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

System Architecture - SS15

Exercise Sheet 4(due: May 18, 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)

Geben Sie die Rekursive Definition des Parallel-Prefix Schaltkreises an!

Aufgabe 2: (3)

Wir wollen einen Schaltkreis mit logarithmischer Tiefe und linearen Kosten konstruieren, der die Stelle der ersten 1 findet.

(a) (1 point) Konstruieren Sie einen Schaltkreis mit input $a \in \mathbb{B}^n$ und output $b \in \mathbb{B}^n$, sodass

$$b_i = \begin{cases} 1 & \exists j \leq i.a_j = 1 \\ 0 & \text{o.w.} \end{cases}$$

(b) (1 point) Konstruieren Sie einen Schaltkreis mit input $a \in \mathbb{B}^n$ und output $c \in \mathbb{B}^n$, sodass

$$c_i = 1 \iff i = \min\{i \mid a_i = 1\}.$$

(c) (1 point) Ihr Schaltkreis soll lineare Kosten und logarithmische Tiefe haben.

Solution: Wir berechnen b durch Parallel Prefix mit Or. Wichtig ist dabei, dass

1. Or assoziativ ist
2. $(\exists j \leq i.a_j = 1) \iff a_0 \vee \ldots a_i$.

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

System Architecture - SS15

Exercise Sheet 4(due: May 18, 2015)

Dann berechnen wir c wie folgt:

$$c_i = (b_{i-1} \oplus b_i),\ c_0 = b_0$$

Es gilt $b_i = a_i \vee b_i$, und deshalb

$$\begin{aligned} c_i &= b_{i-1} \oplus (a_i \vee b_{i-1}) \\ &= (\exists j < i.a_j = 1) \oplus (a_i \vee \exists j < i.a_j = 1) \\ &= \begin{cases} 0 & \exists j < i.a_j = 1 \\ a_i & \text{o.w.} \end{cases} \end{aligned}$$

was genau die Position der ersten 1 beschreibt.

Aufgabe 3: (4)

(a) (1 point) Definieren Sie die generate und propagate bits g_{0j}, p_{0j} für den Carry-Look-Ahead Adder.

(b) (1 point) Konstruieren Sie die Inputs für den Parallel-Prefix Schaltkreis im Carry-Look-Ahead Adder.

(c) (1 point) Zeigen Sie dass der Operator $\circ$ assoziativ ist:

$$a[1:0] \circ b[1:0] = (a_1 \vee (a_0 \wedge b_1), a_0 \wedge b_0)$$
$$(a \circ b) \circ c = a \circ (b \circ c)$$

(d) (1 point) Wieso muss der Operator $\circ$ assoziativ sein? (Wo geht der Beweis sonst schief)?

Solution: Teile 1 - 2 stehen im Buch auf Seite 72f.

Die Assoziativität zeigen wir für jedes Bit einzeln, und für das bit 0 folgt es direkt aus der assoziativität von And:

$$((a \circ b) \circ c)_0 = ((a_0 \wedge b_0) \wedge c_0) = a_0 \wedge (b_0 \wedge c_0) = (a \circ (b \circ c))_0,$$

während das bit 1 etwas aufwändiger ist:

$$\begin{aligned} ((a \circ b) \circ c)_1 &= (a \circ b)_1 \vee (a \circ b)_0 \wedge c_1 \\ &= (a_1 \vee (a_0 \wedge b_1)) \vee (a_0 \wedge b_0) \wedge c_1 \\ &= a_1 \vee (a_0 \wedge b_1 \vee a_0 \wedge (b_0 \wedge c_1)) \\ &= a_1 \vee (a_0 \wedge (b_1 \vee (b_0 \wedge c_1))) \\ &= a_1 \vee (a_0 \wedge (b \circ c)_1) \\ &= (a \circ (b \circ c))_1. \end{aligned}$$

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

System Architecture - SS15

Exercise Sheet 4(due: May 18, 2015)

Der Operator muss assoziativ sein, weil sonst die rekursive Konstruktion nicht das richtige Ergebnis berechnet:

$$o_{2i+2} = (a_{2i+2} \circ a_{2i+1}) \circ o_{2i} \qquad \overset{assoc\circ}{=} a_{2i+2} \circ (a_{2i+1} \circ o_{2i}).$$

Aufgabe 4: (2)

(a) (1 point) Zeigen Sie, dass die folgenden Ausdrücke beide zur Berechnung des Negativ-Bits benutzt werden können (im Fall $u = 0$):

$$\overline{c_n} \wedge (a_{n-1} \oplus d_{n-1}) \vee a_{n-1} \wedge d_{n-1},$$

und

$$c_n \oplus a_{n-1} \oplus d_{n-1}.$$

(b) (1 point) Zur Berechnung des Overflows benutzen wir die Formel

$$c_n \oplus c_{n-1}.$$

Wieso ist das problematisch und wie können wir das Problem effizient umgehen?

Solution: Wir machen zuerst die Beobachtungen dass

$$c_n \to a_{n-1} \vee d_{n-1}$$

und

$$a_{n-1} \wedge d_{n-1} \to c_n.$$

Dann gilt:

$$\begin{aligned}
\overline{c_n} \wedge (a_{n-1} \oplus d_{n-1}) \vee a_{n-1} \wedge d_{n-1} &\iff \overline{c_n} \wedge (a_{n-1} \oplus d_{n-1}) \vee c_n \wedge a_{n-1} \wedge d_{n-1} \vee c_n \wedge (\overline{a_{n-1}} \wedge \overline{d_{n-1}} \\
&\iff \overline{c_n} \wedge (a_{n-1} \oplus d_{n-1}) \vee c_n \wedge \overline{a_{n-1} \oplus d_{n-1}} \\
&\iff c_n \oplus a_{n-1} \oplus d_{n-1}.
\end{aligned}$$

Um Overflow zu berechnen brauchen wir c_{n-1}, welches nicht von allen Addierern berechnet wird (z.B. CSA). Um dennoch an c_{n-1} zu kommen reicht uns die Überlegung dass

$$s_{n-1} = a_{n-1} \oplus d_{n-1} \oplus c_{n-1}$$

und daher

$$c_{n-1} = s_{n-1} \oplus a_{n-1} \oplus d_{n-1},$$

welche alle bekannt sind.