LÖSUNGSVORSCHLAG

20.09.2004

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

Vordiplomprüfung im Fach

Mikrorechnertechnik

H04

Aufgabe 1: **A/D- und D/A-Wandlung**

a) (2)

Für die Adresse werden 2x3= 6 Bits benötigt.
Für den Datenausgang werden ebenfalls 6 Bits benötigt, um alle 2^6 Speicherstellen auslesen zu können.

b) (2)

Adresse	Inhalt
0000	0000
0001	0000
0010	0000
0100	0000
0101	0001
0110	0010
0111	0011
1000	0000
1001	0010
1010	0100
1011	0110
1100	0000
1101	0011
1110	0110
1111	1001

Tabelle 1.1: Adresse und Speicherinhalt

c) (2)

Jeder der Schritte 1-4 benötigt genau eine Volladdiererlaufzeit; hinzu noch die Zeit für die Gatter-Multiplikationen sowie die Abschlussaddition

=> Gesamtlaufzeit= $4*\tau_{VA}$ + 1 ns + 2 ns= 4 * 2ns + 1 ns +2ns = 11ns.

d) (2)

f_{max} = 1/(2ns + 1 ns + 0,5ns) = 285,714 MHz

e) (2)

Der Register muss alle Ergebnisbits vom Schritt 3 Speichern, d.h. pro HA und VA sind jeweils 2 Bits und für mit X gekennzeichneten nicht veränderten Stellen jeweils 1 Bit

=> Register nach Schritt 3 muss 32 Bit lang sein.

Aufgabe 2: **Zahlendarstellung in Mikrorechnerprogrammen**

a), b) (2), (2)

Register	**Inhalt (dezimal)**	**Inhalt (binär)** MSB LSB
R0	–6	**1 1 1 1 1 0 1 0**
R1	18	**0 0 0 1 0 0 1 0**
A	**–108**	**1 0 0 1 0 1 0 0**

c) (1)

$1111\ 1010_b$=>Fraktal: $-1+(2^6+2^5+2^4+2^3+2^1)/2^7 = -1+0{,}953125 = -0{,}046875$
oder schneller: -6 / 128 = -0,046875

d) (1)

$1\ 111{,}1010_b$= > Vorzeichen negativ
Wert der ganzen Zahl= 111_b =7
Nachkommawert= $2^{-1}+2^{-3}=0{,}625$
Dezimalwert= -7,625
oder schneller: -122 / 2^4 = -7,625

e) (4)

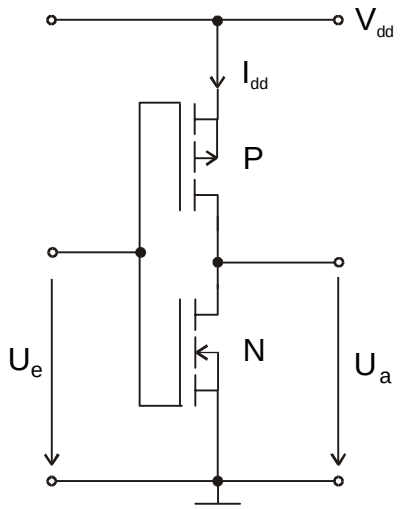
$-21 = -10101_b = -1{,}0101_b \cdot 2^4 = -1{,}0101_b \cdot 2^{131-127}$

- ⇨ Vorzeichenbit = 1
- ⇨ Exponent = 131 = $1000\ 0011_b$
- ⇨ Mantisse = 0101…

Inhalt (dezimal)	MSB **Inhalt (binär)** LSB																															
	31							24	23							16	15							8	7							0
–21	1	1	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
99	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	0	0

Aufgabe 3: **Verlustleistung von CMOS-Schaltungen**

a) (2 Punkte)



U_e = H: (1 Punkt)

Der obere Transistor (p-Kanal) sperrt, der untere Transistor (n-Kanal) leitet; damit liegt der Ausgang an Masse, also auf logisch L.

U_e = L: (1 Punkt)

Der obere Transistor (p-Kanal) leitet, der untere (n-Kanal) sperrt; damit liegt der Ausgang an V_{dd}, also auf logisch H.

b) (2 Punkte)

Umschaltverluste: Verluste durch Umschaltströme, die während jeder Taktflanke beim Umschalten von Invertern und Gattern entstehen

Umladeverluste: Verluste beim Umladen von Kapazitäten (z. B. Busleitungen)

c) (3 Punkte)

$$\bar{I} = N_{Inv} \cdot \frac{1}{T_{Takt}} \int_o^{T_{Takt}} i_d(t)\,dt = N_{Inv} \cdot f_{Takt} \cdot \left[\int_o^{t_r} i_d(t)\,dt + \int_o^{t_f} i_d(t)\,dt \right] = N_{Inv} \cdot f_{Takt} \cdot \left[\frac{t_r}{2} + \frac{t_f}{2} \right] \cdot I_{DP}$$

$$\Rightarrow I_{DP} = \frac{\bar{I}}{N_{Inv} \cdot f_{Takt} \cdot \left[\frac{t_r}{2} + \frac{t_f}{2} \right]} = \frac{48mA}{32.000 \cdot 12{,}5MHz \cdot [0{,}75ns + 0{,}5ns]} = 96\mu A$$

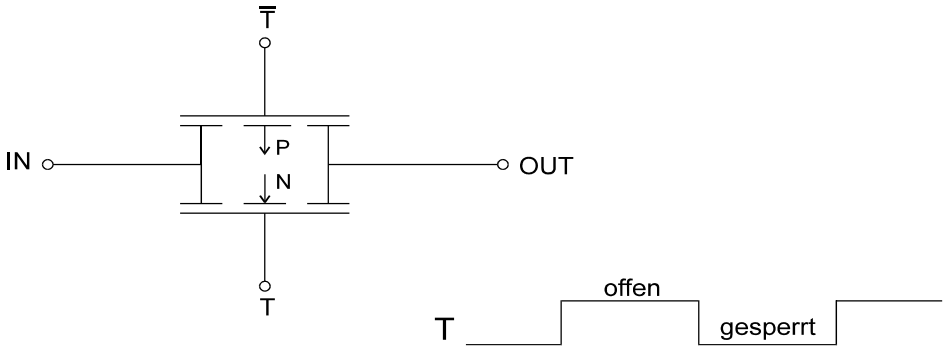
d) (2 Punkte)

$$I_{\max} = i_{ges}(t_p) = N_{Inv} \cdot I_{DP} = 32.000 \cdot 100\mu A = 3{,}2A$$

$$P_{\max} = p_s(t_p) = U \cdot I_{\max} = U \cdot N_{Inv} \cdot I_{DP} = 2{,}5V \cdot 32.000 \cdot 100\mu A = 8W$$

Aufgabe 4: **CMOS-Transfergates**

a) (4 Punkte: 2 Pkt. für die zeichnung, 1 Pkt. für die Schaltungsbeschreibung und 1 Pkt. für die Beschreibung)



Transfergate ist "offen" bei T=1, /T=0
Transfergate ist "gesperrt" bei T=0, /T=1
Mit Transfergates können komplexe Gatter mit weniger Transistoren aufgebaut werden

b) (6 Punkte, jeweils 2 für die richtige Ausgangsbelegung c, d und 2 für die Funktion)

a	b	c	d
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Die Funktion entspricht einem Halbaddierer, wobei der Ausgang c die Summe und der Ausgang d der Übertrag ist.

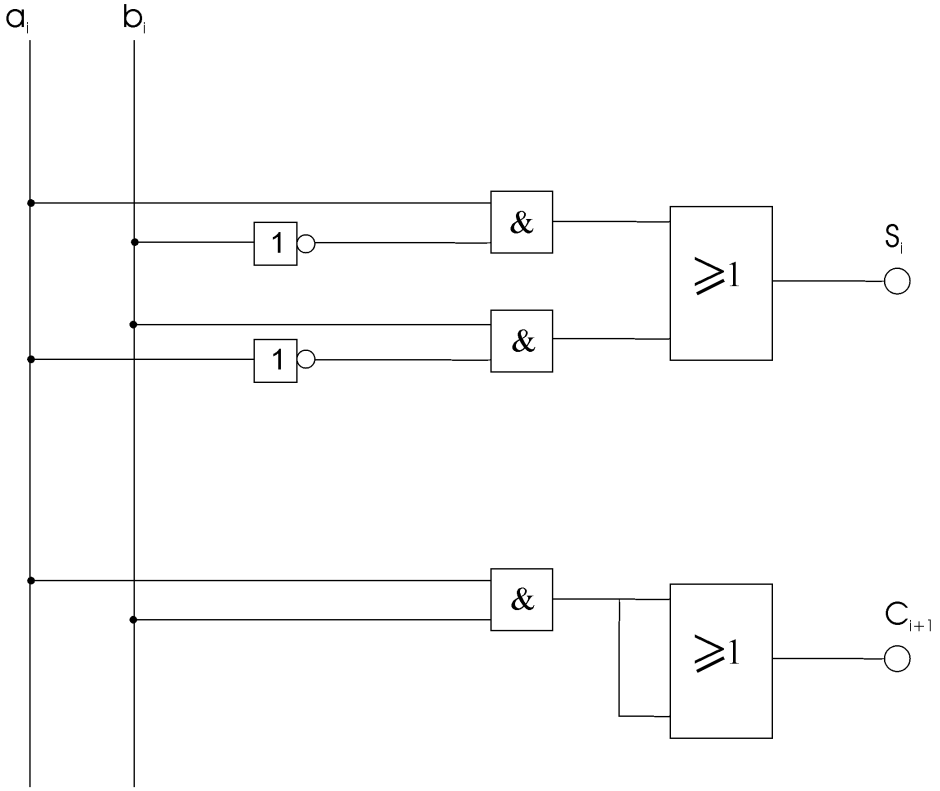
Aufgabe 5: **Addierer**

a) (2 Punkte, jeweils ein Punkt pro Gleichung)

boolesche Gleichungen: Summe: $s_i = a_i \oplus b_i = a_i \cdot \overline{b_i} + \overline{a_i} \cdot b_i$

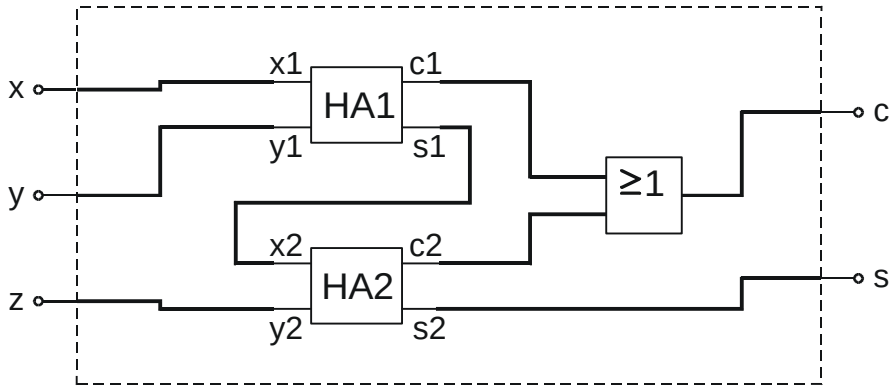
Übertrag: $c_{i+1} = a_i \cdot b_i$

b) (2 Punkte)

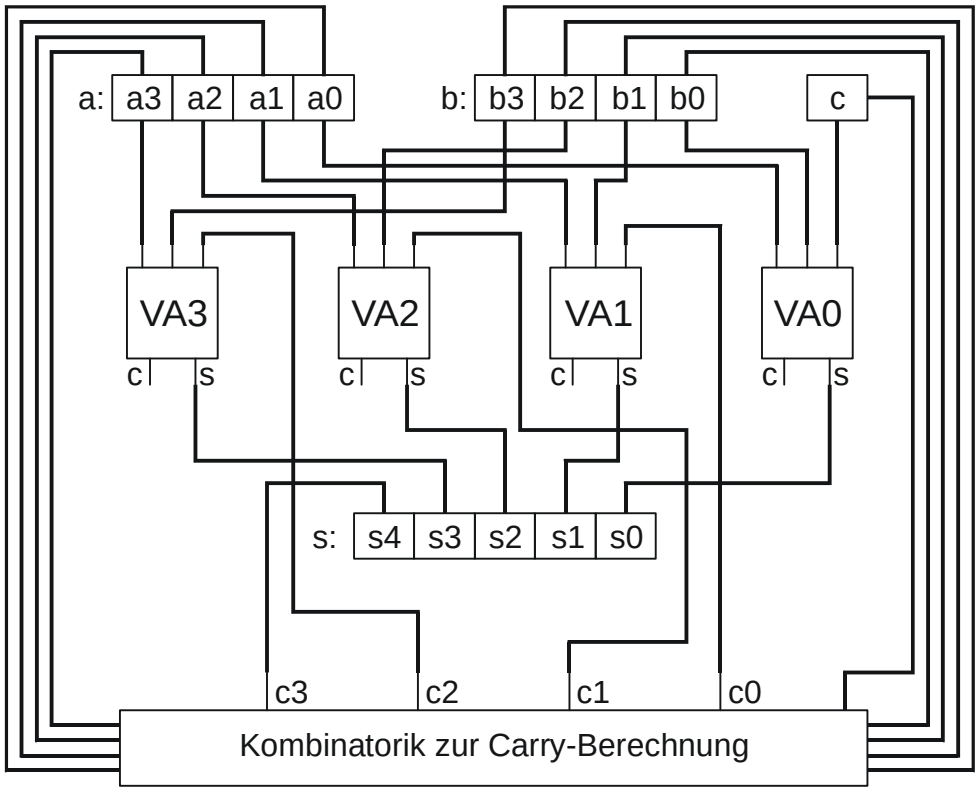


c) (2 Punkte)

Eine Möglichkeit:



d) (4 Punkte)


a:
a3
a2
a1
a0
b:
b3
b2
b1
b0
c
VA3
VA2
VA1
VA0
c
s
c
s
c
s
c
s
s:
s4
s3
s2
s1
s0
c3
c2
c1
c0
Kombinatorik zur Carry-Berechnung

Aufgabe 6: **Entwurf eines Steuerwerks**

a) (4 Punkte)

Befehl	OP4	OP3	OP2	OP1	Beschreibung
INC A	0	0	1	0	Incrementiere A
ANDL A, INR	0	1	0	0	A= (INR) AND (A)
ORL A, INR	0	1	0	1	A= (INR) OR (A)
NOP	1	0	0	1	No Operation

Tabelle 6.1: Realisierte Befehle

b) (1 Punkte)

S_0

c) (2 Punkte)

$(S_0) - S_1 - S_4 - S_0$

d) (3 Punkte)

	IN_LOAD	**ADR_ IN**	**ADR_ LOAD**	**ALE**	**READ**	**WRITE**	**TR_IN**	**X_LOAD**	**ADD**	**OUT_ LOAD**
Schritt 1	0	0	0	0	0	0	1	0	0	1
Schritt 2	1	0	0	1	1	0	0	0	1	1

Für Schritt 1 einen Pkt.
Für Schritt 2 zwei Pkte.

Aufgabe 7: **A/D-Wandlung mit dem Mikrocontroller ADuC832**

a) (4 Punkte)

=> d.h. pro richtige Zeile je einen Pkt.

Tabelle 1.2: Entwicklung der SAR-Inhalte für 3 verschiedene Eingangsspannungen

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

b) (1 Punkt)

Weil der Sättigungswert des A/D-Wandlers bei 7,5 V liegt.

c) (2 Punkte)

Auflösung in Bit: N= ld($f_c \cdot T$)

d) (1 Punkt)

Wenn bei einem Überlauf des Binärzählers der Ausgangsportpin auf eine logische „1“ gesetzt wird, dann muss dieser bei einer Übereinstimmung von Zähler- und Vergleichseinheit auf eine logische „0“ gesetzt werden, und umgekehrt.

e) (2 Punkte)

Registerwert: 3V / 5V *65536= 39321,6 ≈ 39322.

Aufgabe 8: **µC-Programmierung**

a) (7 Punkte)

```
;****************** PROGRAMMAUSSCHNITT ****************************
MOV   R6,ADCDATAL      ;unteres Byte vom ADW
MOV   A,ADCDATAH       ;oberes Byte vom ADW
ANL   A,#07h           ;obere 5 Bits auf Null setzen              ( 1Pkt )
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                                          ( 2Pkte )
CLR      C             ; vorsorglich Carry löschen
MOV      A,R7
RRC      A             ;rechts rotieren,wobei Bit 0 in Carry kommt
MOV      R5,A          ;oberes halbiertes Byte in R5 ablegen
MOV      A,R6
RRC      A             ;halbieren,wobei Carry in Bit 7 und Bit 0 in Carry
MOV      R4,A          ;unteres halbiertes Byte in R4 ablegen

           ;Verdoppeln                                         ( 2Pkte )
CLR      C
MOV      A,R6
RLC      A             ;mal 2 nehmen, wobei Bit 7 in Carry und eine Null
```

```
                        ;in Bit 0 kommt
MOV     R6,A            ;in R6 zurückspeichern (überschreiben)
MOV     A,R7
RLC     A               ;oberes Byte verdoppeln, Carry aus R6 in Bit 0
MOV     R7,A            ;in R7 zurückspeichern (überschreiben)


        ;halben und doppelten Wert addieren              ( 2Pkte )
MOV     A,R4
CLR     C               ;vorsorglich
ADD     A,R6            ;halbes und doppeltes unteres Byte addieren
MOV     R6,A            ;Ergebnis für unteres Byte in R6 ablegen
MOV     A,R5            ;halbiertes oberes Byte holen
ADDC    A,R7            ;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) (3 Punkte)

Berechnung: $(301_h \cdot 2{,}5 / 2) / 2 = 480{,}625 = 01\ E0_h$

- ⇨ Inhalt von R7 = 01
- ⇨ Inhalt von R6 = E0

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

a) Der MAC-Befehl in Verbindung mit Parallel Moves und einer Do-Loop-Schleife (2 Punkte)
b) (6 Punkte)

```
      move      #$1000,R0
      move      #$1000,R4
      clr  A          X:(R0)+,X0     Y:(R4)+,Y0
      do   #16,end
      mac  x0,y0,A    X:(R0)+,X0     Y:(R4)+,Y0
      end
```

c) `move #15, M0` und `move #15, M4` (2 Punkte)

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

a) (3 Punkte, für jeden Fehler einen Punkt abziehen)

b) (5 Punkte, für jeden Fehler einen Punkt abziehen)

```
...
  PORT(clock, clear, reset: IN bit;
       x, y: IN  Integer Range 0 to 63;
       sum: OUT Integer Range 0 to 32767);
...

  SIGNAL zwischen: Integer Range 0 to 4095;

...

  Comp1: mult PORT MAP(clock, reset, x, y, zwischen);

  Comp2: akku PORT MAP(clock, reset, clear, zwischen, sum);

...
```

c) (2 Punkte, für jeden Fehler einen Punkt abziehen)

maximaler Eingang: $2^6 - 1$
maximaler Ausgang: $2^{15} - 1$
mögliche MAC-Operationen: $\frac{2^{15}-1}{\left(2^6-1\right)^2} = 8{,}256$
es sind maximal 8 MAC-Operationen ohne Überlauf möglich.