



UNIVERSITÄT KARLSRUHE (TH)
Fakultät für Informatik
Institut für Rechnerentwurf und Fehlertoleranz (IRF)
Prof. Dr. W. Karl

Musterlösungen
zur Klausur „Technische Informatik I/II“
am 29. Januar 2005, 10.00 - 12.00 Uhr

Name: Bond	Vorname: James	Matrikelnummer: 007
----------------------	--------------------------	-------------------------------

Technische Informatik I	
Aufgabe 1	10 von 10 Punkten
Aufgabe 2	8 von 8 Punkten
Aufgabe 3	11 von 11 Punkten
Aufgabe 4	12 von 12 Punkten
Aufgabe 5	4 von 4 Punkten

Technische Informatik II	
Aufgabe 6	5 von 5 Punkten
Aufgabe 7	12 von 12 Punkten
Aufgabe 8	12 von 12 Punkten
Aufgabe 9	6 von 6 Punkten
Aufgabe 10	10 von 10 Punkten

Gesamtpunktzahl:	90 von 90 Punkten
-------------------------	-------------------

Note:	1,0
--------------	------------

Aufgabe 1

1. (a) DMF:

$$y_{DMF} = c b a \vee \bar{c} \bar{b}$$

				a			
				1	1	0	-
b	0	0	1				
	0	0	1				
	1			-	0	0	
				c			
				d			

(b) KMF:

$$y_{KMF} = (c \vee \bar{b}) \cdot (\bar{c} \vee b) \cdot (\bar{b} \vee a) \quad \text{oder}$$

$$y_{KMF} = (c \vee \bar{b}) \cdot (\bar{c} \vee b) \cdot (\bar{c} \vee a)$$

				a			
				1	1	0	-
b	0	0	1				
	0	0	1				
	1			-	0	0	
				c			
				d			

- (a)
- Kernprimimplikanten: A, C, D und F
 - Wahlprimimplikanten: Keine
 - Entbehrliche Primimplikanten: B und E

(b) DMF:

$$\begin{aligned}
 h &= A \vee C \vee D \vee F \\
 &= \bar{e} \bar{d} \bar{b} \vee \bar{e} \bar{d} c \vee \bar{e} c a \vee c b a
 \end{aligned}$$

2. Beweis:

Absorptionsgesetze: $x = x(x \vee w)$ und $\bar{x} = \bar{x}(\bar{x} \vee u)$

Einsetzen:

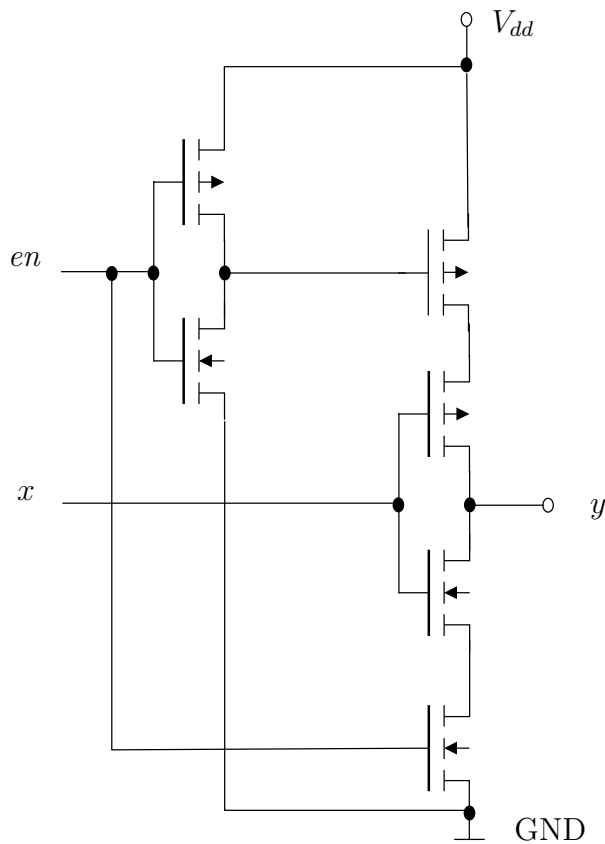
$$\begin{aligned} x u \vee \bar{x} w &= x(x \vee w)u \vee \bar{x}(\bar{x} \vee u)w \\ &= x u \vee x w u \vee \bar{x} w \vee \bar{x} u w \\ &= x u \vee \bar{x} w \vee (x \vee \bar{x}) u w \\ &= x u \vee \bar{x} w \vee u w \end{aligned}$$

Aufgabe 2

1. Schaltfunktion $g(d, c, b, a)$:

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

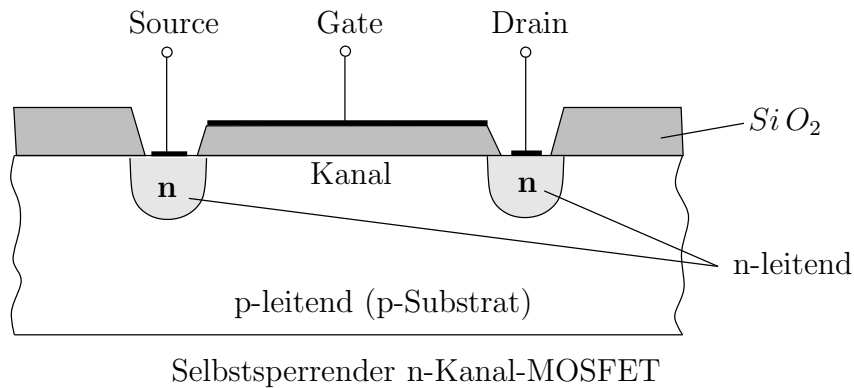
2. CMOS-Tristate-Inverter:



Funktionstabelle:

en	x	y
0	-	hochohmig
1	0	1
1	1	0

3. Aufbau eines nMOS-Transistors:



Aufgabe 3

1. Kodierte Ablaufabelle:

- Ansteuergleichungen und Ausgabefunktion: Ablesen aus dem Schaltbild

$$\begin{aligned}
 j_B^t &= \overline{A}^t \leftrightarrow x^t & k_B^t &= \overline{A}^t \leftrightarrow x^t \\
 j_A^t &= B^t & k_A^t &= x^t \\
 y^t &= \overline{B}^t (\overline{A}^t \leftrightarrow x^t)
 \end{aligned}$$

- Übergangsgleichungen: Die charakteristische Gleichung des JK-Flipflops lautet:

$$q^{t+1} = (j \overline{q} \vee \overline{k} q)^t$$

Mit den Ansteuergleichungen der Flipflops erhält man die Zustandsübergangsgleichungen für A , und B :

$$\begin{aligned}
 B^{t+1} &= (\overline{A}^t \leftrightarrow x^t) \overline{B}^t \vee \overline{(\overline{A}^t \leftrightarrow x^t)} B^t \\
 A^{t+1} &= B^t \overline{A}^t \vee \overline{x} A^t
 \end{aligned}$$

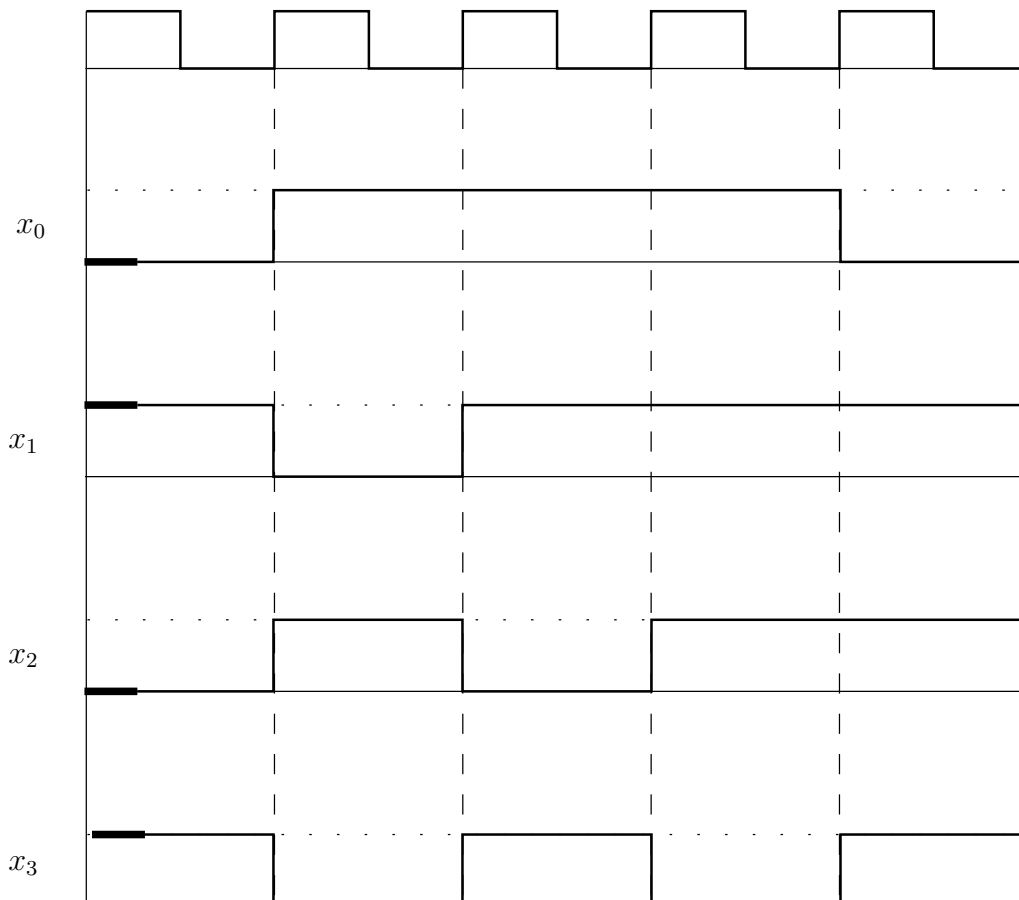
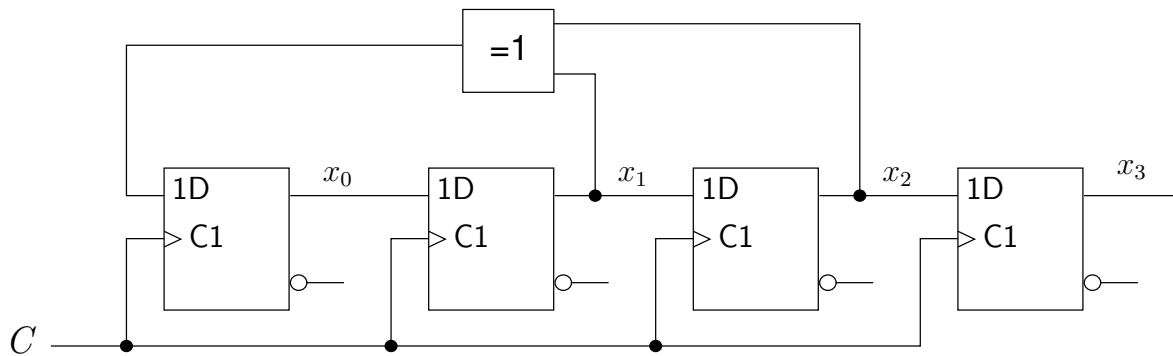
- Kodierte Ablaufabelle:

Zustand		Eingabe	Folgezustand		Ausgabe
A^t	B^t	x^t	A^{t+1}	B^{t+1}	y^t
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	0	1	1
1	1	0	1	1	0
1	1	1	0	0	0

2. Eingabe-, Ausgabe-, und Zustandsfolgen:

t	1	2	3	4	5	6	7	8	9	10
e	1	0	1	1	0	0	1	0	1	1
a	0	0	0	1	1	0	1	1	0	1
Z_i	Z_0	Z_1	Z_2	Z_3	Z_2	Z_0	Z_3	Z_2	Z_0	Z_1

3. Verläufe der Signale x_0, x_1, x_2 und x_3 :



Aufgabe 4

1. Die Basen s und r :

$$1 \cdot r^1 + 2 = 1 \cdot s^2 + 1 \cdot s^1 + 1 \rightarrow r = s^2 + s - 1$$

Es existieren unendlich viele Lösungen:

s	2	3	4	...
r	5	11	19	...

2. (a) 12 binären Stellen: $2^{12} - 1 = 4095$

(b) 4 hexadezimalen Stellen: $16^4 - 1 = 65535$

3. $2005_{10} = 111\ 1101\ 0101_2$

- 32-Bit Zweierkomplement-Format:

0000 0000 0000 0000 0000 0111 1101 0101

- 32-Bit IEEE-754-Gleitkomma-Format:

$111\ 1101\ 0101_2 = 1,11\ 1101\ 0101 \cdot 2^{10}$

$Exp = 10 \Rightarrow Char = Exp + 127 = 137_{10} = 1000\ 1001_2$

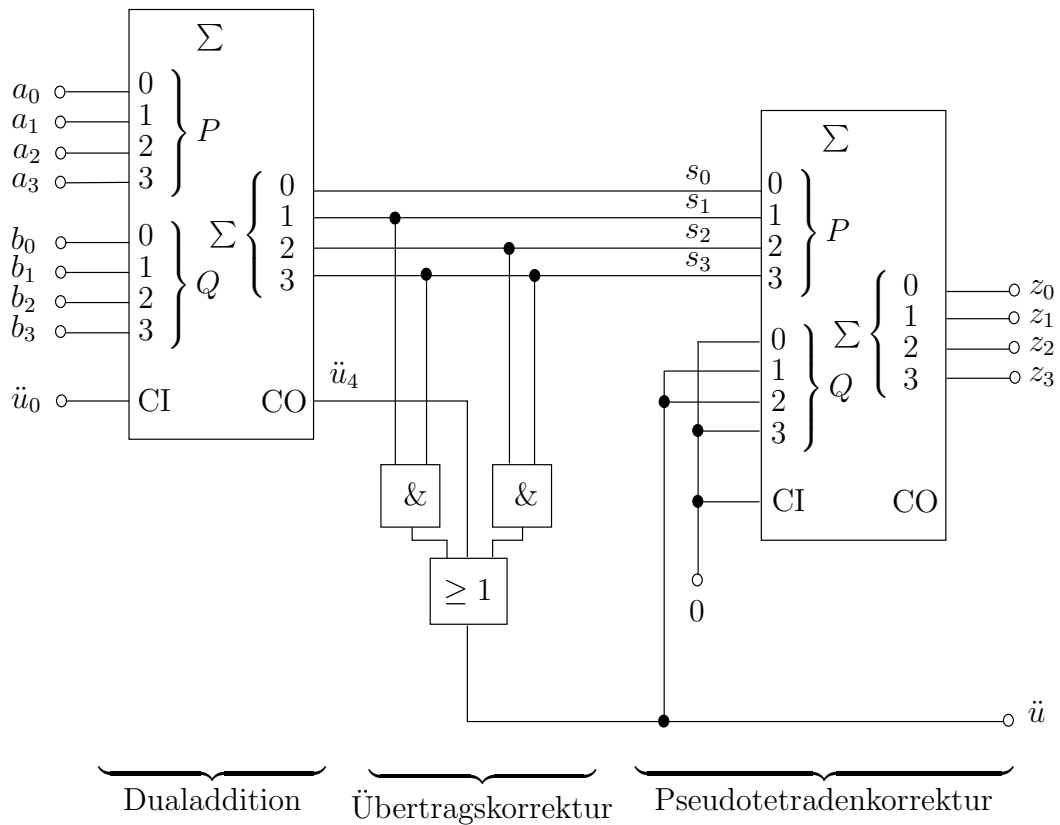
31	30	23	22	...	0
0	1000	1001	1111	0101 0100 0000 ...	000

4. BCD-Addierer für eine Tetrade:

Eine Korrektur ist durch die Addition von 0110 zu $(s_3s_2s_1s_0)$ notwendig, wenn

- ein Übertrag bei der Addition der beiden Tetraden $(a_3a_2a_1a_0)$ und $(b_3b_2b_1b_0)$ auftritt, d. h. $\ddot{u}_4 = 1$ oder
- das Ergebnis $(s_3s_2s_1s_0)$ eine Pseudotetrade ist, d. h. $s_3 = s_2 = 1$ oder $s_3 = s_1 = 1$

Damit ergibt sich: $\ddot{u} = \ddot{u}_4 \vee s_1 s_3 \vee s_2 s_3$



5. Datenwörter: (Man beachte: $k_i = QS_{i-1}$)

Position	12	11	10	9	8	7	6	5	4	3	2	1
	m_8	m_7	m_6	m_5	k_4	m_4	m_3	m_2	k_3	m_1	k_2	k_1
Codewort 1:	1	0	1	0	0	1	0	1	0	0	1	0
Codewort 2:	1	1	0	0	0	1	0	0	0	0	0	1

Die Prüfbits lassen sich nach den folgenden Regeln berechnen:

$$k_1 = k_1 \oplus m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7$$

$$k_2 = k_2 \oplus m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7$$

$$k_3 = k_3 \oplus m_2 \oplus m_3 \oplus m_4 \oplus m_8$$

$$k_4 = k_4 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_8$$

- Codewort 1: **1 0 1 0 0 1 0 1 0 0 1 0** $\Rightarrow k_4 k_3 k_2 k_1 = 0 1 1 0 \Rightarrow$ Es liegt ein Fehler an der 6. Position vor \Rightarrow Datenwort 1: **1 0 1 0 1 1 1 0**
- Codewort 2: **1 1 0 0 0 1 0 0 0 0 0 1** $\Rightarrow k_4 k_3 k_2 k_1 = 0 0 0 1 \Rightarrow$ Es liegt ein Fehler an der 1. Position vor \Rightarrow Datenwort 2: **1 1 0 0 1 0 0 0**

Aufgabe 5

	richtig	falsch
Das Nelson-Verfahren berechnet alle Primterme einer Booleschen Funktion.	×	
Koppelterme sind Primterme für mehrere Schaltfunktionen.		×
Funktionshasards können durch die Veränderung der Schaltnetzstruktur behoben werden.		×
Bei Moore-Schaltwerken hängt die Ausgabe vom Zustand und der Eingabe ab.		×
Flipflops sind kleine asynchrone Schaltwerke.	×	
Bei der Darstellung von Zahlen in Zweierkomplement-Form ist der Zahlenbereich symmetrisch.		×
Bei der Darstellung von Zahlen in Einerkomplement-Form ist der Zahlenbereich nicht symmetrisch.		×
<i>smallreal</i> ist die kleinste normalisierte Zahl, die man zu 1 addieren kann, um einen von 1 verschiedenen Wert zu erhalten.		×

Aufgabe 6

- | | | |
|--|---|------------|
| 1. Takt: $IAR \rightarrow SAR; \quad IAR \rightarrow X; \quad R = 1$ | } | Lese-Phase |
| 2. Takt: $Eins \rightarrow Y; \quad R = 1$ | | |
| 3. Takt: $ALU \text{ auf Addieren}; \quad R = 1$ | | |
| 4. Takt: $Z \rightarrow IAR$ | | |
| 5. Takt: $SDR \rightarrow IR$ | | |

Aufgabe 7

1. Kontrollstruktur in MIPS-Assembler:

```

    add    $a0, $zero, $zero    # i = 0;
loop:  slti  $v0, $a0, 100      # if i < 100 dann $v0 = 1
      beq   $v0, $zero, label   # if $v0 = 0 (d.h., i >= 100),
                                # dann Ende der for-Schleife

      mul   $a1, $a1, $a0       # j = j * i;
      addi  $a0, $a0, 1         # i++;
      beq   $zero, $zero, 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  $a0, add_array      # Abbruchbedingung

```

3. Inhalte der Zielregister:

Befehl	Zielregister = (z. B. \$s6 = 0x0000 F00A)
ori \$s1, \$zero, 0x10	\$s1 = 0x0000 0010
slr \$s2, \$s1, 3	\$s2 = 0x0000 0002
slti \$s3, \$s2, 100	\$s3 = 0x0000 0001
sub \$s4, \$s3, \$s2	\$s4 = 0xFFFF FFFF
lui \$s5, 0x20	\$s5 = 0x0020 0000

4. (a) Registerinhalte:

Register	Inhalt
\$t1	\$t1 = 0x0000 0008
\$t2	\$t2 = 0x0000 001F
\$t3	\$t3 = 0x0000 0011
\$t4	\$t4 = 0x0000 0025

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

```
la $s0, vec
```

Aufgabe 8

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

(b) Anzahl der Einträge:

$$\text{Anzahl der Einträge} = \frac{\text{Kapazität}}{\text{Blockgröße}} = \frac{512 \text{ KByte}}{64 \text{ Byte}} = 8 \text{ K Einträge}$$

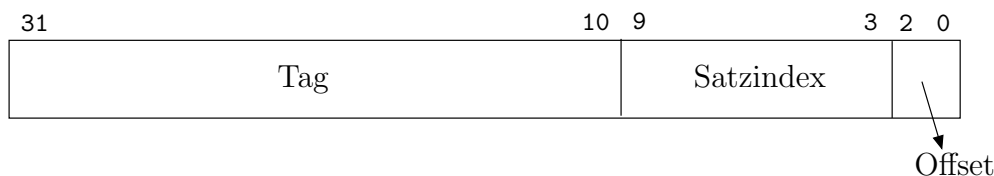
(c) Cache-Organisation:

10 Bit Index-Feld \Rightarrow Es lassen sich $2^{10} = 1 \text{ K}$ Sätze im Cache adressieren

$$\text{Assoziativität} = \frac{8 \text{ K}}{1 \text{ K}} = 8$$

Der Cache ist als 8-fach assoziativer Speicher (*8-way set associative*) organisiert

2. (a) Unterteilung der Hauptspeicheradresse:



(b) Speicherbedarf:

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

- Daten pro Zeile $8 \text{ Byte} \times 8 = 64 \text{ Bit}$
- Tag = $32 - 7 - 3 = 22 \text{ Bit}$ (7 Bit Satzindex und 3 Byte-Offset)

Speicherbedarf für eine Zeile: $22 + 1 + 64 \text{ Bit} = 87 \text{ Bits}$

Speicherbedarf für den gesamten Cache (128 Sätze mit jeweils 5 Zeilen):

$$87 \cdot 128 \cdot 5 = 55680 \text{ Bits} = 6960 \text{ Byte}$$

3. (a) Der DMA-Controller beschreibt eine Speicherzelle, deren vorhandener Inhalt im Cache als gültig eingetragen war, und der Prozessor führt einen Lesezugriff mit der Adresse dieser Speicherzelle durch. Der Prozessor wird dann den inzwischen veralteten Cache-Eintrag lesen.

(b) Der Prozessor führt einen Schreibzugriff mit einer Adresse aus dem gemeinsamen Speicherbereich aus und aktualisiert dabei nur seinen Cache. Der DMA-Controller wird bei einem Lesezugriff mit dieser Adresse den veralteten Hauptspeicher-Eintrag lesen.

- (c)
- Den gemeinsam benutzten Speicherbereich von einer Speicherung im Cache ausschließen. Diese Lösung wird z.B. durch die MMU unterstützt. Dazu wird der Adressbereich als *non-cachable* gekennzeichnet. Dies weist die Cache-Steuerung an, bei allen Prozessorzugriffen, die diesen Adressraum betreffen, nicht aktiv zu werden. Ist keine MMU vorhanden, so kann die Cache-Steuerung dies auch erledigen.
 - *Cache-Clear, Cache-Flush*: Bei einem DMA-Zugriff auf den gemeinsamen Speicherbereich wird der Cache gelöscht, so dass nachfolgende Prozessorzugriffe zu einem Neuladen des Cache führen.
 - *Bus-Schnüffeln (Bus-Snooping)*: Die Cache-Steuerung beobachtet den Bus hinsichtlich der Speichzugriffe des DMA-Controllers.

Aufgabe 9

1. Größe des maximal verfügbaren virtuellen Adressraums in Byte:

$$2^{m+x+r} \text{ Byte} \quad 0 \dots (2^{m+x+r} - 1)$$

Anzahl der Segmente im virtuellen Adressraum:

$$2^m \text{ Segmente} \quad 0 \dots (2^m - 1)$$

2. Anzahl der Seiten pro Segment:

$$2^x \text{ Seiten pro Segment}$$

Größe der Seiten in Byte:

$$2^r \text{ Byte pro Seite}$$

3. Vorteil einer zweistufigen Adressumsetzung gegenüber einer reinen Seitenverwaltung:

Bei einer reinen Seitenverwaltung hat man eine einzige, sehr große Seitentabelle, die im Hauptspeicher entsprechend viel Platz beansprucht. Im Gegensatz dazu gibt es bei einer zweistufigen Adressumsetzung viele, jedoch kleinere Seitentabellen, von denen nur die aktuelle im Hauptspeicher (neben der Segmenttabelle) gehalten werden muss. Die anderen Seitentabellen können in einem Hintergrundspeicher stehen und müssen dann bei Bedarf geladen werden.

Aufgabe 10

1. 512×8-Organisation: 512 Zellen mit 8-Bit Wörter ⇒ 512 Zellen müssen adressiert werden. Dazu sind 9 Adreßleitungen erforderlich.
2. Es sind 16 RAM-Bausteine der Organisation 8k×1 notwendig, um einen Speicher mit einer Kapazität von 8k Wörter und einer Wortbreite von 16 Bit zu realisieren.
3. ROM-Baustein der Speicherkapazität von 2048 Bits und 7 Adreßleitungen ⇒ Es können 128 Zellen adressiert werden ⇒ 128×16-Organisation
4. Speicherhierarchie (Problemstellungen und Lösungsmöglichkeiten):
 - **Block-Abbildungsstrategie:** Wohin wird ein Block abgebildet?
 - Vollasoziative, Satzassoziativ, direkte Abbildung (*direct mapped*)
 - **Block-Identifikation:** Wie kann ein Block gefunden werden?
 - *Tag*, Block
 - **Block-Ersetzungsstrategie:** Welcher Block soll ersetzt werden?
 - *Random, FIFO, LRU, ...*
 - **Aktualisierungsstrategie:** Was passiert bei einem Schreibzugriff?
 - *write back, write through*
5. Timing-Parameter eines DRAM-Bausteins:

