



# Musterlösungen

zur Klausur „Technische Informatik I/II“  
am 13. September 2006, 9.00 - 11.00 Uhr

Name: <b>Bond</b>	Vorname: <b>James</b>	Matrikelnummer: <b>007</b>
----------------------	--------------------------	-------------------------------

<b>Technische Informatik I</b>	
Aufgabe 1	12 von 12 Punkten
Aufgabe 2	5 von 5 Punkten
Aufgabe 3	7 von 7 Punkten
Aufgabe 4	11 von 11 Punkten
Aufgabe 5	10 von 10 Punkten

<b>Technische Informatik II</b>	
Aufgabe 6	5 von 5 Punkten
Aufgabe 7	11 von 11 Punkten
Aufgabe 8	11 von 11 Punkten
Aufgabe 9	10 von 10 Punkten
Aufgabe 10	8 von 8 Punkten

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

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

## Aufgabe 1

1. DNF von  $y_1$ :

$$\begin{aligned}
 y_1 &= (x_0 \vee \bar{x}_1) \wedge (\bar{x}_0 \vee x_2) \\
 &= x_0 \bar{x} \vee x_0 x_2 \vee \bar{x}_1 \bar{x}_0 \vee \bar{x}_1 x_2 \\
 &= x_0 x_2 (x_1 \vee \bar{x}_1) \vee \bar{x}_1 \bar{x}_0 (x_2 \vee \bar{x}_2) \vee \bar{x}_1 x_2 (x_0 \vee \bar{x}_0) \\
 &= x_0 x_2 x_1 \vee x_0 x_2 \bar{x}_1 \vee \bar{x}_1 \bar{x}_0 x_2 \vee \bar{x}_1 \bar{x}_0 \bar{x}_2 \vee \bar{x}_1 x_2 x_0 \vee \bar{x}_1 x_2 \bar{x}_0 \\
 &= x_0 x_2 x_1 \vee x_0 x_2 \bar{x}_1 \vee \bar{x}_1 \bar{x}_0 x_2 \vee \bar{x}_1 \bar{x}_0 \bar{x}_2
 \end{aligned}$$

2. KNF von  $y_2$ :  $y_2$  ist bereits eine KNF!

$$y_2 = \bar{x}_0 \vee x_1 \vee \bar{x}_2$$

3. DMF von  $y_3$ :

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

$$y_3 = ba \vee cb \vee \bar{d} \bar{c} \bar{b}$$

4. (a) Die Schaltfunktion  $y_4$  ist unvollständig definiert.

Begründung:

- Primimplikanten bei vollständig definierten Funktionen überdecken  $2^n$  Minterme. Die Primimplikanten  $C$  und  $E$  überdecken jeweils 5 bzw. 3 Minterme, d. h. er müssen jeweils noch eine Freistelle enthalten. ODER
  - Die Primimplikant  $H$  und  $G$  sind im Primimplikanten  $C$  enthalten. Somit wären beide bei einer vollständig definierten Schaltfunktion keine Primimplikanten  $\rightarrow$  Widerspruch.
- (b) Alle disjunktiven Minimalformen von  $y_4$ :
- Kernprimimplikanten:  $B$  und  $D$ . Streichen der Minterme  $m_4$ ,  $m_{16}$  und  $m_{30}$ .
  - Zeilendominanz:  $G$  wird von  $C$  dominiert und kann gestrichen werden.  $H$  von  $C$  dominiert und kann ebenfalls gestrichen werden.

- Reduzierte Überdeckungstabelle:

	$m_0$	$m_3$	$m_{15}$	$m_{21}$	$m_{24}$	$m_{28}$
$A$	×		×		×	×
$C$	×		×	×		×
$E$		×		×		×
$F$	×	×			×	

- Spaltendominanz: Minterm  $m_{28}$  dominiert Minterm  $m_{21}$  und kann gestrichen werden.  $m_0$  dominiert  $m_{15}$  und kann gestrichen werden.
- Überdeckungsfunktion:

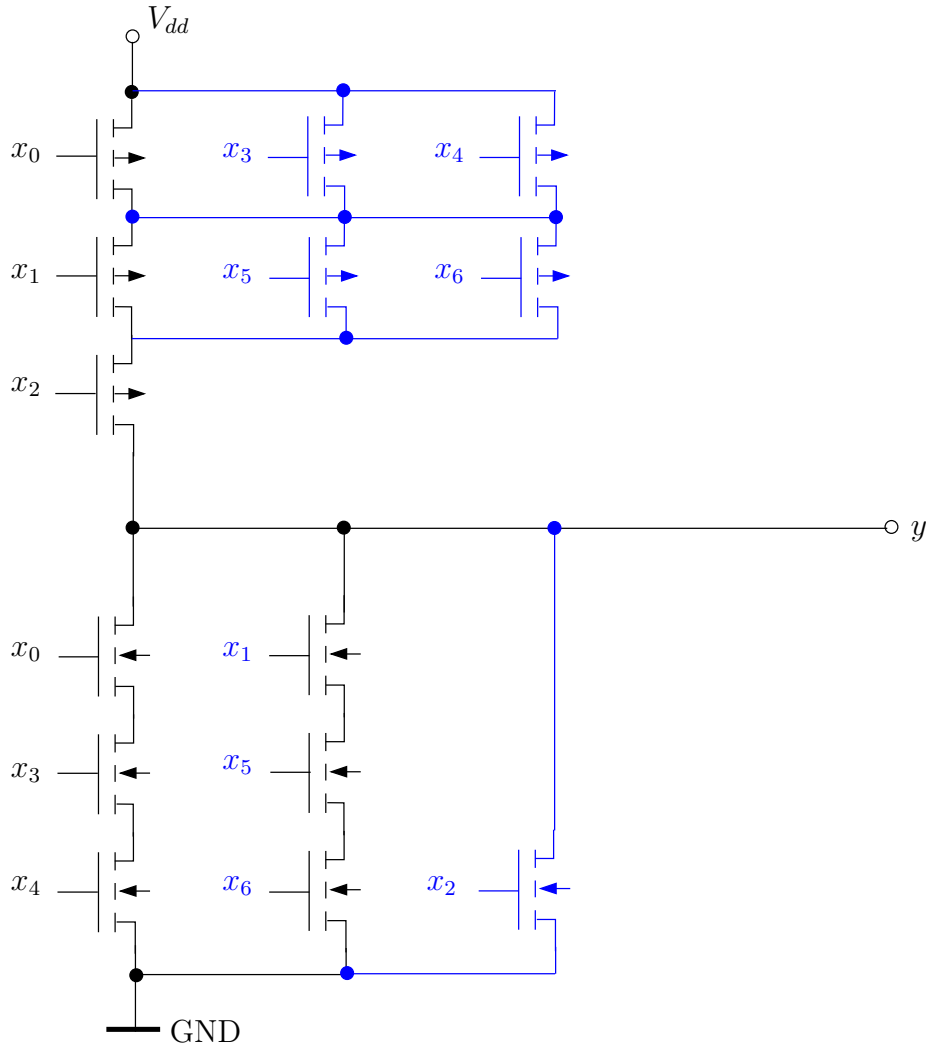
$$\begin{aligned}
 \ddot{u} &= (E \vee F)(A \vee C)(C \vee E)(A \vee F) \\
 &= (AC \vee AE \vee C \vee CE)(EA \vee EF \vee FA \vee F) \\
 &= (C \vee AE)(F \vee EA) \\
 &= CF \vee CEA \vee AEF \vee EA \\
 &= CF \vee EA
 \end{aligned}$$

- Disjunktiven Minimalformen:

$$y_4 = B \vee D \vee \begin{cases} E \vee A \\ C \vee F \end{cases}$$

## Aufgabe 2

1.

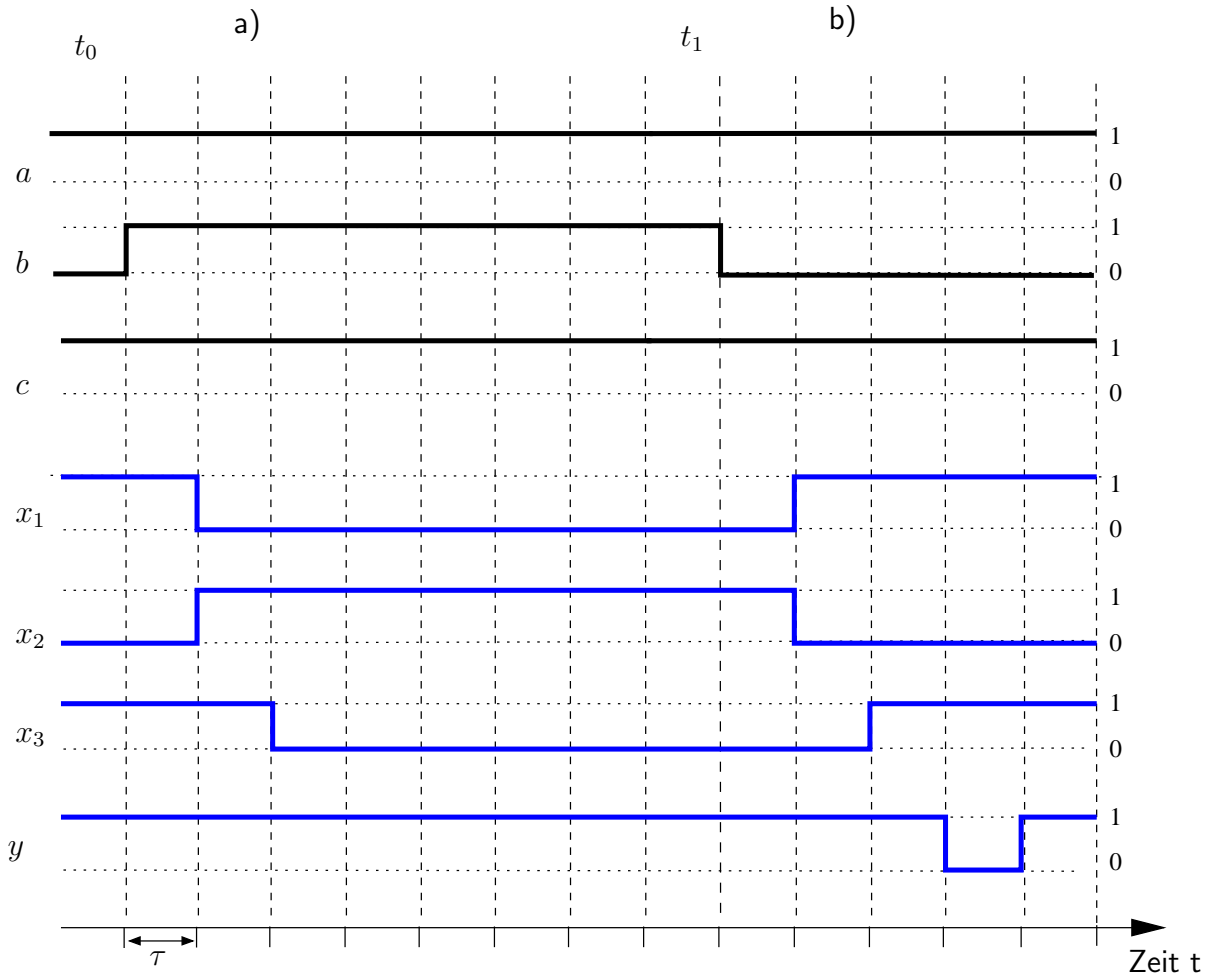


2. Realisierte Schaltfunktion:

$$\begin{aligned}
 y &= \overline{(x_0 \wedge x_3 \wedge x_4) \vee (x_1 \wedge x_5 \wedge x_6) \vee x_2} \\
 &= (\overline{x_0} \vee \overline{x_3} \vee \overline{x_4}) (\overline{x_1} \vee \overline{x_5} \vee \overline{x_6}) \overline{x_2}
 \end{aligned}$$

### Aufgabe 3

1.



2. Typ des Fehlers und Behebungsmöglichkeit:

Es tritt ein Hasardfehler beim Übergang  $B_7 \rightarrow B_5$  zum Zeitpunkt  $t_1$  auf.

Es handelt sich hierbei um einen Übergang, bei dem nur eine Variablen  $b$  ihren Wert wechselt  $\Rightarrow$  Der Übergang ist frei von Funktionshasards; der Hasardfehler tritt nicht aufgrund eines Funktionshasards auf und kann nur durch einen Strukturhasard bedingt sein  $\Rightarrow$  **1-statischer Strukturhasard**.

Behebung:

- Satz von Eichelberger: Realisierung der Schaltfunktion als die Disjunktion aller Primimplikanten (Fehlender Primimplikant  $c a$  in die Realisierung aufnehmen, d. h.  $y = b a \vee c \bar{b} \vee c a$ )
- Die beim Übergang konstant bleibenden Eingangsvariablen ( $a$  und  $c$ ) über ein zusätzliches UND-Gatter verknüpfen und das Ergebnis mit dem Ausgang des Schaltnetzes ODER-verknüpfen.

3. Übergang mit einem statischen 1-Funktionshasard:

	a			
	0 <small>0</small>	0 <small>1</small>	1 <small>5</small>	1 <small>4</small>
b	0 <small>2</small>	1 <small>3</small>	1 <small>7</small>	0 <small>6</small>
	c			

Beispiele für Übergänge mit Funktionshasard:  $B_4 \leftrightarrow B_7, B_4 \leftrightarrow B_7, B_5 \leftrightarrow B_6, B_0 \leftrightarrow B_7$ .  
 Begründung: Jeder Übergang, bei dem die zugehörige Folge der Funktionswerte nicht monoton ist, ist mit einem Funktionshasard behaftet.

### Aufgabe 4

1. (a) Automatentyp: Mealy-Automat

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

(b) Ansteuerfunktion:

$$d^t = (x \vee q)^t (\bar{x} \vee \bar{q})^t$$

Zustandsübergangsgleichung:

$$q^{t+1} = d^t = (x \vee q)^t (\bar{x} \vee \bar{q})^t$$

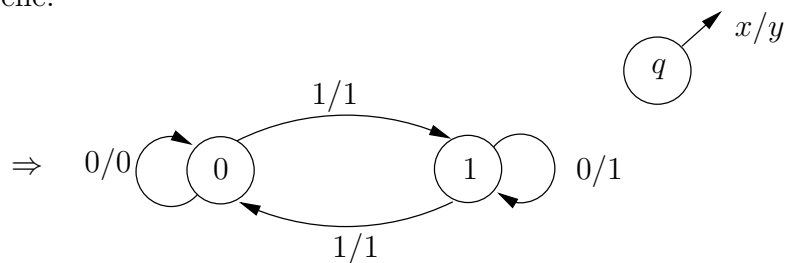
Ausgabefunktion:

$$y^t = (x \vee q)^t$$

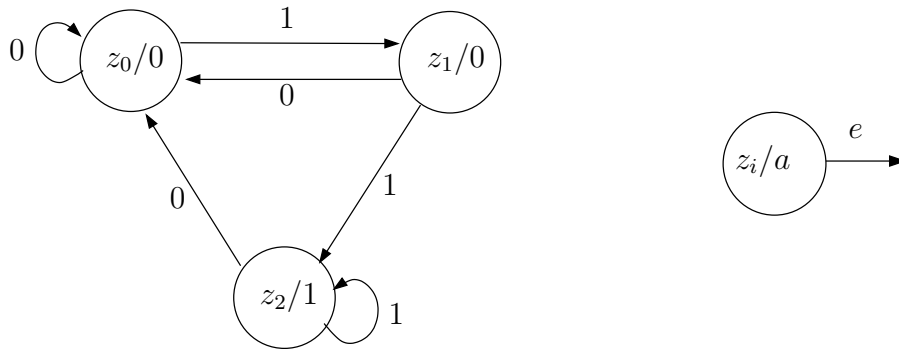
(c) Automatengraph des Schaltwerks:

Kodierte Ablaufabelle:

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



2. (a) Automatengraph des Schaltwerks:



(b) Zustandsübergangsgleichungen:

$q_1^t$	$q_0^t$	$e^t$	$q_1^{t+1}$	$q_0^{t+1}$
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	-	-

$$q_1^{t+1} = (e q_0 \vee e q_1)^t$$

$$q_0^{t+1} = (e \bar{q}_1 \bar{q}_0)^t$$

$q_1^{t+1}$ :

	$e$			
	0	0	1	0
$q_0$	0	1	-	-
	$q_1$			

$q_0^{t+1}$ :

	$e$			
	0	1	0	0
$q_0$	0	0	-	-
	$q_1$			

(c) Ansteuerfunktionen der Flipflops:

Charakteristische Gleichung des JK-Flipflops:  $q_i^{t+1} = (j_i \bar{q}_i \vee \bar{k}_i q_i)^t$

$$q_1^{t+1} = j_1 \bar{q}_1 \vee \bar{k}_1 q_1 = e q_0 \vee e q_1 = e q_0 (q_1 \vee \bar{q}_1) \vee e q_1$$

$$= e q_0 \bar{q}_1 \vee (e q_0 \vee e q_1)$$

$$= e q_0 \bar{q}_1 \vee e q_1$$

$$\Rightarrow j_1 = e q_0; \quad \bar{k}_1 = e \rightarrow k_1 = \bar{e}$$

$$q_0^{t+1} = j_0 \bar{q}_0 \vee \bar{k}_0 q_0 = e \bar{q}_1 \bar{q}_0$$

$$\Rightarrow j_0 = e \bar{q}_1; \quad \bar{k}_0 = 0 \rightarrow k_0 = 1$$

## Aufgabe 5

1. Gleitkommadarstellung von

$$(a) 10,75_{10} = 1010,11_2 = 1,01011_2 \cdot 2^3$$

$$Exp = 3 \rightarrow Char = 127 + 3 = 130_{10} = 1000\ 0010_2$$

$$Mantisse = 01011 \quad VZ = 0$$

	31	30		23	22				0
0	1000	0010	0101	1000	0000	...			000

$$(b) -0,125_{10} = -0,0010_2 = -1,0 \cdot 2^{-3}$$

$$Exp = -3 \rightarrow Char = 127 - 3 = 124_{10} = 0111\ 1100_2$$

$$Mantisse = 0 \quad VZ = 1$$

	31	30		23	22				0
1	0111	1100	0000	0000	...				000

2. (a) 1000 1111 1110 1111 1100 0000 0000 0000 als

- Zweierkomplement:

$$X = 10001111111011111100000000000000$$

$$-X = 01110000000100000100000000000000$$

$$= (2^{30} + 2^{29} + 2^{28} + 2^{20} + 2^{14}) \rightarrow$$

$$X = -(2^{30} + 2^{29} + 2^{28} + 2^{20} + 2^{14})$$

- Gleitkommazahl (IEEE-754):

$$VZ = 1$$

$$Char = 00011111_2 = 31_{10} \rightarrow Exp = Char - 127 = 31 - 127 = -96$$

$$Mantisse = 110111111000000000000000 \rightarrow$$

$$Zahl = (-1)^{VZ} \cdot (1, Mantisse) \cdot 2^{Exp}$$

$$= (-1)^1 \cdot (1, 110111111000000000000000) \cdot 2^{-96}$$

$$= -(1 + 2^{-1} + 2^{-2} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-7} + 2^{-8} + 2^{-9}) \cdot 2^{-96}$$

(b) 0000 0000 0000 0000 0000 0000 0000 0000 als

- Zweierkomplement: 0
- Gleitkommazahl (IEEE-754): +0,0

3. Ausnahmeregel für die Null im IEEE-754-Standard: Liegt an der Darstellung einer normalisierten Zahl, bei der die führende 1 nur implizit dargestellt wird. Auch wenn die Mantisse 0 ist, steht eine 1 vor dem Dezimalpunkt.

4. Datenwort:

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	1	1	0	1	1	1	0	1	1	1

Die Prüfbits lassen sich nach den folgenden Regeln berechnen:

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

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

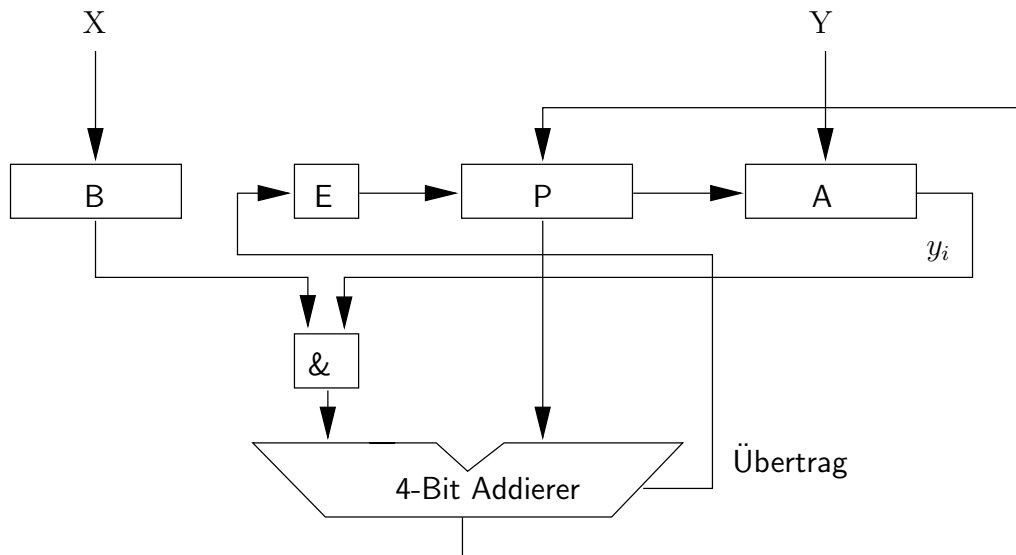
$$k_3 = k_3 \oplus m_2 \oplus m_3 \oplus m_4 \oplus m_8$$

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

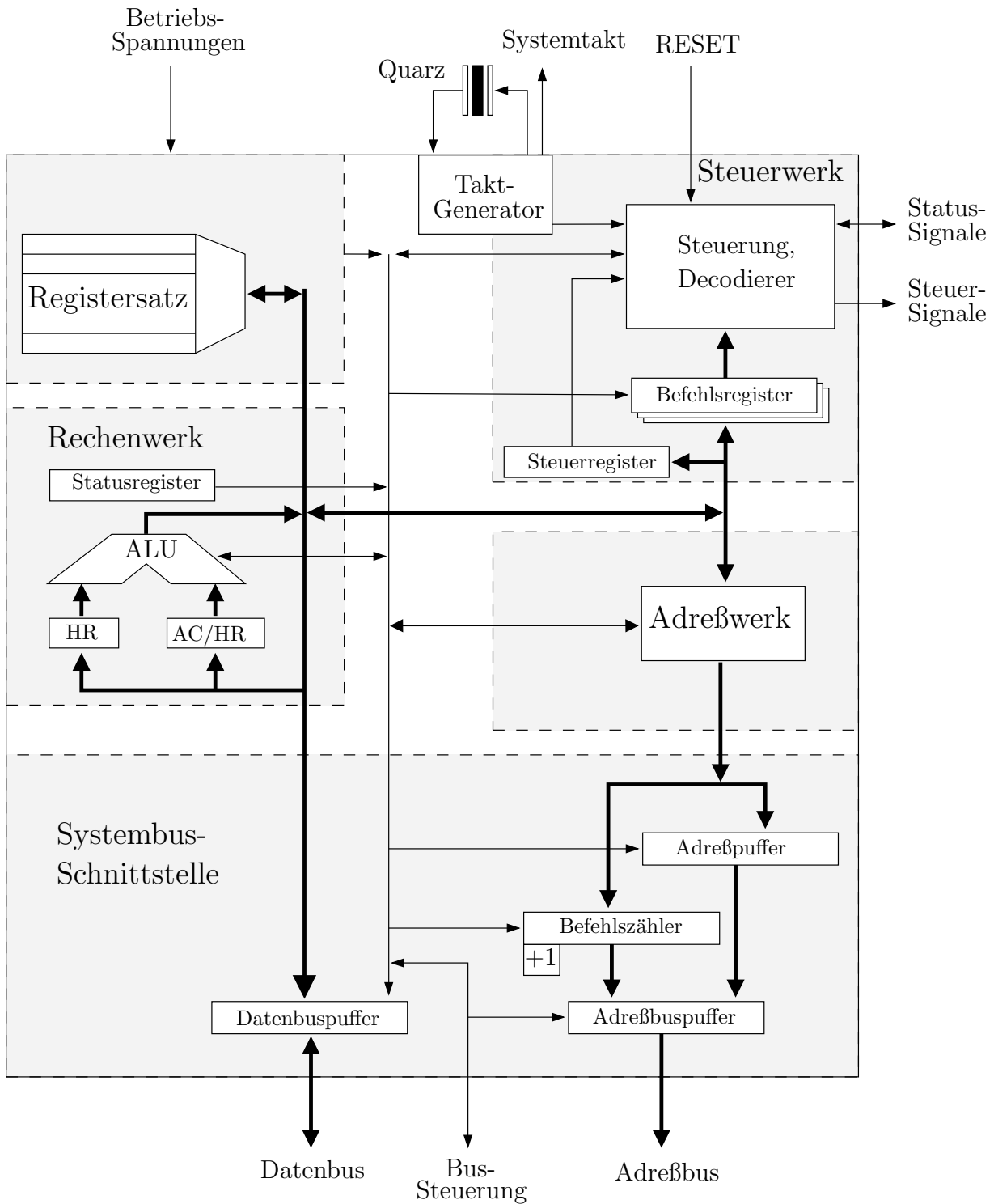
Codewort: **1 1 1 1 0 1 1 1 0 1 1 1**  $\Rightarrow k_4 k_3 k_2 k_1 = 0 0 0 0 \Rightarrow$

Das Codewort ist korrekt  $\Rightarrow$  Datenwort: **1 1 1 1 1 1 1 1**

5. Serieller Multiplizierer nach der PPS-Methode:



### Aufgabe 6



## Aufgabe 7

1. Code in MIPS-Assembler:

```

        la    $t0,a           # $t0 = &a[0]
        move  $a0,0           # sum=0
        move  $a1,0           # i=0
        move  $a3,10          # $a3 = 10
loop:   mult  $a2,$a1,4       # $a2=i*4
        addu  $t1,$t0,$a2     # $t1=&a[i]
        lw    $t2,0($t1)      # $t2=a[i]
        add   $a0,$a0,$t2     # sum=sum+a[i]
        addi  $a1,$a1,1       # i++
        blt   $a1,$a3,loop    # if i<10 goto loop
    
```

2. Fehlerfreie Version:

```

        add   $v0, $zero, $zero # $v0 mit 0 initialisieren
loop:   lw    $v1, 0($a0)       # nächstes Wort lesen
        sw    $v1, 0($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)
        b     loop
done:
    
```

3.

	Registersatz		Hauptspeicher	
	Register	Inhalt	Adresse	Inhalt
ori \$t1, \$zero, 0x20	\$t0	0xFFFF0000	\$0x20	0x10
lw \$t2, 0x0(\$t3)	\$t1	0x20	\$0x24	0x30
add \$t4, \$t3, \$t1	\$t2	0xFFFFF0E0	\$0x28	0x40
sw \$t4, 0x8(\$t3)	\$t3	0x28	\$0x2C	0x50
sub \$t2, \$t1, \$t2	\$t4	0x48	\$0x30	0x48
lui \$t0, 0xFFFF				

4.

(a)

Die Befehlsfolge a)	<i>richtig</i>	<i>falsch</i>
berechnet das Einerkomplement von \$t0.		×
berechnet das Einerkomplement von \$t1.		×
vertauscht die Inhalte der Register \$t0 und \$t1.	×	
verändert die Inhalte der Register \$t0 und \$t1 nicht.		×

(b)

Die Befehlsfolge b)	<i>richtig</i>	<i>falsch</i>
berechnet 4!		×
berechnet 6!	×	
berechnet 5!		×
berechnet $6 \times 5 \times 4 \times 3 \times 2 \times 1 \times 0$		×

(c)

Die Befehlsfolge c)	<i>richtig</i>	<i>falsch</i>
multipliziert den Inhalt von \$t1 mit 256.		×
multipliziert den Inhalt von \$t2 mit 4.		×
Schreibt den Wert ( $t0 \times 260$ ) in \$t3.	×	
Schreibt den Wert ( $t0 \times 261$ ) in \$t0.	×	

## Aufgabe 8

### 1. Datenabhängigkeiten:

- Echte Abhängigkeiten:

$$\begin{array}{ll} S_1 \rightarrow S_2 (\$t1) & S_1 \rightarrow S_3 (\$t1) \\ S_2 \rightarrow S_3 (\$t2) & S_2 \rightarrow S_5 (\$t2) \\ S_3 \rightarrow S_4 (\$t1) & S_3 \rightarrow S_5 (\$t1) \end{array}$$

- Gegen-Abhängigkeiten:

$$S_2 \rightarrow S_4 (\$t1) \quad S_3 \rightarrow S_4 (\$t1)$$

- Ausgabe- Abhängigkeiten:

$$S_1 \rightarrow S_4 (\$t1)$$

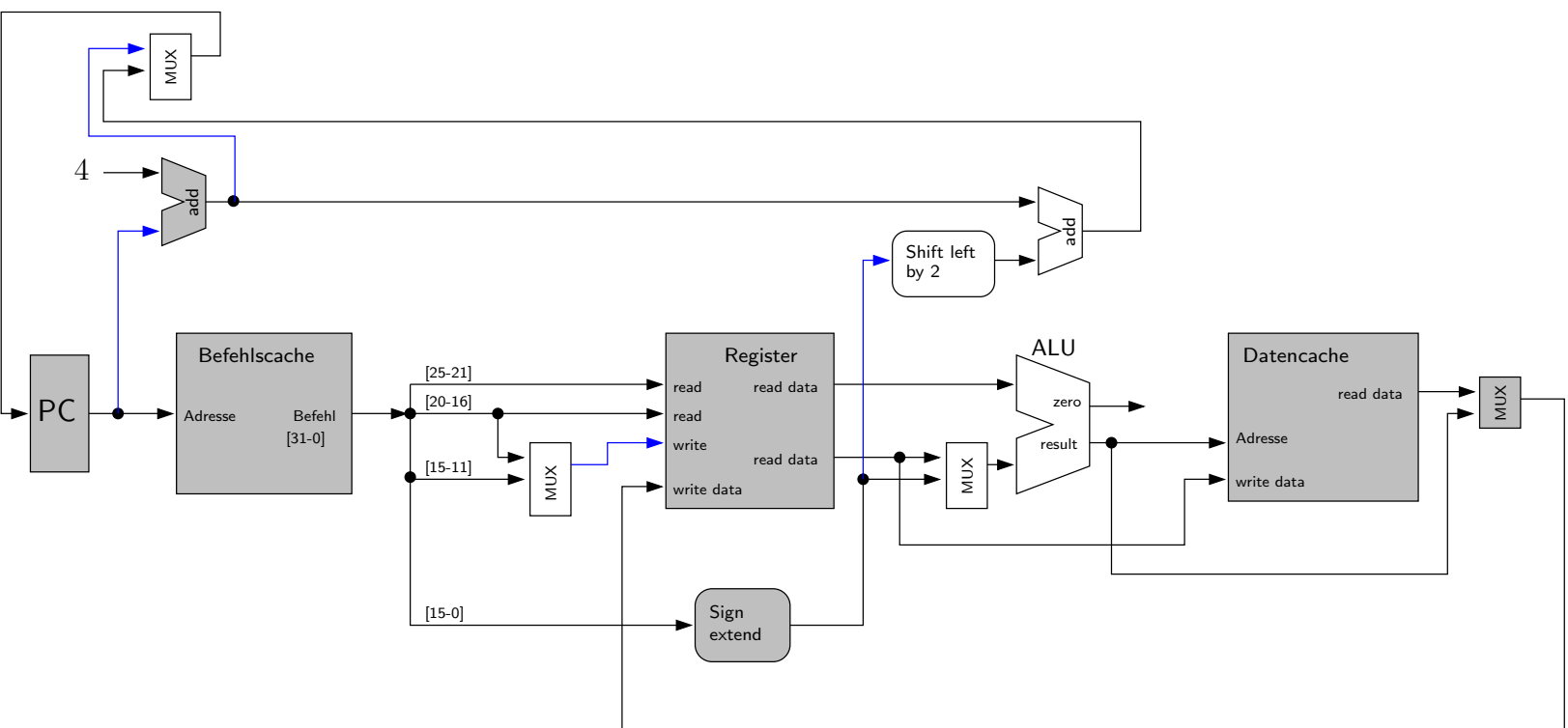
### 2. Beseitigung der Datenkonflikte:

```

S1:    anfang:  andi $t2, $t1, 1
                NOP
                NOP
S2:                beqz $t2, weiter
                NOP
                NOP
                NOP
S3:                subi $t1, $t1, 1
S4:                j anfang
                NOP
                NOP
                NOP
S5:    weiter:  srli $t1, $t1, 1
S6:                j anfang
                NOP
                NOP
                NOP
S7:                addi $t3, $t0, 1

```

3.



## Aufgabe 9

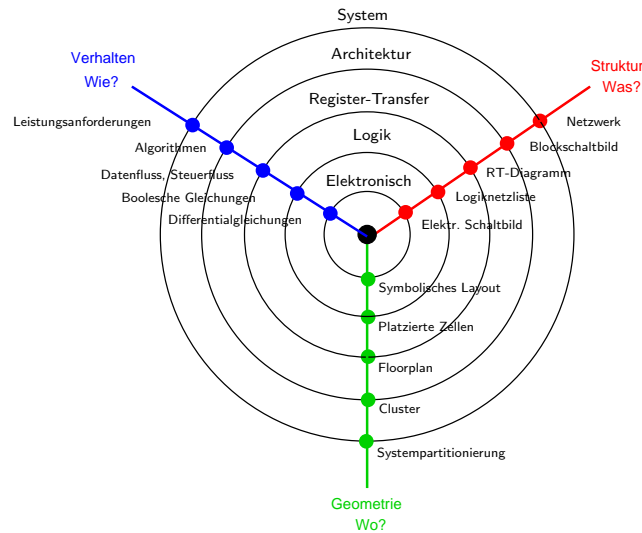
1. (a) Größe eines Cache-Blocks in Byte:  
 5 Bits Byte-Offset  $\rightarrow$  Blockgröße:  $2^5 = 32$  Byte
  
  - (b) Kapazität des Cache-Speichers:  
 11 Bits Index  $\rightarrow 2^{11}$  Sätze  
 A2-Cache  $2 \times 2^{11} = 2^{12}$  Cache-Blöcke mit je 32 Bytes  
 Cache-Kapazität:  $2^{12} \times 32 = 2^{12} \times 2^5$  Byte =  $2^{17}$  Byte = 128 KByte
  
  - (c) Der insgesamt erforderliche Speicherbedarf:  
 Kapazität + (Tag-Länge + 2 Statusbits)  $\times$  (Anzahl der Cache-Blöcke)  
 $128$  KByte +  $(16 + 2) \cdot 2^{12}$  Bit =  $128$  KByte +  $2 \cdot 2^{12}$  Byte +  $2 \cdot 2^{12}$  Bit =  
 $128$  KByte + 8 KByte + 1 KByte = 137 KByte
  
  - (d) Zugriff auf die Adresse 0x00EF1A34:  
 A2-Cache  $\rightarrow$  Es wird ein Vergleich mit 2 Zeilen im durchgeführt.  
 Satz-Index =  $0001\ 1010\ 001_2 = 209_{10}$   
 $\rightarrow$  Der Vergleich wird mit den Zeilen 418 und 419 durchgeführt.
- 2.

Adresse	0x44	0xA0	0xC3	0x9E	0x66	0x2D	0x6B	0x49
read/write	w	r	w	r	r	w	r	w
Hit/Miss	×	—	—	×	×	—	×	—
write back?	nein	nein	ja	nein	nein	nein	nein	ja

## Aufgabe 10

### 1. 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.



2. TLB: *Translation-Lookaside-Buffer (TLB)*: Ein schneller vlassoziativer Cache zur Beschleunigung der Umsetzung virtueller in physikalischen 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.

	<i>richtig</i>	<i>falsch</i>
Besteht ein Assemblerprogramm aus mehreren Dateien, so muss der Assembler alle diese Dateien zur Assemblierungszeit kennen.		×
Mit dem Befehl jr (jump register) können $2^{32}$ Bytes adressiert werden; mit dem Befehl j (jump) dagegen $2^{26}$ Worte.	×	
Je höher die Assoziativität eines Cache-Speichers, desto größer ist die Anzahl der notwendigen Index-Bits.		×
Bei einem virtuellen Cache müssen mehr Bits als <i>Tag</i> gespeichert werden als bei einem physischen Cache	×	
Die notwendige Abbildungsinformation bei der Speicherverwaltung stellt das Betriebssystem zur Verfügung	×	
Das Seitenwechsel-Verfahren spiegelt die logische Programmstruktur wieder		×
Bei einem asynchronen Bus werden Daten ohne Verwendung eines Taktsignals übertragen.	×	
Eine Bus-Arbitrierung nach dem <i>Daisy-Chain</i> -Prinzip erlaubt eine „faire“ Buszuteilung für alle Komponenten		×