Lösungen zum 10. Übungsblatt Informatik II (SS 96)

Jan Niklas Fingerle (zwenti@fsinfo.cs.uni-sb.de)
Michael Mehl (mehl@dfki.uni-sb.de)

Lösung zu Aufgabe 10.1 [Ampel (I)]
Kodierung:

Ampel 1	Ampel2	s_0	s_1	s_2
rot	rot	1	0	1
rot	rot–gelb	1	1	1
rot	grün	1	0	0
rot	gelb	1	1	0
rot–gelb	rot	0	1	1
grün	rot	0	0	1
gelb	rot	0	1	0
weder	noch	0	0	0

Zunächst habe ich die Farbkombinationen geordnet aufgeschrieben. Die Kodierung war dann frei wählbar, daher habe ich versucht, eine zu wählen, die möglichst kurze PAL-Terme erzeugt... Hierbei sei jedoch noch angemerkt, daß eine andere Kodierung die Aufgabe 5 erheblich erleichtern könnte. Näheres findet man bei der Lösung von Aufgabe 5.
Der PAL hat die Eingänge S0, S1 und S2, sowie die Ausgänge R1, R2, GE1, GE2, GR1, GR2. Für einen PAL mit nicht–invertierenden Ausgängen (z.B. 20H8) gelten die linken PAL–Gleichungen. Für einen PAL mit invertierenden Ausgängen (z.B. 20L8) muß man die Signale /R1, /R2, /GE1, /GE2, /GR1, /GR2 berechnen.

R1	=	S0	/R1	=	/S0 * /S1
	+	/S0 * S1 * S2		+	/S0 * /S2
GE1	=	/S0 * S1	/GE1	=	S0
				+	/S1
GR1	=	/S0 * /S1 * S2	/GR1	=	S0
				+	S1
				+	/S2
R2	=	S2	/R2	=	S0 * /S2
	+	/S0 * S1 * /S2		+	/S1 * /S2
GE2	=	S0 * S1	/GE2	=	/S0
				+	/S1
GR2	=	S0 * /S1 * /S2	/GR2	=	/S0
				+	S1
				+	S2

Lösung zu Aufgabe 10.2 [Kontrollogik]
Wichtig bei dieser Betrachtung ist nur noch, ob die Eingangssignale und Ausgangssignale, die man gemeinsam auf einem PAL unterbringen will, auch darauf passen. Die richtige Form der PAL–Gleichungen ist schon im Buch behandelt worden.
Ich habe die Signale in folgender Weise verteilt:

kombinatorische PALs

3 PAL20L8 für ZLS und ZHS. In zwei PALs wird zunächst der Test auf die unteren 12 Bit des jeweiligen Signals durchgeführt, diese Zwischensignale werden dann im dritten PAL verwendet, um mit den restlichen Bits ZL und ZH zu berechnen.
1 PAL20L8: für ALUS0, ALUS1, ALUS2, /CIN, SEXT, FILL, /P2e, /P1e
Dabei ist definiert P2e = E * S1 * /S0; P1e = E * /S1 * S0. Diese Signale werde ich später verwenden. Insgesamt werden nur die Signale I31 – I26, E, S1, S0 verwendet bzw. die invertierte Form. Also brauchen wir neun Eingänge, was ohne Probleme möglich ist.

Register–PALs mit invertierter Clock

1 PAL20R4: für /ALUAdOE, /IAdOE, /PCAdOE
Es werden nur die Signale E, S1, S0 verwendet. Es wäre noch Platz, kombinatorische Signale zu berechnen, jedoch würde damit kein kombinatorischer PAL gespart werden.

Register–PALs mit normaler Clock

1 PAL20R4 für PCLoad. Dieser PAL bräuchte eigentlich 13 Eingänge. Verwendet man jedoch die Hilfsignale P2e und P1e an den Stellen, an denen in der PAL-Gleichung „E * S1 * /S0“ oder „E * /S1 * S0“ steht, so entfallen die Eingänge für E, S1 und S0. Somit benutzen wir nur noch 12 Eingänge
Die verbleibenden Signale werden alle mittels einer Teilmenge von E, S1, S2, I31 – I24 berechnet. Da es sich um 16 Signale handelt, können sie beliebig auf zwei PALs verteilt werden: Ick, ACCck, PCck, IN1ck, IN2ck, /PCLdOE, /IRdOE, /ACCdOE, /IN1LdOE, /IN2LdOE, /0CdOE, /DRdOE, /0RdOE, /ALUDIdOE, /ACCDdOE, /DDIdOE
Es bietet sich an, wenigstens alle Clock–Signale gemeinsam auf einem PAL unterzubringen.

Lösung zu Aufgabe 10.3 [Shift–Befehl]

1. Wir schalten parallel zur ALU einen Barrelshifter, der als Dateneingang den linken Bus hat. Die Ausgaben von ALU und BS werden durch einen MUX ausgewählt, bevor sie durch einen der ALU-Treiber laufen. MUX Eingang 1 kommt von der ALU und der Eingang 0 vom BS.

2. Die Shifthöhe wird direkt mit den unteren 5 Bit des rechten Bus gesteuert. Nun muß noch der MUX gesteuert werden. Dies geschieht mit folgender PAL-Gleichung:

$$\begin{aligned} /Msel &= E * /S1 * /S0 * /I31 * /I30 * /I28 * /I27 * /I26 \\ &+ Msel * S0 * /S1 * E \\ &+ Msel * /S0 * S1 * E \end{aligned}$$

Für Speicher–Compute–Befehle mit F = 000 wird dabei der MUX auch geschaltet; das darf ich, weil dieser Befehl undefiniert ist. Die Halte–Terme entsprechen denen der Treiber.

Lösung zu Aufgabe 10.4 [Register]

1. (a) Für die bisherigen Indexregister muß jeweils ein Tristate–Treiber eingeführt werden, der den Inhalt auf den externen Datenbus treibt, analog zu ACCDd. Diese drei Treiber heissen dann RIDd, wobei I die Nummer des Registers angibt.

   (b) Zunächst müssen ZH2,3–, ZL2,3–Signale berechnet werden, die die bisherigen Indexregister auf den Inhalt Null prüfen. Nun muß die PAL–Gleichung geändert werden, die entscheidet, ob ein neuer Wert in den PC geschrieben wird:

$$
\begin{aligned}
PCload \; := \; & P1e * /I31 * /I30 * /I25 * /I24 \\
+ \; & P1e * /I31 * I30 * /I25 * /I24 \\
+ \; & P1e * I31 * /I30 * I29 * I28 * /I25 * /I24 \\
& \text{; Identisch, da keine JUMP–Befehle} \\
+ \; & P1e * I31 * I30 * I25 * I24 * R131 * I29 \\
+ \; & P1e * I31 * I30 * I25 * I24 * ZL * ZH * I28 \\
+ \; & P1e * I31 * I30 * I25 * I24 * /R131 * /ZH * I27 \\
+ \; & P1e * I31 * I30 * I25 * I24 * /R131 * /ZL * I27 \\
& \text{; I25 * I24 testet auf R1 und R131 ist Bit 31 von R1} \\
& \text{; ansonsten wie im Buch} \\
+ \; & P1e * I31 * I30 * /I25 * I24 * R231 * I29 \\
+ \; & P1e * I31 * I30 * /I25 * I24 * ZL2 * ZP1e * I28 \\
+ \; & P1e * I31 * I30 * /I25 * I24 * /R231 * /ZP1e * I27 \\
+ \; & P1e * I31 * I30 * /I25 * I24 * /R231 * /ZL2 * I27 \\
& \text{; /I25 * I24 testet auf R2 und R231 ist Bit 31 von R2} \\
+ \; & P1e * I31 * I30 * I25 * /I24 * R331 * I29 \\
+ \; & P1e * I31 * I30 * I25 * /I24 * ZL3 * ZH3 * I28 \\
+ \; & P1e * I31 * I30 * I25 * /I24 * /R331 * /ZH3 * I27 \\
+ \; & P1e * I31 * I30 * I25 * /I24 * /R331 * /ZL3 * I27 \\
& \text{; I25 * /I24 testet auf R3 und R331 ist Bit 31 von R3} \\
+ \; & P2e * PCload
\end{aligned}
$$

   Es muß natürlich je nach angegebenem Register auch das richtige Register überprüft werden. Im übrigen ist die PAL–Gleichung analog dem alten PCLoad aufgebaut. Da die PAL–Gleichung zu lang ist, muß man sie zunächst durch Hilfsvariablen in mehrere Teilgleichungen aufspalten und im letzten Schritt „zusammenbauen“.

2. (a) Allgemein:

31 30	29 28 27 26 25 24 23 22	21 0
typ	Spez.	i

(b) Compute:

31 30	29	28 27 26	25	24 23 22	21 0
00	R	F	*	D	i

(c) Load:

31 30	29 28	27 26 25	24 23 22	21 0
01	M	* * *	D	i

(d) Store:

31 30	29 28	27 26 25	24 23 22	21 0
10	M	S	D	i

(e) Jump:

31 30	29 28 27	26 25 24 23 22	21 0
11	C	* * * * *	i

Lösung zu Aufgabe 10.5 [Ampel (II)]

Hier kommen wir nun zu dem Problem, daß bei Aufgabe 1 angesprochen wurde: Da die Farben in einer bestimmten Reihenfolge geschaltet werden, sollte die Kodierung an diese Reihenfolge angelehnt werden:

Ampel 1	Ampel2	s_0	s_1	s_2	t_0	t_1	t_2	S
rot	rot	1	0	1	0	0	0	1
rot	rot–gelb	1	1	1	0	0	1	1
rot	grün	1	0	0	0	1	0	1
rot	gelb	1	1	0	0	1	1	1
rot	rot	1	0	1	1	0	0	1
rot–gelb	rot	0	1	1	1	0	1	1
grün	rot	0	0	1	1	1	0	1
gelb	rot	0	1	0	1	1	1	1
weder	noch	0	0	0	*	*	*	0

Ich habe nun eine neue Numerierung eingeführt, und da es sowieso sinnvolL war, einen neuen PAL zu entwerfen, habe ich S gleich als Eingangssignal für diesen PAL genommen.

Für einen kombinatorischen PAL mit invertierenden Ausgängen erhalten wir also folgende PAL–Gleichungen:

$$
\begin{aligned}
/R1 &= /S + T0 * T1 \\
/R2 &= /S+/T0 * T1 \\
/GE1 &= /S+/T0+/T2 \\
/GE2 &= /S+T0+/T2 \\
/GR1 &= /S+/T0+/T1+T2 \\
/GR2 &= /S+T0+/T1+T2
\end{aligned}
$$

Nun fehlt noch eine Schaltung, mit der die Eingangssignale t_1, t_2 und t_3 berechnet werden können. Dabei muß nur binär gezählt werden, allerdings für die Kombinationen (0;1;0) und (1;1;0) länger gewartet werden. Dazu verwenden wir einen Zähler und einen weiteren, der die Verzögerung bewirkt.