



UNIVERSITÄT KARLSRUHE (TH)  
Fakultät für Informatik  
Institut für Prozessrechentechik, Automation und Robotik (IPR)  
Prof. Dr. U. Brinkschulte, Dr. T. Asfour

## Aufgabenblätter

zur Klausur „Technische Informatik I/II“  
am 12. September 2005, 14.00 – 16.00 Uhr

- Beschriften Sie bitte gleich zu Beginn jedes Lösungsblatt deutlich lesbar mit Ihrem Namen und Ihrer Matrikelnummer.
- Diese Aufgabenblätter werden nicht abgegeben. Tragen Sie Ihre Lösung deshalb ausschließlich in die für jede Aufgabe vorgesehenen Bereiche der Lösungsblätter ein. Lösungen auf separat abgegebenen Blättern werden nicht gewertet.
- Außer Schreibmaterial sind während der Klausur keine Hilfsmittel zugelassen. Täuschungsversuche durch Verwendung unzulässiger Hilfsmittel führen unmittelbar zum Ausschluss von der Klausur und zur Note „nicht bestanden“.
- Soweit in der Aufgabenstellung nichts anderes angegeben ist, tragen Sie in die Lösungsblätter bitte nur die Endergebnisse ein. Die Rückseiten der Aufgabenblätter können Sie als Konzeptpapier verwenden. Weiteres Konzeptpapier können Sie auf Anfrage während der Klausur erhalten.
- Halten Sie Begründungen oder Erklärungen bitte so kurz wie möglich. (Der auf den Lösungsblättern für eine Aufgabe vorgesehene Platz steht übrigens in keinem Zusammenhang mit dem Umfang einer korrekten Lösung!)
- Die Gesamtpunktzahl beträgt 90 Punkte. Zum Bestehen der Klausur sind mindestens 40 Punkte zu erreichen.

*Viel Erfolg und viel Glück !*

### Aufgabe 1 Schaltfunktionen

(11 Punkte)

1. Eine unvollständig definierte Schaltfunktion  $y = f(d, c, b, a)$  sei durch ihre Eins- und *don't care*-Stellen (Abkürzung d) gegeben:

$$y = \text{MINt}(2, 3, 4, 5, 6, 8, 11) \vee \text{d}(7, 9, 14)$$

- (a) Tragen Sie *alle* Prim-Einsblöcke der Schaltfunktion  $y$  ins KV-Diagramm im Lösungsblatt ein. Geben Sie *alle* Primimplikanten von  $y$  an. 3 P.
  - (b) Geben Sie *alle* disjunktiven Minimalformen (DMF) der Funktion  $f$  an. 2 P.
  - (c) Geben Sie die kürzeste Gleichung für  $f$  an, die als Ausgangspunkt für die Gewinnung *aller* Primimplikanten mit Hilfe des Nelson-Verfahrens geeignet ist. 2 P.
2. Gegeben seien das KV-Diagramm einer Schaltfunktion  $h(d, c, b, a)$  und eine Menge von Einsblöcken, die bereits in das KV-Diagramm eingezeichnet wurden (siehe Bild 1).

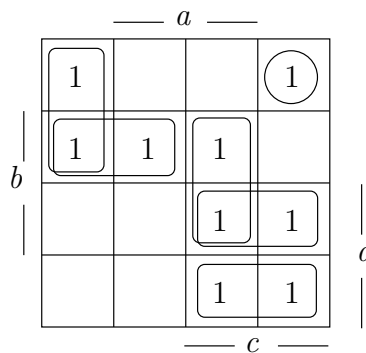


Bild 1: Das KV-Diagramm der Funktion  $h(d, c, b, a)$

- Zeichnen Sie die in Bild 1 nicht dargestellten Consensus-Blöcke in das im Lösungsblatt wiederholt angegebene KV-Diagramm ein. 2 P.
3. Beweisen Sie *schaltalgebraisch* die folgenden Identitäten 2 P.

- (a)  $y \vee \bar{x}z \vee x\bar{y} = x \vee y \vee z$
- (b)  $a \vee a = a$

**Aufgabe 2** *Laufzeiteffekte*

(5 Punkte)

Gegeben ist das in Bild 2 dargestellte Schaltnetz. Alle Gatter haben eine Totzeit von  $5\text{ ns}$ .

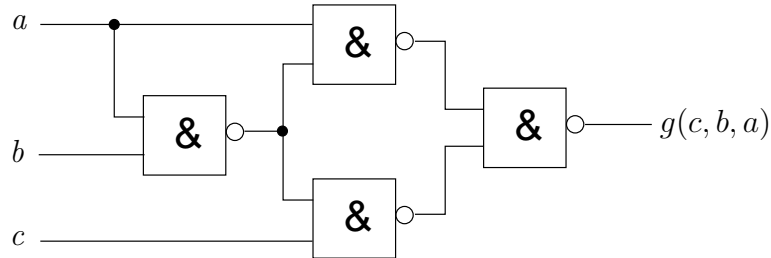


Bild 2: Schaltnetz der Funktion  $g(c, b, a)$

1. Geben Sie das endgültige Totzeitmodell des Schaltnetzes an. Tragen Sie die Pfadvariablen in Ihre Lösung ein und geben Sie die Werte der Pfadverzögerungen an. 2 P.
2. Untersuchen Sie den Übergang  $(c, b, a) : (0, 1, 0) \rightarrow (1, 0, 0)$  auf Funktionshasards. Begründen Sie Ihre Antwort. 1 P.
3. Geben Sie eine Realisierung des Schaltnetzes an, die frei von allen statischen Strukturhasards ist. Begründen Sie Ihre Antwort. 2 P.

### Aufgabe 3 Schaltwerke

(9 Punkte)

1. Vervollständigen Sie die Verläufe der Signale  $Q_{JK}$ ,  $Q_D$  und  $Q_T$  für die angegebenen Eingangssignale. Nehmen Sie an, dass  $Q(t = 0)$  gleich 0 für alle Flipflops ist.

3 P.

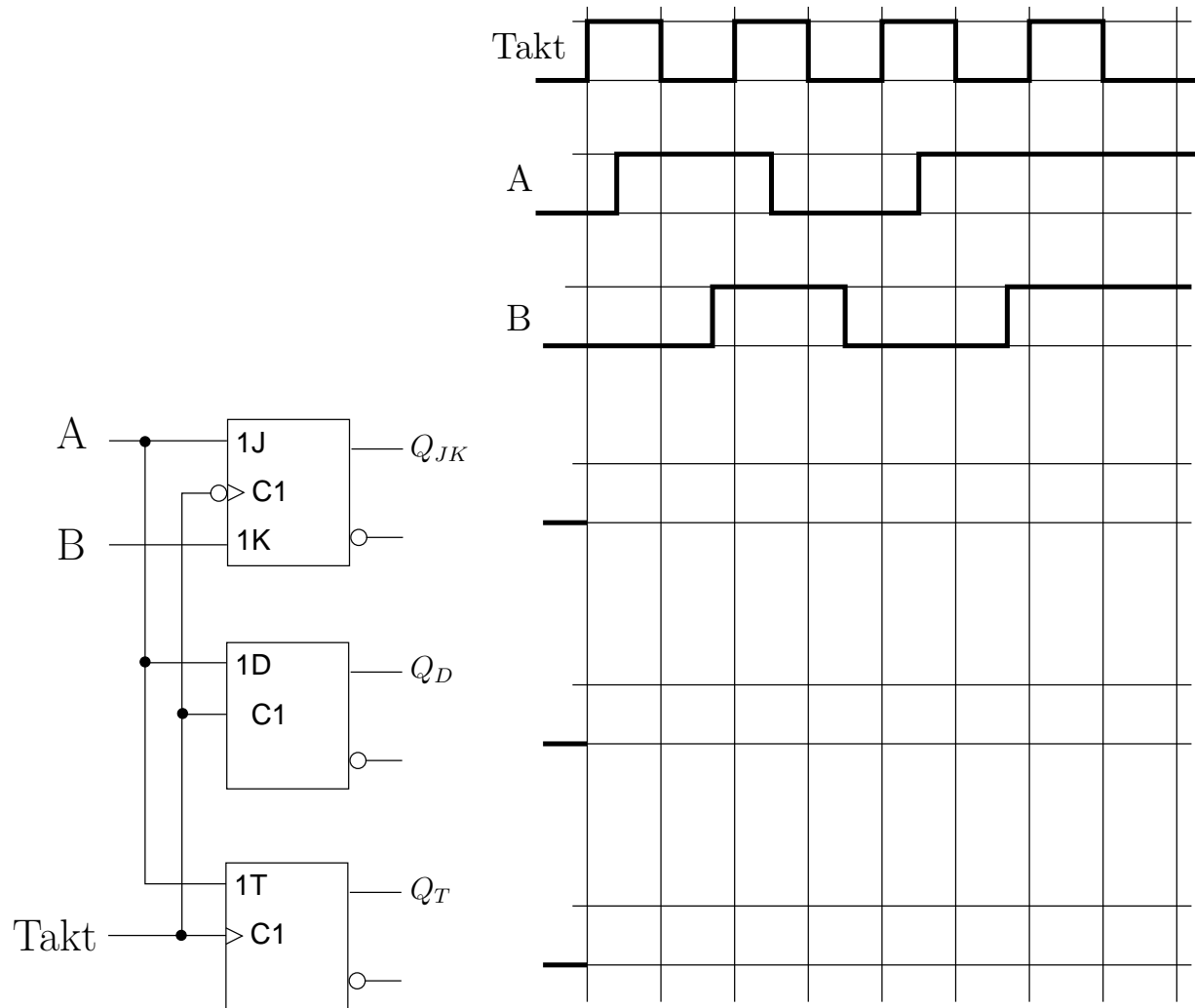


Bild 3: Flipflops und Zeitdiagramm

**Hinweis:** Achten Sie auf die unterschiedliche Ansteuerung der Flipflops.

2. Gegeben ist das in Bild 4 dargestellte Schaltwerk.

- (a) Wie viele Zustände kann das Schaltwerk maximal annehmen? 1 P.  
 (b) Ist es möglich, den Automatentypen anzugeben? Begründen Sie Ihre Antwort. 1 P.

**Hinweis:** Die Ausgabevariablen sind *nicht* gleich den Zustandsvariablen!

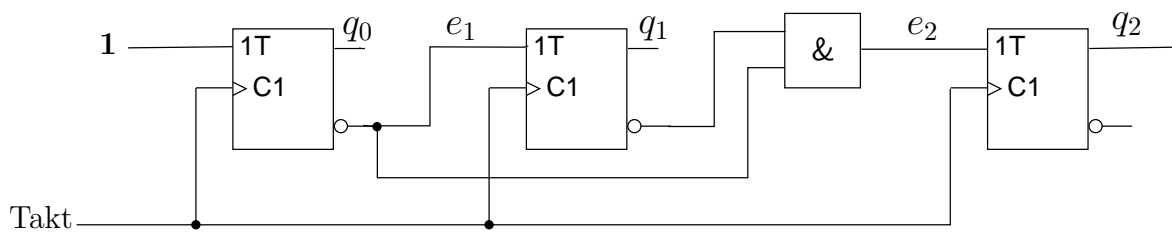


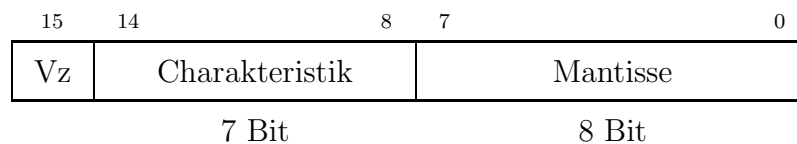
Bild 4: Schaltwerk

- (c) Vervollständigen Sie die Verläufe der Signale  $q_2, q_1$  und  $q_0$  im Lösungsblatt. Nehmen Sie an, dass  $q_i(t = 0) = 0$  für  $i = 0, 1, 2$ . 3 P.
- (d) Welche Funktion erfüllt das Schaltwerk? 1 P.

### Aufgabe 4 Rechnerarithmetik

(11 Punkte)

1. Geben Sie die Dezimalzahl  $0.\bar{2}$  (d. h.  $0.222\cdots$ ) an als Dualzahl, Hexadezimalzahl und als eine Zahl zur Basis 7. 2 P.
2. Wie viele Stellen braucht man im Dualzahlensystem mindestens, damit die Zahl  $32_{10}$  in Zweierkomplement-Form darstellbar ist? Begründen Sie Ihre Antwort. 1 P.
3. Betrachten Sie das folgende 16-Bit Format für Gleitkommazahlen: 4 P.



Dabei sei:

- Vorzeichen:  $Vz = 0$  für positive Zahlen und  $Vz = 1$  für negative Zahlen
- Charakteristik: 7 Bit
- Mantisse: 8 Bit, wobei die führende Eins nicht explizit dargestellt wird.
- Normalisierung: Wie beim IEEE 754-Floating-Point-Standard.

Multiplizieren Sie die Gleitkommazahlen 1100 0100 0011 0000 und 1011 1101 1001 1001 und stellen Sie das Ergebnis im obigen Format in normalisierter Form dar.

**Hinweis:** Bei der Addition der Charakteristiken  $c_1 = e_1 + o$  und  $c_2 = e_2 + o$  muss die Summe außerdem um den Offset  $o$  korrigiert werden, um die richtige Ergebnischarakteristik  $c = (e_1 + e_2) + o$  zu erhalten.

4. Bei einem Multiplizierer werden die Teilprodukte  $p_0, p_1, p_2, p_3, p_4$  und  $p_5$  erzeugt. Für die Addition der Teilprodukte soll eine Schaltung entworfen werden, die sechs binäre Stellen addiert, um eine 3-Bit Summe  $s_2 s_1 s_0$  zu berechnen (siehe Beispiel).

4 P.

**Beispiel**

A	0	1	1
B	0	1	0
C	0	1	1
D	0	1	0
E	0	1	0
+ F	0	1	1
$s_2 s_1 s_0$	000	111	011

Diese Aufgabe soll mit Hilfe der in Bild 5 dargestellten Schaltung gelöst werden. Vervollständigen Sie hierzu die Schaltung im Lösungsblatt. Dabei dürfen Sie *keine* zusätzlichen Gatter verwenden.

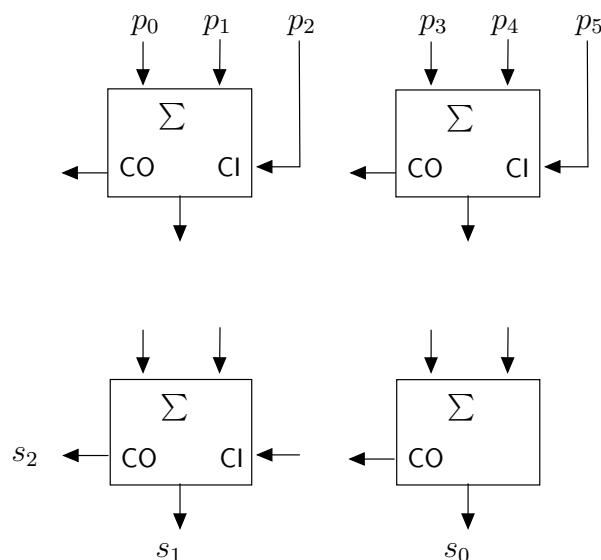


Bild 5: Schaltung zur Berechnung der Summe  $s_2 s_1 s_0$

### Aufgabe 5 Register-Transfer & ALU-Design (9 Punkte)

1. Gegeben sei ein Datenpfad bestehend aus 3 Registern  $RA$ ,  $RQ$  und  $RM$ , vier Flipflops  $S$  (Start),  $F$ ,  $R$  und  $D$  (Done). Betrachten Sie die folgenden bedingten Register-Transfer-Anweisungen (RT-Operationen):

4 P.

$$S : RM \leftarrow 0, \quad S \leftarrow 0, \quad F \leftarrow 1, \quad D \leftarrow 0$$

$$F : F \leftarrow 0, \text{ if } RA = 0 \text{ then } D \leftarrow 1 \text{ else } R \leftarrow 1$$

$$R : RM \leftarrow RM + RQ, \quad RA \leftarrow RA - 1, \quad R \leftarrow 0, \quad F \leftarrow 1$$

Simulieren Sie die obigen RT-Operationen, indem Sie die Tabelle 1 im Lösungsblatt vervollständigen. Die Ausführung der Operationen wird mit  $D = 1$  angehalten.

Takt	$S$	$F$	$R$	$D$	$RA$	$RQ$	$RM$
1	1	0	0	0	1	3	1
2							

Tabelle 1: Register-Transfer-Anweisungen

2. Das  $i$ -te Bit einer arithmetisch logischen Einheit (ALU) ist wie in Bild 6 realisiert. Dabei stellen  $A_i$  und  $B_i$  das  $i$ -te Bit der Operanden  $A$  und  $B$  und  $F_i$  das  $i$ -te Bit des Ergebnisses  $F$  dar.

5 P.

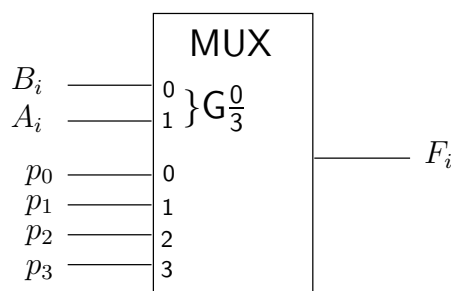


Bild 6: ALU-Design

Vervollständigen Sie die Tabelle im Lösungsblatt, so dass die dort angegebenen Operationen realisiert werden.

**Aufgabe 6** *Multiple Choice*

(4 Punkte)

Kreuzen Sie bitte für jede der Behauptungen im Lösungsblatt an, ob sie Ihrer Meinung nach richtig oder falsch ist. Nicht angekreuzte Behauptungen zählen nicht und gehen somit nicht in die Bewertung ein. Zur Ermittlung der Punktzahl werden von den richtig angekreuzten Behauptungen die falsch angekreuzten Behauptungen abgezogen; ein negativer Übertrag in andere Teilaufgaben erfolgt nicht.

**Aufgabe 7** *MIPS-Assembler*

(12 Punkte)

1. Welche MIPS-Befehlsformate kennen Sie?

1 P.

2. Analysieren Sie den folgenden Auszug aus einem Assemblerprogramm. Kommentieren Sie jede Zeile. Geben Sie in jeder Zeile an, welches oder welche Register welche neuen Werte erhalten. Geben Sie bei den Befehlen `div` und `andi` die Rechnung in Klammern an.

3 P.

```
li    $t2, 0x40
ori   $t3, $zero, 125
div   $t3, $t2
mfhi  $t3
andi  $t3, $t3, 0x0E
```

3. Schreiben Sie die folgende in C-Code angegebene `while`-Schleife in MIPS-Assembler.

4 P.

```
while ( A[i] == k )
    i = i + j;
```

Dabei ist `A` ein Feld aus 32-Bit Integerzahlen. Die Anfangsadresse von `A` sei im Register `$s6` gespeichert. Die Variablen `i`, `j` und `k` stehen in den Registern `$s3`, `$s4` und `$s5`. Verwenden Sie die Register `$t0` und `$t1` zur Speicherung temporärer Variablen.

4. Gegeben sei der folgende C-Code:

2 P.

```
int x = 21; int y = 16; int *p; int *q;
p = &x;
*p = 17;
q = &y;
p = q;
y = 15;
*p = x;
```

Welche Werte haben `x` und `y` nach der Ausführung?

5. Übersetzen Sie den folgenden C-Code nach MIPS-Assembler:

2 P.

(a) `ptr = &i;`

(b) `i = *ptr;`

Dabei gilt: `int i, int *ptr;`

**Aufgabe 8** *Pipelining*

(12 Punkte)

Das folgende Programmstück soll in der DLX-Pipeline abgearbeitet werden.

```
S1:  addi  $t1, $t2, 2
S2:  sub   $t4, $t3, $t1
S3:  muli  $t3, $t5, 5
S4:  addi  $t3, $t3, 3
S5:  addi  $t2, $t1, $t5
```

1. Bestimmen Sie alle Datenabhängigkeiten innerhalb dieses Programmstücks. 3 P.
2. Zu Beginn des Programmstücks seien die Register folgendermaßen belegt: 3 P.

\$t1	\$t2	\$t3	\$t4	\$t5
3	5	7	9	2

Geben Sie die Registerbelegung nach Ablauf des Programmstücks an. Tragen sie hierzu in die Tabelle auf dem Lösungsblatt den Zustand der Pipeline und der Register für jeden Takt ein.

Wie viele Takte werden benötigt, um das Programm abzuarbeiten?

3. Wie wäre die Belegung der Register, wenn der Prozessor keine Pipeline besäße und die Befehle rein sequentiell abarbeitete? 1 P.
4. Die einzige Methode, die Pipelinekonflikte bei diesem Prozessor zu beheben, sei das Einfügen von NOP-Befehlen (*No Operation*) in den Befehlsstrom. 2 P.

Fügen Sie möglichst wenige NOP-Befehle in das Programmstück ein, so dass es zu keinen Konflikten mehr kommt, und das Ergebnis dem der sequentiellen Ausführung entspricht. Geben Sie das modifizierte Programmstück an.

Wie viele Takte werden nun benötigt?

5. Warum stellen bedingte Sprünge ein Problem für die Pipelineimplementierung dar? Geben Sie zwei Möglichkeiten zur Lösung dieses Problems an. 2 P.
6. Ist es möglich, einen Befehlssatz mit Befehlen variabler Länge in der DLX-Pipeline auszuführen? Begründen Sie Ihre Antwort. 1 P.

## Aufgabe 9 Cache

(10 Punkte)

1. Gegeben sei ein 2-fach-satzassoziativer Cache-Speicher (*2-way-set-associative cache*) mit einer Blockgröße von 16 Byte. Die Hauptspeicheradresse ist 32 Bit. Der Satzindex ist 12 Bit breit. Zur Verwaltung eines Cacheblocks werden zwei Statusbits (*Valid*-Bit und *Dirty*-Bit) verwendet.

(a) Wie breit ist der *Tag*?

1 P.

(b) Wie viele Befehle mit einer Länge von 32 Bit können sich gleichzeitig im Cache-Speicher befinden?

1 P.

(c) Bestimmen Sie den insgesamt erforderlichen Speicherbedarf für die Realisierung des Cache-Speichers?

2 P.

2. Gegeben sei ein direkt abgebildeter Cache-Speicher (*direct mapped cache*) mit einer Speicherkapazität von 128 Byte und einer Blockgröße von 16 Byte. Als Aktualisierungsstrategie wird das Rückschreib-Verfahren (*write back*) verwendet. Nehmen Sie an, dass der Cache-Speicher zu Beginn leer ist. Betrachten Sie die folgenden Lese- und Schreibzugriffe auf die in hexadezimaler Schreibweise angegebenen Adressen:

Adresse	54	22	D4	08	D0	6A	92	E0	D3
read/write	w	r	w	r	r	r	r	w	r
Index	5	2							
Tag	0	0							
Hit/Miss	Miss								
write back?	nein								

Vervollständigen Sie diese Tabelle im Lösungsblatt. Verwenden Sie dabei **Miss** für Cache-Miss und **Hit** für Cache-Hit. Geben Sie in der letzten Zeile der Tabelle an, ob der entsprechende Cacheblock in den Hauptspeicher zurückkopiert werden muss (**ja**) oder nicht (**nein**).

6 P.

## Aufgabe 10 *Verschiedenes*

(7 Punkte)

1. Was ist der entscheidende Nachteil von Befehlssätzen mit *OpCodes* variabler Länge? 1 P.
2. Erläutern Sie das *two cycle transfer*-Übertragungsverfahren beim direkten Speicherzugriff (DMA). 2 P.
3. Erläutern Sie den Unterschied zwischen *speicherbezogener (memory mapped I/O)* und *isolierter* Adressierung von Peripherie-Bausteinen. 2 P.
4. Skizzieren Sie den prinzipiellen Aufbau eines Systemschnittstellen-Bausteins. 2 P.