シストリックアルゴリズム開発支援システムに関する研究

東北大学大学院工学研究科 情報工学専攻
白石 勉

目 次

図 目 次

表 目 次

第１章

はじめに

1.1　本研究の背景と目的

世界初のディジタルコンピュータENIACの登場から45年、コンピュータの世界は、その必要性とエレクトロニクス技術の進歩により、めざましい発展をとげた。かつては、部屋いっぱいの空間を占有した計算機も、現在では、数ミリ角のチップ上に載るようになり、また、その処理速度は飛躍的に向上し、1年もかかるような計算がわずか数分、数秒でできるようになった。しかし、信号処理、画像処理、コンピュータグラフィック、物理現象のシミュレーションなどの研究分野では、それでもまだ満足できず、さらなる高速処理のできる計算機を必要としている。

この期待に応えるため、計算機アーキテクチャの分野では、並列処理アーキテクチャを考え出し、それに関する研究が盛んにおこなわれている。従来の計算機が、1つの演算装置により、逐次的に処理をおこなうのに対し、並列処理では、1つの大きな問題を細分化し、それらを並列に処理する。これにより、高速に処理を進めることが可能となるのである。

この並列処理アーキテクチャの1つに、シストリックアレーがある。シストリックアレーは、1978年、KungおよびLeisersonにより提案されたアレープロセッサの一種である。シストリック(systolic)とは、”心臓収縮の”という意味の形容詞であり、心臓が収縮動作を繰り返しながら、血液を全身に送り出し、受け取る様子を個々のプロセッサとデータストリームの動作に見たて、そのようなプロセッサから構成されるアレープロセッサのことをシストリックアレーというのである。シストリックアレーは、単純な処理を行うプロセッサ(セル)が、規則的に配置され、局所的に結合されている。このため、VLSI化が容易であるというメリットをもっている。最近では、シストリックコンピュータの開発や、

シストリックチップを用いたハードウェアへの応用など、その実用性は高く評価され、並列処理の中でも、一つの研究分野として確立されてきている。

しかし、シストリックアレーを利用するには、設計が難しい、動作解析が難しいなどの問題がある。これらの問題に対して、自動設計という研究分野が生まれ、多くの研究者により研究が進められている。

しかしながら、自動設計とはいえ、人手で行うことは大変な労力を必要とする。

そこで、本研究は、提案されている自動設計法の1つを基礎に計算機上で実現できる手法を検討し、得られたシストリックアレーの動作を解析できるシストリックアレー開発支援システムを構築することを目的とする。

1.2 本論文の構成

本論文の構成は以下の通りである。

第1章 本研究の背景と目的について述べる。

第2章 並列処理機構の一つであるシストリックアレーの特徴とその実現上の問題点について述べる。また、シストリックアレーを数学的に定式化する。

第3章 本研究で基礎となっているシストリックアレーの自動設計法について述べる。

第4章 第3章の設計法において、それを機械的に実行する際の問題点について述べる。

第5章 シストリックアルゴリズム開発支援システムについて述べる。

第6章 第5章のシステムを、具体例をとって、その動作を検証する。

第7章 本研究のまとめを述べる。

第 2 章

シストリックアレー

シストリックアレーは、高速フーリエ変換 (FFT)、行列計算などの科学技術計算、画像処理、各種記号処理などのためのソフトウェア (サブルーチンパッケージ) に代わる、特定用途向けハードウェア付加装置として考案された。通常、ノイマン型計算機のシステム・バスに直接接続され、その周辺機器の一つとして使用される。シストリックアレーは数千から数万個のシンプルなプロセッシング・エレメント (セルと呼ばれる) を規則的に接続したものからなる。

2.1 シストリックアレーの特徴と実現上の問題点

シストリックアレーの特徴として次の点が挙げられる。

構造・演算の一様性 各セルは規則的に配置されていて、同じ機能を持っている。これによりハードウエア化する際に設計を容易にし、高集積化を可能にする。

通信の局所性 各セルは隣接するセルとのみ局所的に接続される。したがって、互いに離れたセル間でデータの授受を行うには、そのセル間の距離に等しいステップを必要とする。このように、アレー上でのデータの送受信は局所結合を通じて局所的に行われる。これによりハードウエア化する際の配線の問題を軽減することができる。

これらの点は、計算能力に優れた VLSI の長所を最大限に引き出し、限定された通信という VLSI の短所を回避する[6]。しだがって、シストリックアレーは VLSI 化するのに適しているアレープロセッサであると言える。

シストリックアレーは問題となるアルゴリズムの規則性、再帰性、局所性などの特徴を利用した問題指向 (アルゴリズム指向) のアレープロセッサである。逆に言うと、その

ような特徴を持つアルゴリズムはシストリックアレーで解くことができるということである。これまで様々な応用が考えられてきたが、その一部を示すと次のようになる。

- 信号処理、イメージ処理(コンボリューション、相関フィルタリングなど)
- 行列、ベクトル演算(行列積、QR 分解など)
- 非数値応用(ソーティング、文字認識など)

しかし、シストリックアレーは実現上の問題点として、以下のような問題点が指摘されている。

設計の困難さ シストリックアレーは、これまで、個々の問題に対して、個別にヒューリスティックに設計されてきた。このとき、各計算をどのステップにどのセルで行わせるのか、各セル間のデータの授受をどのように行うのか、アレーへのデータの入出力はどのようにするのか、などの問題が生じる。しかもそれは、シストリックアレーの規則性、局所性という制約を満たしていなければならない。動作の確認が困難であることもあって、このようなシストリックアレーのの設計を人手で行うことは非常に難しい。そのためアルゴリズムから自動的にシストリックアレーを設計する、自動設計手法に関する研究が行われ、幾つかの自動設計法が考案されている。しかし、これらは特定の問題にしか適応できなかったり、人間のヒューリスティックな知識を利用しなけれはならないなどの問題がある。

動作の確認、評価の困難さ シストリックアレーは各セルが並列に動作し、アレー内に多量のデータが流れているため、その動作を確認することが非常に難しい。このことが設計を上で述べた以上に難しくしている。また、そのため、計算時間の評価を行うことが難しい。

そこで、これらの問題に対処するため、一般的な問題に対して、与えたアルゴリズムから自動的にシストリックアルゴリズムを設計する自動設計法が必要とされ、これまで、多くの研究が行われてきている。

2.2 シストリックアレーの定式化とループプログラム

シストリックアレイは次のように定式化することができる[3]。

定義1 シストリックアレイとは次のセル機能F、セル空間G, セル間結線W, 配置領域Rの4項組$< F, G, W, R >$である。

(1)$F =< P_{in}, P_{out}, D, f >$ P_{in}, P_{oput}は入力/出力ポート集合で、$|P_{in}| = n, |P_{out}| = m$とおく。$D$はデータの集合で、
$f =< f_1, f_2, ..., f_a, ..., f_m >$は$f_a : [P_{in} \rightarrow D] \rightarrow D$なる写像の組である。$[P_{in} \rightarrow D] = D^n$は入力ポート上のデータの集合を表し、その元$d$を$< d(i)|i \in P_{in} >$とも表す。$f_a$はその入力データに対して出力ポート$a$の値を決める関数で、セル関数と呼ぶ。

(2)Gはある可換群$G =< G; +; 0 >$で、セル機能を配置する空間の座標の集合である。

(3)wはポート間結線写像$u : P_{in} \rightarrow P_{out}$と近傍写像$N : P_{in} \rightarrow G$との2項組$< u, N >$である。各セル$\alpha$の入力ポート$i$をセル$\alpha - N(i)$の出力ポート$u(i)$に接続することを意味する。

(4)$R \subseteq G$は、実際にセル機能をおくセル座標の集合であり、アレイ領域とも呼ぶ。

この定式化の例を図2.1に示す。

この定式化によりシストリックアレイの動作を次のループプログラムで記述することができる。ここで連結用入力ポートをPA_{cin}とおき、入出力様相を次のものとする。

$y^t \in [P_{in} \times G \rightarrow D]$

$x^t \in [P_{out} \times G \rightarrow D]$

$d^t_{cin} \in [PA_{cin} \rightarrow D]$

for i=1 to M

for $\alpha \in$ R

for a $\in P_{out}$

x^t(a,α)=$f_a(< x^{t-1}(u(i), \alpha - N(i))|i \in P_{in} >)$

但し、$\alpha - N(i) \in R$ のとき

$x^{t-1}(u(i), \alpha - N(i)) = d^{t-1}_{cin}(u(i), \alpha - N(i))$

$P_{in}=\{ 1, 2, 3 \}$
$P_{out}=\{ a, b, c \}$
D: the set of numbers
$f_a(d_1, d_2, d_3)=d_1$
$f_b(d_1, d_2, d_3)=d_2$
$f_c(d_1, d_2, d_3)=d_1 * d_2 + d_3$
$G=Z^2$

u(1)=a	N(1)=(-1, 0)
u(2)=b	N(2)=(0, -1)
u(3)=c	N(3)=(1, 1)

R=(1, 2, 3) x (1, 2, 3)

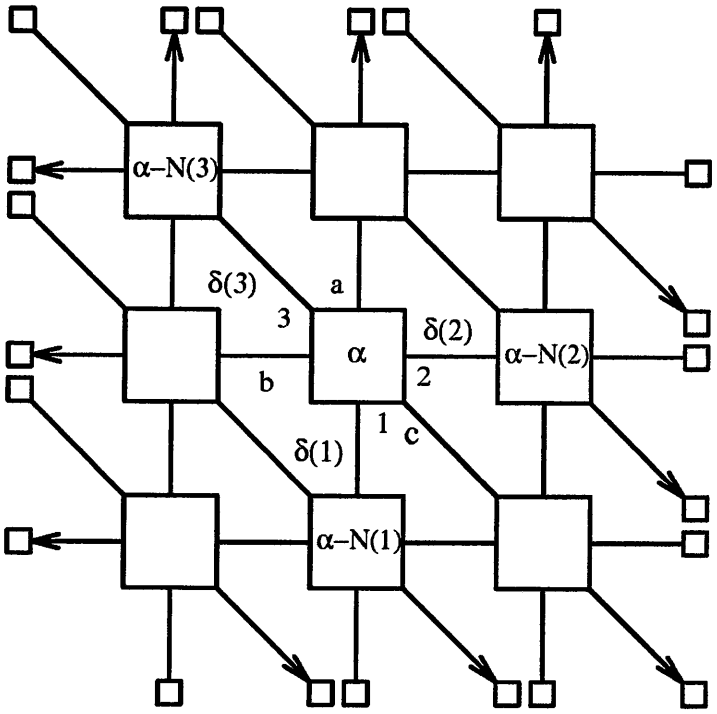

図2.1 シストリックアレイの記述例

第 3 章

シストリックアレー自動設計法

本研究で基礎となっている自動設計法は文献[3] に基づいたものであり、この章では、その設計法に関して説明する。

3.1 ループプログラム

解きたい問題の仕様は次のようなループプログラム (基本ループプログラムと呼ぶことにする) で与えるものとする。

for j_1=b_1 to e_1

...

for j_n=b_n to e_n

for p=1 to M

$a_p[k_p(j_1,...,j_n)] = f_p(< a_p^q[k_p^q(j_1,...,j_n)]|q \in Occur >)$

ここで

p:割当文の番号、その集合を $ID = \{1,...,M\}$ とおく

a_p:p 番目の割当文で値が更新される配列名

$k_p(j_1,...,j_n)$: a_pの要素を指定する添え字式

f_p : a_pの更新の計算式

a_p^q : a_pの q 番目の入力となる配列名

$k_p^q(j_1,...,j_n)$: a_p^qの添え字式

$Occur$:入力の出現番号の集合

以下の議論のため、次を定める。

ループインデックスの集合を

Index=$\{(j_1,...,j_n)|b_1 \leq j_1 \leq e_1,...,b_n \leq j_n \leq e_n\}$

とおき、その元をjと記す。

Arg=ID×Occur

配列名の集合を Sort とおき、s:ID→Sort,arg:Arg→Sort をそれぞれ次のように定める。

$s(p) = a_p, arg(p,q) = a_p^q$

3.2 データ依存関係

前節のループプログラムにおいて、ループ本体の各割当文で参照されている配列変数がどのループインデックスで計算されているのかについての対応を求めるため、次の関数を導入する。

定義2 次の関数 g:Index×Arg→Z^n×(ID∪{0}) を生成時点関数という。g(**j**,< p,q >) は arg(p,q)=s(p')、k_p^q(**j**)=$k_{p'}$(**j'**)、(**j**,p)>(**j'**,p') を満たす(**j'**,p') が存在すれば、そのうちで最大の順位にあるものを値とし、存在しなければ、(**j**,0) を値とする。特に、値の各々を示すため次の記法を用いる。

j'=**j**◁< p,q >、st(**j**,< p,q >)=p'

st を文指定関数、◁を逆行関数と呼ぶ。

ここで、次を定める。

IndexI=$\{$**j**◁< p,q > $|$**j**∈Index,< p,q >∈Arg$\}$,

IndexA=Index∪IndexI

ループプログラムをシストリックアレーで実行させるとき、各ループインデックス**j**における計算はある時刻にあるセルで実行される。その時刻とセル位置を与える写像をそれぞれ、計算時刻関数Tc、計算セル位置関数Ac と呼び、これを求めることがシストリックアレーの設計となる。

Tc:IndexA → T

Ac:IndexA → G

セル機能の一様性、すなわち、どの$\mathbf{j}$に対する計算についても結線および関数機能が同一であるという要請を課すると、ループプログラムは次の条件を満たさなければならない。各$i \in$Argに対して、

(C0) st($\mathbf{j}$,i)=$u(i)$ は$\mathbf{j}$に依存しない。

更に、Ac,Tcは次の条件を満たさなければならない。

(C1) Ac($\mathbf{j}$)-Ac($\mathbf{j} \triangleleft i$)=$N(i)$ は$\mathbf{j}$に依存しない。

(C2) Tc($\mathbf{j}$)-Tc($\mathbf{j} \triangleleft i$)=$\delta(i)$ は$\mathbf{j}$に依存しない。

Nは近傍写像に対応し、δを時間調整写像という。計算の依存性、および、入出力ポート間のデータ転送には1単位時間を必要とするので、次の条件が得られる。

(C3) $\delta(i) \leq 0$ 特に、$\mathbf{j} \neq \mathbf{j} \triangleleft i$ ならば、$\delta(i) > 0$

次に、セル機能を単純にするために、同時刻、同一セルでは、高々1つのインデックスにおける計算しか実行されないことを要請する(最簡機能条件という)。

(C4) Ac($\mathbf{j}$)=Ac($\mathbf{j}'$) ならば Tc($\mathbf{j}$)$\neq$Tc($\mathbf{j}'$)

以上の条件(C1)～(C4)を満たすTc,Ac,N,δが存在すれば、それによりシストリックアレーが定まる。

条件(C1)、(C2)における$N(i)$、$\delta(i)$の定め方から次が成立する。

命題1

(1) 各$i, i' \in$Argについて、ある$\mathbf{j}$に対して、$\mathbf{j} \triangleleft i = \mathbf{j} \triangleleft i'$ ならば、

$$N(i) = N(i'), \delta(i) = \delta(i')$$

(2)$\mathbf{j} \triangleleft i = \mathbf{j}$ ならば、$N(i) = 0, \delta(i) = 0$

そこで、Argを上の(1)に従って、同値類分割し、各同値類をeと表し、リンクと呼ぶ。そして、その集合Eをリンク集合と呼ぶ。

3.3 自動設計法の概要

次にこの自動設計法の概要を示す。

1. 入力変数としてだけ使われている参照変数に関して、流れ化変換、分流変換をする。

2. 各割当文の参照変数に対して生成時点関数を求める次の操作を行う。

(a) 同一の添え字式をもつものをまとめて、リンク候補を作る。

(b) 各リンク候補に対して、逆行関数を求める。

(c) 文指定関数を求め、条件C0を満たすかどうかをチェックする。

3. 逆行関数に基づき、同値類に分割し、リンク集合Eを求める。

4. 条件(C1)と(C2)の方程式から、N, δに関する方程式を作る。

5. 方程式および$\delta > 0$を満たす、N, δを選定する。

6. ループ本体の計算式とδからセル機能を定め、Nと$u(i)$=st($\mathbf{j}$,i)からセル間結線を定める。

第 4 章

自動設計法実現における問題点の解析

4.1 自動設計法実現における問題点

第3章で、本研究で基礎となっているシストリックアレー自動設計法について述べた。これを計算機上で実現するのが本研究の目的である。

しかし、条件 (C1) と (C2) の N,Ac,δ,Tc に関する方程式を計算機上で解き、構成可能条件を導出するのは、非常に困難なことである。そこで、以下の節に示すような工夫を考えた。

4.2 構成可能条件導出法

4.2.1 添え字式のクラス

変数の添え字式のクラスとして

- ループインデックス
- 各々のループインデックスの非線型関数 (例えば、$\lfloor i/2 \rfloor$)

の線型結合を考える。

従来の設計法における添え字式のクラスは、ループインデックスのみの線型結合であったが、これでは、添え字式にループインデックスの非線型関数があるときには、設計が不可能となる。そこで、それを加えたものの線型結合を添え字式のクラスとした。

4.2.2 計算セル位置関数・計算時刻関数

条件(C1)と(C2)は、N、Ac、δ、Tcを未知関数とする方程式とみなすことができる。この方程式を機械的に解くため、Ac,Tcが次のような形をもつものを考える。

すなわち、添え字に使われているループインデックスとその非線型関数を要素にもつベクトルを$\mathbf{J(j)}$として、Ac,Tcは$\mathbf{J(j)}$の線型結合であらわされるものとする。構成されるシストリックアレーの次元数をn、$\mathbf{J(j)}$の次元数をmとすると、

計算セル位置関数 Ac(j) C_Aを$n \times m$行列として、

$$\mathrm{Ac}(\mathbf{j})=C_A \cdot \mathbf{J(j)}$$

計算時刻関数 Tc(j) C_Tを$1 \times m$ベクトルとして、

$$\mathrm{Tc}(\mathbf{j})=C_T \cdot \mathbf{J(j)}$$

4.2.3 構成可能条件

条件(C1),(C2)および前節の2式から係数C_A, C_Tと$N(a), \delta(a)$の関係が得られる。この関係が、すべての$\mathbf{j}$について成立することが、シストリックアレー構成可能条件となる。この係数C_A, C_Tと$N(a), \delta(a)$の関係をあらわす方程式を$C_A, C_T, N(a), \delta(a)$に関するパラメータ方程式と呼ぶことにする。

$$C_A \cdot (\mathbf{J(j)}\text{-}\mathbf{J(j}\triangleleft i\mathbf{)})=N(i)$$

$$C_T \cdot (\mathbf{J(j)}\text{-}\mathbf{J(j}\triangleleft i\mathbf{)})=\delta(i)$$

第5章

シストリックアルゴリズム開発支援システム

第3章で提案した自動設計法に基づき、シストリックアルゴリズム開発支援システムSDS(Surport system for the Design of Systolic algorithm) を構築した。SDS は、入力したループプログラムからシストリックアルゴリズム構成可能条件を導出する部分と、得られた条件を満足するパラメータを選定し入力することで、シストリックアルゴリズムをグラフィカルにシミュレートできるグラフィックシミュレータから構成される (図5.2)。

5.1 入力仕様と支援システムの概要

本システムの入力仕様として、2重または3重ループプログラムを考える。また、その中は、ｉｆ文と割当文とからなるものとする (図5.5)。図5.2に本システムの概要を示す。

5.2 構成可能条件導出部

この部分では、構文解析ツールyacc を用いて入力ループプログラムを以下の手順で変換し、構成可能条件を導出する。

以下に変換法とそのアルゴリズムを示す。

5.2.1 基本ループプログラムへの変換

(1) 処理の内部化

ループの前後にある文をループ本体の中にとりいれる。

一般形：

< Initialize >

for j=b to e

{ < Body > }

< Ending >

変換後：

for j=b-1 to e+1

{ if j=b-1 then < Initialize >

else if j=e+1 then < Ending >

else < Body > }

(2) i f 文の除去

ループ中にある if 文を if-then-else 関数を用いて割当文に変換する。

一般形：

if < exp > then { a_p=f_p();... }

else { a_p=h_p();... }

変換後：

a_p=if < exp > then $f_p()$ else $h_p()$;

(3) 添え字の統一

制御変数に関する条件がある場合、その意味から等価になる添え字式があれば統一する。

典型例：

$a[j]$ =if $j = 0$ then $b[i]$+ ...

else $b[i-j]$+ ...

変換後：

$a[j]$ =if $j = 0$ then $b[i-j]$+ ...

else $b[i-j]$+ ...

5.2.2 流れ化変換・分流変換

(4) 流れ化変換

割当文の左辺に現れていない配列名の参照変数 $b[k(j)]$ について

割当文：$b[k(j)] = b[k(j)]$

を本体の最初に付加する。これはシストリックアレイにおいてデータがパイプライン的に流れることに対応している。

(5) 分流変換

割当文の左辺に現れていない参照変数で、配列名が同一で添え字式が異なりかつそれらの添え字式が互いに排反でないような変数がある場合、それらは別々のデータ流となるので、それらを区別するため新しい変数を用意して割当文を生成し本体に付加する。

5.3 構成可能条件の導出

(6) リンク候補集合の導出

まず、シストリックアレーにおいて、どの変数がデータとして流れているのかを導出しなければならない。そこで、そのような変数である可能性のあるものをリンク候補として、次の手順によりそれを導出する。

以下のようにして参照変数の出現をまとめて得られる同値類をリンク候補とし、まとめられなかった出現もまた一元集合としてリンク候補とする。

(a) 同じ配列名を持ち、添え字式が等しい参照変数で、その生成割当文より共に前にあるものをまとめて、一つの同値類とする。さらに、共に後にあるものもまとめて、もう一つの同値類とする。

(b) 添え字式が同一の参照変数で、各々の前後にあるその変数の生成割当文の左辺の添え字式が同種のものからなっているものをまとめて、一つの同値類とする。

(7)N, δ, C_A, C_Tに関するパラメータ方程式の導出

第4章で述べたN, δ, C_A, C_Tに関するパラメータ方程式を次のようにして導出する。

1. 基本ループプログラムのループに値を与えて、データ流iの添え字式がある値(1つの値を与えてやる)になるときの$\mathbf{J(j)}$の値を求め、記憶する。

2. ループインデックスの値が進んでiの添え字式がまたその値になるときの$\mathbf{J(j)}$の値を求め、記憶しておいた値との差$\mathbf{J(j)\text{-}J(j}\triangleleft i\mathbf{)}$を計算する。

3. こうして、N, δ, C_A, C_Tに関するパラメータ方程式を形成する。形成された式がそれまでに出力したものと異なるものであればそれを出力する。

4. リンク集合のすべてについて1〜3を繰り返す。

(8)N, δの候補のリストアップ

(7) で得られたN, δ, C_A, C_Tに関するパラメータ方程式からNおよびδのみの方程式を構成する。このとき、Ac,Tc はNおよびδを係数とする関数であらわすことができる。

隣接するセルとのみ結合するという制限を設け、Nは-1,0,+1であるとする。また、$\delta > 0$であることよりN,δの値がリストアップされる。

5.4 グラフィックシミュレータ

5.4.1 データ流速度とデータ流レイアウト関数

構成可能条件部で得られた条件を満足するパラメータをもつ計算セル位置関数と計算時刻関数および近傍写像と時間調整写像から、データ流iの速度とデータ流iのある時刻 t における位置をあらわすレイアウト関数が導出される。

データ流速度

$$v(i) = N(i)/\delta(i)$$

データ流レイアウト関数

$$Lo(\mathbf{j},i)=\mathrm{Ac}(\mathbf{j})-(\mathrm{Tc}(\mathbf{j})-t)*v(i)$$

例えば、時刻0におけるデータ流iのレイアウトは、時刻Tc(j)における計算がAc(j)のセルで行われることから、Ac(j)と、時刻0から時刻Tc(j)までにデータ流iが進んだ大きさTc(j)*$v(i)$との差をとることにより得ることができる。

5.4.2 グラフィックシミュレータの機能

グラフィックシミュレータの機能として以下のようなものがある。

- 時刻の前進・後退によるデータ流の動作解析 (forward,backward ボタン)
- アレーにデータが入り、出るまでの様子を表示 (animate ボタン)
- 各セル内で行われている処理の表示 (セル上のクリック)
- アレーサイズ、データ入出力時刻、アクティブなセル数の表示

5.4.3 グラフィックシミュレータの実現

構成可能条件部で得られた条件を満足するパラメータ選定し、図5.5のようなヘッダーファイルを作成する。このヘッダーファイルをインクルードしてコンパイルするとグラフィックシミュレータが作成される(図5.7,5.8)。

5.5 支援システムの利用法

この支援システムでは、まず、対象問題を記述したループプログラムからシストリックアレーの構成可能条件を導出する。次に、この条件を満足するようにパラメータ(N,δ)を選定する。選定されたパラメータの各組に対して、グラフィックシミュレータにより、シストリックアレーを構成し、その動作を解析する。解析した結果を比較することにより、どのパラメータの組のシストリックアレーが、意図に最適なものであるか知ることができる。

```
program : init for ending
init    : /* blank */
        | exprs
ending  : /* blank */
        | exprs
for     : FOR alph '=' term1 TO term2 for_body
        | FOR  alph '=' term1 DOWNTO term2 for_body
for_body: '{' exprs '}'
        | expr
        | '{' init for ending '}'
        | for
exprs   : expr
        | exprs expr
expr    : alloc
        | if
alloc   : Arrs '=' Opers ';'
        | Arrs '=' Opers
Arrs    : alph '[' index1 ']'
        | alph '[' index1 ',' index2 ']'
arrs    : alph '[' index1 ']'
        | alph '[' index1 ',' index2 ']'
opers   : oper
        | opers biop oper
Opers   : opers
        | o_if
o_if    : IF conds THEN opers Opr_end
Opr_end : /* blank */
        | ELSE opers
        | ELSE o_if
oper    : arrs
        | index
if      : IF conds THEN if_body
if_body : '{' if_allcs '}'
        | '{' if_allcs '}' ELSE el_alloc
        | '{' if_allcs '}' ELSE '{' el_allcs '}'
        | if_alloc
        | if_alloc ELSE el_alloc
        | if_alloc ELSE '{' el_allcs '}'
if_allcs: if_alloc
        | if_allcs if_alloc
el_allcs: el_alloc
        | el_allcs el_alloc
if_alloc: Arrs '=' Opers ';'
el_alloc: Arrs '=' Opers ';'
```

図5.1　入力仕様の一部(yaccによる記述)

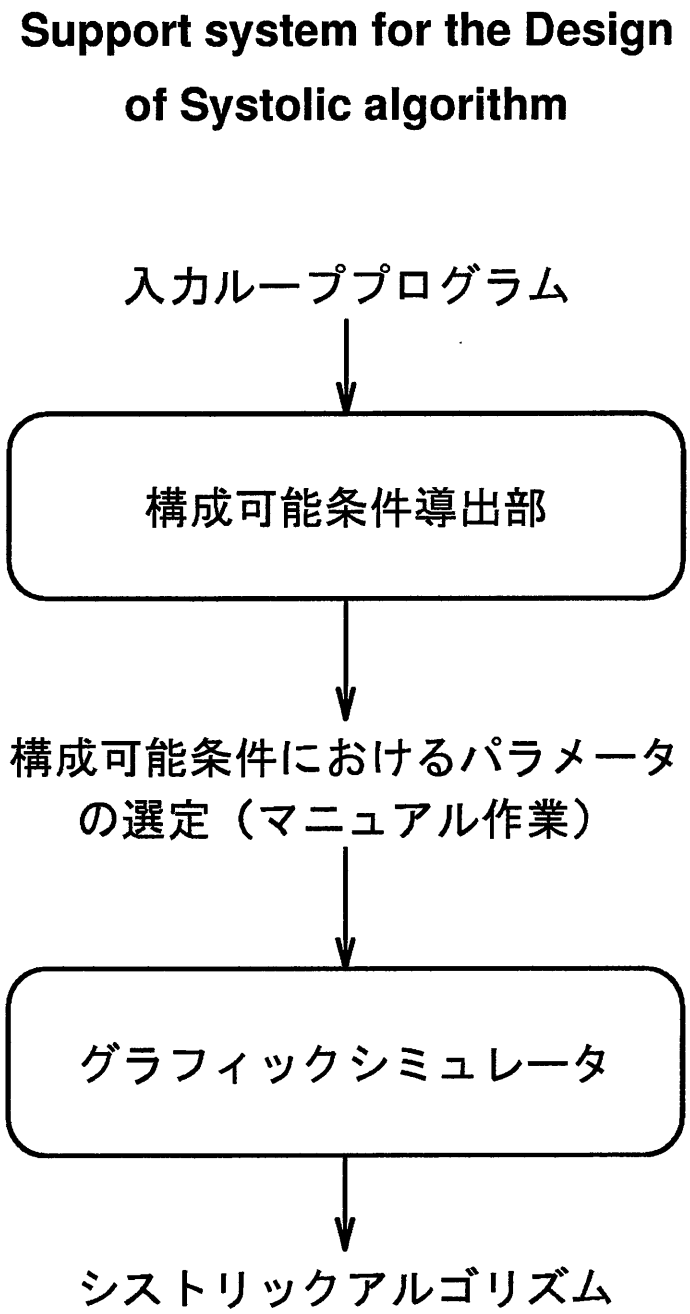

図5.2 シストリックアルゴリズム開発支援システムの構成

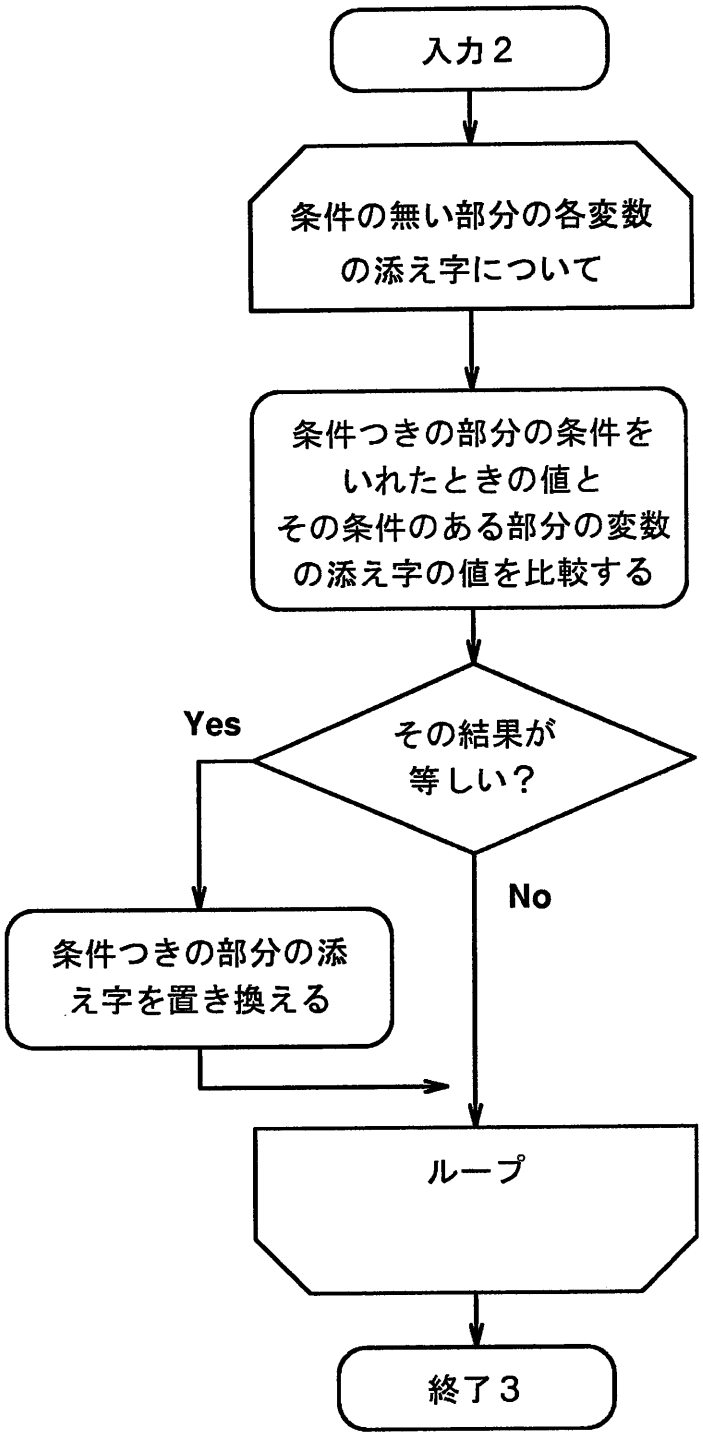

図5.3 添え字統一のフローチャート

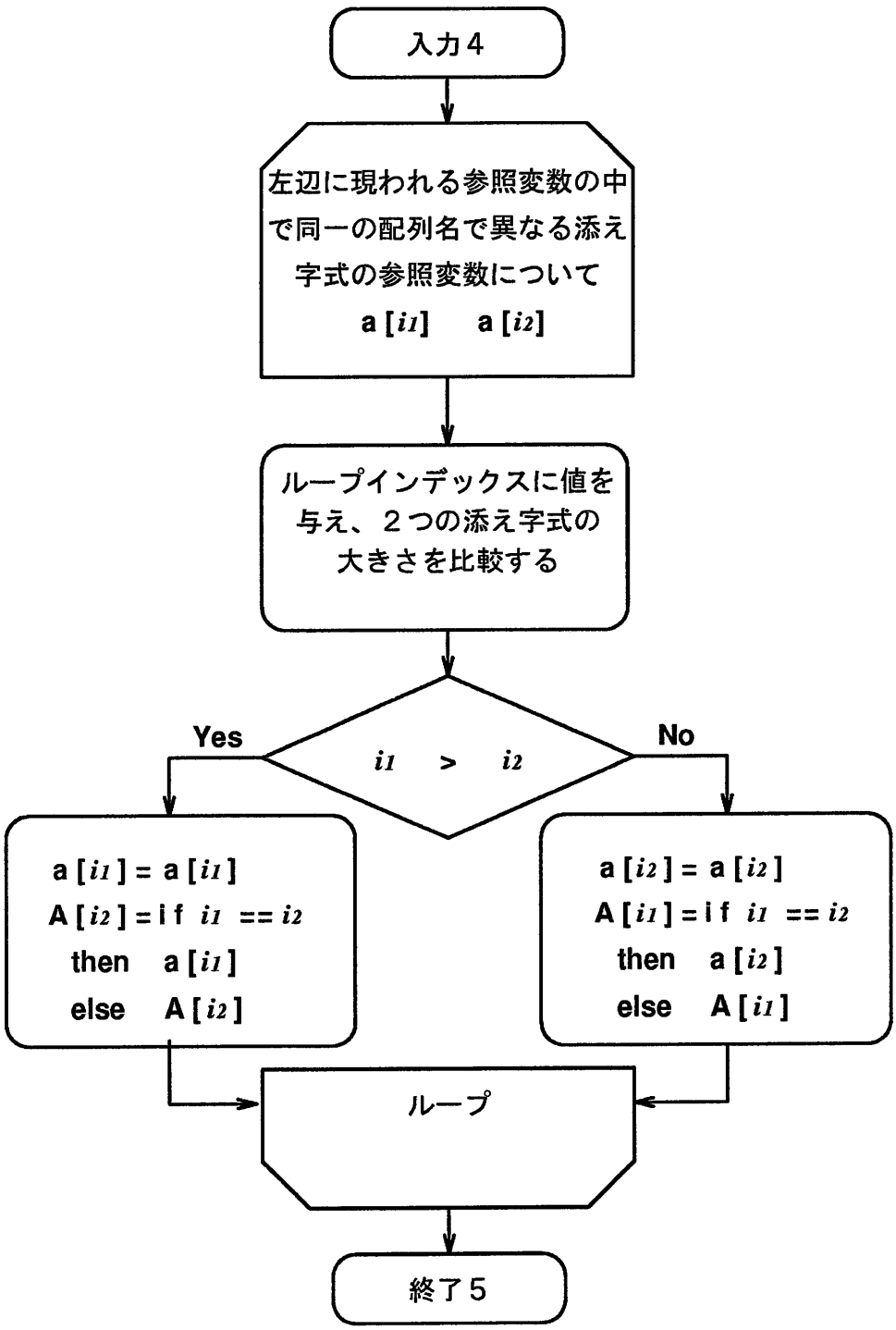

図5.4 分流変換のフローチャート

```
#define da              (1)
#define db              (1)
#define dc              (2)
#define Tc              ((2*i)-(i/2)+(j))
#define NaX             (1)
#define NaY             (0)
#define NbX             (0)
#define NbY             (0)
#define NcX             (1)
#define NcY             (0)
#define AcX             (i-i/2+j)
#define AcY             (0)
#define IDaX            (i)
#define IDaY            (0)
#define IDbX            (i-i/2+j)
#define IDbY            (0)
#define IDcX            (i/2-j)
#define IDcY            (0)
#define ISTART          (0)
#define JSTART          (0)
#define IEND            (SIZE+K)
#define JEND            (i/2)
#define DIMS            (2)
#define Va              "c"
#define Vb              "a"
#define Vc              "A"
#define Fa              "A%d*B%d-->c%d",IDcX,IDcX,IDaX
#define Fa2             "a%d*B%d+A%d*b%d-->c%d",IDbX,IDcX,IDcX,IDbX,IDaX
#define Fa3             "c%d+a%d*B%d+A%d*b%d-->c%d",IDaX,IDbX,IDcX,IDcX,IDbX,IDaX
#define Fb              "a%d-->a%d;b%d-->b%d",IDbX,IDbX,IDbX,IDbX
#define Fc              "a%d-->A%d;b%d-->B%d",IDbX,IDcX,IDbX,IDcX
#define Fc2             "A%d-->A%d;B%d-->B%d",IDcX,IDcX,IDcX,IDcX
```

図5.5　アレー情報ヘッダーファイルの例

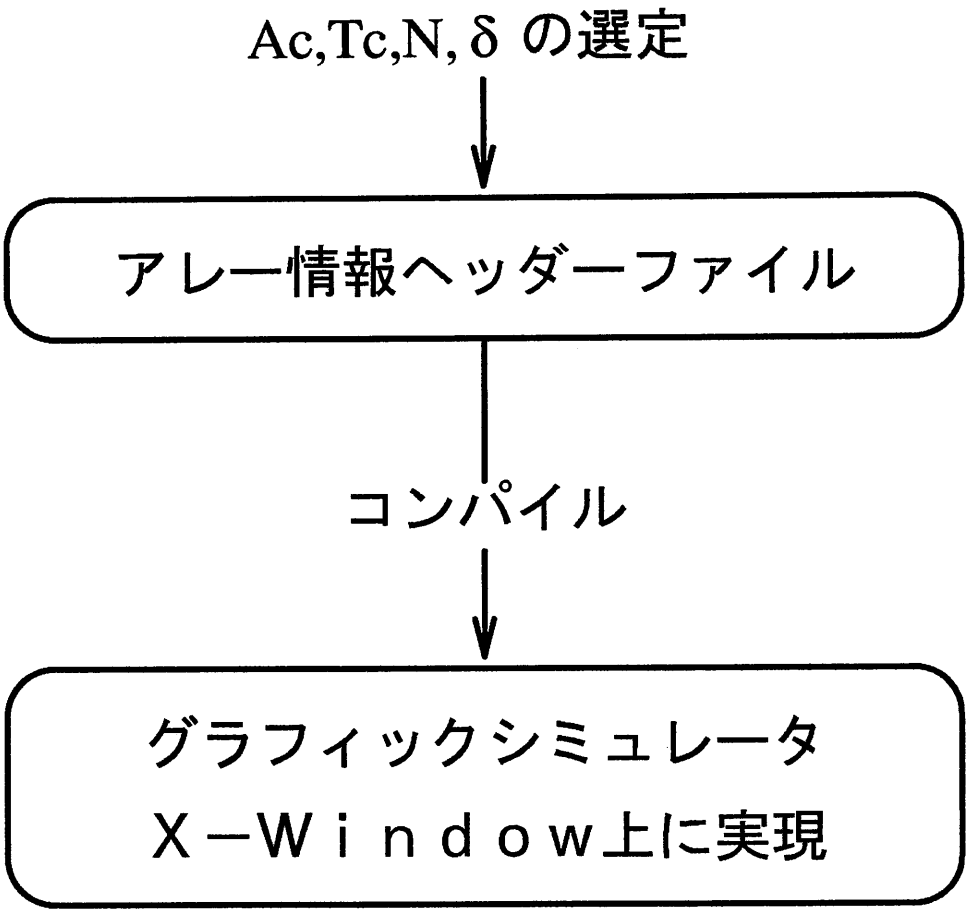

図5.6 グラフィックシミュレータ作成のフローチャート

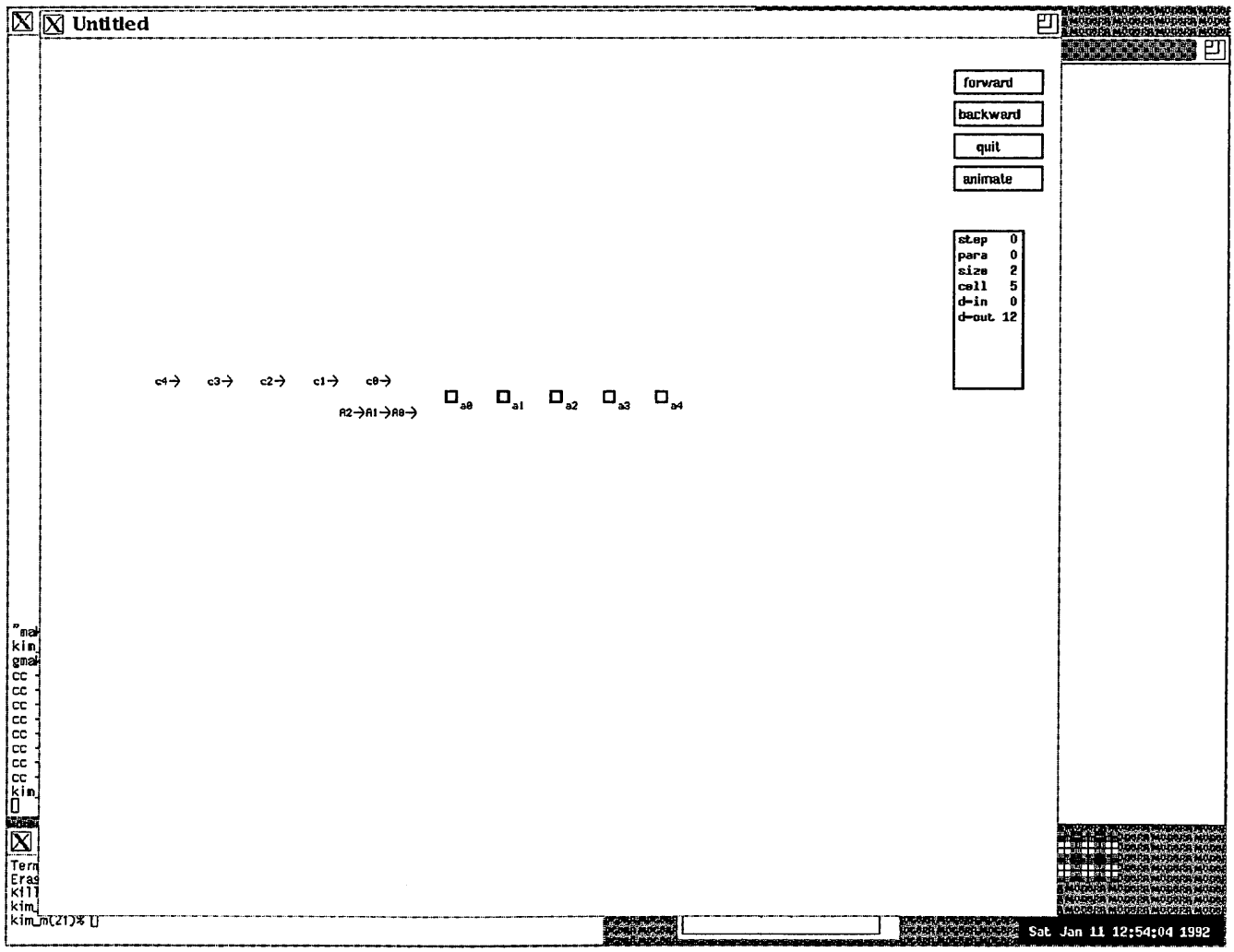

図5.7 グラフィックシミュレータの概観

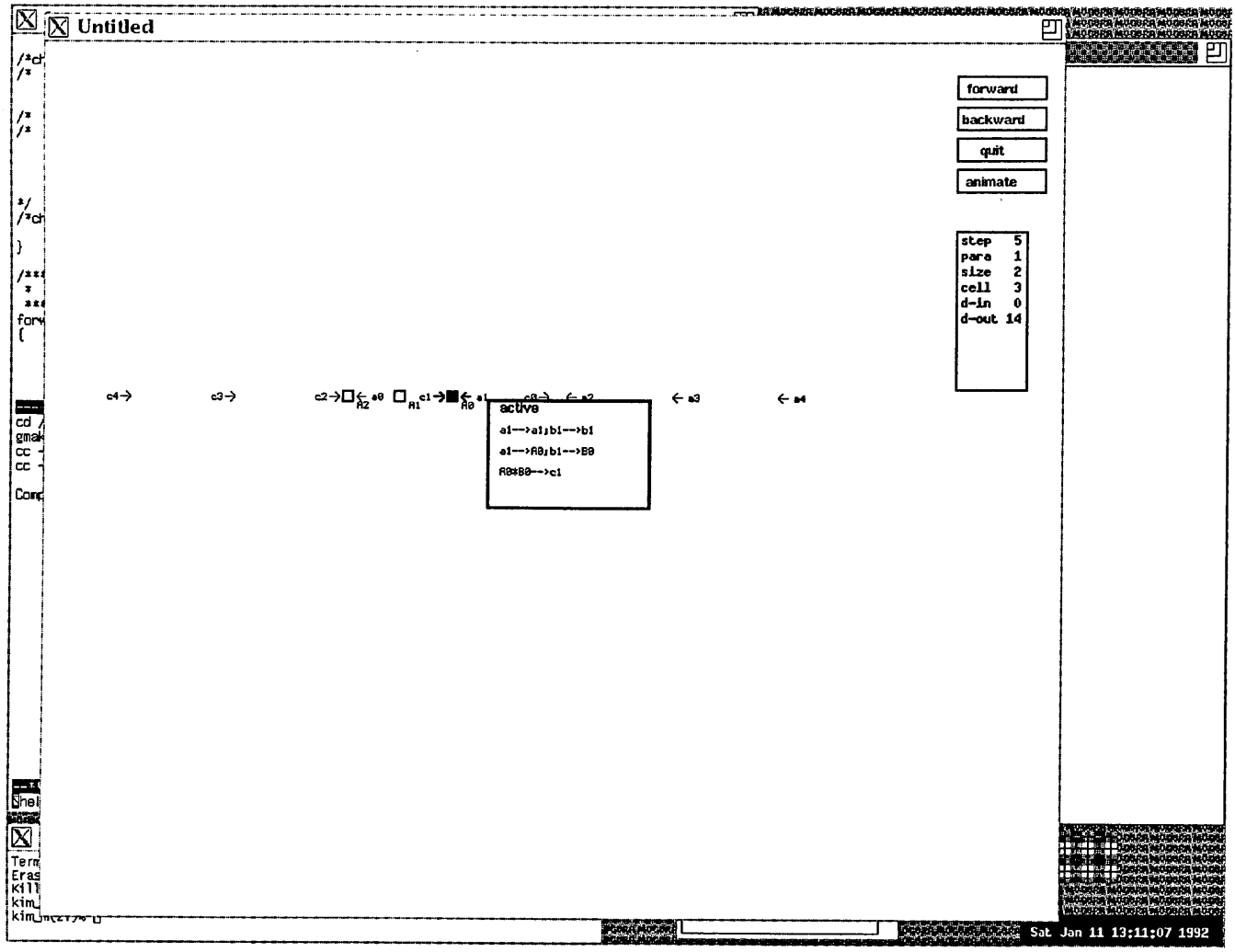

図5.8 グラフィックシミュレータの概観2

第 6 章

支援システムの性能評価

6.1 コンボリューションアルゴリズムに対する実行例

コンボリューションアルゴリズムに対する本システムの実行例を後にに示す。

6.2 支援システムに関する考察

この支援システムは、対象問題をループプログラムの形で与えることにより、機械的にシストリックアレーの構成可能条件を導出できる。したがって、このシステムを利用することにより、これまでのような設計にかかる労力は、大幅に軽減されると思われる。また、構成可能なシストリックアレーをグラフィックシミュレータにより構成できるので、その動作を実際に確かめることが可能となる。このシミュレータは、X-Window 上でマウスにより操作できるので、操作しやすく、シストリックアレーの動作解析も容易である。また、構成したシストリックアレーを比較することにより、意図に最適なものを得ることができる。

```
for i = 1 to N + M {
  if  i %2 == 0  then
      c [ i ] = a [ i/ 2 ] * b [i/ 2 ] ;
  else
      c [ i ] = a [ i- i / 2 ] * b [i / 2 ]
              + a [i/ 2 ] * b [ i- i / 2 ] ;
  for j = 1 to  i / 2
    c [ i ] = c [ i] + a [ i- i/ 2 + j ] * b [ i / 2 - j ]
                     + a [i/ 2 - j ] * b [ i- i/ 2 + j ] ;
}
```

図6.1 入力ループプログラム

```
for i = 0 to N + M

  for j = 0 to  i / 2 {

    if j == 0 then {
      if  i %2 == 0  then
        c [ i ] = a [ i / 2 ] * b [ i / 2 ] ;
      else
        c [ i ] = a [ i - i / 2 ] * b [ i / 2 ]
                + a [ i / 2 ] * b [ i - i / 2 ] ;
    }
    else {
      c [ i ] = c [ i ] + a [ i - i / 2 + j ] * b [ i / 2 - j ]
                        + a [ i / 2 - j ] * b [ i - i / 2 + j ] ;
    }
  }
```

図6.2 処理の内部化

```
for i = 0 to N + M

  for j = 0 to  i / 2 {

    c [ i ] =  if  j == 0  and  i %2 == 0  then
                  a [ i/ 2 ] * b [ i/ 2 ]
               else  if  j == 0  and not ( i %2 ==0 )  then
                  a [ i- i / 2 ] * b [ i / 2 ]
                              + a [ i/ 2 ] * b [ i- i / 2 ]
               else
                  c [ i ] + a [ i- i/ 2 + j ] * b [ i / 2 - j ]
                        + a [ i/ 2 - j ] * b [ i- i/ 2 + j ]
  }
```

図6.3　if文の除去

```
for i = 0 to N + M

  for j = 0 to i / 2 {

    c [ i ] = if j == 0 and i %2 == 0 then
                  a [ i/ 2 - j ] * b [ i/ 2 - j ]
              else if j == 0 and not ( i %2 == 0 ) then
                  a [ i - i / 2 + j ] * b [ i / 2 - j ]
                        + a [ i/ 2 - j ] * b [ i - i / 2 + j ]
              else
                  c [ i ] + a [ i - i/ 2 + j ] * b [ i / 2 - j ]
                        + a [ i/ 2 - j ] * b [ i - i/ 2 + j ]
  }
```

図6.4 添え字の統一

```
for i = 0 to N + M

  for j = 0 to  i / 2 {

    a [ i/ 2 - j ] = a [ i/ 2 - j ]
    b [ i/ 2 - j ] = b [ i/ 2 - j ]
    a [ i- i / 2 + j ] = a [ i- i / 2 + j ]
    b [ i- i / 2 + j ] = b [ i- i / 2 + j ]
    c [ i ] = if j == 0 and i %2 == 0 then
                  a [ i/ 2 - j ] * b [ i/ 2 - j ]
        else if j == 0 and not ( i %2 ==0 ) then
                  a [ i- i / 2 + j ] * b [ i / 2 - j ]
                        + a [ i/ 2 - j ] * b [ i- i / 2 + j ]
        else
                  c [ i ] + a [ i- i/ 2 + j ] * b [ i / 2 - j ]
                        + a [ i/ 2 - j ] * b [ i- i/ 2 + j ]
  }
```

図 6.5　流れ化変換

```
for i = 0 to N+M

  for j = 0 to i/2 {

    a[i- i/2+j] = a[i- i/2+j]
    A[i/2-j] = if i/2-j == i- i/2+j then
                          a[i- i/2+j]
                    else  A[i/2-j]
    b[i- i/2+j] = b[i- i/2+j]
    B[i/2-j] = if i/2-j == i- i/2+j then
                          b[i- i/2+j]
                    else  B[i/2-j]
    c[i] = if j==0 and i%2==0 then
              a[i/2-j]* b[i/2-j]
         else if j==0 and not (i%2==0) then
              a[i- i/2+j]* b[i/2-j]
                    +a[i/2-j]*b[i- i/2+j]
         else
              c[i]+a[i- i/2+j]*b[i/2-j]
                    +a[i/2-j]*b[i- i/2+j]
  }
```

図6.6 分流変換

1: a (i - i / 2 + j)	1: a (i - i / 2 + j)	
2: A (i / 2 - j)	1: a (i - i / 2 + j)	2: A (i / 2 - j)
3: b (i - i / 2 + j)	1: b (i - i / 2 + j)	
4: B (i / 2 - j)	1: b (i - i / 2 + j)	2: B (i / 2 - j)
5: c (i)	1: A (i / 2 - j)	2: B (i / 2 - j)
	3: a (i - i / 2 + j)	4: B (i / 2 - j)
	5: A (i / 2 - j)	6: b (i - i / 2 + j)
	7: c (i)	8: a (i - i / 2 + j)
	9: B (i / 2 - j)	10: A (i / 2 - j)
	11: b (i - i / 2 + j)	

e'-set:

<5, 7> <1, 1> <2, 1> <2, 2> <5, 1>

図6.7 リンク候補集合の導出

パラメータ方程式

$$N(<5,\ 7>) = (0) * co[1] + (0) * co[2] + (1) * co[3]$$

$$N(<1,\ 1>) = (1) * co[1] + (1) * co[2] + (0) * co[3]$$

$$N(<1,\ 1>) = (1) * co[1] + (0) * co[2] + (-1) * co[3]$$

$$N(<2,\ 2>) = (1) * co[1] + (0) * co[2] + (0) * co[3]$$

$$N(<2,\ 2>) = (1) * co[1] + (1) * co[2] + (1) * co[3]$$

計算セル位置関数

$$Ac = c[1] * (i) + c[2] * (i/2) + c[3] * (j)$$

図6.8　構成可能条件

$$N(<5,7>)+N(<1,1>)=N(<2,2>)$$

$$\mathrm{Ac(j)}=N(<2,2>)*(i)-N(<5,7>)*(\lfloor i/2 \rfloor)+N(<5,7>)*(j)$$

$$\delta(<5,7>)+\delta(<1,1>)=\delta(<2,2>)$$

$$\mathrm{Tc(j)}=\delta(<2,2>)*(i)-\delta(<5,7>)*(\lfloor i/2 \rfloor)+\delta(<5,7>)*(j)$$

	$N(<5,7>)$	$N(<1,1>)$	$N(<2,2>)$	$Ac(i)$
1	-1	1	0	$i/2-j$
2	-1	0	-1	$-i+i/2-j$
3	0	1	1	i
4	0	-1	-1	$-i$
5	1	0	1	$i-i/2+j$
6	1	-1	0	$-i/2+j$

	$\delta(<5,7>)$	$\delta(<1,1>)$	$\delta(<2,2>)$	$Tc(i)$
1	1	1	2	$2*i-i/2+j$

表 6.1 構成可能条件を満足する N, δ

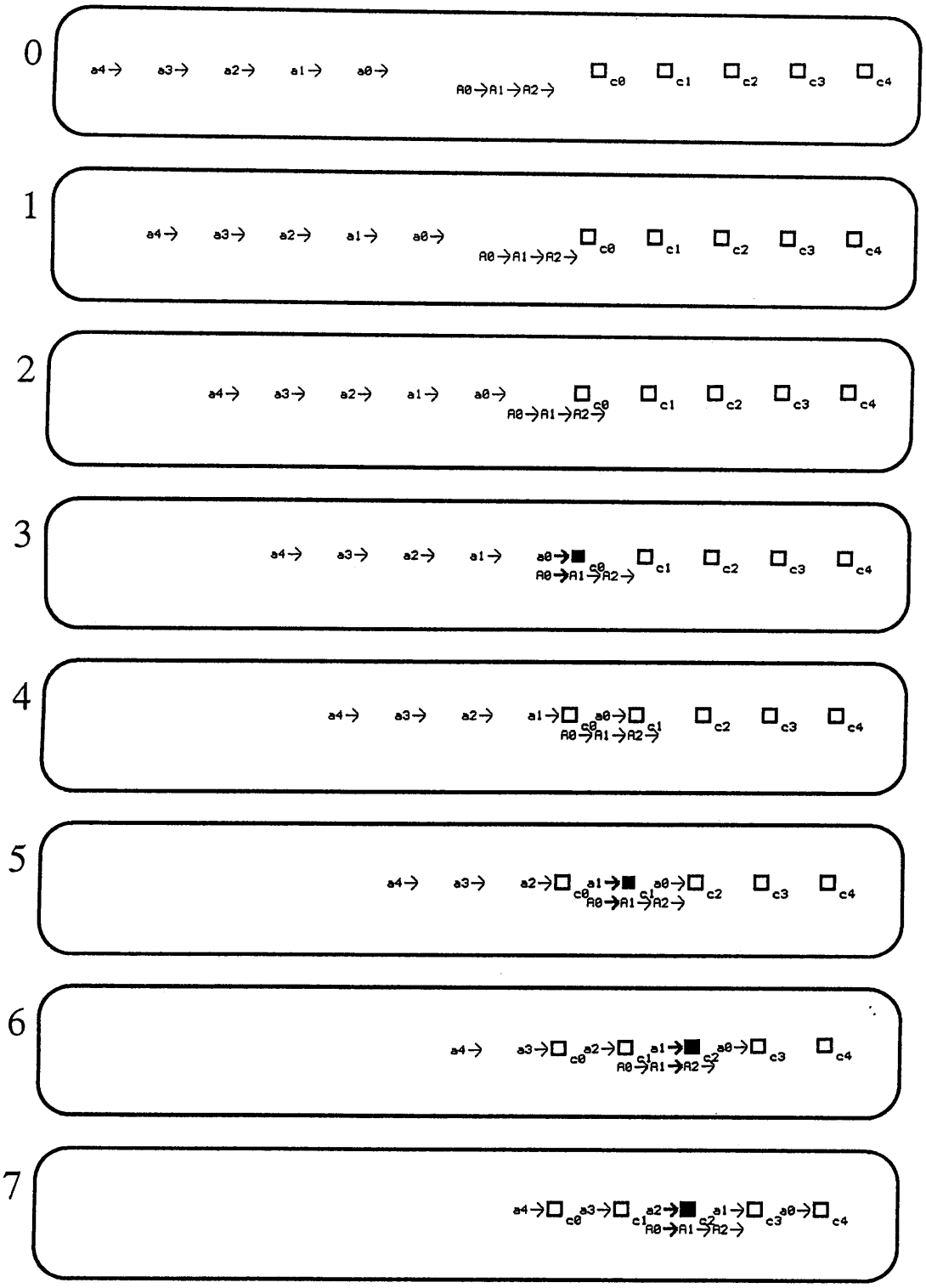

図6.9 タイプ1のシストリックアレーの動作

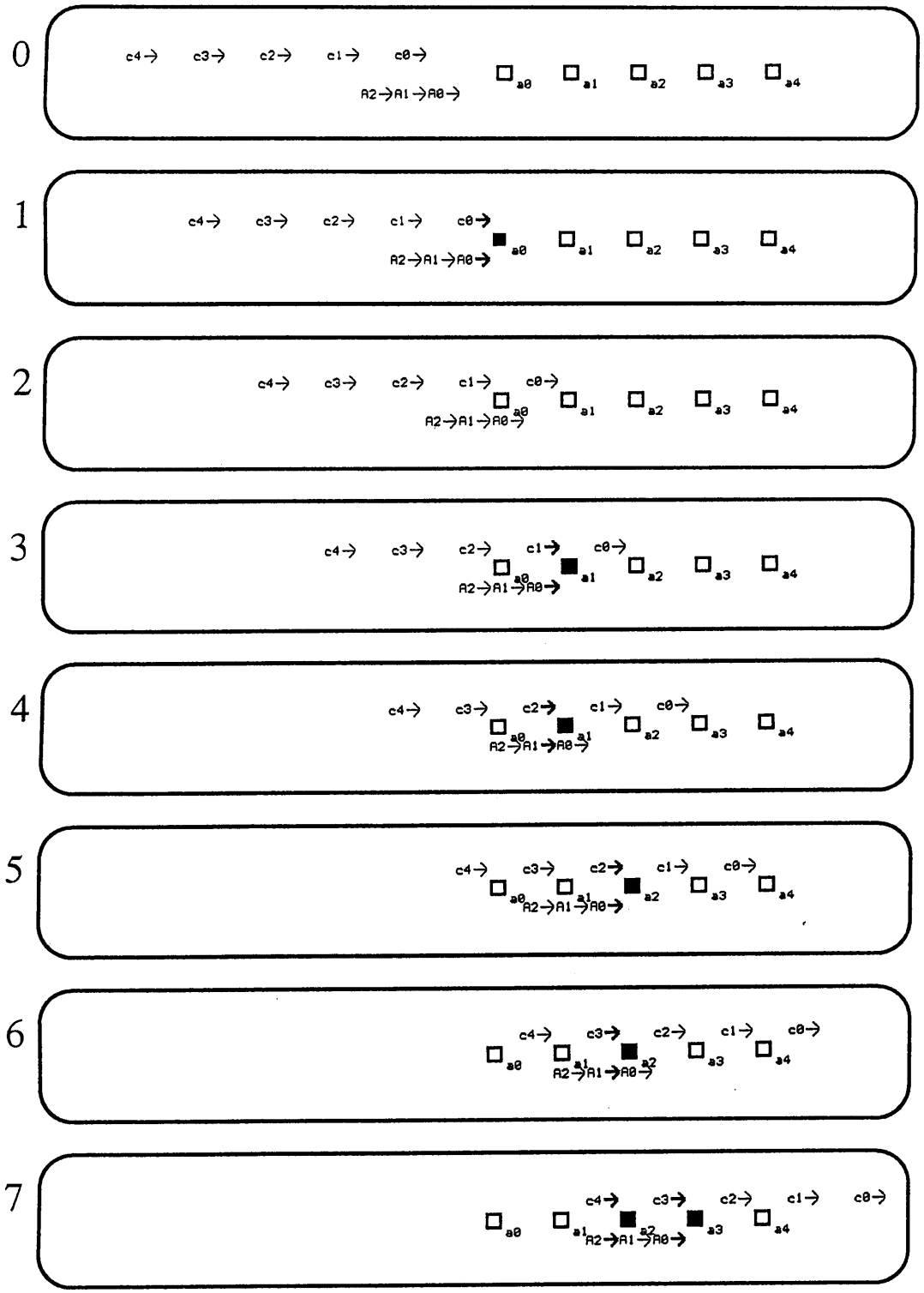

図6.10 タイプ2のシストリックアレーの動作

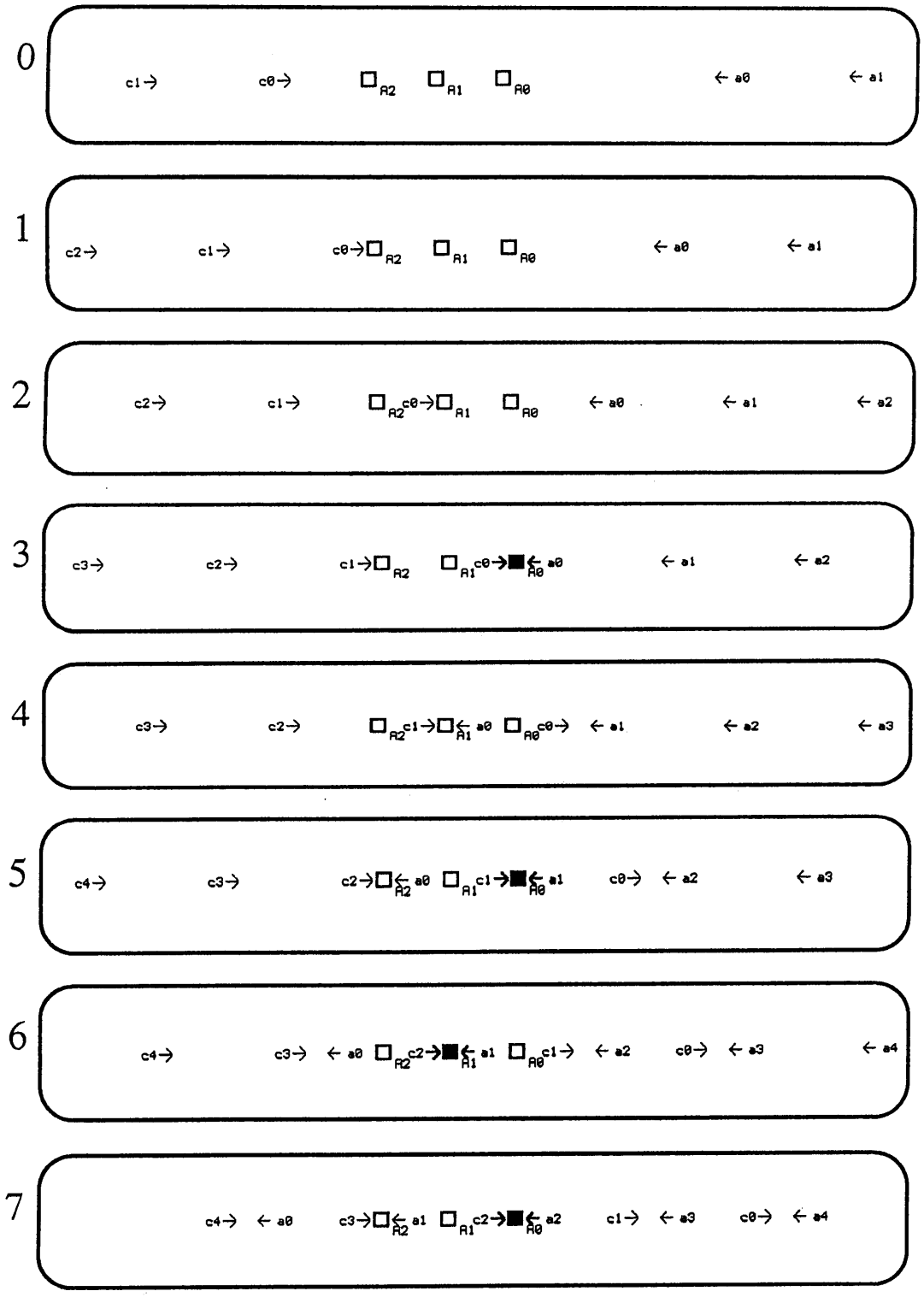

図6.11 タイプ3のシストリックアレーの動作

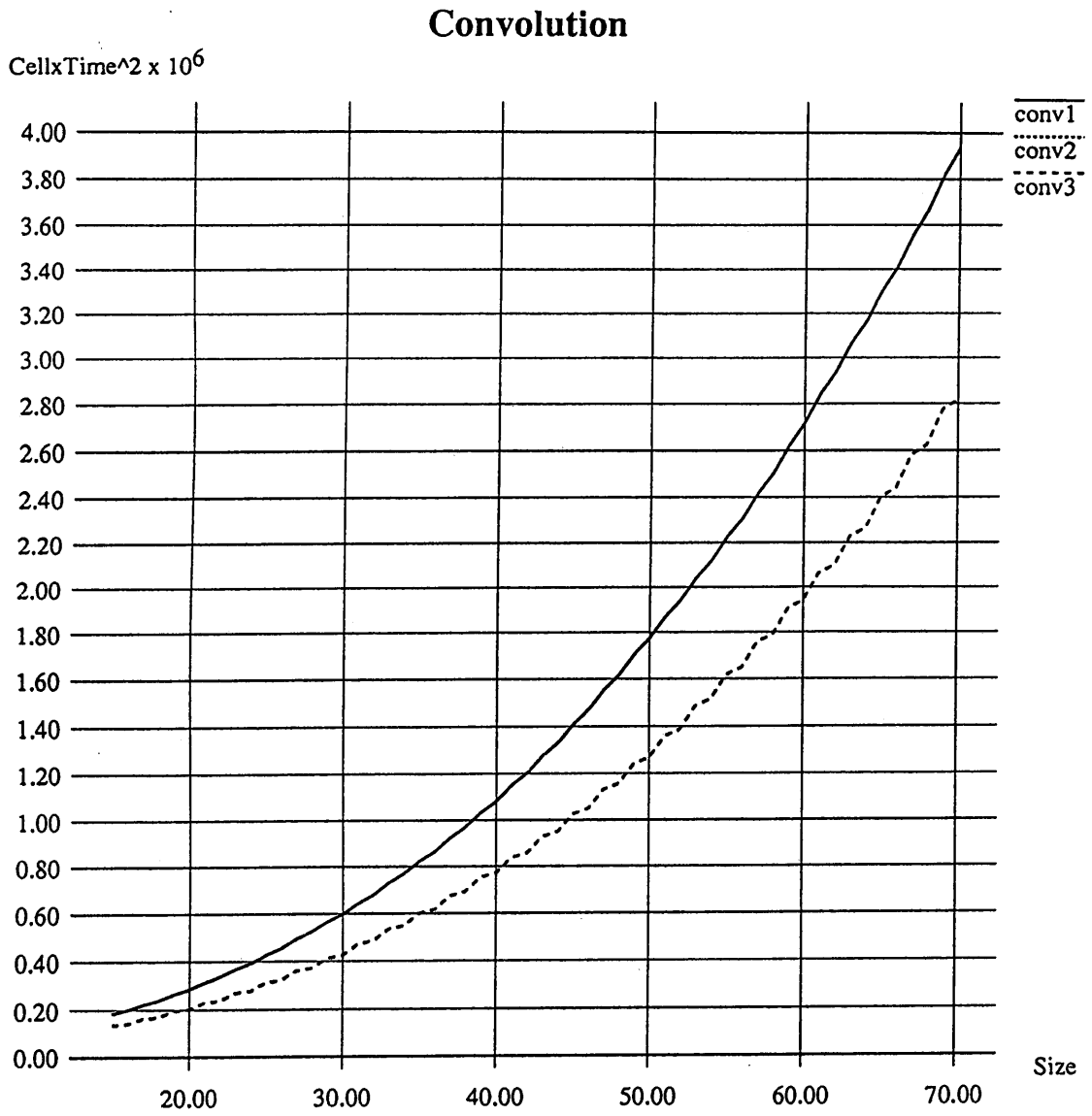

図 6.12　3 つのタイプの性能比較 ((面積)×(計算時間)^2)

第7章

まとめ

本研究では、提案されている自動設計法を計算機上に実現した。その際、入力仕様となるアルゴリズムのクラスを自動設計が容易にできるように明確化した。これは、従来のものより、一般化されているものである。また、逆行関数を求め, データ依存関係を求める部分を設計法そのまま機械的に行わせるのは、非常に困難であることから、この部分は、実際にループインデックスに値を入れて、添え字式の値を計算することにより実行した。

また、計算セル位置関数 Ac, 計算時刻関数 Tc, データ流速度 v, データ流レイアウト関数 Lo を知ることにより、構成可能なシストリックアレーの動作解析のできるグラフィックシミュレータを作成した。これは、X-Window 上に実現されており、マウス操作により、シストリックアレーの動作を容易に解析することができる。

今後の課題として、パラメータの選定の機械化、グラフィックシミュレータの入力となるアレー情報ヘッダーファイル作成の自動化などがあげられる。

謝辞

本研究を行うにあたり、全般的な御指導、御助言を戴いた北陸先端科学技術大学院大学図書館長 木村正行教授、東北大学工学部 阿曽弘具教授、丸岡章教授に心から感謝致します。

また、日頃から多方面にわたり適切な御助言を戴いた東北大学工学部 堀口進助教授、下平博助手に感謝の意を表します。

さらに、東北大学工学部 阿部亨助手、孫寧助手、瀧本英二助手、東北大学大学院大町真一郎氏、成富敬氏には、熱心な御討論、有益な御示唆を戴くとともに、計算機環境の整備を計っていただきました。共に感謝致します。

最後に、日頃よりお世話になった丸岡研究室 (東) の皆様に心より感謝致します。

参考文献

[1] 阿曽弘具 : “ シストリックアルゴリズムの定式化と情報の流れ ”, 電子情報通信学会論文誌 (D), J70-D, No.8, pp.1074–182 (1987 年 6 月)

[2] 阿曽弘具 : “ 並列処理のアルゴリズム ”, bit, Vol.20, No.1, pp.16–24 (1988 年 1 月)

[3] 阿曽弘具 : “ シストリックアレーの自動設計法 ”, 電子情報通信学会論文誌 (D), J71-D, 8, pp.1487–1495 (1988 年 8 月)

[4] 阿曽弘具 : “ ループプログラムに関するシストリックアレー実現可能条件 ”, 電子情報通信学会技術会報, COMP88-13 (1988 年 8 月)

[5] 阿曽弘具 : “ 非同期シストリックアルゴリズムとその設計法 ”, 第 28 回東北大通研シンポジウム「離散アルゴリズム」, pp.163–172 (1991 年 10 月)

[6] CHRISTIAN LENGAUER : “ TOWARDS SYSTOLIZING COMPILATION: AN OVERVIEW ”, PARLE'89 Parallel Architectures and Languages Europe, Vol.2, pp.253–272 (1989 年 6 月)

[7] Chua-Huang Huang, Christian Lengauer : “ The Derivation of Systolic Implementations of Programs ”, Acta Informatica 24, 6, pp.595–632 (1987 年 11 月)

[8] DAN I.MOLDOVAN : “ On the Design of Algorithms for VLSI Systolic Arrays ”, PROCEEDING OF THE IEEE, VOL.71, NO.1, pp.113–120 (1983 年 1 月)

[9] DAN I.MOLDOVAN, JOSE A.B.FORTES : “ Partitioning and Mapping Algorithms into Fixed Size Systolic Arrays ”, IEEE TRANSACTION ON COMPUTERS, VOL.C-35, NO.1, pp.1–12 (1986 年 1 月)

[10] GERHARD FETTWEIS, HEINRICH MEYR : " High-Rate Viterbi Processor: A Systolic Array Solution ", IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL.8, NO.8, pp.1520–1533 (1990 年 10 月)

[11] GUO-JIE LI, BENJIAMIN W.WAH : " The Design of Optimal Systolic Arrays ", IEEE TRANSACTIONS ON COMPUTERS, VOL.C-34, NO.1, pp.66–77 (1985 年 1 月)

[12] H.T.Kung : " Why Systolic Architectures? ", IEEE COMPUTER, January 1982, pp.37–46 (1982 年 1 月)

[13] 飯国洋二, 酒井英昭, 得丸英勝 : " 1 次元シストリックアレーの設計法 ", 電子情報通信学会論文誌 (D), J71-D, No.8, pp.1480–1486 (1988 年 8 月)

[14] 三浦大祐, 阿曽弘具, 稲垣康善 : " 多重ループプログラムを処理するシストリックアルゴリズム ", 電子情報通信学会論文誌 (D), J70-D, No.3, pp.515–524 (1987 年 3 月)

[15] Monica S.Lam, Jack Mostow : " A Transformational Model of VLSI Systolic Design ", IEEE COMPUTER, February 1985, pp.42–52 (1985 年 2 月)

[16] Pierrick Gachet, Brigitte Joinnault, Patrice Quinton : " SYNTHESIZING SYSTOLIC ARRAYS USING DIASTOL ", Systolic Arrays, Adam Hilger, pp.25–36 (1987 年)

[17] SUN-YUAN KUNG, K.S.ARUN, RON J.GAL-EZER, D.V.BHASKAR RAO : " Wavefront Array Processor: Language, Architecture, and Applications ", IEEE TRANSACTION ON COMPUTERS, VOL.C-31, NO.11, pp.1054–1066 (1982 年 11 月)

[18] SUN-YUAN KUNG : " On Supercomputing with Systolic/Wavefront Array Processors ", PROCEEDINGS OF THE IEEE, VOL.72, NO.7, pp.867–884 (1984 年 7 月)

[19] S Y Kung : " VLSI ARRAY PROCESSORS ", Systolic Arrays, Adam Hilger, pp.7–24 (1987 年)

[20] 梅尾博司 : " 超並列計算機アーキテクチャとそのアルゴリズム ", 共立出版, pp.27–59 (1991 年 11 月)

[21] V.K.Prasanna Kumar, Yu-Chen Tsai : " On Synthesizing Optimal Family of Linear Systolic Arrays for Matrix Multiplication ", IEEE TRANSACTIONS ON COMPUTERS, VOL.40, NO.6, pp.770–774 (1991 年 6 月)

[22] 吉田紀彦 : " プログラム変換に基づくシストリックアレイの導出 ", 情報処理学会論文誌, Vol.30, No.12, pp1530–1537 (1989 年 12 月)