卒業論文

最適並列処理方式設計システムの構築

東北大学工学部　阿曽研究室４年

佐藤　　英

目 次

図目次

表目次

第1章

序論

1.1 本研究の背景及び目的

近年、計算機の高速化への期待や、計算機の構成素子の低価格化から、並列処理に対する関心が高まってきており、並列処理に関する多くの研究がおこなわれている。シストリックアレーは、現在、並列処理を実現するアーキテクチャのひとつとして、知られている。

シストリックアレーは、1978年、Kungによって提案され、多数の演算素子(セル)を規則的に配置し、データをパイプライン的に流すことによって、各演算を並列的におこなうアーキテクチャである。構造が一様であるため拡張性に優れ、また通信が局所的であるなどの利点を兼ね備えており、既に信号処理、記号処理などの分野で一部実用化がされている。

しかしその反面、動作がわかりにくいため、設計が難しいといった欠点が指摘されている。シストリックアルゴリズムの設計に関してはいくつかの方法が提案されており、[2]による方法はそのひとつである。この方法は、その設計の有効性が確かめられており、既にそれをX-Window上でシミュレータ化するシステム(Support system for Design and development of Systolic algorithms : SDS) [3]が開発されているが、SDSの完全自動化はなされていない。

SDSの完全自動化への障害の一つとなっているものが、最適化の問題である。SDSはソースプログラムからデータ間に成立する依存関係を導出し、それを方程式の形で出力するが、その方程式の解は性質上、一意に決定できないため、ユーザが適当な値を決めてやらなければならないのが現状である。

そこで本研究では、SDSに与えるプログラムに対して、いくつかの評価値を用いて最適化の計算をおこない、最適であると思われる方程式の解をSDSのシミュ

レータに与えてやることによって、完全自動化を達成することが目的である。

1.2　本論文の構成

本論文の構成は以下のとおりである。
第１章では、序論として、研究の背景及び目的について述べる。
第２章では、本研究の基盤となっている自動設計法について述べる。
第３章では、シストリックアルゴリズム開発支援システム (SDS) について述べる。
第４章では、本研究で用いている配列の評価値、評価方法、及び作成したSDSの最適化機能について述べる。
第５章では、いくつかのプログラムに関して、評価値が左右されやすい要素について述べる。
第６章では、本研究の結論及び今後の問題点について述べる。

第 2 章

シストリックアレーの自動設計法

2.1 自動設計の原理

ここで用いる自動設計法は文献 [2] によるものである。

シスリックアレーとして構成するループプログラムは、流れるデータ間に成立する関係 (逆行関係と呼ばれる) を導出するため、まず標準形に変換される。標準形に変換する手順として、ループプログラムに対して以下の変換をおこなう。

1. 処理内部化、if 文除去
2. 添え字統一
3. 流れ化変換
4. 分流変換

これらのそれぞれの処理について簡単に示す。

2.1.1 処理内部化、if 文除去

処理内部化は、ループプログラムを一つの for 文でくくる処理であり、また、if 文除去は、ループプログラム中に表われる if 文に対し、プログラムの意味を保存しながら除去する処理である。

2.1.2　添え字統一

逆行関係導出の際には、添え字は重要な役割を張たすことになる。添え字統一は、プログラム中に表われる何種類かの添え字のうち、意味的に等しくなる場合には添え字を片方に統一する処理である。

2.1.3　流れ化変換

ループプログラム中に、参照変数としてだけ登場している配列変数について、右辺と左辺が同一の項である割当文を本体の最初に付加する処理である。この変換は、データがパイプライン的に流れることに対応したものである。

2.1.4　分流変換

流れ化変換によって付加された文で、同一の配列を持つ 2 つの参照変数のうち、その添え字式が排反でないときは、異なるインデックスに対して 2ヶ所で値が設定され、同一データを 2ヶ所に流すことを意味する。このとき、片方の参照を新しい変数名で置き換え、付加する処理が分流変換である。

以上の処理を、コンボリューション (多項式の積、図 2.1) のプログラムに適用した場合の流れを図 2.2から図 2.5に示す。

2.2　パラメータ方程式導出

ソースプログラムから標準形への変換を終えたら、標準形の添え字の種類からどの変数がデータとして流れているか (リンク候補) を導出し、次に流れるデータ間に成立する関係 (逆行関係) をパラメータ方程式の形式で導出する。この方程式は、位置に関する関数 (近傍関数)Nと、時間に関する関数 (時間調整写像)δについてそれぞれ考慮し、これによりデータの進行方向及び速度を決定することができる (図 2.6)。

コンボリューション（多項式の積）

```
for i = 0 to 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] ;
}
```

入力データ

$$a_0, a_1, \cdots, a_{M-1}, a_M$$
$$b_0, b_1, \cdots, b_{M-1}, b_M$$

出力データ

$$c_0, c_1, \cdots, c_{M-1}, c_M$$

$$c_i = \sum_{j=0}^{i} a_{i-j} b_j$$

図 2.1: コンボリューションプログラム

```
for i = 0 to N{
    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] ;
}
```

⇓

```
for i = 0 to N
    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 2.2: 処理内部化、if 文除去

```
for i = 0 to N
    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]
    }
```

⇓

```
for i = 0 to N
    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 2.3: 添え字統一

```
for i = 0 to N
    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]
    }
```

⇓

```
for i = 0 to N
    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]
    }
```

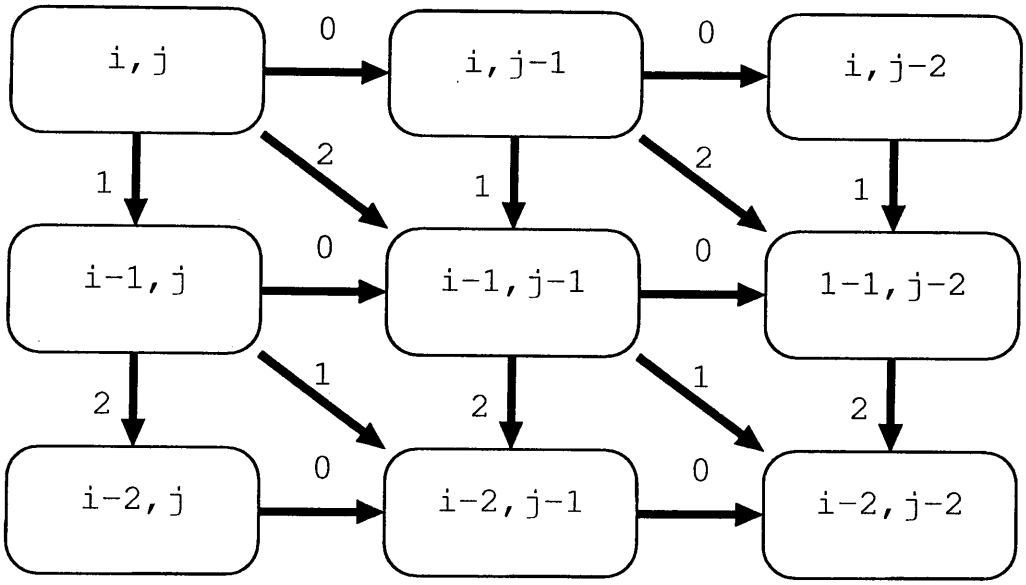
図 2.4: 流れ化変換

```
for i = 0 to N
    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]
    }
```

⇓

```
for i = 0 to N
    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 && i%2 == 0 then A[i/2 − j] * B[i/2 − j]
                else if j == 0 && 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 2.5: 分流変換



i:even

$N(0)+N(1)=N(2)$

$\delta(0)+\delta(1)=\delta(2)$

<parameter equation>

※各リンク候補は以下の変数である

0：N(<5,7>)　　1：N(<1,1>)　　2：N(<2,2>)

図 2.6: データ間の逆行関係とパラメータ方程式

第 3 章

シストリックアルゴリズム開発支援システム (SDS)

3.1 SDS の概要

[3] によるシストリックアルゴリズム開発支援システム (SDS) は、人間の主観によることなく半自動的にシストリックアレーを構成できることを目的に構築されたシステムである。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の逆行関係を導出する部分と、シストリックアルゴリズムを X-Window 上でシミュレートできるグラフィックシミュレータ部とからなる。図 3.1に SDS の処理の流れを示す。

3.2 SDS の利用法

SDS の利用法を以下に示す。

1. 入力ループプログラムから構成可能条件を導出する。
2. 同条件を満足するような位置、時間のパラメータを選定する。
3. 選定されたパラメータから構成されるシストリックアレーをシミュレータ上に実現する。
4. パラメータをいくつか変えてみて同様に実現していき、意図にあったものを選ぶ。

3.3　SDS の問題点

SDS は、ループプログラムから構成可能条件を導出する部分、及び構成可能なパラメータから X-Window 上にシミュレートする部分は自動化されているが、構成可能なパラメータを選定する作業が現在人間の手によっておこなわれており、セル数、ステップ数の面から考慮して最適と思われる構成を得るには、パラメータの選定をユーザが何度も試行錯誤的におこなわなければならないといった問題点がある。そこで本研究では、すべてのデータ流がとりうる近傍関数 N、時間調整関数δの値のとり方から、必要最低限のもののみを取り出して、セル数、ステップ数といった評価値を計算し、独自の評価関数を用いて評価値を計算することによって、最適な構成のシストリックアレーを自動的に得られるようにすることを目的としている。

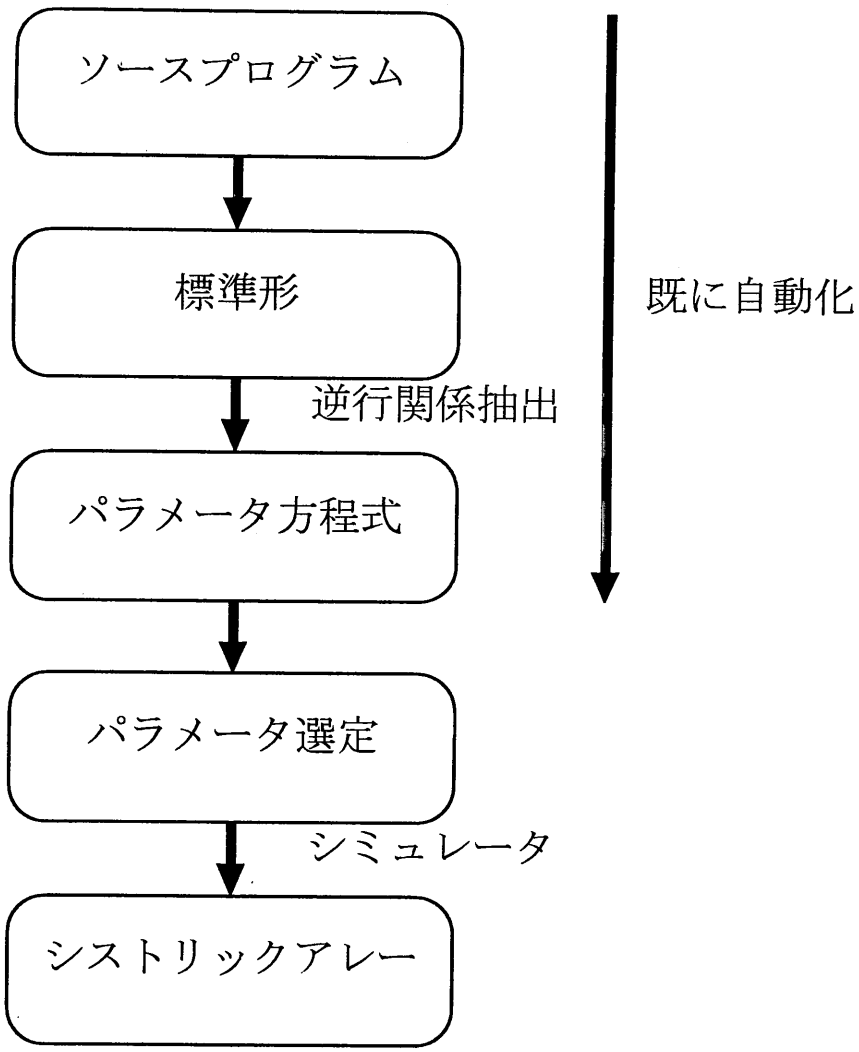


図 3.1: SDS の処理の流れ

第4章

シストリックアレーの評価

4.1 評価基準

シストリックアレーを評価する際には、以下の二つの値をその評価値として考慮する。

1. セル数
2. ステップ数

ここで、ステップ数を定義しておく。

$$
\begin{aligned}
T &= T_{in} + T_{ex} + T_{out} = T_{arrayout} - T_{arrayin} \\
T_{in} &= T_{exfirst} - T_{arrayin} \\
T_{ex} &= T_{exlast} - T_{exfirst} \\
T_{out} &= T_{arrayout} - T_{exlast}
\end{aligned}
$$

T	ステップ数
T_{in}	データ入力時間
T_{ex}	計算時間
T_{out}	データ出力時間
$T_{exfirst}$	最初に計算がおこなわれた時刻
T_{exlast}	最後に計算がおこなわれた時刻
$T_{arrayin}$	最初にデータのアレー入力がおこなわれた時刻
$T_{arrayout}$	最後にデータのアレー出力がおこなわれた時刻

4.2　評価手順

評価は、以下の手順でおこなう。

1. 近傍関数 Nのとる値の絶対値の最大値 N_v及び構成されるアレーの次元数 dim をユーザが決定する。
2. N_v及び流れるデータの個数から、とりうるデータの進行方向のパターンを考える。
3. そのパターンのなかから、回転、反転、拡大、縮小によって重複するものを取り除く。
4. それらのパターンについて、時間調整写像δを決定し、セル数、ステップ数を計算する。
5. 計算したもののうち、最簡機能条件を満たさないものは評価の対象からはずす。
6. 評価値が最も優れているものを最適な構成と考えて、そのときの各パラメータ値をシミュレータにわたす。

ここで、最簡機能条件とは以下のような条件である。

最簡機能条件：
セル機能を簡単にするために、同一時刻、同一セルでは
高々一つのインデックスにおける計算しか実行されない
ものとする。

$$A_c(\mathrm{j}) = A_c(\mathrm{j}\prime) \Rightarrow T_c(\mathrm{j}) \neq T_c(\mathrm{j}\prime)$$

ただし、A_cは計算セル位置関数、T_cは計算時刻関数を表わす。

以下ではこの評価方法について具体的に示す。

4.2.1　近傍関数 Nの決定方法

与えられた N_vと dim から、可能なデータの流れ方の組み合わせを考える。ここでは例として $N_v = 1$、$dim = 2$ の場合について考える。この場合、1 本のデータの進行方向は座標が

$$(x, y) \quad \text{ただし} x = -1, 0, 1 \quad y = -1, 0, 1$$

で表わされる9通りが考えられるので、流れるデータの本数が3本の場合は全部で729通りの進み方が考えられる。このなかから、回転、反転、拡大、縮小により重複するものを取り除くと、図4.1に表わされるものだけが残る。残ったこれらのパターンに対して、時間調整写像δを決定する。

4.2.2 時間調整写像δの決定方法

δは、データが一区画進むのに要する時間なので、最適な状態に近づけるためには、なるべく小さく設定するとよい。ただし、考えている単位が時間なので、$\delta > 0$とする。

また、例外として、同一方向に進むデータが複数存在する場合は、それぞれの値をずらしてタイミングをはかるものとする。

ここでは例として$\delta(a) + \delta(b) = \delta(c)$ の式が成立している場合を考える。図4.2の場合は、δの関係式を満たす最小の正数を選べばよい。図4.3の場合は、同一方向に流れるデータが2本存在するので、片方のδの値をずらし、なおかつδの関係式を満たすように選ぶ。

4.2.3 評価関数

計算の結果、求められたセル数、ステップ数から、以下の評価式によって評価値を計算する。

$$f_{opt} = g_c \frac{c_{ave} - c}{\sigma_c} + g_s \frac{s_{ave} - s}{\sigma_s}$$

$$(g_c + g_s = 1)$$

g_c : セル重み
c : セル数
c_{ave} : 平均セル数
σ_c : セル数の標準偏差
g_s : ステップ重み
s : ステップ数
s_{ave} : 平均ステップ数
σ_s : ステップ数の標準偏差

ここで、セル重み、ステップ重みは、ユーザが任意に決定できるものとする。

4.3　SDS の最適化機能

これまでの評価方法を用いて、最適な構成を与える機能を SDS の付加機能として構成した。その流れを以下に示す。なお、図 4.4から図 4.6 において、実際に最適化機能を実行したときの様子を示しているが、ソースプログラムとしてコンボリューションを使用している。

1. 従来の SDS が、ソースとなるループプログラムから逆行関係を抽出する。
2. SDS が、ユーザに N_v、dim、各重みなどの値を問い合わせる。(図 4.4)
3. 各値を与えると、考えられるデータの流れ方の個々のパターンに対して計算、評価をおこない、計算結果を格納するファイルを作成する。(図 4.5)
4. 計算の結果、評価値が最も優れているものを最適とみなし、そのときの位置、時間のパラメータを SDS のシミュレータにわたす。
5. X-Window 上に、最適な構成のアレーがシミュレートされる。(図 4.6)

なお、評価関数に関しては、前節で定義した関数の他に、VLSI システムを評価する際に一般的に使用される評価式

$$\sqrt{(\text{面積})} \times (\text{時間})$$

を、「面積」を「セル数」に、「時間」を「ステップ数」に置き換えた式の平方、つまり

$$(\text{セル数}) \times (\text{ステップ数})^2$$

の式を使用して最適化をおこなうように設定することもできる。

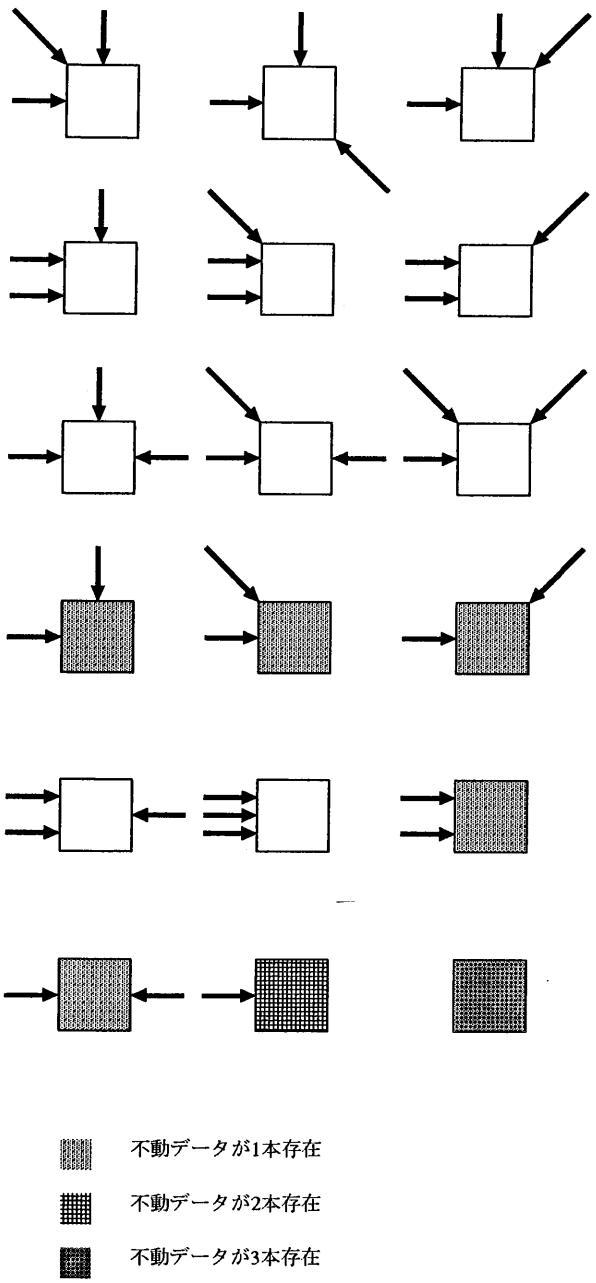
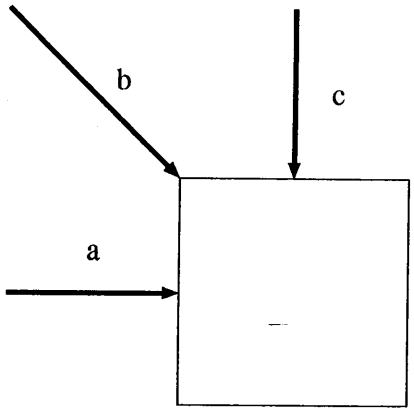
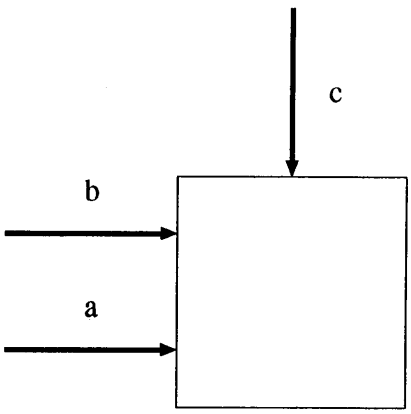


図 4.1: $N_v = 1$、$dim = 2$ のときの 3 本のデータの流れ方のパターン



$\delta(a)+\delta(b)=\delta(c) \rightarrow \delta(a)=1,\delta(b)=1,\delta(c)=2$

図 4.2: データの流れ方の例（１）



$\delta(a)+\delta(b)=\delta(c) \rightarrow \delta(a)=1,\delta(b)=2,\delta(c)=3$

図 4.3: データの流れ方の例（２）

```
#define NVAL    2  /* absolute value of neighborhood function */
#define DIM     1  /* dimension of structured array */

#define N       4
#define K       2

#define G_CELL  0.5  /* grade of cell */
#define G_STEP  0.5  /* grade of number of step */
/*  Please set parameters G_CELL and G_STEP for G_CELL+G_STEP=1  */

#define SORT    6
/*  Please select under.
        1:number of cell   2:whole steps
        3:input steps  4:output steps   5:calculated steps
        6:f_opt   7:cell*(step)^2  */
```

図 4.4: 各パラメータ設定ヘッダファイル

optimize result　　DIM=1　NVAL=2　G_CELL=0.5000　G_STEP=0.5000

NX1	NX2	NX3	DT1	DT2	DT3	cell	step	Tin	Tout	Tex	f_opt	cell*T^2
-1	1	0	1	1	2	3	14	3	3	8	1.1022	588
0	-1	-1	1	1	2	5	12	3	1	8	0.7233	720
-1	0	-1	1	1	2	5	12	1	3	8	0.7233	720
-2	1	-1	1	1	2	5	20	5	7	8	-0.2500	2000
1	-2	-1	1	1	2	7	20	7	5	8	-0.8722	2800
-1	-1	-2	1	2	3	8	22	5	5	12	-1.4266	3872

図 4.5: 計算結果格納ファイル

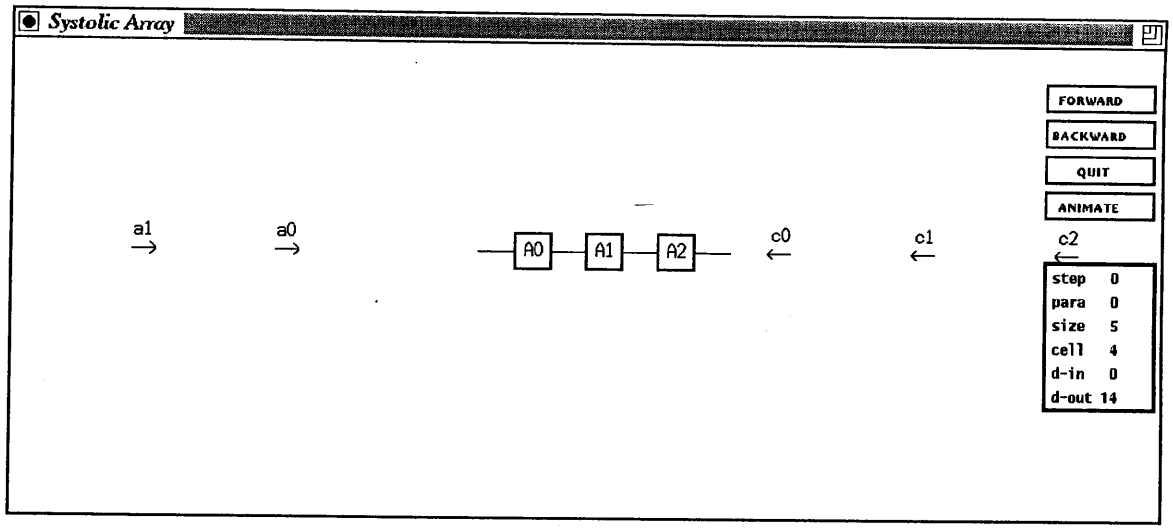


図 4.6: X-Window 上に実現された最適な構成のシストリックアレー

第5章

評価値が左右される要素

5.1 評価値が左右される要素

どのようなプログラムに対しても、最適な構成となるアレーを決定するパラメータや、そのときの評価値などが、ある一般式で表現されれば非常に理想的であるが[4][8]、現実にはそれは簡単ではない。むしろ、それらの式が一般式として表現できるのはごく少数のプログラムに限られる。そのため、流れるデータがどのような要素を含んでいるときに評価値が変化するのかを考えることは非常に有効かつ重要である。ここでは、前章で考えた評価値が、どのような要素に左右されやすいかを考察、計算する。

なお、計算に用いたプログラムは、帯行列積計算プログラム(図5.1)及びコンボリューションである。

5.2 データの方向性

流れるデータがなす角度の最大値が180°のときと、それ以下の場合とで、それぞれのデータ出力時間の平均を計算したところ、表5.1のようになった。

この結果より、帯行列積計算プログラム及びコンボリューションプログラムに関しては、データ流の最大角度が180°以下の場合の方が、計算したすべての場合においてデータ出力時間が小さくなっており、データ出力時間がデータの最大角度（方向性）に関連しているといえる。

5.3　不動データの有無

動かないデータ（不動データ）が存在するときと、存在しないときとで、それぞれのセル数の平均を計算したところ、表5.2のようになった。

この結果より、帯行列積計算プログラム及びコンボリューションプログラムに関しては、データ流の中に不動データが含まれている場合の方が、計算したすべての場合においてセル数が少なくなっており、セル数が不動データの有無に関連しているといえる。

帯行列積

```
for i = 1 to M − K + 1
    for j = 0 to K
{
        if j == 0 then y[i] = 0;
        else y[i] = y[i] + w[j] * x[i + j − 1];
}
```

入力データ

$$\mathbf{W} = \begin{bmatrix} w_1 & w_2 & \cdots & w_K & 0 & \cdots & 0 \\ 0 & w_1 & \cdots & w_{K-1} & w_K & \cdots & 0 \\ & 0 & \ddots & & & & \\ & & & & & \ddots & \\ & & & & & & w_K \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_M \end{bmatrix}$$

出力データ

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{M-K+1} \end{bmatrix} \qquad (\mathbf{Y} = \mathbf{WX})$$

図 5.1: 帯行列積計算プログラム

帯行列積計算の平均データ出力時間 ($dim = 1$)

Nv	$M = 4, K = 2$		$M = 6, K = 3$	
	$\theta < 180°$	$\theta = 180°$	$\theta < 180°$	$\theta = 180°$
10	1.176	2.032	1.559	3.714
20	1.046	1.225	1.146	1.671
30	1.021	1.117	1.068	1.308

コンボリューションの平均データ出力時間 ($dim = 1$)

Nv	$M = 4$		$M = 6$	
	$\theta < 180°$	$\theta = 180°$	$\theta < 180°$	$\theta = 180°$
10	1.147	2.032	1.412	4.222
20	1.038	1.255	1.108	1.796
30	1.018	1.117	1.050	1.366

θ：データの最大角度

表 5.1: 方向性とデータ出力時間の関係

帯行列積計算の平均セル数 ($dim = 1$)

Nv	$M = 4, K = 2$		$M = 6, K = 3$	
	involve	not involve	involve	not involve
10	2.750	8.871	3.750	15.430
20	2.750	8.969	3.750	15.861
30	2.750	8.986	3.750	15.936

コンボリューションの平均セル数 ($dim = 1$)

Nv	$M = 4$		$M = 6$	
	involve	not involve	involve	not involve
10	3.250	8.871	4.500	15.452
20	3.250	8.969	4.500	15.866
30	3.250	8.986	4.500	15,939

表 5.2: 不動データとセル数の関係

第 6 章

結論

本研究では、SDS に最適化機能を付加することとともに、帯行列積計算、コンボリューションといった、各種問題の基礎となるプログラムについて、その問題をアレー化したときにどのような要素によって評価値が変化するかを調べた。

SDS の最適化機能に関しては、評価値を計算する際に計算に用いるセル重み、ステップ重みをユーザが任意に決定することができるようにして、より一層要求に近い構成が得られるように作成した。また、もう一つの評価基準として VLSI システムの評価において頻繁に用いられる評価関数も使えるように設計した。

また、各プログラムの評価値の、要素による変化に関しては、帯行列積計算、コンボリューションに関して、データの方向性がステップ数に、不動データの有無がセル数にそれぞれ関連していることを確かめた。

今後の課題としては、今回調べた、要素に関する評価値の変化は、他のプログラムに関しても一般的に成立するかどうかの検証が残っている。また、作成した最適化機能は、流れるデータの個数に制限があるため、まだ汎用性が低い。しかし、単純に、扱えるデータの個数を増やしても、その増加に従って計算量は指数関数的に増大していくといった問題も残る。そのため、もっと簡単に最適化をおこなえるアルゴリズムが存在するかどうかといった点も考慮すべき問題であろう。

謝辞

本研究を進めるにあたり、全般的に御指導をいただいた、東北大学工学部阿曽弘具教授に心より感謝いたします。

また、いつも熱心に御討論していただいた東北大学情報処理教育センター大町真一郎氏、東北大学大学院工学研究科博士課程後藤英昭氏、黒岩丈介氏、東北大学大学院工学研究科修士課程角田恵実氏に深く感謝いたします。

最後に、計算機を日頃より快適な環境に整備していただいた、東北大学工学部阿曽研究室の諸先輩方、ならびに、ときに励ましあいながら研究活動を共にした同研究室の同期生たちに深く感謝いたします。

参考文献

[1] 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の構成法", 信学論 (D),Vol.J70-D No.3 pp.515-524 87/3

[2] 阿曽弘具:"シストリックアレーの自動設計法", 信学論 (D),Vol.J71-D No.8 pp.1487-1495 88/8

[3] 白石　勉:"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", 92/3 東北大学大学院工学研究科修士論文

[4] 飯国洋二、酒井英昭、得丸英勝:"データ依存グラフを用いたシストリックアレーの自動設計", 信学論 (A),Vol.J71-A,No.6 pp.1282-1290 88/6

[5] 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", 信学論 (D),Vol.J71-D,No.8 pp.1480-1486 88/8

[6] 前場隆史、辰巳昭治:"正射影に基づくシストリックアルゴリズム設計手法", 信学論 (A),Vol.J71-A,No.10 pp.1878-1887 88/10

[7] 小川誠治、前場隆史、阿部健一:"多次元シストリックアレーの系統的設計手法", 信学論 (D-I),Vol.J75-D-I,No.9 pp.879-881 92/9

[8] Dan I.Moldovan: "On the Design of Algorithm for VLSI Systolic Arrays", Proc.IEEE,71,1 pp.113-120 1983