KIT-Fakultät für Informatik
Prof. Dr.-Ing. Uwe D. Hanebeck, Prof. Dr.-Ing. Jörg Henkel

Aufgabenblätter zur Prüfung

Digitaltechnik und Entwurfsverfahren (TI-1)

und

Rechnerorganisation (TI-2)

am 27. März 2021, 9:00 – 11:00 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 Endergebnisse und Rechenweg 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 so kurz und präzise wie möglich. Der auf den Lösungsblättern für eine Aufgabe vorgesehene Platz lässt nicht auf den Umfang einer korrekten Lösung schließen.
- 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)

Eine unvollständig definierte Schaltfunktion $y = f(d, c, b, a)$ ist gegeben durch die folgenden Gleichungen:

$$\begin{aligned} y &= \text{MINt}(0, 1, 2, 6, 8, 9, 10, 15) \\ y &= \text{MAXt}(4, 5, 11, 12, 13, 14) \end{aligned}$$

1. Tragen Sie die Schaltfunktion f in das KV-Diagramm im Lösungsblatt ein. Zeichnen Sie alle Prim-Einsblöcke klar und eindeutig ein. Geben Sie die zugehörigen Primimplikanten an. Unterstreichen Sie alle Kernprimimplikanten. **4 P.**

2. Geben Sie eine disjunktive Minimalform (DMF) von f an. **1 P.**

In Tabelle 1 ist die Überdeckungstabelle einer Schaltfunktion $z = g(x_n, \ldots, x_0)$ mit den Mintermen a, b, c, d, e sowie den Primimplikanten A, B, C, D gegeben.

Primimplikanten	Minterme				
	a	b	c	d	e
A	×			×	×
B		×			×
C			×		
D	×			×	

Tabelle 1: Überdeckungstabelle der Schaltfunktion $z = g(x_n, \ldots, x_0)$

3. Ist die Schaltfunktion z vollständig oder unvollständig definiert? Begründen Sie Ihre Antwort. **2 P.**

4. Welche Primimplikanten sind Kernprimimplikanten? **1 P.**

5. Geben Sie die Überdeckungsfunktion $\ddot{u}_g$ für die gegebene Überdeckungstabelle an. Formen Sie $\ddot{u}_g$ in eine disjunktive Form um. **3 P.**

Aufgabe 2 *CMOS, Spezielle Bausteine* (10 Punkte)

In Abbildung 1 ist eine Teilrealisierung einer Schaltfunktion $y = f(x_5, x_4, x_3, x_2, x_1, x_0)$ in der CMOS-Technologie dargestellt.

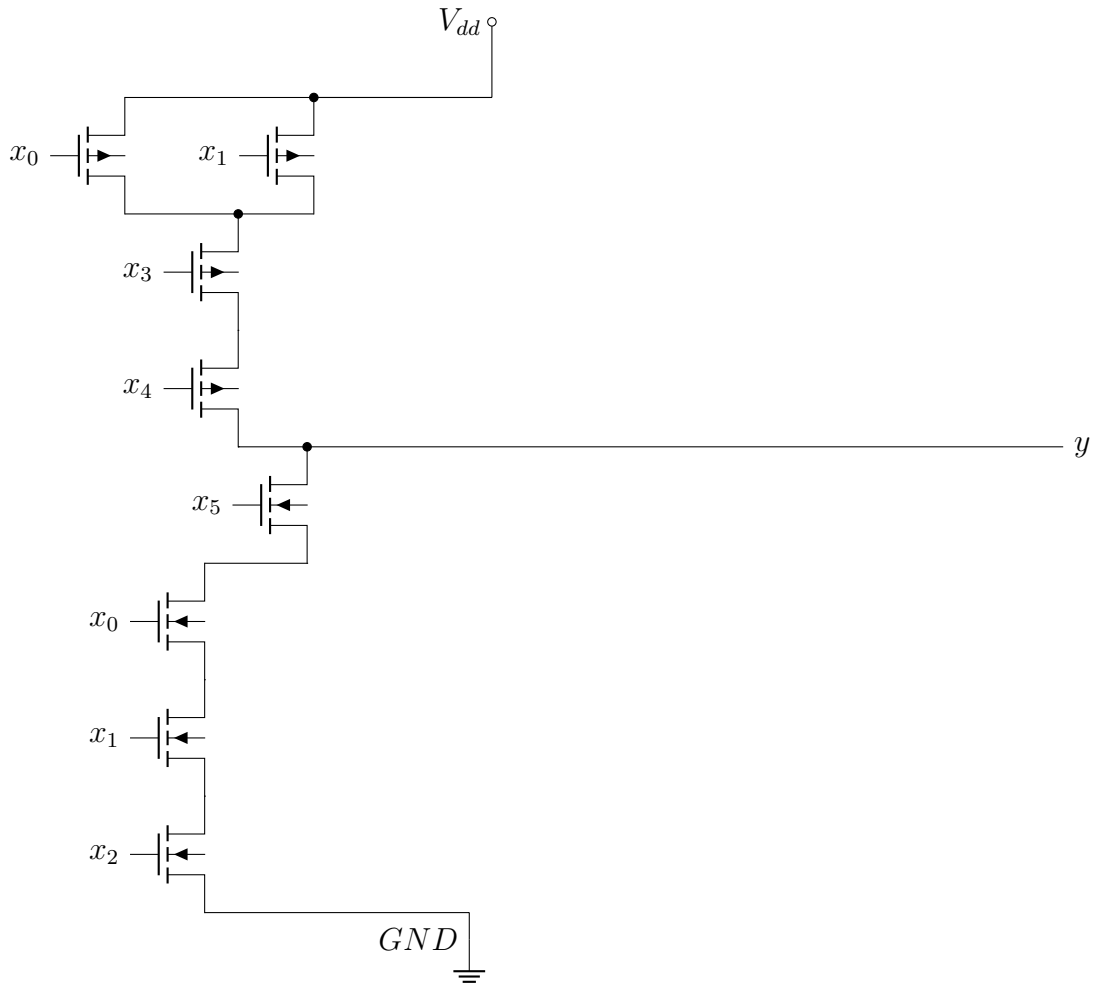

Abbildung 1: CMOS-Transistorschaltung

1. Ergänzen Sie die Schaltung im p- und n-Teil so, dass eine Realisierung der Schaltfunktion y in der CMOS-Technologie entsteht. 4 P.
2. Welche Schaltfunktion wurde realisiert realisiert? Geben Sie eine disjunktive Form an. 2 P.
3. Was ist der Unterschied zwischen einem Halbaddierer und einem Volladdierer? 1 P.
4. Zeichnen Sie das Schaltnetz eines 1-Bit-Volladdierers. Beschriften Sie die Eingänge und Ausgänge Ihrer Schaltung und geben Sie Ihre Bedeutung an. 3 P.

Aufgabe 3 *Laufzeiteffekte* (6 Punkte)

Gegeben sei die Schaltfunktion

$$y = h(d, c, b, a) = d\,c \;\vee\; d\,b \;\vee\; \overline{d}\,b\,a \;\vee c\,\overline{b}\,a$$

1. Untersuchen Sie die folgenden Übergänge auf Funktionshasards. 2 P.

   $$(0,0,0,0) \to (1,0,1,1) \qquad \text{und} \qquad (1,1,0,0) \to (0,1,1,1)$$

   Um welchen Hasardtyp handelt es sich, falls der entsprechende Übergang hasardbehaftet ist. Begründen Sie Ihre Antwort.

2. Geben Sie eine Realisierung der Funktion y an, die frei von allen statischen Strukturhasards ist. Begründen Sie Ihre Antwort. Zeichnen Sie das zugehörige Schaltnetz. 4 P.

Aufgabe 4 *Schaltwerke* (10 Punkte)

Es soll ein synchroner Vorwärts-Rückwärtszähler, der modulo 4 zählt, mit T-Flipflops entworfen werden. Der Zähler soll bei der Eingangsvariablen $X = 0$ vorwärts, bei $X = 1$ rückwärts zählen. Am Ausgang sollen die Zustände des Zählers angezeigt werden.

1. Erstellen Sie den Moore-Automatengraphen des Zählers mit einer möglichst geringen Anzahl von Zuständen. Wie viele Flipflops sind minimal notwendig? 2 P.

2. Die Zustände des Schaltwerks seien dual kodiert. Geben Sie die kodierte Ablauftabelle für eine Realisierung mit T-Flipflops an. Verwenden Sie hierzu die im Lösungsblatt vorbereitete Tabelle. 3 P.

3. Geben Sie die Ansteuerfunktionen der Flipflops in disjunktiver Normalform an. Vereinfachen Sie die booleschen Ausdrücke der Ansteuerfunktionen soweit wie möglich. 2 P.

4. Zur Realisierung des Schaltwerks stehen flankengesteuerte T-Flipflops, zwei Undgatter, ein Odergatter und ein Inverter zur Verfügung. Überführen Sie die Ansteuerfunktionen in eine geeignete Darstellungsform und zeichnen Sie die resultierende Schaltung des Schaltwerks. 3 P.

Aufgabe 5 *Rechnerarithmetik & Codes* (8 Punkte)

1. Geben Sie die Darstellung der Zahl 2021_{10} im [3 P.]

   - 32-Bit Zweierkomplement-Format
   - 32-Bit IEEE-754-Gleitkomma-Format.

   an.

2. Gegeben sind die beiden Codewörter (Hammingcode ohne Paritätsbit) [3 P.]

   - Codewort 1: **1 0 0 0 1 1 0 1 0 1 0**
   - Codewort 2: **0 1 1 0 0 1 0 1 0 1 1**

   Prüfen Sie beide Codewörter auf Ein-Bit-Fehler. Geben Sie die zugehörigen Datenwörter an.

3. Addieren Sie die beiden BCD-Zahlen: [1 P.]

$$\begin{array}{rrr} & 1001 & 0110 \\ + & 0111 & 0101 \\ \hline \end{array}$$

   Der Rechenweg soll klar ersichtlich dargestellt werden. Eine Konvertierung ins Dezimalsystem sei nicht gestattet.

4. Welche Vor- und Nachteile hat die BCD-Arithmetik gegenüber der Dual-Arithmetik? [1 P.]

Aufgabe 6 *Die Programmiersprache C* (9 Punkte)

1. Gegeben sei im Folgenden der jeweilige Codeausschnitt in C. Schreiben Sie auf das Lösungsblatt an entsprechender Stelle die Ausgabe der printf-Funktion.

   (a) Ausschnitt: 1 P.

   ```
   int a = 16 ^ 2;
   printf("C-Teil - 1: Ausgabe lautet %d \n", a);
   ```

   (b) Ausschnitt: 1 P.

   ```
   int b[] = {1,2,3,4,5};
   int* p1 = b+3;
   int c = *(p1-1);
   printf("C-Teil - 2: Ausgabe lautet %d \n", c);
   ```

   (c) Ausschnitt: 2 P.

   ```
   int d[] = {0,1,2,3,4,5,6,7,8,9,10};
   int e = 0;
   int *p2 = d;

   for(int i = 0; i < 10; ++i, p2++)
   {
       e += *(p2+(i%2));
   }

   printf("C-Teil - 3: Ausgabe lautet %d \n", e);
   ```

   (d) Ausschnitt: 3 P.

   ```
   char f[] = "MikroProzessor";

   for(int i = 1;  *(f+i) > 'P'; ++i)
   {
       (i >> 2) % 2 ? (*(f+i) += (char) 1) : (*(f+i) -= (char) 1);
   }

   printf("C-Teil - 4: Ausgabe lautet %s \n", f);
   ```

   (Hinweis: Im ASCII-Alphabet haben große Buchstaben den Wertebereich [65,90] und kleine Buchstaben den Wertebereich [97,122])

2. Was macht der folgende Codeausschnitt und zu was kann er dienen? 2 P.

   ```
   long i
   float z = 1234.0
   i = * (long *) &z
   ```

   (Hinweis: Denken Sie an Operatoren und auf welchen Datentypen diese definiert sind.)

Aufgabe 7 *MIPS-Assembler* (11 Punkte)

3 P.

1. Geben Sie für das folgende MIPS-Programmstück den Inhalt des Zielregisters in hexadezimaler Schreibweise nach der Ausführung des jeweiligen Befehls an.

```
addi   $s1, $zero, 0x4
sll    $s2, $s1, 4
slti   $s3, $s2, 100
lui    $s4, 0x40
xor    $s5, $s1, $s4
```

6 P.

2. Mit der Assemblerschreibweise <Mnemo><Offset>(Bi) wird der MIPS-Befehlssatz um einen weiteren Befehl „inc“ des I-Typs erweitert. Der neue Befehl inkrementiert den Speicherwert, welcher durch die Adressberechnung adressiert wird.
   Führen Sie den folgenden MIPS-Code aus und geben sie die Änderungen in den Register- und Speicherinhalten an. Verwenden Sie die im Lösungsblatt angegebenen Tabellen.

```
addi $t3, $t0, 0x12
lw   $t1, 0($t3)
inc  0x1E($t0)
and  $t4, $t1, $t0
sw   $t4, 0x06($t2)
```

Registersatz

Register	Inhalt
$t0	0x12
$t1	0x32
$t2	0x1E
$t3	0x00
$t4	0xAF

Hauptspeicher

Adresse	Inhalt
$0x20	0x10
$0x24	0x18
$0x28	0x12
$0x2C	0xCF
$0x30	0x66

3. Gegeben sei folgender MIPS-Code:

```
            .data
result:     .word 0xAB124545
            .text
            lbu $t1, result
            lb  $t2, result
```

2 P.

Wie lauten die Inhalte der Register $t1 und $t2, wenn man von einer

(a) Little-Endian

(b) Big-Endian

Byte-Anordnung ausgeht?

Aufgabe 8 *Pipelining* (9 Punkte)

1. Bestimmen Sie alle Datenabhängigkeiten im folgenden Programmstück. Geben Sie jeweils die Art der Abhängigkeit und das betroffene Register an. 5 P.

```
S1:      subi  $t1, $t0, 1
S2:      sll   $t2, $t1, 2
S3:      xor   $t3, $t1, $t2
S4:      srl   $t1, $t3, 4
S5:      add   $t4, $t2, $t3
S6:      lw $t4, 0x24($zero)
```

2. Das folgende Programmstück soll auf einem Prozessor mit einer DLX-Pipeline ohne Forwarding-Techniken ausgeführt werden. Fügen Sie in das Programmstück möglichst wenige `NOP`-Befehle ein, so dass keine Konflikte auftreten. Es ist nicht notwendig den Code abzuschreiben. Stattdessen können die jeweiligen Zeilenbezeichnungen (S1-S7) verwendet werden. 4 P.

```
S1:     anfang:  andi $t2, $t1, 1
S2:              beqz $t2, weiter
S3:              subi $t1, $t1, 1
S4:              j anfang
S5:     weiter:  srl  $t1, $t1, 1
S6:              j anfang
S7:              addi $t3, $t0,1
```

Aufgabe 9 *Cache- & Speicherverwaltung* (12 Punkte)

1. Bei einem Cache-Speicher mit einer Speicherkapazität von 1024 KByte ist die Hauptspeicheradresse in ein 16 Bit Tag-Feld, ein 12 Bit Index-Feld und ein 4 Bit Byte-Offset unterteilt ($K = 2^{10} = 1024$). Geben Sie bei der Beantwortung der folgenden Fragen den Lösungsweg an.

   (a) Bestimmen Sie die Blockgröße in Bytes. 1 P.

   (b) Wie viele Einträge besitzt der Cache-Speicher? 1 P.

   (c) Wie ist der Cache-Speicher organisiert? 2 P.

2. Gegeben sei ein direkt-abgebildeter Cache-Speicher (*direct mapped cache*) mit einer Speicherkapazität von 256 Byte und einer Blockgröße von 16 Byte. Als Aktualisierungsstrategie wird ein Durchschreibverfahren *(write through policy)* verwendet. Bei dieser Aktualisierungsstrategie wird ein CPU-Datum bei einem *write miss* nur in den Speicher geschrieben. Bei einem *write hit* wird ein CPU-Datum sowohl in den Cache als auch in den Speicher geschrieben. Betrachten Sie die folgenden Lese- und Schreibzugriffe auf die in hexadezimaler Schreibweise angegebenen Adressen:

Adresse	**0x000**	**0xA29**	**0xA39**	**0xC26**	**0xA34**	**0x021**	**0x041**	**0xB11**
read/write	r	r	r	w	r	r	r	w
Index								
Tag								
Hit/Miss								

   Vervollständigen Sie die obige Tabelle im Lösungsblatt. Verwenden Sie dabei **M** für Cache-Miss und **H** für Cache-Hit. 4 P.

3. Ein Rechnersystem enthält eine Speicherverwaltungseinheit (MMU) zur Umsetzung von virtuellen in physikalische Seitenadressen (Abbildung 2). Die MMU bildet einen virtuellen Adressraum von 2^V Bytes auf einen physikalischen Adressraum der Größe 2^M Bytes ab und benutzt dabei Seiten der Größe 2^P Bytes. Nehmen Sie an, dass die MMU eine Byte-Adressierung benutzt und die gesamte Seitentabelle im Hauptspeicher ist. In der folgenden Tabelle ist ein Ausschnitt aus der Seitentabelle gegeben. 4 P.

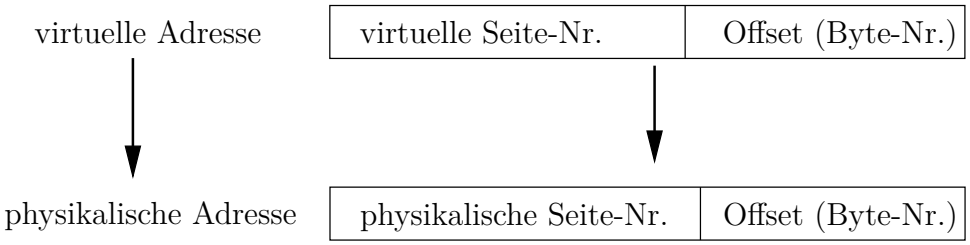

Abbildung 2: Format der virtuellen und physikalischen Adressen

Welchen physikalischen Adressen entsprechen die dezimalen virtuellen Adressen 1444 und 789, wenn $P = 8$ ist? Geben Sie den Lösungsweg an.

Virtuelle Seitennummer	Physikalische Seitennummer
0	8
1	3
2	1
3	2
4	9
5	7
6	6
7	4
⋮	⋮

Aufgabe 10 *Allgemeines* (4 Punkte)

1. Aus wie vielen Sichten und Entwurfsebenen besteht das Y-Diagramm von D.D.Gajski? 1 P.
2. Welche Komponente einer CPU übersetzt zur Laufzeit eines Programms logische in physikalische Adressen, damit ein Speicherzugriff erfolgen kann? 1 P.
3. Beantworten Sie die folgenden Fragen zu RISC-Prozessoren. 2 P.
   (a) Was macht die Dekodierschaltung in einem RISC-Prozessor einfach?
   (b) Was ist eine *Harvard*-Architektur?