Kleinrock L.,
Sequential Processing Machines (S. P. M.) Analysed with a Queuing Theory Model.
Journal ACM, V. 13, N. 2 (1966), S. 179—193.

Kleinrock L.,
Time-Shared Systems: A theoretical Treatment.
Journal ACM, V. 14, N. 2 (1967), S. 242—261.

Nielsen N. R.,
The Simulation of Time-Sharing Systems.
Comm. ACM, V. 10, N. 7 (1967), S. 397—412.

Scherr A. L.,
An Analysis of Time Shared Computer Systems.
The M. I. T. Press, Cambridge (Mass.) (1967).

Smith J. L.,
An Analysis of Time-Shared Computer Systems Using Markov Models.
SJCC Proceedings, Spartan Books, Washington (1966).

Spies P. P.,
Teilnehmer-Rechensysteme.
Elektron. Rechenanl. 9 (1967), H. 3, S. 117—122.

Spies P. P.,
Zur Simulation von Teilnehmer-Rechensystemen.
Teilnehmer-Rechensysteme (Hsg. *W. Händler*), Oldenbourg Verlag, München (1968).

Spies P. P.,
Rechnerstruktur und Ablaufgeschehen in Rechenanlagen — Analysis und Simulation.
Arbeitsberichte des Instituts für Mathematische Maschinen und Datenverarbeitung, B 2, N 9, Erlangen (1969).

Eine Methode zum Entwurf komplexer Schaltwerke unter Verwendung spezieller Ablaufdiagramme

A method for the design of synchronous digital hardware systems

Elektron. Rechenanl. 12 (1970), H. 6, S. 314—323
Manuskripteingang: 23. 8. 1970

von S. WENDT*),
Department of Electrical Engineering, State University of New York, Buffalo, USA

Es wird eine Methode zum Entwurf digitaler, synchron gesteuerter Hardwaresysteme dargestellt, wobei der Schwerpunkt der Betrachtung auf den zu Beginn eines Entwurfs durchzuführenden Schritten liegt. Das System wird zerlegt in ein „Operationswerk" und ein „Steuerwerk", deren Zusammenwirken mit Hilfe einer hier eingeführten Form von Ablaufdiagramm dargestellt wird. Aus dem Ablaufdiagramm wird die Menge der Steuerzustände abgeleitet. Dadurch, daß innerhalb eines Operationswerks selbst wieder komplexe Schaltwerke vorkommen dürfen, besteht die Möglichkeit, den Entwurf sehr komplexer Systeme in eine Hierarchie einzelner getrennter Entwurfsprozesse zu zerlegen. Die Methode ist gekennzeichnet durch einen hohen Grad an Übersichtlichkeit. Ein Beispiel veranschaulicht die praktische Anwendbarkeit der Methode.

The paper introduces a method for the design of synchronous digital hardware systems. The method is mainly concerned with those steps of a design which are executed first. The system is decomposed into two units, the "operational unit" and the "control unit"; the interaction between these units is described by a special type of flow-chart, from which the set of control states can be derived. Since an operational unit may be composed of other complex digital systems, the design of systems with high complexity can be decomposed into a hierarchy of separated design processes. The main advantage of the method is its clearness and simplicity. An example shows that the method is really applicable for practical design problems.

1. Einleitung

Der logische Entwurf eines digitalen Hardwaresystems, welcher in der vorliegenden Arbeit behandelt wird, besteht darin, von einer Beschreibung der erwünschten, extern meßbaren zeitlich-logischen Eigenschaften des zu entwerfenden Systems und einer Liste der zur Verfügung stehenden Typen logischer Bausteine ausgehend eine Auswahl von Bausteinen und die vollständige Verbindungsstruktur zwischen ihnen zu gewinnen. Dieser logische Entwurf geht in mehreren Abschnitten oder Schritten vor sich, von denen einige, vor allem die zuletzt durchzuführenden, schon seit langem recht gut formalisiert sind, während andere, vor allem die zu Beginn des Entwurfs durchzuführenden, noch wenig formal erfaßt wurden. Eine Formalisierung der zu Beginn eines Entwurfsprozesses durchzuführenden Schritte, welche sich in der Praxis als sehr vorteilhaft erwiesen hat, soll in der vorliegenden Arbeit dargestellt werden. Die Formalisierung geht jedoch nicht so weit, daß nun der ganze Entwurfsprozeß automatisch durchgeführt werden könnte; es bleibt vielmehr ein deutlicher „Spielraum" für den Entwurfsingenieur.

Zur Komplexität der zu entwerfenden digitalen Systeme muß noch folgendes gesagt werden: Der Entwurf, wie er hier betrachtet wird, setzt eine Beschreibung der erwünschten Ei-

*) Ein weiterführender Beitrag des Verfassers zum Thema der Systematik von Mikroprogrammwerk-Strukturen erscheint in 13 (1971), H. 1 dieser Zeitschrift.

genschaften des zu entwerfenden Systems voraus, d. h. die ganze Problematik der Erarbeitung einer solchen Beschreibung bleibt außer acht. Während bekanntermaßen im Falle einer Großrechenanlage die Erarbeitung einer Beschreibung der erwünschten Eigenschaften ein äußerst schwieriger und langwieriger Iterationsprozeß ist, kann eine Beschreibung der erwünschten Eigenschaften beispielsweise eines Tischrechners oder einer Magnetplatten-Schreib-/Lesesteuerung verhältnismäßig leicht gegeben werden. Das soll nicht heißen, daß die im folgenden behandelte Entwurfsmethode auf Großrechner grundsätzlich nicht anwendbar sei, sondern es bedeutet lediglich, daß die Entwurfsmethode für Systeme entwickelt wurde, deren externes Verhalten für den Fachmann „überschaubar" ist. Wenn also, was wünschenswert wäre, der Großrechner als eine Hierarchie überschaubarer Teilsysteme entworfen wird, wobei auch noch die Zusammenarbeit der Teilsysteme innerhalb einer Ebene der Hierarchie überschaubar bleibt, dann ist die gegebene Entwurfsmethode für jedes Teilsystem und jede Ebene der Hierarchie uneingeschränkt anwendbar.

Digitale Hardwaresysteme, deren externes Verhalten für den Fachmann überschaubar ist, deren Komplexität aber deutlich über Zählern und Registern liegt, seien im folgenden *komplexe Schaltwerke* genannt.

2. Zur Einordnung des Verfahrens

Der Gedanke, das zu entwerfende komplexe Schaltwerk in zwei zusammenarbeitende Teilwerke zu zerlegen, nämlich ein „gesteuertes Werk" und ein „steuerndes Werk", entspricht dem Grundgedanken jeglicher Systemtheorie — nicht nur im Digitalbereich. Deshalb kann hier auch niemand als Urheber dieses Gedankens zitiert werden. Aufbauend auf dieser Grundzerlegung wurden eine Vielzahl von Entwurfshilfen und Entwurfsmethoden entwickelt, von denen vor allem die automatentheoretische Mikroprogrammbetrachtung von *Glushkov* [1—3] und die Entwurfsmethode von *Gerace* [4] der vorliegenden Arbeit nahestehen. Übereinstimmend mit der in den genannten Arbeiten verwendeten Bezeichnungsweise seien die beiden Teilwerke im folgenden *Operationswerk* und *Steuerwerk* genannt.

Da der Mikroprogrammbegriff beim Vergleich der vorliegenden mit den genannten Arbeiten wesentliche Bedeutung hat, muß kurz darauf eingegangen werden. Die Arbeitsgruppe „Mikroprogrammtechnik" der ACM (Association for Computing Machinery) hat als eine mögliche Definition des Begriffes Mikroprogrammierung folgendes diskutiert: „Realisierung von Steuerlogik durch geordnete Speicherung von Information" („implementing control logic through the ordered storage of information" [5]). Wenn man „Steuerlogik" mit „Steuerwerk" gleichsetzt, dann kennzeichnet diese Definition genau das, was *Wilkes* [6], [7], als er 1951 den Begriff Mikroprogrammierung einführte, darunter verstand, nämlich eine bestimmte Realisierungsform für das Steuerwerk, bei der ein sog. Mikroprogrammspeicher verwendet wird. Da sich das gleiche externe Verhalten des Steuerwerks erreichen läßt sowohl mit als auch ohne Verwendung eines Mikroprogrammspeichers, ist die Frage, ob Mikroprogrammierung vorhanden sei oder nicht, ohne Berücksichtigung des Operationswerkes entscheidbar.

Glushkov, *Gerace* und viele andere (beispielsweise *Rosin* in [8]) betrachten Mikroprogrammierung losgelöst von der Realisierungsfrage des Steuerwerks als Mittel zur Beschreibung des Zusammenspiels zwischen Operationswerk und Steuerwerk, wobei jedes Ausgabeelement des Steuerwerks bestimmte Elementarveränderungen (Mikroschritte) im Operationswerk bewirkt und der Zustand des Operationswerks die Ausgabefolge des Steuerwerks mehr oder weniger mitbestimmt. In diesem Sinne ist jedes beliebige komplexe Schaltwerk mikroprogrammiert. In der folgenden kurzen Betrachtung der oben genannten Arbeiten sei der Begriff Mikroprogrammierung im letzteren, beschreibenden Sinne gemeint.

Sowohl bei *Glushkov* als auch bei *Gerace* wird das Operationswerk ausschließlich aus Registern und logischen Verknüpfungsnetzen zusammengesetzt, und ein Mikroprogrammschritt besteht aus einer oder mehreren Informationsübertragungen von Register zu Register unter möglicher Transformation durch die verbindenden Verknüpfungsnetze. Verzweigungen im Mikroprogramm ergeben sich durch Abfragen von Registerzuständen. In beiden Arbeiten werden formale Mikroprogrammsprachen eingeführt, für welche dann Programminimisierungsalgorithmen entwickelt werden, welche eine Einsparung von Registerstellen und Verknüpfungsgliedern im Fall *Gerace* und von Grundschritten und damit von Verarbeitungszeit im Fall *Glushkov* bewirken sollen. Die Methode von *Glushkov* ist stark automatentheoretisch orientiert und müßte noch ziemlich modifiziert werden, bevor sie nicht nur für Sonderfälle zum praktischen Entwurf verwendet werden kann. Die Methode von *Gerace* ist praxisnäher; sie wurde zur Entwurfsautomation entwickelt und ist ohne die zugehörigen Computerprogramme nicht anwendbar.

Die in der vorliegenden Arbeit dargestellte Methode unterscheidet sich in zwei wesentlichen Punkten von den oben genannten Verfahren:

(1) Das Operationswerk wird nicht zwangsläufig ausschließlich aus Registern und logischen Verknüpfungsnetzen aufgebaut, sondern kann Teilwerke enthalten, die selbst wieder als komplexe Schaltwerke aufgefaßt und in einem getrennten Entwurfsprozeß entworfen werden können. Im Falle großer Komplexität ergibt sich somit eine Hierarchie von Teilwerken und Entwurfsprozessen.

(2) Es wird keine Mikroprogrammsprache eingeführt, sondern das erwünschte Zusammenspiel zwischen Operationswerk und Steuerwerk wird durch eine hier eingeführte Form von Ablaufdiagramm dargestellt. Ziel bei der Entwicklung der Methode war nicht so sehr die Entwurfsautomation als vielmehr die Einführung eines möglichst hohen Grades an Übersichtlichkeit. Dadurch wird die Überprüfbarkeit eines Entwurfs erleichtert und damit die Fehlerwahrscheinlichkeit herabgesetzt; ferner wird die Kommunikation innerhalb eines Entwurfsteams erleichtert, und schließlich wird die Dokumentation brauchbarer, d. h. leichter lesbar.

3. Entwurf des Operationswerks

3.1 Blockstruktur

Gleich zu Beginn sei darauf hingewiesen, daß keine allgemeingültigen Verfahren zur algorithmischen Durchführung des Entwurfs von Operationswerken bekannt sind. Unter Bezug auf beschreibende Mikroprogramme (s. Abschnitt 2) heißt dies, daß keine allgemeingültigen Verfahren zur Auswahl von Mikrooperatoren (Elementarschritten im Operationswerk) und zum Aufstellen von Mikroprogrammen unter Verwendung dieser Operatoren bekannt sind. Auch *Glushkov* und *Gerace* liefern keine solchen Verfahren.

Am Beispiel der Multiplikation zweier Dualzahlen läßt sich leicht einsehen, wie problematisch die Suche nach einem

Algorithmus ist, der als Ergebnis die „zweckmäßigste" Zerlegung des Multiplikationsprozesses in eine Folge einzelner Schritte liefert. Offensichtlich müssen in die Beurteilung der Zweckmäßigkeit noch Kriterien eingehen, welche mit der mathematischen Definition der Multiplikation nichts zu tun haben; denn wie soll entschieden werden, ob die Multiplikation in einem einzigen Schritt mit Hilfe eines entsprechenden Schaltnetzes oder in mehreren Schritten mittels eines der vielen möglichen Schaltwerke durchgeführt werden soll?

Der erste Abschnitt des Entwurfs eines komplexen Schaltwerks, nämlich der Entwurf des Operationswerkes, bleibt also auch bei der hier dargestellten Methode eine weitgehend intuitive Angelegenheit.

Ausgangspunkt für den Entwurf des Operationswerks ist eine Beschreibung der erwünschten Eigenschaften des zu entwerfenden komplexen Schaltwerks. Diese Aufgabenstellung umfaßt die Menge P der Eingabesignale, die Menge Q der Ausgabesignale und die Funktionsbeschreibung, welche die Abhängigkeit der Ausgabesignale von den Eingabesignalen angibt. Die Elemente der Mengen P und Q sind Vektoren $\boldsymbol{P}$ bzw. $\boldsymbol{Q}$ mit binären Komponenten p_i bzw. q_j, wobei jeder Eingangssignalleitung ein p_i und jeder Ausgangssignalleitung ein q_j entspricht. Für die Funktionsbeschreibung gibt es beliebige Darstellungsformen: Text, Tabellen, Zeitdiagramme, logische Gleichungen mit Zeitbedingungen, Ablaufdiagramme u. ä. m.

Ein Beispiel einer Aufgabenstellung ist im Abschnitt 5.1 dargestellt. (Der Geschlossenheit wegen wurde das Beispiel an den Schluß der Arbeit gesetzt; dennoch aber setzen die einzelnen Abschnitte des Beispiels, auf welche im Lauf der Arbeit verwiesen wird, nur die Kenntnis der bis zum jeweiligen Verweis dargestellten Theorie voraus.)

Wie schon erwähnt, ist durch die Aufgabenstellung die innere Struktur des zu entwerfenden Systems i. a. noch nicht definiert. Es muß aus der i. a. sehr umfangreichen Menge in Frage kommender innerer Strukturen eine bestimmte ausgewählt werden. Die Bewertungskriterien, welche diese Auswahl bestimmen — beispielsweise Marktlage, Abhängigkeit von Zulieferern, Diagnosefreundlichkeit etc. — sind dabei teilweise nichttechnischer Natur. Es ist offensichtlich, daß man dieser Aufgabe nur näherungsweise gerecht werden kann, weil i. a. weder die Menge der Möglichkeiten noch der Einfluß aller Bewertungskriterien überschaut werden können.

Von der Vielzahl der Fragen, welche entschieden werden müssen, seien nur einige wichtige genannt: Welche Teilfunktionen sollen seriell, welche parallel ausgeführt werden? Welche Teilfunktionen sollen in Hardware, welche in Software ausgeführt werden? Aus welchen Blöcken soll das Schaltwerk zusammengesetzt werden? Als *Blöcke* sind dabei diejenigen Teilwerke bezeichnet, deren Entwurf nicht mehr zum betrachteten Entwurfsprozeß gehört. Der Komplexität dieser Blöcke ist dabei keine Grenze gesetzt, d. h. ein Block kann ein einfaches logisches NAND-Glied, aber auch ein eigenständiges komplexes Schaltwerk sein.

Falls ein Block, der bei einem Entwurfsprozeß als gegeben angenommen wird, in Wirklichkeit nicht gegeben ist, so muß dieser als neues komplexes Schaltwerk aufgefaßt und in einem getrennten Entwurfsprozeß entworfen werden.

Die Funktion der Blöcke im Operationswerk kann unter Verwendung anschaulicher Verarbeitungsbegriffe gekennzeichnet werden, die meist auch zur Benennung des betreffenden Blockes herangezogen werden; als Beispiele seien genannt: Schieberegister, Zähler, Decodiernetz, Addierwerk, Speicherblock, Fernschreibmaschine, Kartenleser.

Das Ergebnis des Entwurfs des Operationswerks ist die Beschreibung der Blöcke, die Verbindungsstruktur zwischen den Blöcken und ein Ablaufdiagramm, welches den gewünschten Steuerablauf beschreibt. Dieser Steuerablauf muß beschrieben werden, weil der Entwurf des Operationswerks i. a. manche Steuereingänge y_k bei den Blöcken unversorgt läßt, die später dann vom Steuerwerk versorgt werden müssen. Die Ausnahme, bei der kein Steuerwerk benötigt wird, sei als „autonomes Operationswerk" bezeichnet.

Die Blockstruktur des Operationswerks für das Beispiel aus Abschnitt 5.1 ist in Abschnitt 5.2.1 dargestellt.

3.2 Ablaufdiagramm

Der Steuerablauf wird als *synchron* zu einem Steuertakt angenommen. Diese Beschränkung bedeutet nicht, daß die Abläufe innerhalb der einzelnen Blöcke synchron zu einem gemeinsamen Takt sein müssen; sie bedeutet lediglich, daß sich die Steuersignale y_k nur in den vom Steuertakt definierten Übergangsintervallen ändern können und daß sich die — später noch näher zu definierenden — Eingangssignale x_l in das Steuerwerk in den vom Steuertakt definierten Entscheidungsintervallen nicht ändern dürfen. Alle diejenigen Signale x_l, welche ihrer Herkunft nach asynchron zum Steuertakt sind, müssen also synchronisiert werden. Solche Signale sind entweder Eingabesignale des gesamten komplexen Schaltwerks oder Endesignale asynchron arbeitender Blöcke.

Im Ablaufdiagramm, welches den Steuerablauf für das Operationswerk beschreibt, kommt außer den beiden Vektoren $\boldsymbol{P}$ und $\boldsymbol{Q}$ noch ein Vektor $\boldsymbol{A}$ vor mit binären Komponenten a_m, welcher den für den Steuerablauf relevanten Teil der Zustände in den Blöcken des Operationswerks erfaßt. Die entsprechende Zustandsmenge sei A. Für den Steuerablauf irrelevant sind die internen Zwischenzustände derjenigen Blöcke, welche auf ein Startsignal hin eine innerhalb des Blockes gesteuerte Folge von Zuständen durchlaufen und ein Endesignal abgeben, wenn ein Endzustand erreicht ist.

Das Ablaufdiagramm setzt sich aus *Zuweisungen* und *Verzweigungsabfragen* zusammen. Die einzelnen Zuweisungen und Verzweigungsabfragen seien mit Indizes ν bzw. μ gekennzeichnet. In den Verzweigungsabfragen werden logische Funktionen β_μ der beiden Vektoren $\boldsymbol{P}$ und $\boldsymbol{A}$ abgefragt:

$$\beta_\mu(\boldsymbol{P}, \boldsymbol{A}) = \text{„1" ?} \tag{1}$$

In den Zuweisungen werden den beiden Vektoren $\boldsymbol{Q}$ und $\boldsymbol{A}$ Werte zugewiesen, welche sich als Funktionen ω_ν (Ausgabefunktionen) bzw. δ_ν (Übergangsfunktionen) der beiden Vektoren $\boldsymbol{P}$ und $\boldsymbol{A}$ ergeben:

$$\boldsymbol{Q} := \omega_\nu(\boldsymbol{P}, \boldsymbol{A}) \tag{2}$$

$$\boldsymbol{A} := \delta_\nu(\boldsymbol{P}, \boldsymbol{A}) \tag{3}$$

Gleichung (3) erfaßt die Übergänge der für den Steuerablauf relevanten Zustände in den Blöcken des Operationswerks. Im Falle derjenigen Blöcke, welche auf ein Startsignal hin eine innerhalb des Blockes gesteuerte Folge von Zuständen durchlaufen, kann nur das Startsignal eine im Ablaufdiagramm lokalisierbare Zustandsänderung bewirken; dagegen ist die durch das Endesignal verursachte Zustandsänderung von $\boldsymbol{A}$ nicht im Ablaufdiagramm lokalisierbar, weil das Auftreten des Endesignals nicht durch den Steuerablauf festgelegt ist. Das Auftreten des Endesignals muß also logischerweise durch eine Verzweigungsabfrage erfaßt werden.

Da der Ablauf synchron zum Steuertakt verlaufen soll, finden die Zustandsübergänge nach Gleichung (3) nur in den

vom Takt bestimmten Übergangsintervallen statt, und die Ausgabesignale nach Gleichung (2) sind nur in den dazwischenliegenden Intervallen eindeutig bestimmt. Deshalb werden diese Intervalle zweckmäßigerweise im Diagramm durch unterschiedliche Zuweisungssymbole dargestellt, nämlich die sog. *statische Zuweisung* für die Ausgabefunktion ω_ν und die sog. *dynamische Zuweisung* für die Übergangsfunktion δ_ν. Der Index ν kennzeichnet also jeweils ein Zuweisungspaar, bei dem einer statischen Zuweisung eine dynamische Zuweisung folgt.

Bild 1 zeigt ein Beispiel eines Ablaufdiagramms mit zwei Abfragen und drei Zuweisungspaaren.

Diese Art Ablaufdiagramm erfaßt formal nicht diejenigen Schaltwerke, bei denen die Ausgabesignale kurze Impulse sind, die durch Differentiation von Zustandsänderungen gewonnen werden und deshalb in dynamischen Zuweisungen auftreten müßten; aber da man in diesem Fall stets jedem dynamischen Ausgabevektor einen statischen Ausgabevektor $\boldsymbol{Q}$ umkehrbar eindeutig zuordnen kann, bringt das Ablaufdiagramm im Grunde keine Beschränkung.

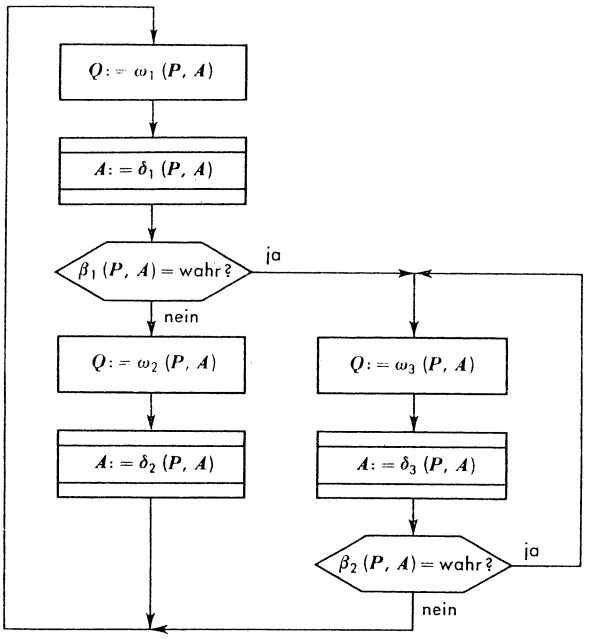


Bild 1. Beispiel eines Ablaufdiagramms.

Die Ablaufdiagramme sind i. a. geschlossen, d. h. sie haben keinen Anfang und kein Ende, sondern bestehen aus einem System von Schleifen (wie in Bild 1). Das läßt sich durch die beiden folgenden Tatsachen begründen:

(1) Der Zustand einer elektronisch realisierten sequentiellen Maschine ist i. a. unmittelbar nach dem Einschalten der Stromversorgung nicht definiert. Um auf einfache Weise einen „Pseudo-Anfangszustand" zu erhalten, fügt man gewöhnlich dem Eingabevektor $\boldsymbol{P}$ eine besondere Komponente bei, das sog. „Grundstellungssignal", welches nach jedem beliebigen Zuweisungspaar wirksam werden kann, d. h. den Ablauf zu einem definierten Zuweisungspaar überführen kann. Dieses Grundstellungssignal müßte deshalb eigentlich im Ablaufdiagramm nach jedem Zuweisungspaar abgefragt werden, was die Übersichtlichkeit des Diagramms stark stören würde; deshalb wird zweckmäßigerweise die Lösung nach Bild 2 eingeführt.

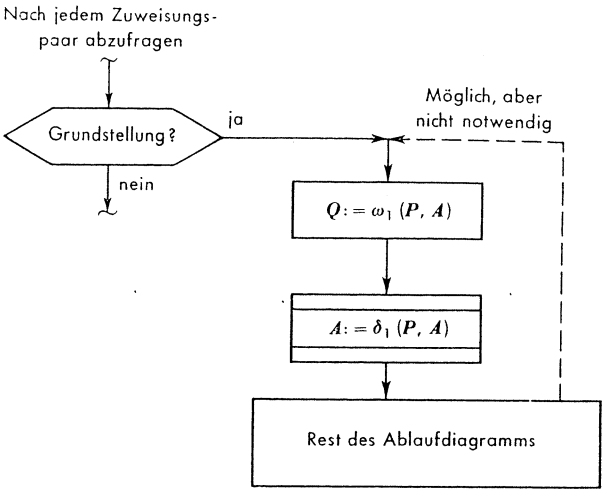


Bild 2. Zur Erfassung der Grundstellung im Ablaufdiagramm.

(2) Der Steuertakt definiert eine quasi unendliche Folge von Übergangsintervallen, so daß es unmöglich ist, einen „letzten" Übergang zu definieren.

Die Funktionen β_μ, ω_ν und δ_ν werden zweckmäßigerweise anschaulich formuliert.

Beispiele für β_μ: Vorzeichen negativ?
Zählerstand = 24?
Endemeldung?

Beispiele für ω_ν: Bereit := „1";
Parity := P (wobei P ein Flipflopausgang im Operationswerk sei);
Summe := $\boldsymbol{R}$ (wobei $\boldsymbol{R}$ ein Registerinhalt im Operationswerk sei);

Beispiele für δ_ν: Register rechts schieben;
$i := i + 1$;
Ausgabewerk starten.

Das Ablaufdiagramm für das Beispiel aus 5.1 und 5.2.1 ist im Abschnitt 5.2.2 dargestellt.

4. Entwurf des Steuerwerks

4.1 Ein- und Ausgabemengen X und Y

Jedes Zuweisungspaar im Ablaufdiagramm besteht aus zwei Funktionen, ω_ν und δ_ν. Es ist nicht notwendig, daß alle Funktionen ω_ν für unterschiedliche Werte von ν verschieden sind oder daß sich alle Funktionen δ_ν unterscheiden. Es gilt lediglich, daß es in nichttrivialen Fällen mindestens *ein* Paar unterschiedlicher Ausgabefunktionen ω_ν oder *ein* Paar unterschiedlicher Übergangsfunktionen δ_ν gibt.

Der Entwurf des gesamten komplexen Schaltwerks muß letztlich *einen* Automaten liefern mit *einer* Ausgabefunktion Ω und *einer* Übergangsfunktion Δ; dieser Automat muß das Ablaufdiagramm realisieren, d. h. die Funktion Ω muß alle Funktionen ω_ν erfassen, und die Funktion Δ muß alle Funktionen δ_ν erfassen. Deshalb muß eine eindeutige Abbildung existieren vom Argument der Funktionen Ω und Δ auf die Menge unterschiedlicher Zuweisungspaare. Um die Existenz einer solchen Abbildung zu erzwingen, wird die Menge Z der sog. *Steuerzustände* eingeführt, deren Elemente nicht zur Menge A gehören. Die Elemente der Menge Z sind

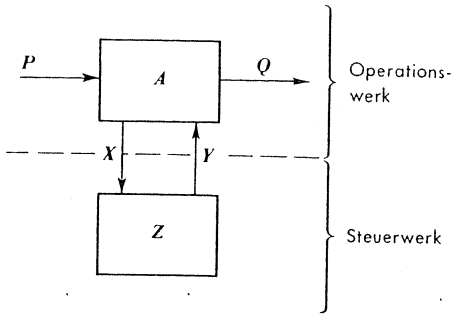


Bild 3. Grundzerlegung des komplexen Schaltwerks.

die internen Zustände des Steuerwerks; sie seien als Vektoren $\boldsymbol{Z}$ mit binären Komponenten z_n dargestellt.

Damit lassen sich nun die Automatengleichungen für das komplexe Schaltwerk anschreiben:

$$\boldsymbol{Q} := \Omega(\boldsymbol{P}, \boldsymbol{A}, \boldsymbol{Z}) \tag{4}$$

$$(\boldsymbol{A}, \boldsymbol{Z}) := \Delta(\boldsymbol{P}, \boldsymbol{A}, \boldsymbol{Z}) \tag{5}$$

Die Zerlegung des komplexen Schaltwerks in Operationswerk und Steuerwerk, wozu die — vorläufig noch formal zu betrachtenden — Verbindungsvektoren $\boldsymbol{X}$ und $\boldsymbol{Y}$ eingeführt werden müssen (s. Bild 3), bringt eine Zerlegung der Funktionen Ω und Δ in drei Ausgabefunktionen und zwei Übergangsfunktionen mit sich.

Die neuen Ausgabefunktionen sind:

$$\boldsymbol{X} := \Omega_X(\boldsymbol{P}, \boldsymbol{A}) \tag{6}$$

$$\boldsymbol{Y} := \Omega_Y(\boldsymbol{X}, \boldsymbol{Z}) \tag{7}$$

$$\boldsymbol{Q} := \Omega_Q(\boldsymbol{P}, \boldsymbol{A}, \boldsymbol{Y}) \tag{8}$$

Die neuen Übergangsfunktionen sind:

$$\boldsymbol{A} := \Delta_A(\boldsymbol{P}, \boldsymbol{A}, \boldsymbol{Y}) \tag{9}$$

$$\boldsymbol{Z} := \Delta_Z(\boldsymbol{X}, \boldsymbol{Z}) \tag{10}$$

In diesen fünf Funktionen treten drei unterschiedliche Argumentbereiche auf: $(\boldsymbol{P}, \boldsymbol{A})$, $(\boldsymbol{P}, \boldsymbol{A}, \boldsymbol{Y})$, $(\boldsymbol{X}, \boldsymbol{Z})$, wodurch die fünf Funktionen in drei getrennt zu betrachtende Klassen eingeteilt werden.

Zum Argumentbereich $(\boldsymbol{P}, \boldsymbol{A})$ gehört nur die eine Funktion Ω_X. Die Definition dieser Funktion Ω_X kann beliebig gewählt werden bis auf eine einzige Bedingung, welche erfüllt sein muß: Für jede Funktion β_μ in den Verzweigungsabfragen des Ablaufdiagramms muß eine Funktion γ_μ existieren, so daß gilt:

$$\beta_\mu(\boldsymbol{P}, \boldsymbol{A}) = \gamma_\mu[\Omega_X(\boldsymbol{P}, \boldsymbol{A})] \tag{11}$$

Dies bedeutet, daß die durch die Funktion Ω_X definierte Menge X mit den Elementen $\boldsymbol{X}$ sämtliche Information enthalten muß, welche in den Verzweigungsabfragen gebraucht wird. Mit anderen Worten ausgedrückt bedeutet es, daß dem Steuerwerk in Form des Vektors $\boldsymbol{X}$ all diejenige aus $\boldsymbol{P}$ und $\boldsymbol{A}$ ableitbare Information zugänglich sein muß, welche den Steuerablauf verzweigen kann.

Da eine entsprechende Wahl der Komponenten x_l von $\boldsymbol{X}$ im praktischen Fall kein Problem darstellt, kann damit die Betrachtung der Funktion Ω_X abgeschlossen werden.

Zum Argumentbereich $(\boldsymbol{P}, \boldsymbol{A}, \boldsymbol{Y})$ gehören die beiden Funktionen Ω_Q und Δ_A. Diese Funktionen müssen alle unterschiedlichen Funktionen ω_ν bzw. δ_ν erfassen, eine Eigenschaft, die auch schon von den Funktionen Ω und Δ gefordert wurde. Der bedeutende Unterschied zwischen den beiden Funktionspaaren (Ω, Δ) und (Ω_Q, Δ_A) besteht darin, daß einmal $\boldsymbol{Z}$, das andere Mal $\boldsymbol{Y}$ in ihren Argumenten enthalten ist. Im allgemeinen ist es unmöglich, die Menge Z so zu wählen, daß eine eins-zu-eins-Abbildung zwischen Z und der Menge unterschiedlicher Zuweisungspaare existiert; denn die Auswahl eines Zuweisungspaares kann vom gegenwärtigen Eingangsvektor $\boldsymbol{P}$ abhängen, der gegenwärtige Zustand $\boldsymbol{Z}$ ist dagegen vom gegenwärtigen Eingang $\boldsymbol{P}$ unabhängig. Demgegenüber ist es immer möglich, eine Menge Y so zu wählen, daß eine eins-zu-eins-Abbildung zwischen Y und der Menge unterschiedlicher Zuweisungspaare existiert, denn erstens darf das gegenwärtige $\boldsymbol{Y}$ nach Gleichung (6) und (7) vom gegenwärtigen $\boldsymbol{P}$ abhängen, und zweitens wurde schon im Abschnitt 3.1 erwähnt, daß durch die Komponenten y_k bestimmte Steuereingänge bei den Blöcken des Operationswerks versorgt werden und damit $\boldsymbol{Y}$ unmittelbaren Einfluß auf das gegenwärtig ausgegebene $\boldsymbol{Q}$ und die anschließende Änderung von $\boldsymbol{A}$ hat. Dadurch ist die geforderte eins-zu-eins-Abbildung gewährleistet.

Dadurch, daß nun also $\boldsymbol{Y}$ alleine die Auswahl eines Zuweisungspaares (ω_ν, δ_ν) bestimmt, ist die Existenz der Funktionen Ω_Q und Δ_A trivial, und da sie nur zur impliziten Definition von Y gebraucht wurden, brauchen sie nicht mehr weiter betrachtet zu werden.

Zum Argumentbereich $(\boldsymbol{X}, \boldsymbol{Z})$ gehören die beiden Funktionen Ω_Y und Δ_Z. Diese beiden Funktionen beschreiben das Steuerwerk als Mealy-Automaten. Der Entwurf des Steuerwerks muß die bislang unbekannte Menge Z und dazu die Funktionen Ω_Y und Δ_Z liefern.

4.2 Zustandsmenge Z

Da das im Abschnitt 3.2 eingeführte Ablaufdiagramm die gesammte Information über den gewünschten Steuerablauf enthält und da außerdem die Mengen X und Y im Abschnitt 4.1 so definiert wurden, daß $\boldsymbol{X}$ die Verzweigungsabfragen entscheidet und $\boldsymbol{Y}$ die Zuweisungspaare kennzeichnet, müssen also nun im ersten Schritt des Steuerwerksentwurfs die Vektoren $\boldsymbol{P}$ und $\boldsymbol{A}$ aus dem Ablaufdiagramm entfernt und dafür die Vektoren $\boldsymbol{X}$ und $\boldsymbol{Y}$ eingeführt werden. Dies geschieht durch folgende drei Substitutionen:

(1) Jede statische Zuweisung

$$\boldsymbol{Q} := \omega_\nu(\boldsymbol{P}, \boldsymbol{A}) \tag{2}$$

wird ersetzt durch

$$\boldsymbol{Y} := \boldsymbol{Y}_\nu, \tag{12}$$

wobei $\boldsymbol{Y}_\nu$ derjenige Steuervektor ist, welcher eindeutig das Funktionspaar (ω_ν, δ_ν) auswählt.

(2) Jede dynamische Zuweisung

$$\boldsymbol{A} := \delta_\nu(\boldsymbol{P}, \boldsymbol{A}) \tag{3}$$

wird ersetzt durch die von δ_ν implizierten Übergangsbeschränkungen für $\boldsymbol{X}$.

Dies soll kurz an einem einfachen Beispiel erläutert werden: Mit x_l werde der Stand eines Zählers abgefragt, der einen Teil von $\boldsymbol{A}$ bildet; x_l sei nur dann „1", wenn der Zählerstand 25 beträgt. Durch δ_ν werde der Zähler zurück, d. h. auf null gesetzt. Damit impliziert δ_ν eine Übergangsbeschränkung für $\boldsymbol{X}$, denn nach Ausführung der dynamischen Zuweisung ist x_l mit Sicherheit „0".

Es gibt drei verschiedene mögliche Typen von Übergangsbeschränkungen für $\boldsymbol{X}$:

$$\varrho(\boldsymbol{X}) := \begin{cases} \dfrac{\varrho(\boldsymbol{X})}{\varrho(\boldsymbol{X})} & \text{(13a)} \\ \text{„1"} & \text{(13b)} \\ & \text{(13c)} \end{cases}$$

Die formale binäre Funktion ϱ gibt dabei irgendeine logische Eigenschaft von $\boldsymbol{X}$ an, d. h. ϱ beschreibt irgendeinen logischen Ausdruck, dessen Variablen die Komponenten von $\boldsymbol{X}$ sind. Die drei Typen von Übergangsbeschränkungen in Gleichung (13) sind wie folgt zu interpretieren:

(13a) Eine bestimmte logische Eigenschaft von $\boldsymbol{X}$ bleibt unverändert.

(13b) Eine bestimmte logische Eigenschaft von $\boldsymbol{X}$ wird invertiert.

(13c) Eine bestimmte logische Eigenschaft von $\boldsymbol{X}$ wird definiert gesetzt.

Für das angeführte Beispiel gilt der Fall (13c) mit $\varrho(\boldsymbol{X}) = \bar{x}_l$. Es hängt vom einzelnen δ_ν ab, wieviele Übergangsbeschränkungen für $\boldsymbol{X}$ impliziert werden, keine, eine oder mehrere.

(3) Jede Verzweigungsabfrage

$$\beta_\mu(\boldsymbol{P}, \boldsymbol{A}) = \text{„1“?} \tag{1}$$

wird ersetzt durch die Abfrage

$$\gamma_\mu(\boldsymbol{X}) = \text{„1“?} \tag{14}$$

Der zweite Schritt des Steuerwerksentwurfs besteht darin, aus dem im ersten Schritt gewonnenen Ablaufdiagramm eine Zustandsmenge Z abzuleiten, welche dann in einem dritten Schritt einem Minimisierungsprozeß unterworfen wird. Die Ableitung einer Zustandsmenge Z aus dem Ablaufdiagramm gestaltet sich äußerst einfach, denn das Ablaufdiagramm ist nichts anderes als eine besondere Form des Übergangsgraphen eines Mealy-Automaten.

Die Zahl der Zustände ist gleich der Anzahl der Zuweisungspaare; der Zustand $\boldsymbol{Z}_\nu$, welcher zu einem bestimmten Zuweisungspaar (ω_ν, δ_ν) gehört, ist definiert als derjenige Zustand, in den das Steuerwerk bei der dynamischen Zuweisung ν übergeht.

Es ist wichtig zu beachten, daß $\boldsymbol{Z}_\nu$ nicht als derjenige Zustand definiert werden kann, in welchem sich das Steuerwerk während der statischen Zuweisung ν befindet.

4.3 Zur Zustandsminimisierung

Damit einer der bekannten Zustandsminimisierungsalgorithmen [9], [10] angewandt werden kann, muß die im Ablaufdiagramm enthaltene Information in eine Automatentafel übertragen werden.

Das Steuerwerk kann ein unvollständig definierter Automat sein, und zwar aufgrund der in Gleichung (13) dargestellten Übergangsbeschränkungen für $\boldsymbol{X}$, welche u. U. bewirken können, daß in einem bestimmten Zustand $\boldsymbol{Z}$ Vektoren $\boldsymbol{X}$ aus einer bestimmten Teilmenge von X nicht vorkommen können.

Die Aufstellung der Automatentafel wird wesentlich durch die i. a. große Zahl von Elementen in der Menge X erschwert. Während ein Ablaufdiagramm mit beispielsweise 8 Komponenten in $\boldsymbol{X}$ normalerweise noch gut überschaubar ist, ist eine Automatentafel, welche pro Element von X eine Spalte, also 256 Spalten enthält, ohne Rechenanlage vielleicht noch mühsam aufzustellen, aber nicht mehr zu bearbeiten. Ein Ablaufdiagramm ist so übersichtlich, weil es nicht die einzelnen Elemente von X betrachtet, sondern eine wesentlich kleinere Zahl von Teilmengen X_α der Menge X, welche formal durch logische Funktionen τ_α definiert werden:

$$X_\alpha = (\boldsymbol{X} \mid \tau_\alpha(\boldsymbol{X}) = \text{„1“}) \tag{15}$$

Die Funktionen τ_α sind Konjunktionen von Funktionen γ_μ oder deren Komplement; beispielsweise könnte ein τ_α die Form $\gamma_1 \cdot \gamma_3 \cdot \bar{\gamma}_4$ haben. Diese Konjunktionen werden durch die möglichen Kaskaden von Verzweigungsfragen im Ablaufdiagramm bestimmt.

Es stellt sich nun natürlich die Frage, ob diese Menge der Teilmengen X_α nicht zu einer Reduktion der Spaltenzahl in der Automatentafel herangezogen werden kann. Bei der Untersuchung dieser Frage muß die wesentliche Tatsache beachtet werden, daß die Menge der Spalten einer Automatentafel einer Partition der Menge X entsprechen muß. Dagegen stellt die Menge der Teilmengen X_α i. a. keine Partition der Menge X dar, sondern die Vereinigung mehrerer solcher Partitionen, da jede Abfragenkaskade eine Partition für sich ergibt.

Es besteht also das Problem, eine Partition der Menge X mit minimaler Anzahl von Partitionsblöcken X_π (Teilmengen von X) zu finden unter der Bedingung, daß jedes X_α entweder einem X_π gleicht oder sich als Vereinigung mehrerer X_π ausdrücken läßt. Die gesuchte Partition ergibt sich als sog. „Produkt“ der in der Menge der Teilmengen X_α enthaltenen Partitionen (s. [9] oder [10]); diese Produktbildung soll hier aber nicht behandelt werden.

In der Praxis zeigt sich, daß sich nur in den seltensten und einfachsten Fällen eine solche Reduktion der Spaltenzahl erzielen läßt, daß die sich ergebende Automatentafel ohne Verwendung einer Rechenanlage sinnvoll bearbeitbar wird.

Glücklicherweise läßt sich durch Betrachtungen am Ablaufdiagramm ohne Aufstellen einer Automatentafel i. a. eine beträchtliche Zustandsreduktion, oft bis zum Minimum, erzielen. Diese Möglichkeit geht zurück auf eine Überlegung, welche im Abschnitt 4.1 angestellt wurde, wonach die Menge Z so gewählt werden muß, daß sich eine eindeutige Abbildung ergibt vom Argument $(\boldsymbol{P}, \boldsymbol{A}, \boldsymbol{Z})$ auf die Menge unterschiedlicher Zuweisungspaare (ω_ν, δ_ν).

Ein Beispiel einer Zustandsminimisierung am Ablaufdiagramm ist im Abschnitt 5.3 zu finden.

4.4 Interne Struktur des Steuerwerks

Für die interne Struktur des Steuerwerks gibt es zwei grundsätzlich unterschiedene Möglichkeiten.

(1) Direkter Schaltwerksentwurf

Es handelt sich um die routinemäßige Realisierung eines synchronen Schaltwerkes [11, 12]. Nachdem die Menge Z der Steuerzustände bestimmt wurde, müssen die Elemente $\boldsymbol{Z}$ binär codiert werden (Problem der Sekundärzuweisung), und es muß eine entsprechende Anzahl Flipflops zur Zustandsspeicherung vorgesehen werden. Die Flipflopansteuerfunktionen und die Ausgabefunktionen folgen dann schematisch aus dem gewählten Zustandscode und dem Ablaufdiagramm (Problem der Schaltnetzminimisierung).

(2) Entwurf eines Mikroprogrammwerks

Im Abschnitt 2 wurde kurz die Frage der Mikroprogrammierung behandelt. Dabei wurde gesagt, daß sie eingeführt wurde als eine bestimmte Realisierungsform für das Steuerwerk. Wenn diese Realisierungsform gewählt wird, dann entfallen sämtliche außer dem ersten der Schritte zum Entwurf des Steuerwerks, welche in 4.2 und 4.3 dargestellt wurden, d. h. es genügt das mit X und Y dargestellte Ablaufdiagramm.

In diesem Fall wird das Steuerwerk als komplexes Schaltwerk im Sinne des Abschnitts 1 aufgefaßt, für das ein Operationswerk entworfen werden muß. Was in 3.1 zum Entwurf des Operationswerks gesagt wurde, gilt auch hier: Es muß aus einer umfangreichen Menge in Frage kommender innerer Strukturen eine bestimmte ausgewählt werden, und zur Durchführung dieser Auswahl stehen keine nützlichen

Algorithmen zur Verfügung. Die in Frage kommenden inneren Strukturen des Operationswerks eines Mikroprogrammwerks (man beachte: *nicht* des vom Mikroprogrammwerk gesteuerten Operationswerks!) haben eines gemeinsam: Unter den Blöcken des Operationswerks befindet sich stets ein ausgezeichneter Block, der sog. Mikroprogrammspeicher, dessen Inhalt das externe Verhalten des Mikroprogrammwerks bestimmt. Die Hardware eines Mikroprogrammwerks stellt also nicht die Realisierung eines einzigen bestimmten Steuerwerks dar, sondern einer Menge von Steuerwerken, wobei die Auswahl eines Elements aus dieser Menge durch Einspeichern einer bestimmten Software in den Mikroprogrammspeicher erfolgt [13], [14].

Oft liefert der Entwurf des Operationswerks für ein Mikroprogrammwerk ein autonomes Operationswerk, bei dem zur Koordination der Zusammenarbeit der Blöcke kein Steuerwerk benötigt wird. Im anderen Fall muß noch ein Steuerwerk innerhalb des Mikroprogrammwerks entworfen werden, was zweckmäßigerweise im direkten Schaltwerksentwurf geschieht.

Eine systematische Betrachtung der Operationswerke von Mikroprogrammwerken würde über den Rahmen der vorliegenden Arbeit hinausgehen.

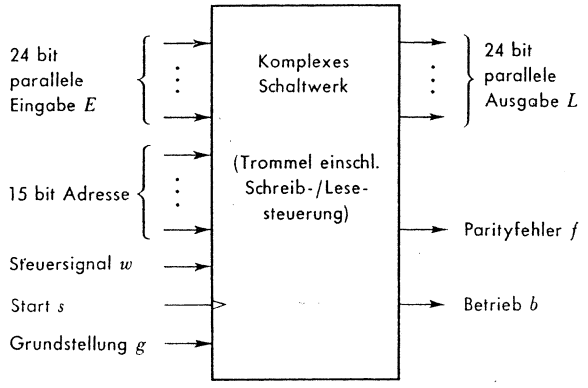


Bild 4. Blockdarstellung des zu entwerfenden komplexen Schaltwerks.

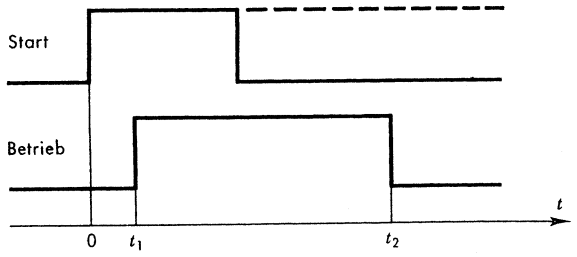


Bild 5. Zeitverhältnisse zwischen Signalen des Werkes in Bild 4.

5. Beispiel

5.1 Aufgabenstellung

Es ist eine Schreib-/Lesesteuerung für eine Magnettrommel zu entwerfen [15]. Die Trommel ist vorgegeben mit 128 Datenspuren. Es sollen pro Spur 256 Wörter zu je 25 bit (24 Datenbit + 1 Paritybit) gespeichert werden. Das Paritybit soll so bestimmt werden, daß die Anzahl der Einsen im 25-bit-Wort gerade ist.

Bild 4 zeigt das komplexe Schaltwerk als Block mit allen Ein- und Ausgangssignalleitungen. Der Eingangsvektor $\boldsymbol{P}$ hat 42 Komponenten, und der Ausgangsvektor $\boldsymbol{Q}$ hat 26 Komponenten. Das Steuersignal w entscheidet den durch „Start" ausgelösten Vorgang: w = „0" bedeutet Lesen und w = „1" bedeutet Schreiben. Bild 5 zeigt, wie die beiden Signale „Start" und „Betrieb" die Zeiten t_1 und t_2 definieren. Bezogen auf dieses Zeitschema gelten folgende Spezifikationen:

(1) Im Intervall zwischen 0 und t_1 kann das einzuschreibende Wort $\boldsymbol{E}$ abgefragt werden, d. h. in diesem Intervall darf es als konstant anliegend vorausgesetzt werden.

(2) Im Intervall zwischen 0 und t_2 können die Adresse und das Steuersignal w abgefragt werden, d. h. in diesem Intervall dürfen sie als konstant anliegend vorausgesetzt werden.

(3) Von t_2 an muß das Ergebnis konstant am Ausgang anliegen; als Ergebnis ist hier entweder das erfolgreich gelesene Wort $\boldsymbol{L}$ oder bei Lesemißerfolg die Parityfehlermeldung f zu verstehen.

(4) Es darf vorausgesetzt werden, daß im Intervall zwischen 0 und t_1 keine Rückflanke und im Intervall zwischen 0 und t_2 keine Vorderflanke von „Start" auftritt.

(5) Solange Parityfehlermeldung f ausgegeben wird, soll „Start" keinen Vorgang auslösen können.

(6) Das Grundstellungssignal g soll jeden evtl. ablaufenden Vorgang abrupt beenden; außerdem soll g die Parityfehlermeldung f löschen.

5.2 Entwurf des Operationswerks

5.2.1 Blockstruktur

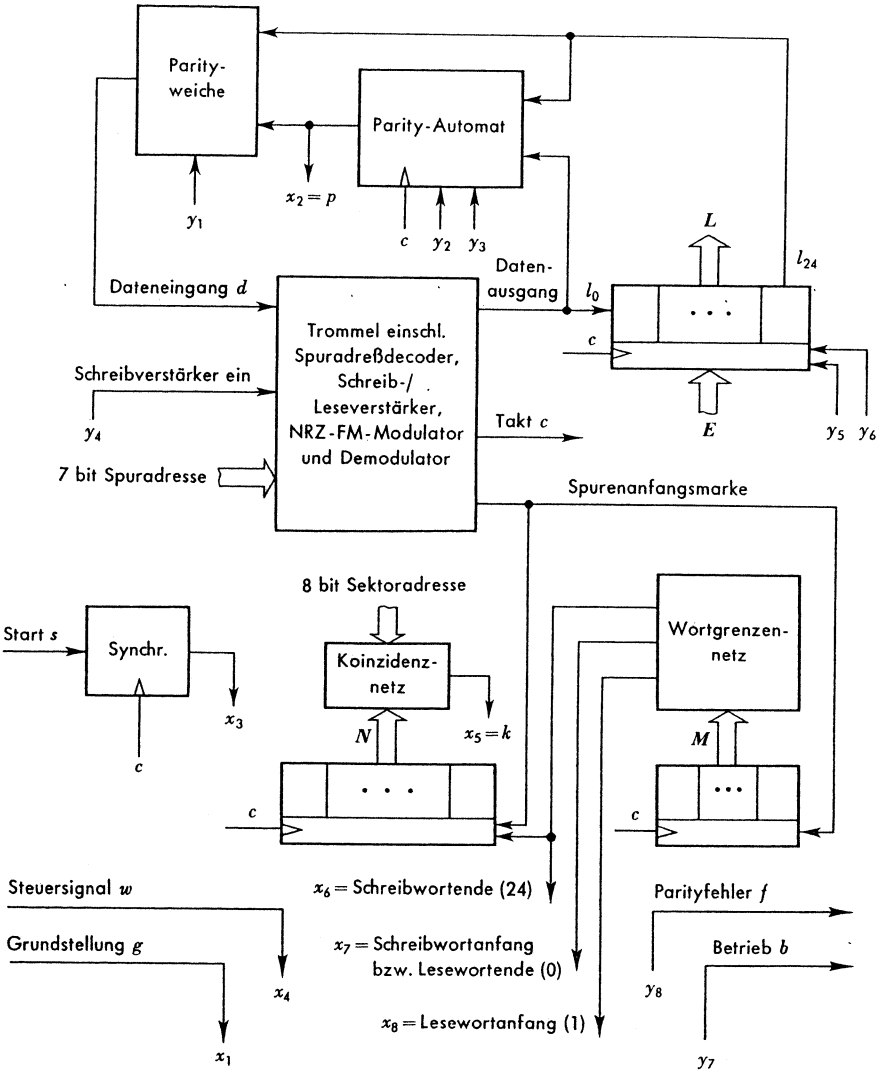
Da das Entwerfen eines Operationswerks kein algorithmischer Vorgang ist, kann hier nicht in Form logischer Schlußfolgerungen dargestellt werden, wie aus der Aufgabenstellung im Abschnitt 5.1 die Blockstruktur in Bild 6 gewonnen wurde.

Das Operationswerk in Bild 6 enthält außer dem Trommelblock nur noch acht weitere Blöcke, so daß es also leicht überschaubar ist. Für das Verständnis des Schreib-/Lesevorgangs sind die Zeitverhältnisse zwischen dem seriellen Dateneingang d und dem seriellen Datenausgang l_0 des Trommelblocks äußerst wichtig. Die an d anliegende Information wird auf diejenigen Stellen der Trommel geschrieben, welche sich unter dem ausgewählten Kopf befinden, solange „Schreibverstärker ein" gilt, d. h. solange y_4 auf „1" steht. Das Schreibverfahren verwendet eine Non-Return-to-Zero Frequenzmodulation, was aber weiter nicht beachtet werden muß. Ein an eine bestimmte Stelle der Trommel geschriebenes Bit kann zwar selbstverständlich genau dann vom Kopf wieder gelesen werden, wenn sich die bestimmte Stelle wieder unter dem Kopf befindet, aber durch die notwendige Demodulation wird dieses Bit am Ausgang l_0 erst dann verfügbar, wenn die zugehörige Stelle den Kopf schon wieder verlassen hat und die zum nächsten Bit gehörige Stelle sich unter dem Kopf befindet. Bild 7 versucht dies zu veranschaulichen, und zwar nicht, wie es für ein einzelnes Bit aussieht, sondern für ein ganzes Wort.

Da die einzelnen Bitintervalle auf der Spur mitgezählt werden müssen, werden zwei Zähler $\boldsymbol{M}$ und $\boldsymbol{N}$ benötigt, wobei $\boldsymbol{M}$ zyklisch von 0 bis 24 zählt und $\boldsymbol{N}$ zyklisch von 0 bis 255. Der Zählvorgang von $\boldsymbol{N}$ wird durch $\boldsymbol{M}$ gesteuert, und zwar zählt $\boldsymbol{N}$ genau nur dann, wenn $\boldsymbol{M}$ von 24 nach 0 springt. Beide Zähler werden durch die Spurenanfangsmarke, einem Impuls, welcher in einer Taktlücke sitzt, auf 0 gesetzt und dadurch mit der Trommelstellung synchronisiert.

Zum Vergleich der in $\boldsymbol{N}$ angezeigten Sektoradresse mit der extern eingegebenen Sektoradresse wird ein Koinzidenznetz benötigt, welches das Koinzidenzsignal k liefert.

Bild 6. Blockzerlegung des Operationswerks.



Zur Steuerung des Schreib-/Lesevorgangs wird Information über die entsprechenden Wortgrenzen benötigt, welche ein Netz aus dem Zählerstand M ableitet.

Zur Serien-Parallel- bzw. Parallel-Serienwandlung wird ein 24-stelliges Schieberegister L benötigt, welches durch einen zweistelligen Steuercode (y_5, y_6) steuerbar sein muß (s. Tabelle 1), damit es je nach Bedarf E parallel übernehmen, rechtsschieben oder stillstehen kann.

Tabelle 1. Funktionssteuerung des Schieberegisters L.

y_5	y_6	Takt bewirkt
0	0	Keine Zustandsänderung
0	1	Parallelübernahme des Eingangswortes
1	0	Nach rechts Schieben um eine Stelle
1	1	Undefinierte Zustandsänderung

Zur Bestimmung bzw. Auswertung des Paritybits ist der Parityautomat vorgesehen, welcher durch einen zweistelligen Steuercode (y_2, y_3) steuerbar sein muß, damit er je nach Bedarf die Bitfolge bei l_0 auswerten, die Bitfolge bei l_{24} auswerten, in Ausgangsstellung gehen oder stillstehen kann. Tabelle 2 beschreibt die Funktion dieses Automaten.

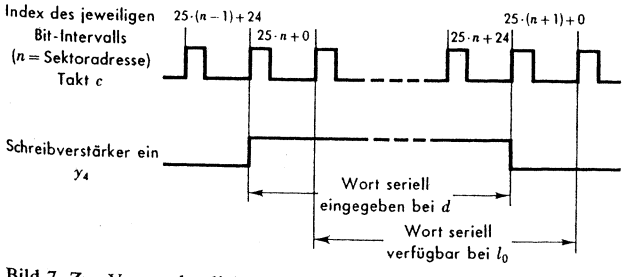


Bild 7. Zur Veranschaulichung der Leseverzögerung.

Tabelle 2. Funktionssteuerung des Parityautomaten.

y_2	y_3	Takt bewirkt
0	0	$p := p$ Keine Zustandsänderung
0	1	$p :=$ „0“ Rücksetzen
1	0	$p := (p \neq l_0)$ Auswertung der Bitfolge bei l_0
1	1	$p := (p \neq l_{24})$ Auswertung der Bitfolge bei l_{24}

Da es zwei Informationsquellen beim Schreiben gibt, nämlich die Registerstelle l_{24} und den Ausgang p des Parityautomaten, muß eine Weiche vorgesehen werden. Die Weichensteuerung sei derart, daß $y_1 =$ „1“ das Paritybit durchschaltet.

Der Start s muß synchronisiert werden. Dagegen ist aufgrund der in Punkt (2) des Abschnitts 5.1 formulierten Abhängigkeit des Steuersignals w vom Startsignal eine Synchronisation von w nicht erforderlich. Auch das Grundstellungssignal g braucht nicht synchronisiert zu werden, da es eine in Punkt (6) des Abschnitts 5.1 formulierte Vormachtstellung genießt.

Da die Bereitstellung der Signale Betrieb b und Parityfehler f nicht in offensichtlichem Zusammenhang mit den betrachteten Blöcken steht, muß sie dem Steuerwerk überlassen werden.

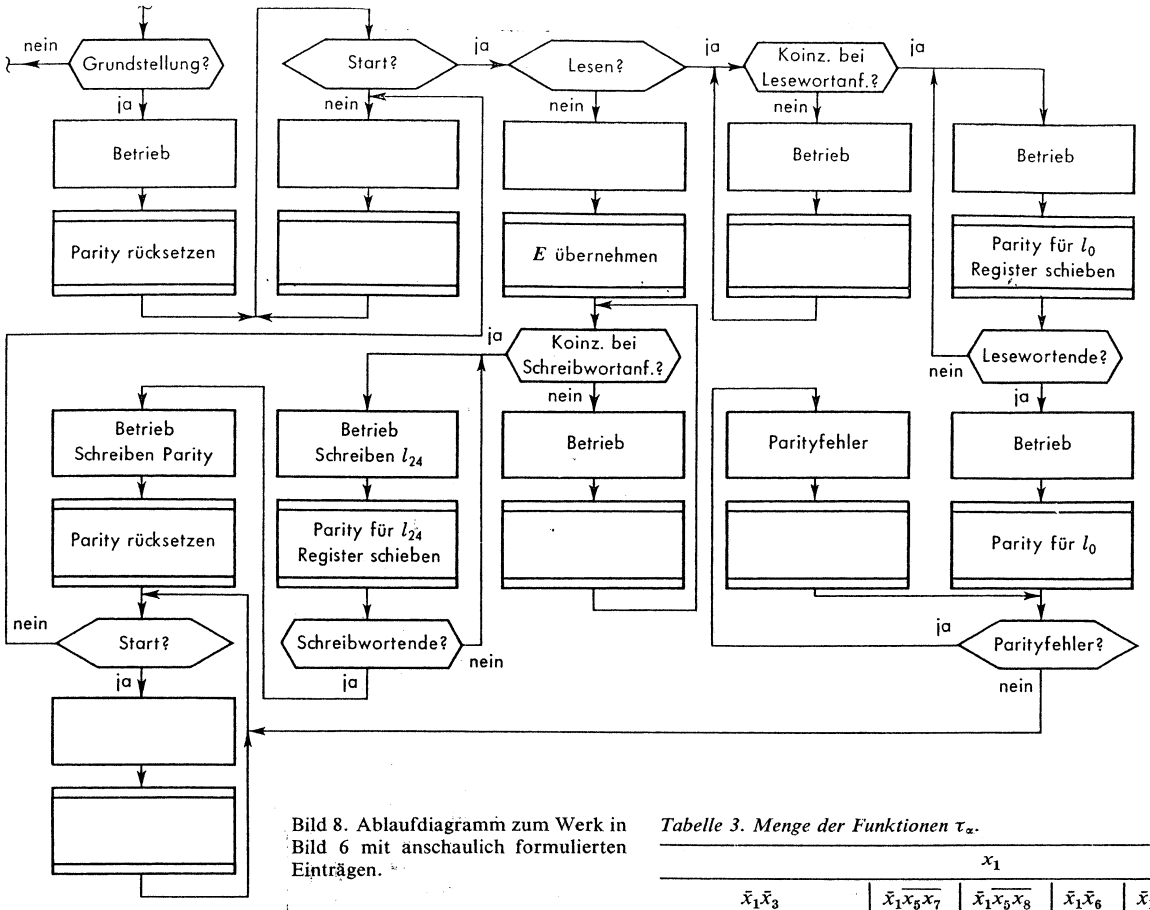


Bild 8. Ablaufdiagramm zum Werk in Bild 6 mit anschaulich formulierten Einträgen.

Tabelle 3. Menge der Funktionen τ_α.

x_1						
$\bar{x}_1\bar{x}_3$		$\bar{x}_1\overline{x_5x_7}$	$\bar{x}_1\overline{x_5x_8}$	$\bar{x}_1\bar{x}_6$	$\bar{x}_1\bar{x}_7$	$\bar{x}_1x_2$
$\bar{x}_1x_3$	$\bar{x}_1x_3x_4$ $\bar{x}_1x_3\bar{x}_4x_5x_8$ $\bar{x}_1x_3\bar{x}_4x_5x_8$	$\bar{x}_1x_5x_7$	$\bar{x}_1x_5x_8$	$\bar{x}_1x_6$	$\bar{x}_1x_7$	$\bar{x}_1\bar{x}_2\bar{x}_3$ $\bar{x}_1\bar{x}_2x_3$

5.2.2 Ablaufdiagramm

Nach den im vorangegangenen Abschnitt gegebenen Erklärungen zur Blockstruktur des Operationswerks müßte das Ablaufdiagramm in Bild 8 ohne weitere Erläuterung verständlich sein.

Der Übersichtlichkeit wegen wurden nur die zum Verständnis des Steuerablaufs notwendigen Zuweisungen eingetragen. Dadurch bleiben manche Signale in einigen Zuweisungen unerwähnt, und etliche Zuweisungssymbole sind sogar ganz leer geblieben. Die nicht erwähnten Zuweisungen können definiert oder mehr oder weniger beliebig sein, was sich durch Einsicht in die Funktion des Operationswerkes erkennen läßt. Beispielsweise müssen die Signale Betrieb *b* und „Schreibverstärker ein" in denjenigen statischen Zuweisungen, wo sie nicht erwähnt werden, den Wert „0" zugewiesen bekommen, während beispielsweise die Steuerung der Parityweiche in den sie nicht erwähnenden Zuweisungen beliebig sein darf.

Die vollständige Information über die im Ablaufdiagramm zum Operationswerk nicht erwähnten Zuweisungen wird zweckmäßigerweise erst beim im nächsten Abschnitt durchgeführten Eintrag der Vektoren $\boldsymbol{Y}$ in das Ablaufdiagramm berücksichtigt.

5.3 Entwurf des Steuerwerks

In der Darstellung der Blockstruktur des Operationswerks in Bild 6 wurden bereits die Komponenten der Vektoren $\boldsymbol{X}$ und $\boldsymbol{Y}$ eingetragen. Beide Vektoren haben jeweils acht Komponenten.

Der Eintrag dieser Vektoren in das Ablaufdiagramm nach den Regeln in Abschnitt 4.2 ergibt das Diagramm in Bild 9.

Das Ablaufdiagramm enthält 11 Zuweisungspaare, was nach Abschnitt 4.2 eine Menge Z mit 11 Zuständen ergibt, welche noch minimisiert werden kann. Zur Bestimmung der minimal notwendigen Zahl von Spalten in der dem Ablaufdiagramm entsprechenden Automatentafel wird die Menge der Teilmengen X_α betrachtet. Tabelle 3 gibt die zugehörigen 17 Funktionen τ_α an, welche sich aus den Abfragekaskaden im Ablaufdiagramm ergeben. Tafel 3 zeigt, daß die Menge der Teilmengen X_α die Vereinigung von sieben Partitionen der Menge X darstellt. Das Produkt dieser sieben Partitionen ergibt eine hier nicht angegebene Partition mit 31 Partitionsblöcken X_π, was bedeutet, daß die dem Ablaufdiagramm entsprechende Automatentafel 31 Spalten hat. Da im gegebenen Fall die Zustandsminimisierung am Ablaufdiagramm wesentlich leichter durchzuführen ist als mit Hilfe einer 31-spaltigen Tafel, erübrigt sich die Angabe der 31 Partitionsblöcke X_π.

Der erste Schritt der Zustandsreduktion am Ablaufdiagramm ist offensichtlich: Sämtliche dynamischen Zuweisungen, welche zu einem gemeinsamen Diagrammpunkt führen, dürfen den Steuerautomaten in denselben Zustand bringen. Damit ergeben sich 7 unterschiedliche Zustände, welche in den linken unteren Ecken der dynamischen Zuweisungen in Bild 9 eingetragen sind.

Die Möglichkeit zur weiteren Zustandsreduktion beruht auf den Übergangsbeschränkungen für $\boldsymbol{X}$. Da das Steuersignal $w = x_4$ solange abfragbar bleibt, wie der Schreib- bzw. Lesevorgang läuft, können in beiden Vorgängen dieselben Steuerzustände verwendet werden. Somit können also die bisherigen Zustände 2 und 3 zu einem Zustand vereinigt werden

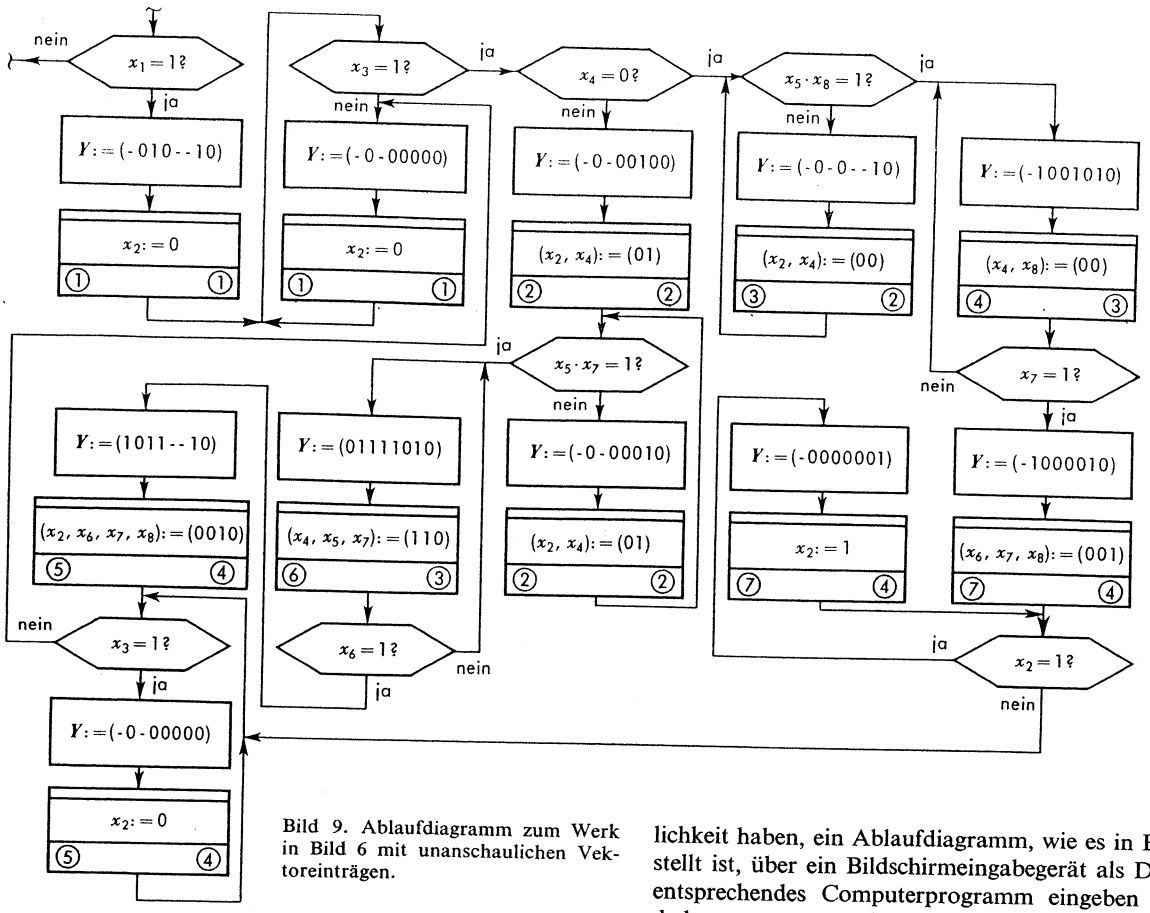


Bild 9. Ablaufdiagramm zum Werk in Bild 6 mit unanschaulichen Vektoreinträgen.

(mögliche Zustandsinterpretation: „Warten auf Koinzidenz"), und ebenso dürfen die bisherigen Zustände 4 und 6 zu einem Zustand vereinigt werden („Schiebebetrieb"). Da im bisherigen Zustand 7 der Ausgang des Parityautomaten $p = x_2$ abgefragt wird und im bisherigen Zustand 5 der Ausgang x_2 definiert auf „0" steht, dürfen diese beiden bisherigen Zustände auch zu einem Zustand vereinigt werden. Die sich somit ergebende minimale Zustandsmenge Z hat also nur noch 4 Zustände, welche in den rechten unteren Ecken der dynamischen Zuweisungen in Bild 9 eingetragen sind.

Der weitere Entwurf des Steuerwerks nach Abschnitt 4.4 Punkt (1) ist bekannte Routine und braucht deshalb hier nicht dargestellt zu werden.

6. Schlußbetrachtung

Die dargestellte Methode zum Entwurf komplexer, synchron gesteuerter Schaltwerke wurde zur Durchführung am Schreibtisch entwickelt; das Ziel, ein möglichst hoher Grad an Übersichtlichkeit, wurde erreicht. Das dargestellte Beispiel zeigt, wie wenig Voraussetzungen an die Struktur des Operationswerkes gestellt werden, wodurch die Methode für eine große Klasse von Aufgaben anwendbar wird.

Da die beim Steuerwerksentwurf i. a. notwendige Schaltnetzminimisierung aufgrund der großen Eingangsvariablenzahl (Summe der Komponentenzahlen von X und Z) ohne Computerprogramm meist nicht durchführbar ist, stellt sich die Frage, ob nicht, wenn sowieso ein Computer verwendet werden muß, auch andere mehr oder weniger algorithmische Schritte des Entwurfs dem Computer übertragen werden sollen, nämlich die Zustandsminimisierung und die Sekundärzuweisung. In diesem Falle müßte man dann die Möglichkeit haben, ein Ablaufdiagramm, wie es in Bild 9 dargestellt ist, über ein Bildschirmeingabegerät als Daten für ein entsprechendes Computerprogramm eingeben zu können, d. h. es wäre wünschenswert, daß man die im Ablaufdiagramm enthaltene Information nicht vor der Eingabe in die Rechenanlage in eine formale Mikroprogrammsprache übersetzen muß. Arbeiten in dieser Richtung sind am Computer Science Department des Case Institute of Technology in Cleveland, Ohio, USA, im Gange.

Literatur

[1] *Glushkov, V. M.*, Automata Theory and Structural Design Problems of Digital Machines. Cybernetics 1 (1965), H. 1, S. 3—9.

[2] *Glushkov, V. M.*, Automata Theory and Formal Microprogram Transformations. Cybernetics 1 (1965), H. 5, S. 1—8.

[3] *Glushkov, V. M.*, Minimization of Microprograms and Algorithm Schemes. Cybernetics 2 (1966), H. 5, S. 1—3.

[4] *Gerace, G. B.*, Digital System Design Automation — a Method for Designing a Digital System as a Sequential Network System. IEEE Trans. Comp. C-17 (1968), S. 1044—1061.

[5] ACM Publications and Services Reference Guide. 6/69, ACM, New York.

[6] *Wilkes, M. V.*, The Best Way to Design an Automatic Calculating Machine. Manchester U. Computer Inaugural Conf., 1951, S. 16.

[7] *Wilkes, M. V.*, The Growth of Interest in Microprogramming: A Literature Survey. Computing Surveys 1 (1969), S. 139.

[8] *Rosin, R. F.*, Contemporary Concepts of Microprogramming and Emulation. Computing Surveys 1 (1969), S. 197.

[9] *Booth, T. L.*, Sequential Machines and Automata Theory. Wiley New York, 1967.

[10] *Unger, S. H.*, Asynchronous Switching Circuits. Wiley-Interscience, New York, 1969.

[11] *Schulte, D.*, Kombinatorische und sequentielle Netzwerke. Oldenbourg, München, 1967.

[12] *McCluskey, E. J.*, Introduction to the Theory of Switching Circuits. McGraw Hill, New York, 1965.

[13] *Tucker, S. G.*, Microprogram Control for the System/360. IBM Systems Journal 6 (1967), S. 222.

[14] *Husson, S. S.*, Microprogramming: Principles and Practices. Prentice Hall, Englewood Cliffs, 1970.

[15] *Eckstein, E.*, Entwurf und Aufbau einer Schreib-/Lesesteuerung für eine Magnettrommel. Diplomarbeit am Institut für Nachrichtenverarbeitung und -übertragung, Universität Karlsruhe, 1968.