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 23. Juli 2018, 14:00 – 16:00 Uhr

Name:	Vorname:	Matrikelnummer:
Bond	James	007

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

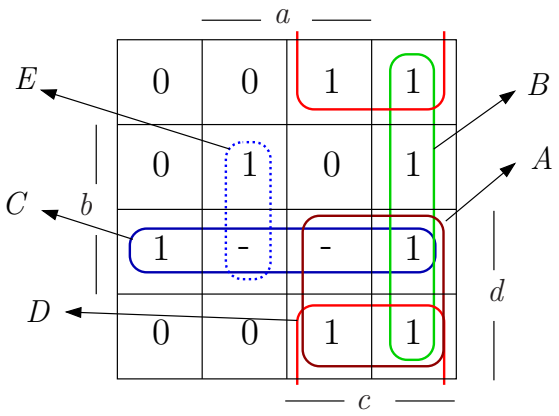
Rechnerorganisation (TI-2)	
Aufgabe 6	10 von 10 Punkten
Aufgabe 7	10 von 10 Punkten
Aufgabe 8	11 von 11 Punkten
Aufgabe 9	8 von 8 Punkten
Aufgabe 10	6 von 6 Punkten

Gesamtpunktzahl:	90 von 90 Punkten

	Note: **1,0**

Aufgabe 1 *Schaltfunktionen* (11 Punkte)

1. $f(d, c, b, a)$: 4 P.



Primimplikanten:

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

$D: \underline{(c\,\overline{b})}$ $\quad E: \underline{(\overline{c}\,b\,a)}$

2. Disjunktive Minimalform von $f(d, c, b, a)$: 1 P.

$$\begin{aligned} f(d,c,b,a) &= B \vee C \vee D \vee E \\ &= c\,\overline{a} \vee d\,b \vee c\,\overline{b} \vee \overline{c}\,b\,a \end{aligned}$$

3. Die Schaltfunktion ist unvollständig definiert, da 2 P.

   - Primimplikanten bei vollständig definierten Funktionen aus 2^n Mintermen bestehen. Der Primimplikanten B überdeckt 3 Minterme, d. h. er muss noch eine Freistelle enthalten. ODER
   - Primimplikant A ist im Primimplikanten B enthalten. Somit wäre A kein Primimplikant $\rightarrow$ Widerspruch.

4. Kernprimimplikanten: C und D 1 P.

5. Überdeckungsfunktion: 3 P.

$$\begin{aligned} \ddot{u}_g &= (A \vee B)(A \vee B)\,CD\,(B \vee D) \\ &= (A \vee B)\,CD\,(B \vee D) \\ &= (ACD \vee BCD)(B \vee D) \\ &= ACDB \vee ACD \vee BCD \vee BCD \\ &= ACD \vee BCD \end{aligned}$$

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

1. Disjunktive Minimalform von $f(d,c,b,a)$: 3 P.

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

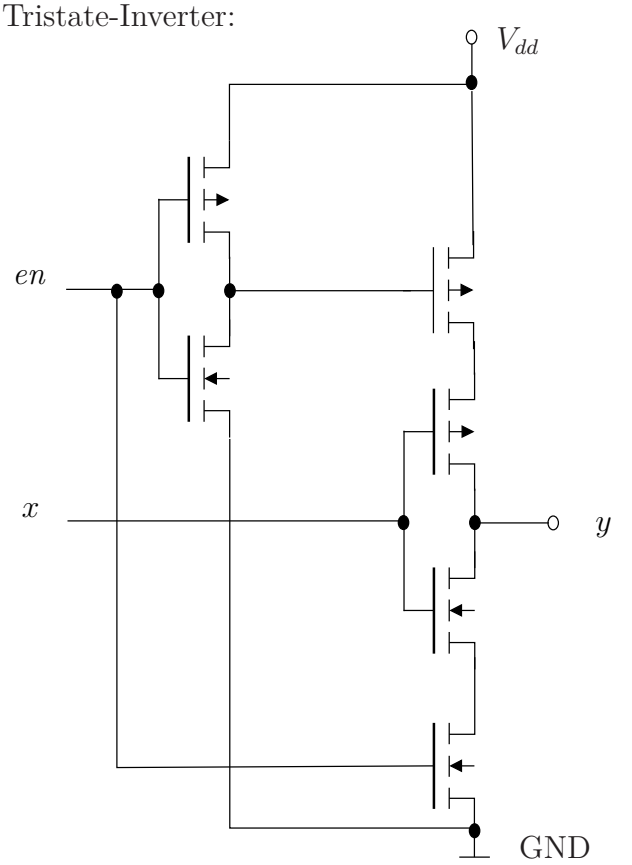
2. Disjunktive Normalform von $g(d,c,b,a)$: 2 P.

$$\begin{aligned} g(d,c,b,a) &= \overline{\overline{(b\,a)} \vee \overline{(d\,c)}} = \overline{\overline{(b\,a)} \cdot \overline{(d\,c)}} \\ &= a\,b\,d\,c = m_{15} \end{aligned}$$

3. Schaltfunktion $h(d,c,b,a)$ in disjunktiver Form: 2 P.

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

4. Tristate-Inverter: 3 P.



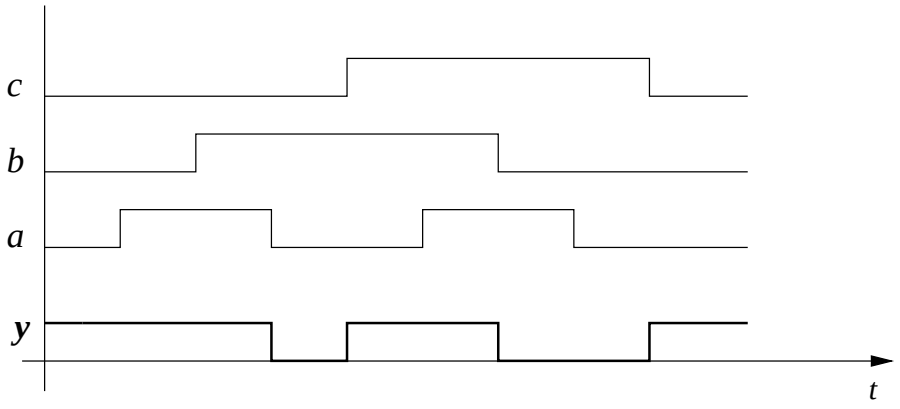
Funktionstabelle:

en	x	y
0	–	hochohmig
1	0	1
1	1	0

Aufgabe 3 *Laufzeiteffekte* (6 Punkte)

1. Verlauf von y: 1 P.

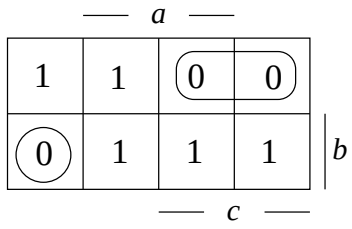
$$y = (a\bar{\wedge}b)\bar{\wedge}(b\bar{\wedge}c)\bar{\wedge}(b \vee c) = ab \vee bc \vee \bar{b}\bar{c}$$



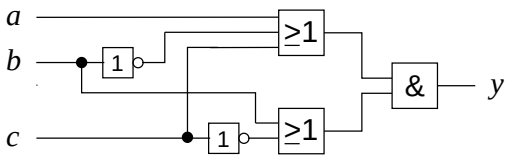
2. Keiner der Übergänge ist mit einem Funktionshasard behaftet. 2 P.
Begründung: Bei allen Übergängen wechselt nur eine Variable. Diese Übergänge sind stets funktionshasardfrei.

   Ja, es kann kurzzeitig ein falscher Wert am Ausgang entstehen.
Begründung: Bei den Übergängen, bei denen b oder c wechseln, können Strukturhasards auftreten, die Hasardfehler verursachen.

3. Ein Schaltnetz, welches die Disjunktion aller Primimplikanten bzw. die Konjunktion aller Primimplikate realisiert, hat dieselbe logische Funktion und weist bei den betrachteten Übergängen keine Hasards auf. Begründung: Satz von Eichelberger. 3 P.



Konjunktion aller Primimplikate: $y = (\bar{c} \vee b) \wedge (c \vee \bar{b} \vee a)$



oder Disjunktion aller Primimplikanten $y = ab \vee bc \vee \bar{b}\bar{c} \vee \bar{c}a$

Aufgabe 4 *Schaltwerke* (9 Punkte)

1. Automatentyp: Mealy-Automat **1 P.**

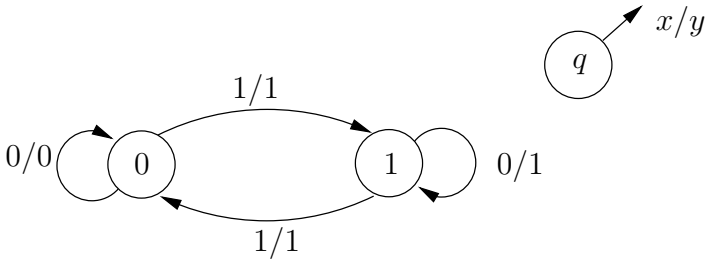
   Begründung: die Ausgabe hängt sowohl vom Zustand als auch von der Eingabe ab.

2. Ansteuerfunktion: $d^t = (x \vee q)^t\,(\overline{x} \vee \overline{q})^t$ **2 P.**
   Zustandsübergangsgleichung: $q^{t+1} = d^t = (x \vee q)^t\,(\overline{x} \vee \overline{q})^t$
   Ausgabefunktion: $y^t = (x \vee q)^t$

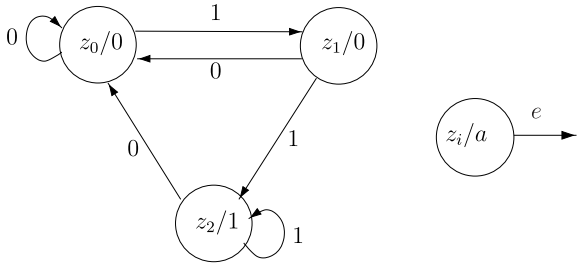
3. Automatengraph des Schaltwerks: **1 P.**
   Kodierte Ablauftabelle:

q^t	x^t	q^{t+1}	y^t
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1

$\Rightarrow$



4. Automatengraph mit minimaler Anzahl Zustände: **3 P.**



5. Zustandsübergangsgleichungen: **2 P.**

q_1^t	q_0^t	e^t	q_1^{t+1}	q_0^{t+1}
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	–	–
1	1	1	–	–

$$q_1^{t+1} = (e\,q_0 \vee e\,q_1)^t$$
$$q_0^{t+1} = (e\,\overline{q}_1\,\overline{q}_0)^t$$

q_1^{t+1}:

q_0 \ e			
	0	0	1	0
	0	1	-	-

q_1

q_0^{t+1}:

q_0 \ e			
	0	1	0	0
	0	0	-	-

q_1

Aufgabe 5 *Rechnerarithmetik & Codes* (9 Punkte)

1. $43,21_5$ in eine Dezimalzahl: 1 P.

$$43,21_5 \;=\; 4\cdot 5^1 + 3\cdot 5^0 + 2\cdot 5^{-1} + 1\cdot 5^{-2} = 23,44_{10} \qquad (=\; 23 + \frac{11}{25})$$

2. $9,6C_{16}$ in eine Zahl zur Basis 8: 1 P.

$$9,6C_{16} \;=\; 1001,0110\ 1100_2 \;=\; 1\ 001,011\ 011\ 00_2 \;=\; 11,33_8$$

3. 1001 0100 0010 0000 0000 0000 0000 0001: 4 P.

   (a) Vorzeichenlose Dualzahl:

$$2^{31} + 2^{28} + 2^{26} + 2^{21} + 2^0$$

   (b) Zahl in Zweierkomplement-Darstellung:

$$-2^{31} + 2^{28} + 2^{26} + 2^{21} + 2^0$$

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

$$\begin{aligned} VZ &= 1 \\ Char &= 0010\ 1000 \;=\; 40 \\ Exp &= Char - 127 \;=\; -87 \\ M &= 010\ 0000\ 0000\ 0000\ 0000\ 0001 \;\Rightarrow \\ \\ Z &= (-1)^1 \cdot (1,01000000000000000000001)\cdot 2^{-87} \\ &= -(1 + 2^{-2} + 2^{-23})\cdot 2^{-87} \end{aligned}$$

4. Datenwörter: 3 P.

Position	11	10	9	8	7	6	5	4	3	2	1
	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	0	0	1	1
Codewort 2:	1	0	0	0	1	0	0	0	0	1	0

   Die Prüfbits lassen sich nach den folgenden Regeln berechnen:

$$\begin{aligned} k_1 &= k_1 \oplus m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 \\ k_2 &= k_2 \oplus m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 \\ k_3 &= k_3 \oplus m_2 \oplus m_3 \oplus m_4 \\ k_4 &= k_4 \oplus m_5 \oplus m_6 \oplus m_7 \end{aligned}$$

   - Codewort 1: **1 0 1 0 0 1 0 0 0 0 1 1** $\Rightarrow$ $k_4\ k_3\ k_2\ k_1 = 0\ 1\ 1\ 1$ $\Rightarrow$ Es liegt ein Fehler an der 7. Postion vor $\Rightarrow$ Datenwort 1 = **1 0 1 1 1 0 0**

   - Codewort 2: **1 0 0 0 1 0 0 0 0 0 1 0** $\Rightarrow$ $k_4\ k_3\ k_2\ k_1 = 1\ 1\ 1\ 0$ $\Rightarrow$ Es liegt angeblich ein Ein-Bit-Fehler an Position 14 vor. Es gibt aber keine Position 14 $\Rightarrow$ es liegt ein Mehrbit-Fehler vor $\Rightarrow$ Datenwort 2 kann nicht ermittelt werden.

Aufgabe 6 *MIPS-Assembler* (10 Punkte)

1. Inhalte der Zielregister: 2 P.

Befehl	Zielregister = Wert (z.B. `$s6 = 0x0000 F00A`)
`ori  $s1, $zero, ox2009`	`$s1 = 0x0000 2009`
`sll  $s2, $s1, 3`	`$s2 = 0x0001 0048`
`slti $s3, $s2, 0x0001 0049`	`$s3 = 0x0000 0001`
`sub  $s4, $s3, $s2`	`$s4 = 0xFFFE FFB9`

2. C-Kontrollstrukturen in MIPS-Assembler: 5 P.

   (a) `if ( k == j) k = 0;`

```
            bne    $a0, $a1, label      #  if (k != j) goto label
            add    $a0, $zero, $zero    #  k = 0;
    label:
```

   (b) `for (i = 0; i < 100; i++) j = j + i;`

```
           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
           add     $a1, $a1, $a0       # j = j + i;
           addi    $a0, $a0, 1         # i++;
           beq     $zero, $zero, loop  # gehe zur Marke loop
    label:
```

3. Fehlerfreie Version: 3 P.

```
       add  $v0, $zero, $zero     # $v0 mit 0 initialisieren
loop:  lw   $v1, 0x0($a0)         # nächstes Wort lesen
       sw   $v1, 0x0($a1)         # Wort schreiben
       beq  $v1, $zero, done      # Nullwort gelesen, dann Ende
       addi $v0, $v0, 1           # Zähler inkrementieren
       addi $a0, $a0, 4           # Zeiger auf nächstes Wort (Quelle)
       addi $a1, $a1, 4           # Zeiger auf nächstes Wort (Ziel)
       j    loop
done:
```

Aufgabe 7 *Pipelining* (10 Punkte)

1. Kategorisierte Datenabhängigkeiten: 3 P.

   - True Dependence (δ^t):

     $S_1 \longrightarrow S_2$ $S_3 \longrightarrow S_4$ $S_1 \longrightarrow S_5$

   - Anti Dependence (δ^a):

     $S_2 \longrightarrow S_3$ $S_2 \longrightarrow S_4$ $S_1 \longrightarrow S_5$

   - Output Dependence (δ^o):

     $S_3 \longrightarrow S_4$

2. Zustand von Pipeline und Registern: 3 P.

Takt	IF	ID/RF	EX	MEM	WB	$t1	$t2	$t3	$t4	$t5
1	S1	—	—	—	—	3	5	7	9	2
2	S2	S1	—	—	—	3	5	7	9	2
3	S3	S2	S1	—	—	3	5	7	9	2
4	S4	S3	S2	S1	—	3	5	7	9	2
5	S5	S4	S3	S2	S1	12	5	7	9	2
6	—	S5	S4	S3	S2	12	5	7	4	2
7	—	—	S5	S4	S3	12	5	46	4	2
8	—	—	—	S5	S4	12	5	20	4	2
9	—	—	—	—	S5	12	14	20	4	2

   Anzahl der Takte: 9

3. Behebung der Pipelinekonflikte durch Einfügen von `NOP`-Befehlen: 2 P.

```
S1:  addi  $t1, $t2, 7
     nop
     nop
S2:  sub   $t4, $t3, $t1
S3:  muli  $t3, $t5, 23
     nop
     nop
S4:  addi  $t3, $t3, 13
S5:  add   $t2, $t1, $t5
```

   Anzahl der Takte: 13

4. Bedingte Sprünge (Problem und zwei Behandlungsmöglichkeiten): 2 P.

   Problem: Erst am Ende der 4. Stufe wird entscheiden, ob gesprungen wird oder nicht. Die folgenden drei Befehle sind schon in der Pipeline und müssen im Falle eines Sprungs gelöscht werden.

   Behandlungsmöglichkeiten:
   - Verzögerte Sprungtechnik *(delayed branch technique)*: z.B. drei Verzögerungszeitschlitze *(delay slots)* mit Leerbefehlen (`NOP`) nach jedem Sprungbefehl.
   - Befehlsumordnung: Befehle, die in der logischen Programmreihenfolge vor dem Sprungbefehl liegen, in die Verzögerungszeitschlitze verschieben.
   - Pipeline-Leerlauf (ineffizient): Die Hardware erkennt die Verzweigungsbefehle in der ID-Phase und lädt keine weiteren Befehle in die Pipeline, bis die Zieladresse berechnet und im Fall bedingter Sprungbefehle die Sprungentscheidung getroffen ist.
   - Sprungvorhersage: Spekulation über Sprungentscheidung und/oder Sprungziel, um möglichst wenig Pipeline-Leerlauf/Delay Slots zu haben.

Aufgabe 8 *Cache-Speicher* (11 Punkte)

1. (a) Anzahl der Cache-Einträge: 1 P.

$$\frac{\text{Kapazität}}{\text{Blockgröße}} = \frac{256\ KByte}{2^4\ Byte} = \frac{2^8\ KByte}{2^4\ Byte} = 16\ K \text{ Einträge}$$

(b) Cache-Organisation: 2 P.

12 Bit Index-Feld ⇒ Es lassen sich $2^{12} = 4$ K Sätze im Cache addressieren

$$\text{Assoziativität} = \frac{16\ K}{4\ K} = 4$$

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

2. Speicherbedarf: 2 P.

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

- Daten pro Zeile 16 Byte = 128 Bit
- Tag = 32 - 6 - 4 = 22 Bit (6 Bit als Satzindex und 4 Bit als Byte-Offset)

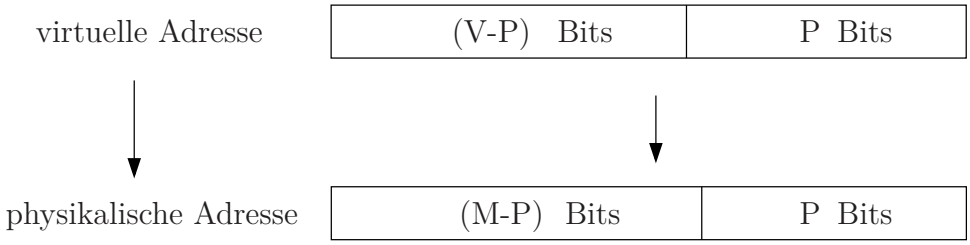
Speicherbedarf für eine Zeile: $22 + 2 + 128$ Bit = 152 Bits = 19 Bytes
Speicherbedarf für den gesamten Cache: $152 \cdot 64 \cdot 7$ = 68096 Bits = 8512 Bytes

3. Speicherzugriffe: 6 P.

Adresse	**64**	**32**	**64**	**0**	**112**	**64**	**128**	**48**	**240**	**0**
read/write	r	r	r	r	w	w	r	r	r	w
Index	0	2	0	0	3	0	0	3	3	0
Tag	1	0	1	0	1	1	2	0	3	0
Hit/Miss	Miss	Miss	Hit	Miss	Miss	Hit	Miss	Miss	Miss	Miss
write back?	nein	nein	nein	nein	nein	nein	nein	nein	ja	ja

Aufgabe 9 *Virtuelle Speicherverwaltung* (8 Punkte)

1. Virtuelle und physikalische Adresse: 2 P.



2. Es können $\left(\frac{2^M}{2^P} = 2^{M-P} \right)$ Seiten auf einmal im physikalischen Adressraum gespeichert werden. Anzahl der Einträge in der Seitentabelle ist gleich: $\left(\frac{2^V}{2^P} = 2^{V-P} \right)$ 2 P.

3. Anzahl der Bits pro Eintrag in der Seitentabelle ist gleich: $(M - P + 2)$. 4 P.
   $V = 24$, $M = 21$ und $P = 10$ $\Rightarrow$ 13 Bits $\Rightarrow$ pro Eintrag sind 2 Bytes notwendig.

   2^{V-P} Einträge $\times$ 2 Bytes $= 2^{15}$ Bytes $= 32$ kByte Speicherbedarf für die Tabelle.
   Anzahl der benötigten Seiten für die Seitentabelle: Eine Seite ist 2^{10} Byte = 1 kByte groß $\Rightarrow$ die Seitentabelle benötigt 32 Seiten.

Aufgabe 10 *Multiple Choice* (6 Punkte)

1. 2 P.

Speicher-Bausteine	*richtig*	*falsch*
SDRAM arbeitet synchron zum Systemtakt und Datenpakete werden sowohl bei steigender als auch bei fallender Taktflanke übertragen.		×
SRAM wird vorwiegend für schnelle Zwischenspeicher wie Register und Caches eingesetzt.	×	
Eine DRAM-Speicherzelle besteht aus zwei rückgekoppelten Invertern.		×
Statische RAM-Bausteine lassen sich dichter als dynamische RAM-Bausteine integrieren.		×

- SDRAM überträgt nur bei einer Flankenart, DDR SDRAM bei Beiden
- Eine DRAM Speicherzelle besteht aus Kondensator und Schalttransistor; SRAM aus Invertern (und Zugriffstransistoren)
- SRAM ~6 Transistoren, DRAM ~1 Transistor

2. 2 P.

Cache-Speicher	*richtig*	*falsch*
Cache-Speicher größer Kapazität werden in der Regel als vollassoziative Cache-Architektur realisiert.		×
Bei einem virtuellen Cache-Speicher werden die höherwertigen Bits der logischen Adresse als Tag abgelegt	×	
Bei einem physikalischen Cache-Speicher werden höchstens genauso viele Bits als Tag gespeichert wie bei einem virtuellen Cache-Speicher.	×	
Bei einem Hit in einem physikalischen Cache-Speicher wird die Speicherverwaltungseinheit (MMU) zur Adressumsetzung nicht benötigt.		×

- Vollassoziativ ist aufwändig (man muss mit allen(!) Tags vergleichen, nicht nur mit denen eines Satzes), darum nur bei kleinen Caches sinnvoll
- Bei physikalischen Cache muss schon vor dem Zugriff die Adressumsetzung gemacht werden

3. 2 P.

Systembusse	*richtig*	*falsch*
Ein semi-synchroner Systembus arbeitet synchron zum Systemtakt.	×	
Die langsamste Komponente an einem synchronen Systembus bestimmt den Systemtakt.	×	
An einem Bus darf es immer nur einen Master geben, aber beliebig viele Slaves.		×
Auf einem Split-Bus werden Adresse und Daten zeitlich nacheinander auf den gleichen Leitungen übertragen.		×

- Ein Bus kann mehrere Master haben, braucht dann aber einen zentralen Arbiter (z.B. bei AHB) oder ein verteiltes Verfahren (z.B. bei I^2C), das sicherstellt, dass nicht mehrere Master den Bus gleichzeitig benutzen
- Bei einem Split-Bus sind Adress- und Datenleitungen physikalisch aufgeteilt (gesplittet), bei einem MUX-Bus werden die gleichen Leitungen nacheinander benutzt (gemultiplexed)