Karlsruher Institut für Technologie

Fakultät für Informatik
Prof. Dr. Jörg Henkel, Prof. Dr. Mehdi B. Tahoori

Musterlösungen zur Klausur

Digitaltechnik und Entwurfsverfahren/Rechnerorganisation

Technische Informatik I/II

am 23. Juli 2015, 14:00 – 16:00 Uhr

Name:	Vorname:	Matrikelnummer:
Bond	James	007

Digitaltechnik und Entwurfsverfahren/TI-1	
Aufgabe 1	10 von 10 Punkten
Aufgabe 2	9 von 9 Punkten
Aufgabe 3	6 von 6 Punkten
Aufgabe 4	11 von 11 Punkten
Aufgabe 5	9 von 9 Punkten

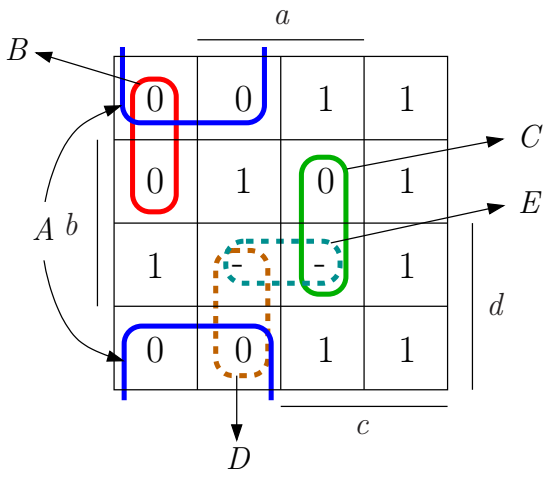
Rechnerorganisation/TI-2	
Aufgabe 6	12 von 12 Punkten
Aufgabe 7	10 von 10 Punkten
Aufgabe 8	12 von 12 Punkten
Aufgabe 9	5 von 5 Punkten
Aufgabe 10	6 von 6 Punkten

Gesamtpunktzahl:	90 von 90 Punkten

	Note: **1,0**

Aufgabe 1

1. $f_1(d, c, b, a)$:



Primimplikate:

$A: \underline{(c \vee b)}$ $\quad B: \underline{(d \vee c \vee a)}$ $\quad C: \underline{(\overline{c} \vee \overline{b} \vee \overline{a})}$

$D: (\overline{d} \vee c \vee \overline{a})$ $\quad E: (\overline{d} \vee \overline{b} \vee \overline{a})$

2. Konjunktive Minimalform von $f_1(d, c, b, a)$:

$$f_1(d, c, b, a) = (c \vee b) \cdot (d \vee c \vee a) \cdot (\overline{c} \vee \overline{b} \vee \overline{a})$$

3. Ausgangsgleichung des Nelson-Verfahrens:

$$f_2(c, b, a) = (b \vee \overline{a}) \cdot (c \vee \overline{a})$$

4. Disjunktive Form von $f_2(c, b, a)$:

$$f_2(c, b, a) = (b \vee \overline{a}) \cdot (c \vee \overline{a}) = bc \vee \overline{a}$$

auch:

$$\begin{aligned} f_2(c, b, a) &= (b \vee \overline{a}) \cdot (c \vee \overline{a}) \\ &= bc \vee b\overline{a} \vee c\overline{a} \vee \overline{a}\,\overline{a} \\ &= bc \vee \overline{a} \end{aligned}$$

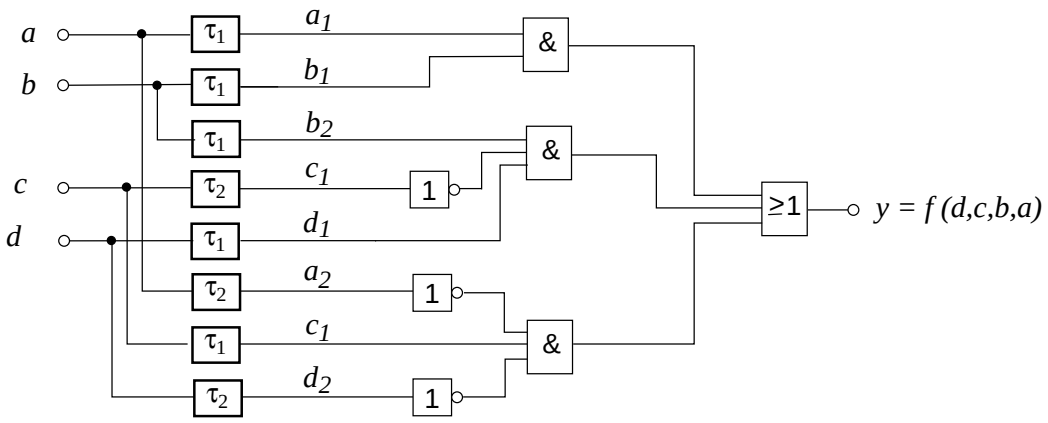
Es gibt keine Terme, die *nur* Freistellen überdecken

5. Vorteile des Consensus-Verfahrens gegenüber dem Quine-McCluskey-Verfahren:

Das Consensus-Verfahren kann mit einer Anfangsüberdeckung, die nicht nur aus Einzelbelegungen besteht, arbeiten. Die Liste aus Würfeln kann durch laufende Streichungen kurz gehalten werden. ⇒ Geringerer Zeit- und Speicheraufwand.

Aufgabe 2

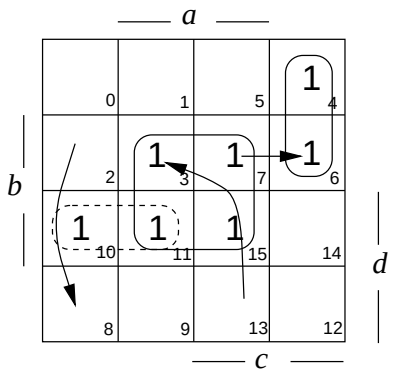
1. Totzeitmodell:



   Dabei ist $\tau_1 = \tau_{AND} + \tau_{OR}$ und $\tau_2 = \tau_{AND} + \tau_{OR} + \tau_{NOT}$.

2. KV-Diagramm:

$$y = f(d, c, b, a) = b\,a \vee d\,\overline{c}\,b \vee \overline{d}\,c\,\overline{a}$$



3. Übergang $(1, 1, 0, 1) \rightarrow (0, 0, 1, 1)$:
   Block $d\,\overline{c}\,b$ ragt in den Übergangsbereich hinein, ohne die Einsbelegung des Übergangs zu überdecken. $\Rightarrow$ Durch Verkleinerung dieses Blockes auf $d\,\overline{c}\,b\,\overline{a}$ wird der Hasard behoben.

4. Übergang $(0, 1, 1, 1) \rightarrow (0, 1, 1, 0)$:
   Satz von Eichelberger: Ergänzung der Funktion durch den Primimplikanten $\overline{d}\,c\,b$, der dann die Anfangs- und Endbelegung des Übergangs überdeckt. $\Rightarrow$ Der Übergang findet innerhalb eines Blockes statt und ist damit hasardfrei.

5. Übergang $(0, 0, 1, 0) \rightarrow (1, 0, 0, 0)$:
   Keine strukturelle Maßnahme ist möglich, da eine zugehörige Folge $(B_2 \rightarrow B_{10} \rightarrow B_8)$ der Funktionswerte zwischen Anfangs- und Endbelegung nicht monoton ist.
   Behebung des *Hasardfehlers* nur mit geeigneter Verzögerung der Eingabewechsel möglich: b muss sich vor d ändern, dann ist die Folge des Funktionswerte $(B_2 \rightarrow B_0 \rightarrow B_8)$ monoton.

Aufgabe 3

1.

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

2.

$$\begin{aligned}
a \leftrightarrow b \leftrightarrow c &= a \leftrightarrow (b \leftrightarrow c) && \text{Assoziativgesetz} \\
&= a \nleftrightarrow \overline{(b \leftrightarrow c)} && (x \leftrightarrow y \;=\; x\,y \;\vee\; \overline{x}\,\overline{y} \;=\; x \nleftrightarrow \overline{y}) \\
&= a \nleftrightarrow (b \nleftrightarrow c) && \text{Inverses Element} \\
&= a \nleftrightarrow b \nleftrightarrow c && \text{Assoziativgesetz}
\end{aligned}$$

Aufgabe 4

1. Dezimalzahl $0.\overline{8}$:
$$\begin{aligned} 0.\overline{8}_{10} &= 0.\overline{111000}_2 \\ 0.\overline{8}_{10} &= 0.\overline{E38}_{16} \\ 0.\overline{8}_{10} &= 0.8_9 \end{aligned}$$

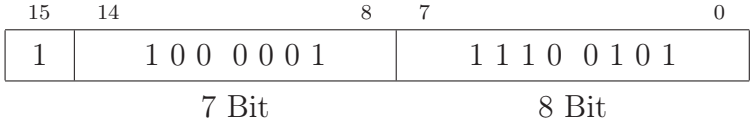
2. Anzahl der Stellen: 6

   Mit n Bit lassen sich Zahlen in Zweierkomplement-Form zwischen $-(2^{n-1})$ und $(2^{n-1}-1)$ darstellen.

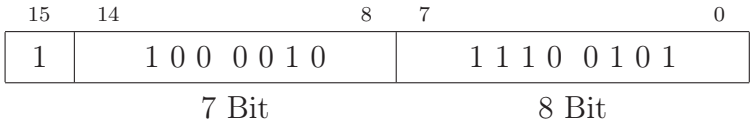
   Für $n = 6$: $-32 \leq Z \leq 31$

3. Multiplikation der Gleitkommazahlen:

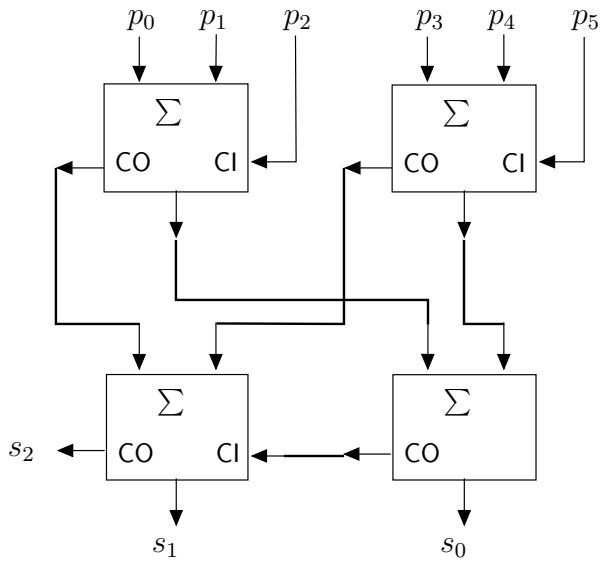
   - $VZ = 0 \not\Leftrightarrow 1 = 1$
   - Charakteristik: $Char_1 + Char_2 - Offset$. $Offset = 64 \rightarrow$
     Charakteristik = 100 0001
     Mantisse: (1) 1 1 1 0 0 1 0 1 1 0 1 1



   - Für $Offset = 63$ (ebenfalls akzeptiert) $\rightarrow$ Charakteristik = 100 0010



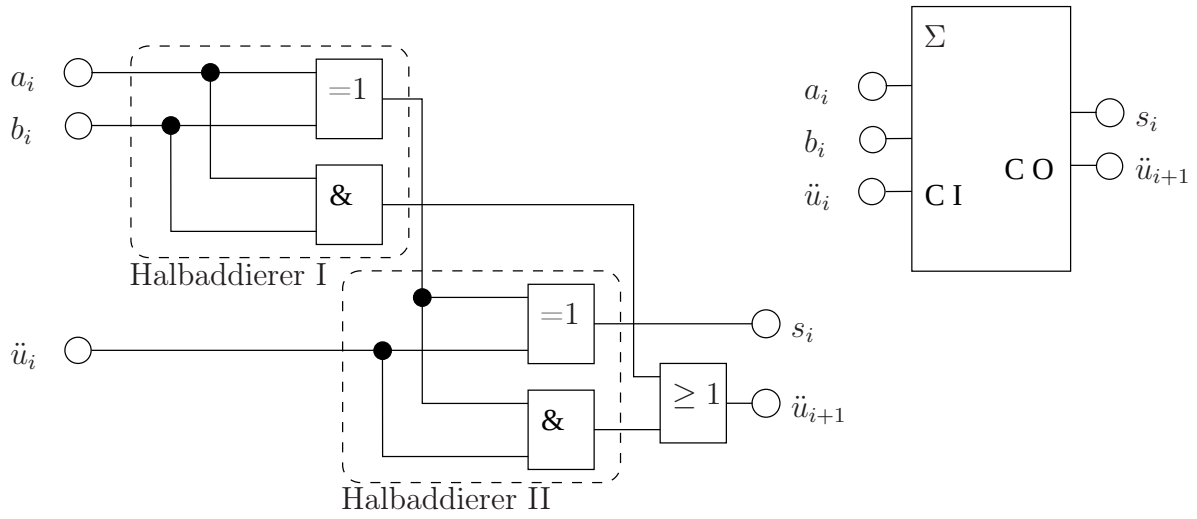
4. Schaltung zur Berechnung der Summe $s_2\, s_1\, s_0$:



Aufgabe 5

1. Unterschied zwischen Halbaddierer und Volladdierer:
   Ein Volladdierer berücksichtigt den Übertrag der vorhergehenden Stellen, deshalb besitzt er, zusätzlich zu den zwei Eingänge für die zu addierenden Dualziffern, einen Eingang für den Übertrag.

2. Schaltbild eines 1-Bit-Volladdierers:



3. Anzahl der Prüfbits:
   Aufwand: $2^k \geq m + k + 1$. Hier: m $= 20 \Rightarrow k = 5$

4. Physikalische Ursache für Hasardfehler:
   Unterschiedliche Laufzeiten der Signale bei Eingabeänderungen aufgrund unterschiedlicher Schaltzeiten der Gatter und Leitungsverzögerungen (unterschiedliche Totzeiten der Signalpfade durch das Schaltnetz).

5. Unterschied zwischen einem PAL-Baustein und PLA-Baustein:
   Bei PAL-Bausteinen ist die ODER-Matrix bereits bei der Herstellung personalisiert, während die UND-Matrix programmierbar ist. Bei PLA-Bausteinen sind sowohl die UND- als auch die ODER-Matrix programmierbar.

6. Schieberegister als:

   - Serien-Parallel-Wandlung
   - Parallel-Serien-Wandlung
   - Warteschlange (FIFO-Speicher) oder Stapelspeicher (LIFO-Speicher)
   - Umlaufspeicher
   - Multiplikation und Division
   - ...

Aufgabe 6

1. Kontrollstruktur in MIPS-Assembler:

```
       addi    $a0, $zero, 100     # i = 100;
loop:  slt     $v0, $zero, $a0     # if 0 < i, dann $v0 = 1
       beq     $v0, $zero, label   # if $v0 = 0 (d.h., i <= 0),
                                   # dann Ende der for-Schleife
       mul     $a1, $a1, $a0       # j = j * i;
       subi    $a0, $a0, 1         # i--;
       b       loop                # gehe zur Marke loop
label:
```

2. Fehlerfreie Version:

```
            add   $v0, $zero, $zero    # summe = 0
add_array:  lw    $t0, 0($a0)          # Nächstes Element lesen
            add   $v0, $v0, $t0        # Addieren
            addi  $a0, $a0, 4          # Zeiger auf das nächste Element
            addi  $a1, $a1, -1         # Zähler dekrementieren
            bgtz  $a1, add_array       # Abbruchbedingung
```

3. Inhalte der Zielregister:

Befehl	Zielregister = (z. B. `$s6 = 0x0000 F00A`)
`subi $s1, $zero, 0x2`	`$s1 = 0xFFFF FFFE`
`srl $s2, $s1, 4`	`$s2 = 0x0FFF FFFF`
`slti $s3, $s2, 100`	`$s3 = 0x0000 0000`
`lui $s4, 0x40`	`$s4 = 0x0040 0000`
`xor $s5, $s1, $s4`	`$s5 = 0xFFBF FFFE`

4. (a) Registerinhalte:

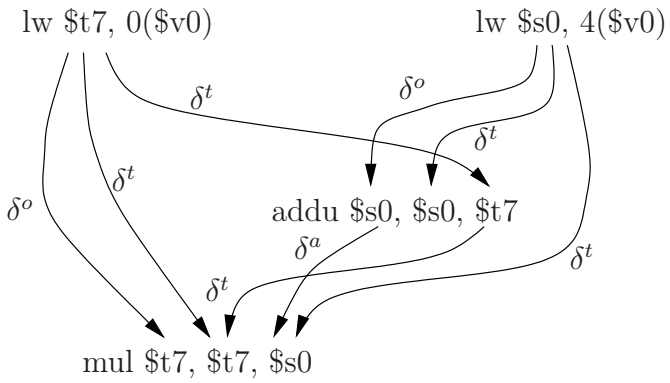
Register	Inhalt
`$t1`	`$t1 = 0x0000 000C`
`$t2`	`$t2 = 0x0000 001F`
`$t3`	`$t3 = 0x0000 0013`
`$t4`	`$t4 = 0x0000 0029`

(b) MIPS-Code zur Speicherung der Adresse von `vec` im Register `$s0`:

```
la $s0, vec
```

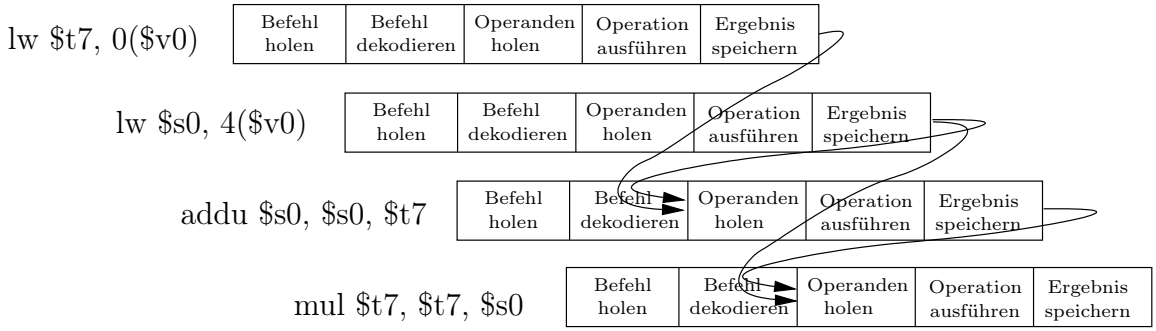
Aufgabe 7

1. Daten– und Kontrollabhängigkeiten:



   Keine Kontrollabhängigkeiten!

2. Pipelinekonflikte: Es gibt vier Pipelinekonflikte.

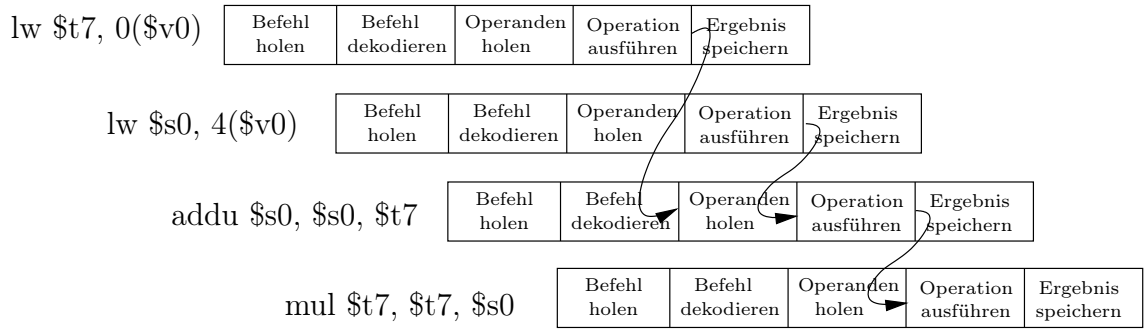


3. Pipelinekonflikte werden von der Hardware nicht erkannt:

```
# In Register $v0 steht die Adresse 0x10008000
lw   $t7, 0($v0)
lw   $s0, 4($v0)
nop
nop
add  $s0, $s0, $t7   # $s0 = $s0 + $t7
nop
nop
mul  $t7, $t7, $s0   # $t7 = $t7 * $s0
```

4. Pipelinekonflikte werden von der Hardware erkannt:

   Keine `nop`-Befehle mehr erforderlich.



Aufgabe 8

1. (a) Blockgröße in Bytes: 6 Bit Byte-Offset ⇒ Blockgröße = $2^6 = 64$ Byte

   (b) Anzahl der Einträge:

$$\text{Anzahl der Einträge} = \frac{\text{Kapazität}}{\text{Blockgröße}} = \frac{128\ KByte}{64\ Byte} = 2\ K \text{ Einträge}$$

   (c) Cache-Organisation:
   11 Bit Index-Feld ⇒ Es lassen sich $2^{11} = 2$ K Zeilen/Sätze im Cache adressieren.

$$\text{Assoziativität} = \frac{2\ K}{2\ K} = 1$$

   Es handelt sich also um einen direkt-abgebildeter Cache-Speicher (*direct mapped cache*).

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

   - Daten pro Zeile 8 Byte × 8 = 64 Bit
   - Tag = 32 - 3 - 3 = 26 Bit (3 Bit Satzindex und 3 Bit Byte-Offset)

   Speicherbedarf für eine Zeile: $26 + 1 + 64$ Bit = 91 Bits

   Speicherbedarf für den gesamten Cache: 91 Bits $\cdot\ 8 \cdot 5\ =$ 3640 Bits = 455 Byte

3.

Adresse	**0**	**16**	**48**	**48**	**56**	**8**	**8**	**56**	**32**	**0**
read/write	r	r	w	r	r	r	r	w	w	r
Index	0	2	2	2	3	1	1	3	0	0
Tag	0	0	1	1	1	0	0	1	1	0
Hit/Miss	Miss	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Hit

Aufgabe 9

1. Hauptaufgaben von Betriebsystemen:

   - Betriebsmittelverwaltung *(resource management)*
   - Auftragsverwaltung *(task management)*
   - Speicherverwaltung *(memory management)*

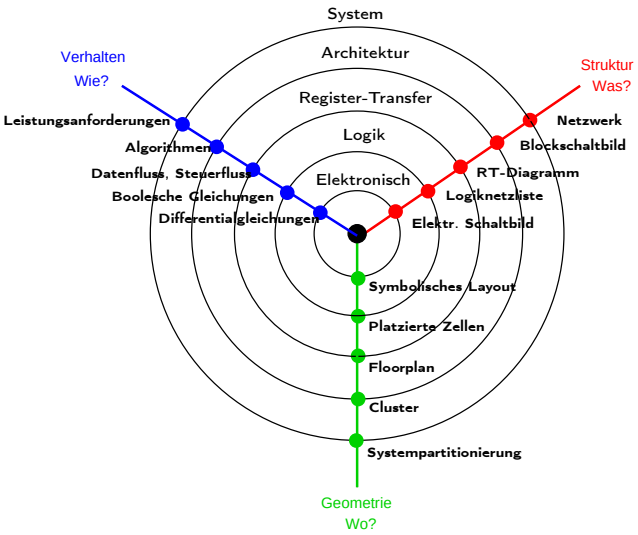
2. Vorteile eines DMA-Controllers:

   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).

3. Y-Diagramm:

   Das Y-Diagramm ist eine Darstellung der unterschiedlichen Ebenen des Entwurfs. Dabei wird der Abstraktionsgrad eines Entwurfs in so genannten Entwurfsebenen dargestellt. Die verschiedenen Repräsentationen eines Entwurfs auf einer Ebene sind als drei prinzipiell unterschiedliche Sichten (funktionell, strukturell, physikalisch) dargestellt.



Aufgabe 10

1.

Speicher-Bausteine	*richtig*	*falsch*
Auch die schnellsten SRAM-Bausteine bremsen die Verarbeitungsgeschwindigkeit moderner Prozessoren.	×	
Bei einem DRAM-Baustein werden die Speicheradressen im Multiplexbetrieb über die gleichen Leitungen dem Baustein zugeführt.	×	
Die Zugriffszeit eines DRAM-Speicher-Bausteins ist kleiner als die Zykluszeit.	×	
Bei FPM-DRAMs wird der Zugriff wesentlich beschleunigt, wenn die zu lesenden oder zu schreibenden Speicherzellen in der gleichen Zeile der Speichermatrix liegen.	×	

2.

Assembler	*richtig*	*falsch*
Pseudobefehle erzeugen keinen Maschinencode.		×
Assemblerdirektiven sind prinzipiell nicht notwendig, da sie durch „echte“ Maschinenbefehle ersetzt werden können.		×
In eingebetteten Systemen mit beschränkten Speicherressourcen wird bevorzugt in Assembler programmiert.	×	
Assemblersprachen sind nicht maschinenspezifisch.		×

3.

Virtuelle Speicherverwaltung	richtig	falsch
Das Betriebsystem ist allein zuständig für die Umsetzung virtueller in physikalische Adressen.		×
Die Umsetzung virtueller in physikalische Adressen wird durch spezielle Hardware beschleunigt.	×	
Die Seitengrößen können je nach Anforderung dynamisch wachsen.		×
Interne Fragmentierung ist ein typisches Problem von Segmentierung.		×