Saarland University
Department 6.2 - Computer Science
Prof. Dr. W. J. Paul
Jonas Oberhauser

System Architecture - SS15

Exercise Sheet 3(due: May 11, 2015)

Wichtig:

- Sie Benötigen 50% aller Übungsblätter die für Klausur X relevant sind, um zu Klausur X zugelassen zu werden. Dieses Blatt ist Relevant für Vor- und Nachklausur.
- Das Übungsblatt muss stets am Montag nach der Vorlesung bei mir in der Office Hour oder, falls zeitgleich, in der Übungsgruppe Ihrer Tutorin abgegeben werden.
- Geben Sie stets Ihren Namen, Ihre Matr. Nr., und den Namen ihrer Tutorin auf der vordersten Seite oben rechts an.
- Sie dürfen Ergebnisse von vorherigen Aufgaben verwenden, auch wenn Sie diese nicht gelöst haben. Markieren sie Gleichungen, in denen Sie ein vorheriges Ergebniss benutzen, mit dem Kürzel E+Aufgabenblatt+Aufgabennummer.
- Wenn Sie sich nicht für die Klausur vorbereiten möchten, aber trotzdem zugelassen werden möchten, schreiben Sie einfach Ihren Namen und Ihre Matrikelnummer auf die Lösung einer kompetenten Mitstudentin. Es besteht auch keine Anwesenheitspflicht in den Übungsgruppen.

Tutor: _____

Namen, Matr. Nummern: _____

Aufgabe 1: (1)

Warum kann man

$$\forall x, y. x \wedge y = y \wedge x$$

aber nicht

$$\forall x, y. x \cdot y = y \cdot x$$

durch Fallunterscheidung lösen, obwohl Boole's Einbettung in die Zahlen

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

definiert?

> **Solution:** Die Menge $\mathbb{B}$ hat nur endlich viele Elemente. Die Menge $\mathbb{N}$ hat unendlich viele und kann durch endlich viele Fälle nicht abgedeckt werden, sondern nur durch schematische Beweise wie Induktion. Die obere Gleichung gilt für x, y in $\mathbb{B}$, die untere für $\mathbb{N}$.

Aufgabe 2: (2)

Die kleine Raupe Nimmersatt hat in den folgenden Gleichungen den dritten Strich in dem Identitätsgleichungssymbol $\equiv$, den zweiten Strich bei der Umformungsäquivalenz $\iff$, und ein paar Klammern gefressen. Fügen Sie sie an den richtigen Stellen wieder ein, so dass die Aussagen stimmen!

(a)

$$e = 1 \leftrightarrow e' = 1 \leftrightarrow e = e'$$

Saarland University
Department 6.2 - Computer Science
Prof. Dr. W. J. Paul
Jonas Oberhauser

System Architecture - SS15

Exercise Sheet 3(due: May 11, 2015)

(b)

$$e = 1 \wedge e' = 1 \rightarrow e = e'$$

(c)

$$e = e' \not\rightarrow e = e'$$

(d)

$$e \wedge e' = 1 \rightarrow e = 1 \wedge e' = 1$$

(e)

$$e = 1 \wedge e' = 1 \rightarrow e \wedge e' \neq 1$$

Solution: Es gibt eventuell mehrere Lösungen.

$$(e = 1 \iff e' = 1) \leftrightarrow e \equiv e'$$

$$e = 1 \wedge e' = 1 \rightarrow e = e'$$

$$e \equiv e' \not\rightarrow e = e'$$

$$e \wedge e' \equiv 1 \rightarrow e \equiv 1 \wedge e' \equiv 1$$

$$e = 1 \wedge e' = 1 \rightarrow e \wedge e' \neq 1$$

Aufgabe 3: (1)

Konstruieren sie einen Volladdierer...

(a) (1 points) ...mit nur 5 Gattern

(b) (1 point (bonus)) ...mit nur 2 Gattern und einem Multiplexer

Solution: Wir definieren

$$z = (a \oplus b),$$
$$s = z \oplus c,$$
$$c' = a \wedge b \vee c \wedge z,$$

oder

$$c' = (z?c : a).$$

Aufgabe 4: (1)

(a) (1 point) Zeigen Sie, dass unsere Semantik für Schaltkreise nicht vollständig ist bezgl. der korrekten Schaltkreise, i.e., dass es Schaltkreise gibt, die zwar (teilweise) Funktionieren, aber deren Verhalten durch unsere Semantik nicht definiert ist.

(b) (1 point (bonus)) Geben Sie eine Möglichkeit an, auch diesen Schaltkreisen ein Wohldefiniertes verhalten zu geben.

Saarland University
Department 6.2 - Computer Science
Prof. Dr. W. J. Paul
Jonas Oberhauser

System Architecture - SS15

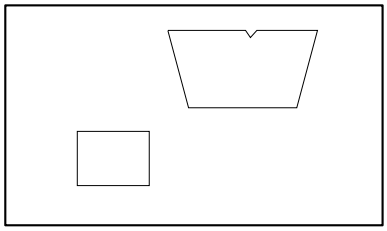
Exercise Sheet 3(due: May 11, 2015)

Solution: Beispiel flip-flop. Semantik definieren zum Beispiel durch Fixpunkt, oder Zeitliche Analyse/Gatterlaufzeit, ...

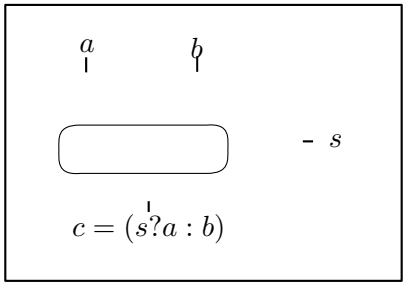
Aufgabe 5: (4)

Die kleine Raupe Nimmersatt hat jetzt auch alle Kabel und Bezeichner aus den folgenden Schaltkreisen gefressen! Fügen Sie sie wieder ein, und beschriften Sie sie vollständig.

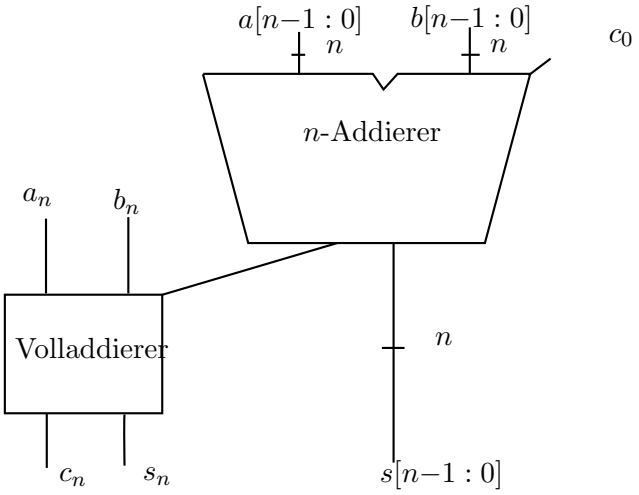
(a) (2 points) $S(n)$-Addierer:



(b) (2 points) Multiplexer (MUX) (der Ausdruck entspricht dem ternären Operator in C):



Solution: Lösung für Multiplexer steht im Buch. Minimallösung für den Addierer:



(Auch Korrekt sind z.B. Annotationen für die unären Kabel)

Aufgabe 6: (2)

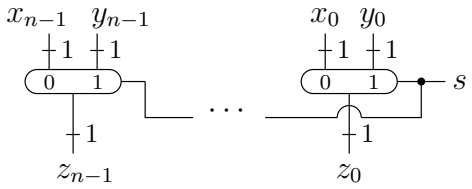
(a) (1 point) Zeigen Sie, dass der Multiplexer aus der Vorlesung funktioniert.

Saarland University
Department 6.2 - Computer Science
Prof. Dr. W. J. Paul
Jonas Oberhauser

System Architecture - SS15

Exercise Sheet 3(due: May 11, 2015)

(b) (1 point) Wieso ist dies eine Definition:



Solution:

$$c = (a \wedge s \vee b \wedge \neg s)$$

Fallunterscheidung s:

- $s = 1$:
$$c = a \wedge 1 \vee b \wedge 0 = a \vee 0 = a = (1?a : b)$$
- $s = 0$:
$$c = a \wedge 0 \vee b \wedge 1 = 0 \vee b = b = (0?a : b)$$

Es ist eine Definition, da das Symbol des Multiplexers für $n = 1$ vorweg definiert wurde, und hier noch für $n > 1$ Überladen wird.

Aufgabe 7: (1)

Lösen Sie die allgemeinen Rekurrenzgleichungen:

$$C_1(1) = c, \qquad C_2(0) = c,$$
$$D_1(1) = c, \qquad D_2(0) = c,$$

und

(a) (1 point)

$$C_1(2n) = C_1(n) + k$$
$$D_2(n+1) = D_2(n) + k$$

(b) (1 point (bonus)) Für $a \geq 2$:

$$D_1(2n) = a \cdot D_1(n) + k$$
$$C_2(n+1) = a \cdot C_2(n) + k$$

Solution: Wir Zeigen nur die Bonusaufgaben. Sei

$$\overline{D}(n) = D_1(2^n).$$

Dann erhalten wir durch mehrfaches einsetzen:

$$\begin{aligned}\overline{D}(n) &= a \cdot \overline{D}(n-1) + k & &= a \cdot (a \cdot \overline{D}(n-2) + k) + k \\ &= a^2 \cdot \overline{D}(n-2) + a \cdot k + k & &= a^3 \cdot \overline{D}(n-3) + a^2 \cdot k + a \cdot k + k\end{aligned}$$

und generalisieren:

$$\overline{D}(n+x) = a^x \cdot \overline{D}(n) + \sum_{i=0}^{x-1} a^i \cdot k.$$

Das Beweisen wir durch Induktion über x und folgern direkt

$$\overline{D}(n) = a^n \cdot \overline{D}(0) + \sum_{i=0}^{n-1} a^i \cdot k$$

(alternativ kann man auch diese Formel erraten und direkt per Induktion über n zeigen).

Um das in geschlossene Form zu bringen, schätzen wir es wie folgt nach oben ab. Annahme: es gibt

$$P(n) \geq \sum_{i=0}^{n-1} a^i$$

$$\begin{aligned}\sum_{i=0}^{S(n)-1} a^i &= a^{S(n)} + \sum_{i=0}^{n-1} a^i \\ &= a \cdot a^n + \sum_{i=0}^{n-1} a^i & &\leq a \cdot a^n + P(n) \leq P(S(n))\end{aligned}$$

Dies gilt z.B., wie wir erraten, für

$$P(n) = a \cdot a^n,\ P(n) = 2 \cdot a^n,\ \ldots$$

und einige mehr.

In jedem Fall ist dann aber

$$\overline{D}(n) \leq a^n \cdot \overline{D}(0) + P(n) \cdot k,$$

und z.B.

$$D_1(2^n) \leq a^n \cdot \overline{c} + 2 \cdot a^n \cdot k.$$

Wir schliessen daraus direkt dass

$$D_1(n) \leq (2 \cdot k + c) \cdot a^{\log_2 n} = (2k + c) \cdot n \cdot a^{\log_2 a}.$$

Man beachte dass $\overline{D}$ auch eine Abschätzung für C_2 ist und daher

$$C_2(n) \leq (2k + c) \cdot a^n.$$

Saarland University
Department 6.2 - Computer Science
Prof. Dr. W. J. Paul
Jonas Oberhauser

System Architecture - SS15

Exercise Sheet 3(due: May 11, 2015)

Aufgabe 8: (2)

(a) (1 point) Definieren Sie einen n-bit Incrementer, i.e., einen Schaltkreis mit input $a \in \mathbb{B}^n$ und outputs $c \in \mathbb{B}, b \in \mathbb{B}^n$ sodass

$$\langle cb \rangle = \langle a \rangle + 1.$$

(b) (1 point) Stellen Sie Rekurrenzgleichungen für Kosten und Delay auf und Lösen sie diese.

(c) (1 point (bonus)) Ihr Incrementer soll lineare Kosten und logarithmischen Delay haben.

Solution: Der Basisfall ist z.B. ein Half-Adder oder ein Negationsgatter:

$$c_0 = a_0,\ b_0 = \neg a_0.$$

Rekursiv definieren wir wie beim CSA, aber mit nur zwei rekursiven Aufrufen:

$$\begin{aligned} c_n b[n{-}1:0] &= inc_n(a[n{-}1:0]) \\ inc_{2n}(a) &= (c_n ? inc_n(a[2n{-}1:n]) : a[2n{-}1:n]) \end{aligned}$$

Das hat die Rekurrenzgleichungen

$$\begin{aligned} D(1) &= 1, \\ D(2n) &= D(MUX_n) + D(n), \\ C(1) &= 1, \\ C(2n) &= C(MUX_n) + 2 \cdot C(n). \end{aligned}$$

Die Tiefe fällt in das Schema der letzten aufgabe und ist somit gelöst. Für die Kosten sehenn wir das pro ebene die gleiche Anzahl an MUXen verwendet wird:

$$C(8n) = C(MUX_{4n}) + 2 \cdot C(MUX_{2n}) + 4 \cdot C(MUX_n) + 8 \cdot C(n) = 4n \cdot C(MUX) + 2 \cdot 2n \cdot C(MUX) + 4 \cdot n \cdot C(MU$$

Daraus erraten wir und beweisen durch Induktion über k

$$C(2^k) = k \cdot 2^k \cdot C(MUX) + 2^k \cdot C(1).$$

Um auf lineare Kosten zu kommen, benutzen wir z.B. einen Parallel-Prefix adder oder ein Parallel-Prefix mit And, um die carries zu berechnen, und berechnen dann direkt das Ergebnis mit Xor gattern:

$$c[n:1] = PP_\wedge(a),\ b_i = a_i \oplus c_i.$$

Offensichtlich gilt

$$c_{i+1} = c_i \wedge a_i,$$

und der Schaltkreis ist korrekt.