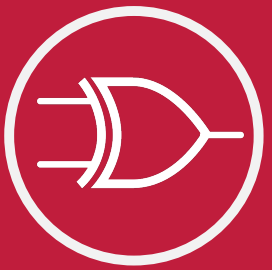
Technische Universität Braunschweig

Informatik für Ingenieure – VL 5

2. ENTWURF UND ANALYSE KOMBINATORISCHER SCHALTUNGEN



Übersicht

Letztes Mal:

- Minimierung von Schaltfunktionen
  - Karnaugh-Veith Mappen
  - Quine-McCluskey

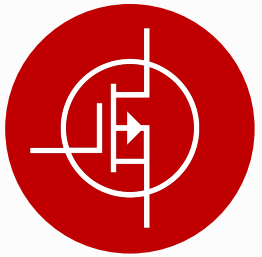
Heute:

- Einführung zu MOSFETS
- Einführung zu CMOS Logik
- Schaltnetzrealisierung

Teile des heutigen Vortrags basiert auf der Vorlesungen von
Prof. H Michalik (TU Braunschweig)
Prof. M. Luisier (ETH Zurich)

und die folgende Bücher
Fundamentals of Digital Logic with VHDL Design, von Brown, Vranesic
Digital Design and Computer Architecture, von Harris, Harris

CMOS Logik



Schaltfunktionen

Eine Schaltfunktion $\boldsymbol{f}$ nimmt mehrere Variablen $\boldsymbol{X_i}$, z.B. $\boldsymbol{N}$, als Eingang und produziert eine einzige Variable $\boldsymbol{Y}$ als Ausgang mit $\boldsymbol{X_i}$ **und** $\boldsymbol{Y} \in \{\mathbf{0}, \mathbf{1}\}$. Die X_i und Y Variablen sind **Bits** genannt

Jede Variable $\boldsymbol{X_i}$ kann durch einen Schalter dargestellt werden:

$X_i = 0$ $\qquad$ $X_i = 1$

Die Schaltfunktion $\boldsymbol{f}$ kann durch eine Kombination von Schaltern repräsentiert werden. Das Ergebnis ist **1**, wenn die am Ausgang **Y** gemessene Spannung, $\mathbf{U_Y}$, der Speisespannung entspricht.

Frage: wie kann man solche einfache Schaltungen elektrisch realisieren?

Mit **MOS-Transistoren** (Halbleiterbauelemente)!

Was sind MOS Transistoren?

MOS

Metal-Oxide-Semiconductor

Metall-Oxid-Halbleiter

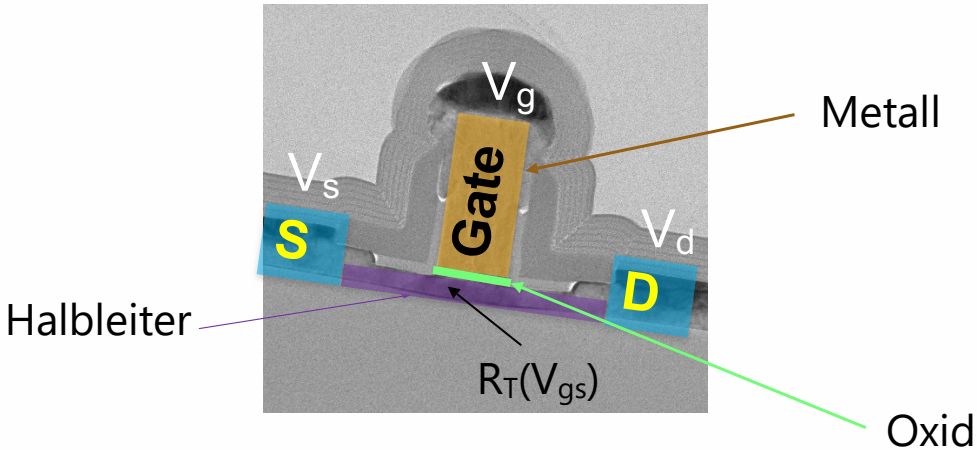
Transistor

Trans-Resistor

Steuerbarer Widerstand

Intel Single-Gate Transistor

R. Chau et al., *DRC*, (2003)



$R_T(V_{gs})$:
steuerbarer
Kanalwiderstand

MOS Transistoren sind elektronische Bauelemente:

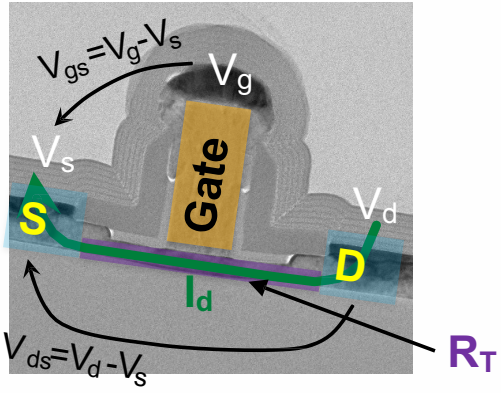
Sie besitzen 3 Kontakte (**S**ource, **D**rain und **Gate**)

Drei Spannungen können angelegt werden: V_s, V_d und V_g

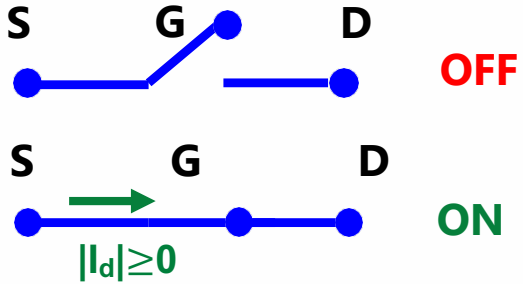
Ladungsträger (Elektron/Loch) fliessen von Source nach Drain

Wie funktionieren MOS Transistoren?

Intel Single-Gate Transistor
R. Chau et al., *DRC*, (2003)



$|V_{gs}|<|V_{th}|$ (Schwellspannung): Schalter auf



$|V_{gs}|>|V_{th}|$ (Schwellspannung): Schalter zu

MOS Transistoren verhalten sich wie Schalter:

Der Kanalwiderstand **R_T** wird durch V_{gs} gesteuert

Wenn **$|V_{gs}|<|V_{th}|$** (Schwellspannung) ist der Kanal gesperrt (hochohmig, $R_T \rightarrow \infty$), der Schalter ist **auf**, es fliesst kein Strom

Wenn **$|V_{gs}|>|V_{th}|$** (Schwellspannung) ist der Kanal leitend ($R_T \rightarrow 0$), der Schalter ist **zu**, ein Strom **I_d** fliesst (wenn $|V_{ds}|>0$)

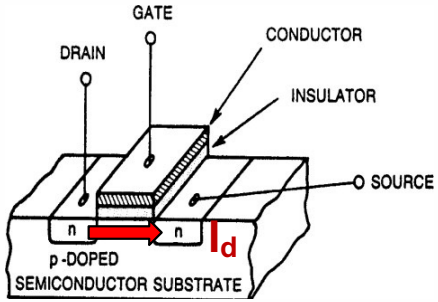
Complementary MOS (CMOS) Technologie (1)

N-Typ (NMOS)

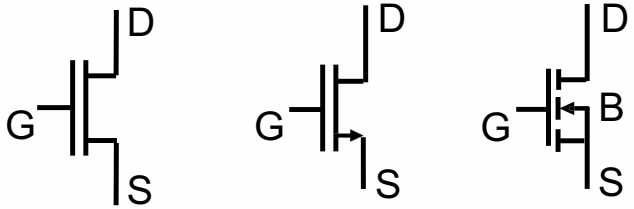
NMOS-Transistor

Transistor Struktur

N: negative Ladungen (Elektronen)



Schaltsymbol



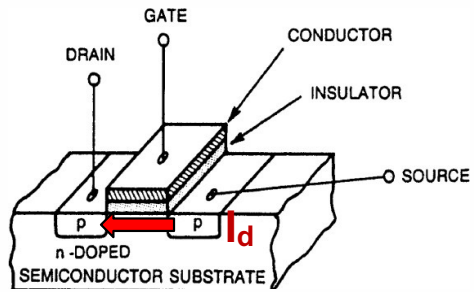
Source (S), Drain (D), Gate (G), Bulk (B)

P-Typ (PMOS)

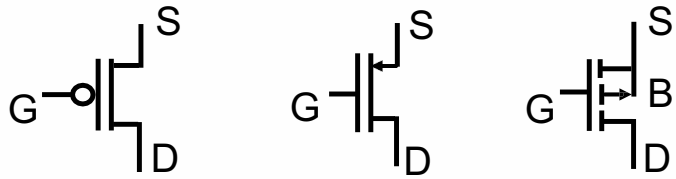
PMOS-Transistor

Transistor Struktur

P: positive Ladungen ('Löcher')



Schaltsymbol

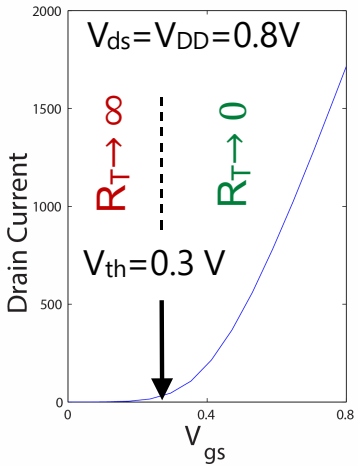


Source (S), Drain (D), Gate (G), Bulk (B)

Complementary MOS Technologie (2)

$R_T(V_{gs})$: Kanalwiderstand

N-Typ (NMOS)

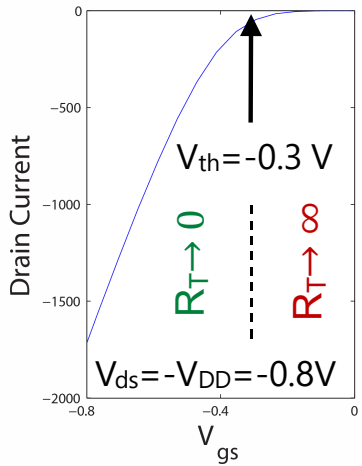


Statische I-V Kennlinien

Drain Strom I_d in Abhängigkeit von V_{gs}, wenn die Speisespannung V_{DD} zwischen Drain und Source angelegt ist

I_d positiv, Schwellspannung V_{th} positiv

P-Typ (PMOS)



Statische I-V Kennlinien

Drain Strom I_d in Abhängigkeit von V_{gs}, wenn die Speisespannung $-V_{DD}$ zwischen Drain und Source angelegt ist

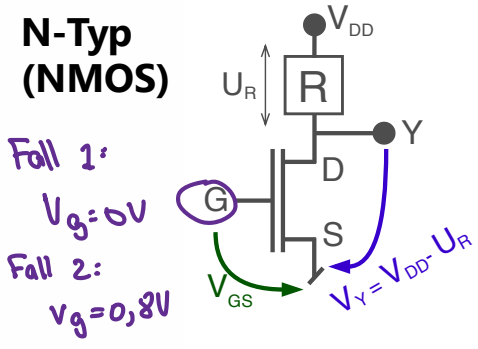
I_d negativ, Schwellspannung V_{th} negativ

Annahmen: wenn $|V_{gs}| \leq |V_{th}|$, Kanalwiderstand **$R_T \rightarrow \infty$**: Entsprechender Schalter **auf**

wenn $|V_{gs}| > |V_{th}|$, Kanalwiderstand **$R_T \rightarrow 0$**: Entsprechender Schalter **zu**

Complementary MOS Technologie (3)

N-Typ (NMOS)



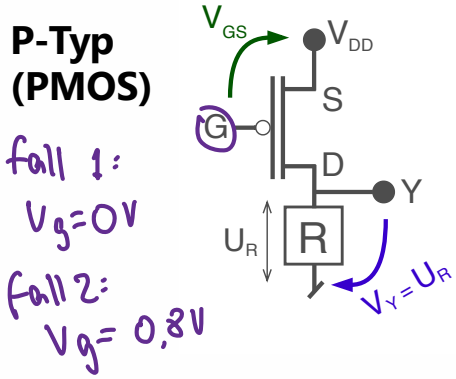
Gesucht:

Wert von Spannung $\mathbf{V_Y}$ (Ausgang) in Abhängigkeit von $\mathbf{V_g}$ (Eingang)

R: Serie Widerstand

$\mathbf{V_{gs}}$: V_g-V_s mit V_s=0

P-Typ (PMOS)



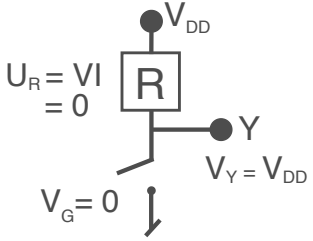
Gesucht:

Wert von Spannung $\mathbf{V_Y}$ (Ausgang) in Abhängigkeit von $\mathbf{V_g}$ (Eingang)

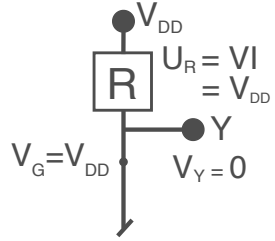
R: Serie Widerstand

$\mathbf{V_{gs}}$: V_g-V_s mit V_s=V_{DD}

Pull-down Schaltung (statisch)

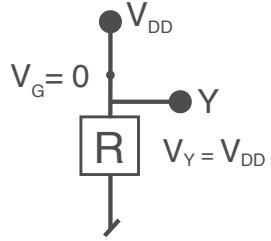


V_{gs}=0, T inaktiv
→ kein Strom
→ V_Y=V_{DD}

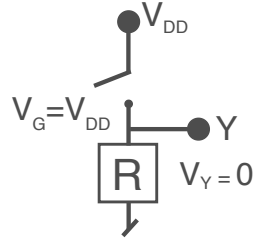


V_{gs}=V_{DD}, T aktiv
→ I_d=V_{DD}/R
→ **V_Y=0**

Pull-up Schaltung (statisch)



V_{gs}=-V_{DD}, T aktiv
→ I_d=V_{DD}/R
→ **V_Y=V_{DD}**



V_{gs}=0, T inaktiv
→ kein Strom
→ V_Y=0

CMOS Zusammenfassung:

Für eine Gatespannung $|V_{gs}|$ **kleiner** als die Schwell-spannung $|V_{th}|$

Kanal zwischen Source und Drain abgeschnürt, also elektrisch nicht leitend (gesperrt, inaktiv) → **Schalter offen (kein Strom)**

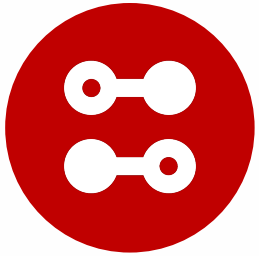
Spannungwerte (V_{th}~0.3 V und V_{DD}=0.8 V) hängen von verwendete Technologie ab

Für $|V_{gs}|$ **grösser** als $|V_{th}|$ ist der Kanal zwischen Source und Drain niederohmig, also elektrisch leitend (aktiv) → **Schalter geschlossen oder zu (Strom)**

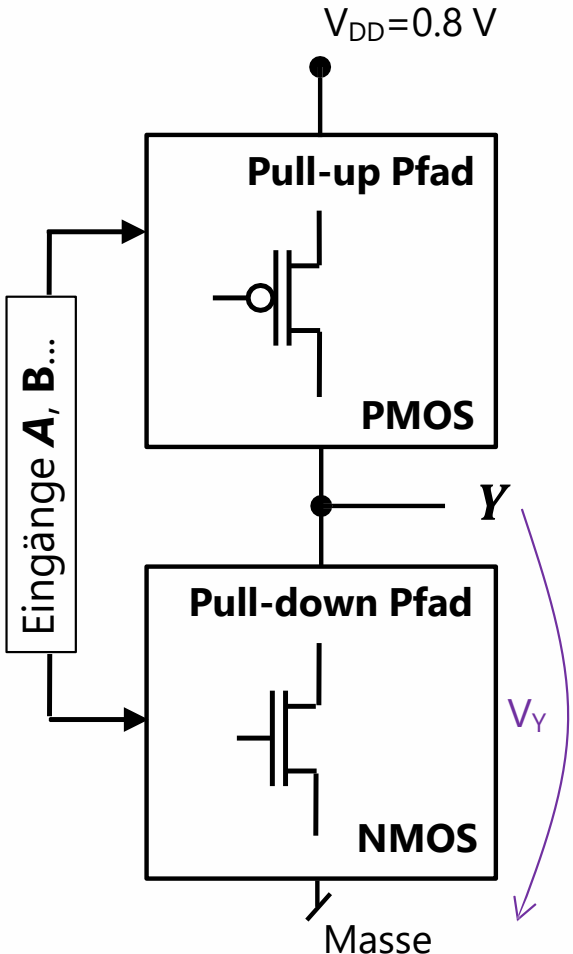
In Schaltungen, wenn **PMOS** Transistoren aktiv sind, dann ist der Ausgang mit dem **High Pegel** gebunden (**Pull-up**)

In gleicher Weise, wenn **NMOS** Transistoren aktiv sind, dann ist der Ausgang mit dem **Low Pegel** gebunden (**Pull-down**)

CMOS Schaltungen



Pull-up / Pull-down Prinzip



Konstruktion von gattern durch kombination von nmos- und pmos-transistoren

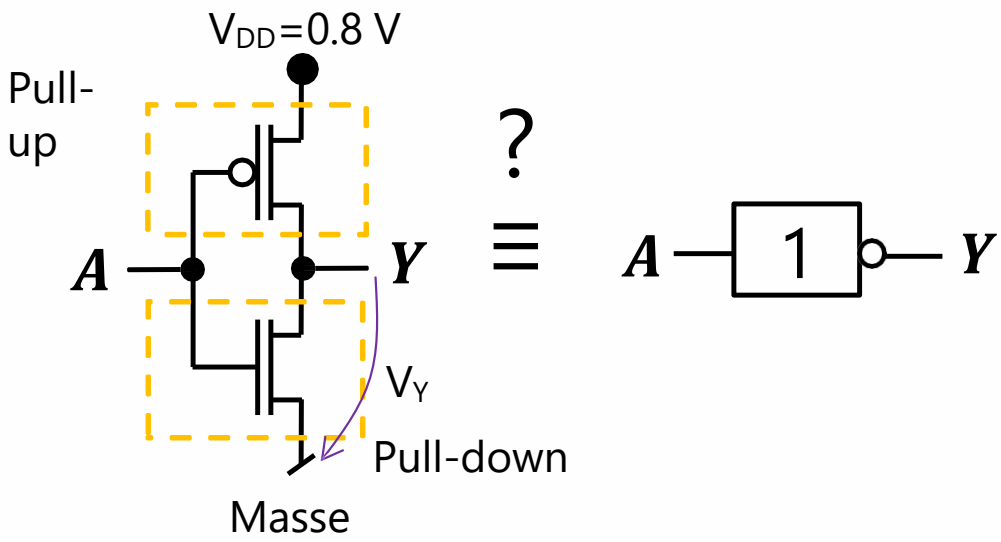
Bei ***m*** Eingängen gibt es ***m*** NMOS und ***m*** PMOS Transistoren

Wenn NMOS Transistoren in **Serie** geschaltet sind (UND-Verknüpfung), dann sind die entsprechenden PMOS **parallel** angeordnet

Wenn NMOS Transistoren **parallel** geschaltet sind (ODER-Verknüpfung), dann sind die entsprechenden PMOS in **Serie** angeordnet

NICHT Gatter in CMOS Technik (CMOS Inverter)

Schaltbild eines NICHT-Gatters mit einem Eingang **A** und Ausgang **Y**

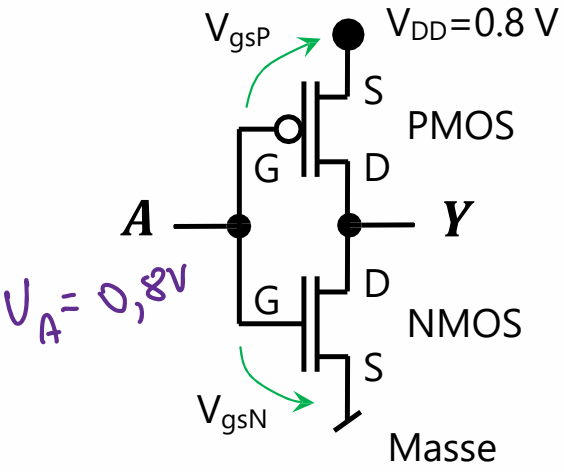


Wie sieht die Wahrheitstabelle von diesem Schaltbild aus?

A	Y
0	1
1	0

Analysieren wir die einzelnen Fälle…

NICHT-Schaltbild Analyse: 1. Fall



$U_A = 0{,}8V$

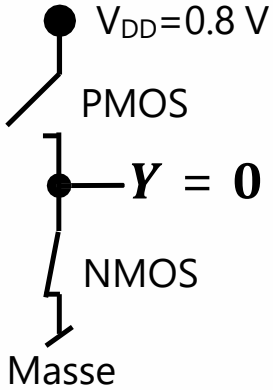
Fall 1: A=1 (High Pegel, V =0.8 V)

→V_{gsP}=0 V, PMOS nicht leitend (Schalter auf)

→ V_{gsN}=0.8 V, NMOS leitend (Schalter zu)

→Ausgang mit der Masse gebunden **Y=0**

Schalterlogik



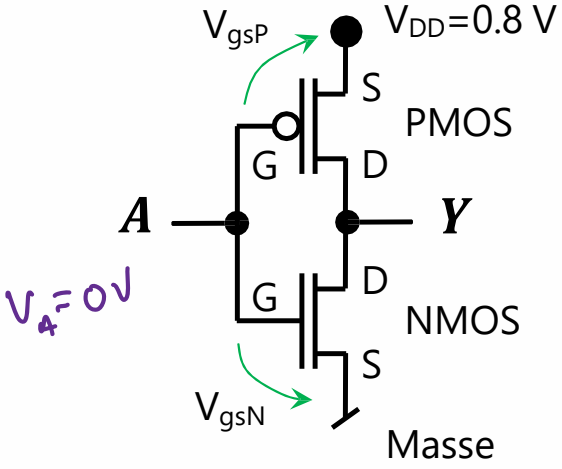
Spannungen

A	PMOS	NMOS	Y
0.8 V	$\|V_{gs}\|<\|V_{th}\|$	$V_{gs}>V_{th}$	0 V

Wahrheitstabelle

A	PMOS	NMOS	Y
1	offen/auf	zu	0

NICHT-Schaltbild Analyse: 2. Fall



Fall 2: A=0 (Low Pegel, V_A=0 V)

→ V_{gsP} =-0.8 V, PMOS leitend (Schalter zu)

→ V_{gsN}=0 V, NMOS nicht leitend (Schalter auf)

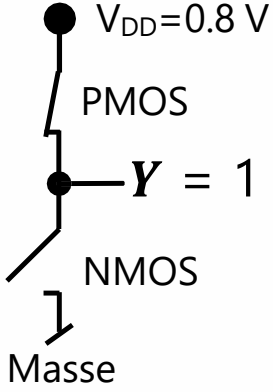
→ Ausgang mit V_{DD} gebunden **Y=1**

Wahrheitstabelle

A	PMOS	NMOS	Y
0	zu	offen	1
0 V	$\|V_{gs}\|>\|V_{th}\|$	$V_{gs}<V_{th}$	0.8 V

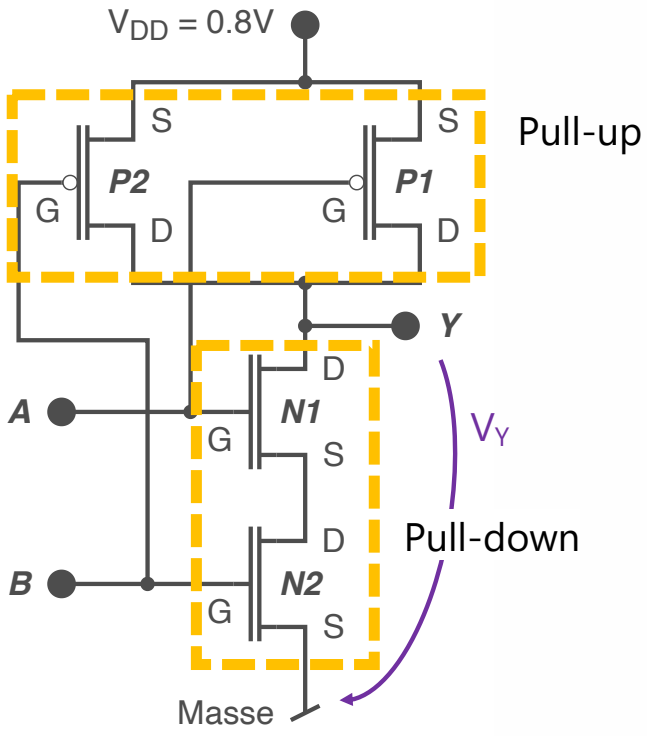
zu: leitend / offen (oder auf): nicht leitend

Schalterlogik



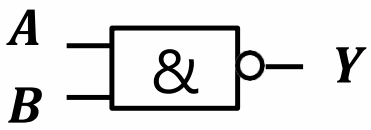
NAND-Gatter in CMOS Technik

Schaltbild eines NAND-Gatters mit zwei Eingängen **A** und **B** und einem Ausgang **Y**



?

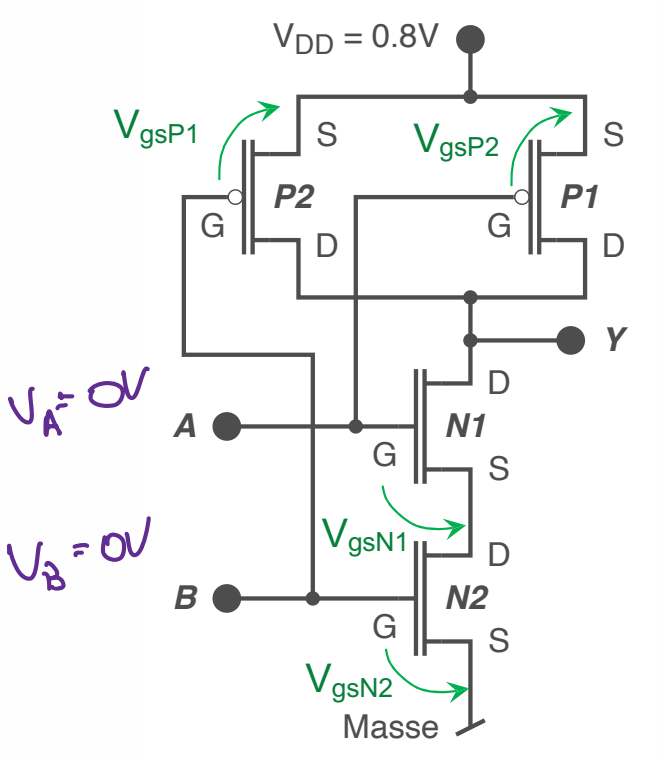
≡



Kann diese Schaltung die **NAND** Wahrheitstabelle korrekt liefern?

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

Analyse vom CMOS NAND-Gatter: 1. Fall

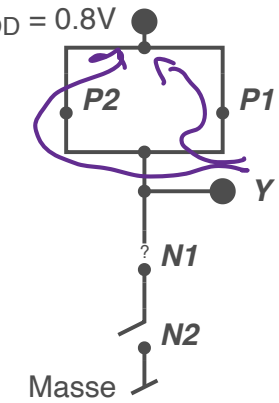


Fall 1: **A=0** (Low Pegel, V_A=0 V),
B=0 (Low Pegel, V_B=0 V)

→ V_{gsP1}=-0.8 V, V_{gsP2}=-0.8 V,
V_{gsN1} unbestimmt, V_{gsN2}=0 V

→ P1 leitend, P2 leitend,
N1 unbestimmt, N2 gesperrt

→ Ausgang mit V_{DD} gebunden **Y=1**

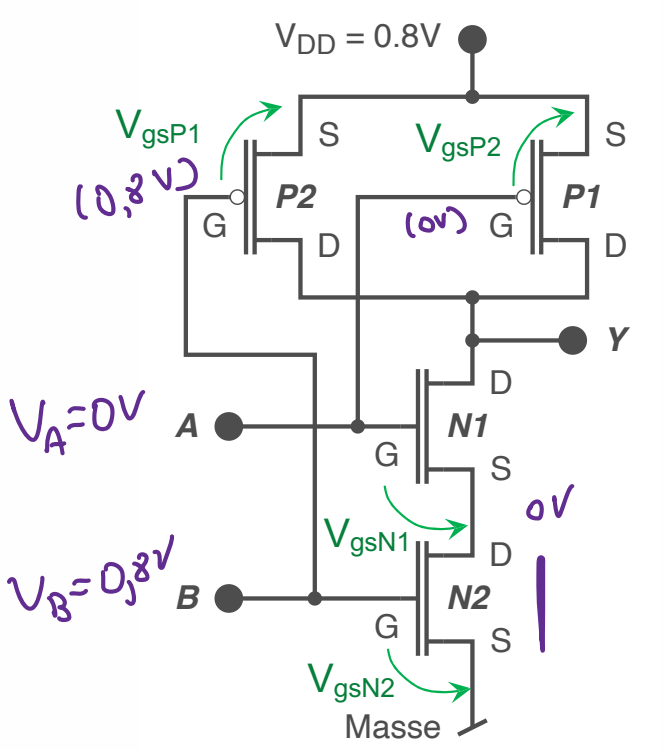


Wahrheitstabelle (Schalterlogik)

A	B	P1	P2	N1	N2	Y
0	0	zu	zu	NN	offen	1

zu: leitend / offen: nicht leitend / NN: unbestimmt

Analyse vom CMOS NAND-Gatter: 2. Fall

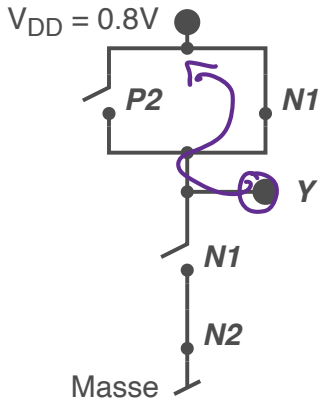


Fall 2: **A=0** (Low Pegel, V_A=0 V),
B=1 (High Pegel, V_B=0.8 V)

→ V_{gsP1}=0 V, V_{gsP2}=-0.8 V,
V_{gsN1}=0 V, V_{gsN2}=0.8 V

→ P1 gesperrt, P2 leitend,
N1 gesperrt, N2 leitend

→ Ausgang mit V_{DD} gebunden **Y=1**

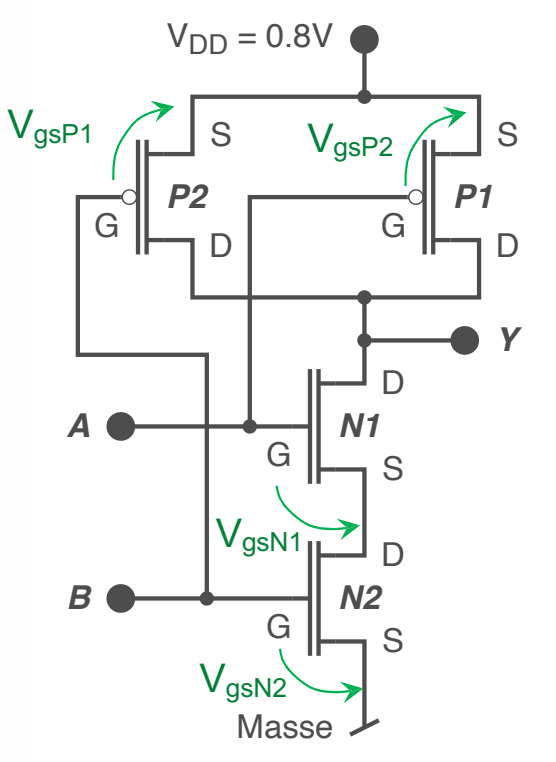


Wahrheitstabelle (Schalterlogik)

A	B	P1	P2	N1	N2	Y
0	1	offen	zu	offen	zu	1

zu: leitend / offen: nicht leitend / NN: unbestimmt

Analyse vom CMOS NAND-Gatter: 3. Fall

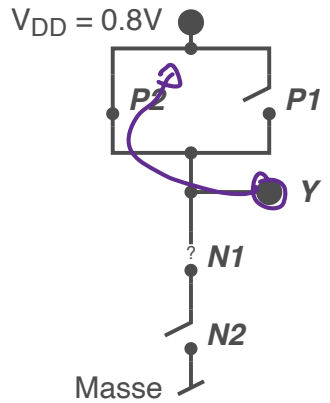


Fall 3: **A=1** (High Pegel, V_B=0.8 V),
B=0 (Low Pegel, V_A=0 V)

→ V_{gsP1}=-0.8 V, V_{gsP2}=0 V,
V_{gsN1} unbestimmt, V_{gsN2}=0 V

→ P1 leitend, P2 gesperrt,
N1 unbestimmt, N2 gesperrt

→ Ausgang mit V_{DD} gebunden **Y=1**

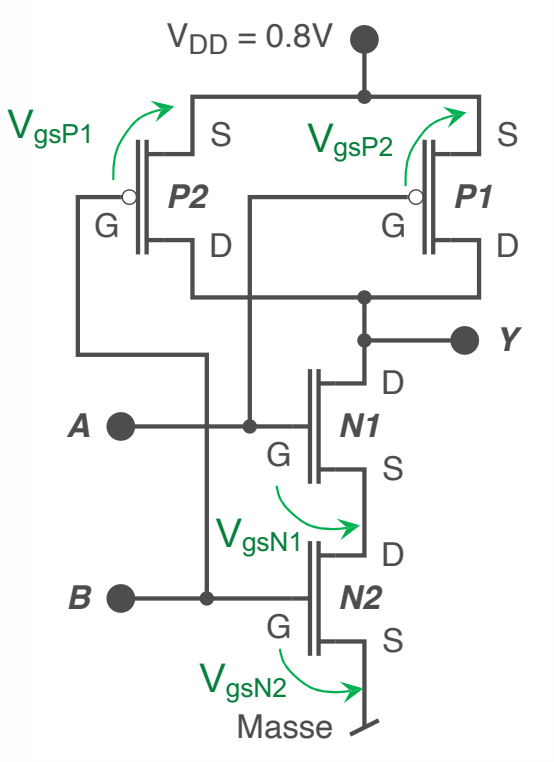


Wahrheitstabelle (Schalterlogik)

A	B	P1	P2	N1	N2	Y
1	0	zu	offen	NN	offen	1

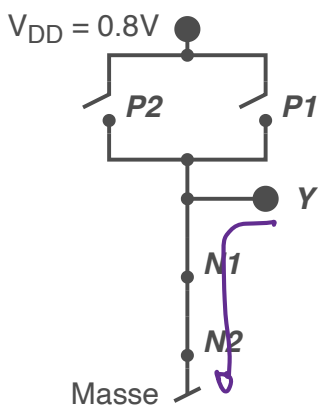
zu: leitend / offen: nicht leitend / NN: unbestimmt

Analyse vom CMOS NAND-Gatter: 4. Fall



Fall 4: **A=1** (High Pegel, V_B=0.8 V),
B=1 (High Pegel, V_B=0.8 V)

→V_{gsP1}=0 V, V_{gsP2}=0 V,
V_{gsN1}=0.8 V, V_{gsN2}=0.8 V
→ P1 gesperrt, P2 gesperrt,
N1 leitend, N2 leitend
→ Ausgang mit V_{DD} gebunden **Y=0**

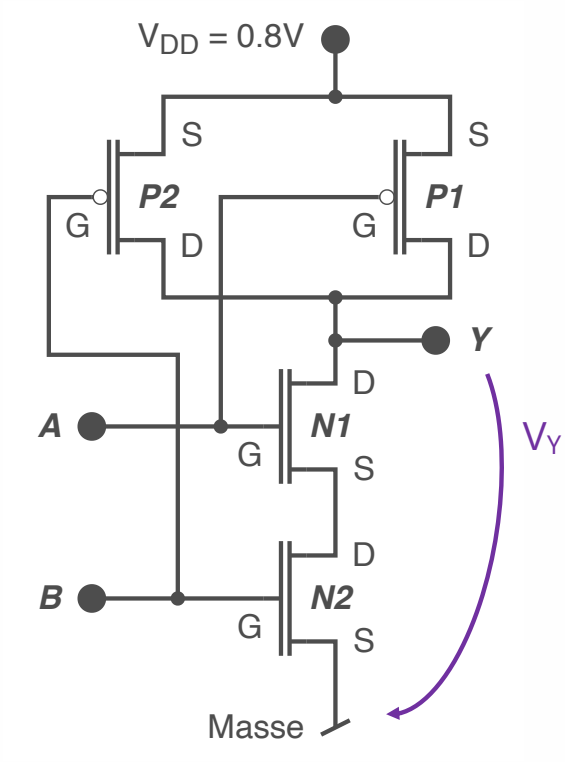


Wahrheitstabelle (Schalterlogik)

A	B	P1	P2	N1	N2	Y
1	1	offen	offen	zu	zu	0

zu: leitend / offen: nicht leitend

NAND-Gatter in CMOS Technik: Zusammenfassung



Mit der entworfenen CMOS Schaltung wird die gewünschte Wahrheitstabelle realisiert

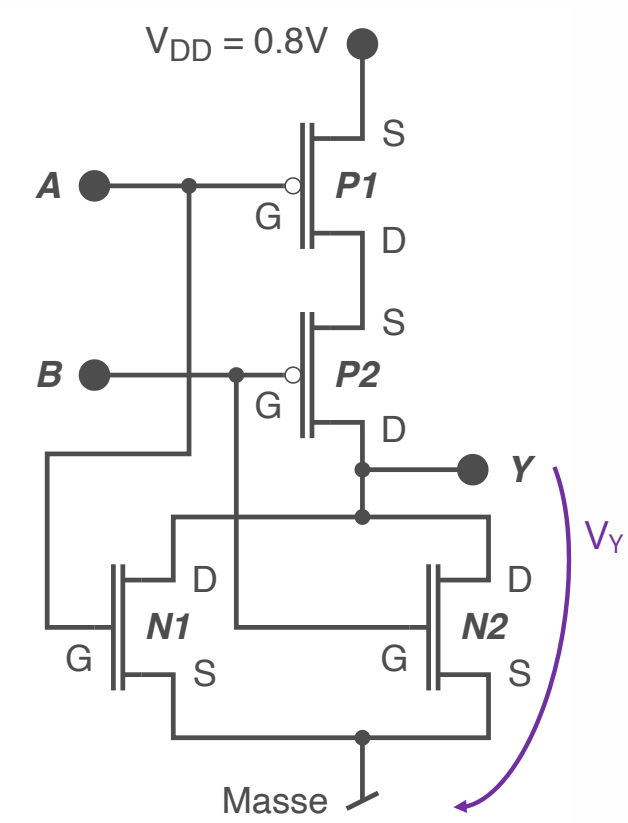
Wahrheitstabelle (Schalterlogik)

A	B	P1	P2	N1	N2	Y
0	0	zu	zu	NN	offen	1
0	1	offen	zu	offen	zu	1
1	0	zu	offen	NN	offen	1
1	1	offen	offen	zu	zu	0

zu: leitend / offen: nicht leitend / NN: unbestimmt

Nur wenn **A=B=1** ist der Ausgang **Y** mit der Masse gebunden

NOR-Gatter in CMOS Technik



Nehmen Sie sich ein paar Minuten um diese Schaltung zu analysieren!

Wie sieht ihre Wahrheitstabelle aus?

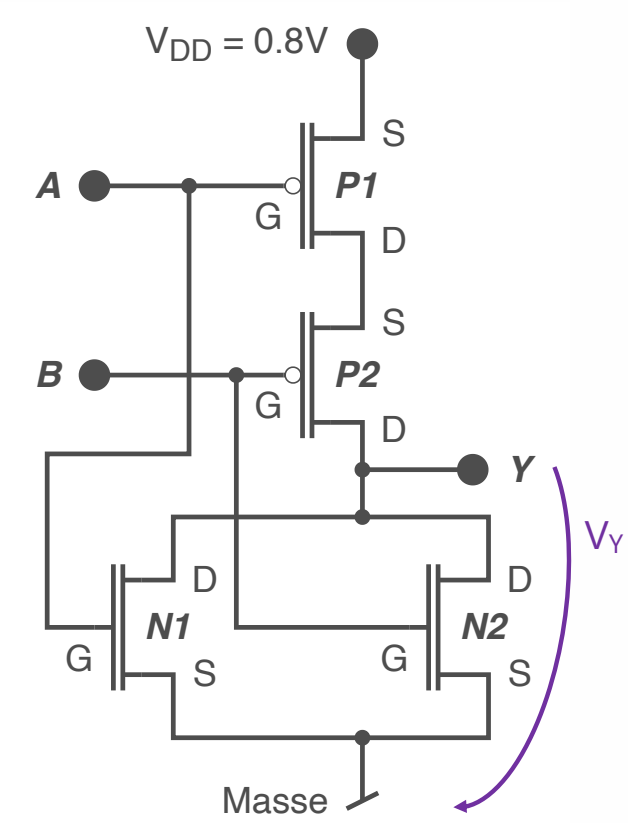
Wahrheitstabelle (Schalterlogik)

A	B	P1	P2	N1	N2	Y
0	0					
0	1					
1	0					
1	1					

zu: leitend / offen: nicht leitend / NN: unbestimmt

Der Ausgang **Y** ist entweder mit V_{DD} oder der Masse gebunden

NOR-Gatter in CMOS Technik



Nehmen Sie sich ein paar Minuten um diese Schaltung zu analysieren!

Wie sieht ihre Wahrheitstabelle aus?

Wahrheitstabelle (Schalterlogik)

A	B	P1	P2	N1	N2	Y
0	0	zu	zu	offen	offen	1
0	1	zu	offen	offen	zu	0
1	0	offen	NN	zu	offen	0
1	1	offen	NN	zu	zu	0

zu: leitend / offen: nicht leitend / NN: unbestimmt

Der Ausgang **Y** ist entweder mit V_{DD} oder der Masse gebunden

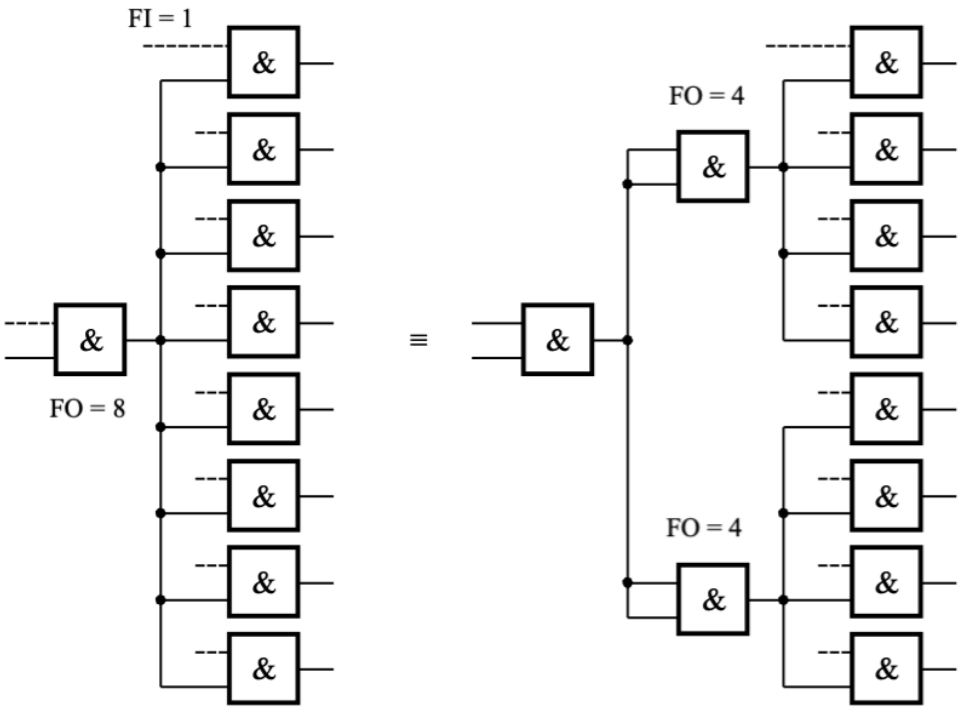
Fan-Out & Fan-In

FAN-OUT (FO, Ausfächerungsfaktor, Ausgangslastfaktor) ist die normierte Ausgangsbelastbarkeit.

Die Normierung bezieht sich auf den Strom, den ein Eingang einer einfachen Schaltung, um eine „1" oder „0“ richtig darstellen zu können.

Gatter, deren Eingänge eine größere Last darstellen werden mit einem "FAN-IN” (FI) > 1 gekennzeichnet.

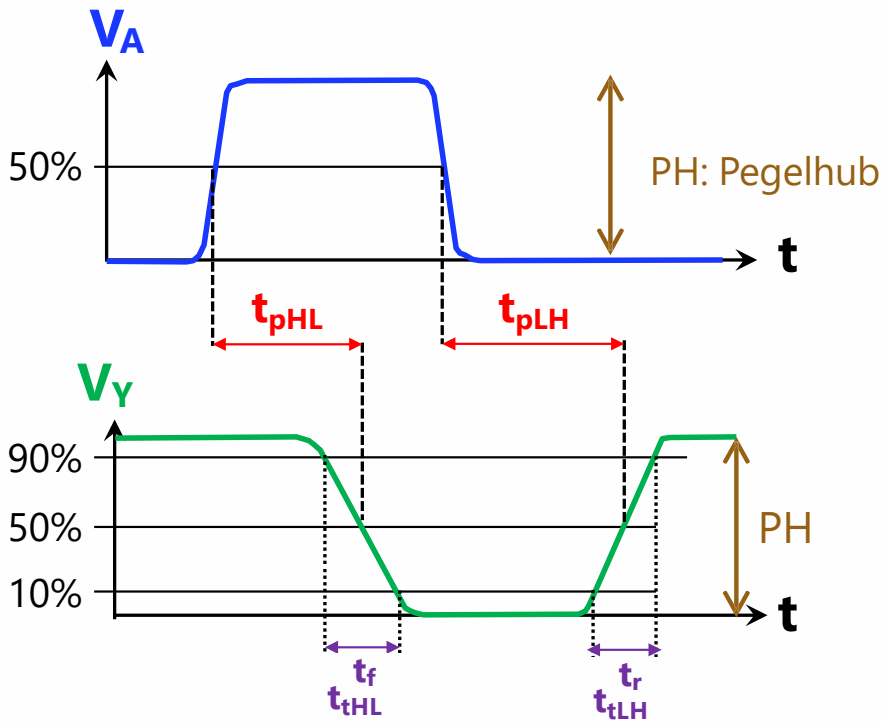
FanOut muss größer gleich sein als die Summe aller FanIn

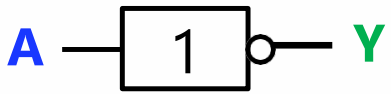


Wie schnell schalten CMOS Gatter?

CMOS Gatter können nicht unendlich schnell schalten.

Ladungen (Elektronen und Löcher) müssen bewegt werden, was Laufzeit Verzögerungen verursacht

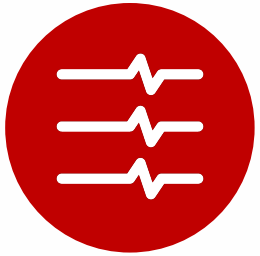




Zeitlaufdiagram eines CMOS Inverters im dynamischen (zeitabhängigen) Betrieb

Die t_{pHL} und t_{pLH} Verzögerungszeiten hängen von den Transistoreigenschaften ab

Glitches



Störimpulse (glitches)

Störimpulse - Eine Änderung eines Eingangs verursacht mehrere Änderungen des Ausgangs

Können durch geeignete Entwurfsdisziplin entschärft werden

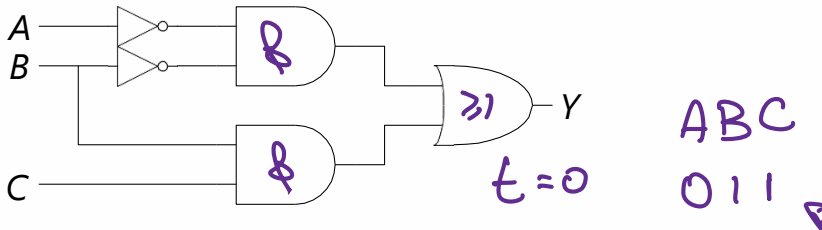
- Können noch auftreten, richten aber keinen Schaden an
- Bei Synchroner Entwurf (später) kann Ausnahmen geben

Sollten aber im Vorfeld erkannt werden

- Sichtbar im Timing-Diagram

Störimpulse Beispiel

Was passiert, wenn A = 0, C = 1, und B fällt von 1→0?

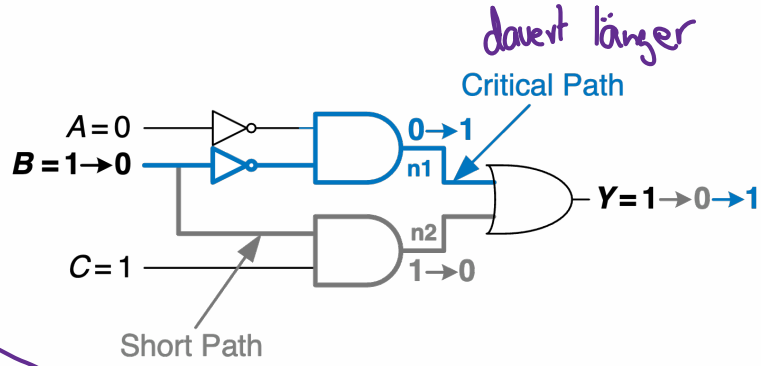


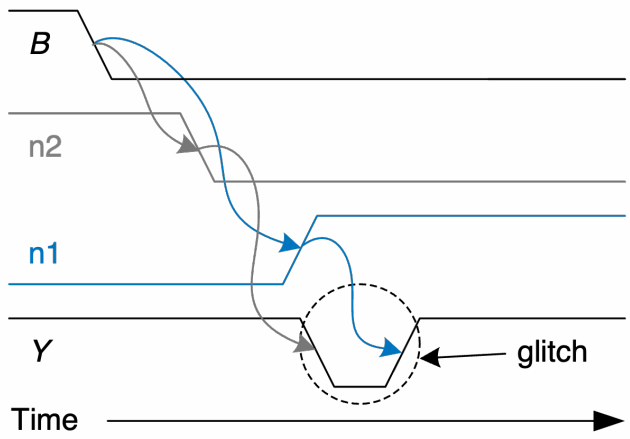
ABC
011
001

Y	AB 00	01	11	10
C 0	1	0	0	0
1	1	1	1	0

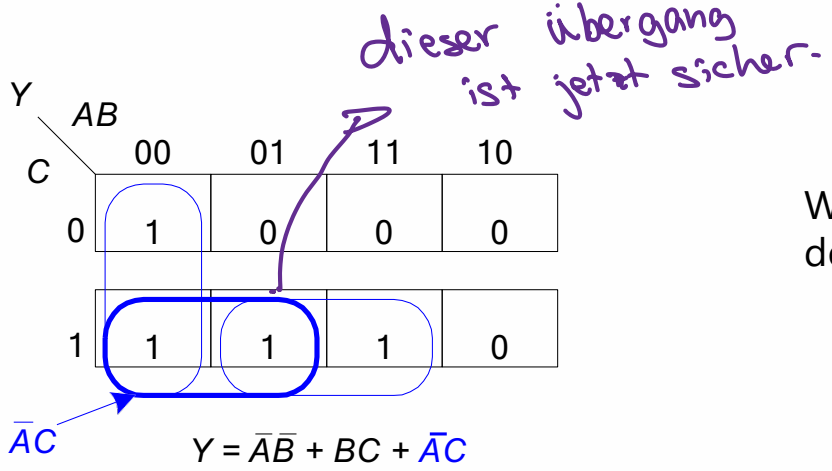
$Y = \overline{A}\,\overline{B} + BC$

dauert länger

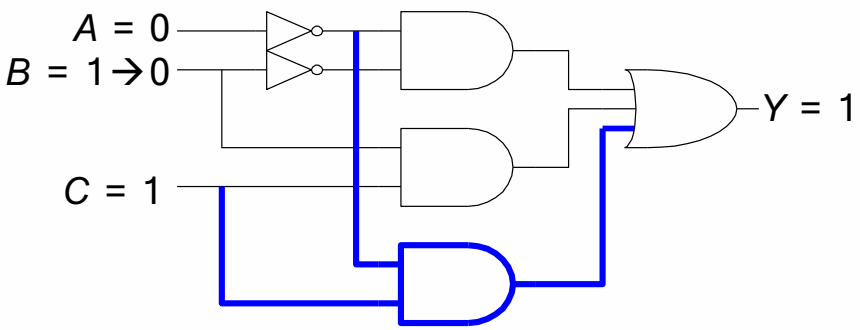




Störimpuls beseitigen



Wenn B fällt von 1→0, gibt das blaue UND-Gatter während des gesamten Übergangs 1 aus. Störimpuls vermeidet!



Im Allgemeinen kann eine Störung auftreten, wenn eine Änderung in einer einzelnen Variablen die Grenze zwischen zwei primären Implikanten in einer K-Map überschreitet.

Wir können den Glitch beseitigen, indem wir der K-Map redundante Implikanten hinzufügen, um diese Grenzen abzudecken. Dies ist natürlich mit zusätzlichen Hardwarekosten verbunden.

Minimierung von Logik Gattern

Die vorgestellten Verfahren führen auf Kanonische Darstellungen, die z.B. für PALs interessant sind.

Aus verschiedenen Gründen (z.B. Glitches) nimmt man manchmal nicht die minimale Form.

Für FPGAs, Standardzellen und Full-Custom Designs versucht man, die angebotenen Ressourcen (z.B. auch gemischte Gatter) möglichst optimal zu nutzen.
Man nennt diesen Vorgang '**Technology Mapping**'

Es handelt sich um komplizierte **Optimierungsverfahren**, die die logische Funktion mit den zur Verfügung stehenden Ressourcen implementieren und dabei z.B. die Chipfläche oder die Verzögerung minimieren.

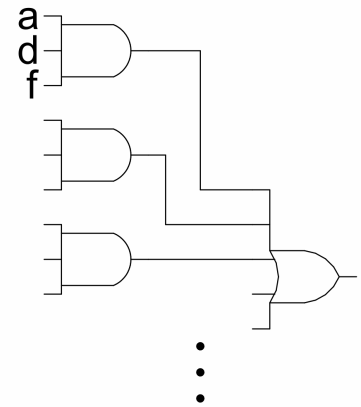
Multilevel-Logik

Beispiel: minimierte SoP Form

x = adf + aef + bdf + bef + cdf + cef + g (2-Ebenen)

Kosten: 1x ODER7, 6x UND3 => ~50 Transistoren

Verzögerung: ODER7 Verzögerung + UND3 Verzögerung

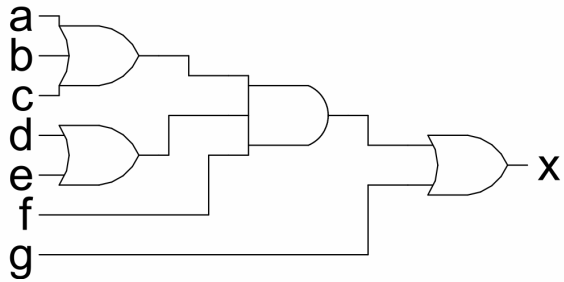


Faktorisierte Form:

x = (a + b +c)(d + e)f + g (3-Ebenen)

Kosten: 1x ODER3, 2x ODER2, 1x UND3 => ~20 Transistoren

Verzögerung: ODER3 + ODER2 + UND3 Verzögerung

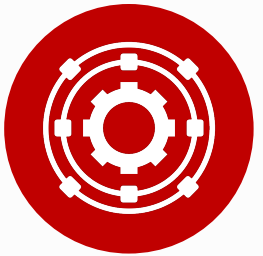


Was ist schneller?

Im Allgemeinen: Die Verwendung mehrerer Ebenen (mehr als 2) senkt die Kosten. Manchmal auch Verzögerung.

Manchmal ist es ein Kompromiss zwischen Kosten und Verzögerung.

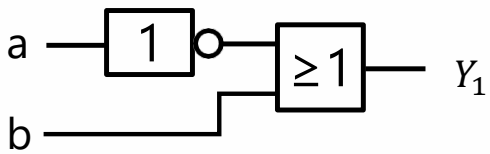
Schaltnetzrealisierung



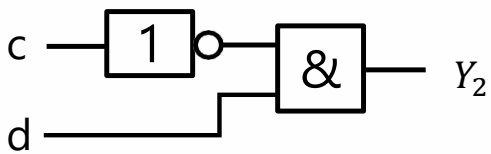
Schaltnetzrealisierung

Bis jetzt haben wir DNF und KNF studiert:

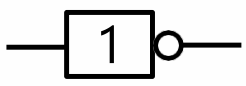
$Y_1 = \bar{a}b$



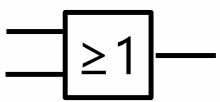
$Y_2 = \bar{c} + d$



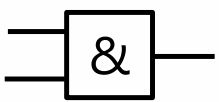
Dass heisst, wir brauchen 3 Typen von Gatter, und 2 Ebenen, um jede Funktion Schaltnetz realisieren zu können



NICHT



ODER



UND

Gibt's andere Möglichkeiten?

Schaltnetzrealisierungen mit anderen Basissystemen

Einige Gatter sind universell - sie können jede Funktion realisieren

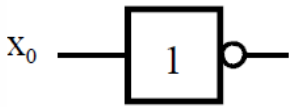
NAND Gatter können direkt jede logische Funktion oder jedes Schaltnetz in disjunktiver Normalform (DNF) realisieren.

NOR Gatter können jede logische Funktion oder jedes Schaltnetz in konjunktiver Normalform (KNF) realisieren.

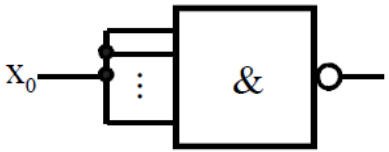
Topologie und Eingangs-/Ausgangsvariablen bleiben bei diesen Transformationen gleich, nur die Zwischenwerte werden invertiert.

Realisierung einer DNF mit NAND Gattern

Negation Äquivalent:



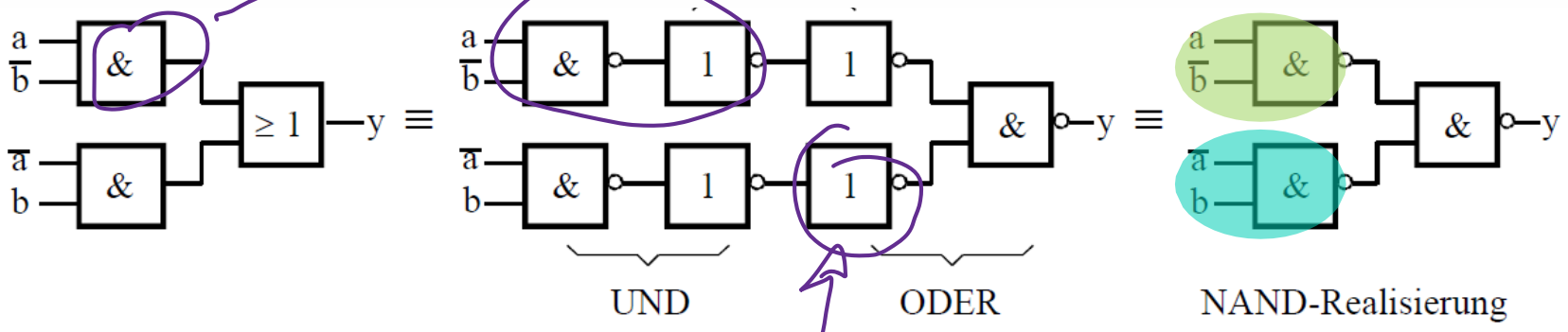
$y = \overline{x_0}$



$y = \overline{x_0 \wedge x_0 \wedge \dots}$

$= \overline{x_0}$

DNF:



$y = a\bar{b} + \bar{a}b$

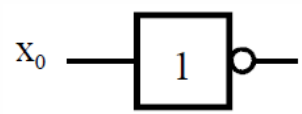
$a\bar{b} = \overline{a \; \textbf{NAND} \; \bar{b}}$

$x + y = \overline{(\bar{x})(\bar{y})}$

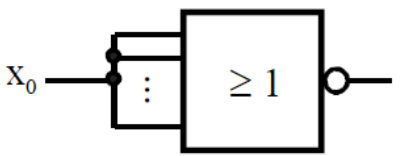
$y = \overline{(\overline{a\bar{b}})(\overline{\bar{a}b})}$

Realisierung einer KNF mit NOR Gattern:

Negation Äquivalent:



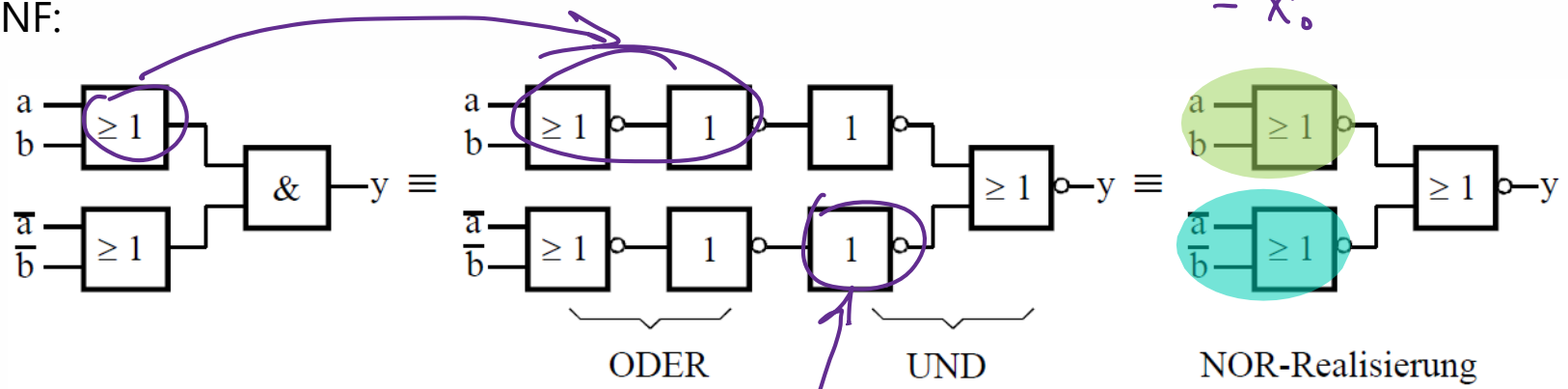
$y = \overline{x_0}$



$y = \overline{x_0 \vee x_0 \vee \dots}$

$= \overline{x_0}$

KNF:



$y = (a + b)(\bar{a} + \bar{b})$

$(a + b) = \overline{a \; NOR \; b}$

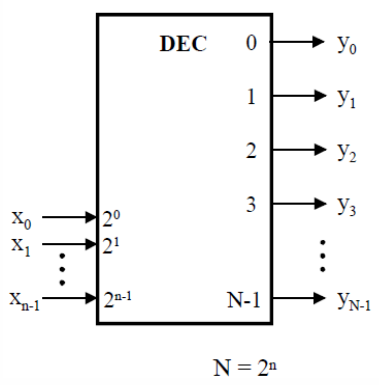
$xy = \overline{(\bar{x} + \bar{y})}$

$y = \overline{\overline{(a + b)} + \overline{(\bar{a} + \bar{b})}}$

Decodierer

Aufgabe des **Decodierers** ist jeder binär codierten Eingangskombination eine eindeutige Ausgangsleitung mit "1"-Belegung zuzuordnen. Damit können z. B: Zeilen bzw. Spalten im Speicherelementen, die in Matrixform angeordnet sind, selektiert werden.

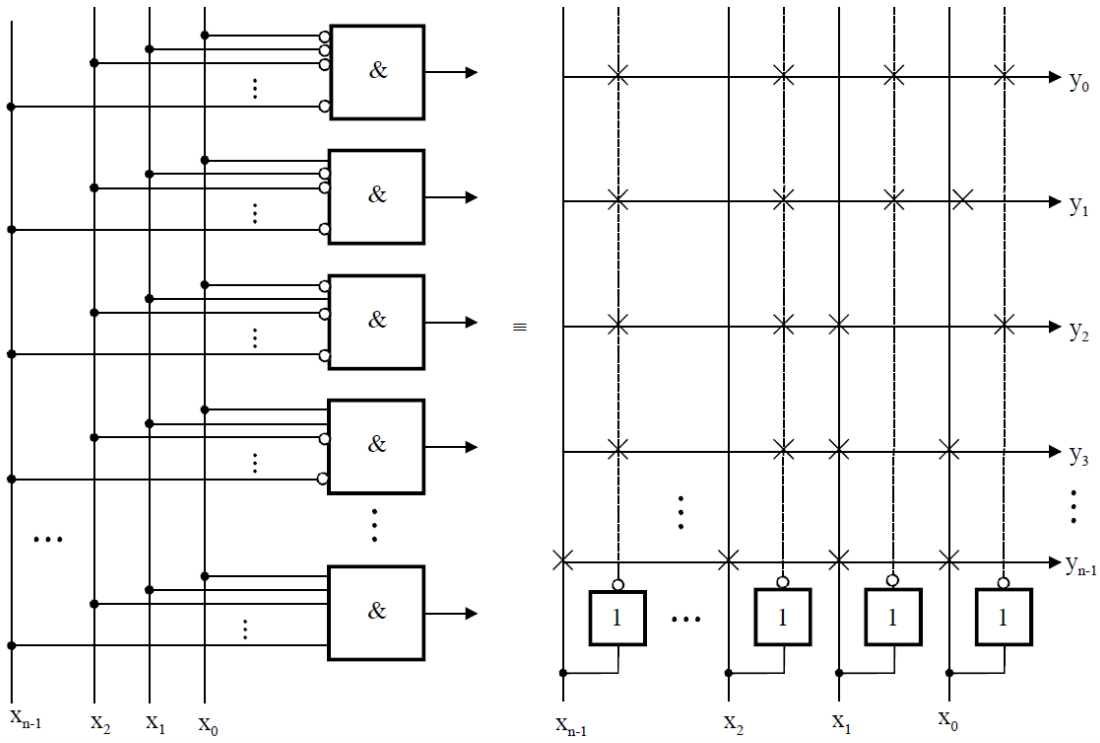
Die Ausgangsfunktionen y_{n-1} ... y_0 stellen den vollständigen Satz der Mintermfunktionen dar. Sie sind daher nicht minimierbar aber allein mit Konjunktionen realisierbar.



x_{n-1}	...	x_1	x_0	y_{N-1}	...	y_3	y_2	y_1	y_0
0	...	0	0	0	...	0	0	0	1
0	...	0	1	0	...	0	0	1	0
0	...	1	0	0	...	0	1	0	0
0	...	1	1	0	...	1	0	0	0
⋮			⋮	⋮			⋮	⋮	⋮
1	...	1	1	1	...	...	0	0	0

Decodierer

Die Realisierung mit UND-Gattern (n Eingänge) und als verdrahtete Logik.



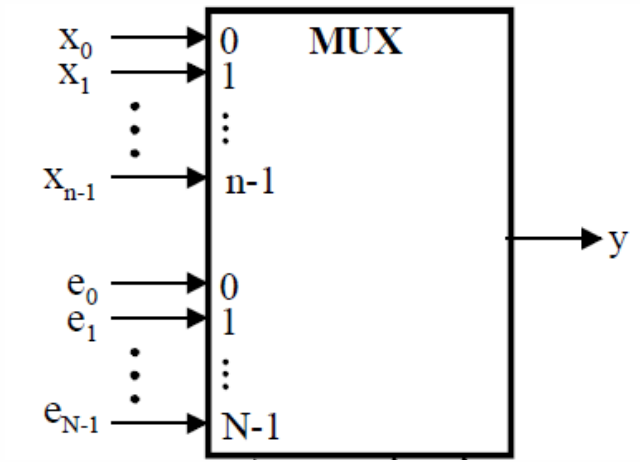
Multiplexer

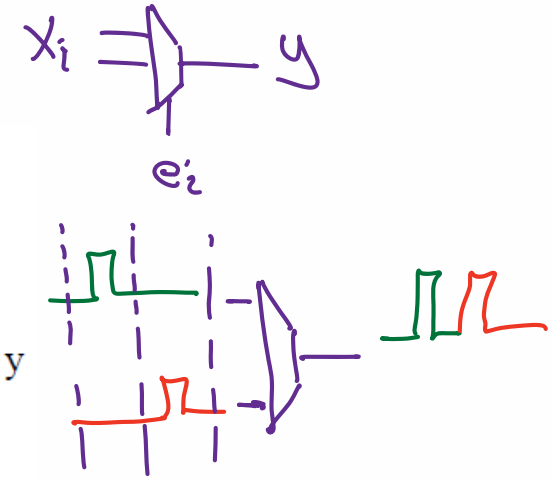
Mit den "Auswahleingängen" x_{n-1} ... x_0 wird **einer von N = 2^n binären Eingangswerten e_i dem Ausgang y zugeordnet**.

Die Realisierung ist wie beim Decoder mit einem zusätzlichen Eingang für die e_i an jedem UND Gatter und einer Disjunktion über alle UND-Gatter-Ausgänge.

Signalen X_i

Auswahl e_i

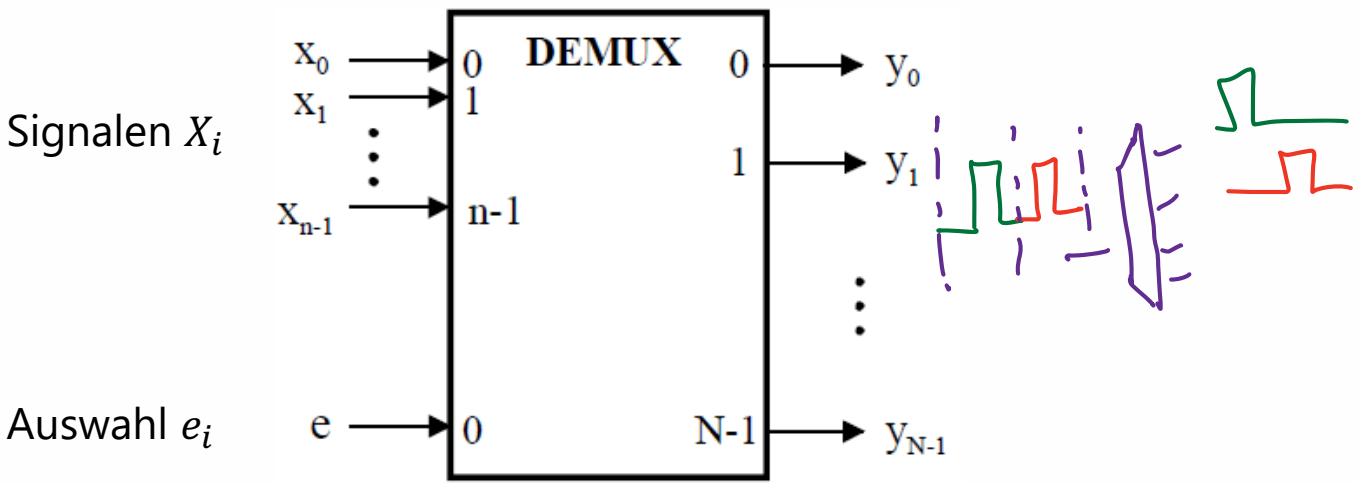




De-Multiplexer

Aufgabe des De-Multiplexers ist, den durch e gegebenen binären **Eingangswert eindeutig einem der Ausgänge y_{N-1} ... y_0 zuzuordnen**, der durch die "Auswahleingänge" x_{n-1} ... x_0 ausgewählt wird.

Realisierung wie beim Decoder mit zusätzlichem Eingang e an jedem UND-Gatter.

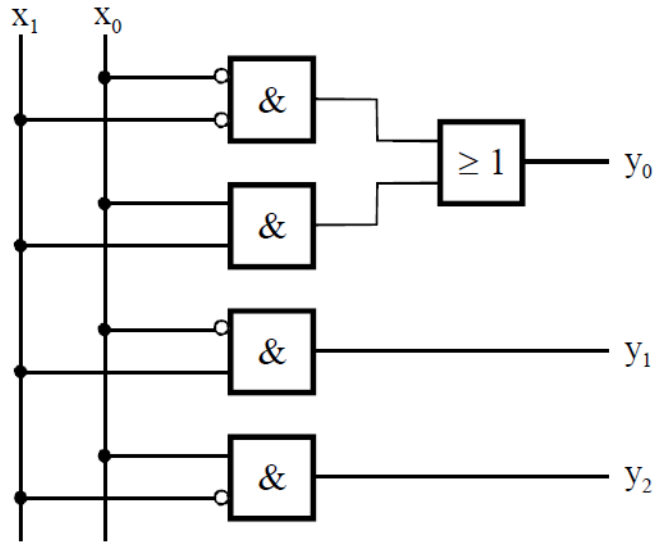


Bitstellenvergleicher

Das Bild zeigt Funktionstabelle und Realisierung eines **Vergleichers** für eine Bitstellle

x_1	x_0	y_2	y_1	y_0
0	0	0	0	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

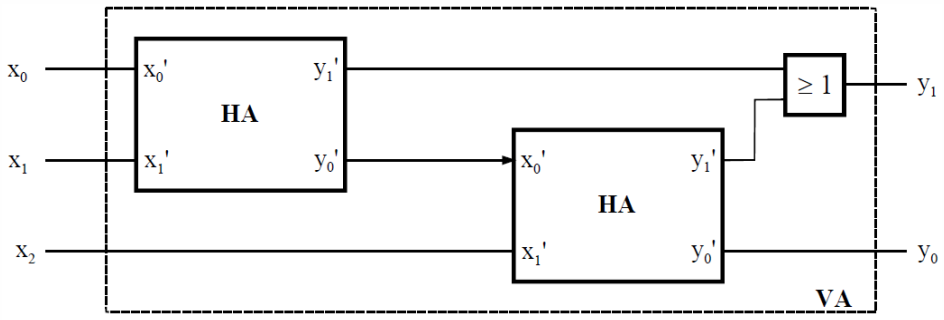
$y_0 = 1$ für $x_1 = x_0$
$y_1 = 1$ für $x_1 > x_0$
$y_2 = 1$ für $x_1 < x_0$



Addierer

Im ersten Teil von Kapitel 2 wurde bereits die Realisierung der Funktion **Volladdierer** mit Addition zweier Binärstellen und Eingangsübertrag sowie Additionsergebnis und Übertrag als Ausgängen als Beispiel vorgestellt.

Der **Halbaddierer** realisiert die binäre **Addition ohne Eingangsübertrag** aber mit Ausgangsübertrag. Durch **Kaskadierung von zwei Halbaddierern** lässt sich wiederum der **Volladdierer** aufbauen.



Der Volladdierer (DNF, minimiert) braucht viele Gattern aber nur 2 Ebenen.

Der Volladdierer aus Halbaddierern benutzt weniger Gattern, aber bis zu 5 Ebenen brauchen.

Allerdings weist dieser eine höhere Stufenanzahl (längere Laufzeit) auf.

Ripple-Carry Addierer

Um einen mehrstelligen Addierer aufzubauen, kann mehrere Volladdierer kaskadieren

Vorteil

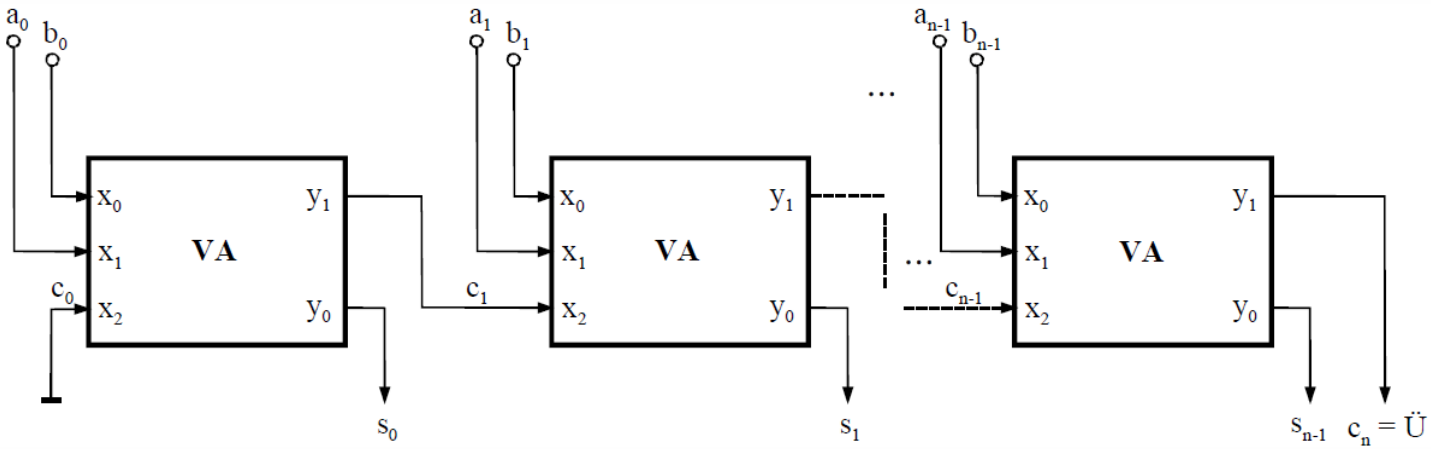
einfache und modulare Aufbau.

Nachteile:

Verzögerung der Übertragung

Mangel an Parallelität

Flächenbedarf



Subtrahierer

Für einen „Vollsubtrahierer“ muss gegenüber dem Volladierer nur die Funktion y_1 (Übertrag) geändert werden.

Um einen mehrstelligen Subtrahierer aufzubauen, kann man im Prinzip auch die Subtraktion als Addition einer Zahl im 2er Komplement mit einem ripple-carry-adder durchführen.

Der Eingangsübertrag wird dabei auf „1“ gelegt (=-1) und eine der Eingangs-zahlen (hier: b_i) bitweise invertiert (s= a-b).