KIT-Fakultät für Informatik
Prof. Dr. Mehdi Tahoori, Prof. Dr. Wolfgang Karl

Musterlösungen zur Klausur

Digitaltechnik und Entwurfsverfahren (TI-1)

und

Rechnerorganisation (TI-2)

am 15. August 2022, 08:00 – 10:00 Uhr

Name:	Vorname:	Matrikelnummer:
Bond	James	007

Digitaltechnik und Entwurfsverfahren (TI-1)	
Aufgabe 1	10 von 10 Punkten
Aufgabe 2	9 von 9 Punkten
Aufgabe 3	7 von 7 Punkten
Aufgabe 4	9 von 9 Punkten
Aufgabe 5	10 von 10 Punkten

Rechnerorganisation (TI-2)	
Aufgabe 6	12 von 12 Punkten
Aufgabe 7	12 von 12 Punkten
Aufgabe 8	12 von 12 Punkten
Aufgabe 9	5 von 5 Punkten
Aufgabe 10	4 von 4 Punkten

Gesamtpunktzahl:	90 von 90 Punkten

	Note: **1,0**

Aufgabe 1 *Schaltfunktionen* (10 Punkte)

1. DMF von f: 3 P.
   Aus dem Schaltnetz ablesen:

$$
\begin{aligned}
&((a \barwedge b) \barwedge c) \barwedge ((a \barwedge b) \barwedge c) \\
&= \overline{\overline{ab}\ c} \barwedge \overline{\overline{ab}\ c} \\
&= \overline{\overline{\overline{ab}\ c}} \\
&= \overline{ab}\ c \\
&= (\overline{a} \vee \overline{b})c \\
&= \overline{a}c \vee \overline{b}c
\end{aligned}
$$

2. • KV-Diagramm von $f(c, b, a)$: 1 P.

$f(c,b,a)$	a		c	
	0	0	1	1
b	0	0	0	1

• DNF von $f(c, b, a)$: 1 P.

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

3. Kern-Primimplikante: C **1 P.**

   Reduzierte Tabelle: (Gestrichene Spalten: a, d)

	b	c	e
A	×	×	
B		×	×
D			×
E	×	×	×

4. Dominierte Minterme: b **1 P.**

   Reduzierte Tabelle: (Gestrichene Spalte: c)

	b	e	Kosten
A	×		$\frac{1}{3}$
B		×	$\frac{1}{3}$
D		×	1
E	×	×	$\frac{1}{3}$

5. Dominierende Primimplikate: E **2 P.**

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

	b	e
E	×	×

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

Aufgabe 2 *Schaltnetze und CMOS-Technologie* (9 Punkte)

1. Eigenschaften *Transmission*-Gate: 2 P.

   - Die beiden Steuereingänge des Transmission-Gates müssen intern negiert und nicht negiert vorhanden sein (komplementär).
   - Ideale Übertragung von „High“ und „Low“ Signalen

2. CMOS-Schaltbild des *Transmission*-Gates: 2 P.

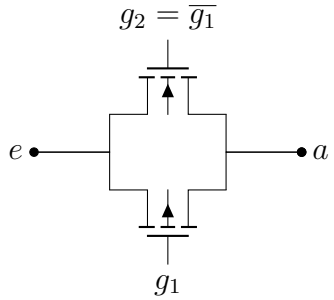


Abbildung 1: CMOS-Schaltbild *Transmission*-Gate

3. $g(d, c, b, a)$: 4 P.
   Das gegebene Schaubild ist kein gültiges CMOS. Somit kann keine Schaltfunktion angegeben werden.

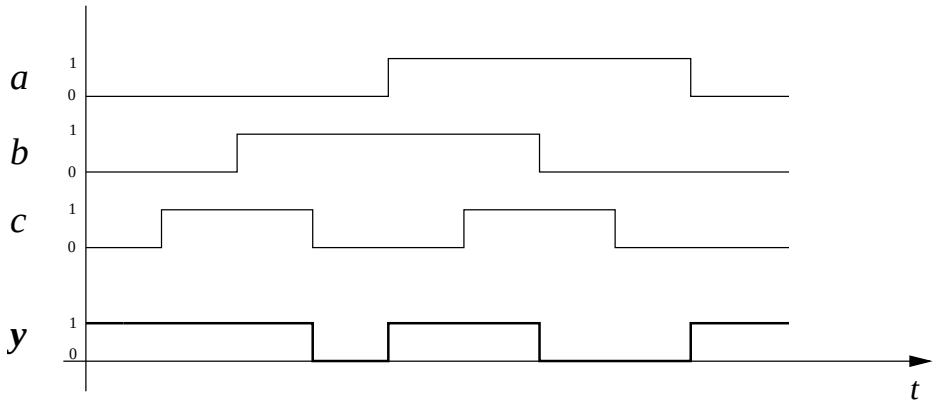
   (Bei $d = 0$ und $c = 1$ wird V_{dd} zu *Ground* durchgeschaltet)

4. Zweistufige disjunktive Form von $g(d, c, b, a)$: 1 P.
   Siehe oben.

Aufgabe 3 *Laufzeiteffekte* (7 Punkte)

1. Verlauf von y: 1 P.
   (Nicht gefordert - nur das Schaubild ist gefordert)

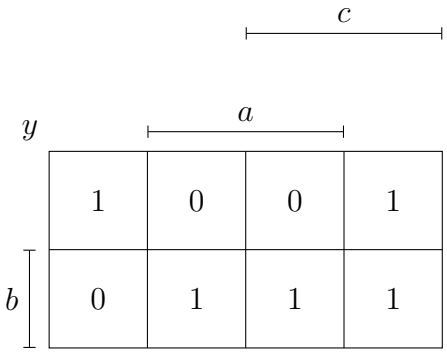
$$\begin{aligned} y &= \mathrm{NAND}_3((c \mathbin{\overline{\wedge}} b), (b \mathbin{\overline{\wedge}} a), (a \vee b)) \\ &= \overline{(c \mathbin{\overline{\wedge}} b) \wedge (b \mathbin{\overline{\wedge}} a) \wedge (a \vee b)} \\ &= cb \vee ba \vee \overline{a}\overline{b} \end{aligned}$$



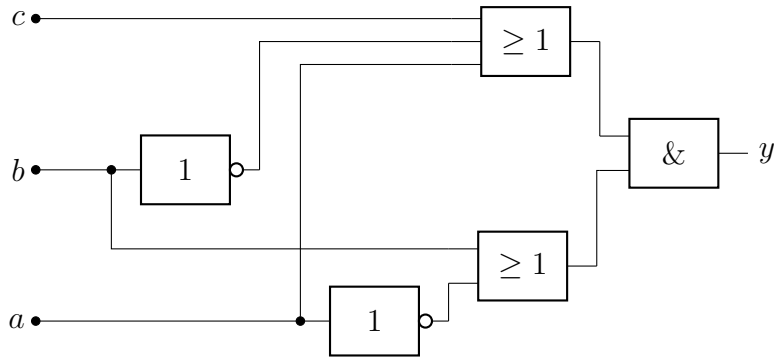
2. Keiner der Übergänge ist mit einem Funktionshasard behaftet. 2 P.
   Begründung: Bei allen Übergängen wechselt nur eine Variable. Diese Übergänge sind stets funktionshasardfrei.

   Ja, es kann kurzzeitig ein falscher Wert am Ausgang entstehen.
   Begründung: Bei den Übergängen, bei denen a oder b wechseln, können Strukturhasards auftreten, die Hasardfehler verursachen.

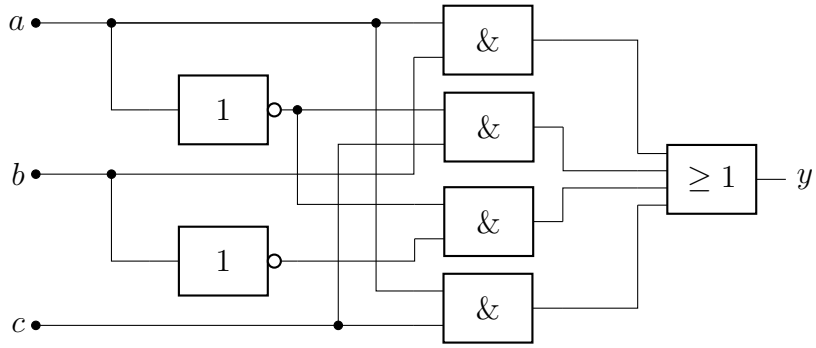
3. Ein Schaltnetz, welches die Disjunktion aller Primimplikanten bzw. die Konjunktion aller Primimplikate realisiert, hat dieselbe logische Funktion und weist bei den betrachteten Übergängen keine Hasards auf. Begründung: Satz von Eichelberger. 4 P.



Konjunktion aller Primimplikate: $y = (b \vee \overline{a}) \wedge (c \vee \overline{b} \vee a)$



oder Disjunktion aller Primimplikanten $y = cb \vee ba \vee \overline{b}\overline{a} \vee c\overline{a}$



Aufgabe 4 *Schaltwerke* (9 Punkte)

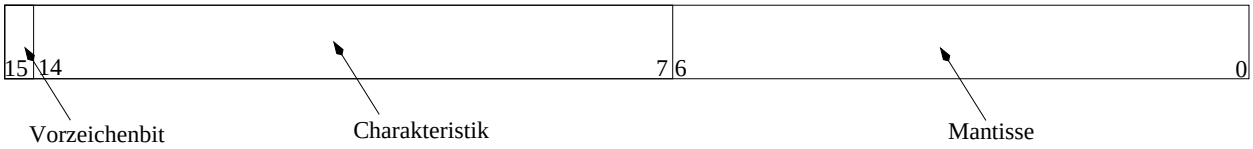
1. Unterschied Schaltnetz und Schaltwerk: 1 P.
   Bei Schaltnetzen hängt die Ausgabe lediglich von den Eingangsvariablen ab. Bei Schaltwerken ist Ausgabe abhängig von den Eingansvariablen und dem aktuellen Zustand des Schaltwerkes.

2. 8 P.

Schaltwerk	**1**	**2**	**3**	**4**
zählt vorwärts	×			×
zählt rückwärts		×		
ist synchron		×	×	×
kann bei *jedem* Zählerstand mit Hilfe von x angehalten werden.	×	×		

Aufgabe 5 *Rechnerarithmetik & Verschiedenes* (10 Punkte)

1. Unterteilung von *Bfloat16*: 1 P.



2. $2,125_{10}$ in *Bfloat16*: 2 P.

$$2,125_{10} \Rightarrow Sign = 0$$
$$2,125 = 1,0001_2 \cdot 2^1 \Rightarrow Man = 0001000$$
$$Exp = 1 \Rightarrow Char \;=\; Exp + 127_{10} = 128_{10} = 1000\ 0000_2$$

15	14 … 7	6 … 0
0	10000000	0001 000

3. Größte positive Zahl *Bfloat16*: 1 P.

15	14 … 7	6 … 0
0	11111110	1111 111

4. Anzahl Möglichkeiten der Null: 1 P.
   2 Möglichkeiten

5. Gründe für die Verwendung: 2 P.

   - Konvertierung zwischen IEEE und Bfloat16 sehr einfach (Weglassen der Mantissen-Stellen)
   - Gleicher Zahlenbereich, nur die Genauigkeit der Darstellung sinkt im Vergleich zu IEEE
   - Reduzierter Speicherbedarf (die Genauigkeit ist in den Anwendungsfelder meist ausreichend)
   - ...

6. Anzahl Prüfbits: 1 P.
   Aufwand: $2^k \geq m + k + 1$. Hier: m = 64 $\Rightarrow k = 7$

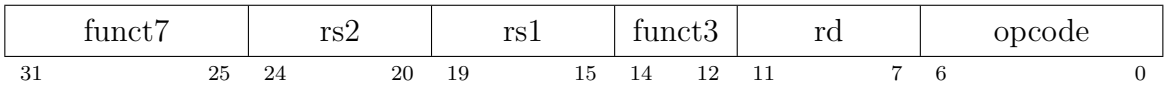
7. Unterschied *Carry-Ripple*-Addierer und *Carry-Lookahead*-Addierer: 2, P.
   Bei *Carry Ripple*-Addierern muss bei der Addition einer Stelle auf den Übertrag aus den vorhergehenden Stelle gewartet werden. Die Additionszeit ist proportional zur Anzahl der Stellen.

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

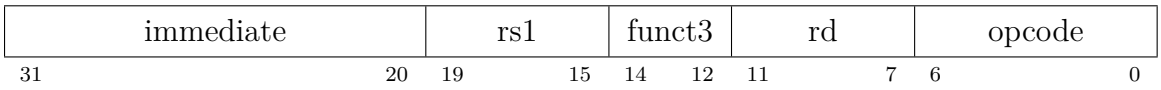
Aufgabe 6 *RISC-V Assembler* (12 Punkte)

1. Anzahl Befehlsformate: 1 P.
   6

2. Kodierung von R-Typ und I-Typ: 3 P.
   R-Typ:



   I-Typ:



3. Unterschied Ausnahme und Unterbrechung: Eine Ausnahme tritt synchron zum Programmablauf aus und steht im Zusammenhang mit bestimmten Befehlen. Zum Beispiel bei ungültigen Befehlen oder arithmetischen Fehlern. 2 P.
   Eine Unterbrechung tritt asynchron zum Programmablauf auf und steht somit nicht in direkter Abhängigkeit zu einem Befehl. Zum Beispiel bei der Kommunikation mit Ein- und Ausgabegeräten (Drücken einer Taste).

4. Ausführungsmodell: 1 P.
   Register-Register-Modell

5. Konvention der Register: 2 P.
   Die Register *t0-t6* sollen laut Konvention für temporäre Variablen verwendet werden. Die Inhalte müssen vor einem Unterprogrammaufruf vom Programm gegebenenfalls gesichert werden.
   Die Register *s0-s11* sollen laut Konvention für langlebige Variablen verwendet werden. Die Inhalte müssen vom Unterprogramm selbst gespeichert werden und nach Beendigung des selbigen wieder hergestellt werden.
   Auf Hardwareebene gibt es allerdings keinen Unterschied zwischen den Registern.

6. RISC-V Assembler: 3 P.

```
        addi a0, zero, 10   # i = 10
loop:   slt t0, zero, a0    # if 0 < i, dann t0 = 1
        beq t0, zero, label # if t0 = 0 (d.h. i <= 0)
                            # dann Ende der for-Schleife
        mul a1, a1, a0      # j = j * i
        addi a0, a0, -1     # i--
        b loop              # gehe zu Marke loop
label:
```

Aufgabe 7 *Pipelining* (12 Punkte)

1. 4 P.

   - Echte Datenabhängigkeiten:

   $$S1 \rightarrow S2(R1) \qquad S2 \rightarrow S3(R4)$$
   $$S3 \rightarrow S4(R1) \qquad S2 \rightarrow S4(R4)$$

   - Gegenabhängigkeiten:

   $$S1 \rightarrow S4(R3) \qquad S2 \rightarrow S4(R3)$$
   $$S2 \rightarrow S3(R1)$$

   - Ausgabeabhängigkeiten:

   $$S1 \rightarrow S3(R1)$$

2. Zustand der Pipeline und Register: 3 P.

Takt	**IF/ID**	**OF**	**EX**	**WB**	**R1**	**R2**	**R3**	**R4**
1	S1	-	-	-	4	7	3	5
2	S2	S1	-	-	4	7	3	5
3	S3	S2	S1	-	4	7	3	5
4	S4	S3	S2	S1	10	7	3	5
5	-	S4	S3	S2	10	7	3	1
6	-	-	S4	S3	35	7	3	1
7	-	-	-	S4	35	7	15	1

Anzahl der Takte: 7

3. Behebung der Pipelinekonflikte durch Einfügen von NOP-Befehlen: 4 P.

```
S1: ADD R1, R2, R3      # R1 = R2 + R3
    NOP
    NOP
S2: SUB R4, R1, R3      # R4 = R1 - R3
    NOP
    NOP
S3: MUL R1, R4, R2      # R1 = R4 * R2
    NOP
    NOP
S4: ADD R3, R1, R4      # R3 = R1 + R4
```

   Anzahl der Takte: 13

4. Forwarding-Techniken: 1 P.

   Unter Forwarding versteht man das Bereitstellen von Ergebnissen von vorherigen Befehlen über einen Bypass bevor diese in ein Universalregister geschrieben worden sind.

Aufgabe 8 *Cache-Speicher* (12 Punkte)

1. Direkt-abgebildeter Cache mit 16 Speicherblöcken: 6 P.

Adresse	Hilfsspalte (Binär)	Tag	Index	Offset	Hit/Miss
0x03	0b0000 0011	0	3	-	Miss
0xb4	0b1011 0100	b	4	-	Miss
0x2b	0b0010 1011	2	b	-	Miss
0x02	0b0000 0010	0	2	-	Miss
0xbf	0b1011 1111	b	f	-	Miss
0x58	0b0101 1000	5	8	-	Miss
0xbe	0b1011 1110	b	e	-	Miss
0x0e	0b0000 1110	0	e	-	Miss
0xb5	0b1011 0101	b	5	-	Miss
0x2c	0b0010 1100	2	c	-	Miss
0xba	0b1011 1010	b	a	-	Miss
0xfd	0b1111 1101	f	d	-	Miss

2. Direkt-abgebildeter Cache mit 8 Speicherblöcken: 6 P.

Adresse	Hilfsspalte (Binär)	Tag	Index	Offset	Hit/Miss
0x03	0b0000 0011	0	1	1	Miss
0xb4	0b1011 0100	b	2	0	Miss
0x2b	0b0010 1011	2	5	1	Miss
0x02	0b0000 0010	0	1	0	Hit
0xbf	0b1011 1111	b	7	1	Miss
0x58	0b0101 1000	5	4	0	Miss
0xbe	0b1011 1110	b	7	0	Hit
0x0e	0b0000 1110	0	7	0	Miss
0xb5	0b1011 0101	b	2	1	Hit
0x2c	0b0010 1100	2	6	0	Miss
0xba	0b1011 1010	b	5	0	Miss
0xfd	0b1111 1101	f	6	1	Miss

Aufgabe 9 *Virtuelle Speicherverwaltung* (5 Punkte)

1. Physikalische Adressen: 5 P.

   - 3088:

   $$\begin{aligned} 3088 : 1024 = 3 \quad R \quad 16 \\ 3 \rightarrow 0 \\ 0 + 16 = \underline{\underline{16}} \end{aligned}$$

   - 1420:

   $$\begin{aligned} 1420 : 1024 = 1 \quad R \quad 396 \\ 1 \rightarrow 3 \\ 3 \cdot 1024 + 396 = \underline{\underline{3468}} \end{aligned}$$

   - 2555:

   $$\begin{aligned} 2555 : 1024 = 2 \quad R \quad 507 \\ 2 \rightarrow - \\ \rightarrow \textit{page fault} \end{aligned}$$

   - 1023:

   $$\begin{aligned} 1023 : 1024 = 0 \quad R \quad 1023 \\ 0 \rightarrow 1 \\ 1 \cdot 1024 + 1023 = \underline{\underline{2047}} \end{aligned}$$

   - 1024:

   $$\begin{aligned} 1024 : 1024 = 1 \quad R \quad 0 \\ 1 \rightarrow 3 \\ 3 \cdot 1024 + 0 = \underline{\underline{3072}} \end{aligned}$$

Aufgabe 10 *Verschiedenes* (4 Punkte)

1. Entscheidende Nachteil: 1 P.

   Dekodierung 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.

2. Vorteile eines DMA-Controllers: 1 P.

   Die Speicherzugriffe für das Holen der Lade-, Speicher- und Schleifenbefehle entfallen, da die Datenübertragung hardwaremäßig ausgeführt wird ⇒ nur zwei (ggf. sogar nur ein) Speicherzugriff/Datum nötig.

   Der Prozessor wird entlastet und kann während des direkten Speicherzugriffs des Controllers andere Aufgaben tun (sofern diese nicht den Systembus benötigen).

3. Beschleunigung durch TLB: 1 P.

   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.

4. Befehlssatzarchitektur: 1 P.

   Unter Befehlssatzarchitektur versteht man die gesamte nach außen sichtbare Architektur eines Prozessors. Sie bildet die Schnittstelle zwischen Software und Hardware und ermöglicht so eine vollständige Abstraktion der Hardware.