2. Die Algorithmen: Technische Realisierung und Leistungsgrenzen

2.1. Das allgemeine Modell

Da der Algorithmenkomplex, der z.B. in /124/, /125/ beschrieben ist, für verschiedene Rechenanlagen programmtechnisch implementiert wurde (vgl. /65/, /156/), bieten naheliegende Beschleunigungsmaßnahmen keine grundsätzlichen Schwierigkeiten; so z.B. der Übergang auf das Mikroprogrammniveau eines gegebenen Rechners ("µP-Assist") bzw. die Schaffung eines "Spezialrechners" auf Basis verfügbarer Bit-Slice-Schaltkreise (vgl. /91/; zu derartigen Schaltkreisen s.z.B. /58/, /61/).
Es wird dann lediglich mit Mikrobefehlen statt mit Maschinenbefehlen programmiert (bei einer gegebenen Technologie - z.B. TTL - läuft ein Mikrobefehl etwa 2 ... 20 mal schneller ab als ein üblicher Maschinenbefehl, und die betreffende Hardware kann besser ausgenutzt werden).
Auch sind die Verknüpfungsoperationen der elementaren Algorithmen z.B. "Testen auf Orthogonalität", "Durchschnittsbildung" usw. (vgl. /124/) offensichtlich auf an sich einfache Weise mit Schaltnetzwerken ausführbar.
Eine ad hoc-Zusammenschaltung von Hardware auf Basis der vorliegenden Algorithmenbeschreibung (Operandenverknüpfungsnetzwerke und sequentielle Steuerung; eventuell auf Basis systematischer Entwurfsverfahren, wie sie z.B. in /23/,/42/ beschrieben sind) ist in ihrer Effektivität zunächst fragwürdig: unreflektierte Umsetzung von Algorithmenbeschreibungen in spezielle Hardware hat zur Folge, daß programmtechnische Flexibilität verloren geht und hat die Gefahr, daß bei manchen Algorithmen der Leistungsgewinn in keinem vernünftigen Verhältnis zu den Aufwendungen steht. Deshalb muß der Nutzen von Leistungsgewinn und Aufwand näher untersucht werden. Um ungeachtet der Vielzahl technischer Ausführungsmöglichkeiten allgemeingültige Resultate zu erhalten, ist eine abstrakte Betrachtungsweise erforderlich.
Die abstrakte Beschreibung und Untersuchung algorithmischer Abläufe betreffend existiert eine recht umfangreiche Literatur (vgl. etwa /40/). Die Motivation der meisten dieser Arbeiten ist allerdings den hier wesentlichen Problemen gerade entgegengesetzt: es handelt sich zumeist darum, von funktionellen Abläufen

in Hardware soweit wie möglich zu abstrahieren, um die Semantik von Programmiersprachen exakt zu beschreiben, Widersprüche in Definitionen aufzufinden usw. Demgegenüber ist hier genau zu untersuchen, wie eine möglichst effektive Hardware für einen gegebenen Algorithmus beschaffen sein muß und welche prinzipiellen Grenzen der Leistungsfähigkeit es gibt. Die Abstraktion betrifft somit nicht primär die Beschreibung von Datenstrukturen und Operationen, sondern deren Abbildung auf technische Mittel (dies erfordert von vornherein eine ingenieurtechnische Betrachtungsweise und schließt eine völlig streng mathematische aus).

Wird von Problemen der Ein- und Ausgabe abgesehen, so beruht die Ausführung eines Algorithmus in einer digitalen informationsverarbeitenden Einrichtung stets darauf, daß aus gespeicherter binär codierter Information (Argument-Information) in diskreten zeitlichen Schritten ein Resultat erzeugt wird, und zwar entweder als Ja/Nein-Aussage (Bedingung) oder als ebenfalls gespeicherte Resultat-Information.

Die einzelnen Abläufe heißen im folgenden Aktionen. Jede Aktion ist durch Signalflüsse von Schaltmitteln über Signalleitungen zu Schaltmitteln gekennzeichnet. Die gespeicherte Information (Argument oder Resultat) ist der ihr unterlegten Bedeutung nach irgendwie strukturiert, obwohl sie technisch durch eine oft homogen erscheinende Bitfolge repräsentiert wird. Jede Informationsstruktur (die Argument, Zwischenresultat oder Endresultat eines Algorithmus sein kann) wird als Objekt bezeichnet (im Sinne von /146/ und der darauf aufbauenden Arbeiten). Objekte können ihrerseits aus Objekten (Komponenten) aufgebaut sein; die Zerlegung endet sinnvollerweise beim einzelnen Bit.

Resultate werden durch Aktionen erzeugt, wobei die binären Repräsentationen von Objekten (die Komponenten höher aggregierter Objekte sein können) miteinander verknüpft werden. Dies erfordert technische Mittel, und zwar:

1. Speichermittel für die binären Repräsentationen von Argumenten und Resultat
2. Verknüpfungsschaltungen
3. Verbindungen zwischen 1. und 2.

Das Schema ist in Bild 1 dargestellt. Aktionen, die aus Argumentwerten Resultate erzeugen, heißen Operationen. Das Schema nach Bild 1 liegt jeder technischen Realisierung zugrunde, hat aber

recht enge Grenzen, was die direkte Ausführung von Algorithmen betrifft: für jedes Bit der binären Repräsentation von Argumenten und Resultat ist eine Speichereinrichtung erforderlich; jede Speichereinrichtung ist über eine Leitung an die Verknüpfungsschaltungen anzuschließen. Diese müssen jeder möglichen Argumentbelegung die entsprechenden Resultatwerte zuordnen, so daß nur recht elementare Operationen direkt nach diesem Schema ausführbar sind. Komplexere müssen gewissermaßen stückweise in mehreren Schritten ausgeführt werden. Um die jeweils benötigten Abschnitte der binären Repräsentationen den Verknüpfungsschaltungen zuzuführen bzw. in das Gesamt-Resultat einzuordnen, wird eine weitere Klasse von Aktionen benötigt. Derartige Aktionen werden im folgenden als Selektionen bezeichnet. Damit wird das allgemeine Schema wie folgt modifiziert (Bild 2):
Es sind Speichermittel vorgesehen, die zur abschnittsweisen Speicherung der binären Repräsentationen eingerichtet sind. Diese sind an weitere Schaltmittel (Selektionseinrichtungen) angeschlossen, so daß in jedem diskreten Zeitabschnitt ein Zugriff zu einem Abschnitt einer binären Repräsentation in jeder Speichereinrichtung möglich ist. Das Schema läßt sich leicht in das einer Registermaschine mit zentralem Speicher überführen, indem ein einziger Speicher mit einer entsprechend ausgestalteten Selektionseinrichtung beschaltet wird (d.h. mit allen Selektionseinrichtungen von Bild 2, einem Auswahlnetzwerk und einem Steuerautomaten) und indem Zwischenspeicher (Register) für jeweils einen Abschnitt der Argumente und des Resultats angeordnet werden (Bild 3). Damit wird die Bildung jedes Resultatabschnittes in mehrere Schritte zerlegt:

1. Selektion des Abschnittes des ersten Argumentes
2. Selektion des Abschnittes des zweiten Argumentes
3. Bildung des Resultatabschnittes durch Verknüpfung
4. Selektion des Resultatabschnittes im Speicher und Transport.

Der Steuerautomat (Sequencer) steuert die einzelnen Schritte durch Aktivieren von Auswahl- und Taktimpulsleitungen. Der Übergang von Bild 3 zum üblichen "v. Neumann-Rechner" ist offensichtlich, aber nicht Ziel der Untersuchungen. Dieses besteht ja gerade darin, einen gegebenen Algorithmus schneller auszuführen als es auf die übliche Weise durch programmierte Folgen elementarer Operationen möglich ist.

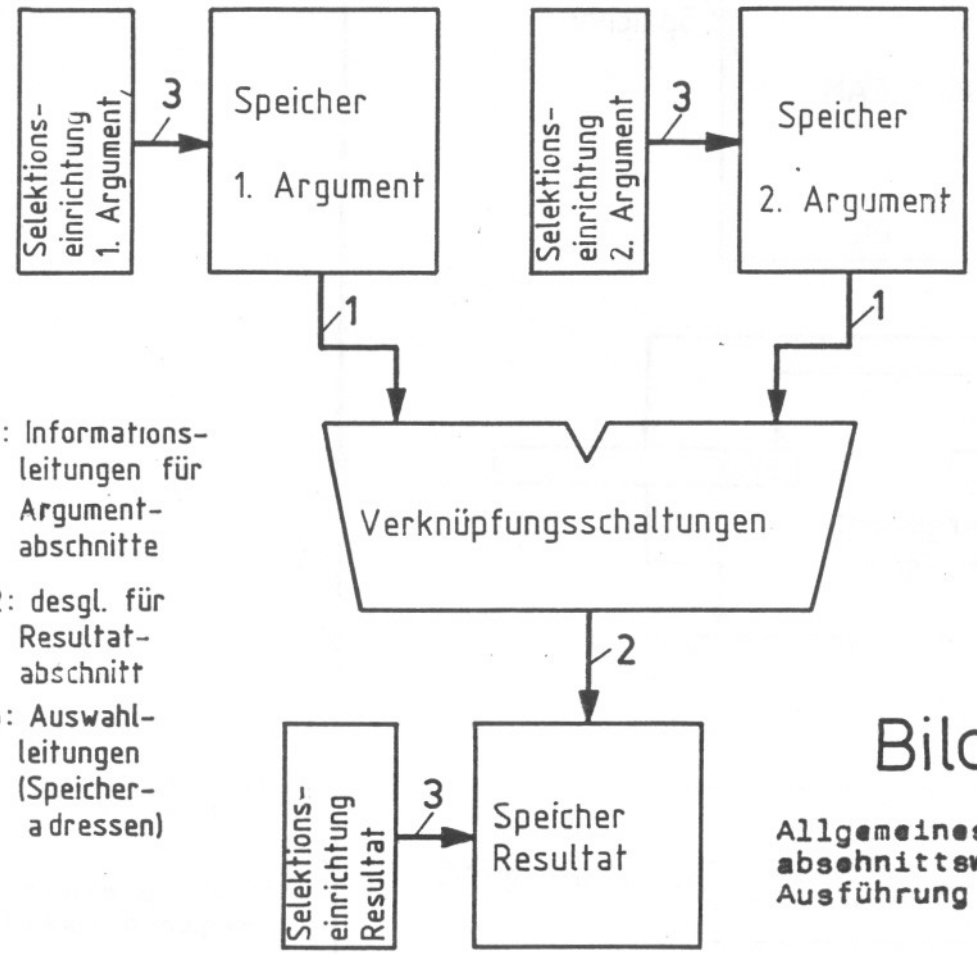
In diesem Sinne ist die direkte Ausführung gemäß Bild 1 die wünschbarste Lösung, da das Resultat in einem einzigen Zeitintervall geliefert wird.
Ein Algorithmus A sei wie folgt repräsentiert:

$$A = (N, D, SEQ) \qquad (1)$$

Es bedeuten:
N eine Menge von Einzelnamen: N = (NA, NR, NO) mit
NA: Menge der Namen der Argumentobjekte: NA = $(A_1 \ldots A_n)$
NR: Menge der Namen der Resultatobjekte: NR = $(R_1 \ldots R_m)$
NO: Menge der Namen der Operationen: NO = $(O_1 \ldots O_p)$
D eine Menge von Beschreibungen, die aus ebensovielen Teilmengen wie N besteht, wobei die jeweils korrespondierenden Teilmengen von N und D gleichmächtig sind:
DA: Menge der Argumentbeschreibungen: DA = $(DA_1 \ldots DA_n)$
DR: Menge der Resultatbeschreibungen: DR = $(DR_1 \ldots DR_m)$
DO: Menge der Operationsbeschreibungen: DO = $(DO_1 \ldots DO_p)$
damit ergibt sich D = (DA, DR, DO)
SEQ repräsentiert die Ablaufbeschreibung.

Dies korrespondiert mit der üblichen Darstellung eines Algorithmus in einer Programmiersprache. Die Menge N, D symbolisieren die Deklarationen der Datenstrukturen und elementaren Operationen (etwa im Sinne eines "package" in Ada, eines "cluster" in Clu usw.; s.z.B. /30/, /90/, /144/).
SEQ beschreibt den eigentlichen Ablauf, z.B. als eine Prozedur, in der die definierten Datenstrukturen und Operationen genutzt werden.
Die konkrete Form der jeweiligen Beschreibung ist ohne Bedeutung, es muß aber möglich sein, daraus Aussagen abzuleiten, z.B. hinsichtlich der Anzahl der Komponenten eines Objekts, der Anzahl der Bits in einer binären Repräsentation usw. Für diese Aussage seien folgende Funktionen definiert:

CARDC (Obj) repräsentiert die Anzahl der Komponenten des Objekts Obj, in die es in der nächstfolgenden Stufe der Deklaration zerlegt wird (bzw. aus denen es unmittelbar zusammengesetzt ist).



1: Informations-
leitungen für
Argument-
abschnitte

2: desgl. für
Resultat-
abschnitt

3: Auswahl-
leitungen
(Speicher-
adressen)

Bild 2

Allgemeines Schema der abschnittsweisen direkten Ausführung eines Algorithmus

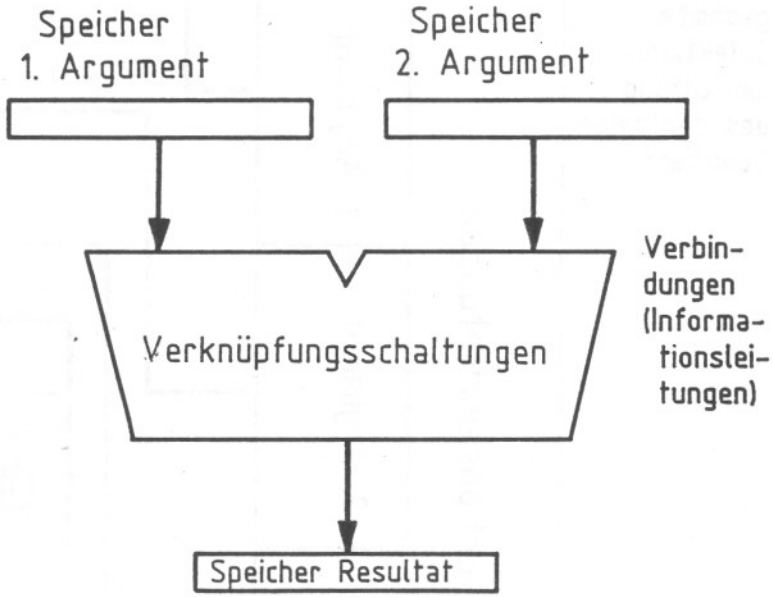


Bild 1

Allgemeines Schema der direkten Ausführung eines Algorithmus

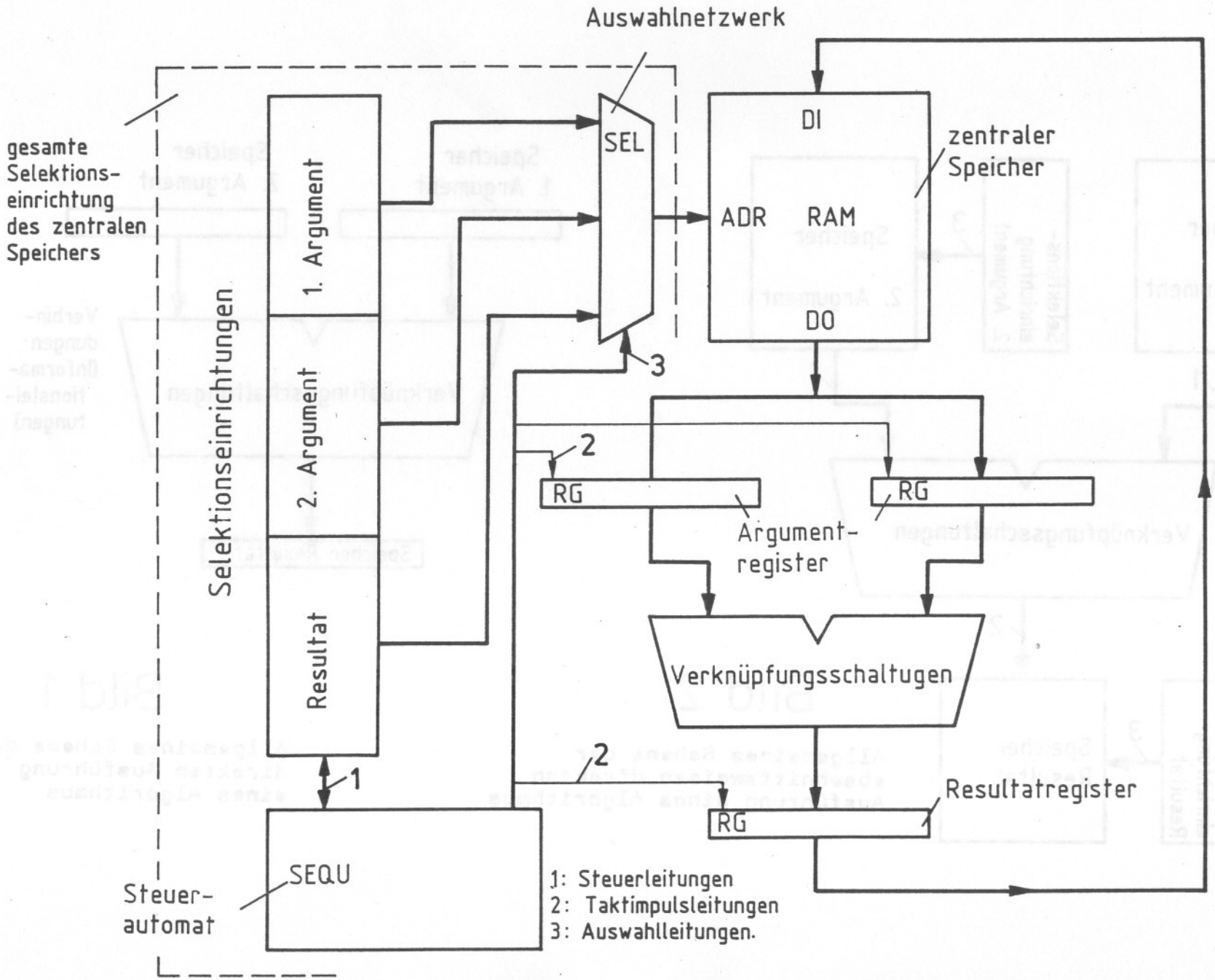


Bild 3

Schema einer Registermaschine

CARDB (Obj) gibt die Anzahl der Bits der binären Repräsentation des Objekts Obj an.

Mit diesen Voraussetzungen können die Bedingungen formuliert werden, unter denen ein Algorithmus nach dem Schema von Bild 1 ausführbar ist.

DEF. 1: Ein Algorithmus A, der dadurch ausgeführt werden kann, daß alle binären Repräsentationen seiner Argumente einem technisch realisierbaren kombinatorischen Netzwerk zugeführt werden, das die binäre Repräsentation des gesamten Resultats liefert, heißt durch Zuordnung ausführbar.

Die notwendigen und hinreichenden Bedingungen dafür sind:

1. Die Anzahl der Bits aller binären Repräsentationen von Argumenten und Resultat übersteigt nicht eine jeweils durch technische Gegebenheiten bedingte Obergrenze:

$$\sum_{i=1}^{n} \text{CARDB}\ (A_i) \leq \text{ARG\_LIMIT} \qquad (2)$$

$$\sum_{j=1}^{m} \text{CARDB}\ (R_j) \leq \text{RES\_LIMIT}$$

   Die Grenzwerte ARG_LIMIT, RES_LIMIT sind gegeben durch Aspekte wie
   - mögliche Anzahl von Kontaktverbindungen (an Schaltkreisen bzw. Leiterplatten)
   - physische Größe von Leiterplatten bzw. Schaltkreisen
   - die Packungsdichte der Komponenten
   - Überkopplung von Störungen.

   Plausible Obergrenzen sind:
   - für ARG_LIMIT 64 ... 128
   - für RES_LIMIT 32 ... 64.

   Das entspricht der üblichen Auslegung arithmetisch-logischer Einheiten (ALU's), mit denen aus zwei Argumenten zu je 32 oder 64 Bit ein Resultat gleicher Länge erzeugt wird.

2. Die durch die Beschreibung D und SEQ repräsentierten Aktionen sind auf BOOLEsche Gleichungen zurückführbar, in denen alle

Resultatbits als kombinatorische Verknüpfungen von Argumentbits erscheinen, und diese BOOLEschen Gleichungen sind technisch realisierbar.

Theoretische Erwägungen, die die Kompliziertheit BOOLEscher Gleichungen betreffen (s.z.B. /43/), sind für die Beurteilung der technischen Realisierbarkeit nicht ausreichend. Entscheidend sind in erster Linie die verfügbare Technologie und einschränkende Bedingungen (Kosten, physische Größe, Verlustleistung, Laufzeit usw.).

So ist eine einzelne BOOLEsche Gleichung unabhängig von ihrer Kompliziertheit technisch realisierbar, wenn dafür ein Zuordnerspeicher (ROM bzw. RAM) eingesetzt werden kann. Die Grenze ist zum einen durch die mit der Variablenzahl exponentiell wachsende Speicherkapazität gegeben und zum anderen durch die Speicherzugriffszeit sowie durch Kosten und Verlustleistung.

Richtwerte:

- Variablenzahl 8 ... 12 (perspektivisch: etwa 16)
- Zugriffszeit 10 ns für 8 ... 10 Variable (ECL-ROM);
  60 ns für 8 ... 12 Variable (Schottky-TTL);
  100 ... 500 ns für bis zu 16 Variable (MOS).

Soll eine BOOLEsche Gleichung mit PLA-Strukturen oder Gattern (einzelne oder innerhalb von gate array-Schaltkreisen) realisiert werden, so bestimmt neben einer allgemeinen Aufwandsgrenze (z.B. Beschränkung auf einen Schaltkreis) die Anzahl der hintereinander zu schaltenden Gatter ("Kettenlänge") letztlich die Realisierbarkeit.

Im besonderen steigen die Chancen der Realisierbarkeit, wenn wenigstens einer der beiden folgenden Definitionen erfüllt ist:

DEF. 2: Eine Menge BOOLEscher Gleichungen, die die Zuordnung aller Resultatbits aus allen Argumentbits beschreibt, heißt gruppenweise zerlegbar, wenn sich die Bildung aller Resultatbits auf kombinatorische Zusammenfassungen unabhängiger Verknüpfungen von Gruppen von Argumentbits zurückführen läßt und wenn sowohl die Verknüpfungen der Gruppen als auch deren kombinatorische Zusammenfassung technisch realisierbar sind.

DEF. 3: Eine Menge BOOLEscher Gleichungen heißt gruppenweise parallelisierbar, wenn sich für die Resultatbits eine Einteilung in Gruppen wie folgt durchführen läßt:

- zur Bildung von Resultatbits einer Gruppe wird kein Argumentbit benötigt, das zur Bildung von Resultatbits anderer Gruppen beiträgt
- die Lösungsmengen der BOOLEschen Gleichungen, die die Verknüpfungen für jede Gruppe beschreiben, sind untereinander isomorph (d.h. die Verknüpfungen für jede Gruppe können durch identische Netzwerke gebildet werden, die lediglich mit verschiedenen Argument- und Resultatleitungen beschaltet sind)
- die betreffenden Netzwerke sind technisch realisierbar.

(Der Begriff der Gruppe wird hier im üblichen Sinne zur Bezeichnung einer Zusammenfassung nach einem Ordnungskriterium gebraucht; nicht als mathematische Kategorie.)
Definition 2 wird durch Bild 4 veranschaulicht: es sind Netzwerke vorgesehen, die unabhängige Gruppen von Argumentbits miteinander verknüpfen. Die Resultatbits werden gebildet, indem die Ausgänge dieser Netzwerke mit weiteren Netzwerken zusammengefaßt werden.

Bild 5 illustriert Definition 3. Gruppen von Resultatbits werden durch Netzwerke gebildet, die jeweils mit verschiedenen Argumentbits beschaltet sind. Die Netzwerke der verschiedenen Gruppen sind jeweils untereinander identisch.
Da der Ausführbarkeit von Algorithmen durch direkte Zuordnung gemäß Bild 1 offensichtlich recht enge Grenzen gezogen sind, erweist sich die abschnittsweise Ausführung (Bild 2) als anstrebenswert: die Argumente werden den Verknüpfungsschaltungen gemäß der Anzahl der vorhandenen Informationsleitungen zugeführt, und die Resultate werden entsprechend abtransportiert. Damit wird in jedem diskreten Zeitabschnitt ein Beitrag zum Resultat geliefert, der mit den jeweiligen Aufwendungen (Gestaltung der Verknüpfungsschaltungen, Anzahl der Leitungen usw.) korrespondiert.
Es ist unmittelbar einsichtig, daß solche technischen Lösungsprinzipien hinsichtlich des Verhältnisses von Leistung und Aufwand die grundsätzlich günstigste Form der Implementierung von Algorithmen darstellen, da zur Erzeugung des Resulats nur soviele diskrete Zeitabschnitte benötigt werden, wie zum Transport der Argument- und Resultatabschnitte auf Grund der gegebenen Anzahl der Verbindungsleitungen erforderlich sind. D.h., in jedem Maschinenzyklus wird ein Beitrag zum Resultat geliefert, der der

Verarbeitungsbreite der Anordnung entspricht.
Ein solches Entwicklungsziel läßt hinreichend Raum für Intentionalität bei der konkreten Ausgestaltung der technischen Lösungswege.
Diese werden wesentlich durch allgemeine technisch-ökonomische Zielstellungen bestimmt. So hängt beispielsweise die Wahl der Schaltkreistechnologie und der Verarbeitungsbreite davon ab, ob etwa ein Zusatz zu einem Personalcomputer oder ein spezieller Hochleistungsprozessor gewünscht ist. In jedem Fall erhält man bei der beschriebenen Auslegung das Maximum an Verarbeitungsleistung, das mit den vorgesehenen Aufwendungen überhaupt zu realisieren möglich ist.
Um die potentielle Eignung von Algorithmen, auf die beschriebene Weise technisch implementiert werden zu können, näher zu charakterisieren, wird der Begriff der "Implementierungseffizienz" e_i wie folgt eingeführt:

$$e_i = \frac{\sum_{i=1}^{n} \mathrm{CARDB}\,(A_i) + \sum_{j=1}^{m} \mathrm{CARDB}\,(R_j)}{z \cdot (\mathrm{ARG\_LINES} + \mathrm{RES\_LINES})} \qquad (3)$$

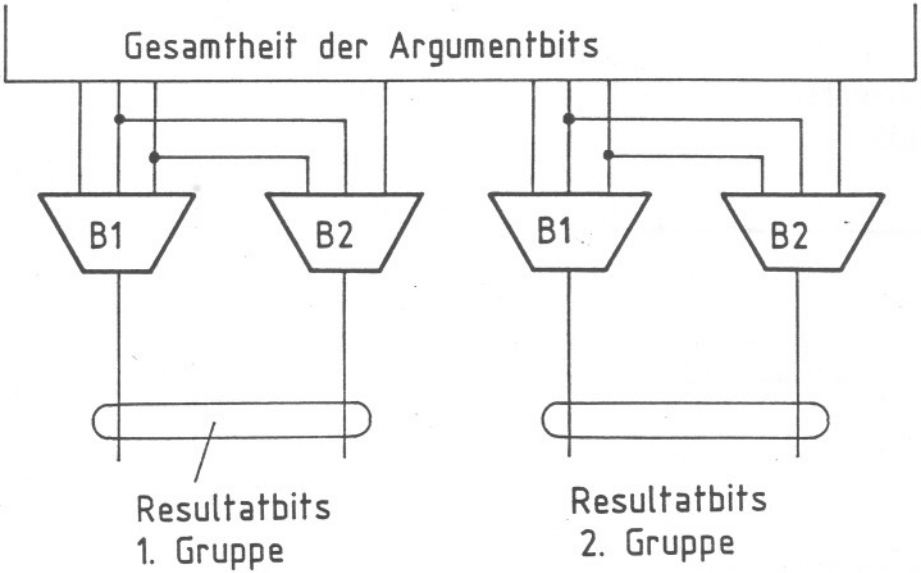
Bedeutung der bisher noch nicht erklärten Symbole:

z: Anzahl der internen Maschinenzustände, die für die Bildung eines Resultat-Abschnittes benötigt werden (Maschinenzyklen; $z \geqslant 1$)

ARG_LINES: Anzahl der Leitungen für die Zuführung der Argument-Abschnitte (bzw. Anzahl der parallel gelesenen Argumentbits)

RES_LINES: Anzahl der Leitungen für den Abtransport der Resultat-Abschnitte (Anzahl der parallel geschriebenen Resultatbits).

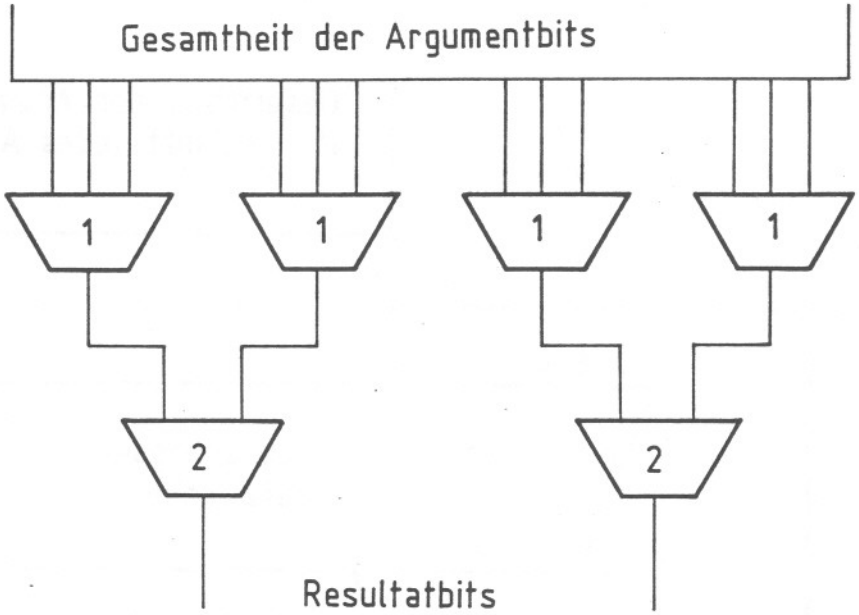
Wird kein Resultat gespeichert (z.B. wenn dieses eine Bedingung repräsentiert, die anderweitig ausgewertet wird), so sind die Resultat-Terme in Zähler und Nenner von (3) gleich Null zu



B1, B2, Netzwerke für jeweils identische BOOLESCHE Gleichungen

Bild 5

Illustration der gruppenweisen Parallelisierbarkeit



1: Netzwerke zur Verknüpfung unabhängiger Gruppen von Argumentbits

2: Netzwerke zur kombinatorischen Zusammenfassung

Bild 4

Illustration der gruppenweisen Zerlegkarbeit

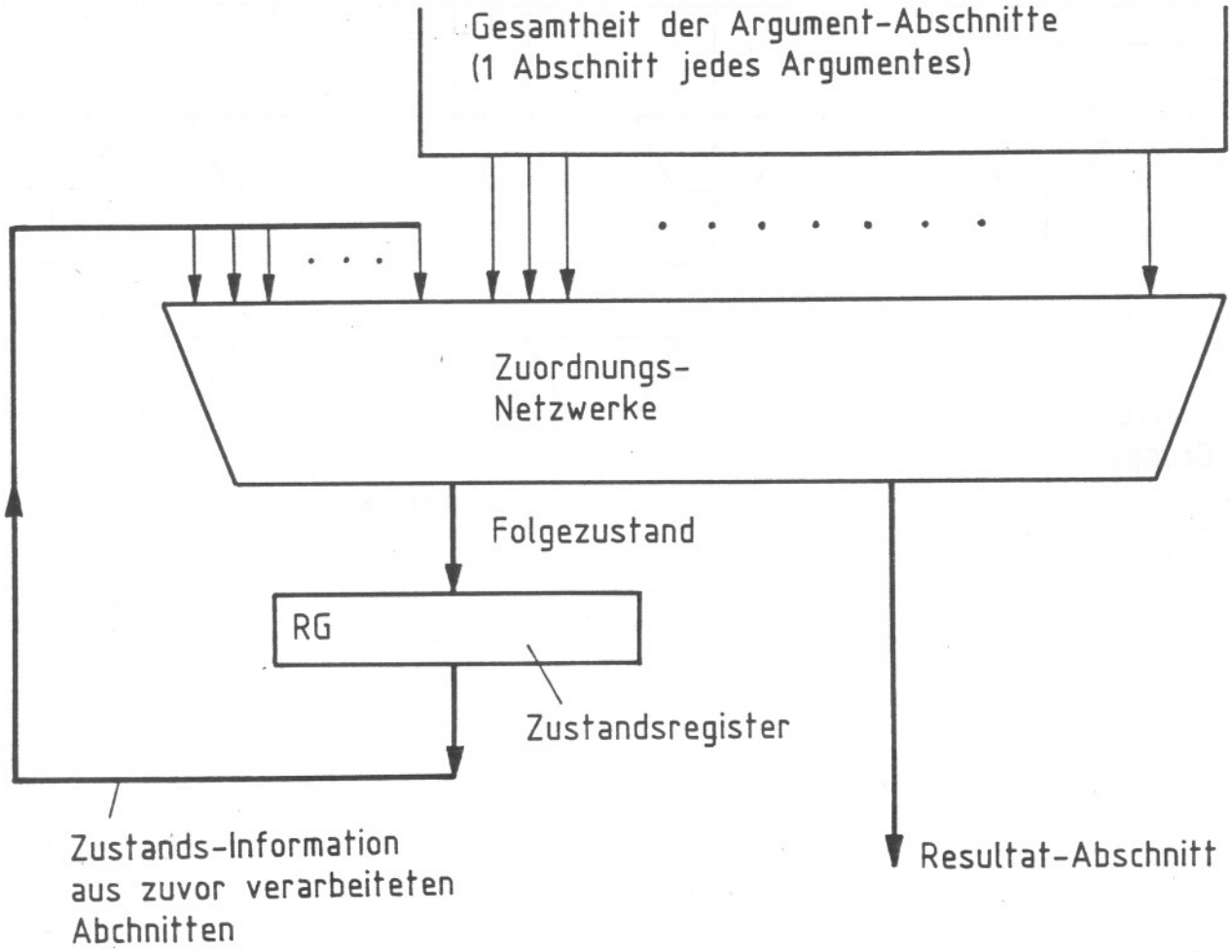


Bild 6

Illustration des Einfließens von Zustands-Information bei der abschnittsweise parallelen Verarbeitung mit $e_i = 1$

setzen. Ist eine technisch gegebene Leitungszahl größer als die Anzahl der Bits des betreffenden Abschnittes, so ist letztere zu verwenden.

Die Implementierungseffizienz ist somit eine dimensionslose Zahl im Intervall $0 < e_i \leq 1$, und es ist ersichtlich, daß im Fall der beschriebenen zweckmäßigsten Implementierung $e_i = 1$ ist.
Damit eine Implementierungseffizienz von Eins erreichbar ist, müssen die folgenden beiden Sätze erfüllt sein:

SATZ 1: Eine Implementierungseffizienz von Eins kann grundsätzlich nur dann erreicht werden, wenn für jeden Abschnitt des Resultats gilt, daß dieser durch Zuordnung (im Sinne von Definition 1) aus den jeweils aktuellen Abschnitten der Argumente erzeugt werden kann, wobei in diese Zuordnung höchstens noch Zustands-Information einbezogen ist, die eindeutig durch die zuvor verarbeiteten Abschnitte bestimmt ist.

SATZ 2: Um eine Implementierungseffizienz von Eins praktisch realisieren zu können, müssen die durch D, SEQ (in (1)) beschriebenen Abläufe so zerlegbar sein, daß nur kombinatorische Verknüpfungen gemeinsam selektierbarer Abschnitte der binären Repräsentationen der Argumente, erforderlichenfalls zusammen mit Zustands-Information im Sinne von Satz 1, jeweils einen Abschnitt des Resultats erzeugen und erforderlichenfalls die besagte Zustands-Information modifizieren.

Satz 1 muß stets erfüllt sein, da nur so gewährleistet ist, daß zur Gewinnung eines Resultatabschnittes keine zusätzliche interne Zustandswechsel erforderlich sind. Solche werden hingegen benötigt, wenn auch nur ein Resultatabschnitt von Argumentabschnitten abhängt, die nicht im aktuellen Zyklus zugänglich sind oder in voraufgegangenen Zyklen bereits verarbeitet wurden, da dann besondere Zugriffe zu diesen ausgeführt werden müßten.
Es ist dabei nicht notwendig, daß ein Resultatabschnitt nur von den aktuellen Argumentabschnitten abhängt: es können auch Abhängigkeiten von zuvor verarbeiteten Abschnitten existieren, die über Zustands-Information vermittelt werden (Bild 6).

Satz 2 charakterisiert die technischen Voraussetzungen. Er hat die Konsequenz, daß $e_i = 1$ nur von bestimmten Verarbeitungsbreiten an realisierbar ist, d.h. nur dann, wenn alle für die Bildung eines Abschnittes der Resultatbits notwendigen Argumentbits parallel zugeführt werden können. Daraus folgt sofort

SATZ 3: Die Implementierungseffizienz von Eins ist bei einer gegebenen Verarbeitungsbreite potentiell stets erreichbar, wenn alle Verknüpfungen im Rahmen dieser Verarbeitungsbreite gruppenweise parallelisierbar (im Sinne von Definition 3) sind.

Die minimal erforderliche Verarbeitungsbreite entspricht dann der Anzahl der Resultatbits einer Gruppe.

Der hier eingeführte Effizienzbegriff reflektiert somit nicht primär eine Eigenschaft der Hardware, sondern eine Eigenschaft des betreffenden Algorithmus, nämlich die, potentiell mit Schaltungsanordnungen implementiert werden zu können, die in jedem ihrer interen Maschinenzyklen einen direkten Beitrag zum Resultat liefern, wobei dieser Beitrag von der jeweiligen Verarbeitungsbreite abhängt, bzw. eine grundsätzliche Leistungsgrenze dann zu besitzen, wenn in beliebigen technisch realisierbaren Hardwarestrukturen prinzipiell mehr als ein interner Zustandswechsel (Maschinenzyklus) notwendig ist, um einen gewissen Beitrag zum Resultat zu ermitteln.
Grundsätzlich ist die Implementierungseffizienz dann kleiner als Eins, wenn

- die abschnittsweise Zuordnung technisch nicht realisierbar ist (Aufwand, Kompliziertheit), so daß eine Folge von Maschinenzyklen erforderlich ist, um einen Abschnitt des Resultats zu erzeugen
- Resultatbits von Argumentbits abhängen, die sich in verschiedenen Abschnitten befinden können (um diese Resultatbits zu erzeugen, sind Zugriffe zu mehreren Abschnitten nötig).

Jeder einigermaßen umfangreiche Algorithmus wird in Teilalgorithmen zerlegbar (bzw. aus solchen aufgebaut) sein. Teilalgorithmen sind Aktionsfolgen, die Teilmengen der Objekte als Argumente haben.

Für derartige Algorithmen seien zwei Sätze angegeben, die Bedingungen beschreiben, unter denen bei Anordnung entsprechender Schaltmittel eine beschleunigte Ausführung möglich ist:

SATZ 4: Ist ein Teilalgorithmus (d.h. eine Aktionsfolge) auf mehrere Teilmengen anzuwenden, wobei diese jeweils unabhängig aus der Menge NA (nach (1)) selektiert werden (d.h. daß die Teilmengen zueinander elementfremd sind) und sind die Teilmengen der erzeugten Resultate ebenfalls zueinander elementfremd, so sind die Teilalgorithmen untereinander parallel ausführbar.

SATZ 5: Ein Algorithmus ist unbeschränkt parallelisierbar, wenn er eine Implementierungseffizienz $e_i = 1$ hat und in Teilalgorithmen zerlegbar ist, die untereinander parallel ausführbar sind.

Genügen Teilalgorithmen eines Algorithmus Satz 4, so können mehrere Hardware-Einrichtungen vorgesehen werden, von denen jede einen Teilalgorithmus ausführt. Die praktischen Entsprechungen von Satz 4 betreffen sowohl die physische Parallelanordnung als auch die zeitsequentielle Mehrfachausnutzung von Schaltmitteln ("pipelining").

Satz 5 bedeutet, daß es für entsprechende Algorithmen keine prinzipielle Leistungsgrenze der Implementierung gibt, d.h. daß bei hinreichendem Aufwand die Bildung des gesamten Resultats in einem einzigen diskreten Zeitabschnitt möglich ist und daß in der Praxis eine Beschleunigung sowohl dadurch erreicht werden kann, daß mehrere Teilalgorithmen parallel ausgeführt werden als auch dadurch, daß für jeden Teilalgorithmus das Prinzip der abschnittsweisen direkten Zuordnung realisiert wird.

2.2. Auswahl der Algorithmen

2.2.1. Datenstrukturen

Argumente und Resultate des Algorithmenkomplexes ("System 0"; vgl. z.B./135/,/156/) werden durch eine überschaubare Anzahl verschiedener Datenstrukturen repräsentiert, so daß deren Beschreibung der Analyse der Algorithmen vorgeordnet werden kann. (Die nachfolgenden Darstellungen betreffen somit die Mengen DA und DR gem. (1).)