

**Vorname:** \_\_\_\_\_

**Name:** \_\_\_\_\_

**Matr.-Nr.:** \_\_\_\_\_

**Note:** \_\_\_\_\_

04.10.2007, 14.00 Uhr - 16.00 Uhr

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

**Vordiplomprüfung im Fach**

***Mikrorechnertechnik***

Die Prüfung umfasst **10 Aufgaben auf 23 Seiten**.

Bitte schreiben Sie auf **alle** Lösungsblätter Ihren Namen und Ihre Matrikelnummer.

Geben Sie bei allen Aufgaben den kompletten Lösungsweg an! Die alleinige Nennung des Endresultats ist zur Erlangung der vollen Punktzahl einer Aufgabe nicht ausreichend.

Die Verwendung eigenen Papiers ist nicht erlaubt.

Bei Bedarf kann zusätzliches Schreibpapier bei der Aufsicht angefordert werden.

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	9	11	10	10	11	9	10	10	100

## Aufgabe 1: Parallele Multiplikation

10 Punkte

In Abbildung 1.1 ist ein speicherbasierter Multiplizierer für zwei vorzeichenlose 4 bit-Binärzahlen skizziert.

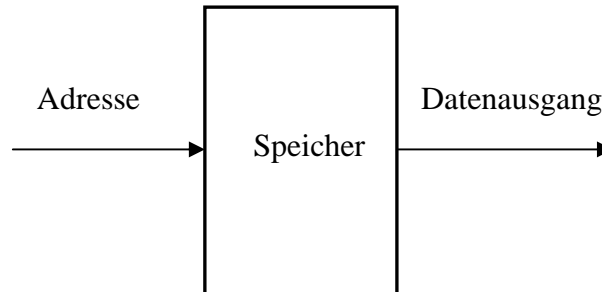


Abbildung 1.1: Speicherbasierter Multiplizierer für 4 bit-Binärzahlen

- 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	
0011	
0100	
0101	
0110	
0111	
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	

Tabelle 1.1: Adresse und Speicherinhalt

## Fortsetzung der 1. Aufgabe:

Im Folgenden wird die Arbeitsgeschwindigkeit eines Parallelmultiplizierers für zwei 8-bit-Zweierkomplementzahlen  $A$  und  $B$  betrachtet, der gemäß Gleichung 1.1 ein 16 bit langes Produkt  $P$  als Ergebnis liefert:

$$P = A \cdot B = -2^{15} + 2^8 + a_7 \cdot b_7 \cdot 2^{14} + \sum_{j=0}^6 \sum_{i=0}^6 a_i \cdot b_j \cdot 2^{i+j} + \sum_{i=0}^6 \overline{a_7 b_i} \cdot 2^{i+7} + \sum_{i=0}^6 \overline{b_7 a_i} \cdot 2^{i+7} \quad (\text{Gleichung 1.1})$$

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  $1,5 \text{ ns}$ .

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 Abschlussaddition.

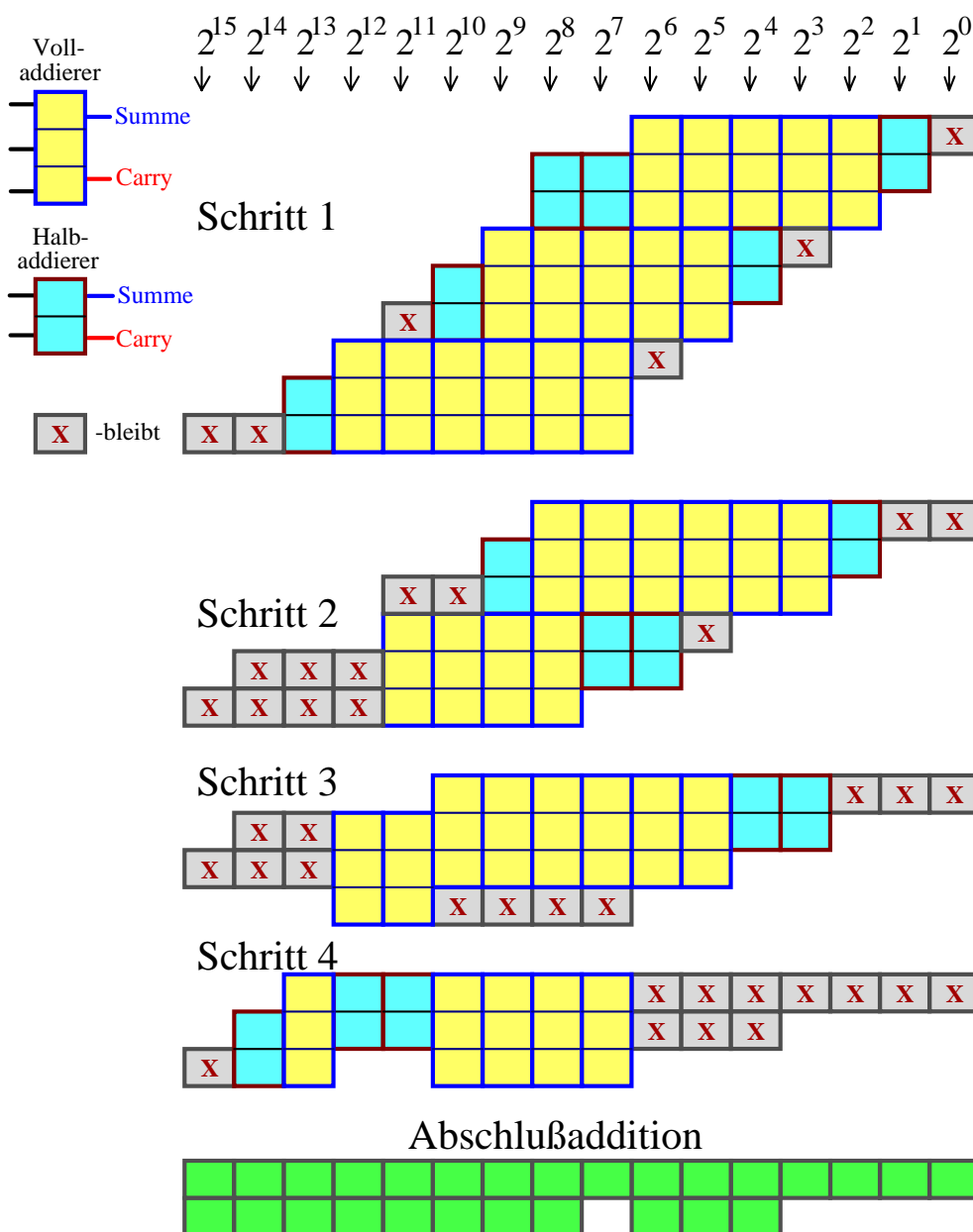


Abbildung 1.2: Schematische Darstellung der Abarbeitung der Multiplikation nach Gleichung 1.1 bis zur Abschlussaddition mit Hilfe von 15 Halb- und 39 Volladdierern in 4 Schritten

**Fortsetzung der 1. Aufgabe:**

- c) Wie lange dauert die Ausführung einer Multiplikation, wenn ein Volladdierer und der Carry-Look-Ahead-Addierer für die Abschlussaddition jeweils  $2,5\text{ ns}$  Durchlaufzeit haben?

Jetzt wird nach jedem der obigen Schritte 1...4 ein Pipelining-Register eingebaut. Alle Register weisen eine Durchlaufzeit von  $1\text{ ns}$  auf.

- d) Mit welcher maximalen Taktfrequenz  $f_{\max}$  können die Pipelining-Register betrieben werden, damit, nachdem die gesamte Pipeline gefüllt ist, mit jedem Taktschritt ein korrektes 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 Zweierkomplement-Darstellung verwendet.

In Tabelle 2.1 ist eine Anfangsbelegung des Registers R1 in binärer 8 bit-Zweierkomplement-Darstellung und des Registers R0 in Dezimaldarstellung angegeben.

Register	Inhalt (binär)								Inhalt (dezimal)
	MSB							LSB	
R0									-21
R1	1	0	0	1	1	0	1	0	
A									

Tabelle 2.1: Registerbelegung eines Mikrocontrollers

- a) Tragen Sie in Tabelle 2.1 den Inhalt des Registers *R0* in binärer Zweierkomplementdarstellung sowie den Inhalt des Registers *R1* in Dezimaldarstellung ein!

Der im Akkumulator stehende Wert sei durch eine Subtraktion des Registers *R0* von Register *R1* zu berechnen.

- b) Berechnen Sie aus den Inhalten der Register *R0* und *R1* den Akkumulatorinhalt und tragen Sie ihn in Tabelle 2.1 sowohl in Dezimal- als auch in binärer Zweierkomplementdarstellung ein!
- c) Geben Sie den Dezimalwert des Akkumulatorinhalts an, wenn dieser als Festkommazahl mit 4 Stellen nach dem Komma interpretiert wird!
- d) Geben Sie den Dezimalwert des Inhalts von Registers *R1* an, wenn dieser als Fraktalzahl interpretiert wird!

**Fortsetzung der 2. Aufgabe:**

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

- e) Ergänzen Sie in Tabelle 2.2 in der ersten Zeile für die Zahl 123 die 32 bit-Gleitkommazahl und in der zweiten Zeile den Dezimalwert der angegebenen Gleitkommazahl! (Hinweis: Die binäre Darstellung entspricht dem IEEE-P754 Single-Precision-Standard).

[illegible]

Tabelle 2.2: Speicherbelegung eines digitalen Signalprozessors

### Aufgabe 3: MOS-Technologie

9 Punkte

- a) Skizzieren Sie den Schaltplan eines CMOS-Inverters. Kennzeichnen Sie *Gate*, *Drain* und *Source* an den verwendeten Transistoren! Kennzeichnen Sie, welche Art von Transistoren Sie verwendet haben (n-Kanal/p-Kanal)! Erläutern Sie die Funktionsweise des Inverters in Stichworten!

- b) Benennen Sie die Anschlüsse  $A$ - $F$  ihrer Funktion entsprechend mit *Gate*, *Drain* und *Source*!



#### Aufgabe 4: CMOS-Transfergates

11 Punkte

Die CMOS-Technologie ermöglicht den Aufbau besonderer Schaltungsstrukturen mit Hilfe von Transfergates.

- a) In Abbildung 4.1 sind vier Transfergates, ein Treiber sowie zwei Inverter gegeben. Vervollständigen Sie die Schaltung derart, dass Sie ein vorderflankengetriggertes D-Flip-Flop erhalten! Verwenden Sie hierzu die Steuersignale  $T$  und  $\bar{T}$ , ein Eingangssignal  $D$  sowie die Ausgangssignale  $Q$  und  $\bar{Q}$ !

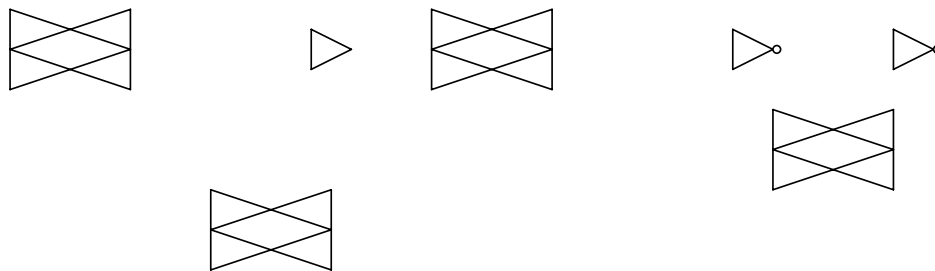


Abbildung 4.1: Transfergates, Treiber, Inverter zum Aufbau eines D-Flip-Flops

- b) Bezeichnen Sie die Zustände der 4 Transfergates (offen, gesperrt) innerhalb Ihres vorderflankengetriggerten D-Flip-Flops für den Fall  $T = 0$ .

(**Hinweis:** Bezeichnen Sie dabei die Transfergates beginnend von links mit den Namen  $TG1$  bis  $TG4$ !)

**Fortsetzung der 4. Aufgabe:**

- c) An den Eingängen eines rückflankengetriggerten D-Flip-Flops liegen nun ein Takt  $T$  sowie das Datensignal  $D$  gemäß Abbildung 4.2 an. Vervollständigen Sie das Timingdiagramm in Abbildung 4.2, indem Sie die resultierenden Signale  $Q$  und  $\bar{Q}$  einzeichnen!

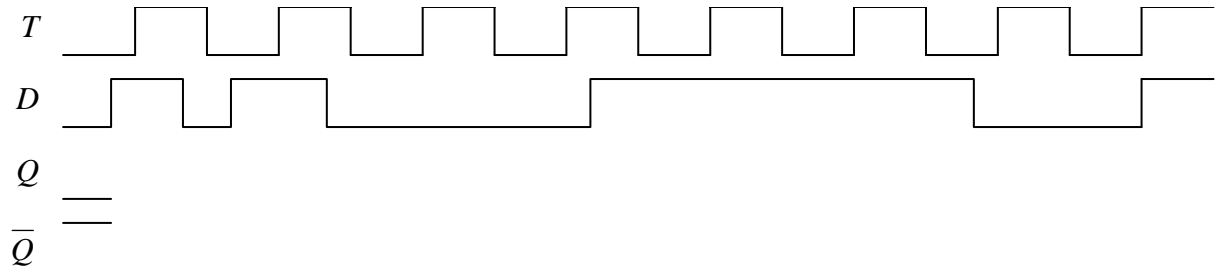


Abbildung 4.2: Timingdiagramm eines rückflankengetriggerten D-Flip-Flops

## Aufgabe 5: Steuerwerksbeschreibung in Form einer FSM

10 Punkte

In Bild 5.1 ist der Zustandsgraph des Steuerwerks einer einfachen programmgesteuerten Maschine abgebildet. Das Steuerwerk realisiert die in Tabelle 5.1 aufgeführten Befehle. Als Arbeitsregister stehen ein Akkumulator (A) und ein Eingaberegister (INR) zur Verfügung. Das Signal *INIT* ist das Hardware-Resetsignal.

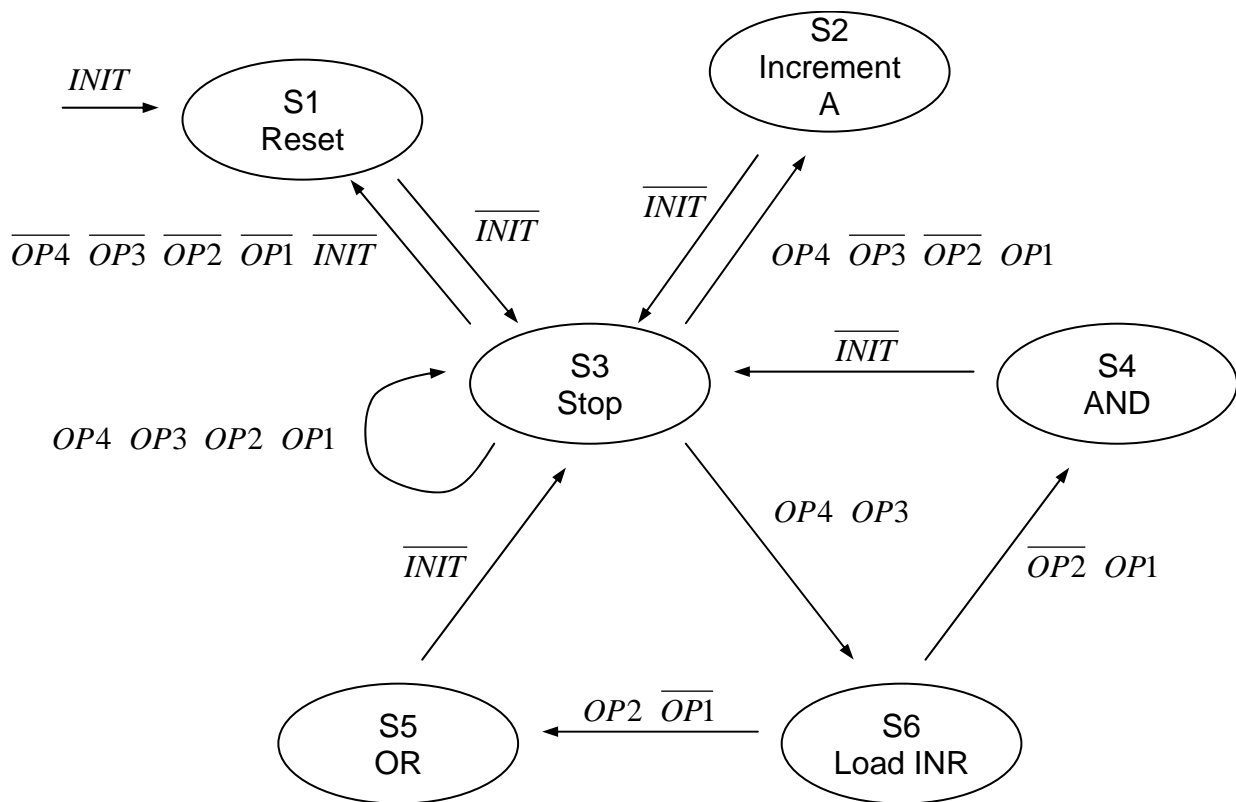


Abbildung 5.1: Zustandsgraph des Steuerwerks

- a) Ordnen Sie den Befehlen in Tabelle 5.1 die jeweilige Binärkombination (*OP4*, *OP3*, *OP2*, *OP1*), d.h. die OP-Codes zu!

Befehl	OP4	OP3	OP2	OP1	Beschreibung
INC A					increment A
ANDL A, INR					A=(INR) AND (A)
ORL A, INR					A=(INR) OR (A)
NOP					no operation

Tabelle 5.1: Befehle einer programmgesteuerten Maschine

### Fortsetzung der 5. Aufgabe:

In Abbildung 5.2 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 5.2 ist die Zuordnung einiger Zustände zu den Steuersignalen der Mikrosequenzer-Hardware aufgeführt.

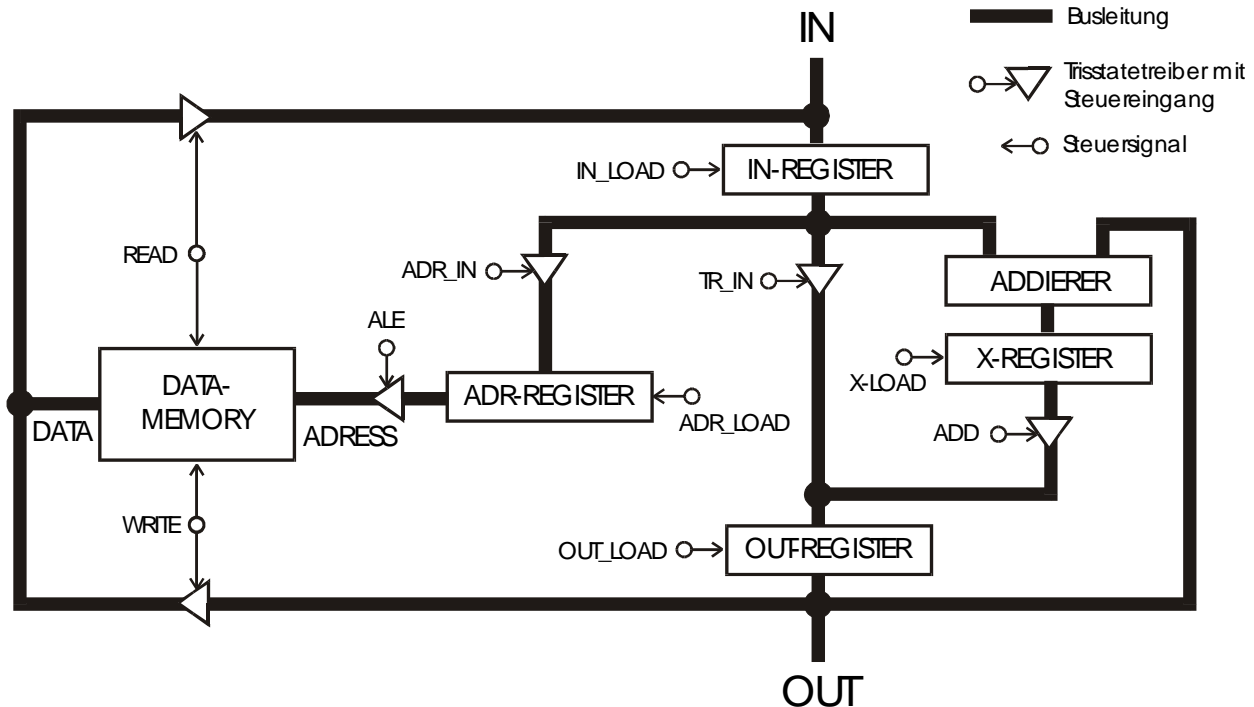


Abbildung 5.2: Mikrosequenzer-Hardware auf Registertransferebene

Zustand	IN_LOAD	ADR_IN	ADR_LOAD	ALE	READ	WRITE	TR_IN	X_LOAD	ADD	OUT_LOAD
S <sub>0</sub>	0	1	1	0	0	0	0	0	0	0
S <sub>1</sub>	0	0	0	1	0	1	0	0	0	0
S <sub>2</sub>	0	0	0	0	0	0	0	1	0	0
S <sub>3</sub>	1	0	0	0	0	0	0	0	0	0
S <sub>4</sub>	0	0	0	0	0	0	0	0	0	0

Tabelle 5.2: Zuordnung der Zustände zu den Steuersignalen

- b) Geben Sie die Folge von Zuständen  $S_i \dots S_j$  an, die notwendig sind, um einen Wert der am Eingang (IN) anliegt, in das ADR-Register zu schreiben (entspricht dem Befehl *MOV ADR, IN*)!

**Fortsetzung der 5. Aufgabe:**

- c) Geben Sie die Folge von Zuständen  $S_i \dots S_j$  an, die notwendig sind, um einen Wert, der im *OUT*-Register liegt, in das *DATA-MEMORY* an die Adresse, die am Eingang *IN* anliegt, zu schreiben!

- d) Tragen Sie in der folgenden Tabelle 5.3 die benötigten Werte der Hardware-Steuersignale für die folgenden beiden 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 des *DATA-MEMORY* in das *IN*-Register schreiben.

	IN_LOAD	ADR_IN	ADR_LOAD	ALE	READ	WRITE	TR_IN	X_LOAD	ADD	OUT_LOAD
<b>Schritt 1</b>										
<b>Schritt 2</b>										

Tabelle 5.3: Zuordnung der Zustände zu den Steuersignalen

## Aufgabe 6: Assemblerprogrammierung eines 80C51

10 Punkte

Im den folgenden Teilaufgaben sollen mittels der 8051-Assemblersprache kurze, einfache Programmabschnitte entwickelt werden, die leicht abgewandelt beim Entwurf komplexer Programme einsetzbar sind.

(**Hinweis:** Verwenden Sie ausschließlich die Assemblerbefehle, die in Tabelle 6.1 gegeben sind!)

CLR C	Carry löschen -> CLR bit ; bit löschen
INC Rn	Register <i>n</i> inkrementieren -> INC A ; Akku inkrr.
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,Rn	Subtraktion mit „Borgen“, $A := A - Rn - C$
JB bit,Adresse	Sprung an Adresse, wenn 'bit' gesetzt ist
JMP Adresse	unbedingter Sprung an Adresse
NOP	keine Operation
RET	Rücksprung aus einem Unterprogramm
RRC A	Akku nach rechts durch Carry schieben -> RLC A; nach links
RL A	Akku nach links schieben aber nicht durch Carry (vgl. RRC)
JNB bit,Adresse	Sprung an Adresse, wenn 'bit' nicht gesetzt ist
ADD A,source	Inhalt von A um den Inhalt von 'source' erhöhen

Tabelle 6.1: Befehlsauszug der 8051-Assemblersprache

- a) Schreiben Sie ein Unterprogramm *Sub35*, das vom Inhalt des Registers R0 den Wert 35 (dezimal) subtrahiert! Verwenden Sie zur Berechnung den Akkumulator A!

**Fortsetzung der 6. Aufgabe:**

- 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!

Zur Übertragung von Daten über die serielle Schnittstelle eines 80C51 an einen PC wird das Senderegister *SBUF* verwendet.

- c) Schreiben Sie ein Unterprogramm *Send*, das so lange wartet, bis das Interruptflag *TI* gesetzt ist, dieses anschließend löscht und den Akkumulatorinhalt in das Senderegister *SBUF* verschiebt!

### Aufgabe 7: Serielle Schnittstelle des 80C51

11 Punkte

Unter Verwendung des integrierten Timers 1 bietet der Mikrocontroller 80C51 vielfältige Möglichkeiten der Baudratengenerierung für die asynchrone serielle Datenübertragung. Dazu soll im Folgenden die Betriebsart 1 der seriellen Schnittstelle verwendet werden. Diese erlaubt die asynchrone Datenübertragung mit variabler Baudrate, wobei als Zeitbasis die Überlaufrate von Timer 1 dient.

- a) Erläutern Sie die Begriffe „Simplex-“, „Halbduplex-“ und „Vollduplexbetrieb“ im Zusammenhang mit der asynchronen seriellen Datenübertragung!

Die Taktfrequenz des Mikrocontrollers betrage 10 MHz. Um die Baudratengenerierung mit möglichst geringer Prozessorbelastung durchzuführen, soll Timer 1 im „Autoreload-Mode“ betrieben werden. Die serielle Schnittstelle werde in der Betriebsart (Mode) 1 betrieben.

- b) Berechnen Sie die Baudrate in Abhängigkeit vom Wert des Steuerbits SMOD, falls das Reloadregister *TH1* von Timer 1 den Wert 65 (hexadezimal) enthält!



---

**Fortsetzung der 7. Aufgabe:**

- c) Geben Sie in Tabelle 7.1 die notwendigen Einstellungen der Spezialfunktionsregister *SCON*, *TCON*, *TMOD* und *IE* an und kennzeichnen Sie irrelevante Bits durch „X“!

	Bit 7				Bit 0			
SCON								
TCON								
TMOD								
IE								

Tabelle 7.1: Initialisierung der Register *SCON*, *TCON*, *TMOD* und *IE*

- d) Berechnen Sie die maximale Baudrate, die in der angegebenen Betriebsart (Schnittstelle in Mode 1, Timer 1 im Autoreload-Mode) generiert werden kann!

## Aufgabe 8: Digitale Signalprozessoren

9 Punkte

Der DSP 56000 zeichnet sich unter anderem durch vielseitige Möglichkeiten der Registeradressierung aus. Tabelle 8.1 gibt einen Auszug aus der Register- und Speicherbelegung des DSP an. Ausgehend von dieser Speicherbelegung werden die folgenden 3 Befehle ausgeführt:

```
MOVE  X:(R0)+, X0      Y:(R4)+N4, Y0 ; Befehl 1
MOVE  X:(R0+N0), X0    Y:R4, Y0      ; Befehl 2
MOVE  X:R0+N0, X0      Y:-(R4), Y0   ; Befehl 3
```

Register	Inhalt		Register	Inhalt		Adresse	X-Speicher	Y-Speicher
R0:	\$0002		A2:	\$00		\$0005	\$600000	\$A60000
M0:	\$0003		A1:	\$200000		\$0004	\$500000	\$B50000
N0:	\$0002		A0:	\$000000		\$0003	\$400000	\$C40000
R4:	\$0000					\$0002	\$300000	\$D30000
M4:	\$0003		X0:	\$400000		\$0001	\$200000	\$E20000
N4:	\$0004		Y0:	\$400000		\$0000	\$100000	\$F10000

Tabelle 8.1: Auszug aus der Anfangsbelegung von Registern und Speichern

- a) Geben Sie in Tabelle 8.2 die Registerbelegung nach der Ausführung der 3 Befehle an!

Register	Inhalt nach Befehl 1	Inhalt nach Befehl 2	Inhalt nach Befehl 3
R0			
X0			
R4			
Y0			

Tabelle 8.2: Registerbelegung nach Ausführung der Befehle

- b) Benennen Sie in der nachfolgenden Tabelle 8.3 für jeden der 3 Befehle die jeweils verwendete Adressierungsart, sowohl für den X-, als auch für den Y-Speicher!

Befehl	Adressierungsart X-Speicher	Adressierungsart Y-Speicher
1		
2		
3		

Tabelle 8.3: Adressierungsarten bei den Befehlen 1-3

## Aufgabe 9: Logikbeschreibung mit VHDL

10 Punkte

Gegeben sei der folgende VHDL-Code, der näher untersucht werden soll.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity exam is
end exam;

architecture exam_arch of exam is
signal clk1 : std_logic;
signal reset1: std_logic;
signal a1, b1: integer range 0 to 128;
signal c1 : integer range 0 to 256;

component examination
port (clk : in std_logic;
      reset : in std_logic;
      a : in std_logic;
      b : in integer range 0 to 128;
      c : out integer range 0 to 256);
end component;

for all: examination
use entity work.examination(examination_arch);

begin

exam_comp : examination
port map (clk1, reset1, a1, b1, c1);

process
begin
clk1 <= '1';
wait for 10 ns;
clk1 <= '0';
wait for 10 ns;
end process;

reset1 <= '0',
        '1' after 15 ns;

a1 <= 0,
    21 after 60 ns,
    5 after 100 ns;
```

**Fortsetzung der 9. Aufgabe:**

```
b1 <= 0,  
      3 after 40 ns,  
      7 after 80 ns,  
      16 after 100 ns,  
      0 after 120 ns;  
  
end exam_arch;
```

- a) Die Entity des VHDL-Modells ist leer. Welches Konstrukt wird damit mittels VHDL erzeugt?
- b) Vervollständigen Sie im nachstehenden Timingdiagramm in Abbildung 9.1 die Zeitverläufe der Signale *clk1*, *reset1*, *a1* und *b1* anhand des VHDL-Codes!

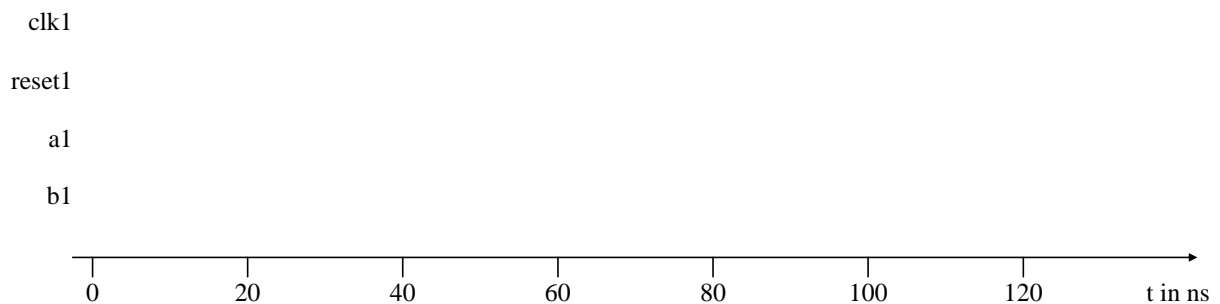


Abbildung 9.1: Timingdiagramm der Zeitverläufe

Im Folgenden soll die Funktion der im obigen VHDL-Code eingebundenen Komponente *examination* entwickelt werden. Hierbei sollen die Eingangssignale *a* und *b* jeweils mit der fallenden Taktflanke des Taktes *clk* addiert werden. Das Ergebnis soll dem Signal *c* zugewiesen werden. Bei einem Low-Pegel des Signals *reset* sollen die Signale *a*, *b* und *c* jeweils den Wert Null zugewiesen bekommen.

- c) Vervollständigen Sie die nachstehende Architektur und den Prozess, um die geforderte Funktionalität bereitzustellen!

```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.std_logic_arith.all;  
use ieee.std_logic_unsigned.all;  
  
entity examination is  
port (clk    : in  std_logic;  
      reset  : in  std_logic;  
      a      : in  std_logic;  
      b      : in  integer range 0 to 128;  
      c      : out integer range 0 to 256);  
end examination;
```

**Fortsetzung der 9. Aufgabe:**

architecture examination\_arch of examination is

begin

process( )

end process;

end examination\_arch;

## Aufgabe 10: Analyse von VHDL-Modellen

10 Punkte

VHDL stellt neben Signalen auch Variablen zur Verfügung. Im Folgenden sollen zwei VHDL-Codes auf ihr Zeitverhalten in entsprechenden Simulationen untersucht werden.

- a) Vervollständigen Sie Tabelle 10.1, indem Sie den nachfolgenden VHDL-Code untersuchen und die jeweils resultierenden Werte der Variablen in die Tabelle eintragen!  
(**Hinweis:** Die Variable  $x$  wird extern entsprechend Tabelle 10.1 mit einer Taktrückflanke gesetzt und dem Prozess übergeben!)


```
process (clk)

variable y1, y2, y3: integer;

begin

    if (clk'event and clk='0') then
        y1 := x;
        y2 := y3+2;
        y3 := y1+3;
    end if;

end process;
```



clk						
x	6	20	1	7	13	
y1						
y2						
y3						

Tabelle 10.1: Timingdiagramm unter Verwendung von Variablen

- b) Vervollständigen Sie Tabelle 10.2, indem Sie den nachfolgenden VHDL-Code untersuchen und die jeweils resultierenden Werte der Signale in die Tabelle eintragen!  
(**Hinweis:** Das Signal  $x$  wird extern entsprechend Tabelle 10.2 mit einer Taktrückflanke gesetzt und dem Prozess übergeben!)

```
signal y1, y2, y3: integer;

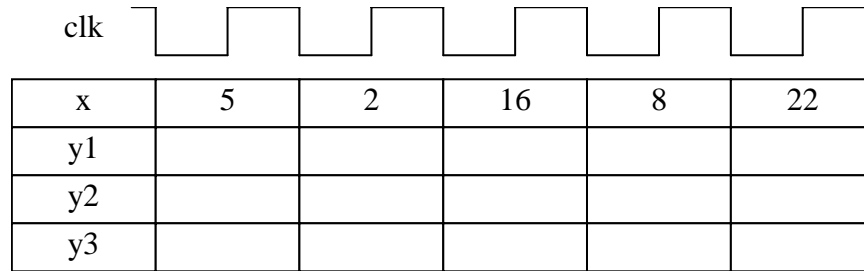
process (clk)

begin

    if (clk'event and clk='0') then
        y1 <= x;
        y2 <= y3+2;
        y3 <= y1+3;
    end if;

end process;
```

**Fortsetzung der 10. Aufgabe:**



clk						
x	5	2	16	8	22	
y1						
y2						
y3						

Tabelle 10.2: Timingdiagramm unter Verwendung von Signalen

- c) Nennen Sie drei Objekte, die VHDL zur Verfügung stellt!
- d) Nach welcher Anweisung kann in VHDL eine „Sensitivity-Liste“ folgen? Wozu wird diese Liste verwendet?
- e) Was wird mit VHDL beschrieben, d.h. wozu wird diese Sprache vorzugsweise eingesetzt? Welche Hauptunterschiede gibt es im Vergleich zu herkömmlichen Programmiersprachen wie C oder Pascal?