Übungsblatt 5

Jan Hendrik Dithmar (Matrikelnummer 2031259)
Carsten Saar (Matrikelnummer 2030107)
Thomas Ospelt (Matrikelnummer 2029959)
Timo Adam (Matrikelnummer 2031946)
Übungsgruppe 8 - Di, 15-17 Uhr
Steffen Heil

Aufgabe 1

$$\begin{aligned}
[s] &= -s_{n-1}2^{n-1} + \langle s[n-2:0]\rangle \\
[a] &= -a_{n-1}2^{n-1} + \langle a[n-2:0]\rangle \\
[b] &= -b_{n-1}2^{n-1} + \langle b[n-2:0]\rangle
\end{aligned}$$

$$\begin{aligned}
[a]=[b] \quad &\Leftrightarrow \quad -a_{n-1}2^{n-1} + \left(\sum_{i=0}^{n-2} a_i 2^i\right) = -b_{n-1}2^{n-1} + \left(\sum_{i=0}^{n-2} b_i 2^i\right) \\
&\Leftrightarrow \quad -a_{n-1}2^{n-1} + \left(\sum_{i=0}^{n-2} a_i 2^i\right) - \left[-b_{n-1}2^{n-1} + \left(\sum_{i=0}^{n-2} b_i 2^i\right)\right] = 0 \\
&\Leftrightarrow \quad -a_{n-1}2^{n-1} + \left(\sum_{i=0}^{n-2} a_i 2^i\right) + b_{n-1}2^{n-1} - \left(\sum_{i=0}^{n-2} b_i 2^i\right) = 0 \\
&\Leftrightarrow \quad (b_{n-1}-a_{n-1})2^{n-1} + \left(\sum_{i=0}^{n-2} (b_i - a_i) 2^i\right) = 0 \\
&\Leftrightarrow \quad b_i - a_i = 0 \ \ \forall i \\
&\Leftrightarrow \quad b_i = a_i \ \ \forall i \\
&\Leftrightarrow \quad a_i = b_i \ \ \forall i \ \ (*)
\end{aligned}$$

$$\begin{aligned}
[s] &= [a]-[b] = (a_{n-1}-b_{n-1})2^{n-1} + \left(\sum_{i=0}^{n-2}(b_i-a_i)2^i\right) \\
&= -s_{n-1}2^{n-1} + \left(\sum_{i=0}^{n-2} s_i 2^i\right) = 0 \\
&\Leftrightarrow \quad s_i = a_i - b_i \ \ (**)
\end{aligned}$$

„⇒“: $s[n-1:0] = 0^n \Rightarrow [a]=[b]$:

$$\begin{aligned}
s[n-1:0] = 0^n \quad &\Leftrightarrow \quad s_i = 0 \ \ \forall i \\
&\overset{(**)}{\Leftrightarrow} \quad a_i - b_i = 0 \ \ \forall i \\
&\Leftrightarrow \quad a_i = b_i \ \ \forall i \\
&\overset{(*)}{\Leftrightarrow} \quad [a]=[b]
\end{aligned}$$

„⇐“: $[a]=[b] \Rightarrow s[n-1:0] = 0^n$:

$$\begin{aligned}
[a]=[b] \quad &\overset{(*)}{\Leftrightarrow} a_i = b_i \ \ \forall i \\
&\Leftrightarrow \quad a_i - b_i = 0 \\
&\overset{(**)}{\Leftrightarrow} \quad s_i = 0 \\
&\Leftrightarrow \quad s[n-1:0] = 0^n
\end{aligned}$$

Aufgabe 2

Soll-Funktionsweise des Decoders:

$$\mathrm{Dec}_n(x[n-1:0]) = d[2^n-1:0] \text{ mit } d_i = \begin{cases} 0 & \text{für } i \neq \langle x\rangle \\ 1 & \text{für } i = \langle x\rangle \end{cases}$$

Funktionsweise des Decoders aus der Vorlesung:

Sei im folgenden $x_{n-1} = 1$. Nach Konstruktions-Definition ist

$$d[2^{n-1} - 1 : 0] = 0^{2^{n-1}}$$
$$d[2^n - 1 : 2^{n-1}] = Dec_{n-1}(x[n-2:0])$$

Beweis:

Die Korrektheit des Dec_{n-1} ist als Voraussetzung gegeben. Somit gilt

$$d_i = 0 \quad \forall i \in \{0, \ldots, 2^{n-1} - 1\}$$
$$d_i = 1 \quad \Longleftrightarrow \quad i = \langle x[n-2:0] \rangle + 2^{n-1} \quad \forall i \in \{2^{n-1}, \ldots, 2^n - 1\}$$

Da $\forall i < 2^{n-1} \quad d_i = 0$ folgt offensichtlich

$$d_i = 1 \quad \Longleftrightarrow \quad i = \langle x[n-2:0] \rangle + 2^{n-1} \quad \forall i \in \{0, \ldots, 2^n - 1\}$$
$$d_i = 1 \quad \Longleftrightarrow \quad i = \langle x[n-1:0] \rangle \quad \forall i \in \{0, \ldots, 2^n - 1\}$$

Dies entspricht der Definition der Funktionsweise des Decoders. □

Aufgabe 3

(a) $\text{FDec}_1(x[0:0]) := d[1:0]$ mit $d_0 = \overline{x_0}$ und $d_1 = x_0$
$\text{FDec}_n(x[n-1:0]) := d[2^n - 1 : 0]$ mit
$l\left[2^{n/2} - 1 : 0\right] = \text{FDec}_{\frac{n}{2}}\left(x\left[\frac{n}{2} - 1 : 0\right]\right)$ und
$h\left[2^{n/2} - 1 : 0\right] = \text{FDec}_{\frac{n}{2}}\left(x\left[n - 1 : \frac{n}{2}\right]\right)$ und
$d_i := l_{i \bmod 2^{n/2}} \wedge h_{i \operatorname{div} 2^{n/2}}$

(b) Soll-Funktionsweise des Decoders:

$$\text{FDec}_n(x[n-1:0]) = d[2^n - 1 : 0] \text{ mit } d_i = \begin{cases} 0 & \text{für } i \neq \langle x \rangle \\ 1 & \text{für } i = \langle x \rangle \end{cases}$$

Induktionsanfang $n = 1$:

$$\text{FDec}_n(x[n-1:0]) = \text{FDec}_1(x_0) = d[1:0]$$

$$\begin{aligned}
d_0 &= \begin{cases} 0 & \text{für } 0 = i \neq \langle x \rangle = x_0 \\ 1 & \text{für } 0 = i = \langle x \rangle = x_0 \end{cases} \\
d_0 &= \begin{cases} 0 & \text{für } x_0 \neq 0 \\ 1 & \text{für } x_0 = 0 \end{cases} \\
d_0 &= \overline{x_0} \quad \text{(nach Soll-Funktionsweise)} \\
d_0 &= \overline{x_0} \quad \text{(nach Konstruktion)} \\
d_1 &= \begin{cases} 0 & \text{für } 1 = i \neq \langle x \rangle = x_0 \\ 1 & \text{für } 1 = i = \langle x \rangle = x_0 \end{cases} \\
d_1 &= \begin{cases} 0 & \text{für } x_0 \neq 1 \\ 1 & \text{für } x_0 = 1 \end{cases} \\
d_1 &= x_0 \quad \text{(nach Soll-Funktionsweise)} \\
d_1 &= x_0 \quad \text{(nach Konstruktion)}
\end{aligned}$$

Induktionsschritt $n \rightsquigarrow 2n$**:**
Die Korrektheit des FDec_n sei als Induktionsvoraussetzung gegeben.

$$
\begin{aligned}
l[2^n-1:0] &= \mathrm{FDec}_n(x[n-1:0]) \\
l_i &\overset{\text{IV}}{=} \begin{cases} 0 & \text{für } i \neq \langle x[n-1:0]\rangle \\ 1 & \text{für } i = \langle x[n-1:0]\rangle \end{cases} \\
h[2^n-1:0] &= \mathrm{FDec}_n(x[2n-1:n]) \\
h_i &\overset{\text{IV}}{=} \begin{cases} 0 & \text{für } i \neq \langle x[2n-1:n]\rangle \\ 1 & \text{für } i = \langle x[2n-1:n]\rangle \end{cases} \\
l_i \wedge h_j = 1 &\iff i = \langle x[n-1:0]\rangle \quad \wedge \quad j = \langle x[2n-1:n]\rangle \\
d_i = 1 &\iff l_{i \bmod 2^n} \quad \wedge \quad h_{i \operatorname{div} 2^n} = 1 \\
&\iff i \bmod 2^n = \langle x[n-1:0]\rangle \quad \wedge \quad i \operatorname{div} 2^n = \langle x[2n-1:n]\rangle \\
&\quad \text{daraus sollte folgen, dass} \\
d_i = 1 &\iff i = \langle x[2n-1:0]\rangle \\
&\quad \text{andererseits wissen wir, dass gilt} \\
i &= (i \operatorname{div} 2^n) \cdot 2^n + (i \bmod 2^n) \\
&= (\langle x[2n-1:n]\rangle) \cdot 2^n + (\langle x[n-1:0]\rangle) \\
&= \langle x[2n-1:0]\rangle
\end{aligned}
$$

Somit folgt, dass das i-te Ergebnis-Bit des FDec_n genau dann gleich 1 ist, wenn $i = \langle x[2n-1:0]\rangle$. Dies stimmt mit den geforderten Eigenschaften des Decoders überein. □

(c) $C(\mathrm{FDec}_1) = 1, \quad C(\mathrm{FDec}_n) = 2 \cdot C(\mathrm{FDec}_{\frac{n}{2}}) + 2^n$
Behauptung:

$$C(\mathrm{FDec}_n) = n + n \cdot \sum_{k=1}^{\log_2 n} 2^{2^k - k}$$

Induktionsanfang $n = 1$**:**

$$C(\mathrm{FDec}_n) = C(\mathrm{FDec}_1) = 1 \quad \text{(nach Definition)}$$

$$C(\mathrm{FDec}_n) = C(\mathrm{FDec}_1) = 1+1 \cdot \sum_{k=1}^{\log_2 1} 2^{2^k-k} = 1+\sum_{k=1}^{0} 2^{2^k-k} = 1 \quad \text{(nach Behauptung)}$$

Induktionsschritt $n \rightsquigarrow 2n$:

$$
\begin{aligned}
C(\mathrm{FDec}_{2n}) &\overset{\text{Def}}{=} 2 \cdot C(\mathrm{FDec}_{\frac{2n}{2}}) + 2^{2n} \\
&= 2 \cdot C(\mathrm{FDec}_n) + 2^{2n} \\
&\overset{\text{Beh}}{=} 2 \cdot \left(n + n \cdot \sum_{k=1}^{\log_2 n} 2^{2^k - k} \right) + 2^{2n} \\
&= 2n + 2n \cdot \left(\sum_{k=1}^{\log_2 n} 2^{2^k - k} \right) + 2^{2n} \\
&= 2n + 2n \cdot \left[\left(\sum_{k=1}^{\log_2(2n)} 2^{2^k - k} \right) - 2^{2^{\log_2(2n)} - \log_2(2n)} \right] + 2^{2n} \\
&= 2n + 2n \cdot \left[\left(\sum_{k=1}^{\log_2(2n)} 2^{2^k - k} \right) - \frac{2^{2^{\log_2(2n)}}}{2^{\log_2(2n)}} \right] + 2^{2n} \\
&= 2n + 2n \cdot \left[\left(\sum_{k=1}^{\log_2(2n)} 2^{2^k - k} \right) - \frac{2^{2n}}{2n} \right] + 2^{2n} \\
&= 2n + 2n \cdot \left(\sum_{k=1}^{\log_2(2n)} 2^{2^k - k} \right) - 2^{2n} + 2^{2n} \\
&= 2n + 2n \cdot \left(\sum_{k=1}^{\log_2(2n)} 2^{2^k - k} \right)
\end{aligned}
$$

□

Aufgabe 4

siehe Anhang (Zeichnungen)

Aufgabe 5

Berechne unter der Voraussetzung

- dass die Breite der Eingangssignale 32 Bit beträgt und
- dass als Addierer ein Conditional Sum Adder verwendet wird

die Tiefe des in der Vorlesung vorgestellten ALU-Schaltkreises.

(i) Betrachte zunächst die Tiefe für den CSA. Für $n = 2^k,\ k \in \mathbb{N}_{>0}$ gilt

$$D(\mathrm{CSA}_1) = D(\mathrm{FA}) = 3$$

$$D(\mathrm{CSA}_n) = D(\mathrm{CSA}_{\frac{n}{2}}) + D(\mathrm{MUX}_{\frac{n}{2}+1}) = D(\mathrm{CSA}_{\frac{n}{2}}) + 3$$

Beziehungsweise in geschlossener Form für $n = 2^k,\ k \in \mathbb{N}$

$$D(\mathrm{CSA}_n) = 3 \log_2 n + 3$$

(ii) Betrachte zunächst die für die Addition und Subtraktion zu durchlaufenden Pfade. Die Eingangssignale a_i kommen direkt zum CSA, die Eingangssignale b_i müssen jedoch bevor sie zum CSA gelangen, jeweils ein XOR-Gatter durchlaufen (Tiefe 1). Der CSA hat eine Tiefe von $3\log_2 n + 1 = 3\log_2 32 + 3 = 3 \cdot 5 + 3 = 18$. Bis zum Endergebnis werden zwei MUXe durchlaufen, jeder von diesen hat für sich betrachtet die Tiefe 3, allerdings entstammen beide Steuerbits dem Kommandosignal und sind nicht vom Ergebnis des CSA abhängig. Die erforderlichen Negierungen der Steuersignale im Inneren der MUXe können somit für die Tiefenbetrachtung außer Acht gelassen werden, womit sich die Tiefe beider MUXe in dieser Betrachtung auf 2 reduziert (in jedem der MUXe durchläuft jedes Signal jeweils ein AND-Gatter und ein OR-Gatter). Die maximale Gesamttiefe der Addition bzw. Subtraktion beträgt somit $1 + 18 + 2 + 2 = 23$.

(iii) Betrachte jetzt den Logikteil, der für die Operationen $\wedge_{32}, \vee_{32}, \oplus_{32}$ und $b[15:0] \cdot 2^{16}$ zuständig ist. Die Pfade für die Operationen $\wedge_{32}, \vee_{32}$ und $\oplus_{32}$ sind gleich lang; jedes Bit der Eingangssignale durchläuft ein AND-, OR- bzw. XOR-Gatter (Tiefe 1) und vier MUXe (Tiefe jeweils 2 nach Vorbetrachtung analog zu Punkt (ii). Für die Operation $b[15:0] \cdot 2^{16}$ werden nur vier MUXe durchlaufen, dieser Pfad ist somit kürzer als für die anderen drei Operationen und muss somit bei der Betrachtung der maximalen Tiefe der ALU nicht weiter berücksichtigt werden. Die Tiefe der Operationen $\wedge_{32}, \vee_{32}$ und $\oplus_{32}$ beträgt somit $1 + 4 \cdot 4 = 17$.

(iv) Betrachte schließlich den Vergleichsteil der ALU, der für die Operationen $0_{32}, >, =, \geq, <, \neq, \leq$ und 1_{32} verantwortlich ist. Der Tester erhält drei Signale zero, positive und negative, die aussagen, ob die Differenz $[a] - [b]$ gleich null, positiv oder negativ ist. Zur Erzeugung dieser Signale ist es erforderlich, die Signale b_i zu invertieren (Tiefe 1) und zusammen mit den unveränderten Signalen a_i in einen CSA zu geben (Tiefe 18), wobei der Carry-Eingang des CSA auf 1 gesetzt wird. Das Signal negative erhält man, indem man a_{31}, b_{31} und c_{31} per XOR-Gatter verknüpft (verschachtelte Anordnung zweier XOR-Gatter, Tiefe 2). Das Signal zero erhält man, indem man alle Bits der Differenz miteinander verodert (zuerst werden mit 16 AND-Gattern jeweils zwei Bits verodert, mit 8 AND-Gattern werden dann jeweils zwei der Zwischenergebnisse verodert usw., Tiefe $\log_2 n = 5$) und das Ergebnis anschließend negiert (Tiefe 1). Das Signal positive erhält man, indem man die Signale negative und zero verodert (Tiefe 1) und anschließend negiert (Tiefe 1). Die größte Tiefe hat damit das Signal positive mit einer Tiefe von $1 + 18 + \max\{2, 6\} + 1 + 1 = 27$. Jedes der Signale zero, positive und negative wird mit einem Bit des Kommandobits per AND-Gatter verknüpft (Tiefe 1) und diese Ergebnisse dann per OR-Gatter verknüpft. Dabei ist eine verschachtelte Anordnung der OR-Gatter erforderlich, da aber das Signal positive die größte Tiefe hat, wird man es an das untere OR-Gatter anschließen, so dass die zusätzliche Tiefe nur 1 ist. Dieses Ergebnis muss anschließend noch einen MUX (Tiefe 2, Begründung wie oben) durchlaufen. Die Gesamttiefe des Vergleichsteils der ALU ist somit $27 + 1 + 1 + 2 = 31$.

(v) **Konklusion:** Die größte Tiefe hat wie gezeigt des Vergleichsteil mit einer Tiefe von 31, so dass die Gesamttiefe der ALU 31 beträgt.

Aufgabe 6

(a)

siehe Anhang (Zeichnung)
Ich betrachte im folgenden die binäre Darstellung der Zahlen a und b. Ihre besonderen Eigenschaften als BCD-Ziffern werden als Voraussetzung berücksichtigt. Nach Voraussetzung gilt

$$\langle a \rangle < 10, \quad \langle b \rangle < 10, \quad c \in \{0, 1\}$$

Somit gilt offensichtlich

$$\langle s \rangle = \langle a \rangle + \langle b \rangle + c < 20$$

Zunächst addieren wir mittels eines CSA_4 beide BCD-Zahlen unter Einbeziehung des Carry-Eingangs binär (Tiefe $3 \cdot \log_2 4 + 3 = 3 \cdot 2 + 3 = 9$). Dann gilt es zu prüfen, ob das 5 Bit breite Ergebnis größer oder gleich 10 ist. Aus der Darstellung

$$\langle s \rangle = 16 \cdot s_4 + 8 \cdot s_3 + 4 \cdot s_2 + 2 \cdot s_1 + s_0$$

läßt sich erkennen, dass $\langle s \rangle \geq 10$ genau dann gilt, wenn

$$o := s_4 \vee (s_3 \wedge (s_2 \vee s_1)) = 1$$

Diese Funktion läßt sich durch einen Schaltkreis der Tiefe 3 berechnen. Ist $o = 0$, also $\langle s \rangle < 10$, dann ist das errechnete binäre Ergebnis bereits die gewünschte untere BCD-Ziffer. Die zweite, obere BCD-Ziffer setzt sich in diesem Fall einfach binär aus vier Nullen zusammen. Ist hingegen $o = 1$, also $\langle s \rangle \geq 10$, dann muss das errechnete binäre Ergebnis um 10 vermindert werden, bevor es als untere BCD-Ziffer ausgegeben wird. Die zweite BCD-Ziffer setzt sich binär aus drei Nullen gefolgt von einer Eins zusammen. Im Schaltkreis wird aus der binären Summe gleichzeitig die um 10 verminderte Summe berechnet. Per Multiplexer wird anhand des Overflow-Signals o ausgewählt, welches Ergebnis ausgegeben werden soll. Genaue Realisierung des Schaltkreises anhand Zeichnung.

(b)

Für einen n-Ziffern-BCD-Addierer gilt

$$\text{BCD}^{-1}(s) = \underbrace{\text{BCD}^{-1}(a)}_{\in \{0,\ldots,10^n-1\}} + \underbrace{\text{BCD}^{-1}(b)}_{\in \{0,\ldots,10^n-1\}} + \underbrace{c}_{\in \{0,1\}}$$

Somit gilt offensichtlich

$$\text{BCD}^{-1}(s) \in \{0, \ldots, 2 \cdot (10^n - 1) + 1\} = \{0, \ldots, 2 \cdot 10^n - 1\}$$

Das Ergebnis eines n-Ziffern-BCD-Addierers ist per Definition eine $(n+1)$-Ziffern-BCD-Zahl, welche einen maximal möglichen Wertebereich von $\{0, \ldots, 10^{n+1} - 1\}$ hat. Per Definition enthält die $(n+1)$-te BCD-Ziffer das Ergebnis der ganzzahligen Division von der Summe $\text{BCD}^{-1}(s)$ durch 10^n. Da nach obiger Betrachtung für die Summe gilt

$$\text{BCD}^{-1}(s) < 2 \cdot 10^n$$

heißt dies für die $(n+1)$-te BCD-Ziffer, dass diese immer kleiner als 2 sein muss. Eine BCD-Ziffer stellt Zahlen aus dem Bereich $\{0, \ldots, 9\}$ in Binärkodierung dar. Alle Zahlen < 2 lassen sich bereits mit einem Bit kodieren, so dass automatisch gelten muss

$$s_{4n+3} = s_{4n+2} = s_{4n+1} = 0$$

Daraus folgt, dass n-Ziffern-BCD-Addierer kaskadierbar sind, indem das niederwertigste Bit der $(n+1)$-ten BCD-Ziffer als Carry-In-Bit für den folgenden n-Ziffern-BCD-Addierer verwendet wird. □