Allgemeine Hinweise zur Vorlesung

Rechnerstrukturen (SS 2000)

Dozent: **Prof. Gerhard Lakemeyer, Ph. D.**
Lehr- und Forschungsgebiets Informatik V

Sprechstunde: Donnerstag 13–14 Uhr (Raum 6212a)

Termine

- Die Vorlesung findet statt:

Mittwoch	8:15–9:45 Uhr	im Roten Hörsaal
Donnerstag	8:15–9:45 Uhr	im Audimax

  Beginn: Mittwoch, den 12.04.2000

- Übungsgruppen finden statt:

Dienstag	8:15– 9:45 Uhr
Dienstag	16:00–17:30 Uhr
Mittwoch	10:45–12:15 Uhr
Mittwoch	12:15–13:45 Uhr
Donnerstag	11:30–13:00 Uhr
Freitag	8:15– 9:45 Uhr

  Beginn: in der Woche ab dem 17.04.2000

- Die Teilnehmer an den von Feiertagen betroffen Übungsgruppen können in den entsprechenden Wochen an einer anderen Übungsgruppe teilnehmen.

Vorlesungs-Homepage

- `http://www-i5.informatik.rwth-aachen.de/LuFG/Lehre/SS00/RS/`

  Hier lassen sich allgemeine und aktuelle Informationen zu Vorlesung und Übungen sowie die Folien der Vorlesung und die Aufgabenblätter abrufen.

Ansprechpartner

- Prof. Gerhard Lakemeyer, Ph. D.
- Dipl.-Inform. Gero Iwan
  Lehr- und Forschungsgebiets Informatik V

  Sprechstunde: Dienstag 14–15 Uhr (Raum 6213)
- und natürlich die Übungsgruppenleiter

Lehrbuch

- Die Vorlesung orientiert sich stark an dem Buch

  Walter Oberschelp / Gottfried Vossen,
  Rechneraufbau und Rechnerstrukturen,
  R. Oldenbourg Verlag, 7. Auflage, 1998.

  Inzwischen ist die 8. Auflage dieses Buches erschienen.
  Zu diesem Lehrbuch sind Hörerscheine erhältlich.

- Empfehlenswert ist auch

  Andrew S. Tanenbaum,
  Structured Computer Organization,
  Prentice Hall, 4th edition, 1999.

Folien

- Die Folien der Vorlesung (und die Aufgabenblätter) werden unter der Vorlesungs-Homepage bereitgestellt:

  User ID: RS2000
  Password:

- Zum Lehrbuch gibt es Folien im WWW:

  einfach dem Lehrbuch-Link auf der Vorlesungs-Homepage folgen

Aufgabenblätter

- Es wird ca. 9 Aufgabenblätter zur Vorlesung geben, die in der Vorlesung ausgegeben und (ebenso wie die Folien) unter der Vorlesungs-Homepage bereitgestellt werden.

Ausgabe:	in der Regel mittwochs in der Vorlesung
Abgabe:	in der darauffolgenden Woche in den Übungsgruppen
Bearbeitung:	in Gruppen von 2–3 Personen (keine „1er-Gruppen“)

  Zur Korrektur abgegeben werden können nur die mit Punkten versehenen Aufgaben.
  Die Abgabe von Lösungen ist freiwillig.

- Die Teilnehmer an den von Feiertagen betroffen Übungsgruppen können ihre Lösungen – mit dem Namen des Übungsgruppenleiters versehen (!) – in den entsprechenden Wochen mittwochs in der Vorlesung abgeben.

Schein-/Test-Klausur

- Am Ende der Vorlesungszeit wird es für diejenigen, die einen Vorlesungs-Schein erwerben müssen, eine Klausur geben:

  (voraussichtlich) statt der Vorlesung am 13.07.2000

- Diese Klausur kann aber auch einfach als Test-Klausur mitgeschrieben werden.
- Die Anmelde-Modalitäten werden rechtzeitig in der Vorlesung und auf der Vorlesungs-Homepage bekannt gegeben.

Vordiploms-Klausur

- Die Vordiploms-Klausur zu dieser Vorlesung findet im Anschluss an und in Kombination mit der Vorlesung „Systemprogrammierung“ des WS 2000/2001 statt.

Lehr- und Forschungsgebiet Informatik V
Prof. G. Lakemeyer, Ph.D.

19.04.2000
G. Iwan, S. Sklorz

Rechnerstrukturen (SS 2000)

Aufgabenblatt 1

Abgabe: 24.04.–28.04.2000 in den Übungen

Bearbeiten Sie zur Vertiefung der Vorlesung nach Möglichkeit stets alle Aufgaben!
Mit Punkten ausgezeichnete Aufgaben können in den Übungen zur Korrektur abgegeben werden.
Bitte kommentieren Sie ihre Lösungen dann stets ausführlich!

Aufgabe 1.1
Geben Sie für jede der folgenden Zahlen deren Ziffernschreibweisen im Dezimal-, Dual-, Oktal- und Hexadezimal-System an.

a) $(2748)_{10}$

b) $(1010011011)_2$

c) $(52056)_8$

d) $(\mathrm{D1})_{16}$

Aufgabe 1.2 (20 Punkte)
Verwenden Sie die Gesetze einer Booleschen Algebra, um

a) die *Resolutionsregel*

$$(x \cup y) \cap (\overline{x} \cup y) = y$$

zu beweisen.

b) den Booleschen Ausdruck

$$(\overline{(x \downarrow a)} \cap ((\overline{x} \uparrow \overline{b}) \cup c)) \cap ((x \cap (b \cup x)) \rightarrow (a \cap (b \cup (c \cap (d \cup 1)))))$$

so weit wie möglich zu vereinfachen.[1] Achten Sie darauf, dass ihr Lösungsweg nachvollziehbar ist (z. B. dadurch, dass Sie schrittweise vorgehen und jeweils angeben, an welchen Stellen im obigen Ausdruck Sie welche Gesetze verwendet haben).

Aufgabe 1.3

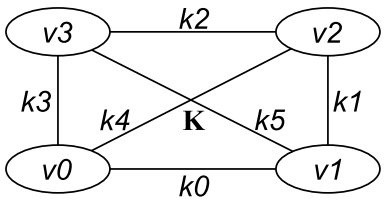
a) Erklären Sie den Aufbau der DNF einer Booleschen Funktion mit eigenen Worten.

b) Erklären Sie den Aufbau der KNF einer Booleschen Funktion zunächst zurückgeführt auf ihre DNF und dann ohne diesen „Umweg“.

[1] Zur Erinnerung: $x \rightarrow y = \overline{x} \cup y$, $x \downarrow y = \overline{x \cup y}$, $x \uparrow y = \overline{x \cap y}$.

Aufgabe 1.4 (20 Punkte)

Die folgende Abbildung zeigt eine Figur, die durch eine Knotenmenge $\{v0, v1, v2, v3\}$ und die zwischen diesen Knoten verlaufenden Kanten $k0, k1, \ldots, k5$ definiert ist. Dabei bilden zwei Kanten eine mit **K** bezeichnete Kreuzung. Eine *Teilfigur* entsteht durch Wegnahme einer beliebigen Teilmenge der eingezeichneten Kanten, wobei die Knoten bestehen bleiben.



a) Wie codieren Sie eine Teilfigur als Bitstring? Wieviele Teilfiguren gibt es?

b) Geben Sie eine Boolesche Funktion wahlweise in KNF oder DNF an, die dem Code einer Teilfigur genau dann eine 1 zuordnet, wenn die Teilfigur „kreuzungsfrei“ ist, d. h., **K** nicht enthält.

   Hinweis: Überlegen Sie zuvor anhand Ihrer Codierung, wieviele Maxterme die KNF der Funktion und wieviele Minterme ihre DNF enthält.

Aufgabe 1.5

Zeigen Sie: $\{\rightarrow\}$ ist funktional vollständig.

Lehr- und Forschungsgebiet Informatik V
Prof. G. Lakemeyer, Ph.D.

26.04.2000
G. Iwan, S. Sklorz

Rechnerstrukturen (SS 2000)

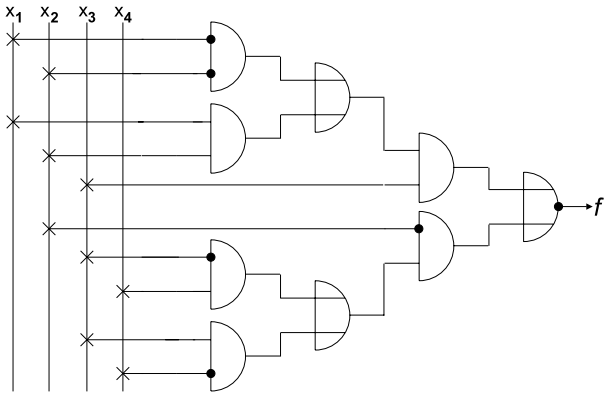
Aufgabenblatt 2

Abgabe: 01.05.–05.05.2000 in den Übungen

Bearbeiten Sie zur Vertiefung der Vorlesung nach Möglichkeit stets alle Aufgaben!
Mit Punkten ausgezeichnete Aufgaben können in den Übungen zur Korrektur abgegeben werden.
Bitte kommentieren Sie ihre Lösungen dann stets ausführlich!

Aufgabe 2.1

Gegeben sei folgendes Schaltnetz zur Realisierung einer Booleschen Funktion f:



a) Bestimmen Sie die DNF von f.

b) Minimieren Sie die DNF von f durch algebraische Vereinfachung unter Ausnutzung der Resolutionsregel $\phi\alpha\psi + \phi\overline{\alpha}\psi = \phi\psi$ und der Absorptionsregel $\phi\alpha\psi + \alpha = \alpha$ (Verschmelzungsgesetz), d. h., bestimmen Sie für die Funktion eine „einfachste“ disjunktive Form.

Beispiel Resolutionsregel: $x_1x_2x_3 + x_1x_2\overline{x}_3 + x_1x_2\overline{x}_4 = x_1x_2 + x_1x_2\overline{x}_4$.
Beispiel Absorptionsregel: $x_1x_2 + x_1x_2\overline{x}_4 = x_1x_2$.

Hinweis: Nutzen Sie stets alle Resolutionsmöglichkeiten.[1]

c) Zeichnen Sie ein Schaltnetz für Ihre Lösung des vorherigen Aufgabenteils.

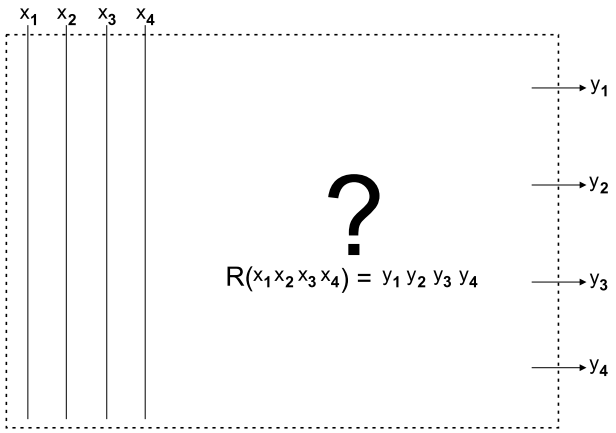
Aufgabe 2.2 (10 Punkte)

Sei $I = \{0, \ldots, 15\}$. Jedes $i \in I$ kann dargestellt werden durch eine vierstellige Dualzahl $d(i)$. Entwerfen Sie ein Schaltnetz für den Baustein eines Countdown-Zählers modulo 16, der durch die Schaltfunktion

$$R : B^4 \to B^4 \qquad \text{mit} \qquad R(d(i)) := d(\,(i-1) \bmod 16\,)$$

[1] Beispiel: $x_1x_2x_3 + x_1x_2\overline{x}_3 + x_1\overline{x}_2\overline{x}_3 = x_1x_2 + x_1\overline{x}_3$, denn wegen $\alpha = \alpha + \alpha$ gilt
$x_1x_2x_3 + x_1x_2\overline{x}_3 + x_1\overline{x}_2\overline{x}_3 = x_1x_2x_3 + x_1x_2\overline{x}_3 + x_1x_2\overline{x}_3 + x_1\overline{x}_2\overline{x}_3 = x_1x_2 + x_1\overline{x}_3$.

beschreibbar ist[2], d. h., realisieren Sie den „schwarzen Kasten“ der folgenden Abbildung:

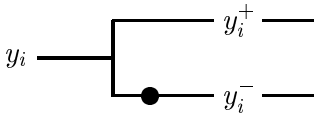


Aufgabe 2.3 (30 Punkte)

a) Für die Anzahl $G(d)$ der Und- und Oder-Gatter eines rekursiv konstruierten d-MUX gilt die Rekursionsbeziehung $G(2d) = (2^d + 1) \cdot G(d)$ und $G(1) = 3$. Beweisen Sie durch vollständige Induktion die geschlossene Form $G(d) = 3 \cdot (2^d - 1)$ für jede 2er-Potenz d.

b) Sei $S(d)$ die Anzahl der Stufen eines rekursiv konstruierten d-MUX, wobei nur Und- und Oder-Gatter gezählt werden. Stellen Sie eine Rekursionsbeziehung für S analog zu G auf. Finden Sie eine geschlossene Form von $S(d)$ für jede 2er-Potenz d und beweisen Sie diese durch vollständige Induktion.

c) Angenommen, ein MUX bekommt anstelle jedes Steuer-Inputs y_i direkt die beiden komplemtären Steuer-Inputs $y_i^+ = y_i$ und $y_i^- = \overline{y}_i$.[3]



$F(d)$ bezeichne den maximalen Fan-Out eines Steuer-Inputs y_i^+ oder y_i^- eines rekursiv konstruierten d-MUX. Stellen Sie eine Rekursionsbeziehung für F auf. Finden Sie eine geschlossene Form von $F(d)$ für jede 2er-Potenz d und beweisen Sie diese durch vollständige Induktion.

Aufgabe 2.4

Sei $f : B^4 \to B$ die Boolesche Funktion mit $f(x_1, x_2, x_3, x_4) = 1$ gdw.[4] $(\mathrm{x_1x_2x_3x_4})_2$ durch 5 oder 7 teilbar ist.[5] Skizzieren Sie eine Realisierung von f mittels eines 3-MUX.

Aufgabe 2.5

Entwerfen Sie ein Schaltnetz zum Multiplizieren von dreistelligen Dualzahlen $x_2x_1x_0$ mit zweistelligen Dualzahlen y_1y_0, das aus (höchstens) 2 Addiernetzen und 5 Multiplexern als Bausteinen besteht, wobei Sie auch an einigen Eingängen der Bausteine stets eine 0 anliegen lassen können. Die Größe der Bausteine (d. h. deren Anzahl der Inputs) sollen Sie selbst festlegen.

`http://www-i5.informatik.rwth-aachen.de/LuFG/Lehre/SS00/RS/`

[2] Beachte: $(-1) \bmod 16 = 15$

[3] Das macht die Aufgabe etwas einfacher.

[4] „gdw.“ = „genau dann, wenn“

[5] Beachte: 0 ist durch jede Zahl teilbar.

Lehr- und Forschungsgebiet Informatik V
Prof. G. Lakemeyer, Ph.D.

03.05.2000
G. Iwan, S. Sklorz

Rechnerstrukturen (SS 2000)

Aufgabenblatt 3

Abgabe: 08.05.–12.05.2000 in den Übungen

Bearbeiten Sie zur Vertiefung der Vorlesung nach Möglichkeit stets alle Aufgaben!
Mit Punkten ausgezeichnete Aufgaben können in den Übungen zur Korrektur abgegeben werden.
Bitte kommentieren Sie ihre Lösungen dann stets ausführlich!

Überhaupt lernet niemand etwas durch bloßes Anhören,
und wer sich in gewissen Dingen nicht selbst tätig bemühet,
weiß die Sachen nur oberflächlich und halb.

Goethe [1]

Aufgabe 3.1

Die einschlägigen Indizes einer Booleschen Funktion $f(x_1, x_2, x_3, x_4)$ seien 0, 4, 5, 6, 7, 8, 11, 12, 13 (vgl. Aufgabe 2.1).

a) Bestimmen Sie mit Hilfe eines Karnaugh-Diagramms alle Primimplikanten von f.

b) Geben Sie ein **Minimalpolynom** für f an, d. h. eine Darstellung von f als disjunktive Form mit minimalen Kosten.[2] Wie hoch sind die Kosten?

c) Ist das von Ihnen angegebene Minimalpolynom eindeutig? Begründen Sie Ihre Antwort kurz anhand des Karnaugh-Diagramms.

Aufgabe 3.2 (20 Punkte)

Die Boolesche Funktion $f(x_1, x_2, x_3, x_4)$ sei gegeben durch:

x_1	x_2	x_3	x_4	f
0	0	0	0	D
0	0	0	1	1
0	0	1	$*$	0
0	1	$*$	$*$	1
1	0	$*$	$*$	1
1	1	0	$*$	0
1	1	1	$*$	D

wobei $*$ sowohl für 0 als auch für 1 stehen kann.

a) Bestimmen Sie mit Hilfe eines Karnaugh-Diagramms alle Terme M, für die gilt:

$$M(x) = 1 \;\Rightarrow\; f(x) = 1 \text{ oder } f(x) = \mathrm{D} \quad \text{für alle } x \in B^4$$

und keine echte Verkürzung von M hat diese Eigenschaft.[3]

Hinweis: Solche Terme sind Primimplikanten einer Vervollständigung von f.

[1] `http://gutenberg.aol.de/eckerman/gesprche/gsp2084.htm`

[2] Zur Erinnerung: Die Kosten $K(d)$ einer disjunktiven Form d sind gleich der Anzahl der in d vorkommenden Disjunktions- und Konjunktions-Operatoren.

[3] Vgl. die Definition von Implikanten und Primimplikanten.

b) Bestimmen Sie alle disjunktiven Formen mit minimalen Kosten, die außer in den Don't-Care-Argumenten mit f übereinstimmen. Wie hoch sind die Kosten?

Hinweis: Solche disjunktiven Formen sind Minimalpolynome für geeignete Vervollständigungen von f.

Aufgabe 3.3 (20 Punkte)

Eine disjunktive Form ist bekanntlich eine Disjunktion von Konjunktionen von Variablen oder negierten Variablen. Eine **konjunktive Form** ist eine Konjunktion von Disjunktionen von Variablen oder negierten Variablen. Die Kosten $K(c)$ einer konjunktiven Form c sind analog zu denen einer disjunktiven Form definiert.[4] Für jede disjunktive Form d erhält man aus $\overline{d}$ durch Anwendung der deMorgan-Regeln und des Gesetzes $\overline{\overline{x}} = x$ eine konjunktive Form, die hier mit $\widetilde{d}$ bezeichnet werde.

a) Beweisen Sie: Für jede Boolesche Funktion f und jede disjunktive Form d von $\overline{f}$ mit minimalen Kosten ist $\widetilde{d}$ eine konjunktive Form von f mit minimalen Kosten.

b) Verwenden Sie diesen Zusammenhang, um eine konjunktive Form mit minimalen Kosten zu finden, die außer in den Don't-Care-Argumenten mit der Boolesche Funktion f aus Aufgabe 3.2 übereinstimmt. Wie hoch sind die Kosten?

Bemerkung: $\overline{\mathrm{D}} = \mathrm{D}$.

Aufgabe 3.4

Die einschlägigen Indizes einer Booleschen Funktion $f(x_1, x_2, x_3, x_4, x_5)$ seien 0, 1, 2, 3, 4, 5, 6, 7, 10, 16, 17, 20, 21, 24, 25, 26, 27, 28, 29, 30, 31.

a) Bestimmen Sie mit Hilfe von zwei Karnaugh-Diagrammen alle Primimplikanten der fünfstelligen Funktion f. Verwenden Sie dazu je ein Karnaugh-Diagramm für $x_5 = 0$ und $x_5 = 1$. Jedes Feld in einem der Diagramme besitzt dann außer seinen vier Nachbarn in diesem Diagramm noch einen „fünften Nachbarn" an der entsprechenden Feld-Stelle in dem anderen Diagramm.

b) Bestimmen Sie alle Minimalpolynome für f.

Hinweis: Ein Primimplikant von f heißt **Kernimplikant** von f, falls er mindestens eine Eingabe abdeckt, die von keinem anderen der Primimplikanten von f abgedeckt wird. Jedes Minimalpolynom für f enthält mindestens alle Kernimplikanten von f.

c) Wieviele Karnaugh-Diagramme braucht man bei diesem Verfahren für eine sechsstellige und wieviele für eine siebenstellige Funktion?

Bemerkung: Spätestens für sieben Stellen wird die Darstellung unsinnig.

Aufgabe 3.5 (10 Punkte)

Bestimmen Sie mit dem Verfahren von Quine und McCluskey alle Primimplikanten und Minimalpolynome der durch die DNF

$$\overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_1\overline{x}_2x_3x_4 + \overline{x}_1x_2x_3\overline{x}_4 + x_1x_2\overline{x}_3\overline{x}_4 + \overline{x}_1x_2x_3x_4 + \overline{x}_1\overline{x}_2x_3\overline{x}_4 + \overline{x}_1x_2\overline{x}_3\overline{x}_4 + \overline{x}_1\overline{x}_2\overline{x}_3x_4$$

gegebenen Booleschen Funktion.

`http://www-i5.informatik.rwth-aachen.de/LuFG/Lehre/SS00/RS/`

[4] D. h.: Die Kosten $K(c)$ einer konjunktiven Form c sind gleich der Anzahl der in c vorkommenden Konjunktions- und Disjunktions-Operatoren.

Lehr- und Forschungsgebiet Informatik V
Prof. G. Lakemeyer, Ph.D.

10.05.2000
G. Iwan, S. Sklorz

Rechnerstrukturen (SS 2000)

Aufgabenblatt 4

Abgabe: 15.05.–19.05.2000 in den Übungen

Bearbeiten Sie zur Vertiefung der Vorlesung nach Möglichkeit stets alle Aufgaben!
Mit Punkten ausgezeichnete Aufgaben können in den Übungen zur Korrektur abgegeben werden.
Bitte kommentieren Sie ihre Lösungen dann stets ausführlich!

Aufgabe 4.1 (20 Punkte)

In der Vorlesung wurde die *schaltungsabhängige Diagnose* vorgestellt. Bei der **schaltungsunabhängigen Diagnose** geht es darum, Testmengen zu finden, mit denen sich bestimmte Fehler in jeder Realisierung einer Booleschen Funktion finden lassen.

> Sei $f : B^n \to B$ eine Boolesche Funktion und $i \in \{1, \ldots, n\}$. f ist *abhängig vom i-ten Argument* gdw. es $a, b \in B^n$ gibt, sodass sich a und b nur an der i-ten Stelle unterscheiden und $f(a) \neq f(b)$ gilt. Ein solches Paar $\{a, b\}$ heißt *f-Testpaar für das i-te Argument.* (Daraus folgt: f ist abhängig vom i-ten Argument gdw. es ein f-Testpaar für das i-te Argument gibt.) Eine *f-Testmenge* ist eine Menge $T \subseteq B^n$, sodass es für jedes Argument, von dem f abhängig ist, ein f-Testpaar $\{a, b\} \subseteq T$ gibt. Eine f-Testmenge ist *minimal* gdw. keine ihrer echten Teilmengen eine f-Testmenge ist.

Durch Anlegen der Tests in einer f-Testmenge an eine beliebige Realisierung von f kann also festgestellt werden, ob diese Realisierung einen Fehler hat, der die Abhängigkeit von einem Argument, von dem f abhängig ist, zerstört.

> **Beispiel:** Bei $f(x_1, x_2, x_3) = \overline{x}_1 x_3 + x_2$ ergeben sich als f-Testpaare für das
>
> 1-te Argument: $\{(0,0,1),(1,0,1)\}$
> 2-te Argument: $\{(0,0,0),(0,1,0)\}$, $\{(1,0,0),(1,1,0)\}$, $\{(1,0,1),(1,1,1)\}$
> 3-te Argument: $\{(0,0,0),(0,0,1)\}$
>
> Die minimalen f-Testmengen sind: $T_1 = \{(0,0,0),(0,0,1),(0,1,0),(1,0,1)\}$
> $T_2 = \{(0,0,0),(0,0,1),(1,0,0),(1,0,1),(1,1,0)\}$
> $T_3 = \{(0,0,0),(0,0,1),(1,0,1),(1,1,1)\}$
>
> T_1 und T_3 haben zudem die minimale Anzahl an Elementen.

Sei $f : B^4 \to B$ die Boolesche Funktion mit den einschlägigen Indizes 2, 3, 6, 7, 10, 14.

a) Von welchen Argumenten ist f abhängig? Bestimmen Sie alle f-Testpaare für die Argumente.

b) Finden Sie eine f-Testmenge mit der minimalen Anzahl an Elementen.[1]

Aufgabe 4.2

Sei $f : B^4 \to B$ die Boolesche Funktion aus Aufgabe 4.1, wo ja schaltungsunabhängige Diagnose behandelt wird. In dieser Aufgabe hier geht es um schaltungsabhängige Diagnose.

a) Bestimmen Sie das (in diesem Fall eindeutige) Minimalpolynom von f.

b) Zeichnen Sie wahlweise den zugehörigen DAG oder das zugehörige Schaltnetz mit „reinen" Und-, Oder-, Nicht-Gattern (d. h.: keine Inverter direkt an Ein- oder Ausgängen von Gattern) und nummerieren Sie die „Drähte".

[1] f-Testmengen mit der minimalen Anzahl an Elementen sind auch minimale f-Testmengen (Warum?).

c) $f_i : B^4 \to B$ bezeichne die Boolesche Funktion, die dem DAG oder Schaltnetz entspricht, wenn der Draht mit Nummer i gerissen ist (0-Verklemmung, „stuck-at 0“). Bestimmen Sie die (möglichst kompakten) Darstellungen der f_i und welche der f_i identisch sind. Erstellen Sie die Ausfallmatrix und die Fehlermatrix.

d) Welche minimalen Testmengen[2] gibt es?

Allgemeiner Hinweis:
In der Vorlesung interessieren wir uns weniger für die technologischen Einzelheiten von Schaltwerken als vielmehr für ihr Verhalten aus logischer Sicht. Im folgenden gehen wir daher (vereinfachend) von der Annahme aus, dass die angenommenen Taktzeiten[3] für die betrachteten Schaltwerke stets länger sind als die Schaltzeiten[4] der zugehörigen Schaltnetze. Damit entsprechen die (Schalt-)Schritte genau den Takten.

Aufgabe 4.3
Demonstrieren Sie die Arbeitsweise eines von-Neumann-Addierwerks und eines Parallel-Addierwerks sowie die Vorgänge in einem Serien-Addierwerk für die Aufgabe $31 + 1$, indem Sie die Inhalte vorhandener Delays vor und nach jedem der jeweils benötigten Schritte protokollieren.

Aufgabe 4.4 (30 Punkte)
Angenommen, das Laden von zwei zu addierende Summanden in Akkumulator und Puffer eines Addierwerks ist bereits geschehen. (Beachten Sie, dass dabei auch das Übertrags-Delay U (mit 0) und das Status-Delay S (mit 1) geladen werden.) Aus der Vorlesung ist bereits bekannt:

- Ein n-Bit-Parallel-Addierwerk benötigt einen Schritt zur Addition der Summanden.
- Ein n-Bit-Serien-Addierwerk benötigt n Schritte zur Addition der Summanden.
- Ein n-Bit-von-Neumann-Addierwerk benötigt durchschnittlich (im „average case“) $\log_2 n$ Schritte zur Addition der Summanden.

a) Zeigen Sie: Ein n-Bit-von-Neumann-Addierwerk benötigt im schlechtesten Fall (im „worst case“) n Schritte zur Addition der Summanden.

b) Wieviele Schritte benötigt ein n-Bit-von-Neumann-Addierwerk im besten Fall (im „best case“) zur Addition der Summanden? Formulieren Sie eine allgemeine Bedingung, die charakterisiert, wann für die Summanden $A_{n-1} \ldots A_0$ und $P_{n-1} \ldots P_0$ dieser beste Fall eintritt.

c) Vergleichen Sie Parallel-, Serien- und von-Neumann-Addierwerk unter den Aspekten: Hardware-Aufwand, Anzahl der benötigten Schritte sowie benötigte Schaltzeiten. Formulieren Sie ein kurzes Fazit, dass Ihre Resultate im Hinblick auf die Größe der Schaltwerke und die von ihnen für eine Berechnung benötigte Zeit zusammenführt.

Aufgabe 4.5
Wieviele Schritte braucht ein n-Bit-von-Neumann-Addierwerk, um zu einem Akkumulator, der am Anfang eine 0 (in Binärdarstellung) enthält, $(2^n - 1)$-mal eine 1 (in Binärdarstellung) zu addieren, wenn auch der Puffer am Anfang eine 0 (in Binärdarstellung) enthält?

Bemerkung: Vielleicht können Sie die Gleichung $\sum_{\ell=0}^{n-1} \ell \cdot 2^\ell = (n-2)2^n + 2$ gebrauchen.[5]

`http://www-i5.informatik.rwth-aachen.de/LuFG/Lehre/SS00/RS/`

[2] Zur Erinnerung (weil Testmengen bei schaltungsabhängiger Diagnose anders definiert sind als bei schaltungsunabhängiger Diagnose): Eine Testmenge liegt vor, wenn sich durch Anlegen der Tests aus der Testmenge erkennen lässt, ob ein Drahtrissfehler vorliegt oder nicht. (Drahtrisse, die keinen Fehler verursachen, brauchen/können nicht betrachtet zu werden.)

[3] Zur Erinnerung: Die Taktzeit ist die Zeit zwischen zwei Taktimpulsen eines Rechners.

[4] Zur Erinnerung: Die Schaltzeit eines (asynchronen) Schaltnetzes ist die Zeit, die nach dem Anlegen der Inputs vergeht, bis sich ein stabiler Zustand der Outputs einstellt.

[5] Als Übung können Sie ja mal die allgemeine Form diese Gleichung für $x \neq 1$ beweisen:

$$\sum_{\ell=0}^{n-1} \ell \cdot x^\ell = [\,(x-1)(n-1)x^n - x^n + x\,] \,/\, (x-1)^2 = [\,(xn - n - x)x^n + x\,] \,/\, (x-1)^2.$$

Das geht sowohl mit vollständiger Induktion als auch „direkt“ (z. B. durch geschicktes Umsortieren/Anordnen der Summanden). Beachten Sie, dass $\sum_{\ell=0}^{n-1} x^\ell = (x^n - 1)/(x-1)$ für $x \neq 1$ gilt (Beweis? Einfach!).

Lehr- und Forschungsgebiet Informatik V
Prof. G. Lakemeyer, Ph.D.

17.05.2000
G. Iwan, S. Sklorz

Rechnerstrukturen (SS 2000)

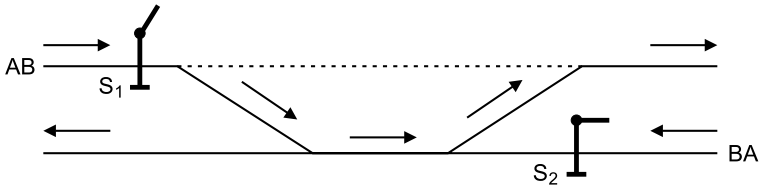
Aufgabenblatt 5

Abgabe: 22.05.–26.05.2000 in den Übungen

Bearbeiten Sie zur Vertiefung der Vorlesung nach Möglichkeit stets alle Aufgaben!
Mit Punkten ausgezeichnete Aufgaben können in den Übungen zur Korrektur abgegeben werden.
Bitte kommentieren Sie ihre Lösungen dann stets ausführlich!

Aufgabe 5.1 (30 Punkte)

Zwischen zwei Orten A und B gibt es eine Schienenverbindung mit zwei Gleiswegen AB und BA. Dabei nutzen von A nach B fahrende Züge den Gleisweg AB, während von B kommende Züge den Gleisweg BA befahren. Bauarbeiten auf einem Teilabschnitt von AB führen jedoch zu einer Ausnahmesituation, die in der folgenden Abbildung illustriert ist:



Danach müssen AB befahrende Züge für einen kurzen Teilabschnitt auf den Gleisweg BA ausweichen. Der Zugverkehr auf diesem Teilabschnitt soll durch zwei einfache Signalanlagen S_1 und S_2 geregelt werden, die entweder „freie Fahrt“ oder „unbedingter Halt“ signalisieren können. Weiterhin sollte die Regelung derart erfolgen, dass die beiden Signalanlagen in ihren Signalen als Paar (S_1, S_2) aufgefasst stets in einem Zyklus die Zustände $(0,0)$, $(1,0)$, $(0,0)$, $(0,1)$ und wieder $(0,0)$ durchlaufen, wobei „freie Fahrt“ mit 1 und „unbedingter Halt“ mit 0 codiert ist.

a) Entwerfen Sie ein optimiertes (= minimiertes) (Steuer-)Schaltwerk, das die beiden Signalanlagen in der beschriebenen Weise und unter Verwendung der angegebenen Codierung steuert.

b) Demonstrieren Sie die Arbeitsweise Ihres Steuerwerks, indem Sie die Inhalte aller vorhandener Delays vor und nach jedem Takt für einen kompletten Zyklus protokollieren.

c) Erweitern Sie Ihr Steuerwerk um eine Steuerleitung Z, deren Wert darüber bestimmt, ob der regelnde Betrieb stattfindet oder ob die beide Signalanlagen ständig auf „unbedingter Halt“ gesetzt sind.

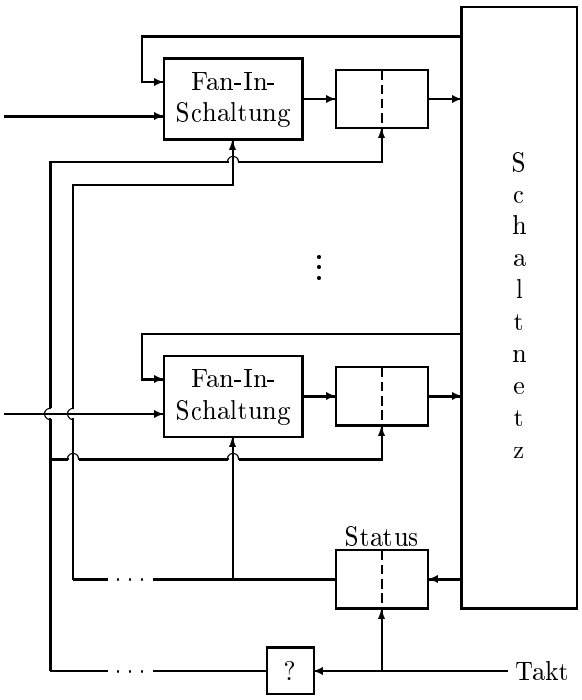
   Hinweis: Dies sollte sich mit zwei zusätzlichen Gattern realisieren lassen.

Aufgabe 5.2

Gegeben seien die folgenden idealisierten Annahmen:

- Jedes Gatter benötigt 10 psec um zu schalten.
- Die Schaltzeit der Fan-In-Schaltung in der nachfolgenden Abbildung ist 30 psec.
- Die Schaltzeit des Schaltnetzes in der nachfolgenden Abbildung ist 280 psec.
- Die Setzphase der (z. B. durch D-Flip-Flops realisierten) Delay-Bausteine (□□) sind die ersten 50 psec jedes hohen Taktsignals, die Arbeitsphase der Rest des Taktzyklus.
- Der Takt ist symmetrisch mit einer Taktfrequenz von 500 MHz.
  D. h.: Das Taktsignal ist immer abwechselnd jeweils 1000 psec hoch und niedrig.

Das nebenstehende Schaltwerk könnte also wesentlich schneller schalten als es der Takt zulässt, nämlich maximal fünfmal so schnell.



a) Entwerfen Sie eine Schaltung für den als ? eingezeichneten Baustein, die an ihrem Ausgang einen symmetrischen „Sekundär-Takt“ mit einer Taktfrequenz von $5 \cdot 500\ \text{MHz} = 2500\ \text{MHz}$ liefert, der mit dem Eingangs-Takt so synchronisiert ist, dass, wenn der Takt am Eingang eine aufsteigende Flanke hat, auch der Takt am Ausgang eine aufsteigende Flanke hat (was wichtig ist für die Synchronisation mit dem Status, der ja die Fan-In-Schaltung steuert).

   Nehmen Sie an, dass Ihnen x psec-Bausteine für $x = 10,\ 20,\ 30,\ 40,\ \ldots$ zur Verfügung stehen, die ein Signal um x psec verzögern. Verwenden Sie außer diesen Bausteinen nur Inverter, UND- und ODER-Gatter.

b) Warum kann das Schaltwerk nicht maximal zwei- oder drei- oder sechsmal, sondern maximal fünfmal schneller schalten, wenn die Synchronisation der beiden Takte zu beachten ist?

Aufgabe 5.3 (20 Punkte)

Die Funktionsweise eines SR-Latches kann beschrieben werden durch:

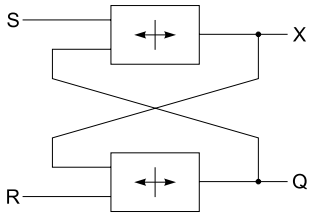
$$(Q_{\text{neu}}, \overline{Q}_{\text{neu}}) = \begin{cases} (Q_{\text{alt}}, \overline{Q}_{\text{alt}}) & \text{falls } (S,R) = (0,0) \text{ und } Q_{\text{alt}} \neq \overline{Q}_{\text{alt}} \\ (1,0) & \text{falls } (S,R) = (1,0) \\ (0,1) & \text{falls } (S,R) = (0,1) \\ (\text{D},\text{D}) & \text{sonst}^1 \end{cases}$$

a) Entwerfen Sie eine Schaltung, die diese Funktionalität besitzt[2] und ausschließlich NAND-Gatter enthält.[3]

b) Falls Ihre Schaltung mehr als vier Gatter hat:
   Finden Sie eine Schaltung, die höchstens vier NAND-Gatter enthält und die Funktionalität hat. Unterscheiden sich die beiden Schaltungen hinsichtlich ihres Verhaltens? Wenn ja, wie? [4]

Aufgabe 5.4

Betrachten Sie die nebenstehende Schaltung in Analogie zum NOR-Latch als ein „XOR-Latch“ und analysieren Sie die Arbeitsweise der Schaltung unter dieser Betrachtung. Macht es Ihrer Ansicht nach einen Sinn oder keinen Sinn, ein Latch auf diese Weise zu realisieren?



---

http://www-i5.informatik.rwth-aachen.de/LuFG/Lehre/SS00/RS/

[1] Zur Erinnerung: D steht für „Don't-Care“.

[2] Bis auf die Don't-Care-Argumente natürlich.

[3] Wenn Sie anders nicht weiter kommen, können Sie bedenken, dass {NAND} funktional vollständig ist.

[4] Da beide Schaltungen die „richtige“ Funktionalität besitzen, kann sich ihr Verhalten natürlich nur bei den Don't-Care-Argumenten und bei (zeitlichen) Veränderungen der Eingabewerte unterscheiden.

Rechnerstrukturen (SS 2000)

Aufgabenblatt 6

Abgabe: 29.05.–02.06.2000 in den Übungen

Bearbeiten Sie zur Vertiefung der Vorlesung nach Möglichkeit stets alle Aufgaben!
Mit Punkten ausgezeichnete Aufgaben können in den Übungen zur Korrektur abgegeben werden.
Bitte kommentieren Sie ihre Lösungen dann stets ausführlich!

Falls Sie die ersten beiden Aufgaben nicht für allgemeines b lösen können, versuchen Sie, die betreffenden Teilaufgaben für $b = 2$ zu bearbeiten.

Aufgabe 6.1 (70 Punkte)

Seien n und b natürliche Zahlen mit $n \geq 1$ und $b \geq 2$. Für $z = (z_{n-1}, \ldots, z_0)_b$ ist

$$K_{b-1}^{b,n}(z) = ((b-1) - z_{n-1}, \ldots, (b-1) - z_0)_b$$

das n-stellige $(b-1)$-Komplement von z zur Basis b und

$$K_b^{b,n}(z) = [K_{b-1}^{b,n}(z) + 1] \bmod b^n$$

das n-stellige b-Komplement von z zur Basis b.[1]

Seien x, y natürliche Zahlen kleiner als b^n. Beweisen Sie die folgenden Aussagen:

a) $K_{b-1}^{b,n}(x) = (b^n - 1) - x$ (und damit $K_{b-1}^{b,n}(0) = b^n - 1$)

b) $K_b^{b,n}(x) = b^n - x$ für $x \neq 0$ und $K_b^{b,n}(0) = 0$

Bemerkung: Auch $K_{b-1}^{b,n}(x)$ und $K_b^{b,n}(x)$ sind natürliche Zahlen kleiner als b^n.

c) $K_{b-1}^{b,n}(K_{b-1}^{b,n}(x)) = x$ und $K_b^{b,n}(K_b^{b,n}(x)) = x$

Falls $x + y < b^n$:

d) $K_{b-1}^{b,n}(x + y) = [K_{b-1}^{b,n}(x) + K_{b-1}^{b,n}(y) + 1] \bmod b^n$

e) $K_b^{b,n}(x + y) = [K_b^{b,n}(x) + K_b^{b,n}(y)] \bmod b^n$

Falls $x - y \geq 0$:

f) $x - y = [x + K_{b-1}^{b,n}(y) + 1] \bmod b^n$

g) $x - y = [x + K_b^{b,n}(y)] \bmod b^n$

Hinweis: Mit Hilfe von a) bzw. b) lassen sich die anderen Teilaufgaben leicht lösen.

Aufgabe 6.2 (50 Punkte)

Seien n und b natürliche Zahlen mit $n \geq 1$ und $b \geq 2$, wobei b gerade sei. In dieser Aufgabe sollen positive und negative ganze Zahlen (und Null) im b-adischen Zahlensystem dargestellt werden, wobei n Stellen zur Verfügung stehen.

$z = (z_{n-1}, \ldots, z_0)_b$ stelle die Zahl z selbst dar gdw. $z_{n-1} \in \{0, \ldots, b/2 - 1\}$.[2]
$z = (z_{n-1}, \ldots, z_0)_b$ stelle die Zahl $-K_b^{b,n}(z)$ dar gdw. $z_{n-1} \in \{b/2, \ldots, b - 1\}$.[3]
(Beachte: z und $K_b^{b,n}(z)$ sind beide nicht-negativ (d. h. positiv oder Null)!)

---

[1] In der Vorlesung wurden (bei gegebenem n) $K_1 = K_1^{2,n}$ und $K_2 = K_2^{2,n}$ eingeführt.
[2] Z. B.: $z = (3, 2, 1, 0)_8 = 1672$ stellt die Zahl 1672 dar.
[3] Z. B.: $z = (4, 2, 1, 0)_8 = 2184$ stellt die Zahl $-K_8^{8,n}(z) = -(3, 5, 7, 0)_8 = -1912$ $(= 8^4 - z)$ dar.

a) Sei w die durch z dargestellte Zahl. Zeigen Sie:
$w \geq 0$ gdw. $z < \frac{1}{2}b^n$ gdw. $w = z$.
$w < 0$ gdw. $z \geq \frac{1}{2}b^n$ gdw. $w = -K_b^{b,n}(z)$.
Wenn $w < 0$, dann $w = -K_b^{b,n}(z) = z - b^n$ und $z = K_b^{b,n}(-w) = w + b^n$.

b) Was ist die größte so darstellbare ganze Zahl $Z_+^{b,n}$?
Was ist die kleinste so darstellbare ganze Zahl $Z_-^{b,n}$?

Seien u und v ganze Zahlen mit $u, v \in \{Z_-^{b,n}, \ldots, Z_+^{b,n}\}$. Man spricht von einem **Überlauf** (*Overflow*) bei einer Rechenoperation $u \pm v$ gdw. $u \pm v \notin \{Z_-^{b,n}, \ldots, Z_+^{b,n}\}$.

D. h.: $u \pm v$ ist darstellbar gdw. bei $u \pm v$ kein Überlauf auftritt.

Sei u durch x und v durch y dargestellt. $z_\oplus = [x + y] \bmod b^n$. $z_\ominus = [x + K_b^{b,n}(y)] \bmod b^n$.

c) Beweisen Sie: $u + v$ wird durch $z_\oplus$ dargestellt, wenn kein Überlauf auftritt.

d) Beweisen Sie: $u - v$ wird durch $z_\ominus$ dargestellt, wenn kein Überlauf auftritt.

D. h.: Die Addition ganzer Zahlen kann durch die Addition der Summanden-Darstellungen und die Subtraktion ganzer Zahlen durch die Addition der Minuend-Darstellung und des Komplements der Subtrahend-Darstellung realisiert werden.

e) Finden (und beweisen) Sie möglichst einfach zu überprüfende Bedingungen an x, y, $z_\oplus$, $z_\ominus$, die charakterisieren, wann ein Überlauf bei $u + v$ bzw. $u - v$ eintritt.

Hinweis: Evtl. sind Fallunterscheidungen der Form $w \asymp 0$ hilfreich mit $w \in \{u, v, u + v, u - v\}$ und $\asymp \in \{<, >, =, \leq, \geq\}$.

Aufgabe 6.3 (20 Punkte)

Im folgenden sei die Wortlänge gleich 8. (D. h.: Es wird mit Bytes gearbeitet.)

a)
i) Wie ist die Darstellung von -50 im Zweier-Komplement?
ii) Wie ist die Darstellung von -62 im Einer-Komplement?
iii) Wie ist die Darstellung von $+44$ im Einer-Komplement?
iv) Berechnen Sie $76 - 44$ im Einer-Komplement?
v) Berechnen Sie $116 - 29$ im Zweier-Komplement?
vi) Welche Zahl wird durch 183 im Zweier-Komplement dargestellt?
vii) Welche Zahl wird durch 186 im Einer-Komplement dargestellt?
viii) Berechnen Sie $44 - 76$ im Zweier-Komplement?
ix) Berechnen Sie $-163 + 76$ im Einer-Komplement?
x) Berechnen Sie $-35 - (-100)$ im Zweier-Komplement?
xi) Berechnen Sie $-18 - (-100)$ im Einer-Komplement?
xii) Wie ist die Darstellung von -88 im Einer-Komplement?
xiii) Wie ist die Darstellung von -45 im Zweier-Komplement?
xiv) Berechnen Sie $-47 - 16$ im Zweier-Komplement?

b) Interpretieren Sie die Beträge der Ergebnisse des vorigen Aufgabenteils der Reihe nach als ASCII-Codierungen mit ungerader Parität. Korrigieren Sie eventuell falsche Paritätsbits.

Aufgabe 6.4 (20 Punkte)

a) Stellen Sie die folgenden Werte als Gleitkomma-Zahlen im Dualsystem in normalisierter Form mit Exponenten-Basis 2 dar, wobei die Mantisse auf 7 Nachkommstellen gerundet werde.

i) $(10101010,10)_2 \cdot 2^{-10}$ ii) $(\mathrm{EFF},\mathrm{EFF})_{16}$ iii) $(0,0576171875)_{10}$
iv) $(1,3)_{10}$ v) $(7,72)_{10} \cdot 10^{-5}/(705)_8$ vi) $\sqrt{(2)_{10}}$

b) Berechnen Sie alle Summen und Produkte von jeweils zwei der ersten vier Gleitkomma-Zahlen.

Um die Unterschiede zu verdeutlichen, geben Sie in beiden Aufgabenteilen die „eigentlichen Werte“ und die Werte der Gleitkomma-Zahlen auch als Dezimalzahl an (evtl. geeignet gerundet).

Lehr- und Forschungsgebiet Informatik V
Prof. G. Lakemeyer, Ph.D.

31.05.2000
G. Iwan, S. Sklorz

Rechnerstrukturen (SS 2000)

Aufgabenblatt 7

Abgabe: 05.06.–09.06.2000 in den Übungen

Bearbeiten Sie zur Vertiefung der Vorlesung nach Möglichkeit stets alle Aufgaben!
Mit Punkten ausgezeichnete Aufgaben können in den Übungen zur Korrektur abgegeben werden.
Bitte kommentieren Sie ihre Lösungen dann stets ausführlich!

Aufgabe 7.1 (30 Punkte)

Gegeben sei ein Schaltnetz mit vier Eingängen x_1, x_2, x_3, x_4 und den Ausgängen p_1, p_2, p_3, p_4, das eine Schaltfunktion $F : B^4 \to B^4$ realisiert, wobei $F(x_1, x_2, x_3, x_4) = (z_1, z_2, z_3, z_4)$ gilt gdw. $(z_1 z_2 z_3 z_4)_2 = (x_1 x_2)_2 \times (x_3 x_4)_2$ ist (*Multiplikation zweier zweistelliger Dualzahlen*). Sei $F = (f_1, f_2, f_3, f_4)$, d. h., jedes f_i ist eine Boolesche $f_i : B^4 \to B$.

a) Realisieren Sie diese Schaltfunktion mit einem PLA, indem Sie die Minterme für jede Funktion f_i bestimmen und in einer geeigneten Weise in eine PLA-Matrix übertragen.

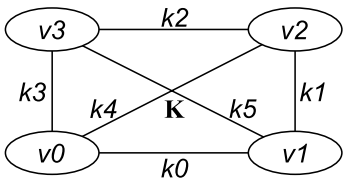
b) Bestimmen Sie je ein Minimalpolynom für jede Funktion f_i. Hätte Ihnen diese Optimierung beim Aufbau des PLA's zu einer Einsparung von Spalten verhelfen können? Falls ja, warum hätten Sie wieviele Spalten sparen können? Falls nein, warum hätten Sie nichts sparen können?

c) Gibt es ein PLA mit 7 Spalten, das die obige Schaltfunktion realisiert? Falls nein, warum nicht? Falls ja, finden Sie eine zugehörige PLA-Matrix?

Hinweis: Die letzten beiden Aufgabenteile lassen sich sehr gut mittels Karnaugh-Diagrammen bearbeiten, deren Verwendung daher ausdrücklich empfohlen wird.

Aufgabe 7.2 (30 Punkte)

Die nebenstehende Abbildung zeigt noch einmal die Figur aus Aufgabe 1.4. Wir erinnern uns, dass eine Teilfigur durch Wegnahme einer Teilmenge der eingezeichneten Kanten unter Beibehaltung der Knoten entsteht, und dass jede Teilfigur mit 6 Bits codiert werden kann.



Gesucht wird nun eine Boolesche Funktion $f : B^6 \to B$, die dem Code einer Teilfigur genau dann eine 1 zuordnet, wenn die Teilfigur genau 4 Kanten hat, von denen 3 Kanten über 3 Knoten zu einem Dreieck verbunden sind.

a) Realisieren Sie die Funktion f mit einem PLA, indem Sie die Minterme der Funktion in einer geeigneten Weise in eine PLA-Matrix übertragen.

b) Reduzieren Sie die Zeilenanzahl in Ihrem PLA um mindestens zwei Zeilen, indem Sie eine hierzu geeignete Faltung der Und-Ebene vornehmen. Geben Sie ihre Lösung in einer punkt-orientierten PLA-Darstellung an.

c) Wieso bringt die Minimierung der Minterm-Darstellung einer Booleschen Funktion i. allg. einen Vorteil bei der Faltung? Wieso hier nicht?

Aufgabe 7.3 (20 Punkte)

Das *River-Crossing-Problem* bzw. *Wolf-Kohlkopf-Ziege-Problem* (WKZ) besteht darin, dass ein Bauer drei „Objekte", nämlich einen Wolf, einen Kohlkopf und eine Ziege, mit seinem Boot über einen Fluss transportieren möchte, ohne dabei ein Objekt aufgrund der folgenden Randbedingungen zu verlieren: 1.) Das Boot bietet nur dem Bauern und einem Objekt genügend Platz, 2.) Wölfe fressen Ziegen, 3.) Ziegen fressen Kohlköpfe, und 4.) Der Bauer kann 2.) und 3.) durch seine Anwesenheit verhindern.

a) Wenden Sie das Konzept der Mikroprogrammierung an, um das WKZ-Problem zu lösen, indem Sie ein PLA programmieren und in ein (Steuer-)Schaltwerk mit drei Ausgängen O_0, O_1, O_2 integrieren, dass zyklisch eine Signalfolge berechnet, die unter der folgenden Codierung einer (korrekten) Lösung für das WKZ entspricht:

O_0	Transportrichtung
0	vom Ausgangs- zum Zielufer
1	vom Ziel- zum Ausgangsufer

O_1	O_2	Objekt
0	0	„keins"
0	1	Kohlkopf
1	0	Ziege
1	1	Wolf

Skizzieren Sie das gesamte Steuerwerk. Geben Sie das PLA als PLA-Matrix an.

b) Erläutern Sie die Funktionsweise Ihrer Steuerschaltung, indem Sie die Inhalte aller vorhandenen Delays für einen kompletten Zyklus protokollieren und interpretieren.

Aufgabe 7.4 (20 Punkte)

Eine hier nicht näher bezeichnete Behörde fängt via E-Mail versendete Nachrichten ab, um sie auf bestimmte Schlüsselwörter zu durchsuchen. Dazu werden die Nachrichten in einen 8-Bit ASCII-Code ungerader Parität übertragen und jeweils zeichenweise auf eine 8-Bit Datenleitung $(D_0, \ldots, D_7)$ gelegt, die in jedem Takt einen Buchstaben an eine Reihe von (Steuer-)Schaltwerken weiterleitet. So ist beispielsweise eines der Steuerwerke dafür zuständig, über eine Alarmglocke ein anhaltendes Läuten auszulösen, sobald eine Nachricht das Wort `bomb` enthält.

a) Realisieren Sie das „bomb"-Steuerwerk mit einem PLA im Schaltnetzteil. Geben Sie hierzu das PLA als PLA-Matrix an.

b) Erläutern Sie die Funktionsweise Ihrer Steuerschaltung, indem Sie den Nutzen der von Ihnen gewählten Delays und deren Inhalt im laufenden Betrieb erörtern.

Lehr- und Forschungsgebiet Informatik V
Prof. G. Lakemeyer, Ph.D.

07.06.2000
G. Iwan, S. Sklorz

Rechnerstrukturen (SS 2000)

Aufgabenblatt 8

Abgabe: 19.06.–23.06.2000 in den Übungen

Bearbeiten Sie zur Vertiefung der Vorlesung nach Möglichkeit stets alle Aufgaben!
Mit Punkten ausgezeichnete Aufgaben können in den Übungen zur Korrektur abgegeben werden.
Bitte kommentieren Sie ihre Lösungen dann stets ausführlich!

Aufgabe 8.1 (100 Punkte)

1946 wurde das von-Neumann-Rechnermodell vorgestellt, das die Rechnerarchitektur bis heute maßgeblich beeinflusst. Arbeiten Sie die grundlegenden Organisationsprinzipien und Besonderheiten dieses Modells anhand des achten Kapitels (im Lehrbuch von Oberschelp/Vossen) heraus, indem Sie folgende Fragen möglichst prägnant und in eigenen Worten beantworten.

a) Mit dem von-Neumann-Rechnermodell wurde erstmalig das Konzept für einen echten „general-purpose Computer“ vorgeschlagen. Was ist darunter zu verstehen?

b) „Programme sind auch nur Daten“ ist eine grundlegende und eng mit dem von-Neumann-Rechnermodell verbundene Sichtweise. Was ist darunter zu verstehen?

c) Das von-Neumann-Rechnermodell setzt sich aus drei Hauptbestandteilen zusammen. Welche Bestandteile sind dies und welchem Zweck dienen sie?

d) Im von-Neumann-Rechnermodell ist der Datenprozessor ein Bestandteil der CPU. Welche Aufgaben werden von welchen Komponenten dieses Prozessors erfüllt?

e) Im von-Neumann-Rechnermodell ist der Befehlsprozessor ein Bestandteil der CPU. Welche Aufgaben werden von welchen Komponenten dieses Prozessors erfüllt?

f) Das von-Neumann-Rechnermodell unterscheidet zwischen Daten- und Adressbus. Warum macht das Sinn? Es ergeben sich auch Zusammenhänge zwischen der Größe (in Bits) des MAR, des MBR, des Speichers, einer Speicherzelle sowie der Speicherzellenanzahl. Welche?

g) Die Arbeitsweise eines von-Neumann-Rechners wird durch die Bezeichnung SISD allgemein charakterisiert. Welches Prinzip verbirgt sich hinter dieser Abkürzung?

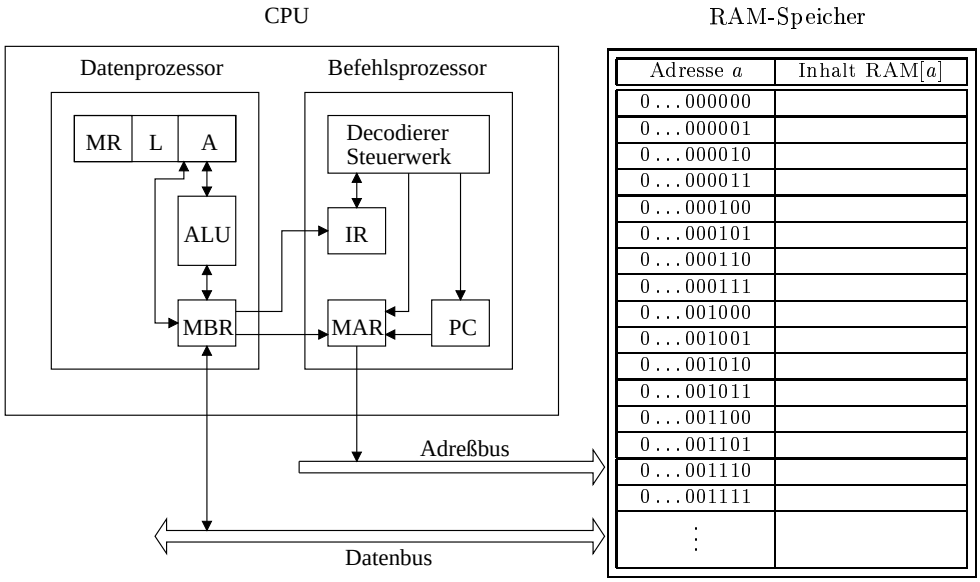
h) Bahnbrechend neu am von-Neumann-Rechnermodell war das Konzept einer quasi universellen Programmierbarkeit. Erörtern Sie in diesem Zusammenhang die Begriffe Maschinencode, Assemblersprachen sowie Ein- und Mehr-Adress-Befehle.

i) Charakteristisch für das von-Neumann-Rechnermodell ist ein Zwei-Phasen-Konzept der Befehlsverarbeitung. Welches Problem wird damit auf welche Weise gelöst?

j) Die Architektur eines klassischen von-Neumann-Rechners führte schon bald zu einem gewichtigen Problem, dem von-Neumannschen „Flaschenhals“. Was ist darunter zu verstehen und wie versuchte man später dieses Problem zunächst zu umgehen?

Aufgabe 8.2 (50 Punkte)

Folgende Abbildung zeigt noch einmal die Struktur einer CPU und ihre Verbindung zum RAM-Speicher. Für eine Speicheradresse a bezeichne RAM[a] die zugehörige Speicherzelle bzw. deren Inhalt.



Zur Vereinfachung werden MR und L im folgenden vernachlässigt: statt [MR][L][A] wird nur ein Register [Akku] verwendet. Nehmen Sie an, dass diesem Rechner die in der folgenden Tabelle durch ihre Operationscodes identifizierten Befehle/Operationen zur Verfügung stehen:

Operationscode	Wirkung(en) der Operation	
0		$PC \leftarrow PC + 1$
1	$RAM[RAM[PC + 1]] \leftarrow MBR$	$PC \leftarrow PC + 2$
2	$MBR \leftarrow RAM[RAM[PC + 1]]$	$PC \leftarrow PC + 2$
3	$MBR \leftarrow RAM[PC + 1]$	$PC \leftarrow PC + 2$
4	$MBR \leftarrow Akku$	$PC \leftarrow PC + 1$
5	$Akku \leftarrow MBR$	$PC \leftarrow PC + 1$
6	$Akku \leftarrow Akku + MBR$	$PC \leftarrow PC + 1$
7	$Akku \leftarrow Akku - MBR$	$PC \leftarrow PC + 1$
8	$Akku \leftarrow Akku * MBR$	$PC \leftarrow PC + 1$
9	$Akku \leftarrow Akku \div MBR$	$PC \leftarrow PC + 1$
10		$PC \leftarrow RAM[PC + 1]$
11	Falls $Akku = 0$:	$PC \leftarrow RAM[PC + 1]$
	Falls $Akku \neq 0$:	$PC \leftarrow PC + 2$

Der Rechner kenne nur ganze Zahlen (keine Gleikomma-Zahlen). Dementsprechend steht $\div$ für die ganzzahlige Division (z. B. $26 \div 7 = 3$).

a) Beschreiben Sie, wozu die Operationen (mit dem Operationscode) 1, 2, 3, 10, 11 dienen.

Im folgenden „Speicher-Auszug“ sind neben die Dualzahlen (mit denen „echte Rechner“ natürlich arbeiten) auch die entsprechenden Dezimalzahlen (klein und in Klammern) geschrieben.

In Ihren Lösungen brauchen Sie aber nur Dezimalzahlen anzugeben.

b) Die CPU führe hier bei jeder Befehlsverarbeitung die folgenden „Zuweisungen“ (in dieser Reihenfolge) durch:

MAR ← PC
IR ← RAM[MAR]
Wirkung(en) der Operation IR
(PC-Änderung zuletzt)

Protokollieren Sie den Inhalt der Register Akku, MBR, MAR, IR und PC nach jeder Befehlsverarbeitung bis PC den Wert 25 enthält, wobei zu Anfang alle Register 0 enthalten und der Speicher wie nebenstehend gefüllt ist.

c) Betrachten Sie nun den nebenstehenden Speicherinhalt von Adresse 0 bis Adresse 21 (einschließlich) und interpretieren Sie ihn als ein Programm. Was wird durch dieses Programm berechnet?

Adresse a	Inhalt RAM[a]
(0) 0...000000	0...00000010 (2)
(1) 0...000001	0...00010110 (22)
(2) 0...000010	0...00000101 (5)
(3) 0...000011	0...00000010 (2)
(4) 0...000100	0...00010111 (23)
(5) 0...000101	0...00000111 (7)
(6) 0...000110	0...00000100 (4)
(7) 0...000111	0...00000001 (1)
(8) 0...001000	0...00011000 (24)
(9) 0...001001	0...00001011 (11)
(10) 0...001010	0...00011001 (25)
(11) 0...001011	0...00000011 (3)
(12) 0...001100	0...01100100 (100)
(13) 0...001101	0...00000101 (5)
(14) 0...001110	0...00000010 (2)
(15) 0...001111	0...00011000 (24)
(16) 0...010000	0...00001001 (9)
(17) 0...010001	0...00000100 (4)
(18) 0...010010	0...00000001 (1)
(19) 0...010011	0...00011000 (24)
(20) 0...010100	0...00001010 (10)
(21) 0...010101	0...00011001 (25)
(22) 0...010110	0...00001011 (11)
(23) 0...010111	0...00000100 (4)
(24) 0...011000	0...00000000 (0)
⋮	⋮

Sie kennen vermutlich schon die folgende Anekdote aus dem Leben von C. F. Gauß[1] (bzw. eine Variante davon):

Um seine Schüler (bzw. Gauß) eine Zeitlang zu beschäftigen (bzw. zu bestrafen), gab der Lehrer ihnen (bzw. ihm) die Aufgabe, die Zahlen von 1 bis 100 zu addieren. Doch der kleine Gauß hatte die Lösung schon nach kurzer Zeit gefunden. Er hatte nämlich erkannt, dass man statt $1+2+3+\cdots+98+99+100$ zu rechnen auch 50 „Paare“ $1+100$, $2+99$, $3+98$, ..., $50+51$ bilden kann, die jeweils 101 ergeben, und sich so die Lösung $50 \cdot 101 = 5050$ leicht berechnen lässt.

D. h.: Gauß benutzte die (nach ihm benannte) Summenformel $\sum_{k=1}^{n} k = \frac{1}{2}n(n+1)$.

d) Mit dem oben gegebenen Befehlssatz ist $[(n+1) * n] \div 2$ auch viel einfacher zu berechnen als $[\cdots[n+(n-1)]+\cdots+2]+1$. Warum? Warum ist $[\cdots[1+2]+\cdots+(n-1)]+n$ noch umständlicher? Warum ist $[n \div 2] * (n+1)$ sogar falsch? Gibt es einen Ausdruck, der den gleichen Wert liefert, aber noch einfacher zu berechnen ist als $[(n+1) * n] \div 2$?

e) Schreiben Sie ein Programm, das

$$s = \begin{cases} \sum_{k=1}^{n} k^2 = n(n+1)(2n+1)/6 & \text{falls } m = 2 \\ \sum_{k=1}^{n} k \;\; = n(n+1)/2 & \text{sonst} \end{cases}$$

berechnet und das Ergebnis bei Adresse 4 speichert, wobei m bei Adresse 2 und n bei Adresse 3 gespeichert seien und das Programm selbst bei Adresse 5 beginnt. Sie können davon ausgehen, dass $n \geq 0$ gilt. Kommentieren Sie Ihr Programm unbedingt ausführlich!

Adresse a	Inhalt RAM[a]
(0) 0...000000	0...00001010 (10)
(1) 0...000001	0...00000101 (5)
(2) 0...000010	*...******** (m)
(3) 0...000011	*...******** (n)
(4) 0...000100	*...******** (s)
⋮	⋮

Hinweis: Nutzen Sie (u. a.) aus, dass $n(n+1)(2n+1)/6 = [\,n(n+1)/2\,] \cdot (2n+1) \;/\; 3$ gilt.

---

http://www-i5.informatik.rwth-aachen.de/LuFG/Lehre/SS00/RS/

[1] http://www.matheprisma.uni-wuppertal.de/Module/Quadrat/biogauss.htm
http://www.ham.nw.schule.de/friedensschule/mathematik/seite20.htm
http://www.solidaritaet.com/ibykus/1997/1/gauss.htm