EC81-34

データフロー計算機構成のための一考察

A CONSIDERATION ON CONSTRUCTION OF DATA-FLOW COMPUTER

松原康夫
Yasuo MATSUBARA
（山梨大学工学部）
(Yamanashi University)

野口正一
Shoichi NOGUCHI
（東北大学電気通信研究所）
(Tohoku University)

1、まえがき

データの流れによって、自然に高度な並列処理を実現する計算機として、データフロー計算機が研究されている。[1]~[4] データフロー計算機はノイマン型の計算機とは異った原理で作動するため、ハードウェアも従来とは異った構成法が要求される。実際、MITのDennis等のアーキテクチャでは、構成要素間でパケットをやりとりすることによって高度に並列な動作を実現している。

本稿では、データフローグラフとして計算機のハードウェアを構成することを考える。そのために、データフローグラフの有向枝と節点をそれぞれP要素，t要素と呼ぶ非同期式の順序回路としてモデル化するが、実際に使用する全ての種類のt要素を、モジュールまたはICとして用意することは不可能である。そこで、本稿ではできるだけ少い種類のもので、任意のネットワークが実現できるような万能なt要素集合を見い出すことを目的とする。

パケット（トークン）のデータ長はbビットに固定し、このbビットのデータは同時に移動するものとする。より大きいデータ長のものは並列に移動する複数のパケットで代用することにする。

Keller[5]は非同期回路に対して万能なモジュール集合を求めているが、データフローグラフではトークンの有無よりその内容の方が重要となるので、異った定式化が必要である。

2、諸定義

［定義1］非同式順序回路（以後単に順序回路という）Aは4項組 $A=(X, Y, Z, E)$ である。ここで、

1) X, Y, Z はそれぞれ入力変数，内部変数，出力変数の有限集合であり、制御変数の集合 X^c, Y^c, Z^c とデータ変数の集合 X^d, Y^d, Z^d に分けられる。$X=X^c \cup X^d$, $Y=Y^c \cup Y^d$, $Z=Z^c \cup Z^d$

現在の状態における内部変数 $y \in Y$ に対して、次の状態における y の値を y' で表わし、$Y'=\{y' \mid y \in Y\}$ とする。

Y^c の変数によって表わされる状態を制御状態といい、Y^d の変数は制御状態が変化するときにだけ変化するものとする。$Y^d=\{M_1, \cdots, M_a\}$ とし、第i成分が M_i であるa次元ベクトルをMで表わす。

2) E は、XとYの変数の値からZとY'の変数の値を決定する論理式の集合である。但し Z^c の変数の値は Y^c の変数だけの関数とする。

［定義2］順序回路Aの仕様は、次のようにして得られる有向グラフである。

I、各制御状態に対応する節点を描き、各制御状態に於ける出力変数の値を論理式で与える。初期状態は0で表わす。

II、制御状態間の直接的な遷移関係を有向枝で表わし、その遷移が起こるために X と Y^d の変数が満たすべき条件を与える。そしてMが変化するときは、Mの新しい値を式で与える。

（定義終）

データフローグラフの有向枝に対応して、パケットを貯えるキューとしてP要素(P-element)を用意する。このキューは高々1個のパケットを貯える。パケットの内容bビットで表わされる。

[定義3]†自然数iに対してP要素p_iは$X^c=\{\Lambda_i, r_i\}$, $X^d=\{z_i^1,\cdots,z_i^b\}$, $Y^c=\{e_i, f_i\}$, $Y^d=\{x_i^1,\cdots,x_i^b\}$である順序回路$(X, Y, Z, E)$であり、図1の仕様を持つ。ここで、$z_i$, x_iはそれぞれ第j成分がz_i^j, x_i^jであるb次元ベクトルである。

制御状態0のときp_iが空であるといい、制御状態2のときp_iにパケットがあるという。1と3はそれぞれ書き込みとリセットの過渡状態である。(定義終)

そして、データフローグラフの節点に対応して、入力P要素のパケットを処理して出力P要素にパケットを置く回路としてt要素を考える。t要素は、入力P要素のパケットの有無と内容、出力P要素のパケットの有無を知って、入力P要素をリセットしたり、出力P要素にパケットを置く等の操作をする。

[定義4]††t要素tは3項組$t=(I, O, A)$である。ここで、1) I, Oはそれぞれ入力P要素と出力P要素のサフィックスの順序付けられた集合であり、2) Aは$X^c=\{f_i \mid i\in I\}\cup\{e_i \mid i\in O\}$, $X^d=\{x_i^1,\cdots,x_i^b \mid i\in I\}$, $Y^c=\{r_i \mid i\in I\}\cup\{\Lambda_i \mid i\in O\}$, $Y^d=\{z_i^1,\cdots,z_i^b \mid i\in O\}$であるような順序回路であり、次のI～IVの条件を満たすものとする。(図2)

I、tの各制御状態qにおいて$I_q=\{i\in I \mid r_i=1\}$, $O_q=\{j\in O \mid \Lambda_j=1\}$としたとき、tが制御状態$q$に移るのは$X^c$の変数が$(\forall i\in I_q) f_i=1$, $(\forall j\in O) e_j=1$を満たしているときに限り、$(\exists i\in I_q) f_i=1$又は$(\exists j\in O_q) e_j=1$の間はqに留まる。そして$(\forall i\in I_q) f_i=0$, $(\forall j\in O_q) e_j=0$になると、出力変数が$(\forall i\in I) r_i=0$, $(\forall j\in O)\Lambda_j=0$であるような制御状態に移る。

II、$i\in I$について、$f_i=1$でもなく$\Lambda_i=1$でもないときに$x_i^1,\cdots,x_i^b$のいずれかに依存するような$Z\cup Y'$の変数は存在しない。

† Pは Packet又は place のP。e, f, Λ, rはそれぞれ empty, full, set, reset を意味する。

†† tは transmit又は transform のt。

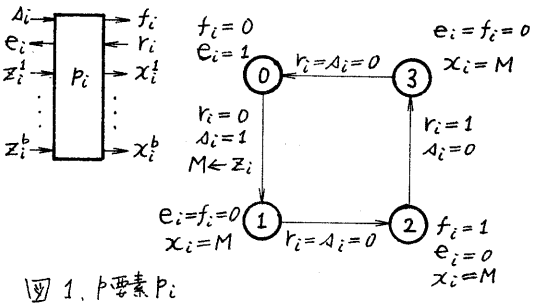


図1. P要素p_i

III、tは$j\in O$について$\Lambda_j=1$の間$z_j^1,\cdots, z_j^b$を変化させない。

IV、初期状態においては$(\forall i\in I) r_i=0$, $(\forall j\in O)\Lambda_j=0$とする。(定義終)

次に、いくつかのP要素の間をt要素で接続したものとしてネットワークを定義する。

[定義5]ネットワークNは4項組$N=(\mathfrak{P}, \mathfrak{T}, \mathfrak{C}, \mathfrak{M})$である。ここで1) $\mathfrak{P}=\{p_1,\cdots, p_{|\mathfrak{P}|}\}$: P要素の有限集合。2) $\mathfrak{T}=\{t_1,\cdots, t_{|\mathfrak{T}|}\}$: t要素の有限集合であり、$t_i=(I_i, O_i, A_i)\in\mathfrak{T}$について$\{p_j \mid j\in I_i\cup O_i\}\subset\mathfrak{P}$とする。ネットワークの初期状態において各t要素は制御状態0にあるものとする。3) $\mathfrak{C}=\mathfrak{C}_1\cup\mathfrak{C}_2$であり、自然数の集合$\mathbb{N}$に対して、$\mathfrak{C}_1\subset\mathfrak{P}\times\mathfrak{T}\times\mathbb{N}$, $\mathfrak{C}_2\subset\mathfrak{T}\times\mathbb{N}\times\mathfrak{P}$である。$(p_i, t_j, k)\in\mathfrak{C}_1$は$p_i$が$t_j$のk番目の入力P要素であることを示し、$(t_i, j, p_k)\in\mathfrak{C}_2$は$p_k$が$t_i$のj番目の出力P要素であることを示す。4) $\mathfrak{M}: \mathfrak{P}\to\{\lambda\}\cup\{0,1\}^b$はマーキングであり、ネットワークの初期状態におけるp要素の状態を表わす。$\mathfrak{M}(p_i)=\lambda$は$p_i$が空であることを表わし、$\mathfrak{M}(p_i)=a\in\{0,1\}^b$は$p_i$に内容$a$のパケットが存在することを表わす。

$t_i\in\mathfrak{T}$, $p_j\in\mathfrak{P}$に対して、p_jがt_iの入力(出力)P要素であるとき、t_iがp_jの出力(入力)t要素であるという。

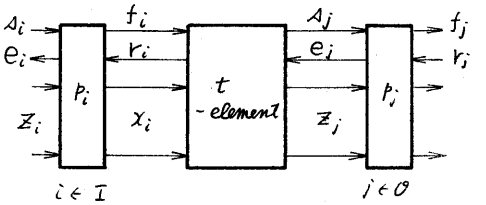


図2. t要素とP要素の間の接続

P要素は高々1つの入力t要素と高々1つの出力t要素を持つ。入力(出力)t要素のないP要素は、ネットワークの入力(出力)P要素であり、ネットワークはこれらを通して外部とやりとりする。

ネットワークは、t要素を節点とし、P要素を入力t要素から出力t要素に向う有向枝とする有向グラフによって表わされる。

[定義6] t要素 $t=(I,O,A)$ の $I\cup O$ の濃度をtの結合数という。t要素の集合の結合数は、各t要素の結合数の最大値である。濃度n、結合数nのt要素の集合を (m,n) の集合という。 (定義終)

t要素をIC化するときには、結合数と $(b+2)$ の積がピン数を与えるので、結合数の小さいことが特に望ましい。結合数2では限られたネットワークしか構成できないので、特に結合数3のt要素だけで任意のネットワークを構成できるかどうかが問題となる。

[定義7] ネットワークNの初期状態から開始して、外部から、i番目の入力P要素にパケット列 $u_i\in(\{0,1\}^b)^*$ を送り込む。$(i=1\sim k)$、このとき、j番目の出力P要素から外部に取り出されるパケット列を $v_j\in(\{0,1\}^b)^*\cup(\{0,1\}^b)^\omega$ とする。$(j=1\sim\ell)$ 但しNはk個の入力P要素とℓ個の出力P要素を持つものとする。このとき、$u=(u_1,\cdots,u_k)$ を入力様相、$v=(v_1,\cdots,v_\ell)$ を出力様相という。

1つの入力様相に対して1つの出力様相が定まるとき、Nは決定的な様相関係を持つといい、$g(u)=v$ である関数gをNの様相関数という。 (定義終)

3. 直接的なt要素

次に、入力P要素にパケットの到着するタイミングや、出力P要素が空になるタイミングに依存しないため、構成が容易になるサブクラスを考える。

[定義8] 順序回路 $A=(X,Y,Z,E)$ において、入力変数vに着目する。制御状態qに対して関数 $g_q:\{0,1\}^{|X\cup Y^d|}\to\{0,1\}^{|Y\cup Z|}$ は制御状態がqであるときに変数 $v,u_1,\cdots,u_k$ $(\{v,u_1,\cdots,u_k\}=X\cup Y^d)$ の値から $Y\cup Z$ の変数の値の組 $g_q(v,u_1,\cdots,u_k)$ を与えるものとする。

Aが制御状態qにおいてvに依存するとは $g_q(0,u_1,\cdots,u_k)\neq g_q(1,u_1,\cdots,u_k)$ であるような $u_1,\cdots,u_k$ の値の組合せが存在することをいう。

[定義9] t要素 $t=(I,O,A)$ は次のIとIIの条件を満たすとき直接的であるという。又、t要素が全て直接的なネットワークを、直接的なネットワークという。

I、$i\in I$ について、Aが制御状態qにおいて $r_i=0$ であり、しかも $f_i,x_i^1,\cdots,x_i^b$ のいずれかに依存するならば、$f_i=1$ のときにだけqからの遷移が定義され、そして必らず $r_i=1$ の状態に移る。

II、$j\in O$ について、Aが制御状態qにおいて $a_j=0$ であり、しかも e_j に依存するならば、$e_j=1$ のときにだけqからの遷移が定義され、そして必らず $a_j=1$ の状態に移る。 (定義終)

直接的なt要素は、入力P要素を待つときはパケットが到着するのを待ち、セットし、出力P要素を待つときは空になるのを待ち、パケットを置く。そして、待ち合わせている事象が全て起らないと動作しない。

[定理1] 直接的なネットワークは、決定的な様相関係を持つ。

(証明) 直接的なt要素 $t=(I,O,A)$ について考える。今仮に、制御状態qにおいて、$a_j=0$ であってもが e_j に依存するようなjの集合を $O_q\subset O$ とし、$r_i=0$ であってもが $f_i,x_i^1,\cdots,x_i^b$ のいずれかに依存するようなiの集合を $I_q\subset I$ とする。

tにおいては $(\forall j\in O_q)\ e_j=1,(\forall i\in I_q)f_i=1$ のときにだけqからの遷移が定義される。そして、そのときの動作は Y^d の変数と $i\in I_q$ に対する $x_i^1,\cdots,x_i^b$ の値によって定められる。

従ってt要素の動作がタイミングによって異ることはなく、あるときに出力するパケットの内容はそのときおよびそれ以前に入力したパケットの内容の関数となる。従って、ネットワークの出力パケットの内容は、初期状態におけるP要素の内容と入力様相の関数となる。又、t要素の動作が他のt要素の動作を阻止することはないので、何個までのパケットが出力されるかも一意的に定まる。 (証明終)

〔定義10〕ネットワークNにおいて、t要素の系列 $t_1, \cdots, t_m \in T$ が待ち状態にあり、$i=2\sim n$ について、t_{i-1} と t_i が1つのP要素の入出力t要素であり、t_{i-1} が動作しない限り t_i が動作できない関係にあるとき、この系列を t_1 から t_m への待ち合わせの系列という。

〔定理2〕ネットワークの様相関数で、直接的なネットワークでは実現できないものが存在する。

(証明) 図3のt要素を考える。$(0\cdots0)$ から成る、出力パケット列の長さは、入力パケット列の長さの和となる。このt要素の様相関数が、直接的なネットワークNで実現されたと仮定する。p_1 の出力t要素を t_1, p_2 の出力t要素を t_2, p_3 の入力t要素を t_3 とすると、入力パケットが1つも到着しないとき、t_1 から t_3 への待ち合わせの系列と、t_2 から t_3 への待ち合わせの系列が存在する。何故なら、直接的なt要素がこの状態で動いているならば、パケットの到着を知ることができないからである。t_2, t_3 から見ていって、2つの系列が共有する最初のt要素を t_m とする。t_m は直接的なので、p_1 と p_2 の両方にパケットが到着しないと動作しないので、矛盾を引き起こす。

(証明終)

〔定義11〕ある自然数 k, ℓ について、k個の入力P要素と ℓ 個の出力P要素を持ち、ある関数 $g: \{0,1\}^{b\times k} \to \{0,1\}^{b\times \ell}$ に対して、仕様が図4のように表わされるt要素を関数要素という。又、関数要素と同じ様相関数を持つネットワークを関数ネットワーク又は単に関数という。

(定義終)

ここでいくつかの関数要素を導入しておく。(図5)

〔定理3〕任意の関数要素に対して、同じ様相関数を持つネットワークが、Copy, NAND, Up, Downから実現できる。

(証明) 入力パケットはコピーできるので、k入力1出力の場合を考えれば十分である。

先ず、各入力パケットが1つずつ到着したことを確認するために、入力パケットのコピーをとって $k-1$ 個のNANDの木により1つのパケットにまとめる。これを図6b)の回路を通して $(1\cdots1)$ というパケットを作る。そして

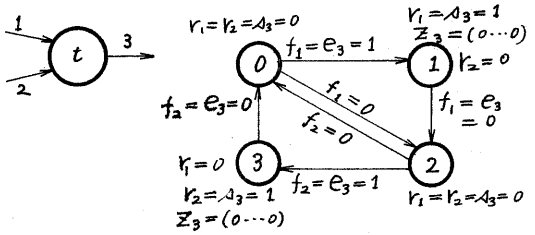


図3. あるt要素

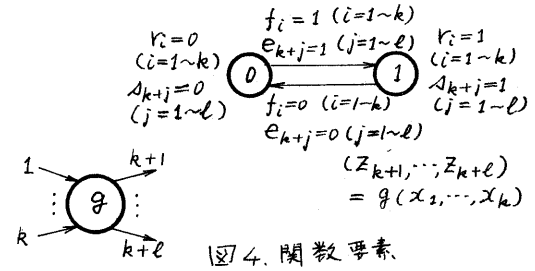


図4. 関数要素

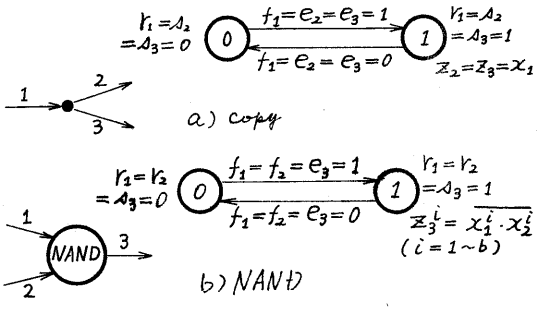


図5. いくつかの関数要素

これに $b-i$ 回Upを行ったものと $i-1$ 回Downを行ったもののANDを行い、ビットiだけが1で他が0のパケットをつくる。これをマスクパケットという。

他方、出力パケットのビットiが、入力パケットの中のm個のビットに依存するならば、それらのビットを含むm個のパケットを作る。そして各ビットを、ビットiの位置に移動する。

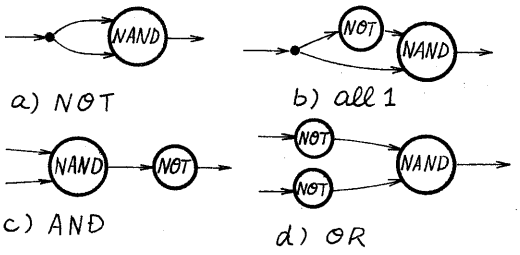


図6. Copy と NANDによる関数

これらに Copy と NANDを何回か適用することにより、任意の論理関数が実現できる。最後に、マスクパケットとANDをとってビット i を取り出し、$i=1\sim b$ についてのORをとって出力パケットとする。（証明終）

次に、パケットの内容によって動作が左右されるt要素を導入する。（図7）

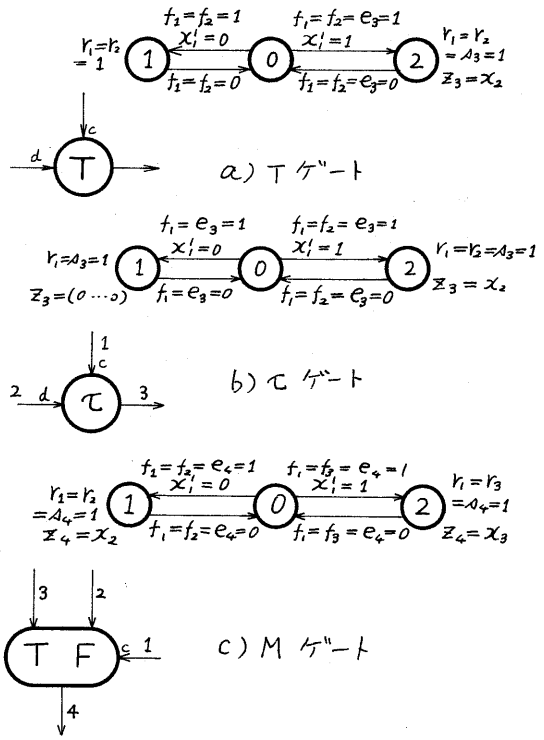


図7. パケットによって制御されるt要素

［定義12］t要素が、Copy, NAND, Up, Down, Tゲート, Mゲートのいずれかであるようなネットワークを前基本データフローグラフ（PDFG）という。（定義終）

ここでは、Dennis等のPIDFLに対して関数を強化したものを考えている。

［定理4］任意の直接的なネットワークNに対して、t要素がCopy, NAND, Up, Down, Tゲート, でゲートから成るネットワークで、同じ様相関数を持つものが構成できる。

（証明）最初にNを次のように変換してN′とする。

先ず、N全体の入出力P要素以外の各P要素 P_i に対して、逆向きのP要素 $P_{i'}$ を付け加える。そして、初期マーキングにおいては P_i にパケットが存在するならば $P_{i'}$ は空であり、P_i が空ならば $P_{i'}$ には（1…1）というパケットが在るものとする。

そして各t要素tを次のように改造する。出力P要素 P_j にパケットを置くときは $P_{j'}$ をリセットするようにし、入力P要素 P_i をリセットするときは $P_{i'}$ に（1…1）を置くようにする。但し、$P_{j'}$ にパケットが在ることや $P_{i'}$ が空であることを確認してから行うものとする。

改造したN′の各t要素tを次のような回路で模倣する。

tの内部変数を、いくつかのP要素に貯えられるパケットで表わす。但し、パケットの1ビットが1つの内部変数に対応する。

tの入力P要素に対して、これをα入力とするでゲートを用意し、tの出力P要素に対して、これを出力とするTゲートを用意する。

内部変数から、各入力P要素毎にパケットが到着するのを待ち合わせるかどうかを計算する関数を用意する。この出力をでゲートのC入力に入れる。でゲートは待ち合わせる場合は入力パケットを待って同じ内容のパケットを出力し、そうでない場合は（0…0）を出力する。

次に、でゲートの出力と内部変数とを入力とする3種類の関数を用意する。第1は、tの各出力P要素毎に、パケットを出力するかどうかを計算するものであり、その結果をTゲートのC入力に入れる。第2は出力P要素毎の出力パケットの内容を計算するものであり、結果をTゲートのα入力に入れる。Tゲートは、パケットを出力しないとき、α入力のパケットを吸収してしまう。第3の関数は、次の状態における内部変数の値を計算するものであり、結果を元

のP要素に戻す。

N′において、tの入力P要素 P_i をリセットし、さらに P_i′にパケットを置くことが、Nにおいて P_i をリセットしたことになる。

（証明終）

[定理5] PDFGによって実現される構相関数のクラスは、直接的なネットワークによって実現される構相関数のクラスに等しい。

（証明）てゲートが、Copy, NAND, Mゲートを使って図8のように実現できることと、Mゲートが直接的なt要素であることから明らか。

（証明終）

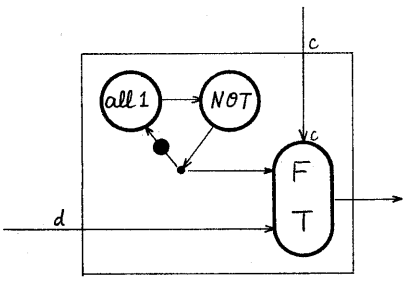


図8 てゲートの実現

4. 一般のt要素

直接的なt要素は構成が容易であり、定理5から分るように、データフローによる情報処理機能の重要なサブクラスを与える。しかし、一つの計算システムを構成するのには不十分である。

例えば、MITのDennis等のアーキテクチャでは、裁定ネット（A-net）及び分配ネット（D-net）を通してパケットを目的地に送るが、これらはパケットの衝突と分岐を繰り返すものであり、図9 a）の裁定要素とb）のスイッチのようなものから構成される。同図b）において $P(x_1)$ は x_1 を引数とする述語関数とする。ところが、裁定要素のように衝突を処理するものは、直接的なt要素からは実現できない。これは、定理2と同様にして証明できる。又、相互排除だけでなく、どちらか空いている方にパケットを分配する図9 c）の分配要素も直接的なt要素からは実現できない。

図9において、各制御状態に対して、Z^c の変数で、値が1のものだけを記している。以後

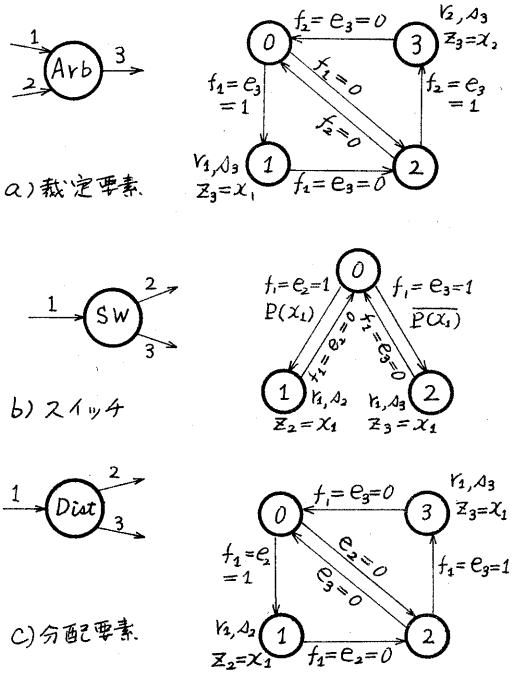


図9. a）Arb, b）SW c）Dist

同様の略記法を用いる。

次に、任意のネットワークを構成するために、2つのt要素IGとOGを導入する。（図10）

IGは、P_1 に（0…0）が来ると、$f_2=1$ のときは（0…01）と x_2 の2つのパケットを返す。$f_2=0$ のときは（0…0）を2つ返す。

又、P_1 に（10…0）が来ると何もせずに（10…0）を返し、P_1 に（1…1）が来ると、P_2 をリセットして（10…0）を返す。

OGは、（0…0）が来ると（0…0f_2）を P_3 に出力する。（10…0）が来ると、次に来るパケットを吸収して（1…1）を P_3 に出力し、（1…1）が来ると、次に来るパケットを P_2 に出力して（1…1）を P_3 に出力する。

[定理5] Copy, NAND, Up, Down, Tゲート, てゲート, IG及びOGで任意のネットワークが実現できる。

（証明）ネットワークを構成する各t要素tを次のような回路で模倣する。

内部変数をいくつかのパケットで表わす。パケットの1ビットが1つの変数に対応する。tの入力P要素 P_i に対して、P_i をIGの入

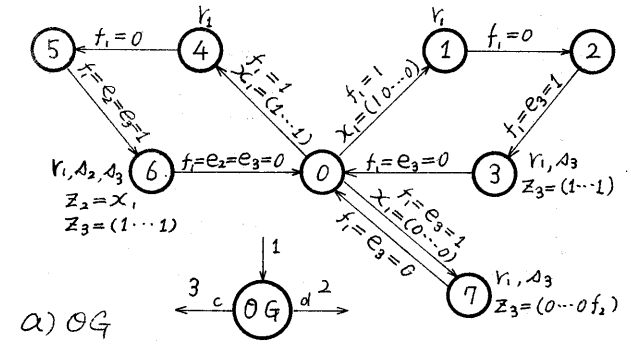


図10. OGとIG

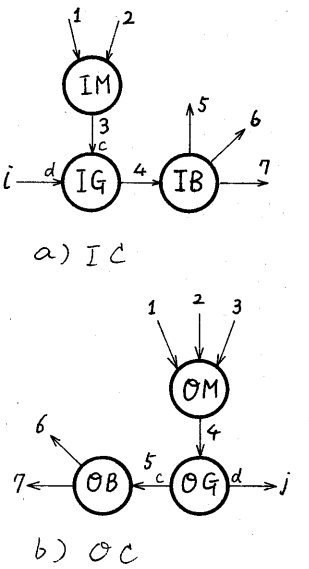


図11. ICとOC

力とする図11 a)の回路ICを用意し、tの出力P要素 p_j に対して、p_j をOGのd出力とする図11 b)の回路OCを用意する。但し、IM, IB, OM, OBはそれぞれ図12に示す仕様を持つ直接的な要素とする。

そして、内部変数から パケット(0…0)を作り出す関数を用意する。これのコピーをとって、ICとOCのそれぞれの入力1に入れる。これに対して、ICの5と6; OCの6に入出力P要素の状態を表わすパケットが出力される。

IC及びOCからのパケットと内部変数を入力とする、4種類の関数を用意する。

第1は、各入力P要素に対して、リセットするとき(1…1)を出力し、リセットしないとき(10…0)を出力する関数であり、この出力をICの入力2に入れる。

第2は、各出力P要素に対して、パケットを置くとき(1…1)を出力し、置かないとき(10…0)を出力する関数であり、これをOCの入力2に入れる。

第3は、各出力P要素に置くパケットの内容を計算する関数であり、出力をOCの入力3に入れる。

そして第4の関数は、ICの出力7からの応答パケットと、OCの出力7からの応答パケットが全て揃ったときに、次の状態における内部変数の値を計算するものである。

こうして、IGとOG以外の部分は、Copy, NAND, Up, Down, Tゲート及びてゲートから構成できる。 (証明終)

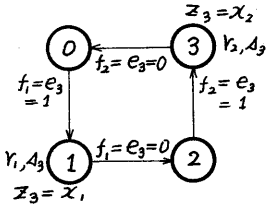
5. 万能な集合

前節までにおいて、万能な集合を見い出したが、この集合のも要素の種類をさらに減らすことを考える。

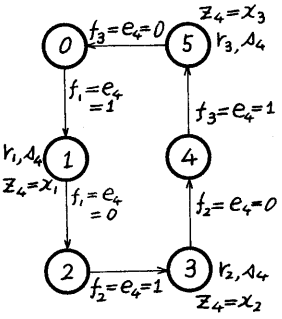
[定理6] 任意のネットワークは(4,3)の集合で実現できる。

(証明) Up, Down, Tゲート, てゲート, IGを兼ねるも要素としてGを考える。Gは図13 a)に示すように、2入力1出力のも要素であり、制御状態0においてc入力に到着したパケットは表1のように解釈される。ここでXは0又は1のいずれでもよいことを表わす。

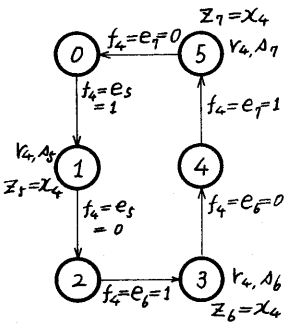
このGから、図13 b), c)に示すようにしてUp, Downを実現できる。



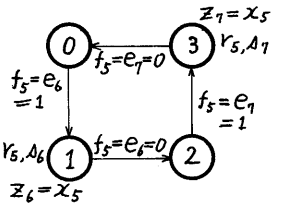
a) IM の仕様



c) OM の仕様

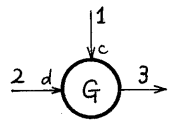


b) IB の仕様

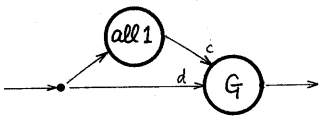


d) OB の仕様

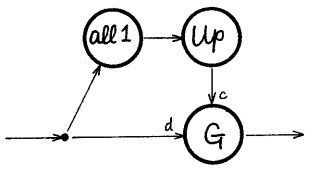
図12. IM, IB, OM, OB の仕様



a) G 要素



b) Up の実現



c) Down の実現

図13. G要素による Up, Down

次に、1入力1出力の3つの関数 g_1, g_2, g_3 を用意する。但し、g_1 は $(0X\cdots X)$ に $(0\cdots 0)$ を対応させ、$(1X\cdots X)$ に $(0\cdots 01)$ を対応させる関数である。g_2 は $(0X\cdots X)$ に $(0\cdots 010)$ を対応させる関数である。g_3 は $(110\cdots 0)$, $(10\cdots 0)$, $(0\cdots 0)$ のそれぞれに $(0\cdots 0100)$, $(0\cdots 0101)$, $(0\cdots 0110)$ を対応させる関数である。これらの関数をC入力に接続して、Tゲート, Tゲート, IGが実現される。

従って、Copy, NAND, G, OGの4種類のセ要素から任意のネットワークが実現できる。 (証明終)

又、G要素の Up, Down の機能だけを使う場合には、関数要素として実現でき、Upからセゲートまでの機能を使う場合には直接的なセ要素として実現できるので次の系が成り立つ。

[系1] 任意の関数要素に対して、同じ縦相関数を持つネットワークが、(3, 3) の関数要素の集合から実現できる。

[系2] 任意の直接的なネットワークに対して、同じ縦相関数を持つネットワークが、(3,3) の直接的なセ要素の集合から実現できる。

表 1. G要素のC入力の解釈

C入力	解釈
$(1\cdots 1)$	Up
$(1\cdots 10)$	Down
$(0\cdots 0)$	Tゲート $(0X\cdots X)$
$(0\cdots 01)$	〃 $(1X\cdots X)$
$(0\cdots 010)$	Tゲート $(0X\cdots X)$
$(0\cdots 011)$	〃 $(1X\cdots X)$
$(0\cdots 0100)$	IG $(110\cdots 0)$
$(0\cdots 0101)$	〃 $(100\cdots 0)$
$(0\cdots 0110)$	〃 $(0\cdots 0)$

実際にモジュール化又はIC化する場合には1つのセ要素とその出力ド要素を1つのモジュールとして実現することが考えられる。そのとき、上の集合の濃度は、そのままモジュール又はICの種類の数を与えることになる。

又、外部から配線によって任意の定数を供給することを許した場合、(2,4) 又は (1,5) で万能な集合を示すことができる。

これらの集合が、実際に使い易いかどうかは別問題であり、例えばメモリなどの、よく使われるものについてはある程度まとまった機能のものを用意する必要がある。しかし、どのような集合を用意する場合でも、それが万能である

ためには、本稿で示した集合の機能を実現できることが要求される。

又、Kellerが非同期回路について示した、(5,5)又は(3,7)で全てのモジュールが構成できるという結果と比較することは興味深い。

6、おわりに

データフロー計算機を、データフローグラフとして実現するために、有向枝と節点をP要素とT要素としてモデル化し、特に(4,3)で万能な集合等を示した。これらは、IC化する場合にどのようなものを用意したらよいかを判断するためのよい討金石となる。

各要素の具体的な回路については別の機会に譲るが、P要素と直接的なT要素は容易に実現できる。しかし、IGとOGについてはクリティカルなタイミングから生ずる、メタステーブルオペレーション(MSO)の問題を考慮する必要がある。

パケットを交換することによって動作する限り、どんなデータフロー計算機でも本稿の方法で構成することができる。ソフトウェアの問題をも含めた具体的な構成上の問題については、今後の課題とする。

謝辞

熱心に御討論いただいた野口研究室の諸氏に感謝します。

文献

(1) Dennis,J.B. and Misunas,D.P.:"A preliminary architecture for a basic data-flow processor",Comp. Struc. Group Memo. 102,Project MAC,MIT(1974)

(2) Amikura,K.:"A logic design for the cell block of a data-flow processor",TM-93, Laboratory for Computer Science,MIT(1977)

(3) 宇都宮公訓：“データフロー計算機”, bit, vol 12, no.2～no9.(1980) 共立出版

(4) 松崎 稔：“コンピュータ構造を変革するデータフローコンピュータの動向”, 日経エレクトロニクス(1979.5.28-6.11)

(5) Keller,R.M.:"Towards a theory of universal speed-independent modules",IEEE,Trans. Comp.vol.C-23,no.1,pp.21-33(1974)

(6) Marino,L.R.:"General theory of metastable operation",IEEE Trans. Comput. vol.C-30, pp.107-115,Feb.1981.