Abstrakte Automaten, algebraische Feedback-Theorie und Entwurf digitaler Folgeschaltungen. Teil 1[1)]

K.-A. Zech, Berlin

Mitteilung aus dem Institut für Nachrichtentechnik Berlin

Deskriptoren: **Automatentheorie, rechnergestützter Schaltungsentwurf;** digitale synchrone Folgeschaltung; algebraische Strukturtheorie endlicher Automaten; Feedbackpartition

Im Zuge der Bestrebungen, informationsverarbeitende Prozesse zu formalisieren und theoretisch zu erfassen, entwickelten sich zu Beginn der 50er Jahre die ursprünglich 2 Richtungen der Automatentheorie: die Theorie der abstrakten Automaten (sequentieller Maschinen) [1] [2] [3] und die mehr mathematisch orientierte Theorie der logischen Netze [4] [5]. Inzwischen sind diese beiden Gebiete nicht mehr zu trennen, betrachten sie doch äquivalente Objekte [6] [7] [12].

Bei der Formulierung von Entwurfsproblemen der digitalen Schaltungstechnik für die Rechnerunterstützung hat sich in großem Maße das Modell des abstrakten Automaten durchgesetzt. Es ist einerseits hinreichend abstrakt, um allgemeine Verfahren günstig anwenden zu können, bietet aber andererseits noch genügend Anschauung [1] [3].

Ist das Verhalten einer zu entwerfenden Folgeschaltung durch die Tabellen eines abstrakten Automaten vorgeschrieben, so gilt es, diesen durch eine Codierung seiner Parameter in eine (binäre) digitale Schaltung umzusetzen. Dabei kommt es insbesondere darauf an, den notwendigen Realisierungsaufwand zu minimieren. Neben intuitiven Algorithmen, die diese Forderung berücksichtigen, hat darauf die im wesentlichen auf die Mathematiker *J. Hartmanis* und *R. E. Stearns* zurückgehende algebraische Strukturtheorie der endlichen Automaten [7] bedeutenden Einfluß gewonnen. Bei ihr werden die Eigenschaften bestimmter Algebren betrachtet, die die reduzierten Abhängigkeiten der Speicherelemente einer Folgeschaltung untereinander widerspiegeln.

In [7] ist ein Kapitel dem Problem gewidmet, einen Automaten durch eine in bestimmter Weise stufenweise verschaltete Anordnung von Komponenten darzustellen. Innerhalb dieser Struktur ist der „Informationsfluß" nur in einer Richtung erlaubt. Notwendige Rückführungen werden durch eine sogenannte Feedback-Funktion realisiert.

Im vorliegenden Beitrag wird eine Möglichkeit beschrieben, abstrakte Automaten bei Vorgabe derartiger Funktionen mit reduziertem Aufwand zu codieren. Das Verfahren entstand bei der Bearbeitung der Aufgabe, ein spezielles digitales Entwurfsproblem aus der Nachrichtentechnik mit automatentheoretischen Methoden zu lösen. Es ergaben sich Schaltungen, deren Aufwand mit dem des „Handentwurfs" vergleichbar war.

1. Zusammenhang zwischen abstrakten Automaten und digitalen Folgeschaltungen

Eine digitale, synchron-getaktete Folgeschaltung ist eine Verknüpfung von kombinatorischen und Speicherelementen gemäß Bild 1. Die mit $\delta_i{}^B$ bezeichneten Booleschen Funktionen berechnen aus der Belegung $[y_1, y_2, \ldots, y_n]$ der Speicher (Delay-Flipflops) und $[x_1, \ldots, x_n]$ der Eingangsklemmen im Takt t die Speicherbelegung $[y_1', y_2', \ldots, y_n']$ des Folgetaktes $t + 1$. $\delta_i{}^B$ heißt daher auch Ansteuerungsfunktion des Flipflops i. Die Funktionen $\lambda_j{}^B$ sind über der gleichen Argumentmenge definiert und bestimmen das Ausgangssignal $[z_1, \ldots, z_i, \ldots, z_k]$ des gleichen Taktes.

Abstrahiert man von der Struktur der Eingangs- und Ausgangssignale sowie der Zustände als Boolesche Vektoren, so erhält man den Begriff des abstrakten (Mealy-)Automaten:

$A = [X, Z, Y, \delta, \lambda]$ heißt (endlicher) abstrakter Automat (kurz: AA), falls X, Z und Y nichtleere, endliche Mengen sind und falls das kartesische Produkt $Y \times X$ durch δ in Y und durch λ in Z eindeutig abgebildet wird. X heißt Menge der Eingangs-, Z Menge der Ausgangssignale, und Y ist die Zustandsmenge von A. Die Überführungsfunktion δ ordnet jedem Zustand y und jedem Eingabesignal x eindeutig einen Folgezustand $y' = \delta(y, x)$ zu, den A im nächsten Takt einnimmt. $z = \lambda(y, x)$ ist das durch die Ergebnisfunktion λ bestimmte Ausgangssignal von A. Als Beispiel wird die Schaltung nach Bild 2 betrachtet. Die 4 möglichen Zustände sind [0,0], [0,1], [1,0], [1,1]. Das Verhalten der Schaltung wird durch die Tafeln 1 und 2 beschrieben, aus denen $\delta_1{}^B$, $\delta_2{}^B$ und $\lambda_1{}^B$ ablesbar sind.

Werden den Speicherbelegungen abstrakte Symbole zugewiesen, etwa [0,0]⟶1, [0,1]⟶2, [1,0]⟶3, [1,1]⟶4 (*) und bezeichnet man die Eingangs- und Ausgangssignale mit den ursprünglichen Symbolen, so erhält man den AA

$A = [\{0,1\}, \{0,1\}, \{1, 2, 3, 4\}, \delta, \lambda]$. δ und λ gehen aus den Tafeln 3 und 4 hervor.

[1)] Diesem Beitrag liegt der Vortrag des Autors „Die Verbesserung der feedback-orientierten Codierungsmethode abstrakter Automaten" auf dem 6. Fachkolloquium für Informationstechnik am 17. bis 19. April 1973 in Dresden zugrunde.

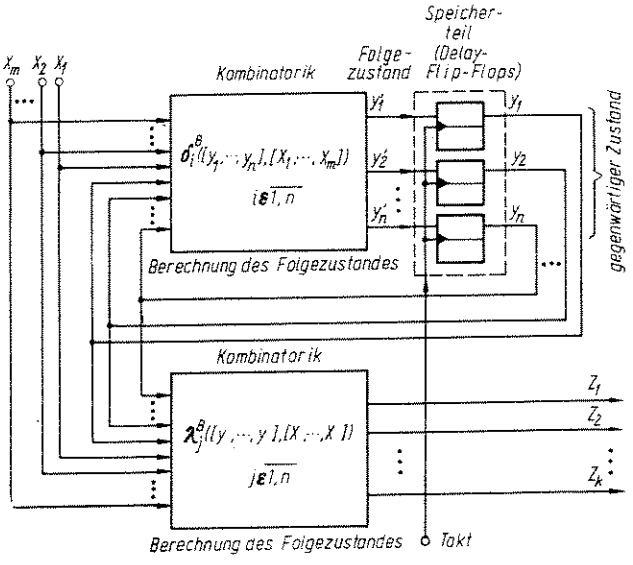

Bild 1. Schema einer digitalen, synchron-getakteten Folgeschaltung

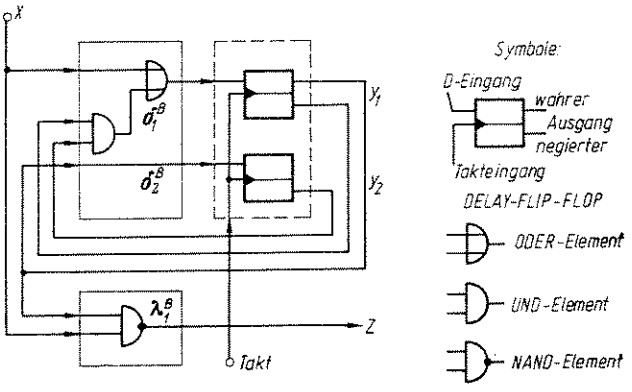

Bild 2. Beispiel einer nach Bild 1 angeordneten Folgeschaltung

Tafel 1. Überführungstabelle

z	[0,0]	[0,1]	[1,0]	[1,1]	gegenwärtige Speicherbelegung
Eingangssignal 0	[1,0]	[0,0]	[0,1]	[0,1]	
Eingangssignal 1	[1,0]	[1,0]	[1,1]	[1,1]	
	Folgebelegung				

Tafel 2. Ergebnistabelle

x	[0,0]	[0,1]	[1,0]	[1,1]
0	0	0	1	1
1	1	1	1	1
	Ausgangssignal			

Jede Speichervariable y_i, $i \in \overline{1, 2}$ bestimmt durch die Symbolzuweisung (*) eine Menge von Teilmengen von Y. Ist beispielsweise $y_1 = 1$, so sind damit die Zustände 3, 4 festgelegt, oder bei $y_2 = 0$ befindet sich die Schaltung in einem der Zustände 1 oder 3. Liegt eine beliebige endliche Menge Y vor, deren Elemente — wie hier im Fall der Zustände — den Elementen einer Menge Y' von Booleschen Vektoren der Form $[y_1, \ldots, y_i, \ldots, y_n]$ eineindeutig zugeordnet sind, so induziert jede Variable y_i eine Zerlegung π_i von Y. Dabei werden alle die y aus Y in einem „Block" von π_i zusammengefaßt, für die y_i den gleichen Wert hat. Ebenso wird durch mehrere Variable $y_{i1}, \ldots, y_{ini}$ eine Zerlegung $\pi_{i1}, \ldots, i_{ni}$ von Y dadurch definiert, daß alle die y aus Y einem gemeinsamen Block angehören, für die die angegebenen Variablen gleiche Werte haben. Im 1. Fall ergeben sich Zerlegungen mit genau 2, im 2. mit 2^{ni} Blöcken. Im Schaltungsbeispiel nach Bild 2 erhält man für die Zustandsmenge $\pi_1 = (1, 2/3, 4)$, $\pi_2 = (1, 3/2, 4)$ sowie $\pi_{1,2} = (1/2/3/4)$.

Mit den Zerlegungen kann man den Begriff der Information verbinden. Das Wissen um den tatsächlich vorliegenden Zustand bei Kenntnis des ihn enthaltenden Zerlegungsblocks ist um so genauer, je kleiner die Blöcke dieser Zerlegung sind.

Es erweist sich, daß sich die Abhängigkeiten der Speicher voneinander in bestimmten Eigenschaften der entsprechenden Zerlegungen widerspiegeln. Bei der Synthese digitaler Schaltungen mit den Mitteln der Automatentheorie versucht man umgekehrt, solche Zerlegungen zu finden, bei denen die dazugehörigen Speicher möglichst wenig voneinander abhängen, so daß der kombinatorische Ansteueraufwand minimal wird.

2. Grundlagen der algebraischen Strukturtheorie abstrakter Automaten von *Hartmanis* und *Stearns*

Zu Beginn der 60er Jahre entwickelten *J. Hartmanis* und *R. Stearns* [7] die algebraische Strukturtheorie der endlichen Automaten, die für eine ganze Reihe von Codierungsverfahren die mathematische Grundlage liefert. Nach einem kurzen Einblick in diese Theorie wird, darauf aufbauend, ein feedback-orientierter Codierungsalgorithmus formuliert (Abschnitt 5).

Eine Zerlegung der Menge Y ist eine Menge π von Teilmengen von Y mit $\cup \pi = Y$, wobei für 2 beliebige Blöcke B, B' von π $B \cap B' = \varnothing$ gilt, sobald $B \neq B'$ ist. Zerlegungen der Zustandsmenge eines AA sollen hier auch Partitionen genannt werden. In der Menge $\underline{M}$ aller Partitionen über einer fixierten Menge Y kann man eine Ordnungsrelation einführen: Die Partition π heißt höchstens feiner als τ ($\pi \leqq \tau$), wenn jeder π-Block in einem τ-Block enthalten ist; π heißt (echt) feiner als τ ($\pi < \tau$), wenn $\pi \leqq \tau$ und $\pi \neq \tau$. Mit dieser Relation werden die Operationen $\cdot$ und $+$ definiert: $\pi \cdot \tau$ ist die größte Partition ϱ, für die $\varrho \leqq \pi$ und $\varrho \leqq \tau$ gilt, d.h. die Menge aller nichtleeren Durchschnitte von π-Blöcken und τ-Blöcken. $\pi + \tau$ ist das kleinste ϱ mit $\varrho \geqq \pi$ und $\varrho \geqq \tau$. M bildet mit den eingeführten Operationen bzw. mit der Relation $\leqq$ einen Verband, dessen kleinstes (feinstes) Element, die Zerlegung in Einermengen, mit 0 bezeichnet wird. Im folgenden wird auf Automaten der Form $A = [X, Y, \delta]$ beschränkt, d.h., es interessiert nur das Zustandsverhalten.

π und τ seien 2 Partitionen von Y mit der Eigenschaft, daß für jedes x aus X und für jeden Block B aus π die Menge $\delta(B, x) = \{\delta(z, x) | z \in B\}$ ganz in einem τ-Block liegt.

Dann heißt $[\pi, \tau]$ Partitionspaar (PP) für A. Als Beispiel wird der AA von Abschn. 1 mit $\pi = (1, 2/3, 4)$ und $\tau = (1, 3/2,4)$ betrachtet. Aus der Kenntnis des π-Blockes, in dem sich der gegenwärtige Zustand befindet, kann der τ-Block ermittelt werden, der den Folgezustand enthält. Beispielsweise gilt $\delta(\{1, 2\},0) = \{1, 3\} \in \tau$. Dabei ist überdies der τ-Block unabhängig vom Eingangssignal x aus X bestimmbar.

Mit Partitionspaaren lassen sich solche Codierungen finden, für die die Abhängigkeit eines Speichers von anderen reduziert ist. Zur Berechnung des Folgeblocks von τ benötigt man im ebengenannten Beispiel nur die Information darüber, in welchem Block von π sich der gegenwärtige Zustand befindet, ohne daß man ihn genau kennen muß.

Ist $[\pi, \tau]$ ein PP und $\tau \leqq \tau'$, so ist auch $[\pi, \tau']$ ein PP. Das feinste τ mit dieser Eigenschaft heißt $m(\pi)$. Umgekehrt ist auch $[\pi', \tau]$ ein PP, sobald $\pi' \leqq \pi$ und $[\pi, \tau]$ PP ist. Das gröbste π mit dieser Eigenschaft heißt $M(\tau)$. Die Operatoren m und M sind monoton, d.h., aus $\pi \geqq \pi'$ folgt $m(\pi) \geqq m(\pi')$ und $M(\pi) \geqq M(\pi')$.

Das Codierungsproblem für einen AA besteht darin, jeden Zustand y durch mindestens einen Booleschen Vektor $[y_1, \ldots, y_n]$ mit $n \geqq$ entier ld (Anzahl (Y)] zu codieren, wobei die Kompliziertheit der sich daraus und aus δ ergebenden Booleschen Funktionen δ_i^B ($i \in \overline{1, n}$) möglichst klein sein soll. Unter Berücksichtigung des im Abschn. 1. erläuterten Zusammenhangs zwischen Codierung und Partitionen kann diese Aufgabe angenähert folgendermaßen formuliert werden: Man finde eine möglichst kleine Menge

$$\Theta = \{\tau_1, \ldots, \tau_n\} \text{ mit } n \geqq \text{ld [Anzahl } (Y)]$$

von Zwei-Block-Partitionen derart, daß $\prod_{i=1}^{n} \tau_i = 0$ und daß für alle $i \in \overline{1, n}$ die $M(\tau_i)$ so groß sind, daß die Mengen $\Theta_i \subseteq \Theta$ mit $\varrho_i = \prod_{\tau \in \Theta_i} \tau \leqq M(\tau_i)$ eine minimale Anzahl von Partitionen enthalten. Die τ_j, $i \in \overline{1, n}$ heißen Codierungspartitionen. In der Literatur werden vielfältige Algorithmen angeboten, die für einen gegebenen AA eine derartige, mehr oder weniger optimale Menge $\{[\varrho_1, \tau_1], \ldots, [\varrho_n, \tau_n]\}$ von Partitionspaaren liefern. In den folgenden Abschnitten wird

Tafel 5. Schema für die Ansteuerung des ersten Speichers

x	Block in 0-Partition	Codierung gemäß τ_1 und τ_2	Block in τ_1 mit Folgezustand	Codierung
0	1	0 0	3,4	1
	2	0 1	1,2	0
	3	1 0	1,2	0
	4	1 1	1,2	0
1	1	0 0	3,4	1
	2	0 1	3,4	1
	3	1 0	3,4	1
	4	1 1	3,4	1

Tafel 3

δ	1	2	3	4
0	3	1	2	2
1	3	3	4	4

Tafel 4

z	1	2	3	4
0	0	0	1	1
1	1	1	1	1

ein Verfahren zur Codierung eines AA beschrieben, bei dem man, von einer vorgegebenen Partition π ausgehend, die Codierungspartitionen bestimmt.
Als Beispiel soll abschließend der AA vom Abschnitt 1. (rück-) codiert werden. Wie schon bemerkt, ist mit $(\pi =)$ $\tau_1 = (1, 2/3, 4)$ und $(\tau =) \tau_2 = (1, 3/2, 4)$ $[\tau_1, \tau_2]$ ein PP, d.h. $M(\tau_2) \geqq \tau_1$. Ferner gilt $M(\tau_1) = (1/2, 3, 4)$; so daß $\Theta_1 = \{\tau_1, \tau_2\}$ mit $\tau_1 \cdot \tau_2 = \varrho_1 = 0$. Für die Codierung von A erhält man somit $\{[0, \tau_1], [\tau_1, \tau_2]\}$. Den ersten Blöcken soll bei der Zuweisung der Zustandsvariablen stets der Wert 0, den zweiten 1 (L) entsprechen. Für das PP $[0, \tau_1]$, d.h. für die Ansteuerung des ersten Speichers, ergibt sich ein Schema nach Tafel 5, oder $y_1 = x \vee \bar{y}_1 \bar{y}_2$. Für den zweiten Speicher erhält man $y_2 = y_1$.

Eingegangen am 5. November 1973

NaA 7254
(wird fortgesetzt)

Ein Verfahren zur modularen Realisierung von Automaten

J. Hoppe, Karl-Marx-Stadt

Deskriptoren: **Automatentheorie, rechnergestützter Schaltungsentwurf;** Automat; Eingangssignal; Ausgangssignal; Folgesteuerung; Verfahren, Ausführung; Modul; Mikroelektronik; Gruppenintegration; Ökonomie; Entwicklungstendenz

Die Entwicklung der elektronischen Rechen- und Steuerungstechnik führt zu Anlagen, bei denen Arbeitsgeschwindigkeit, Leistungsfähigkeit und Einsatzmöglichkeiten ständig steigen. Diese Parameter bewirken, daß immer mehr und immer schnellere elektronische Bauelemente benötigt werden. Da dem Preis und dem Platzbedarf der Anlagen von vornherein Grenzen gesetzt sind, werden in den Rechnern der 3. Generation die logischen Funktionen in steigendem Maße durch den Einsatz von integrierten Halbleiterschaltungen realisiert. Diese Schaltungen sind gegenüber solchen aus diskreten Bauelementen wesentlich kleiner. Sie schaffen damit eine notwendige Voraussetzung für die hohen Arbeitsgeschwindigkeiten, und sie sind preisgünstiger.
Während zunächst nur integrierte Schaltungen vorhanden waren, die einzelne Gatterfunktionen (NAND, NOR) realisieren, sind inzwischen im internationalen Maßstab wesentlich umfangreichere Schaltkreise im Angebot, z.B. Zähler, Multiplexer, Speicherelemente. Allen diesen Schaltungen ist gemeinsam, daß sie aus rein systemfunktionellen Aufspaltungen von Rechenanlagen hervorgegangen sind. Als Schaltkreis, der sich zur Realisierung von beliebigen logischen Funktionen eignet, wurde bisher im wesentlichen nur der Multiplexer propagiert [1] [5] [10], während über Angebote oder in nächster Zeit realisierungsfähige Vorschläge für Schaltkreise zur modularen Realisierung von kompletten Folgesteuerungen (endlichen Automaten) so gut wie nichts bekannt ist. Auch die auf dem Gebiet der Theorie der endlichen Automaten durchgeführten Untersuchungen zur modularen Realisierung führten bisher nicht zu Ergebnissen, die sich bereits praktisch durchgesetzt hätten.
In dieser Arbeit soll der Versuch unternommen werden, auf der Basis vorhandener bzw. mit der gegenwärtigen Technologie der Bauelementeindustrie in naher Zukunft herstellbarer integrierter Schaltkreise ein Verfahren zur modularen Realisierung von Automaten anzugeben. Dabei wird — ausgehend von notwendigen Grundlagen und bereits bekannten Ergebnissen — zunächst das Realisierungsprinzip dargelegt. Danach wird der Nachweis der Anwendbarkeit dieses Prinzips für Automaten beliebiger Struktur und Größe geführt. In einer späteren Arbeit sollen Verfahren skizziert werden, mit denen die Codierung der verschiedenen Alphabete des Automaten auf relativ einfache Art erfolgt, so daß ein nahezu minimaler Aufwand an Modulen benötigt wird.

1. Grundlagen

Zur Problematik der modularen Zerlegung von Automaten existiert eine Vielzahl von Arbeiten. *Hartmanis* und *Stearns* beschäftigten sich mit der Zerlegung von Automaten durch Partitionen mit Substitutionseigenschaft und durch Partitions-Paare [2]. Diese Methoden führten jedoch nicht zu einem für beliebige Automaten stets hinreichenden Satz von Modulen. Außerdem sind sie nicht stets einsetzbar, weil die genannten Partitionen nicht bei jedem Automaten in der für eine Zerlegung notwendigen Weise existieren.
Eine auf der Theorie von *Krohn* und *Rhodes* [7] basierende Zerlegungsmethode für beliebige Automaten wurde von *Zeiger* [11] angegeben. Sie führt zu Kaskadenschaltungen von Rücksetzautomaten mit 2 Zuständen und von Permutationsautomaten, deren Gruppen einfach und homomorphes Abbild einer Unterhalbgruppe des zu zerlegenden Automaten sind. Die Methode basiert auf der Anwendung von Hüllen (Überdeckungen, Mengensysteme) mit Substitutionseigenschaft und ist stets anwendbar. Auf Grund der extremen Erweiterung der Zustandsmenge des Originalautomaten ist, wie auch *Zeiger* in einer neueren Veröffentlichung bestätigt [12], diese Methode jedoch aus Aufwandsgründen praktisch nicht einführbar.
Weiner, Ullman, u.a. [6] [8] [9], empfehlen zur modularen Realisierung von Automaten die Anwendung von AOD-Modulen (And-Or-Delay-Modul, Bild 1). Diese können betrachtet werden als Zusammensetzung aus einem Multiplexer mit t Selektoreingängen $U_0, \ldots, U_{t-1}$ und $g = 2^t$ Dateneingängen $T_0, \ldots, T_{g-1}$ (gestrichelt umrahmt) sowie einem D-Flipflop. Der vorgeschlagene Algorithmus zur Anwendung dieser Module geht aus von Moore-Automaten mit einem Ausgangsalphabet $Z = \{z_0, \ldots, z_{p-1}\}$, das durch q Ausgangssignale $w_0, \ldots, w_{q-1}$ codiert ist. Jedem Ausgangssignal w_k wird zunächst ein Modul zugeordnet, der in genau jenen Zuständen am Ausgang ein L erzeugt, in denen das für w_k gefordert ist. Er realisiert damit die Partition P_k auf die Zustandsmenge $S = \{s_0, \ldots, s_{n-1}\}$. An die Eingänge $U_0, \ldots, U_{t-1}$ jedes Moduls werden die Eingangssignale $p_0, \ldots, p_{t-1}$, die das Eingangsalphabet $X = \{x_0, \ldots, x_{m-1}\}$ des Automaten darstellen, angeschaltet. Jeder Modul muß daher für $t + 2^t$ Eingänge eingerichtet sein. An jeden AOD-Modul, der eine Partition P_k realisiert, ist an den Eingängen $T_0, \ldots, T_{g-1}$, die durch eines der Eingangssymbole x_j aktiviert werden, die Partition $P_k x_j$ anzuschalten. Das ist die Partition auf die Vorgängerzustände von P_k unter dem Eingangssymbol x_j. Jeder derartigen Vorgängerpartition ist ein weiterer Modul zuzuordnen, sofern sie nicht bereits realisiert wurde oder trivial ist. Das führt im Maximalfall zu

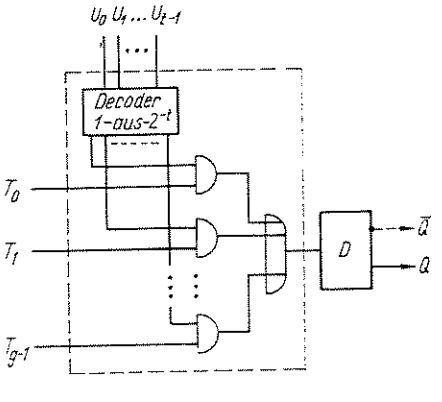

Bild 1
AOD-Modul

Abstrakte Automaten, algebraische Feedback-Theorie und Entwurf digitaler Folgeschaltungen. Teil 2

K.-A. Zech, KDT, Berlin

Mitteilung aus dem Institut für Nachrichtentechnik Berlin

3. Die Theorie der Feedback-Information in Folgeschaltungen

Es sei folgendes angenommen: Am „Eingang" einer durch die Tabellen eines AA beschriebenen Schaltung S stehen die externe Eingangsinformation (Eingangssignale) und über den gegenwärtigen Schaltungszustand (Speicherbelegung) lediglich die Kenntnis des ihn enthaltenden Blocks B einer Partition π zur Verfügung (Bild 3*a*). Über den Folgezustand von S ist also an dieser Stelle gemäß Abschnitt 2. nur bekannt, in welchem Block B' aus $m(\pi)$ er sich befinden wird, wobei $\delta(B, x) \subseteq B'$. Man setzt $\tau_1 = m(\pi)$ und betrachtet τ_1 als Codierungspartition. Da diese im allgemeinen mehr als 2 Blöcke hat, wird sie in mehrere 2-Block-Partitionen $\tau_{1_1}, \ldots, \tau_{1_{n1}}$ mit $\tau_{1_1} \cdots \tau_{1_{n1}} = \tau_1$ zerlegt. Die 2-Block-Partitionen werden durch die Binärspeicher (D-Flipflops) $F_{1_1}, \ldots, F_{1_{n1}}$ realisiert (Bild 3*b*). Nun verfeinert sich die Kenntnis π über den gegenwärtigen Zustand am Eingang von S zu $\pi \cdot m(\pi)$. Daraus kann für den Folgezustand ($\pi \cdot m(\pi)$) abgeleitet werden. Da bereits

$$\tau_1 \geqq m[\pi \cdot m(\pi)]$$

fixiert ist, muß ein τ_2 mit $\tau_1 \cdot \tau_2 = m[\pi \cdot m(\pi)]$ aufgesucht werden. Dieses ist nicht eindeutig bestimmt, und seine Auswahl hat wesentlichen Einfluß auf den kombinatorischen Aufwand zur Ansteuerung der zu τ_2 gehörigen Flipflops. Aus π, τ_1 und τ_2 läßt sich nun $m\{\pi \cdot m[\pi \cdot m(\pi)]\}$ berechnen usw. Bei Fortsetzung dieses Verfahrens ergibt sich eine Schaltungsrealisierung gemäß Bild 3*c*). π heißt rückgekoppelte Information oder Feedback-Partition.

Zur Illustration dieser schrittweisen Ermittlung der Schaltung soll noch einmal der AA von Abschn. 1. betrachtet werden. Als Feedback-Partition sei $\pi = (1{,}2/3{,}4)$ mit der Codierung (0/1) vorgegeben. Mit $m(\pi) = (1{,}3/2{,}4)$ liegen die Codierungspartition τ_1 der „1. Stufe" [Codierung: (0/1)] und die Gesamtinformation $\pi \cdot m(\pi) = (1/2/3/4) = 0$ vor. Wegen $m(0) = 0$ eignet sich für τ_2 jede Partition mit $\tau_1 \cdot \tau_2 = 0$. Eine solche ist π selbst. Daher kann man $\tau_2 = \pi$ setzen und erhält die gleiche Schaltung wie nach Bild 2, jedoch formal in anderer Anordnung (Bild 4).

Nicht für jede Partition π ist eine derartige Realisierung möglich. Im folgenden werden die Bedingungen angegeben, unter denen π als rückgekoppelte Information ausreicht, um eine vollständige Codierung zu ermöglichen.

Für eine beliebige Partition π wird abkürzend definiert:

$$A^1(\pi) = \pi,\ A^{i+1}(\pi) = \pi \cdot m[A^i(\pi)],\ i \geqq 1.$$

$A^i(\pi)$ ist die am Eingang der i-ten Stufe (Bild 3*c*) zur Verfügung stehende Information, woraus sich $m[A^i(\pi)]$ und damit ein τ_i mit $m[A^i(\pi)] = \tau_i \cdot m[A^{i-1}(\pi)]$ berechnen läßt. ($m[A^{i-1}(\pi)]$ ist die bis zur i-1-ten Stufe fixierte Information.)

Es gelten folgende Sätze:

1. Für $i \geqq j$ ist $A^i(\pi) \leqq A^j(\pi)$, daher $m[A^i(\pi)] \leqq m[A^j(\pi)]$
2. Ist n die Zahl der Zustände des vorliegenden AA und ist $j \geqq n$, so gilt $A^j(\pi) = A^n(\pi)$
3. Für $i > j$ und $A^j(\pi) = A^{j+1}(\pi)$ gilt $A^j(\pi) = A^i(\pi)$.

Eine Codierung ist dann vollständig und eindeutig, wenn das Produkt der Codierungspartitionen $\tau_1, \ldots, \tau_i, \ldots = 0$ ist. Wegen $m[A^i(\pi)] = \tau_1 \cdots \tau_i$ muß für eine Feedback-Partition $m[A^{n-1}(\pi)] = 0$ gelten, da auf Grund von (2) und (3) für $i > n-1$ keine Verfeinerung durch weitere Stufen stattfinden kann. In [8] und [10] werden Verfahren angegeben, nach dem sich für einen vorgegebenen AA Feedback-Partitionen finden lassen.

4. Verringerung der Speicherzahl und des kombinatorischen Ansteueraufwands bei Anwendung des Feedback-Formalismus

Für den AA aus Abschnitt 2. und für die Partition $\varrho = (1/2, 4/3)$ ist $\alpha = m(\varrho) = (1{,}2/3{,}4)$. Umgekehrt gilt $\beta = M(\alpha) = (1/2{,}3{,}4) > \varrho$. Zur „Berechnung" von α genügt also $\beta > \varrho$. Es gilt allgemein $Mm(\varrho) \geqq \varrho$ für beliebiges ϱ. Bei der Anwendung des Feedback-Formalismus zur Codierung eines AA (Abschnitt 3.) wird bei jeder Stufe $i \geqq 1$ die Partition $m(\pi \cdot \tau_1 \cdot \ldots \cdot \tau_{i-1})$ gebildet, für die ein τ_i mit

$$\tau_i \cdot \tau_1 \cdot \ldots \cdot \tau_{i-1} = m(\pi \cdot \tau_1 \cdot \ldots \cdot \tau_{i-1})$$

aufzusuchen ist. Das ebengenannte Beispiel lehrt, daß wegen

$$Mm[A^i(\pi)] \geqq A^i(\pi) = \pi \cdot \tau_1 \cdot \ldots \cdot \tau_{i-1} \text{ und}$$
$$m[Mm[A^i(\pi)]] = m[A^i(\pi)]$$

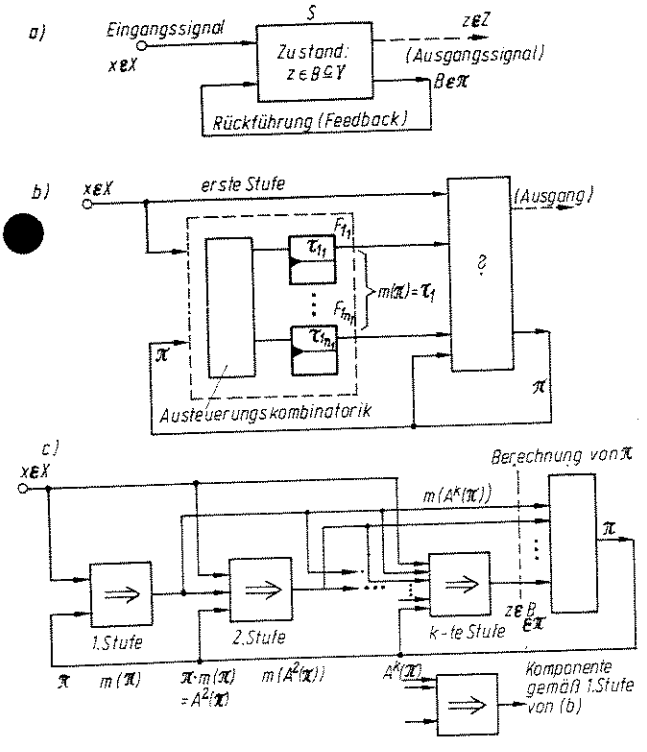

Bild 3. Schrittweise Ermittlung der binären Schaltung bei Vorgabe der Feedback-Partition π:
a) Ausgangspunkt; *b*) die erste Stufe fixiert die aus π und x berechenbare Information über den Folgezustand; *c*) das vollständige Feedback-Schema (ohne Ausgangszuordner)

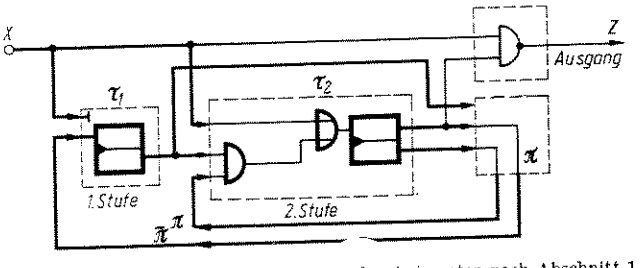

Bild 4. Feedback-orientierte Darstellung des Automaten nach Abschnitt 1.

die Partitionen τ_j ($j \in \overline{1, i-1}$) eventuell durch gröbere τ_j' ersetzt werden können. Dabei muß gelten

$$Mm\,[A^i(\pi)] \geqq \pi \cdot \tau_1' \cdots \tau_{i-1}' \geqq A^i(\pi) = \\ = \pi \cdot \tau_1 \cdot \ldots \cdot \tau_{i-1}.$$

Diese zusätzlichen Freiheiten bei der Wahl der τ_i können dafür benutzt werden, die Zahl der Flipflops einer Stufe zu verringern (es sind weniger 2-Block-Partitionen zur Realisierung eines τ_i notwendig) und ihren Ansteuerungsaufwand herabsetzen. Dazu wird jedes τ_i so festgelegt, daß möglichst wenige τ_j^i aus $\{\pi, \tau_1, \ldots, \tau_{i-1}\}$ zur Berechnung der τ_i-Blöcke ausreichen, d.h., daß $m(\pi \cdot \tau_1^i \cdot \ldots \cdot \tau_j^i \cdot \ldots) \leqq \tau_i$.

Für die feedback-orientierte Codierung erhält man folgenden verbesserten Algorithmus:

1. 1. Stufe: Man berechne $m(\pi)$ und $m[\pi \cdot m(\pi)] = m[A^2(\pi)]$ (bereits 2. Stufe) sowie $Mm[A^2(\pi)]$ $[\geqq \pi \cdot m(\pi)]$.
   Da π als bereits codiert vorausgesetzt wird, sind für π m [$m \geqq$ ld (Zahl der π-Blöcke)] 2-Block-Partitionen $\pi_1, \ldots, \pi_m$ fixiert. Man wähle τ_1 so, daß für eine möglichst kleine Teilmenge P^1 von $\{\pi_1, \ldots, \pi_m\}$ bei $\pi \cdot \tau_1 \leqq Mm[A^2(\pi)]$ gilt $m\left(\prod_{\pi^* \in P^1} \pi^*\right) \leqq \tau_1$. Wegen $m(\pi \cdot \tau_1) = m[A^2(\pi)]$ realisiert die 1. Stufe genügend „Information" für die folgenden Stufen, selbst wenn $\tau_1 > m(\pi)$.
   Ist τ_1 nicht schon 2-Block-Partition, so bestimme man die 2-Block-Partitionen $\tau_{11}, \ldots, \tau_{1n_1}$ mit $\prod_{j=1}^{n_1} \tau_{1j} = \tau_1$ so, daß für $1 \leqq j \leqq n_1$ die Mengen $P_j^1 \subseteq P^1$ mit $m\left(\prod_{\pi^* \in P_j^1} \pi^*\right) \leqq \tau_{1j}$ möglichst klein sind. Das sichert, daß die den τ_{1j} entsprechenden Flipflops von einer Minimalzahl von π-Variablen abhängen, so daß der Ansteuerungsaufwand (quasi-)minimal wird. Stets ist $[\pi, \tau_{1j}]$ ein PP.
2. Man berechnet $A^3(\pi) = \pi \cdot m[A^2(\pi)]$ sowie $m[A^3(\pi)]$ und $Mm[A^3(\pi)] \geqq A^3(\pi)$ und wähle τ_2 so, daß für eine möglichst kleine Teilmenge

   $$P^2 \subseteq \{\pi_1, \ldots, \pi_m, \tau_{11}, \ldots, \tau_{1n_1}\}\; m\left(\prod_{\tau^* \in P^2} \tau^*\right) \leqq \tau_2$$

   ist, wobei $\pi \cdot \tau_1 \cdot \tau_2 \leqq Mm[A^3(\pi)]$. Hat τ_2 mehr als 2 Blöcke, so bestimme man die 2-Block-Partitionen $\tau_{21}, \ldots, \tau_{2n_2}$ mit $\prod_{j=1}^{n_2} \tau_{2j} = \tau_2$ derart, daß möglichst kleine Mengen P_j^2 existieren mit

   $$m\left(\prod_{\tau^* \in P_j^2} \tau^*\right) \leqq \tau_{2j}.$$
3. Man wiederhole Schritt 2 für $\tau_3, \tau_4, \ldots, \tau_{k-1}$ bis $m[A^k(\pi)] = 0$ ist.
4. Man bilde τ_k so, daß $\tau_1 \cdot \ldots \cdot \tau_{k-1} \cdot \tau_k = 0$, wobei $P^k \subseteq \{\pi_1, \ldots, \pi_m, \tau_{11}, \ldots, \tau_{1n_1}, \ldots, \tau_{k-1_{n_{k-1}}}\}$ mit $m\left(\prod_{\tau^* \in P^k} \tau^*\right) \leqq \tau_k$ ebenso minimal sein soll wie die zu den 2-Block-Partitionen τ_{kj} für τ_k gehörenden Mengen P_j^k.
5. Man codiere alle τ_{ij} mit (0/1) und bestimme für jedes τ_{ij} die Funktionen δ_{ij}, die jedem Block B_1 von $\prod_{\tau^* \in P_j^i} \tau^*$ und jedem Eingangssignal x den Block B_2 aus τ_{ij} zuweist mit $\delta(B_1, x) \subseteq B_2$.
   Man ermittle aus den δ_{ij} die Booleschen Ansteuerungsfunktionen δ_{ij}^B und minimiere sie.

In [9] wurde eine Divisionsoperation für Partitionen eingeführt, die es gestattet, diesen Algorithmus noch weiter zu verbessern. Man bestimmt nicht nur

$$Mm[A^j(\pi)] \geqq \pi \cdot \tau_1 \cdot \ldots \cdot \tau_{j-1}$$

mit

$$\frac{Mm[A^j(\pi)]}{\pi \cdot \tau_1 \cdot \ldots \cdot \tau_{j-2}} \geqq \tau_{j-1}$$

sondern berücksichtigt die Beziehung

$$\frac{Mm[A^j(\pi)]}{\pi \cdot \tau_1 \cdot \ldots \cdot \tau_{j-2}} \leqq \frac{M\left(\frac{Mm[A^{j+1}(\pi)]}{\pi}\right)}{\pi \cdot \tau_1 \cdot \ldots \cdot \tau_{j-2}} \leqq \cdots \leqq \beta \geqq \tau_{j-1}.$$

Damit liegt ein Instrument vor, um für jede Stufe i die maximalen „Toleranzen" $m[A^i(\pi)] \leqq \tau_i \leqq \beta$ zu bestimmen, bei denen die Zahl der Stufen fest bleibt.

In der gleichen Arbeit wird eingehender untersucht, wie diese Freiheiten für die Festlegung der τ_i zu nutzen sind, um den Aufwand für die Flipflop-Ansteuerungen niedrig zu halten. Zugrunde gelegt werden dort Überdeckungen, eine Verallgemeinerung des Partitionsbegriffs. Bei Überdeckungen wurde die Forderung der paarweisen Durchschnittsfremdheit der Blöcke fallengelassen und dadurch ersetzt, daß kein Block in einem anderen enthalten ist. Die Operationen m und M, die Multiplikation und der Begriff Überdekkungspaar werden analog definiert. Ein Beispiel für eine Überdeckung ist $\Phi = (1,2/2,3/4)$ für den AA aus Abschnitt 1.

5. Beispiel und Diskussion

Betrachtet wird der Automat $A = [\{0,1\}, \{0,1\}, \{1, 2, 3, \ldots, 16\}, \delta, \lambda]$ mit Überführungs und Ergebnistabelle nach Tafel 6. Es ist eine aufwandsarme feedback-orientierte Codierung über die Feedback-Partition $\pi = (1, 2, 5-16/3,4)$ zu konstruieren. Der vorliegende AA hat die Besonderheit, daß die Zustände 3 bis 9 „nicht erreichbar" sind, d.h., sie können nicht als Folgezustände auftreten. Die gesuchten Partitionen lauten:

$$A^1(\pi) = \pi = (1, 2, 5-16/3,4)$$

$$m(\pi) = (1/2/10-16/\underbrace{3/4/5/6/7/8/9}_{\text{nicht erreichbare Zustände}})$$

$$A^2(\pi) = \pi \cdot m(\pi) = m(\pi), \quad \text{da} \quad \pi > m(\pi)$$

$m[A^2(\pi)] = (1/2/10/11/12/13-16/$ nicht erreichbare Zustände)

$$A^3(\pi) = m[A^2(\pi)]$$

$m[A^3(\pi)] = (1/2/10/11/12/13/14/15, 16/$ nicht erreichbare Zustände)

$$A^4(\pi) = m[A^3(\pi)]$$

$$m[A^4(\pi)] = 0.$$

Die einzelnen nicht erreichbaren Zustände können ohne Veränderung der Ebenenzahl auf die übrigen Blöcke aufgeteilt werden. Das wird zweckmäßigerweise so geschehen, daß die Verfeinerung der m-Partitionen von Schritt zu Schritt maximal ist. Ohne Anwendung der im Abschnitt 4. beschriebenen Modifikation des Algorithmus wären für die 1. Stufe mindestens 3 Blöcke zu realisieren, was 2 Flipflops entspricht. Die Partition τ_2, die dann τ_1 zu $\tau_1 \cdot \tau_2 = m[A^2(\pi)]$ (bis auf die nicht erreichbaren Zustände!) verfeinern müßte, enthält 4 Blöcke, denn /10−16/ wird zerlegt in

Tafel 6. Überführungstabelle δ und Ergebnistabelle λ zum Beispiel

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
0	10	10	2	2	12	12	12	12	14	14	14	14	16	16	16	16	Überführungs-
1	10	10	1	1	12	12	11	11	14	14	13	13	16	16	15	15	tabelle δ
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	Ergebnis-
1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	Tabelle λ

/10/11/12/13−16/. Das entspricht weiteren 2 Speichern. Insgesamt erforderte der Speicherteil einen Aufwand von 7 Flipflops.

Bei Anwendung der im Abschnitt 4. gezeigten Möglichkeit sind die entsprechenden M-Partitionen und daraus die Codierungspartitionen zu ermitteln: $M\{m[A^2(\pi)]\} = (1,2/3,4/5,6/7,8/9-16)$: Wären keine unerreichbaren Zustände vorhanden, so müßte τ_1 derart bestimmt werden, daß $\tau_1 \cdot \pi \leqq M\{m[A^2(\pi)]\}$ ist mit $\tau_1 \geqq m(\pi)$. Im vorliegenden Fall kann aber der linke Ausdruck der ersten Ungleichung noch mit $\varrho = (1,2,10-16/3/4/5/6/7/8/9)$ multipliziert werden.

$\tau_1 = (1-8/9-16)$ entspricht den Forderungen.

$M\{m[A^3(\pi)]\} = (1,2/3,4/5,6/7,8/9-12/13-16)$; τ_2 ist so zu bestimmen, daß $\tau_2 \geqq m[A^2(\pi)]$ und $\pi \cdot \tau_1 \cdot \varrho \cdot \tau_2 \leqq M\{m[A^3(\pi)]\}$ gelten.

Bei $m(\tau_1) = (1, 2, 10, 11, 12/13-16/$ nicht erreichbare Zustände) wird τ_2 so festgelegt, daß $\tau_2 \geqq m(\tau_1)$: $\tau_2 = (1-4, 9-12/5-8, 13-16)$. $[\tau_1, \tau_2]$ ist ein Partitionspaar, wobei der Folgeblock von τ_2 nicht von x abhängt. Es gilt

$$\tau_1 \cdot \tau_2 \cdot \pi \cdot \varrho = (1, 2/3,4/5/6/7/8/9/10-12/13-16) \leqq M\{m[A^3(\pi)]\}.$$

$M\{[m[A^4(\pi)]\} = M(0) = (1,2/3,4/5,6/7,8/9, 10/11, 12/13, 14/15, 16)$:

für τ_3

Für [illegible] hat $\tau_1 \cdot \tau_2 \cdot \tau_3 \cdot \pi \cdot \varrho = M(0)$ zu gelten.

Es ist $\tau_1 \cdot \tau_2 \cdot \pi \cdot \varrho = (1,2/3,4/5/6/7/8/9/10-12/13-16)$. τ_3 muß 10 von 11, 12 sowie 13, 14 und 15, 16 trennen. Betrachtet man $m(\tau_2) = (1,2, 10, 13, 14/11, 12, 15, 16/$ nicht erreichbare Zustände), so erweist es sich als sinnvoll, die Festlegung $\tau_3 = (1, 2, 5, 6, 9, 10, 13, 14/3, 4, 7, 8, 11, 12, 15, 16)$ zu treffen, wobei $\tau_3 \geqq m(\tau_2)$ erfüllt ist. Die Aufteilung der beliebig zuweisbaren Blöcke (τ_3 ist eine „Vergrößerung" von $\tau_1 \cdot \tau_2 \cdot \tau_3 \cdot \pi \cdot \varrho$) erfolgt dabei so, daß $\tau_1 \cdot \tau_2 \cdot \tau_3$ möglichst gleichgroße Blöcke hat, damit durch τ_4 eine Minimalzahl von Zuständen getrennt werden muß (wegen der Forderung $\tau_1 \cdot \tau_2 \cdot \tau_3 \cdot \tau_4 = 0$). Es ist $\tau_1 \cdot \tau_2 \cdot \tau_3 = (1,2/3,4/5,6/7,8/9, 10/11, 12/13, 14/15, 16)$. τ_4 muß alle diese Zustandspaare trennen. Man setzt $\tau_4 =$ (gerade Zustände/ungerade Zustände) $\geqq$ (2, 10, 12, 14, 16/1, 11, 13, 15 nicht erreichbare Zustände) $= m(\tau_3)$.

Es erweist sich, daß $\pi > \tau_1 \cdot \tau_2 \cdot \tau_3$, so daß man den Block /3,4/ von π unmittelbar gemäß τ_1, τ_2 und τ_3 durch $\overline{0\ 0\ 1}$ codieren kann, während der andere Block mit $\overline{0\ 0\ 1}$ codiert wird. Es folgt die Berechnung der Ansteuerungsfunktionen. Die Codierungspartitionen werden sämtlich durch (0/1) codiert

1. Stufe:

Tafel 7; $y_1 = \overline{y_1\ y_2\ y_3}$;

Tafel 7. Zu Stufe 1: Folgeblock von τ_1

x \ π	[3,4] 0 0 1	Restliche Blöcke $\overline{0\ 0\ 1}$
0	0	1
1	0	1

Tafel 8. Zu Stufe 2

τ_1	0	1
τ_2	0	1

Tafel 9. Zu Stufe 4

x \ $\tau_1 \cdot \tau_2 \cdot \tau_3$	000	001	010	011	100	101	110	111
0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	1

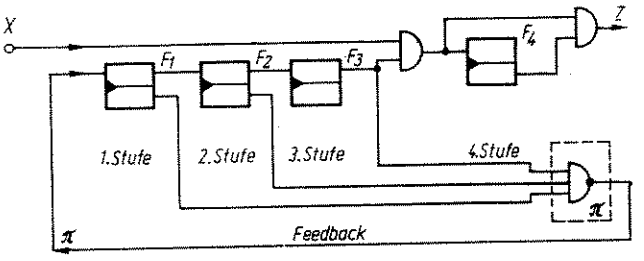

Bild 5. Beispiel für die feedback-orientierte Codierung eines Automaten

2. Stufe:

Wegen $\tau_2 > m(\tau_1)$ hängt τ_2 nur von τ_1 ab, nicht aber von π und — wie schon bemerkt — nicht von x. $y_2 = y_1$; Tafel 8.

3. Stufe:

τ_3 ist ebenfalls unabhängig von x und nur abhängig von τ_2. $y_3 = y_2$.

4. Stufe:

Wegen $\pi > \tau_1 \cdot \tau_2 \cdot \tau_3$ ist $\pi \cdot \tau_1 \cdot \tau_2 \cdot \tau_3 = \tau_1 \cdot \tau_2 \cdot \tau_3$. $y_4 = x\ y_3$; Tafel 9.

Für den Ausgangszuordner ergibt sich $z = \overline{y_4} \vee x\ y_3$ (Bild 5).

Auf die Einsparung von Flipflops bei Benutzung des modifizierten Verfahrens wurde schon hingewiesen. Durch die getroffene Festlegung von τ_2 und τ_3 wird erreicht, daß τ_2 nur von τ_1 und τ_3 nur von τ_2 abhängt, so daß sich ein Schieberegister ergibt.

Bei Vorgabe eines AA ist es im allgemeinen nicht unmittelbar möglich, den Feedback-Algorithmus anzuwenden, da eine Feedback-Partition π in der Regel noch bestimmt werden muß. In [8] und [10] wird ein Verfahren zum Aufsuchen solcher Partitionen angegeben. Die Auswahl einer günstigen Feedback-Partition kann gegenwärtig nur nach Berechnung des Lösungsschemas erfolgen. Die Anwendung des feedback-orientierten Codierungsalgorithmus ist also sinnvoll, wenn entweder π vorgegeben ist oder wenn ein solches π gefunden werden kann, für das die π realisierende Boolesche Funktion relativ unkompliziert ist.

In [9] wird an einem Beispiel gezeigt, daß die Abhängigkeit der Speicher von einer minimalen Anzahl der weiter „links" liegenden Flipflops und von den Feedback-Komponenten noch keine gesamtoptimale Schaltung garantiert. Eine aufwendiger realisierte Stufe kann eine Aufwandsverminderung nachfolgender Stufen bewirken.

Eingegangen am 5. November 1973 NaA 7254

Literatur

[1] *Mealy, G. H.:* A method for synthesizing sequential circuits. Bell. System Techn. J. 34 (1955) H. 5, S. 1045−1079.

[2] *Moore, E. F.:* Gedanken-Experiments on sequential machines. Automata Studies, Princeton University Press, Princeton, N. J. (1956) H. 34, S. 129−153.

[3] *Huffman, D. A.:* The synthesis of sequential switching circuits. J. Franklin Inst., 257 (1954) H. 3/4, S. 161−190 und 275−303.

[4] *Mc Culloch, W. S.*, und *Pitts, W.:* A logical calculus of the ideas immanent in nervous activity. Bull. math. biophys. (1943) H. 5, S. 115−133.

[5] *Kleene, S. C.:* Representation of events in nerve nets and finite automata. Automata Studies, Princeton University Press, Princeton, N. J. (1956) H. 34, S. 3−41.

[6] *Starke, P. H.:* Abstrakte Automaten Berlin: VEB Deutscher Verlag der Wissenschaften, 1969.

[7] *Hartmanis, J.*, und *Stearns, R. E.:* Algebraic structure theory of sequential machines. Pentrice Hall, Englewood Cliffs 1966.

[8] *Dabadgaho, S. V.:* A method for finding feedback partitions for sequential machines. IEEE Trans. Comp., C-18 (1969) H. 5, S. 465−467.

[9] *Zech, K.-A.:* Feedback-orientierte Kodierung endlicher Automaten EIK 9 (1973) H. 10, S. 635−650.

[10] *Zech, K.-A.:* Zum Auffinden von Feedbackpartitionen für endliche Automaten, EIK 10 (1974) H. 8/9, S. 489−494.

[11] *König, C.-D., Ramin, K., Schuppe, W.*, und *Werrmann, G.:* Computergestützter Entwurf digitaler Schaltungen. INT-Interne Information 6 (1971) Sonderheft.

[12] *Gluschkow, W. M.:* Theorie der abstrakten Automaten. Berlin, VEB Deutscher Verlag der Wissenschaften, 1963.