Vorname: _____ **Name:** _____

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

18.03.2005 - 14^{00} Uhr bis 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 25 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:	9	10	12	9	9	10	10	10	10	11	100

Aufgabe 1: A/D-Wandlung — 9 Punkte

Zunächst wird ein A/D-Wandler betrachtet, der nach dem Flash-Prinzip arbeitet.

a) Verbinden Sie die in Abbildung 1.1 dargestellten Blöcke des Wandlers mit einem Eingangssignal U_i, einer Referenzspannung U_{ref} und einem digitalen Ausgang D_a! Vervollständigen Sie die Schaltung des Wandlers! Ergänzen Sie weiterhin die Polaritäten der Komparatoren und beschriften Sie die Codierlogik am Ausgang!

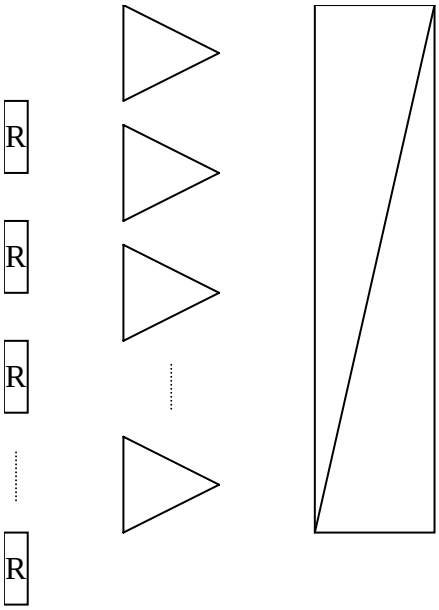

Abbildung 1.1: Blockschaltbild eines Flash-Wandlers

b) Der Flash-Wandler sei mit 1024 Komparatoren aufgebaut. Welche Auflösung N in bit hat der Wandler?

Fortsetzung der 1. Aufgabe:

Neben dem Flash-Prinzip gibt es weitere Verfahren zur A/D-Wandlung.

c) Nennen Sie 4 weitere Verfahren und erklären Sie diese kurz in Stichworten!

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 R0 in binärer 8 bit-Zweierkomplement-darstellung, sowie der Inhalt des Akkumulators in Dezimaldarstellung angegeben.

Register	**Inhalt (binär)** MSB LSB								**Inhalt (dezimal)**
R0	0	0	1	0	1	1	0	0	
R1									
A									-53

Tabelle 2.1: Registerbelegung eines Mikrocontrollers

a) Tragen Sie in Tabelle 2.1 den Inhalt des Registers R0 in Dezimaldarstellung, sowie den Inhalt des Akkumulators in binärer Zweierkomplementdarstellung ein!

Der im Akkumulator stehende Wert sei durch eine Addition der Register R0 und R1 berechnet worden.

b) Berechnen Sie aus den Werten des Registers R0 und des Akkumulators den Registerwert R1 und tragen Sie ihn in Tabelle 2.1 in Dezimal- und binärer Zweierkomplementdarstellung ein!

c) Geben Sie den Dezimalwert des Akkumulators an, wenn dieser als Festkommazahl mit 5 Stellen nach dem Komma interpretiert wird!

d) Geben Sie den Dezimalwert des Akkumulators 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 58 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).

Inhalt (dezimal)	MSB **Inhalt (binär)** LSB																													
	31							24	23							16	15							8	7							0
58																														
	1	1	0	0	0	1	0	0	1	1	0	1	0	0	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: Beschreibung einer FSM — 12 Punkte

In Tabelle 3.1 sind die Binärkombinationen einer einfachen programmgesteuerten Maschine gegeben. Das Steuerwerk realisiert die aufgeführten Befehle. Als Arbeitsregister stehen ein Akkumulator (A) und ein Eingaberegister (INR) zur Verfügung. Das Signal INIT ist das Hardware-Resetsignal.

Befehl	OP4	OP3	OP2	OP1	Beschreibung
INC A	1	0	0	1	increment A
ANDL A, INR	1	1	0	0	A=(INR) AND (A)
ORL A, INR	1	1	1	0	A=(INR) OR (A)
NOP	0	1	1	0	no operation

Tabelle 3.1: Realisierte Befehle

a) Vervollständigen Sie den Zustandsgraphen des Steuerwerks in Abbildung 3.1 um die noch fehlenden Zustandsübergänge! Geben Sie jeweils die OP-Codes in der in Beispiel 3.1 dargestellten Form an!

Beispiel 3.1: $\overline{OP4}\ \overline{OP3}\ \overline{OP2}\ \overline{OP1}\ \overline{INIT}$

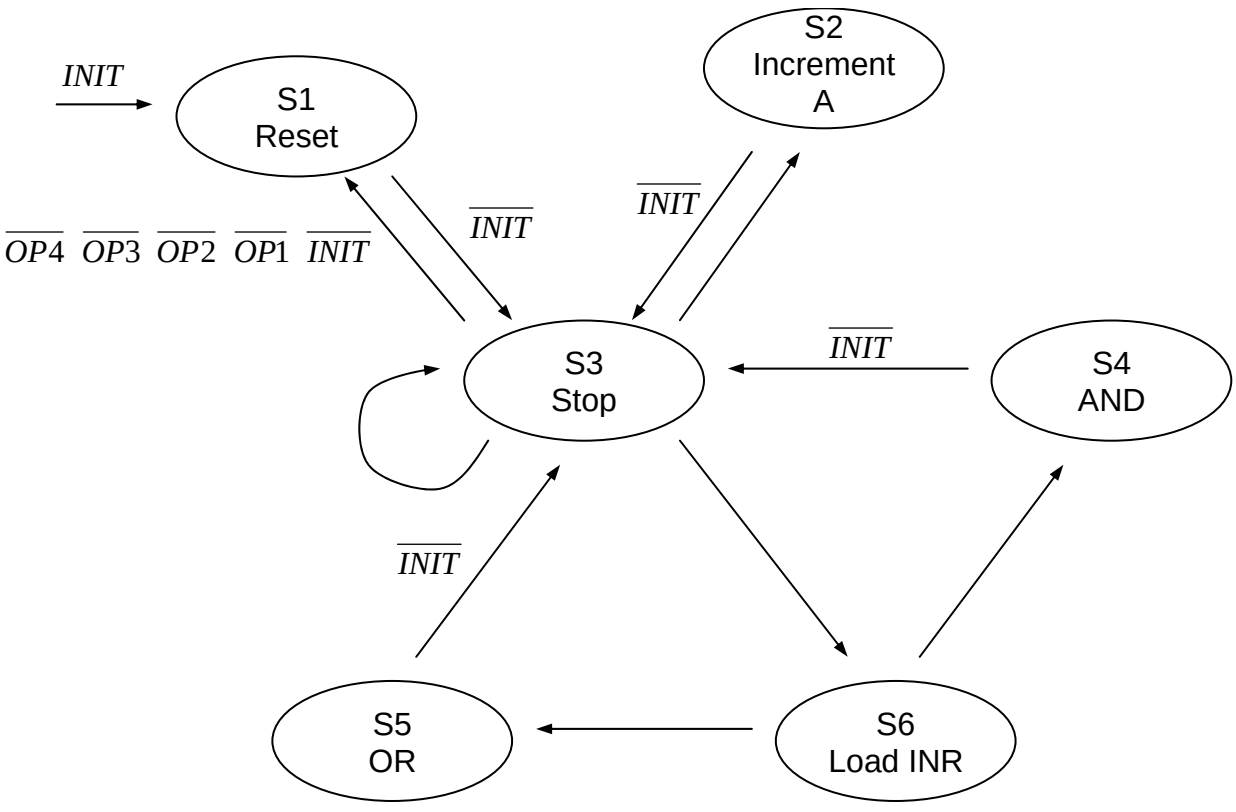

Abbildung 3.1: Zustandsgraph des Steuerwerks

Fortsetzung der 3. Aufgabe:

In Abbildung 3.2 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 3.2 ist die Zuordnung einiger Zustände zu den Eingängen der Steuersignale der Mikrosequenzer-Hardware aufgeführt.

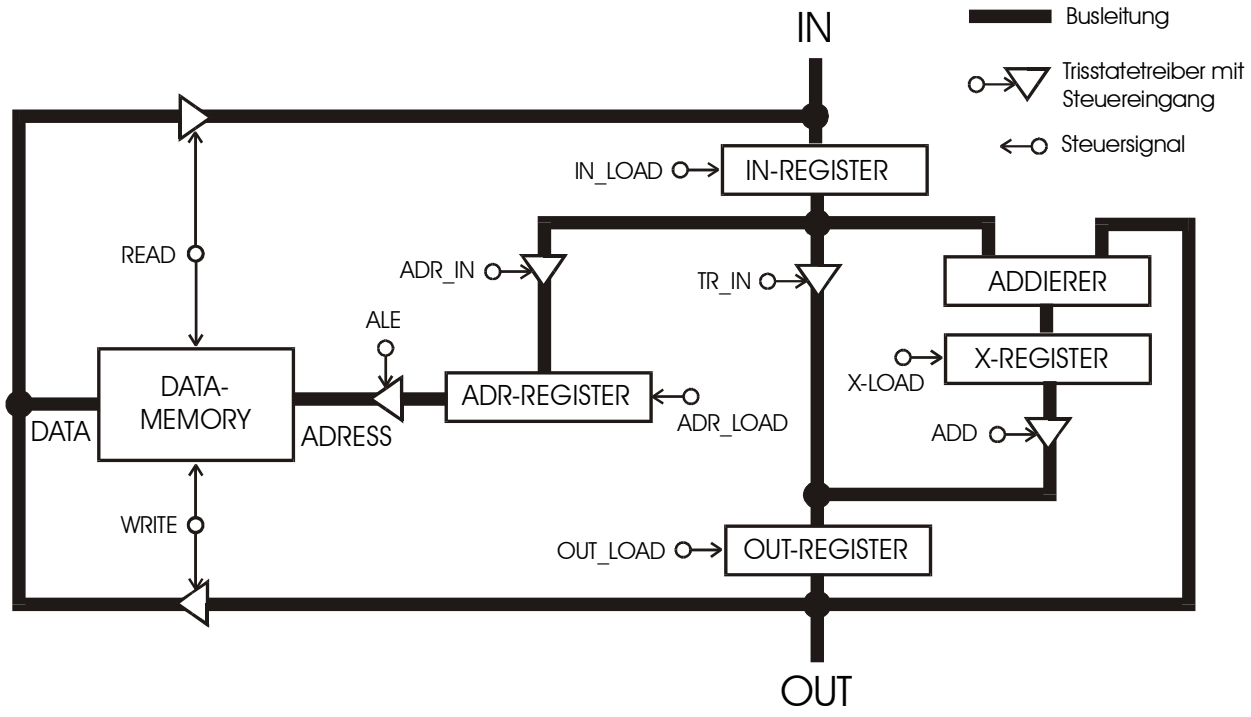

Abbildung 3.2: Mikrosequenzer auf Registertransferebene

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

Tabelle 3.2: Zuordnung der Zustände zu den Eingangssignalen

b) Geben Sie die Folge von Zuständen $S_i \ldots S_j$ 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 3. Aufgabe:

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

d) Tragen Sie in der folgenden Tabelle 3.3 die benötigten Werte der Hardware-Steuersignale für die folgenden zwei Schritte ein:

Schritt 1: *IN*-Register in *OUT*-Register schreiben!
Schritt 2: *X*-Register in *OUT*-Register schreiben und gleichzeitig 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 3.3: Hardware-Steuersignale

Aufgabe 4: CMOS-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!

Fortsetzung der 4. Aufgabe:

In Abbildung 4.1 ist nun der Aufbau eines CMOS-Inverters auf Silizium im Querschnitt dargestellt.

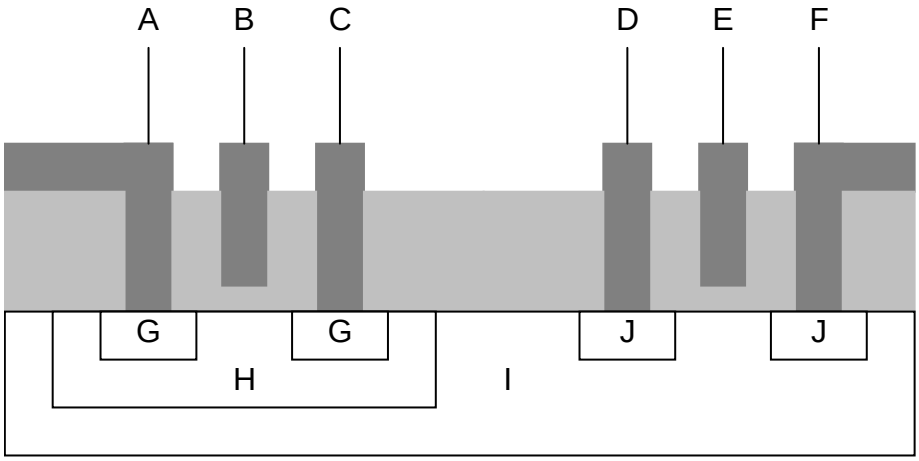

Abbildung 4.1: Technologischer Aufbau eines CMOS-Inverters

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

c) Wie sind die Bereiche *G, H, I* und *J* jeweils dotiert (p oder n)?

Aufgabe 5: Verlustleistung von CMOS-Schaltungen 9 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 15.000 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 30 mA.

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

Fortsetzung der 5. Aufgabe:

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

d) 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^* = 5 nF modelliert.

e) Wie groß sind die Umladeverluste des Mikrocontrollers?

Aufgabe 6: Baudratengenerierung mit dem 80C51 — 10 Punkte

Unter Verwendung des integrierten Timers 1 bietet der Mikrocontroller 80C51 vielfältige Möglichkeiten der Baudratengenerierung für die serielle Datenübertragung.

Im Folgenden soll die Betriebsart 1 der seriellen Schnittstelle verwendet werden. Diese erlaubt die asynchrone Datenübertragung mit variabler Baudrate (8 bit UART), wobei als Zeitbasis die Überlaufrate von Timer 1 dient. Die Taktfrequenz des Mikrocontrollers betrage 16 MHz.

Zunächst soll eine Baudrate von 4800 bit/s generiert werden. Das Steuerbit SMOD habe den Wert 1. Um die Baudratengenerierung mit möglichst geringer Prozessorbelastung durchzuführen, soll Timer 1 im Autoreload-Modus betrieben werden.

a) Geben Sie die Belegung der Spezialfunktionsregister SCON und TMOD in Tabelle 6.1 an und kennzeichnen Sie irrelevante Bits durch 'x'!

	Bit 7							Bit 0
SCON								
TMOD								

Tabelle 6.1: Belegung der Spezialfunktionsregister

b) Berechnen Sie den Autoreload-Wert für Timer 1!

c) Wie groß ist der relative Fehler der erzeugten Baudrate, wenn der Prozessor mit 16 MHz getaktet wird?

Fortsetzung der 6. Aufgabe:

d) Geben Sie für SMOD = 1 die maximale Baudrate bei Betrieb des Timers im Autoreload-Modus an, wenn der Prozessor nun mit 12 MHz getaktet wird!

Aufgabe 7: Addierer und Multiplizierer — 10 Punkte

a) Vervollständigen Sie Abbildung 7.1 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!

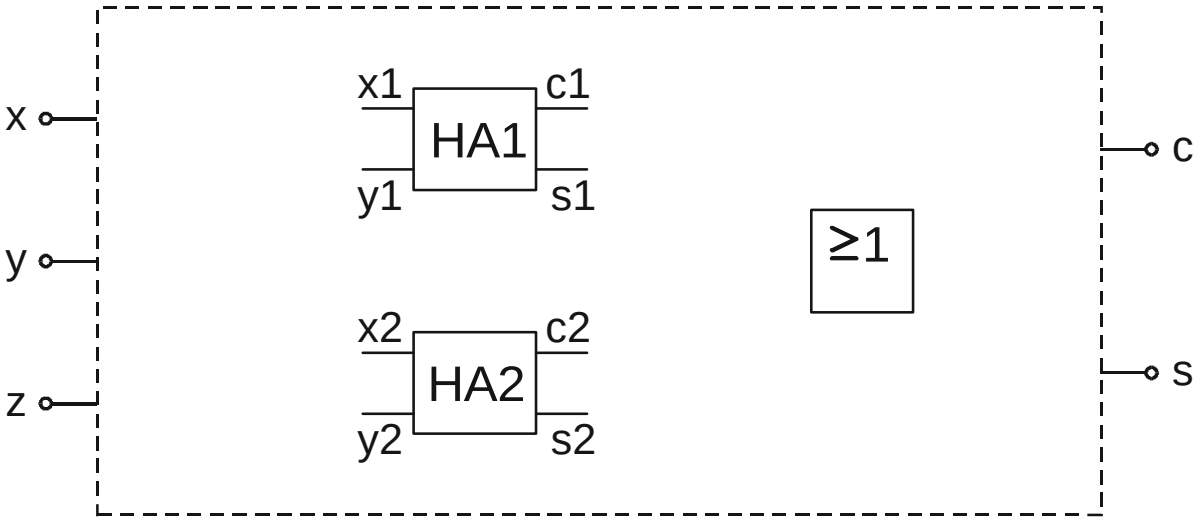

Abbildung 7.1: Aufbau eines Volladdierers

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.

b) Ergänzen Sie Abbildung 7.2 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.

Fortsetzung der 7. Aufgabe:

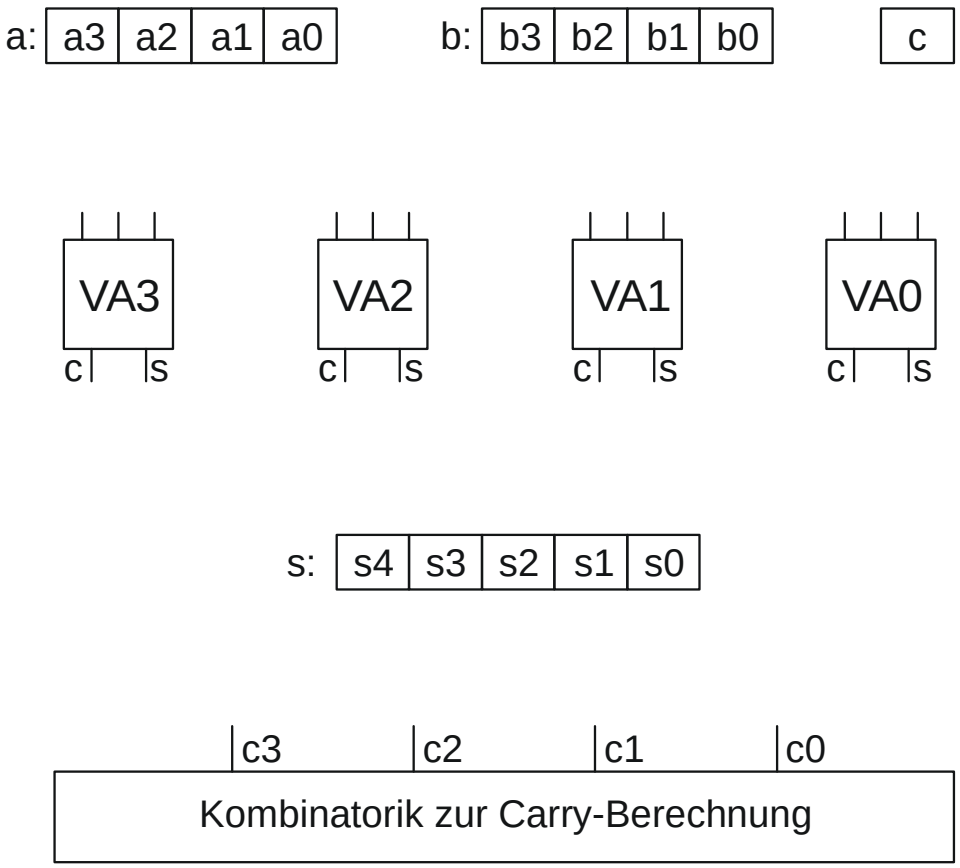

Abbildung 7.2: Carry-Look-Ahead-Addierer zur Addition zweier 4 bit-Zahlen

Im Weiteren wird nun die Arbeitsgeschwindigkeit eines Parallelmultiplizierers für zwei 8-bit-Zweierkomplementzahlen *A* und *B* betrachtet, der gemäß Gleichung 7.1 ein 16 bit langes Produkt *P* als Ergebnis liefert:

$$P = A \cdot B = -2^{15} + 2^8 + a_7 \cdot b_7 \cdot 2^{14} + \sum_{j=0}^{6} \sum_{i=0}^{6} a_i \cdot b_j \cdot 2^{i+j} + \sum_{i=0}^{6} \overline{a_7 b_i} \cdot 2^{i+7} + \sum_{i=0}^{6} \overline{b_7 a_i} \cdot 2^{i+7} \qquad (7.1)$$

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 Abbildung 7.3, unter Einsatz von Halb- und Volladdierern, sowie eines schnellen Carry-Look-Ahead-Addierers für die Abschlussaddition.

Fortsetzung der 7. Aufgabe:

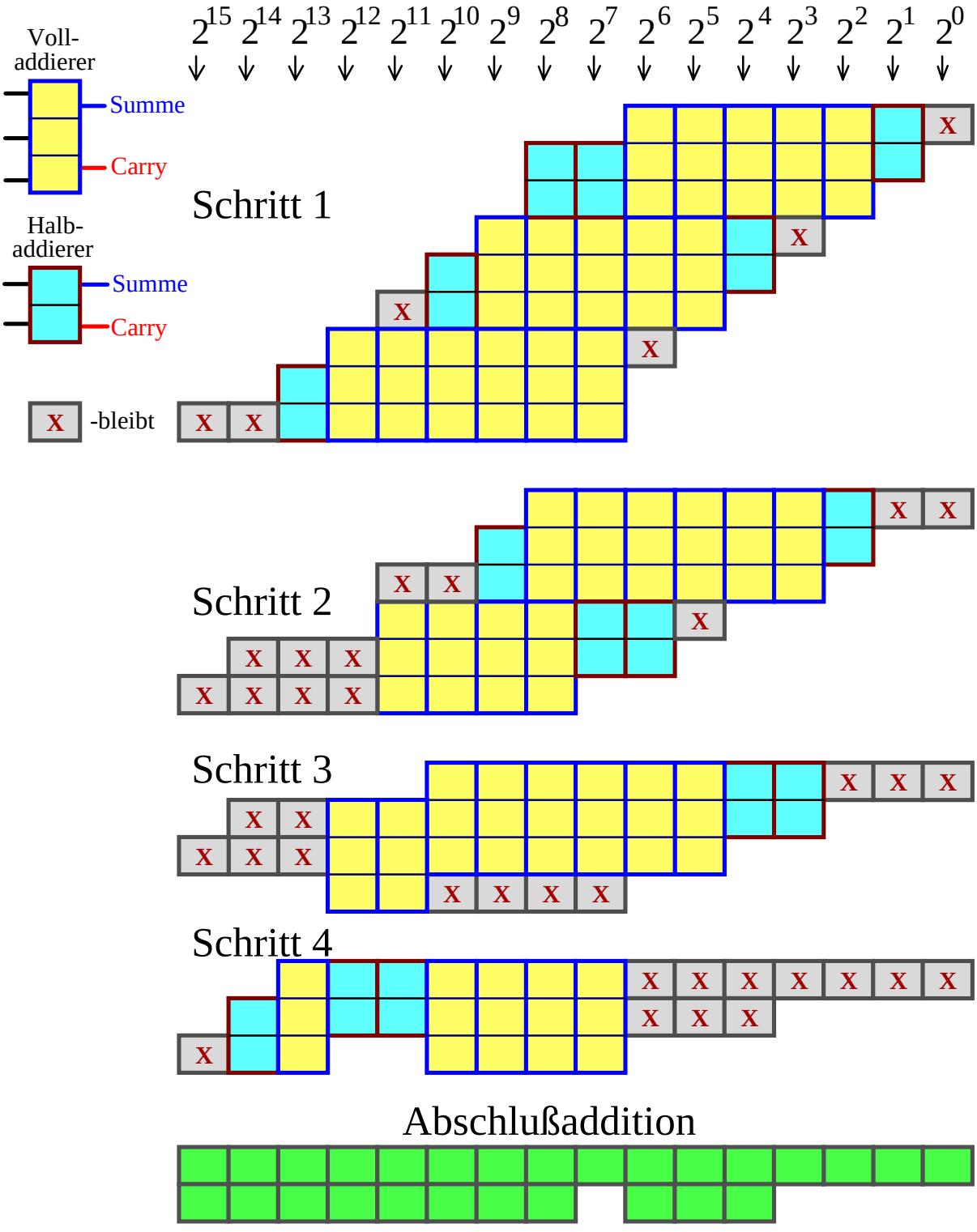

Abbildung 7.3: Schematische Darstellung der Abarbeitung der Multiplikation nach Gleichung 7.1 bis zur Abschlussaddition mit Hilfe von 15 Halb- und 39 Volladdierern in 4 Schritten

Fortsetzung der 7. Aufgabe:

c) Wie lange dauert die Ausführung einer Multiplikation, wenn ein Volladdierer und der Carry-Look-Ahead-Addierer für die Abschlussaddition jeweils 1,5 ns Durchlaufzeit haben?

Jetzt wird nach jedem der obigen Schritte 1...4 jeweils ein Pipelining-Register eingebaut. Jedes dieser Register weist eine Durchlaufzeit von 0,5 ns 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 2 einzufügen ist, speichern?

Aufgabe 8: CMOS-Transfergates 10 Punkte

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

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

Mit Hilfe von Transfergates soll nun möglichst einfach der auf der folgenden Seite in Abbildung 8.1 dargestellte 4:1-Multiplexer realisiert werden, der mittels seiner beiden Auswahlsignale *S0* und *S1* jeweils einen der Eingänge *X0*...*X3* auf den Ausgang *Y* legt.

b) Vervollständigen Sie die dargestellte Multiplexerschaltung, so dass die Funktion aus Tabelle 8.1 erfüllt ist!

S0	S1	Y
0	0	X0
1	0	X1
0	1	X2
1	1	X3

Tabelle 8.1: Zustandsbelegung

Ergänzen Sie dazu die fehlenden Verbindungen im folgenden Schaltbild und kennzeichnen Sie an den Transfergates den negierten Steuereingang jeweils mit einem 'o'!

Fortsetzung der 8. Aufgabe:

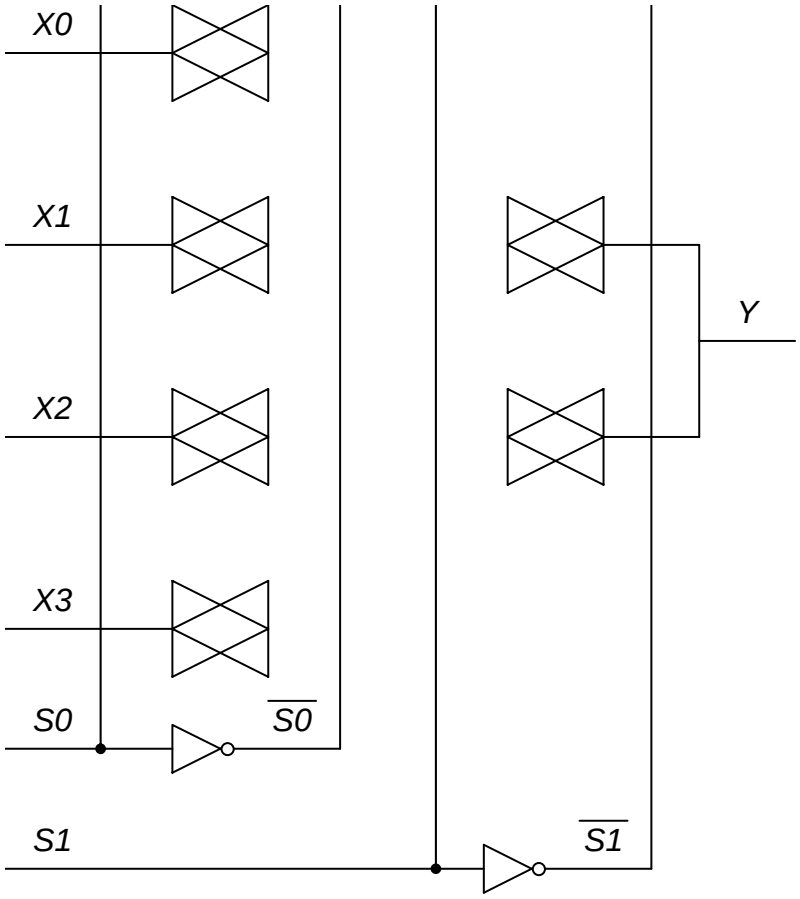

Abbildung 8.1: 4:1-Multiplexer mit Transfergates

Aufgabe 9: Matrizenoperation mit dem DSP 56002 — 10 Punkte

Algorithmen der digitalen Signalverarbeitung können oft sehr kompakt in Matrizenschreibweise dargestellt werden. Die Hardwarearchitektur von digitalen Signalprozessoren ist besonders für die Ausführung von Algorithmen in Matrixdarstellung geeignet.

Gegeben seien die Vektoren

$a^T = [a_1 a_2 ... a_N]$ und $b^T = [b_1 b_2 ... b_N]$

mit jeweils der Länge N, wobei N eine natürliche Zahl ist, sowie die Matrix

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & ... & m_{1,N} \\ m_{2,1} & m_{2,2} & ... & m_{2,N} \\ ... & ... & ... & ... \\ m_{N,1} & m_{N,2} & ... & m_{N,N} \end{bmatrix}.$$

Es soll die Operation $b = M \cdot a$ ausgeführt werden.

a) Geben Sie eine Summenformel zur Berechnung der Elemente b_i des Vektors b für $i = 1,2,\ldots,N$ an!

b) Welche beiden Befehle des DSP 56002 ermöglichen eine effiziente Berechnung der Matrizenoperationen?

Fortsetzung der 9. Aufgabe:

c) Die Vektorlänge N sei 16. Geben Sie ein möglichst kompaktes Assemblerprogramm an, welches das Element b_3 berechnet! Der Vektor a ist im X-Speicher ab der Adresse $100 abgelegt und soll mit dem Register R0 adressiert werden. Die Matrix M ist zeilenweise im Y-Speicher ab Adresse $100 abgelegt und soll mit Register R4 adressiert werden. Das Ergebnis soll nach der Berechnung im Akkumulator A stehen.

Aufgabe 10: Beschreibung und Analyse von Schaltungen mit VHDL 11 Punkte

Zunächst soll ein Multiplizierwerk entwickelt werden. Die beiden zu multiplizierenden Faktoren seien vom Typ *integer* und können Werte im Bereich von 0 bis 127 annehmen.

a) Vervollständigen Sie die nachfolgende ENTITY und die zugehörige ARCHITECTURE, so dass ein entsprechendes Multiplizierwerk entsteht! (Hinweis: Die Entity soll neben den beiden Eingängen einen entsprechend zu dimensionierenden Ausgang, einen Takt-, sowie einen Reset-Eingang enthalten. Falls das Reset-Signal den Wert 0 annimmt, soll der Ausgang des Multiplizierers ebenfalls auf 0 gesetzt werden. Im anderen Fall (Reset = 1) soll jeweils mit der fallenden Taktflanke des Taktes die Multiplikation der beiden Faktoren durchgeführt werden).

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

entity mult is
port(




                                                    );
end mult;

architecture mult_arch of mult is



begin
    process(                              )
    begin









    end process;
end mult_arch;
```

Fortsetzung der 10. Aufgabe:

Nun wird der nachfolgende VHDL-Code des FSM-Modells eines Moore-Automaten betrachtet.

```
entity moore is
port(clk, res: in  bit;
     ein     : in  bit_vector(1 downto 0);
     aus     : out bit_vector(3 downto 0));
end moore;

architecture moore_arch of moore is
  type   zustand is (st0, st1);
  signal zust_akt, zust_folg: zustand;
begin

-- Definition der FSM-Register
process (clk, res)
begin
    if res = '1' then
      zust_akt <= st0;
    elsif (clk'event and clk = '0') then
      zust_akt <= zust_folg;
    end if;
end process;

-- Kombinatorik der FSM
process (zust_akt, ein)
begin
  case zust_akt is
    when st0 =>
      case ein is
        when "00" => zust_folg <= st1;
        when "01" => zust_folg <= st0;
        when "10" => zust_folg <= st0;
        when "11" => zust_folg <= st1;
      end case;
    when st1 =>
      case ein is
        when "00"   => zust_folg <= st0;
        when "01"   => zust_folg <= st0;
        when others => zust_folg <= st1;
      end case;
  end case;
end process;

-- Ausgabewerte
process (zust_akt)
begin
  case zust_akt is
    when st0    => aus <= "1011";
    when st1    => aus <= "0111";
  end case;
end process;
end moore_arch;
```

Fortsetzung der 10. Aufgabe:

b) Vervollständigen Sie das Blockschaltbild des Moore-Automaten in Abbildung 10.1 mit Ein- und Ausgängen und beschriften Sie diese!

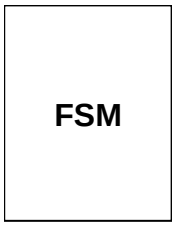

Abbildung 10.1: Blockschaltbild des Moore-Automaten

c) Skizzieren Sie den Zustandsgraphen der FSM! Bezeichnen Sie die Knoten des Graphen entsprechend den Zuständen *st0* und *st1*! Falls irrelevante Eingangswerte auftreten, können Sie die Übergangsbedingung an der entsprechenden Stelle mit *'x'* bezeichnen.