrenten können ihr Bit nicht mehr setzen.
- Löschen des Semaphore-Bit (Release) gibt die Ressource wieder frei.
- Beim DPRAM oft mit integriert, aber Beachtung des Semaphor wird nicht per HW erzwungen, d.h. Zugriff auf blockierte Ressource ist möglich.

© A. Steininger / TU Wien

Funktion des Semaphor

- Default (frei):
  - SemL = SemR = 0
- Sem. „Lock“ durch L:
  - WR SemL => 1
  - RD SemL ergibt „1“
  - WR SemR => 1
    wird ignoriert
  - RD SemR ergibt „0“
- Release von L:
  - WR 0 auf SemL

Semaphor-Bit „links“ Semaphor-Bit „rechts“

Aktion	L	R	Status RAM
init	0	0	frei
Request L	1	0	Locked für L
Request R	1	**0**	Locked für L (!)
Release L	0	0	frei
Request R	0	1	Lock für R
Release R	0	0	frei

© A. Steininger / TU Wien

FIFO - Prinzip

- FIFO = „First In First Out“
- keine Adresse => kein wahlfreier Zugriff
- Am WR-Port kann eine Sequenz von Datenworten in den Speicher geschrieben werden, am RD-Port können die Datenworte genau in der gleichen Sequenz wieder („konsumierend“) gelesen werden.
- Entsprechend der Speichertiefe wird eine Anzahl geschriebener Datenworte gepuffert
- Lesen und Schreiben sind voneinander unabhängig und können überlappen.

© A. Steininger / TU Wien

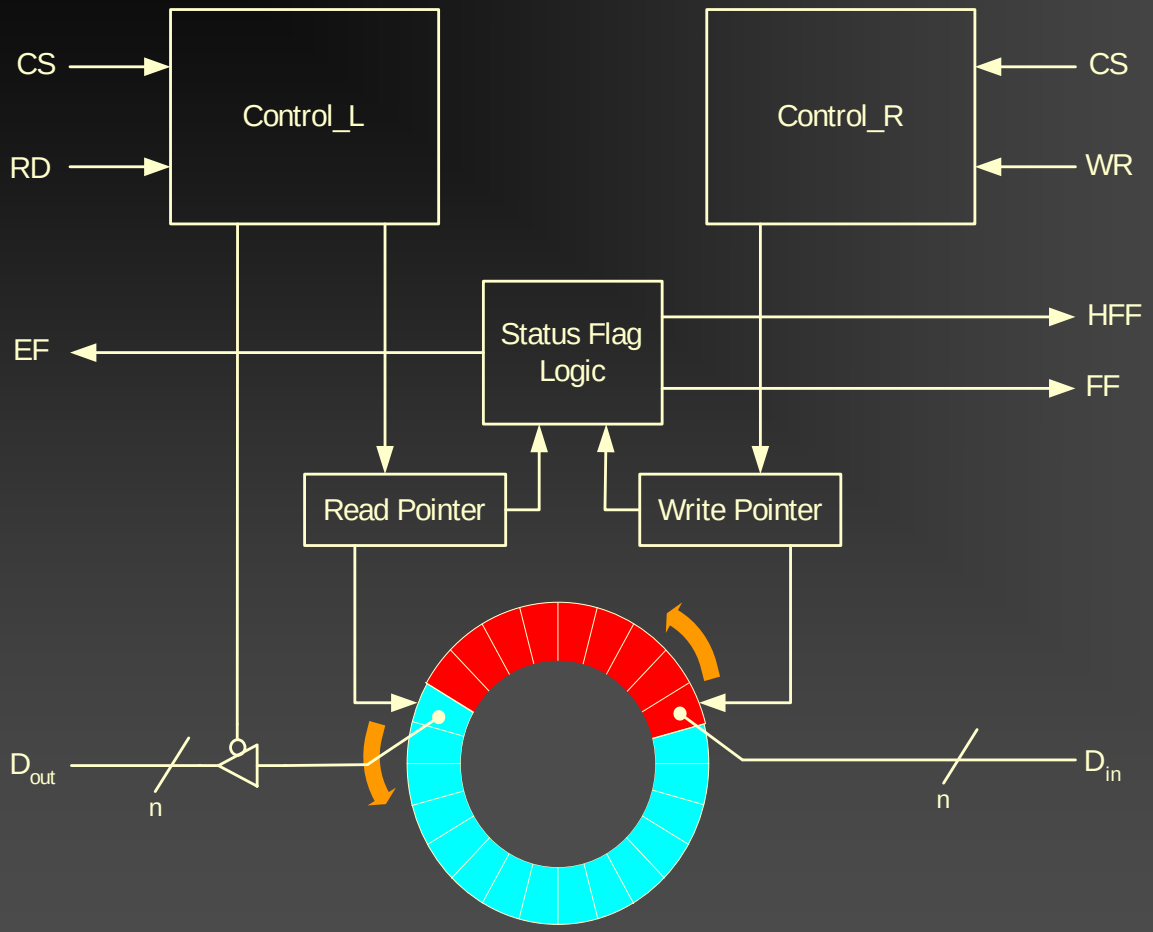
FIFO - Flags



- Full-Flag (FF) wird aktiviert, wenn alle verfügbaren Speicherzellen (Speichertiefe des FIFO) beschrieben und noch nicht gelesen sind.
  ➡ Vor dem nächsten WR muss zuerst gelesen werden, sonst wird der jüngste Eintrag überschrieben.
- Empty-Flag (EF) wird aktiviert, wenn keine Daten im Speicher verfügbar sind, die noch nicht gelesen wurden.
  ➡ Vor dem nächsten RD müssen Daten geschrieben werden, sonst wird ein undefinierter Wert gelesen.
- Half-Full-Flag (HFF) wird aktiviert, sobald das FIFO zur Hälfte voll ist.
  ➡ als „Vorwarnung" für Speicherverwaltung verwendbar

© A. Steininger / TU Wien

FIFO - Aufbau



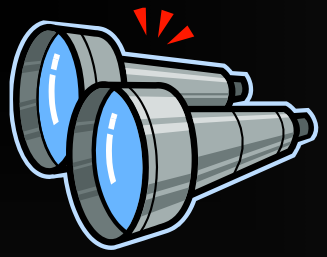
© A. Steininger / TU Wien

Anwendung DPRAM & FIFO

... typischerweise für die Kommunikation zweier asynchron arbeitender Systeme:

- Es können Daten ausgetauscht werden
- die Kopplung im Zeitverhalten der Systeme ist minimal (nur bei FIFO Full/Empty)
- Für kontinuierliche Datenströme ist FIFO besser (weil einfacher & billiger), ggf. für bidirektionale Kommunikation 2 FIFOs
- Beim DPRAM stellt die Adresse zusätzlich semantische Information dar (z.B.: Sensorwert 1 liegt immer auf Adresse 0x01 etc.)

© A. Steininger / TU Wien



Überblick

- Funktionsprinzipien von Speichern
- Klassifikation von Speichern
- ROM, SRAM, DRAM:
  Aufbau, Varianten, Anwendung, Timing
- Multiport-RAM und FIFO
- **Error Detection & Correction**

© A. Steininger / TU Wien

Error Detection & Correction



Prinzip: Hinzufügen redundanter Information (= Prüfbits) entsprechend einem Code so dass nicht alle darstellbaren Codeworte auch gültige Codeworte sind, Prüfbits werden beim WR generiert und mitgespeichert, beim RD neuerlich generiert und mit den gespeicherten verglichen

- Hamming-Distanz: minimale Anzahl von gleichzeitig auftretenden Bitfehlern die im ungünstigsten Fall nicht mehr erkannt werden
- EDC: Error Detecting Code
  Bitfehler (auch in der Prüfinformation) werden erkannt
- ECC: Error Correcting Code
  Bitfehler werden erkannt UND korrigiert

© A. Steininger / TU Wien

Typische Verfahren

- Parity: 1 Prüfbit ergänzt die Bitsumme der *n* Datenbits auf gerade („even") bzw. ungerade („odd"), typisch 1 Bit/Byte
  Hamming-Distanz = 2, d.h. nur das Umfallen eines einzigen Bits wird mit Sicherheit erkannt
- Hamming Code: mehrere Parity-Bits jeweils über Gruppen von Datenbits erlauben Hamming-Distanz von 3 sowie Korrektur für 1-bit Fehler.
- Block-Signatur (cyclic redundancy check, CRC):
  Generiert Prüfwort aus sequentiellem Datenstrom (Block) mittels rückgekoppeltem Schieberegister. Üblicherweise bei EPROMs, da bei WR im RAM schwer nachzuführen.

© A. Steininger / TU Wien

Parity: Beispiel

Datenwort 11001101

Parity generieren

$P = d0 \oplus d1 \ldots \oplus dn$

0

Datenwort 1100 1101 1

+ even Parity

Parity prüfen

$C = d0 \oplus d1 \ldots \oplus dn \oplus P$

Codewort hat gerade Anzahl von „1en“ ?

✓Nein

© A. Steininger / TU Wien

Beispiel: (7,4) Hamming Code:

- Codierung:

  Datenwort: d0 d1 d2 d3

  Codewort: p0 p1 d0 p2 d1 d2 d3

  Prüfbits:
  $p0 = d0 \oplus d1 \oplus d3$
  $p1 = d0 \oplus d2 \oplus d3$
  $p2 = d1 \oplus d2 \oplus d3$

- Prüfung:

  Checkbits:
  $c0 = p0 \oplus d0 \oplus d1 \oplus d3 = 0?\ 1$
  $c1 = p1 \oplus d0 \oplus d2 \oplus d3 = 0?\ 1$
  $c2 = p2 \oplus d1 \oplus d2 \oplus d3 = 0?\ 0$

011 = Bitposition 3

© A. Steininger / TU Wien

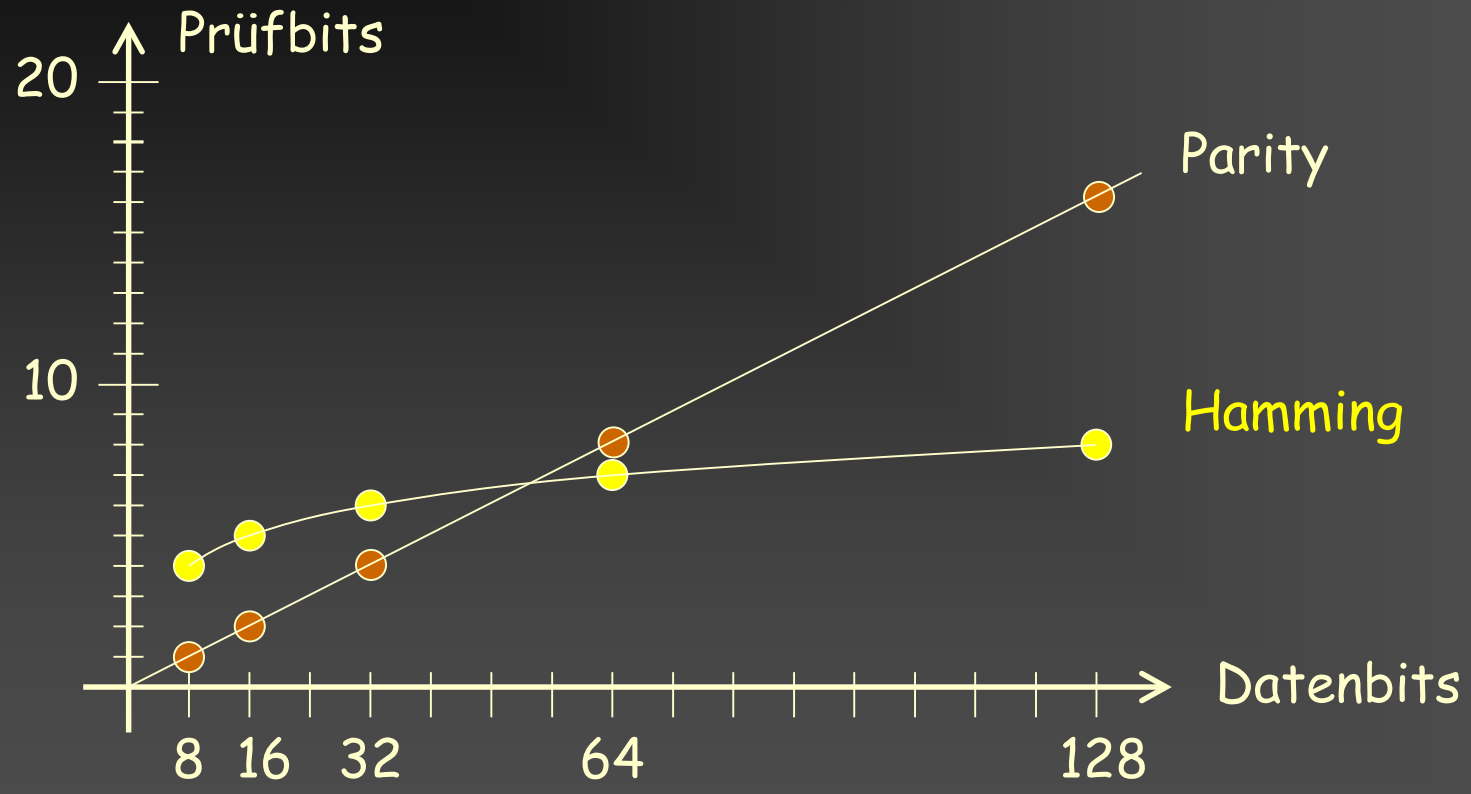
Hamming Code: Grenzen

- mehr als 2-Bit-Fehler nicht sicher erkannt
  (außer durch zus. Prüfbits)
- 0000000 und 1111111 sind gültige Codeworte
  Beispiel: 1010010 -> 0000000 nicht erkannt
- Gefahr falscher Korrekturen
  Beispiel: korrekt 1010010
  falsch 1110110
  korrigiert 1100110

© A. Steininger / TU Wien

Hamming-Code: Aufwand

- r Prüfbits für 2^r Bits (Daten + Prüfbits)



© A. Steininger / TU Wien

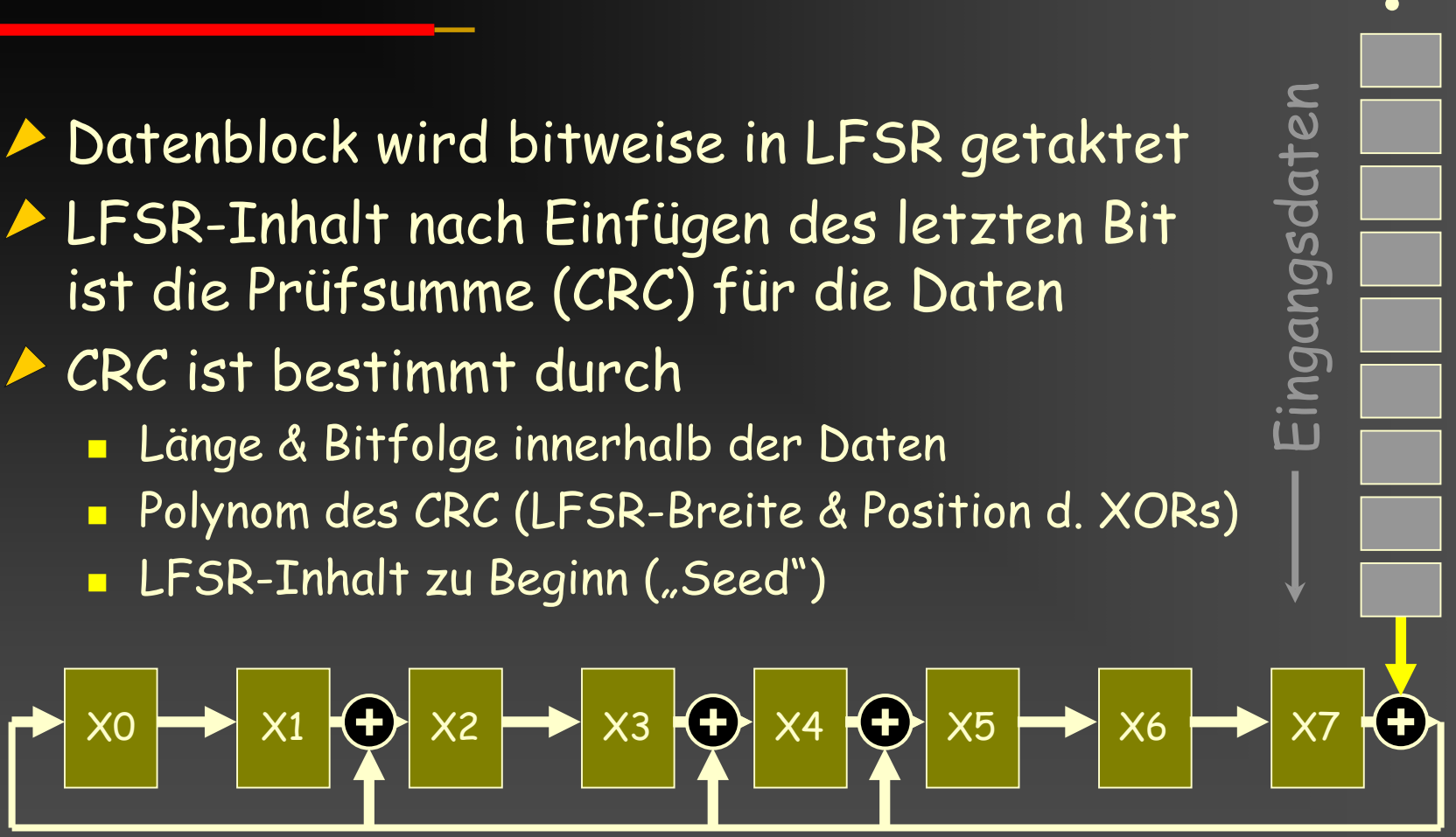
Typische Verfahren

- Parity: 1 Prüfbit ergänzt die Bitsumme der *n* Datenbits auf gerade („even") bzw. ungerade („odd"), typisch 1 Bit/Byte Hamming-Distanz = 2, d.h. nur das Umfallen eines einzigen Bits wird mit Sicherheit erkannt
- Hamming Code: mehrere Parity-Bits jeweils über Gruppen von Datenbits erlauben Hamming-Distanz von 3 sowie Korrektur für 1-bit Fehler.
- Block-Signatur (cyclic redundancy check, CRC): Generiert Prüfwort aus sequentiellem Datenstrom (Block) mittels rückgekoppeltem Schieberegister. Üblicherweise bei EPROMs, da bei WR im RAM schwer nachzuführen.

© A. Steininger / TU Wien

Bildung eines CRC in Hardware

- Datenblock wird bitweise in LFSR getaktet
- LFSR-Inhalt nach Einfügen des letzten Bit ist die Prüfsumme (CRC) für die Daten
- CRC ist bestimmt durch
  - Länge & Bitfolge innerhalb der Daten
  - Polynom des CRC (LFSR-Breite & Position d. XORs)
  - LFSR-Inhalt zu Beginn („Seed“)



© A. Steininger / TU Wien

Schutz durch CRC



- CRC wird an Daten angefügt und kann jederzeit neu berechnet & geprüft werden
- bietet Schutz gegen Bitfehler
- bietet keinen unbegrenzten Schutz
  (Es gibt mehrere – daher auch falsche – Datenblöcke die zum gleichen CRC führen !)
- kein Schutz gegen bösartige Manipulation (Änderung, Zugriff) im Sinne von Security!

© A. Steininger / TU Wien



Zusammenfassung (1)

- Digitale Speicher sind eine strukturierte Anordnung von Bits mit zwei klar unterscheidbaren Zuständen.
- Neben Aufbau und physikalischem Speicherprinzip sind Volatility, Beschreibbarkeit und Random Access charakteristische Merkmale.
- Neben ihrer eigentlichen Funktion als Datenspeicher sind Speicher auch für vielfältige logische Funktionen verwendbar.
- Zur Vereinfachung der Decodierlogik sind Speicher als zweidimensionales Array strukturiert.

© A. Steininger / TU Wien



Zusammenfassung (2)

- Bei den ROMs unterscheidet man zwischen Mask-ROM, OTP, UV-EPROM und EEPROM.
- Das SRAM beruht wie das Latch auf einer Speicherschleife aus rückgekoppelten Invertern, ist aber platzsparender.
- Das DRAM verwendet einen Kondensator als Speicherelement, eine Speicherzelle ist daher deutlich kleiner als beim SRAM. Es ist allerdings ein periodischer Refresh erforderlich.

© A. Steininger / TU Wien



Zusammenfassung (3)

- Das MRAM ist ein schneller nicht-flüchtiger Speicher. Es beruht auf magnetischer Polarisation.
- Multiport-Speicher und FIFO erlauben die Kopplung unsynchronisierter Systeme.
- Durch Hinzufügen von Prüfbits können Bitfehler erkannt und eventuell auch korrigiert werden. Übliche EDC/ECC-Verfahren sind Parity, Hamming Code und CRC.

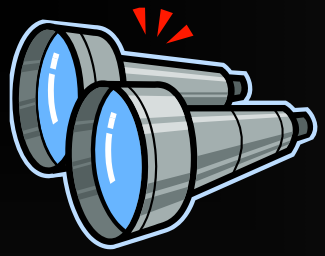
© A. Steininger / TU Wien

Zieltechnologien

Welcher ASIC-Typ passt zu meinem Design?



© A. Steininger / TU Wien



Überblick

- Terminologie
- ASIC-Typen und ihre Charakteristika
- Programmable Logic Devices: Prinzipien
- Programmierbare Logikzellen
- Programmierbare I/O-Zellen
- Programmierbarer Interconnect

© A. Steininger / TU Wien

Was ist ein ASIC ?

Application **S**pecific **I**ntegrated **C**ircuit

10% der Umsätze

Beispiele: PC-Chipset, Spielzeug, Satellit

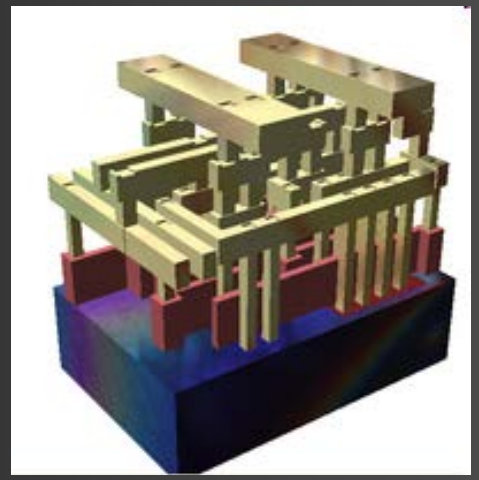
Gegenbeispiele: Pentium, DRAM, 74xxx

90% der Umsätze

„Standard-ICs"

© A. Steininger / TU Wien

Aufbau eines Die



- Interconnect:

  für die Verbindungen stehen weitere 5...12 Layers zur Verfügung (Metallisierung)

- Zellen:

  die Transistoren / Zellen benötigen 5...10 Layers (Diffusionsprozesse auf dem Silizium)

  für jeden Layer gibt es eine „Maske"

© A. Steininger / TU Wien

Full-Custom ASIC

- alle Masken (Zellen + Interconn.) sind völlig anwenderspezifisch
  - \+ beliebig optimierbar (Fläche, Leistungsaufnahme, Geschwindigkeit)
  - – besonders hoher Aufwand (Design, Test, Fertigung)
  - – keine Garantie bei der Fertigung

➡ Anwendung nur in Sonderfällen

© A. Steininger / TU Wien

Standard-Cell ASIC (CBIC)

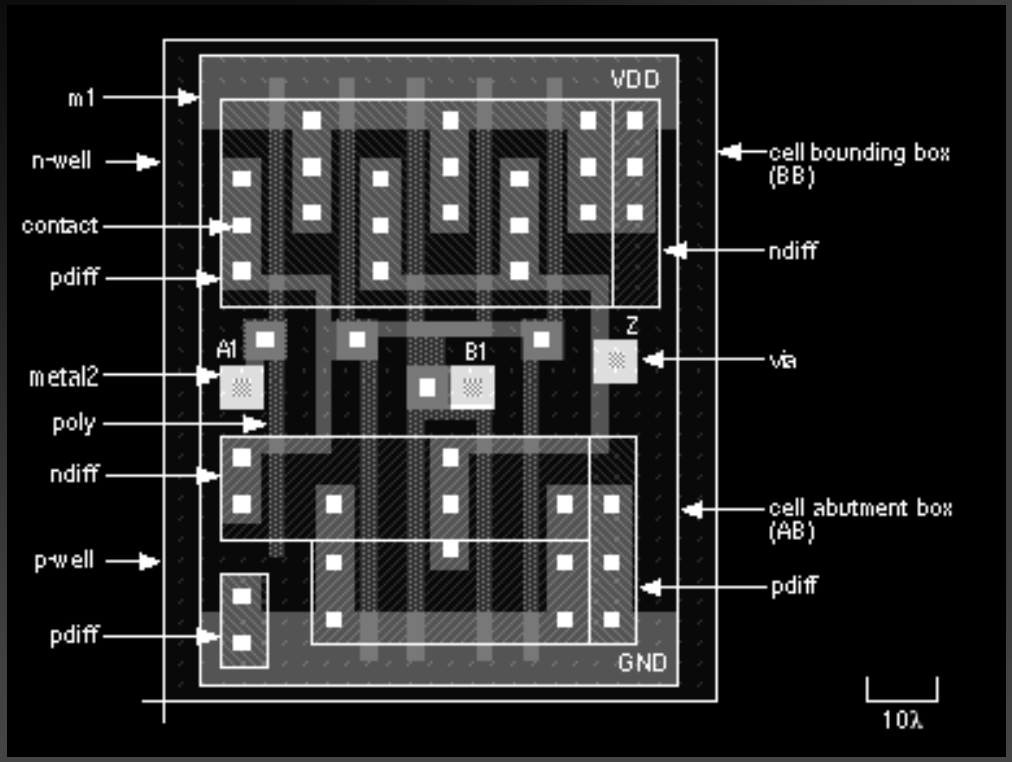
- Zellen in „Library“ vordefiniert
- Design verwendet nur Elemente daraus
- kann auch Cores („Mega-Cells“) enthalten
  z.B. RAM, ROM, IP-Core
- Anordnung & Interconnect anw.-spezif.

\+ Entwicklung viel effizienter
(Zellen fertig entwickelt, optimiert & getestet)

– in der Fertigung immer noch hoher Aufwand + Wartezeit (*alle* Masken anwenderspezifisch)

© A. Steininger / TU Wien

Standard-Cell – ein Beispiel

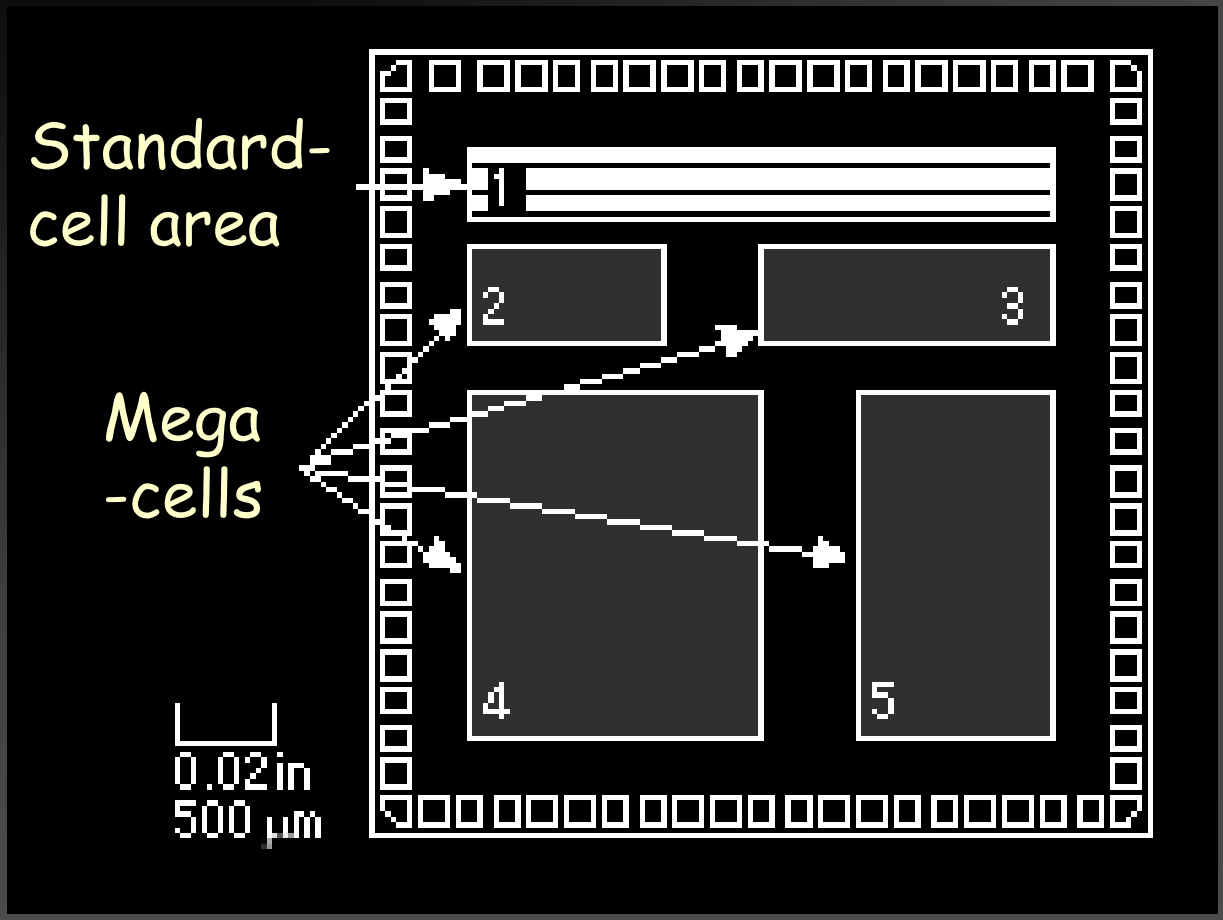


internes Layout schon vordefiniert und getestet

„Standard-Cells passen zusammen wie Ziegel in einer Wand"

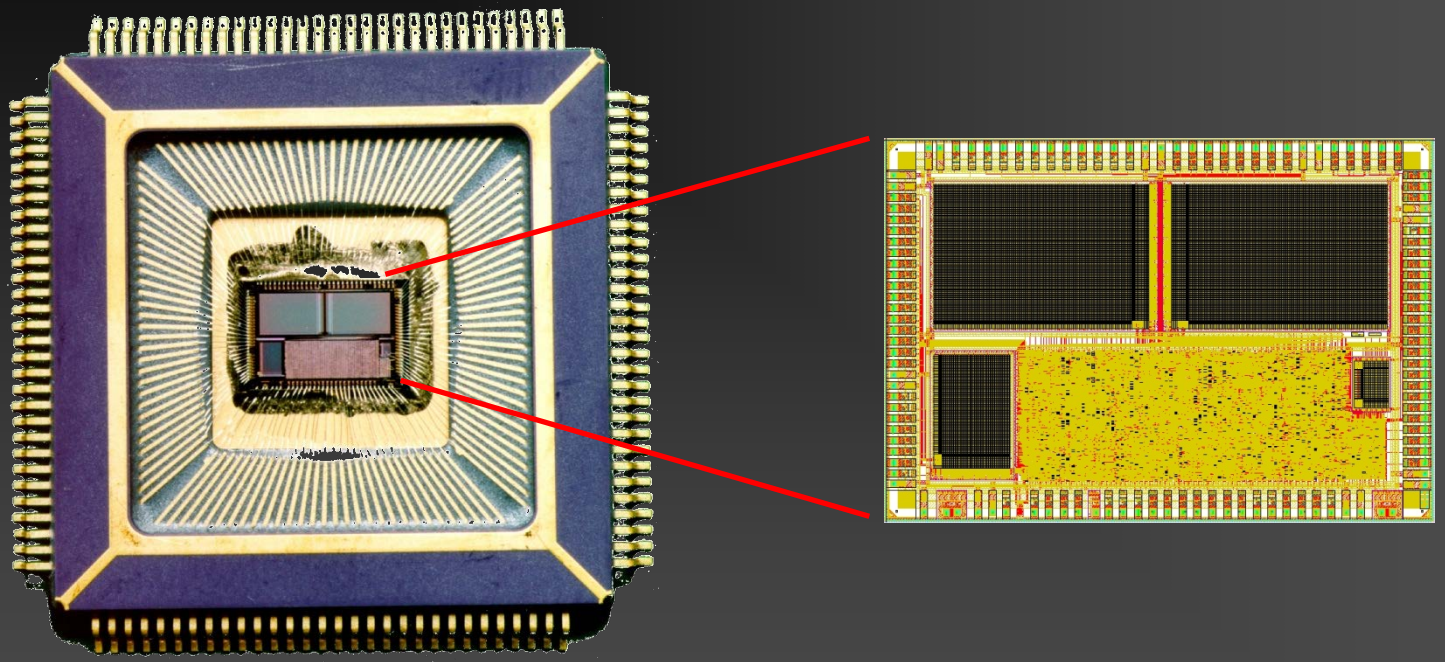
© A. Steininger / TU Wien

Standard-Cell ASIC



© A. Steininger / TU Wien

Standard-Cell ASIC – Beispiel



© A. Steininger / TU Wien

Zellen-Library



- hunderte von Funktionen
  AND, OR, FFs mit verschiedenen Optionen, ...
- fertig spezifiziert aus Datenbuch wählbar
  Funktion, Layout, Timing, Simulationsmodell...
- für
  - Standard-Cells
  - Gate-Array-Macros und auch
  - FPGAs (Hard & Soft Macros)
- erstellt
  - meist vom ASIC-Hersteller
  - oder von einem Library-Vendor
- Beispiel: LSI_10k.lib

© A. Steininger / TU Wien

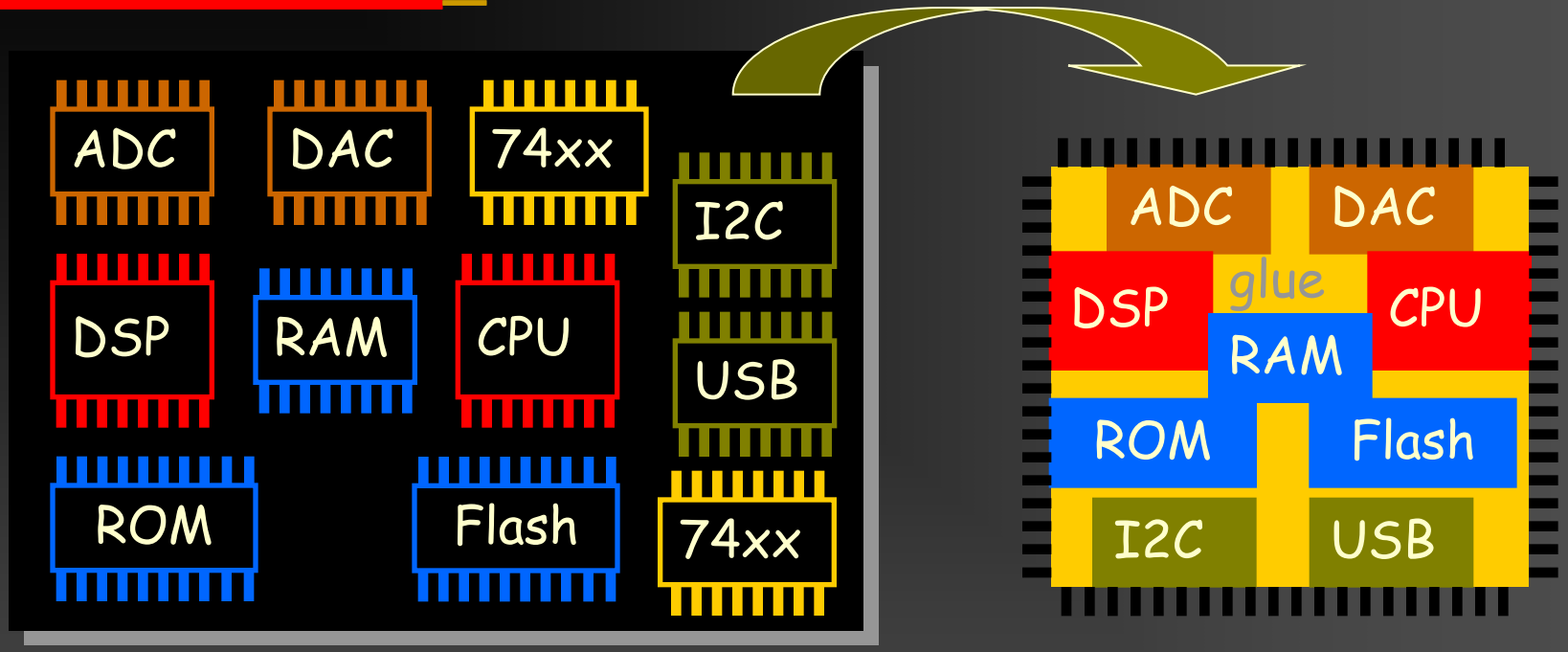
Was ist ein IP-Core?

- Eine fertig entwickelte und spezifizierte komplexe Funktionseinheit, die als Macro in das Design eingebunden werden kann (z.B. UART, Microcontroller; vgl. IC auf einer Platine)
- Kann beim Designer des IP-Core (IP = Intellectual Property) gekauft werden
  - Hard Macro: fertig geroutete „Black Box"
  - Soft Macro: nur Netzliste, technologieunabh.
- Spart Entwicklungsaufwand, erhöht Produktivität
- beliebt bei System on a Chip

© A. Steininger / TU Wien

Wh...

IP-Cores & System on a chip



- alle für die Anwendung benötigten Funktionen werden auf einem Chip (Die) untergebracht

diese Funktionen sind oft als IP-Cores realisiert

© A. Steininger / TU Wien

Gate-Arrays (MGAs)

- vorgefertigte Wafer enthalten „Basiszellen“ (definierte Anordnung von Transistoren) in regelmäßiger Anordnung
- Zellen werden als Macros realisiert
- nur Verbindungen anwenderspezifisch

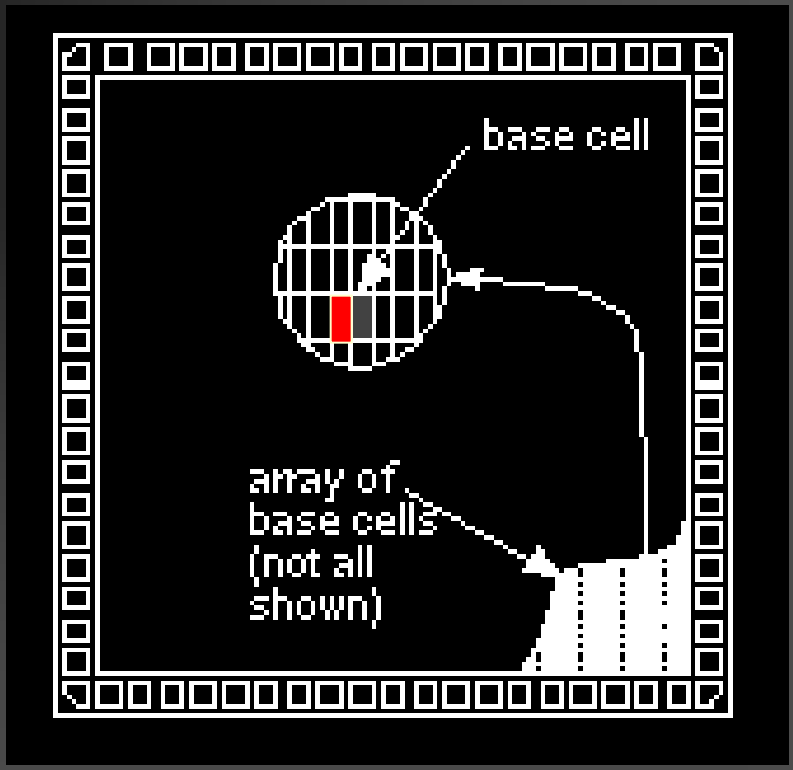
+ effiziente Entwicklung (weiterhin Cell-Library)
+ vorgefertigte Wafer können auf Lager gelegt werden => schneller und billiger
– Basiszellen fixer Größe => weniger optimierbar

© A. Steininger / TU Wien

Channelless Gate-Array

- Anschlüsse der Transistoren (contact layer) nicht vorgegeben.
- Zwischen den Basiszellen sind keine Kanäle freigehalten.
- Interconnect über unbenützte Transistoren geroutet.

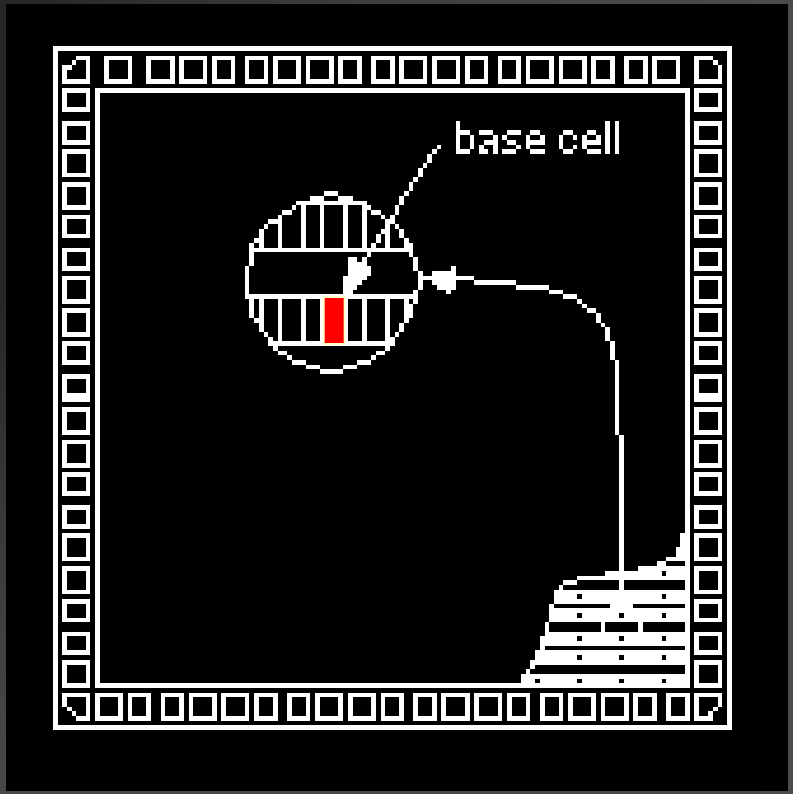
auch „Sea of Gates"



© A. Steininger / TU Wien

Channelled Gate-Array

- Anschlüsse der Transistoren („contact layer“) fix vorgegeben.
- Zwischen den Basis-zellen „Kanäle“ fixer Höhe freigehalten.
- Diese Kanäle stehen für Interconnect zur Verfügung.



© A. Steininger / TU Wien

Structured Gate-Array

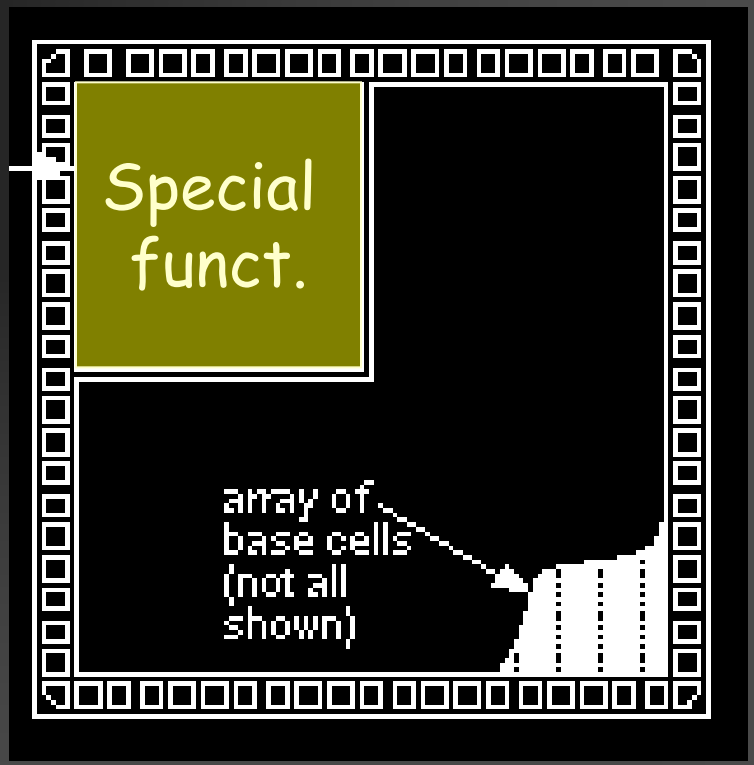
auch „embedded GA“

Teil der Chip-Fläche für

- spezielle Funktion (RAM, ROM) oder
- anderen Typ von Basiszelle

reserviert

meist verschiedene Varianten auf Lager



© A. Steininger / TU Wien

Structured GA vs. Std.-Cell

- Standardzellen sind weiter optimierbar als die Makros beim GA.
- Cores sind beim Standard-Cell-ASIC frei wählbar, beim Structured GA (in Funktion, Größe und Position) fix vorgegeben – Wafer sind ja vorgefertigt.
- Herstellung von Structured GA ist wesentlich schneller und billiger (Wafer vorgefertigt).

© A. Steininger / TU Wien

Programmable Logic Device

- *alle* Layer (Interconnect und Zellen) fix vorgegeben => Herstellung abgeschlossen
- vorgegebene Matrix aus „Makrozellen“
- Interconnect programmierbar

\+ billig, extrem kurze Entwicklungszeit
\+ Einfach änderbar (manchmal sogar on-line)
– Komplexität und Optimierbarkeit sehr beschränkt

Beispiele: ROM, PLA, PAL, CPLD, FPGA

© A. Steininger / TU Wien

Read Only Memory ROM

Logik (= Verbindungsmatrix) wandelt Adresse (= Eingang) in Daten (= Ausgang) um, Wahrheitstabelle programmierbar

- Programmierung:
  - elektrisch / löschbar: EPROM
  - elektrisch / permanent: PROM, OTP
  - mit Maske, als Core: mask-progr. ROM
- Löschen (nur für EPROM möglich):
  - mit UV-Licht: UV-EPROM
  - elektrisch: EEPROM (electr. erasable PROM)

© A. Steininger / TU Wien

Programmable Array Logic

kombinatorische Logik als Array aus AND-Gattern und OR-Gattern (logic array)

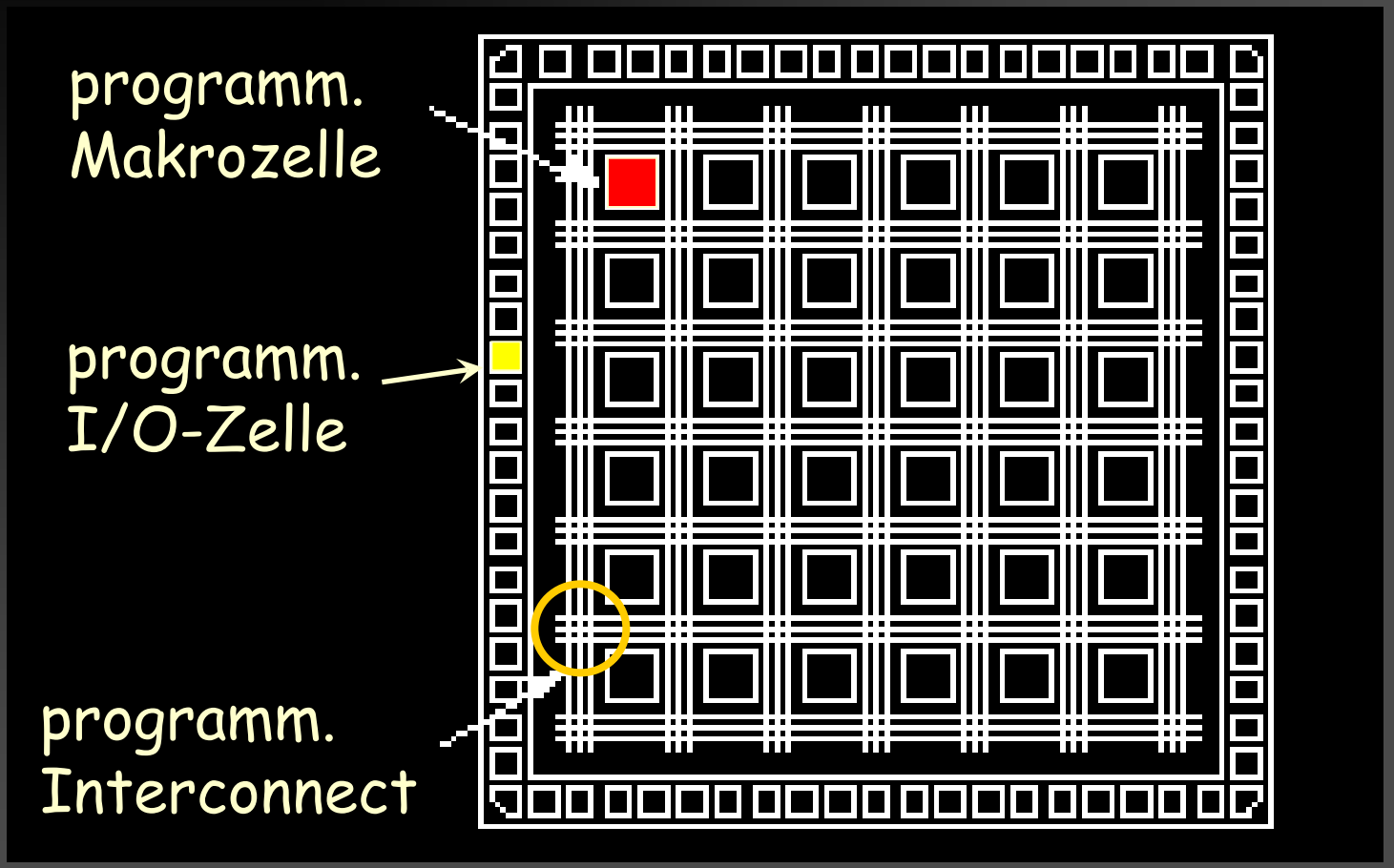
➡ Abbild der disjunktiven Normalform;

dahinter Speicherelement (Latch, FF)

- PAL: nur das AND-Array ist programmierbar
- PLA: AND und OR-Array sind programmierbar (meist als Core)

© A. Steininger / TU Wien

Field Programmable Gate-Array



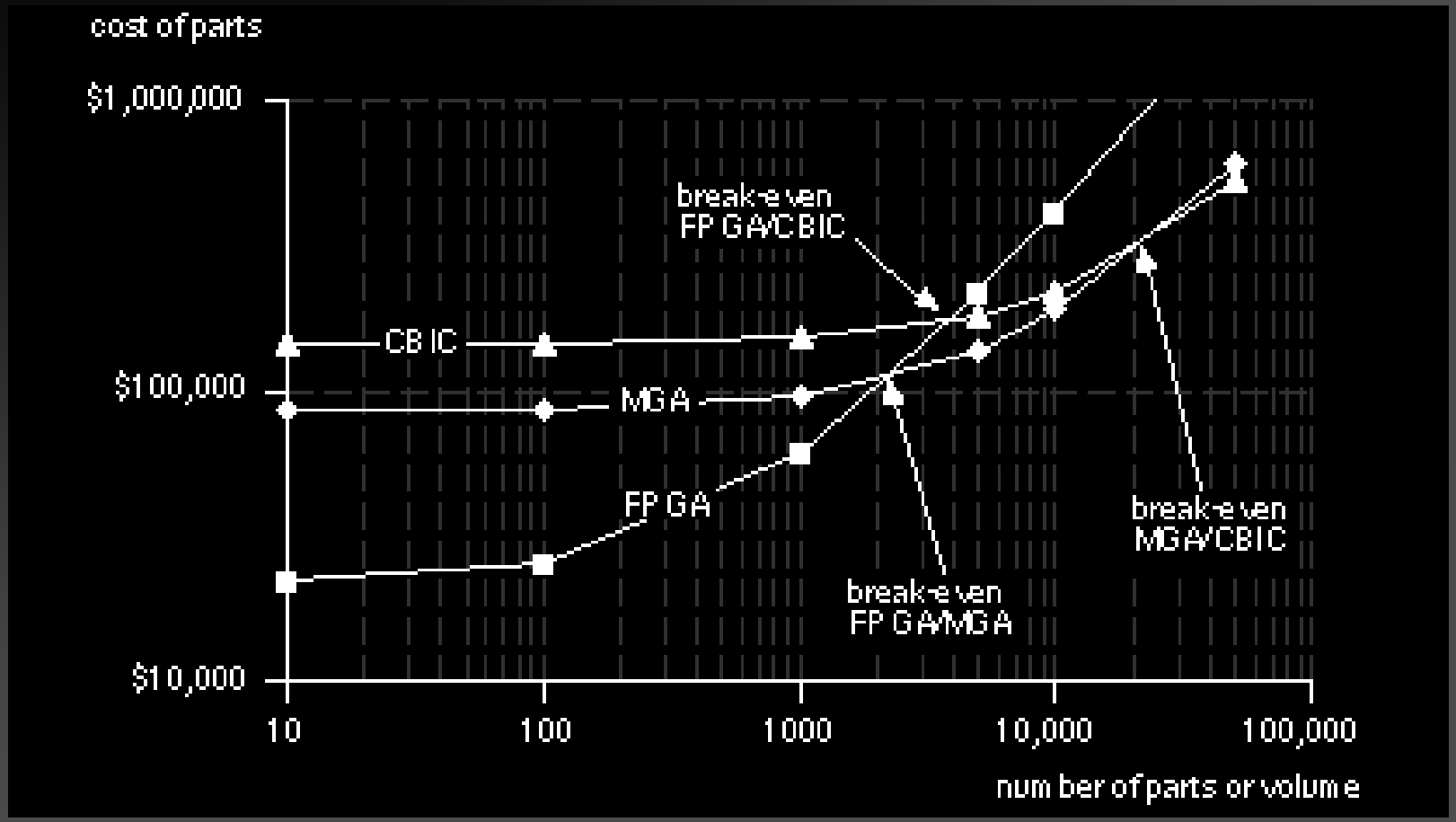
© A. Steininger / TU Wien

ASIC-Technologien – Überblick

			cell design	cell placemt	cell connect
Full Custom			*user*	*user*	*user*
Semi-custom	Standard Cell (CBIC)		*library*	*user*	*user*
	Gate-Array (MGA)	channelled channelless structured	*fixed / macros*	*fixed*	*user*
PLD		ROM PAL, PLA CPLD, FPGA	*fixed*	*fixed*	*fixed / progr.*

© A. Steininger / TU Wien

ASICs: Break-Even Analyse



© A. Steininger / TU Wien

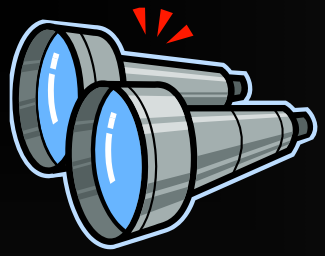
FPGA versus Standard Cell

Evaluation über einen Mix aus Design-Blöcken

	Faktor FPGA/ASIC
Fläche	8,5 … 70
Delay	1,9 … 6,7
dyn. Leistung	5,3 … 52
stat. Leistung	5,4 … 87

Vergleich aus [I.Kuon and J. Rose, Measuring the Gap between FPGAs and ASICs, Trans. on CAD, vol 26, no 2, 2/2007]

© A. Steininger / TU Wien



Überblick

- Terminologie
- ASIC-Typen und ihre Charakteristika
- **Programmable Logic Devices: Prinzipien**
- Programmierbare Logikzellen
- Programmierbare I/O-Zellen
- Programmierbarer Interconnect

© A. Steininger / TU Wien

ASIC-Technologien – Überblick

			cell design	cell placemt	cell connect
Full Custom			*user*	*user*	*user*
Semi-custom	Standard Cell (CBIC)		*library*	*user*	*user*
	Gate-Array (MGA)	channelled channelless structured	*fixed / mask*	*fixed*	*user*
PLD		ROM PAL, PLA CPLD, FPGA	*fixed*	*fixed*	*fixed / progr.*

© A. Steininger / TU Wien

Modell für den Gewinn



Annahmen:

- Zeitpunkt max. Kaufinteresses sowie Ende des Kaufinteresses unabh. von Einführung (Konkurrenz)
- Anstieg der Verkaufszahlen begrenzt (Produktionssteigerung)

© A. Steininger / TU Wien

FPGA:Was ist programmierbar?

Makrozellen	Position	fix (Array)
	innere Struktur	fix
	Funktion	prog.-bar
I/O-Zellen	Position	fix (Rand)
	innere Struktur	fix
	Funktion	prog.-bar
Verbindungen	Möglichkeiten	fix
	Auswahl	prog.-bar

© A. Steininger / TU Wien

Wie wird programmiert?

Die Konfiguration lässt sich vollständig durch

schaltbare Verbindungen

realisieren.

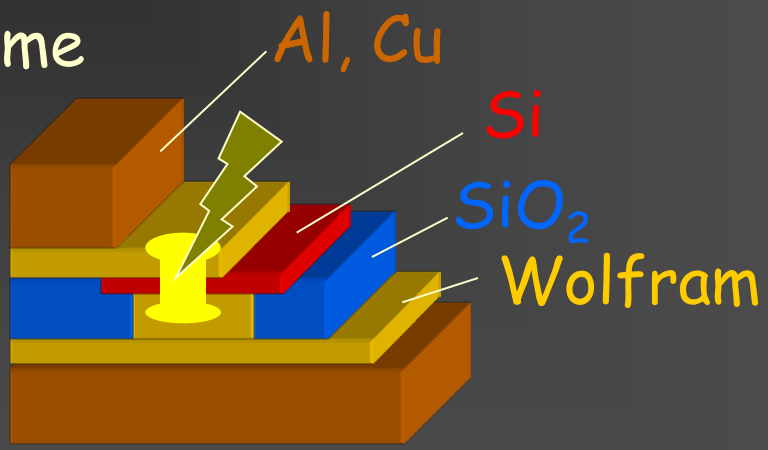
Varianten:

- Verbindungen (permanent) „brennen“
  Antifuse
- Transistor-Schalter ansteuern
  SRAM, EPROM

© A. Steininger / TU Wien

Antifuse-Konfiguration

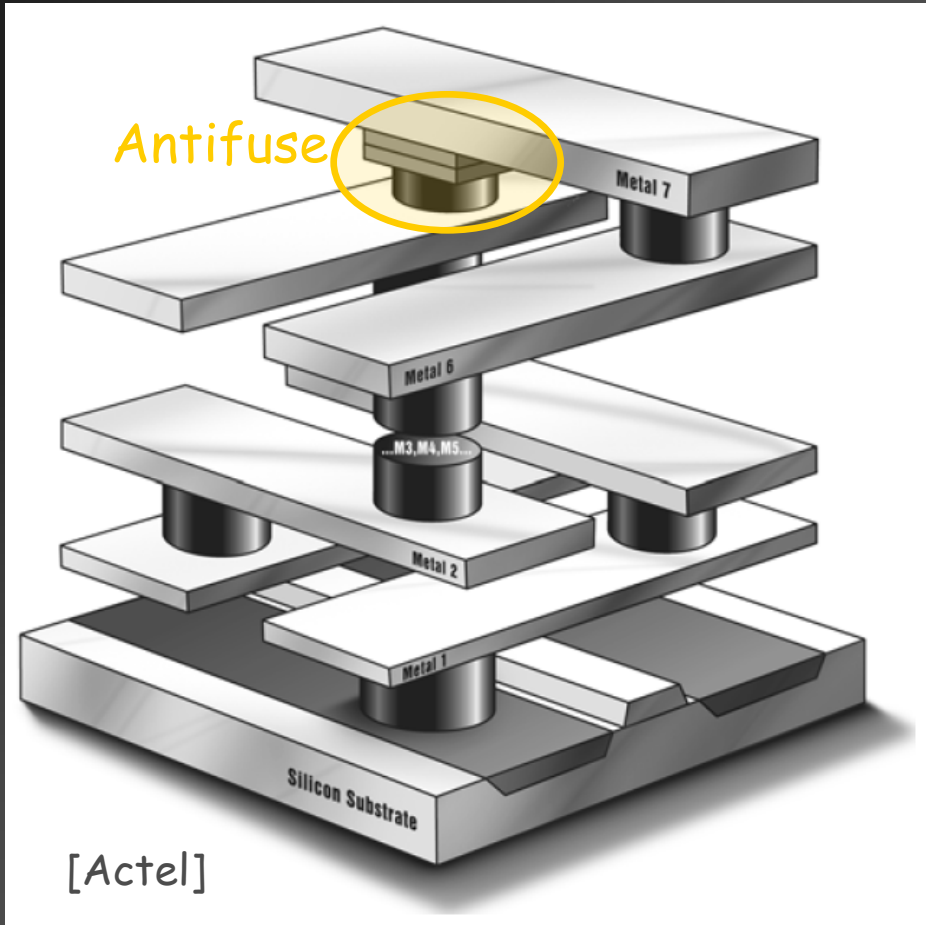
- Programmierung in eigenem Programmer
- Programmierstrom führt zu thermischer Zerstörung einer Isolationsschicht => Kontakt
- Irreversibel / OTP (one time programmable)
- Non-volatile & Radiation hard
- Mögliche Alterungsprobleme durch Elektromigration
- „Kontaktwiderstand" ist kritischer Parameter
- Beispiel: div. Actel



© A. Steininger / TU Wien

Vorteile der Antifuse-Techn.

- radiation hard
- Kopierschutz
- Schutz gegen Reverse Engineering
- klein
- allein bootfähig (kein ext. Speicher)



[Actel]

© A. Steininger / TU Wien

SRAM-Konfiguration

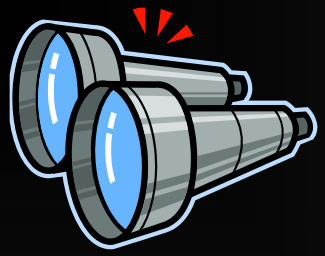
- Speicher-Bitzelle steuert FET bzw. TG auf/zu
- In-System-Programmierung (ISP) möglich (vom PC aus oder aus PROM)
- Reconfigurable Hardware (= im Betrieb!)
- Reversibel / löschbar
- Volatile / Neuprogrammierung nach Abschalten
- störanfällig, nicht Radiation hard
- SRAM hat viel höheren Platzbedarf als Antifuse
- Beispiele: Xilinx Virtex, Altera Stratix („FPGAs")

© A. Steininger / TU Wien

EPROM-Konfiguration

- EPROM-Bitzelle steuert FET bzw. TG auf/zu
- Programmierung & Löschen wie EPROM
- EEPROM: Elektrisch löschbar, In-System-Programmierung (ISP) möglich
- Non-volatile, aber löschbar (außer OTP-Typen)
- störanfällig, nicht Radiation hard
- EPROM hat ähnlichen Platzbedarf wie Antifuse
- Beispiele: Xilinx XC9500, Altera MAX („CPLDs")

© A. Steininger / TU Wien



Überblick

- Terminologie
- ASIC-Typen und ihre Charakteristika
- Programmable Logic Devices: Prinzipien
- **Programmierbare Logikzellen**
- Programmierbare I/O-Zellen
- Programmierbarer Interconnect

© A. Steininger / TU Wien

FPGA:Was ist programmierbar?

Makrozellen	Position	fix (Array)
	innere Struktur	fix
	Funktion	prog.-bar
I/O-Zellen	Position	fix (Rand)
	innere Struktur	fix
	Funktion	prog.-bar
Verbindungen	Möglichkeiten	fix
	Auswahl	prog.-bar

© A. Steininger / TU Wien

Programmierbare Logikzellen

Angewandte Prinzipien:

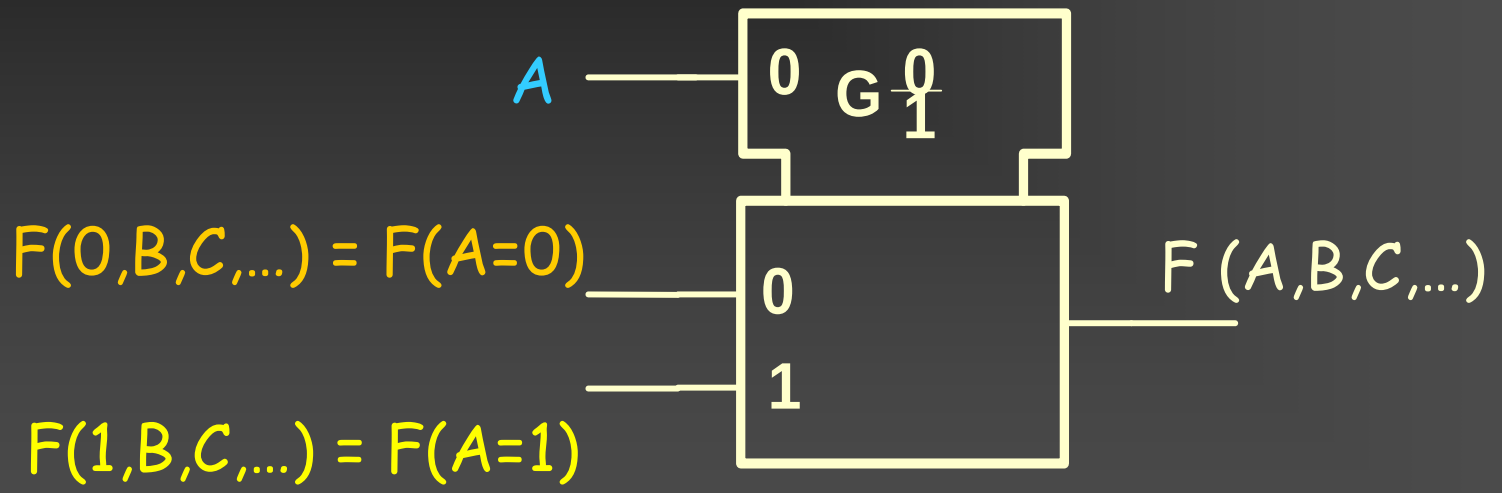
- Mux-basierte Logikzellen
  - Beispiel: Actel Axcelerator
- Look-up Table-basierte Logikzellen
  - Beispiele: Xilinx Virtex, Altera Stratix
- PAL-basierte Logikzellen
  - Beispiel: Altera MAX, Xilinx XC9500

© A. Steininger / TU Wien

Mux-basierte Logik: Prinzip

Shannon's Erweiterungstheorem:

$$F(A,B,C\ldots) = [\neg A \wedge F(0,B,C,\ldots)] \vee [A \wedge F(1,B,C,\ldots)]$$

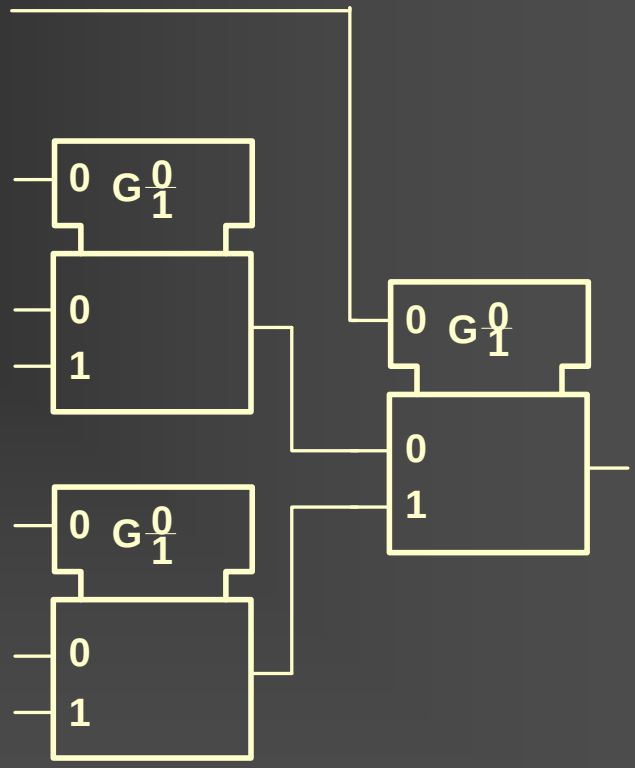


© A. Steininger / TU Wien

Mux-basierte Logik: Beispiel

$F = (A \wedge B) \vee (\neg B \wedge C) \vee D$

$F = [\neg B \wedge (C \vee D)] \vee [B \wedge (A \vee D)]$



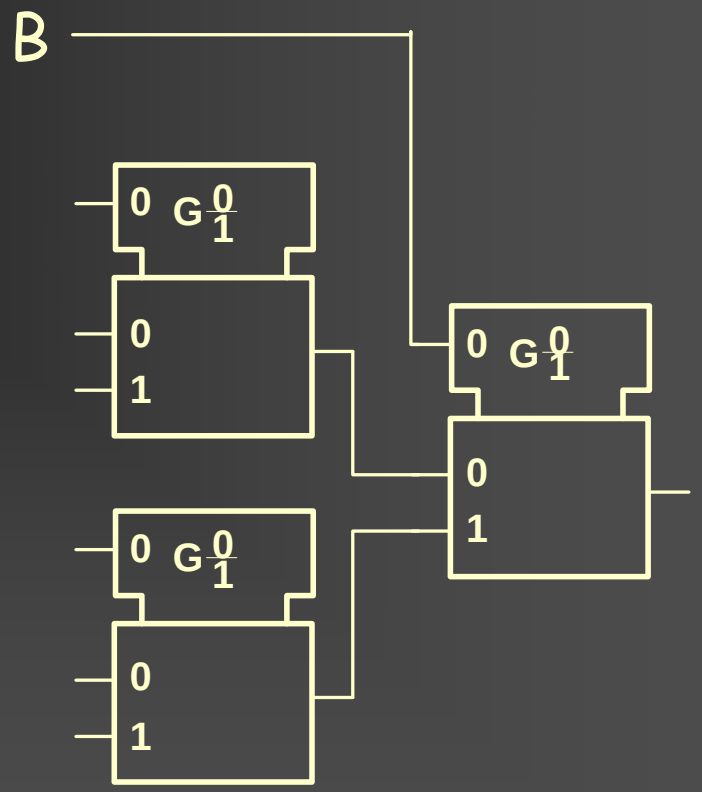
© A. Steininger / TU Wien

Mux-basierte Logik: Beispiel

$F = (A \wedge B) \vee (\neg B \wedge C) \vee D$

$F = [\neg B \wedge (C \vee D)] \vee [B \wedge (A \vee D)]$

$F_{\overline{B}} = [\neg C \wedge D)] \vee [C \wedge 1]$



© A. Steininger / TU Wien

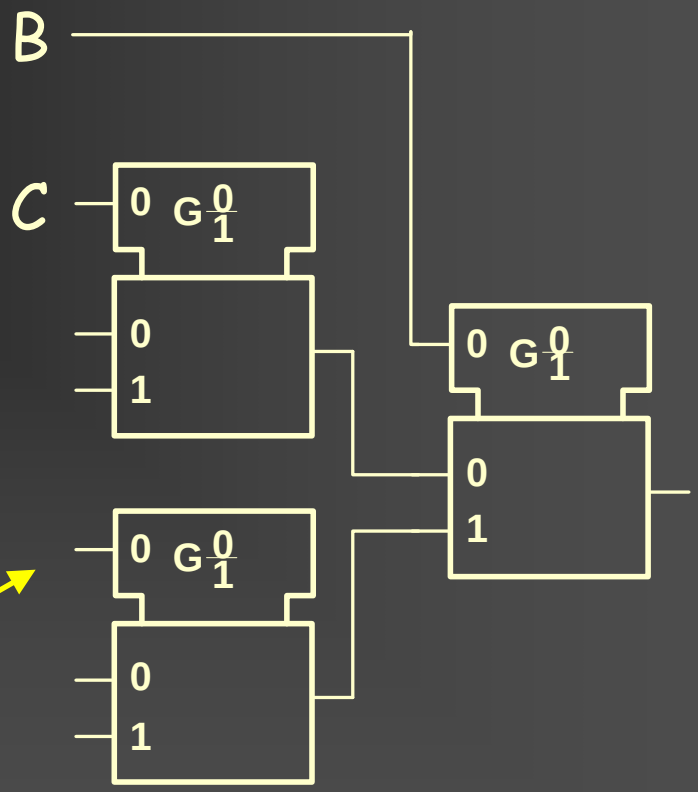
Mux-basierte Logik: Beispiel

$F = (A \wedge B) \vee (\neg B \wedge C) \vee D$

$F = [\neg B \wedge (C \vee D)] \vee [B \wedge (A \vee D)]$

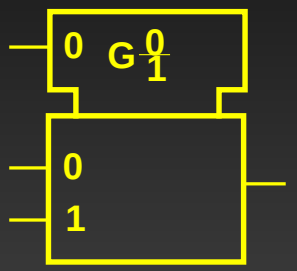
$F_{\overline{B}} = [\neg C \wedge D] \vee [C \wedge 1]$

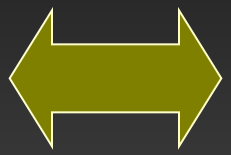
$F_B = [\neg A \wedge D)] \vee [A \wedge 1]$

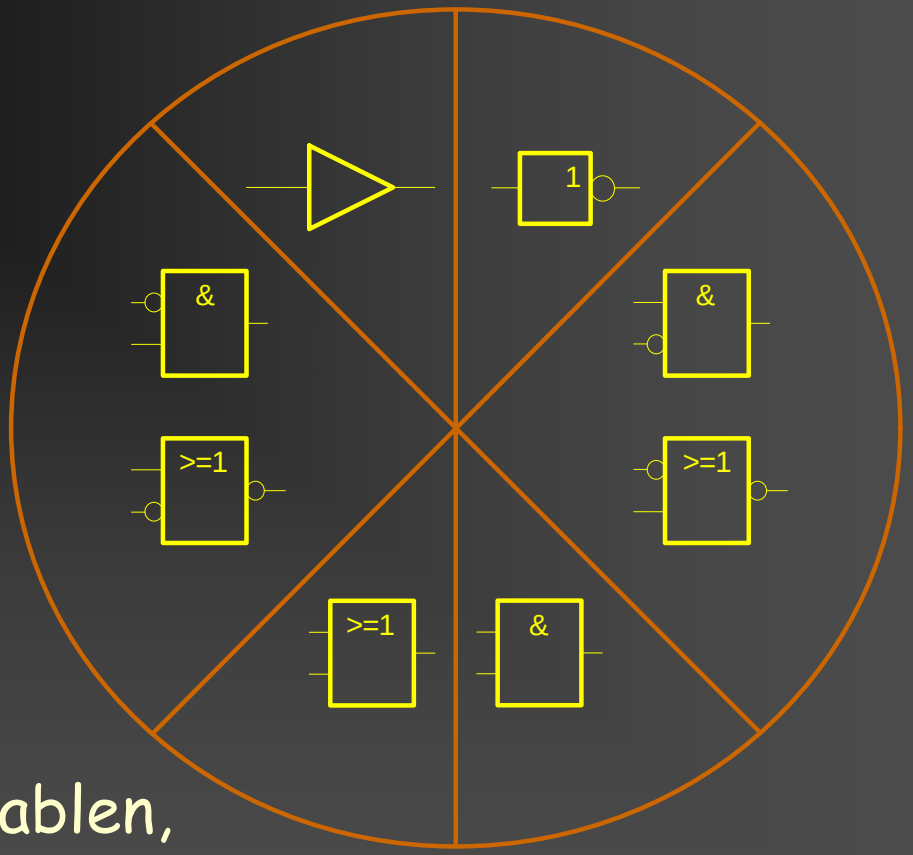


© A. Steininger / TU Wien

MUX: Realisierbare Funktionen



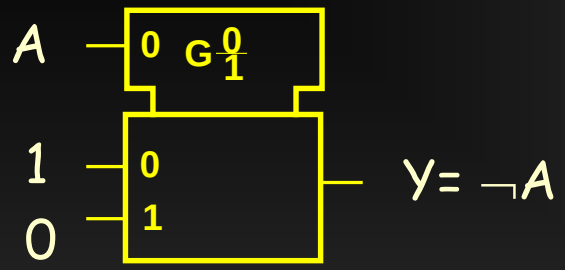




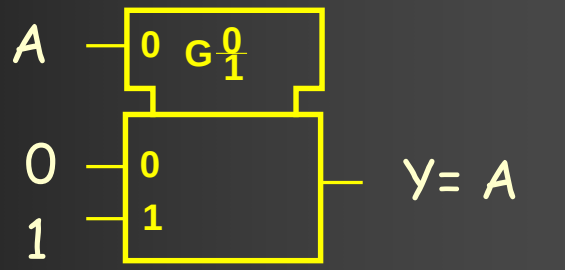
Realisierbar sind *alle* Funktionen mit 2 Variablen, manche in 3 Variablen

© A. Steininger / TU Wien

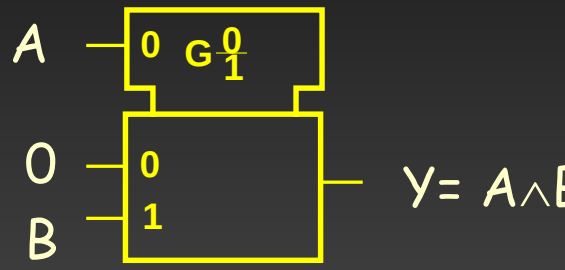
MUX-Realisierung: Beispiele



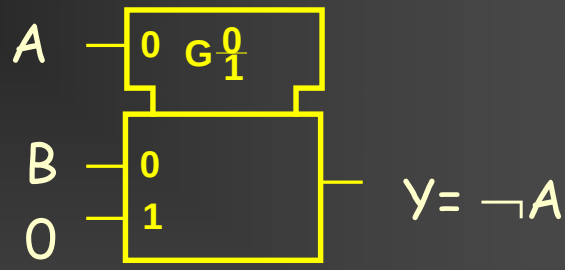
Y= ¬A



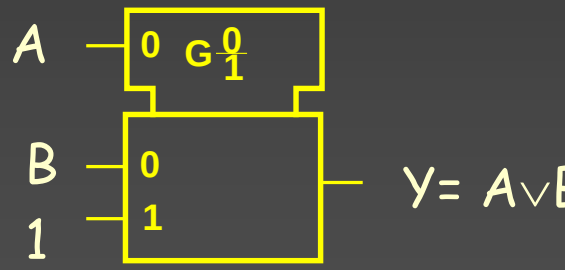
Y= A



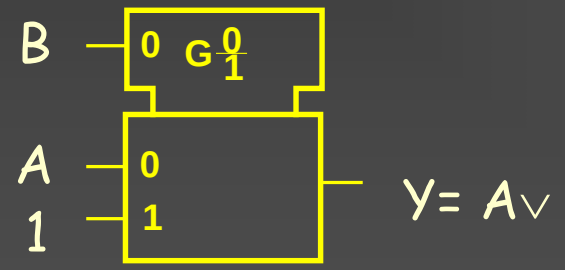
Y= A∧B



Y= ¬A∧B



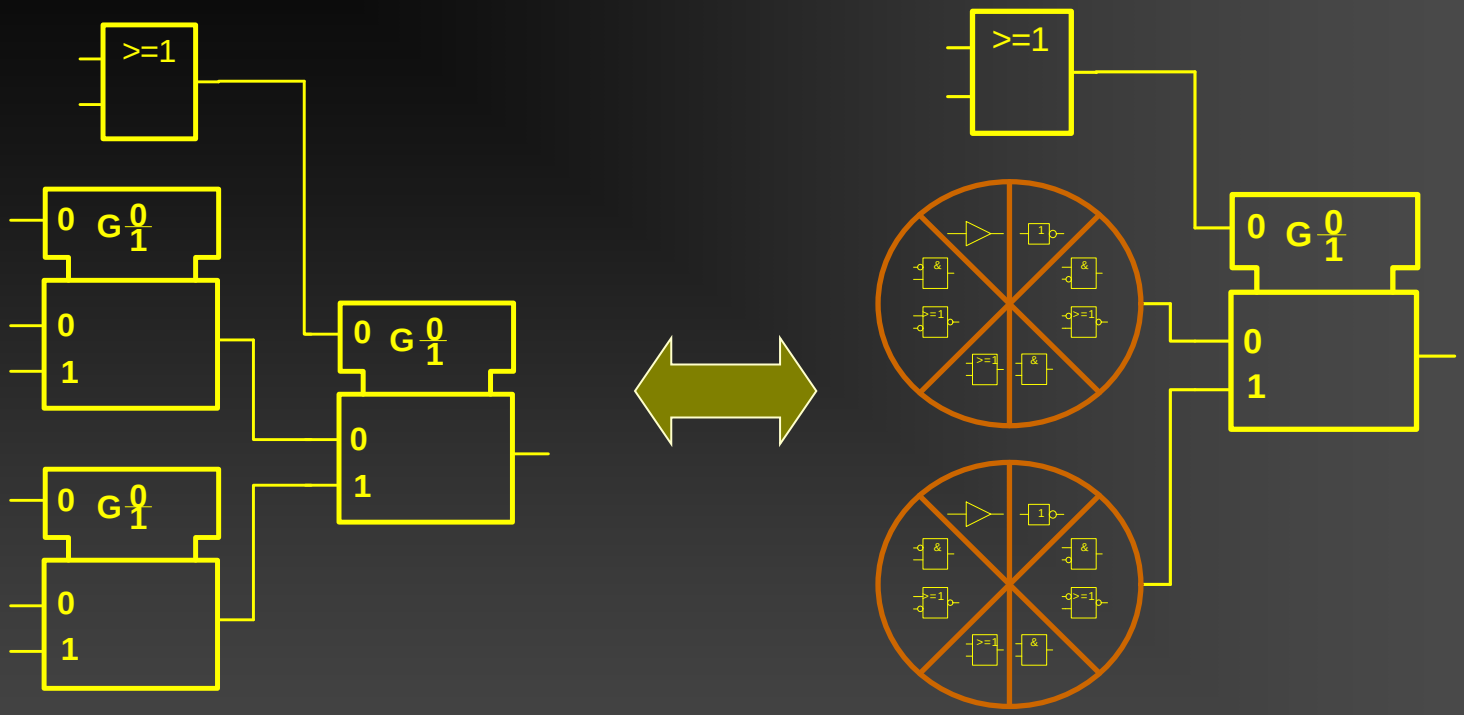
Y= A∨B



Y= A∨ ¬B

© A. Steininger / TU Wien

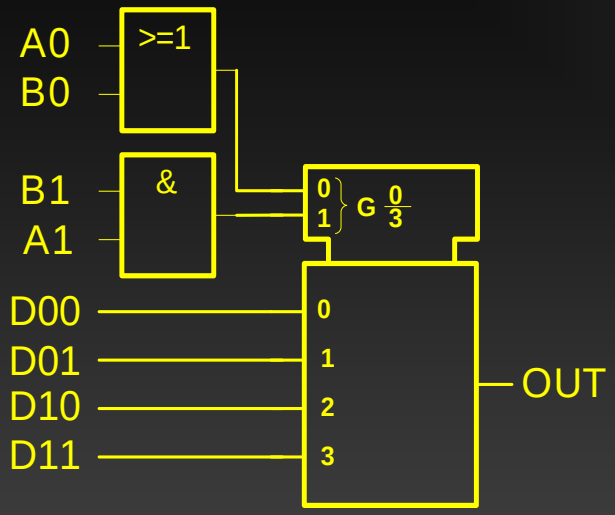
ACT1: Realisierbare Funktionen



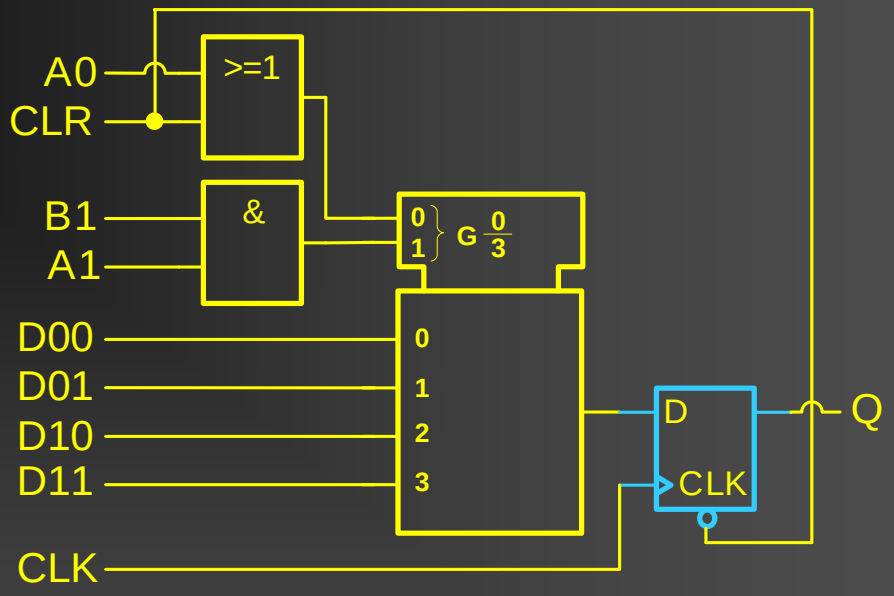
alle Funktionen mit 2 Variablen, Latch (=1/2 FF),
fast alle mit 3 Variablen und viele mit 4 Variablen

© A. Steininger / TU Wien

ACT2: Verbesserungen



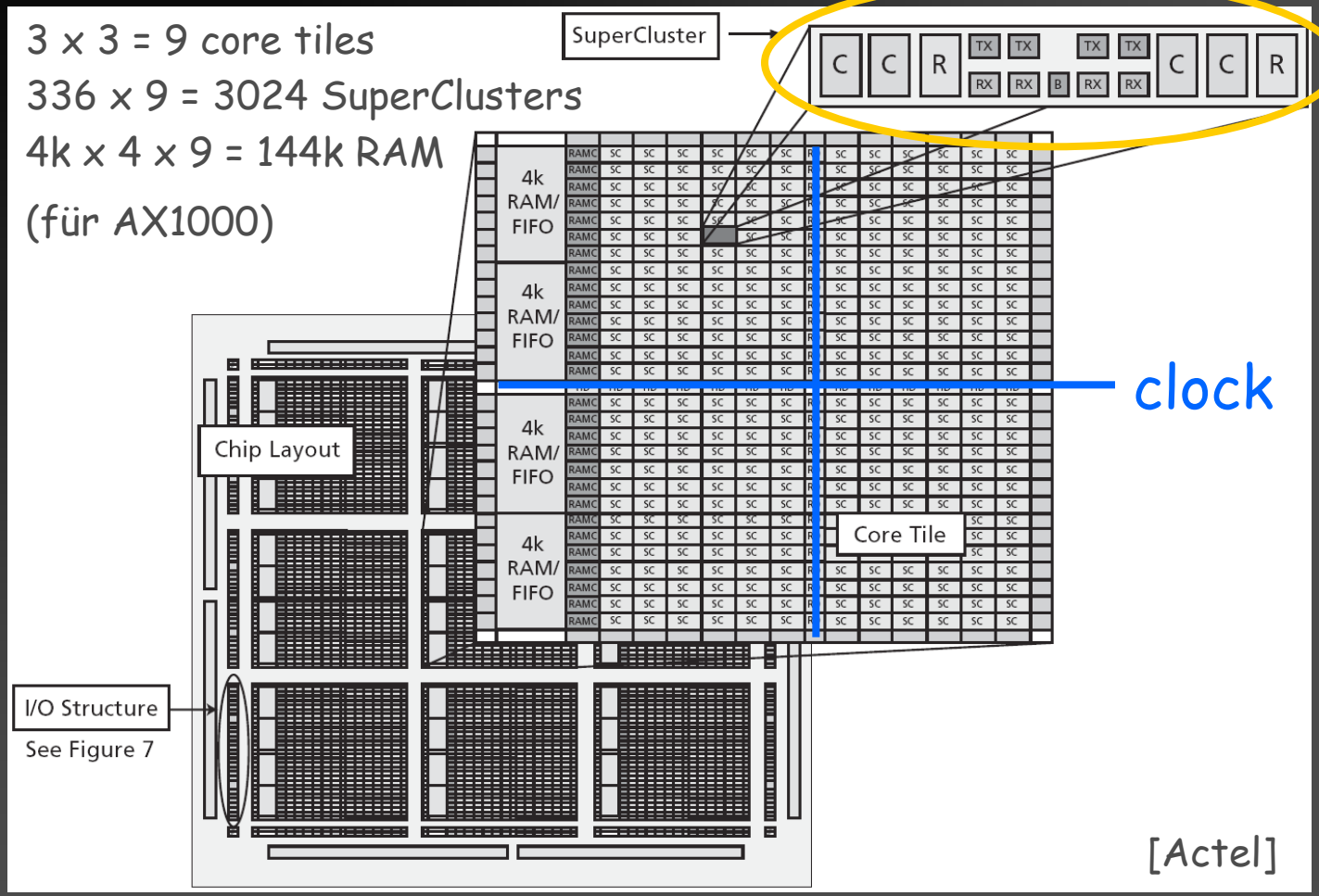
C-Modul mit
mehr Eingängen



S-Modul mit zusätzlichem
Sequential Element „SE" = D-FF

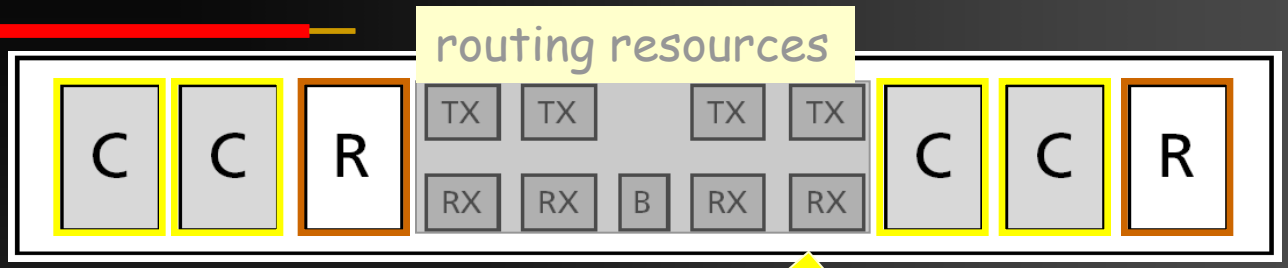
© A. Steininger / TU Wien

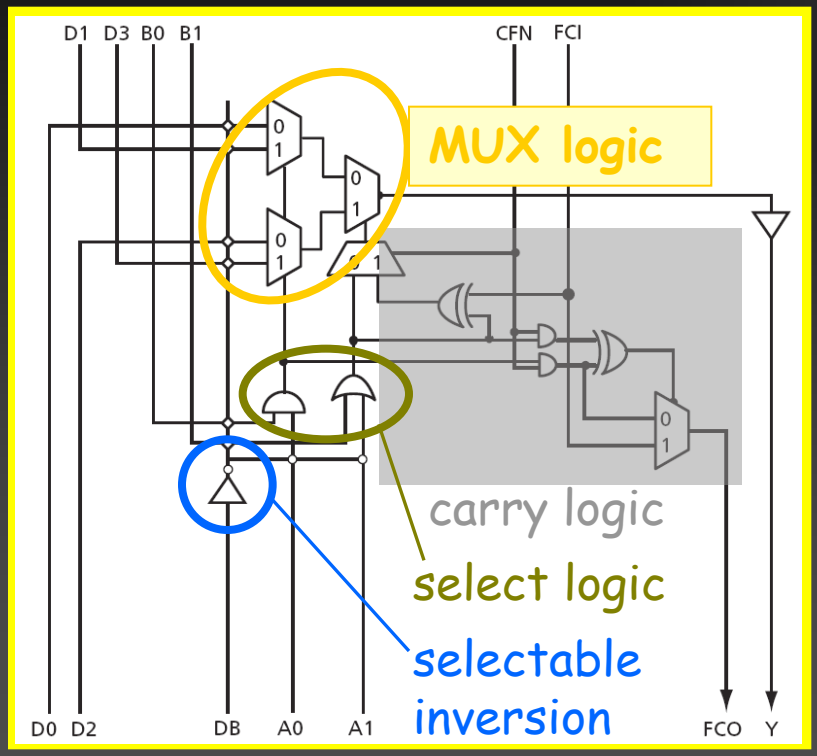
Actel Axcelerator-Familie

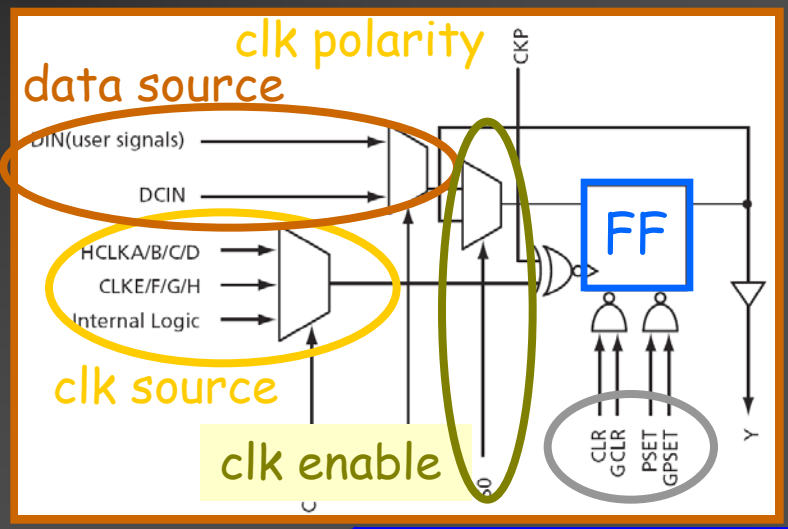


© A. Steininger / TU Wien

Actel's „Supercluster"







© A. Steininger / TU Wien

Programmierbare Logikzellen

Angewandte Prinzipien:

- ✓ Mux-basierte Logikzellen
  - Beispiel: Actel Axcelerator
- ▲ Look-up Table-basierte Logikzellen
  - Beispiele: Xilinx Virtex, Altera Stratix
- ▲ PAL-basierte Logikzellen
  - Beispiel: Altera MAX, Xilinx XC9500

© A. Steininger / TU Wien

Look-up Table (LUT)

- Kombinatorische Verknüpfung von *n* Variablen ist mittels Wahrheitstabelle eindeutig darstellbar
- Es gibt 2^n Eingangskombinationen
- Realisierung der Wahrheitstabelle als Speicher mit 2^n x 1Bit
- Eingangsvariable werden als Adressen angelegt, Speicherinhalt bestimmt Verknüpfungsfunktion
- LUT auch als RAM verwendbar (teuer!)
- konstantes Timing für alle Funktionen
- Beispiele: Altera Cyclone & Stratix, Xilinx Virtex

© A. Steininger / TU Wien

Altera Stratix-Familie

I/O-Elements



[Altera]

Embedded DSP Blocks

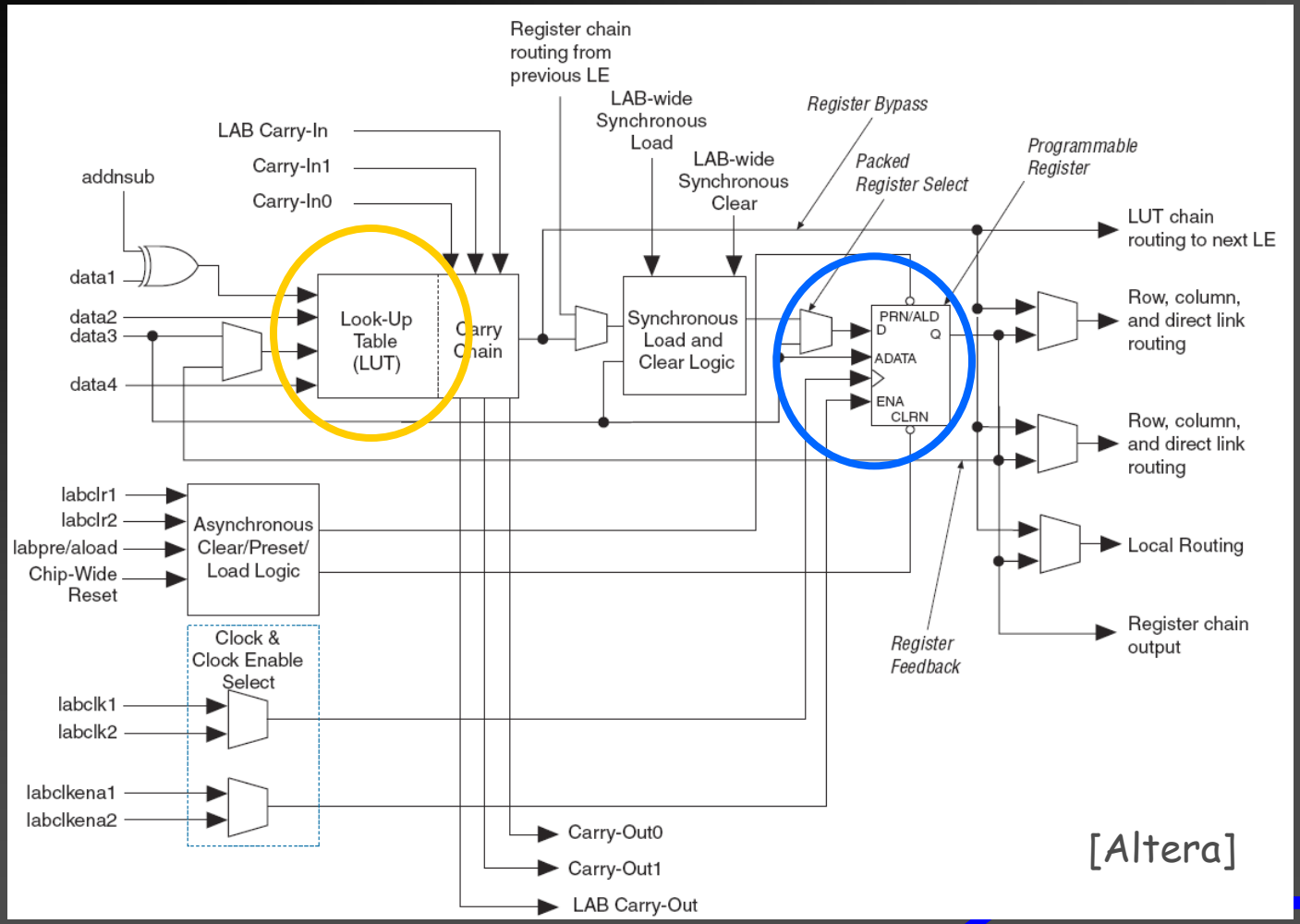
Embedded RAM blocks

Logic Array Block (LAB)
1 LAB = 10 LEs

??

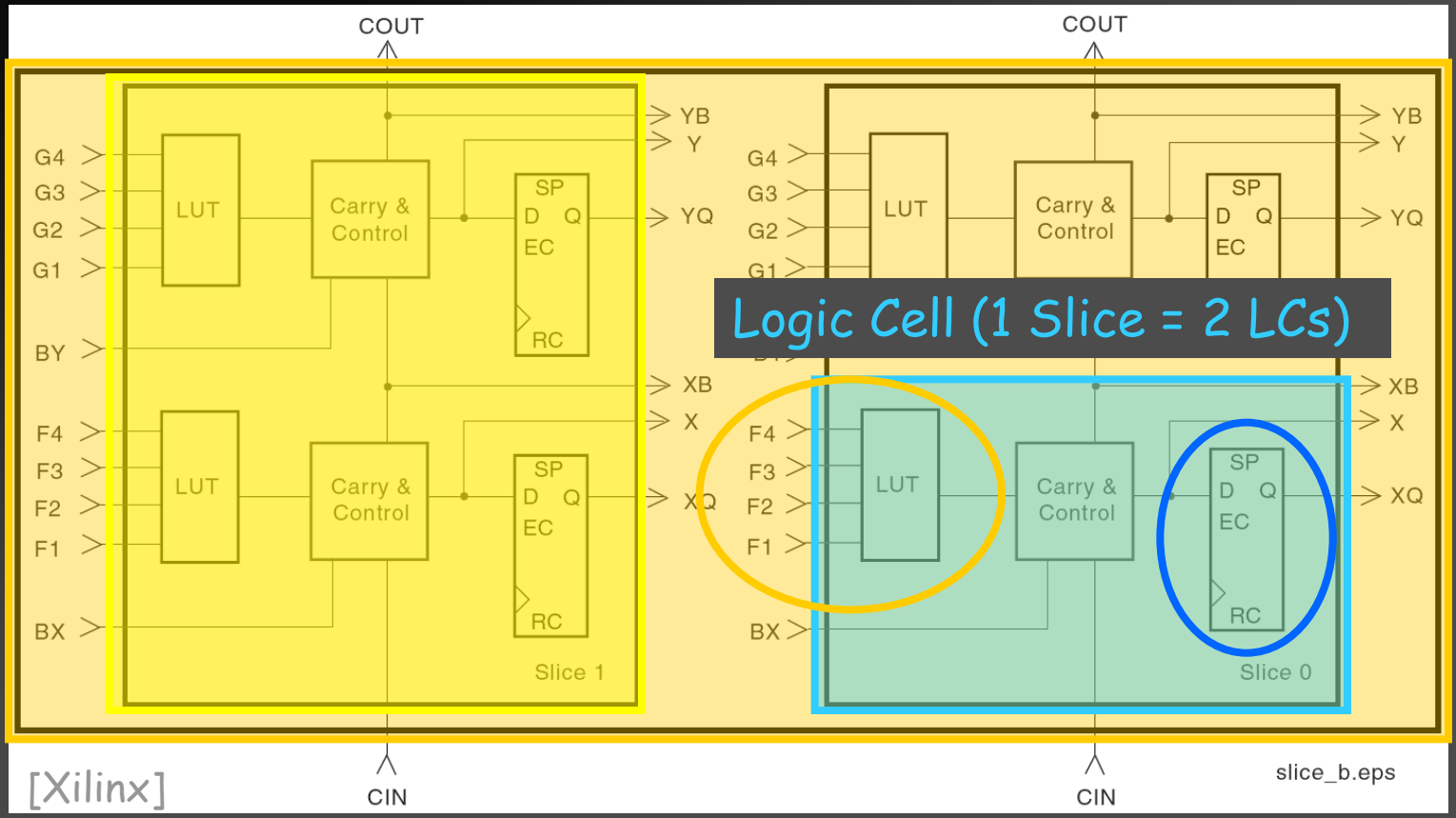
© A. Steininger / TU Wien

Altera's „Logic Element"



© A. Steininger / TU Wien

Xilinx Virtex „CLB"



Slice (1 CLB = 2 Slices)

Configurable Logic Block (CLB)

© A. Steininger / TU Wien

LUT-basierte FPGAs - Beispiele

- Altera Stratix EP1S25
  - 2566 LABs = 25660 LEs
  - 2Mbit RAM
  - 706 I/O Pins
  - 80 embedded multipliers (9x9),
- Xilinx Virtex XCV300
  - 32x48 CLBs = 6912 LCs
  - 64kbit RAM
  - 316 I/O Pins

© A. Steininger / TU Wien

Programmierbare Logikzellen

Angewandte Prinzipien:

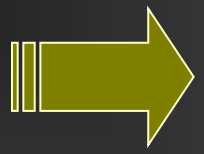
- ✓ Mux-basierte Logikzellen
  - Beispiel: Actel Axcelerator
- ✓ Look-up Table-basierte Logikzellen
  - Beispiele: Xilinx Virtex, Altera Stratix
- ▲ PAL-basierte Logikzellen
  - Beispiel: Altera MAX, Xilinx XC9500

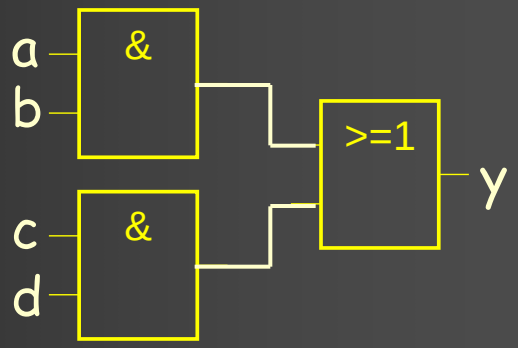
© A. Steininger / TU Wien

Grundprinzip des PAL

- direkte Implementierung der disjunktiven Normalform

$$y = (a \wedge b) \vee (c \wedge d)$$



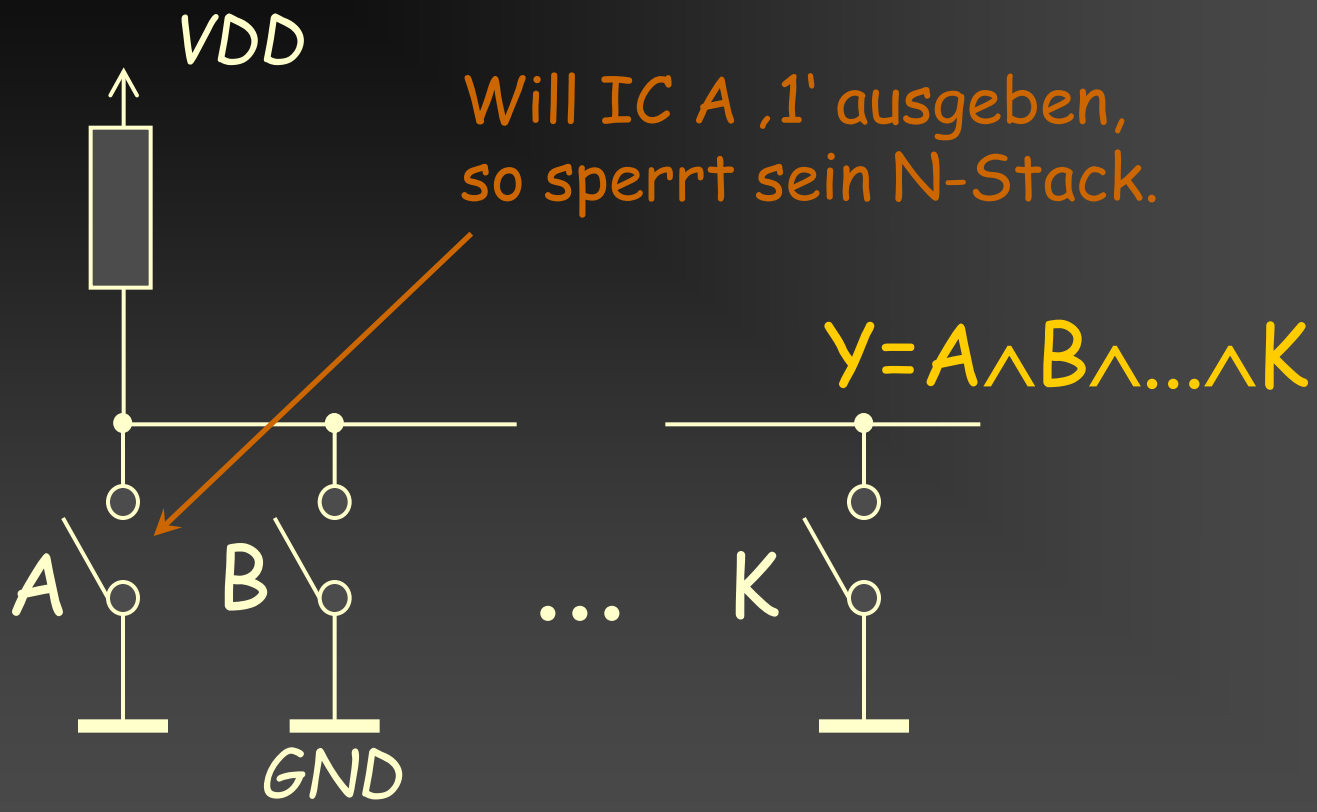


- mehrere AND-Gatter bilden die Minterme
- ein OR-Gatter fasst diese zusammen
- dahinter FF um sequenzielle Logik realisieren zu können

© A. Steininger / TU Wien

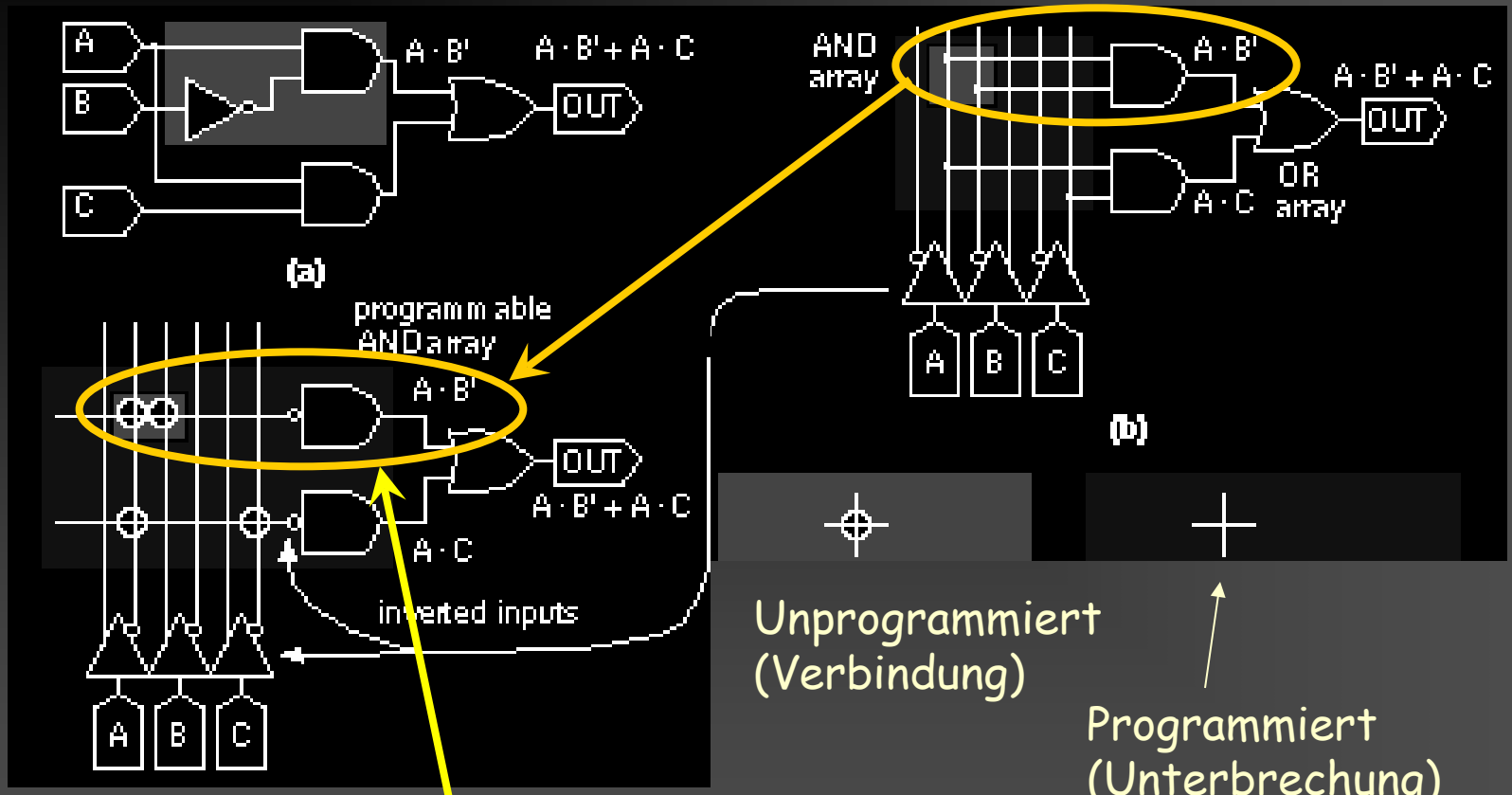
Wh...

Prinzip des „Wired AND"



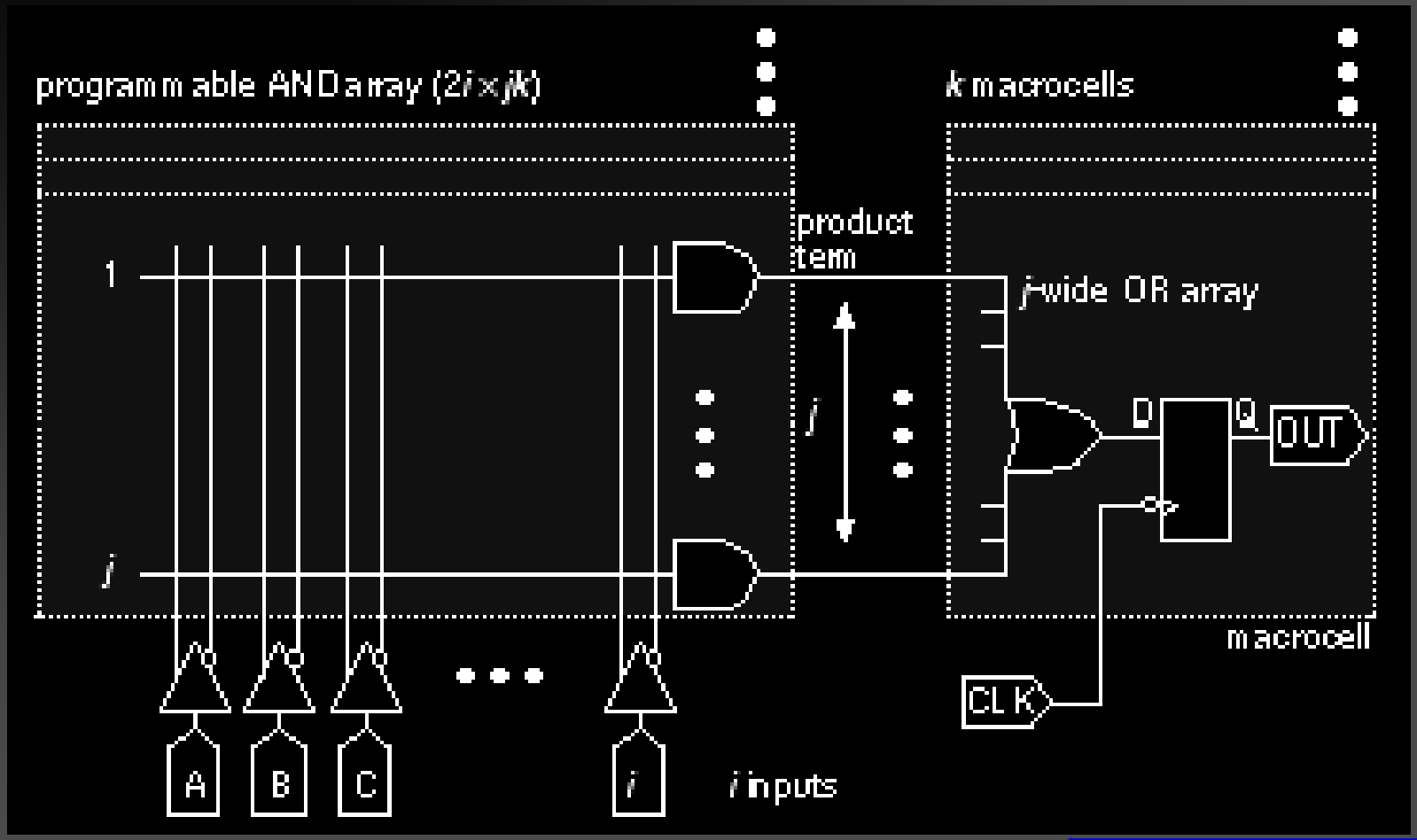
© A. Steininger / TU Wien

PLD: Notation



© A. Steininger / TU Wien

Standard PLD Struktur



© A. Steininger / TU Wien

Erweiterte Fähigkeiten

Ergeben sich in kanonischer Form mehr Produktterme als zur Verfügung stehen (Eing. des OR):

- Kaskadierung von Stufen, d.h. Teilberechnung in einer Stufe => als Input für die nächste Stufe (Parallel Expander). Doppelte Durchlaufzeit!
- Durch Umformung einzelne Produktterme herauslösen, die mittels Logic Expander vorverarbeitet werden (zusätzliche Durchlaufzeit!).
- Inverse Funktion realisieren und mit einem programmierbaren Inverter (XOR) am Ausgang wieder richtig stellen.

© A. Steininger / TU Wien

Logic Expander: Prinzip

$F = (\neg A \wedge C \wedge D) \vee (\neg B \wedge C \wedge D) \vee (A \wedge B) \vee (B \wedge \neg C)$

hat 4 Produktterme => benötigt OR4

$F = [(\neg A \vee \neg B) \wedge C \wedge D] \vee [B \wedge (A \vee \neg C)]$

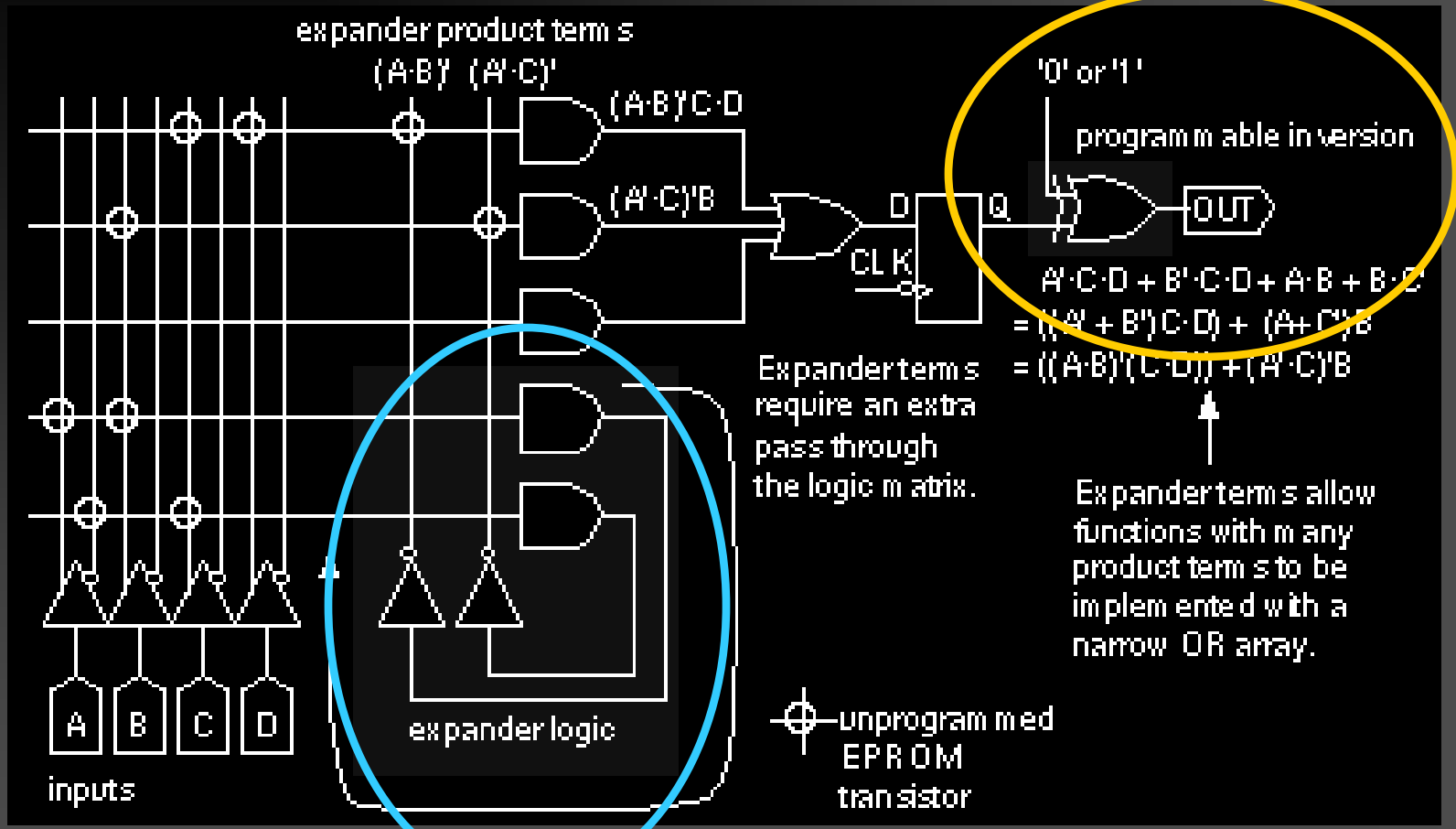
$= [\neg(A \wedge B) \wedge C \wedge D] \vee [B \wedge \neg(\neg A \wedge C)]$

ist äquivalent und benötigt nur OR2, sofern Ausdrücke in runder Klammer „vorverarbeitet" sind.

Vorverarbeitung ist mit einem *einzelnen* Produktterm möglich

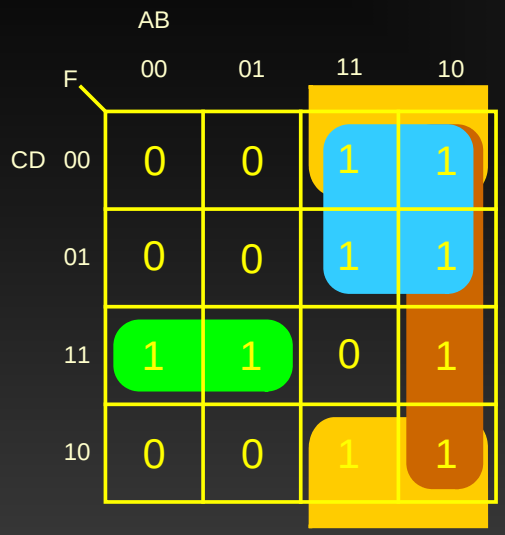
© A. Steininger / TU Wien

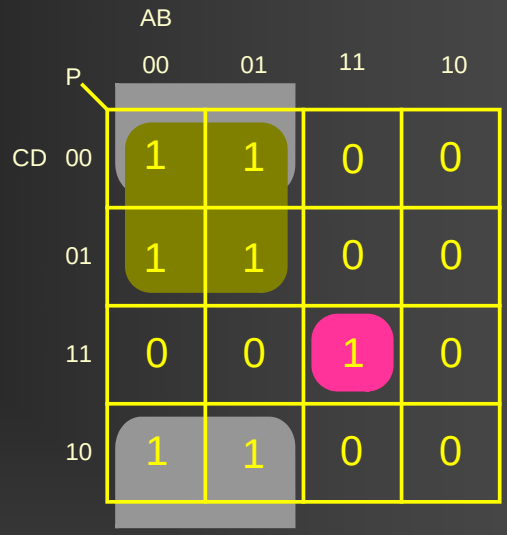
Realisierung im Standard-PLD



© A. Steininger / TU Wien

Programmierbare Inversion





$F = (A \wedge \neg B) \vee (A \wedge \neg C) \vee (A \wedge \neg D) \vee (\neg A \wedge C \wedge D)$

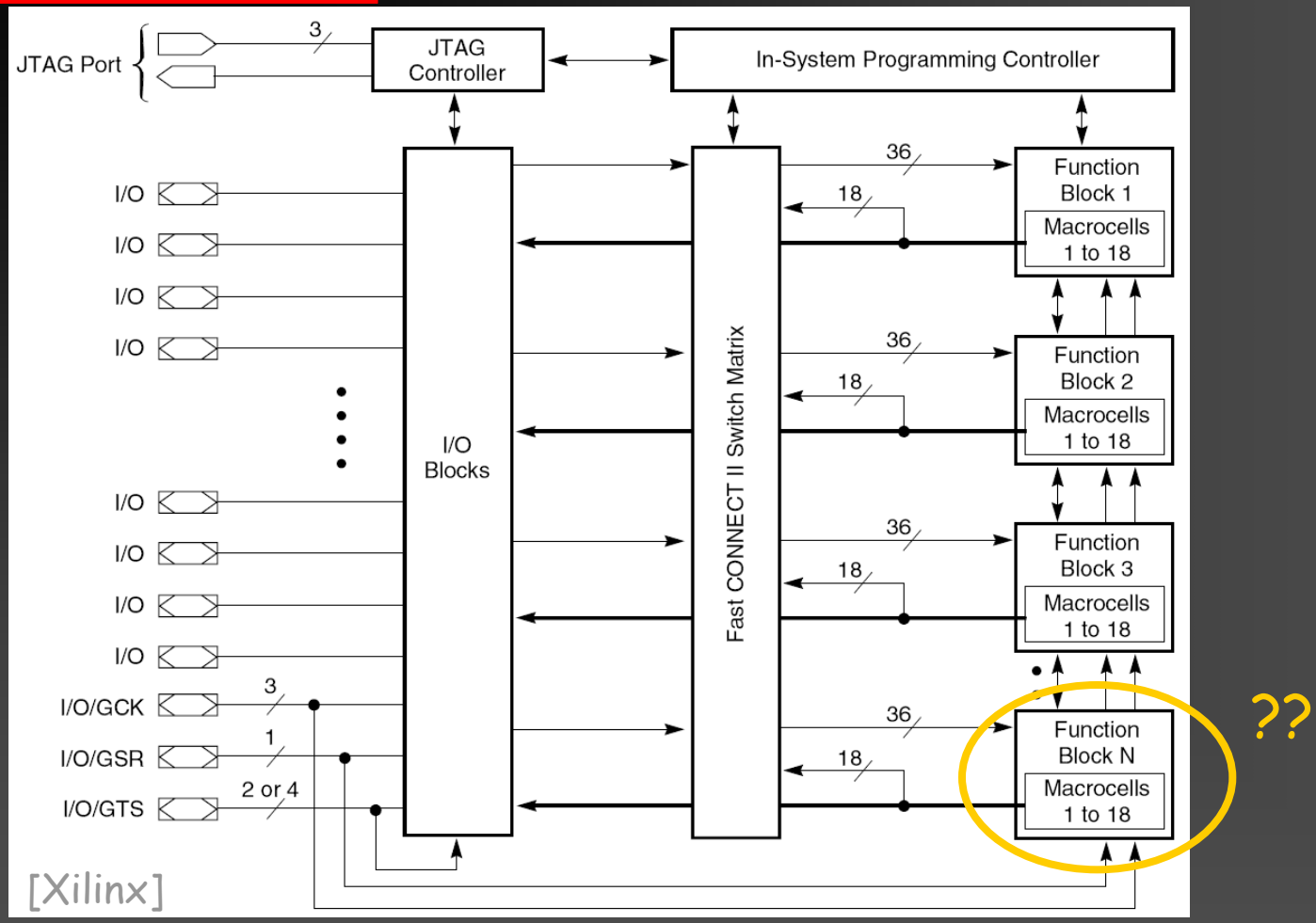
$\neg F = (A \wedge B \wedge C \wedge D) \vee (\neg A \wedge \neg D) \vee (\neg A \wedge \neg C)$

4 Produktterme

3 Produktterme

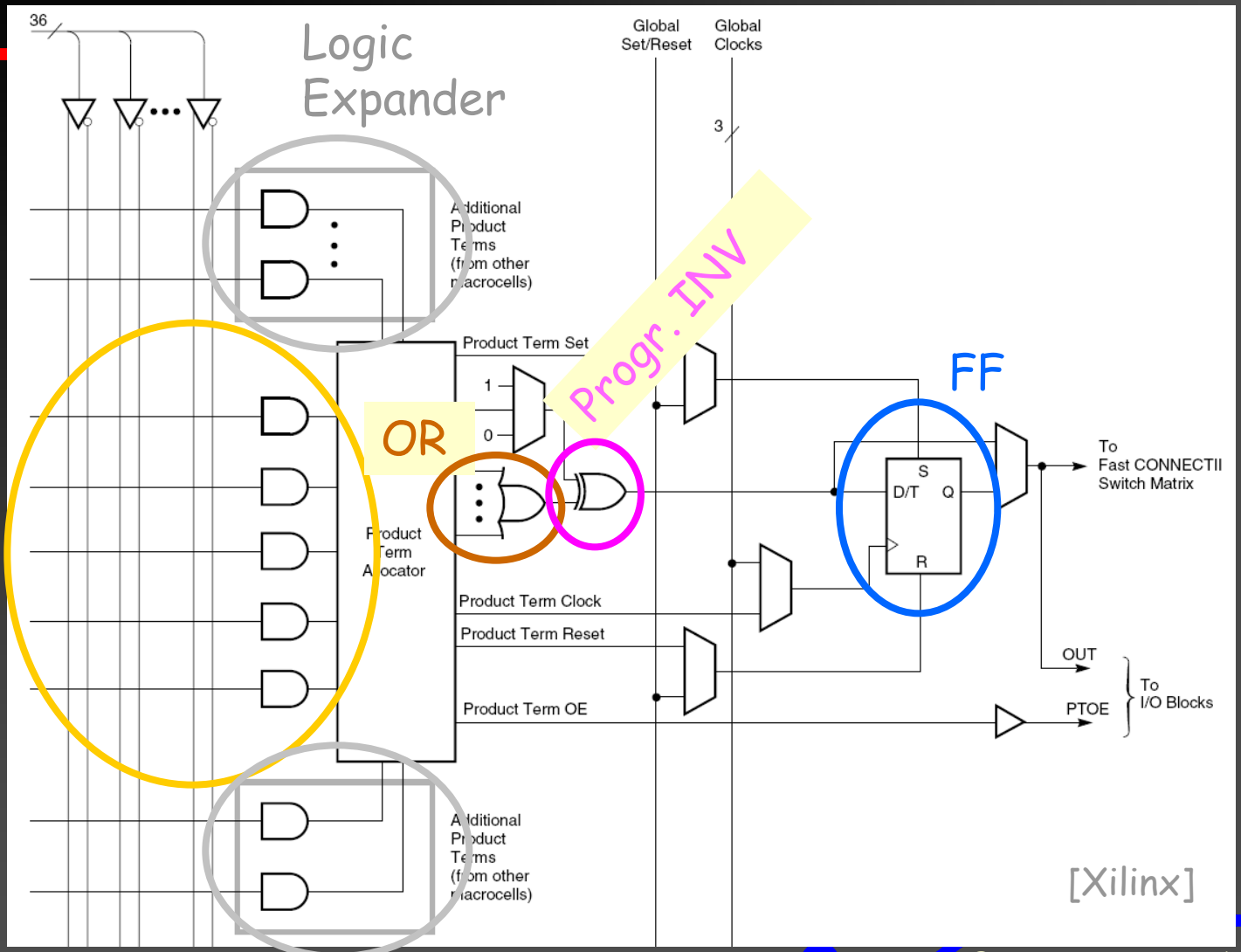
© A. Steininger / TU Wien

Xilinx XC9500 Familie



© A. Steininger / TU Wien

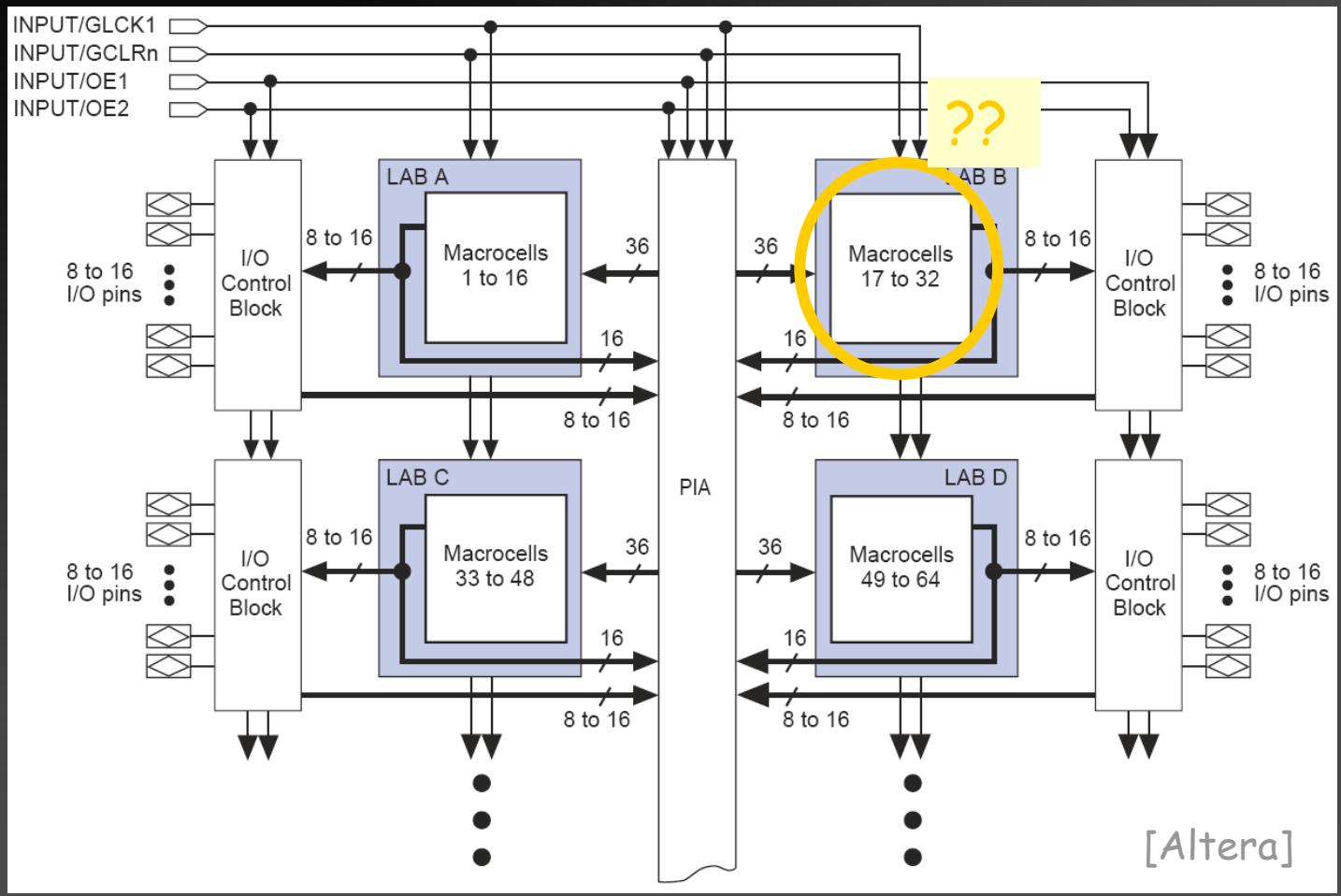
Xilinx XC9500 Macrocell



AND-Matrix

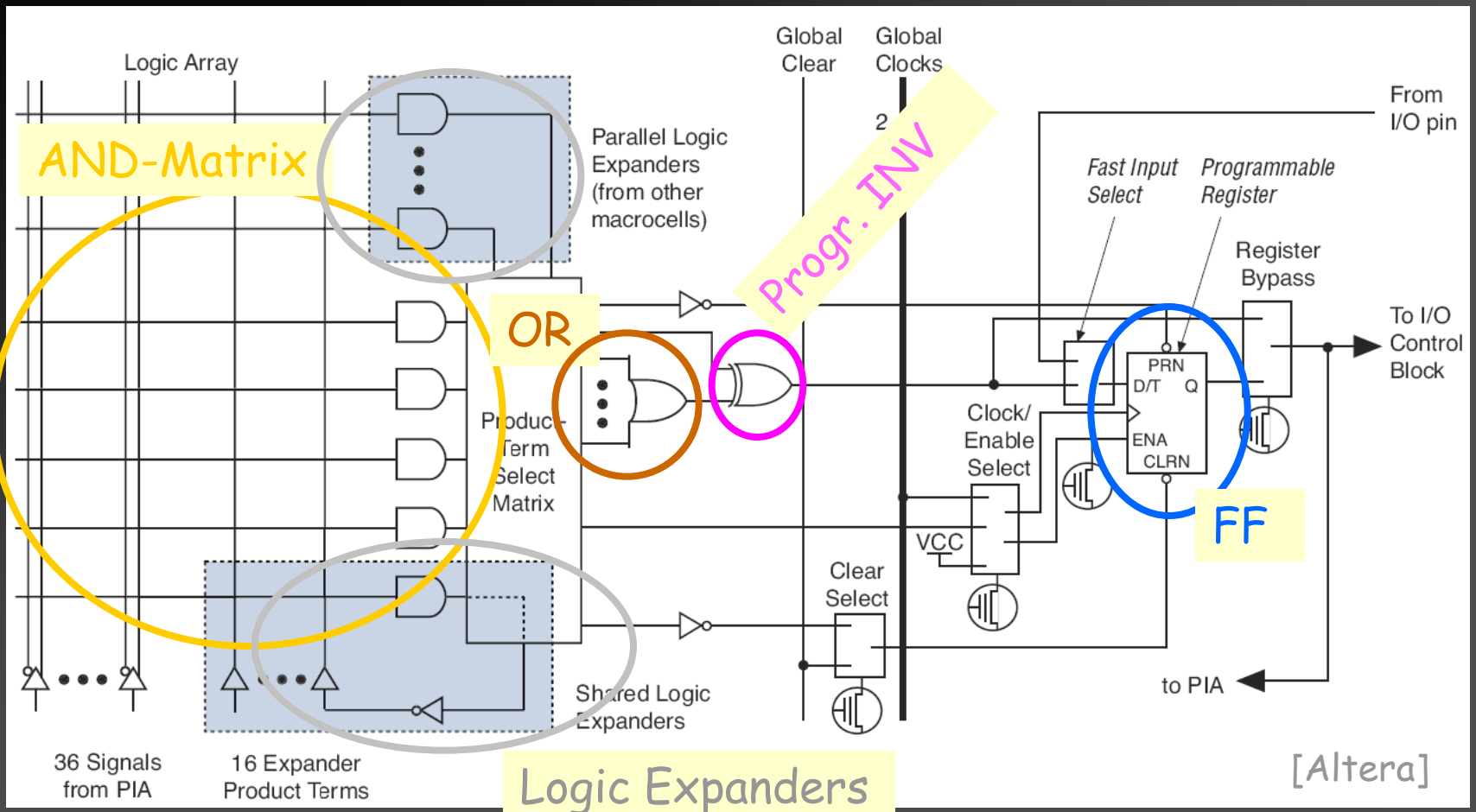
© A. Steininger / TU Wien

Altera MAX7000 Familie



© A. Steininger / TU Wien

Altera MAX7000 Macrocell

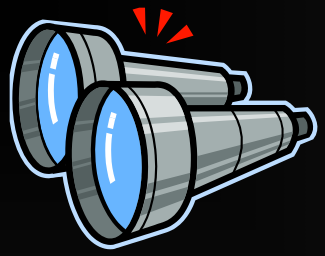


© A. Steininger / TU Wien

PAL-basierte CPLDs - Beispiele

- Xilinx XC95144
  - 8 Function Blocks = 144 Macrocells
  - 81/133 I/O Pins (je nach Gehäuse)
- Altera MAX7128
  - 8 LABs = 128Macrocells
  - 100 I/O Pins

© A. Steininger / TU Wien



Überblick

- Terminologie
- ASIC-Typen und ihre Charakteristika
- Programmable Logic Devices: Prinzipien
- Programmierbare Logikzellen
- **Programmierbare I/O-Zellen**
- Programmierbarer Interconnect

© A. Steininger / TU Wien

FPGA:Was ist programmierbar?

✓ Makrozellen	Position	fix (Array)
	innere Struktur	fix
	Funktion	prog.-bar
▲ I/O-Zellen	Position	fix (Rand)
	innere Struktur	fix
	Funktion	prog.-bar
Verbindungen	Möglichkeiten	fix
	Auswahl	prog.-bar

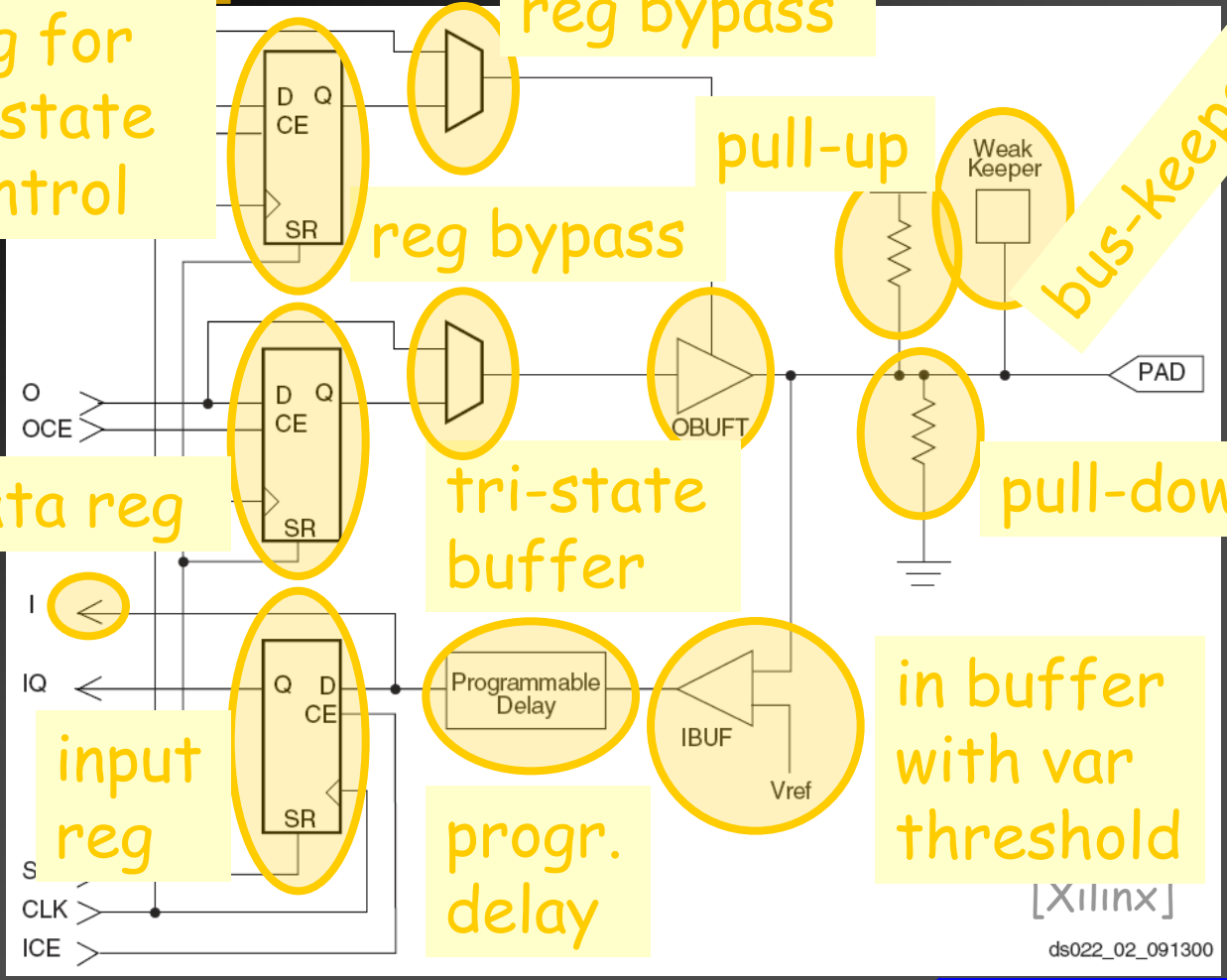
© A. Steininger / TU Wien

Typische I/O-Block Features

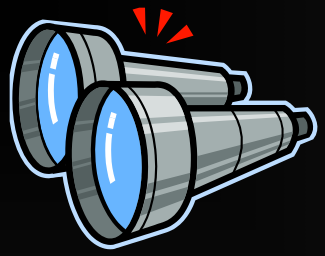
- Eingang
  - unterschiedliche Spannungen/Standards
  - direkt oder über eigenes Input-FF / Latch
  - Verzögerung zuschaltbar
- Ausgang
  - unterschiedliche Spannungen/Standards
  - passiver Pull-up und Pull-down (50kΩ)
  - Tri-State und Open Drain
  - Ausgang direkt oder über FF / Latch
  - wählbare Flankensteilheit und Treiberstärke

© A. Steininger / TU Wien

Xilinx Virtex I/O Block



© A. Steininger / TU Wien



Überblick

- Terminologie
- ASIC-Typen und ihre Charakteristika
- Programmable Logic Devices: Prinzipien
- Programmierbare Logikzellen
- Programmierbare I/O-Zellen
- **Programmierbarer Interconnect**

© A. Steininger / TU Wien

FPGA:Was ist programmierbar?

✓ Makrozellen	Position	fix (Array)
	innere Struktur	fix
	Funktion	prog.-bar
✓ I/O-Zellen	Position	fix (Rand)
	innere Struktur	fix
	Funktion	prog.-bar
▲ Verbindungen	Möglichkeiten	fix
	Auswahl	prog.-bar

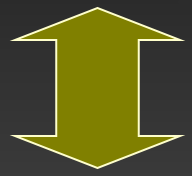
© A. Steininger / TU Wien

Interconnect-Optimierung

maximale Flexibilität

beliebige Verbindungen zwischen Blöcken

viele Verbindungen und Anschlüsse



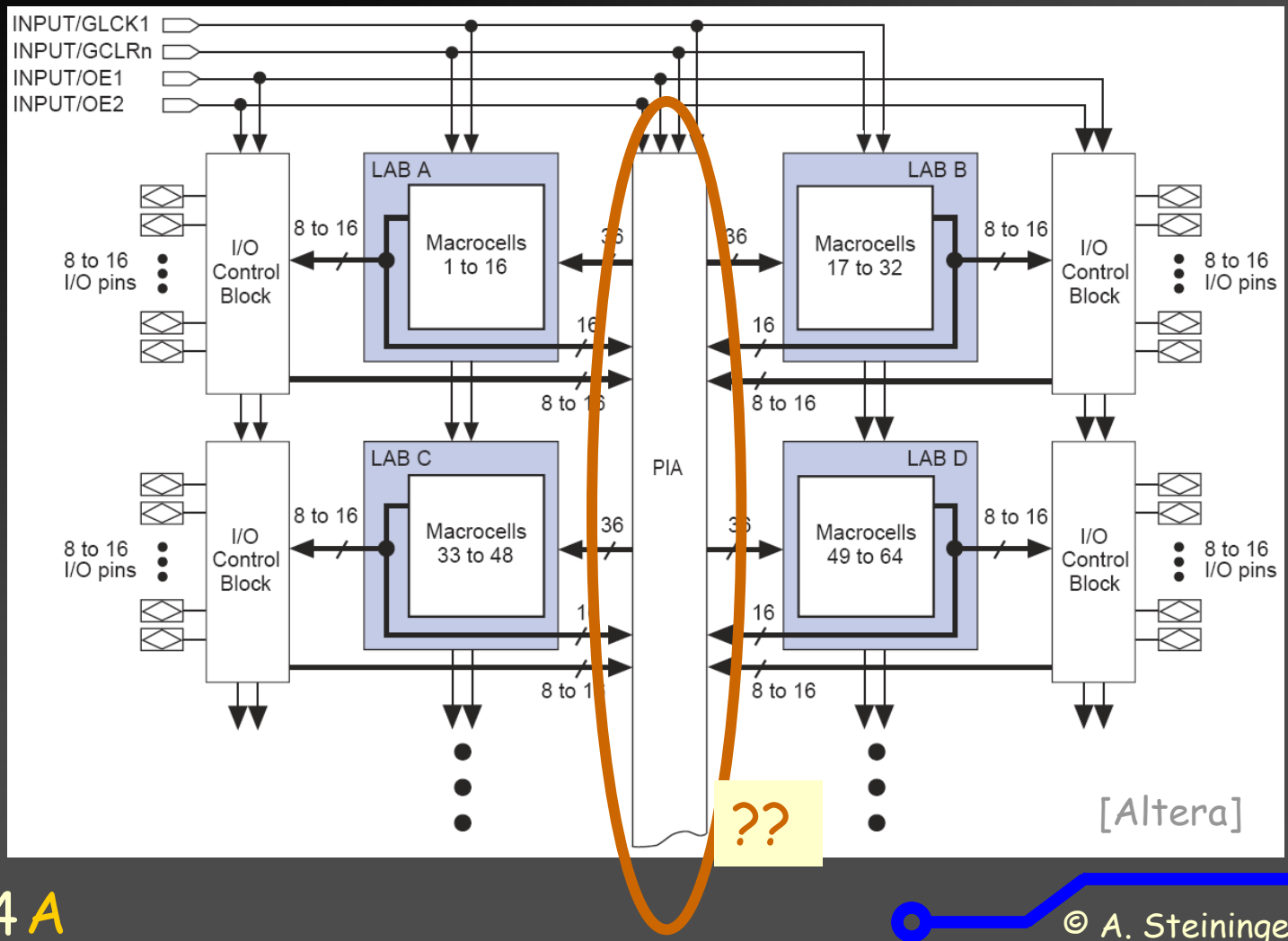
maximale Geschwindigkeit

minimale Kapazitäten der Verbindungen & Anschlüsse

wenige Verbindungen und Anschlüsse

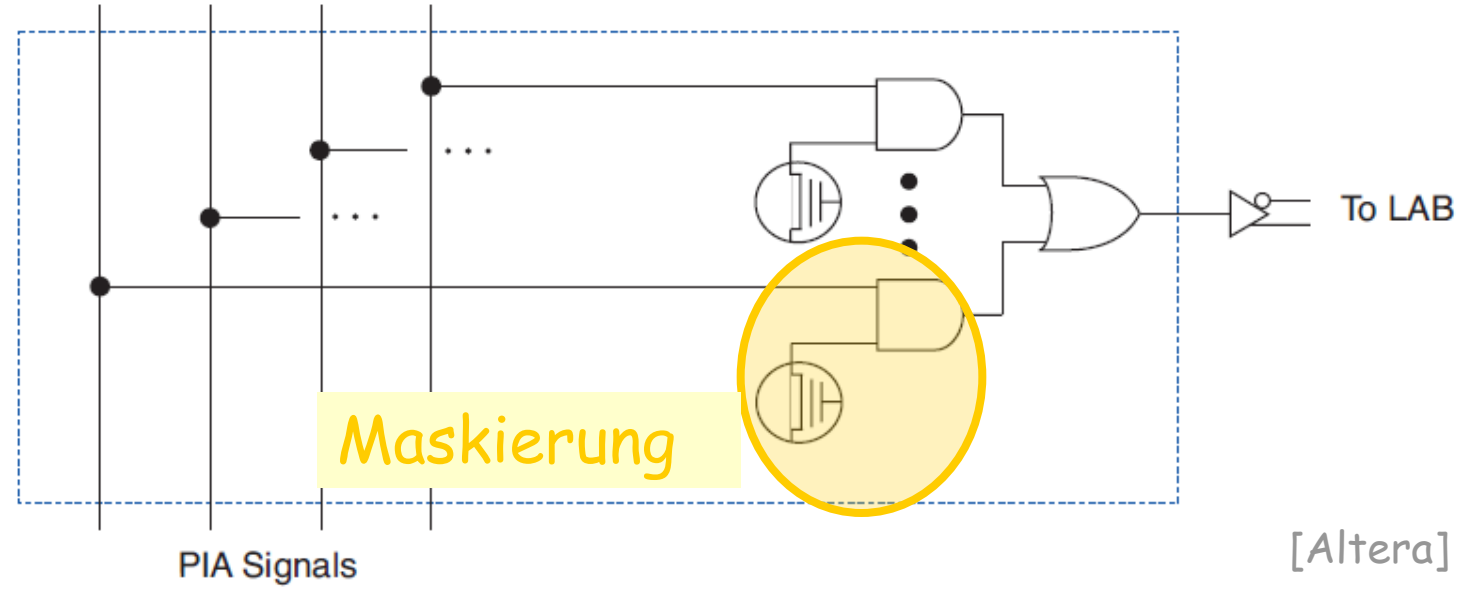
© A. Steininger / TU Wien

MAX7000 CPLD Interconnect



© A. Steininger / TU Wien

MAX Progr. Interconn. Array

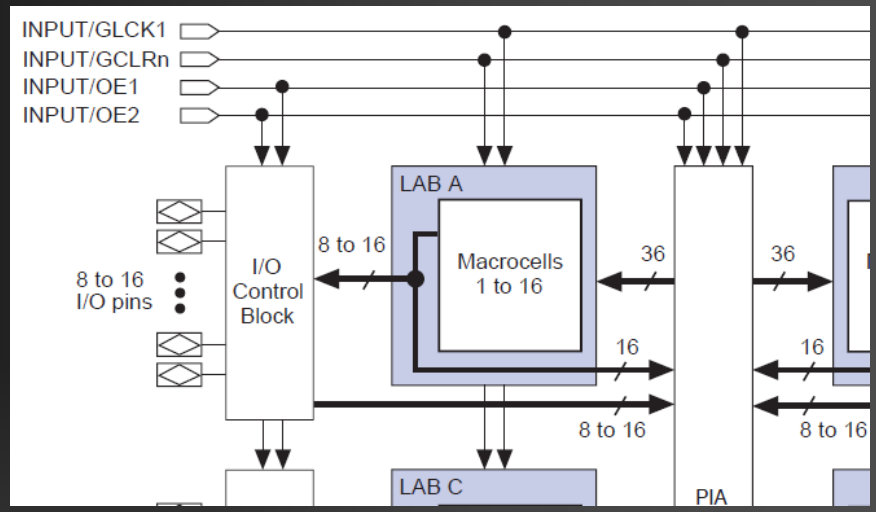


mittels Maskierung (Konfig.) wird genau jenes Signal gewählt, das zum LAB soll

© A. Steininger / TU Wien

Signale im PIA

- Senken
  - alle LAB Inputs
- Quellen
  - alle I/O Pins
  - alle „dedicated Inputs“
  - alle Macrocell Outputs



[Altera]

PIA ist ein völlig symmetrisches globaler Bus von jeder Quelle zu jeder Senke

© A. Steininger / TU Wien

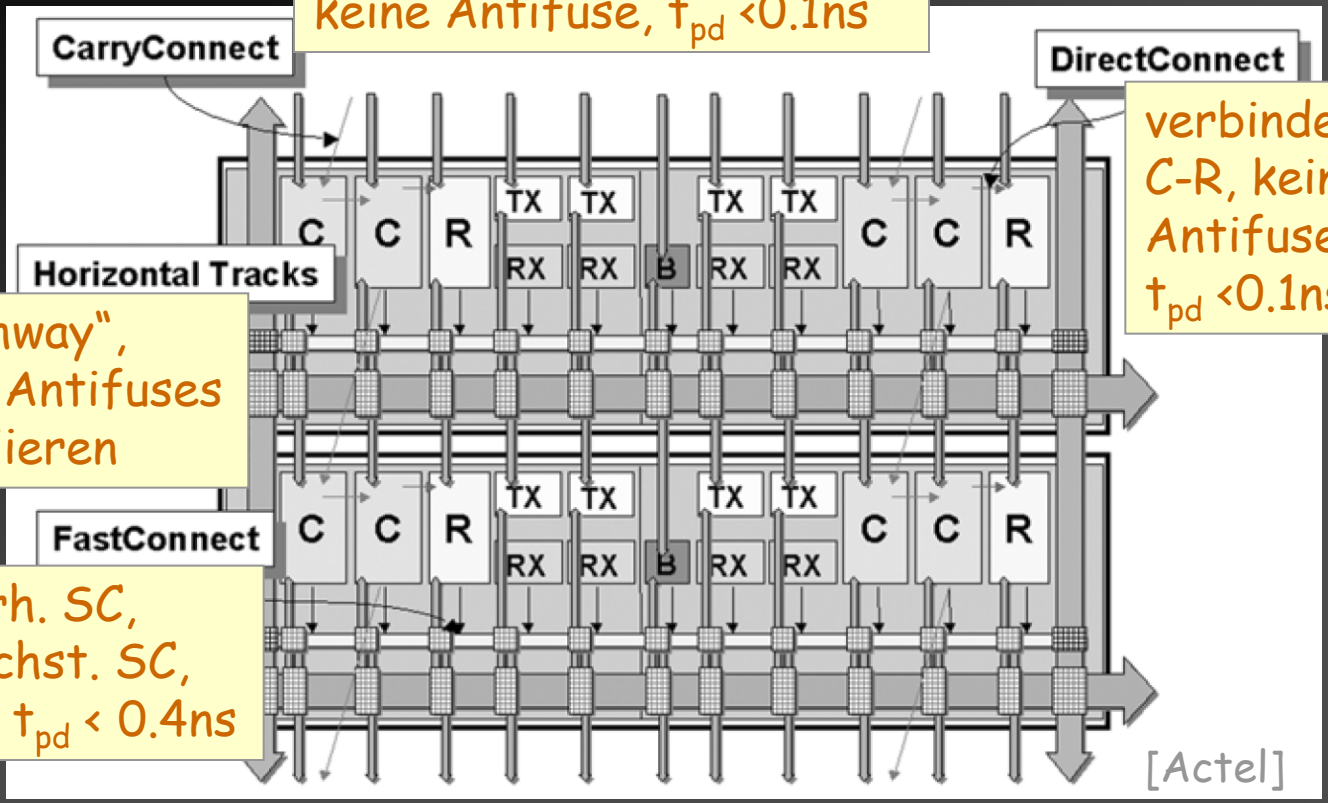
Bewertung CPLD-Interconnect

- hoch regulär und symetrisch
  - möglich wegen einfacher Struktur der CPLDs
  - voll verbunden
  - PPR einfach
  - Timing gut vorhersagbar
- Interconnect bestimmt auch Funktion
  - Funktion und Struktur der Logik fix (PLD)
  - ABER: welcher Product Term verwendet welche Eingänge

© A. Steininger / TU Wien

Actel Axcelerator Routing

verbindet carry logic, keine Antifuse, t_{pd} <0.1ns



verbindet C-R, keine Antifuse, t_{pd} <0.1ns

global „highway", Segmente, Antifuses und t_{pd} variieren

horiz. innerh. SC, vert. zu nächst. SC, 1 Antifuse, t_{pd} < 0.4ns

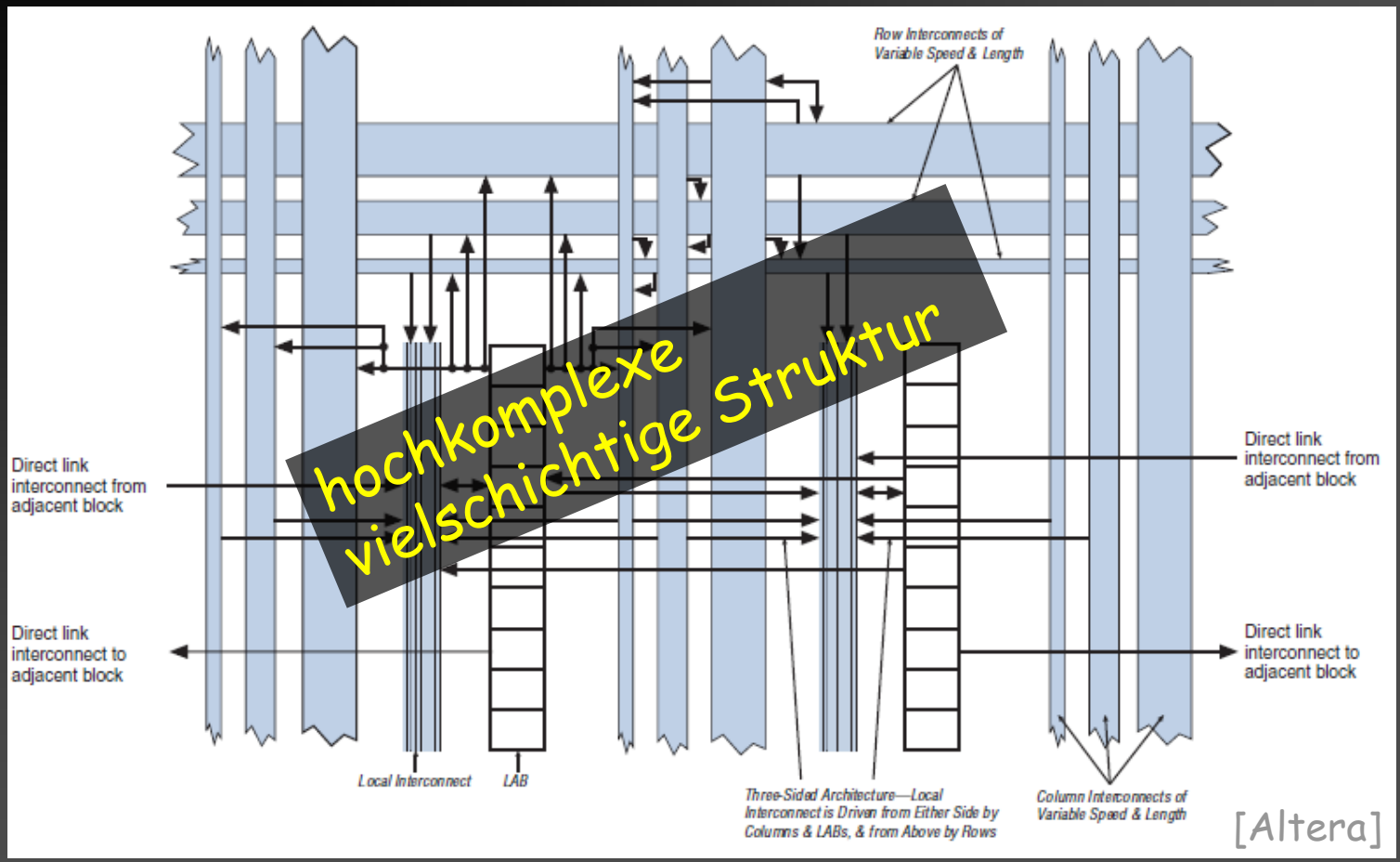
© A. Steininger / TU Wien

Bewertung Actel-Interconnect

- Antifuse erlaubt effiziente Verbindung auf kleinstem Raum
- lokal sehr regulär, „highway" variabel
  - Timing schwieriger vorhersagbar
  - highway ist potentieller Flaschenhals
  - PPR schwierig, muss nicht aufgehen
- Interconnect bestimmt auch Funktion
  - Struktur und Funktion der Logik fix (MUX)
  - ABER: welche Signale an welche MUX-Eingänge

© A. Steininger / TU Wien

Stratix FPGA Interconnect



© A. Steininger / TU Wien

Bewertung FPGA Interconnect

- hochkomplexes Netzwerk
  - mehrstufig segmentiert
  - Routing oft über mehrere Segmente in Serie
  - Timing kaum vorhersagbar
  - nicht voll vernetzt => potentieller Flaschenhals
- Routing und Funktion getrennt
  - Funktion durch LUT bestimmt
  - Routing wird getrennt konfiguriert

© A. Steininger / TU Wien

Konfiguration von CPLDs

- Speichermedium = on-chip Flash Memory
  - non-volatile, löschbar
  - Programmierdaten ca. 1Mbit
  - Security-Bit verhindet Auslesen (wenn aktiv)
- In-System Programmierung
  - über JTAG-Interface (Standard-Testschnittstelle)
  - Ansprechen vom PC (USB, Centronix)
  - Dauer: 5…10s
- Alternativ: in externem Programmer

© A. Steininger / TU Wien

Konfiguration von FPGAs

- Einspielen der Konfiguration durch
  - externen oder embedded (!) Microcontroller
  - Configuration Device (In-System progr.)
    (= OTP/Flash Mem + IF controller)
  - Download vom PC (seriell)

  ca. 8Mbit für Stratix EP1S25
- Interfaces
  - seriell, parallel (8Bit), JTAG
- Auswahl durch Beschaltung von Pins

© A. Steininger / TU Wien

Hardcopy Devices

- vereinfacht Migration von FPGA zu ASIC
- Angaben von Altera für Stratix Hardcopy:
  - gleiche Architektur, gleiches Pinout
  - 50% schneller
  - 40% weniger Energie
  - 8 Wochen Lieferzeit

© A. Steininger / TU Wien



Zusammenfassung (1)

- Der Full-Custom-ASIC bietet maximale Flexibilität für Optimierungen, wird jedoch aus Aufwandsgründen nur für Spezialanwendungen eingesetzt.
- Der Standardzellen IC (CBIC) geht von einer Library aus vorgefertigten und getesteten Logikzellen aus. Dies vereinfacht das Design, bei der Fertigung sind aber alle Layer kundenspezifisch.
- Beim Gate Array sind Basisfunktionen vorgegeben, durch kundenspezifische Metallisierungslayer kann über Makros jede Funktion implementiert werden. Man unterscheidet zwischen channelled, chanelless und structured GAs.

© A. Steininger / TU Wien



Zusammenfassung (2)

- Time to market ist ein entscheidendes Erfolgskriterium für ein Design. Das ist ein entscheidendes Argument für programmierbare Logik.
- Bei den programmierbaren Logikbausteinen (Programmable Logic Devices, PLDs) unterscheidet man zwischen ROM, PAL/PLA und FPGA/CPLD.
- Bei den FPGAs sind die Funktion der Logikzellen, Funktion der I/O-Blöcke und Verbindungen programmierbar.
- Programmiert werden schaltbare Verbindungen, (Antifuse oder TG über EPROM bzw. SRAM).

© A. Steininger / TU Wien



Zusammenfassung (3)

- Programmierbare Logikzellen lassen sich auf der Basis von Multiplexern, Look-up Tables (LUT) oder Wired AND (PAL-Struktur) realisieren.
- Bei den I/O-Blöcken sind üblicherweise Ausgangspolarität, Verzögerungen, Latches im Datenpfad, Treiberstärke/Anstiegszeit, Pull-ups, Tri-State etc. programmieren.
- Beim Interconnect bedeutet jede programmierbare Verbindung eine Vezögerung. Es muß daher ein Tradeoff zwischen Flexibilität und Geschwindigkeit gefunden werden.

© A. Steininger / TU Wien



Zusammenfassung (4)

- Der Interconnect ist bei den CPLDs völlig regulär und sein Timing daher einfach vorhersagbar.
- Die wesentlich leistungsfähigeren FPGAs haben auch einen komplexeren Interconect, der dadurch auch im PPR sowie im Zeitverhalten schwerer beherrschbar ist.
- Die Konfiguration ist bei EEPROM- und Antifuse-basierten CPLDs ist nicht-flüchtig; bei den SRAM-basierten FPGAs muss sie bei jedem power-on nachgeladen werden. Hierzu gibt es vielfältige Möglichkeiten.

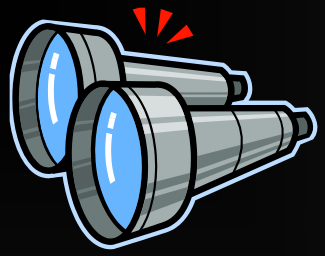
© A. Steininger / TU Wien

Datenblatt-Angaben

Unter welchen Bedingungen funktioniert mein Chip?



© A. Steininger / TU Wien



Überblick

- Temperatur und Wärmeabfuhr
- Komponenten der Verlustleistung
- Versorgungsspannung
- Spannungspegel & Störspannungsabstand
- Ausgangsstrom & Ausgangskennlinie
- Spezifikation des Zeitverhaltens
- Optimierung des Zeitverhaltens

© A. Steininger / TU Wien

Beispiel-Datenblatt



- gewähltes Beispiel für die folgenden Datenblatt-Auszüge:
  - FPGA aus der Stratix-Familie
  - Hersteller: Altera
  - Quelle: http://www.altera.com

- Bei der Laborübung verwendeter Typ:

  EP1S25F672C6

© A. Steininger / TU Wien

Absolute Maximum Ratings

Table 4–1. Stratix Device Absolute Maximum Ratings Notes (1), (2)

Symbol	Parameter	Conditions	Minimum	Maximum	Unit
V_{CCINT}	Supply voltage	With respect to ground	–0.5	2.4	V
V_{CCIO}			–0.5	4.6	V
V_I	DC input voltage *(3)*		–0.5	4.6	V
I_{OUT}	DC output current, per pin		–25	40	mA
T_{STG}	Storage temperature	No bias	–65	150	°C
T_J	Junction temperature	BGA packages under bias		135	°C

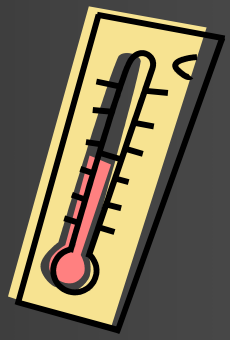
nominal 1.5V

nominal 3.3V

…sind keine Betriebsbedingungen, sondern Grenzen, bei deren Überschreitung der Chip zerstört werden kann.

© A. Steininger / TU Wien

Temperatureinfluß



- bei hoher Temperatur
  - wird der Chip langsamer
  - steigt die Fehlerrate erheblich an (exponentiell!)
- entscheidend ist „junction"-Temperatur
  - = Temperatur unmittelbar am Die
  - maximal 125...150°C, günstig < 100°C
- meßbar sind
  - Gehäuse-Temperatur („case") und
  - Umgebungs-Temperatur („ambient")

© A. Steininger / TU Wien

Wärmeabgabe an Umgebung

- Wärmeleitung und Konvektion
- es gilt vereinfachte Wärmeleitungsgleichung

$$T_{junction} = T_{ambient} + P \cdot \theta_{JA}$$

Fourier'sches Gesetz

- Umgebungstemperatur $T_{ambient}$
- Verlustleistung P des Chip:
  = Versorgungsspannung x mittl. Stromaufnahme (abzüglich von Last aufgenommener Leistung)
- θ_{JA} ... Wärmewiderstand „junction to ambient"

© A. Steininger / TU Wien

Wärmewiderstände beim Chip

- Wärmewiderstand junction/case (θ_{JC})
  - Transportmechanismus ist Wärmeleitung
  - abh. von Gehäusetyp (Material, Geometrie)
- Wärmewiderstand case/ambient (θ_{CA})
  - Transportmechanismus ist Konvektion (& Strahlung)
  - abhängig von
    - Gehäusetyp (Oberfläche)
    - Einbaulage
    - Kühlkörper / Leiterplatte
    - Lüfter,...
- Wärmewiderstand junction/ambient (θ_{JA})
  - ist Summe der obigen: $\theta_{JA} = \theta_{JC} + \theta_{CA}$

© A. Steininger / TU Wien

Wärmewiderstand θ



- Maß für das Vermögen zur Wärmeleitung
- spezifischer Wärmewid. $1/\lambda$: Materialkonstante
- absoluter Wärmewid. θ: berücksichtigt Geometrie
  - Einheit „Kelvin pro Watt" [K/W]
  - *großes* θ bedeutet *schlechte* Wärmeableitung.

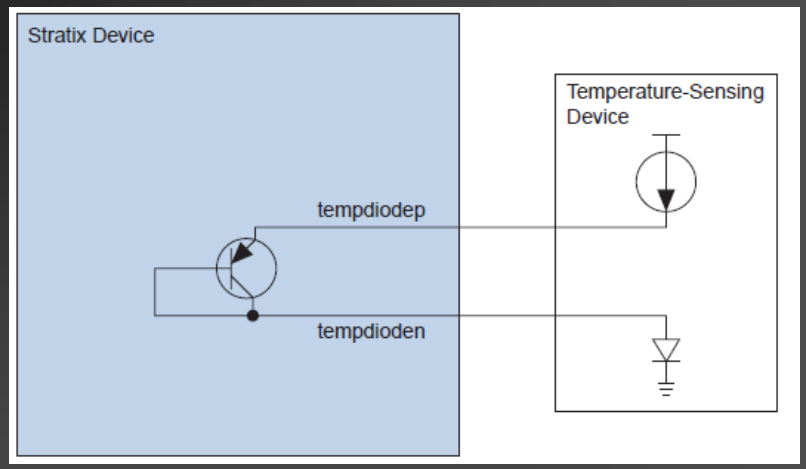
Table 8. Thermal Resistance of Stratix Devices

Device	Pin Count	Package	θ_{JC} (° C/W)	θ_{JA} (° C/W) Still Air	θ_{JA} (° C/W) 100 ft./min.	θ_{JA} (° C/W) 200 ft./min.	θ_{JA} (° C/W) 400 ft./min.
EP1S25	672	Wire Bond BGA	2.2	14.8	11.7	10.0	8.7
	672	Wire Bond FBGA	2.3	15.3	12	10.4	9
	780	Flip Chip FBGA	0.25	10.5	8.5	7.1	6.0
	1020	Flip Chip FBGA	0.25	10.0	8.0	6.6	5.5

© A. Steininger / TU Wien

Temperaturfühler bei Stratix

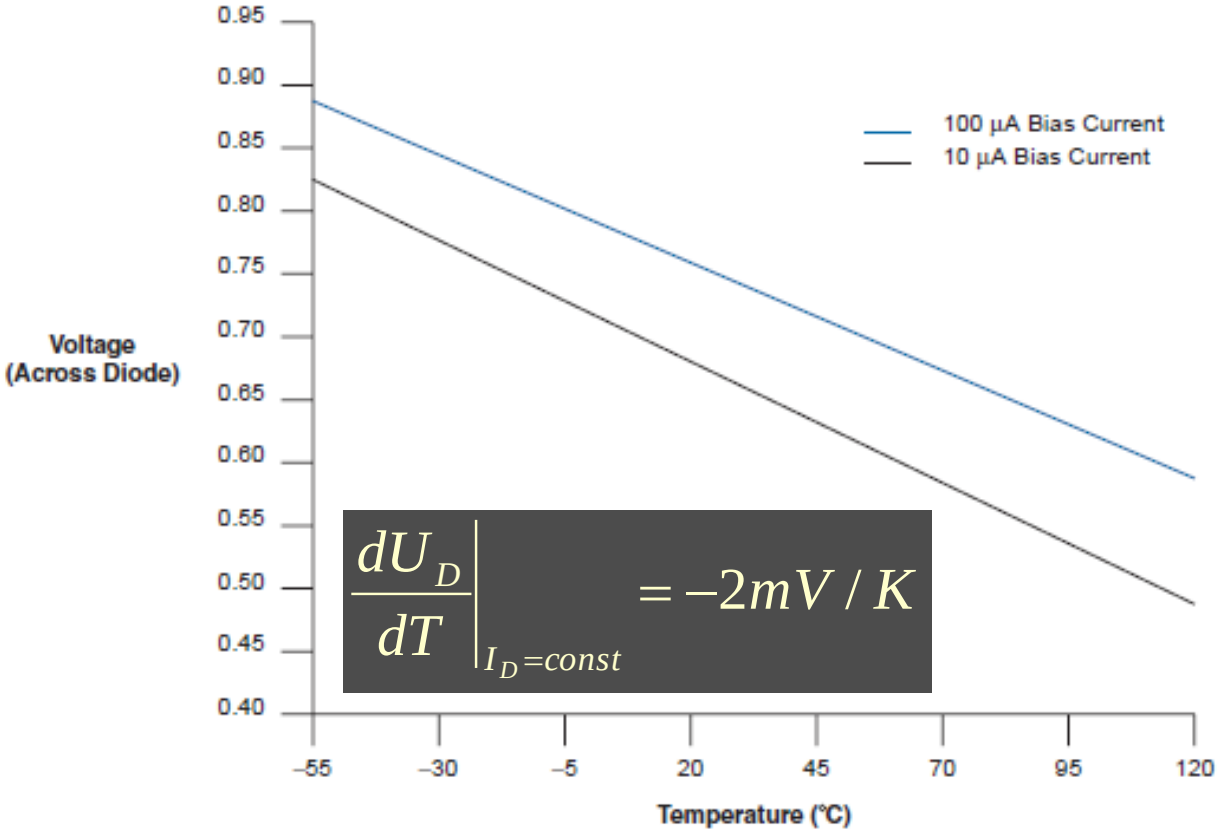
- Prinzip: temperaturabh. Verhalten von Halbleitern
- Diode: für konstanten Flussstrom ändert sich die Spannung um ca. -2mV/K
- Stratix: Diode am Die mit extern zugänglichen Anschlüssen erlaubt direkte Messung von $T_{junction}$
- ermöglicht intelligentes Power-Management



© A. Steininger / TU Wien

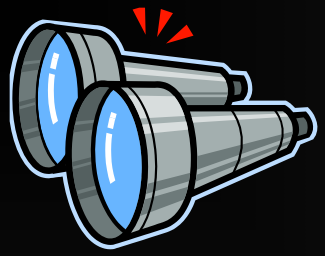
T-Fühler: Charakteristik

Figure 3–6. Temperature vs. Temperature-Sensing Diode Voltage



$$\left.\frac{dU_D}{dT}\right|_{I_D=const} = -2mV/K$$

© A. Steininger / TU Wien



Überblick

- Temperatur und Wärmeabfuhr
- Komponenten der Verlustleistung
- Versorgungsspannung
- Spannungspegel & Störspannungsabstand
- Ausgangsstrom & Ausgangskennlinie
- Spezifikation des Zeitverhaltens
- Optimierung des Zeitverhaltens

© A. Steininger / TU Wien

Verlustleistung

- statischer Anteil (= im Ruhezustand)
  - Ruheströme („quiescent current“)
    bisher vernachlässigbar (außer Batteriebetrieb!)
- dynamischer Anteil (= beim Schalten)
  - Ladeströme („switching current“)
    meist dominierend
  - transiente Kurzschlüsse („crowbar current“)
    typ. etwa 20% der Ladeströme

© A. Steininger / TU Wien

Statischer Strom bei CMOS

- Tunnelstrom durch das Gateoxid
  - steigt bei Verringerung der Schichtdicke exponentiell
- Leckströme über gesperrten Transistor
  - extrem klein, aber es gibt ja Millionen von Transistoren
  - auffällig hoher Ruhestrom weist auf einen Defekt hin
    Prinzip des IDDQ-Test
- Sperrströme über parasitäre Dioden
- Ströme über Pull-up-Widerstände
  - Open Drain Ausgänge, wired AND in PLA-Strukturen
- Ströme über eine angeschlossene Last
  - LED, Relais, ...

© A. Steininger / TU Wien

Verlustleistung

- ✓ statischer Anteil (= im Ruhezustand)
  - Ruheströme („quiescent current“)
    meist vernachlässigbar (außer Batteriebetrieb!)
- ▲ dynamischer Anteil (= beim Schalten)
  - Ladeströme („switching current“)
    meist dominierend
  - transiente Kurzschlüsse („crowbar current“)
    typ. etwa 20% der Ladeströme

© A. Steininger / TU Wien

Ladeströme bei CMOS

Periodisches Schalten von 0 auf V_{DD} mit Frequenz f bewirkt eine Verlustleistung P von:

$$P_{lade} = C_{äqu} \cdot f \cdot V_{DD}^{\ 2}$$

⇨ möglichst seltenes Umladen
(geringe Frequenz f, Schalten nur wenn nötig)

⇨ möglichst geringe Spannung V_{DD}

© A. Steininger / TU Wien

Beispiel zur Verlustleistung

- Ein Chip habe bei 1,5V Versorgung eine Junction-Temperatur von 70°C. Wieviel größer ist seine Verlustleistung bei 3,3V Versorgung und welche Die-Temperatur hätte er dann?

  (Ann: 20° Raumtemp., Ladeströme dominieren)

- $P \propto U^2$; $P(3,3V) / P(1,5V) = 3,3^2 / 1,5^2 = 4,84$;

  Die Verlustleistung steigt auf das 4,8-fache

- $T_{junction} = T_{ambient} + \theta \times P$; $T_{ambient} = 20°C$

  $\theta \times P(1,5V) = 70°C - 20°C = 50°C$

  $T_{junction}(3,3V) = 20° + 50°C \times 4,8 = \underline{260°C}$!

© A. Steininger / TU Wien

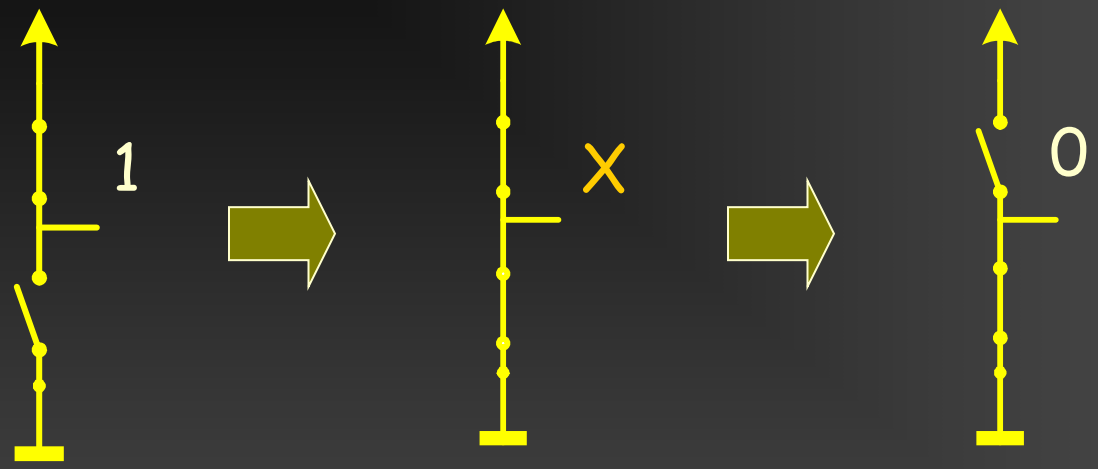
Verlustleistung

- ✓ statischer Anteil (= im Ruhezustand)
  - Ruheströme („quiescent current")
    meist vernachlässigbar (außer Batteriebetrieb!)
- dynamischer Anteil (= beim Schalten)
  - ✓ Ladeströme („switching current")
    meist dominierend
  - transiente Kurzschlüsse („crowbar current")
    typ. etwa 20% der Ladeströme

© A. Steininger / TU Wien

Transiente Kurzschlüsse

- Beispiel: Umschaltvorgang beim CMOS-Inverter

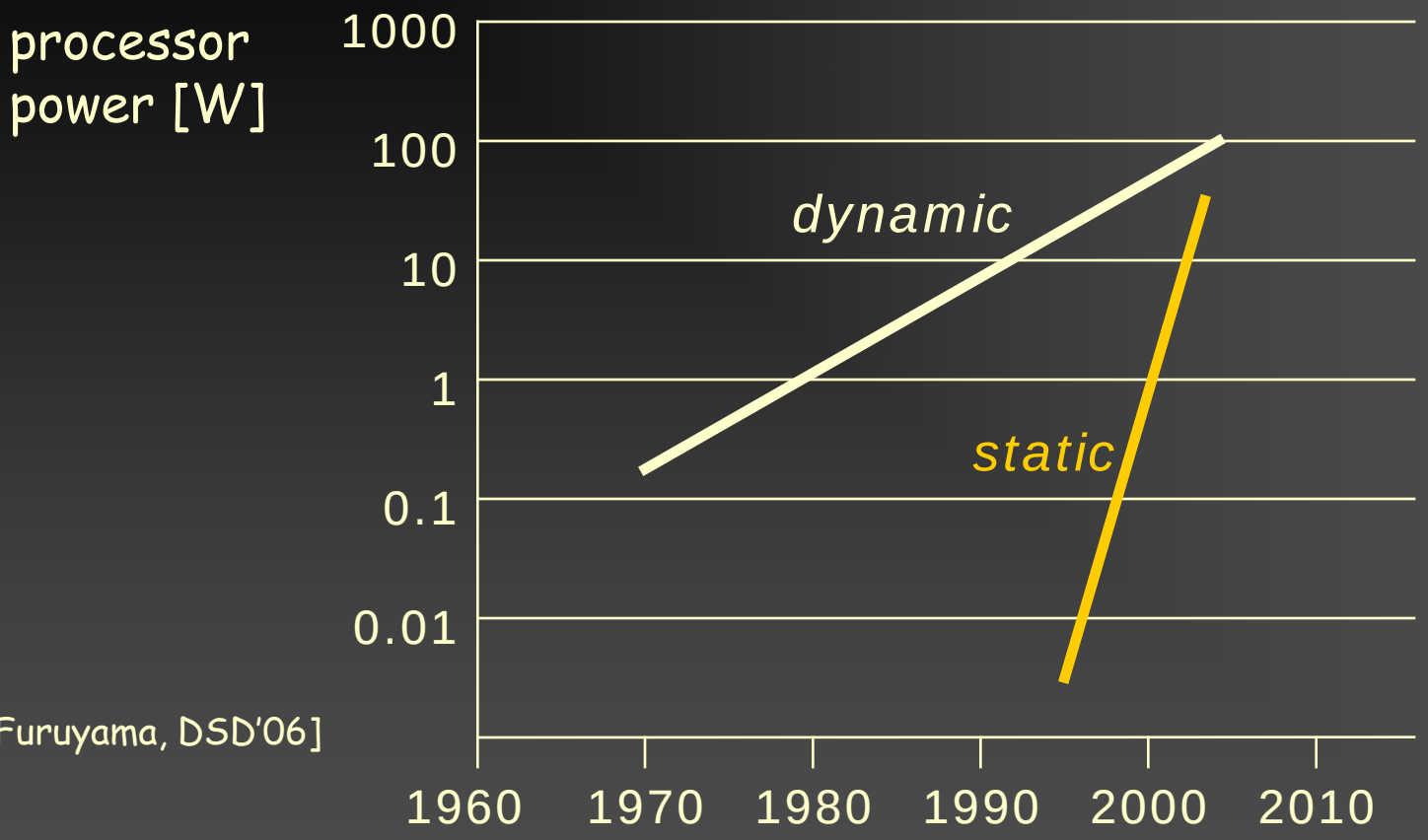


⇨ Asymmetrie der Schaltzeitpunkte klein halten
(Technologie, Zellen-Layout)

⇨ möglichst seltenes Umschalten
(geringe Frequenz, schalten nur wenn nötig)

© A. Steininger / TU Wien

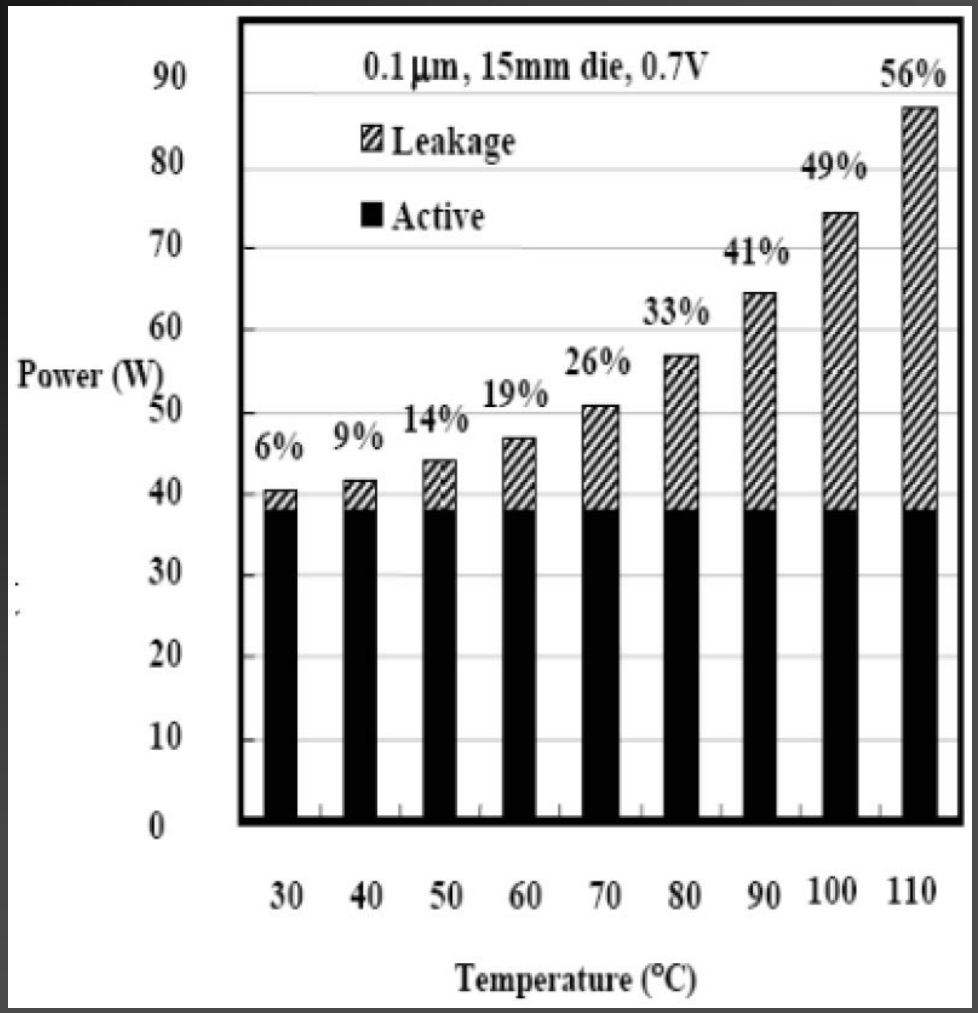
Verlustleistung: Trends



[Furuyama, DSD'06]

© A. Steininger / TU Wien

Verlustleistung: T-Abhängigkeit



[Kopetz 2010]

© A. Steininger / TU Wien

Energieverbrauch IT

"3 billion personal computers
use more than 1% of all energy consumed,
and 30 million computer servers
use an added 1.5% of all electricity
at an annual cost of $14 billion to 18 billion"

[Erik D. Demaine , MIT (2013)]

© A. Steininger / TU Wien

Energieeffizienz

… gemessen in

„computations per W“ oder

„pJ per instruction“

wird exponentiell besser (60%/a)

„…if a modern-day MacBook Air operated at the energy efficiency of computers from 1991, its fully charged battery would last all of 2.5 sec

[Jonathan Koomey, MIT (2012)]

© A. Steininger / TU Wien

Betriebstemperatur

Table 4–2. Stratix Device Recommended Operating Conditions (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Maximum	Unit
V_{CCINT}	Supply voltage for internal logic and input buffers	*(4)*	1.425	1.575	V
V_{CCIO}	Supply voltage for output buffers, 3.3-V operation	*(4)*, *(5)*	3.00 (3.135)	3.60 (3.465)	V
	Supply voltage for output buffers, 2.5-V operation	*(4)*	2.375	2.625	V
V_I	Input voltage	*(3)*, *(6)*	–0.5	4.0	V
V_O	Output voltage		0	V_{CCIO}	V
T_J	Operating junction temperature	For commercial use	0	85	°C
		For industrial use	–40	100	°C

© A. Steininger / TU Wien

Temperaturbereiche für ICs

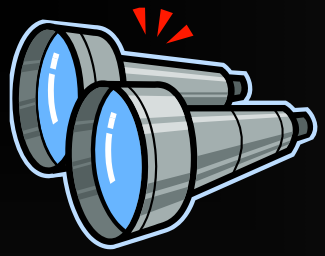
	ambient	junction
„commercial“	0°C...70°C	0°C...85°C
„industrial“	-40°C...85°C	-40°C...100°C
„military“		-55°C...125°C

(Bereiche z.T v. Hersteller selbst definiert, siehe Datenblatt)

Zusätzlich muss jedoch auch

- für ausreichende Kühlung gesorgt werden
- Die maximal zulässige Verlustleistung beachtet werden

© A. Steininger / TU Wien



Überblick

- Temperatur und Wärmeabfuhr
- Komponenten der Verlustleistung
- Versorgungsspannung
- Spannungspegel & Störspannungsabstand
- Ausgangsstrom & Ausgangskennlinie
- Spezifikation des Zeitverhaltens
- Optimierung des Zeitverhaltens

© A. Steininger / TU Wien

Versorgungsspannung

Table 4–2. Stratix Device Recommended Operating Conditions (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Maximum	Unit
V_{CCINT}	Supply voltage for internal logic and input buffers	*(4)*	1.425	1.575	V
V_{CCIO}	Supply voltage for output buffers, 3.3-V operation	*(4)*, *(5)*	3.00 (3.135)	3.60 (3.465)	V
	Supply voltage for output buffers, 2.5-V operation	*(4)*	2.375	2.625	V
	Supply voltage for output buffers, 1.8-V operation	*(4)*	1.71	1.89	V
	Supply voltage for output buffers, 1.5-V operation	*(4)*	1.4	1.6	V

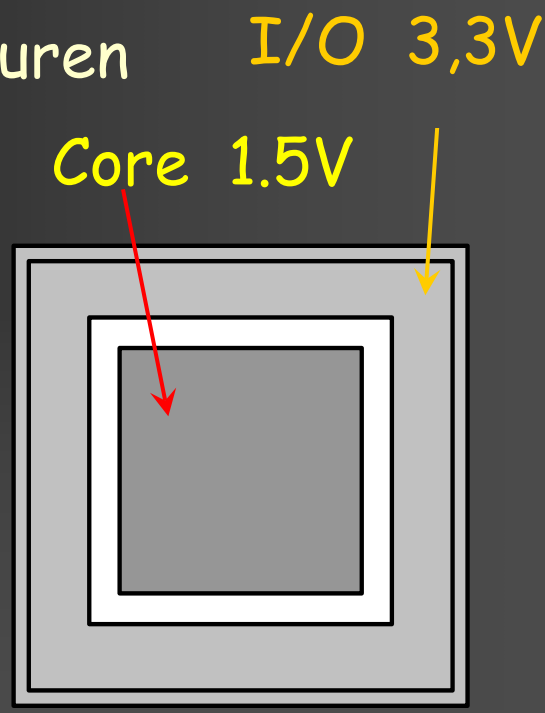
© A. Steininger / TU Wien

Gemischte Versorgung

Reduktion von 3,3V auf z.B. 1,5V Versorgung

- Verringert die Verlustleistung auf 20%
- Nötig für kleinere CMOS-Strukturen
- Oft nur für Core angewendet, I/O-Blöcke bleiben auf 3,3V
- Wegen Kompatibilität mit diversen Bus-Standards oft auch andere I/O Spannungen:

  3,3V (LVTTL, LVCMOS) / 2,5V / 1,8V / 1,5V



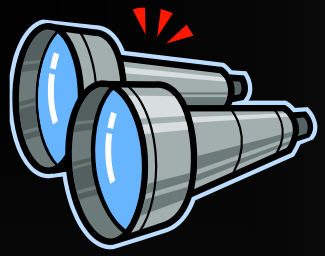
© A. Steininger / TU Wien

Ungeeignete Versorgung



- Bei zu hoher Versorgungsspannung
  - wird der Chip heiß
  - steigt daher die Ausfallsrate stark an
  - kann der Chip sofort defekt sein (Gate-Oxid!)
- Bei zu niedriger Versorgungsspannung
  - wird der Chip langsam
  - stimmen Pegel und Noise-Margins nicht mehr
  - können bei Schnittstellen unerwartet hohe Ströme auftreten

© A. Steininger / TU Wien



Überblick

- Temperatur und Wärmeabfuhr
- Komponenten der Verlustleistung
- Versorgungsspannung
- Spannungspegel & Störspannungsabstand
- Ausgangsstrom & Ausgangskennlinie
- Spezifikation des Zeitverhaltens
- Optimierung des Zeitverhaltens

© A. Steininger / TU Wien

Eingangsspannung



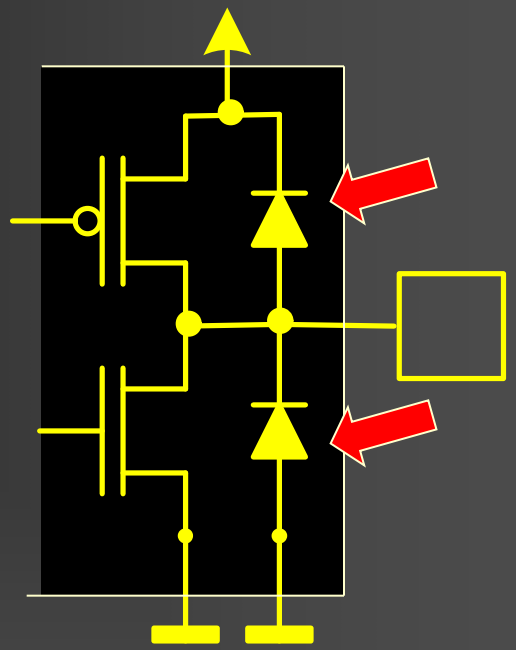
Table 4–2. Stratix Device Recommended Operating Conditions (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Maximum	Unit
V_{CCINT}	Supply voltage for internal logic and input buffers	*(4)*	1.425	1.575	V
V_{CCIO}	Supply voltage for output buffers, 3.3-V operation	*(4)*, *(5)*	3.00 (3.135)	3.60 (3.465)	V
	Supply voltage for output buffers, 2.5-V operation	*(4)*	2.375	2.625	V
V_I	Input voltage	*(3)*, *(6)*	–0.5	4.0	V
V_O	Output voltage		0	V_{CCIO}	V
T_J	Operating junction temperature	For commercial use	0	85	°C
		For industrial use	–40	100	°C

© A. Steininger / TU Wien

Klemmdioden (clamp diodes)

- …verhalten sich wie Schalter, die schließen, sobald die Spannung am IO-Pin größer als VDD + 0.6V bzw. kleiner als GND – 0.6V wird.
- Dadurch leiten sie unerwünschte Überspannungen an die Versorgung ab und schützen die CMOS-Strukturen.
- Bei zu hohen Leistungen werden sie allerdings selbst defekt.



© A. Steininger / TU Wien

Spannungspegel am Ausgang

- ideale Darstellung der Logikpegel:
  - 1 = VDD 0 = GND
- Real erfolgt aber eine Verschiebung durch
  - Schwellspannungen etc.
  - Spannungsabfälle durch Ausgangsstrom
- Abhängig von der Logikfamilie und der Versorgungsspannung gibt es Grenzwerte:
  - Minimal zulässige Ausgangsspannung für 1
  - Maximal zulässige Ausgangsspannung für 0

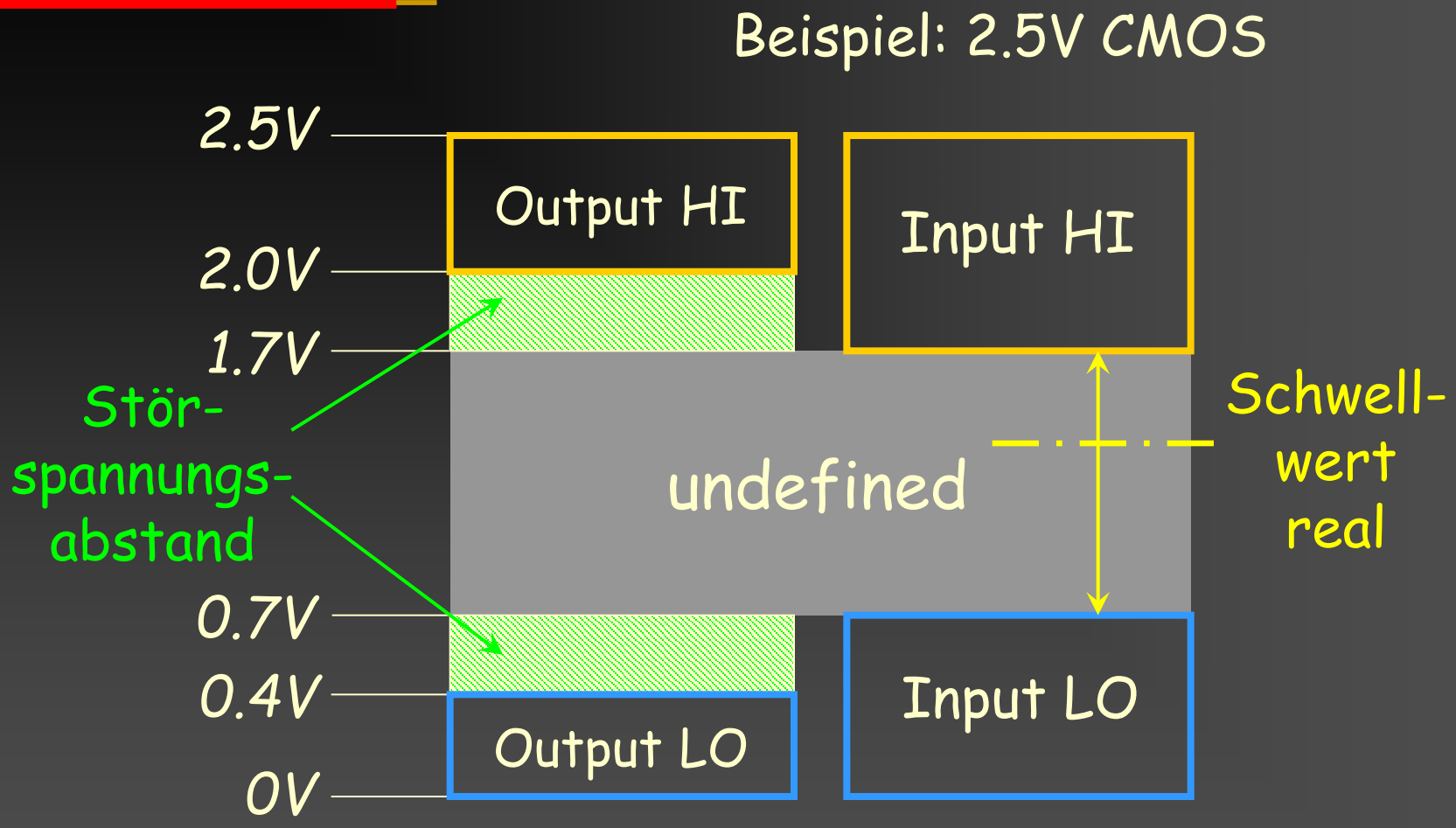
© A. Steininger / TU Wien

Spannungspegel am Eingang

- Angegeben werden
  - jener Spannungspegel unterhalb dessen der Eingang sicher als 0 erkannt wird
  - jener Spannungspegel oberhalb dessen der Eingang sicher als 1 erkannt wird
- Beide Werte haben einen „Sicherheitsabstand" zu den zugehörigen garantierten Ausgangspegeln
  => Störspannungsabstand
- De facto gibt es nur einen Schwellwert! Die spezifizierten Pegel sind nur dessen Grenzwerte.

© A. Steininger / TU Wien

Störspannungsabstand



© A. Steininger / TU Wien

Logikpegel beim Stratix/2.5V

Table 4–6. 2.5-V I/O Specifications

Symbol	Parameter	Conditions	Minimum	Maximum	Unit
V_{CCIO}	Output supply voltage		2.375	2.625	V
V_{IH}	High-level input voltage		1.7	4.1	V
V_{IL}	Low-level input voltage		–0.5	0.7	V
V_{OH}	High-level output voltage	I_{OH} = –1 mA *(10)*	2.0		V
V_{OL}	Low-level output voltage	I_{OL} = 1 mA *(10)*		0.4	V

kleinste Spannung die sicher noch als 1 erkannt wird

größte Spannung die sicher noch als 0 erkannt wird

für 1 wird zumindest diese Spannung ausgegeben

für 0 wird höchstens diese Spannung ausgegeben

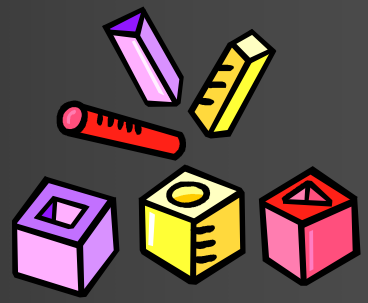
© A. Steininger / TU Wien

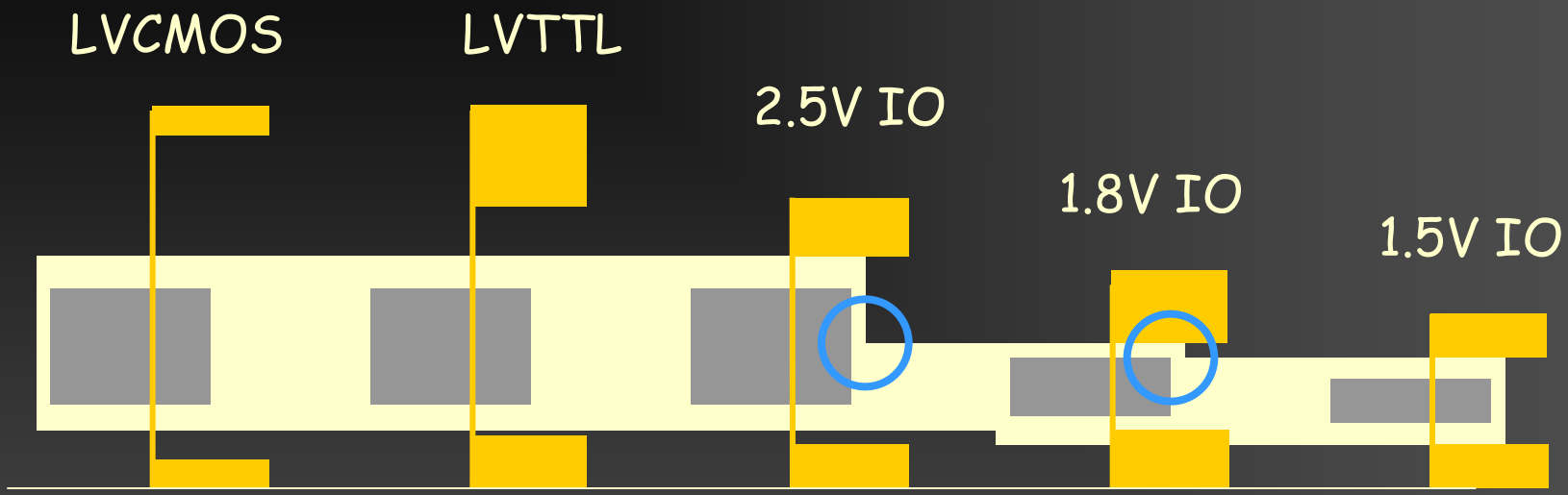
Stratix: I/O Voltages

	V_{DD} [V]	V_{OH} [V]	V_{IH} [V]	V_{OL} [V]	V_{IL} [V]	Strom [mA]
LVTTL	3.3±10%	2.4	1.7	0.45	0.7	±4...24
LVCMOS	3.3±10%	3.1	1.7	0.2	0.7	±0.1
2.5V IO	2.5±5%	2.0	1.7	0.4	0.7	±1
1.8V IO	1.8±8%	1.35	1.17	0.45	0.63	±2...8
1.5V IO	1.5±7%	1.125	0.975	0.375	0.525	±2

© A. Steininger / TU Wien

Kompatibilität





✓

zu niedrige Spannung für 1

zu niedrige Spannung für 1

© A. Steininger / TU Wien

Flankensteilheit (z.B. FLEX10K)

Table 18. FLEX 10K 5.0-V Device Recommended Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CCINT}	Supply voltage for internal logic and input buffers	*(3), (4)*	4.75 (4.50)	5.25 (5.50)	V
V_{CCIO}	Supply voltage for output buffers, 5.0-V operation	*(3), (4)*	4.75 (4.50)	5.25 (5.50)	V
	Supply voltage for output buffers, 3.3-V operation	*(3), (4)*	3.00 (3.00)	3.60 (3.60)	V
V_I	Input voltage		–0.5	V_{CCINT} + 0.5	V
V_O	Output voltage		0	V_{CCIO}	V
T_A	Ambient temperature	For commercial use	0	70	° C
		For industrial use	–40	85	° C
T_J	Operating temperature	For commercial use	0	85	° C
		For industrial use	–40	100	° C
t_R	Input rise time			40	ns
t_F	Input fall time			40	ns

© A. Steininger / TU Wien

Anstiegzeit (Rise time)

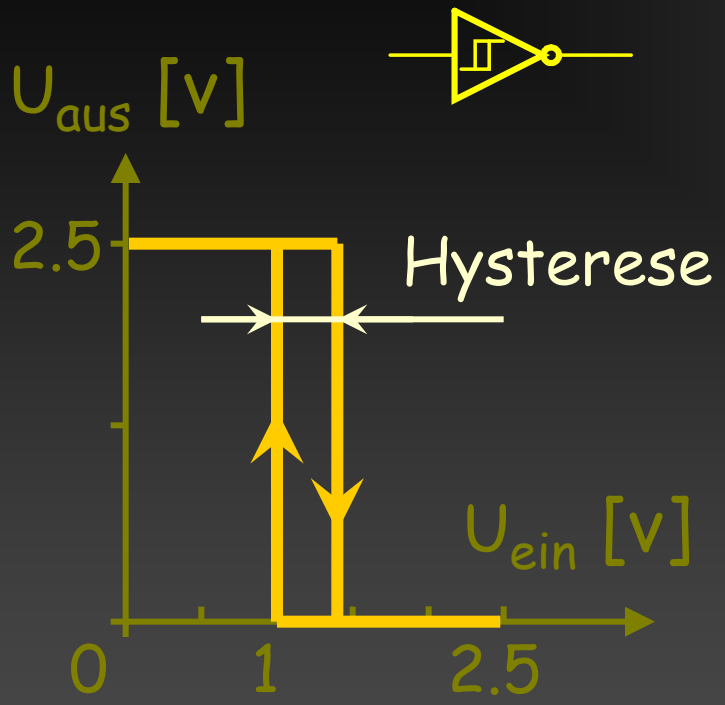


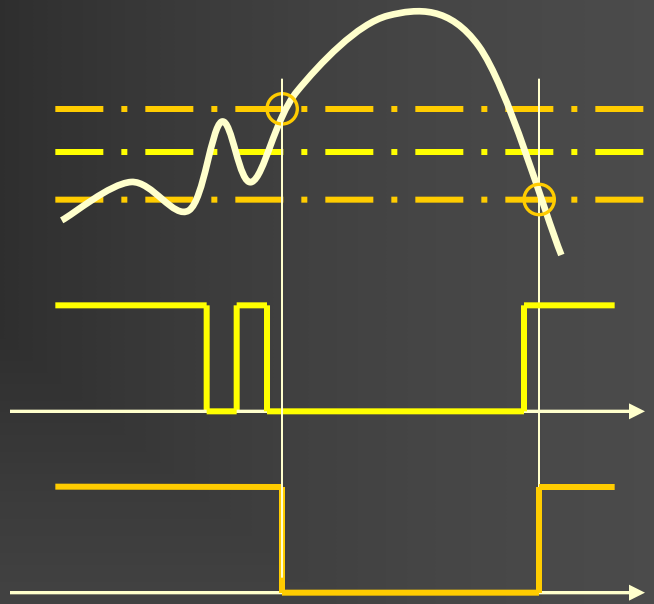


Die Anstiegszeit ist die Dauer des Signalanstiegs von 10% auf 90% des Endwertes.

© A. Steininger / TU Wien

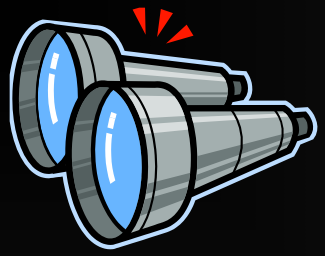
Schmitt-Trigger Eingang





typ. 2.5V FPGA: 100mV Hysterese um 1.2V herum

© A. Steininger / TU Wien



Überblick

- Temperatur und Wärmeabfuhr
- Komponenten der Verlustleistung
- Versorgungsspannung
- Spannungspegel & Störspannungsabstand
- Ausgangsstrom & Ausgangskennlinie
- Spezifikation des Zeitverhaltens
- Optimierung des Zeitverhaltens

© A. Steininger / TU Wien

Zulässige Ausgangsströme

Table 4–1. Stratix Device Absolute Maximum Ratings Notes (1), (2)

Symbol	Parameter	Conditions	Minimum	Maximum	Unit
V_{CCINT}	Supply voltage	With respect to ground	–0.5	2.4	V
V_{CCIO}			–0.5	4.6	V
V_I	DC input voltage *(3)*		–0.5	4.6	V
I_{OUT}	DC output current, per pin		–25	40	mA
T_{STG}	Storage temperature	No bias	–65	150	°C
T_J	Junction temperature	BGA packages under bias		135	°C

- Grenzwerte für Ausgangsstrom (worst case)
  - besonders bei TTL (traditionell) asymmetrisch
  - zul. dynam. Ströme viel höher (z.B. 0.2A/10ns)

© A. Steininger / TU Wien

Typische Ausgangsströme

Table 4–6. 2.5-V I/O Specifications

Symbol	Parameter	Conditions	Minimum	Maximum	Unit
V_{CCIO}	Output supply voltage		2.375	2.625	V
V_{IH}	High-level input voltage		1.7	4.1	V
V_{IL}	Low-level input voltage		–0.5	0.7	V
V_{OH}	High-level output voltage	I_{OH} = –1 mA *(10)*	2.0		V
V_{OL}	Low-level output voltage	I_{OL} = 1 mA *(10)*		0.4	V

Table 4–4. LVTTL Specifications

Symbol	Parameter	Conditions	Minimum	Maximum	Unit
V_{CCIO}	Output supply voltage		3.0	3.6	V
V_{IH}	High-level input voltage		1.7	4.1	V
V_{IL}	Low-level input voltage		–0.5	0.7	V
V_{OH}	High-level output voltage	I_{OH} = –4 to –24 mA *(10)*	2.4		V
V_{OL}	Low-level output voltage	I_{OL} = 4 to 24 mA *(10)*		0.45	V

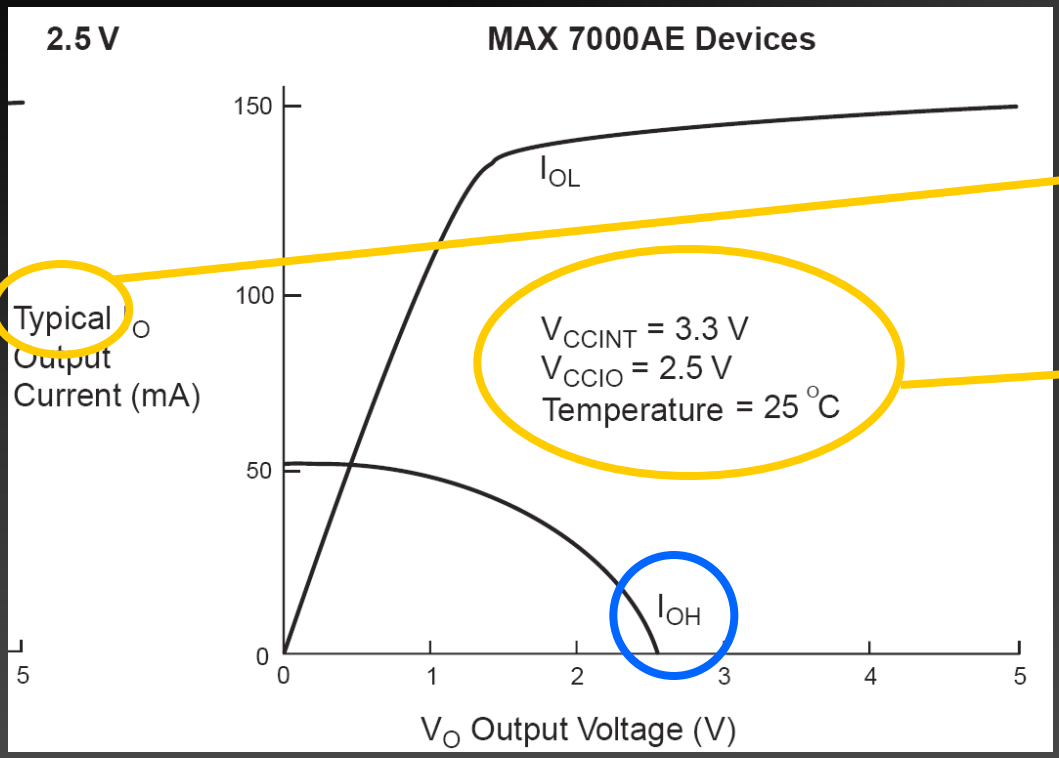
© A. Steininger / TU Wien

Wirkung der Ausgangsströme

- Wichtige Ursache für die Abweichung der Ausgangsspannung vom Idealwert
  - Spannungsabfall am Ausgangswiderstand
- Bei zu hohem Ausgangsstrom
  - Grenzwerte für 0 und 1 nicht mehr eingehalten
  - Signalverzögerung steigt, Flanken werden flach
  - Ausgangstreiber kann überlastet werden

© A. Steininger / TU Wien

Ausgangskennlinie (MAX7000)



Unterliegt Exemplar-streuungen und Abhängigkeiten (VDD, T).

Ist daher nur Richtwert (nicht worst case!)

Zusammenhang zwischen Spannung und Strom am Ausgang

© A. Steininger / TU Wien

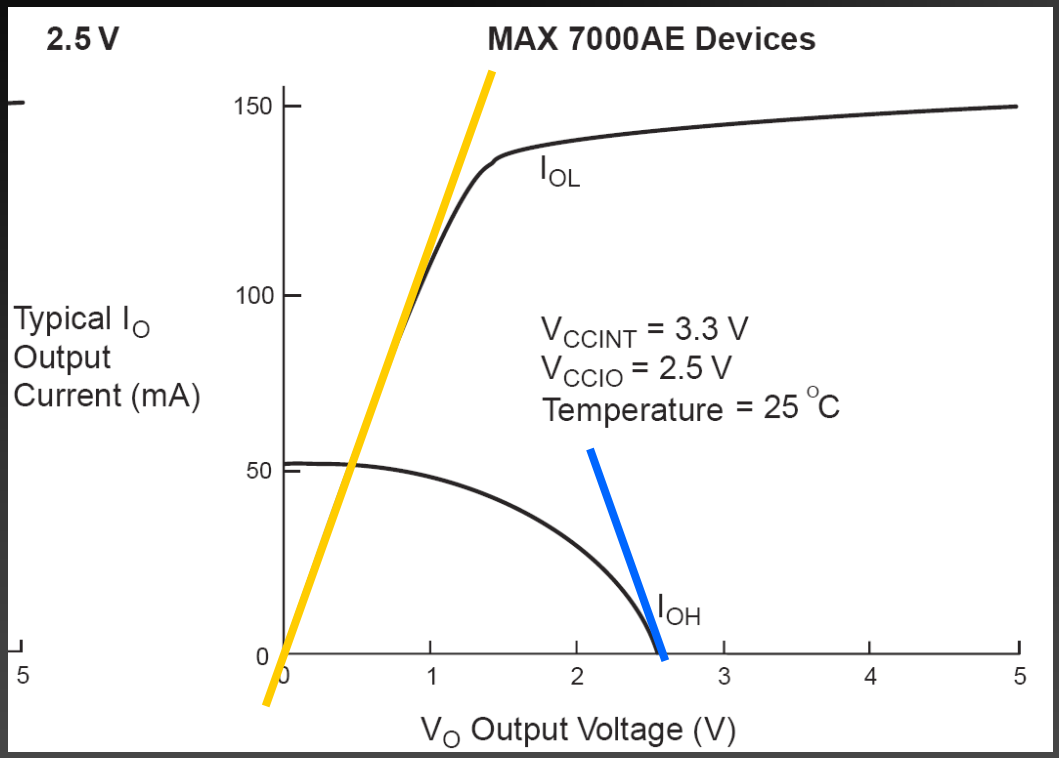
Worst Case Angaben



- sind solche, die unter den denkbar schlechtesten Bedingungen (Temperatur, Spannungsversorgung, Exemplarstreuung etc.) immer noch gültig sind.
- alle anderen Annahmen sind optimistisch, können bei Massenprodukten zu Ausfällen und schmerzhaften Mehrkosten führen (Murphy!)
  - „Typical values" beim Design völlig unbrauchbar
  - Messergebnisse an Chips bzw. funktionierender Prototyp sind allein noch kein hinreichender Funktionsnachweis für ein Massenprodukt.

© A. Steininger / TU Wien

Steigung der Kennlinie



- Steiler Anstieg bedeutet wenig Spannungsänderung bei großer Stromänderung = niedrigen Ausgangswiderstand

© A. Steininger / TU Wien

Ausgangswiderstand

- bewirkt eine Abweichung der Ausgangsspannung vom Idealwert.
- verzögert das Umladen von Kapazitäten (Leitungen, Eingänge): RC-Verhalten.
- kann aus der Steigung der Ausgangskennlinie abgelesen werden ($ra = \Delta Ua/\Delta Ia$).
- ist üblicherweise nicht konstant sondern vom Ausgangsstrom und vom Logikpegel abhängig.
- kann im Design durch die Treiberstärke (W/L) eingestellt werden. Der Idealwert Null ist jedenfalls niemals erreichbar.

© A. Steininger / TU Wien

Eingangsströme, Fan-out

Table 4–3. Stratix Device DC Operating Conditions Note (7) (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
I_I	Input pin leakage current	$V_I = V_{CCIOmax}$ to 0 V *(8)*	–10		10	µA
I_{OZ}	Tri-stated I/O pin leakage current	$V_O = V_{CCIOmax}$ to 0 V *(8)*	–10		10	µA

Ein angeschlossener Eingang belastet durch seine Eingangsströme den vorangehenden Ausgang. Der Fan-out gibt an, wie viele Eingänge (in der gleichen Technologie) an den Ausgangstreiber angeschlossen werden dürfen. Meist ist die kapazitive Belastung aber das eigentliche Limit (siehe später).

© A. Steininger / TU Wien

Überzählige Eingänge



- Problem:

  benötigte Anzahl von Inputs nicht immer genau verfügbar (Library); z.B. AND5

- Lösung:
  - größeres Gatter wählen (z.B. AND6) und überzähligen Eingang

  a) mit anderem zu Paar zusammenschließen (aber: belastet Quelle stärker!)

  b) logisch „neutralisieren“: an VDD bzw. GND (über R zum Schutz vor Spannungsspitzen)

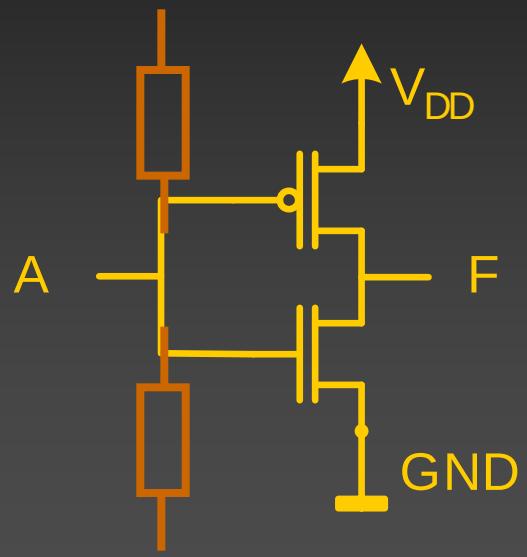
© A. Steininger / TU Wien

Offene Eingänge



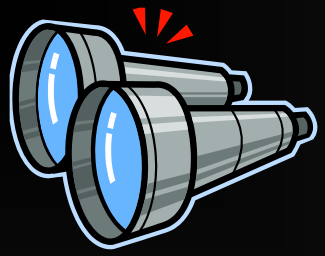
- Was passiert mit offenen Eingängen?

=> Transistormodell



- parasitäre Effekte (Isolationswiderstände und Kapazitäten) bestimmen Spannung
- bei CMOS typ. „0“ aber grundsätzl. undefiniert
- Spannungswert sehr labil & störempfindlich
- => subtile Fehler
- im FPGA automatisch korrekt behandelt

© A. Steininger / TU Wien



Überblick

- Temperatur und Wärmeabfuhr
- Komponenten der Verlustleistung
- Versorgungsspannung
- Spannungspegel & Störspannungsabstand
- Ausgangsstrom & Ausgangskennlinie
- Spezifikation des Zeitverhaltens
- Optimierung des Zeitverhaltens

© A. Steininger / TU Wien

Typische Timing-Angaben







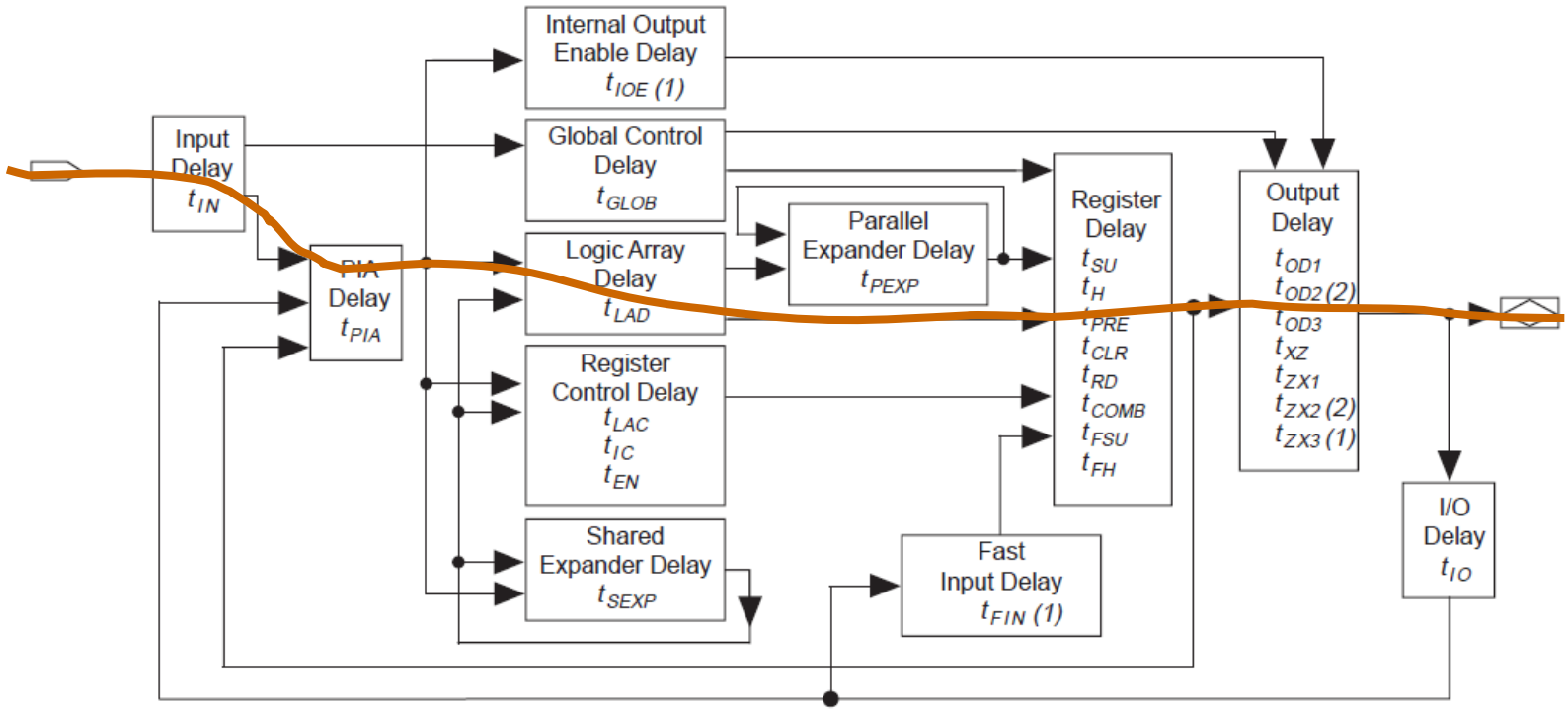
- Kombinatorische Logik
  - Propagation Delay
  - Output Enable; Output Disable
- Sequentielle Logik
  - Flip Flop Input: Setup- & Hold Time
  - Flip Flop Output: Clock to Data out
  - Steuersignale (Clk Enable, Preset, Clear) delay
- Interconnect
  - Delay für div. Leitungslängen, Switch-Elemente, Vias, …

© A. Steininger / TU Wien

Timing-Model (MAX7000)

CPLD => sehr einfach!

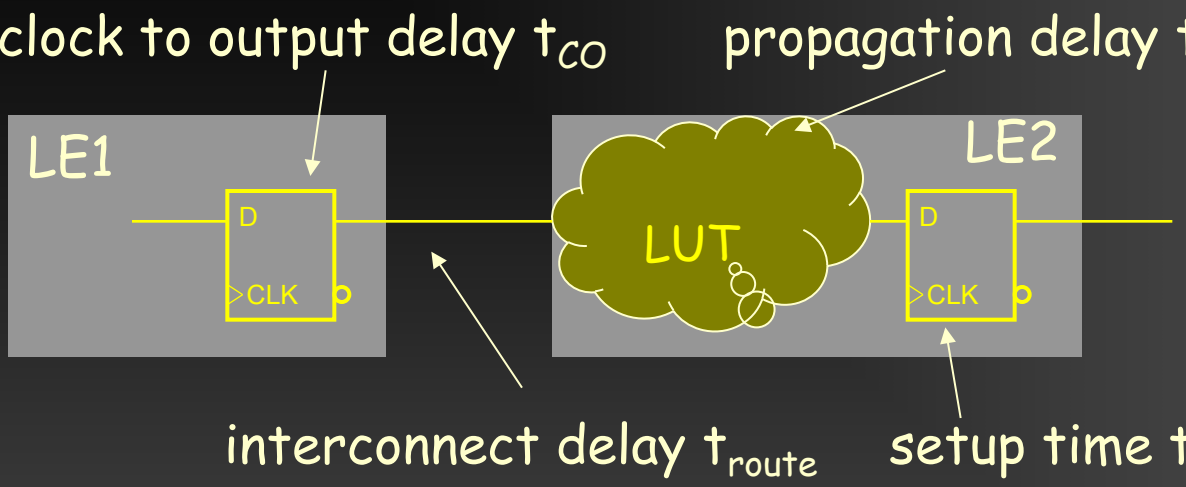
Figure 12. MAX 7000 Timing Model



Summe aller relevanten Delays im Signalpfad

© A. Steininger / TU Wien

Stratix: Beispiel zum Timing



Datenblatt: ??

$t_{CO} + t_{LUT} + t_{SU} + t_{route} = 176ps + 459ps + 10ps + 349ps = 994ps$

Max. Taktfrequenz: $f_{max} = 1 / 1ns = 1GHz$??

© A. Steininger / TU Wien

Timing-Logic Element (Stratix)

Table 4–37. LE Internal Timing Microparameter Descriptions		-6	
Symbol	Parameter	Min	Max
t_{SU}	LE register setup time before clock	10	
t_H	LE register hold time after clock	100	
t_{CO}	LE register clock-to-output delay		176
t_{LUT}	LE combinatorial LUT delay for data-in to data-out		459
t_{CLR}	Minimum clear pulse width	100	
t_{PRE}	Minimum preset pulse width	100	
t_{CLKHL}	Register minimum clock high or low time. The maximum core clock frequency can be calculated by $1/(2 \times t_{CLKHL})$.	1111	

Unit
ps

© A. Steininger / TU Wien

Speed Grades (Stratix)



Table 4–44. LE Internal Timing Microparameters

Parameter	-5		-6		-7		-8		Unit
	Min	Max	Min	Max	Min	Max	Min	Max	
t_{SU}	10		10		11		13		ps
t_{H}	100		100		114		135		ps
t_{CO}		156		176		202		238	ps
t_{LUT}		366		459		527		621	ps
t_{CLR}	100		100		114		135		ps
t_{PRE}	100		100		114		135		ps
t_{CLKHL}	1000		1111		1190		1400		ps

Speed-Grade ist nicht direkt in Performance umrechenbar, Unterschied läßt sich nur im Zusammenhang mit einer konkreten Applikation ermitteln !

© A. Steininger / TU Wien

Interconnect-Timing (Stratix)

Table 4–43. Routing Delay Internal Timing Microparameter Descriptions (Part 1 of 2)

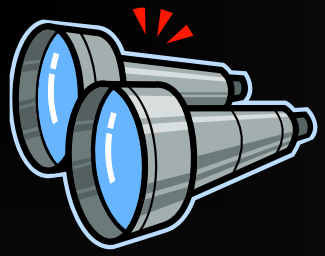
Symbol	Parameter	-6 Min	-6 Max
t_{R4}	Delay for an R4 line with average loading; covers a distance of four LAB columns.		295
t_{R8}	Delay for an R8 line with average loading; covers a distance of eight LAB columns.		349
t_{R24}	Delay for an R24 line with average loading; covers a distance of 24 LAB columns.		512
t_{C4}	Delay for a C4 line with average loading; covers a distance of four LAB rows.		484
t_{C8}	Delay for a C8 line with average loading; covers a distance of eight LAB rows.		634
t_{C16}	Delay for a C16 line with average loading; covers a distance of 16 LAB rows.		489
t_{LOCAL}	Local interconnect delay, for connections within a LAB, and for the final routing hop of connections to LABs, DSP blocks, RAM blocks and I/Os.		345

© A. Steininger / TU Wien

Timing IO-Element (Stratix)

Table 4–38. IOE Internal Timing Microparameter Descriptions		-6	
Symbol	**Parameter**	**Min**	**Max**
t_{SU_R}	Row IOE input register setup time	280	
t_{SU_C}	Column IOE input register setup time	280	
t_{H}	IOE input and output register hold time after clock	71	
t_{CO_R}	Row IOE input and output register clock-to-output delay		179
t_{CO_C}	Column IOE input and output register clock-to-output delay		179
$t_{PIN2COMBOUT\_R}$	Row input pin to IOE combinatorial output		1,295
$t_{PIN2COMBOUT\_C}$	Column input pin to IOE combinatorial output		1,141
$t_{COMBIN2PIN\_R}$	Row IOE data input to combinatorial output pin		4,089
$t_{COMBIN2PIN\_C}$	Column IOE data input to combinatorial output pin		4,494
t_{CLR}	Minimum clear pulse width	289	
t_{PRE}	Minimum preset pulse width	273	
t_{CLKHL}	Register minimum clock high or low time. The maximum I/O clock frequency can be calculated by $1/(2 \times t_{CLKHL})$. Performance may also be affected by I/O timing, use of PLL, and I/O programmable settings.	1,111	

© A. Steininger / TU Wien



Überblick

- Temperatur und Wärmeabfuhr
- Komponenten der Verlustleistung
- Versorgungsspannung
- Spannungspegel & Störspannungsabstand
- Ausgangsstrom & Ausgangskennlinie
- Spezifikation des Zeitverhaltens
- Optimierung des Zeitverhaltens

© A. Steininger / TU Wien

Timing: Einflussfaktoren

- ✓ Versorgungsspannung (Derating Factors)
- ✓ Temperatur (Derating Factors)
- ✓ Treiberstärke (Datenblatt/Library)
- ✓ Anzahl der angeschlossenen Eingänge
- ✓ Routing
- ✓ Exemplarstreuungen („worst case")

© A. Steininger / TU Wien

Wiederholung

Derating Factors (Wh.)



	4.50 V	4.75 V	5.00 V	5.25 V	5.50 V
-40°	0.77	0.73	0.68	0.64	0.61
0°	1.00	0.93	0.87	0.82	0.78
25°	1.14	1.07	1.00	0.94	0.90
85°	1.50	1.40	1.33	1.26	1.20
100°	1.60	1.49	1.41	1.34	1.28
125°	1.76	1.65	1.56	1.47	1.41

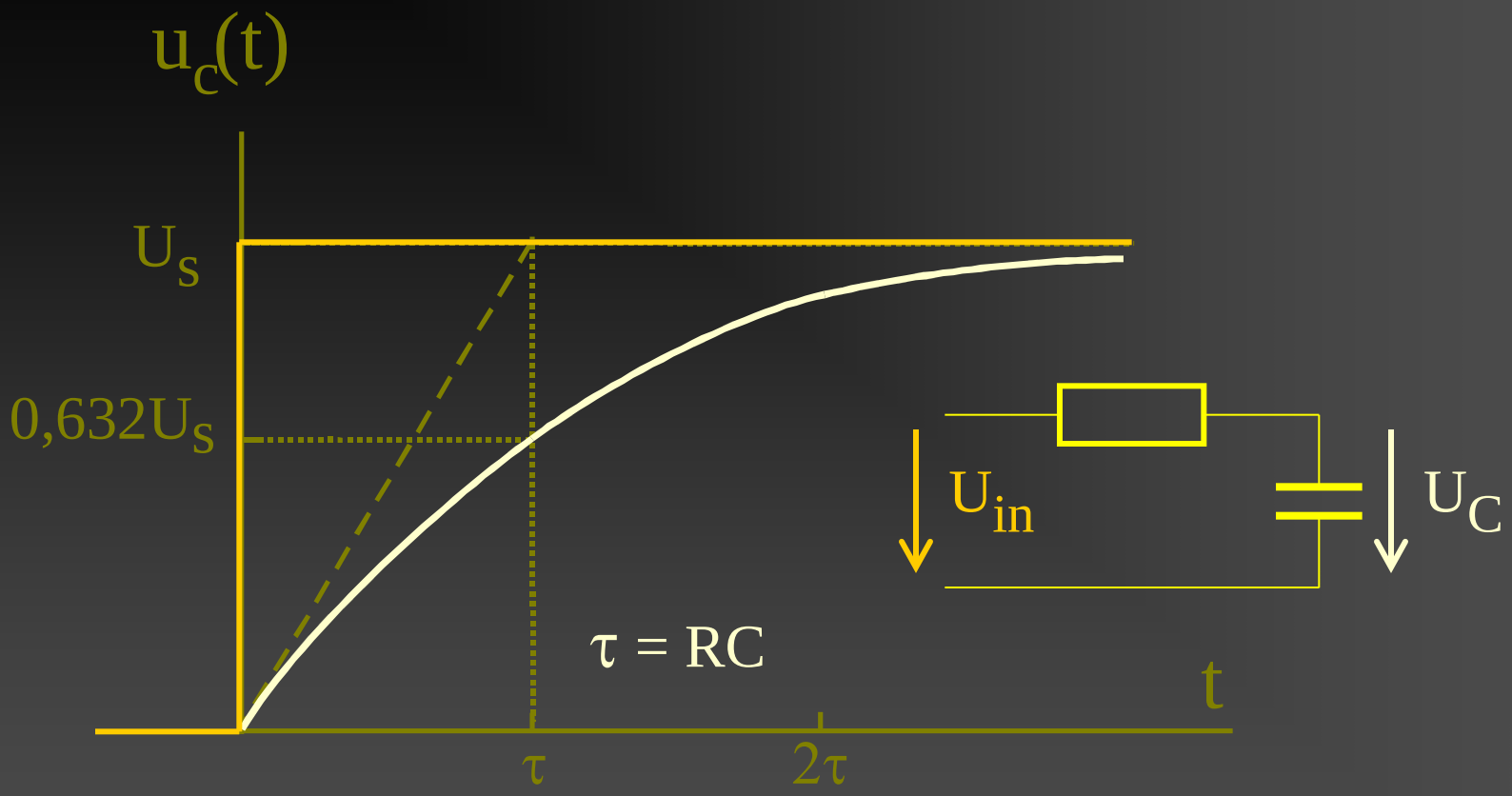
Die Angaben im Datenblatt sind der worst case innerhalb eines spezifizierten Bereiches.
Grundsätzlich gilt: Hohe Temperatur und niedrige Versorgung verlangsamen den Chip.

© A. Steininger / TU Wien

Wiederholung

Die RC-Ladekurve





© A. Steininger / TU Wien

Wodurch sind R & C bestimmt?

- Widerstand R:
  - Ausgangswiderstand (Steigung der Kennlinie)
  - Dimensionierung der FETs am Ausgang (W/L)
  - Widerstand (Länge) der Leitungen
- Kapazität C:
  - Kapazität (Länge) der Leitungen (ca. 0.2pF/cm)
  - Kapazität und Anzahl der angeschlossenen Gatter / Lasten
  - Programmierbare Verbindungen (Transistoren, Antifuses)

© A. Steininger / TU Wien

Einfluss der Lastkapazität

Table 4–111. Output Delay Adder for Loading on LVTTL/LVCMOS Output Buffers *Note (1)*

Conditions		Output Pin Adder Delay (ps/pF)				
Parameter	Value	3.3-V LVTTL	2.5-V LVTTL	1.8-V LVTTL	1.5-V LVTTL	LVCMOS
Drive Strength	24mA	15	–	–	-	8
	16mA	25	18	–	–	–
	12mA	30	25	25	–	15
	8mA	50	35	40	35	20
	4mA	60	–	–	80	30
	2mA	–	75	120	160	60

Table 4–33. Stratix Device Capacitance *Note (5)*

Symbol	Parameter	Minimum	Typical	Maximum	Unit
C_{IOTB}	Input capacitance on I/O pins in I/O banks 3, 4, 7, and 8.		11.5		pF
C_{IOLR}	Input capacitance on I/O pins in I/O banks 1, 2, 5, and 6, including high-speed differential receiver and transmitter pins.		8.2		pF

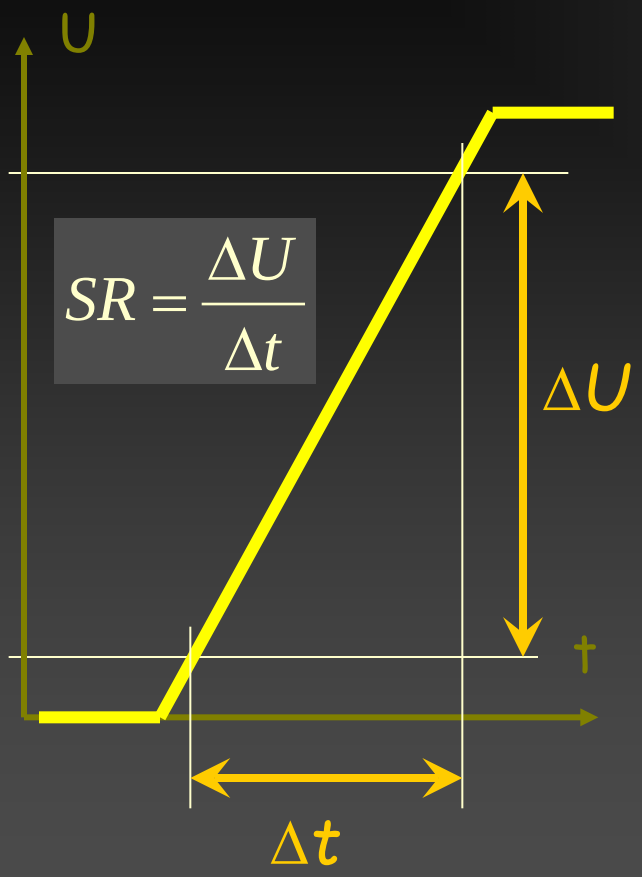
© A. Steininger / TU Wien

Optimierung des Timing

- Verwendung starker Treiber
  - kostet Chipfläche
- Verringern des Fan-out
  - z.B. durch Duplizieren von Funktionen
- Sinnvolle Constraints
  - Timing-Constraints lenken Optimierung auf das wesentliche
  - Festlegen von Pins nur wenn unbedingt nötig
  - Layout-Vorgaben (Hard Macros etc.) vermindern Optimierungspotential

© A. Steininger / TU Wien

Slew-Rate und ihre Begrenzung



- Slew-Rate (SR) ist die maximale Steilheit des Spannungsanstieges am Ausgang.
- Für hohe SR muß der Ausgang hohe Stromspitzen liefern (Laden der Cs).
- Hohe SR vermindert den Delay, die Stromspitzen verursachen jedoch Störungen in der Spannungsversorgung.

© A. Steininger / TU Wien

Konfigurierbare Slew Rate

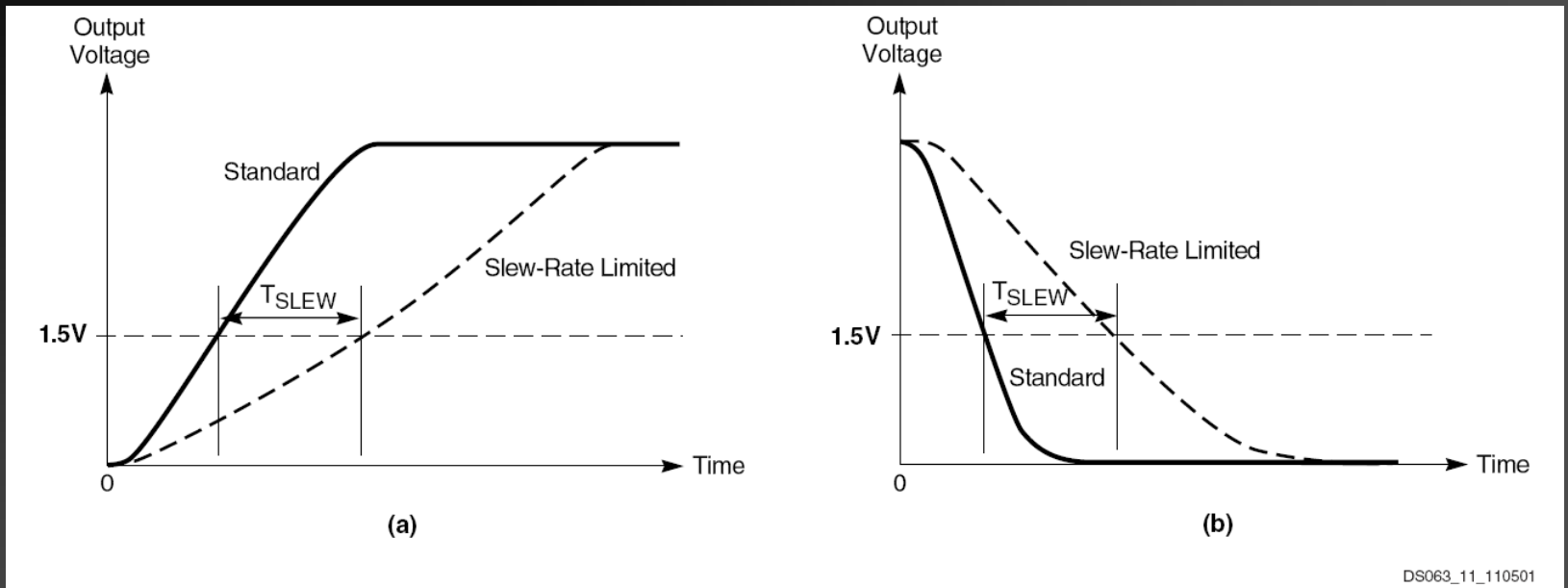
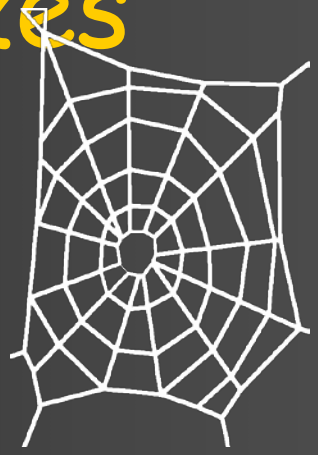


Figure 11: **Output slew-Rate for (a) Rising and (b) Falling Outputs**

[Xilinx: XC9500 Datasheet]

© A. Steininger / TU Wien

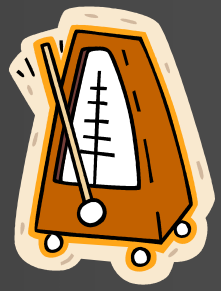
Zur Rolle des Takt-Netzes



- Am weitesten verzweigtes Netz
- Extrem hoher Fan-Out
- Höchste Schaltfrequenz
- signifikanter Anteil an der Verlustleistung
- Signalausbreitung muss im Taktnetz besonders rasch erfolgen
- Signalausbreitung muss im Taktnetz besonders gleichmäßig erfolgen (Skew!)

© A. Steininger / TU Wien

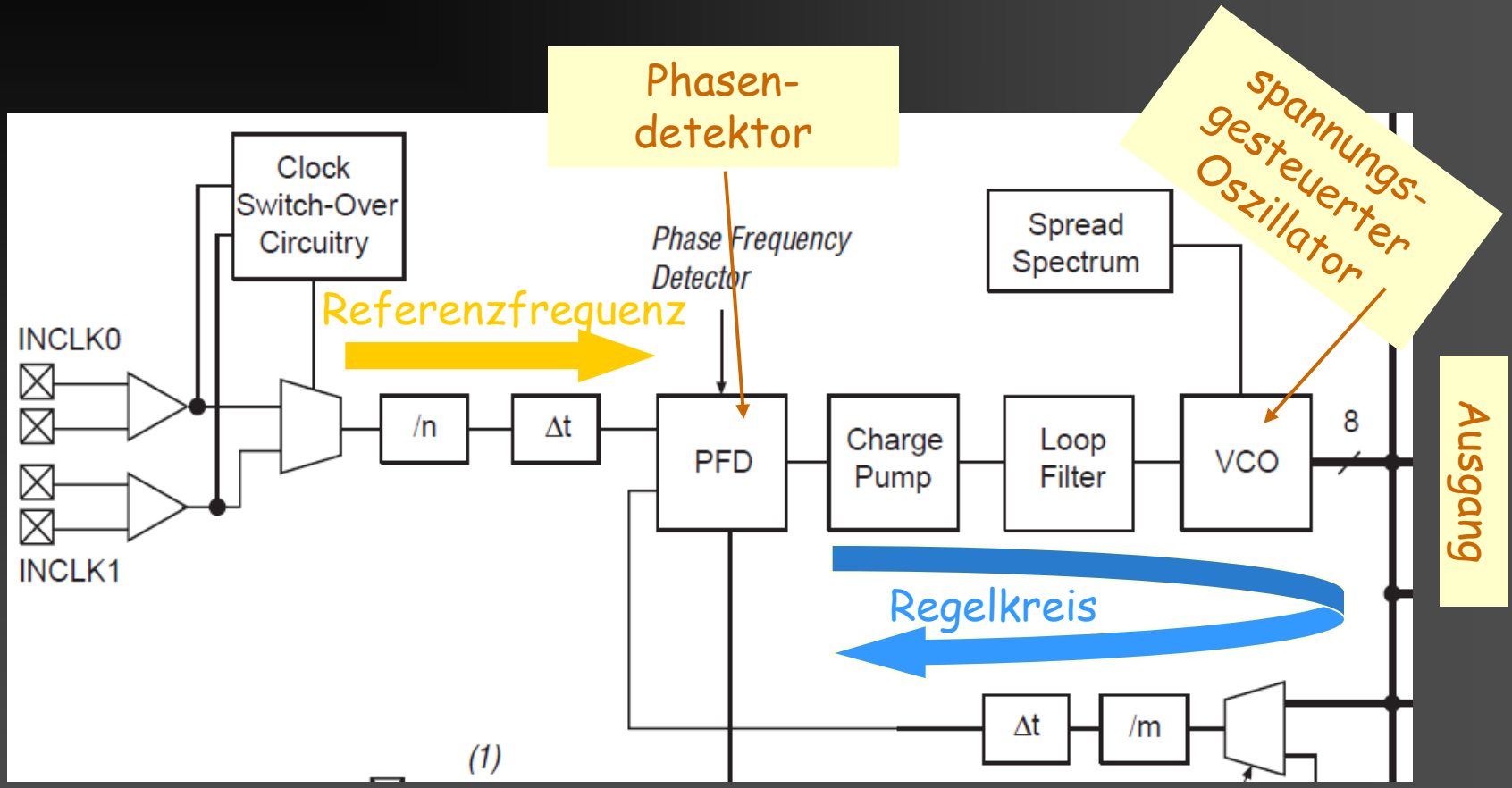
Clocking beim Stratix



- hoch reguläre hierarchische Struktur
  - 16 globale Taktnetze
  - 16 + 8 lokale Taktnetze (regional + fast regional)
  - bis zu 40 Taktdomänen
  - 16 Takteingänge (4 an jeder Kante)
  - minimales Delay & Skew
- 4 PLLs + 2 „enhanced" PLLs (Phase-locked loops)
  - Vervielfachen bzw. Teilen einer Referenzfrequenz
  - programmierbare Phasenbeziehung
  - programmierbare Verzögerung
  - programmierbares Tastverhältnis …

© A. Steininger / TU Wien

Prinzip einer PLL



© A. Steininger / TU Wien

Sonstige Features (Stratix)

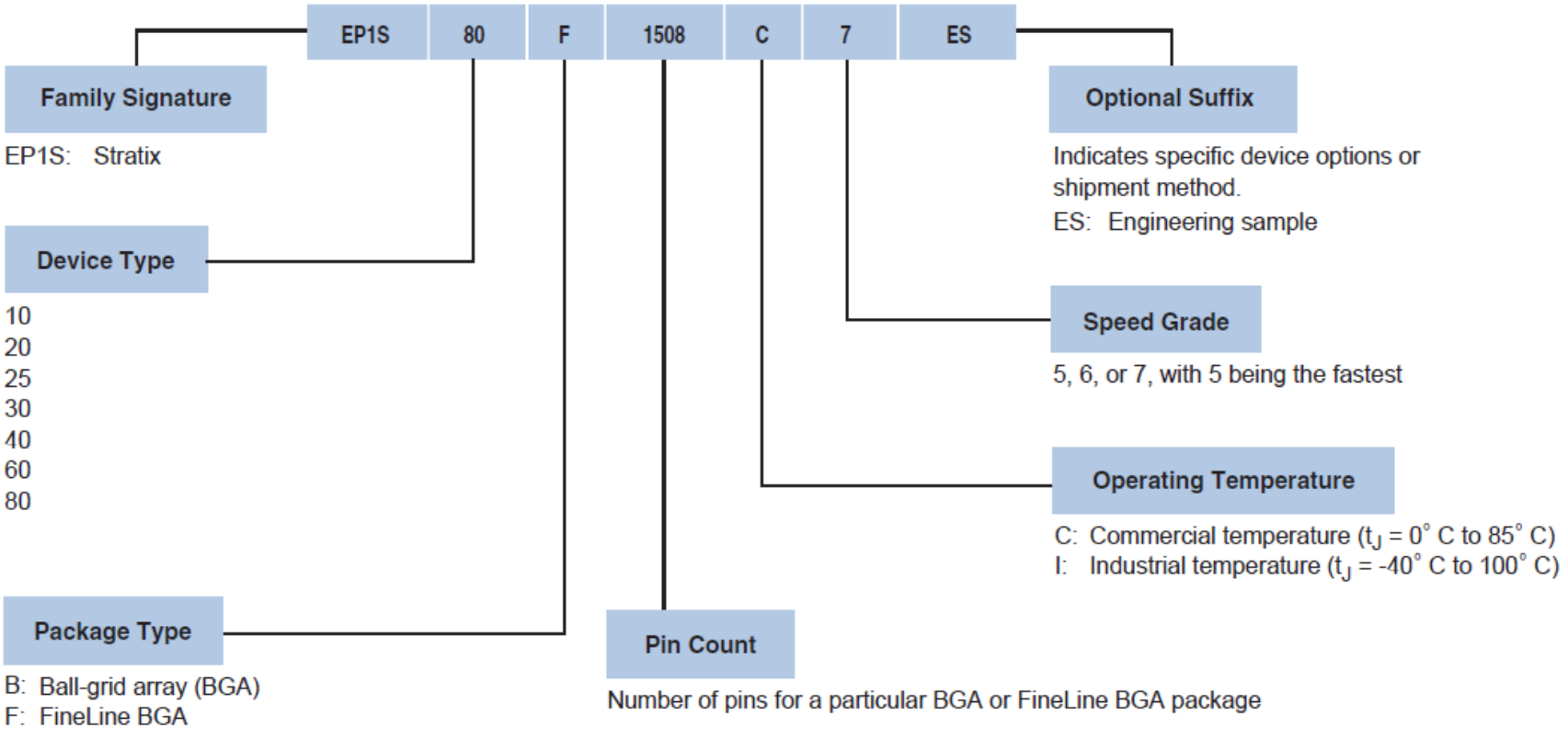
- Automatische Upset-Detection
  - mittels CRC über Konfiguration
- Eingebauter „Logikanalysator“
  - „Signal Tap“
- Vielzahl von Protokollen direkt unterstützt
  - serielle HI-speed standards (auch differentiell)
  - PCI bus
  - Speicheranbindung

© A. Steininger / TU Wien

Ordering Information

EP1S25F672C6

Figure 5–1. Stratix Device Packaging Ordering Information



© A. Steininger / TU Wien



Zusammenfassung (1)

- Die Temperatur hat einen wesentlichen Einfluß auf die Funktion eines Chips
- Entscheidend ist in jedem Fall die „Junction Temperature“. Sie ergibt sich aus Umgebungs-temperatur, Verlustleistung und thermischem Widerstand.
- Für die Umgebungstemperatur sind verschiedene Bereiche spezifiziert: commercial, industrial und military.
- Die Verlustleistung umfasst drei Komponenten: statische Ströme, Ladevorgänge und transiente Kurzschlusse.

© A. Steininger / TU Wien



Zusammenfassung (2)

- Dominant sind im aktiven Betrieb die dynamischen Ladeströme. Sie sind proportional zur Takt-frequenz und zum Quadrat der Spannung.
- Um ein Hin- und Herschalten der Eingänge zu vermeiden, müssen entweder die Signalflanken entsprechend steil sein, oder man verwendet Schmitt-Trigger-Eingänge. Diese weisen eine Hysterese auf.
- Um eine Störspannungsabstand zu gewährleisten sehen die Spezifikationen für die Ausgänge engere Grenzen vor als für die Eingänge.

© A. Steininger / TU Wien



Zusammenfassung (3)

- Der Ausgangsstrom ist stets begrenzt (Ausgangswiderstand). An einen Ausgang darf daher nur eine begrenzte Anzahl von Eingängen angeschlossen werden (Fan-Out).
- Ausserdem ergibt der Ausgangswiderstand im Zusammenwirken mit den Kapazitäten der Leitungen und der Eingänge ein RC-Glied, das die Umschaltvorgänge verzögert.
- Typische Timing-Angaben sind Setup- und Hold-Time, Durchlaufzeit, clock-to-output-delay sowie Routing delay.
- Hohe Temperatur, niedrige Versorgungsspannung und hohes Fan-Out machen einen Chip langsamer.

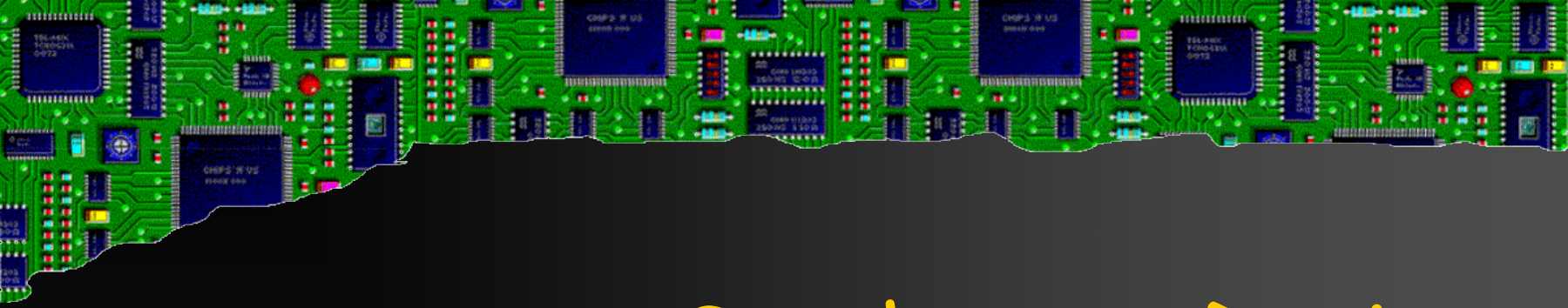
© A. Steininger / TU Wien



Zusammenfassung (4)

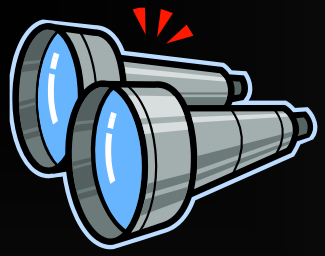
- Das Routing ist ein entscheidender Einflussfaktor für das Timing
- Das Taktnetz spielt eine besondere Rolle im Interconnect. Zur Erreichung eines geringen Delay und eines geringen Skew trotz des hohen Fan-Out und des weit verzeigten Netzes werden spezielle starke Treiber und spezielle Topologien eingesetzt.
- Taktnetze sind daher stets mit gesonderten Pins verbunden und sind im Design gesondert zu behandeln.

© A. Steininger / TU Wien



Synchrones Design und seine Grenzen

von der heilen Welt
in die Realität



Überblick

- Boolesche Logik
- Delay & Skew
- Grundproblem des Logikdesign
- Synchrones Design
  - Lösungsprinzip, Vorteile
  - konzeptioneller Mangel
  - Grenzen
- Metastabilität

© A. Steininger / TU Wien

Die Boole'sche Logik



- eindeutige zeitfreie Beschreibung
  - kombinatorischer Logik
    z.B. als Wahrheitstabelle
  - sequentieller Logik
    z.B. über Zustandsgraph (Abfolge)
- unabhängig von der Implementierung
- explizite Zeitbezüge spielen keine Rolle (außer Abfolge)



© A. Steininger / TU Wien

Die praktische Realisierung

- Es treten Signalverzögerungen auf
  - in allen Transistoren / Gattern
  - an allen Verbindungen
- Diese Signalverzögerungen
  - sind prinzipiell nicht vermeidbar
  - sind nicht deterministisch

Warum ?

© A. Steininger / TU Wien

Wiederholung

Grenzen der Geschwindigkeit



- Wellenausbreitung
  - Information kann sich niemals schneller als mit Lichtgeschwindigkeit ausbreiten.
- Ladevorgänge
  - Das Laden von Kapazitäten mit begrenztem Strom beansprucht Zeit.
- Bewegung der Ladungsträger
  - Bewegung/Diffusion von Ladungsträgern im Halbleiter erfolgt nur mit begrenzter Geschwindigkeit. (Sättigungswert bei Si typ. 0,1 mm/ns)

© A. Steininger / TU Wien

Einflußfaktoren auf Delay

- Anzahl der durchlaufenen Logikstufen
  - Komplexität der Operation
  - Optimierung & Mapping
- Routing
  - Geometrie (Länge, Abstand der Verbindungen)
  - Vias bzw. Switches
- Exemplarstreuung & Betriebsbedingungen
  - Prozessvariationen
  - Temperatur
  - Versorgungsspannung

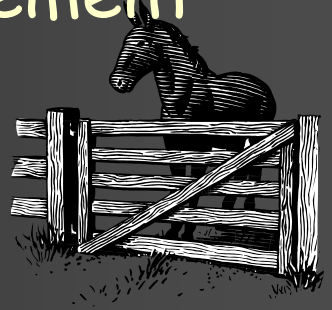
© A. Steininger / TU Wien

Wiederholung

Komponenten des Delay



- Gate Delay
  - ✓ Durchlaufzeit durch ein Logikelement
  - ✓ kaum abhängig vom Routing
  - ✓ relativ gut vorhersagbar



- Interconnect Delay
  - ✓ Signallaufzeit auf den Leitungen
  - ☹ stark abhängig vom Routing
  - ☹ schlecht vorhersagbar

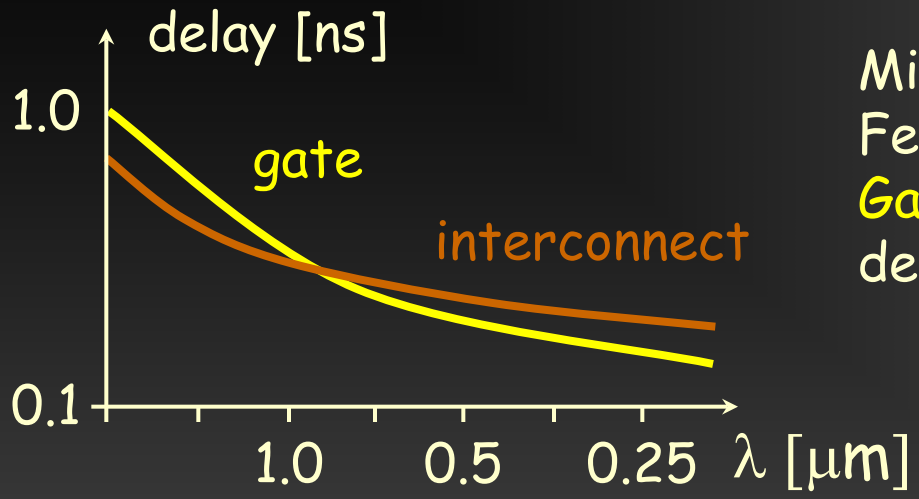


© A. Steininger / TU Wien

Wiederholung

Trends beim Delay





Mit Verkleinerung der Feature-Size sinkt der Gate Delay rascher als der Interconnect Delay

Bei den heute üblichen Technologien überwiegt der Interconnect Delay klar. Eine realistische Vorhersage des Timings ist daher erst nach dem Routing möglich, Optimierungen vor dem Routing werden immer schwieriger.

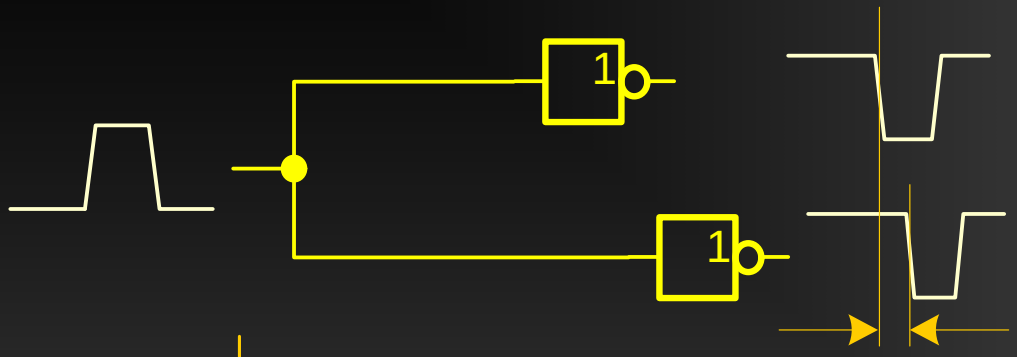
© A. Steininger / TU Wien

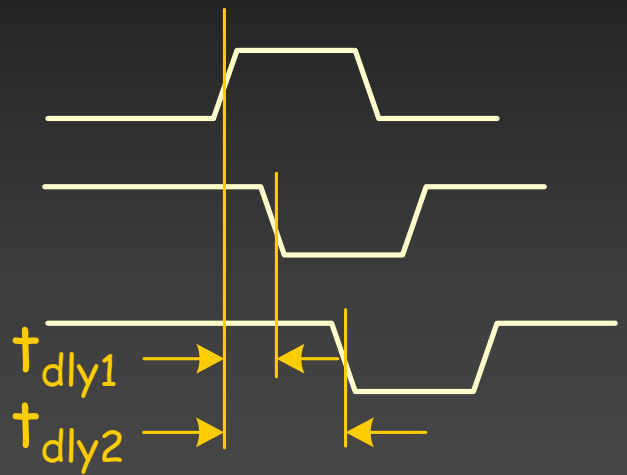
Konsequenzen für das Design

- Einer endlosen Steigerung der Geschwindigkeit von Elektronik sind Grenzen gesetzt.
- Die Laufzeiten von Signalen auf unterschiedlichen Pfaden können niemals als gleich angenommen werden, es tritt stets ein so genannter Skew auf.

© A. Steininger / TU Wien

Was ist Skew ?





$t_{skew} = |t_{dly1} - t_{dly2}|$

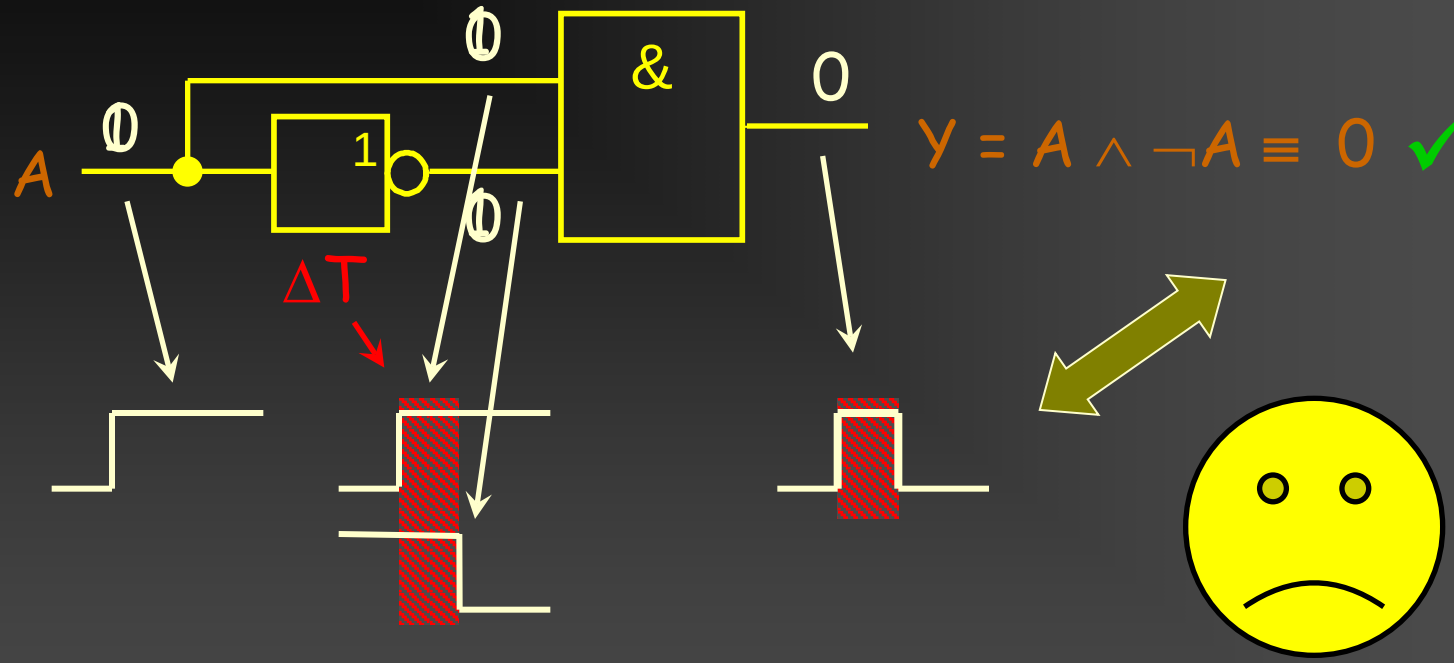
Skew ist der Unterschied im Signal-Delay

- entlang unterschiedl. Äste einer verzweigten Leitung
- entlang eines Busses
- durch gleichartige Gatter

© A. Steininger / TU Wien

Folgen des Skew





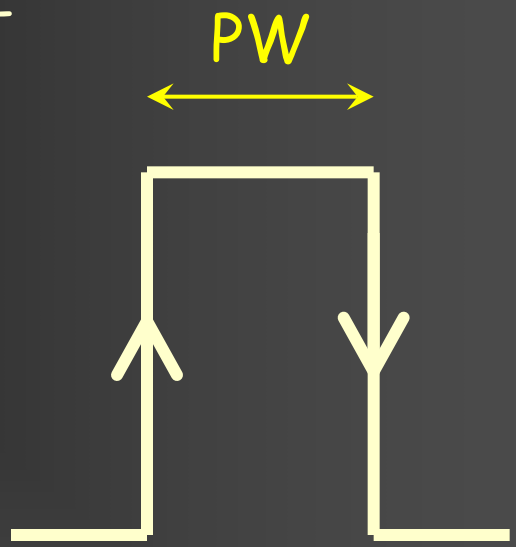
$Y = A \wedge \neg A \equiv 0$ ✓

Im eingeschwungenen Zustand ist Y immer "0"
Beim Umschalten treten jedoch Glitches auf

© A. Steininger / TU Wien

Pulse & Glitches

- **Puls:** Abfolge zweier entgegengesetzt gerichteter Flanken
  - positiv: ↑↓
  - negativ: ↓↑
- **Pulsbreite:** Zeitabstand (PW) zwischen diesen Flanken
- **Glitch:** sehr kurzer, meist unerwünschter Puls



© A. Steininger / TU Wien

Hazard…

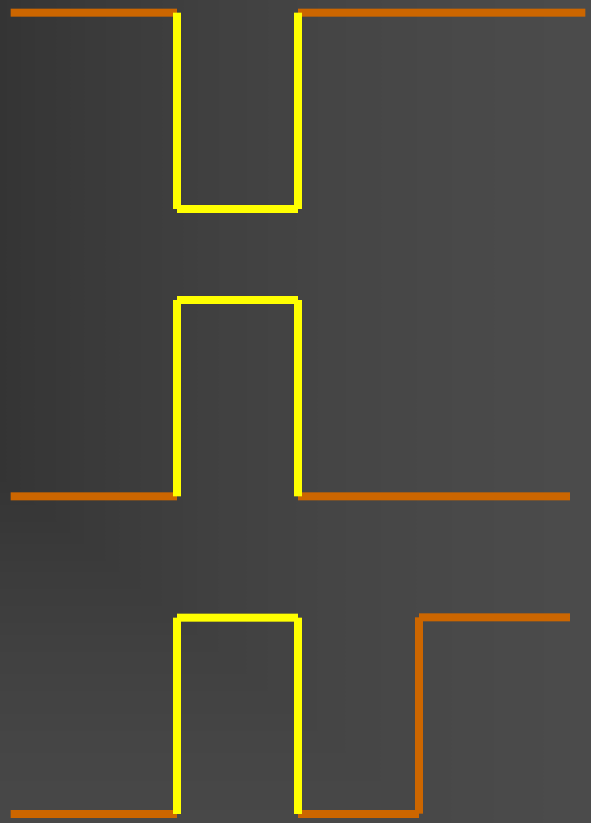


- …ist die Möglichkeit für das Entstehen von Glitches in einer Schaltung (abh. von relativen Laufzeiten, Temperatur, Versorgung, Eingangsmustern).
- Ein Glitch ist somit die Manifestation eines Hazard in einer Implementierung der Schaltung.
- Ein Glitch wird gefährlich, wenn er (von seiner transienten Erscheinung) in einen Zustand übernommen wird, indem
  - seine Flanke eine Auswirkung hat (Steuersignal), oder
  - der transiente falsche Wert in ein Speicherelement übernommen wird.

© A. Steininger / TU Wien

Kombinatorische Hazards

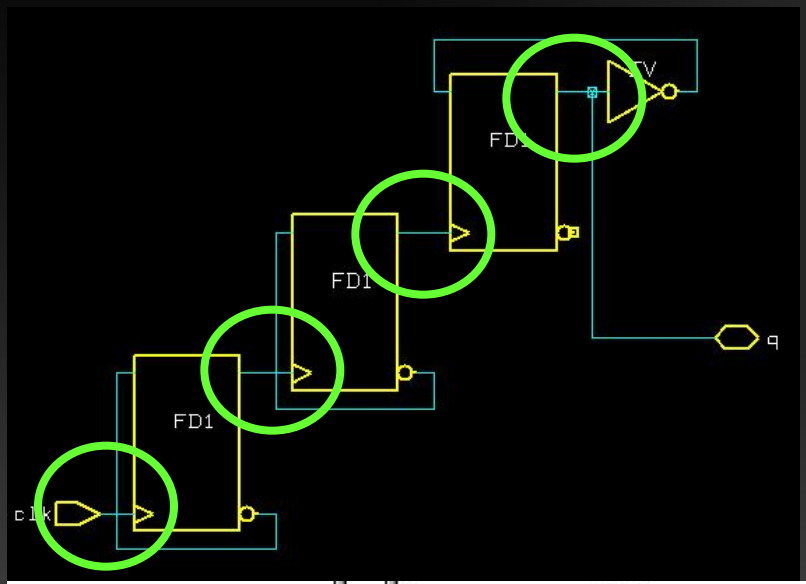
- **static 1:** unerwünschter negativer Glitch bei einem Übergang von 1 auf 1
- **static 0:** unerwünschter positiver Glitch bei einem Übergang von 0 auf 0
- **dynamischer Glitch:** Glitch vor einem vorgesehenen Zustandswechsel



© A. Steininger / TU Wien

Asynchroner Teiler





"Zählerstände":

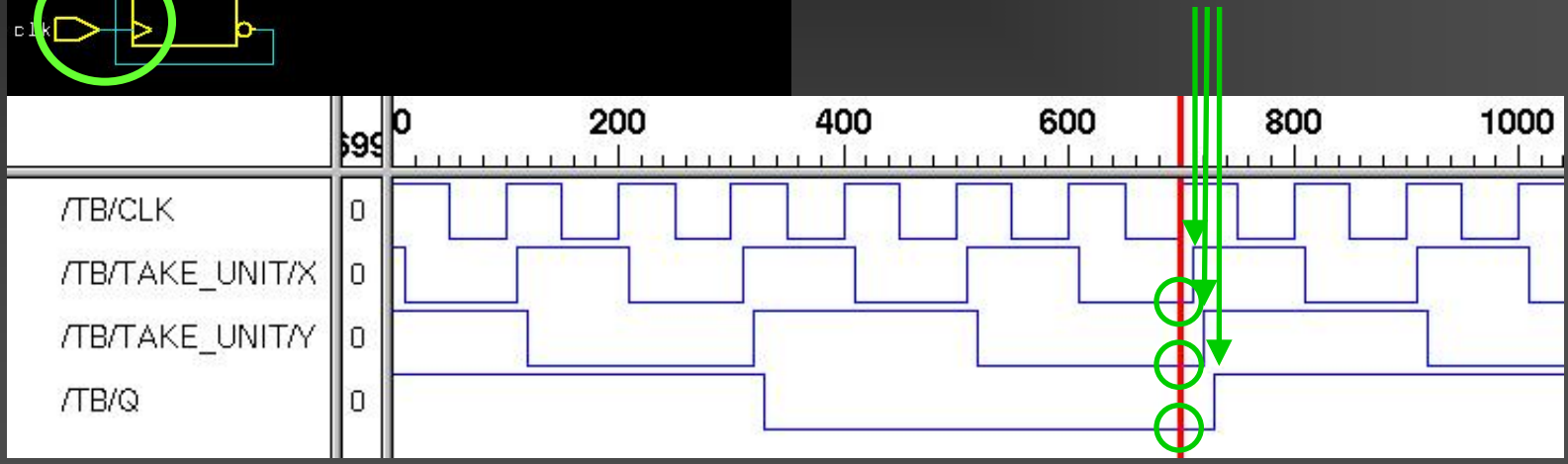
(Q,Y,X) = 0,0,0

~~0,0,1~~

~~0,1,1~~

ungültig

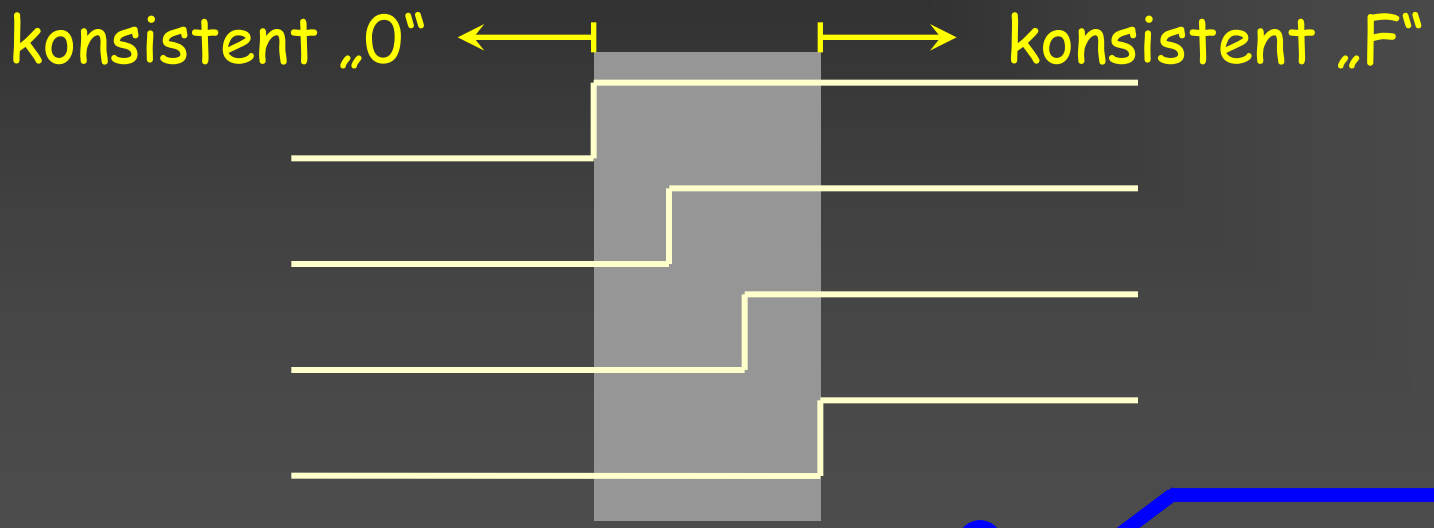
1,1,1



© A. Steininger / TU Wien

Datenkonsistenz

- Einzel-Daten die für eine gemeinsame Interpretation vorgesehen sind, müssen zeitlich richtig korreliert sein
- Beispiel „asynchroner Zähler“ von vorhin:



© A. Steininger / TU Wien

Die Boole'sche Logik



- eindeutige zeitfreie Beschreibung
  - kombinatorischer Logik
    z.B. als Wahrheitstabelle
  - sequentieller Logik
    z.B. über Zustandsgraph (Abfolge)
- unabhängig von der Implementierung
- explizite Zeitbezüge
  ~~spielen keine Rolle~~
  sind nicht darstellbar !



© A. Steininger / TU Wien

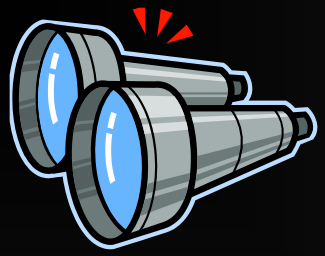
Ein kurzes Resumee ...

- Boole'sche Logik beschreibt das Input/ Output-Mapping zeitfrei
- Es wird also von kontinuierlich gültigen Eingängen ausgegangen
- Skew verursacht aber inkonsistente Eingänge und daher ungültige Zwischenzustände

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



© A. Steininger / TU Wien



Überblick

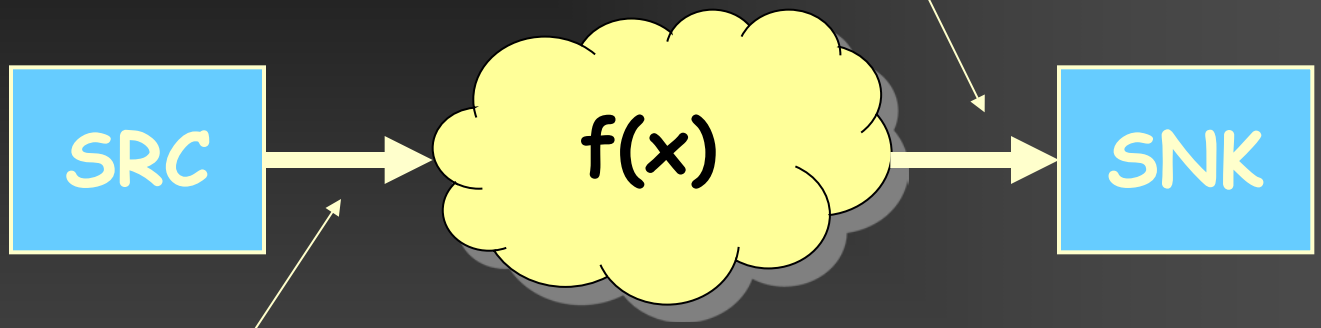
- Boolesche Logik
- Delay & Skew
- Grundproblem des Logikdesign
- Synchrones Design
  - Lösungsprinzip, Vorteile
  - konzeptioneller Mangel
  - Grenzen
- Metastabilität

© A. Steininger / TU Wien

Wozu eine Design-Methode?

Wann kann SNK die Daten verwenden?

Wenn sie gültig und konsistent sind



Wann kann SRC die nächsten Daten senden?

Wenn SNK die vorigen konsumiert hat

© A. Steininger / TU Wien

Grundproblem beim Design

- ✓ Es darf nur mit konsistenten, eingeschwungenen Zuständen operiert werden
- ? Wie unterscheidet man konsistente Zustände von Übergangszuständen oder Glitches ?

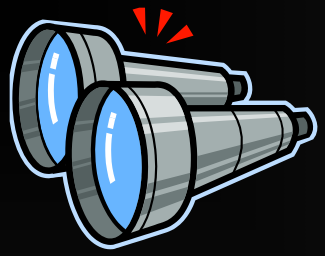
Zeitbedingungen für Konsistenz

- synchrones Design
- asynchrone „bounded delay" Verfahren (Timer)

Hinzufügen von „Meta-Information"

- asynchrone „delay-insensitive" Verfahren (Codierung)

© A. Steininger / TU Wien



Überblick

- Boolesche Logik
- Delay & Skew
- Grundproblem des Logikdesign
- Synchrones Design
  - Lösungsprinzip, Vorteile
  - konzeptioneller Mangel
  - Grenzen
- Metastabilität

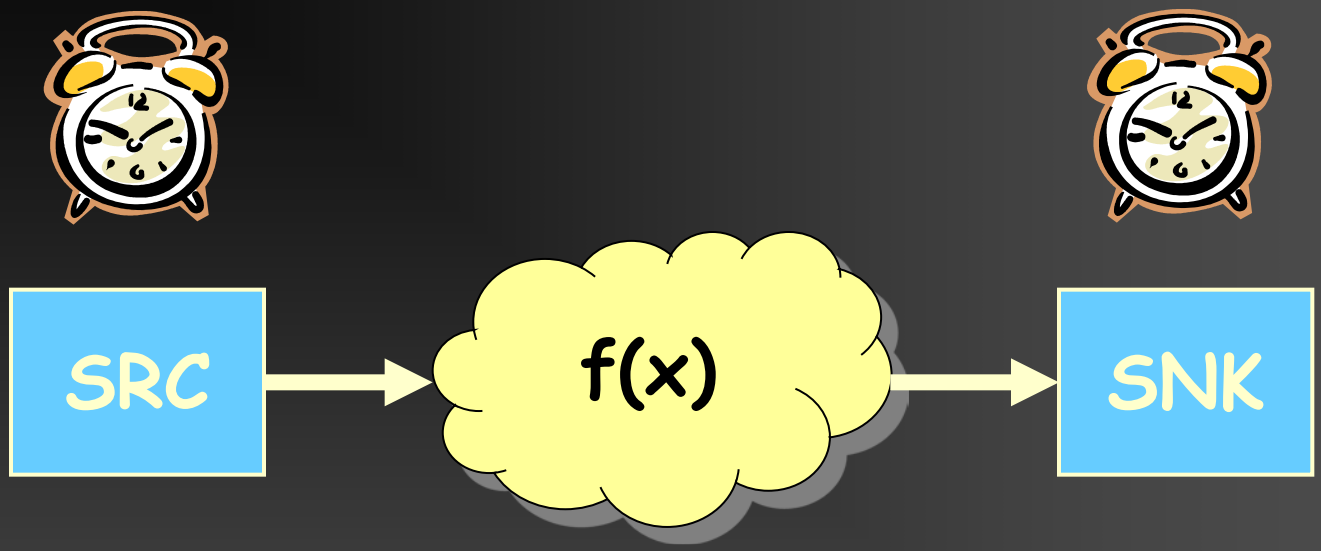
© A. Steininger / TU Wien

Synchrones Design: Philosophie

- „Wenn der Ursprung Problems im Zeitbereich liegt, dann ist es auch im Zeitbereich lösbar“
- Eingänge werden erst verarbeitet, wenn alle Einschwingvorgänge sicher beendet sind.
- Taktsignal gibt jeweils die Zeitpunkte vor, zu denen die Daten gültig sind.

© A. Steininger / TU Wien

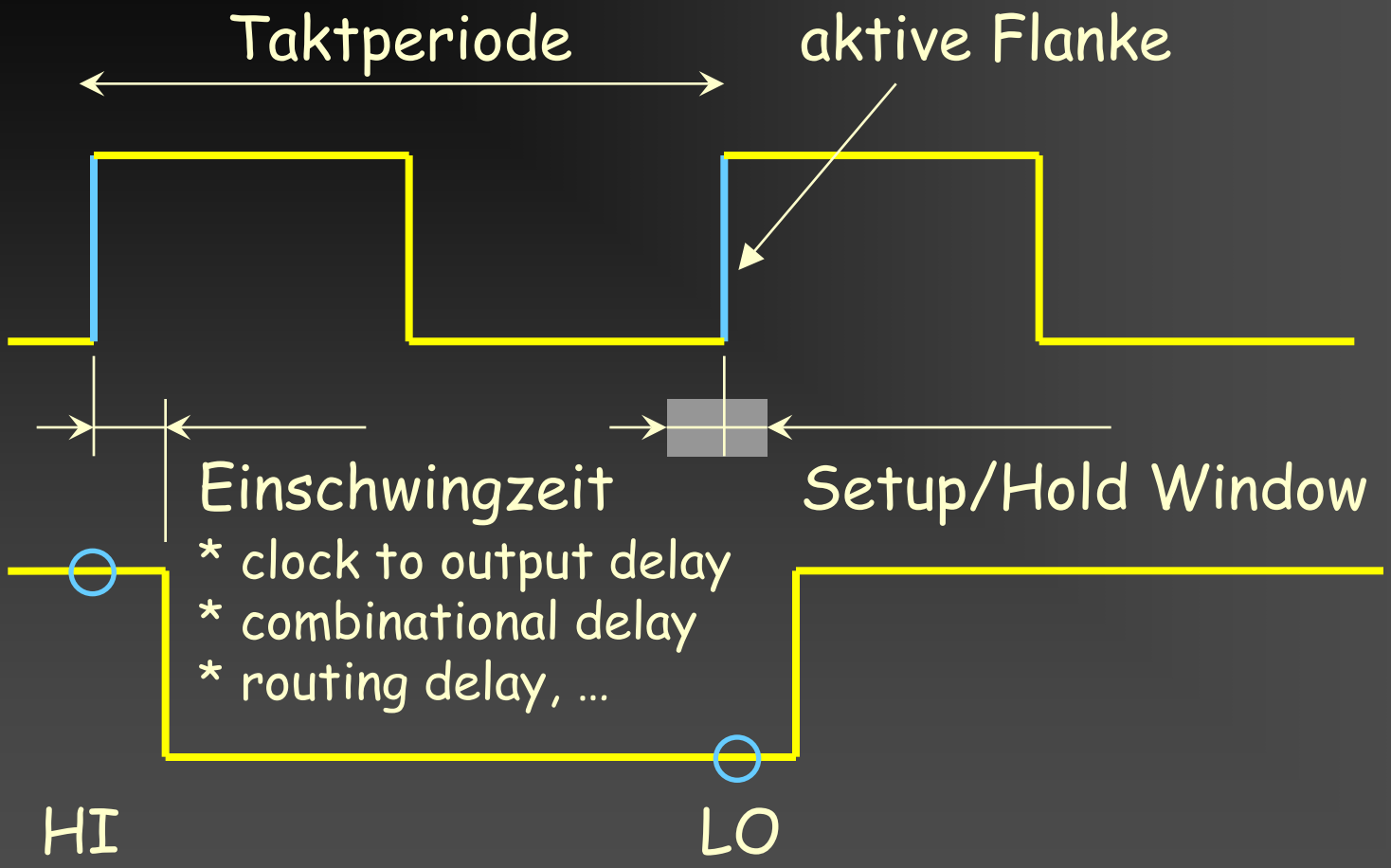
Der synchrone Ansatz



Nach einer bestimmten ZEIT T_{clk} kann SNK den Ausgang von f(x) verwenden, und zur gleichen ZEIT kann SRC neue Daten senden.

© A. Steininger / TU Wien

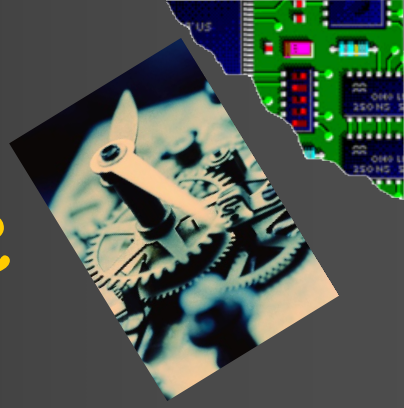
Timing bei synchroner Logik

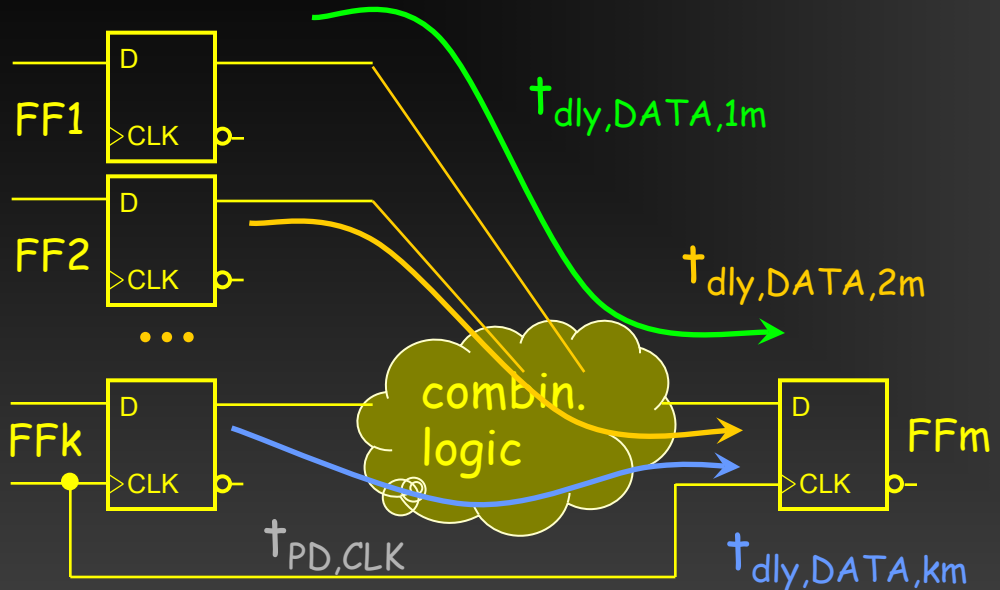


© A. Steininger / TU Wien

Wiederholung

Statische Timinganalyse





Delays im Daten- und Clock-Pfad werden für jedes Flip-Flop systematisch analysiert

$$1/f_{CLK,max} = \max(t_{dly,DATA,ij}) + t_{SU} - \min(t_{dlyD,CLK,i})$$

- ✓ Überprüfung vorgegebener Constraints
- ✓ Ermittlung der maximal zulässigen Taktfrequenz

© A. Steininger / TU Wien

Synchrone digitale Logik

- **"Digital"** = Amplituden nicht mehr kontinuierlich
  - es werden nur zwei Logikpegel unterschieden
  - eigentliche Spannungsverläufe sind irrelevant
- **"Synchron"** = Zeit nicht mehr kontinuierlich
  - Logikpegel werden nur zu bestimmten Zeitpunkten betrachtet (aktive Taktflanke)
  - Vorgänge dazwischen (Einschwingen) irrelevant



Es wird stets nur mit eingeschwungenen Zuständen gearbeitet

© A. Steininger / TU Wien

Vorteile synchroner Logik

- Einfachheit und Produktivität
  - ✓ Arbeiten auf hoher Abstraktionsebene
  - ✓ Wahrheitstabellen mit "vorigem" Zustand
  - ✓ Einschwingvorgänge irrelevant
    => betrachtete Zustände stets klar definiert
  - ✓ Timing-Analysen erst nachträglich
  - ✓ klare Trennung zwischen Takt und Daten vereinfacht die Timinganalyse

© A. Steininger / TU Wien

Weitere Vorteile

- Effizienz:
  - ✓ nur ein zusätzliches Steuersignal für das gesamte Design
  - ✓ periodischer Takt ist einfach generierbar
  - ✓ Daten können single-rail codiert sein
  - ✓ minimale Anzahl von Flanken auf den Datenleitungen
  - ✓ Takt liefert implizit auch einen Zeitbegriff

© A. Steininger / TU Wien

Alle Probleme gelöst ?

Ermittlung der zulässigen Taktfrequenz erfordert Kenntnis aller Laufzeiten

Kennen wir die Laufzeiten wirklich ?

Im Grunde ist der Schluss von einer Zeitbedingung auf Daten-konsistenz und indirekt und jede Zeitbedingung willkürlich



© A. Steininger / TU Wien

Einflußfaktoren auf Delay

- Anzahl der durchlaufenen Gatter
  - Operation
  - Optimierung & Mapping
- Routing
  - Geometrie (Länge der Verbindungen)
  - bzw. Switches
- Exemplarstreuung & Betriebsbedingungen
  - Prozessvariationen
  - Temperatur
  - Versorgungsspannung

Implementierung & Technologie festgelegt

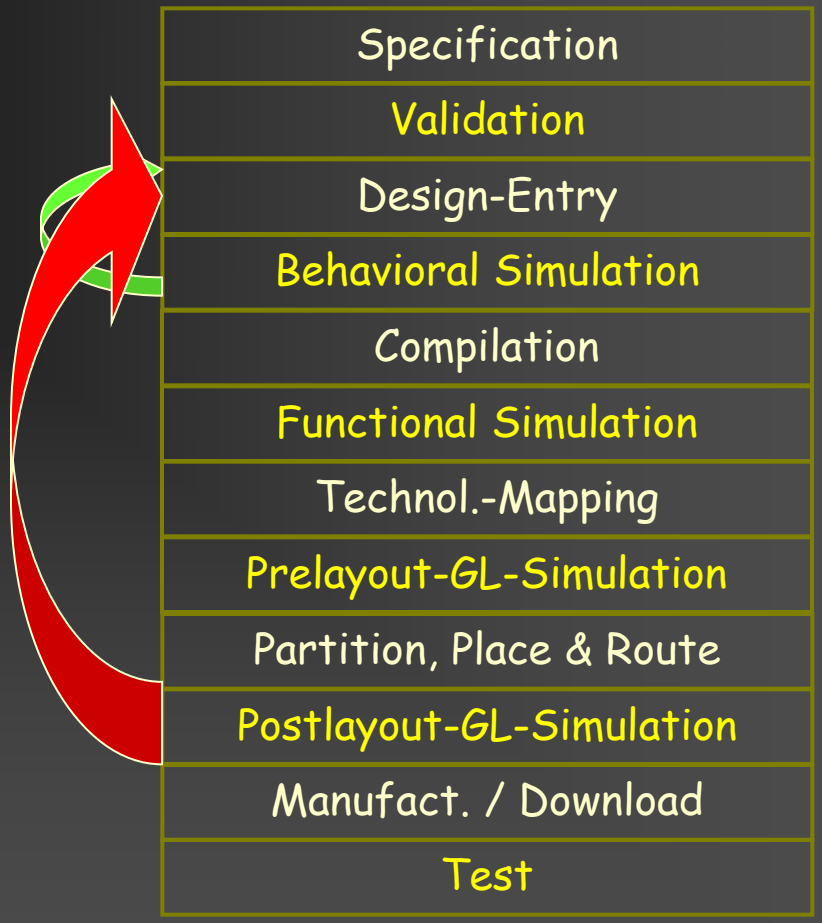
Routing fertig & Technologie festgelegt

unbekannt => worst case Annahmen

© A. Steininger / TU Wien

Problem Timinganalyse

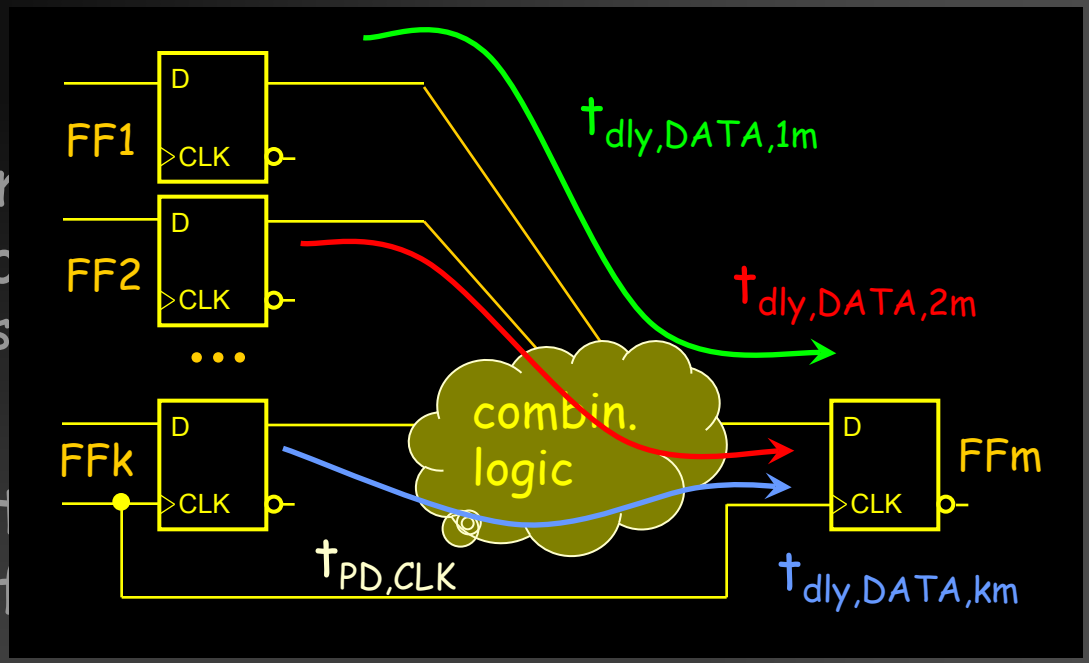
- erfolgt erst am Ende des Design-Flow (große Iterationsschleife bei Adaptionen)
- erfordert mittlerweile gewaltigen Aufwand



© A. Steininger / TU Wien

Problem Timinganalyse

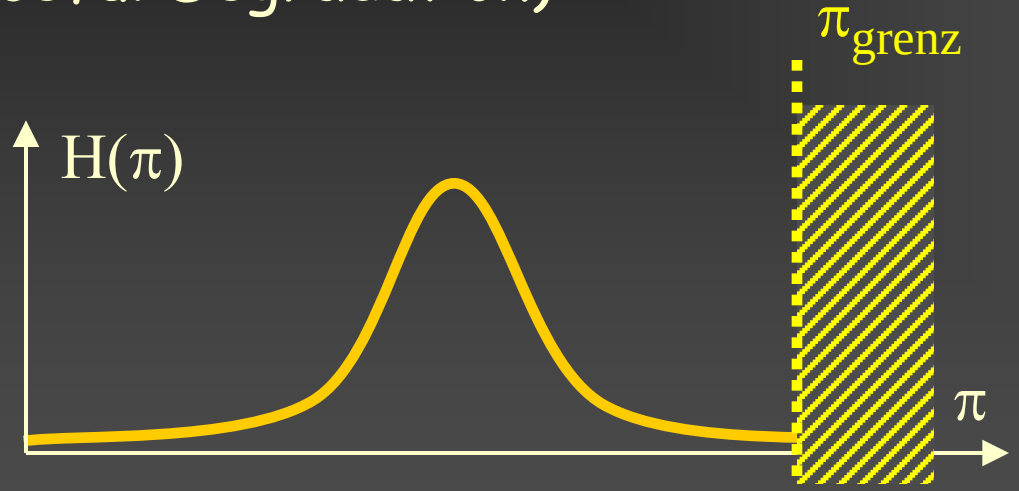
- ✓ erfolgt erst ar
  des Design-Flo
  (große Iterations
  bei Adaptionen)
- ✓ erfordert mitt
  gewaltigen Au
- ▲ nur mit „idealem“ Taktnetz praktisch machbar



© A. Steininger / TU Wien

Problem Worst-Case Annahme

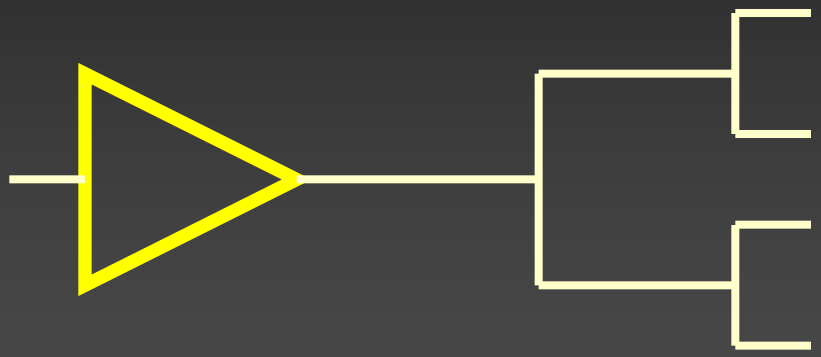
- sind im Normalfall zu pessimistisch
  - (Chip könnte schneller sein)
- keine Robustheit bezüglich Überschreitung
  - (Graceful Degradation)



© A. Steininger / TU Wien

Problem Taktverteilung

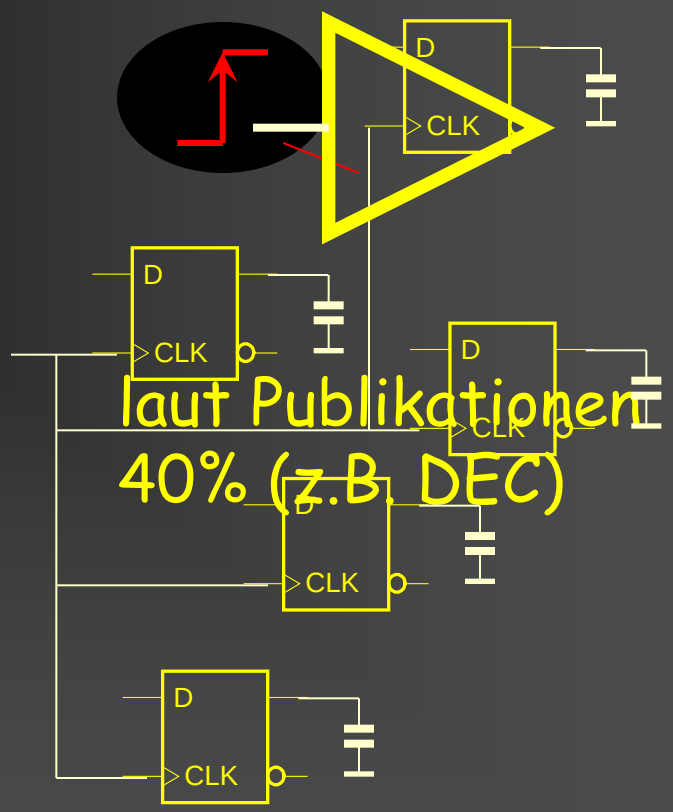
- Taktverteilung
  - weit verzweigte Taktnetze im Chip
  - Minimierung von Delay & Skew aufwendig



© A. Steininger / TU Wien

Problem Verlustleistung

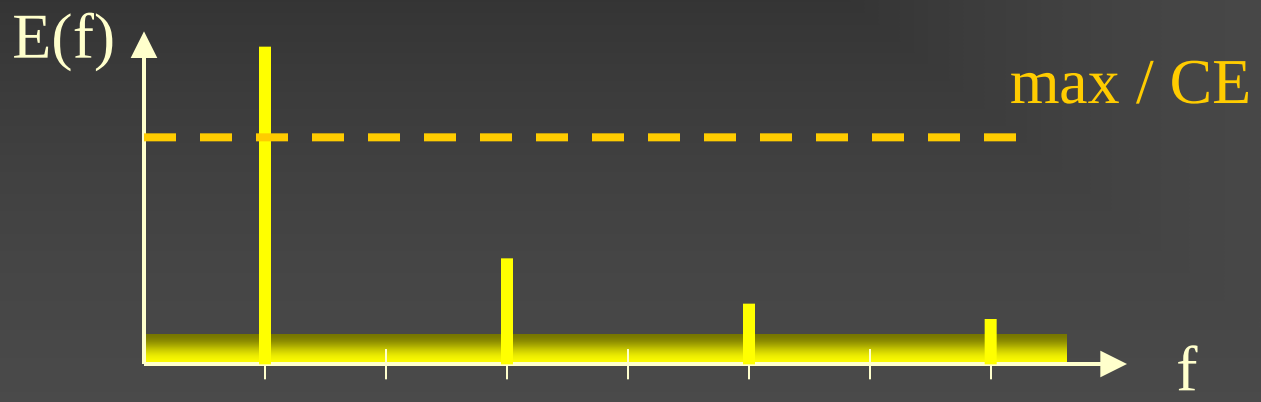
- hoher Leistungs-verbrauch im Taktnetz
- gleichzeitiges Schalten
  => Stromspitzen
  => Probleme mit Spannungsversorgung
- ständiges Schalten
  => viel unnötige Aktivität



© A. Steininger / TU Wien

Problem EMV

- Die langen Taktnetze sind gute Antennen.
- Fast die gesamte Energie der Störabstrahlung ist auf eine einzige spektrale Komponente konzentriert.



© A. Steininger / TU Wien

Problem Erweiterbarkeit

- Jede kleine Änderung der Schaltung erfordert neuerlich eine vollständige Timing-Analyse.
- Jeder Umstieg auf eine andere Technologie verändert das Timing komplett.
- Zusammenspiel von IP-Cores auf einem Chip stellt hohen Anforderungen an Spec

© A. Steininger / TU Wien

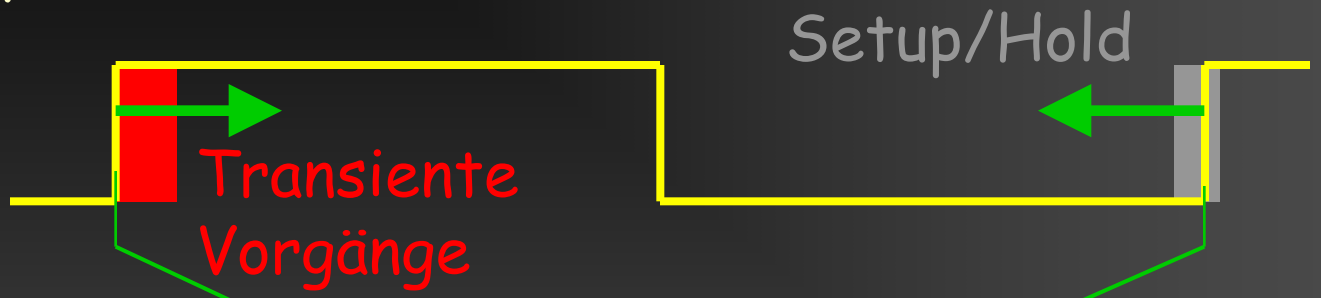
Grenzen synchroner Logik

- Die Voraussetzungen für synchrone Logik werden zunehmend unrealistisch
  - Die extremen Taktfrequenzen lassen kaum mehr Spielraum für Timing-Margins.
  - Jede sinnvolle Schaltung hat Schnittstellen zur Umwelt. Diese sind zwangsläufig asynchron und führen daher immer zu Setup/Hold-Violations.
  - In sehr vielen Designs gibt es mehrere Clock-Domains. An den Schnittstellen dazwischen kommt es ebenfalls zu Setup/Hold-Violations.

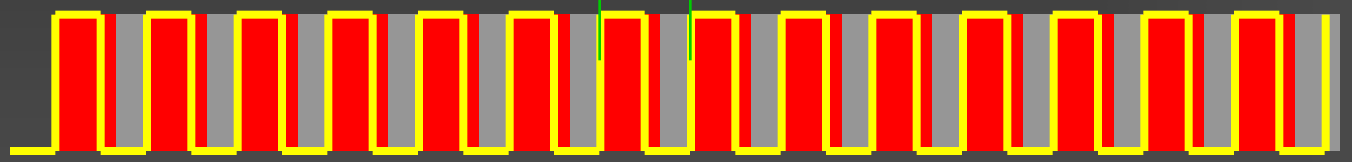
© A. Steininger / TU Wien

Schwindende Timing-Margins

Synchrones Modell:



Realität:



© A. Steininger / TU Wien

Synchrones Design am Limit ?

- Die Idealvorstellung beim synchronen Design war es, Einschwingvorgänge nicht beachten zu müssen.
- Tatsächlich besteht bei High-End-Designs ein überwiegender Anteil am Aufwand darin, die Einschwingvorgänge so zu optimieren und alle Reserven so weit auszureizen, dass sich maximale Taktfrequenz erreichen lässt.
- Die Komplexität dieser Optimierungsaufgabe übersteigt die Fähigkeit der verfügbaren Tools bei weitem. Angesichts der rasant wachsenden Taktraten ist anzunehmen, dass sich diese Situation in Zukunft noch verschlechtert.

© A. Steininger / TU Wien

Grenzen synchroner Logik

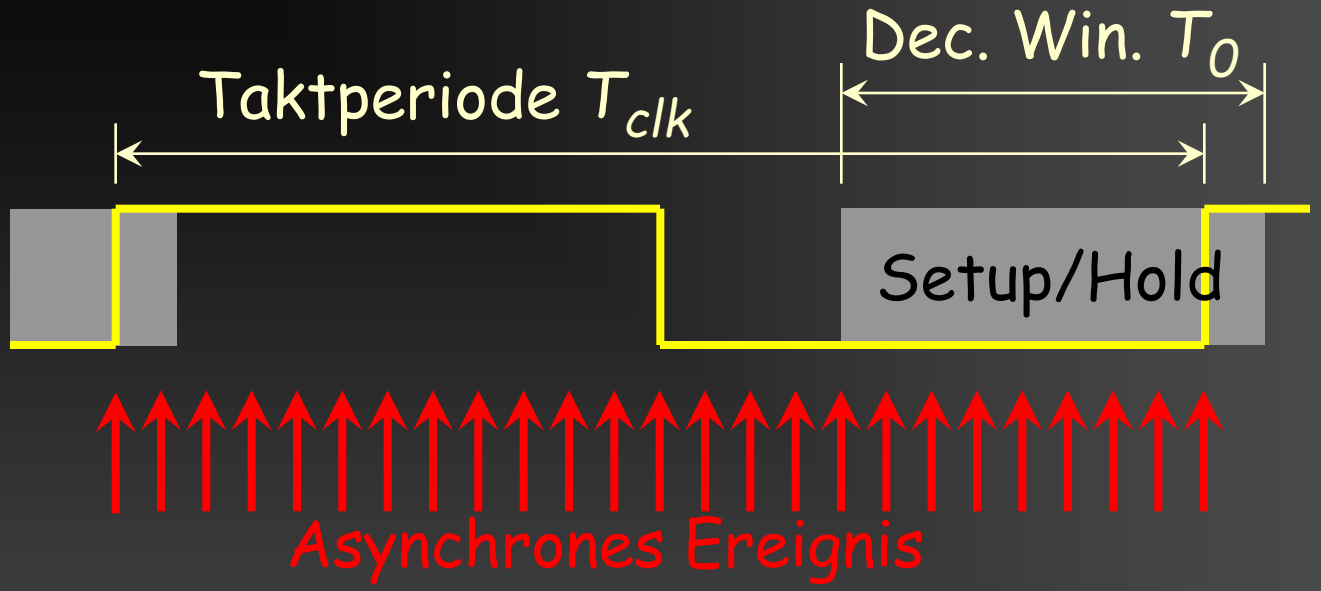
Die Voraussetzungen für synchrone Logik werden zunehmend unrealistisch

- ✓ Die extremen Taktfrequenzen lassen kaum mehr Spielraum für Timing-Margins. Ein worst-case-Design ist im Einzelfall unnötig langsam.
- Jedes sinnvolle System hat Schnittstellen zur Umwelt. Diese sind zwangsläufig asynchron und führen daher immer zu Setup/Hold-Violations.
- In sehr vielen Designs gibt es mehrere Clock-Domains. An den Schnittstellen dazwischen kommt es ebenfalls zu Setup/Hold-Violations.

© A. Steininger / TU Wien

Asynchrone Eingänge





Wahrscheinlichkeit für Setup/Hold-Violation:

$$P_{violate} = \frac{T_0}{T_{clk}} > 0$$

© A. Steininger / TU Wien

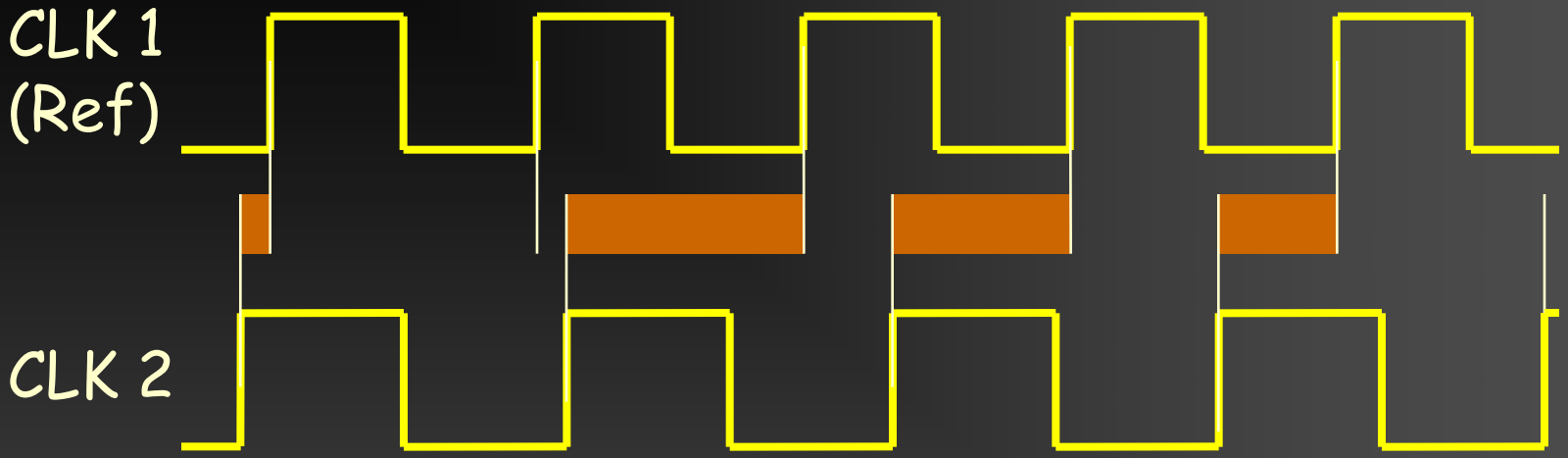
Grenzen synchroner Logik (3)

Die Voraussetzungen für synchrone Logik werden zunehmend unrealistisch

- ✓ Die extremen Taktfrequenzen lassen kaum mehr Spielraum für Timing-Margins. Ein worst-case-Design ist im Einzelfall unnötig langsam.
- ✓ Jede sinnvolle Schaltung hat Schnittstellen zur Umwelt. Diese sind zwangsläufig asynchron und führen daher immer zu Setup/Hold-Violations.
- In sehr vielen Designs gibt es mehrere Clock-Domains. An den Schnittstellen dazwischen kommt es ebenfalls zu Setup/Hold-Violations.

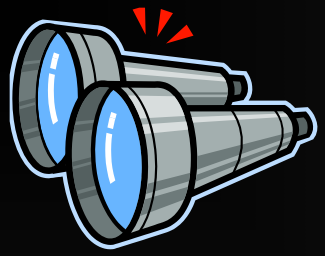
© A. Steininger / TU Wien

Mehrere Clock-Domänen



Die aktiven Flanken der beiden Takte haben beliebige Phasenlage zueinander. Es kommt daher zwangsläufig zu Setup/Hold-Violations.
Eine ähnliche Situation ergibt sich auch bei nur einem Takt mit großem (unbestimmtem) Skew.

© A. Steininger / TU Wien



Überblick

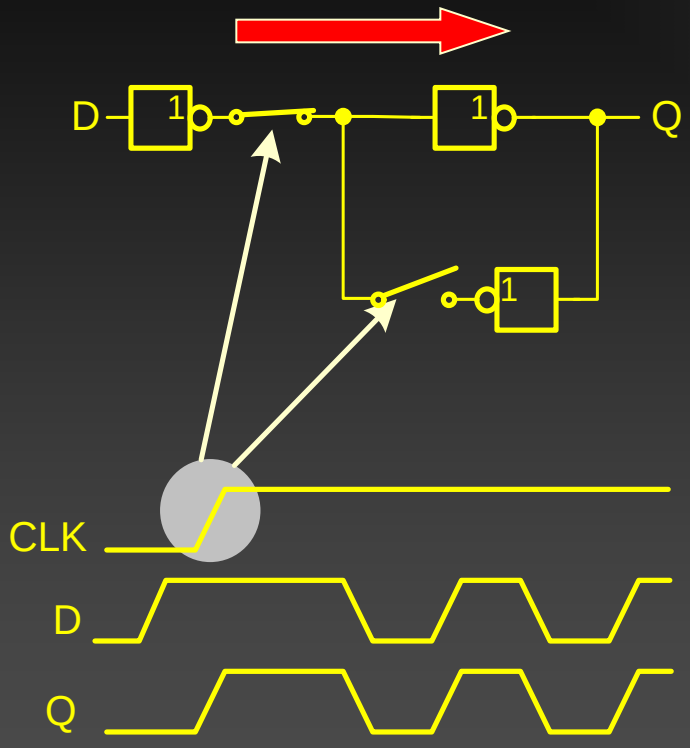
- Boolesche Logik
- Delay & Skew
- Grundproblem des Logikdesign
- Synchrones Design
  - Lösungsprinzip, Vorteile
  - konzeptioneller Mangel
  - Grenzen
- Metastabilität

© A. Steininger / TU Wien

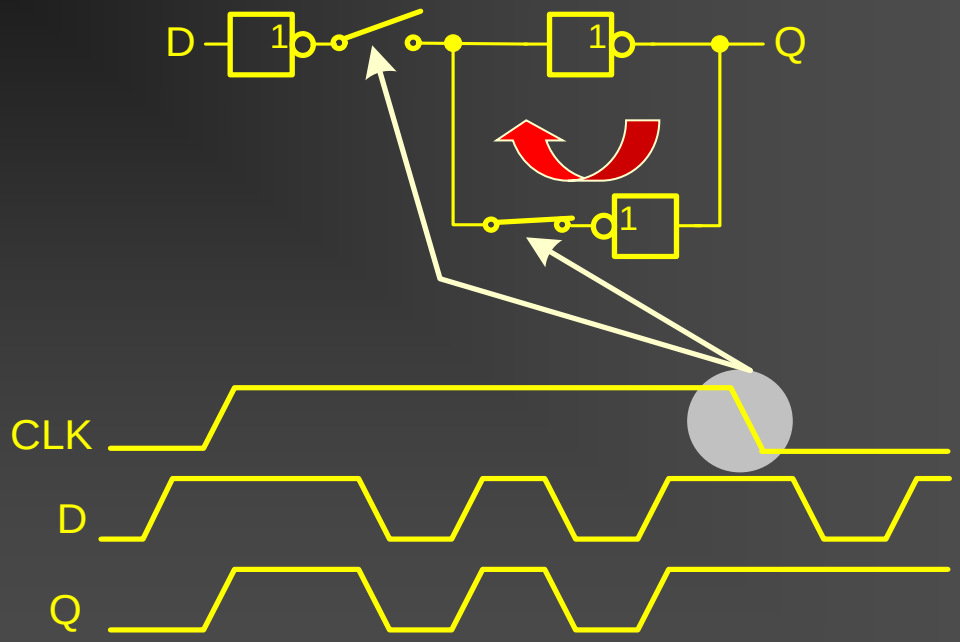
Wiederholung

Funktionsmodell eines Latch

transparent



hold



© A. Steininger / TU Wien

Setup- und Hold-Time

- Eine Änderung am Eingang muss die Rückkopplungsschleife vollständig durchlaufen
  UND
- Die TGs müssen umgeschalten werden

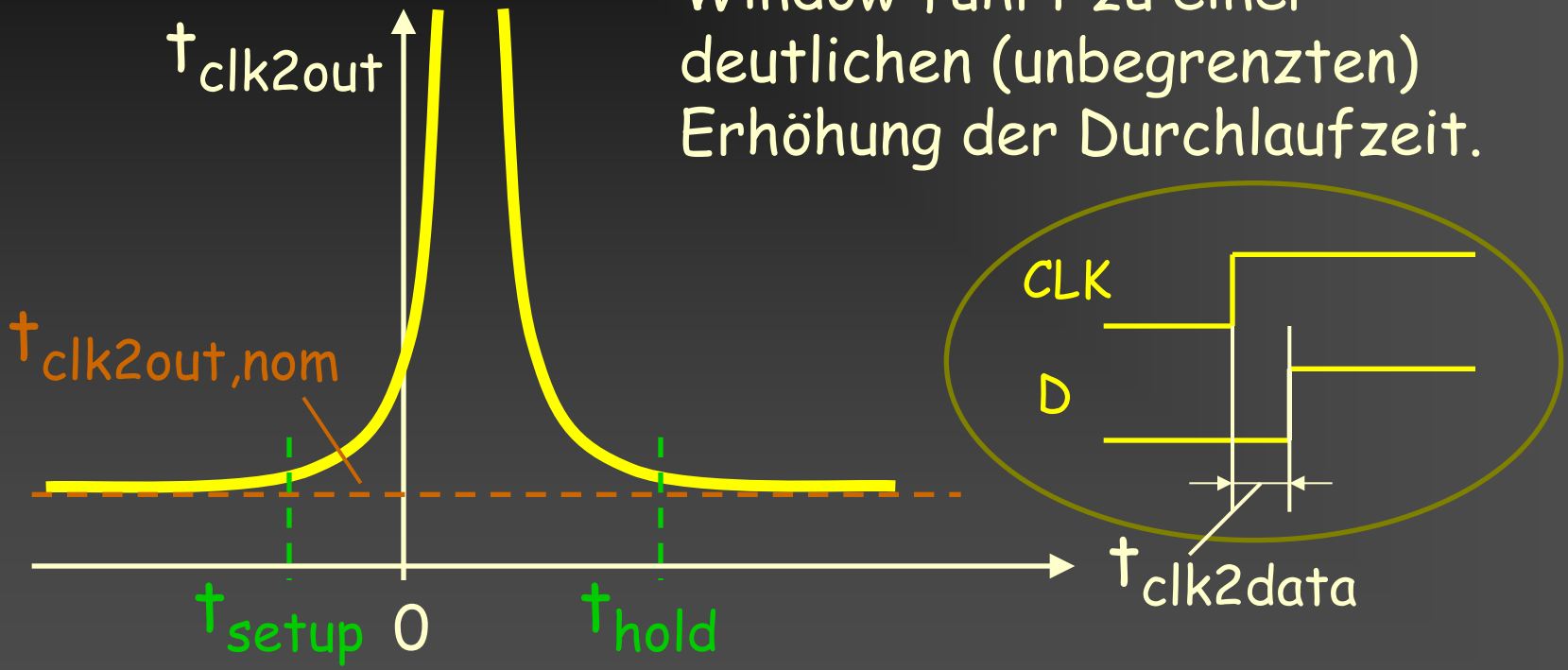
Diese Vorgänge brauchen **Zeit:**
„Decision Window" (= SetupTime + Hold-Time)

Innerhalb dieses „Decision Window" dürfen keine Flanken am Eingang auftreten (Metastabilität!)

© A. Steininger / TU Wien

Response-Time eines Flip-Flop

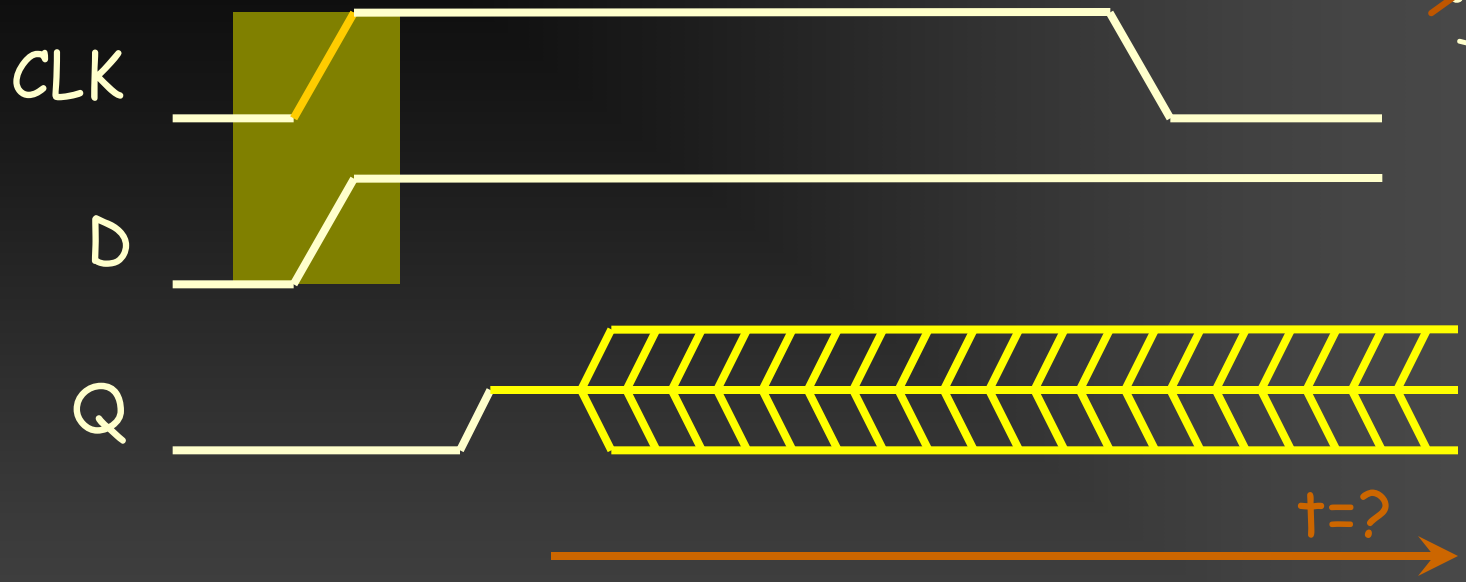
Eine Änderung des Eingangs während des Decision-Window führt zu einer deutlichen (unbegrenzten) Erhöhung der Durchlaufzeit.



© A. Steininger / TU Wien

Metastabilität: Zeitverhalten

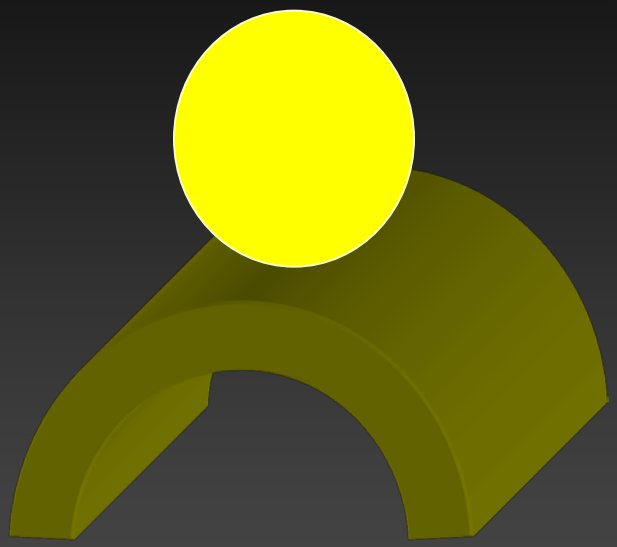




Der Ausgang des DFF bleibt für unbestimmte Zeit in einem undefinierten *„metastabilen"* Zustand und geht dann zufällig auf ´1´ oder ´0´.

© A. Steininger / TU Wien

Metastabilität: phys. Vergleich



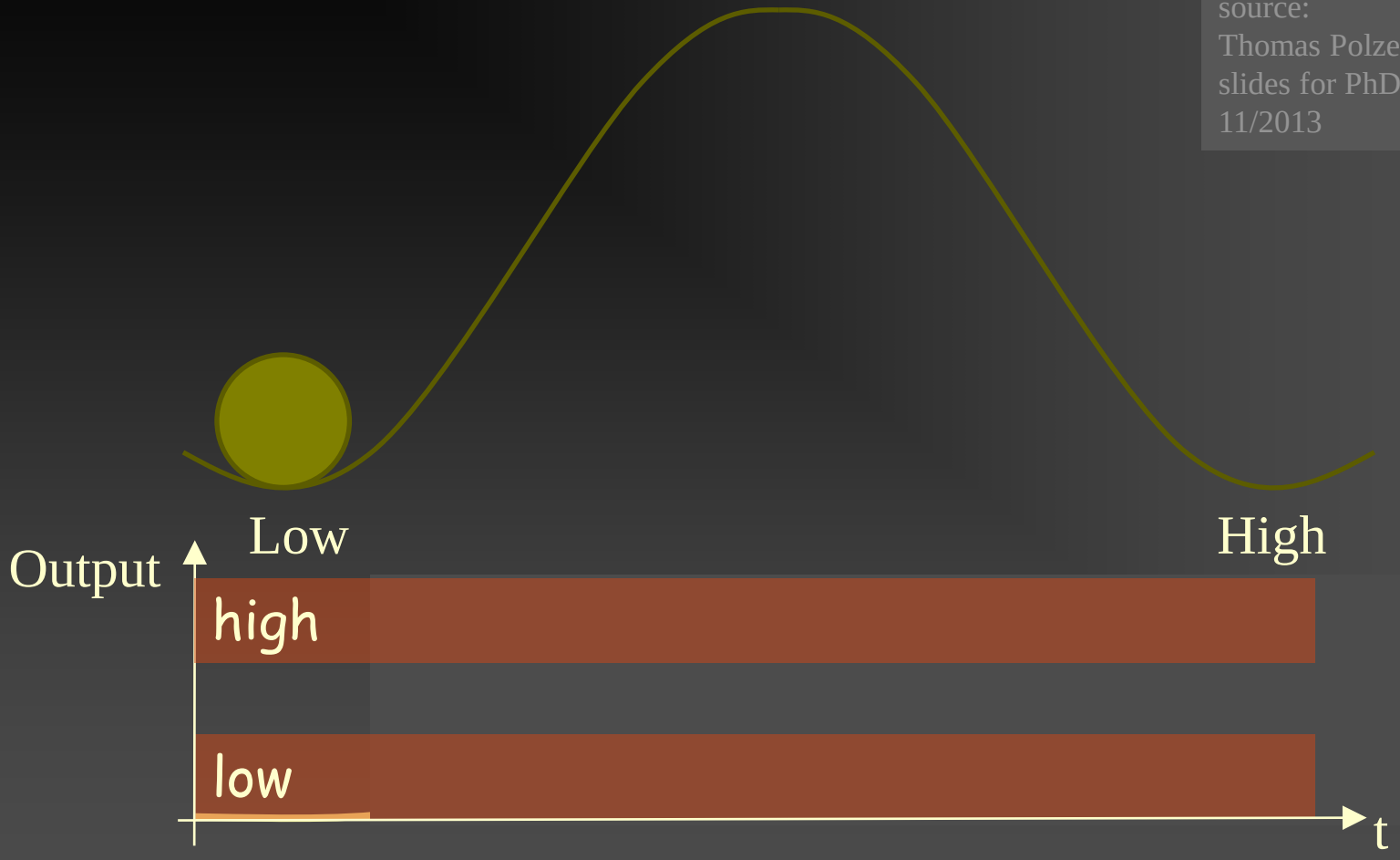
- Die Kugel bleibt für unbestimmte Zeit im labilen Gleichgewicht.
- Die kleinste Störung lenkt die Kugel zufällig in eine Richtung

Wieviel Schwung befördert die Kugel von einer Seite auf die andere? Was passiert bei zu wenig Schwung?

© A. Steininger / TU Wien

Zu kleiner Impuls

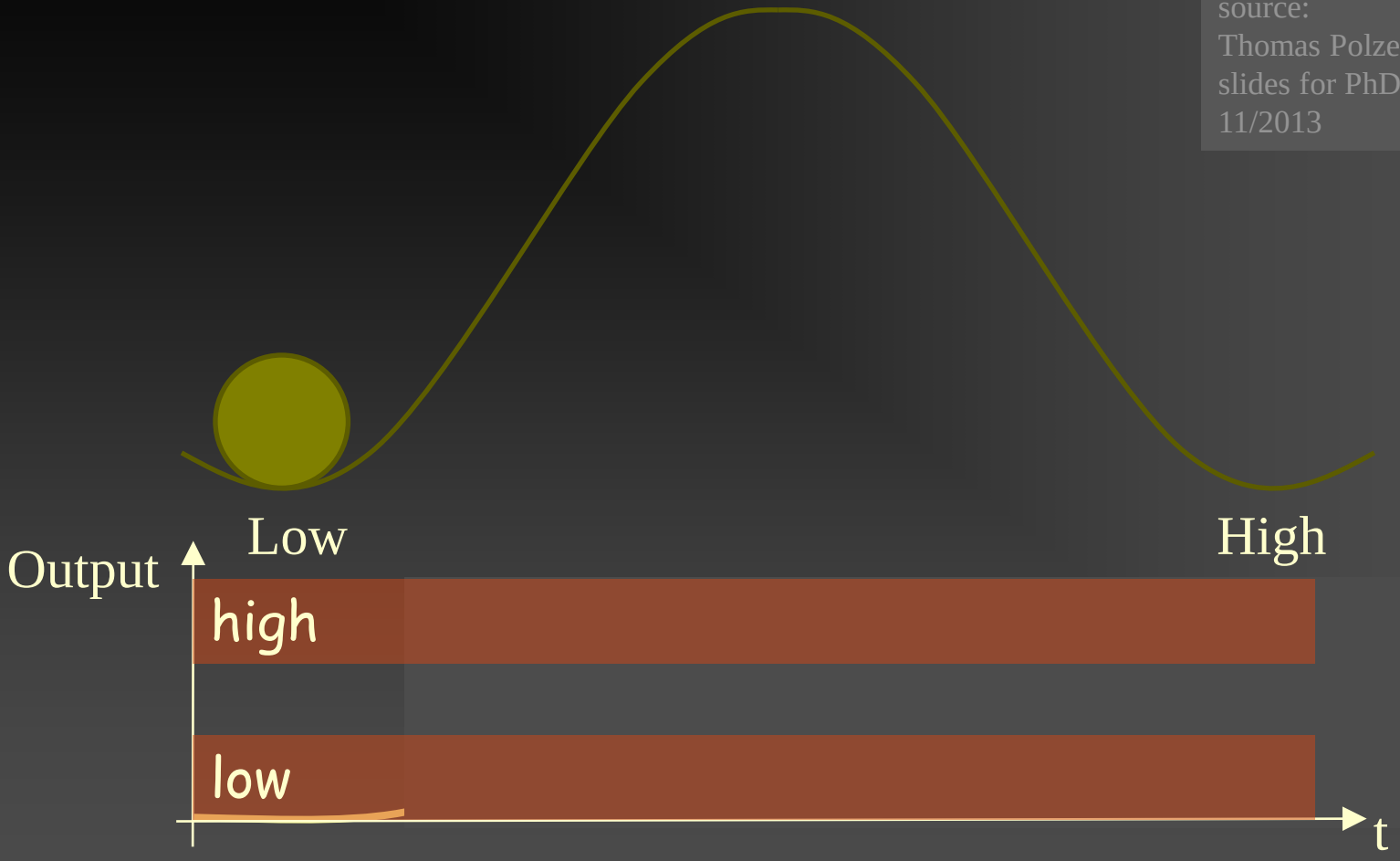
source:
Thomas Polzer
slides for PhD defense,
11/2013



© A. Steininger / TU Wien

Genügend großer Impuls

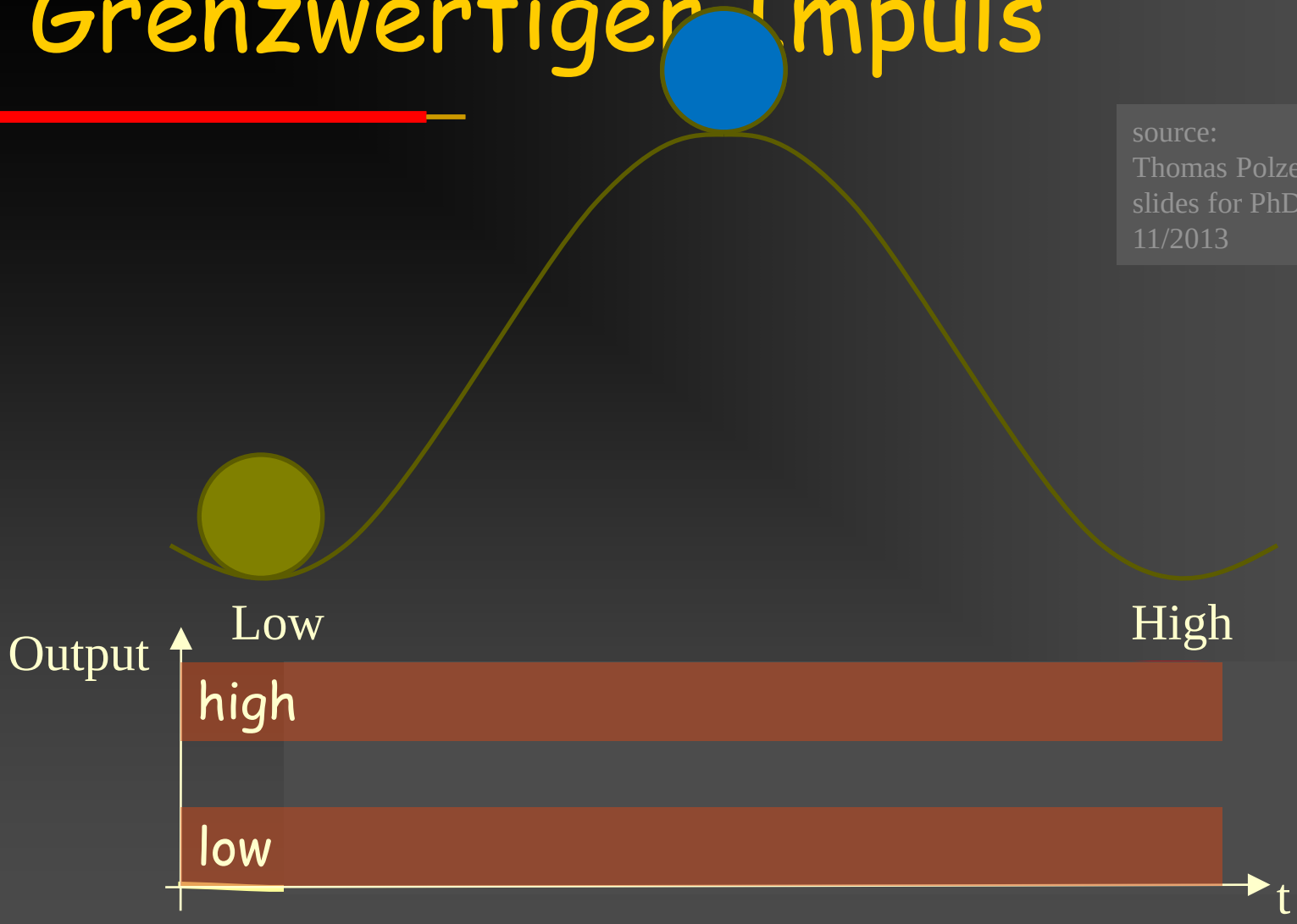
source:
Thomas Polzer
slides for PhD defense,
11/2013



© A. Steininger / TU Wien

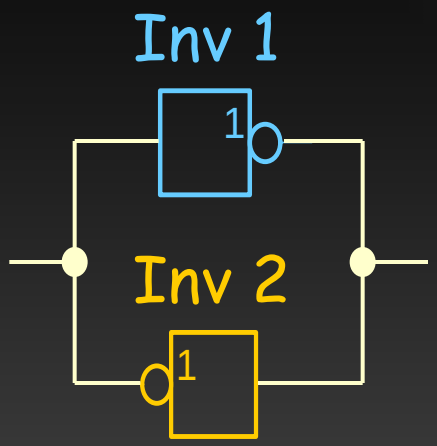
Grenzwertiger Impuls

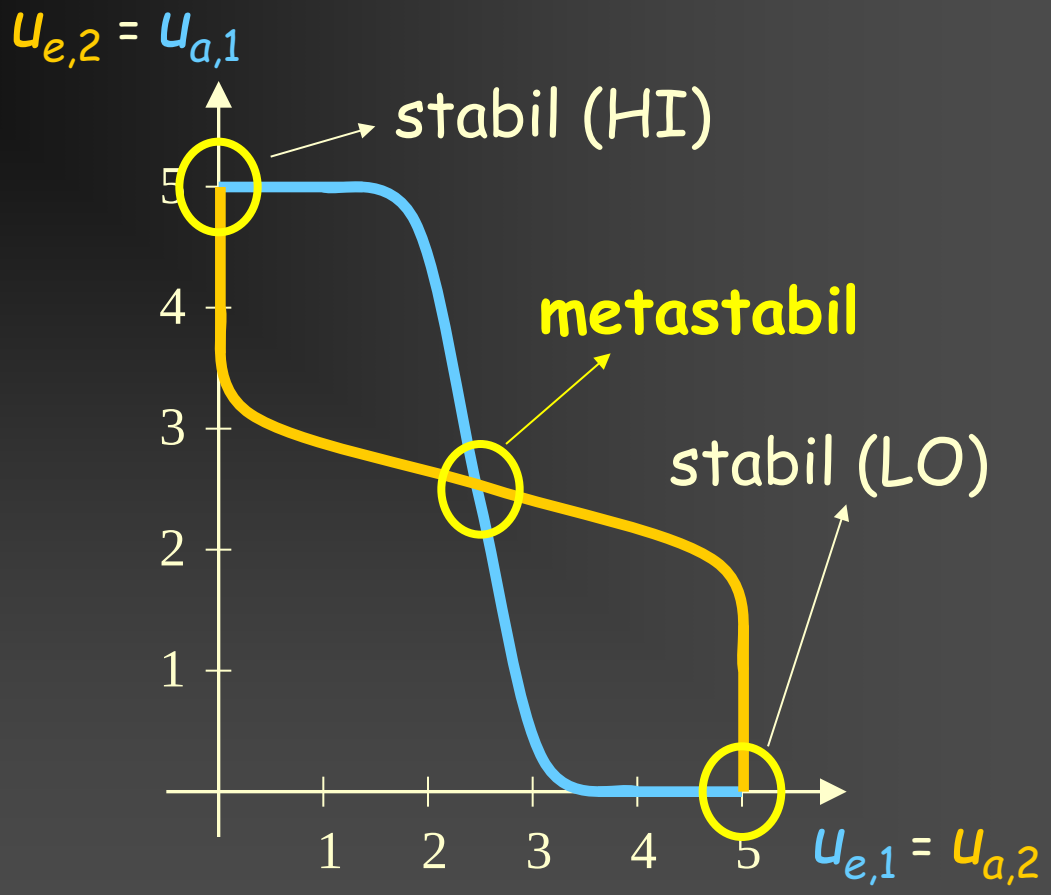
source:
Thomas Polzer
slides for PhD defense,
11/2013



© A. Steininger / TU Wien

Metastabilität: Kennlinie





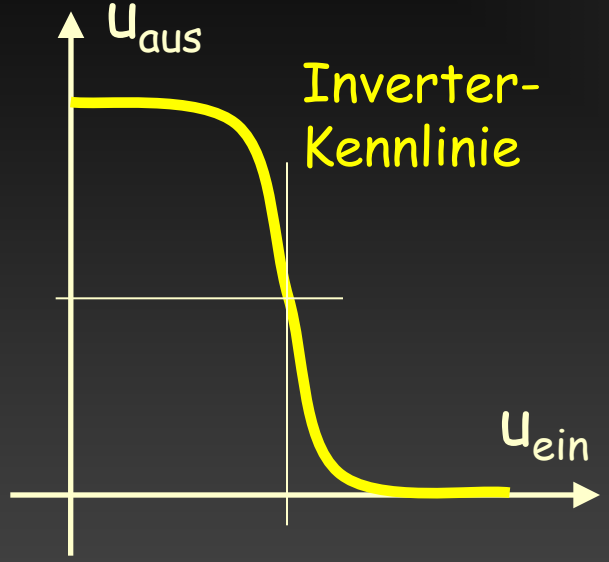
© A. Steininger / TU Wien

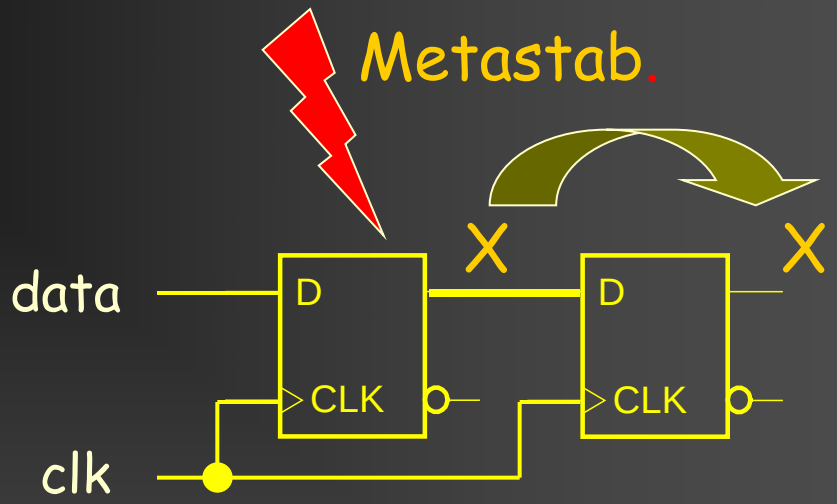
Folgen der Metastablität

- Es ist unerheblich, in welchen Zustand das FF letztendlich kippt
- Problematisch ist der Zeitraum der Meta-stabilität, denn der "verbotene" Pegel
  - kann von einem nachfolgenden FF übernommen werden und dieses ebenfalls in die Metastabilität treiben => Fortpflanzung
  - kann von mehreren nachfolgenden FFs unterschiedlich interpretiert werden und daher zu Inkonsistenzen führen

© A. Steininger / TU Wien

Fortpflanzung d. Metastability

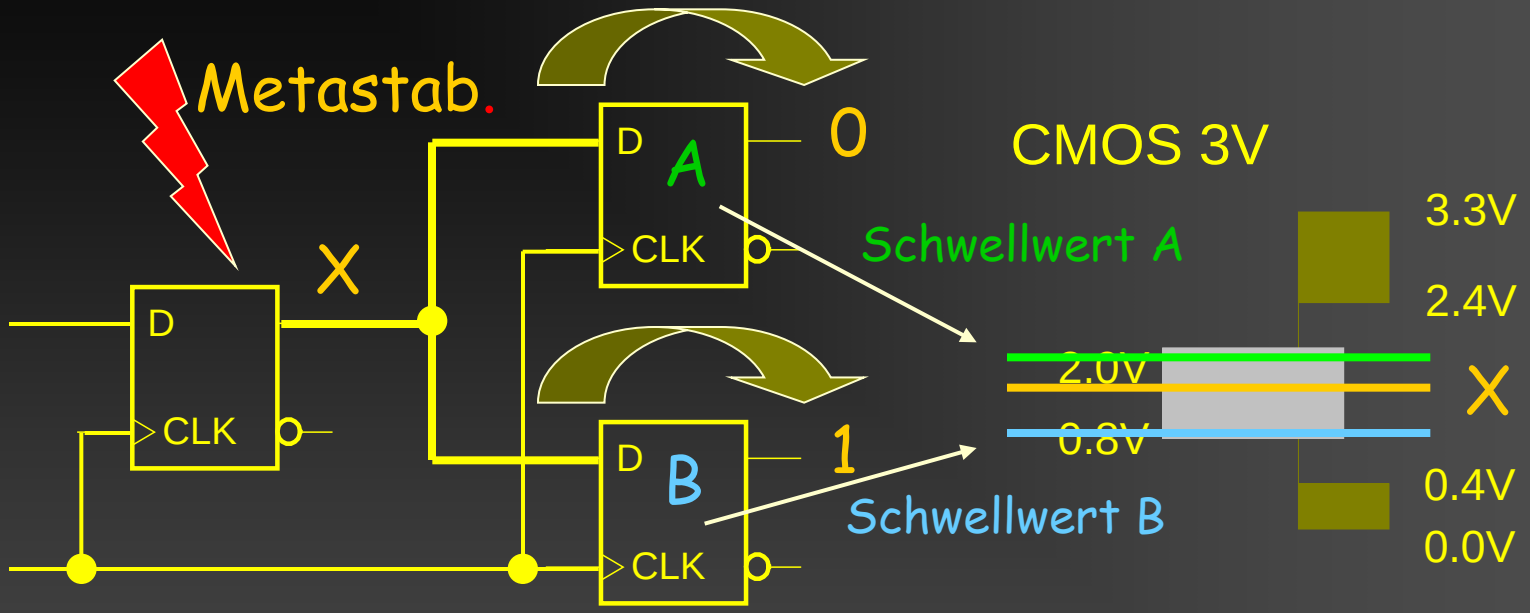




Die Inverter-Kennlinie beschreibt ein analoges Übertragungsverhalten, ein metastabiler Zustand kann daher eventuell auch übertragen werden.

© A. Steininger / TU Wien

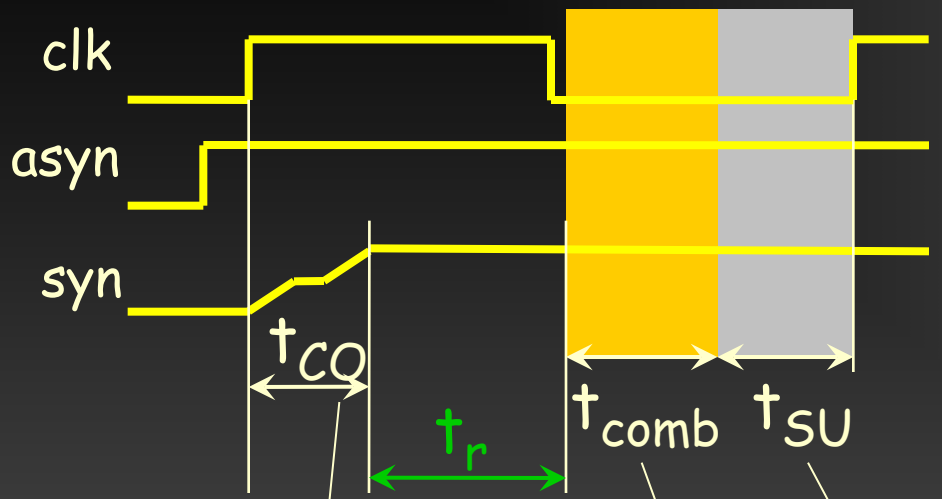
Inkonsistente Wahrnehmung

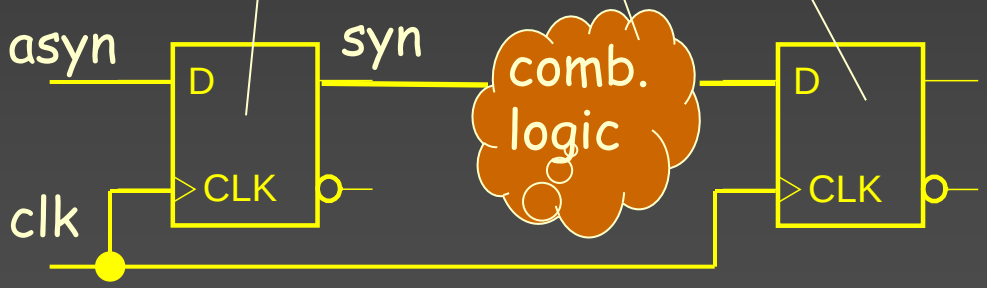


Der metastabile Zustand wird von einem FF als "1" wiedergegeben, von einem anderen als "0"

© A. Steininger / TU Wien

Was ist die Resolution Time?





$$t_r = T_{clk} - t_{CO} - t_{comb} - t_{SU}$$

Resolution time t_r ist jene Zeit die für das Auflösen der Meta-stabilität maximal zur Verfügung steht

Upset:

Auflösen der Meta-stabilität braucht länger als t_r

© A. Steininger / TU Wien

Mean Time Between Upset



- „Upset“
  - metastabiler Ausgang wird nach t_r von nachfolgenden FF am Eingang übernommen
- Mean Time Between Upset (MTBU)
  - Erwartungswert für das Zeitintervall zwischen zwei Upsets (statistische Größe !)

$$MTBU = \exp\left(\frac{t_r}{\tau_c}\right) \cdot \frac{1}{T_0 \cdot f_{clk} \cdot \lambda_{dat}}$$

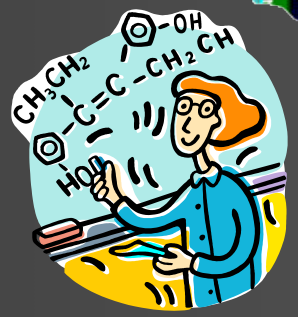
© A. Steininger / TU Wien

Parameter in der Formel

- Resolution Time t_r
  - Jene Zeit die dem Ausgang zum Einschwingen zur Verfügung steht
- Bauteilparameter des FF τ_c , T_0
  - werden experimentell bestimmt
  - Zeitkonstante τ_c abh. v. Transitfrequenz
  - T_0 aus effektiver Breite des decision window
- Taktperiode des FF $T_{clk} = 1/f_{clk}$
- Average Rate of Change λ_{dat}
  - mittlere Rate der Flanken am FF-Eingang

© A. Steininger / TU Wien

Erläuterung der Formel



Upset-Rate

Häufigkeit der Flanken am Eingang

$$\frac{1}{MTBU} = R_{upset} = \exp\left(\frac{-t_r}{\tau_c}\right) \cdot \frac{T_0}{T_{clk}} \cdot \lambda_{dat}$$

Wahrscheinlichkeit, dass metastabiler Zustand während t_r nicht aufgelöst wird

Wahrscheinlichkeit dass Flanke im „decision window" erfolgt

© A. Steininger / TU Wien

Zahlenbeispiel MTBU

FF mit 100MHz Takt und 500kHz Eingangssignal.

t_{comb} = 3ns, t_{SU} = 0.5ns,
τ_c = 0.25ns, T_0 = 0.1ns
t_{CO} = 1.5ns

$$t_r = \frac{1}{f_{clk}} - t_{CO} - t_{comb} - t_{SU} = 5ns$$

$$\lambda_{dat} = 2 f_{dat} = 1MHz$$

$$MTBU = \exp\left(\frac{t_r}{\tau_c}\right) \cdot \frac{1}{T_0 \cdot f_{clk} \cdot \lambda_{dat}} =$$

$$= e^{20} \cdot 10^{10} \cdot 10^{-8} \cdot 10^{-6} s = 4.8 \cdot 10^4 s$$

- ein Upset alle 13 Stunden.
- Bei 16 Kanälen also mehr als ein Upset stündlich.
- Bei 10000 Produkten/1Kanal im Schnitt alle 5s !

© A. Steininger / TU Wien

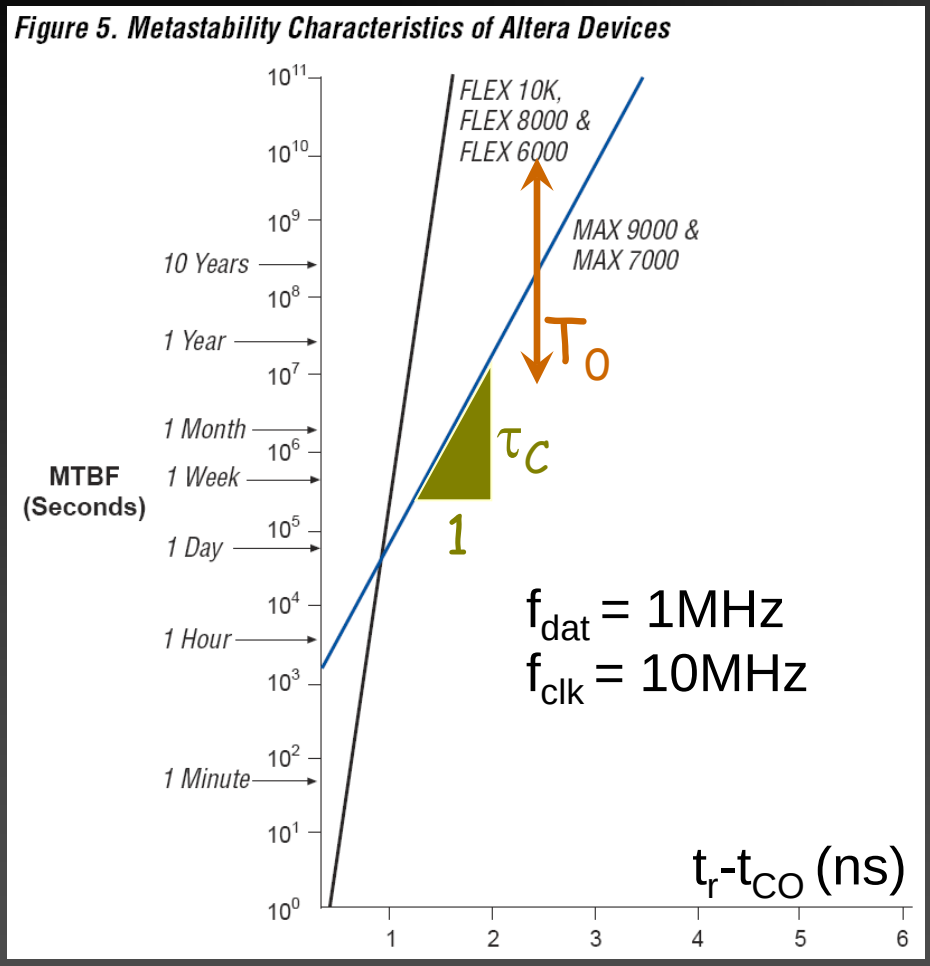
Erhöhen der MTBU im Design

Metastabilität prinzipiell nicht vermeidbar, aber MTBU lässt sich hinreichend erhöhen:

- λ_{dat} und f_{clk} möglichst niedrig halten
  - meist jedoch durch Applikation vorgegeben
- T_0 und τ_c möglichst klein halten
  - durch Technologie, Speed Grade bestimmt
- t_r möglichst gross halten
  - hat extrem starken Einfluß
  - Grundidee für diverse *Synchronizer*

© A. Steininger / TU Wien

Technologie und MTBU



Je nach Zieltechnologie sind die Parameter unterschiedlich. Das hat gewaltige Auswirkungen auf die MTBU.

© A. Steininger / TU Wien

Resolution Time erhöhen (1)

Taktfrequenz vermindern

80MHz

FF mit ~~100MHz~~ Takt und 500kHz Eingangssignal.

t_{comb} = 3ns, t_{SU} = 0.5ns,

τ_c = 0.25ns, T_0 = 0.1ns

t_{CO} = 1.5ns

7.5ns

$$t_r = \frac{1}{f_{clk}} - t_{CO} - t_{comb} - t_{SU} = 5ns$$

$$\lambda_{dat} = 2 f_{dat} = 1MHz$$

$$MTBU = \exp\left(\frac{t_r}{\tau_c}\right) \cdot \frac{1}{T_0 \cdot f_{clk} \cdot \lambda_{dat}} =$$

$$= e^{20} \cdot 10^{10} \cdot 10^{-8} \cdot 10^{-6} s = 4.8 \cdot 10^{4} s$$

30 $1.25 \cdot 10^{-8} s$

$1.3.10^9$ s

- ein Upset alle ~~13 Stunden~~. 42 Jahre

© A. Steininger / TU Wien

Resolution Time erhöhen (2)

Verminderung des Taktes selten möglich.
Alternativen:

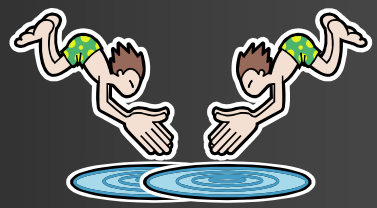
- ✓ Schnelleres FF => verringerte t_{SU}
- 2 oder mehr Taktperioden für Synchronisation

Voriges Beispiel (bei 100MHz):

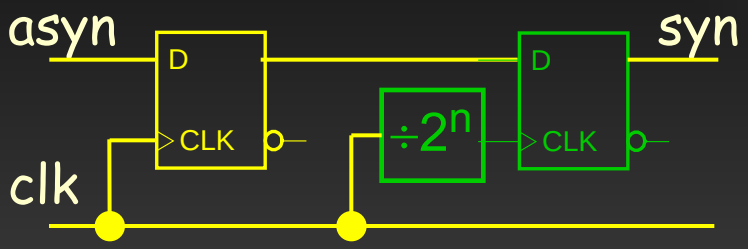
- 1 Takt länger warten ergibt t_r = 15ns
- => MTBU steigt um 17 Größenordnungen (!)
  von 13 Stunden auf $724*10^{12}$ Jahre

© A. Steininger / TU Wien

Synchronizer

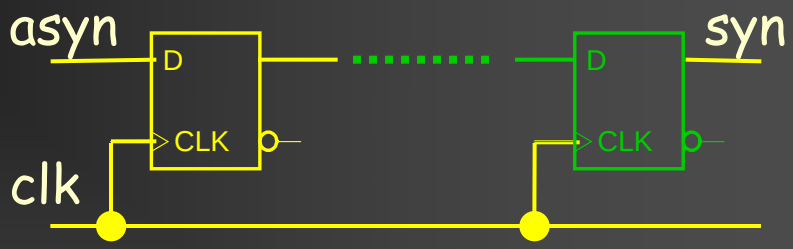


- Zweites Eingangs-FF mit geteiltem Takt



- Beste MTBU erzielbar (t_{CO} und t_{SU} nur 1x)
- Skew problematisch

- Kaskade von *n* Eingangs-FFs

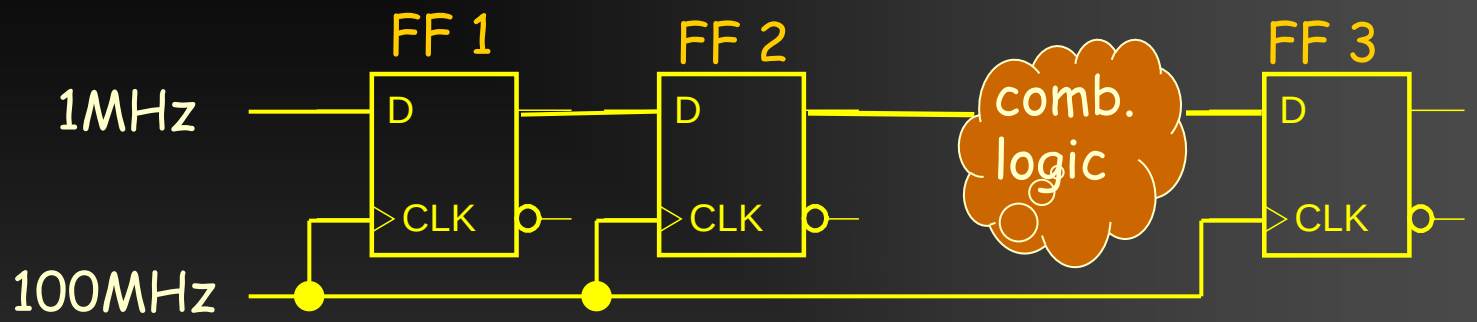


- MTBU etwas schlechter (t_{CO} und t_{SU} je Stufe)
- einfach & problemlos

In jedem Fall wird das Eingangssignal um *n* Taktzyklen verzögert weitergegeben!

© A. Steininger / TU Wien

Beispiel 2-Flop Synchronizer



t_{comb} = 3ns, t_{SU} = 0.5ns, τ_c = 0.25ns, T_0 = 0.1ns , t_{CO} = 1.5ns

Schaltung A aus FF2 + FF3 wie zuvor, aber durch Vorschalten von FF1 erhöht sich die effektive Resolution Time:

$$t_{res,ges} = t_{res,FF2} + t_{res,FF1}$$

$t_{res,ges}$ = 8ns+5ns, $MTBU_{ges} = 1{,}2*10^{11}$ Jahre

© A. Steininger / TU Wien

Ist Metastabilität vermeidbar?

- In konventioneller synchroner Architektur
  - NEIN, das Auftreten von Upsets läßt sich nur hinreichend unwahrscheinlich machen
- In adaptierten synchronen Architekturen
  - JA, man benötigt einen abschaltbaren Takt
- In asynchronen Architekturen:
  - JA, sie verwenden Handshaking anstatt eines starren Taktes

© A. Steininger / TU Wien

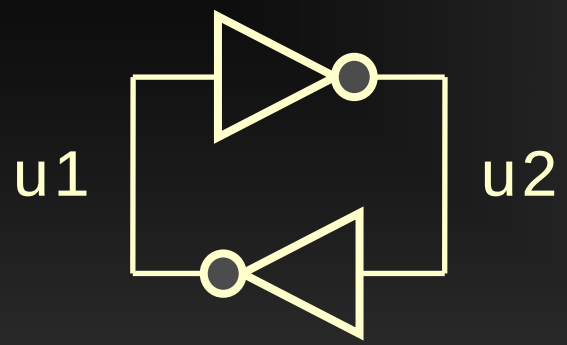
Modell für Metastability

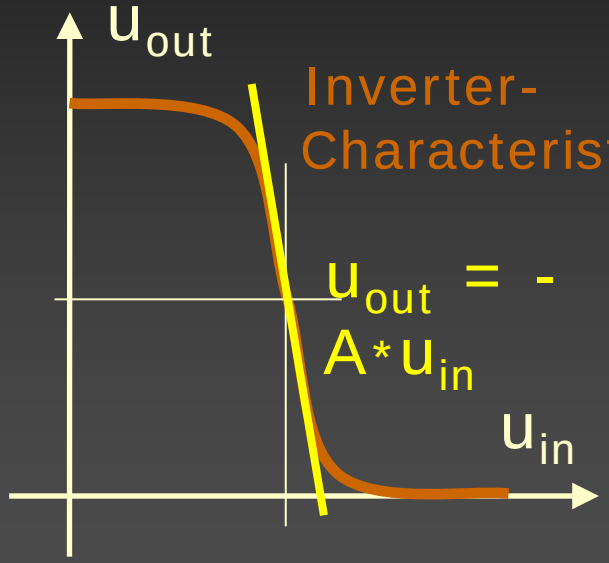
- Wie kommt man zu der MTBU-Formel?
- Welches Modell benötigt man?



© A. Steininger / TU Wien

Einfaches Modell

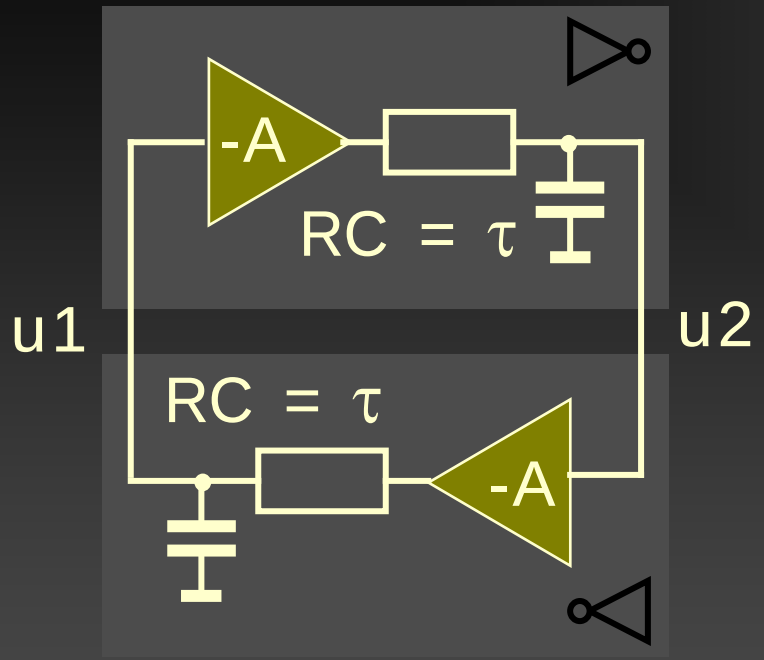




- modelliere bistabile elemente als Paar von Invertern
- lineares Modell für Inverter, um den Mittelpunkt der Transferfunktion („balance point")
- Betrachte „homogenen" Fall, d.h. geschlossene Schleife

© A. Steininger / TU Wien

Berücksichtigung der Dynamik



- Näherung mit dynamischem Modell 1.Ordnung: RC Glied
- Annahme von Symmetrie (gleiches A, RC für beide Inverter) zur Vereinfachung
- Annahme symmetrischer Versorgung (+VCC/-VCC) gegen GND

© A. Steininger / TU Wien

Differentialgleichungen

- Vorwärtspfad:

$$u_2 = -A \cdot u_1 - R \cdot C \cdot \frac{du_2}{dt}$$

$$u_R = R \cdot i_R$$

$$i_C = C \cdot \frac{du_C}{dt}$$

- Rückwärtspfad:

$$u_1 = -A \cdot u_2 - R \cdot C \cdot \frac{du_1}{dt}$$

- Laplace:

$$L\left\{\frac{du(t)}{dt}\right\} = s \cdot U(s) - u^0$$

$$U_2 = -A \cdot U_1 - \tau \cdot \left(s \cdot U_2 - u_2^0\right)$$

$$U_1 = -A \cdot U_2 - \tau \cdot \left(s \cdot U_1 - u_1^0\right)$$

- Lösung im Zeitbereich:

$$u_2(t) = \frac{u_2^0 - u_1^0}{2} \cdot \exp\left(\frac{A-1}{\tau} \cdot t\right) + \frac{u_2^0 + u_1^0}{2} \cdot \exp\left(-\frac{A+1}{\tau} \cdot t\right)$$

© A. Steininger / TU Wien

Die Lösung

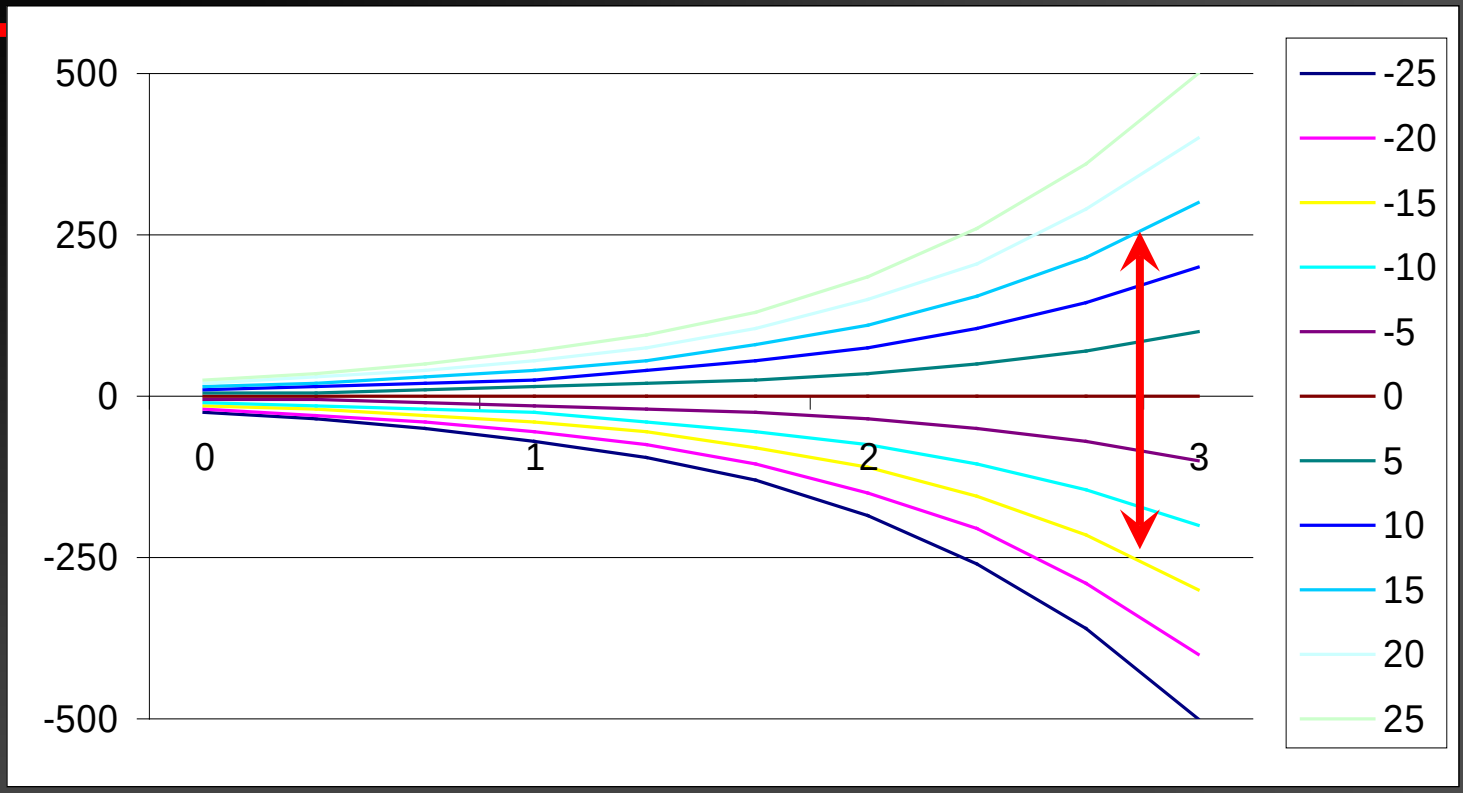
$$u_2(t) \approx \frac{u_2^0 - u_1^0}{2} \cdot \exp\left(\frac{A-1}{\tau} \cdot t\right)$$

- u_2^0-u_1^0 … Differenz der Initialspannungen (Ladungen an Cs); Null am balance point
- τ … RC Konstante, Bandbreite = 1/RC
- A … Verstärkung des Inverters (am balance point)
- A/τ … Verstärkungs-/Bandbreite-Produkt des Inv.

beginnend bei der Initialspannung steigt u_2 exponentiell mit der Zeit gegen die positive oder negative Versorgungsspannung

© A. Steininger / TU Wien

Plot von u_2 über der Zeit



Für jeden gegebenen Betrachtungszeitpunkt t können wir den verbotenen Eingangsspannungsbereich auf einen verbotenen Bereich der Initialspannunsdifferenz abbilden

© A. Steininger / TU Wien

Verbotener Anfangsbereich

$$u_2(t) \approx \frac{u_2^0 - u_1^0}{2} \cdot \exp\left(\frac{A-1}{\tau} \cdot t\right)$$

u_0

$$\pm u_{0,border}(t_r) = \pm U_{out,border} \cdot \exp\left(-\frac{A-1}{\tau} \cdot t_{res}\right)$$

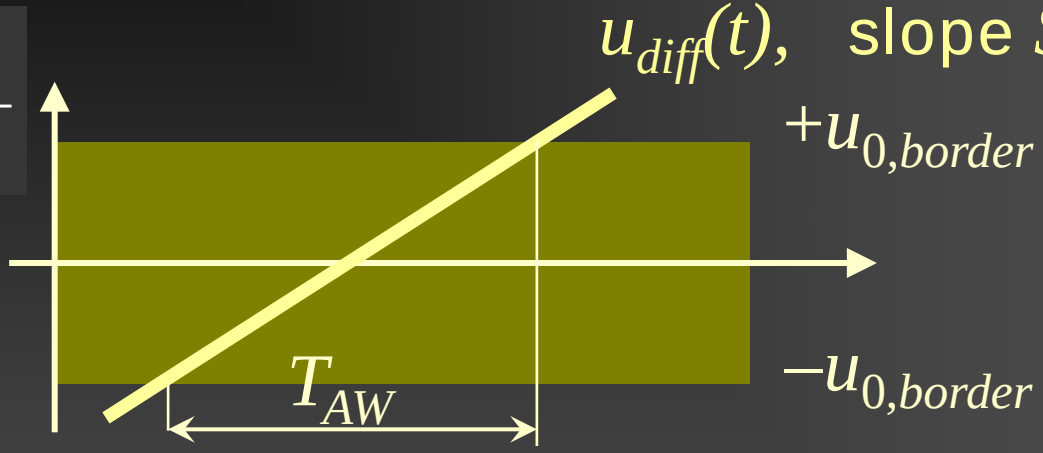
Der verbotene Eingangsspannungsbereich korres-pondiert mit einem verbotenen Bereich der Initialspannungs-differenz (gleich nach dem Umschalten). Dieser wird exponentiell kleiner mit größerer Resolution-time t_{res} und größerem Verstärkungs-/Bandbreite-Produktrigh A/t.

© A. Steininger / TU Wien

Aperture Window T_{AW}

- Wie lange bleibt die Eingangsspannung im verbotenen Bereich?

$$T_{AW} = \frac{2u_{0,border}}{S}$$



- Das hängt von der Steigung des rückgekoppelten Signales ab
  (...und von der Eingangsspannung!)

© A. Steininger / TU Wien

Kalibrierung von T_{AW}

- T_{AW} hängt ab von $u_{0,border}$, diese von t_{res}

$$T_{AW} = \frac{2u_{0,border}}{S} = \frac{2U_{0,border}}{S} \cdot \exp\left(-\frac{A-1}{\tau} \cdot t_{res}\right)$$

- Wenn das Ausgangssignal sofort verwendet wird:

$$T_{AW}(t_{res} = 0) = \frac{2U_{0,border}}{S} = T_{W0}$$

Technologie-parameter

- daher

$$T_{AW} = T_{W0} \cdot \exp\left(-\frac{A-1}{\tau} \cdot t_{res}\right)$$

© A. Steininger / TU Wien

Flanke im Aperture-Window

- Mit exponentialverteilten inter-arrival times der Eingangsereignisse (Rate λ_{dat}) und Abtastung mit Periode T_{clk} (d.h. Fenster T_{AW} wiederholt sich) lässt sich die Upset Rate berechnen zu

$$\lambda_{upset} = \lambda_{dat} \cdot \frac{T_{AW}}{T_{clk}}$$

- Damit ergibt sich die MTBU zu

$$MTBU = \frac{1}{\lambda_{upset}} = \frac{1}{\lambda_{dat}} \cdot \frac{T_{clk}}{T_{AW}}$$

© A. Steininger / TU Wien

Zusammensetzen der Lösung

$$MTBU = \frac{1}{\lambda_{upset}} = \frac{1}{\lambda_{dat}} \cdot \frac{T_{clk}}{T_{AW}}$$

$$T_{AW} = T_{W0} \cdot \exp\left(-\frac{A-1}{\tau} \cdot t_{res} \right)$$

$$MTBU = \frac{1}{\lambda_{dat}} \cdot \frac{T_{clk}}{T_{W0}} \cdot \exp\left(\frac{A-1}{\tau} \cdot t_{res} \right)$$

T_0

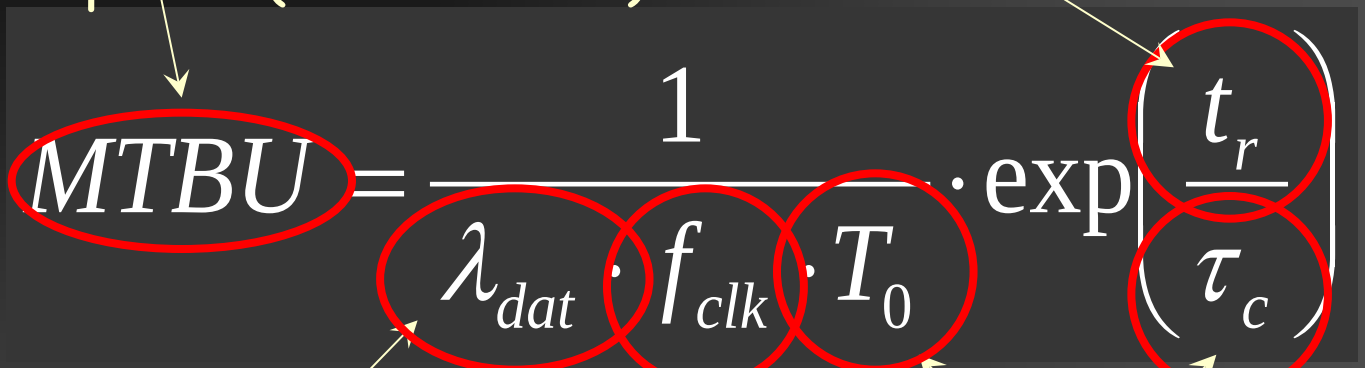
$1/\tau_C$

© A. Steininger / TU Wien

Die allg. verwendete Formel

Erwartungswert für Intervall zwischen Upsets (statistisch!)

Verfügbare Resolution time



Rate der Eingangsereignisse

Abtastfrequenz

Technologie-parameter

© A. Steininger / TU Wien

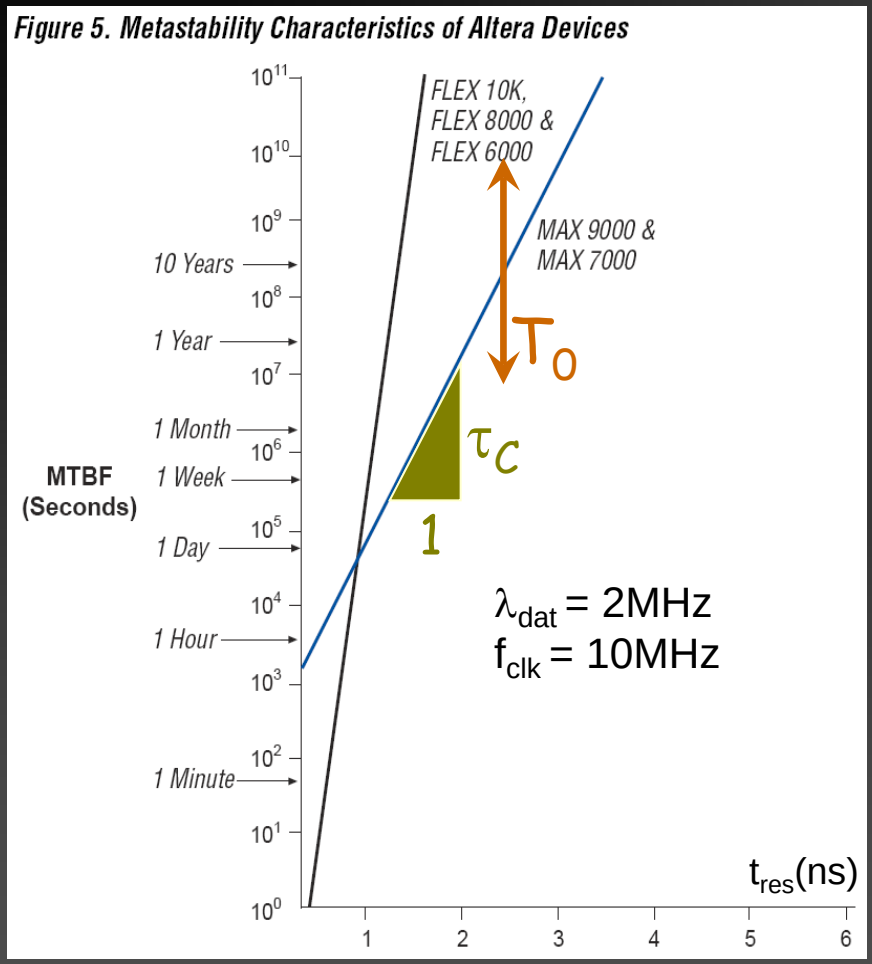
Provozieren von Metastability

- asynchronous inputs
- multiple clock domains
- clock divider (uncontrolled delay)

- low timing margins
- slow technology (gain/BW prod)
- supply drop (excessive delay)
- Operation under high temperature

© A. Steininger / TU Wien

Ermittlung von T_0, τ_C



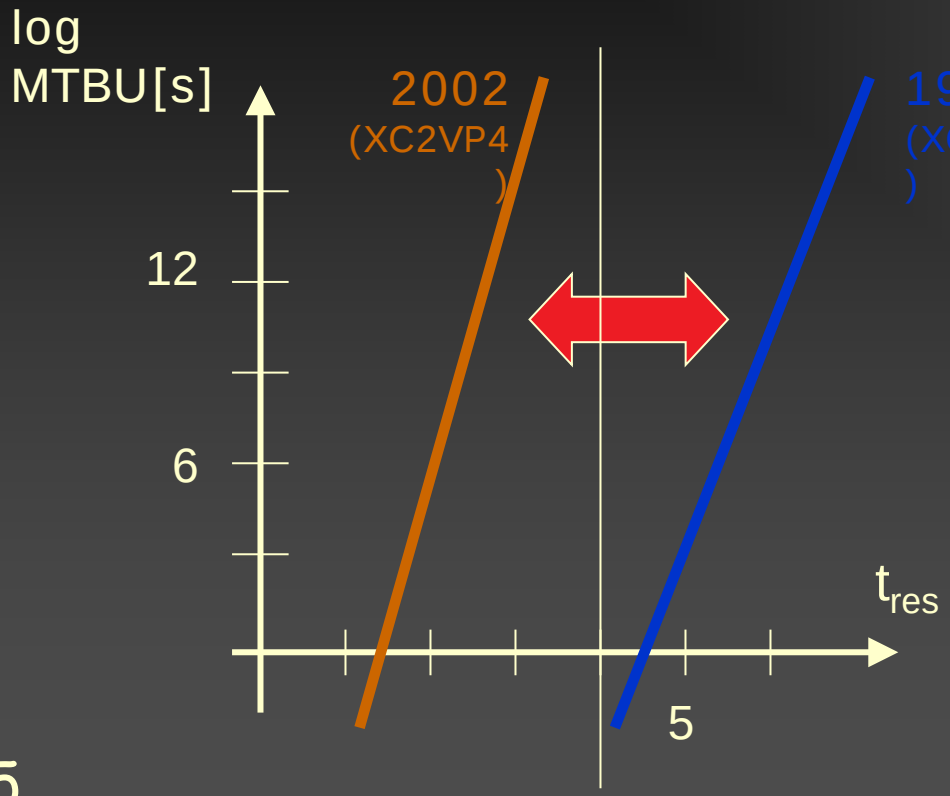
experimentell:

- variiere t_{res}
- messe MTBU
- log graph => Gerade
- Anstieg -> τ_C
- Offset -> T_0
- Typische Werte

© A. Steininger / TU Wien

Metastability - Trends

- Behauptung: „Metastabilität ist in neuen Technologien nicht mehr relevant“



ABER: Die Taktraten sind in diesem Zeitraum um einen Faktor 16 gestiegen –

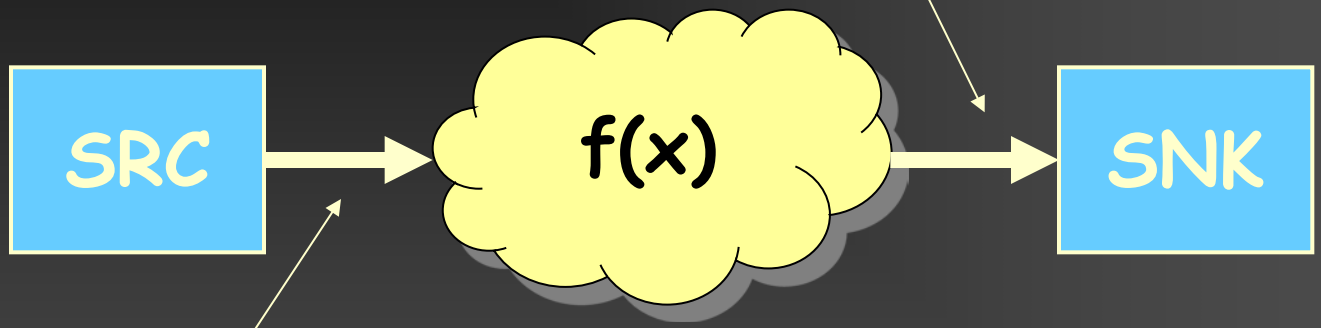
und die Zeitreserven haben sich im selben Maß verkleinert!

© A. Steininger / TU Wien

Das ursprüngliche Problem

Wann kann SNK die Daten verwenden?

Wenn sie gültig und konsistent sind

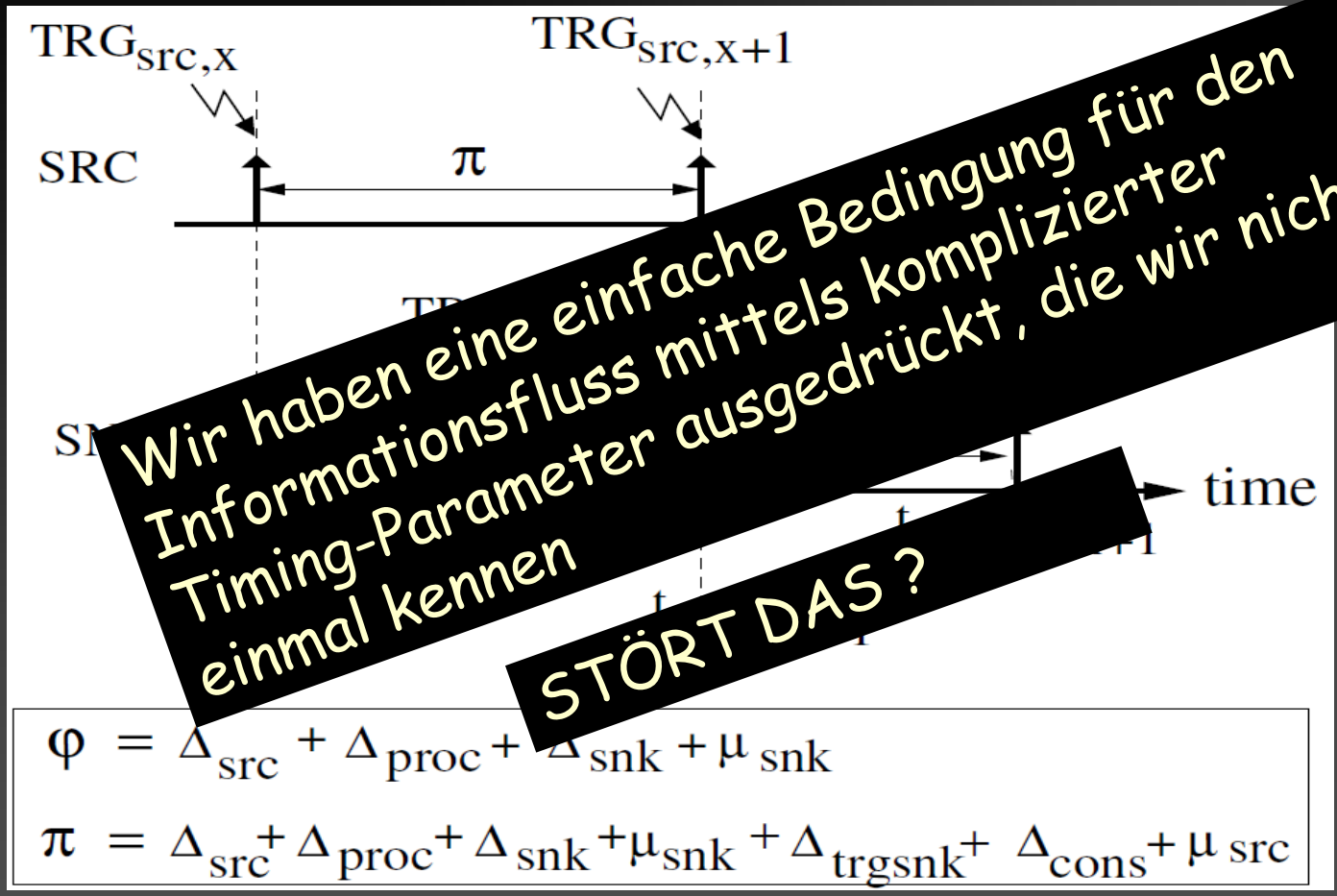


Wann kann SRC die nächsten Daten senden?

Wenn SNK die vorigen konsumiert hat

© A. Steininger / TU Wien

Der synchrone Ansatz



Wir haben eine einfache Bedingung für den Informationsfluss mittels komplizierter Timing-Parameter ausgedrückt, die wir nicht einmal kennen

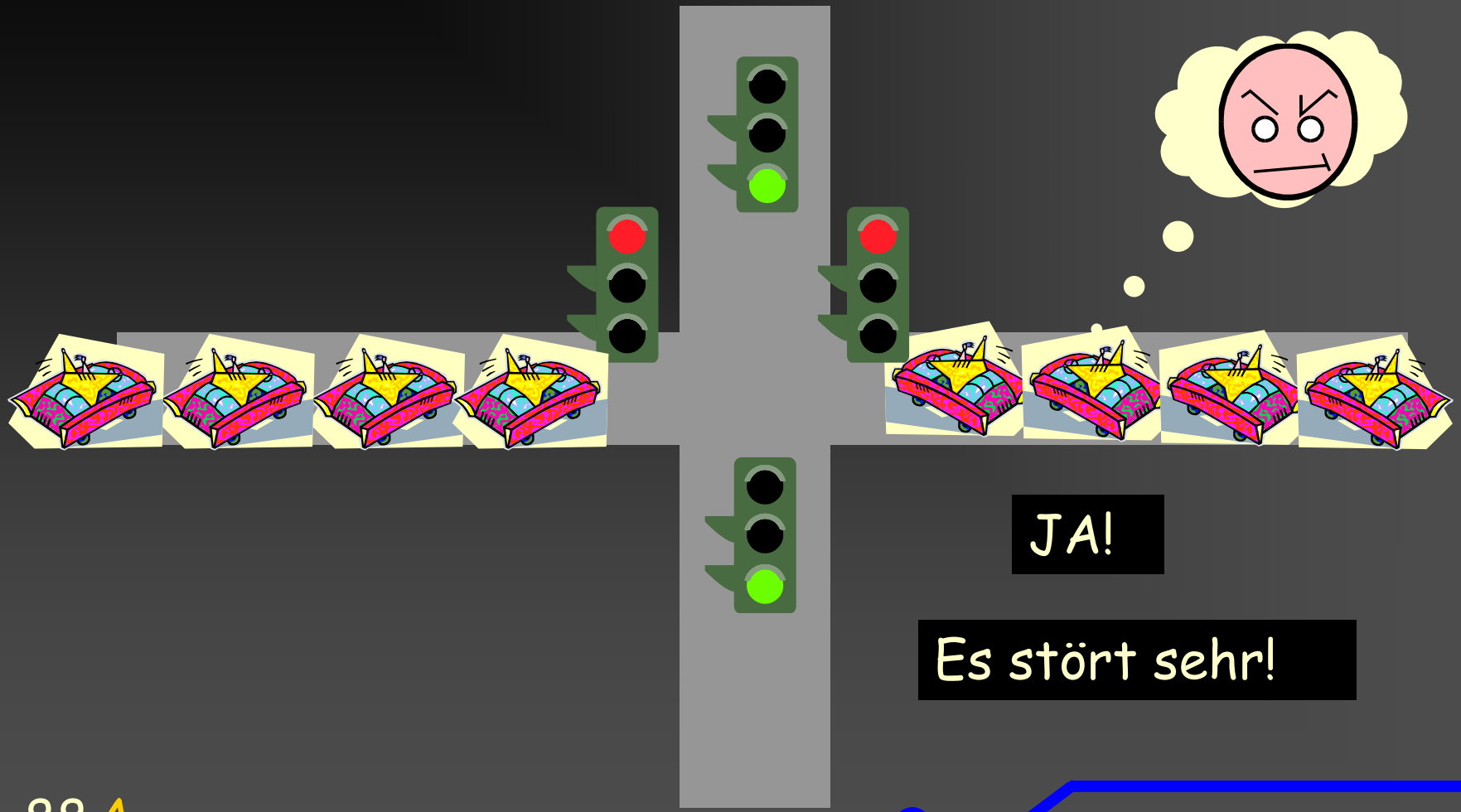
STÖRT DAS ?

$$\varphi = \Delta_{src} + \Delta_{proc} + \Delta_{snk} + \mu_{snk}$$

$$\pi = \Delta_{src} + \Delta_{proc} + \Delta_{snk} + \mu_{snk} + \Delta_{trgsnk} + \Delta_{cons} + \mu_{src}$$

© A. Steininger / TU Wien

Diese verflixte Ampel ...



© A. Steininger / TU Wien

Diese verflixte…

- Ampel
  - Wie viele Autos warten?
- Mikrowellenherd
  - Ist das Essen schon heiß?
- Scheibenwischer
  - Ist die Sicht durch die Scheibe gut?
- Treppenhauslicht
  - Hält sich jemand im Steigenhaus auf?

© A. Steininger / TU Wien

Der synchrone Ansatz



Eine Beziehung zwischen Datenfluss und Zeit ist sehr praktisch und effektiv, aber in Realität existiert eine solche Beziehung nicht (von sich aus).

Wir müssen beim Design eine solche Beziehung künstlich herstellen (Timing-Analyse, Constraints)

Na… T_{clk} kann SNK den …(x) verwenden, und zur gleichen … kann SRC neue Daten senden.

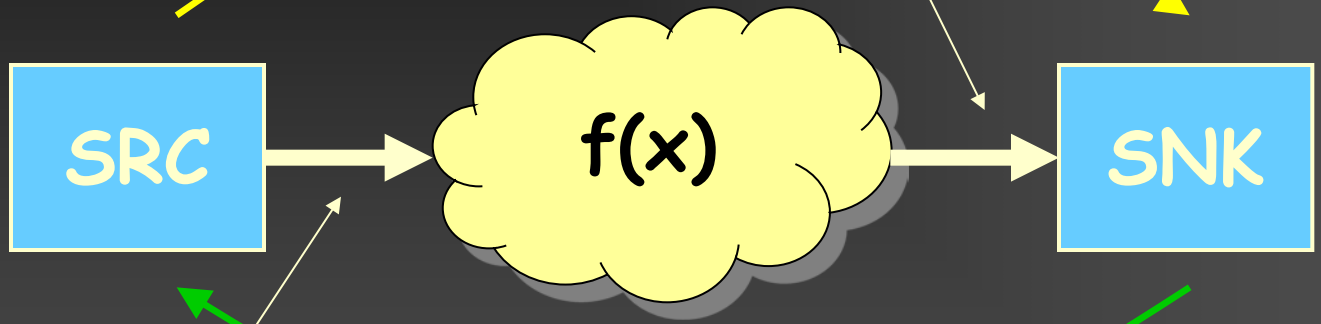
© A. Steininger / TU Wien

Geht es auch besser?

ACK: „Datenwort gültig, kann verwendet werden“

Wann kann SNK die Daten verwenden?

Wenn sie gültig und konsistent sind



Wann kann SRC die nächsten Daten senden?

Wenn SNK die vorigen konsumiert hat

REQ: „Daten konsumiert, bitte nächste senden“

© A. Steininger / TU Wien

Handshake als Lösung

- Der Handshake bildet eine **Regelschleife** für den Datenfluss zwischen SRC und SNK.
- Dies macht den Betrieb robuster als im Falle einer Steuerung
- Die Kunst des asynchronen Design liegt darin, die vielfältigen Regelschleifen richtig zu koordinieren.
- Das ist viel komplizierter als beim synchronen Design.

© A. Steininger / TU Wien



Zusammenfassung (1)

- In realen Schaltungen sind Signallaufzeiten unvermeidlich.
- Die Differenzen dieser Laufzeiten (Skew) variieren in nicht vorhersagbarer Weise.
- Boole'sche Logik geht von kontinuierlich gültigen und konsistenten Eingängen aus.
- Zeitliche Zusammenhänge sind in Boole'scher Logik nicht beschreibbar.
- Genau diesen Mangel muss eine Design-Methode kompensieren.

© A. Steininger / TU Wien



Zusammenfassung (2)

- Synchrones Design verwendet Zeitbedingungen, um die die Datenkonsistenz zu beurteilen.
- Diese Methode ist effizient und bewährt.
- Der Schluss von Zeitbedingung auf Konsistenz ist aber indirekt und willkürlich.
- Synchrones Design ist daher keine natürliche Lösung des Grundproblems.
- Dieser Mangel wird mit steigenden Taktfrequenzen immer deutlicher spürbar.

© A. Steininger / TU Wien



Zusammenfassung (3)

- Bei Timing Violations und an den Grenzen von Clock Domains kann es zu Metastabilität kommen.
- Eine Verletzung der Setup-/Hold time kann zu beliebig langem Verweilen des Ausgangs eines Flip-Flop in einem undefinierten Pegel führen.
- Übernimmt ein nachfolgendes Flip-Flop diesen undefinierten Zustand („Upset"), kommt es zu Fortpflanzung und zu inkonsistenter Interpretation.
- Die Wahrscheinlichkeit dafür kann berechnet werden. Sie steigt exponentiell mit sinkenden Timing-Reserven, hängt aber auch von der Technologie ab.

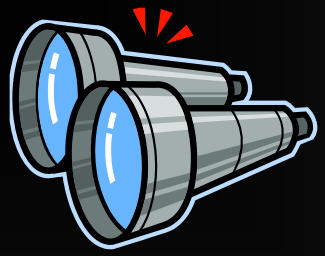
© A. Steininger / TU Wien

Defekte & Fehler

Warum auch Chips nicht ewig leben



© A. Steininger / TU Wien



Überblick

- Klassifikation der Ausfallmechanismen
- Wichtigste Ausfallursachen im Betrieb
  - Gate-oxid wear-out
  - Elektromigration
  - Electrostatic Discharge
- Welche Faktoren begünstigen den Ausfall?
- Die Badewannenkurve & Burn-in
- Fehlermanifestation & Test

© A. Steininger / TU Wien

Fehlerquellen im Aufwind

- Wachsender Zeitdruck beim Design
- Zunehmender Anteil von Fremd-Design
- Steigende Komplexität
- Kleinere Strukturgrößen
- Sinkende Versorgungsspannung
- Steigende Taktraten
- Widrige Umgebungsbedingungen
- Laien als Anwender, …

© A. Steininger / TU Wien

Fehlertypen im Lebenszyklus

- Designfehler
  - Fehlerhafte Spezifikation bzw. Fehler b. deren Umsetzung
  - Abhilfe: Validation und Simulation
    Test ist ungeeignet weil zu spät !
- Fertigungsfehler (Defekte)
  - Fehler b. d. physikal. Realisierung eines korrekten Designs
  - Abhilfe: Erkennen von Defekten im „Factory-Test“
- Fehler im Betrieb (Ausfälle)
  - Ausfall einer Komponente während des Einsatzes (Umwelteinflüsse, Alterung)
  - Abhilfe: on-line Fehlererkennung, Fehlertoleranz

© A. Steininger / TU Wien

Fertigung & Integration

Ursachen für Ausfälle noch vor dem Einsatz:

- **Wafer:** Verunreinigungen, Kristalldefekte, Microcracks, ...
- **Prozesse:** Masken-Alignment, Unterätzung, Ionen, ...
- **Packaging:** Hohlräume, Bonding-Defekte, Microcracks
- **Transport:** Handhabung, Umweltbedingungen
- **Bestückung:** Handhabung, Kurzschlüsse, kalte Lötstellen,...
- **Inbetriebnahme/Systemintegration:** Buskonflikte,...

© A. Steininger / TU Wien

Fehlertypen im Lebenszyklus

- Designfehler
  - Fehlerhafte Spezifikation bzw. Fehler b. deren Umsetzung
  - Abhilfe: Validation und Simulation
    Test ist ungeeignet weil zu spät !
- Fertigungsfehler (Defekte)
  - Fehler b. d. physikal. Realisierung eines korrekten Designs
  - Abhilfe: Erkennen von Defekten im „Factory-Test"
- Fehler im Betrieb (Ausfälle)
  - Ausfall einer Komponente während des Einsatzes (Umwelteinflüsse, Alterung)
  - Abhilfe: on-line Fehlererkennung, Fehlertoleranz

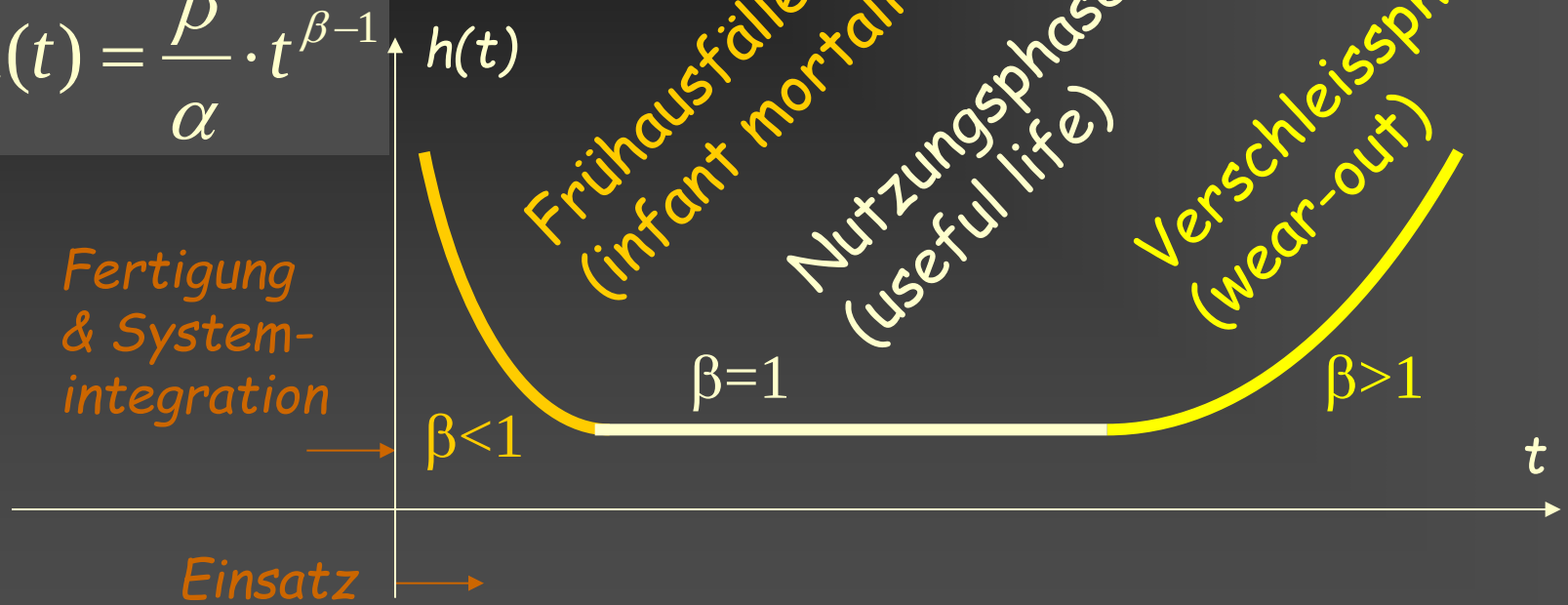
© A. Steininger / TU Wien

Die „Badewannenkurve"



Wahrscheinlichkeit $h(t)$ dass ein Produkt bis zum Zeitpunkt t gut funktioniert und exakt dann ausfällt, gehorcht der sog. „Weibull-Verteilung"

$$h(t) = \frac{\beta}{\alpha} \cdot t^{\beta-1}$$



© A. Steininger / TU Wien

Fehlerursachen im Einsatz

- Electrical Stress

  durch unsachgemäßes Handling, Design, etc.
  Auftreten kontinuierlich über die Lebensdauer

- Intrinsic

  durch Material- oder Prozessfehler bedingt
  meist als infant Mortality, selten Wear-out

- Extrinsic

  durch Verbindungen, Passivierung, Packaging oder Umweltbedingungen

  tritt in allen Phasen auf (yield loss ... wear-out)

© A. Steininger / TU Wien

Electrical Stress

- Electrostatic Discharge (ESD)

  Elektrostatische Entladung führt zu Strom- oder Spannungsspitzen mit Dauer 0,1ns ... 1μs

- Electrical Overstress (EOS)

  zu hohe Spannung / zu hohen Strom für Zeitdauer > 1μs (typ. > 1ms)

- Latch-up

  Spannungsspitze zündet parasitären Thyristor

© A. Steininger / TU Wien

Elektrostatische Entladung

- Ursache:
  - unsachgemäße Handhabung (Erdungsband!)
- mögliche Folgen:
  - Durchbruch oder Beschädigung des Gate-Oxid
  - therm. Zerstörung durch hohe Drainströme
- Abhilfe:
  - Klemmdioden (clamp diodes)
  - Spezielle Geometrien und Dotierungen

© A. Steininger / TU Wien

Electrical Overstress

- Ursachen:
  - Spannungsspitzen in der Versorgung
  - Überspannungen durch Blitzschlag
  - schlechtes Design („out of spec")
- mögliche Folgen:
  - therm. Zerstörung durch hohe Drainströme
  - Durchbruch oder Beschädigung des Gate-Oxid
- Abhilfe:
  - gut geschützte stabile Versorgung
  - „konservatives" Design

© A. Steininger / TU Wien

Latch-Up

- Ursache:

  In den Strukturen am Chip treten unerwünschte Thyristor-Strukturen auf, die durch Über- oder Unterschwingen der Eingangsspannung bzw. der Versorgung gezündet werden können.

- mögliche Folgen:

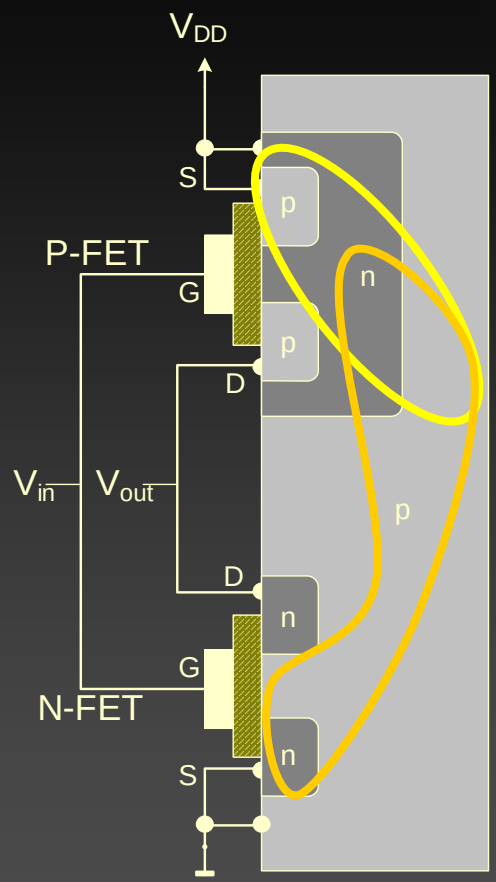
  thermische Zerstörung durch zu hohe Ströme

- Abhilfe:

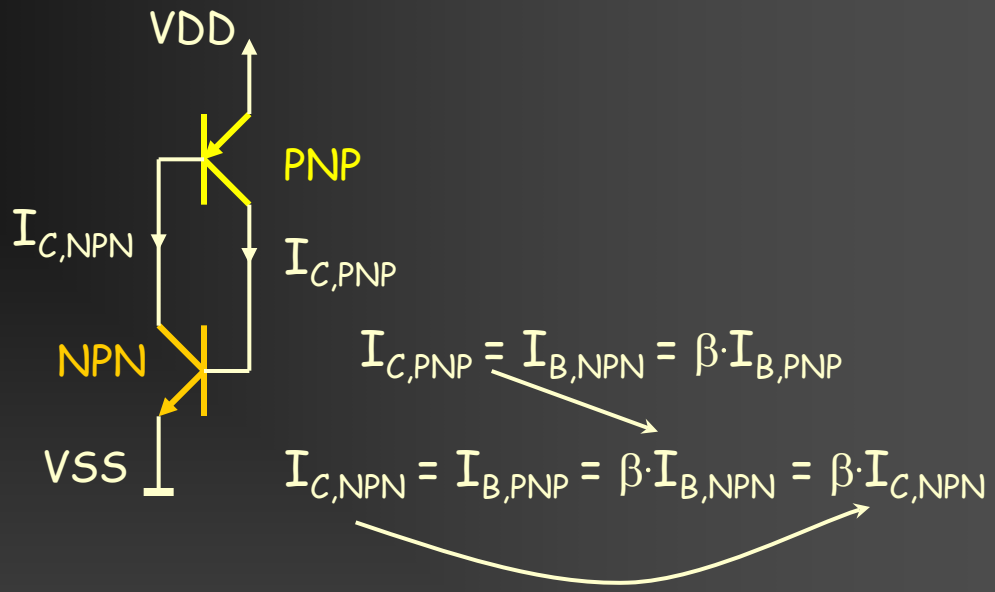
  Kann durch spezielle Maßnahmen („Latch-up Protection“) im Design vermieden werden.

© A. Steininger / TU Wien

Latch-up beim CMOS Inverter







Für Stromverstärkung $\beta > 1$ schaukeln sich die parasitären Transistoren gegenseitig zu hohen Strömen auf

© A. Steininger / TU Wien

Fehlerursachen im Einsatz

- ✓ Electrical Stress:
  - ✓ Electrostatic Discharge
  - ✓ Electrical Overstress
  - ✓ Latch-Up
- Intrinsic

  durch Material- oder Prozessfehler bedingt
  meist als infant Mortality, selten Wear-out
- Extrinsic

  durch Verbindungen, Passivierung, Packaging oder Umweltbedingungen

© A. Steininger / TU Wien

Intrinsische Fehlerquellen

- Gate-oxide Wear-out
  schlechtes Gate-oxid, oftmaliges Programmieren
- Ionic contamination
  Verunreinigungen stören Ladungsverhältnisse
- Oberflächenladungen
  Ladungen im Isolator (z.B. „heisse Elektronen")
- Kristalldefekte
  Leckströme an gesperrten pn-Übergängen
- Piping
  durch Störstellen induzierte Strompfade

© A. Steininger / TU Wien

Gate-Oxide wear-out

- Wear-out:
  - Störstellen im Gate-oxid bilden „Stützpunkte" für Ladungsträger im Isolator

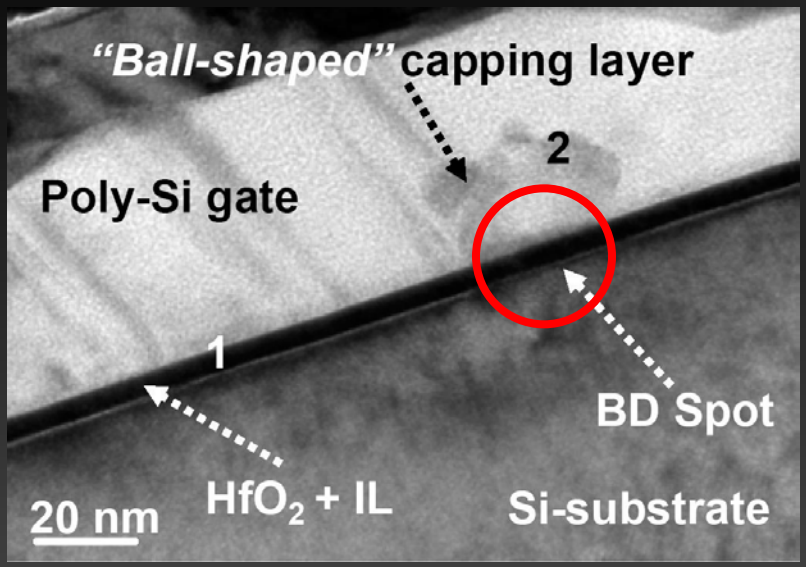
  => verminderte effektive Dicke des Isolators

  => drastisch erhöhte Tunnelströme („Leakage")
- Breakdown:
  - Ausbildung von Strompfaden und weiteren Störstellen
  - Bildung eines durchgängigen Strompfades

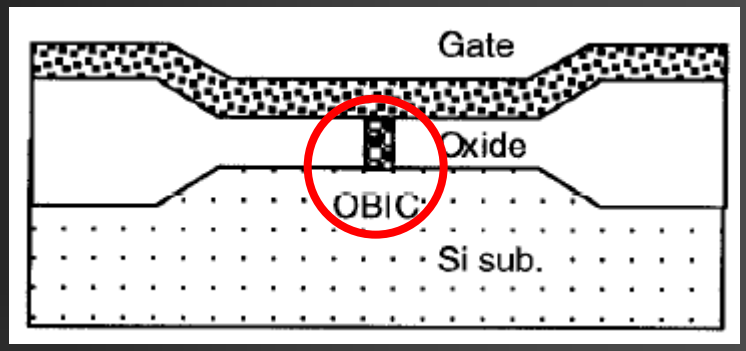
  => Durchbruch des Isolators

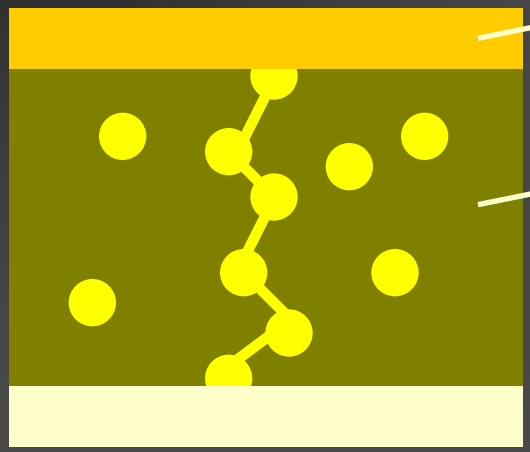
© A. Steininger / TU Wien

Gate-Oxid-Breakdown: Bilder



Quelle: SPIE society
http://www.spie.org





© A. Steininger / TU Wien

Fehlerursachen im Einsatz

- ✓ Electrical Stress:
  - ✓ Electrostatic Discharge
  - ✓ Electrical Overstress
  - ✓ Latch-Up
- ✓ Intrinsic

  durch Material- oder Prozessfehler bedingt

  Beispiel: ✓ Gate-Oxide wear-out
- Extrinsic

  durch Verbindungen, Passivierung, Packaging oder Umweltbedingungen

  tritt in allen Phasen auf (yield loss ... wear-out)

© A. Steininger / TU Wien

Extrinsische Fehlerquellen / 1

- **Elektromigration**
  Wanderung v. Metall-Atomen durch hohe Stromdichte
- **Kontaktmigration**
  Diffusion Metall/Si am Kontakt dch. hohe Stromdichte
- **Stress-induzierte Migration**
  Wanderung v. Metall-Atomen infolge mechanischer Beanspruchung (z.B. „thermal Mismatch")
- **Microcracks**
  mechanische Bruchstellen

© A. Steininger / TU Wien

Extrinsische Fehlerquellen /2

- **Die attach Failures**
  unerwünschte Hohlräume => Korrosion, Überhitzung
- **Bonding Failures**
  Reißen, Ablösen, Kurzschluß, Whisker
- **Popcorn Effect**
  Ausdehnung von Feuchtigkeit => Die bricht
- **Korrosion**
  chemische Effekte (bei Wärme, Feuchtigkeit, Ionen)
- **Soft Errors**
  Bit-Flips durch Strahlung (Luftfahrt, Gehäuse!)

© A. Steininger / TU Wien

Elektromigration: Prinzip

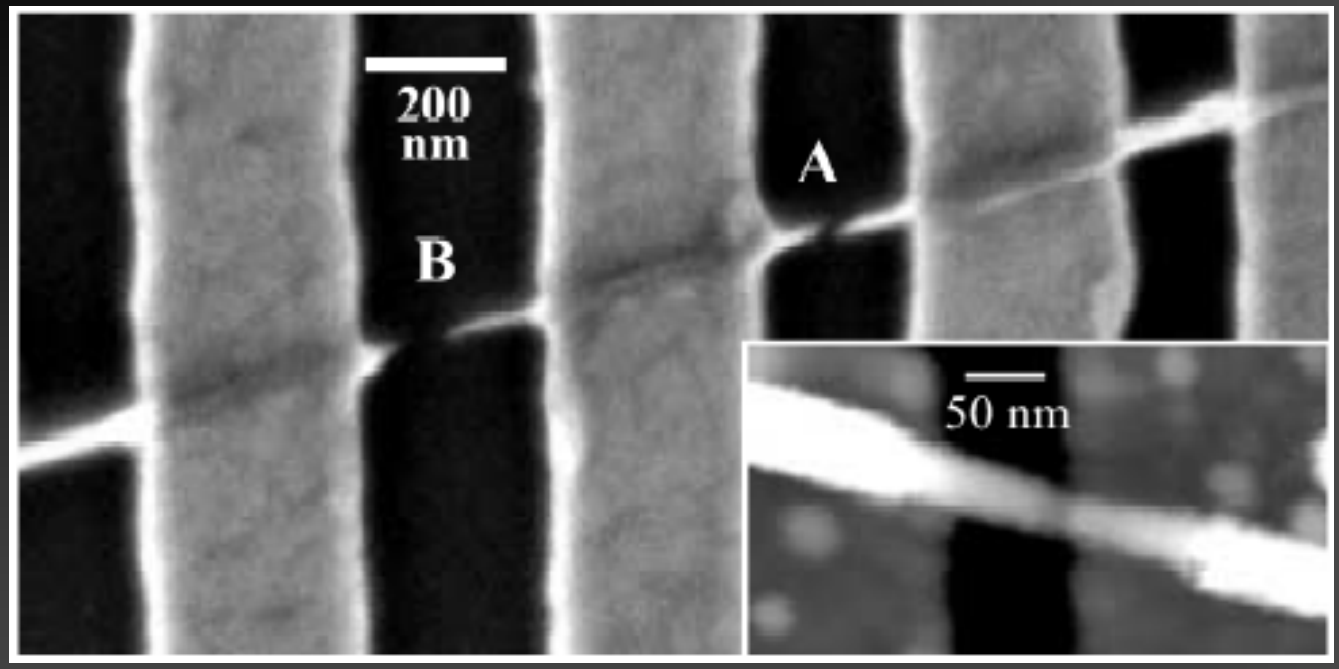
Bei hoher Stromdichte bildet die Vielzahl von rasch bewegten Elektronen einen sog. „Elektronenwind", der durch Stoßprozesse die Atome bewegt und dadurch mit der Zeit Material verschiebt.



Elektromigration betrifft also nur leitende Materialien. Der Materietransport findet in der *Bewegungsrichtung* der Elektronen statt.

© A. Steininger / TU Wien

Electromigration: Voiding



Abwanderung von Material führt zu Unterbrechungen (siehe A und B)

© A. Steininger / TU Wien

Electromigration: Hillocks



Anlagerung von Material führt zu unerwünschten Auswüchsen („Hillocks“) an Materialgrenzen

© A. Steininger / TU Wien

Ausmaß der Elektromigration

Black's Law:

$$MTTF = \frac{A}{J^2} \exp\left(\frac{E}{kT}\right)$$

J...Stromdichte [A/cm^2], E_{act} = 0.5 ... 1.5eV, A ... Konstante
T...Temperatur [K], k ... Boltzmann-Konst. = $8.6*10^{-5}$eV/K

➡ Hohe Temperatur und hohe Stromdichte begünstigen Elektromigration und führen daher zu höheren Defektraten. Die Stromdichte geht quadratisch ein, die Temperatur exponentiell.

(Beobachtung: Kurze Leitungen sind weniger anfällig)

© A. Steininger / TU Wien

Zuverlässigkeits-Kenngrößen

- **Fehlerrate**
  Beschreibt Häufigkeit des Auftretens von (unkorrelierten) Fehlern (z.B.: 10^{-6}/h)
- **Reliability *R(t)***
  Wahrscheinlichkeit, dass das System nach erfolgreichem Start bei $t = 0$ zum Zeitpunkt $t = t_1$ noch ordnungsgemäß funktioniert (z.B.: R (3h) = 0,2)
- **Mean Time to Failure (MTTF)**
  Erwartungswert der Betriebsdauer von erfolgreichem Start bis zum Ausfall (z.B.: 10^7h)

© A. Steininger / TU Wien

Black's Law – Beispiel 1

- Problem:

  Um wie viel reduziert sich die MTTF einer metallischen Verbindung im Chip aufgrund der Elektromigration, wenn sie aufgrund eines Fertigungsfehlers nur den halben Querschnitt aufweist?

- Lösung:
  - Halber Querschnitt => doppelte Stromdichte J
  - MTTF ist proportional $1/J^2$ (Black's Law)
  - MTTF wird daher auf $1/2^2$ = 25% reduziert

© A. Steininger / TU Wien

Black's Law – Beispiel 2

- Problem:

  Auf wie viel reduziert die Elektromigration die MTTF einer Al-Verbindung (E_{act}=0.7eV) in einem Commercial-Chip (bis 70°C), wenn er im Military-Temperaturbereich betrieben wird (125°C)?

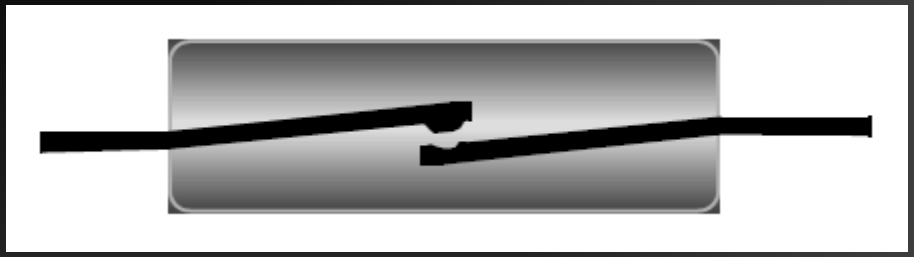
- Lösung:

$8.6*10^{-5}eV/K$

$$\frac{MTTF_{mil}}{MTTF_{com}} = \frac{\frac{A}{J^2}}{\frac{A}{J^2}} \cdot \frac{\exp\left(\frac{E}{k \cdot (125+273)K}\right)}{\exp\left(\frac{E}{k \cdot (70+273)K}\right)} \approx \frac{1}{26} \approx 4\%$$

© A. Steininger / TU Wien

Electromigration: Beispiel



[Agilent]

- Kontakt eines Reed-Relais altert durch Electromigration (Kontaktwiderstand ↑)
- tritt nur bei Gleichstrom auf, nicht bei Wechselstrom

  ➡ für Wechselstrom sind höhere Werte zulässig als für Gleichstrom

© A. Steininger / TU Wien

Dehnungskräfte im Chip

- Ein Cu-Stab hat bei 430°C (Aufbringen der Passivierung) die Länge $L+\Delta L$, bei 30°C Länge L.

  Berechne die relative Längenänderung $\varepsilon = \Delta L/L$!

  $\varepsilon = \Delta L/L = \alpha_{Cu} * \Delta T = 16{,}5 * 10^{-6}/°C * 400°C = 0{,}66\%$
- Welche Kraft müsste man aufbringen, um bei 30°C die Länge $\Delta L+L$ beizubehalten? ($E_{y,Cu}$ = 120GPa)

  $\sigma = \varepsilon * E_y = 0{,}0066 * 120 * 10^9\ Pa = 792MPa$

  zum Vergleich: Cu reißt bei ca. 300MPa

  bei Querschnitt $2{,}5mm^2$ => $792MN/m^2 * 2{,}5mm^2 = 1980N$

  (entspricht einem Gewicht von 195kg)

© A. Steininger / TU Wien

Folgen der Dehnungskräfte

- Die enormen Dehnungskräfte führen zu
  - mechanischen Bruchstellen im Silizium-Die (Microcracks)
  - Stress-induzierter Migration
- Diese Kräfte treten auf
  - besonders bei der Fertigung, aber auch
  - durch Temperaturschwankungen in der Umgebung (auch bei Lagerung!) bzw.
  - bei Erwärmung infolge Leistungsverbrauch.

© A. Steininger / TU Wien

Stress-induzierte Migration

- **Effekt:**

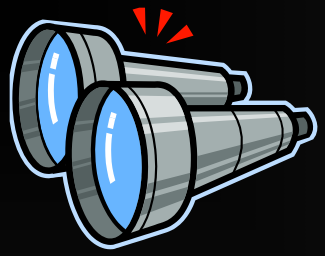
  Materialwanderung infolge
  mechanischer Spannung
- **Ursache:**

  Unterschiede im Dehnungsverhalten (Dehnungs-koeffizienten) verschiedener Materialien
- **Folge:**

  Unterbrechungen durch „Voids“
- **Besonderheit:**

  Tritt auch im ausgeschalteten Zustand auf

© A. Steininger / TU Wien



Überblick

- Welche Ursachen für Ausfälle gibt es?
- Fehlerquellen im Lebenszyklus von Chips
- Klassifikation der Ausfallmechanismen
- Wichtigste Ausfallursachen im Betrieb
  - Gate-oxid wear-out
  - Elektromigration
  - Electrostatic Discharge
- Welche Faktoren begünstigen den Ausfall?

© A. Steininger / TU Wien

Hitliste der Ausfallursachen

#1: Gate oxide wear out

- Wichtigster Alterungsdefekt, bremst Miniaturisierung, hohe Forschungsaktivität

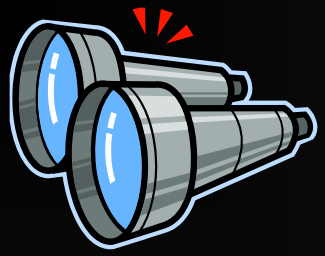
#2: Electromigration

- Hat Ende der 60er Jahre fast die Halbleiter-industrie gestoppt, nun technologisch im Griff

#3: Electrostatic Discharge

- Die MTTF eines Designs lässt sich „einstellen". Beim Design muss stets ein Kompromiss zwischen Performance, MTTF und Preis getroffen werden!

© A. Steininger / TU Wien



Überblick

- Klassifikation der Ausfallmechanismen
- Wichtigste Ausfallursachen im Betrieb
  - Gate-oxid wear-out
  - Elektromigration
  - Electrostatic Discharge
- Welche Faktoren begünstigen den Ausfall?
- Die Badewannenkurve & Burn-in
- Fehlermanifestation & Test

© A. Steininger / TU Wien

Wie baut man kurzlebige Chips?

- Unreines Wafermaterial, Kristalldefekte
- Verunreinigungen durch Prozesse
- Dünnes, unreines Gate-oxid
- Metallisierung mit geringem Querschnitt
- fehlender ESD-Schutz
- fehlender Schutz gegen Latch-up
- undichtes Package

© A. Steininger / TU Wien

Wie provoziert man Ausfälle ?

- hohe Temperaturen
  - Elektromigration, Korrosion, ...
- Temperaturzyklen
  - Dehnungskräfte
- hohe Spannung
  - Hohe Feldstärken, starke Beanspruchung des Gate-Oxide, Erwärmung,...
- hohe Strombelastung
  - Hohe Verlustleistung => Erwärmung, Elektromigration
- unprofessionelles Handling (ESD)
- Betrieb hart an den / jenseits der Grenzwerte

© A. Steininger / TU Wien

Arrhenius-Gleichung

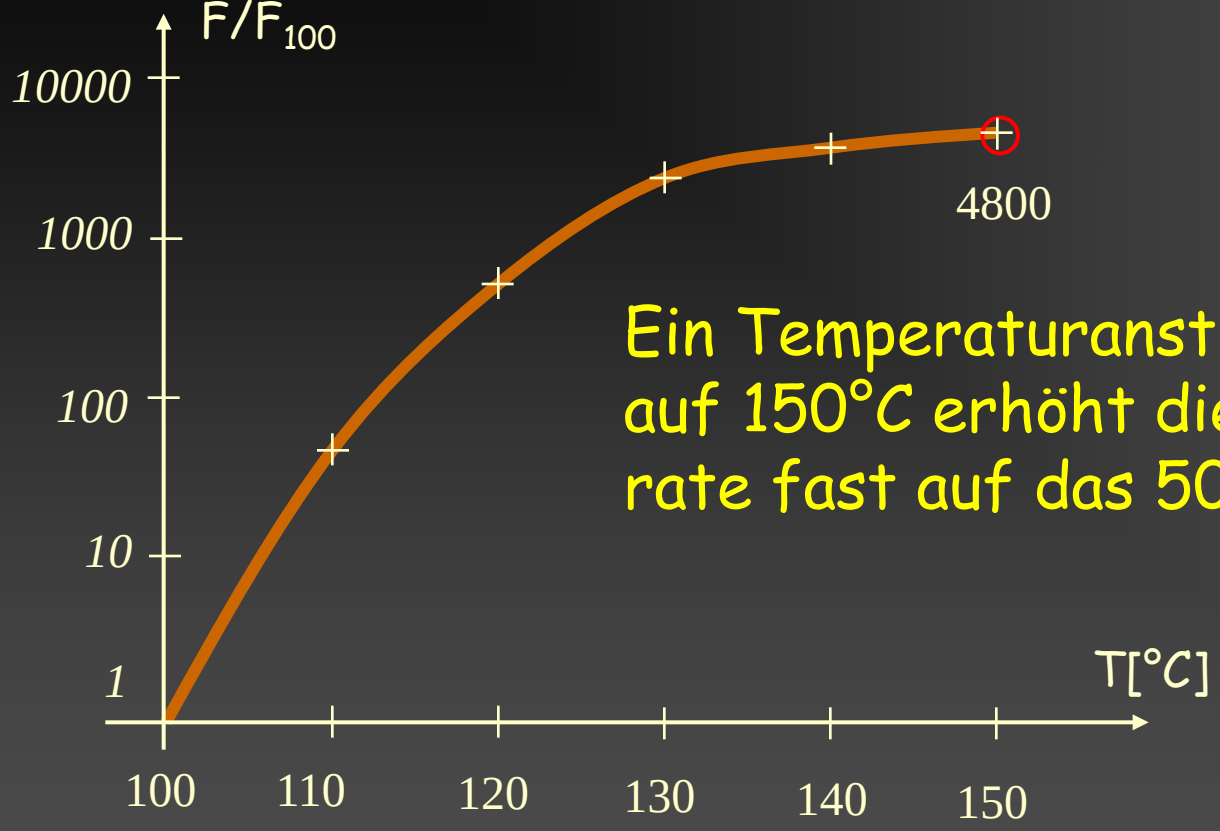
Für viele Fehlermechanismen gilt:

Fehlerrate $$F = C \cdot \exp\left(\frac{-E_{act}}{k \cdot T}\right) = \frac{1}{MTTF}$$

Eine Temperaturerhöhung bewirkt ein exponentielles Ansteigen der Fehlerrate !

© A. Steininger / TU Wien

Arrhenius: Messergebnisse



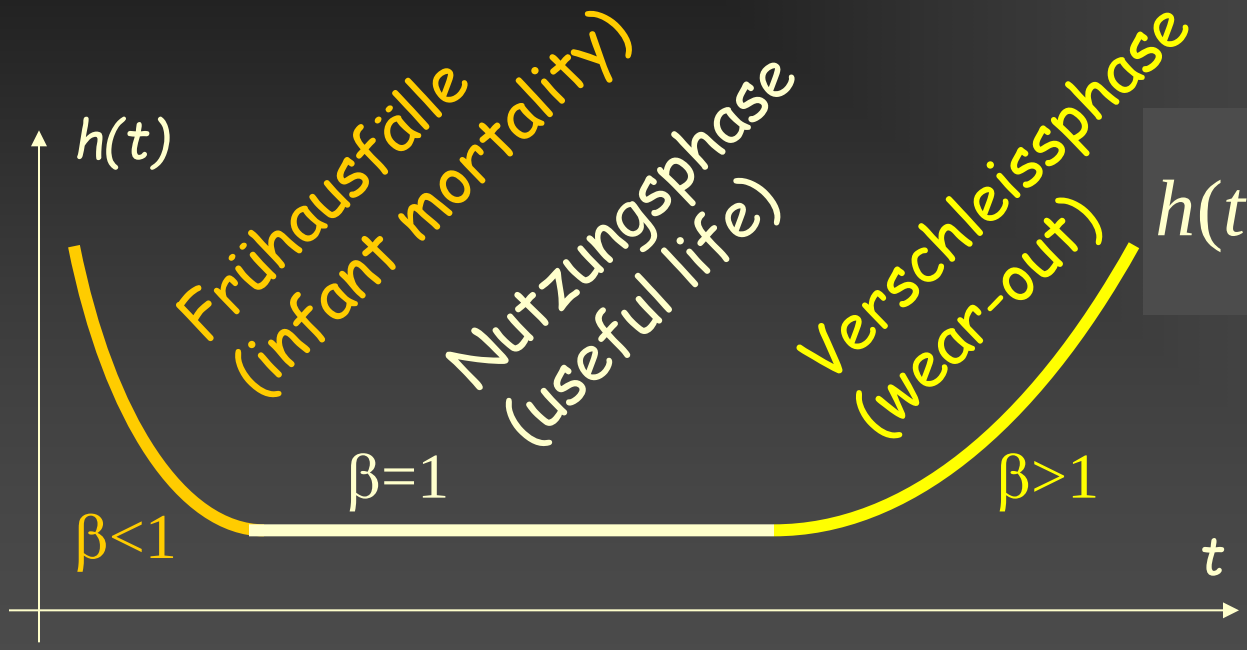
Ein Temperaturanstieg von 100 auf 150°C erhöht die Fehlerrate fast auf das 5000-fache !

Quelle: Texas Instruments

© A. Steininger / TU Wien

Die „Badewannenkurve"

Wahrscheinlichkeit $h(t)$ dass ein Produkt bis zum Zeitpunkt t gut funktioniert und exakt dann ausfällt, gehorcht der sog. „Weibull-Verteilung"

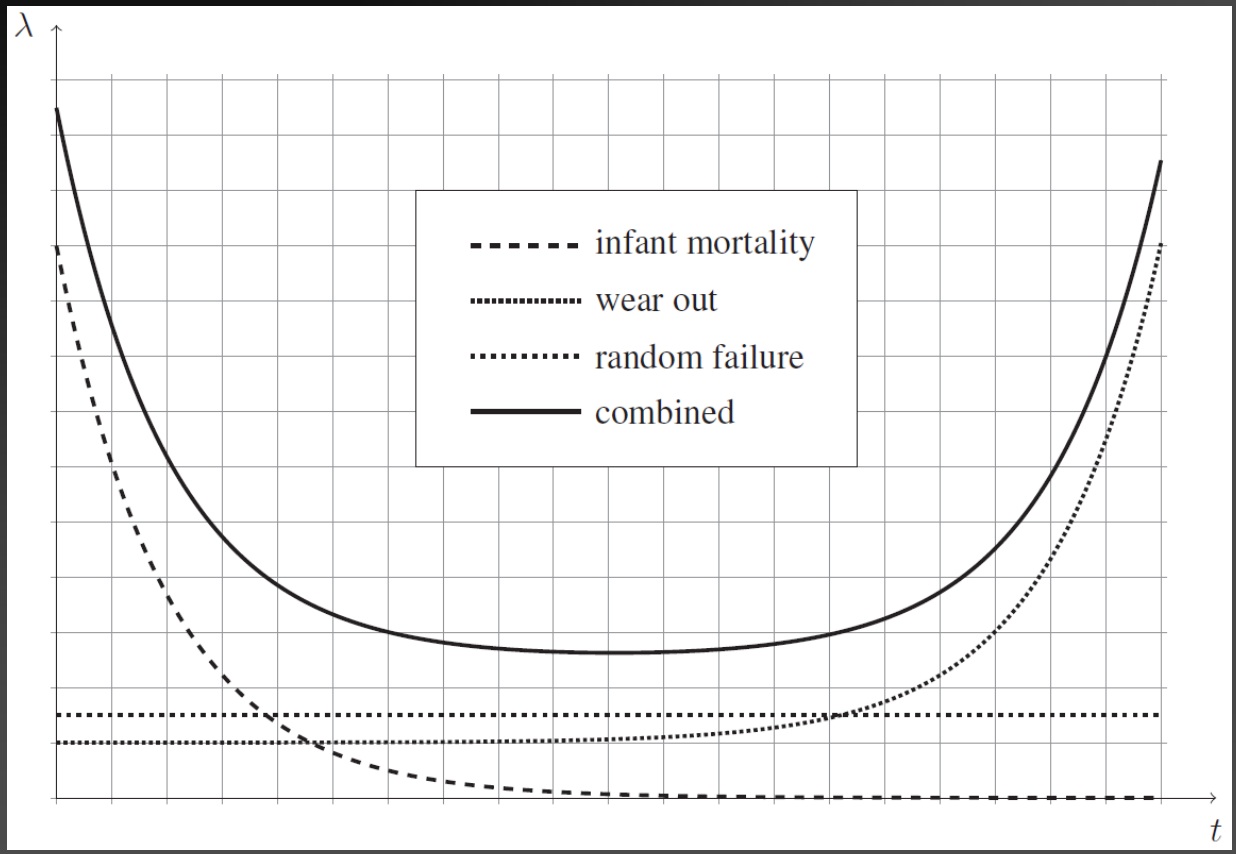


$$h(t) = \frac{\beta}{\alpha} \cdot t^{\beta-1}$$

© A. Steininger / TU Wien

„Badewannenkurve" - Details

Überlagerung aus 3 unabhängigen Effekten



© A. Steininger / TU Wien

Burn-in

- Gemäß der Arrhenius-Gleichung treten bei erhöhter Temperatur latente Fehlermechanismen schneller zutage.
- Gleiches gilt für andere „Stress-bedingungen" wie z.B. hohe Versorgungsspannung.
- Mittels Burn-in (= Testen unter hohem Stress) kann man daher das Auftreten der infant mortality beschleunigen („künstliche Alterung") und gelangt rasch zur Nutzungsphase („Boden der Badewanne").

© A. Steininger / TU Wien

Künstliche Alterung

Äquivalentes Alter = Acceler. Factor x Testdauer

Temperature Acceleration Factor

$$AF_T = \exp\left(\frac{E_{act}}{k \cdot T_{normal}} - \frac{E_{act}}{k \cdot T_{stress}}\right)$$

x

Voltage Acceleration Factor

$$AF_V = \exp\left[\gamma \cdot \left(U_{stress} - U_{normal}\right)\right]$$

© A. Steininger / TU Wien

Beispiel zum Burn-in

Ein Chip wird beim Burn-in 96 Stunden (= 4 Tage) lang bei T_j = 130°C und 6V betrieben. Welcher Betriebsdauer entspricht das bei T_j = 70°C und 5V?

(k = 8,6*10^{-5}eV/K; E_{act} = 0,7eV; γ = 3.2V^{-1})

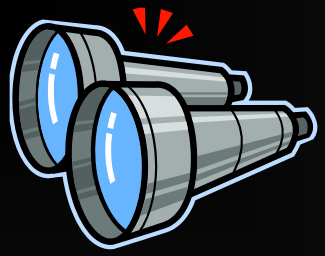
$$AF_T = \exp\left(\frac{E_{act}}{k \cdot T_{normal}} - \frac{E_{act}}{k \cdot T_{stress}}\right) \approx 34{,}2$$

$$AF_V = \exp\left[\gamma \cdot \left(U_{stress} - U_{normal}\right)\right] \approx 24{,}5$$

$$AF \approx 838$$

Der Test entspricht einer Betriebsdauer von 80.000 Stunden, das sind mehr als 9 Jahre.

© A. Steininger / TU Wien



Überblick

- Klassifikation der Ausfallmechanismen
- Wichtigste Ausfallursachen im Betrieb
  - Gate-oxid wear-out
  - Elektromigration
  - Electrostatic Discharge
- Welche Faktoren begünstigen den Ausfall?
- Die Badewannenkurve & Burn-in
- Fehlermanifestation & Test

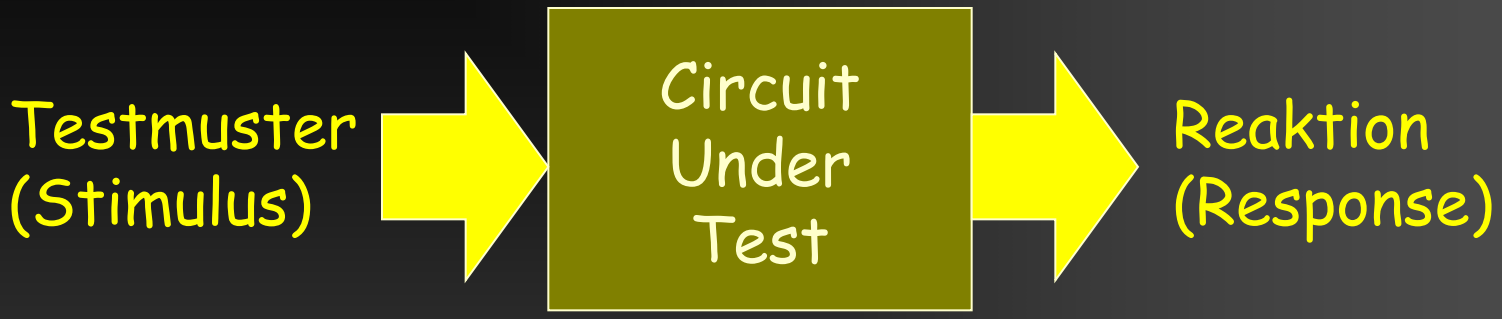
© A. Steininger / TU Wien

Fehlermanifestation

- **Kontaktprobleme**
  Unterbrechung, hoher Widerstand, Wackelkontakt
- **Isolationsprobleme**
  Kurzschluss, ungewollt niederohmige Verbindung
- **Parametrische Fehler**
  hohe Leckströme/Leistung, Schwellwertverschiebung
- **Dynamische Fehler**
  reduzierte Geschwindigkeit
- **Speicherfehler**
  „Bit-Flips“

© A. Steininger / TU Wien

Prinzip des Testens



Auf logischer Ebene wird die Reaktion des Testobjekts („Circuit under Test") auf einen Stimulus überprüft.

- Entspricht diese den Erwartungen (= der Spezifikation) => Test bestanden („pass"),
- anderenfalls => Testobjekt ist defekt („fail")

© A. Steininger / TU Wien

Fault Mapping

Defekte
treten in der
physikalischen
Domäne
auf

Der
Test
überprüft die
logische
Funktion



Wie bilden sich Defekte auf die logische Funktion ab?

© A. Steininger / TU Wien

Physikalisches Beispiel /1

F1: Kurzschluß Metallisierung
n1 an VSS

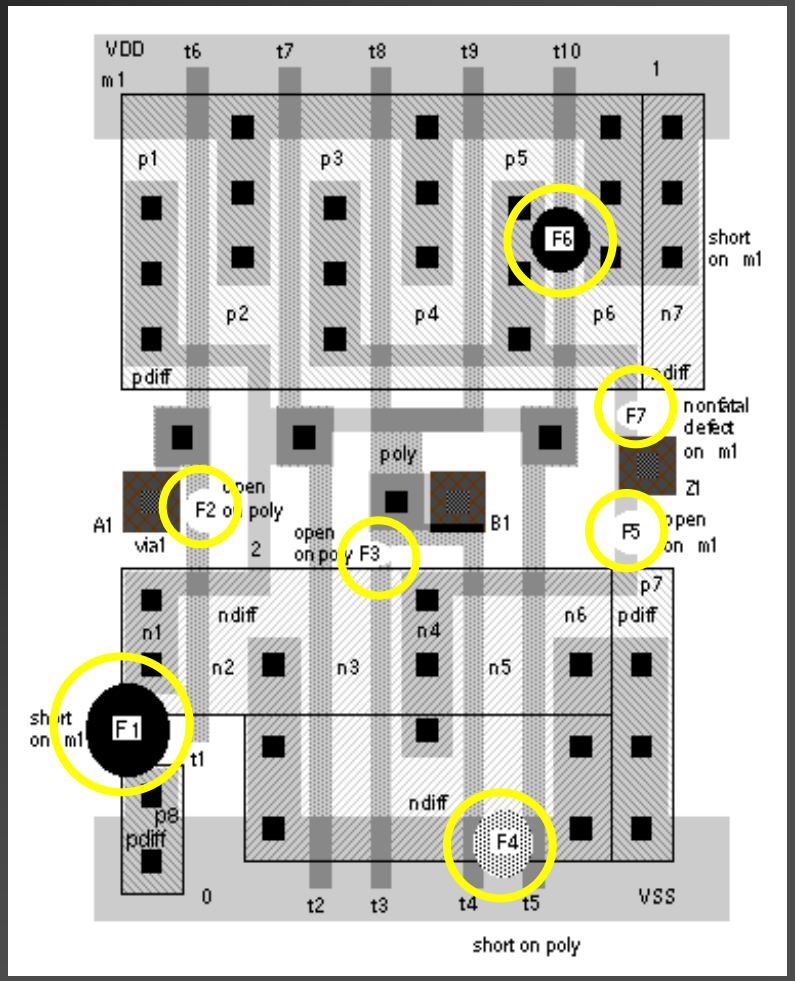
F2 & F3: Unterbrechung Poly
T1.G floatet & T3.G floatet

F4: Kurzschluß Poly
T4.G und T5.G verbunden

F5: Unterbrechung Metallisierg
n4 nicht mehr am Ausgang

F6: Kurzschluß Metallisierung
p5 und p6 verbunden

F7: Einschnürung Metallisierung
p-Stack noch am Ausgang



© A. Steininger / TU Wien

Physikalisches Beispiel /2

F1: Kurzschluß Metallisierung
n1 an VSS

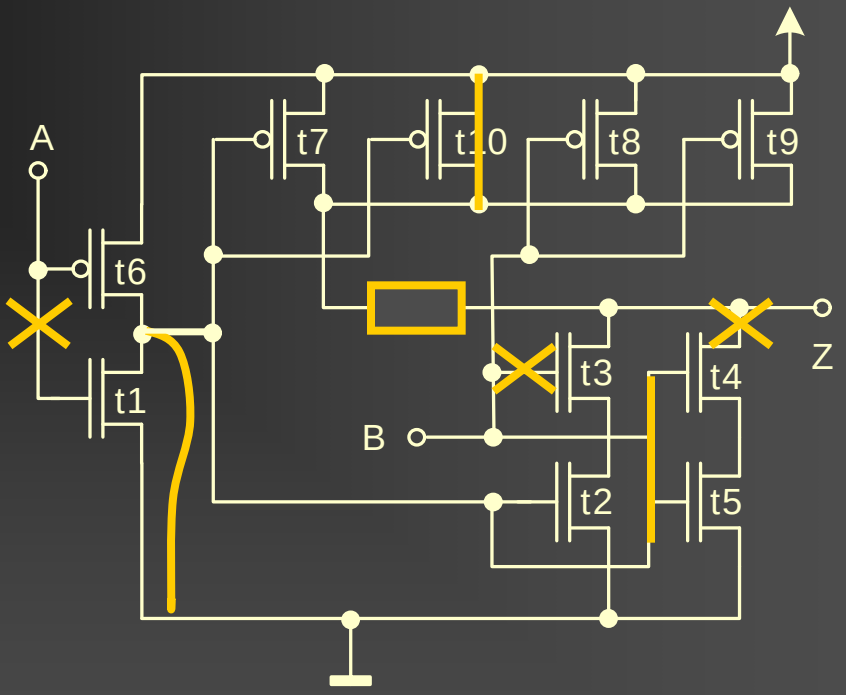
F2 & F3: Unterbrechung Poly
T1.G floatet & T3.G floatet

F4: Kurzschluß Poly
T4.G und T5.G verbunden

F5: Unterbrechung Metallisierg
n4 nicht mehr am Ausgang

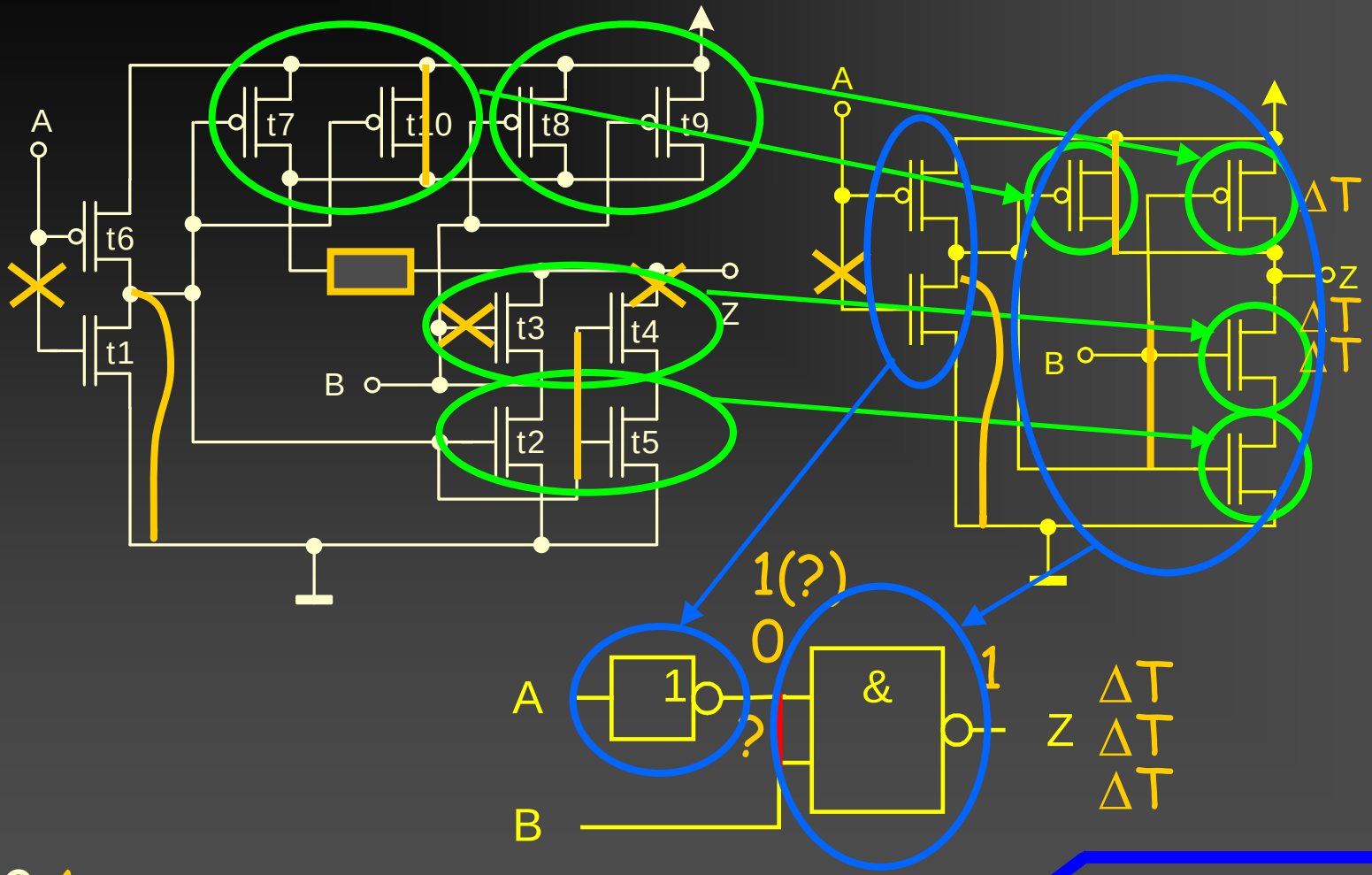
F6: Kurzschluß Metallisierung
p5 und p6 verbunden

F7: Einschnürung Metallisierung
p-Stack noch am Ausgang



© A. Steininger / TU Wien

Physikalisches Beispiel /3



© A. Steininger / TU Wien

Das „Stuck-at" Fehlermodell

Annahmen:

- alle (!) Fehler wirken sich logisch als eine von folgenden zwei Alternativen aus
  - „Festsitzen" eines Knotens auf ´1´:
    stuck-at-1 (SA1)
  - „Festsitzen" eines Knotens auf ´0´:
    stuck-at-0 (SA0)
- es gibt nur einen einzelnen Fehler
  „*single* stuck-at-Modell (SSF)"

© A. Steininger / TU Wien

Das SSF im physikal. Beispiel

- ✓ **F1:** n1 SA0 = Eingang A1 SA1
- ? F2: n1 (wahrscheinlich) SA1 = Eingang A1 SA0
- ☹ F3: pull-down-Stack halbiert
  => doppelte fall-time = Delay fault
- ☹ F4: „bridging fault“,
  Auswirkung abh. von relativen Treiberstärken
- ☹ F5: wie F3
- ✓ **F6:** p5 SA1 = Ausgang Z SA1
- ? F7: Kandidat für burn-in: Elektromigration =>
  pull-up-Stack abgetrennt => Ausgang Z SA0

© A. Steininger / TU Wien

Bewertung des SSF

- Das SSF ist nur für einen kleinen Teil der in der Praxis auftretenden Fehler zutreffend, aber:
- Die Berücksichtigung von Mehrfachfehlern, und anderen Fehlertypen (z.B. „stuck-at open") wäre realistischer, ist aber wesentlich komplizierter
- In der Praxis hat sich gezeigt, daß das SSF „nebenbei" weitaus mehr Fehler aufdeckt als nur „stuck-at"-Faults
- Das Single stuck-at-0/1-Modell hat sich in der Praxis sehr bewährt und wird wegen seines guten Kosten/Nutzen-Verhältnisses gern verwendet

© A. Steininger / TU Wien



Zusammenfassung (1)

- Fehlerquellen sind über den gesamten Lebens-zyklus eines Chips verteilt: vom Design über die Fertigung und Inbetriebnahme bis zur Applikation.
- Designfehler werden durch Simulation entdeckt.
- Ursachen von Defekten bei der Fertigung liegen in Wafer-Material, Lithographie, Entwicklung & Ätzen (Verunreinigungen), Bonding, Packaging,...
- Die Badewannenkurve beschreibt die Verteilung der Ausfälle über die Betriebszeit: Nach einer hohen Ausfallsrate zu Beginn (infant mortality) folgt eine Periode mit konstanter, niedriger Ausfallrate (useful life), danach steigt die Ausfallrate stark an (wear-out).

© A. Steininger / TU Wien



Zusammenfassung (2)

- Häufigste Ursachen für Defekte im Betrieb sind Gate-Oxide-Breakdown, Electromigration und Electrical Overstress.
- Beim Gate-Oxide-Breakdown bilden Störstellen in der extrem dünnen Isolationsschichte des Gate-Oxid einen leitenden Pfad. Solche Störstellen entstehen z.B. durch unreines Material bzw. durch Beanspruchung durch hohe Felder.
- Electromigration ist die Verschiebung von Material durch einen Elektronenwind, der sich bei extrem hoher Stromdichte bildet.
- Gemäß Black's Law sinkt die MTTF quadratisch mit d. Stromdichte und exponentiell mit d. Temperatur

© A. Steininger / TU Wien



Zusammenfassung (3)

- Electrical Overstress ist eine Überbeanspruchung durch zu hohe Spannungen, wie z.B. Spannungsspitzen bzw. Blitzschläge.
- Hohe Temperatur, Temperaturzyklen sowie hohe Spannungen und Ströme wirken sich negativ auf die Lebensdauer eines Chips aus. Ebenso kann unsachgemäßes Handling zur Zerstörung führen (ESD).
- Der Einfluss der Temperatur auf die Lebenserwartung eines Chips wird durch die Arrhenius-Gleichung beschrieben: Temperaturerhöhung bewirkt ein exponentielles Ansteigen der Fehlerrate.

© A. Steininger / TU Wien



Zusammenfassung (4)

- Beim Burn-in versucht man die Phase der infant mortality durch Temperaturzyklen noch vor der Auslieferung zu überwinden.
- Physikalische Defekte können sich in verschiedenster Weise auf die logische Funktion eines Chips auswirken. Typische Manifestationen sind Kontaktprobleme, Isolationsprobleme, parametrische Fehler, dynamische Fehler und Speicherfehler.
- Das Stuck-at Fehlermodell nimmt als Fehlermanifestation an, dass ein Schaltungsknoten auf einem bestimmten Logikpegel „festsitzt". Dieses Modell trifft zwar selten wirklich zu, hat sich in der praktischen Anwendung jedoch bewährt.

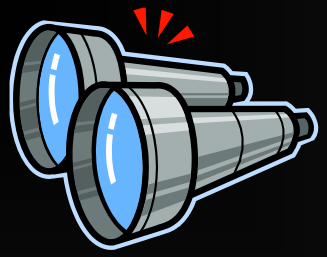
© A. Steininger / TU Wien

Der Test

Ein oft unterschätzter Anteil am Design



© A. Steininger / TU Wien



Überblick

- Bedeutung des Testens
- Test im Lebenszyklus
- Ziele und Qualität von Tests
- Methoden zur Testvektorgenerierung
- Sequentielle Logik & Scan Test
- Speicher & March Tests
- Boundary Scan & Test Access Port
- Built-in Self-Test

© A. Steininger / TU Wien

Kosten eines Defekts

„Rule of Ten“: Die Kosten eines Defekts steigen mit jedem Assemblierungsschritt um eine Größenordnung











© A. Steininger / TU Wien

Testqualität

- Defect level [% bzw. ppm]

  Wie hoch ist der Anteil an nicht ausgeschiedenen fehlerhaften Produkten ?

- Test Coverage [%]

  Wie hoch ist der Anteil an durch den Test erkennbaren Fehlern (bezogen auf alle Fehler im Fehlermodell)



Die Abwesenheit von Fehlern läßt sich grundsätzlich nie beweisen, nur deren Anwesenheit

© A. Steininger / TU Wien

Bedeutung des Fehlermodells

- Das Stuck-at Modell eignet sich i.a. nicht zur Entdeckung von Delay Faults (beim Test nur statische Muster angelegt).
- Selbst mit 100% test coverage können damit daher Delay Faults nicht ausgeschlossen werden.
- In neueren Technologien werden Delay Faults zunehmend problematisch (➡ Defect level!).
- Das Fehlermodell muss daher erweitert und die Tests entsprechend ergänzt werden (gegen-läufige Flanken mit Zeitmessung dazwischen).

© A. Steininger / TU Wien

Zusammenhang d. Kenngrössen

fault coverage	*avg. defect level*	*avg. quality*
50%	7%	93%
90%	3%	97%
95%	1%	99%
99%	0.1%	99.9%
99.9%	0.01%	99.99%

(experimentelle Messergebnisse nach [Smith], stuck-at Fehlermodell)

schon ein einfacher Test erkennt sehr viele Defekte, die verbleibenden Defekte erfordern jedoch ungleich mehr Testaufwand

© A. Steininger / TU Wien

Beispiel zum Defect Level

ASIC X: Preis 10 €; Defect-Level = d

PCB-Board Y: 100.000 Stück, jedes enthält 1 ASIC X
Austausch eines defekten Chip kostet 200 €.

defect level d	*defekte ASICs*	*Reparaturkosten*
0.01 %	10	2.000 €
0.1 %	100	20.000 €
1.0 %	1.000	200.000 €
5.0 %	5.000	1.000.000 €

Vergleiche: Wert der 100.000 ASICs = 1.000.000 €

© A. Steininger / TU Wien

Kosten auf Systemebene

ASIC X: Preis 10 €; Defect-Level = d

Computer Z: 100.000 Stück, jeder enthält 1 ASIC X
Ausfall + Reparatur kosten 10.000 €.
Integrationstest erkennt 90% der Defekte

defect level d (ASIC)	*Defekte bei*		*Reparaturkosten (Computer)*
	ASIC	*Computer*	
0.01 %	10	1	10.000 €
0.1 %	100	10	100.000 €
1.0 %	1000	100	1.000.000 €
5.0 %	5000	500	5.000.000 €

© A. Steininger / TU Wien

Kosten eines Systemausfalls

- „Die Stehzeit eines industriellen Systems kostet im Mittel 1300$ je Minute“
  (Studie: USA, 1995)
- Extrembeispiel: 9 Stunden Ausfall kosten AT&T 60 Mio. $
  (USA, 1990)
- weitere Beispiele gibt es aus Raum- und Luftfahrt, Börsewesen, Medizin, etc.

© A. Steininger / TU Wien

Test in allen Lebensphasen ...

- Factory-Test (Process-Feedback, Self-Repair)
- Eingangstest (Total Quality Management)
- Systemintegration (IP core/SOC, Chip, Board,...)
- System-Startup (redundante Komponenten)
- On-line Test (Fehlererkennung)
- Diagnose und Wartung (Fernwartung, Error-Log)

© A. Steininger / TU Wien

Zielsetzungen für einen Test

- Defect Detection
  - frühzeitige Erkennung
- Defect Location (Diagnose)
  - Reparatur / Ersatz durch „Spare“
  - Prozess-Feedback, Problem-Identifikation
  - Klärung der Verantwortlichkeit

© A. Steininger / TU Wien

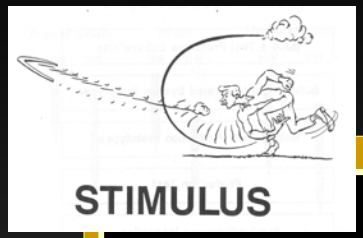
Grundprinzip des Testens



[Agilent]

© A. Steininger / TU Wien

Ablauf eines Tests





test pattern

response

Test Pattern Generator

Circuit under Test

Response Analysis

execute

next pattern

Test Controller

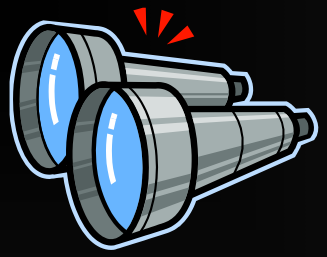
pass/fail

Tester

start test

accept/reject

© A. Steininger / TU Wien

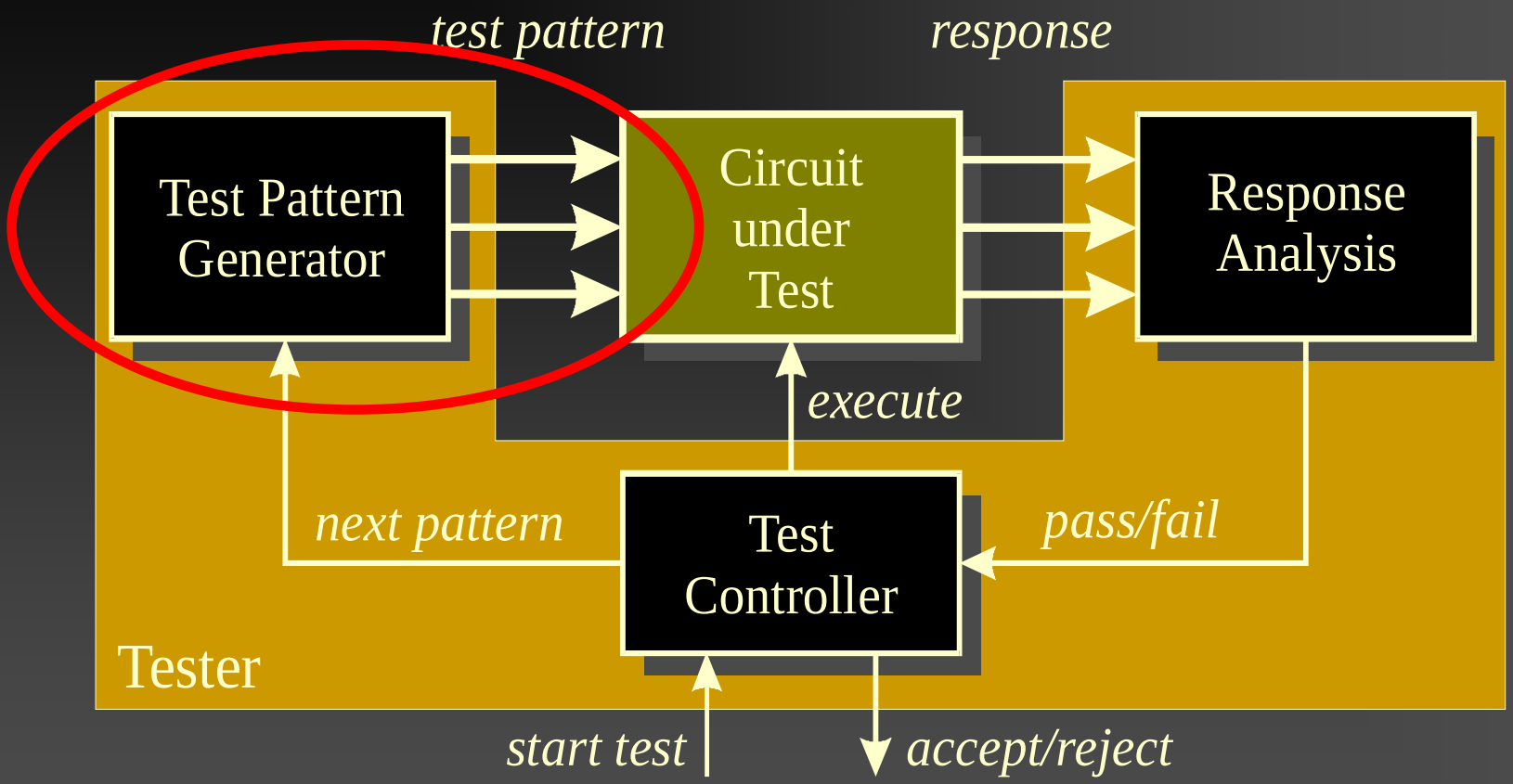


Überblick

- Bedeutung des Testens
- Test im Lebenszyklus
- Ziele und Qualität von Tests
- Methoden zur Testvektorgenerierung
- Sequentielle Logik & Scan Test
- Speicher & March Tests
- Boundary Scan & Test Access Port
- Built-in Self-Test

© A. Steininger / TU Wien

Wahl der Testvektoren



© A. Steininger / TU Wien

Testvektorgeneration: Optionen

- Exhaustive Testing
  - alle möglichen Vektoren (Bitkombinationen)
- Deterministic Testing
  - algorithmische Suche nach optimalem Testprogramm
- Nondeterministic Testing
  - Folge von (pseudo-)zufälligen Vektoren bildet das Testprogramm

© A. Steininger / TU Wien

Exhaustive Testing

- Anlegen aller möglichen Eingangsmuster
- Die Funktion wird vollständig überprüft
- Testaufwand für kombinatorische Logik:
  bei *n* Eingängen ⇒ 2^n Testvektoren
- Beispiel: 100MHz Takt
  - 32 Eingänge ⇒ 43s Testdauer
  - 40 Eingänge ⇒ 3h Testdauer
  - 64 Eingänge ⇒ 5.8 Jahrhunderte

© A. Steininger / TU Wien

Testvektorgeneration: Optionen

- ✓ Exhaustive Testing
  - alle möglichen Vektoren (Bitkombinationen)
- Deterministic Testing
  - algorithmische Suche nach optimalem Testprogramm
- Nondeterministic Testing
  - Folge von (pseudo-)zufälligen Vektoren bildet das Testprogramm

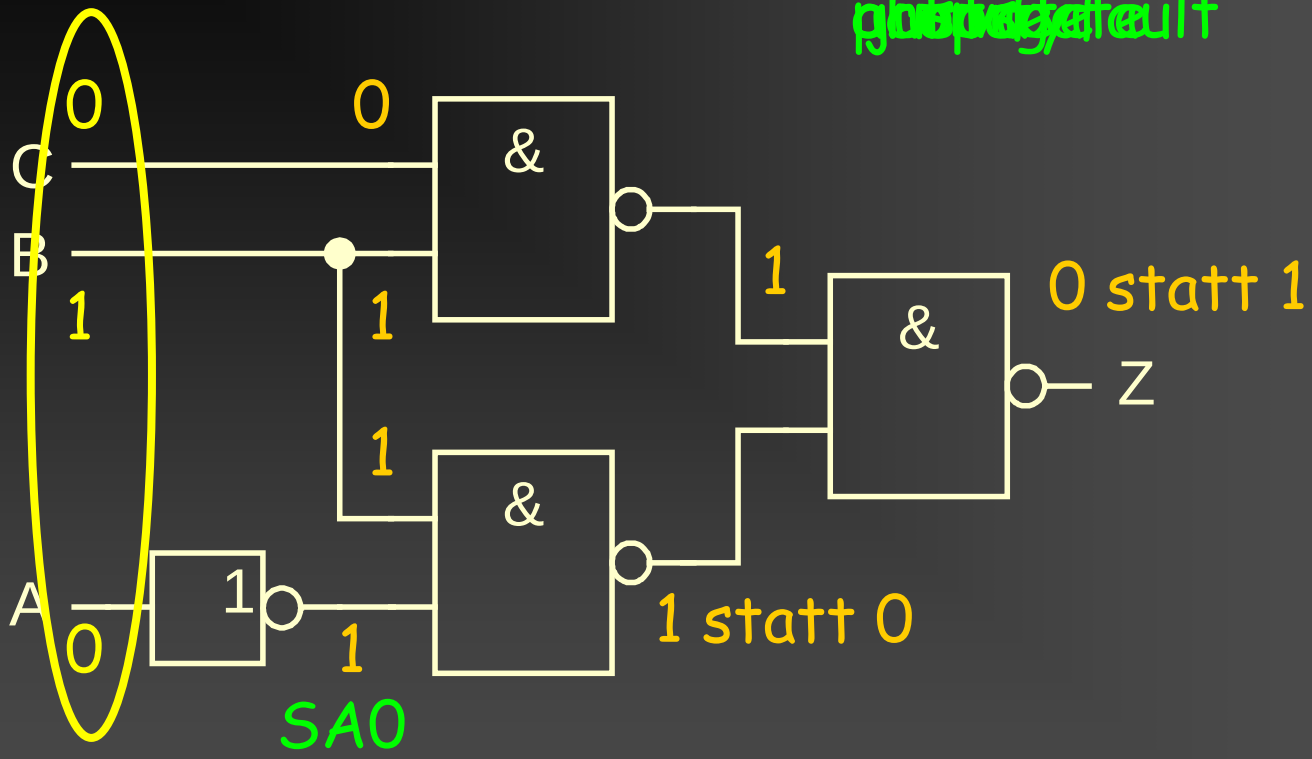
© A. Steininger / TU Wien

Deterministic Testing: Prinzip

- Annahme eines Fehlers (z.B. SA0)
- Aktivierung (Activation)
  z.B.: einzustellen ist ´1´ im fehlerfreien Zustand
  Einstellen (Justification)
  Bedingungen für Primary Inputs (PIs) ermitteln
- Weiterleiten (Propagation)
  zu den Primary Outputs über „sensitized path“,
  Einstellen (Justification)
  Bedingungen für Primary Inputs (PIs) ermitteln

© A. Steininger / TU Wien

Deterministic Testing: Beispiel



Zugehöriger Testvektor = (0,1,0)

© A. Steininger / TU Wien

Fehlersimulation

- Liste aller Fehler lt. Fehlermodell erstellen
- Reduktion der Liste durch
  - Elimination äquivalenter Fehler
  - Fehlerdominanz, etc.
- Sukzessives Abarbeiten der Liste
  - Testvektor für obersten Eintrag ermitteln
  - Streichen dieses Eintrags von der Liste
  - Streichen weiterer Einträge, die durch diesen Testvektor abgedeckt werden (TV Compaction)

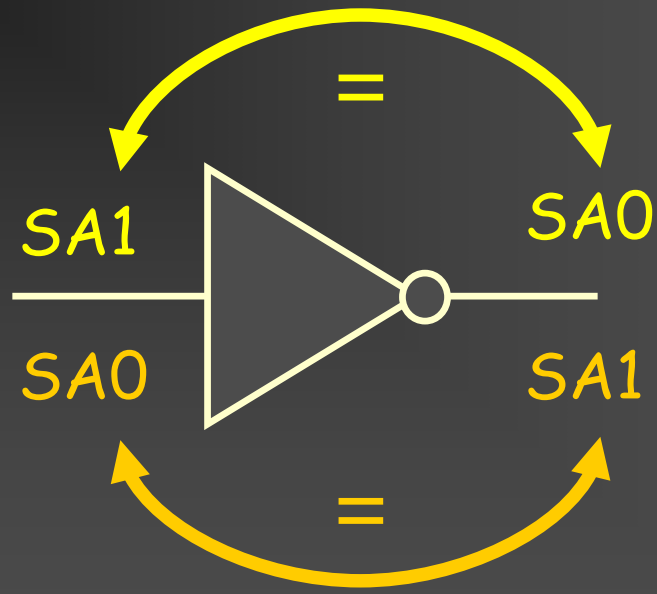


© A. Steininger / TU Wien

Äquivalente Fehler

Fehler an unterschiedlichen Stellen, die dennoch zur gleichen logischen Wirkung führen

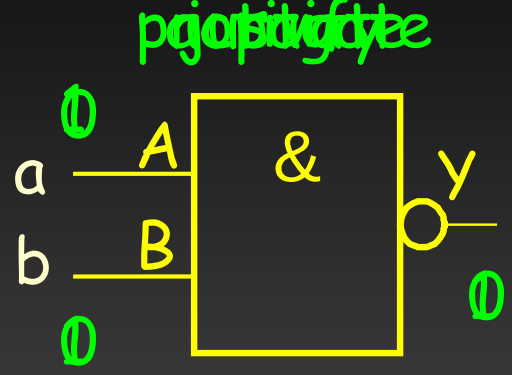
Beispiel: Inverter



© A. Steininger / TU Wien

Fehlerdominanz

- Beispiel NAND:



Testvektoren (a,b):

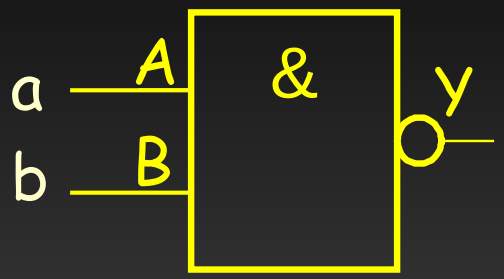
	SA0	SA1
A	(1,1)	(0,1)
B	(1,1)	(1,0)
Y	(0,1) (1,0) (0,0)	(1,1)

- ✓ A/SA0, B/SA0 und Y/SA1 sind äquivalent, d.h. im Test nicht unterscheidbar (gleiche Vektoren)
- ✓ Y/SA0 dominiert A/SA1 (bzw. B/SA1) : Y/SA0 wird von jedem Testmuster für A/SA1 mitentdeckt, aber nicht umgekehrt

© A. Steininger / TU Wien

Test Vector Compaction

- Beispiel NAND:



Testvektoren (a,b):

	SA0	SA1
A	(1,1)	(0,1)
B	(1,1)	(1,0)
Y	(0,1) (1,0) (0,0)	(1,1)

Vec	Detection	Coverage	Cumul. coverage
(1,1)	A/SA0, B/SA0, Y/SA1	3/6 = 50%	50%
(1,0)	B/SA1, Y/SA0	2/6 = 33%	83%
(0,1)	A/SA1, Y/SA0	2/6 = 33%	100%
(0,0)	Y/SA0	1/6 =17%	100%

© A. Steininger / TU Wien

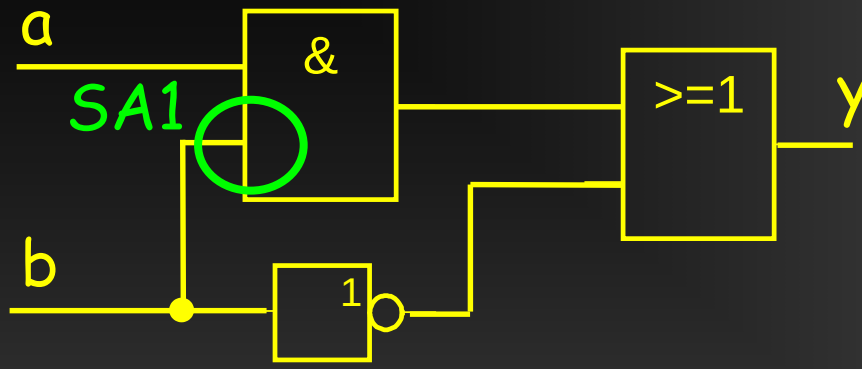
Verbleibende Probleme

- Abdeckung von "hard-to-detect" Faults erfordert weit überproportionalen Suchaufwand

  => vernünftige Kompromisse bei der Coverage
- Auflösung widersprüchlicher Bedingungen für die PIs schwierig und manchmal unmöglich.
- Fehler in redundanter Logik sind prinzipiell nicht erkennbar.
- Aufwand für die Testmustersuche steigt mit der 2. bis 3. Potenz der Gatterzahl.

© A. Steininger / TU Wien

Redundante Logik



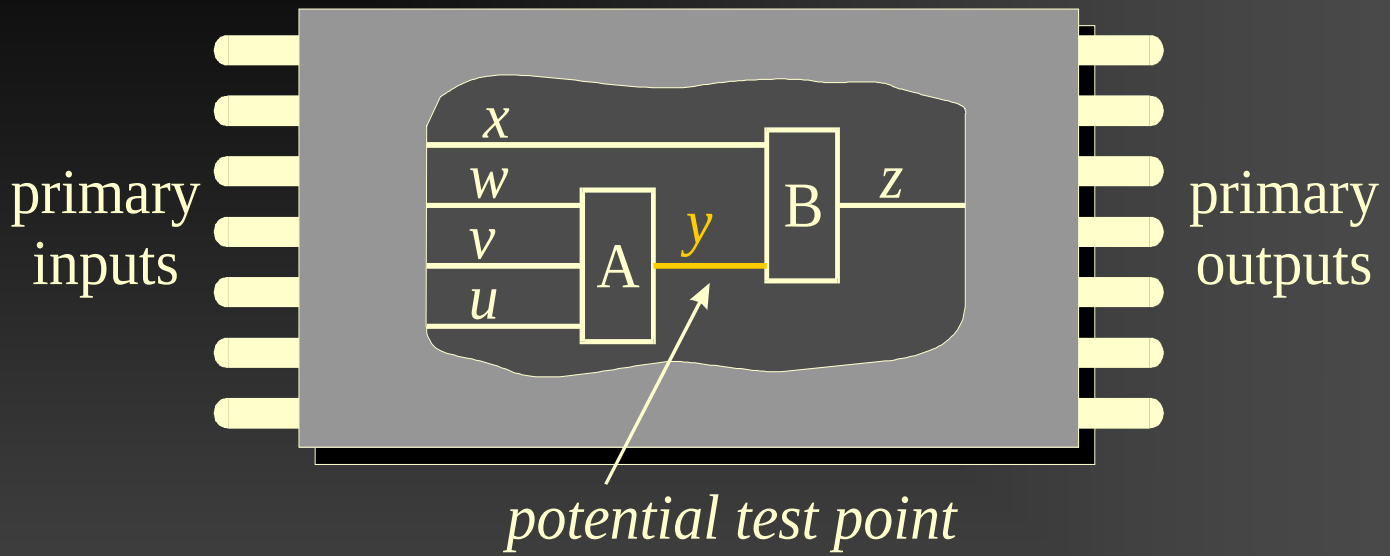
$$y = (a \wedge b) \vee \neg b$$
$$= (a \vee \neg b) \wedge (b \vee \neg b)$$
$$= a \vee \neg b$$

Die AND-Verknüpfung ist logisch redundant und lässt sich wegkürzen. Ein SA1-Fault am unteren Eingang des AND bleibt wirkungslos und kann folglich von einem Test nicht detektiert werden.

Redundante Logik erschwert die Testvektorgeneration und verschlechtert (scheinbar) die Test Coverage.

© A. Steininger / TU Wien

Der Begriff der Testbarkeit



Controllability (Steuerbarkeit) von y über PIs

Observability (Beobachtbarkeit) von y an POs

© A. Steininger / TU Wien

Testvektorgeneration: Optionen

- ✓ Exhaustive Testing
  - alle möglichen Vektoren (Bitkombinationen)
- ✓ Deterministic Testing
  - algorithmische Suche nach optimalem Testprogramm
- ▲ Nondeterministic Testing
  - Folge von (pseudo-)zufälligen Vektoren bildet das Testprogramm

© A. Steininger / TU Wien

Nondeterministic Testing

- pseudozufällige (wiederholbare!) Folge von Testvektoren
- führt für vernünftige Vorgaben der Coverage sehr rasch zum Ergebnis
- einfach auch per HW generierbar (LFSR)
- bessere Erkennung von "nontarget" Faults
- Zufallsfolge kann so gewählt werden, dass sie vorgegebene deterministische Vektoren enthält ("pseudo-deterministic")
- Problem: Vermeidung unerwünschter Vektoren (z.B. gleichzeitige Aktivierung mehrerer Bustreiber)

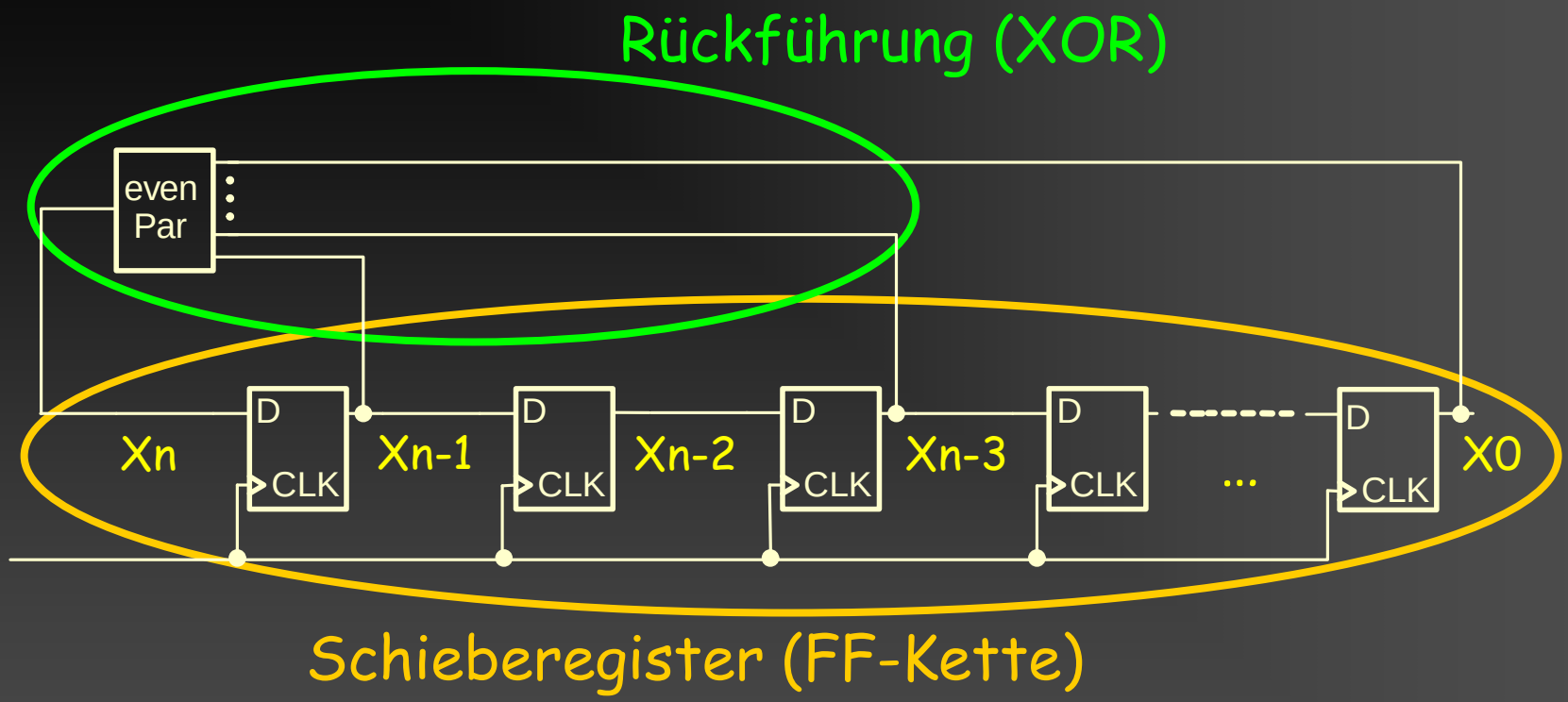
© A. Steininger / TU Wien

Linear Feedback Shift Register

- Aufbau wie Schieberegister (unaufwändiger als Zähler!), jedoch Rückführung über XOR.
- Durchläuft periodisch best. Sequenz von Mustern
- Eingänge des XOR durch „Polynom" beschrieben.
- Wahl des Polynoms ist kritisch für Periodizität („maximum length sequence")
- Verw. als Pseudo-Zufallsgenerator f. Testmuster
- Durch geschickte Wahl von Polynom und Startwert ("Seed") bzw. "reseeding" lassen sich vorgegebene Vektoren inkludieren.

© A. Steininger / TU Wien

LFSR: Implementierung



Beispiel: $Xn = Xn\text{-}1 \oplus Xn\text{-}3 \oplus \ldots \oplus X0$

© A. Steininger / TU Wien

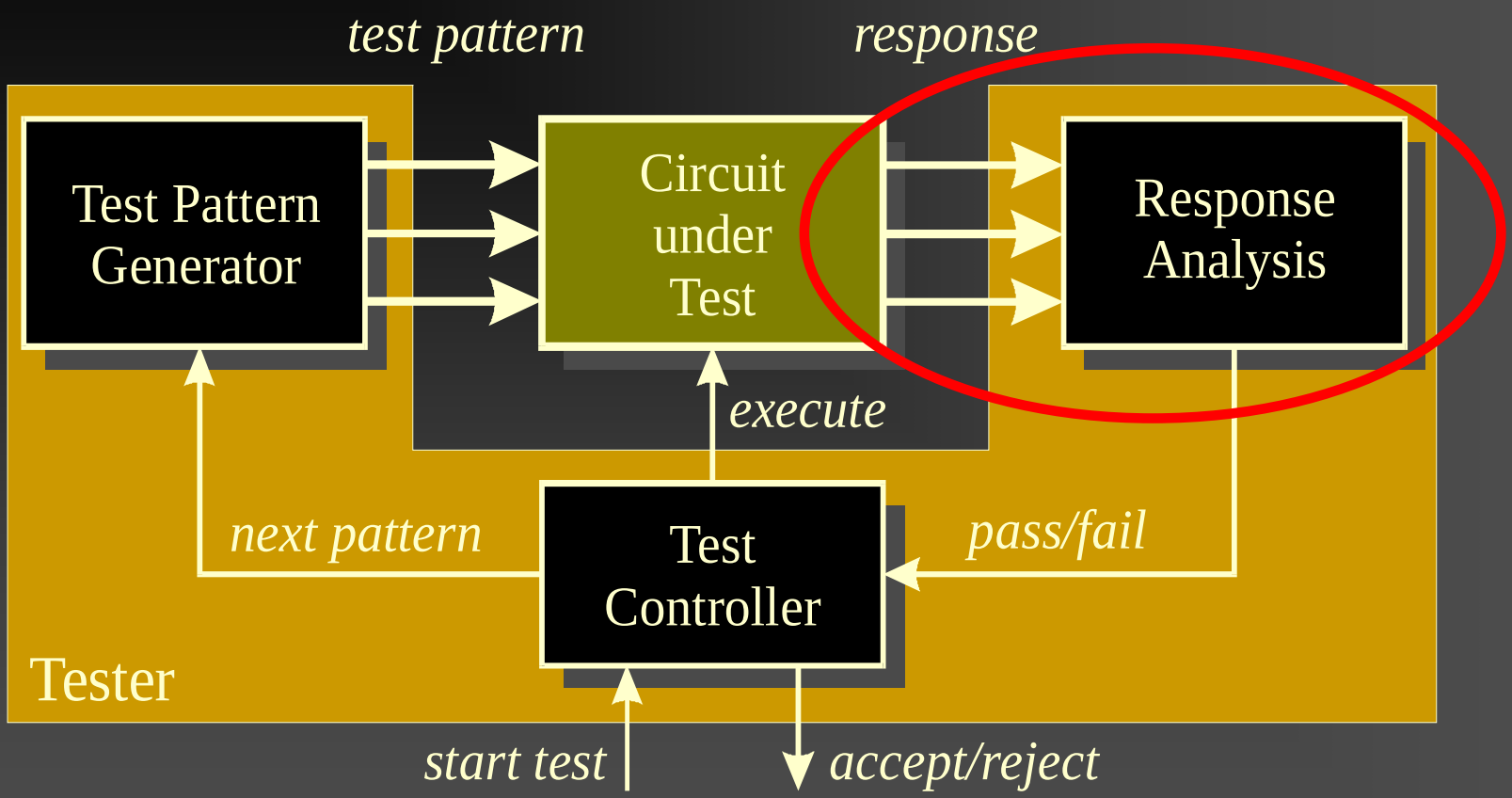
LFSR: Beispiel für CRC16

X16	X15	X14	X13	X12	X11	X10	X9	X8	X7	X6	X5	X4	X3	X2	X1	X0
1	1	1	0	0	0	1	0	1	1	0	1	1	0	0	1	1
1	1	1	1	0	0	0	1	0	1	1	0	1	1	0	0	1
0	1	1	1	1	0	0	0	1	0	1	1	0	1	1	0	0
0	0	1	1	1	1	0	0	0	1	0	1	1	0	1	1	0
1	0	0	1	1	1	1	0	0	0	1	0	1	1	0	1	1
1	1	0	0	1	1	1	1	0	0	0	1	0	1	1	0	1
1	1	1	0	0	1	1	1	1	0	0	0	1	0	1	1	0
0	1	1	1	0	0	1	1	1	1	0	0	0	1	0	1	1
1	0	1	1	1	0	0	1	1	1	1	0	0	0	1	0	1
1	1	0	1	1	1	0	0	1	1	1	1	0	0	0	1	0
1	1	1	0	1	1	1	0	0	1	1	1	1	0	0	0	1
1	1	1	1	0	1	1	1	0	0	1	1	1	1	0	0	0
1	1	1	1	1	0	1	1	1	0	0	1	1	1	1	0	0

CRC16: $X16 = X5 \oplus X4 \oplus X3 \oplus X0$

© A. Steininger / TU Wien

Ablauf eines Tests

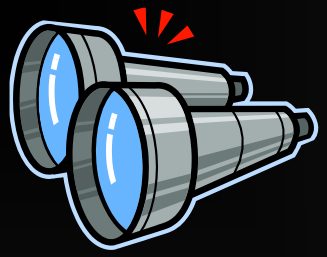


© A. Steininger / TU Wien

Auswertung der Reaktion

- Für jeden Testvektor wird das Verhalten des Testobjekts überprüft.
- Die Referenzdaten dazu erhält man aus den Simulationen während des Design.
- Abweichungen der beobachteten Reaktion von den Referenzdaten weisen auf einen Fehler hin.
- Analyse und Interpretation dieser Abweichungen erlauben eine Fehlereingrenzung (=> Diagnose).
- Für die Speicherung tausender Testvektoren und der zugehörigen Referenz-Responses benötigt ein Tester massiven Speicherplatz.

© A. Steininger / TU Wien



Überblick

- Bedeutung des Testens
- Test im Lebenszyklus
- Ziele und Qualität von Tests
- Methoden zur Testvektorgenerierung
- Sequentielle Logik & Scan Test
- Speicher & March Tests
- Boundary Scan & Test Access Port
- Built-in Self-Test

© A. Steininger / TU Wien

Kombinatorisch vs. Sequentiell

- kombinatorische Logik
  - ✓ vollständig durch zeitinvariante Wahrheits-tabelle beschrieben
  - ✓ ein einziger Testvektor je Fehler nötig
  - ✓ bei n Eingängen sind max. 2^n Testvektoren nötig
- sequentielle Logik
  - ☹ enthält speichernde Elemente (FFs, Latches)
  - ☹ Fehlererkennung erfordert bestimmten Testvektor in bestimmtem Zustand
  - ☹ daher deutlich mehr Testvektoren nötig

© A. Steininger / TU Wien

Sequentielle Logik: Probleme

- Vor dem Anlegen jedes Testvektors muss die Logik in einen bestimmten Zustand gebracht werden, aber
  - Die Identifikation des aktuellen Zustandes ist manchmal schwierig (wirkt der Reset auf alle Speicherelemente ?)
  - Das Einstellen des gewünschten Zustandes erfordert meist mehrere Testvektoren
- Der Testaufwand wächst daher exponentiell (!) mit der Anzahl der Zustände.

© A. Steininger / TU Wien

Sequentielle Logik: Beispiel

- Problemstellung:

  Das Carry-Bit eines 32-Bit Counters ist zu testen.

- Lösung:

  Der Zähler wird mittels Reset auf "0" initialisiert (Zustandsidentifikation wird dadurch trivial), danach müssen $2^{32} \approx 4*10^9$ Taktzyklen abgewartet werden, bis der Zähler überläuft. Bei 100 MHz Takt sind das 40 Sekunden. Bei 0.25€ Kosten pro Sekunde Testzeit sind das über 10€ je Chip ! Damit ist erst ein einzelnes Feature getestet.

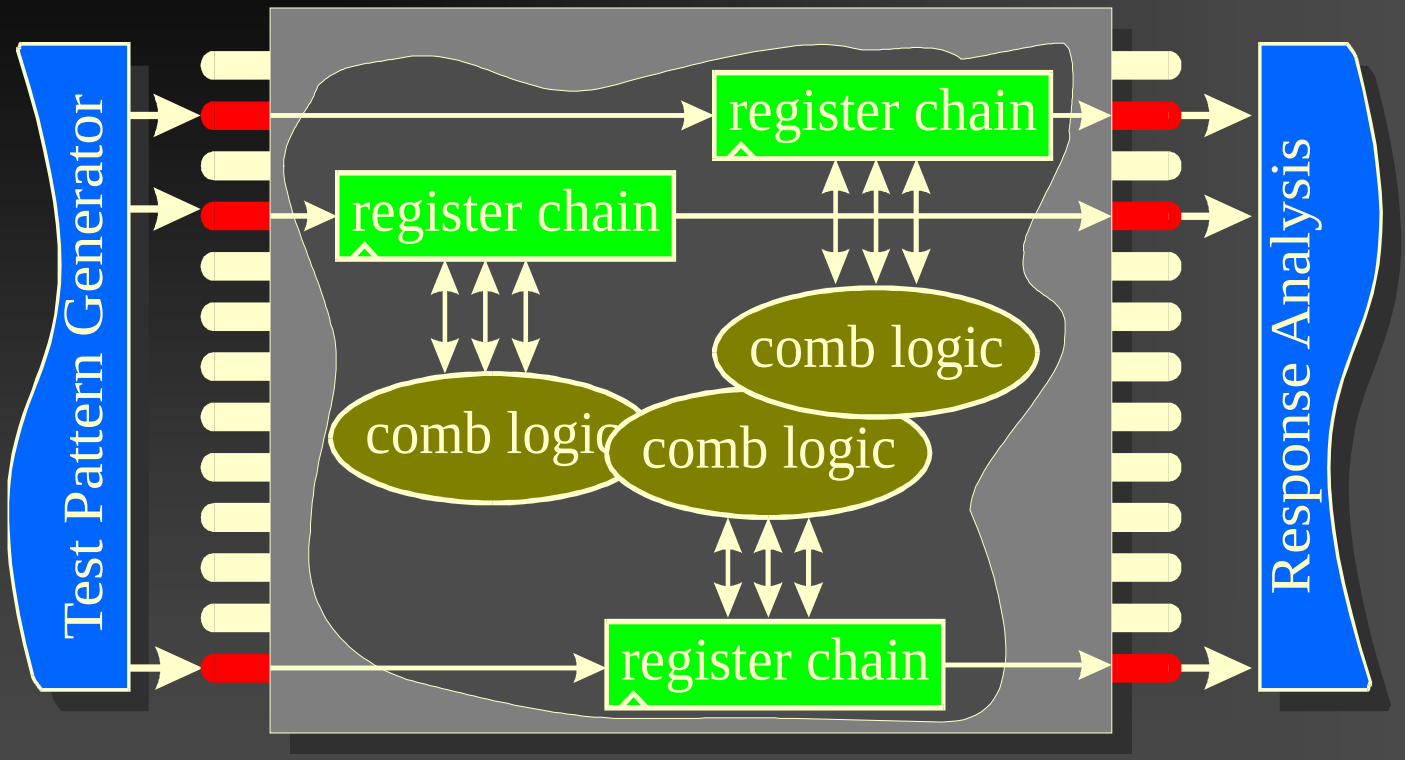
© A. Steininger / TU Wien

Functional vs. Structural Test

- Funktionaler Test
  - System wird als Black-Box betrachtet
  - Es wird unmittelbar überprüft, ob das System seine Funktion lt. Spezifikation erfüllt.
  - Für komplexe Funktionen nicht durchführbar
- Struktureller Test
  - System wird als Kollektiv logischer Primitive (Gatter, FFs) betrachtet.
  - Indirekter Funktionsnachweis: Funktionieren alle Bestandteile, funktioniert auch das System.

© A. Steininger / TU Wien

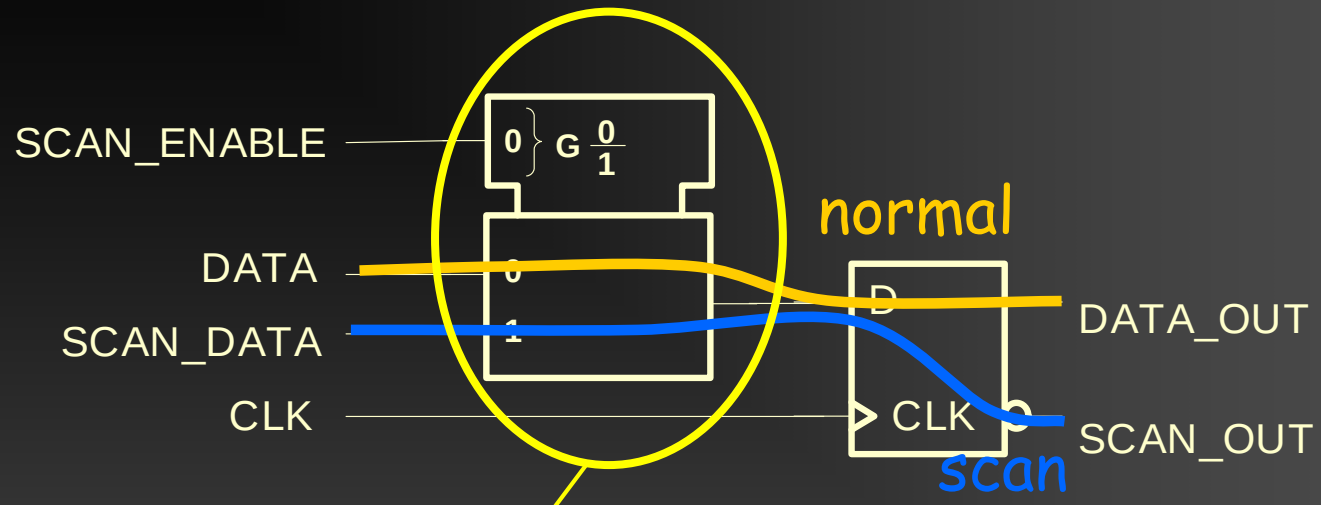
Der Scan-Pfad



Speicherelemente werden zu Schieberegistern verbunden

© A. Steininger / TU Wien

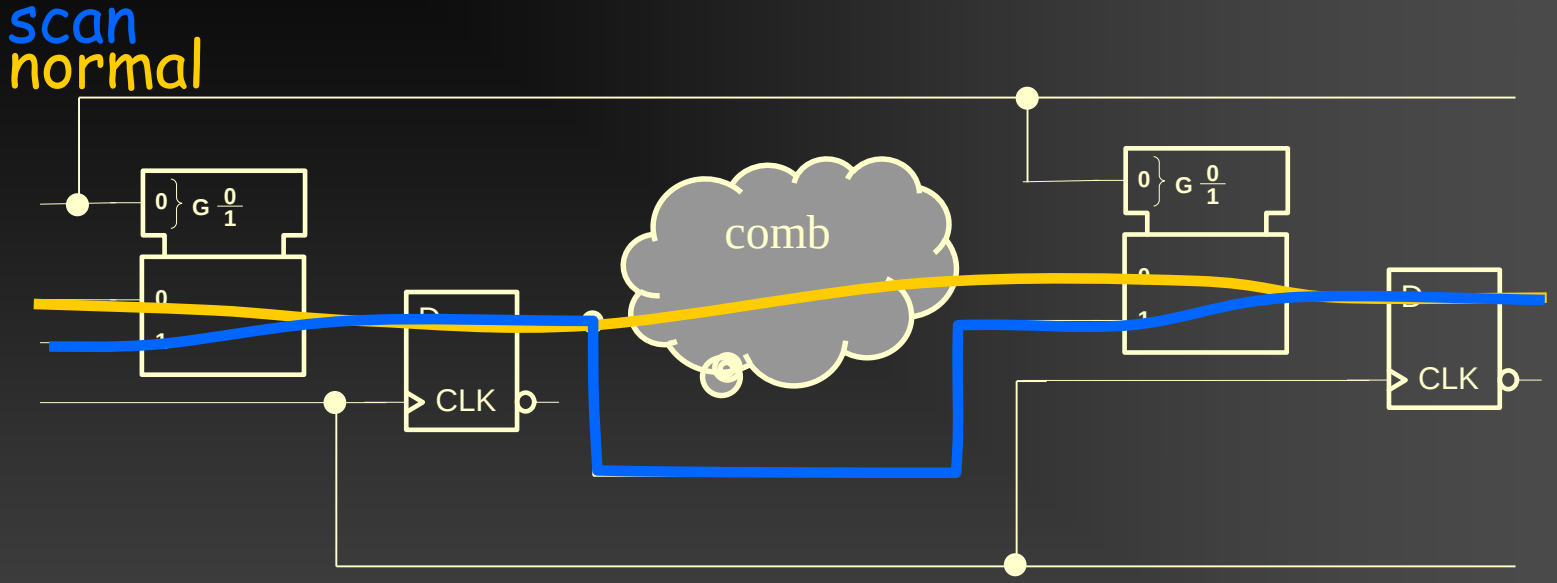
Scan-Register: Overhead



- zusätzlicher MUX im Signalpfad
  - ca. 10% mehr Fläche je FF
  - Verlängerung der kritischen Pfades
  - Verdrahtungsaufwand

© A. Steininger / TU Wien

Bildung des Scan-Pfad



Im Scan-Mode schaltet der Multiplexer den Eingang des FF direkt an den Ausgang eines anderen FF und bildet so ein Schieberegister. Die kombinatorische Logik wird umgangen.

© A. Steininger / TU Wien

Ablauf eines Scan-Test

- Aktivierung des Testmodus
  - Multiplexer verbinden die FFs zu Scan-Chain
- Testvektor anlegen
  - Testvektor seriell in die Scan-Chain shiften
- Testvektor anwenden
  - gewünschten Verarbeitungsschritt (z.B. 1 x takten) im Normalmodus mit geeigneten Eingangsmustern an den Pins durchführen
- Response auslesen
  - Zustände der FFs seriell über Scan-Chain auslesen

© A. Steininger / TU Wien

Vorteile des Scan-Tests

- Reduktion des sequentiellen Testproblems auf ein kombinatorisches Problem
- direktes Setzen interner Zustände möglich (Identifikation und Initialisierung entfallen)
- jedes FF wird zu einem virtuellen Testpunkt („pseudoprimary" inputs & outputs) => gute Testbarkeit
- Scan-chain ist einfach automatisch generierbar
- Zugriff auf die Registerkette über wenige Pins
- Problem: Dauer des Shift bei langen Scan-Chains

© A. Steininger / TU Wien

Scan-Test: Varianten

- Full scan
  - alle FFs werden in die Scan-Chain eingebunden
- Partial scan
  - einige FFs nicht in die Scan-chain eingebunden
  - geringerer Verbindungsaufwand, kürzere Chain
  - es verbleiben sequentielle Anteile
- Multiple Scan-chains
  - Aufteilung auf mehrere (disjunkte) Scan-chains
  - rascheres Ein- und Auslesen, mehr Testpins

© A. Steininger / TU Wien

Der IDDQ-Test

statischer Stromverbrauch in CMOS ist extrem klein (Sperrströme, Pull-ups,...)



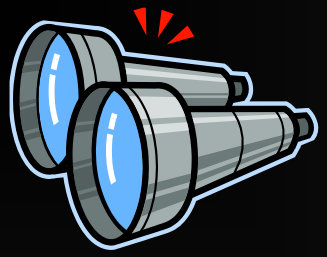
ungewöhnlich hoher statischer Stromverbrauch als Indikator für Fehler

IDDQ

Versorgungsstrom IDD — im Ruhezustand („Quiescent")

Abhängigkeit vom Testvektor bleibt bestehen !

© A. Steininger / TU Wien



Überblick

- Bedeutung des Testens
- Test im Lebenszyklus
- Ziele und Qualität von Tests
- Methoden zur Testvektorgenerierung
- Sequentielle Logik & Scan Test
- Speicher & March Tests
- Boundary Scan & Test Access Port
- Built-in Self-Test

© A. Steininger / TU Wien

Fehlermodelle für Speicher

- Stuck-at Fault
  - Bitzelle "sitzt fest" auf 0 oder 1
- Transition Fault
  - Zustandswechsel der Bitzelle in eine Richtung (↑ oder ↓) ist nicht möglich
- Coupling Fault
  - Zustand (Aktivität) einer Zelle beeinflusst eine andere: z.B.: Kurzschluss zweier Zellen
  - Sonderfall: Neighborhood pattern sensitive fault

© A. Steininger / TU Wien

Test von Speicherblöcken

- Scan-Test wäre viel zu aufwendig
  - Extrem viele Speicherzellen, dichte Struktur
- Funktionaler Test ist einfach möglich
  - Einfache Funktion: Daten schreiben bzw. lesen
- March-Tests sind besonders effizient
  - finden auch komplizierte Fehler Coupling Faults
  - Testzeit wächst nur linear mit Speichergröße
  - einfach in HW implementierbar (Zähler & FSM)

© A. Steininger / TU Wien

March-Test: ein Beispiel

- March C- Algorithmus
  (am Beispiel eines N x 1 –Speicher)

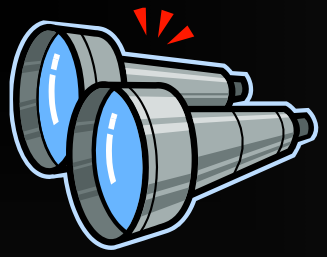
  (1) initialisiere alle N Speicherzellen mit 0
  (2) überprüfe die 0 und schreibe 1; sequentiell für alle Zellen, in ansteigender Adressfolge
  (3) überprüfe die 1 und schreibe 0; sequentiell für alle Zellen, in ansteigender Adressfolge
  (4) überprüfe die 0 in allen Zellen
  (5) - (7) wiederhole (2) – (4); diesmal in abfallender Adressfolge

© A. Steininger / TU Wien

Test von PLDs /FPGAs

- Nach der Fertigung
  - Grundfunktionen werden vollständig getestet.
  - Programmierbarkeit ist nicht testbar bei OTP
- Nach der Programmierung
  - Korrekte Programmierung wird beim Download getestet (Rücklesen)
  - Korrektheit des Designs muss der Anwender sicherstellen (Simulation)
  - Programmierter Baustein ist genau so testbar wie ein nicht programmierbarer.

© A. Steininger / TU Wien



Überblick

- Bedeutung des Testens
- Test im Lebenszyklus
- Ziele und Qualität von Tests
- Methoden zur Testvektorgenerierung
- Sequentielle Logik & Scan Test
- Speicher & March Tests
- Boundary Scan & Test Access Port
- Built-in Self-Test

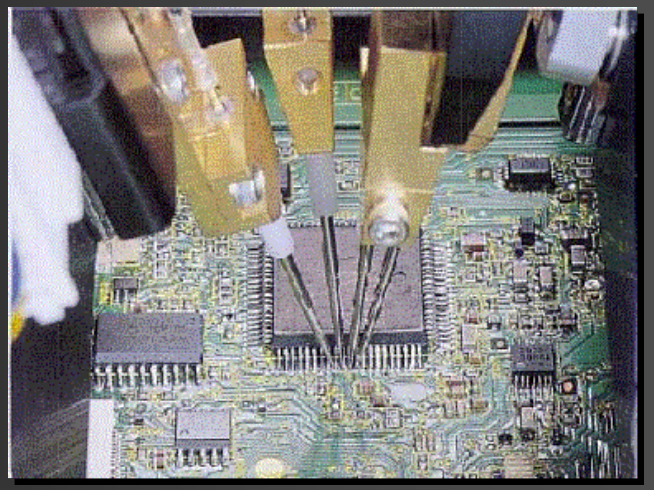
© A. Steininger / TU Wien

Test in allen Lebensphasen ...

- ✓ Factory-Test (Process-Feedback, Self-Repair)
- Eingangstest (Total Quality Management)
- **Systemintegration** (IP core/SOC, Chip, Board,...)
- System-Startup (redundante Komponenten)
- On-line Test (Fehlererkennung)
- Diagnose und Wartung (Fernwartung, Error-Log)

© A. Steininger / TU Wien

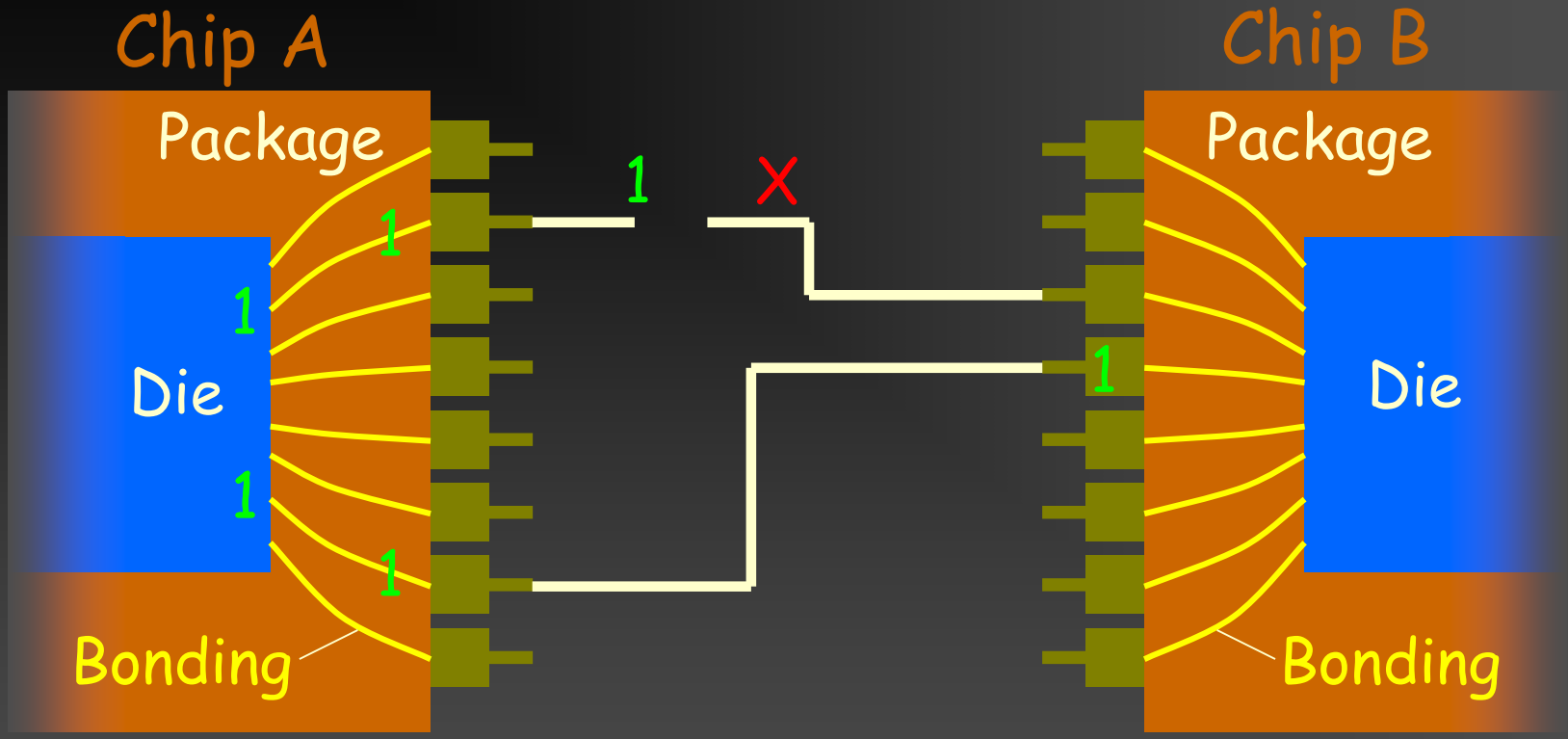
Das Testen von Leiterplatten



- Unbestückte Leiterplatte
  - „Nadelbettadapter"
- Bestückte Leiterplatte
  - Nadelbettadapter (Zugänglichkeit SMD, PGA ?)
  - Funktionstest (Coverage ? Diagnose ?)
  - „Flying Probes" (siehe Bild; Aufwand !)
  - Boundary-Scan

© A. Steininger / TU Wien

Boundary-Scan: Prinzip



Testmuster werden von Chip zu Chip gesendet

➡ Test der Verbindungen

© A. Steininger / TU Wien

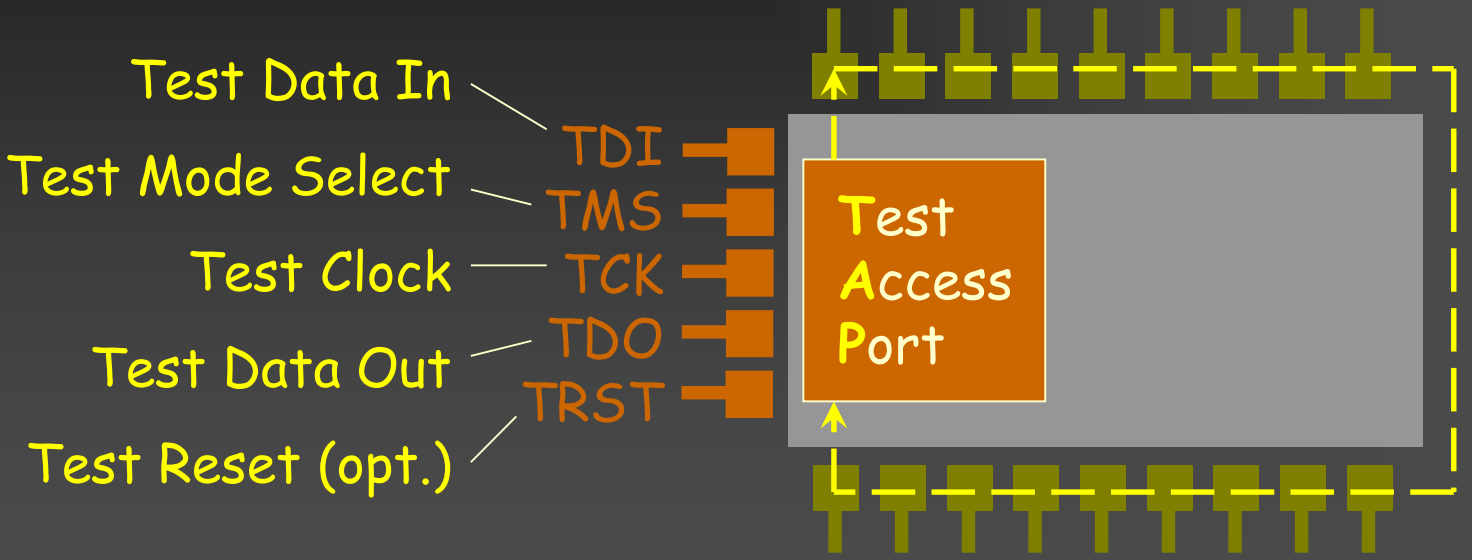
Boundary-Scan: I/O Zellen

- Verwendung spezieller „DR-cells“ an jedem Pin erlaubt
  - transparenten Betrieb (Normalbetrieb)
  - Betrieb als Schieberegister zur Nachbarzelle
  - Ausgabe von Daten
  - Einlesen von Daten
- Spezielle Zellen zur Unterstützung des Boundary Scan sind in Libraries verfügbar bzw. kann Boundary-Scan von Design-Tools automatisch implementiert werden.

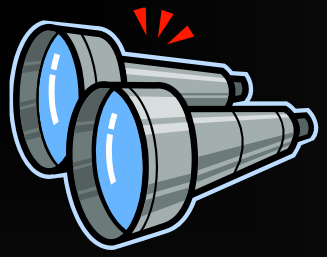
© A. Steininger / TU Wien

Der JTAG Test Access Port

- TAP Controller nach IEEE 1149.1
  - Realisierung als FSM mit 16 Zuständen
  - Standard-Interface aus 4 Pins (5. Pin optional)



© A. Steininger / TU Wien



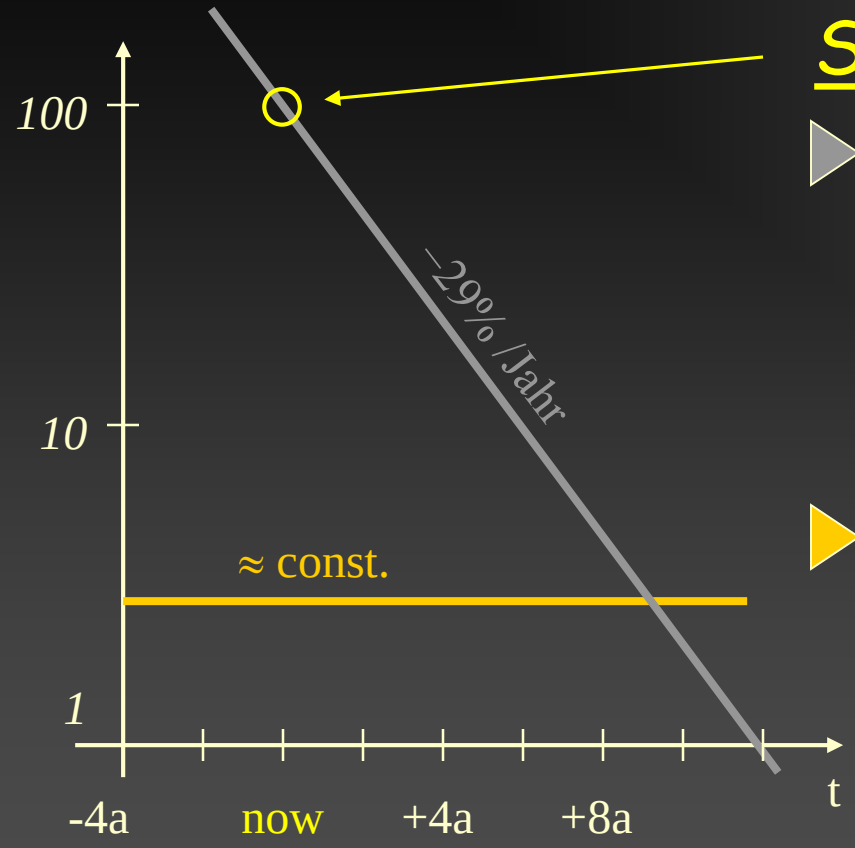
Überblick

- Bedeutung des Testens
- Test im Lebenszyklus
- Ziele und Qualität von Tests
- Methoden zur Testvektorgenerierung
- Sequentielle Logik & Scan Test
- Speicher & March Tests
- Boundary Scan & Test Access Port
- Built-in Self-Test

© A. Steininger / TU Wien

Wiederholung

Factory-Test: Kosten



Stand 2007:

- Herstellungskosten
  - DRAM: 1 ct /Mbit
  - Prozessor: 13 ct / Mio. Trans.
- Testkosten
  - >50% der Herstellungskosten

© A. Steininger / TU Wien

Design for Test (DFT)

Bereits beim Design wird auf die Bedürfnisse des Tests Rücksicht genommen:

- Testbarkeit als wesentliche Anforderung an das Design
- Einhaltung spezieller Design-Rules
- Integration von Logik zu Testzwecken
  - Beispiel: Built-in Self-Test (BIST)

© A. Steininger / TU Wien

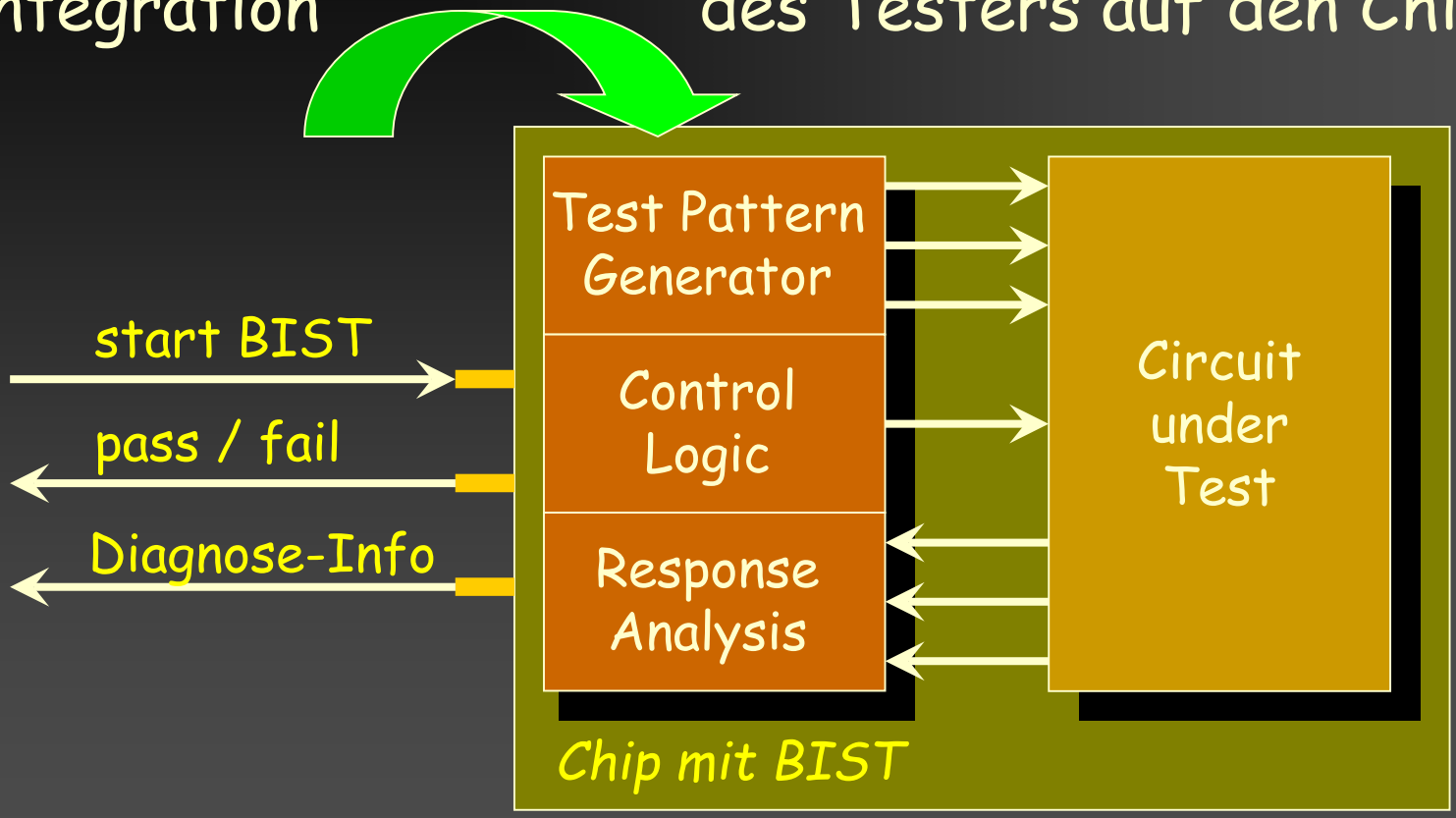
Einige DFT-Regeln

- Komplexe Logik sinnvoll partitionieren, Testhilfen in lange Zählerketten einfügen
- Initialisierung für sequentielle Logik vorsehen
- Redundante Logik vermeiden
- keine Verzögerungsglieder als funktionale Elemente
- strikte Trennung von Takt und sonstigen Signalen
- Clock Gating vermeiden
- Self-Resetting Logic vermeiden
- Bus-Strukturen bevorzugt verwenden (Partitionierung)

© A. Steininger / TU Wien

Built-in Self-Test: Prinzip

Integration des Testers auf den Chip



© A. Steininger / TU Wien

BIST-Implementierung

- Testmustergenerator
  - Speichern determinist. Vektoren ineffektiv
  - Verwendung eines Pseudo-Random Generators
  - realisiert als rückgekoppeltes Schieberegister
- Response Analyser
  - Speichern aller Responses ist ineffektiv
  - response data compaction (Signatur)
  - realisiert mittels Multiple-Input Shift Register
  - Gefahr des Aliasing

© A. Steininger / TU Wien

Response Data Compaction

- Die Sequenz der Responses auf die einzelnen Testvektoren wird mittels MISR auf eine Signatur abgebildet („Compaction").
- Compaction führt zwangsläufig zu *Aliasing*:

  Auch einzelne fehlerhafte Responses führen zu einer richtigen Signatur
- Wahrscheinlichkeit für Aliasing ist etwa $p \approx 2^{-R}$ (R ist die Breite des Signaturregisters)

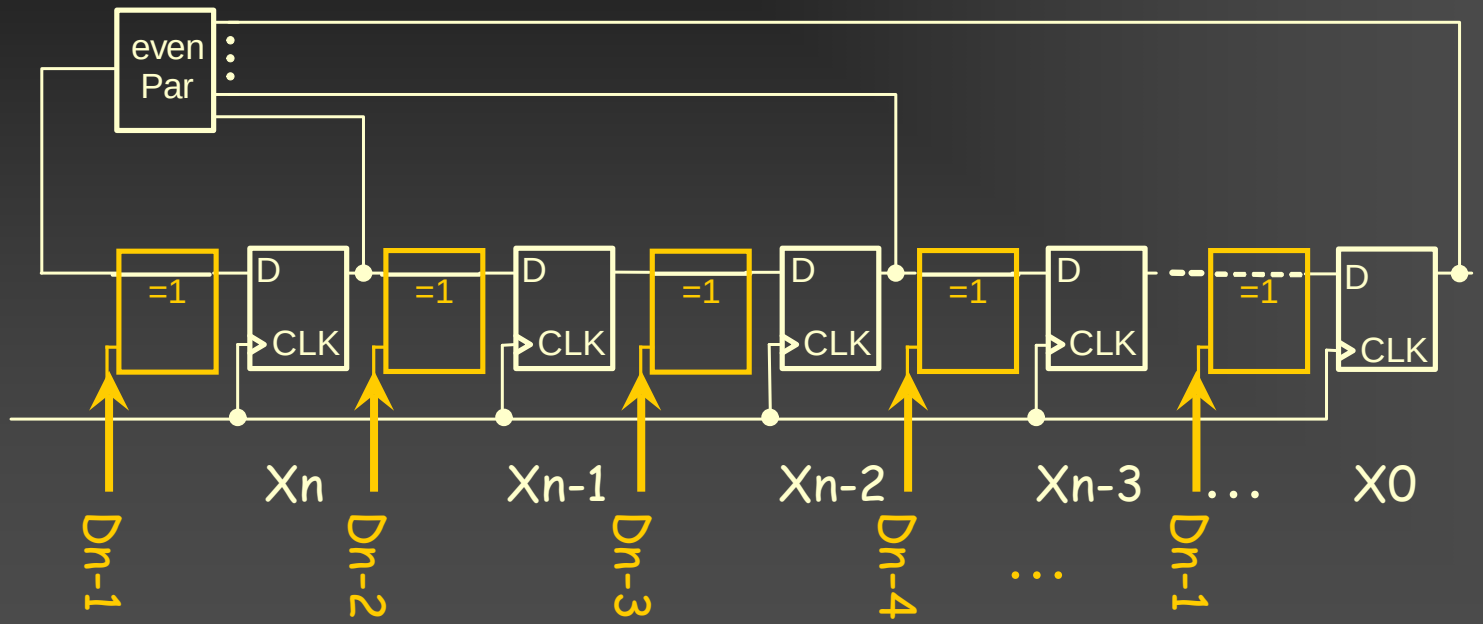
  unter der (unzutreffenden) Annahme, dass alle Bitkombinationen gleich wahrscheinlich als fehlerhafter Output auftreten

© A. Steininger / TU Wien

MISR: Implementierung

Linear Feedback Shift Register (LFSR)

➡ Multiple-Input Shift Register (MISR)

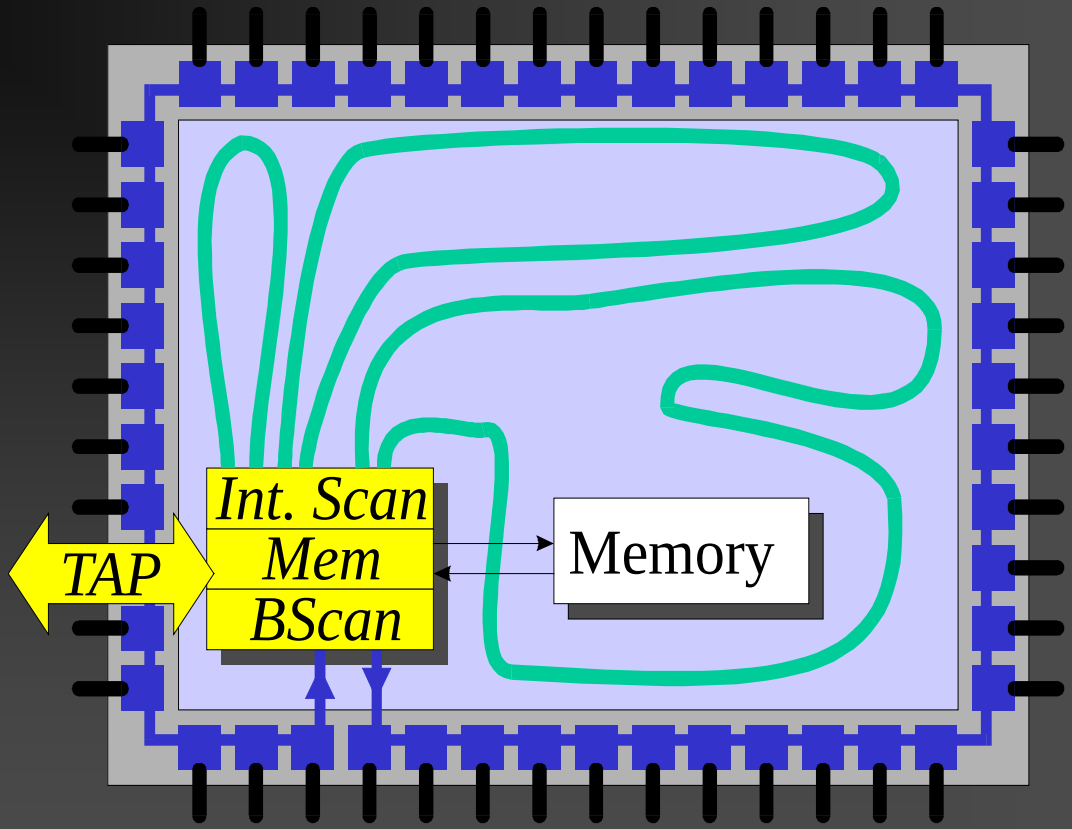


© A. Steininger / TU Wien

Eine typische BIST-Lösung

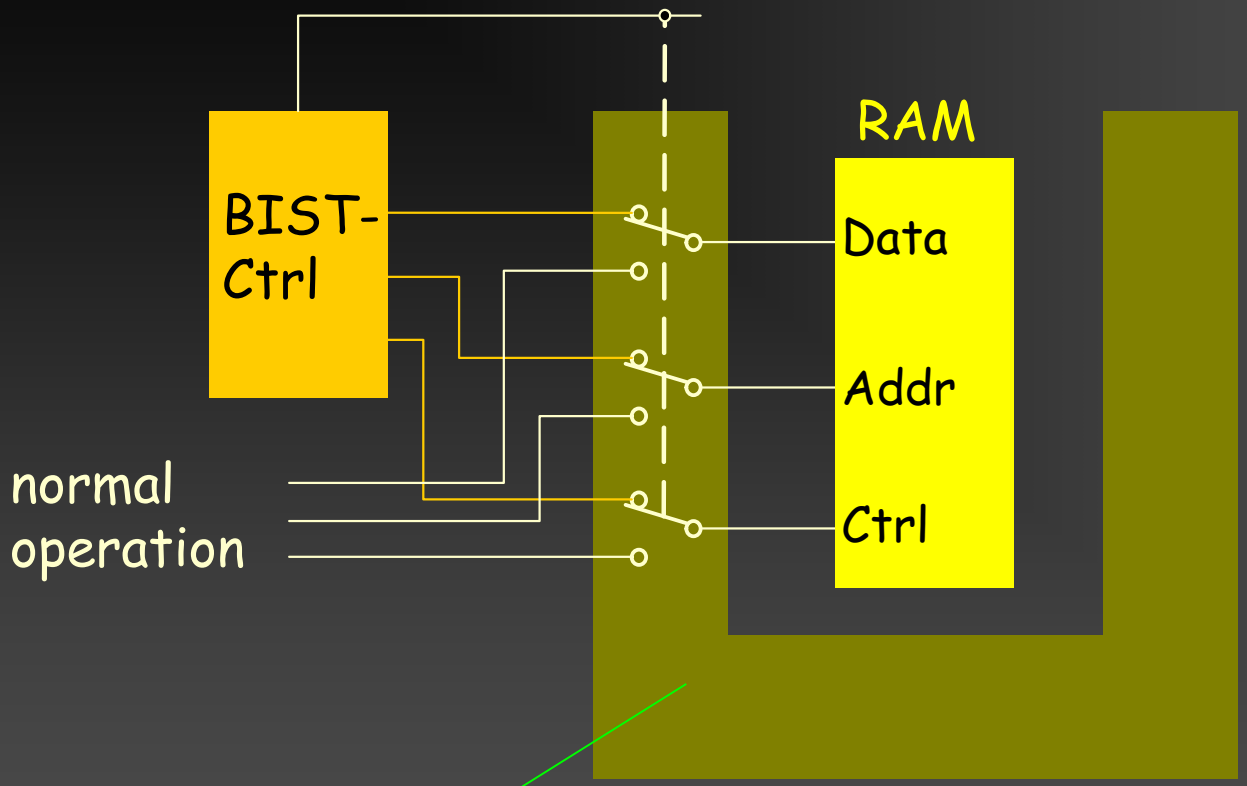
Gemeinsamer *Test Access Port* (5 pins) für

- ✓ boundary scan
- ✓ internal scan
- ✓ Speichertest
- ✓ Konfiguration



© A. Steininger / TU Wien

BIST von Embedded Memory



Collar-Logic isoliert Speicher während des Tests

© A. Steininger / TU Wien

Vorteile von BIST (1)

- Das kritische Interface zwischen Tester und Testobjekt wird auf den Chip verlagert.
- Das Interface nach außen wird extrem einfach.
- Der Test kann „at speed" ablaufen.
- Die Zugänglichkeit zu Testpunkten ist besser; das erhöht die Testbarkeit.
- Die extrem hohen Investitionskosten für Tester werden drastisch vermindert.
- Der Overhead für die BIST-Logik liegt in der Größenordnung von 10%.

© A. Steininger / TU Wien

Vorteile von BIST (2)

- Der Designer eines IP-Core kann BIST mit implementieren und braucht daher auch für den Test keine funktionalen Details preisgeben.
- BIST kann hierarchisch weiterverwendet werden: Self-Testing Chip, Self-Testing Board, Self-Testing System
- BIST erlaubt rasche Diagnose/ Maintenance im Feld, ggf. auch remote
- BIST kann für eine Start-up Test bzw. auch on-line wiederverwendet werden.

© A. Steininger / TU Wien

On-line Testing

- Zweck: Auffinden von Defekten unmittelbar während des Betriebes
- Erforderlich insbesondere bei langen ununterbrochenen kritischen Missionen (z.B. Raumfahrt, Weichen-Stellwerk)
- Problem: Transparenz des Tests
  - Test darf Daten und inneren Zustand des Testobjektes nicht verändern
  - Test darf Zeitverhalten des Testobjektes nicht verändern

© A. Steininger / TU Wien



Zusammenfassung (1)

- Gemäß der „Rule of ten" steigen die Kosten für einen Defekt mit jedem Assemblierungsschritt um den Faktor 10.
- Tests werden nicht nur in der Fertigung durchgeführt. Sie begleiten den Chip während seines gesamten Lebenszyklus.
- Übliche Maße für die Testqualität sind Defect Level und Test Coverage.
- Fehler in redundanter Logik können prinzipiell durch einen Test nicht erkannt werden.
- Die Testbarkeit ist bestimmt durch Beobachtbarkeit und Steuerbarkeit.

© A. Steininger / TU Wien



Zusammenfassung (2)

- Beim Exhaustive Testing werden alle möglichen Testvektoren angelegt und das Testobjekt somit vollständig getestet. Diese Methode führt in der Praxis meist zu einer unrealistisch hohen Anzahl von Testvektoren.
- Beim Deterministic Testing wird versucht, systematisch eine Liste von Testvektoren zu erstellen. Dabei bedient man sich der Fehlersimulation sowie einiger Techniken zur Reduktion (Äquivalenz, Dominanz)
- Beim nondeterministic Testing werden die Testvektoren nach dem Zufallsprinzip gewählt. In der Praxis bewährt sich diese Methode (bei sinnvoller Handhabung) sehr gut.

© A. Steininger / TU Wien



Zusammenfassung (3)

- Der Test sequentieller Logik ist besonders problematisch, da zunächst das Testobjekt in einen bestimmten Zustand gebracht werden muss.
- Der Scan-Test vermeidet dieses Problem, indem alle Register zu einem Schieberegister verbunden werden. Mit diesem wird der Testvektor an die verbleibende kombinatorische Logik geführt.
- Der Scan Test ist ein struktureller Test: Er überprüft (im Gegensatz zum funktionalen Test) nicht die spezifizierte Funktion der Gesamtanordnung, sondern nur die Funktion der verwendeten logischen Gatter.

© A. Steininger / TU Wien



Zusammenfassung (4)

- Beim Speicher erweist sich aufgrund seiner einfachen Funktionalität und der etwas anderen Fehlermodelle ein funktionaler Test als die bessere Lösung.
- March-Tests bieten eine besonders hohe Effizienz beim Speichertest.
- Für das Testen von Leiterplatten hat wurde der Boundary-Scan standardisiert.
- Der Test-Access Port ermöglicht eine effiziente Kommunikation mit der Testlogik auf dem Chip.

© A. Steininger / TU Wien



Zusammenfassung (5)

- Beim Built-in Self-Test befindet sich die Testlogik auf dem Chip. Dies verursacht zwar einen HW-Overhead, erhöht die Test-Performance jedoch entscheidend und erlaubt eine Wiederverwendung in anderen Testphasen.
- Für das Generieren von pseudozufälligen Testmustern werden Linear Feedback Shift Register (LFSR) verwendet.
- Die Compaction der Responses erfolgt in einem Multiple Input Shift Register (MISR). Sie führt zu Aliasing.

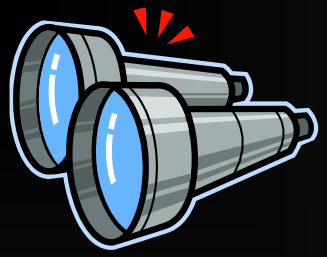
© A. Steininger / TU Wien

Der Logikanalysator

Wenn man's genau wissen muss ...



© A. Steininger / TU Wien



Überblick

- Vergleich der Messgeräte
- Aufbau eines Logikanalysators
- Digitalisierung des Signalverlaufes
- Sampling (State und Timing-Mode)
- Speicher
- Trigger
- Darstellung des Ergebnisses
- Datenblatt

© A. Steininger / TU Wien

Wahl des geeigneten Mittels

#Kanäle

- Multimeter
  - relativ stabile Kennwerte (VDC, VRMS) — 1
- Oszilloskop
  - genauer Spannungsverlauf über der Zeit, — 2 - 4
- Timing-Analyzer
  - Logikpegel über der Zeit — 16 - 512
- State Analyzer
  - Folge von Zuständen in synchronem System

© A. Steininger / TU Wien

Multimeter





- 1 Kanal
- Meßwert wählbar
  - Spannung, Strom, R, C, ...
  - Rest nicht erkennbar
- kein Zeitverlauf
- Frequenzbereich wenige kHz
- bei raschen Änderungen
  - Mittelung oder
  - falsche Anzeige

© A. Steininger / TU Wien

Oszilloskop



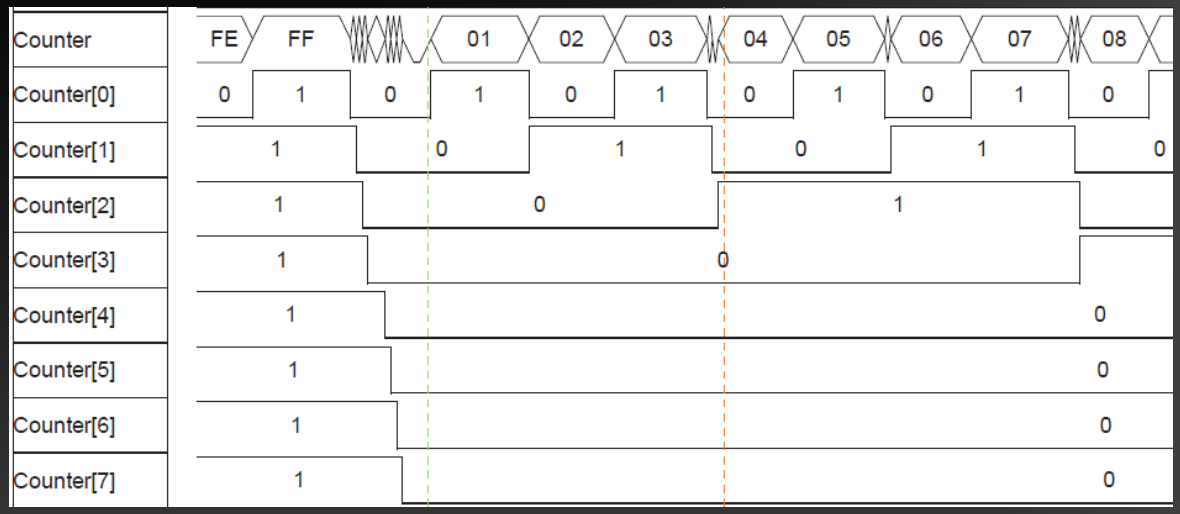


- bis zu 4 Kanäle
- nur Spannung messbar
- Zeitverlauf sichtbar
- alle Signaldetails
- viele Messfunktionen
- Frequenzbereich bis zu mehreren GHz

© A. Steininger / TU Wien

Timing Analyzer



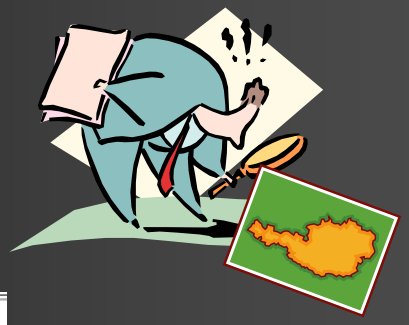


typ. bis zu
512 Kanäle

- nur Logikpegel messbar (HI/LO), keine Signaldetails
- Darstellung als Bus, unterschiedl. Zahlenbasis, …
- viele Messfunktionen
- Zeitverlauf sichtbar
- Frequenzbereich bis zu einzelnen GHz

© A. Steininger / TU Wien

State Analyzer



Counter	FF	00	01	02	03	04	05	06	07	08
Counter[0]	1	0	1	0	1	0	1	0	1	0
Counter[1]	1	0		1		0		1		0
Counter[2]	1	0				1				
Counter[3]	1	0								
Counter[4]	1									0
Counter[5]	1									0
Counter[6]	1									0
Counter[7]	1									0

typ. bis zu
512 Kanäle

- nur Logikpegel meßbar (HI/LO), keine Signaldetails
- Zeitverlauf nur synchron sichtbar (zu Taktflanken)
- Darstellung als Bus, unterschiedl. Zahlenbasis, …
- viele Messfunktionen
- Frequenzbereich bis zu GHz

© A. Steininger / TU Wien

Kontinuierlich vs. diskret



© A. Steininger / TU Wien

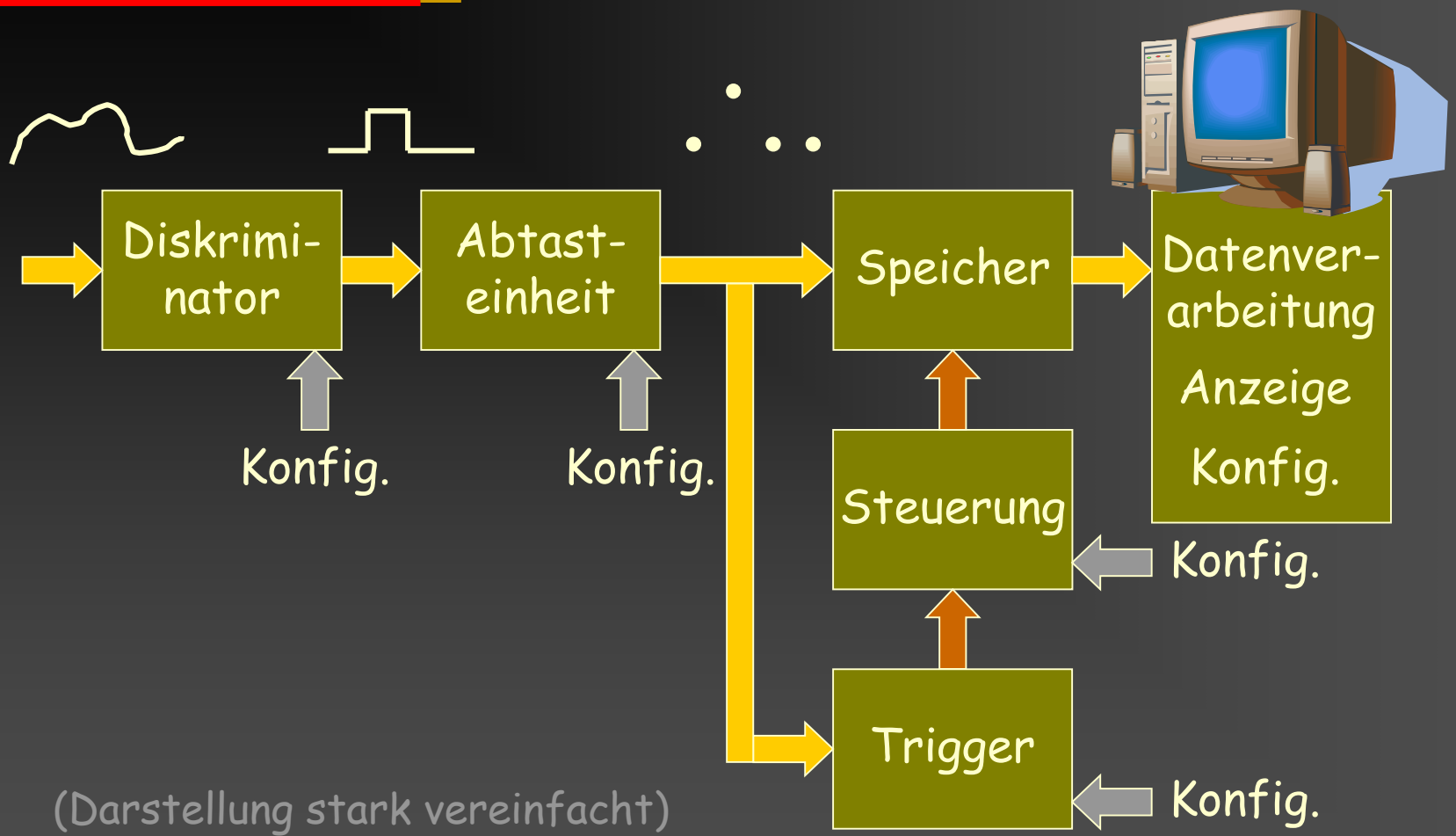
Logikanalysator



[Agilent]

© A. Steininger / TU Wien

Funktionsblöcke eines LA



(Darstellung stark vereinfacht)

© A. Steininger / TU Wien

Diskriminator

- Digitalisierung des Signals
  - Umsetzung von analogem Spannungsverlauf auf zwei Logikpegel (HI/LO)
  - beim Speicheroszilloskop sitzt hier ADC und liefert feinstufigere Amplitudeninformation
- Schwellwert ist Basis für Digitalisierung
  - einstellbar für viele Logikstandards sowie „user defined“
  - dennoch kann Interpretation anders sein als in der untersuchten HW (vor allem im Fehlerfall)!

© A. Steininger / TU Wien

Fehlerquelle Schwellwert (1)

- **Problem:** Schwellwert des LA stimmt nicht mit jenem der HW überein
- **Effekt:** es werden falsche Pegel angezeigt

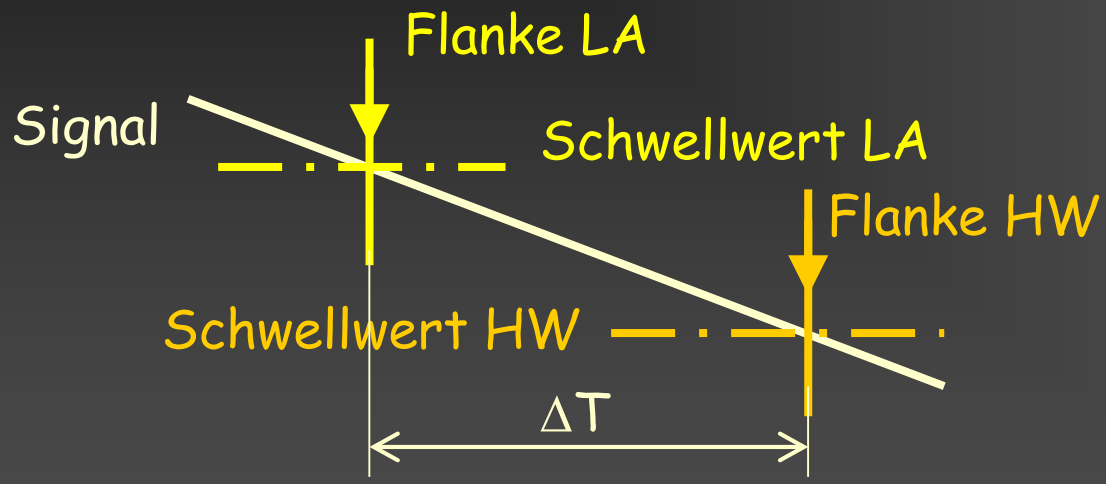


tritt bei richtig eingestelltem Logikstandard nur auf, wenn Signalpegel im verbotenen Bereich ist

© A. Steininger / TU Wien

Fehlerquelle Schwellwert (2)

- **Problem:** Schwellwert des LA stimmt nicht mit jenem der HW überein
- **Effekt:** Flanken werden verschoben dargestellt



Verschiebung nur bei sehr flachen Flanken relevant

© A. Steininger / TU Wien

Abtasteinheit (Sampler)

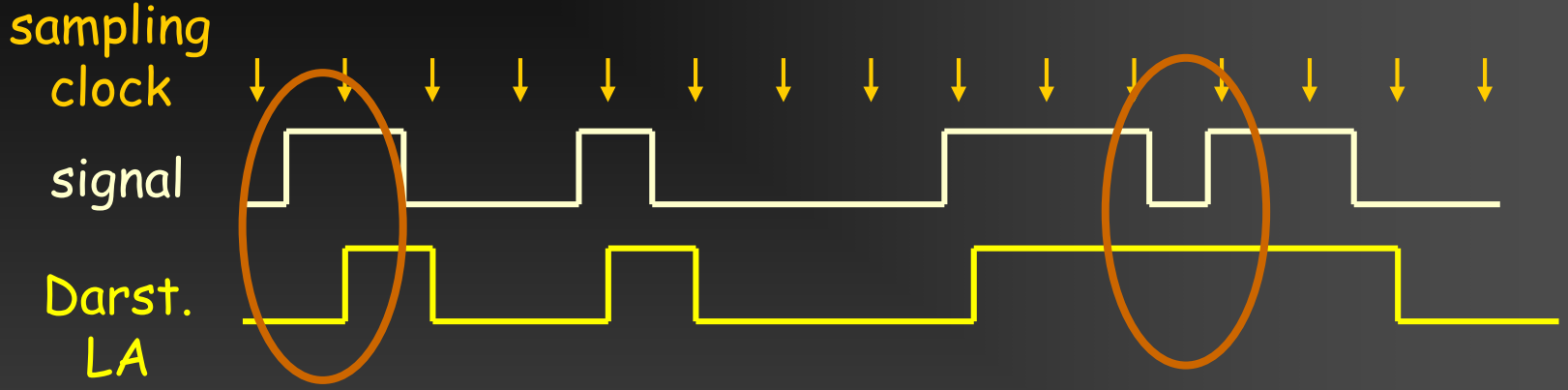


- Zeitliche Diskretisierung
  - Voraussetzung für Speicherung
  - bei „Unterabtastung“ gehen Details verloren
- Takt als Basis für Abtastung
  - Timing Analyzer: LA liefert hochfrequenten Takt => Erfassung aller Signaldetails (gleiches Prinzip wie Speicheroszilloskop)
  - State Analyzer: Target liefert Takt => Synchrone Abtastung der „Zustände“ Fehlinterpretation bei Phasenverschiebung!

© A. Steininger / TU Wien

Fehlerquelle Unterabtastung

nur im Timing Mode

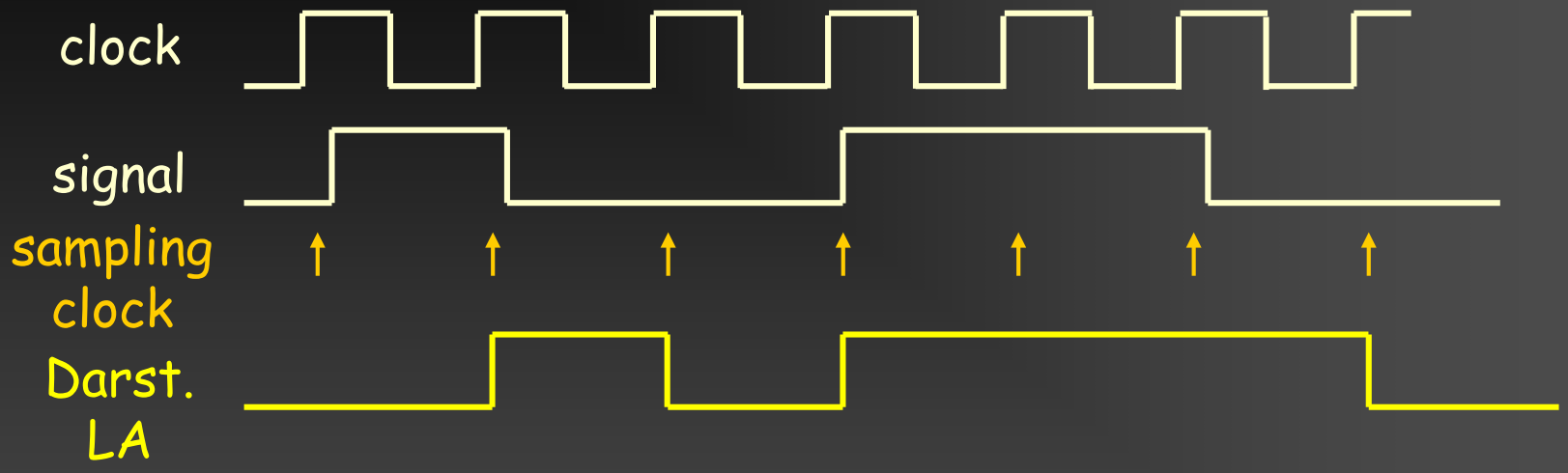


- wegen begrenzter Zeitauflösung erscheinen Flanken verschoben (erst bei nächstem Sample)
- Pulse innerhalb einer Sampling-Periode gehen völlig verloren

© A. Steininger / TU Wien

Fehlerquelle Phasenversch.

nur im State Mode



- Phasenverschiebung beim Sampling Clock führt zu unterschiedlicher Interpretation der Zustände und zu Setup/Hold Violations im LA

© A. Steininger / TU Wien

Speicher



- Ablegen der Samples
  - Grenze zwischen Real-Time und Offline:
  - Ablegen in Echtzeit (> 1GHz !)
  - Bearbeitung und Darstellung offline
- Tiefe typisch 1MSample
  - bei Abtastung mit 1GHz nach 1ms voll
  - das ist in manchen Fällen zu kurz!
  - Wer durchsucht 1 Mio States nach Fehler?

  => gezielt aufzeichnen => Trigger

© A. Steininger / TU Wien

Aufzeichnungsmodi



nur im Timing Mode

- Verringern der Abtastrate
  - längeres Intervall passt in den Speicher:

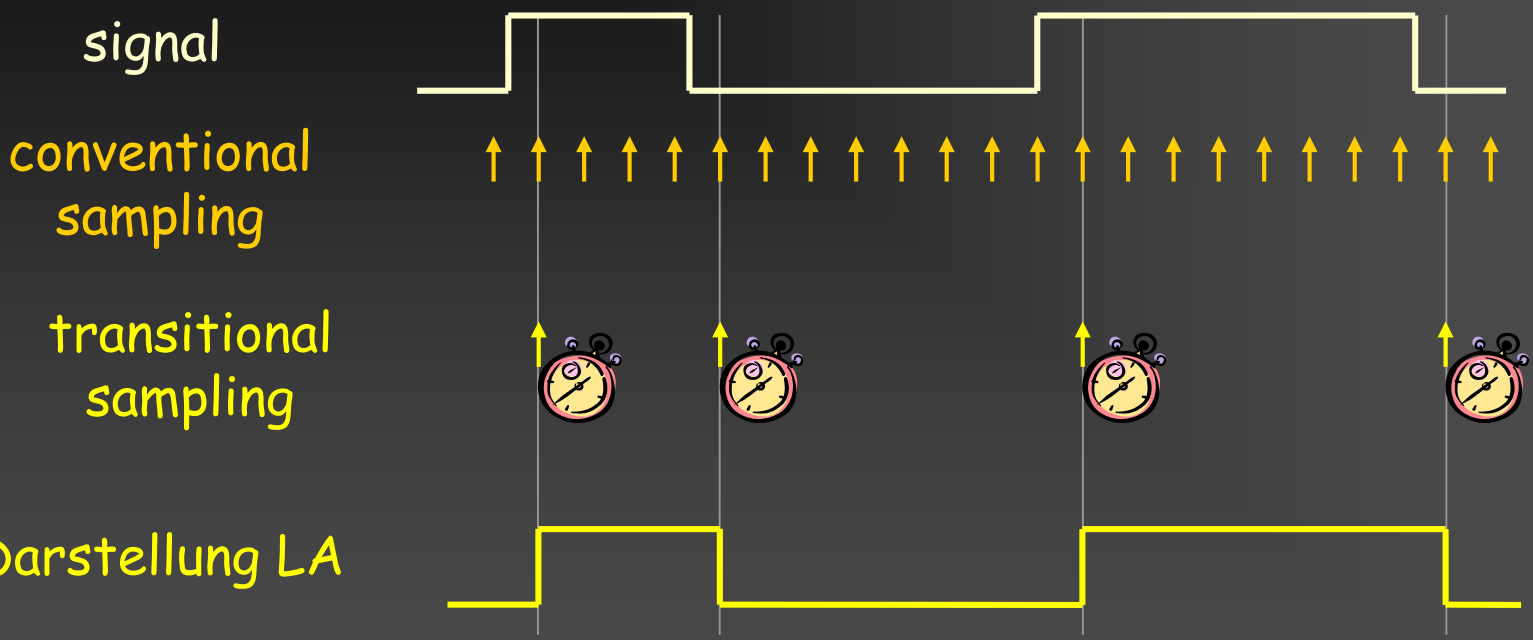
    Beob.-Intervall = Speichertiefe/Abtastrate
  - geringere Zeitauflösung

    Zeitauflösung = 1/Abtastrate
- Transitional sampling
  - Speichern von Flanke und Zeitpunkt statt äquidistanter Samples

© A. Steininger / TU Wien

Transitional Sampling

nur im Timing Mode



© A. Steininger / TU Wien

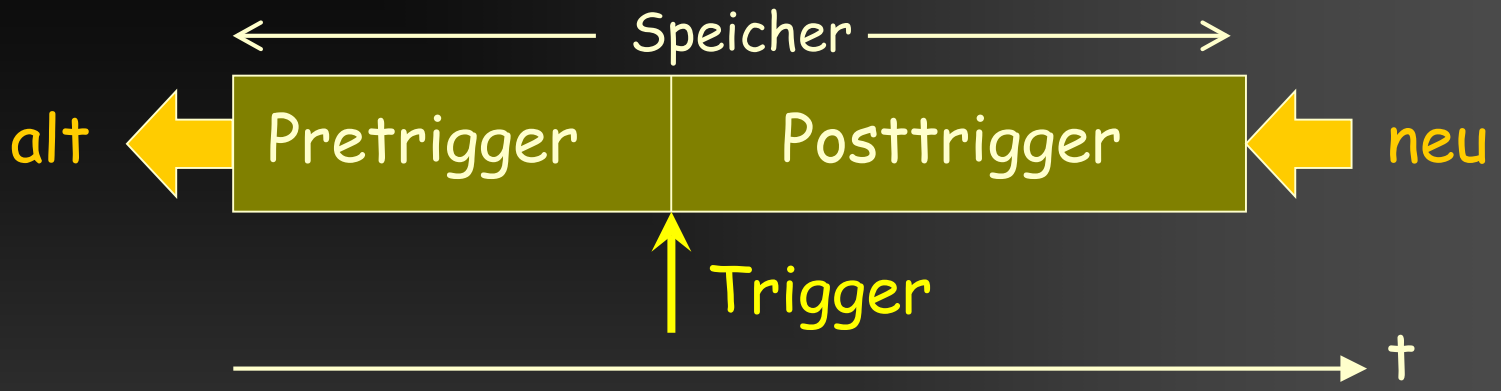
Trigger



- Beginn der Aufzeichnung festlegen
  - durch Beschreiben des interessierenden Events
  - sicherstellen, dass das interessierende Event im aufgezeichneten Intervall enthalten ist (LA kann auch Stunden auf trigger warten …)
- Markierung später im Display
  - erleichtert Auffinden der relevanten Daten
  - Referenz für Messfunktionen

© A. Steininger / TU Wien

Pretrigger und Posttrigger



- Speicher als FIFO betrieben
  - es wird ständig aufgezeichnet
  - bei Trigger: nur noch Posttrigger mit neuen Daten vollschreiben, danach Ende d. Aufzeichnung
  - Pretrigger daher nicht überschrieben

© A. Steininger / TU Wien

Triggerbedingungen

- Flanke
- Zustand (ein oder mehrere Bits)
- Pulsdauer (länger/kürzer als xx)
- …

- Kombination von Bedingungen möglich (AND, OR)
- Sequenz möglich („trigger levels“)
- Verzweigungen möglich (If/then/else)
- unterschiedl. Aktivitäten (store, stop, …)

Triggermöglichkeiten im State mode komplexer

© A. Steininger / TU Wien

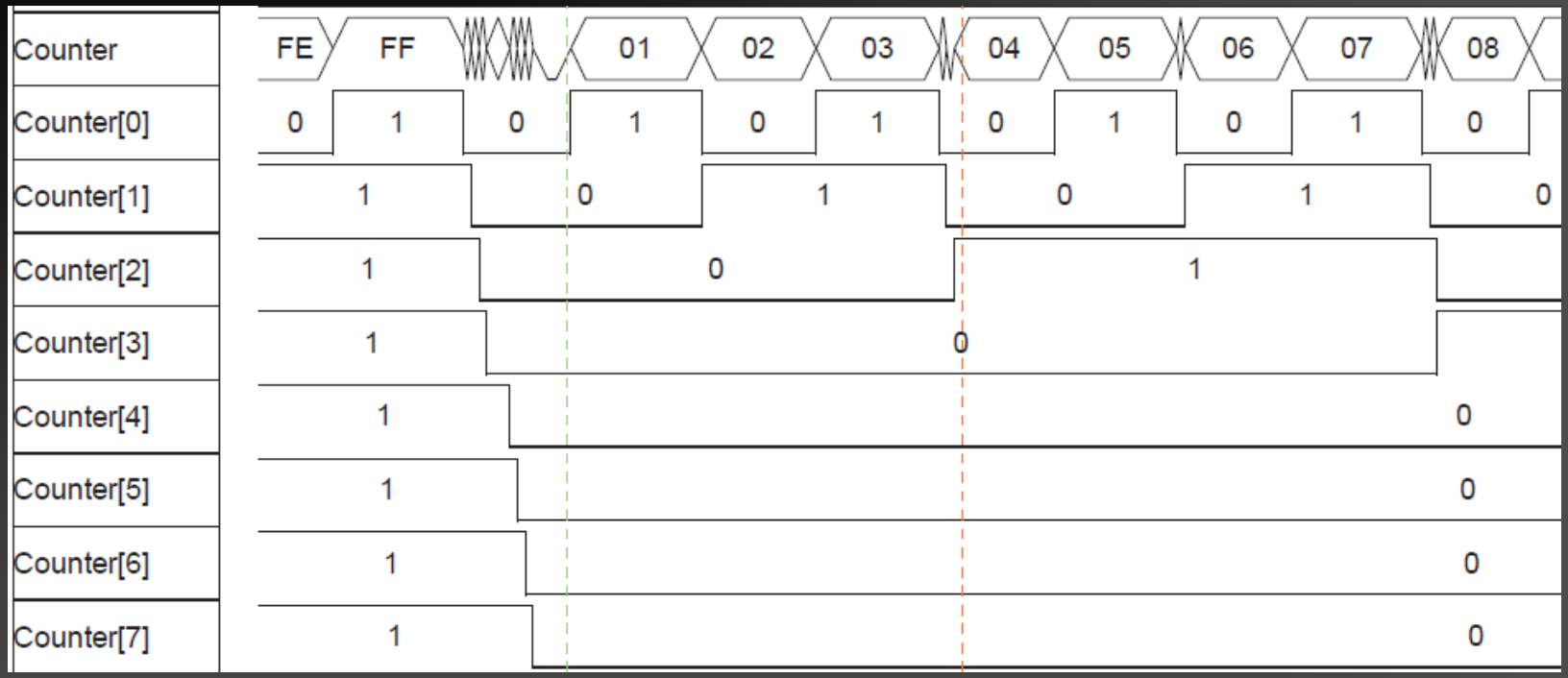
Cursor



- Markierung einer bestimmten Position im Zeitbereich (horizontal)
- direktes Ablesen eines Wertes auf der Cursorposition (Zeit und Daten)
- Suche nach Ereignissen in der Aufzeichnung (Cursor wird autom. positioniert)
- Differenzmessungen (Cursor1 to Cursor2, Cursor to Trigger)

© A. Steininger / TU Wien

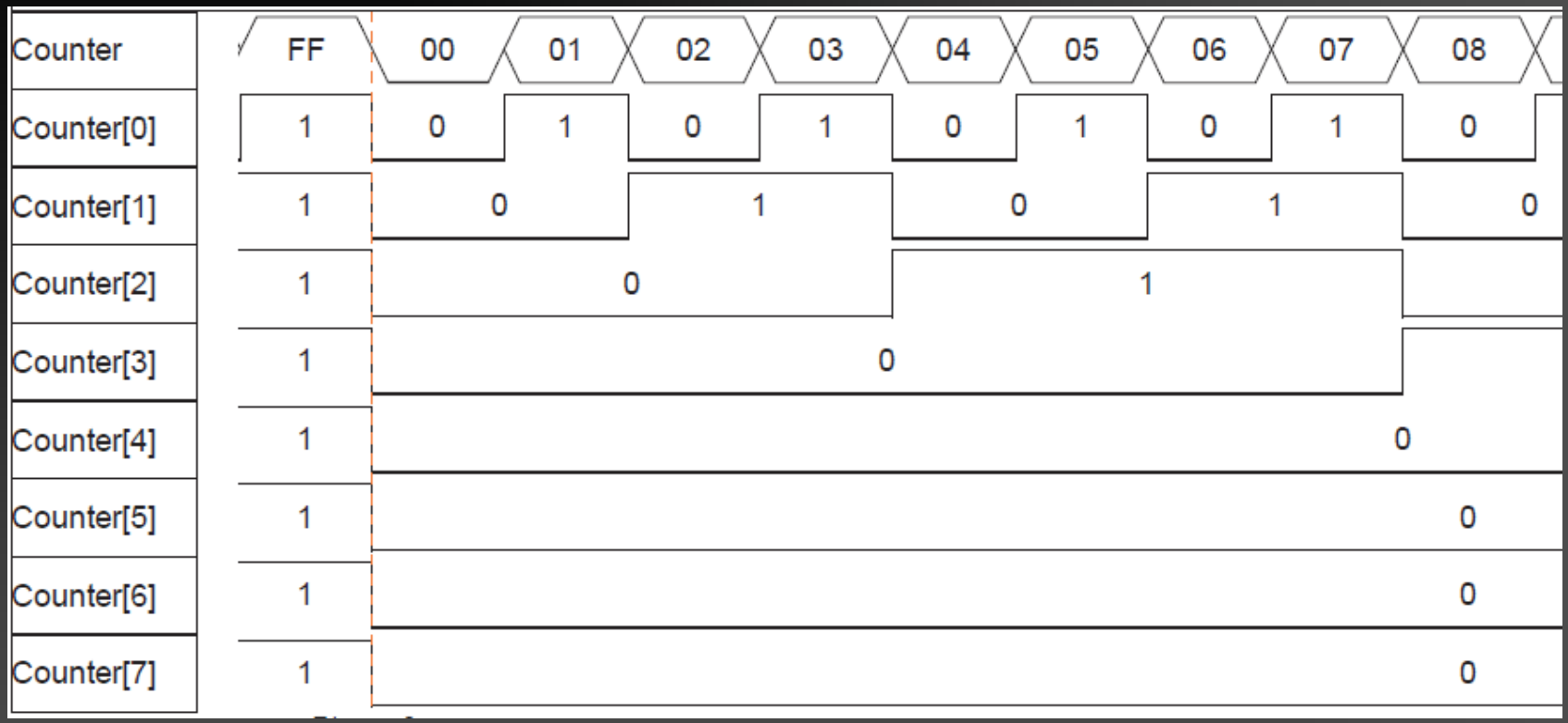
Darstellung: Timing-Diagramm



[Aufzeichnung erfolgte im Timing Mode]

© A. Steininger / TU Wien

Darstellung: State-Diagramm



[Aufzeichnung erfolgte im State Mode]

© A. Steininger / TU Wien

Darstellung: State-Listing

	Sample Number	Counter	Time
	-11	F5	-344 ns
	-10	F6	-312 ns
	-9	F7	-282 ns
	-8	F8	-250 ns
	-7	F9	-218 ns
	-6	FA	-188 ns
	-5	FB	-156 ns
M1	-4	FC	-124 ns
	-3	FD	-94 ns
	-2	FE	-62 ns
	-1	FF	-32 ns
T	0	00	0 s
	1	01	32 ns
	2	02	62 ns
	3	03	94 ns
	4	04	126 ns
	5	05	156 ns
	6	06	188 ns
	7	07	218 ns
	8	08	250 ns
	9	09	282 ns
	10	0A	312 ns
M2	11	0B	344 ns
	12	0C	376 ns
	13	0D	406 ns

[Aufzeichnung erfolgte im State Mode]

© A. Steininger / TU Wien

Probing: Prinzip



Zweck

- mechanische & elektrische Verbindung
- Pegelanpassung
- Impedanzanpassung (Belastung, Reflexionen)

Probleme

- GHz über lange Kabel
- Kapazitäten und Induktivitäten

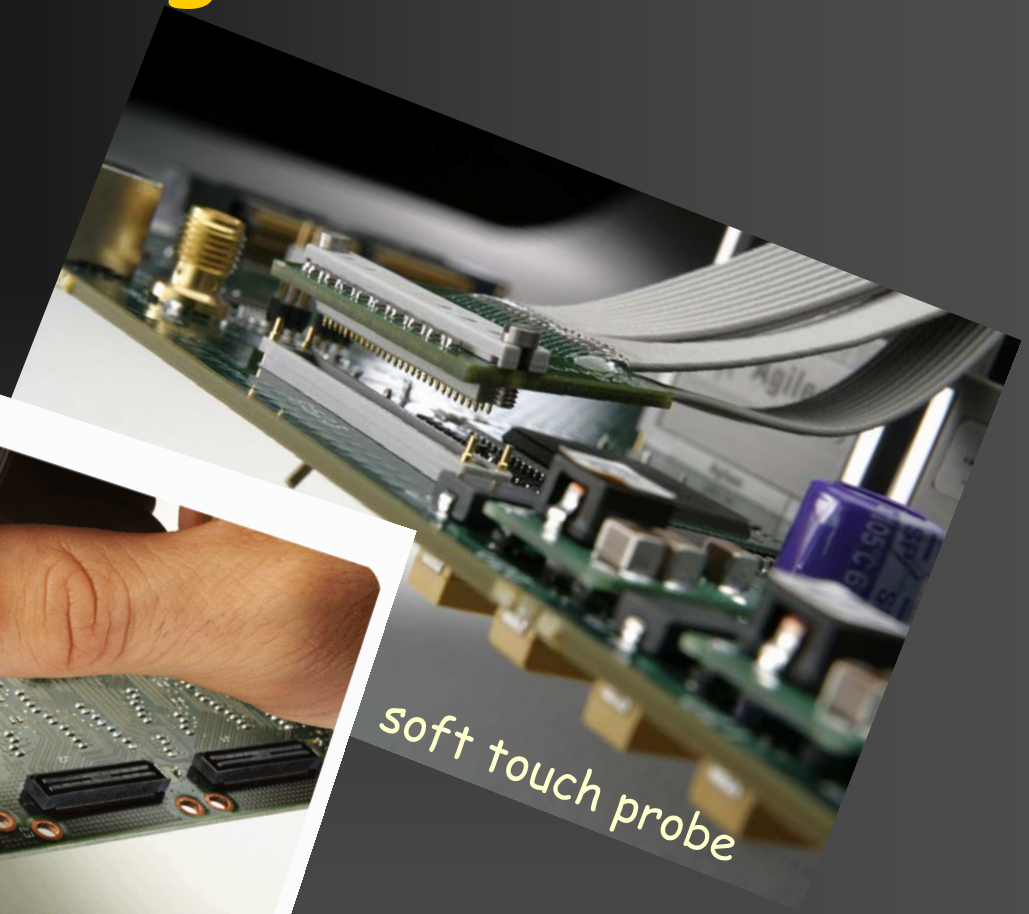
  => Signalform und Zeitbezüge ??

© A. Steininger / TU Wien

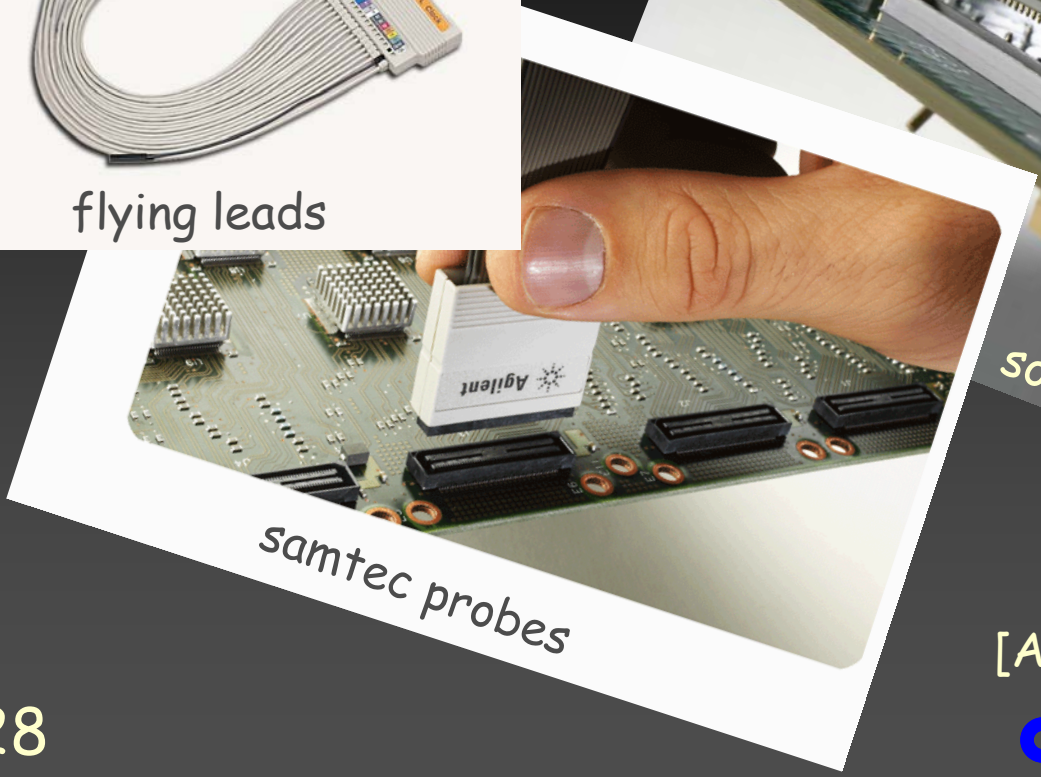
Probing: Lösungen



flying leads



soft touch probe



samtec probes

[Agilent]

© A. Steininger / TU Wien

Kenndaten eines LA

Agilent Model Number	16801A, 16821A[1]	16802A, 16822A[1]	16803A, 16823A[1]	16804A	16806A
Logic analyzer channels	34	68	102	136	204
Pattern generator channels[1]	48	48	48	N/A	N/A
High-speed timing zoom	4 GHz (250 ps) with 64 K depth	4 GHz (250 ps) with 64 K depth			
Maximum timing sample rate (Half/full ch)	1.0 GHz (1.0 ns) / 500 MHz (2.0 ns)	1.0 GHz (1.0 ns) / 500 MHz (2.0 ns)			
Maximum state clock rate	250 MHz with option 250	450 MHz with option 500 250 MHz with option 250			
Maximum state data rate	250 Mb/s with option 250	500 Mb/s with option 500 250 Mb/s with option 250			
Maximum memory depth	1 M with option 001 4 M with option 004 16 M with option 016 32 M with option 032	1 M with option 001 4 M with option 004 16 M with option 016 32 M with option 032			
Supported signal types	Single-ended	Single-ended			
Automated threshold/sample position Simultaneous eye diagrams, all channels	Yes	Yes			
Probe compatibility	40-pin cable connector	40-pin cable connector			

© A. Steininger / TU Wien



Zusammenfassung (1)

- Ein Logikanalysator kann wesentlich mehr Kanäle gleichzeitig darstellen als ein Oszilloskop, eignet sich aber nicht zur Darstellung des analogen Signalverlaufs.
- Das Eingangssignal wird zunächst digitalisiert, wobei der Schwellwert an den Logikstandard der HW anzupassen ist.
- Für die nachfolgende zeitliche Diskretisierung wird ein Takt benötigt. Im Timing Mode wird dieser vom Messgerät geliefert, im State Mode vom Target.

© A. Steininger / TU Wien



Zusammenfassung (2)

- Bei entsprechender Überabtastung kann im Timing Mode der Zeitverlauf noch recht genau verfolgt werden.
- Im State Mode ist die zeitliche Position der Flanken nicht mehr genau sichtbar, man erkennt dafür die Folge der synchronen Zustände.
- Da der Speicher begrenzt ist, muss im Timing Mode ggf. die Abtastrate verringert werden um ein genügend großes Intervall aufzuzeichnen. Für Signale mit geringer Aktivität ist Transitional Sampling vorteilhaft.

© A. Steininger / TU Wien



Zusammenfassung (3)

- Der Trigger dient als Referenzpunkt für die Aufzeichnung. Als Triggerbedingung wird daher üblicherweise das interessierende Event formuliert.
- Im Pretrigger-Bereich ist die Aktivität unmittelbar vor dem Trigger sichtbar, im Posttrigger die unmittelbar nachfolgende Aktivität.
- Die aufgezeichneten Daten können entweder in Form einer Liste oder als Timing-Diagramm dargestellt werden.
- Richtiges Probing ist wesentlich, um unverfälschte Messergebnisse zu erhalten.

© A. Steininger / TU Wien