

Tutorat 0

Organisatorisches und Grundlagenwissen

Gruppe 9

Präsentator:
Jürgen Mattheis
(juergmatth@gmail.com)

Vorlesung von:
Prof. Dr. Scholl

Übungsgruppenbetreuung:
Tobias Seufert

19. August 2023

Universität Freiburg, Lehrstuhl für Rechnerarchitektur

Gliederung

Organisatorisches

Bonus

Literatur

Organisatorisches

Organisatorisches

Studienleistung

- ▶ Anmeldung zur Übung in unserem Übungsportal
- ▶ Anmeldung zur Studienleistung im HisInOne
- ▶ Zu erbringende Leistungen:
 - ▶ Mindestens 75% der Aufgaben in den Übungsblättern müssen sinnvoll bearbeitet werden.
 - ▶ Das Bearbeiten einer Teilaufgabe einer Aufgabe zählt bereits als sinnvolles Bearbeiten der gesamten Aufgabe.
 - ▶ Die Aufgabe muss nicht korrekt gelöst sein, es muss nur sichtbar sein, dass versucht wurde diese Aufgabe zu lösen.
 - ▶ Eine Aufgabe aus den Übungen muss im Tutorat vorgerechnet werden.
 - ▶ Regelmäßige, aktive Teilnahme an den Übungsgruppen 😊.

Organisatorisches

Infos zum Tutorat

- ▶ Tutorat wird aufgezeichnet:
<https://www.youtube.com/playlist?list=PLmsC317bB1b0T1981vGTddbqThQ6gDDG2>
- ▶ **BITTE** gebt RETI-Code **getippt** ab und nicht handschriftlich.



- ▶ Kritik am Tutorat: <https://forms.gle/gLJHVMZhQWcSK2N18>

Organisatorisches Hilfsmittel

- ▶ Mindmap zum Vorlesungsstoff:
https://github.com/matthejue/Mindmaps/releases/download/main/Technische_Informatik.pdf
- ▶ zum Überprüfen der RETI-Abgaben: <https://github.com/matthejue/PicoC-Compiler/releases>
 - ▶ Dokumentation: https://github.com/matthejue/Bachelorarbeit_Dokumentation_out/blob/main/Dokumentation.pdf
 - ▶ Einführung:
https://github.com/matthejue/PicoC-Compiler/blob/master/doc/getting_started.md
 - ▶ Bugs melden: <https://github.com/matthejue/PicoC-Compiler/issues>

Organisatorisches Gruppenbildung

- ▶ Wer würde sich gerne mit einer anderen Person aus dem Tutorat (Tutor ausgenommen) zu einer Gruppe zusammenschließen?
- ▶ Die Studenten freuen sich wegen Arbeitsteilung über weniger Tipparbeit. Der Tutor freut sich auch über weniger Korrekturen.



Bonus

Bonus

Anzahl Formeln

- ▶ Anzahl Zeilen in Wahrheitstabelle: $2^{\# \text{ Variablen}}$
- ▶ Anzahl Aussagenlogische Formeln: $2^{\#\text{Zeilen}} = 2^{(2^{\#\text{Variablen}})}$
- ▶ bei 3 Aussagenlogischen Variablen gibt es $2^3 = 8$ Zeilen in der Wahrheitstabelle und damit $2^{(2^3)} = 256$ verschiedenen Aussagenlogische Formeln, da man diese 2^3 Zeilen auch nochmal auf exponentiell $2^{\#\text{Zeilen}}$ viele verschiedene Arten belegen kann

| a | b | f_0 | f_1 | f_2 | f_3 | f_4 | f_5 | f_6 | f_7 | f_8 | f_9 | f_{10} | f_{11} | f_{12} | f_{13} | f_{14} | f_{15} |
|---|---|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|----------|----------|----------|----------|----------|----------|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Bonus

Minterme und Maxterme

- ▶ 16 mögliche Logikfunktionen für 2 Aussagenlogische Variablen

| a | b | f_0 | f_1 | f_2 | f_3 | f_4 | f_5 | f_6 | f_7 | f_8 | f_9 | f_{10} | f_{11} | f_{12} | f_{13} | f_{14} | f_{15} |
|-----|-----|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|----------|----------|----------|----------|----------|----------|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

- ▶ f_1, f_2, f_4 und f_8 sind Minterme (für genau eine Variation der Eingabewerte den Wert 1)
- ▶ f_7, f_{11}, f_{13} und f_{14} sind Maxterme (für genau eine Variation der Eingabewerte den Wert 0)

Bonus

Minterme und Maxterme

- ▶ die 4 Minterme können als Konjunktionen dargestellt werden:
 $m_0(a, b) = \bar{a} \cdot \bar{b}, m_1(a, b) = \bar{a} \cdot b, m_2(a, b) = a \cdot \bar{b}, m_3(a, b) = a \cdot b$
- ▶ die 4 Maxterme können als Disjunktionen dargestellt werden:
 $M_0(a, b) = \bar{a} + \bar{b}, M_1(a, b) = \bar{a} + b, M_2(a, b) = a + \bar{b}, M_3(a, b) = a + b$
- ▶ Vergleich:

| a | b | $\neg a \cdot b$ | $a + \neg b$ |
|-----|-----|------------------|--------------|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 |

- ▶ $\neg(\neg a \wedge b) = a \vee \neg b$: „alles außer“ $\neg a \wedge b$ ist 1 $\rightarrow (a = 0, b = 1)$ ist als einziges 0

Bonus

Konjunktive und Disjunktive Normalform

- ▶ aus drei **Basistypen** (Disjunktion / Konjunktion und Negation) lassen sich alle anderen **Logikfunktion** erzeugen
- ▶ Jede Logikfunktion $f : \mathbb{B}^2 \rightarrow \mathbb{B}$ lässt sich in **Disjunktiver Normalform (DNF)**:
$$f(a, b) = f(0, 0) \cdot \bar{a} \cdot \bar{b} + f(0, 1) \cdot \bar{a} \cdot b + f(1, 0) \cdot a \cdot \bar{b} + f(1, 1) \cdot a \cdot b$$
- ▶ als auch in **konjunktiver Normalform (KNF)** darstellen:
$$f(a, b) = (f(0, 0) + a + b) \cdot (f(0, 1) + a + \bar{b}) \cdot (f(1, 0) + \bar{a} + b) \cdot (f(1, 1) + \bar{a} + \bar{b})$$
- ▶ man möchte **Logische Funktion** (Wertetabelle) mit möglichst wenig Schaltelementen realisieren → schauen, ob **DNF** oder **KNF** kürzer ist, je nachdem, ob die Logische Funktion (Menge an Formeln) mehr oder weniger **Modelle** besitzt, also mehr oder weniger Variationen aus Aussagenlogischen Variablen besitzt, die 1 ergeben

Bonus

Konjunktive und Disjunktive Normalform

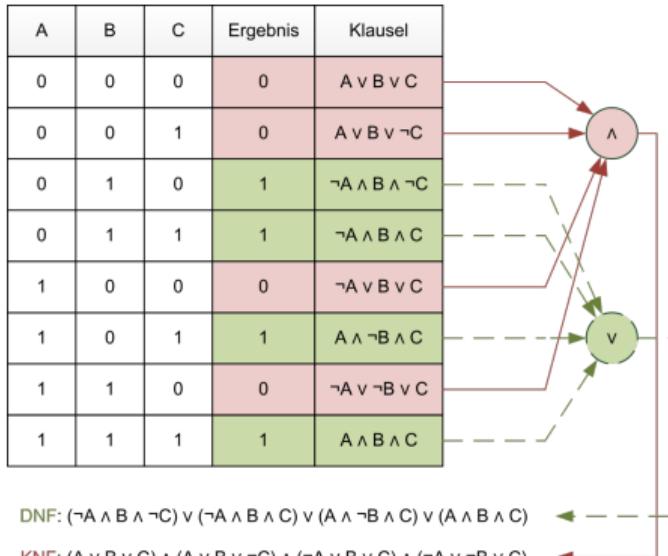


Abbildung 1: Disjunktive und Konjunktive Normalform [1]

Bonus

Konjunktive und Disjunktive Normalform

- Beispiel: „höchstens 2 wahre aussagenlogische Variablen“

- DNF:

$$(\neg a \cdot \neg b \cdot \neg c) + (\neg a \cdot \neg b \cdot c) + (\neg a \cdot b \cdot \neg c) + (\neg a \cdot b \cdot c) + (a \cdot \neg b \cdot \neg c) + (a \cdot \neg b \cdot c) + (a \cdot b \cdot \neg c)$$

- KNF: $(\neg a + \neg b + \neg c)$

Literatur

Online

- [2] *File:Handshake icon black circle.svg - Wikipedia.* 21. Mai 2020. URL:
https://commons.wikimedia.org/wiki/File:Handshake_icon_black_circle.svg (besucht am 27. 04. 2023).

Sonstiges

- [1] *Disjunktive Normalform*. In: Wikipedia. Page Version ID: 230680696. 8. Feb. 2023. URL: https://de.wikipedia.org/w/index.php?title=Disjunktive_Normalform&oldid=230680696 (besucht am 27.04.2023).

Tutorat 1

Vollständige Induktion, Aussagenlogik, O-Notation

Gruppe 9

Präsentator:

Jürgen Mattheis

(juergmatth@gmail.com)

Vorlesung von:
Prof. Dr. Scholl

Übungsgruppenbetreuung:
Tobias Seufert

15. August 2023

Universität Freiburg, Lehrstuhl für Rechnerarchitektur

Gliederung

Aufgabe 1

Aufgabe 2

Aufgabe 3

Bonus

Aufgabe 1

Aufgabe 1 |

Vollständige Induktion

Voraussetzungen 1.1



- ▶ $\sum_{i=0}^n z^i = \frac{z^{n+1} - 1}{z - 1}$
- ▶ *es gelte:* $z \in \mathbb{R} \setminus \{1\}$ und $n \in \mathbb{N}$

Aufgabe 1 II

Lösung 1.1



- *Induktionsanfang:* $n = 0$

$$\sum_{i=0}^0 z^i = 1 = \frac{z-1}{z-1} = \frac{z^{0+1}-1}{z-1}$$

- *Induktionsschritt:* $n \rightarrow n + 1$

$$\begin{aligned}
 \sum_{i=0}^{n+1} z^i &= \left(\sum_{i=0}^n z^i \right) + z^{n+1} \\
 &= \frac{z^{n+1}-1}{z-1} + z^{n+1} \quad \text{wegen Induktionsvoraussetzung} \\
 &= \frac{z^{n+1}-1 + z^{n+1} \cdot (z-1)}{z-1} \\
 &= \frac{z^{n+1}-1 + z^{n+2} - z^{n+1}}{z-1} \\
 &= \frac{z^{n+2}-1}{z-1}
 \end{aligned}$$

Aufgabe 2

Aufgabe 2 |

Aussagenlogik

Voraussetzungen 2.1



$$x \vee y := x + y - x \cdot y$$

$$x \wedge y := x \cdot y$$

$$\neg x := 1 - x$$

- *es gilt:* $x^2 = x \cdot x = x$, da $x \in \{0,1\}$ oder wenn man auf den \wedge -Operator zurückführt:
 $x^2 = x \cdot x = x \wedge x = x$

Aufgabe 2 II

Aussagenlogik

Lösung 2.1



| x | y | z | $x \wedge y$ | $(x \wedge y) \wedge z$ | $y \wedge z$ | $x \wedge (y \wedge z)$ |
|-----|-----|-----|--------------|-------------------------|--------------|-------------------------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Aufgabe 2 III

Aussagenlogik

Lösung 2.1



| x | y | z | $x \vee y$ | $(x \vee y) \vee z$ | $y \vee z$ | $x \vee (y \vee z)$ |
|-----|-----|-----|------------|---------------------|------------|---------------------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Aufgabe 2 IV

Aussagenlogik

Lösung 2.1



| x | y | $y \wedge \neg y$ | $x \vee (y \wedge \neg y)$ | $y \vee \neg y$ | $x \wedge (y \vee \neg y)$ |
|-----|-----|-------------------|----------------------------|-----------------|----------------------------|
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |

Aufgabe 2 |

Aussagenlogik

Lösung 2.2



$$\begin{aligned}x \vee (x \wedge y) &= x + (x \cdot y) - x(x \cdot y) \\&= x + xy - x^2y \\&\stackrel{*}{=} x + xy - xy \\&= x\end{aligned}$$

Aufgabe 2 II

Aussagenlogik

Lösung 2.2



$$\begin{aligned}x \wedge (x \vee y) &= x \cdot (x + y - xy) \\&= x^2 + xy - x^2y \\&\stackrel{*}{=} x + xy - xy \\&= x\end{aligned}$$

Aufgabe 2 III

Aussagenlogik

Lösung 2.2



$$\begin{aligned}(x \vee y) \wedge (x \vee z) &= (x + y - xy) \cdot (x + z - xz) \\&= x(x + z - xz) + y(x + z - xz) + xy(x + z - xz) \\&= x^2 + xz - x^2z + xy + yz - xyz - x^2y - xyz + x^2yz \\&= x + xz - x^2z + xy + yz - xyz - xy - xyz + xyz \\&= x + yz - xyz \\&= x + (y \wedge z) - x \wedge (y \wedge z) \\&= x \vee (y \wedge z)\end{aligned}$$

Aufgabe 2 IV

Aussagenlogik

Lösung 2.2



$$\begin{aligned}(x \wedge y) \vee (x \wedge z) &= xy + xz - (xy \cdot xz) \\&= xy + xz - (x^2yz) \\&= xy + xz - (xyz) \\&= x(y + z - yz) \\&= x \cdot (y \vee z) \\&= x \wedge (y \vee z)\end{aligned}$$

Aufgabe 3

Aufgabe 3 |

O-Notation

Voraussetzungen 3.1



$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} \leq c \Rightarrow f(n) \in O(g(n)) \quad (c \in \mathbb{N}_0) \quad (1)$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty \Rightarrow f(n) \notin O(g(n)) \quad (2)$$

$$\lim_{n \rightarrow \infty} \frac{\log(c \cdot n)}{n} = 0 \quad (c \in \mathbb{N}_{\geq 1}) \quad (3)$$

$$\lim_{n \rightarrow \infty} \frac{n}{(\log n)^c} = \infty \quad (c \in \mathbb{N}_{\geq 1}) \quad (4)$$

Aufgabe 3 II

O-Notation

Lösung 3.1



$$\begin{aligned}\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} &= \lim_{n \rightarrow \infty} \frac{100n + \log n}{n + (\log n)^2} \leq \lim_{n \rightarrow \infty} \frac{100n + \log n}{n + 1^2} = \lim_{n \rightarrow \infty} \frac{100 + \frac{\log n}{n}}{1 + \frac{1}{n}} \\ &\stackrel{(3)}{=} \frac{100 + 0}{1 + 0} = 100 \in \mathbb{N}_0 \stackrel{(1)}{\Rightarrow} f(n) \in O(g(n))\end{aligned}$$

Aufgabe 3 III

O-Notation

Lösung 3.1



$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} &= \lim_{n \rightarrow \infty} \frac{n + (\log n)^2}{100n + \log n} = \lim_{n \rightarrow \infty} \frac{1 + \frac{(\log n)^2}{n}}{100 + \frac{\log n}{n}} \\ &\stackrel{(4)}{=} \frac{1+0}{100+0} = \frac{1}{100} \leq 1 \in \mathbb{N}_0 \stackrel{(1)}{\Rightarrow} g(n) \in O(f(n)) \end{aligned}$$

Aufgabe 3 |

O-Notation

Lösung 3.2



$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{\frac{n^2}{\log n}}{n(\log n)^2} = \lim_{n \rightarrow \infty} \frac{\frac{n}{\log n}}{(\log n)^2} = \lim_{n \rightarrow \infty} \frac{n}{(\log n)^3} \stackrel{(4)}{=} \infty \stackrel{(2)}{\Rightarrow} f(n) \notin O(g(n))$$

Lösung 3.2



$$\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = 0 \in \mathbb{N}_0 \stackrel{(1)}{\Rightarrow} g(n) \in O(f(n))$$

ODER

$$f(n) \notin O(g(n)) \Leftrightarrow g(n) \notin \Omega(f(n)) \Leftrightarrow g(n) \in o(f(n)) \Rightarrow g(n) \in O(f(n))$$

Bonus

Bonus I

Vollständige Induktion über natürliche Zahlen

- ▶ für Allbehauptungen: $\forall n \in \mathbb{N} : \mathcal{E}(n)$ (Eigenschaft \mathcal{E})
- ▶ die allgemeine Form wird als Strukturelle Induktion bezeichnet, Vollständige Induktion ist zum Beweis von Aussagen für alle Natürlichen Zahlen
 - ▶ Menge der natürlichen Zahlen \mathbb{N} weißt geignete Struktur auf, da auf ihnen die Nachfolgerbeziehung besteht
- ▶ man führt Teilbeweise für 2 Fälle:
 - ▶ Basisfall: $\mathcal{E}(0)$, d.h. die natürliche Zahl 0 hat die Eigenschaft ϵ
 - ▶ Induktionsfall: $\forall i \in \mathbb{N} (\mathcal{E}(i) \Rightarrow \mathcal{E}(i + 1))$, d.h. die Eigenschaft ϵ überträgt sich von jeder natürlichen Zahl i auf ihren Nachfolger $i + 1$

Bonus II

Vollständige Induktion über natürliche Zahlen

- ▶ bei beweisender Allbehauptung für den Induktionsfall wird Implikation: Sei $i \in \mathbb{N}$ beliebig. Zu zeigen: $\epsilon(i) \Rightarrow \epsilon(i+1)$ zerlegt in: Sei $i \in \mathbb{N}$ beliebig. Es gelte: $\epsilon(i)$. Zu zeigen: $\epsilon(i+1)$
- ▶ Diese Darstellung des Induktionsfalls legt seine Zerlegung in die üblichen drei Bestandteile **Induktionsannahme**, **Induktionsbehauptung**, **Induktionsschritt** nahe:

Behauptung: $\forall n \in \mathbb{N} \ \mathcal{E}(n)$

Beweis: Durch vollständige Induktion nach n .

Basisfall $n = 0$: *Teilbeweis, dass $\mathcal{E}(0)$ gilt.*

Induktionsfall $n \rightarrow n + 1$: Sei $n \in \mathbb{N}$ beliebig.

Induktionsannahme: $\mathcal{E}(n)$

Induktionsbehauptung: $\mathcal{E}(n+1)$

Induktionsschritt: *Teilbeweis, dass die Induktionsbehauptung gilt.*

In diesem kann die Induktionsannahme verwendet werden.

Bonus III

Vollständige Induktion über natürliche Zahlen

- ▶ Induktionsfall beweist Allbehauptung mit darin enthaltener Implikationsbehauptung
- ▶ Wenn für Eigenschaft ϵ Basisfall und Induktionsfall bewiesen sind, gilt, dass jede natürliche Zahl die Eigenschaft ϵ hat, denn:
 0. Wegen des Basisfalls gilt $\epsilon(0)$.
 1. Mit $\epsilon(0)$ gilt wegen des Induktionsfalls auch $\epsilon(1)$.
 2. Mit $\epsilon(1)$ gilt wegen des Induktionsfalls auch $\epsilon(2)$.
 3. Mit $\epsilon(2)$ gilt wegen des Induktionsfalls auch $\epsilon(3)$.
 4. ...

Bonus IV

Vollständige Induktion über natürliche Zahlen

Behauptung: Für alle $n \in \mathbb{N}$ gilt $\sum_{i=0}^n i = \frac{n(n+1)}{2}$

Beweis: Durch vollständige Induktion nach n .

Basisfall $n = 0$:

$$\sum_{i=0}^0 i = 0 = \frac{0 \cdot (0+1)}{2}$$

Induktionsfall $n \rightarrow n + 1$: Sei $n \in \mathbb{N}$ beliebig.

Induktionsannahme: $\sum_{i=0}^n i = \frac{n(n+1)}{2}$

Induktionsbehauptung: $\sum_{i=0}^{n+1} i = \frac{(n+1)((n+1)+1)}{2}$

Bonus V

Vollständige Induktion über natürliche Zahlen

Induktionsschritt:

$$\begin{aligned}\sum_{i=0}^{n+1} i &= \left(\sum_{i=0}^n i \right) + (n+1) \\&= \frac{n(n+1)}{2} + (n+1) \quad (\text{Induktionsannahme}) \\&= \frac{n(n+1) + 2(n+1)}{2} \\&= \frac{(n+2)(n+1)}{2} = \frac{(n+1)((n+1)+1)}{2} \quad \blacksquare\end{aligned}$$

Bonus

O-Notation

- ▶ Man will Funktionen irgendwie vergleichen und das einzige sinnvolle was man vergleichen kann ist die **Wachstumsrate**, denn für zwei Funktionen kann gleichzeitig gelten: $g = O(f)$, also g wächst nicht stärker als f und $g > f$, also g ist überall echt größer als f
- ▶ Konstante Faktoren und Summanden $c \cdot x$ und $c + x$ sollen keine Rolle, aber es geht um **Wachstumsrate**, Konstante Faktoren und Summanden spielen keine Rolle
- ▶ Die Operatoren $O, \Omega, \Theta, o, \omega$ sind auf Funktionen, was die Operatoren $\leq, \geq, =, <, >$ auf Zahlen sind
- ▶ $f = O(g)$ wird gesprochen als „ f ist GroB-O von g “, wobei eigentlich $f \in O(g)$ mathematisch korrekt ist
 - ▶ man sagt die „Laufzeit eines Algorihmus ist: $\Omega(n \cdot \log(n))$ “, man sagt **NICHT**, dass ein „Algorithmus Laufzeit mindestens $O(n \cdot \log(n))$ hat“

Bonus |

Landau-Symbole

► formal:

- ▶ $O(g) = \{h : \mathbb{N} \rightarrow \mathbb{R} \mid \exists n_0 \in \mathbb{N}, \exists c > 0, \forall n \geq n_0, h(n) \leq c \cdot g(n)\}$
 - ▶ $f \in O(g)$ bedeutet „ f wächst höchstens so stark wie g “
- ▶ $\Omega(g) = \{h : \mathbb{N} \rightarrow \mathbb{R} \mid \exists n_0 \in \mathbb{N}, \exists c > 0, \forall n \geq n_0, h(n) \geq c \cdot g(n)\}$
 - ▶ $f \in \Omega(g)$ bedeutet „ f wächst mindestens so stark wie g “
- ▶ $\Theta(g) = \{h : \mathbb{N} \rightarrow \mathbb{R} \mid \exists n_0 \in \mathbb{N}, \exists c_1 > 0, \exists c_2 > 0, \forall n \geq n_0, c_1 \cdot g(n) \leq h(n) \leq c_2 \cdot g(n)\}$
 - ▶ $f \in \Theta(g)$ bedeutet „ f wächst genauso stark wie g “
- ▶ $o(g) = \{h : \mathbb{N} \rightarrow \mathbb{R} \mid \forall c > 0, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, h(n) \leq c \cdot g(n)\}$

Bonus II

Landau-Symbole

- ▶ $f \in o(g)$ bedeutet „ f wächst (strikt) langsamer als g “
- ▶ $\omega(g) = \{h : \mathbb{N} \rightarrow \mathbb{R} \mid \forall c > 0, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, h(n) \geq c \cdot g(n)\}$
- ▶ $f \in \omega(g)$ bedeutet „ f wächst (strikt) schneller als g “

Anmerkungen

- ▶ $o(g) \cap \omega(g) = \emptyset$,
- ▶ $\Theta(g) = O(g) \cap \Omega(g)$
- ▶ $f(n) \in \Theta(g(n)) \Leftrightarrow f(n) \in O(g(n)) \text{ and } f(n) \in \Omega(g(n))$

- ▶ Transpose symmetry:

- ▶ $f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$

Bonus III

Landau-Symbole

- ▶ $f(n) = o(g(n)) \Leftrightarrow g(n) = \omega(f(n))$
- ▶ Bestimmung über Grenzwerte:

$$1. \ f = O(g) \Leftrightarrow \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$$

$$2. \ f = \Omega(g) \Leftrightarrow \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} > 0$$

$$3. \ f = \Theta(g) \Leftrightarrow \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} > 0 \text{ und } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$$

$$4. \ f = o(g) \Leftrightarrow \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

Bonus IV

Landau-Symbole

5. $f = \omega(g) \Leftrightarrow \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$

Bonus

Regel von L'Hopital

$$\blacktriangleright \lim_{x \rightarrow c} \frac{f(x)}{g(x)} = \lim_{x \rightarrow c} \frac{f'(x)}{g'(x)}$$

- ▶ wird verwendet bei Grenzwertbetrachtung bei der man bei Zähler und Nenner entweder den Fall $\frac{0}{0}$ oder $\frac{\pm\infty}{\pm\infty}$ hat und beide Ableitungen differenzierbar
- ▶ dann Zähler und Nenner des Bruches getrennt voneinander ableiten und dann nochmal Grenzwertbetrachtung
- ▶ wenn nochmal unbestimmter Ausdruck rauskommt und wieder entweder der Fall $\frac{0}{0}$ oder $\frac{\infty}{\infty}$ und beide Ableitungen differenzierbar, dann nochmal anwenden oder sonst Pech gehabt

Tutorat 2

RETI, Huffman-Kodierung

Gruppe 9

Präsentator:
Jürgen Mattheis
(juergmatth@gmail.com)

Vorlesung von:
Prof. Dr. Scholl

Übungsgruppenbetreuung:
Tobias Seufert

15. August 2023

Universität Freiburg, Lehrstuhl für Rechnerarchitektur

Gliederung

Aufgabe 1

Aufgabe 2

Aufgabe 3

Bonus

Aufgabe 1

Aufgabe 1 |

RETI

Aufgabe 1.1



Es gilt für $n \geq 0$:

$$f(0) = 0$$

$$f(1) = 1$$

$$f(n) = f(n-1) + f(n-2) \quad (\text{für } n \geq 2)$$

Aufgabe 1 II

RETI

Lösung 1.1



```
1 int fib(int fib_nr) {  
2     int fib_i;  
3     int fib_i_1 = 0;  
4     int fib_i_2 = 1;  
5     while (fib_nr) {  
6         fib_i = fib_i_1 + fib_i_2;  
7         fib_i_2 = fib_i_1;  
8         fib_i_1 = fib_i;  
9         fib_nr = fib_nr - 1;  
10    }  
11    return fib_i;  
12 }  
13  
14 void main() {  
15     print(fib(input()));  
16 }
```

Aufgabe 1 III

RETI

Lösung 1.1

```
1 # INPUT
2 CALL INPUT ACC
3 STORE ACC 30
4 # PROGRAM
5 LOADI ACC 0
6 STORE ACC 31
7 LOADI ACC 1
8 STORE ACC 32
9 LOAD ACC 30
10 SUBI ACC 1
11 STORE ACC 30
12 JUMP== 9
```

```
13 LOAD ACC 31
14 ADD ACC 32
15 STORE ACC 33
16 LOAD ACC 32
17 STORE ACC 31
18 LOAD ACC 33
19 STORE ACC 32
20 JUMP -11
21 # OUTPUT
22 LOAD ACC 33
23 CALL PRINT ACC
```

Aufgabe 1 IV

RETI

Lösung 1.1



```

Index: 61
instruction: STORE ACC 31;
ACC: 0
ACC_SIMPLE: 3
IN1: 0
IN1_SIMPLE: 0
IN2: 0
IN2_SIMPLE: 0
PC: 2147483666
PC_SIMPLE: 2147483666
SP: 45
SP_SIMPLE: 45
BAF: 2147483658
BAF_SIMPLE: 2
CS: 2147483651
CS_SIMPLE: 3
DS: 2147483671
DS_SIMPLE: 23
SRAM:
00000 JUMP 0;
00001 2147483648
00002 0 <- BAF
00003 CALL INPUT ACC; <- CS
00004 LOADI DS 30;
00005 LOADI ACC 7;
00006 STORE ACC 31;
00007 LOADI ACC 1;
00008 STORE ACC 32;
00009 LOAD ACC 30;
00010 SUBI ACC 1;
00011 STORE ACC 30;
00012 ADDI DS 1;
00013 LOAD ACC 31;
00014 ADD ACC 32;
00015 STORE ACC 33;
00016 LOAD ACC 32;
00017 STORE ACC 33;
00018 LOAD ACC 33; <- PC
00019 STORE ACC 32;
00020 JUMP -11;
:nch
00021 LOAD ACC 33;
00022 CALL PRINT ACC;
00023 ADDI DS 0;
00024 0
00025 0
00026 0
00027 0
00028 0
00029 0
00030 6
00031 1
00032 3
00033 5
00034 0
00035 0
00036 0
00037 0
00038 0
00039 0
00040 0
00041 0
00042 0
00043 0
00044 0
00045 0 <- SP
UART:
00000 0
00001 0
00002 0
EPROM:
00008 LOADI DS -2097152; <- IN1 <- IN2
00009 MULT DS IN2;
00010 MOVE DS SP;
00011 MOVE DS BAF; <- ACC
00012 MOVE DS CS;
00013 ADDI SP 45;
00014 ADDI BAF 2;
00015 ADDI CS 3;
00016 ADDI DS 23;
00017 MOVE CS PC;

```

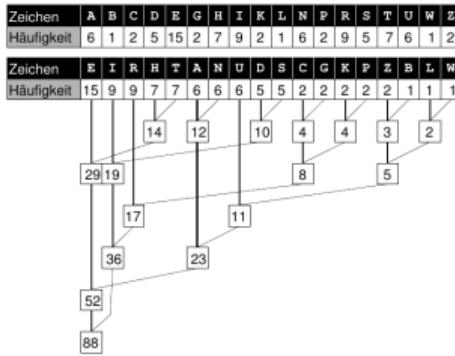


Aufgabe 2

Aufgabe 2 |

Huffman-Kodierung

Lösung 2.1



Anmerkungen

- ▶ die Codewörter müssen ausgehend von der Wurzel zu den Blättern aufgebaut werden, sonst ergibt sich kein Präfixcode

Aufgabe 2 II

Huffman-Kodierung

Lösung 2.1



| Zeichen | E | I | R | H | T | A | N | U | D | S | C | G | K | P | Z | B | L | W |
|---------|-----|-----|-----|------|------|------|------|------|------|------|-------|-------|-------|-------|--------|--------|--------|--------|
| Code | 000 | 100 | 110 | 0010 | 0011 | 0100 | 0101 | 0110 | 1010 | 1011 | 11100 | 11101 | 11110 | 11111 | 011100 | 011101 | 011110 | 011111 |

Anmerkungen

- ▶ Ein Präfixcode ist ein Code, bei dem kein Codewort Präfix eines anderen Codewortes ist.
- ▶ Z.B. wäre ein Code mit den Codewörtern $a = 10$, $b = 101$ kein Code, da man beim Dekodieren einer Nachricht beim Lesen von 10 nicht weiß, ob damit a oder, bei einer nachfolgenden 1, das Wort b gemeint ist.
- ▶ Einen Präfixcode kann man also „online“ dekodieren, d.h. man liest Zeichen und bei einem Matching mit einem Wort im Wörterbuch hat man das ursprüngliche Wort gefunden.

Aufgabe 2 III

Huffman-Kodierung Lösung 2.2

- Dekodierung: TI IST KLASSE

0011 100 100 1011 0011 11110 011110 0100 1011 1011 000
T I I S T K L A S S E

- Mögliches Problem: Codierung von Blättern hin zur Wurzel → kein Präfix-Code, Dekodierung nicht möglich (s.o.)

Lösung 2.3

- Damit andere die Nachricht dekodieren (d.h. lesen) können, muss man natürlich auch das Wörterbuch (d.h. die generierte Code-Tabelle) mitliefern.
- Da damit jedem, der die kodierte Nachricht erhält, auch die Dekodierungstabelle zugänglich ist, kann natürlich jeder die Nachricht dekodieren. D.h. die kodierte Nachricht genügt keinen kryptologischen Ansprüchen.
- Was man allerdings erreicht hat, ist eine Datenkomprimierung (bei großen Texten).

Aufgabe 3

Aufgabe 3 |

Huffman-Kodierung

Voraussetzungen 3.1



$$\sum_{j=1}^m p(a_j) = 1, p(a_i) > 0.5$$

Lösung 3.1



$$\sum_{j=1}^{i-1} p(a_j) + p(a_i) + \sum_{j=i+1}^m p(a_j) = 1$$

$$\sum_{j=1}^{i-1} p(a_j) + \sum_{j=i+1}^m p(a_j) < 0.5$$

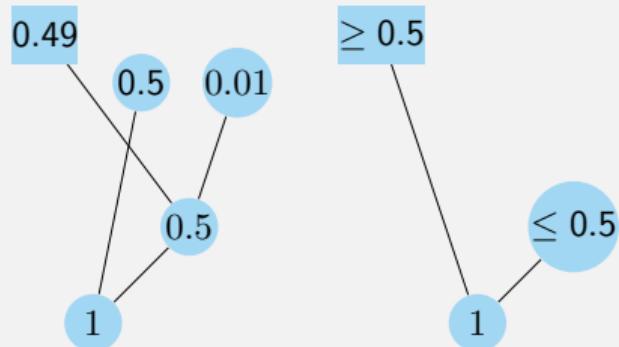
Aufgabe 3 II

Huffman-Kodierung

Lösung 3.1



- ▶ Beim Bau des binären Baums sind alle Häufigkeitsteilsummen kleiner als $p(a_i)$, daher wird a_i erst zu dem Baum hinzugefügt, nachdem alle anderen Knoten bereits zu einem Knoten zusammengefügt wurden.
- ▶ Dieser wird durch eine Kante mit einem neuen Knoten (Wurzel) verbunden, die andere Kante der Wurzel führt direkt zu a_i .



Aufgabe 3 |

Huffman-Kodierung

Voraussetzungen 3.1

- ▶ $p(a_i) < \frac{1}{3}$ und $|c(a_i)| = 1$



Lösung 3.1

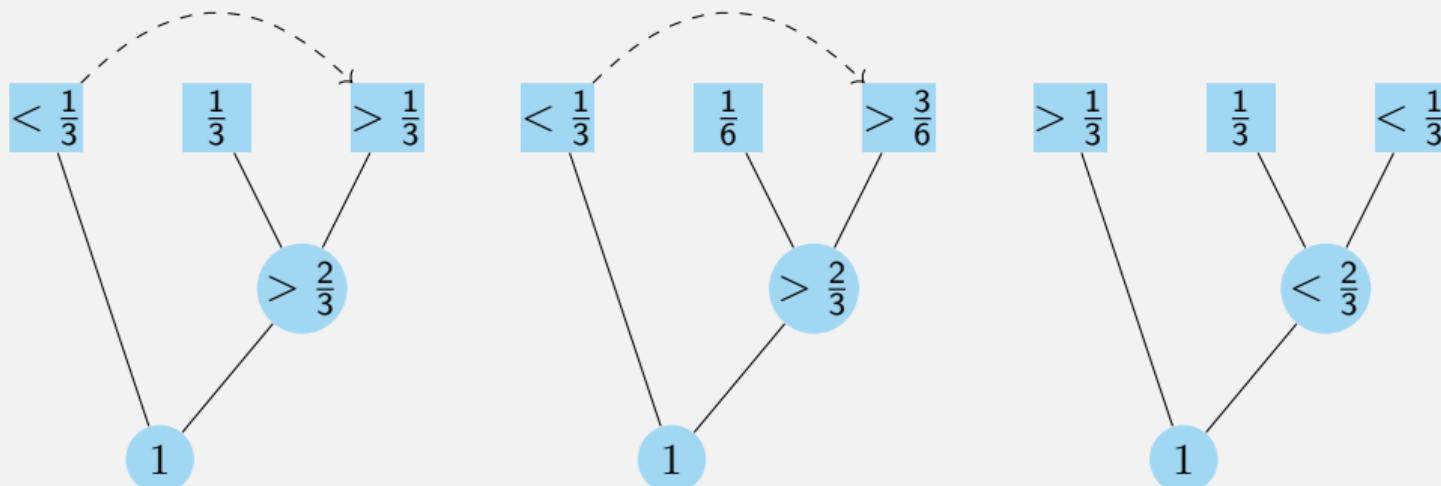


- ▶ Wegen $|c(ai)| = 1$ ist a_i einer der beiden direkten Vorgänger der Wurzel des Codebaumes. Für den anderen Vorgänger v gilt:
 - a) Er ist mit einer Häufigkeitssumme $> 2/3$ beschriftet, da die Summe an der Wurzel 1 ist und $p(a_i) < 1/3$.
 - b) Der Vorgänger v ist kein Blatt wegen $m \geq 3$.
- ▶ Also hat v mindestens einen Vorgänger v_1 mit Häufigkeitssumme größer als $\frac{1}{3}$. Der Algorithmus hätte dann aber zum Zeitpunkt des Einfügens von v nicht v_1 , sondern einen Knoten mit einer kleineren Häufigkeitssumme als Vorgänger von v auswählen müssen. Ein solcher Kandidat wäre z.B. a_i mit $p(a_i) < 1/3$ gewesen.
- ▶ Widerspruch!

Aufgabe 3 II

Huffman-Kodierung

Lösung 3.1



Bonus

Task 4 |

Beweis durch Widerspruch

- ▶ anstatt einen mathematischen Satz S direkt zu beweisen, kann man seine Negation $\neg S$ durch logische Schlussfolgerungen zu einem Widerspruch führen.
- ▶ Wenn man die Widerspruchsanahme $\neg S$ zu einem Widerspruch geführt hat, weiß man, dass $\neg S$ immer falsch sein muss. Damit ist die doppelte Negation $\neg\neg S$ von S wahr. Da $\neg\neg S \Leftrightarrow S$ eine Tautologie ist, ist $\neg\neg S$ genau dann wahr, wenn S wahr ist. Damit muss S wahr sein.
- ▶ Zerlegung der Implikation:

$$\begin{aligned} & Voraussetzung_1 \wedge \cdots \wedge Voraussetzung_n \rightarrow Behauptung \\ \Leftrightarrow & \neg(Voraussetzung_1 \wedge \cdots \wedge Voraussetzung_n) \vee Behauptung \\ \Leftrightarrow & \neg((Voraussetzung_1 \wedge \cdots \wedge Voraussetzung_n) \wedge \neg Behauptung) \\ \Leftrightarrow & Voraussetzung_1 \wedge \cdots \wedge Voraussetzung_n \wedge \neg Behauptung \rightarrow \perp \end{aligned}$$

Task 4 II

Beweis durch Widerspruch

- ▶ Anders gesagt: Aus den Voraussetzungen folgt logisch die Behauptung *genau dann wenn* $Voraussetzung_1 \wedge \dots \wedge Voraussetzung_n \wedge \neg\text{Behauptung}$ NICHT gilt.

Anmerkungen

- ▶ es gibt neben expliziten Voraussetzungen auch implizite Voraussetzungen, die nicht ausdrücklich genannt werden (Rechenregeln und Standard-Definitionen)
- ▶ In der Behauptung steht immer eine wahre Aussage. Hat man eine Aussage, die nicht wahr ist, muss man sie negiert in die Behauptung schreiben.

Task 4 III

Beweis durch Widerspruch

Behauptung: \mathcal{F}

Beweis: Durch Widerspruch.

Annahme: Die Behauptung wäre falsch, das heißt, $\neg\mathcal{F}$ würde gelten.

Teilbeweis unter Verwendung dieser Annahme, der mit einem Widerspruch endet.

Die Annahme gilt also nicht, deshalb gilt die Behauptung.

Task 4 IV

Beweis durch Widerspruch

Wir zeigen: Eine natürliche Zahl mit geradem Quadrat ist selbst gerade.

Voraussetzung:

$$(*) \quad \forall n \in \mathbb{N} (\neg \text{gerade}(n) \Leftrightarrow \exists k \in \mathbb{N} \text{ mit } n = 2k + 1) \quad (\text{Eigenschaft von } \text{gerade})$$

Behauptung: $\forall n \in \mathbb{N} (\text{gerade}(n^2) \Rightarrow \text{gerade}(n))$

Beweis: Durch Widerspruch.

Annahme: Die Behauptung wäre falsch, das heißt, es würde gelten:

$$\neg \forall n \in \mathbb{N} (\text{gerade}(n^2) \Rightarrow \text{gerade}(n)).$$

$$\begin{aligned} &\Rightarrow \exists n \in \mathbb{N} [\text{gerade}(n^2) \wedge \neg \text{gerade}(n)] && (\text{Rechenregeln für } \neg) \\ &\Rightarrow \exists n \in \mathbb{N} [\text{gerade}(n^2) \wedge \exists k \in \mathbb{N} \text{ mit } n = 2k + 1] && (\text{wegen } (*)) \\ &\Rightarrow \exists n \in \mathbb{N} [\text{gerade}(n^2) \wedge \exists k \in \mathbb{N} \text{ mit } n^2 = (2k + 1)^2] && (\text{Quadrierung}) \\ &&& = 4k^2 + 4k + 1 && (\text{arithmetische Umformung}) \\ &&& = 2(2k^2 + 2k) + 1] && (\text{arithmetische Umformung}) \\ &\Rightarrow \exists n \in \mathbb{N} [\text{gerade}(n^2) \wedge \neg \text{gerade}(n^2)] && (\text{wegen } (*)) \\ &\qquad\qquad\qquad \text{Widerspruch.} \end{aligned}$$

Die Annahme gilt also nicht, deshalb gilt die Behauptung. ■

Tutorat 3

Kodierung von Zahlen, Zahlensysteme, Darstellung negativer
Festkommazahlen

Gruppe 9

Präsentator:
Jürgen Mattheis
(juergmatth@gmail.com)

Vorlesung von:
Prof. Dr. Scholl

Übungsgruppenbetreuung:
Tobias Seufert

10. Juni 2023

Universität Freiburg, Lehrstuhl für Rechnerarchitektur

Gliederung

Organisatorisches

Aufgabe 1

Aufgabe 2

Aufgabe 3

Appendix

Literatur

Organisatorisches

Organisatorisches

Abgaben

- ▶ schreibt bitte euren Namen und Matrikelnummer auf die Abgaben
- ▶ Vorrechnen nicht vergessen:
 1. entweder generell immer mal wieder Melden, dann zählt das irgendwann als Vorrechnen
 2. oder vor dem Tutorat ansprechen, dann werdet ihr während des Tutorats dazu gefragt, ob ihr zu einer Aufgabe vielleicht irgendetwas gehaltvolles sagen könnt
- ▶ um die Leute, die in die Tutorate kommen zu belohnen und dem Ereignis entgegenzuwirken, dass das Tutorat irgendwann leer ist, werden die Folien während des Tutorats auf einem USB-Stick verteilt
 - ▶ die Folien aller bisherigen Tutorate werden immer auch alle auf dem USB-Stick sein

Aufgabe 1

Aufgabe 1 |

Zahlensysteme und Darstellung negativer Kommazahlen

Lösung 1.1



$$\begin{aligned}2342_{10} &= 100100100110_2 \\&= [0100100100110]_{BV} \\&= [0100100100110]_1 \\&= [0100100100110]_2\end{aligned}$$

Aufgabe 1 II

Zahlensysteme und Darstellung negativer Kommazahlen

Lösung 1.2



$$\begin{aligned}-BFCD_{16} &= -101111111001101_2 \\&= [1101111111001101]_{BV} \\&= [1010000000110010]_1 \\&= [1010000000110011]_2\end{aligned}$$

Aufgabe 1 III

Zahlensysteme und Darstellung negativer Kommazahlen

Lösung 1.3



$$\begin{aligned}-10_{16} &= -10000_2 \\&= [110000]_{BV} \\&= [101111]_1 \\&= [110000]_2\end{aligned}$$

Aufgabe 1 IV

Zahlensysteme und Darstellung negativer Kommazahlen

Lösung 1.4



$$\begin{aligned} 255_8 &= 10101101_2 \\ &= [010101101]_{BV} \\ &= [010101101]_1 \\ &= [010101101]_2 \end{aligned}$$

Anmerkungen

- ▶ Most Significant Bits, die 0 sind fallen weg

Aufgabe 2

Aufgabe 2 |

Darstellung von Festkommazahlen - Invertieren

Voraussetzungen 2.1



- ▶ *Geometrische Summenformel:* $\sum_{k=0}^n q^k = \frac{1 - q^{n+1}}{1 - q}$
- ▶ *wie im Beweis auf nächster Folie:* $\sum_{k=0}^{n-1} q^k = \frac{1 - q^{n-1+1}}{1 - q} = \frac{1 - q^n}{1 - q} \cdot \frac{-1}{-1} = \frac{q^n - 1}{q - 1}$
- ▶ *für $q = 2$:* $\sum_{k=0}^n 2^k = \frac{1 - 2^{n+1}}{1 - 2} = \frac{1 - 2^{n+1}}{-1} = 2^{n+1} - 1$

Aufgabe 2 II

Darstellung von Festkommazahlen - Invertieren

Lösung 2.1



$$Z_2: [a']_2 + 1 + [a]_2 = 0$$

$$\begin{aligned}
 [a']_2 + 1 + [a]_2 &= \sum_{i=0}^{n-1} a'_i 2^i - a'_n 2^n + 1 + \sum_{i=0}^{n-1} a_i 2^i - a_n 2^n \\
 &= \sum_{i=0}^{n-1} (a'_i + a_i) 2^i - (a'_n + a_n) 2^n + 1 \\
 &= \sum_{i=0}^{n-1} 1 \cdot 2^i - 1 \cdot 2^n + 1 \quad (a'_i + a_i = 1 \text{ da } a'_i = a_i \text{ invertiert}) \\
 &= \frac{2^n - 1}{2 - 1} - 2^n + 1 \\
 &= 2^n - 1 - 2^n + 1 \\
 &= 0
 \end{aligned}$$

Aufgabe 3

Aufgabe 3 |

Darstellung von Festkommazahlen - Dezimalsystem

Lösung 3.1



- ▶ Für $n = 4$: 09999, für $n > 0$: $10^n - 1$ (*größte darstellbare Zahl*)
- ▶ $x = 10^n - 1$, $[d_nd_{n-1} \dots d_0]_9 = \sum_{i=0}^{n-1} d_i \cdot 10^i - d_n \cdot x$
- ▶ Für einen symmetrischen Zahlenbereich kann man sich erstmal überlegen, dass man zwei Darstellung für die 0 braucht und auch die Negation der größten darstellbaren Zahl darstellbar sein muss. D.h. für z.B. $n = 4$ muss man auch -9999 darstellen können. Und diesen Grenzfall erreicht man, indem man von 0 9999 abzieht.

Aufgabe 3.2



- ▶ Wie das *Komplementieren* der Ziffern in der *Neuner-Komplementdarstellung* definiert werden muss, damit das folgende Lemma gilt:
- ▶ **Lemma:** Sei a eine Festkommazahl im Dezimalsystem, a' die Festkommazahl im Dezimalsystem, die aus a durch Komplementieren aller Ziffern hervorgeht. Dann gilt $[a']_9 + [a]_9 = 0$.

Aufgabe 3 II

Darstellung von Festkommazahlen - Dezimalsystem

Lösung 3.2



- ▶ Das Einerkomplement im Binärsystem interpretiert Zahlen mit führenden 0en als nichtnegative Zahlen und Zahlen mit führenden 1en als negative Zahlen. Um das Einerkomplement einer negativen Zahl in BV-Präsentation zu bekommen werden alle Bits invertiert.
- ▶ Analog dazu kann man das Neunerkomplement im Dezimalsystem bilden. In der Aufgabenstellung ist definiert, dass auch hier die höchstwertigste Ziffer nur 0 oder 1 sein kann.
- ▶ Das Invertieren wäre hier die Ersetzung nach folgender Funktion:

$$\text{inv} : \{0, 1, \dots, 8, 9\} \rightarrow \{0, 1, \dots, 8, 9\} : d_i \mapsto \begin{cases} 9 - d_i & 0 \leq i < n \\ 1 - d_i & i = n \text{ und } d_i \in \{0, 1\} \end{cases}$$

- ▶ Für 01784 ist die komplementierte Zahl: 18215

$$[01784]_0 + [18215]_9 = 1784 + (8215 - 9999) = (1784 + 8215) - 9999 = 9999 - 9999 = 0$$

Aufgabe 3 III

Darstellung von Festkommazahlen - Dezimalsystem

Lösung 3.3



- ▶ Beim Zweier-Komplement wird im Gegensatz zum Einer-Komplement bei der Auswertung negativer Zahlen 1 mehr abgezogen.
- ▶ Analog dazu unterscheidet sich das Zehner-Komplement auch vom Neuner-Komplement durch die zusätzliche Subtraktion von 1 bei der Auswertung negativer Zahlen.
- ▶ Die Argumentation ist, dass die Darstellung von -0 zu -1 geshiftet wird und somit keine doppelte Darstellung der 0, sondern eine durchgehende Darstellung gewährleistet ist. Es wird bei negativen Zahlen also $y = 10^n$ statt $x = 10^n - 1$ abgezogen.

Aufgabe 3 IV

Darstellung von Festkommazahlen - Dezimalsystem

Lösung 3.3



► Beweis:

$$\begin{aligned}[a']_{10} + [a]_{10} + 1 &= \sum_{i=0}^{n-1} a'_i b^i - a'_n 10^n + \sum_{i=0}^{n-1} a_i b^i - a_n b^n + 1 \\&= \sum_{i=0}^{n-1} (a'_i + a_i) 10^i - (a'_n + a_n) 10^n + 1 \\&= \sum_{i=0}^{n-1} (9) \cdot 10^i - (1) \cdot 10^n + 1 \\&= 9 \cdot \sum_{i=0}^{n-1} 10^i - 10^n + 1 \\&= 9 \cdot \frac{10^n - 1}{10 - 1} - 10^n + 1 \\&= 0\end{aligned}$$

Aufgabe 3 V

Darstellung von Festkommazahlen - Dezimalsystem

Aufgabe 3.4



- Wie muss man das Komplementieren definieren bei allgemeiner Basis b , sodass $[a']_{b-1} + [a]_{b-1} = 0$ sowie $[a']_b + [a]_b + 1 = 0$ gelten?

Aufgabe 3 |

Darstellung von Festkommazahlen - Dezimalsystem

Lösung 3.4



- ▶ Damit die Lemmata gelten für beliebige b , muss das Komplementieren einer Ziffer $d_i (i = 0, \dots, n-1)$ für die Basis b als die Differenzrechnung $(b-1) - d_i$ definiert werden.
- ▶ Sowohl in Lemma 1 als auch in Lemma 2 kann man dann $10^{i/n}$ durch $b^{i/n}$ ersetzen, wie im folgenden zu sehen ist:

Aufgabe 3 II

Darstellung von Festkommazahlen - Dezimalsystem

Lösung 3.4



► *1er-Komplement:*

$$\begin{aligned}
 [a']_{b-1} + [a]_{b-1} &= \sum_{i=0}^{n-1} ((b-1) - a_i)b^i - (1 - a_n)(b^n - 1) + \sum_{i=0}^{n-1} a_i b^i - a_n(b^n - 1) \\
 &= \sum_{i=0}^{n-1} ((b-1) - a_i + a_i)b^i - (1 - a_n + a_n)(b^n - 1) \\
 &= \sum_{i=0}^{n-1} (b-1) \cdot b^i - (1) \cdot (b^n - 1) \\
 &= (b-1) \cdot \sum_{i=0}^{n-1} b^i - (b^n - 1) \\
 &= (b-1) \cdot \frac{b^n - 1}{b-1} - b^n + 1 \\
 &= 0
 \end{aligned}$$

Aufgabe 3 III

Darstellung von Festkommazahlen - Dezimalsystem

Lösung 3.4



► *2er-Komplement:*

$$\begin{aligned}
 [a']_b + [a]_b + 1 &= \sum_{i=0}^{n-1} ((b-1) - a_i)b^i - (1 - a_n)b^n + \sum_{i=0}^{n-1} a_i b^i - a_n b^n + 1 \\
 &= \sum_{i=0}^{n-1} ((b-1) - a_i + a_i)b^i - (1 - a_n + a_n)b^n + 1 \\
 &= \sum_{i=0}^{n-1} (b-1) \cdot b^i - (1) \cdot b^n + 1 \\
 &= (b-1) \cdot \sum_{i=0}^{n-1} b^i - b^n + 1 \\
 &= (b-1) \cdot \frac{b^n - 1}{b-1} - b^n + 1 \\
 &= 0
 \end{aligned}$$

Aufgabe 3 IV

Darstellung von Festkommazahlen - Dezimalsystem

Anmerkungen

- ▶ Man sieht das Komplementieren funktioniert unabhängig von 1er- oder 2er-Komplement gleich, da der Unterschied zwischen beiden Darstellungen nur darin liegt, dass man entweder $b^n - 1$ (1er-Komplement) oder b^n (2er-Komplement) subtrahiert um die Symmetrie bei zwei 0en (1er-Komplement) oder einer 0 (2er-Komplement) herzustellen und daher zum Ausgleich bei der einen Darstellung nichts dazuaddieren muss (1er-Komplement) oder 1 dazuaddieren (2er-Komplement) muss um bei der Addition einer Zahl mit ihrem Komplement 0 zu erhalten.
- ▶ Man muss -1 rechnen, da es zwar 10 Zeichen gibt, aber die Zahl mit der höchsten Wertigkeit, also die Zahl 9 auch nur die Wertigkeit 9 hat und nicht 10, da die 0 die Anzahl ist, wo nichtszählbares vorliegt.

Appendix

Appendix I

Zahlendarstellungen zur Basis b - Stellenwertsysteme

- ▶ Zahlensystem $S = (b, Z, \delta)$
 - ▶ Zahlensystem wird durch Anhängen der Basis als Index an die Ziffernfolge $d_{n-1} \dots d_0_b$ vermittelt
 - ▶ Basis $b \in \mathbb{N}, b > 1$ mit welcher für jede Stelle i der Stellenwert b^i berechnet wird
 - ▶ Ziffernmenge Z
 - ▶ Ziffernwertigkeit δ ordnet jeder Ziffer bzw. Symbol ihre Wertigkeit zu

Appendix II

Zahlendarstellungen zur Basis b - Stellenwertsysteme

Anmerkungen 

- ▶ bei der n -stelligen Binärdarstellung einer Zahl werden dem **LSB** und **MSB** jeweils die Stellenwerte b^0 und b^{n-1} zugeordnet
- ▶ Im **Binärsystem / Dualsystem** werden den beiden Ziffern 0 und 1 jeweils die Wertigkeiten 0 und 1 zugeordnet
- ▶ Im **Dezimalsystem** werden den zehn Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8 und 9 jeweils die Wertigkeiten 0 bis 9 in der konventionellen Reihenfolge zugeordnet
- ▶ Im **Hexadezimalsystem** werden den sechzehn Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E und F jeweils die Werte der Dezimalzahlen von 0 bis 15 zugeordnet.
 - ▶ **Eselsbrücken:** Cwölf, Dreizehn, Fünfzehn

- ▶ positiver Wert $\langle d \rangle = \sum_{i=0}^{n-1} \delta(d_i) \cdot b^i$ einer **nicht-negativen Natürlichen Zahl**, wobei $d = d_{n-1} \dots d_0$ mit $d_i \in \mathbb{Z}$ eine **Folge** von n **Ziffern** bzw. Symbolen ist

Appendix I

Zahlendarstellung zur Basis b - Stellenwertsysteme

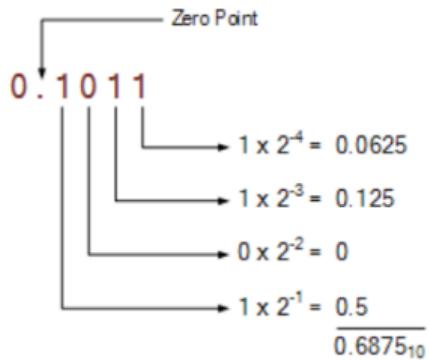
- ▶ positiver Wert $\langle d \rangle = \sum_{i=-k}^{n-1} d_i \cdot b^i$ einer nicht-negativen Festkommazahl, wobei $d = d_{n-1} \dots d_0 \dots d_{-k}$
mit $d_i \in Z$

- ▶ die Anzahl der Nachkommastellen ist fest:
 $\langle d_{n-1} \dots d_0 d_{-1} \dots d_{-k} \rangle \cdot 2^{-k} = \langle d_{n-1} \dots d_0.d_{-1} \dots d_{-k} \rangle$
- ▶ Beispiel 3-Bit Festkommazahlen mit $n = 1$ und $k = 2$:

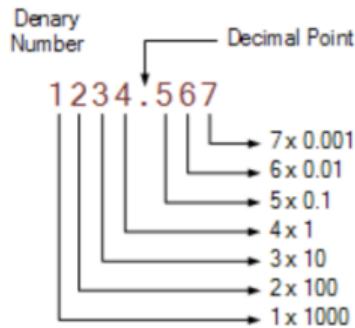
| | | | | | | | | |
|---------------------|------|------|------|------|------|------|------|------|
| d | 0.00 | 0.01 | 0.10 | 0.11 | 1.00 | 1.01 | 1.10 | 1.11 |
| $\langle d \rangle$ | 0.0 | 0.25 | 0.5 | 0.75 | 1.0 | 1.25 | 1.5 | 1.75 |

Appendix II

Zahlendarstellung zur Basis b - Stellenwertsysteme



(a) Binärerbrüche



(b) Dezimalbrüche

- ▶ Darstellung negativer Festkommazahlen, wobei $d = d_n d_{n-1} \dots d_0 \dots d_{-k}$ mit $\forall i \langle n : d_i \in Z$ und $d_n \in \{0, 1\}\}$:

Appendix III

Zahlendarstellung zur Basis b - Stellenwertsysteme

- ▶ potentiell negativer Wert $[d]_{BV} = (-1)^{d_n} \sum_{i=0}^{n-1} \delta(d_i)2^i$ in Darstellung durch Betrag und Vorzeichen
- ▶ Beispiel 3-Bit Festkommazahlen mit Vorzeichenbit, $n = 1$ und $k = 1$:

| | | | | | | | | |
|------------|------|------|------|------|------|------|------|------|
| d | 11.1 | 11.0 | 10.1 | 10.0 | 00.0 | 00.1 | 01.0 | 01.1 |
| $[d]_{BV}$ | -1.5 | -1.0 | -0.5 | 0.0 | 0.0 | 0.5 | 1.0 | 1.5 |

- ▶ potentiell negativer Wert $[d]_1 = \sum_{i=0}^{n-1} \delta(d_i)2^i - \delta(d_n)(2^n - 2^{-k})$ in Einerkomplement-Darstellung
- ▶ Beispiel 3-Bit Festkommazahlen mit Vorzeichenbit, $n = 1$ und $k = 1$:

| | | | | | | | | |
|---------|------|------|------|------|------|------|------|------|
| d | 10.0 | 10.1 | 11.0 | 11.1 | 00.0 | 00.1 | 01.0 | 01.1 |
| $[d]_1$ | -1.5 | -1.0 | -0.5 | 0.0 | 0.0 | 0.5 | 1.0 | 1.5 |

Appendix IV

Zahlendarstellung zur Basis b - Stellenwertsysteme

- ▶ im Negativen hat eine Folge mit mehr 1en anders als im Positiven betragsmäßig eine kleineren Wert, denn umso mehr 1en da sind, umso größer wird die Summe $\sum_{i=0}^{n-1} \delta(d_i)2^i$ und umso mehr kann von der subtrahierten größten positiven Zahl $2^n - 1$ ausgeglichen werden
- ▶ um den Wert einer negativen Dezimalzahl binär darzustellen gibt es 2 Methoden
 1. mit größtmöglichen positiven Zahl anfangen $-(2^n - 1)$ und überlegen, welche 2er Potenzen (Stellenwerte) man draufaddieren muss, um die gewünschte Zahl zu erhalten und an den entsprechenden Bits, die diesen 2er Potenzen entsprechen 1en setzen
 2. die passende Ziffernfolge wie gewohnt erstellen, aber mit vertauschten 1en und 0en und das Vorzeichenbit ist eine 1
- ▶ potentiell negativer Wert $[d]_2 = \sum_{i=0}^{n-1} \delta(d_i)2^i - \delta(d_n)2^n$ in Zweierkomplement-Darstellung

Appendix V

Zahlendarstellung zur Basis b - Stellenwertsysteme

- ▶ Beispiel 3-Bit Festkommazahlen mit Vorzeichenbit, $n = 1$ und $k = 1$:

| | | | | | | | | |
|---------|------|------|------|------|------|------|------|------|
| d | 10.0 | 10.1 | 11.0 | 11.1 | 00.0 | 00.1 | 01.0 | 01.1 |
| $[d]_2$ | -2 | -1.5 | -1.0 | -0.5 | 0.0 | 0.5 | 1.0 | 1.5 |

- ▶ genauso wie beim Einerkompliment bedeuten mehr 1en im Negativen einen betragsmäßig kleineren Wert
- ▶ um den Wert einer negativen Dezimalzahl binär darzustellen gibt es 2 Methoden
 1. mit der größtmöglichen positiven Zahl + kleinmöglichen positiven Zahl ungleich 0 anfangen $-(2^n)$ und überlegen, welche 2er Potenzen (Stellenwerte) man draufaddieren muss, um die gewünschte Zahl zu erhalten und an den entsprechenden Bits, die diesen 2er Potenzen entsprechen 1en setzen

Appendix VI

Zahlendarstellung zur Basis b - Stellenwertsysteme

2. die passende Ziffernfolge wie gewohnt erstellen, aber die kleinstmögliche positive Zahl ungleich 0 wird als Startwert genommen und die 1en und 0en sind vertauscht, sowie das Vorzeichenbit ist eine 1
- wenn man die kleinste negative Zahl komplementiert: $[10.0]_2' = [01.1]_1 + 0.5 = [10.0]_2$, dann erhält man erneut die kleinste negative Zahl. Das passt auch ganz gut, da die kleinste negative Zahl im Zweierkomplement keine komplementäre positive Zahl hat. Folglich auch hier: $[a]_2 + [a']_2 + 2^{-k} = [1.00]_2 + [0.11]_2 + 0.5 = -2 + 1.5 + 0.5 = 0$ für a kleinste Zweierkomplement-Zahl

Appendix VII

Zahlendarstellung zur Basis b - Stellenwertsysteme

Anmerkungen

- ein weiterer Vorteil ist, dass die Einerkomplement- und Zweierkomplement-Darstellung **zyklisch** sind:

$$\begin{aligned}[01.1]_2 + 0.5 &= [10.0]_2 && \text{(nach 1.5 geht es mit } -2 \text{ weiter)} \\ [01.1]_1 + 0.5 &= [10.0]_1 && \text{(nach 1.5 geht es mit } -1.5 \text{ weiter)}\end{aligned}$$

- Darstellung **negativer Gleitkommazahlen**, wobei $\underbrace{d_n}_{sign} \underbrace{d_{n-1} \dots d_0}_{exponent} \underbrace{d_{-1} \dots d_{-k}}_{fraction/mantissa}$ mit exponent, fraction bits und sign bit aus $\{0, 1\}$:
 - varierende Anzahl von Nachkommastellen im Gegensatz zu Festkommazahlen

Appendix VIII

Zahlendarstellung zur Basis b - Stellenwertsysteme

► Normalisierte Zahlen:

$$(-1)^{d_n} \cdot \langle 1d_{-1} \dots d_{-k} \rangle \cdot 2^{-k + \langle d_{n-1} \dots d_0 \rangle - (2^{n-1} - 1)} = (-1)^{\text{sign}} \times (1 + \text{fraction}) \times 2^{\text{exponent} - \text{bias}}$$

► Bias: $2^{n-1} - 1 = \frac{2^2}{2} - 1 = 2 - 1 = \boxed{01_2}$, größter, kleinster Exponent: $11_2 - 01_2 = \boxed{10_2}$,

$$01_2 - 01_2 = \boxed{00_2}$$

► Betragmäßig kleinste, größte Zahl: $\pm 1.0 \times 2^{1-1} = \pm 1.0 (\boxed{\begin{array}{c|c|c} 0 \\ \hline 1 \end{array} | 01 | 0})$, $\pm 1.5 \times 2^{2-1} = \pm 3$

$$(\boxed{\begin{array}{c|c|c} 0 \\ \hline 1 \end{array} | 10 | 1})$$

► Denormalisierte Zahlen:

$$(-1)^{d_n} \cdot \langle 0d_{-1} \dots d_{-k} \rangle \cdot 2^{-k + \langle d_{n-1} \dots d_0 \rangle - (2^{n-1} - 2)} = (-1)^{\text{sign}} \times (0 + \text{fraction}) \times 2^{-\text{bias}}$$

Appendix IX

Zahlendarstellung zur Basis b - Stellenwertsysteme

- ▶ Bias: $2^{n-1} - 2 = \frac{2^2}{2} - 2 = 2 - 2 = \boxed{00_2}$, einziger Exponent: $00_2 - 00_2 = \boxed{00_2}$

- ▶ Betragmäßig kleinste, größte Zahl: $0.0 \times 2^{0-0} = 0 (\boxed{\begin{array}{c|cc|c} 0 & | & 00 & | 0 \\ \hline 1 & \end{array}})$, $\pm 0.5 \times 2^{0-0} = \pm 0.5 (\boxed{\begin{array}{c|cc|c} 0 & | & 00 & | 1 \\ \hline 1 & \end{array}})$

Anmerkungen

- ▶ $1.d_{-1} \dots d_{-k}$ wird als **normalized significand** bezeichnet
- ▶ $0.d_{-1} \dots d_{-k}$ wird als **normed significand** bezeichnet

- ▶ der **Exponent** ist immer als **nicht-negative** Natürliche Zahl zu interpretieren, die **Mantissa** ist immer als **nicht-negativer Bruch** zu interpretieren, also mit 2^{-k} zu multiplizieren

Appendix X

Zahlendarstellung zur Basis b - Stellenwertsysteme

- ▶ der **Bias** macht den *encoded_exponent* immer **positiv**
 - ▶ $\text{encoded_exponent} = \text{real_exponent} + \text{bias}$
 - ▶ $\text{real_exponent} = \text{encoded_exponent} - \text{bias}$
- ▶ Zero: $\begin{array}{c} 0 \\ \hline 1 \end{array} | 00 | 0$, Infinity: $\begin{array}{c} 0 \\ \hline 1 \end{array} | 11 | 0$, NaN: $\begin{array}{c} 0 \\ \hline 1 \end{array} | 11 | 1$
- ▶ Beispiel 4-Bit Gleitkommazahl mit Vorzeichenbit, 2 Exponentbits und Mantissabit:

| | | | | | | | | |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| d | 1 00 0 | 1 00 1 | 1 01 0 | 1 01 1 | 1 10 0 | 1 10 1 | 1 11 0 | 1 11 1 |
| $[d]_{GK}$ | 0.0 | -0.5 | -1.0 | -1.5 | -2.0 | -3.0 | $-\infty$ | NaN |
| d als BV | 0.0 | -0.1 | -1.0 | -1.1 | -10.0 | -11.0 | - | - |

Appendix XI

Zahlendarstellung zur Basis b - Stellenwertsysteme

| | | | | | | | | |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| d | 0 00 0 | 0 00 1 | 0 01 0 | 0 01 1 | 0 10 0 | 0 10 1 | 0 11 0 | 0 11 1 |
| $[d]_{GK}$ | 0.0 | 0.5 | 1.0 | 1.5 | 2.0 | 3.0 | ∞ | NaN |
| d als BV | 0.0 | 0.1 | 1.0 | 1.1 | 10.0 | 11.0 | - | - |

- ▶ Beispiel ohne angepassten Bias (-2) für Denormalisierte Zahlen mit 4-Bit Gleitkommazahl mit Vorzeichenbit, 2 Exponentbits und Mantissabit:

| | | | | | | | | |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| d | 0 00 0 | 0 00 1 | 0 01 0 | 0 01 1 | 0 10 0 | 0 10 1 | 0 11 0 | 0 11 1 |
| $[d]_{GK}$ | 0.0 | 0.25 | 1.0 | 1.5 | 2.0 | 3.0 | ∞ | NaN |
| d als BV | 0.0 | 0.01 | 1.0 | 1.1 | 10.0 | 11.0 | - | - |

- ▶ Beispiel ohne um -1 geshifteten Bias mit 4-Bit Gleitkommazahl mit Vorzeichenbit, 2 Exponentbits und Mantissabit:

| | | | | | | | | |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| d | 0 00 0 | 0 00 1 | 0 01 0 | 0 01 1 | 0 10 0 | 0 10 1 | 0 11 0 | 0 11 1 |
| $[d]_{GK}$ | 0.0 | 0.25 | 0.5 | 0.75 | 1.0 | 1.5 | ∞ | NaN |
| d als BV | 0.0 | 0.01 | 0.1 | 0.11 | 1.0 | 1.1 | - | - |

Appendix XII

Zahlendarstellung zur Basis b - Stellenwertsysteme

- ▶ es gibt mehr positive Exponenten als negative Exponenten
- ▶ die Zahl 1.0 und damit viele Festkommazahlen mit so vielen Mantissa-Bits, wie für die Gleitkommazahl zu Verfügung stehen sind sehr einfach zu kodieren, da
$$(1 + \text{fraction}) \cdot 2^{\sum_{i=0}^{n-2} 1 \cdot 2^i - (2^{n-1} - 1)} = (1 + \text{fraction}) \cdot 2^{2^{n-1}-1 - 2^{n-1} + 1} = (1 + \text{fraction}) \cdot 2^0 = (1 + \text{fraction})$$
- ▶ es gibt verschiedene Standards, wie Bfloat16 (1 sign bit, 8 exponent bits, 7 fraction bits), Single-precision (1 sign bit, 8 exponent bits, 23 fraction bits) und Double-precision (1 sign bit, 11 exponent bits, 52 fraction bits), die sich in der Anzahl der Bits für Exponent und Mantissa unterscheiden
- ▶ Runden mit GRS-Bits:
 - ▶ mit GRS braucht man nur 3 zusätzliche Bits und braucht so keinen großen und langsamen Addierer mit dem man die Berechnungen erstellt. Ohne GRS müsste man die Berechnungen mit allen Bits machen und am Ende runden

Appendix XIII

Zahlendarstellung zur Basis b - Stellenwertsysteme

- ▶ **Guard und Round bit:** Zwei zusätzliche Bits die bei Berechnungen mit Gleitkommazahlen rechts angehängt werden, um die Rundungsgenauigkeit zu erhöhen
- ▶ **Sticky bit:** Zusätzliches Bits rechts der Bits Guard und Round, dass gesetzt wird, wann immer es Bits rechts davon gibt, die nicht 0 sind
- ▶ **Ties to even:** Numbers exactly in the middle between two integer numbers („ties“) are rounded towards the even number

| | G | R | S | Ergebnis |
|----------|-----|-----|-----|----------|
| 0.5 → 0, | | | 1/8 | 0.125 |
| 1.5 → 2, | | | 1/4 | 0.25 |
| 2.5 → 2 | 1/2 | | 1/8 | 0.375 |
| | 1/2 | | | 0.5 |
| | 1/2 | 1/2 | | 0.625 |
| | 1/2 | 1/2 | 1/4 | 0.75 |
| | 1/2 | 1/2 | 1/4 | 0.875 |

Appendix XIV

Zahlendarstellung zur Basis b - Stellenwertsysteme

Anmerkungen 

- ▶ the Anordnung, dass der Exponent immer vor der Mantissa steht und der Fakt, dass der kodierte Exponent positiv ist, ist so gewählt, damit man Gleitkommazahlen möglichst einfach vergleichen kann
- ▶ Festkommazahlen haben eine kleinere Repräsentationsspanne, da sie nur eine feste Anzahl an Nachkommastellen haben. Gleitkommazahlen sind nicht gleichmäßig verteilt, man hat eine sehr hohe Dichte kleiner Zahlen nahe der 0, neben einem schmalen Bereich nahe der 0 mit gar keinen Zahlen.

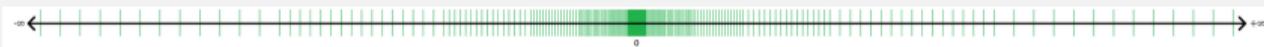


Abbildung 2: Verteilung von Gleitkommazahlen

| | | | | |
|---|--------|-------|------|-----|
| $d[i]$ | 0.101 | 1.01 | 10.1 | 101 |
| $\langle d[i] \rangle$ | 0.625 | 1.25 | 2.5 | 5 |
| $\langle d[i] \rangle - \langle d[i - 1] \rangle$ | 0.3125 | 0.625 | 1.25 | 2.5 |

Tabelle 1: Abstände erhöhen sich exponentiel

Appendix I

Hexadezimalsystem zu Binärsystem

$$\begin{aligned}ba_{16} &= b_{16} \cdot 16^1 + a_{16} \cdot 16^0 \\&= 13_8 \cdot 16^1 + 12_8 \cdot 16^0 \\&= (1_8 \cdot 8^1 + 3_8) \cdot 16^1 + (1_8 \cdot 8^1 + 2_8) \cdot 16^0 \\&= (1_8 \cdot 8^1 + 3_8) \cdot (8 \cdot 2)^1 + (1_8 \cdot 8^1 + 2_8) \cdot (8 \cdot 2)^0 \\&= 2_8 \cdot 8^2 + 6_8 \cdot 8^1 + 1_8 \cdot 8^1 \cdot (8 \cdot 2)^0 + 2_8 \cdot (8 \cdot 2)^0 \\&= 2_8 \cdot 8^2 + 7_8 \cdot 8^1 + 2_8 \cdot 1 = 272_8\end{aligned}$$

$$\begin{aligned}ba_{16} &= b_{16} \cdot 16^1 + a_{16} \cdot 16^0 \\&= 1011_2 \cdot (2^4)^1 + 1010_2 \cdot (2^4)^0 = 10111010_2\end{aligned}$$

Literatur

Online

- [1] *Zahlendarstellung Zur Basis.* URL: <https://www.inf.hs-flensburg.de/lang/informatik/zahlendarstellung.htm> (besucht am 26.05.2023).

Tutorat 4

Nachkommastellen beim den Darstellungen von Festkommazahlen, RETI,
CMOS, p-Kanal und n-Kanal Transistoren

Gruppe 9

Präsentator:
Jürgen Mattheis
(juergmatth@gmail.com)

Vorlesung von:
Prof. Dr. Scholl

Übungsgruppenbetreuung:
Tobias Seufert

22. August 2023

Gliederung

Organisatorisches

Aufgabe 1

Aufgabe 2

Aufgabe 3

Appendix

Organisatorisches

Organisatorisches Abgaben

- ▶ schreibt bitte euren Namen und Matrikelnummer auf die Abgaben
- ▶ Vorrechnen nicht vergessen:
 1. entweder generell immer mal wieder Melden, dann zählt das irgendwann als Vorrechnen
 2. oder vor dem Tutorat ansprechen, dann werdet ihr während des Tutorats dazu gefragt, ob ihr zu einer Aufgabe vielleicht irgendetwas gehaltvolles sagen könnt
- ▶ um die Leute, die in die Tutorate kommen zu belohnen und dem Ereignis entgegenzuwirken, dass das Tutorat irgendwann leer ist, werden die Folien während des Tutorats auf einem USB-Stick verteilt
 - ▶ die Folien aller bisherigen Tutorate werden immer auch alle auf dem USB-Stick sein

Aufgabe 1

Aufgabe 1 |

Nachkommastellen bei den Darstellungen von Festkommazahlen

Voraussetzungen 1.1

```

1 #!/usr/bin/env python
2
3 LENGTH = 4
4 NUM_DECIMAL_BITS = 2
5 NUM_BITS = 4
6
7
8 def value_is_complement(bits):
9     acc = 0
10    i = -1
11    for i, ai in enumerate(map(lambda bit: int(bit), reversed(bits[1:]))):
12        acc += ai * 2 ** (4 - NUM_DECIMAL_BITS)
13    return acc - int(bits[0]) * (2 ** (4 + i - NUM_DECIMAL_BITS) - 2**-2)
14
15
16 def value_2s_complement(bits):
17     acc = 0
18     i = -1
19     for i, ai in enumerate(map(lambda bit: int(bit), reversed(bits[1:]))):
20         acc += ai * 2 ** (4 - NUM_DECIMAL_BITS)
21     return acc - int(bits[0]) * 2 ** (i + 1 - NUM_DECIMAL_BITS)
22
23
24 if __name__ == "__main__":
25     print("n", end="")
26     i = 0
27     while i < 2*NUM_BITS:
28         bits = str(bin(i))[2:]
29         bits = "0" * (LENGTH - len(bits)) + bits
30         print(bits, end="|->(")
31         print(value_is_complement(bits), end=" ")
32         print(value_2s_complement(bits), end=" ")
33         i += 1
34     if i != 2*NUM_BITS:
35         print("n", end="")
36     print(")n", end="")

```

Aufgabe 1 II

Nachkommastellen bei den Darstellungen von Festkommazahlen

Voraussetzungen 1.1



```
> ./value_of_representation.py
{0000->(0.0, 0.0), 0001->(0.25, 0.25), 0010->(0.5, 0.5), 0011->(0.75, 0.75), 0100->(1.0, 1.0), 0101->(1.25, 1.25),
0110->(1.5, 1.5), 0111->(1.75, 1.75), 1000->(-1.75, -2.0), 1001->(-1.5, -1.75), 1010->(-1.25, -1.5),
1011->(-1.0, -1.25), 1100->(-0.75, -1.0), 1101->(-0.5, -0.75), 1110->(-0.25, -0.5), 1111->(0.0, -0.25)}
```

Aufgabe 1 III

Nachkommastellen bei den Darstellungen von Festkommazahlen

Lösung 1.1



- ▶ $[a]_1 = \sum_{i=-k}^{n-1} a_i \cdot 2^i - a_n \cdot (2^n - 2^{-k})$
- ▶ Beim *Einerkomplement* wird, da es zwei 0en gibt wegen der Symmetrie die größtmögliche positive Zahl abgezogen. Diese ist:

$$\begin{aligned}
 \sum_{i=-k}^{n-1} 1 \cdot 2^i &= \sum_{i=0}^{n-1} 1 \cdot 2^i + \sum_{i=-k}^{-1} 1 \cdot 2^i \\
 &= \sum_{i=0}^{n-1} 2^i + \sum_{i=0}^{k-1} 2^{i-k} \\
 &= \sum_{i=0}^{n-1} 2^i + \sum_{i=0}^{k-1} \frac{2^i}{2^k}
 \end{aligned}$$

Aufgabe 1 IV

Nachkommastellen bei den Darstellungen von Festkommazahlen

Lösung 1.1



$$\begin{aligned}&= \sum_{i=0}^{n-1} 2^i + \frac{\sum_{i=0}^{k-1} 2^i}{2^k} \\&= 2^n - 1 + \frac{2^k - 1}{2^k} \\&= 2^n - 1 + 1 - \frac{1}{2^k} \\&= 2^n - 2^{-k}\end{aligned}$$

Aufgabe 1 |

Nachkommastellen bei den Darstellungen von Festkommazahlen

Lösung 1.1



- ▶ $[a]_2 = \sum_{i=-k}^{n-1} a_i \cdot 2^i - a_n \cdot 2^n$
- ▶ Beim **Zweierkomplement** wird, da es nur eine 0 gibt wegen dem dadurch resultierenden asymmetrischen Zahlenbereich (-2^{n-1} ist die größte negative Zahl und $2^{n-1} - 1$ ist die größte positive Zahl) die größtmögliche Zahl größer 0 + die kleinstmögliche Zahl größer 0 (da die zusätzliche 0 nicht mehr eine Kodierung braucht und übersprungen wird, sodass man direkt zur kleinstmöglichen Zahl größer 0 geht, also 2^{-k}) abgezogen:

$$\sum_{i=-k}^{n-1} 1 \cdot 2^i + 2^{-k} = \sum_{i=0}^{n-1} 1 \cdot 2^i + \sum_{i=-k}^{-1} 1 \cdot 2^i + 2^{-k} = 2^n - 2^{-k} + 2^{-k} = 2^n$$

Aufgabe 1b

Lösung 1.2



1. $-1,0_{10} = -32 + 16 + 8 + 4 + 2 + 1 = -0.25 - 0.5 - 0.25 = [111111.00]_2$
2. $2,25_{10} = 2 + 0.25 = [000010.01]_2$
3. $-2.4_{16} = -(2 \cdot 16^0 + 4 \cdot 16^{-1})$
 $= -([0010]_2 \cdot (2^4)^0 + [0100]_2 \cdot (2^4)^{-1}) = -([00100100]_2 \cdot 2^{-4}) = -([0010.0100]_2 \cdot 2^2)$
 $= -[000010.01]_2 = [111101.10]_2 + 2^{-2} = [111101.10]_2 + [000000.01]_2 = [111101.11]_2$
 $\blacktriangleright -2.4_{16} = -(2 \cdot 16^0 + \frac{4}{16}) = -2.25_{10} = -32 + 16 + 8 + 4 + 1 + 0.5 + 0.25_{10} = -0.25 - 2 = [111101.11]_2$
4. *größte darst. Zahl:* $31.75 = 16 + 8 + 4 + 2 + 1 + 0,5 + 0,25 = [011111.11]_2$
5. *größte darst. Zahl < 1:* $0.75 = 0.5 + 0.25 = [000000.11]_2$
6. *größte darst. Zahl < 0:* $-0.25 = -32 + 16 + 8 + 4 + 2 + 1 + 0.5 + 0.25 = -0.25 = [111111.11]_2$

Anmerkungen

- ▶ 8 Bits $d_5 d_4 d_3 d_2 d_1 d_0 d_{-1} d_{-2}$, wobei d_5 sign bit ist, daher $2^5 - 2^{-2} + 2^{-2} = 2^5 = 32$ (größte positive Zahl mit 5 Vorkommastellenbits + kleinster Zahlenabstand mit 2 Nachkommastellenbits)

Aufgabe 2

Aufgabe 2 |

RETI

Voraussetzungen 2.1



- ▶ Seien $a, b \in \mathbb{N}$. Wenn wir wiederholt den *Euklidischer Algorithmus* durchführen:

$$a = b \cdot q_1 + r_1$$

$$b = r_1 \cdot q_2 + r_2$$

$$r_1 = r_2 \cdot q_3 + r_3$$

...

$$r_{n-3} = r_{n-2} \cdot q_{n-1} + r_{n-1}$$

$$r_{n-2} = r_{n-1} \cdot q_n + 0$$

dann gilt: $r_{n-1} = \text{ggt}(a, b)$.

- ▶ $\text{ggt}(a, b) = \text{ggt}(\max(a, b) - \min(a, b), \min(a, b))$ und $\text{ggt}(a, 0) = a$

Aufgabe 2 II

RETI

Voraussetzungen 2.1



- ▶ Beweis: <https://www.youtube.com/watch?v=8cikffEcyPI>
 - ▶ Anmerkung zum Beweis im Video: Wann immer wir eine geordnete Folge von Zahlen haben, deren Wert immer kleiner wird und nichtnegativ sind, können wir sicher sein, dass diese Folge den Wert 0 erreicht.
 - ▶ Man braucht folgende Definition für GGT: Eine Zahl g wird „größter gemeinsamer Teiler“ von a und b genannt, wenn sie die folgenden beiden Bedingungen erfüllt:
 1. g ist a „gemeinsamer Teiler“ von a und b . In anderen Worten: g teilt a und g teilt b
 2. Wenn d ein gemeinsamer Teiler von a und b ist dann: d teilt g
- ▶ Wichtig für RSA, da Primfaktorzerlegung mit der man den GGT zweier Zahlen berechnen kann sehr langsam ist, aber der Euklidische Algorithmus den GGT zweier Zahlen sehr schnell berechnen kann ohne die beiden Zahlen vollständig faktorisieren zu müssen.

Aufgabe 2 III

RETI

Voraussetzungen 2.1



- Was eigentlich gerechnet wird: Division mit Rest

$$a = b \cdot q_1 + r_1$$

eigentlich

$$\frac{a}{b} = q \text{ Rest } r$$

- Algorithmus vereinfacht ausgedrückt:

- Divisor nach vorne

- Rest zum Divisor

- den Quotienten ignorieren

- Euklidischer Algorithmus am Beispiel $a = 12$ und $b = 16$:

$$12 = 16 \cdot 0 + 12$$

$$16 = 12 \cdot 1 + \boxed{4}$$

$$12 = \boxed{4} \cdot 3 + 0$$

dann gilt: $4 = \text{ggt}(12, 16)$.

Aufgabe 2 IV

RETI

Lösung 2.1

```
1 def gcd(front: int, divisor: int) -> int:
2     while divisor:
3         remainder = front % divisor # calculate remainder
4         front = divisor           # divisor to front
5         divisor = remainder        # remainder to divisor
6     return front                  # as soon as the divisor becomes 0, the divisor goes to the front
7
8
9 if __name__ == "__main__":
10    print(gcd(12, 16))
```

- ▶ letzter Durchlauf: $\boxed{4} = 0 \cdot \text{whatever} + 4$

Aufgabe 2 V

RETI

Lösung 2.1



```
1 int ggt(int a, int b) {
2     while(a != b) {
3         if(a < b)
4             b = b-a;
5         else // (a > b)
6             a = a-b;
7     }
8     return a;
9 }
10
11 void main() {
12     print(ggt(16, 12));
13 }
```

Aufgabe 2 VI

RETI

Lösung 2.1



- ▶ Modulo ist q Mal die eine Zahl von der anderen abziehen: $a \bmod b = a - q \cdot b$. Für den GGT muss das Modulo nehmen wie aus der Funktion \gcd bekannt fortgeführt werden. Bei genauer Betrachtung lässt sich das gesamte Vorgehen auf folgende Fälle beschränken:

$$\begin{cases} b = b - a & \text{if } a < b, \\ a = a - b & \text{if } a > b, \\ a \text{ oder } b & \text{if } a = b. \end{cases}$$

bis: $a = b$

- ▶ damit am Ende der Rest 0 sein kann, muss es am Ende der Fall sein, dass $a = b$, da nur so 0 als Rest rauskommen kann, wenn man dann $a - b$ bzw. $b - a$ rechnet

Aufgabe 2 VII

RETI

Lösung 2.1



► Beispiel:

$$20 = 32 - 12$$

$$8 = 20 - 12$$

$$4 = 12 - 8$$

$$\boxed{4} = 8 - \boxed{4}$$

Aufgabe 2 VIII

RETI

Lösung 2.1



```
1 # INITIALISATION
2 LOADI ACC 24
3 STORE ACC 40
4 LOADI ACC 16
5 STORE ACC 41
6 # PROGRAM
7 LOAD ACC 40
8 SUB ACC 41
9 JUMP== 10
10 JUMP> 5
11 LOAD ACC 41
12 SUB ACC 40
13 STORE ACC 41
14 JUMP 4
15 LOAD ACC 40
16 SUB ACC 41
17 STORE ACC 40
18 JUMP -11
19 LOAD ACC 40
20 STORE ACC 42
```

Aufgabe 2 IX

RETI

Lösung 2.1



```

Index:      22
Instruction: SUB ACC 41;
ACC:        8
ACC_SIMPLE: 8
IN1:        0
IN1_SIMPLE: 0
IN2:        0
IN2_SIMPLE: 0
PC:         2147483667
PC_SIMPLE: 10
SP:         2147483693
SP_SIMPLE: 45
BAF:        2147483658
BAF_SIMPLE: 2
CS:         2147483651
CS_SIMPLE: 0
DS:         2147483671
DS_SIMPLE: 23
SRAM:
00000 JUMP 0;
00001 2147483648
00002 0 <= BAF;
00003 LOAD ACC 24; <- CS
00004 STORE ACC 48;
00005 LOAD ACC 16;
00006 STORE ACC 41;
00007 LOAD ACC 48;
00008 SUB ACC 41;
00009 JUMP== 12;
00010 LOAD ACC 48;
00011 SUB ACC 41;
00012 JUMP> 5;
00013 LOAD ACC 41;
00014 SUB ACC 48;
00015 STORE ACC 41;
00016 JUMP 4;
00017 LOAD ACC 48;
00018 SUB ACC 41;
00019 STORE ACC 48; <- PC
00020 JUMP -13;
In0h

```

▶

```

00021 LOAD ACC 48;
00022 STORE ACC 47;
00023 0 <= 05
00024 0
00025 0
00026 0
00027 0
00028 0
00029 0
00030 0
00031 0
00032 0
00033 0
00034 0
00035 0
00036 0
00037 0
00038 0
00039 0
00040 24
00041 16
00042 0
00043 0
00044 0
00045 0 <- SP;
UMAT:
00000 0
00001 0
00002 0
EPROM:
00000 LOAD DS -2097152; <- IN1 <- IN2
00001 MULTI DS 1024;
00002 MOVE DS SP;
00003 MOVE DS BAF;
00004 MOVE DS CS;
00005 ADDI SP 45;
00006 ADDI BAF 2;
00007 ADDI CS 3;
00008 ADDI DS 1; <- ACC
00009 MOVE CS PC;

```

Aufgabe 3

Appendix I

CMOS, P-Kanal und N-Kanal Transistoren

Lösung 3.1



| a | b | c | out |
|---|---|---|-----|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

Tabelle 1: Wertetabelle für boolesche Funktion $\bar{c} \wedge (\bar{a} \vee \bar{b})$

Appendix II

CMOS, P-Kanal und N-Kanal Transistoren

Lösung 3.1

► DNF: $\bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b}\bar{c}$

Karnaugh Map:

| | | bc | $\bar{b}\bar{c}$ | $\bar{b}c$ | $b\bar{c}$ | $b\bar{c}$ |
|-----|-----------|-----------|------------------|------------|------------|------------|
| | | a | 1 | 0 | 0 | 1 |
| | | \bar{a} | 1 | 0 | 0 | 0 |
| a | \bar{a} | | | | | |

► Oder Webseite zur Hilfe nehmen: <http://www.32x8.com/index.html>

Appendix III

CMOS, P-Kanal und N-Kanal Transistoren

Lösung 3.1



► Vereinfachter Boolescher Ausdruck (1en): $\bar{b}\bar{c} + \bar{a}\bar{c} = \bar{c} \cdot (\bar{a} + \bar{b})$

| a | b | c | $\bar{b}\bar{c} + \bar{a}\bar{c}$ |
|---|---|---|-----------------------------------|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

Appendix IV

CMOS, P-Kanal und N-Kanal Transistoren

Lösung 3.1

► **DNF:** $(\bar{a} \cdot \bar{b} \cdot c) + (\bar{a} \cdot b \cdot c) + (a \cdot \bar{b} \cdot c) + (a \cdot b \cdot \bar{c}) + (a \cdot b \cdot c)$

Karnaugh Map:

| | | bc | $\bar{b}\bar{c}$ | $\bar{b}c$ | bc | $b\bar{c}$ |
|-----------|-----------|------|------------------|------------|------|------------|
| | | a | 1 | 0 | 0 | 1 |
| \bar{a} | \bar{a} | 1 | 0 | 0 | 1 | 0 |
| | a | 1 | 0 | 0 | 0 | 0 |

Appendix V

CMOS, P-Kanal und N-Kanal Transistoren

Lösung 3.1



► Vereinfachter Boolescher Ausdruck (0en): $c + ab$

| a | b | c | $c + ab$ |
|---|---|---|----------|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Appendix VI

CMOS, P-Kanal und N-Kanal Transistoren

Lösung 3.1

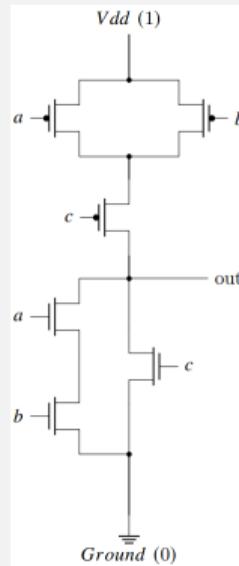


- ▶ auf der einen Seite mit den P-Kanal-Transistoren wird der vereinfachte Ausdruck umgesetzt, der für genau die Modelle wahr ist, für welche die logische Funktion $\bar{c} \cdot (\bar{a} + b)$ wahr ist.
- ▶ Auf der anderen Seite mit den N-Kanal Transistoren wird der vereinfachte Ausdruck umgesetzt, der für genau die Modelle wahr ist, für welche die logische Funktion $\bar{c} \cdot (\bar{a} + \bar{b})$ falsch ist.
- ▶ damit wird sichergestellt, dass es zu keinem Kurzschluss kommen kann, weil immer nur entweder die eine oder anderen Seite ein Signal 0 (Ground) oder 1 (Vdd) zu out durchlässt, aber niemals beide, da der eine mit P-Kanal-Transistoren umgesetzte Ausdruck für genau die Modelle wahr ist, für die der andere mit N-Kanal-Transistoren umgesetzte Ausdruck falsch ist.

Appendix VII

CMOS, P-Kanal und N-Kanal Transistoren

Lösung 3.1



Appendix

Appendix I

Hinweis zu Hypercubes und Karnaugh Map

- Wenn sich zwei Knoten nur durch genau 1 Bit unterscheiden, dann werden sie durch eine Kante verbunden.

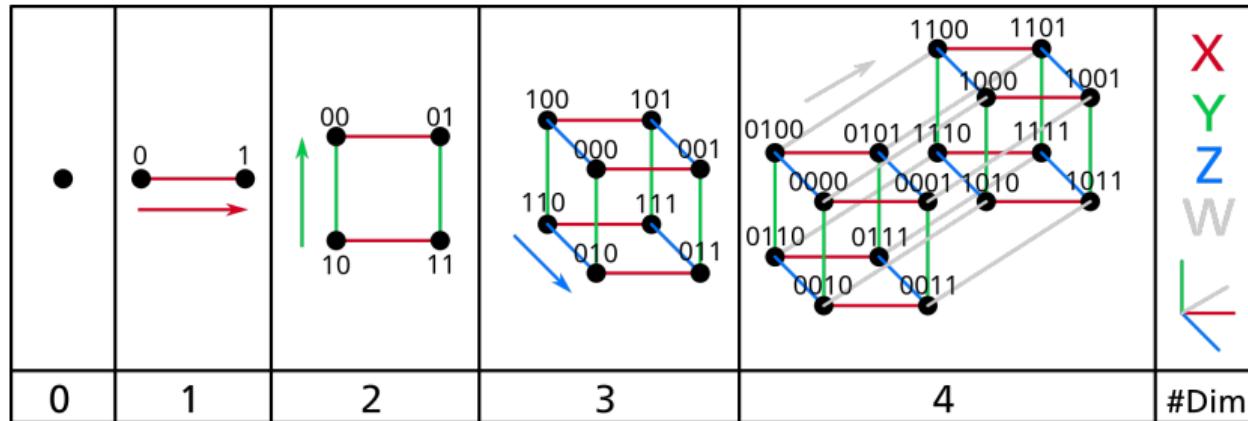
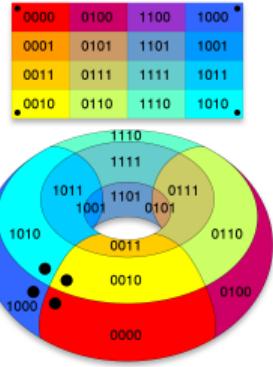


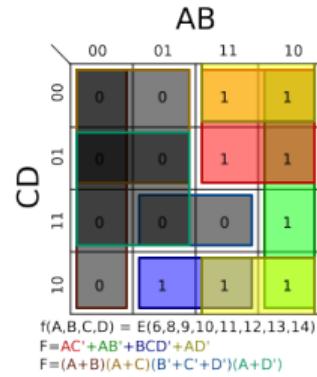
Abbildung 1: Hypercubes von $n = 0$ bis $n = 4$

Appendix II

Hinweis zu Hypercubes und Karnaugh Map



(a) Torus oder umgangssprachlich Donut



(b) Beispiel für Karnaugh Map

- ▶ Anordnung ist so, weil man einen Hypercube bis $n = 4$ (Tesseract) in einer flachen Tabelle darstellen will und im Hypercube gibt es immer nur eine Kante, wenn es nur genau 1 Bit Unterschied bei 2 Knoten gibt.
- ▶ In der Karnaugh Map sind entsprechend immer nur Bitvektoren benachbart, die sich nur in einem Bit unterscheiden.

Appendix III

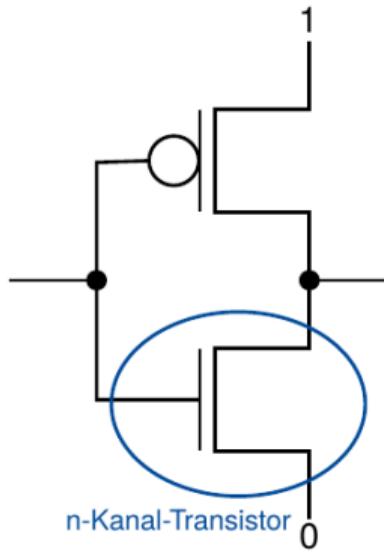
Hinweis zu Hypercubes und Karnaugh Map

- ▶ Wie bei einem Donut sind Reihen und Spalten miteinander von unten nach oben und von links nach rechts über die Tabelle hinausgehend benachbart.
- ▶ Von $\bar{a}b$ zu ab gibt es nur 1 Änderung. Von $\bar{a}b$ zu $a\bar{b}$ gibt es dagegen 2 Änderungen. Daher sind $\bar{a}b$ und ab benachbart.

Appendix I

Kurzschluss

- ▶ $I = \frac{U}{R}$
- ▶ $P = U \cdot I$



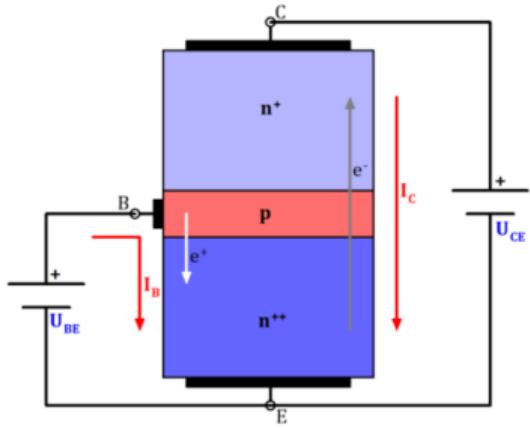
Appendix

Transistoren

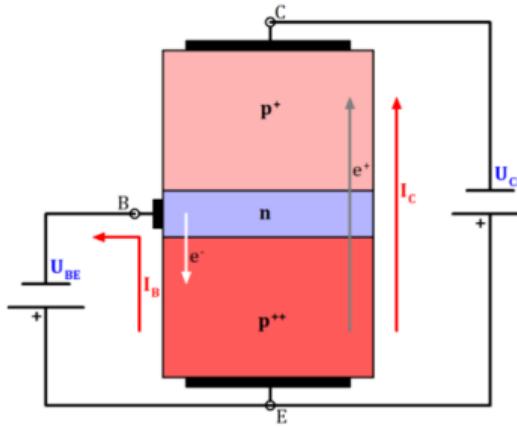
- ▶ Ströme steuern. Man kann den Stromfluss innerhalb einer elektrischen Schaltung abbremsen, sodass überhaupt kein Strom fließt (der Transistor funktioniert als Schalter). Man kann aber den Stromfluss auch stark beschleunigen, wodurch ein viel stärkerer Strom fließt (der Transistor funktioniert als Verstärker).
- ▶ im Kern ist ein Transistor entweder ein strom- oder spannungsgesteuerter Widerstand. Feldeffekttransistoren sind spannungsgesteuerte, Bipolartransistoren sind stromgesteuerte Widerstände.
- ▶ Unterschiede bei Art der **Ladungsträger**, die zum Stromfluss beiträgt. Bei einem **Bipolartransistor** sind das Elektronen und Löcher. Daher kommt auch der Teil „Bi“ im Namen. Bei einem **Feldeffekttransistor** sind das hingegen entweder Elektronen oder Löcher, also nur eine Art an Ladungsträger.
- ▶ Feldeffekttransistoren werden überwiegend überall dort verwendet, wo hohe Ströme , Bipolartransistoren hingegen dort, wo geringe Ströme fließen.

Appendix I

Bipolartransistoren



(a) PNP-Transistor



(b) npn-Transistor

- **drei Anschlüsse:** Collector (C), Base (B) und Emitter (E)

Appendix I

Feldeffekttransistoren

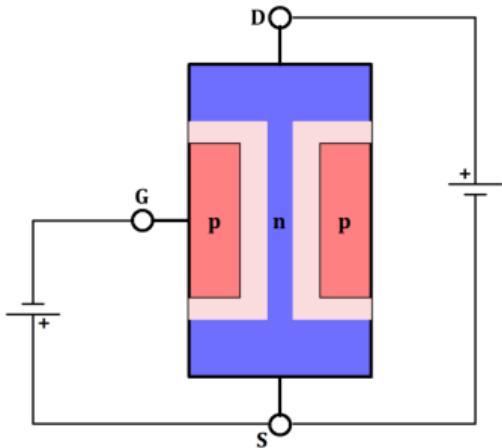


Abbildung 4: n-Kanal-Feldeffekttransistor

- **drei Anschlüsse:** Drain (D, „Senke, Abfluss“), Gate (G, „Tor“) und Source (S, „Quelle, Zufluss“)

Appendix II

Feldeffekttransistoren

- ▶ Sperrsicht-Feldeffekttransistor (SFET, engl. JFET)
- ▶ Je nachdem wie der Kanal, durch denen die Ladungsträger fließen, dotiert ist, findest man die Bezeichnungen n-Kanal-FFET und p-Kanal-FFET

Tutorat 5

Logikfunktionen, Formale Beschreibung von Schaltkreisen, Boolesche Algebra

Gruppe 9

Präsentator:
Jürgen Mattheis
(juergmatth@gmail.com)

Vorlesung von:
Prof. Dr. Scholl

Übungsgruppenbetreuung:
Tobias Seufert

11. Juni 2023

Universität Freiburg, Lehrstuhl für Rechnerarchitektur

Gliederung

Aufgabe 1

Aufgabe 2

Aufgabe 3

Appendix

Literatur

Aufgabe 1

Aufgabe 1 |

Logikfunktionen

Lösung 1.1

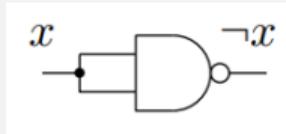


| x | y | f_1 | f_2 | f_3 | f_4 | f_5 | f_6 | f_7 | f_8 | f_9 | f_{10} | f_{11} | f_{12} | f_{13} | f_{14} | f_{15} | f_{16} | |
|-------------------|-----|---|-------|------------------------|-------|-------------------|-------|----------|-------|--|----------|---|----------|---|----------|-----------------|------------|-----------------|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| $x \wedge \neg x$ | | $x \text{ NOR } y, \quad \neg x \wedge y$ | | $\neg x \wedge \neg y$ | | $x \wedge \neg y$ | | $\neg y$ | | $x \text{ XOR } y, \quad (\neg x \wedge y) \vee (x \wedge \neg y)$ | | $x \text{ NAND } y, \quad \neg x \vee \neg y$ | | $x \text{ XNOR } y, \quad (x \wedge y) \vee (\neg x \wedge \neg y)$ | | $\text{BUF}(y)$ | | $\neg x \vee y$ |
| | | | | | | | | | | | | | | $\text{BUF}(x)$ | | $x \vee \neg y$ | $x \vee y$ | $x \vee \neg x$ |

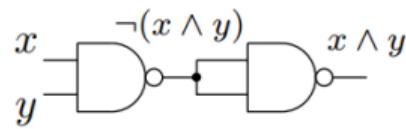
Aufgabe 1 II

Logikfunktionen

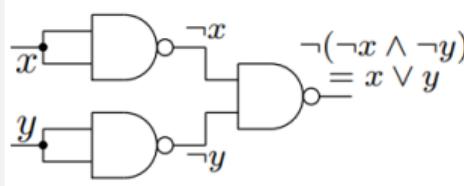
Lösung 1.2



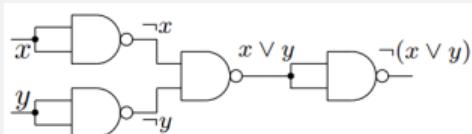
(a) NOT



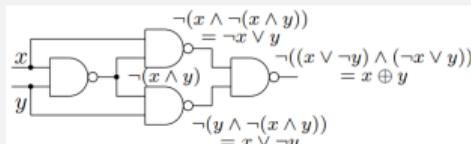
(b) AND



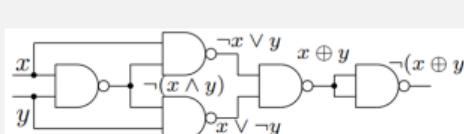
(c) OR



(d) NOR



(e) XOR



(f) XNOR

Aufgabe 1 III

Logikfunktionen

Anmerkungen

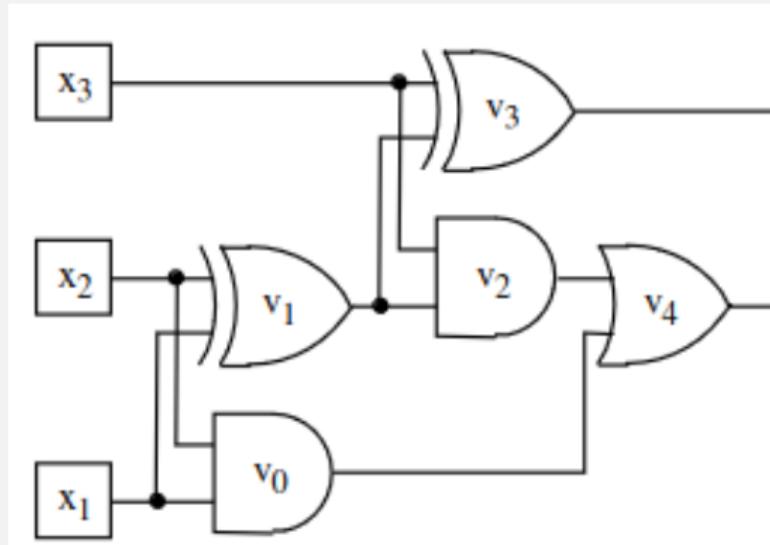
- ▶ Aufgaben a) und b) zusammen zeigen, dass man mit Hilfe des NAND_2 alle Funktionen $f : \mathbb{B}^2 \rightarrow \mathbb{B}$ realisieren kann.

Aufgabe 2

Aufgabe 2 |

Formale Beschreibung von Schaltkreisen

Lösung 2.1



Aufgabe 2 II

Formale Beschreibung von Schaltkreisen Lösung 2.2



| x_1 | x_2 | x_3 | v_3 | v_4 |
|-------|-------|-------|-------|-------|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Aufgabe 3

Aufgabe 3 |

Boolesche Algebra

Lösung 3.1



| x | y | $\neg x$ | $\neg y$ | $\neg x \vee \neg y$ | $x \wedge y$ | $\neg(x \wedge y)$ | $\neg x \wedge \neg y$ | $x \vee y$ | $\neg(x \vee y)$ |
|-----|-----|----------|----------|----------------------|--------------|--------------------|------------------------|------------|------------------|
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

Appendix

Appendix I

Radix-Komplement

- Das Radix-Komplement (r 's Komplement) von einer Zahl y mit n -Ziffern mit Radix b :

$$b^n - y$$

- Es gibt auch ein verringertes Radix-Komplement (($r-1$)'s Komplement), und zwar:

$$(b^n - 1) - y$$

[2]

Appendix II

Radix-Komplement

- ▶ Komplementärverfahren (engl. Method of complements[1]): Subtraktion als Addition berechnen. Beispiele sind:

$$\begin{aligned}622_{10} - 451_{10} \text{ ist } & 622_{10} + (1000_{10} - 451_{10}) - 1000_{10} = 622_{10} + (999_{10} - 451_{10} + 1) - 1000_{10} \\& = 622_{10} + 549_{10} - 1000_{10} \\& = 1171_{10} - 1000_{10} = 171_{10}\end{aligned}$$

$$\begin{aligned}110_2 - 011_2 \text{ ist } & 110_2 + (1000_2 - 011_2) - 1000_2 = 110_2 + (999_2 - 011_2 + 1) - 1000_2 \\& = 110_2 + 101_2 - 1000_2 \\& = 1011_2 - 1000_2 = 11_2\end{aligned}$$

Appendix III

Radix-Komplement

Anmerkungen

- ▶ Das **verringerte Radix-Komplement** kann man leicht erhalten, indem man einfach die Ziffern einer Zahl mit den Ziffern, die man benötigt um $\text{Radix} - 1$ zu erhalten, ersetzt. Zum Beispiel, dass verringerte Radix-Komplement für die 2-Ziffern Dezimalzahl 56 ist 43. Man kann das **Radix-Komplement** einfach erhalten, indem man Eins zu dem verringerten Radix-Komplement addiert $43 + 1 = 44$
- ▶ Im Dezimalzahlensystem ist das Radix-Komplement auch als **Zehnerkomplement** (10'er Komplement) und das verringerte Radix-Komplement als **Neunerkomplement** (9'er Komplement) bekannt
- ▶ Im Binärsystem ist das Radix-Komplement als **Zweierkomplement** (2'er Komplement) und das verringerte Radix-Komplement als **Einerkomplement** (1'er Komplement) bekannt. Ein Einerkomplement kann man einfach durch das Umkehren von Bits einer Zahl erhalten. Zweierkomplemente werden in Computern für die Darstellung von negativen Ganzzahlen verwendet

Appendix I

Halbaddierer

$$\begin{array}{r}
 & a \\
 + & b \\
 \hline
 c_{out} & s
 \end{array}$$

| a | b | c_{out} | s |
|-----|-----|-----------|-----|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

- ▶ der HA kann 2 Inputs zusammenrechnen, daher sind die möglichen Outputs 00, 01 und 10
- ▶ ein HA ist einfach nur die direkte Umsetzung von $a + b = c_{out} \cdot 2^1 + s \cdot 2^0$ nach der obigen Tabelle:

Appendix I

Volladdierer

$$\begin{array}{r}
 & a \\
 & b \\
 + & c_{in} \\
 \hline
 c_{out} & s
 \end{array}$$

| c_{in} | a | b | c_{out} | s |
|----------|-----|-----|-----------|-----|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

- der FA kann 3 Inputs zusammenrechnen, daher sind die möglichen Outputs 00, 01 10 und 11

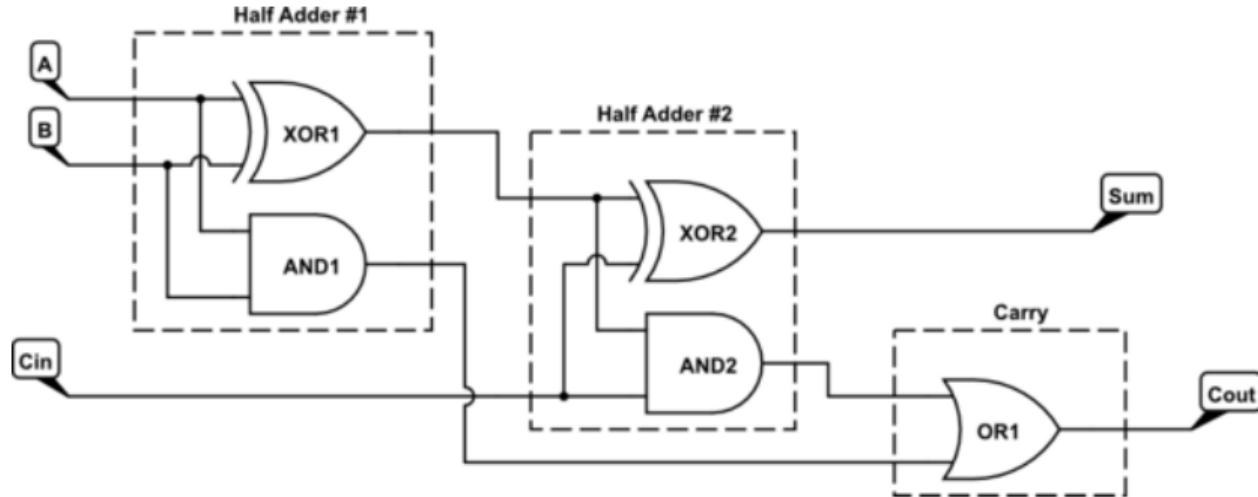
Appendix II

Volladdierer

- ▶ ein FA ist einfach nur die Umsetzung von $a + b + c_{in} = c_{out} \cdot 2^1 + s \cdot 2^0$, ein HA übernimmt $a + b$ und der andere HA übernimmt $s_{a+b} + c_{in}$, wobei es entweder beim ersten HA oder beim zweiten HA die Möglichkeit gibt, dass ein Übertrag entsteht
- ▶ ob die beiden Überträge am Ende ge \oplus rt oder ge \vee rt werden spielt keine Rolle, da es sowieso niemals vorkommen kann, dass beide HA einen Übertrag haben
 - ▶ beim ersten HA kommt es zu einem Übertrag, wenn a und b beide 1 sind
 - ▶ beim zweiten HA kommt es zu einem Übertrag, wenn s und c_{in} beide 1 sind
 - ▶ wenn der erste HA einen Übertrag hat, bedeutet es für den zweiten, dass $s = 0$ sein muss es somit beim zweiten zu keinem Übertrag kommen kann. Bei HA kann es nicht vorkommen, dass $c_{out} = 1$ und $s = 1$
 - ▶ wenn der zweite HA einen Übertrag hat, bedeutet es für den ersten, dass bei diesem $s = 1$ ist und dieser daher keinen Übertrag haben kann

Appendix III

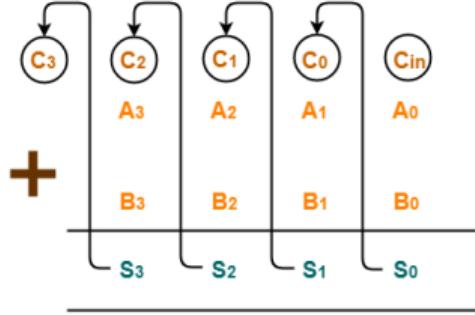
Volladdierer



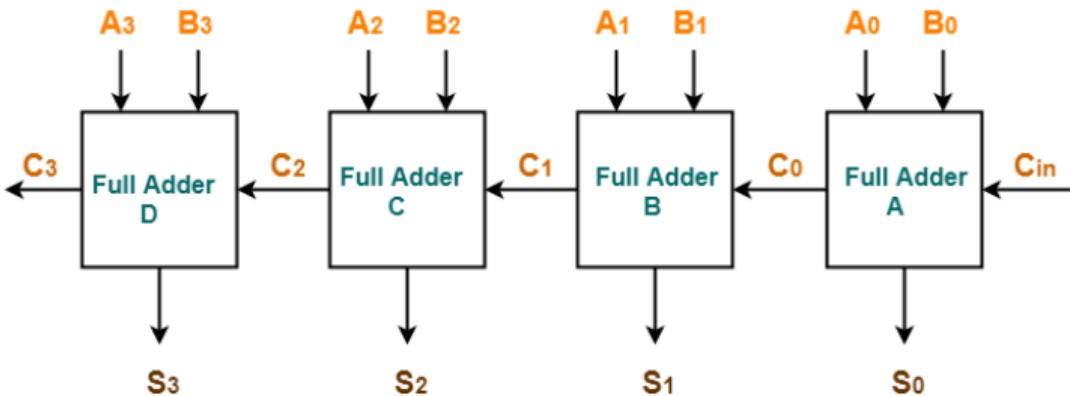
- es spielt bei FA bei der Verwendung keine Rolle, welches der 3 Inputs eigentlich für das Carry vorgesehen ist

Appendix I

4-Bit Addierer



Adding two 4-bit Numbers



4-bit Ripple Carry Adder

- ▶ FA FA FA HA, wobei man allerdings meistens FA FA FA FA herstellt und das carry vom letzten FA unberührt lässt aber theoretisch Addierer zu größeren Addierern zusammenfügen kann

Literatur

Online

- [1] *Method of Complements.* Wikipedia. 29. März 2023. URL:
https://en.wikipedia.org/w/index.php?title=Method_of_complements&oldid=1147184194 (besucht am 08.06.2023).
- [2] *Online-Rechner: Numerische Komplemente.* URL:
<https://de.planetcalc.com/8574/> (besucht am 08.06.2023).

Tutorat 6

Eindeutigkeit des Komplements, Boolesche Algebra, Formale Beschreibung von Schaltkreisen, Programmable Logic Arrays

Gruppe 9

Präsentator:
Jürgen Mattheis
(juergmatth@gmail.com)

Vorlesung von:
Prof. Dr. Scholl

Übungsgruppenbetreuung:
Tobias Seufert

15. Juni 2023

Universität Freiburg, Lehrstuhl für Rechnerarchitektur

Gliederung

Aufgabe 1

Aufgabe 2

Aufgabe 3

Appendix

Aufgabe 1

Aufgabe 1 |

Eindeutigkeit des Komplements, Boolesche Algebra

Lösung 1.1



- ▶ Existenz und Eindeutigkeit neutraler Elemente bereits in Vorlesung gezeigt \rightarrow Es kann $1 = x + \neg x$ verwendet werden
- ▶ Annahme 1: $x + y = 1$
- ▶ Annahme 2: $x \cdot y = 0$
- ▶ Analoges Korollar zur Vorlesung: $1 = x + \neg x$

Aufgabe 1 II

Eindeutigkeit des Komplements, Boolesche Algebra

Lösung 1.1



$$\begin{aligned}y &= y \cdot 1 && (\text{Neutrales Element}) \\&= y \cdot (x + \neg x) && (\text{Korollar}) \\&= (y \cdot x) + (y \cdot \neg x) && (\text{Distributivität}) \\&= (x \cdot y) + (y \cdot \neg x) && (\text{Kommutativität}) \\&= 0 + (y \cdot \neg x) && (\text{Annahme 2}) \\&= (x \cdot \neg x) + (y \cdot \neg x) && (\text{Korollar}) \\&= \neg x \cdot (x + y) && (\text{Kommutativität + Distributivität}) \\&= \neg x \cdot 1 && (\text{Annahme 1}) \\&= \neg x && (\text{Neutrales Element})\end{aligned}$$

Aufgabe 1 III

Eindeutigkeit des Komplements, Boolesche Algebra

Lösung 1.1



$$\begin{aligned}y &= y + 0 && (\text{Neutrales Element}) \\&= y + (x \cdot \neg x) && (\text{Komplement}) \\&= (y + x) \cdot (y + \neg x) && (\text{Distributivität}) \\&= (x + y) \cdot (y + \neg x) && (\text{Kommutativität}) \\&= 1 \cdot (y + \neg x) && (\text{Annahme 1: } x + y = 1) \\&= (x + \neg x) \cdot (y + \neg x) && (\text{Neutrales Element der Konjunktion}) \\&= (\neg x + x) \cdot (\neg x + y) && (\text{Assoziativität}) \\&= \neg x + (x \cdot y) && (\text{Distributivität}) \\&= \neg x + 0 && (\text{Annahme 2: } x \cdot y = 0) \\&= \neg x && (\text{Neutrale Element der Disjunktion})\end{aligned}$$

Aufgabe 1 IV

Eindeutigkeit des Komplements Boolesche Algebra

Lösung 1.1



$$\begin{aligned}
 (x \cdot y) + (\neg x \cdot z) &= (x \cdot y) + ((x \cdot y) \cdot z) + ((\neg x \cdot z) + ((\neg x \cdot z) \cdot y)) && (\text{Absorption}) \\
 &= (x \cdot y) + (\neg x \cdot z) + ((x \cdot y) \cdot z) + ((\neg x \cdot z) \cdot y) && (\text{Kommutativität}) \\
 &= (x \cdot y) + (\neg x \cdot z) + (x \cdot (y \cdot z)) + (\neg x \cdot (z \cdot y)) && (\text{Assoziativität}) \\
 &= (x \cdot y) + (\neg x \cdot z) + (x \cdot (y \cdot z)) + (\neg x \cdot (y \cdot z)) && (\text{Kommutativität}) \\
 &= (x \cdot y) + (\neg x \cdot z) + ((x + \neg x) \cdot (y \cdot z)) && (\text{Kommutativität} + \text{Distributivität}) \\
 &= (x \cdot y) + (\neg x \cdot z) + (y \cdot z) && (\text{Kommutativität} + \text{Komplement})
 \end{aligned}$$

Anmerkungen

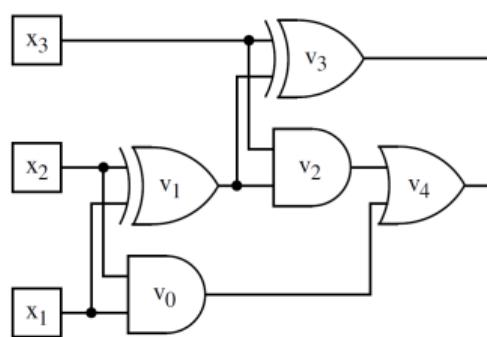
- ▶ Der zweite Teil geht analog durch Verweisen auf das Dualitätsprinzip oder durchrechnen mit genau derselben Axiomanwendung in genau derselben Reihenfolge

Aufgabe 2

Aufgabe 2 |

Formale Beschreibung von Schaltkreisen

Lösung 2.1



Aufgabe 2 II

Formale Beschreibung von Schaltkreisen

Lösung 2.2



| Gatterausgang | Funktion |
|---------------|---|
| v_0 | $x_1 \wedge x_2$ |
| v_1 | $x_1 \oplus x_2$ |
| v_2 | $(x_1 \oplus x_2) \wedge x_3$ |
| v_3 | $(x_1 \oplus x_2) \oplus x_3$ |
| v_4 | $(x_1 \wedge x_2) \vee ((x_1 \oplus x_2) \wedge x_3)$ |

Lösung 2.3



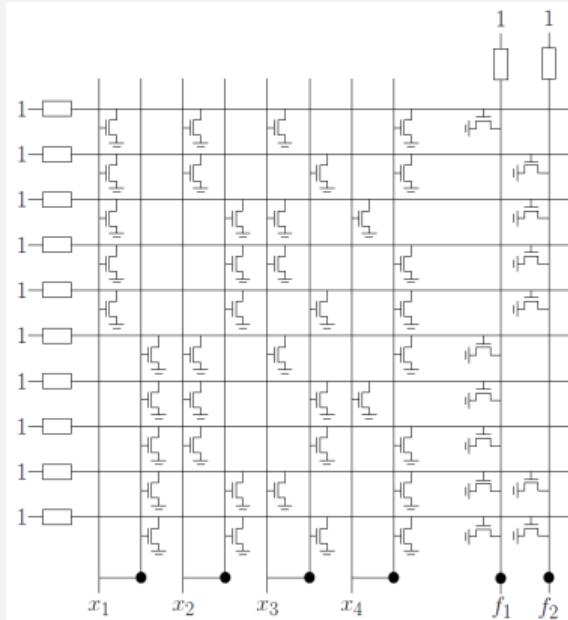
$\text{depth}(C) = 3$, über den Pfad v_1, v_2, v_4

Aufgabe 3

Aufgabe 3 I

Programmable Logic Arrays

Voraussetzungen 3.1



Aufgabe 3 II

Programmable Logic Arrays

Aufgabe 3.1



Gesucht sind die beiden Polynome $f_1, f_2 \in \mathbb{B}_4$, die durch dieses PLA implementiert werden

Lösung 3.1



$$f_1 = \bar{x}_1 \bar{x}_2 \bar{x}_3 x_4 + x_1 \bar{x}_2 \bar{x}_3 x_4 + x_1 \bar{x}_2 x_3 \bar{x}_4 + x_1 \bar{x}_2 x_3 x_4 + x_1 x_2 \bar{x}_3 x_4 + x_1 x_2 x_3 x_4$$
$$f_2 = \bar{x}_1 \bar{x}_2 x_3 x_4 + \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 x_2 \bar{x}_3 x_4 + \bar{x}_1 x_2 x_3 \bar{x}_4 + x_1 x_2 \bar{x}_3 x_4 + x_1 x_2 x_3 x_4$$

Aufgabe 3.2



Kosten des PLA

Aufgabe 3 III

Programmable Logic Arrays

Lösung 3.2



Primäre Kosten $cost_1(f_1, f_2) = 10$ (Zeilen des PLA bzw. Anzahl der Monome)

Sekundäre Kosten $cost_2(f_1, f_2) = 52$ (Zahl der Transistoren im PLA)

Aufgabe 3.3



Hypercubes von f_1 und f_2 zeichnen

Aufgabe 3 IV

Programmable Logic Arrays

Lösung 3.3

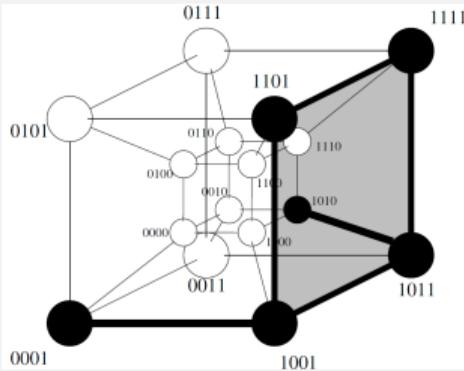


Abbildung 1: Funktion f_1

Aufgabe 3 V

Programmable Logic Arrays

Lösung 3.3

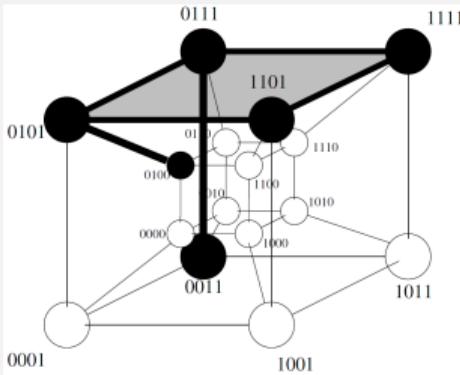


Abbildung 2: Funktion f_2

Appendix

Appendix I

Kleine Korollare

$$\begin{aligned}x + 1 &= (x + 1) \cdot 1 \\&= (x + 1) \cdot (x + \neg x) \\&= x + (1 \cdot \neg x) \\&= x + \neg x \\&= 1\end{aligned}$$

Neutrales Element der Konjunktion

Komplement

Distributivität

Neutrale Element der Konjunktion

Komplement

Appendix II

Kleine Korollare

$$\begin{aligned}x \cdot 0 &= (x \cdot 0) + 0 \\&= (x \cdot 0) + (x \cdot \neg x) \\&= x \cdot (0 + \neg x) \\&= x \cdot \neg x \\&= 0\end{aligned}$$

Neutrales Element der Disjunktion

Komplement

Distributivität

Neutrale Element der Disjunktion

Komplement

Appendix III

Kleine Korollare

$$\begin{aligned}y \cdot y &= y \cdot y + 0 \\&= y \cdot y + y \cdot \neg y \\&= (y + \neg y) \cdot y \\&= 1 \cdot y \\&= y\end{aligned}$$

Neutrales Element

Komplement

Distributivität

Komplement

Neutrales Element

Tutorat 7

Implikanten und Primimplikanten, ON-Menge und Literale,
Quine-McCluskey-Algorithmus, Hypercubes und Kosten eines Polynoms

Gruppe 9

Präsentator:
Jürgen Mattheis
(juergmatth@gmail.com)

Vorlesung von:
Prof. Dr. Scholl

Übungsgruppenbetreuung:
Tobias Seufert

24. August 2023

Gliederung

Aufgabe 1

Aufgabe 2

Aufgabe 3

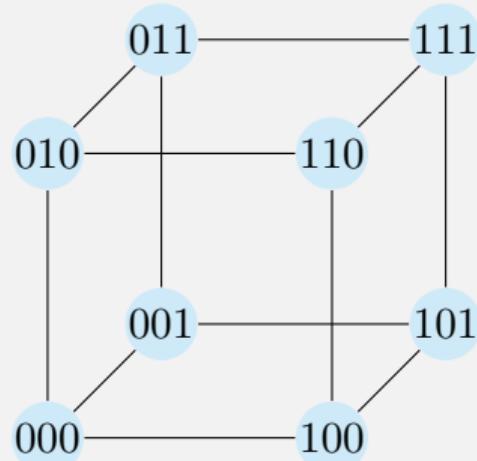
Appendix

Aufgabe 1

Aufgabe 1 |

Implikanten und Primimplikanten

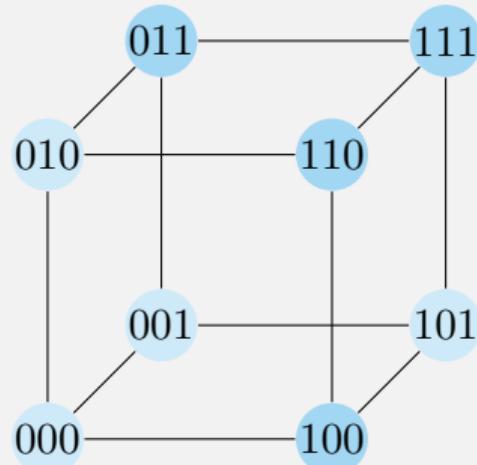
Aufgabe 1.1



Aufgabe 1 II

Implikanten und Primimplikanten

Aufgabe 1.1



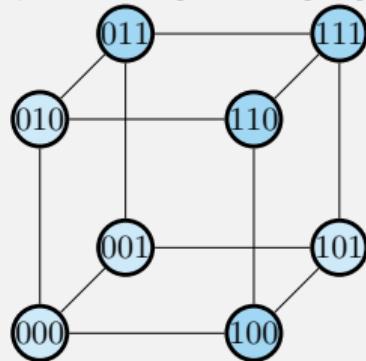
Aufgabe 1 III

Implikanten und Primimplikanten

Lösung 1.1



- Die konstante 1-Funktion ist kein Implikant, es gibt Belegungen $a \in \mathbb{B}^3$ für die $f(a) = 0$

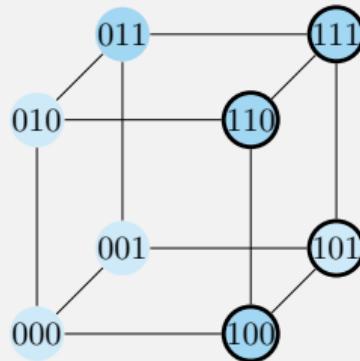


Aufgabe 1 IV

Implikanten und Primimplikanten

Lösung 1.2

- *a ist kein Implikant, also auch kein Primimplikant. Es gilt nicht, dass $\psi(a) \leq f$, denn $\psi(a)(1, 0, 1) = 1 \neq f(1, 0, 1)$*

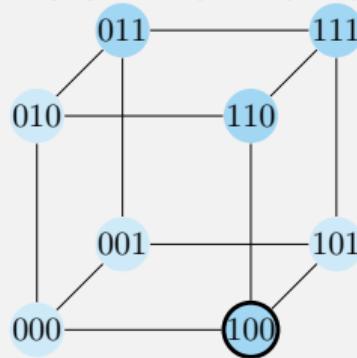


Aufgabe 1 V

Implikanten und Primimplikanten

Lösung 1.3

- $a \cdot \bar{b} \cdot \bar{c}$ ist ein Implikant von f , denn $\psi(a \cdot \bar{b} \cdot \bar{c}) \leq f$ $a \cdot \bar{b} \cdot \bar{c}$ ist aber kein Primimplikant, denn es existiert ein „größerer“ Implikant $a \cdot \bar{c}$ ($\psi(a \cdot \bar{b} \cdot \bar{c}) < \psi(a \cdot \bar{c})$)



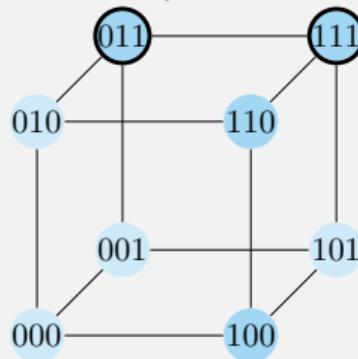
Aufgabe 1 VI

Implikanten und Primimplikanten

Lösung 1.4



- $b \cdot c$ ist ein Primimplikant (und damit natürlich auch ein Implikant). $b \cdot c$ ist maximal, denn es kann kein Literal gestrichen werden, so dass wieder ein Implikant entsteht (weder b noch c sind Implikanten von f)



Aufgabe 2

Aufgabe 2 |

ON-Menge und Literale

- ▶ Behauptung: $m \leq m' \Rightarrow L(m') \subseteq L(m)$
- ▶ sei $L : BE(X_n) \rightarrow \mathcal{P}(\{\bar{s} \mid s \in X_n\} \cup X_n)$

- ▶ Beweis durch Kontraposition:

- ▶ $\nexists z: L(m') \not\subseteq L(m) \Rightarrow ON(m) \not\subseteq ON(m')$
- ▶ es gilt: $L(m) \subset L(m')$, wobei m ein Monom ist und $m' = mx_i^{\omega_i}$
- ▶ man betrachte $(\omega_1, \dots, \omega_i, \dots, \omega_n) \in ON(m)$
- ▶ da $x_i^{\omega_i}$ nicht in m vorkommt, gilt auch $(\omega_1, \dots, \overline{\omega_i}, \dots, \omega_n) \in ON(m)$
- ▶ aber $(\omega_1, \dots, \overline{\omega_i}, \dots, \omega_n) \notin ON(m')$
- ▶ daraus folgt: $ON(m) \not\subseteq ON(m')$
- ▶ daher gilt die Behauptung \square

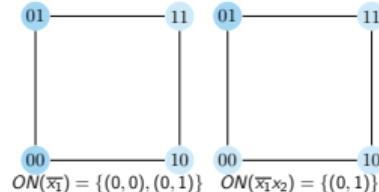


Abbildung 1: Veranschaulichung anhand eines Beispiels

- ▶ Beweis durch Widerspruch:

- ▶ Annahme: $ON(m) \subseteq ON(m') \Rightarrow L(m') \subseteq L(m)$ gilt nicht, also $ON(m) \subseteq ON(m') \wedge L(m) \subset L(m')$
- ▶ es gilt: $L(m) \subset L(m')$, wobei m ein Monom ist und $m' = mx_i^{\omega_i}$
- ▶ man betrachte $(\omega_1, \dots, \omega_i, \dots, \omega_n) \in ON(m)$
- ▶ da $x_i^{\omega_i}$ nicht in m vorkommt, gilt auch $(\omega_1, \dots, \overline{\omega_i}, \dots, \omega_n) \in ON(m)$
- ▶ aber $(\omega_1, \dots, \overline{\omega_i}, \dots, \omega_n) \notin ON(m')$
- ▶ daraus folgt: $ON(m') \subset ON(m)$
- ▶ Widerspruch, denn $ON(m) \subset ON(m')$!
- ▶ die Annahme gilt nicht, also gilt die Behauptung \square

Aufgabe 3

Aufgabe 3 |

Quine-McCluskey-Algorithmus, Hypercubes und Kosten eines Polynoms

Voraussetzungen 3.1



- ▶ $f = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1x_2\bar{x}_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3x_4 + \bar{x}_1x_2x_3\bar{x}_4 + \bar{x}_1x_2x_3x_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3\bar{x}_4 + x_1x_2\bar{x}_3\bar{x}_4 + x_1x_2\bar{x}_3x_4 + x_1x_2x_3\bar{x}_4$

Aufgabe 3 II

Quine-McCluskey-Algorithmus, Hypercubes und Kosten eines Polynoms

Lösung 3.1



1. Schleifeniteration:

$$L_0^{\{x_1, x_2, x_3, x_4\}} \\ 0000$$

$$0001 \\ 0100 \\ 1000$$

$$0101 \\ 0110 \\ 1010 \\ 1100$$

$$0111 \\ 1110$$

$$1111$$

$$\text{Prim} = \emptyset$$

Aufgabe 3 III

Quine-McCluskey-Algorithmus, Hypercubes und Kosten eines Polynoms

Lösung 3.1



2. Schleifeniteration:

$$\begin{array}{r} L_1^{\{x_1, x_2, x_3\}} \\ \hline 000- \\ 010- \\ \hline 011- \\ \hline 111- \end{array}$$

$$\begin{array}{r} L_1^{\{x_1, x_3, x_4\}} \\ \hline 0-00 \\ 0-01 \\ 1-00 \\ \hline 1-10 \end{array}$$

$$\begin{array}{r} L_1^{\{x_1, x_2, x_4\}} \\ \hline 01-0 \\ 10-0 \\ \hline 01-1 \\ 11-0 \end{array}$$

$$\begin{array}{r} L_1^{\{x_2, x_3, x_4\}} \\ \hline -000 \\ -100 \\ -110 \\ \hline -111 \end{array}$$

$$Prim = \emptyset$$

Aufgabe 3 IV

Quine-McCluskey-Algorithmus, Hypercubes und Kosten eines Polynoms

Lösung 3.1



3. Schleifeniteration

$$L_2^{\{x_1, x_2\}} \\ 01-$$

$$L_2^{\{x_2, x_3\}} \\ -11-$$

$$L_2^{\{x_1, x_3\}} \\ 0-0-$$

$$L_2^{\{x_2, x_4\}} \\ -1-0$$

$$L_2^{\{x_1, x_4\}} \\ 1-0$$

$$L_2^{\{x_3, x_4\}} \\ -00$$

$$Prim = \emptyset$$

Aufgabe 3 V

Quine-McCluskey-Algorithmus, Hypercubes und Kosten eines Polynoms

Lösung 3.1



4. Schleifeniteration

$$L_3^{\{x_1\}} = \emptyset$$

$$L_3^{\{x_2\}} = \emptyset$$

$$L_3^{\{x_3\}} = \emptyset$$

$$L_3^{\{x_4\}} = \emptyset$$

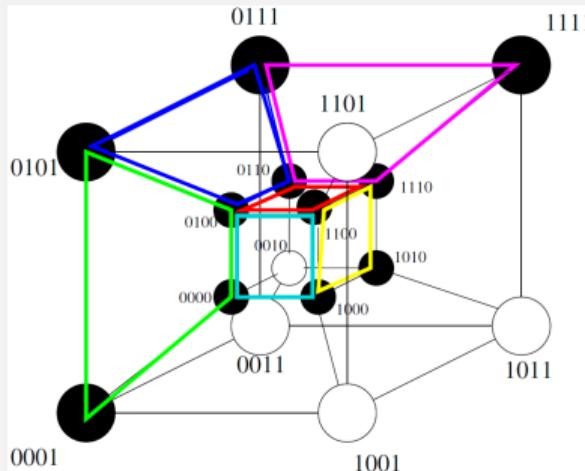
$$\text{Prim} = \{01-, 0-0-, 1-0, -11-, -1-0, -00\}$$

$\cup_M L_3^M(f) = \emptyset \Rightarrow$ Schleifenabbruch, Prim wird zurückgegeben

Aufgabe 3 VI

Quine-McCluskey-Algorithmus, Hypercubes und Kosten eines Polynoms

Lösung 3.2



[01--] [0-0-] [1--0] [-11-] [-1-0] [--00]

Aufgabe 3 VII

Quine-McCluskey-Algorithmus, Hypercubes und Kosten eines Polynoms

Lösung 3.3

- ▶ **primäre Kosten:** # PLA-Zeilen, d.h. #Monome
- ▶ **sekundäre Kosten:** #PLA-Transistoren, d.h. #Literale + #Monome

Lösung 3.3

$$f = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1x_2\bar{x}_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3x_4 + \bar{x}_1x_2x_3\bar{x}_4 + \bar{x}_1x_2x_3x_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3\bar{x}_4 + x_1x_2\bar{x}_3\bar{x}_4 + x_1x_2\bar{x}_3x_4 + x_1x_2x_3\bar{x}_4 + x_1x_2x_3x_4$$

- ▶ $\text{cost}_1 = \#\text{Monome} = 11$
- ▶ $\text{cost}_2 = \#\text{Literale} + \#\text{Monome} = 44 + 11 = 55$

Lösung 3.3

$$f_{red} = \bar{x}_1x_2 + \bar{x}_1\bar{x}_3 + x_1\bar{x}_4 + x_2x_3 + x_2\bar{x}_4 + \bar{x}_3\bar{x}_4$$

- ▶ $\text{cost}_1 = \#\text{Monome} = 6$
- ▶ $\text{cost}_2 = \#\text{Literale} + \#\text{Monome} = 12 + 6 = 18$

Appendix

Appendix I

Verschiedene Interpretationen von Implikation

1. Implikation als If-Statement

- ▶ $a \rightarrow b \Leftrightarrow \neg a \vee b \Leftrightarrow \text{if}(a)\{b\}$, d.h. Lazy Evaluation, b wird nur ausgewertet, wenn $\psi(a)(\omega) = 1$ bzw. $\psi(\neg a)(\omega) = 0$, da 0 der Non-Controlling Value der ODER-Operation ist und daher das Ergebnis erst feststeht, sobald der zweite Operand ausgewertet ist

2. Teilmenge \subseteq

| a | b | f | g | h | $f \rightarrow h$ | $g \rightarrow h$ |
|-----|-----|-----|-----|-----|-------------------|-------------------|
| 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | | | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | | | 1 | 1 | 1 |

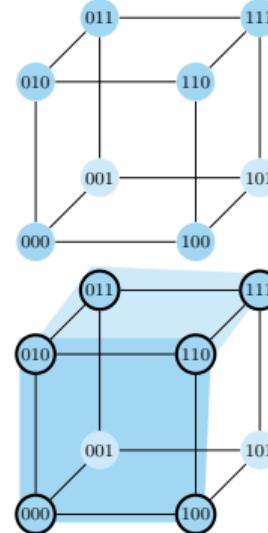
3. Implikant

Appendix II

Verschiedene Interpretationen von Implikation

- ▶ ein **Implikant** von f ist ein Monom q mit $q \leq f$. Ein **Primimplikant** von f ist ein maximaler Implikant q von f , d.h. es gibt keinen Implikanten s ($s \neq q$) von f mit $q \leq s$
- ▶ $ON(f) = \{000, 100, 010, 011, 110, 111\}$
- ▶ $f = \neg a \neg b \neg c \vee a \neg b \neg c \vee \neg ab \neg c \vee \neg abc \vee ab \neg c \vee abc$
- ▶ $f_{red} = b \vee \neg c$
- ▶ Warum nicht $f_{red} = bc \vee \neg c$?

| a | b | c | b | bc | $\neg c$ | $bc \vee \neg c$ | $b \vee \neg c$ |
|-----|-----|-----|-----|------|----------|------------------|-----------------|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |



Appendix I

Beweis durch Kontraposition

- ▶ $\mathcal{F} \rightarrow \mathcal{G} \Leftrightarrow \neg\mathcal{F} \vee \mathcal{G} \Leftrightarrow \neg\neg\mathcal{G} \vee \mathcal{F} \Leftrightarrow \neg\mathcal{G} \rightarrow \neg\mathcal{F}$
- ▶ Kontraposition der Behauptung, die eine Implikation ist wird bewiesen. Immer mit Beweismuster für Implikation kombiniert, da Kontraposition der Behauptung auch eine Implikation ist

Behauptung: $\mathcal{F} \Rightarrow \mathcal{G}$

Beweis: Beweis durch Kontraposition, zu zeigen: $\neg\mathcal{G} \Rightarrow \neg\mathcal{F}$

Es gelte $\neg\mathcal{G}$.

Teilbeweis, dass dann $\neg\mathcal{F}$ gilt.

Appendix II

Beweis durch Kontraposition

Wir zeigen: Eine natürliche Zahl mit geradem Quadrat ist selbst gerade.

Voraussetzung:

$$(*) \quad \forall n \in \mathbb{N} (\neg \text{gerade}(n) \Leftrightarrow \exists k \in \mathbb{N} \text{ mit } n = 2k + 1) \quad (\text{Eigenschaft von } \text{gerade})$$

Behauptung: $\forall n \in \mathbb{N} (\text{gerade}(n^2) \Rightarrow \text{gerade}(n))$

Beweis: Sei $n \in \mathbb{N}$ beliebig.

Beweis durch Kontraposition, zu zeigen: $\neg \text{gerade}(n) \Rightarrow \neg \text{gerade}(n^2)$

Es gelte $\neg \text{gerade}(n)$

$$\Rightarrow \exists k \in \mathbb{N} \text{ mit } n = 2k + 1 \quad (\text{wegen } *)$$

$$\Rightarrow \exists k \in \mathbb{N} \text{ mit } n^2 = (2k + 1)^2 \quad (\text{Quadrierung})$$

$$\begin{aligned} &= 4k^2 + 4k + 1 \\ &= 2(2k^2 + 2k) + 1 \end{aligned} \quad (\text{arithmetische Umformung})$$

$$\Rightarrow \neg \text{gerade}(n^2) \quad (\text{wegen } *)$$

Da $n \in \mathbb{N}$ beliebig gewählt war, gilt die Behauptung. ■

Appendix I

Links

- ▶ <https://www.mathematik.uni-marburg.de/~thormae/lectures/ti1/code/qmc/>
- ▶ <https://we.tl/t-xUFDLiFCy0>
- ▶ <https://we.tl/t-tnBjVtcgZH>

Tutorat 8

Primimplikantentafel, Methode von Petrick, XOR, Quine-McCluskey,
Primimplikanten, Tiefe

Gruppe 9

Präsentator:
Jürgen Mattheis
(juergmatth@gmail.com)

Vorlesung von:
Prof. Dr. Scholl

Übungsgruppenbetreuung:
Tobias Seufert

21. August 2023

Gliederung

Aufgabe 1

Aufgabe 2

Aufgabe 3

Appendix

Aufgabe 1

Aufgabe 1 |

Primimplikantentafel

Aufgabe 1.1



- ▶ On-Menge: $ON(f) = \{0000, 0001, 0100, 0101, 0110, 0111, 1000, 1010, 1100, 1110, 1111\}$
- ▶ Primimplikanten: $Prim(f) = \{--00, -1-0, -11-, 1--0, 0-0-, 01--\}$

Lösung 1.1



| | 0000 | 0001 | 0100 | 0101 | 0110 | 0111 | 1000 | 1010 | 1100 | 1110 | 1111 |
|------|------|------|------|------|------|------|------|------|------|------|------|
| -00 | 1 | | 1 | | | | 1 | | 1 | | |
| -1-0 | | | 1 | | 1 | | | | 1 | 1 | |
| -11- | | | | | 1 | 1 | | | | 1 | 1 |
| 1-0 | | | | | | | 1 | 1 | 1 | 1 | |
| 0-0- | 1 | 1 | 1 | 1 | | | | | | | |
| 01- | | | 1 | 1 | 1 | 1 | | | | | |

Aufgabe 1 |

Primimplikantentafel - Anwenden der Reduktionsregeln in richtiger Reihenfolge

Lösung 1.1



| | 0001 | 0110 | 0111 | 1000 | 1010 | 1100 | 1110 | 1111 |
|------|------|------|------|------|------|------|------|------|
| -00 | | | | 1 | | 1 | | |
| -1-0 | | 1 | | | | 1 | 1 | |
| -11- | | 1 | 1 | | | | 1 | 1 |
| 1-0 | | | | 1 | 1 | 1 | 1 | |
| 0-0- | 1 | | | | | | | |
| 01- | | | | | | | | |

Aufgabe 1 II

Primimplikantentafel - Anwenden der Reduktionsregeln in richtiger Reihenfolge

Lösung 1.1



| | 0001 | 0110 | 0111 | 1000 | 1010 | 1100 | 1110 | 1111 |
|------|------|------|------|------|------|------|------|------|
| -00 | | | | 1 | | 1 | | |
| -1-0 | | | 1 | | | 1 | 1 | |
| -11- | | | 1 | 1 | | | 1 | 1 |
| 1-0 | | | | | 1 | 1 | 1 | 1 |
| 0-0- | 1 | | | | | | | |
| 01- | | | 1 | 1 | | | | |

- 1. Regel: 0-0-, 1-0 und -11- sind wesentlich

Aufgabe 1 III

Primimplikantentafel - Anwenden der Reduktionsregeln in richtiger Reihenfolge

Lösung 1.1



| | |
|------|-------------|
| -00 | \emptyset |
| -1-0 | |
| -11- | |
| 1-0 | |
| 0-0- | |
| 01- | |

- 1. Regel: 0-0-, 1-0 und -11- sind wesentlich
- Leere Tabelle, kein Petrick notwendig
- $f_{min} = \bar{x}_1 \bar{x}_3 + x_1 \bar{x}_4 + x_2 x_3$

Aufgabe 1 |

Primimplikantentafel - Anwenden der Reduktionsregeln in falscher Reihenfolge

Lösung 1.1



| | 0001 | 0110 | 0111 | 1000 | 1010 | 1100 | 1110 | 1111 |
|------|------|------|------|------|------|------|------|------|
| -00 | | | | 1 | | 1 | | |
| -1-0 | | 1 | | | | 1 | 1 | |
| -11- | | 1 | 1 | | | | 1 | 1 |
| 1-0 | | | | 1 | 1 | 1 | 1 | |
| 0-0- | 1 | | | | | | | |
| 01- | | 1 | 1 | | | | | |

► 2. Regel: 0000, 0100, 0101 dominieren 0001 \Rightarrow Lösche 0000, 0100 und 0101

Aufgabe 1 II

Primimplikantentafel - Anwenden der Reduktionsregeln in falscher Reihenfolge

Lösung 1.1



| | 0001 | 1000 | 1010 | 1100 | 1111 |
|------|------|------|------|------|------|
| -00 | | 1 | | 1 | |
| -1-0 | | | | 1 | |
| -11- | | | | | 1 |
| 1-0 | | 1 | 1 | 1 | |
| 0-0- | 1 | | | | |
| 01- | | | | | |

- 2. Regel: 0110, 0111 und 1110 dominieren 1111 \Rightarrow Lösche 0110, 0111 und 1110

Aufgabe 1 III

Primimplikantentafel - Anwenden der Reduktionsregeln in falscher Reihenfolge

Lösung 1.1



| | 0001 | 1010 | 1111 |
|------|------|------|------|
| -00 | | | |
| -1-0 | | | |
| -11- | | | 1 |
| 1-0 | | 1 | |
| 0-0- | 1 | | |
| 01- | | | |

- 2. Regel: 1000 und 1100 dominieren 1010 \Rightarrow Lösche 1000 und 1100

Aufgabe 1 IV

Primimplikantentafel - Anwenden der Reduktionsregeln in falscher Reihenfolge

Lösung 1.1



| | |
|------|-------------|
| -00 | \emptyset |
| -1-0 | |
| -11- | |
| 1-0 | |
| 0-0- | |
| 01- | |

- 1. Regel: 0-0-, 1-0 und -11- sind wesentlich
- Leere Tabelle, kein Petrick notwendig
- $f_{min} = \bar{x}_1 \bar{x}_3 + x_1 \bar{x}_4 + x_2 x_3$

Aufgabe 2

Aufgabe 2 |

Methode von Petrick

Voraussetzungen 2.1



- ▶ *Absorption:* $A + AB = A$
- ▶ *Korollar:* $AA = A$

Aufgabe 2 II

Methode von Petrick

Aufgabe 2.1



| | a | b | c | d | e | f |
|---|---|---|---|---|---|---|
| A | | 1 | | 1 | | |
| B | 1 | | 1 | 1 | | |
| C | | | 1 | 1 | | 1 |
| D | | 1 | | | 1 | |
| E | | | 1 | 1 | 1 | |
| F | 1 | 1 | | | | 1 |

- Minterme: $\{a, b, c, d, e, f\}$
- Primimplikanten: $\{A, B, C, D, E, F\}$

Aufgabe 2 III

Methode von Petrick

Lösung 2.1



| | a | b | c | e | f |
|---|---|---|---|---|---|
| A | | 1 | | | |
| B | 1 | | 1 | | |
| C | | | 1 | | 1 |
| D | | 1 | | 1 | |
| E | | | 1 | 1 | |
| F | 1 | 1 | | | 1 |

- Regel 2: d dominiert c \Rightarrow Lösche d.

Aufgabe 2 IV

Methode von Petrick

Lösung 2.1



| | a | b | c | e | f |
|---|---|---|---|---|---|
| B | 1 | | 1 | | |
| C | | | 1 | | 1 |
| D | | 1 | | 1 | |
| E | | | 1 | 1 | |
| F | 1 | 1 | | | 1 |

- *Regel 3: D (oder F) dominiert A \Rightarrow Lösche A*

Aufgabe 2 V

Methode von Petrick

Lösung 2.1



| | a | b | c | e | f |
|---|---|---|---|---|---|
| B | 1 | | 1 | | |
| C | | | 1 | | 1 |
| D | | 1 | | 1 | |
| E | | | 1 | 1 | |
| F | 1 | 1 | | | 1 |

- ▶ Keine weitere Reduktionsregel mehr anwendbar \Rightarrow Petrick

Aufgabe 2 VI

Methode von Petrick

Lösung 2.1



► *Petrick:*

$$\underbrace{(B+F)}_{\text{überdecken } a} \quad \underbrace{(D+F)}_{\text{überdecken } b} \quad \underbrace{(B+C+E)}_{\text{usw.}} (D+E)(C+F)$$

$$\begin{aligned}
 &= (BD + BF + FD + F) \cdot (BD + BE + CD + CE + ED + E) \cdot (C + F) \\
 &= (BD + BF + FD + F) \cdot (BDC + BEC + CD + CE + EDC + BDF + BEF + CDF + CEF + EDF + EF) \\
 &= (BD + BF + FD + F) \cdot (CD + CE + BDF + EF) \\
 &= BDC + BDCE + BDF + BDEF + BFCD + BFCE + BFD + BFE + FDC + FDCE + FDB + FDE + FCD + FCE + FBD + FE \\
 &= BDC + BDF + BFD + FDC + FBD + FE
 \end{aligned}$$

- Produkt mit den wenigsten Termen auswählen. Wenn es zwei solche Produkte gibt, dann wählt man die Terme mit den wenigsten Literalen
 ► Minimalpolynom $f_{min} = F + E$

Aufgabe 3

Aufgabe 3 |

XOR, Quine-McCluskey, Primimplikanten, Tiefe

Lösung 3.1



- ▶ Minterme: $x'_1x'_2x_3$, $x'_1x_2x'_3$, $x_1x'_2x'_3$, $x_1x_2x_3$
- ▶ Quine-McCluskey:

| $L_0^{x_1, x_2, x_3}$ |
|-----------------------|
| 001 |
| 010 |
| 100 |
| 111 |

- ▶ $\text{Prim}(x_{or_3}) = \emptyset$
- ▶ Es gibt keine Kombinationsmöglichkeiten der Minterme aus L_0 . Daher sind alle L_1 -Mengen leer und die Primmenge im nächsten Schritt enthält alle vier Minterme. Der Grund ist, dass sich alle Minterme in (mindestens) zwei Literalen unterscheiden, d.h. alle Minterme sind Primimplikanten.

Aufgabe 3 II

XOR, Quine-McCluskey, Primimplikanten, Tiefe

Lösung 3.1



| | 1 | 2 | 4 | 7 |
|---------------|---|---|---|---|
| $x_1'x_2'x_3$ | | | 1 | |
| $x_1'x_2x_3'$ | | | | 1 |
| $x_1x_2'x_3'$ | | | | 1 |
| $x_1x_2x_3$ | | | | 1 |

- ▶ Es ist direkt ersichtlich, dass alle Primimplikanten wesentlich sind (Regel 1). Die Disjunktion aller Minterme ist also das einzige Minimalpolynom für xor_3

Aufgabe 3 III

XOR, Quine-McCluskey, Primimplikanten, Tiefe

Lösung 3.2



- ▶ $\bigvee_{\substack{(b_1, \dots, b_n) \in \mathbb{B}^n \\ \text{mit } \sum_{i=1}^n b_i \text{ ungerade}}} (x_1^{b_1} \wedge \dots \wedge x_n^{b_n})$, wobei $x_i^0 := x'_i$ und $x_i^1 := x_i$
- ▶ Der angegebene Ausdruck ist die Disjunktion aller Minterme für xor_n . In Aufgabenteil a) haben wir schon gesehen, dass alle Minterme wesentlich sind um xor_n darzustellen, da weder durch Quine-McCluskey noch Primimplikantentafel Reduktionen erreicht werden können. Deswegen beschreibt auch hier die Disjunktion der Minterme das Minimalpolynom. Es sind 2^{n-1} Primimplikanten, je mit Länge n .

Voraussetzungen 3.3



- ▶ $\text{xor}_n(x_1, \dots, x_n) = g(\text{xor}_i(x_1, \dots, x_i), \text{xor}_{n-i}(x_{i+1}, \dots, x_n))$ für $1 \leq i \leq n-1$

Aufgabe 3 IV

XOR, Quine-McCluskey, Primimplikanten, Tiefe

Lösung 3.3



- Sei $b_1 := \text{xor}_i(x_1, \dots, x_i)$, $b_2 := \text{xor}_{n-i}(x_{i+1}, \dots, x_n)$

- Fall $b_1 = 0, b_2 = 0$:

Dann ist $\sum_{j=1}^i x_j$ gerade, $\sum_{j=i+1}^n x_j$ gerade $\Rightarrow \sum_{j=1}^n x_j$ gerade $\Rightarrow g(0, 0) = 0 = \text{xor}_2(0, 0)$

- Fall $b_1 = 1, b_2 = 0$:

Dann ist $\sum_{j=1}^i x_j$ ungerade, $\sum_{j=i+1}^n x_j$ gerade $\Rightarrow \sum_{j=1}^n x_j$ ungerade $\Rightarrow g(1, 0) = 1 = \text{xor}_2(1, 0)$

- Fall $b_1 = 0, b_2 = 1$:

Dann ist $\sum_{j=1}^i x_j$ gerade, $\sum_{j=i+1}^n x_j$ ungerade $\Rightarrow \sum_{j=1}^n x_j$ ungerade $\Rightarrow g(0, 1) = 1 = \text{xor}_2(0, 1)$

- Fall $b_1 = 1, b_2 = 1$:

Dann ist $\sum_{j=1}^i x_j$ ungerade, $\sum_{j=i+1}^n x_j$ ungerade $\Rightarrow \sum_{j=1}^n x_j$ gerade $\Rightarrow g(1, 1) = 0 = \text{xor}_2(1, 1)$

Aufgabe 3 V

XOR, Quine-McCluskey, Primimplikanten, Tiefe

Lösung 3.3



► Zusammengefasst:

| b_1 | b_2 | g |
|-------|-------|-----|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$\Rightarrow g = xor_2$, also $xor_n(x_1, \dots, x_n) = xor_2(xor_i(x_1, \dots, x_i), xor_{n-i}(x_{i+1}, \dots, x_n))$ für $1 \leq i \leq n - 1$

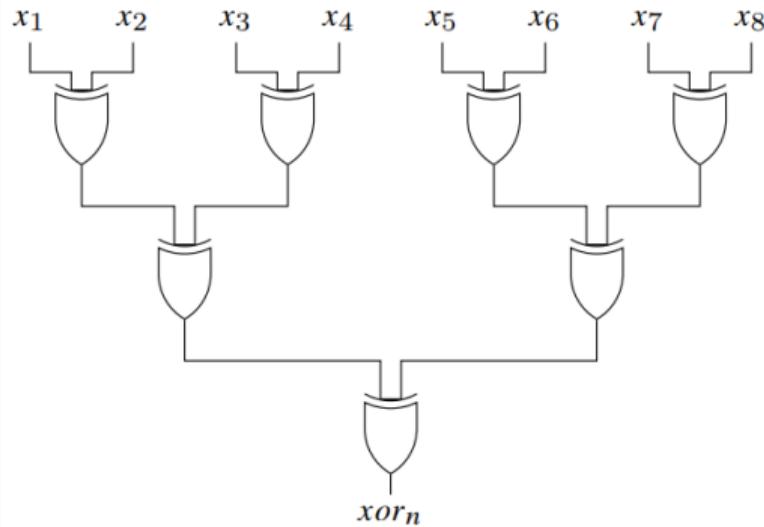
Aufgabe 3 VI

XOR, Quine-McCluskey, Primimplikanten, Tiefe

Lösung 3.4



- ▶ Ein balancierter Baum hat die geringste Tiefe. D.h. wähle $i = \lceil \frac{n}{2} \rceil$



Aufgabe 3 VII

XOR, Quine-McCluskey, Primimplikanten, Tiefe

Lösung 3.5



- ▶ Für einen balancierten Baum ist Kosten = #XOR-Gatter = $n - 1$ und längster Pfad = $\lceil \log(n) \rceil$. Bzw. da vorgegeben, dass $n = 2^k$ gilt, ist auch $\log(n)$ korrekt.

Appendix

Aufgabe 4 |

Primimplikantentafel, 1te Regel

- ▶ da es nur darum geht, dass alle Minterme vom Anfang abgedeckt sind und die wesentlichen Minterme auf jeden Fall genommen werden müssen, werden auch die Minterme, die sie abdecken auf jeden Fall genommen. Aus diesem Grund ist es nicht mehr notwendig diese Minterme in der Primimplikantentafel miteinzubeziehen, da man Minterme nur in der Primimplikantentafel miteinbezieht, wenn man noch eine Abdeckung für diese Minterme sucht.

Aufgabe 4 |

Primimplikantentafel, 2te Regel

| | 010 | 100 | 110 |
|------------------|-----|-----|-----|
| $ab\bar{c}$ | | | 1 |
| $a\bar{c}$ | | 1 | 1 |
| $\bar{a}\bar{c}$ | 1 | 1 | |

| | 010 | 100 | 110 |
|------------------|-----|-----|-----|
| $b\bar{c}$ | 1 | | 1 |
| $a\bar{c}$ | | 1 | 1 |
| $\bar{a}\bar{c}$ | 1 | 1 | |

- ▶ eines der Momome, das den Minterm abdeckt der dominiert wird, wird irgendwann aufgrund weiterer Reduktionen mit der ersten und dritten Reduktionsregel wesentlich und wird dann genommen. Sobald dieses Monom genommen wird ist es aufgrund der Teilmengenbeziehung sicher, dass dieses Monom auch den Minterm der den ersteren Minterm dominiert hat abdeckt. Der zweitere Minterm ist somit nicht mehr von Interesse für die Primimplikantentafel, weil man bereits eine Abdeckung für diesen Minterm gefunden hat und es nur darum geht, dass man alle Minterme vom Anfang abgedeckt hat.

Aufgabe 4 |

Primimplikantentafel, 3te Regel

- ▶ Wenn die On-Menge eines Monoms Teilmenge der On-Menge eines anderen Monoms ist, dann nimmt das Monom mit der größeren On-Menge, da größere ON-Menge weniger Literale bedeutet und somit weniger Kosten. Mit einer größeren On-Menge kann man mehr Minterme auf einmal abdecken und es geht nur darum, dass man am Ende alle Minterme vom Anfang irgendwie abgedeckt hat.

Aufgabe 4 |

Primimplikantentafel am Hypercube

| | 0000 | 0001 | 0100 | 0101 | 0110 | 0111 | 1000 | 1010 | 1100 | 1110 | 1111 |
|------|------|------|------|------|------|------|------|------|------|------|------|
| 0-0- | 1 | 1 | 1 | 1 | | | | | | | |
| -00 | 1 | | 1 | | | | 1 | | 1 | | |
| 01- | | | 1 | 1 | 1 | 1 | | | | | |
| -11- | | | 1 | | 1 | | | | 1 | 1 | |
| -1-0 | | | | | 1 | 1 | | | | 1 | 1 |
| 1-0 | | | | | | | 1 | 1 | 1 | 1 | |

Aufgabe 4 II

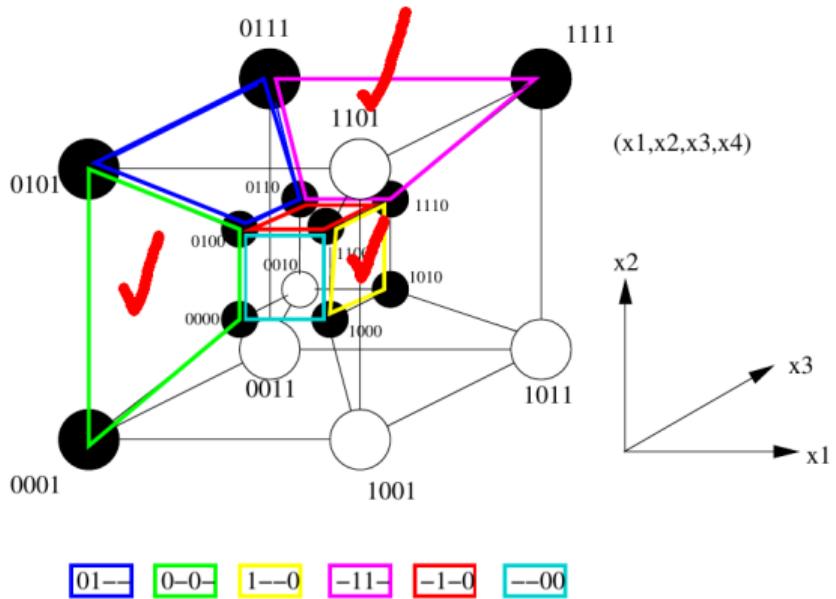
Primimplikantentafel am Hypercube

| | 0000 | 0001 | 0100 | 0101 | 0110 | 0111 | 1000 | 1010 | 1100 | 1110 | 1111 |
|------|------|------|------|------|------|------|------|------|------|------|------|
| 0-0- | 1 | 1 | 1 | 1 | | | | | | | |
| -00 | 1 | | | 1 | | | | 1 | | 1 | |
| 01- | | | 1 | 1 | 1 | 1 | | | | | |
| -11- | | | 1 | | 1 | | | | 1 | 1 | |
| -1-0 | | | | | 1 | 1 | | | | 1 | 1 |
| 1-0 | | | | | | | 1 | 1 | 1 | 1 | |

- ▶ $(\overline{x_0} \overline{x_2}) \vee (x_1 \overline{x_3}) \vee (x_0 \overline{x_3})$, $\{0-0-, -1-0, 1--0\}$

Aufgabe 4 III

Primimplikantentafel am Hypercube



Aufgabe 4 |

Binärbaum, wobei $n = 2^k$

- Anzahl Blätter (vollständiger Binärbaum):

$$n = 2^d = 2^3 = 8$$

- Anzahl Knoten: $k = \sum_{i=0}^d 2^i = \frac{2^{d+1} - 1}{2 - 1} = 2^{d+1} - 1 = 2^{3+1} - 1 = 1 + 2 + 4 + 8 = 15$

- Tiefe mithilfe Anzahl Blätter: $d = \log_2(n) = \log_2(8) = 3$

- Tiefe mithilfe Anzahl Knoten (vollständiger Baum):

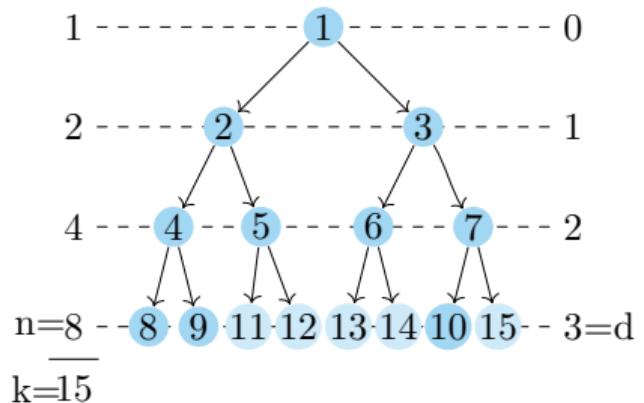
$$d = \log_2(k + 1) - 1 = \log_2(15 + 1) - 1 = 4 - 1 = 3$$

- Herleitung: $2^{d+1} - 1 = k \Leftrightarrow 2^{d+1} = k + 1 \Leftrightarrow \log_2(2^{d+1}) = \log_2(k + 1) \Leftrightarrow d + 1 = \log_2(k + 1) \Leftrightarrow d = \log_2(k + 1) - 1$

- Tiefe mithilfe Anzahl Knoten:

$$d = \lfloor (\log_2(k_{\text{real}})) \rfloor = \lfloor \log_2(10) \rfloor = 3$$

- $d = \lfloor (\log_2(k_{\text{real}})) \rfloor = \lfloor \log_2(8) \rfloor = 3$
- $d = \lfloor (\log_2(k_{\text{real}})) \rfloor = \lfloor \log_2(7) \rfloor = 2$



Tutorat 9

4-Bit-Carry-Ripple-Addierer, Tiefe, n-Bit Inkrementer INC_n , Signextension,
Zweierkomplementzahlen

Gruppe 9

Präsentator:
Jürgen Mattheis
(juergmatth@gmail.com)

Vorlesung von:
Prof. Dr. Scholl

Übungsgruppenbetreuung:
Tobias Seufert

22. August 2023

Gliederung

Aufgabe 1

Aufgabe 2

Aufgabe 3

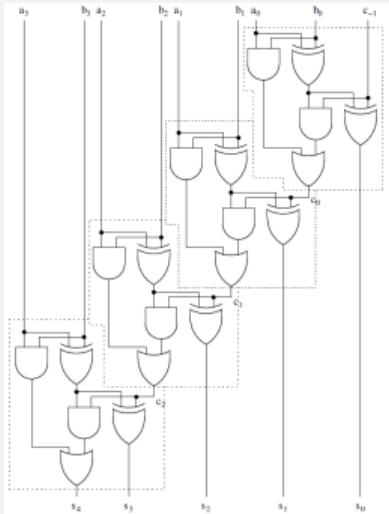
Aufgabe 4

Aufgabe 1

Aufgabe 1 |

4-Bit-Carry-Ripple-Addierer, Tiefe

Lösung 1.1



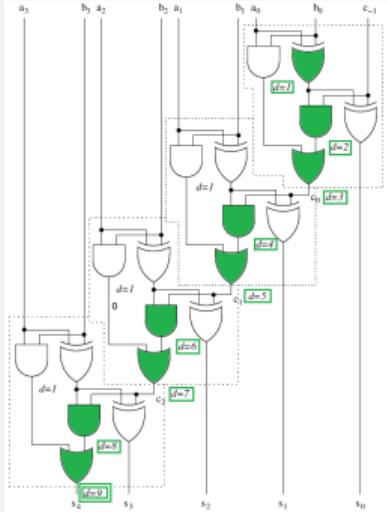
Aufgabe 1 II

4-Bit-Carry-Ripple-Addierer, Tiefe

Lösung 1.2



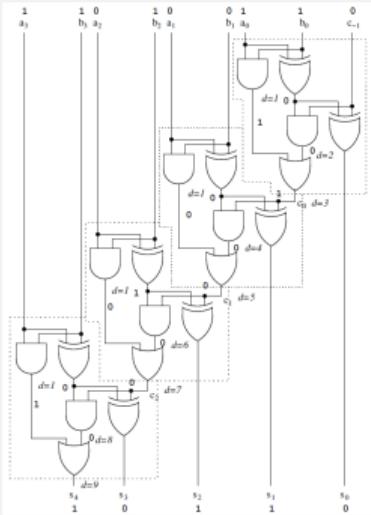
- Allgemein gilt $\text{depth}(CR_n) = 3 + 2(n - 1)$. Hier: $\text{depth}(CR_4) = 3 + 2(4 - 1) = 9$



Aufgabe 1 III

4-Bit-Carry-Ripple-Addierer, Tiefe

Lösung 1.3

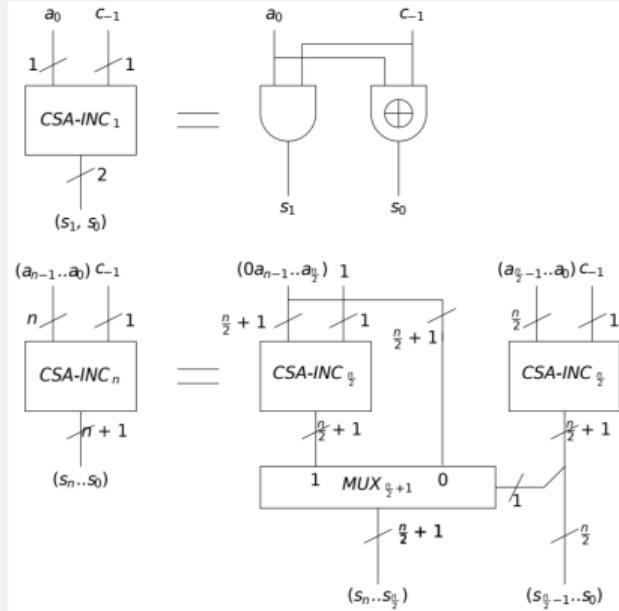


Aufgabe 2

Aufgabe 2

n-Bit Inkrementer INC_n

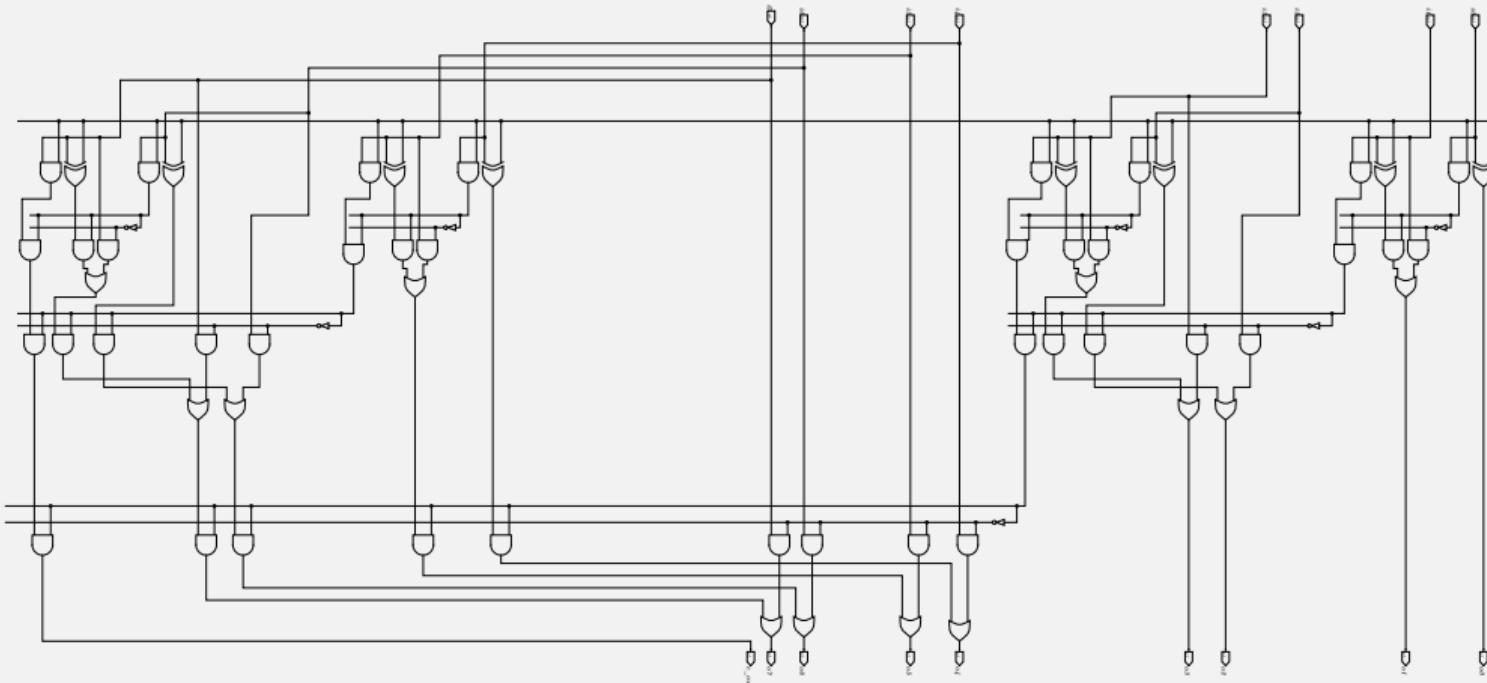
Lösung 2.1



Aufgabe 2

Digital System I

Lösung 2.1



Aufgabe 2

n-Bit Inkrementer INC_n

Lösung 2.2



► Tiefe:

- $k = 0: 1$
- $k = 1: 1 + 3 \cdot 1 = 4$
- $k = 2: 1 + 3 \cdot 2 = 7$
- $k = 3: 1 + 3 \cdot 3 = 10$
- allgemein:

$$\begin{aligned}
 & \text{► } \text{depth}(\text{CSA} - \text{INC}_{\frac{n}{2^k}}) = \text{depth}(\text{CSA} - \text{INC}_1) = \text{depth}(\text{HA}) = 1 \\
 & \text{► } \text{depth}(\text{CSA} - \text{INC}_n) = \text{depth}(\text{CSA} - \text{INC}_{\frac{n}{2^1}}) + \text{depth}(\text{MUX}_{\frac{n}{2}+1}) = \\
 & \quad \text{depth}(\text{CSA} - \text{INC}_{\frac{n}{2^1}}) + 3 = \text{depth}(\text{CSA} - \text{INC}_{\frac{n}{2^2}}) + 3 + 3 = \\
 & \quad \text{depth}(\text{CSA} - \text{INC}_{\frac{n}{2^k}}) + 3 \cdot k = 3 \cdot k + 1 = 3 \cdot \log(n) + 1
 \end{aligned}$$

Aufgabe 2

Digital Communication

Lösung 2.3



► Kosten:

- $k = 0$: 2
- $k = 1$: $2 \cdot 2 + (3 \cdot 1 + 2) \cdot 1 = 9$
- $k = 2$: $2 \cdot 4 + (3 \cdot 1 + 2) \cdot 2 + (3 \cdot 2 + 2) \cdot 1 = 26$
- $k = 3$: $2 \cdot 8 + (3 \cdot 1 + 2) \cdot 4 + (3 \cdot 2 + 2) \cdot 2 + (3 \cdot 4 + 2) \cdot 1 = 66$

► allgemein:

$$\begin{aligned} & \text{cost(CSA - INC}_{\frac{n}{2^k}}\text{)} = \text{cost(CSA - INC}_1\text{)} = \text{cost(HA)} = 2 \\ & \text{cost(CSA - INC}_n\text{)} = 2 \cdot \text{cost(CSA - INC}_{\frac{n}{2^1}}\text{)} + \text{cost(MUX}_{\frac{n}{2^1}+1}\text{)} = 2 \cdot \text{cost(CSA - INC}_{\frac{n}{2^1}}\text{)} + \\ & (3 \cdot 2^{k-1} + 2) \cdot 1 = 2 \cdot 2 \cdot \text{cost(CSA - INC}_{\frac{n}{2^2}}\text{)} + (3 \cdot 2^{k-2} + 2) \cdot 2 + (3 \cdot 2^{k-1} + 2) \cdot 1 = \\ & 2^k \cdot \text{cost(CSA - INC}_{\frac{n}{2^k}}\text{)} + (3 \cdot 2^0 + 2) \cdot 2^{k-1} + (3 \cdot 2^1 + 2) \cdot 2^{k-2} + \dots + (3 \cdot 2^{k-2} + 2) \cdot 2^1 + (3 \cdot 2^{k-1} + \\ & 2) \cdot 2^0 = n \cdot 2 + \sum_{i=0}^{k-1} (3 \cdot 2^{k-i-1} + 2) \cdot 2^i = n \cdot 2 + \sum_{i=0}^{\log(n)-1} (3 \cdot 2^{\log(n)-i-1} + 2) \cdot 2^i = 4n + 3n \cdot \frac{\log(n)}{2} - 2 \end{aligned}$$

Aufgabe 3

Aufgabe 3 |

Signextension

Lösung 3.1



1. Zurückführung auf Sign Extension um 1 Bit:

Lemma: Sei $a \in \mathbb{B}^n$, $a = a_{n-1} \dots a_0$. Dann gilt $[a]_2 = [a_{n-1}a]_2$.

Beweis:

$$[a_{n-1}a]_2 = -a_{n-1} \cdot 2^n + \sum_{i=0}^{n-1} a_i \cdot 2^i = -a_{n-1} \cdot 2^n + \left(a_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} a_i \cdot 2^i \right) = -a_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} a_i \cdot 2^i = [a]_2$$

Damit: $[y]_2 = [y_{n-1}^1 y]_2 = [y_{n-1}^2 y_{n-1}^1 y]_2 = \dots = [y_{n-1}^k \dots y_{n-1}^1 y]_2 = [\text{sext}_k(y)]_2$.

Aufgabe 3 II

Signextension

Lösung 3.1



2. Direkter Beweis: Sei $y \in \mathbb{B}^n$, $y = y_{n-1} \dots y_0$

$$\begin{aligned}
 [\text{sext}_k(y)]_2 &= [\underbrace{y_{n-1} \dots y_{n-1}}_{k-\text{mal}} y]_2 = \sum_{i=0}^{n-2} y_i \cdot 2^i + \sum_{i=n-1}^{n+k-2} y_{n-1} \cdot 2^i - y_{n-1} \cdot 2^{n+k-1} \\
 &= \sum_{i=0}^{n-2} y_i \cdot 2^i + y_{n-1} \cdot \left(\sum_{i=0}^{n+k-2} 2^i - \sum_{i=0}^{n-2} 2^i \right) - y_{n-1} \cdot 2^{n+k-1} \\
 &= \sum_{i=0}^{n-2} y_i \cdot 2^i + y_{n-1} \cdot \left(\frac{2^{n+k-1} - 1}{2 - 1} - \frac{2^{n-1} - 1}{2 - 1} \right) - y_{n-1} \cdot 2^{n+k-1} \\
 &= \sum_{i=0}^{n-2} y_i \cdot 2^i + y_{n-1} \cdot 2^{n+k-1} - y_{n-1} \cdot 2^{n-1} - y_{n-1} \cdot 2^{n+k-1} \\
 &= \sum_{i=0}^{n-2} y_i \cdot 2^i - y_{n-1} \cdot 2^{n-1} \\
 &= [y]_2
 \end{aligned}$$

Aufgabe 4

Aufgabe 4 |

Addition von Zweierkomplementzahlen

Lösung 4.1

$$\begin{array}{r}
 100000 = -32 \\
 + 011111 = 31 \\
 \hline
 (\ddot{U}) \quad 0000000 \\
 \hline
 111111 = -1
 \end{array}$$



Lösung 4.2

$$\begin{array}{r}
 100000 = -32 \\
 + 100000 = -32 \\
 \hline
 (\ddot{U}) \quad \textcolor{red}{1}000000 \\
 \hline
 \textcolor{red}{(1)}000000 = 0
 \end{array}$$



\Rightarrow Überlauf

Aufgabe 4 II

Addition von Zweierkomplementzahlen

Lösung 4.3

$$\begin{array}{r} 010001 \\ + 011011 \\ \hline (\ddot{U}) \quad 0100110 \\ \hline (0)101100 \end{array} = \begin{array}{r} 17 \\ 27 \\ \\ -20 \end{array}$$

\Rightarrow Überlauf



Tutorat 10

Betrag Zweierkomplementzahl, Basiszelle ALU, NOR RS-Flipflop, D-Flip-Flop

Gruppe 9

Präsentator:
Jürgen Mattheis
(juergmatth@gmail.com)

Vorlesung von:
Prof. Dr. Scholl

Übungsgruppenbetreuung:
Tobias Seufert

15. August 2023

Universität Freiburg, Lehrstuhl für Rechnerarchitektur

Gliederung

Aufgabe 1

Aufgabe 2

Aufgabe 3

Aufgabe 4

Appendix

Aufgabe 1

Aufgabe 1 |

Betrag Zweierkomplementzahl

Aufgabe 1.1



Entwickle einen Schaltkreis zu:

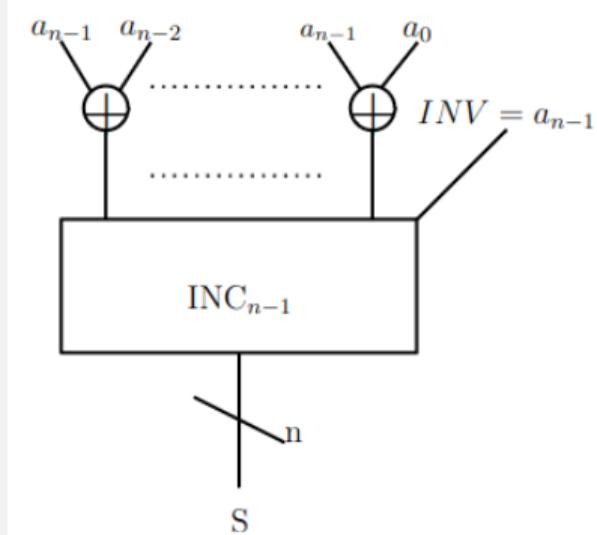
$$abs_n : \mathbb{B}^n \rightarrow \mathbb{B}^n, (a_{n-1}, \dots, a_0) \mapsto (s_{n-1}, \dots, s_0)$$

$$\langle s_{n-1}, \dots, s_0 \rangle = |[a_{n-1}, \dots, a_0]|$$

Aufgabe 1 II

Betrag Zweierkomplementzahl

Lösung 1.1



Aufgabe 1 III

Betrag Zweierkomplementzahl

Lösung 1.1



$$\begin{aligned} \text{cost}(\text{abs}_n) &= \text{cost}(\text{INC}_n - 1) + (n - 1) \cdot \text{cost}(\text{XOR}) = (n - 1) \cdot \text{cost}(\text{HA}) + (n - 1) \\ &= 3(n - 1) \end{aligned}$$

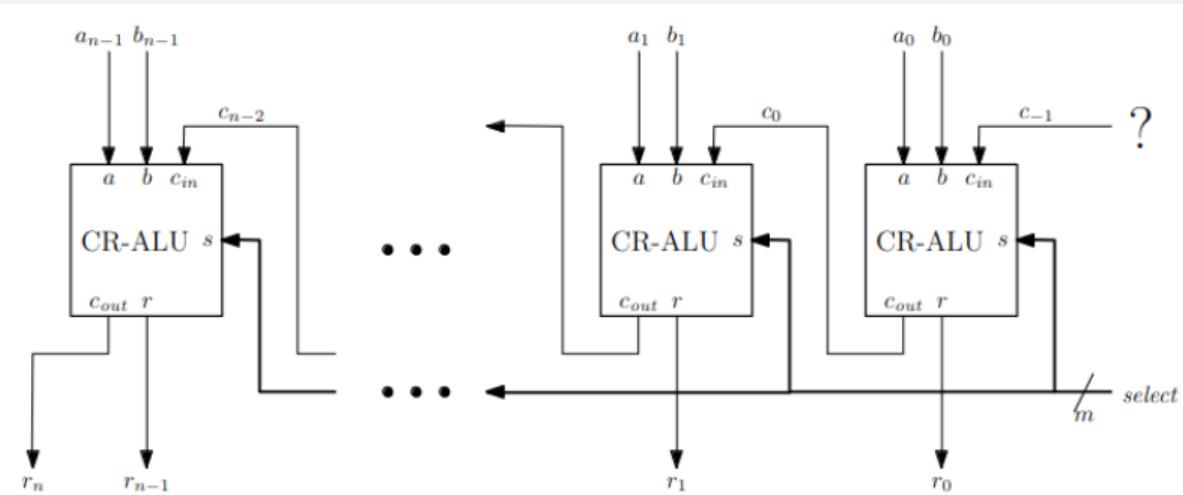
Es gibt **keinen** Überlauf! $|[10\dots 0]| = \langle 10\dots 0 \rangle$

Aufgabe 2

Aufgabe 2 |

Basiszelle Carry-Ripple-ALU

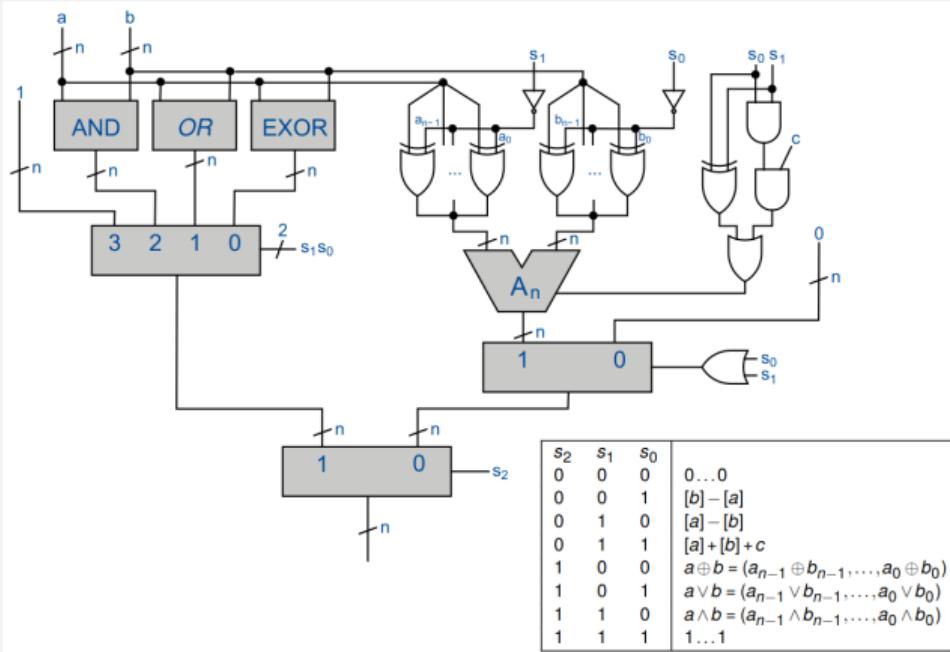
Aufgabe 2.1



Aufgabe 2 II

Positionelle Grundoperationen ALU

Voraussetzungen 2.1



Aufgabe 2 III

Basiszelle Carry-Ripple-ALU

Lösung 2.1

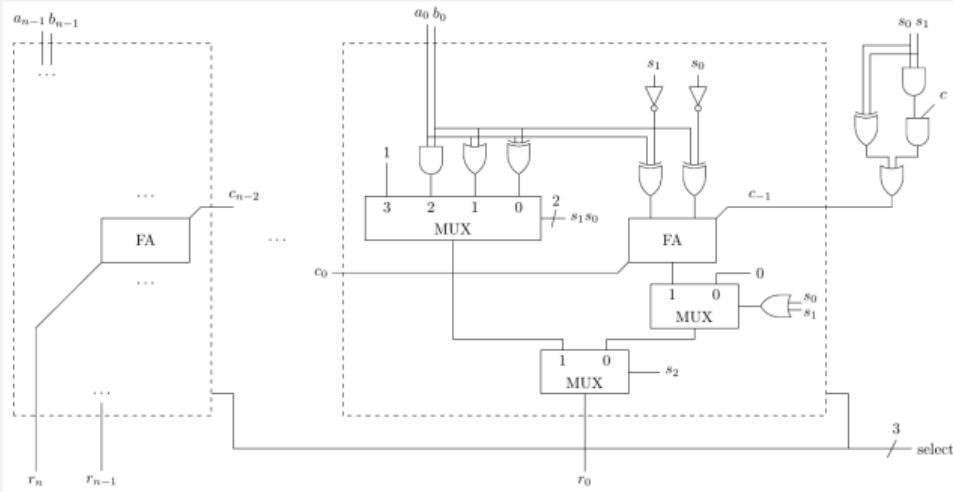
CR-ALU ist wie auf vorheriger Folie bei $n = 1$ mit folgenden Änderungen:

- ▶ A_n ist ein Volladdierer
- ▶ c wird direkt in den Volladdierer übergeben
- ▶ es gibt einen weiteren Ausgang an A_n , um das Carry für die nächste Zelle zu übergeben
- ▶ c_{-1} kann mit der wegfällenden Schaltung zur Carry-Generierung aus der ALU berechnet werden

Aufgabe 2 IV

Basiszelle Carry-Ripple-ALU

Lösung 2.1



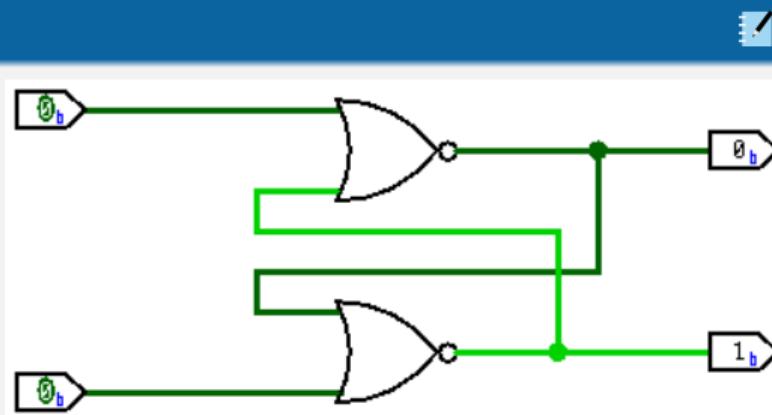
Aufgabe 3

Aufgabe 3

NOR RS-Flipflop

Lösung 3.1

- ▶ Es gibt fünf stabile Belegungen:
 1. $a = 0, b = 0, c = 0, d = 1$
 2. $a = 0, b = 0, c = 1, d = 0$
 3. $a = 0, b = 1, c = 1, d = 0$
 4. $a = 1, b = 0, c = 0, d = 1$
 5. $a = 1, b = 1, c = 0, d = 0$

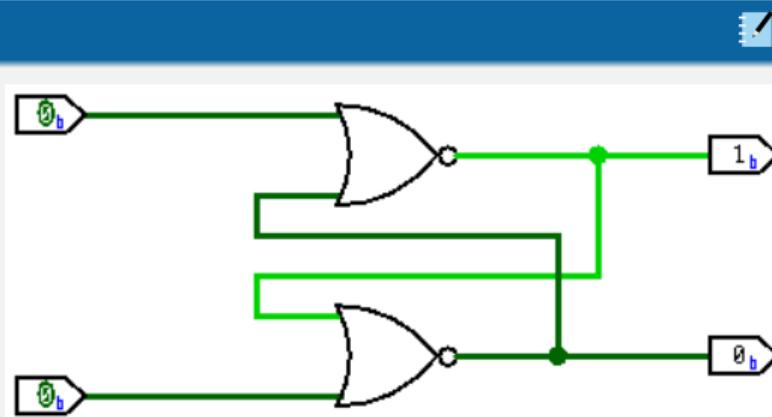


Aufgabe 3

NOR RS-Flipflop

Lösung 3.1

- Es gibt fünf stabile Belegungen:
 1. $a = 0, b = 0, c = 0, d = 1$
 2. $a = 0, b = 0, c = 1, d = 0$
 3. $a = 0, b = 1, c = 1, d = 0$
 4. $a = 1, b = 0, c = 0, d = 1$
 5. $a = 1, b = 1, c = 0, d = 0$

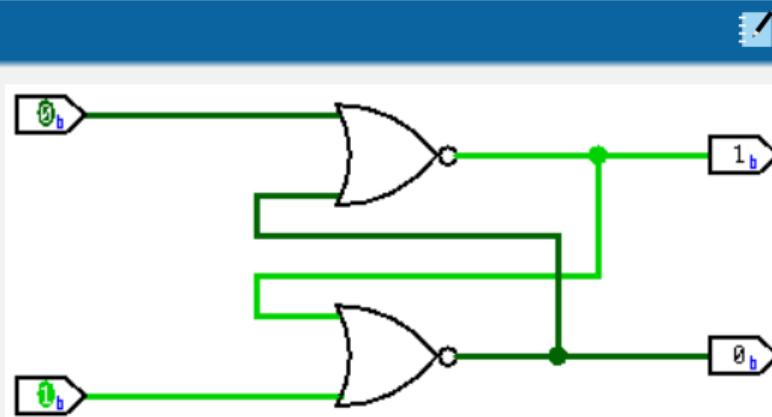


Aufgabe 3

NOR RS-Flipflop

Lösung 3.1

- ▶ Es gibt fünf stabile Belegungen:
 1. $a = 0, b = 0, c = 0, d = 1$
 2. $a = 0, b = 0, c = 1, d = 0$
 3. $a = 0, b = 1, c = 1, d = 0$
 4. $a = 1, b = 0, c = 0, d = 1$
 5. $a = 1, b = 1, c = 0, d = 0$

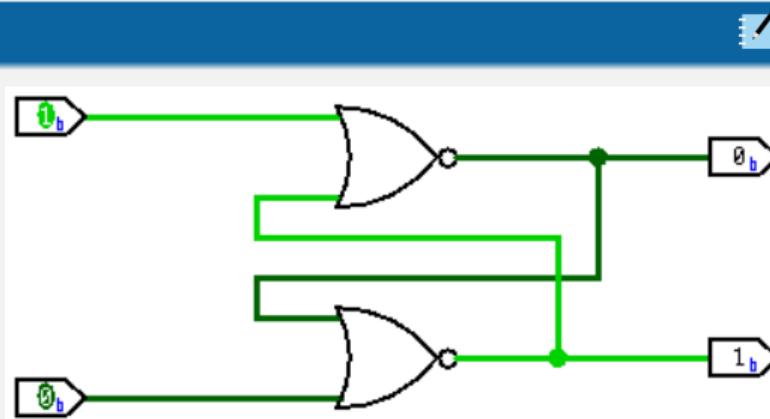


Aufgabe 3

NOR RS-Flipflop

Lösung 3.1

- Es gibt fünf stabile Belegungen:
 1. $a = 0, b = 0, c = 0, d = 1$
 2. $a = 0, b = 0, c = 1, d = 0$
 3. $a = 0, b = 1, c = 1, d = 0$
 4. $a = 1, b = 0, c = 0, d = 1$
 5. $a = 1, b = 1, c = 0, d = 0$

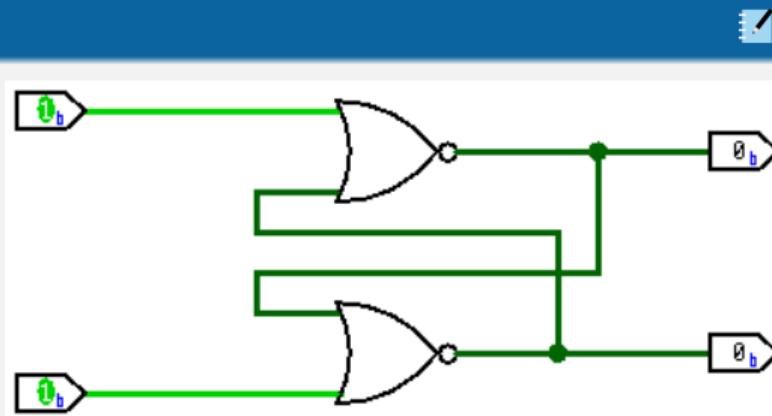


Aufgabe 3

NOR RS-Flipflop

Lösung 3.1

- ▶ Es gibt fünf stabile Belegungen:
 1. $a = 0, b = 0, c = 0, d = 1$
 2. $a = 0, b = 0, c = 1, d = 0$
 3. $a = 0, b = 1, c = 1, d = 0$
 4. $a = 1, b = 0, c = 0, d = 1$
 5. $a = 1, b = 1, c = 0, d = 0$



Aufgabe 3 |

NOR RS-Flipflop

Lösung 3.2



- ▶ bei $a = b = 0$ wird der aktuelle Wert gehalten
- ▶ bei $a = 0, b = 1$ wird c auf 1 und d auf 0 gesetzt
- ▶ bei $a = 1, b = 0$ wird c auf 0 und d auf 1 gesetzt

Lösung 3.3



- ▶ a und b sind active-high, da sie durch das Heben auf 1 aktiviert werden

Lösung 3.4



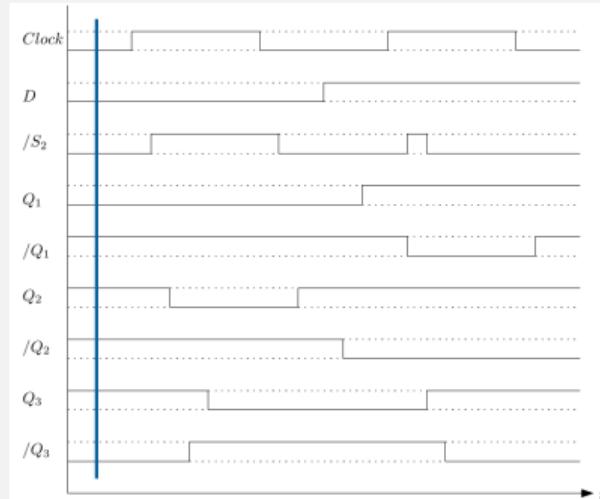
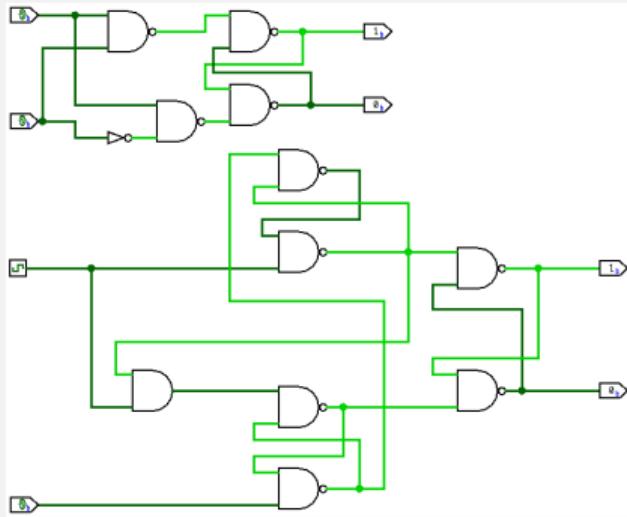
- ▶ die Belegung $a = 1, b = 1$ ergibt keinen Sinn, da
 - ▶ es bei gleichzeitigem Absenken von a und b zu Flackern kommen kann
 - ▶ da es für die diese Eingangsbelegung nur einen stabilen Zustand gibt. Daher kann nur ein Wert „gespeichert“ werden

Aufgabe 4

Aufgabe 4

D-Flip-Flop

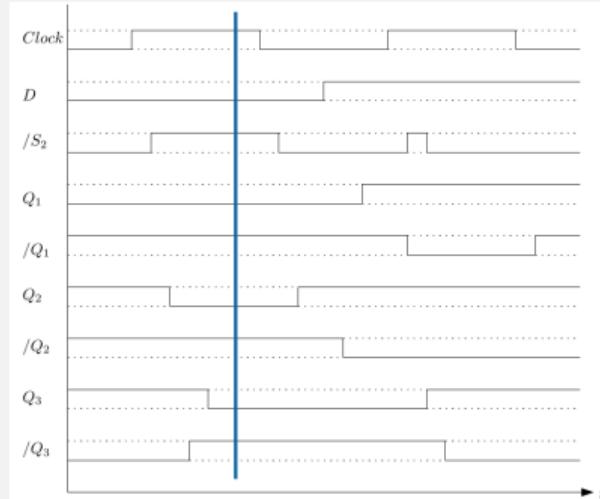
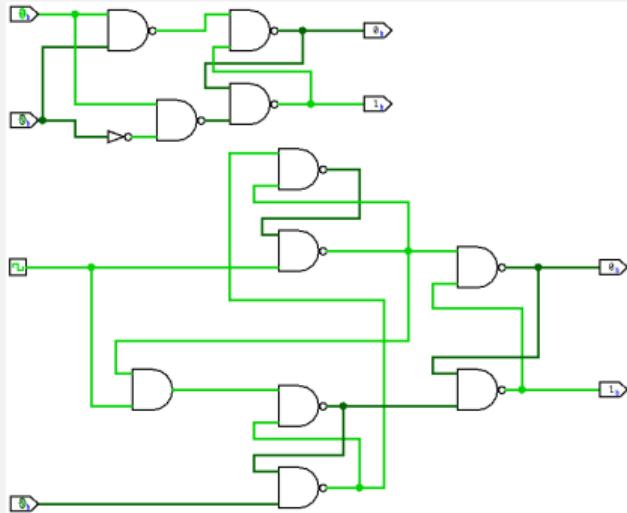
Lösung 4.1



Aufgabe 4

D-Flip-Flop

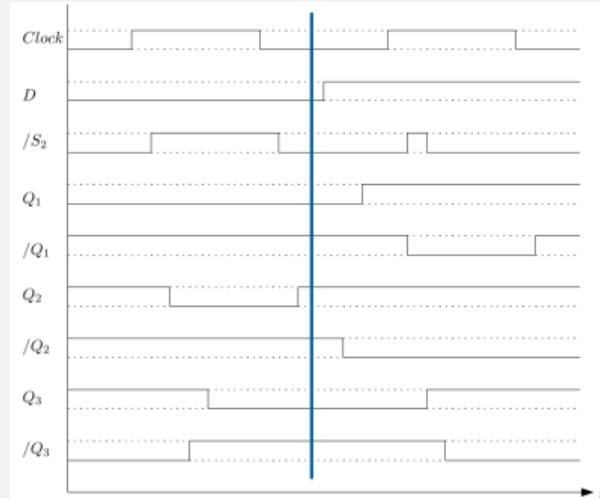
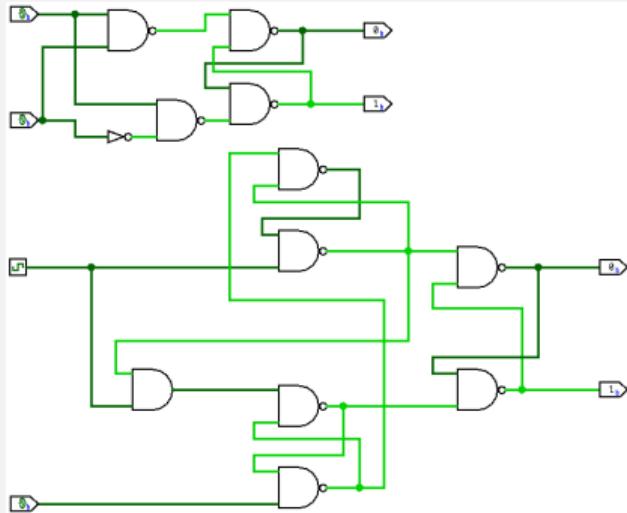
Lösung 4.1



Aufgabe 4

D-Flip-Flop

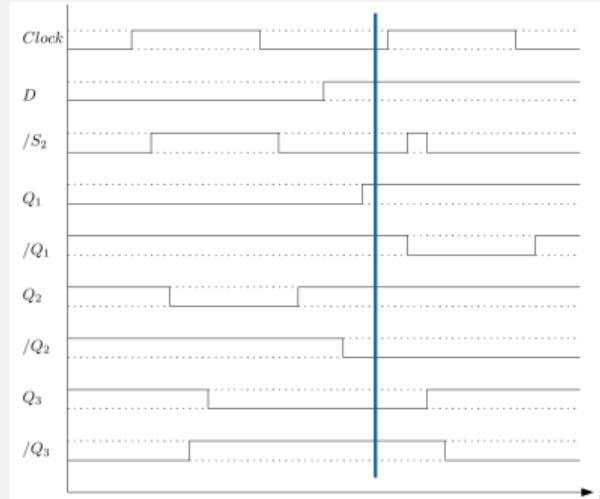
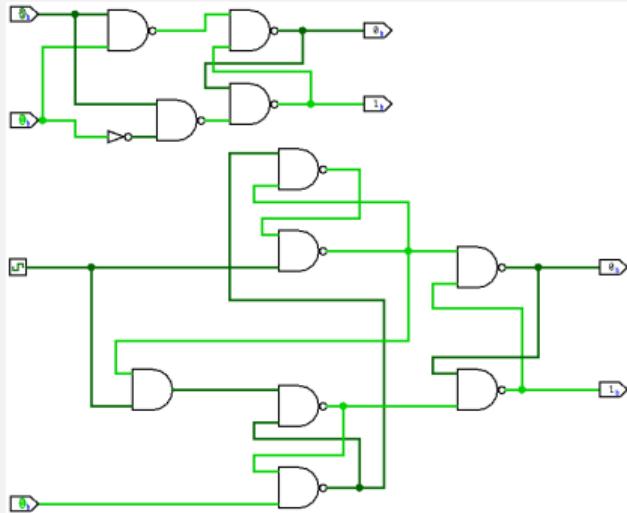
Lösung 4.1



Aufgabe 4

D-Flip-Flop

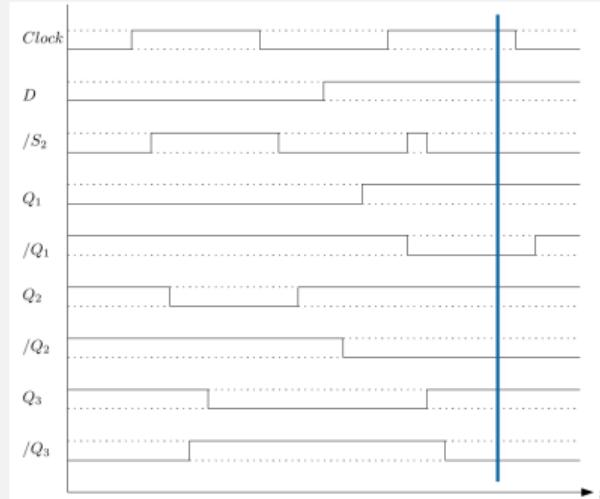
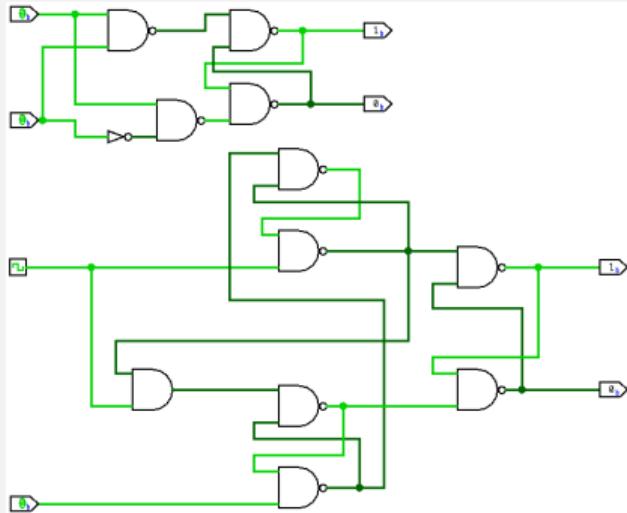
Lösung 4.1



Aufgabe 4

D-Flip-Flop

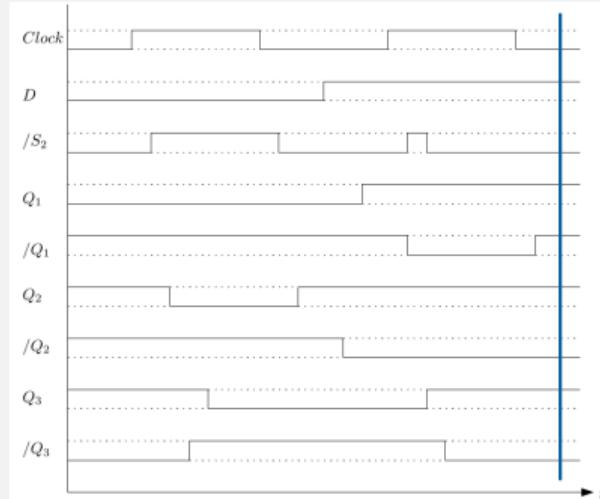
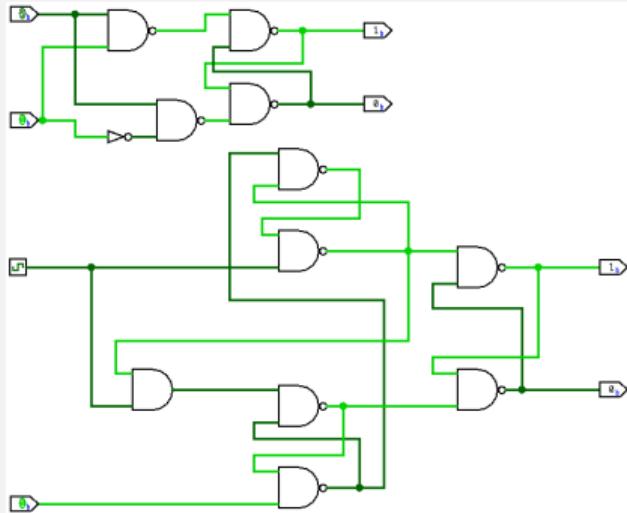
Lösung 4.1



Aufgabe 4

D-Flip-Flop

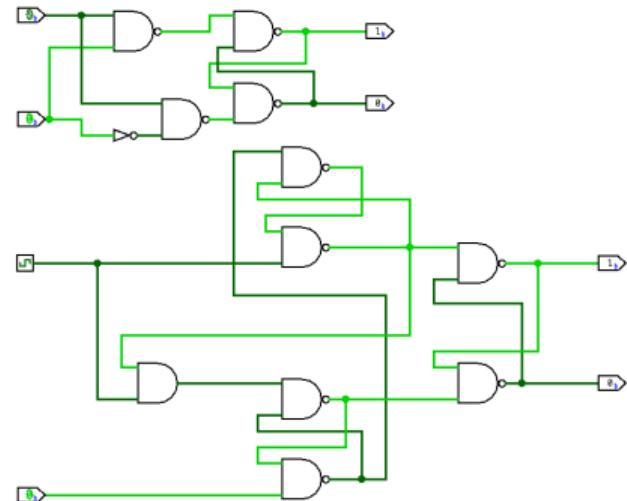
Lösung 4.1



Appendix

Appendix

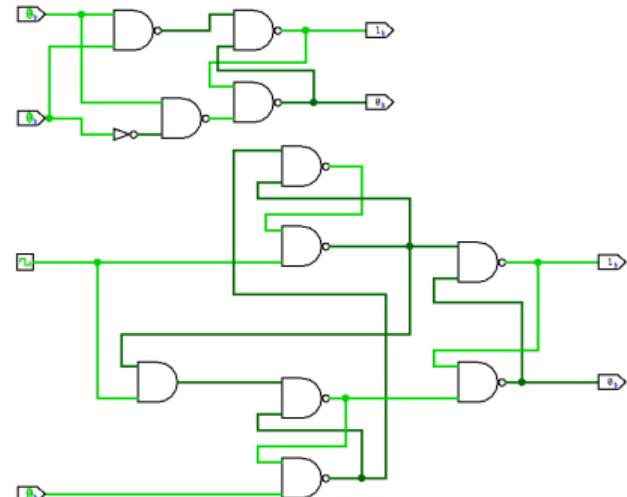
Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop



- ▶ oberer RS-Flip-Flop ist im Metastabilen Zustand und hält dadurch den oberen Eingang des rechten RS-Flip-Flop auf 1
- ▶ unterer RS-Flip-Flop ist im Set Zustand und hält dadurch den unteren Eingang des rechten RS-Flip-Flop auf 1
- ⇒ rechter RS-Flip-Flop ist dadurch im Wert-Speichern Zustand

Appendix

Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop

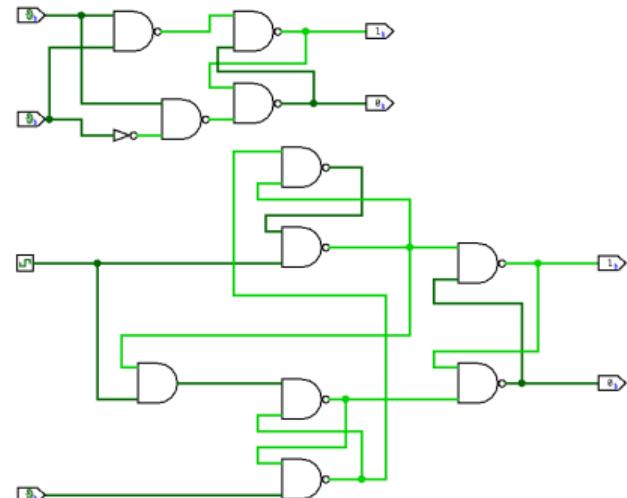


► bei $d = 1$

- ▶ oberer RS-Flip-Flop geht bei *clock* auf 1 in den Set Zustand über und hält dadurch den oberen Eingang des rechten RS-Flip-Flop auf 0
- ▶ unterer RS-Flip-Flop geht bei *clock* auf 1 in den Set Zustand über und hält dadurch den unteren Eingang des rechten RS-Flip-Flop auf 1
- ⇒ rechter RS-Flip-Flop ist dadurch im Set Zustand

Appendix

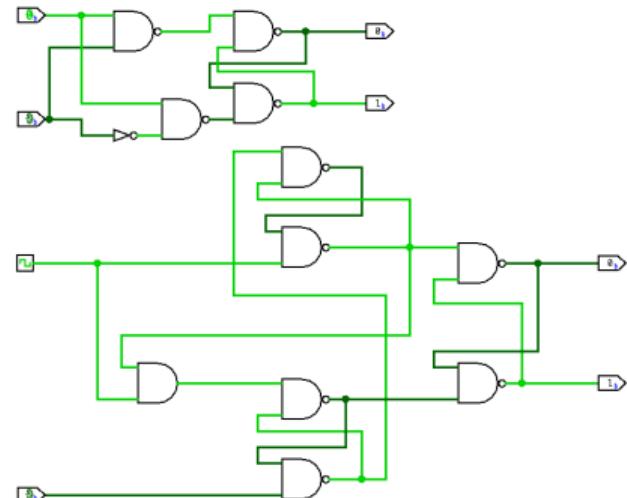
Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop



- ▶ oberer RS-Flip-Flop ist im Reset Zustand und hält dadurch den oberen Eingang des rechten RS-Flip-Flop auf 1
- ▶ unterer RS-Flip-Flop ist im Metastabilen Zustand und hält dadurch den unteren Eingang des rechten RS-Flip-Flop auf 1
- ⇒ rechter RS-Flip-Flop ist dadurch im Wert-Speichern Zustand

Appendix

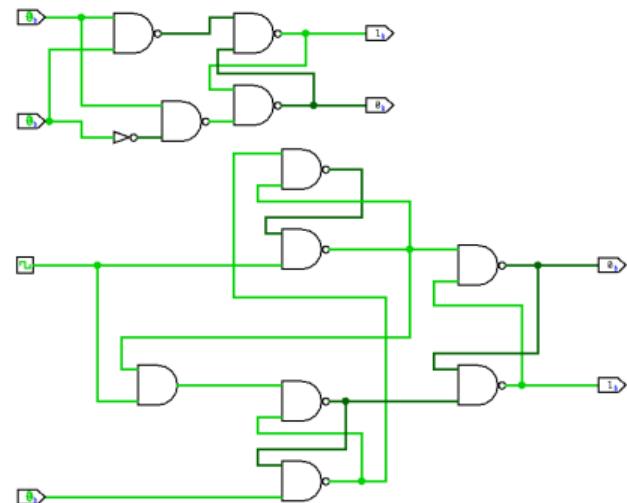
Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop



- ▶ bei $d = 0$
- ▶ oberer RS-Flip-Flop geht bei **clock** auf 1 in den **Wert-Speichern Zustand** über und hält dadurch den oberen Eingang des rechten RS-Flip-Flop auf 1. Der obere RS-Flip-Flop muss vorher in am unteren Eingang den Wert 1 gehalten haben, da der obere RS-Flip-Flop wenn die Clock auf 0 ist dafür sorgen muss, dass der rechte RS-Flip-Flop im **Wert-Speichern Zustand** ist, indem er eine 1 hält
- ▶ unterer RS-Flip-Flop geht bei **clock** auf 1 in den **Reset Zustand** über und hält dadurch den unteren Eingang des rechten RS-Flip-Flop auf 0
- ⇒ rechter RS-Flip-Flop ist dadurch im **Reset Zustand**

Appendix

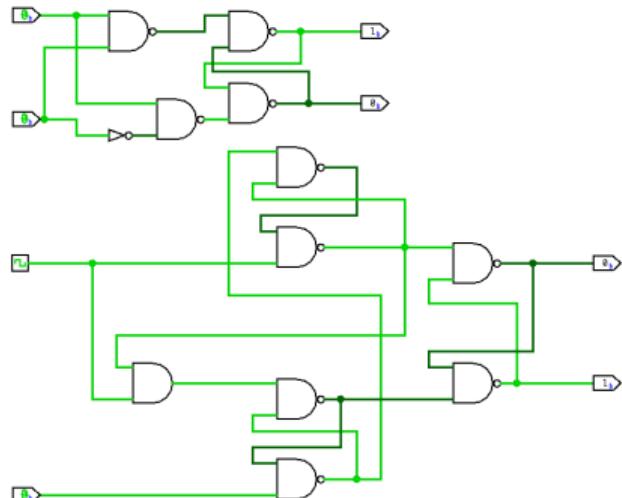
Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop



- ▶ der **D-Flip-Flop** verhält sich gleich wie ein **D-Latch** (siehe Aufgabe 4) allerdings mit dem **Unterschied**, dass beim D-Flip-Flop der zu speichernde Wert in D beim Anheben der Clock von 0 auf 1 feststehen muss und im Nachhinein **nicht geändert** werden kann solange die clock 1 ist und auch nicht während die Clock 0 ist
 - ▶ das wird erreicht indem die Werte an den beiden Eingängen des rechten RS-Flip-Flop durch den oberen und unteren RS-Flip-Flop gehalten werden wenn die Clock 1 ist
 - ▶ und indem der Wert 1 an einem der beiden Eingänge des rechten RS-Flip-Flop durch den oberen oder unteren RS-Flip-Flop passend gehalten wird wenn die Clock 0 ist

Appendix

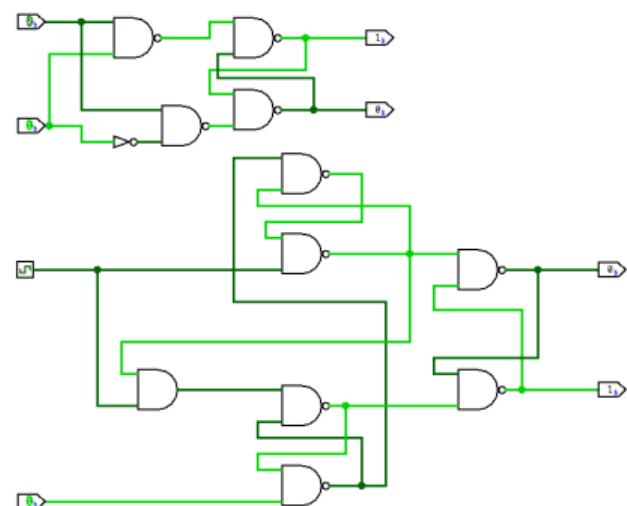
Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop



- ▶ der **D-Flip-Flop** verhält sich gleich wie ein **D-Latch** (siehe Aufgabe 4) allerdings mit dem **Unterschied**, dass beim D-Flip-Flop der zu speichernde Wert in D beim Anheben der Clock von 0 auf 1 feststehen muss und im Nachhinein **nicht geändert** werden kann solange die clock 1 ist und auch nicht während die Clock 0 ist
 - ▶ das Ändern von **D** ändert die Zustände des oberen und unteren RS-Flip-Flop zwar, aber niemals so, dass sich dadurch die an den Eingängen des rechten RS-Flip-Flop gehaltenen Werte ändern
 - ▶ das Ändern von der **Clock** kann nur einen der beiden Eingänge des rechten RS-Flip-Flop zwischen 0 und 1 wechseln lassen

Appendix

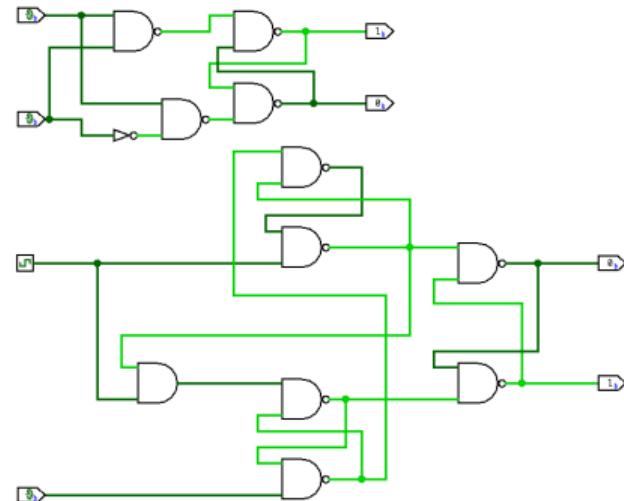
Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop



- ▶ beim D-Latch kann der gespeicherte Wert beliebig durch Ändern des Wertes an D geändert werden solange die Clock 1 ist
 - ▶ nur während die Clock 0 ist wird der Wert gehalten und kann nicht geändert werden
 - ▶ damit der D-Latch sinnvoll verwendet werden kann müsste der Wert von D die ganze Zeit über gehalten werden, während die Clock 1 ist + Setup-Zeit und Hold-Zeit, da jede Änderung während die Clock 1 ist den gespeicherten Wert ändern würde
- ▶ beim D-Flip-Flop wird der gespeicherte Wert sowohl während die Clock 1 oder 0 ist gehalten und kann nur geändert werden, wenn die Clock von 0 auf high wechselt
 - ▶ beim Wechsel der Clock auf 0 geht der rechte RS-Flip-Flop in den Zustand Wert-Speichern über

Appendix

Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop



- ▶ D sorgt dafür, dass der obere und untere RS-Flip-Flop jeweils im „richtigen“ Zustand sind, so dass ein Flankenwechsel der Clock auf 1 den rechten RS-Flip-Flop richtig einstellt

Tutorat 11

Reduktion von Mealy Automaten, Schaltwerke, Zustandsdiagramme,
Mealy-Automaten, Implementierung eines Tristate Treibers, Transistoren

Gruppe 9

Präsentator:
Jürgen Mattheis
(juergmatth@gmail.com)

Vorlesung von:
Prof. Dr. Scholl

Übungsgruppenbetreuung:
Tobias Seufert

24. August 2023

Gliederung

Aufgabe 2

Aufgabe 3

Aufgabe 4

Appendix

Aufgabe 2

Aufgabe 2

Reduktion von Mealy Automaten, Schaltwerke

Aufgabe 2.1



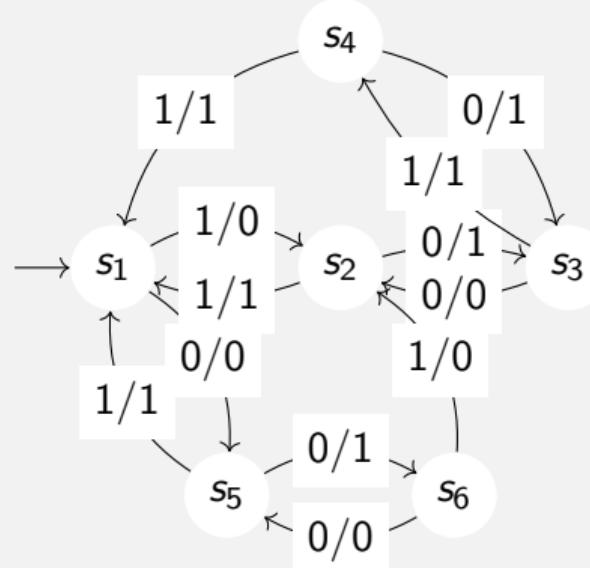
- ▶ Mealy-Automat gegeben durch: $M = (\{0, 1\}, \{0, 1\}, \{s_1, \dots, s_6\}, \{s_1\}, \delta, \lambda)$

| s^t | x | s^{t+1} | y |
|-------|-----|-----------|-----|
| s_1 | 0 | s_5 | 0 |
| s_1 | 1 | s_2 | 0 |
| s_2 | 0 | s_3 | 1 |
| s_2 | 1 | s_1 | 1 |
| s_3 | 0 | s_2 | 0 |
| s_3 | 1 | s_4 | 1 |
| s_4 | 0 | s_3 | 1 |
| s_4 | 1 | s_1 | 1 |
| s_5 | 0 | s_6 | 1 |
| s_5 | 1 | s_1 | 1 |
| s_6 | 0 | s_5 | 0 |
| s_6 | 1 | s_2 | 0 |

Aufgabe 2

Reduktion von Mealy Automaten, Schaltwerke

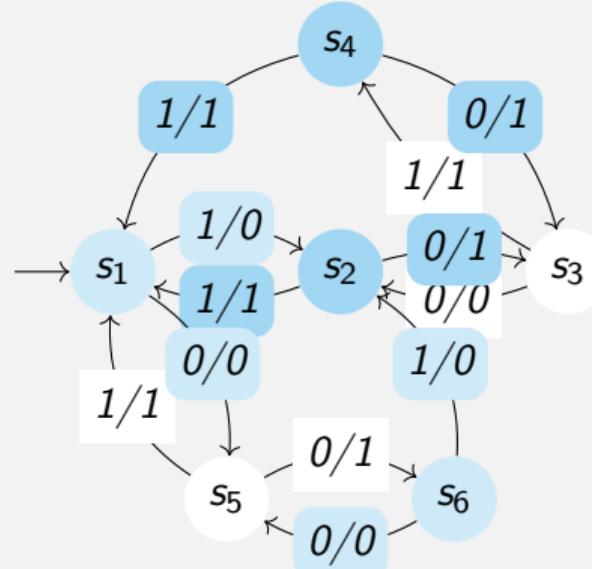
Aufgabe 2.1



Aufgabe 2

Reduktion von Mealy Automaten, Schaltwerke

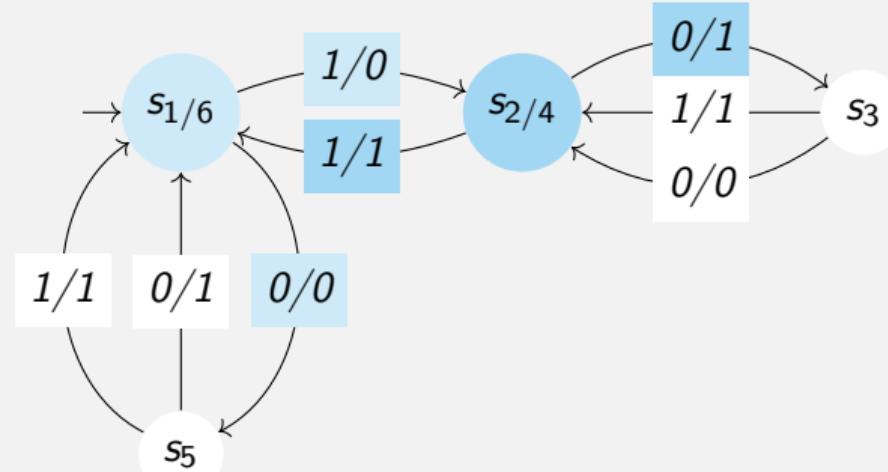
Lösung 2.1



Aufgabe 2

Reduktion von Mealy Automaten, Schaltwerke

Lösung 2.1



Aufgabe 2

Reduktion von Mealy Automaten, Schaltwerke

Lösung 2.2



- $s_i = (z_1, z_0), s_{1,6} = (0, 0), s_{2,4} = (0, 1), s_3 = (1, 0), s_5 = (1, 1)$

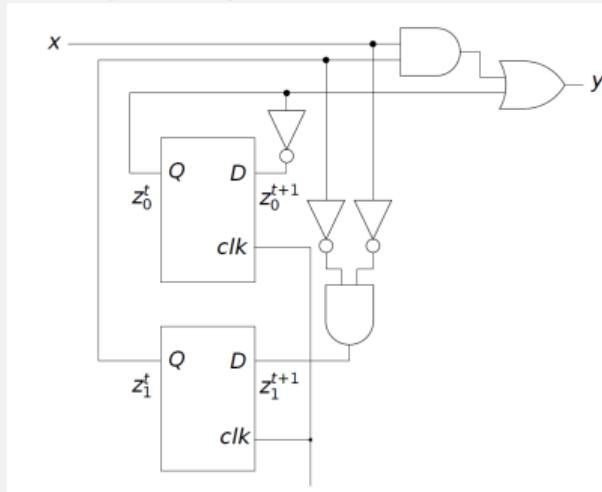
| x | state | | next state | | y |
|-----|---------|---------|-------------|-------------|-----|
| | z_1^t | z_0^t | z_1^{t+1} | z_0^{t+1} | |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |

Aufgabe 2

Reduktion von Mealy Automaten Schaltwerke

Lösung 2.3

- ▶ $z_0^{t+1} = \bar{x}\overline{z_1^t z_0^t} + x\overline{z_1^t z_0^t} + \bar{x}z_1^t \overline{z_0^t} + xz_1^t \overline{z_0^t}$
- ▶ $z_1^{t+1} = \bar{x}z_1^t z_0^t + \bar{x}z_1^t z_0^t$
- ▶ $y = \bar{x}z_1^t z_0^t + xz_1^t z_0^t + xz_1^t \overline{z_0^t} + \bar{x}z_1^t z_0^t + xz_1^t z_0^t$

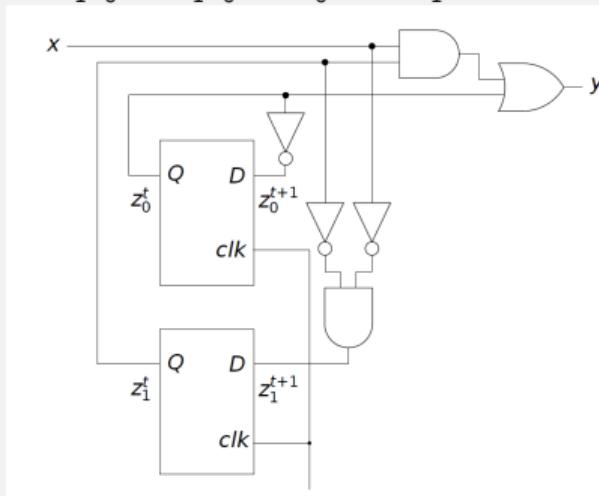


Aufgabe 2

Reduktion von Mealy-Automaten, Schaltwerke

Lösung 2.4

- ▶ $z_0^{t+1} = \bar{x}\bar{z}_1^t z_0^t + x\bar{z}_1^t z_0^t + \bar{x}z_1^t \bar{z}_0^t + xz_1^t \bar{z}_0^t \stackrel{\text{Red.}}{=} \bar{z}_0^t$
- ▶ $z_1^{t+1} = \bar{x}\bar{z}_1^t z_0^t + \bar{x}z_1^t z_0^t \stackrel{\text{Red.}}{=} \bar{x} + z_1^t = \bar{x} \cdot \bar{z}_1^t$
- ▶ $y = \bar{x}\bar{z}_1^t z_0^t + x\bar{z}_1^t z_0^t + xz_1^t \bar{z}_0^t + \bar{x}z_1^t z_0^t + xz_1^t z_0^t \stackrel{\text{Red.}}{=} z_0^t + x \cdot z_1^t$

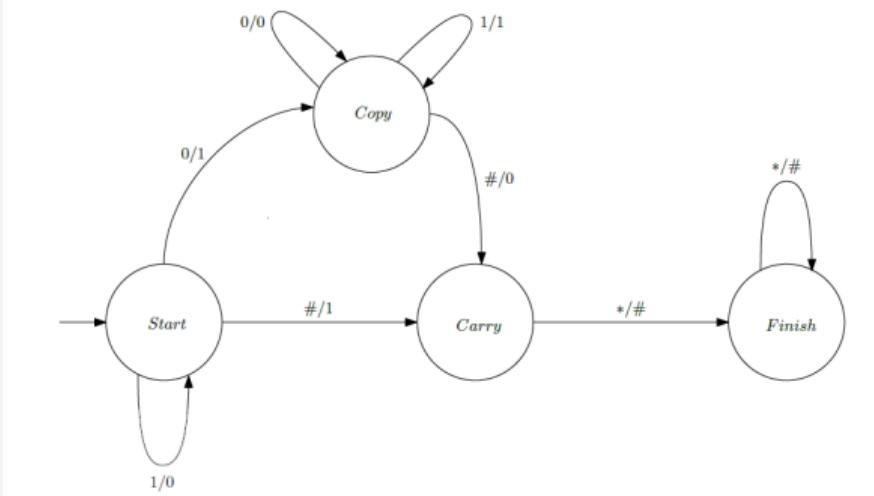


Aufgabe 3

Aufgabe 3

Zustandsdiagramme, Mealy-Automaten

Lösung 3.1



Aufgabe 3 |

Zustandsdiagramm, Mealy-Automaten

Lösung 3.1



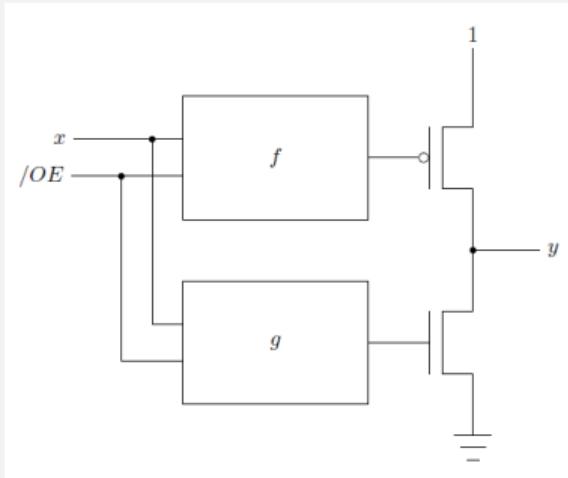
- ▶ Zum Inkrementieren müssen vom niederwertigsten Bit an alle 1en in 0en verwandelt werden, bis man entweder zum Wortende oder zur ersten Null gelangt.
- ▶ Im Falle einer Null, wird diese in eine Eins verwandelt und man geht in den Zustand Copy. Hier wird bis zum Wortende alles kopiert.
- ▶ Liest man das erste #, so wird das Übertragsbit ausgegeben
 - ▶ Hat man schon 0en gefunden (Copy), ist das Übertragsbit 0, ansonsten 1
- ▶ Dann ist man am Wortende angelangt (d. h. man liest zum zweiten Mal #), wird an die Zahl vorne ein # angehängt und man geht zum Endzustand
- ▶ Wegen Vollständigkeit muss jeder Zustand je eine Kante für jeden Eingabebuchstaben besitzen. * kennzeichnet eine Kante, die für alle möglichen Eingabebuchstaben gilt

Aufgabe 4

Aufgabe 4 |

Implementierung eines Tristate Treibers, Transistoren

Aufgabe 4.1



Aufgabe 4 II

Implementierung eines Tristate Treibers, Transistoren

Lösung 4.1



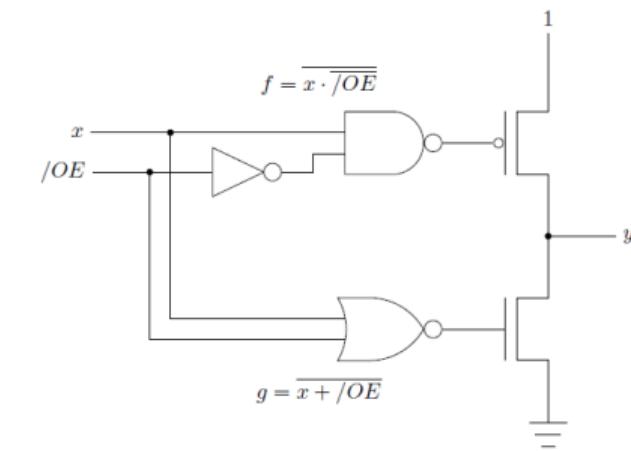
| x | $/OE$ | f | g | Erklärung |
|-----|-------|-----|-----|---|
| 0 | 0 | 1 | 1 | NMOS schaltet durch, zieht y gegen 0. |
| 0 | 1 | 1 | 0 | weder P- noch NMOS dürfen leiten. |
| 1 | 0 | 0 | 0 | PMOS schaltet durch, zieht y gegen 1. |
| 1 | 1 | 1 | 0 | weder P- noch NMOS dürfen leiten. |

$$\begin{aligned} \blacktriangleright f &= \bar{x} \cdot \overline{/OE} + \bar{x} \cdot /OE + x \cdot /OE = \bar{x} + /OE \\ &= \overline{x \cdot /OE} \end{aligned} \quad \begin{aligned} \blacktriangleright g &= \bar{x} \cdot \overline{/OE} = \overline{x + /OE} \end{aligned}$$

Aufgabe 4 |

Implementierung eines Tristate Treibers, Transistoren

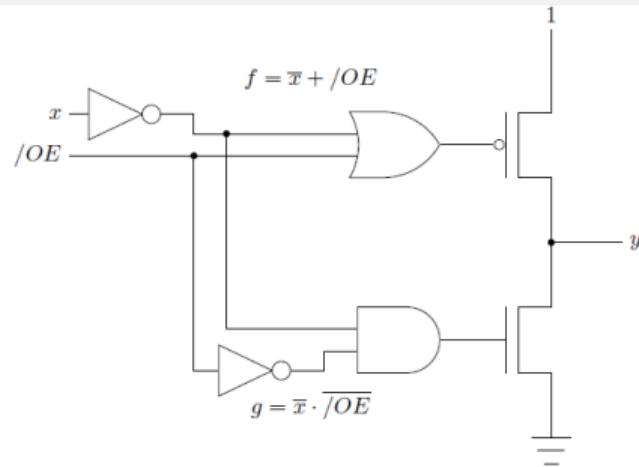
Lösung 4.2



Aufgabe 4 II

Implementierung eines Tristate Treibers, Transistoren

Lösung 4.2



Appendix

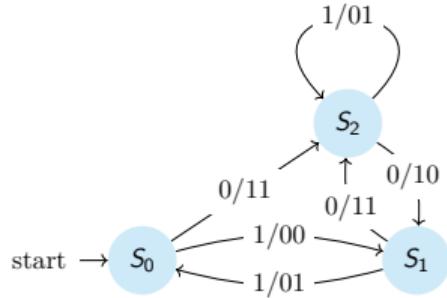
Appendix

Reduktion von Zustandsdiagrammen

- ▶ **Hinreichende Bedingung:** Wenn bei zwei Zuständen bei gleicher Eingabe auch die gleiche Ausgabe erzeugt wird und der gleiche Folgezustand angenommen wird, dann sind die Zustände äquivalent

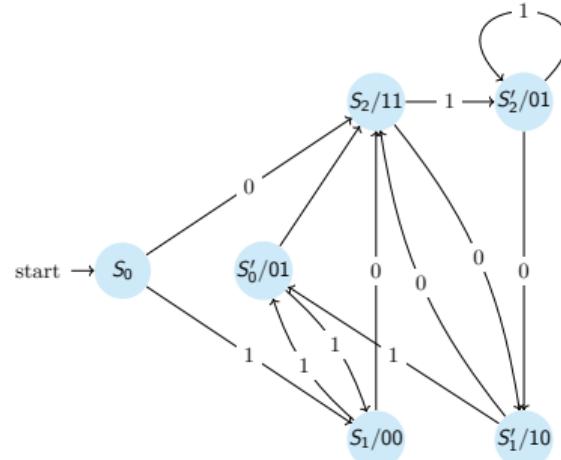
Appendix

Mealy-Automaten zu Moore-Automaten umwandeln und vice versa



► Mealy-Automat:

- ▶ Übergangsfunktion $\delta : S \times I \rightarrow S$
- ▶ Ausgabefunktion $\lambda : S \times I \rightarrow O$



► Moore-Automat:

- ▶ Übergangsfunktion $\delta : S \times I \rightarrow S$
- ▶ Ausgabefunktion $\lambda : S \rightarrow O$

Appendix

Gruppen von endlichen Automaten

- ▶ Akzeptoren
- ▶ Transduktoren, darunter fallen:
 - ▶ Moore-Automaten
 - ▶ Mealy-Automaten

Appendix

Tricks beim Design von endlichen Automaten

- ▶ Äquivalenzrelation Nerode-Rechtskongruenz: $x \sim_L y \iff (\forall z \in \Sigma^* : xz \in L \iff yz \in L)$

Technische Informatik

Musterlösung zu Übungsblatt 12 (Bonus)

Hinweis:

- Alle Aufgaben auf diesem Blatt sind Bonusaufgaben. Diese Aufgaben zählen nicht in die Gesamtheit der Aufgaben, bei sinnvoller Bearbeitung werden sie jedoch zur Menge der sinnvoll bearbeiteten Aufgaben gerechnet.
- Die Verzögerungszeiten für alle relevanten Komponenten finden Sie bei den Vorlesungsmaterialien unter "Zusatzmaterial".

Aufgabe 1 (3 Punkte)

Betrachten Sie die Befehlstabelle (zu finden bei den Vorlesungsmaterialien unter "Zusatzmaterial") und die Datenpfade der ReTI (s. Abb. 1).

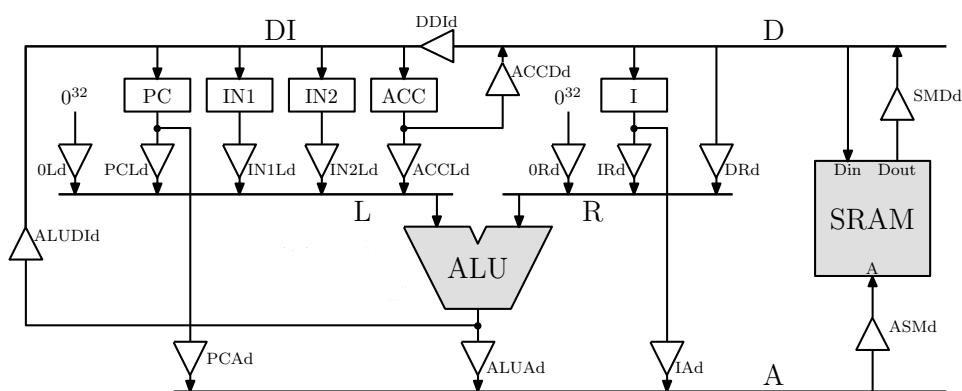


Abbildung 1: Datenpfade von ReTI

Bei welchen Befehlen werden in der Execute-Phase die folgenden Treiber enabled?

- a) /IAdoe

- b) /IN1Ldoe
c) /ALUAdoe

Lösung:

Hinweis: je 0.5P

In der Execute-Phase sind bei den folgenden Befehlen die unten genannten Treiber enabled:

- a) /IAdoe: Compute Memory, LOAD i, STORE i,
b) /IN1Ldoe: LOADIN1 i, STOREIN1 i, MOVE S D mit S=IN1, Compute Immediate und Memory mit D=IN1
c) /ALUAdoe: LOADIN1 i, LOADIN2 i, STOREIN1 i, STOREIN2 i

Aufgabe 2 (1 + 1 + 1 + 1 + 1 Punkte)

Prüfen Sie, ob folgende Befehle bzw. Befehlsklassen mit den vorgestellten Datenpfaden der ReTI und der vorgestellten groben zeitlichen Planung durch idealisierte Timing-Diagramme realisierbar sind. Vernachlässigen Sie dabei eventuelle Probleme mit der Kodierung der Befehle und der Unterbringung neben den bereits definierten Befehlen.

Geben Sie für die realisierbaren Befehle an, welche Treiber in der Execute-Phase aktiviert werden. Geben Sie für die nicht oder nur teilweise realisierbaren Befehle an, unter welchen Einschränkungen hinsichtlich S, S_1, S_2 bzw. D sie realisierbar wären. Ist ein Befehl durch Hinzufügen von zusätzlichen Treibern realisierbar, geben Sie an, welche Treiber hinzugefügt werden müssten, um die Befehle komplett realisierbar zu machen.

Geben Sie auch an, welche Clock-Enable-Signale der Register aktiviert werden.

Es gelte $S, S_1, S_2, D \in \{ACC, IN1, IN2, PC\}$.

Hinweis: Für das $cken$ Signal eines Registers $S, S_1, S_2, D \in \{ACC, IN1, IN2, PC\}$ darf abkürzend $Scken, S_1cken, S_2cken$ bzw. $Dcken$ geschrieben werden.

| | Befehl | Wirkung |
|----|--|--|
| a) | LOADREL $D i$ | $D := M(\langle PC \rangle + [i])$ |
| b) | STOREX $S i$ | $M(\langle i \rangle) := S$ |
| c) | SUBXI $S D i$ ADDXI $S D i$ OPLUSXI $S D i$ ORXI $S D i$ ANDXI $S D i$ | $[D] := [S] - [i]$ $[D] := [S] + [i]$ $D := S \oplus 0^8i$ $D := S \vee 0^8i$ $D := S \wedge 0^8i$ |
| d) | SUBXR $S_1 S_2 D$... | $[D] := [S_1] - [S_2]$... |
| e) | SUBXMEM $S i$... | $[M(\langle i \rangle)] := [M(\langle i \rangle)] - [S]$... |

jeweils zusätzlich $\langle PC \rangle := \langle PC \rangle + 1$, wenn $D \neq PC$.

Lösung:

- a) LOADREL $D i$: Realisierbar, PCLd, IRd, ALUAd, ASMd, SMDd, DDId, Dcken
- b) STOREX $S i$: Nur für $S = ACC$ realisierbar, wäre komplett realisierbar, wenn PCDD, IN1Dd und IN2Dd existieren würden. Dann: IAd, ASMd und SDd. Alternativ: Ein Treiber DIDd einfügen, dann aktiv: IAd, SLd, 0Rd, ALUDId, DIDd. (Man legt den Inhalt von S an die ALU an und addiert mit der konstanten 0. Das Ergebnis schiebt man dann über den neuen Treiber DIDd an den Speicher)
- c) SUBXI $S D i, \dots$: Realisierbar, SLd, IRd, ALUDId und Clocksignal Dcken.
- d) SUBXR $S_1 S_2 D, \dots$: Nur für $S_1 = ACC$ oder $S_2 = ACC$ realisierbar:
 - Für SUBXR, da die Subtraktion in beide Richtungen wählbar ist (vgl. Kapitel 3.6, Folie 7)
 - Für die Befehle ADDXR, OPLUSXR, ORXR und ANDXR unter Ausnutzung der Kommutativität.
- e) SUBXMEM $S i, \dots$: Überhaupt nicht machbar, da doppelte Belegung von D notwendig.

Punkte: Realisierbar, teilweise realisierbar oder nicht realisierbar je [0,5]; richtige Treiber [0,5] (eventuelle Folgefehler: wenn jemand IMMER einen bestimmten Treiber vergisst, reicht auch insgesamt [-0,5])

Aufgabe 3 (2 + 2 + 2 Punkte)

Für die ReTI benötigen wir eine Reihe von Kontrollsignalen, die die Register, Treiber, ALU und SRAM ansprechen (Siehe Kap. 4.5).

Erstellen Sie beispielhaft für die folgenden Kontrollsignale des ReTI-Rechners einen dazugehörigen Booleschen Ausdruck für die Kontrollogik.

- a) $/ACCDdoe$ (Verbindung ACC mit Datenbus)

Hinweis: Erstellen Sie hier einen Booleschen Ausdruck für $ACCDdoe_{pre}$, ähnlich wie in der Vorlesung z.B. für $PCLdoe_{pre}$ vorgeführt. $/ACCDdoe$ ist dann der negierte Ausgang des D-FFs mit Eingang $ACCDdoe_{pre}$.

- b) $f[2:0]$ (Funktionsauswahl der ALU. Kodierung wie in Kap. 3.6)

- c) $sext$ (Sign Extension)

Bestimmen Sie zunächst zu welchen Zeitpunkten und dann unter welchen Bedingungen (Befehlsart) die Signale aktiv sein müssen. Beachten Sie dabei, dass einige Signale active-low sind.

Erläutern Sie Ihre Vorgehensweise.

Hinweis: Für die Signale $f[2:0]$ und $sext$ benötigen Sie keine Kodierung des Taktes.

Lösung:

- a) $/ACCDdoe$

- 1) $/ACCDdoe$ muss aktiv sein bei STORE, STOREIN1, STOREIN2 und MOVE, wenn $S = ACC$. Es soll aktiv werden im 1. Takt von Execute (siehe Abb. 2). Das Signal muss schon ein Takt vorher aktiv sein (also bei $E = 1$, $s_1 = 0$, $s_0 = 0$). Zudem muss es insgesamt 3 Takte aktiv sein.
Damit:

$$\begin{aligned} ACCDdoe_{pre} := & \quad E \cdot (\bar{s}_1 \bar{s}_0 + \bar{s}_1 s_0 + s_1 \bar{s}_0) &&; \text{Startzeit bei } P_0, P_1, P_2 \text{ von Execute} \\ & \cdot (\bar{I}_{31} \bar{I}_{30}) &&; \text{STORE und MOVE} \\ & \cdot (\bar{I}_{29} \bar{I}_{28} + \bar{I}_{29} I_{28} + I_{29} \bar{I}_{28} + I_{29} I_{28} I_{27} I_{26}) &&; \text{STORE i, STOREIN1 i,} \\ & && \text{STOREIN2 i, MOVE mit } S=ACC \end{aligned}$$

Dann ist $/ACCDdoe = \neg ACCDdoe_{pre}$.

- b) $f[2 : 0]$:

- Bei Compute Befehlen ist $f[2 : 0] = I[28 : 26]$, bei allen anderen Befehlen soll die ALU addieren ($f[2 : 0] = 011$).
- $f[2] = \bar{I}_{31} \bar{I}_{30} I_{28}$.
- $f[1] = \bar{I}_{31} \bar{I}_{30} I_{27} + \overline{(\bar{I}_{31} \bar{I}_{30})}$.
- $f[0] = \bar{I}_{31} \bar{I}_{30} I_{26} + (\bar{I}_{31} \bar{I}_{30})$.

- c) $sext$:

- Sign Extension wird bei all den Befehlen durchgeführt, bei denen der Parameter i als Zweierkomplementzahl interpretiert wird.

- Sign Extension wird bei ADDI, SUBI, LOADINj, STOREINj und JUMP (**Wichtig:** JUMP wurde in der Vorlesung vergessen, daher muss es keinen Abzug geben – weist aber darauf hin) durchgeführt.
- Sign Extension darf nicht bei LOADI und logischen Compute-Immediate Befehlen durchgeführt werden (in diesen Fällen wird mit 0^8 aufgefüllt).
- Boolescher Ausdruck:

$$\begin{aligned} & (I_{31} \oplus I_{30}) \cdot (I_{29} \oplus I_{28}) + && (\text{LOADINj und STOREINj}) \\ & \bar{I}_{31} \cdot \bar{I}_{30} \cdot \bar{I}_{29} \cdot \bar{I}_{28} + && (\text{ADDI, SUBI}) \\ & I_{31} \cdot I_{30} && (\text{JUMP}) \end{aligned}$$

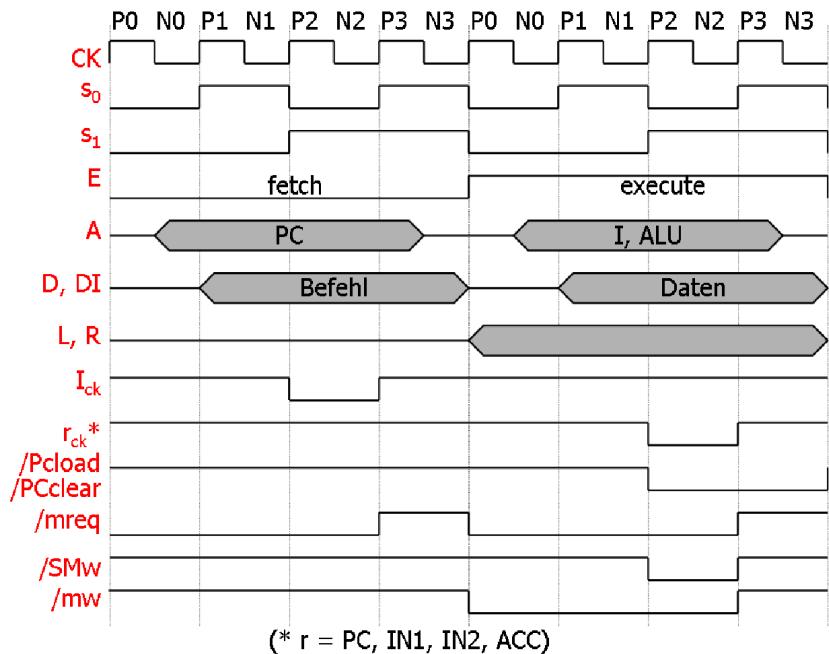


Abbildung 2: Timingdiagramm ReTI

Aufgabe 4 (1 + 3 Punkte)

- Zeigen Sie für die Übergänge an den Eingängen eines OR-Gatters bei denen ein Spike am Ausgang auftreten kann, wie und unter welchen zeitlichen Bedingungen spikefreies Umschalten sicher gewährleistet werden kann. Die Verzögerungszeiten für ein OR-Gatter finden Sie in den Zusatzmaterialien.
- Sie finden das in der Vorlesung vorgestellte RS-FlipFlop langweilig und beschließen daher, einen alternativen Schaltplan (siehe Abb. 3) als speicherndes Element zu betrachten — jedenfalls bis zum Ende dieser Aufgabe. Die stabilen Belegungen finden Sie in Tabelle 1.

Geben Sie an, in welchen Intervallen die Gatter bei einem (genügend langen) Puls auf S bzw. $/R$ schalten.

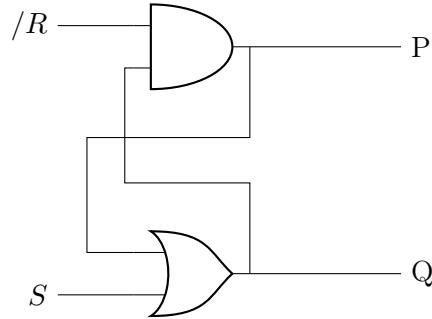


Abbildung 3: Schon wieder ein anderes RS-Flipflop.

| S | $/R$ | Q | P | |
|-----|------|-----|-----|-------------------------------------|
| 0 | 1 | 0 | 0 | Werte von Q und P halten. |
| 0 | 1 | 1 | 1 | Werte von Q und P halten. |
| 1 | 1 | 1 | 1 | Werte von Q und P auf 1 setzen. |
| 0 | 0 | 0 | 0 | Werte von Q und P auf 0 setzen. |
| 1 | 0 | 0 | 1 | Verbotene Belegung. |

Tabelle 1: Stabile Belegungen von Ihrem RS-Flipflop.

Wie lange muss – ausgehend von der Q und P haltenden Eingangsbelegung – der Puls auf S bzw. $/R$ mindestens sein, damit ein spikefreies Umschalten der Ausgänge sichergestellt ist? Begründen Sie Ihre Lösungswege.

Lösung:

- a) Bei $a = 0, b = 1 \leftrightarrow a = 1, b = 0$ (oder umgekehrt) kann ein Spike auftreten. Um dem vorzubeugen, erst das Low Signal anheben, dann das High Signal absenken; zB $a = 0, b = 1 \rightarrow a = 1, b = 1 \rightarrow a = 1, b = 0$.
 $t_{PLH}^{max} + 2\delta = 0.11\text{ ns} + 0.26\text{ ns} = 0.37\text{ ns}$. (t_{PLH}^{max} , weil bei a von 0 → 1 c theoretisch von 0 nach 1 wechseln könnte. Diese Zeit wird abgewartet, auch wenn in diesem Fall natürlich nichts passiert.)
- b) $S: 0 \rightarrow 1$
 - Q steigt zur Zeit $t_0 + \tau_{PLH}^{OR} = t_0 + [0.02, 0.12]$
 - P steigt zur Zeit $t_0 + [0.02, 0.12] + \tau_{PLH}^{AND} = t_0 + [0.02, 0.12] + [0.02, 0.11] = [0.04, 0.23]$
 - S kann wg. spikefreiem Umschalten am OR erst 0.37 ns nach dem Steigen von Q gesenkt werden, also frühestens nach $0.23\text{ ns} + 0.37\text{ ns} = 0.60\text{ ns}$.

$/R: 1 \rightarrow 0$

- P fällt zur Zeit $t_0 + \tau_{PLH}^{AND} = t_0 + [0.02, 0.12]$
- Q fällt zur Zeit $t_0 + [0.02, 0.12] + \tau_{PLH}^{OR} = t_0 + [0.02, 0.12] + [0.04, 0.14] = [0.06, 0.26]$.
- $/R$ kann wg. spikefreiem Umschalten am AND erst 0.38 ns nach dem Fallen von P angehoben werden, also frühestens nach $0.26\text{ ns} + 0.38\text{ ns} = 0.64\text{ ns}$.

Abgabe: 19. Juli 2023, 13⁰⁰ über das Übungsportal