

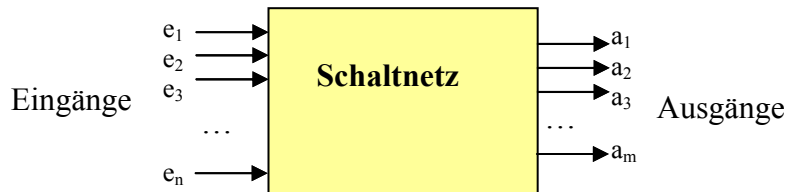
# Schaltnetze: Test und Minimierung

## **Inhalt:**

1. Schaltnetze
  - 1.1 Schaltwerttabelle und Schaltfunktionen
  - 1.2 Vereinfachung von Schaltfunktionen
    - 1.2.1 Die Gesetze der Schaltalgebra
    - 1.2.2 Vereinfachung durch Algebra
    - 1.2.2 Vereinfachung nach Quine-McCluskey
  - 1.3 Die Überprüfung von Schaltfunktionen
    - 1.3.1 Schaltwerttabellen mit Delphi
    - 1.3.2 Schaltwerttabellen mit Java
  - 1.4 Gatterdarstellung und Simulation
  - 1.5 Aufgaben

# 1. Schaltnetze

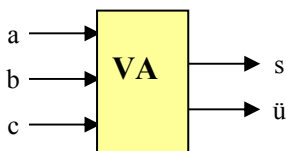
Unter einem Schaltnetz versteht man eine Verknüpfung von Schaltern, die auf der einen Seite Eingangsleitungen  $e_1, e_2, \dots, e_n$  hat, die mit den Werten 0 bzw. 1 belegt werden können, und auf der anderen Seite Ausgangsleitungen  $a_1, a_2, \dots, a_m$ , die die gleichen Werte annehmen. Die Ausgangswerte sind ausschließlich durch die Eingangswerte bestimmt. (Das unterscheidet Schaltnetze von Schaltwerken, die auch innere Zustände besitzen und deshalb unterschiedlich auf die Eingangswerte reagieren können.)



Man beschreibt Schaltnetze, indem man zu jeder möglichen Eingangsbelegung die gewünschte Ausgangsbelegung in Form einer Tabelle aufschreibt. Da an einem Schaltnetz mit  $n$  Eingängen nur  $2^n$  unterschiedliche Eingangsbelegungen möglich sind, ist die Tabelle endlich und kann zumindest prinzipiell vollständig aufgeschrieben werden. Es ist Aufgabe der **Schaltalgebra**, den Zusammenhang zwischen Ein- und Ausgängen funktional zu beschreiben und die gefundenen Terme zu vereinfachen. Für jeden Ausgang muss eine eigene Schaltfunktion gefunden werden, die als Argumente die Belegung der Eingänge erhält. Aus den Funktionen werden dann systematisch Schaltungen erzeugt.

## 1.1 Schaltwerttabelle und Schaltfunktionen

Da Ein- und Ausgänge jeweils zwei unterschiedliche Werte annehmen können (0 und 1), beschreibt man deren Zustand durch **Schaltvariable**. Die Eingangsvariablen bezeichnen wir mit  $e_i$  ( $i=1..n$ ), die Ausgangsvariablen mit  $a_j$  ( $j = 1..m$ ). Als Beispiel wollen wir einen Volladdierer VA beschreiben, der die Summe seiner drei Eingänge ( $a, b$  und  $c$ ) über zwei Ausgänge  $s$  (Summe) und  $\bar{u}$  (Übertrag) anzeigt. Bei solch speziellen Schaltnetzen ist es üblich, die Eingänge mit  $a, b, c, \dots$  zu bezeichnen und auch den Ausgängen Namen zu geben, die ihrer Funktion entsprechen. Da wir drei Eingänge haben, gibt es genau  $2^3 = 8$  unterschiedliche Eingangsbelegungen. Die können wir leicht aufschreiben – der Übersicht halber in der üblichen binären Zählweise:



a	b	c	ü	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Jede Zeile der Tabelle ist eindeutig gekennzeichnet durch die Kombination der Werte der Schaltvariablen auf der linken Seite. Man bezeichnet solche Kombinationen als *Miniterme*:

**Ein Miniterm stellt eine Kombination aus allen auftretenden Schaltvariablen dar, die durch das logische UND ( $\wedge$ ) verknüpft werden. Die Variablen treten entweder einfach oder in der negierten Form auf**

Miniterme mit drei Variablen sind z. B. (die Negation wird durch einen Strich über der Variablen gekennzeichnet):

$$(a) \ a \wedge \bar{b} \wedge c \quad (b) \ \bar{a} \wedge \bar{b} \wedge c \quad (c) \ \bar{a} \wedge b \wedge c$$

Ein Miniterm liefert genau dann den Wert 1, wenn seine einfach auftretenden Variablen alle den Wert 1, die negierten alle den Wert 0 haben. Deshalb lassen sich die Zeilen der Tabelle durch die entsprechenden Miniterme eindeutig kennzeichnen: (a) entspricht der sechsten Zeile der angegebenen Tabelle, (b) der zweiten und (c) der vierten. Verknüpfen wir mehrere Miniterme durch logische ODER, dann erhalten wir einen Term, der nur dann 1 ergibt, wenn mindestens einer der auftretenden Miniterme eine 1 liefert. Da die Miniterme eindeutig den Tabellenzeilen zugeordnet sind, müssen wir nur die Miniterme „aufzählen“, die den Einsen der rechten Tabellenspalten entsprechen, um eine korrekte Schaltfunktion für diese Spalte zu erhalten. (In allen anderen Fällen liefern alle Miniterme die 0.)

Für unseren Volladdierer bedeutet das:

$$\begin{aligned} \ddot{u}(a,b,c) &= (\bar{a} \wedge b \wedge c) \vee (a \wedge \bar{b} \wedge c) \vee (a \wedge b \wedge \bar{c}) \vee (a \wedge b \wedge c) \\ s(a,b,c) &= (\bar{a} \wedge \bar{b} \wedge c) \vee (\bar{a} \wedge b \wedge \bar{c}) \vee (a \wedge \bar{b} \wedge \bar{c}) \vee (a \wedge b \wedge c) \end{aligned}$$

Die ODER-verknüpften Miniterme bilden die **disjunktiven Normalformen** der Schaltfunktionen für die Ausgänge.

## 1.2 Vereinfachung von Schaltfunktionen

Schaltfunktionen in der disjunktiven Normalform lassen sich direkt den Schaltwerttabellen entnehmen und auch leicht auf Schreibfehler kontrollieren (wenn man sich an die „richtige“ Reihenfolge hält). Sie sind aber meist viel zu kompliziert. Da wir möglichst einfache Schaltungen entwickeln wollen, versuchen wir die Schaltfunktionen zu vereinfachen. Das kann natürlich nur dann funktionieren, wenn es unterschiedliche Terme gibt, die die gleiche Funktion beschreiben. Ähnlich wie in der „üblichen“ Schulmathematik ist das der Fall.

### 1.2.1 Die Gesetze der Schaltalgebra

Für Terme der Schaltalgebra gelten ähnlich wie in der „normalen“ Algebra Gesetze, mit deren Hilfe wir Termumformungen durchführen können. Wir *beweisen* die Gesetze, indem wir Schaltwerttabellen mit allen Möglichkeiten aufstellen. Da diese endlich sind, ist dieses „Ausprobieren“ tatsächlich ein Beweis. Es gelten

1. Kommutativgesetze:  $a \wedge b = b \wedge a$  bzw.  $a \vee b = b \vee a$

a	b	$a \wedge b$	$b \wedge a$
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

a	b	$a \vee b$	$b \vee a$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Da die beiden Spalten die gleichen Werte enthalten, ist der Beweis vollbracht!

2. Assoziativgesetz:  $a \wedge b \wedge c = a \wedge (b \wedge c) = (a \wedge b) \wedge c$

a	b	c	$a \wedge b \wedge c$	$a \wedge (b \wedge c)$	$(a \wedge b) \wedge c$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	1	1	1

$a \vee b \vee c = a \vee (b \vee c) = (a \vee b) \vee c$

a	b	c	$a \vee b \vee c$	$a \vee (b \vee c)$	$(a \vee b) \vee c$
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

3. Distributivgesetz:  $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$

a	b	c	$a \wedge (b \vee c)$	$(a \wedge b) \vee (a \wedge c)$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$$

a	b	c	$a \wedge (b \vee c)$	$(a \vee b) \wedge (a \vee c)$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

4. Gesetze von de Morgan:  $\overline{a \wedge b} = \bar{a} \vee \bar{b}$  bzw.  $\overline{a \vee b} = \bar{a} \wedge \bar{b}$

a	b	$\overline{a \wedge b}$	$\bar{a} \vee \bar{b}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

a	b	$\overline{a \vee b}$	$\bar{a} \wedge \bar{b}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

5. Absorptionsgesetze:  $a \wedge (a \vee b) = a$  bzw.  $a \vee (a \wedge b) = a$   
 6. Gesetze der neutralen Elemente:  $a \wedge 1 = a$  bzw.  $a \vee 0 = a$   
 7. Gesetze der komplementären Elemente:  $a \wedge \bar{a} = 0$  bzw.  $a \vee \bar{a} = 1$   
 8. Idempotenzgesetze:  $a \wedge a = a$  bzw.  $a \vee a = a$

### 1.2.2 Vereinfachung durch Algebra

Mithilfe der Gesetze der Schaltalgebra können Schaltterme systematisch vereinfacht werden. Ziel ist es meist, durch Ausklammern etc. und durch Anwendung der Gesetze der neutralen und komplementären Elemente Nullen oder Einsen zu erzeugen, die dann zum Wegfall von Termen oder einzelnen Schaltvariablen führen. Ausgangspunkt sind nach Möglichkeit Miniterme, die sich nur in einer Schaltvariablen unterscheiden. Wir wollen als Beispiel die Schaltfunktion  $\ddot{u}$  des Volladdierers vereinfachen. Dazu beachten wir, dass Miniterme mehrfach zur Vereinfachung herangezogen werden können. Wir benutzen den letzten Term dreimal an:

$$\begin{aligned}
 \ddot{u}(a,b,c) &= (\bar{a} \wedge b \wedge c) \vee (a \wedge \bar{b} \wedge c) \vee (a \wedge b \wedge \bar{c}) \vee (a \wedge b \wedge c) \\
 &= (\bar{a} \wedge (b \wedge c)) \vee (a \wedge (\bar{b} \wedge c)) \vee (a \wedge (b \wedge \bar{c})) \vee (a \wedge (b \wedge c)) \\
 &= ((\bar{a} \vee a) \wedge (b \wedge c)) \vee ((\bar{b} \vee b) \wedge (a \wedge c)) \vee ((\bar{c} \vee c) \wedge (a \wedge b)) \\
 &= (1 \wedge (b \wedge c)) \vee (1 \wedge (a \wedge c)) \vee (1 \wedge (a \wedge b)) \\
 &= (b \wedge c) \vee (a \wedge c) \vee (a \wedge b)
 \end{aligned}$$

Die Schaltfunktion der Summe lässt sich nicht weiter vereinfachen.

### 1.2.2 Vereinfachung nach Quine-McCluskey

Da der Umgang mit den Gesetzen der Schaltalgebra etwas ungewohnt ist, lohnt es sich ein systematisches Verfahren zur Schaltungsvereinfachung einzusetzen. Wir nummerieren dazu die Minterme der disjunktiven Normalform binär durch:

$$\begin{aligned} \bar{a} \wedge \bar{b} \wedge \bar{c} &\text{ entspricht } 0 \text{ (000)} \\ \bar{a} \wedge \bar{b} \wedge c &\text{ entspricht } 1 \text{ (001)} \\ \bar{a} \wedge b \wedge \bar{c} &\text{ entspricht } 2 \text{ (010) usw.} \end{aligned}$$

Danach schreiben wir sie absteigend oder aufsteigend geordnet untereinander auf. Dabei trennen wird die Gruppen von Mintermen, die sich in der Anzahl der negierten Schaltvariablen unterscheiden, jeweils durch einen Querstrich. Für die Übertragungsfunktion des Volladdierers erhalten wir (absteigend geordnet) zwei Gruppen:

$$\ddot{u}(a,b,c) = (\bar{a} \wedge b \wedge c) \vee (a \wedge \bar{b} \wedge c) \vee (a \wedge b \wedge \bar{c}) \vee (a \wedge b \wedge c)$$

Kennziffer	Minterm
7	$a \wedge b \wedge c$
6	$a \wedge b \wedge \bar{c}$
5	$a \wedge \bar{b} \wedge c$
3	$\bar{a} \wedge b \wedge c$

Jetzt prüfen wir, ob sich zwei Minterme nur in einer Variablen unterscheiden. Solche Terme können sich nur in benachbarten Gruppen befinden. Wir kombinieren die dualen Kennziffern (z.B. zu 76 wenn wir die Terme 7 und 6 miteinander verknüpfen) und notieren die vereinfachten Terme. Benutzte Minterme werden „abgehakt“. Unbenutzte Terme werden in die zweite Tabelle übernommen.

Kennziffer	Minterm (1. Liste)	benutzt?	Kennziffer	Term (2. Liste)
7	$a \wedge b \wedge c$	ja	76	$a \wedge b$
6	$a \wedge b \wedge \bar{c}$	ja	75	$a \wedge c$
5	$a \wedge \bar{b} \wedge c$	ja	73	$b \wedge c$
3	$\bar{a} \wedge b \wedge c$	ja		

Schon nach diesem ersten Schritt wurden einerseits alle Minterme benutzt, andererseits eine einzelne Gruppe erzeugt in der alle Terme gleich viel negierte Variable (gar keine) haben. Der Vereinfachungsprozess ist damit abgeschlossen.

Wir prüfen jetzt nur noch, ob wir auch überflüssige Terme gefunden haben. Dazu notieren wir die Nummern der Ausgangs-Minterme in einer Tabelle oben am Rand und die gefundenen vereinfachten Terme links. Aus der Kennziffer der Vereinfachung ergibt sich, welche Ausgangsterme benutzt wurden. Benötigt werden nur so viele Terme, dass alle Ausgangsterme abgedeckt sind.

Kennziffer	Minterm	3	5	6	7
76	$a \wedge b$			✓	✓
75	$a \wedge c$		✓		✓
73	$b \wedge c$	✓			✓

Zwar wird der Minterm 7 von allen drei Termen abgedeckt, die anderen aber nur jeweils von einem. Es sind also keine Terme überflüssig.

### 1.3 Die Überprüfung von Schaltfunktionen

Beim Vereinfachen von Schaltfunktionen unterlaufen einem manchmal Fehler. Setzen wir eine fehlerhafte Schaltfunktion in eine Gatterdarstellung und später in eine Schaltung aus echter Hardware um, dann kann diese gar nicht wie erwartet funktionieren. Es empfiehlt sich deshalb, schon die vereinfachte Schaltfunktion durch ein Programm zu überprüfen.

Die Schaltvariablen lassen sich direkt booleschen Variablen zuordnen. Benutzen wir diese als Laufvariable von Zählschleifen, dann erzeugen wir schnell alle möglichen Kombinationen der Eingangswerte durch eine Schachtelung von Zählschleifen. Die Schaltfunktionen selbst lassen sich direkt in das Programm eingeben, da logische Operatoren in alle Programmiersprachen eingebaut sind. Wenn wir nicht *false* und *true*, sondern *0* und *1* als Ausgabewerte in einer Tabelle wollen, dann setzen wir die booleschen Werte in diese Zahlen um. Als Beispiel wählen wir wieder unseren Volladdierer.

schreibe einen Tabellenkopf mit Überschriften, Linien, ...													
FÜR a VON false BIS true TUE													
<table border="1"> <tr> <td colspan="2">FÜR b VON false BIS true TUE</td> </tr> <tr> <td colspan="2"> <table border="1"> <tr> <td colspan="2">FÜR c VON false BIS true TUE</td> </tr> <tr> <td colspan="2">s ← (not a and not b and c) or (not a and b and not c) or (a and not b and not c) or (a and b and c)</td> </tr> <tr> <td colspan="2">ü ← (a and b) or (a and c) or (b and c)</td> </tr> <tr> <td colspan="2">schreibe eine Tabellenzeile mit a,b,c und ü und s</td> </tr> </table> </td> </tr> </table>		FÜR b VON false BIS true TUE		<table border="1"> <tr> <td colspan="2">FÜR c VON false BIS true TUE</td> </tr> <tr> <td colspan="2">s ← (not a and not b and c) or (not a and b and not c) or (a and not b and not c) or (a and b and c)</td> </tr> <tr> <td colspan="2">ü ← (a and b) or (a and c) or (b and c)</td> </tr> <tr> <td colspan="2">schreibe eine Tabellenzeile mit a,b,c und ü und s</td> </tr> </table>		FÜR c VON false BIS true TUE		s ← (not a and not b and c) or (not a and b and not c) or (a and not b and not c) or (a and b and c)		ü ← (a and b) or (a and c) or (b and c)		schreibe eine Tabellenzeile mit a,b,c und ü und s	
FÜR b VON false BIS true TUE													
<table border="1"> <tr> <td colspan="2">FÜR c VON false BIS true TUE</td> </tr> <tr> <td colspan="2">s ← (not a and not b and c) or (not a and b and not c) or (a and not b and not c) or (a and b and c)</td> </tr> <tr> <td colspan="2">ü ← (a and b) or (a and c) or (b and c)</td> </tr> <tr> <td colspan="2">schreibe eine Tabellenzeile mit a,b,c und ü und s</td> </tr> </table>		FÜR c VON false BIS true TUE		s ← (not a and not b and c) or (not a and b and not c) or (a and not b and not c) or (a and b and c)		ü ← (a and b) or (a and c) or (b and c)		schreibe eine Tabellenzeile mit a,b,c und ü und s					
FÜR c VON false BIS true TUE													
s ← (not a and not b and c) or (not a and b and not c) or (a and not b and not c) or (a and b and c)													
ü ← (a and b) or (a and c) or (b and c)													
schreibe eine Tabellenzeile mit a,b,c und ü und s													

#### 1.3.1 Schaltwerttabellen mit Delphi

Da wir nur schnell eine Tabelle erzeugen und anzeigen wollen, fügen wir in ein Formular eine Memo-Komponente ein, die die Tabelle darstellt. Wird das Formular erzeugt (und das Ereignis `onCreate` ausgelöst), dann kann sofort losgezeichnet werden.

```

procedure TForm1.FormCreate(Sender: TObject);
var a,b,c,s,ue: boolean; h: string;
begin
  memo1.lines.clear;
  memo1.Lines.Add(' a | b | c || ü | s ');
  memo1.Lines.add('-----');
  for a:= false to true do
    for b := false to true do
      for c := false to true do
        begin
          s := (not a and not b and c) or (not a and b and not c)
              or (a and not b and not c) or (a and b and c);
          ue := (a and b) or (a and c) or (b and c);
          if a then h := ' 1 |' else h := ' 0 |';
          if b then h := h + ' 1 |' else h := h + ' 0 |';
          if c then h := h + ' 1 ||' else h := h + ' 0 ||';
          if ue then h := h + ' 1 |' else h := h + ' 0 |';
          if s then h := h + ' 1 ' else h := h + ' 0 ';
          memo1.lines.add(h);
        end
      end
    end
  end;
end;

```

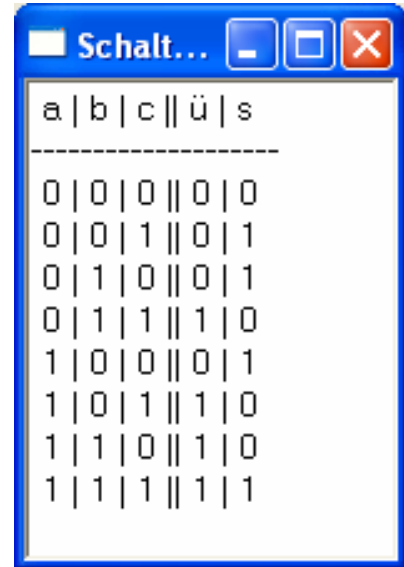
a	b	c	ü	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

### 1.3.2 Schaltwerttabellen mit Java

Da wir nur schnell eine Tabelle erzeugen und anzeigen wollen, fügen wir in ein Formular eine *RichEdit*-Komponente ein, die die Tabelle darstellt. Wird das Formular erzeugt (im Konstruktor *Schaltwerttabelle()*), dann kann sofort losgezeichnet werden.

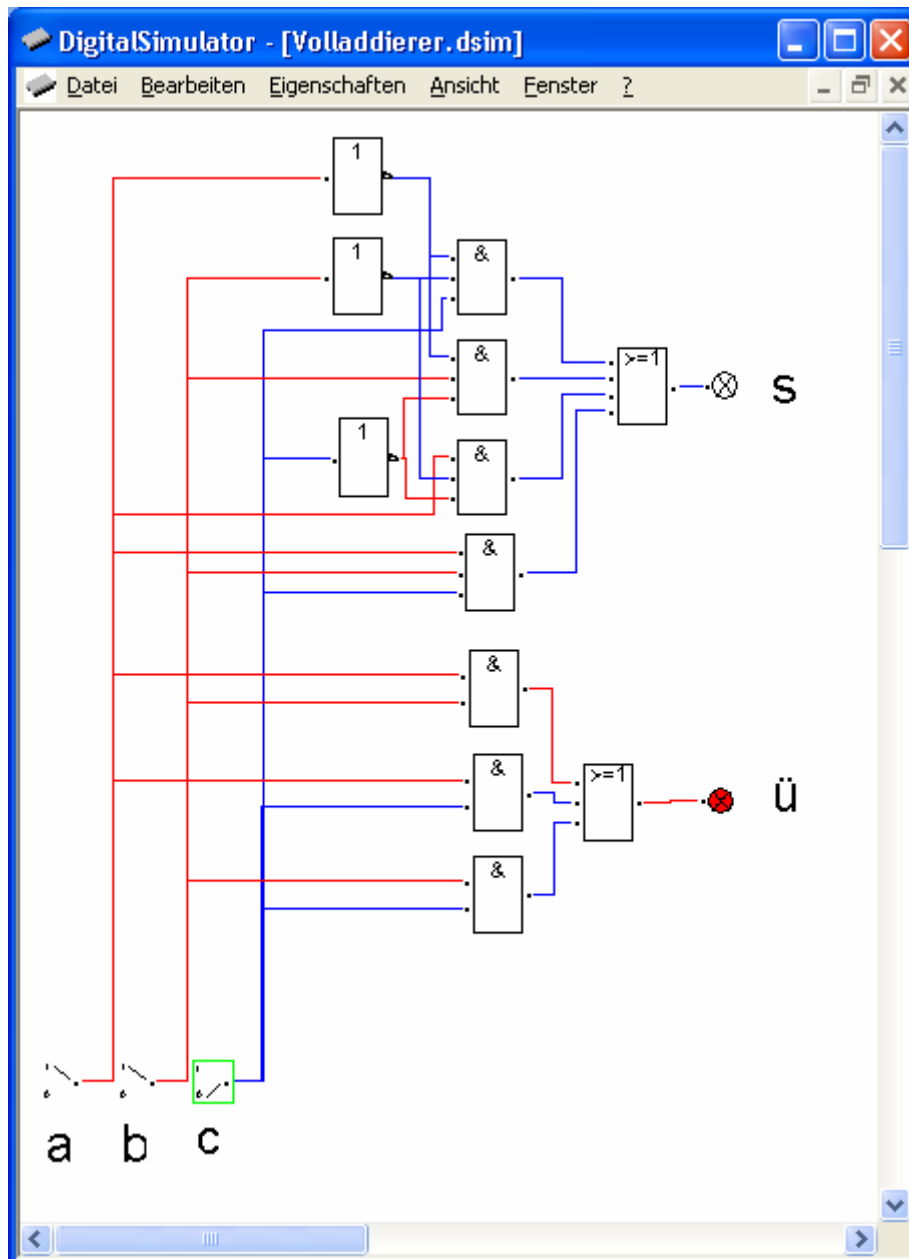
```
public Schaltwerttabelle()
{
    super();
    initForm();

    richEdit1.clear();
    String[] zeilen = new String[10];
    String h;
    boolean a,b,c,s,ue;
    int i;
    zeilen[0] = " a | b | c || ü | s ";
    zeilen[1] = "-----";
    i = 2;
    a = false;
    do
    {
        b = false;
        do
        {
            c = false;
            do
            {
                s = (!a && !b && c) || (!a && b && !c) ||
                    (a && !b && !c) || (a && b && c);
                ue = (a && b) || (a && c) || (b && c);
                if(a) h = " 1 |"; else h = " 0 |";
                if(b) h = h + " 1 |"; else h = h + " 0 |";
                if(c) h = h + " 1 ||"; else h = h + " 0 ||";
                if(ue)h = h + " 1 |"; else h = h + " 0 |";
                if(s) h = h + " 1 "; else h = h + " 0 ";
                zeilen[i] = h;
                i++;
                c = !c;
            }
            while (c != false);
            b = !b;
        }
        while(b != false);
        a = !a;
    }
    while(a != false);
    richEdit1.setLines(zeilen);
}
}
```



## 1.4 Gatterdarstellung und Simulation

An dieser Stelle der Arbeit sollten wir über eine Schaltfunktion verfügen, die das gesuchte Schaltnetz vollständig und richtig beschreibt. Wir setzen es jetzt in eine Gatterdarstellung um, am besten mithilfe eines Logiksimulators, anhand dessen wir die Funktion direkt überprüfen können. Hier benutzen wir den *DigitalSimulator von A. Herz*. Als Beispiel dient wieder der Volladdierer.



Auch hier überprüfen wir die Schaltung, indem wir an den Eingabeschaltern alle Eingangsbelegungen durchprobieren. Arbeitet sie korrekt, dann setzen wir sie in richtige Hardware um. In dieser können jetzt nur noch Schaltfehler, technische Probleme durch Laufzeiten der Signale, Schaltzeiten, ... auftreten – und das sind immer noch genug.

## 1.5 Aufgaben

Behandeln Sie die folgenden Aufgaben in den Schritten

- Vereinfachung durch algebraische Umformungen – wenn möglich!
- Vereinfachung nach Quine-McCluskey – wenn möglich!
- Erzeugen einer Schaltwerttabelle für die ursprünglichen Terme und ggf. die vereinfachten durch ein Programm.
- Test durch eine Gatterdarstellung mithilfe eines Digitalsimulators.

1.  $f(a,b) = (\bar{a} \vee \bar{b}) \wedge (\bar{a} \vee b) \wedge (a \vee \bar{b})$

2.  $f(a,b,c) = (a \wedge b) \vee (a \wedge c) \vee (b \wedge \bar{c})$

3.  $f(a,b,c) = (\bar{a} \wedge \bar{b} \wedge \bar{c}) \vee (\bar{a} \wedge \bar{b} \wedge c) \vee (\bar{a} \wedge b \wedge c) \vee (a \wedge \bar{b} \wedge c) \vee (a \wedge b \wedge c)$

4.  $f(a,b,c) = (a \wedge b) \vee (a \wedge c) \vee (b \wedge \bar{c})$

5.  $f(a,b) = (a \wedge b) \vee (a \wedge \bar{b}) \vee (\bar{a} \wedge \bar{b})$

6.  $f(a,b) = (\bar{a} \vee \bar{b}) \wedge (\bar{a} \vee b) \wedge (a \vee b)$

7.  $f(a,b,c) = (a \wedge b) \vee (b \wedge \bar{c}) \vee (\bar{a} \wedge \bar{c})$

5.  $f(a,b,c) = (\bar{a} \wedge b \wedge \bar{c}) \vee (a \vee b) \wedge (\bar{a} \vee \bar{c})$

2. Entnehmen Sie der Schaltwerttabelle die angegebenen Schaltfunktionen, vereinfachen Sie sie und setzen Sie sie in eine Gatterdarstellung um.

a	b	c	f(a,b,c)	g(a,b,c)	h(a,b,c)
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	1	1

3. a: Ein *Halbsubtrahierer* bildet die Differenz zweier einstelliger Dualzahlen. Er arbeitet mit einer Differenz  $d$  und einer geborgten Ziffer  $g$ . Stellen Sie in Analogie zum Halbaddierer die Schaltwerttabelle auf und gewinne daraus eine Gatterdarstellung.
- b: Entwickeln Sie in Analogie zu Volladdierer einen *Vollsubtrahierer*, der z. B. in einem 3-Bit-Parallelsubtrahierwerk eingesetzt werden kann.