Rechnersysteme und -netze Wintersemester 2020/2021

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

7. Übungsblatt - zu bearbeiten bis 21.12.2020

Aufgabe 1 Arithmetisch-logische Einheit

In der Vorlesung wurde die arithmetisch-logische Einheit (ALU) der Hack-Architektur aus „The Elements of Computing Systems: Building a Modern Computer from First Principles“ von Noam Nisan und Shimon Schocken betrachtet. Die von dieser ALU zu berechnende Funktion wird durch sechs Steuerbits bestimmt: zx, nx, zy, ny, f und no. Die obere Hälfte der folgenden Tabelle wurde in der Vorlesung erklärt. Erklären Sie nun für die untere Hälfte (analog zur Vorlesung), wie die Steuerbits die entsprechenden Ausgaben bewirken!

These bits instruct how to preset the input x		These bits instruct how to preset the input y		This bit selects between + and &	This bit instructs how to postset the output out	Resulting ALU output
zx	nx	zy	ny	f	no	out=
if zx then x = 0	if nx then x = ~x	if zy then y = 0	if ny then y = ~y	if f then out = x + y else out = x & y	if no then out = ~out	f(x,y) =
1	0	1	0	1	0	0
1	1	1	1	1	1	1
1	1	1	0	1	0	-1
0	0	1	1	0	0	x
1	1	0	0	0	0	y
0	0	1	1	0	1	~x
1	1	0	0	0	1	~y
0	0	1	1	1	1	-x
1	1	0	0	1	1	-y
0	1	1	1	1	1	x+1
1	1	0	1	1	1	y+1
0	0	1	1	1	0	x-1
1	1	0	0	1	0	y-1
0	0	0	0	1	0	x+y
0	1	0	0	1	1	x-y
0	0	0	1	1	1	y-x
0	0	0	0	0	0	x&y
0	1	0	1	0	1	x|y

Legen Sie besonderen Wert auf die Erklärung der Subtraktion! Denn man beachte: Diese arithmetisch-logische Einheit implementiert offenbar die Subtraktion *nicht*, wie an früherer Stelle in der Vorlesung gezeigt, durch eine Kette von XOR-Gattern (zur bedingten Berechnung des Einerkomplementes) und die Eingabe eines Übertrags (*carry*) für die unterste Binärstelle (zur Addition von 1, um das Zweierkomplement zu erhalten), sondern auf andere Weise. Was ist das Prinzip dieser Berechnung?

Aufgabe 2 Arithmetisch-logische Einheit

Geben Sie fünf verschiedene Belegungen der Steuerbits der Hack-ALU an, die als Ausgabe -1 erzeugen (und zwar unabängig von den Eingaben x und y der ALU)!
(Zur Bedeutung der Steuerbits zx, nx, zy, ny, f und no siehe die Tabelle in Aufgabe 1.)

Zusatzaufgabe Implementierung der Multiplikation

Die arithmetisch-logische Einheit (ALU) des Hack-Systems ist sehr einfach und in ihren Funktionen sehr eingeschränkt. Sie kann z.B. keine Multiplikation von Binärzahlen, ja nicht einmal eine Multiplikation mit einer Zweierpotenz direkt ausführen (denn es gibt z.B. keine Bit-Schiebe-Operation). Wie kann man mit den Möglichkeiten dieser arithmetisch-logischen Einheit dennoch ein Analogon des Standard-Multiplikationsalgorithmus implementieren? (Als Antwort auf diese Frage ist kein konkretes Programm gesucht, sondern vielmehr ein allgemeines Algorithmenschema, das mit den Operationen der Hack-ALU auskommt, um eine Multiplikation durchzuführen.)

Hinweise: Braucht man zur Implementierung des Standardalgorithmus zwingend eine Operation ASR (*arithmetic shift right*, arithmetisches Bit-Schieben nach rechts)? Könnte man stattdessen auch mit einer Bit-Schiebe-Operation nach links arbeiten? Wodurch kann man eine Bit-Schiebe-Operation nach links ersetzen, wenn sie nicht direkt verfügbar ist (wie hier der Fall)? Wie geht man bei diesem (abgeänderten) Algorithmus am besten mit negativen Zahlen um?

Aufgabe 3 Bistabile Kippstufen/Riegel

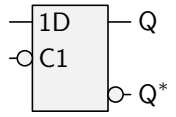
a) Eine bistabile Kippstufe, speziell ein SR-Riegel, kann auf verschiedene Weise implementiert werden. Beschreiben Sie die Unterschiede zwischen einer Implementierung mit NOR- und einer mit NAND-Gattern!

b) Welche Eingaben führen bei einer bistabilen Kippstufe (SR-Riegel) zu einem metastabilen Zustand? Sind dies bei der Variante mit NOR- und der mit NAND-Gattern die gleichen Eingaben? Wieso ist der metastabile Zustand problematisch?

c) Nennen Sie (mindestens) zwei Möglichkeiten, um bei einer bistabilen Kippstufe mit dem metastabilen Zustand umzugehen!

Aufgabe 4 Bistabile Kippstufen/Riegel

Wir betrachten in dieser Aufgabe einen getakteten D-Riegel (siehe Schaltzeichen rechts) und wollen uns u.a. den zeitlichen Ablauf einer Zustandsänderung dieses Riegels genauer klar machen.



a) Wann reagiert der gezeigte D-Riegel auf eine Änderung der Eingabe D?

b) Geben Sie eine Implementierung eines solchen D-Riegels durch Gatter an!

c) Geben Sie für die von Ihnen in Teilaufgabe b) gewählte Implementierung die logischen Funktionen für die Ausgänge Q und Q^* an! Beachten Sie, daß Sie dafür auch den vorangehenden Zustand Q_0 bzw. Q_0^* benötigen, also z.B.: $Q = f(D, C, Q_0, Q_0^*)$.

d) Geben Sie die Wahrheitstafeln für Q und Q^* an! (Auch hier benötigen Sie wieder den vorangehenden Zustand Q_0 bzw. Q_0^*.) Gilt stets $Q^* = \overline{Q} = \neg Q$?

e) Die in Teilaufgabe d) aufgestellte Wahrheitstafel beschreibt, welche Ausgaben sich direkt (nach einer Gatterlaufzeit) einstellen. Da diese Ausgaben rückgekoppelt werden, ist die Zustandsänderung des Riegels damit aber u.U. noch nicht abgeschlossen. Erweitern Sie daher die Wahrheitstabelle um die Folgezustände Q_2 und Q_2^*, die sich

(nach zwei Gatterlaufzeiten) ergeben, wenn man die logischen Funktionen aus Teilaufgabe a) bzw. die Wahrheitstafel aus Teilaufgabe b) auf die neuen Werte $\mathsf{Q}_1 = \mathsf{Q}$ und $\mathsf{Q}_1^* = \mathsf{Q}^*$ anwendet! Gilt stets $\mathsf{Q}_2^* = \overline{\mathsf{Q}}_2 = \neg\mathsf{Q}_2$?