

Kapitel 3 – Kombinatorische Logik

1. **Kombinatorische Schaltkreise**

2. Boolesche Algebren
3. Boolesche Ausdrücke, Normalformen, zweistufige Synthese
4. Berechnung eines Minimalpolynoms
5. Arithmetische Schaltungen
6. Anwendung: ALU von ReTI

Albert-Ludwigs-Universität Freiburg

Prof. Dr. Christoph Scholl

Institut für Informatik

WS 2015/16

Definition

Kombinatorische Logik ist ein Modell von Hardware, die eine boolesche Funktion $f : \mathbb{B}^n \rightarrow \mathbb{B}^m$ ($n, m \in \mathbb{N}$) implementiert.

- Ein **kombinatorischer Schaltkreis** (Schaltnetz) hat n Eingänge und m Ausgänge. Legt man an den Eingängen den Vektor $i \in \mathbb{B}^n$ an, berechnet der Schaltkreis den Vektor $f(i) \in \mathbb{B}^m$ und stellt ihn an den Ausgängen bereit.
- Es gibt weitere Arten von Hardware:
 - Sequentielle Logik mit speichernden Elementen (später).
 - Analog- und Mixed-Signal-Blöcke (nicht in TI).

- **Kombinatorische Logiksynthese** ist das Problem, zu einer gegebenen Booleschen Funktion einen möglichst effizienten kombinatorischen Schaltkreis, d. h. einen mit möglichst geringen Kosten, zu finden.
- ~~Die Definition von~~ **Kosten** hängen von der verwendeten Technologie ab und können sich auf die **Größe**, **Verzögerung**, **Energieverbrauch** des Schaltkreises beziehen und eventuell weitere Parameter (Zuverlässigkeit, Testbarkeit, ...) berücksichtigen.

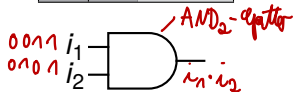
- Wir konzentrieren uns hier auf zwei Arten von Technologien:
 - 1 **Programmierbare Logikfelder** (Programmable Logic Arrays, PLAs).
 - Implementieren sogenannte zweistufigen Realisierungen, siehe später (Kapitel 3.3).
 - 2 Mehrstufige Realisierungen mit allgemeinen Bibliothekszellen (**Logik-Gattern**).

- Gatter sind kleine kombinatorische Blöcke, in der Regel mit bis zu 4 Eingängen und einem Ausgang.
- Gatter werden mit Transistoren realisiert.
- Gatter können zu größeren Schaltungen verbunden werden.
- Die Menge der verfügbaren Gattern ergibt eine Standardzellen-Bibliothek.

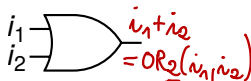
Gatter - " .

Einige wichtige Gatter

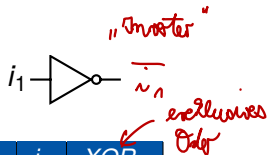
i_1	i_2	AND_2
0	0	0
0	1	0
1	0	0
<u>1</u>	<u>1</u>	<u>1</u>



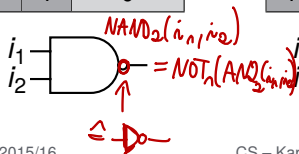
i_1	i_2	OR_2
<u>0</u>	<u>0</u>	<u>0</u>
0	1	1
1	0	1
1	1	1



i_1	NOT_1
0	1
1	0



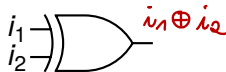
i_1	i_2	$NAND_2$
0	0	1
0	1	1
1	0	1
1	1	0



i_1	i_2	NOR_2
0	0	1
0	1	0
1	0	0
1	1	0

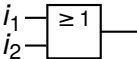

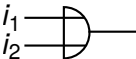
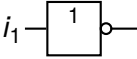
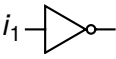
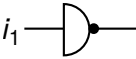


i_1	i_2	XOR_2
0	0	0
0	1	1
1	0	1
1	1	0



Logikgatter - verschiedene Notationen

Es gibt verschiedene Notationen für Logikgatter.

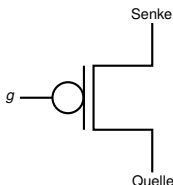
	IEC	ANSI	DIN
OR_2			
NOT			

Wir werden in dieser Vorlesung die **ANSI-Notation** verwenden.

Transistoren

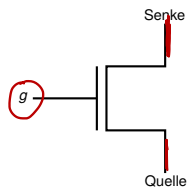
- Einen Transistor kann man vereinfacht als spannungsgesteuerten Schalter sehen:
 - Leitung g (gate) regelt Leitfähigkeit zwischen Quelle und Senke.

p-Kanal Transistor

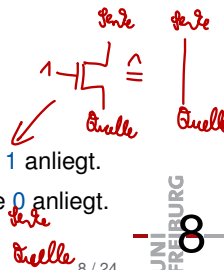


- Leitet, wenn an g eine 0 anliegt.
- Sperrt, wenn an g eine 1 anliegt.

n-Kanal Transistor

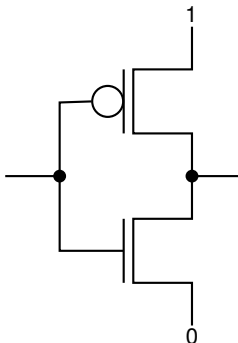


- Leitet, wenn an g eine 1 anliegt.
- Sperrt, wenn an g eine 0 anliegt.



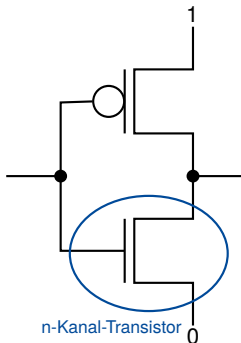
Realisierung von Gattern in CMOS-Technologie

- Complementary Metal Oxide Semiconductor.
- Es werden p- und n-Kanal-Transistoren verwendet.
- Beispiel: CMOS-Inverter.



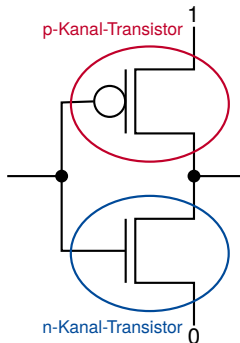
Realisierung von Gattern in CMOS-Technologie

- Complementary Metal Oxide Semiconductor.
- Es werden p- und n-Kanal-Transistoren verwendet.
- Beispiel: CMOS-Inverter.



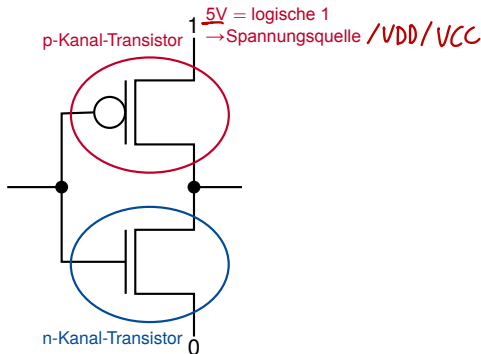
Realisierung von Gattern in CMOS-Technologie

- Complementary Metal Oxide Semiconductor.
- Es werden p- und n-Kanal-Transistoren verwendet.
- Beispiel: CMOS-Inverter.



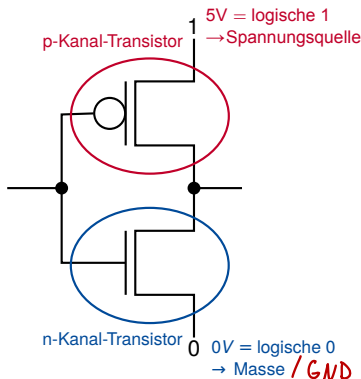
Realisierung von Gattern in CMOS-Technologie

- Complementary Metal Oxide Semiconductor.
- Es werden p- und n-Kanal-Transistoren verwendet.
- Beispiel: CMOS-Inverter.



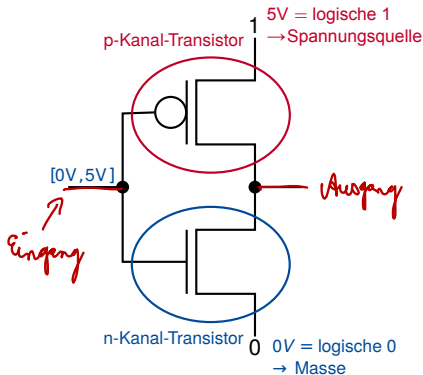
Realisierung von Gattern in CMOS-Technologie

- Complementary Metal Oxide Semiconductor.
- Es werden p- und n-Kanal-Transistoren verwendet.
- Beispiel: CMOS-Inverter.



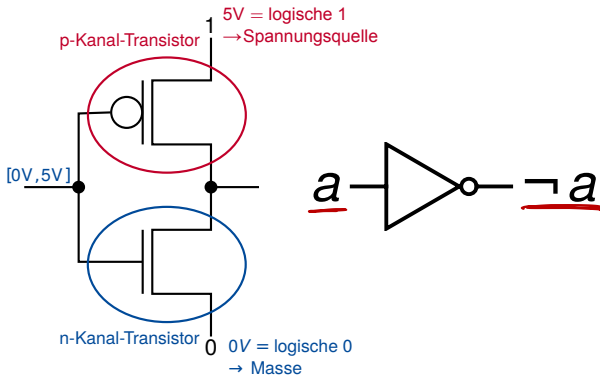
Realisierung von Gattern in CMOS-Technologie

- Complementary Metal Oxide Semiconductor.
- Es werden p- und n-Kanal-Transistoren verwendet.
- Beispiel: CMOS-Inverter.



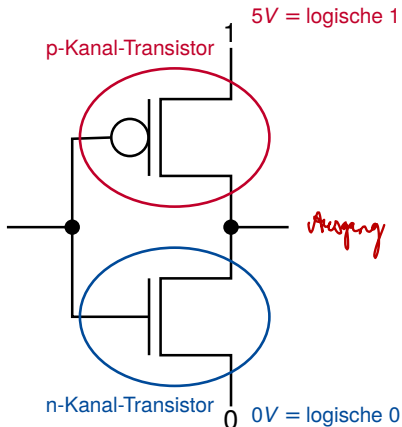
Realisierung von Gattern in CMOS-Technologie

- Complementary Metal Oxide Semiconductor.
- Es werden p- und n-Kanal-Transistoren verwendet.
- Beispiel: CMOS-Inverter.



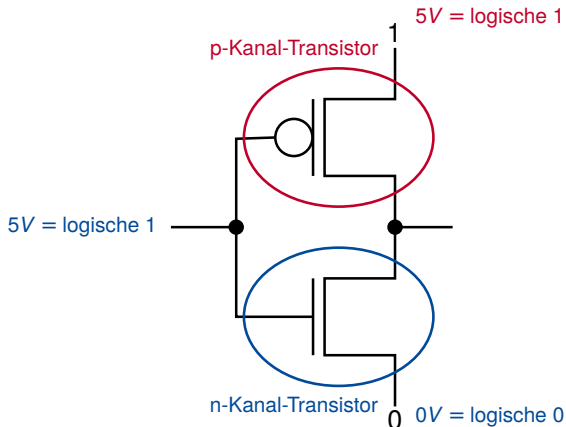
CMOS-Inverter mit 1 am Gate

- Leitender Pfad zwischen Ausgang und Masse (logische 0).



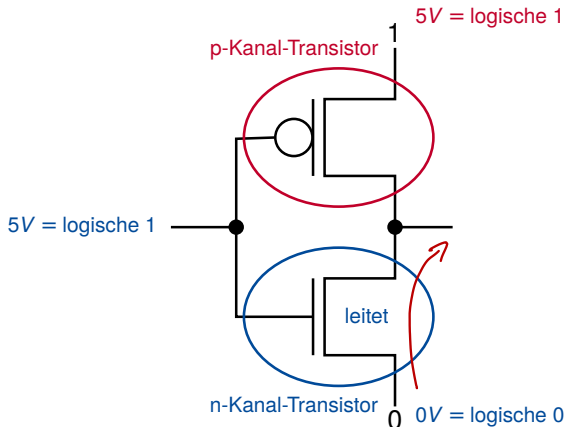
CMOS-Inverter mit 1 am Gate

- Leitender Pfad zwischen Ausgang und Masse (logische 0).



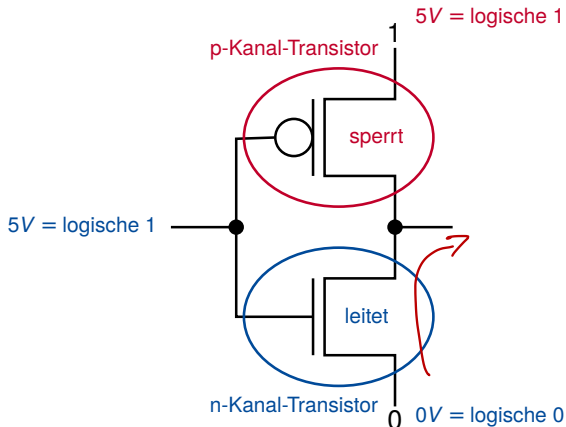
CMOS-Inverter mit 1 am Gate

- Leitender Pfad zwischen Ausgang und Masse (logische 0).



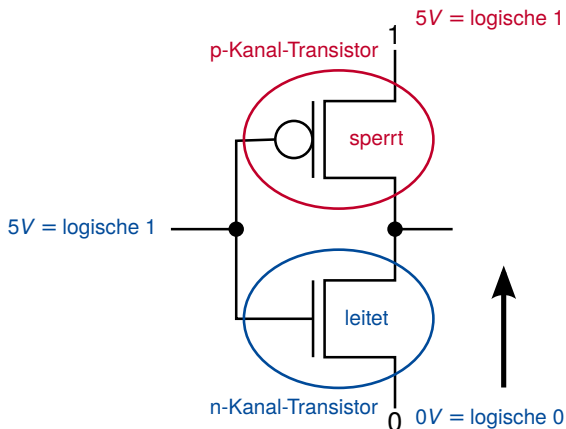
CMOS-Inverter mit 1 am Gate

- Leitender Pfad zwischen Ausgang und Masse (logische 0).



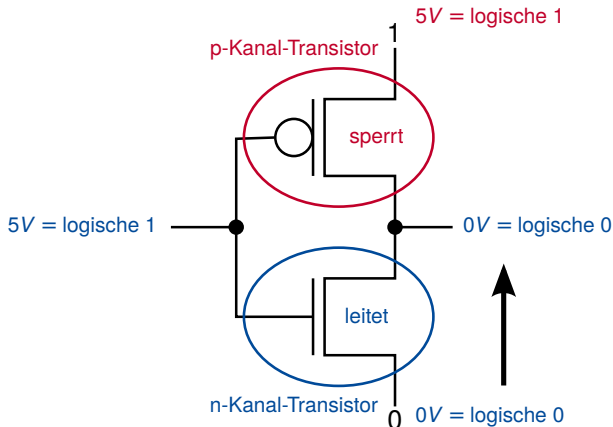
CMOS-Inverter mit 1 am Gate

- Leitender Pfad zwischen Ausgang und Masse (logische 0).



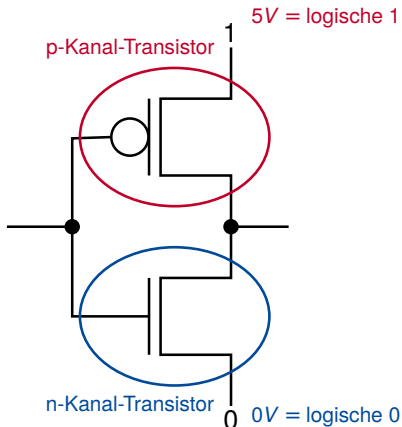
CMOS-Inverter mit 1 am Gate

- Leitender Pfad zwischen Ausgang und Masse (logische 0).



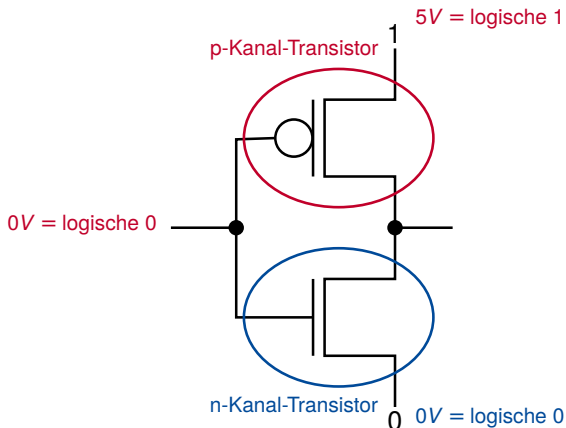
CMOS-Inverter mit 0 am Gate

- Leitender Pfad zwischen Ausgang und Spannungsversorgung (logische 1).



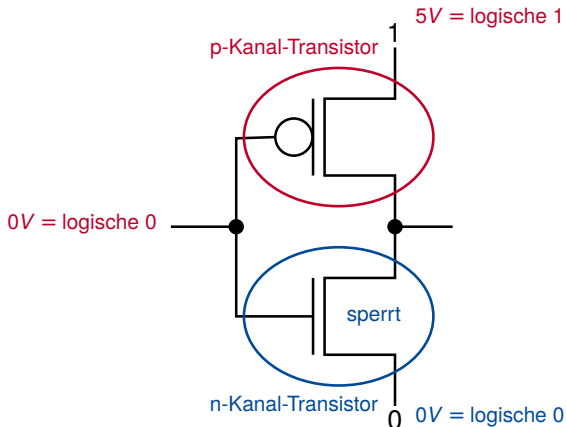
CMOS-Inverter mit 0 am Gate

- Leitender Pfad zwischen Ausgang und Spannungsversorgung (logische 1).



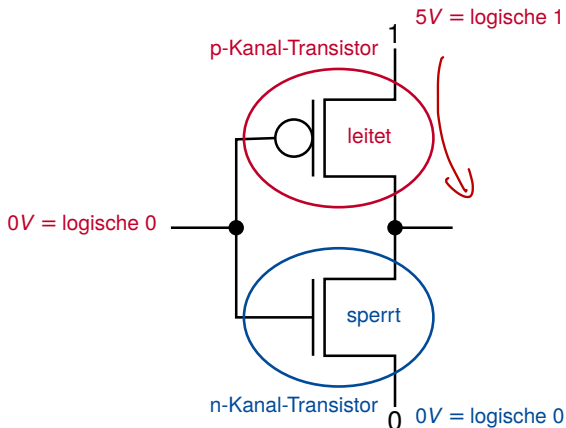
CMOS-Inverter mit 0 am Gate

- Leitender Pfad zwischen Ausgang und Spannungsversorgung (logische 1).



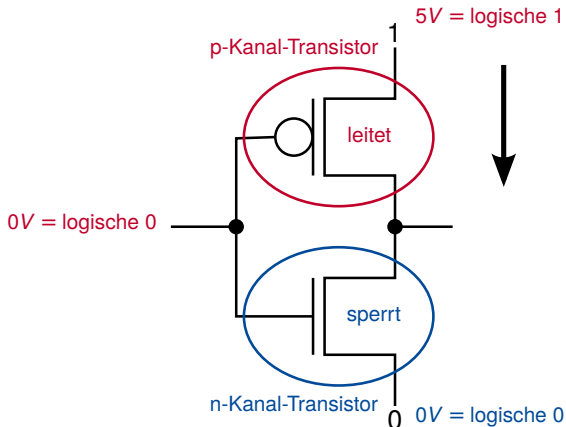
CMOS-Inverter mit 0 am Gate

- Leitender Pfad zwischen Ausgang und Spannungsversorgung (logische 1).



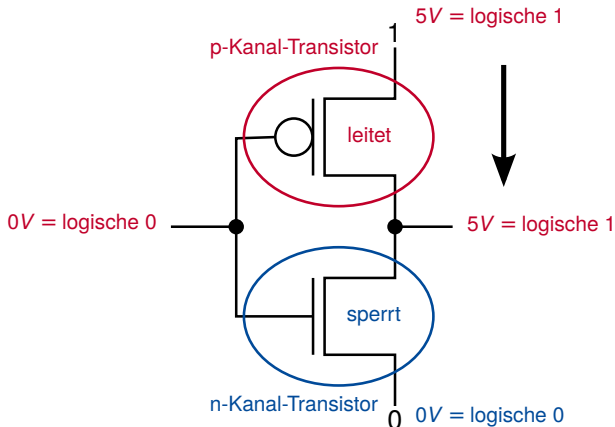
CMOS-Inverter mit 0 am Gate

- Leitender Pfad zwischen Ausgang und Spannungsversorgung (logische 1).



CMOS-Inverter mit 0 am Gate

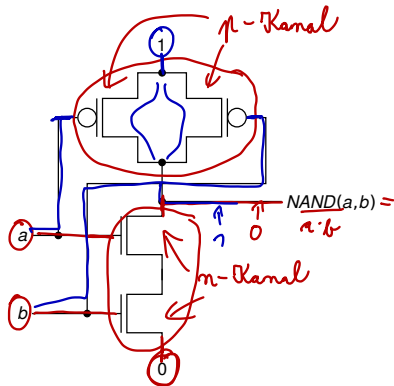
- Leitender Pfad zwischen Ausgang und Spannungsversorgung (logische 1).



CMOS-NAND-Gatter

$$NAND = NOT(AND)$$

$$NAND(i_1, i_2) = \overline{i_1 \cdot i_2}$$



■ Ausgang ist 0

⇔ Es existiert ein leitender Pfad von 0 zum Ausgang

⇔ beide n-Kanal-Transistoren leiten

⇔ $a = b = 1, a \wedge b = 1$

⇔ $NAND(a,b) = 0$

■ Ausgang ist 1

⇔ Es existiert ein leitender Pfad von 1 zum Ausgang

⇔ einer der p-Kanal-Transistoren leitet

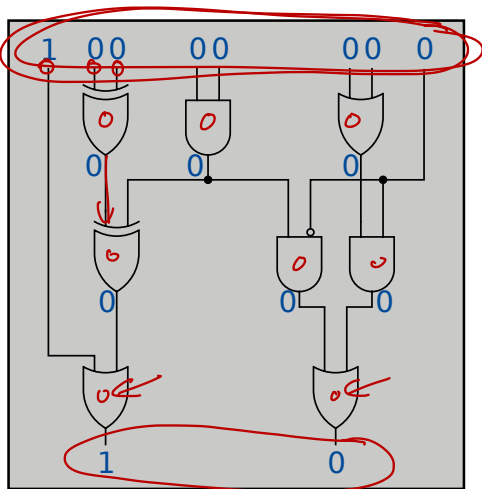
⇔ $a = 0$ oder $b = 0, \neg a \vee \neg b = 1$

⇔ $NAND(a,b) = 1$

$$\neg a \vee \neg b = \neg(a \wedge b) \\ = NAND(a, b)$$

- Es gibt **keine „direkte“ Implementierung** von AND- und OR-Gattern. Sie werden aus NAND-/NOR-Gattern plus Invertern zusammengesetzt.
- Zu jedem p-Kanal Transistor gibt es stets einen **komplementären n-Kanal-Transistor**, der genau dann sperrt, wenn der erste Transistor leitet und umgekehrt. Dadurch gibt es niemals einen leitenden Pfad von der Stromversorgung zur Masse. Dies reduziert Leistungsverluste.

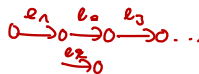
Schaltkreis: Zunächst informal durch Beispiel ($f \in \mathbb{B}_{8,2}$)



- Idee:
„gerichteter Graph mit einigen zusätzlichen Eigenschaften“

Gerichtete Graphen (Wiederholung!)

- $G = (V, E)$ ist ein **gerichteter Graph**, genau dann, wenn
 - V endliche nichtleere Menge („Knoten“)
 - E endliche Menge („Kanten“)
 - Auf E sind Abbildungen $Q, Z : E \rightarrow V$ definiert
($Q(e)$ heißt **Quelle**, $Z(e)$ **Ziel** einer Kante e)
- Die Abbildung **$\text{indeg} : V \rightarrow \mathbb{N}$** , $\text{indeg}(v) = |\{e \mid Z(e) = v\}|$ gibt den **Eingangsgrad** eines Knotens $v \in V$ an.
- Die Abbildung **$\text{outdeg} : V \rightarrow \mathbb{N}$** , $\text{outdeg}(v) = |\{e \mid Q(e) = v\}|$ gibt den **Ausgangsgrad** eines Knotens $v \in V$ an.
- Ein **Pfad** (der Länge k) in G ist eine Folge von k Kanten e_1, e_2, \dots, e_k ($k \geq 0$) mit $Z(e_i) = Q(e_{i+1}) \forall i$ mit $k-1 \geq i \geq 1$.
 $Q(e_1)$ heißt Quelle, $Z(e_k)$ Ziel des Pfades.
- Ein **Zyklus** in G ist ein Pfad der Länge ≥ 1 in G , bei dem Ziel und Quelle identisch sind.
- G heißt **azyklisch**, falls kein Zyklus in G existiert.
- Die **Graph-Tiefe** eines azyklischen Graphen ist definiert als die Länge des längsten Pfades in G .



Modellierung durch Schaltkreise (1/2)

$$\text{z.B. } BIB = \{ \text{and}_2, \text{or}_2, \text{not}_1, \text{xor}_2 \}$$

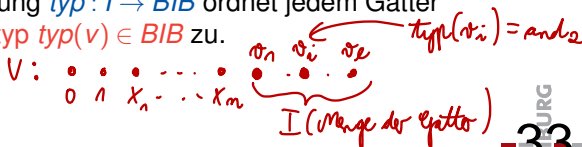
- Eine **Zellenbibliothek** $BIB \subset \bigcup_{n \in \mathbb{N}} \mathbb{B}_n$ enthält Basisoperatoren, die den Grundgattern entsprechen.
- Ein 5-Tupel $SK = (\vec{X}_n, \underline{G}, \underline{typ}, \underline{IN}, \vec{Y}_m)$ heißt **Schaltkreis** mit n **Eingängen** und m **Ausgängen** über der Zellenbibliothek BIB genau dann wenn

- $\vec{X}_n = (x_1, \dots, x_n)$ ist eine endliche Folge von Eingängen.

- $G = (V, E)$ ist ein azyklischer, gerichteter Graph mit $\{0, 1\} \cup \{x_1, \dots, x_n\} \subseteq V$.

- Die Menge $I = V \setminus (\{0, 1\} \cup \{x_1, \dots, x_n\})$ heißt **Menge der Gatter**. Die Abbildung $typ: I \rightarrow BIB$ ordnet jedem Gatter $v \in I$ einen **Zellentyp** $typ(v) \in BIB$ zu.

- ...



Modellierung durch Schaltkreise (2/2)



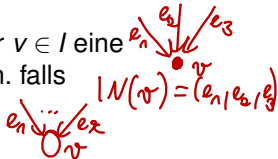
■ ...

■ Für jedes Gatter $v \in I$ mit $\text{typ}(v) \in \mathbb{B}_k$ gilt $\text{indeg}(v) = k$.

■ $\text{indeg}(v) = 0$ für $v \in \{0, 1\} \cup \{x_1, \dots, x_n\}$.

■ Die Abbildung $IN: I \rightarrow E^*$ legt für jedes Gatter $v \in I$ eine Reihenfolge der eingehenden Kanten fest, d.h. falls $\text{indeg}(v) = k$, dann ist $IN(v) = (e_1, \dots, e_k)$ mit $Z(e_i) = v \forall 1 \leq i \leq k$.

■ Die Folge $\vec{Y}_m = (y_1, \dots, y_m)$ zeichnet Knoten $y_i \in V$ als Ausgänge aus.

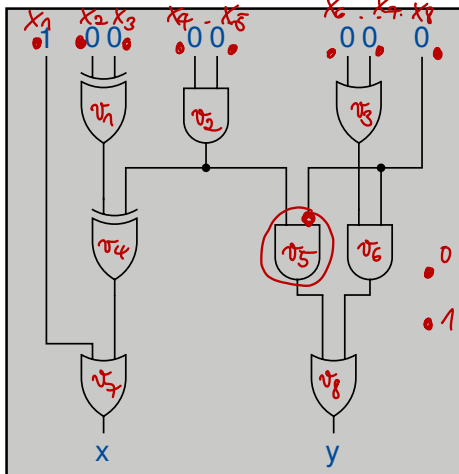


Beispiel für einen mehrstufigen komb. Schaltkreis ($f \in \mathbb{B}_{8,2}$)

BIB = $\{e_{2,2}, r_{2,2}\}$

$\sigma_2, f\}$

x_1	x_2	f
0	0	0
0	1	0
1	0	1
1	1	0



$$IN(\sigma_5) = (e_8, e_9)$$

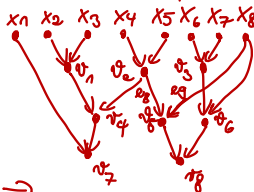
Ausführliche Schreibweise:
 $SK = (X_8, G, typ, N, Y_2)$

$$X_8 = (x_1, \dots, x_8)$$

$$G = (V, E)$$

$$V = \{0, 1\} \cup \{x_1, \dots, x_8\} \cup \{\sigma_1, \dots, \sigma_8\}$$

$$E = \{ \dots \} Q(\dots), Z(\dots)$$

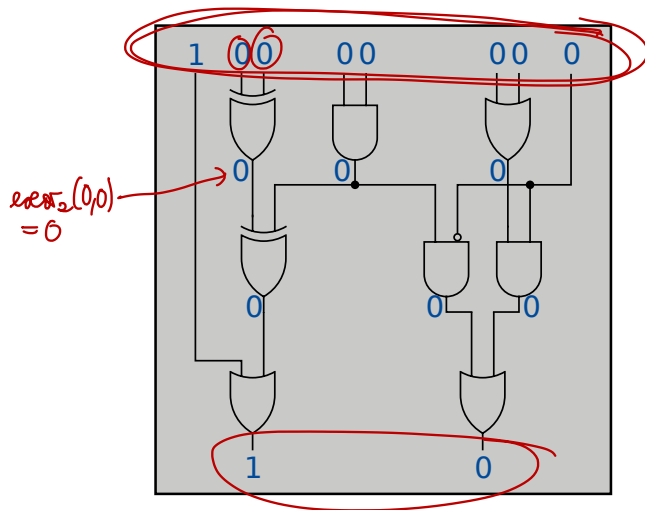


$$Y_2 = (\sigma_7, \sigma_8)$$

$$typ: \{\sigma_1, \dots, \sigma_8\} \rightarrow BIB$$

$$typ(\sigma_1) = e_{2,2}, typ(\sigma_2) = r_{2,2}$$

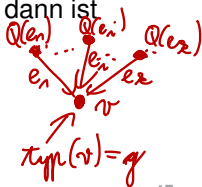
Beispiel für einen mehrstufigen komb. Schaltkreis ($f \in \mathbb{B}_{8,2}$)



Formale Semantikdefinition für Schaltkreise (1/2)

- Sei $SK = (\vec{X}_n, G, typ, IN, \vec{Y}_m)$ ein Schaltkreis über einer Zellenbibliothek BIB .
- Sei eine Eingangsbelegung $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{B}^n$ gegeben.
- Eine Belegung $\Phi_{SK, \alpha} : V \rightarrow \mathbb{B}$ für alle Knoten $v \in V$ ist dann gegeben durch die folgenden Definitionen:
 - $\Phi_{SK, \alpha}(x_i) = \alpha_i \quad \forall 1 \leq i \leq n.$
 - $\Phi_{SK, \alpha}(0) = 0, \Phi_{SK, \alpha}(1) = 1.$
 - falls $v \in I$ mit $typ(v) = g \in \mathbb{B}_k, IN(v) = (e_1, \dots, e_k)$, dann ist

$\Phi_{SK, \alpha}(v) = g(\Phi_{SK, \alpha}(Q(e_1)), \dots, \Phi_{SK, \alpha}(Q(e_k))).$
- Warum ist das wohldefiniert?
- Weil G azyklisch!



Graph, der nicht zyklisch ist:



Für Graphen, die nicht zyklisch sind, ist die Semantik entw. nicht wohldefiniert.

Formale Semantikdefinition für Schaltkreise (2/2)

- $(\Phi_{SK,\alpha}(y_1), \dots, \Phi_{SK,\alpha}(y_m))$ ist dann die unter Eingangsbelegung $\alpha = (\alpha_1, \dots, \alpha_n)$ berechnete **Ausgangsbelegung** des Schaltkreises SK .
- Die Berechnung von $\Phi_{SK,\alpha}$ bei Eingangsbelegung α heißt auch **Simulation** von SK für Belegung α .
- Die an einem Knoten v berechnete Boolesche Funktion $\Psi(v) : \mathbb{B}^n \rightarrow \mathbb{B}$ ist definiert durch

$$\Psi(v)(\alpha) := \Phi_{SK,\alpha}(v)$$

für ein beliebiges $\alpha \in \mathbb{B}^n$.

- Die durch den Schaltkreis berechnete Funktion ist

$$f_{SK} : \mathbb{B}^n \rightarrow \mathbb{B}^m, f_{SK} = (\Psi(y_1), \dots, \Psi(y_m)).$$

- Eine **Standardzellen-Bibliothek** enthält eine Menge von Gattern ~~und kleinen kombinatorischen Schaltungen~~ (Standardzellen).
 - Z. B. AND-Gatter mit 4 Eingängen
- Für jedes Element der Bibliothek werden Parameter wie **Fläche** auf dem Chip, **Schaltgeschwindigkeit**, **Leistungsaufnahme** des Gatters bzw. der Standardzelle abgespeichert.
- Es sind oft z. B. mehrere Inverter unterschiedlicher Größe und Geschwindigkeit vorhanden.

Kombinatorische Logiksynthese

- Allgemeine kombinatorische Logiksynthese optimiert mehrere Parameter gleichzeitig.
- Exakte Verfahren existieren, stoßen aber schon für kleinste Schaltkreise an ihre Grenzen.
- In der Praxis werden Heuristiken eingesetzt, die auf Ausschnitten eines großen Schaltkreises lokale Optimierungen durchführen.
- Hier beschränken wir uns bei der Logiksynthese auf eine wichtige Unterklasse von kombinatorischen Schaltkreisen: Die zweistufige Logik.
- Allgemeinere kombinatorische Schaltkreise betrachten wir später bei der Einführung arithmetischer Schaltkreise.

