Rechnersysteme und -netze Wintersemester 2020/2021

Bastian Goldlücke, Ole Johannsen,
Fred Kunze, Frederik Lattner, Anton Zickenberg, Gregor Diatzko, Alice Hildebrand

6. Übungsblatt - zu bearbeiten bis 14.12.2020

Aufgabe 1 Hardware-Beschreibungssprache (HDL)

Bearbeiten Sie diese Aufgabe mit dem Hardware-Simulator von der Webseite

`http://nand2tetris.org/software.php`

a) Implementieren Sie einen Halb- und einen Volladdierer in HDL!
Zur Implementierung können Sie vorgebene logische Gatter wie z.B. NAND, NOR, AND, OR, NOT, XOR etc. verwenden (siehe `nand2tetris/tools/builtInChips`).

b) Implementieren Sie einen 8-Bit-Übertragskette-Addierer in HDL!
(vgl. Kapitel 3, Folien zu „n-Bit-Übertragskette-Addierer“)

c) Implementieren Sie einen 8-Bit-Inkrementierer in HDL!
(Ein Inkrementierer erhöht seine Eingabe um 1.)

d) Implementieren Sie einen 8-Bit-Übertragsauswahl-Addierer in HDL!
(vgl. Kapitel 3, Folien zu „Optimierung: n-Bit-Übertragsauswahl-Addierer“)

Aufgabe 2 Arithmetik: Multiplizieren

Gegeben seien die beiden im Zweierkomplement zu interpretierenden 4-Bit-Zahlen $x = \texttt{1011}_{2K}$ und $y = \texttt{0111}_{2K}$. In der Vorlesung wurde das Produkt $y \cdot x$ berechnet.

a) Berechnen Sie nun mit dem gleichen Schema (**ohne** Ausnutzen der Kommutativität) das Produkt $x \cdot y$! Vergleichen Sie Ihr Ergebnis mit dem der Vorlesung!
(vgl. Kapitel 3, Folien zu „Multiplikation: Negative Zahlen“)

b) Welches Problem tritt hier auf? Was ist der Grund für dieses Problem?
Wie kann man mit diesem Problem umgehen? (2 Möglichkeiten)

c) Warum führt die einfachere Lösung zu Problemen in bezug auf die Implementierung des Standardalgorithmus mit Hilfe eines Verbundregisters, das das Ergebnis R (im oberen Teil) und den Faktor A (im unteren Teil) enthält?
(vgl. Kapitel 3, Folien zu „Multiplikation: Standardalgorithmus (8 Bit)“)

Aufgabe 3 Arithmetik: Multiplizieren

a) Verifizieren Sie die Rechnung $11_{10} \times 19_{10} = 209_{10}$ im Binärsystem!

b) Verifizieren Sie die Rechnung $(-11)_{10} \times (-19)_{10} = 209_{10}$ im Binärsystem!

c) Berechnen Sie mit Hilfe von Booths Algorithmus $\texttt{011110}_2 \times \texttt{001101}_2$!
Worin besteht hier der Vorteil von Booths Algorithmus?

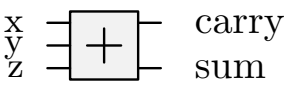
d) Berechnen Sie im Dezimalsystem $9990_{10} \times 88_{10}$ unter Verwendung einer „Dezimalversion“ von Booths Algorithmus!

e) Berechnen Sie im Oktalsystem $511_{10} \times 33_{10}$ unter Verwendung einer „Oktalversion“ von Booths Algorithmus!

Aufgabe 4 Implementierung der Multiplikation in Hardware

Erstellen Sie ein Schaltnetz zur Multiplikation von zwei 4-Bit-Zahlen!

Gesucht ist also eine Gatterschaltung, die zwei Eingaben a und b mit jeweils 4 Binärstellen erhält und eine Ausgabe r mit 8 Binärstellen liefert, die das Produkt der beiden Eingaben ist.
Zusätzlich zu den grundlegenden Gattern dürfen Sie auch den aus der Vorlesung bekannten Volladdierer in einer Kurzform (z.B. wie rechts) verwenden.



Die 4-Bit-Zahlen a und b sollen als vorzeichenlos interpretiert werden. Eine Implementierung, die mit einer Interpretation im Zweierkomplement arbeitet, ist zwar auch möglich, aber aufwendiger, und daher hier nicht gefordert.

Zusatzaufgabe Arithmetik: Multiplizieren

(Zusatzaufgaben können wie normale Aufgaben votiert werden. Eine solche Votierung zählt für die Zahl der abgegebenen Votierungen. Aber Zusatzaufgaben werden nicht für die Zahl der Aufgaben gezählt, aus der sich über die 60%-Quote die Zahl der Aufgaben ergibt, die man votieren muß, um eines der beiden Kriterien der Prüfungszulassung zu erfüllen. Votiert man alle Aufgaben, sowohl alle normalen als auch alle Zusatzaufgaben, kann man folglich über 100% Votierungen erreichen. Ein Vortragen einer Zusatzaufgabe zählt wie das Vortragen einer normalen Aufgabe.)

Schreiben Sie eine Java-Funktion, die Booths Algorithmus für zwei 16-Bit-Ganzzahlen implementiert, in Analogie zu der auf einer der Folien „Multiplikation: Standardalgorithmus“ angegebenen Java-Funktion, die den Standardalgorithmus implementiert!