KERNFORSCHUNGSANLAGE JÜLICH GmbH

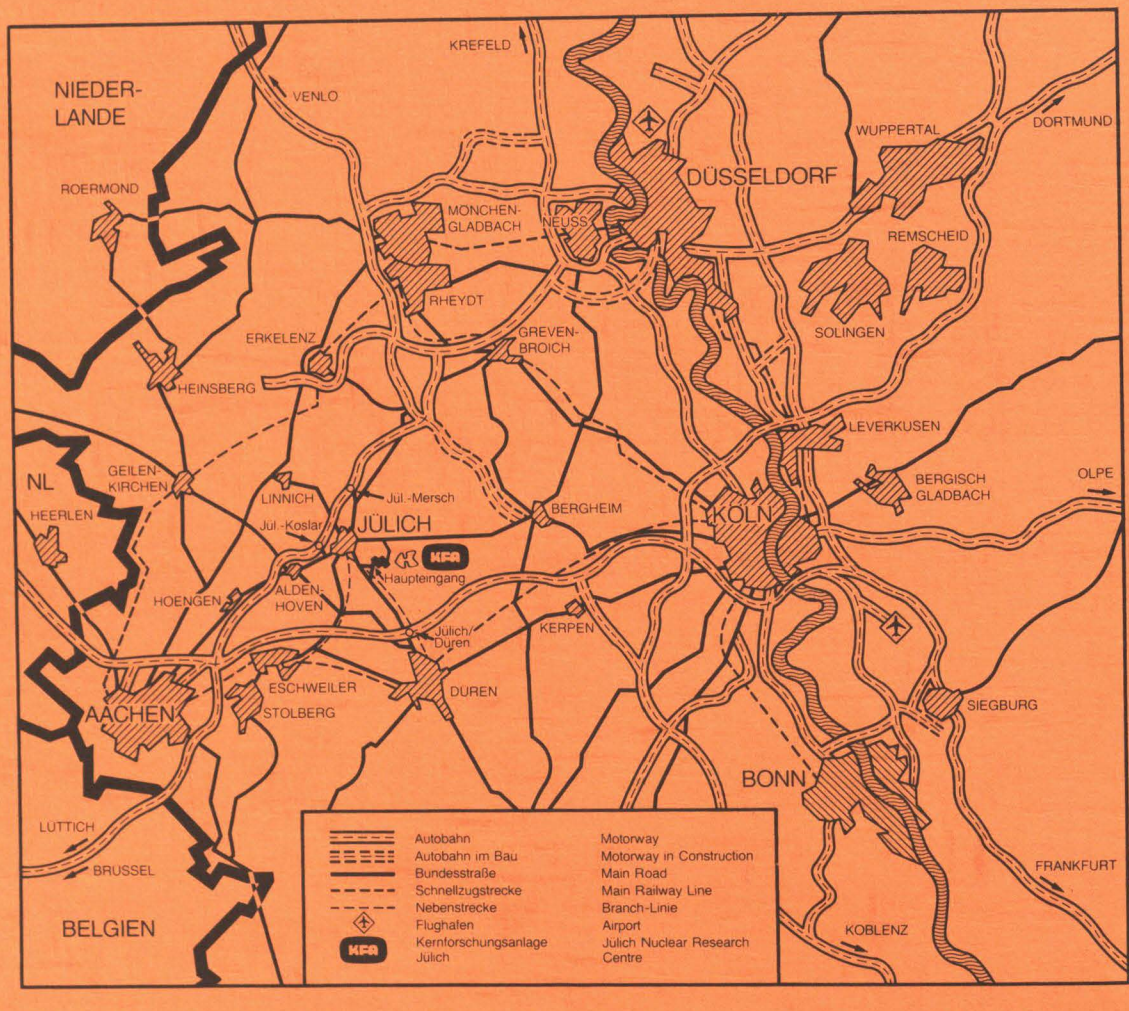
Zentralinstitut für Angewandte Mathematik

# Scheduling – Strategien für Pipeline-Rechner

von

A. Sommer

Jül – Spez – 190
Januar 1983
ISSN 0343 – 7639




Als Manuskript gedruckt

**Spezielle Berichte der Kernforschungsanlage Jülich – Nr. 190**

Zentralinstitut für Angewandte Mathematik Jül – Spez – 190

Zu beziehen durch: ZENTRALBIBLIOTHEK der Kernforschungsanlage Jülich GmbH

Postfach 1913 · D-5170 Jülich (Bundesrepublik Deutschland)

Telefon: 02461/610 · Telex: 833556 kfa d

# Scheduling – Strategien für Pipeline-Rechner

von

A. Sommer

INHALTSVERZEICHNIS

## 1.0 EINLEITUNG

Die Leistung der heutigen nach dem von-Neumann-Prinzip arbeitenden Großrechner reicht trotz der enormen technologischen Entwicklung und der Integration neuer Systemprinzipien (z.B. Überlappung der Ein/Ausgabe durch Verarbeitung in Kanälen, SPOOL-Prinzip, Wechselpuffertechnik, virtuelles Speicherkonzept, Cache, Speicherverschränkung) bei weitem nicht aus, um eine Vielzahl wissenschaftlich-technischer Probleme zu lösen. Da die Leistung bei rechenintensiven Problemen im wesentlichen von den Zykluszeiten des Leitwerks und des Hauptspeichers abhängt, kann sie durch technische Verbesserungen noch gesteigert werden. Der technologischen Entwicklung sind jedoch physikalische Grenzen gesetzt, die bald erreicht sein dürften. In Schätzungen wird eine technologisch bedingte Leistungssteigerung um noch einen Faktor 10 für möglich gehalten. Höhere Leistungen können daher nur durch den Einsatz neuer Rechnerarchitekturen erzielt werden.

Der wesentliche Nachteil des von-Neumann-Rechners liegt in der sequentiellen Verarbeitung von Instruktionen und Daten (single instruction, single data - SISD). Durch Rechnerarchitekturen, in denen Daten (single instruction, multiple data - SIMD) bzw. Instruktionen und Daten (multiple instruction, multiple data - MIMD) parallel verarbeitet werden, kann dieser Engpaß umgangen und eine höhere Leistung erzielt werden.

Eine Methode, Parallelität in einem Rechnersystem zu erzielen, ist das Pipelining. Das Prinzip des Pipelining ist die Segmentierung eines sich wiederholenden, sequentiellen Berechnungsprozesses in Teilprozesse, die auf speziellen, voneinander unabhängigen Hardware-Einheiten (Pipelinesegmenten) ablaufen können. Die Pipelinesegmente können gleichzeitig arbeiten, so daß eine Überlappung aufeinanderfolgender Prozesse möglich ist.

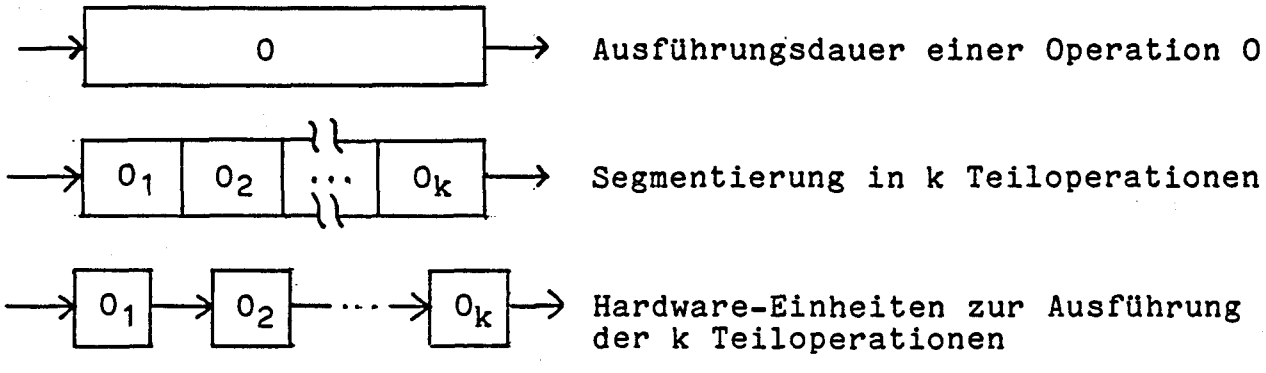
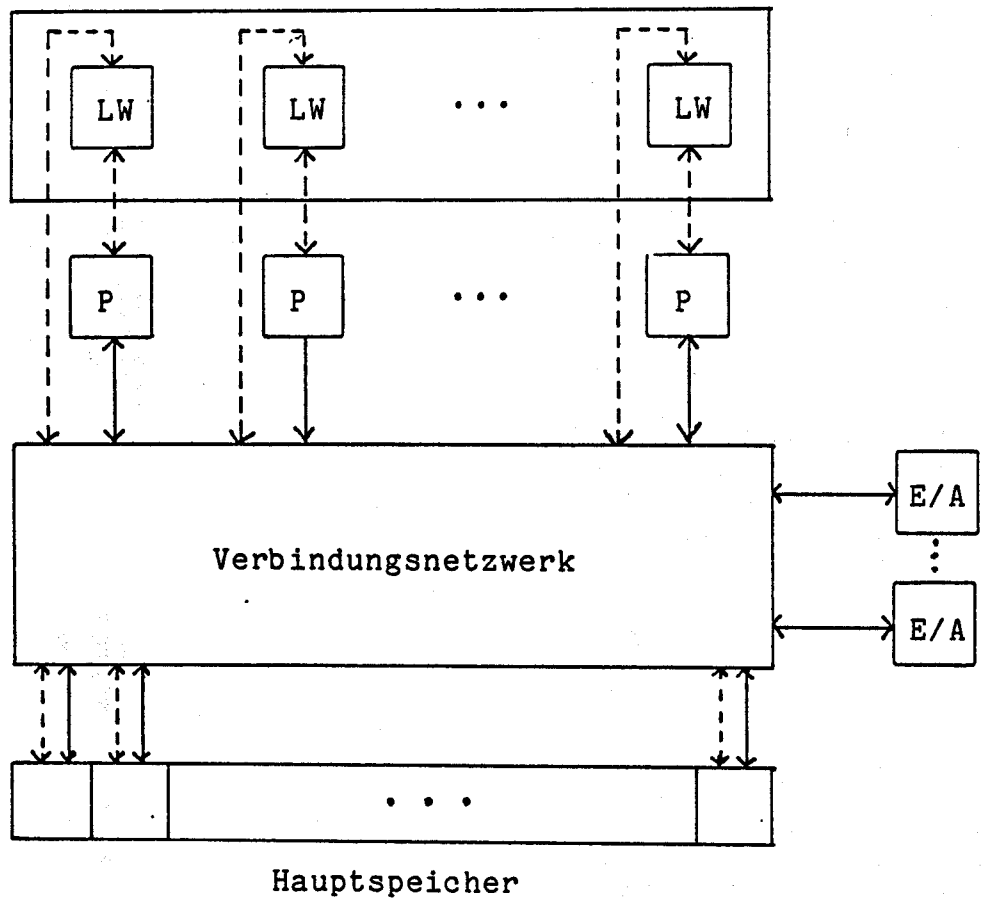



Bild 1.1. Prinzip des Pipelining

Eine weitere Möglichkeit der Parallelverarbeitung bietet ein Parallelprozessorsystem. Ein solches System besteht aus p Prozessoren (P), einem oder p Leitwerken (LW), einem in der Regel in 2p bis 3p Speichereinheiten unterteilten Hauptspeicher, Ein/Ausgabe-Kanälen (E/A) und einem Verbindungsnetzwerk.




⟷ Datenfluß
⇠⇢ Kontrollfluß

Bild 1.2. Struktur eines Parallelprozessorsystems

Der wesentliche Vorteil eines Parallelprozessorsystems liegt darin, daß gleichzeitig auf jedem Prozessor eine Operation ausgeführt werden kann.

Im folgenden Diagramm soll vedeutlicht werden, wie sich die Ergebnisrate r, also die Anzahl der Ergebnisse pro Zeiteinheit $\tau$, bei der Ausführung einer Operation O in einer Pipeline bzw. auf einem Parallelprozessorsystem gegenüber der Ausführung der Operation auf einem von-Neumann-Rechner steigert.

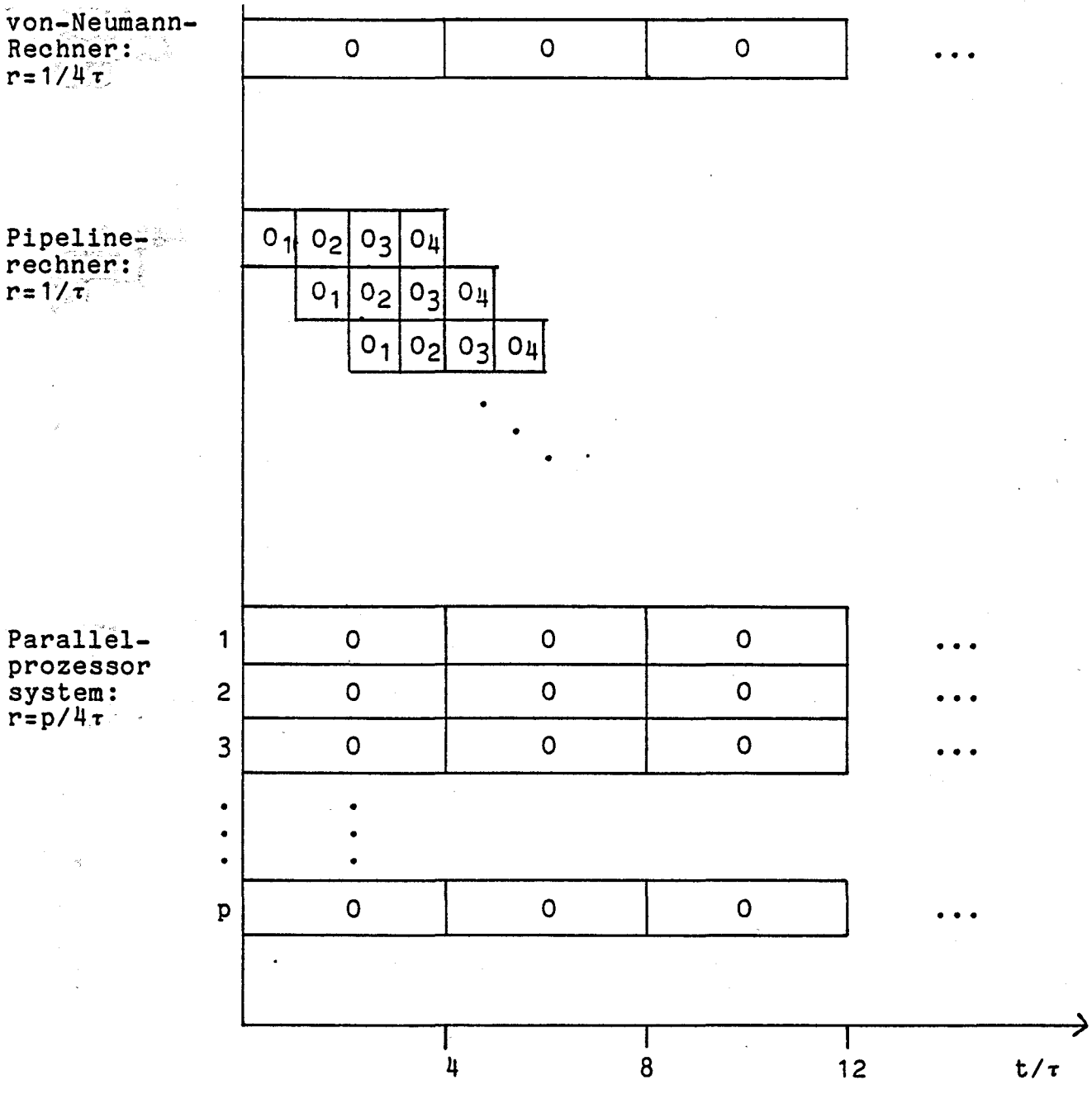



Bild 1.3. Gegenüberstellung von-Neumann-Rechner, Pipelinerechner mit einer Pipeline mit 4 Segmenten und Parallelprozessorsystem mit p Prozessoren bei Ausführung einer Operation O, deren Ausführung sequentiell 4 Zeiteinheiten erfordert.

## 2.0 GRUNDLAGEN DES PIPELINING

### 2.1 INSTRUKTIONS- UND FUNKTIONSPIPELINING

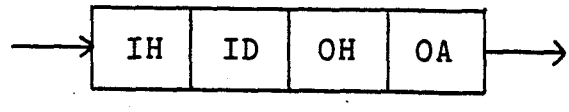
Pipelining kann auf der Ebene der Instruktionverarbeitung und auf der Ebene der Funktionsverarbeitung realisiert werden.

Das Instruktionspipelining sieht die Segmentierung der Verarbeitung einer Instruktion vor. Die Instruktionseinheit des Rechners wird entsprechend als Pipeline organisiert (vgl. /Ra,Li 77/).

Üblicherweise wird die Verarbeitung einer Instruktion in vier Teilprozesse unterteilt:

1. Instruktion holen (IH)
2. Instruktion dekodieren (ID)
3. Operanden holen (OH)
4. Operation ausführen (OA)

Damit läßt sich die Instruktionseinheit wie folgt als Pipeline organisieren:



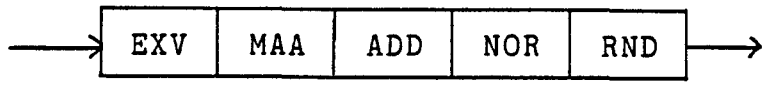

Unter der Annahme, daß jeder Teilprozeß die gleiche Zeit erfordert und die sequentielle Verarbeitung einer Instruktion in vier Prozessorzyklen erfolgt, kann mit Hilfe des Instruktionspipelining nach einer Anlaufzeit von drei Zyklen (Füllen der Pipeline) in jedem Zyklus die Verabeitung einer Instruktion beendet werden.

Pipelining auf der Funktionsebene, das ausschließlich Gegenstand der weiteren Überlegungen sein soll, bezieht sich auf die Ausführung von Operationen wie z.B. der Addition oder Multiplikation in als Pipeline organisierten Funktionseinheiten.

Die Ausführung der Addition läßt sich zum Beispiel in folgende fünf Teilprozesse zerlegen:

1. Vergleich der Exponenten (EXV)
2. Angleichen der Mantissen (MAA)
3. Addition der Mantissen (ADD)
4. Normalisieren (NOR)
5. Runden (RND)

Die dazugehörige Additionspipeline sieht wie folgt aus:




Unter der Annahme, daß jeder Teilprozeß die gleiche Ausführungszeit (jeweils eine Zeiteinheit) erfordert, läßt sich die Arbeitsweise dieser Additionspipeline in einem Raum-Zeit Diagramm wie folgt darstellen:

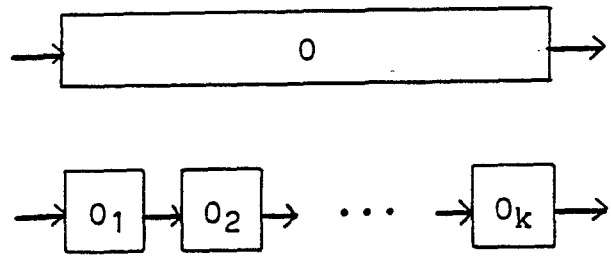
| | 0 | 1 | 2 | 3 | 4 | | N-1 | N | N+1 | N+2 | N+3 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| EXV | 1 | 2 | 3 | 4 | 5 | | N | | | | |
| MSH | | 1 | 2 | 3 | 4 | | N-1 | N | | | |
| ADD | | | 1 | 2 | 3 | | N-2 | N-1 | N | | |
| NOR | | | | 1 | 2 | | N-3 | N-2 | N-1 | N | |
| RND | | | | | 1 | | N-4 | N-3 | N-2 | N-1 | N |

$t/\tau$

Bild 2.1. Raum-Zeit Diagramm zur Addition zweier Vektoren der Länge N in der Additionspipeline

## 2.2 PIPELINEKONZEPT

O sei eine Operation, die sich in die Teiloperationen $O_1$, $O_2$, ... $O_k$ zerlegen läßt:




Für die Ausführung dieser Operation in einer Pipeline gelten folgende Eigenschaften:

1. Die Ausführung von O ist äquivalent zur sequentiellen Ausführung der Teiloperationen $O_1$, $O_2$, ..., $O_k$.

2. Die Operanden einer Teiloperation $O_i$, $2 \leq i \leq k$, sind die Resultate der Teiloperation $O_{i-1}$ in der Ausführungssequenz.

3. Außer der Weitergabe der Resultate an die jeweils nächste Teiloperation gibt es keine Verbindung zwischen den einzelnen Teiloperationen.

4. Zur Ausführung jeder Teiloperation steht jeweils ein Hardwaremodul zur Verfügung.

5. Zur Synchronisation werden zwischen zwei Moduln ein oder mehrere Speicherelemente (latches) eingefügt, welche die Ausgaben des davorliegenden Moduls aufnehmen und beim Durchschalten eines Taktimpulses in den dahinterliegenden Modul weitergeben (vgl. /Ko 81/). Es ist zwar auch ein asynchroner Pipelinebetrieb denkbar, doch erfordert dieser einen erheblich größeren Kontrollaufwand bei der Realisierung einer überlappten Verarbeitung (siehe 7.). Zu beachten ist jedoch, daß die Verarbeitungszeit für eine Operation in einer synchronen Pipeline durch die Verweilzeit der Daten in den Speicherelementen höher ist als im asynchronen Fall.

6. Als Pipelinesegment soll im weiteren ein Modul mit den unmittelbar dahinterliegenden Speicherelementen bezeichnet werden. Die Ausführungszeit eines Pipelinesegments, also die Zeit zwischen der Eingabe der Daten in das Segment und der Weitergabe der Resultate an das nächste Segment, ist durch die Synchronisation ein Vielfaches der Zeiteinheit $\tau$ zwischen zwei Taktimpulsen des Systems.

Die Ergebnisrate einer Pipeline, welche angibt, wieviele Ergebnisse durchnittlich pro Zeiteinheit geliefert werden, hängt von dem Pipelinesegment mit der höchsten Ausführungszeit ab (vgl. /Ra,Li 77/):

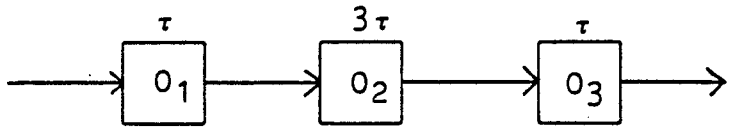



Bild 2.2.1. Pipeline mit 3 Segmenten der Ausführungszeiten $\tau$, $3\tau$ und $\tau$

Zum Beispiel beträgt die Ergebnisrate der Pipeline im oberen Bild maximal $1/3\tau$, obwohl das erste und das dritte Pipelinesegment in jeder Zeiteinheit ein Ergebnis liefern können. Der durch das zweite Pipelinesegment verursachte Engpaß kann durch eine Parallelschaltung mehrerer gleicher Pipelinesegmente oder eine weitere Segmentierung des zweiten Pipelinesegments umgangen werden.

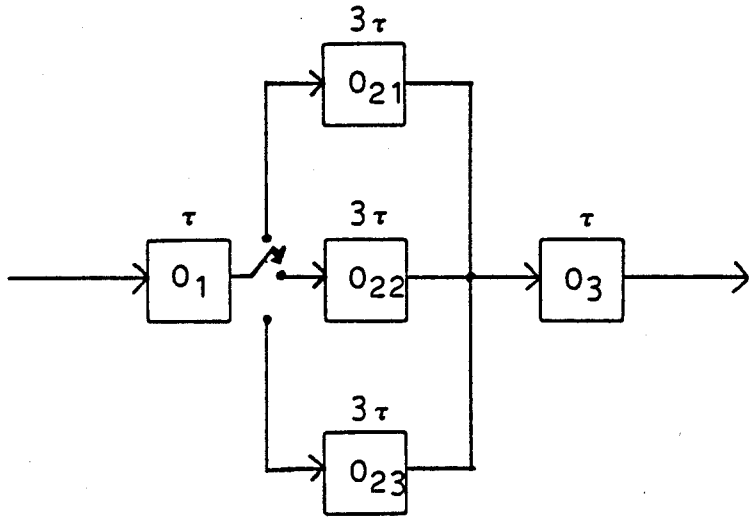



Bild 2.2.2. Parallelschaltung mehrerer gleicher Pipelinesegmente

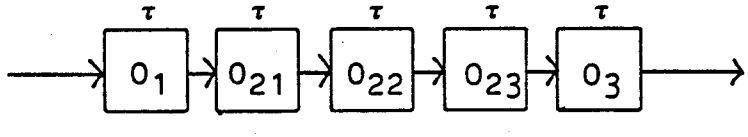



Bild 2.2.3. Weitere Segmentierung des zweiten Pipelinesegments

In beiden Fällen beträgt die maximale Ergebnisrate $1/\tau$, jedoch ist bei der Parallelschaltung der Hardware-Aufwand durch Verteilungsprobleme größer.

Für die weiteren Überlegungen soll vorausgesetzt werden, daß die Ausführungszeiten der verschiedenen Pipelinesegmente gleich sind und jeweils eine Zeiteinheit $\tau$ betragen.

Die Anzahl k der Pipelinesegmente liegt in der Praxis normalerweise zwischen 2 und 8.

7. Bei jedem Taktimpuls, der gleichzeitig an alle Pipelinesegmente erteilt wird, können Daten in ein Segment eingegeben, bis zum nächsten Taktimpuls darin verarbeitet und die Resultate in die Speicherelemente des Segments abgelegt werden. Dadurch ist es möglich, daß bei jedem Taktimpuls eine Eingabe in die Pipeline erfolgen und die Ausführung einer Operation beginnen kann. Während in einem Segment i die i-te Teiloperation für die p-te Ausführung in der Pipeline durchgeführt wird, kann zur gleichen Zeiteinheit das (i-1)-te Segment die (i-1)-te Teiloperation für die (p+1)-te Ausführung durchführen. Ein Beispiel für die überlappte Verarbeitung in einer Pipeline läßt sich aus dem Raum-Zeit Diagramm in Bild 2.1. entnehmen.

8. Durch die überlappte Verarbeitung in einer Pipeline läßt sich gegenüber der sequentiellen Verarbeitung auf einem Monoprozessor eine Geschwindigkeitssteigerung (Speedup) erzielen. Als Speedup $S_k$ wird das Verhältnis der Zeit $T_1$ für die n-fache Hintereinanderausführung einer Operation auf einem Monoprozessor zur Zeit $T_k$ n-fache Ausführung bei überlappter Verarbeitung in einer Pipeline mit k Segmenten bezeichnet.

$$S_k := T_1/T_k$$

$T_k$ setzt sich aus der Zeit zum Füllen der Pipeline (Aufsetzzeit, set up time) und der Zeit, in der bei jedem Taktimpuls ein Ergebnis einer Operation aus der Pipeline ausgegeben wird, zusammen:

$$T_k = (k-1+N)\tau$$

Dieser Zusammenhang soll im folgenden Bild noch einmal für die Additionspipeline aus Abschnitt 2.1. verdeutlicht werden:

| | 0 | 1 | 2 | 3 | 4 | | N-1 | N | N+1 | N+2 | N+3 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| EXV | 1 | 2 | 3 | 4 | 5 | | N | | | | |
| MSH | | 1 | 2 | 3 | 4 | | N-1 | N | | | |
| ADD | | | 1 | 2 | 3 | | N-2 | N-1 | N | | |
| NOR | | | | 1 | 2 | | N-3 | N-2 | N-1 | N | |
| RND | | | | | 1 | | N-4 | N-3 | N-2 | N-1 | N |

4τ (Aufsetzzeit) Nτ (1Resultat/Takt)

Gesamtausführungszeit: $T_k = (k-1+N)\tau = (N+4)\tau$

Bild 2.2.4. Ausführungszeit bei überlappter Verarbeitung in der Additionspipeline aus Abschnitt 2.1.

Unter der Voraussetzung, daß $T_1=T_k$ für k=1 ist, wobei die Zeit, die zusätzlich durch die Synchronisierung in der Pipeline entsteht, vernachlässigt wird, gilt für N Ausführungen auf dem Monoprozessor:

$$T_1 = (N \cdot k)\tau$$

Damit gilt für den Speedup $S_k$:

$$S_k = T_1/T_k = N \cdot k/(k-1+N)$$

Daraus folgt: $1 \leq S_k \leq k$

9. Als Effizienz $E_k$ einer Pipeline mit k Segmenten wird der Quotient aus dem Speedup $S_k$ und der Anzahl k bezeichnet:

$$E_k := S_k/k = N/(k-1+N) \leq 1$$

## 2.3 ERWEITERTE PIPELINEKONZEPTE

### 2.3.1 ALLGEMEINE ERWEITERUNGEN

Gegenüber dem in Abschnitt 2.2. vorgestellten Pipelinekonzept, das auch als lineares Pipelinekonzept bezeichnet wird, da die einzelnen Pipelinesegmente zur Ausführung einer Operation nacheinander nur jeweils einmal aktiv sind, sollen folgende Erweiterungen getroffen werden:

1. Es ist möglich, daß die Ausgaben eines Pipelinesegments bei einem Taktimpuls an mehrere Pipelinesegmente weitegegeben werden.

2. Es ist möglich, daß die Ausgaben eines Pipelinesegments an frühere, daß heißt in der Ausführungssequenz davorliegende Pipelinesegmente, zurückgeführt werden (feedback).

3. Zur gleichen Zeiteinheit können bei der Ausführung einer Operation mehrere Pipelinesegmente aktiv sein. Das bedeutet, daß mehrere Teiloperationen parallel ausgeführt werden.

4. Ein Pipelinesegment kann zu mehreren Zeiteinheiten aktiv sein. Das bedeutet, daß die entsprechende Teiloperation mehrfach bei der Ausführung einer Operation in der Pipeline durchgeführt wird.

### 2.3.2 EIN- UND MEHRFUNKTIONALE PIPELINES

Eine Pipeline ist einfunktional, wenn mit ihr nur eine Operationsart ausgeführt werden kann. Beispiel für eine einfunktionale Pipeline ist die in Abschnitt 2.1. angegebene Additionspipeline.

Einer einfunktionalen Pipeline liegt ein statisches Pipelinekonzept zugrunde. Das bedeutet, daß jede Ausführung einer Operation den gleichen Ablauf von Aktivitäten in der Pipeline erfordert (vgl. /Ra,Li 77/).

Eine Pipeline ist mehrfunktional, wenn in ihr die Ausführung mehrerer verschiedener Operationen, z.B. Addition und Multiplikation, möglich ist. Mehrfunktionalen Pipelines liegt entweder ein statisches oder ein dynamisches Pipelinekonzept zugrunde (vgl. /Ra,Li 77/).

In einer statischen mehrfunktionalen Pipeline sind zu einer gegebenen Zeit nur Ausführungen von Operationen der gleichen Operationsart möglich. Erst nach Beendigung dieser Ausführungen kann die Operationsart gewechselt werden. Statische mehrfunktionale Pipelines arbeiten also zwischen den Operationswechseln wie einfunktionale Pipelines.

Dynamische mehrfunktionale Pipelines lassen die überlappte Verarbeitung von Operationen verschiedener Operationsarten zu. Dadurch läßt sich die Ergebnisrate zwar gegenüber statischen mehrfunktionalen Pipelines weiter steigern, jedoch ist die Kontrolle einer dynamischen Pipeline aufwendiger.

Bemerkung:

Es wird weiterhin gefordert, daß zu Beginn der Berechnungen in einer statischen oder dynamischen Pipeline jeweils genau das Muster über den Ablauf der Aktivitäten in der Pipeline für jede Ausführung vorliegt. Es darf keine Abhängigkeiten von Pipelineaktivitäten von irgendwelchen Resultaten oder Zwischenresultaten geben.

### 2.3.3 ASYNCHRONES PIPELINEKONZEPT

Ein asynchrones Pipelinekonzept besitzt folgende Merkmale:

1. Es kann Abhängigkeiten zwischen verschiedenen Ausführungen von Operationen in der Pipeline geben.

2. Der Ablauf der Aktivitäten zur Ausführung einer Operation in der Pipeline liegt zu Beginn der Ausführung nicht unbedingt fest, sondern kann sich während der Ausführung in Abhängigkeit von irgendwelchen Zwischenresultaten oder Statusbedingungen ändern.

3. Die Ausführungszeiten der verschiedenen Hardware-Moduln hängen von den Eingaben und der Hardware ab und sind daher im allgemeinenen verschieden (vgl. auch Abschnitt 2.2. Punkt 5).

Wegen der Notwendigkeit sehr komplizierter Kontroll- und Synchronisierungsmechanismen läßt sich ein asynchrones Pipelinekonzept nur schwer realisieren. Im folgenden soll auf ein asynchrones Pipelinekonzept nicht weiter eingegangen werden.

## 2.4 HILFSMITTEL UND BEGRIFFE ZUR BESCHREIBUNG UND ANALYSE DER VORGÄNGE IN EINER PIPELINE

### 2.4.1 DIE RESERVIERUNGSTAFEL

Eine Reservierungstafel ist eine Tabelle, die das Muster über den den Ablauf der Aktivitäten zur Ausführung einer Operation in einer Pipeline wiedergibt. In dieser Tabelle sind horizontal die Zeiteinheiten und vertikal die Pipelinesegmente aufgetragen. Eine Eine durch einen Großbuchstaben gekennzeichnete Markierung auf der Position (i,j) der Reservierungstafel bedeute, daß das Segment i zur Zeiteinheit j aktiv ist, also eine Teiloperation ausführt (vgl. /Ko 81/).

Die Ausführungszeit einer Reservierungstafel ist die Zeit, die zur Ausführung einer Operation in der Pipeline gemäß der Reservierungstafel erforderlich ist. Sie beträgt m Zeiteinheiten, wenn m die Anzahl der in der Reservierungstafel horizontal aufgetragenen Zeiteinheiten ist.

Es ist möglich, daß in einer Zeile oder Spalte einer Reservierungstafel mehrere Markierungen auftreten.

Mehrere Markierungen in einer Zeile bedeuten, daß das dazugehörige Pipelinesegment für die Ausführung einer Operation zu mehreren Zeiteinheiten aktiv ist, also seine Teiloperation mehrfach ausführt.

Die Anzahl der Markierungen in der i-ten Zeile der Reservierungstafel und damit die Anzahl der Pipelineaktivitäten des i-ten Segments für eine Ausführung gemäß der Reservierungstafel wird im folgenden mit W(i) bezeichnet.

Mehrere Markierungen in einer Spalte bedeuten, daß mehrere Pipelinesegmente zur gleichen Zeiteinheit aktiv sind, also parallel arbeiten.

Im folgenden werden einige Beispiele für Reservierungstafeln gegeben.

Beispiel 2.4.1.

a) Reservierungstafel für eine Pipeline mit 4 Segmenten (k=4):

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1 | A | A | | | | A | | |
| 2 | | | | A | A | | | |
| 3 | | A | A | | | | A | |
| 4 | A | | | | | | | A |

Ausführungszeit: $8\tau$ (m=8)
W(1)=W(3)=3, W(2)=W(4)=2

b) Reservierungstafel für eine Pipeline mit 4 Segmenten (k=4):

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 1 | | | B | B | |
| 2 | B | B | | | |
| 3 | B | | | | B |
| 4 | | | | B | |

Ausführungszeit: $5\tau$ (m=5)
W(1)=W(2)=W(3)=2, W(4)=1

c) Reservierungstafel für die lineare Additionspipeline aus Abschnitt 2.1. mit 5 Segmenten (k=5):

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 1 | C | | | | |
| 2 | | C | | | |
| 3 | | | C | | |
| 4 | | | | C | |
| 5 | | | | | C |

Ausführungszeit: $5\tau$ (m=5)
W(i)=1 für i=1(1)5

## 2.4.2 INITIIERUNG, KOLLISION, VERZÖGERUNG

Die Initiierung einer Reservierungstafel erfolgt, wenn nach deren Muster eine Ausführung in der Pipeline gestartet wird.

In einer statischen Pipeline sind alle Initiierungen von derselben Reservierungstafel, in einer dynamischen Pipeline können verschiedene Reservierungstafeln initiiert werden.

Eine Kollision tritt auf, wenn durch zwei oder mehr Initiierungen versucht wird, zur gleichen Zeiteinheit dasselbe Pipelinesegment zu benutzen.

Eine Initiierung heißt zulässig, wenn durch sie keine Kollisionen mit vorherigen Initiierungen verursacht werden.

Als Verzögerung l (latency) wird die Anzahl der Zeiteinheiten zwischen zwei Initiierungen bezeichnet. l ist eine nicht-negative ganze Zahl. Der Fall l=0 kann nur in dynamischen Pipelines auftreten und bedeuted die Initiierungen von zwei verschiedenen Reservierungstafeln zur gleichen Zeiteinheit.

Eine Verzögerungsfolge $\langle l_1, l_2, \ldots, l_N \rangle$ enthält die Verzögerungen entsprechend der Reihenfolge der Initiierungen bei N+1 Ausführungen in der Pipeline.

Eine Verzögerungsfolge heißt zulässig, wenn im Verlauf der Initiierungen gemäß dieser Verzögerungsfolge keine Kollisionen in der Pipeline auftreten.

Ein Verzögerungszyklus $(l_1', l_2', \ldots, l_p')$, $p \geq 1$, tritt auf, wenn eine Verzögerungsfolge eine sich beliebig oft wiederholenden Teilfolge $\langle l_1', l_2', \ldots, l_p' \rangle$ enthält.

Die Umlaufdauer T eines Verzögerungszyklus ist, gemessen in Zeiteinheiten, gleich der Summe der Verzögerungen der sich wiederholenden Teilfolge. Es gilt also:

$$T = \left( \sum_{i=1}^{p} l_i' \right) \tau$$

Die Summe der Verzögerungen $T/\tau$ der sich wiederholenden Teilfolge wird im weiteren als Umlaufzahl bezeichnet.

Die Initiierungsrate I gibt die durchschnittliche Anzahl von Initiierungen pro Zeiteinheit an.

Die mittlere Verzögerung L=1/I gibt die durchschnittliche Anzahl von Zeiteinheiten zwischen zwei Initiierungen an.

Die mittlere Verzögerung L einer Verzögerungsfolge $\langle l_1, l_2, \ldots l_N \rangle$ beträgt:

$$L = \left( \sum_{i=1}^{N} l_i \right) / N$$

Die mittlere Verzögerung L' eines Verzögerungszyklus ($l_1'$, $l_2'$, ... , $l_p'$) beträgt:

$$L' = \left( \sum_{i=1}^{p} l_i' \right) / p$$

Bemerkung:

Die mittlere Verzögerung einer statischen Pipeline ist bei überlappter Verarbeitung immer kleiner oder gleich m, wenn m die Ausführungszeit der Reservierungstafel ist.:

$$L \leq m$$

Beweis: Für eine statische Pipeline mit einer Verzögerungsfolge $\langle l_1, \ldots, l_N \rangle$ gilt $l_i \leq m$ für alle i, $1 \leq i \leq N$, und damit:

$$N \cdot L = \sum_{i=1}^{N} l_i \leq N \cdot m$$

Im folgenden sollen die bisher in diesem Abschnitt eingeführten Begriffe mit Hilfe von Beispielen verdeutlicht werden.

Beispiel 2.4.2.

a) Aktivitäten in einer statischen Pipeline gemäß der Reservierungstafel aus Beispiel 2.4.1.a). Die Aktivitäten der Pipelinesegmente für die erste Ausführung gemäß der Reservierungstafel werden dabei jeweils mit $A_1$, diejenigen für die zweite Ausführung mit $A_2$ usw. bezeichnet.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $A_1$ | $A_1$ | $A_2$ | $A_2$ | | $A_1$ | | $A_2$ | $A_3$ | $A_3$ | $A_4$ |
| 2 | | | | $A_1$ | $A_1$ | $A_2$ | $A_2$ | | | | |
| 3 | | $A_1$ | $A_1$ | $A_2$ | $A_2$ | | $A_1$ | | $A_2$ | $A_3$ | $A_3$ |
| 4 | $A_1$ | | $A_2$ | | | | | $A_1$ | $A_3$ | $A_2$ | $A_4$ |

| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $A_4$ | | $A_3$ | | $A_4$ | $A_5$ | $A_5$ | $A_6$ | $A_6$ | | |
| 2 | $A_3$ | $A_3$ | $A_4$ | $A_4$ | | | | | $A_5$ | $A_5$ | |
| 3 | $A_4$ | $A_4$ | | $A_3$ | | $A_4$ | $A_5$ | $A_5$ | $A_6$ | $A_6$ | |
| 4 | | | | | $A_3$ | $A_5$ | $A_4$ | $A_6$ | | | |

Initiierungen zu den Zeiteinheiten 0, 2, 8, 10, 16, 18, ...
Verzögerungsfolge : <2,6,2,6,...>
Verzögerungszyklus : (2,6)
Umlaufzahl eines Zyklus : 8
Initiierungsrate : 1/4
mittlere Verzögerung : 4

b) Aktivitäten in einer dynamischen Pipeline mit Initiierungen gemäß den Reservierungstafeln aus Beispiel 2.4.1.a) und b). Zu beachten ist, daß in einer dynamischen Pipeline eine überlappte Verarbeitung von Operationen gemäß verschiedener Reservierungstafeln stattfinden kann. Im Beispiel werden die Aktivitäten für die erste Ausführung gemäß der Reservierungstafel aus Beispiel 2.4.1.a) mit $A_1$, diejenigen für die zweite Ausführung gemäß dieser Reservierungstafel mit $A_2$ usw. bezeichnet. Entsprechend werden die Aktivitäten für die Ausführungen gemäß der Reservierungstafel aus Beispiel 2.4.1.b) mit $B_1$, $B_2$, usw. bezeichnet.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $A_1$ | $A_1$ | $B_1$ | $B_1$ | | $A_1$ | | $B_2$ | $B_2$ | | $A_2$ |
| 2 | $B_1$ | $B_1$ | | $A_1$ | $A_1$ | $B_2$ | $B_2$ | | | | $B_3$ |
| 3 | $B_1$ | $A_1$ | $A_1$ | | $B_1$ | $B_2$ | $A_1$ | | | $B_2$ | $B_3$ |
| 4 | $A_1$ | | | $B_1$ | | | | $A_1$ | $B_2$ | | $A_2$ |

| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $A_2$ | $B_3$ | $B_3$ | | $A_2$ | | $B_4$ | $B_4$ | | $A_3$ | | |
| 2 | $B_3$ | | $A_2$ | $A_2$ | $B_4$ | $B_4$ | | | | $B_5$ | | |
| 3 | $A_2$ | $A_2$ | | $B_3$ | $B_4$ | $A_2$ | | | $B_4$ | $B_5$ | | |
| 4 | | | $B_3$ | | | | $A_2$ | $B_4$ | | $A_3$ | | |

Initiierungen zu den Zeiteinheiten 0, 0, 5, 10, 10, 15, ...
Verzögerungsfolge : <0,5,5,0,5,5,...>
Verzögerungszyklus : (0,5,5)
Umlaufzahl eines Zyklus : 10
Initiierungsrate : 3/10
mittlere Verzögerung : 10/3

c) Für die in Abschnitt 2.1. angegebene Additionspipeline gilt:

Initiierungen zu den Zeiteinheiten 0, 1, 2, 3, ...
Verzögerungsfolge : <1,1,1,...>
Verzögerungszyklus : (1)
Umlaufzahl eines Zyklus : 1
Initiierungsrate : 1
mittlere Verzögerung : 1

## 2.4.3 SEGMENTAUSLASTUNG

Die Segmentauslastung U(i) (Utilization) gibt an, wie oft das Pipelinesegment i durchschnittlich zwischen zwei Initiierungen aktiv ist. Sie ist gleich dem Produkt aus der Initiierungsrate I und der durchschnittlichen Anzahl W(i) von Aktivitäten des i-ten Segments für eine Ausführung in der Pipeline.

Sei N die Anzahl der Initiierungen, $W_j(i)$ die Anzahl der Aktivitäten des i-ten Segments für die j-te Ausführung, j=1(1)N. Dann folgt:

$$\overline{W}(i) := \sum_{j=1}^{N} W_j(i)/N$$

Damit gilt:

$$U(i) = I \cdot \overline{W}(i) = \overline{W}(i)/L$$

Bemerkungen:

1. Für statische Pipelines gilt: $\overline{W}(i) = W(i)$.

2. Da die Auslastung eines Pipelinesegments nicht mehr als 100% betragen kann, gilt:

$$U(i) \leq 1 \text{ für } i=1(1)k$$

Beispiel 2.4.3.

a) Für die statische Pipeline aus Beispiel 2.4.2.a) gilt:
U(1)=U(3)=3/4, U(2)=U(4)=1/2

b) Für die dynamische Pipeline aus Beispiel 2.4.2.b) gilt:
U(1)=U(3)=7/10, U(2)=6/10, U(4)=4/10

c) Für die Additionspipeline aus Abschnitt 2.1. gilt:
U(i)=1 für i=1(1)5

### 2.4.4 KLEINSTE ERREICHBARE MITTLERE VERZÖGERUNG

Als kleinste erreichbare mittlere Verzögerung MAL (Minimum Achievable Latency) wird das Minimum aller mittleren Verzögerungen L(i) aller zulässigen Verzögerungsfolgen bezeichnet:

$$MAL = MIN(L(i))$$

Lemma (Shar 1972):

Für eine statische Pipeline ist die kleinste erreichbare mittlere Verzögerung immer größer oder gleich der maximalen Anzahl von Markierungen in einer Zeile der Reservierungstafel der Pipeline:

$$MAL \geq MAX(W(i))$$

Beweis: Bei einer Initiierungsrate I und W(i) Markierungen in der i-ten Zeile der Reservierungstafel gilt für die maximale Segmentauslastung:

$$MAX(U(i)) = I \cdot MAX(W(i)) \leq 1$$

$$I \leq 1/MAX(W(i))$$

$$L \geq MAX(W(i))$$

$$MAL \geq MAX(W(i))$$

Zu beachten ist, daß nicht notwendig gilt: MAL = MAX(W(i)). Durch Kollisionen wird diese untere Grenze für die kleinste erreichbare mittlere Verzögerung und damit eine hundertprozentige Auslastung eines Pipelinesegments nicht immer erreicht.

Für dynamische Pipelines mit k Segmenten gilt: $MAL \geq 1/k$

Eine kleinere mittlere Verzögerung ist nicht möglich, da zu jeder Zeiteinheit höchstens k Initiierungen verschiedener Reservierungstafeln erfolgen können.

## 2.4.5 AUSFÜHRUNGSZEIT STATISCHER UND DYNAMISCHER PIPELINES BEI SEQUENTIELLER UND ÜBERLAPPTER VERARBEITUNG

Sequentielle Verarbeitung in einer Pipeline bedeutet, daß die Initiierung einer Reservierungstafel erst dann erfolgt, wenn die laufende Ausführung in der Pipeline gemäß der letzten Initiierung beendet ist.

Überlappte Verarbeitung in einer Pipeline bedeutet, daß die Initiierung einer Reservierungstafel erfolgen kann, bevor die Ausführungen in der Pipeline gemäß vorherigen Initiierungen beendet sind. Voraussetzung für eine Initiierung ist jedoch, daß sie zulässig ist, also keine Kollisionen verursacht.

Im folgenden wird mit $T_s$ die Zeit für N Ausführungen bei sequentieller Verarbeitung, mit $T_k$ die Zeit für N Ausführungen bei überlappter Verarbeitung in einer Pipeline mit k Segmenten bezeichnet.

Für eine statische Pipeline mit einer Reservierungstafel, deren Ausführungszeit m Zeiteinheiten beträgt, gilt bei einer Verzögerungsfolge $\langle l_1, l_2, \ldots, l_{N-1} \rangle$ :

$$T_s = (N \cdot m)\tau$$

$$T_k = ((\sum_{i=1}^{N-1} l_i) + m)\tau = (L \cdot (N-1) + m)\tau$$

Für eine dynamische Pipeline mit Ausführungen gemäß Reservierungstafeln mit den Ausführungszeiten $m_1\tau$, $m_2\tau$, ... , $m_N\tau$ gilt:

$$T_s = (\sum_{i=1}^{N} m_i)\tau$$

bzw. mit $$\bar{m} := (\sum_{i=1}^{N} m_i)/N$$

$$T_s = (N \cdot \bar{m})\tau$$

$$T_k = MAX(m_1, l_1+m_2, l_1+l_2+m_3, \ldots, \sum_{i=1}^{N-1} l_i + m_N)\tau$$

bzw. mit $l_0 := 0$

$$T_k = \max_j((\sum_{i=0}^{j} l_i) + m_{j+1})\tau, \quad 0 \le j \le N-1$$

## 2.4.6 BETRACHTUNGEN FÜR ÜBERLAPPTE VERARBEITUNG BEI PIPELINES MIT VERZÖGERUNGSZYKLUS

Der zeitliche Ablauf der Aktivitäten in einer Pipeline, in der ein Verzögerungszyklus $(l_1', l_2', \ldots, l_p')$ auftritt, läßt sich in einen stationären Bereich, einen Aufsetzbereich und einen Auslaufbereich unterteilen.

Als stationärer Bereich wird der Bereich bezeichnet, in dem die Pipelineaktivitäten nach einem festen, sich wiederholenden Ablaufmuster erfolgen. Der stationäre Bereich läßt sich in Teilbereiche einteilen, die im weiteren als stationäre Intervalle bezeichnet werden. Ein stationäres Intervall ist der Bereich, in dem ein Ablauf der Pipelineaktivitäten nach dem sich wiederholenden Ablaufmuster erfolgt. Die Dauer eines stationären Intervalls ist gleich der Umlaufdauer des Verzögerungszyklus. Sie beträgt also p L' Zeiteinheiten. Ist H die Anzahl der stationären Intervalle, gilt für die Dauer $t_s$ des stationären Bereichs:

$$t_s = H \cdot p \cdot L' \tau$$

Der Bereich, in dem die Pipeline gefüllt wird, also der Bereich vom Beginn der Pipelineaktivitäten gemäß dem Verzögerungszyklus bis zum Beginn des stationären Bereichs wird als Aufsetzbereich bereich bezeichnet. Die Dauer $t_a$ des Aufsetzbereichs wird entsprechend als Aufsetzzeit (set up time) bezeichnet.

Für die Aufsetzzeit einer statischen Pipeline gilt:

$$t_a = (m - l_p')\tau \ ,$$

wenn $m\tau$ die Ausführungszeit der Reservierungstafel ist.

Für die Aufsetzzeit einer dynamischen Pipeline gilt:

$$t_a = \underset{i}{\mathrm{MAX}}\left(m_i - \sum_{j=i}^{p} l_j'\right)\tau \ , \quad 1 \leq i \leq N,$$

wenn nacheinander Reservierungstafeln mit den Ausführungszeiten $m_1\tau$, $m_2\tau$, ... , $m_p\tau$ initiiert werden.

An den stationären Bereich schließt sich der Auslaufbereich an, in dem noch maximal p-1 Initiierungen erfolgen können. Für die Dauer $t_e$ des Auslaufbereichs gilt:

$$t_e = T_k - t_a - t_s \ ,$$

wobei $T_k$ die Gesamtausführungszeit der Pipeline ist (vgl. Abschnitt 2.4.5.).

Demnach gilt für eine statischen Pipeline:

$$t_e = (L \cdot (N-1) + l_p' - H \cdot p \cdot L')\tau$$

Beispiel 2.4.6.

a) Für die statische Pipeline aus Beispiel 2.4.2.a) gilt:

Aufsetzzeit: $2\tau$
Dauer eines stationären Intervalls: $8\tau$
Dauer des Auslaufbereichs: $8\tau$ für den Fall N=2H,
$6\tau$ für den Fall N=2H+1,
wenn H die Anzahl der stationären Intervalle ist.

Raum-Zeit Diagramm für den Fall N=2H+1:

Aufsetz-bereich:

| | 0 | 1 |
|---|---|---|
| 1 | $A_1$ | $A_1$ |
| 2 | | |
| 3 | | $A_1$ |
| 4 | $A_1$ | |

Stationärer Bereich:

| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|
| 1 | $A_2$ | $A_2$ | | $A_1$ | | $A_2$ | $A_3$ | $A_3$ |
| 2 | | $A_1$ | $A_1$ | $A_2$ | $A_2$ | | | |
| 3 | $A_1$ | $A_2$ | $A_2$ | | $A_1$ | | $A_2$ | $A_3$ |
| 4 | $A_2$ | | | | | $A_1$ | $A_3$ | $A_2$ |

| | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $A_4$ | $A_4$ | | $A_3$ | | $A_4$ | $A_5$ | $A_5$ | | |
| 2 | | $A_3$ | $A_3$ | $A_4$ | $A_4$ | | | | | |
| 3 | $A_3$ | $A_4$ | $A_4$ | | $A_3$ | | $A_4$ | $A_5$ | | |
| 4 | $A_4$ | | | | | $A_3$ | $A_5$ | $A_4$ | | |

| | 2+4(N-3) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | $A_{N-1}$ | $A_{N-1}$ | | $A_{N-2}$ | | $A_{N-1}$ | $A_N$ | $A_N$ |
| 2 | | $A_{N-2}$ | $A_{N-2}$ | $A_{N-1}$ | $A_{N-1}$ | | | |
| 3 | $A_{N-2}$ | $A_{N-1}$ | $A_{N-1}$ | | $A_{N-2}$ | | $A_{N-1}$ | $A_N$ |
| 4 | $A_{N-1}$ | | | | | $A_{N-2}$ | $A_N$ | $A_{N-1}$ |

Auslauf-bereich:

| | 2+4(N-1) | | | | | |
|---|---|---|---|---|---|---|
| 1 | | | | $A_N$ | | |
| 2 | | $A_N$ | $A_N$ | | | |
| 3 | $A_N$ | | | | $A_N$ | |
| 4 | | | | | | $A_N$ |

b) Für die dynamische Pipeline aus Beispiel 2.4.2.b) gilt:

Aufsetzzeit: $0\tau$
Dauer eines stationären Intervalls: $10\tau$
Dauer des Auslaufbereichs: $0\tau$ für den Fall N=3H,
$8\tau$ für den Fall N=3H+1,
$8\tau$ für den Fall N=3H+2,
wenn H die Anzahl der stationären Intervalle ist.

Raum-Zeit Diagramm für den Fall N=3H:

Stationärer
Bereich:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $A_1$ | $A_1$ | $B_1$ | $B_1$ | | $A_1$ | | $B_2$ | $B_2$ | |
| 2 | $B_1$ | $B_1$ | | $A_1$ | $A_1$ | $B_2$ | $B_2$ | | | |
| 3 | $B_1$ | $A_1$ | $A_1$ | | $B_1$ | $B_2$ | $A_1$ | | | $B_2$ |
| 4 | $A_1$ | | | $B_1$ | | | | $A_1$ | $B_2$ | |

| | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $A_2$ | $A_2$ | $B_3$ | $B_3$ | | $A_2$ | | $B_4$ | $B_4$ | | | |
| 2 | $B_3$ | $B_3$ | | $A_2$ | $A_2$ | $B_4$ | $B_4$ | | | | | |
| 3 | $B_3$ | $A_2$ | $A_2$ | | $B_3$ | $B_4$ | $A_2$ | | | $B_4$ | | |
| 4 | $A_2$ | | | $B_3$ | | | | $A_2$ | $B_4$ | | | |

| | 10(H-1) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $A_H$ | $A_H$ | $B_{2H-1}$ | $B_{2H-1}$ | | $A_H$ | | $B_{2H}$ | $B_{2H}$ | |
| 2 | $B_{2H-1}$ | $B_{2H-1}$ | | $A_H$ | $A_H$ | $B_{2H}$ | $B_{2H}$ | | | |
| 3 | $B_{2H-1}$ | $A_H$ | $A_H$ | | $B_{2H-1}$ | $B_{2H}$ | $A_H$ | | | $B_{2H}$ |
| 4 | $A_H$ | | | $B_{2H-1}$ | | | | $A_H$ | $B_{2H}$ | |

## 2.4.7 RELATIVER SPEEDUP

Als relativer Speedup $S_r$ einer Pipeline mit k Segmenten soll im weiteren der Faktor bezeichnet werden, um den sich die Ausführungsgeschwindigkeit bei überlappter Verarbeitung gegenüber sequentieller Verabeitung bei N Ausführungen in der Pipeline steigert.

Mit Abschnitt 2.4.5. folgt daher:

Für den relativen Speedup einer statischen Pipeline mit einer Reservierungstafel, deren Ausführungszeit $m\tau$ beträgt, gilt:

$$S_r = T_s/T_k = N \cdot m/(L \cdot (N-1)+m)$$

Für den relativen Speedup einer dynamischen Pipeline mit N Reservierungstafeln der Ausführungszeiten $m_1\tau$, $m_2\tau$, ... , $m_N\tau$ gilt:

$$S_r = T_s/T_k = N \cdot m/MAX((\sum_{i=0}^{j} l_i)+m_{j+1}),\ 0 \leq j \leq N-1$$

Bemerkungen:

1. Es gilt: $1 \leq S_r \leq k$

2. Im stationären Bereich einer Pipeline gilt:

   $S_r = m/L$ bei statischen Pipelines,

   $S_r = m/L$ bei dynamischen Pipelines

Beispiel 2.4.7.

a) Für die statische Pipeline aus Beispiel 2.4.2.a) gilt im stationären Bereich: $S_r = 2$

b) Für die dynamische Pipeline aus Beispiel 2.4.2.b) gilt im stationären Bereich: $S_r = 9/5$

c) Für die Additionspipeline aus Abschnitt 2.1. gilt : $S_r = 5$

## 2.4.8 EFFIZIENZ

Die Effizienz $E_k$ einer Pipeline mit k Segmenten ist das Verhältnis der Anzahl aller tatsächlichen Pipelinaktivitäten für N Ausführungen in der Pipeline zur Anzahl aller in der dafür benötigten Ausführungszeit möglichen Pipelineaktivitäten.

Für eine statische Pipeline mit einer Reservierungstafel, deren Ausführungszeit $m\tau$ beträgt, gilt bei einer mittleren Verzögerung L:

$$E_k = N \cdot \sum_{i=1}^{k} W(i)/(k \cdot (L \cdot (N-1)+m)) \leq 1$$

Für eine dynamische Pipeline mit Reservierungstafeln, deren Ausführungszeiten $m_1\tau$, $m_2\tau$, ... , $m_N\tau$ betragen, gilt bei einer mittleren Verzögerung L:

$$E_k = N \cdot \sum_{i=1}^{k} \overline{W}(i)/(k \cdot \mathrm{MAX}_j((\sum_{i=0}^{j} l_i)+m_{j+1}) \leq 1, \quad 0 \leq j \leq N-1$$

Bemerkung:

Im stationären Bereich einer Pipeline ist die Effizienz gleich der mittleren Auslastung aller Pipelinesegmente (siehe dazu auch Abschnitt 2.4.3.):

$$E_k = \sum_{i=1}^{k} U(i)/k$$

Beispiel 2.4.8.

a) Für die Pipeline aus Beispiel 2.4.2.a) gilt im stationären Bereich: $E_k = 5/8$

b) Für die Pipeline aus Beispiel 2.4.2.b) gilt im stationären Bereich: $E_k = 3/5$

c) Für die Additionspipeline aus Abschnitt 2.1. gilt im stationären Bereich: $E_k = 1$

# 3.0 SCHEDULING

Scheduling bedeutet die Terminierung oder Ablaufplanung von Vorgängen. Beispiele hierfür sind das Erstellen von Arbeits-, Produktions-, Fahr- oder Stundenplänen.

## 3.1 SCHEDULING IN RECHNERN

Scheduling in Rechnern ist zur Auswahl von Prozessen bei der Betriebsmittelvergabe erforderlich.

Als Prozeß wird dabei die Folge der Aktivitäten bezeichnet, die zur Ausführung eines Programms oder Programmteils erforderlich ist. Zum Ablauf seiner Aktivitäten benötigt ein Prozeß Betriebsmittel. Da in einem Rechnersystem viele Prozesse gleichzeitig existieren können und die Zahl der Betriebsmittel beschränkt ist, ergibt sich bei der Vergabe der Betriebsmittel laufend die Situation, daß mehrere Prozesse gleichzeitig auf die Zuteilung desselben Betriebsmittels warten. In diesem Fall muß nach einer vorgegebenen Scheduling-Strategie ein Prozeß ausgewählt werden, der nach erfolgter Zuteilung seine Ausführungen fortsetzen kann (vgl. /Kr,Ne,Un 75/).

## 3.2 ZIEL EINER SCHEDULING-STRATEGIE FÜR PIPELINES

Sollen in einer Pipeline N Ausführungen von Operationen erfolgen, bedeutet dies, daß N Berechnungsprozesse auf die Zuteilung des Betriebsmittels Pipeline warten. Bei Zuteilung der Pipeline an einen Prozeß erfolgt die Initiierung der Reservierungstafel, die diesen Prozeß beschreibt.

Ziel einer Scheduling-Strategie für eine Pipeline ist die Festlegung einer Folge von Zeitpunkten, zu denen Initiierungen erfolgen können, ohne daß dadurch Kollisionen verursacht werden. Äquivalent dazu ist das Ziel, eine zulässige Verzögerungsfolge für die Pipeline zu entwickeln.

Als optimale Scheduling-Strategie für eine Pipeline wird eine Strategie bezeichnet, gemäß der eine Verzögerungsfolge erstellt wird, deren mittlere Verzögerung gleich der kleinsten erreichbaren mittleren Verzögerung ist.

## 3.3 GREEDY-STRATEGIE

Als Greedy[1] -Strategie wird die Scheduling-Strategie bezeichnet, die jeweils die kleinstmögliche zulässige Verzögerung zwischen zwei Initiierungen auswählt (vgl. /Ko 81/). Die Greedy-Strategie ist jedoch nicht unbedingt optimal, da Abhängigkeiten zwischen später zulässigen Verzögerungen zu einer zuvor gewählten Verzögerung bestehen können, wie das folgende Beispiel zeigt:

Beispiel 3.3.

Greedy-Strategie und optimale Strategie für eine statische Pipeline mit der Reservierungstafel aus Beispiel 2.4.1.a).

a) Raum-Zeit Diagramm bei Initiierung nach der Greedy-Strategie

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $A_1$ | $A_1$ | $A_2$ | $A_2$ | | $A_1$ | | $A_2$ | $A_3$ | $A_3$ | $A_4$ |
| 2 | | | | $A_1$ | $A_1$ | $A_2$ | $A_2$ | | | | |
| 3 | | $A_1$ | $A_1$ | $A_2$ | $A_2$ | | $A_1$ | | $A_2$ | $A_3$ | $A_3$ |
| 4 | $A_1$ | | $A_2$ | | | | | $A_1$ | $A_3$ | $A_2$ | $A_4$ |

| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $A_4$ | | $A_3$ | | $A_4$ | $A_5$ | $A_5$ | $A_6$ | $A_6$ | | | |
| 2 | $A_3$ | $A_3$ | $A_4$ | $A_4$ | | | | | $A_5$ | $A_5$ | | |
| 3 | $A_4$ | $A_4$ | | $A_3$ | | $A_4$ | $A_5$ | $A_5$ | $A_6$ | $A_6$ | | |
| 4 | | | | | $A_3$ | $A_5$ | $A_4$ | $A_6$ | | | | |

Verzögerungszyklus: (2,6)
mittlere Verzögerung: 4

b) Raum-Zeit Diagramm bei Initiierung nach einer optimalen Strategie

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $A_1$ | $A_1$ | | $A_2$ | $A_2$ | $A_1$ | $A_3$ | $A_3$ | $A_2$ | $A_4$ | $A_4$ |
| 2 | | | | $A_1$ | $A_1$ | | $A_2$ | $A_2$ | | $A_3$ | $A_3$ |
| 3 | | $A_1$ | $A_1$ | | $A_2$ | $A_2$ | $A_1$ | $A_3$ | $A_3$ | $A_2$ | $A_4$ |
| 4 | $A_1$ | | | $A_2$ | | | $A_3$ | $A_1$ | | $A_4$ | $A_2$ |

| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $A_3$ | $A_5$ | $A_5$ | $A_4$ | $A_6$ | $A_6$ | $A_5$ | $A_7$ | $A_7$ | $A_6$ | | |
| 2 | | $A_4$ | $A_4$ | | $A_5$ | $A_5$ | | $A_6$ | $A_6$ | | | |
| 3 | $A_4$ | $A_3$ | $A_5$ | $A_5$ | $A_4$ | $A_6$ | $A_6$ | $A_5$ | $A_7$ | $A_7$ | | |
| 4 | | $A_5$ | $A_3$ | | $A_6$ | $A_4$ | | $A_7$ | $A_5$ | | | |

Verzögerungszyklus: (3)
mittlere Verzögerung: 3

Diese Strategie ist wegen MAL$\geq$3 (vgl. Abschnitt 2.4.4.) optimal.

---

[1] greedy (englisch) : gierig, habsüchtig

# 4.0 ENTWICKLUNG EINER OPTIMALEN SCHEDULING-STRATEGIE FÜR STATISCHE PIPELINES

Ziel der weiteren Überlegungen ist die Entwicklung einer optimalen Scheduling-Strategie für statische Pipelines. Dazu sollen zunächst einige Hilfsmittel (vgl. /Da 71/, /Sh 72/, /Th,Da 74/, /Da 74/, /Da,Th,Sh,Pa 75/, /Pa,Da 76/, /Ko 81/) angegeben werden. Danach soll gezeigt werden, welche Schritte von einer optimalen Scheduling-Strategie durchzuführen sind. Anschließend wird ein Verfahren vorgestellt, das alle optimalen Verzögerungsfolgen zu einer statischen Pipeline liefern soll.

## 4.1 HILFSMITTEL

### 4.1.1 KOLLISIONSVEKTOR

Als Kollisionsvektor wird ausgehend von einem festen Zeitpunkt während des Ablaufs von Berechnungsprozessen in einer statischen Pipeline ein Bitvektor mit den folgenden Eigenschaften bezeichnet:

1. Die Länge des Vektors ist m, wenn die Reservierungstafel der Pipeline eine Ausführungszeit von m Zeiteinheiten hat.

2. Das Bit b auf der Position i des Vektors, i=0(1)m-1, zeigt an, ob zur i-ten Zeiteinheit ausgehend von der momentanen Zeiteinheit eine Initiierung zulässig ist (b=0) oder nicht (b=1).

Als Anfangskollisionsvektor oder Kollisionsvektor der Reservierungstafel wird der Kollisionsvektor für den Start der ersten Ausführung in der Pipeline zur Zeiteinheit 0 bezeichnet.

Der Anfangskollisionsvektor ergibt sich dadurch, daß die Reservierungstafel um jeden Wert i, i=0(1)m-1, nach rechts geschoben und jeweils auf die ursprüngliche Reservierungstafel gelegt wird. Kommt es dabei zur Überlagerung zweier Markierungen, d.h. tritt eine Kollision auf, ist das i-te Bit des Kollisionsvektors 1, sonst 0.

Beispiel 4.1.1. Konstruktion des Kollisionsvektors für die Reservierungstafel aus Beispiel 2.4.1.a).

Die Markierungen der ursprünglichen Reservierungstafel werden im folgenden mit A, die der jeweils verschobenen mit O gekennzeichnet. Da sich die Markierungen der Reservierungstafeln nur im Bereich der ersten m Zeiteinheiten überlagern können, werden die weiteren Zeiteinheiten nicht berücksichtigt. Aus diesem Grund besitzt der Kollisionvektor auch nur m Positionen.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1 | AO | AO | | | | AO | | |
| 2 | | | | AO | AO | | | |
| 3 | | AO | AO | | | | AO | |
| 4 | AO | | | | | | | AO |

Verzögerung 0: Kollision (1)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1 | A | AO | O | | | A | O | |
| 2 | | | | A | AO | O | | |
| 3 | | A | AO | O | | | A | O |
| 4 | A | O | | | | | | A |

Verzögerung 1: Kollision (1)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1 | A | A | O | O | | A | | O |
| 2 | | | | A | A | O | O | |
| 3 | | A | A | O | O | | A | |
| 4 | A | | O | | | | | A |

Verzögerung 2: keine Kollision (0)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1 | A | A | | O | O | A | | |
| 2 | | | | A | A | | O | O |
| 3 | | A | A | | O | O | A | |
| 4 | A | | | O | | | | A |

Verzögerung 3: keine Kollision (0)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1 | A | A | | | O | AO | | |
| 2 | | | | A | A | | | O |
| 3 | | A | A | | | O | AO | |
| 4 | A | | | | O | | | A |

Verzögerung 4: Kollision (1)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1 | A | A | | | | AO | O | |
| 2 | | | | A | A | | | |
| 3 | | A | A | | | | AO | O |
| 4 | A | | | | | O | | A |

Verzögerung 5: Kollision (1)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1 | A | A | | | | A | O | O |
| 2 | | | | A | A | | | |
| 3 | | A | A | | | | A | O |
| 4 | A | | | | | | O | A |

Verzögerung 6: keine Kollision (0)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1 | A | A | | | | A | | O |
| 2 | | | | A | A | | | |
| 3 | | A | A | | | | A | |
| 4 | A | | | | | | | AO |

Verzögerung 7: Kollision (1)

Damit lautet der Kollisionsvektor : (1,1,0,0,1,1,0,1)

Bemerkung: Bit 0 des Kollisionsvektors hat immer den Wert 1.

### 4.1.2 ZUSTANDSDIAGRAMM EINER STATISCHEN PIPELINE

Das Zustandsdiagramm einer statischen Pipeline gibt an, zu welchen Zeiteinheiten, ausgehend von einer gegebenen Zeiteinheit, Initiierungen zulässig sind.

Ein Zustandsdiagramm besteht aus einer Menge von Zuständen und Zustandsübergängen (Transitionen). Zu jeder Zeiteinheit befindet sich die Pipeline in genau einem dieser Zustände, der durch den Kollisionsvektor der Pipeline zu der Zeiteinheit eindeutig markiert ist. Ein Zustand enthält also die Informationen darüber, zu welchen der nächsten m Zeiteinheiten Initiierungen zulässig sind oder nicht. Mit dem Ablauf einer Zeiteinheit erfolgt im Zustandsdiagramm der Übergang zu einem nächsten Zustand.

Konstruktion des Zustandsdiagramms (Davidson, 1971):

Ausgehend von einem Anfangszustand mit dem Anfangskollisionsvektor lassen sich alle weiteren Zustände nach der folgenden Vorschrift bestimmen: Angenommen, der Kollisionvektor zur Zeiteinheit i sei bekannt. Dann ergeben sich die möglichen Kollisionsvektoren und damit die möglichen Zustände zur Zeiteinheit i+1 wie folgt:

1. Der Kollisionsvektor für die Zeiteinheit i wird um eine Position nach links geschoben. Das Bit auf der Position m-1 wird zu null gesetzt.

2. Hat das Bit auf der Position 0 dieses Vektors den Wert 1, ist dieser der neue Kollisionsvektor.

3. Hat das Bit auf der Position 0 des Vektors den Wert 0, gibt es zwei Möglichkeiten:

   3.1. Falls eine Initiierung mit Beginn der Zeiteinheit i+1 erfolgt, ergibt sich der neue Kollisionsvektor durch bitweise logische Oder-Verknüpfung des Vektors mit dem Anfangskollisionsvektor.

   3.2. Falls keine Initiierung zur Zeiteinheit i+1 erfolgt, ist der Vektor der neue Kollisionsvektor.

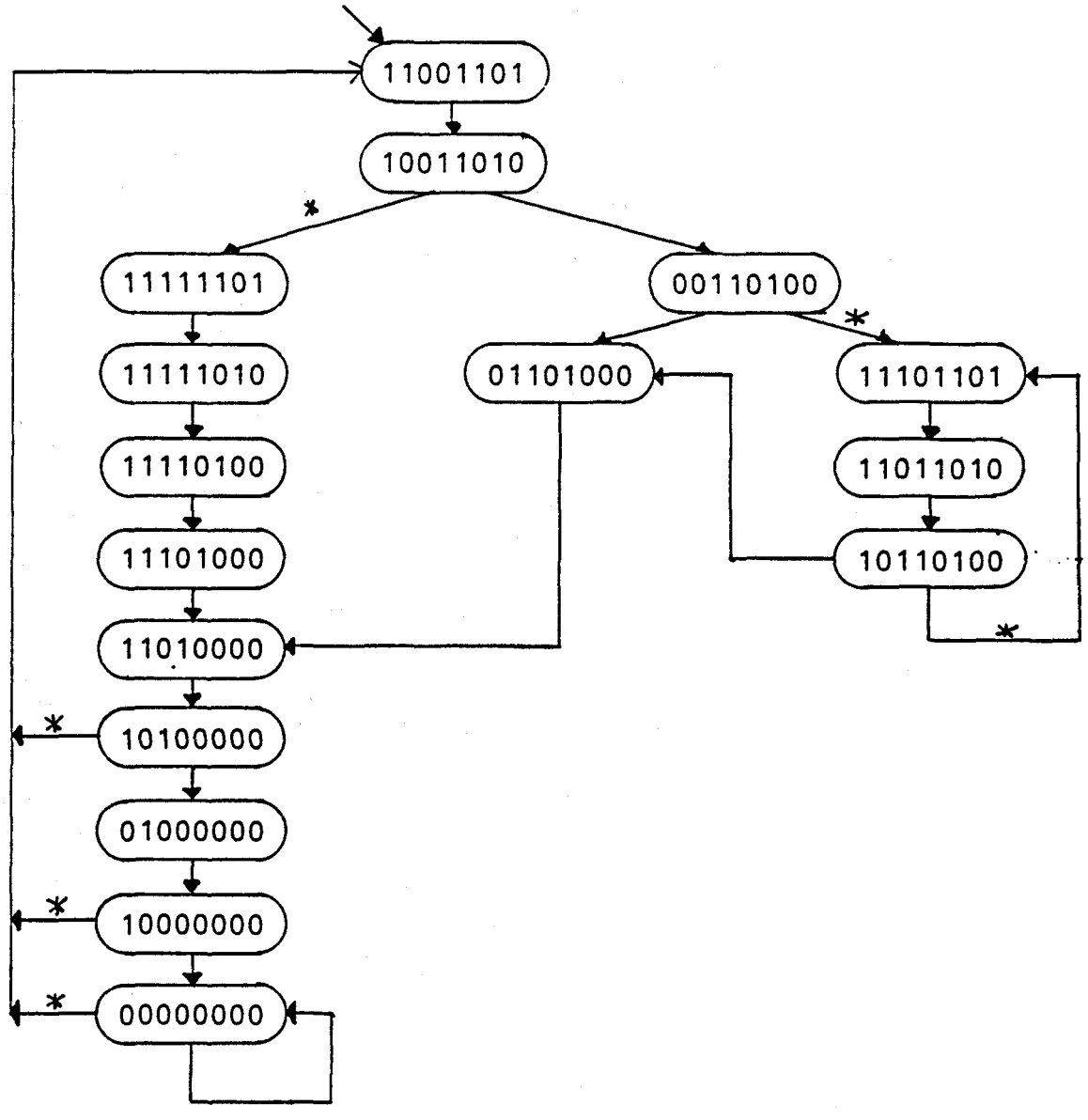
4. Ist der neue Kollisionsvektor identisch mit dem Kollisionsvektor eines früheren Zustands, erfolgt der Übergang vom momentanen Zustand zu diesem früheren Zustand.
Andernfalls wird ein neuer Zustand mit diesem Kollisionsvektor festgelegt, zu dem der Übergang erfolgt.

Bemerkung:

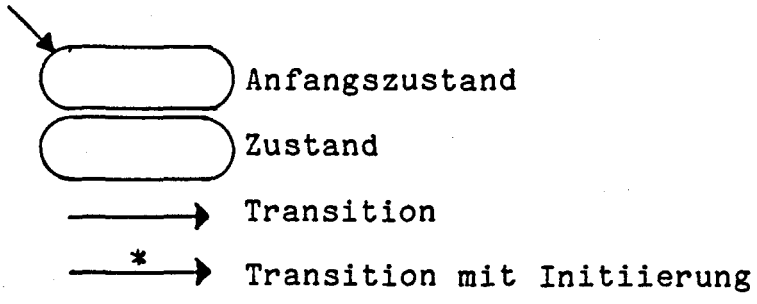
Die Anzahl der Zustände ist wegen der endlichen Länge des Kollisionsvektors beschränkt. Ist $K_1$ die Anzahl der Nullen, die links von der letzten im Anfangskollisionsvektor auftretenden Eins stehen, gilt für die Anzahl n der Zustände:

$$n \leq 2^{K_1}$$

Beispiel 4.1.2. Zustandsdiagramm der statischen Pipeline mit der Reservierungstafel aus Beispiel 2.4.1.a)




Zeichenerklärung:



## 4.1.3 MODIFIZIERTES ZUSTANDSDIAGRAMM

Zur besseren Übersicht wird das in Abschnitt 4.1.2. beschriebene Zustandsdiagramm wie folgt geändert (Davidson, 1971):

1. Alle Zustände bis auf den Anfangszustand und diejenigen Zustände, zu denen eine Transition mit Initiierung erfolgt, werden eliminiert.

2. Es erfolgt im modifizierten Zustandsdiagramm nur dann eine Transition zwischen zwei Zuständen, wenn diese im ursprünglichen Zustandsdiagramm durch einen Transitionsweg verbunden sind, in dem nur mit der letzten Transition eine Initiierung erfolgt. Die Anzahl der Transitionen dieses Transitionswegs wird im modifizierten Zustandsdiagramm an die dafür geschaffene neue Transition geschrieben. Als Transitionsweg wird eine Folge von Transitionen bezeichnet, die in einem Zustandsdiagramm nacheinander durchlaufen werden können.

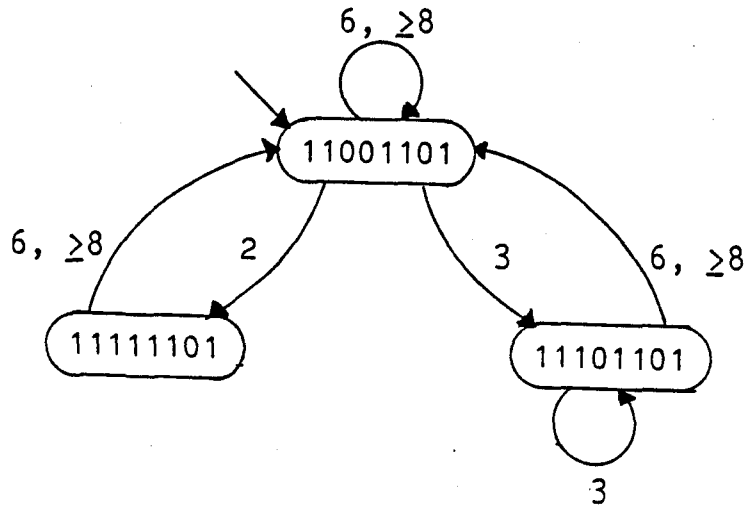
Beispiel 4.1.3.a) Erstellen des modifizierten Zustandsdiagramms für das in Beispiel 4.1.2. angegebene Zustandsdiagramm:

1. Elimination aller Zustände außer dem Anfangszustand, also (11001101), und den Zuständen, zu denen eine Transition mit Initiierung erfolgt, also (11111101) und (11101101).

2.

| Transitionen | | Transitionszahl |
|---|---|---|
| von | nach | |
| 11001101 | 11001101 | 6, ≥8 |
| 11001101 | 11111101 | 2 |
| 11001101 | 11101101 | 3 |
| 11111101 | 11001101 | 6, ≥8 |
| 11101101 | 11001101 | 6, ≥8 |
| 11101101 | 11101101 | 3 |

Modifiziertes Zustandsdiagramm:



Konstruktion des modifizierten Zustandsdiagramms (Davidson, 1971):

1. Auszeichnung eines Anfangszustands mit dem Anfangskollisionsvektor der Länge m.

2. Die weiteren Zustände und die Transitionen werden sukzessive nach der folgenden Vorschrift ermittelt:

Für jeden bereits ermittelten Zustand, von dem noch keine Transition ausgeht, und für jede Zahl i, i=0(1)m-1, für die das Bit auf der Position i des Kollisionsvektors zu diesem Zustand den Wert 0 hat, wird folgendes durchgeführt:

2.1. Schieben des Kollisionsvektors um i Positionen nach links und Hereinschieben von i Nullen von rechts.

2.2. Der dadurch entstandene Vektor wird durch eine bitweise logische Oder-Verknüpfung mit dem Anfangskollisionsvektor verknüpft.

2.3. Ergebnis dieser Verknüpfung ist der Kollisionsvektor für einen Folgezustand. Ist dieser Kollisionsvektor gleich dem eines bereits ermittelten Zustands, erfolgt eine Transition zu diesem. Andernfalls wird ein neuer Zustand mit diesem Kollisionsvektor und eine Transition dorthin geschaffen. In beiden Fällen wird die Zahl i an die Transition geschrieben.

3. Von jedem Zustand aus wird eine Transition mit den Zahlen, die größer oder gleich m sind und im Diagramm mit $\geq$m bezeichnet werden, zum Anfangszustand geschaffen.

Bemerkungen:

1. Im modifizierten Zustandsdiagramm ist die an eine Transition geschriebene Zahl gleich der Verzögerung zwischen der Initiierung, die mit dieser Transition erfolgt, und der davor zuletzt erfolgten Initiierung.

2. Eine zulässige Verzögerungsfolge ist gleich der Folge von Zahlen entlang eines vom Anfangszustand ausgehenden Transitionswegs.

3. Das modifizierte Zustandsdiagramm liefert alle zulässigen Verzögerungsfolgen der diesem Diagramm zugrunde liegenden statischen Pipeline. Aufgabe einer Scheduling-Strategie ist es also nur, eine Verzögerungsfolge aus dem modifizierten Zustandsdiagramm auszuwählen.

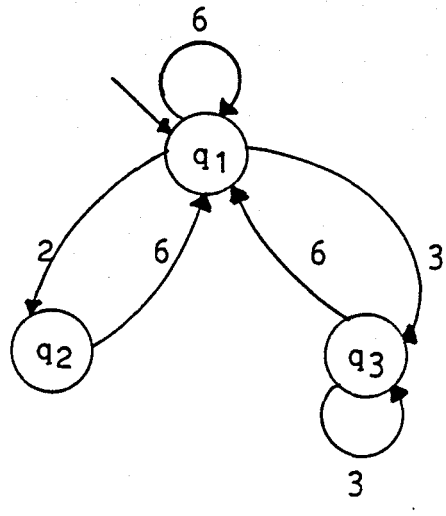
4. Ziel einer optimalen Scheduling-Strategie ist es, aus dem modifizierten Zustandsdiagramm die Verzögerungsfolgen zu ermitteln, deren mittlere Verzögerung jeweils gleich der kleinsten erreichbaren mittleren Verzögerung ist, und daraus dann eine geeignete auszuwählen.

Dazu kann das modifizierte Zustandsdiagramm noch wie folgt vereinfacht werden:

Sei $Q=\{q_1,q_2,\ldots,q_n\}$ die Menge aller Zustände des modifizierten Zustandsdiagramms. Führen von einem Zustand $q_i \in Q$ mehrere Transitionen zu einem Zustand $q_j \in Q$, dann werden alle Transitionen bis auf diejenige mit der kleinsten Verzögerung eliminiert.

Beispiel 4.1.3.b)

Das modifizierte Zustandsdiagramm aus Beispiel 4.1.3.a) sieht nach der Vereinfachung gemäß Bemerkung 4 wie folgt aus, wobei der Zustand (11001101) mit $q_1$, (11111101) mit $q_2$ und (11101101) mit $q_3$ bezeichnet wird:



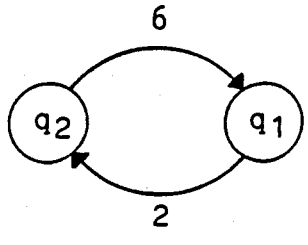
## 4.2 AUFGABEN EINER OPTIMALEN SCHEDULING-STRATEGIE

Mit einer optimalen Scheduling-Strategie sollen alle zulässigen Verzögerungsfolgen mit der kleinsten erreichbaren mittleren Verzögerung unabhängig davon bestimmt werden, wieviele Initiierungen in der Pipeline erfolgen. Bei einer beliebigen Anzahl von Initiierungen und einer beschränkten Anzahl von Zuständen des modifizierten Zustandsdiagramms ergibt sich, daß entsprechend der Initiierungen im modifizierten Zustandsdiagramm Schleifen durchlaufen werden. Als Schleife wird eine Menge von Zuständen und Transitionen bezeichnet, welche die Eigenschaft haben, daß ein Transitionsweg, der von einem beliebigen Zustand ausgeht, wieder zu diesem zurückführt. Die Folge von Verzögerungen $l'_1,l'_2,\ldots,l'_p$ mit $p\geq 1$ entlang einer Schleife bildet den Verzögerungszyklus $(l'_1,l'_2,\ldots,l'_p)$ mit der mittleren Verzögerung

$$\sum_{i=1}^{p} l'_i/p$$

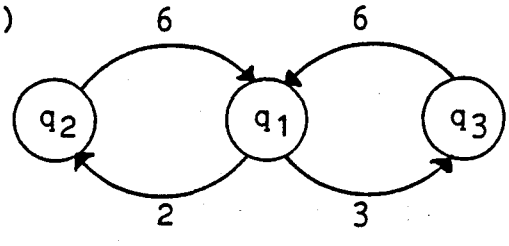
Beispiel 4.2. Das modifizierte Zustandsdiagramm aus Beispiel 4.1.3.b) enthält unter anderem folgende Schleifen, wobei die Zustände wie folgt bezeichnet werden: $q_1:=(11001101)$, $q_2:=(11111101)$, $q_3:=(11101101)$.

a)




Verzögerungszyklus (2,6)

b)




Verzögerungszyklus (2,6,3,6)

c)


Verzögerungszyklus (3)

Eine optimale Scheduling-Strategie muß demnach folgendes leisten:

1. Ermittlung der Menge der Verzögerungszyklen aller Schleifen des modifizierten Zustandsdiagramms.

2. Ermittlung der optimalen Verzögerungszyklen dieser Menge, also derjenigen mit der minimalen mittleren Verzögerung.

3. Ermittlung der Menge der Verzögerungsfolgen zu den kürzesten Transitionswegen vom Anfangszustand zu einem Zustand einer Schleife mit optimalem Verzögerungszyklus.

Im folgenden soll gezeigt werden, daß es bei der Ermittlung der Verzögerungszyklen aller Schleifen bereits genügt, sich auf Kreise zu beschränken. Als Kreis wird eine Schleife bezeichnet, in der jeder Zustand bei einem Durchlauf, ausgehend von einem beliebigen Zustand, nur einmal durchlaufen wird.

Beispiel: Die Schleifen aus Beispiel 4.2.a) und c) sind Kreise, die Schleife aus b) bildet keinen Kreis, da der Zustand $q_1$ zweimal durchlaufen wird.

Lemma (Shar,1972)

In einem modifizierten Zustandsdiagramm gibt es zu jeder Schleife mit einer mittleren Verzögerung L einen Kreis, dessen mittlere Verzögerung nicht größer als L ist.

Beweis:

Für jede Schleife des modifizierten Zustandsdiagramms gilt:

Entweder ist die Schleife ein Kreis, dann ist das Lemma dafür bewiesen.

Ist die Schleife kein Kreis, besteht sie aus einer Kombination von Kreisen. Beispiel: Die Schleife aus Beispiel 4.2.b) ist eine Kombination der Kreise mit den Verzögerungszyklen (2,6), (3,6).

Voraussetzung: Die Schleife bestehe aus M Kreisen, wovon der i-te Kreis bei einem Schleifendurchlauf K(i) mal durchlaufen wird, S(i) Zustände enthält und dessen Verzögerungszyklus die Umlaufzahl P(i) hat.

Für die mittlere Verzögerung L der Schleife gilt dann:

$$L = \frac{\sum_{i=1}^{M} K(i)\cdot P(i)}{\sum_{i=1}^{M} K(i)\cdot S(i)}$$

Annahme: Das Lemma ist falsch. Dann ist die mittlere Verzögerung jedes der M Kreise größer als die der Schleife. Es gilt also:

$$\frac{\sum_{i=1}^{M} K(i)\cdot P(i)}{\sum_{i=1}^{M} K(i)\cdot S(i)} < \frac{P(j)}{S(j)} \quad \text{für alle } j,\ 1\leq j\leq M$$

Mit $\frac{P(r)}{S(r)} := \underset{j}{\mathrm{MIN}} \frac{P(j)}{S(j)}$ , $1\leq j\leq M$, folgt

$$\sum_{\substack{i=1\\ i\neq r}}^{M} K(i)\cdot (P(i)\cdot S(r)-P(r)\cdot S(i)) < 0$$

Wegen $K(i)>0$ für alle i, $1\leq i\leq M$, folgt, daß für mindestens ein i gilt:

$$P(i)\cdot S(r)-P(r)\cdot S(i) < 0$$

Daraus folgt aber $\frac{P(i)}{S(i)} < \frac{P(r)}{S(r)}$ ,

was ein Widerspruch zur Definition von P(r)/S(r) ist.

## 4.3 OPTIMALE SCHEDULING-STRATEGIE

Im folgenden wird zuerst ein Verfahren konstruiert, das die Verzögerungszyklen aller Kreise des modifizierten Zustandsdiagramms liefern soll.

Als formales Hilfsmittel werden zur Beschreibung von Transitionswegen und ihren dazugehörigen Verzögerungsfolgen reguläre Ausdrücke benutzt (siehe /In,Kl 81/,Kapitel 2.5 und 2.6). Das bietet sich an, da das modifizierte Zustandsdiagramm der Transitionsgraph eines endlich erkennenden Automaten ist und jede Menge zulässiger Verzögerungsfolgen eine reguläre Sprache ist.

Mit Hilfe der regulären Ausdrücke werden ausgehend von jedem Zustand des modifizierten Zustandsdiagramms nacheinander alle Transitionswege bestimmt, die aus einer, zwei usw. bis maximal aus n Transitionen bestehen und keinen Zustand mehrfach durchlaufen, wobei n die Anzahl der Zustände ist. Wird dabei ein Transitionsweg bestimmt, der wieder zurück zu seinem Ausgangszustand führt, wird dieser als Kreis erkannt und ausgesondert; sein Verzögerungszyklus wird durch einen diesen Kreis beschreibenden regulären Ausdruck mitgeliefert. Auf diese Weise sollen die Verzögerungszyklen aller Kreise bestimmt werden.

### 4.3.1 REGULÄRE AUSDRÜCKE

Die für das Verfahren notwendigen Begriffe und Eigenschaften regulärer Ausdrücke sollen zuvor kurz eingeführt werden. Für weitere Erläuterungen sei auf /In,Kl 81/, Kapitel 2 verwiesen.

Eigenschaften regulärer Ausdrücke

Sei $\Sigma = \{a_1, a_2, \ldots, a_n\}$ eine endliche, nicht-leere Menge.
Die Menge $RA(\Sigma)$ regulärer Ausdrücke über $\Sigma$ besitzt unter anderem folgende Eigenschaften:

$\Lambda \in RA(\Sigma)$

$a_i \in RA(\Sigma)$ für alle $a_i \in \Sigma$

$\alpha, \beta \in RA(\Sigma) \rightsquigarrow (\alpha \vee \beta) \in RA(\Sigma)$

$\alpha, \beta \in RA(\Sigma) \rightsquigarrow (\alpha \vee \beta) \in RA(\Sigma)$

Bemerkungen:

1. $\vee$ ist das Zeichen einer Oder-Verknüpfung
2. $\cdot$ ist das Zeichen für die Verkettung (Konkatenation)
3. Konvention: $\cdot$ bindet stärker als $\vee$, der Punkt $\cdot$ wird nicht notiert. Beispiel: statt $(\alpha \vee (\beta \cdot \gamma))$ wird $\alpha \vee \beta\gamma$ geschrieben.

## Semantik regulärer Ausdrücke

Für die regulären Ausdrücke ist eine Semantik $[\![\,]\!] : RA(\Sigma) \to P(\Sigma^*)$ unter anderem durch die folgenden Festlegungen definiert:

$[\![\Lambda]\!] := \varnothing$
$[\![a_i]\!] := \{a_i\}$ für alle $a_i \in \Sigma$
$[\![(\alpha \vee \beta)]\!] := [\![\alpha]\!] \cup [\![\beta]\!]$ für alle $\alpha, \beta \in RA(\Sigma)$
$[\![(\alpha \cdot \beta)]\!] := [\![\alpha]\!] \cdot [\![\beta]\!]$ für alle $\alpha, \beta \in RA(\Sigma)$

Bemerkung: $\Sigma^* = \{b_1 b_2 \ldots b_k \mid b_i \in \Sigma,\ k \geq 0\}$

## Äquivalenz regulärer Ausdrücke

Seien $\alpha, \beta \in RA(\Sigma)$. $\alpha$ ist äquivalent zu $\beta$ bzw. $\alpha \sim \beta$ heißt gültige Formel genau dann wenn $[\![\alpha]\!] = [\![\beta]\!]$ .

## Axiome für die Äquivalenz regulärer Ausdrücke

Seien $\alpha, \beta, \gamma \in RA(\Sigma)$. Dann gilt:

$\alpha \vee (\beta \vee \gamma) \sim (\alpha \vee \beta) \vee \gamma$
$\alpha \cdot (\beta \cdot \gamma) \sim (\alpha \cdot \beta) \cdot \gamma$
$\alpha \vee \beta \sim \beta \vee \alpha$
$\alpha(\beta \vee \gamma) \sim \alpha\beta \vee \alpha\gamma$
$(\alpha \vee \beta)\gamma \sim \alpha\gamma \vee \beta\gamma$
$\alpha \vee \alpha \sim \alpha$
$\Lambda\alpha \sim \Lambda$
$\alpha \vee \Lambda \sim \alpha$

Außerdem gilt die folgende Substitutionsregel:

*Für $\alpha, \beta, \gamma, \gamma', \delta, \nu, \omega \in RA(\Sigma)$ mit $\gamma = \nu\alpha\omega$ und $\gamma' = \nu\beta\omega$ gilt:*
*(S) $\alpha \sim \beta$, $\gamma \sim \delta \rightsquigarrow \gamma' \sim \gamma$ und $\gamma' \sim \delta$*

## 4.3.2 KONSTRUKTION DES VERFAHRENS

Sei $Q = \{q_1, q_2, \ldots, q_n\}$ die Zustandsmenge mit $q_1$ als Anfangszustand und $V = \{l_1, l_2, \ldots, l_r\}$ die Menge aller Verzögerungen eines modifizierten Zustandsdiagramms.

Weiterhin sei eine Transitionsfunktion $t: Q \times V \rightarrow Q$ definiert durch $t(q_i, l) = q_j$ genau dann, wenn das modifizierte Zustandsdiagramm mit der Verzögerung $l$ von $q_i$ nach $q_j$ übergeht.

Eine Menge $\Sigma$ sei definiert durch
$\Sigma = \{ l_j \mid l \in V,\ q_j \in Q$ und es gibt $q_i \in Q$ mit $t(q_i, l) = q_j \}$.

Außerdem werden im weiteren folgende Bezeichnungen verwendet:

$x_i$ , $1 \leq i \leq n$, ist eine Variable für einen regulären Ausdruck über $\Sigma$, welcher vom Zustand $q_i$ ausgehende Transitionswege mit ihren Verzögerungsfolgen beschreibt.

$\alpha_{ij}$ , $1 < i,j < n$ , ist gegeben durch $\alpha_{ij} = \begin{cases} l_j & \text{falls } t(q_i, l) = q_j \\ \Lambda & \text{sonst} \end{cases}$

Aus dem modifizierten Zustandsdiagramm läßt sich dann folgendes Äquivalenzensystem ÄS(0) regulärer Ausdrücke über $\Sigma$ ableiten:

$$x_1 \sim \alpha_{11} x_1 \vee \alpha_{12} x_2 \vee \ldots \vee \alpha_{1n} x_n$$
$$x_2 \sim \alpha_{21} x_1 \vee \alpha_{22} x_2 \vee \ldots \vee \alpha_{2n} x_n$$
$$\vdots$$
$$x_n \sim \alpha_{n1} x_1 \vee \alpha_{n2} x_2 \vee \ldots \vee \alpha_{nn} x_n$$

ÄS(0) ist Ausgangspunkt für das folgende Verfahren, das maximal n mal durchgeführt werden muß. Bei der p-ten Durchführung, $1 \leq p \leq n$, wird ausgehend von einem in der (p-1)-ten Durchführung erstellten Äquivalenzensystem ÄS(p) eine Menge $Z_p$ bestimmt, welche die Menge der Verzögerungszyklen aller Kreise mit genau p Zuständen liefern soll. Anschließend wird aus ÄS(p) das Äquivalenzensystem ÄS(p+1) bestimmt. Wie noch gezeigt werden soll, bricht das Verfahren spätestens nach dem n-ten Durchlauf ab.

Verfahren

Ausgehend vom Äquivalenzensystem ÄS(p), $1 \leq p \leq n$, der Form

$$x_1 \sim \beta_{11} x_1 \vee \beta_{12} x_2 \vee \ldots \vee \beta_{1n} x_n$$
$$x_2 \sim \beta_{21} x_1 \vee \beta_{22} x_2 \vee \ldots \vee \beta_{2n} x_n$$
$$\vdots$$
$$x_n \sim \beta_{n1} x_1 \vee \beta_{n2} x_2 \vee \ldots \vee \beta_{nn} x_n$$

mit $\beta_{ij} \in RA(\Sigma)$ für alle $1 \leq i,j \leq n$ , insbesondere $\beta_{ij} = \alpha_{ij}$ für $p=1$,

ergeben sich die Menge $Z_p$ und das Äquivalenzensystem ÄS(p) wie folgt:

1. Bilden der Mengen $Z_p(i) := [\![\beta_{ii}]\!]$ für alle i, $1 \leq i \leq n$, und

$$Z_p := \bigcup_{i=1}^{n} Z_p(i) \ .$$

Danach ersetzen aller in ÄS(p) vorkommenden $\beta_{ii}x_i$ durch $\Lambda$.

2. Ersetzen der übrigen auf der rechten Seite von ÄS(p) vorkommenden Variablen durch die dazu in ÄS(0) gegebenen äquivalenten regulären Ausdrücke.

3. Umformung des so erhaltenen Äquivalenzensystems mit Hilfe der Axiome für die Äquivalenz regulärer Ausdrücke derart, daß jede Äquivalenz die folgende Form hat:

$x_i \sim \gamma_{i1}x_1 \vee \gamma_{i2}x_2 \vee \ldots \vee \gamma_{in}x_n$, wobei für alle $\gamma_{ij}$, $1 \leq i,j \leq n$, gilt:

$\gamma_{ij} = \Lambda$ oder

$\gamma_{ij} = (\delta_1 \vee \delta_2 \vee \ldots \vee \delta_k)$ , $k \geq 1$, wobei für alle $\delta_q$, $1 \leq q \leq k$ gilt:

$\delta_q = l_1 j_1 l_2 j_2 \ldots l_p j_p l_j$ , $l_{i'} j_{i'} \in \Sigma$ , $1 \leq i' \leq p$, und $l_j \in \Sigma$

4. Ersetzen aller in dem so erhaltenen Äquivalenzensystem vorkommenden Ausdrücke der Form $1_{1j_1}1_{2j_2} \ldots 1_{p_jp}1_j$ durch $\Lambda$, falls $j \in \{j_1, j_2, \ldots, j_p\}$.

Das ergibt das Äquivalenzensystem ÄS(p+1), das nach Konstruktion die gleiche Form wie ÄS(p) hat. Sind auf der rechten Seite von ÄS(p+1) alle Variablen eliminiert, gilt also $\beta_{ij} = \Lambda$ für alle $1 \leq i,j \leq n$, bricht das Verfahren mit $p_o = p$ ab. Andernfalls wird das Verfahren ausgehend von ÄS(p+1), zur Bestimmung von $Z_{p+1}$ und ÄS(p+2) wiederholt.

Wie noch gezeigt werden soll, liefert jede Menge $Z_p(i)$ die Verzögerungszyklen aller von $q_i$ ausgehenden Kreise mit genau p Zuständen, $Z_p$ demnach die Verzögerungszyklen aller Kreise mit genau p Zuständen und

$$Z := \bigcup_{p=1}^{p_o} Z_p$$

die Verzögerungszyklen aller Kreise.

Außerdem soll noch gezeigt werden, daß das Verfahren spätestens nach n Durchläufen abbricht, also gilt: $p_o \leq n$.

Bevor die Korrektheit des Verfahrens bewiesen werden soll, wird dieses an einem Beispiel demonstriert:

Beispiel 4.3.2.

Für das modifizierte Zustandsdiagramm aus Beispiel 4.1.3.b) ergibt sich mit $Q=\{q_1,q_2,q_3\}$, $V=\{2,3,6\}$ und $\Sigma = 2_2,3_3,6_1$ das folgende Äquivalenzensystem $\ddot{A}S(0)=\ddot{A}S(1)$:

$$x_1 \sim 6_1x_1 \vee 2_2x_2 \vee 3_3x_3$$

$$x_2 \sim 6_1x_1$$

$$x_3 \sim 6_1x_1 \vee 3_3x_3$$

$Z_1$ und $\ddot{A}S(2)$ ergeben sich daraus wie folgt:

1. Aus $Z_1(1)=\{6_1\}$, $Z_1(2)=\emptyset$ und $Z_1(3)=\{3_3\}$ folgt $Z_1=\{6_1,3_3\}$.

Danach werden alle $\beta_{ii}x_i$ durch $\Lambda$ ersetzt.

2. Ersetzen der noch auf der rechten Seite von $\ddot{A}S(1)$ vorkommenden Variablen durch die dazu in $\ddot{A}S(0)$ gegebenen äquivalenten regulären Ausdrücke ergibt:

$$x_1 \sim 2_26_1x_1 \vee 3_3(6_1x_1 \vee 3_3x_3)$$

$$x_2 \sim 6_1(6_1x_1 \vee 2_2x_2 \vee 3_3x_3)$$

$$x_3 \sim 6_1(6_1x_1 \vee 2_2x_2 \vee 3_3x_3)$$

3. Umformung gemäß den Axiomen ergibt:

$$x_1 \sim (2_26_1 \vee 3_36_1)x_1 \vee 3_33_3x_3$$

$$x_2 \sim 6_16_1x_1 \vee 6_12_2x_2 \vee 6_13_3x_3$$

$$x_3 \sim 6_16_1x_1 \vee 6_12_2x_2 \vee 6_13_3x_3$$

4. Ersetzen aller $1_1i1_2i$, $1 \leq i \leq 3$, durch $\Lambda$ ergibt dann $\ddot{A}S(2)$:

$$x_1 \sim (2_26_1 \vee 3_36_1)x_1$$

$$x_2 \sim 6_12_2x_2 \vee 6_13_3x_3$$

$$x_3 \sim 6_12_2x_2 \vee 6_13_3x_3$$

Da auf der rechten Seite von ÄS(2) noch Variablen auftreten, wird das Verfahren zur Ermittlung von $Z_2$ und ÄS(3) wiederholt:

1. Mit $Z_2(1)=\{2_26_1,3_36_1\}$, $Z_2(2)=\{6_12_2\}$ und $Z_2(3)=\{6_13_3\}$ folgt $Z_2=\{2_26_1,3_36_1,6_12_2,6_13_3\}$.

Danach werden alle $\beta_{ii}x_i$ durch $\Lambda$ ersetzt.

2. Ersetzen der noch auf der rechten Seite von ÄS(2) vorkommenden Variablen durch die dazu in ÄS(0) gegebenen äquivalenten regulären Ausdrücke ergibt:

$x_1 \sim \Lambda$

$x_2 \sim 6_13_3(6_1x_1 \vee 3_3x_3)$

$x_3 \sim 6_12_26_1x_1$

3. Umformung gemäß den Axiomen ergibt:

$x_1 \sim \Lambda$

$x_2 \sim 6_13_36_1x_1 \quad 6_13_33_3x_3$

$x_3 \sim 6_12_26_1x_1$

4. Ersetzen aller $1_1i1_2j1_3i$ und $1_1i1_2j1_3j$, $1\leq i,j\leq 3$, durch $\Lambda$ ergibt dann ÄS(3):

$x_1 \sim \Lambda$

$x_2 \sim \Lambda$

$x_3 \sim \Lambda$

Da auf der rechten Seite von ÄS(3) keine Variablen mehr auftreten, bricht das Verfahren ab ($p_o=2$).

Damit ergibt sich die Menge $Z = \{6_1, 2_26_1, 3_36_1, 6_12_2, 3_3, 6_13_3\}$ und damit die Menge $\{(6), (2,6), (3,6), (6,2), (3), (6,3)\}$ der Verzögerungszyklen aller Kreise des modifizierten Zustandsdiagramms aus Beispiel 4.1.3.b).

## 4.3.3 BEWEIS DES VERFAHRENS

Zum Beweis der Korrektheit des Verfahrens soll zunächst folgende Behauptung gezeigt werden, aus der dann folgt, daß die Mengen $Z_p$, $1 \leq p \leq p_o$, jeweils die Verzögerungszyklen aller Kreise mit genau p Zuständen bestimmen.

Behauptung:

Jedes Äquivalenzensystem ÄS(p), $1 \leq p \leq n+1$, hat die Form

$$x_1 \sim \beta_{11} x_1 \vee \beta_{12} x_2 \vee \ldots \vee \beta_{1n} x_n$$
$$x_2 \sim \beta_{21} x_1 \vee \beta_{22} x_2 \vee \ldots \vee \beta_{2n} x_n$$
$$\vdots$$
$$x_n \sim \beta_{n1} x_1 \vee \beta_{n2} x_2 \vee \ldots \vee \beta_{nn} x_n$$

mit der folgenden Eigenschaft:

Jedes $\beta_{ij}$, $1 \leq i,j \leq n$, beschreibt nach Konstruktion von ÄS(p) die Menge aller Verzögerungsfolgen zu den Transitionswegen des modifizierten Zustandsdiagramms, die von $q_i$ über p-1 weitere Zustände nach $q_j$ führen, wobei kein Zustand mehrfach durchlaufen wird. Es gilt:

$\beta_{ij} = \Lambda$: Es existiert kein solcher Transitionsweg.

oder

$\beta_{ij} = (\delta_1 \vee \delta_2 \vee \ldots \vee \delta_k)$ , $k \geq 1$, wobei für alle $\delta_q$, $1 \leq q \leq k$ gilt:

$\delta_q = 1_{1j_1} 1_{2j_2} \ldots 1_{pj_p}$ , $j_p = j$, und $i, j_1, j_2, \ldots, j_p$ paarweise verschieden mit der Ausnahme, daß i gleich j sein kann.

Jedes $\delta_q$ bestimmt jeweils Verzögerungen 11,12, ... ,1p entlang eines Transitionswegs, der von $q_i$ über p-1 weitere Zustände nach $q_j$ führt, wobei $q_i$, die p-1 weiteren Zustände und $q_j$ paarweise verschieden sind mit der Ausnahme, daß $q_i$ gleich $q_j$ sein kann.

Beweis durch Induktion über p

Induktionsanfang: Für ÄS(1) gilt die Behauptung nach Definition.

Induktionsschluß: Gelte die Behauptung für ÄS(p), $1 \leq p \leq n$ .

ÄS(p+1) ergibt sich nach dem Verfahren aus ÄS(p) wie folgt:

1.Ersetzen aller $\beta_{ii}x_i$ durch $\Lambda$ .

2. Ersetzen aller noch auf der rechten Seite von ÄS(p) vorkommenden Variablen durch die dazu in ÄS(0) gegebenen äquivalenten regulären Äusdrücke ergibt dann für alle i, $1 \leq i \leq n$:

$$x_i \sim \beta_{i1}(\alpha_{11}\, x_1 \vee \alpha_{12}\, x_2 \vee \ldots \vee \alpha_{1n}\, x_n) \vee$$

$$\vdots$$

$$\vee \beta_{i,i-1}(\alpha_{i-1,1}\, x_1 \vee \alpha_{i-1,2}\, x_2 \vee \ldots \vee \alpha_{i-1,n}\, x_n)$$

$$\vee \beta_{i,i+1}(\alpha_{i+1,1}\, x_1 \quad \alpha_{i+1,2}\, x_2 \vee \ldots \vee \alpha_{i+1,n}\, x_n) \vee$$

$$\vdots$$

$$\vee \beta_{in}(\alpha_{n1}\, x_1 \vee \alpha_{n2}\, x_2 \vee \ldots \vee \alpha_{nn}\, x_n)$$

3. Umformung mit Hilfe der Axiome ergibt für alle i, $1 \leq i \leq n$:

$x_i \sim \gamma_{i1}x_1 \vee \gamma_{i2}x_2 \vee \ldots \vee \gamma_{in}x_n$ , wobei für alle $\gamma_{ij}, 1 \leq j \leq n$ gilt:

$\gamma_{ij} = (\beta_{i1}\alpha_{1j} \vee \ldots \vee \beta_{i,i-1}\alpha_{i-1,j} \vee \beta_{i,i+1}\alpha_{i+1,j} \vee \ldots \vee \beta_{in}\alpha_{nj})$
für alle $1 \leq i,j \leq n$, wobei für jedes $\beta_{ik}\alpha_{kj}$, $1 \leq k \leq n$, $i \neq k$ gilt:

$\beta_{ik}\alpha_{kj} = \Lambda$ oder

$\beta_{ik}\alpha_{kj} = \delta_1 \vee \delta_2 \vee \ldots \vee \delta_m$, $m \geq 1$, wobei für alle $\delta_q, 1 \leq q \leq m$, gilt:

$\delta_q = l1_{j1}l2_{j2}\ldots lp_{jp}l_j$ , $jp = k$ und $i, j1, j2, \ldots, jp$ paarweise verschieden.

$\beta_{ik}\alpha_{kj}$ bestimmt die Verzögerungen $l1, l2, \ldots, lp, 1$ entlang aller Transitionswege, die von $q_i$ über (p-1) weitere Zustände nach $q_k$ und von dort nach $q_j$ führen, wobei $q_i$, die weiteren Zustände und $q_k$ nach Induktionsannahme und Schritt 1 des Verfahrens paarweise verschieden sind.

4. Ersetzen aller Ausdrücke $l1_{j1}l2_{j2} \ldots lp_{jp}l_j$ mit $j \in \{j1, j2, \ldots jp\}$ durch $\Lambda$.

Für alle $\gamma_{ij}$ mit $1 \leq i,j \leq n$ gilt dann:

$\gamma_{ij} = \Lambda$ oder

$\gamma_{ij} = (\delta_1 \vee \delta_2 \vee \ldots \vee \delta_m)$ , $m \geq 1$, wobei für alle $\delta_q$, $1 \leq q \leq m$, gilt:

$\delta_q = l1_{j1}l2_{j2} \ldots lp_{jp}l_j$ , wobei $i, j1, \ldots, jp, j$ paarweise verschieden sind mit der Ausnahme, daß i gleich j sein kann.

$\gamma_{ij}$ bestimmt die Verzögerungen $l1, l2, \ldots, lp, l$ entlang aller Transitionswege, die von $q_i$ über p weitere Zustände nach $q_j$ führen, wobei $q_i$, die p weiteren Zustände und $q_j$ paarweise verschieden sind mit der Ausnahme, daß $q_i$ gleich $q_j$ sein kann.

Nach der Umbenennung $\beta_{ij} := \gamma_{ij}$ ergibt sich das Äquivalenzensystem ÄS(p+1) mit den geforderten Eigenschaften.

Aus der Behauptung folgt direkt für jedes Äquivalenzensystem $ÄS(p)$, $1 \leq p \leq p_o$ :

$\beta_{ii}$, $1 \leq i \leq n$, liefert die Verzögerungszyklen aller Kreise mit genau p Zuständen, die vom Zustand $q_i$ ausgehen. Es gilt:

$\beta_{ii} = \Lambda$: Es existiert kein derartiger Kreis.

oder

$\beta_{ii} = (\delta_1 \vee \delta_2 \vee \ldots \vee \delta_k)$ , $k \geq 1$, wobei für alle $\delta_q$, $1 \leq q \leq k$ gilt:

$\delta_q = l1_{j1} l2_{j2} \ldots lp_{jp}$ mit $jp = i$ und $j1, j2, \ldots, jp$ paarweise verschieden.

Jedes $\delta_q$ beschreibt jeweils einen Kreis, der von $q_i$ aus nacheinander Zustände $q_{j1}, q_{j2}, \ldots, q_{j,p-1}$ durchläuft und wieder nach $q_i$ zurückkehrt, wobei der Verzögerungszyklus des Kreises $(l1, l2, \ldots, lp)$ ist.

Die Menge $Z_p(i) = [\![\beta_{ii}]\!]$ liefert dann alle Kreise, die von $q_i$ ausgehen und genau p Zustände enthalten, die Menge $Z_p$ alle Kreise mit genau p Zuständen.

Als nächstes soll noch gezeigt werden, daß das Verfahren nach spätestens $p_o=n$ Durchläufen abbricht.

Behauptung:

Spätestens für das nach n Durchläufen bestimmte Äquivalenzensystem ÄS(n+1) gilt: $x_i \sim \Lambda$ für alle i, $1 \leq i \leq n$.

Beweis:

Entweder sind schon vor dem n-ten Durchlauf des Verfahrens alle Ausdrücke auf der rechten Seite eines Äquivalenzensystems ÄS(p), $p<n$, durch $\Lambda$ ersetzt. Dann gilt die Behauptung.

Ist dies nicht der Fall, tritt auf der rechten Seite von ÄS(n) mindestens ein Ausdruck der Form $\beta_{ij}x_j$ auf, wobei $\beta_{ij}$ mindestens eine Folge von Verzögerungen entlang eines Transitionswegs bestimmt, der vom Zustand $q_i$ über n-1 weitere Zustände nach $q_j$ führt, wobei $q_i$, die (n-1) weiteren Zustände und $q_j$ paarweise verschieden sind, mit der Ausnahme, daß $q_i$ gleich $q_j$ sein kann. Bei n Zuständen insgesamt folgt damit, daß $q_i$ gleich $q_j$ ist. Das bedeutet, daß $\beta_{ij}x_j$ gleich $\beta_{ii}x_i$ ist. Im ersten Schritt des n-ten Durchlaufs des Verfahrens werden jedoch alle $\beta_{ii}x_i$ durch $\Lambda$ ersetzt, womit alle Variablen auf der rechten Seite eliminiert sind und das Verfahren abbricht.

## 4.3.4 BESTIMMUNG ALLER OPTIMALEN VERZÖGERUNGSFOLGEN

Die optimalen Verzögerungsfolgen werden aus der Menge Z (siehe Abschnitt 4.3.2.) ermittelt, welche die Verzögerungszyklen aller Kreise bestimmt. Das geschieht wie folgt:

1. Bestimmung der Menge OZ der Elemente aus Z, welche die optimalen Verzögerungszyklen aller Kreise beschreiben:

$$OZ = \{ l1_{j1} l2_{j2} \ldots lk_{jk} \in Z \mid (\sum_{i=1}^{k} li)/k = MIN((\sum_{j=1}^{m} l'm)/m), \; l'1_{j1} l'2_{j2} \ldots l'm_{jm} \in Z \}$$

OZ enthält also die Elemente aus Z, welche die Verzögerungszyklen bestimmen, deren mittlere Verzögerung ein Minimum annimmt.

Bemerkung:

Für jedes $l1_{j1} l2_{j2} \ldots lk_{jk} \in OZ$ gilt: $q_{jk}$ ist jeweils Ausgangszustand eines Kreises mit einem optimalem Verzögerungszyklus (l1,l2,...,lk).

2. Zunächst wird die folgende Menge

$$O_1 = \{ (l1,l2,\ldots,lk) \mid l1_{j1} l2_{j2} \ldots lk_{jk} \in OZ \text{ mit } jk=1 \}$$

aller optimalen Verzögerungszyklen zu Kreisen, die vom Anfangszustand ausgehen, gebildet.

Danach wird die Menge der Verzögerungsfolgen entlang der Transitionswege ermittelt, die vom Anfangszustand zu einem Ausgangszustand $q_i$, $2 \leq i \leq n$, eines Kreises mit optimalem Verzögerungszyklus führen und für welche die Summe ihrer Verzögerungen ein Minimum annimmt. Dabei genügt es bereits, sich auf einfache Transitionswege zu beschränken. Ein Transitionsweg ist einfach, wenn er keinen Zustand mehrfach durchläuft.

Die Menge der Verzögerungsfolgen entlang aller vom Anfangszustand ausgehenden einfachen Transitionswege läßt sich bereits, wie im folgenden gezeigt werden soll, aus der Menge

$$Z(1) := \bigcup_{p=1}^{p_o} Z_{p(1)}$$

ermitteln, welche die Menge aller vom Anfangszustand ausgehenden Kreise bestimmt (vgl. Abschnitt 4.3.2.).

Behauptung:

Zu jedem einfachen Transitionsweg, der vom Anfangszustand $q_1$ zu einem Zustand $q_i$, $2 \leq i \leq n$, führt, gibt es mindestens einen Kreis, der von $q_1$ ausgeht und mit dem einfachen Transitionsweg beginnt.

Beweis:

Jeder Transitionsweg, der mit einem einfachen Transitionsweg beginnt, der von $q_1$ nach $q_i$ führt und von dort direkt mit einer Transition nach $q_1$ zurückführt, bildet bereits einen Kreis. Da von jedem Zustand $q_i \in Q$ eine Transition zum Anfangszustand führt, gilt die Behauptung für alle i, $2 \leq i \leq n$.

Daraus ergibt sich, daß die folgende Menge F bereits die Verzögerungsfolgen entlang aller einfachen Transitionswege von $q_1$ zu einem Ausgangszustand $q_i$, $2 \leq i \leq n$, eines Kreises mit optimalem Verzögerungszyklus bestimmt:

$$F = \{ l1_{i1} l2_{i2} \ldots lk_{ik} \mid q_{ik} \neq q_1 \text{ Ausgangszustand eines Kreises mit optimalem Verzögerungszyklus, } l1_{i1} \ldots lk_{ik} l'1_{j1} \ldots l'm_{jm} \in Z(1) \}$$

Daraus läßt sich dann die folgende Menge OF bestimmen:

$$OF = \{ l1_{i1} l2_{i2} \ldots lk_{ik} \in F \mid \sum_{i'=1}^{k} li' = MIN( \sum_{j'=1}^{m} l'j' ), \; l'1_{j1} l'2_{j2} \ldots l'm_{jm} \in F \}$$

OF bestimmt die Menge aller Verzögerungsfolgen, die vom Anfangszustand zu einem Ausgangszustand $q_i$, $2 \leq i \leq n$, eines Kreises mit optimalem Verzögerungszyklus führen und für welche die Summe ihrer Verzögerungen ein Minimum annimmt.

Aus den Mengen OF und OZ wird die folgende Menge $O_2$ aller Verzögerungsfolgen mit optimalen Verzögerungszyklen zu Kreisen, die von einem Zustand $q_{ik}=q_{jm} \neq q_1$ ausgehen, gebildet:

$$O_2 = \{ <l1, l2, \ldots, lk, l'1, l'2, \ldots l'm, l'1, l'2, \ldots l'm, l'1, \ldots> \mid l1_{i1} l2_{i2} \ldots lk_{ik} \in OF, \; l'1_{j1} l'2_{j2} \ldots l'm_{jm} \in OZ \text{ und } lk_{ik} = l'm_{jm} \}$$

Die Menge O aller optimalen Verzögerungsfolgen ergibt sich dann aus den Mengen O1 und O2 wie folgt:

a) Fall $O_1=\emptyset$: $O=O_2$

b) Fall $O_2=\emptyset$: $O=O_1$

c) Fall $O_1 \neq \emptyset$ und $O_2 \neq \emptyset$:

$O=O_2$ , falls $(\sum_{i=1}^{k} li)/k < (\sum_{j=1}^{m} l'j)/m$

für alle $<l1,l2,...,lk,l'1,l'2,...,l'm,l'1,l'2, ... > \epsilon O_2$

$O=O_1 \cup O_2$ , falls $(\sum_{i=1}^{k} li)/k = (\sum_{j=1}^{m} l'j)/m$

für alle $<l1,l2,...,lk,l'1,l'2,...,l'm,l'1,l'2, ... > \epsilon O_2$

$O=O_1$ sonst

Aus der Menge O aller optimalen Verzögerungsfolgen wird schließlich eine Verzögerungsfolge mit einem Verzögerungszyklus ausgewählt, dessen Anzahl von Verzögerungen und damit dessen Umlaufzahl ein Minimum annimmt. Für den Verzögerungszyklus (l1,l2,...,lk) einer solchen Verzögerungsfolge aus O muß also gelten:

k=MIN(m), (l'1,l'2,...,l'm) Verzögerungszyklus einer Folge aus O.

Beispiel 4.3.4.

Für das modifizierte Zustandsdiagramm aus Beispiel 4.1.3.b) folgt mit Beispiel 4.3.2.:

$$Z = \{6_1,\ 2_2 6_1,\ 3_3 6_1,\ 6_1 2_2,\ 3_3,\ 6_1 3_3\}$$

1. Ermittlung der Menge OZ der Elemente aus Z, die einen Verzögerungszyklus bestimmen, dessen mittlere Verzögerung ein Minimum annimmt:

| Element aus Z | mittlere Verzögerung |
|---|---|
| $6_1$ | 6 |
| $2_2 6_1$ | 4 |
| $3_3 6_1$ | 9/2 |
| $6_1 2_2$ | 4 |
| $3_3$ | 3 |
| $6_1 3_3$ | 9/2 |

Daraus folgt: $OZ = \{3_3\}$

2. Der Zustand $q_3$ ist Ausgangszustand des einzigen Kreises mit optimalem Verzögerungszyklus, nämlich (3). Daraus folgt zunächst $O_1 = \emptyset$.

Nach Beispiel 4.3.2. gilt: $Z(1) = Z_1(1) \cup Z_2(1) = \{6_1,\ 2_2 6_1,\ 3_3 6_1\}$

Damit ergeben sich die folgenden Mengen F, OF, $O_2$ und O:

$F = OF = \{3_3\}$ , $O = O_2 = \{<3,3,\ldots>\} = \{(3)\}$.

O enthält also nur eine optimale Verzögerungsfolge, nämlich <3,3,...>. Diese bildet den Verzögerungszyklus (3).

## 4.4 BEMERKUNGEN ZUR KOMPLEXITÄT

Im folgenden sollen einige Angaben gemacht werden, aus denen sich Aussagen über die Komplexität der einzelnen Stufen des gesamten Verfahrens zur Entwicklung einer optimalen Scheduling-Strategie für statische Pipelines ergeben.

1. Für die Konstruktion des modifizierten Zustandsdiagramms nach Abschnitt 4.1.3. gilt:

a) Die Bestimmung des Anfangskollisionsvektors (vgl. Abschnitt 4.1.1.) erfordert

$$k \cdot \sum_{i=1}^{m-1} i = k \cdot (m-1) \cdot m/2 = O(m^2)$$

Vergleichsoperationen zur Überprüfung, ob eine Kollision vorliegt oder nicht. Dabei ist k die Anzahl der Pipelinesegmente und $m_\tau$ die Ausführungszeit der Reservierungstafel.

b) Ist m die Länge des Anfangskollisionsvektors, n die Anzahl Zustände und $K_1$ die Anzahl der Nullen links von der letzten Eins des Anfangskollisionsvektors. Diese Eins befinde sich auf der Position j, $0 \leq j \leq m-1$. $K_i$, $2 \leq i \leq n$, sei dann die Anzahl der Nullen links von der j-ten Position des Kollisionsvektors des i-ten Zustands. Dann gilt: Der zweite Schritt des Verfahrens muß

$$S = \sum_{i=1}^{n} K_i$$

mal durchgeführt werden. Für $n>1$ gilt mit $K_i < K_1 \leq n$ für $2 \leq i \leq n$: $S < n^2$ .

Jede Durchführung des zweiten Schritts beinhaltet folgende Operationen:

- eine Schiebeoperation
- eine bitweise logische Oder-Verknüpfung
- maximal n Vergleiche von Kollisionsvektoren
- das Schaffen einer neuen Transition und eventuell eines neuen Zustands

Damit ergeben sich insgesamt für den zweiten Schritt:

- S Schiebebefehle
- S bitweise logische Oder-Verknüpfungen
- maximal S·n Vergleiche
- S Transitionen
- n-1 neue Zustände

c) Schritt 3 des Verfahrens erfordert das Schaffen von n Transitionen.

d) Die Vereinfachung nach Bemerkung 4 aus Abschnitt 4.1.3. kann durch Abfragen jeweils in Schritt 2 und Schritt 3 des Konstruktionsverfahrens mit durchgeführt werden. Das bedeuted maximal S zusätzliche Vergleiche; dafür brauchen jedoch im allgemeinen weniger Transitionen geschaffen zu werden.

2. Im folgenden soll der Aufwand für die in Abschnitt 4.3.2. und 4.3.4. angegebenen Verfahren nach oben hin abgeschätzt werden. Dazu wird der ungünstigste Fall angenommen, daß von jedem Zustand des modifizierten und nach Bemerkung 4, Abschnitt 4.1.3., vereinfachten Zustandsdiagramms n Transitionen ausgehen, wobei n die Anzahl der Zustände ist.

Dann gilt für den p-ten Durchlauf, $1 \leq p \leq n$, des Verfahrens aus Abschnitt 4.3.2.:

a) Für das Äquivalenzensystem ÄS(p) gilt:

$$\mathrm{card}\left(\bigcup_{i=1}^{n} \bigcup_{j=1}^{n} [\![\beta_{ij}]\!]\right) = (n-p+1) \cdot \frac{n!}{(n-p)!}$$

Im ersten Schritt wird die Menge $Z_p$ mit

$$\mathrm{card}(Z_p) = \frac{n!}{(n-p)!}$$

gebildet und alle $\beta_{ij}$ aus ÄS(p) eliminiert. Für p=n kann das Verfahren an dieser Stelle beendet werden. Für p<n gilt dann für die noch verbleibenden Ausdrücke $\beta_{ij}$:

$$\mathrm{card}\left(\bigcup_{i=1}^{n} \bigcup_{\substack{j=1 \\ j \neq i}}^{n} \beta_{ij}\right) = \frac{n!}{(n-p-1)!}$$

b) Im zweiten Schritt werden $n \cdot (n-1)$ Variablen $x_i$, $1 \leq i \leq n$, durch $\alpha_{i1}x_1 \vee \alpha_{i2}x_2 \vee \ldots \vee \alpha_{in}x_n$ mit

$$\mathrm{card}\left(\bigcup_{j=1}^{n} [\![\alpha_{ij}]\!]\right) = n$$

ersetzt.

c) Für die im dritten Schritt gebildeten Ausdrücke $\gamma_{ij}$ gilt für p<n:

$$\text{card}\left(\bigcup_{i=1}^{n}\bigcup_{j=1}^{n}[\![\gamma_{ij}]\!]\right) = n\cdot\frac{n!}{(n-p-1)!}$$

d)

Aus der Menge $\bigcup_{i=1}^{n}\bigcup_{j=1}^{n}[\![\gamma_{ij}]\!]$ werden im vierten Schritt für p<n

$p\cdot\frac{n!}{(n-p-1)!}$ Elemente eliminiert; dazu sind

$p\cdot n\cdot\frac{n!}{(n-p-1)!}$ Vergleichsoperationen erforderlich.

Für p<n ergibt sich dann das Äquivalenzensystem ÄS(p+1) mit

$$\text{card}\left(\bigcup_{i=1}^{n}\bigcup_{j=1}^{n}[\![\beta_{ij}]\!]\right) = (n-p)\cdot\frac{n!}{(n-p-1)!}$$

Für das Verfahren aus Abschnitt 4.3.4. gilt:

a) Die Ermittlung der Menge OZ erfordert mit

$$\text{card}(Z) = \sum_{p=1}^{n}\frac{n!}{(n-p)!}$$

$\sum_{p=2}^{n}(p-1)\cdot\frac{n!}{(n-p)!}$ Additionen,

$\sum_{p=2}^{n}\frac{n!}{(n-p)!}$ Divisionen und

$\sum_{p=1}^{n}\frac{n!}{(n-p)!} - 1$ Vergleiche.

b) Sei card(OZ) = C ; die Ermittlung von $O_1$ erfordert dann C Vergleiche. Weiterhin sei $card(O_1)=C_1$. Mit

$$card(Z(1)) = \left(\sum_{p=1}^{n} \frac{n!}{(n-p)!}\right)/n = \sum_{p=1}^{n} \frac{(n-1)!}{(n-p)!}$$

sind für die Ermittlung der Menge F maximal

$$(C-C_1)\cdot\left(\sum_{p=2}^{n} (p-1)\cdot\frac{(n-1)!}{(n-p)!}\right)$$

Vergleiche erforderlich.

Für die Menge F gilt:

$$card(F) \leq (C-C_1)\cdot\left(\sum_{p=2}^{n} \frac{(n-1)!}{(n-p)!}\right)$$

Die Ermittlung der Menge OF erfordert maximal

$$(C-C_1)\cdot\left(\sum_{p=3}^{n} (p-2)\cdot\frac{(n-1)!}{(n-p)!}\right)$$ Additionen und

$$(C-C_1)\cdot\left(\sum_{p=2}^{n} \frac{(n-1)!}{(n-p)!}\right) - 1$$ Vergleiche .

Zur Ermittlung der Menge $O_2$ sind $(C-C_1)\cdot card(OF)$ Vergleiche erforderlich.

Zur Ermittlung der Menge O sind maximal ein Vergleich und eine Division erforderlich; die übrigen Operationen wurden schon zur Bestimmung der Mengen OZ und OF durchgeführt.

Die Auswahl eines geeigneten Elements aus O erfordert schließlich card(O)-1 Vergleiche.

Für die Verfahren aus Abschnitt 4.3.2. und 4.3.4. ergibt sich nach den obigen Angaben für großes n ein sehr hoher Aufwand, der in der Praxis nicht mehr vertretbar sein dürfte. Zu beachten ist jedoch, daß der Grenzfall, daß von jedem Zustand n Transitionen ausgehen, im allgemeinen bei weitem nicht erreicht wird, so daß der Aufwand in den meisten Fällen deutlich geringer ist. Für eine kleine Anzahl von Zuständen, zum Beispiel $n\leq 10$, oder einer mittleren Anzahl von Zuständen, zum Beispiel $n\leq 20$, und wenigen Transitionen, zum Beispiel insgesamt $2\cdot n$, dürfte der Aufwand bei der Verarbeitung langer Eingabevektoren auf jeden Fall vertretbar sein.

# 5.0 ANWENDUNG DES VERFAHRENS AUF DEN PIPELINE-RECHNER AP-190L

Im folgenden soll als Anwendungsbeispiel gezeigt werden, wie die in den Abschnitten 3 und 4 vorgestellten Hilfsmittel bei der Optimierung von Assembler-Programmen für den Pipeline-Rechner AP-190L der Firma Floating Point Systems, INC., Portland/Oregon, eingesetzt werden können. Dazu soll als erstes eine Beschreibung des Rechners gegeben werden.

## 5.1 BESCHREIBUNG DES AP-190L

Der AP-190L, im folgenden mit AP bezeichnet, kann nur über einen Steuerrechner (Host) betrieben werden. Der Host-Rechner muß alle auf dem AP auszuführenden Programme und Daten an diesen übergeben und die Ausführung starten. Der AP ist vor allem zur Entlastung der Host-Zentraleinheit bei rechenintensiven Problemen gedacht. Die Bezeichnung "Array-Prozessor" (AP) deutet an, daß sich der AP vor allem zur Verarbeitung von Datenfeldern (Vektoren, Matrizen) eignet; er ist kein Array-Prozessor (Feldrechner, Parallelprozessor) im eigentlichen Sinn (vgl. Abschnitt 1).

Im folgenden werden zunächst die wichtigsten Bestandteile der Hardware beschrieben, da dies für das weitere Verständnis erforderlich ist. Für genauere Erläuterungen sei auf die FPS Reference Manuals (siehe /FPS 78/) verwiesen.

### 5.1.1 HARDWARE

Der AP besitzt zwei jeweils als Pipeline organisierte Gleitkomma-Recheneinheiten, mehrere Register- und Speichereinheiten, eine Einheit zur Berechnung von Programmadressen und einen Ein/Ausgabeteil, auf den jedoch im weiteren nicht eingegangen wird. Die einzelnen Einheiten des AP sind untereinander durch verschiedene Datenbusse miteinander verbunden. Der AP arbeitet mit einem 38-Bit-Gleitkomma-Datenformat (10-Bit-Exponent, 28-Bit-Mantisse), 16-Bit-Adressen und 64-Bit-Instruktionen; die Taktzeit beträgt für alle Einheiten 167 ns . Das folgende Bild soll einen Überblick über den Aufbau des AP geben. Danach erfolgt eine Kurzbeschreibung der einzelnen Einheiten.

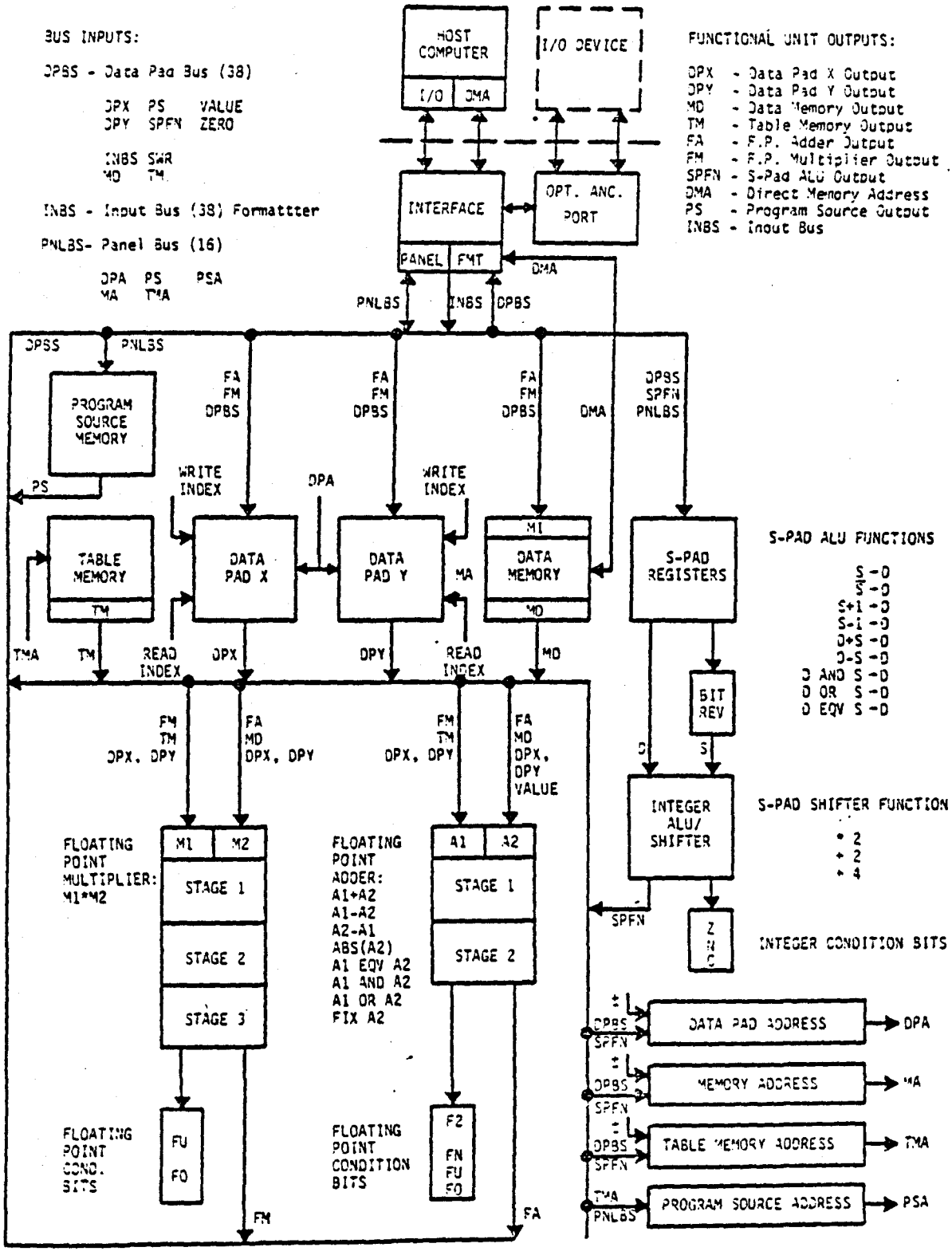



Bild 5.1.1. Aufbau des AP-190L

Gleitkomma-Recheneinheiten

Der AP enthält einen 38-Bit-Gleitkomma-Addierer und einen 38-Bit-Gleitkomma-Multiplizierer.

Der Addierer ist eine aus zwei Segmenten bestehende lineare Pipeline mit zwei Eingabebussen A1 und A2 und einem Ausgabebus FA. Neben der Addition können im Addierer weitere Operationen mit einem oder zwei Operanden ausgeführt werden (vgl. Bild 5.1.1.)

Der Multiplizierer ist eine aus drei Segmenten bestehende lineare Pipeline mit zwei Eingabebussen M1 und M2 und einem Ausgabebus FM.

Mit jedem Takt kann im Addierer und im Multiplizierer jeweils eine Operation gestartet werden. Von welchen Einheiten Eingaben in den Addierer bzw. Multiplizierer erfolgen können, ist Bild 5.1.1. zu entnehmen.

Register und Speicher

Der AP enthält zwei Datenregistersätze (Data Pad X - DPX, Data Pad Y - DPY), einen Adreßregistersatz (S-Pad), einen Datenspeicher (MD), einen Tabellenspeicher (TM), einen Programmspeicher (PS) und vier 16-Bit-Register für die Programm-Adresse (PSA), die Datenspeicher-Adresse (MA), die Tabellenspeicher-Adresse (TMA) und die Datenregister-Adresse (DPA).

Die beiden Datenregistersätze bestehen aus jeweils 32 38-Bit-Registern und dienen als Zwischenspeicher für die Recheneinheiten bzw. als Datenpuffer zwischen dem Datenspeicher und den Recheneinheiten.

Es ist möglich, in jedem Zyklus in jeden Registersatz einen Wert zu schreiben und einen Wert zu lesen. Dabei ist auch das Lesen und Beschreiben desselben Registers im gleichen Zyklus möglich (read before write). Die Zugriffsadressen der Datenregister ergeben sich durch Addition der 3-Bit-Schreib- bzw. Leseindizes XR, YR, XW und YW zur Datenregister-Adresse (DPA). Ein Zugriff kann daher nur auf die Register mit den Adressen DPA-4 bis DPA+3 erfolgen.

In welche Einheiten Werte aus den Datenregistern gebracht werden können bzw. von welchen Werte in die Datenregister geschrieben werden können, ist Bild 5.1.1. zu entnehmen.

Der Adreßregistersatz (S-Pad) besteht aus 16 16-Bit-Registern zur Aufnahme von Adressen oder ganzzahligen 16-Bit-Werten z.B. zur Schleifenzählung.

Der Datenspeicher (Main Data Memory - MD) ist ein RAM-Speicher (Random Access Memory) und dient zur Aufnahme von 64K 38-Bit-Daten. Neben dem eigentlichen Speicherteil enthält der Datenspeicher ein Eingaberegister (Memory Input Buffer - MI) und ein Ausgaberegister (Memory Destination Buffer - MD). Der Datenspeicher wird über das MA-Register (Memory Address) adressiert und ist vierfach verschränkt, wobei die Speicherbänke wie folgt aufgeteilt sind:

| | | | | | |
|---|---|---|---|---|---|
| Bank 0: Speicheradressen | 0, | 2, | 4, | ... | , 32766 |
| Bank 1: Speicheradressen | 1, | 3, | 5, | ... | , 32767 |
| Bank 2: Speicheradressen | 32768, | 32770, | 32772, | ... | , 64534 |
| Bank 3: Speicheradressen | 32769, | 32771, | 32773, | ... | , 64535 |

Zu jedem zweiten Takt kann auf dieselbe Speicherbank und zu jedem Takt auf verschiedene Speicherbänke zugegriffen werden. Wird jeweils zu zwei aufeinanderfolgenden Takten ein Zugriffsbefehl auf dieselbe Speicherbank erteilt, wird intern ein Wartezyklus eingeschoben, in welchem der AP keinen anderen Befehl ausführt. Es dauert drei Zyklen, bis ein Wert aus dem Speicher in das MD-Register geholt bzw. ein Wert aus dem MI-Register in den Speicher geschrieben worden ist.

Bild 5.1.1. ist zu entnehmen, aus welchen Einheiten Werte in den Speicher bzw. in welche Einheiten Werte aus dem Speicher gebracht werden können.

Der Tabellenspeicher (Table Memory - TM) besteht aus einem ROM-Teil (Read Only Memory) und einem RAM-Teil (optional).

Der ROM-Teil (TMROM) enthält 2,5K 38-Bit-Konstanten, insbesondere eine Cosinus-Tabelle zur schnellen Durchführung von Fast-Fourier-Transformationen.

Der RAM-Teil (TMRAM) kann als Hilfsspeicher benutzt werden und umfaßt 2K 38-Bit-Speicherplätze. Der RAM-Teil enthält zusätzlich ein Eingaberegister (Table Memory Input Buffer - TMI) zur Aufnahme von Daten, die in den RAM-Teil gebracht werden sollen.

Der Tabellenspeicher enthält außerdem ein Ausgaberegister (Table Memory Buffer - TM) zur Aufnahme von Daten, die aus dem ROM- oder RAM-Teil gelesen worden sind. Die Adressierung des Tabellenspeichers erfolgt über das TMA-Register (Table Memory Address). Zu jedem Takt kann ein Zugriff auf den Tabellenspeicher erfolgen. Zwei Takte nach dem Beginn des Zugriffs erscheint ein gelesener Wert im TM-Register bzw. wurde ein im TMI-Register befindlicher Wert in den RAM-Teil gebracht.

Der Programmspeicher ist ein RAM-Speicher und dient zur Aufnahme von 4K 64-Bit-Instruktionen. Der Programmspeicher wird über das PSA-Register adressiert, in welches während der Ausführung einer Instruktion automatisch die Adresse der nächsten Instruktion geladen wird.

## Adreßeinheit

Die Adreßeinheit (S-Pad Unit) dient in der Hauptsache zur Berechnung von Datenadressen und zur Schleifenzählung. Sie enthält eine 16-Bit-arithmetisch-logische Einheit (S-Pad ALU) zur Durchführung der aus Bild 5.1.1. zu entnehmenden Operationen auf ganzzahligen 16-Bit-Werten aus den S-Pad-Registern S (Source) und D (Destination). Das Ergebnis einer S-Pad-Operation, die sogenannte S-Pad Funktion (SPFN) wird im S-Pad D-Register abgelegt und kann außerdem in die Register MA, TMA und DPA geladen werden.

## Busstruktur

Die verschiedenen Einheiten des AP sind über mehrere 38-Bit-Datenbusse miteinander verbunden. Vier Operandenbusse (A1, A2, M1, M2) liefern die Eingaben für die Gleitkomma-Recheneinheiten, zwei Ergebnisbusse (FA, FM) führen die Ergebnisse aus den Gleitkomma-Recheneinheiten zurück und ein weiterer Datenbus (Data Pad Bus - DPBS) verbindet die verschiedenen Register- und Speichereinheiten miteinander. Bild 5.1.1. gibt eine Übersicht darüber, wie die einzelnen Einheiten des AP miteinander verbunden sind.

## 5.1.2 ASSEMBLER-PROGRAMMIERUNG

Ein APAL-Programm (Array Processor Assembly Language) besteht aus einer Folge von Assembler-Direktiven und AP-Instruktionen. Auf die Assembler-Direktiven wird im weiteren nicht eingegangen. Im folgenden soll der Aufbau einer AP-Instruktion beschrieben werden. Für genauere Erläuterungen sei auf die FPS Reference Manuals (siehe /FPS 78/) verwiesen.

Ein AP-Instruktionswort hat eine Länge von 64 Bit und ist in mehrere Felder für die einzelnen Befehlsgruppen der verschiedenen AP-Einheiten unterteilt. Bild 5.1.2. zeigt den Aufbau einer AP-Instruktion.

Übersicht über die einzelnen Befehlsgruppen:

| Befehlsgruppe | Funktion |
|---|---|
| S-Pad | Kontrolle der Adreßeinheit |
| Adder | Kontrolle des Addierers |
| Branch | Kontrolle von Programmverzweigungen |
| Data Pad | Kontrolle der Datenregister und des Data Pad Bus |
| Multiplier | Kontrolle des Multiplizierers |
| Memory | Kontrolle der Speichereinheiten |

Jedes Feld für eine Befehlsgruppe ist wiederum in Unterfelder unterteilt (siehe Bild 5.1.2.). Die Unterfelder enthalten den Befehlscode und die Operandenangaben für die Befehle der jeweiligen Befehlsgruppe. Die Felder 'SPEC OPER', 'I/O' und 'VALUE' (siehe Bild 5.1.2.) sind für weitere Befehle reserviert, auf die im folgenden jedoch nicht eingegangen wird.

Der Aufbau einer AP-Instruktion ermöglicht die gleichzeitige Ausführung von bis zu zehn Operationen:

- eine S-Pad Operation
- eine Operation des Addierers
- eine Programmverzweigung
- laden eines DPX-Registers
- laden eines DPY-Registers
- eine Datenübertragung über den Data Pad Bus
- eine Multiplikation
- einen Zugriff auf den MD-Speicher
- ändern des DPA-Registers
- einen Zugriff auf den Tabellenspeicher

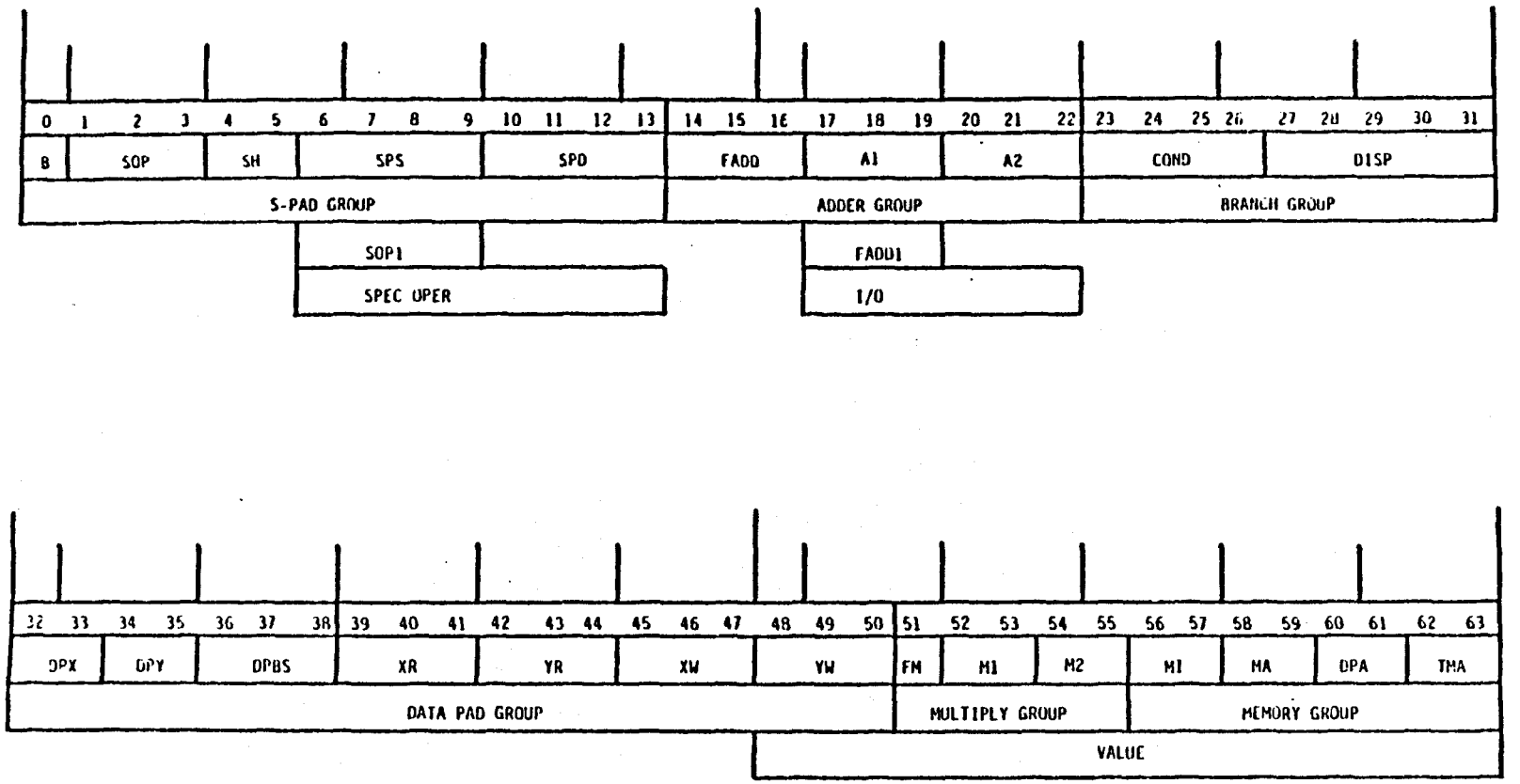



Bild 5.1.2. Aufbau einer AP-Instruktion

Da bei der Assembler-Programmierung des AP eine Fülle von Hardware-Gegebenheiten zu berücksichtigen ist, empfiehlt es sich, für den zeitlichen Ablauf von Aktivitäten auf dem AP zur Lösung eines Problems eine Reservierungstafel zu erstellen, in der vertikal die wichtigsten Elemente des AP aufgetragen sind.

Bild 5.1.3. zeigt die Reservierungstafel für die Linearkombination dreier Vektoren X, Y und Z nach der Formel

$$V(I) = A \cdot X(I) + B \cdot Y(I) + C \cdot Z(I) \quad \text{für } I=1(1)N$$

In der Reservierungstafel ist genau das Muster für die Berechnung eines Vektorelements V(I) angegeben. Die Vektorelemente X(I), Y(I), Z(I) und V(I) werden in der Reservierungstafel abkürzend mit X, Y, Z und V bezeichnet, deren Adressen mit IX, IY, IZ und IV. Außerdem wird vorausgesetzt, daß die Konstanten A, B und C vor den Berechnungen in die Datenregister DPX(0), DPY(0) und DPY(1) geladen wurden. Die Bedeutung der Eintragungen in einer Reservierungstafel für den AP wird anschließend angegeben.

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA MEMORY | MI | . | | | | | | | | | | | V |
| | MA | IX+1 | IY+1 | IZ+1 | | | | | | | | | IV+1 |
| | | | X | Y | Z | | | | | | | | |
| | | | | X | Y | Z | | | | | | | |
| | MD | | | | X | Y | Z | | | | | | |
| TABLE MEMORY | TMA | | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| | TM | | | | | | | | | | | | |
| DATA PROB | DPX | | | | 0> | | | 1<AX | 1> | 2<CZ | 2> | | |
| | DPY | | | | | 0> | 1> | | | | | | |
| I/O | OB | | | | | | | | | | | | |
| MULTIPLIER | FMUL | | | | A*X | B*Y | C*Z | | | | | | |
| | | | | | | A*X | B*Y | C*Z | | | | | |
| | | | | | | | A*X | B*Y | C*Z | | | | |
| | FM | | | | | | | A*X | B*Y | C*Z | | | |
| ADDER | FADD | | | | | | | | AX+BY | | AX+BY+CZ | | |
| | | | | | | | | | | AX+BY | | AX+BY+CZ | |
| | FA | | | | | | | | | | AX+BY | | AX+BY+CZ |
| SPAD | SPFN | IX+1 | IY+1 | IZ+1 | | | | | | | | N-1 | IV+1 |
| | BR | | | | | | | | | | | | N>0:0 |

Bild 5.1.3. Reservierungstafel für die Linearkombination dreier Vektoren auf dem AP-190L.

Bedeutung der Eintragungen in den entsprechenden Zeilen der Reservierungstafel:

MI : Der eingetragene Wert wird während des Zyklus in das MI-Register des Datenspeichers gebracht.

MA : Hierin steht die Zugriffsadresse für den Datenspeicher, unter welcher ein Zugriff gestartet wird.

MD : Bei einem Lesezugriff auf den Datenspeicher erscheint hierin drei Takte nach Beginn des Zugriffs der gelesene Wert.

TMA : Hierin steht die Zugriffsadresse für den Tabellenspeicher, unter welcher ein Zugriff gestartet wird.

TM : Bei einem Lesezugriff auf den Tabellenspeicher erscheint hierin zwei Takte nach Beginn des Zugriffs der gelesene Wert.

DPX : Eine Eintragung I<W bedeuted, daß das DPX-Register I mit dem Wert W geladen wird; eine Eintragung J> bedeuted, daß das DPX-Register J gelesen wird; eine Eintragung J>;I<W bedeuted, daß beide Operationen ausgeführt werden.

DPY : analog zu DPX

DB : Der eingetragene Wert wird über den Datenbus übertragen.

FMUL : Eine Eintragung U•V bedeuted den Start der Multiplikation zweier Werte U und V. Die beiden nachfolgenden Zeilen kennzeichnen die Weitergabe von Zwischenresultaten von einem Pipelinesegment an das jeweils nächste

FM : Der eingetragene Wert erscheint einen Takt nach Weitergabe von Zwischenresultaten an das letzte Pipelinesegment am Ausgang des Multiplizierers.

FADD : Eine Eintragung U+V bedeuted den Start der Addition von zwei Werten U und V. Die nachfolgende Zeile kennzeichnet die Weitergabe von Zwischenresultaten vom ersten an das zweite Pipelinesegment.

FA : Der eingetragene Wert erscheint einen Takt nach Weitergabe von Zwischenresultaten an das zweite Pipelinesegment am Ausgang des Addierers.

SPFN : Der hier eingetragene Wert ist die Ausgabe der arithmetisch-logischen S-Pad-Einheit in diesem Takt.

BR : Eine Eintragung b:I gibt eine Verzweigungsbedingung b und die Verzweigungsadresse I an; eine Eintragung I ist die Adresse bei einer unbedingten Verzweigung. I ist entsprechend der Nummerierung der obersten Zeile angegeben.

## 5.2 ANWENDUNG DES VERFAHRENS

Ausgehend von der Reservierungstafel als Muster für einen Berechnungsprozeß auf dem AP kann das dazugehörige modifizierte Zustandsdiagramm erzeugt werden, aus dem dann eine optimale Scheduling-Strategie zur Überlappung von N Berechnungsprozessen nach diesem Muster ermittelt werden kann. Die Idee dabei ist, den ganzen AP-Rechner als eine statische Pipeline aufzufassen, deren Segmente die einzelnen Elemente des AP sind. Zu beachten ist, daß dabei Pipelinesegmente wie der Addierer oder Multiplizierer selbst wieder als Pipeline organisiert sind.

Da die Reservierungstafel zugleich die AP-Instruktionen für den von ihr beschriebenen Berechnungsprozeß bestimmt, liefert die optimale Scheduling-Strategie zugleich ein für die N-fache Ausführung des Berechnungsprozesses optimales Assemblerprogramm.

### Beispiel 5.2.1.

Im folgenden soll ein optimales Assemblerprogramm zur Ermittlung der Linearkombination dreier Vektoren nach der in Bild 5.1.3. angegebenen Reservierungstafel erzeugt werden.

Ausgehend vom Anfangskollisionsvektior (1,1,1,0,1,0,1,0,1,1,1,1) ergibt sich folgendes modifizierte Zustandsdiagramm, das bereits entsprechend Bemerkung 4 aus Abschnitt 4.1.3. vereinfacht ist:

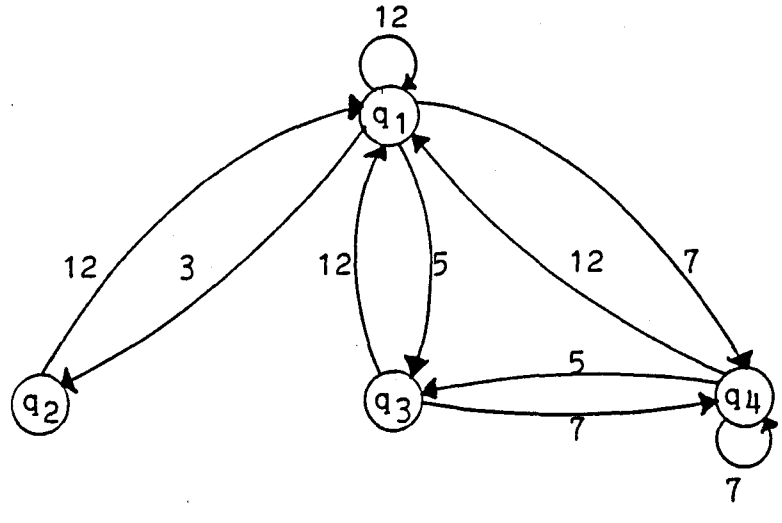



Bild 5.2.1. Modifiziertes Zustandsdiagramm für die Reservierungstafel aus Bild 5.1.3.

Aus dem modifizierten Zustandsdiagramm läßt sich entsprechend dem in Abschnitt 4.2. angegebenen Verfahren folgendes Äquivalenzensystem ÄS(0) = ÄS(1) regulärer Ausdrücke ableiten:

$x_1 \sim 12_1x_1 \vee 32x_2 \vee 53x_3 \vee 74x_4$

$x_2 \sim 12_1x_1$

$x_3 \sim 12_1x_1 \vee 74x_4$

$x_4 \sim 12_1x_1 \vee 53x_3 \vee 74x_4$

Daraus ergibt sich die Menge $Z_1 = \{12_1, 74\}$ und das folgende Äquivalenzensystem ÄS(2):

$x_1 \sim (3_212_1 \vee 5_312_1 \vee 7_412_1)x_1 \vee 7453x_3 \vee 5374x_4$

$x_2 \sim 12_132x_2 \vee 12_153x_3 \vee 12_174x_4$

$x_3 \sim 7412_1x_1 \vee 12_132x_2 \vee (12_153 \vee 7453)x_3 \vee 12_174x_4$

$x_4 \sim 5312_1x_1 \vee 12_132x_2 \vee 12_153x_3 \vee (12_174 \vee 5374)x_4$

Daraus ergibt sich die Menge $Z_2 = \{3_212_1,\ 5_312_1,\ 7_412_1,\ 12_132,\ 12_153\ 7453,\ 12_174,\ 5374\}$ und das folgende Äquivalenzensystem ÄS(3):

$x_1 \sim (7453_312_1 \vee 5374_12_1)x_1$

$x_2 \sim 12_17453x_3 \vee 12_15374x_4$

$x_3 \sim 7412_132x_2 \vee (7412_153 \vee 12_17453)x_3$

$x_4 \sim 5312_132x_2 \vee (5312_174 \vee 12_15374)x_4$

Daraus ergibt sich die Menge $Z_3 = \{7453_312_1,\ 5374_12_1,\ 7412_153,\ 12_17453,\ 5312_174,\ 12_15374\}$ und das Äquivalenzensystem ÄS(4):

$x_1 \sim \Lambda$

$x_2 \sim \Lambda$

$x_3 \sim \Lambda$

$x_4 \sim \Lambda$

Damit bricht das Verfahren ab und es ergeben sich die Mengen

$OZ = \{7453,\ 5374\}$ ,

$F = \{53,\ 74,\ 7453,\ 5374\}$ ,

$OF = \{53\}$ und

$O = \{<5,7,5,7,\ldots>\} = \{(5,7)\}$ .

Die Menge O enthält also nur den optimalen Verzögerungszyklus (5,7) bei einer kleinsten erreichbaren mittleren Verzögerung MAL=6 .

Bemerkung:

Die Greedy-Strategie liefert den Verzögerungszyklus (3,12) und ist damit nicht optimal.

Bei einem Verzögerungszyklus (5,7) ergibt sich mit der Reservierungstafel aus Bild 5.1.3. das in Bild 5.2.2. angegebene Berechnungsmuster zur Linearkombination dreier Vektoren. Die Bezeichnungen sind dabei die gleichen wie die in der Reservierungstafel. Nach einem Aufsetzbereich mit einer Dauer von 5 Zeiteinheiten folgt der stationäre Bereich, der in Bild 5.2.2. mit Hilfe einer Schleife dargestellt wird. Die Dauer eines stationären Intervalls beträgt 12 Zeiteinheiten. Bei jedem Schleifendurchlauf erfolgen zwei Initiierungen bzw. werden zwei Ergebnisse geliefert. Da die Berechnungen bis zum Ende (N=0) nach dem Muster eines stationären Intervalls ablaufen können, erübrigt sich ein Auslaufbereich.
Die Dauer des stationären Bereichs Bereichs beträgt

$12 \cdot N/2$ Zeiteinheiten, falls N gerade ist, und
$12 \cdot (N-1)/2 + 7$ Zeiteinheiten, falls N ungerade ist.

Zu beachten ist, daß zur ersten Berechnung die Adressen IX, IY, IZ und IV die Anfangsadressen der entsprechenden Vektoren und damit die Adressen der jeweils ersten Vektorelemente sind. Erst ab der zweiten Berechnung müssen diese Adressen jeweils inkrementiert werden.

Das APAL-Programm zu dem in Bild 5.2.2. angegebenen Berechnungsmuster ist dem Anhang dieser Arbeit unter dem Namen V3LKA zu entnehmen.

Wie man in Bild 5.2.2. erkennt, ist trotz optimaler Überlappung keine AP-Einheit hundertprozentig ausgelastet. Die höchste Auslastung, nämlich die der S-Pad-Einheit, beträgt 5/6. Entsprechend erreicht die kleinste erreichbare mittlere Verzögerung nicht den für sie kleinstmöglichen Wert, der nach Abschnitt 2.4.4. 5 beträgt.

Eine hundertprozentige Auslastung der S-Pad-Einheit kann nur durch eine Veränderung des zugrunde liegenden Berechnungsmusters, also der Reservierungstafel, erzielt werden. In den anschließenden beiden Beispielen wird für verschiedene Reservierungstafeln zur Linearkombination dreier Vektoren auf dem AP jeweils eine optimale Scheduling-Strategie entwickelt.

| | | 0 | 1 | 2 | 3 | 4 | LOOP | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA MEMORY | MI | | | | | | | | | | | | V |
| | MA | IX | IY | IZ | | | IX+1 | IY+1 | IZ+1 | | | | IV+1 |
| | | | X | Y | Z | | | X | Y | Z | | | |
| | | | | X | Y | Z | | | X | Y | Z | | |
| | MD | | | | X | Y | Z | | | X | Y | Z | |
| TABLE MEMORY | TMA | | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| | TM | | | | | | | | | | | | |
| DATA PROG | DPX | | | | 0> | | | I<AX | I> | 0>:2<CZ | 2> | | I<AX |
| | DPY | | | | | 0> | I> | | | | 0> | | |
| I/O | DB | | | | | | | | | | | | |
| MULTIPLIER | FMUL | | | | A*X | B*Y | C*Z | | | A*X | B*Y | C*Z | |
| | | | | | | A*X | B*Y | C*Z | | | A*X | B*Y | C*Z |
| | | | | | | | A*X | B*Y | C*Z | | | A*X | B*Y |
| | FM | | | | | | | A*X | B*Y | C*Z | | | A*X |
| ADDER | FADD | | | | | | | | AX+BY | | AX+BY+CZ | | |
| | | | | | | | | | | AX+BY | | AX+BY+CZ | |
| | FA | | | | | | | | | | AX+BY | | AX+BY+CZ |
| SPAD | SPFN | IX | IY | IZ | IV-1 | | IX+1 | IY+1 | IZ+1 | | | N-1 | IV+1 |
| | BR | | | | | | | | | | | | N=0:END |

| | | 12 | 13 | 14 | 15 | 16 | END |
|---|---|---|---|---|---|---|---|
| DATA MEMORY | MI | | | | | V | |
| | MA | IX+1 | IY+1 | IZ+1 | | IV+1 | |
| | | | X | Y | Z | | |
| | | | | X | Y | Z | |
| | MD | | | | X | Y | |
| TABLE MEMORY | TMA | | | | | | |
| | | | | | | | |
| | TM | | | | | | |
| DATA PROG | DPX | I> | 2<CZ | 2> | 0> | | |
| | DPY | | | | | 0> | |
| I/O | DB | | | | | | |
| MULTIPLIER | FMUL | | | | A*X | B*Y | |
| | | | | | | A*X | |
| | | C*Z | | | | | |
| | FM | B*Y | C*Z | | | | |
| ADDER | FADD | AX+BY | | AX+BY+CZ | | | |
| | | | AX+BY | | AX+BY+CZ | | |
| | FA | | | AX+BY | | AX+BY+CZ | |
| SPAD | SPFN | IX+1 | IY+1 | IZ+1 | N-1 | IV+1 | |
| | BR | | | | | N>0:LOOP | |

Bild 5.2.2. Berechnungsmuster für die Linearkombination dreier Vektoren nach der Reservierungstafel aus Bild 5.1.3.

Beispiel 5.2.2.

Eine Ursache dafür, daß bestimmte Verzögerungen für die Reservierungstafel aus Bild 5.1.3. zu Kollisionen führen, sind die Operationen auf den Datenregistern. Entsprechend dem Berechnungsmuster der in Bild 5.2.3. angegebenen Reservierungstafel werden zusätzliche durch Datenregister-Zugriffe verursachte Kollisionen zu den von anderen Einheiten verursachten Kollisionen vermieden.

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA MEMORY | MI | | | | | | | | | | | | V |
| | MA | IX+1 | IY+1 | | IZ+1 | | | | | | | | IV+1 |
| | | | X | Y | | Z | | | | | | | |
| | | | | X | Y | | Z | | | | | | |
| | MD | | | | X | Y | | Z | | | | | |
| TABLE MEMORY | TMA | | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| | TM | | | | | | | | | | | | |
| DATA PADS | DPX | | | | 0> | 1> | | 2> | | | | | |
| | DPY | | | | | | | 0<AX | 0> | | | | |
| I/O | OB | | | | | | | | | | | | |
| MULTIPLIER | FMUL | | | | A X | B Y | | C Z | | | | | |
| | | | | | | A X | B Y | | C Z | | | | |
| | | | | | | | A X | B Y | | C Z | | | |
| | FM | | | | | | | A X | B Y | | C Z | | |
| ADDER | FADD | | | | | | | | AX+BY | | AX+BY+CZ | | |
| | | | | | | | | | | AX+BY | | AX+BY+CZ | |
| | FA | | | | | | | | | | AX+BY | | AX+BY+CZ |
| SPAD | SPFN | IX+1 | IY+1 | | IZ+1 | | | | | | | N-1 | IV+1 |
| | BR | | | | | | | | | | | | N>0:0 |

Bild 5.2.3. Reservierungstafel zur Linearkombination dreier Vektoren auf dem AP-190L

Für die Reservierungstafel aus Bild 5.2.3. ergibt sich mit dem Anfangskollisionsvektor (1,1,1,1,0,0,0,1,1,1,1,1) folgendes modifizierte und nach Bemerkung 4 aus Abschnitt 4.1.3. vereinfachte Zustandsdiagramm:

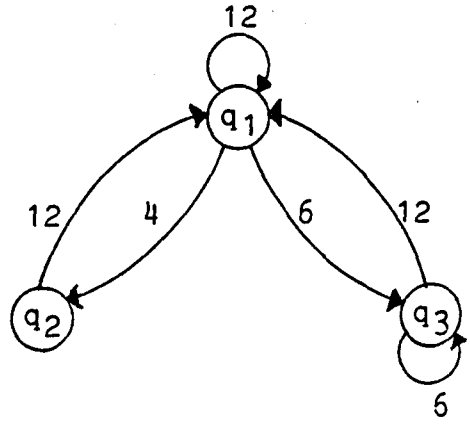



Bild 5.2.4. Modifiziertes Zustandsdiagramm für die Reservierungstafel aus Bild 5.2.3.

Das daraus ableitbare Äquivalenzensystem ÄS(0) = ÄS(1) lautet:

$$x_1 \sim 12_1x_1 \vee 4_2x_2 \vee 6_3x_3$$

$$x_2 \sim 12_1x_1$$

$$x_3 \sim 12_1x_1 \vee \quad 6_3x_3$$

Daraus ergibt sich die Menge $Z_1=\{12_1,6_3\}$ und das folgende Äquivalenzensystem ÄS(2):

$$x_1 \sim (4_212_1 \vee 6_312_1)x_1$$

$$x_2 \sim 12_14_2x_2 \vee 12_16_3x_3$$

$$x_3 \sim 12_14_2x_2 \vee 12_16_3x_3$$

Aus ÄS(2) ergibt sich die Menge $Z_2=\{4_212_1,\ 6_312_1,\ 12_14_2,\ 12_16_3\}$ und das folgende Äquivalenzensystem ÄS(3):

$$x_1 \sim \Lambda$$

$$x_2 \sim \Lambda$$

$$x_3 \sim \Lambda$$

Damit bricht das Verfahren ab und es ergeben sich die Mengen

$$OZ = \{6_3\}$$

$$F = OF = \{6_3\} \quad \text{und}$$

$$O = \{<6,6,6, \ldots >\} = \{(6)\} \ .$$

Die Menge O enthält nur den optimalen Verzögerungszyklus (6); die Greedy-Strategie (4,12) ist auch hier nicht optimal.

In Bild 5.2.5. ist das Berechnungsmuster für den Verzögerungszyklus (6) angegeben. Bei einem Aufsetzbereich von 6 Zeiteinheiten und einer Dauer des stationären Bereichs von 6·N Zeiteinheiten ergibt sich zwar keine Verbesserung gegenüber dem in Bild 5.2.2. angegebenen Berechnungsmuster, jedoch besteht der stationäre Bereich jetzt aus N Intervallen mit einer Dauer von jeweils 6 Zeiteinheiten. Daurch sinkt der Programmieraufwand, wie das dazugehörige Assemblerprogramm, daß dem Anhang unter dem Namen V3LKB zu entnehmen ist, zeigt.

| | | 0 | 1 | 2 | 3 | 4 | 5 | LOOP | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA MEMORY | MI | | | | | | | | | | | | V |
| | MA | IX | IY | | IZ | | | IX+1 | IY+1 | | IZ+1 | | IV+1 |
| | | | X | Y | | Z | | | X | Y | | Z | |
| | | | | X | Y | | Z | | | X | Y | | Z |
| | MD | | | | X | Y | | Z | | | X | Y | |
| TABLE MEMORY | TMA | | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| | TM | | | | | | | | | | | | |
| DATA REG | DPX | | | | 0> | 1> | | 2> | | | 0> | 1> | |
| | DPY | | | | | | | 0<AX | 0> | | | | |
| BUS | DB | | | | | | | | | | | | |
| MULTIPLIER | FMUL | | | | A X | B Y | | C Z | | | A X | B Y | |
| | | | | | | A X | B Y | | C Z | | | A X | B Y |
| | | | | | | | A X | B Y | | C Z | | | A X |
| | FM | | | | | | | A X | B Y | | C Z | | |
| ADDER | FADD | | | | | | | | AX+BY | | AX+BY+CZ | | |
| | | | | | | | | | | AX+BY | | AX+BY+CZ | |
| | FA | | | | | | | | | | AX+BY | | AX+BY+CZ |
| SPAD | SPFN | IX | IY | | IZ | IV-1 | | IX+1 | IY+1 | | IZ+1 | N-1 | IV+1 |
| | BR | | | | | | | | | | | | N>0:LOOP |

Bild 5.2.5. Berechnungsmuster für die Linearkombination dreier Vektoren nach der Reservierungstafel aus Bild 5.2.3.

Beispiel 5.2.3.

Um eine Verbesserung der kleinsten erreichbaren mittleren Verzögerung und damit eine hundertprozentige Segmentauslastung zu erzielen, wird die Reservierungstafel aus Bild 5.2.3. so geändert, daß bei den Verzögerungen 5, 10, 15, ... keine Kollisionen auftreten. Damit ergibt sich, wie im folgenden gezeigt wird, der optimale Verzögerungszyklus (5).

Bild 5.2.7. gibt die Reservierungstafel an, die aus der Reservierungstafel von Bild 5.2.3. durch Verzögerung des Abspeicherns eines Ergebnisses V und der Schleifenzählung hervorgegangen ist.

Mit dem Anfangskollisionsvektor (1,1,1,1,0,0,1,0,1,1,0,1,1,1,1,0) ergibt sich für die Reservierungstafel aus Bild 5.2.7. folgendes modifizierte und nach Bemerkung 4, Abschnitt 4.1.3., vereinfachte Zustandsdiagramm:

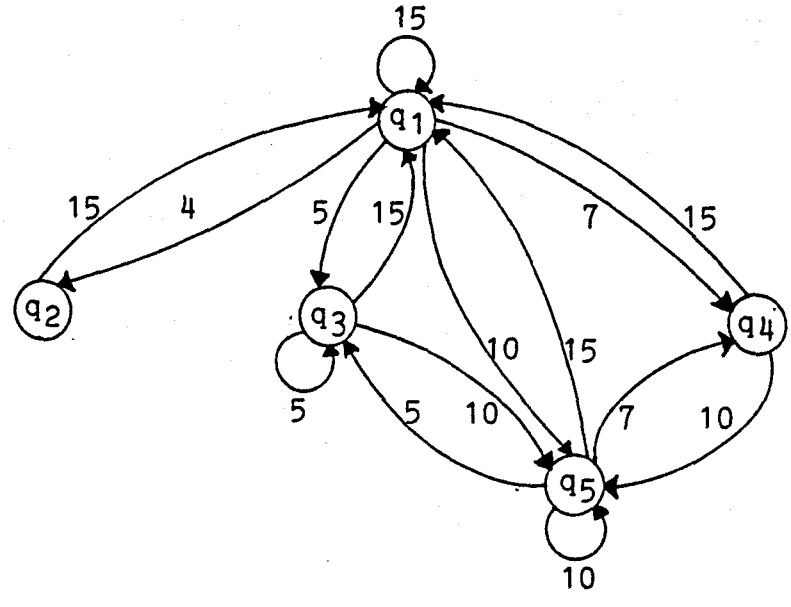



Bild 5.2.6. Modifiziertes Zustandsdiagramm für die Reservierungstafel aus Bild 5.2.7

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA MEMORY | MI | | | | | | | | | | | | |
| | MA | IX+1 | IY+1 | | IZ+1 | | | | | | | | |
| | | | X | Y | | Z | | | | | | | |
| | | | | X | Y | | Z | | | | | | |
| | MD | | | | X | Y | | Z | | | | | |
| TABLE MEMORY | TMA | | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| | TM | | | | | | | | | | | | |
| DATA PADS | DPX | | | | 0> | 1> | | 2> | | | | | 3<V |
| | DPY | | | | | | | 0<AX | 0> | | | | |
| DBB | DB | | | | | | | | | | | | |
| MULTIPLIER | FMUL | | | | A X | B Y | | C Z | | | | | |
| | | | | | | A X | B Y | | C Z | | | | |
| | | | | | | | A X | B Y | | C Z | | | |
| | FM | | | | | | | A X | B Y | | C Z | | |
| ADDER | FADD | | | | | | | | AX+BY | | AX+BY+CZ | | |
| | | | | | | | | | | AX+BY | | AX+BY+CZ | |
| | FA | | | | | | | | | | AX+BY | | AX+BY+CZ |
| SPAD | SPFN | IX+1 | IY+1 | | IZ+1 | | | | | | | | |
| | BR | | | | | | | | | | | | |

| | | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|
| DATA MEMORY | MI | V | | | |
| | MA | IV+1 | | | |
| | | | | | |
| | | | | | |
| | MD | | | | |
| TABLE MEMORY | TMA | | | | |
| | | | | | |
| | TM | | | | |
| DATA PADS | DPX | >3 | | | |
| | DPY | | | | |
| DBB | DB | | | | |
| MULTIPLIER | FMUL | | | | |
| | | | | | |
| | | | | | |
| | FM | | | | |
| ADDER | FADD | | | | |
| | | | | | |
| | FA | | | | |
| SPAD | SPFN | IV+1 | | N-1 | |
| | BR | | | | N>0:0 |

Bild 5.2.7. Reservierungstafel zur Linearkombination dreier Vektoren auf dem AP-190L

Mit dem modifizierten Zustandsdiagramm aus Bild 5.2.6. ergibt sich folgendes Äquivalenzensystem ÄS(0) = ÄS(1) regulärer Ausdrücke:

$x_1 \sim 15_1x_1 \vee 4_2x_2 \vee 5_3x_3 \vee 7_4x_4 \vee 10_5x_5$

$x_2 \sim 15_1x_1$

$x_3 \sim 15_1x_1 \vee 5_3x_3 \vee 10_5x_5$

$x_4 \sim 15_1x_1 \vee 10_5x_5$

$x_5 \sim 15_1x_1 \vee 5_3x_3 \vee 7_4x_4 \vee 10_5x_5$

Daraus ergibt sich die Menge $Z_1 = \{15_1, 5_3, 10_5\}$ und das folgende Äquivalenzensystem ÄS(2):

$x_1 \sim (4_215_1 \vee 5_315_1 \vee 7_415_1 \vee 10_515_1)x_1 \vee 10_553x_3 \vee 10_574x_4$
$\vee (5_310_5 \vee 7_410_5)x_5$

$x_2 \sim 15_14_2x_2 \vee 15_153x_3 \vee 15_174x_4 \vee 15_110_5x_5$

$x_3 \sim 10_515_1x_1 \vee 15_14_2x_2 \vee (15_153 \vee 10_553)x_3$
$\vee (15_174 \vee 10_574)x_4 \vee 15_110_5x_5$

$x_4 \sim 10_515_1x_1 \vee 15_14_2x_2 \vee (15_153 \vee 10_553)x_3$
$\vee (15_174 \vee 10_574)x_4 \vee 15_110_5x_5$

$x_5 \sim (5_315_1 \vee 7_415_1)x_1 \vee 15_14_2x_2 \vee 15_153x_3 \vee 15_174x_4$
$\vee (15_110_5 \vee 5_310_5 \vee 7_410_5)x_5$

Daraus ergibt sich die Menge $Z_2 = \{4_215_1, 5_315_1, 7_415_1, 10_515_1, 15_14_2, 15_153, 10_553, 15_174, 10_574, 5_310_5, 7_410_5, 15_110_5\}$ und das folgende Äquivalenzensystem ÄS(3):

$x_1 \sim (10_5531515_1 \vee 10_5741515_1 \vee 5_3105151 \vee 7_4105151)x_1$
$\vee 7_410_553x_3 \vee 5_310_574x_4$

$x_2 \sim 15_110_553x_3 \vee 15_110_574x_4 \vee (15_153105 \vee 15_174105)x_5$

$x_3 \sim 10_574_15_1x_1 \vee 10_515_14_2x_2 \vee (10_515_153 \vee 15_110_553)x_3$
$\vee (10_515_174 \vee 15_110_574)x_4 \vee 15_174105x_5$

$x_4 \sim 10_5531515_1x_1 \vee 10_515_14_2x_2 \vee (10_515_153 \vee 15_110_553)x_3$
$(10_515_174 \vee 15_110_574)x_4 \vee 15_153105x_5$

$$x_5 \sim (5_3 15_1 4_2 \vee 7_4 15_1 4_2)x_2 \vee 5_3 15_1 7_4 x_3 \vee 7_4 15_1 5_3 x_4$$
$$\vee (5_3 15_1 10_5 \vee 7_4 15_1 10_5 \vee 15_1 5_3 10_5 \vee 15_1 7_4 10_5)x_5$$

Daraus ergibt sich die Menge $Z_3 = \{10_5 5_3 15_1, 10_5 7_4 15_1, 5_3 10_5 15_1, 7_4 10_5 15_1, 10_5 15_1 5_3, 15_1 10_5 5_3, 10_5 15_1 7_4, 15_1 10_5 7_4, 5_3 15_1 10_5, 7_4 15_1 10_5, 15_1 5_3 10_5, 15_1 7_4 10_5\}$ und das Äquivalenzensystem ÄS(4):

$x_1 \sim (7_4 10_5 5_3 15_1 \vee 5_3 10_5 7_4 15_1)x_1$

$x_2 \sim 15_1 7_4 10_5 5_3 x_3 \vee 15_1 5_3 10_5 7_4 x_4$

$x_3 \sim 10_5 7_4 15_1 4_2 x_2 \vee (10_5 7_4 15_1 5_3 \vee 15_1 7_4 10_5 5_3)x_3$

$x_4 \sim 10_5 5_3 15_1 4_2 x_2 \vee (10_5 5_3 15_1 7_4 \vee 15_1 5_3 10_5 7_4)x_4$

$x_5 \sim (5_3 15_1 7_4 10_5 \vee 7_4 15_1 5_3 10_5)x_5$

Daraus ergibt sich die Menge $Z_4 = \{7_4 10_5 5_3 15_1, 5_3 10_5 7_4 15_1, 10_5 7_4 15_1 5_3, 15_1 7_4 10_5 5_3, 10_5 5_3 15_1 7_4, 15_1 5_3 10_5 7_4, 5_3 15_1 7_4 10_5, 7_4 15_1 5_3 10_5\}$ und das Äquivalenzensystem ÄS(4):

$x_1 \sim \Lambda$

$x_2 \sim \Lambda$

$x_3 \sim \Lambda$

$x_4 \sim \Lambda$

$x_5 \sim \Lambda$

Damit bricht das Verfahren ab und es ergeben sich die Mengen

$OZ = \{5_3\}$ ,

$F = \{5_3, 10_5 5_3, 7_4 10_5 5_3\}$ ,

$OF = \{5_3\}$ und

$O = \{<5,5,5, \ldots >\} = \{(5)\}$ .

Mit der in Bild 5.2.7. angegebenen Reservierungstafel ergibt sich damit als optimale Scheduling-Strategie der Verzögerungszyklus (5), womit die kleinste erreichbare mittlere Verzögerung ihre untere Grenze, nämlich 5, erreicht; die S-Pad Einheit ist hundertprozentig ausgelastet. Die Greedy-Strategie (4,15) ist auch hier nicht optimal. Bild 5.2.8. gibt das Berechnungsmuster für den Verzögerungszyklus (5) an. Die Aufsetzzeit beträgt 11 Zeiteinheiten, die Dauer des stationären Bereichs 5·N Zeiteinheiten bei N stationären Intervallen. Das entsprechende APAL-Programm ist im Anhang unter dem Namen V3LKC aufgeführt.

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA MEMORY | MI | | | | | | | | | | | |
| | MA | IX | IY | | IZ | | IX+1 | IY+1 | | IZ+1 | | IX+1 |
| | | | X | Y | | Z | | X | Y | | Z | |
| | | | | X | Y | | Z | | X | Y | | Z |
| | MD | | | | X | Y | | Z | | X | Y | |
| TABLE MEMORY | TMA | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | TM | | | | | | | | | | | |
| DATA PATH | DPX | | | | 0> | 1> | | 2> | | 0> | 1> | |
| | DPY | | | | | | | 0<AX | 0> | | | |
| | OB | | | | | | | | | | | |
| MULTIPLIER | FMUL | | | | A X | B Y | | C Z | | A X | B Y | |
| | | | | | | A X | B Y | | C Z | | A X | B Y |
| | | | | | | | A X | B Y | | C Z | | A X |
| | FM | . | | | | | | A X | B Y | | C Z | |
| ADDER | FADD | | | | | | | | AX+BY | | AX+BY+CZ | |
| | | | | | | | | | | AX+BY | | AX+BY+CZ |
| | FA | | | | | | | | | | AX+BY | |
| SPAD | SPFN | IX | IY | | IZ | IV-1 | IX+1 | IY+1 | | IZ+1 | | IX+1 |
| | BR | | | | | | | | | | | |

| | | LOOP | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|
| DATA MEMORY | MI | | V | | | |
| | MA | IY+1 | IV+1 | IZ+1 | | IX+1 |
| | | X | Y | | Z | |
| | | | X | Y | | Z |
| | MD | Z | | X | Y | |
| TABLE MEMORY | TMA | | | | | |
| | | | | | | |
| | TM | | | | | |
| DATA PATH | DPX | 2>;3<V | 3> | 0> | 1> | |
| | DPY | 0<AX | 0> | | | |
| | OB | | | | | |
| MULTIPLIER | FMUL | C Z | | A X | B Y | |
| | | | C Z | | A X | B Y |
| | | B Y | | C Z | | A X |
| | FM | A X | B Y | | C Z | |
| ADDER | FADD | | AX+BY | | AX+BY+CZ | |
| | | | | AX+BY | | AX+BY+CZ |
| | FA | AX+BY+CZ | | | AX+BY | |
| SPAD | SPFN | IY+1 | IV+1 | IZ+1 | N-1 | IX+1 |
| | BR | | | | | N>0;LOOP |

Bild 5.2.8. Berechnungsmuster zur Linearkombination dreier Vektoren auf dem AP nach der Reservierungstafel aus Bild 5.2.7.

Bemerkung:

Bei der Anwendung des Verfahrens auf den AP-190L ist folgendes zu beachten:

Es muß sichergestellt werden, daß zwischen dem Beschreiben und anschließenden Lesen eines Registers oder Speicherplatzes in einer Berechnung keine Änderungen dieses Registers oder Speicherplatzes durch eine nachfolgende Berechnung bei überlappter Verarbeitung erfolgt. Dies kann zum Beispiel dadurch gewährleistet werden, daß in der Reservierungstafel zusätzliche Markierungen eingetragen werden, die jeweils anzeigen, daß ein Register oder ein Speicherplatz in der jeweiligen Zeiteinheit nicht beschrieben werden darf.

## 6.0 SCHLUSSBETRACHTUNG

Zum Start von Berechnungsprozessen in einer statischen oder dynamischen Pipeline ist eine Kontrolle erforderlich, die verhindert, daß es zu Kollisionen mit zuvor gestarteten Berechnungsprozessen in der Pipeline kommt. Aufgabe einer Scheduling-Strategie für Pipelines ist die Auswahl geeigneter Initiierungszeitpunkte. Eine mögliche Scheduling-Strategie ist die sogenannte Greedy-Strategie, die sich jedoch nicht immer als optimal erweist.

Als optimale Scheduling-Strategie für eine statische Pipeline wird eine Strategie bezeichnet, mit der die höchstmögliche Initiierungsrate beziehungsweise die kleinste erreichbare mittlere Verzögerung erzielt wird.

In dieser Arbeit wurde ein Verfahren zur Entwicklung einer optimalen Scheduling-Strategie vorgestellt. Ausgangspunkt für das Verfahren ist jeweils ein fest vorgegebenes Berechnungsmuster, das durch eine Reservierungstafel repräsentiert wird. Für die Reservierungstafel wird mit Hilfe des Verfahrens eine optimale Überlappung erzielt. Wie die Anwendungsbeispiele in Abschnitt 5 dieser Arbeit zeigen, nimmt die kleinste erreichbare mittlere Verzögerung trotz optimaler Überlappung der Reservierungstafel nicht unbedingt den Wert ihrer unteren Grenze an, was die hundestertprozentige Auslastung mindestens eines Pipelinesegments bedeuted. Erst durch Änderung der Reservierungstafel kann die kleinste erreichbare mittlere Verzögerung, falls überhaupt, ihre untere Grenze erreichen. Die Änderung einer Reservierungstafel kann unter anderem durch das Einfügen von Verzögerungsschritten (vgl. Beispiel 5.2.3.) erfolgen, falls dies möglich ist. Es bleibt zu untersuchen, ob sich ein Verfahren konstruieren läßt, das eine optimale Reservierungstafel in dem Sinne generiert, daß mit dieser Reservierungstafel eine hundertprozentige Auslastung mindestens eines Pipelinesegments erreicht werden kann.

Bei der Anwendung des Verfahrens zur Entwicklung einer optimalen Scheduling-Strategie sind außerdem folgende Punkte zu beachten:

1. Bestehen zwischen zwei verschiedenen Berechnungen Abhängigkeiten in den Daten, zum Beispiel durch Rekursionen, muß sichergestellt werden, daß die Daten, die für eine Berechnung benötigt werden, auch rechtzeitig verfügbar sind. Bei Datenabhängigkeiten läßt sich im allgemeinen keine oder nur eine schlechte Überlappung der Berechnungen erzielen. Einzige Möglichkeit, das Problem schneller zu lösen, bleibt in solchen Fällen eine Umstellung des Algorithmus.

2. Bei komplexeren Anwendungen mit längeren Reservierungstafeln, aus denen im allgemeinen eine hohe Anzahl von Zuständen des modifizierten Zustandsdiagramms resultiert, kann das Verfahren sehr aufwendig werden, jedoch kann sich im Fall von Berechnungen, die sehr oft oder auf langen Vektoren durchzuführen sind, ein höherer Aufwand durchaus lohnen.

## 7.0 LITERATURVERZEICHNIS

/Ch 71/
Chen T. C. 1971: "Parallelism, Pipelining and Computer Efficiency", Computer Design, January, 69-74

/Ch 75/
Chen T. C. 1975: "Overlap and Pipeline Processing", in Stone H. S. (ed.): Introduction to Computer Architecture, Science Research Associates, 375-431

/Da 71/
Davidson E. S. 1971: "The Design and Control of Pipelined Function Generators", Proc. 1971 Int. IEEE Conf. on Systems, Networks, and Computers, Oaxtepec, Mexico, January, 19-21

/Da 74/
Davidson E. S. 1974: "Scheduling for Pipelined Processors", Proc. 7th Hawaii Conf. on System Sciences, 58-60

/Da,Th,Sh,Pa 75/
Davidson E. S., Thomas A. T., Shar L. E., Patel J. H. 1975: "Effective Control for Pipelined Computers", Proc. Spring COMPCON, IEEE No. 75CH 0920-9C, 181-184

/FPS 78/
"AP-190L Reference Manuals Vol. I-III", Floating Point Systems, INC., 1978

/Ho 80/
Hoßfeld F. 1980: "Parallelverarbeitung - Konzepte und Perspektiven", Angewandte Informatik, 12/80

/In,Kl 81/
Indermark K., Klaeren H. 1981: "Automatentheorie und Formale Sprachen I", Schriften zur Informatik und Angewandten Mathematik Nr.71 der RWTH Aachen

/Ko 81/
Kogge P. M. 1981: "The Architecture of Pipelined Computers", Mc Graw-Hill

/Kr,Ne,Un 75/
Krayl H., Neuhold E. J., Unger C. 1975: "Grundlagen der Betriebssysteme", Sammlung Göschen, de Gruyter

/Ku 78/
Kuck D. J. 1978: "The Structure of Computers and Computations", John Wiley & Sons

/Pa,Da 76/
Patel J. H., Davidson E. S. 1976: "Improving the Throughput of a Pipeline by Insertion of Delays", IEEE/ACM 3rd Ann. Symp. Computer Arch., IEEE No. 76CH 0143-5C, 159-163

/Ra,Li 77/
Ramamoorthy, C. V., Li H. F. 1977: "Pipeline Architecture", Computing Surveys, vol. 9, No. 1, March, 61-102

/Sh 72/
Shar L. E. 1972: "Design and Scheduling of Statically Configured Pipelines", Digital Systems, Lab. Report SU-SEL-72-042, Stanford University, Stanford, California, September

/Th,Da 74/
Thomas A. T., Davidson E. S. 1974: "Scheduling of Multiconfigurable Pipelines", Proc 12th Ann. Allerton Conf. Circuits and System Theory, University of Illinois, Champaign-Urbana, 658-669

/Wi 82/
Winkens H.-D. 1982: "Pipeline-Rechner: Architektur und Programmierung", Berichte der KFA Jülich - Nr.164

# ANHANG

Bedeutung der in den folgenden Programmen verwendeten APAL-Befehle:

MOV I,J Der Inhalt des S-Pad-Registers I wird in diesem Zyklus zum Ausgang (SPFN) der S-Pad-Einheit und in das S-Pad-Register J gebracht.

INC I Der Inhalt des S-Pad-Registers I wird in diesem Zyklus inkrementiert und zum Ausgang (SPFN) der S-Pad-Einheit gebracht.

DEC I Der Inhalt des S-Pad-Registers I wird in diesem Zyklus dekrementiert und zum Ausgang (SPFN) der S-Pad-Einheit gebracht.

SETMA Laden des MA-Registers mit der Ausgabe der S-Pad-Einheit (SPFN) und Starten eines Zugriffs auf den MD-Speicher.

MI<E Laden des MI-Registers mit E. E kann unter anderem die Ausgabe des Addierers (FA) oder des Multiplizierers (FM) sein.

DPX(I)<E Das DPX-Register I relativ zur Adresse DPA wird mit E geladen. E kann unter anderem die Ausgabe des Addierers (FA), des Multiplizierers (FM) oder des MD-Speichers (MD) sein.

DPY(I)<E analog zu DPX(I)<E

FMUL M1,M2 Start der Multiplikation von M1 und M2 und Weitergabe von Zwischenresultaten vorher begonnener Multiplikationen in der Pipeline (Weiterschieben der Pipeline). M1 kann unter anderem ein Registerinhalt (DPX, DPY) oder eine Ausgabe des Multiplizierers (FM), M2 ein Registerinhalt (DPX, DPY), eine Ausgabe des MD-Speichers oder eine Ausgabe des Addierers (FA) sein.

FMUL Weiterschieben der Multiplikationspipeline

FADD A1,A2 Start der Addition von A1 und A2 und Weiterschieben der Additionspipeline. A1 kann unter anderem ein Registerinhalt (DPX, DPY) oder eine Ausgabe des Multiplizierers (FM), A2 ein Registerinhalt (DPX, DPY), eine Ausgabe des MD-Speichers oder eine Ausgabe des Addierers (FA) sein.

FADD Weiterschieben der Additionspipeline

BGT ADR Verzweigung zur symbolischen Adresse ADR, falls im vorhergehenden Zyklus ein Wert größer als Null am Ausgang (SPFN) der S-Pad-Einheit erschien.

BEQ ADR Verzweigung zur symbolischen Adresse ADR, falls im vorhergehenden Zyklus ein Wert gleich Null am Ausgang (SPFN) der S-Pad-Einheit erschien.

```
" APAL-UNTERPROGRAMM ZUR BERECHNUNG DER LINEARKOMBINATION
" VON 3 VEKTOREN DER LAENGE N NACH DER FORMEL
"    V(I) = A*X(I) + B*Y(I) + C*Z(I)    FUER I=1(1)N
"
" AUSFUEHRUNGSZEIT DER RESERVIERUNGSTAFEL: 12 ZEITEINHEITEN
" VERZOEGERUNGSZYKLUS (5,7)
" AUFSETZZEIT: 5 ZEITEINHEITEN
" DAUER DES STATIONAEREN BEREICHS:
"           12*N/2 ZEITEINHEITEN          FUER N GERADE
"           (12*(N-1)/2)+7 ZEITEINHEITEN  FUER N UNGERADE
"
"
        $TITLE V3LKA
        $ENTRY V3LKA,10
"
"
" FESTLEGUNG VON SYMBOLEN ZUR BEZEICHNUNG VON S-PAD-REGISTERN
"
        IA $EQU 0
        IX $EQU 1
        IB $EQU 2
        IY $EQU 3
        IC $EQU 4
        IZ $EQU 5
        IV $EQU 6
         N $EQU 7
"
"
" FESTLEGUNG VON SYMBOLEN ZUR BEZEICHNUNG VON DATA-PAD-REGISTERN
"
" DPX-REGISTER:
"
         A $EQU 0
        AX $EQU 1
        CZ $EQU 2
"
" DPY-REGISTER:
"
         B $EQU 0
         C $EQU 1
"
"
"REGISTER LADEN MIT KONSTANTEN
"
V3LKA: MOV IA,IA; SETMA
       MOV IB,IB; SETMA
       MOV IC,IC; SETMA
"
"
" AUFSETZBEREICH DER PIPELINE
"
       MOV IX,IX; SETMA; DPX(A)<MD
       MOV IY,IY; SETMA; DPY(B)<MD
       MOV IZ,IZ; SETMA; DPY(C)<MD
"
       DEC IV; FMUL DPX(A),MD
       FMUL DPY(B),MD
"
"
```

```
600 " STATIONAERER BEREICH
610 "
620  LOOP: INC IX; SETMA; FMUL DPY(C),MD
630        INC IY; SETMA; DPX(AX)<FM; FMUL
640        INC IZ; SETMA; FMUL; FADD FM,DPX(AX)
650        DPX(CZ)<FM; FMUL DPX(A),MD; FADD
660        FMUL DPY(B),MD; FADD DPX(CZ),FA
670        DEC N; FMUL DPY(C),MD; FADD
680        INC IV; SETMA; MI<FA; DPX(AX)<FM; FMUL; BEQ END
690 "
700        INC IX; SETMA; FMUL; FADD FM,DPX(AX)
710        INC IY; SETMA; DPX(CZ)<FM; FADD
720        INC IZ; SETMA; FADD DPX(CZ),FA
730        DEC N; FMUL DPX(A),MD; FADD
740        INC IV; SETMA; MI<FA; FMUL DPY(B),MD; BGT LOOP
750 "
760 "
770 " PROGRAMMENDE MIT RUECKSPRUNG INS HAUPTPROGRAMM
780 "
790  END:  RETURN
800        $END
```

```
" APAL-UNTERPROGRAMM ZUR BERECHNUNG DER LINEARKOMBINATION
" VON 3 VEKTOREN DER LAENGE N NACH DER FORMEL
"    V(I) = A*X(I) + B*Y(I) + C*Z(I)  FUER I=1(1)N
"
" AUSFUEHRUNGSZEIT DER RESERVIERUNGSTAFEL: 12 ZEITEINHEITEN
" VERZOEGERUNGSZYKLUS (6)
" AUFSETZZEIT: 6 ZEITEINHEITEN
" DAUER DES STATIONAEREN BEREICHS: 6*N ZEITEINHEITEN
"
"
        $TITLE V3LKB
        $ENTRY V3LKB,10
"
"
" FESTLEGUNG VON SYMBOLEN ZUR BEZEICHNUNG VON S-PAD-REGISTERN
"
        IA $EQU 0
        IX $EQU 1
        IB $EQU 2
        IY $EQU 3
        IC $EQU 4
        IZ $EQU 5
        IV $EQU 6
         N $EQU 7
"
"
" FESTLEGUNG VON SYMBOLEN ZUR BEZEICHNUNG VON DATA-PAD-REGISTERN
"
" DPX-REGISTER:
"
         A $EQU 0
         B $EQU 1
         C $EQU 2
"
" DPY-REGISTER:
"
        AX $EQU 0
"
"
" REGISTER LADEN MIT KONSTANTEN
"
V3LKB: MOV IA,IA; SETMA
        MOV IB,IB; SETMA
        MOV IC,IC; SETMA
"
"
" AUFSETZBEREICH
"
        MOV IX,IX; SETMA; DPX(A)<MD
        MOV IY,IY; SETMA; DPX(B)<MD
        DPX(C)<MD
"
        MOV IZ,IZ; SETMA; FMUL DPX(A),MD
        DEC IV; FMUL DPX(B),MD
        FMUL
"
"
```

```
580 " STATIONAERER BEREICH DER PIPELINE
590 "
600  LOOP: INC IX; SETMA; DPY(AX)<FM; FMUL DPX(C),MD
610        INC IY; SETMA; FMUL; FADD FM,DPY(AX)
620        FMUL; FADD
630        INC IZ; SETMA; FMUL DPX(A),MD; FADD FM,FA
640        DEC N; FMUL DPX(B),MD; FADD
650        INC IV; SETMA; MI<FA; FMUL; BGT LOOP
660 "
670 "
680 " PROGRAMMENDE MIT RUECKSPRUNG INS HAUPTPROGRAMM
690 "
700        RETURN
710        $END
```

```
" APAL-UNTERPROGRAMM ZUR BERECHNUNG DER LINEARKOMBINATION
" VON 3 VEKTOREN DER LAENGE N NACH DER FORMEL
" V(I) = A*X(I) + B*Y(I) + C*Z(I)  FUER I=1(1)N
"
" AUSFUEHRUNGSZEIT DER RESERVIERUNGSTAFEL: 16 ZEITEINHEITEN
" VERZOEGERUNGSZYKLUS (5)
" AUFSETZZEIT: 11 ZEITEINHEITEN
" DAUER DES STATIONAEREN BEREICHS: 5*N ZEITEINHEITEN
"
"
        $TITLE V3LKC
        $ENTRY V3LKC,10
"
"
" FESTLEGUNG VON SYMBOLEN ZUR BEZEICHNUNG VON S-PAD-REGISTERN
"
        IA $EQU 0
        IX $EQU 1
        IB $EQU 2
        IY $EQU 3
        IC $EQU 4
        IZ $EQU 5
        IV $EQU 6
         N $EQU 7
"
"
" FESTLEGUNG VON SYMBOLEN ZUR BEZEICHNUNG VON DATA-PAD-REGISTERN
"
" DPX-REGISTER:
"
         A $EQU 0
         B $EQU 1
         C $EQU 2
         V $EQU 3
"
" DPY-REGISTER:
"
        AX $EQU 0
"
" REGISTER MIT KONSTANTEN LADEN
"
V3LKC: MOV IA,IA; SETMA
       MOV IB,IB; SETMA
       MOV IC,IC; SETMA
"
"
" AUFSETZBEREICH
"
       MOV IX,IX; SETMA; DPX(A)<MD
       MOV IY,IY; SETMA; DPX(B)<MD
       DPX(C)<MD
       MOV IZ,IZ; SETMA; FMUL DPX(A),MD
       DEC IV; FMUL DPX(B),MD
"
       INC IX; SETMA; FMUL
       INC IY; SETMA; DPY(AX)<FM; FMUL DPX(C),MD
       FMUL; FADD FM,DPY(AX)
       INC IZ; SETMA; FMUL DPX(A),MD; FADD
       FMUL DPX(B),MD; FADD FM,FA
"
       INC IX; SETMA; FMUL; FADD
"
```

```
640 " STATIONARER BEREICH DER PIPELINE
650 "
660  LOOP: INC IY; SETMA; DPX(V)<FA; DPY(AX)<FM; FMUL DPX(C),MD
670        INC IV; SETMA; MI<FA; FMUL; FADD FM,DPY(AX)
680        INC IZ; SETMA; FMUL DPX(A),MD; FADD
690        DEC N; FMUL DPX(B),MD; FADD FM,FA
700        INC IX; SETMA; FMUL; FADD; BGT LOOP
710 "
720 "
730 " PROGRAMMENDE MIT RUECKSPRUNG INS HAUPTPROGRAMM
740 "
750        RETURN
760        $END
```

Die vorliegende Arbeit wurde als Diplomarbeit am Institut für allgemeine Elektrotechnik und Datenverarbeitungssysteme der RWTH Aachen bei Herrn Prof. Dr.-Ing. O. Lange im Zentralinstitut für Angewandte Mathematik der KFA Jülich angefertigt. Für die Unterstützung durch das Zentralinstitut für Angewandte Mathematik bedanke ich mich beim Direktor des Instituts, Herrn Dr. F. Hoßfeld.