Vorname:_____ **Name:**_____

Matr.-Nr.:_____ **Note:**

12.03.2004
10^{00} – 12^{00} Uhr

UNIVERSITÄT KARLSRUHE
Institut für Industrielle Informationstechnik
- Prof. Dr.-Ing. habil. K. Dostert -

Vordiplomprüfung im Fach

Mikrorechnertechnik

Die Prüfung umfasst **10 Aufgaben**.

Bitte schreiben Sie auf dieses Deckblatt sowie auf alle zusätzlichen Lösungsblätter Ihren Namen und Ihre Matrikelnummer.

Bei den jeweiligen Aufgaben finden Sie Platz, um Ihre Rechnung und die Lösung einzutragen. Sollte dieser Platz nicht ausreichen, kann zusätzliches Schreibpapier bei der Aufsicht angefordert werden. **Die Verwendung eigenen Papiers ist nicht erlaubt.**

Separat zu diesem Aufgabensatz finden Sie einige Hilfsblätter, deren Inhalt Ihnen eine Gedächtnisstütze bei der Lösung der Aufgaben bietet.

Als Hilfsmittel sind Schreib- und Zeichenzeug sowie Taschenrechner mit zu Beginn der Klausur **gelöschtem** Speicher zugelassen.

Aufgabe:	1	2	3	4	5	6	7	8	9	10	gesamt
Punkte:											
erreichbare Punktzahl:	12	8	10	8	10	12	10	10	10	10	100

Aufgabe 1: **A/D- und D/A-Wandlung** **(12 Punkte)**

Zunächst wird ein A/D-Wandler, der nach dem Zählverfahren arbeitet, betrachtet. Bild 1.1 zeigt das unvollständige Blockschaltbild eines solchen Wandlers.

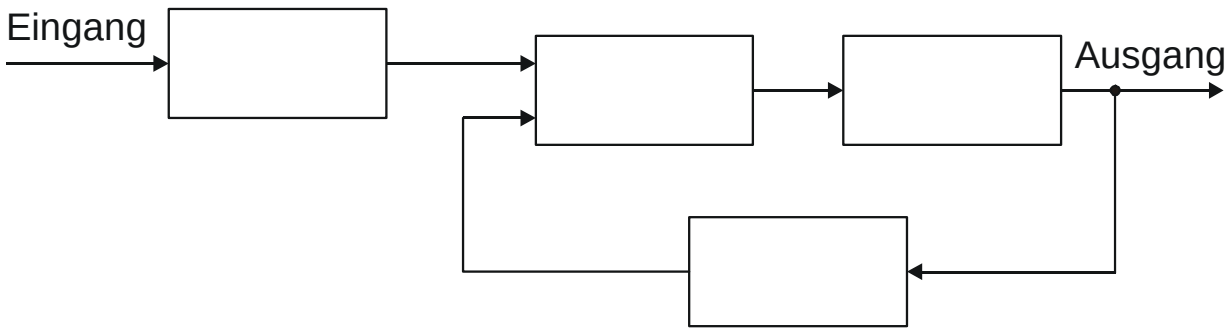

Bild 1.1: A/D-Wandler nach dem Zählverfahren

a) Vervollständigen Sie das Blockschaltbild in Bild 1.1, indem Sie in die Kästchen die Bezeichnungen der entsprechenden dort benötigten Funktionen eintragen!

b) Wie viele Taktschritte sind für einen Wandelvorgang bei einer Auflösung von *N* bit notwendig?

Nun wird ein A/D-Wandler betrachtet, der nach dem Verfahren der sukzessiven Approximation arbeitet.

c) Welche beiden prinzipiellen Änderungen sind notwendig, um den nach der Zählmethode arbeitenden Wandler in einen Wandler mit sukzessiver Approximation zu überführen?

d) Wie viele Taktschritte sind bei der sukzessiven Approximation für einen Wandelvorgang bei einer Auflösung von *N* bit notwendig?

Nun wird ein R/2R-Wandler zur D/A-Wandlung betrachtet, der aus einem Widerstandsnetzwerk aufgebaut ist. Bild 1.2 zeigt das Prinzip eines solchen Wandlers mit einer Auflösung von 4 bit. Die dargestellte Schalterstellung entspricht dem Digitalwert „1011“.

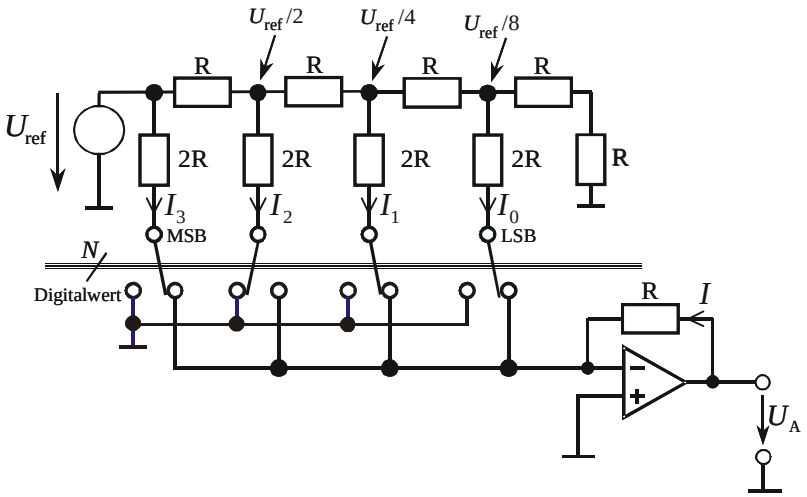

Bild 1.2: D/A-Wandler nach dem *R*/2*R*-Prinzip

e) Geben Sie die Ströme I_3, I_2, I_1 und I_0 in Abhängigkeit von R und U_{ref} an!

f) Geben Sie den Strom I an, der durch den Gegenkoppelwiderstand R des Operationsverstärkers fließt!

g) Geben Sie die Ausgangsspannung U_A des Operationsverstärkers in Abhängigkeit von U_{ref} an!

Aufgabe 2: **Zahlendarstellung in Mikrorechnerprogrammen (8 Punkte)**

Zur Zahlendarstellung in Mikrorechnerprogrammen finden verschiedene Zahlenformate Verwendung. Zunächst wird ein 8 bit-Mikrocontroller betrachtet, der die Zweierkomplementdarstellung verwendet.

In Tabelle 2.1 ist eine Anfangsbelegung der Register R0 und R1 in Dezimaldarstellung angegeben.

Tabelle 2.1: Registerbelegung eines Mikrocontrollers

Register	**Inhalt (dezimal)**	**Inhalt (binär)** MSB LSB
R0	–6	
R1	19	
A		

a) Tragen Sie in Tabelle 2.1 die Inhalte der Register R0 und R1 in binärer 8 bit-Zweierkomplementdarstellung ein! Die Abkürzungen MSB und LSB stehen für das höchstwertige (most significant) und das niederwertigste (least significant) Bit.

Der Mikrocontroller führt nun eine Multiplikation der Werte aus den Registern R0 und R1 durch, wonach das 8 bit-Ergebnis im Akkumulator A steht.

b) Tragen Sie in Tabelle 2.1 das Multiplikationsergebnis in Dezimal- und Zweierkomplement-Binärdarstellung ein!

c) Geben Sie den Dezimalwert des Registerinhalts von R0 an, wenn dieser als Fraktalzahl interpretiert wird!

Nun wird ein digitaler Signalprozessor betrachtet, der das Gleitkommaformat nach IEEE-P754 mit einfacher Genauigkeit verwendet. Dabei dient 1 bit für das Vorzeichen, 8 bit für den Exponenten und 23 bit für die Mantisse.

d) Geben Sie in Tabelle 2.2 die Inhalte der 32 bit-Register R0 und R1 des digitalen Signalprozessors an!

Tabelle 2.2: Registerbelegung eines digitalen Signalprozessors

Register	**Inhalt (dezimal)**	**Inhalt (binär)** MSB LSB
R0	–6	
R1	19	

Aufgabe 3: **Verlustleistung von CMOS-Schaltungen** **(10 Punkte)**

Die Verlustleistung in CMOS-Mikrorechnersystemen setzt sich im Wesentlichen aus zwei Komponenten zusammen: den Umschaltverlusten sowie den Umladeverlusten.

a) Wodurch werden diese Verluste jeweils verursacht?

Zunächst werden nur Umschaltverluste betrachtet. Dabei wird ein Mikrocontrollermodell angenommen, bei dem im Mittel bei jeder Taktflanke 20.000 Inverter gleichzeitig schalten. Der Mikrocontroller werde mit 10 MHz getaktet, wobei die Flankensteilheit des Taktes, d. h. jeweils Anstiegs- und Abfallzeit, 1 ns betrage. Die Strom-Zeit-Fläche bei einem Schaltvorgang eines Inverters sei ein gleichschenkliges Dreieck. Bei 3,3 V Versorgungsspannung beträgt der mittlere durch die Umschaltverluste verursachte Gesamtstrom 40 mA.

b) Wie hoch ist die Stromspitze I_{DP} in der Strom-Zeit-Fläche bei einem Schaltvorgang eines Inverters?

c) Wie groß ist der maximale durch Umschaltverluste verursachte Strom auf der Versorgungsleitung und die maximale durch Umschaltverluste verursachte Verlustleistung?

d) Nennen Sie eine schaltungstechnische Maßnahme, um die negativen Auswirkungen der hohen Spitzenströme auf die Stabilität der Versorgungsspannung zu verringern!

e) Wie ändert sich die Höhe einer einzelnen durch Umschaltverluste verursachten Stromspitze und die mittlere durch Umschaltverluste verursachte Stromaufnahme, wenn man die Taktfrequenz auf 20 MHz erhöht?

Nun werden die Umladeverluste des Mikrocontrollers ebenfalls bei 3,3 V Betriebsspannung betrachtet. Diese Verluste werden durch eine äquivalente Frequenz $f^* = 10\,\text{MHz}$ und eine äquivalente Kapazität C^* = 6 nF modelliert.

f) Wie groß sind die Umladeverluste des Mikrocontrollers?

Aufgabe 4: **CMOS-Transfergates** **(8 Punkte)**

Bild 4.1 zeigt das Schaltsymbol eines Transfergates mit Ein- und Ausgang und dem Takteingang T sowie dem invertierten Takteingang $\overline{T}$.

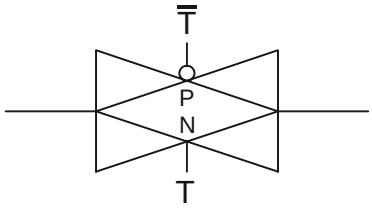

Bild 4.1: Transfergate

a) Skizzieren Sie den Aufbau eines Transfergates in CMOS-Technologie, bestehend aus einem n-Kanal- und einem p-Kanal-MOS-FET!

b) Geben sie an, welche Pegel an T und $\overline{T}$ anliegen müssen, damit das Transfergate leitet bzw. sperrt!

Bild 4.2 zeigt eine Flip-Flop-Schaltung aus Transfergates. Die beiden Treiber (mit „1“ in einem Rechteck symbolisiert) sind Signalverstärker und dienen dem Aufrechterhalten der entsprechenden logischen Zustände.

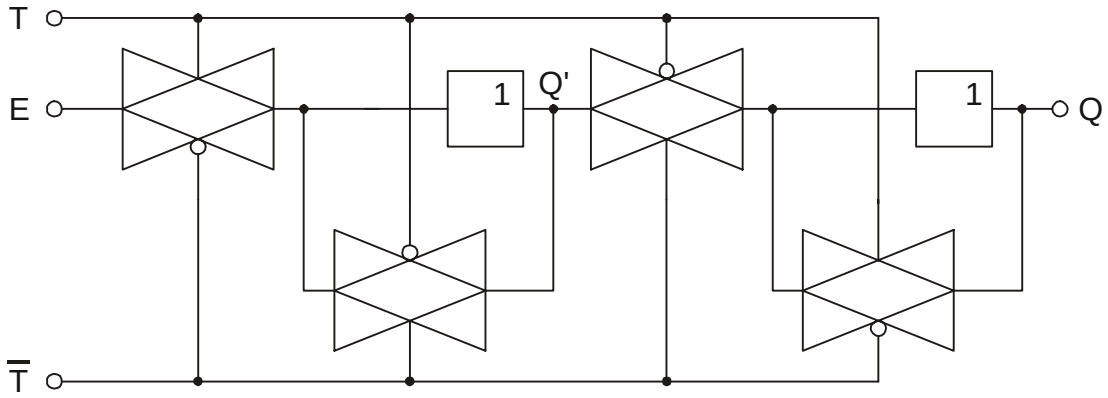

Bild 4.2: Flip-Flop-Schaltung aus Transfergates

An den Eingang E der Schaltung nach Bild 4.2 werde ein Eingangssignal nach Bild 4.3 angelegt.

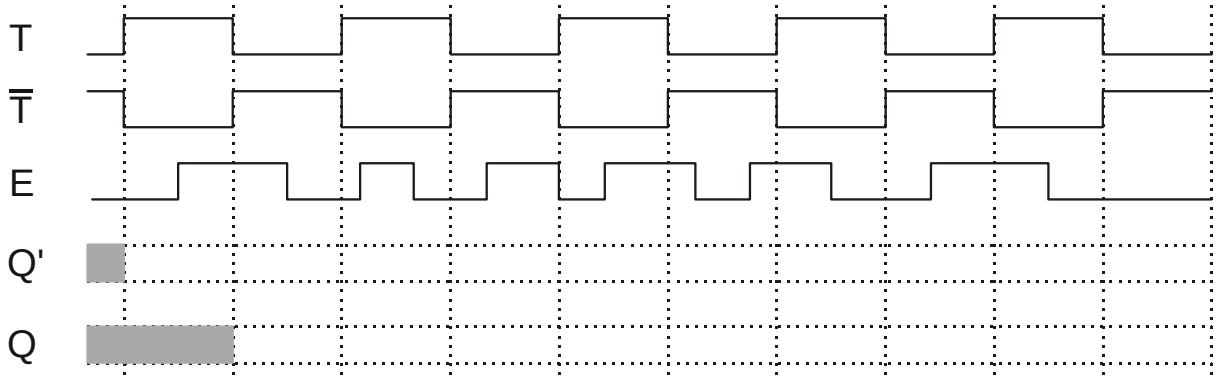

Bild 4.3: Signalverläufe in der Schaltung aus Bild 4.2

c) Tragen Sie in Bild 4.3 die Verläufe des Zwischensignals Q’ sowie des Ausgangssignals Q der Schaltung aus Bild 4.2 ein! An den grau hinterlegten Stellen sind die Signale undefiniert.

Aufgabe 5: **Multiplizierer** **(10 Punkte)**

Das Produkt zweier vorzeichenbehafteter 6 bit-Zweierkomplementzahlen *A* und *B* mit

$$A = -a_5 \cdot 2^5 + \sum_{i=0}^{4} a_i 2^i \quad \text{und} \quad B = -b_5 \cdot 2^5 + \sum_{i=0}^{4} b_i 2^i \tag{5.1}$$

lässt sich in der Form

$$A \cdot B = -2^{11} + 2^6 + a_5 b_5 \cdot 2^{10} + \sum_{i=0}^{4}\sum_{j=0}^{4} a_i b_j 2^{i+j} + \sum_{i=0}^{4} \overline{a_5 b_i} 2^{i+5} + \sum_{i=0}^{4} \overline{b_5 a_i} 2^{i+5} \tag{5.2}$$

darstellen.

a) Wie viele UND-Gatter werden für die Realisierung der Binärprodukte in der Doppelsumme in (5.2) benötigt?

b) Wie viele NAND-Gatter werden für die Realisierung der Binärprodukte in den gemischten Termen in (5.2) benötigt?

c) Leiten Sie aus (5.2) ein Rechenschema zur Multiplikation ab, indem Sie die einzelnen Binärprodukte $a_i b_j$ bzw. deren Negation sowie die Konstanten in Tabelle 5.1 entsprechend ihrer Wertigkeit eintragen!

Tabelle 5.1: Rechenschema zur Multiplikation zweier 6 bit-Zahlen

-2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Um das Ergebnis der Multiplikation zu bestimmen, müssen die einzelnen Zeilen aus Tabelle 5.1 addiert werden. Um eine langsame zeilenweise Addition zu vermeiden, wird die Tabelle schrittweise so weit wie möglich mit Halb- und Volladdierern bedeckt, bis nur noch zwei Zeilen übrig bleiben. Die Schlussaddition erfolgt dann mit einem Carry-Look-Ahead-Addierer. Dieses Rechenschema wird als Wallace-Tree bezeichnet.

d) Reduzieren Sie die sich aus Tabelle 5.1 ergebende Matrix schrittweise! Tragen Sie die Ergebnisse Ihrer Reduktionsschritte in Tabelle 5.2 ein! Gehen Sie dazu vom Schema aus Tabelle 5.1 aus und kennzeichnen Sie durch Umrahmen der jeweiligen Tabelleneinträge, an welchen Stellen Sie Halb- oder Volladdierer einsetzen! Markieren Sie verbleibende Einträge jeweils durch ein Kreuz!

Tabelle 5.2: Schrittweise Abarbeitung der Multiplikation mit Halb- und Volladdierern

Schritt 1

Schritt 2

Schritt 3

Abschlussaddition

Aufgabe 6: **Beschreibung einer FSM** **(12 Punkte)**

Das Steuerwerk stellt ein wichtiges Kernstück eines jeden Mikrorechners dar. Im Folgenden wird ein Ausschnitt aus einer Mikrosequencer-Steuerung als Teil des Steuerwerks in Form eines endlichen Automaten (FSM) betrachtet. Bild 6.1 zeigt das Blockschaltbild einer solchen FSM. Die FSM bildet den Befehlscode (OP-Code) auf eine Folge von Zuständen (A) ab. Außerdem sind Eingänge für den Takt, für eine Bedingung (COND) sowie zum Rücksetzen (RESET) vorhanden.

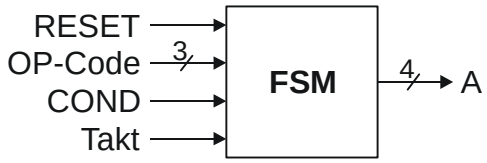

Bild 6.1: Blockschaltbild einer FSM als Bestandteil eines Mikrosequencers

Es werden neun Zustände betrachtet. Tabelle 6.1 zeigt die Zuordnung der Zustände zu den Befehlen, wobei für einen Befehl, je nach Komplexität, bis zu vier Zustände benötigt werden.

Tabelle 6.1: Zuordnung der Zustände der FSM zu den Befehlen

Zustand	Befehl	Beschreibung
s1	RES	Reset
s2 s3	BRA_1 BRA_2	Verzweigung
s4	JMP	Sprung
s5 s6 s7 s8	JMS_1 JMS_2 JMS_3 JMS_4	Sprung in Unterprogramm
s9	–	Ruhezustand

In Tabelle 6.2 ist ein Ausschnitt aus der Flusstabelle zur Beschreibung der FSM angegeben. Dabei sind die Zustände mit s1 … s9, die Folgezustände mit f1 … f9 bezeichnet. Der Eingangsvektor x = (OP3, OP2, OP1, COND) enthält 3 bit für den Befehlscode und ein Bedingungsbit. Das RESET-Bit wurde der Übersichtlichkeit halber weggelassen. Der Eingangsvektor bestimmt, ausgehend vom aktuellen Zustand s1 … s9, jeweils den Folgezustand f1 … f9. Irrelevante Bedingungen sind durch „–“ gekennzeichnet.

a) Vervollständigen Sie den Zustandsgraphen in Bild 6.2, indem Sie in den Knoten die Zustände und Befehle gemäß Tabelle 6.1 eintragen!

b) Vervollständigen Sie den Zustandsgraphen in Bild 6.2, indem Sie die fehlenden Zustandsübergänge als Kanten mit den zugehörigen Zustandsübergangsbedingungen gemäß Tabelle 6.2 eintragen!

Tabelle 6.2: Flusstabelle zur Beschreibung der FSM

relevant =		OP3	OP2	OP1	COND;	
s9	, x	0	0	0	-	,f1;
s1	, x	-	-	-	-	,f9;
s9	, x	0	1	0	-	,f2;
s9	, x	0	1	1	1	,f2;
s2	, x	-	-	-	-	,f3;
s3	, x	-	-	-	-	,f9;
s9	, x	1	0	0	-	,f4;
s4	, x	-	-	-	-	,f9;
s9	, x	1	0	1	1	,f4;
s9	, x	1	1	0	-	,f5;
s5	, x	-	-	-	-	,f6;
s6	, x	-	-	-	-	,f7;
s7	, x	-	-	-	-	,f8;
s8	, x	-	-	-	-	,f9;

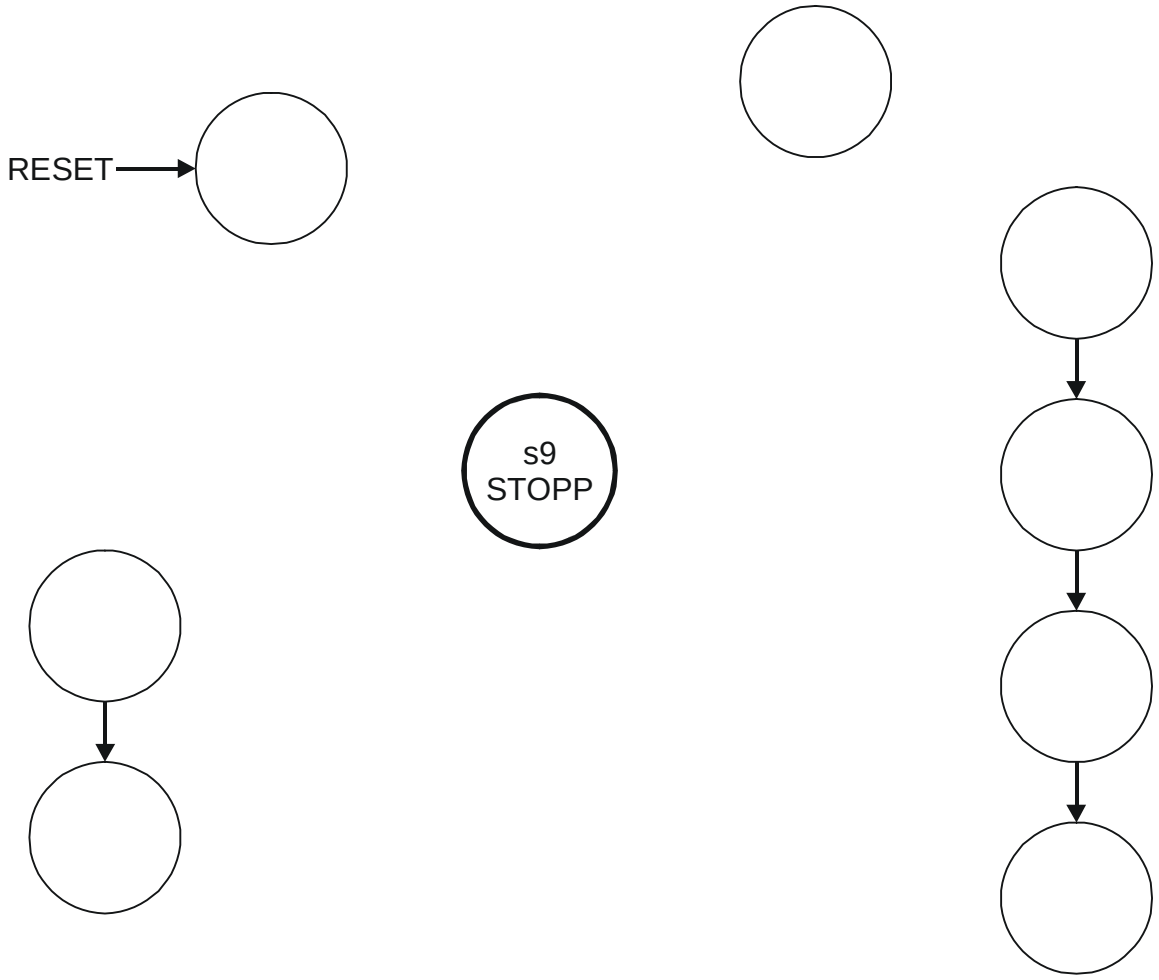

Bild 6.2: Zustandsgraph zur Beschreibung der FSM

In Tabelle 6.3 sind die Befehlscodes einiger Befehle angegeben, die mit der FSM realisiert werden können.

c) Ordnen Sie den Befehlscodes in Tabelle 6.3 die Befehle „Reset“, „Sprung in Unterprogramm“, „bedingter Sprung“ und „unbedingte Verzweigung“ zu!

Tabelle 6.3: Zuordnung von Befehlen zu Befehlscode

Befehl	OPC3	OPC2	OPC1
	0	1	0
	1	1	0
	0	0	0
	1	0	1

Aufgabe 7: **A/D-Wandlung mit dem Mikrocontroller ADuC832 (10 Punkte)**

Mit dem integrierten A/D-Wandler des Mikrocontrollers ADuC832 soll eine Spannung mit 12 bit digitalisiert und der Spannungswert über die serielle Schnittstelle an einen PC übergeben werden. Zur Lösung dieser Aufgabe sollen im Folgenden Teilaufgaben bearbeitet werden. Die benötigten 8051-Assemblerbefehle sind in der Hilfsblattsammlung aufgelistet.

Zur Umwandlung einer BCD-Zahl in ein ASCII-Zeichen wird eine Routine benötigt, die den Wert 48 addiert.

a) Schreiben Sie ein Unterprogramm `Add48`, das zum Inhalt des Registers R0 den Wert 48 (dezimal) addiert! Verwenden Sie zur Berechnung den Akkumulator A.

```
Add48:




ret
```

Zur Umrechnung des gewandelten Wertes wird unter anderem ein Unterprogramm benötigt, das eine Division durch 2 durchführt.

b) Schreiben Sie ein Unterprogramm `Divider`, das eine 16 bit-Zahl, die in den Registern R1 und R0 steht (höherwertiger Anteil in R1), durch 2 dividiert!

Hinweise:

- *Beachten Sie, dass eine Division durch 2 durch eine Schiebeoperation realisiert werden kann.*
- *Verwenden Sie für die Berechnungen den Akkumulator A.*
- *Das Ergebnis soll am Ende wieder in den Registern R1 und R0 stehen.*
- *Gehen Sie davon aus, dass das Carry-Bit zu Beginn des Unterprogramms gelöscht ist.*

```
Divider:




ret
```

Zum Senden eines Zeichens über die serielle Schnittstelle an den PC wird ein Unterprogramm benötigt.

c) Schreiben Sie ein Unterprogramm `Senden`, das so lange wartet, bis das Interruptflag TI gesetzt ist, dieses anschließend löscht und den Akkuinhalt in das Senderegister SBUF schreibt!

```
Senden:




ret
```

Aufgabe 8: **Programmierung der seriellen Schnittstelle im 8051 (10 Punkte)**

Mit einem Mikrocontroller des Typs 8051 soll eine serielle Schnittstelle betrieben werden, die sowohl Sende- als auch Empfangsbetrieb zulässt. Dazu wird die Betriebsart 1 der seriellen Schnittstelle verwendet, die eine asynchrone Datenübertragung mit variabler Baudrate ermöglicht, wobei als Zeitbasis die Überlaufrate von Timer 1 dient. Der Prozessor wird mit 10 MHz getaktet. Um die CPU-Belastung möglichst gering zu halten, wird der Timer 1 als 8 bit-Zeitgeber im Autoreload-Modus betrieben. Es sollen keine Interrupts von Timer 1 zugelassen werden. Das Steuerbit SMOD im Register PCON habe den Wert 1.

Hinweis: *In der beiliegenden Hilfsblattsammlung finden Sie eine Übersicht über die Funktionsweise des Timers und der Schnittstelle sowie eine Kurzbeschreibung der verwendeten Spezialfunktionsregister.*

a) Geben Sie die notwendigen Einstellungen der Spezialfunktionsregister SCON, TCON, TMOD und IE in Tabelle 8.1 an! Kennzeichnen Sie irrelevante Bits durch ‚X’!

Tabelle 8.1: Belegung der Spezialfunktionsregister

	Bit 7							Bit 0
SCON								
TCON								
TMOD								
IE								

b) Berechnen Sie den Autoreload-Wert für Timer 1, wenn die Baudrate 600 erzeugt werden soll!

c) Berechnen Sie die tatsächliche Baudrate und den daraus resultierenden relativen Fehler, wenn der Mikrocontroller mit 10 MHz getaktet ist!

d) Geben Sie die minimal mögliche Baudrate an, die in dieser Betriebsart mit SMOD = 1 bei 10 MHz Taktfrequenz möglich ist!

Aufgabe 9: **Digitale Signalprozessoren** **(10 Punkte)**

Der DSP 56000 zeichnet sich unter anderem durch vielseitige Möglichkeiten der Registeradressierung sowie durch die MAC-Operation aus.

Tabelle 9.1 gibt einen Auszug aus der Register- und Speicherbelegung des DSPs an. Das „$“-Zeichen kennzeichnet dabei Hexadezimalzahlen. Ausgehend von dieser Speicherbelegung wird folgender Assemblerbefehl ausgeführt:

```
MOVE  X:(R1)+N1,X0  Y:R6,Y0        ; Befehl 1
```

Anschließend wird ein weiterer Assemblerbefehl betrachtet, wobei vor der Befehlsausführung wieder die Speicherbelegung aus Tabelle 9.1 gilt:

```
MOVE  X:(R1)+,X0    Y:(R6+N6),Y0   ; Befehl 2
```

Tabelle 9.1: Anfangsbelegung der Register und Speicher

R1:	$ 0002	A2:	$ 00		…	…
N1:	$ 0002	A1:	$ 400000	$ 0004	$ 500000	$ E00000
M1:	$ 0003	A0:	$ 000000	$ 0003	$ 400000	$ D00000
R6:	$ 0001			$ 0002	$ 300000	$ C00000
N6:	$ 0001	X0:	$ 200000	$ 0001	$ 200000	$ B00000
M6:	$ FFFF	Y0:	$ EC0000	$ 0000	$ 100000	$ A00000
				Adresse	X-Speicher	Y-Speicher

a) Tragen Sie in Tabelle 9.2 die Registerbelegung nach Ausführung der Befehle 1 und 2 ein!

Tabelle 9.2: Registerbelegung nach Ausführung der Befehle 1 und 2

Register	Inhalt nach Befehl 1	Inhalt nach Befehl 2
R1	$	$
X0	$	$
R6	$	$
Y0	$	$

Nun wird die folgende MAC-Operation betrachtet:

```
MAC X0,Y0,A
```

Die Register- und Speicherbelegung vor Ausführung dieser Operation entspreche wieder der Anfangsbelegung aus Tabelle 9.1.

b) Geben Sie das in A stehende Ergebnis nach Ausführung der MAC-Operation an! Beachten Sie, dass der DSP 56000 mit Fraktalzahlendarstellung arbeitet und geben Sie die Zwischenschritte Ihrer Lösung an!

A2	A1	A0
$	$	$

Aufgabe 10: **Schaltungsbeschreibung mit VHDL** **(10 Punkte)**

Bild 10.1 zeigt das FSM-Modell eines Mooreautomaten. Die Eingänge des Automaten werden im Zustandsgraphen durch Kanten, die Ausgänge durch Knoten repräsentiert.

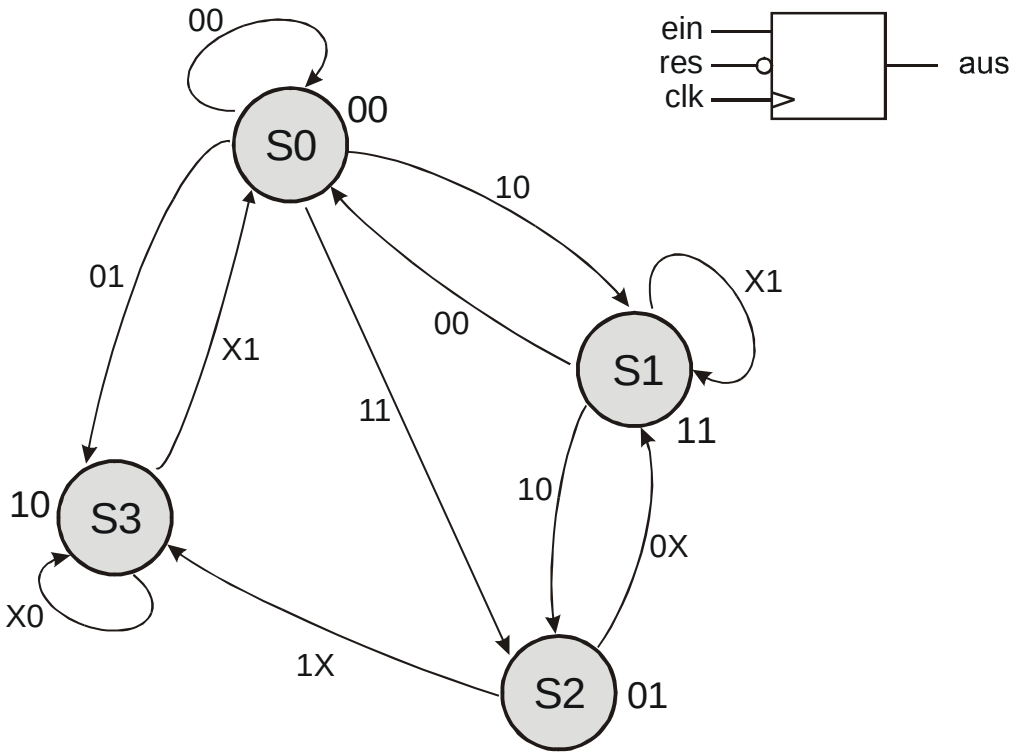

Bild 10.1: FSM-Modell eines Mooreautomaten

a) Vervollständigen Sie in der nachstehenden ENTITY `Automat` der FSM aus Bild 10.1 die Port-Deklaration! Verwenden Sie dazu die Datentypen `std_logic` bzw. `std_logic_vector`!

```
LIBRARY IEEE;
USE ieee.std_logic_vector_1164.all;
ENTITY Automat is
  PORT (




        );
END Automat;
```

b) Vervollständigen Sie im nachstehenden Prozess `FSM`, der den Folgezustand in Abhängigkeit vom aktuellen Zustand `akt_zust` und dem Eingangsvektor bestimmt, die fett gedruckten Zeilen!

```
FSM: PROCESS (akt_zust, ein)
BEGIN
  case akt_zust IS
    WHEN S0 =>
      CASE ein IS
        WHEN "    "  => folg_zust <=       ;
        WHEN "    "  => folg_zust <=       ;
        WHEN "    "  => folg_zust <=       ;
        WHEN "    "  => folg_zust <=       ;
        WHEN others  => folg_zust <= null ;
      END CASE;
    WHEN S1 =>
      CASE ein IS
        WHEN "    "  => folg_zust <=       ;
        WHEN "    "  => folg_zust <=       ;
        WHEN others  => folg_zust <=       ;
      END CASE;
    WHEN S2 =>
      CASE ein IS
        WHEN "    "  => folg_zust <=       ;
        WHEN "    "  => folg_zust <=       ;
        WHEN others  => folg_zust <=       ;
      END CASE;
    WHEN S3 =>
      CASE ein IS
        WHEN "    "  => folg_zust <=       ;
        WHEN "    "  => folg_zust <=       ;
        WHEN others  => folg_zust <=       ;
      END CASE;
    WHEN others => folg_zust <= S0;
  END CASE;
END PROCESS FSM;
```

c) Vervollständigen Sie im nachstehenden Prozess `Ausgabe`, der den Ausgangsvektor in Abhängigkeit vom aktuellen Zustand `akt_zust` bestimmt, die fett gedruckten Zeilen!

```
AUSGABE: PROCESS (akt_zust)
BEGIN
  CASE akt_zust is
    WHEN        => aus <= "    ";

    WHEN        => aus <= "    ";

    WHEN        => aus <= "    ";

    WHEN        => aus <= "    ";

    WHEN others => aus <= "00";
  END CASE;
END PROCESS AUSGABE;
```