



UNIVERSITÄT KARLSRUHE
Fakultät für Informatik
Industrielle Anwendungen der Informatik und Mikrosystemtechnik (IAIM)
Prof. Dr. R. Dillmann

Musterlösungen
zur Klausur „Technische Informatik I/II“
am 03. März 2004, 14.00 - 16.00 Uhr

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

Technische Informatik I	
Aufgabe 1	11 von 11 Punkten
Aufgabe 2	8 von 8 Punkten
Aufgabe 3	9 von 9 Punkten
Aufgabe 4	8 von 8 Punkten
Aufgabe 5	10 von 10 Punkten

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

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

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

Aufgabe 1

1. KNF von $f_1(c, b, a)$:

$$f_1(c, b, a) = (\bar{c} \vee b \vee \bar{a}) \cdot (\bar{c} \vee \bar{b} \vee a)$$

2.

	richtig	falsch
Alle Maxterme von f_1 sind Primimplikate	×	
Alle Primiplikanten von f_1 sind Kernprimimplikanten	×	
$\bar{c}a$ ist ein Primiplikant von f_1		×
$\bar{b}\bar{a}$ ist ein entbehrllicher Primimplikant von f_1		×

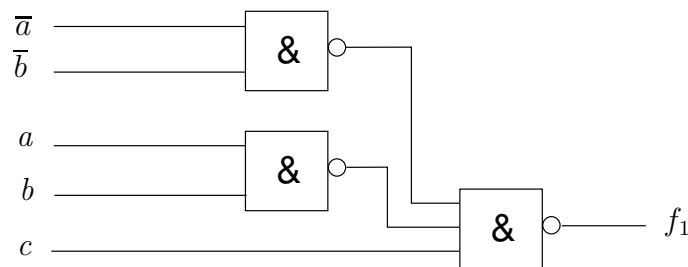
3. Primimplikanten mit Hilfe des Nelson-Verfahrens:

$$\begin{aligned} f_1(c, b, a) &= (\bar{c} \vee b \vee \bar{a}) \cdot (\bar{c} \vee \bar{b} \vee a) \\ &= \bar{c}\bar{c} \vee \bar{c}\bar{b} \vee \bar{c}a \vee b\bar{c} \vee b\bar{b} \vee ba \vee \bar{a}\bar{c} \vee \bar{a}\bar{b} \vee \bar{a}a \\ &= \bar{c} \vee \bar{a}\bar{b} \vee ba \end{aligned}$$

4. Schaltnetz:

DMF:

$$\begin{aligned} f_1(c, b, a) &= \bar{c} \vee \bar{a}\bar{b} \vee ba = \overline{\overline{\bar{c} \vee \bar{a}\bar{b} \vee ba}} \\ &= \overline{c \wedge (\bar{a} \wedge \bar{b}) \wedge (b \wedge a)} \end{aligned}$$



5. f_2 in minimierter Form:

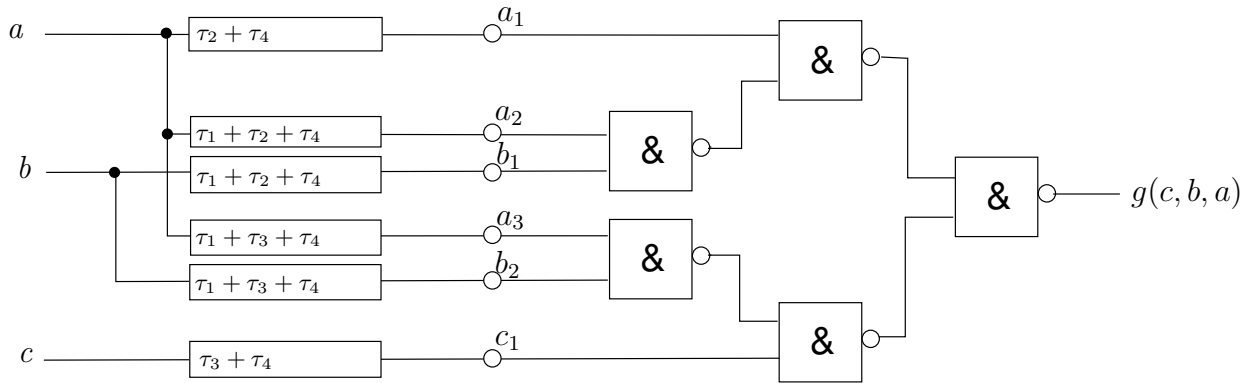
$$f_2(d, c, b, a) = \overline{\bar{b}\bar{a} \wedge b\bar{a} \wedge \bar{c}\bar{d}} = \bar{0} = 1$$

6. f_3 in konjunktiver Form:

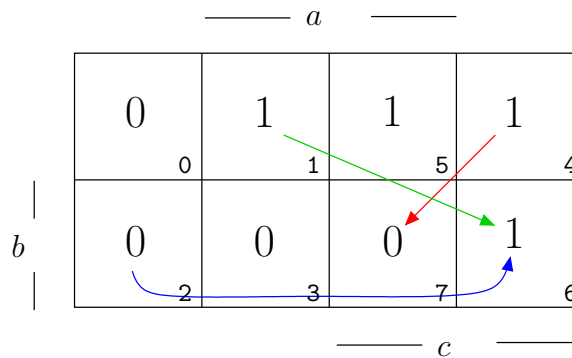
$$f_3(d, c, b, a) = \overline{ba \vee cd} = (\bar{b} \vee \bar{a}) \cdot (\bar{c} \vee \bar{d})$$

Aufgabe 2

1. Totzeitmodell:



2. KV-Diagramm für g :



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

Bei dem Übergang ändert nur eine Variable den Wert \Rightarrow frei von Funktionshasards, da die zugehörige Folge der Funktionswerte (hier $0 \rightarrow 1$) immer monoton ist.

• Übergang 2 $(c, b, a) : (0, 0, 1) \rightarrow (1, 1, 0)$

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. $B_1 \rightarrow B_3 \rightarrow B_7 \rightarrow B_6$ nicht monoton.

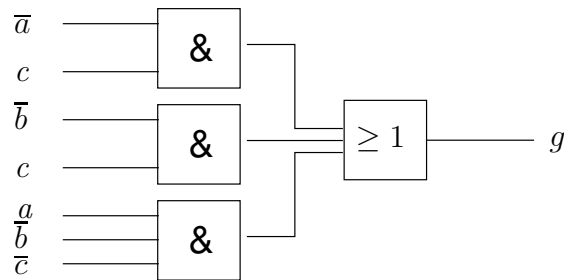
Deshalb ist der Übergang mit einem statischen 1-Funktionshasards behaftet.

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

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

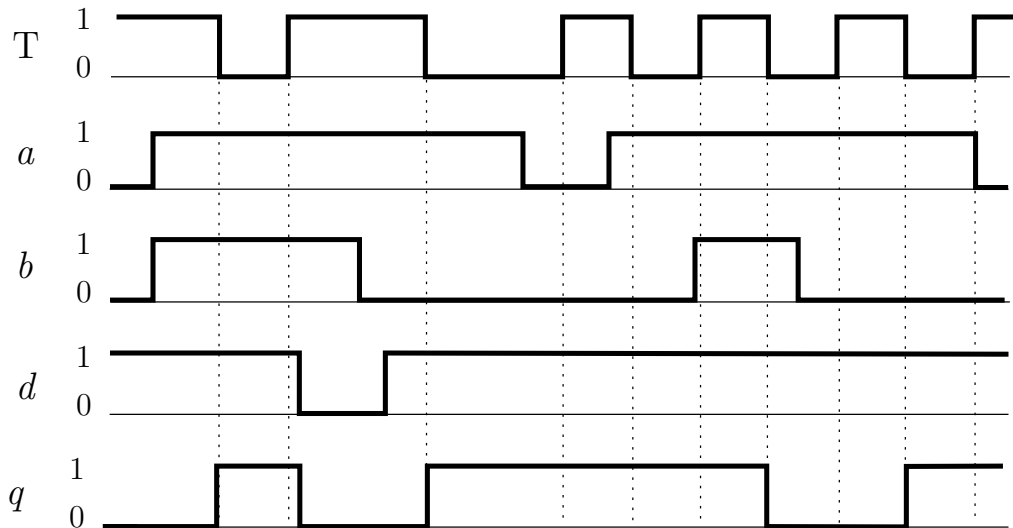
Schaltnetz:

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

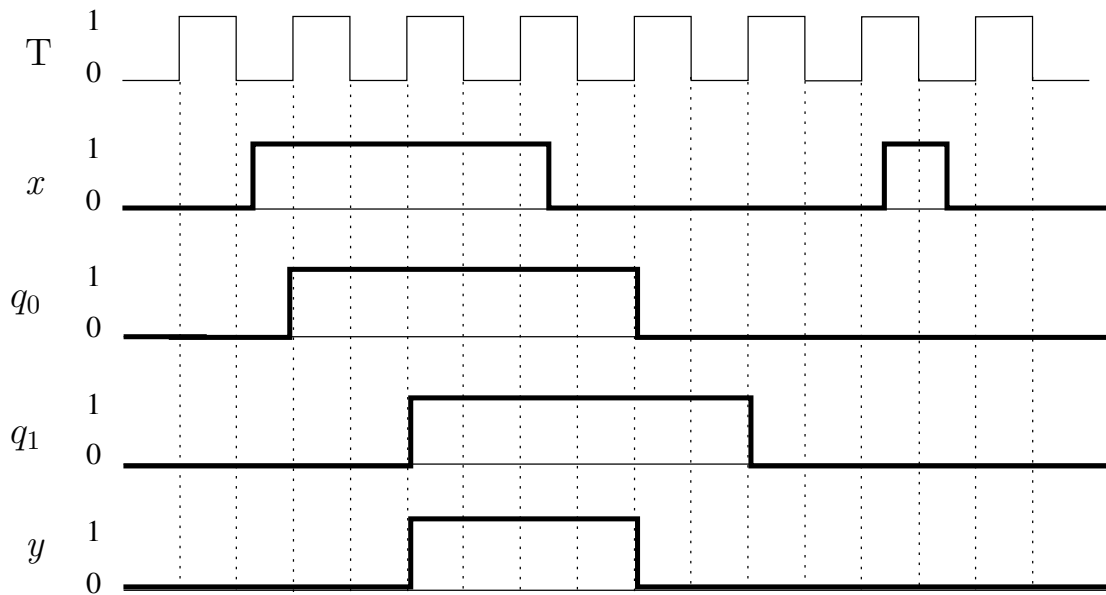


Aufgabe 3

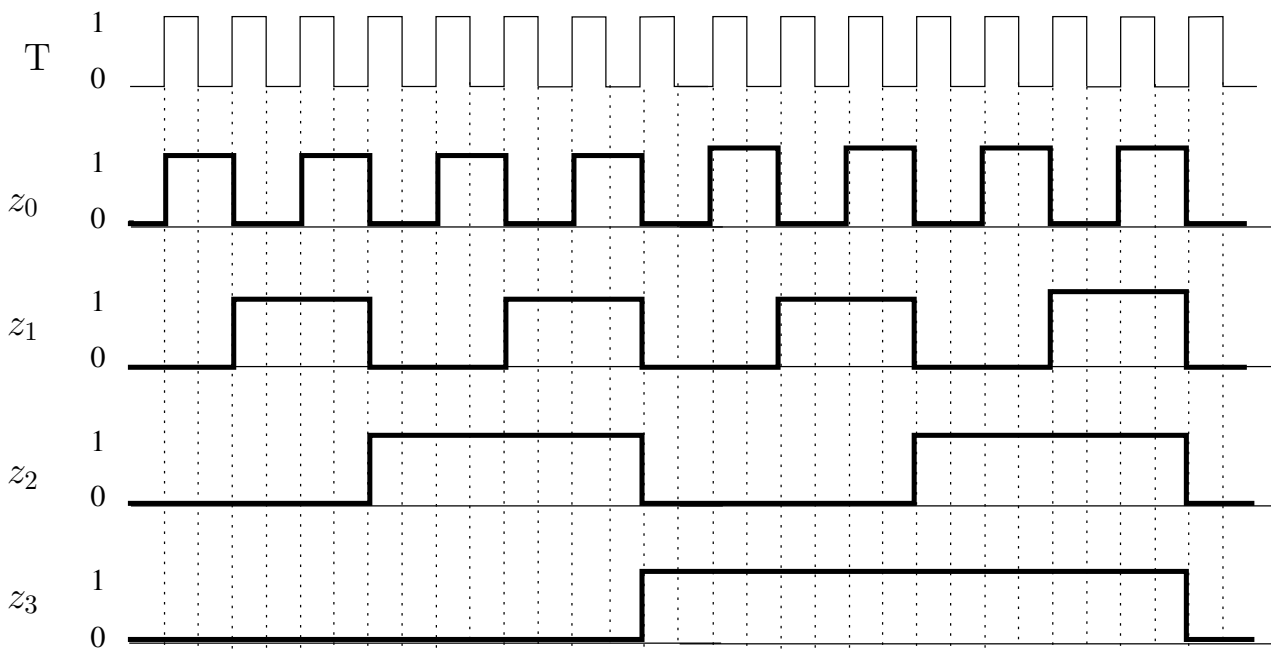
1.



2.

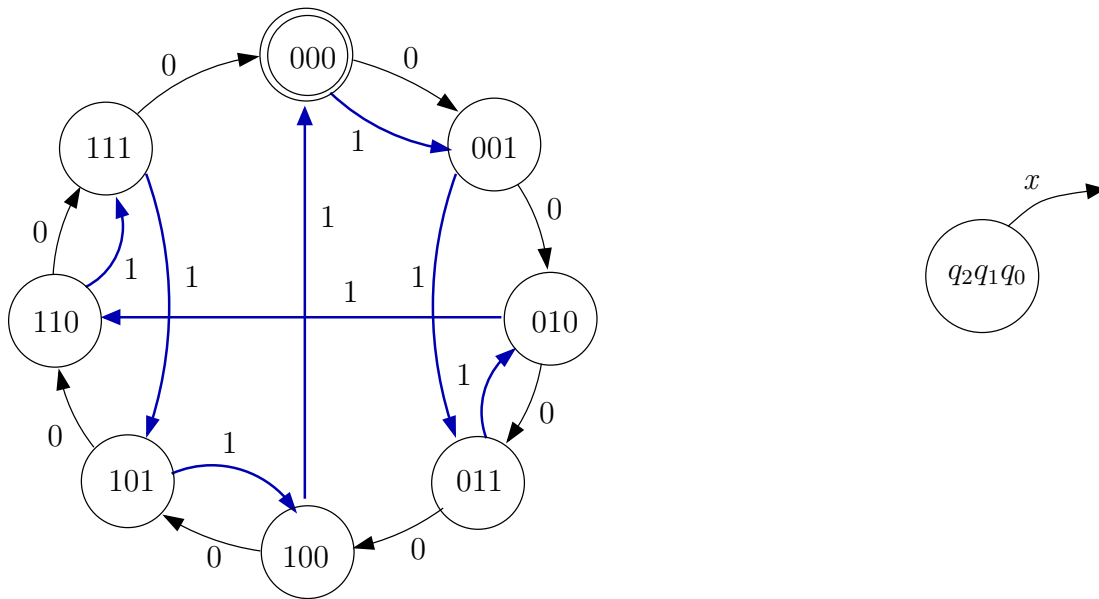


3.



Aufgabe 4

1. Automatengraph:



2. Kodierte Ablaufabelle:

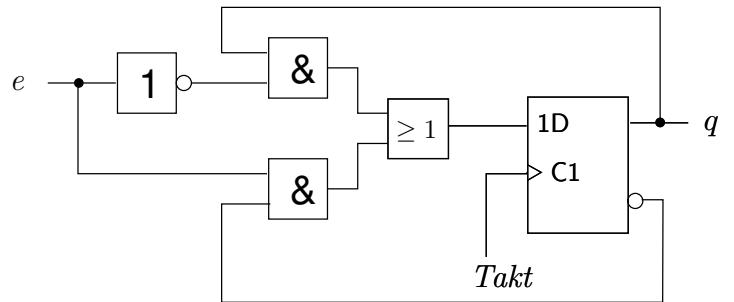
Eingabe x	Zustand			Folgezustand			Ansteuerfunktionen der Flipflops		
	q_2^t	q_1^t	q_0^t	q_2^{t+1}	q_1^{t+1}	q_0^{t+1}	e_2^t	e_1^t	e_0^t
0	0	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	0	0	1
0	0	1	1	1	0	0	1	1	1
0	1	0	0	1	0	1	0	0	1
0	1	0	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	0	1
0	1	1	1	0	0	0	1	1	1
1	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	1	0	1	0
1	0	1	0	0	1	1	0	0	1
1	0	1	1	0	1	0	0	0	1
1	1	0	0	0	0	0	1	0	0
1	1	0	1	1	0	0	0	0	1
1	1	1	0	1	1	1	0	0	1
1	1	1	1	1	0	1	0	1	0

3. T-Flipflops aus einem D-Flipflop:

e^t	q^t	q^{t+1}
0	0	0
0	1	1
1	0	1
1	1	0

$$q^{t+1} = e^t \bar{q}^t \vee \bar{e}^t q^t$$

$$q^{t+1} = d^t$$



Aufgabe 5

1. Dezimalzahl:

$$VZ = 1$$

$$Char = 1000\ 0100_2 = 132_{10} \Rightarrow Exp = (132 - 127)_{10} = 5_{10}$$

$$Mantisse = 1100 \dots 0$$

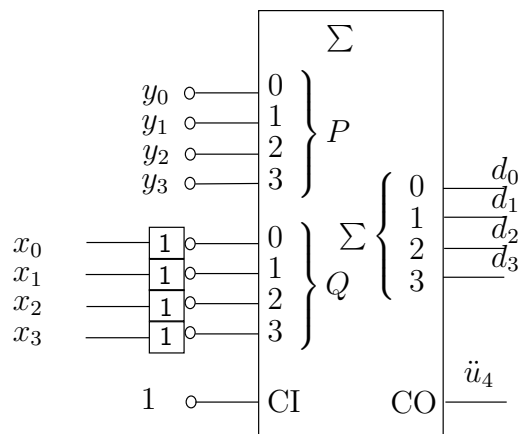
$$Dezimalzahl = (-1)^1 \cdot (1, Mantisse)_2 \cdot 2^{Exp}$$

$$= -1,11 \cdot 2^5 = -(1 + 0,5 + 0,25) \cdot 2^5 = -56_{10}$$

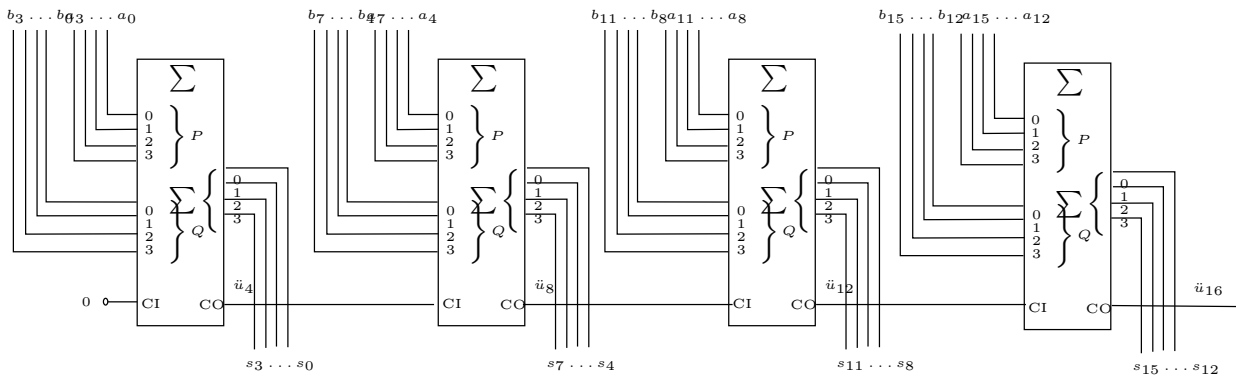
2.

Zahl	Vorzeichen-Betrag	Zweierkomplement
$+127_{10}$	0000 0000 0111 1111	0000 0000 0111 1111
-23_{10}	1000 0000 0001 0111	1111 1111 1110 1001

3. (a) Schaltung zur Subtraktion von 4-Bit Zweierkomplementzahlen:



(b) 16-Bit-Addierer aus 4-Bit-Carry-Lookahead-Addierern:



4. Datenwörter:

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:

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

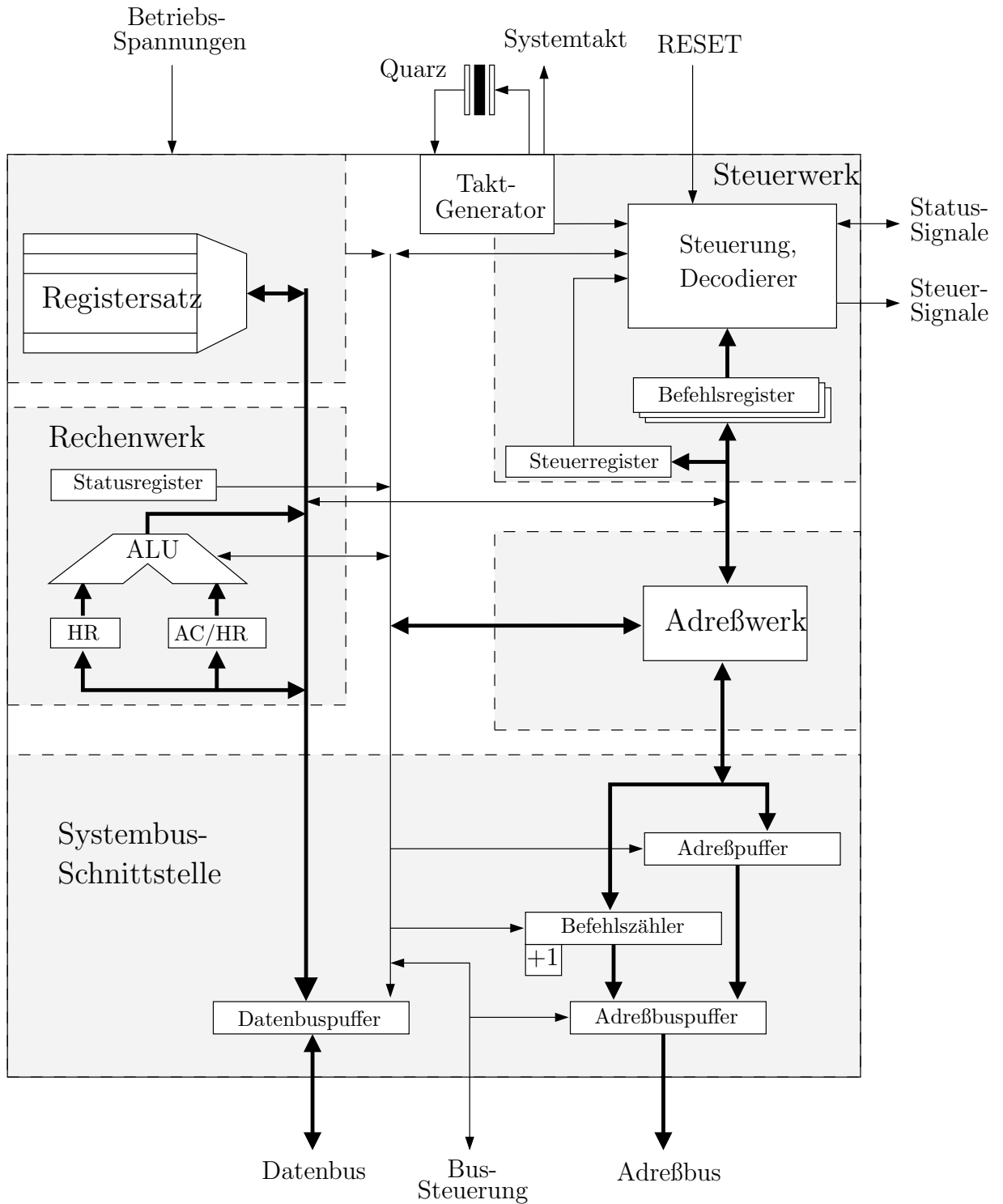
$$k_2 = k_2 \oplus m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7$$

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

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

- 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. Position 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 kein Ein-Bit-Fehler vor (Es liegt ein Ein-Bit-Fehler an der Position 14 vor oder es liegt ein Mehrbitfehler vor).
 - Kein Ein-Bit-Fehler \Rightarrow Datenwort 2: $1\ 0\ 0\ 1\ 0\ 0\ 0$
 - Mehrbitfehler \Rightarrow Datenwort 2 kann nicht ermittelt werden.

Aufgabe 6



Aufgabe 7

1. Kodierung des Mikroprogramms für die Lese-Phase:

Takt	Adresse	Befehl in hexadezimaler Schreibweise
1. Takt	0x00	2 1 0 8 8 0 1 (X = P _w = S = 1; R = 1)
2. Takt	0x01	1 4 0 0 8 0 2 (Y = E = 1; R = 1)
3. Takt	0x02	0 0 0 1 8 0 3 (C ₂ -C ₀ = 001; R = 1)
4. Takt	0x03	0 A 0 0 0 0 4 (Z = P _r = 1)
5. Takt	0x04	0 0 9 0 0 0 5 (I _r = D _w = 1)

2. Mikroprogramme:

LDV	STV
7. Takt: IR → SAR; R = 1	7. Takt: Akku → SDR
8. Takt: R = 1	8. Takt: IR → SAR; W = 1
9. Takt: R = 1	9. Takt: W = 1
10. Takt: SDR → Akku	10. Takt: W = 1
EQL	JMP
7. Takt: IR → SAR; R = 1	7. Takt: IR → IAR
8. Takt: Akku → X; R = 1	
9. Takt: R = 1	
10. Takt: SDR → Y	
11. Takt: ALU auf Vergleich	
12. Takt: Z → Akku	

Aufgabe 8

1. Aufgaben der einzelnen Pipeline-Stufen der DLX-Pipeline für

- arithmetisch-logische Befehle:

In der Instruction Fetch-Stufe wird der Befehl aus dem Speicher geladen. Die Instruction Decode-Stufe erzeugt die prozessorinternen Steuersignale und leitet die Inhalte der beiden Quellregister weiter. In der Execute-Stufe wird die Operation ausgeführt. Die Memory-Stufe wird bei arithmetisch-logischen Befehlen nicht benötigt und leitet deshalb die Daten weiter. In der Write-Back-Stufe wird das Ergebnis der Operation in das Zielregister geschrieben.

- Lade-/Speicher-Befehle:

In der Instruction Fetch-Stufe wird der Befehl aus dem Speicher geladen. Die Instruction Decode-Stufe erzeugt die prozessorinternen Steuersignale und leitet bei einem Ladebefehl den Registerinhalt mit der Adresse weiter. Bei einem Speicherbefehl wird das Register mit dem zu speichernden Wert ausgelesen. In der Execute-Stufe wird die Speicheradresse aus dem Registerinhalt und dem Displacement berechnet. Der eigentliche Speicherzugriff erfolgt in der Memory-Stufe. Bei einem Ladebefehl speichert die Write-Back-Stufe den geladenen Wert in das Zielregister ab.

2. (a) Echte Datendbhängigkeiten:

$S1 \rightarrow S2$ $S1 \rightarrow S3$ $S2 \rightarrow S3$ $S3 \rightarrow S7$ $S3 \rightarrow S8$
 $S4 \rightarrow S5$ $S4 \rightarrow S6$ $S5 \rightarrow S6$ $S6 \rightarrow S7$ $S6 \rightarrow S8$

(b)

```

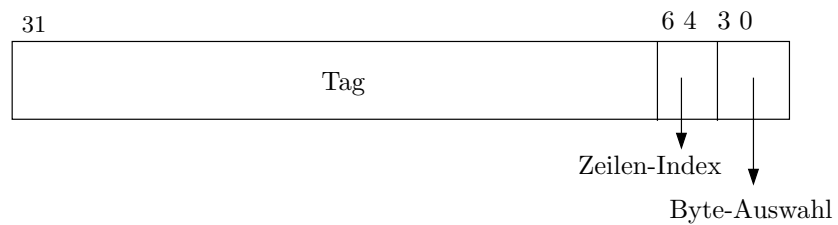
S1:  addi  $t1, $zero, 10
S4:  addi  $t4, $zero, 5
     NOP
S2:  sll   $t2, $t1, 4
S5:  sll   $t5, $t4, 4
     NOP
S3:  or    $t3, $t1, $t2
S6:  or    $t6, $t4, $t5
     NOP
     NOP
S7:  or    $t7, $t3, $t6
S8:  and   $t8, $t3, $t6
    
```

(c) Inhalte von \$t7 und \$t8 nach der Ausführung:

Register	Inhalt
\$t7	0x0000 00FF
\$t8	0x0000 0000

Aufgabe 9

1. Unterteilung der Hauptspeicheradresse:



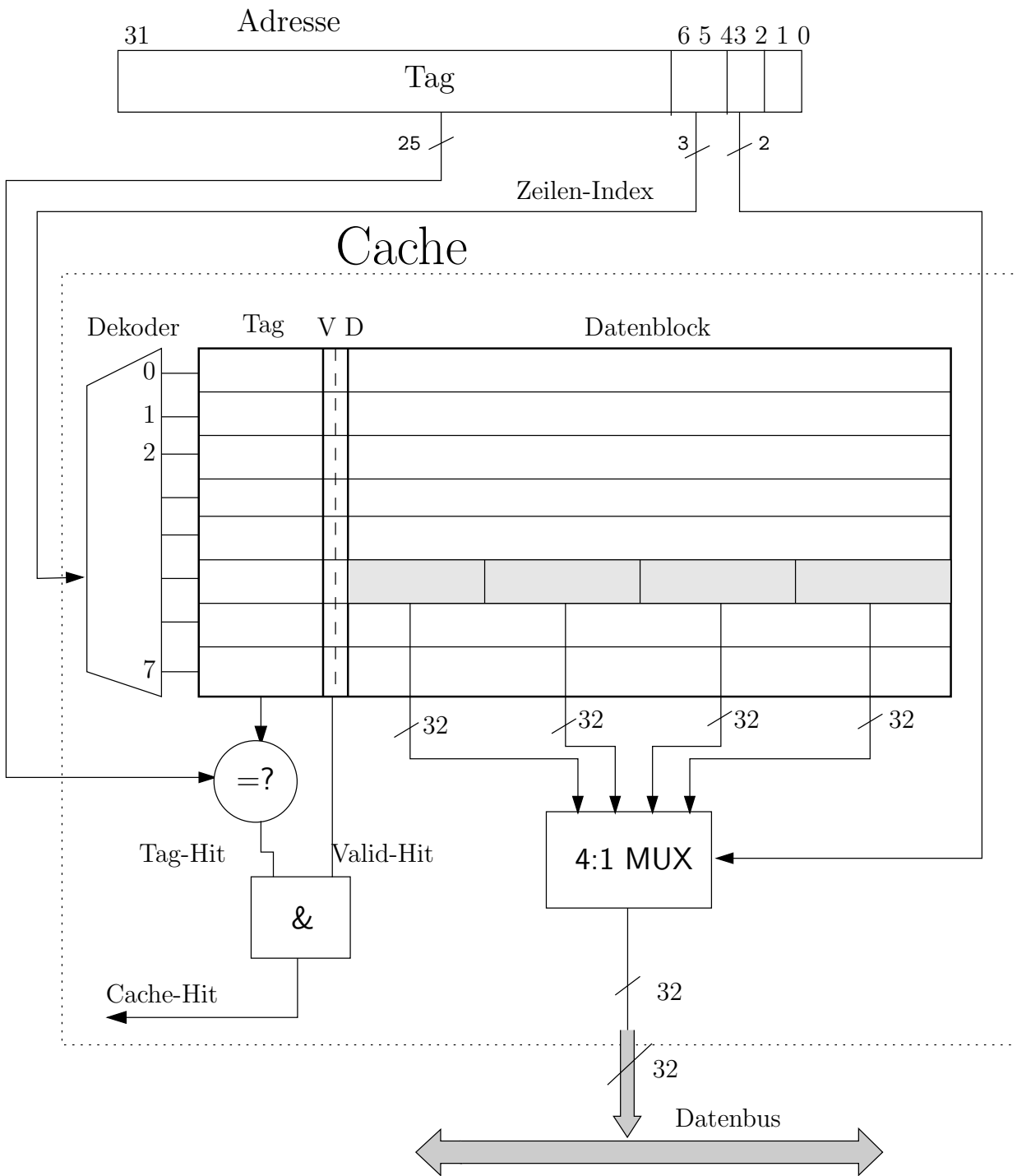
2. Speicherbedarf:

$$(25 + 2) \cdot 2^3 \text{Bit} + 128 \text{ Byte} = (27 + 128) \text{ Byte} = 155 \text{ Byte}$$

3.

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

4. Schematischer Aufbau des Cache-Speichers:



Aufgabe 10

1. Tristate-Treiber:

Gatterform, die nicht nur Hi und Lo weiterleiten kann, sondern auch einen dritten, gegen Spannungen beider Polaritäten, hochohmigen Zustand haben können. Dadurch kann z. B. ein Baustein vom Bus abgetrennt werden. Sie dienen zum „Abschalten“ des gleichzeitigen Zugriffs mehrerer Komponenten auf Systembusse.

2. Translation-Lookaside-Buffer (TLB):

Ein schneller vollassoziativer 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. Programmierbare Systemsteuerbausteine:

- DMA-Controller
- Cache-Controller
- Unterbrechungs-Steuerbausteine (Interrupt Controller), Zeitgeber-/Zähler-Bausteine, Speicherverwaltungsbausteine (MMU), Echtzeit-Uhren (Real-Time Clocks),

Nicht programmierbare Systemsteuerbausteine:

- Taktgeneratoren: Systemtakt und Synchronisationssignale
- Bus-Steuerbausteine (Bus-Controller)
- Steuerung des Systembuszugriffs (Bus Arbiter)
- Steuerbausteine für DRAM (Dynamic RAM Controller)

4. Speicherhierarchie:

Anordnung von Speicher verschiedener Technologie und damit auch verschiedener Eigenschaften nach absteigenden Zugriffsgeschwindigkeiten und aufsteigenden Speicherkapazitäten. In der Hierarchie nehmen die Zugriffszeiten und die Kapazität in den unteren Ebenen der Speicherhierarchie zu, während der Preis pro Bit sinkt.

Annahmen:

Zeitliche und örtliche Lokalität von Programmen.

5. (a) Speicherbausteine: Statische RAM-Bausteine, da schneller und außerdem kein „refresh“ notwendig.

(b) Anzahl der Speicherbausteine:

Die Kapazität eines $8K \times 4$ -bit-Baustein ist $32K$ Bit.

Anzahl der notwendigen Bausteine:

$$\frac{128K \text{ Byte}}{32K \text{ Bit}} = 2^5 = 32 \text{ Bausteine}$$