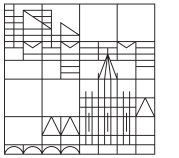
Universität Konstanz



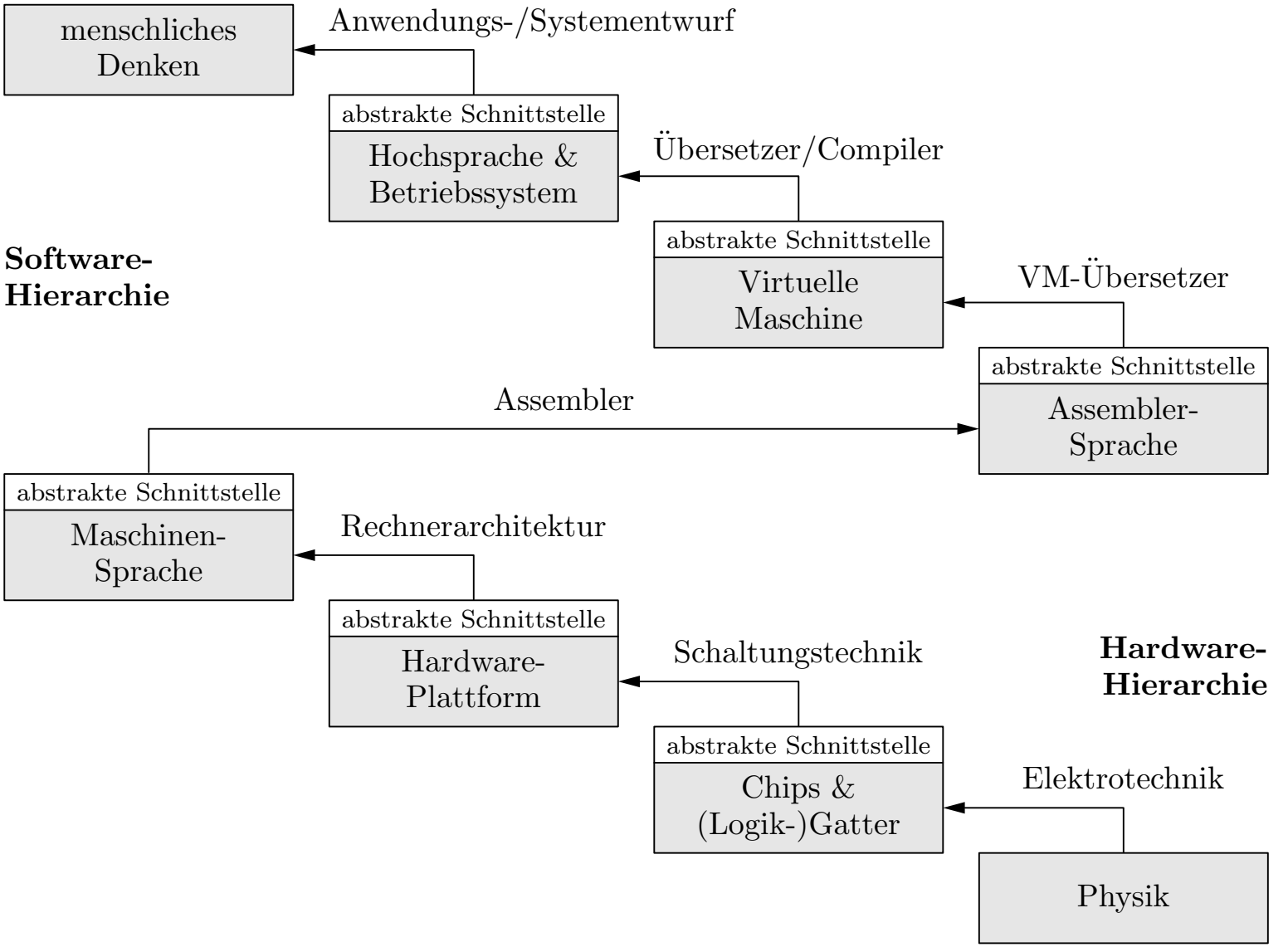
# Rechnersysteme und -netze
# Kapitel 4
# Sequentielle Logik

**Bastian Goldlücke**

Universität Konstanz

WS 2020/21

# Rechnersysteme: Plan der Vorlesung



# Erinnerung: Schaltungstechnik I

- **Boolesche Algebra / Schaltalgebra**
  - Definitionen, Operatoren, Schreibweisen etc.
  - Wahrheitstafeln und Boolesche Funktionen
  - Disjunktive und konjunktive Normalform
  - Vollständige Operationenmengen
- **Gatterlogik**
  - Elementare (Logik-)Gatter, Schaltzeichen für Gatter
  - Zusammengesetzte Gatter: Schnittstelle und Implementierung
  - Bitmustererkennung, Datenflußsteuerung, Multiplexer und Dekodierer
- **Implementierung durch Schaltkreise**
  - Elektrotechnische Grundlagen, Schalter, Transistoren etc.
  - Integrierte Schaltungen und ihre Herstellung

# Erinnerung: Schaltungstechnik II

- **Minimierung Boolescher Formeln**
  - Transformationen mit Hilfe von Äquivalenzen
  - Das Minimierungsverfahren von Karnaugh–Veitch (Karnaugh–Veitch-Diagramme)
  - Das Minimierungsverfahren von Quine–McCluskey (Methode der Primimplikanten, Petricks Algorithmus)
  - Programmierbare Logikarrays (programmable logic arrays, PLAs) (Implementierung von Disjunktionen von Konjunktionen von Literalen)

- **Hardware-Beschreibungssprache** (hardware description language, HDL)
  - Beschreibung der Zusammenschaltung von Gattern (Schnittstelle und Implementierung)
  - Simulationsumgebung für Gatterschaltungen (Schaltungstest auf Grundlage einer Spezifikation)

# Erinnerung: Binäre Arithmetik

- **Arithmetik** (im wesentlichen Erinnerung an Bekanntes)
  - Zahlensysteme (Basis 10, andere Basen als 10)
  - Addition und Subtraktion, Übertrag
  - Multiplikation und Division, Stellenprodukte
  - Implementierung durch mechanische Rechner

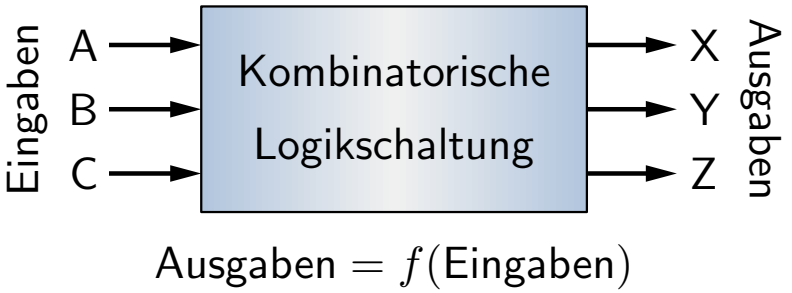
- **Arithmetisch-logische Einheit** (Arithmetic Logic Unit, ALU)
  - Binärarithmetik (Rechnen im Zahlensystem mit Basis 2)
  - Halbaddierer und Volladdierer (1-Bit-Addierer)
  - *n*-Bit-Addierer (mit Übertragskette und Übertragsauswahl)
  - Multiplikation: Bit-Schieben, Standardalgorithmus
  - Multiplikation: negative Zahlen, Booths Algorithmus
  - Hack-Architektur (arithmetisch-logische Einheit, Prozessor)

# Inhalt

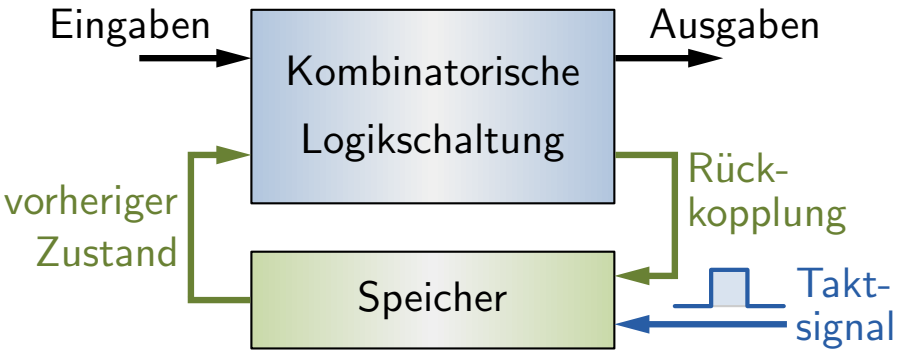
# Inhalt

# Rechnersysteme: Logikschaltungen

Bisher: **Kombinatorische Logik**




Jetzt: **Sequentielle Logik**




Implementiert durch

**Schaltnetze:**

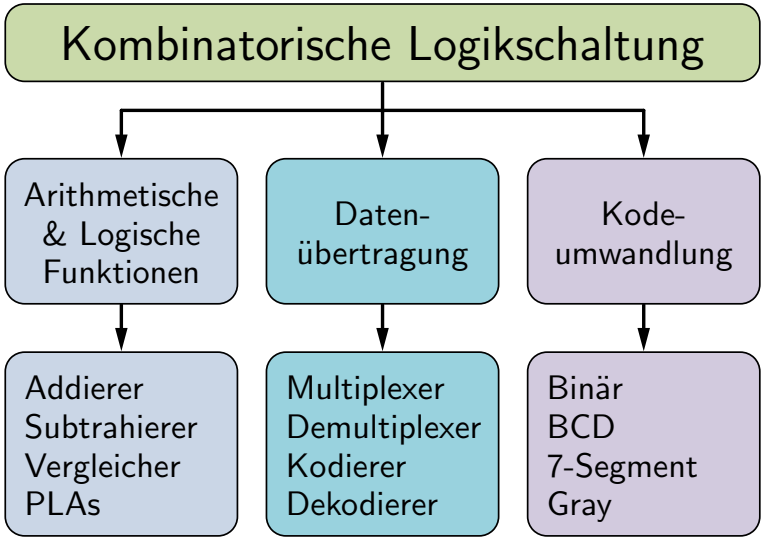
- Einfaches Berechnen von Ausgaben aus Eingaben
- Kein Speichern von Informationen
- Zustandslosigkeit
- Meist idealisierte Annahme: verzögerungsfreie Berechnung

**Schaltwerke:**

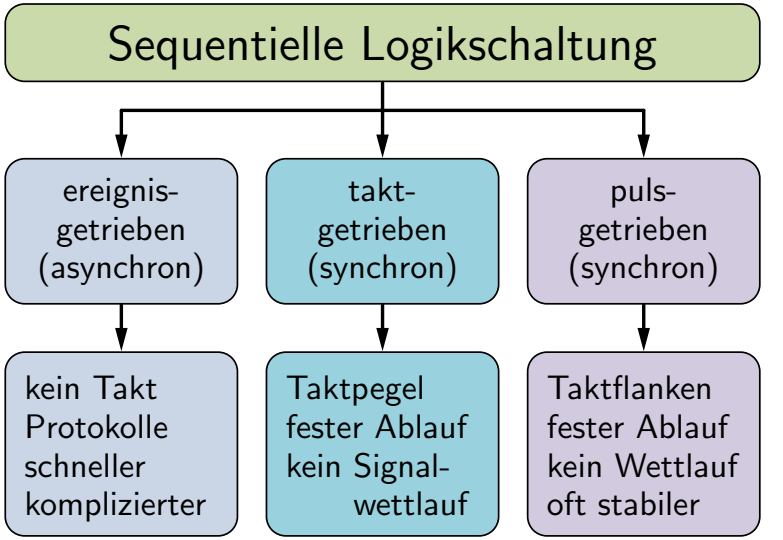
- Rückkopplung von Ausgaben auf Eingaben
- Explizites Speichern von Informationen
- Unterscheidung von Zuständen
- Gatterlaufzeiten werden explizit berücksichtigt, ggf. Synchronisierung

# Rechnersysteme: Logikschaltungen

Bisher: **Kombinatorische Logik**
**Schaltnetze**




Jetzt: **Sequentielle Logik**
**Schaltwerke**

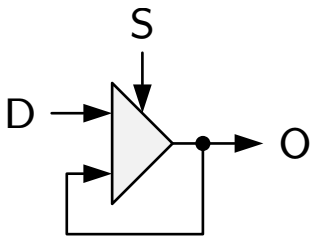



Hauptfragen der sequentiellen Logik:

- Wie kann man Informationen speichern?
  $\Rightarrow$ **Rückkopplungen**
- Wie kann man Abläufe und Berechnungen verläßlich synchronisieren?
  $\Rightarrow$ Gatterlaufzeitbetrachtungen, **Taktsignal**

# Prinzip der Rückkopplung

**Einfaches Beispiel:**



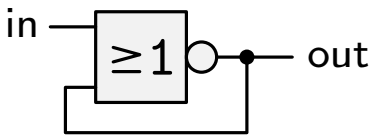

- Rückgekoppelter 2-Wege-Multiplexer
- Wenn S $= 0$, wird die Eingangsleitung D auf den Ausgang O durchgeschaltet.
- Wenn S $= 1$, wird der Ausgang O festgehalten, unabhängig von der Eingangsleitung D.

- Durch eine Rückkopplung werden Ausgaben wieder zu Eingaben.
- Dadurch kann z.B. ein (Ausgabe-)Zustand festgehalten werden, bis ein Ereignis (anderes Steuersignal/andere Eingabe) ihn wieder ändert.
- Alle **Schaltnetze** (kombinatorische Logik) sind gerichtete azyklische Graphen von (Logik-)Gattern (keine Rückkopplungen, keine gerichteten Kreise).
- Alle **Schaltwerke** (sequentielle Logik) sind gerichtete zyklische Graphen von (Logik-)Gattern (Rückkopplungen, gerichtete Kreise).

Richtungen der Kanten der Graphen: Daten-/Signalübertragungsrichtung.

# Probleme der Rückkopplung

**Einfaches Beispiel:**




- Rückgekoppeltes NOR-Gatter ($a \downarrow b$)
  Annahme: Gatterlaufzeit $\Delta t$
- Wenn in $= 1$, wird der Ausgang out $= 0$.
- Wenn in $= 0$, wechselt der Ausgang out seinen Wert nach jeweils $\Delta t$ immer wieder.

- Durch Rückkopplungen kann es zu (unerwünschten) **Schwingungen** kommen.
- Solche Schwingungen sind ein Beispiel für **Signallaufzeitprobleme**, die durch Rückkopplungen auftreten können.
- Ein weiteres Beispiel sind sogenannte **Wettlaufsituationen** (race conditions), die auftreten, wenn sich Signale auf zwei oder mehr Wegen ausbreiten, und die Ausgabe davon abhängen kann, welches Signal schneller weitergegeben wird.
- Rückkopplungsprobleme lassen sich am leichtesten durch ein (zentral erzeugtes) **Taktsignal** vermeiden, das bestimmt, wann Eingaben übernommen werden.

# Asynchrone und synchrone Schaltwerke

**Asynchrone Schaltwerke** (seltener verwendet)

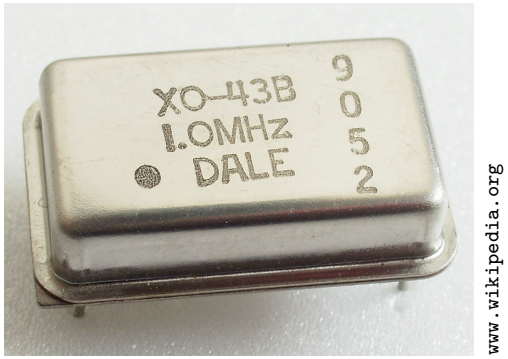
- Direkte Steuerung durch Änderung der Eingangssignale.
- Wann (und ggf. ob überhaupt) ein stabiler Zustand erreicht wird (und damit stabile Ausgangssignale), hängt von der Gatterlaufzeit ab.
- Oft komplizierter, aufwendiger Entwurf.
- Aber dadurch sehr schnelle Schaltwerke möglich.

**Synchrone Schaltwerke** (häufiger verwendet)

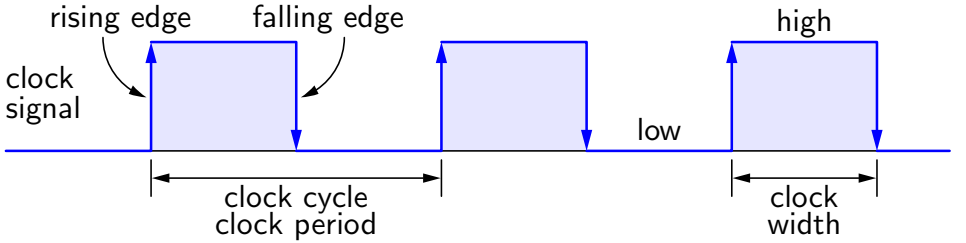
- Steuerung durch ein (zentral erzeugtes) Taktsignal.
- Eingangssignale werden nur zu bestimmten, durch das Taktsignal festgelegten Zeitpunkten übernommen.
- Meist einfacher, systematischer Entwurf.
- Aber: Langsamstes Bauteil bestimmt maximal mögliche Taktfrequenz.

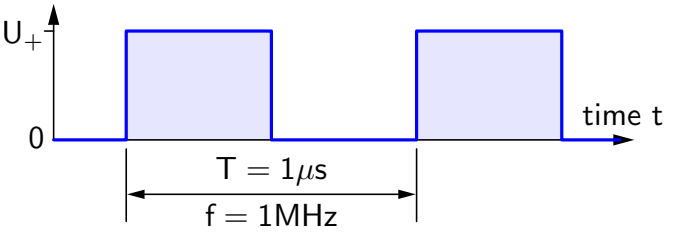
# Taktsignal (Clock)

- Ein **Taktsignal** (clock signal) wird meist durch einen **Quarzoszillator** erzeugt.
- Durch elektromagnetische Resonanz eines piezoelektrischen Quarzkristalls (Schwingquarz) entsteht ein Taktsignal mit einer festen Frequenz.



www.wikipedia.org
(Rainer Knäpper)

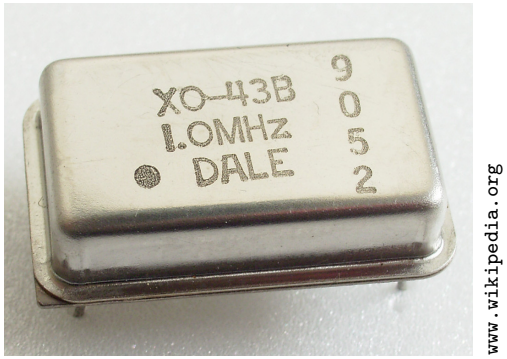


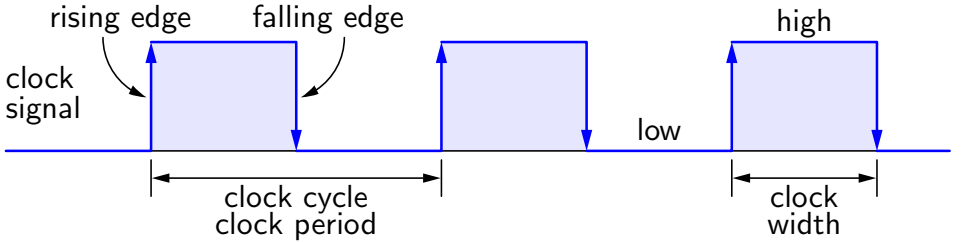

- Idealisierend beschreiben wir ein Taktsignal als Folge von Rechteckpulsen.
- Die Zeit $T$ zwischen einer steigenden Flanke (rising edge) und der nächsten (oder die Zeit zwischen einer fallenden Flanke (falling edge) und der nächsten), heißt **Taktzyklus** (clock cycle) oder **Taktperiode** (clock period).
- Die **Taktfrequenz** $f$ ist der Reziprokwert der Taktperiode $T$.

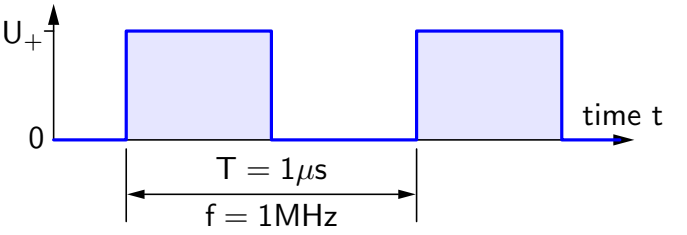
# Taktsignal (Clock)

- Ein **Taktsignal** (clock signal) wird meist durch einen **Quarzoszillator** erzeugt.
- Durch elektromagnetische Resonanz eines piezoelektrischen Quarzkristalls (Schwingquarz) entsteht ein Taktsignal mit einer festen Frequenz.



www.wikipedia.org
(Rainer Knäpper)



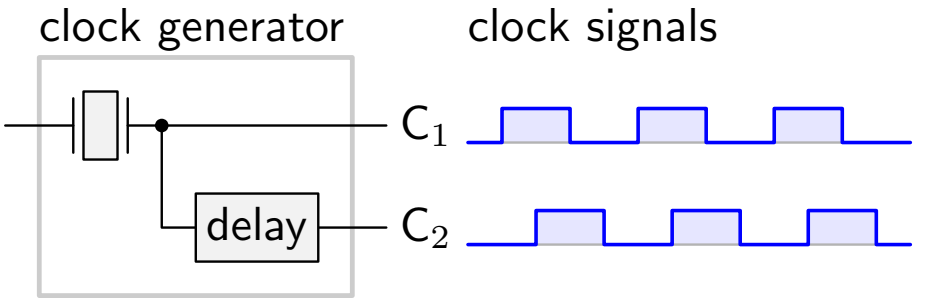



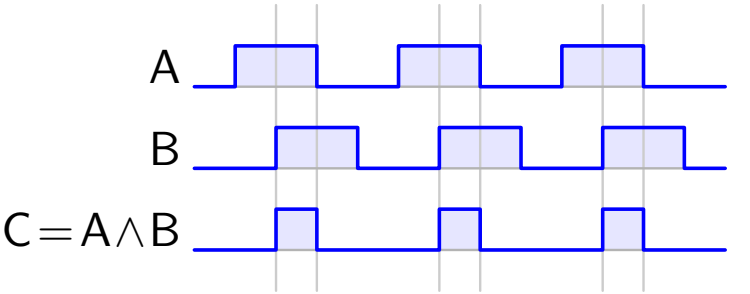

- Die Zeit zwischen einer steigenden und der darauffolgenden fallenden Flanke heißt **Taktbreite** (clock width). Sie muß nicht gleich der halben Taktperiode sein.
- Die Phasen eines Taktzyklus unterscheidet man nach ihrem **Taktpegel** oder **Taktzustand**: 1-Pegel (Versorgungsspannung, high) und 0-Pegel (Masse, low).
- Der Taktpegel und/oder die Taktflanken werden zur Ablaufsteuerung benutzt.

# Taktsignal (Clock)

- Oft ist die Reihenfolge von Berechnungen wichtig:
  - Manche Berechnungen müssen/können parallel abgearbeitet werden.
  - Andere Berechnungen müssen strikt sequentiell abgearbeitet werden.
- Das Taktsignal dient als „Zeitgeber“ für solche Berechnungen.
- Typische Taktfrequenzen liegen zwischen einigen KiloHertz (kHz) und einigen GigaHertz (GHz). (benannt nach Heinrich Hertz, 1857–1894)

  $\Rightarrow$ Taktzyklen im Millisekunden- (ms) bis Nanosekundenbereich (ns).
- Manchmal werden asymmetrische Taktsignale benötigt, die durch Verzögerung (delay) und logische Und-Verknüpfung mit dem Original erzeugt werden.

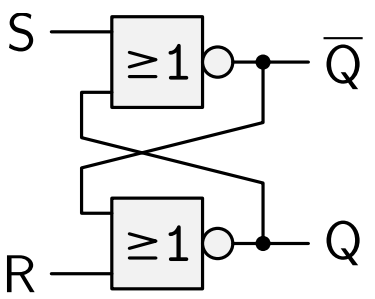


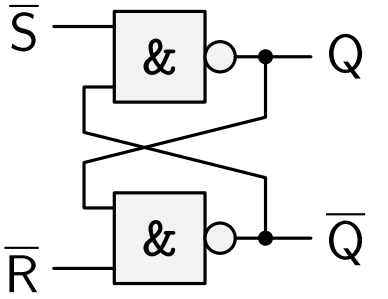



# Inhalt

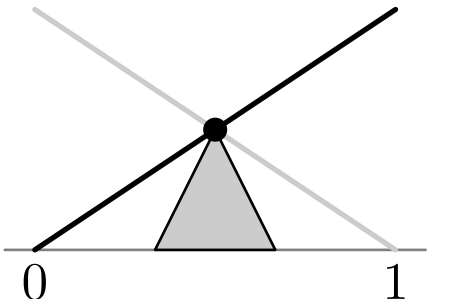
# Bistabile Kippstufen („Flipflops“)

- Eine **bistabile Kippstufe**, auch **bistabiles Kippglied** oder **Riegel** (latch), ist eine rückgekoppelte Schaltung aus zwei NOR- oder zwei NAND-Gattern mit zwei Eingängen und zwei Ausgängen (auch speziell **SR-Riegel** genannt).



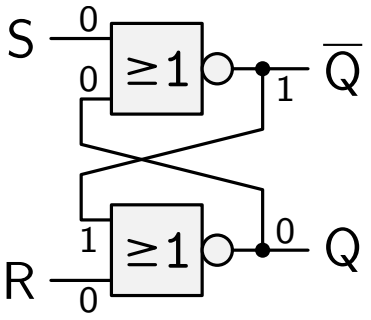



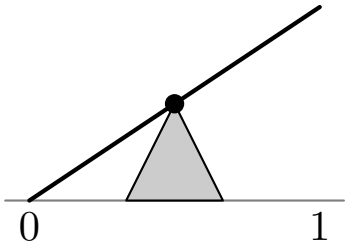

- Oft wird einfach, aber ungenau, von **Flipflop** gesprochen (später mehr dazu).
- Eine bistabile Kippstufe hat zwei stabile (daher die Bezeichnung bistabil) und einen metastabilen Zustand (mit inkonsistenten Ausgaben).
- Man kann sich eine bistabile Kippstufe wie eine Wippe vorstellen, die ja ebenfalls in einer von zwei stabilen Lagen sein kann: Balken ist links unten oder rechts unten.



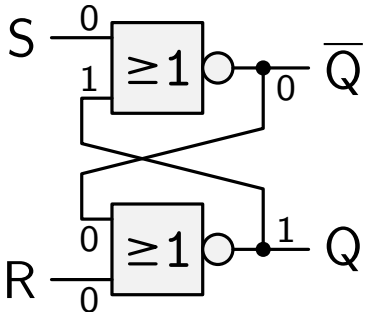
# Einfache Bistabile Kippstufen (SR-Riegel)

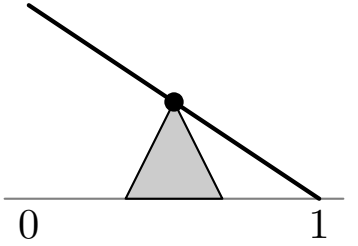
Wir betrachten zunächst die aus NOR-Gattern aufgebaute bistabile Kippstufe:




- Die Ausgabe $Q = 0$ (und folglich $\overline{Q} = 1$) ist bei Eingaben $S = R = 0$ stabil.
- Dies ist der erste von zwei stabilen Zuständen.



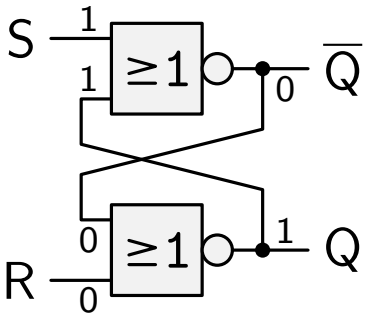

- Die Ausgabe $Q = 1$ (und folglich $\overline{Q} = 0$) ist bei Eingaben $S = R = 0$ stabil.
- Dies ist der zweite von zwei stabilen Zuständen.

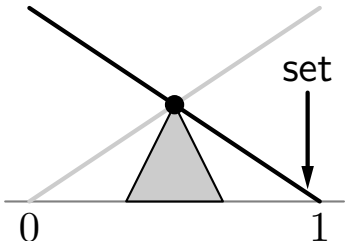
- Die Eingaben $S = R = 0$ können also als Speicherzustand interpretiert werden, in dem die Kippstufe ihren Zustand (entweder 0 oder 1) beibehält.
- Man beachte, daß die Ausgänge Q und $\overline{Q}$ stets komplementär zueinander sind.

# Einfache Bistabile Kippstufen (SR-Riegel)

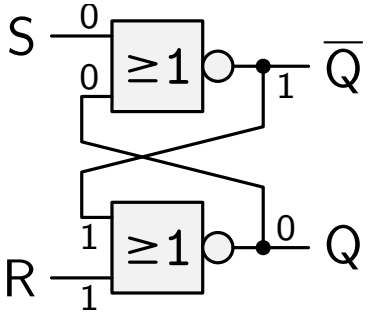
Wir betrachten zunächst die aus NOR-Gattern aufgebaute bistabile Kippstufe:

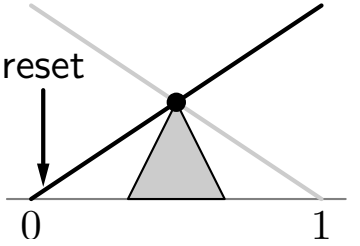
- Die Eingaben S $= 1$ und R $= 0$ bzw. S $= 0$ und R $= 1$ können als Setzen (set) bzw. Rücksetzen (reset) des Zustands interpretiert werden, durch das die Kippstufe definiert in den Zustand 1 bzw. 0 gebracht wird.




- Die Ausgabe Q $= 1$ (und folglich $\overline{Q} = 0$) wird durch S $= 1$ und R $= 0$ gesetzt.
- Dieser Zustand bleibt erhalten, wenn die Eingabe S von 1 auf 0 wechselt.



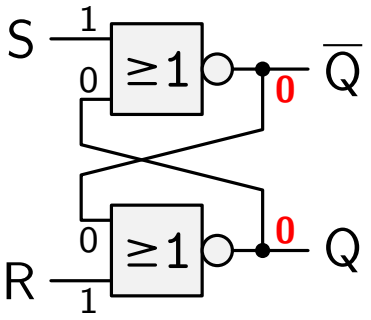



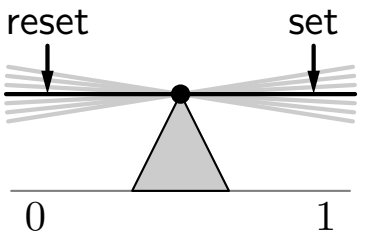

- Die Ausgabe Q $= 0$ (und folglich $\overline{Q} = 1$) wird durch S $= 0$ und R $= 1$ gesetzt.
- Dieser Zustand bleibt erhalten, wenn die Eingabe R von 1 auf 0 wechselt.

# Einfache Bistabile Kippstufen (SR-Riegel)

Wir betrachten zunächst die aus NOR-Gattern aufgebaute bistabile Kippstufe:

- Die Eingaben $S = R = 1$ führen zu einem inkonsistenten Zustand,
  da in diesem Fall $Q = \overline{Q} = 0$ gilt, also die Ausgänge nicht komplementär sind,
  außerdem der Übergang in den Speicherzustand $S = R = 0$ undefiniert ist.



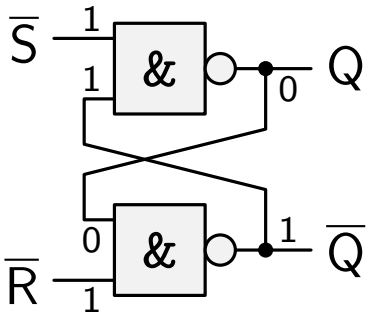

- Die Ausgaben Q und $\overline{Q}$ sind
  bei Eingaben $S = R = 1$ inkonsistent.
- Dieser Zustand wird metastabil genannt,
  da der Speicherzustand undefiniert wird.

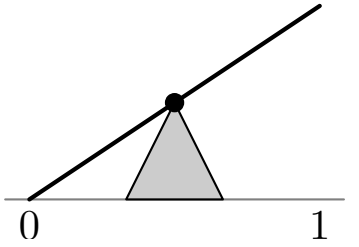
- Allgemein nennt man einen Zustand **metastabil**, wenn er stabil
  bei kleinen Änderungen, aber instabil bei großen Änderungen ist.
- Solange sich die Eingaben S und R (ungefähr) die Waage halten,
  bleibt die Kippstufe in diesem metastabilen Zustand $Q = \overline{Q} = 0$.

  Gewinnt eine Eingabe die Oberhand, geht sie in den zugehörigen Zustand.

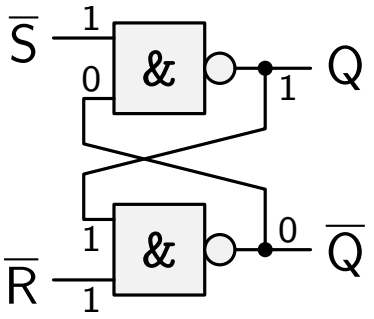
# Einfache Bistabile Kippstufen (SR-Riegel)

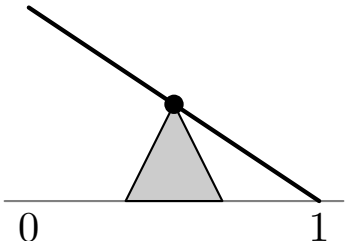
Wir betrachten nun die aus NAND-Gattern aufgebaute bistabile Kippstufe:




- Die Ausgabe $Q = 0$ (und folglich $\overline{Q} = 1$) ist bei Eingaben $\overline{S} = \overline{R} = 1$ stabil.
- Dies ist der erste von zwei stabilen Zuständen.



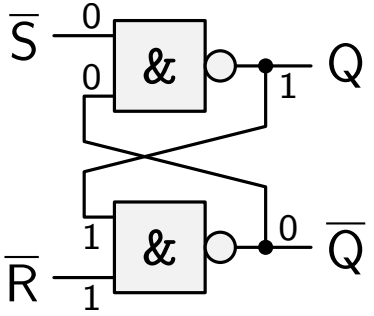

- Die Ausgabe $Q = 1$ (und folglich $\overline{Q} = 0$) ist bei Eingaben $\overline{S} = \overline{R} = 1$ stabil.
- Dies ist der zweite von zwei stabilen Zuständen.

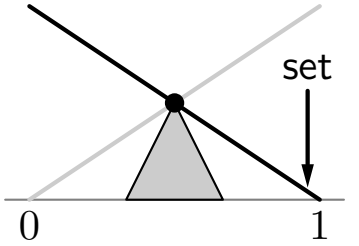
- Die Eingaben $\overline{S} = \overline{R} = 1$ können also als Speicherzustand interpretiert werden, in dem die Kippstufe ihren Zustand (entweder 0 oder 1) beibehält.
- Man beachte, daß die Eingänge gegenüber der NOR-Kippstufe negiert sind. Außerdem sind die Ausgänge Q und $\overline{Q}$ vertauscht.

# Einfache Bistabile Kippstufen (SR-Riegel)

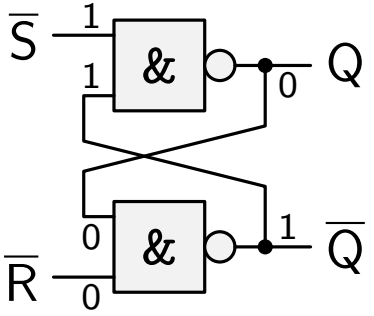
Wir betrachten nun die aus NAND-Gattern aufgebaute bistabile Kippstufe:

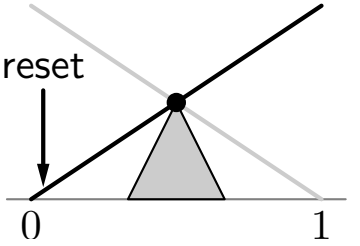
- Die Eingaben $\overline{S} = 0$ und $\overline{R} = 1$ bzw. $\overline{S} = 1$ und $\overline{R} = 0$ können als Setzen (set) bzw. Rücksetzen (reset) des Zustands interpretiert werden, durch das die Kippstufe definiert in den Zustand 1 bzw. 0 gebracht wird.




- Die Ausgabe $Q = 1$ (und folglich $\overline{Q} = 0$) wird durch $\overline{S} = 0$ und $\overline{R} = 1$ gesetzt.
- Dieser Zustand bleibt erhalten, wenn die Eingabe $\overline{S}$ von 0 auf 1 wechselt.



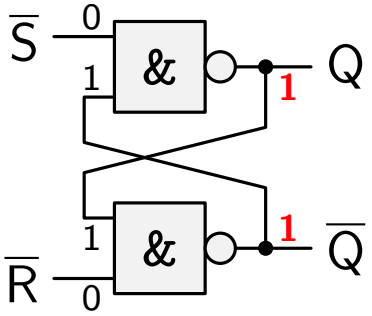



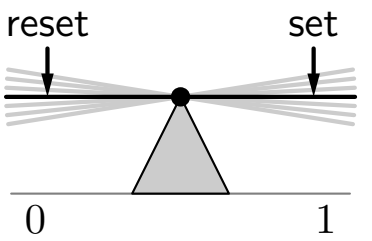

- Die Ausgabe $Q = 0$ (und folglich $\overline{Q} = 1$) wird durch $\overline{S} = 1$ und $\overline{R} = 0$ gesetzt.
- Dieser Zustand bleibt erhalten, wenn die Eingabe $\overline{R}$ von 0 auf 1 wechselt.

# Einfache Bistabile Kippstufen (SR-Riegel)

Wir betrachten nun die aus NAND-Gattern aufgebaute bistabile Kippstufe:

- Die Eingaben $\overline{S} = \overline{R} = 0$ führen zu einem inkonsistenten Zustand, da in diesem Fall $Q = \overline{Q} = 1$ gilt, also die Ausgänge nicht komplementär sind, außerdem der Übergang in den Speicherzustand $\overline{S} = \overline{R} = 1$ undefiniert ist.

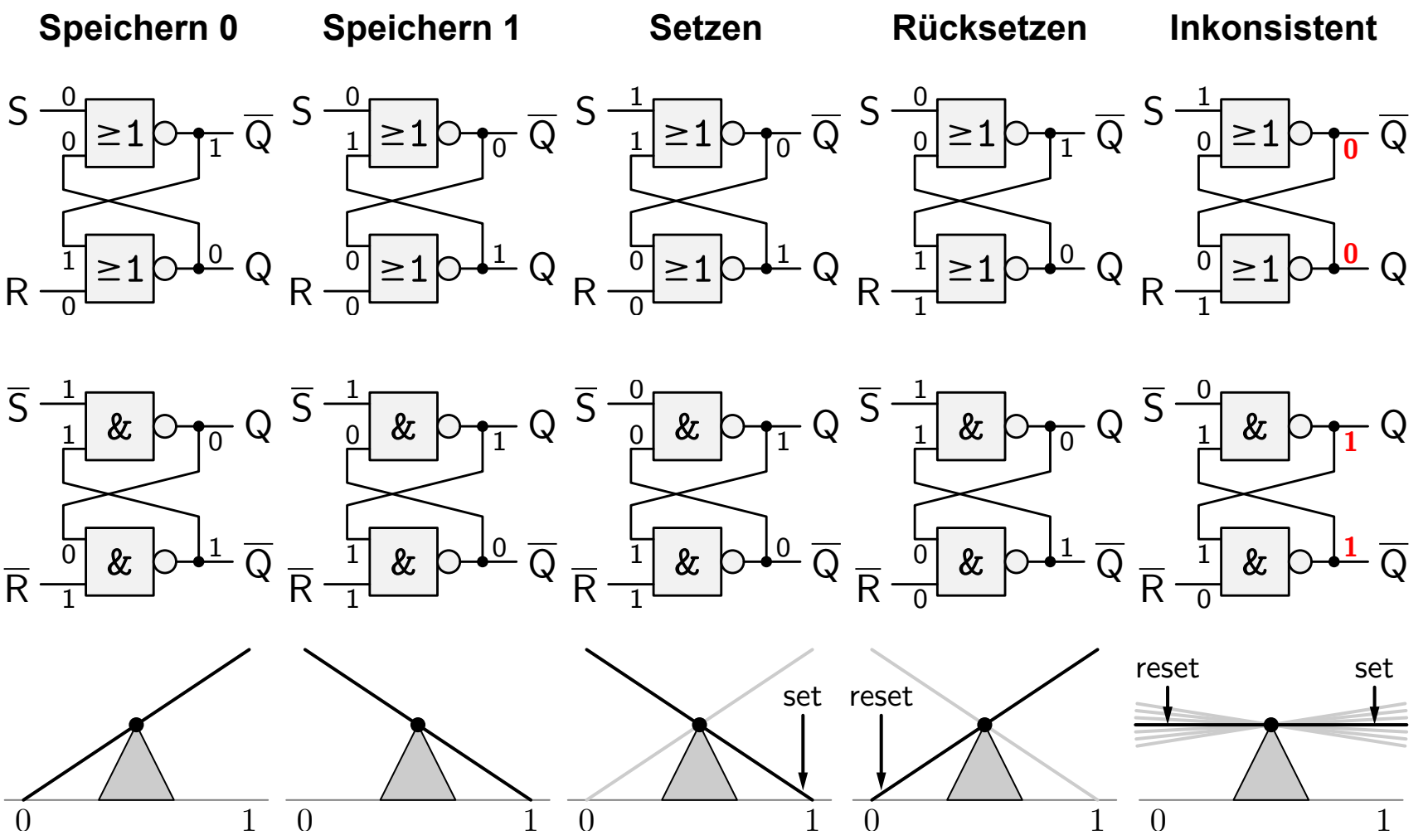



- Die Ausgaben Q und $\overline{Q}$ sind bei Eingaben $\overline{S} = \overline{R} = 0$ inkonsistent.
- Dieser Zustand wird metastabil genannt, da der Speicherzustand undefiniert wird.

- Die Kippstufe bleibt in diesem metastabilen Zustand, bis eine Eingabe die Oberhand gewinnt und sie in den zugehörigen stabilen Zustand bringt.
- Die im folgenden betrachteten erweiterten bistabilen Kippstufen unterscheiden sich i.w. darin, wie sie mit dem möglichen metastabilen Zustand umgehen, d.h., wie sie diesen Zustand vermeiden bzw. durch andere Funktionalität ersetzen.

# Einfache Bistabile Kippstufen: Zusammenfassung

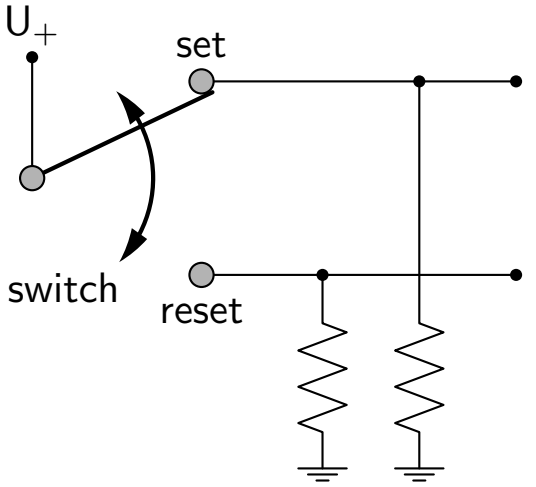


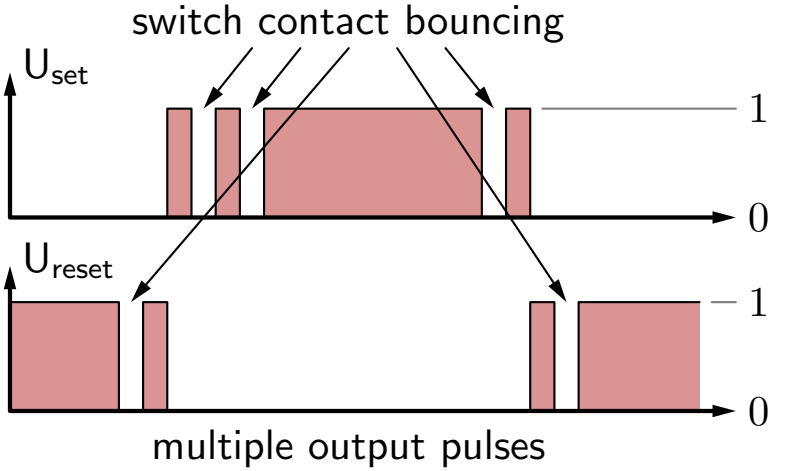

- Bistabile Kippstufen können durch ihre beiden stabilen Zustände ein Bit speichern, aber der metastabile Zustand ist problematisch.

# Einfache Bistabile Kippstufen: Beispielanwendung

**Beispielanwendung: Prellfreier Schalter/Taster**

- Bei einem elektromechanischen Schalter kommt es durch mechanische Störeffekte oft zu einem sogenannten „Prellen“ (bouncing) des Schalters.
- Statt eines sofortigen Schließens (bzw. Öffnens) des elektrischen Kontaktes ruft die Betätigung des Schalters kurzzeitig ein mehrfaches Schließen und Öffnen hervor.
- Ursache ist ein elastisches Zurückprallen gegen die Federung des Schalters. Dieses ist beim Schließen des Schalters wesentlich häufiger als beim Öffnen.

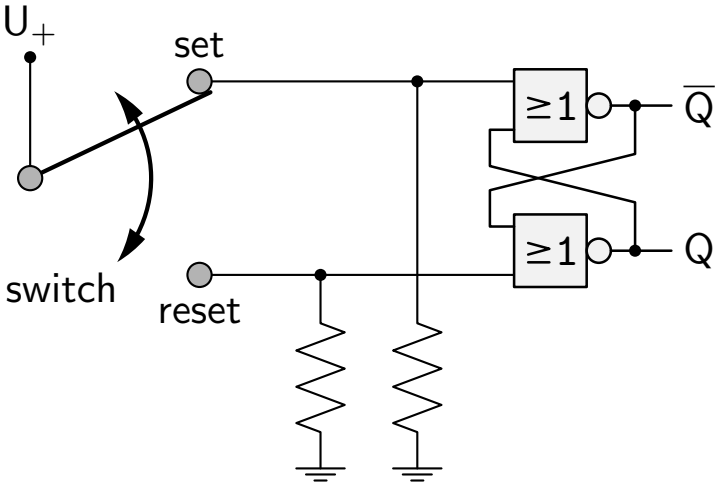


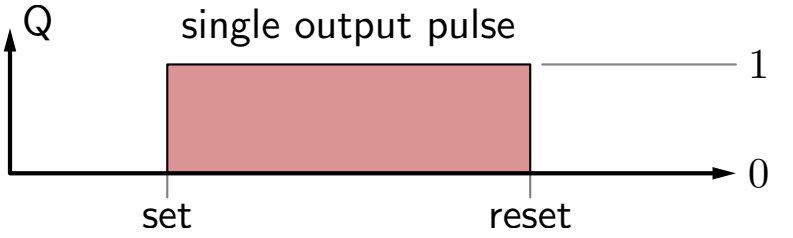



# Einfache Bistabile Kippstufen: Beispielanwendung

## Beispielanwendung: Prellfreier Schalter/Taster

- Wird an die Schalterkontakte eine bistabile Kippstufe angeschlossen, so wird das Prellen unterdrückt, da ein mehrfaches Einschaltsignal nur einmalig zu einer Zustandsänderung führt.
- Erst ein explizites Ausschaltsignal setzt die Kippstufe zurück.
- Solch ein prellfreier Schalter ist für bestimmte Anwendungen wesentlich (z.B. für eine manuelle Taktung zur Schaltungsprüfung).



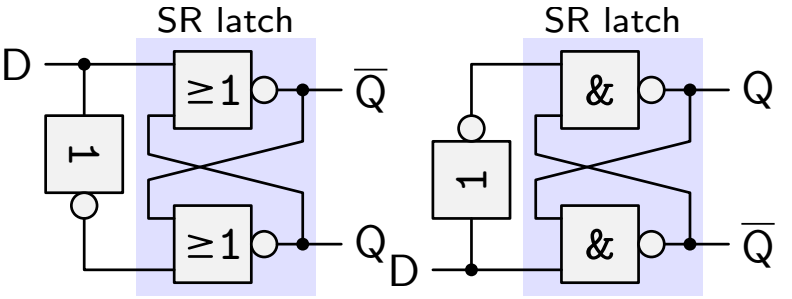
# Bistabile Kippstufen: Erweiterungen

- Durch vorgeschaltete Zusatzgatter kann die problematische Eingabe $S = R = 1$ bzw. $\overline{S} = \overline{R} = 0$ ausgeschlossen werden.
- Einfachste Möglichkeit: Stelle sicher, daß $R = \neg S$. Dies ergibt den **D-Riegel**.

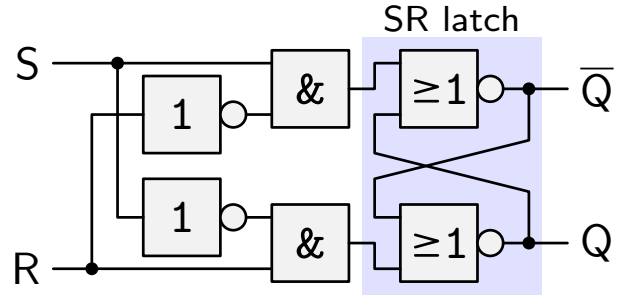
  Nachteil: Die Speichereingabe $S = R = 0$ geht ebenfalls verloren!
- Ebenfalls sehr einfache Möglichkeit: Bilde die Eingabe $S = R = 1$ auf $S = R = 0$ ab, so daß auch bei dieser Eingabe der Zustand gespeichert wird. Dies ergibt den **E-Riegel**.

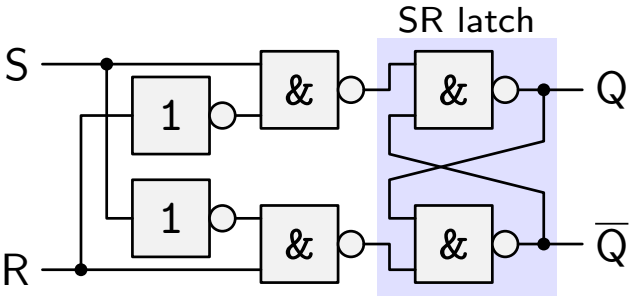
  Man beachte, daß die NAND-Form keine negierten Eingaben mehr hat!

**D-Riegel** (D latch, data, delay)




**E-Riegel** (E latch)

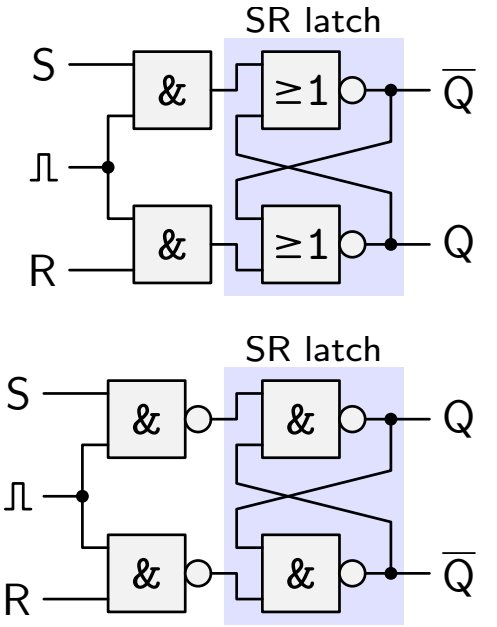


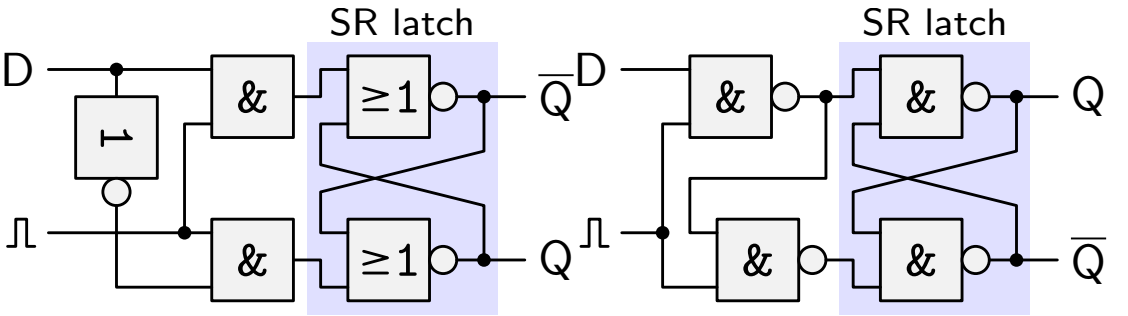
# Bistabile Kippstufen: Taktpegelsteuerung

- Eine weitere einfache Möglichkeit besteht in einer bedingten Sperrung der Eingaben, so daß die Eingabe $S = R = 1$ zumindest im Sperrzustand keinen Schaden anrichten kann. Dies führt zum **sperrbaren SR-Riegel**.
- Die bedingte Sperrung wird gewöhnlich mit einem Taktsignal („⎍“) gesteuert, so daß nur bei 1-Taktpegel die Eingaben übernommen werden: **Taktpegel-** oder **Taktzustandssteuerung**
- Man beachte wieder, daß die NAND-Form keine negierten Eingaben mehr hat!

**sperrbarer SR-Riegel**
(gated SR latch)

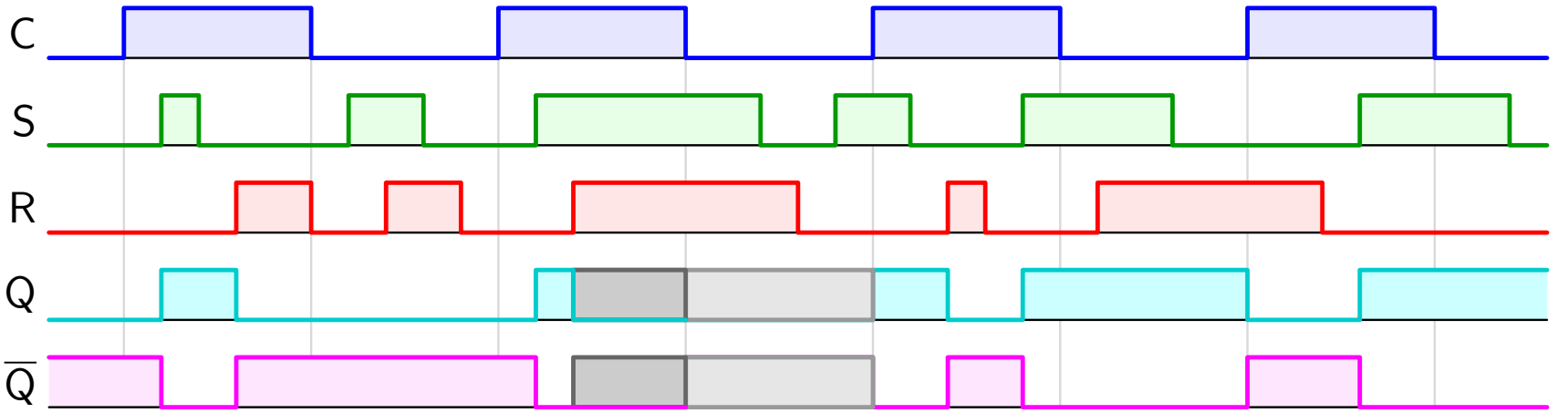



- Zusätzliches Sicherstellen von $R = \neg S$ ergibt den **sperrbaren D-Riegel** (gated D latch).



# Einfache Bistabile Kippstufen (SR-Riegel)

**Beispiel-Signalverlauf für einen sperrbaren/getakteten SR-Riegel:**

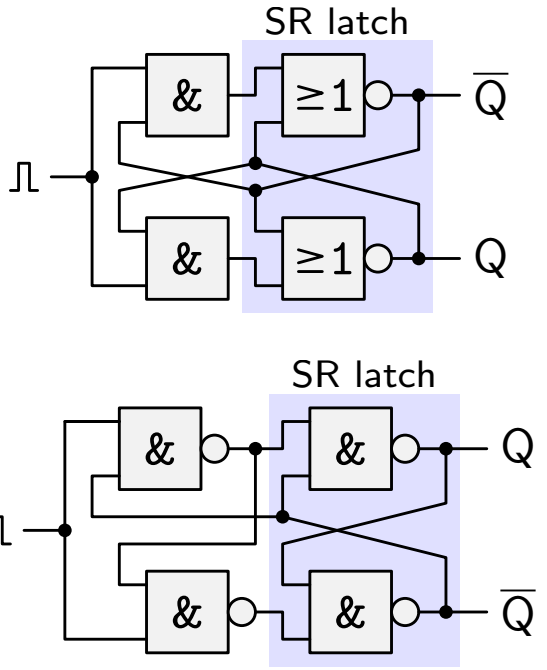



- Man beachte, daß die Eingaben S (set) und R (reset) nur eine Wirkung auf die Ausgaben Q und $\overline{Q}$ haben, wenn das Taktsignal C auf 1-Pegel ist.
- Die Ausgaben Q und $\overline{Q}$ bleiben während des 0-Pegels des Taktsignals auf dem Wert, den sie hatten, als das Taktsignal auf 0-Pegel wechselte.
- In den grau markierten Abschnitten ist der SR-Riegel in einem inkonsistenten Zustand (dunkelgrau) oder hat unbekannte Ausgaben (hellgrau).

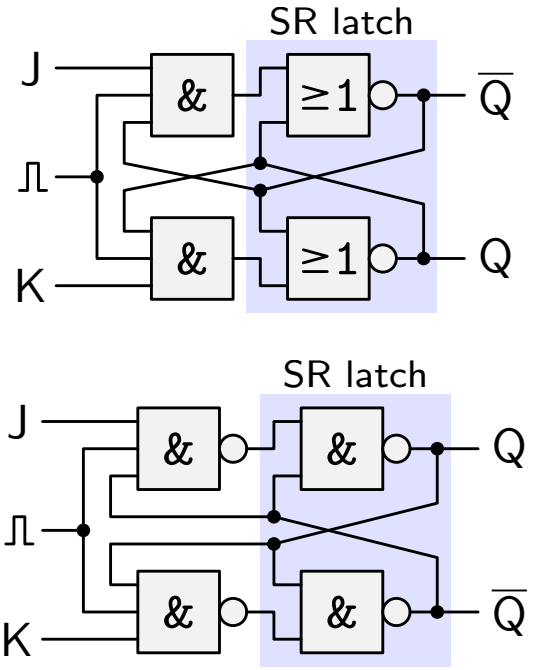
# Bistabile Kippstufen: Erweiterungen

- Durch eine Rückkopplung der Ausgaben des SR-Riegels auf die Sperrgatter läßt sich erreichen, daß die Eingabe $S = R = 1$ eine neue Funktion erhält: das Umkehren (toggle) des bestehenden Speicherzustandes.

**T-Riegel** (T latch, toggle)




**JK-Riegel** (JK latch)

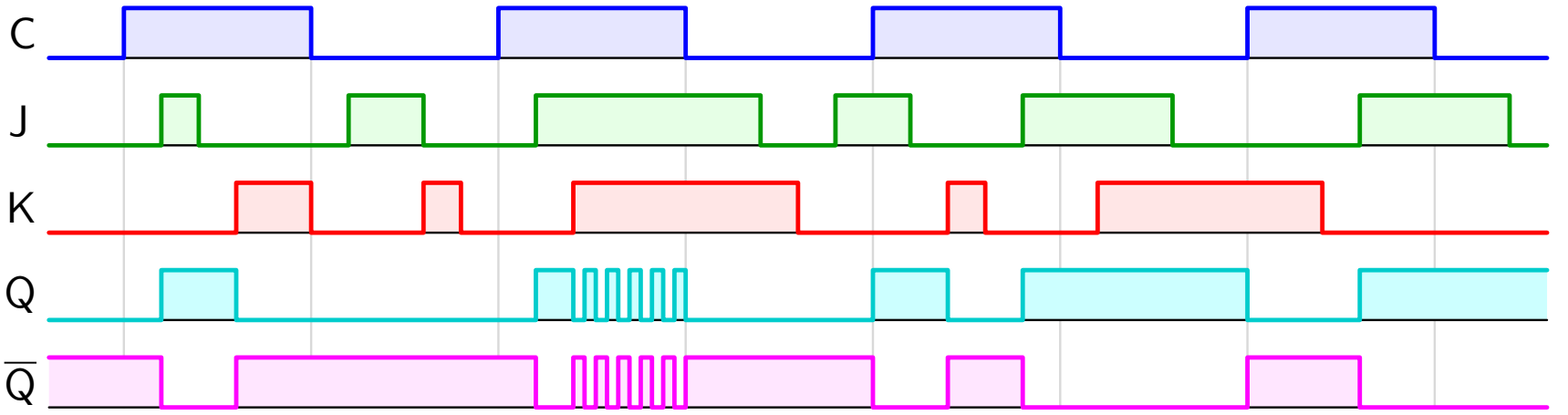



Der JK-Riegel ist nicht(!) nach Jack Kilby, einem früheren Angestellten von Texas Instruments, benannt. Die Bezeichnungen $J$ und $K$ wurden bereits von [Eldred C. Nelson 1953] in einem Patentantrag benutzt (US 2850566 A).

- Man beachte aber: Das durch die Eingabe $S = R = 1$ bewirkte Umkehren (toggle) führt zu Laufzeitproblemen (Schwingungen), da sich kein stabiler Zustand einstellt!

# Bistabile Kippstufen: Erweiterungen

**Beispiel-Signalverlauf für einen JK-Riegel:**

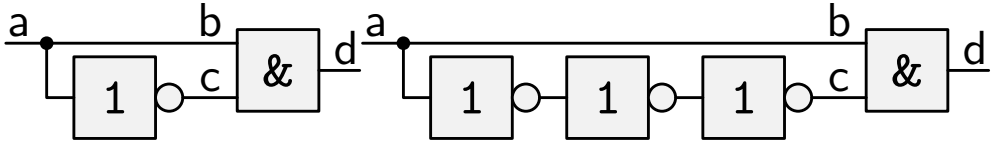



- Wie bei einem SR-Riegel haben die Eingaben S und R nur eine Wirkung auf die Ausgaben Q und $\overline{Q}$, wenn das Taktsignal C auf 1-Pegel ist.
- Die Ausgaben Q und $\overline{Q}$ bleiben während des 0-Pegels des Taktsignals auf dem Wert, den sie hatten, als das Taktsignal auf 0-Pegel wechselte.
- Sind die Eingaben $J = K = 1$, wird der Zustand des JK-Riegels in schneller Folge (Gatterlaufzeit) immer wieder umgekehrt (toggle).

# Bistabile Kippstufen: Taktflankensteuerung

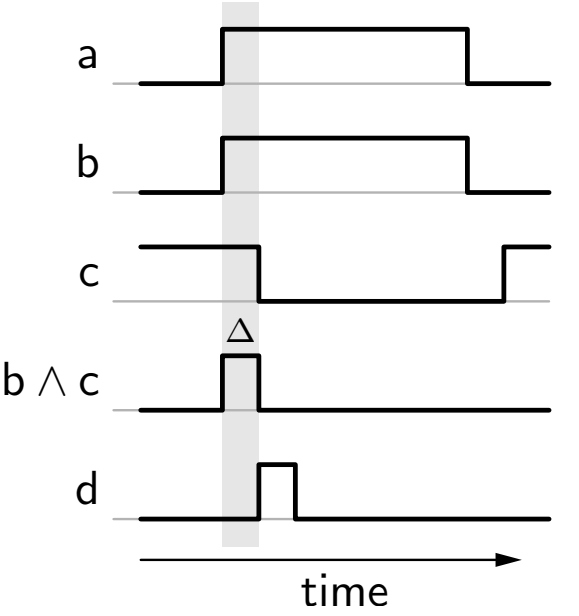
- Problem der **Taktpegelsteuerung** ist, daß die Eingaben während der gesamten 1-Phase des Taktes eine Wirkung auf den Zustand des Riegels ausüben.

- Besser wäre eine Übernahme der Eingaben nur an der steigenden Flanke (oder nur für eine kurze Zeit nach der steigenden Flanke): **Taktflankensteuerung**.




- NOT-Gatter (Inverter) können als **Verzögerungsglieder** dienen.

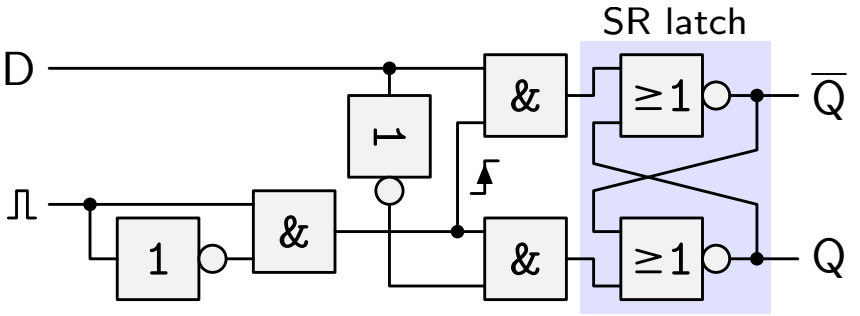
- Zusammen mit einem AND-Gatter bilden sie ein **Impulsglied**, durch das aus einer Taktflanke ein kurzer **Taktimpuls** erzeugt wird.

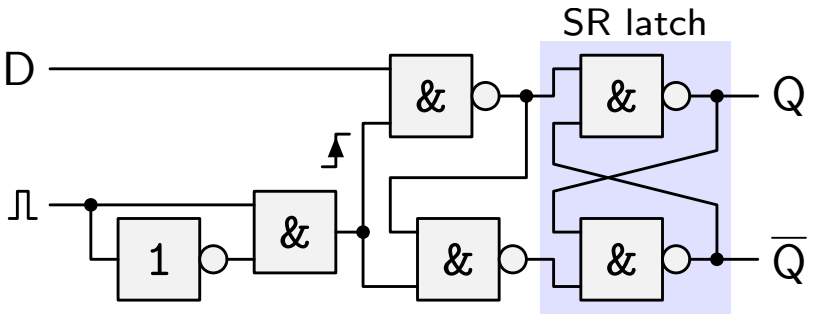



- Durch geschickte Abstimmung der Taktimpulsbreite und der Gatterlaufzeiten des Riegels kann vermieden werden, daß Schwingungen/Oszillationen auftreten.

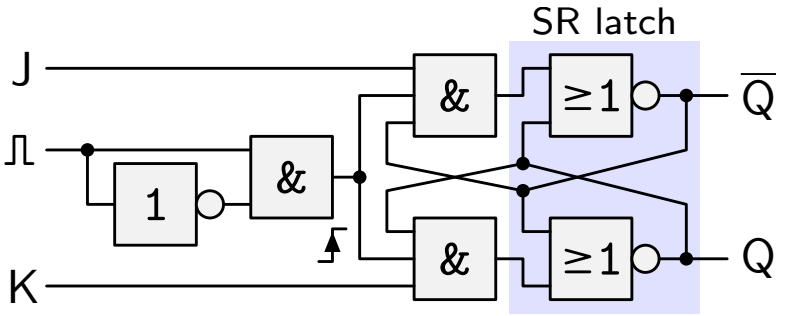
# Bistabile Kippstufen: Taktflankensteuerung

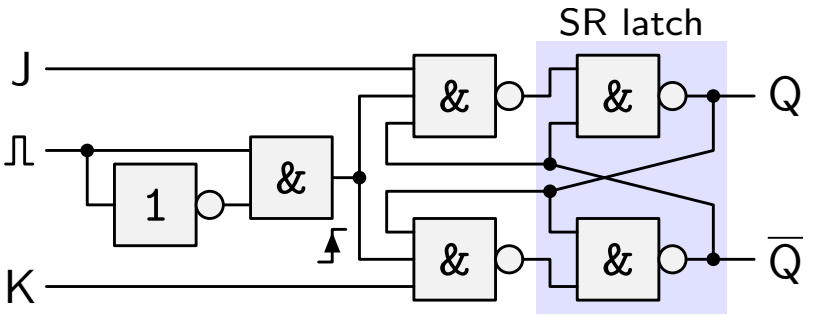
**D-Flipflop** (D flip flop)




**JK-Flipflop** (JK flip flop)

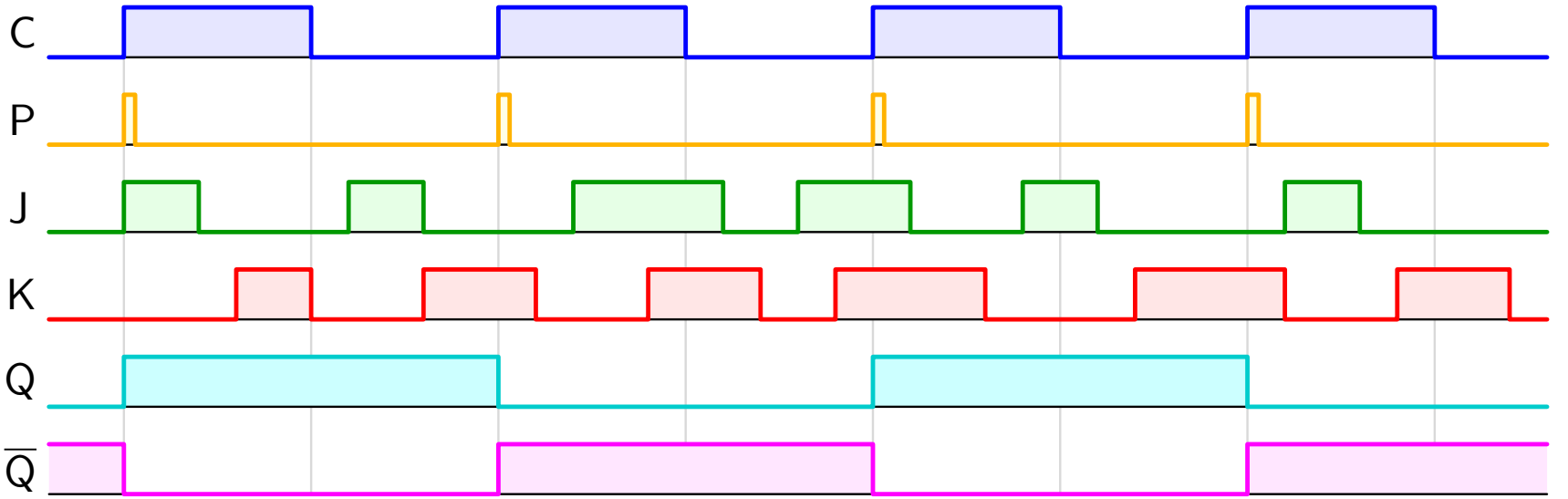



- Bei **Taktflankensteuerung** spricht man nicht mehr von Riegel (dieser ist direkt oder taktpegelgesteuert), sondern von **Flipflop**.
- Taktflankensteuerung mindert Laufzeitprobleme und macht Schaltungen robuster. Da die Gatterlaufzeiten von den Umgebungsbedingungen abhängen können (z.B. Temperatur), kann es aber trotzdem noch zu Laufzeitproblemen kommen.

# Bistabile Kippstufen: Taktflankensteuerung

**Beispiel-Signalverlauf für ein JK-Flipflop:**




- Durch ein Impulsglied wird aus dem Taktsignal (aus seinen steigenden Flanken) eine Folge P kurzer Taktimpulse erzeugt, die zur Steuerung des Flipflops dienen.

  Nur während dieser kurzen Taktimpulse haben die Eingaben J und K Wirkung.

- Man beachte, daß der Zustand bei $J = K = 1$ nur einmal umgekehrt wird.
  Dazu muß aber die Taktimpulsbreite klein genug sein (Gatterlaufzeit).

# Bistabile Kippstufen: Master-Slave-Prinzip

- Taktflankensteuerung mindert Laufzeitprobleme, behebt sie aber nicht vollständig.

- Besonders hintereinandergeschaltete Flipflops können noch zu Problemen führen, wenn eine vorangehende Stufe schon ihre Ausgaben ändert, bevor die nachfolgende Stufe die Eingabeauswertung abgeschaltet hat. (Gatterlaufzeiten!)

- Derartige Probleme werden durch das **Master-Slave-Prinzip** vermieden.

  - Es werden zwei Riegel (z.B. ein D-, ein SR-Riegel) hintereinandergeschaltet. Den ersten (vorderen) Riegel nennt man den Master-Riegel, den zweiten (hinteren) Riegel nennt man den Slave-Riegel.

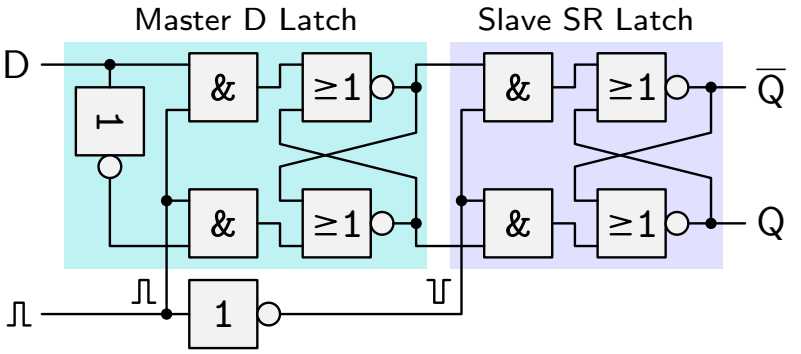
  - Während des 1-Taktpegels wertet der Master-Riegel die Eingaben aus. In dieser Zeit sind die Ausgaben des Slave-Riegels stabil.

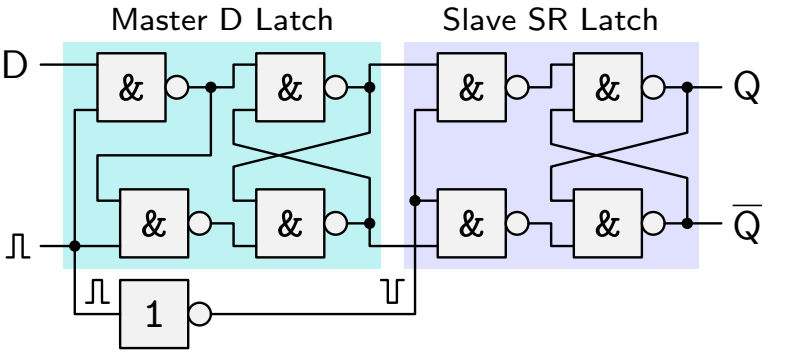
  - Während des 0-Taktpegels übernimmt der Slave-Riegel die Ausgaben des Master-Riegels. Der Master-Riegel wertet in dieser Zeit keine Eingaben aus.

  - Bei Ketten aus Master-Slave-Riegeln ändern sich während des 1-Taktpegels nur die Master-Riegel, während des 0-Taktpegels nur die Slave-Riegel. Dies vermeidet (fast) alle Laufzeitprobleme.

# Bistabile Kippstufen: Master-Slave-Prinzip

**Master-Slave-D-Riegel** (master slave D latch)

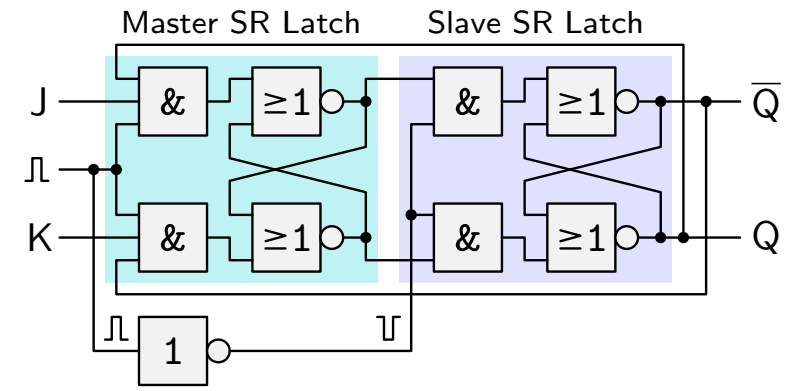


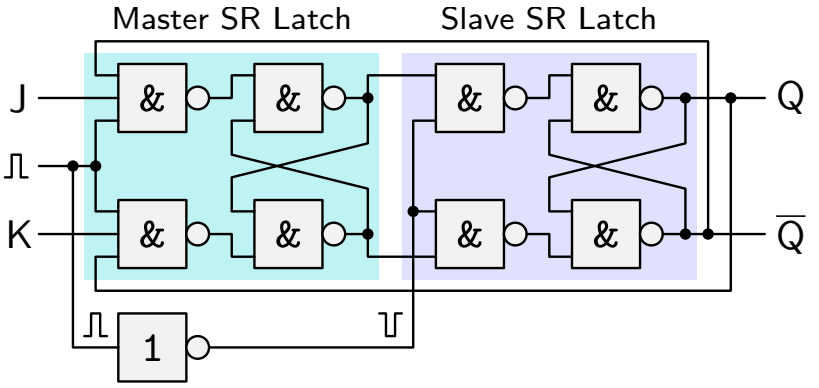




- Der Slave-Riegel bekommt ein Taktsignal, das gegenüber dem des Master-Riegels invertiert ist: **Zweipegelsteuerung** statt einfacher Taktpegelsteuerung.
- Dadurch sind Master-Riegel und Slave-Riegel nie gleichzeitig aktiv:
  - Die Ausgaben des Master-Riegels sind stabil, wenn sie der Slave-Riegel als Eingaben auswertet.
  - Die Ausgaben des Slave-Riegels sind stabil, wenn der Master-Riegel einer Folgestufe (oder einer Rückkopplung!) sie als Eingaben auswertet.

# Bistabile Kippstufen: Master-Slave-Prinzip

**Master-Slave-JK-Riegel**
(master slave JK latch)

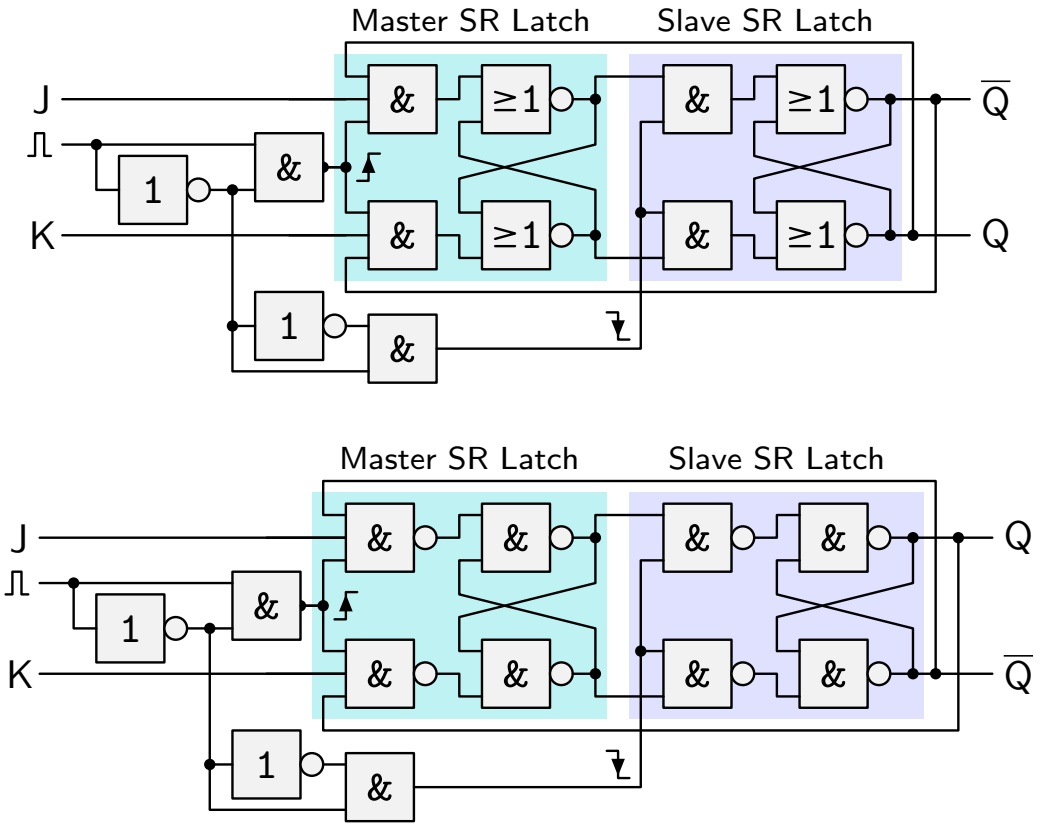



- Durch das Master-Slave-Prinzip ist auch ein Umkehren des Zustands (toggle) bei Eingabe von $J = K = 1$ kein Problem mehr.
- Denn die Ausgabe des Slave-Riegels ist stabil, wenn der Master-Riegel die Eingaben auswertet; der Slave-Riegel übernimmt den Zustand erst, wenn die Ausgaben des Master-Riegels stabil sind, weil seine Eingaben abgeschaltet sind.
- Es kommt daher nicht zu Schwingungen durch ständiges Umkehren des Zustandes (weil eine veränderte Ausgabe sofort auf die Eingabe rückwirkt), sondern nur zu einer einzigen Zustandsumkehr je Taktzyklus.

# Bistabile Kippstufen: Master-Slave-Prinzip

**Master-Slave-JK-Flipflop**
(master slave JK flip flop)




Noch sicherer:
Kombination von
Master-Slave-Prinzip und
Taktflankensteuerung:
**Zweiflankensteuerung**

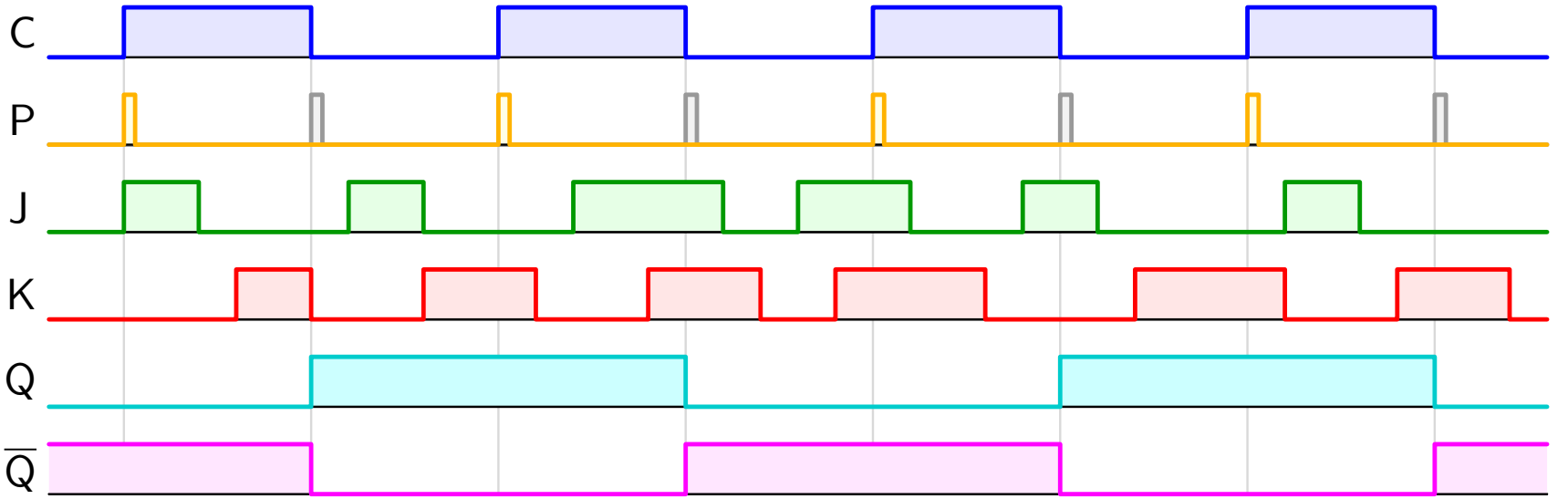
(Man beachte die Impulsglieder!)

Der Master-Riegel wertet
nur bei steigender Flanke,
der Slave-Riegel wertet
nur bei fallender Flanke
seine Eingaben aus.

Dadurch sehr saubere
zeitliche Trennung
aller Berechnungen;
keine Laufzeitprobleme
mehr möglich.

# Bistabile Kippstufen: Master-Slave-Prinzip

**Beispiel-Signalverlauf für ein Master-Slave-JK-Flipflop:**

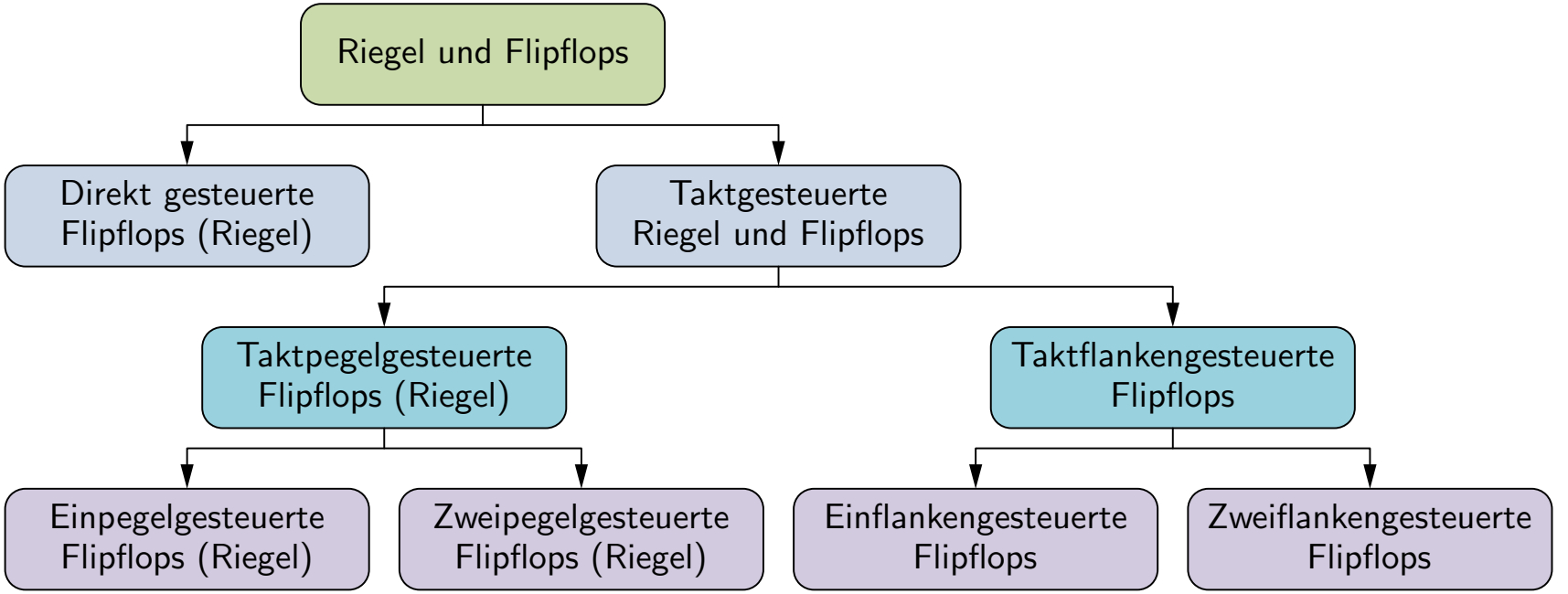



- Durch ein Impulsglied wird aus dem Taktsignal (aus seinen steigenden Flanken) eine Folge P kurzer Taktimpulse erzeugt, die zur Steuerung des Flipflops dienen.

  Nur während dieser kurzen Taktimpulse haben die Eingaben J und K Wirkung.

- Man beachte, daß die Ausgaben erst bei der fallenden Flanke des Taktsignals C ihren neuen Zustand annehmen („retardierte“, d.h., verzögerte Ausgänge).

# Bistabile Kippstufen: Zusammenfassung




- Im Englischen unterscheidet man meist recht sauber zwischen **Riegel** (latch, direkt oder taktpegelgesteuert) und **Flipflop** (flip flop, taktflankengesteuert).

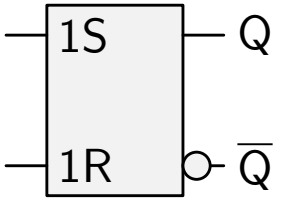
- Im Deutschen werden dagegen oft alle auf bistabilen Kippstufen beruhenden Schaltungen als Flipflops bezeichnet (so auch in obiger Hierarchie).
  Die Unterschiede müssen dann durch Adjektive zum Ausdruck gebracht werden.

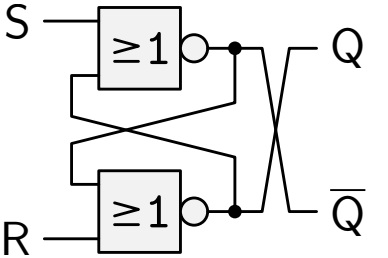
# Riegel und Flipflops: Schaltzeichen

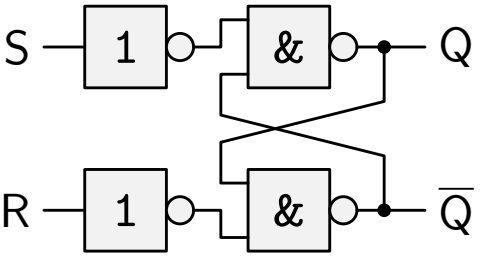
- Da Riegel und Flipflops sehr häufig verwendete Schaltungsbausteine sind, werden sie gewöhnlich nicht durch einzelne, zusammengeschaltete Gatter, sondern durch eigene Schaltzeichen dargestellt.

- Diese Schaltzeichen sind einfache Rechtecke, mit
  - den Eingängen links und den Ausgängen rechts,
  - gewöhnlich unnegierten Eingängen (wenn nicht durch „Negationsblasen“ angezeigt),
  - dem „Setzen“-Eingang (z.B. S, D, J) oben und dem „Rücksetzen“-Eingang (z.B. R, K) unten,
  - dem Ausgang Q oben und dem Ausgang $\overline{Q}$ unten,
  - einer Negationsblase am Ausgang $\overline{Q}$, es sei denn, das Rechteck ist durch eine horizontale gestrichelte Linie in der Mitte in zwei Hälften geteilt.

- Man beachte: Das gleiche Schaltzeichen kann für verschiedene konkrete Gatterimplementierungen stehen.



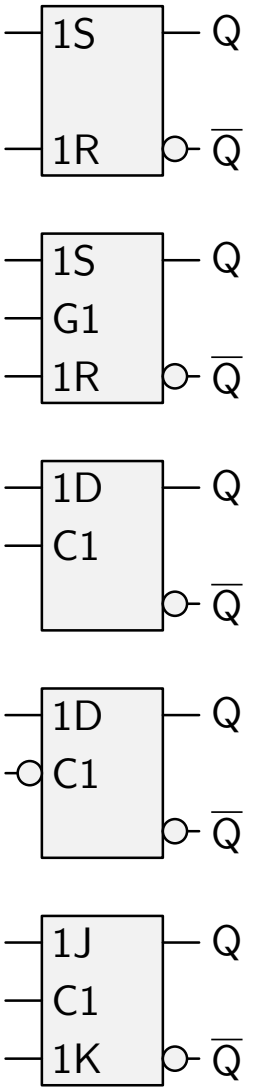
# Riegel und Flipflops: Schaltzeichen

- Weitere Regeln für Riegel- und Flipflop-Schaltzeichen:
  - Die „Setzen“- und „Rücksetzen“-Eingänge werden mit einer 1 links von dem sie bezeichnenden Buchstaben geschrieben (z.B. 1S, 1R, 1D, 1J, 1K).

    Diese Eingänge sind links oben und ggf. unten.
  - Ein Eingang, der die Wirkung der „Setzen“- und „Rücksetzen“-Eingänge modifiziert (z.B. der Takteingang), wird mit einer 1 rechts von dem ihn bezeichnenden Buchstaben geschrieben (z.B. G1, C1).

    Dieser Eingang ist links in der Mitte oder unten (falls dort kein anderer Eingang ist).

    Es steht G für „gated“ und C für „clock“.
  - Hat der Takteingang eine „Negationsblase“, so reagiert der Riegel bei 0-Taktpegel, sonst bei 1-Taktpegel (d.h. auch: Standard ist **Taktpegelsteuerung**).

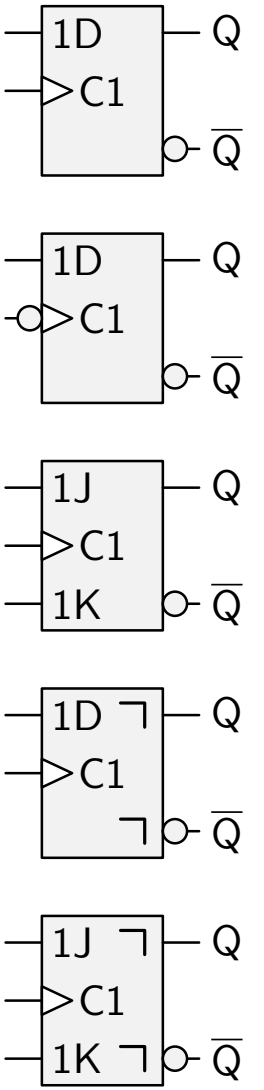


# Riegel und Flipflops: Schaltzeichen

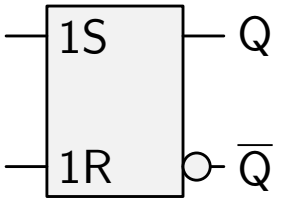
- Weitere Regeln für Riegel- und Flipflop-Schaltzeichen:
  - Ein (weißes) Dreieck an einem Takteingang bedeutet **Taktflankensteuerung** (Vorhandensein eines Impulsgliedes). In diesem Fall handelt es sich um ein **Flipflop** (sonst um einen **Riegel**).
  - Hat der Takteingang eine „Negationsblase“, so reagiert das Flipflop bei fallender, sonst bei steigender Taktflanke.
  - Winkel an den Ausgängen Q und $\overline{Q}$ deuten durch eine stilisierte fallende Flanke Master-Slave-Flipflops an.

    Man sagt in diesem Fall auch, die Ausgänge seien „retardiert“ (verzögert), weil die Ausgänge auf Änderungen der Eingänge mit einer Verzögerung (nämlich bis zur fallenden Flanke) reagieren.

    (Zwangsläufige Folge des Master-Slave-Prinzips.)

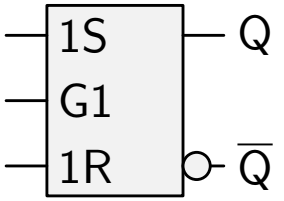


# Riegel und Flipflops: Schaltverhalten



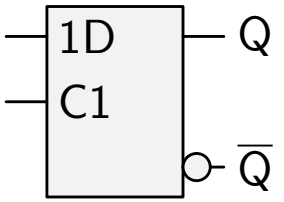

| S | R | Q | $\overline{Q}$ |
|---|---|---|---|
| 0 | 0 | Q | $\overline{Q}$ |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | ? | ? |

- einfacher SR-Riegel
- asynchron (kein Takteingang)
- metastabiler Zustand möglich




| S(t) | R(t) | Q(t+1) | $\overline{Q}$(t+1) |
|---|---|---|---|
| 0 | 0 | Q(t) | $\overline{Q}$(t) |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | ? | ? |

- sperrbarer/getakteter SR-Riegel
- synchron (Sperr-/Takteingang)
- metastabiler Zustand möglich

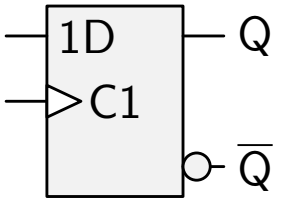



| D(t) | Q(t+1) | $\overline{Q}$(t+1) |
|---|---|---|
| 1 | 1 | 0 |
| 0 | 0 | 1 |

- taktpegelgesteuerter D-Riegel
- kein metastabiler Zustand möglich
- kein längerfristiges Speichern

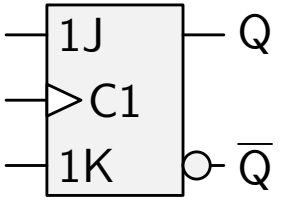
- Die Zeit $t$ ist ganzzahlig und benennt den Taktzyklus. Die Wertetabellen geben also an, wie sich der neue Zustand berechnet aus Eingaben und aktuellem Zustand.
- Der D-Riegel kann einen Zustand nicht mehr (längerfristig) speichern:
  Er gibt stets die Eingabe D des vorangehenden Taktzyklus aus.
  Das D im Namen wird daher auch gern als „delay“ interpretiert.

# Riegel und Flipflops: Schaltverhalten



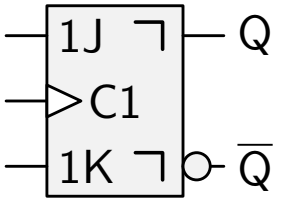

| D(t) | Q(t+1) | $\overline{Q}$(t+1) |
|---|---|---|
| 1 | 1 | 0 |
| 0 | 0 | 1 |

- taktflankengesteuertes D-Flipflop
- kein metastabiler Zustand möglich
- kein längerfristiges Speichern




| J(t) | K(t) | Q(t+1) | $\overline{Q}$(t+1) |
|---|---|---|---|
| 0 | 0 | Q(t) | $\overline{Q}$(t) |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | $\overline{Q}$(t) | Q(t) |

- taktflankengesteuertes JK-Flipflop
- kein metastabiler Zustand möglich
- Zustandsumkehrfunktion (toggle)

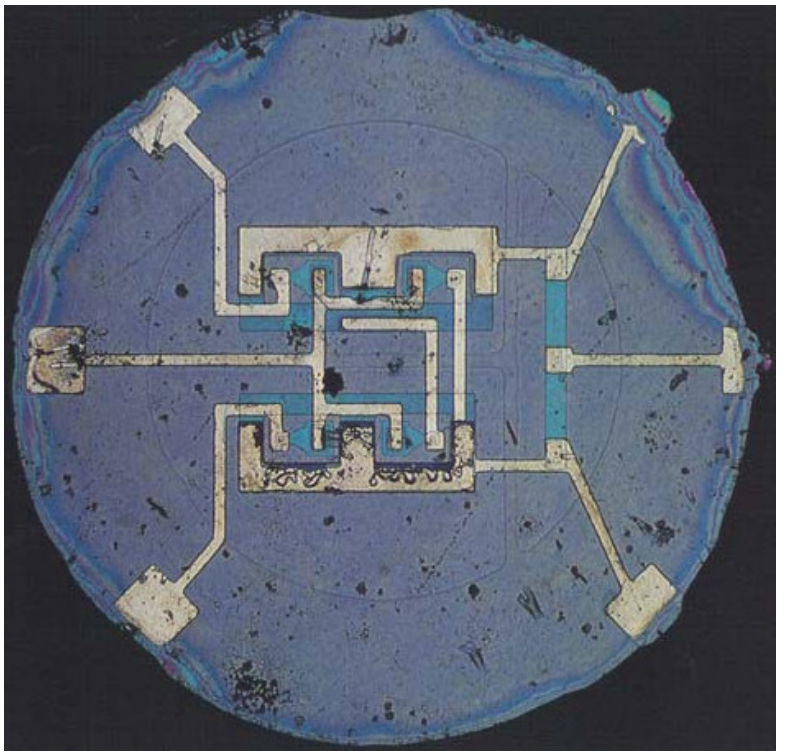



| J(t) | K(t) | Q(t+1) | $\overline{Q}$(t+1) |
|---|---|---|---|
| 0 | 0 | Q(t) | $\overline{Q}$(t) |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | $\overline{Q}$(t) | Q(t) |

- taktflankengesteuertes Master-Slave-JK-Flipflop
- Zustandsumkehrfunktion (toggle)
- verzögerte Änderung der Ausgaben

- Die Zeit $t$ ist ganzzahlig und benennt den Taktzyklus. Die Wertetabellen geben also an, wie sich der neue Zustand berechnet aus Eingaben und aktuellem Zustand.
- Beachte: Bei einem Master-Slave-Flipflop (Winkel an den Ausgängen) ändern sich die Ausgaben erst mit der fallenden Taktflanke, also um eine Taktbreite gegenüber den Eingaben verzögert („retardiert“).

# Flipflops als Elektronikbauteile





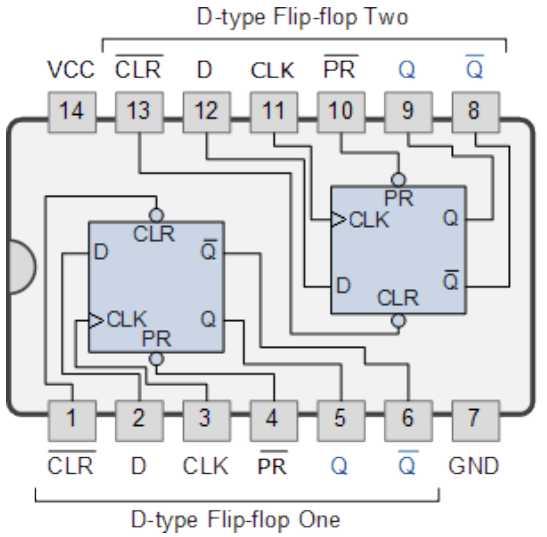

www.computerhistory.org

- Flipflop, US-Patent für den ersten integrierten Schaltkreis [Jack Kilby 1958] (Texas Instruments, Nobelpreis für Physik 2000)

  Der JK-Riegel ist jedoch **nicht** nach Jack Kilby benannt. Die Bezeichnungen $J$ und $K$ wurden bereits von [Eldred C. Nelson 1953] in einem Patentantrag benutzt (US 2850566 A).

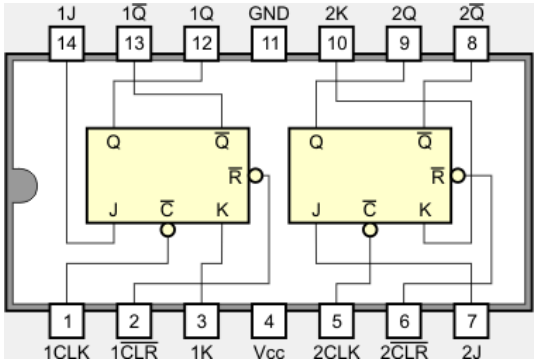
- Parallelentwicklung von [Robert Noyce 1961] (Fairchild Semiconductor)

# Flipflops als Elektronikbauteile



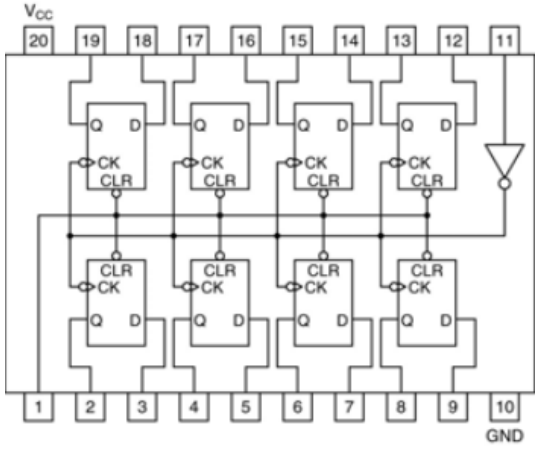

www.electronics-tutorials.ws


www.radiomuseum.org
eduinf.waw.pl





www.radiomuseum.org

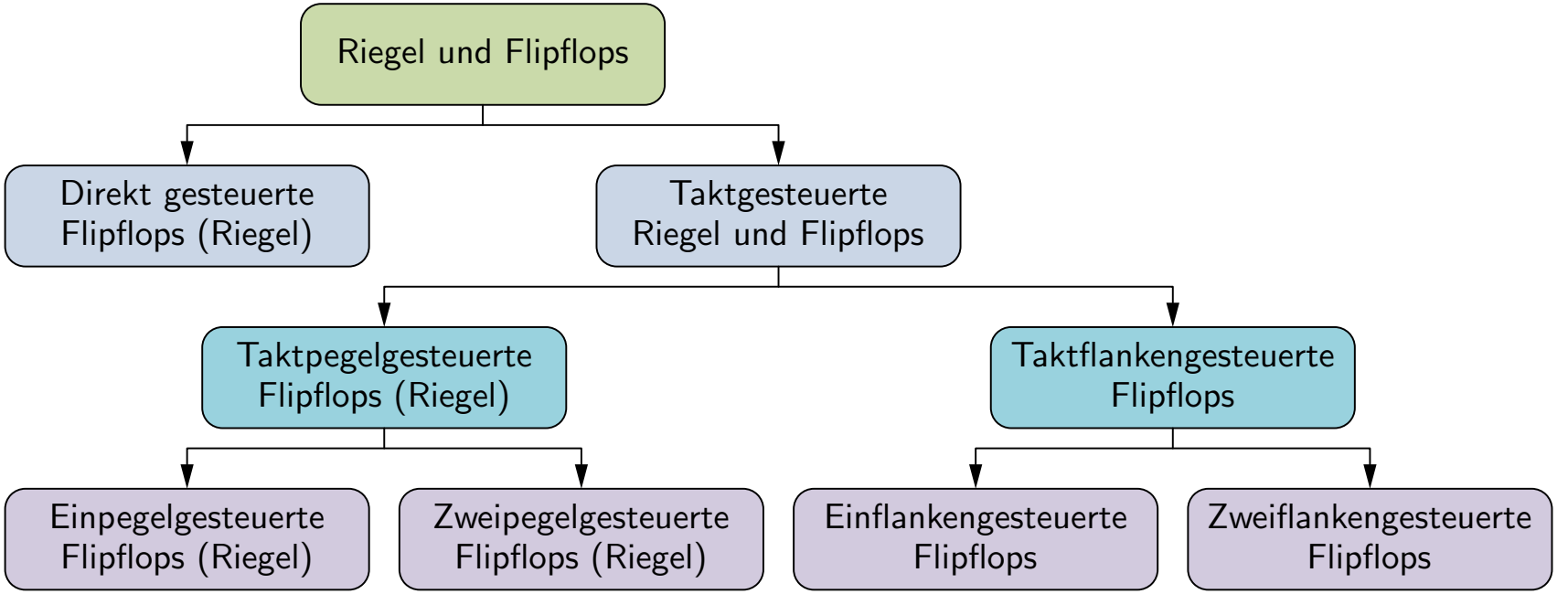



www.elemania.altervista.org


www.gpecsrl.net

- Oben links: Texas Instruments 7474
  2 taktflankengesteuerte D-Flipflops
- Unten links: Texas Instruments 74273
  8 taktflankengesteuerte D-Flipflops
- Oben rechts: Texas Instruments 7473
  2 taktpegelgesteuerte
  Master-Slave-JK-Riegel

# Bistabile Kippstufen: Zusammenfassung

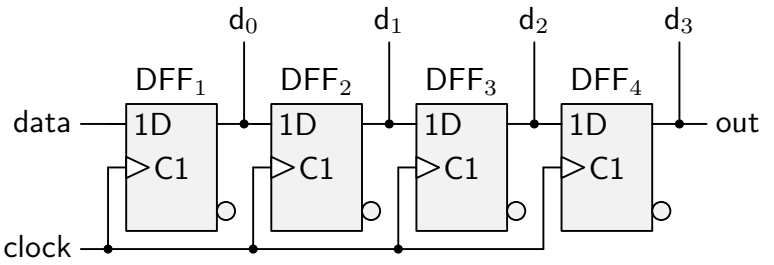


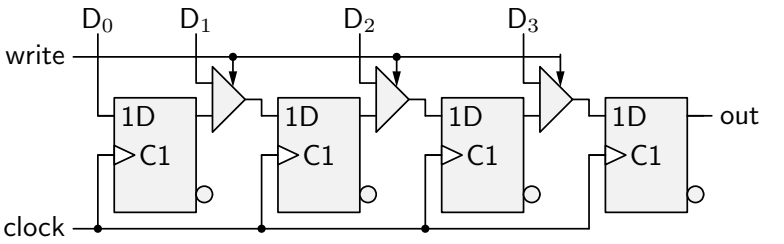

- Im Englischen unterscheidet man meist recht sauber zwischen **Riegel** (latch, direkt oder taktpegelgesteuert) und **Flipflop** (flip flop, taktflankengesteuert).

- Im Deutschen werden dagegen oft alle auf bistabilen Kippstufen beruhenden Schaltungen als Flipflops bezeichnet (so auch in obiger Hierarchie).
  Die Unterschiede müssen dann durch Adjektive zum Ausdruck gebracht werden.

# Inhalt

# Schieberegister und Parallel-Seriell-Wandler



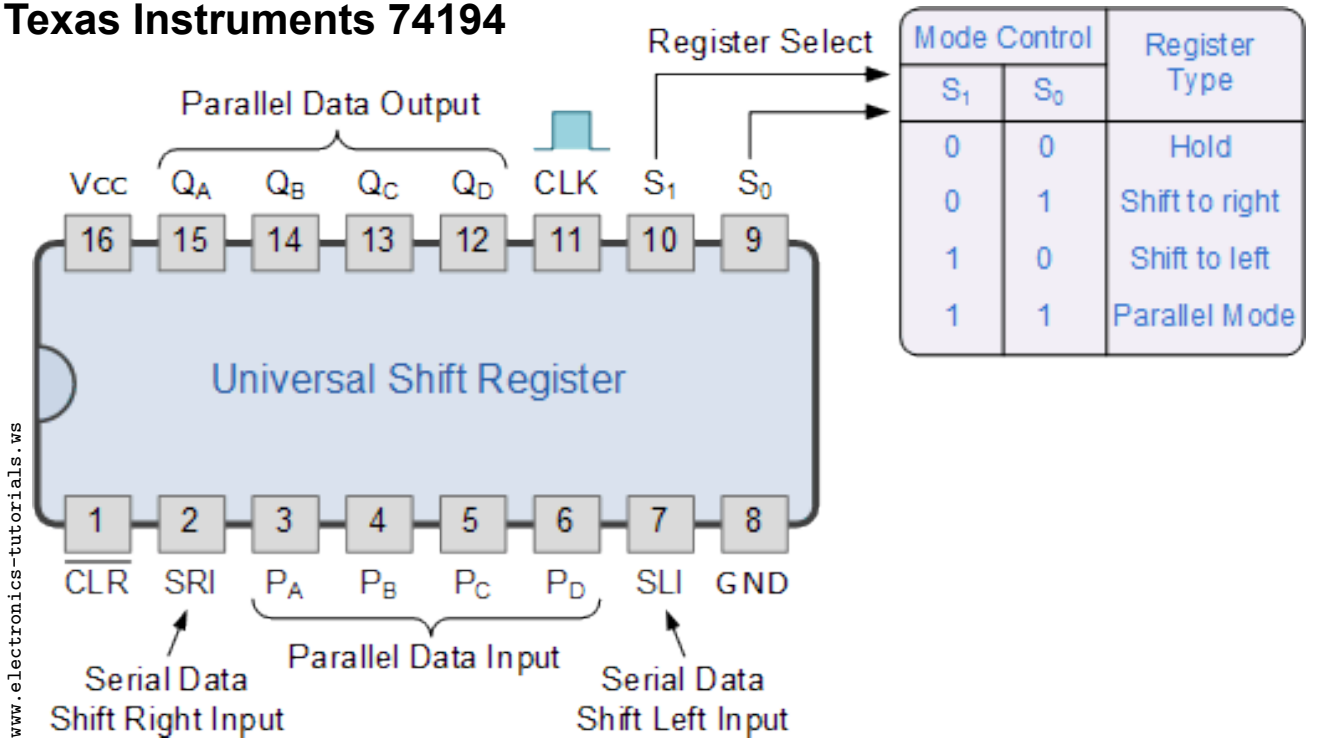

- Durch Verkettung von D-Flipflops erhält man ein **Schieberegister**: Mit jedem Taktzyklus wird der Speicherinhalt der Flipflops um ein Flipflop weitergeschoben.
- Ein Schieberegister hat eine feste Anzahl von Speicherplätzen (hier: 4)
- Schieberegister arbeiten wie Warteschlangen nach dem FIFO-Prinzip (first in, first out): Das zuerst eingespeicherte Bit verläßt das Schieberegister auch als erstes wieder.

- Schieberegister können zur Wandlung eines seriellen Datenstroms in einen parallelen (links) oder eines parallelen Datenstroms in einen seriellen (rechts) benutzt werden.
- Alle $k$ Taktzyklen (hier: $k = 4$) steht ein neues Datenwort zum Auslesen zur Verfügung (seriell nach parallel, links) oder alle $k$ Taktzyklen muß ein neues Datenwort eingespeichert werden (parallel nach seriell, rechts).

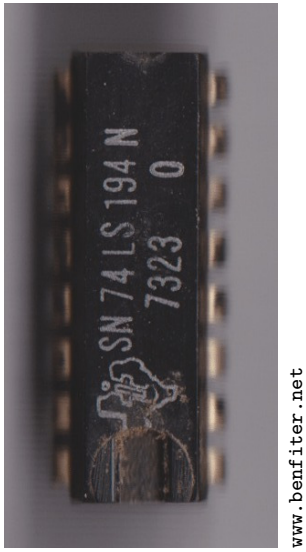
  (Beachte die 2-Wege-Multiplexer!)

# Schieberegister als Elektronikbauteile

## Texas Instruments 74194



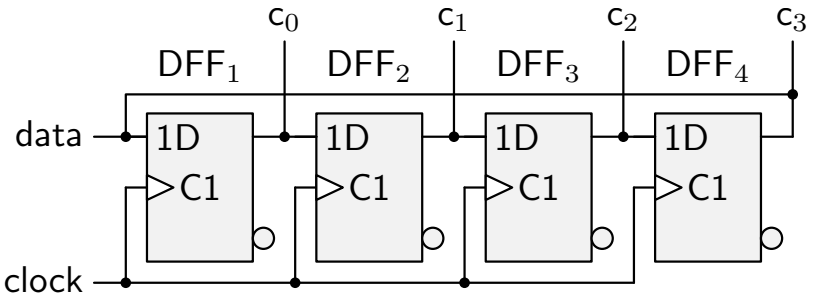

| Mode Control | | Register Type |
|---|---|---|
| $S_1$ | $S_0$ | |
| 0 | 0 | Hold |
| 0 | 1 | Shift to right |
| 1 | 0 | Shift to left |
| 1 | 1 | Parallel Mode |




- Schieberegister, die man sogar in beiden Richtungen betreiben kann (Rechts- oder Linksschieben), und die man parallel mit Daten bestücken bzw. aus denen man parallel Daten auslesen kann, gibt es als fertige Elektronikbauteile.
- Man beachte, daß diese Bauteile leicht miteinander verkettbar sind, um Schieberegister mit größerer Registerbreite zu erhalten.

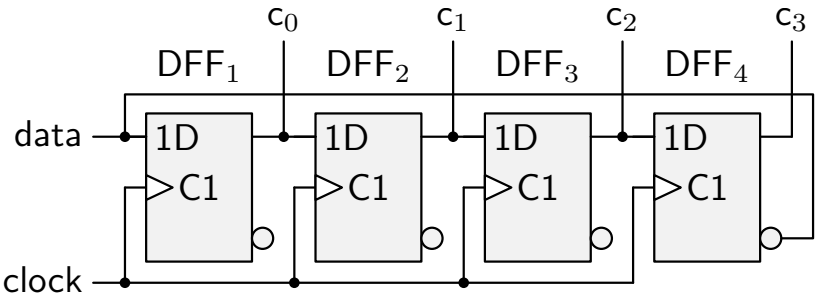
# Einfach Rückgekoppelte Schieberegister: Ringzähler

## Einfacher 4-Bit-**Ringzähler**




| Takt | $c_0$ | $c_1$ | $c_2$ | $c_3$ |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 |
| 7 | 0 | 0 | 1 | 0 |

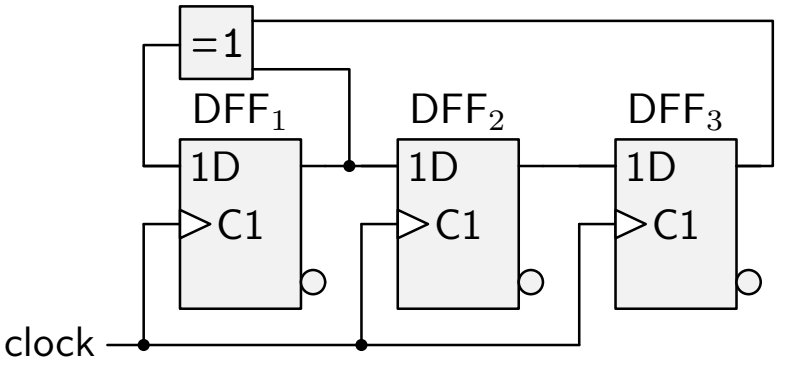
## 4-Bit-**Johnson-Ringzähler**

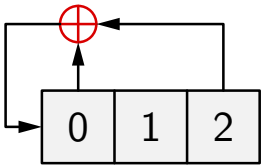


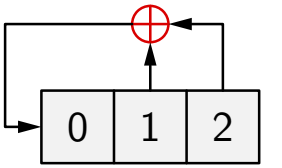

| Takt | $c_0$ | $c_1$ | $c_2$ | $c_3$ |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 1 | 1 | 1 | 0 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 1 | 1 |
| 6 | 0 | 0 | 1 | 1 |
| 7 | 0 | 0 | 0 | 1 |

- Man beachte die unterschiedliche Rückkopplung der beiden Schieberegister, durch die unterschiedliche Bitmusterfolgen bewirkt werden.

# Linear Rückgekoppelte Schieberegister



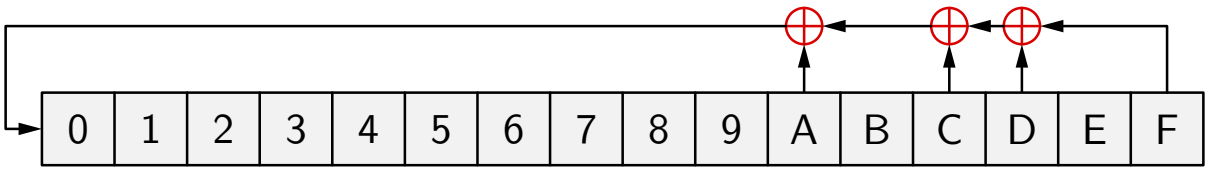




- Durch kompliziertere Rückkopplungen, die nicht nur vom letzten Flipflop, sondern auch von dazwischenliegenden rückkoppeln (Verknüpfung der Rückkopplungen über ein exklusives Oder) kann man kompliziertere Bitfolgen erzeugen.
- Solche **linear rückgekoppelten Schieberegister** (linear feedback shift register, LFSR) werden vereinfacht wie rechts gezeigt dargestellt.

| Takt | $b_0$ | $b_1$ | $b_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |
| 2 | 1 | 1 | 0 |
| 3 | 1 | 1 | 1 |
| 4 | 0 | 1 | 1 |
| 5 | 1 | 0 | 1 |
| 6 | 0 | 1 | 0 |
| 7 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 |
| 9 | 1 | 1 | 0 |
| ⋮ | ⋮ | ⋮ | ⋮ |

| Takt | $b_0$ | $b_1$ | $b_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 |
| 3 | 1 | 0 | 1 |
| 4 | 1 | 1 | 0 |
| 5 | 1 | 1 | 1 |
| 6 | 0 | 1 | 1 |
| 7 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 |
| 9 | 0 | 1 | 0 |
| ⋮ | ⋮ | ⋮ | ⋮ |

# Linear Rückgekoppelte Schieberegister




- Das obige **linear rückgekoppelte Schieberegister** (linear feedback shift register, LFSR) hat Rückkopplungen von den Positionen 10, 12, 13 und 15.

- Es erzeugt Bitmuster mit einer Zykluslänge von $65535 = 2^{16} - 1$.
  Dies ist die maximal mögliche Zykluslänge für solche Schieberegister mit Breite 16.

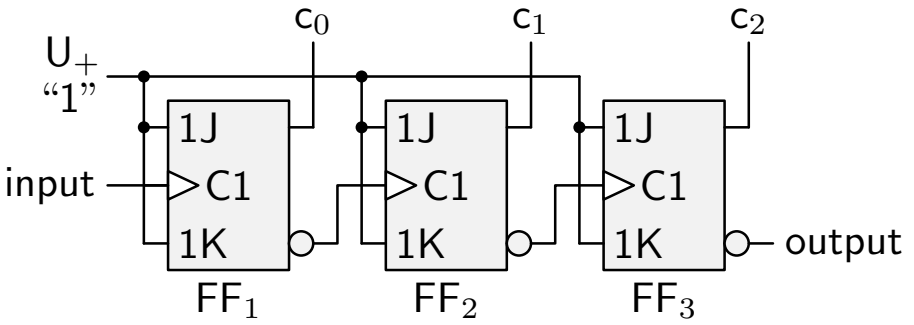
  (Theorie: Darstellung binärer Polynome, Fibonacci- und Galois-LFSR,
  nach [Leonardo Fibonacci 1170–1240] und [Évariste Galois 1811–1832])

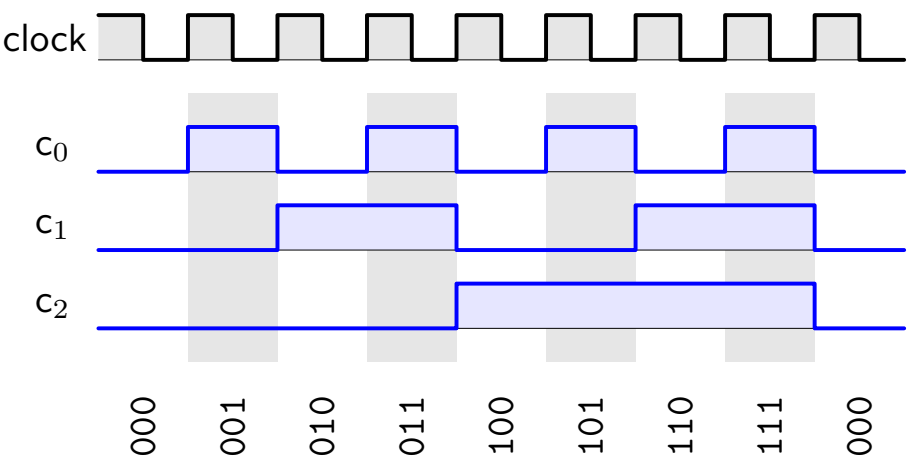
- Verwendung solcher linear rückgekoppelten Schieberegister z.B.:

  - **Erzeugung von Pseudozufallzahlen**
    (zwar sind die Zahlen nicht wirklich zufällig, da durch ein deterministisches Verfahren erzeugt, aber sie erfüllen viele Anforderungen an Zufallszahlen)

  - **Zyklische Redundanzprüfung**
    (ein Verfahren zur Erkennung von Fehlern bei der Datenübertragung)

# Zähler



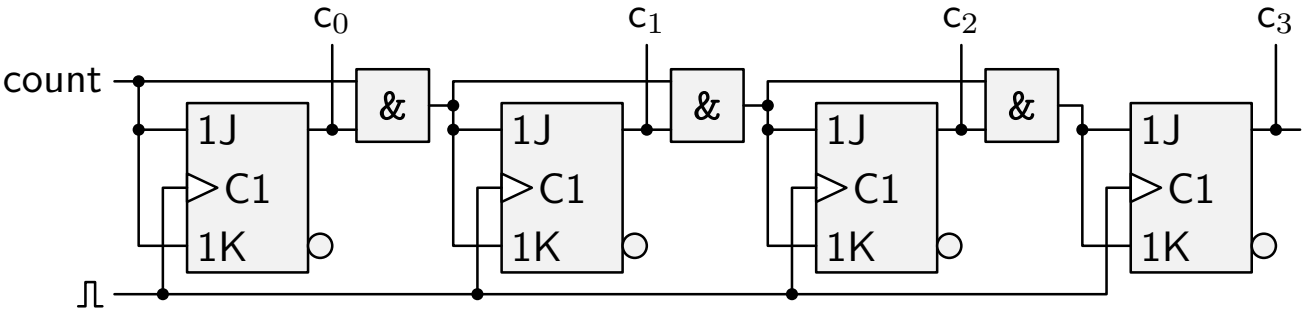

- Bekanntlich kehrt ein JK-Flipflop bei Eingaben J = K = 1 seinen Zustand mit jeden Taktzyklus um (toggle).
- Daher: Werden die Eingänge J und K fest mit der Versorgungsspannung (logische „1“) verbunden, so bewirkt ein JK-Flipflop eine Takthalbierung.
- Durch Verbinden eines Ausgangs mit dem Takteingang eines weiteren JK-Flipflops wird der Takt geviertelt etc.

- Auf diese Weise erhält man auch einen einfachen **Zähler**, nämlich, indem man die Ausgänge der Stufen
als eine Binärzahl interpretiert:

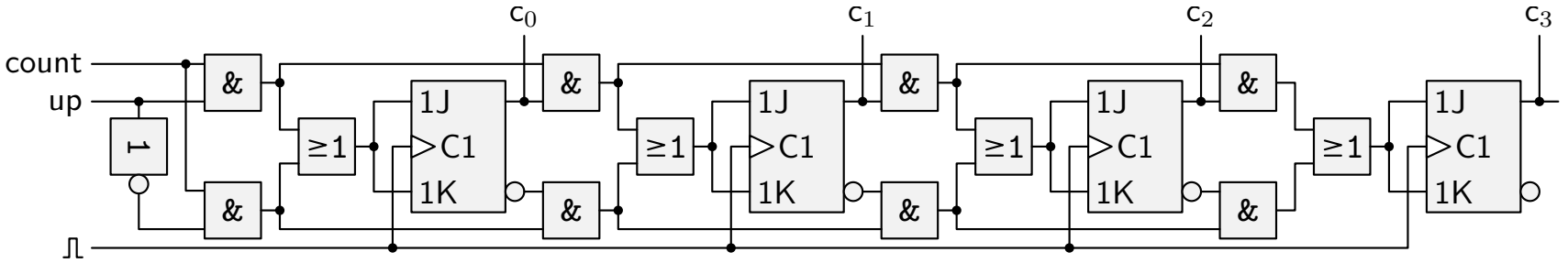
  Jeder Taktzyklus schaltet
den Zähler einen Schritt weiter.
- Zähler gehören mit
zu den wichtigsten Schaltungen.

# Steuerbare Zähler

Zähler ergeben sich auch aus anderen Verkettungen von JK-Flipflops:



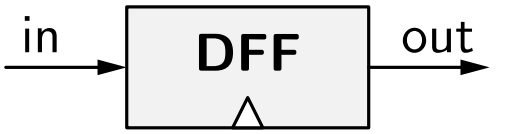

- Dieser 4-Bit-Zähler zählt nur, wenn der Eingang count $= 1$ ist.
- Man kann durch eine analoge Verknüpfung der invertierten Ausgänge $\overline{Q}$ (statt der normalen Ausgänge Q) einen Zähler erhalten, der abwärts zählt.
- Parallele Implementierung der beiden Ansätze ergibt einen Zähler, der wahlweise aufwärts (Eingang up $= 1$) oder abwärts (Eingang up $= 0$) zählen kann.



# Speicherzellen/Speicherregister

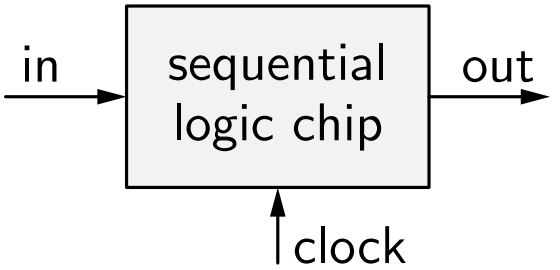
## D-Flipflop als Grundbaustein

- D-Flipflops sind grundlegende Speicherbausteine, von denen ausgehend Speicherzellen und dann Speicherregister gebildet werden können.
- Im folgenden vernachlässigen wir die Implementierung des D-Flipflop (Abstraktion: nur Schnittstelle).

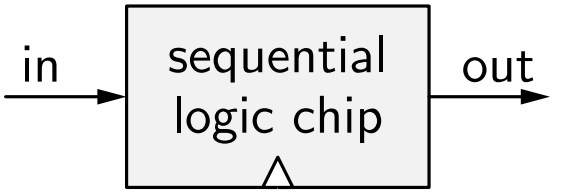



$$\text{out}(t) = \text{in}(t-1)$$

- Speicherzellen und Speicherregister bestehen aus mehreren Flipflops, die durch das gleiche (zentral erzeugte) Taktsignal gesteuert werden.
- Im folgenden bedeutet ein (weißes) Dreieck allgemein einen **Takteingang** (keine Unterscheidung von Taktpegel- und Taktflankensteuerung mehr).



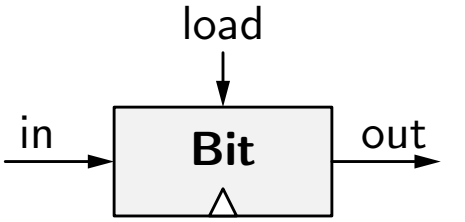

wird vereinfacht zu



# Speicherzellen/Speicherregister

Gesucht: **Speicherzelle** (genannt **Bit**), die

- ihren Zustand auf eine gegebene Eingabe ändern kann,
- ihren Zustand über die Zeit erhalten kann (bis er explizit wieder geändert wird).

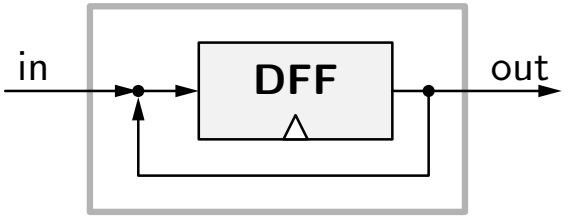



$$\begin{aligned}&\text{if } \mathsf{load}(t-1)\\ &\text{then } \mathsf{out}(t) = \mathsf{in}(t-1)\\ &\text{else } \ \mathsf{out}(t) = \mathsf{out}(t-1)\end{aligned}$$

Grundbaustein

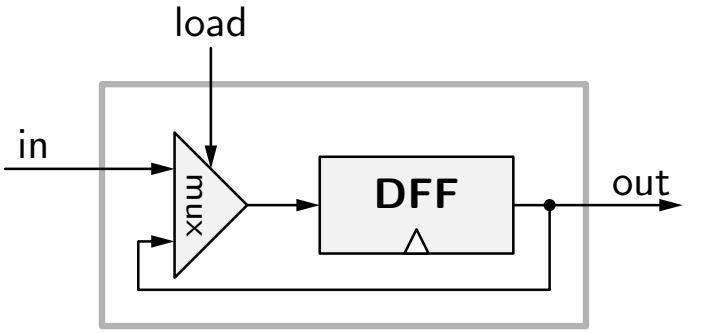

$\mathsf{out}(t) = \mathsf{in}(t-1)$

1. Versuch ✗




$\mathsf{out}(t) = \mathsf{out}(t-1)?$
$\mathsf{out}(t) = \mathsf{in}(t-1)?$

2. Versuch ✓

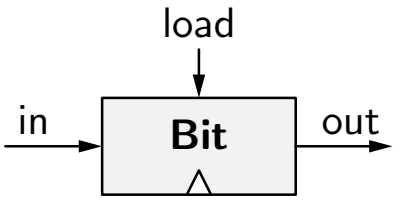



$$\begin{aligned}&\text{if } \mathsf{load}(t-1)\\ &\text{then } \mathsf{out}(t) = \mathsf{in}(t-1)\\ &\text{else } \ \mathsf{out}(t) = \mathsf{out}(t-1)\end{aligned}$$

- Ein Multiplexer ist zur Auswahl nötig, da bei einfacher Rückkopplung die Eingabe des D-Flipflop unbestimmt ist.

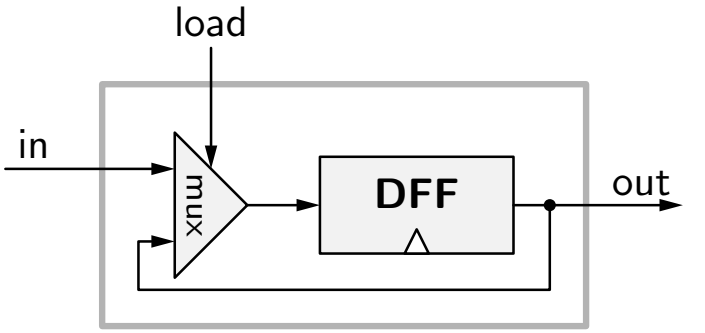
# Speicherzellen/Speicherregister

## Schnittstelle



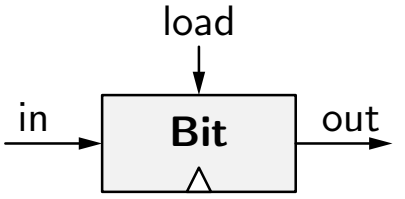

if load$(t-1)$
then out$(t) =$ in$(t-1)$
else out$(t) =$ out$(t-1)$

## Implementierung



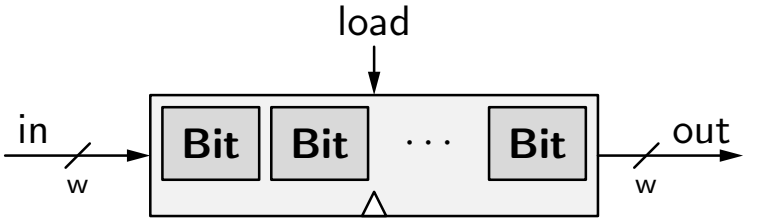

- Triviale Erweiterung: Kombination mehrerer 1-Bit-Zellen zu einem Register.
  Parameter: **Registerbreite** (durch Schrägstrich an Ein-/Ausgang angegeben)

## 1-Bit-Register




if load$(t-1)$
then out$(t) =$ in$(t-1)$
else out$(t) =$ out$(t-1)$

## w-Bit-Register



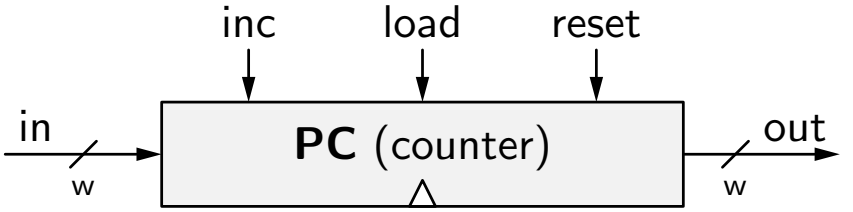

if load$(t-1)$
then out$(t) =$ in$(t-1)$
else out$(t) =$ out$(t-1)$

# Programmzähler

Gesucht: Speicherregister, das

- seinen Zustand auf einen gegebenen Wert setzen kann,
- seinen Zustand in jedem Taktzyklus um 1 erhöhen kann,
- seinen Zustand über mehrere Taktzyklen festhalten kann,
- seinen Zustand zurücksetzen kann (Zustand wird auf Wert 0 gesetzt).

**Schnittstelle**




$$
\begin{array}{lll}
\text{if } \mathsf{reset}(t-1) & \text{then} & \mathsf{out}(t) = 0 \\
\text{else if } \mathsf{load}(t-1) & \text{then} & \mathsf{out}(t) = \mathsf{in}(t-1) \\
\text{else if } \mathsf{inc}(t-1) & \text{then} & \mathsf{out}(t) = \mathsf{out}(t-1) + 1 \\
\text{else} & & \mathsf{out}(t) = \mathsf{out}(t-1)
\end{array}
$$

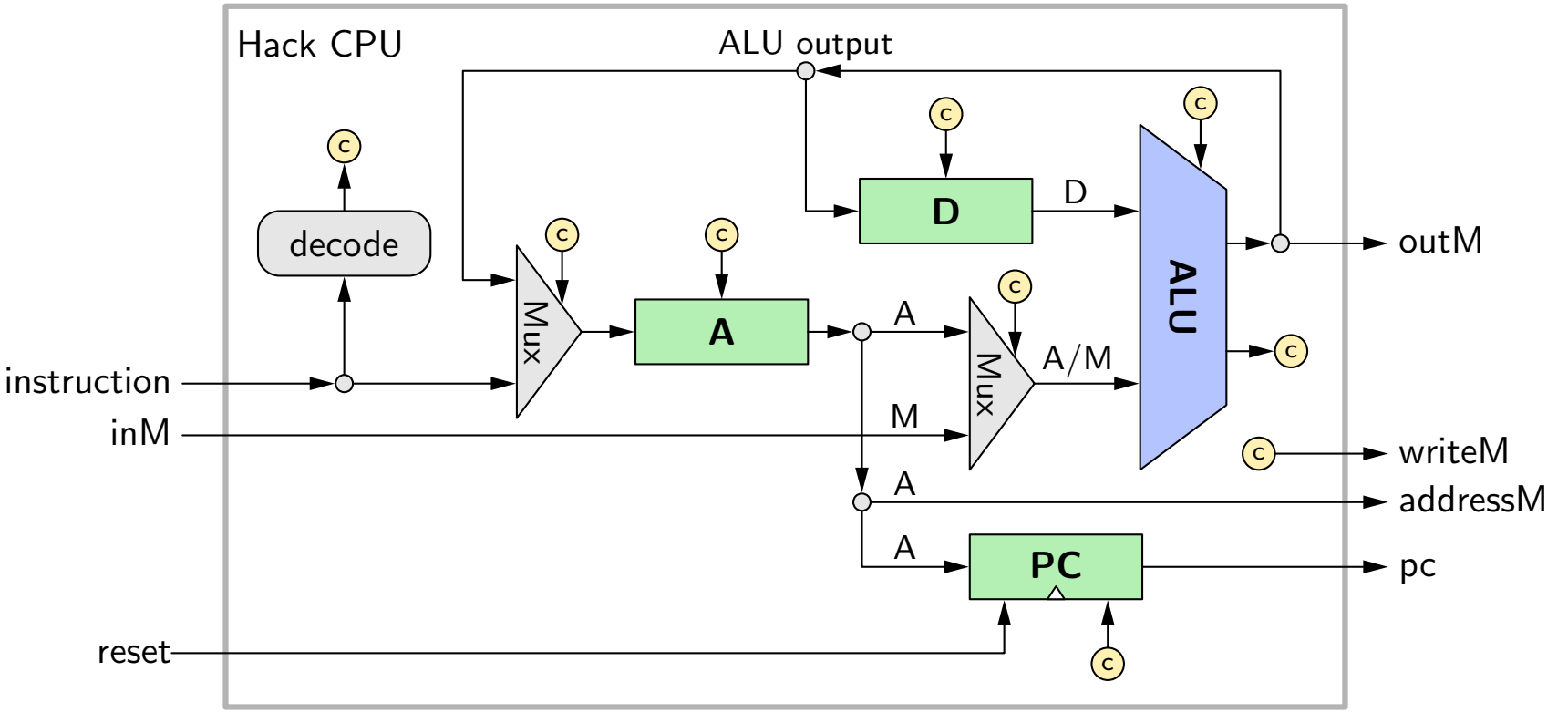
- Typische Verwendung: **Programmzähler** (program/instruction counter) d.h. ein Register, das angibt, welches die nächste auszuführende Anweisung ist.

  (Annahme: die auszuführenden Anweisungen sind numeriert, so daß der Registerinhalt als Anweisungsnummer interpretiert werden kann.)

- **Implementierung:** Speicherregister mit zusätzlicher kombinatorischer Logik.

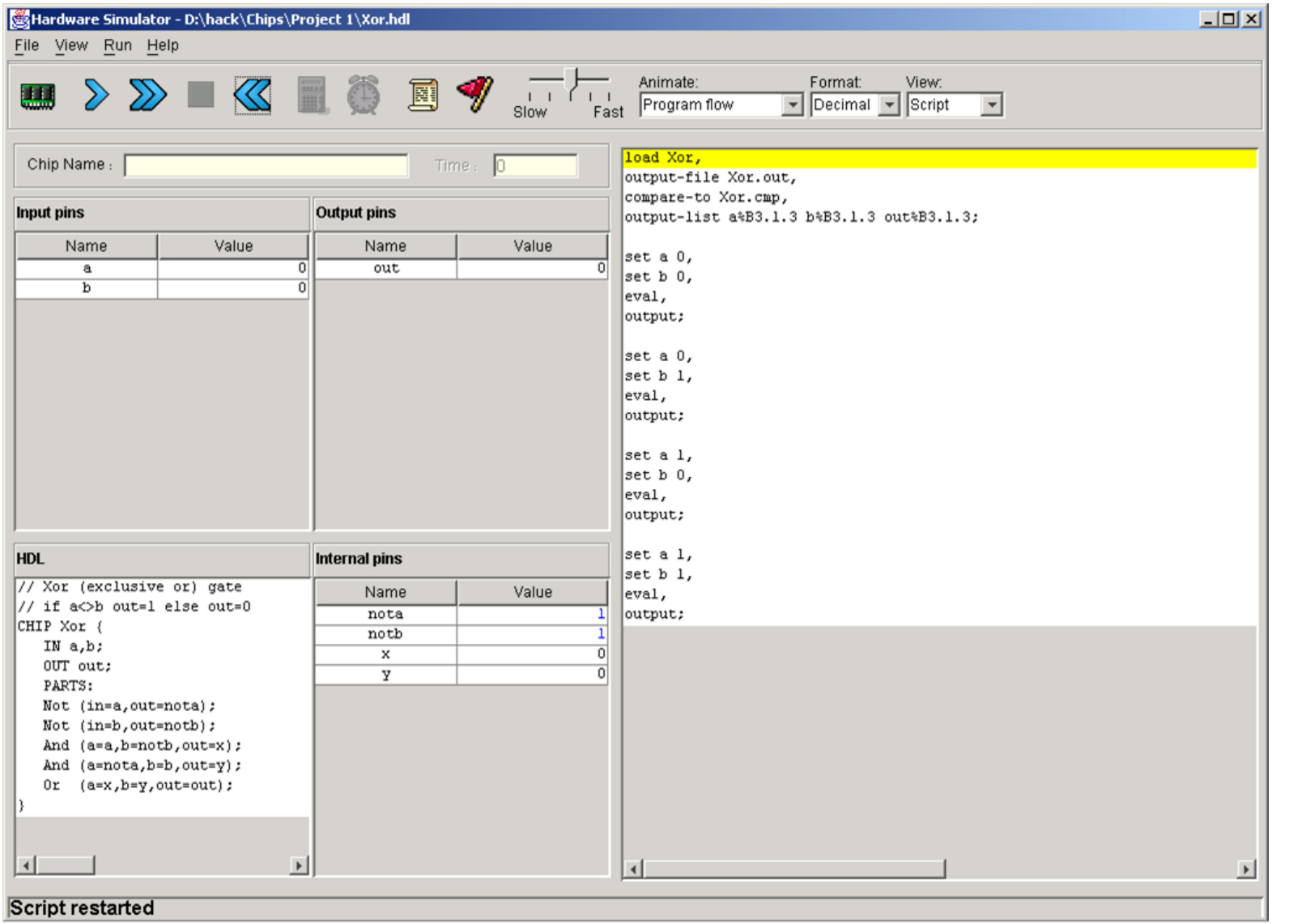
# Inhalt

# Hack-Architektur: Prozessor (CPU)




- Nur **Daten- und Adreßpfade** sind gezeigt (d.h., Verbindungen, die Daten und Adressen transportieren), nicht jedoch die **Steuerlogik**, mit Ausnahme der Ein- und Ausgaben von Steuerbits, die durch (c) markiert sind.

# Erinnerung: Hardware-Simulator

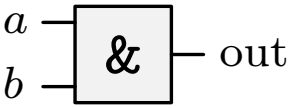



www.nand2tetris.org

# Erinnerung: Hardware-Beschreibungssprache

**And.cmp**

| *a* | *b* | out |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |




Ziel: Aufbau eines AND-Gatters
aus elementareren Gattern;
Test dieses Aufbaus.

**And.hdl**

```
CHIP And
{  IN a, b;
   OUT out;
   // implementation missing
}
```

**And.tst**

```
load And.hdl,
output-file And.out,
compare-to And.cmp,
output-list a b out;
set a 0, set b 0, eval, output;
set a 0, set b 1, eval, output;
set a 1, set b 0, eval, output;
set a 1, set b 1, eval, output;
```

Hardware-Beschreibung eines Bauteils durch drei Dateien:

- `*.cmp` Beschreibung der Schnittstelle durch Ein-/Ausgabetupel.
- `*.hdl` Beschreibung der Implementierung durch Gatterzusammenschaltung.
- `*.tst` Befehle zum Durchführen eines Funktionstests.

# Hardware-Simulation sequentieller Logik

- **The Elements of Computing Systems:**
  **Building a Modern Computer from First Principles**
  Noam Nisan & Shimon Schocken
  MIT Press, Cambridge, MA, USA 2008

- Für die Übungen zu HDL laden Sie die Software zu diesem Buch herunter:

  `http://nand2tetris.org/software.php`

- Relevante Themen aus der Hardware-Simulator-Anleitung:

  - **Getaktete Chips**

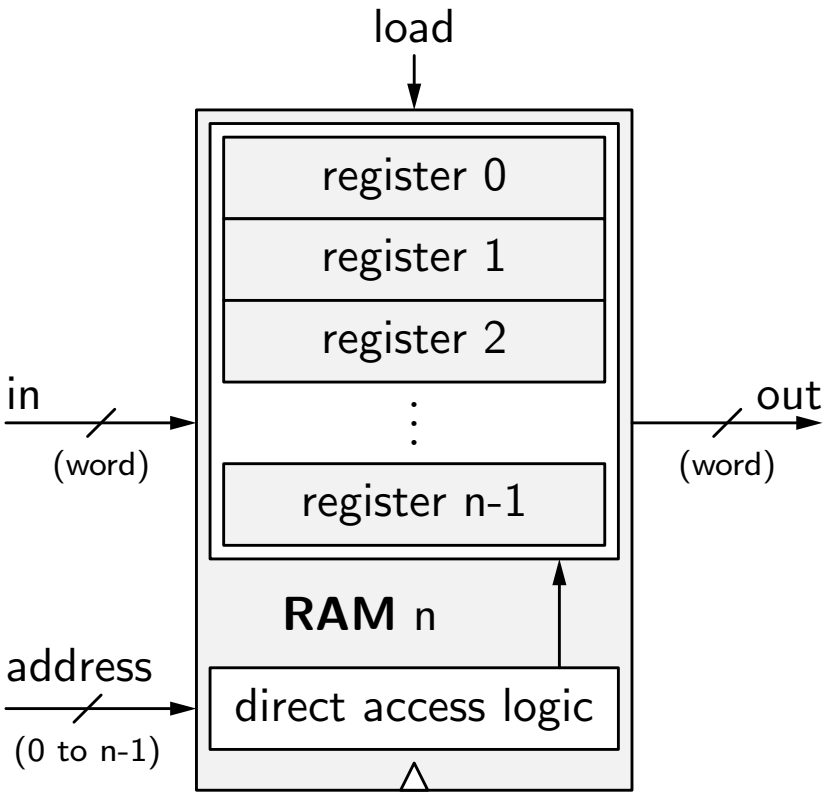
    Wenn ein getakteter Chip in den Simulator geladen wird,
    wird das Taktsymbol aktiviert, mit dem ein Benutzer den Takt steuern kann.

  - **Vorgegebene/elementare Chips**

    - haben eine Standard-HDL-Schnittstelle, aber eine Java-Implementierung,
    - stellen Dienste für die Verhaltenssimulation bereit,
    - können die Anzeigen der Benutzerschnittstelle beeinflussen.

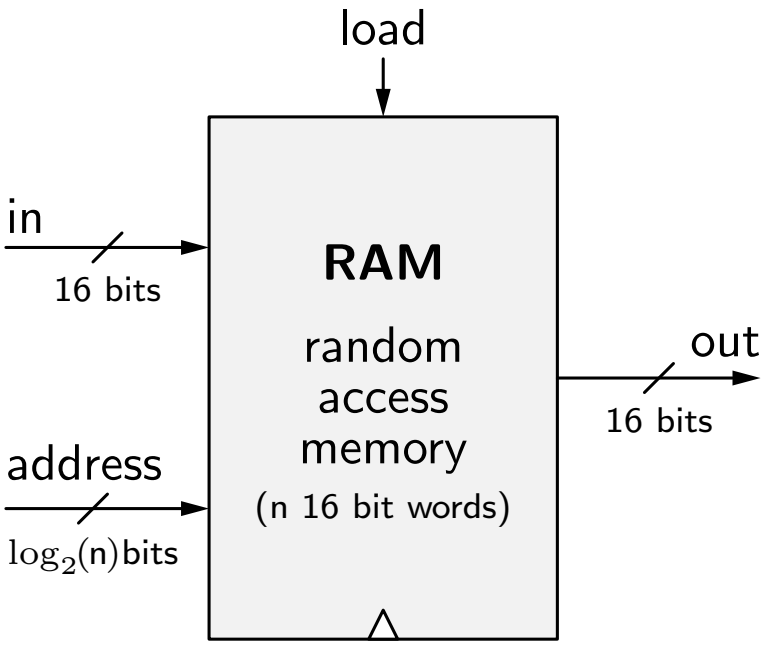
# Hauptspeicher: Freizugriffsspeicher

- Zum Ablegen von Daten und von Programmbefehlen wird **Speicher** gebraucht. Dieser wird zusammengefaßt und in eine Einheit außerhalb des Prozessors (CPU) ausgelagert.
- Solche Speichereinheiten bestehen, konzeptuell betrachtet, aus einer Reihe von Speicherregistern.
- Diese Register sind numeriert und können über ihre Nummer, **(Speicher-)Adresse** genannt, angesprochen werden.
- Die Leitungen für die Übertragung dieser Adresse heißen **Adreßbus**, die Datenleitungen **Datenbus**.




Bemerkung: Normaler Hauptspeicher ist nicht aus Flipflops aufgebaut, sondern aus speziellen Speicherzellen (später mehr dazu), doch vermeidet die Hack-Plattform diese Verkomplizierung. In der Hack-Plattform besteht aller Hauptspeicher aus normalen Speicherregistern.

# Hauptspeicher: Freizugriffsspeicher




Bemerkung: Normaler Hauptspeicher ist nicht aus Flipflops aufgebaut, sondern aus speziellen Speicherzellen (später mehr dazu), doch vermeidet die Hack-Plattform diese Verkomplizierung. In der Hack-Plattform besteht aller Hauptspeicher aus normalen Speicherregistern.

Hardware-Beschreibung:

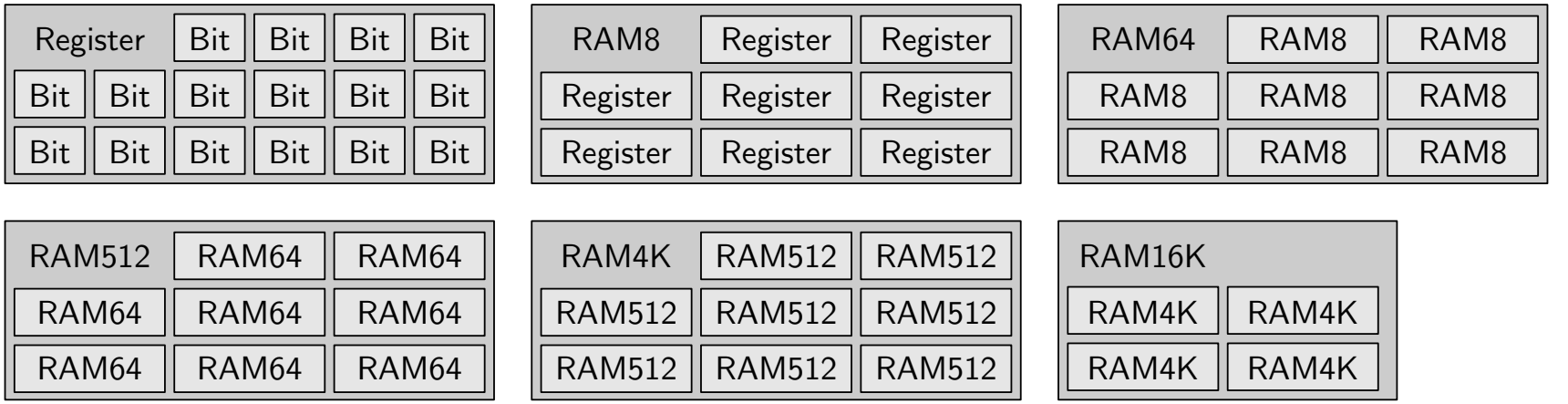
```
Chip name: RAMn // n and k listed below
Inputs:    in[16], address[k], load
Outputs:   out[16]
Function:  out[t] = RAM[address(t)](t)
           If load(t-1) then
           RAM[address(t-1)](t) = in(t-1)
Comment:   "=" is a 16-bit operation
```

Die spezifischen, für die Hack-Plattform benötigten Speicherchips sind ($n = 2^k$):

| Chip name | $n$ | $k$ |
|---|---|---|
| RAM8 | 8 | 3 |
| RAM64 | 64 | 6 |
| RAM512 | 512 | 9 |
| RAM4K | 4096 | 12 |
| RAM16K | 16384 | 14 |

# Hauptspeicher: Freizugriffsspeicher

- Die Hack-Architektur definiert den Hauptspeicher durch rekursiven Aufstieg:




- Der Grund für diese rekursive Struktur ist nicht prinzipiell, sondern liegt nur darin, daß die Hardware-Beschreibungssprache (hardware description language, HDL) keine Schleifen kennt.

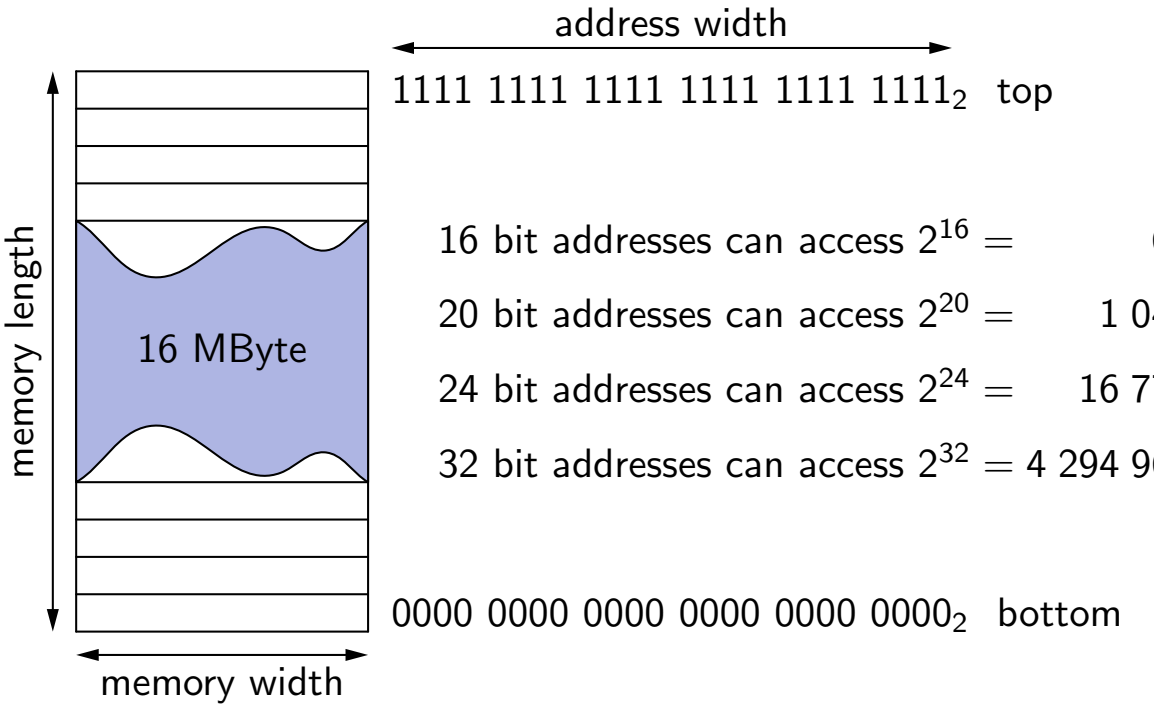
- Eine direkte Definition des RAM16K-Chips als 16384 16-Bit-Register ist daher zwar prinzipiell möglich, aber sehr unhandlich.

- Durch die rekursive Definition halten sich die Chip-Beschreibungen in Grenzen.

# Speichergrößen und Addreßbreiten

- Die **Breite des Adreßbusses** bestimmt,
  wie viel Hauptspeicher ein Prozessor adressieren kann:

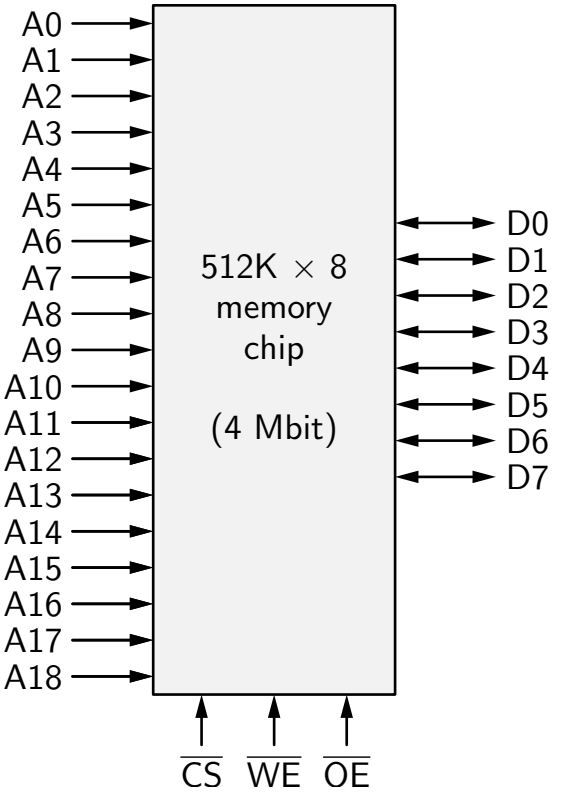
  Jede Adreßleitung stellt ein Bit der Speicheradresse dar.

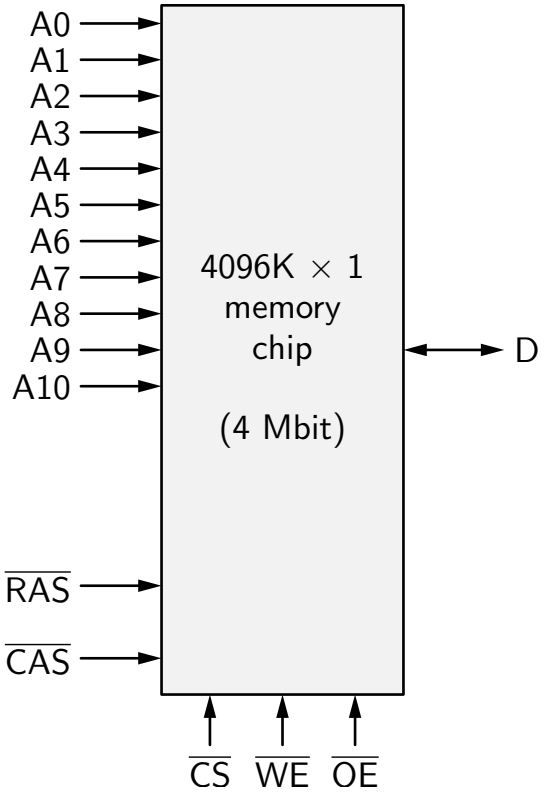



- Jede Speicherstelle enthält gewöhnlich 8 Bit (ein **Byte**),
  auch wenn der Datenbus heutzutage meist wesentlich breiter ist (z.B. 64 Bit).

# Speicher: Addressierung

- Speicher kann sehr verschieden organisiert sein:





SRAM 512K × 8


DRAM 4096K × 1

CS: Chip Select, WE: Write Enable, OE: Output Enable,
RAS: Row Address Select (zuerst), CAS: Column Address Select (anschließend).

- Insbesondere kann die Breite der Speicherregister variieren, was dann unterschiedliche Zugriffslogiken erfordert.

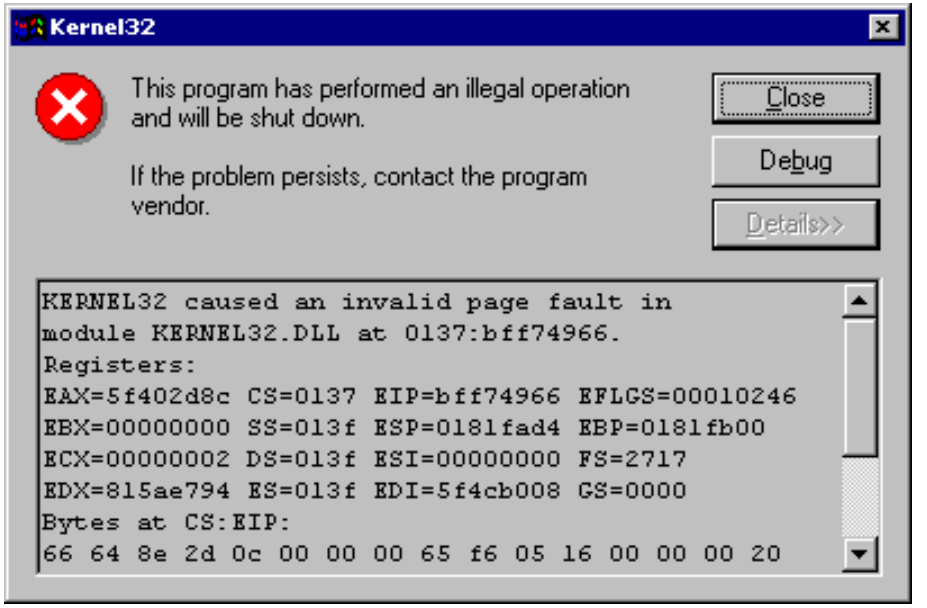
# Hexadezimale Speicheradressen

- Speziell bei großen Adressen (z.B. 32 Bit oder heute sogar 64 Bit) ist die Angabe einer (Speicher-)Adresse als Binärzahl unhandlich (jedenfalls für Menschen).
  Daher werden 4 binäre Ziffern zu einer hexadezimalen Ziffer zusammengefaßt.
- Beispiel: (32-Bit-Adresse in Binär- und Hexadezimaldarstellung)

| 1010 | 0011 | 1101 | 0010 | 0101 | 1111 | 1110 | 0001 |
|---|---|---|---|---|---|---|---|
| A | 3 | D | 2 | 5 | F | E | 1 |

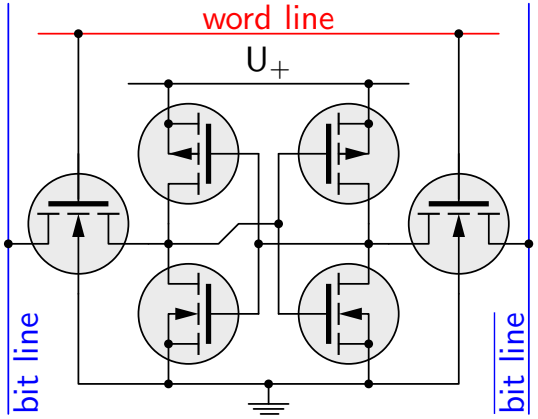
| Dec | Bin | Hex |
|---|---|---|
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0011 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |

| Dec | Bin | Hex |
|---|---|---|
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |



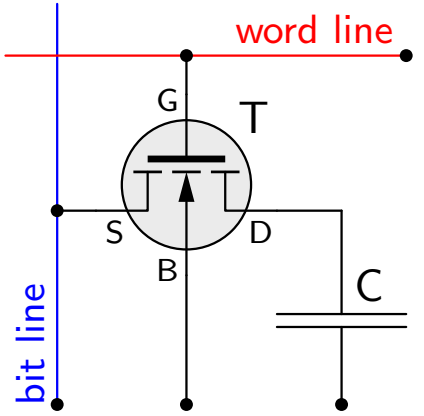
# Speichertypen (für einzelne Bits)

**Statischer Speicherbaustein** (static random access memory, SRAM)



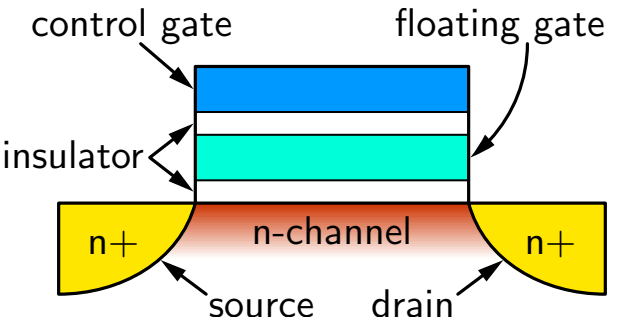

- 6 Transistoren je Speicherzelle
- sehr schnelles Auslesen
- keine Auffrischung benötigt

**Dynamischer Speicherbaustein** (dynamic random access memory, DRAM)



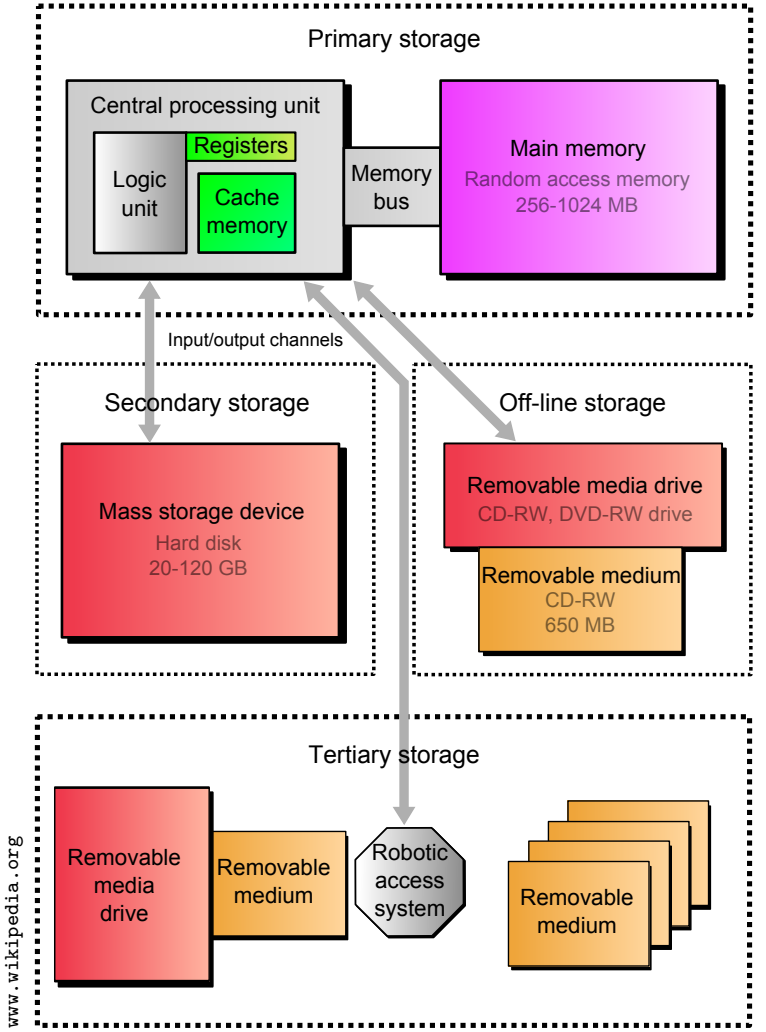

- ein Transistor und ein Kondensator je Speicherzelle
- schnelles Auslesen
- benötigt ständige Auffrischung

**Floating-Gate-Transistor** (floating gate transistor, FGMOSFET)




- ein Transistor je Speicherzelle
- schnelles Auslesen, langsames Löschen (blockweise)
- sog. Flash Memory

# Speichertypen / Speicherhierarchie




- Je größer eine Speichereinheit, desto teurer ihre Herstellung.
- Je schneller ein Speichertyp, desto teurer seine Herstellung.
- Deshalb: **Speicherhierarchie** mit
  - wenig sehr schnellem Speicher (z.B. Prozessorcache) aus statischen Speicherbausteinen,
  - mehr etwas langsamerem Speicher (z.B. Hauptspeicher) aus dynamischen Speicherbausteinen,
  - viel deutlich langsamerem Speicher (z.B. Festplatten, externe Medien) für langfristige Speicherung.

# Zusammenfassung: Sequentielle Logik

- **Sequentielle Logik**
  - Kombinatorische und sequentielle Logik
  - Schaltnetze und Schaltwerke
  - Rückkopplung (feedback) und Taktsignal (clock)
- **Bistabile Kippstufen („Flipflops“)**
  - SR-Riegel (SR latch) als Grundbaustein
  - D-, E-, T- und JK-Riegel (D, E, T, JK latch)
  - Taktpegel- und Taktflankensteuerung (latches vs. flip flops)
  - Master-Slave-Riegel und -Flipflops
- **Register, Zähler und Speicher**
  - Schieberegister, Parallel-Seriell-Wandler, Zähler
  - Speicherregister, Hauptspeicherorganisation