KIT-Fakultät für Informatik
Prof. Dr. Mehdi Tahoori, Prof. Dr. Wolfgang Karl

Musterlösungen zur Klausur

Digitaltechnik und Entwurfsverfahren (TI-1)

und

Rechnerorganisation (TI-2)

am 29. März 2023, 11:00 – 13:00 Uhr

Name:	Vorname:	Matrikelnummer:
Bond	James	007

Digitaltechnik und Entwurfsverfahren (TI-1)	
Aufgabe 1	9 von 9 Punkten
Aufgabe 2	10 von 10 Punkten
Aufgabe 3	5 von 5 Punkten
Aufgabe 4	12 von 12 Punkten
Aufgabe 5	9 von 9 Punkten

Rechnerorganisation (TI-2)	
Aufgabe 6	7 von 7 Punkten
Aufgabe 7	5 von 5 Punkten
Aufgabe 8	15 von 15 Punkten
Aufgabe 9	6 von 6 Punkten
Aufgabe 10	7 von 7 Punkten
Aufgabe 11	5 von 5 Punkten

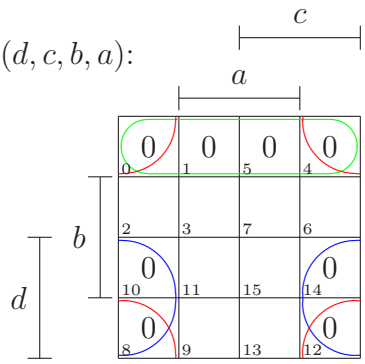
Gesamtpunktzahl:	90 von 90 Punkten

	Note: **1,0**

Aufgabe 1 *Schaltfunktionen* (9 Punkte)

1. 3 P.

$f(d, c, b, a)$:



Die Primimplikate sind:

1. $(d \vee b)$ **2.** $(b \vee a)$ **3.** $(\overline{d} \vee a)$

2. Konjunktive Minimalformen: es existiert nur eine KMF und sie lautet: 1 P.

$$\begin{aligned} y_{KMF} &= \mathbf{1} \wedge \mathbf{3} \\ &= (d \vee b) \wedge (\overline{d} \vee a) \end{aligned}$$

3. Ausgangsgleichung für das Nelson-Verfahren: 1 P.
Die kürzeste Gleichung ist die KMF (siehe Aufgabenteil 1.2)

4. 4 P.

Nr.	gebildet aus	Würfel				gestrichen wegen
1		1	1	–	1	$\subset 6$
2		1	0	–	1	$\subset 6$
3		1	0	–	0	$\subset 9$
4		0	1	1	–	
5		0	0	1	1	$\subset 7$
6	2,1	1	–	–	1	
7	5,4	0	–	1	1	$\subset 10$
8	6,4	–	1	1	1	$\subset 10$
9	6,3	1	0	–	–	
10	7,6	–	–	1	1	

Die Menge der Primimplikanten: $\{\overline{d}cb, \quad da, \quad d\overline{c}, \quad ba\}$

Aufgabe 2 *Schaltnetze und CMOS-Technologie* (10 Punkte)

1 P.

1. y :

$$\begin{aligned} y &= \overline{d}\,\overline{a}\,(1) \vee \overline{d}\,a\,(\overline{c}) \vee d\,\overline{a}(0) \vee d\,a\,(\overline{c} \vee b) \\ &= \overline{d}\,\overline{a} \vee \overline{d}\,a\,\overline{c} \vee d\,a\,\overline{c} \vee d\,a\,b \end{aligned}$$

1 P.

2. Minimalform von y:

$$\begin{aligned} y &= \overline{d}\,\overline{a} \vee \overline{d}\,a\,\overline{c} \vee d\,a\,\overline{c} \vee d\,a\,b \\ &= \overline{d}\,\overline{a} \vee d\,b\,a \vee \overline{c}\,a\,(d \vee \overline{d}) \\ &= \overline{d}\,\overline{a} \vee d\,b\,a \vee \overline{c}\,a \end{aligned}$$

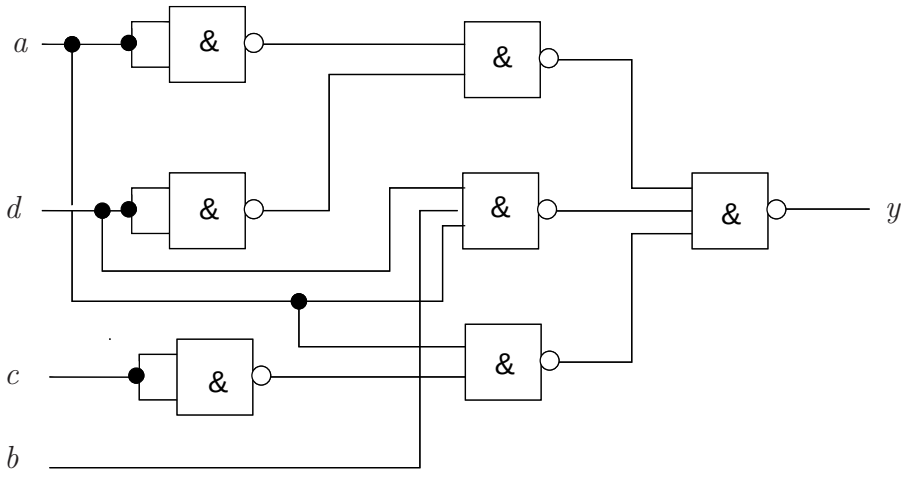
3 P.

3. Minimalform von y in NAND-Form:

$$\begin{aligned} y &= \overline{\overline{\overline{d}\,\overline{a} \;\vee\; d\,b\,a \;\vee\; \overline{c}\,a}} \\ &= \overline{\overline{\overline{d}\,\overline{a}} \;\wedge\; \overline{d\,b\,a} \;\wedge\; \overline{\overline{c}\,a}} \\ &= \mathrm{NAND}_3\left(\mathrm{NAND}_2(\overline{d},\overline{a}),\ \mathrm{NAND}_3(d,b,a), \mathrm{NAND}_2(\overline{c},a)\right) \end{aligned}$$

$$(\overline{x} = x \,\overline{\wedge}\, x)$$

Schaltnetz:

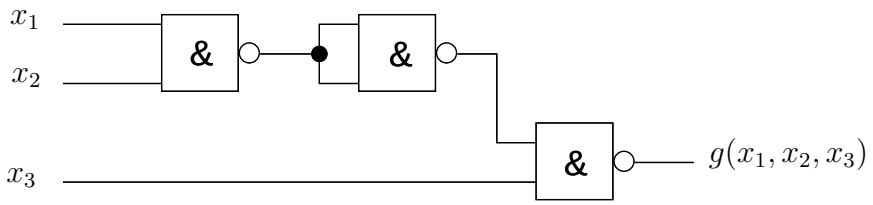


2 P.

4. Realisierung von $g(x_1, x_2, x_3)$ mit NAND-Gattern:

$$\begin{aligned} g(x_1, x_2, x_3) &= \overline{x_1 \ \wedge \ x_2 \ \wedge x_3} \\ &= \overline{(x_1 \ \wedge \ x_2) \ \wedge x_3} \\ &= (x_1 \ \wedge \ x_2) \ \overline{\wedge} \ x_3 \\ &= \overline{\overline{(x_1 \ \wedge \ x_2)}} \ \overline{\wedge} \ x_3 \\ &= \overline{(x_1 \ \overline{\wedge} \ x_2)} \ \overline{\wedge} \ x_3 \end{aligned}$$

Schaltbild:

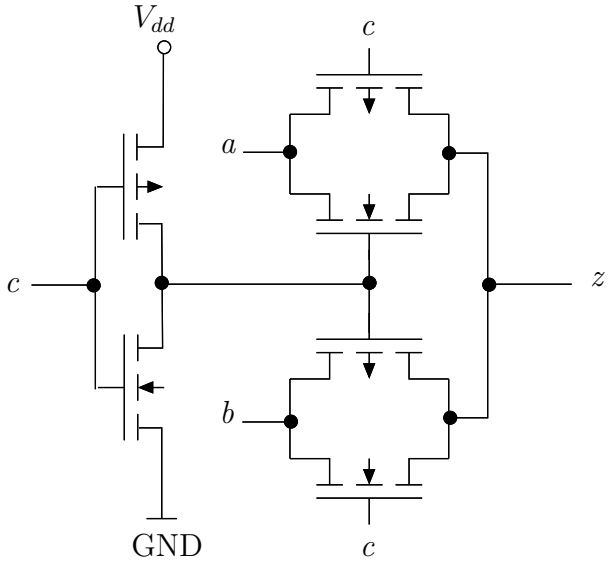


3 P.

5. CMOS-Realisierung eines 2:1-Multiplexers:

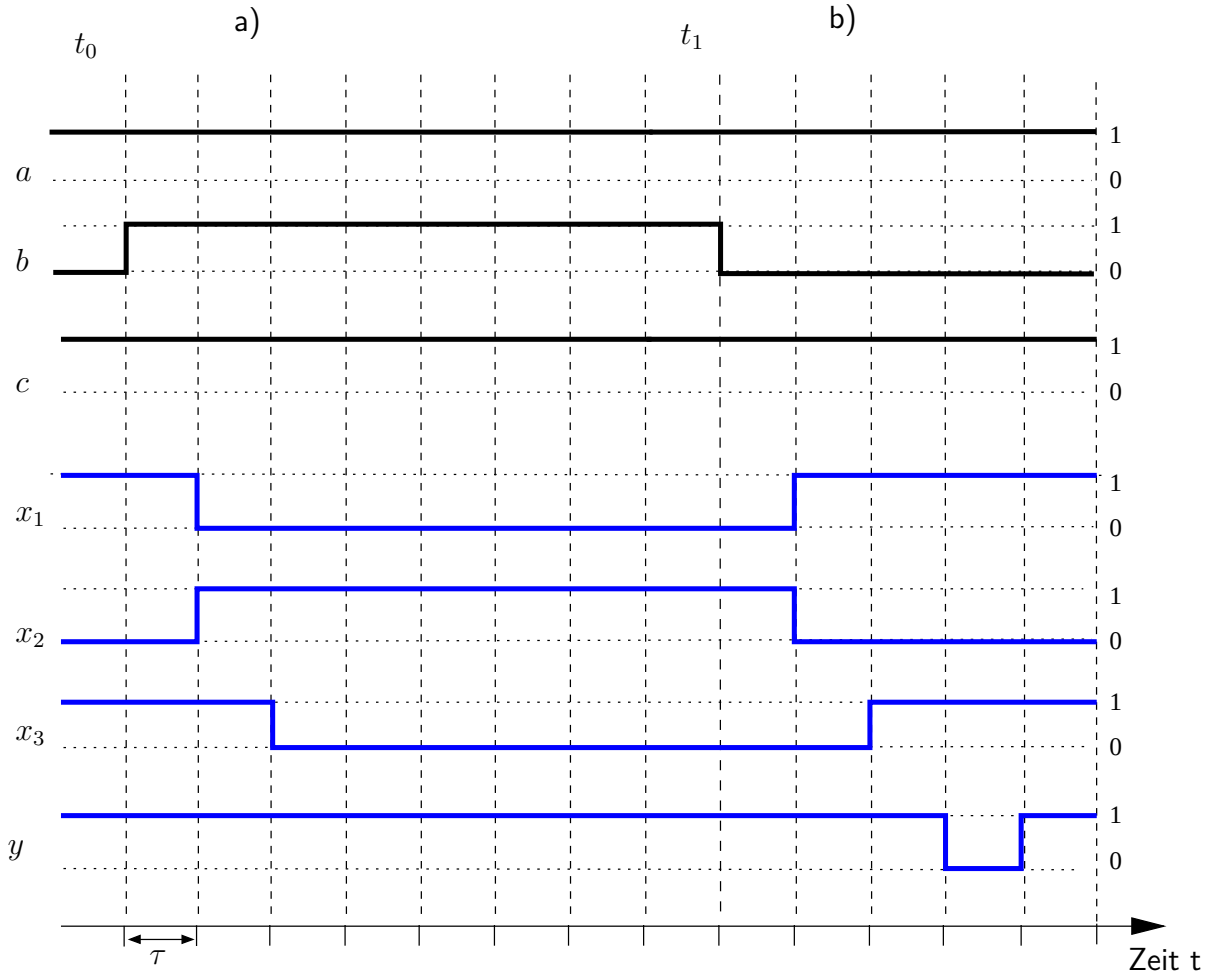
Schaltfunktion: $z(c, b, a) \ = \ \overline{c} \ a \ \vee c \ b$

CMOS-Schaltbild:



Aufgabe 3 *Laufzeiteffekte* (5 Punkte)

1. 3 P.



2. Typ des Fehlers und Behebungsmöglichkeit: 2 P.

   Es tritt ein Hasardfehler beim Übergang $B_7 \rightarrow B_5$ zum Zeitpunkt t_1 auf.

   Es handelt sich hierbei um einen Übergang, bei dem nur eine Variable b ihren Wert wechselt $\Rightarrow$ Der Übergang ist frei von Funktionshasards; der Hasardfehler tritt nicht aufgrund eines Funktionshasards auf und kann nur durch einen Strukturhasard bedingt sein $\Rightarrow$ **1-statischer Strukturhasard**.

   Behebung:

   - Satz von Eichelberger: Realisierung der Schaltfunktion als die Disjunktion aller Primimplikanten (Fehlender Primiplikant $c\,a$ in die Realisierung aufnehmen, d. h. $y = b\,a \vee c\,\bar{b} \vee c\,a$
   - Die beim Übergang konstant bleibenden Eingangsvariablen (a und c) über ein zusätzliches UND-Gatter verknüpfen und das Ergebnis mit dem Ausgang des Schaltnetzes ODER-verknüpfen.

Aufgabe 4 *Schaltwerke* (12 Punkte)

1 P.

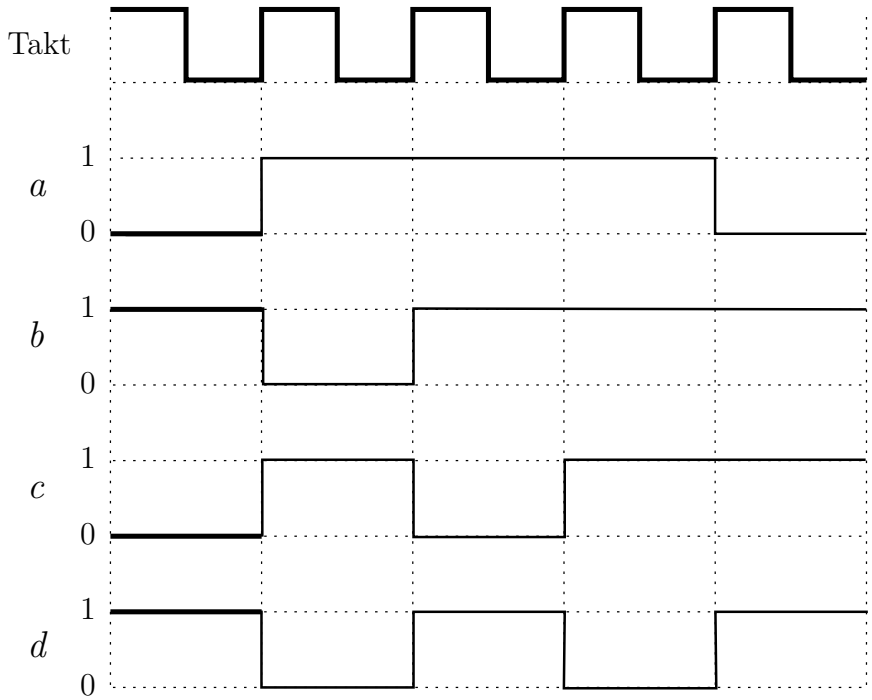
1. (a) Das Schaltwerk ist *synchron*

1 P.

(b) Maximale Anzahl der Zustände ist: $2^4 = 16$ Zustände

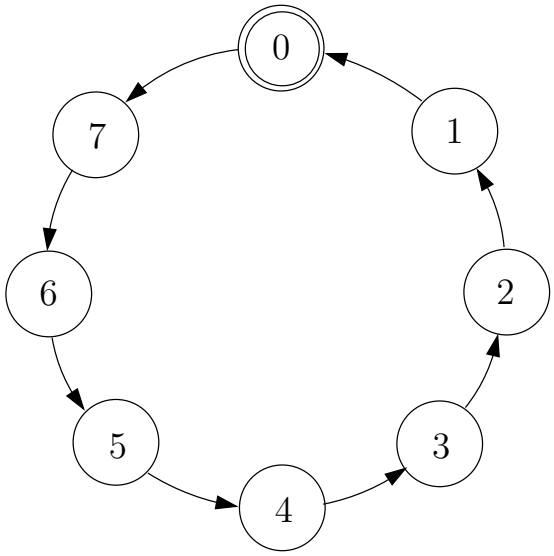
4 P.

(c) Verläufe der Signale a, b, c und d:



2 P.

2. (a) Automatengraph:



1 P.

(b) Kodierte Ablauftabelle:

q_2^t	q_1^t	q_0^t	q_2^{t+1}	q_1^{t+1}	q_0^{t+1}	e_2^t	e_1^t	e_0^t
0	0	0	1	1	1	1	1	1
0	0	1	0	0	0	0	0	1
0	1	0	0	0	1	0	1	1
0	1	1	0	1	0	0	0	1
1	0	0	0	1	1	1	1	1
1	0	1	1	0	0	0	0	1
1	1	0	1	0	1	0	1	1
1	1	1	1	1	0	0	0	1

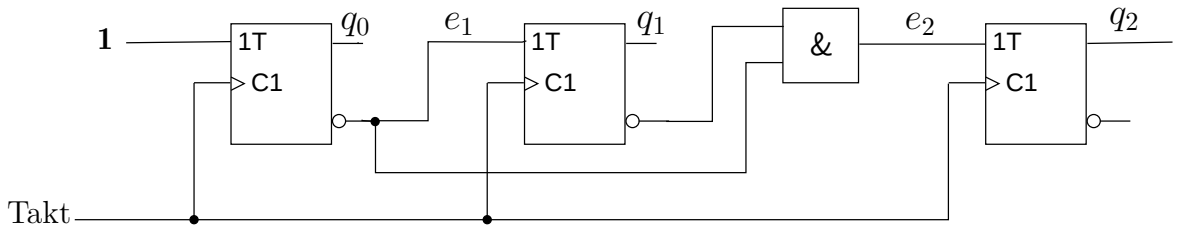
1 P.

(c) Minimalformen der Ansteuerfunktionen der Flipflops: Aus der Ablauftabelle ablesbar.

$$
\begin{aligned}
e_2^t &= \overline{q}_2^t\ \overline{q}_1^t\ \overline{q}_0^t\ \vee\ q_2^t\ \overline{q}_1^t\ \overline{q}_0^t\ =\ \overline{q}_1^t\ \overline{q}_0^t \\
e_1^t &= \overline{q}_0^t \\
e_0^t &= 1
\end{aligned}
$$

2 P.

(d) Schaltbild des Zählers:



Aufgabe 5 *Rechnerarithmetik* (9 Punkte)

1 P.

1. Anzahl der Prüfbits:

   Aufwand: $2^k \geq m + k + 1$. Hier: m $= 200 \Rightarrow k = 8$

2 P.

2. *Carry Ripple*-Addierer und *Carry Lookahead*-Addierer:

   Bei *Carry Ripple*-Addierern muss bei der Addition einer Stelle auf den Übertrag aus den vorhergehenden Stelle gewartet werden. Die Additionszeit ist proportional zur Anzahl der Stellen.

   Bei *Carry Lookahead*-Addierern werden alle Überträge direkt aus den Eingangsvariablen berechnet.

2 P.

3. -70 als 7-Bit-Zweierkomplement Zahl: $+70 = 100\ 0110_2 \Rightarrow$ Es sind mindestens 8 Bit zur Darstellung von -70 als Zweierkomplementzahl notwendig.

   - -70 mit minimaler Bitanzahl: $+70 = 0100\ 0110 \Rightarrow -70 = 1011\ 1001 + 1 = 1011\ 1010$
   - -70 als 16-Bit Zweierkomplementzahl: 1111 1111 1011 1010

4 P.

4. 1000 0011 0101 1000 0000 0000 0000 0101

   (a) BCD: 83 580 005

   (b) Vorzeichenlose Dualzahl: $2^{31} + 2^{25} + 2^{24} + 2^{22} + 2^{20} + 2^{19} + 2^2 + 2^0$

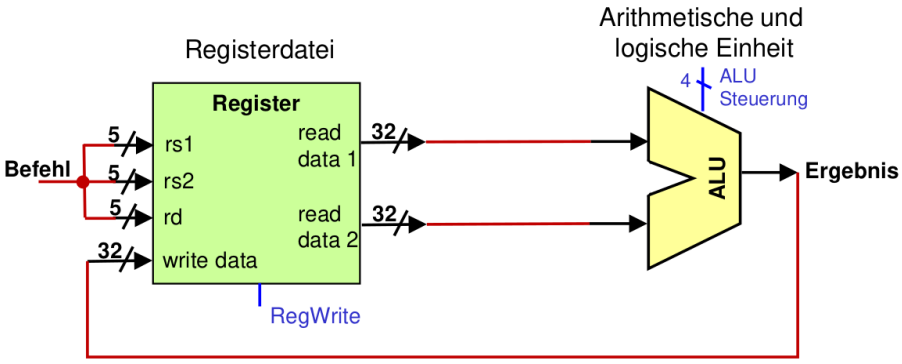
   (c) Gleitkomma-Zahl im IEEE-754-Standard in einfacher Genauigkeit:

$$
\begin{aligned}
VZ &= 1 \\
Char &= 000\ 0011\ 0 = 6 \\
Exp &= Char - 127 = -121 \\
M &= 101\ 1000\ 0000\ 0000\ 0000\ 0101 \Rightarrow \\
\\
Z &= (-1)^1 \cdot (1,101\ 1000\ 0000\ 0000\ 0000\ 0101) \cdot 2^{-121} \\
&= -(1 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-21} + 2^{-23}) \cdot 2^{-121}
\end{aligned}
$$

Aufgabe 6 *RISC-V* (7 Punkte)

3 P.

1. Zeichnung der Hardware-Komponenten:



2. Inhalte der Zielregister: 4 P.

Befehl	Zielregister = (z. B. `x7 = 0x0000 F00A`)
`addi x1, zero, 0x69`	`x1 = 0x0000 0069`
`lui x2, 0x06`	`x2 = 0x0000 6000`
`andi x3, x1, 0x0a`	`x3 = 0x0000 0008`
`srai x4, x2, 8`	`x4 = 0x0000 0060`
`xor x5, x4, x3`	`x5 = 0x0000 0068`
`slt x6, x5, x2`	`x6 = 0x0000 0001`

Aufgabe 7 *MIMA-Architektur* (5 Punkte)

5 P.

1. Takt: IAR $\rightarrow$ SAR; IAR $\rightarrow$ X; R = 1
2. Takt: Eins $\rightarrow$ Y; R = 1
3. Takt: ALU auf Addieren; R = 1
4. Takt: Z $\rightarrow$ IAR
5. Takt: SDR $\rightarrow$ IR

} Lese-Phase

Aufgabe 8 *Cache-Speicher* (15 Punkte)

1 P.

1. (a) Blockgröße in Bytes: 3 Bit Byte-Offset ⇒ Blockgröße = $2^3 = 8$ Byte

2 P.

(b) Cache-Organisation:
12 Bit Index-Feld ⇒ Es lassen sich $2^{12} = 4$ Ki Sätze im Cache adressieren.

$$\text{Assoziativität} = \frac{128\ KiByte}{4\ Ki \cdot 8\ Byte} = 4$$

Der Cache ist als 4-fach assoziativer Speicher (*4-way set associative*) organisiert.

3 P.

2. Speicherbedarf:
Für jede Zeile sind (Tag + 1 Statusbit + Daten pro Zeile) Bits erforderlich.

- Daten pro Zeile 8 Byte × 8 = 64 Bit
- Tag = 32 - 4 - 3 = 25 Bit (4 Bit Satzindex und 3 Bit Byte-Offset)

Speicherbedarf für eine Zeile: $25 + 1 + 64$ Bit = 90 Bits

Speicherbedarf für den gesamten Cache: 90 Bits $\cdot\ 16 \cdot 2\ $ = 2880 Bits = 360 Byte

4 P.

3.

Adresse	**0**	**8**	**40**	**52**	**4**	**8**	**52**	**32**	**2**
read/write	r	r	w	r	r	r	w	w	r
Index	0	2	2	5	1	2	5	0	0
Tag	0	0	1	1	0	0	1	1	0
Byte-Offset	0	0	0	0	0	0	0	0	2
Hit/Miss	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Hit

5 P.

4. Direkt-abgebildeter Cache mit 16 Speicherblöcken:

Adresse	Hilfsspalte (Binär)	Tag	Index	Offset	Hit/Miss
`0x04`	`0b0000 0100`	0	1	0	Miss
`0x34`	`0b0011 0100`	0	d	0	Miss
`0xcf`	`0b1100 1111`	3	3	3	Miss
`0x02`	`0b0000 0010`	0	0	2	Miss
`0x4c`	`0b0100 1100`	1	3	0	Miss
`0xcf`	`0b1100 1111`	3	3	3	Miss
`0x84`	`0b1000 0100`	2	1	0	Miss
`0xb6`	`0b1011 0110`	2	d	2	Miss
`0xb5`	`0b1011 0101`	2	d	1	Hit
`0x07`	`0b0000 0111`	0	1	3	Miss

Aufgabe 9 *Virtuelle Speicherverwaltung* (6 Punkte)

1 P.

1. Unterteilung der virtuellen Adresse:

4 P.

2. Physikalische Adressen:

Virtuelle		Physikalische	
Adresse	Seitennummer	Seitennummer	Adresse
1023	0	3	$3 \cdot 1024 + 1023 = 4095$
1024	1	1	$1 \cdot 1024 + 0 = 1024$
4204	4	2	$2 \cdot 1024 \cdot 108 = 2156$
6200	6	0	$0 \cdot 1024 + 56 = 56$

1 P.

3. Breite des *Tags*:

   Seitengröße ist 4 KiByte $\Rightarrow$ Byte-Offset ist 12 Bit breit.

   Der Tag ist dann $(32 - 12) = 20$ Bits breit

Aufgabe 10 *Pipelining* (7 Punkte)

4 P.

1. Echte Datenabhängigkeiten *(True Dependence)*:

$$
\begin{array}{lll}
S_1 \rightarrow S_3\ (t1) & & \\
S_2 \rightarrow S_3\ (t2) & S_2 \rightarrow S_4\ (t2) & S_2 \rightarrow S_6\ (t2) \\
S_3 \rightarrow S_6\ (t3) & S_3 \rightarrow S_9\ (t3) & \\
S_5 \rightarrow S_7\ (t4) & & \\
S_6 \rightarrow S_8\ (t5) & &
\end{array}
$$

3 P.

2. Behebung der Konflikte:

```
S1:    lw    t1, 100(t0)
S2:    lw    t2, 104(t0)
       NOP
       NOP
S3:    add   t3, t2, t1
S4:    addi  t1, t2, 8
S5:    xori  t4, t0, 7
S6:    andi  t5, t3, t2
       NOP
S7:    sw    t4, 100(t0)
S8:    sw    t5, 104(t0)
S9:    sw    t3, 108(t0)
```

Aufgabe 11 *Verschiedenes* (5 Punkte)

1 P.

1. RISC Befehlssatzarchitekturen versuchen die mittlere Anzahl der Zyklen pro Instruktion CPI auf 1 zu minimieren.

2 P.

2. Komponenten eines allgemeinen Schnittstellenbausteins:

   - Statusregister
   - Steuerregister
   - Befehlsregister
   - Ausführungseinheit
   - Datenbuspuffer
   - Steuerwerk

1 P.

3. Aufgaben der Busarbitrierung:

   - Gewährleistet, dass nur eine aktive Komponente die Kontrolle über den Bus besitzt
   - Priorisierung der zu einem Zeitpunkt von mehreren aktiven Komponenten kommenden Anforderungssignale

1 P.

4. Hauptunterschied zwischen PCI und PCI-E:
   PCI ist als Bus und PCI-E als Punkt-zu-Punkt Verbindung implementiert.