Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität Kiel
Olshausenstr. 40-60
D-2300 Kiel 1
Tel. (0431) 880 - 2766

Ein praktisches Verfahren zum Entwurf synchroner Schaltwerke

P. Marwedel

Bericht Nr. 1/77
Januar 1977

Diesen Bericht hat der Verfasser im Rahmen der Forschungsgruppe "Rechnerorganisation und Schaltwerke" des überregionalen Forschungsprogramms Informatik des Bundesministeriums für Forschung und Technologie am Institut für Informatik und Praktische Mathematik an der Universität Kiel angefertigt.

Inhaltsverzeichnis

1. Einleitung

Wenn man sich mit dem praktischen Entwurf synchroner Schaltwerke beschäftigt, so hat man bislang im wesentlichen zwei Möglichkeiten : 1. Von einigen Flipflops und Gattern ausgehend konstruiert man sich ein Schaltwerk, das man anschliessend solange erweitert, bis es die gestellten Anforderungen vermeintlich erfüllt. Diese Methode ist nicht nur unsystematisch, sie führt auch zu Fehlern. 2. Man stellt eine Automatentafel auf und benutzt dann eine Rechenanlage zum Auffinden einer optimalen Zustandskodierung und zur Minimierung. Diese Methode setzt den Zugang zu einer Rechenanlage voraus. Die Automatentafeln sind häufig sehr groß und unanschaulich. Die Rechenprogramme gehen von vereinfachten Kostenfunktionen aus und sind z.T. sehr speicherintensiv.

Ziel der hier dargelegten Arbeit war die Entwicklung eines Verfahrens, das es dem Entwerfer gestattet, auch bei geringen Stückzahlen mittelgroße Schaltungen kostengünstig zu entwickeln und verständlich zu dokumentieren. Das Verfahren setzt eine Zustandskodierung voraus. Erfahrungen mit Algorithmen zum Auffinden einer optimalen Zustandskodierung [4] haben gezeigt, daß der Erfolg dieser Programme in der Regel recht bescheiden ist. Durch Verzicht auf eine optimale Zustandskodierung kann eine Rechnerunterstützung (und damit stückzahlunabhängige Kosten) vermieden werden. Zustandskodierungen werden dem Entwerfer häufig z.B. durch die Ausgabefunktion nahegelegt. - Der Verständlichkeit und besseren Dokumentation dient die Verwendung von Zustandsgraphen. Das Verfahren liefert Schaltwerke mit Störunterdrückungseigenschaften, in einer speziellen Realisierung (mit Multiplexern) kann die Äquivalenz von Schaltwerk und Zustandsgraph ohne Umwege geprüft werden.

2. Entwicklung eines Verfahrens zum Entwurf synchroner Schaltnetze

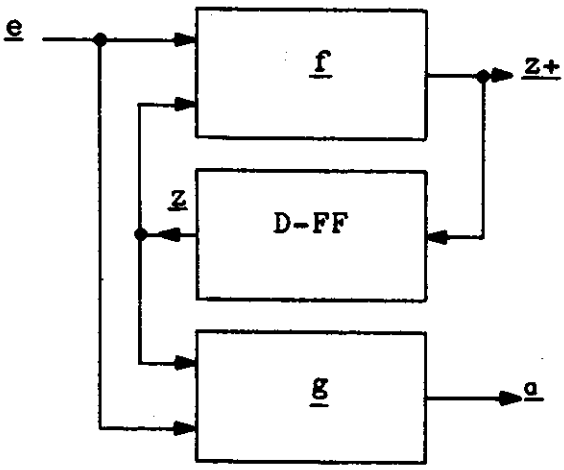
Ein synchrones Schaltwerk wird beschrieben durch ein Paar von Vektorgleichungen

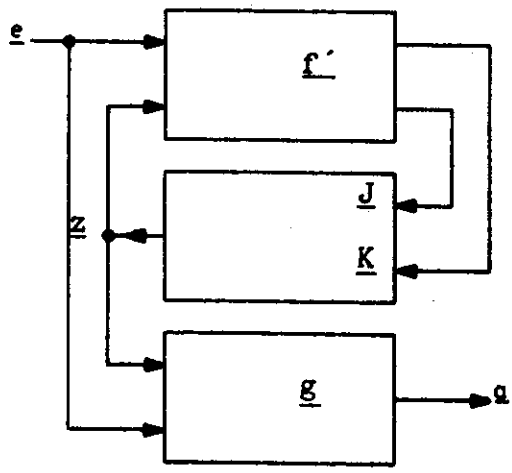
$$\underline{z}+ = \underline{f}(\underline{e},\underline{z}) \qquad \underline{a} = \underline{g}(\underline{e},\underline{z}) \qquad 1)$$

mit den zugehoerigen Komponentengleichungen

$$z+_j = f_j \qquad a_j = g_j \qquad 2)$$

Die Realisierung dieser Gleichungen erfolgt ueblicherweise mit D-, RS- oder JK-Flipflops nach dem folgenden Schema :





Ausser beim D-Flipflop steht der Folgezustand $\underline{z}+_j$ nicht direkt zur Verfuegung. Waehrend die einzelnen D-FFs direkt mit jeweils einer Komponente von f beschaltet werden, ergeben sich die Beschaltungen fuer RS- und JK-Flipflop aus dem Vergleich ihrer charakteristischen Gleichungen mit der Uebergangsfunktion des Schaltwerks [1]. Diese separiert man zu diesem Zweck nach dem gerade betrachteten Zustandsbit :

$$\underline{z}+_j = f_j(\underline{e},..z_j=0,..)*\overline{z_j} + f_j(\underline{e},..,z_j=1,..)*z_j \qquad 3)$$

Die charakteristischen Gleichungen lauten :

fuer das JK-Flipflop: $z+_j = J_j z_j + K_j z_j$ 4)

fuer das RS-Flipflop: $z+_j = S_j \overline{z}_j + \overline{R}_j z_j / S_j * R_j = 0.$ 5)

Durch Vergleich erhaelt man eine moegliche Loesung:

$$J_j = f_j(\underline{e},..,z_j=0,..) \text{ und } \quad K_j = \overline{f_j(\underline{e},..,z_j=1,..)} \qquad 6)$$

Waehlt man nun:

$$S_j = f_j(\underline{e},..,z_j=0,..)*z_j \text{ und } R_j = \overline{f_j(\underline{e},..,z_j=1,..)*z_j} \qquad 7)$$

so erhaelt man aus 4):

$$z+_j = f_j(\underline{e},..,z_j=0,..)*\overline{z_j}*\overline{z_j} + \overline{f_j(\underline{e},..,z_j=1,..)}*z_j*z_j$$

$$z+_j = f_j(\underline{e},..,z_j=0,..)*\overline{z_j} + [f_j(\underline{e},..,z_j=1,..)+\overline{z_j}]*z_j$$

$$= f_j(\underline{e},..,z_j=0,..)*\overline{z_j} + f_j(\underline{e},..,z_j=1,..)*z_j$$

Die Gleichungen 7) erzeugen so die gewuenschte Uebergangsfunktion und genuegen der Randbedingung $S_j * R_j = 0$ wegen $z_j * \overline{z_j} = 0$. Ist f_j dargestellt als Summe von Produkten, so streicht die Einschraenkung auf die Stelle $z_j=0$ Terme, die z_j enthalten und kuerzt jene, die $\overline{z_j}$ enthalten. Die Multiplikation mit $\overline{z_j}$ streicht ebenfalls jene Terme, die z_j enthalten und macht bei jenen, die $\overline{z_j}$ enthalten, die Kuerzung wieder rueckgaengig. Daher kann die Einschraenkung entfallen:

$$S_j = f_j(\underline{e},\underline{z})*\overline{z_j} = z_j+ * \overline{z_j}$$

und $$R_j = \overline{f_j(\underline{e},\underline{z})}*z_j = \overline{z_j+} * z_j \qquad 8)$$

Die letzten Formeln zeigen:
S ist genau dann Eins, wenn das Zustandsbit z von 0 auf 1 wechselt und R ist genau dann Eins, wenn z von 1 auf 0 wechselt. Um diese Tatsache zur direkten Niederschrift der Flipflopbeschaltungen aus den Zustandsgraphen oder Flussdiagrammen ausnutzen zu koennen, separieren wir die Funktion f_j vollstaendig nach allen Variablen z_j:

$$f_j = \sum_k f_{jk}(\underline{e},\underline{z})$$

und erhalten f_j in einer Summendarstellung ueber die moeglichen Zustaende. f_{jk} ist derjenige Anteil von f_j, der aus Uebergaengen vom Zustand k aus entsteht. Da diese Anteile disjunkt sind (jeder enthaelt eine anderen Minterm in den z_j), gilt:

$$\overline{f_j} = \sum_k \overline{f_{jk}(\underline{e},\underline{z})}.$$

Daraus folgt:

$$J_j = (\sum_k f_{jk}(\underline{e},..,z_j=0,..)) \qquad K_j = (\overline{\sum_k f_{jk}(\underline{e},..,z_j=1,..)})$$

$$J_j = \sum_k f_{jk}(\underline{e},..,z_j=0,..) \qquad K_j = \sum_k \overline{f_{jk}(\underline{e},..,z_j=1,..)} \qquad 9)$$

$$S_j = \sum_k f_{jk}(\underline{e},\underline{z}) * \overline{z_j} \qquad R_j = \sum_k \overline{f_{jk}(\underline{e},\underline{z}) * z_j} \qquad 10)$$

Aus Formel 10) erkennt man nun, dass man die Summanden fuer S bzw. R Zustand fuer Zustand aufaddieren kann. Fuer jeden Zustand schreibt man also an, wann das Bit z von 0 auf 1 bzw von 0 auf 1 wechselt. Damit faehrt man -die einzelnen Terme addierend- fort, bis alle Zustaende erfasst sind. Dies kann

direkt aus dem Zustandsgraphen oder aus dem Ablaufdiagramm heraus geschehen.
Die Einschraenkungen der Funktion bedeuten, dass man bei den entsprechenden Termen in 9) die Variablen z_j bzw. $\bar{z}_j$ streichen darf. In S_j und J_j brauchen nur solche Zustaende beruecksichtigt werden, bei denen $z_j = 0$ ist. Bei R_j und K_j liefern nur Zustaende mit $z_j = 1$ Beitrage zur Summe.
Die ausgewaehlten Loesungen der Gleichungspaare 3) und 4) bzw 3) und 5) haben folgende Vorteile:

1. Aufgrund der einfachen Konstruktionsvorschriften nach den Gleichungen 7) und 8) koennen die Beschaltungsfunktionen direkt aus Zustandsgraphen oder Flussdiagrammen niedergeschrieben werden. Wertetabellen entfallen ganz.
2. Es gehen nur Wechsel der Zustandsbits in die Rechnung ein, was folgende Vorteile hat:
   2.1 Bei Ueberganengen von Zustaenden in sich bleiben alle Zustandsbits erhalten. Diese Uebergaenge brauchen daher nicht eingezeichnet zu werden.
   2.2 Haeufig ist bei gegebenem Problem nur bekannt, wann Uebergaenge stattfinden, nicht wann sie unterbleiben sollen. In diesen Faellen entfaellt die Komplementbildung.
   2.3 Im Ggs. zur Realisierung mit D-Flipflops entfaellt die Beschaltung der Eingaenge fuer Uebergaenge der Zustaende in sich.
   2.4 Tauchen durch Stoerungen Eingaben auf, die beim Entwurf nicht eingeplant waren, so koennen sie keine Uebergaenge bewirken.
3. In Verbindung mit dem von Reusch angegebenen Verfahren [1] koennen die Primterme der Beschaltungsfunktionen leicht gefunden werden.

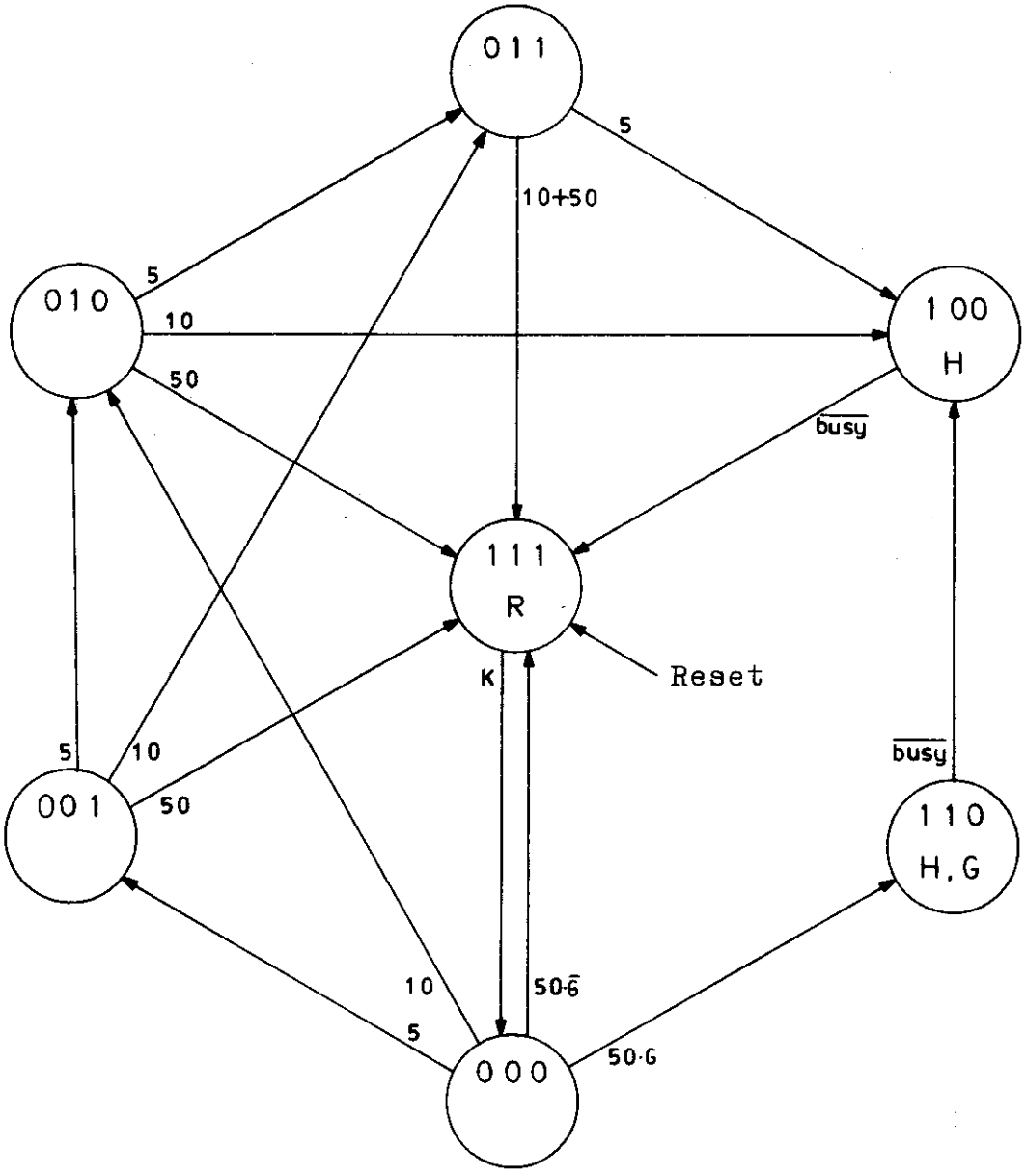
Satz (Reusch) : Sei $f = f(..,x_j=0,..)*\bar{x}_j + f(..,x_j=1,..)*x_j$ und seien $P=\{P1,..,Pl\}$ und $Q=\{Q1,...,Qk\}$ die Primimplikanten von $f(..,x_j=0,..)$ bzw. $f(..,x_j=1,..)$. Dann enthaelt die Menge $M = \{P1*Q1,..,Pl*Qk,P1*x_j,..,Pl*x_j,Q1*\bar{x}_j,..,Qk*\bar{x}_j\}$ alle Primimplikanten von f.
Bemerkung:

i) Streicht man diejenigen Terme von M, von denen eine Verkuerzung in M enthalten ist, so erhaelt man genau die Menge aller Primimplikanten.
ii) Sind die Primterme der eingeschraenkten Funktionen nicht bekannt, so wiederholt man das Verfahren, bis einzelne Variablen, Nullen oder Einsen erscheinen.
iii) Bei der Bildung der Menge M uebernimmt man zunaechst in P und Q gleiche Elemente unveraendert nach M und streicht diese fuer die weitere Bildung von P und Q.

Ein Vorteil der direkten Methode zur Aufstellung der Beschaltungsfunktionen ist, dass die Entwicklung nach den z_j bereits vorweggenommen ist.

Beispiel
Zu konstruieren sei ein Kaffeeautomat. Eine Tasse (subventionierten) Kaffees soll 20 Pfennige kosten. Es soll 2 Moeglichkeiten geben: 1. Man kann 5- und 10-Pfennigstuecke in beliebiger Reihenfolge einwerfen, bis 20 Pfennige erreicht sind, und dann eine Portion Kaffee in Empfang nehmen. 2. Man kann ein 50 Pfennigstueck einwerfen

und erhaelt dann eine doppelte Portion Kaffee und einen Groschen zurueck. Das Geld jedes Kunden soll zwischengespeichert werden und bei unsinnigen Eingaben oder fehlenden Vorraeten zurueckerstattet werden. Es gibt demnach folgende Eingaben : 5 Pfennig, 10 Pfennig, 50 Pfennig, Kaffee und Wasser (fuer mindestens 2 Tassen) vorraetig und 10 Pfennig vorraetig, sowie die Ausgaben: gebe Kaffee aus, gebe Groschen zurueck, leere Zwischenspeicher in Hauptspeicher, entleere Zwischenspeicher ins Rueckgabefach, sowie (zur Vereinfachung des Schaltwerks) sperre die Eingabe. Fuer die ersten drei Eingaben existieren Vorrichtungen, die sie pro Eingabe nur einen Uebergang ausloesen lassen.
Bezeichnung der Ausgaben :
H Entleere Geld in Hauptspeicher,gebe Kaffee aus, sperre Eingabe
R Entleere Zwischenspeicher ins Rueckgabefach
G gebe Groschen zurueck
Eine moegliche Beschreibung des Automaten ist die folgende :



Daraus erhaelt man die folgenden Gleichungen :

$S_2 = 50*\bar{z}_2\bar{z}_1\bar{z}_0+50*\bar{z}_2\bar{z}_1 z_0+(10+50)*\bar{z}_2 z_1\bar{z}_0+(5+10+50)*\bar{z}_2 z_1 z_0$

$J_2 = 50*\bar{z}_1\bar{z}_0 + 50*\bar{z}_1 z_0 + (10+50)*z_1\bar{z}_0 + (5+10+50)*z_1 z_0$

$R_2 = K*z_2 z_1 z_0$

$K_2 = K*z_1 z_0$

$S_1 = (10+50)*\bar{z}_2\bar{z}_1\bar{z}_0+(5+10+50)*\bar{z}_2\bar{z}_1 z_0+\overline{busy}*z_2\bar{z}_1\bar{z}_0$

$J_1 = (10+50)*\bar{z}_2\bar{z}_0 + (5+10+50)*\bar{z}_2 z_0 + \overline{busy}*z_2\bar{z}_0$

$R_1 = 10*\bar{z}_2 z_1\bar{z}_0+5*\bar{z}_2 z_1 z_0+\overline{busy}*z_2 z_1\bar{z}_0$

$K_1 = 10*\bar{z}_2 z_0 + 5*\bar{z}_2 z_0 + \overline{busy}*z_2\bar{z}_0$

$S_0 = (5+50*\bar{G})*\bar{z}_2\bar{z}_1\bar{z}_0+(5+50)*\bar{z}_2 z_1\bar{z}_0+\overline{busy}*z_2\bar{z}_1\bar{z}_0$

$J_0 = (5+50*\bar{G})*\bar{z}_2\bar{z}_1 + (5+50)*\bar{z}_2 z_1 + \overline{busy}*z_2\bar{z}_1$

$R_0 = 5*\bar{z}_2\bar{z}_1 z_0+5*\bar{z}_2 z_1 z_0+K*z_2 z_1 z_0$

$K_0 = 5*\bar{z}_2\bar{z}_1 + 5*\bar{z}_2 z_1 + K*z_2 z_1$

Die Bestimmung der Primterme sei am Beispiel gezeigt :

$J_1 = (10+50)*\bar{z}_2\bar{z}_0 + (5+10+50)*\bar{z}_2 z_0 + \overline{busy}*z_2\bar{z}_0$

$J_1(..,z_2=0,..) = (10+50+5*z_0)$, $J(..,z_2=1,..) = \overline{busy}*\bar{z}_0$

$J_1 = 10\,\bar{z}_2 + 50\,\bar{z}_2 + 5*\bar{z}_2*z_0 + \overline{busy}*z_2\bar{z}_0 + 10*\overline{busy}*\bar{z}_0 +$

$+50*\overline{busy}*\bar{z}_0$

Diese Summe enthaelt alle Primterme.

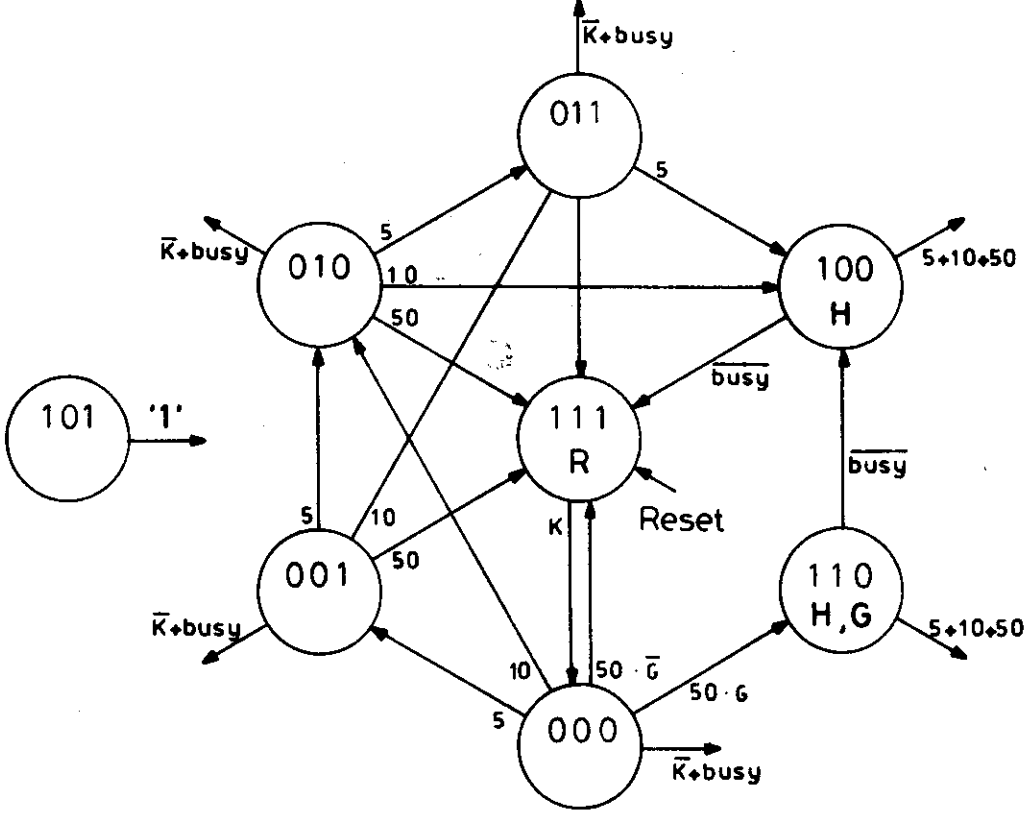
3. Berücksichtigung zusätzlicher Redundanzen

Verzichtet man auf die Unterdrueckung aller Uebergaenge durch unerwartete Eingaben, so kann man von Redundanzen Gebrauch machen. Seien die Funktionen J,J´,j sowie K,K´,k derart gegebenen, dass gilt :

J=1 -> J´=1, (J=0 & j=0) -> J´=0
K=1 -> K´=1, (K=0 & k=0) -> K´=0

Bekanntlich ist dann jede irredundante Formel fuer J´ und K´ eine Summe von Pimimplikanten von (J+j) bzw. (K+k). Daher kann man das vorangegangene Verfahren fuer (J+j) bzw (K+k) wiederholen und auf diese Weise die Primimplikanten der Beschaltungsfunktionen fuer redundante Graphen erhalten. Die Funktionen J_j und K_j stimmen mit den bisherigen J_j und K_j ueberein, waehrend j_j bzw. k_j angeben, wann J_j und K_j auch noch 1 werden duerfen.

Der folgende Graph ist um "unmoegliche" Uebergaenge erweitert :



Zum Beispiel kann die Variable K nur waehrend des Kaffeeschenkens auf Null gehen, nicht aber waehrend der Automat bereits Muenzen annimmt. Aus dem Graphen liest man z.B. ab :

$$j_1 = (\overline{K}+busy)*\overline{z}_2\,\overline{z}_0 + (\overline{K}+busy)*\overline{z}_2\,z_0 + (5+10+50)*z_2\,\overline{z}_0 + z_2\,z_0$$

Damit ist :

$$J_1+j_1 = (10+50+\overline{K}+busy)*\overline{z}_2\,\overline{z}_0 + (5+10+50+\overline{K}+busy)*\overline{z}_2\,z_0 +$$
$$+ (5+10+50+busy)*z_2\,\overline{z}_0 + z_2\,z_0 \ .$$

Die Primterme gewinnt man aus folgender Rechnung :

$$(J_1+j_1)(..,z_2=0,..) = 10 + 50 + \overline{K} + 5z_0 + busy$$

$$(J_1+j_1)(..,z_2=1,..) = 5 + 10 + 50 + busy + z_0$$

$$J_1+j_1 = 10 + 50 + busy + 5*z_0 + 5*\overline{K} + \overline{K}*z_0$$

4. Technische Realisierung

Die endgueltige technische Realisierung kann u.a. auf folgende Arten erreicht werden :

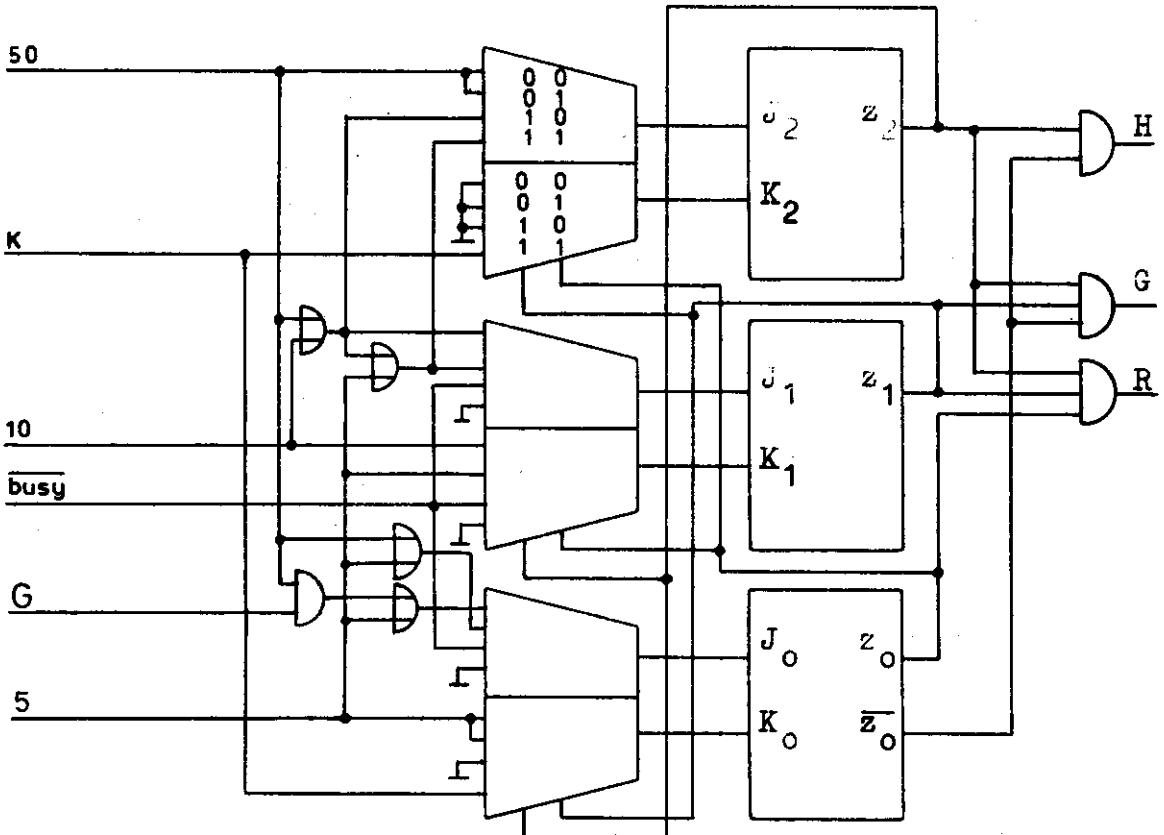
1. Mit NAND/AND/NOR/OR - Gattern ("SSI"=smale scale integration) werden die Gleichungen fuer R und S realisiert.
2. Unter Verwendung eines Dekoders wird eine 1 aus n Zustandskodierung erzeugt und die S bzw. R- Gleichungen dadurch reduziert.
3. Die JK - Gleichungen werden mit SSI - Schaltkreisen realisiert.
4. Es werden Multiplexer verwendet, deren Steuereingaenge mit den Zustandsbits beschaltet werden. Keinerlei Rechnungen.

5.wie 3., jedoch Minimierung
6.wie 4., jedoch Minimierung
7.wie 5., jedoch Verwendung der unvollstaendigen Funktion.
8.wie 6., jedoch Verwendung der unvollstaendigen Funktion.
9.Benutzung eines programmierbaren logischen Arrays (PLA)

Eine Uebersicht ueber den jeweiligen Aufwand (in IC's) liefert die folgende Tabelle anhand einiger Beispiele (je 8 Zustaende)

Fall	1	2	3	4	5	6	7	8
FF-Typ	RS	RS	JK	JK	JK	JK	JK	JK
Gatter-Typ	SSI	Dek	SSI	Mpx	SSI	Mpx	SSI	Mpx
Rechnung	-	-	-	-	Minimierung		unvollstaend.	
I	-	10	10	5,33	7	5	6	5
II	-	-	-	-	5,25	4	-	-
III	7	6,33	5,25	3,75	5,25	3,75	-	-
IV	-	-	-	-	2,5	3,5	1,5	3,5

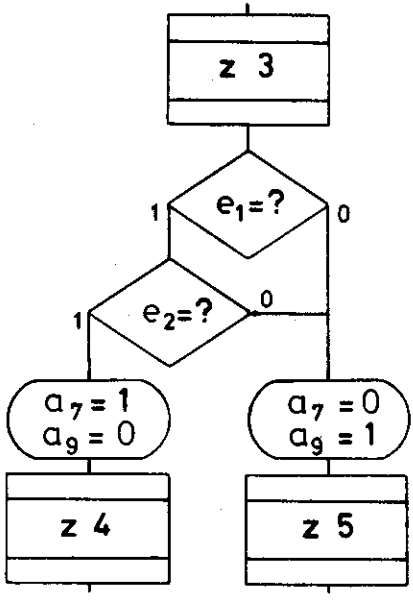
Im Fall I waere ein einziges PLA-IC noch nicht einmal zur Haelfte ausgenutzt, jedoch sind die Kosten bislang noch zu hoch. Die Tabelle zeigt den Vorteil der JK-Loesung mit Multiplexer: bereits ohne Rechnung wird ein kostenguenstiger Aufwand erreicht. Die Minimierung bleibt hier relativ erfolglos, da nicht ueber mehrere Zustaende minimiert werden kann. Lediglich bei sehr einfachen Beschaltungsfunktionen (wie dem Fall IV) ist eine SSI-Loesung guenstiger, da bei der Multiplexer - Realisierung der Mindestaufwand fuer die Multiplexer immer vorhanden ist. Ebenso wird man fuer besonders schnelle Schaltungen die SSI - Loesung vorziehen.
Das folgende Bild zeigt den Kaffeeautomaten in einer Schaltung nach Fall 6.



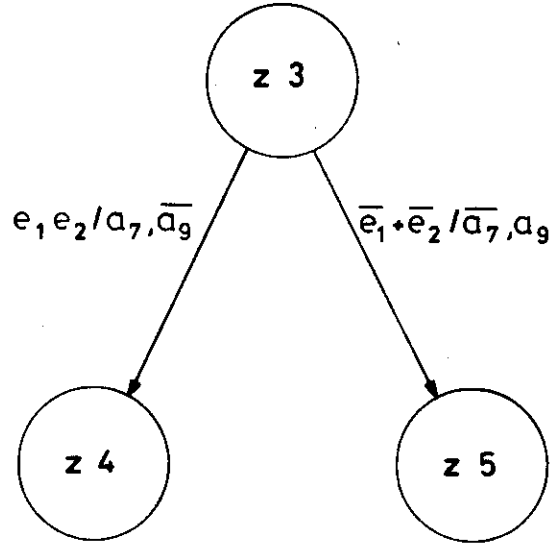
5. Varianten der graphischen Beschreibung bei verschiedenen Automatentypen

Die Gleichungen 1) und 2), von denen wir im vorangegangenen Abschnitt ausgegangen sind, beschreiben Automaten vom Mealy - Typ. Weitere Beschreibungsmoeglichkeiten sind die folgenden :

Flussdiagramm

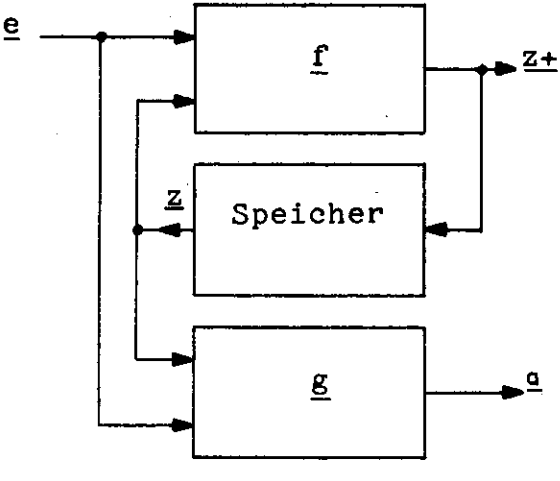


Zustandsgraph



Die rechteckigen Kaesten, sogenannte dynamische Wertzuweisungen [3], kennzeichnen die jeweils durch Taktimpulse uebernommenen neuen Zustaende.

Blockschaltbild



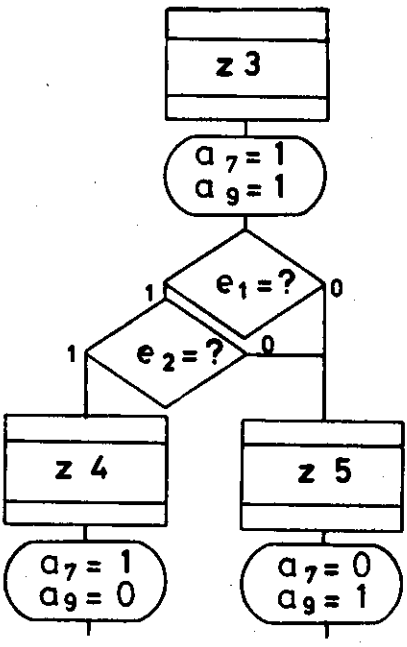
Ausgabefunktion

a7 = z3*e1*e2 + ..

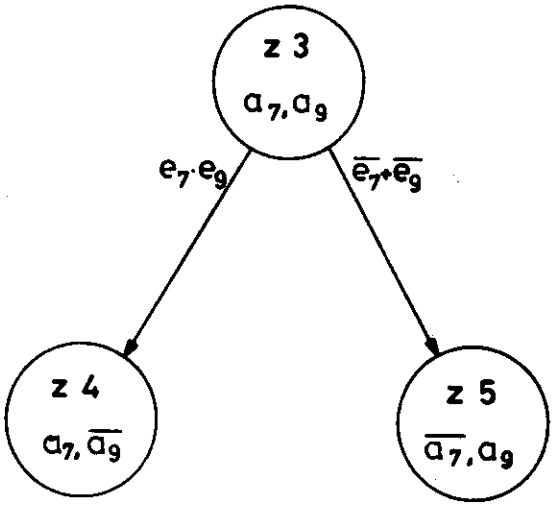
a9 = z3*(ē1+ē2)

Ein Spezialfall des Mealy - Automaten ist der Speicher - Automat [3], bei dem die Ausgabefunktion nicht von der Eingabe abhaengt ($\underline{a}=\underline{g}(\underline{z})$). Man kann dann die vorhergehenden Diagramme etwas umzeichnen :

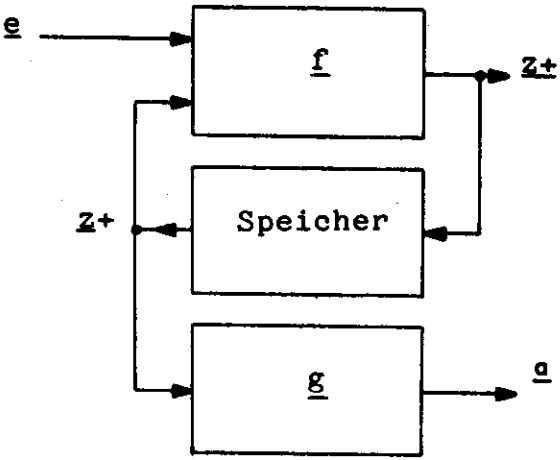
Flussdiagramm



Zustandsgraph



Blockschaltbild



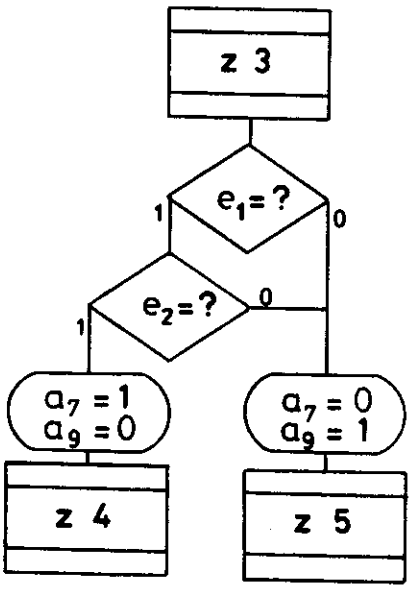
Ausgabefunktion

$a_7 = z3 + z4 + ..$

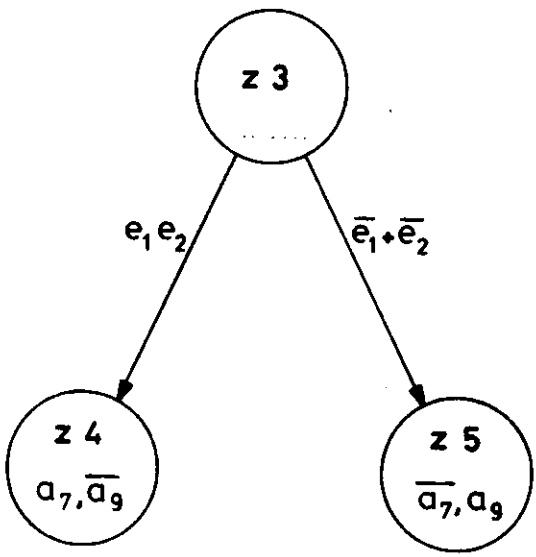
$a_9 = z3 + z5 + ..$

Neben dem vorgenannten Automaten wird haufig noch der Moore - Automat verwendet mit der Ausgabefunktion $\underline{a}=\underline{g}(\underline{z}+)$ und den graphischen Darstellungen

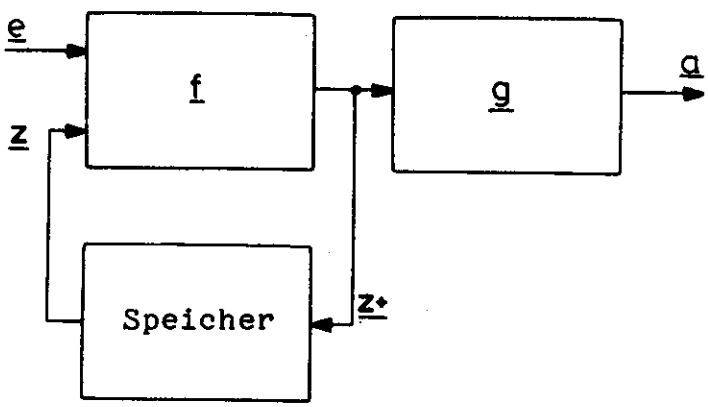
Flussdiagramm



Zustandsgraph



Blockschaltbild



Ausgabefunktion

$a_7 = z4^* + \ldots.$

$a_9 = z5^* + \ldots.$

Beim Zusammenschalten von synchronen Schaltwerken koennen asynchrone Rueckkopplungen auftreten, wenn nicht in allen Rueckkopplungswegen getaktete Speicher liegen. Asynchrone Rueckkopplungen werden vermieden, wenn hoechstens ein Schaltwerk nicht vom Speicher - Typ ist. Sowohl beim Mealy als auch beim Moore - Automaten liegen zwischen Ein - und Ausgabeleitungsbuendeln keine getakteten Speicher. Damit besitzt der Moore - Automat folgende Nachteile :

1. Da die Ausgabe nur von den Zustaenden abhaengen darf, werden fuer ein gegebenes Problem relativ viele Zustaende benoetigt.
2. Im Ggs. zum Speicherautomaten koennen trotzdem asynchrone Rueckkopplungen auftreten [4].
3. Eine direkte Realisierung ist nur mit D - Flipflops moeglich, da nur hier $\underline{z}+$ explizit zur Verfuegung steht.

Versieht man den Moore - Automaten mit einem Ausgabespeicher um diese Nachteile zu beseitigen, so verhaelt er sich im wesentlichen wie ein Speicher - Automat.

Quellenverzeichnis

1 W. Giloi und H. Liebig : Logischer Entwurf digitaler Systeme, Berlin, 1973

2 B. Reusch : Generation of Prime Implicants from Subfunctions and a Unifying Approach to the Covering Problem, IEEE Trans C-24 (1975), 924-930

3 S. Wendt : Entwurf komplexer Schaltwerke, Berlin, 1974

4 J. Beister, persönliches Gespräch, Dortmund, 1976

Berichte

aus dem Institut für Informatik und Praktische Mathematik der Universität Kiel

1/74 Schadach, D.: Teilraummethoden in der Zeichenerkennung.

2/74 Jürgensen, H.: On Some Semigroup-Theoretic Aspects of the Theories of Automata and Formal Languages.

1/75 Hansen, R., Hoffmann, E.G., Simon, F.: ALTID, eine algorithmische Sprache für Lehr- und Informationsdialoge.

2/75 Kalhoff, B., Simon, F.: Programmieren in LISP 1.5 - Benutzerhandbuch - .

3/75 Schmeck, H.: Korrektheit von Übersetzungen.

4/75 Schadach, D.: Grundlagen und Anwendungen einer nicht-Booleschen Informationstheorie auf Teilraumverbänden von Tensorprodukten separabler Hilbert-Räume.

1/76 Jürgensen, H., Kalmbacher, J., Wick, P.: Halbgruppenprogramme.

2/76 Langmaack, H.: Eine Einführung in Aufgaben des Übersetzerbaus für Programmiersprachen.

3/76 Göbel, D.: Über Zerlegung, Approximation und approximative Zerlegung von stochastischen Automaten.

4/76 Schadach, D.: A Simple Algorithm for Maximum-Likelihood Factor Analysis and Its Application to Some Sets of Data.

5/76 Schmeck, H.: Zur Theorie der Mikroprogrammierung

6/76 Kölsch, R.T.: Untersuchungen zur Emulation des PASCAL-Stackcomputers auf der Zentraleinheit 7.750 .