Skriptum
„Schaltwerke und Rechnerorganisation“
WS 2002/03

Benedikt Meurer
bmeurer@unix-ag.org

29. Januar 2003

Inhaltsverzeichnis

1 Einleitung **3**
- 1.1 Schichtenmodell . . . 3
- 1.2 Programmhierarchie . . . 3
- 1.3 Begriffserklärung . . . 5

2 Grundlagen **6**
- 2.1 BOOLEsche Algebra . . . 6
- 2.2 Aussagenlogik . . . 6
- 2.3 Schaltalgebra . . . 7
- 2.4 BOOLEsche Funktionen . . . 7
  - 2.4.1 Schaltfunktionen . . . 8
  - 2.4.2 Darstellungsformen BOOLEscher Funktionen und ihre Umwandlungen . . . 12

3 Schaltnetze **13**
- 3.1 Beschreibung und Entwurfsmethodik . . . 13
  - 3.1.1 Entwurfsmethodik . . . 13
  - 3.1.2 Verfahren von QUINE/McCLUSKEY für Funktionsbündel 14
  - 3.1.3 Umformung von Schaltfunktionen . . . 14

4 Schaltwerke **15**
- 4.1 Beschreibung . . . 15
  - 4.1.1 Übergangsgraph . . . 16
- 4.2 Schaltwerksanalyse . . . 17
- 4.3 Elementare Schaltwerke . . . 18
  - 4.3.1 Schaltwerke zur Speicherung einer binären Variablen . . . 19
  - 4.3.2 Taktflankengesteuertes RS-Flipflop . . . 21
  - 4.3.3 Getaktetes Vorspeicher-Flipflop (Master-Slave-Flipflop) . 22
  - 4.3.4 JK-Flipflop . . . 24
  - 4.3.5 D-Flipflop . . . 25
- 4.4 Schaltwerksynthese . . . 26
  - 4.4.1 Ansteuergleichungen für Speicherglieder . . . 26
- 4.5 Spezielle Schaltwerke . . . 30
  - 4.5.1 Zählschaltungen . . . 30
  - 4.5.2 Register . . . 32
- 4.6 Entwurfsbeispiel: Getränkteautomat . . . 34
  - 4.6.1 Aufgabenstellung . . . 34
  - 4.6.2 Spezifikation . . . 34

4.6.3 Flipflop-Ansteuerung und Ausgangsfunktion 36

Kapitel 1

Einleitung

1.1 Schichtenmodell

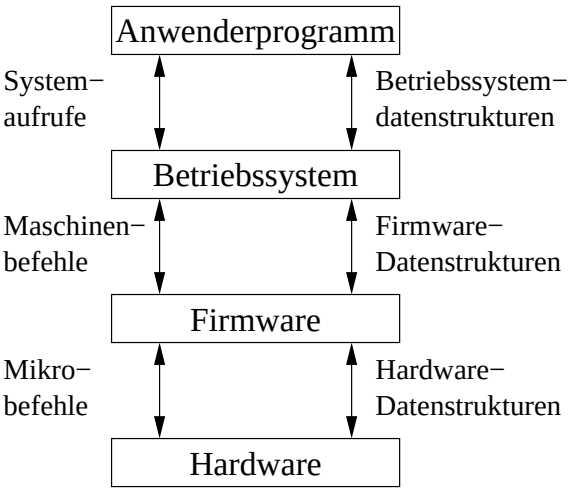


Abbildung 1.1: Schichtenmodell eines Rechners

Im Schichtenmodell werden von einer Schicht Befehle und Datenstrukturen bereitgestellt, die von der darüberliegenden Schicht benutzt werden können. Eine Schicht ist eine abstrakte Maschine mit Datentypen und Befehlssatz zur Interpretation der darüber liegenden Schicht. Die unterste Schicht entspricht der realen Maschine.

1.2 Programmhierarchie

Algorithmen müssen vom Programmierer in einer Programmiersprache geschrieben werden, die vom Computer ausgeführt werden kann. Unglücklicherweise sind höhere Programmiersprachen (wie zum Beispiel C, Pascal, Java oder Fortran), obwohl sie für den menschlichen Benutzer bequem sind, für die direkte

Ausführung durch Computer ungeeignet, weil die Konstruktionskosten eines solchen Computers sich als unzulässig hoch herausstellen. Die Computerhersteller haben deshalb Computer gebaut, die Programme in einer weitaus einfacheren Sprache, der sogenannten *Maschinensprache*, ausführen. Daraus ergibt sich, daß es Mechanismen für die Übersetzung von höheren Programmen in Maschinensprachen geben muß. Diese Übersetzung wird meist über eine Zwischenstufe realisiert, die zunächst das Programm aus der höheren Programmiersprache in eine leßbarere Variante der Maschinensprache, die sogenannte *Assemblersprache*, übersetzt und dieses wird anschließend in ein Maschinenprogramm übersetzt. Die Übersetzer für eine höhere Programmiersprache in eine Assemblersprache werden üblicherweise als *Compiler* bezeichnet, die Programme für das Übersetzen von Assemblersprache in Maschinensprache werden als *Assembler* bezeichnet.

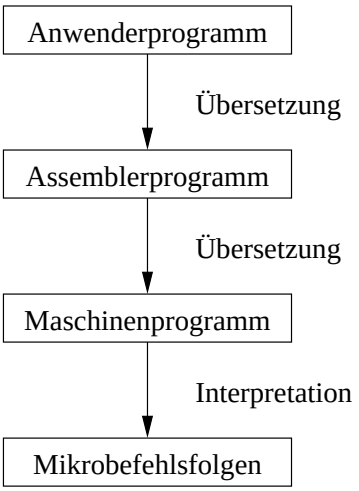


Abbildung 1.2: Programmhierarchie

Die ersten gebauten Computer konnten Programme, die in der Maschinensprache des Computers geschrieben waren, direkt ausführen. Es zeigte sich jedoch schnell, daß die verschiedenen Anweisungen einer jeden Maschinensprache untereinander viele Ähnlichkeiten aufweisen. Es ist möglich, eine kleine Anzahl grundlegender Operationen - sogenannte *Mikrobefehle* (engl. *micro instructions* oder auch *micro code*) - auszuwählen und jede Anweisung der Maschinensprache durch eine kleine Menge dieser Mikroanweisungen auszudrücken. So beinhaltet zum Beispiel jede Anweisung der Maschinensprache die Übertragung von Daten einer Stelle zu einer anderen Stelle innerhalb des Computers.

Aus ökonomischen Gründen sind moderne Computer so aufgebaut, daß sie eher Mikrobefehle ausführen als Programme in der Maschinensprache. Die Computerhersteller stellen auch ein Programm bereit, das man *Interpreter* (geschrieben in Form von Mikroanweisungen) nennt und das den Computer anweist, wie jede Anweisung in der Maschinensprache zu lesen, zu verstehen und auszuführen ist. Deshalb ist der Computer zusammen mit dem Interpreter (dieser befindet sich üblicherweise in der zentralen Recheneinheit integriert) in der Lage, Programme, die in Maschinensprache vorliegen, auszuführen. Computer, die nach dieser Vorstellung gebaut sind, nennt man *mikroprogrammierte Computer*.

Zwei weitere Ausdrücke, die häufig im Zusammenhang mit Computern verwendet werden, sind *Software* und *Hardware*. Mit Software meint man alle Programme, die zu einem Computer gehören. Mit Hardware meint man die physikalischen Einrichtungen, aus denen ein Computer zusammengesetzt ist. Programme, die unter Verwendung von Mikrobefehlen geschrieben sind, erhalten manchmal den speziellen Namen *Firmware* oder *Mikroprogramme*. So ist zum Beispiel der im vorherigen Absatz erläuterte Interpreter ein Mikroprogramm oder ein Teil der Firmware.

1.3 Begriffserklärung

Übersetzung Übertragung eines vollständigen Programmes von einer Ebene in die andere.

Interpretation Ersetzen von Befehlen durch Befehlsfolgen, und zwar zur Laufzeit.

Digitalrechner verarbeiten digitale Daten.

Zeichen sind Elemente zur Darstellung von Informationen aus einer endlichen Menge verschiedener Elemente, dem Zeichenvorrat.

Worte sind Zeichenfolgen.

Sprache wird gebildet aus Worten und Syntax.

Signale sind physikalische Größen zur Darstellung digitaler Daten.

Binäre Signale verfügen nur über zwei diskrete Werte und haben dadurch große Toleranzbereiche.

Abbildung der Gegebenheiten der realen Welt auf die Möglichkeiten digitaler Systeme.

Diskretisieren Darstellung eines kontinuierlichen Signals durch eine Folge von Abtastwerten (Abtasttheorem von SHANNON).

Binärkodierung Darstellung (Codierung) der Abtastwerte durch $0, 1$-Folgen.

Kapitel 2

Grundlagen

Systeme, die binäre Signale verarbeiten können werden beschrieben mit Methoden, die Variablen verwenden, die zwei diskrete Werte annehmen können, und den zugehörigen Operationen.

Beispiele für boolesche Algebren sind die Aussagenlogik und die Schaltalgebra.

2.1 BOOLEsche Algebra

ist ein deduktives mathemathische System, das definiert wird durch eine Menge von Elementen, Operationen und Axiomen.

Abgeschlossenheit:

$\forall a, b, c \in \mathbf{M} : c = a \otimes b$

De Morgan'sche Gesetze erlauben die Negation eines Klammerausdrucks zu überführen in die Negation zweier einzelner Variablen.

2.2 Aussagenlogik

Die Aussagenlogik ist der Teil der formalen Logik, der sich mit Aussagen und Verknüpfungen von Aussagen befaßt. *Aussagen* sind sprachliche Gebilde, die nach syntaktischen Regeln aufgebaut sind und denen die Eigenschaften *wahr* oder *falsch* zugeordnet werden können.

Aussagen werden mit kleinen Buchstaben bezeichnet (den sogenannten *Aussagenvariablen*). Die Verknüpfungen von Aussagenvariablen bilden *Formeln*, denen ein Wahrheitswert zugeordnet wird.

Definition 2.1 Die *Konjunktion* zweier Aussagen ist genau dann *wahr*, wenn beide Aussagen *wahr* sind, sonst ist sie *falsch* (siehe Tabelle 2.1). Die *Disjunktion* zweier Aussagen ist genau dann *falsch*, wenn beide Aussagen *falsch* sind, sonst ist sie *wahr* (siehe Tabelle 2.2). Die *Negation* einer *wahren* Aussage ist *falsch*, die einer *falschen* Aussage *wahr* (siehe Tabelle 2.3).

a	b	$a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

Tabelle 2.1: Konjunktion zweier Aussagen

a	b	$a + b$
0	0	0
0	1	1
1	0	1
1	1	1

Tabelle 2.2: Disjunktion zweier Aussagen

a	$\overline{a}$
0	1
1	0

Tabelle 2.3: Negation eines Aussage

2.3 Schaltalgebra

Die Schaltalgebra wurde 1936 von dem Mathematiker Claude SHANNON eingeführt, um Schaltungen mit bistabilen Elementen zu beschreiben [1]. Die Schaltalgebra ist *isomorph* zur Aussagenlogik. Das heißt Aussagen in der Aussagenlogik entsprechen geöffneten und geschlossenen Schaltern in der Schaltalgebra, und Verknüpfungen werden symbolisiert durch Reihen und Parallelschalter.

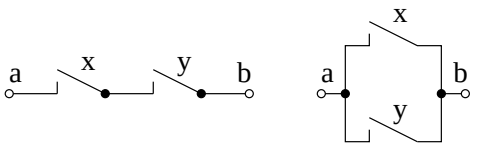


Abbildung 2.1: Serien- und Parallelschaltung von Schaltern

Hierbei wird der stets geöffnete Schalter mit „0“ und der stets geschlossene Schalter mit „1“ bezeichnet.

2.4 BOOLEsche Funktionen

Boolesche Funktionen sind geeignet, um komplexe Schaltanordnungen zu beschreiben. Sie sind Ausdrücke aus endlich vielen Zahlen mit denen Variablen

[1] ursprünglich entwickelt für Relayschaltungen

und Konstanten durch boolesche Operatoren verknüpft werden. Der Funktionswert ergibt sich durch Auswertung der Ausdrücke.

Beispiel 2.1 Seien a, b, c Elemente einer booleschen Algebra, dann sind

$$f = (a + \overline{b})\overline{c} + b + 0$$
$$f = (a + 1)\overline{(b + c)}$$

BOOLEsche Funktionen.

Auswertungsreihenfolge der Operatoren

1. „$-$“ (Negation, NICHT-Verknüpfung)
2. „$\cdot$“ (Konjunktion, UND-Verknüpfung)
3. „$+$“ (Disjunktion, ODER-Verknüpfung)

2.4.1 Schaltfunktionen

Schaltfunktionen sind eine spezielle Klasse boolescher Funktionen. Ausgehend von der binären Trägermenge $\mathbf{B} = \{0, 1\}$ werden *Abbildungen* der Art

$$f : \mathbf{B}^n \rightarrow \mathbf{B}, (n \in \mathbf{N})$$

definiert, d.h. einem n-Tupel $(x_1, x_2, \ldots, x_n)$ wird ein Element aus $\{0, 1\}$ zugeordnet. In den Wertetabellen werden die Tupel auch als *Kombinationen* bezeichnet.

Eigenschaften von Schaltfunktionen

1. Für gegebenes n umfaßt eine Schaltfunktion genau 2^n Elemente, da $B^n = \{0, 1\}^n$ gilt und somit alle Kombinationen von 0 und 1 auftreten.
2. Die Anzahl möglicher n-stelliger Schaltfunktionen entspricht der Anzahl möglicher Abbildungen einer 2^n-elementigen Menge auf eine 2-elementige Menge, d.h. $2^{(2^n)}$ (2 Variablen bilden $2^4 = 16$ Schaltfunktionen).

Index	x_1	x_0	$y = x_1 \not\equiv x_0$
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

Tabelle 2.4: Antivalenz (XOR-Verknüpfung)

Polynomdarstellung für Schaltfunktionen

Die Elemente des Definitionsbereichs werden als Variablen betrachtet. Diese werden durch Operationen der Schaltalgebra verknüpft.

Definition 2.2 Gegeben seien n Variable $x_0, x_1, \ldots, x_{n-1}$ und die Konstanten 0 und 1. Mit diesen läßt sich ein Polynom P wie folgt definieren:

- die Variablen $x_0, x_1, \ldots, x_{n-1}$ und die Konstanten sind Polynome
- sind P_1 und P_2 Polynome, dann auch $P_1 + P_2$ und $P_1 \cdot P_2$
- ist P ein Polynom, dann auch $\overline{P}$.

Jedes Polynom definiert eine Abbildung $y : \mathbf{B}^n \rightarrow \mathbf{B}$. Dadurch wird jedem Polynom eine Schaltfunktion aus der Menge der Abbildungen $\mathbf{B}^n \rightarrow \mathbf{B}$ zugeordnet. Das heißt aber, daß unterschiedliche Polynome durch diesselbe Schaltfunktion beschrieben werden können.

Beispiel 2.2 Antivalenz (XOR-Verknüpfung)

$$\begin{aligned} P_1 &= (x_0 + x_1) \cdot (\overline{x}_0 + \overline{x}_1) \\ P_2 &= (\overline{x}_0 \cdot x_1) + (x_0 \cdot \overline{x}_1) \end{aligned}$$

$$\begin{aligned} P_1 &= (x_0 + x_1) \cdot (\overline{x}_0 + \overline{x}_1 \\ &= x_0\overline{x}_0 + x_0\overline{x}_1 + \overline{x}_0 x_1 + x_1\overline{x}_1 \\ &= \overline{x}_0 x_1 + x_0\overline{x}_1 \\ &= P_2 \end{aligned}$$

Normalformen

Definition 2.3 Ein *Minterm* (*Maxterm*) von n Variablen ist ein Polynom, in dem alle Variablen genau einmal auftreten (negiert oder nicht negiert) und konjunktiv (disjunktiv) verknüpft werden. Ein Minterm (Maxterm) entspricht einer Volldisjunktion (Volldisjunktion).

Index	x_2	x_1	x_0	Minterme mn_i	Maxterme mx_i	y
0	0	0	0	$\overline{x}_2 \cdot \overline{x}_1 \cdot \overline{x}_0$	$x_2 + x_1 + x_0$	1
1	0	0	1	$\overline{x}_2 \cdot \overline{x}_1 \cdot x_0$	$x_2 + x_1 + \overline{x}_0$	0
2	0	1	0	$\overline{x}_2 \cdot x_1 \cdot \overline{x}_0$	$x_2 + \overline{x}_1 + x_0$	0
3	0	1	1	$\overline{x}_2 \cdot x_1 \cdot x_0$	$x_2 + \overline{x}_1 + \overline{x}_0$	0
4	1	0	0	$x_2 \cdot \overline{x}_1 \cdot \overline{x}_0$	$\overline{x}_2 + x_1 + x_0$	1
5	1	0	1	$x_2 \cdot \overline{x}_1 \cdot x_0$	$\overline{x}_2 + x_1 + \overline{x}_0$	1
6	1	1	0	$x_2 \cdot x_1 \cdot \overline{x}_0$	$\overline{x}_2 + \overline{x}_1 + x_0$	0
7	1	1	1	$x_2 \cdot x_1 \cdot x_0$	$\overline{x}_2 + \overline{x}_1 + \overline{x}_0$	1

Tabelle 2.5: Wertetabelle der Funktion y

Demnach lauten zum Beispiel die Gleichungen für den Minterm mit dem Index 4 (mn_4) und den Maxterm mit dem Index 3 (mx_3) wie folgt:

$$mn_4 = \quad x_2 \cdot \overline{x}_1 \cdot \overline{x}_0 \quad = \begin{cases} 1, & \text{wenn } x_2 = 1, x_1 = 0, x_0 = 0, \\ 0, & \text{sonst} \end{cases}$$

$$mx_3 = \quad x_2 + \overline{x}_1 + \overline{x}_0 \quad = \begin{cases} 0, & \text{wenn } x_2 = 0, x_1 = 1, x_0 = 1, \\ 1, & \text{sonst} \end{cases}$$

Definition 2.4 Eine disjunktive (konjunktive) *Normalform* besteht aus einer endlichen Anzahl von Mintermen (Maxtermen), die disjunktiv (konjunktiv) miteinander verknüpft werden.

Definition 2.5 Eine Funktion y der Variablen $x_n, x_{n-1}, \ldots, x_0$ läßt sich in disjunktiver (konjunktiver) Normalform darstellen, indem man die Minterme (Maxterme) für die sie den Wert 1 (0) annimmt disjunktiv (konjunktiv) verknüpft.

Definition 2.6 Zwei Schaltfunktionen sind äquivalent, wenn sie in ihrem Normalformen übereinstimmen.[2]

Erstellung der Normalformen

Beispiel 2.3 Erstellen der disjunktiven Normalform von $y = \overline{a} + bc$:

$$\begin{aligned} y &= \overline{a}(b + \overline{b})(c + \overline{c}) + (a + \overline{a})bc \\ &= (\overline{a}b + \overline{a}\overline{b})(c + \overline{c}) + abc + \overline{a}bc \\ &= \overline{a}bc + \overline{a}b\overline{c} + \overline{a}\overline{b}c + \overline{a}\overline{b}\overline{c} + abc + \overline{a}bc \\ &= \overline{a}bc + \overline{a}b\overline{c} + \overline{a}\overline{b}c + \overline{a}\overline{b}\overline{c} + abc \end{aligned}$$

Darstellung mit Diagrammen

In KV-Diagramme[3] können Schaltfunktionen eingetragen werden, indem die Felder (deren Minterme) für die die Funktion den Wert 1 annimt markiert werden.

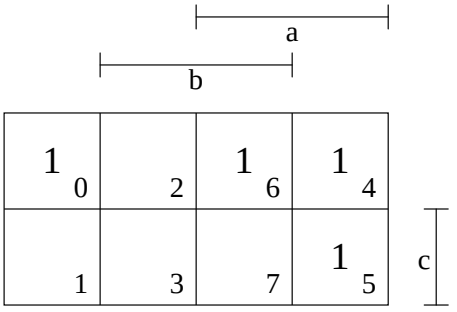


Abbildung 2.2: KV-Diagramm für drei Variable

Erweiterung auf mehr Variablen durch Spiegelung, d.h. hinzukommnder Bereich entspricht der neuen nicht negierten Variable.

[2]Ein weiteres Äquivalenzkriterium ist die Übereinstimmung der Wertetabellen.
[3]Das Kürzel „KV“ steht für KARNALLCH und VEITCH.

Vollständige Systeme von Operatoren

Mit den bisher bekannten Operatoren UND, ODER und NICHT lassen sich drei funktional vollständige Mengen darstellen.

1. {UND, ODER, NICHT}
2. {UND, NICHT}
3. {ODER, NICHT}

Zur Verdeutlichung noch der Beweis für die letzten beiden Mengen:

$$a \cdot b = \overline{\overline{a} + \overline{b}} = \overline{\overline{a}} \cdot \overline{\overline{b}} \tag{2.1}$$

$$a + b = \overline{\overline{a} \cdot \overline{b}} = \overline{\overline{a}} + \overline{\overline{b}} \tag{2.2}$$

SHEFFER-Funktion

Die SHEFFER-Funktion - meist als NAND oder UND-NICHT bezeichnet - stellt eine UND-Verknüpfung dar, bei der das Ergebnis negiert ist.

$$\boxed{y = a|b = \overline{a \cdot b}}$$

a	b	$y = a\|b$
0	0	1
0	1	1
1	0	1
1	1	0

Tabelle 2.6: Wertetabelle der SHEFFER-Funktion

Ein Vorteil der SHEFFER-Funktion ist, daß sie für sich genommen eine funktional vollständige Menge darstellt, das heißt, daß man Schaltfunktionen ohne Zuhilfenahme weiterer Verknüpfungen allein mit NANDs darstellen kann. Dazu kommt, dass sich ein NAND sehr einfach als Bauteil realisieren läßt.

PIERCE-Funktion

Die PIERCE-Funktion - meist als NOR oder ODER-NICHT bezeichnet - stellt eine ODER-Verknüpfung dar, bei der das Ergebnis negiert ist.

$$\boxed{y = a * b = \overline{a + b}}$$

Ähnlich wie die SHEFFER-Funktion, stellt auch die PIERCE-Funktion für sich genommen eine funktional vollständige Menge dar, und bietet somit denselben Vorteil. Auch ein NOR ist als Baustein sehr einfach zu realisieren.

a	b	$y = a * b$
0	0	1
0	1	0
1	0	0
1	1	0

Tabelle 2.7: Wertetabelle der PIERCE-Funktion

2.4.2 Darstellungsformen BOOLEscher Funktionen und ihre Umwandlungen

1. **Tabelle → DNF:**
   Alle Minterme der Tabelle entnehmen (Zeilen mit dem Funktionswert Eins) und ODER-verknüpfen.
   DNF → Tabelle:
   Alle Minterme der DNF entnehmen und mit dem Funktionswert Eins in der Tabelle notieren.

2. **Tabelle → KNF:**
   Alle Maxterme der Tabelle entnehmen (Zeilen mit dem Funktionswert Null) und UND-verknüpfen.
   KNF → Tabelle:
   Alle Maxterme der KNF entnehmen und mit dem Funktionswert Null in der Tabelle notieren.

3. **DNF → KNF:**
   Nummern der Minterme auslesen und KNF mit dem komplementären Maxtermen bilden.
   KNF → DNF:
   Nummern der Maxterme auslesen und KNF mit dem komplementären Mintermen bilden.

4. **DNF → KV-Diagramm:**
   Größe des KV-Diagramms durch die Anzahl der Variablen festlegen und zeichnen, Minterme eintragen (Eins-Muster).
   KV-Diagramm → DNF:
   Jede Eins im KV-Diagramm entspricht einem Minterm der Funktion.

5. **KNF → KV-Diagramm:**
   Größe des KV-Diagramms durch die Anzahl der Variablen festlegen und zeichnen, Maxterme eintragen (Null-Muster).
   KV-Diagramm → KNF:
   Jede Null im KV-Diagramm entspricht einem Maxterm der Funktion.

6. **Tabelle → KV-Diagramm:**
   Jede Zeile in der Tabelle entspricht einem Feld im KV-Diagramm und umgekehrt. Eine schnelle Zuordnung kann über die Dezimaläquivalente der Variablenbelegung erfolgen.

Kapitel 3

Schaltnetze

Schaltnetze ergeben sich als technische Realisierung von Schaltfunktionen (Schaltungen ohne Gedächtnis). In der Schaltung gibt es keine Rückkopplungen (Verbindungen von Ausgängen auf die Eingänge).

3.1 Beschreibung und Entwurfsmethodik

Schaltnetze werden beschrieben durch eine Menge von Schaltfunktionen (ein sogenanntes *Funktionsbündel*) $F = \{f_1, f_2, \ldots, f_n\}$ mit gemeinsamen Variablen $e_1, e_2, \ldots, e_n$, die die Ausgänge $a_1, a_2, \ldots, a_n$ bilden.

3.1.1 Entwurfsmethodik

1. *Spezifikation* legt fest welche Aufgabe das Schaltnetz lösen soll (verbal, teils formalisiert).
2. *Formale Beschreibung* der Spezifikation, zum Beispiel durch Wertetabellen, Diagramme oder auch ein Programm.
3. *Entwurf* besteht in der Überführung der formalen Beschreibung in eine Form, welche die gewünschte Realisierung wiederspiegelt.
4. *Optimierung 1* besteht in der Minimierung der Variablenzahlen und der Anzahl der Verknüpfungen.
5. *Umformung* für eine bestimmte Realisierung unter Berücksichtigung der zur Verfügung stehenden Bausteine bzw. Bibliotheken.
6. *Optimierung 2* unter Berücksichtigung der Realisierung zum Beispiel hinsichtlich Laufzeit und Flächenbedarf.
7. *Verifikation* vergleicht das Entwurfsergebnis mit der Spezifikation.

3.1.2 Verfahren von QUINE/McCLUSKEY für Funktionsbündel

Funktionsbündel enthalten Funktionen mit gemeinsamen Variablen, können aber auch gemeinsame Terme enthalten, sog. *Koppelterme*. Diese werden *einmal* realisiert und mehrfach benutzt.

3.1.3 Umformung von Schaltfunktionen

Das Ergebnis der Optimierung soll so umgeformt werden, daß die Funktion mit einer bestimmten Bauelementebibliothek realisiert werden kann. Interessant sind NAND- und NOR-Realisierungen, weil diese beiden Operationen *funktional vollständig* sind und zugleich die einfachsten Realisierungen darstellen. Verwendung von De Morgan und den Axiomen und Gesetzen.

Beispiel 3.1 Umformung der Schaltfunktion $y = abc + \overline{a}b\overline{c} + \overline{a}\overline{b}c$ in eine Form, die für die Realisierung mit NAND/NAND geeignet ist.

$$\begin{aligned} y &= abc + \overline{a}b\overline{c} + \overline{a}\overline{b}c \\ &= \overline{\overline{abc + \overline{a}b\overline{c} + \overline{a}\overline{b}c}} \\ &= \overline{\overline{abc} \cdot \overline{\overline{a}b\overline{c}} \cdot \overline{\overline{a}\overline{b}c}} \end{aligned}$$

Kapitel 4

Schaltwerke

Schaltnetze beschreiben einen funktionalen Zusammenhang zwischen Eingängen und Ausgängen. Zu jedem beliebigen Zeitpunkt hängen die Ausgänge nur von den Eingängen zu diesem Zeitpunkt ab. Damit sind nur Vorgänge *ohne Gedächtnis* modellierbar.

Schaltwerke - auch als sequentielle Schaltungen bezeichnet - sind in der Lage mit Hilfe von Speicherelementen zurückliegende Ereignisse zu registrieren. Ein einfaches Beispiel für ein Schaltwerk wäre ein Zähler. Ein weiteres Beispiel wäre der aus der Vorlesung bekannte Aufzug, der sich merken muss, in welchem Stockwerk er sich befindet, um entscheiden zu können, ob nach oben oder nach unten fahren muss, um in ein anderes Stockwerk zu gelangen.

4.1 Beschreibung

Zurückliegende Einwirkungen werden bei Schaltwerken durch *Zustände* erfaßt, das heißt zu einem bestimmten Zeitpunkt kann ein Schaltwerk unterschiedliche Zustände annehmen abhängig von Einwirkungen. Die Menge möglicher Zustände definiert einen *Zustandsraum* Z.

- $l = 2^k$ Elemente bei einem Schaltwerk mit k Speicherplätzen
- Zustandsvektoren $\vec{z}_i, 1 \leq i \leq l$
- Zustandsraum Z: $Z = \{\vec{z}_1, \vec{z}_2, \ldots, \vec{z}_l\}, \vec{z}_i \in \mathbf{B}^k = \{0,1\}^k$
- Ausgangsvektor zum Zeitpunkt t: $\vec{a}^t = F(\vec{z}^t, \vec{e}^t), \vec{z}^t \in Z$
- *Zustandsübergangsverhalten*: $\vec{z}^{t+\tau} = G(\vec{z}^t, \vec{e}^t), \vec{z}^t \in Z, \vec{z}^{t+\tau} \in Z$

Ein Schaltwerk besteht allgemein aus k Speicherplätzen, m Ausgängen und n Eingängen. Das heißt die Funktionsbündel F und G sind $(k+n)$-stellige Schaltfunktionen:

$$\begin{aligned} f_i: &\quad \mathbf{B}^{k+n} \to \mathbf{B}^m, \quad 1 \leq i \leq m \\ g_j: &\quad \mathbf{B}^{k+n} \to \mathbf{B}^k, \quad 1 \leq j \leq k \end{aligned}$$

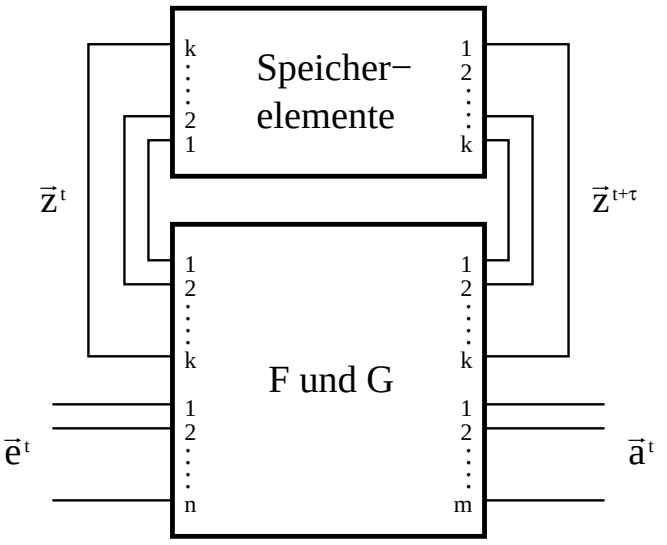


Abbildung 4.1: Schematische Darstellung eines Schaltwerks

Definition 4.1 Die Mengen A, E sind das Aus- und Eingabealphabet, das heißt $A = \mathbf{B}^m$ und $E = \mathbf{B}^n$. Z ist der Zustandsraum. Die Abbildungen $F : E \times Z \to A, F(\vec{e}, \vec{z}) = \vec{a}$ und $G : E \times Z \to Z, G(\vec{e}, \vec{z}) = \vec{z}^*$ ordnen jedem geordneten Paar $(\vec{e}, \vec{z})$ einen Ausgangsvektor $\vec{a}$ bzw. einen Zustandsvektor $\vec{z}^*$ zu. Die Mengen E, A und Z repräsentieren zusammen mit den Abbildungen F und G einen sequentiellen Automaten, abgekürzt (E, A, Z, F, G).

Der MEALY-Automat stellt den allgemeinen Fall dar, so wie in Definition 4.1 angegeben. Der MOORE-Automat unterliegt der Einschränkung, daß die Abbildung F nur von Z abhängt, das heißt $F(\vec{z}) = \vec{a}$.

Spricht man von synchronen Schaltwerken (siehe Abschnitt 4.3 Elementare Schaltwerke), so läßt sich zum MOORE-Automaten sagen, daß hier die Ausgänge mit dem Takt synchronisiert werden, während hingegen beim MEALY-Automaten sich die Ausgänge spontan mit den Eingängen ändern, also nicht durch den Takt synchronisiert werden.

Als Eselbrücke für die Begriffe möge vielleicht folgende Wortlautanalogie dienen: *MOORE* → „*nur*“ von den Zuständen abhängig.

4.1.1 Übergangsgraph

Zusätzlich unterscheiden sich die beiden Automaten dadurch, daß beim MEALY-Automaten die Ausgänge den Zuständen zugeordnet werden, während beim MOORE-Automaten die Ausgänge den Übergangen, und damit den Eingängen, zugeordnet werden.

Abbildung 4.2: MEALY-Automat

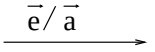


Abbildung 4.3: MOORE-Automat

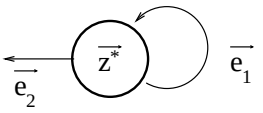


Abbildung 4.4: Stabile Zustände

4.2 Schaltwerksanalyse

Die Schaltwerksanalyse beschreibt die Untersuchung der Funktionen einer gegebenen Schaltung. Speicher in Schaltwerken sind rückgekoppelte Schaltelemente. Rückkopplungen werden aufgetrennt, so daß sich die Struktur eines Schaltwerks ergibt, wie in Abbildung 4.5 zu sehen. An der Schnittstelle entsteht eine eingangs- und eine ausgangsseitige Komponente der Zustandsvariablen $\vec{z}$ und $\vec{z}^*$. Die Schaltfunktionen aller Zustandsvariablen beschreiben das Verhalten der Schaltung. Sind $\vec{z}$ und $\vec{z}^*$ für eine Kombination identisch, dann ist der Zustand stabil, anderenfalls handelt es sich um einen Übergangszustand.

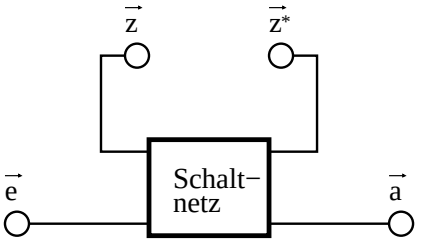


Abbildung 4.5: Schaltnetz

Allgemeines Vorgehen bei der Analyse von Schaltungen

1. Erkennen, ob es sich um ein Schaltnetz oder ein Schaltwerk handelt. Schaltwerke besitzen im Gegensatz zu Schaltnetzen Rückkopllungspfade, zu erkennen an vollständig geschlossenen Kreisen der Signalwege in Datenrichtung.

2. Handelt es sich um ein Schaltwerk, müssen alle Rückkopplungen aufgetrennt werden. An den Auftrennung werden zusätzliche Variablen (Zustandsvariablen), eingeführt - eingangsseitig q_i^* und ausgangsseitig q_i. Beim Auftrennen der Rückkopllungen sollten die folgenden Regeln beachtet werden:

   - möglichst wenig Auftrennung verwenden
   - so viele Rückkopllungen auf einmal wie möglich auftrennen
   - so nah am Ausgang wie möglich auftrennen
   - möglichst direkt hinter einem Gatter auftrennen

   Nach dem Auftrennen ergibt sich die Struktur eines Schaltnetzes.

3. Zur Analyse eines Schaltnetzes (vorgegeben oder durch Auftrennungen entstanden) werden nun zunächst die Funktionsgleichungen für alle kombinatorischen Ausgänge und die durch die Auftrennung entstandenen Zustandvariablen q_i aus der Schaltung bestimmt.

4. Danach wird die Funktionstabelle aufgestellt.

5. War die Ausgangsschaltung ein Schaltnetz, kann nun noch versucht werden, die Funktion des Schaltnetzes zu ermitteln.
   → **Die Schaltnetzanalyse ist damit abgeschlossen.**

6. War die Ausgangsschaltung ein Schaltwerk, werden in der Funktionstabelle nun die stabilen Zustände gekennzeichnet. Dies sind solche Zustände, bei denen bei einer bestimmten Eingangskombination auf beiden Seiten der Schnittstelle (z_i^* und z_i) identische Werte auftreten.
   Treten unterschiedliche Variablenwerte auf beiden Seiten einer Schnittstelle auf, so ist der betreffende Übergangszustand instabil. Er kann nur kurzzeitig auftreten und muss schließlich in einen stabilen Zustand übergehen.

7. Ein Zustand kann für unterschiedliche Eingangskombinationen stabil oder instabil sein. Zur einfacheren Beschreibung des Schaltwerkes ist es i.a. ausreichend, nur diejenigen Zustände zu berücksichtigen, für die mindestens eine Eingangskombination stabil ist. Hiermit kann man die Übergangstabelle verkürzen und das Schaltverhalten anschaulich in einem Übergangsdiagramm darstellen. Instabile Zustände bewirken hierbei immer Zustandsübergange.

8. Anhand des Zustandsdiagramms kann versucht werden, die Funktion der vorgegebenen Schaltung zu ermitteln.
   → **Die Schaltwerksanalyse ist damit abgeschlossen.**

4.3 Elementare Schaltwerke

Man unterscheidet im allgemeinen zwischen *asynchronen* und *synchronen* Schaltwerken.

Asynchrone Schaltungen sind durch spontane Schaltvorgänge gekennzeichnet. Mit einer Änderung der Eingangssignale ändern sich Zustand und Ausgänge. Daraus folgt eine maximale Geschwindigkeit, undefinierte Laufzeiten und hohe Störanfälligkeit.

Synchrone Schaltungen hingegen werden durch ein Signal (den *Takt*) als Auslösesignal gesteuert. Die Eingänge werden hierbei nur zu bestimmten definitierten Zeiten ausgewertet, das heißt Schaltungsteile werden synchronisiert und zwischenzeitliche Störungen, wie sie bei asynchronen Schaltungen auftreten können, werden unterdrückt.

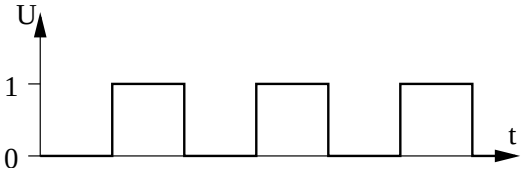


Abbildung 4.6: Taktverlauf für synchrone Schaltungen

4.3.1 Schaltwerke zur Speicherung einer binären Variablen

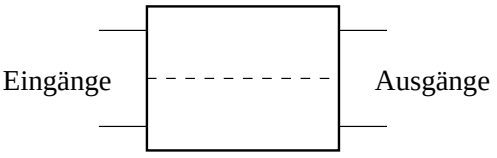


Abbildung 4.7: Schaltzeichen eines Flipflop

Anforderungen an ein Speicherelement:

- *Speicherung*: Es sind zwei stabile Zustände für die Speicherung der binären Variable erforderlich.
- *Schreiben*: Die Zustände müssen von außen einstellbar sein.
- *Lesen*: Die Speicherinhalte sollen negiert und nicht negiert an den Ausgängen ablesbar sein.

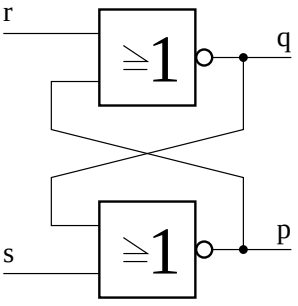


Abbildung 4.8: Schaltwerk aus zwei gekoppelten NOR-Gliedern

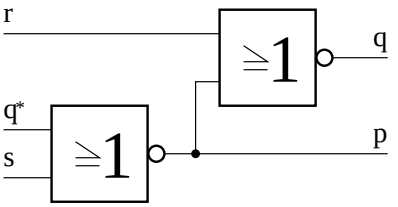


Abbildung 4.9: Schaltwerk mit aufgetrennter Rückkopplung

Nr.	r	s	q^*	p	q
1	0	0	0	1	0
2	0	0	1	0	1
3	0	1	0	0	1
4	0	1	1	0	1
5	1	0	0	1	0
6	1	0	1	0	0
7	1	1	0	0	0
8	1	1	1	0	0

Tabelle 4.1: Funktionstabelle der Schaltung nach Abbildung 4.8

Nr.	r	s	q^*	q	p	Bemerkung
1	0	0	0	0	1	stabiler Zustand
3	0	1	0	1	0	instabiler Übergangszustand
5	1	0	0	0	1	stabiler Zustand
7	1	1	0	-	-	nicht zulässig
2	0	0	1	1	0	stabiler Zustand
4	0	1	1	1	0	stabiler Zustand
6	1	0	1	0	1	instabiler Übergangszustand
8	1	1	1	-	-	nicht zulässig

Tabelle 4.2: Funktionstabelle 4.1 in anderer Darstellung

Daraus ergeben sich folgende Schaltfunktionen für die Ausgänge p und q des Flipflop:

$$\begin{aligned} p &= \overline{s + q^*} \\ q &= \overline{r + p} \\ &= \overline{r} \cdot \overline{p} \end{aligned}$$

r	s	q	Bemerkung
0	0	q^*	Speichern
0	1	1	Setzen (set)
1	0	0	Zurücksetzen (reset)
1	1	–	nicht zulässig

Tabelle 4.3: Verkürzte Übergangstabelle

Dieses asynchrone Schaltwerk zur Speicherung einer binären Variablen wie oben beschrieben wird als RS-Flipflop bezeichnet (siehe Abbildung 4.10). Da der durch die Eingänge in das Speicherelement gespeicherte Wert - 0 bei $r = 1, s = 0$ bzw. 1 bei $r = 0, s = 1$ - sofort am Ausgang q bzw. $\overline{q}$ sichtbar ist, spricht man hier auch von einem *transparenten Flipflop*.

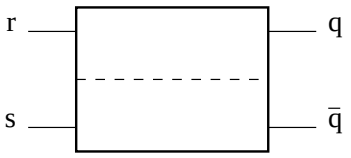


Abbildung 4.10: Schaltzeichen des asynchronen RS-Flipflop

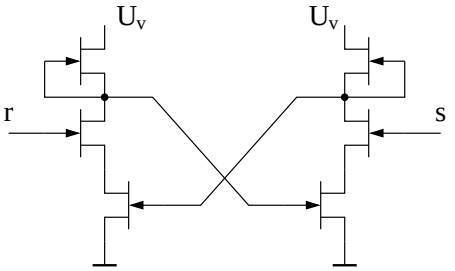


Abbildung 4.11: NOR-Schaltung rückgekoppelt (MNMOS Technologie)

Die Eingabe $r = 1$ und $s = 0$ wird als Zurücksetzen (engl. *reset*), die Eingabe $r = 0$ und $s = 1$ wird als Setzen (engl. *set*) bezeichnet. Setzen wir dagegen $r = s = 1$, nehmen sowohl q als auch $\overline{q}$ den Wert 0 an, und es ist nicht vorhersagbar, welcher Zustand angenommen wird, wenn danach $r = s = 0$ gesetzt wird. Bei realen Gattern hängt das von den stets vorhandenen Asymmetrien ab. Die Eingabe $r = s = 1$ (also gleichzeitiges Zurücksetzen und Setzen) ist also zu vermeiden, wenn diese Schaltung als Speicherelement verwenden werden soll.

4.3.2 Taktflankengesteuertes RS-Flipflop

Synchrone Schaltwerke werden wie oben bereits erwähnt durch ein Taktsignal gesteuert bzw. synchronisiert. Man unterscheidet hierbei zwischen *Taktzustandssteuerung* und *Taktflankensteuerung*. Bei der Taktzustandssteuerung wirkt der gesamte positive Takt als Auswertezeitraum für die Eingangswerte. Abbildung 4.12 zeigt das Schaltbild für ein taktzustandsgesteuertes RS-Flipflop.

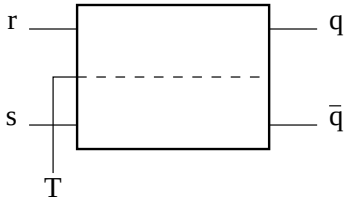


Abbildung 4.12: Schaltbild für ein taktzustandsgesteuertes RS-Flipflop

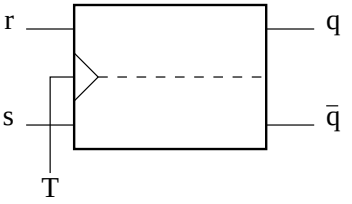


Abbildung 4.13: Schaltbild für ein taktflankengesteuertes RS-Flipflop

Der Pfeil auf dem Schaltbild des taktflankengesteuerten RS-Flipflop in Abbildung 4.13 kennzeichnet einen sogenannten *dynamischen Eingang*, das heißt, daß nicht, wie bei der Zustandssteuerung, der gesamte Takt T wirksam ist, sondern nur die steigende Flanke des Taktes T, wie in Abbildung 4.14 gezeigt.

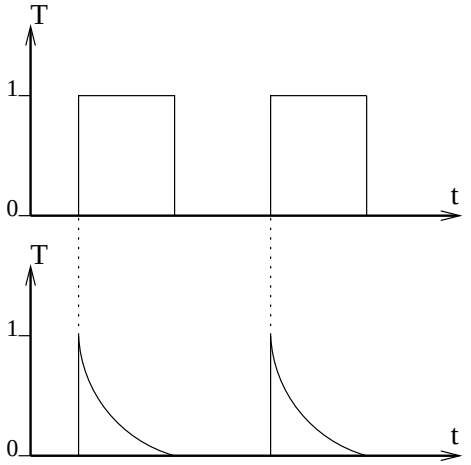


Abbildung 4.14: Taktzustands- und Taktflankensteuerung

Der Auswertezeitraum der Eingangssignale r und s wird bei der Flankensteuerung gegenüber einer Zustandssteuerung stark verkürzt. Ein Zustandswechsel kann nur bei steigender Taktflanke erfolgen.

Die Signale r und s werden zu *Verarbeitungseingängen* und der Takt T wirkt als *Auslöseeingang.*

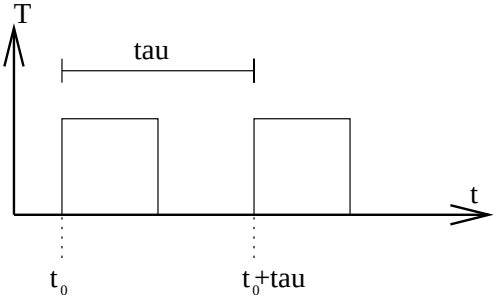


Abbildung 4.15: Taktabstände

4.3.3 Getaktetes Vorspeicher-Flipflop (Master-Slave-Flipflop)

Ein Vorspeicher-Flipflop - häufig auch als *nicht transparentes* Flipflop bezeichnet - unterscheidet sich von den bisher betrachteten Flipflop Typen dadurch, daß sich der Zeitpunkt der Übernahme der Eingangswerte r, s von dem des Ausgangsübergangs q_1 bzw. $\overline{q}_1$§ unterscheided. Die wesentlichen Eigenschaften eines nicht transparenten Flipflops werden durch zwei Zeitintervalle erfaßt. Das eine Intervall gibt die Zeit an, während der das Eingangssignal „gemessen“ wird, d.h. während der das Eingangssignal für die durch das Taktsignal ausgelöste Zustandsänderung im Flipflop relevant ist. Das andere Intervall gibt die Zeit an, während der sich eine Zustandsänderung im Flipflop als Binärübergang des Ausgangssignals bemerkbar machen kann. Beide Intervalle dürfen sich nicht überlappen, will man solche Flipflops als Speicherlemente für ein Schaltwerk verwenden. Dieses Verhalten läßt sich durch das Master-Slave-Prinzip erreichen.

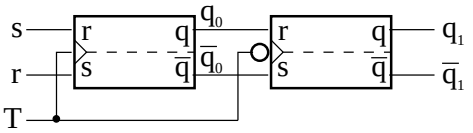


Abbildung 4.16: Schaltbild eines Vorspeicher-Flipflop

In Abbildung 4.16 wird ein Master-Slave-Flipflop dargestellt, welches mit Hilfe von zwei taktflankengesteuerten RS-Flipflops realisiert wird.

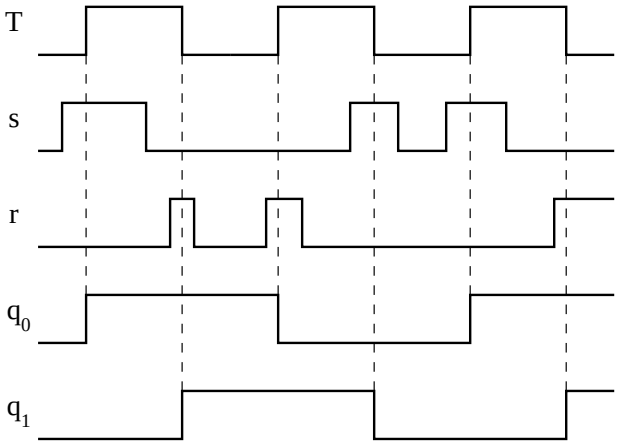


Abbildung 4.17: Impulsplan des Vorspeicher-Flipflop

Wie in Abbildung 4.17 zu erkennen, wird also bei steigender Taktflanke der Eingangswert - zur Erinnerung, $s = 1$ bedeutet Setzen und $r = 1$ bedeutet Zurücksetzen - in das erste Flipflop, welches durch die Zustandsvariable q_0 beschrieben wird, gespeichert. Und erst bei fallender Taktflanke wird der in q_0 gespeicherte Wert in das zweite Flipflop, welches durch die Zustandsvariable q_1 beschrieben wird, übernommen und liegt damit am Ausgang des Vorspeicher-Flipflops bereit. Dieses Prinzip der Taktung wird auch als *Zweiflankensteuerung* bezeichnet.

$$q_0 = \overline{(r \cdot p) + \overline{(q_0^* + (s \cdot p))}}$$
$$q_1 = \overline{(\overline{q}_0^* \cdot n) + \overline{(q_1^* + (q_0^* \cdot n))}}$$
$$(r = s = 1 \text{ unzulässig})$$

Wir treffen folgende Zuordnung:

q_0^*	q_1^*	Zustand
0	0	Zustand 1
1	0	Zustand 2
1	1	Zustand 3
0	1	Zustand 4

Tabelle 4.4: Zuordnung der Zustände

Bedingungen aus den Übergangsfunktionen, unter welchen Bedingungen Zustände beibehalten bzw. unter welchen Bedingungen Zustände verlassen werden.

Zustand	**Bedingung**	**Folgezustände**		
Zustand 1		Zustand 1	Zustand 2	
$q_0^* = 0$	$q_0 = \overline{r}sp$	$0 = r + \overline{s} + \overline{p}$	$1 = \overline{r}sp$	
$q_1^* = 0$	$q_1 = 0$	$0 = 0$	$0 = 0$	
Zustand 2		Zustand 1	Zustand 2	Zustand 3
$q_0^* = 1$	$q_0 = \overline{r} + \overline{p}$	$0 = rp$	$1 = \overline{r} + \overline{p}$	$1 = n$
$q_1^* = 0$	$q_1 = n$	$0 = \overline{n}$	$0 = \overline{n}$	$1 = n$
Zustand 3		Zustand 3	Zustand 4	
$q_0^* = 1$	$q_0 = \overline{r} + \overline{p}$	$1 = \overline{r} + \overline{p}$	$0 = r \cdot p$	
$q_1^* = 1$	$q_1 = 1$	$1 = 1$	$1 = 1$	
Zustand 4		Zustand 4	Zustand 3	Zustand 1
$q_0^* = 0$	$q_0 = \overline{r}sp$	$0 = r + \overline{s} + \overline{p}$	$1 = \overline{r}sp$	$0 = r + \overline{s} + \overline{p}$
$q_1^* = 1$	$q_1 = \overline{n}$	$1 = \overline{n}$	$1 = \overline{n}$	$0 = n$

Tabelle 4.5: Erweiterte Zustandsübergangstabelle für ein Master-Slave-Flipflop

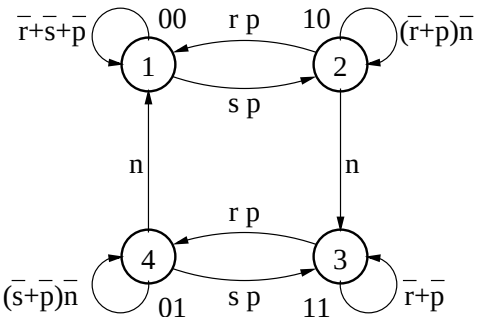


Abbildung 4.18: Übergangsdiagramm des Vorspeicher-Flipflop

4.3.4 JK-Flipflop

Dieses Universal-Flipflop funktioniert im Prinzip wie ein RS-Flipflop. Es bietet die Möglichkeit den Zustandswert des Flipflop zu speichern ($j = k = 0$), den Wert auf 1 zu setzen ($j = 1, k = 0$) und den Wert des Flipflop auf 0 zurückzusetzen ($j = 0, k = 1$). Zusätzlich bewirkt der für das RS-Flipflop verbotene Fall $j = k = 1$ ein Kippen (engl. *trigger*) des Zustandwerts.

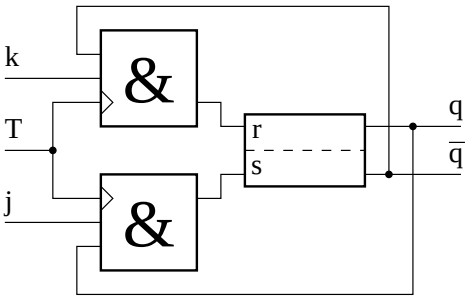


Abbildung 4.19: Schaltbild eines JK-Flipflop

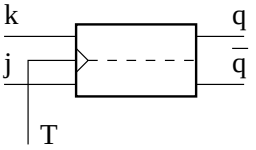


Abbildung 4.20: Schaltzeichen eines JK-Flipflop

Für die Eingänge am RS-Flipflop gelten also die folgenden Gleichungen:

$$r = k \cdot q \cdot p = k^t \cdot q^t$$
$$s = j \cdot \overline{q} \cdot p = j^t \cdot \overline{q}^t$$

Die Eingangsgrößen $j = 1, k = 1$ sind jetzt zulässig, wie die folgende kurze Zwischenrechnung zeigt:

$$\begin{aligned} r \cdot s &= (k^t \cdot q^t) \cdot (j^t \cdot \overline{q}^t) \\ &= (j^t \cdot k^t) \cdot \underbrace{(q^t \cdot \overline{q}^t)}_{=0} = 0 \end{aligned}$$

Damit sieht die Übergangsgleichung für das JK-Flipflop wie folgt aus:

$$\begin{aligned} q^{t+1} &= (j^t \cdot \overline{q}^t) + [\overline{(k^t \cdot q^t)} \cdot q^t] \\ &= (j^t \cdot \overline{q}^t) + (\overline{k}^t \cdot q^t) + (q^t \cdot \overline{q}^t) \\ &= (j^t \cdot \overline{q}^t) + (\overline{k}^t \cdot q^t) \end{aligned}$$

j^t	k^t	q^{t+1}	Bemerkung
0	0	q^t	speichern (save)
1	0	1	setzen (set)
0	1	0	zurücksetzen (reset)
1	1	$\overline{q}^t$	wechseln (toggle)

Tabelle 4.6: Übergangstabelle des JK-Flipflop

4.3.5 D-Flipflop

Das D-Flipflop - das D steht für *Delay* (Verzögerung) - hat nur einen Vorbereitungseingang, dessen Wert bei Eintreffen eines Taktsignals gespeichert wird (Verzögerungselement). Die Übergangsfunktion für das D-Flipflop lautet $q^{t+1} = D^t$.

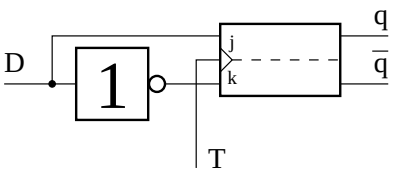


Abbildung 4.21: Schaltbild eines D-Flipflop

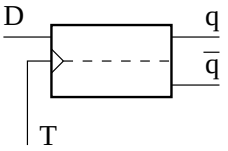


Abbildung 4.22: Schaltzeichen eines D-Flipflop

D^t	q^{t+1}
0	0
1	1

Tabelle 4.7: Übergangstabelle des D-Flipflop

4.4 Schaltwerksynthese

Schaltwerksynthese ist der *Entwurf eines Schaltwerks mit vorgegebener Funktionalität*. Dabei sind die einzelnen Schritte wie folgt:

1. Spezifikation (verbale Beschreibung)
2. Formale Beschreibung (Eingangs-/Ausgangsfunktionen, Übergangsfunktionen)
   - Festlegung der Anzahl der Zustände
   - Zustandscodierung
3. Speicherelemente realisieren bzw. ansteuern

Bei asynchronen Schaltwerken werden die Speicherelemente direkt realisiert, indem für eine bestimmte Zustandsvariable z_i eine Verbindung zwischen z_i und z_i^* in Form einer Rückkopplung hergestellt wird.

Bei synchronen Schaltungen wird für jede Zustandsvariable ein Flipflop so angesteuert, daß die Übergangsfunktion der entsprechenden Variablen realisiert wird.

4.4.1 Ansteuergleichungen für Speicherglieder

Das Aufstellen der Ansteuergleichungen für Speicherglieder läuft darauf hinaus eine Übereinstimmung zwischen z_i^{t+1} und q_i^{t+1} herzustellen. Daraus ergibt sich, daß die Ansteuerbedingungen für ein Flipflop aus der Zustandsübergangsfunktion der entsprechenden Variablen berechenbar sind, es gilt also $z_i^{t+1} = q_i^{t+1}$.

Methode des Koeffizientenvergleichs

$$f_i(z^t, e^t) = g_i(q_i^t, \vec{v}^t)$$

diese Gleichung läßt sich nur dann erfüllen, wenn die Koeffizienten von q^t auf beiden Seiten übereinstimmen.

Koeffizientenvergleich

$$\begin{aligned} s_0^t &= \overline{z}_0^t & s_1^t &= z_0^t \cdot \overline{z}_1^t \\ \overline{r}_0^t &= 0 \text{ bzw. } r_0^t = 1 & r_1^t &= z_0^t \end{aligned}$$

Einbringung der Nebenbedingung

$$\begin{aligned} z_0^{t+1} &= \overline{z}_0^t + \overbrace{\overline{z}_0^t z_0^t}^{=0} & z_1^{t+1} &= z_0^t \overline{z}_1^t + \overline{(z_0^t z_1^t)} z_1^t \\ \Rightarrow s_0^t &= \overline{z}_0^t \wedge r_0^t = z_0^t & \Rightarrow s_1^t &= z_0^t \cdot \overline{z}_1^t \wedge r_1^t = z_0^t \cdot z_1^t \end{aligned}$$

Methode der Fallunterscheidung

Entwicklungssatz besagt, daß man jede Schaltfunktion nach jeder in ihr auftretenden unabhängigen Variablen entwickeln kann, indem man diese Variable zu 0 bzw. zu 1 setzt und mit der negierten bzw. nicht negierten Variablen konjunktiv verknüpft.

$$q_i^{t+1} = q_i(\vec{q}^t, \vec{e}^t) = j_i^t \cdot \overline{q}_i^t + \overline{k}_i^t \cdot q_i^t \tag{4.1}$$

$$\begin{aligned} f(x_0, x_1, \ldots, x_i, \ldots, x_{n-1}) = f(x_0, x_1, \ldots, x_i = 0, \ldots, x_{n-1}) \cdot \overline{x}_i \\ + f(x_0, x_1, \ldots, x_i = 1, \ldots, x_{n-1}) \cdot x_i \end{aligned} \tag{4.2}$$

Als Abkürzung soll die folgende Notation dienen:

$$f(x_0, x_1, \ldots, x_{n-1}) = f\big|_{x_i=0} \cdot \overline{x}_i + f\big|_{x_i=1} \cdot x_i \tag{4.3}$$

Formuliert man den Entwicklungssatz für eine Zustandsvariable z_i^t, so ergibt sich:

$$z_i^{t+1} = f_i(\vec{z}^t, \vec{e}^t)\big|_{z_i^t=0} \cdot \overline{z}_i^t + f_i(\vec{z}^t, \vec{e}^t)\big|_{z_i^t=1} \cdot z_i^t \tag{4.4}$$

Zusammen mit der Flipflop-Übergangsfunktion

$$q_i^{t+1} = j_i^t \overline{q}_i^t + \overline{k}_i^t q_i^t \tag{4.5}$$

ergeben sich dann die Bestimmungsgleichungen für die Flipflop-Eingänge wie folgt:

$$j_i^t = f_i(\vec{z}^t, \vec{e}^t)\big|_{z_i^t=0} = z_i^{t+1}\big|_{z_i^t=0} \tag{4.6}$$

$$k_i^t = \overline{f_i(\vec{z}^t, \vec{e}^t)}\big|_{z_i^t=1} = \overline{z_i^{t+1}}\big|_{z_i^t=1} \tag{4.7}$$

$$f_i(\vec{z}^t, \vec{e}^t)\big|_{z_i^t=1} \cdot z_i^t + f_i(\vec{z}^t, \vec{e}^t)\big|_{z_i^t=0} \cdot \overline{z}_i^t = s_i^t + \overline{r}_i^t \cdot q_i^t \tag{4.8}$$

Daraus ergeben sich die beiden Ansteuergleichungen.

$$\overline{r}_i^t = f_i(\vec{z}^t, \vec{e}^t)\big|_{z_i^t=1} = z_i^{t+1}\big|_{z_i^t=1} \tag{4.9}$$

$$s_i^t = f_i(\vec{z}^t, \vec{e}^t)\big|_{z_i^t=0} \cdot \overline{z}_i^t = z_i^{t+1}\big|_{z_i^t=0} \cdot \overline{z}_i^t \tag{4.10}$$

Entwurfsbeispiel

Es soll ein Schaltwerk entworfen werden, daß Dualzahlen seriell addiert. Dazu brauchen wir zwei n-Bit Register, die die Operanden A und B enthalten, ein Speicherelement, welches das Übertragsbit, das bei jeder Addition entsteht speichert, und ein Schaltnetz, das die eigentliche Addierarbeit verrichtet. Das Ergebnis der Addition soll nach Abschluß der Addition im Register B bereit liegen. Abbildung 4.24 zeigt den Aufbau der Operandenregister und Abbildung 4.23 zeigt den aus unseren Überlegungen entstandenen Schaltplan für den Addierer.

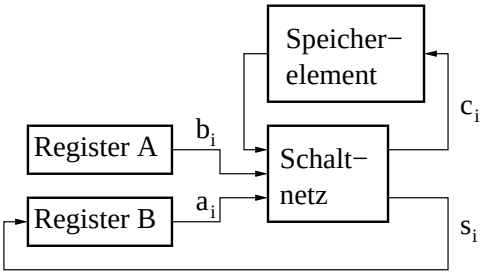


Abbildung 4.23: Schaltplan des seriellen Addierwerks

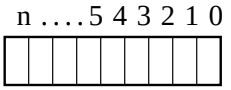


Abbildung 4.24: Aufbau der Operandenregister

Dazu muß der Übertrag c_i (engl. *carry*) gespeichert werden und bei der Verarbeitung (in diesem Fall bei der Addition) des Ziffernpaares mit dem nächsthöheren Index mitberücksichtigt werden.

Benötigt wird ein Schaltnetz mit 3 Eingängen a_i, b_i, c_i und zwei Ausgängen s_i und c_i. i ist bei dieser Betrachtung der räumliche Index, im nächsten Schritt gehen wir nun über zum zeitlichen Index, den man bei Schaltwerken immer hat.

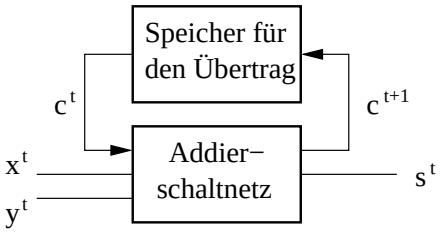


Abbildung 4.25: Blockschaltbild des Serienaddierers

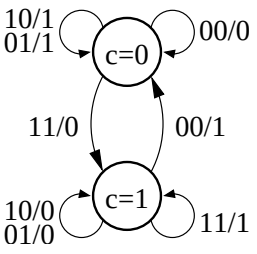


Abbildung 4.26: Übergangsdiagramm des Übertragsspeichers (MEALY-Automat)

Nr.	x^t	y^t	c^t	c^{t+1}	s^t
1	0	0	0	0	0
2	0	1	0	0	1
3	1	0	0	0	1
4	1	1	0	1	0
5	0	0	1	0	1
6	0	1	1	1	0
7	1	0	1	1	0
8	1	1	1	1	1

Tabelle 4.8: Summen- und Übertragsfunktion bei der Addition zweier Dualzahlen

Aus Tabelle 4.8 läßt sich erkennen, daß sich für s^t in Abhängigkeit von x^t und y^t für $c^t = 0$ eine Antivalenz und für $c^t = 1$ eine Valenz ergibt. Das läßt schon Rückschlüsse auf die benötigte Summenfunktion des Addierwerks zu. Diese Symetrie ergibt sich einfach aus der Tatsache, daß das Ergebnis einer Addition von drei Dualzahlen genau dann 1 ergibt, wenn entweder einer oder drei der Operanden den Wert 1 haben, in jedem anderen Fall ist das Ergebnis 0. Für den Übertrag der Addition gilt, daß er genau dann 1 ist, wenn zwei oder mehr Operanden den Wert 1 haben, ansonsten 0.

Realisierung des Übertragsspeichers mit einem JK-Flipflop. Ansteuerbedingungen

$$q^{t+1} = j^t\overline{q}^t + \overline{k}^t q^t \quad \text{(Allgemeine Form)}$$

$$\text{bzw.} \quad c^{t+1} = j^t\overline{c}^t + \overline{k}^t c^t \quad \text{(unser Fall mit der Übertragsvariable}$$

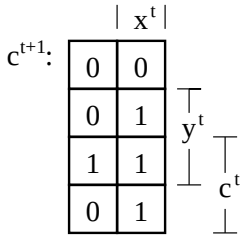
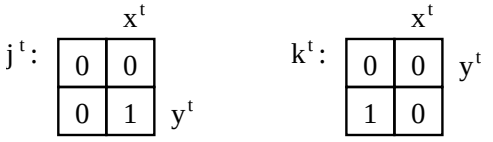


Abbildung 4.27: KV Diagramm der Übergangsfunktion



$$
\begin{aligned}
c^{t+1} = j^t\big|_{c^t=0}: \quad & \Rightarrow j^t = c^{t+1} \\
& \Rightarrow j^t = x^t \cdot y^t \\
\\
c^{t+1} = \overline{k}^t\big|_{c^t=1}: \quad & \Rightarrow k^t = \overline{c}^{t+1} \\
& \Rightarrow k^t = \overline{x^t + y^t} \\
& \Rightarrow k^t = \overline{x}^t \cdot \overline{y}^t
\end{aligned}
$$

$s^t = \{f(c,x,y)\}^t$ minimieren:

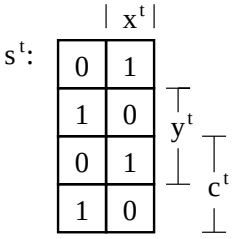


Abbildung 4.28: KV Diagramm der Summenfunktion

$$
\begin{aligned}
s^t &= [(\overline{x} \cdot \overline{y} \cdot c) + (\overline{x} \cdot y \cdot \overline{c}) + (x \cdot \overline{y} \cdot \overline{c}) + (x \cdot y \cdot c)]^t \\
&= (x \equiv y)^t \cdot c^t + (x \not\equiv y)^t \cdot \overline{c}^t
\end{aligned}
$$

Mit der Summenfunktion und den Flipflop-Ansteuergleichungen ergibt sich das vollständige Schaltbild für ein Serienaddierwerk mit JK-Flipflop.

4.5 Spezielle Schaltwerke

In diesem Abschnitt sollen zwei Vertreter der Klasse *spezieller Schaltwerke* betrachtet werden, zum einen die Zähler und desweiteren die Register.

4.5.1 Zählschaltungen

Zählschaltungen dienen zur Erfassung von Zählereignissen. Der Zähler zurückgesetzt und auftretende Zählereignisse werden aufsummiert. Jeder Zahl von Zählereignissen entspricht ein Zustand der Zähleinheit. Daraus ergbit sich folgende Übergangsgleichung:

$$\vec{z}^{t+\tau} = G(\vec{z}^t, e^t) \tag{4.11}$$

Wichtig hierbei zu beachten, ist daß wir nur einen Eingang haben, und keinen Eingangsvektor haben. Das heißt wir zählen bei jedem Signal auf e^t den Zähler um eins hoch. e^t wird deshalb auch als Zählgröße bezeichnet.

Die Ausgangsgleichung sieht dementsprechend wie folgt aus:

$$\vec{a}^t = F(\vec{z}^t) \tag{4.12}$$

Bei dem Zählwerk handelt es sich demnach um einen MOORE-Automat, da alle Ausgänge zum Zeitpunkt t nur von den Zuständen zum Zeitpunkt t

abhängen, nicht aber von den Eingängen.

Man unterscheidet die folgenden Typen von Zählern:

- *Dualzähler*: Die Zustandscodierung erfolgt durch Dualzahlen
- *Dezimalzähler*: Die Zustandscodierung erfolgt durch BCD (Binary Coded Dezimal) Zahlen
- $\binom{n}{1}$ *Zähler*: Jedem Zählerstand entspricht ein Flipflop

Zusätzlich unterscheided man Zähler anhand ihrer Arbeitsweise:

- *serielle Zähler*: Zählereignis steuert das erste Flipflop an, dessen Ausgang das zweite Flipflop, und so fort
- *parallele Zähler*: Zählereignis gleichzeitig auf alle Flipflops

Welche Art von Zähler man verwendet hängt davon ab, inwiefern die *Zählgeschwindigkeit* wichtiger ist als der *Schaltungsaufwand*.

Ein paralleler Dezimalzähler

Realisiert mit 4 JK-Flipflops

Zur Erinnerung, die Übergangsfunktion für ein JK-Flipflop lautet

$$q_i^{t+1} = j_i^t \overline{q}_i^t + \overline{k}_i^t q_i^t$$

und die Ansteuerbedingungen für die Eingänge j und k des Flipflop sind:

$$j_i^t = q_i^{t+1}\big|_{q_i^t=0}$$
$$k_i^t = \overline{q_i^{t+1}}\big|_{q_i^t=1}$$

1. Speicherelement:

j-Eingang:

$z_1^{t+1}\big|_{z_1^t=0}$:

z_4^t \ z_2^t	1	1
1	1	1
X	X	z_3^t
1	X	

$\boxed{j_1^t = 1}$

k-Eingang:

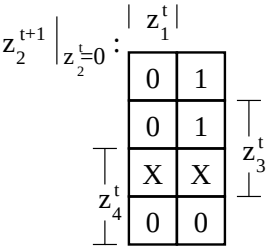
$\overline{z}_1^{t+1}\big|_{z_1^t=1}$:

z_4^t \ z_2^t	1	1
1	1	1
X	X	z_3^t
1	X	

$\boxed{k_1^t = 1}$

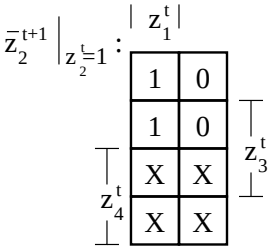
2. Speicherlement:

j-Eingang:



$$\boxed{j_2^t = z_1^t \cdot \overline{z}_4^t}$$

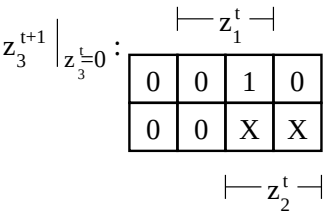
k-Eingang:



$$\boxed{k_2^t = z_1^t}$$

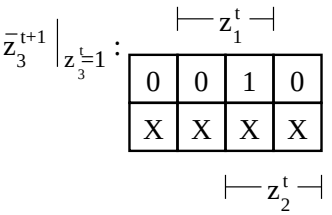
3. Speicherlement:

j-Eingang:



$$\boxed{j_3^t = z_1^t \cdot z_2^t}$$

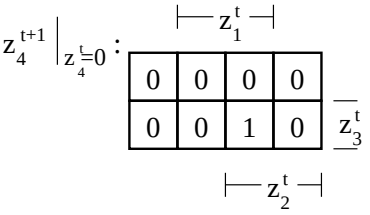
k-Eingang:



$$\boxed{k_3^t = z_1^t \cdot z_2^t}$$

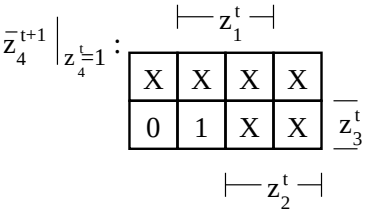
4. Speicherlement:

j-Eingang:



$$\boxed{j_4^t = z_1^t \cdot z_2^t \cdot z_3^t}$$

k-Eingang:



$$\boxed{k_4^t = z_1^t}$$

4.5.2 Register

Register bestehen aus mehreren Speichergliedern (Flipflops), die über einen gemeinsamen *Arbeitstakt* betrieben werden.

Eine spezielle Form von Registern stellen die *Schieberegister* dar. Hierbei werden Register so verbunden, daß sich der Zustand eines Speicherelementes auf das nachfolgende übertragen läßt, usf. Verbindet man zusätzlich den Ausgang des Schieberegisters (der Ausgang des letzten Speicherlements in der Kette) mit dem Eingang des Schieberegisters (der Eingang des ersten Speicherelements in

der Kette), so erhält man eine ringförmige Struktur und kann den Inhalt eines Speichergliedes umlaufen lassen.

Aufbau

Problematik: Wenn alle Speicherglieder (Flipflops) denselben Takt benutzen (siehe Abschnitt 4.3 Elementare Schaltwerke), kann es bei der Hintereinanderschaltung von Flipflops vorkommen, daß die Auslösezeit (es ist technisch nicht möglich diese Auslösezeit auf einen Zeitpunkt zu beschränken, man erhält immer einen Zeitraum) größer ist als die Umschaltzeit der Speicherelemente, und sich daraus unerwünschte Zustandswechsel der nachfolgenden Speicherglieder ergeben.

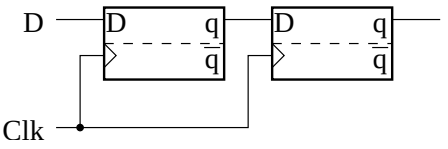


Abbildung 4.29: 2-Bit Schieberegister

Abhilfe schaffen hier Zwischenspeicher zur Entkopplung für die Dauer der aktiven Taktzeit, die bereits bekannten Vorspeicher-Flipflops. Abbildung 4.29 zeigt ein 2-Bit Schieberegister realisiert mit zwei Master-Slave-D-Flipflops.

Die Eingabe und Ausgabe kann entweder *seriell*, das heißt jedes Bit wird einzeln in das Schieberegister gespeichert bzw. aus dem Schieberegister ausgelesen werden, oder *parallel*, das heißt es können alle Bits gleichzeitig gesetzt bzw. gelesen werden, erfolgen. Für parallele Ein/Ausgabe ist es notwending, daß die Eingänge und/oder Ausgänge der einzelnen Flipflops zugänglich sind.

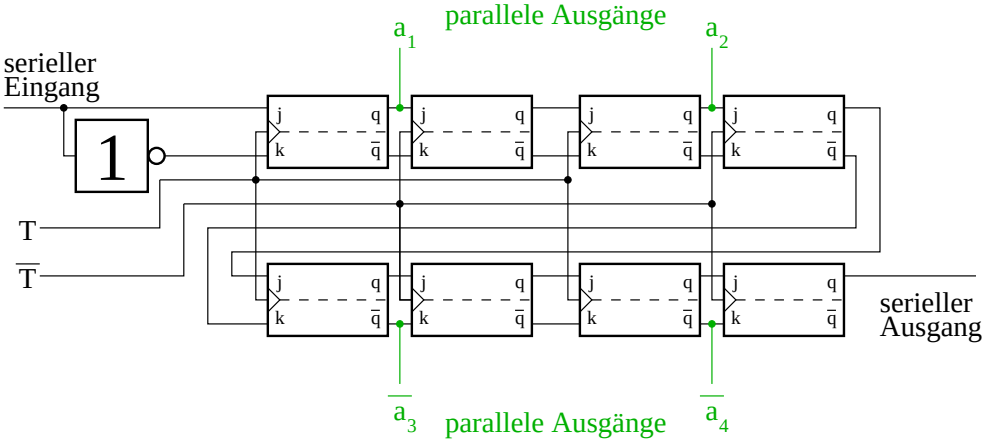


Abbildung 4.30: 4-Bit Schieberegister mit parallelen Ausgängen

Abbildung 4.30 zeigt ein kombiniertes Schieberegister, daß sowohl seriellen Schreib- und Lesezugriff als auch parallelen Lesezugriff erlaubt.

4.6 Entwurfsbeispiel: Getränkteautomat

4.6.1 Aufgabenstellung

Es soll ein einfacher Getränkeautomat entworfen werden, der beim Einwurf von mindestens 1,50 Euro ein Getränk ausgibt. Der Automat soll einen Münzschlitz haben, und er soll 50-Cent- und 1-Euro-Münzen akzeptieren. Ein mechanischer Sensor soll der Steuerung anzeigen, ob 50 Cent oder 1 Euro eingeworfen wurden. Ausgabe besteht in einem elektronischen Signal, das die Ausgabe der Flasche auslößt. Überzahlung ist möglich, es wird allerdings kein Rückgeld erstattet.

4.6.2 Spezifikation

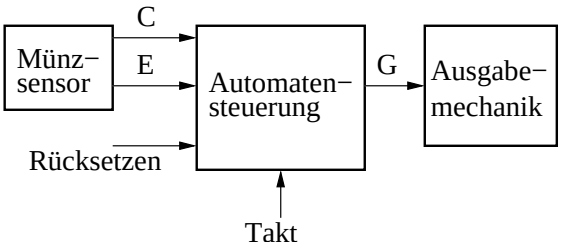


Abbildung 4.31: Blockdiagramm des Getränkeautomaten

- Eingänge des Automaten (Münzsensor):
  - C 50-Cent-Münze eingeworfen
  - E 1-Euro-Münze eingeworfen
- Ausgang des Automaten (Ausgabemechanik):
  - G Getränk ausgeben
- Es wird extern dafür gesorgt, daß andere Münzen den Münzschlitz ohne Einwirkungen passieren und wieder ausgeworfen werden.
- Nach der Getränkeausgabe wird ein Rücksetzen veranlaßt.

Daraus ergeben sich folgende mögliche Eingaben:

- E, C - 1,50 Euro
- C, E - 1,50 Euro
- C, C, C - 1,50 Euro
- C, C, E - 2,00 Euro (Überzahlung)
- E, E - 2,00 Euro (Überzahlung)

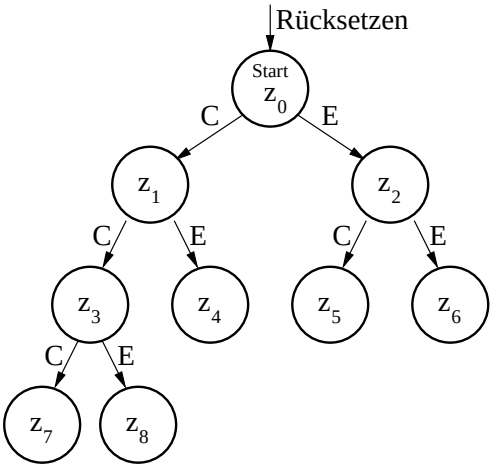


Abbildung 4.32: Zustandsgraph des Getränkeautomaten (nicht vollständig)

Zu Optimierung dieses Graphen können Zustände mit gleichen Eigenschaften zusammengefaßt werden.

$$z_2, z_3 : z_2^*$$
$$z_4, z_5, z_6, z_7, z_8 : z_3^*$$

Hierbei müssen die 1,50 Euro- und 2,00 Euro-Klassen nicht unterschieden werden, da diese Unterscheidung im Bezug auf die Aufgabenstellung irrelevant ist (es wird kein Wechselgeld zurückerstattet).

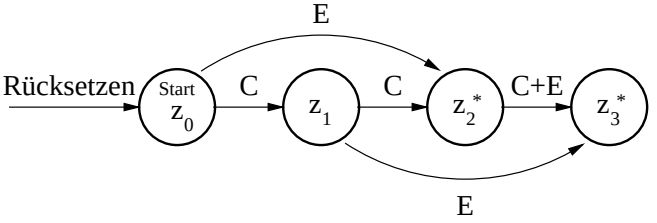


Abbildung 4.33: Optimierter Übergangsgraph (nicht vollständig)

Index	Eingänge		Zustand			Folgezustand			Ausgang
	E	C		q_1^t	q_0^t		q_1^{t+1}	q_0^{t+1}	G^t
0	0	0	z_0	0	0	z_0	0	0	0
1	0	1	z_0	0	0	z_1	0	1	0
2	1	0	z_0	0	0	z_2^*	1	0	0
3	1	1	z_0	0	0	X	X	X	X
4	0	0	z_1	0	1	z_1	0	1	0
5	0	1	z_1	0	1	z_2^*	1	0	0
6	1	0	z_1	0	1	z_3^*	1	1	0
7	1	1	z_1	0	1	X	X	X	X
8	0	0	z_2^*	1	0	z_2^*	1	0	0
9	0	1	z_2^*	1	0	z_3^*	1	1	0
10	1	0	z_2^*	1	0	z_3^*	1	1	0
11	1	1	z_2^*	1	0	X	X	X	X
12	0	0	z_3^*	1	1	z_3^*	1	1	1
13	0	1	z_3^*	1	1	z_3^*	1	1	1
14	1	0	z_3^*	1	1	z_3^*	1	1	1
15	1	1	z_3^*	1	1	X	X	X	X

Tabelle 4.9: Übergangstabelle des Getränkeautomaten

4.6.3 Flipflop-Ansteuerung und Ausgangsfunktion

Als Zustandsspeicher sollen JK-Flipflops verwendet werden. Zu Erinnerung noch einmal die Ansteuergleichungen für ein JK-Flipflop:

$$q^{t+1} = j\overline{q}^t + \overline{k}q^t$$
$$j^t = q^{t+1}\big|_{q^t=0}$$
$$k^t = \overline{q}^{t+1}\big|_{q^t=1}$$

Anmerkung: In der Praxis empfiehlt es sich allerdings in solchen Situationen auf die besser geeigneteren D-Flipflops zurückzugreifen. Hier sollen die JK-Flipflops nur zur Übung verwendet werden.

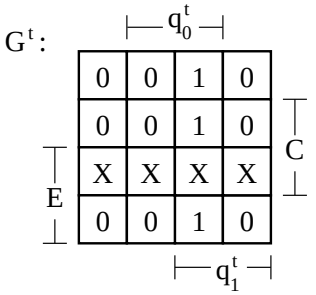


Abbildung 4.34: KV Diagramm für die Ausgangsfunktion

Es ergibt sich folgende Ausgangsfunktion:

$$G^t = q_1^t \cdot q_0^t$$

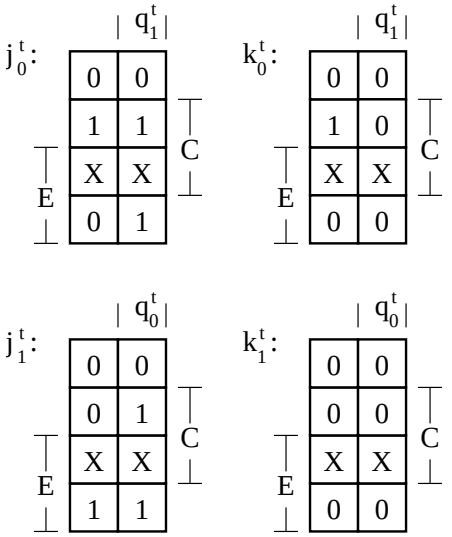


Abbildung 4.35: KV Diagramme für die Flipflop-Ansteuergleichungen

Aus den KV-Diagrammen ergeben sich folgende Ansteuergleichungen:

$$j_0^t = q_0^{t+1}\big|_{q_0^t=0} = C^t + q_1^t \cdot E^t$$
$$k_0^t = q_0^{t+1}\big|_{q_0^t=1} = \overline{q}_1^t \cdot C^t$$

$$j_1^t = q_1^{t+1}\big|_{q_1^t=0} = E^t + q_0^t \cdot C^t$$
$$k_1^t = q_1^{t+1}\big|_{q_1^t=1} = 0$$

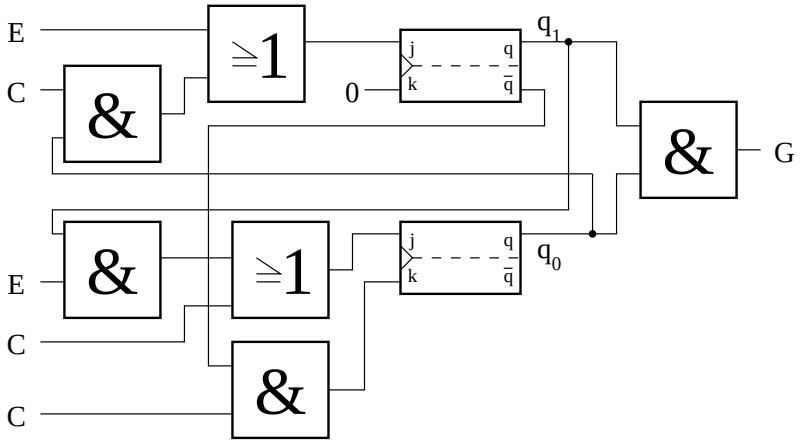


Abbildung 4.36: Schaltbild der Steuerung des Getränkeautomaten