Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Claudia Linnhoff-Popien

mobile and distributed systems group

Tutoriumsblatt 1
Rechnerarchitektur im SoSe 2020

Zu den Modulen A, B

Tutorium: Im Rahmen eines einstündigen wöchentlichen Tutoriums werden Aufgaben zur vorangegangenen Vorlesung vorgestellt. Die Aufzeichnungen zum Tutorium 1 werden am 23. April 2020 (17 Uhr) online zur Verfügung gestellt.

Ankündigungen: Um kurzfristige Ankündigungen nicht zu verpassen, bitten wir Sie regelmäßig die Website zur Vorlesung zu besuchen:
`http://www.mobile.ifi.lmu.de/lehrveranstaltungen/rechnerarchitektur-sose20/`

Aufgabe T1: Bunte Bilder

(– Pkt.)

Manche Digitalkameras der 16-Megapixel-Klasse haben eine Auflösung von 4992 $\times$ 3328 Punkten (Pixeln). (Das bedeutet, ein Foto besteht aus 4992 $\times$ 3328 Punkten). Gehen Sie bei den folgenden Aufgaben davon aus, dass 1 KB = 2^{10} Byte entspricht.

a. Wie viel Speicher (in Bytes, KB und MB) wird benötigt, um ein unkomprimiertes Bild zu speichern, wenn

   (i) jedes Pixel nur schwarz oder weiß ist?

   (ii) für jedes Pixel 8-Bit Graustufen verwendet werden?

   (iii) für jedes Pixel drei (rot, grün und blau) 8-Bit Farbskalen verwendet werden?

b. Wie viel Speicher (in GB) wird benötigt, um - ohne Kompression - eine Minute Film zu speichern, wenn die Kamera 25 Bilder pro Sekunde aufzeichnet? Geben Sie die Antwort wieder für

   (i) Schwarz-Weiß-Bilder,

   (ii) 8-Bit Graustufen-Bilder und

   (iii) 3*8 Bit-Farbbilder an.

c. Wie viele Sekunden unkomprimierten Videos können auf einer DVD-5 mit 4,700,000,000 Byte ($\approx$ 4,38 GB) Kapazität gespeichert werden? Antwort wieder für

   (i) Schwarz-Weiß-Bilder,

   (ii) 8-Bit Graustufen-Bilder und

   (iii) 3*8 Bit-Farbbilder.

Aufgabe T2: Bits und Bytes

(– Pkt.)

Verschiedene Speichermedien besitzen unterschiedliche Kapazitäten. Im Folgenden sind einige Speichermedien und mögliche Kapazitäten aufgeführt:

i. Diskette/USB-Stick: a) 1,44 MBytes, b) 2 GBytes

ii. CDs/Blu-ray Disc Dual Layer: a) 700 MBytes, b) 50 GBytes

a. Konvertieren Sie für jedes Medium die Kapazität in α) Anzahl Bits, β) Anzahl Bytes, γ) Anzahl Kilobytes, δ) Anzahl Megabytes, ϵ) Anzahl Gigabytes und ϕ) Anzahl Terabytes. Gehen Sie davon aus, dass 1 KByte 2^{10} Byte entspricht und geben Sie ungerade Ergebnisse mit genügend Nachkommastellen an, so dass Ihr Ergebnis korrekt überprüft werden kann. Ein Rechenweg ist nicht notwendig.

b. Ist eine solche Umrechnung jeweils sinnvoll? Begründen Sie Ihre Antwort!

Aufgabe T3: Von-Neumann-Modell

(– Pkt.)

Bearbeiten Sie folgende Teilaufgaben zum Von-Neumann-Modell.

a. Skizzieren Sie die grundlegende Architektur der Von-Neumann-Rechner und beschreiben Sie kurz die Funktionalität der einzelnen Komponenten.

b. Erklären Sie, wie eine Programmzeile Code im Von-Neumann-Modell abgearbeitet wird und welcher Vorteil sich ergibt, wenn Programm und Daten in dem selben Speicher gehalten werden.

c. Beschreiben Sie die Probleme der von-Neumann-Architektur und deren Lösungsmöglichkeiten.

d. Der Speicher hat 2^n Zellen. Jede Zelle kann 4 Byte aufnehmen. Wie breit müssen jeweils Adress- und Datenbus sein (d.h. aus wie vielen Leitungen bestehen die Busse) unter der Annahme, dass pro Leitung 1 Bit kodiert werden kann?

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Claudia Linnhoff-Popien

mobile and distributed systems group

Tutoriumsblatt 2
Rechnerarchitektur im SoSe 2020

Zu den Modulen C, D

Tutorium: Die Aufgaben werden in Tutorien-Videos vorgestellt, die am 30. April 2020 (17 Uhr) veröffentlicht werden.

Aufgabe T4: Boolesche Algebra

(– Pkt.)

Beweisen Sie unter Verwendung des Kommutativ-, Distributiv-, Identitäts- und Komplementärgesetzes (und nur mit diesen alleine) die Gültigkeit folgender Aussagen (Es reicht also nicht die Eigenschaften für {0, 1} zu zeigen!). Hinweis: Sie können bereits bewiesene Aussagen verwenden, um darauf folgende Aussagen zu beweisen.

a. Idempotenz

   (i) $a \cdot a = a$ bzw. (ii) $a + a = a$

b. Null- und Einsgesetz

   (i) $a \cdot 0 = 0$ bzw. (ii) $a + 1 = 1$

c. Absorptionsgesetz

   (i) $a \cdot (a + b) = a$ bzw. (ii) $a + (a \cdot b) = a$

Aufgabe T5: Funktionstabelle

(– Pkt.)

Gegeben sei folgende Booleschen Funktion $f(a, b, c) = a \wedge b \wedge (a \vee \overline{c})$. Füllen Sie folgende Funktionstabelle aus:

a	b	c	$f(a, b, c) = a \wedge b \wedge (a \vee \overline{c})$

Aufgabe T6: Decoder

(– Pkt.)

Bearbeiten Sie die folgenden Aufgaben:

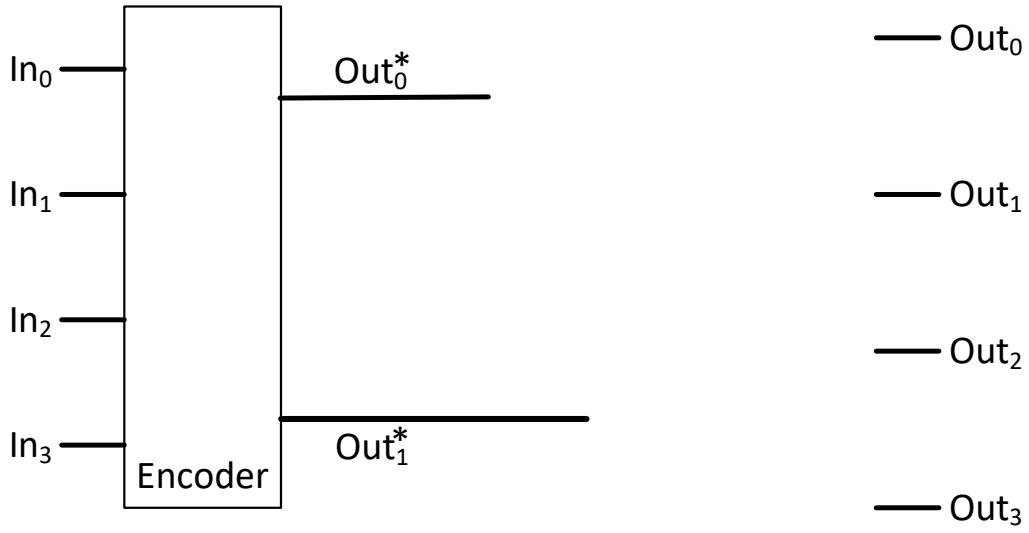
a. Wie viele Ausgänge können beim Decoder gleichzeitig den Wert `wahr` annehmen?

b. Wie viele Eingangsleitungen benötigt ein Decoder, der 16 Ausgangsleitungen besitzt?

c. Stellen Sie die Kurzform der Funktionstabelle eines 2-zu-4-Decoders mit den Eingangsleitungen In_0, In_1 und den Ausgangsleitungen $Out_0, Out_1, Out_2, Out_3$ auf. Tragen Sie Ihre Lösung in die folgende Tabelle ein:

In_0	In_1	Out_0	Out_1	Out_2	Out_3

d. Ergänzen Sie das folgende Schaltnetz so, dass stets gilt $Out_0 = In_0$, $Out_1 = In_1$, $Out_2 = In_2$ und $Out_3 = In_3$. Bei Ihrer Ergänzung dürfen Sie nur auf das Signal an den Leitungen Out_0^* und Out_1^* zugreifen. Es dürfen ausschließlich Leitungen, `NOT`-, `AND`- und `OR`-Bausteine ergänzt werden.



Aufgabe T7: 2-zu-1 Multiplexer

(– Pkt.)

In dieser Aufgabe soll ein 2-zu-1 Multiplexer entworfen werden. Als Input erhält der Multiplexer zwei 1-Bit Kanäle A und B sowie eine 1-Bit Auswahlleitung S. Als Ausgabe liefert der Multiplexer einen 1-Bit Kanal Z. Der Multiplexer soll den Kanal A auf Z schalten, wenn die Auswahlleitung S auf 0 steht. Wenn die S auf 1 steht, soll der Multiplexer den Kanal B auf Z schalten.

a. Erläutern Sie kurz die Funktionsweise eines Multiplexers.

b. Geben Sie die Funktionstabelle, die Boolesche Funktion und das Schaltnetz an.

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

Tutoriumsblatt 3
Rechnerarchitektur im SoSe 2020

Zu den Modulen E, F

Tutorium: Die Aufgaben werden in Tutorien-Videos vorgestellt, die am 07. Mai 2020 (17 Uhr) veröffentlicht werden.

Aufgabe T8: Normalformen einer Schaltfunktion

(– Pkt.)

Gegeben ist folgende Wahrheitstabelle:

a	b	c	d	f(a,b,c,d)
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

a. Geben Sie die Schaltfunktion von f in disjunktiver Normalform (DNF) an.

b. Geben Sie die Schaltfunktion von f in konjunktiver Normalform (KNF) an.

Aufgabe T9: Schaltfunktion in DNF bzw. KNF und Entwurf eines Schaltnetzes

(– Pkt.)

In einer Gefahrenmeldeanlage sollen drei Gefahrentypen durch drei Lämpchen angezeigt werden. Spricht nur einer der drei Melder (a, b, c) an, soll die gelbe Lampe G leuchten ($G = 1$). Melden zwei Melder gleichzeitig, soll die orange Lampe O leuchten ($O = 1$) und nur wenn alle drei Melder Alarm geben, soll die rote Lampe R aufleuchten ($R = 1$).

a. Stellen Sie die Funktionstabelle der Gefahrenmeldeanlage auf.

b. Leiten Sie aus der Funktionstabelle die Schaltfunktionen für Ausgang R sowohl in disjunktiver Normalform (DNF), als auch in konjunktiver Normalform (KNF) her.

c. Welche der beiden Darstellungen (KNF, DNF) ist in diesem Fall günstiger? Begründen Sie ihre Aussage.

d. Zeichnen Sie ein Schaltbild für den Ausgang G.

Aufgabe T10: Programmierbare logische Arrays

(– Pkt.)

a. Erläutern Sie kurz die grundlegende Idee eines PLAs!

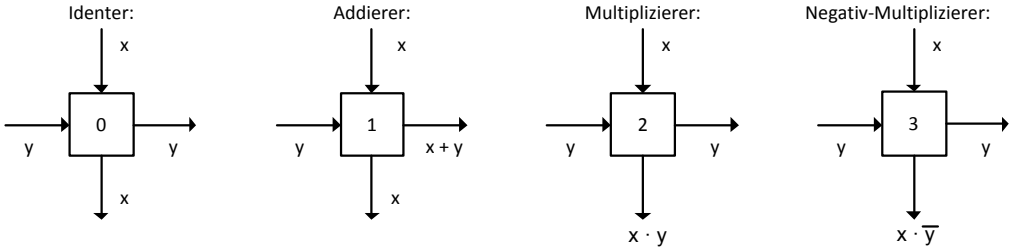
b. Erläutern Sie, was es bedeutet, wenn Eingänge

(i) neutralisiert werden!

(ii) gesperrt werden!

c. Ein normiertes PLA besteht aus einer Und-Ebene und aus einer Oder-Ebene. Erklären Sie diese beiden Begriffe kurz. Ausgehend von einem 5-mal-4-PLA: Wie groß werden Und- und Oder-Ebene jeweils, wenn durch das PLA eine dreistellige Boolesche Funktion realisiert werden soll?

d. Intern ist jedes PLA gitterförmig verdrahtet, wobei sich an jedem Kreuzungspunkt von zwei Drähten einer von vier möglichen Bausteinen befindet. Diese Bausteine sind:



Zeichnen Sie das Schaltbild für jeden der vier Bausteine. Verwenden Sie dazu Und-, Oder- und Nicht-Gatter!

e. Gegeben sei die folgende Boolesche Funktion $f : B^3 \to B^2$:

$$f(x, y, z) = (x \wedge y \wedge \neg z) \vee (x \wedge z), (x \wedge y \wedge \neg z) \vee (x \wedge \neg y \wedge z).$$

Realisieren Sie diese Funktion durch ein normiertes PLA, welches aus der minimal möglichen Anzahl an Zeilen und Spalten besteht. Verwenden Sie ausschließlich die in Aufgabenteil d) gegebenen Bausteine vom Typ 0 bis 3. Kennzeichnen Sie in Ihrer Skizze die Und- und die Oder-Ebene. Markieren Sie gesperrte und neutralisierte Eingänge. Beschriften Sie jeden Pfeil (sowohl ausgehende als auch die innerhalb des PLAs) mit der jeweils anliegenden logischen Funktion.

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

Tutoriumsblatt 4
Rechnerarchitektur im SoSe 2020

Zu den Modulen G, H

Tutorium: Die Aufgaben werden in Tutorien-Videos vorgestellt, die am 14. Mai 2020 (17 Uhr) veröffentlicht werden.

Aufgabe T11: Minimierung mittels Karnaugh

(– Pkt.)

Minimieren Sie folgende Funktionen mit Hilfe des Karnaugh-Diagramms.
Geben Sie dabei sowohl das jeweilige gezeichnete Karnaugh-Diagramm, als auch die zugehörige minimierte Funktion in disjunktiver Form an!

a. $y_1 = (x_1 x_2 \overline{x}_3) + (x_1 \overline{x}_2 \overline{x}_3) + (\overline{x}_1 \overline{x}_2 \overline{x}_3) + (\overline{x}_1 x_2 \overline{x}_3) + (x_1 \overline{x}_2 x_3) + (x_1 x_2 x_3)$

b. $y_2 = (\overline{x}_2 x_3 x_4) + (\overline{x}_1 x_2 x_3 x_4) + (x_1 x_2 \overline{x}_3 x_4) + (\overline{x}_1 x_2 \overline{x}_3 x_4) + (\overline{x}_1 x_2 x_3 \overline{x}_4) + (\overline{x}_1 x_2 \overline{x}_3 \overline{x}_4) + (\overline{x}_1 \overline{x}_2 \overline{x}_3 \overline{x}_4)$

Aufgabe T12: Schaltfunktion

(– Pkt.)

Gegeben ist folgende Wahrheitstabelle:

a	b	c	d	f(a,b,c,d)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

a. Geben Sie die Schaltfunktion von f in disjunktiver Normalform (DNF) an.

b. Vereinfachen Sie die Funktion unter Verwendung eines Karnaugh-Diagramms.

c. Nehmen Sie an, dass die Wahrheitstabelle wie oben gegeben ist, jedoch ohne die letzte Zeile. Das heißt, die neue Funktion f' ist auf dem Eingabe-4-Tupel (a=1, b=1, c=1, d=1) undefiniert. Wie wirkt sich das auf Ihre Möglichkeiten aus, die neue Funktion f' zu vereinfachen? Verdeutlichen Sie Ihre Antwort an einem neuen Karnaugh-Diagramm, und geben Sie eine möglichst einfache Darstellung von f' an.

Aufgabe T13: Quine-McCluskey

(– Pkt.)

a. Vereinfachen Sie den folgenden Booleschen Term unter Anwendung des Algorithmus von Quine-McCluskey:
$f(x) = \overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_1x_2x_3x_4 + x_1\overline{x}_2\overline{x}_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3x_4 + x_1\overline{x}_2x_3x_4 + x_1x_2\overline{x}_3\overline{x}_4 + x_1x_2\overline{x}_3x_4$
Geben Sie dabei alle notwendigen Schritte an!

b. Berechnen Sie die Kosten vor und nach der Optimierung. Wie viel kann an Kosten eingespart werden?

c. Begründen Sie, ob in diesem Beispiel auch eine Optimierung mittels Karnaugh-Diagrammen möglich wäre.

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

Tutoriumsblatt 5
Rechnerarchitektur im SoSe 2020

Zu den Modulen C, D, G

Tutorium: Die Aufgaben werden in Tutorien-Videos vorgestellt, die am 20. Mai 2020 (17 Uhr) veröffentlicht werden.

Aufgabe T14: Boolesche Aussagen

(– Pkt.)

In dieser Aufgabe sind Beispiele für aussagenlogische Ausdrücke $z = f(x, y)$ gegeben. Stellen Sie für jedes Beispiel die Wahrheitstabelle auf und ordnen Sie dem Beispiel eine der 16 zweistelligen Boolesche Funktionen von Seite 52 des Skriptes zu! Entscheiden Sie zudem, ob es günstiger wäre, die Funktion in DNF oder KNF anzugeben und geben Sie die jeweilige DNF oder KNF an!

a. x bedeutet: Es regnet.
   y bedeutet: Ich habe einen Schirm dabei.
   z bedeutet: Ich kann nach draußen gehen ohne nass zu werden.

b. x bedeutet: Es ist ein Gang eingelegt (die Kupplung soll nicht beachtet werden).
   y bedeutet: Das Gaspedal wird betätigt.
   z bedeutet: Das Fahrzeug bewegt sich nach vorn.

c. x bedeutet: Es ist nicht windig.
   y bedeutet: Die Sonne scheint.
   z bedeutet: Ich kann einen Drachen steigen lassen.

d. x bedeutet: Der Zug kommt zu spät.
   y bedeutet: Es steht ein Taxi als Alternativverbindung zur Verfügung.
   z bedeutet: Ich komme zu spät zu meinem Termin.

e. x bedeutet: Team X zieht am Tau.
   y bedeutet: Team Y zieht am Tau.
   z bedeutet: Es gewinnt eines der Teams (beide sind gleich stark) beim Tauziehen.

Aufgabe T15: Encoder

(– Pkt.)

Ein Encoder besitzt die umgekehrte Funktionalität eines Dekoders. Er besitzt 4 Eingänge I_0, I_1, I_2, I_3 und die zwei Ausgänge Out_0 und Out_1. Es wird angenommen, dass stets genau einer der Eingänge mit einer 1 belegt ist. Ist ein Eingang I_j mit einer 1 belegt, so ist (Out_1, Out_0) die duale Darstellung der Deziamalzahl j. Bearbeiten Sie dazu folgende Teilaufgaben:

a. Bestimmen Sie die Schaltfunktion des 4-zu-2-Encoders. Verwenden Sie dabei die Bezeichnungen gemäß der obigen Beschreibung.

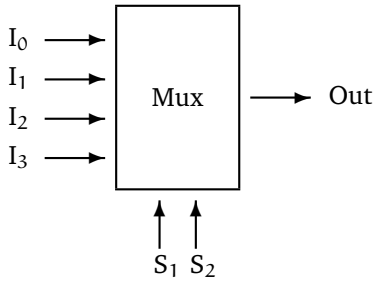
b. Zeichnen Sie das Schaltnetz eines 4-zu-2-Encoders gemäß der obigen Beschreibung eines 4-zu-2-Encoders! *Hinweis:* Die Erstellung der zugehörigen Wahrheitstabelle kann hierbei hilfreich sein.

Aufgabe T16: Multiplexer

(– Pkt.)

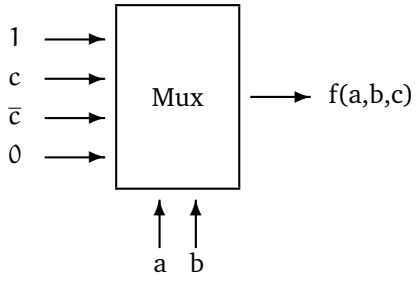
Für einem 4-Eingaben-Multiplexer gilt folgende verkürzte Funktionstabelle:

S_1	S_2	Out
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



Mit Hilfe eines 4-Eingaben-Multiplexers kann die Boolesche Funktion $f(a, b, c)$ dargestellt werden, indem dessen Eingänge bzw. Steuerleitungen wie folgt belegt werden.

	a	b	c	f(a, b, c)
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0



Geben Sie analog zum Beispiel eine Belegung der Eingänge eines 4-Eingaben-Multiplexers $(I_0, \ldots, I_3)$ sowie der Steuerleitungen S_1 und S_2 an, so dass dieser die Boolesche Funktion

$$g(a, b, c) = (\overline{a} \cdot \overline{b} \cdot \overline{c}) + (\overline{a} \cdot b \cdot \overline{c}) + (\overline{a} \cdot b \cdot c) + (a \cdot \overline{b} \cdot c) + (a \cdot b \cdot c)$$

realisiert.
Sie dürfen ausschließlich die Werte a, b, c, $\overline{c}$ sowie 0 und 1 benutzen. Es dürfen keine weiteren Bausteine oder Gatter verwendet werden.

Aufgabe T17: Resolutionsregel

(– Pkt.)

Vereinfachen Sie den folgenden Booleschen Term unter Anwendung der Resolutionsregel soweit wie möglich:

$$f(x_1, x_2, x_3, x_4) = \overline{x}_1\overline{x}_2x_3\overline{x}_4 + \overline{x}_1x_2x_3\overline{x}_4 + x_1\overline{x}_2x_3x_4 + x_1x_2\overline{x}_3x_4 + x_1x_2x_3x_4$$

Geben Sie dabei **alle** notwendigen Zwischenschritte an!

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

Tutoriumsblatt 6
Rechnerarchitektur im SoSe 2020

Zu den Modulen I, J

Tutorium: Die Aufgaben werden in Tutorien-Videos vorgestellt, die am 28. Mai 2020 (17 Uhr) veröffentlicht werden.

Aufgabe T18: Darstellung ganzer Zahlen

(– Pkt.)

a. Geben Sie die folgenden Dezimalzahlen als Dualzahlen in ihrer 1er-Komplement-, 2er-Komplement- und in Sign/Magnitude-Darstellung an (jeweils 10 Bit). Bei der Sign/Magnitude-Darstellung wird das hochwertigste Bit als Vorzeichen interpretiert: $(b_9 ... b_1 b_0)_2 = (-1)^{b_9} * \sum_{i=0}^{8} b_i 2^i$

   (i) $(123)_{10}$

   (ii) $(-123)_{10}$

b. Wandeln Sie folgende Dualzahlen in ihre Dezimaldarstellung um. Interpretieren Sie die Dualzahlen jeweils als in 1er- und 2er-Komplement-Darstellung sowie in Sign/Magnitude-Darstellung gegeben.

   (i) $(1111101011)_2$

   (ii) $(0001011010)_2$

c. Geben Sie jeweils in 1er- und 2er-Komplement-Darstellung und in Sign/Magnitude-Darstellung bei Verwendung von 10 Bits an:

   (i) die größte darstellbare positive Zahl,

   (ii) die kleinste darstellbare positive Zahl,

   (iii) die größte darstellbare negative Zahl (d.h. die negative Zahl, die den geringsten Abstand zur Null hat),

   (iv) die kleinste darstellbare negative Zahl (d.h. die negative Zahl, die den größten Abstand zur Null hat),

   (v) die Zahl Null.

d. Gibt es einen Unterschied zwischen „2er-Komplement“ und „2er-Komplement-Darstellung“? Wenn ja, welchen?

Aufgabe T19: Addition von Dualzahlen

(– Pkt.)

In dieser Aufgabe sollen die Grundlagen der Addition in Einer- bzw. Zweierkomplement-Darstellung vertieft werden. Verwenden Sie zur binären Darstellung sämtlicher vorkommenden Zahlen jeweils 8 Bits.

a. Gegeben seien die Zahlen $(-17)_{10}$ sowie $(7)_{10}$.

   (i) Geben Sie die Einerkomplement-Darstellung der beiden Zahlen an.

   (ii) Geben Sie die Zweierkomplement-Darstellung der beiden Zahlen an.

b. Addieren Sie die Zahlen $(-17)_{10}$ und $(7)_{10}$ binär. Verwenden Sie dazu

   (i) die Einerkomplement-Darstellung.

   (ii) die Zweierkomplement-Darstellung.

c. Addieren Sie nun die Zahlen $(-56)_{10}$ und $(-72)_{10}$ binär. Verwenden Sie dazu

   (i) die Einerkomplement-Darstellung.

   (ii) die Zweierkomplement-Darstellung.

   Beantworten Sie zusätzlich jeweils die Frage, ob ein Überlauf stattgefunden hat. Begründen Sie ihre Antwort kurz.

Aufgabe T20: Gleitkommazahlen

(– Pkt.)

Nach dem IEEE 754 Standard gilt:

$$(-1)^S \cdot (1 + \text{Signifikant}) \cdot 2^{(\text{Exponent}-\text{Bias})}$$

wobei der Standard

– für das Vorzeichen S ein Bit,

– für den Signifikanten (Mantisse) 23 Bit bei einfacher und 52 Bit bei doppelter Genauigkeit,

– für den Exponenten 8 Bit bei einfacher und 11 Bit bei doppelter Genauigkeit

reserviert und den Bias auf $127 = 2^{8-1} - 1$ bei einfacher bzw. auf $1023 = 2^{11-1} - 1$ bei doppelter Genauigkeit setzt.

a. Geben Sie die Darstellung folgender Zahlen als Gleitkommazahl nach IEEE 754 in einfacher (32-Bit) Genauigkeit an:

   (i) $(11,25)_{10}$

   (ii) $(0,2)_{10}$

b. Wandeln Sie folgende Zahl, die in Gleitkommadarstellung (IEEE 754) gegeben ist, in ihre Dezimaldarstellung um.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	1	1	0	1	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	Exponent								Significand																						

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

Tutoriumsblatt 7
Rechnerarchitektur im SoSe 2020

Zu den Modulen K

Tutorium: Die Aufgaben werden in Tutorien-Videos vorgestellt, die am 04. Juni 2020 (17 Uhr) veröffentlicht werden.

Aufgabe T21: Test des MIPS Simulators

(– Pkt.)

Für diese Aufgabe sollten Sie sich mit dem MIPS-Simulator SPIM vertraut machen. Sie können einen MIPS-Simulator von der Vorlesungshomepage herunterladen.

- Laden Sie sich das Assemblerprogramm `simple.s` von der Rechnerarchitektur–Homepage herunter und speichern Sie es in Ihrem Home–Verzeichnis ab.
- Starten Sie Ihren Simulator.
- Laden Sie das Programm `simple.s` in den Simulator und führen Sie es aus. Dabei sollte eine Konsole erscheinen, über die die Ein- und Ausgabe erfolgt.

Beantworten Sie nun folgende Fragen.

a. Welches Ergebnis liefert das Programm für die Eingabefolge "6, 7, 8, 9, 0"(d.h. nach Start des Programms erfolgt über die Konsole die Eingabe "6", gefolgt von *Enter*, dann die Eingabe "7", gefolgt von *Enter*, usw.)?

b. Die folgenden Kommentare beschreiben Teile des Programms `simple.s`. Ordnen Sie den Kommentaren jeweils die minimale Anzahl an Codezeilen zu, die benötigt werden, um das beschriebene Verhalten im Code darzustellen, und geben Sie die Zeilennummer(n) dieser Zeile(n) an!

   i) `str1` wird auf der Konsole ausgegeben.

   ii) Es wird eine Zahl von der Konsole eingelesen.

   iii) Das Programm wird beendet.

   iv) Eine Zählvariable wird um den Wert 1 erhöht.

c. In welchem Wertebereich müssen sich die eingegebenen Zahlen befinden, damit keine Fehlerbehandlung stattfindet (= damit das Label *error* nicht angesprungen wird).

d. Welche mathematische Funktion berechnet das Programm?

Aufgabe T22: Umsetzung Boolescher Ausdrücke

(– Pkt.)

Übersetzen Sie folgendes Pseudocodefragment in MIPS-Code. Gehen Sie davon aus, dass der Wert der Variablen `a` bereits in das Register `$t0` geladen wurde.

```
1 IF (a < 0) OR (a > 99) THEN
2     a := a - 10;
3 ELSE
4     a := a - 1;
5 END;
```

Bedenken Sie dabei insbesondere: Der Ausdruck `a > 99` wird nur dann ausgewertet, wenn `a < 0` fehlgeschlagen ist.

Aufgabe T23: SPIM Programmieraufgabe

(– Pkt.)

Erstellen Sie ein **vollständiges** SPIM–Programm, das folgendes durchführt:

- Es werden zwei positive Integer–Zahlen von der Konsole eingelesen.
- Es wird der Durchschnitt dieser beiden Zahlen auf eine Nachkommastelle genau berechnet.
- Das Ergebnis der Berechnung wird ausgegeben.

Tipp: Programmieren Sie diejenigen Schritte, die Sie auch beim handschriftlichen Dividieren durchführen!

Beachten Sie hierbei folgendes:

- Verwenden Sie nur die **unten aufgeführten Befehle**.
- Verwenden Sie für die Vorkommazahl das Register `$s0` und für die Nachkommazahl das Register `$s1`, ansonsten nur die temporären Register.
- **Kommentieren** Sie ihr Programm sinnvoll!
- Sowohl die Eingabe als auch die Ausgabe soll mit einem Anweisungstext versehen werden, wie z.B. *"Geben Sie die 1. Zahl ein: "*, etc.

Überblick über die wichtigsten SPIM Assemblerbefehle		
Befehl	Argumente	Wirkung
add	Rd, Rs1, Rs2	Rd := Rs1 + Rs2 (mit Überlauf)
sub	Rd, Rs1, Rs2	Rd := Rs1 - Rs2 (mit Überlauf)
addu	Rd, Rs1, Rs2	Rd := Rs1 + Rs2 (ohne Überlauf)
subu	Rd, Rs1, Rs2	Rd := Rs1 - Rs2 (ohne Überlauf)
addi	Rd, Rs1, Imm	Rd := Rs1 + Imm
addiu	Rd, Rs1, Imm	Rd := Rs1 + Imm (ohne Überlauf)
div	Rd, Rs1, Rs2	Rd := Rs1 DIV Rs2
rem	Rd, Rs1, Rs2	Rd := Rs1 MOD Rs2
mul	Rd, Rs1, Rs2	Rd := Rs1 $\times$ Rs2
b	label	unbedingter Sprung nach label
j	label	unbedingter Sprung nach label
jal	label	unbed.Sprung nach label, Adresse des nächsten Befehls in $ra
jr	Rs	unbedingter Sprung an die Adresse in Rs
beq	Rs1, Rs2, label	Sprung, falls Rs1 $=$ Rs2
beqz	Rs, label	Sprung, falls Rs $=$ 0
bne	Rs1, Rs2, label	Sprung, falls Rs1 $\neq$ Rs2
bnez	Rs1, label	Sprung, falls Rs1 $\neq$ 0
bge	Rs1, Rs2, label	Sprung, falls Rs1 $\geq$ Rs2
bgeu	Rs1, Rs2, label	Sprung, falls Rs1 $\geq$ Rs2
bgez	Rs, label	Sprung, falls Rs $\geq$ 0
bgt	Rs1, Rs2, label	Sprung, falls Rs1 $>$ Rs2
bgtu	Rs1, Rs2, label	Sprung, falls Rs1 $>$ Rs2
bgtz	Rs, label	Sprung, falls Rs $>$ 0
ble	Rs1, Rs2, label	Sprung, falls Rs1 $\leq$ Rs2
bleu	Rs1, Rs2, label	Sprung, falls Rs1 $\leq$ Rs2
blez	Rs, label	Sprung, falls Rs $\leq$ 0
blt	Rs1, Rs2, label	Sprung, falls Rs1 $<$ Rs2
bltu	Rs1, Rs2, label	Sprung, falls Rs1 $<$ Rs2
bltz	Rs, label	Sprung, falls Rs $<$ 0
not	Rd, Rs1	Rd := $\neg$ Rs1 (bitweise Negation)
and	Rd, Rs1, Rs2	Rd := Rs1 & Rs2 (bitweises UND)
or	Rd, Rs1, Rs2	Rd := Rs1 | Rs2 (bitweises ODER)
syscall		führt Systemfunktion aus
move	Rd, Rs	Rd := Rs
la	Rd, label	Adresse des Labels wird in Rd geladen
lb	Rd, Adr	Rd := MEM[Adr]
lw	Rd, Adr	Rd := MEM[Adr]
li	Rd, Imm	Rd := Imm
sw	Rs, Adr	MEM[Adr] := Rs (Speichere ein Wort)
sh	Rs, Adr	MEM[Adr] MOD 2^{16} := Rs (Speichere ein Halbwort)
sb	Rs, Adr	MEM[Adr] MOD 256 := Rs (Speichere ein Byte)

Funktion	Code in $v0	Funktion	Code in $v0
print_int	1	read_float	6
print_float	2	read_double	7
print_double	3	read_string	8
print_string	4	sbrk	9
read_int	5	exit	10

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

Tutoriumsblatt 8
Rechnerarchitektur im SoSe 2020

Zu den Modulen K

Tutorium: Die Aufgaben werden in Tutorien-Videos vorgestellt, die am 11. Juni 2020 (17 Uhr) veröffentlicht werden.

Aufgabe T24: Parameterübergabe bei Unterprogrammaufrufen

(– Pkt.)

Für die Parameterübergabe bei Prozeduraufrufen existieren verschiedene Möglichkeiten.

a. Erläutern Sie zunächst die Begriffe *call by value* und *call by reference*. Geben Sie zu beiden Konzepten je ein Beispiel in einer Hochsprache an.

b. Schreiben Sie nun ein SPIM-Programm, das den Durchschnitt der Werte eines Feldes berechnet. Die Berechnung selbst soll dabei ein Unterprogramm erledigen. Die Übergabe des Feldes soll nach dem Konzept *call by value* erfolgen.

   Achtung: Das Hauptprogramm soll dem Unterprogramm **alle** zur Berechnung notwendigen Werte über den Stack zur Verfügung stellen! Sie dürfen bei Ihrer Implementierung davon ausgehen, dass sich das Feld bereits im Speicher befindet.

c. Schreiben Sie Ihr Programm aus Aufgabe b) so um, dass die Übergabe des Feldes nach dem Konzept *call by reference* funktioniert.

   Achtung: Das Hauptprogramm soll dem Unterprogramm **ausschließlich** Speicheradressen zur Berechnung zur Verfügung stellen! Sie dürfen wieder davon ausgehen, dass sich das Feld bereits im Speicher befindet. Sie dürfen zur Übergabe der Adressen an das Unterprogramm die laut Konvention dafür vorgesehenen Register $a0 - $a3 verwenden. Das Ergebnis des Unterprogrammaufrufes dürfen Sie dem Hauptprogramm über das Register $v0 zur Verfügung stellen.

Aufgabe T25: SPIM: 2er-Komplement-Darstellung

(– Pkt.)

a. Schreiben Sie ein MIPS-Assembler-Programm, das eine positive bzw. eine negative Dezimalzahl einliest und deren Binärdarstellung unter Verwendung der 2er-Komplement-Darstellung ausgibt. Verwenden sie den Systemaufruf `read_int ($v0 := 5)`, um die Dezimalzahl von der Konsole einzulesen. Testen Sie Ihr Programm mit verschiedenen positiven und negativen Eingaben.

b. Was ist die größte und die kleinste Dezimalzahl für die Ihr Programm korrekt funktioniert? Begründen sie Ihre Antwort

Überblick über die wichtigsten SPIM Assemblerbefehle		
Befehl	Argumente	Wirkung
add	Rd, Rs1, Rs2	Rd := Rs1 + Rs2 (mit Überlauf)
sub	Rd, Rs1, Rs2	Rd := Rs1 - Rs2 (mit Überlauf)
addu	Rd, Rs1, Rs2	Rd := Rs1 + Rs2 (ohne Überlauf)
subu	Rd, Rs1, Rs2	Rd := Rs1 - Rs2 (ohne Überlauf)
addi	Rd, Rs1, Imm	Rd := Rs1 + Imm
addiu	Rd, Rs1, Imm	Rd := Rs1 + Imm (ohne Überlauf)
div	Rd, Rs1, Rs2	Rd := Rs1 DIV Rs2
rem	Rd, Rs1, Rs2	Rd := Rs1 MOD Rs2
mul	Rd, Rs1, Rs2	Rd := Rs1 $\times$ Rs2
b	label	unbedingter Sprung nach label
j	label	unbedingter Sprung nach label
jal	label	unbed.Sprung nach label, Adresse des nächsten Befehls in $ra
jr	Rs	unbedingter Sprung an die Adresse in Rs
beq	Rs1, Rs2, label	Sprung, falls Rs1 = Rs2
beqz	Rs, label	Sprung, falls Rs = 0
bne	Rs1, Rs2, label	Sprung, falls Rs1 $\neq$ Rs2
bnez	Rs1, label	Sprung, falls Rs1 $\neq$ 0
bge	Rs1, Rs2, label	Sprung, falls Rs1 $\geq$ Rs2
bgeu	Rs1, Rs2, label	Sprung, falls Rs1 $\geq$ Rs2
bgez	Rs, label	Sprung, falls Rs $\geq$ 0
bgt	Rs1, Rs2, label	Sprung, falls Rs1 > Rs2
bgtu	Rs1, Rs2, label	Sprung, falls Rs1 > Rs2
bgtz	Rs, label	Sprung, falls Rs > 0
ble	Rs1, Rs2, label	Sprung, falls Rs1 $\leq$ Rs2
bleu	Rs1, Rs2, label	Sprung, falls Rs1 $\leq$ Rs2
blez	Rs, label	Sprung, falls Rs $\leq$ 0
blt	Rs1, Rs2, label	Sprung, falls Rs1 < Rs2
bltu	Rs1, Rs2, label	Sprung, falls Rs1 < Rs2
bltz	Rs, label	Sprung, falls Rs < 0
not	Rd, Rs1	Rd := $\neg$ Rs1 (bitweise Negation)
and	Rd, Rs1, Rs2	Rd := Rs1 & Rs2 (bitweises UND)
or	Rd, Rs1, Rs2	Rd := Rs1 | Rs2 (bitweises ODER)
syscall		führt Systemfunktion aus
move	Rd, Rs	Rd := Rs
la	Rd, label	Adresse des Labels wird in Rd geladen
lb	Rd, Adr	Rd := MEM[Adr]
lw	Rd, Adr	Rd := MEM[Adr]
li	Rd, Imm	Rd := Imm
sw	Rs, Adr	MEM[Adr] := Rs (Speichere ein Wort)
sh	Rs, Adr	MEM[Adr] MOD 2^{16} := Rs (Speichere ein Halbwort)
sb	Rs, Adr	MEM[Adr] MOD 256 := Rs (Speichere ein Byte)

Funktion	Code in $v0	Funktion	Code in $v0
print_int	1	read_float	6
print_float	2	read_double	7
print_double	3	read_string	8
print_string	4	sbrk	9
read_int	5	exit	10

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

Tutoriumsblatt 9
Rechnerarchitektur im SoSe 2020

Zu den Modulen L

Tutorium: Die Aufgaben werden in Tutorien-Videos vorgestellt, die am 18. Juni 2020 (17 Uhr) veröffentlicht werden.

Aufgabe T27: Entwurf eines 4-Bit-Addiernetzes

(– Pkt.)

Es soll systematisch ein Addiernetz entworfen werden, das in der Lage ist, zwei 4-stellige Dualzahlen zu addieren. Dazu wird das Problem aufgespaltet, indem man überlegt, wie eine einzelne Stelle addiert wird.

a. Entwerfen Sie einen Halbaddierer, der in der Lage ist, zwei einstellige Dualzahlen zu addieren.

b. Entwerfen Sie einen Volladdierer, der in der Lage ist, eine beliebige Stelle zweier n-stelliger Dualzahlen zu addieren.

c. Entwerfen Sie nun das Addiernetz, indem Sie Halb- und Volladierer verwenden.

Aufgabe T28: Einfache ALU

(– Pkt.)

Entwerfen Sie eine einfache 1-Bit ALU, die den folgenden Spezifikationen genügt:

i. Operationen: `AND`, `OR`, Addition und Subtraktion.

ii. Inputs: Operanden a und b, CarryIn (Übertrag aus einer vorgeschalteten ALU), gewisse Steuerleitungen (z.B. zur Auswahl des Typs der Operation).

iii. Outputs: Resultat, CarryOut (Übertrag).

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Claudia Linnhoff-Popien

mobile and distributed systems group

Tutoriumsblatt 10
Rechnerarchitektur im SoSe 2020

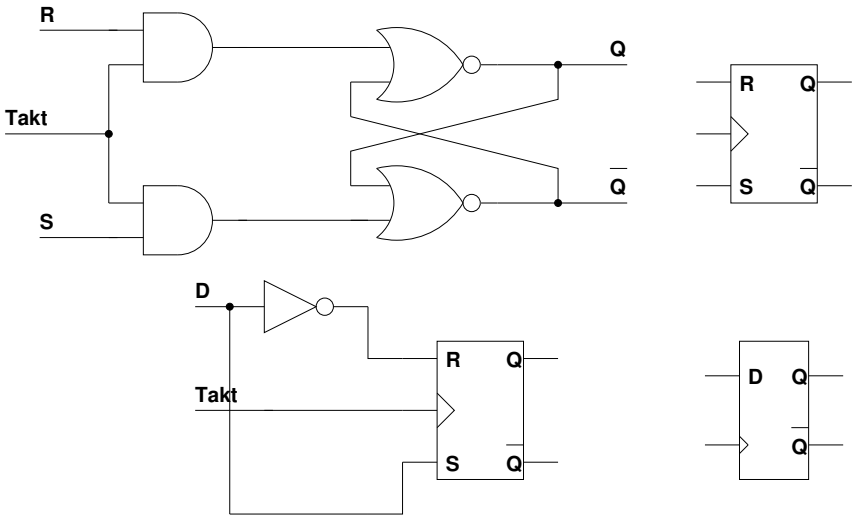
Zu den Modulen M

Tutorium: Die Aufgaben werden in Tutorien-Videos vorgestellt, die am 25. Juni 2020 (17 Uhr) veröffentlicht werden.

Aufgabe T29: Arbeitsweise von Latches

(– Pkt.)

Betrachten Sie die folgenden Schaltbilder eines RS- und eines D-Latches:



a. Machen Sie sich die Funktionsweise der Latches klar, indem Sie die Zustandstabellen aufstellen. Jede Tabelle soll folgendermaßen aufgebaut sein:

- Jede Spalte entspricht einem Ein- bzw. Ausgang. Ein RS-Latch zum Beispiel verfügt über die drei Eingänge S, R und C (Clock/Takt), sowie über die Ausgänge Q und $\overline{Q}$.
- Jede Zeile entspricht einem bestimmten Zustand des Latches, abhängig von den Signalen an den Eingängen.
- Mögliche Zustände sind: Set, Reset, Speichern und Kippen. Geben Sie hinter jeder Tabellenzeile an, welcher Zustand vorliegt. (Hinweis: Nicht alle Zustände kommen bei jedem Latch vor.)
- Kennzeichnen Sie unzulässige Zustände als solche.
- Verwenden Sie *Don't-Care-Argumente*, falls es keinen Unterschied für die Belegung der Ausgänge macht, ob an einem Eingang eine 0 oder 1 anliegt. Verwenden Sie zur Kennzeichnung solcher Belegungen in der Tabelle ein D.

- Verwenden Sie die Notation Q^*, um den alten Wert von Q zu symbolisieren, falls dieser in diesem Zustand nicht explizit (0 oder 1) bekannt ist.

b. Welchen Vorteil besitzt das D-Latch gegenüber dem RS-Latch?

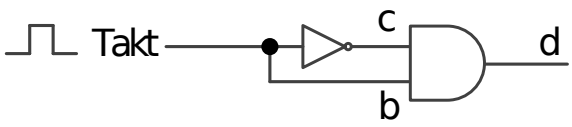
c. Welches Problem ergibt sich aber beim D-Latch im Hinblick auf das Speichern über mehrere Takte hinweg? Verdeutlichen Sie das Problem durch ein Impulsdiagramm, das die Verläufe der Signale D, C (Clock/Takt) und Q darstellt. Setzen Sie alle drei Signale anfangs auf 0, zeichnen Sie dann zunächst den Verlauf für das Taktsignal und führen Sie anschließend eine Set-Operation durch.

Aufgabe T30: Flip-Flops

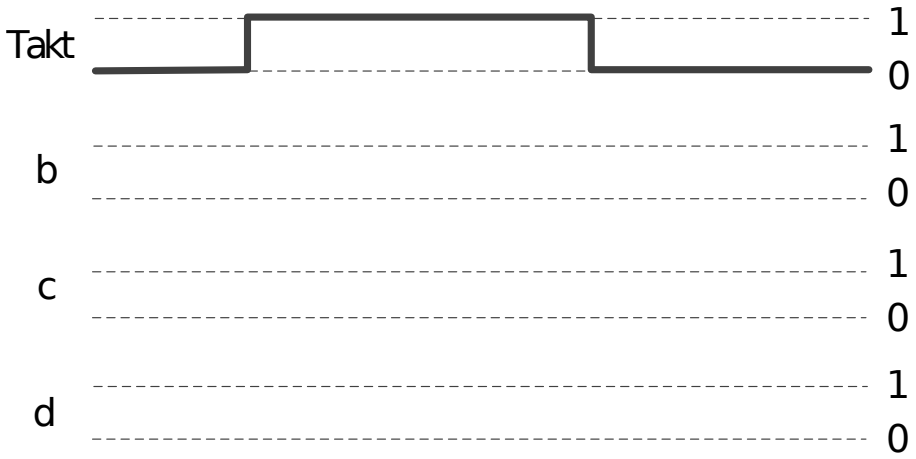
(– Pkt.)

Bearbeiten Sie die folgenden Teilaufgaben zum Thema Flip-Flops:

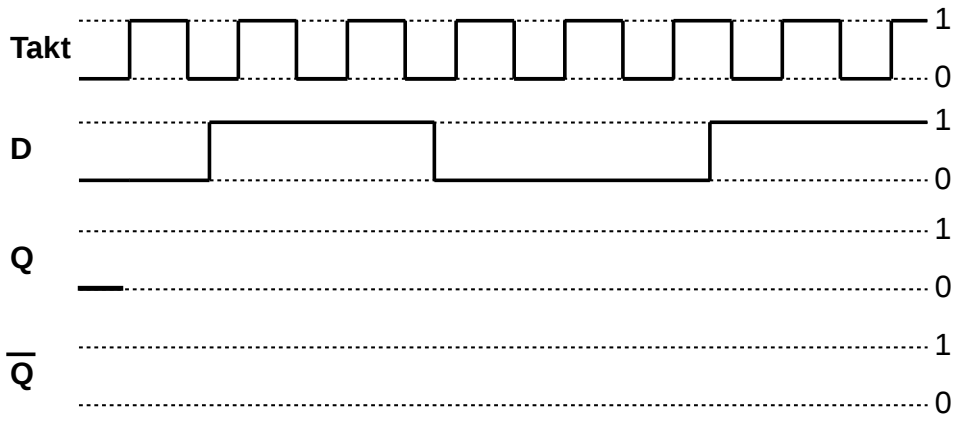
a. Gegeben sei folgendes Schaltnetz, welches einen Impulsgenerator realisiert, der aus Taktflanken kurze Impulse erzeugt:



Ergänzen Sie folgende Vorlage zu einem Impulsdiagramm für die Ausschnitte b, c, d basierend auf dem eingezeichnetem Takt. Gehen Sie davon aus, dass das AND–Gatter keine Verzögerung verursacht und das NOT–Gatter eine nicht vernachlässigbare Verzögerung verursacht, deren Auswirkungen im Impulsdiagramm deutlich werden müssen:



b. Gegeben sei das nachfolgende Impulsdiagramm eines D-Flip-Flops mit dem Taktgeber aus der vorherigen Teilaufgabe a. Vervollständigen Sie das folgende Impulsdiagramm für die Ausgänge Q und $\bar{Q}$ unter der Annahme, dass der Baustein ohne Zeitverzögerung schaltet:


Takt
D
Q
$\overline{Q}$
1
0
1
0
1
0
1
0

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

Tutoriumsblatt 11
Rechnerarchitektur im SoSe 2020

Zu den Modulen N

Tutorium: Die Aufgaben werden in Tutorien-Videos vorgestellt, die am 02. Juli 2020 (17 Uhr) veröffentlicht werden.

Aufgabe T31: Gate-Assignment mittels Quantenannealing

(– Pkt.)

In dieser Aufgabe sollen Sie das Gate-Assignment-Problem für drei Flugzeuge $\{1, 2, 3\}$ und drei Gates $\{A, B, C\}$ als QUBO formulieren. Es gilt analog zur Vorlesung, dass die (Flugzeug, Gate)-Paare $(1, A)$, $(2, B)$ und $(3, C)$ jeweils zu einer Fluggesellschaft gehören und es als besonders günstig zu bewerten ist, wenn sich die Flugzeuge jeweils am Gate ihrer Fluggesellschaft befinden. Es müssen aber auch „katastrophale“ Ereignisse bewertet werden, wie das Ereignis, dass sich ein Flugzeug gleichzeitig an zwei Gates befindet oder zwei Flugzeuge an einem Gate.
Erstellen Sie eine QUBO-Matrix für dieses Problem und füllen Sie diese mit den Zahlenwerten $\{-2, 0, 5\}$, je nachdem, wie günstig eine Zustandskombination zu bewerten ist, so dass die Optimierung (Minimierung) mittels Quantenannealing stattfinden kann.

Aufgabe T32: Traveling-Salesman-Problem und Annealing

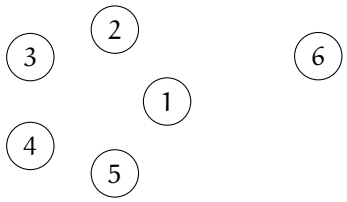
(– Pkt.)

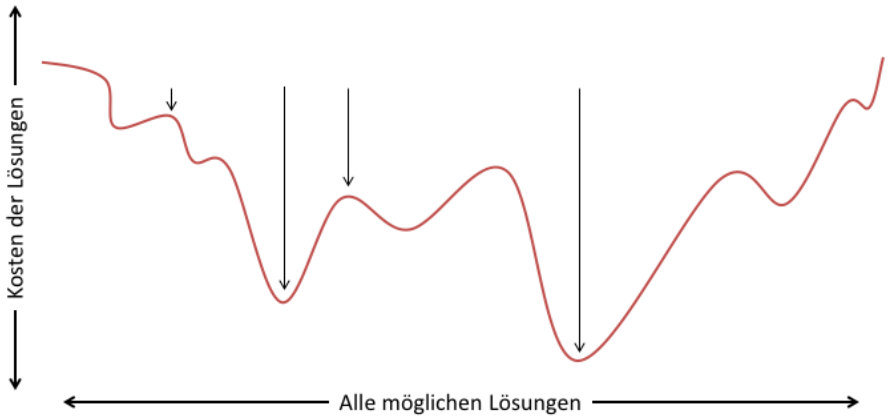
In der Vorlesung haben Sie das Traveling-Salesman-Problem (TSP) und die Optimierung mittel Simulated Annealing kennengelernt. Bearbeiten Sie folgenden Aufgaben dazu:

a. Worum geht es beim TSP?

b. Beschreiben Sie die Optimierungsmethode des Simulated Annealing!

c. Angenommen es liegt eine Graphstruktur vor, bei der die Knoten geografisch wie in folgender Abbildung angeordnet sind (jeder Knoten ist von jedem direkt erreichbar). Ordnen Sie Lösungskandidaten für das TSP auf dieser Graphstruktur den Pfeilen der unten gegebenen Lösungslandschaft zu.




Kosten der Lösungen
Alle möglichen Lösungen