KIT-Fakultät für Informatik
Prof. Dr. Tamim Asfour, Prof. Dr. Uwe D. Hanebeck, Prof. Dr. Jörg Henkel

Musterlösungen zur Klausur

Digitaltechnik und Entwurfsverfahren & Rechnerorganisation

und

Technische Informatik I/II

am 26. Februar 2016, 14:00 – 16:00 Uhr

Name:	Vorname:	Matrikelnummer:
Bond	James	007

Digitaltechnik und Entwurfsverfahren/TI-1	
Aufgabe 1	7 von 7 Punkten
Aufgabe 2	10 von 10 Punkten
Aufgabe 3	8 von 8 Punkten
Aufgabe 4	11 von 11 Punkten
Aufgabe 5	9 von 9 Punkten

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

Gesamtpunktzahl:	90 von 90 Punkten

	Note: **1,0**

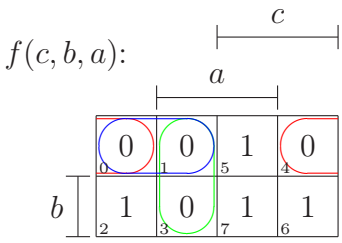
Aufgabe 1

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

   (Ablesen aus der Tabelle)

$$\begin{aligned} f(c,b,a) &= \text{MINt}(2,5,6,7) \\ &= (\overline{c}\ b\ \overline{a}) \vee (c\ \overline{b}\ a) \vee (c\ b\ \overline{a}) \vee (c\ b\ a) \end{aligned}$$

2. KV-Diagramm:



   Primimplikate:

   - Kernprimimplikate: $b \vee a$, $c \vee \overline{a}$
   - Entbehrliche Primimplikate: $c \vee b$

   KMF von $f(c, b, a)$: $(b \vee a) \wedge (c \vee \overline{a})$

3. Schaltnetz:

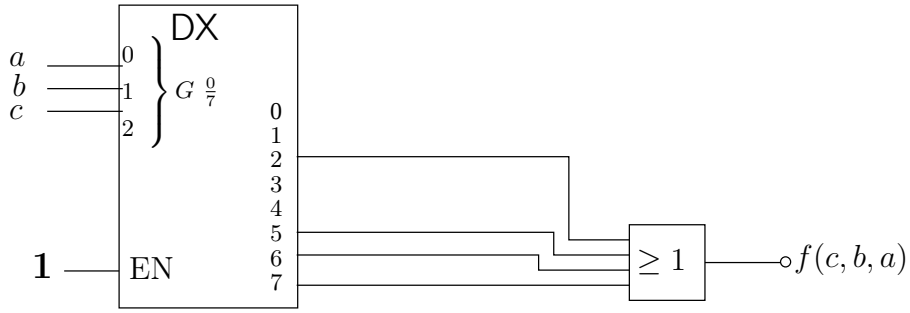


Abbildung 1: Schaltnetz

4. Schaltnetz:

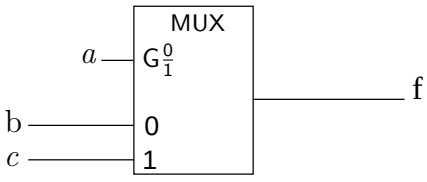


Abbildung 2: Schaltnetz

Aufgabe 2

1. y :

$$\begin{aligned} y &= \overline{d}\,\overline{a}\,(1) \vee \overline{d}\,a\,(\overline{c}) \vee d\,\overline{a}(0) \vee d\,a\,(\overline{c} \vee b) \\ &= \overline{d}\,\overline{a} \vee \overline{d}\,a\,\overline{c} \vee d\,a\,\overline{c} \vee d\,a\,b \end{aligned}$$

2. Minimalform von y:

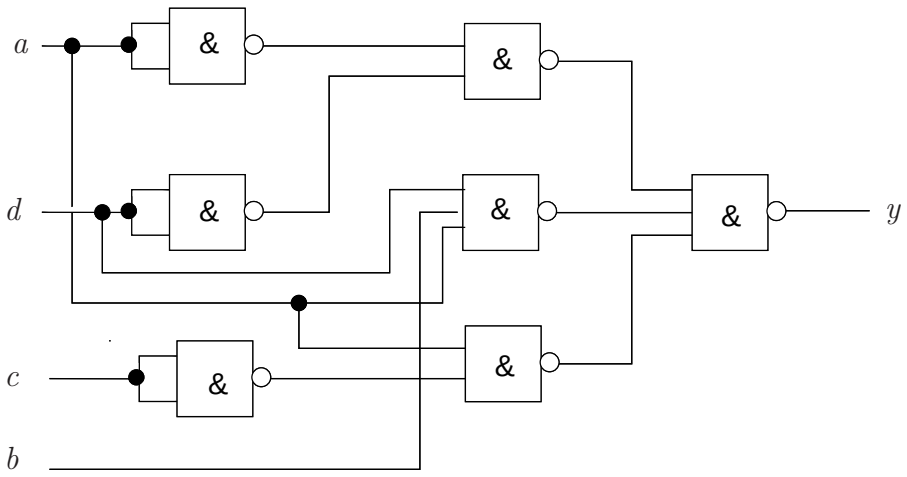
$$\begin{aligned} y &= \overline{d}\,\overline{a} \vee \overline{d}\,a\,\overline{c} \vee d\,a\,\overline{c} \vee d\,a\,b \\ &= \overline{d}\,\overline{a} \vee d\,b\,a \vee \overline{c}\,a\,(d \vee \overline{d}) \\ &= \overline{d}\,\overline{a} \vee d\,b\,a \vee \overline{c}\,a \end{aligned}$$

3. Minimalform von y in NAND-Form:

$$\begin{aligned} y &= \overline{\overline{\overline{d}\,\overline{a} \vee d\,b\,a \vee \overline{c}\,a}} \\ &= \overline{\overline{\overline{d}\,\overline{a}} \wedge \overline{d\,b\,a} \wedge \overline{\overline{c}\,a}} \\ &= \mathrm{NAND}_3\left(\mathrm{NAND}_2(\overline{d},\overline{a}),\ \mathrm{NAND}_3(d,b,a), \mathrm{NAND}_2(\overline{c},a)\right) \end{aligned}$$

$$(\overline{x} = x \,\overline{\wedge}\, x)$$

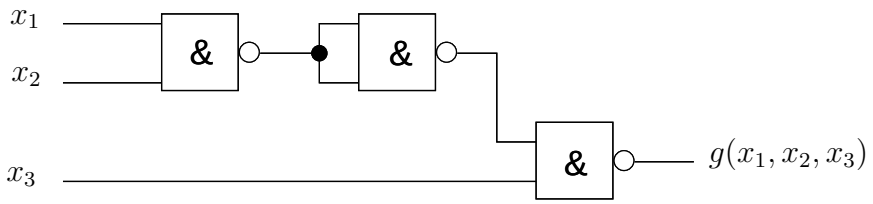
Schaltnetz:



4. Realisierung von $g(x_1, x_2, x_3)$ mit NAND-Gattern:

$$\begin{aligned} g(x_1, x_2, x_3) &= \overline{x_1 \ \wedge \ x_2 \ \wedge x_3} \\ &= \overline{(x_1 \ \wedge \ x_2) \ \wedge x_3} \\ &= (x_1 \ \wedge \ x_2) \ \overline{\wedge} \ x_3 \\ &= \overline{\overline{(x_1 \ \wedge \ x_2)} \ \overline{\wedge} \ x_3} \\ &= \overline{(x_1 \ \overline{\wedge} \ x_2) \ \overline{\wedge} \ x_3} \end{aligned}$$

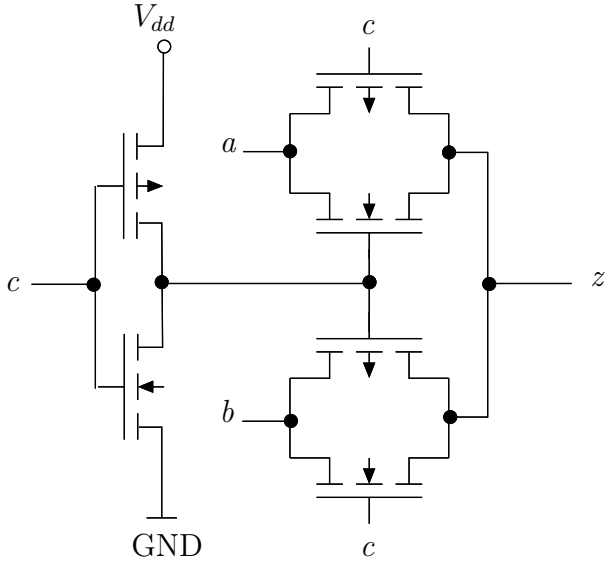
Schaltbild:



5. CMOS-Realisierung eines 2:1-Multiplexers:

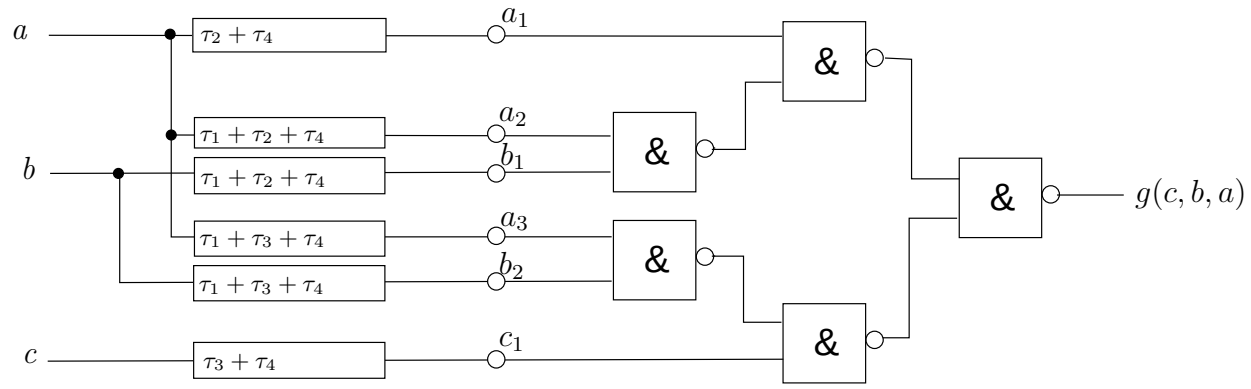
Schaltfunktion: $z(c, b, a) \ = \ \overline{c} \ a \ \vee c \ b$

CMOS-Schaltbild:

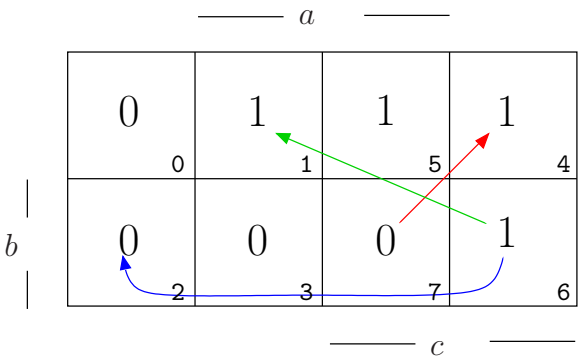


Aufgabe 3

1. Totzeitmodell:



2. KV-Diagramm für g:



3.
   - Übergang 1 $(c, b, a): \ (1, 1, 0) \to (0, 1, 0)$
     Bei dem Übergang ändert nur eine Variable den Wert $\Rightarrow$ frei von Funktionshasards, da die zugehörige Folge der Funktionswerte (hier $1 \to 0$) immer monoton ist.

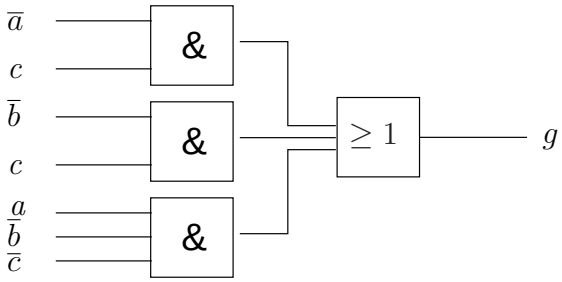
   - Übergang 2 $(c, b, a): \ (1, 1, 0) \to (0, 0, 1)$
     Bei dem Übergang ändern 3 Variablen den Wert $\Rightarrow 3! = 6$ Wege von der Anfangs- zur Endbelegung. Von den zugehörigen Folgen der Funktionswerte ist mindestens eine, z. B. $B_6 \to B_7 \to B_3 \to B_1$ nicht monoton.
     Deshalb ist der Übergang ist mit einem statischen 1-Funktionshasards behaftet.

4. Übergang (c, b, a) : $(1, 1, 1) \rightarrow (1, 0, 0)$

   Der Strukturhasard kann genau dann durch ein zweistufiges disjunktives Schaltnetz vermieden werden, wenn jeder Einsblock, der in den Übergangsbreich hineinragt, die Einsbelegung des Übergangs umschließt. Der Übergangsbereich ist $(1, -, -) \Rightarrow$ der Einsblock $\overline{b}\,a$ darf nicht in die Disjunktion aufgenommen werden.

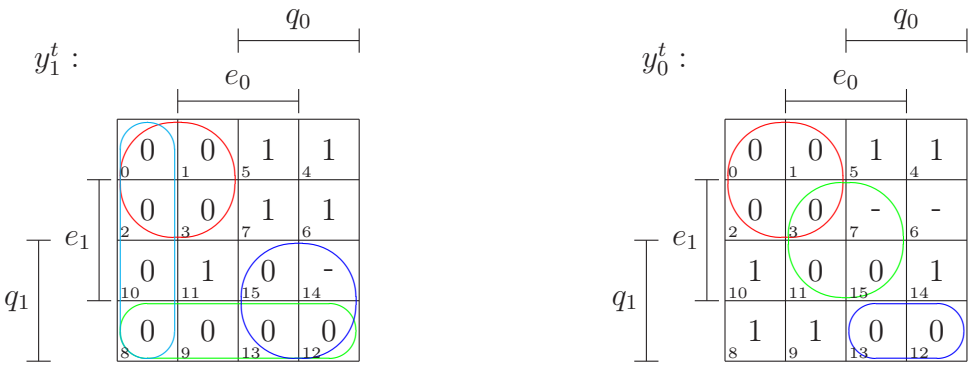
   Schaltnetz:

$$g(c, b, a) \;=\; c\,\overline{a} \vee c\,\overline{b} \vee \overline{c}\,\overline{b}\,a$$



Aufgabe 4

1. KMF der Ausgabefunktionen:

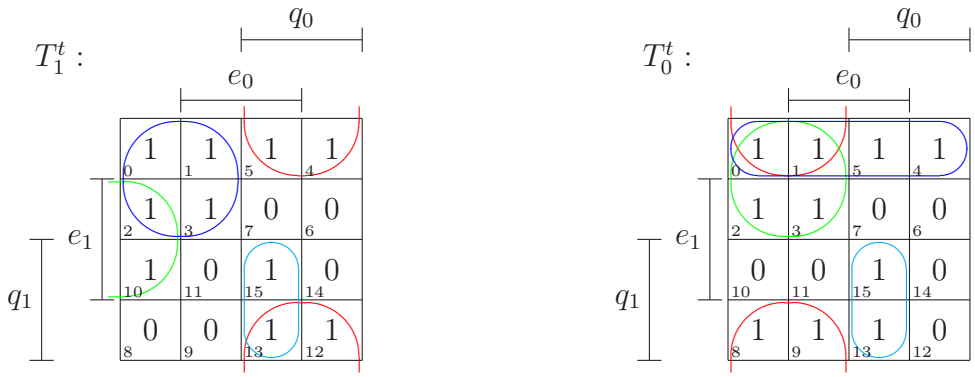


$$y_1 = (q_1 \vee q_0)(q_0 \vee e_0)(\overline{q}_1 \vee e_1)(\overline{q}_1 \vee \overline{q}_0)$$

$$y_0 = (q_1 \vee q_0)(\overline{e}_1 \vee \overline{e}_0)(\overline{q}_1 \vee \overline{q}_0 \vee e_1)$$

2. DMF der Ansteuerfunktionen:

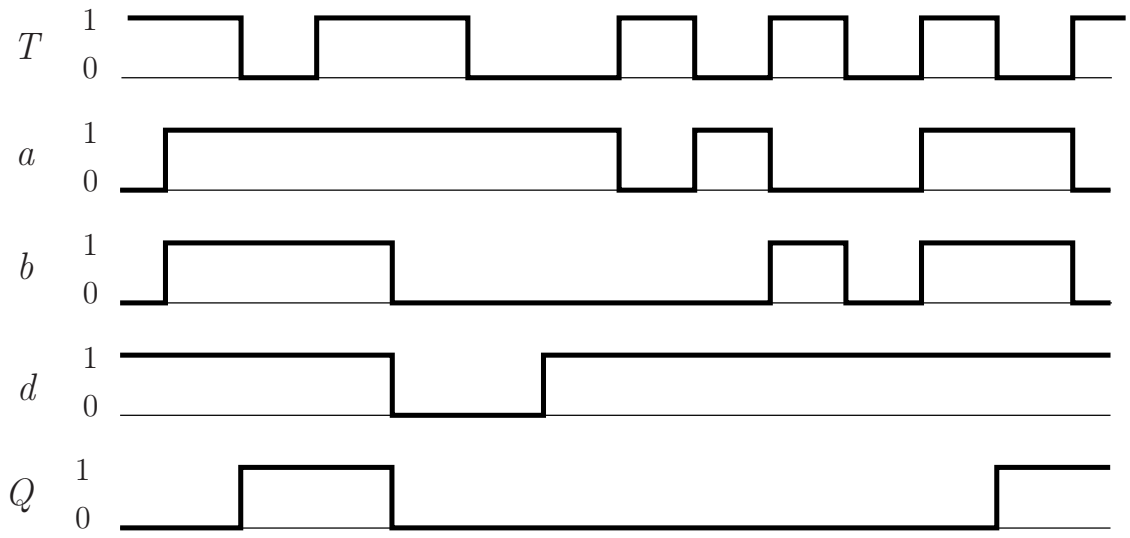
q_1^t	q_0^t	e_1^t	e_0^t	q_1^{t+1}	q_0^{t+1}	y_1^t	y_0^t	T_1^t	T_0^t
0	0	–	–	1	1	0	0	1	1
0	1	0	–	1	0	1	1	1	1
0	1	1	–	0	1	1	–	0	0
1	0	0	–	1	1	0	1	0	1
1	0	1	0	0	0	0	1	1	0
1	0	1	1	1	0	1	0	0	0
1	1	0	0	0	1	0	0	1	0
1	1	1	0	1	1	–	1	0	0
1	1	–	1	0	0	0	0	1	1



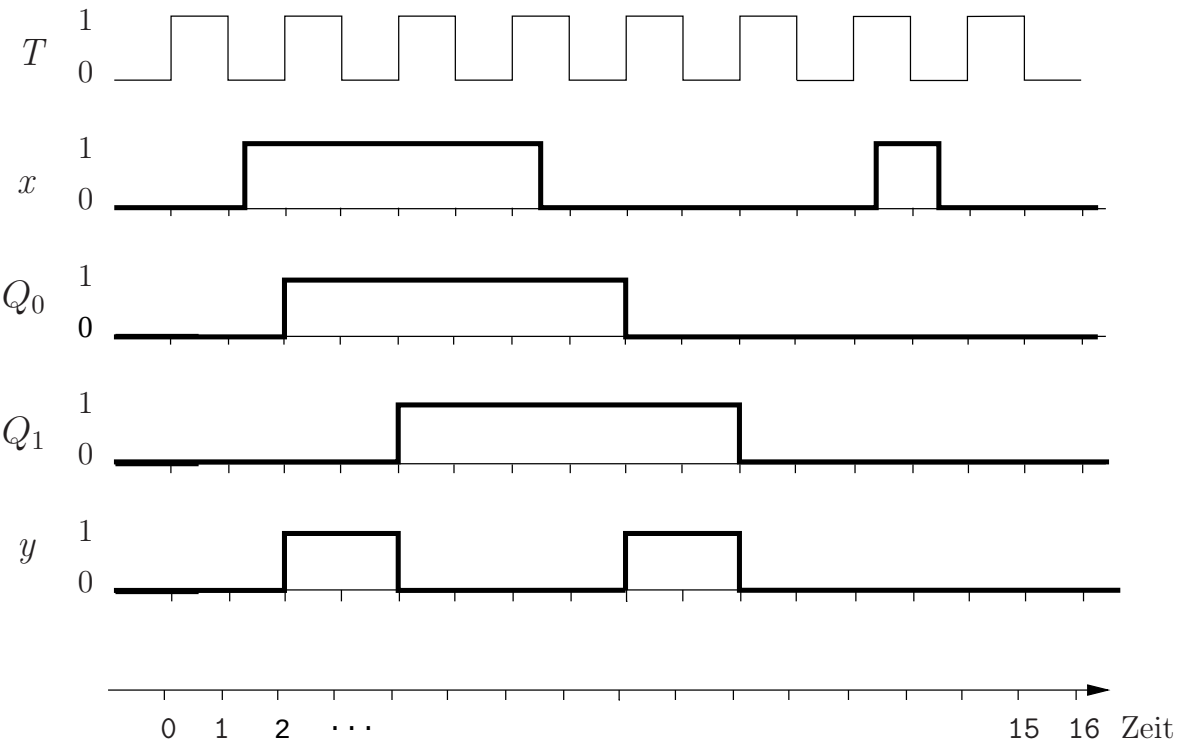
$$T_1 = \overline{q}_1\overline{q}_0 \vee q_0\overline{e}_1 \vee \overline{q}_0 e_1\overline{e}_0 \vee q_1 q_0 e_0$$

$$T_0 = \overline{q}_1\overline{q}_0 \vee \overline{q}_1\overline{e}_1 \vee \overline{q}_0\overline{e}_1 \vee q_1 q_0 e_0$$

3. Verlauf von Q:



4. Verläufe von Q_0, Q_1 und y:



5. Funktion des Schaltwerkes: sog. synchronen Änderungsdetektor.

   Das Schaltwerk detektiert Änderungen am Eingang x (sowohl 01-Wechsel als auch 10-Wechsel) und liefert einen taktsynchronen Ausgangsimpuls y, dessen Dauer eine Taktperiode beträgt. Impluse von x, die kürzer als eine Taktperiode sind, werden vom Schaltwerk jedoch nicht erfasst.

Aufgabe 5

1. Anzahl der Prüfbits:

   Aufwand: $2^k \geq m + k + 1$. Hier: m $= 200 \Rightarrow k = 8$

2. *Carry Ripple*-Addierer und *Carry Lookahead*-Addierer:

   Bei *Carry Ripple*-Addierern muss bei der Addition einer Stelle auf den Übertrag aus den vorhergehenden Stelle gewartet werden. Die Additionszeit ist proportional zur Anzahl der Stellen.

   Bei *Carry Lookahead*-Addierern werden alle Überträge direkt aus den Eingangsvariablen berechnet.

3. -70 als 7-Bit-Zweierkomplement Zahl: $+70 = 100\ 0110_2 \Rightarrow$ Es sind mindestens 8 Bit zur Darstellung von -70 als Zweierkomplementzahl notwendig.

   - -70 mit minimaler Bitanzahl: $+70 = 0100\ 0110 \Rightarrow -70 = 1011\ 1001 + 1 = 1011\ 1010$
   - -70 als 16-Bit Zweierkomplementzahl: 1111 1111 1011 1010

4. 1000 0011 0101 1000 0000 0000 0000 0101

   (a) BCD: 83 580 005

   (b) Vorzeichenlose Dualzahl: $2^{31} + 2^{25} + 2^{24} + 2^{22} + 2^{20} + 2^{19} + 2^2 + 2^0$

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

$$
\begin{aligned}
VZ &= 1 \\
Char &= 000\ 0011\ 0 = 6 \\
Exp &= Char - 127 = -121 \\
M &= 101\ 1000\ 0000\ 0000\ 0000\ 0101 \Rightarrow \\
\\
Z &= (-1)^1 \cdot (1,101\ 1000\ 0000\ 0000\ 0000\ 0101) \cdot 2^{-121} \\
&= -(1 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-21} + 2^{-23}) \cdot 2^{-121}
\end{aligned}
$$

Aufgabe 6

1. MIPS-Assembler:

   (a)

```
            bne  $s1, $s2, label
            sub  $s3, $s1, $s2
label:      ...
```

   (b)

```
            beq   $s1, $s2, label1
            sub   $s3, $s1, $s2
            j     label2
label1:     move  $s3, $s2
label2:     ...
```

   (c)

```
            slt  $s3, $s2, $s1
```

2. Register- und Speicherinhalte nach der Ausführung:

Registersatz

Register	Inhalt
$t0	0x30
$t1	**0x34**
$t2	0x300
$t3	**0x24**
$t4	**0x54**

Hauptspeicher

Adresse	Inhalt
$0x20	0x24
$0x24	0x30
$0x28	0x34
$0x2C	**0x54**
$0x30	0x44

3. Das MIPS-Befehlsformat sieht eine Breite von 32 Bit für ein Befehlswort vor.

4. Unterschied Maschinensprache zu Assemblersprache:

   Maschinensprache ist eine Repräsentation von Anweisungen, die für einen Mikroprozessor unmittelbar verständlich sind.

   Assemblersprache ist eine symbolische Repräsentation der Maschinensprache, die für den Menschen verständlich und (anschaulich) ist.

Aufgabe 7

1. Wert der Speicherstelle `0x10010000` nach sequentieller Ausführung:

$$2016 \cdot 5 - 6080 = 10080 - 6080 = 4000$$

2. Datenabhängigkeiten:
   - Echte Abhängigkeiten:
     - S1-S3 (`$s1`)
     - S1-S5 (`$s1`)
     - S2-S3 (`$s0`)
     - S2-S6 (`$s0`)
     - S3-S4 (Lo-Register)
     - S4-S5 (`$s1`)
     - S5-S6 (`$s0`)
   - Gegenabhängigkeiten:
     - S3-S4 (`$s1`)
     - S3-S5 (`$s0`)
   - Ausgabeabhängigkeiten:
     - S1-S4 (`$s1`)
     - S2-S5 (`$s0`)

3. Ausführung in einer Standard DLX-Pipeline

```
S1
S2
NOP
NOP
S3
NOP
NOP
S4
NOP
NOP
S5
NOP
NOP
S6
```

4. Ausführung in einer DLX-Pipeline mit Load und Result Forwarding

```
S1
S2
NOP
S3
S4
S5
S6
```

5. Berechnung der Ausführungszeit:

   - Sequentielle Ausführung: $6 \cdot 5 = 30$ Taktzyklen
   - DLX-Pipeline ohne Forwarding: $14 + 4 = 18$ Taktzyklen
   - DLX-Pipeline mit Forwarding: $7 + 4 = 11$ Taktzyklen

Aufgabe 8

1. (a) Blockgröße in Bytes: 5 Bit Byte-Offset $\Rightarrow$ Blockgröße $= 2^5 = 32$ Byte

   (b) Anzahl der Einträge:

   $$\text{Anzahl der Einträge} = \frac{\text{Kapazität}}{\text{Blockgröße}} = \frac{512\ KByte}{32\ Byte} = 16\ K \text{ Einträge}$$

   (c) Cache-Organisation:

   12 Bit Index-Feld $\Rightarrow$ 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:

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

   - Daten pro Zeile 8 Byte $\times$ 8 = 64 Bit
   - Tag = 32 - 8 - 3 = 21 Bit (8 Bit Satzindex und 3 Byte-Offset)

   Speicherbedarf für eine Zeile: $21 + 1 + 64$ Bit = 86 Bits
   Speicherbedarf für den gesamten Cache: $86 \cdot 256 \cdot 2$ = 44032 Bits = 5504 Byte

3.

Adresse	**0**	**32**	**96**	**16**	**112**	**32**	**16**	**112**	**64**	**0**
read/write	r	r	r	r	r	r	r	w	w	r
Index	0	2	2	1	3	2	1	3	0	0
Tag	0	0	1	0	1	0	0	1	1	0
Hit/Miss	Miss	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Hit

Aufgabe 9

1. Unterteilung der virtuellen Adresse: 1 P.

2. Physikalische Adressen: 4 P.

Virtuelle		Physikalische	
Adresse	Seitennummer	Seitennummer	Adresse
1024	0	2	$2 \cdot 2048 + 1024 = \mathbf{5120}$
2047	0	2	$2 \cdot 2048 + 2047 = \mathbf{6143}$
2048	1	0	$0 \cdot 2048 + 0 = \mathbf{0}$
2102	1	0	$0 \cdot 2048 + 54 = \mathbf{54}$
4095	1	0	$0 \cdot 2048 + 2047 = \mathbf{2047}$
4096	2	–	*page fault*
8192	4	1	$1 \cdot 2048 + 0 = \mathbf{2048}$
8202	4	1	$1 \cdot 2048 + 10 = \mathbf{2058}$

3. Eine Beschleunigung der Adressumsetzung durch den *TLB* wird erst beim zweiten Zugriff auf eine Seite und solange die entsprechenden Einträge aus dem Seitentabellen-Verzeichnis und der Seitentabelle aus dem TLB nicht verdrängt wurden erreicht. 1 P.

4. Breite des *Tags*: 2 P.

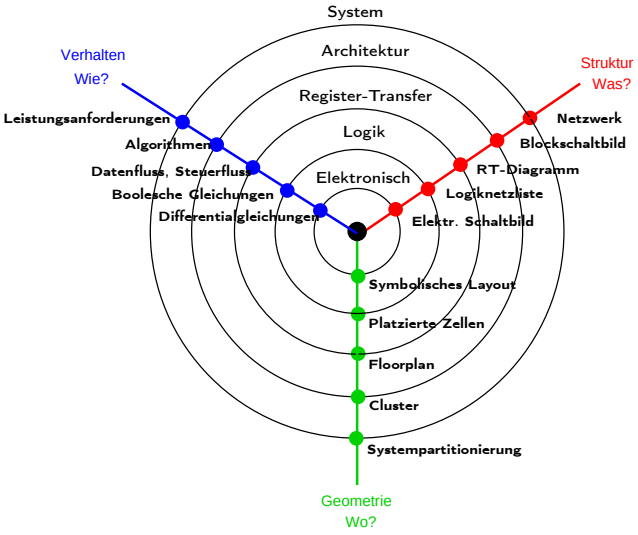
   Seitengröße ist 4 KByte $\Rightarrow$ Byte-Offset ist 12 Bit breit.

   Der Tag ist dann $(n - 12)$ Bits breit

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. Nachteil eines Befehlsformates variabler Länge:
   Decodierung ist schwieriger. Dekodierschaltung komplizierter.

   Befehlsbearbeitung in einer Pipeline ist wesentlich schwieriger, da die Adresse des nächsten Befehls erst nach dem Holen und Dekodieren des aktuellen Befehls berechnet werden kann.

3. “ALU” steht für **A**rithmetic **L**ogic **U**nit oder *Arithmetisch Logische Einheit*

4. “RISC” steht für **R**educed **I**nstruction **S**et **C**omputer