Rechnersysteme und -netze Winter 2017/2018
Christian Borgelt, Christoph Doell,
Heiko Dreyer, Alice Hildebrand, Josua Sattler, Simon Suckut

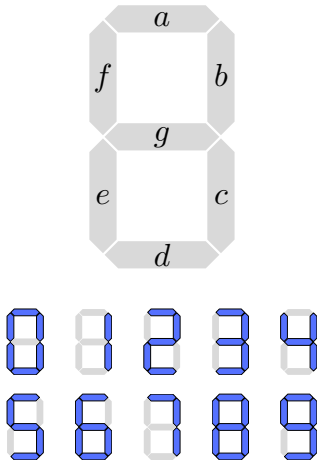
3. Übungsblatt

Aufgabe 1 Minimierung: Karnaugh–Veitch-Diagramme

7-Segment-Anzeigen können zur Darstellung von Dezimalzahlen eingesetzt werden. Um die Segmente korrekt anzusprechen, wird ein sogenannter Decoder benötigt, der die binär kodierten Dezimalzahlen, die an den vier Eingabeleitungen (A, B, C, D) anliegen, in die Ansteuerung der 7-Segment-Anzeige umwandelt. Die Wahrheitstafel für einen solchen Decoder ist gegeben als ("×" in der Wahrheitstafel entspricht "don't care"):

Eingabe				Segmente						
D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	×	×	×	×	×	×	×
1	0	1	1	×	×	×	×	×	×	×
1	1	0	0	×	×	×	×	×	×	×
1	1	0	1	×	×	×	×	×	×	×
1	1	1	0	×	×	×	×	×	×	×
1	1	1	1	×	×	×	×	×	×	×

7-Segment-Digitalanzeige



a) Stellen Sie die Booleschen Funktionen für die Segmente c und e auf!

b) Minimieren Sie die Funktionen aus a) mit Hilfe Boolescher Algebra!

c) Minimieren Sie die Funktionen aus a) mit Hilfe von Karnaugh–Veitch-Diagrammen!

d) Zeichnen Sie die Schaltungen für die optimierten Funktionen aus b) und c)!

Aufgabe 2 Minimierung: Quine–McCluskey-Algorithmus

Minimieren sie die Boolesche Funktion (disjunktive Normalform)

$$\begin{aligned} f(A,B,C,D) \;=\; & \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} \vee \overline{A}\,\overline{B}\,\overline{C}D \vee \overline{A}\,\overline{B}C\overline{D} \vee \overline{A}B\overline{C}\,\overline{D} \vee \overline{A}BCD \\ \vee\; & AB\overline{C}\,\overline{D} \vee AB\overline{C}D \vee ABCD \vee ABC\overline{D} \vee A\overline{B}CD \end{aligned}$$

mit Hilfe des Quine–McCluskey-Algorithmus!

a) Wie viele Primimplikanten gibt es und welche sind dies?

b) Wie viele *wesentliche* Primimplikanten gibt es und welche sind dies?

c) Reichen die *wesentlichen* Primimplikanten aus, um die Funktion darzustellen? Begründen Sie Ihre Antwort!

d) Mit welchen Einsichten kann man es in diesem Fall ggf. vermeiden, Petricks Algorithmus ausführen zu müssen?

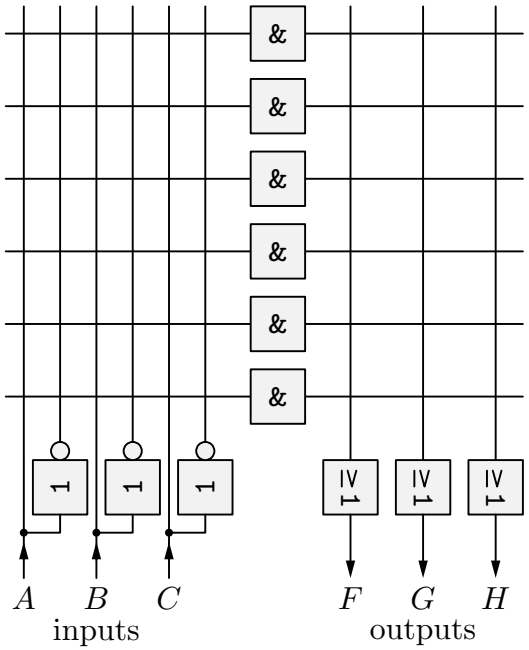
Aufgabe 3 Programmierbare Logikarrays (PLA)

Realisieren Sie die folgenden Funktionen in einem programmierbaren Logikarray (PLA, siehe unten). Markieren Sie die Kreuzungspunkte in den beiden Matrizen (AND-Gitter und OR-Gitter), die verbunden werden müssen.

a) $F = \overline{A}C \vee AB$

b) $G = \overline{A}C \vee \overline{B}C \vee A$

c) $H = A \vee BC \vee \overline{C}$



Aufgabe 4 Hardware Description Language (HDL)

Alle zur Arbeit mit HDL nötigen Informationen finden Sie auf

`http://www.nand2tetris.org/chapters`

(lesen Sie insbesondere die Anhänge A und B) sowie die zugehörige Software auf

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

a) Implementieren Sie die primitiven Gatter NOT, AND und OR lediglich mit Hilfe von NAND-Gattern in HDL.

b) Implementieren Sie NOR und einen Multiplexer in HDL.

c) Schreiben Sie Tests (Appendix B), um Ihre Gatter auf Korrektheit zu überprüfen.

d) Implementieren Sie die Funktion $(a \wedge \overline{b}) \vee (b \wedge c) \vee (\overline{a} \wedge \overline{b})$