Vorname:_____ **Name:**_____

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

20.09.2004
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:	10	10	11	9	10	10	10	10	10	10	100

Aufgabe 1: **Parallele Multiplikation** **(10 Punkte)**

In Bild **1.1** ist ein speicherbasierter Multiplizierer für zwei vorzeichenlose 3 bit-Binärzahlen skizziert.

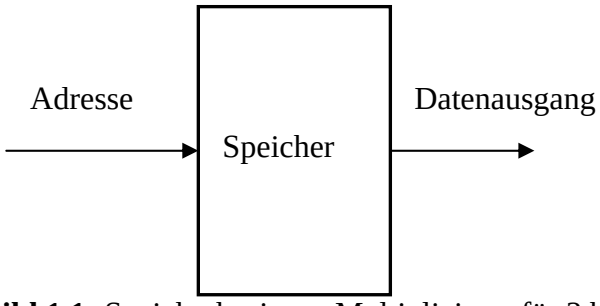

Bild 1.1: Speicherbasierter Multiplizierer für 3 bit-Binärzahlen

a) Geben Sie an, wie viele Bits die Adresse und der Datenausgang haben müssen!

Nun ist der Speicherinhalt für die speicherbasierte Multiplikation von zwei vorzeichenlosen 2 bit Binärzahlen anzugeben.

b) Vervollständigen Sie dazu die folgende Tabelle 1.1!

Adresse	Inhalt
0000	0000
0001	
0010	
0100	
0101	
0110	
0111	
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	

Tabelle 1.1: Adresse und Speicherinhalt

Im weitern wird die Arbeitsgeschwindigkeit eines Parallelmultiplizierers für zwei 8-bit-Zweikomplementzahlen *A* und *B* betrachtet, der gemäß Gleichung 1 ein 16 bit langes Produkt *P* als Ergebnis liefert:

$$P = A \cdot B = -2^{15} + 2^8 + a_7 \cdot b_7 \cdot 2^{14} + \sum_{j=0}^{6} \sum_{i=0}^{6} a_i \cdot b_j \cdot 2^{i+j} + \sum_{i=0}^{6} \overline{a_7 b_i} \cdot 2^{i+7} + \sum_{i=0}^{6} \overline{b_7 a_i} \cdot 2^{i+7} \qquad \textbf{Gleichung 1}$$

Die Produktbildung der Komponenten $a_i \cdot b_j$ erfolgt mittels UND-Gattern und die der negierten Komponenten $\overline{b_i \cdot a_j}$ mittels NAND-Gattern. Die Gatter haben jeweils eine Durchlaufzeit von 1ns.

Die Abarbeitung bis hin zum Endergebnis erfolgt nach dem Schema in Bild 1.2, unter Einsatz von Halb- und Volladdierern, sowie eines schnellen Carry-Look-Ahead-Addierers für die Abschussaddition.

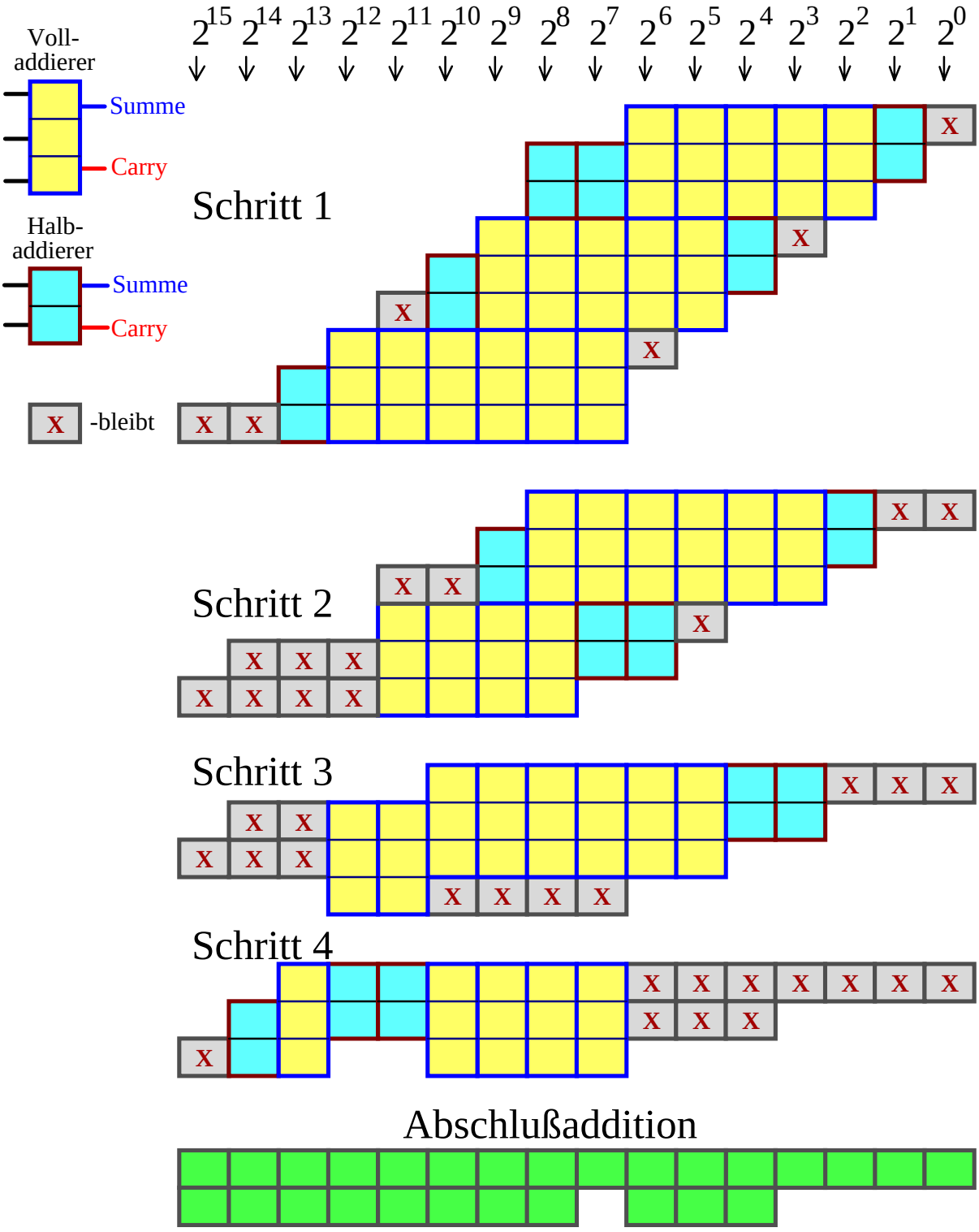

Bild 1.2: Schematische Darstellung der Abarbeitung der Multiplikation nach Gleichung 1 bis zur Abschlussaddition mit Hilfe von 15 Halb- und 39 Volladdierern in 4 Schritten

c) Wie lange dauert die Ausführung einer Multiplikation, wenn ein Volladdierer und der Carry-Look-Ahead-Addierer für die Abschlussaddition jeweils 2ns Durchlaufzeit haben?

Jetzt wird nach jedem der obigen Schritte 1...4 ein Pipelining-Register eingebaut. Alle Register weisen Durchlaufzeit von 0,5ns auf.

d) Mit welcher Taktfrequenz f_{max} können die Pipelining-Register betrieben werden, damit, nachdem die gesamte Pipeline gefüllt ist, mit jedem Taktschritt ein Multiplikationsergebnis geliefert wird?

e) Wie viele Bits muss das Register, das nach Schritt 3 einzufügen ist, speichern?

Aufgabe 2: **Zahlendarstellung in Mikrorechnerprogrammen (10 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 binärer 8 bit-Zweierkomplementdarstellung angegeben. Die Abkürzungen MSB und LSB stehen für das höchstwertige (most significant) und das niederwertigste (least significant) Bit. Das MSB gibt das Vorzeichen an.

Register	**Inhalt (dezimal)**	**Inhalt (binär)** MSB							LSB
R0		1	1	1	1	1	0	1	0
R1		0	0	0	1	0	0	1	0
A									

Tabelle 2.1: Registerbelegung eines Mikrocontrollers

a) Tragen Sie in Tabelle 2.1 die Inhalte der Register R0 und R1 in Dezimaldarstellung ein!

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 der Mikrocontroller mit Fraktalzahlendarstellung arbeitet!

d) Geben Sie den Dezimalwert des Registerinhalts von R0 an, wenn die Binärzahl nicht mehr in Zweierkomplentdarstellung sondern als Festkommazahl mit 4 Stellen nach dem Komma interpretiert wird!

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

e) Ergänzen Sie in Tabelle 2.2 in der ersten Zeile für die Zahl -21 die 32bit Gleitkommazahl und in der zweiten Zeile den Dezimalwert der gegebenen Gleitkommazahl!
Die binäre Darstellung entspricht dabei dem IEEE-P754 Single-Precision-Standard.

Inhalt (dezimal)	MSB **Inhalt (binär)** LSB																													
	31							24	23							16	15							8	7							0
–21																														
	0	1	0	0	0	0	1	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabelle 2.2: Speicherbelegung eines digitalen Signalprozessors

Aufgabe 3: **Verlustleistung von CMOS-Schaltungen** **(11 Punkte)**

Hochintegrierte digitale Schaltkreise werden nahezu ausschließlich in CMOS-Technologie gefertigt. Ein wichtiges Element solcher Schaltungen sind Inverter.

a) Skizzieren Sie den Aufbau eines CMOS-Inverters, bestehend aus einem p- und einem n-Kanal MOS-Transistor! Kennzeichnen Sie Eingang und Ausgang der Schaltung und erläutern Sie die Funktionsweise des Inverters!

b) Bei Schaltvorgängen des Inverters treten Verluste auf. Aus welchen zwei Komponenten setzen sich diese im Wesentlichen zusammen und wodurch werden sie jeweils verursacht?

In Bild 3.1 ist der Sromverlauf durch einen Inverter beim Umschalten dargestellt. Zur Vereinfachung wird die Strom-Zeit-Fläche während eines Umschaltvorganges durch ein gleichschenkliges Dreieck angenähert.
Beachten Sie, dass die Breite der Dreiecke für beide Taktflanken nicht gleich ist!

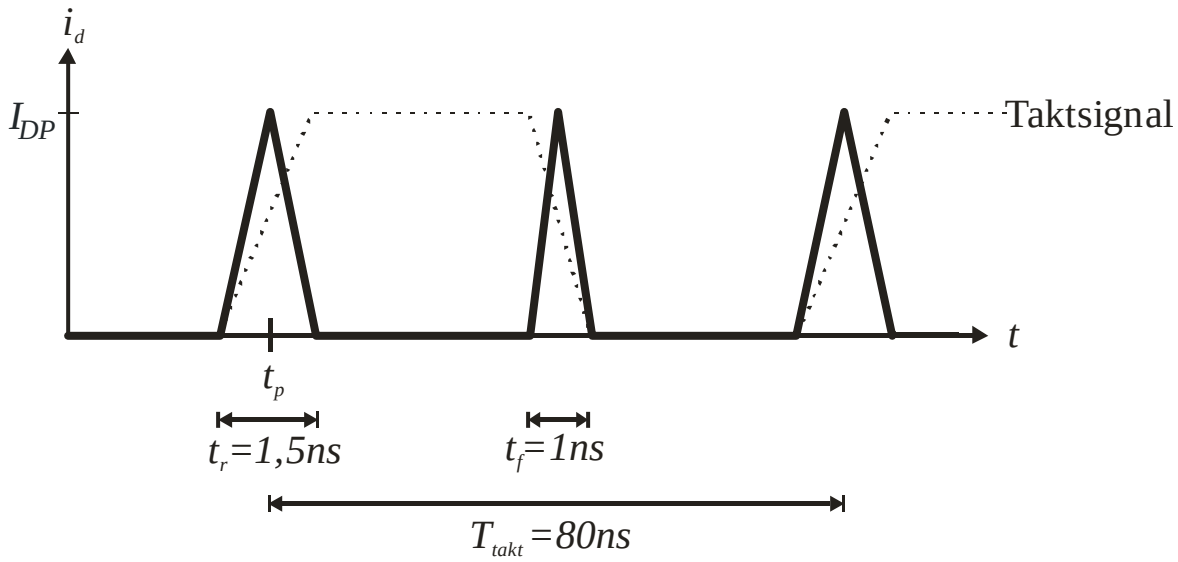

Bild 3.1: Stromverlauf durch eine Inverterstufe bei Umschaltvorgängen

c) Wie hoch ist die in Bild 3.1 skizzierte Stromspitze I_{DP}, wenn ein Microcontrollermodell betrachtet wird, bei dem im Mittel bei jeder Taktflanke 32.000 Inverter gleichzeitig schalten und der pro Taktperiode verursachte mittlere Gesamtstrom 48 mA beträgt?
Hinweis: Die Vorder- und Rückflanke des Taktes haben verschiedene Strom-Zeit-Flächen!

d) Wie groß sind der maximale durch Umschaltverluste verursachte Strom und die maximale Verlustleistung bei einer Versorgungsspannung mit einem Spitzenwert von 2,5 V?

Aufgabe 4: **CMOS-Transfergates** **(9 Punkte)**

Die CMOS-Technologie ermöglicht den Aufbau besonderer Schaltungsstrukturen in Form von Transfergates.

a) Skizzieren Sie den Aufbau eines CMOS-Transfergates und nennen Sie die wesentlichen Eigenschaften !

b) In Bild 4.1 ist eine mit Transfergates aufgebaute Schaltung abgebildet. Füllen Sie die Wahrheitstabelle (Tabelle 4.1) aus! Welche Funktion wird realisiert?

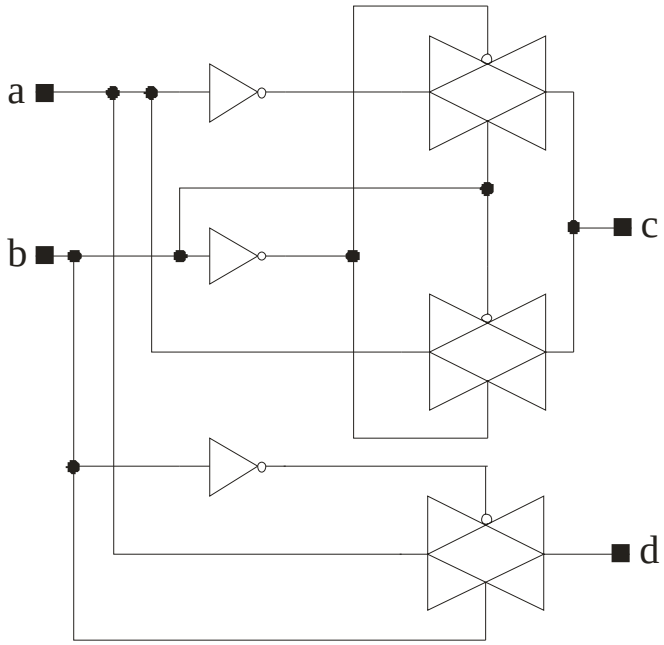

Bild 4.1: Schaltung mit Transfergates

a	b	c	d

Tabelle 4.1: Wahrheitstabelle

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

Zwei wichtige Grundschaltungen zum Aufbau von Addierern sind der Halbaddierer und der Volladdierer. Der Halbaddierer addiert zwei 1 bit-Zahlen *x* und *y* und liefert als Ergebnis die Summe *s* und den Übertrag *c*. Der Volladdierer addiert drei 1 bit-Zahlen und liefert ebenfalls eine Summe *s* und einen Übertrag *c*.

a) Geben Sie die booleschen Gleichungen für die Summe s_i und den Übertrag c_{i+1} bei der Addition zweier einstelliger Dualzahlen a_i und b_i an!

b) Der Halbaddierer soll nun in Hardware umgesetzt werden.
Ergänzen Sie hierzu das Schaltnetz in Bild 5.1 durch Ausfüllen der Schaltungsblöcke derart, dass ein Halbaddierer entsteht!

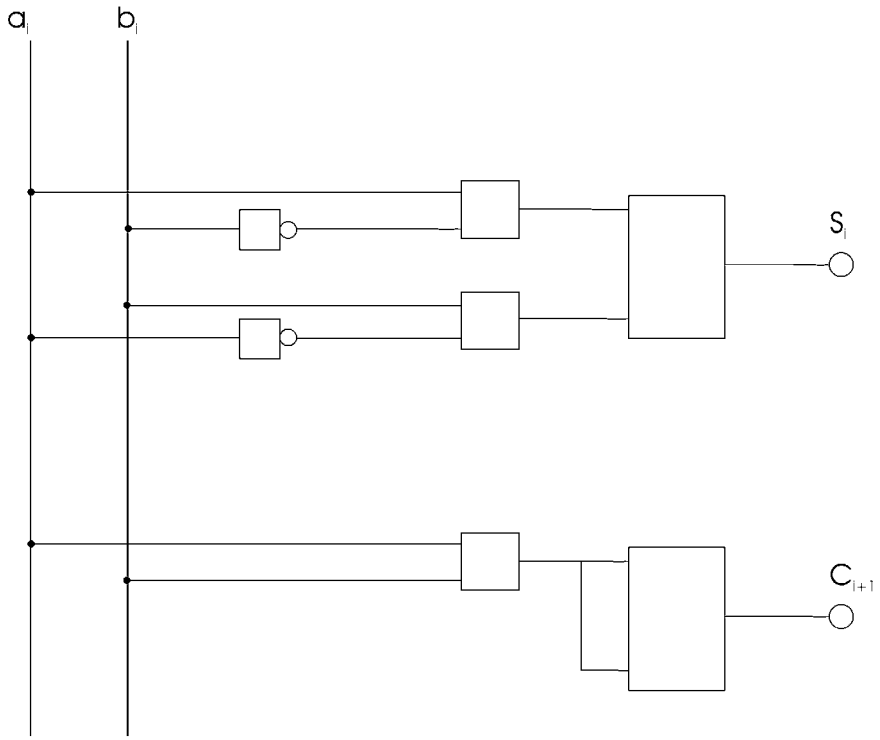

Bild 5.1: Schaltnetz zur Realisierung eines Halbaddierers

c) 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!

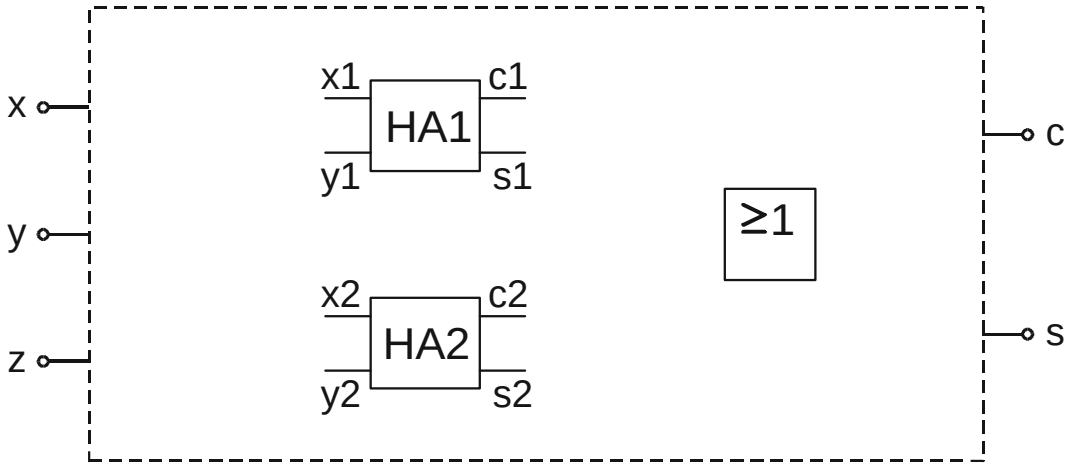

Bild 5.2: Aufbau eines Volladdierers

Nun soll ein Carry-Look-Ahead-Addierer aus Volladdierern und einer kombinatorischen Logikschaltung aufgebaut werden. Die kombinatorische Logikschaltung berechnet die Überträge *c0* bis *c3* der Volladdierer VA0 bis VA3 im Voraus.

d) Ergänzen Sie Bild 5.3 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.

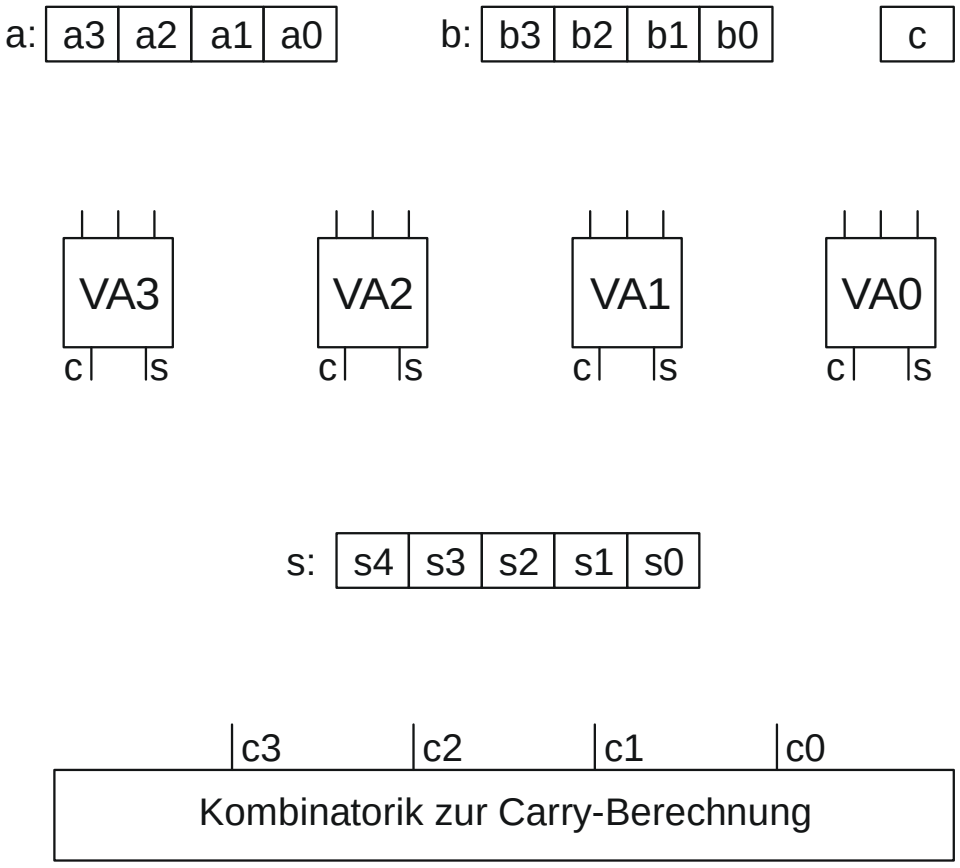

Bild 5.3: Carry-Look-Ahead-Addierer zur Addition zweier 4 bit-Zahlen

Aufgabe 6: **Beschreibung einer FSM** **(10 Punkte)**

In Bild 6.1 ist der Zustandsgraph des Steuerwerks einer einfachen programmgesteuerten Maschine abgebildet. Das Steuerwerk realisiert die in Tabelle 6.1 aufgeführten Befehle. Es stehen als Arbeitsregister ein Akkumulator (A) und ein Eingaberegister (INR) zur Verfügung. Das von außen kommende Signal INIT ist ein Hardware-Reset.

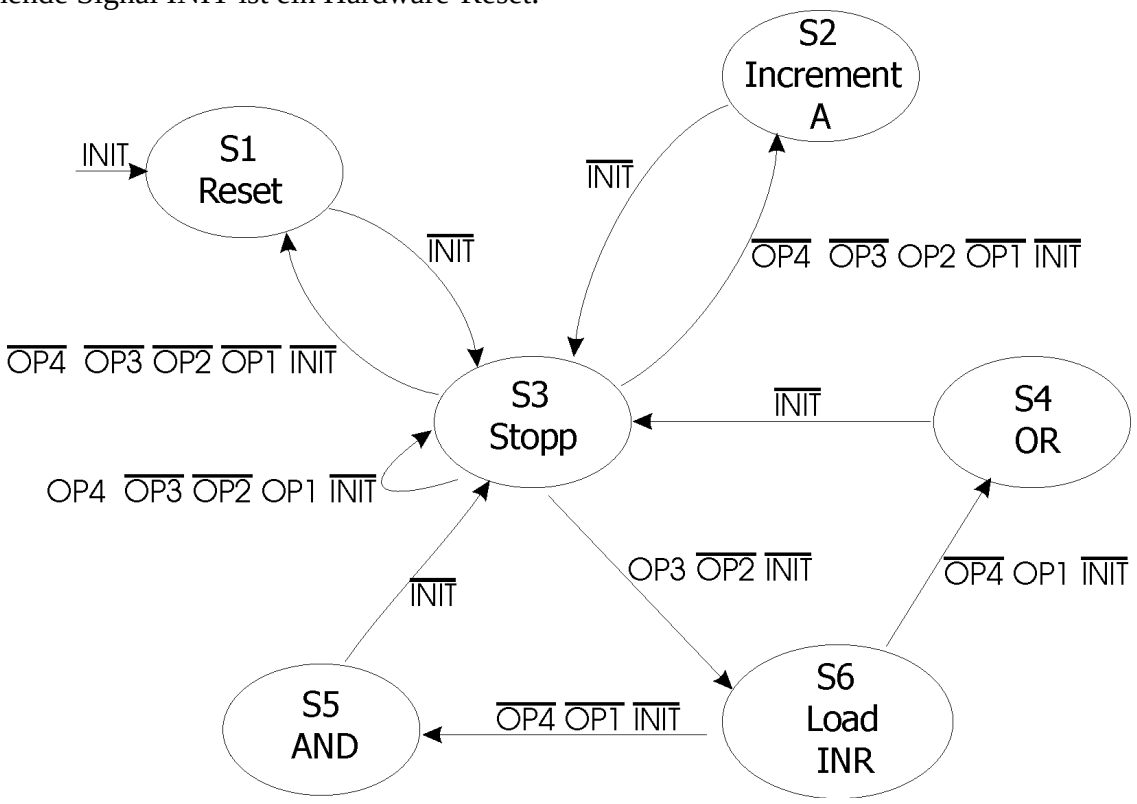

Bild 6.1: Zustandsgraph eines Steuerwerks

a) Ordnen Sie den Befehlen in Tabelle 6.1 die jeweiligen Binärkombinationen (OP4, OP3, OP2, OP1), d.h. die OP-Codes zu!

Befehl	OP4	OP3	OP2	OP1	Beschreibung
INC A					Inkrementiere A
ANDL A, INR					A= (INR) AND (A)
ORL A, INR					A= (INR) OR (A)
NOP					keine Operation

Tabelle 6.1: Realisierte Befehle

In Bild 6.2 ist eine Mikrosequenzer-Hardware auf Registertransferebene dargestellt, mit der Werte aus einem Speicher gelesen, in einen Speicher geschrieben und addiert werden können.
In der folgenden Tabelle 6.2 ist die Zuordnung einiger Zustände der als Moore-Automat realisierten Finite State Machine (FSM) zu den Eingängen der Mikrosequenzer-Hardware aufgeführt.

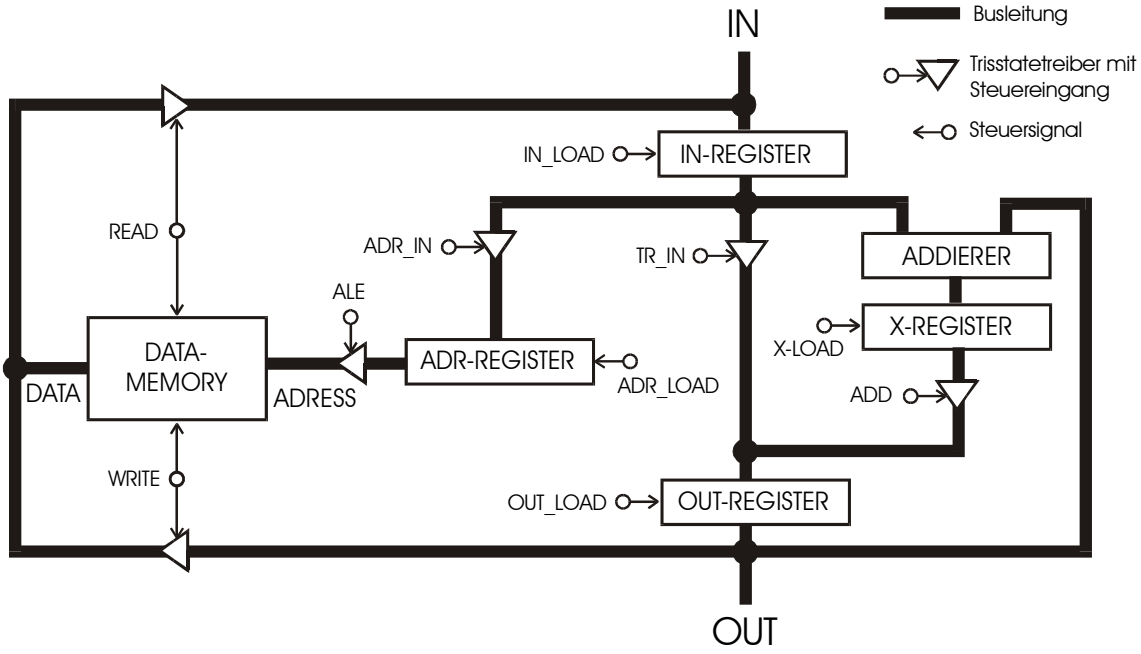

Bild 6.2: Mikrosequenzer auf Registertransferebene

Zustand	IN_LOAD	ADR_IN	ADR_ LOAD	ALE	READ	WRITE	TR_IN	X_LOAD	ADD	OUT_ LOAD
S_0	0	0	0	0	0	0	0	0	0	0
S_1	1	0	0	0	0	0	0	0	0	0
S_2	0	0	0	0	0	0	0	1	0	0
S_3	0	0	0	1	0	1	0	0	0	0
S_4	0	1	1	0	0	0	0	0	0	0

Tabelle 6.2: Zuordnung der Zustände zu den Eingangssignalen

b) Geben Sie den Zustand $\mathbf{S}_i$ bzw. die Folge von Zuständen $\mathbf{S}_i \ldots \mathbf{S}_j$ an, die notwendig sind, um den Befehl (**NOP** = keine Operation) auszuführen!

c) Geben Sie den Zustand $\mathbf{S}_i$ bzw. die Folge von Zuständen $\mathbf{S}_i \ldots \mathbf{S}_j$ an, die notwendig sind, um einen Wert, der am Eingang (IN) anliegt, in das ADR-Register zu schreiben!

d) Tragen Sie in der folgenden Tabelle 6.3 die benötigten Werte der Hardware-Eingangssignale für die folgenden zwei Schritte ein!

Schritt 1: IN-Register in OUT-Register schreiben
Schritt 2: X-Register in OUT-Register schreiben und **gleichzeitig** dazu den Wert der vom ADR-Register adressierten Speicherstelle in das IN-Register laden

	IN_LOAD	**ADR_ IN**	**ADR_ LOAD**	**ALE**	**READ**	**WRITE**	**TR_IN**	**X_LOAD**	**ADD**	**OUT_ LOAD**
Schritt 1										
Schritt 2										

Tabelle 6.3: Hardware-Eingangssignale

Aufgabe 7: **A/D- und D/A-Wandlung in Mikrocontrollern** **(10 Punkte)**

In Bild 7.1 ist das Blockschaltbild eines A/D-Wandlers mit sukzessiver Approximation dargestellt, der in *N* Taktschritten einen *N* bit breiten digitalen Ausgangswert erzeugt. In jedem Taktschritt wird bei diesem Wandler ein Bit des Ausgangswertes – beginnend mit dem MSB - bestimmt.

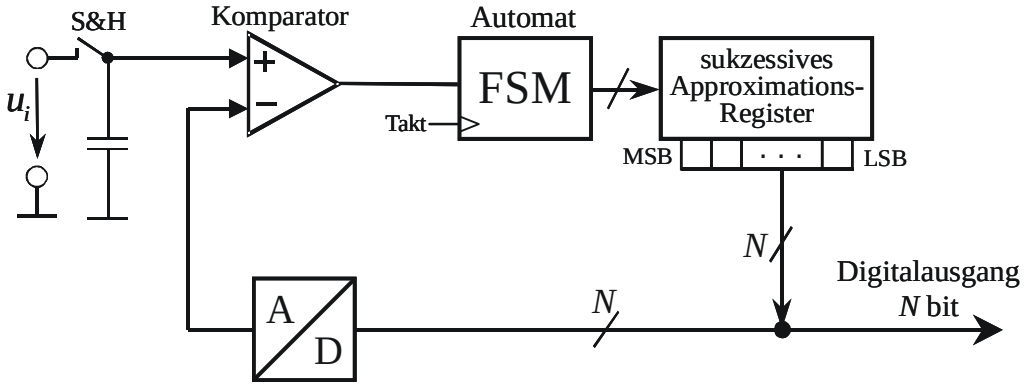

Bild 7.1: A/D-Wandler mit sukzessiver Approximation

Der Wandler soll im Folgenden ein Ausgangssignal der Breite 4 bit (*N*=4) für einen Eingangsspannungsbereich u_i=0...8 V liefern. Tabelle 7.1 gibt die Ausgangsspannung des D/A-Wandlers im Rückführungszweig in Abhängigkeit vom Inhalt des sukzessiven Approximationsregisters (SAR) an. Der SAR-Inhalt stellt gleichzeitig den digitalen Ausgangswert dar.

SAR-Inhalt	Spannung am –Eingang des Komparators
0000	0,0 V
0001	0,5 V
0010	1,0 V
0011	1,5 V
0100	2,0 V
0101	2,5 V
0110	3,0 V
0111	3,5 V
1000	4,0 V
1001	4,5 V
1010	5,0 V
1011	5,5 V
1100	6,0 V
1101	6,5 V
1110	7,0 V
1111	7,5 V

Tabelle 7.1: Ausgangsspannung des D/A-Wandlers in Abhängigkeit vom SAR-Inhalt

a) Tragen Sie in Tabelle 7.2 den Inhalt des SAR ein, der sich jeweils im Verlauf der 4 Wandelschritte für die angegebenen Eingangsspannungen einstellt! (Die Werte für Schritt 1 bei 2V sind als Gedächtnisstütze vorgegeben).

u_i →	**2V**	**6V**	**8V**
SAR-Inhalt	MSB LSB	MSB LSB	MSB LSB
Schritt 1 Anfang	1 0 0 0		
Schritt 1 Ende	0 0 0 0		
Schritt 2 Anfang			
Schritt 2 Ende			
Schritt 3 Anfang			
Schritt 3 Ende			
Schritt 4 Anfang			
Schritt 4 Ende			

Tabelle 7.2: Entwicklung der SAR-Inhalte für 3 verschiedene Eingangsspannungen

b) Warum kann der Eingangsspannungswert u_i=8 V nicht exakt dargestellt werden?

Im Weiteren wird die Pulsweitenmodulation (PWM) zur einfachen D/A-Wandlung betrachtet – siehe Bild 7.2. Bei der PWM wird ein Rechtecksignal der Höhe U mit konstanter Periode T und variablem Tastverhältnis t_i/T an einem Portpin eines Mikrorechners ausgegeben.
Zur Generierung des Rechtecksignals wird ein 16 bit-Binärzähler mit einer Taktfrequenz f_c verwendet, der periodisch den Zahlenbereich 0 bis 2^{16}-1 durchläuft. Die Pulsdauer t_i und damit das Tastverhältnis t_i/T wird mit Hilfe eines 16 bit-Vergleichsregisters festgelegt, dessen Inhalt mittels eines 16 bit-Komparators ständig mit dem Zählerstand verglichen wird.

In Bild 7.2 ist ein solches Ausgangssignal für 3 verschiedene Tastverhältnisse t_i/T dargestellt. Das Rechtecksignal wird durch ein RC-Glied (Tiefpassfilter) so geglättet, dass schließlich eine Gleichspannung U_A als Ausgangsgröße zur Verfügung steht.

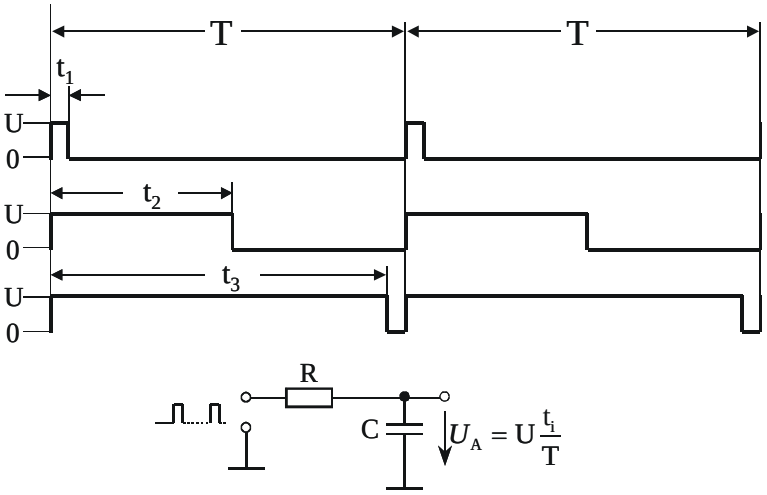

Bild 7.2: Pulsweitenmodulation der D/A-Wandlung

c) Geben Sie die Auflösung N (in Bit) des beschriebenen PWM-D/A-Wandlers an!

d) Welche Aktionen müssen jeweils am Portpin bei einem Überlauf des Binärzählers bzw. bei einer Übereinstimmung von Zähler- und Vergleichsregisterinhalt stattfinden, um die Signalverläufe nach Bild 7.2 zu erhalten?

e) Geben Sie für U=5 V den Inhalt des Vergleichsregisters in Dezimaldarstellung an, der nötig ist, um eine Ausgangsspannung von U_A=3 V zu erzeugen!

Aufgabe 8: **Mikrocontroller-Programmierung** **(10 Punkte)**

Ein 8 bit-Mikrocontroller vom Typ 8052 enthält als integrierte Peripheriekomponente einen 10 bit Analog/Digitalwandler (ADW), dessen Wandelergebnis in hexadezimaler Darstellung den Zahlenbereich von 000h...400h umfasst. Da mehr als 8bit benötigt werden, stehen zwei Spezialfunktionsregister, nämlich ADCDATAL für das untere Byte und ADCDATAH für die verbleibenden drei oberen Bits zur Verfügung.

Der Maximalwert der Eingangsspannung des ADW ist 1V, so dass 1V dem Wandelwert 400h entspricht.

Mittels eines Mikrorechnerprogramms ist der Wertebereich 000h...400h auf einen dem dezimalen Zahlenbereich 0...1000 entsprechenden Wertebereich 000h...3E8h zu skalieren, damit mittels Binär-BCD-Wandlung, die hier nicht betrachtet wird, eine Darstellung in der Einheit Millivolt erzielt wird.

Zur Skalierung ist der Wandelwert ADCDATAL/H durch 2^{10} zu dividieren und mit 1000 zu multiplizieren. Während eine Division durch 2^{10} durch geeignet verteiltes 10-maliges Rechtsschieben erfolgen kann, ist es vorteilhaft, die Multiplikation mit 1000 in die folgenden 3 Teilschritte zu zerlegen:

$$1000 = (4 \cdot 2{,}5) \cdot (4 \cdot 2{,}5) \cdot (4 \cdot 2{,}5)$$

Hieraus kann die Zweierpotenz $4^3=2^6=64$ extrahiert und mit der Division durch 2^{10} verrechnet werden, so dass schließlich $(\text{ADCDATAH/L} : 2^4) \cdot (2{,}5)^3$ zu berechnen ist.

Eine Multiplikation mit 2,5 ist vorteilhaft dadurch zu realisieren, dass der Multiplikand einmal links und einmal rechts geschoben wird und dann diese beiden Schiebeergebnisse addiert werden.
Die Berechnung von $(\text{ADCDATAH/L} : 2^4) \cdot (2{,}5)^3$ soll so durchgeführt werden, dass dreimal hintereinander ADCDATAH/L mit 2,5/2 multipliziert und das Endergebnis durch Rechtsschieben nochmal durch 2 geteilt wird.
Das obere Byte ADCDATAH werde ins Register R7 (nur die Bits 2 ... 0 sind gültig) und das untere Byte ADCDATAL ins Register R6 eingelesen.
Die Register R5 und R4 werden zur Aufnahme der rechtsgeschobenen (halbierten) Werte verwendet. Nach jeder Multiplikation mit 2,5/2 sind die Inhalte von R7, R6 mit dem Ergebnis überschrieben. Das ist zulässig, da die ursprünglichen Inhalte im weiteren nicht mehr benötigt werden.
Die Voreinstellungen des Mikrocontrollers einschließlich der ADW-Konfiguration sind als vorgegeben anzunehmen.

a) Vervollständigen Sie den folgenden Programmabschnitt an den mit **.....** markierten Stellen!

```
;****************** PROGRAMMAUSSCHNITT ****************************
MOV   R6,ADCDATAL      ;unteres Byte vom ADW
MOV   A,ADCDATAH       ;oberes Byte vom ADW
.....................;obere 5 Bits auf Null setzen
MOV   R7,A
              ;Das Unterprogramm SKALIERE wird dreimal aufgerufen
CALL  SKALIERE
CALL  SKALIERE
```

```
CALL SKALIERE
```

```
        ;abschließend ist noch durch 2 zu teilen
MOV     A,R7
RRC     A           ;rechtsschieben, wobei Bit 0 in Carry kommt
MOV     R7,A        ;oberes halbiertes Byte in R7 ablegen => Ergebnis
MOV     A,R6
RRC     A           ;rechtsschieben, wobei Carry in Bit 7 kommt
MOV     R6,A        ;unteres halbiertes Byte in R6 ablegen => Ergebnis
                    ;Ergebnis steht jetzt in R7 und R6
Halt:
JMP     Halt        ;endlose Warteschleife

          ;Unterprogramm Skalieren
SKALIERE:
          ;Halbieren
CLR     C           ; vorsorglich Carry löschen
MOV     A,R7
.......................  ;rechts rotieren,wobei Bit 0 in Carry kommt
.......................  ;oberes halbiertes Byte in R5 ablegen
MOV     A,R6
.......................  ;halbieren,wobei Carry in Bit 7 und Bit 0 in
                         ;Carry kommt
.......................  ;unteres halbiertes Byte in R4 ablegen
          ;Verdoppeln
CLR     C
MOV     A,R6
.......................  ;mal 2 nehmen, wobei Bit 7 in Carry und eine
                         ;Null in Bit 0 kommt
.......................  ;in R6 zurückspeichern (überschreiben)
MOV     A,R7
.......................  ;oberes Byte verdoppeln, wobei Carry aus R6 in
                         ;Bit 0 kommt
.......................  ;in R7 zurückspeichern (überschreiben)

          ;halben und doppelten Wert addieren
MOV     A,R4
CLR     C           ;vorsorglich
.....................;halbes und doppeltes unteres Byte addieren
MOV     R6,A        ;Ergebnis für unteres Byte in R6 ablegen
MOV     A,R5        ;halbiertes oberes Byte holen
..................... ;halbes und doppeltes oberes Byte addieren
                    ;mit Überlauf (C) aus dem unteren Byte
MOV     R7,A        ;Ergebnis für oberes Byte in R7 ablegen
```

```
          ;durch 2 teilen
CLR     C           ;vorsorglich
MOV     A,R7
RRC     A           ;rechtsschieben, wobei Bit 0 in Carry kommt
MOV     R7,A        ;oberes halbiertes Byte in R7 ablegen
MOV     A,R6
RRC     A           ;rechtsschieben, wobei Carry in Bit7 kommt
MOV     R6,A        ;unteres halbiertes Byte in R6 ablegen
RET

END
;****************** END of PROGRAMMAUSSCHNITT *********************
```

b) Welcher Hexadezimalwert steht in den Registern R7 und R6 nach der ersten Abarbeitung des Unterprogramms SKALIERE, wenn aus ADCDATAH/L der Wert 301h geliefert wird?

Aufgabe 9: **Berechnung von Skalarprodukten mit dem DSP** **(10 Punkte)**

Ein in der Signalverarbeitung häufig auftretender Algorithmus ist die Berechnung des Skalarprodukts zweier Vektoren. Hierfür ist die Architektur von digitalen Signalprozessoren besonders geeignet.

Gegeben seien die beiden *n* bit langen Zeilenvektoren: $x = [x_1 x_2 \cdots x_n],\ y = [y_1 y_2 \cdots y_n]$

Es soll das Skalarprodukt nach der Formel: $z = x \cdot y^T = \sum_{i=1}^{n} x_i \cdot y_i$

berechnet werden.

a) Welche beiden Befehle des DSP 56000 ermöglichen eine effiziente Berechnung von Skalarprodukten?

b) Geben Sie für die Berechnung des Skalarproduktes von zwei Vektoren mit je *n*=16 Elementen ein möglichst kompaktes Assemblerprogramm an! Die beiden Vektoren sind im X- und Y-Speicher jeweils ab Adresse $1000 abgelegt.

c) Um die Berechnung von Skalarprodukten mehrmals hintereinander auszuführen, ist das Einrichten von Ringspeichern sinnvoll. Mit welchen beiden Befehlen kann eine Initialisierung geeigneter Ringspeicher für die Vektoren aus Aufgabenteil b) erfolgen!

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

Aus einem Multiplizierer (**`mult`**) und einem Akkumulator (**`akku`**) soll durch Hintereinanderschalten eine MAC-Einheit aufgebaut werden. Gegeben sind die folgenden VHDL-Beschreibungen der beiden Module **`mult`** und **`akku`**.

```
ENTITY mult IS
 PORT(clock, reset: IN bit;
      x, y: IN Integer Range 0 to 63;
      e: OUT Integer Range 0 to 4095
      );
END mult;

ARCHITECTURE behave OF mult IS
BEGIN
 PROCESS (clock, reset)
 BEGIN
  IF reset='1' THEN
    e <= 0;
  ELSIF(clock'event AND clock='1')THEN
    e <= x * y;
  END IF;
 END PROCESS;
END behave;
```

```
ENTITY akku IS
 PORT(clock, reset, clear: IN bit;
     ein: IN Integer Range 0 to 4095;
     sum: OUT Integer Range 0 to 32767
      );
END akku;

ARCHITECTURE behave OF akku IS
BEGIN
 PROCESS (clock, reset)
 BEGIN
  IF reset='1' THEN
    sum <= 0;
  ELSIF(clock'event AND clock='1')THEN
    IF clear='1' THEN
      sum <= ein;
    ELSE
      sum <= sum + ein;
    END IF;
  END IF;
 END PROCESS;
END behave;
```

a) Tragen Sie im untenstehenden Blockdiagramm die Ein- und Ausgangssignale der entstehenden MAC-Einheit ein und geben Sie jeweils die Anzahl der Bits bzw. der benötigen Portleitungen an!

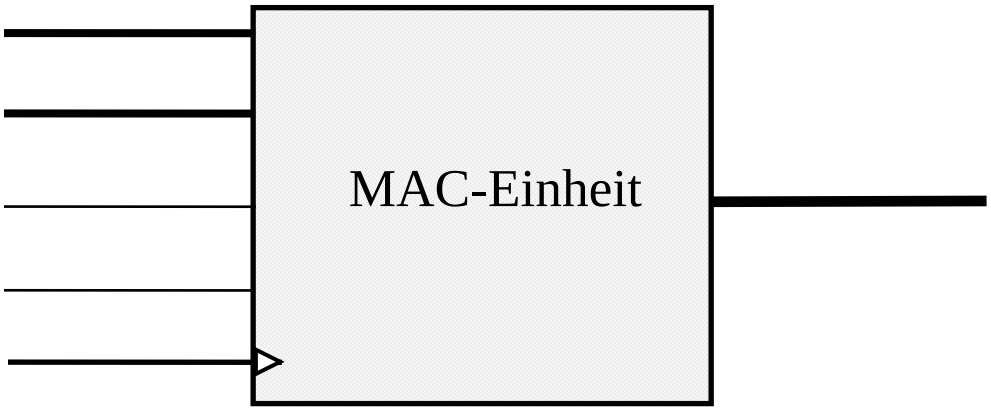

Bild 10.1: Blockdiagramm einer MAC-Einheit

b) Vervollständigen Sie die nachfolgende ENTITY mac_eh und die zugehörige ARCHITECTURE so, dass eine MAC Einheit entsteht!

```
LIBRARY IEEE;
USE ieee.std_logic_vector_1164.all;
ENTITY mac_eh IS
  PORT (




                                                    );
END Automat;


ARCHITECTURE behave OF mac_eh IS

   SIGNAL zwischen:                                 ;

   COMPONENT mult IS
   PORT(clock, reset: IN bit;
        x, y: IN  Integer Range 0 to 63;
        e   : OUT Integer Range 0 to 4095);

   COMPONENT akku IS
   PORT(clock, reset, clear: IN bit;
        ein: IN  Integer Range 0 to 4095;
        sum: OUT Integer Range 0 to 32767);

BEGIN

   Comp1: mult PORT MAP(                                );

   Comp2: akku PORT MAP(                                );

END behave;
```

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