KIT-Fakultät für Informatik
Prof. Dr. Uwe Hanebeck, Prof. Dr.-Ing. Jörg Henkel

Musterlösungen zur Klausur

Digitaltechnik und Entwurfsverfahren (TI-1)

und

Rechnerorganisation (TI-2)

am 27. März 2024, 12:00 – 14:00 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	9 von 9 Punkten
Aufgabe 4	6 von 6 Punkten
Aufgabe 5	11 von 11 Punkten

Rechnerorganisation (TI-2)	
Aufgabe 6	12 von 12 Punkten
Aufgabe 7	6 von 6 Punkten
Aufgabe 8	12 von 12 Punkten
Aufgabe 9	10 von 10 Punkten
Aufgabe 10	5 von 5 Punkten

Gesamtpunktzahl:	90 von 90 Punkten

	Note: **1,0**

Aufgabe 1 *Rechnerarithmetik* (9 Punkte)

1. Die Zahl 640_{10} im IEEE–Format: 2 P.

   $640_{10} = 1010000000_2 = 1,010000000 \cdot 2^9$

   Mantisse = (1)0100 00000 Vorzeichen = 0

   Charakteristik = Exponent + 127 = $136_{10} = 10001000_2$

   $$640_{10} \mathrel{\widehat{=}} 0100\ 0100\ 0010\ 0000\ 0000\ 0000\ 0000\ 0000 \mathrel{\widehat{=}} 4420\ 0000_{16}$$

2. Bereiche: Vorzeichen und Exponent 2 P.

   Begründung: -4 ist negative $\Rightarrow$ Das Vorzeichen ändert sich. Die Multiplikation mit einer 2er Potenz ändert nur den Exponenten. Die Mantisse bleibt gleich.

3. Bereiche: Vorzeichen, Exponent und Mantisse 1 P.

   Begründung: Vorzeichen und Exponent (siehe Aufgabenteil 2). Die Mantisse ändert sich am Randbereich. Wenn der Exponent überläuft, wird die Mantisse auf Null gesetzt.

4. Kleinste normalisierte positive Zahl: (minreal) 1 P.

   $$0\underbrace{000\ 0000\ 1}_{Char=1}\underbrace{000\ 0000 \ldots 0000}_{Mantisse} \;=\; +1,0 \cdot 2^{-126}$$

5. Kleinste denormalisierte positive Zahl: 1 P.

   $$0\underbrace{000\ 0000\ 0}_{Char=0}\underbrace{000\ 0000 \ldots 0001}_{Mantisse} \;=\; 2^{-23} \cdot 2^{-126} \;=\; 2^{-149}$$

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

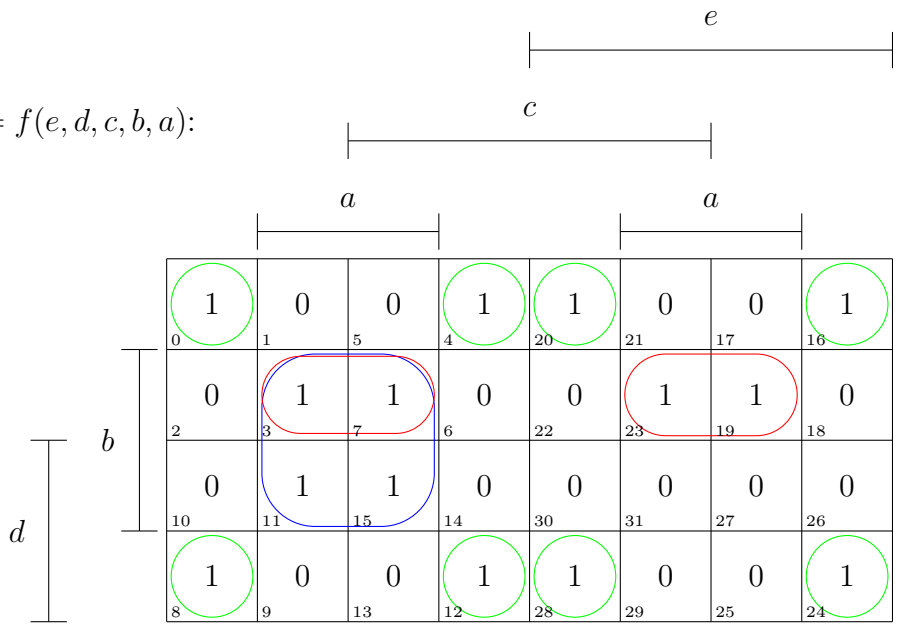
   Bei *Carry Ripple*-Addierern muss bei der Addition einer Stelle auf den Übertrag aus den vorhergehenden Stelle gewartet werden. Die Additionszeit ist proportional zur Anzahl der Stellen.

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

Aufgabe 2 *Schaltfunktion* (10 Punkte)

1. KV-Diagramm: 4 P.

$y = f(e, d, c, b, a)$:



Die Primimplikante sind:

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

2. Disjunktive Minimalformen (DMF): 1 P.

$$y = \overline{d}\ b\ a \vee \overline{b}\ \overline{a}\ \vee \overline{e}\ b\ a$$

3. Minimierte Schaltfunktionen: 4 P.

(a) $z_0 = \Big(a \wedge (a \vee b)\Big) \wedge \Big(a \vee (a \wedge b)\Big) = (a) \wedge (a) = a$

(b) $z_1 = \Big((a \vee b) \wedge (a \vee \overline{b})\Big) \vee \Big((b \wedge a) \vee (b \wedge \overline{a})\Big) = \Big(a \vee (b \wedge \overline{b})\Big) \vee \Big(b \wedge (a \vee \overline{a})\Big)$
$= (a \vee 0) \vee (b \wedge 1) = a \vee b$

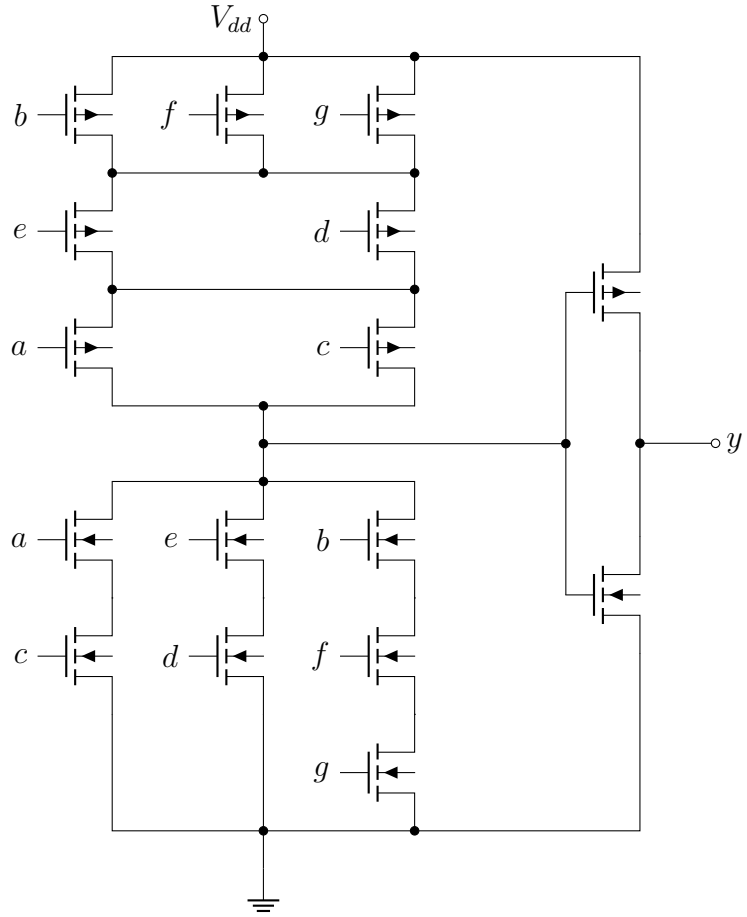
(c) $z_2 = (a \to b) \vee (b \leftarrow a) = a \to b = b \vee \overline{a}$

(d) $z_3 = (a \mathbin{\overline{\vee}} b) \mathbin{\overline{\wedge}} (a \leftrightarrow b) = \overline{(\overline{a \vee b}) \wedge (a \leftrightarrow b)} = \overline{(\overline{a \vee b}) \wedge \Big((a \wedge b) \vee (\overline{a} \wedge \overline{b})\Big)}$
$= (a \vee b) \vee \overline{\Big((a \wedge b) \vee (\overline{a} \wedge \overline{b})\Big)} = (a \vee b) \vee \Big(\overline{(a \wedge b)} \wedge \overline{(\overline{a} \wedge \overline{b})}\Big) = (a \vee b) \vee \Big((\overline{a} \vee \overline{b}) \wedge (a \vee b)\Big)$
$= a \vee b$

4. Dualitätsprinzip: 1 P.
Zu jeder Aussage, die auf die Booleschen Axiome zurückgeführt werden kann, existiert eine duale Aussage, die durch gleichzeitiges Vertauschen der Operationen $\wedge$ und $\vee$ sowie „0 “ und „1 “ entsteht.

Aufgabe 3 *CMOS-Technologie* (9 Punkte)

1. CMOS-Schaltnetz: 6 P.

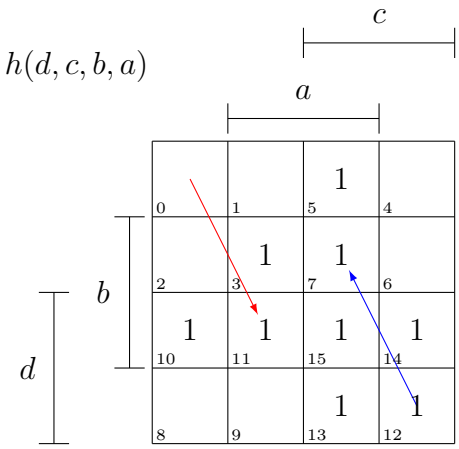


2. Schaltfunktion z: 3 P.

$$z = \overline{f\,a \vee f\,e\,d\,b \vee f\,e\,d\,c \vee g\,e\,d\,a \vee g\,b \vee g\,c}$$

Aufgabe 4 *Laufzeiteffekte* (6 Punkte)

1. 2 P.



Übergang $(0,0,0,0) \rightarrow (1,0,1,1)$:

Übergang mit 3-Variablen-Wechsel $\Rightarrow$ $3! = 6$ Wege. Alle zugehörigen Folgen der Funktionswerte sind monoton $\Rightarrow$ Übergang ist frei von Funktionshazards.

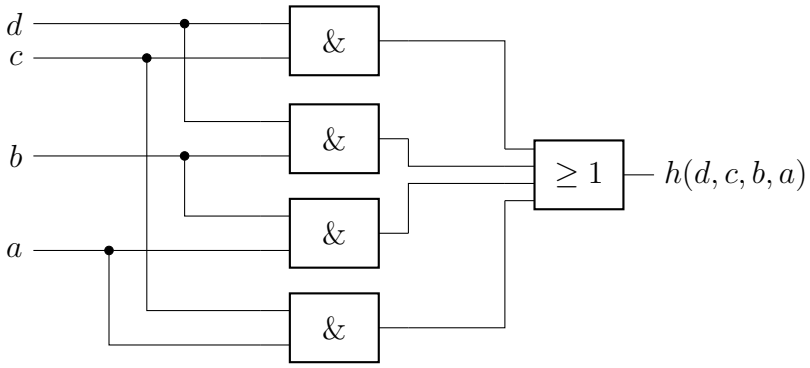
Übergang $(1,1,0,0) \rightarrow (0,1,1,1)$:

Übergang mit 3-Variablen-Wechsel $\Rightarrow 3! = 6$ Wege. Es existiert mindestens eine nicht-monotone Folge der Funktionswerte (z.B. $B_{12} \rightarrow B_{14} \rightarrow B_6 \rightarrow B_7$) $\Rightarrow$ Übergang ist mit einem statischen 1-Funktionshazard behaftet.

2. Strukturhazardfreie Realisierung: 4 P.

Disjunktion aller Primimplikanten oder Konjunktion aller Primimplikate (Satz von Eichelberger)

$$h(d,c,b,a) \;=\; d\,c \;\vee\; d\,b \;\vee b\,a \;\vee\; c\,a$$



Aufgabe 5 *Schaltwerke* (11 Punkte)

1. + 2. Kodierte Ablauftabelle mit Ansteuervariablen: 5 P.

q_2^t	q_1^t	q_0^t	e	q_2^{t+1}	q_1^{t+1}	q_0^{t+1}	r_2	s_2	r_1	s_1	r_0	s_0
0	0	0	0	0	1	0	-	0	0	1	-	0
0	0	0	1	1	1	1	0	1	0	1	0	1
0	0	1	0	0	1	1	-	0	0	1	0	-
0	0	1	1	1	1	0	0	1	0	1	1	0
0	1	0	0	1	0	0	0	1	1	0	-	0
0	1	0	1	0	0	0	-	0	1	0	-	0
0	1	1	0	1	0	1	0	1	1	0	0	-
0	1	1	1	0	0	1	-	0	1	0	0	-
1	0	0	0	1	1	0	0	-	0	1	-	0
1	0	0	1	0	1	0	1	0	0	1	-	0
1	0	1	0	1	1	1	0	-	0	1	0	-
1	0	1	1	0	1	1	1	0	0	1	0	-
1	1	0	0	0	0	1	1	0	1	0	0	1
1	1	0	1	1	0	0	0	-	1	0	-	0
1	1	1	0	0	0	0	1	0	1	0	1	0
1	1	1	1	1	0	1	0	-	1	0	0	-

3. Ansteuerfunktionen der Flipflops: 3 P.

r_2:

$q_2 q_0 \backslash q_1 e$	00	01	11	10
00	-	0	-	0
01	-	0	-	0
11	0	1	0	1
10	0	1	0	1

$\Rightarrow q_2\overline{q}_1 e \vee q_2 q_1 \overline{e}$

s_2:

$q_2 q_0 \backslash q_1 e$	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	-	0	-	0
10	-	0	-	0

$\Rightarrow \overline{q}_2\overline{q}_1 e \vee \overline{q}_2 q_1 \overline{e}$

r_1:

$q_2 q_0 \backslash q_1 e$	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

$\Rightarrow q_1$

s_1:

$q_2 q_0 \backslash q_1 e$	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	1	1	0	0
10	1	1	0	0

$\Rightarrow \overline{q}_1$

r_0:

$q_2 q_0 \backslash q_1 e$	00	01	11	10
00	-	0	-	-
01	0	1	0	0
11	0	0	0	1
10	-	-	-	0

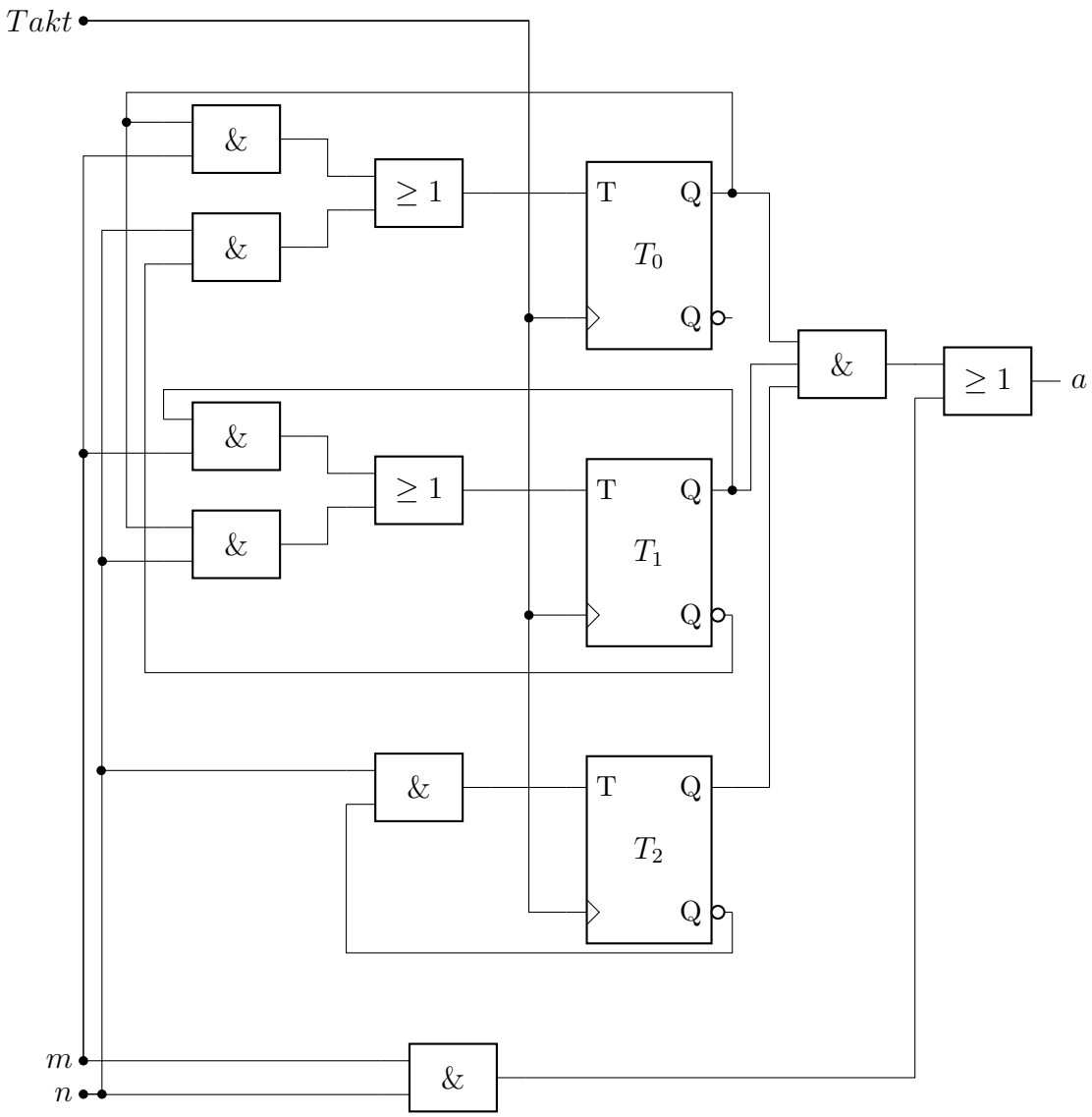
$\Rightarrow \overline{q}_2\overline{q}_1 q_0 e \vee q_2 q_1 q_0 \overline{e}$

s_0:

$q_2 q_0 \backslash q_1 e$	00	01	11	10
00	0	1	0	0
01	-	0	-	-
11	-	-	-	0
10	0	0	0	1

$\Rightarrow \overline{q}_2\overline{q}_1\overline{q}_0 e \vee q_2 q_1 \overline{q}_0 \overline{e}$

4. Realisierung synchrones Schaltwerk: 3 P.



Aufgabe 6 *RISC-V Assembler* (12 Punkte)

1. (a) `do-while`-Schleife: 3 P.

```
do:   add s2, s2, s3
      add t1, zero, s2
      add t1, t1, t1
      add t1, t1, t1
      add t1, t1, s5
      lw  t0, 0(t1)
      beq t0, s4, do
```

(b) `if-else`-Anweisungen: 5 P.

```
      bne s2, s3, else1
      add t1, zero, s4
      add t1, t1, t1
      add t1, t1, t1
      add t1, t1, s5
      lw  t0, 0(t1)
      add s2, s2, t0
      j fertig
else1: bne s2, s4, else2
      add t1, zero, s3
      add t1, t1, t1
      add t1, t1, t1
      add t1, t1, s5
      lw  t0, 0(t1)
      add s2, s2, t0
      j fertig
else2: add s2, s3, s4
fertig:
```

2. Unterschied Maschinensprache und Assemblersprache: 1 P.
   Maschinensprache ist eine Repräsentation von Anweisungen, die für einen Mikroprozessor unmittelbar verständlich sind.
   Assemblersprache ist eine symbolische Repräsentation der Maschinensprache, die für den Menschen verständlich (und anschaulich) ist

3. `.space 2`: 1 P.
   Allokiert 2 Bytes im aktuellen Datensegment.

4. RISC-V Befehlssatz Instruktionen: 2 P.
   Nein.

   Der Befehlssatz kann aus mehr als 128 Befehlen bestehen. Durch den OpCode (Bits $6 \ldots 0$) sind Gruppen von Befehlen definiert (z.B. Speicheroperationen, arithmetische Operationen, ...). Innerhalb dieser Gruppen werden zusätzliche Bitfelder im Befehlsformat verwendet, um zwischen verschiedenen Befehlen (Addition, Subtraktion, Multiplikation, ...) der Gruppe zu unterscheiden.

Aufgabe 7 *MIMA-Architektur* (6 Punkte)

1. Register-Transfer von „ADDA“: 6 P.

Lese-Phase

Dekodier-Phase

7. Takt: IR $\rightarrow$ SAR, X; R = 1

8. Takt: EINS $\rightarrow$ Y; R = 1

9. Takt: ALU auf ($C_2C_1C_0 = 001$), R = 1

10. Takt: Z $\rightarrow$ SAR; R = 1

11. Takt: SDR $\rightarrow$ X; R = 1

12. Takt: R = 1

13. Takt: SDR $\rightarrow$ Y

14. Takt: ALU auf ADD ($C_2C_1C_0 = 001$)

15. Takt: Z $\rightarrow$ AKKU

Aufgabe 8 *Pipelining* (12 Punkte)

1. Anzahl Taktzyklen: 1 P.
   Mit 7 Pipelinestufen (k) und 500 Befehlen (n) folgt: $n + k - 1 = 500 + 7 - 1 = 506$

2. Echte Datenabhängigkeiten: 3 P.

```
S1-S3 (t1)
S2-S3 (t2)   S2-S5 (t2)
S3-S4 (t3)
S4-S5 (t3)   S4-S6 (t3)
S5-S7 (t4)
```

3. Unterschied Datenabhängigkeit und Konflikt: 1 P.
   Eine Datenabhängigkeit ist eine Eigenschaft des Programms. Sie kann nicht behoben werden. Diese Abhängigkeiten können zu einem Konflikt führen. Konflikte aufgrund von Datenabhängigkeiten können von der Software oder von der Hardware vermieden werden.

4. Inhalt Register/Speicherstelle 3 P.

Register/Speicherstelle	Inhalt (z. B. `42`)
`t1`	7
`t2`	7
`t3`	47
`t4`	10
`0 (t0)`	5
`4 (t0)`	5

5. Programmstück ohne Konflikte: 4 P.

```
1        S1:   lw  t1, 0(t0)
2        S2:   lw  t2, 4(t0)
3              NOP
4              NOP
5              NOP
6              NOP
7              NOP
8        S3:   add t3, t1, t2
9              NOP
10             NOP
11             NOP
12             NOP
13             NOP
14       S4:   addi t3, t3, 42
15             NOP
16             NOP
17             NOP
18             NOP
19             NOP
20       S5:   add t4, t3, t2
21       S6:   sw t3, 0 (t0)
22             NOP
23             NOP
24             NOP
25             NOP
26       S7:   sw t4, 4 (t0)
27
```

Aufgabe 9 *Caches* (10 Punkte)

1. Mittlere Zugriffszeit t_{access} System A: 2 P.

$$\begin{aligned} t_{access} &= h_{L1} \cdot t_{hit} + m_{L1} \cdot t_{miss} \\ &= 0,8 \cdot 20\ ns + 0,2 \cdot (20\ ns + 100\ ns) \\ &= 16\ ns + 24\ ns \\ &= 40\ ns \end{aligned}$$

2. Allgemeine Formel: 2 P.

$$t_{access} = h_{L1} \cdot t_{hit\_L1} + m_{L1} \cdot \Big(h_{L2} \cdot (t_{hit\_L1} + t_{hit\_L2}) + m_{L2} \cdot (t_{hit\_L1} + t_{hit\_L2} + t_{HS})\Big)$$

3. Mittlere Zugriffszeit t_{access} System B: 2 P.

$$\begin{aligned} t_{access} &= h_{L1} \cdot t_{hit\_L1} + m_{L1} \cdot \Big(h_{L2} \cdot (t_{hit\_L1} + t_{hit\_L2}) + m_{L2} \cdot (t_{hit\_L1} + t_{hit\_L2} + t_{HS})\Big) \\ &= 0,8 \cdot 20\ ns + 0,2 \cdot \Big(0,8 \cdot (20\ ns + 30\ ns) + 0,2 \cdot (20\ ns + 30\ ns + 100\ ns)\Big) \\ &= 16\ ns + 0,2 \cdot \Big(0,8 \cdot 50\ ns + 0,2 \cdot 150\ ns\Big) \\ &= 16\ ns + 0,2 \cdot (40\ ns + 30\ ns) \\ &= 16\ ns + 14\ ns = 30\ ns \end{aligned}$$

4. Länge der Hauptspeicheradresse: 1 P.
   Byteoffset:

$$32 \text{ Byte} = 2^5 \text{ Byte} \Rightarrow 5 \text{ Bit Byteoffset}$$

   Anzahl Zeilen:

$$\#\text{Zeilen} = \frac{64 \text{ KiByte}}{32 \text{ Byte}} = 2 \text{ Ki} = 2048 \Rightarrow 11 \text{ Bit Indexfeld}$$

   Länge der Speicheradresse:

$$\text{Taglänge} + \text{Indexfeld} + \text{Byteoffset} = 24 \text{ Bit} + 11 \text{ Bit} + 5 \text{ Bit} = 40 \text{ Bit}$$

5. Speicherzugriffe: 3 P.

Adresse	1024	1045	1088	65560	65561	65565
Tag	0	0	0	1	1	1
Index	32	32	34	0	0	0
H/M	M	H	M	M	H	H

Aufgabe 10 *Verschiedenes* (5 Punkte)

1. Nachteil variabler Länge: 1 P.
   Decodierung ist schwieriger. Dekodierschaltung komplizierter.

   Befehlsabarbeitung in einer Pipeline ist wesentlich schwieriger, da die Adresse des nächsten Befehls erst nach dem Holen und Dekodieren des aktuellen Befehls berechnet werden kann.

2. TLB: 1 P.
   Translation-Lookaside-Buffer: Ein schneller Cache zur Beschleunigung der Umsetzung virtueller in physikalischer Adressen. Der TLB speichert die zuletzt benutzten Einträge aus dem Seitentabellenverzeichnis und den Seitentabellen. Im Trefferfall muss nicht auf die im Hauptspeicher liegenden Tabellen zugegriffen werden.

3. Vorteile DMA-Controller: 1 P.
   Die Speicherzugriffe für das Holen der Lade-, Speicher- und Schleifenbefehle entfallen, da die Datenübertragung hardwaremäßig ausgeführt wird ⇒ nur zwei (ggf. sogar nur ein) Speicherzugriff/Datum nötig.
   Der Prozessor wird entlastet und kann während des direkten Speicherzugriffs des Controllers andere Aufgaben tun (sofern diese nicht den Systembus benötigen)

4. USB: 2 P.
   USB steht für *Universal Serial Bus*. Das B steht zwar für Bus, allerdings wird eine Baum-Topologie verwendet.