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 31. Juli 2017, 14:00 – 16:00 Uhr

Name:	Vorname:	Matrikelnummer:
Bond	James	007

Digitaltechnik und Entwurfsverfahren/TI-1	
Aufgabe 1	9 von 9 Punkten
Aufgabe 2	10 von 10 Punkten
Aufgabe 3	6 von 6 Punkten
Aufgabe 4	8 von 8 Punkten
Aufgabe 5	12 von 12 Punkten

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

Gesamtpunktzahl:	90 von 90 Punkten

	Note: **1,0**

Aufgabe 1

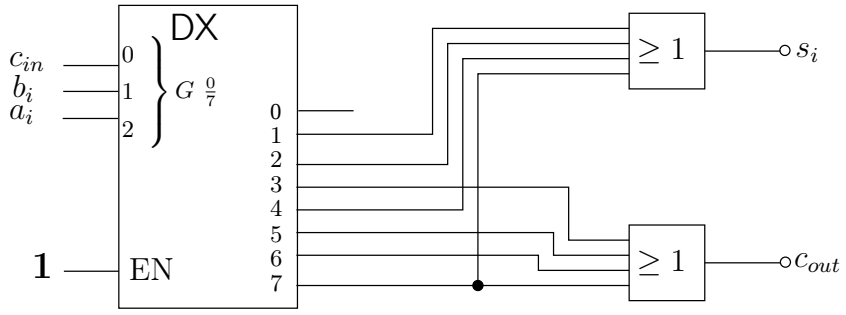
1. DNF von $s_i(a_i, b_i, c_{in})$:

$$s_i \;=\; \overline{a}_i\,\overline{b}_i\,c_{in} \;\vee\; \overline{a}_i\,b_i\,\overline{c}_{in} \;\vee\; a_i\,\overline{b}_i\,\overline{c}_{in} \;\vee\; a_i\,b_i\,c_{in} \;=\; \mathrm{MINt}(1,2,4,7)$$

2. KNF von $c_{out}(a_i, b_i, c_{in})$:

$$\begin{aligned} c_{out} &= (a_i \vee b_i \vee c_{in}) \cdot (a_i \vee b_i \vee \overline{c}_{in}) \cdot (a_i \vee \overline{b}_i \vee c_{in}) \cdot (\overline{a}_i \vee b_i \vee c_{in}) \\ &= \mathrm{MAXt}(0,1,2,4) \end{aligned}$$

3. Schaltnetz:



$$s_i \;=\; \mathrm{MINt}(1,2,4,7) \qquad\qquad c_{out} \;=\; \mathrm{MINt}(3,5,6,7)$$

4.

s_i:

b_i \ c_{in}	0	1	0	1
1	1	0	1	0

a_i

c_{out}:

b_i \ c_{in}	0	0	1	0
0	0	1	1	1

a_i

Konjunktive Minimalform von c_{out}:

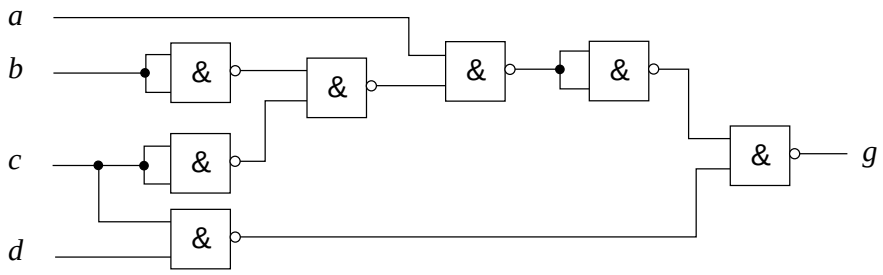
$$c_{out} \;=\; (a_i \vee c_{in}) \cdot (a_i \vee b_i) \cdot (b_i \vee c_{in})$$

Aufgabe 2

1. Schaltfunktion:
$$y \;=\; \Big((\overline{c} \leftrightarrow b)(\overline{b} \vee a)\Big) \vee \Big((\overline{c}\;\overline{a})(b \vee (a \nleftrightarrow b)\Big)$$

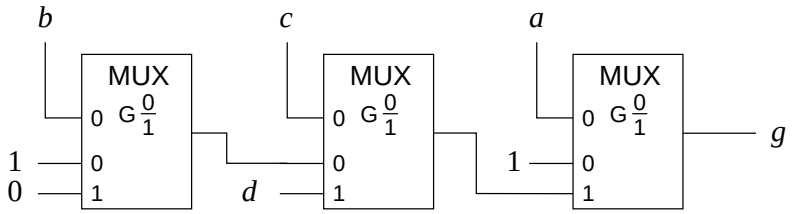
2. Zweistufige disjunktive Form von y
$$\begin{aligned}
y &= \Big((\overline{c} \leftrightarrow b)(\overline{b} \vee a)\Big) \vee \Big((\overline{c}\;\overline{a})(b \vee (a \nleftrightarrow b))\Big) \\
&= \Big((\overline{c}\;b \vee c\;\overline{b})(\overline{b} \vee a)\Big) \vee \Big((\overline{c}\;\overline{a})(b \vee (b\;\overline{a} \vee \overline{b}\;a))\Big) \\
&= \underbrace{\overline{c}\;b\;\overline{b}}_{=0} \vee \overline{c}\;b\;a \vee \underbrace{c\;\overline{b}\;\overline{b}}_{=c\;\overline{b}} \vee c\;\overline{b}\;a \vee \overline{c}\;\overline{a}\;(\underbrace{b \vee b\;\overline{a}}_{=b} \vee \overline{b}\;a) \\
&= \overline{c}\;b\;a \vee c\;\overline{b} \vee c\;\overline{b}\;a \;\vee \overline{c}\;\overline{a}\;b \;\vee\; \overline{c}\;\overline{a}\;\overline{b}\;a \\
&= \overline{c}\;b\;a \vee c\;\overline{b} \vee c\;\overline{b}\;a \;\vee \overline{c}\;b\;\overline{a} \\
&= c\;\overline{b} \;\vee \overline{c}\;b\;a \vee \overline{c}\;b\;\overline{a} \;=\; c\;\overline{b} \vee \overline{c}\;b\;(a \vee \overline{a}) \\
&= c\;\overline{b} \;\vee \overline{c}\;b
\end{aligned}$$

3. NAND-Schaltnetz:
$$\begin{aligned}
g(d,c,b,a) &= \overline{\overline{\overline{(\overline{a} \;\vee \overline{c}\;\overline{b}\,)} \;\vee c\;d}} \;=\; \overline{(a \;\wedge\; \overline{(\overline{c} \wedge \overline{b}\,)) \wedge \overline{(c \wedge d)}}} \\
&= \overline{\overline{\overline{(a \;\wedge\; \overline{(\overline{c} \wedge \overline{b}\,)})}} \wedge \overline{(c \wedge d)}}
\end{aligned}$$



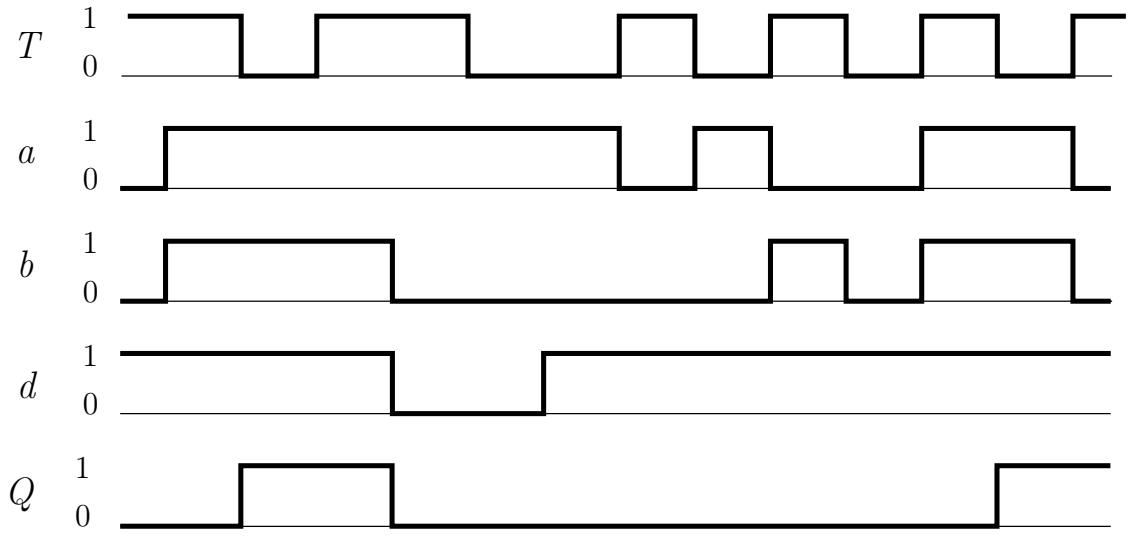
4. Multiplexer-Schaltnetz:

$$\begin{aligned} g(d,c,b,a) &= \overline{a} \vee \overline{c}\,\overline{b} \vee c\,d \\ &= a\left[\overline{c}\,\overline{b} \vee c\,d\right] \vee \overline{a}\left[1\right] \\ &= a\left[\overline{c}\,(\overline{b}) \vee c\,(d)\right] \vee \overline{a}\left[1\right] \\ &= a\left[\overline{c}\left(b(0) \vee \overline{b}(1)\right) \vee c\left(d(1) \vee \overline{d}(0)\right)\right] \vee \overline{a}\left[1\right] \end{aligned}$$

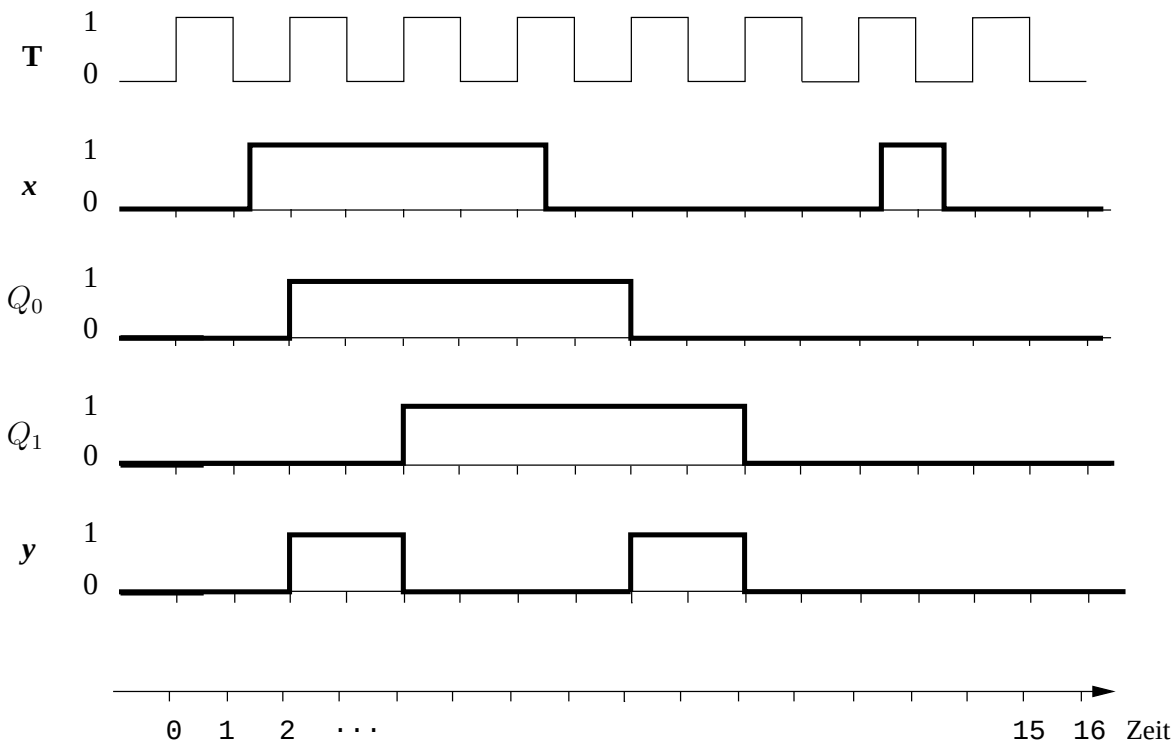


Aufgabe 3

1. Verlauf von Q:



2. Verläufe von Q_0, Q_1 und y:

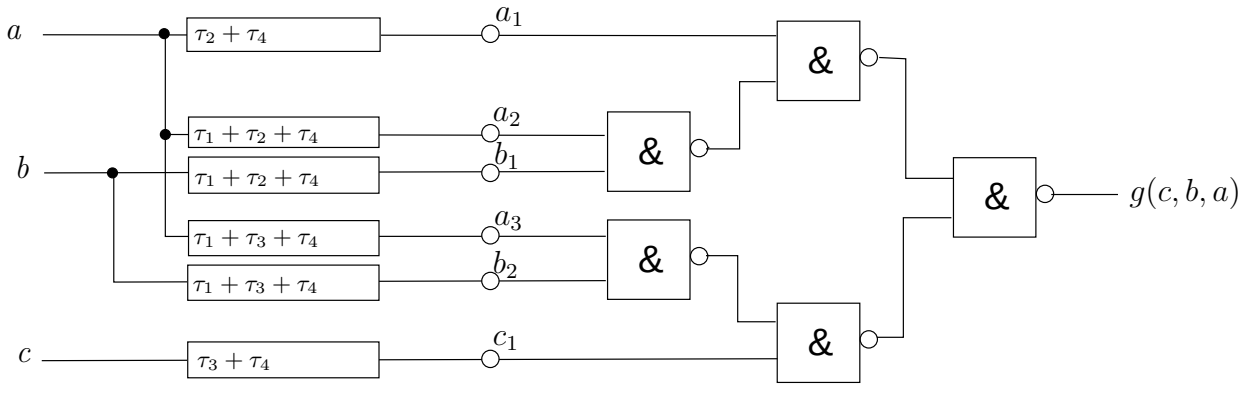


3. Funktion des Schaltwerkes: sog. synchronen Änderungsdetektor.

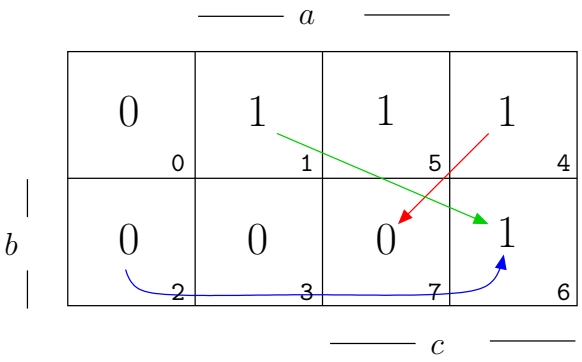
   Das Schaltwerk detektiert Änderungen am Eingang x (sowohl 01-Wechsel als auch 10-Wechsel) und liefert einen taktsynchronen Ausgangimplus y, dessen Dauer eine Taktperiode beträgt. Impluse von x, die kurzer als eine Taktperiode sind, werden vom Schaltwerk jedoch nicht erfaßt.

Aufgabe 4

1. Totzeitmodell:



2. KV-Diagramm für g:



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

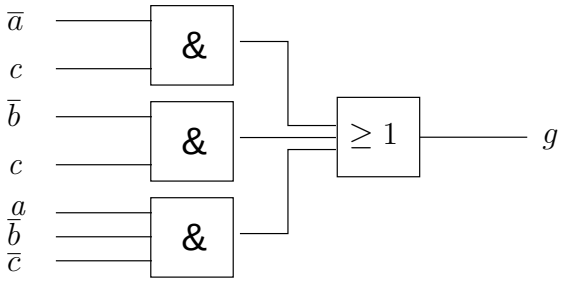
   - Übergang 2 $(c, b, a): \ (0, 0, 1) \to (1, 1, 0)$
     Bei dem Übergang ändern 3 Variablen den Wert $\Rightarrow 3! = 6$ Wege von der Anfangs- zur Endbelegung. Von den zugehörigen Folgen der Funktionswerte ist mindestens eine, z. B. $B_1 \to B_3 \to B_7 \to B_6$ nicht monoton.
     Deshalb ist der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

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

   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.

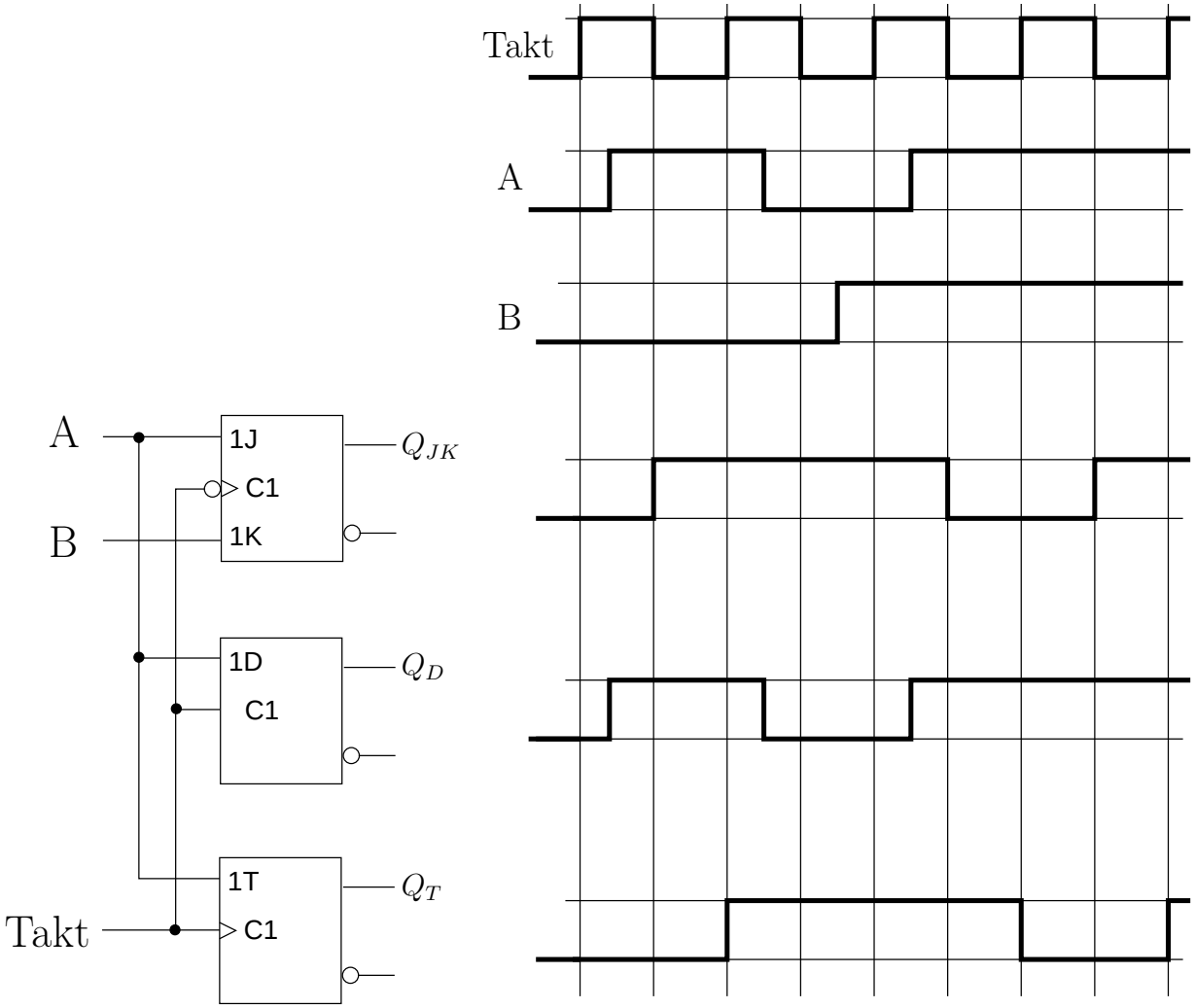
   Schaltnetz:

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



Aufgabe 5

1. Verläufe der Signale Q_{JK}, Q_D und Q_T:

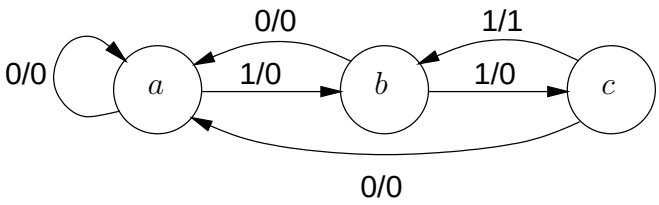


2.+3. Ablauftabelle: (Eine mögliche Lösung)

Z^t	e^t	Z^{t+1}	y^t_{Mealy}	y^t_{Moore}
a	0	a	0	0
a	1	b	0	0
b	0	a	0	0
b	1	c	0	0
c	0	a	0	1
c	1	b	1	1

Die Ausgabe hängt beim Mealy-Automaten vom Zustand und von der Eingabe ab. Beim Moore-Automaten hängt die Ausgabe nur vom Zustand ab.

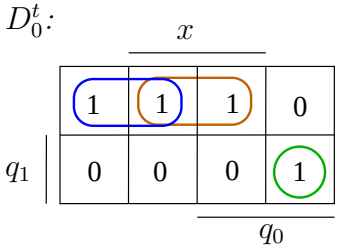
4. Automatengraph:



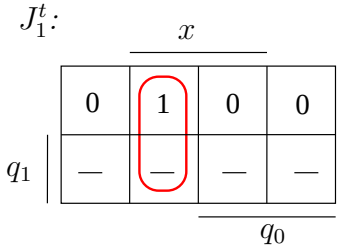
5. DMF der Ansteuerfunktionen:

q_0^t	q_1^t	x^t	q_0^{t+1}	q_1^{t+1}	D_0^t	J_1^t	K_1^t
0	0	0	1	0	1	0	–
0	0	1	1	1	1	1	–
0	1	0	0	0	0	–	1
0	1	1	0	1	0	–	0
1	0	0	0	0	0	0	–
1	0	1	1	0	1	0	–
1	1	0	1	0	1	–	1
1	1	1	0	1	0	–	0

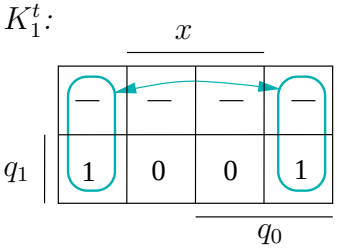
D_0^t:



J_1^t:



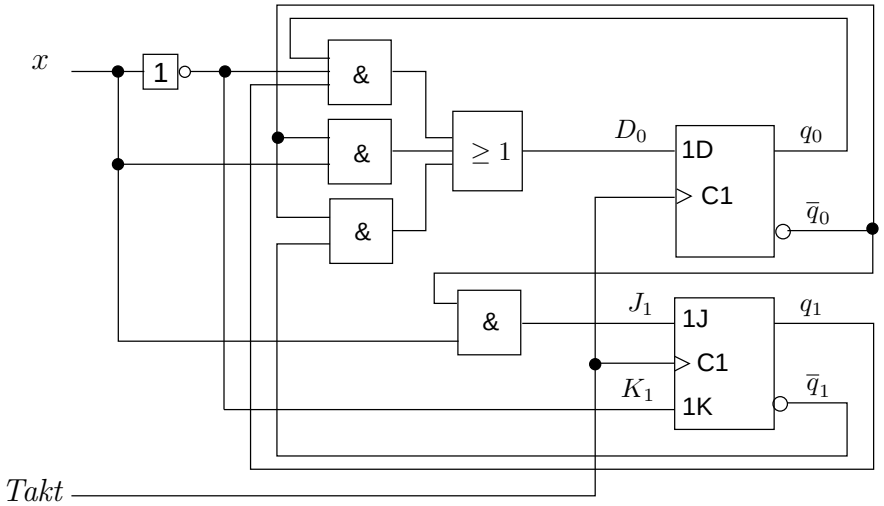
K_1^t:



Man erhält:

$$
\begin{aligned}
D_0 &= \overline{x}\, q_0\, q_1 \;\vee\; x\, \overline{q}_1 \;\vee\; \overline{q}_1 \overline{q}_0 \\
J_1 &= x\, \overline{q}_0 \\
K_1 &= \overline{x}
\end{aligned}
$$

6. Schaltung des Schaltwerks:



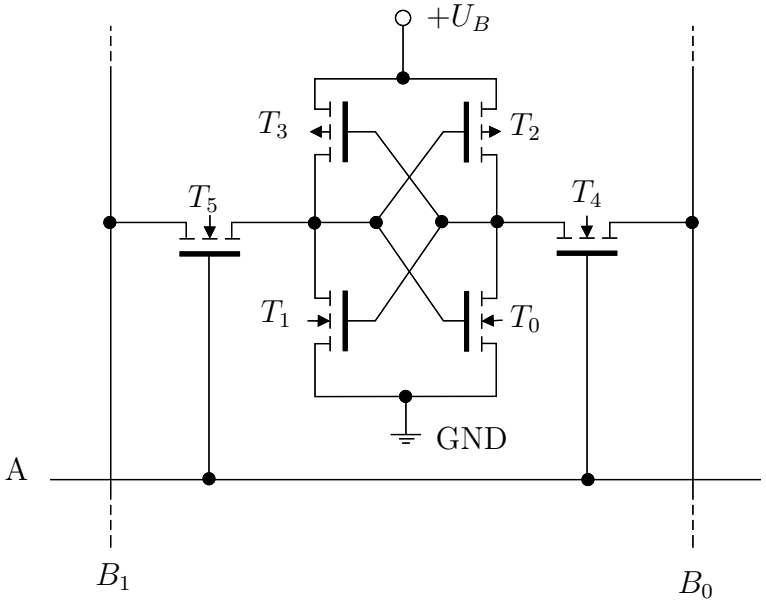
Aufgabe 6

1. *von-Neumann-Flaschenhals:* Daten und Maschinenbefehle werden über die gleiche Verbindungseinrichtung zwischen dem Prozessor und dem Hauptspeicher transportiert. Befehle und Operanden können nur nacheinander aus dem Speicher geladen werden.

2. Bits im Statusregister:

   - Übertragsbit *(Carry Flag CF)*: Übertrag bei Addition oder Subtraktion (Borrow).
   - Hilfsübertragsbit *(Auxiliary Carry AF)*: Übertrag von Bit 3 in Bit 4 des Ergebnisses bei BCD-Arithmetik.
   - Nullbit *(Zero Flag ZF)*: Zeigt an, ob das Ergebnis der letzten Operation gleich 0 war (bedingte Programmverzweigungen, Zählschleifen).
   - Vorzeichenbit *(Sign Flag SF)*: Zeigt an, dass das Ergebnis negativ ist.
   - Überlaufbit *(Overflow Flag OF)*: Bereichsüberschreitung im Zweierkomplement.
   - Even Flag *(EF)*: zeigt an, ob das Ergebnis eine gerade Zahl ist.
   - Paritätsbit *(Parity Flag PF)*: Signalisiert ungerade Parität des Ergebnisses

3. Statische CMOS-Speicherzelle:



Aufgabe 7

1. C-Kontrollstruktur in MIPS-Assembler:

```
addi    $a0, $zero, 100     # i = 100;
loop:  slt     $v0, $zero, $a0     # if 0 < i, dann $v0 = 1
beq     $v0, $zero, label   # if $v0 = 0 (d.h., i <= 0),
# dann Ende der for-Schleife
mul     $a1, $a1, $a0       # j = j * i;
subi    $a0, $a0, 1         # i--;
b       loop                # gehe zur Marke loop
label:
```

2. Fehlerfreie Version:

```
add   $v0, $zero, $zero    # summe = 0
add_array:  lw    $t0, 0($a0)          # Nächstes Element lesen
add   $v0, $v0, $t0        # Addieren
addi  $a0, $a0, 4          # Zeiger auf das nächste Element
addi  $a1, $a1, -1         # Zähler dekrementieren
bgtz  $a1, add_array       # Abbruchbedingung
```

3. Inhalte der Zielregister:

Befehl	Zielregister = (z. B. `$s6 = 0x0000 F00A`)
`subi $s1, $zero, 0x2`	`$s1 = 0xFFFF FFFE`
`srl  $s2, $s1, 4`	`$s2 = 0x0FFF FFFF`
`slti $s3, $s2, 100`	`$s3 = 0x0000 0000`
`lui  $s4, 0x40`	`$s4 = 0x0040 0000`
`xor  $s5, $s1, $s4`	`$s5 = 0xFFBF FFFE`

4. (a) Registerinhalte:

Register	Inhalt
`$t1`	`$t1 = 0x0000 000C`
`$t2`	`$t2 = 0x0000 001F`
`$t3`	`$t3 = 0x0000 0013`
`$t4`	`$t4 = 0x0000 0029`

   (b) MIPS-Code zur Speicherung der Adresse von `vec` im Register `$s0`:

```
la $s0, vec
```

Aufgabe 8

1. Echte Datenabhängigkeiten und Steuerflussabhängigkeiten:
   - Echte Abhängigkeiten:

     $S1 \rightarrow S3$ $S1 \rightarrow S6$ $S1 \rightarrow S7$ $S1 \rightarrow S9$ $S1 \rightarrow S10$ $S2 \rightarrow S4$
     $S3 \rightarrow S4$ $S4 \rightarrow S5$ $S4 \rightarrow S6$ $S5 \rightarrow S6$ $S7 \rightarrow S9$ $S7 \rightarrow S10$
     $S9 \rightarrow S10$
   - Steuerflussabhängigkeiten:

     $S8 \rightarrow S9$
2. Behebung der Pipelinekonflikte:

```
S1:     add  $t1, $zero, $zero
S2:     lw   $t3, 0x1500($zero)
S3:     lw   $t4, 0x5000($t1)
nop
S4:     add  $t5, $t4, $t3
S5:     ori  $t5, $t5, 0x10
S6:     sw   $t5, 0x400($t1)
S7:     addi $t1, $t1, 4
S8:     bnez $t2, exit
nop
nop
nop
S9:     srli $t1, $t1, 2
S10:    sw   $t1, 0x2000($zero)
```

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

1. (a) Blockgröße in Bytes: 4 Bit Byte Offset $\Rightarrow$ Blockgröße $= 2^4 = 16$ Byte

   (b) Anzahl der Einträge:

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

   (c) Cache-Organisation:

   12 Bit Index-Feld $\Rightarrow$ Es lassen sich $2^{12} = 4$ K Sätze im Cache addressieren

   $$\text{Assoziativität} = \frac{32\ K}{4\ K} = 8$$

   Der Cache ist als 8-fach assoziativer Speicher (*8-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)

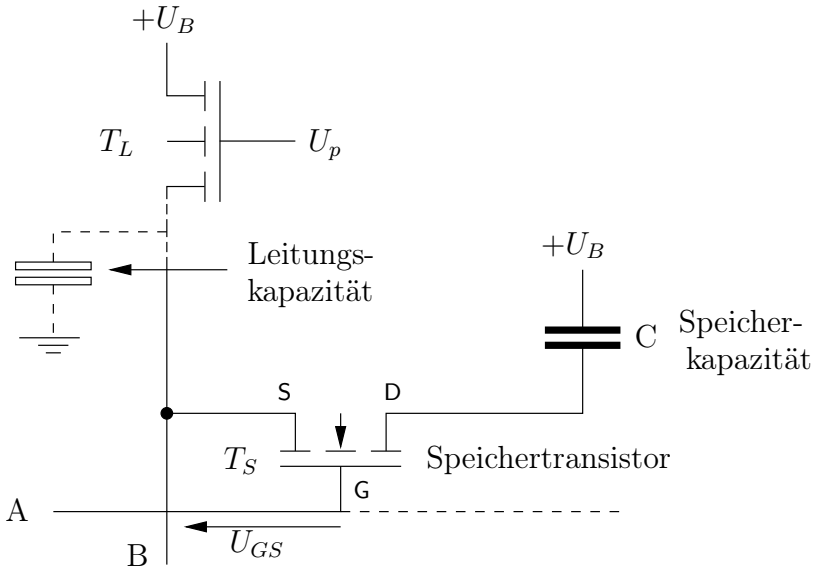
   Speicherbedarf für eine Zeile: $21 + 1 + 64$ Bit = 86 Bits
   Speicherbedarf für den gesamten Cache: $86 \cdot 256 \cdot 4$ = 88064 Bits = 11008 Byte

3.

Adresse	**0**	**16**	**48**	**8**	**56**	**16**	**8**	**56**	**32**	**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 10

1. Aufbau einer dynamischen RAM-Speicherzelle:



2. 512×8-Organisation: 512 Zellen mit 8-Bit Wörter ⇒ 512 Zellen müssen adressiert werden. Dazu sind 9 Adreßleitungen erforderlich.

3. Es sind 8 RAM-Bausteine der Organisation 8k×1 notwendig, um einen Speicher mit einer Kapazität von 8k Wörter und einer Wortbreite von 8 Bit zu realisieren.

4. ROM-Baustein der Speicherkapazität von 2048 Bits und 8 Adreßleitungen ⇒ Es können 256 Zellen adressiert werden ⇒ 256×8-Organisation

5. (a) RISC-Prozessor: Prozessor mit einem reduzierten Befehlssatz.
   (b) Direkter Speicherzugriff (DMA): Datentransfer ohne Beteiligung des Prozessors.