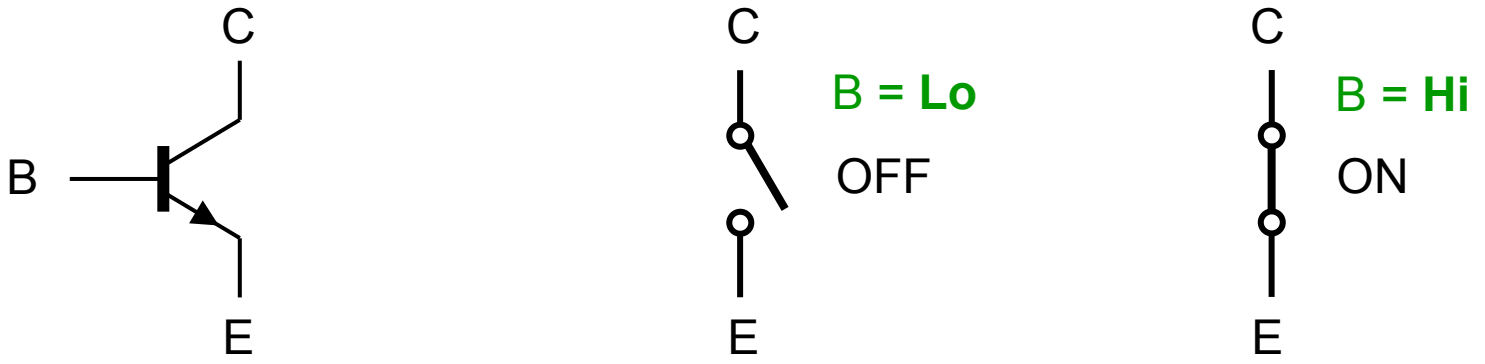
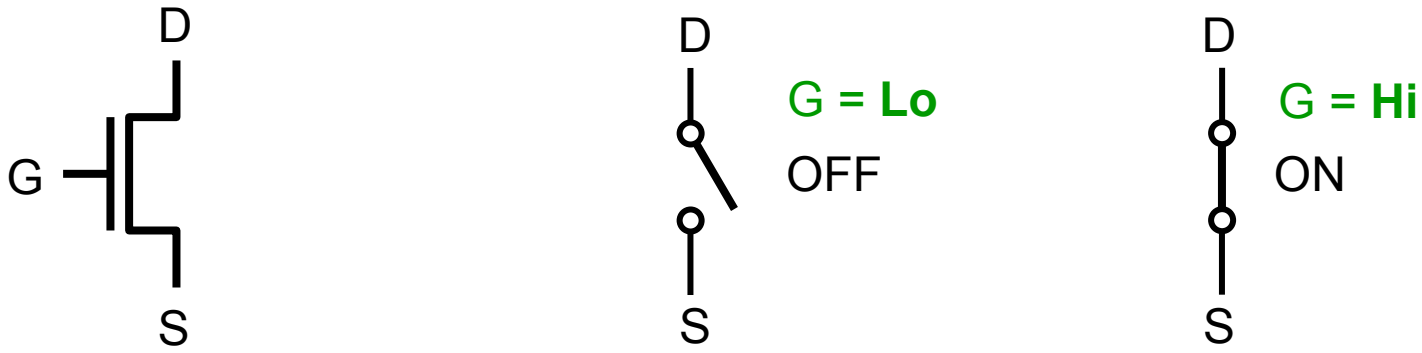
8. Digitale Grundschaltungen

Gesteuerte Schalter

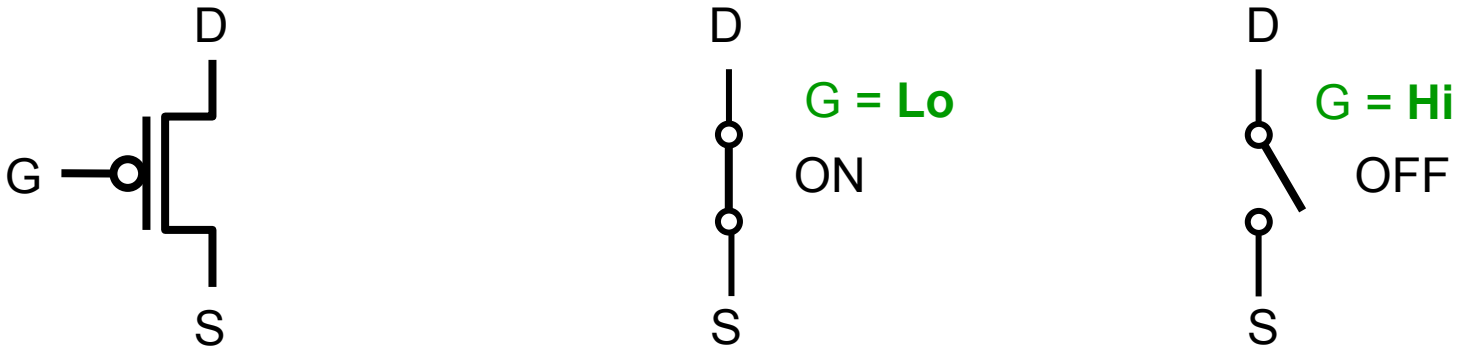
Bipolar-transistor



MOSFET
nMOS



MOSFET
pMOS

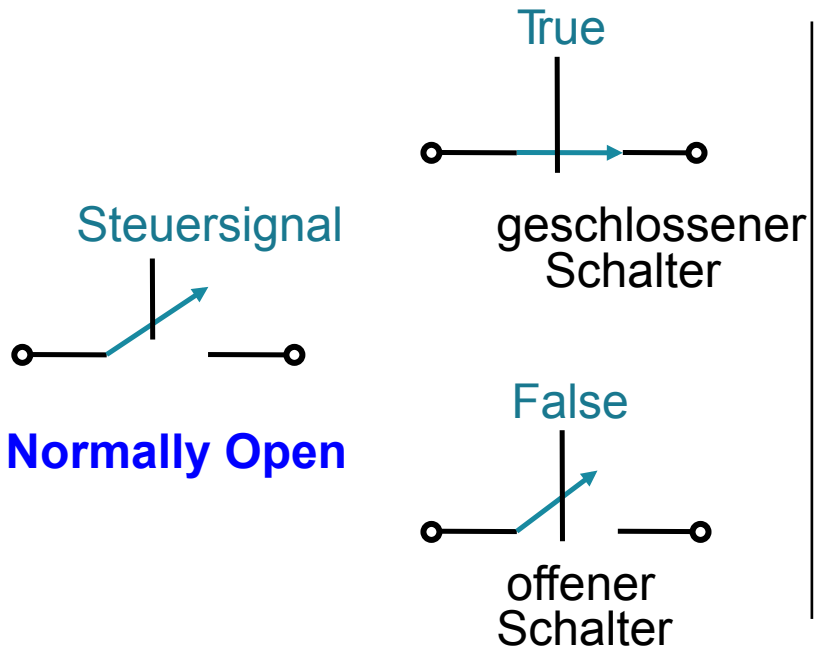


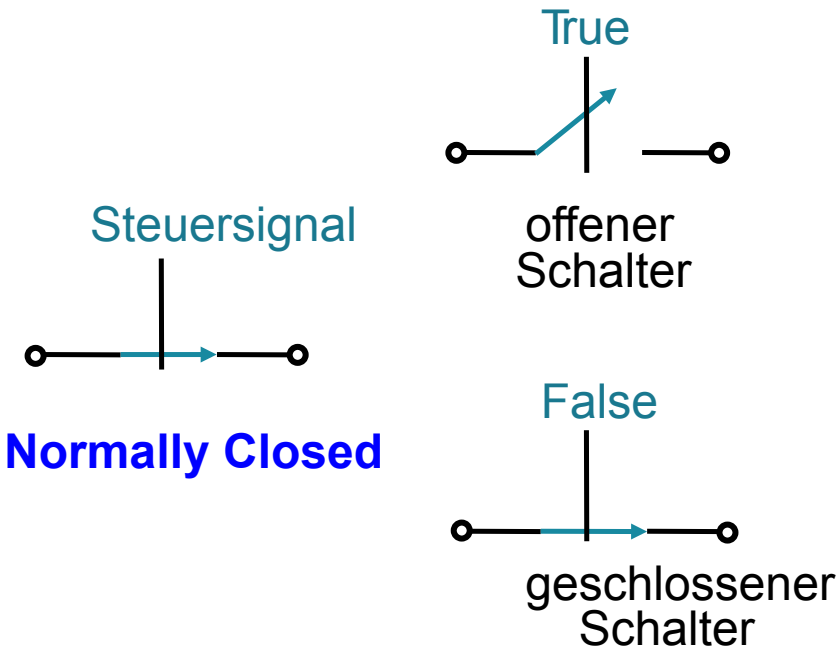
Schaltsymbole gesteuerter Schalter

gesteuerter Schalter Ein gesteuerter Schalter verbindet zwei Punkte in Abhängigkeit von einem Steuersignal.

Normally Open wenn das Steuersignal 0 (*false*) ist, ist der Schalter **offen**
wenn das Steuersignal 1 (*true*) ist, ist der Schalter geschlossen

Normally Closed wenn das Steuersignal 0 (*false*) ist, ist der Schalter **geschlossen**
wenn das Steuersignal 1 (*true*) ist, ist der Schalter offen

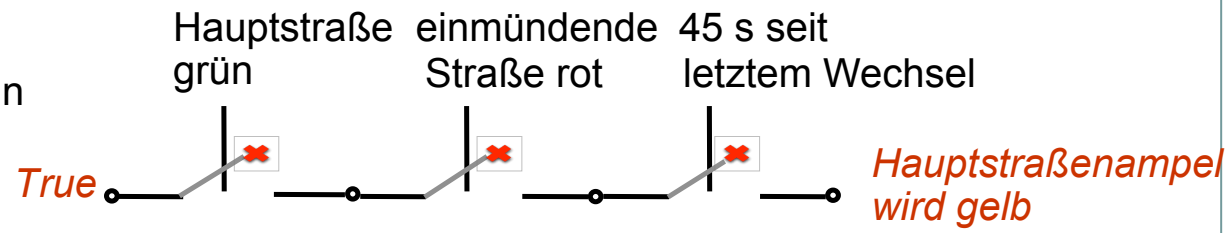




Schalternetzwerke

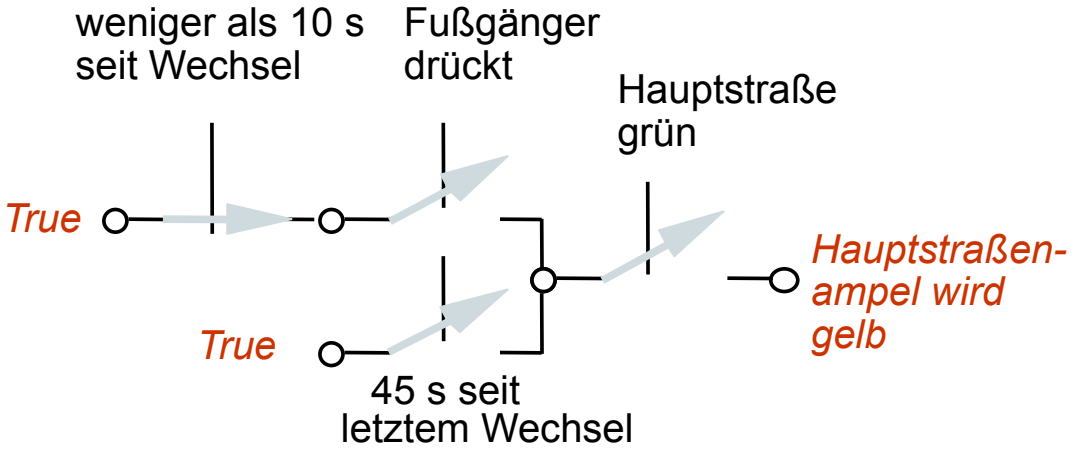
Beispiel:

WENN Ampel für Hauptstraße grün
UND Ampel für einmündende Straße rot
UND 45 s seit letztem Wechsel
DANN Ampel für Hauptstraße wird gelb



Beispiel:

WENN 45 s seit letztem Wechsel
ODER (Fußgänger drückt
UND NICHT weniger als 10s seit letztem Wechsel)
UND Hauptstraße grün
DANN Hauptstraße wird gelb



Knoten im Schwebezustand (floating):
Was passiert, wenn Hauptstraße nicht grün? Ausgang nicht spezifiziert (im Schwebezustand, *floating*). Physikalische Folge: Ausgang behält den letzten Wert (Kapazität wird nicht umgeladen)

Vollständige Schalternetzwerke

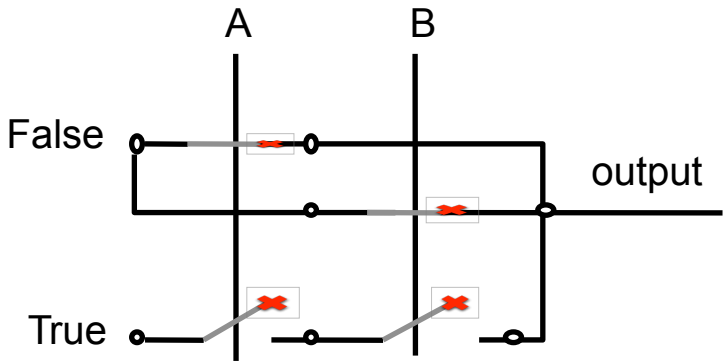
Wichtige Prinzipien in einem Schalternetzwerks: Für alle Werte der Steuersignale gilt:

(1) alle Ausgänge müssen über irgendeinen Pfad entweder mit *True* oder mit *False* verbunden sein (kein Schwebezustand)

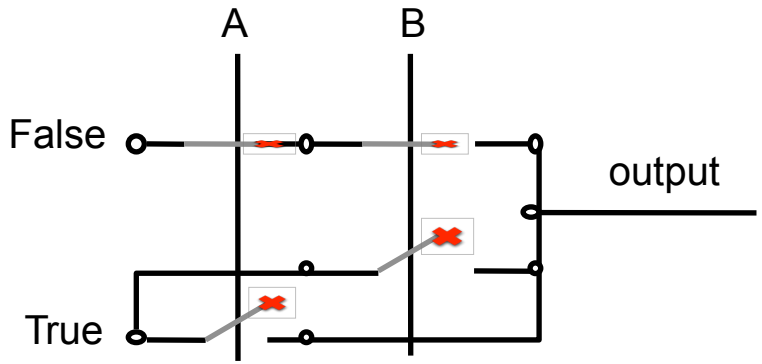
(2) kein Ausgang ist gleichzeitig mit *True* und *False* verbunden (keine Konflikte)

Folge: Die Verbindungsnetzwerke zu *True* und *False* müssen komplementär sein.

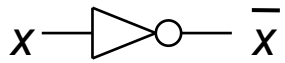
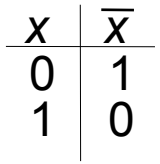
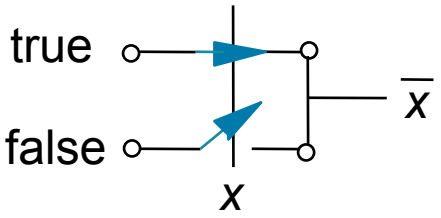
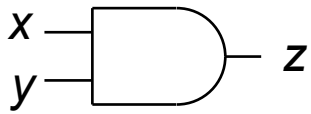
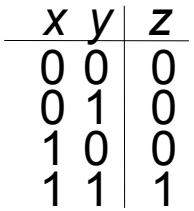
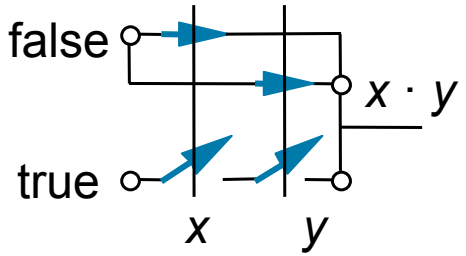
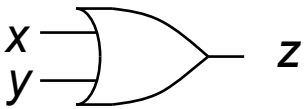
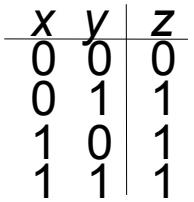
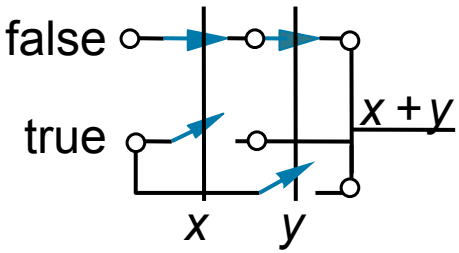
Eine UND-Verknüpfung (Konjunktion) kann durch eine Serienschaltung zu *True* implementiert werden.



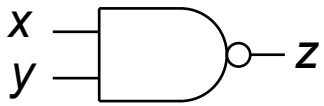
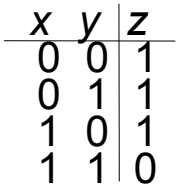
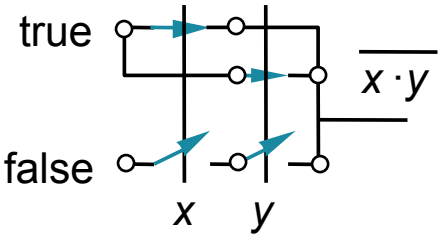
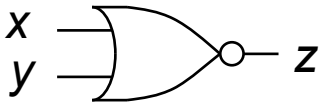
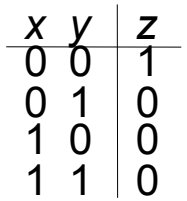
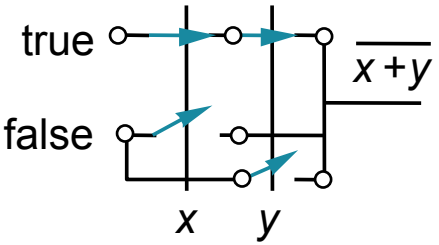
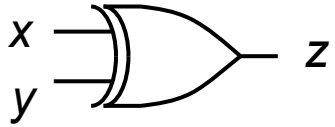
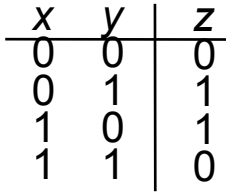
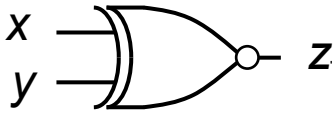
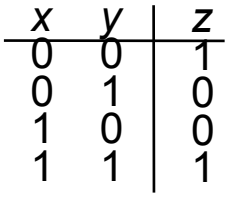
Eine ODER-Verknüpfung (Disjunktion) kann durch eine Parallelschaltung zu *True* implementiert werden.



Implementierung Boole‘scher Verknüpfungen

	Gattersymbol	Wahrheitstabelle	Schalternetzwerk
NOT (Negation)			
AND (Konjunktion)			
OR (Disjunktion)			

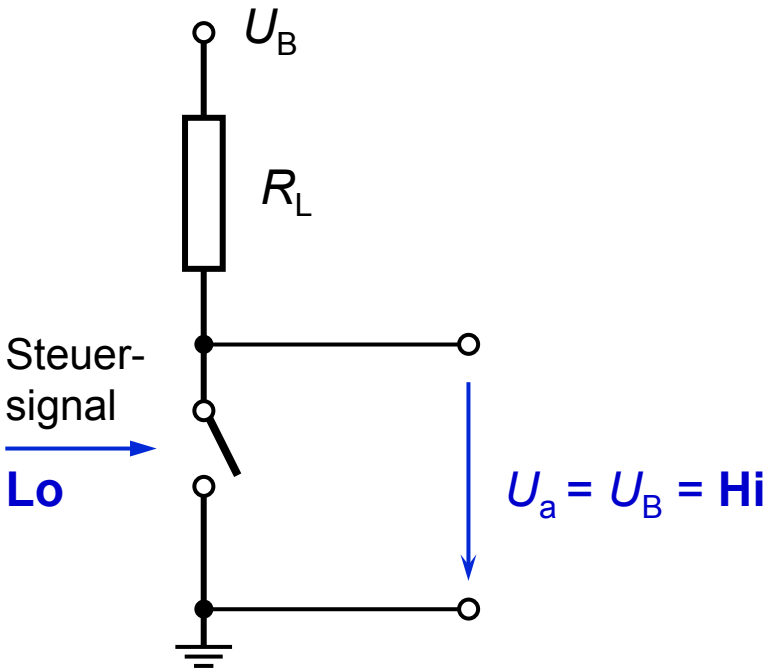
Implementierung Boole‘scher Verknüpfungen

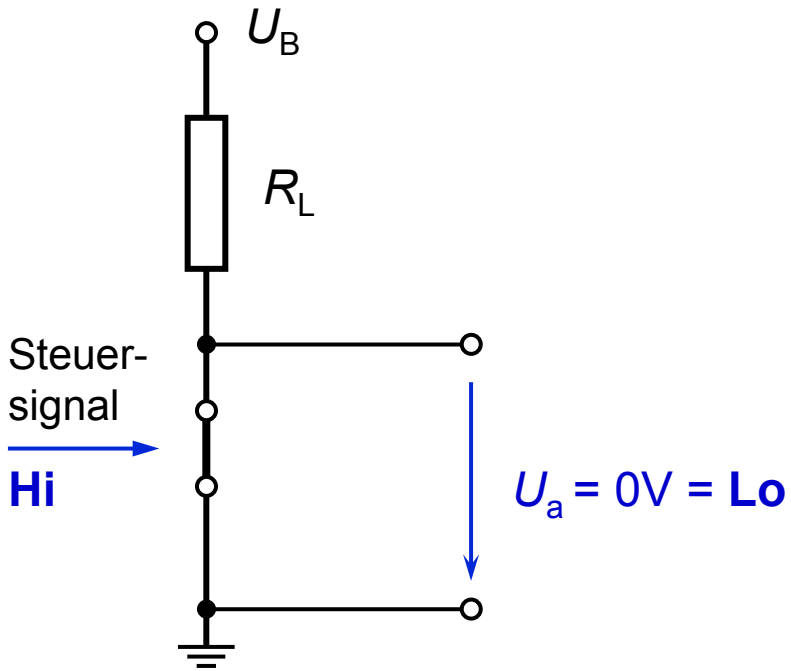
	Gatter	Wahrheitstabelle	Schalternetzwerk
NAND			
NOR			
XOR			$x \oplus y = x\,y' + x'\,y$
XNOR			$(x \oplus y)' = x\,y + x'\,y'$

Schalternetzwerke mit Widerstand

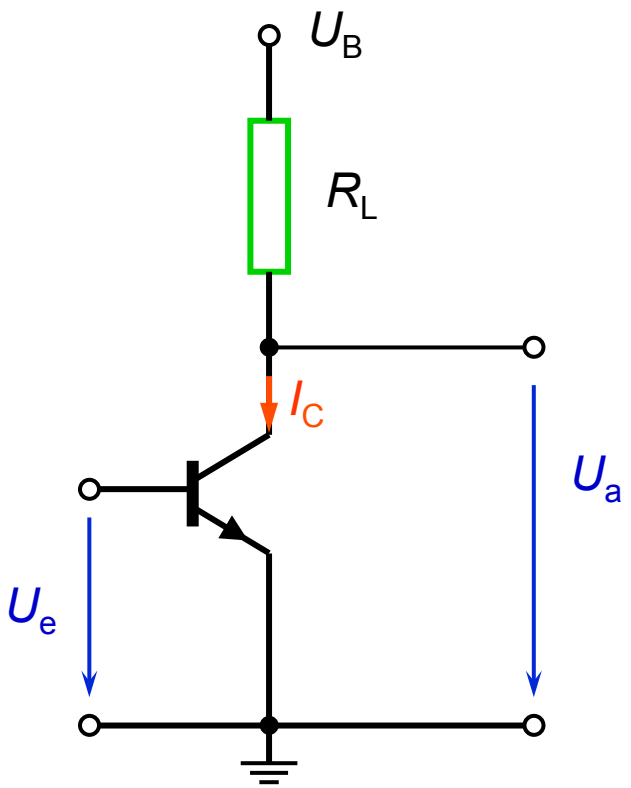
Die Einbeziehung eines Lastwiderstands ermöglicht es, die „Verdopplung“ der Schalter (komplementäre Netzwerke zu *True* und *False* zu vermeiden.

Hier: Inverterschaltung mit Verbindung zu True (Hi, U_B) mittels Lastwiderstand.

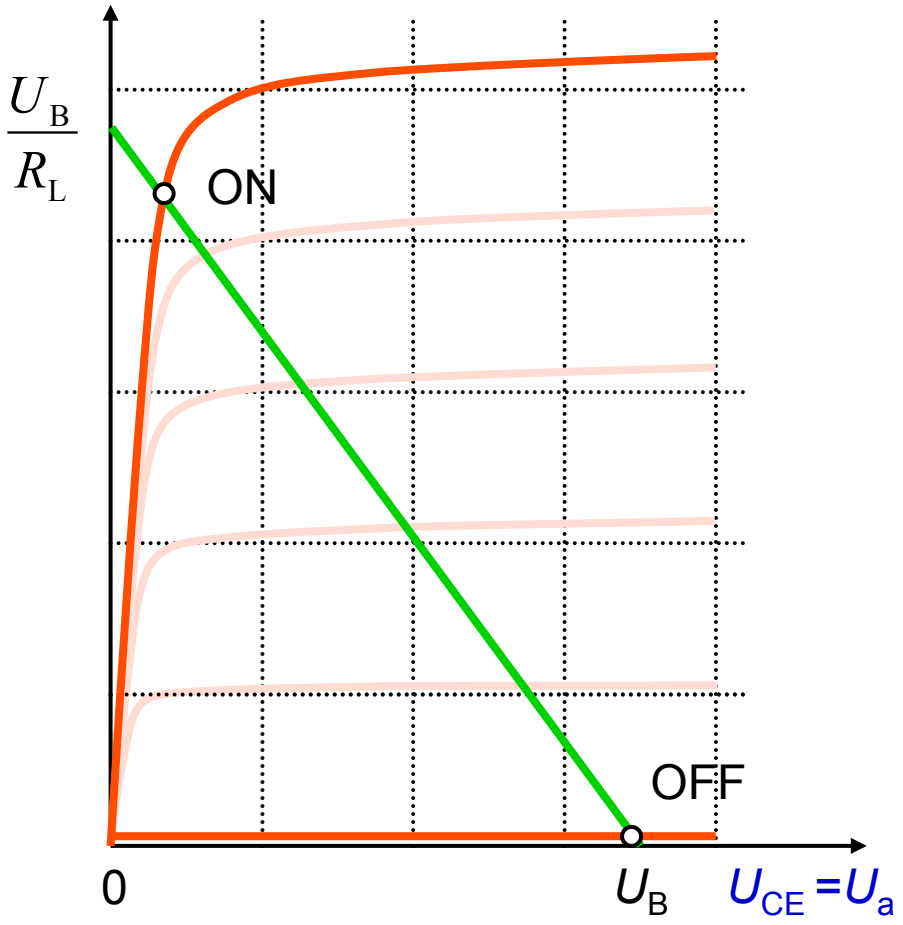




Inverter mit Bipolartransistor



Inverterschaltung
mit bipolarem Transistor



Kennlinienfeld des Bipolartransistors
mit Lastgeraden für R_L

Arbeitspunkte des Bipolar-Inverters

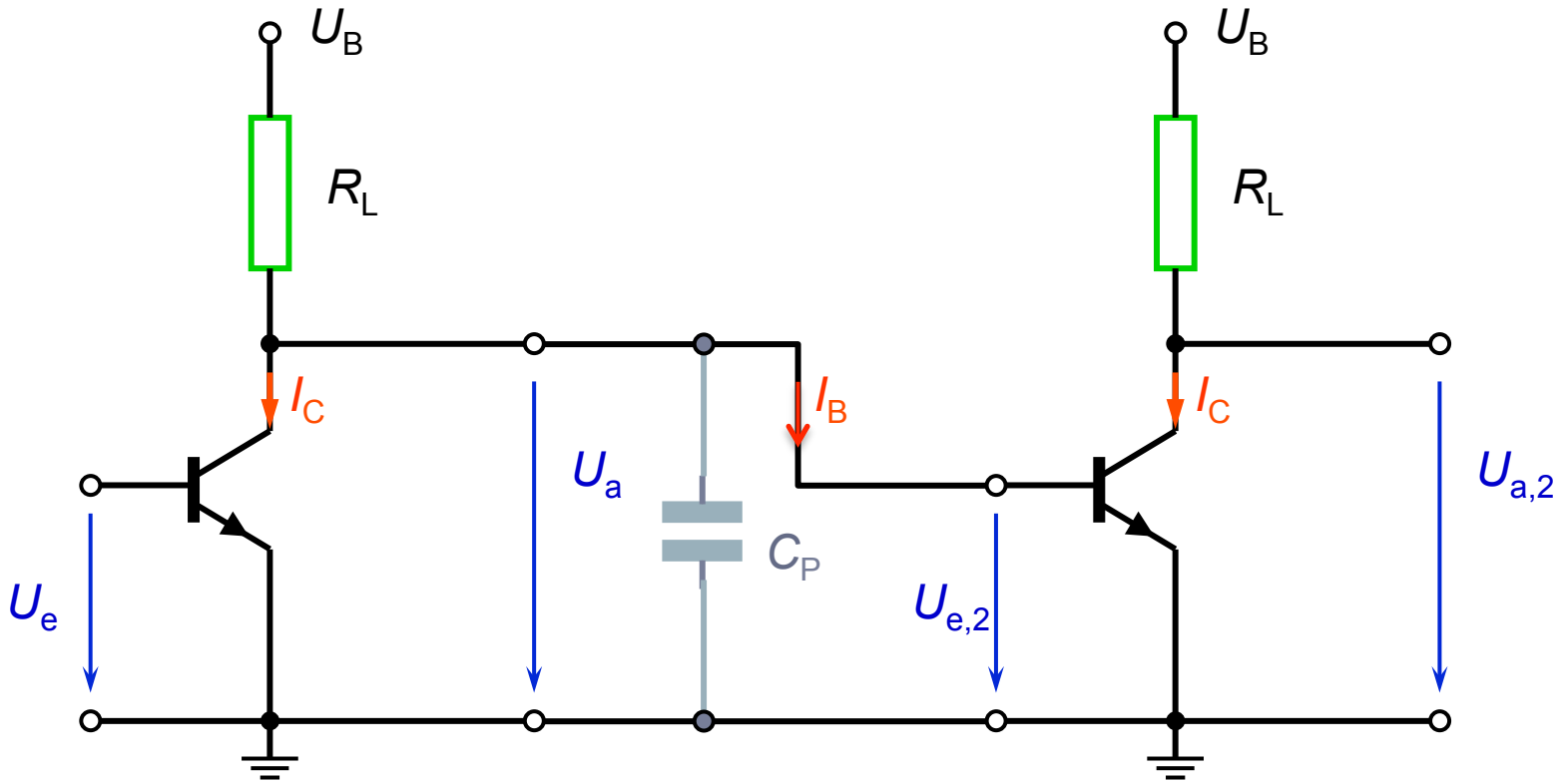
Arbeitspunkt "OFF":

Liegt an den Eingangsklemmen des Inverters ein **Lo**-Pegel (d.h. für $U_e = U_{BE} < U_S \approx 0{,}7V$), so sperrt der Transistor. Es fließt ein vernachlässigbar kleiner Kollektorstrom. Am Lastwiderstand R_L fällt keine nennenswerte Spannung ab. An den Ausgangsklemmen stellt sich der hohe Spannungspegel **Hi** ein (im Sperrbereich des Kennlinienfeldes).

Arbeitspunkt "ON":

Liegt an den Eingangsklemmen des Inverters ein **Hi**-Pegel (d.h. für $U_e = U_{BE} > 0{,}7V$), so leitet der Transistor. Der Lastwiderstand begrenzt dabei den Strom. (Würde der Transistor ideal leiten, so wäre der Strom $I_C = U_B / R_L$.) Der Transistor befindet sich im gesättigten Bereich (die Basis-Kollektor-Diode ist in Flussrichtung gepolt). Es fällt die Sättigungsspannung $U_{CE,sat}$ über dem Transistor ab. Diese Spannung ist kleiner als die Schleusenspannung U_S und entspricht damit einem **Lo**-Pegel. Ein an die Ausgangsklemmen angeschlossener Bipolartransistor (z.B. der eines nachfolgenden Gatters) würde also sperren.

Hintereinanderschaltung von Logikgattern

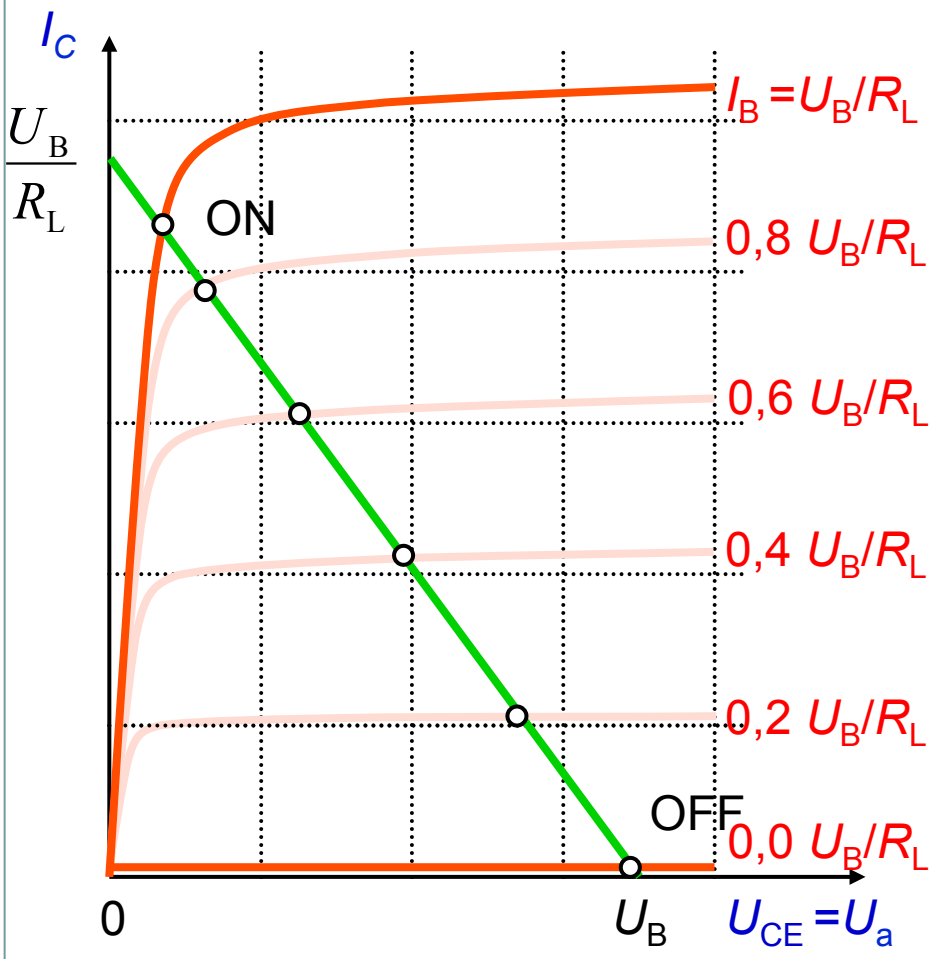


Inverter in Bipolar-Technik

Nachgeschalteter zweiter Inverter

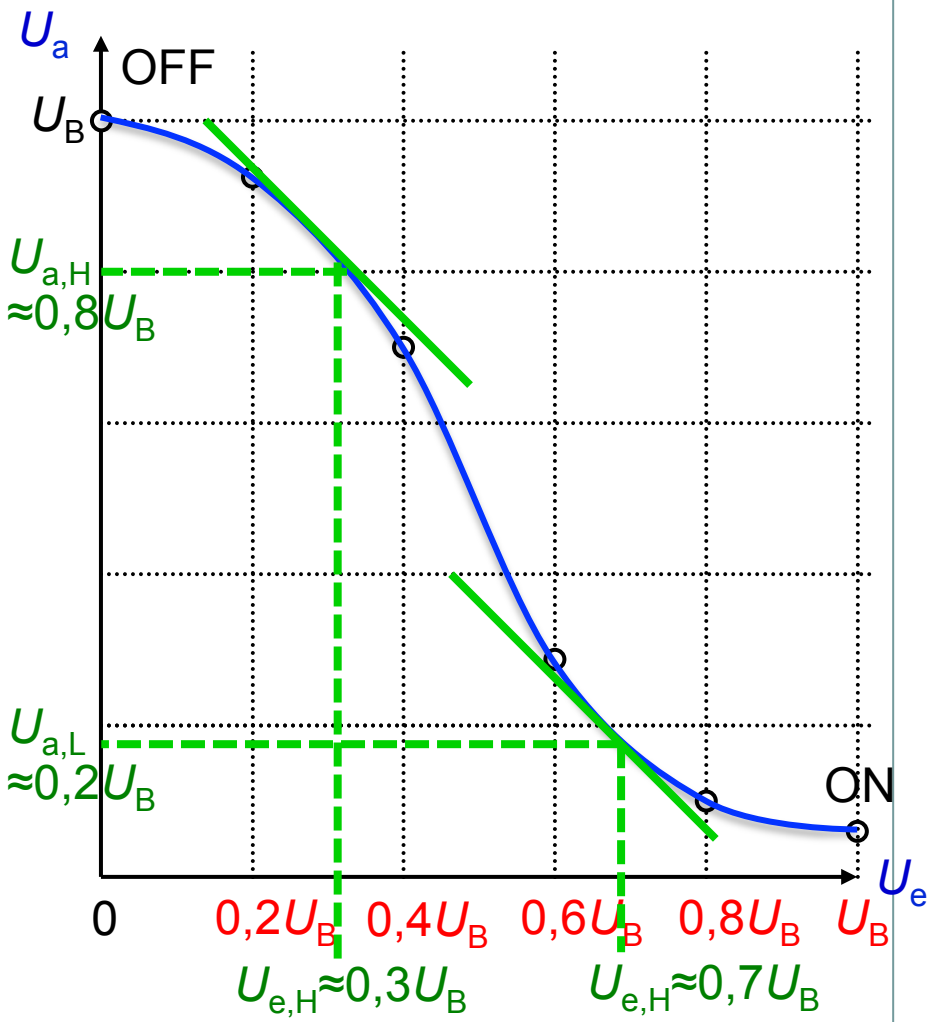
Der Ausgang eines Gatters muss in der Lage sein, nachfolgende Gatter korrekt anzusteuern. Das bedeutet, dass die Ausgangsspannung U_a gültige **Hi**- und **Lo**-Spannungspegel erzeugen muss.

Übertragungskennlinie des Bipolar-Inverters



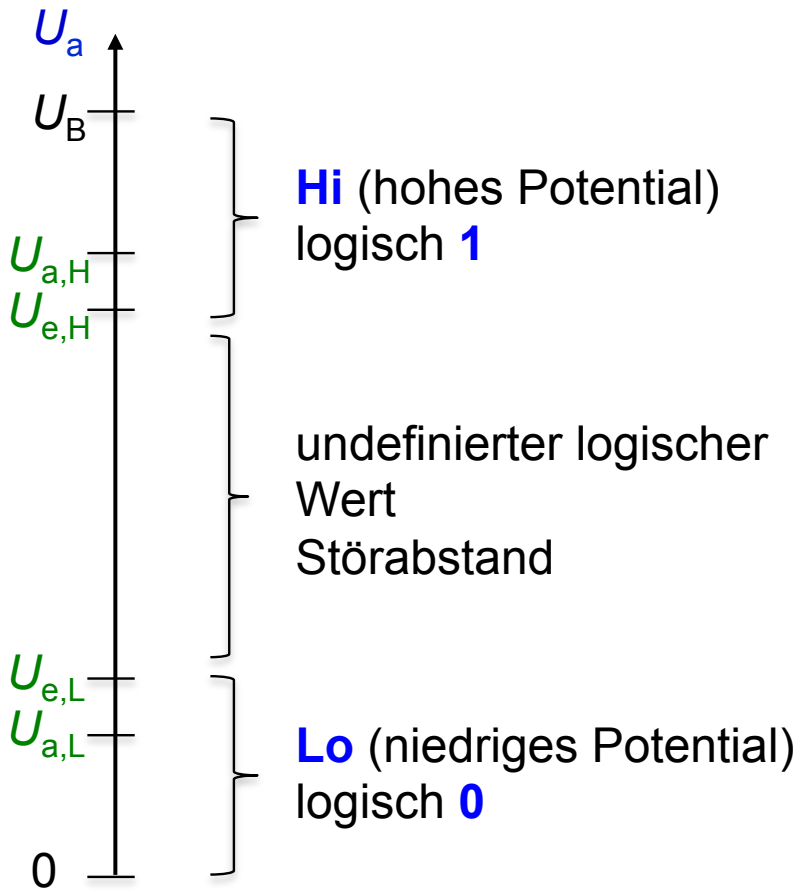
Kennlinienfeld des Bipolartransistors mit Lastgeraden für R_L

=>



Übertragungskennlinie des Bipolar-Inverters

Signalpegel



Zuordnung zwischen Spannungswerten **Lo**, **Hi** und logischen Werten **0**, **1**

Theoretisch / ideal:
0V = logisch 0
U_B = logisch 1

In der Praxis:
Spannungsbereiche
(s. links)

Am Ausgang: gegenüber dem Eingang verbesserte Spannungspegel (näher am Ideal), vergrößerter Störabstand

Verlustleistung & Verzögerung (qualitativ)

Verlustleistung

Annahme: Transistor schaltet ideal.
Falls Ausgang Hi: dauerhafter Strom I_B durch Basis des nachgeschalteten Transistors
Falls Ausgang Lo: dauerhafter Querstrom I_C durch den Inverter selbst

⇒ Zur Darstellung der logischen Werte fließen dauerhaft (statisch) Ströme

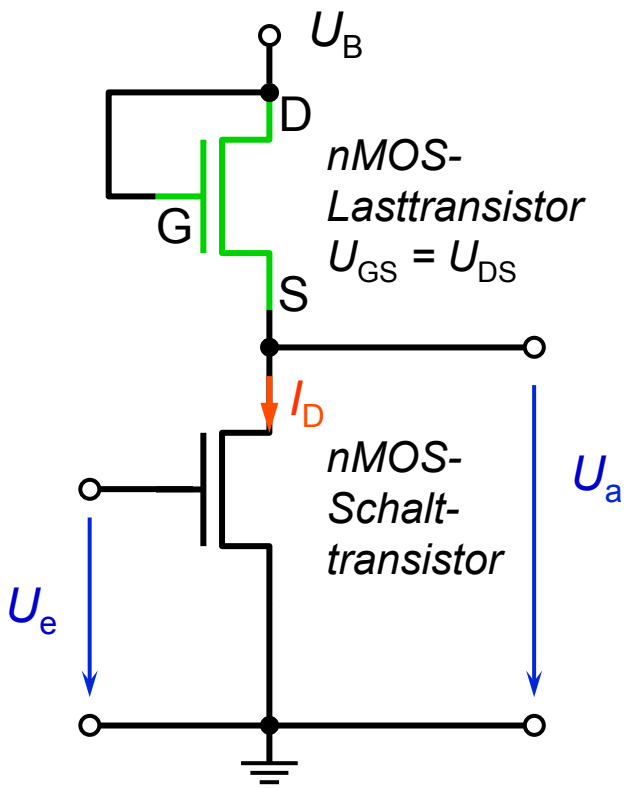
⇒ Verlustleistung $P = U_B \cdot \left(I_B \cdot p(Hi) + I_C \cdot p(Lo) \right)$
wobei $p(X)$ die Wahrscheinlichkeit des Signalpegels X ist.

Verzögerung

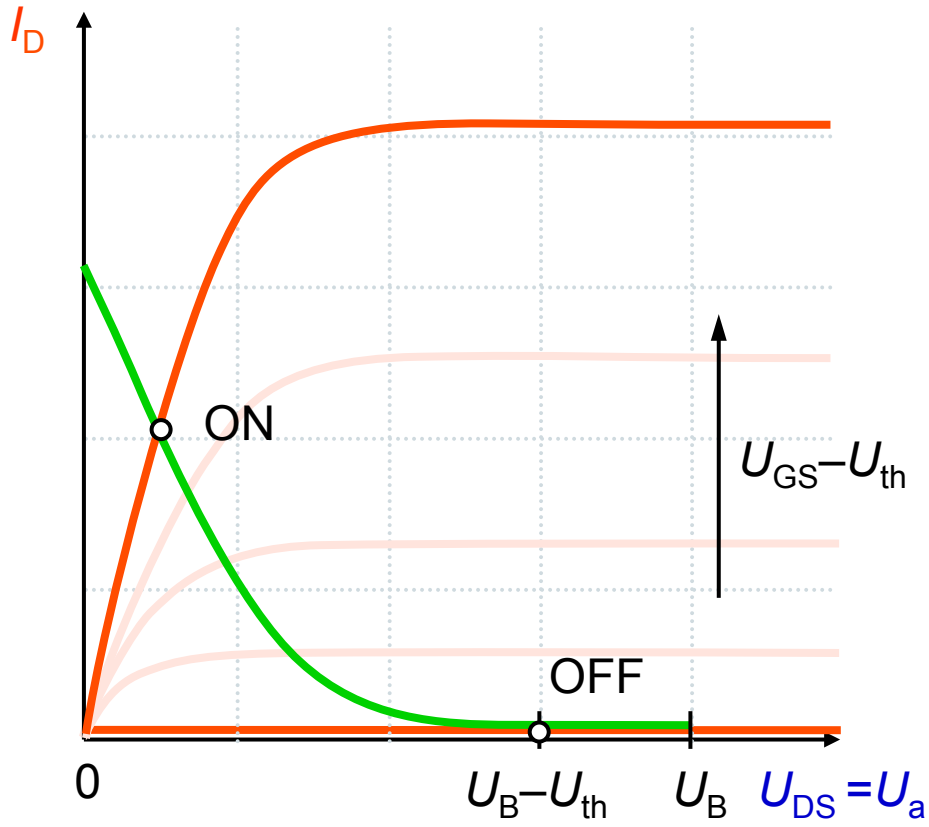
Eine Verzögerung beim Umschalten zwischen Lo und Hi entsteht durch die Notwendigkeit, parasitäre Kapazitäten z.B. von Leitungen (modelliert als C_P auf Folie 11) umzuladen:
Umladen Lo → Hi über Lastwiderstand mit Zeitkonstante $\tau_{LH} = R_L C_P$
Umladen Hi → Lo über Transistor (geringerer Widerstand R_T) $\tau_{HL} = R_T C_P < \tau_{LH}$

Die Verzögerung ist also unsymmetrisch, für eine *steigende Flanke* (Lo → Hi) größer als für eine *fallende Flanke* (Hi → Lo). Beim Entwurf muss zwischen den Flanken differenziert werden oder vom worst-case ausgegangen werden.

Inverter in nMOS-Technik



Inverterschaltung
in statischer nMOS-Technik
mit Lasttransistor



Kennlinienfeld des nMOSFET mit
Lastkennlinie des Lasttransistors

Arbeitspunkte des nMOS-Inverters

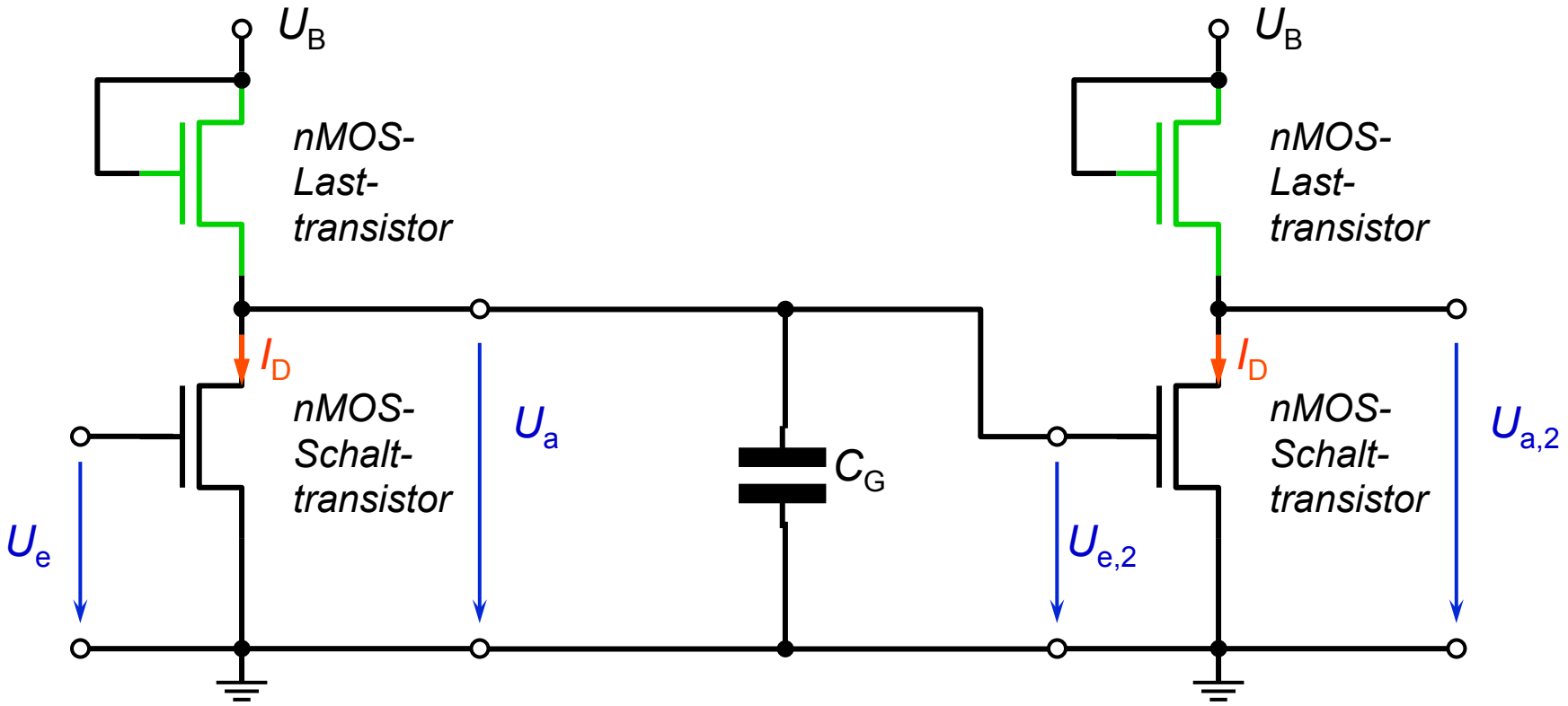
Arbeitspunkt "OFF":

Liegt an den Eingangsklemmen des Inverters ein **Lo**-Pegel (d.h. für $U_e = U_{GS} < U_{th}$), so sperrt der Schalttransistor. Der Lasttransistor lädt die nachfolgende Eingangskapazität C_G. Dabei steigt die Ausgangs-spannung an, allerdings nicht bis U_B sondern nur bis $U_B – U_{th}$, denn der Lasttransistor sperrt ebenfalls, sobald seine Gate-Source-Spannung unter die Schwellspannung fällt. Der Drainstrom des Schalttransistors I_D ist 0. Der Endwert der Ausgangsspannung $U_a = U_B – U_{th}$ ist ein **Hi**-Pegel.

Arbeitspunkt "ON":

Liegt an den Eingangsklemmen des Inverters ein **Hi**-Pegel (d.h. für $U_e = U_{GS} > U_{th}$), so leitet der Schalttransistor. Der Lasttransistor leitet ebenfalls. Über die Wahl der Kanallängen L und Kanalbreiten W der beiden Transistoren können ihre Widerstände eingestellt werden. Der Last-transistor muss einen hohen, der Schalttransistor einen niedrigen Widerstand besitzen, so dass im Arbeitspunkt "ON" die Ausgangsspannung U_a kleiner ist als die Schwellspannung U_{th} des Eingangstransistors des nachfolgenden Gatters. Diese Ausgangsspannung stellt einen **Lo**-Pegel dar.

Hintereinanderschaltung von nMOS-Gattern



Inverter in nMOS-Technik

nachgeschalteter zweiter Inverter

Die Ausgangsspannung des ersten Inverters muss zwar keinen stationären Strom an das nachgeschaltete Gatter liefern. Sie muss allerdings die Gate-Kapazität C_G des nachgeschalteten Transistors aufladen. Diese Kapazität wird durch den eingezeichneten Kondensator modelliert.

Verlustleistung und Verzögerung (qualitativ)

Verlustleistung

Annahme: Transistor schaltet ideal.
Falls Ausgang Hi: *vorübergehender* Strom, um C_G aufzuladen
Falls Ausgang Lo: dauerhafter Querstrom I_D durch den Inverter selbst

⇒ Teilweise Verbesserung ggü. Bipolar-Inverter

⇒ Verlustleistung $P \approx U_B \cdot \left(C_G \cdot U_B \cdot f(Lo \rightarrow Hi) + I_D \cdot p(Lo) \right)$
wobei *f*(*Lo*→ *Hi*) die Frequenz (Anzahl pro Sekunde) der steigenden Flanke ist.

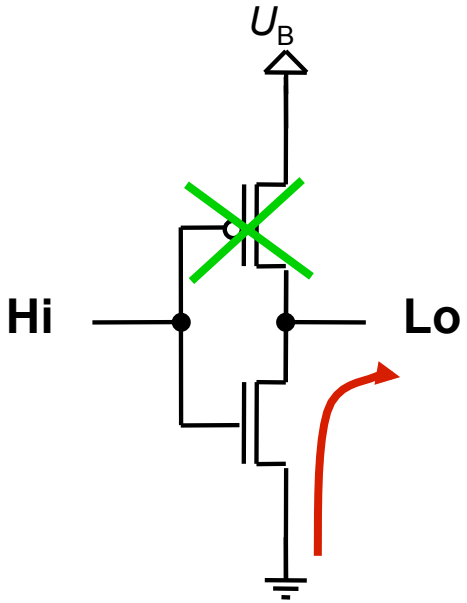
Verzögerung

Wie beim Bipolar-Inverter erfordert das Aufladen von C_G über den Lasttransistor aufgrund des höheren Widerstands mehr Zeit als das Entladen über den Schalttransistor.

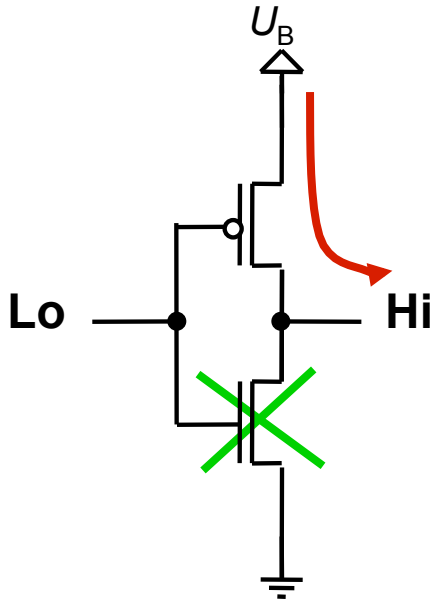
Zur Verringerung der Verzögerung bietet sich an, den Widerstand des Lasttransistors zu verkleinern. Dann steigt jedoch der Querstrom und die Verlustleistung. => *Trade-off*: Verzögerung (Latenz) vs. Verlustleistung

Idealerweise wäre der Widerstand des Lasttransistors variabel => **CMOS**

Inverter mit nMOS- und pMOS-Transistor

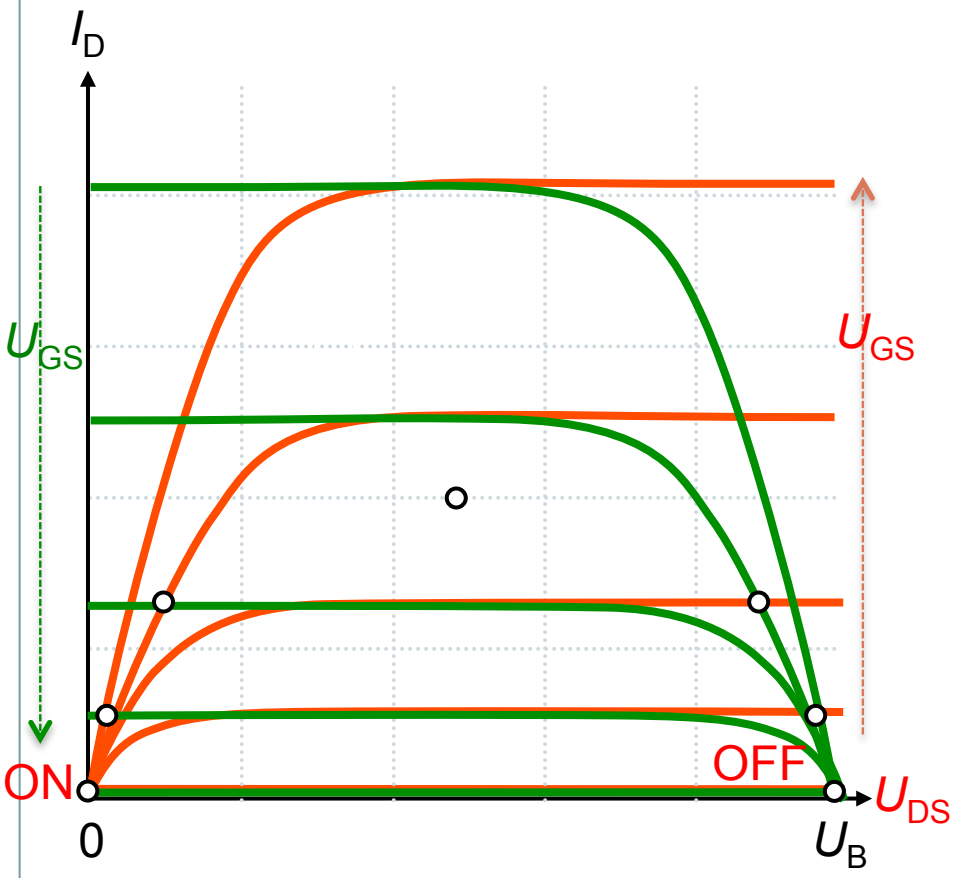


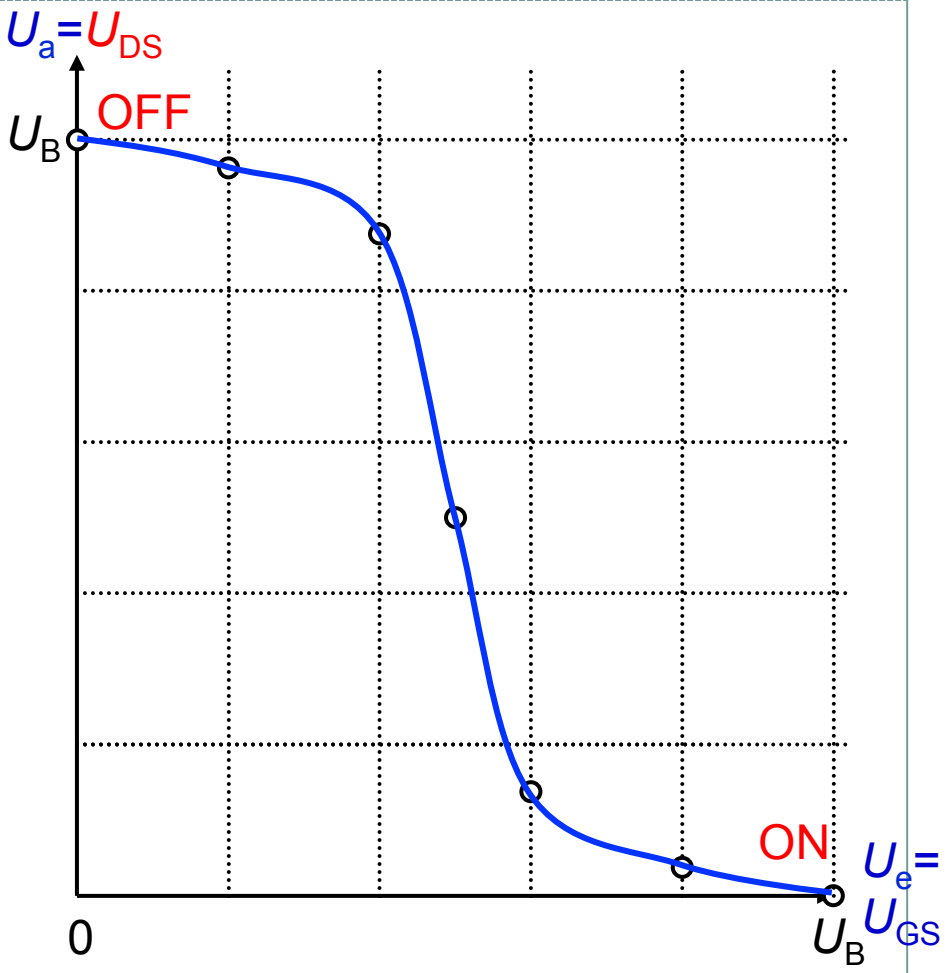
Eingang ist **Hi**
Lasttransistor sperrt
Schalttransistor leitet
Ausgang mit **Lo** verbunden



Eingang ist **Lo**
Lasttransistor leitet
Schalttransistor sperrt
Ausgang mit **Hi** verbunden

Arbeitspunkte des CMOS-Inverters



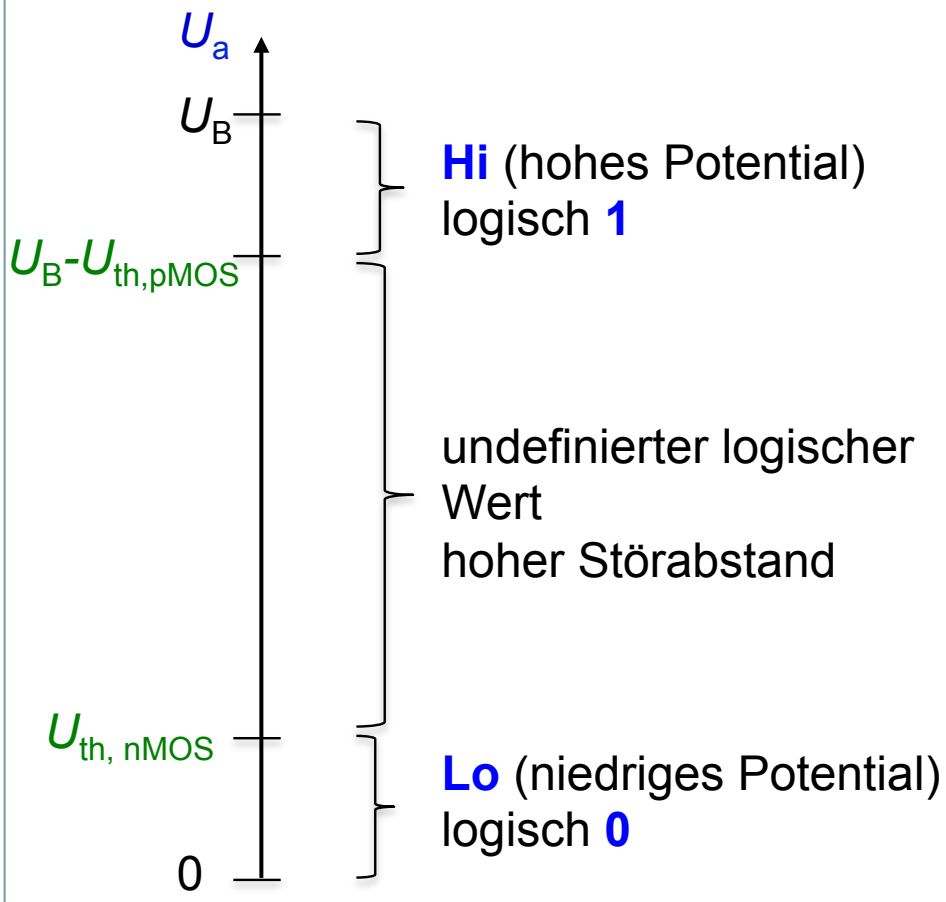


Ausgangskennlinienfeld
PMOS **NMOS**

=>

Übertragungskennlinie

Signalpegel in CMOS-Logik



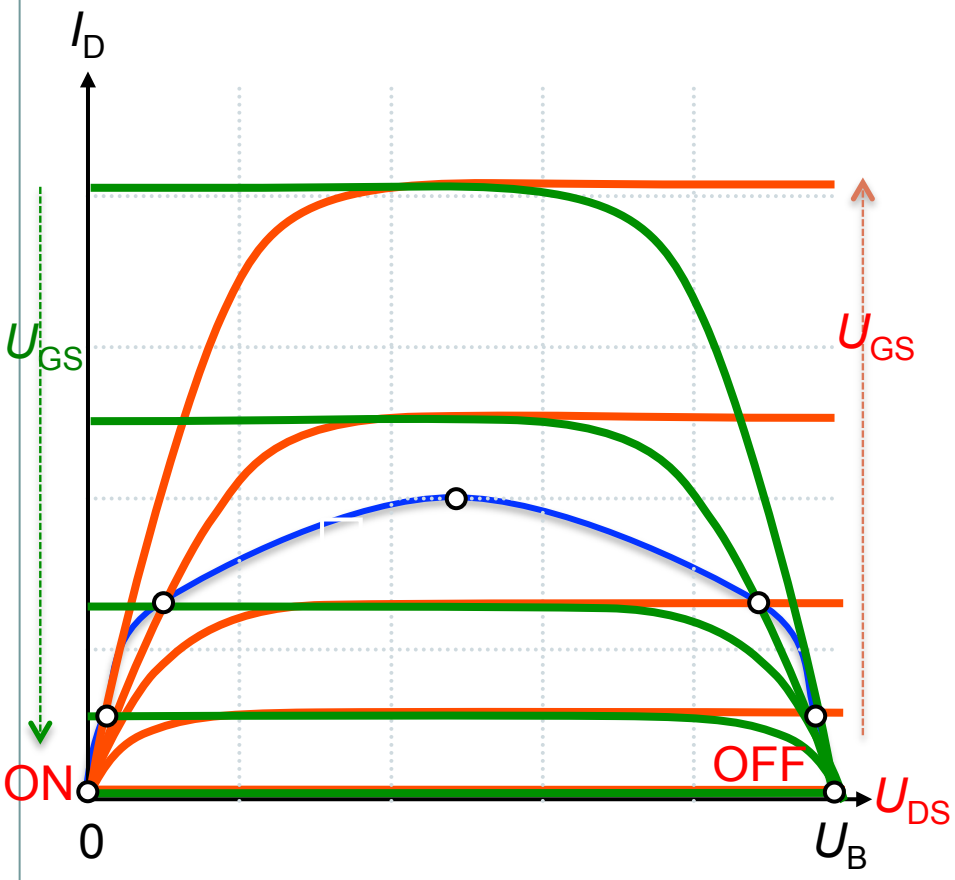
Die Pegelbereiche Lo, Hi könnten größer gewählt werden, aber dann würde ein statischer Querstrom durch den Inverter fließen. Stattdessen:

Hi-Potentialbereich (am Eingang) ist nicht größer als $U_{th,pMOS}$
⇒ pMOS sperrt, nMOS leitet
⇒ Lo-Potential am Ausgang

Lo-Potentialbereich (am Eingang) ist nicht größer als $U_{th,nMOS}$
⇒ nMOS sperrt, pMOS leitet
⇒ Hi-Potential am Ausgang

Verlustleistung des CMOS-Inverters

In den Arbeitspunkten ON und OFF fließt (im Ggs. zu bipolar, nMOS) kein Strom mehr. Die logischen Zustände 0 und 1 können ohne Stromfluss eingenommen werden. *Idealisiert* fließen Ströme nur beim Übergang 0 ←→ 1 bzw. ON ←→ OFF:



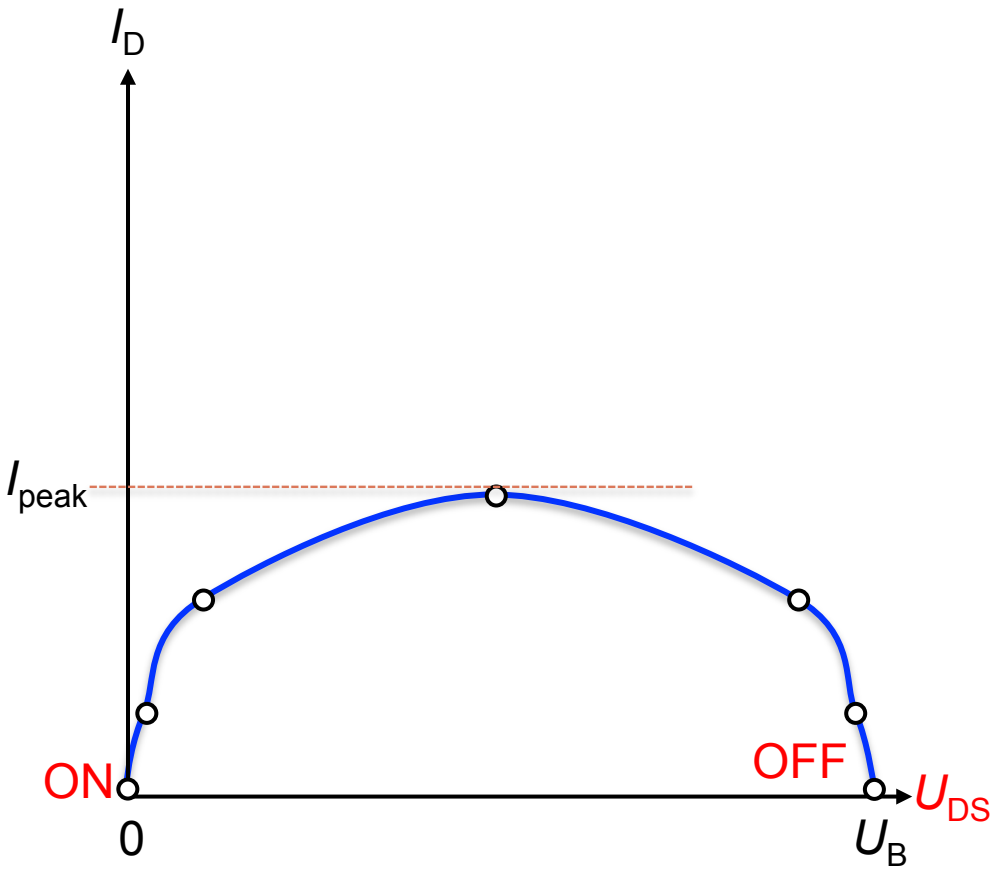
- Kurzschluss-Strom I_{sc} von U_B zu Masse (0V) in dem kurzen Zeitraum, wenn beide Transistoren leiten;
- Dynamischer Umlade-Strom I_{dyn} durch einen der Transistoren, um die Kapazität C_G des nächsten Gatters aufzuladen bzw. zu entladen.

In der *Realität* kommt ein statischer Leckstrom I_{leak} u.a. aufgrund von nicht-idealen Isolierungen hinzu.

Die Verlustleistung P_{ges} besteht entsprechend aus 3 Komponenten:

$$P_{ges} = P_{sc} + P_{dyn} + P_{leak}$$

Kurzschluss-Strom im CMOS-Inverter



- Maximalstrom I_{peak}
- Betriebsspannung U_B
- Dauer des Umschaltvorgangs (während dieser Zeit leiten beide Transistoren) t_{sc}

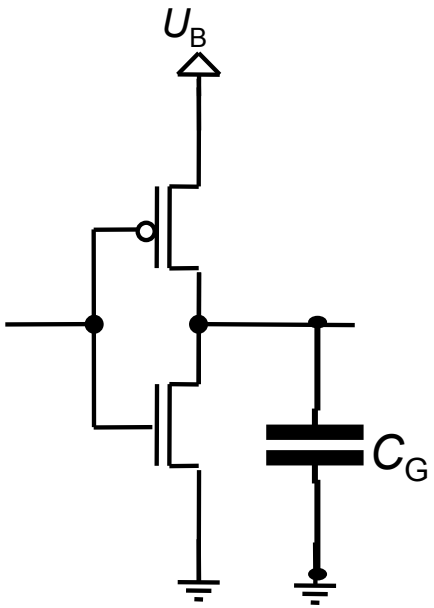
⇒ Energie für 1x Umschalten ist $E_{sc} \approx I_{peak} U_B t_{sc}$

- Schaltfrequenz f (Umschaltvorgänge pro Sekunde)

⇒ durchschnittliche Verlustleistung $P_{sc} = E_{sc} f \approx I_{peak} U_B t_{sc} f$

Verringerung durch Senken von U_B und/oder schnelles Umschalten t_{sc} (erfordert hohe Flankensteilheit am Eingang).

Umlade-Strom im CMOS-Inverter



- Lastkapazität (i.W. der Transistor-Gates nachfolgender Logikgatter) C_G
- Betriebsspannung U_B
- Energie im Kondensator lt. Kap. 5, Folie 31

⇒ Energie für 1x Umladen ist
$E_{dyn} = ½\ C_G U_B^2$

- Umschaltfrequenz f
(Anzahl 0 → 1 → 0 pro Sekunde)

⇒ durchschnittliche Verlustleistung
$P_{dyn} = 2E_{dyn} f = C_G U_B^2 f$

Verringerung durch Senken von U_B (wirkt quadratisch!), Verkleinerung der Schaltungen (dadurch sinkt C_G) oder Codes, die die Umschaltfrequenz f verringern (z.B. Gray-Code).

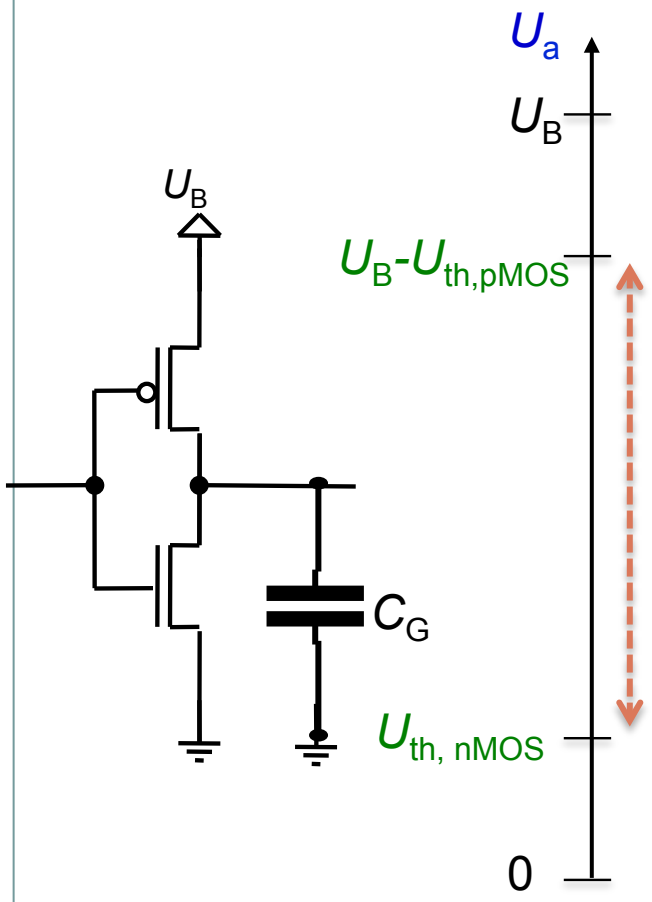
Statischer Leckstrom

I_{leak} und damit $P_{leak} = U_B I_{leak}$ entstehen aus einer Reihe von Vorgängen in den MOS-Transistoren:

- *subthreshold currents* zwischen Source und Drain
- *gate induced drain leakage currents*: von Drain zu Bulk aufgrund leichter Überlappung zwischen Gate und Drain
- *tunneling* zwischen Gate und Halbleitermaterial (Elektronen überwinden die SiO_2-Isolierschicht des MOS-Kondensators)
- *punchthrough* zwischen Drain und Source
- *pn-junction leakage*

Die Leckströme sind für einzelne Transistoren gering, aber die Anzahl von Transistoren auf einem Chip ist von wenigen 1000 (1970) dank fortschreitender Miniaturisierung auf Milliarden gestiegen. Damit summieren sich die entstehenden Verlustleistungsbeiträge auf und machen in modernen Schaltungen bis zur Hälfte der Verlustleistung aus. Dazu trägt auch die geringere Dicke von Isolierschichten bei.

Verzögerung des CMOS-Inverters



Ein CMOS-Gatter reagiert am Ausgang mit einer kleinen Verzögerung auf Änderungen am Eingang. Die Verzögerung entsteht durch die Zeit für das Umladen der Kapazität C_G nachfolgender Gatter zwischen oberer Grenze des Lo-Potentials und unterer Grenze des Hi-Potentials.

Ausgang Lo → Hi:
Aufladen von C_G
über pMOS

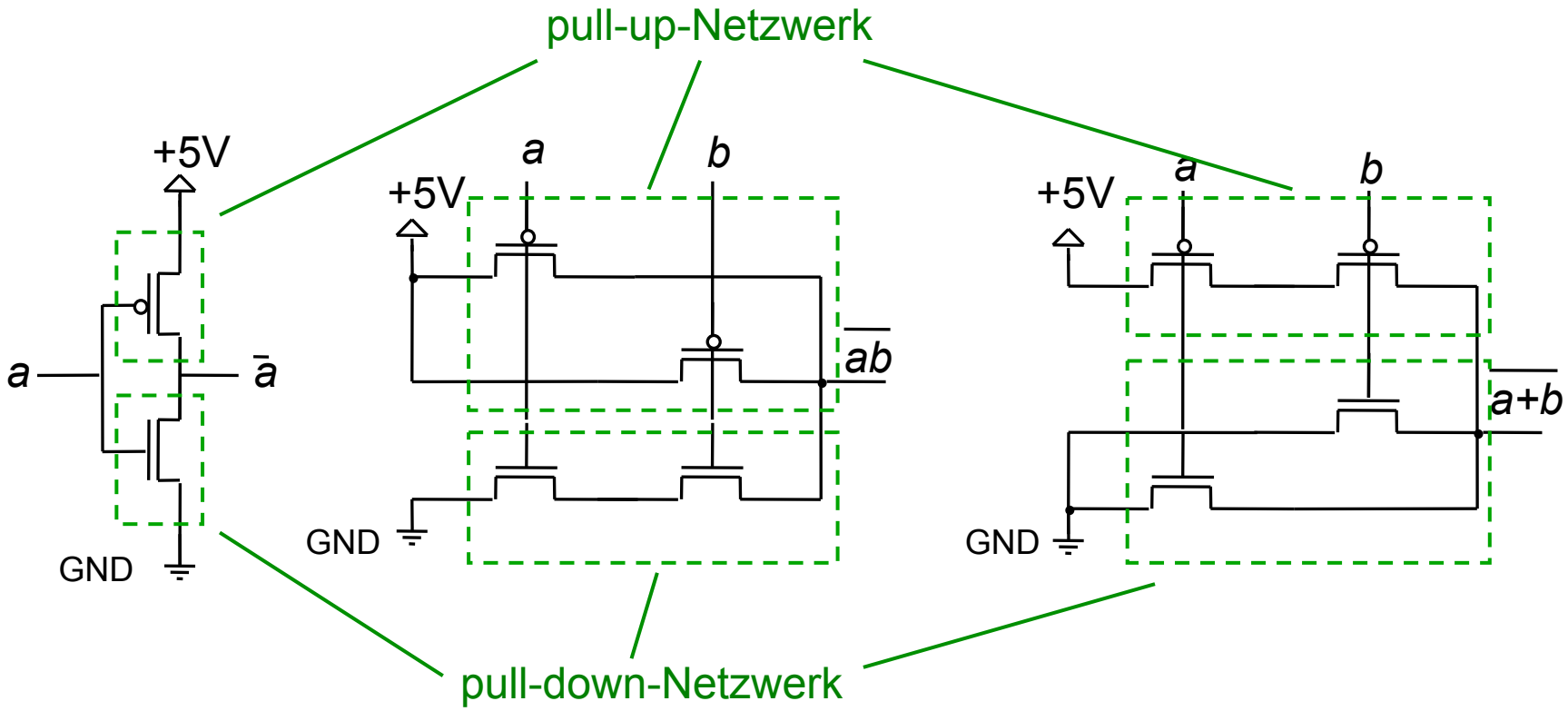
$$t_{LH} \approx 3\frac{C_L}{\beta_p U_B}$$

Ausgang Hi → Lo:
Aufladen von C_G
über nMOS

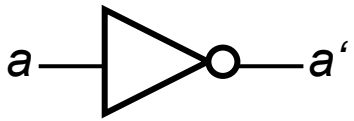
$$t_{HL} \approx 3\frac{C_L}{\beta_n U_B}$$

Damit $t_{LH} = t_{HL}$ wird, muss $\beta_p = \beta_n$ gelten. Die höhere Elektronenbeweglichkeit in β_n (im Vergleich zur Löcherbeweglichkeit in β_p) muss durch Anpassung der W/L Verhältnisse der Transistoren ausgeglichen werden (siehe Def. β in Kap. 7, Folie 43).

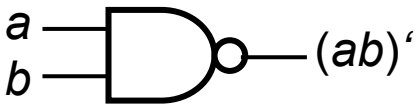
Implementierung Boole‘scher Verknüpfungen



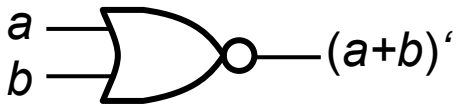
Inverter



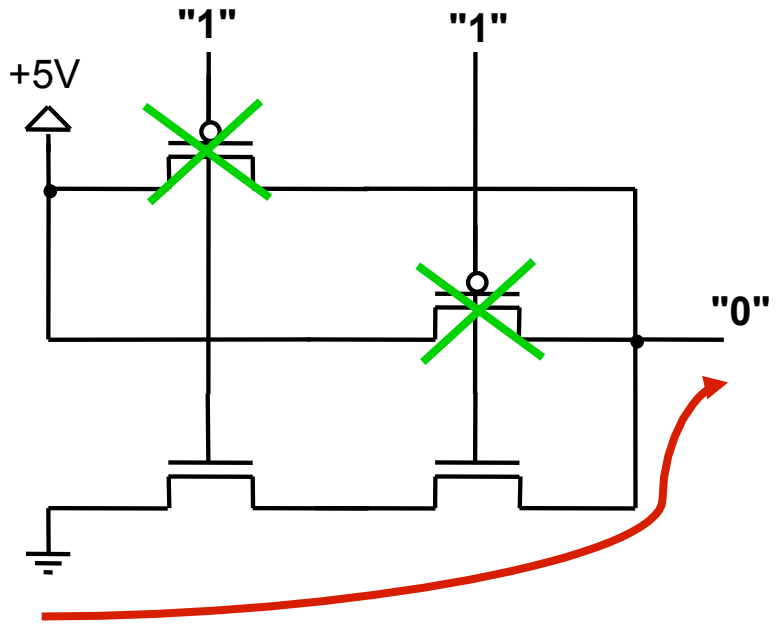
NAND-Gatter

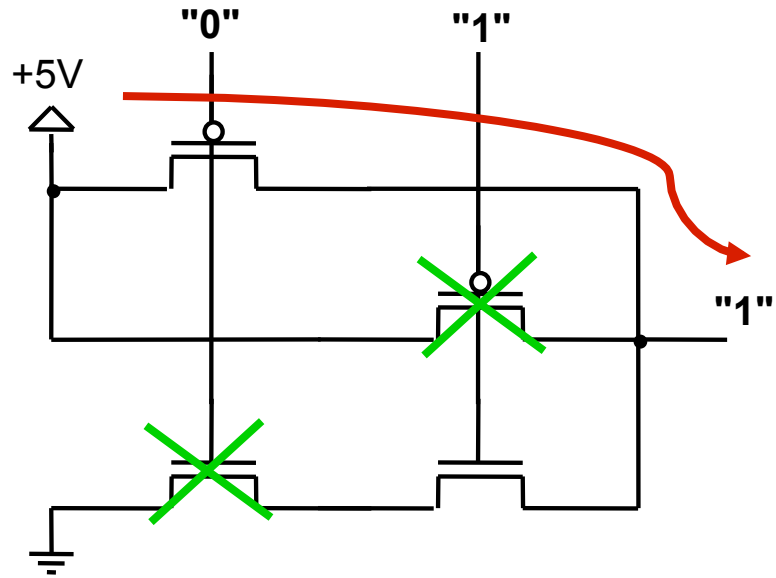


NOR-Gatter



Funktionsweise des CMOS-NAND-Gatters





a = 1, *b* = 1
Pull-up-Netzwerk sperrt
Pull-down-Netzwerk leitet
Ausgang mit **Lo** verbunden

a = 0, *b* = 1
Pull-up-Netzwerk öffnet Pfad zu **Hi**
Pull-down-Netzwerk hat keinen Pfad zu **Lo**
Ausgang mit **Hi** verbunden

Kann man das NAND-Gatter in ein AND-Gatter verwandeln, indem man im Transistornetzwerk des NAND-Gatters +5V und Masse vertauscht? **Nein!**

Prinzipien der CMOS-Digitalschaltungen

Für den pMOS-Transistor gilt Entsprechendes, allerdings jeweils für den komplementären Signalpegel. Zusammenfassend kann man sagen:

- Ein nMOS-Transistor leitet den Lo-Pegel ohne Pegelverlust weiter (aber nicht den Hi-Pegel), d.h. er eignet sich als **pull-down**-Transistor.
- Ein pMOS-Transistor leitet den Hi-Pegel ohne Pegelverlust weiter (aber nicht den Lo-Pegel), d.h. er eignet sich als **pull-up**-Transistor.

"Pull-up-Netzwerk": besteht aus pMOS-Transistoren

"Pull-down-Netzwerk": besteht aus nMOS-Transistoren

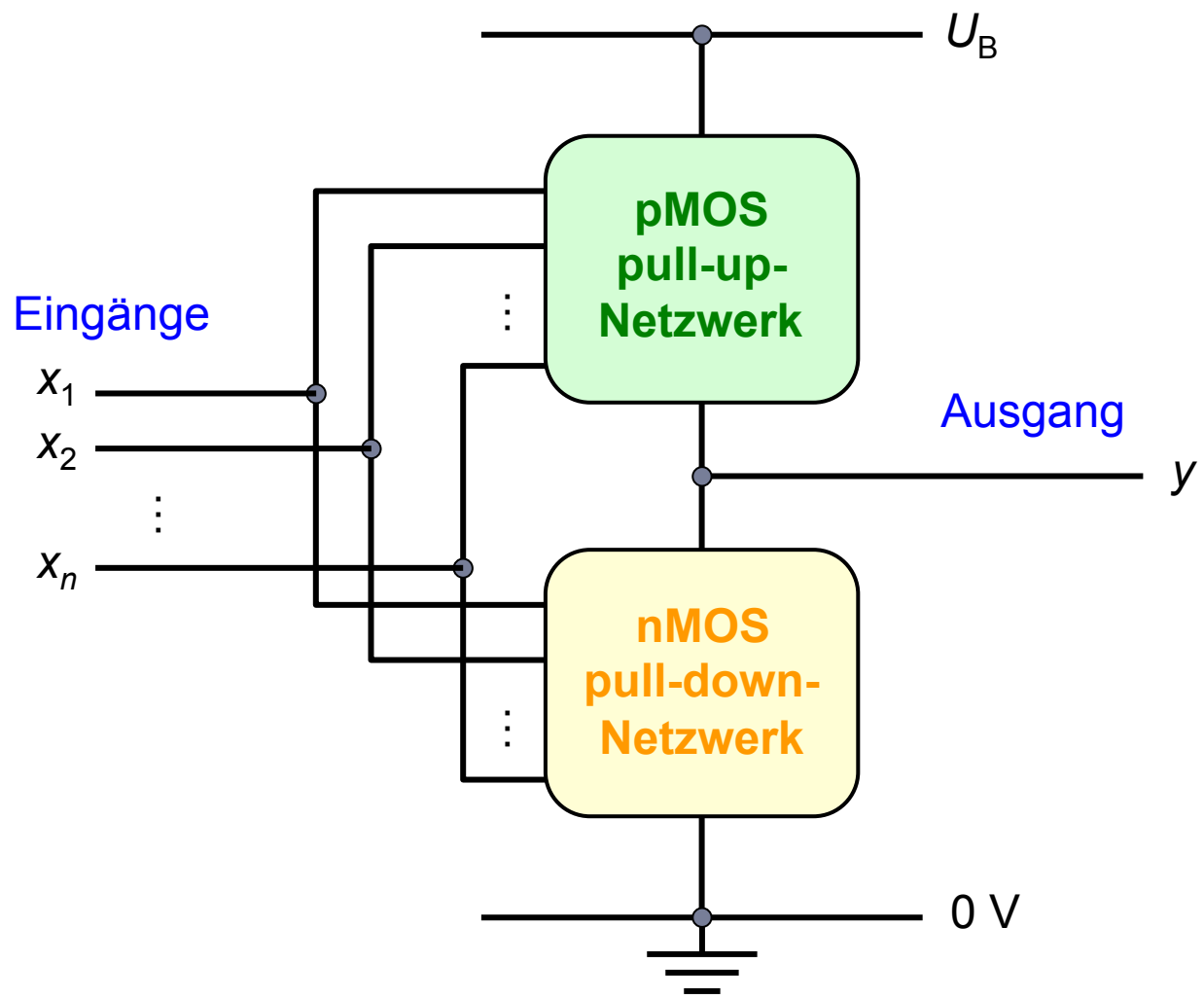
Ein AND-Gatter kann daher in CMOS-Technik nicht einfach durch Vertauschung von Masse und Betriebsspannung aus einem NAND-Gatter gewonnen werden.

Ein AND-Gatter muss durch NAND und Inverter realisiert werden.

Komplementärer Aufbau von Pull-up- und Pull-down-Netzwerk:
("complementary MOS", "CMOS")

- Einer Reihenschaltung im pull-up-Netzwerk entspricht eine Parallelschaltung im pull-down-Netzwerk und umgekehrt.
- Für jede Signalbelegung existiert ein Pfad **entweder** nach **Hi** **oder** nach **Lo**.

Allgemeiner Aufbau von CMOS-Logikgattern



CMOS-Komplexgatter

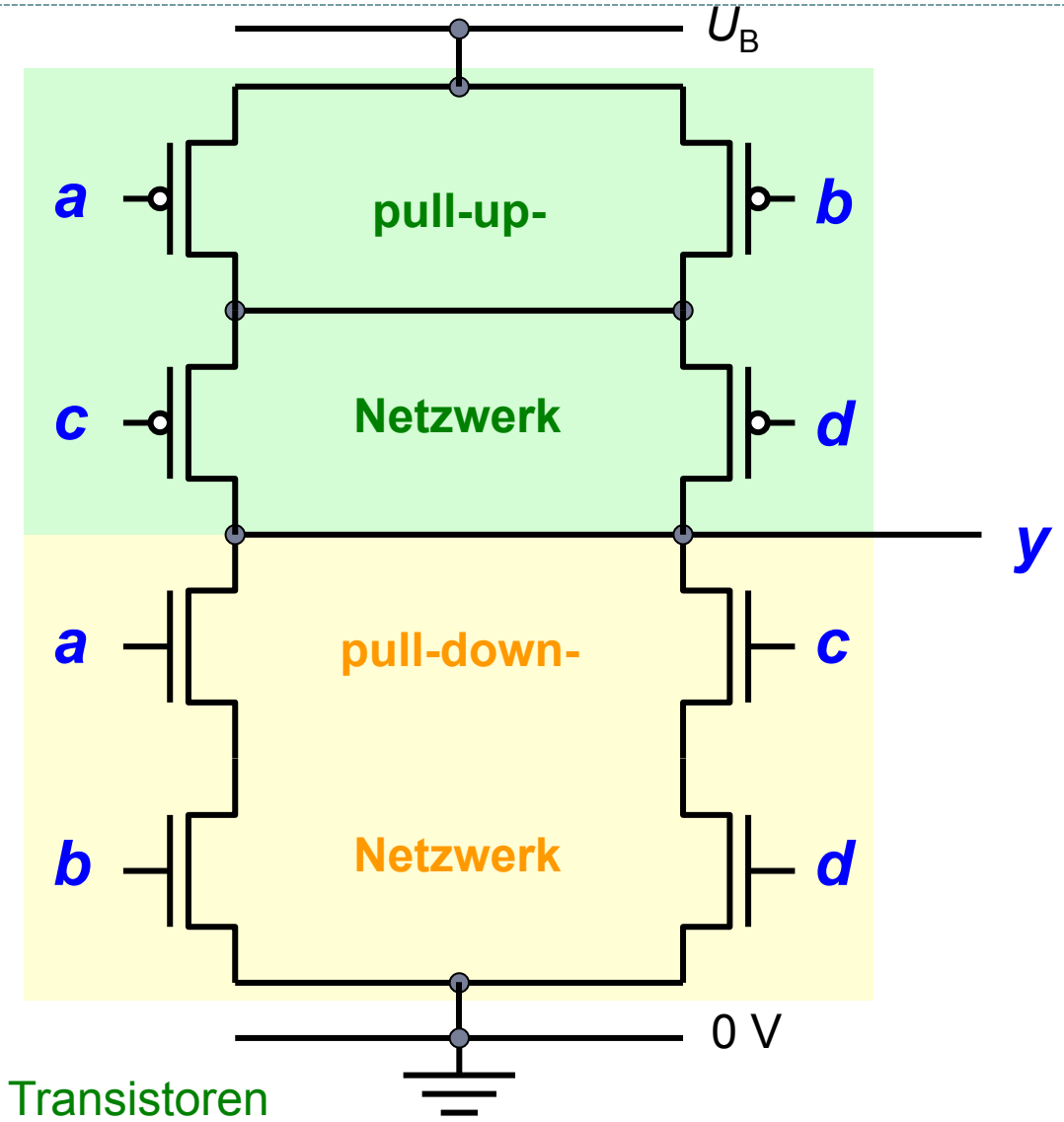
Beispiel:
Implementierung der Schaltfunktion

$$y = \overline{(a \cdot b) + (c \cdot d)}$$

Komplexgatter:

Schaltung realisiert eine komplexe Schaltfunktion.

Hier: Funktion genannt "AND-OR-INVERT-22" oder "AOI22"

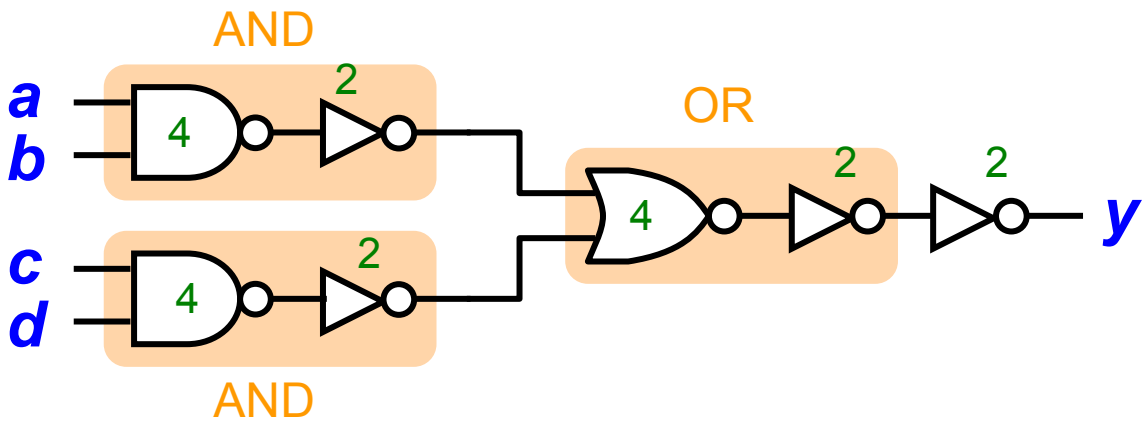


Implementierungsaufwand: 8 Transistoren

Vergleich mit mehrstufiger Logik

(Beispiel)

Ineffiziente Implementierung derselben Schaltfunktion aus einzelnen CMOS-Primitivgattern:



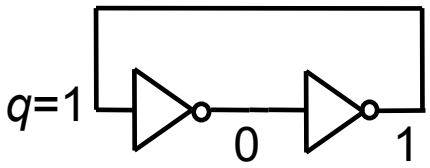
Implementierungsaufwand: 20 Transistoren

Mit Komplexgattern (engl. compound gates) lassen sich bestimmte Funktionen kompakt und effizient implementieren.

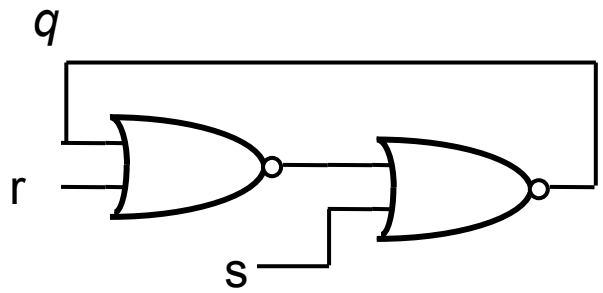
Grundprinzip der Speicherung binärer Werte

Speicherung ist die Übertragung eines Werts *q* vom Ursprungsort, so dass er *am Ursprungsort* zu *einer späteren Zeit* verfügbar ist.

Spätere Zeit => **Verzögerung** erforderlich, z.B. durch 2 aufeinanderfolgende Inverter
Gleicher Ort => **Rückkopplung** (Rückführung zum Ursprungsort) erforderlich.



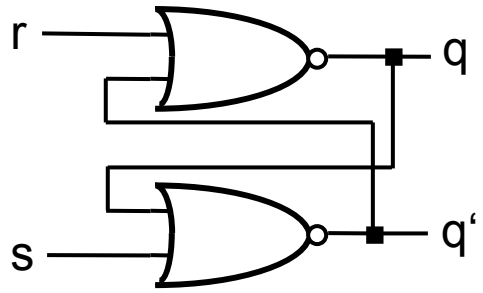
Beispiel: kaskadierte Inverter, speichert Wert 1
Problem: Wie kann der gespeicherte Wert geändert werden?



wie kaskadierte Inverter,
allerdings mit der Fähigkeit,
den Wert *q* auf 0 (reset)
oder 1 (set) zu stellen

Asynchrones RS-Speicherelement

Alternative Anordnung („über Kreuz“) der letzten Schaltung



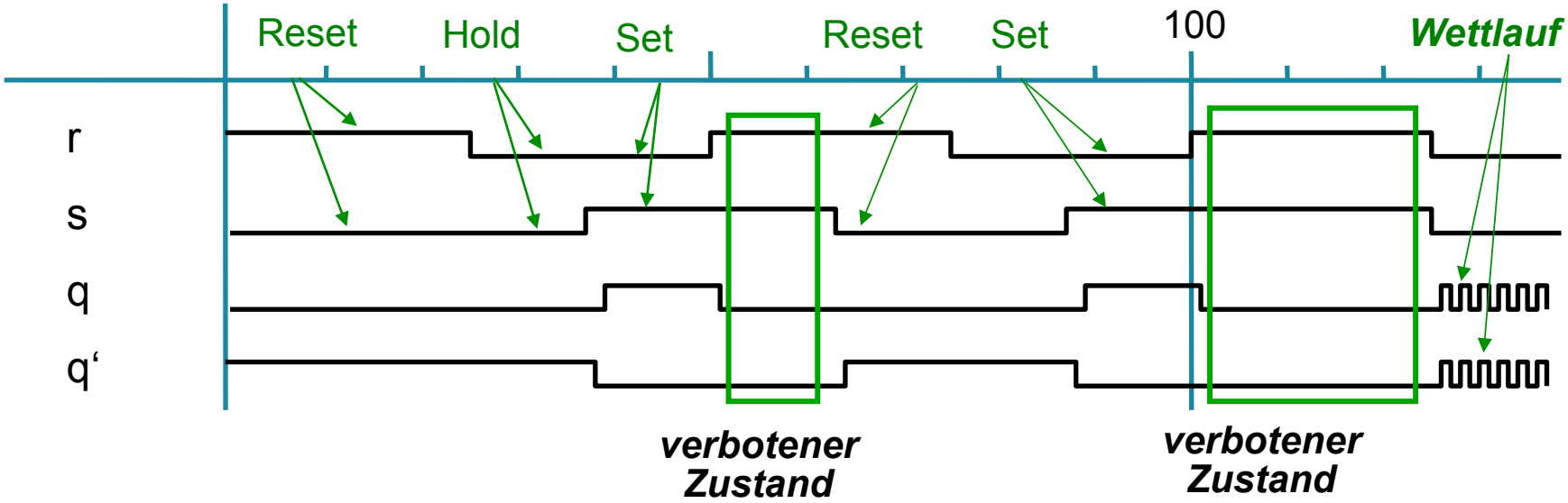
r = 0, s = 0 => wie kaskadierte Inverter, Wert *q* bleibt gespeichert

r = 1, s = 0 => Wert *q* := 0 (reset)

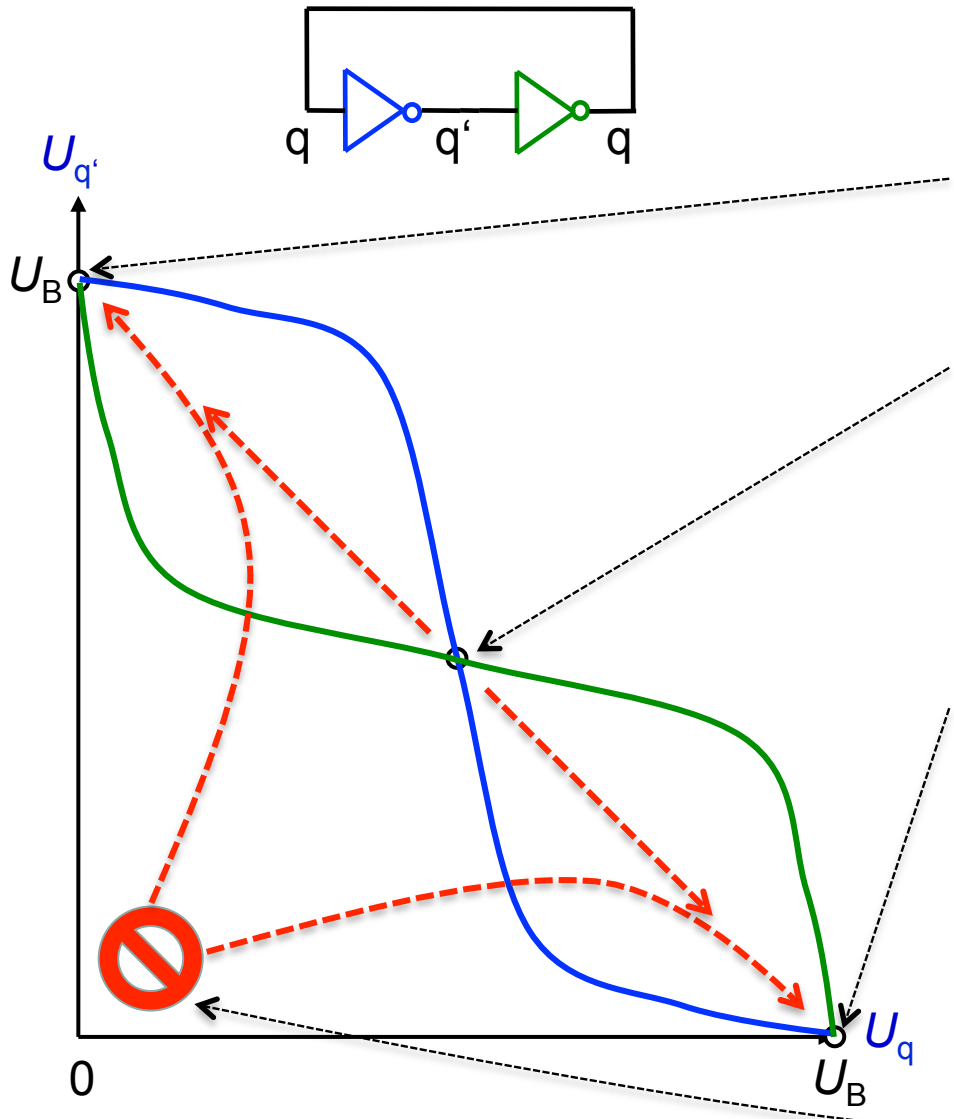
r = 0, s = 1 => Wert *q* := 1 (set)

r = 1, s = 1 => ???

Zeitdiagramm (x-Achse: Zeit, y-Achse: Signalverläufe)



Arbeitspunkte des Speicherelements



Stabiler Arbeitspunkt
Logische **0** gespeichert.

Metastabiler Zustand
Signalpegel „in der Mitte“ zwischen logisch 0 und 1 (ungültig).
Wird „zufällig“ in Richtung eines stabilen Arbeitspunkts verlassen

Stabiler Arbeitspunkt
Logische **1** gespeichert.

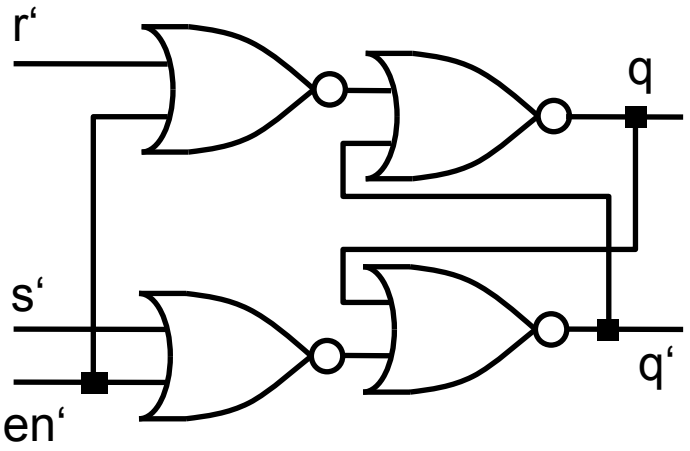
Verbotener Zustand
Kein Arbeitspunkt – Transition zu einem der beiden stabilen Arbeitspunkte.

Pegelgesteuertes RS-Speicherelement

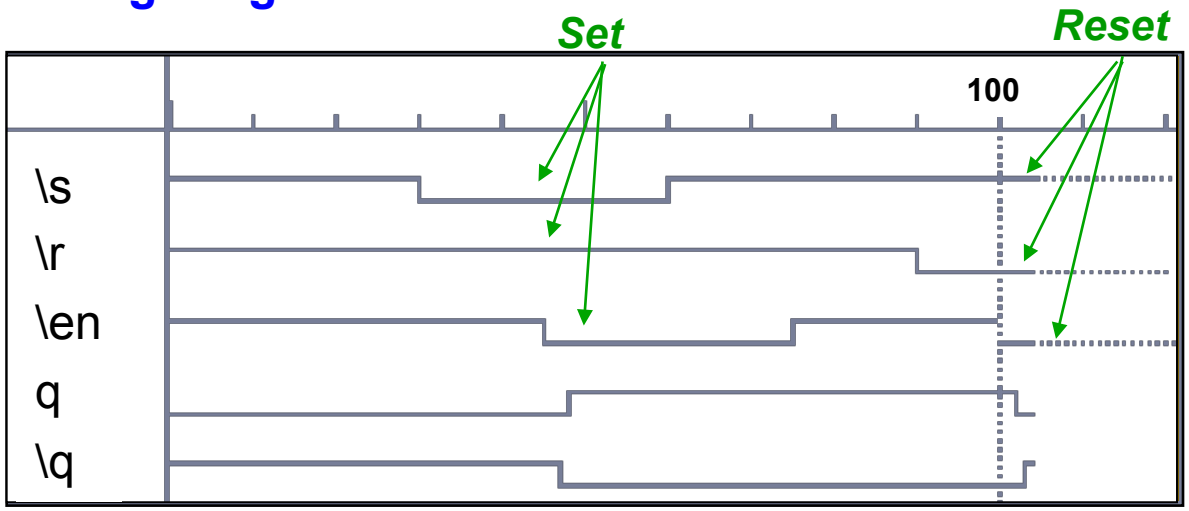
Funktionsweise:
Abhängig vom **Pegel** (Zustand)
von en‘

Solange en‘ = 0 („active low enable“),
ist set / reset möglich.

Sobald en‘ = 1, ist keine Änderung
des gespeicherten Werts mehr möglich.



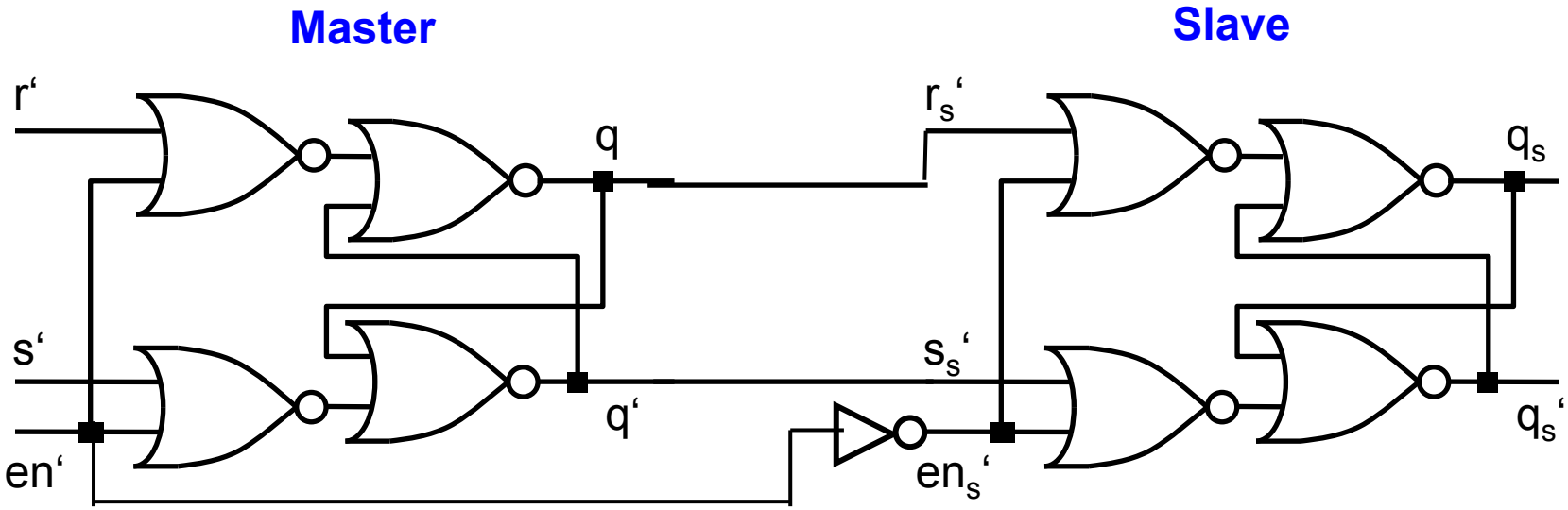
Timing-Diagramm:



Funktionstabelle:

en‘	r‘	s‘	q
0	0	0	?
0	0	1	0
0	1	0	1
0	1	1	q
1	–	–	q

Flankengesteuertes RS-Speicherelement



Solange en‘ = 0 („active low enable“), ist set / reset des Master-Elements möglich.

Beim Umschalten auf en‘ = 1 wird das Master-Element „disabled“, d.h. der zuvor eingestellte Wert wird gespeichert.

Währenddessen beim Slave en_s‘ = 1, d.h. „disabled“, speichert alten Wert.

Gleichzeitig beim Slave en_s‘ = 0, d.h. der vom Master gespeicherte Wert wird in den Slave übernommen.

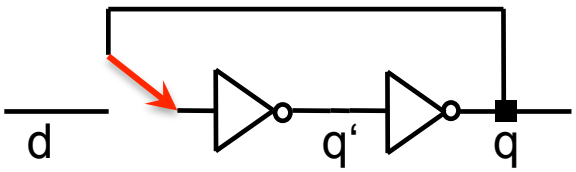
Am Ausgang des Slave-Elements wird bei steigender Flanke von en‘ der zuvor mittels r‘ und s‘ eingestellte Wert sichtbar.

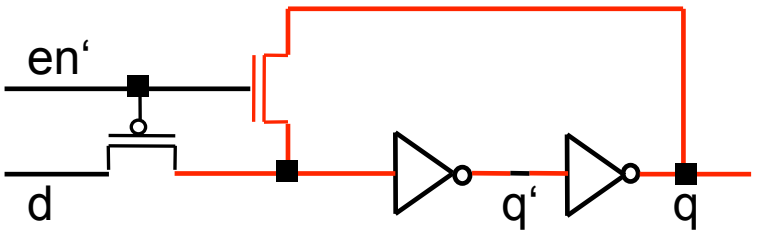
Pegelgesteuertes D-Speicherelement

Bei einem D-Speicherelement wird am Eingang d (**data**) direkt der zu speichernde Wert angelegt, statt diesen über Set und Reset einzustellen.

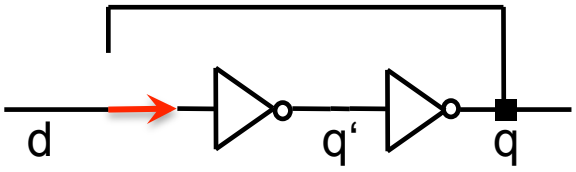
Negativ (active low) taktpegelgesteuertes (auch: taktzustandsgesteuertes) D-Speicherelement:

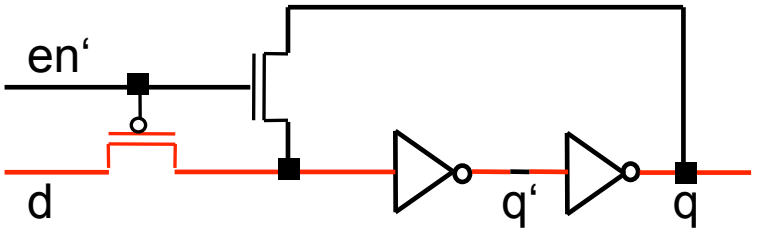
en‘ = 1 => q := q (gespeicherter Wert bleibt unverändert)





en‘ = 0 => q := d (Eingangsdaten werden in den Speicher übernommen)



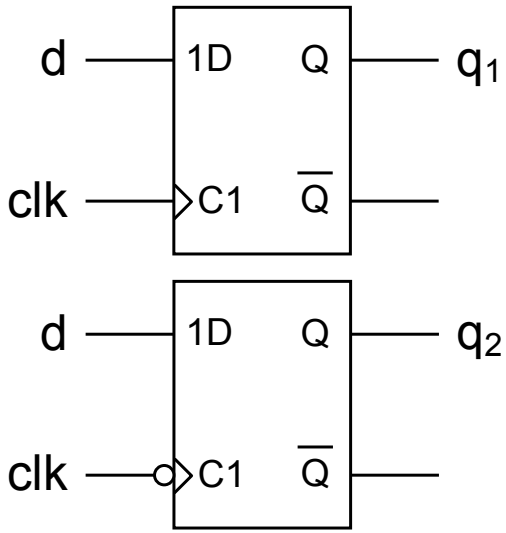


Klassifizierung und Begriffsübersicht

Bistabile Kippschaltung („Flipflop“ im Allgemeinen)

- Asynchron
- Taktpegelgesteuert (taktzustandsgesteuert, *Latch*)
  - positiv taktpegelgesteuert (*active high*)
  - negativ taktpegelgesteuert (*active low*)
- Taktflankengesteuert („*Flipflop*“ im Speziellen)
  - positiv taktflankengesteuert (*rising edge*)

    clk : Taktsignal (clock), bislang als en‘ (enable) bezeichnet
  - negativ taktflankengesteuert (*falling edge*)



Kontrollfragen

- In welchem Fall befindet sich ein Knoten eines elektrischen Netzwerks im Schwebezustand („floating“)? In welcher Hinsicht ist das ein Problem für digitale Logikschaltungen?
- Welche Möglichkeiten gibt es, um Schwebezustände auszuschließen?
- Beschreiben Sie die Gemeinsamkeiten und Unterschiede der Arbeitspunkt-bestimmung für Bipolar-, NMOS- und CMOS-Logikgatter.
- Wie beeinflusst die Hintereinanderschaltung von Logikgattern das elektrische Verhalten der Gatter, und welche Anforderungen ergeben sich daraus für den Entwurf der Gatter?
- Erstellen Sie eine Tabelle, in der Sie die Mechanismen der Entstehung von Verlustleistung für Bipolar, NMOS und CMOS Gatter gegenüberstellen.
- Konstruieren Sie ein hypothetisches „AND“-Logikgatter als elementares Gatter (d.h. keine Kombination von NAND und Inverter) in CMOS-Technologie. Was ist das Problem mit diesem Gatter?
- Was bedeutet „komplementär“ bezogen auf den Aufbau von pull-up und pull-down Netzwerken und warum ist diese Eigenschaft erforderlich?
- Wie unterscheiden sich CMOS-Komplexgatter von „normalen“ CMOS-Logikgattern und von Gatternetzlisten?

Kontrollfragen

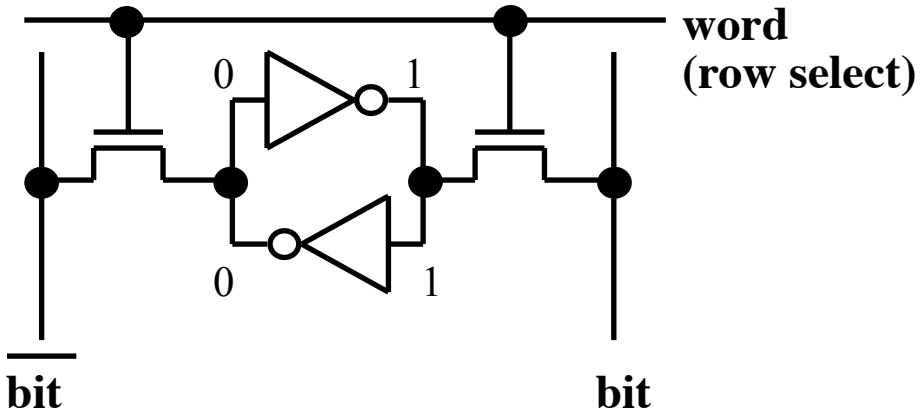
- Wenn sich der logische Wert am Eingang eines CMOS-Gatters ändert, zeigt sich ein ggf. veränderter Ausgangswert erst mit einiger Verzögerung. Welcher Mechanismus ist für den Großteil dieser Verzögerung verantwortlich?
- Worin unterscheiden sich asynchrone, taktzustands- und taktflankengesteuerte RS-Speicherelemente bezüglich ihrer Funktionalität?
- Wie spiegeln sich diese Unterschiede im Aufbau (Schaltungstopologie) der Speicherelemente wider?
- Bringen Sie Funktionalität (Verhalten) und Aufbau (Struktur) in Zusammenhang mit dem Y-Diagramm. Auf welcher Entwurfsebene befinden wir uns jeweils?
- Welche Unterschiede und ggf. Gemeinsamkeiten gibt es zwischen SRAM- und DRAM-Zellen auf der einen Seite und D-Flipflops auf der anderen?
- Vergleichen Sie den Aufbau der SRAM-Speichermatrix mit dem der DRAM-Speichermatrix.
- Wie unterscheiden sich die externen Schnittstellen (Interfaces) von SRAM und DRAM, und welche Auswirkungen hat dies auf die Zugriffsprotokolle?
- Welche Unterschiede gibt es bei einem Lesezugriff zwischen dem SRAM-Protokoll und dem DRAM-Protokoll?
- Identifizieren Sie mit dem Wissen um den Aufbau und das elektrische Verhalten von CMOS-Gattern Konflikte zwischen den Optimierungszielen Performanz (= geringe Verzögerung, hohe Taktfrequenz) und geringe Verlustleistung.

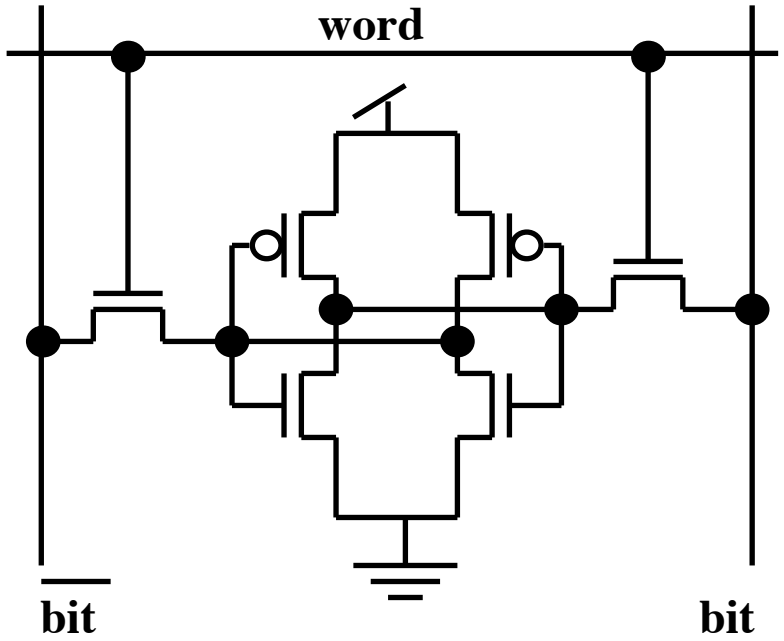
ANHANG

Grundprinzipien RAM-Speicher

- Ziel: Speicherung von ***mehreren*** Werten
- Random Access Memory (RAM): Speicher mit *wahlfreiem Zugriff*
  - Werte können in beliebiger Reihenfolge gelesen und gespeichert werden
  - im Unterschied z.B. zu einem Bandspeicher mit sequentiellem Zugriff
  - Auswahl des zu lesenden / schreibenden Werts über eine Adresse
- Möglichst einfache elementare Speicherzelle für einzelnen Wert (geringer Flächenbedarf => hohe Speicherkapazität pro Chipfläche)
- Reguläre Anordnung vieler Speicherzellen in einer Matrixstruktur mit
  - Zeilen (rows)
  - Spalten (columns)
- Interface und Protokoll für externen Zugriff (Lesen, Schreiben) auf beliebige Speicherzellen

SRAM-Speicherzelle („6T-Zelle“)

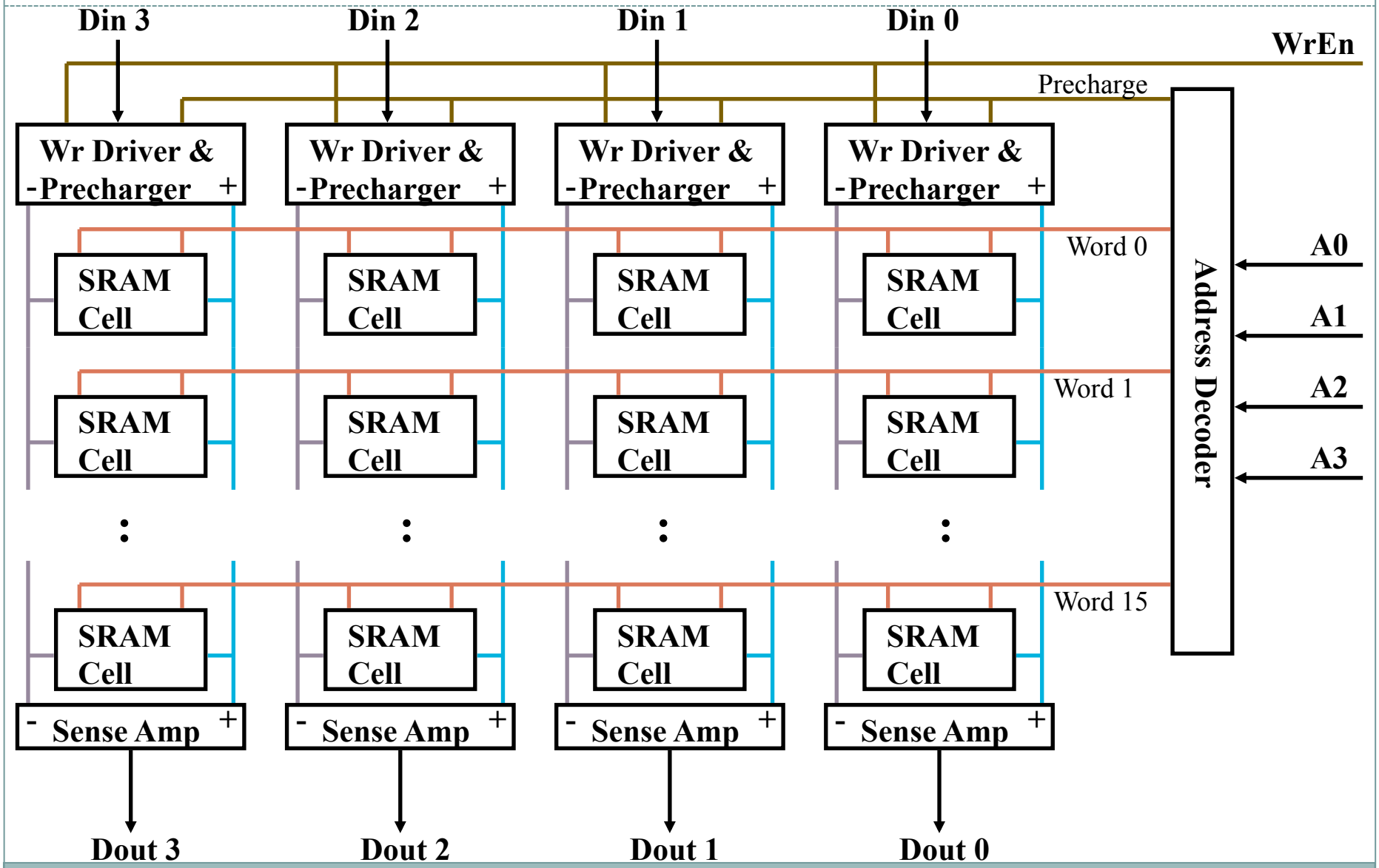




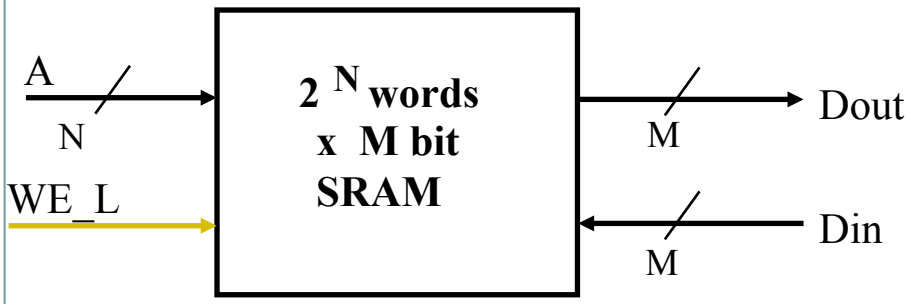
- Schreiben:
  1. Treibe (*drive*) bit lines (z.B. bit=1, $\overline{\text{bit}}$=0)
  2. Wähle row an (word=1)
- Lesen:
  1. Lade bit und $\overline{\text{bit}}$ auf (*precharge*)
  2. Wähle row an (word=1)
  3. Zelle entlädt eine der bit lines
  4. Leseverstärker (*sense amp*) detektiert den Unterschied zwischen bit und $\overline{\text{bit}}$

SRAM = *Static* RAM:
behält gespeicherte Werte dauerhaft (solange Versorgungsspannung ”ein”).

SRAM-Matrixaufbau (hier: 16 Worte á 4 Bit)

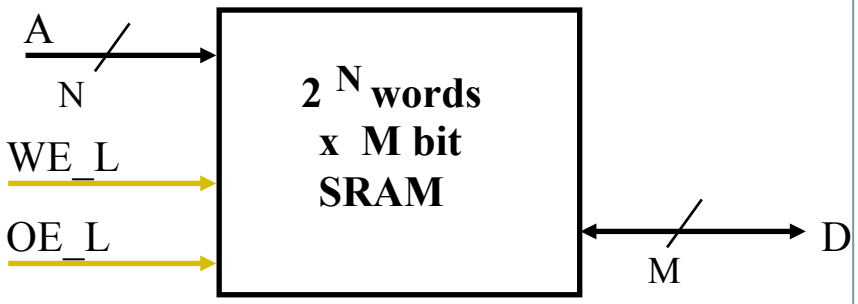


Typische SRAM-Interfaces



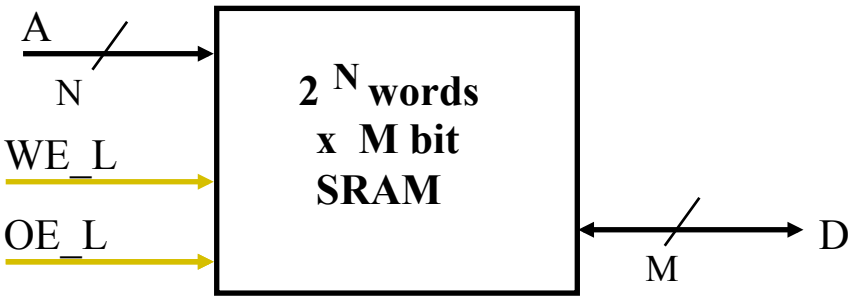
- WE_L: write enable, active low
- A: Adresse (zum Schreiben und Lesen)
- Din: Daten-Eingang (zum Schreiben)
- Dout: Daten-Ausgang (zum Lesen), hat immer den Wert von Wort A

- SRAM (besonders on-chip) kann taktgesteuert sein => zusätzlicher Eingang clk



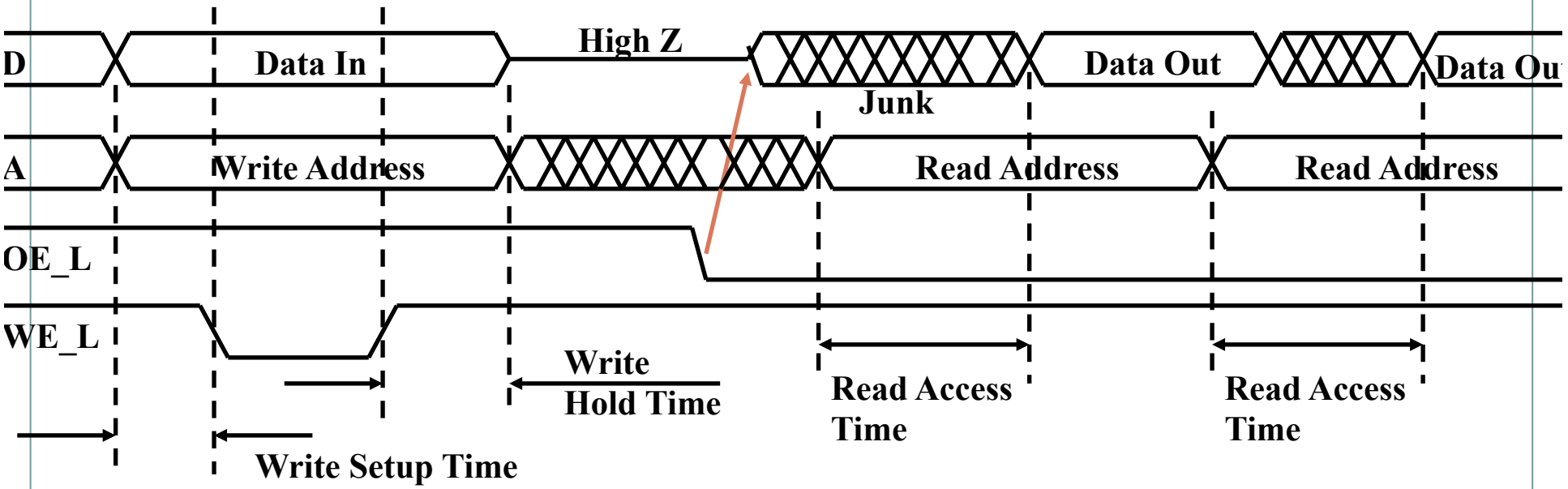
- WE_L: write enable, active low
- A: Adresse (Schreiben & Lesen)
- D: Datenein- und –ausgang, bidirektional (=> weniger Anschlüsse)
- OE_L: output enable, active low
  - WE_L = 1 & OE_L = 1
    - weder Lesen noch Schreiben
  - WE_L = 0 & OE_L = 1
    - D ist Dateneingang
  - WE_L = 1 & OE_L = 0
    - D ist Datenausgang
  - WE_L = 0 & OE_L = 0
    - verboten

Typisches SRAM-Protokoll



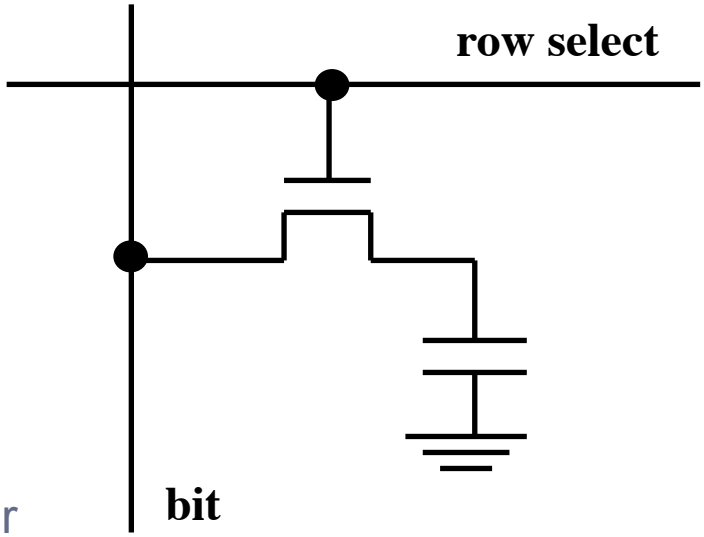
Write Timing:

Read Timing:



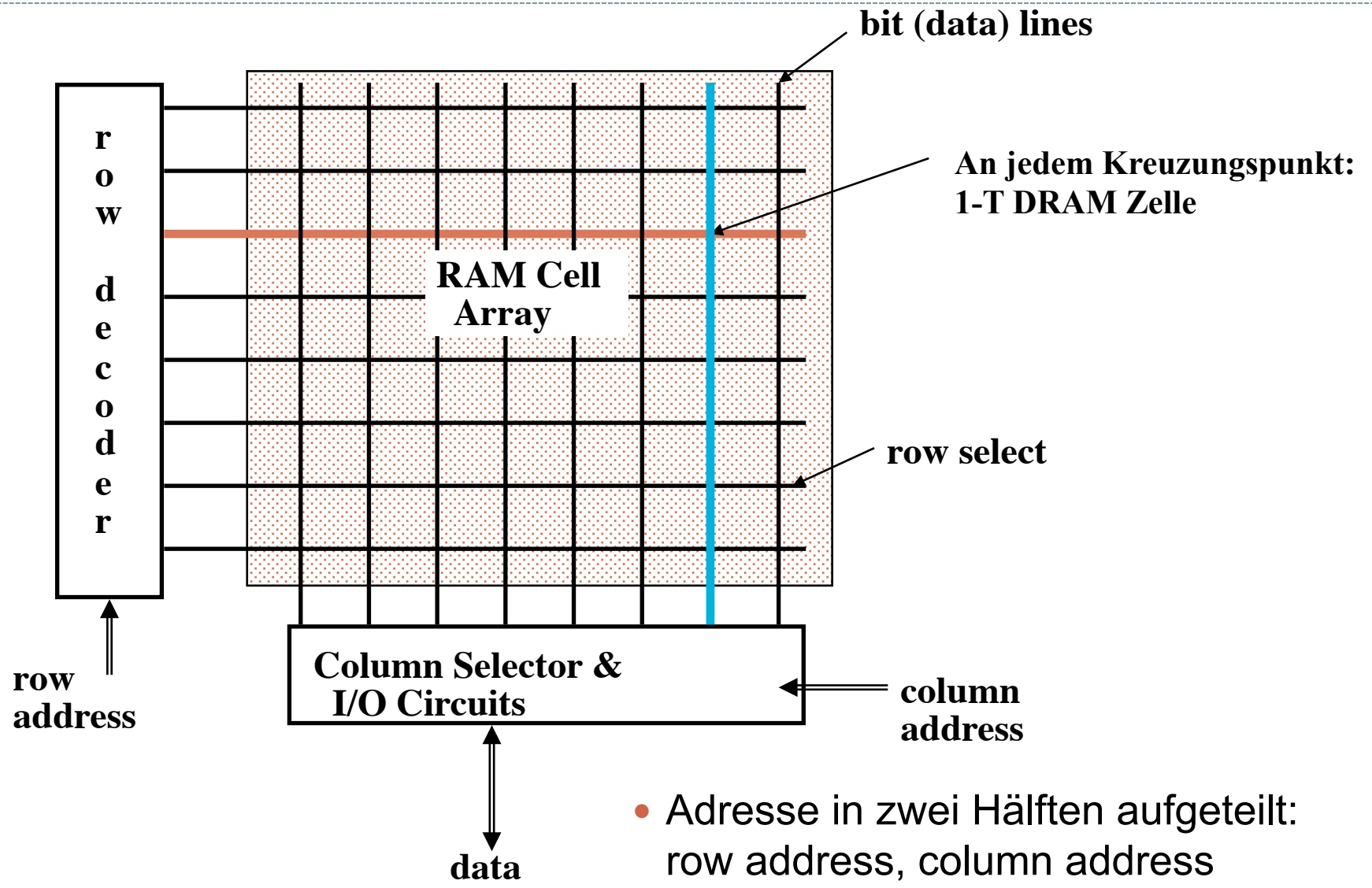
DRAM-Speicherzelle („1T-Zelle“)

- Schreiben:
  - 1. Treibe (*drive*) bit line
  - 2. Wähle row an (row select = 1)
- Lesen:
  - 1. Lade bit line auf (*precharge*)
  - 2. Wähle row an (row select = 1)
  - 3. Ladungsausgleich durch leitenden Transistor
    - geringfügige Spannungsänderung auf bit line
  - 4. hochempfindlicher Leseverstärker (sense amp)
    - erkennt Ladungsänderung von ~1 Million Elektronen
  - 5. Schreiben: gelesenen Wert wiederherstellen
- Refresh:
  - 1. Jede Zelle muss regelmäßig ausgelesen werden, da die Ladung auch bei gesperrtem Transistor langsam verloren geht.



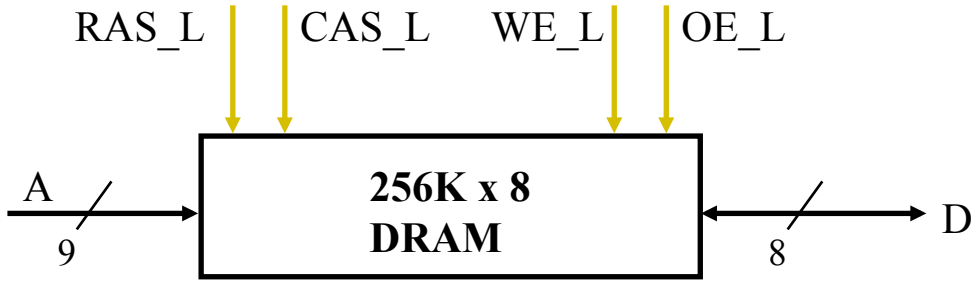
DRAM = *Dynamic* RAM: Speicherinhalte gehen über die Zeit verloren, müssen aufgefrischt werden.

DRAM-Matrixaufbau



- Adresse in zwei Hälften aufgeteilt: row address, column address

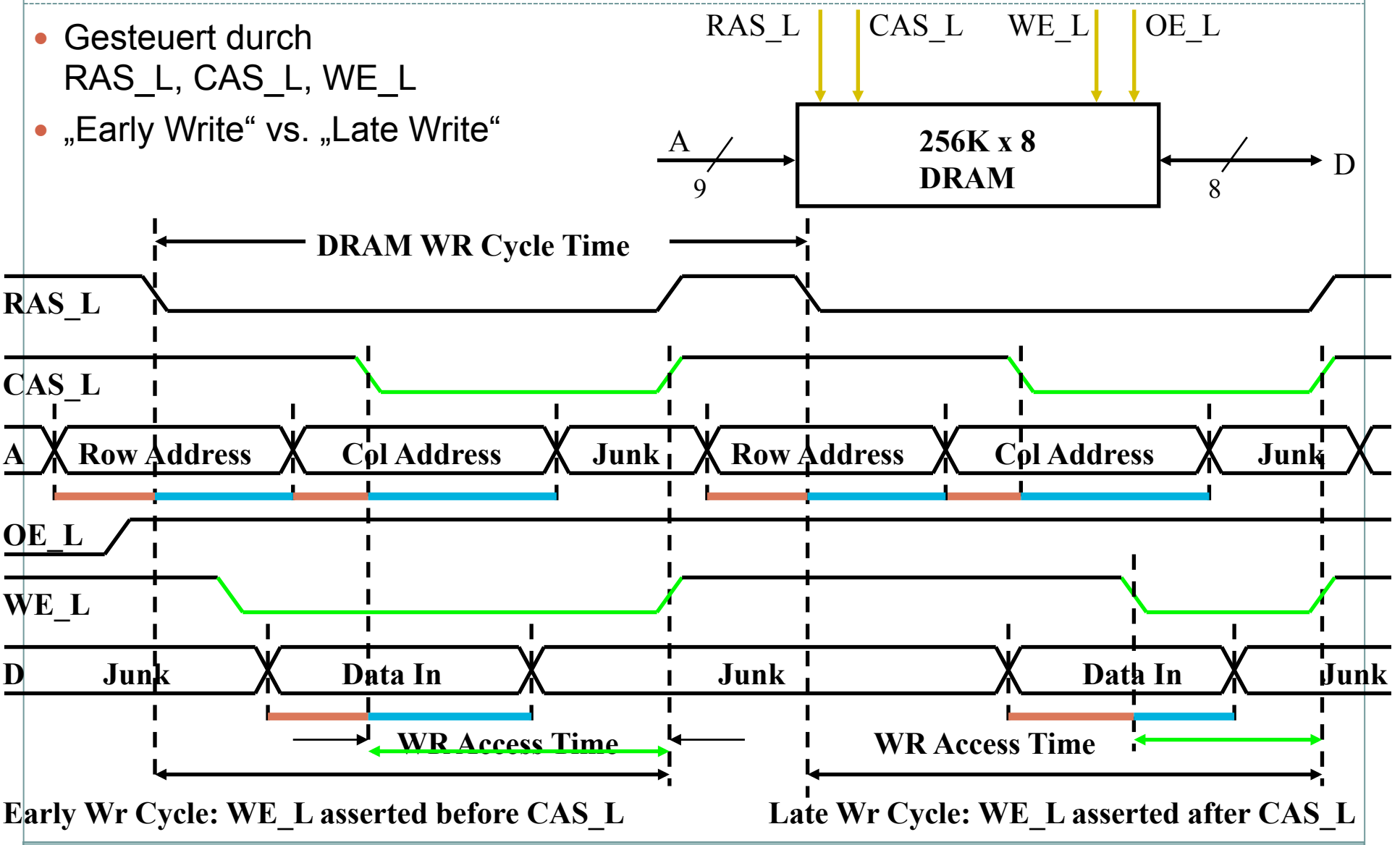
Typisches DRAM-Interface



- Steuersignale (RAS_L, CAS_L, WE_L, OE_L) active low
- Din und Dout sind kombiniert (D):
  - WE_L = 0 & OE_L = 1
    - D dient als Dateneingang
  - WE_L = 1 & OE_L = 0
    - D dient als Datenausgang
- Row address und column address nacheinander über den gleichen Eingang (A)
  - RAS_L (Row Address Strobe) → 0: row address wird über A eingelesen
  - CAS_L (Column Address Strobe) → 0: column address wird über A eingelesen
  - RAS/CAS sind überlicherweise flankengesteuert

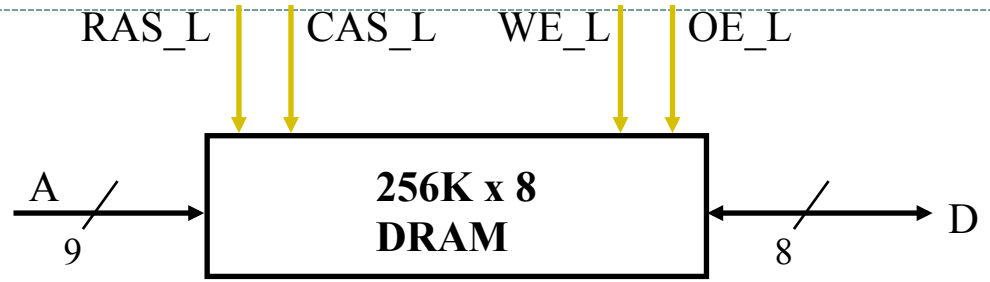
Typisches DRAM-Protokoll: Schreiben

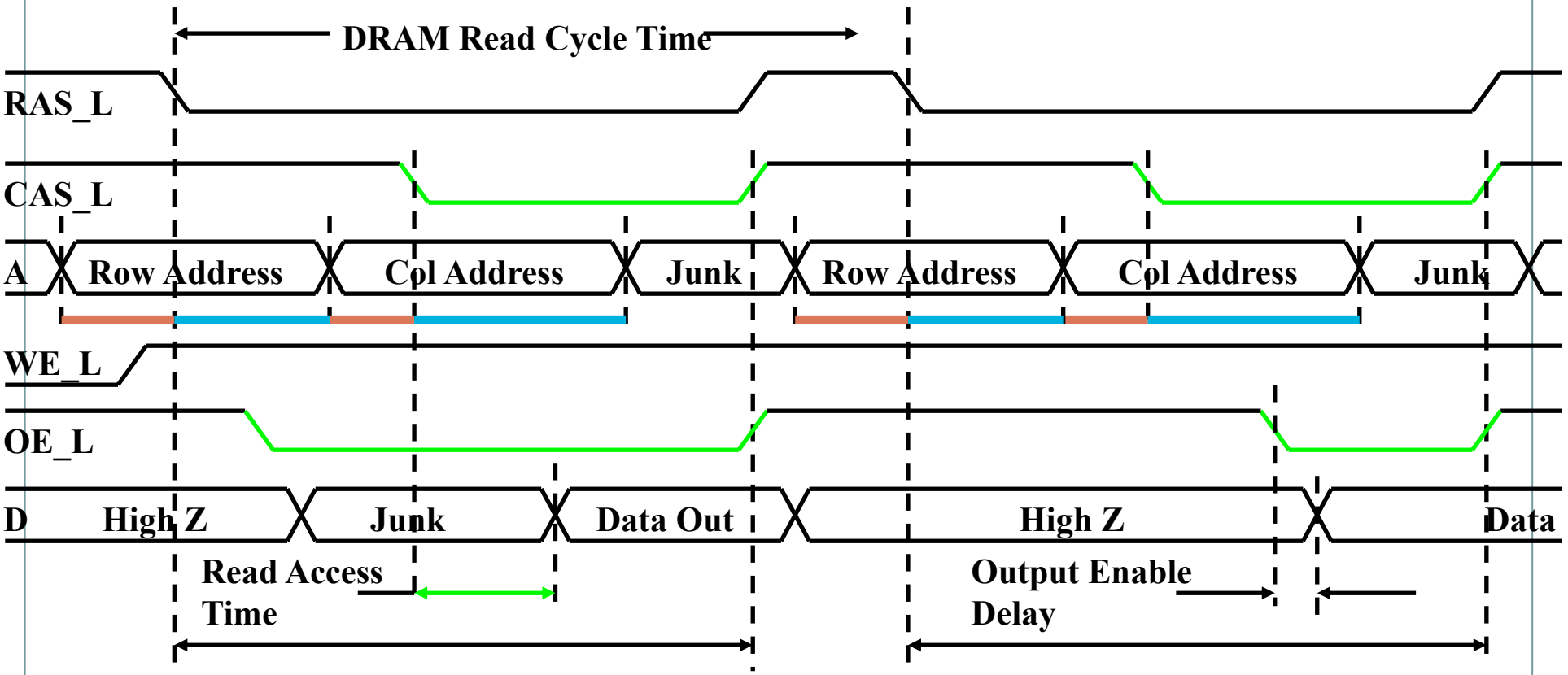
- Gesteuert durch RAS_L, CAS_L, WE_L
- „Early Write“ vs. „Late Write“



Typisches DRAM-Protokoll: Lesen

- Gesteuert durch RAS_L, CAS_L, OE_L
- „Early Read“ vs. „Late Read“





Early Read Cycle: OE_L asserted before CAS_L

Late Read Cycle: OE_L asserted after CAS_L

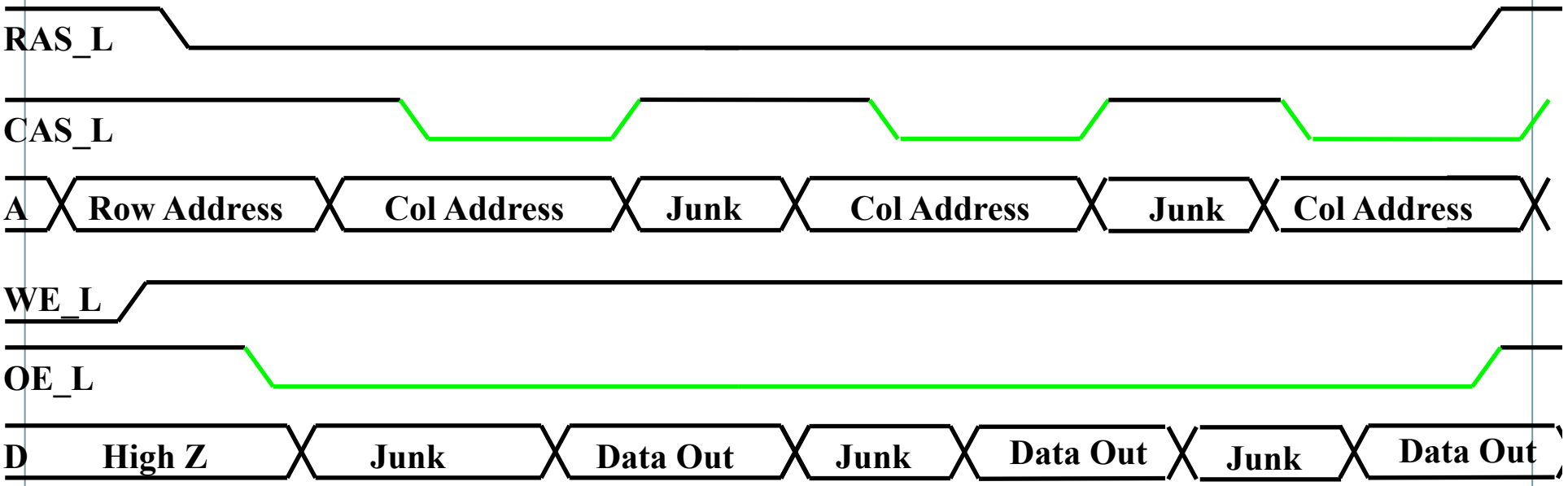
Wichtige Zeitparameter von DRAMs

- t_{RAC}: read access cycle time
  - Minimale Zeitspanne zwischen RAS-Flanke und gültigem Datenausgang
  - „Latenz“ eines DRAM
  - Typischer Wert: t_{RAC} = 60 ns
- t_{RC}: read cycle time
  - Minimale Zeitspanne zwischen Beginn eines Lesevorgangs bis zum nächsten (d.h. zwischen zwei RAS-Flanken)
  - Bestimmt den Datendurchsatz eines DRAM
  - Typischer Wert: t_{RC} = 110 ns (4Mbit DRAM mit t_{RAC} = 60 ns)
- t_{CAC}: column access cycle time
  - Minimale Zeitspanne zwischen CAS-Flanke und gültigem Datenausgang
  - Typischer Wert: t_{CAC} = 15 ns (4Mbit DRAM mit t_{RAC} = 60 ns)
- t_{PC}: page cycle time
  - Minimale Zeitspanne zwischen zwei CAS-Flanken
  - 35 ns for a 4Mbit DRAM with a t_{RAC} of 60 ns

=> Schnellerer Datenzugriff bei konstanter row address möglich

Fast Page Mode

- Schneller Zugriff auf Speicheradresse in der gleichen row (page)
- Spaltenadressen werden ohne Wiederholung der row address geändert



- … viele weitere DRAM-Varianten mit diversen Optimierungen: EDO-RAM, SDRAM, Rambus-DRAM, DDR-RAM, Multi-Port RAM, NVRAM, …