KIT-Fakultät für Informatik
Prof. Dr. Wolfgang Karl, Prof. Dr.-Ing. Uwe Hanebeck

Musterlösungen zur Klausur

Digitaltechnik und Entwurfsverfahren (TI-1)

und

Rechnerorganisation (TI-2)

am 09. August 2019, 12:30 – 14:30 Uhr

Name:	Vorname:	Matrikelnummer:
Bond	James	007

Digitaltechnik und Entwurfsverfahren (TI-1)	
Aufgabe 1	10 von 10 Punkten
Aufgabe 2	5 von 5 Punkten
Aufgabe 3	6 von 6 Punkten
Aufgabe 4	12 von 12 Punkten
Aufgabe 5	12 von 12 Punkten

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

Gesamtpunktzahl:	90 von 90 Punkten

	Note: **1,0**

Aufgabe 1 *Schaltfunktionen* (10 Punkte)

1. KNF von $f(w,x,y,z)$: (KV-Diagramm oder algebraisch) 2 P.

$$\begin{aligned} f(w,x,y,z) &= \text{MAXt}(0,2,6,7,14,15) \\ &= (w \vee x \vee y \vee z) \cdot (w \vee x \vee \overline{y} \vee z) \cdot (w \vee \overline{x} \vee \overline{y} \vee z) \cdot \\ &\quad (w \vee \overline{x} \vee \overline{y} \vee \overline{z}) \cdot (\overline{w} \vee \overline{x} \vee \overline{y} \vee z) \cdot (\overline{w} \vee \overline{x} \vee \overline{y} \vee \overline{z}) \end{aligned}$$

2. DMF von $g(c,b,a)$: 3 P.

$$\begin{aligned} g(c,b,a) &= \Big(a \mathbin{\overline{\wedge}} (a \mathbin{\overline{\wedge}} b)\Big) \mathbin{\overline{\wedge}} \Big(c \mathbin{\overline{\wedge}} (a \mathbin{\overline{\wedge}} b)\Big) \\ &= a\,(a \mathbin{\overline{\wedge}} b) \;\vee\; c\,(a \mathbin{\overline{\wedge}} b) \;=\; a\,(\overline{a} \;\vee\; \overline{b}) \;\vee\; c\,(\overline{a} \;\vee\; \overline{b}) \\ &= a\,\overline{b} \;\vee \overline{a}\,c \vee \overline{b}\,c \\ &= a\,\overline{b} \;\vee \overline{a}\,c \qquad \text{(Consensus Regel)} \end{aligned}$$

3. Kern-Primimplikate: C 1 P.

   Reduzierte Tabelle: (Gestrichene Spalten: a, d)

	b	c	e
A	$\times$	$\times$	
B			$\times$
D	$\times$		$\times$
E	$\times$	$\times$	$\times$

4. Dominierte Maxterme: c 1 P.

   Reduzierte Tabelle: (Gestrichene Spalte: b)

	c	e	Kosten
A	$\times$		$\frac{1}{2}$
B		$\times$	$\frac{1}{2}$
D		$\times$	$\frac{1}{3}$
E	$\times$	$\times$	$\frac{1}{3}$

5. Dominierende Primimplikate: E 2 P.

   Reduzierte Tabelle: (Gestrichene Zeilen: A, B und D)

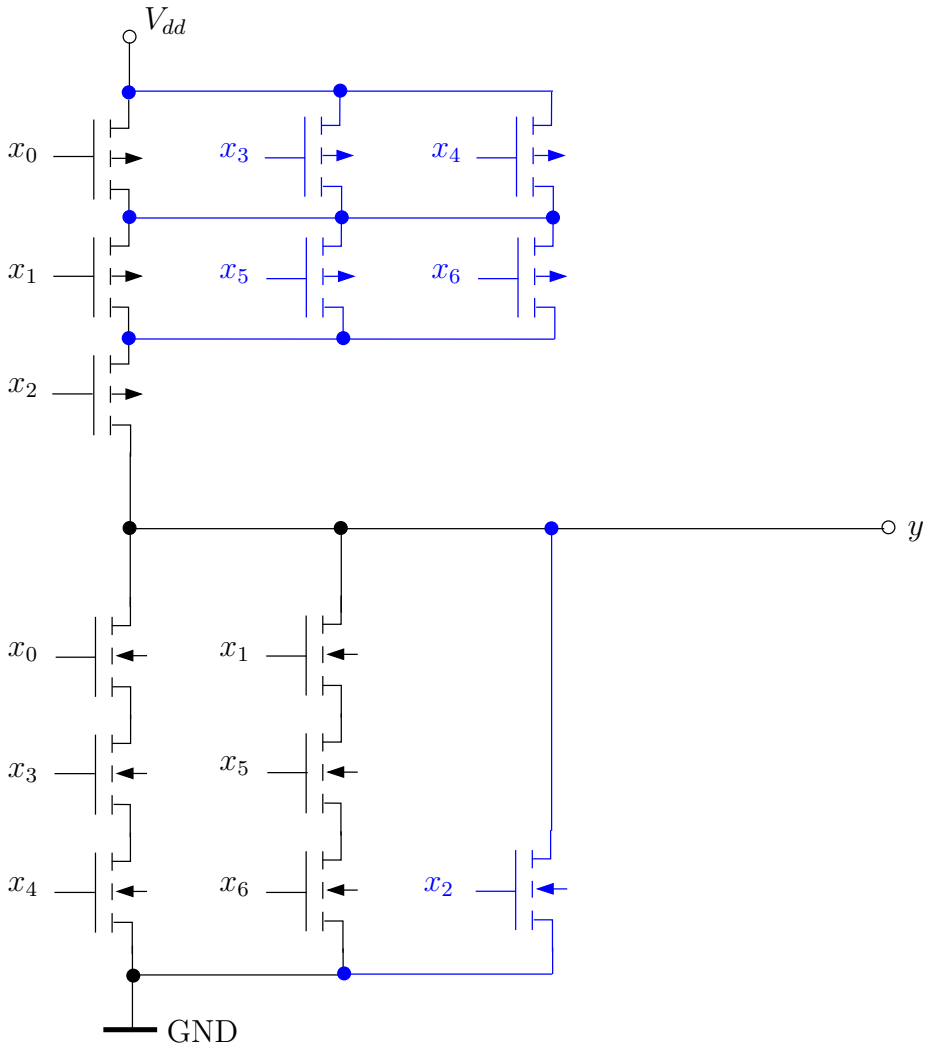
	c	e
E	$\times$	$\times$

6. Minimalform der Funktion z: $z = C \cdot E$ 1 P.

Aufgabe 2 *CMOS-Technologie* (5 Punkte)

4 P.

1.



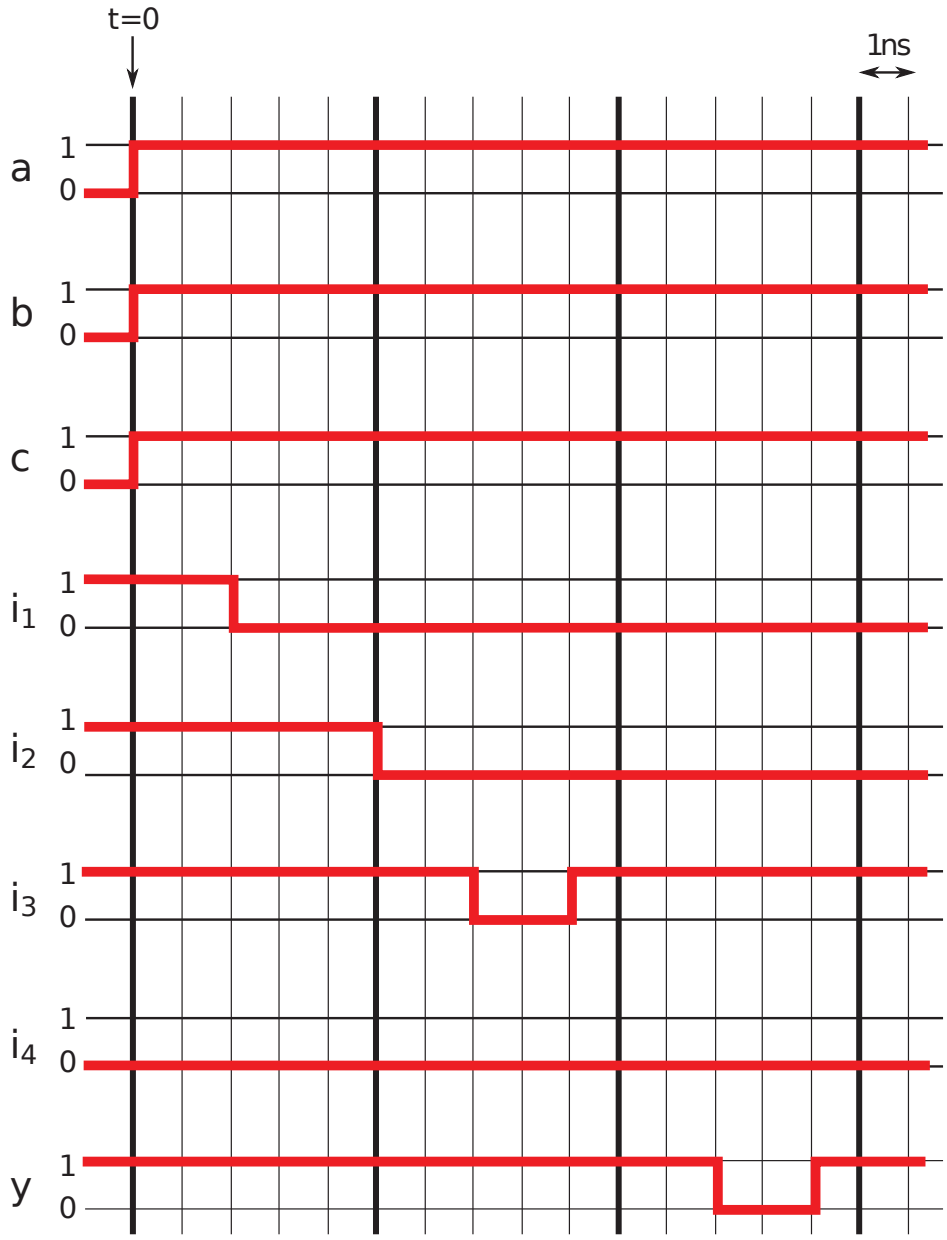
2. Realisierte Schaltfunktion:

1 P.

$$\begin{aligned} y &= \overline{(x_0\ x_3\ x_4)\ \vee\ (x_1\ x_5\ x_6)\ \vee\ x_2} \\ &= (\overline{x}_0\ \vee\ \overline{x}_3\ \vee\ \overline{x}_4)\ (\overline{x}_1\ \vee\ \overline{x}_5\ \vee\ \overline{x}_6)\ \overline{x}_2 \end{aligned}$$

Aufgabe 3 *Laufzeiteffekte* (6 Punkte)

1. Zeitdiagramm: 3 P.



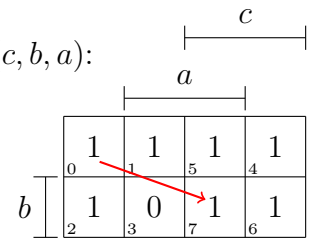
2. Hasardfehler (falls ja, Analyse): 3 P.

   Ja, es tritt ein Hasardfehler auf, denn y ist zu Beginn und Ende des Übergangs 1, wechselt während des Übergangs jedoch kurzzeitig auf 0.

   Betrachtet man nun die möglichen Folgen von Funktionswerten beim Übergang $(0,0,0) \rightarrow (1,1,1)$, stellt man fest, dass die Folge von Funktionswerten bei bestimmten Eingabewechseln (z.B. a, b, c) nicht monoton ist (siehe KV-Diagramm auf nächster Seite). Somit handelt es sich um einen Funktionshasard.

   Insgesamt ist der Hasard also als statischer 1-Funktionshasard zu klassifizieren.

$y(c, b, a)$:

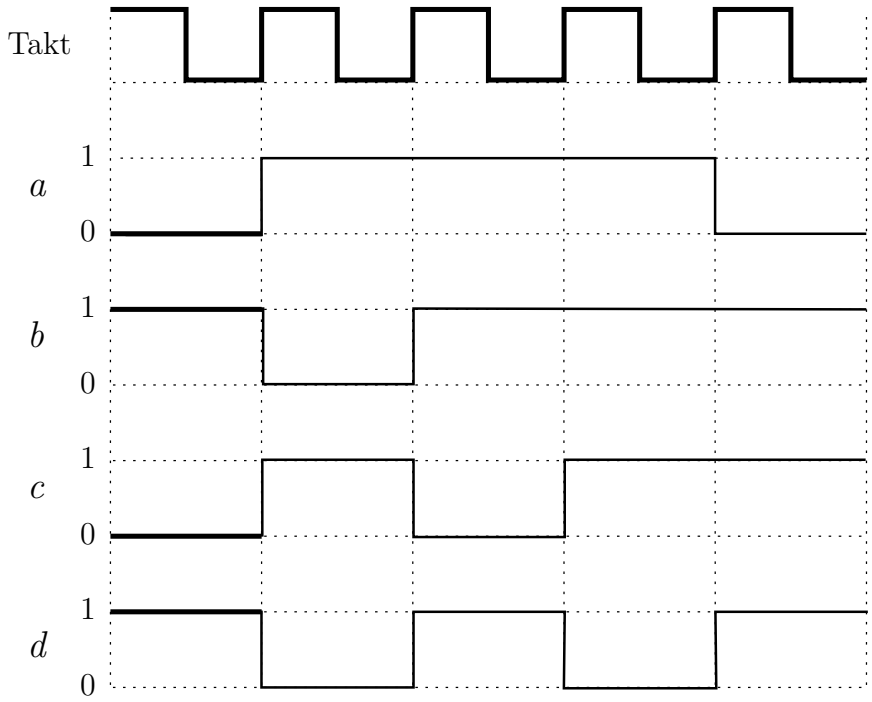


Aufgabe 4 *Schaltwerke* (12 Punkte)

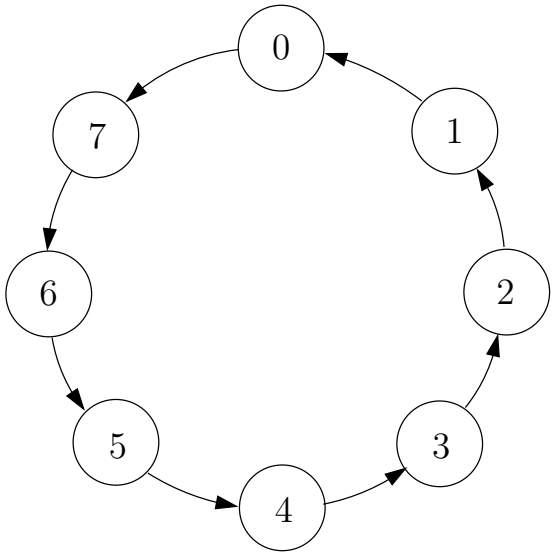
1. (a) Das Schaltwerk ist *synchron* 1 P.

   (b) Maximale Anzahl der Zustände ist: $2^4 = 16$ Zustände 1 P.

   (c) Verläufe der Signale a, b, c und d: 4 P.



2. (a) Automatengraph: 2 P.



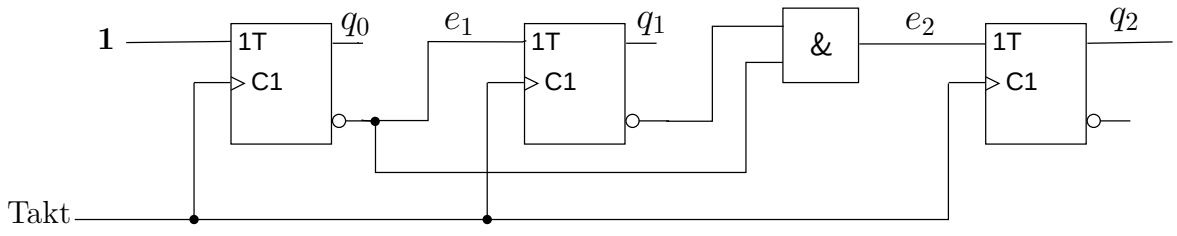
(b) Kodierte Ablauftabelle: 1 P.

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	1	1	1	1	1	1
0	0	1	0	0	0	0	0	1
0	1	0	0	0	1	0	1	1
0	1	1	0	1	0	0	0	1
1	0	0	0	1	1	1	1	1
1	0	1	1	0	0	0	0	1
1	1	0	1	0	1	0	1	1
1	1	1	1	1	0	0	0	1

(c) Minimalformen der Ansteuerfunktionen der Flipflops: Aus der Ablauftabelle ablesbar. 1 P.

$$
\begin{aligned}
e_2^t &= \overline{q}_2^t\,\overline{q}_1^t\,\overline{q}_0^t \;\vee\; q_2^t\,\overline{q}_1^t\,\overline{q}_0^t \;=\; \overline{q}_1^t\,\overline{q}_0^t \\
e_1^t &= \overline{q}_0^t \\
e_0^t &= 1
\end{aligned}
$$

(d) Schaltbild des Zählers: 2 P.



Aufgabe 5 *Rechnerarithmetik* (12 Punkte)

1. Die Basen s und r: 1 P.

$$1 \cdot r^1 + 2 \;=\; 1 \cdot s^2 + 1 \cdot s^1 + 1 \;\rightarrow\; r = s^2 + s - 1$$

Es existieren unendlich viele Lösungen:

s	2	3	4	...
r	5	11	19	...

2 P.

2. (a) 12 binären Stellen: $2^{12} - 1 \;= 4095$
   (b) 4 hexadezimalen Stellen: $16^4 - 1 = 65535$ 3 P.

3. $2005_{10} \;=\; 111\ 1101\ 0101_2$

   - 32-Bit Zweierkomplement-Format:

$$0000\ 0000\ 0000\ 0000\ \ 0000\ 0111\ 1101\ 0101$$

   - 32-Bit IEEE-754-Gleitkomma-Format:

$$111\ 1101\ 0101_2 \;=\; 1,11\ 1101\ 0101 \cdot 2^{10}$$

$Exp = 10 \Rightarrow Char \;=\; Exp + 127 = 137_{10} = 1000\ 1001_2$

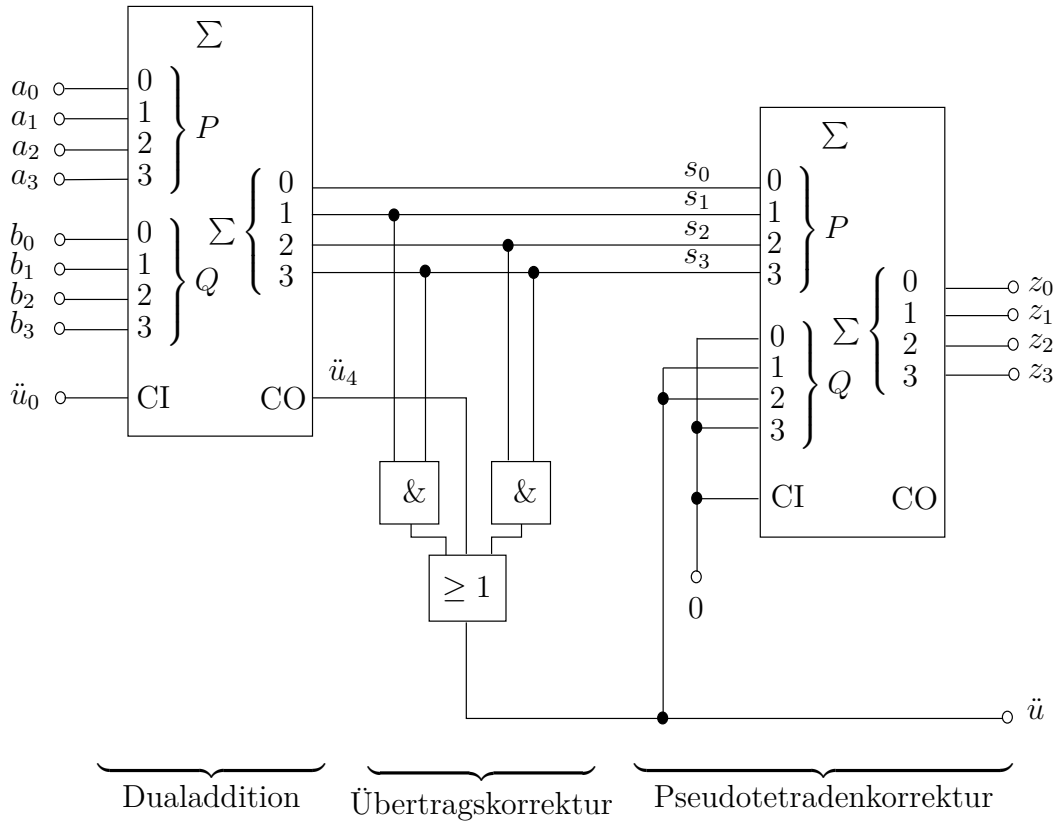
31	30 23	22 0
0	1000 1001	1111 0101 0100 0000 ... 000

4. BCD-Addierer für eine Tetrade: 3 P.

   Eine Korrektur ist durch die Addition von 0110 zu $(s_3 s_2 s_1 s_0)$ notwendig, wenn

   - ein Übetrag bei der Addition der beiden Tetraden $(a_3 a_2 a_1 a_0)$ und $(b_3 b_2 b_1 b_0)$ auftritt, d. h. $\ddot{u}_4 \;=\; 1$ oder
   - das Ergebniss $(s_3 s_2 s_1 s_0)$ eine Pseudotetrade ist, d. h. $s_3 = s_2 = 1$ oder $s_3 = s_1 = 1$

   Damit ergibt sich: $\ddot{u} \;=\; \ddot{u}_4 \;\vee\; s_1\, s_3 \;\vee\; s_2\, s_3$



5. Datenwörter: (Man beachte: $k_i = QS_{i-1}$) 3 P.

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	0	1	0	0	1	0	1	0	0	1	0
Codewort 2:	1	1	0	0	0	1	0	0	0	0	0	1

Die Prüfbits lassen sich nach den folgenden Regeln berechnen:

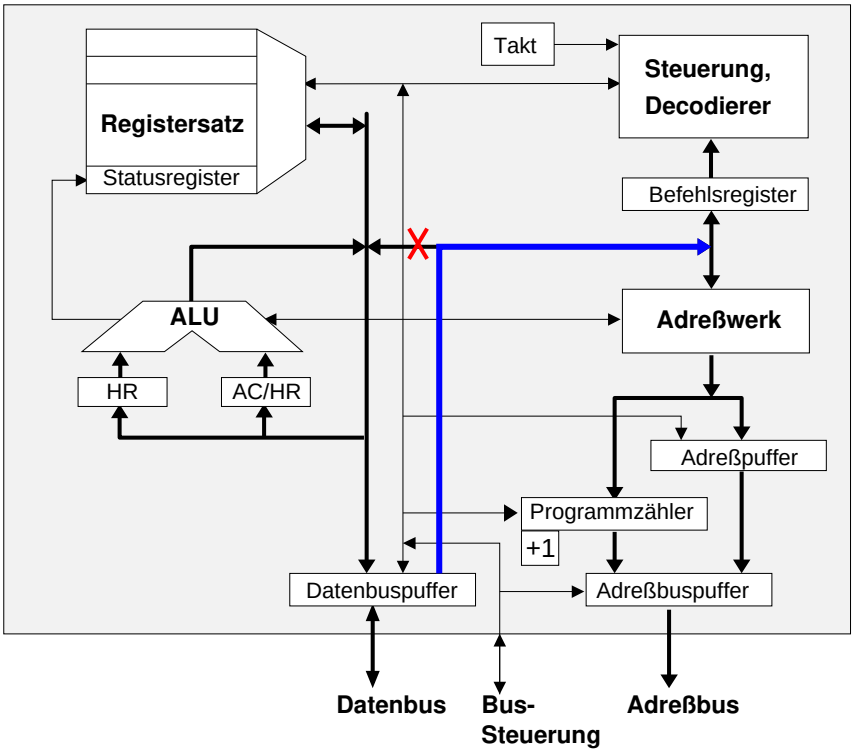
$$\begin{aligned} k_1 &= k_1 \oplus m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 \\ k_2 &= k_2 \oplus m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 \\ k_3 &= k_3 \oplus m_2 \oplus m_3 \oplus m_4 \oplus m_8 \\ k_4 &= k_4 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_8 \end{aligned}$$

- Codewort 1: **1 0 1 0 0 1 0 1 0 0 1 0** $\Rightarrow$ $k_4\ k_3\ k_2\ k_1 = 0\ 1\ 1\ 0$ $\Rightarrow$ Es liegt ein Fehler an der 6. Postion vor $\Rightarrow$ Datenwort 1: **1 0 1 0 1 1 1 0**

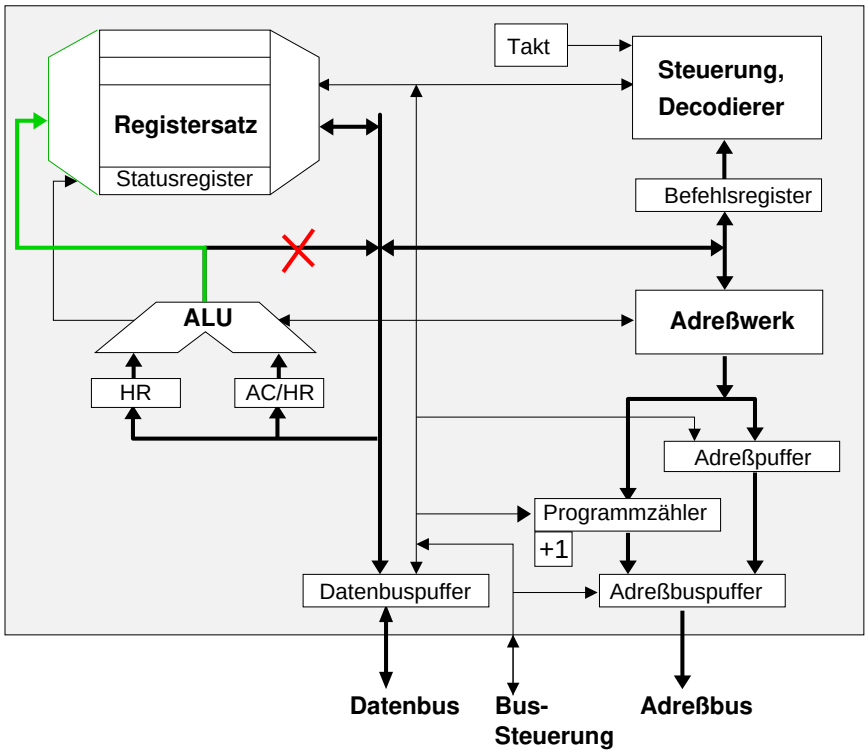
- Codewort 2: **1 1 0 0 0 1 0 0 0 0 0 1** $\Rightarrow$ $k_4\ k_3\ k_2\ k_1 = 0\ 0\ 0\ 1$ $\Rightarrow$ Es liegt ein Fehler an der 1. Postion vor $\Rightarrow$ Datenwort 2: **1 1 0 0 1 0 0 0**

Aufgabe 6 *Mikroprozessor* (6 Punkte)

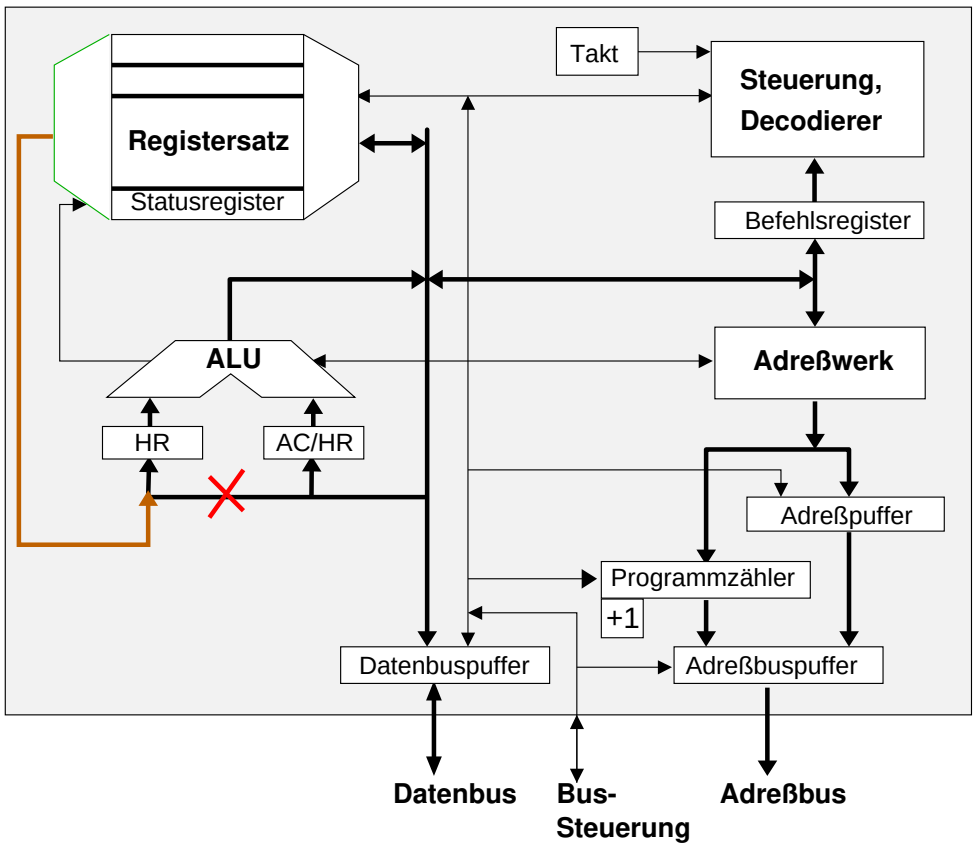
1. 2 P.



2. 2 P.



3. 2 P.


Takt
Steuerung,
Decodierer
Registersatz
Statusregister
Befehlsregister
ALU
Adreßwerk
HR
AC/HR
Adreßpuffer
Programmzähler
+1
Datenbuspuffer
Adreßbuspuffer
Datenbus
Bus-
Steuerung
Adreßbus

Aufgabe 7 *MIPS-Assembler* (12 Punkte)

1. (a) `do-while`-Schleife in MIPS: 3 P.

```
Loop:    add $s3, $s3, $s4
add $t1, $s3, $s3
add $t1, $t1, $t1
add $t1, $t1, $s6
lw  $t0, 0($t1)
beq $t0, $s5, Loop
```

   (b) `if-else`-Anweisungen in MIPS: 5 P.

```
bne $s3, $s4, Else1
add $t1, $s5, $s5
add $t1, $t1, $t1
add $t1, $t1, $s6
lw  $t0, 0($t1)
add $s3, $s3, $t0
j   fertig
Else1:   bne $s3, $s5, Else2
add $t1, $s4, $s4
add $t1, $t1, $t1
add $t1, $t1, $s6
lw  $t0, 0($t1)
add $s3, $s3, $t0
j   fertig
Else2:   add $s3, $s4, $s5
fertig:
```

2. Unterschied Maschinensprache zu Assemblersprache: 1 P.

   Maschinensprache ist ein 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.

3. Assemblerdirektive `.align 2`: 1 P.
   `.align 2` bewirkt, dass die folgenden Daten an der nächstmöglichen Adresse gespeichert werden, die durch $2^2 = 4$ teilbar ist.

4. MIPS-Befehlssatz: 2 P.

   Nein.

   Der MIPS-Befehlssatz besteht aus mehr als 64 Befehlen. Durch den OpCode (Bits: 31...26) sind Gruppen von Befehlen definiert (z.B. Gleitkomma-Operationen). Innerhalb dieser Gruppen werden zusätzliche Bitfelder im Befehlsformat verwendet, um zwischen verschiedenen Befehlen (Addition, Subtraktion, Multiplikation, ... usw.) der Gruppe zu unterscheiden.

Aufgabe 8 *Pipelining* (9 Punkte)

1. Datenabhängigkeiten: 3 P.

   - Echte Abhängigkeiten:

   $$S_1 \rightarrow S_2 \ (\texttt{\$t1}) \qquad S_1 \rightarrow S_3 \ (\texttt{\$t1})$$
   $$S_2 \rightarrow S_3 \ (\texttt{\$t2}) \qquad S_2 \rightarrow S_5 \ (\texttt{\$t2})$$
   $$S_3 \rightarrow S_4 \ (\texttt{\$t3}) \qquad S_3 \rightarrow S_5 \ (\texttt{\$t3})$$

   - Gegen-Abhängigkeiten:

   $$S_2 \rightarrow S_4 \ (\texttt{\$t1}) \qquad S_3 \rightarrow S_4 \ (\texttt{\$t1})$$

   - Ausgabe- Abhängigkeiten:

   $$S_1 \rightarrow S_4 \ (\texttt{\$t1})$$

2. Beseitigung der Daten- und Steuerflusskonflikte: 4 P.

```
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. Ausgabeabhängigkeiten (*Output dependence*) und Gegenabhängigkeiten (*Anti-dependence*) können in der DLX-Pipeline nicht zu Konflikten führen, da 2 P.

   - das Lesen aus Registern immer in der Stufe 2 (ID/RF) und
   - das Schreiben in Register immer in der Stufe 5 (WB) erfolgt.

Aufgabe 9 *Cache-Speicher* (10 Punkte)

1. (a) Größe eines Cache-Blocks in Byte: 1 P.

   5 Bits Byte-Offset → Blockgröße: $2^5 = 32$ Byte

   (b) Kapazität des Cache-Speichers: 1 P.

   11 Bits Index → 2^{11} Sätze

   A2-Cache $2 \times 2^{11} = 2^{12}$ Cache-Blöcke mit je 32 Bytes

   Cache-Kapzität: $2^{12} \times 32 = 2^{12} \times 2^5$ Byte $= 2^{17}$ Byte $= 128$ KByte

   (c) Der insgesamt erforderliche Speicherbedarf: 2 P.

   Kapazität + (Tag-Länge + 2 Statusbits) × (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`: 2 P.

   A2-Cache → Es wird ein Vergleich mit 2 Zeilen im Cache durchgeführt.
   Satz-Index = 0001 1010 $001_2 = 209_{10}$
   → Der Vergleich wird mit den Zeilen 418 und 419 durchgeführt.

   4 P.

2.

Adresse	**0x44**	**0xA0**	**0xC3**	**0x9E**	**0x66**	**0x2D**	**0x6B**	**0x49**
Index	4	2	4	1	6	2	6	4
Tag	0	1	1	1	0	0	0	0
read/write	w	r	w	r	r	w	r	w
Hit/Miss	×	—	—	×	×	—	×	—
write back?	nein	nein	ja	nein	nein	nein	nein	ja

Aufgabe 10 *Virtuelle Speicherverwaltung* (8 Punkte)

1. Unterteilung der virtuellen Adresse: 1 P.

2. Physikalische Adressen: 4 P.

Virtuelle		Physikalische	
Adresse	Seitennummer	Seitennummer	Adresse
1023	0	3	$3 * 1024 + 1023 = 4095$
1024	1	1	1 * 1024 + 0 = 1024
4204	4	2	2 * 1024 + 108 = 2156
6200	6	0	0 * 1024 + 56 = 56

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. Alternatives Stichwort: Lokalitätsprinzip 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