ストレージバス結合並列計算機

A Paralell Computer with Storage Bus

有田　五次郎　　兼田　荒三　　村上　健一郎

Itsujiro Arita　　Genzo Kaneda　　Ken-ichiro Murakami

九州大学　工学部　情報工学科

The Department of Computer Science and Communucation Engineering,Kyushu University

1. はじめに

高多重並列計算機における大きな問題は、オペレーション間の同期と，データアクセスの競合である。最も一般的な並列処理機械であるMIMD型並列計算機は，この2つの問題のため高多重化は困難とされてきた。

ここで述べる並列計算機は，待ちなし並列プログラムの概念を用いた同期ハードウェアを持つストレージバスの概念によってアクセス競合を減少させた，高多重化可能な並列計算機である。

2. ストレージバス

ストレージバスは並列度が増加した時にアクセスタイムは多少増加するが，アクセス競合は増加せず，しかもハードウェアの増加がCPUの数に比例するものである。

2.1　3次元8ノードストレージバス

3次元8ノードのストレージバスの構成を図1に示す。

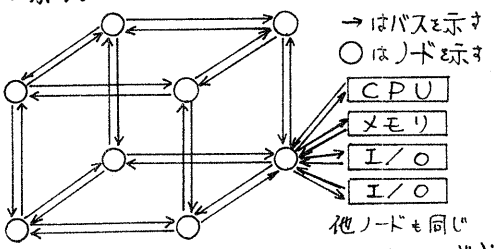


図1　3次元8ノードストレージバスの構成

ノードから出ている7個のバスすなわち4個のデバイス及び他の3個のノードへのバスは同一のバスである。従って、この並列計算機では同時に異なった8個のデータ転送バスが存在する事になる。

ノードの構成を図2に示す。ノードから外へ出るストレージバスは、他の3個のノードのBUF$_x^c$, BUF$_y^d$, BUF$_z^e$（ここまで）計3ケを経由して終わりとなる。

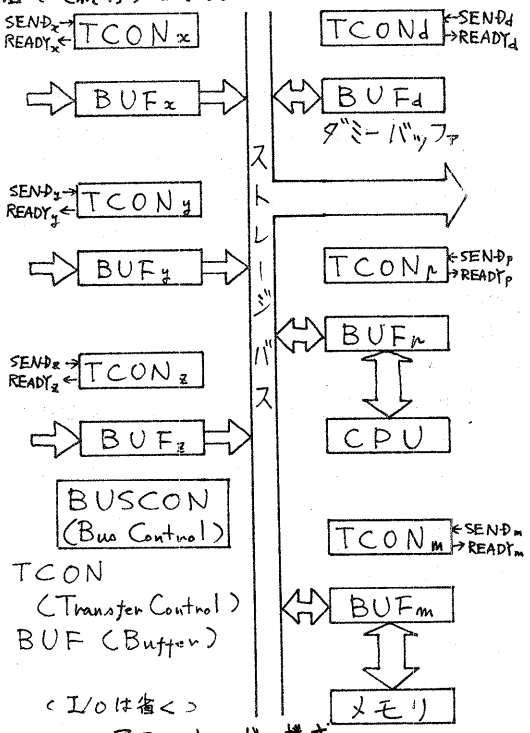


図2　ノードの構成

ストレージバス上のデータ及びコントロール信号を表1に示す。各ノードには図3に示す様に各ノードにシステムアドレス，各デバイスにデバイスアドレスが付けられている。

SSAはストレージバスを介してデータを転送する時の出発地ノードのシステムアドレスであり，DSAは目的地ノードのシステムアドレスである。

SDA，DDAはそれぞれ出発地，目的地のデバイスアドレスである。

PCNTはBUF_n, BUF_mからストレージバスにデータが出力される時にクリアされ，各ノードのBUF_x, $_y$, $_z$を通過する度に1加算される。この値が，最も離れたノード間の距離以上になるとTCONにおいて，そのストレージバス上のデータは抹消される。BUF_dでは，PCNTは何も加算されない。

SENDは前段のノードからストレージバス上にデータが出力されている事を示し，READYはBUFが空である事を前段のノードに知らせる，ストレージバスのコントロール信号である。

データ	
MA (Memory Address)	15 bit
M (Modifier)	16 bit
MDP (Memory Data Parity)	1 bit
R/W (Read or Write)	1 bit
TAS (Test and Set)	1 bit
SSA (Source System Address)	6 bit
SDA (Source Device Address)	2 bit
DSA (Destination System 〃)	6 bit
DDA (Destination Device 〃)	2 bit
コントロール	
SEND	1 bit
READY	1 bit

表1 ストレージバス上のデータ及びコントロール

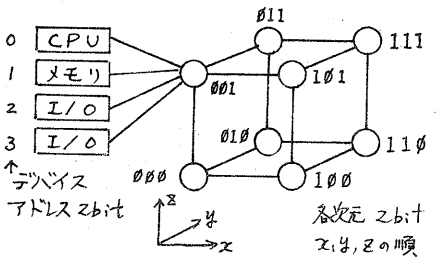


図3 システムアドレス・デバイスアドレス

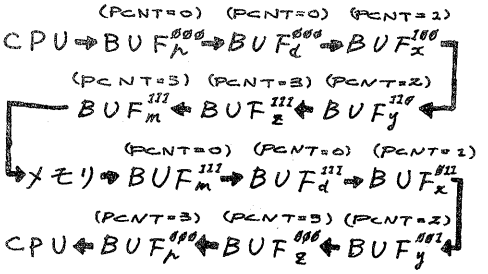
2.2 データ転送

このシステムにおけるデータ転送では，ノード間の距離が同じであれば，どの経路を通っても構わない。(ルーチング) 図2，図3を用いて実際のデータ転送の流れを示す。

1) CPUがプログラム(自ノードのメモリ)をフェッチする場合。

CPU → BUF_n → BUF_m → メモリ
CPU ← BUF_n ← BUF_m ←

2) システムアドレス000のノードのCPUがシステムアドレス111のノードのメモリをアクセスする場合。



右上の添字はシステムアドレスを示す。

データ転送可能な次元の次段のノードがすべてREADYである時には x, y, z の次元の順に優先度が低くなる。上記の2)は6通りある経路のうちの1例である。

2.3 ダミーバッファ

1つの次元にノードが3個以上あるシステムの場合，非同期転送を行なうとデッドロックを起こす。これを防ぐ為に，各次元に1個以上の余分なバッファを持つ必要がある。このシステムでは各ノードに1個のダミーバッファを設けている。

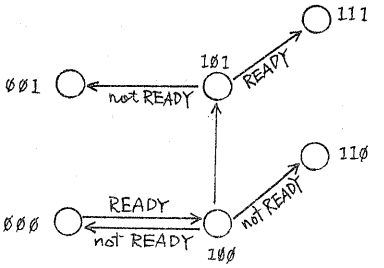


図4 ダミーバッファの効用

x, y, z の方向の転送においてその行先がすべてnotREADYである時に，このダミーバッファにデータがストアされる。そして，notREADYのどれかがREADYになると，その次元の次段のノードへデータが転送される。

このダミーバッファには他に次にあげる2つの効用がある。

1) CPUが他ノードのメモリをアクセスしようとする時はこのダミーバッファを介してデータを転送する。これによりCPUとストレージバスをつなぐTCONのハードウェア量が軽減する。

2) 図3から抜き出した図4の状態において次の様なデータ転送が行なわれた場合を考える。

a) ノード100のCPUがノード001のメモリをアクセスの為にノード101へデータを転送する。

b) ノード000のCPUがノード111のメモリをアクセスする為にノード100へデータを転送する。（これはxの優先度が最も高い為である。）

この時に，ノード100からノード101へ入ったデータがBUF_x^{101}に入ると，ノード000からノード100に入ったデータはノード101のBUF_z^{101}を通ってノード111へ行く事ができ転送が速くなる。

すなわち，転送待ちのデータの一時退避場所として使用でき，他の転送可能データの妨害とならない。

また，これが有効に使えるのは，ノードの通過時間がCPUのサイクルタイムメモリのアクセスタイムに比べて短いからである。もし，ノードの通過時間が同程度以上であれば，この効用は薄れてくる。

2.4 転送制御及びルーチング回路

TCONはデータ転送の制御を行なう回路で前段のノードからのデータの受け取り，次に転送すべきノード又はCPU，メモリ，ダミーバッファの決定そしてデータの送り出しを行なう。

BUFはラッチとバッファゲートから成る。

図5にTCONとBUF(x,y,z,d)のブロック図，図6に動作図を示す。

TCONは前段のノードからのSENDを受け付けた時に，果して転送可能な次元の次段のノードがREADYであるかどうかを調べる前に，BUSCONに対してREQを出してしまう。そこで，もしすべての該当するREADYがすべてnotREADYであり，どこにもデータが転送できない時，またたとえREADYであってもBUSCONの優先順位の為に他の同一ノード内のTCONにバス使用許可をとられてしまい，すべてnotREADYになった時に一度受け取ったバス使用許可AVLを返上しなければならない。その為にCANという信号が出力される。これはいわゆる先行制御であり，それより優先順位の低いREQが同時に出されていた時のみ妨害となる。

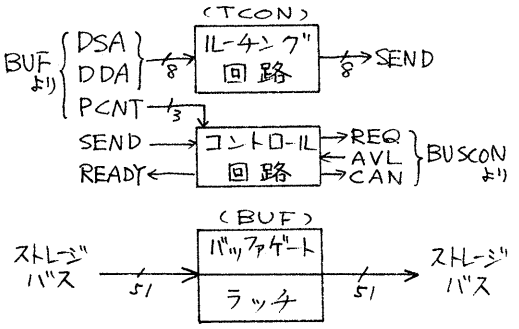


図5 TCONとBUF(x,y,z,d)ブロック図

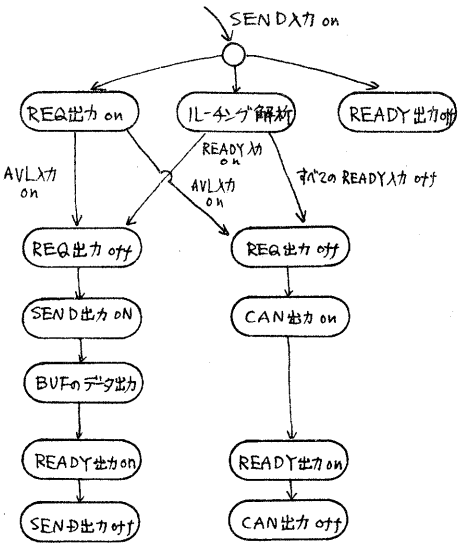


図6 TCONとBUFの動作図

T_k ($k=x,y,z,d,u,v,n,m$) を BUF$_k$ へのデータ転送可能を示す信号とする。FSA をノード固有のシステムアドレスとする。デバイスアドレスを

ϕ = n (CPU)
1 = m (メモリ)
2 = u (I/O)
3 = v (I/O)

とする。

その時、ルーチンブロックが今まで述べたデータ転送を行なう為には、下記の論理式が成り立たなければならない。

$$T_x = \overline{DSA_0 \oplus FSA_0} \cdot \overline{DSA_1 \oplus FSA_1}$$
$$T_y = \overline{DSA_2 \oplus FSA_2} \cdot \overline{DSA_3 \oplus FSA_3}$$
$$T_z = \overline{DSA_4 \oplus FSA_4} \cdot \overline{DSA_5 \oplus FSA_5}$$
$$T_d = T_x + T_y + T_z$$
$$T_n = \overline{T_x} \cdot \overline{T_y} \cdot \overline{T_z} \cdot \overline{DDA_0} \cdot \overline{DDA_1}$$
$$T_m = \overline{T_x} \cdot \overline{T_y} \cdot \overline{T_z} \cdot DDA_0 \cdot \overline{DDA_1}$$
$$T_u = \overline{T_x} \cdot \overline{T_y} \cdot \overline{T_z} \cdot \overline{DDA_0} \cdot DDA_1$$
$$T_v = \overline{T_x} \cdot \overline{T_y} \cdot \overline{T_z} \cdot DDA_0 \cdot DDA_1$$
$$SEND_x = T_x \cdot READY_x$$
$$SEND_y = T_y \cdot READY_y \cdot \overline{SEND_x}$$
$$SEND_z = T_z \cdot READY_z \cdot \overline{SEND_x} \cdot \overline{SEND_y}$$
$$SEND_d = T_d \cdot READY_d \cdot \overline{SEND_x} \cdot \overline{SEND_y} \cdot \overline{SEND_z}$$
$$SEND_n = T_n \cdot READY_n$$
$$SEND_m = T_m \cdot READY_m$$
$$SEND_u = T_u \cdot READY_u$$
$$SEND_v = T_v \cdot READY_v$$

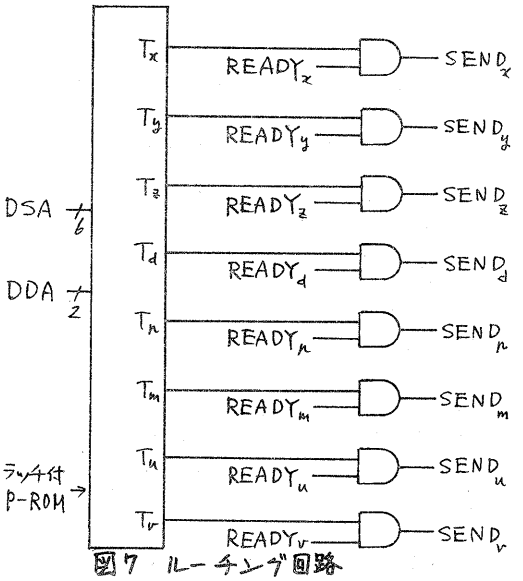


図7 ルーチンブロック回路

これらの論理式を実現する為に図7の回路を用いる。ラッチ付P-ROMで一部を処理しているのがこの回路の特徴である。

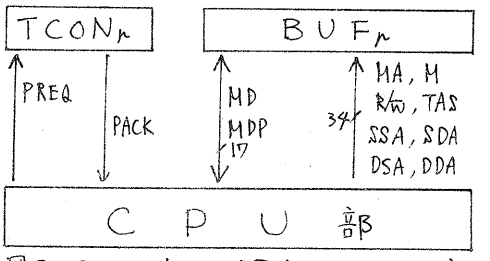


図8 TCON$_n$ 及び BUF$_n$ と CPU部のインタフェイスブロック図

2.5 プロセッサインタフェイス回路

TCON$_n$ 及び BUF$_n$ は CPU部とストレージバスのインタフェイスを受け持つ。TCON$_n$ 及び BUF$_n$ と CPU部のインタフェイスを図8にブロック図で示す。

ここで M (Modifier) bit は ϕ の時 CPU が自ノードのメモリを、1の時 CPU が他ノードのメモリをアクセスする事を示す。この M bit の使用と、CPU・メモリが他ノードへデータを転送する時にダミーバッファを使用する事、この2つによって前述のルーチンブロック回路が TCON$_n$ から省かれる。これは TCON$_m$ についても同様であり、ハードウェア量の軽減の為に行なわれている。

図9に TCON$_n$, BUF$_n$ のブロック図を示す。

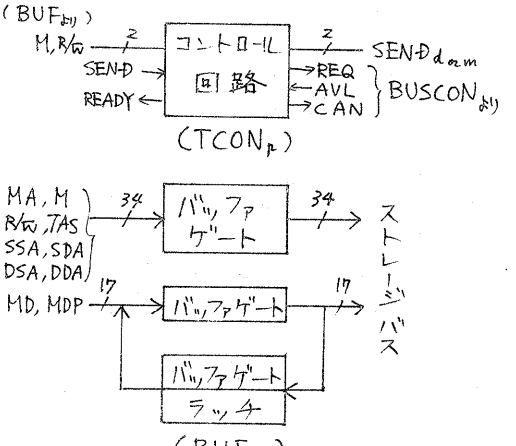


図9 TCON$_n$, BUF$_n$ ブロック図

TCONの動作を図10に示す。TCON$_n$はPREQによって起動される。そしてストレージバスを占有してデータを送り出した後、READYの状態でSEND（メモリからの応答）待ちとなる。

ここで、メモリからの応答（リード，ライト共）が返ってきた時に、果してそのデータが本当に目的ノードからの応答なのかどうかを確認する必要がある。その為にはSSA，SDAとDSA，DDAを比較しなければならないが，ここではTCON$_n$のハードウェア量を軽減する為に省略する。メモリ側からSSA，SDAは送られてこない。

2.6 メモリインタフェース回路

TCON$_m$及びBUF$_m$はメモリ部とストレージバスのインタフェイスを受け持つ。TCON$_m$及びBUF$_m$とメモリ部のインタフェイスを図11にブロック図で示す。

メモリ部はメモリセルのデータを出力するだけであるからDSA,DDA，SSA，SDAはメモリ部には入力されない。

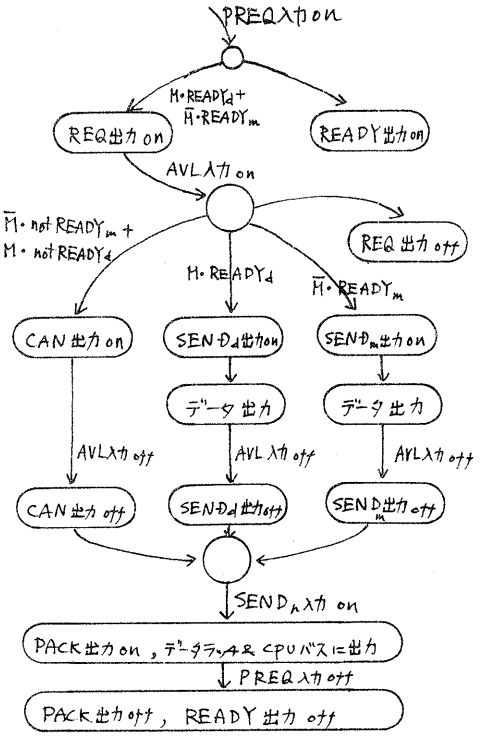


図10 TCON$_n$の動作図

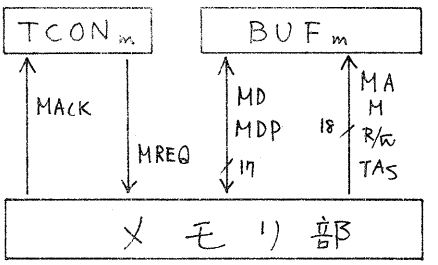


図11 TCON$_m$及びBUF$_m$とメモリ部のインタフェイスブロック図

図12にTCON$_m$とBUF$_m$のブロック図を示す。TCON$_n$と同じくルーチンプ回路が省略されている。

図13にTCON$_m$の動作図を示す。TCON$_m$はSEND$_m$によって起動される。そして，MREQをメモリ部に出力し，メモリのアクセスを開始させ，メモリ部からのMACK入力によってデータを得た後 ストレージバスを占有しCPUへデータを転送する。

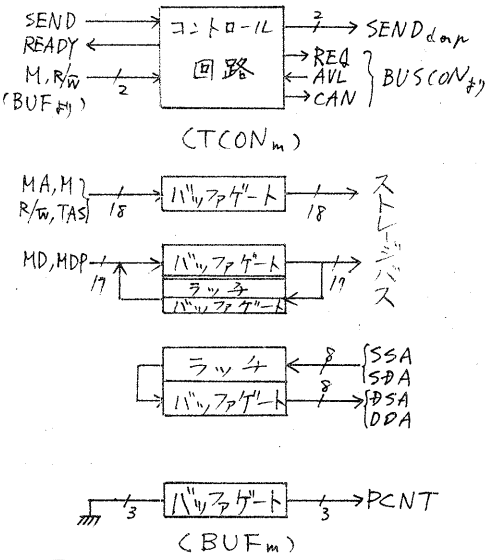


図12 TCON$_m$とBUF$_m$のブロック図

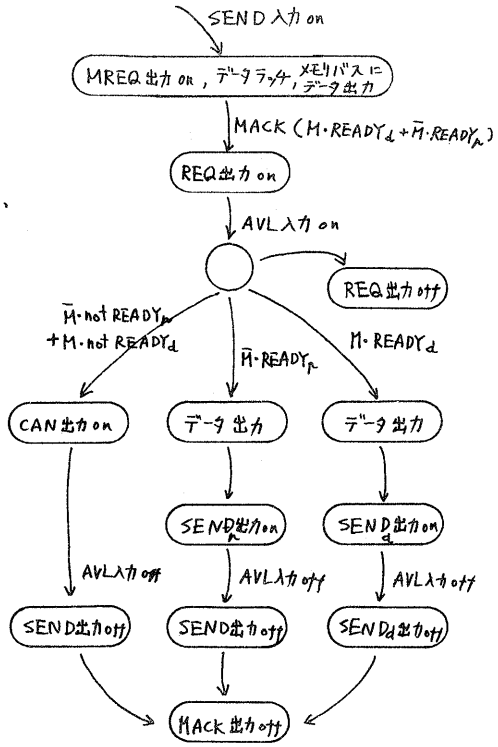


図13 TCON$_m$の動作図

2.7 バスコントロール回路

BUSCONはTCON$_k$ (k=ア, イ, ウ, エ, オ, カ) から出されるストレージバス使用要求REQを優先度 ($n, m, \ell, x, y, z, u, v$ の順に低くなる) を考慮して調停する。ブロック図を図14に示す。図15に動作図を示す。クロックを用いたサンプリング方式であるので、多少時間のロスが生じるがラッチを2段持っているので、引き続いてREQがくる時には単位時間当たりの処理数は多くなる。いわゆるパイプライン処理と同様である。

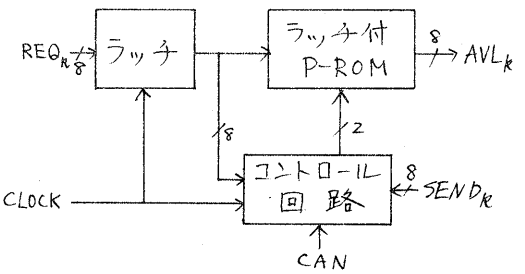


図14 BUSCONのブロック図

2.8 転送時間

BUFからBUFへデータが転送されるのに要する時間は，構成素子にSタイプのTTLを使用した場合に素子のtyp.値で計算して250ns程である。ただし、BUSCONがサンプリング方式なのでこれは平均値である。

従ってプログラムのアクセスの時のデータの流れは2.2-1)より、BUFからBUFへの転送が，BUF$_n$→BUF$_m$，BUF$_m$→BUF$_n$の2回であるから500nsメモリのアクセスタイムが延びる事になる。これはCPUのメモリサイクルの約25%増になる。この程度であれば、CPUとメモリの間に特別なバスを設けた場合のハードウェア量の増加と相殺できるであろう。

他ノードのメモリをアクセスする場合、システムの次元数をm，1次元のノード数を等しくaとすると、そのシステム内の最も離れたノード間の遅延時間Tは

$$T = 250(2+m\cdot a+k) \quad \text{——(1)}$$

で表わされる。ただしkは途中に通過したダミーバッファの個数である。

ここでシステムのノード数Nは

$$N = a^m \quad \text{——(2)}$$

で表わされる。

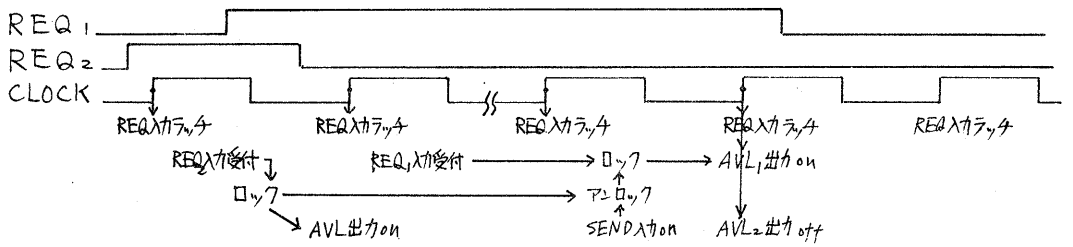


図15 BUSCONの動作図

(1)式と(2)式を比較して、ノード数が指数関数であるのに、最大遅延時間Tは1次関数でしかふえないのが良くわかる。

実際，3次元8ノードの場合(1)式より

$T = 250(2+3\times2)$ $(k=0)$

$= 2.0\,\mu s$

3次元64ノードの場合

$T = 250(2+3\times4)$

$= 3.5\,\mu s$

4次元81ノードの場合

$T = 250(2+4\times3)$

$= 3.5\,\mu s$

4次元256ノードの場合

$T = 250(2+4\times4)$

$= 4.5\,\mu s$

にしか過ぎない。これだけメモリのアクセスタイムが延びるだけである。

3. CPU・メモリ

3.1 待ちなし並列プログラム

並列プログラムはグラフ的に図16(a)のように表わされる。各ノードはオペレーション，各アークは制御の移動を表わし、同一列のノードは同一プロセッサで実行される。

このプログラムは適当な変換によって(b)のような木構造に変換できる。(b)はプロセッサが first come first serve で処理している時は(a)と同価であり，自然に同期がとれている。このようなプログラムを待ちなし並列プログラムと呼び制御が渡ると一度に実行されるオペレーションの集合をタスクと呼ぶ。

並列プログラムを実行するプロセッサに要求される命令は次の2つである。

1) 他プロセッサ上にタスクを発生させるための命令
2) 1つのタスクを終了後，次のタスクを実行するための命令

これらは比較的簡単なハードウェアで実現でき，従って同期のオーバヘッドも少ない。この並列計算機では図16(b)の並列プログラムを実行させる事を目的としている。

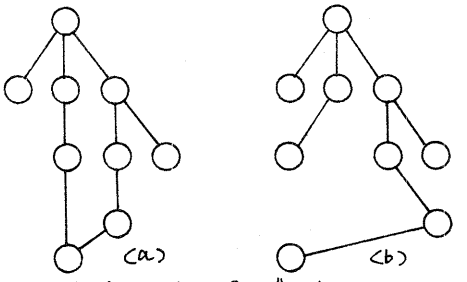


図16 並列プログラム

3.2 CPU部・メモリ部の構成

今回の並列計算機製作においては、ハードウェア量の軽減に主眼を置いた。なぜなら，このハードウェア量の多少がシステムが実現するかどうかの鍵を握るからである。その為にマイクロプログラム方式の東芝製TOSBAC40-Lという5チップ構成のCPUを採用した。表2にその主な仕様を示す。又、TAS(Test & Set)の端子がチップに出ているのもメモリの排他制御に非常に都合が良い。

図17にCPU部，メモリ部のブロック図を示す。通常のCPU・メモリに比べて外部レジスタが付加されているに過ぎない。表3に外部レジスタの構成を示す。

EPIRはそのノードのメモリに登録されている実行可能なタスクのIDを示す。タスクは最高8個(8bit)まで可能である。

CCRはソフトウェアでコンソール制御を行なう為のもので，ステップ動作，コンソール割り込みが可能である。

FIFOSTATUSはタスク管理ハードウェアとして使われるFIFO(タスクキュー)のEMPTY，FULLを検出する為のレジスタで，リードでロック，ライトでアンロックとなっている。

RSAは現在実行中のタスクを起動したノードのシステムアドレスを示す。

PIDは現在実行中のタスクのIDを示す。

NPSWは新しく実行するタスクのPSWである。

RMAは現在実行中のタスクの戻り先のメモリアドレスを示す。

以上の4つはFIFOから4回プルする事によって得られ、RSA，PID，RMAはCPU側のRSAR，PIDR，RMARに格されます。

3．3　タスク管理ハードウェア

この並列計算機ではタスク管理の為に各CPUがタスクキュー(FIFO)を持っている。1つのタスクは，RSA，PID，RMA，NPSWで表わされる。タスクキューには10個のタスクが登録できる。

このタスクキューを操作する為の拡張機械語命令（マイクロ命令による）は，PB（Parallel Branch），EXT（Exchange Task），CKT（Check Task）の3つがあり，更にストレージバス使用の為にSDSA（Set Destination System Address）命令がある。これらにより並列プログラムの実行が可能になる。

PB命令は，他プロセッサ上にタスクを発生させる命令で，そのプロセッサのFIFOにタスクを書き込む。

EXT及びCKT命令はFIFOからタスクを取り出し，そのタスクがEPIRに登録されているかどうかを調べ，登録されていれば実行する。登録されていなければ，それが共有タスク（I/Oルーチン等）かどうかをNPSWをみて調べ，そうであれば実行する。そうでなければそのタスクはアボートされたタスクと見なされ実行されない。EXTは，FIFOがEMPTYの場合にタスクが新しく書き込まれるまで待つがCKTは待たずに次の命令に進む。

図18に各命令のマイクロ命令による処理のフローチャートを示す。

演算論理	2進並列，2の補数表示，固定/浮動小数点表示
命令数	126種
アドレス空間	64KByte
アドレス指定	全アドレス直接指定，インデックス修飾
レジスタ	ゼネラルレジスタ　16bit×16個 （内15個は，インデックスレジスタ兼用） 浮動小数点レジスタ　32bit×8個 （主記憶上 X'0' ～ 'F'）
制御方式	マイクロプログラム制御
語形式	命令語　16bit(RR,SF型)，32bit(RS,RX型) 固定小数点数　1,8,16,32 bit/data 浮動小数点数　32bit/data 制御語(psw)　32bit
割り込み	内部割り込み　6レベル 外部割り込み　基本外部割り込み 又は，オートマチックI/O

表2　TOSBAC40-L仕様

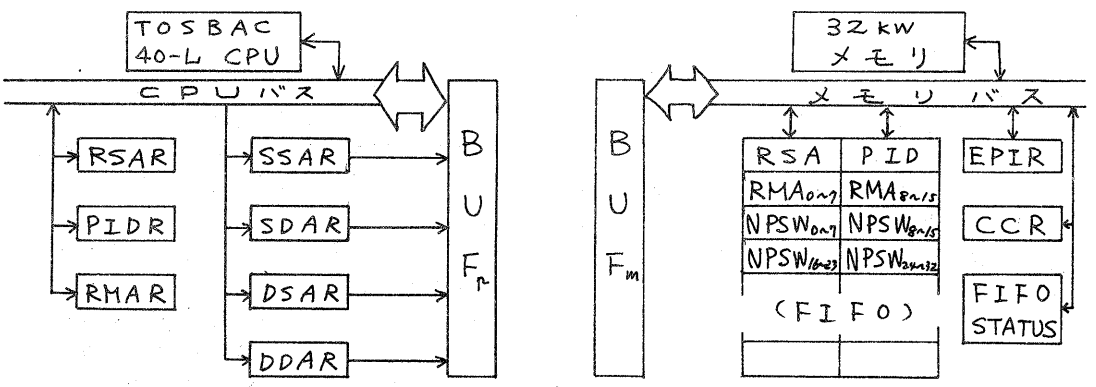


図17　CPU部・メモリ部

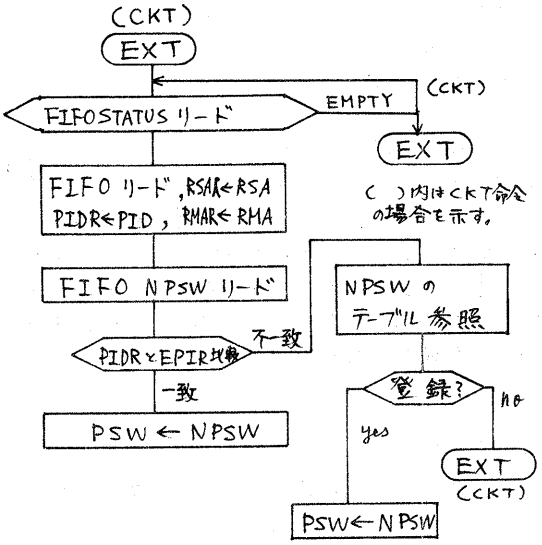


図 18 (a) EXT. CKT命令フローチャート

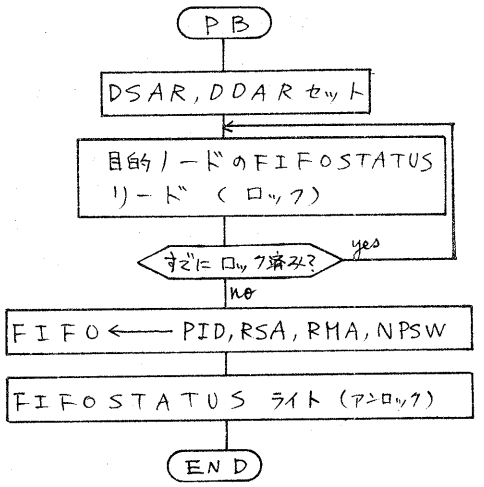


図 18 (b) PB命令フローチャート

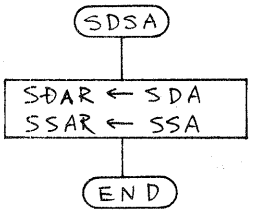


図 18 (c) SDSA命令フローチャート

4. 実装

表4にユノード分について使用ICの個数を示す。使われているICはNANDゲート, フリップフロップ, ラッチ付バッファゲート, バッファゲートがほとんどであり, 特殊なICはラッチ付P-ROMぐらいである。

図19にボードの形状, 図20にコネクタの接続図を示す. ボード4枚でユノード(I/O除く)を構成し, 50cm × 45cm × 10cm のラックに収納される。そしてそのラックに入る信号線は 50pin, 20pin フラットケーブルがそれぞれ3本出ていく信号線が 50pin, 20pin フラットケーブルでそれぞれ4本である。

$TCON_{x,y,z}$	15ヶ
$TCON_{\ell}$	14ヶ
$TCON_{m}$	8ヶ
$TCON_{p}$	7ヶ
$BUF_{x,y,z,\ell}$	7ヶ
BUF_{m}	10ヶ
BUF_{n}	11ヶ
BUSCON	6ヶ
CPU部	45ヶ
メモリ部	53ヶ
	計 176ヶ

表4 使用ICの個数

アドレス	bit	
20	8bit	EPIR (Execution Process ID Registen)
21	3bit	CCR (Console Control Registen)
BE	8bit	SSAR (6bit), SDAR (2bit)
BF	8bit	DSAR (bit), DDAR (2bit)
C0	5bit	FIFO STATUS
C2	8bit	(RSA or $RMA_{0\sim7}$ or $NPSW_{0\sim7}$ or $NPSW_{16\sim23}$
C3	8bit	PID or $RMA_{8\sim15}$ or $NPSW_{8\sim15}$ or $NPSW_{24\sim32}$) FIFO
C4	8bit	PIDR (Process ID Registen)
C5	8bit	RSAR (Return Syatem Address Registen)
C6,C7	16bit	RMAR (Return Memony Address Registen)

表3 外部レジスタの構成

5. おわりに

以上述べた並列計算機は現在プリントパタンを描き終わり、エッチング、穴開けを外部に発注しているところである。

この並列計算機上のソフトウェアはまだほとんど未開発であり、ハードウェアが出来次第取り掛からなければならない。

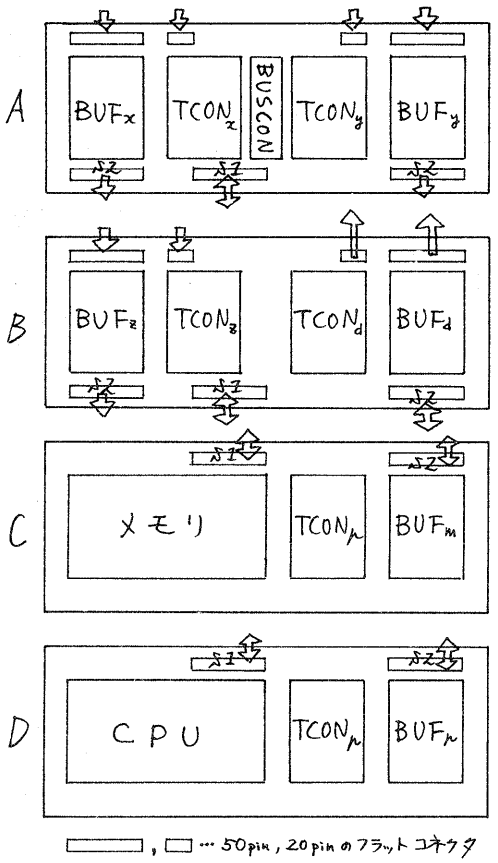


図19 ボード図

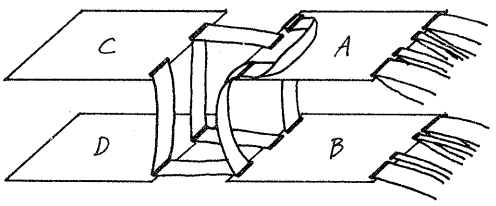


図20 コネクタ接続図

IPLの為に、I/Oとしてホストコンピュータをいずれかのノードに接続し、各ノードのメモリにプログラムをロードするつもりである。

この並列計算機では、シーケンスコントロール、アドレスデコーダ、ラッチ付P-ROMを多く用いている。これがハードウェア量の軽減に大きく貢献をしている。特にCPU、メモリ部において著しい。しかし、P-ROMのアクセスタイムが50nsとSSI・MSI TTLに比べて遅いので、高速を目指した所へは応用できない。

(参考文献)

1. 有田「並列処理システムの一構成法(I) ストレージバス」
S54 電気四学会九州支部大会論文集
2. 有田, 兼田「並列処理システムの一構成法(II) 3次元ストレージバスの実現」
S54 電気四学会九州支部大会論文集
3. 有田「MIMD型高多重複合計算機-MMCS(I)システムアーキテクチャ」
S55 電子通信学会全国大会
4. 有田, 兼田, 村上「MIMD型高多重複合計算機MMCS(II)ストレージバス」
S55 電子通信学会全国大会
5. 有田, 村上, 兼田「MIMD型高多重複合計算機MMCS(III)CPU,メモリ部」
S55 電子通信学会全国大会
6. 村上「ストレージバス結合による複合計算機システムの設計」
九州大学 工学部 昭和54年卒論
7. 「TOSBAC-40L アプリケーションガイド LSI」東芝