Vorname: _____ **Name:** _____

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

19.09.2005 - 14^{00} Uhr bis 16^{00} Uhr

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

Diplomprüfung im Fach

"Mikrorechnertechnik"

Musterlösung
H05

Aufgabe:	1	2	3	4	5	6	7	8	9	10	gesamt
Punkte:											
Erreichbare Punktzahl:	9	10	10	11	10	9	10	11	10	10	100

Aufgabe 1: Speicher und speicherbasierte Anwendungen 9 Punkte

a) ROM steht für Read Only Memory, also ein Speicher, aus dem während der Laufzeit der Hardware nur gelesen werden kann. Ein EEPROM ist ein ROM, das elektrisch gelöscht und wieder beschrieben werden kann. EEPROM sind wesentlich langsamer als ein RAM, daher werden sie nicht als RAM verwendet.

b)

Adresse	Inhalt
00000000	00000000
01010010	00001010
00110110	00010010
00111001	00011011
01111000	00111000
11100111	01100010
11001010	01111000
10101101	10000010
10111101	10001111
11101100	10101000
11111111	11100001

c) Für große Wortbreiten würden sehr große Speicher benötigt. Für einen 16x16bit-Multiplizierer wäre ein ROM mit einer Speicherkapazität von 16 GByte nötig.

Aufgabe 2: Zahlendarstellung in Mikrorechnerprogrammen — 10 Punkte

a) $-2^{31} + 2^{26} + 2^{24} + 2^{23} + 2^{21} + 2^{14} + 2^{13} + 2^{12} = -2053083136$

b) keine gültige BCD-Zahl

c) $1000\ 0101\ 1010\ 0000\ 0111\ 0000\ 0000\ 0000_b$

=> Fraktal: $-1 + \frac{2^{26} + 2^{24} + 2^{23} + 2^{21} + 2^{14} + 2^{13} + 2^{12}}{2^{31}} = -0{,}95604133605957$

d) $1000\ 0101\ 1010\ 0000\ 0111\ 0000\ 0000\ 0000_b$

=> VZ= negativ

Exponent: $0000\ 1011_b = 11$ => Exp= $11-127 = -116_d$

Mantisse: m= 010 0000 0111 0000 0000 0000

Zahl= $(-1)^1 \bullet [1 + m] \bullet 2^{Exp}$

$= (-1)^1 \bullet [1 + (2^{-2} + 2^{-9} + 2^{-10} + 2^{-11})] \bullet 2^{-116}$

$= (-1)^1 \bullet [1 + 2^{-11} \bullet (2^9 + 2^2 + 2^1 + 1)] \bullet 2^{-116}$

$= (-1)^1 \bullet 2^{-11} \bullet [2^{11} + 2^9 + 2^2 + 2^1 + 1] \bullet 2^{-116}$

$= -\frac{2567}{2^{127}}$ ($\approx -1.50874699928 * 10^{-35}$)

e) $-8\frac{3}{4} = (-1)^1 \bullet 8\frac{3}{4}$

$= (-1)^1 \bullet [2^3 + \frac{1}{2} + \frac{1}{4}] = (-1)^1 \bullet [2^3 + 2^{-1} + 2^{-2}]$

$= (-1)^1 \bullet 2^3 \bullet [1 + 2^{-4} + 2^{-5}]$

$= (-1)^1 \bullet [1 + 2^{-4} + 2^{-5}] \bullet 2^{130-127}$

=> VZ= negativ

Exponent: $130_d = 1000\ 0010_b$

Mantisse: m= 000 1100 0000 0000 0000 0000

=> Gleitkommazahl= 1100 0001 0000 1100 0000 0000 0000 0000

f)

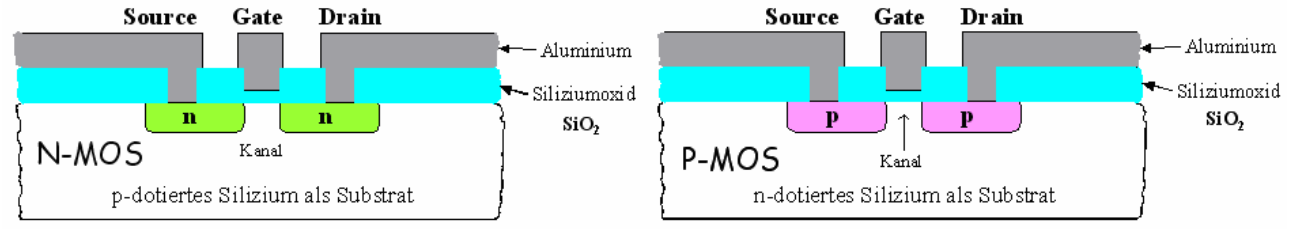
Betrag der Zahl $8\frac{3}{4}$ = 0000 0000 0000 1000 . 1100 0000 0000 0000

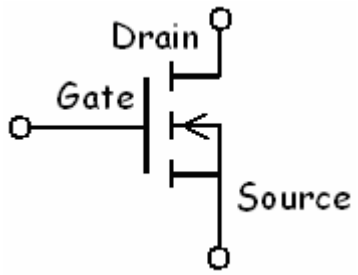
Davon das 2er-Komplement durch Negation und Addition von 1 an der niederwertigsten Stelle, also nicht an der ersten Vorkommastelle:

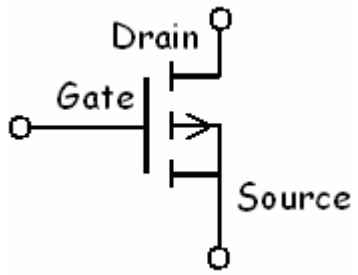
=> $-8\frac{3}{4}$ = 1111 1111 1111 0111 . 0100 0000 0000 0000

Aufgabe 3: CMOS-Technologie — 10 Punkte

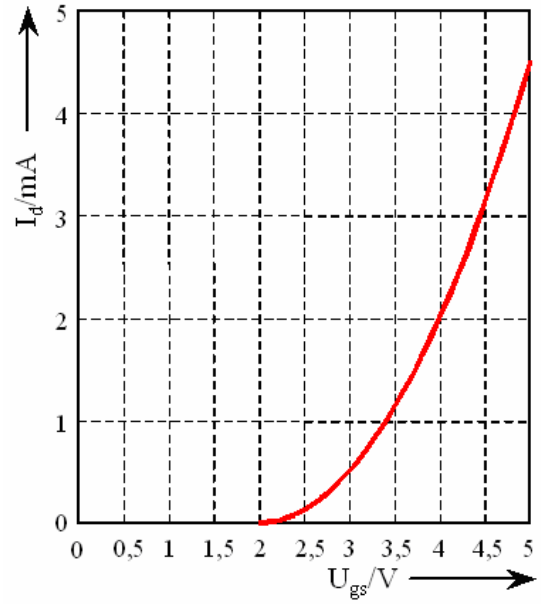
a)

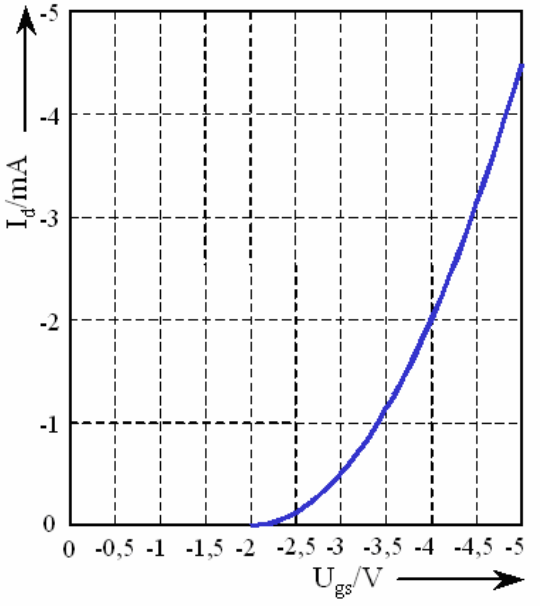






b)





n-Kanal — p-Kanal

Schwellspannung U_{th} bei 2V bzw. -2V.

c) n-Kanal => p-Dotierung. Durch die p-Dotierung ergibt sich ein „Mangel“ an negativen Ladungen im Substrat. Legt man eine positive Spannung zwischen Gate und Source werden negative Ladungen zum Gate gezogen, Bildung eines Überschusses an negativen Ladungen in diesem Bereich => man spricht von einem Kanal. Je höher die Spannung wird, desto breiter ist der Kanal. Bei entsprechender Spannung fließt dann ein Drainstrom (Drain -> Source).

Aufgabe 4: CMOS-Transfergates 11 Punkte

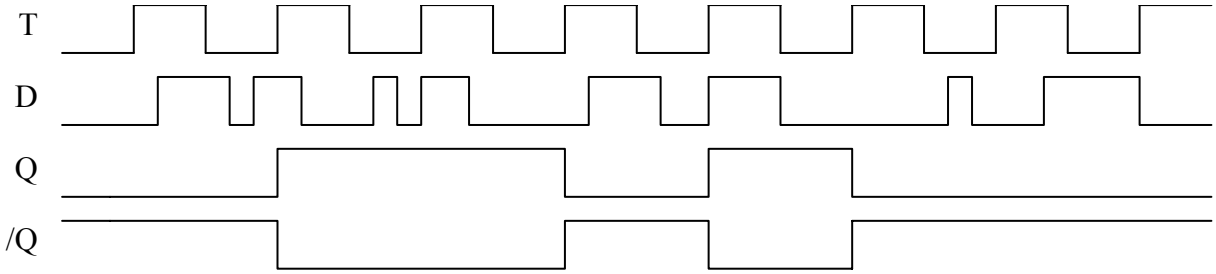
a)

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

Es wird eine XNOR-Verknüpfung (Äuqivalenz) realisiert.

b) TG1: offen, TG2: gesperrt, TG3: gesperrt, TG4: offen

c)



d) Es handelt sich um ein vorderflankengetriggertes D-Flip-Flop.

e) Das Bit wird in Schleife S2 gespeichert. Transfergate 4 ist für den Fall T=0 geöffnet, damit wird der Wert des Ausgangs Q durch die Schleife zurückgegeben und dort gespeichert.

Aufgabe 5: Arithmetik-Schaltungen — 10 Punkte

a) Zustandstabelle eines Volladdierers:

a_i	b_i	c_i	d_i	e_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

b)

„d“ stellt das Summenbit dar
„e“ stellt das Übertragsbit dar
insgesamt wird die Funktion eines Volladdierers realisiert

c) Zustandstabelle eines Halbsubtrahierers:

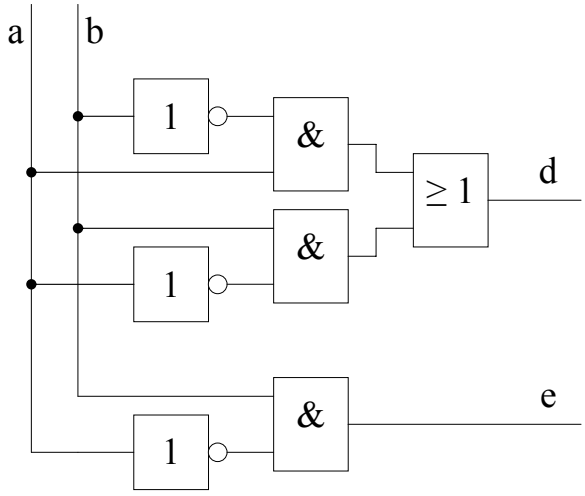
a_i	b_i	d_i	e_i
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

d)

boolesche Gleichungen: Differenz: $d_i = a_i \oplus b_i = a_i \cdot \overline{b_i} + \overline{a_i} \cdot b_i$

Entleihung: $e_i = \overline{a_i} \cdot b_i$

d) Schaltnetz eines Halbsubtrahierers



Aufgabe 6: Automaten, FSM — 9 Punkte

a)

Eingabe	*Zustand*	*Ausgabe*
-	A	-
00	B	0
01	C	1
10	C	1
11	D	0
11	E	1
00	C	1
00	B	0
11	D	0
10	D	0
11	E	1
11	E	1

b)
Moore-Automat, da die Ausgabe im Zustand erfolgt.

c)
Ausgabebit entspricht der Summe der Eingabebits.

Zustandsbit entspricht dem Übertrag aus der Summe der Eingabebits.

d)

***flow-table**

relevant = OPC3, OPC2, OPC1, COND, INIT;

s7	, x	**0**	**0**	**0**	–	–	, **f1**;	von STOP zu Reset bei RES
s7	, x	–	–	–	–	**1**	, **f1**;	von STOP zu Reset bei START
s1	, x	**–**	**–**	**–**	**–**	0	, **f7**;	nach Reset-Ausführung ⇒ STOP
s7	, x	**1**	**0**	**0**	**–**	**0**	, **f2**;	JMP
s7	, x	**1**	**0**	**1**	**1**	**0**	, **f2**;	JMC mit COND = 1
s2	, x	**–**	**–**	**–**	**–**	**0**	, **f7**;	
s7	, x	**1**	**1**	**0**	**–**	**0**	, **f3**;	JMS
s3	, x	**–**	**–**	**–**	**–**	**0**	, **f4**;	
s4	, x	**–**	**–**	**–**	**–**	**0**	, **f7**;	

s7	, x	**1**	**1**	**1**	**–**	**0**	, **f6**;	RET
s6	, x	**–**	**–**	**–**	**–**	**0**	, **f5**;	
s5	, x	**–**	**–**	**–**	**–**	**0**	, **f7**;	

s[1..s7]	, xrest	, **f7**;	alle übrigen Eingaben bewirken STOP

Aufgabe 7: Mikrocontrollerprogrammierung — 10 Punkte

a) Assembler-Programm:

```
------------------------------------------------
BIN_BCD:
MOV   R5,#00h     ;Zwischenregister leeren
MOV   R4,#00h

MOV   A,ADL
MOV   B,#100      ;Inhalt von ADL durch 100 teilen
DIV   AB          ;wobei das ganzzahlige Ergebnis (Hunderter) in A
                  ;und der Rest in B steht
MOV   R5,A        ;Hunderter abspeichern
MOV   A,B         ;Rest (Zehner und Einer) in Akku holen
MOV   B,#10
DIV   AB          ;Rest durch 10 teilen (Zehner in A, Einer in B)
SWAP  A           ;Zehner in oberes Halbbyte
ADD   A,B         ;Einer dazuaddieren
MOV   R4,A        ;Zehner und Einer abspeichern

;jetzt das höchstwertige Bit in ADH prüfen:
;-----------------------------------------
MOV   A,ADH       ;höherwertiges Byte holen

; - falls Bit 0 gesetzt, 256 zu R5, R4 addieren mit Dezimalkorrektur

JNB   ACC.0,FERTIG     ;falls "0", fertig
;256 zu R5,R4 addieren
;----------------------
MOV   A,R4        ;Zehner und Einer holen
ADD   A,#56h      ;Achtung: Dezimalwert als HEX-Zahl
DA    A           ;Dezimalkorrektur
MOV   R4,A        ;korrigierten Wert abspeichern
MOV   A,R5        ;Hunderter holen
ADDC  A,#02h      ;mit Carry aus Zehnern addieren
DA    A           ;Dezimalkorrektur
MOV   R5,A        ;Tausender - Hunderter abspeichern
FERTIG:
;ASCII-Codes erzeugen und wie folgt ablegen:
;     - Hunderter in AD_H
;     - Zehner   in AD_Z
;     - Einer in AD_E

MOV   A,R5        ;Hunderter holen
ADD   A,#30h      ;eine "3" voranstellen
MOV   AD_H,A      ;ASCII-Hunderter fertig in AD_H

MOV   A,R4        ;Zehner und Einer holen
SWAP  A           ;Zehner in unteres Halbbyte
ANL   A,#0Fh      ;oberes Halbbyte (Einer) löschen
ADD   A,#30h      ;eine "3" voranstellen
MOV   AD_Z,A      ;ASCII-Zehner fertig in AD_Z

MOV   A,R4        ;nochmal Zehner und Einer holen
ANL   A,#0Fh      ;oberes Halbbyte (Zehner) löschen
ADD   A,#30h      ;eine "3" voranstellen
MOV   AD_E,A      ;ASCII-Einer fertig in AD_E

RET
```

b)

A = 01h
A = 31h
AD_H = 31h
A = 23h
A = 32h
A = 02h
A = 32h
AD_Z = 32h
A = 23h
A = 03h
A = 33h
AD_E = 33h

Aufgabe 8: Langzeitintervall-Timer eines ADuC842 — 11 Punkte

a)

	Bit 7							Bit 0
PLLCON	0	0	0	0	0	1	1	1
TIMECON	0	0	1	1	0	0	0	0
INTVAL	0	1	1	1	1	0	0	0

b)

```
;Intervall-Timer vorbereiten, damit er 5 Tage Pause erzeugt
MOV    TIMECON,#00110000b          ;Intervallzähler nicht freigeben
                                 ;noch keinen Timer-Takt setzen
                                 ;Zeitbasis für 5 Tage setzen

MOV    INTVAL,#120               ;5 Tage Pause vorgeben


CALL   PAUSE                     ; “CALL“ bei ADuC832=“ACALL“ bei 80C51

;--------- Unterprogramm für 5 Tage PAUSE -----------------------
PAUSE:
MOV    PLLCON,#07h               ;energiesparsamsten Modus EIN

ORL    TIMECON,#00000011b          ;Bit 0 und Bit 1 setzen=>
                                 ;Bit 0=TCEN =Intervalltimer starten
                                 ;Bit 1=TIEN =Freigabe des Intvallzählers
WART:
MOV    A,TIMECON                 ;Bit 2 = TII in TIMECON prüfen und
JNB    ACC.2,WART                ;Sprung zu WART

MOV    TIMECON,#00110000b         ;Stopp und Reload Langzeittimer
MOV    PLLCON,#00h                ;Taktfrequenz wieder schneller
RET

;--------- END OF 5 Tage PAUSE ----------------------
```

c) Abfrage TIMECON.2 geht nicht, da TIMECON nicht bitaddressierbar ist.

d) Warten bis Timer-Interrupt-Bit automatisch beim Overflow gesetzt wird.

Aufgabe 9: Analyse von Assembler-Code für den DSP56001 — 10 Punkte

a)
Der Code ist fehlerhaft und muss wie folgt abgeändert werden:

Zeile 2: MOVE #$FF, M0
Zeile 3: MOVE #$0, R4
Zeile 4: MOVE #$FF, M4
Zeile 6: REP #254
Zeile 8: MAC X0,Y0,A X:(R0),X0 Y :(R4)+,Y0
Zeile 9: MAC X0,Y0,A X:X1,(R0)

b)

	R0 (S)	N0	R0 (E)	A	Bezeichnung
MOVE X:-(R0),A	**$153**	**beliebig**	**$152**	$222	**indirect with predecrement**
MOVE X: (R0)-N0, A	**$153**	$2	$151	$111	indirect with postdecrement by offset
MOVE X:(R0+N0),A	**$14F**	**$4**	$14F	$111	indirect by offset

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

a)

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

entity mac is
port (clk   : in  std_logic;
      reset : in  std_logic;
      a, b  : in  integer range 0 to 127;
      result: out integer range 0 to 145160);
end mac;

architecture mac_arch of mac is

signal mac: integer range 0 to 145160;

begin
    process(clk, reset)
    begin

    if reset = '0' then
        mac    <= 0;
        result <= 0;
    else
        if clk'event and clk = '0' then
            mac <= a*b+mac;
        end if;
    end if;
    end process;

result <= mac;

end mac_arch;
```

b) Der Überlauf tritt nach 9 Schritten auf.

c) Bei Signalen werden die zugewiesenen Werte um einen Delta-Delay verzögert übernommen, bei Variablen werden die Werte direkt übernommen. Signale werden benötigt um Flip-Flops zu modellieren.

d) Mit VHDL wird eine Hardware beschrieben. Im Gegensatz zu herkömmlichen Sprachen wie C, mit denen eine Software beschrieben wird, können mit VHDL parallele und sequentielle Vorgänge beschrieben werden. Mit herkömmlichen Programmiersprachen zur Entwicklung von Software können nur sequentielle Abläufe entwickelt werden.