

# Musterlösungen zur Klausur

Rechnerorganisation/Digitaltechnik und Entwurfsverfahren

Technische Informatik I/II

am 01. März 2010, 18:00 – 20:00 Uhr

Name:	Vorname:	Matrikelnummer:
Bond	James	007

<b>Digitaltechnik und Entwurfsverfahren/TI-1</b>	
Aufgabe 1	6 von 6 Punkten
Aufgabe 2	11 von 11 Punkten
Aufgabe 3	11 von 11 Punkten
Aufgabe 4	8 von 8 Punkten
Aufgabe 5	9 von 9 Punkten

<b>Rechnerorganisation/TI-2</b>	
Aufgabe 6	9 von 9 Punkten
Aufgabe 7	10 von 10 Punkten
Aufgabe 8	8 von 8 Punkten
Aufgabe 9	11 von 11 Punkten
Aufgabe 10	7 von 7 Punkten

<b>Gesamtpunktzahl:</b>	90 von 90 Punkten
-------------------------	-------------------

<b>Note:</b>	<b>1,0</b>
--------------	------------

## Aufgabe 1

1. 
$$\begin{aligned} (a \leftrightarrow b) \leftrightarrow c &= (a \leftrightarrow b)c \vee (a \nleftrightarrow b)\bar{c} \\ &= (ab \vee \bar{a}\bar{b})c \vee (a\bar{b} \vee \bar{a}b)\bar{c} \\ &= abc \vee \bar{a}\bar{b}c \vee a\bar{b}\bar{c} \vee \bar{a}b\bar{c} \\ &= a(bc \vee \bar{b}\bar{c}) \vee \bar{a}(b\bar{c} \vee \bar{b}c) \\ &= a(b \leftrightarrow c) \vee \bar{a}(b \nleftrightarrow c) \\ &= a \leftrightarrow (b \leftrightarrow c) \end{aligned}$$
2. 
$$\begin{aligned} a \leftrightarrow b \leftrightarrow c &= a \leftrightarrow (b \leftrightarrow c) && \text{Assoziativ Gesetz} \\ &= a \nleftrightarrow \overline{(b \leftrightarrow c)} && (x \leftrightarrow y = xy \vee \bar{x}\bar{y} = x \nleftrightarrow \bar{y}) \\ &= a \nleftrightarrow (b \nleftrightarrow c) && \text{Inverses Element} \\ &= a \nleftrightarrow b \nleftrightarrow c && \text{Assoziativ Gesetz} \end{aligned}$$

## Aufgabe 2

1. KMF von  $f(c, b, a)$ :

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

- 2.

Nr.	gebildet aus	Würfel			gestrichen wegen
		c	b	a	
1		0	1	0	$\subset 6$
2		1	1	0	$\subset 6$
3		1	0	1	$\subset 9$
4		0	1	1	$\subset 7$
5		1	1	1	$\subset 7$
6	2,1	-	1	0	$\subset 8$
7	5,4	-	1	1	$\subset 8$
8	7,6	-	1	-	<b>Primimplikant</b>
9	8,3	1	-	1	<b>Primimplikant</b>

Die Primimplikanten sind:  $b$  und  $c a$

3. (a)  $h(d, c, b, a)$  vollständig oder unvollständig definiert? unvollständig definiert.

Begründung: Die Primimplikanten bei vollständig definierten Funktionen überdecken  $2^n$  Minterme.

- $B$  (Minterm 8) wäre kein Primimplikant, wenn die Funktion vollständig definiert ist, da  $B$  dann in  $C$  (Mintermen 8 und 9) enthalten wäre.
- Der Primimplikant  $F$  überdeckt drei Minterme  $\rightarrow$  Widerspruch

(b) DMF von  $h(d, c, b, a)$

	4	5	6	8	9	10	13
A	×						
B				×			
C				×	×		
D				×		×	
E					×		×
F	×	×	×				
G		×					×

- Kern-Primimplikanten:  $F$  und  $D \rightarrow$  Spalten mit den Mintermen 4, 5, 6, 8 und 10 werden gestrichen  $\rightarrow$  reduzierte Tabelle:

	9	13
A		
B		
C	×	
E	×	×
G		×

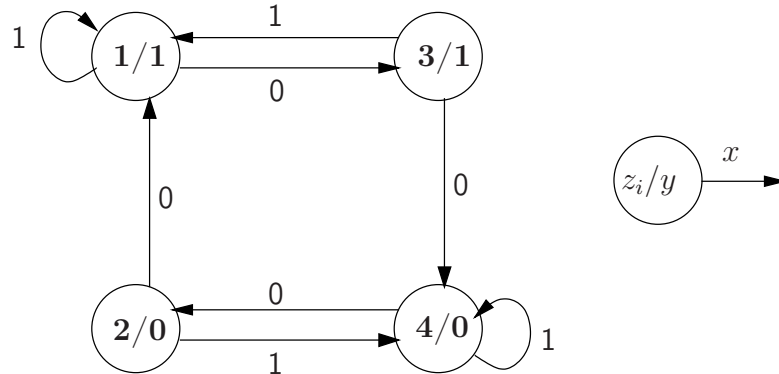
- Primimplikant  $E$  dominiert  $C$  und  $G$  (überdeckt sowohl 9 als auch 13)
- $A$  und  $B$  sind entbehrlich.

DMF:

$$h(d, c, b, a) = F \vee D \vee E = \bar{d}c \vee d\bar{c}\bar{a} \vee d\bar{b}a$$

### Aufgabe 3

1. Automatengraph:



2. Automatengraph:

- Ansteuergleichungen und Ausgabefunktion: Ablesen aus dem Schaltbild

$$\begin{aligned}
 j_1^t &= 1 & k_1^t &= \bar{q}_0^t & d_0^t &= q_1^t e^t \\
 a^t &= \bar{q}_1^t q_0^t
 \end{aligned}$$

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

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

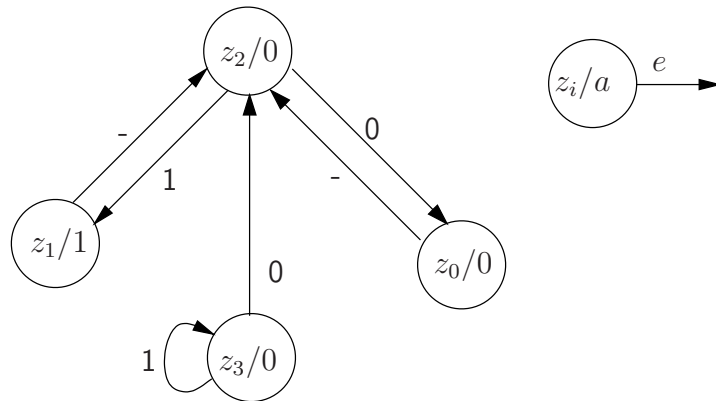
Mit den Ansteuergleichungen der Flipflops erhält man die Zustandsübergangsgleichungen für  $A$ , und  $B$ :

$$q_1^{t+1} = \bar{q}_1^t \vee q_0^t q_1^t \qquad q_0^{t+1} = d_0^t = q_1^t e^t$$

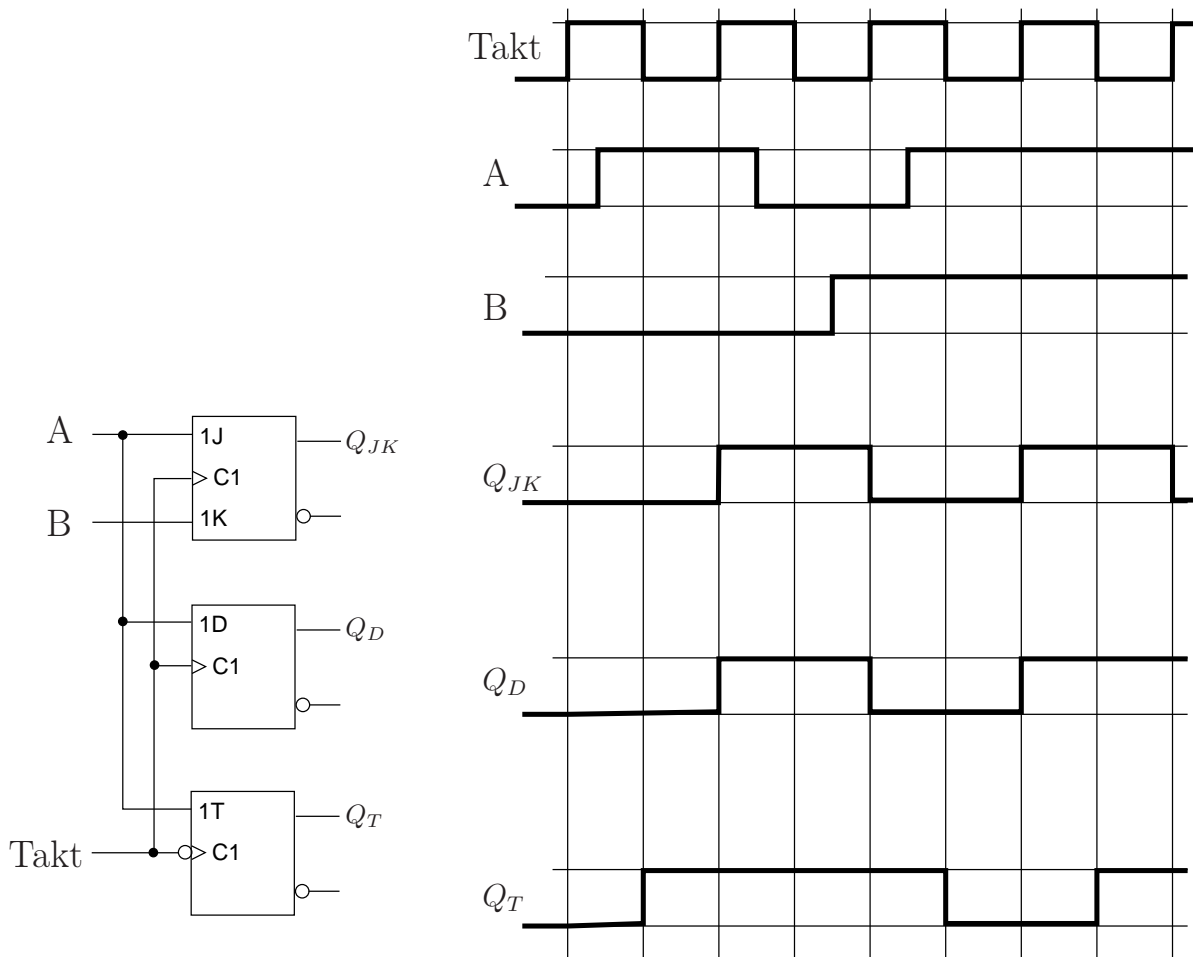
- Kodierte Ablauf-tabelle:

$q_1^t$	$q_0^t$	$e^t$	$q_1^{t+1}$	$q_0^{t+1}$	$a^t$
0	0	0	1	0	0
0	0	1	1	0	0
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	0

- Automatengraph:



3.



4. Anzahl der Zustände des Gesamtsystems:  $2^{2+4} = 64$

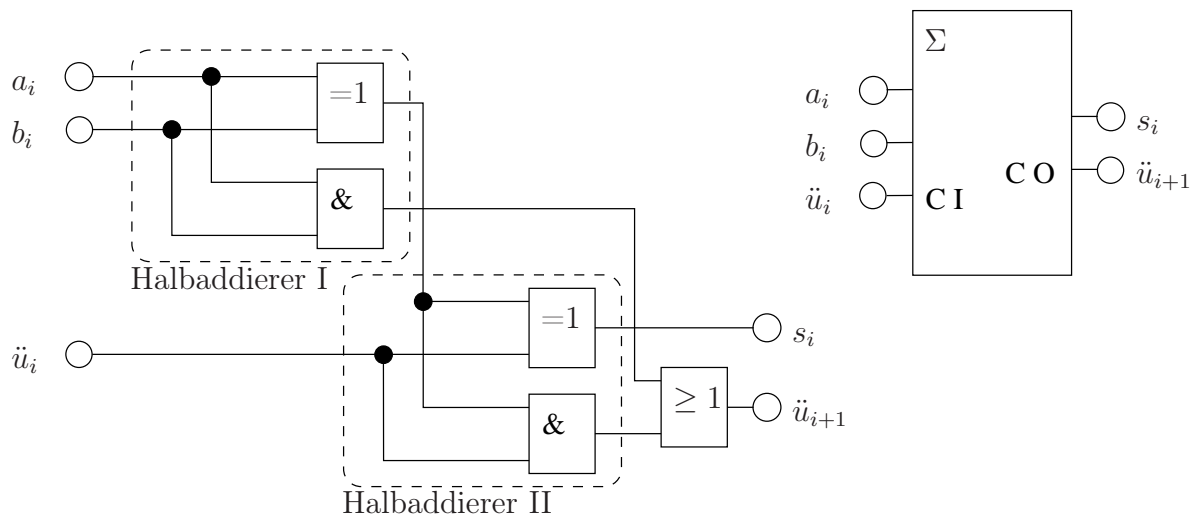
## Aufgabe 4

1.  $37_{10} = 0000\ 0000\ 0010\ 0101_{ZK}$
2.  $-37_{10} = 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101\ 1011_{ZK}$
3. Mögliche Interpretationen der 32-Bit Folge 1010000110010101111111101101101
  - Vorzeichenlose Dualzahl
  - Dualzahl in Vorzeichen-Betrag-Form
  - Dualzahl in Einerkomplement-Form
  - Dualzahl in Zweierkomplement-Form
  - Festkomma-Zahl
  - Gleitkomma-Zahl
  - 32-Bit Maschinen-Befehl
4.  $10011011 \times 10000101$

M(0:7)		
1000 0101		
A(0:7)	Q(0:7)	
0000 0000	1001 1011	
000 0101	1001 1011	Addiere M(1:7) zu A
000 0010	1100 1101	Rechtsschieben
000 0111	1100 1101	Addiere M(1:7) zu A
000 0011	1110 0110	Rechtsschieben
000 0001	1111 0011	Rechtsschieben
000 0110	1111 0011	Addiere M(1:7) zu A
000 0011	0111 1001	Rechtsschieben
000 1000	0111 0011	Addiere M(1:7) zu A
000 0100	0011 1100	Rechtsschieben
000 0010	0001 1110	Rechtsschieben
000 0001	0000 1111	Rechtsschieben
<b>0000 0001</b>	<b>0000 1110</b>	<b>Korrektur</b>

## Aufgabe 5

- Unterschied zwischen Halbaddierer und Volladdierer:  
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.
- Schaltbild eines 1-Bit-Volladdierers:



- Anzahl der Prüfbits:  
Aufwand:  $2^k \geq m + k + 1$ . Hier:  $m = 32 \Rightarrow k = 6$
- Physikalische Ursache für Hazardfehler:  
Unterschiedliche Laufzeiten der Signale bei Eingabeänderungen aufgrund unterschiedlicher Schaltzeiten der Gatter und Leitungsverzögerungen (unterschiedliche Totzeiten der Signalfade durch das Schaltnetz).
- 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.
- Schieberegister als:
  - Serien-Parallel-Wandlung
  - Parallel-Serien-Wandlung
  - Warteschlange (FIFO-Speicher) oder Stapelspeicher (LIFO-Speicher)
  - Umlaufspeicher
  - Multiplikation und Division
  - ...

## Aufgabe 6

1. Komponenten eines von-Neumann-Rechners:  
Steuerwerk, Rechenwerk, Speicher, Verbindungseinrichtung (Bus) und Eingabe-/Ausgabe-Einheiten
2. (a) Befehlsregister  
(b) Statusregister  
(c) Programmzähler  
(d) Rechenwerk (ALU)
3. (a) Einheitliche Befehlslänge (und einheitliches Befehlsformat)  
(b) Der Zugriff auf den Speicher erfolgt nur über *Load-Store-Befehle*  
(c) festverdrahtet  
(d) Getrennte Speicher und Busse für Befehle und Daten
- 4.

<i>Adressierung</i>	richtig	falsch
Bei einem von-Neumann-Rechner kann anhand der Adresse eines Datums erkannt werden, ob es sich bei diesem Datum um einen Operanden oder um einen Befehl handelt.		×
Beim Little-Endian-Datenformat ist das niedrigstwertige Byte eines Wortes an seiner niedrigsten Adresse.	×	
Bei einer speicherbezogenen Adressierung ( <i>memory mapping</i> ) von Peripherie-Bausteinen existieren zwei getrennte Adressräume für Speicher und Peripherie.		×
Durch das Multiplexen bestimmter Signalgruppen kann die Anzahl der Anschlüsse am Prozessorgehäuse verringert werden.	×	
<i>Speicher-Bausteine</i>	richtig	falsch
Der Inhalt von RAM-Speicher-Bausteinen ist nicht flüchtig.		×
SRAM-Bausteine speichern die Information in Zellen, die aus Flip-flops bestehen.	×	
DRAM-Bausteine müssen periodisch refreshed werden, damit ihr Inhalt nicht verloren geht	×	
Die Zugriffszeit ( <i>access time</i> ) eines Speicher-Bausteins ist die maximale Zeitdauer, die vom Anlegen einer Adresse an den Speicher bis zur Ausgabe der gewünschten Daten vergehen muss.	×	

## Aufgabe 7

1. MIPS-Assembler:

```
(a)      bne $s4, $s3, label
         add $s5, $s4, $s3
label:   ...
```

```
(b)      beq $s4, $s3, label1
         add $s5, $s4, $s3
         j label2
label1:  sub $s5, $s4, $s3
label2:  ....
```

```
(c)      slt $s5, $s3, $s4
```

2. Laden von 1111 0000 0011 1101 0000 1001 0000 1001 ins Register \$s0:

```
lui $s0, 1111 0000 0011 1101 # load upper immediate
ori $s0, 0000 1001 0000 1001
```

oder auch

```
lui $s0, 1111 0000 0011 1101
addi $s0, $s0, 0000 1001 0000 1001
```

3. Die 2 niedrigstwertigen Bits einer Wortadresse haben den Wert 0

4. Register- und Speicherinhalte nach der Ausführung:

Registersatz		Hauptspeicher	
Register	Inhalt	Adresse	Inhalt
\$t0	0x10	\$0x20	0x22
\$t1	<b>0x30</b>	\$0x24	0x30
\$t2	0x16	\$0x28	<b>0x30</b>
\$t3	<b>0x20</b>	\$0x2C	0x50
\$t4	<b>0x30</b>	\$0x30	0x60

## Aufgabe 8

### 1. Datenabhängigkeiten:

- Echte Abhängigkeiten (*True Dependence*)

$$\begin{array}{llll}
 S_1 \rightarrow S_3 (R1) & S_1 \rightarrow S_9 (R1) & & \\
 S_2 \rightarrow S_3 (R2) & S_2 \rightarrow S_4 (R2) & S_2 \rightarrow S_6 (R2) & \\
 S_3 \rightarrow S_6 (R3) & & & \\
 S_4 \rightarrow S_9 (R1) & & & \\
 S_5 \rightarrow S_7 (R4) & & & \\
 S_6 \rightarrow S_8 (R5) & & & 
 \end{array}$$

- Gegenabhängigkeiten (*Anti-Dependence*):  $S_3 \rightarrow S_4 (R1)$
- Ausgabe-Abhängigkeiten (*Output Dependence*):  $S_1 \rightarrow S_4 (R1)$

### 2. Behebung der Konflikte:

```

S1:   lw    R1, 1000(R0)
S2:   lw    R2, 1004(R0)
      NOP
      NOP
S3:   add   R3, R2, R1
S4:   addi  R1, R2, 8
S5:   subi  R4, R0, 2
S6:   and   R5, R3, R2
      NOP
S7:   sw    R4, 1000(R0)
S8:   sw    R5, 1004(R0)
S9:   sw    R1, 1008(R0)

```

## Aufgabe 9

1. Unterteilung der Hauptspeicheradresse:



2. Speicherbedarf:

Es sind für jede Zeile (Tag + 2 Statusbits + Daten pro Zeile) Bits erforderlich

Für den gesamten Cache:

$$(26 + 2 + 128) \cdot 16 = 156 \cdot 16 \text{ Bit} = 156 \cdot 2 \cdot 8 \text{ Bit} = 312 \text{ Byte}$$

3. Cache-Miss (**M**) oder Cache-Hit(**H**):

Adresse	Hit/Miss
0xA04	M
0x844	M
0xA07	H
0x7C0	M
0xC82	M
0xCC3	M
0x9E7	M
0x813	M
0x9E4	H
0x1B2	M
0x81D	H
0x7C7	H

4. *Dirty*-Bit: Kennzeichnet diejenigen Daten im Cache, die beim Verdrängen aus dem Cache in den Hauptspeicher zurückgeschrieben werden müssen, da sie gegenüber ihrem Original im Hauptspeicher verändert worden sind (Cache-Speicher mit *write back*-Schreibverfahren)

## Aufgabe 10

1.  $512 \times 8$ -Organisation: 512 Zellen mit 8-Bit Wörter  $\Rightarrow$  512 Zellen müssen adressiert werden. Dazu sind 9 Adressleitungen erforderlich.
2. Es sind 8 RAM-Bausteine der Organisation  $8k \times 1$  notwendig, um einen Speicher mit einer Kapazität von  $8k$  Wörter und einer Wortbreite von 8 Bit zu realisieren.
3. ROM-Baustein der Speicherkapazität von 2048 Bits und 8 Adressleitungen  $\Rightarrow$  Es können 256 Zellen adressiert werden  $\Rightarrow$   $256 \times 8$ -Organisation
4. Speicherhierarchie (Problemstellungen und Lösungsmöglichkeiten):
  - **Block-Abbildungsstrategie:** Wohin wird ein Block abgebildet?
    - Vollassoziative, 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*