**Hochschule Bremerhaven**

INSTITUT FÜR AUTOMATISIERUNGS- UND ELEKTROTECHNIK

Prof. Dr.-Ing. Kai Müller

# Übungsklausur DMP / PT SS03

Bearbeitungszeit 90 Minuten
– Unterlagen gestattet –



**Name:** _____

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

| 1 | 2 | 3 | 4 | 5 | 6 | Σ |
|---|---|---|---|---|---|---|
| | | | | | | |

## (1) Konversion zwischen Zahlensystemen und Rechnen mit Dualzahlen

Gegeben seien die Zahlen $x = 148_{10}$ und $y = 19_{10}$ im Dezimalsystem.

(1.1) Konvertieren beide Zahlen in das Dualzahlensystem. Die Berechnung muss nachvollziehbar durch Bestimmung der Reste erfolgen.

(1.2) Multiplizieren Sie beide Zahlen im Dualzahlensystem (Rechnung ausführlich durchführen) $z = x\,y$.

(1.3) Stellen Sie das Ergebnis $z$ als Hexadezimalzahl dar. Erläutern Sie Ihre Vorgehensweise.

(1.4) Stellen Sie $-z$ als 16-Bit Zahl im Zweierkomplement dar.

## (2) Minimierung logischer Funktionen mit KV-Diagramm

Für die folgende Wahrheitstabelle soll eine digitale Schaltung entworfen werden.

| x2 | x1 | x0 | y1 | y0 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |

Aufgrund der unterschiedlichen Anzahl der Nullen und Einsen in $y_1$ und $y_2$ bieten sich konjunktive und disjunktive Normalformen zur Verwirklichung der Funktionen an.

| (1) = 5 | (2) = 5 | (3) = 5 | (4) = 5 | (5) = 10 | Σ = 30 |
|---|---|---|---|---|---|
| 1.0 ≥ 27 | 1.3 ≥ 25 | 1.7 ≥ 23 | 2.0 ≥ 21 | 2.3 ≥ 19 | 2.7 ≥ 18 | 3.0 ≥ 17 | 3.3 ≥ 15 | 3.7 ≥ 13 | 4.0 ≥ 12 |

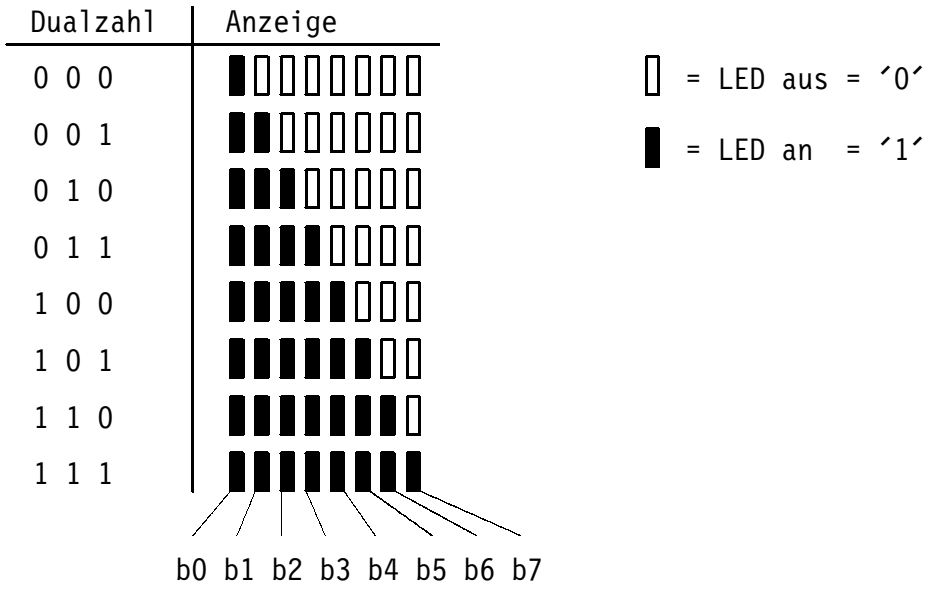
(2.1) Finden Sie eine minimale Realisierung der Funktion $y_0 = F_0(x_0, x_1, x_2)$ in disjunktiver Normalform. Zeichen Sie die Schaltung nur mit Invertern, UND- und ODER-Gattern.

(2.2) Finden Sie eine minimale Realisierung der Funktion $y_1 = F_1(x_0, x_1, x_2)$ in konjunktiver Normalform. Zeichen Sie die Schaltung nur mit Invertern, UND- und ODER-Gattern.

(2.3) Realisieren Sie $\bar{y}_1$ (Invertierung von $y_1$). Wenden Sie zu diesem Zweck auf $F_1$ das DeMorgansche Theorem an. Welche(s) Gatter wird/werden nun benötigt (kein Zeichnen der Schaltung erforderlich)?

## (3) Leuchtbandanzeige mit LEDs

Zur Anzeige von analogen Daten verwendet man gelegentlich sogenannte Leuchtbandanzeigen (z.B. als Drehzahlmesser in Formel-1-Autos, Aussteuerungsanzeige in Recordern). Es soll eine Schaltung zur Darstellung von 3-Bit-Zahlen entworfen werden.




(3.1) Entwerfen Sie eine Schaltung zur Verwirklichung der Ansteuersignale für die Segmente $b_1$, $b_2$, $b_3$, $b_6$ sowie $b_7$ (ignorieren Sie die übrigen Segmente). Sie können beliebige Gatter verwenden. Die Schaltung soll möglichst einfach werden.

(3.2) Zeichnen Sie die gesamte Schaltung für die Segmente $b_1$, $b_2$, $b_3$, $b_6$ sowie $b_7$.

## (4) Sequenzielle Schaltung

Ein 2-Bit Zähler (Ausgänge "00"→"01"→"10"→"11"→"00"→"01"→... mit jeder positiven Taktflanke) kann als Moore-Machine angesehen werden. Die Zustände sind

gleichzeitig die Ausgänge des Zustands-Automaten. Das einzige Eingangsgröße (neben *Clock* natürlich) ist ein *Reset*-Signal, das den Zähler in der nächsten positiven Taktflanke auf "00" setzt.

(4.1) Zeichnen Sie das Zustandsdiagramm mit allen 4 Zuständen.

(4.2) Entwerfen Sie die Eingangslogik $F(u, x)$.

(4.3) Zeichnen Sie die gesamte Schaltung für den 2-Bit-Zähler.

## (5) 8051-Assembler-Programm

Aus einem Feld mit einer bestimmten Anzahl Bytes soll der maximale Wert herausgefunden werden (nur Zahlen ohne Vorzeichen, d.h. Wertebereich ist 0..255). Dazu soll ein Unterprogramm in 8051-Assembler geschrieben werden, dass das Minimum im Akku zurückgibt. Das Unterprogramm "Maxim" wird folgendermaßen aufgerufen:

```
; Aufgabe (5) der Probeklausur SS03
; KPM/21-JUN-2003

            ORG 0000H
            mov       R0, #30h            ; Startadresse fuer Array
            mov       R1, #10h            ; Anzahl zu durchsuchender Bytes
            call      Maxim               ; Maximum in A
L1:         jmp       L1                  ; Arbeit getan, gehe in Endlosschleife
```

(5.1) Zeichnen Sie ein Struktogramm, dass Ihren Algorithmus zur Lösung der Aufgabe beschreibt.

(5.2) Kodieren Sie Ihren Algorithmus in 8051-Assembler.

**Hochschule Bremerhaven**

INSTITUT FÜR AUTOMATISIERUNGS- UND ELEKTROTECHNIK

Prof. Dr.-Ing. Kai Müller

# Übungsklausur DMP / PT SS03

Bearbeitungszeit 90 Minuten
– Unterlagen gestattet –

# Lösungen

## (1) Konversion zwischen Zahlensystemen und Rechnen mit Dualzahlen

(1.1)

```
148 : 2 = 74,          Rest = 0
74 : 2 = 37,           Rest = 0
37 : 2 = 18,           Rest = 1
18 : 2 = 9,                      Rest = 0
9 : 2 = 4,             Rest = 1
4 : 2 = 2,             Rest = 0
2 : 2 = 1,             Rest = 0
1 : 2 = 0,             Rest = 1   (Ende der Berechnung)
⇒  x = 10010100₂  (aus der Bestimmung der Reste)

19 : 2 = 9,                      Rest = 1
9 : 2 = 4,             Rest = 1
4 : 2 = 2,             Rest = 0
2 : 1 = 1,             Rest = 0
1 : 2 = 0,             Rest = 1
⇒  y = 10011₂  (aus der Bestimmung der Reste)
```

(1.2)

```
10010100 x 10011
        10010100
       10010100
      00000000
     00000000
    10010100
----------------
    101011111100 = z
================
```

(1.3) Zur Bildung der Hexadezimalzahl können immer vier Stellen zu einer Hexadezimalziffer zusammengefasst werden.
$z$ = '1010'1111'1100$_2$ = AFC$_{16}$.

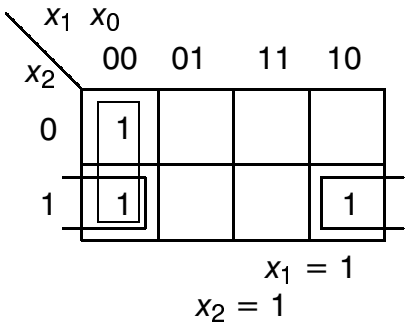
(1.4) Erweiterung auf 16 Bit: z = 0000101011111100.
Bildung des Zeierkomplements

| (1) = 5 | (2) = 5 | (3) = 5 | (4) = 5 | (5) = 10 | Σ = 30 |
|---|---|---|---|---|---|
| 1.0 ≥ 27 | 1.3 ≥ 25 | 1.7 ≥ 23 | 2.0 ≥ 21 | 2.3 ≥ 19 | 2.7 ≥ 18 | 3.0 ≥ 17 | 3.3 ≥ 15 | 3.7 ≥ 13 | 4.0 ≥ 12 |

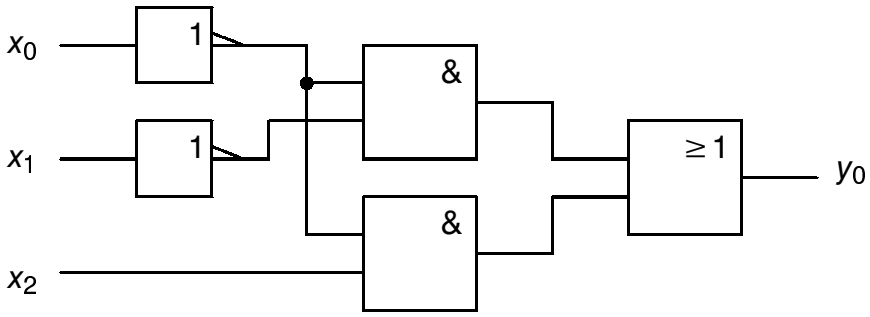
```
  0000101011111100
  1111010100000011    (Inversion)
+ 0000000000000001    (Addtion von 1)
------------------
  1111010100000100    (Ergebnis -z)
```

## (2) Minimierung logischer Funktionen mit KV-Diagramm

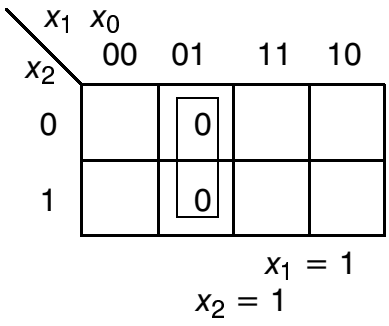
(2.1) KV-Diagramm




$$y_0 = \bar{x}_1 \wedge \bar{x}_0 \;\vee\; x_2 \wedge \bar{x}_0$$

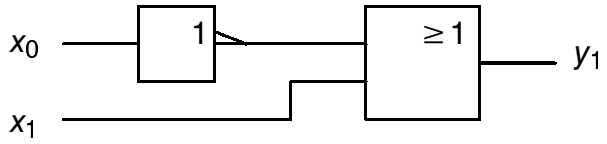



(2.2) KV-Diagramm




Nur ein Term, deshalb keine “Verundung” nötig:

$$y_1 = x_1 \vee \bar{x}_0$$




(2.3) $\bar{y}_1 = \overline{x_1 \vee \bar{x}_0} = \bar{x}_1 \wedge x_0$

Es wird ein UND-Gatter mit 2 Eingängen benötigt.

## (3) Leuchtbandanzeige mit LEDs

(3.1) Die boolschen Gleichungen können unmittelbar aus der Wahrheitstabelle

| x2 | x1 | x0 | b7 | b6 | b3 | b2 | b1 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

abgeleitet werden. Eine Ausnahme ist $b_3$, für die ein KV-Diagramm hilfreich ist.

$b_1 = x_2 \vee x_1 \vee x_0\,,$ (KNF)

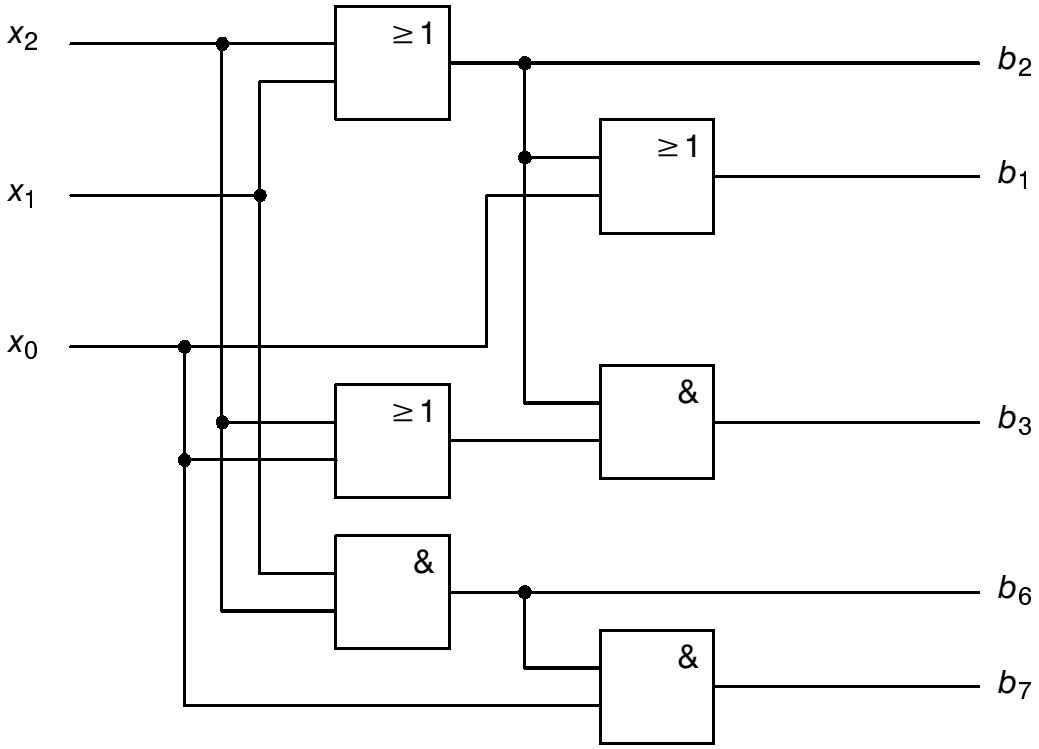
$b_2 = x_2 \vee x_1\,,$ (KNF)

$b_3 = (x_2 \vee x_1) \wedge (x_2 \vee x_0)\,,$ (KNF)

$b_6 = x_2 \wedge x_1\,,$ (DNF)

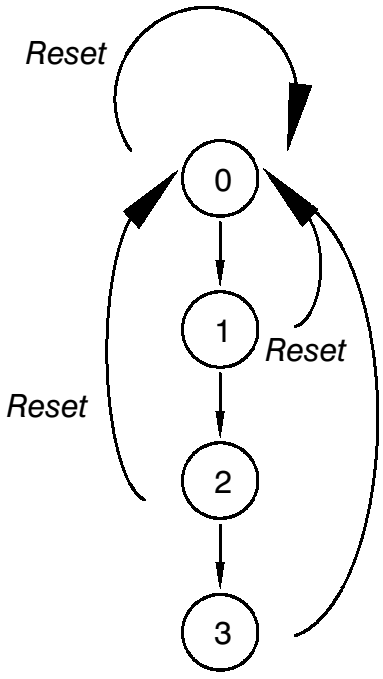
$b_7 = x_2 \wedge x_1 \wedge x_0\,.$ (DNF)

(3.2) Dargestellt ist nur eine Möglichkeit.




## (4) Sequenzielle Schaltung

(4.1)




(4.2)

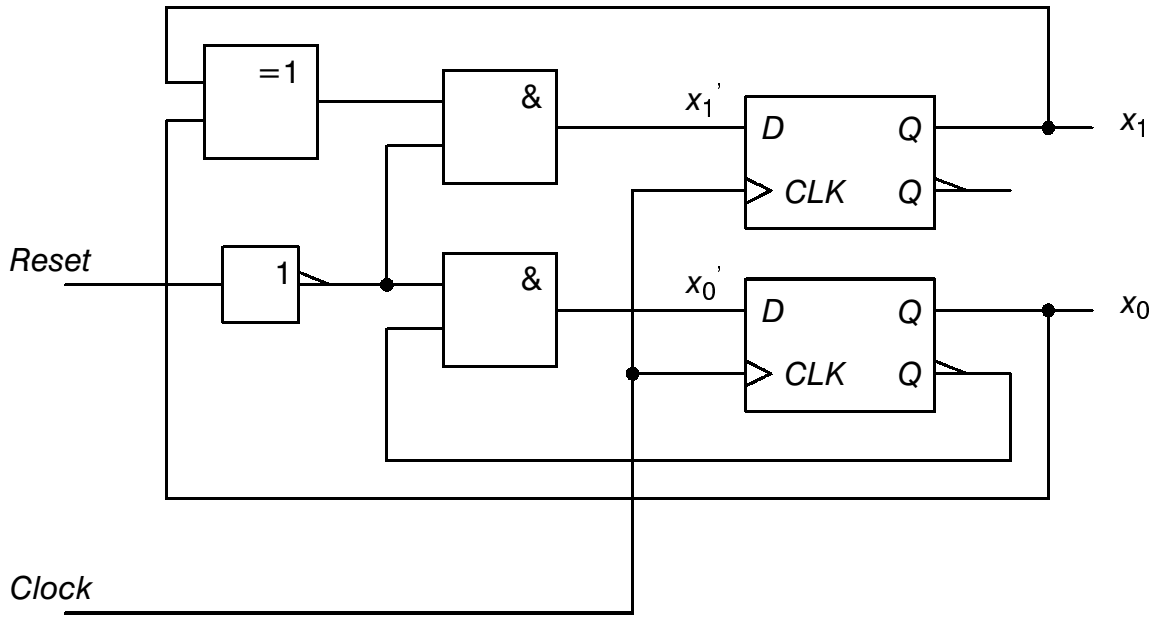
| Reset | x1 | x0 | x1’ | x0’ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | x | x | 0 | 0 |

Boolsche Gleichungen:

$$x_1' = \overline{Reset} \wedge (x_1 \oplus x_0) \qquad \text{(XOR)}$$

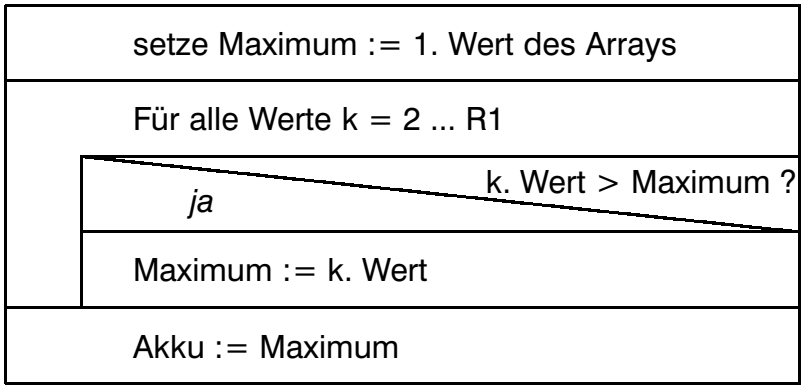
$$x_0' = \overline{Reset} \wedge \bar{x}_0$$

(4.3)




## (5) 8051-Assembler-Programm

(5.1) Struktogramm (abhängig vom gewählten Algorithmus) muss mit der Lösung in (5.2) übereinstimmen.




(5.2) Vollständiges Assember-Programm

```
LOC   OBJ         LINE  SOURCE

                     1  # 1 "a5.asm"
                     1  ; Loesung der Aufgabe 5. (Probeklausur SS03)
                     2  ; KPM/21-JUN-2003
                     3
                     4          ORG 0000H
0000: 7830           5          mov       R0, #30h   ; Startadresse Array
0002: 7910           6          mov       R1, #10h   ; Anzahl zu durchsuchender
                                                     ; Bytes
0004: 1108           7          call      Maxim      ; Maximum in A
0006: 80FE           8  L1:     jmp       L1         ; Arbeit getan, gehe in
                                                     ; Endlosschleife
                     9
                    10  ;
                    11  ; Aufsuchen des maximalen Wertes in einem Array von Bytes
                    12  ; -> R0: Anfangsadresse des Arrays
                    13  ; -> R1: Anzahl Bytes des Arrays
                    14  ; <- A:  Maximum des Feldes
                    15  ; -- R2: enthält ebenfalls Maximum
                    16  ;
0008: E6            17  Maxim:   mov       A, @R0
0009: FA            18          mov       R2, A      ; R2 = bisheriges Minimum
000A: 19            19          dec       R1         ; mit R1-1 Zahlen vergleichen
                    20
000B: 08            21  MAX_1:   inc       R0
000C: C3            22          clr       C          ; subb subtrahiert immer
                                                     ; Carry Flag
000D: 96            23          subb      A, @R0
000E: 5002          24          jnc       MAX_2      ; Zahl <= R2
0010: E6            25          mov       A, @R0     ; neues Maximum setzen
0011: FA            26          mov       R2, A
0012: EA            27  MAX_2:   mov       A, R2
0013: D9F6          28          djnz      R1, MAX_1
                    29
0015: 22            30          ret
```

```
                    31
0016:               32          END
```