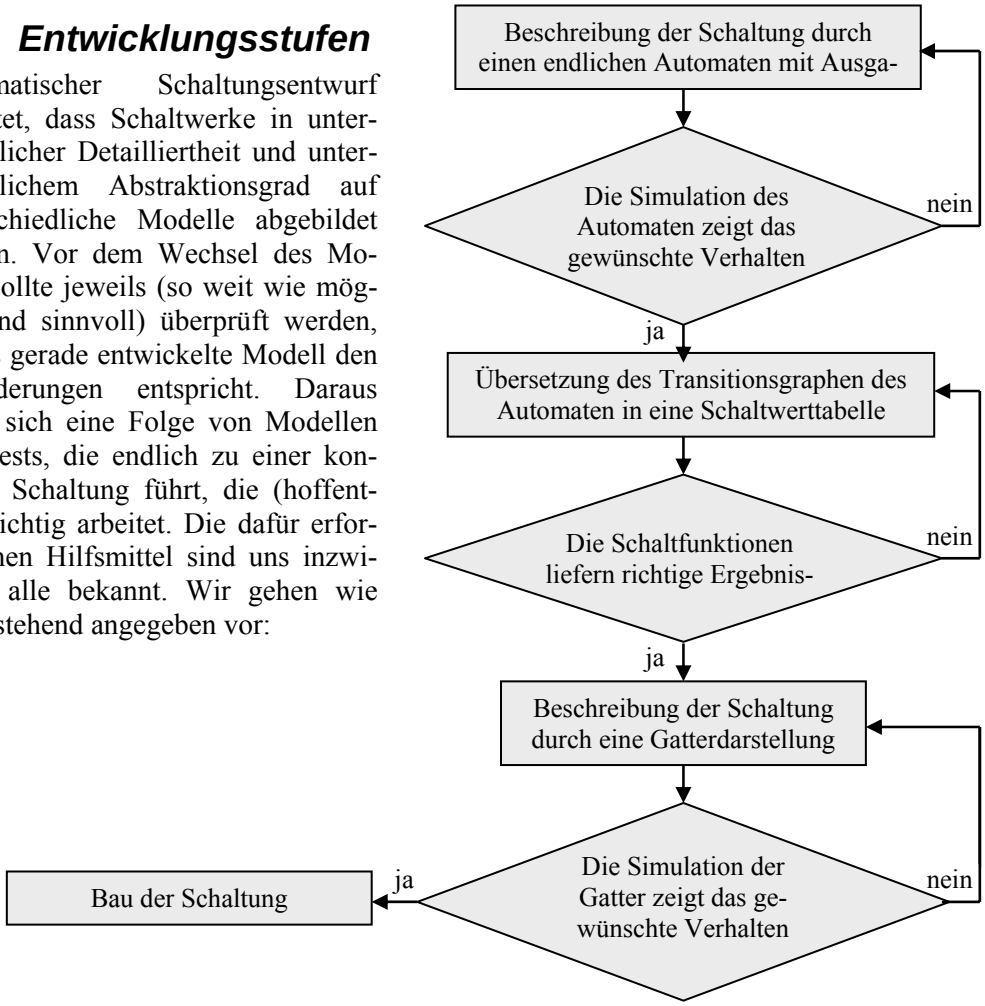
# 6 Schaltwerke und endliche Automaten

In diesem Abschnitt wird gezeigt, wie das Blockschaltbild aus 1.4 realisiert werden kann. Mithilfe der entwickelten Speicherbausteine und der Methode der Schaltnetzentwicklung kann jedes Schaltwerk erzeugt werden, wenn wir seine genaue Funktion im Automatenmodell beschreiben.[1]

## 6.1 Entwicklungsstufen

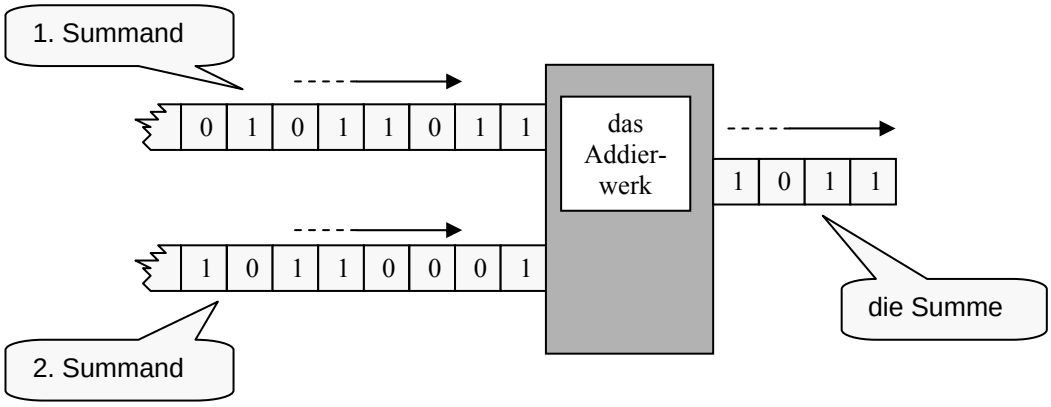
Systematischer Schaltungsentwurf bedeutet, dass Schaltwerke in unterschiedlicher Detailliertheit und unterschiedlichem Abstraktionsgrad auf unterschiedliche Modelle abgebildet werden. Vor dem Wechsel des Modells sollte jeweils (so weit wie möglich und sinnvoll) überprüft werden, ob das gerade entwickelte Modell den Anforderungen entspricht. Daraus ergibt sich eine Folge von Modellen und Tests, die endlich zu einer konkreten Schaltung führt, die (hoffentlich) richtig arbeitet. Die dafür erforderlichen Hilfsmittel sind uns inzwischen alle bekannt. Wir gehen wie nebenstehend angegeben vor:




[1] Praktisch beschränkt sich diese Beschreibung allerdings auf kleinere Schaltwerke.

## 6.2 Beispiel: Serienaddierwerk

Bevor wir weiter auf die Entwicklung von Schaltwerken eingehen, wollen wir zur Übung ein sehr einfaches Beispiel für die Beschreibung eines Schaltwerkes durch Automaten betrachten: gesucht ist das bekannte Serienaddierwerk. Es soll also ein Automat entwickelt werden, der zwei Dualzahlen stellenweise, beginnend mit den niederwertigsten Bits, addiert. Dabei werden die jeweils an dem Eingang des Automaten anliegenden beiden Dualziffern zum nächsten Bit des Ergebnisses zusammengefasst.




Ebenso wie bei dem BCD-Ziffern-Erkenner hängt auch hier das Ergebnis nicht nur von den beiden gerade zu addierenden Ziffern ab, sondern auch von einem Teilergebnis der vorhergehenden Addition - dem Übertrag. Da weitere zu merkende Größen nicht auftreten, können wir unseren Addierer durch einen Automaten beschreiben, der mit zwei Zuständen auskommt:

$s_0$: eine Null als Übertrag gespeichert (der Anfangszustand)

$s_1$: eine Eins als Übertrag gespeichert

Damit erhalten wir die Zustandsmenge $S = \{ s_0 , s_1 \}$ mit dem Anfangszustand $s_0$.

Wenden wir uns dem Eingabealphabet zu. In unserem Modell liest der Automat immer nur ein einziges Zeichen, unser Addierer dagegen erhält immer zwei Dualziffern als Eingabe. Wir müssen also die beiden Dualziffern als ein einziges Zeichen interpretieren! Das ist auch leicht möglich, wenn wir den vier auftretenden Kombinationen (*00*, *01*, *10* und *11*) an den Eingängen des Addierers vier einzelne Zeichen zuordnen. Dabei ist diese Zuordnung völlig willkürlich, z. B.:

$00 \leftrightarrow N$ (Null) $\quad 01 \leftrightarrow E$ (Eins) $\quad 10 \leftrightarrow A$ (Auch Eins) $\quad 11 \leftrightarrow Z$ (Zwei)

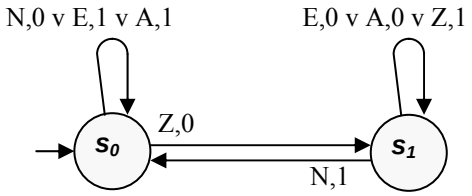
Wir finden damit das Eingabealphabet $E = \{ N , E , A , Z \}$

Die Arbeitsweise des Automaten ergibt sich, indem man die Anzahl der Einsen, die am Eingang des Automaten liegen, mit dem gespeicherten Wert des Übertrags (der sich aus dem Zustand ergibt) verarbeitet. Der Automat bleibt im Anfangszustand, wenn er die Zeichen *N* (*00*), *E* (*01*) oder *A* (*10*) liest, da dann kein Übertrag bei der Addition auftritt. Entsprechend bleibt er im Zustand $s_1$, wenn er ein *E*, *A* oder *Z* liest, da diese Eingaben zusammen mit dem Übertrag zu einem neuen Übertrag führen.

Der Automat wechselt von $s_0$ zu $s_1$, wenn er das Zeichen *Z* (*11*) liest und beim Zeichen *N* (*00*) in umgekehrter Richtung. Ein ausgezeichneter Endzustand ist aus der Aufgabenstellung nicht abzuleiten, da nach unserer Beschreibung ein Ende der Addition nicht angegeben wurde.

Das Ausgabealphabet besteht einfach aus den beiden Dualziffern. $A = \{ 0 , 1 \}$

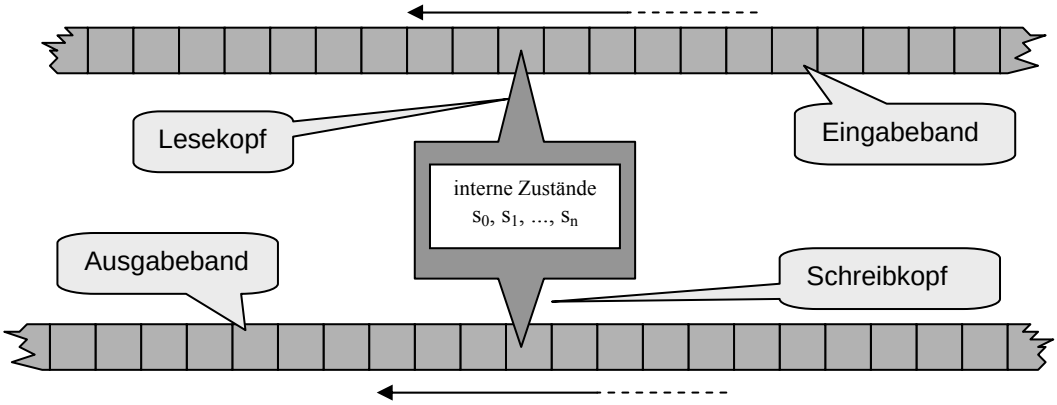
Das Zustandsdiagramm:




Da nur dann eine Ausgabe erfolgen kann, wenn vorher eine Eingabe vorgenommen wurde, kann bei *n* Eingabezeichen auch die Ausgabe nur *n* Zeichen umfassen. Die Eingabe sollte deshalb immer mit einer (führenden) Null abgeschlossen werden, damit der letzte für das Ergebnis zählende Übertrag auch noch ausgegeben wird.

## 6.3 Endliche Automaten und Schaltwerttabellen

Zur Erinnerung betrachten wir noch einmal das schon bekannte Modell eines endlichen Automaten $M = (E,A,S,s_0,\Sigma,u,g)$ mit einen Alphabeten und Funktionen:



| | |
|---|---|
| $E = \{ e_1, e_2, \ldots\ldots\ldots, e_r \}$ | das Eingabealphabet |
| $A = \{ a_1, a_2, \ldots\ldots\ldots, a_m \}$ | das Ausgabealphabet |
| $S = \{ s_0, s_1, \ldots\ldots\ldots, s_n \}$ | die Zustandsmenge |
| $s_0 \varepsilon S$ | den Anfangszustand |
| $\Sigma \subset S$ | die Menge der Endzustände |
| $u: (e_i, s_j) \rightarrow s_k$ | die Überführungsfunktion, mit ($1 \le i \le r,\ 0 \le j \le n,\ 0 \le k \le n$) |
| $g: (e_i, s_j) \rightarrow a_l$ | die Ausgabefunktion, mit ($1 \le i \le r,\ 0 \le j \le n,\ 1 \le l \le m$) |

Wir können die Beschreibung von Schaltwerken durch endliche Automaten als Erzeugung von Modellen des gesuchten Systems auffassen. Die Simulation der Automaten durch Programme entspricht dann dem Test des Modells. Nach diesem Test sollten wir einigermaßen sicher sein, dass unser Modell das Gewünschte leistet. Wir können dann auf dem Weg zur echten Technik fortschreiten. Da wir ein digitales System erzeugen wollen, benötigen wir eine Umsetzung der den Automaten beschreibenden Größen (Eingabealphabet, …) in eine digitale Darstellung. Wir codieren diese Größen.

1. Schritt: Codierung

Beschreiben müssen wir die *Eingabezeichen*, die *Ausgabezeichen* und die *Zustände* des Automaten. Als digitale Zeichen stehen uns nur die „*0*“ und die „*1*“ zur Verfügung. Wie kommen wir zu einer geeigneten Codierung? Als Beispiel wollen wir ein Eingabealphabet codieren, das drei unterschiedliche Zeichen enthält. Um diese unterscheiden zu können, benötigen wir zwei Bits, da diese in vier unterschiedlichen Kombinationen (also einer zu viel) auftreten können. Man kann mit zwei Bits maximal vier Zeichen codieren. Haben wir mehr Größen zu beschreiben, dann benötigen wir entsprechend mehr Bits. Bezeichnen wir mit $P_2(n)$ die Zweierpotenz (also 2, 4, 8, …), die größer oder gleich $n$ ist, dann ergibt sich die Anzahl $m$ der benötigten Bits aus $m = ld(P_2(n))$.

In unserem Fall: $P_2(3) = 4 \rightarrow m = ld(4) = 2$

Als Bezeichnungen für die Bits hat sich eingebürgert:

- die Bits der Eingabezeichen werden mit $e_0, e_1, \ldots$ bezeichnet
- die Bits der Ausgabezeichen werden mit $a_0, a_1, \ldots$ bezeichnet
- die Bits der Zustände werden mit $z_0, z_1, \ldots$ bezeichnet

*Wie* wir die Zuordnung zwischen Größen und Bitkombination wählen, ist eigentlich egal. Es bewährt sich allerdings meist, möglichst viele Nullen zu benutzen und „ähnliche“ Größen (z. B. Ein- und Ausgabezeichen oder bestimmte Zustände und Ausgabezeichen) auch „ähnlich“ zu codieren, damit die gefundenen Schaltungen sich dann auch „ähneln“, also relativ einfach werden, weil Teile mehrfach zu verwenden sind.

Unser Automat „berechnet“ aus der Kombination von aktuellem Zustand und dem zuletzt gelesenen Eingabezeichen des Eingabebands den Folgezustand und das nächste Ausgabezeichen. Diese Zuordnungen beschreiben wir durch die Überführungs- und die Ausgabefunktion. In der digitalen Version muss also die Überführungsfunktion alle $n$ Bits des nächsten Zustands berechnen. Damit besteht sie aus $n$ „Teilfunktionen“, die jeweils das entsprechende Bit liefern. Entsprechendes gilt für die Ausgabefunktion:

- die digitalen Überführungsfunktionen werden mit $u_0, u_1, \ldots$ bezeichnet.
- die digitalen Ausgabefunktionen werden mit $g_0, g_1, \ldots$ bezeichnet.

Liefert eine Überführungsfunktion in einem Fall z. B. $u(s_1,e_3) = s_2$ und haben wir die Zustände und Zeichen einfach durch die entsprechenden Dualzahlen codiert, dann lautet die digitale Version dieses Falls $u(01,11) = 10$. Da bei endlichen Automaten alle Mengen endlich sind, gibt es nur endlich viele Kombinationen aus Zustand und Eingabezeichen. Das nutzen wir aus!

### 2. Schritt: Aufstellen der Schaltwerttabelle

Schreiben wir alle Kombinationen aus codierten Zuständen und codierten Eingabezeichen zusammen mit den codierten Folgezuständen und Ausgabezeichen in Form einer Tabelle auf, dann haben wir eine vollständige digitale Beschreibung des Automaten gefunden.

| **Zustand** | | | | **Eingabezeichen** | | | **Folgezustand** | | | | **Ausgabezeichen** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $z_0$ | $z_1$ | $z_2$ | … | $e_0$ | $e_1$ | … | $u_0$ | $u_1$ | $u_2$ | … | $g_0$ | $g_1$ | … |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … |

Als Beispiel wollen wir den Serienaddierer so beschreiben:

Codierung der Mengen:

| Eingabe-zeichen | $e_0$ | $e_1$ |
|---|---|---|
| N | 0 | 0 |
| E | 0 | 1 |
| A | 1 | 0 |
| Z | 1 | 1 |

| Zustand | z |
|---|---|
| $s_0$ | 0 |
| $s_1$ | 1 |

| Ausgabe-zeichen | a |
|---|---|
| 0 | 0 |
| 1 | 0 |

Schaltwerttabelle:

| **Zustand** | **Eingabezeichen** | | **Folgezustand** | **Ausgabezeichen** |
|---|---|---|---|---|
| z | $e_0$ | $e_1$ | u | g |
| 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 |

Man sieht schnell, dass die Zeilenzahl *n* der Tabellenlänge abhängt von

*n = (Anzahl der Zustände) * (Anzahl der Eingabezeichen).*

Aus praktischen Gründen ist also die Zahl der Automaten, die sich wirklich in dieser Form noch „von Hand“ bearbeiten lassen, beschränkt.

3. Schritt: Entnehmen und Vereinfachen der Schaltfunktionen

4. Schritt: Gatterdarstellung der Schaltfunktionen

Die letzten beiden Schritte sind uns inzwischen gut bekannt. Es ist also möglich, jede durch einen endlichen Automaten beschreibbare Schaltung zu realisieren.

## *6.4 Blockschaltbild des Automaten*

Fassen wir unsere bisherigen Kenntnisse zusammen:

Ein als Schaltwerk realisierter Automat benötigt

- **Speicher**, um den momentanen Zustand dauerhaft zu repräsentieren. Für jedes Bit des Zustands wird ein eigener Speicher benötigt,
- **Schaltfunktionen**, die aus dem momentanem Zustand und dem nächsten Eingabezeichen den Folgezustand und das nächste Ausgabezeichen zu bestimmen,
- **Leitungen**, die die Bits des Eingabezeichens der Schaltung zuführen sowie die Bits des Ausgabezeichens aus der Schaltung leiten
- und einen **Takt**, der die Abläufe synchronisiert.

Wir erhalten damit das Blockschaltbild eines endlichen Automaten, der als digitales Schaltwerk realisiert wird: