Design digitaler Schaltkreise

- Komplexere CMOS Digitalzellen
- NAND, NOR…
- Multiplexer, Dekoder…
- Latches, Flip-Flops

- Kombinatorisch gibt es $2^4 = 16$ Booleschen Funktionen von zwei Variablen
- Die Länge der Ergebnistabelle ist 4 und für jede Zeile haben wir zwei Möglichkeiten.
- Die wichtigsten Booleschen Funktionen mit zwei Variablen sind NAND, NOR, EXNOR (Gleichwertigkeit, Äquivalenz).
- Da es Inverter gibt, können wir aus NAND, NOR und EXNOR AND, OR und die EXOR bauen

NAND

a	b	y
0	0	1
0	1	1
1	0	1
1	1	0

NOR

a	b	y
0	0	1
0	1	0
1	0	0
1	1	0

EXNOR

a	b	y
0	0	1
0	1	0
1	0	0
1	1	1

- Warum sind nur drei (bzw. sechs) Funktionen genug?
- 8 Booleschen Funktionen kann man aus16 durch Negation bekommen - wie AND aus NAND.
- Zwei Funktionen (von 8 – wir betrachten z.B. nur diese die 1 in der ersten Zeile haben) sind eigentlich keine Funktionen von zwei sondern nur einer Variable.
- Eine Funktion von 8 ist Konstante.

!a

a	b	y
0	0	1
0	1	1
1	0	0
1	1	0

!b

a	b	y
0	0	1
0	1	0
1	0	1
1	1	0

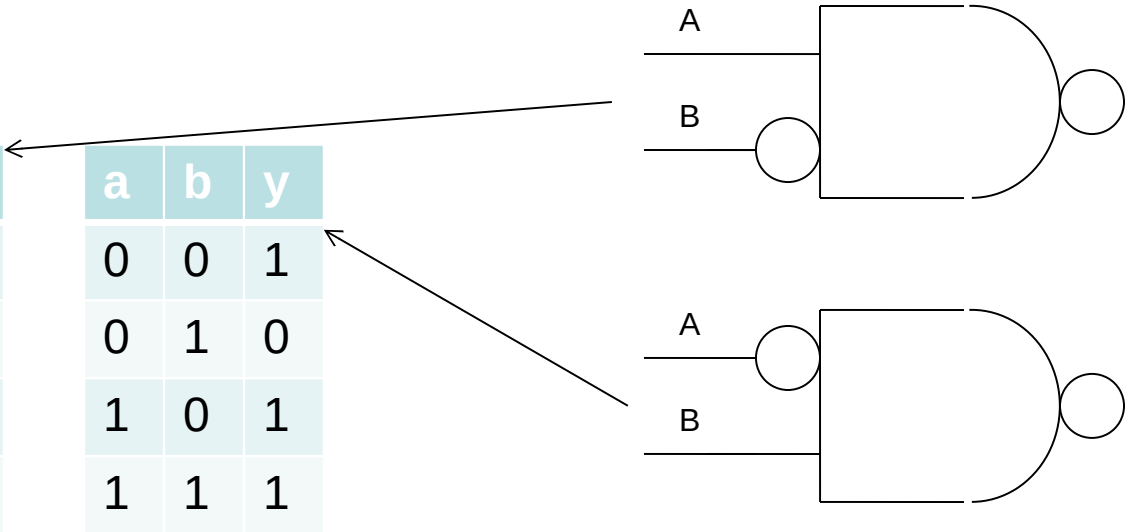
1

a	b	y
0	0	1
0	1	1
1	0	1
1	1	1

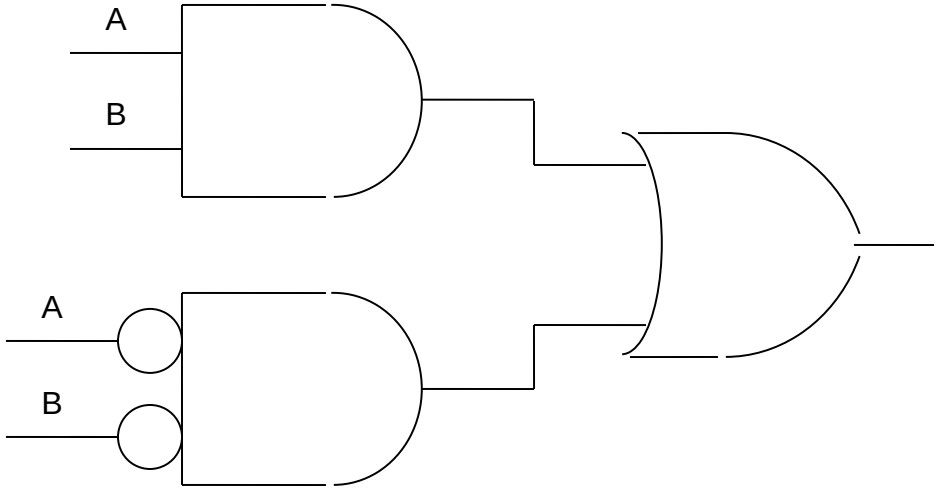
- Zwei Funktionen können mit NAND mit invertierten Eingängen realisiert werden.
- -> NAND, NOR, EXNOR und Inverter sind ausreichend

a	b	y
0	0	1
0	1	1
1	0	0
1	1	1

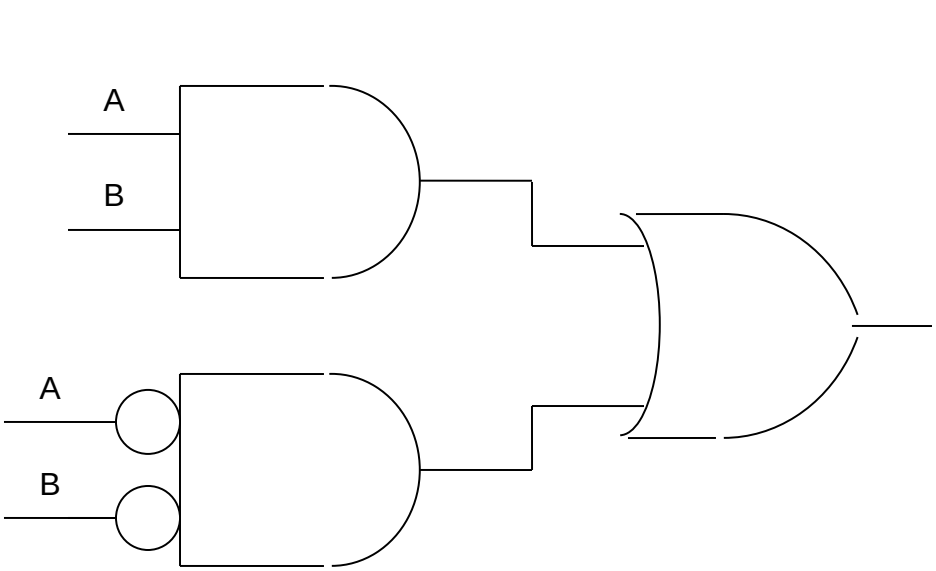
a	b	y
0	0	1
0	1	0
1	0	1
1	1	1

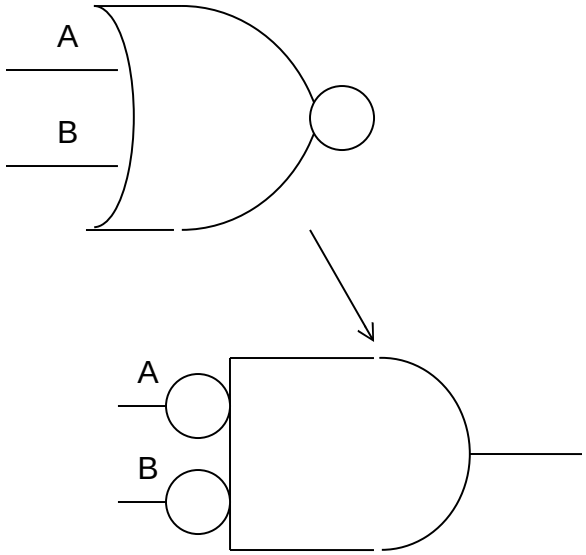


- EXNOR kann man mit (N)AND, (N)OR und Inverter realisieren
- NOR kann man in NAND umwandeln.
- Streng genommen wäre z.B. NAND genug.

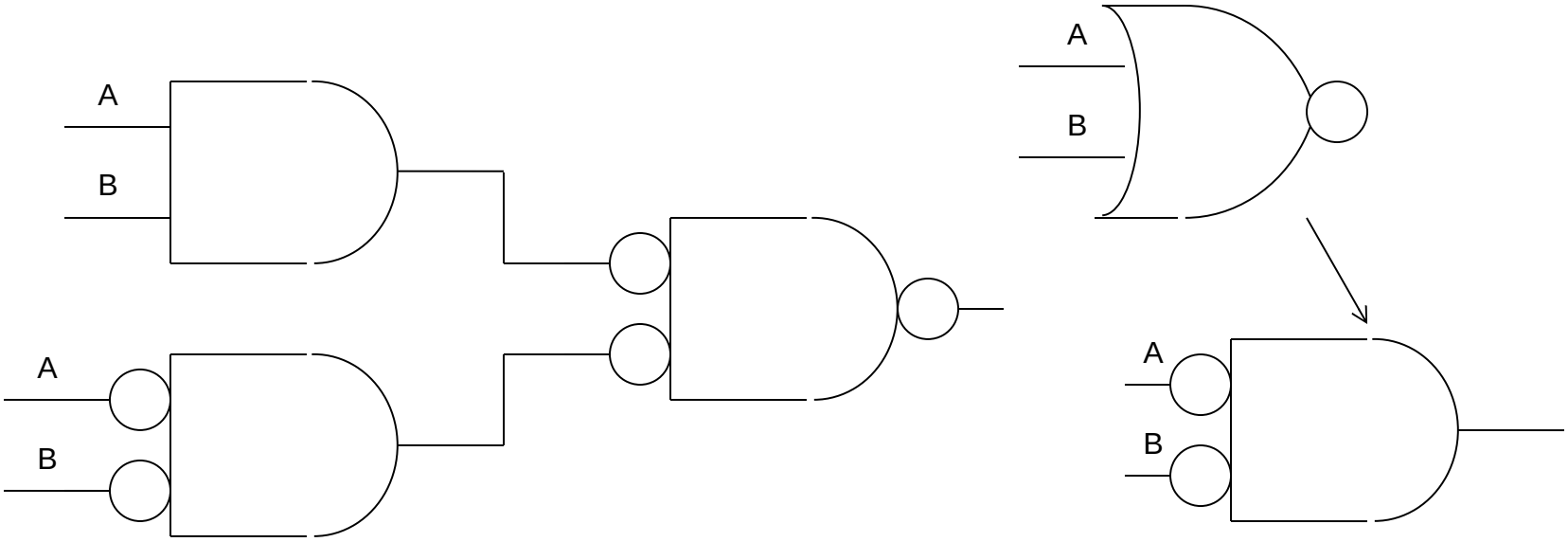


- EXNOR kann man mit (N)AND, (N)OR und Inverter realisieren
- NOR kann man in NAND umwandeln.
- Streng genommen wäre z.B. NAND genug.

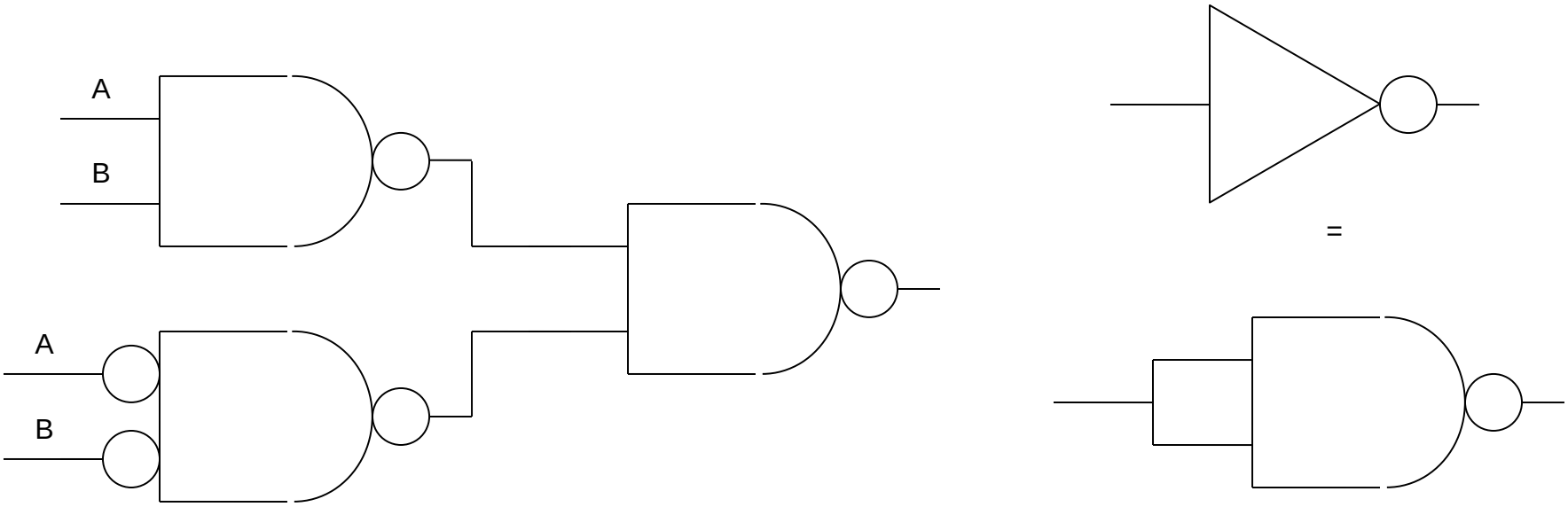




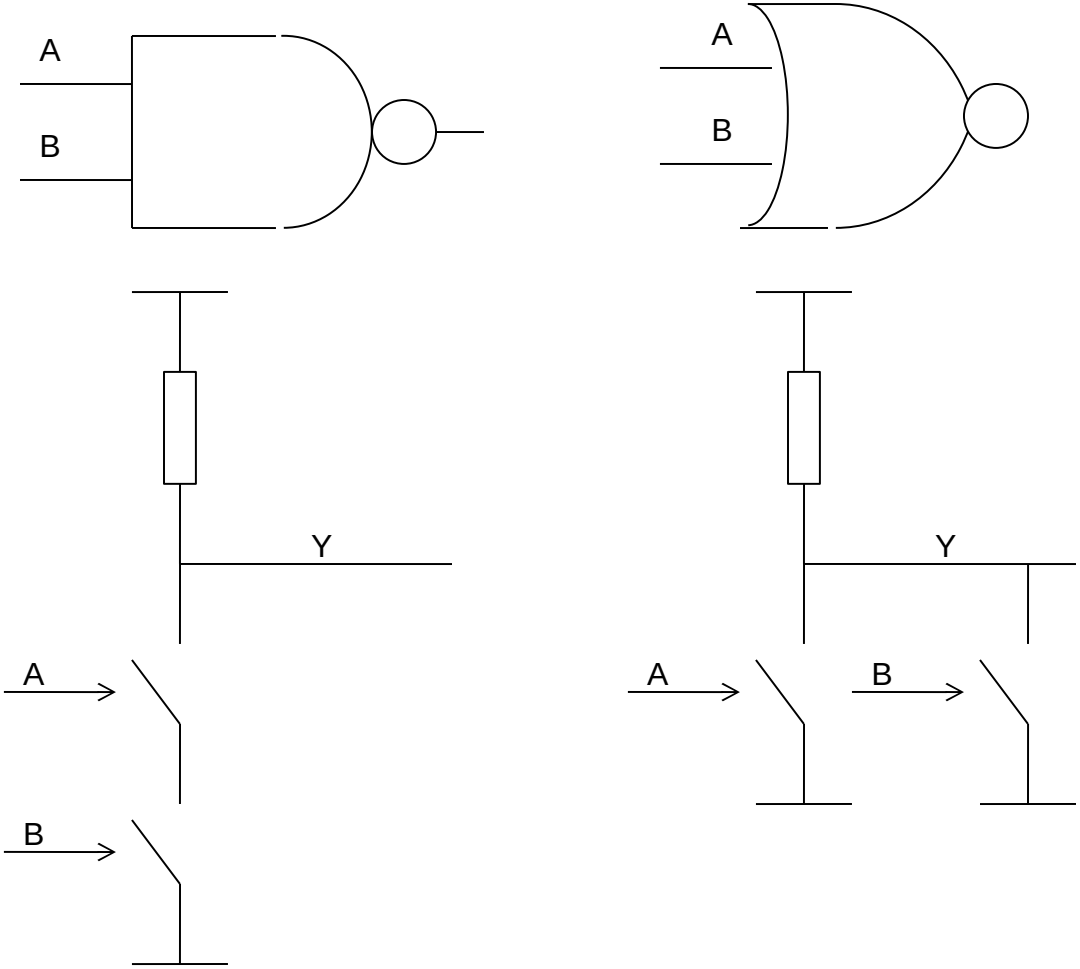
- EXNOR kann man mit (N)AND, (N)OR und Inverter realisieren
- NOR kann man in NAND umwandeln.
- Streng genommen wäre z.B. NAND genug.



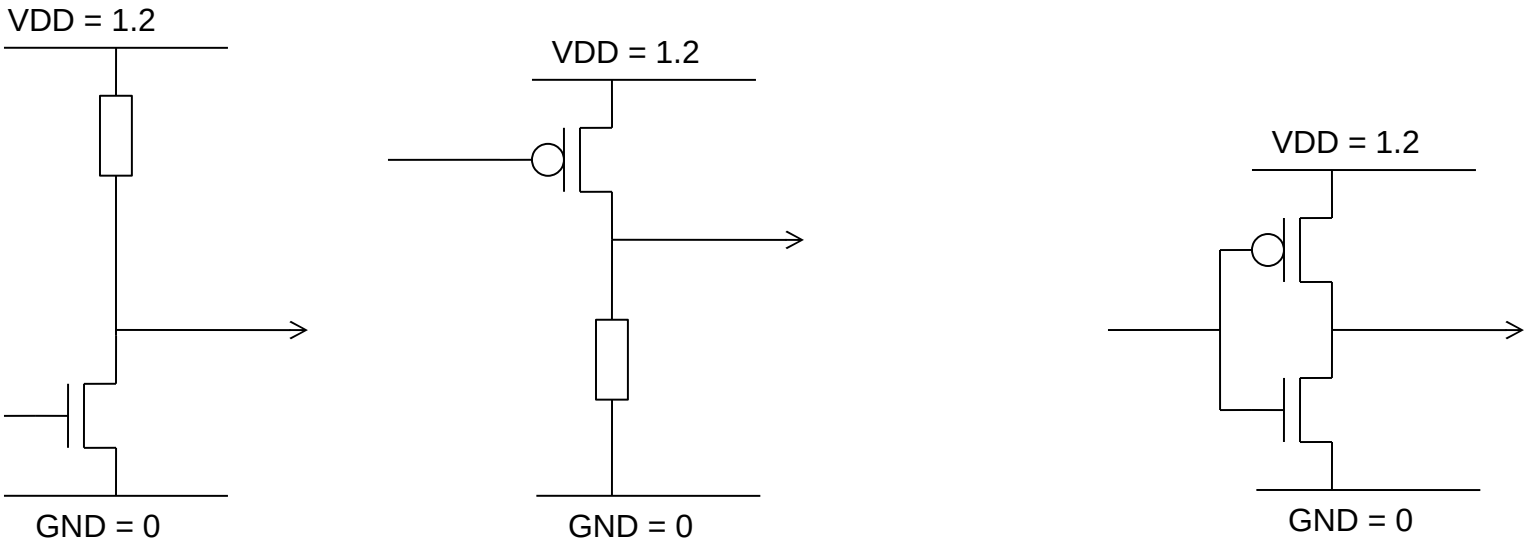
- EXNOR kann man mit (N)AND, (N)OR und Inverter realisieren
- NOR kann man in NAND umwandeln.
- Streng genommen wäre z.B. NAND genug.



- NAND und NOR als Schalter-Widerstand Logik



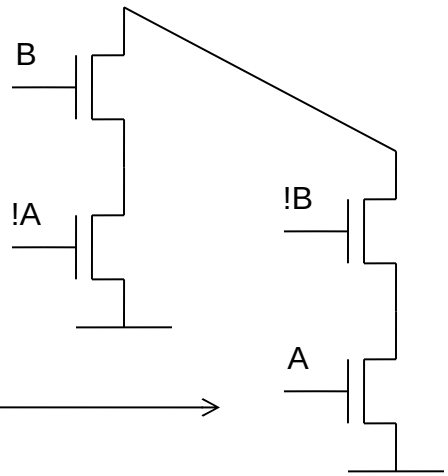
- Inverter als RTL Logik
- NMOS und Pullup oder mit PMOS und Pulldown Widerstand
- -> CMOS Inverter.
- Vorteile sind kein DC Strom und ein kleines Layout.

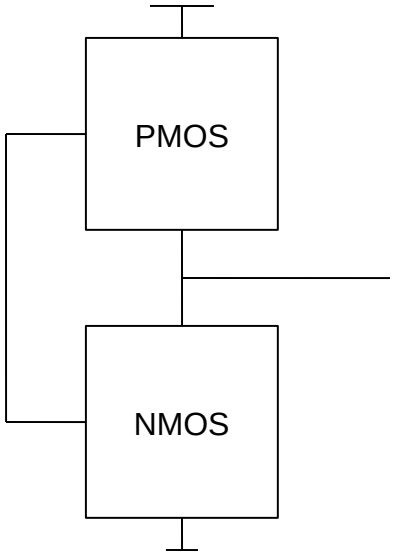


- NAND, NOR und co. als CMOS
- Wie wird ein CMOS Gate gemacht?
- Wenn NMOS Teil leitet, soll PMOS Teil nicht leiten, und umgekehrt
- Kein Kurzschluss VDD-GND, oder floating-Ausgang
- Jede Zeile mit dem Ergebnis 0 -> Serienschaltung von zwei (oder mehreren) NMOS Transistoren die nur für die Eingangswerte dieser Zeile leiten
- Man muss alle Eingänge = null invertieren.
- Das ganze NMOS Netzwerk ist die Parallelschaltung aller Reihenschaltungen, die Zeilen = 0 entsprechen.

EXNOR

a	b	y
0	0	1
0	1	0
1	0	0
1	1	1

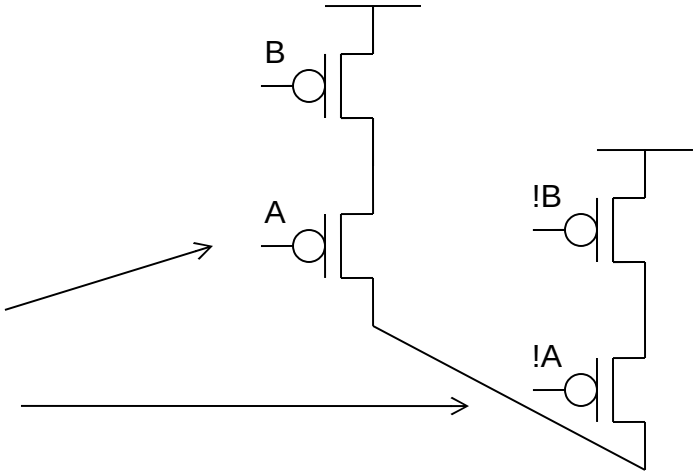




- PMOS Teil macht man dual
- Beachten wir, dass PMOS für niedriges Gate-Potential leitet
- Man muss alle Eingänge = eins invertieren.

EXNOR

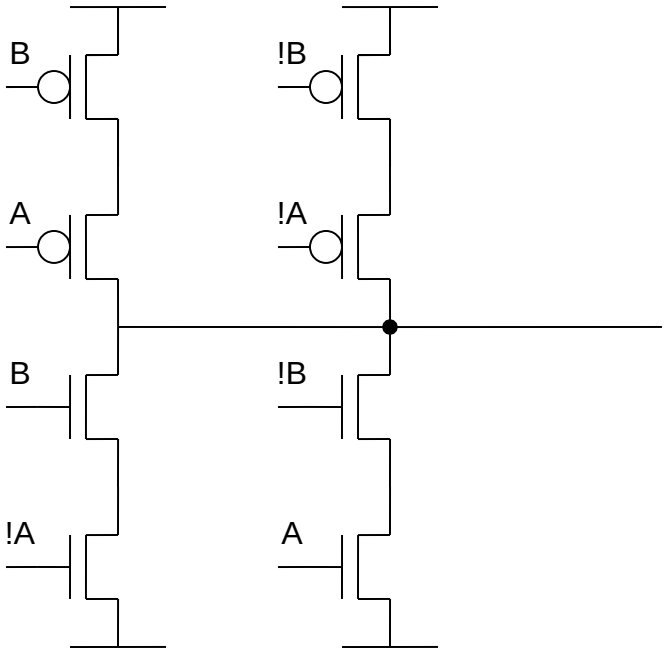
a	b	y
0	0	1
0	1	0
1	0	0
1	1	1



- EXNOR

EXNOR

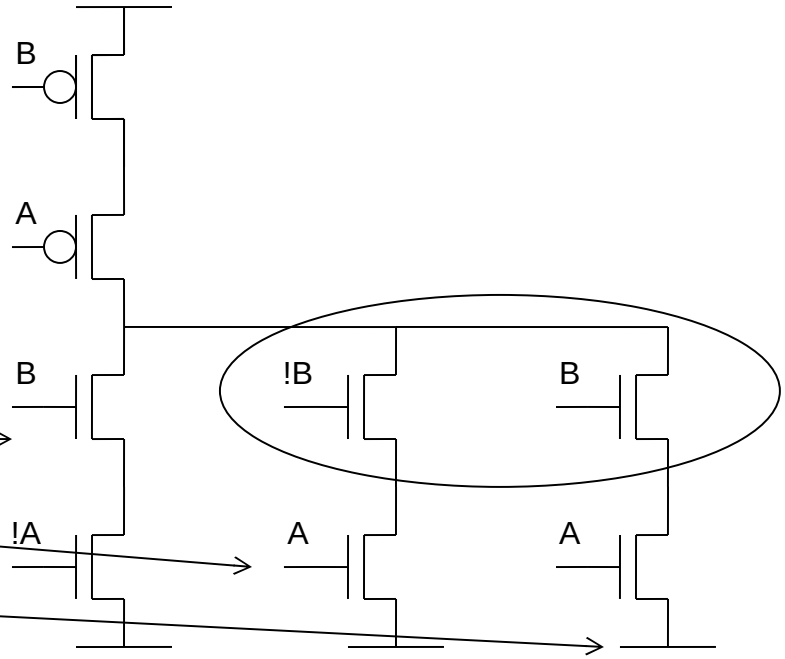
a	b	y
0	0	1
0	1	0
1	0	0
1	1	1



- Oft kann man die logische Funktion vereinfachen

NOR

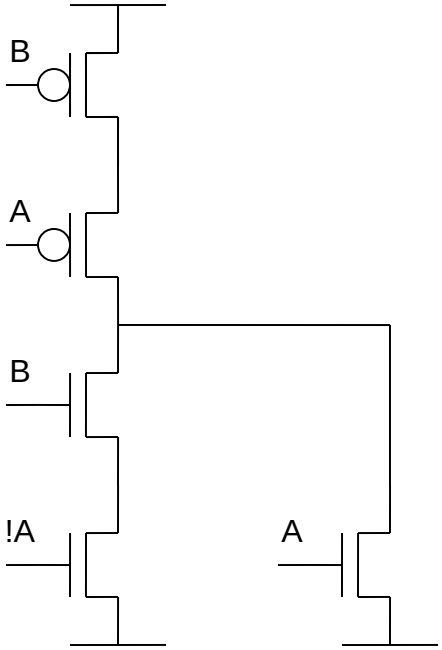
a	b	y
0	0	1
0	1	0
1	0	0
1	1	0



- Oft kann man die logische Funktion vereinfachen

NOR

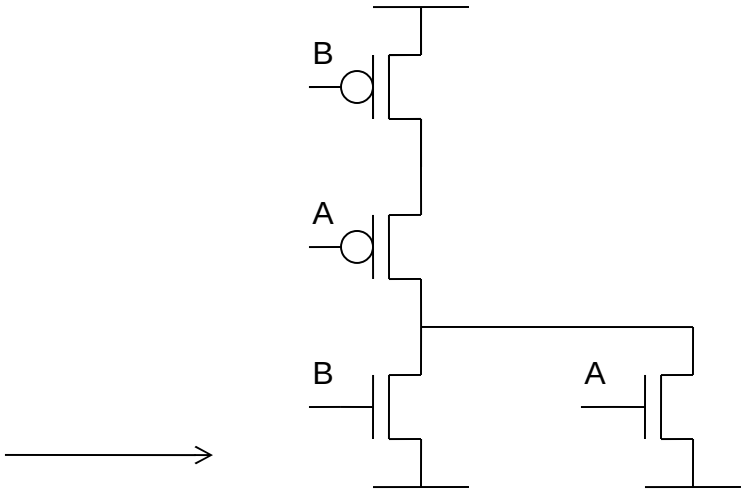
a	b	y
0	0	1
0	1	0
1	0	0
1	1	0



- Oft kann man die logische Funktion vereinfachen
- CMOS NOR
- PMOS Netzwerk leitet für die Eingangskombination 00 – Reihenschaltung
- NMOS Netzwerk leitet immer außer für 00 – Parallelschaltung.

NOR

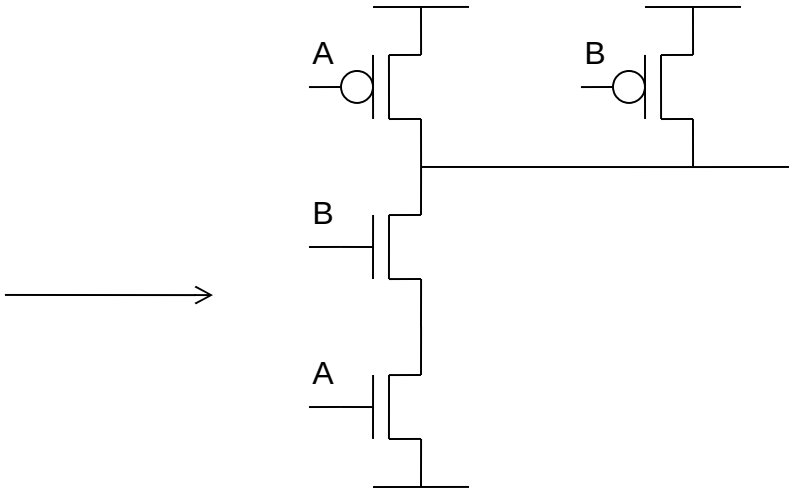
a	b	y
0	0	1
0	1	0
1	0	0
1	1	0



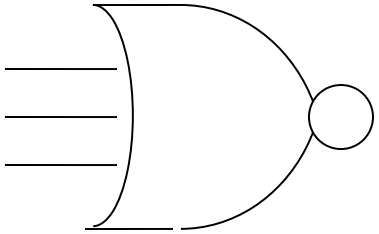
- CMOS NAND

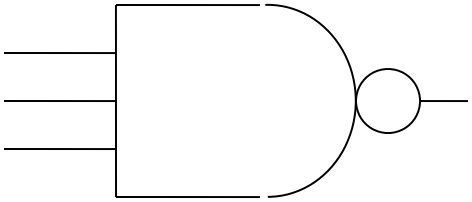
NAND

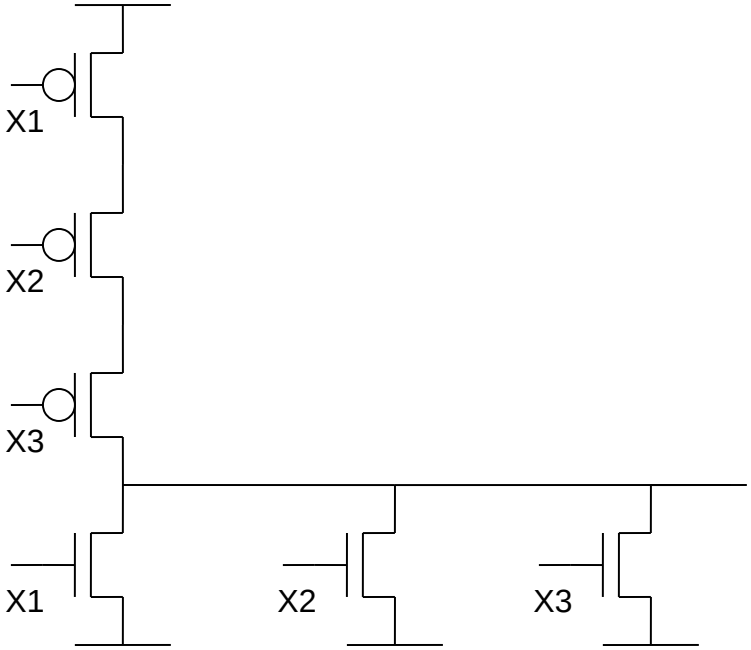
a	b	y
0	0	1
0	1	1
1	0	1
1	1	0

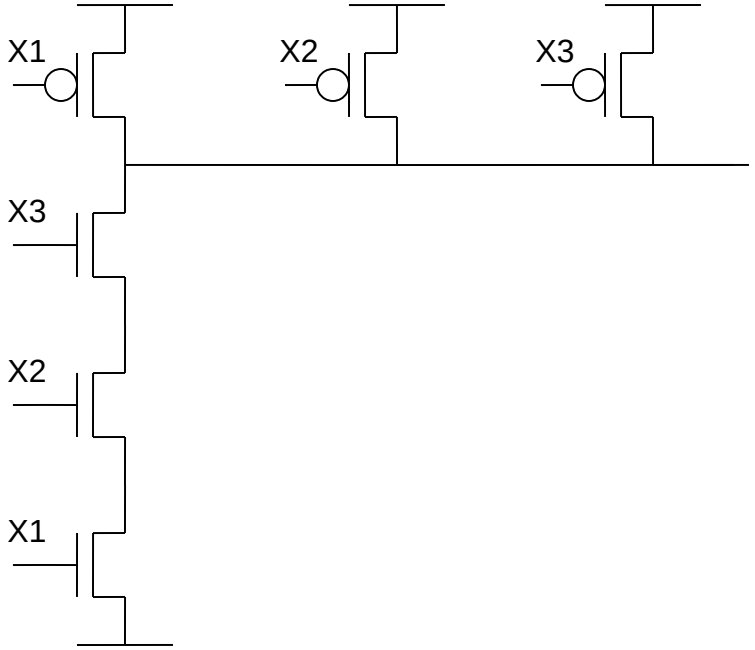


- Es ist leicht die NAND und NOR auf mehr als 2 Eingänge zu erweitern







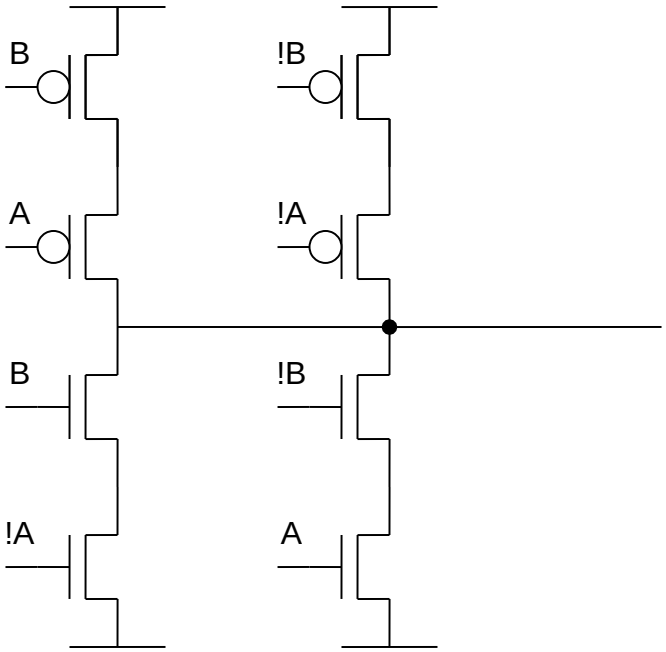


- Einige Regeln
- NMOS Teil leitet für die Zeilen mit null-Ergebnis
- PMOS Teil leitet für die Zeilen mit eins-Ergebnis
- PMOS und NMOS Teile dürfen nie gleichzeitig leiten, sonst hätten wir einen großen Querstrom und der Ausgang wäre undefiniert
- PMOS und NMOS Teil sollen auch nie gleichzeitig offene Verbindungen sein. In dem Fall wäre der Ausgang von den Versorgungslinien getrennt. Der logische Wert wäre undefiniert
- Gate mit offenem Ausgang befindet sich im hochohmigen Zustand

- EXNOR

Variante 1

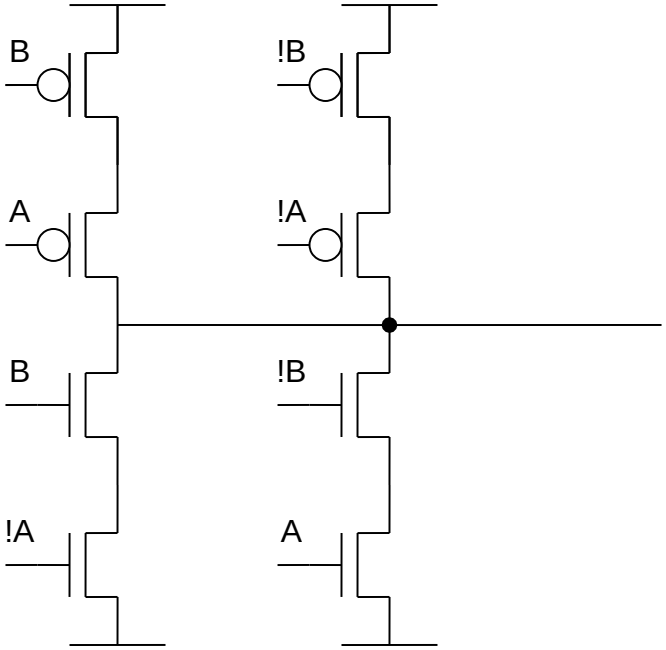
8 Transistoren



- EXNOR
- Disjunktive Normalform
- EXOR = (!A & !B) | (A & B)

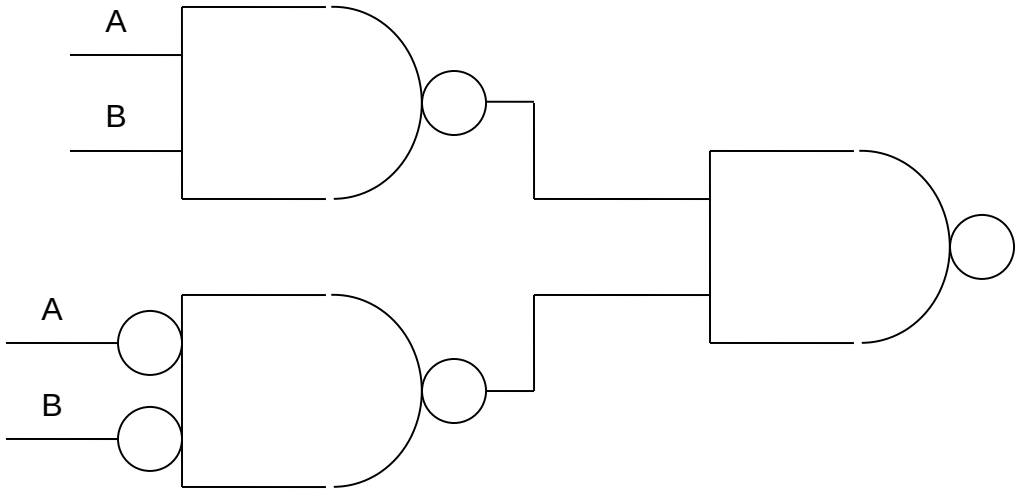
Variante 1

8 Transistoren

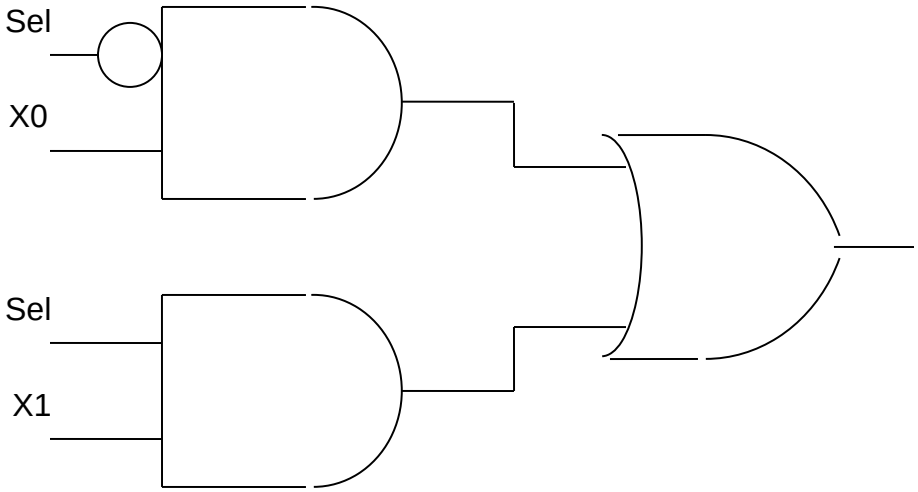


Variante 2

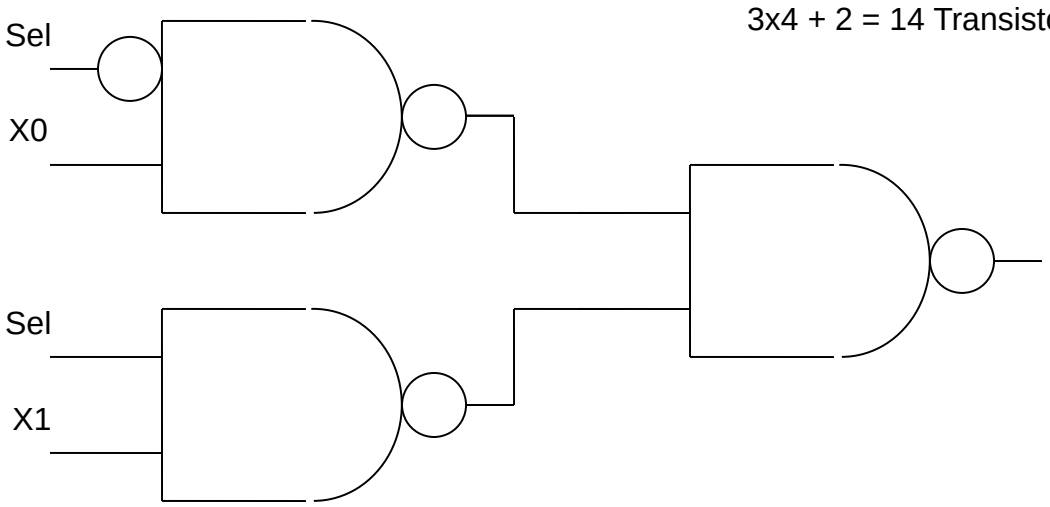
4 (INVs) + 12 (NANDs) Transistoren



- Noch komplexere Gates
- Gatter mit 3 Eingängen
- Der wichtigste und vielseitigste Bauteil in Digitaltechnik ist der Multiplexer
- Je nachdem ob der Select-Eingang null oder eins ist, ist der Ausgang X0 oder X1
- Im Verilog Code: Y = sel ? X1 : X0
- Disjunktive Normalform: Y = !sel & X0 | sel & X1

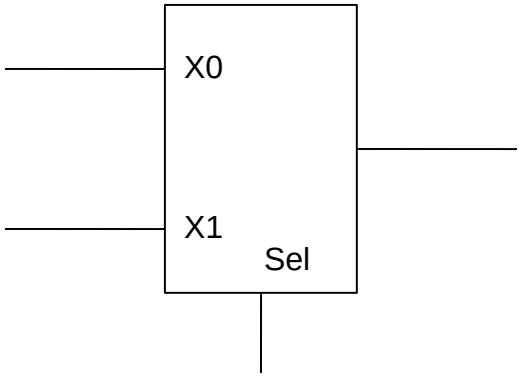


- Noch komplexere Gates
- Gatter mit 3 Eingängen
- Der wichtigste und vielseitigste Bauteil in Digitaltechnik ist der Multiplexer
- Je nachdem ob der Select-Eingang null oder eins ist, ist der Ausgang X0 oder X1
- Im Verilog Code: Y = sel ? X1 : X0
- Disjunktive Normalform: Y = !sel & X0 | sel & X1

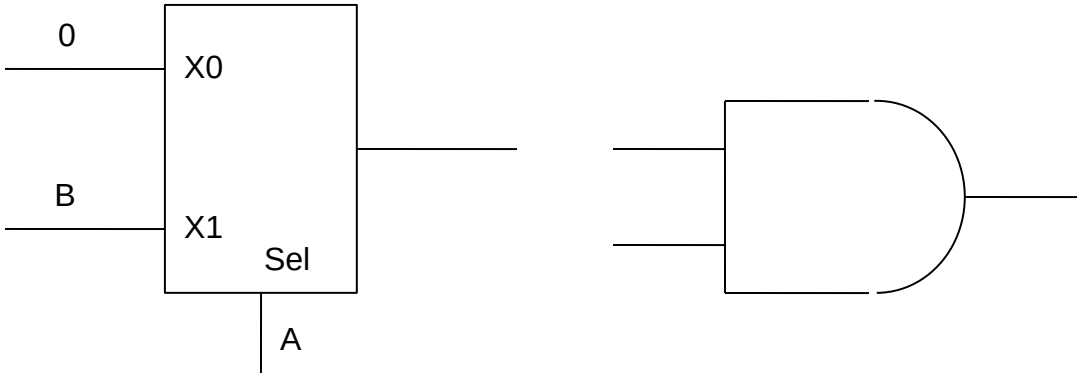


3x4 + 2 = 14 Transistoren

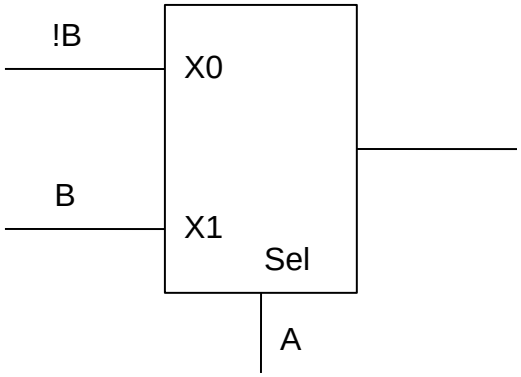
- Warum ist ein Multiplexer so wichtig?
- Jede logische Funktion kann mit Multiplexern, Invertern und logischen Konstanten realisiert werden

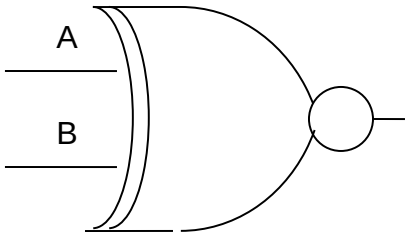


- Beispiel: AND
- AND ist null wenn die Variable A null ist, unabhängig von B
- -> A an Select anschließen, an Eingang X0 schließen wir die logische 0
- Wenn A eins ist (Select = 1), hängt das Ergebnis von Variable B ab
- -> Variable B wird an Eingang X1 angeschlossen.
- AND = A ? B : 0

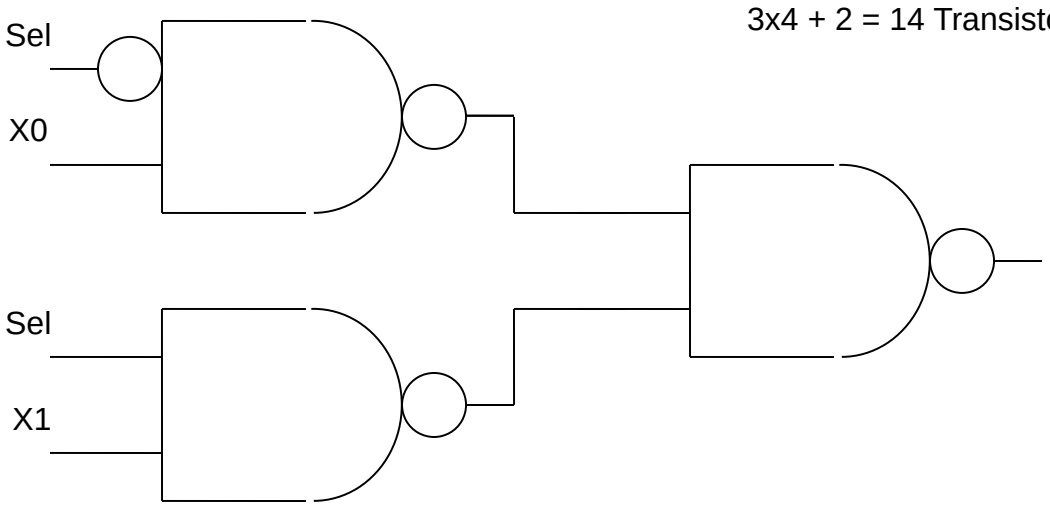


- Beispiel EXNOR
- EXNOR = A ? B : !B



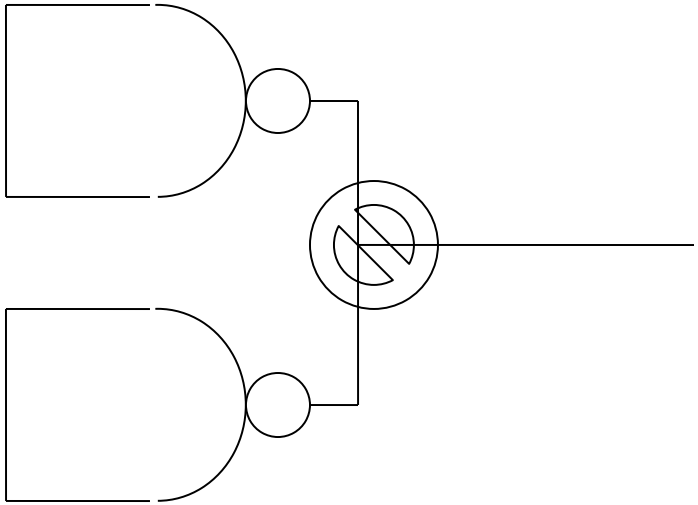


- Multiplexer kann auch einfacher realisiert werden.

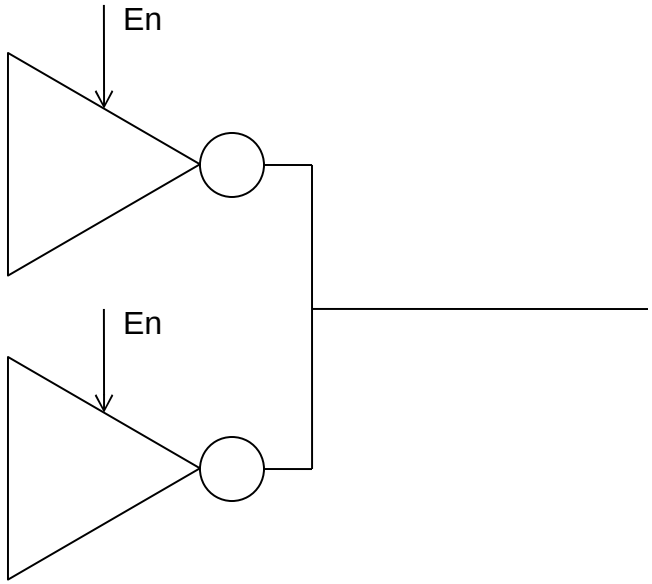


3x4 + 2 = 14 Transistoren

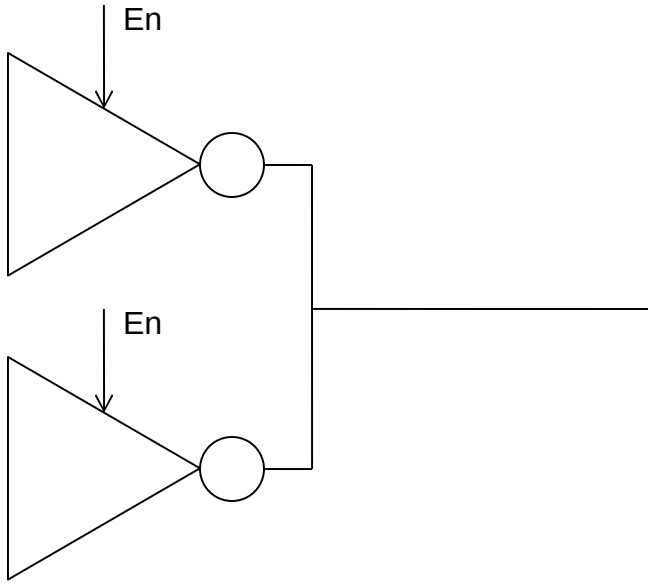
- Multiplexer kann auch einfacher realisiert werden.



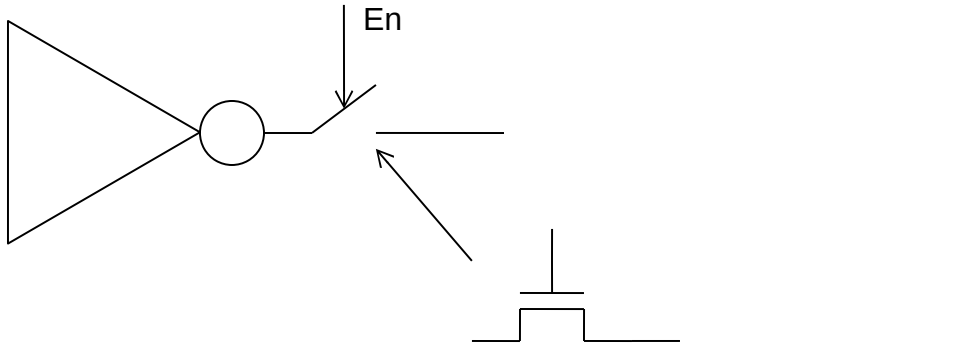
- Wir können die Gatter so erweitern, dass sie sich im hochohmigen Zustand befinden können
- -> Gated Inverter
- Wenn der Enable Eingang eins ist, funktioniert der Inverter wie ein gewöhnlicher Inverter



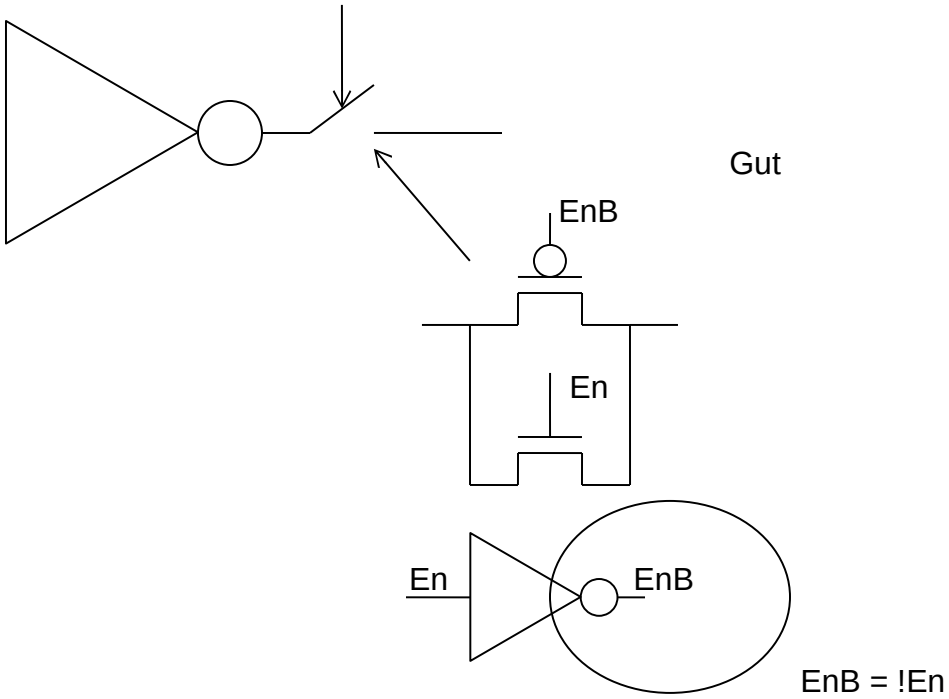
- Mit Enable = null, ist der Ausgang von VDD und GND getrennt, der Ausgang „schwebt“ (float) im hochohmigen (high impedance) Zustand



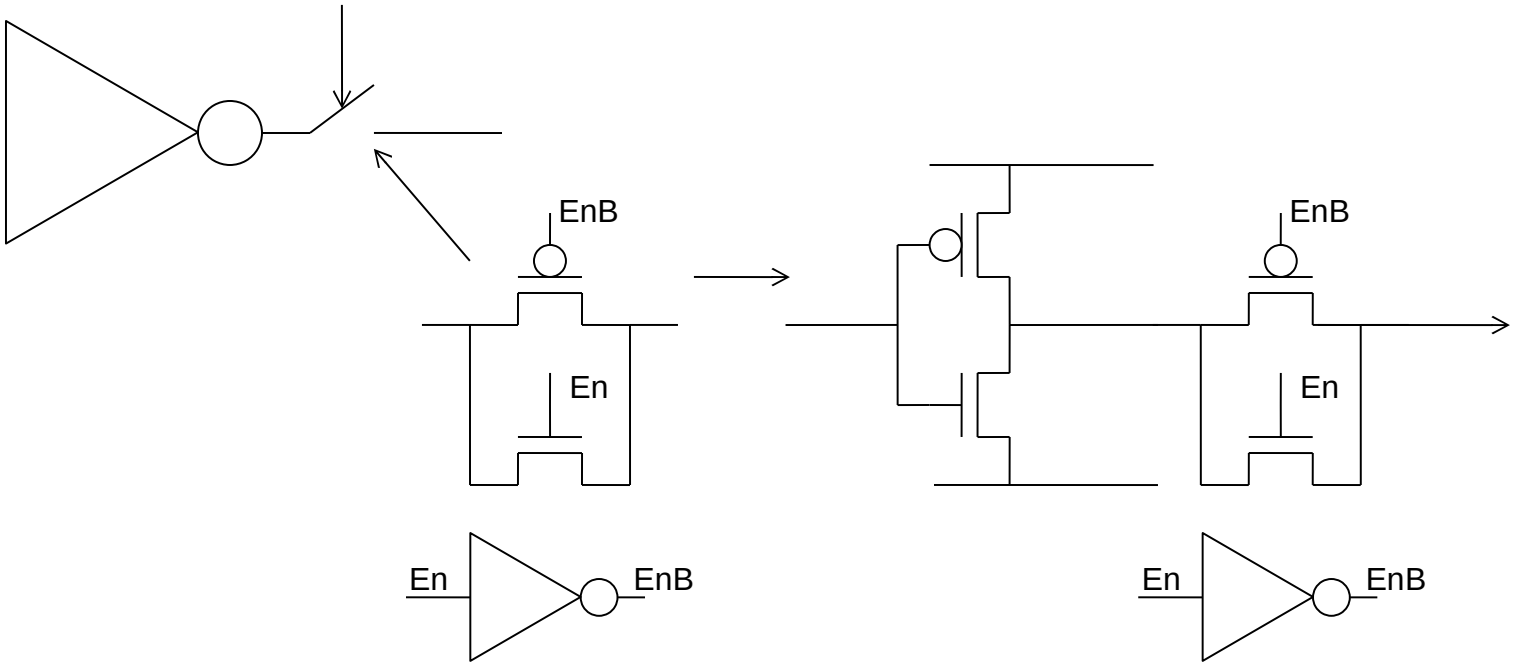
- Schaltung



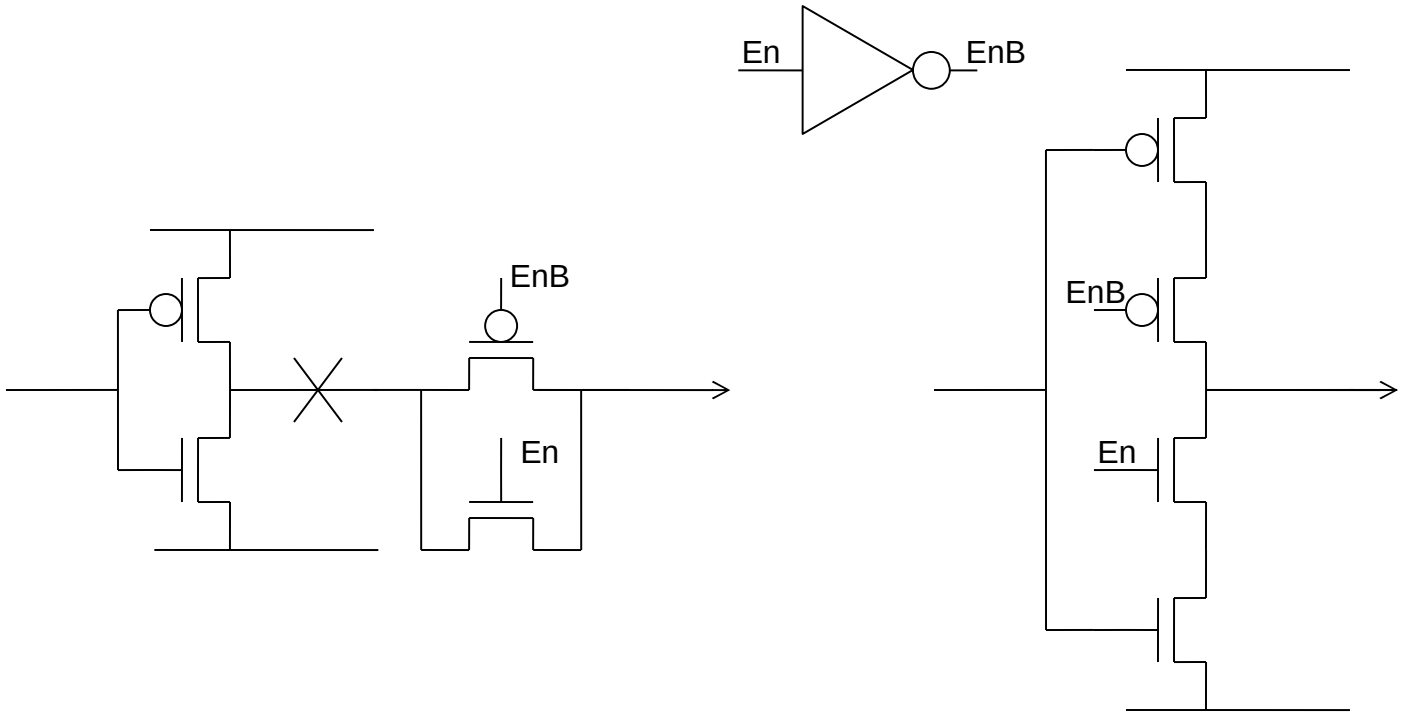
- Schaltung



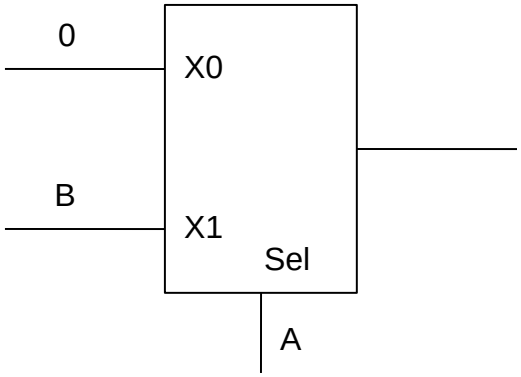
- Schaltung

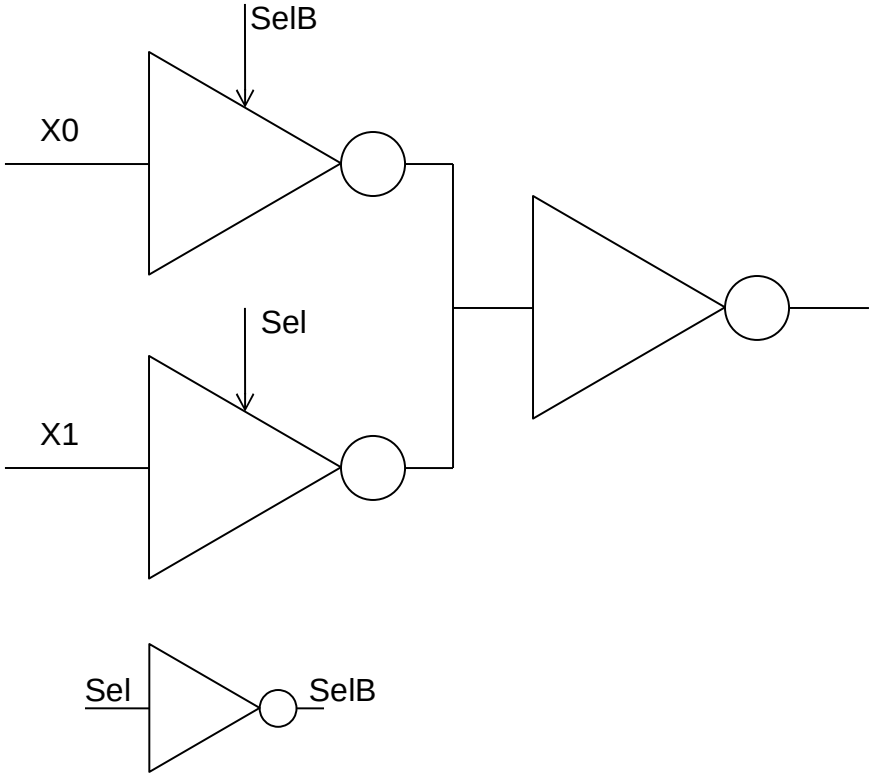


- Schaltung

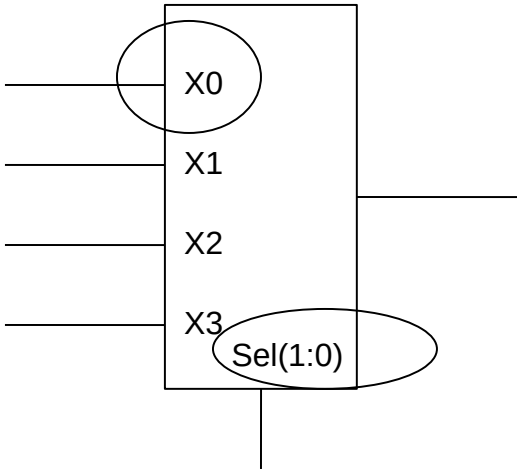


- Multiplexer mit Gated Invertern
- Wir brauchen zwei normale- und zwei Gated Invertern – es sind insgesamt 2 x 2 + 2 x 4 = 12 Transistoren - besser

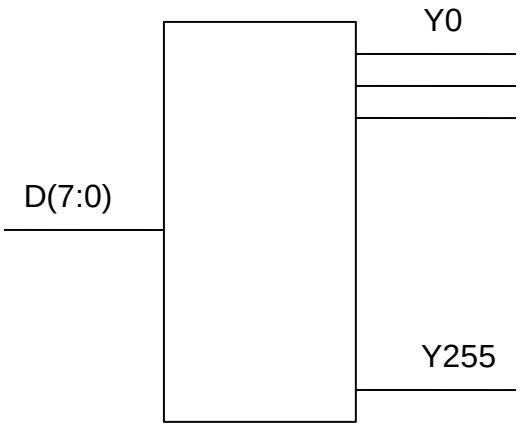




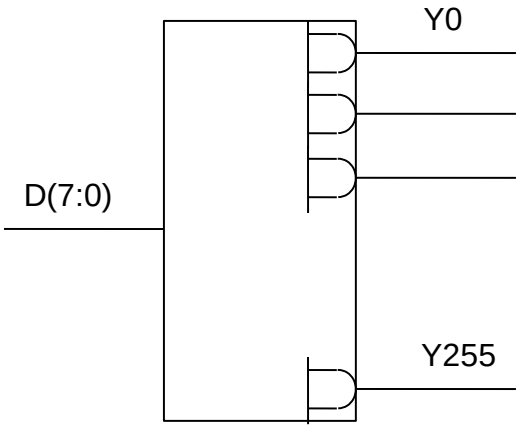
- Oft werden auch mehrfache Multiplexer verwendet.
- ZB wenn man die digitalen Signale von mehreren Quellen über eine Leitung übertragen möchte.
- 2->1 Multiplexer
- 4->1 Multiplexer…

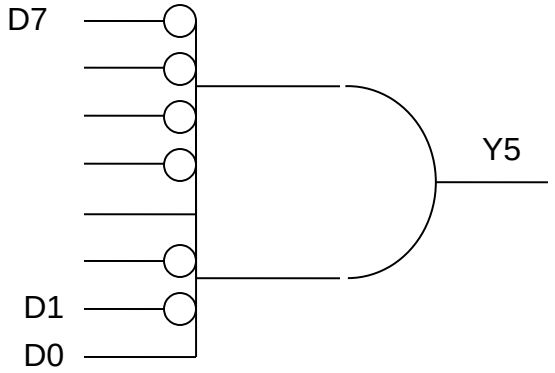


- Decoder.
- Beispiel: 8-bit Eingang D(7:0) (binäre Zahl) und 2^8 Ausgänge.
- Falls Eingang = m (binär Kodiert) ist der m-te Ausgang 1. Alle anderen Ausgänge sind null.

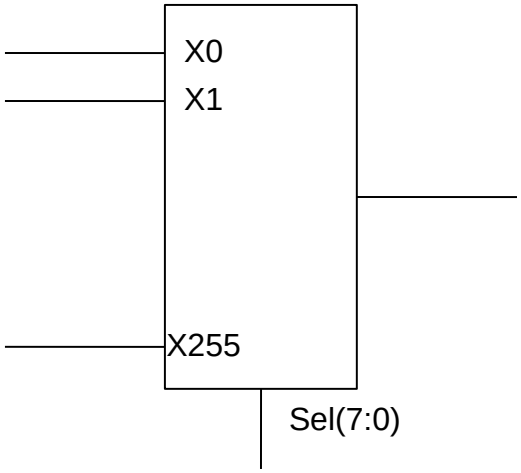


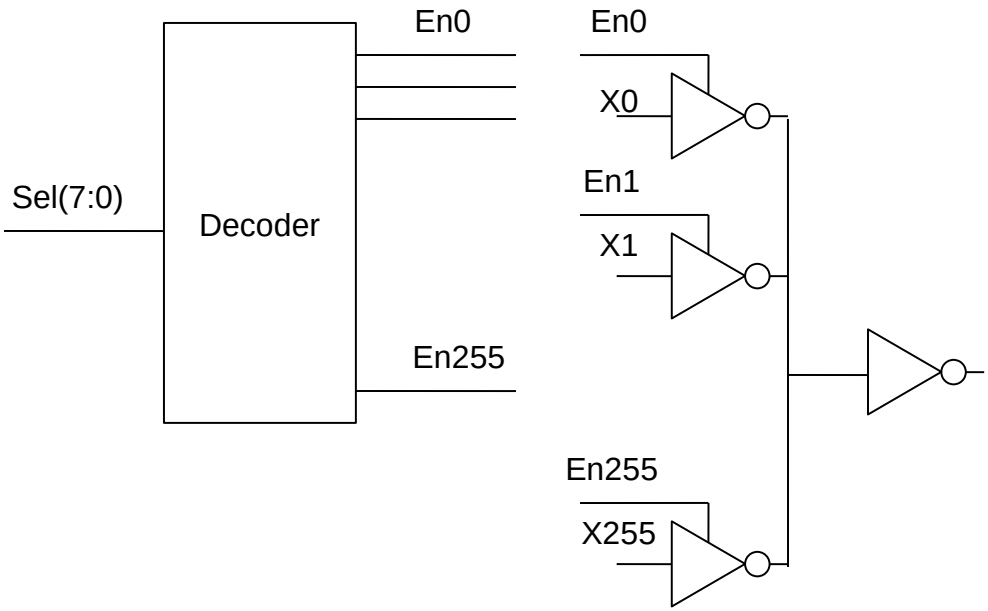
- Realisierung
- 2^8 AND Gattern mit n Eingängen
- Wenn z.B. das AND Gate dem Ausgang 5 gehört, sollte es „1“ für die binäre Zahl D(7:0) = 0000_1001 erzeugen
- Y5 = !D7 & !D6 & !D5 & !D4 & D3 & !D2 & !D1 & D0
- Alle Variablen, die null sind, werden negiert
- In solcher Realisierung brauchen wir 256 ANDs mit 8 Eingängen und 8 Invertern. Das sind insgesamt 256 x 16 + 8 x 2 ~ 4000 Transistoren.



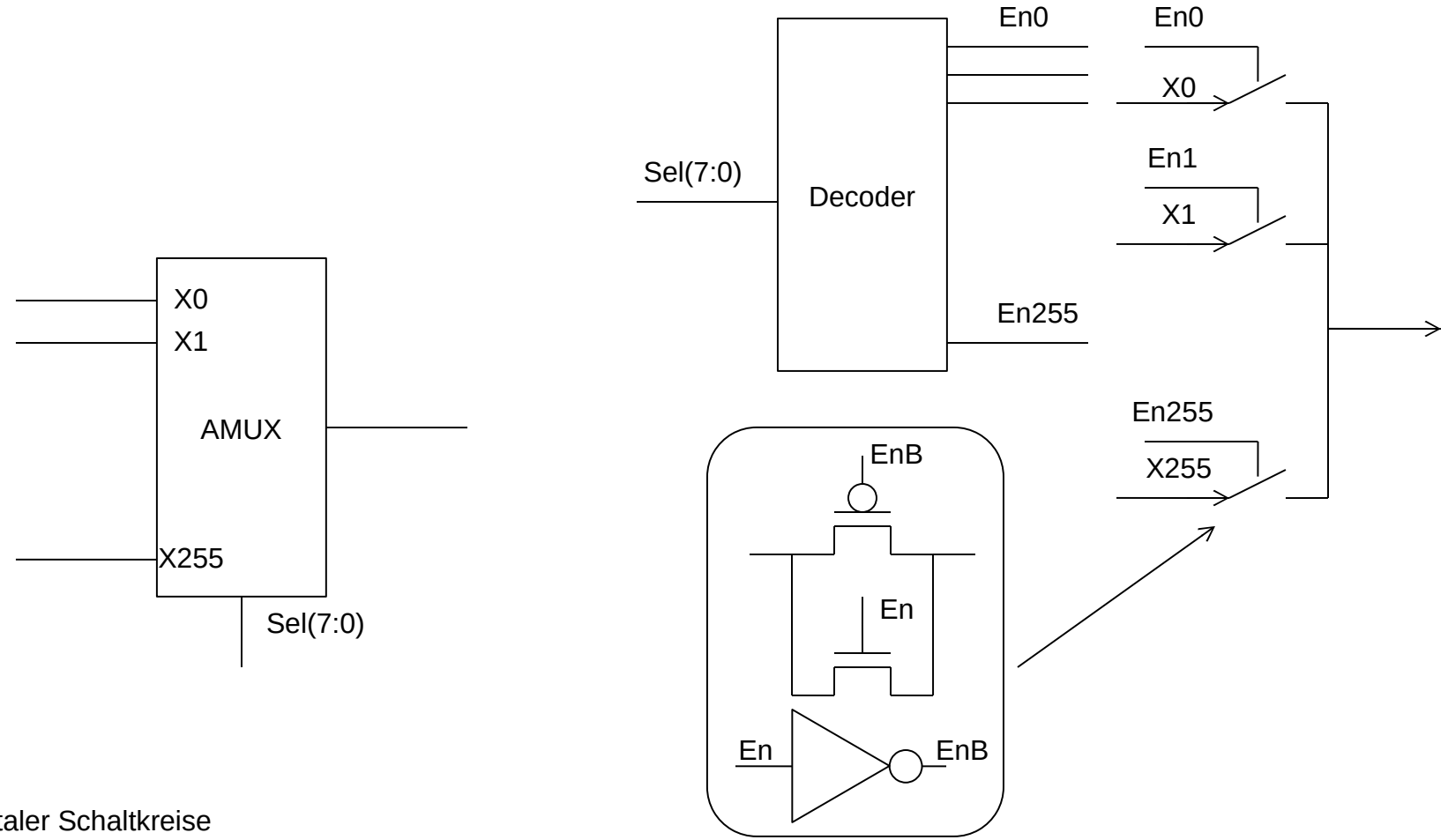


- 256->1 Multiplexer – Realisierung
- 1x4000T + 256 x 6T + 2T = 5500 Transistoren

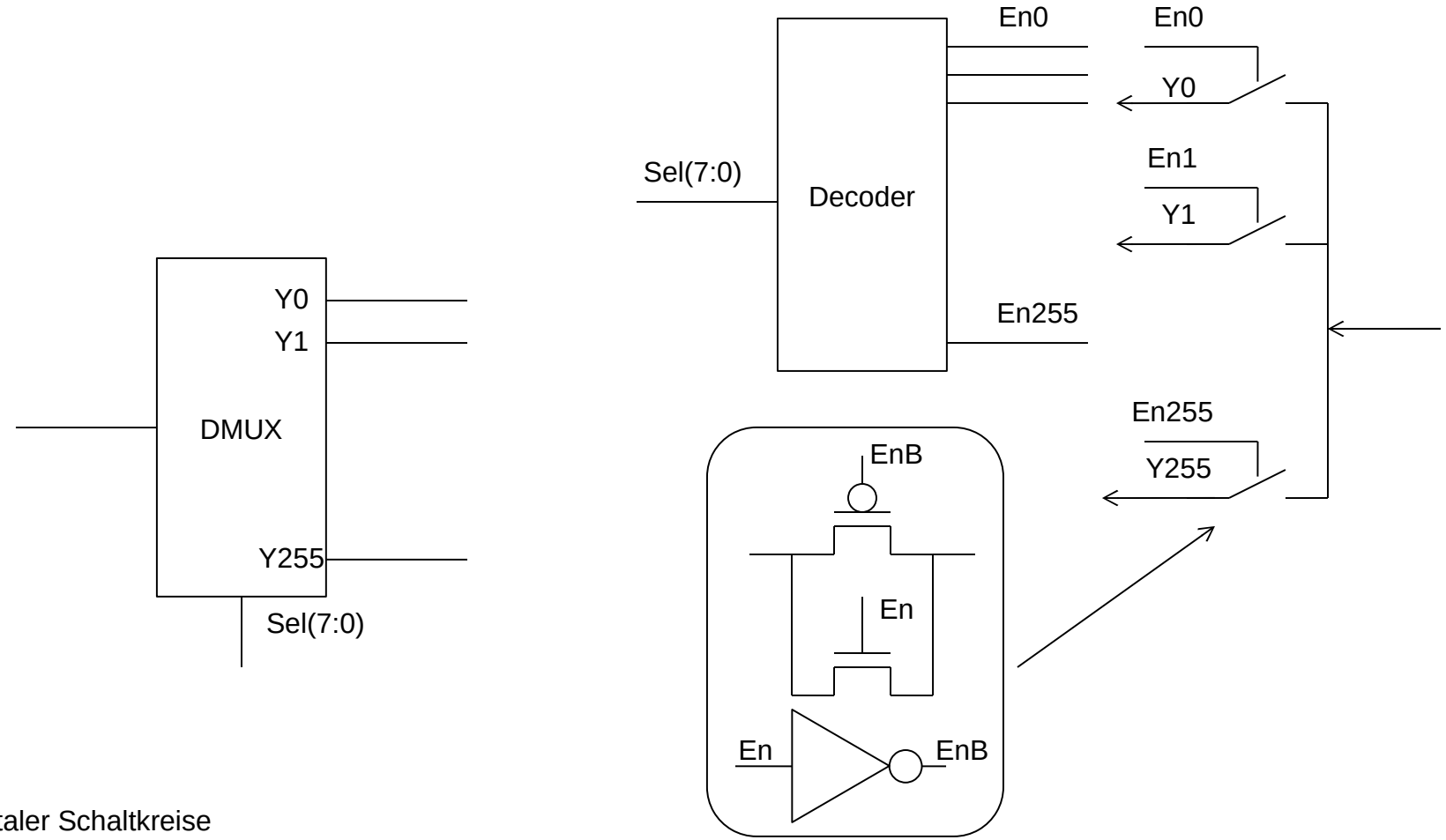




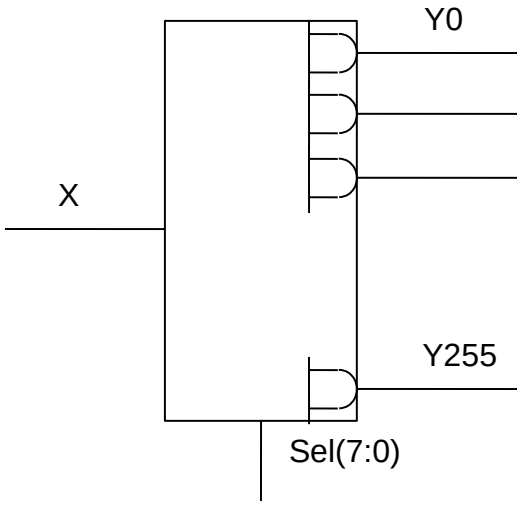
- 256->1 Analog Multiplexer – Realisierung

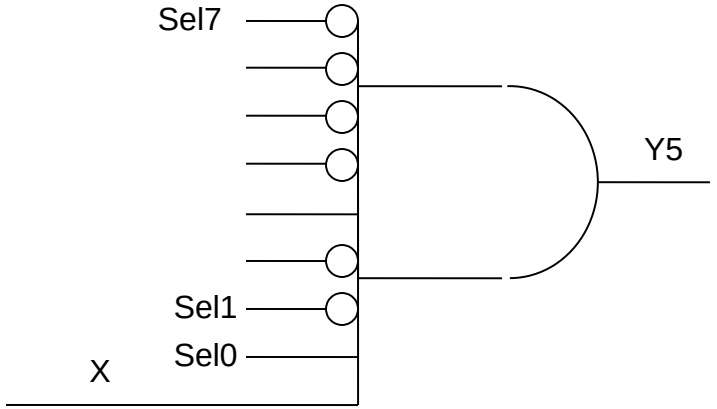


- Analog Multiplexer leitet in beide Richtungen
- Demultiplexer
- Wenn man in einem Analogmultiplexer die Eingänge und Ausgänge „vertauscht“ bekommt man einen Demultiplexer.

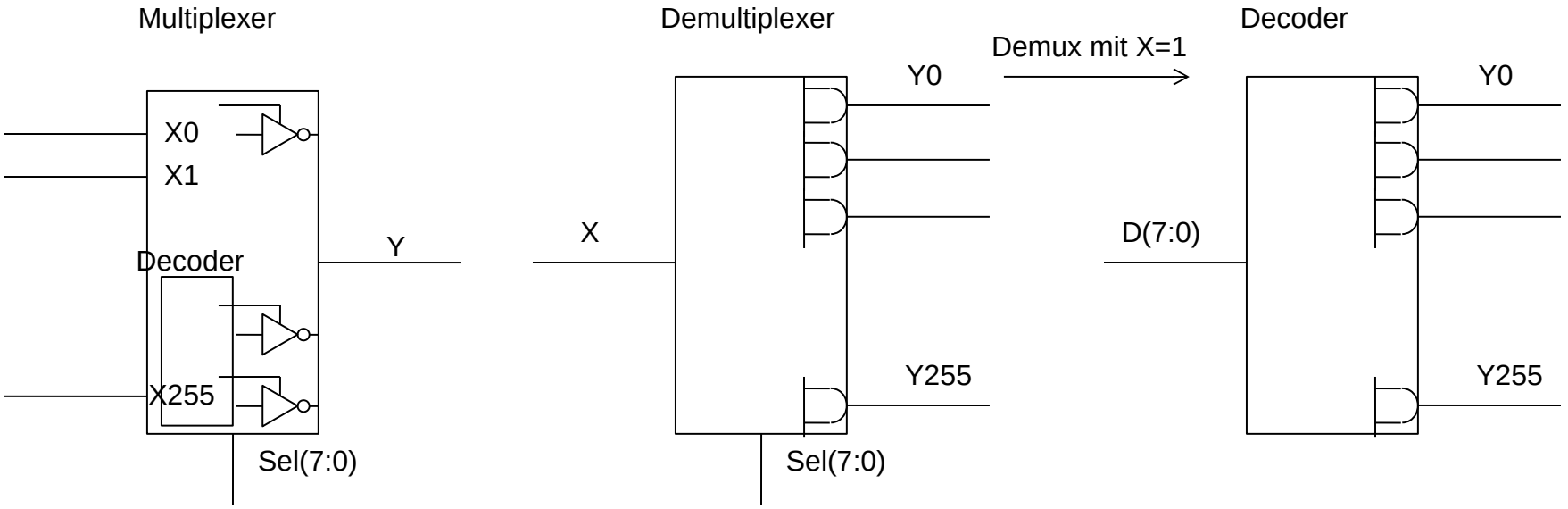


- Digital Demultiplexer
- Demultiplexer mit Eingang 1 -> Dekoder
- 256 x 20T + 8 x 2T ~ 5000T

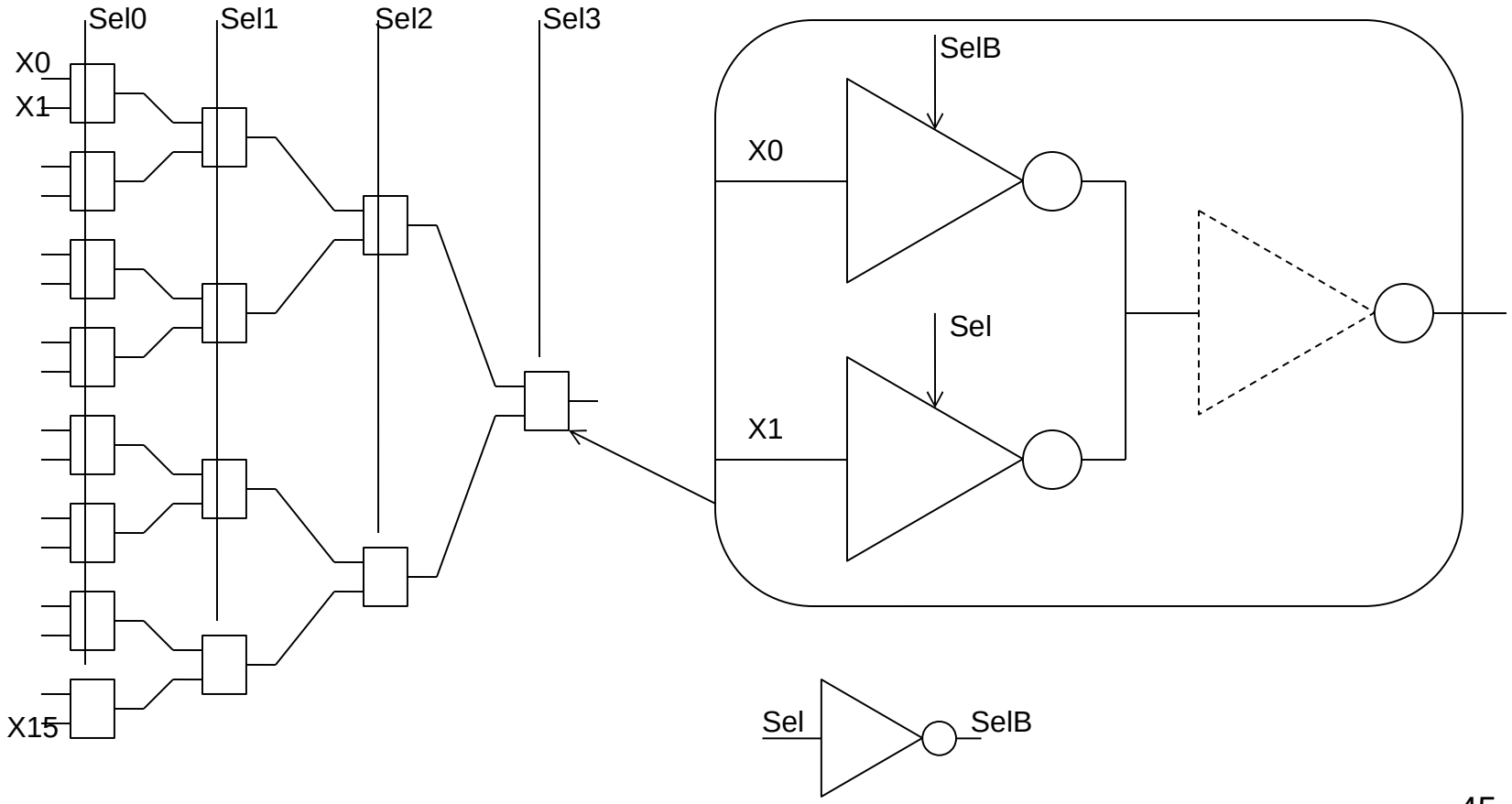




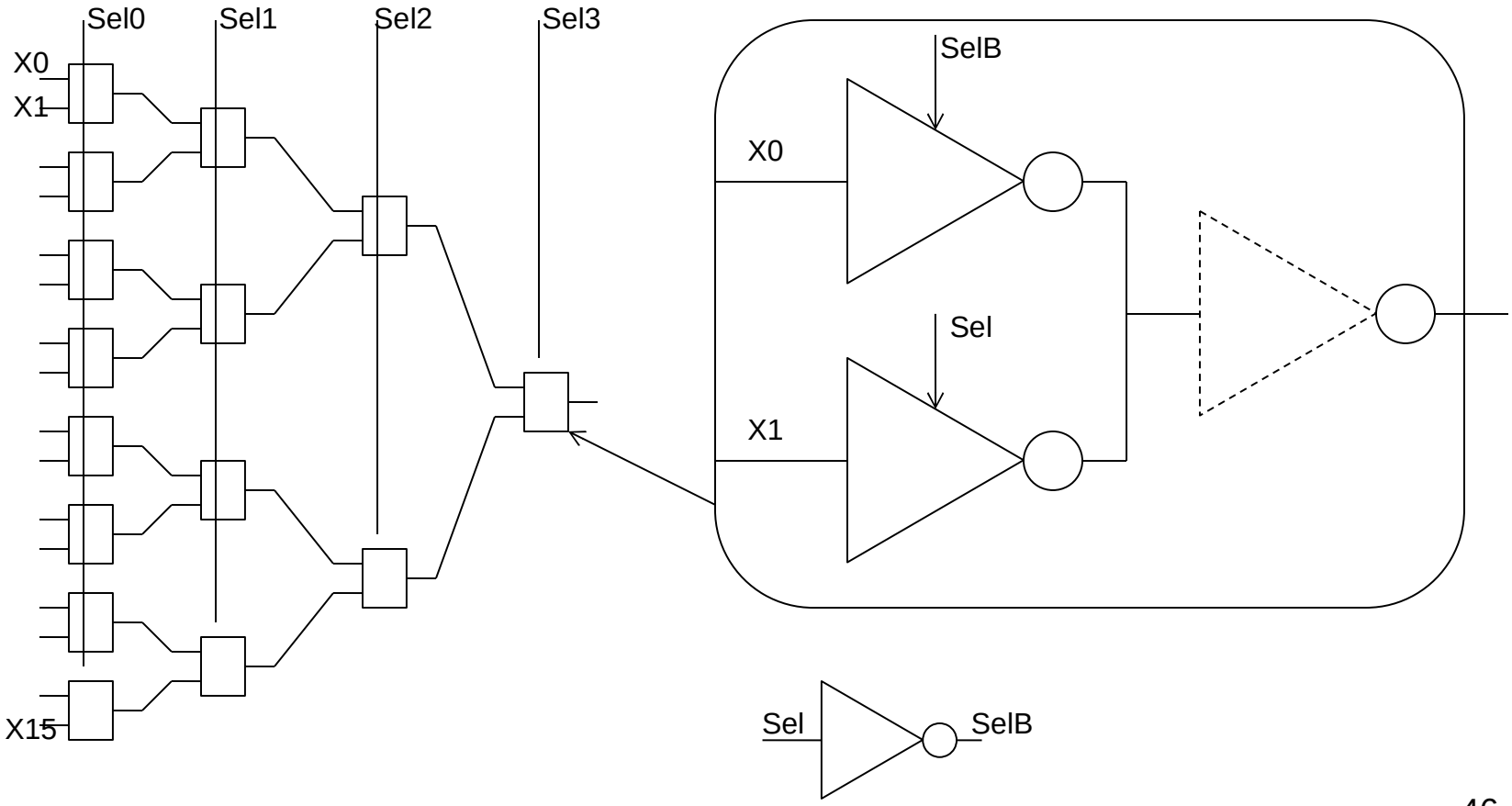
- Multiplexer
- Demultiplexer
- Dekoder
- Coder (nächstes mall)



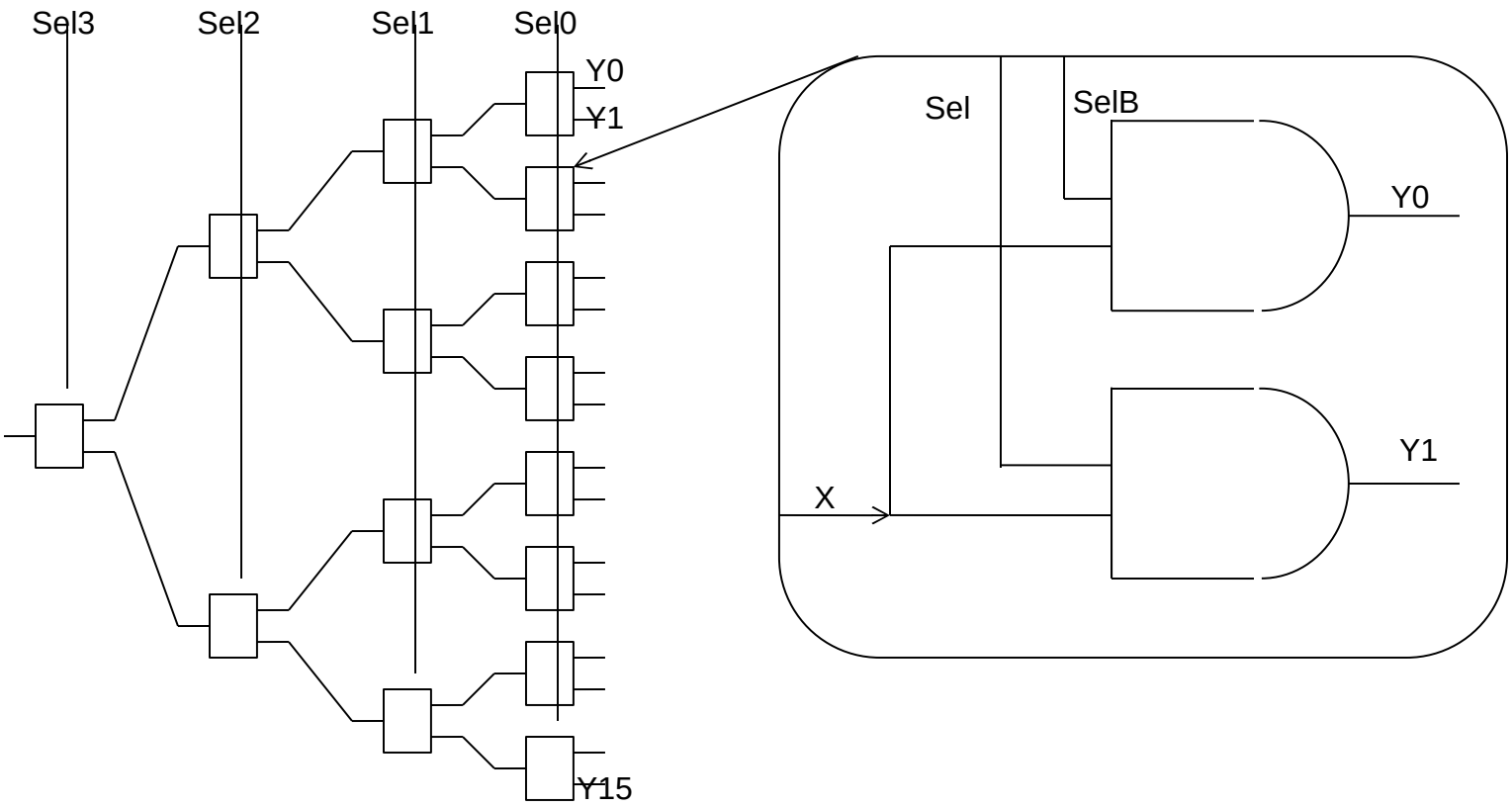
- Multiplexer kann mit weniger Transistoren realisiert werden
- Baumstruktur
- In jedem Knoten verwenden wir jeweils einen (2->1) Multiplexer. Der Select Eingang vom Multiplexer der ersten Stufe wird an Sel0 angeschlossen, usw.



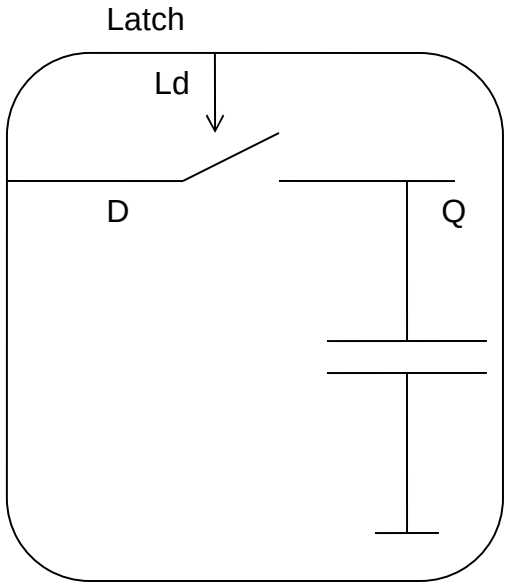
- 256->1 Multiplexer
- 255 (2->1) x 4T + 8x2T ~ 1000 Transistoren (5.5x kleiner)
- Die Schaltung ist langsamer – mehrere Stufen

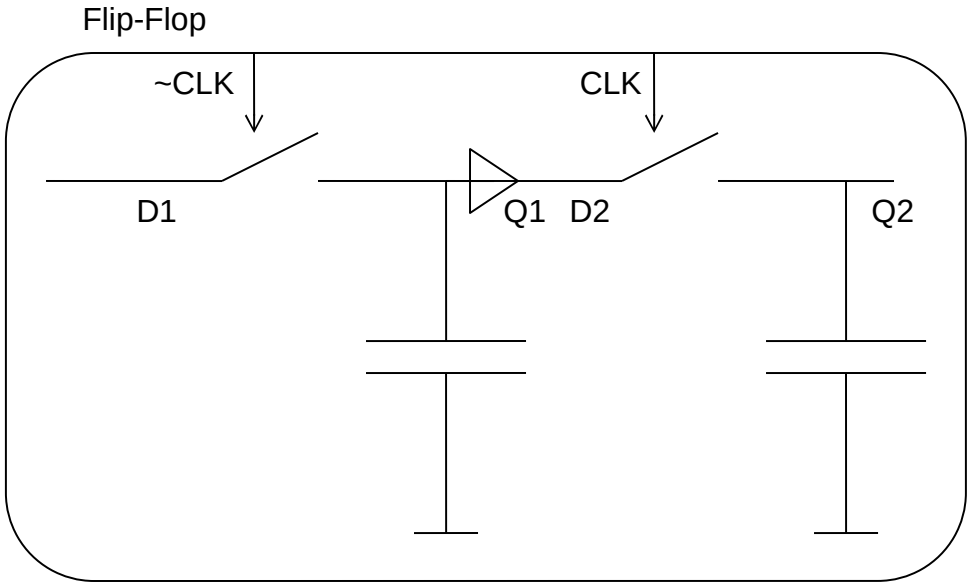


- Auch ein Demultiplexer (und Dekoder) kann als Baumstruktur realisiert werden
- In jedem Knoten verwenden wir jeweils einen (1->2) Demultiplexer
- 255 (1->2) x 12T + 8x2T ~ 3000 Transistoren (2000 mit Tricks)

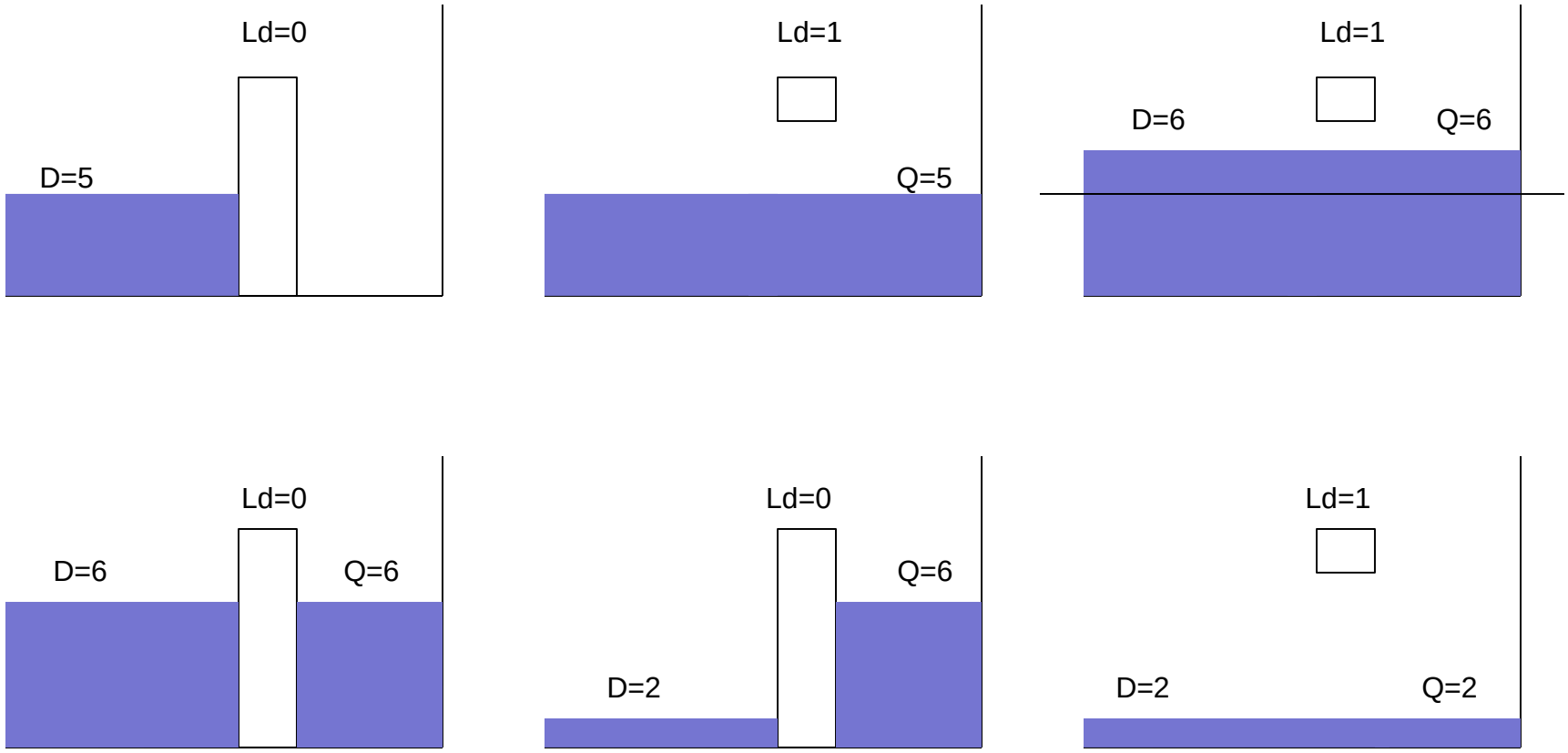


- Sequenzschaltungen
- S. Vorlesung 1 – Latch und Flip-Flop
- Latch – speichert ein Eingangsniveau (auf einem Kondensator) wenn Load Signal = 1. Wenn Load = 0, der Zustand bleibt erhalten
- Flip-Flop – 2 Latch-es in Reihe
- Der Eingangswert D wird im Moment der steigenden Talkflanke gespeichert
- Spätere Änderungen am D-Eingang haben keine Wirkung auf den Ausgang bis zur nächsten Taktflanke

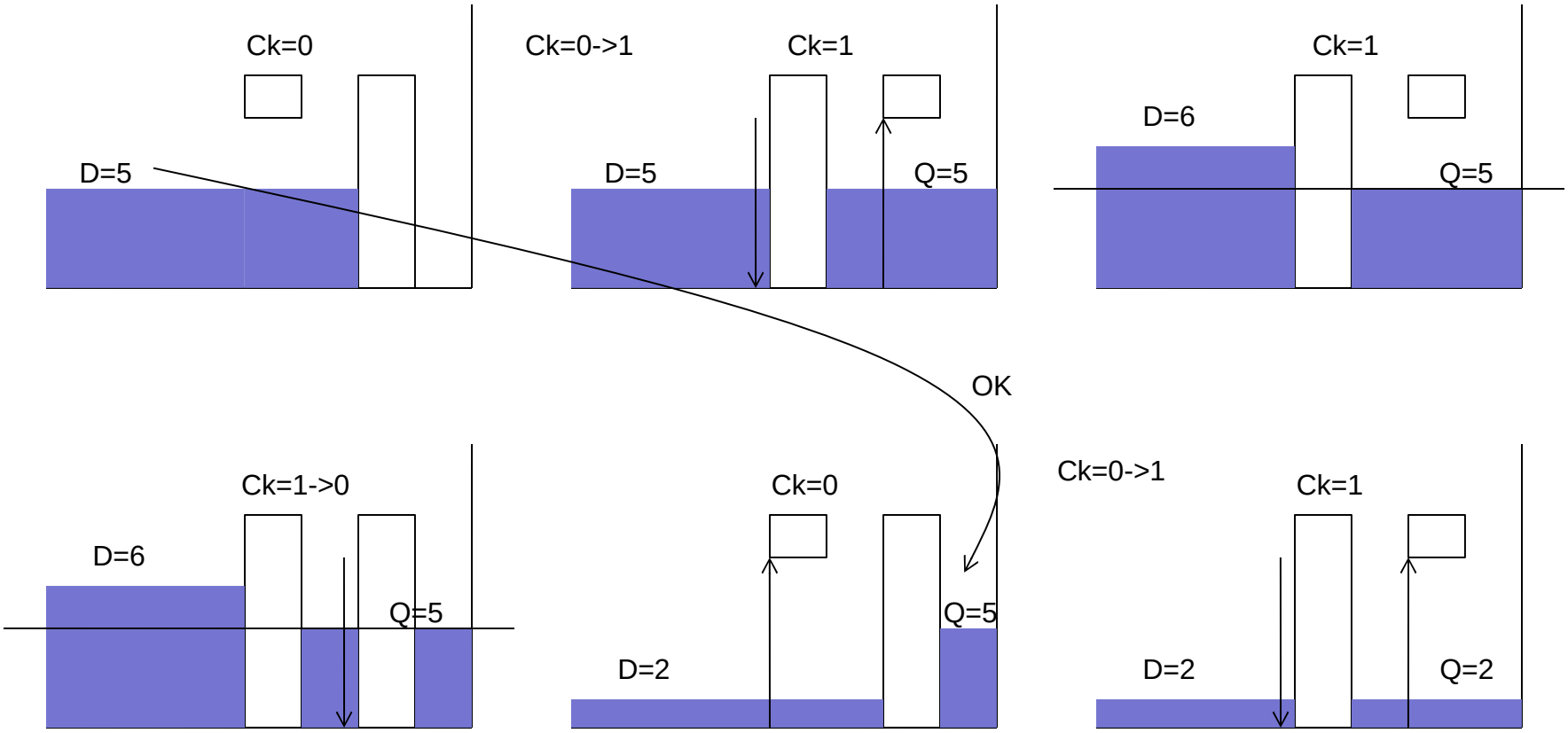




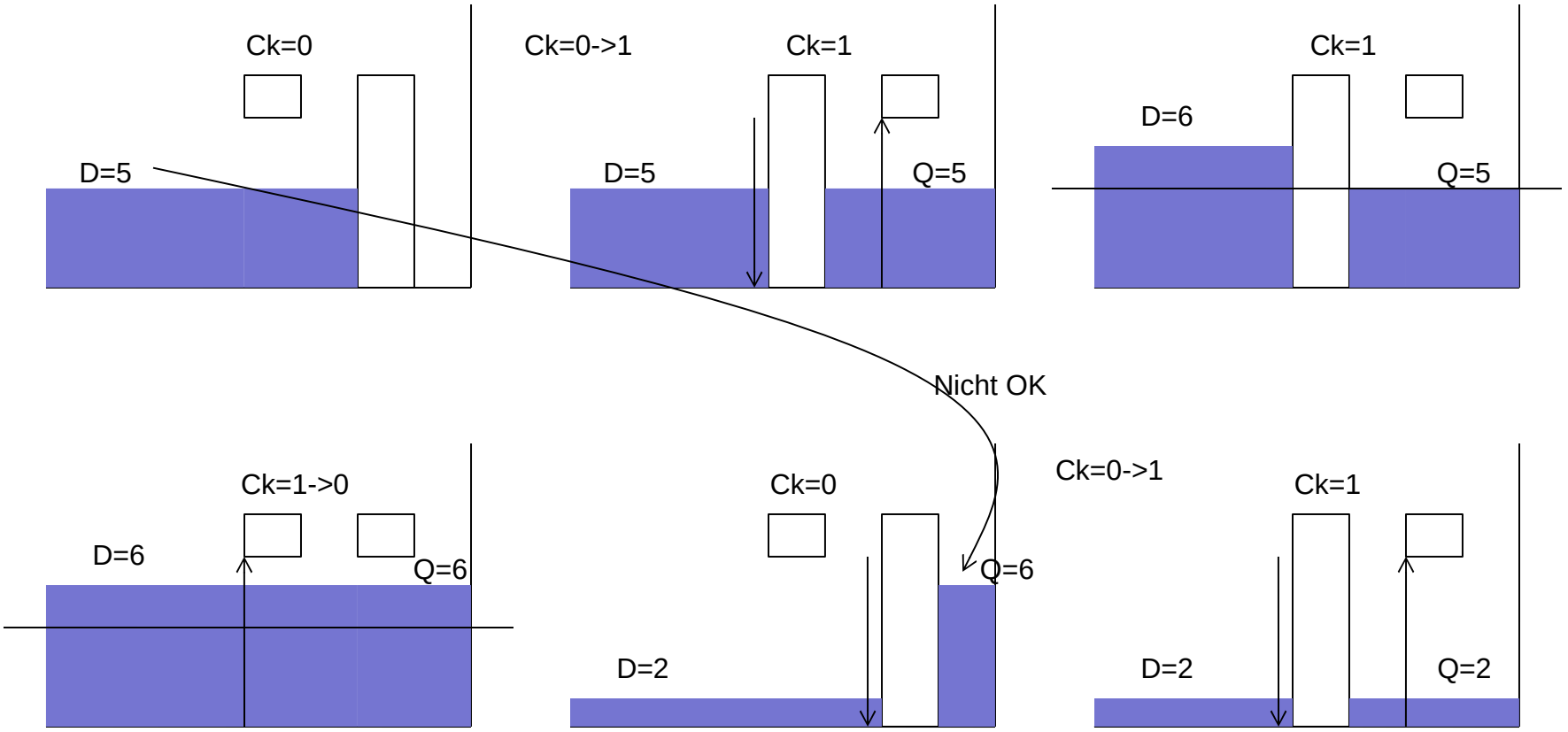
- Es erinnert an ein System mit Schleusen.



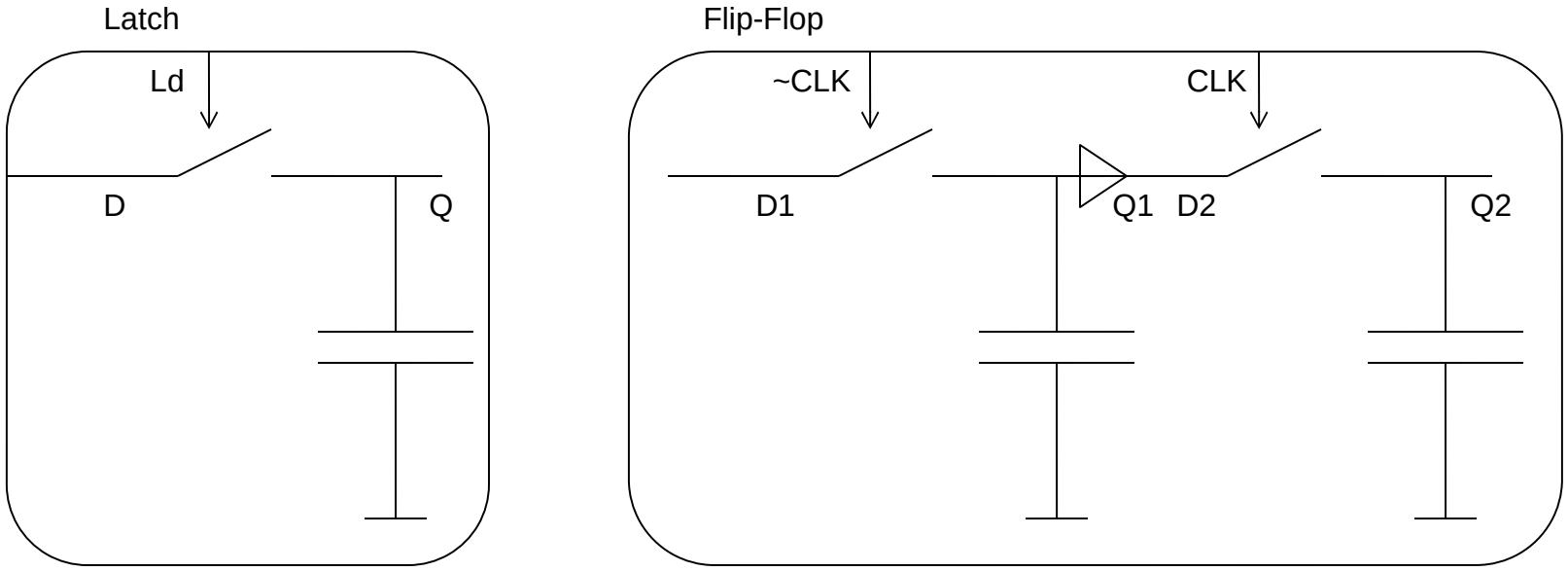
- Es erinnert an ein System mit Schleusen.



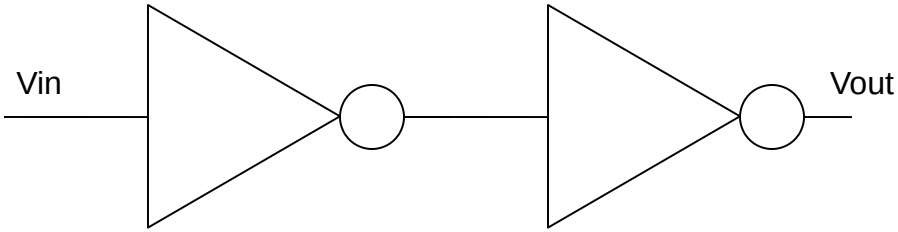
- Es erinnert an ein System mit Schleusen.

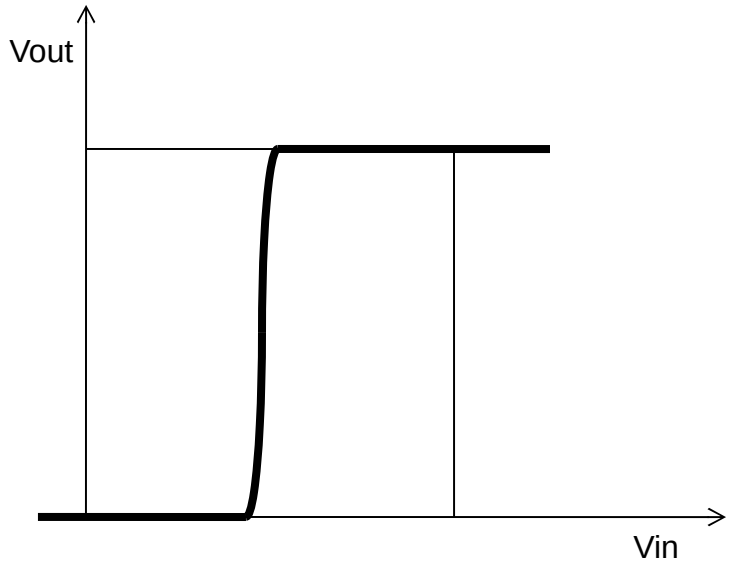


- Nachteil einer Latch Schaltung mit Kondensator – sie kann den Zustand nicht beliebig lange halten.
- Der Kondensator wird langsam entladen.
- Dynamische Logik.

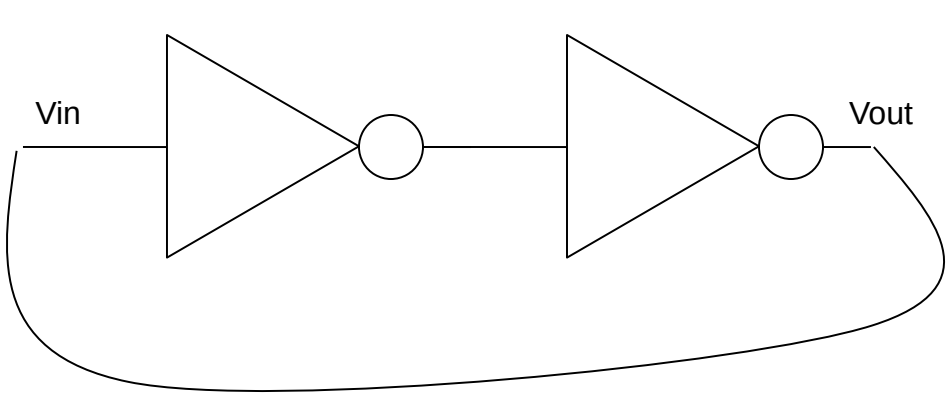


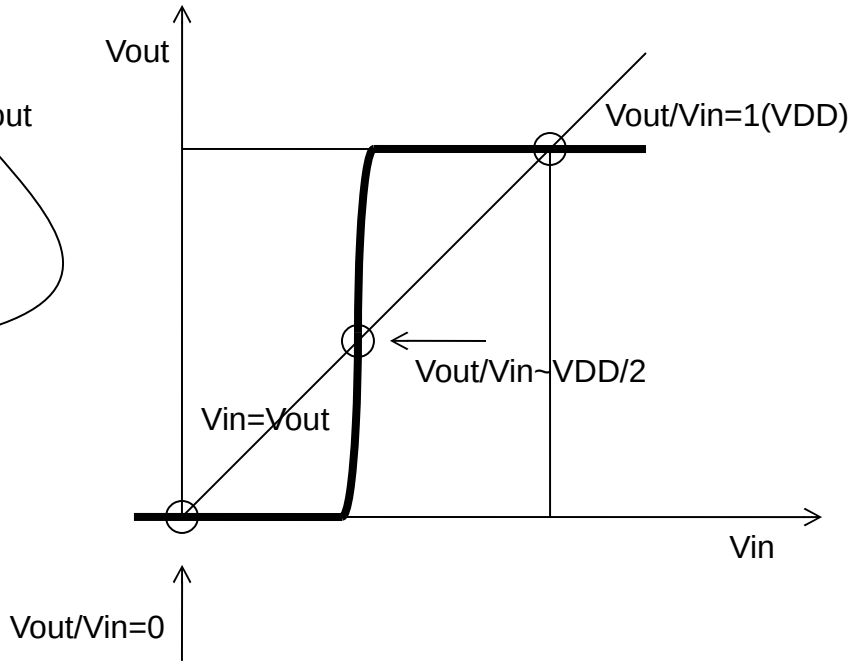
- Statische Speicherzellen



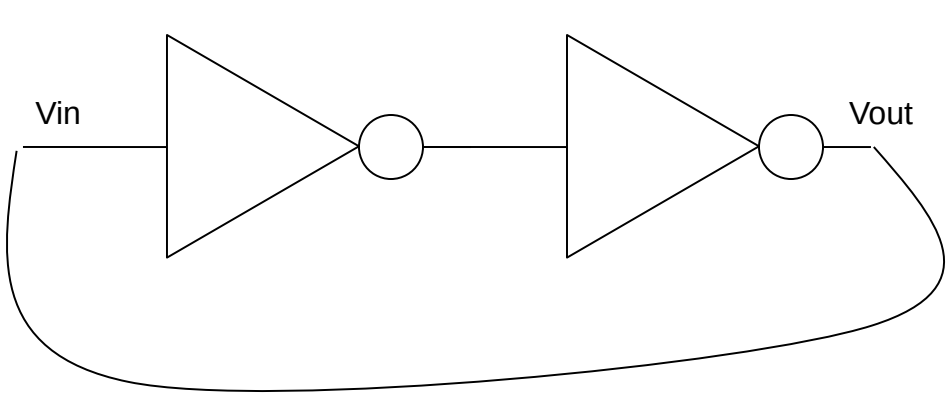


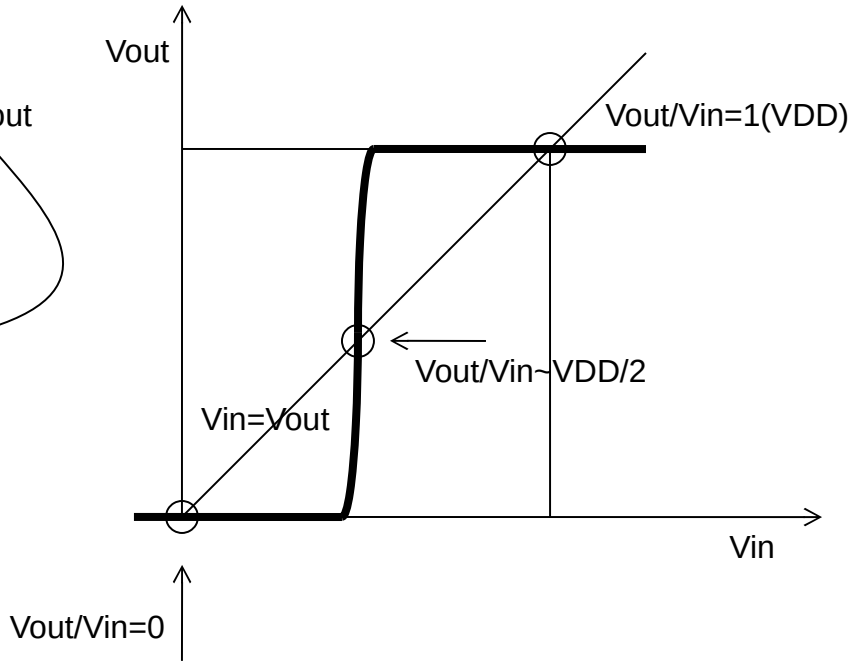
- Statische Speicherzellen
- Wenn wir den Ausgang des zweiten Inverters mit dem Eingang des ersten verbinden, haben wir Vin = Vout.
- Der Zustand der Schaltung liegt im Schnittpunkt der Kennlinie Vout = f(Vin) und der Gerade Vout = Vin.



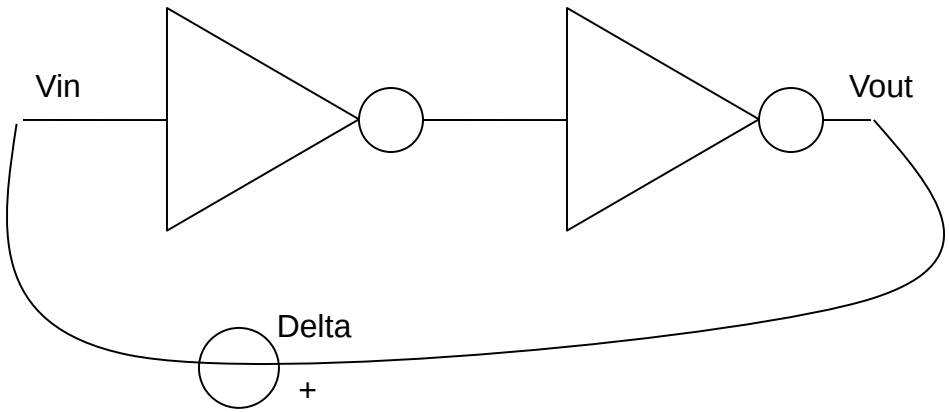


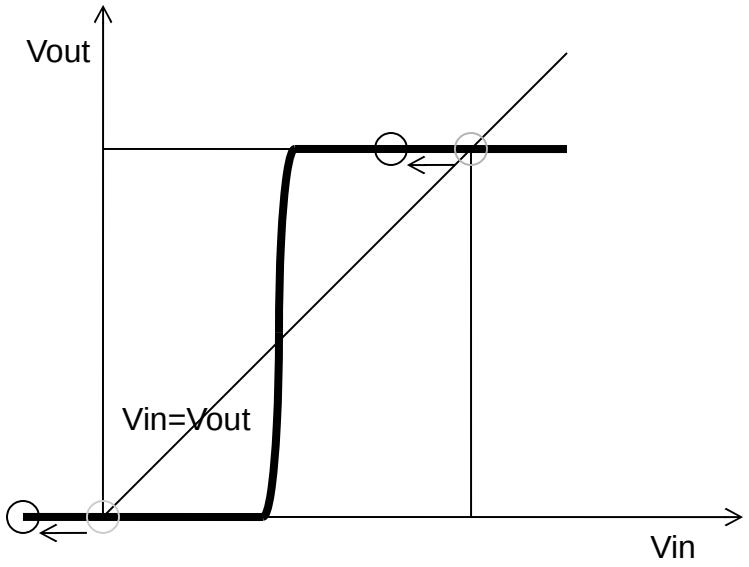
- Drei Schnittpunkte
- 1. Vout/Vin = 0 (logische 0)
- 2. Vout/Voin = VDD (logische 1)
- 3. Vout=Vin ~ VDD/2 (undefiniert).



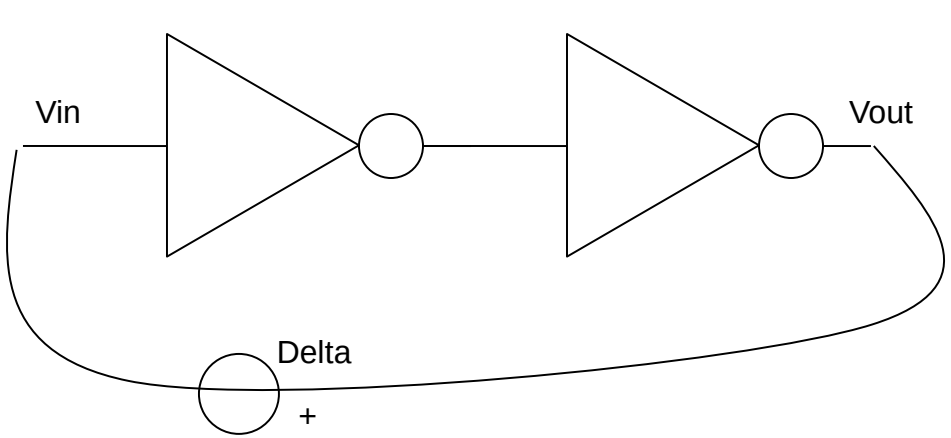


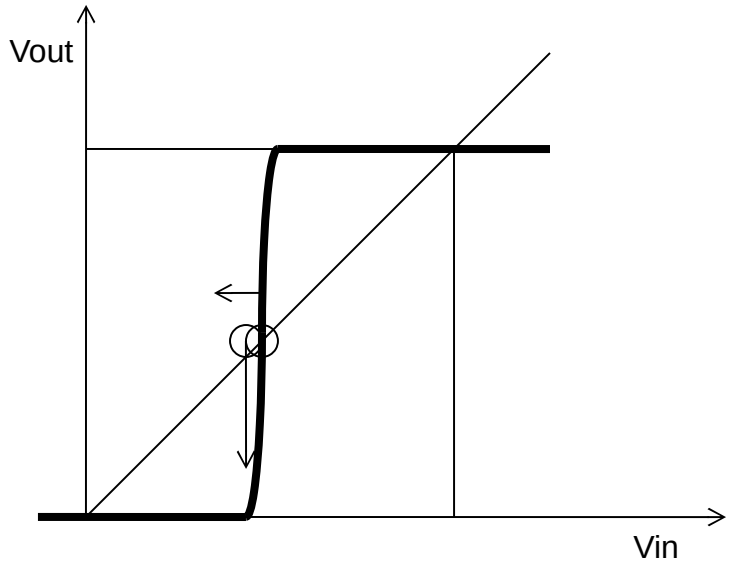
- Die ersten zwei Arbeitspunkte sind stabil
- kleine Störung Delta
- Vin = Vout – Delta
- Vout wird nicht beeinflusst



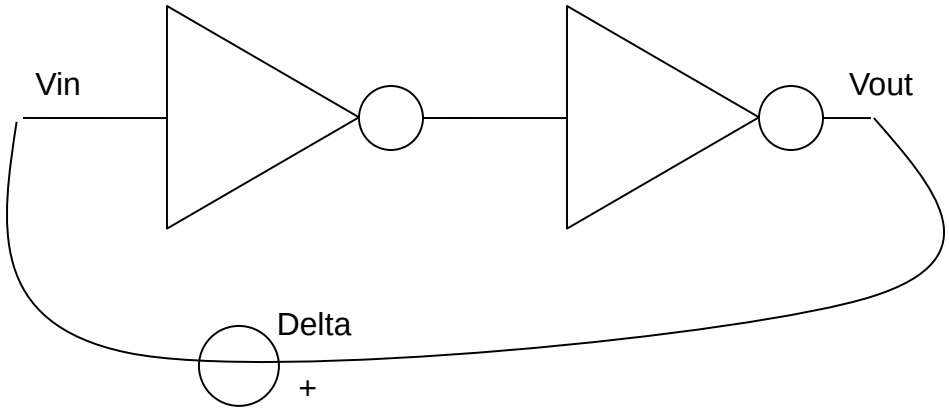


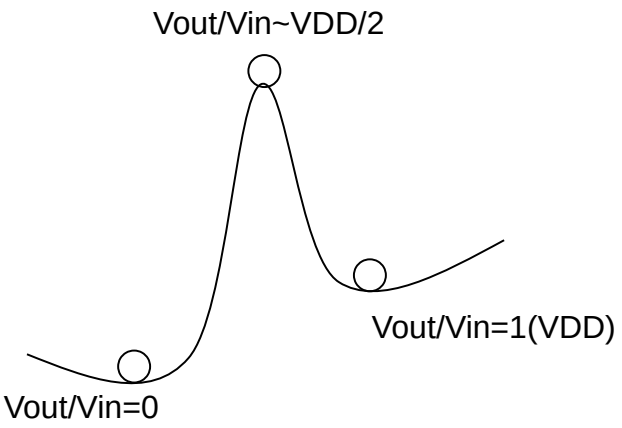
- Der dritte Arbeitspunk ist instabil
- Vin = Vout – Delta
- Verringerung von Vin führt zu noch größerer Verringerung von Vout
- Die Schaltung kommt aus dem instabilen Arbeitspunkt immer in den Arbeitspunkt Vout/Vin = 0 oder in den Arbeitspunkt Vout/Voin = VDD



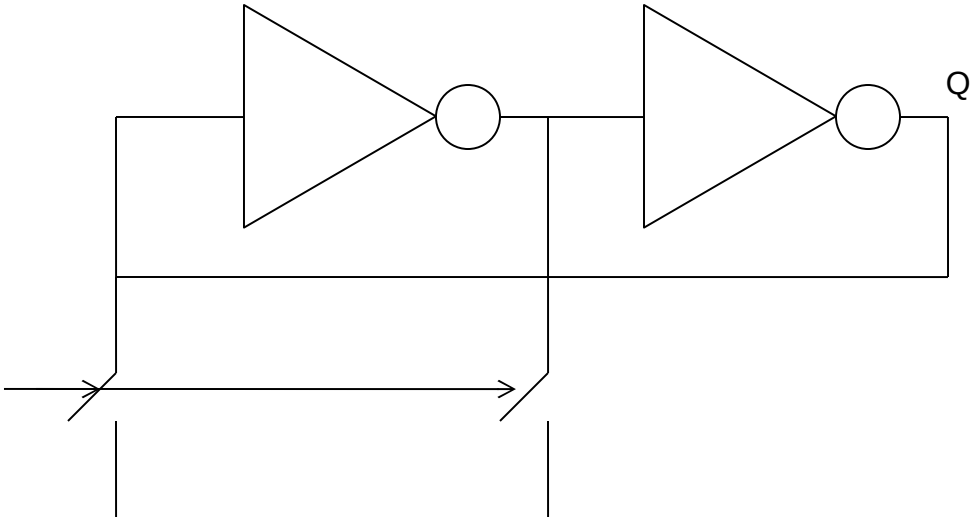


- Der dritte Arbeitspunk ist instabil
- Vin = Vout – Delta
- Verringerung von Vin führt zu noch größerer Verringerung von Vout
- Die Schaltung kommt aus dem instabilen Arbeitspunkt immer in den Arbeitspunkt Vout/Vin = 0 oder in den Arbeitspunkt Vout/Voin = VDD

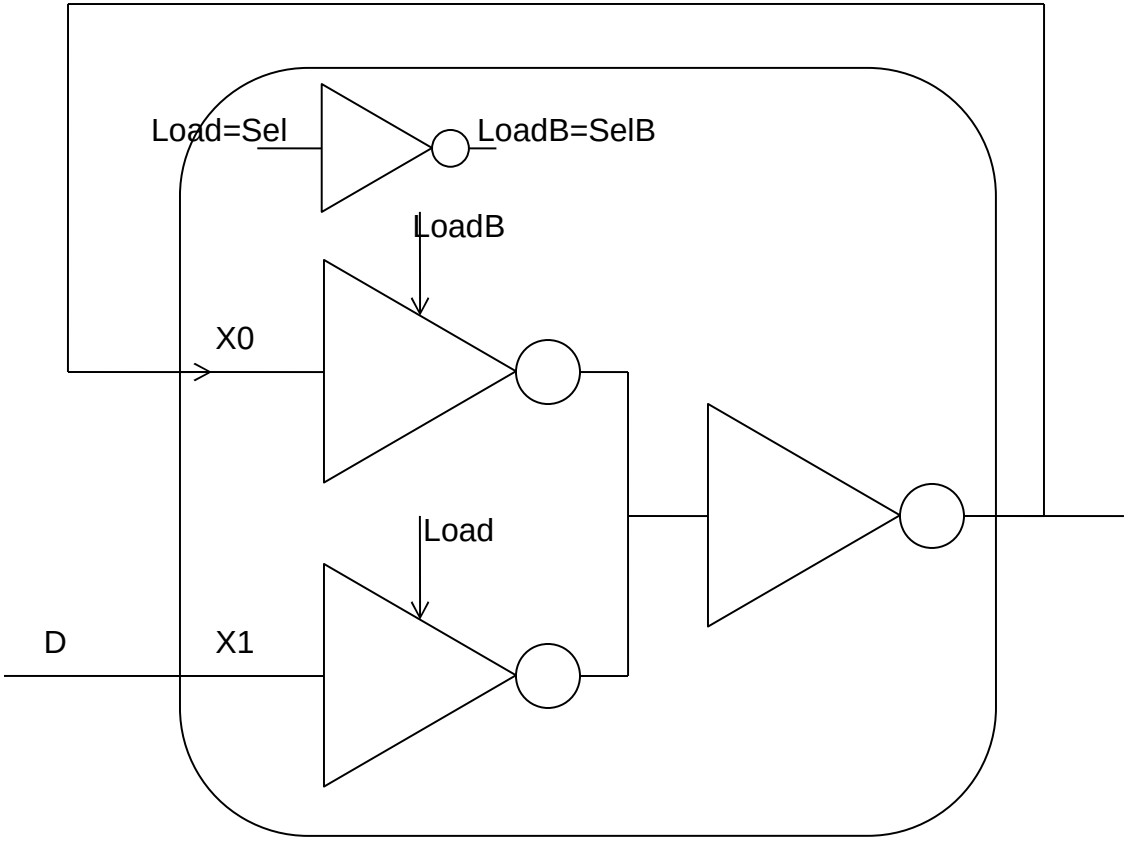




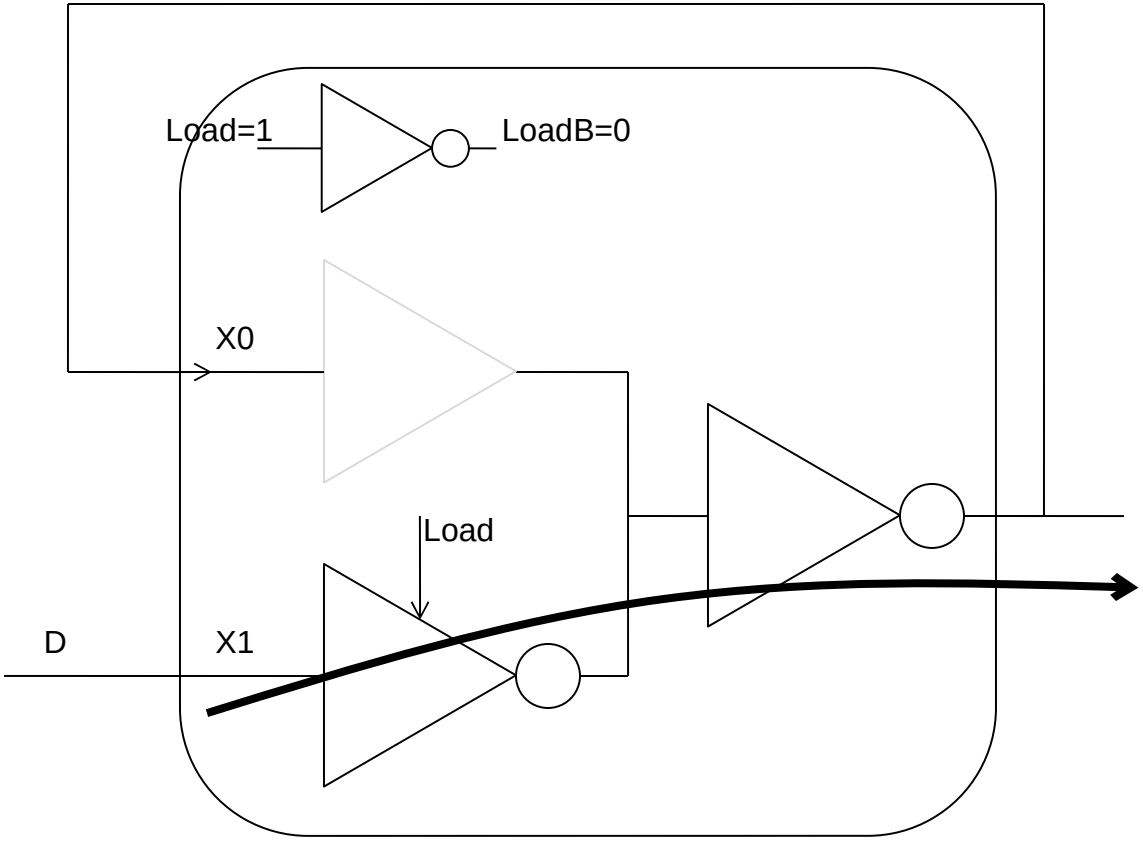
- Die Schaltung ist die Basis einer SRAM Zelle



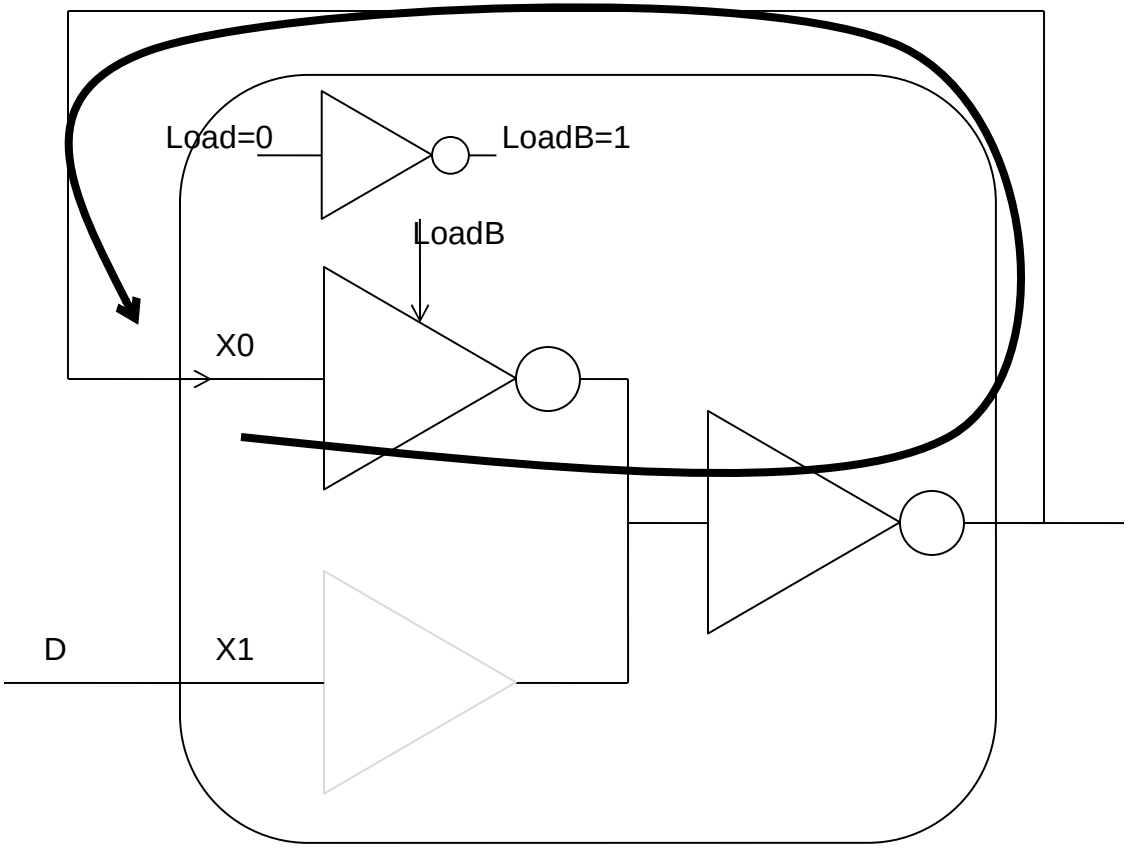
- Ein Latch basiert normalerweise auf einer modifizierten Version der Speicherzelle.
- Multiplexer wird benutzt, Select Eingang ist an Load Signal angeschlossen



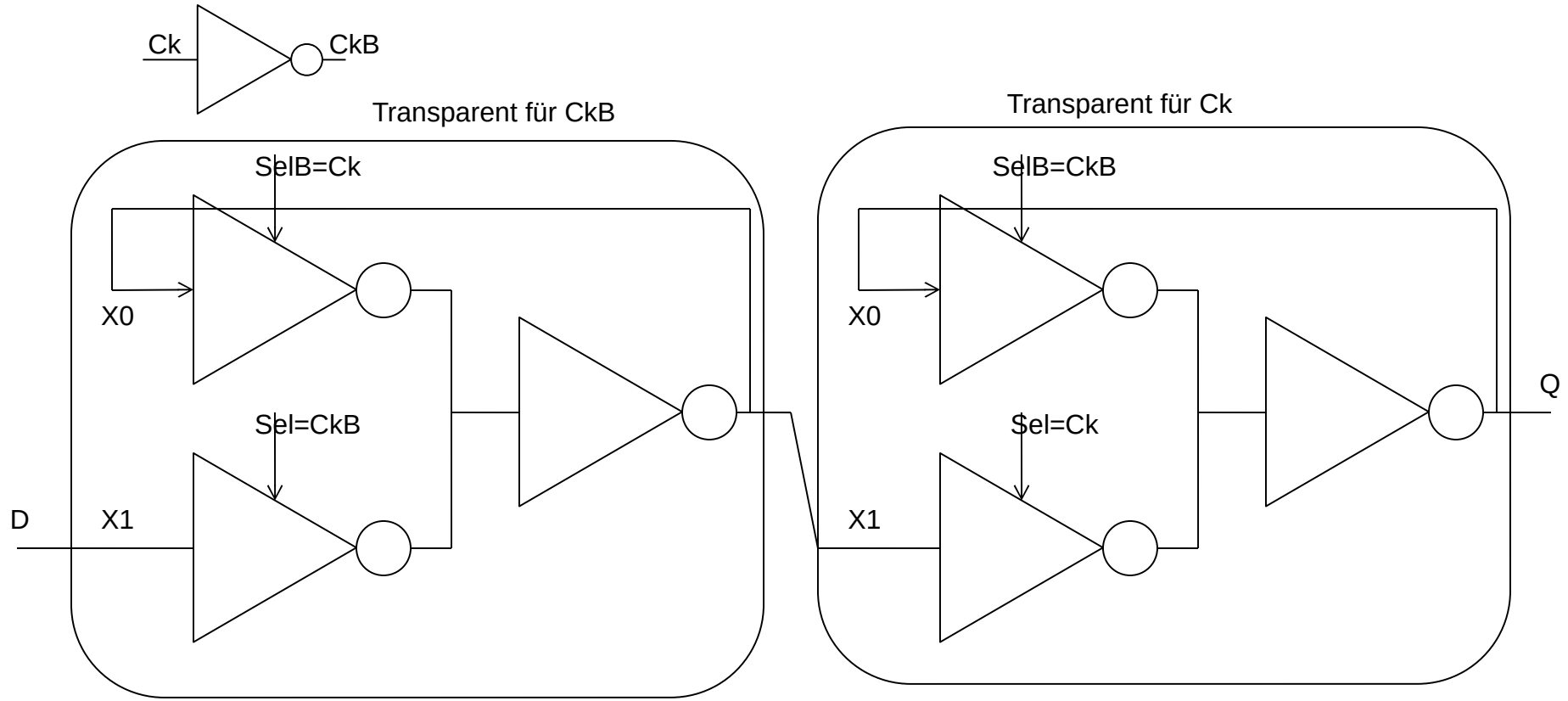
- Wenn Load = 1, das Latch ist „transparent“ – der Eingang ist direkt am Ausgang sichtbar.



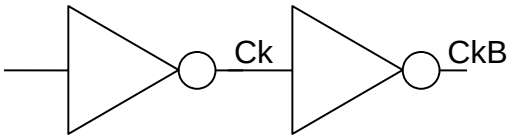
- Wenn Load = 0, haben wir dieselbe Schaltung wie in einer RAM Zelle. Der Multiplexer behält den Zustand

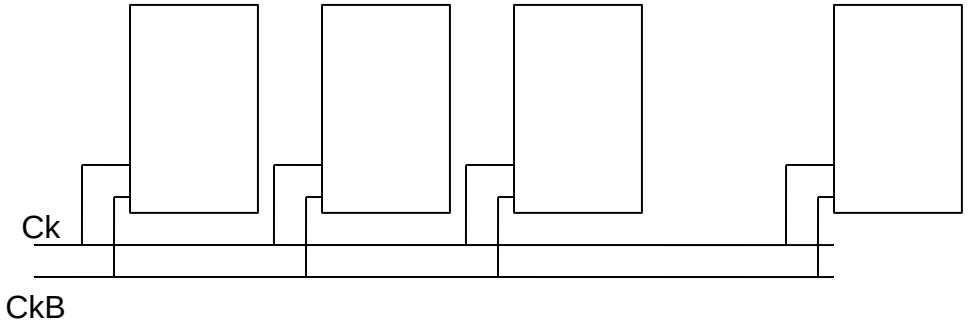


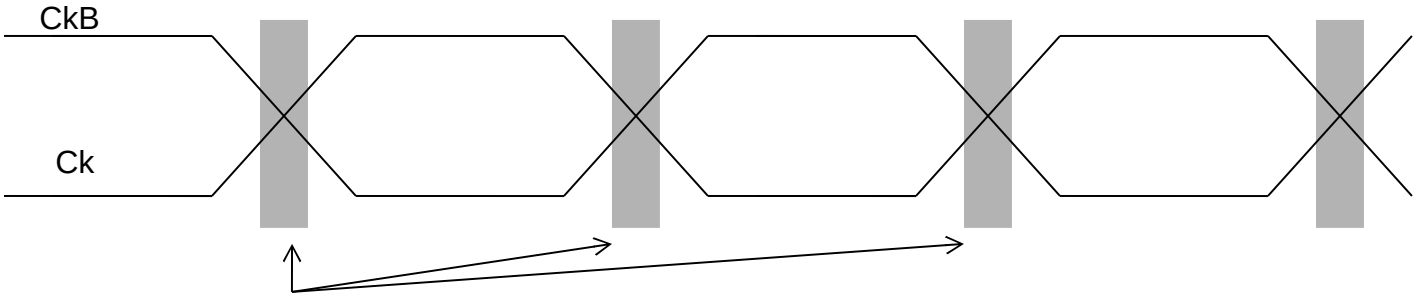
- Einen Flip-Flop bilden wir aus zwei Latches
- Es soll dabei vermieden werden, dass beide Latches gleichzeitig transparent werden, vor allem wenn wich Ck von 1 auf 0 ändert (inaktive Flanke)



- Kapazitive Last verlangsamt die CMOS Schaltungen
- Schlechte Idee – viele Flip-Flops teilen zwei Taktinvertern
- Layout kleiner aber funktioniert nicht

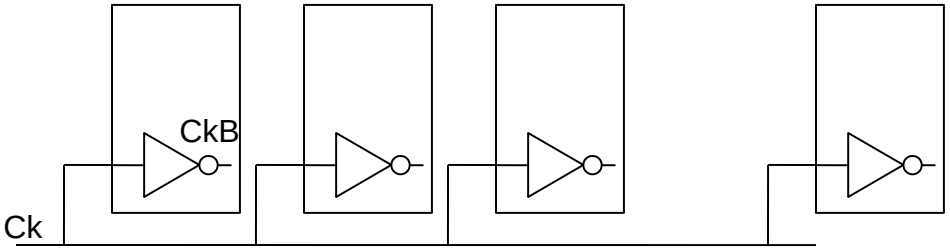


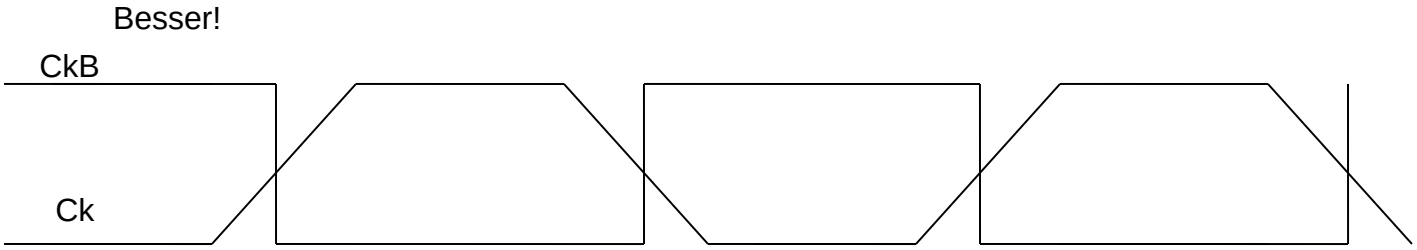




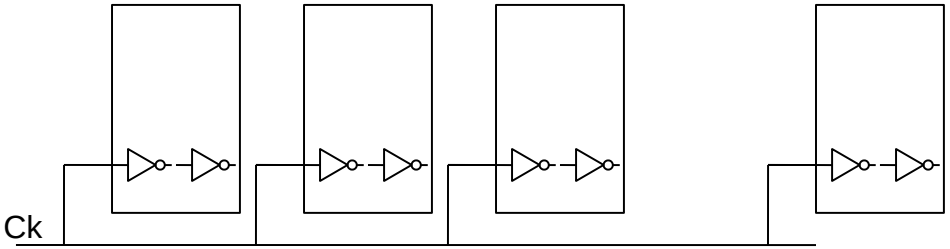
Beide Latches im FF transparent?

- Um solche Probleme zu vermeiden, werden die Takt-Inverter in der Regel im Flip-Flop eingebaut.

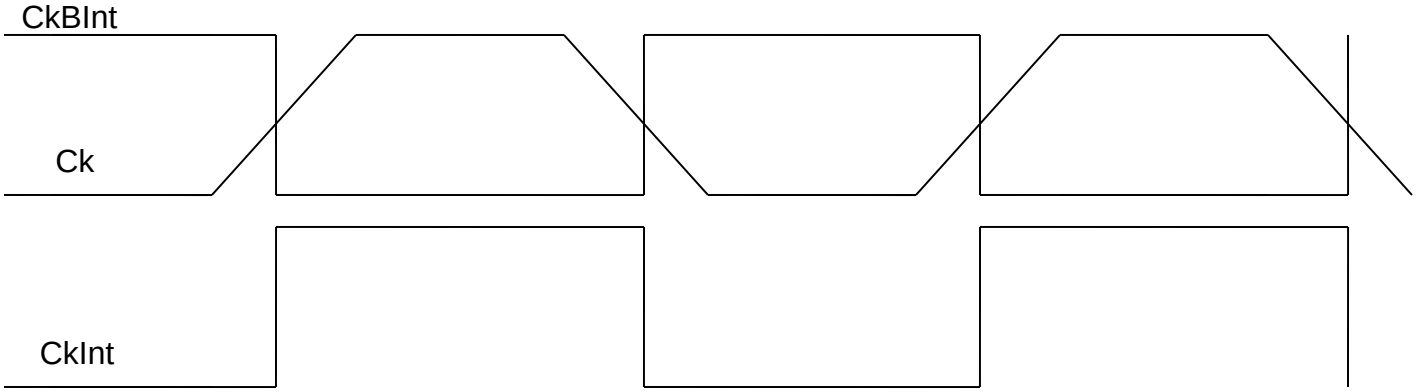




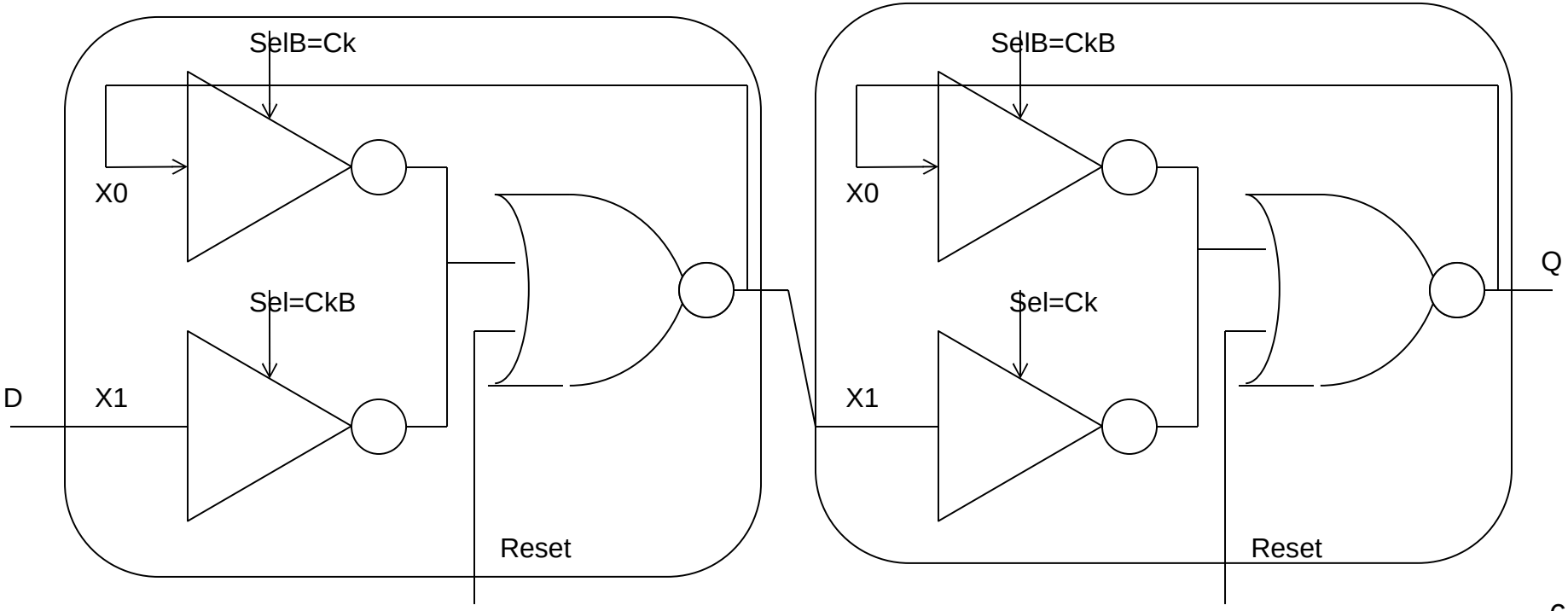
- Um solche Probleme zu vermeiden, werden die Takt-Invertern in der Regel im Flip-Flop eingebaut.



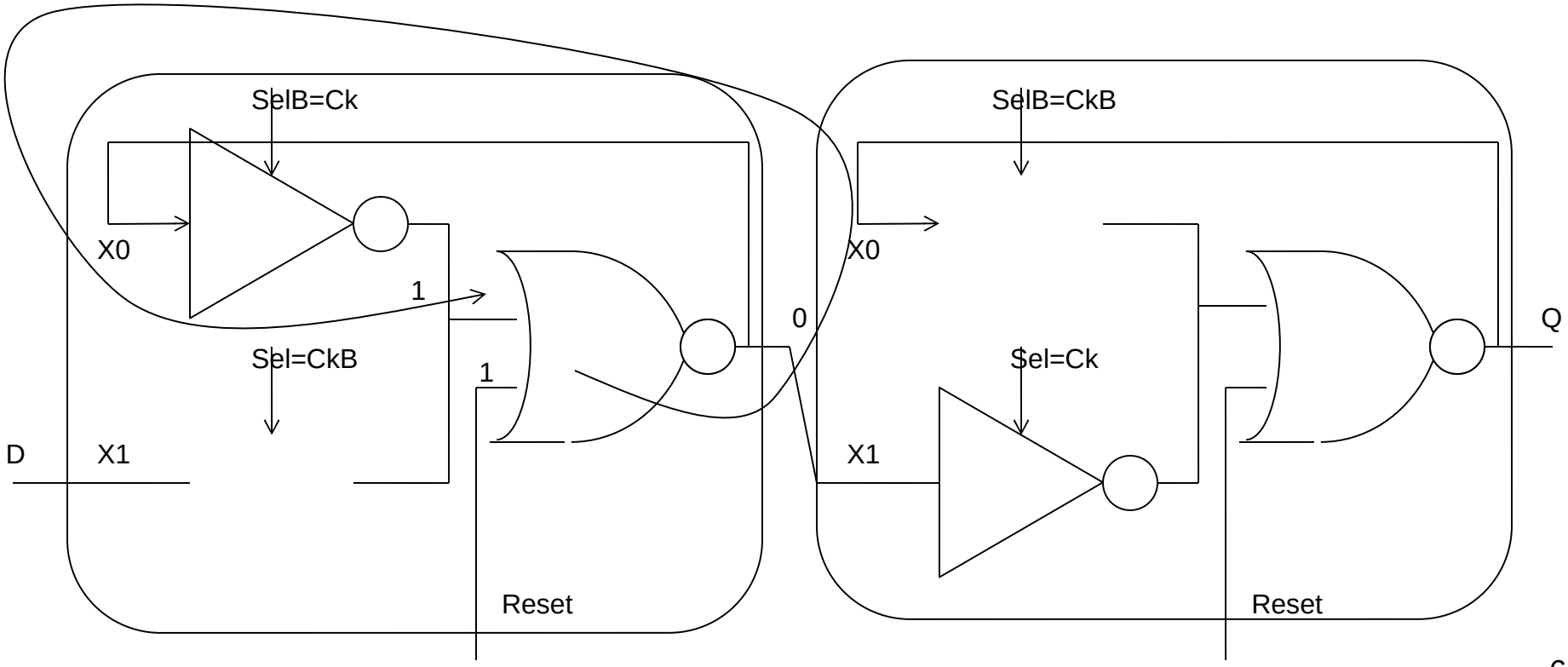
Noch Besser!



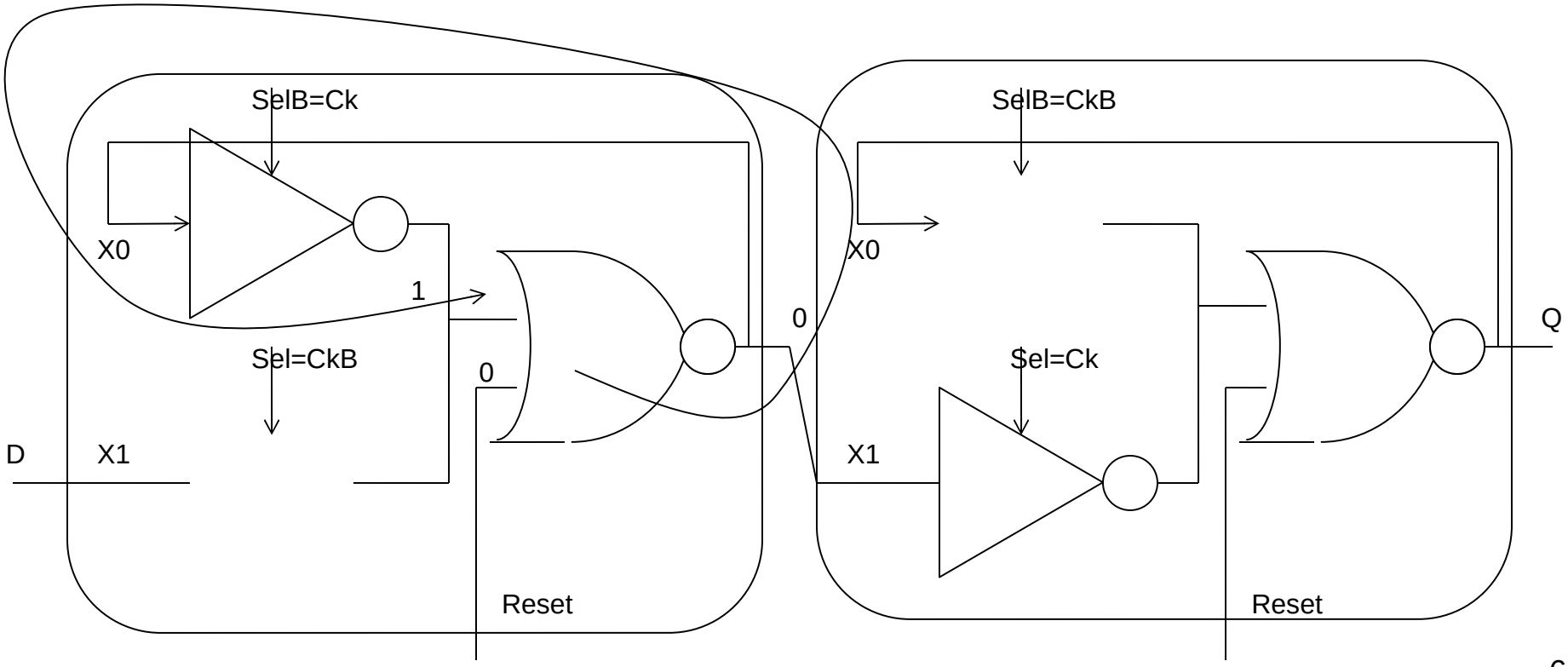
- Nach dem Einschalten der Spannungsversorgung befindet sich ein Flip-Flop, genauso wie eine RAM Zelle, in einem unbekannten logischen Zustand.
- Wir können uns vorstellen, dass sich zuerst alle Flip-Flops in den astabilen Zustand befinden und dann in logisch Eins oder Null Zustand kommen.
- Um einen unbekannten Anfangszustand zu vermeiden, werden die Flip-Flops oft so erweitert, dass sie ein asynchrones Reset Signal haben.



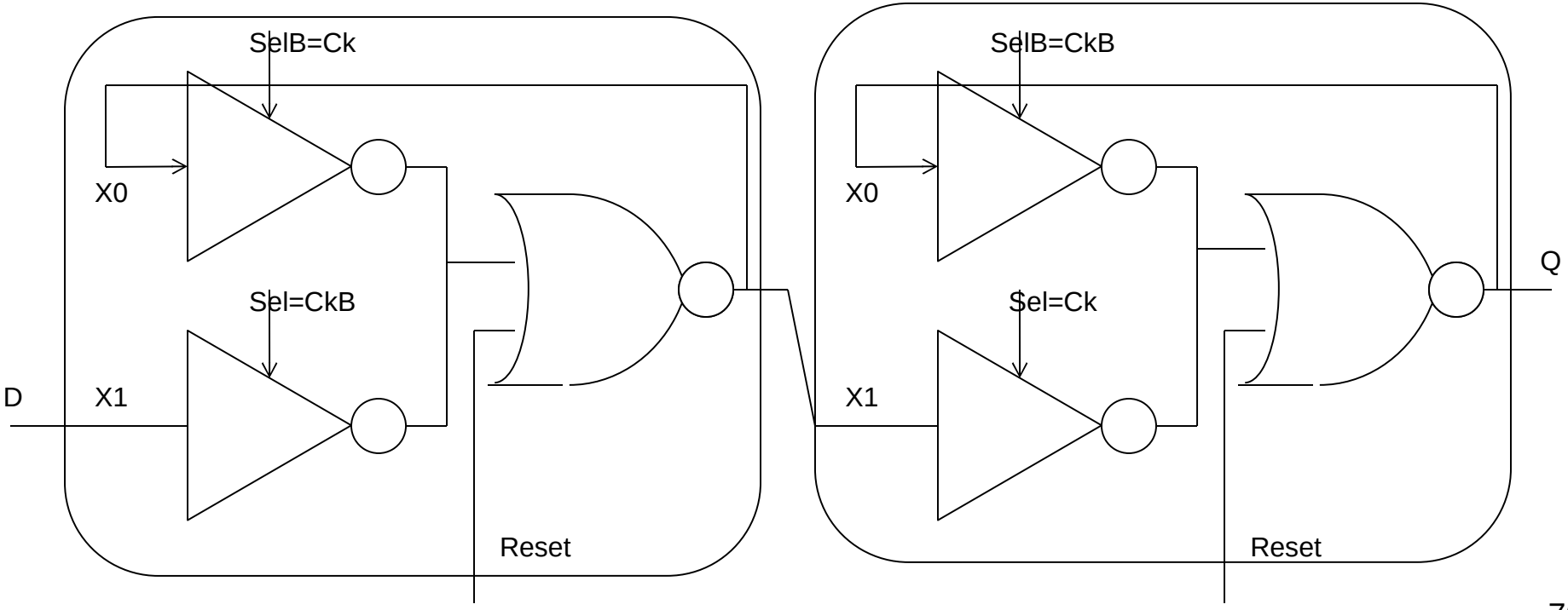
- Betrachten wir einen solch erweiterten FF im Ck=0 Zustand
- In dem fall ist das erste Latch im Speichermodus
- Reset = 1 (aktiv high) erzwingt logische Null am Ausgang, sie wird rückgekoppelt. (Eins kommt an den zweiten NOR Eingang)
- Auf diese Weise bleibt Null gespeichert auch wenn Reset wieder inaktiv (null) wird.



- Betrachten wir einen solch erweiterten FF im Ck=0 Zustand
- In dem fall ist das erste Latch im Speichermodus
- Reset = 1 (aktiv high) erzwingt logische Null am Ausgang, sie wird rückgekoppelt. (Eins kommt an den zweiten NOR Eingang)
- Auf diese Weise bleibt Null gespeichert auch wenn Reset wieder inaktiv (null) wird.

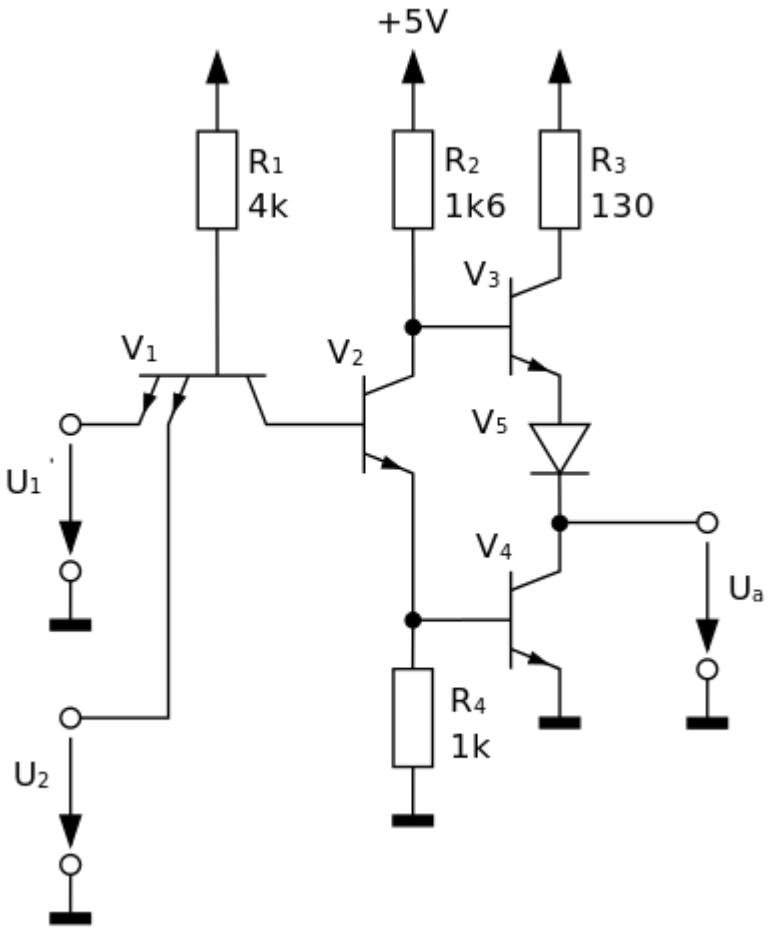


- Im Flip-Flop ist immer wenigstens ein Latch im Speicherzustand, so dass ein Reset immer möglich ist wenn beide Latches die Reset Logik enthalten.
- Asynchron Reset ist *stärker* als der Takteingang. Sobald Reset = 1 wird, wird der Flip-Flop Ausgang null, unabhängig von D und Ck Eingängen



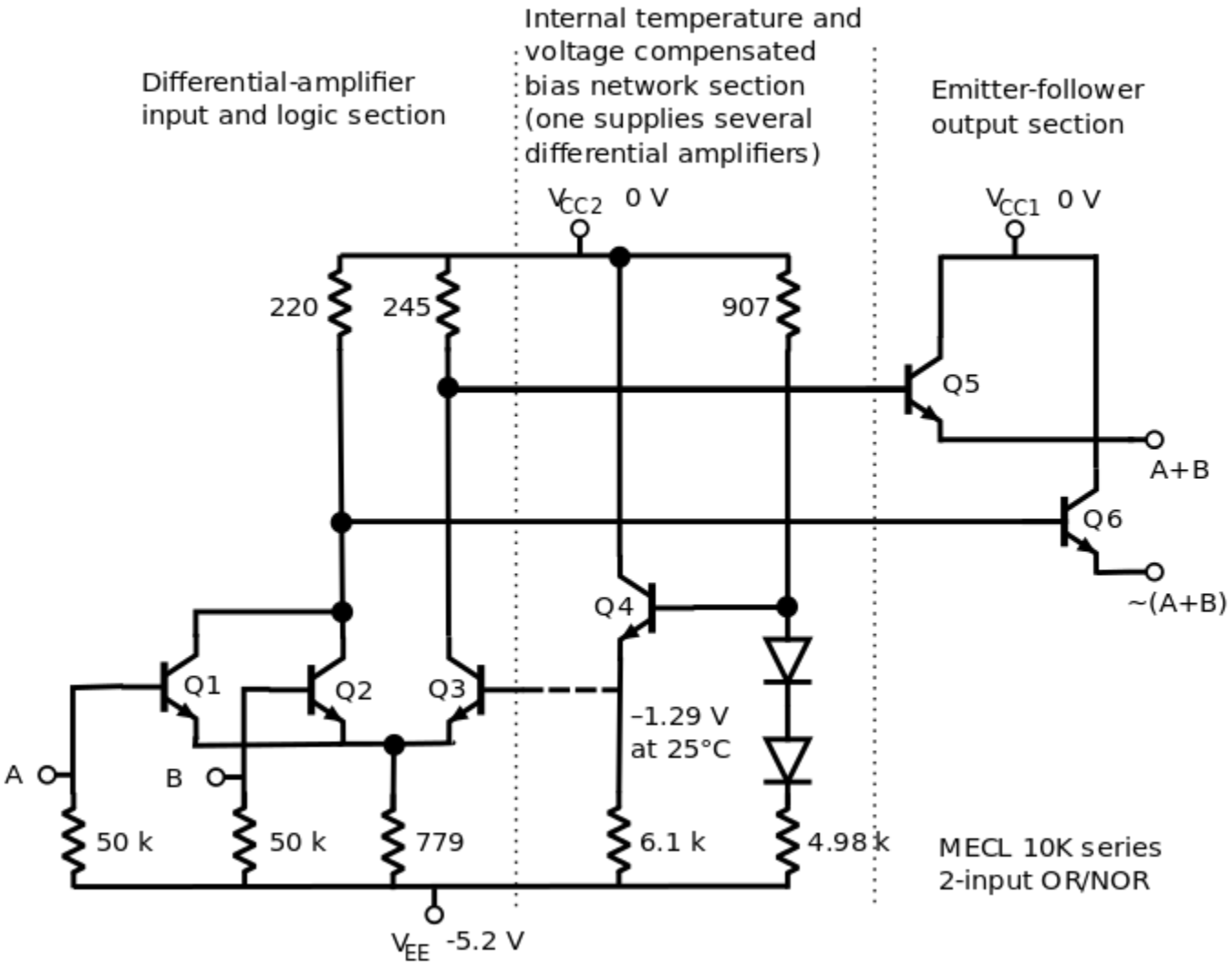
- Weitere Beispiele der Logikelementen

- TTL NAND

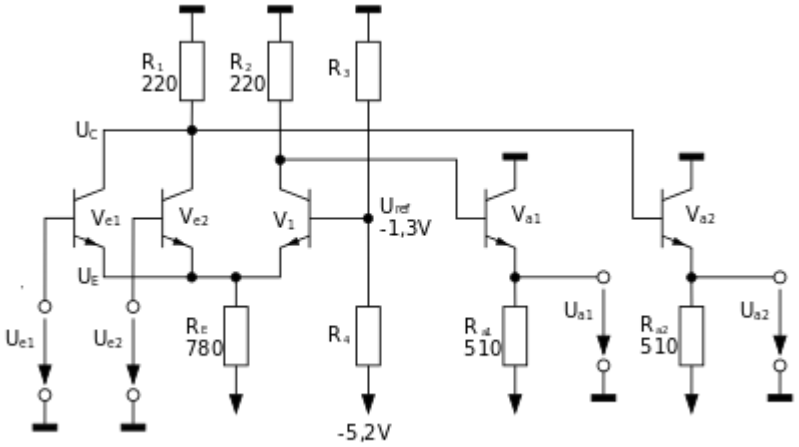


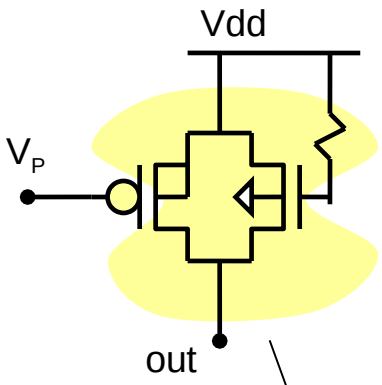
- Logische Bausteine in TTL-Technik haben gegenüber CMOS-Bausteinen den Vorteil, dass sie unempfindlicher gegenüber elektrostatischen Entladungen sind. Der Nachteil liegt wegen der stromgesteuerten Transistoren in einer im Vergleich zu CMOS deutlich höheren Leistungsaufnahme (Stromverbrauch) bei statischem Betrieb.
- Eine Besonderheit von TTL-Schaltungen besteht darin, dass an Eingängen jedes Potential zwischen 0 V und 5 V liegen darf und sie daher auch unbeschaltet bleiben dürfen, ohne dass untolerierbar große Querströme entstehen. Eine Besonderheit einer diskret aufgebauten TTL-Schaltung besteht darin, dass unbeschaltete Eingänge wirken, als lägen sie auf High-Pegel.

- …

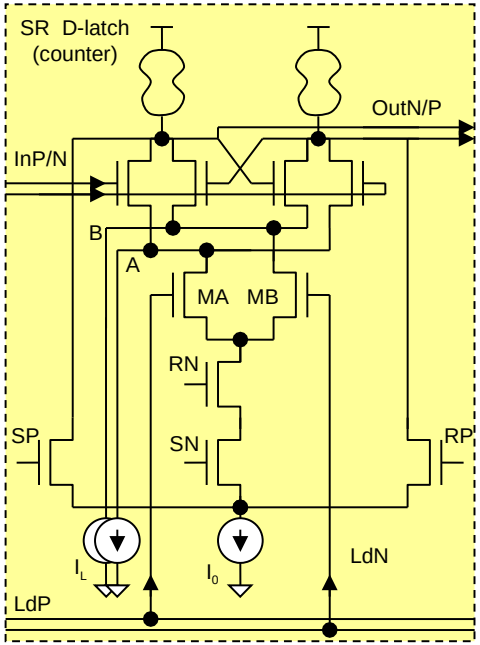

Differential-amplifier input and logic section
Internal temperature and voltage compensated bias network section (one supplies several differential amplifiers)
Emitter-follower output section
V_{CC2} 0 V
V_{CC1} 0 V
220
245
907
Q5
A+B
Q6
~(A+B)
Q4
Q1
Q2
Q3
–1.29 V at 25°C
A
B
50 k
50 k
779
6.1 k
4.98 k
MECL 10K series 2-input OR/NOR
V_{EE} -5.2 V

- Die ECL-Familie gehört zu den schnellsten erhältlichen Logikfamilien. Dies wird erreicht, da (anders als zum Beispiel bei der Transistor-Transistor-Logik) im normalen Betriebszustand kein Transistor in Sättigung geht.

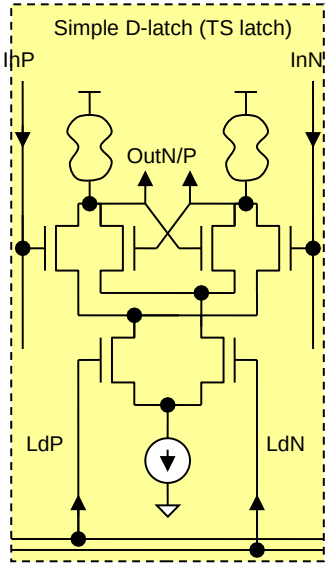




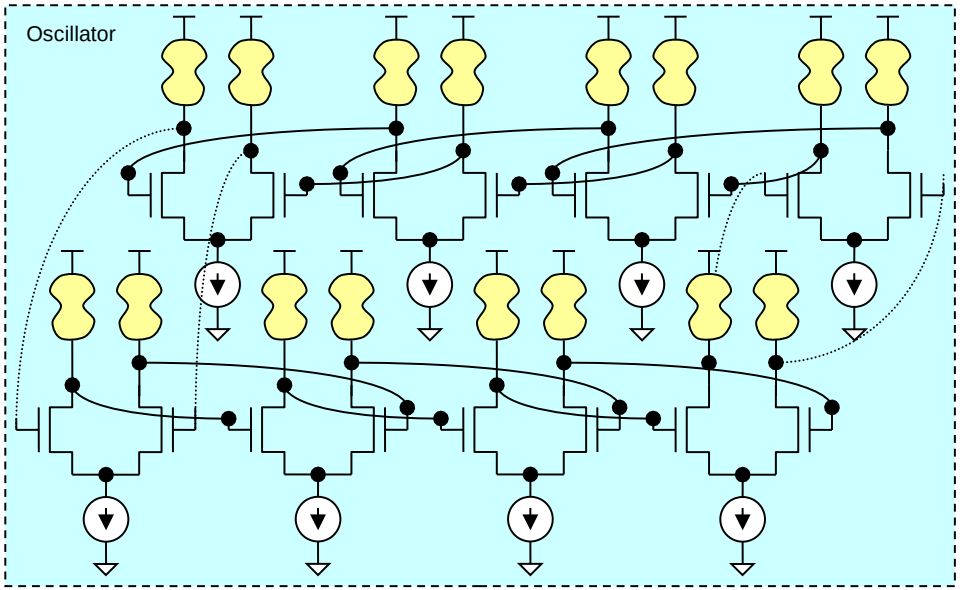
SR-Latch



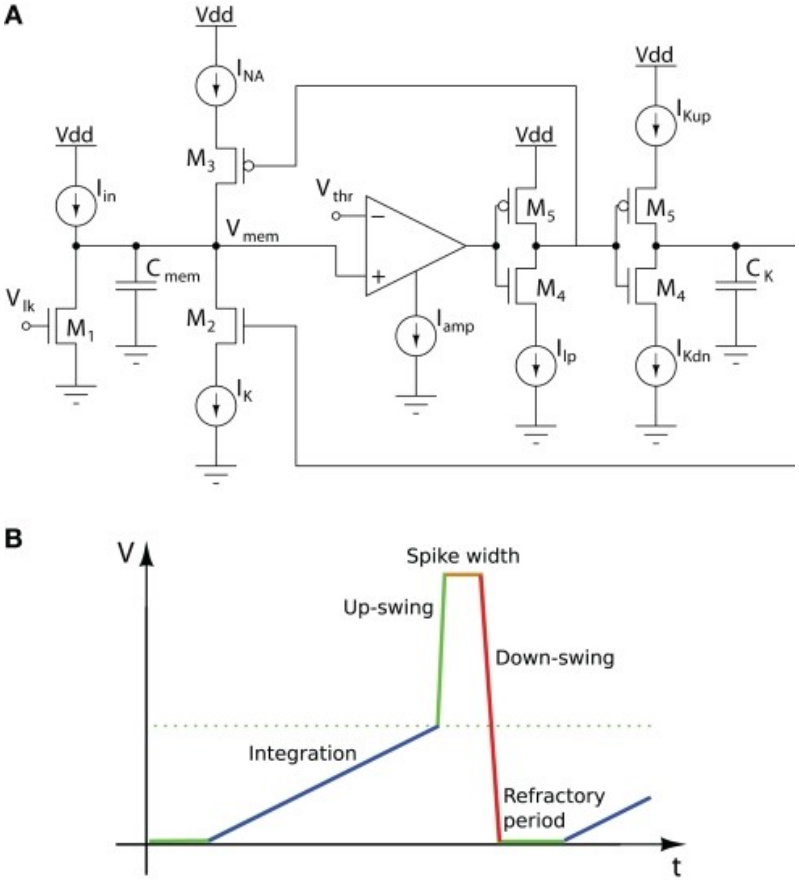
D-Latch



Ringoszillator



- Neuromorphic silicon neuron circuits



- https://www.youtube.com/watch?v=ZoT82NDpcvQ