単項演算に対する局所計算可能な符号化†

安浦 寛人††

並列アルゴリズムの設計において，符号化は重要な役割を担っている．ある符号化において，実現したい演算の結果の符号の各桁が，オペランドの符号の一部の桁だけの情報から計算できるとき，局所計算可能であるという．局所計算可能性と符号の冗長性の間には密接な関係があることが知られており，冗長符号化を利用した優れた並列アルゴリズムも幾つか知られている．実現の対象となる集合と演算の代数的な性質と局所計算可能性の関係を調べる問題は，並列アルゴリズムの設計に関する基本的な問題である．ここでは，複数の単項演算が定義された有限集合に対する符号化について，冗長性と局所計算可能性の基本的な関係を議論する．まず，非冗長な符号では，一般的には局所計算可能性が実現できないことを示す．さらに，任意の集合とその上の複数の単項演算に対して，結果の各桁がオペランドの2桁だけから決まるような局所計算可能な冗長な符号化を与える．これらの結果は，最近設計自動化技術の分野で話題となっている順序回路の状態割当問題と密接な関係がある．上述の局所計算可能な符号化は，任意の順序機械に対して，パイプライン化した順序回路を構成する冗長な状態割当が存在することを示している．

1. はじめに

並列アルゴリズムの設計においては，いかに計算の並列性を抽出し，並列に計算できる部分を独立に計算するかが重要な問題である．さらに，各々の独立した計算が局所的な情報のみに依存するようにできれば，個々の独立した計算は単純なものとなり，計算全体における通信量も小さくなる．このように，並列計算を局所的な通信のみを必要とする単純な計算の集まりにすることができれば，通信や計算に要する時間や資源を小さくすることができる．

このようなアイディアに基づき，局所計算可能性 (Local Computability) という概念が提案され，それを実現するために符号化の工夫が有効であることが示された[1]．集合Sとその上での演算fが与えられたとき，ある符号化Cのもとで，fによる演算結果の各桁が，オペランドの高々ℓ桁にのみ依存するような計算規則が作れるとき，演算fは符号化Cのもとでℓ-局所計算可能であるという．(S, f)が有限可換群であれば，Sの要素数にかかわらず$\ell=28$となる2値の等長符号化が存在する[1]．この符号化はSの各要素に2つ以上の符号を割り当てる冗長符号化であり，この冗長性が本質的に，計算の局所化に寄与している．実用的にも算術演算における SD 表現[2)~4)]や冗長2進表現[5),6)]の利用は，局所計算可能性を実現する符号化の応用であると考えられる．

† Locally Computable Coding for Unary Operations by HIROTO YASUURA (Department of Electronics, Faculty of Engineering, Kyoto University).

†† 京都大学工学部電子工学科

一般に符号の冗長度と局所計算可能性の間には，なんらかのトレードオフがあると予想される．また，対象となる集合の代数的な構造と局所計算可能性との関係は，数学的にも並列アルゴリズムの設計の立場からも，興味深い問題である．本論文では，最も単純で基本的な代数構造である複数の単項演算が定義された有限集合について，符号の冗長性と局所計算可能性の関係について考察する．

この問題は，最近論理回路の CAD の分野で活発に議論されている順序回路の自動合成における状態割当の問題[7),8)]と密接に関係している．すなわち，順序機械の状態集合を対象とする集合として，各入力による次状態への遷移を単項演算と考えれば，符号化はまさに状態割当である．局所計算可能な符号化は，回路の規模や段階（計算速度に対応する）が小さな順序回路を与える状態割当となる．回路の性能を向上させる手法としてパイプライン処理がしばしば用いられるが，これは，回路中の各記憶素子の次の時刻の値を近傍の記憶素子の値だけから計算できるようにするもので，局所計算可能な符号化を用いた状態割当の有力な実例である．すなわち，局所計算可能な状態割当は，順序回路のパイプライン化の一般化と考えることもできる．このように，単項演算が定義された有限集合の符号化問題は，実用的にも重要な問題である．

本論文では，まず，符号の冗長性と局所計算可能性の関係を議論するために，種々の概念を定義する．符号の冗長性として，集合の各要素に1つずつ符号を割り当てる単一符号化と，複数の符号を割り当てる多重符号化を考える．単一符号化でかつ最短なものを非冗

長符号化と呼ぶ．単一符号化では使用しない符号（非符号語）を導入することで冗長性が実現される．多重符号化は，より一般的な冗長性を含む．まず，単項演算についての局所計算可能性に対するこれらの符号化の能力を明らかにする．最も重要な結果として，多重符号化を用いると，任意の単項演算（複数）が定義された有限集合で，すべての単項演算を 2-局所計算可能とできることが示される．この結果は，任意の順序機械に対し，入次数制限のある論理素子を使用して，単一のクロックでパイプライン化した順序回路を設計できることを示している．

2. 諸 定 義

2.1 冗長符号化

S を有限集合とする．S の要素数を n とする．

$\{OP_1, OP_2, \cdots, OP_m\}$ を S 上で定義された単項演算の集合とする．Σ を有限のアルファベットとし，その要素数を a とする．S に対する Σ 上の符号化 C は，Σ^k の部分集合から S への全射（上への写像）として定義する．k を符号長と呼ぶ．ここでは，符号長が一定の等長符号のみを考える．

$c\in\Sigma^k$, $s\in S$ について $C(c)=s$ であるとき，c を s に対する符号語という．符号語 c は $(x_1, x_2, \ldots, x_k)$（ただし $x_i\in\Sigma$）とベクトル表現され，x_i を第 i 桁と呼ぶ．Σ^k の要素 c で $C(c)$ が定義されていない c を非符号語と呼ぶ．$k=\lceil\log_a n\rceil$ のとき，C を最短符号化と呼ぶ．すべての s に対応する符号が唯一であるとき，C は単一符号化であるといい，ある s に対応する符号が2つ以上あるとき，C は多重符号化であるという．最短かつ単一の符号化を非冗長符号と呼び，それ以外を冗長符号化と呼ぶ．以下の議論は簡単のため2値符号化（$\Sigma=\{0,1\}$，すなわち $a=2$）を仮定するが，一般性は失わない．

2.2 単項演算の局所計算可能性

$f: \Sigma^k\to\Sigma^k$ に対し，$C(f(c))=\mathrm{OP}(C(c))$ が成り立つとき，f は符号化 C の下で S 上の単項演算 OP を実現するという．このとき f は

$$\begin{aligned}&(z_1, z_2, \cdots, z_k)\\&\quad=f(x_1, x_2, \cdots, x_k)\\&\quad=(g_1(x_1, x_2, \cdots, x_k), g_2(x_1, x_2, \cdots, x_k),\\&\qquad\cdots, g_k(x_1, x_2, \cdots, x_k))\end{aligned}$$

と書ける．すなわち，

$$z_j=g_j(x_1, x_2, \cdots, x_k)$$

と書ける．各 g_j $(j=1, 2, \cdots, k)$ が高々 ℓ 個の入力変数にしか依存しないとき，f は ℓ-局所計算可能（ℓ-locally computable）であるという．S 上の単項演算 OP が符号化 C の下で ℓ-局所計算可能であるとは，OP を実現する ℓ-局所計算可能な関数 f が存在することである．

［定義1］ S 上の単項演算の集合 $\{OP_1, OP_2, \cdots, OP_m\}$ が符号化 C の下で ℓ-弱局所計算可能（weakly ℓ-locally computable）であるとは，各演算 OP_i が ℓ-局所計算可能であることである．

［定義2］ S 上の単項演算の集合 $\{OP_1, OP_2, \cdots, OP_m\}$ が符号化 C の下で ℓ-弱局所計算可能であるとする．各演算 OP_i を実現する関数 f_i $(i=1, 2, \cdots, m)$ において，各 z_j $(j=1, 2, \cdots, k)$ が依存する ℓ 個の入力変数を共通にすることができるとき，単項演算の集合は符号化 C の下で ℓ-強局所計算可能（strongly ℓ-locally computable）であるという．

2.3 順序回路の状態割当

順序機械は，状態の集合 S とその上での入力による状態遷移（S から S への写像）として与えられる．これは，S 上の単項演算の集合が与えられることと同じである．順序機械を，論理回路（順序回路）として実現するためには，各状態を2値符号で表し，符号の各桁を記憶素子に対応させるための状態割当が必要となる．すなわち，順序回路の状態割当は，複数の単項演算が定義された有限集合に対する符号化そのものである．

順序機械は，(S, I, δ) で与えられる．S は状態の集合，I は入力の集合，δ は次状態関数で $S\times I\to S$ である．ここでは，出力については考えない．すなわち，δ で定義される S 上の $|I|$ 個の単項演算を考えることに等しい．順序機械を実現するのに，**図1**のような同期式順序回路を考える．回路は，状態を符号化して

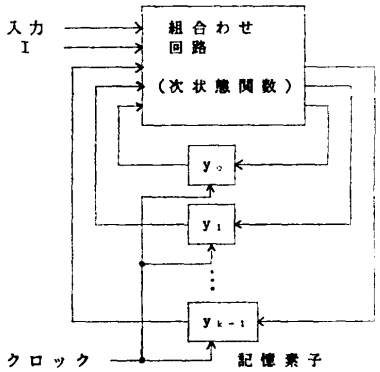


図1 順序回路
Fig. 1 A sequential circuit.

記憶する記憶素子と次状態を計算する組合せ回路からなる．記憶素子としては，Dフリップフロップを仮定し，組合せ論理回路は入次数制限のある論理素子で構成されるとする．各素子の遅延は同一であり，配線などの遅延はないとする．記憶素子には単相のクロック信号が与えられ，それに同期して回路は動作する．

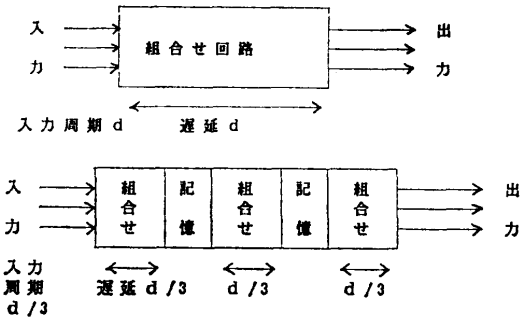
順序回路の回路規模や動作速度（クロック周期）は，次状態を計算する組合せ論理回路の複雑さに大きく依存する．一般に論理回路の複雑さは，実現する論理関数の入力変数の数に大きく依存している[9),10)]．たとえば，入次数制限のある素子で回路を構成する場合，n 入力論理関数を実現するのに必要な素子数は，ほとんどの関数で $\Theta(2^n/n)$ となることが知られている[10)]．また，回路の段数（遅延時間に比例する）もほとんどの関数で $\Theta(n)$ となる．したがって，次状態関数が強局所計算可能となるように状態集合の符号化が行えれば，それを実現する回路は規模や段数の小さなものとなることが期待できる．特に，順序回路のクロック周期は，次状態関数を計算する組合せ回路の段数に比例するから，段数の小さな回路を構成することは，回路の高速化に大きく貢献する．

論理回路の計算能力を向上させる有力な手法として，パイプライン処理がある．組合せ回路においては，回路を適当な段数で区切り，そこに記憶素子を挿入することで，回路への入力周期を短くすることがパイプライン処理である（**図2**(a)参照）．順序回路においては，次状態関数の計算がフィードバックループを形成するので，組合せ回路のように簡単ではない．種々の形態の順序回路のパイプライン化が提案され，実用されているが，その基本となる考え方は，各記憶素子の次の時刻の値を近傍の記憶素子の値だけから計算することにある．次状態関数が局所計算可能になるような状態割当を行えば，各状態変数（符号の桁）の計算が局所的に行え，フィードバックループの遅延が小さくでき，回路のクロック周期を短くできる．これは，順序回路のパイプライン処理の一般的な形と考えられる（図2(b)参照）．

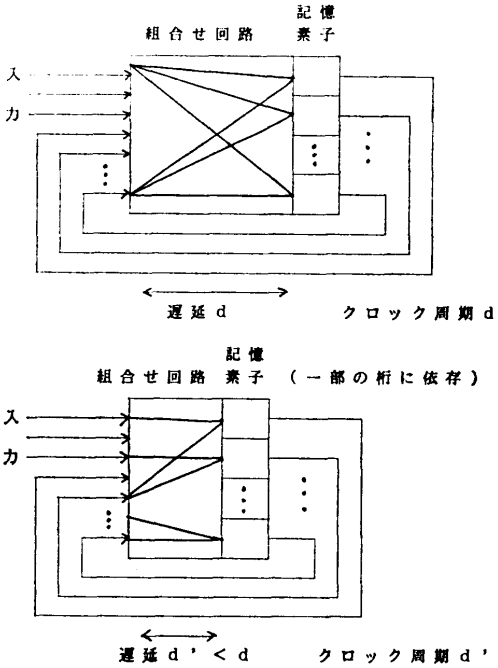
3. 非冗長な符号化

非冗長符号化は，単一かつ最短（すなわち $k=\lceil\log_2 n\rceil$）で，最も一般的に用いられている符号化である．強い制限のため，局所計算可能性は一般には小さくできない．

［定理1］ 非冗長符号化のもとでは，弱局所計算可能性に関して，$l<k$ $(=\lceil\log_2 n\rceil)$ とはできない単項演算の集合が存在する．



(a) 組合せ回路のパイプライン化
(a) Pipelining of combinational circuits.



(b) 順序回路のパイプライン化
(b) Pipelining of sequential circuits.

図2 論理回路のパイプライン処理
Fig. 2 Pipelining of logic circuits.

表1 定理1の証明
Table 1 Proof of theorem 1.

要素	単項演算 …OP…	符号語 x_1	$x_2\cdots x_i\cdots x_j$	x_k
s_0	s_0	0	$0\cdots 0\cdots 0$	$\cdots 0$
s_1	s_1	a_1	$a_2\cdots 1\cdots a_j$	$\cdots a_k$
s_2	s_1			
⋮	⋮			
s_j	s_1	0	$0\cdots 0\cdots 1$	$\cdots 0$
⋮	⋮			
s_{n-1}	s_1			

(証明) $n=2^k$ (k は正整数) とする．**表1**のようにある演算 OP で s_0 は s_0, その他の要素は s_1 に写像されるとする．$l<k$ と仮定する．s_0 に割り当てられる符号を $(0,0,\ldots,0)$ としても一般性を失わない．s_1 に割り当てられる符号を $(a_1,a_2,\ldots,a_k)$ とする．このベクトル中で i 番目の桁が0でなかったとする．ここで，仮定より x_i が依存しない桁が少なくとも1つ存在するから，それを x_j とする．符号の中で j 番目の桁だけが1であり，他のすべての桁が0であるものを考える．この符号を割り当てられた S の要素を s_j とすると，桁 x_i は桁 x_j に依存しないから，演算 OP の下で s_j と s_0 の写像先の x_i の値は一致しなければならない．しかし，$a_i=1$ と選んだので矛盾である．□

強局所計算可能性は，弱局所計算可能性より条件が強いので，上記定理と同じことが強局所計算可能性に関してもいえる．

上記の結果は，非冗長符号化のもとでは，どのような符号化を行っても，符号語のすべてのビットを見なければ各ビットの新しい値が決まらないような演算があることを示している．すなわち，局所計算可能性を実現するためには，本質的に冗長性が必要であることを示している．

4. 単一符号化

符号の冗長性には，2つの種類が考えられる．1つは，集合 S の各要素に対する符号を1つにしたままで，非符号語を入れることで冗長性を増す手法（単一符号化）と，S の要素に2つ以上の符号を割り当てる多重符号化である．

単一符号化は実用的にも，1ホット符号等として広く用いられている．まず，簡単な場合として，1つの単項演算について考える．

[定理2] 1つの単項演算 OP が定義された集合 $S(|S|=n)$ に対し，符号長 n の単一符号化で，1-局所計算可能性を実現するものが構成できる．

(証明) S の要素 s は，OP の複数回の適用によって $s=\mathrm{OP}^*(s)$ (OP^* は OP の1回以上の適用を表すとする）とできる巡回要素と，そうでない過渡要素に分けることができる．今，巡回要素の集合を $\{s_1,s_2,\cdots,s_r\}$ とし，過渡要素の集合を $\{s_{r+1},s_{r+2},\cdots,s_n\}$ とする．S の各要素 s_i, s_j に対し，次のような2項関係を定義する．

$$s_i\sim s_j \text{ iff } \mathrm{OP}^n(s_i)=\mathrm{OP}^n(s_j)$$

ここに OP^n は OP の n 回の適用を表す．この関係は，S 上の同値関係であり，S をこの同値関係で分割することができる．各同値類は巡回要素をちょうど1つずつ含むので，巡回要素 s_i を含む同値類を S_i と書く．過渡要素については，各要素を節点とし，$s_i=\mathrm{OP}(s_j)$ のとき s_j から s_i に有向枝をつけたグラフを考えると，葉から根に向かって有向枝がついた複数の木となる．各木の中で葉から根に至る最長の経路を1つ選ぶ．この経路を幹と呼ぶ．各木に属する過渡要素は，根までの距離（その要素に対応する節点から根に至る経路上の枝の数）によって同値類に分割できる．各同値類は幹の節点をちょうど1つずつ含む．過渡要素 t_i がある木の幹に属するとき，t_i を含む同値類を T_i と書く．幹に属さなかった要素は，やはり木の集まりをなす．そこで再帰的に各木について幹を選び，同値類に分割することを繰り返す．符号化 C は次のように定義する．

$$C(s)=(x_1,x_2,\cdots,x_n)$$

ただし，

$1\leqq i\leqq r$ に対し，$x_i=1$ iff $s\in S_i$

$r+1\leqq i\leqq n$ に対し，$x_i=1$ iff $s\in T_i$

この符号化では，巡回要素に対する符号語は第 r 桁まで（前半部）の1桁のみが1であり，過渡要素に対する符号語は前半部の1桁と，$r+1$ 桁以降（後半部）の複数桁が1となる．前半部の各桁は，前半部の1桁にのみ依存する．後半部の各桁も後半部の1桁にのみ依存する．簡単な符号化の例を**図3**に示す．□

単項演算の数を増やしても，1-弱局所計算可能とするような符号化が構成できる．

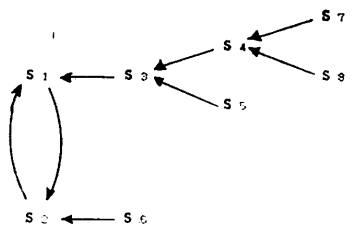
[定理3] 複数の単項演算が定義された任意の集合 $S(|S|=n)$ に対し，1-弱局所計算可能となる単一符号化が構成できる．このとき，符号長は $O(2^n)$ で十分である．

(証明) 単一符号化のもとでは，符号の各桁 x_i は，S の上の1つの2分割に対応している．任意の2分割 $\{S_1,S_2\}$ について，各演算における写像に対応して，S_3 の各要素は S_1 中の要素に写し，S_4 中の要素は S_2 へ写すような2分割 $\{S_3,S_4\}$ が存在する．すべての2分割に対応するように各桁を選んだ符号長 2^n の単一符号化は，任意の単項演算の集合を 1-弱局所計算可能とする．すなわち，上記の $\{S_1,S_2\}$ に対応する桁を x_i, $\{S_3,S_4\}$ に対応する桁を x_j とすると，

$$x_i=g_i(x_j)$$

の形で計算される．実用的には，与えられた単項演算の集合に対して 2^n より小さな必要最小限の符号長の

単項演算



分割

$S_1=\{s_1, s_4, s_5, s_8\}$
$S_2=\{s_2, s_3, s_7, s_9\}$
$T_3=\{s_3\}$ $T_4=\{s_4, s_5\}$
$T_5=\{s_5\}$ $T_6=\{s_6\}$
$T_7=\{s_7, s_9\}$ $T_8=\{s_8\}$

符号化

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
s_1	1	0	0	0	0	0	0	0
s_2	0	1	0	0	0	0	0	0
s_3	0	1	1	0	0	0	0	0
s_4	1	0	0	1	0	0	0	0
s_5	1	0	0	1	1	0	0	0
s_6	1	0	0	0	0	1	0	0
s_7	0	1	0	0	0	0	1	0
s_8	0	1	0	0	0	0	1	1

図 3 定理2の符号化例
Fig. 3 An example of coding of theorem 2.

符号化を構成的に求めることができる．□

各単項演算で参照する桁を同じにする条件をつけた強局所計算可能性では，上記の方法では，符号の全部の桁の情報が必要となってしまうことがある．

［定理4］ m 個の単項演算が定義された任意の集合 $S(|S|=n)$ に対し，$\min(m, \lceil\log_2 n\rceil)$-強局所計算可能となる単一符号化が構成できる．また，単一符号化のもとでは，強局所計算可能性の ℓ を $\min(m, \lceil\log_2 n\rceil)$ より小さくできない集合 S が存在する．

（証明） 定理3の証明と同様の議論で，一般に，

$$x_i=g_i(x_1, x_2, \cdots, x_k)$$
$$=\mathrm{OP}_1 x_{i1}^* + \mathrm{OP}_2 x_{i2}^* + \cdots + \mathrm{OP}_m x_{im}^* \quad (1)$$

(OP_j は j 番目の演算が指定されたことを表す論理変数であり，x_{ij} は j 番目の演算で x_i が依存する桁を表す．x^* は x の肯定または否定を表す）となるように符号化を決めることができる．$m<\lceil\log_2 n\rceil$ のときは(1)式を満たす符号化，それ以外は任意の最短符号化でよい．下界は以下のようにして示される．$\lceil\log_2 n\rceil<m$ の場合は，次のような演算が定義された集合を考える．ただし，$n=2^p$ (p は正整数) とする．S の要素 s_j $(j=0, 1, \ldots, n-1)$ の演算 OP_i $(i=0, 1, \ldots, \lceil\log_2 n\rceil-1)$ における写像は，j の $\lceil\log_2 n\rceil$ ビットの2進数表現の 2^i の桁が0のとき s_0，1のとき s_1 とする（**表2**参照）．要素 s_0 と s_1 を分離する符号語の桁 z は少なくとも1個存在する．このとき，$s\neq s'$ なる任意の S の要素について，演算 OP_i $(i=0, 1, \ldots, \lceil\log_2 n\rceil-1)$ のいずれかで，これらの写像先は異なるので，z はすべての要素を分離するのに必要な少なくとも $\lceil\log_2 n\rceil$ 個の入力変数に依存する．次に $\lceil\log_2 n\rceil\geq m$ の場合を考える．同様に要素数 n の集合 S' $(n=2^p, p$ は正整数) を考える．S' の要素 s_j $(j=0, 1, \ldots, n-1)$ の演算 OP_i $(i=0, 1, \cdots, m-1)$ における写像先は，j の $\lceil\log_2 n\rceil$ ビットの2進数表現の 2^i の桁が0のとき s_0，1のとき s_1 である．要素 s_0 と s_1 を分離する桁 z は少なくとも1個存在する．S' の要素は 2^m 個の部分集合に分かれ，異なる集合に属する要素においては，ある適当な演算によって写像先が異なる．すなわち，z は 2^m 個の部分集合を分離するのに必要な m 個以上の変数に依存する．□

表2 定理4の証明
Table 2 Proof of theorem 4.

要素	単項演算 OP_0 OP_1 $\mathrm{OP}_2\cdots\mathrm{OP}_{p-1}\cdots\mathrm{OP}_m$	符号語 $\cdots z\cdots$
S_0	s_0 s_0 $s_0\cdots s_0$	0
S_1	s_1 s_0 $s_0\cdots s_0$	1
S_2	s_0 s_1 $s_0\cdots s_0$	
S_3	s_1 s_1 $s_0\cdots s_0$	
S_n	s_1 s_1 $s_1\cdots s_1$	

このように，単一符号化による冗長性では，一般に強局所計算可能性を n に依存しない定数まで落とすことはできない．

5. 多重符号化

多重符号化は，1つの要素に複数の符号を割り当てることを許すもので，符号化の自由度は飛躍的に増大する．この自由度を利用して局所計算可能性を小さくできる．複数の単項演算が定義された任意の集合 S に対し，2-強局所計算可能とするような符号化が構成できる．

［定理5］ m 個の単項演算が定義された要素数 n の任意の集合 S に対し，その演算の集合を 2-強局所計算可能とするような符号長 $O(n^2\log n)$ の多重符号

化が構成できる．

（証明） 符号化には，並列プレフィックス計算（PPC）を利用する[10)~12)]．ここでは，単項演算の強局所計算可能な符号化を構成する問題を，順序回路の状態割当問題として考える．以下に，単項演算を実現する順序回路の構成法を示す．与えられた有限集合 S と単項演算の集合に対し，順序機械 (S, I, δ) を考える．入力集合 I は単項演算の集合そのものである．次状態関数は，各単項演算で指定される写像の集合として一意に決まる．

1) 集合 S に対し，S から S の中への自己写像全体の集合 M を考える．S の単項演算は，M の中の写像の1つに対応する．自己写像の合成を2つの写像に対する2項演算と見ると，M はこの演算によって，モノイドをなす．M の要素は n^n 個であるから，その符号化には $n\log_2 n$ ビットあればよい．この符号化は，S の要素を $\log_2 n$ ビットで単純に非冗長符号化し（たとえば，通常の2進符号を用いる），s_0 の写像先から s_{n-1} の写像先までの符号を順に並べたものとする．

2) モノイドの演算を計算する2入力論理素子のみで構成した組合せ回路（以下I回路と呼ぶ）を構成する．モノイド演算は，第1オペランドの要素 s_i の写像先が s_j ならば，演算結果の要素 s_i の写像先を第2オペランドの要素 s_j の写像先とすればよい．すなわち，図3に示すような n 個のセレクタ回路によって実現できる．セレクタ回路は，段数 $O(\log_2 n)$，素子数 $O(n\log_2 n)$ で実現できる．I回路のすべての入力から出力に至る経路上の素子の数が同じになるように，1入力のバッファを適当に挿入する（同期化）．このとき，回路の段数は増加しないようにできる．また，素子数のオーダも変わらない．I回路の段数を D とする．すなわち，I回路は段数 $D=O(\log_2 n)$，素子数 $O(n^2\log_2 n)$ で実現できる（**図4**）．以下，議論を簡単にするため，D は2のべき乗とする．

3) 単項演算 OP_i から対応する M の要素の符号に変換する回路（Ip回路）を用意する．与えられた単項演算は m 個であるが，$\lceil\log_2 m\rceil$ ビットで2進符号化されていると仮定する．Ip回路はこれを制御入力とする簡単なセレクタ回路で構成できる．セレクタのデータ入力側は各単項演算に対応する M の要素の符号であり，固定入力である．Ip回路の素子数は $O(mn\log_2 n)$，段数は $O(\log_2 m)$ とできる．I回路と同じように同期化しておく．

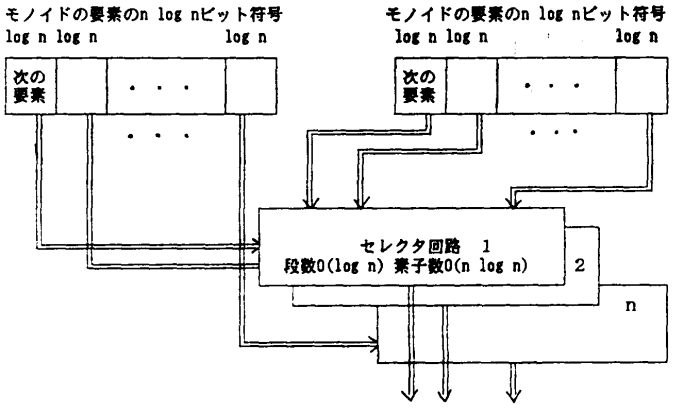


図4 I回路の構成
Fig. 4 A block diagram of I circuit.

4) I回路からなる高さ $(\log_2 D)-1$ の完全2分木を作る．根のI回路の出力をさらに別のI回路（最終I回路という）の第2オペランド入力とし，その出力は自分自身の第1オペランド入力とする．

5) 入力された演算 OP_i を Ip 回路を通し，長さ D のシフトレジスタの入力とする．また，長さ D の1ビットの巡回シフトレジスタを用意し，その中の1つのビットを基準ビットとする．巡回シフトレジスタには，1ケ所のみ1をセットしておく．基準ビットが1のとき，入力シフトレジスタの内容が一斉に2分木回路の入力に送られるように回路を組む（**図5**）．すなわち，D クロックごとにシフトレジスタの内容が2分木の葉節点に送られる．シフトレジスタ部分は，素子数 $O(n\log^2 n)$ である．また，順序機械の実現のためには，初期状態を保存するために $\log_2 n$ ビットのレジスタが必要である．

6) Ip 回路を除き，回路を構成するすべての論理素子の出力に記憶素子（フリップフロップ FF）を挿入する（パイプライン化，図5(b)）．このようにすれば，回路中のすべての記憶素子は高々2つの他の記憶素子の出力にしか依存しないようにできる．

7) このようにして組んだ順序回路（図5）のすべての記憶素子を符号の各桁に対応させる．各記憶素子（桁）は高々2個の記憶素子（桁）にしか依存しないので，次状態関数は2-局所計算可能となっている．

8) この回路が集合 S 上の任意の単項演算の集合を計算することは次のように示される．最終I回路は D クロックごとに M 上の演算を計算する．シフトレジ

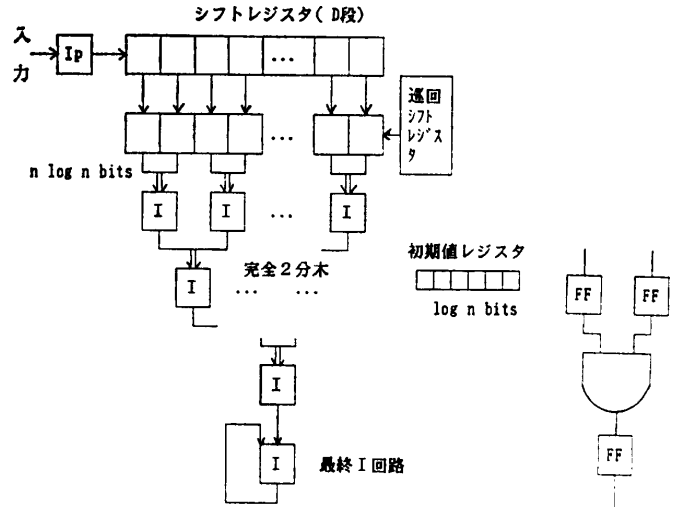


図 5 単項演算を実現する順序回路
Fig. 5 A sequential circuit computing unary operations.

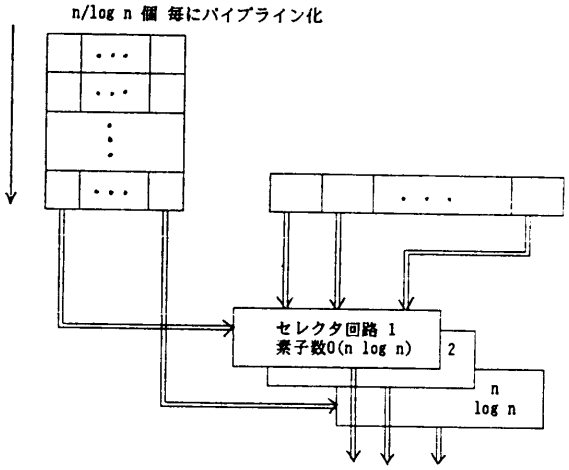


図 6 パイプライン化した I 回路
Fig. 6 Pipelining of I circuit.

スタは，D クロック分の入力を記憶して，D クロックごとに2分木に入力情報を供給する．D クロック分の入力情報は D クロックごとに I 回路を1段通り，半分のデータ量に圧縮される．$D(\log_2 D-1)$ クロックで入力情報は木の根に到達し，元の D 個の入力の演算結果が得られる．これを，最終 I 回路が累積する．すなわち，2分木の並列計算の速度を利用して，入力を D 個ずつまとめてパイプライン的に演算している．この基本的な演算の手法は PPC として広く知られているものである[11),12)]．

9) 次に，符号長について考える．論理素子の出力に記憶素子が接続されているので，回路の素子数は記憶素子数と同じオーダとなる（ただし，Ip 回路は除く）．I 回路は図4のような単純なモノイドの符号化を使えば，段数 $O(\log n)$，素子数 $O(n^2\log n)$ で作れる．2分木回路全体では，$O(\log n)$ 個の I 回路が必要であるから $O(n^2\log^2 n)$ の記憶素子数となる．その他，シフトレジスタ部が $O(n\log^2 n)$ の記憶素子を持つ．よって，符号長は，$O(n^2\log^2 n)$ である．

10) I 回路は D クロックごとにしか動作していない．そこで，I 回路での計算をさらにパイプライン化することが可能である．I 回路を**図 6** のように $n/\log_2 n$ ビットごとにパイプライン化すれば，I 回路の素子数は $O(n^2)$ とでき，全体の符号長は $O(n^2\log n)$ とできる．□

定理5の証明は，順序機械のパイプライン化の手法を与えている．

［系］（パイプライン化定理） 任意の順序機械に対し，すべての状態変数が高々2つの状態変数にしか依存しないような状態割当が構成できる．すなわち，すべての順序機械はパイプライン化できる．

いいかえれば，どのような順序機械でも多重符号化を用いた状態割当によって，クロック周期を論理素子1段分の遅延の程度まで短くできる．これは，高性能の論理回路の設計に対する1つの大きな可能性を与える結果である．

6. あとがき

単項演算の局所計算可能性が符号の冗長性とどのように関係するかを明らかにした．非冗長符号化では，一般的には局所計算可能性は実現できない．単一符号化を用いると，弱局所計算可能性は実現できるが，強局所計算可能性は実現できない．多重符号化では，任意の単項演算の集合が 2-強局所計算可能とできる．これらの結果は，符号化の冗長性と局所計算可能性の関係を明確に与えている．

さらに，これらの結果は，順序機械の状態割当にも応用できる．特に，多重符号化を用いたパイプライン化定理は，高性能回路の設計に大きな展望を与えるものと考える．

今後の課題としては，符号長に関する限界を明確にすることが重要である．また，単項演算だけでなく，2項演算についての同様の議論も興味深い課題である．並列計算の基礎的な理論として局所計算可能性と

冗長性の問題は，今後実用的にも理論的にも大いに研究されるべき問題であると考える．

謝辞 ご討論いただいた本学電子工学教室田丸啓吉教授ならびに同研究室の各位，情報工学教室矢島研究室の各位に感謝いたします．また，有益なご意見をいただいた査読者に深謝いたします．

参考文献

1) 安浦，高木，矢島：冗長符号化を利用した高速並列アルゴリズムについて，信学論，Vol. J70-D, No. 3, pp. 525-533 (1987).
2) Avizienis, A.: Signed-Digit Number Representation for Fast Parallel Arithmetic, *IEEE Trans. Elec. Comput.*, Vol. EC-10, No. 3, pp. 389-400 (1961).
3) Spaniol, O.: *Computer Arithmetic*, p. 194, John Wiley & Sons, New York (1981).
4) 樋口，亀山：多値情報処理，p. 102, 昭晃堂，東京 (1989).
5) 高木，安浦，矢島：冗長2進加算木を用いたVLSI向き高速乗算器，信学論，Vol. J66-D, No. 6, pp. 683-690 (1983).
6) 高木，矢島：算術演算のハードウェアアルゴリズム，情報処理，Vol. 26, No. 6, pp. 632-639 (1985).
7) Hartmanis, J.: On the State Assignment Problem for Sequential Machines, *IEEE Trans. Elec. Comput.*, Vol. EC-10, No. 2, pp. 157-167 (1961).
8) De Michelli, G., Brayton, R. K. and Sangiovanni-Vincentelli, A. L.: Optimal State Assignment for Finite State Machine, *IEEE Trans. CAD*, Vol. CAD-4, pp. 269-285 (1985).
9) Savage, J. E.: *The Complexity of Computing*, John Wiley & Sons, New York (1976).
10) 安浦：論理回路の複雑さの理論，情報処理，Vol. 26, No. 6, pp. 575-582 (1985).
11) Unger, S. H.: Tree Realization of Iterative Circuits, *IEEE Trans. Comput.*, Vol. C-26, No. 4, pp. 365-383 (1977).
12) Ladner, R. E. and Fischer, M. J.: Parallel Prefix Computation, *J. ACM*, Vol. 27, No. 4, pp. 831-838 (1980).

(平成元年9月7日受付)
(平成2年2月13日採録)

安浦 寛人（正会員）

昭和28年生．昭和51年京都大学工学部情報工学科卒業．昭和53年同大学院修士課程修了．昭和55年4月より京都大学工学部情報工学教室助手．昭和61年11月，京都大学工学部電子工学教室助教授となり現在に至る．工学博士．非同期回路，論理回路の複雑さ，VLSI向きハードウェアアルゴリズム，論理設計のCAD等の研究に従事．IEEE, ACM, 電子情報通信学会，EATCS, LAシンポジウム，日本ソフトウェア科学会各会員．