Vol. 27 No. 3 情報処理学会論文誌 Mar. 1986

最適実時間シストリック・多項式除算アルゴリズム†

梅尾博司†† 杉岡俊幸†††

H. T. Kung らの提案以来，数多くのシストリック・アルゴリズムが考案されている．本論文では，実時間で多項式除算を実行するシストリック・アルゴリズムを提案する．本アルゴリズムは，n次の多項式をm次の多項式 ($n \geqq m$) で割ったときの商と余りを，$\lceil \min(n-m, m)/2 \rceil + 0(1)$個のセルを使用し，$n+0(1)$ステップで得ることができる．このアルゴリズムはさきに文献4）で報告されている実時間パラレル・コンボルバのデータ・ルーティングを変更することにより得られたものである．

1. まえがき

H. T. Kung らの提案以来，数多くのシストリック・アルゴリズムが考案されている[1)-5)]．シストリック・アレイは，入出力を直列にアレイ上での処理を並列 (serial-I/O pararell-processing) に，かつそれらの動作を重畳的に行うという特徴を持つ．

本稿では，実時間で多項式除算を実行するシストリック・アルゴリズムを提案する．本アルゴリズムは，n次の多項式をm次の多項式 ($n \geqq m$) で割ったときの商と余りを，$\lceil \min(n-m, m)/2 \rceil + 0(1)$個のセルを使用し，$n+0(1)$ステップで得ることができる．このアルゴリズムは，先に文献 4) で報告された実時間パラレル・コンボルバのデータ・ルーティングを変更することにより得られたものである．$2n$ あるいは $3n$ ステップの時間計算量を持つ多項式除算アルゴリズムが 1), 3) にて報告されているが，本アルゴリズムは，係数まで考慮した場合の最適実時間アルゴリズムの一つであり，シストリック・アーキテクチャを仮定する限りこれ以上速くはできない．

まず，2章では以下での議論の準備として，実時間パラレル・コンボルバに関する説明を与える．3章では，シストリック・除算アルゴリズムを提案する．

2. 実時間パラレル・コンボルバ

二つの数列

$$A=(a_0, a_1, \cdots, a_{n-1}) \quad (1)$$

$$B=(b_0, b_1, \cdots, b_{m-1}) \quad (2)$$

のコンボリューションとは，次式で定義される数列

$$C=(c_0, c_1, \cdots, c_{n+m-1}) \quad (3)$$

$$\begin{aligned} c_0 &= a_0 b_0 \\ c_1 &= a_1 b_0 + a_0 b_1 \\ c_i &= \sum_{j=0}^{m-1} a_{i-j} b_j \quad (4) \\ &\vdots \\ c_{n+m-1} &= 0 \end{aligned}$$

である．ただし，$k<0$ 及び $k \geqq n$ なる k について，$a_k=0$ と考える．

図1に示すような1個のバッファと無限個のセル C_i, $i=1, 2, \cdots$ から構成されるシストリック・アレイ M を考える．Mは，上記のコンボリューションを実時間で計算できることが知られている[4)]．入力，出力はアレイの左端のみから行われる．1ステップにつき (a_i, b_i) の対1個が，入力ラインを通じてMにホスト計算機から入力される．すなわち，$t=0$ 時に (a_0, b_0) が，$t=1$ 時に (a_1, b_1) が，$t=2$ 時に (a_2, b_2) が…という具合である．出力は出力ラインより $t=k$ 時から1個ずつ $c_0, c_1, c_2, c_3, \cdots$ の順になされる．1ステップにつき1個である．ここで k は定数である．

次の意味で実時間と呼ぶ．すなわち，任意の i に対し，時刻 $t=i+k$ 時に c_i が出力される．

基本セル：各セル C_i ($i \geqq 1$) は5個のレジスタ R_j ($1 \leqq j \leqq 4$) 及び R_e からなる（**図2**参照）．

各 R_i はそれぞれ二つのサブレジスタ A_i, B_i に分割される．これらは，それぞれ a_i, b_i のデータ保持のために使用される．セル C_i 上のレジスタ R_j の時刻 t における内容を $R_j(i)$ で表す．他のレジスタについても同様，レジスタRが空であること，空でないこと，レジスタ R_2 から R_1 への内容のコピーを通常の記法，すなわち，$R=\phi$, $R \neq \phi$, $R_1 \leftarrow R_2$ で表現する．バッファは三つのレジスタを持ち，これらはセルの R_3, R_4 ならびに R_e に相当する．

各セルのレジスタは次の目的のために使用される．

† An Optimum Real-Time Systolic Polynomial Division Algorithm by HIROSHI UMEO (Department of Electronic Engineering, Faculty of Engineering, Osaka Electro-Communication University) and TOSHIYUKI SUGIOKA (Kamitani Electronic Industry).

†† 大阪電気通信大学工学部応用電子工学科
††† 神谷電子(株)

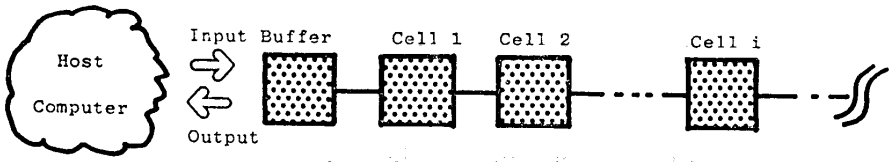


図 1 シストリック・アレイ
Fig. 1 Systolic array.

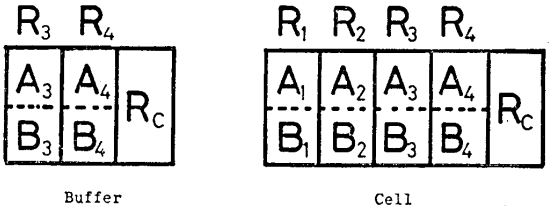


図 2 コンボルバMのセル及びバッファの内部構成
Fig. 2 A systolic cell and a buffer of M.

R_1, R_2: データの保持レジスタとして,
R_3, R_4: 右隣接セルへのデータ伝達のためのパイプラインとして,
R_c: コンボリューションの部分積和の記憶レジスタとして使用される.

基本セルの動作: セルの動作は R_1, R_2, R_3, R_4 上へのデータ・ルーティングならびに R_c 上での部分積の計算の二つに分けて考えると理解しやすい. これら二つの動作は同時に並行して進行する. まず R_1, R_2, R_3, R_4 が関与するデータ・ルーティングについて説明する.

データ・ルーティング

● **バッファの動作**:

$t=0$ のとき: $R_3^0(\text{buffer})=(a_0, b_0)$, $R_4^0(\text{buffer})=\phi$ と仮定する.

時刻 t ($t\geqq 1$) には: $R_3^t(\text{buffer})=(a_t, b_t)$, $R_4^t(\text{buffer})=R_3^{t-1}(\text{buffer})$ なる動作をする.

● **C_i ($i\geqq 1$) の動作**:

$t=0$ のとき: $R_j^0(i)=\phi$ $(1\leqq j\leqq 4)$.

C_i ($i\geqq 1$) 上への時刻 t ($t\geqq 1$) におけるデータの移動は, 次の規則に従う. 以下では, $i=1$ のとき $R_3^{t-1}(i-1)$ を $R_3^{t-1}(\text{buffer})$ とみなす. $R_3^{t-1}(i-1)=\phi$ のとき, C_i は時刻 t には何もしない. ↗

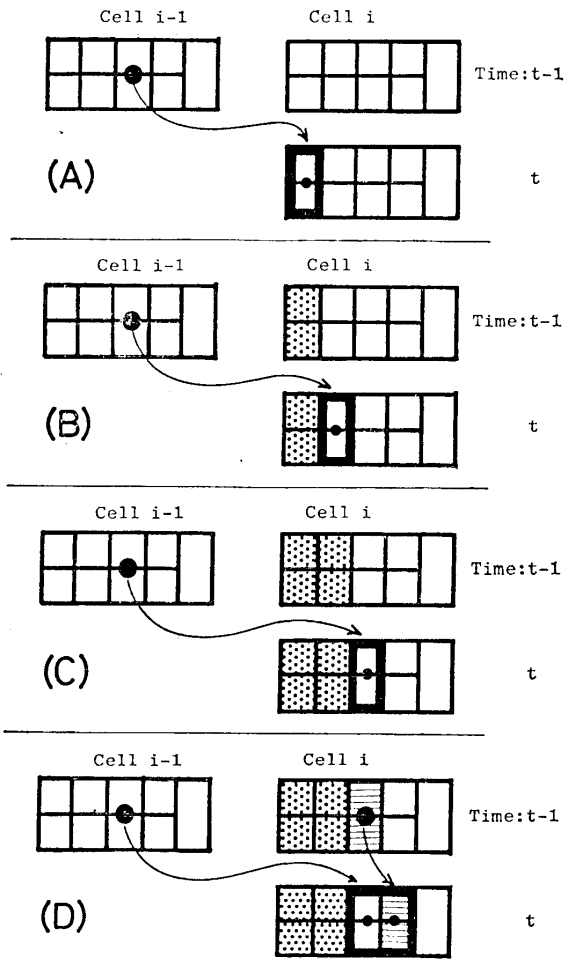


図 3 データ・ルーティング・スキーム
Fig. 3 A data routing scheme for parallel convolver.

↘これは, 静止状態におけるセルの動作を規定している. 今 $R_3^{t-1}(i-1)\neq\phi$ と仮定する. このとき, 次の動作をする (図3参照).

---

if $(R_1^{t-1}(i)=\phi)$ then $\{R_1^t(i)\leftarrow R_3^{t-1}(i-1)$, R_2, R_3, and R_4 are unchanged$\}$ (A)

else if $(R_2^{t-1}(i)=\phi)$ then $\{R_2^t(i)\leftarrow R_3^{t-1}(i-1)$, R_1, R_3, and R_4 are unchanged$\}$ (B)

else if $(R_3^{t-1}(i)=\phi)$ then $\{R_3^t(i)\leftarrow R_3^{t-1}(i-1)$, R_1, R_2, and R_4 are unchanged$\}$ (C)

else $\{R_3^t(i)\leftarrow R_3^{t-1}(i-1)$, $R_4^t(i)\leftarrow R_3^{t-1}(i)$, R_1 and R_2 are unchanged$\}$ (D)

C_i は C_{i-1} の R_3 にデータが入った次の時刻から，3ステップで(A)，(B)，(C)の三つの規則を順次実行し，以後(D)の動作を永久に続ける．

上記の規則により，次のようなルーティングがアレイ上で実行される．すなわち，バッファを経由して入力されたデータ対は，右隣接セルのレジスタ R_1, R_2 が詰まっている限り，各セルの R_3 上を右方向にスピード1，すなわち1セル/1ステップの速度で伝達する．そのデータは最初に出会った空の R_1, R_2 レジスタのうちの一つに記憶される．このとき R_1, R_2 が空いておればまず R_1 に，R_1 がすでに詰まっておれば R_2 に記憶される．ひとたび R_1, R_2 に記憶されたデータは，永久にそのレジスタから移動しない．R_4 はいつも1ステップ前の R_3 のデータをコピーしそれを一時的に記憶している．

コンボリューションの計算

部分積は各セルの R_c で計算される．アレイからの出力は $t=3$ 時から開始され，バッファの R_c に一時貯えられた後，ホスト計算機に送られる．

● **バッファの動作**：

$t=0,1,2$ 時：$R_c^t(\text{buffer})=0$ $(t=0,1,2)$

t $(\geqq 3)$ 時：$R_c^t(\text{buffer})\leftarrow R_c^{t-1}(1)$．

● **$C_i(i\geqq 1)$ の動作**：

$t=0$ 時：$R_c^0(i)=0$ $(i\geqq 1)$.

$t(\geqq 1)$ 時：各セル $C_i(i\geqq 1)$ の時刻 $t(\geqq 1)$ における動作は，時刻 $t-1$ における自身の R_1, R_2, R_3, R_4 にロードされているデータ数に応じて決定される．それぞれの動作を rule 1, 2, 3, 4 と呼び以下に示す(**図4**参照)．

○ C_i が時刻 $t-1$ にデータを含まない場合
($R_1=R_2=R_3=R_4=\phi$ のとき)
C_i は R_c に対してなにもしない．

○ C_i が時刻 $t-1$ に1対のデータを含む場合
($R_1\neq\phi$, $R_2=R_3=R_4=\phi$ のとき)
$R_c^t(i)\leftarrow A_1^{t-1}(i)B_1^{t-1}(i)$ Rule 1

○ C_i が時刻 $t-1$ に2対のデータを含む場合
($R_1\neq\phi$, $R_2\neq\phi$, $R_3=R_4=\phi$ のとき)
$R_c^t(i)\leftarrow A_1^{t-1}(i)B_2^{t-1}(i)+A_2^{t-1}(i)B_1^{t-1}(i)$ Rule 2

○ C_i が時刻 $t-1$ に3対のデータを含む場合
($R_1\neq\phi$, $R_2\neq\phi$, $R_3\neq\phi$, $R_4=\phi$ のとき)

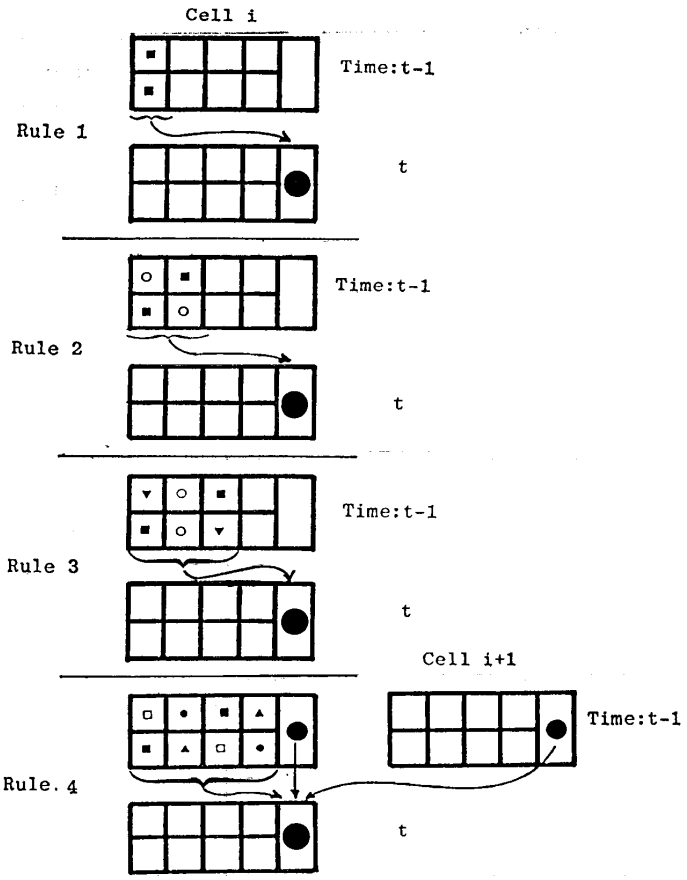


図4 コンボリューション計算
Fig. 4 Rules for the convolution computation.

$$R_c^t(i)\leftarrow A_1^{t-1}(i)B_3^{t-1}(i)+A_2^{t-1}(i)B_2^{t-1}(i)+A_3^{t-1}(i)B_1^{t-1}(i) \quad \text{Rule 3}$$

○ C_i が時刻 $t-1$ に4対のデータを含む場合
($R_1\neq\phi$, $R_2\neq\phi$, $R_3\neq\phi$, $R_4\neq\phi$ のとき)

$$R_c^t(i)\leftarrow R_c^{t-1}(i+1)+A_1^{t-1}(i)B_3^{t-1}(i)+A_2^{t-1}(i)B_4^{t-1}(i)+A_3^{t-1}(i)B_1^{t-1}(i)+A_4^{t-1}(i)B_2^{t-1}(i) \quad \text{Rule 4}$$

図5は，$n=8$, $m=4$ の場合のMによるコンボリューション計算の様子を示したものである．記号“＊”は入力記号列の終端を示す．＊記号との掛け算は0になる．@記号の付与はそれが部分積和であることを意味する．正当性の証明に関しては文献 4) を参照されたい．コンボリューション計算に必要なセル数，時間を次の定理としてまとめる．

[定理 1] 上記アレイMは $a_i(i=0,1,2,\cdots n-1)$. $b_i(i=0,1,2,\cdots,m-1)$ のコンボリューション $c_i, i=0,1,2,\cdots,n+m-1$, を実時間—すなわち，任意の i について，c_i を時刻 $t=i+3=i+0(1)$ ステップ時

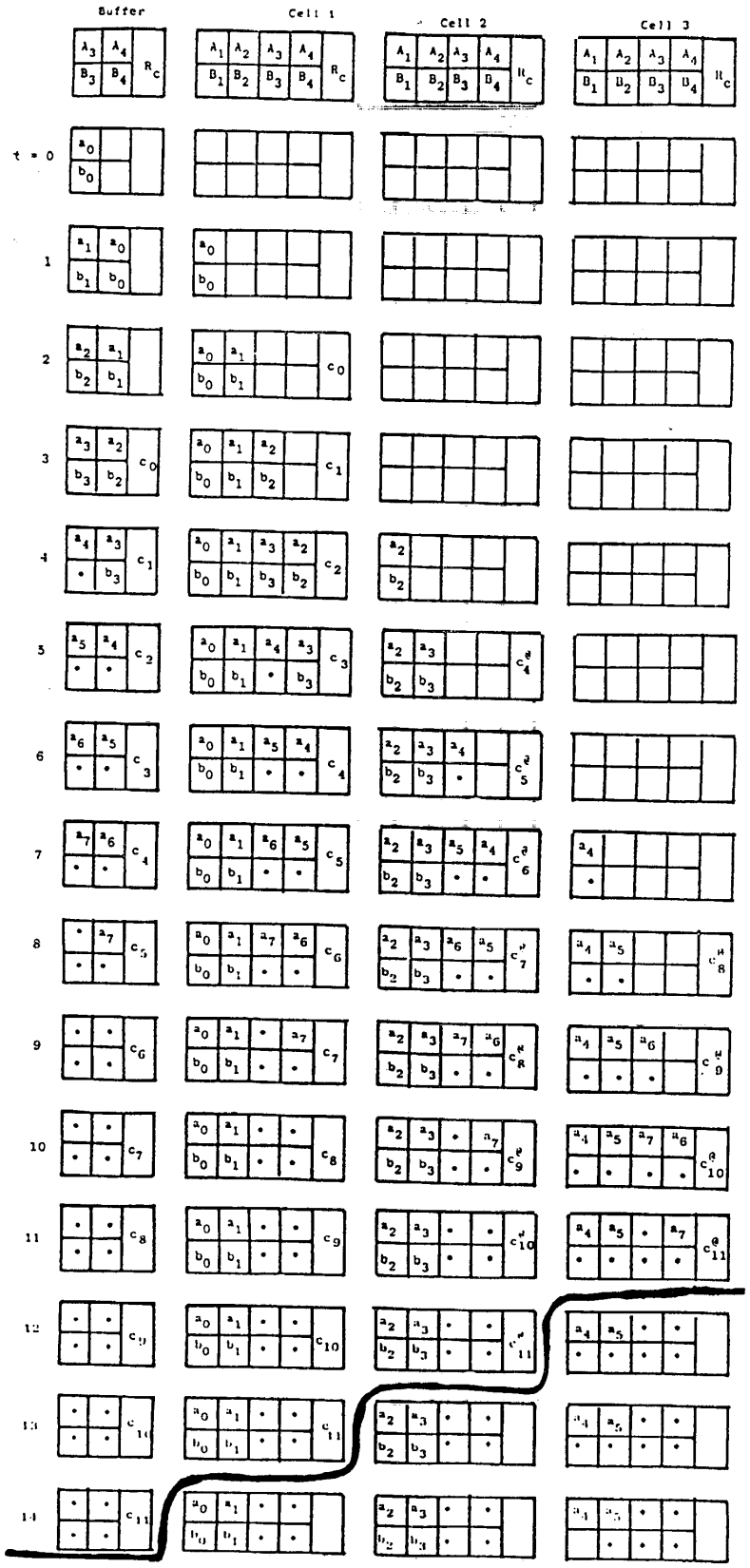


図 5 Mによるコンボリューションの計算 (n=8, m=4 の場合)
Fig. 5 A snapshot of the parallel convolver from time t=0 to 14 in the case where n=8 and m=4.

に出力する一で計算する．c_i(i=0, 1, 2, ⋯, $n+m-1$) の出力は，$n+m+0$(1)ステップの最適時間で可能で，このとき「min(m, n)/2⌉+0(1)個のセルが使用される．

3. 実時間シストリック・除算アルゴリズム

本章では，n次多項式をm次多項式で割ったときの余りと商を実時間で計算するシストリック・アレイを設計する．n, m を $n \geqq m$ なる任意の非負整数とする．次の二つの多項式 $A(x)$, $B(x)$ を考える．

$$A(x)=a_0x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n \quad (5)$$

$$B(x)=b_0x^m+b_1x^{m-1}+\cdots+b_{m-1}x+b_m \quad (6)$$

$A(x)$ を $B(x)$ で割ったときの商 $C(x)$ (たかだか $n-m$ 次式)，及び余り $D(x)$ ($m-1$ 次式) をそれぞれ，

$$C(x)=c_0x^{n-m}+c_1x^{n-m-1}+\cdots+c_{n-m-1}x+c_{n-m} \quad (7)$$

$$D(x)=d_0x^{m-1}+d_1x^{m-2}+\cdots+d_{m-2}x+d_{m-1} \quad (8)$$

とする．

$A(x)$, $B(x)$, $C(x)$, 及び $D(x)$ の間には，

$$A(x)=B(x)\cdot C(x)+D(x) \quad (9)$$

が成立する．

$\langle b_0, b_1, b_2, \cdots, b_m\rangle$ と $\langle c_0, c_1, c_2, \cdots, c_{n-m}\rangle$ のコンボリューションを z_i(i=0, 1, 2, ⋯, $n+1$) とする．

このとき，上記多項式の各係数間には次の関係式が成立する．

$$a_i=\begin{cases} z_i, & 0\leqq i\leqq n-m \\ z_i+d_{i-n+m-1}, & n-m+1\leqq i\leqq n \end{cases} \quad (10)$$

すなわち，

$a_0=b_0c_0$

$a_1=b_1c_0+b_0c_1$

$a_2=b_2c_0+b_1c_1+b_0c_2$

⋮

$$a_i=\sum_{j=0}^{i-1} b_{i-j}c_j+b_0c_i \qquad (11)$$

$$\vdots$$

$$a_{n-m}=b_{n-m}c_0+b_{n-m-1}c_1+\cdots+b_0c_{n-m}$$

$$a_{n-m+1}=b_{n-m+1}c_0+b_{n-m}c_1+\cdots+b_1c_{n-m}+d_0$$

$$\vdots$$

$$a_j=\sum_{i=0}^{n-m} b_{j-i}c_i+d_{j-n+m-1}, \qquad (12)$$

$$n-m+1\leqq j\leqq n$$

$$\vdots$$

$$a_n=b_mc_{n-m}+d_{m-1}.$$

したがって，c_i $(0\leqq i\leqq n-m)$ 及び d_i $(0\leqq i\leqq m-1)$ は，次式より求まる．

$$c_0=a_0/b_0$$

$$c_1=(a_1-b_1c_0)/b_0$$

$$c_2=(a_2-b_2c_0-b_1c_1)/b_0$$

$$\vdots$$

$$c_i=\{a_i-(b_ic_0+b_{i-1}c_1+\cdots+b_1c_{i-1})\}/b_0 \qquad (13)$$

$$0\leqq i\leqq n-m.$$

$$d_0=\left\{a_{n-m+1}-\left(\sum_{j=0}^{n-m} b_{n-m+1-j}c_j\right)\right\}$$

$$\vdots$$

$$d_i=a_{i+n-m+1}-\sum_{j=0}^{n-m} b_{i+n-m+1-j}c_j \qquad (14)$$

$$0\leqq i\leqq m-1$$

$$d_{m-1}=a_n-b_mc_{n-m}.$$

以下では n 次多項式（(5)式）を m 次多項式（(6)式）で割ったときの商（(7)式）と余り（(8)式）を，実時間で計算するシストリック・アレイAを設計する．

ホスト計算機より1対／1ステップの割合で，多項式の係数 (a_i, b_i) $(i=0, 1, 2, \cdots, n)$ が入力される．ただし，$m+1\leqq i\leqq n$ なる i に対して，$b_i=$"*" とする．出力は，$t=3$ より開始され，最初 $c_0, c_1, \cdots, c_{n-m}$ が，つづいて $d_0, d_1, \cdots, d_{m-1}$ が1個／1ステップの割合で出力される．Aは(13)及び(14)式に基づいて，c_i $(0\leqq i\leqq n-m)$ 及び d_i $(0\leqq i\leqq m-1)$ を計算する．↗

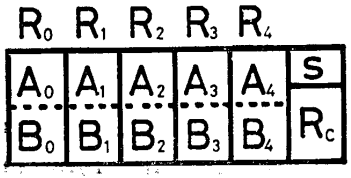


図 6 割り算アレイAの Cell 1の内部構成
Fig. 6 A systolic cell, C_1, of the polynomial divider A.

↘$\langle b_0, b_1, \cdots, b_m\rangle$ と $\langle c_0, c_1, \cdots, c_{n-m}\rangle$ のコンボリューション計算にMを利用する．したがって，定理1より計算に必要なセル数は，$\lceil \min(n-m+1, m+1)/2\rceil=\lceil \min(n-m, m)/2\rceil+0$(1)個である．

AとMの違いはセル1の設計にある．バッファ及びセル 2, 3, 4, …の仕様はMのそれと全く同一である．以下ではセル1に関する変更点について述べる．

セル1は**図6**に示すレジスタ群からなる．レジスタSは "q" あるいは "r" のいずれかの状態を取り，セル1の機能切り替えのために使用される．セル1の機能の切り替えは，$t=n-m+1$ 時に発せられるホスト計算機からの信号(バッファを経由する)によりなされる．したがってセル1のレジスタSは，時刻 $t=0$ から $t=n-m+1$ まで "q" 状態をとり，$t=n-m+2$ より $t=n+3$ の計算終了まで "r" 状態をとる．"q" 及び "r" は，それぞれセル1に商の係数かあるいは余りの係数計算のいずれかをするように指示する．A_0, B_0 はそれぞれ a_i $(0\leqq i\leqq n-1)$, b_0 の記憶用に使用される．他のレジスタはMのセル1と同様の目的で使用される．

セル1は次の動作をする．

$t=0$ 時：$A_i^0(1)=\phi$, $B_i^0(1)=\phi$ $(0\leqq i\leqq 4)$,
$S^0=$"q", $R_i^0(1)=0$.

$t=1$ 時：$A_0^1(1)=a_0$, $B_0^1(1)=b_0$, 他のレジスタは変化しない．

$t\geqq 2$ 時：セル1のSは "q" か "r" のいずれかの状態をとり，それぞれの状態で次の動作をする．

S が "q" 状態のとき：

```
while (s^{t-1}="q") do {
      A_0^t(1)←A_3^{t-1} (buffer); B_0^t(1)←B_0^{t-1}(1);
if (R_1^{t-1}(1)=φ) then {A_1^t(1)←[+]; B_1^t(1)←B_3^{t-1} (buffer); R_c^t(1)←[+]}
else if (R_2^{t-1}(1)=φ) then {A_2^t(1)←[++]; B_2^t(1)←B_3^{t-1} (buffer); R_c^t(1)←[++]}
else if (R_3^{t-1}(1)=φ) then {A_3^t(1)←[+++]; B_3^t(1)←B_3^{t-1} (buffer); R_c^t(1)←[+++]}
else if (R_4^{t-1}(1)=φ) then {A_3^t(1)←[++++]; B_3^t(1)←B_3^{t-1} (buffer); R_c^t(1)←[++++];
                               A_4^t(1)←A_3^{t-1}(1); B_4^t(1)←B_3^{t-1}(1)}
else {A_3^t(1)←[+++++]; B_3^t(1)←B_3^{t-1} (buffer); R_c^t(1)←[+++++];
      A_4^t(1)←A_3^{t-1}(1); B_4^t(1)←B_3^{t-1}(1)}
```

};

上記のプログラムで [±], [±±], [±±±], [±±±±], [±±±±±] はそれぞれ次式により決定される.

[±]: $A_0^{t-1}(1)/B_0^{t-1}(1)$.

[±±]: $(A_0^{t-1}(1)-A_1^{t-1}(1)B_1^{t-1}(1))/B_0^{t-1}(1)$.

[±±±]: $(A_1^{t-1}(1)-A_1^{t-1}(1)B_2^{t-1}(1)-A_2^{t-1}(1)B_1^{t-1}(1))/B_0^{t-1}(1)$.

[±±±±]: $(A_0^{t-1}(1)-A_1^{t-1}(1)B_3^{t-1}(1)-A_2^{t-1}(1)B_2^{t-1}(1)-A_3^{t-1}(1)B_1^{t-1}(1))/B_0^{t-1}(1)$.

[±±±±±]: $(A_0^{t-1}(1)-A_1^{t-1}(1)B_3^{t-1}(1)-A_2^{t-1}(1)B_4^{t-1}(1)-A_3^{t-1}(1)B_1^{t-1}(1)-A_4^{t-1}(1)B_2^{t-1}(1)-R_c^{t-1}(2))/B_0^{t-1}(1)$.

[±] を除く他の規則は，前節での rule 1～rule 4 に対応するものである．"q" 状態のとき，セル1は [±], [±±], ……, [±±±±±] の規則に基づいて $c_0, c_1, \cdots, c_{n-m}$ を計算する．得られた値を $R_c(1)$ に貯えるとともに，以後の計算に使用するため自身の A_1, A_2, A_3 のいずれかのレジスタに記憶する．(13)式から明らかなように，c_i を求めるのに $c_0, c_1, \cdots, c_{i-1}$ を必要とする．セル1は，まず [±] により $c_0(=a_0/b_0)$ を求め，これを $R_c(1)$ と $A_1(1)$ に貯える．$R_c(1)$ の内容は，次の時刻にバッファに送られ出力となる．$A_1(1)$ の内容は，以後 $c_1, c_2, \cdots, c_{n-m}$ を求める際に使用される．次に [±±] により，$c_1(=(a_1-b_1c_0)/b_0)$ を求め，これを $A_2(1)$ と $R_c(1)$ に貯える．$A_2(1)$ の内容も $A_1(1)$ と同様に以後の $c_2, c_3, \cdots$ の計算に使用される．以後，[±±±], [±±±±] により c_2, c_3 を求め，さらに [±±±±±] により $c_4, c_5, \cdots$ を求める．求められた値はいつも $A_3(1)$ に貯えられる．$A_3(1)$ に貯えられたデータは，Mのデータ・ルーティング規則 (A), (B), (C), (D)に基づいて，以後右側のセル 2, 3, 4, …上へと移動する．求めた係数 $c_0, c_1, c_2, \cdots$ の $A_1(1)$, $A_2(1)$, $A_3(1)$ へのデータ入力は，セル1の $t=0$ における動作開始より $t=n-m+2$ まで続けられる．

$n \geqq m$ より $n-m+2 \geqq 2$ が成立し，Sが "r" 状態をとるのは $t=2$ 時以降である．したがってセル1の "r" 状態における動作は $t=3$ 時以降である．Sが "r" 状態であるかぎり，セル1は次の動作をする．

S が "r" 状態のとき:

```
while (s^{t-1}="r") do {
      A_0^t(1)←A_3^{t-1}(buffer); B_0^t(1)←B_0^{t-1}(1);
    if (R_2^{t-1}(1)=φ) then {A_2^t(1)←"*"; B_2^t(1)←B_3^{t-1}(buffer); R_c^t(1)←[S]}
else if (R_3^{t-1}(1)=φ) then {A_3^t(1)←"*"; B_3^t(1)←B_3^{t-1}(buffer); R_c^t(1)←[SS]}
else if (R_4^{t-1}(1)=φ) then {A_3^t(1)←"*"; B_3^t(1)←B_3^{t-1}(buffer); R_c^t(1)←[SSS];
                               A_4^t(1)←A_3^{t-1}(1); B_4^t(1)←B_3^{t-1}(1)}
else {A_3^t(1)←"*"; B_3^t(1)←B_3^{t-1}(buffer); R_c^t(1)←[SSSS]; A_4^t(1)←A_3^{t-1}(1); B_4^t(1)←B_3^{t-1}(1)}
                }
```

上記のプログラム中で，[S], [SS], [SSS], [SSSS] はそれぞれ次の式である．

[S]: $A_0^{t-1}(1)-A_1^{t-1}(1)B_1^{t-1}(1)$.

[SS]: $A_0^{t-1}(1)-A_1^{t-1}(1)B_2^{t-1}(1)-A_2^{t-1}(1)B_1^{t-1}(1)$.

[SSS]: $A_0^{t-1}(1)-A_1^{t-1}(1)B_3^{t-1}(1)-A_2^{t-1}(1)B_2^{t-1}(1)-A_3^{t-1}(1)B_1^{t-1}(1)$.

[SSSS]: $A_0^{t-1}(1)-A_1^{t-1}(1)B_3^{t-1}(1)-A_2^{t-1}(1)B_4^{t-1}(1)-A_3^{t-1}(1)B_1^{t-1}(1)-A_4^{t-1}(1)B_2^{t-1}(1)-R_c^{t-1}(2)$.

$t=n-m+3$ 以降，セル1は上に示した "r" 状態における動作で，[S], …, [SSSS] により余りの係数計算をする．求めた $d_0, d_1, \cdots, d_{m-1}$ の値は，$R_c(1)$ のみに貯えられ，$A_1(1)$, $A_2(1)$, $A_3(1)$ にはかわりに "*" 記号が入力される．

図7にAによる多項式除算の様子 ($n=7$, $m=3$) を示す．

以上より次の定理を得る．

[定理 2] 上記シストリック・アレイAは，n 次多項式を m 次多項式で割ったときの商と余りを，「$\min(n-m, m)/2\rceil+0(1)$ 個のセルを使用し，$n+0(1)$ ステップの最適実時間で計算可能である．ただし n, m は $n \geqq m$ なる任意の自然数である．

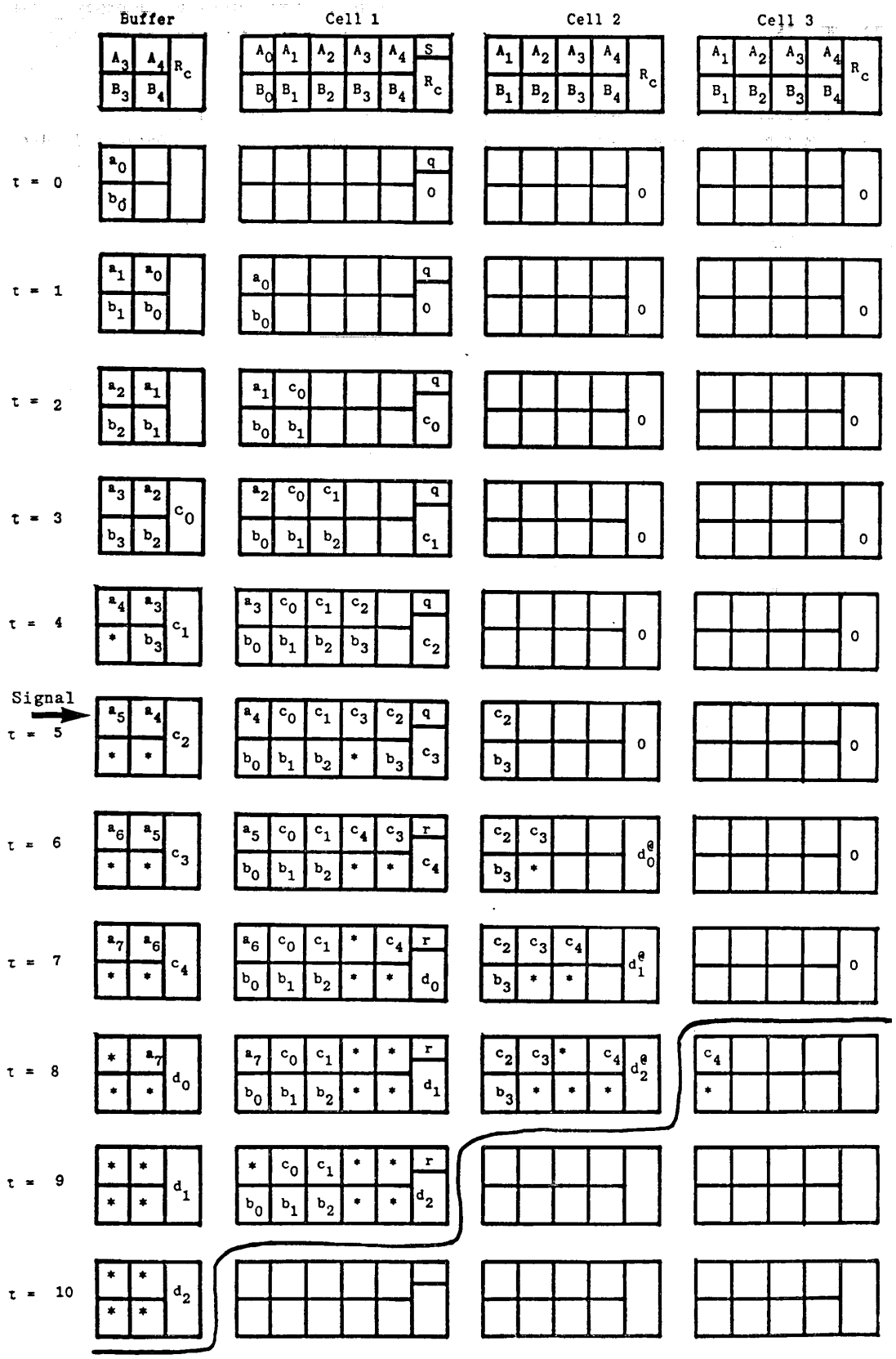


図 7 Aによる多項式除算 ($n=7$, $m=3$ の場合)
Fig. 7 A snapshot of the polynomial divider A (in the case where $n=7$ and $m=3$).

4. むすび

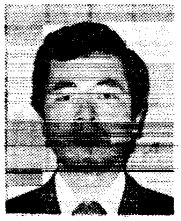
実時間で多項式除算を実行するシストリック・アルゴリズムを提案した．本アルゴリズムは，n次の多項式を，m次の多項式 ($n \geqq m$) で割ったときの商と余りを，$\lceil \min(n-m, m)/2 \rceil + 0(1)$ 個のセルを使用し，$n+0(1)$ ステップで得ることができる．このアルゴリズムはさきに文献 4)で報告された実時間パラレル・コンボルバのデータ・ルーティングを変更することにより得られるものである．本アルゴリズムは，係数まで考慮した場合の最適実時間アルゴリズムの一つであり，シストリック・アーキテクチャを仮定する限りこれ以上速くはできない．

謝辞 筆者らが多項式除算の実時間アルゴリズムを設計したきっかけは，文献 3) における和田，水野，川口の各氏らによる研究発表ならびに和田幸一博士との議論にある．ここに記して謝意を表す．

参考文献

1) Kung, H. T.: Use of VLSI in Algebraic Computation: Some Suggestions, Proc. of 1981 ACM Symp. on Symbolic and Algebraic Computation, pp. 218-222 (1981).
2) Kung, H. T.: Why Systolic Architecture?, *Comput. Mag.*, Vol. 15, No. 1, pp. 37-46 (1982).
3) 和田，水野，川口：多項式乗除算に対するシストリックアルゴリズムについて，電子通信学会技術報告，AL 84-60, pp. 97-106 (1984).
4) 梅尾：パラレル・コンボルバについて，電子通信学会技術報告，AL 84-1, pp. 1-10 (1984).
5) Ullman, J. D.: *Computational Aspects of VLSI*, p. 495, Computer Science Press, Maryland (1984).

(昭和60年7月1日受付)
(昭和60年10月17日採録)



梅尾 博司

昭和24年生．昭和48年大阪大学基礎工学部生物工学科卒業．昭和53年同大学大学院博士課程修了．工学博士．日本学術振興会研究員を経て，大阪電気通信大学に勤務．現在，同大学工学部応用電子工学科助教授．IEEE, ACM 等各会員．

杉岡 俊幸

昭和37年生．昭和60年大阪電気通信大学工学部応用電子工学科卒業．現在，神谷電子工業(株)に勤務．