解　説

3. ハードウェアアルゴリズムの設計法

3.2 ハードウェアアルゴリズムの記述と検証†

萩原兼一††　和田幸一†††

1. まえがき

集積回路の VLSI 化に伴い，設計の誤りをどのようになくすかが重要な問題である．ハードウェアの場合は実際に製造してからのフィードバックがソフトウェアに比較して非常に困難になるなどの理由により，回路が正しく設計されていることを保証することの重要性は大きい．

設計の誤りを見つける方法としては，シミュレーションによる方法[21]があるが，この方法はすべての場合を検査するのでない限り，誤りが存在しないことを保証するものではない．本稿で取り扱う VLSI の検証はシミュレーションによらないもので，望みの回路動作を形式的に記述（回路の入出力仕様）し，設計した回路がその入出力仕様を満足することを形式的に証明するものである．

ハードウェアの検証技法としては，

- テンポラル・ロジックを用いるもの[1),7),19),20)]，
- Floyd のアサーション法を用いるもの[22)]，
- イベント・表現を用いるもの[17)]，
- 形式言語を用いるもの[6),10)]

等が提案されている．これらのうち，7),17),19),20) などは，計算機による自動化を前提とした検証技法について述べられている．また，6),10)では，ハードウェアアルゴリズムをうまくモデル化し，そのモデル上で検証を行っている．しかしながら，いずれも大規模で複雑な VLSI を対象とするには今後の研究が待たれる．また，ハードウェアの検証がソフトウェアの場合と比べて難しい点は非同期回路のタイミングが正しく行われているかどうかを保証することであるが，この問題を検証の立場から取り扱ったものも多くない[19)]．なお，テンポラル・ロジックを用い，計算機による自動化を前提とした技法，及びタイミングの検証については，15),16)に解説がある．

回路の大規模化に対処するためのアルゴリズムの設計法として，Kung らによって提案されたシストリック・アルゴリズムがある[13)]．シストリック・アルゴリズムは，個々のセルが心臓の拍動のようにリズミカルに活動し，周辺の直接接続されているセルへデータを送りだしかつ受取る．そのようなセルが規則的に接続されたものである．シストリック・アルゴリズムは 1)セルの単純さ，2)データの流れの一様性，3)単純な接続関係等の利点を持ち，VLSI に適したアルゴリズムとして，多くのシストリック・アルゴリズムが提案されており，実際にチップも製造されている[2),5)]．特にシストリック・アルゴリズムでは性質 2) によって回路を同期回路とみなせるのでタイミングの問題の取り扱いも容易になる．

本稿 2 章は和田が担当し，シストリック・アルゴリズムの利点を生かしたモデルを構築し，そのモデル上でシストリック・アルゴリズムの正しさを証明する技法を紹介し，3 章は萩原が担当し，アルゴリズム・レベルでの変換法の研究に関して述べる．

2. シストリック・アルゴリズムの検証技法

ここでは，ソーティングのシストリック・アルゴリズムの例を用いて，プログラムの正当性の基礎的な手法を用いた検証技法[9)]，シストリック・アルゴリズムの抽象的モデルを用いた検証技法[18)]について述べる．これらの方法は，必ずしも計算機による自動化を前提とするものではないが，シストリック・アルゴリズムの利点を生かしたモデル化をすることによって，アルゴリズムの検証を容易にするものである．

† Verification and Transformation for Systolic Algorithms by Ken'ichi HAGIHARA (Department of Information and Computer Sciences, Osaka University) and Koichi WADA (Department of Electrical and Computer Engineering, Nagoya Institute of Technology).

†† 大阪大学基礎工学部情報工学科
††† 名古屋工業大学電気情報工学科

2.1 ソーティングの仕様とシストリック・アルゴリズム

ソーティングの仕様は次のように与えられる.

$x_1, \cdots, x_n$ を入力としたとき，ソーティングの出力 $y_1, \cdots, y_n$ は次の条件を満たす.

$y_1, \cdots, y_n$ は $x_1, \cdots, x_n$ の置換であり，各 $i(1 \leqq i \leqq n-1)$ に対して $y_i \leqq y_{(i+1)}$ が成り立つ.

このとき，各出力 y_i は次のように表されることが知られている[11]. ここで，それぞれ $\max(x, y)$, $\min(x, y)$ を $x+y$, $x \cdot y$ で表すとする.

$$y_1 = x_1 \cdot \cdots \cdot x_n$$
$$y_2 = x_1 \cdot \cdots \cdot x_{(n-2)} \cdot x_{(n-1)} + \cdots + x_2 \cdot \cdots \cdot x_n$$
$$\vdots$$
$$y_{(n-1)} = x_1 \cdot x_2 + x_1 \cdot x_3 + \cdots + x_{(n-1)} \cdot x_n$$

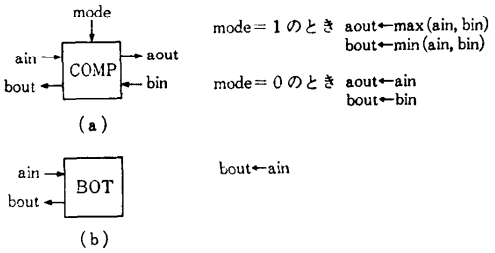


図-1 比較セルとボトムセル

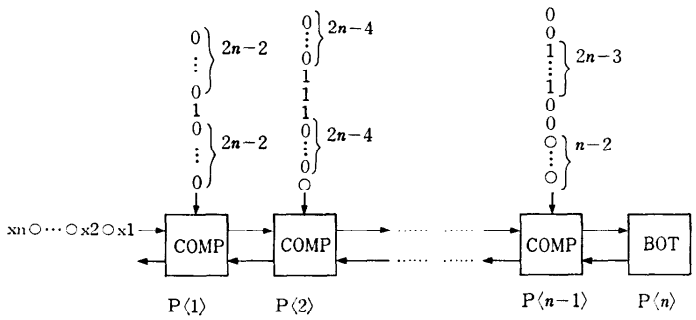


図-2 ソーティングのシストリック・アルゴリズム

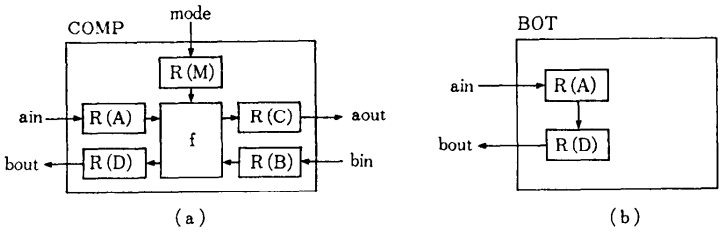


図-3 セル内の仮想的なレジスタ

$$y_n = x_1 + x_2 + \cdots + x_n$$

すなわち，$S_i(x_1, \cdots, x_n)$ で $x_1, \cdots, x_n$ に対する i 次の基本対称式を表すと，各 $i(1 \leqq i \leqq n)$ に対して，

$$y_i = S_{n-i+1}(x_1, \cdots, x_n)$$

が成り立つ.

ここで，検証の対象とするのは，n 個のデータのソーティングを**図-1**(a)の比較セル $n-1$ 個と**図-1**(b)のボトムセル1個を，**図-2** のように一次元に接続して行うシストリック・アルゴリズムである. このシストリック・アルゴリズムは，Chin らが提案しているユニフォーム・ラダー (uniform laddar) を用いたパラレル・バブル・ソート[4]をプロセッサ・ネットワークで実現したものである. また，Chen らが提案したリバウンド・ソート[3]と本質的に同じである.

2.2 プログラムの正当性証明の手法を用いた検証法[9]

2.2.1 形式的記述

まず，図-2 のシストリック・アルゴリズムの入出力仕様，及び，セルの動作を形式的に記述することを考える.

各セル内に仮想的に**図-3** のようなレジスタを設ける. **図-3**(a)の比較セルには5つのレジスタ R(A), R(B), R(C), R(D), R(M) があり，それぞれ ain, bin, aout, bout, mode の値が格納され，**図-3**(b)のボトム・セルには2つのレジスタ R(A)' R(D) があり，それぞれ ain, bout の値が格納されるとする.

図-2 において，左側から l 番目 $(1 \leqq l \leqq n)$ のセルを $P\langle l \rangle$ と表し，セルが存在する位置の集合を $L_n = \{l \mid 1 \leqq l \leqq n\}$ と表す.

時刻 t において，$P\langle l \rangle$ のレジスタ R(A), R(B), R(C), R(D), R(M) 内にあるデータをそれぞれ $A\langle t, l \rangle$, $B\langle t, l \rangle$, $C\langle t, l \rangle$, $D\langle t, l \rangle$, $M\langle t, l \rangle$ と表すと，ネットワークの初期設定，入出力，及びネットワークの動作の仕様は形式的に次のように表される. ただし，$t(i) = 2i-1$, $T = \{0, 1, 2, \cdots\}$, $t(n, l) = \{t \mid t \in T$ かつ $2n-l \leqq t \leqq 2n+l-2\}$, $t'(n, l) = \{t \mid t \in T$ かつ $(l \leqq t \leqq 2n-l-1)$ かつ $(2n+l-1 \leqq t \leqq 4n-l-2)\}$, $h(n, i) = 2n+2i-3$ とする. また，max, min を

それぞれ+，・で表す．

〔初期設定仕様〕

$A\langle 0,l\rangle = D\langle 0,l\rangle = 0 \quad (1 \leqq l \leqq n)$

$B\langle 0,l\rangle = C\langle 0,l\rangle = M\langle 0,l\rangle = 0 \quad (1 \leqq l \leqq n-1)$

〔入力仕様〕

$A\langle t(i),1\rangle = x_i \quad (1 \leqq i \leqq n)$

$M\langle t,l\rangle = 1 \quad (1 \leqq l \leqq n-1, t \in t(n,l))$

$M\langle t,l\rangle = 0 \quad (1 \leqq l \leqq n-1, t \in t'(n,l))$

〔出力仕様〕

$y_i = D\langle h(n,i),1\rangle \quad (1 \leqq i \leqq n)$

〔ネットワークの動作〕*

$1 \leqq l \leqq n-1$ に対して，

$M\langle t,l\rangle = 0 \rightarrow C\langle t,l\rangle = A\langle t,l\rangle$

$M\langle t,l\rangle = 1 \rightarrow C\langle t,l\rangle = A\langle t,l\rangle + B\langle t,l\rangle$

$M\langle t,l\rangle = 0 \rightarrow D\langle t,l\rangle = B\langle t,l\rangle$

$M\langle t,l\rangle = 1 \rightarrow D\langle t,l\rangle = A\langle t,l\rangle \cdot B\langle t,l\rangle$

$D\langle t,n\rangle = A\langle t,n\rangle$

$A\langle t+1,l+1\rangle = C\langle t,l\rangle$

$B\langle t+1,l-1\rangle = D\langle t,l\rangle \quad (2 \leqq l \leqq n)$

初期設定仕様と入力仕様を基礎にして，ネットワークの動作を用いると，数学的帰納法により次の定理が得られる．

［定理 1］ $n+1 \leqq t \leqq 3n-3$ のとき，$t=n+k(k=1, 2, \cdots, 2n-3)$ とし，$0 \leqq j \leqq k/2-1$ とするとき，

$$C\langle n+k, n-k+2j\rangle = x_{(k+1-j)} \cdot S_{n-j}(x_1, \cdots, x_{(k-j)})$$

$$D\langle n+k, n-k+2j\rangle = S_{n-j}(x_1, \cdots, x_{(k+1-j)})$$

が成り立つ．ただし，$S_p(x_1, \cdots, x_q)$ は $x_1, \cdots, x_p$ における p 次の基本対称式である．■

出力仕様と定理 1 より，図-2 のソーティングのシストリック・アルゴリズムが正しいこと，すなわち $y_i = S_{n-i+1}(x_1, \cdots, x_n)$ $(1 \leqq i \leqq n)$ が示される．

この手法はシストリック・アルゴリズムの利点であるセルの構造，データの流れが一様であることより，ネットワークの動作が比較的簡単に記述ができるので，検証も容易に行える．また，この手法は，出力が入力と演算子の式として表現できるものに適しており，同様のシストリック・アルゴリズムの検証にも容易に適用できる[9]．

2.3 抽象的シストリックモデルを用いた検証法[18]

2.3.1 シストリック・アルゴリズムの抽象的モデル

このモデルは，von Neuman のセルラ・アレイ[23]の変形であるオートマトン・ネットワークと呼ばれるモデル[8]及び，3章で述べる同期システムを等価なシストリック・システムに変換できることを証明するモデル[14]を拡張したものに相当し，前節で述べた手法をより数学的な取り扱いができるようにしたものと考えられる．なお，このモデルでは，入力の到着時刻が前もって予測できないようなシストリック・アルゴリズムは除外して考えている[18]．

N を正整数の集合，R を実数の集合とする．$\delta \notin R$ を don't care 要素と呼ぶ特別な要素とし，$R_\delta = R \cup \{\delta\}$ とする．

データ列は R_δ の要素の無限列であり，N から R_δ への写像 η で定義する．データ列の i 番目の要素を $\eta(i)$ と表し，すべてのデータ列の集合を $R_\delta^* = \{\eta \mid \eta : N \rightarrow R_\delta\}$ で表す．有界なデータ列とは，R の要素を有限個含むデータ列と定義し，すべての有界なデータ列の集合を $\bar{R}_\delta (\subset R_\delta^*)$ と表す．以降，データ列といえば，有界なデータ列をさすものとする．有界なデータ列に対して，終了関数 (ternimation function) $T: \bar{R}_\delta \rightarrow N$ を，$\eta \in \bar{R}_\delta$ に対して，$T(\eta) = i \leftrightarrow \eta(i) \neq \delta$ かつ $\eta(j) = \delta (j > i)$ と定義する．T は有界なデータ列の最後の δ でない要素の位置を与える関数である．

R 上の任意の演算子を次の 2 つの方法によって，R_δ 上に拡張する．

(1) δ を含む演算の結果は δ とする (δ-正則演算子と呼ぶ)

(2) δ を演算の結果に影響を与える特別の要素とみなす (δ-非正則演算子と呼ぶ)

例えば，δ-非正則演算子としては，$\min_\delta (\max_\delta)$ があり，次のように定義される．

$$\min_\delta(x,y)(\max_\delta(x,y)) = \begin{cases} \min(x,y)(\max(x,y)) & \text{if } x, y \neq \delta \\ y(x) & \text{if } x = \delta \text{ または } y = \delta \end{cases}$$

ここで，min(max) は R 上の通常の最小値（最大値）をきめる演算子である．

R_δ 上の演算は R_δ^* 上の演算にデータ列の要素ごとに演算を適用することにより拡張するものとする．例えば，$\eta \in R_\delta^*, w \in R$ に対して $\xi = w \cdot \eta \in R_\delta^*$ は各 i に対して $\xi(i) = w \cdot \eta(i)$ と定義される．

有界なデータ列はシストリック・ネットワークの各セルの動作を記述するために用いられるが記述のための演算子として，シフト演算子 $\Omega^k : \bar{R}_\delta \rightarrow \bar{R}_\delta$ $(k \geqq 1)$* と展開演算子 (spread operator) $\theta^r : \bar{R}_\delta \rightarrow \bar{R}_\delta$ $(r \geqq 1)$** を

---

* 導線をデータが伝わるのに 1 単位時間かかるとし，セルの計算時間はそれに含めて考える．

* $k=1$ の場合，Ω^1 を Ω と略記する．

** $r=1$ の場合，θ^1 を θ と略記する．

を次のように定義する．シフト演算子 Ω^k はデータ列の前に k 個の δ を挿入するものであり，展開演算子 θ^r はデータ列の各要素の間にそれぞれ r 個の δ を挿入するものである．例えば，$\xi=a_1, a_2, a_3, a_4, \delta, \delta, \cdots$ ならば，

$\Omega^3\xi=\delta, \delta, \delta, a_1, a_2, a_3, a_4, \delta, \delta, \cdots$

$\theta^2\xi=a_1, \delta, \delta, a_2, \delta, \delta, a_3, \delta, \delta, a_4, \delta, \delta, \cdots$

となる．

データ列上の n 項演算子 $\Gamma: [\bar{R}_\delta]^n \to \bar{R}_\delta$* が（弱）因果演算子（(weakly) causal operator）であるとは，値域のデータ列の i 番目の値は，定義域の各データ列の最初の $i-1(i)$ 番目の値のみに依存することである．定義より，因果演算子 Γ^1 と弱因果演算子 Γ^2 を結合した演算子 $\Gamma^1\Gamma^2$（または $\Gamma^2\Gamma^1$）は因果演算子となる．また，シフト演算子 Ω^k は因果演算子であり，展開演算子 θ^r は弱因果演算子であり，$\Omega^k\theta^r$ は因果演算子である．

抽象的シストリック・モデルとは，次のものからなる．

[A1] 自己ループを含まない多重グラフ $G[V, E, \phi_-, \phi_+]$ はシストリック・アルゴリズムのネットワークを指定する．V は節点の集合，E は有向辺の集合，$\phi_-(\phi_+): E \to V$ は任意の $e\in E$ に対して，$\phi_-(e)\neq\phi_+(e)$ を満たす写像である．$\phi_-(e)(\phi_+(e))$ は e の始点(終点)を表す．各 $v\in V$ に対して $\{e|\phi_-(e)=v\}$（$\{e|\phi_+(e)=v\}$）を v の OUT(IN) 有向辺と呼ぶ．入次数 0 の節点をソース，出次数 0 の節点をシンク，その他の節点を内部節点と呼ぶ．内部節点，ソース，シンクはそれぞれ，ネットワークにおける計算セル，入力セル，出力セルに対応する．

[A2] 各有向辺にはラベルがつけられているものとする．ここで，各節点の IN(OUT) 有向辺は同じラベルを持たないと仮定する．

[A3] 各有向辺 $e\in E$ に対して，有界なデータ列 $\xi_e(\in\bar{R}_\delta)$ が指定される．

有向辺はセル間を接続する導線に対応し，データ列 ξ_e は各時刻にその導線に現れるデータを表している．

[A4] 入次数 m，出次数 n の内部節点 $v\in V$ に対して，n 個の m 項因果演算子 $\Gamma_v^i: [\bar{R}_\delta]^m \to \bar{R}_\delta (1\leqq i\leqq n)$ が与えられる（これを節点の I/O 記述と呼ぶ）．

これは各セルが実行する計算に対応している．時刻 t において，各セルで実行される計算は t 以前のデータにのみ依存するので，各セルには，因果演算子を対応させている．

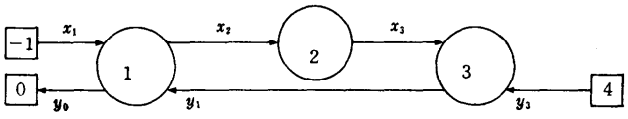


図-4 シストリック・ネットワークの例

シンクの IN 有向辺に指定されたデータ列をネットワーク出力列，ソースの OUT 有向辺に指定されたデータ列をネットワーク入力列と呼ぶ．

[例 1] 図-4 のグラフをもつネットワークを考える．図-4 において，内部節点は丸で，ソース，シンクは四角で表してある．各節点及び有向辺は図のようにラベルがつけられているとする．x_i, y_j に対応するデータ列をそれぞれ ξ_i, η_i で表す．各節点に以下の因果演算子が与えられているとする．

節点 1: $\xi_2=\Omega[\xi_1+\eta_1]$ (1)

$\eta_0=\Omega[\xi_1*\eta_1]$ (2)

節点 2: $\xi_3=\Omega\xi_2$ (3)

節点 3: $\eta_1=\Omega[\xi_3*\eta_3]$ (4)

このネットワークにおいては，η_3 と ξ_1 がネットワーク入力列であり，η_0 がネットワーク出力列である．(1)〜(4)を解いて η_0 を η_3 と ξ_1 で表現したものをネットワークの I/O 記述と呼ぶ．■

例 1 からわかるように，このモデルにおけるシストリック・アルゴリズムの検証は入力仕様をデータ列で表現したとき（ネットワーク入力列），内部節点の因果演算子で定義された節点の I/O 記述を因果演算子の性質などを用いて，ネットワーク出力列について解き，ネットワーク入力列で直接表現したものが出力仕様を満足することを示せばよい．図-2 のソーティングのシストリック・アルゴリズムをこの手法により実際に検証する．

2.3.2 ソーティングの例への適用

n 個の入力の集合を $X=\{x_1, \cdots, x_n\}$ $(x_i\in R)$ と表し，n 個の要素はすべて異なるものとする．$N=\{1, \cdots, n\}$，$D=\{(i, j)\in N\times N | 1\leqq i\leqq j\leqq n\}$ とし，順位関数 $f_x: D\to X$ を $f_x(i, j)$ が $x_1, \cdots x_j$ の中で i 番目に小さい要素であると定義する．すなわち，$fx(i, j)=S_{j-i+1}(x_1, \cdots, x_j)$ である．

図-2 のネットワークに対応するグラフが図-5 のようにラベルづけされているとする．また，p_i, s_j に対応するデータ列を π_i, σ_j で表すとすると，ネットワーク入力列 π_n（入力仕様）と出力仕様は次のよう

* $[\bar{R}_\delta]^n$ は $\bar{R}_\delta$ の n 個の直積を表す．

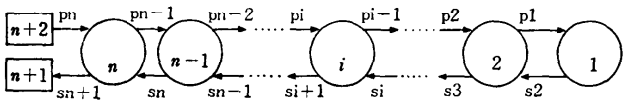
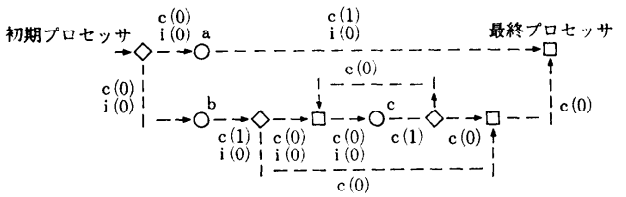


図-5　図-2のシストリック・ネットワークに対応するグラフ



◇　分岐プロセッサ
○　文字×の認識プロセッサ
□　融合プロセッサ

図-6　正規表現 $a+bc^*$ の表す正規集合の認識機械 Mのタイミング・ダイアグラム G

に与えられる.

[入力仕様]　ξ を $T(\xi)=n$, $\xi(t)=x_t(1\leqq t\leqq n)$ とするとき, $\pi_n=\theta\xi$.

[出力仕様]　η を $T(\eta)=n$, $\eta(t)=f_z(t,n)(1\leqq t\leqq n)$ とするとき, $\sigma_{n+1}=\Omega^{2n-1}\theta\eta$.

各セルの動作は次のように表される.

$$\sigma_2=\Omega\pi_1 \qquad (5)$$

$$\pi_{i-1}=\Omega\max_\delta(\pi_i,\sigma_i)\quad(2\leqq i\leqq n) \qquad (6)$$

$$\sigma_{i-1}=\Omega\min_\delta(\pi_i,\sigma_i)\quad(2\leqq i\leqq n) \qquad (7)$$

ここで, $\max_\delta(\min_\delta)$ は前節で定義した δ-非正則演算子である. 図-2 の各比較セルは制御信号 mode によって比較を行うか単にデータを伝達するかを選択しているが, ここでは $\max_\delta(\min_\delta)$ という δ-非正則演算子でその選択を表現している. すなわち, $\sigma_i(t)=\delta$ または $\pi_i(t)=\delta$ ならば, $\pi_{i+1}(t+1)=\pi_i(t)$, $\sigma_{i+1}(t+1)=\sigma_i(t)$ となり, それ以外は $\pi_i(t+1)=\max(\pi_i(t),\ \sigma_i(t))$, $\sigma_{i+1}(t+1)=\min(\pi_i(t),\ \sigma_i(t))$ となる.

入力仕様と(5)～(7)及び因果演算子 Ω, θ の性質を用いると π_i, σ_i は次のように表される.

[定理 2]　α_i, β_i を $T(\alpha_i)=T(\beta_i)=n$,

$$\alpha_i(t)=\begin{cases} x_t & (1\leqq t\leqq i)\\ \max\{x_t, f_z(t-i,t-1)\} & (i<t\leqq n)\end{cases}$$

$$\beta_i(t)=\begin{cases} f_z(t,t+i-2) & (1\leqq t\leqq n+1-i)\\ f_z(t,n) & (n+1-i<t\leqq n)\end{cases}$$

とするとき,

$$\pi_i=\Omega^{n-i}\theta\alpha_i\quad(1\leqq i\leqq n)$$

$$\sigma_i=\Omega^{n+i-2}\theta\beta_i\quad(2\leqq i\leqq n+1)$$

が成り立つ. ■

定理2の σ_i で $i=n+1$ とおくと, 出力仕様が満たされる.

シストリック・アルゴリズムの場合, 一種類（あるいは数種類）の機能を持つセルの繰り返しでネットワークが構成される場合が多い. この抽象モデルに基づく検証技法の利点は, ネットワークを表すグラフの内部節点が同一の I/O 記述をもつ（このときネットワークは homogeneous という）[18] 場合は, I/O 記述が単純な漸化式として表現され, ネットワーク出力列が自動的に計算できるということである. 例えば, I/O 記述が $\sigma_{i+1}=\Omega\sigma_i+\Delta_i(i=1,2,\cdots,k+1)$ の場合, その解は $\sigma_r=\Omega^{r-1}\sigma_1+\sum_{j=1}^{r-1}\Omega^{j-1}\Delta_{r-j}(r=2,3,\cdots,k+1)$ で与えられる. これを利用すれば, 一次元のネットワークで実現されたたたみ込みのシストリック・アルゴリズムなどは, 検証が容易に行える[18]. 複雑なシストリック・アルゴリズムの検証のために I/O 記述で表現される漸化式を解くいろいろな手法の研究が, このモデルを用いて検証するための今後の課題であろう.

3. ハードウェアアルゴリズムの変換

検証とは少し立場が違うが, 最終的に正しく動くハードウェアを作成する方法として, 設計者が比較的簡単に正しいと確信できる仕様（仮想ハードウェアアルゴリズム）から, ある変換規則により実現可能なハードウェアアルゴリズムを導出するものがある[12),14),24)]. この変換規則が正しいことだけを予め確認しておけば, 後は正しい仮想ハードウェアアルゴリズムを作成することだけに注意を払えばよい. これは, ソフトウェアの世界でのプログラム変換に相当するもので, 今後重要性が増すと思われるが, 変換対象の広さや変換後のハードウェアアルゴリズムの性能のよさなど, 解決されるべき問題は山積されている. 本章では, 例を用いて文献 14) の内容を中心に紹介する.

3.1 タイミング・ダイアグラムの例

[例 2]　正規表現 $a+bc^*$ が表す記号列の集合の認識機械 M を考える. M のタイミング・ダイアグラムと呼ばれるグラフ G を図-6 に示す. ここで, グラフの節点はプロセッサを, 節点 n から節点 m へのラベル $X(d)$ が付けられた弧は, プロセッサ n からプロセッサ m へデータ・ストリーム X が遅延 d（ただし,

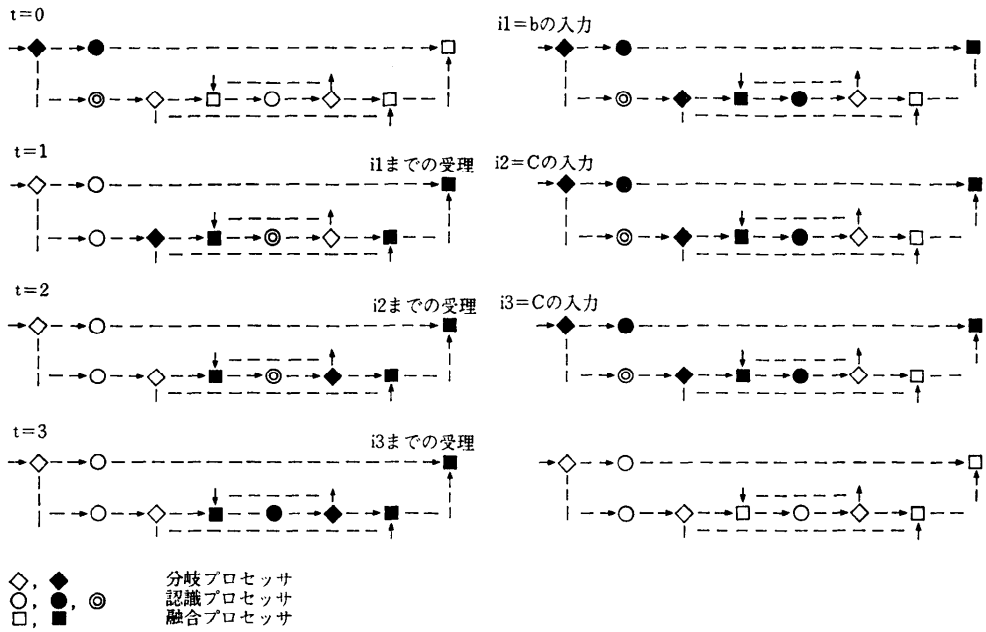


図-7 Mに入力 "bcc" を印加したときの状態遷移（非シストリックな場合 入力周期は1）（左側は制御ストリームの，右側は入力ストリームの流れを示す）

$d \geqq 0$) で流れることを表す.

プロセッサは，認識プロセッサ，分岐プロセッサ，融合プロセッサの3種類ある．認識プロセッサ r は，r に入力されるデータ i を識別し，i の値により後続プロセッサへ（i の値と）制御情報を伝える．分岐プロセッサ f は，f に入力されたデータ i や制御信号 c を各後続プロセッサにコピーして伝える．融合プロセッサ j は，複数個の先行プロセッサからの信号を合流させて，j の後続プロセッサに伝える．

図-6 では，2種類の信号ストリームがある．1つは制御ストリーム c で，各プロセッサに意味のある動作をすべきか否かを伝える．分岐プロセッサ及び融合プロセッサは，各後続プロセッサへ遅延0で信号 c を伝えている．認識プロセッサは遅延1で c を（必要ならば c の値を変化させて）伝える．例えば，入力データ i が受理すべきものでない場合，$c=0$ として後続プロセッサに伝えることにより，後続プロセッサの受理判定活動を止めるように指令する．受理すべきものであれば，c の値をそのまま伝える．もう1つの信号は，入力 i である．入力は，初期プロセッサに印加されるものとし，図-6 の場合は，初期プロセッサから遅延0で他の認識プロセッサへ伝えられている．

［例3］ M の初期プロセッサに，入力 "bcc" を入力周期1で印加したときの，各時刻のストリームの様子を図-7 に示す．各時刻 $i(i=1,2,3)$ に，最終プロセッサは，入力系列の先頭から第 i 番目までの文字列を，受理したか否かの結果を出力している．

3.2 非シストリック・アルゴリズムからシストリック・アルゴリズムへの変換

タイミング・ダイアグラムの記述において，図-6 のように例えば遅延0での信号伝播を許すならば，そのもののハードウェアとしての実現可能性はともかく，所望の仕様通りに動作することを確信しやすい仮想ハードウェアを設計できる．しかし，全く自由に0遅延の弧を書くことを許すと，例えば1つのプロセッサから遅延0で任意複数個のプロセッサへ情報を伝えられることになり，これを実際のハードウェアとして実現するのは容易ではない．したがって，0遅延が無いようにすることを考える．

タイミング・ダイアグラム G のすべての遅延が1以上のとき G（の表すアルゴリズム）は "シストリック" であるという．

ここでは，非シストリック（遅延0の弧を持つ）ではあるが，遅延0の弧だけから成るサイクルが存在し

ないタイミング・ダイアグラムGから，アルゴリズムの性能は変わる（最初の出力時刻がある時間遅れ，入力及び出力の周期は，定数倍される（このことを，“スローダウン”と言う））ことがあるが，機能は同じ（スローダウンを補正して考えて，同じ入力系列に対して同じ出力系列が出る）でかつシストリックなタイミング・ダイアグラムG'を得る機械的な変換法を述べる．

この変換法では，次の2つの基本的なアイデアを用いる．

［アイデア 1］ 節点vにはm個の入力弧とn個の出力弧があるとする．各入力弧の遅延を$d_i(1\leqq i\leqq m)$，各出力弧の遅延を$d_i'(1\leqq i\leqq n)$とする．このとき，次の①または②の操作をしても機能は変わらない．

① 任意の$i(1\leqq i\leqq n)$に関して，$d<d_i'$となる任意の正数をdとする．このとき，各入力弧の遅延d_iをd_i+dに，各出力弧の遅延d_i'を$d_i'-d$にする．

② 任意の$i(1\leqq i\leqq m)$に関して，$d<d_i$となる任意の正数をdとする．このとき，各入力弧の遅延d_iをd_i-dに，各出力弧の遅延d_i'を$d_i'+d$にする．

［アイデア 2］ タイミング・ダイアグラムのすべての遅延に任意の正数kを掛けても，機能は変わらない．ただし，アルゴリズムの速さは$1/k$になる（これをk倍のスローダウンという）．

［記法］ タイミング・ダイアグラムGのすべての遅延を1だけ減らしたタイミング・ダイアグラムをG-1と書く．G-1では，負の遅延が存在し得る．

［例 4］ 図-8に，図-6のGに対するタイミング・ダイアグラム G-1を示す．

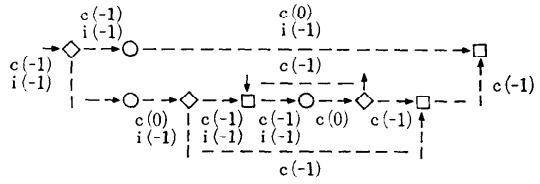


図-8 タイミング・ダイアグラム G-1

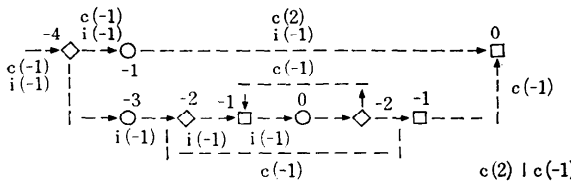


図-9 Gの lag の値と $3G$-1 のタイミング・ダイアグラム（節点近くの数字は，その節点の lag の値を示す）

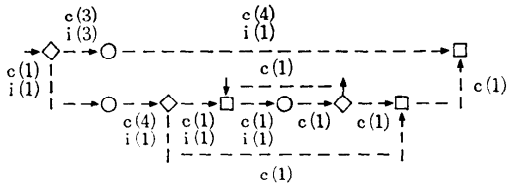


図-10 方法2でGをシストリックにしたときのタイミング・ダイアグラム G'

［定理 3］ G-1に遅延の和が負となるサイクルが存在しなければ，Gをシストリックにできる．■

そのやり方を，次の方法1に示す．方法1では，［アイデア 1］のみを用いている．

［定義］ 外部との入出力を行うプロセッサを“ホスト”と言う．あるホストを1つ固定する．G-1において，各節点uからホストへ行く道（path）の遅延の和のうち，最小な値を lag(u) と定義する．

［例 5］ 図-6 の G では，最終節点をホストと考える．この場合の lag の値を，図-9 に示す．

〈方法 1〉 Gの各弧 (u, v) の遅延dを$d+\text{lag}(v)-\text{lag}(u)$にする．■

［定理 4］ G-1に遅延の和が負となるサイクルが存在すれば，［アイデア 1］を用いるだけでは，Gをシストリックにできない．■

（注意） この例(図-8)では，遅延の和が負となるサイクルが存在するので，方法1ではシストリックにできない．

定理4の内容は，［アイデア 1］に拘らず，もう少し強いことが言える．

［定理 4′］ G-1 に遅延の和が負となるサイクルが存在すれば，入(出)力周期を変えないで，Gをシストリックにはできない．■

［定理 5］ G-1に遅延の和が負となるサイクルが存在しても，Gに遅延の和が0のサイクルが存在しなければ，［アイデア 1］と［アイデア 2］を用いて，Gをシストリックにできる．■

そのやり方を次の方法2に示す．

［記法］ タイミング・ダイアグラムGのすべての遅延をk倍したタイミング・ダイアグラムを kG と書く．

〈方法 2〉 ①kG-1 に遅延の和が負となるサイクルが存在し

ないような，最小のkの値をhとする．②hGを新たなGとして，〈方法1〉を行う．■

〈方法2〉で得られたタイミング・ダイアグラムは，入(出)力周期がhである．hの最小性と［定理4′］より，機能がGと同じで，入(出)力周期がh未満であるようなシストリックなタイミング・ダイアグラムは存在しない．

［例6］ 図-6のタイミング・ダイアグラムを，方

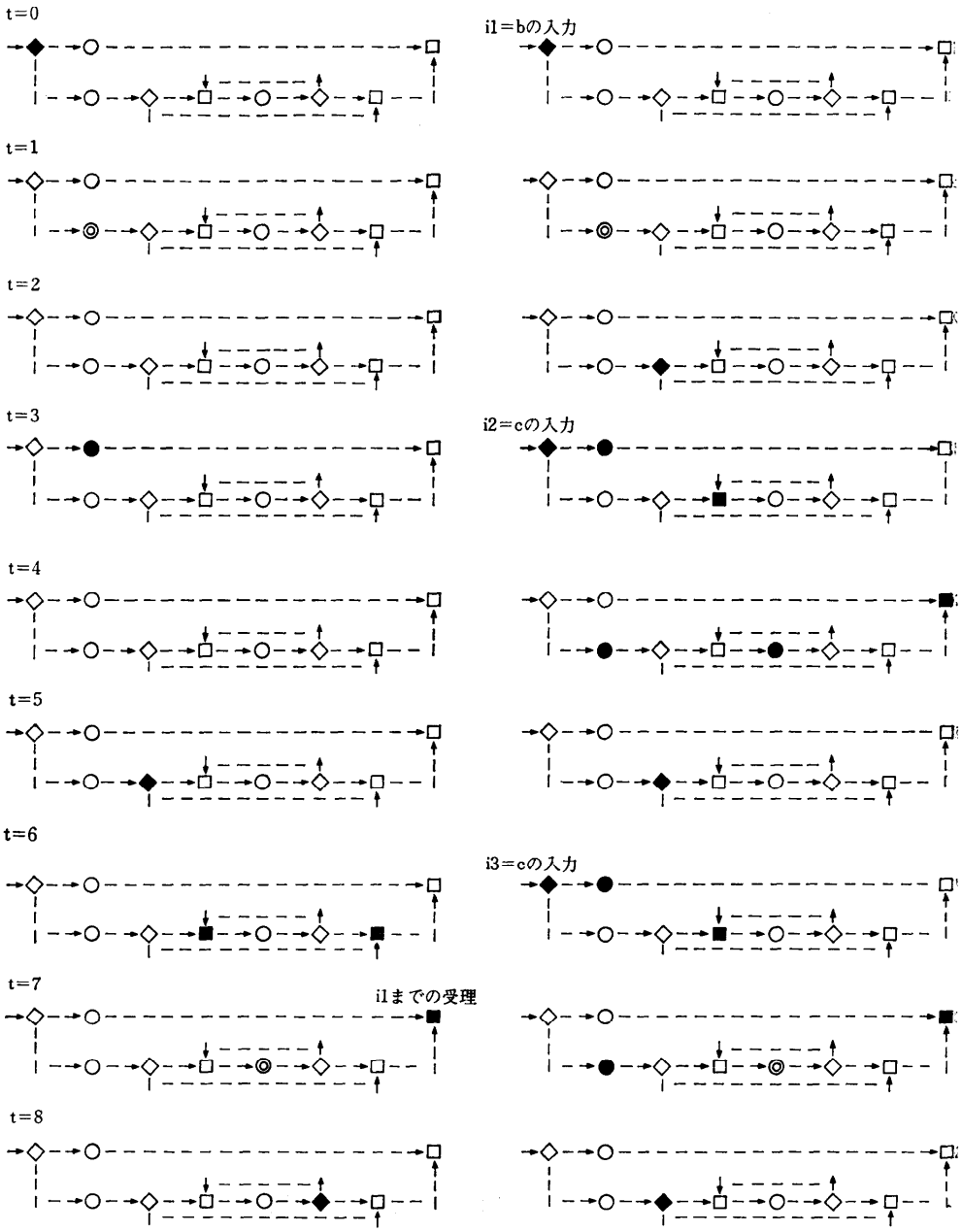


図-11 G'の表す認識機械に入力"bcc"を印加したときの状態遷移（シストリックな場合 入力周期は3．ただし，時刻8まで）

法2でシストリックに変換した結果を，図-10 に示す．この場合，$h=3$ (図-9 に $3G$-1 のタイミング・ダイアグラムを示す) である．また，これに，入力 "bcc" を印加したときの，各時刻の様子を図-11 に示す．最初の出力が，時刻7となり，入(出)力周期が3となっている．この例の場合，入出力周期が3未満のシストリック化は不可能である．

3.3 その他の研究

ハードウェアアルゴリズムの記述は，よく "グラフ" や "ネットワーク" で行われる．節点がプロセッサを，弧がワイヤを示す．このような表現法は，ハードウェアアルゴリズムの仕様を記述するには適当かもしれないが，アルゴリズムの設計そのものを操作するのには向いていないという立場から，代数的表現 (algebraic representation) と呼ばれるハードウェアアルゴリズムの記述法を文献 24), 12) は提案している．そして，線形代数でよく使われるのと類似の代数的変換 (algebraic transformation) を用いて，あるハードウェアアルゴリズムの設計から，データの転送に関する望ましい性質 (局所性や規則性) を持ち，かつ "機能が同じ" ハードウェアアルゴリズムの設計を導出できることを示している．

4. むすび

本稿では，VLSI 向けハードウェアアルゴリズムの正しさを保証するという立場から，シストリック・アルゴリズムの検証技法とシストリック・アルゴリズムへの変換法について述べた．今後，ますます大規模化する VLSI に対処するためのハードウェアアルゴリズムの検証技法に関する研究が待たれる．

参考文献

1) Bochmann : Hardware Specification with Temporal Logic : Example, IEEE Trans. on Comput., Vol. C-31, No. 3, pp. 223-231 (1982).
2) Brent, R. P. and Kung, H. T. : Systolic VLSI Array for Polynomial GCD Computation, IEEE Trans. on Comput., Vol. C-33, No. 8, pp. 731-736 (1984).
3) Chen, T. C., Lum, V. Y. and Tung, C. : The Rebound Sorter : an Efficient Sort Engine for Large Files, Proc. of the 4th Int. Conf. on Very Large Database (1979).
4) Chin, F. Y. and Fok, K. S. : Fast Sorting Algorithms on Uniform Ladders (Multiple Shift-register Loops), IEEE Trans. on Comput., Vol. C-29, No. 7, pp. 618-631 (1980).
5) Fisher, A. L., Kung, H. T., Monier, L. M., Walker, H. and Dohi, Y. : Design of the PSC : A Programmable Systolic Chip, Proc. of 3rd Caltech Conf. on VLSI. pp. 282-302 (1983).
6) Foster, M. I. : Syntax-Directed Verification of Circuit Function, VLSI SYSTEMS AND COMPUTATIONS. pp. 196-202 (1981).
7) 藤田，田中，元岡：時相論理によるハードウェア仕様記述と Prolog を用いたゲート回路の検証，情報処理学会論文誌，Vol. 25, No. 2, pp. 173-179 (1984).
8) Grefenstette, I. : Automaton Networks and Parallel Rewriting Systems, Ph. D. dissertation, Dept. of Comput. Scie., Univ. of Pittsburgh, Pittsburgh, PA. (1980).
9) 萩原，増沢，都倉：VLSI アルゴリズムの検証，電気関係学会関西支部連大，G 6-5 (1981).
10) 木村，安浦，矢島：入力制約を用いた論理回路の形式的検証について，信学技報，AL 83-63 (1984).
11) Knuth, D. E. : The Art of Computer Programming Vol. 3 : Sorting and Searching, Addison-Wesley (1973).
12) Kung, H. T. and Lin, W. T. : An Algebra for VLSI Algorithm Design, CMU-CS-84-100. Carnegie-Mellon Univ. (1983).
13) Kung, H. T. : Why Systolic Architecture, Comput. Mag., 15, pp. 37-46 (1982).
14) Leiserson, C. E. and Saxe, J. B. : Optimizing Synchronous Systems, CMU-CS-82-101. Carnegie-Mellon Univ. (1982).
15) 丸山，上原：方式・機能・論理設計の検証，情報処理，Vol. 25, No. 10, pp. 1062-1070 (1984).
16) 松下浩明：タイミング検証，情報処理，Vol. 25, No. 10, pp. 1056-1061 (1984).
17) McFarland, M. C. and Parker, A. C. : An Abstract Model of Behavior for Hardware Descriptions, IEEE Trans. on Comput., Vol. C-32, No. 7, pp. 621-637 (1983).
18) Melhem, R. G. and Rheinbold, W. C. : A Mathematical Model for the Verification of Systolic Network, SIAM J. Comput., Vol. 13, No. 3, pp. 541-565 (1984).
19) Mishra, B. and Clarke, E. M. : Automatic and Hierarchical Verification of Asynchronous Circuits Using Temporal Logic, CMU-CS-83-155 (1983).
20) Moszkowski, B. : A Temporal Logic for Multi-Level Reasoning about Hardware, Tech. Report STAN-CS-82-953, Dept. of Comput. Scie., Stanford Univ. (1982).
21) 村上，平川：方式・機能・論理シミュレーション，情報処理，Vol. 25, No. 10, pp. 1048-1055 (1984).
22) Pitchumani, V. and Stabler, E. P. : An Induc-

tive Assertion Method for Register Transfer Level Design Verification, IEEE Trans. on Comput., Vol. C-32, No. 12, pp. 1073-1080 (1983).
23) von Neumann, I.: Theory of Self Reproducting Automata, Univ. of Illinois Press, Urbara, IL (1966).
24) Weiser, U. and Davis, A.: A Wavefront Notation Tool for VLSI Array Design, in Kung, H. T., Sproull, R. F. and Steele, G. L., Jr. (editors). VLSI Systems and Computations, pp. 226-234. Computer Science Press (1981).

(昭和60年5月7日受付)