Vorname:_____ **Name:**_____

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

15.09.2003
14^{00} – 16^{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 das Verfahren der sukzessiven Approximation verwendet, betrachtet. Bild 1.1 zeigt das Blockschaltbild eines solchen Wandlers.

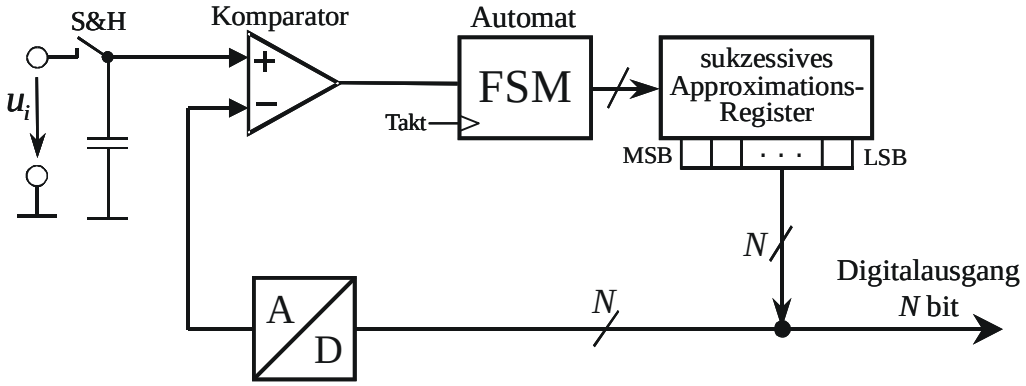

Bild 1.1: A/D-Wandler mit sukzessiver Approximation

Bei diesem Verfahren wird zunächst das höchstwertige Bit (MSB) im sukzessiven Approximationsregister (SAR) auf 1 gesetzt, die restlichen Bits sind 0. Der Wert des SAR wird auf einen D/A-Wandler gegeben und dessen Ausgangssignal wird mit dem Eingangswert u_i verglichen. Ist die Eingangsspannung höher als der D/A-gewandelte SAR-Wert, dann bleibt das gesetzte Bit auf 1, ansonsten wird es auf 0 gesetzt.
Anschließend wird das nächstniederwertige Bit gesetzt und auf die gleiche Weise entschieden, ob es gesetzt bleibt oder nicht. Das Verfahren wird bis zum niederwertigsten Bit (LSB) fortgesetzt.

Im Folgenden wird ein 3 bit-A/D-Wandler mit einem Eingangsbereich von 0 V … 1 V betrachtet, d. h. $u_{i,max}$ = 1 V. Tabelle 1.1 gibt die Ausgangsspannung des D/A-Wandlers aus Bild 1.1 in Abhängigkeit vom Inhalt des SAR an.

Tabelle 1.1: Ausgangsspannung des D/A-Wandlers in Abhängigkeit vom digitalen Eingangswert

Digitaler Eingangswert	Ausgangsspannung in Volt
000	0
001	0,125
010	0,25
011	0,375
100	0,5
101	0,625
110	0,75
111	0,875

a) Tragen Sie in Tabelle 1.2 den Inhalt des SAR ein, der sich nach ein, zwei und drei Wandelschritten für die angegebenen Eingangsspannungen ergibt!

Tabelle 1.2: Inhalt des SAR bei verschiedenen Eingangsspannungen

u_i in Volt	0,1		0,45		0,85	
Inhalt des SAR:	MSB	LSB	MSB	LSB	MSB	LSB
nach 1. Schritt						
nach 2. Schritt						
nach 3. Schritt						

Nun soll ein Analogsignal mit einer Auflösung von 10 bit und einer Wandelrate von 500 kHz in ein Digitalsignal gewandelt werden.

b) Geben Sie die Frequenz an, mit der die Wandelschritte ablaufen müssen, wenn der A/D-Wandler das Verfahren der sukzessiven Approximation verwendet!

c) Geben Sie die Taktfrequenz für den Zähler an, wenn ein A/D-Wandler nach dem Zählverfahren vorliegt!

Die Pulsweitenmodulation ermöglicht eine einfache D/A-Wandlung mit Mikrocontrollern, indem ein Rechtecksignal mit konstanter Periode und variablem Tastverhältnis t_i/T an einem Portpin ausgegeben wird. In Bild 1.2 ist das Ausgangssignal an einem Portpin für verschiedene Tastverhältnisse dargestellt. Das Rechtecksignal wird anschließend durch ein RC-Glied (Tiefpassfilter) geglättet.

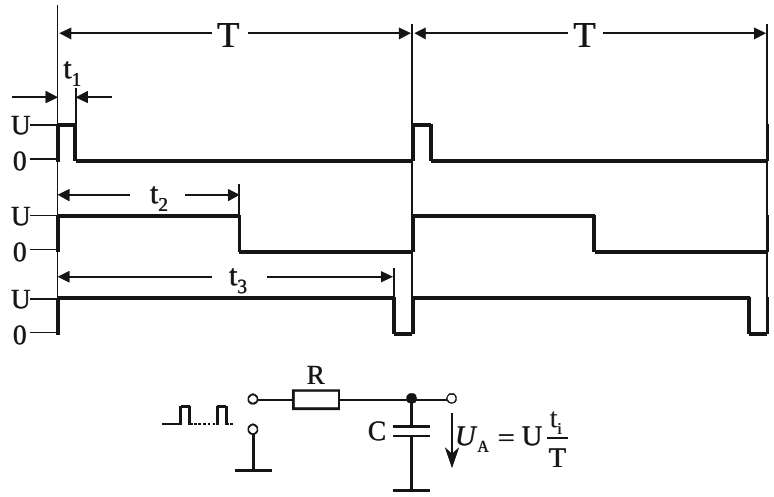

Bild 1.2: Prinzip der Pulsweitenmodulation

Zur Generierung des Rechtecksignals wird ein 8 bit-Zähler verwendet, der periodisch von 0 bis 255 durchläuft. Das Tastverhältnis wird mit einem Vergleichsregister festgelegt. Bei einem Überlauf des Zählers wird das Ausgangssignal auf U gesetzt. Sobald der Zählerstand mit dem Inhalt des Vergleichsregisters übereinstimmt, wird das Ausgangssignal auf 0 V zurückgesetzt. Die Spannung U betrage 5 V.

d) Berechnen Sie den Wert des Vergleichsregisters in Dezimaldarstellung, der nötig ist, um am Ausgang des Tiefpassfilters eine Spannung von 3,3 V zu erzeugen!

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. Zur Interpretation der Ergebnisse von Rechenoperationen werden neben dem Akkumulator A und den Registern R0 und R1 auch die Flags für Übertrag (Carry-Flag C) und Überlauf (Overflow-Flag OV) sowie das Negativ-Flag (N) betrachtet.

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

Tabelle 2.1: Registerbelegung eines Mikrocontrollers

Register	**Inhalt (dezimal)**	**Inhalt (binär)** MSB LSB
A	–100	
R0	123	
R1	107	

a) Tragen Sie in Tabelle 2.1 die Registerbelegungen in binärer 8 bit-Zweierkomplementdarstellung ein! Die Abkürzungen MSB und LSB stehen für das höchstwertige (most significant) und das niederstwertige (least significant) Bit.

Nun werden zwei Befehle ausgeführt: der erste Befehl addiert zum Akku den Inhalt von R0. Zu diesem neuen Akkuinhalt addiert der zweite Befehl den Inhalt von R1.

b) Tragen Sie in Tabelle 2.2 den Inhalt des Akkus sowie der Flags C, N und OV nach Ausführung des ersten und des zweiten Befehls ein! Verwenden Sie zur Darstellung des Akkuinhalts binäre 8 bit-Zweierkomplementdarstellung!

Tabelle 2.2: Register- und Flagbelegung nach Addition

	Akku (binär) MSB LSB	**C**	**N**	**OV**
nach Befehl 1				
nach Befehl 2				

Nun wird dieselbe Rechnung mit einem digitalen Signalprozessor durchgeführt, der Gleitkommadarstellung nach dem IEEE-P754-Standard mit einfacher Genauigkeit verwendet. Dieses Zahlenformat benutzt für die Darstellung einer Zahl 32 bit, wobei ein Bit für das Vorzeichen, acht Bit für den Exponenten und 23 Bit für die Mantisse benötigt werden. Das Ergebnis der Berechnungen wird auch hier in den Akkumulator A geschrieben, der jetzt eine Breite von 32 bit hat.

c) Tragen Sie das Ergebnis der ersten Berechnung (Befehl 1) in Tabelle 2.3 in Binärdarstellung ein!

Tabelle 2.3: Inhalt des Akkus nach 1. Addition

Bit-Nr.	31 24	23 16	15 8	7 0
Inhalt				

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

Die Verlustleistung in CMOS-Mikrorechnersystemen setzt sich im Wesentlichen aus zwei Komponenten zusammen: den Umladeverlusten, die beim Umladen der Schaltungskapazitäten entstehen, sowie den Schaltverlusten durch Querströme beim Umschalten von Invertern und Gattern.

Zunächst werden nur die Umschaltverluste betrachtet. Dabei wird ein Mikrocontrollermodell angenommen, bei dem im Mittel bei jeder Taktflanke 25.000 Inverter gleichzeitig schalten. Die äquivalente Taktfrequenz sei 10 MHz, wobei die Flankensteilheit des Taktes, d. h. jeweils Anstiegs- und Abfallzeit, 1,5 ns betrage. Der Stromverlauf durch einen Inverter ist in Bild 3.1 dargestellt. Zur Vereinfachung wird die Strom-Zeit-Fläche während eines Umschaltvorgangs durch ein gleichschenkliges Dreieck angenähert. Die Versorgungsspannung beträgt 3,3 V, der mittlere durch die Umschaltverluste verursachte Gesamtstrom 40 mA.

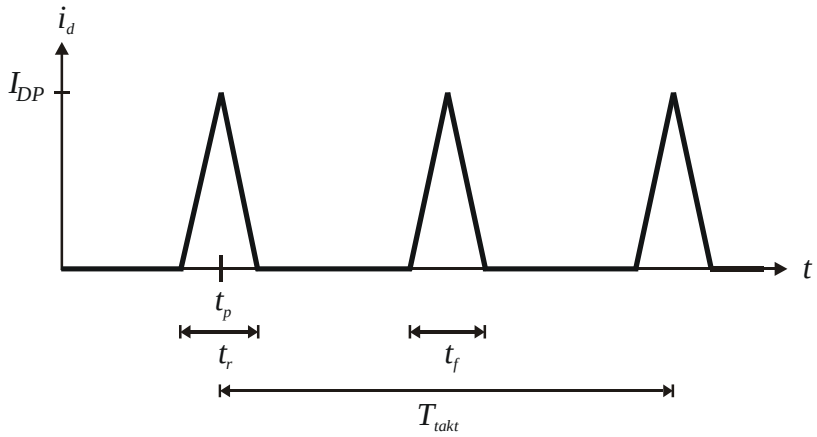

Bild 3.1: Stromverlauf durch eine Inverterstufe

a) Wie hoch ist die Stromspitze I_{DP} in der Strom-Zeit-Fläche nach Bild 3.1 bei einem Schaltvorgang eines Inverters?

b) Wie groß sind der maximale durch Umschaltverluste verursachte Strom und die maximale durch Umschaltverluste verursachte Verlustleistung der Schaltung?

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

c) Wie groß sind die Umladeverluste des Mikrocontrollers?

d) Um wie viel Prozent verringern sich die Umladeverluste der Schaltung, wenn die Versorgungsspannung auf 2,5 V abgesenkt wird?

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

Mit Hilfe der CMOS-Transfergatetechnologie soll ein EXOR-Gatter mit den beiden Eingängen X1 und X2 und dem Ausgang Y aufgebaut werden. Die Logiktabelle eines EXOR-Gatters ist in Tabelle 4.1 angegeben.

Tabelle 4.1: Logiktabelle eines EXOR-Gatters

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

a) Ergänzen Sie Bild 4.1 durch das Einzeichnen der fehlenden Verbindungen zu einem EXOR-Gatter!

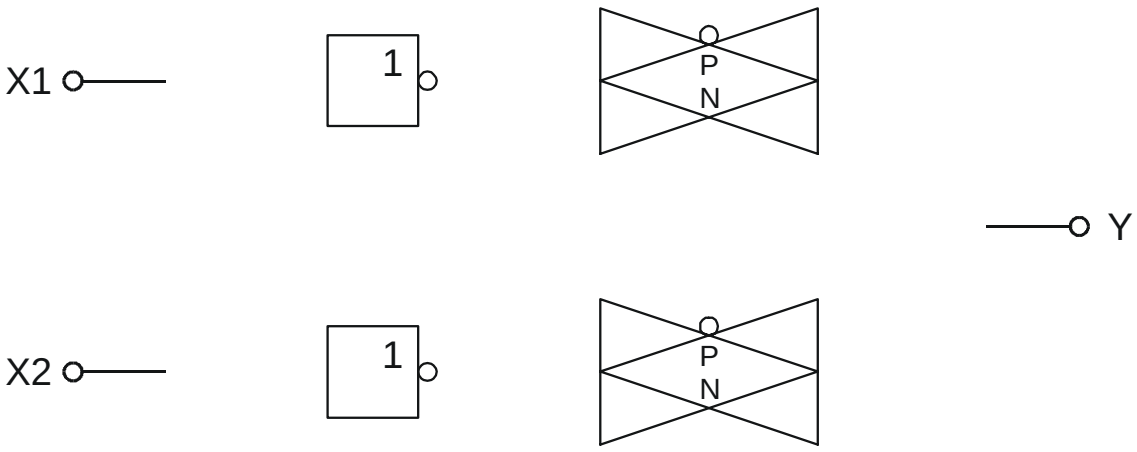

Bild 4.1: EXOR-Gatter in Transfergatetechnologie

Nun wird die in Bild 4.2 dargestellte Schaltung, bestehend aus zwei Transfergates und zwei Invertern, betrachtet. Der Eingang der Schaltung ist mit E bezeichnet, invertierter Ausgang und Ausgang mit Q1 und Q2. Die Anschlüsse T und $\overline{T}$ sind Eingänge für den nichtinvertierten und den invertierten Takt.

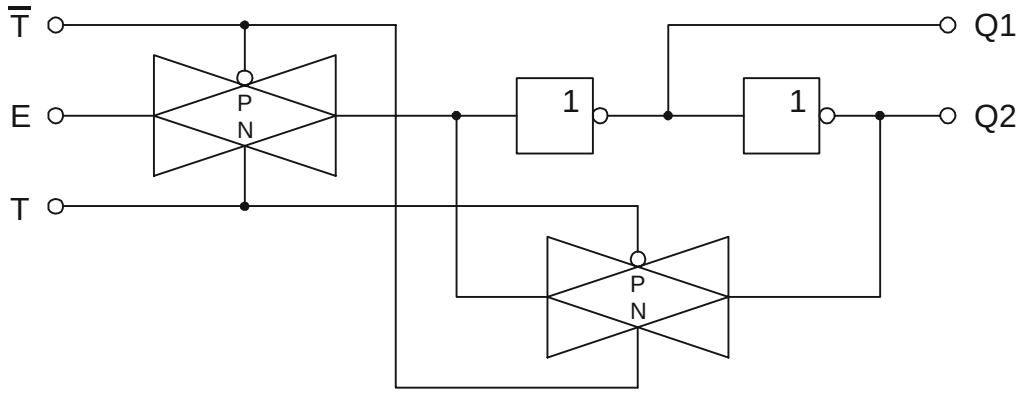

Bild 4.2: Schaltung mit Transfergates

An die Schaltung aus Bild 4.2 wird das Eingangssignal E nach Bild 4.3 angelegt.

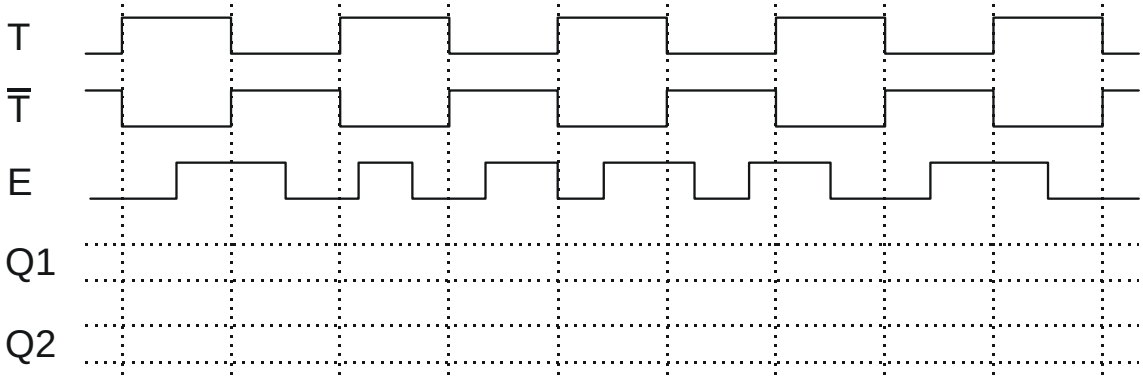

Bild 4.3: Signalverläufe in der Schaltung aus Bild 4.2

b) Tragen Sie in Bild 4.3 die Ausgangssignale Q1 und Q2 der Schaltung aus Bild 4.2 ein!

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

Zwei wichtige Grundschaltungen zum Aufbau von Addierern sind der Halbaddierer und der Volladdierer. Bild 5.1 zeigt das Blockschaltbild eines Halbaddierers (HA) und eines Volladdierers (VA).

Bild 5.1: Elementare Addierschaltungen

Der Halbaddierer addiert zwei 1 bit-Zahlen **x** und **y**. Er liefert als Ergebnis die Summe **s** und den Übertrag **c**. Der Volladdierer addiert drei 1 bit-Zahlen **x**, **y** und **z**. Er liefert ebenfalls eine Summe **s** und einen Übertrag **c**.

a) Vervollständigen Sie Bild 5.2 derart, dass aus den beiden Halbaddierern HA1 und HA2 sowie dem Oder-Gatter ein Volladdierer mit den Eingängen **x**, **y** und **z** sowie den Ausgängen **s** und **c** entsteht!

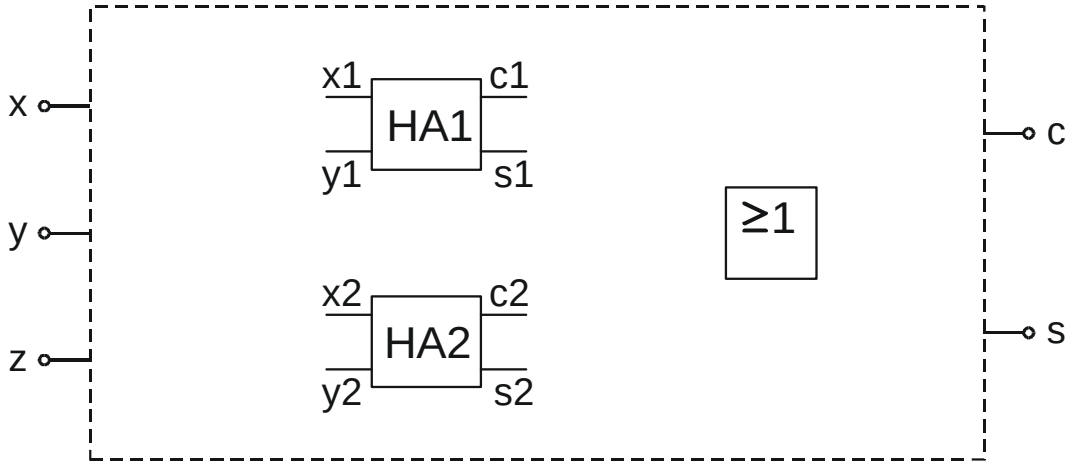

Bild 5.2: Aufbau eines Volladdierers

Nun soll ein Ripple-Carry-Addierer aufgebaut werden, der zwei 4 bit-Zahlen **a** und **b** sowie ein Übertragsbit **c** addiert. Als Ergebnis liefert der Addierer die 5 bit-Summe **s**.

b) Ergänzen Sie Bild 5.3 zu einem Ripple-Carry-Addierer, indem Sie alle fehlenden Verbindungen einzeichnen!

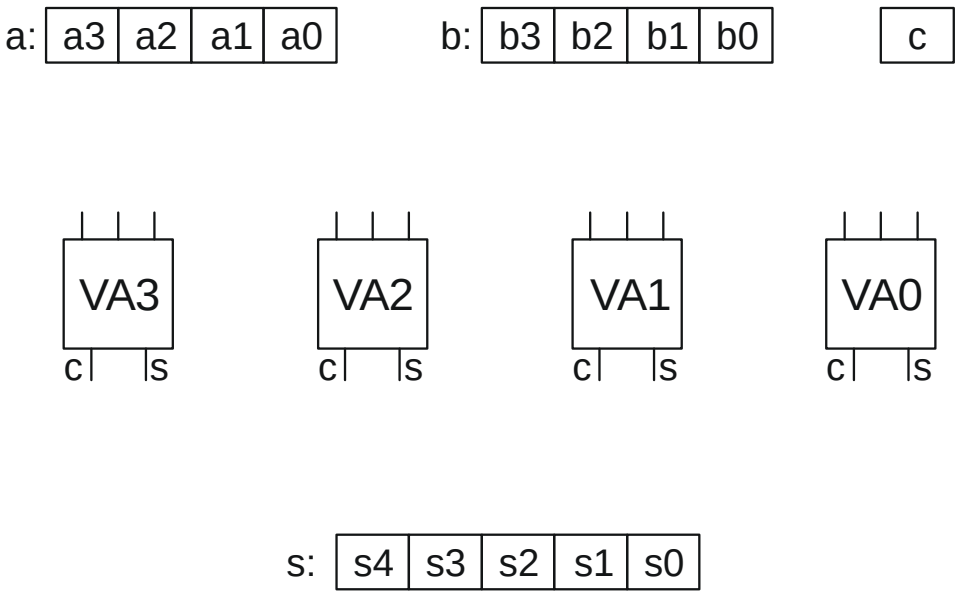

Bild 5.3: Ripple-Carry-Addierer zur Addition zweier 4 bit-Zahlen

Ein Carry-Look-Ahead-Addierer enthält zusätzliche kombinatorische Logik zur Vorausberechnung der Überträge. Ein solcher Addierer soll nun aufgebaut werden. Dazu stehen Volladdierer sowie eine kombinatorische Schaltung zur Berechnung der Überträge zur Verfügung. Diese Kombinatorik berechnet die Überträge c0 bis c3 der Volladdierer VA0 bis VA3 im Voraus.

c) Ergänzen Sie Bild 5.4 zu einem Carry-Look-Ahead-Addierer, der zwei 4 bit-Zahlen **a** und **b** sowie ein Übertragsbit **c** addiert, indem Sie alle fehlenden Verbindungen einzeichnen! Als Ergebnis liefert der Addierer die 5 bit-Summe **s.**

Hinweis: Es werden nicht alle Ausgänge der Volladdierer benötigt.

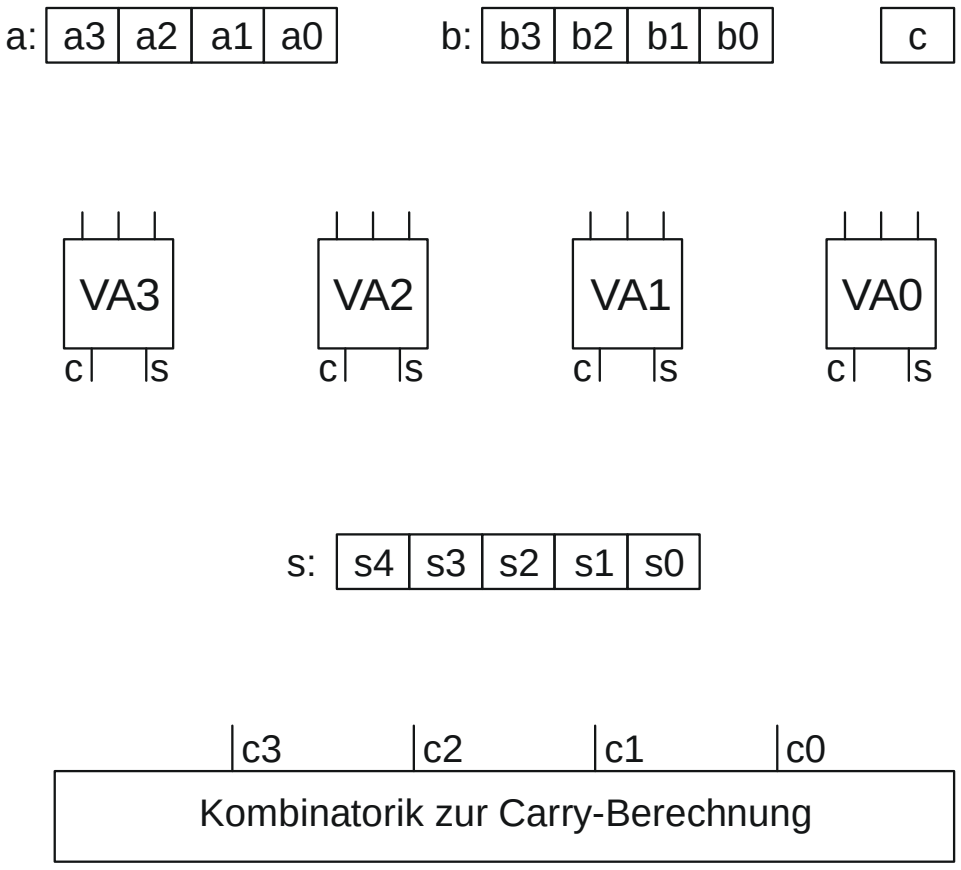

Bild 5.4: Carry-Look-Ahead-Addierer zur Addition zweier 4 bit-Zahlen

d) Nennen Sie einen Vorteil und einen Nachteil des Carry-Look-Ahead-Addierers aus Teil c) im Vergleich zum Ripple-Carry-Addierer aus Teil b)!

Aufgabe 6: **Entwurf eines Steuerwerks** **(12 Punkte)**

Das Steuerwerk stellt ein wichtiges Kernstück eines jeden Mikrorechners dar. Im Folgenden wird ein solches Steuerwerk in Form eines Mikrosequencers betrachtet. Bild 6.1 zeigt einen Mikrosequencer mit den zugehörigen Ein- und Ausgängen.

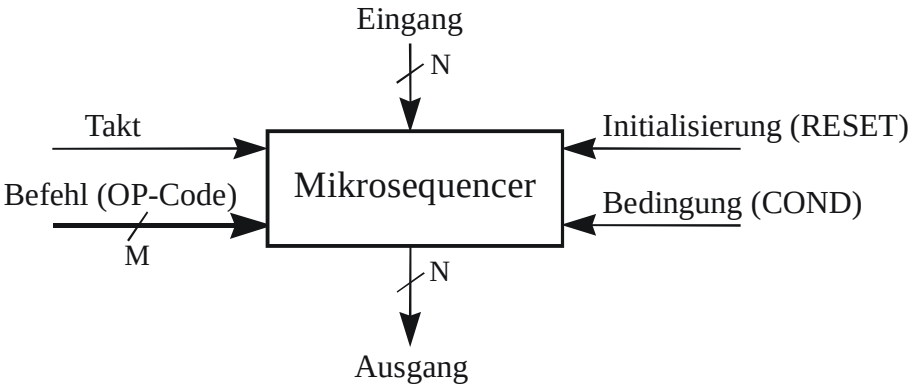

Bild 6.1: Blockschaltbild eines Mikrosequencers

Nun wird der Mikrosequencer in einer Anwendung für einen 8 bit-Mikrorechner untersucht. Der Mikrosequencer kann sechs verschiedene Befehle ausführen. Die FSM (Finite State Machine) des Mikrosequencers benötigt zur Ablaufsteuerung 14 Zustände. Zur Ansteuerung der Mikrosequencer-Hardware werden zehn Steuersignale benötigt. Bild 6.2 zeigt den unvollständigen Aufbau des Mikrosequencers aus Bild 6.1.

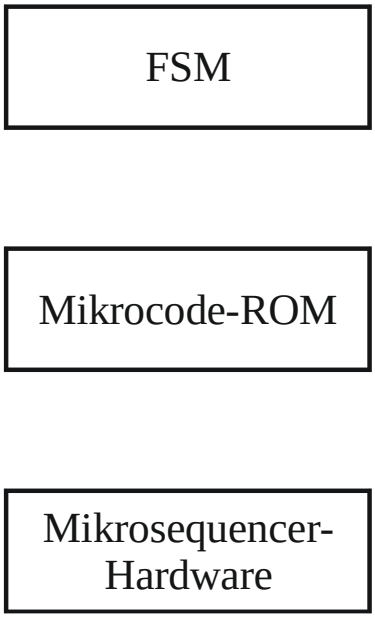

Bild 6.2: Aufbau des Mikrosequencers aus Bild 6.1

a) Zeichnen Sie in Bild 6.2 alle Eingangs- und Ausgangssignale sowie die Signale zwischen den Blöcken ein! Geben Sie bei allen Signalen deren Bitbreite an! Gehen Sie davon aus, dass die Befehle und Zustände so kodiert sind, dass die Bitbreiten der Signale minimal sind.

Bild 6.3 zeigt die Mikrosequencer-Hardware aus Bild 6.2 in Register-Transfer-Darstellung.

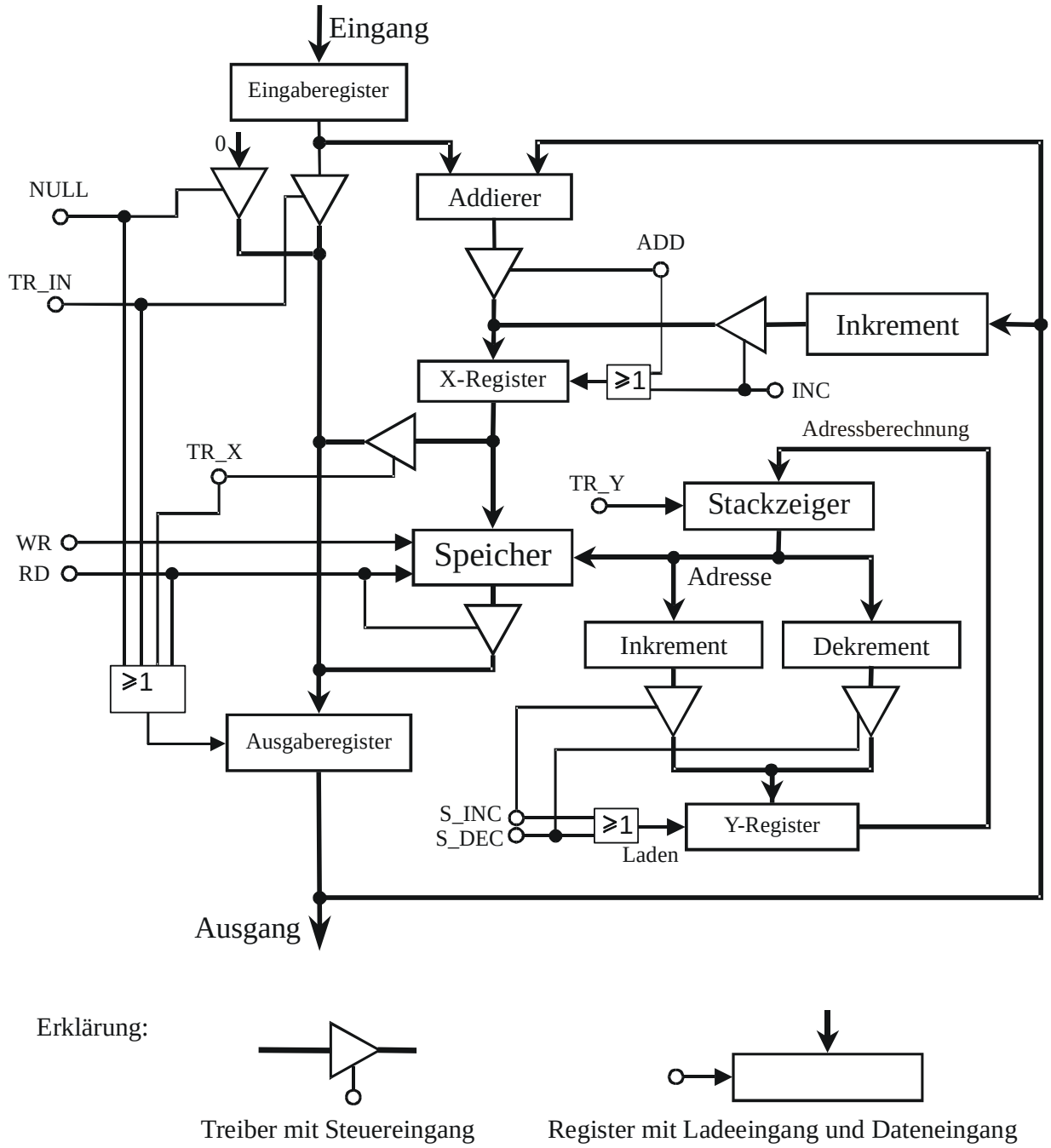

Bild 6.3: Hardware des Mikrosequencers in Register-Transfer-Darstellung

Nun soll der Befehl zum Sprung in ein Unterprogramm analysiert werden. Dieser Befehl inkrementiert die aktuelle Adresse (Ausgang) und legt sie im Speicher ab. Anschließend wird die neue Adresse (Eingang) an den Ausgang geschrieben. Außerdem wird der Stackzeiger inkrementiert. Dazu muss die Mikrosequencer-Hardware aus Bild 6.3 folgende Teilaufgaben ausführen:

1. Ausgangswert inkrementieren und in das X-Register schreiben
2. X-Register in Speicher schreiben und Eingaberegister in Ausgaberegister schreiben
3. Inkrementierten Stackzeiger ins Y-Register schreiben
4. Y-Register in Stackzeiger schreiben

b) Tragen Sie in Tabelle 6.1 die benötigten Werte der Hardware-Steuersignale für die vier oben beschriebenen Schritte ein!

Tabelle 6.1: Hardware-Steuersignale zur Ausführung des Befehls „Sprung in Unterprogramm“

	Null	TR_IN	TR_X	ADD	INC	S_INC	S_DEC	TR_Y	RD	WR
Schritt 1										
Schritt 2										
Schritt 3										
Schritt 4										

Aufgabe 7: **Analyse eines Mikrocontrollerprogramms** **(10 Punkte)**

Es soll ein Programmabschnitt in 8051-Assemblersprache analysiert werden, der arithmetische Operationen mit den Inhalten der Register R0 und R1 durchführt. Die beiden Register enthalten positive 8 bit-Binärzahlen, wobei die Bedingung „Inhalt von R0 $\geqslant$ Inhalt von R1“ stets erfüllt ist. Beachten Sie bei der Lösung der Aufgabe, dass alle Berechnungen im Zweierkomplement erfolgen.

Es wird folgender Programmabschnitt in 8051-Assemblercode betrachtet:

```
Anfang:
    CLR  C
    MOV  R2,#0
    MOV  A,R0
Schleife:
    MOV  R3,A
    SUBB A,R1
    JB   ACC.7,Endlos
    INC  R2
    JMP  Schleife
Endlos:
    NOP
    JMP  Endlos
```

a) Analysieren Sie den Programmcode, indem sie das Flussdiagramm in Bild 7.1 vervollständigen!

Hinweise:

`CLR C`	Carry löschen
`INC R`*n*	Register *n* inkrementieren
`ACC.7`	Bit 7 (höchstwertiges Bit) des Akkumulators; gibt bei Zweierkomplementdarstellung das Vorzeichen des Akkumulators an
`MOV dest,source`	source nach dest schreiben
`SUBB A,R`*n*	Subtraktion mit „Borgen“, A := A – R*n* – C
`JB bit,Adresse`	Sprung an Adresse, wenn bit gesetzt ist
`JMP Adresse`	unbedingter Sprung an Adresse
`NOP`	keine Operation

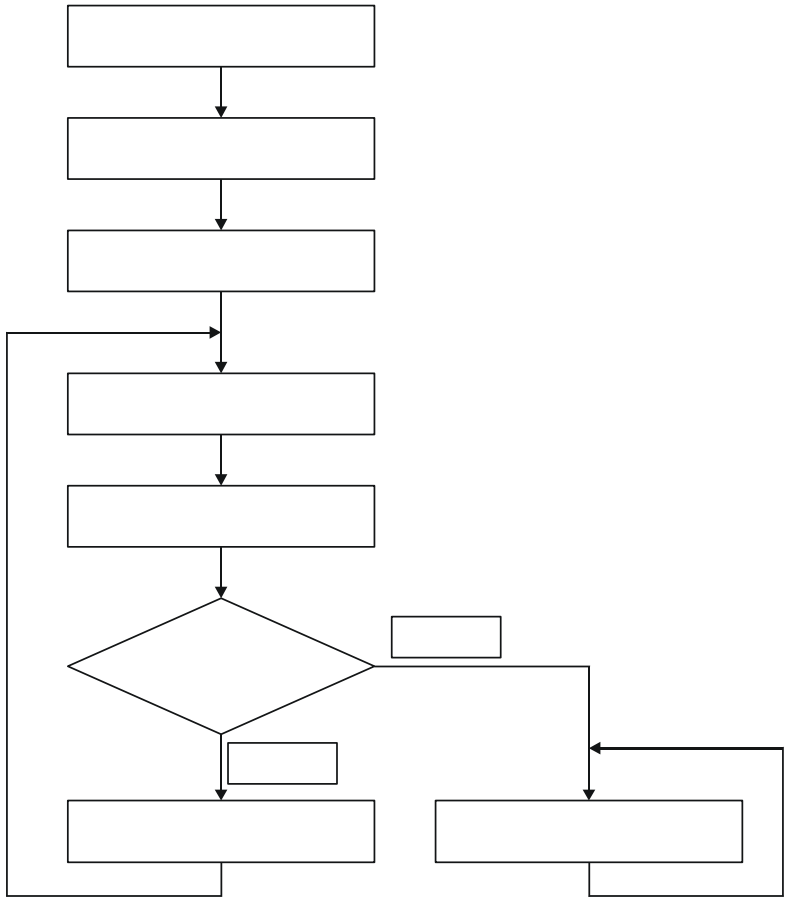

Bild 7.1: Flussdiagramm des Programmabschnitts

b) Welche arithmetische(n) Operation(en) führt der Programmabschnitt mit den Inhalten der Register R0 und R1 aus? In welchem Register bzw. in welchen Registern steht das Ergebnis/die Ergebnisse?

In Tabelle 7.1 ist die Registerbelegung zu Beginn des Programmabschnitts in Dezimaldarstellung angegeben.

Tabelle 7.1: Registerbelegung bei Programmbeginn

Register	**R0**	**R1**
Registerinhalt	35	10

c) Tragen Sie in Tabelle 7.2 die Registerbelegung nach Ende des Programmabschnitts in Dezimaldarstellung ein!

Tabelle 7.2: Registerbelegung nach Programmende

Register	**R0**	**R1**	**R2**	**R3**
Registerinhalt				

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

In einem Mikrocontroller 8051 soll die integrierte serielle Schnittstelle in Betriebsart 1 (8 bit UART mit variabler Baudrate) programmiert werden. Zur Baudratengenerierung dient die Überlaufrate von Timer 1. Tabelle 8.1 zeigt die Belegung der relevanten Spezialfunktionsregister des Mikrocontrollers. Irrelevante Bits sind durch ‚X’ gekennzeichnet.

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

Tabelle 8.1: Belegung der Spezialfunktionsregister

	Bit 7							Bit 0
PCON	1	X	X	X	X	X	X	X
SCON	0	1	X	1	X	X	0	0
TMOD	0	0	1	0	X	X	X	X
TH1	1	0	0	1	1	0	0	0
IE	1	X	X	1	0	X	X	X

a) Bestimmen Sie die Baudrate, wenn der Mikrocontroller mit 12 MHz getaktet ist!

Nun soll der Reloadwert für Timer 1 so verändert werden, dass die erzeugte Baudrate möglichst niedrig ist. Die Taktfrequenz sowie die Belegung der übrigen Register gemäß Tabelle 8.1 bleiben unverändert.

b) Geben Sie den Reloadwert für Timer 1 sowie die damit erzeugte Baudrate an!

Nun soll eine Baudrate von 9600 mit möglichst hoher Genauigkeit erzeugt werden. Die Betriebsarten des Timers und der Schnittstelle gemäß Tabelle 8.1 bleiben unverändert. Es stehen drei handelsübliche Quarze zur Auswahl, deren Frequenzen in Tabelle 8.2 aufgeführt sind.

Tabelle 8.2: Quarze zur Takterzeugung für den Mikrocontroller 8051

Quarz	Frequenz in MHz
1	11,0000
2	11,0592
3	12,0000

c) Bestimmen Sie den am besten geeigneten Quarz, indem Sie den Fehler der erzeugten Baudrate minimieren!

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

Ein DSP56000 führt mit Daten, die über einen A/D-Wandler eingelesen werden, den untenstehenden Programmausschnitt aus.
Der externe A/D-Wandler ist unter der X-Datenspeicher-Adresse $1000 angeschlossen und liefert in fünf aufeinanderfolgenden Maschinenzyklen die folgenden Werte in Dezimaldarstellung:

1/32, 1/16, 1/8, 1/4, 1/2

Im Y-Datenspeicher ist ein Ringpuffer der Länge 5 ab Adresse $40 mit folgendem Inhalt angelegt:

Adresse	Inhalt (dezimal)
$40	1/2
$41	1/4
$42	1/8
$43	1/16
$44	1/32

Es wird der folgende Programmausschnitt betrachtet:

```
MOVE   #$1000,R1          ; Werte von externem ADW einlesen
MOVE   #$40,R5            ; Pointer auf internen Y-Datenspeicher

MOVE   X:(R1),X0          ; Startwert in X0
MOVE   Y:(R5)+,Y0         ; Startwert in Y0
CLR    A                  ; Akku A löschen

DO     #5,ENDE
MAC    X0,Y0,A    X:(R1),X0    Y:(R5)+,Y0

ENDE
```

Hinweise:
- *ein $-Zeichen nach dem # bedeutet, dass eine Zahl im Hexadezimal-Format folgt*
- *kein Zeichen nach dem # bedeutet, dass eine Dezimalzahl folgt*
- *Für das Ergebnis ist die bei DSPs übliche Fraktalzahlendarstellung zu verwenden.*

a) Tragen Sie in Tabelle 9.1 in Hexadezimalformat die Inhalte in die Teilstücke des Akkumulators A ein, die sich nach Abarbeitung des Programmausschnitts ergeben!

Tabelle 9.1: Akkumulatorinhalt nach Abarbeitung des Programmausschnitts

A2	A1	A0
$	$	$

b) Welche Zahl – in Dezimaldarstellung – ist in welches Modifier-Register M*n* einzutragen, damit der benötigte Ringpuffer eingerichtet wird?

c) Erklären Sie, weshalb bei der MAC-Instruktion im obigen Programmausschnitt in der Operandengruppe `X:(R1),X0` kein + Zeichen hinter (R1) steht!

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

Gegeben ist das folgende Blockdiagramm einer MAC-Einheit:

Bild 10.1: Blockdiagramm einer MAC-Einheit

Die MAC-Einheit aus Bild 10.1 wird durch folgenden unvollständigen VHDL-Code beschrieben:

```
ENTITY MAC_EH IS
  PORT (clock, RESET, CLEAR: IN  bit;
        EIN_X, EIN_Y:        IN  Integer Range 0 to 255;
        ERGEBNIS:            OUT Integer Range 0 to 524287
       );
END MAC_EH;

ARCHITECTURE MAC_CAL OF MAC_EH IS
  SIGNAL INTERN: Integer Range 0 to 524287;
BEGIN
  PROCESS (clock, RESET)
    BEGIN
      IF RESET = '0' THEN
        INTERN <= 0;
      ELSIF (                                  ) THEN
         IF CLEAR = '0' THEN
           INTERN <=                                ;
         ELSE
           INTERN <=                                ;
         END IF;
      END IF;
    END PROCESS;
  ERGEBNIS <=                                       ;
END MAC_CAL;
```

a) Tragen Sie in der VHDL-Beschreibung in den fett gedruckten Zeilen die fehlenden Codestücke ein!

b) Wie viele MAC-Operationen sind maximal möglich, ohne dass ein Überlauf entsteht, wenn die Eingangsoperanden immer ihre größtmöglichen Werte haben?

Bild 10.2 zeigt das FSM-Modell eines Mooreautomaten. Ein „x“ als Übergangsbedingung bedeutet, dass der entsprechende Eingangswert irrelevant ist.

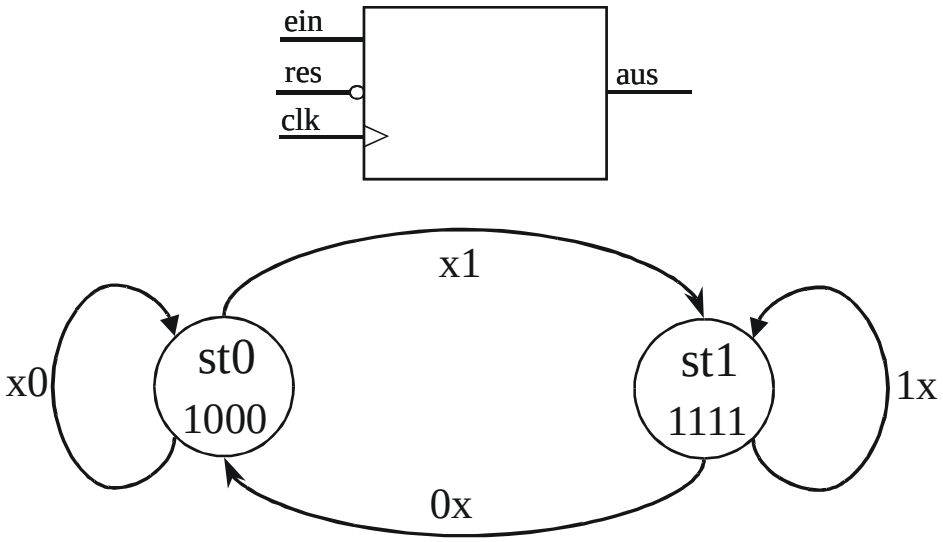

Bild 10.2: Ein einfacher Automat mit zwei Zuständen

VHDL-Beschreibung des Automaten aus Bild 10.2:

```
ENTITY M_AUT is
  PORT (clk, res: IN  bit;
        ein:      IN  bit_vector (1 downto 0);
        aus:      OUT bit_vector (3 downto 0)
  );
END M_AUT;

ARCHITECTURE FUNCT OF M_AUT IS
  TYPE   zustands_werte IS (st0, st1);
  SIGNAL akt_zust, folg_zust: zustands_werte;

BEGIN
-- Definition der FSM-Register
zust_reg: PROCESS (clk, res)
  BEGIN
    IF (res = '0') THEN
      akt_zust <= st0;
    ELSIF (clk'event AND clk = '1') then
      akt_zust <= folg_zust;
    END IF;
END PROCESS zust_reg;

-- Kombinatorik der FSM
FSM: PROCESS (akt_zust, ein)
```

```
BEGIN
  case akt_zust IS
    WHEN st0 =>
      CASE ein IS
        WHEN "    " => folg_zust <=       ;

        WHEN "    " => folg_zust <=       ;

        WHEN "    " => folg_zust <=       ;

        WHEN "    " => folg_zust <=       ;
      END CASE;
    WHEN st1 =>
      CASE ein IS
        WHEN "00"   => folg_zust <=       ;

        WHEN "01"   => folg_zust <=       ;

        WHEN OTHERS => folg_zust <=       ;
      END CASE;
  END CASE;
END PROCESS FSM;

-- Ausgabewerte
AUSGABE: PROCESS (akt_zust)
BEGIN
  CASE akt_zust is
    WHEN st0    => aus <= "       ";

    WHEN st1    => aus <= "       ";

    WHEN OTHERS => aus <= "1111";
  END CASE;
END PROCESS AUSGABE;

END FUNCT;
```

c) Tragen Sie in der VHDL-Beschreibung in den fett gedruckten Zeilen die fehlenden Codestücke ein, so dass der Automat nach Bild 10.2 realisiert wird!