EC85－64

ＰＩＥの構造メモリ試作ハードウェアの設計について

A Design of Structure Memory Pilot Machine of ＰＩＥ

平田 圭二、 猪股 宏文、 垂井 俊明、 松原 健二、
Keiji Hirata, Hirofumi Inomata, Toshiaki Tarui, Kenji Matsubara,
小池 汎平、 田中 英彦、 元岡 達
Hanpei Koike, Hidehiko Tanaka , Tohru Moto-oka

東京大学 工学部
Faculty of Engineering, The University of Tokyo

１． はじめに

現在我々は高並列推論エンジンＰＩＥ［１］の研究を進めている。ＰＩＥはゴール書き換えモデルに基づき論理型プログラミング言語に内在するＯＲ並列性を最大限に活かしつつ、高速に処理を行うマシンである。

ＰＩＥに構造データ共有方式を導入すると、大規模な構造データが出現した場合でも高い処理能力を維持できることは、ソフトウェアシミュレーションによって確認した［２］。構造データ共有方式を効率良く実現するためには、Ground Instance 格納用の共有メモリとして、構造メモリ（Structure Memory ：SM）の導入が不可欠である［３］。本稿では現在試作中のＰＩＥの構造メモリ試作ハードウェア［４］［５］について報告する。

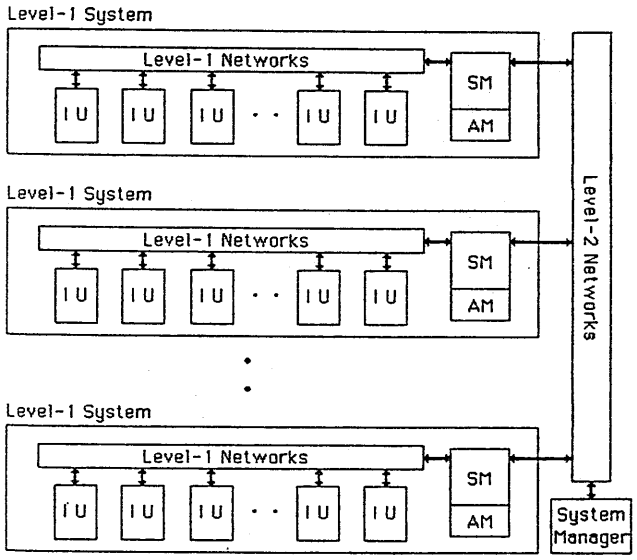


Fig.1 Global Architecture of PIE-II

２． 構造データ共有方式について

２．１ 基本操作

構造データ共有方式とは推論ユニット（ＩＵ）間で転送される基本処理単位（ゴールフレーム：ＧＦ）に含まれる構造データの内、Ground Instance（ＧＩ）の部分のみを切り分け、メモリに格納し、ＧＦはそれを指すポインタだけ持ち運ぶ方式である。ポインタだけ持ち運び、ＯＲ並列処理を行う結果としてＧＦ間でのＧＩの共有が生じる。この構造データ共有方式によって縮退、転送時間の短縮を図ることができる。構造データ共有方式を導入したＰＩＥ（ＰＩＥ－Ⅱ）は、１６台のＩＵ毎に１台のSMを置いた、階層構成のアーキテクチャを持つ（図１）。ＧＩの処理のため、SM－ＩＵ間でコマンドやデータの転送などを行う基本操作として以下の４つを定めた。

① Lazy Fetch

単一化時、必要となったSM中の構造データをＩＵに読み出すことをLazy Fetch（ＬＦ）と呼ぶ。ＩＵはＧＩのアドレスをＬＦネットワーク（ＬＦＮ）に送出し、SMはそれに対しＧＩを１ノードだけ返送する（図２）。

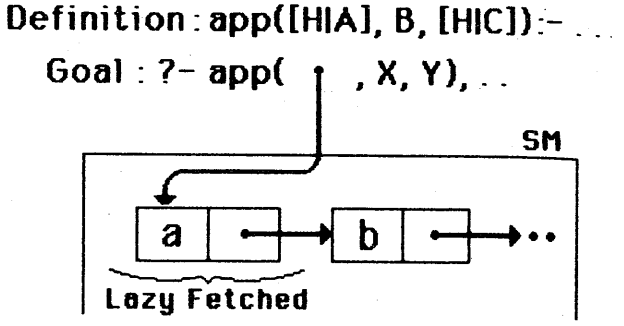


Fig.2 Lazy Fetch

② Ground Instance の切り分け・格納

構造データがＧＩか否かの検査は、縮退時に新ＧＦに含まれる構造データの末端の１ノードに対してのみ行う。従ってＧＩの格納は１ノードずつ実行される（図３）。

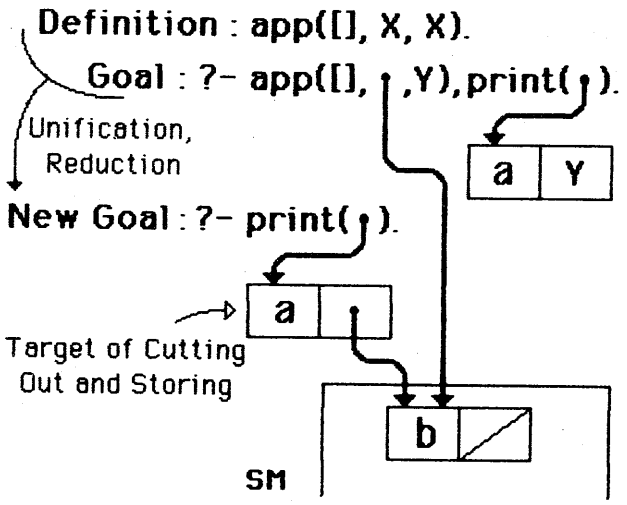


Fig.3 Cutting out and Storing of Ground Instance

③ 参照カウント命令

新ＧＦを生成する時に参照カウント（ＲＣ）命令を発行する。ＩＵ内でバッファリングし、最適化を施し、まとめてＳＭに転送する。新たに生成され、ＳＭに格納されるＧＩに対するＲＣ命令はＩＵ側で発行せず、ＳＭ内で自動生成する（図４）。

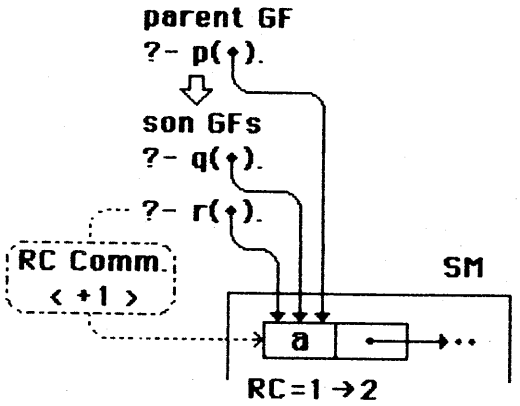
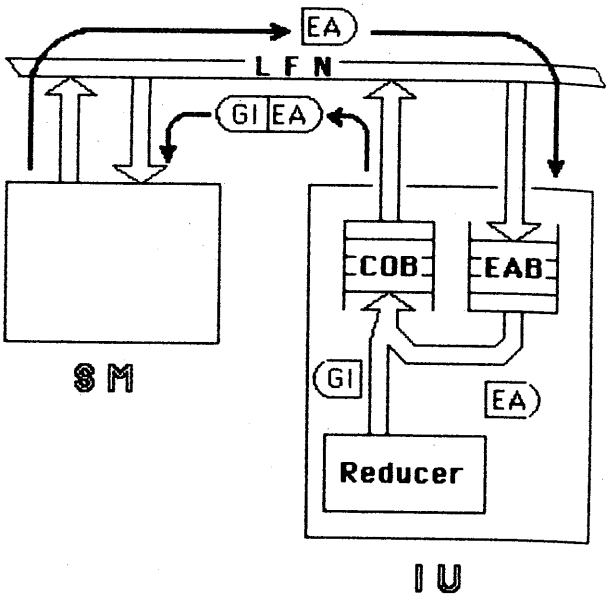


Fig.4 Processing of RC Command

④ ＳＭ空き番地管理

縮退時ＵＰからＳＭにＧＩを格納する際に、ＧＦがその格納番地を知る必要があるので、ＳＭはあらかじめ空き番地（Ｅmpty Ａddress：ＥＡ）を各ＩＵにＬＦＮを通して配布しておく。ＩＵは縮退時、格納したいＧＩとＥＡを組にしてＳＭに送出する（図５）。



LFN: Lazy Fetch Network
COB: Command Output Buffer
EAB: Empty Address Buffer

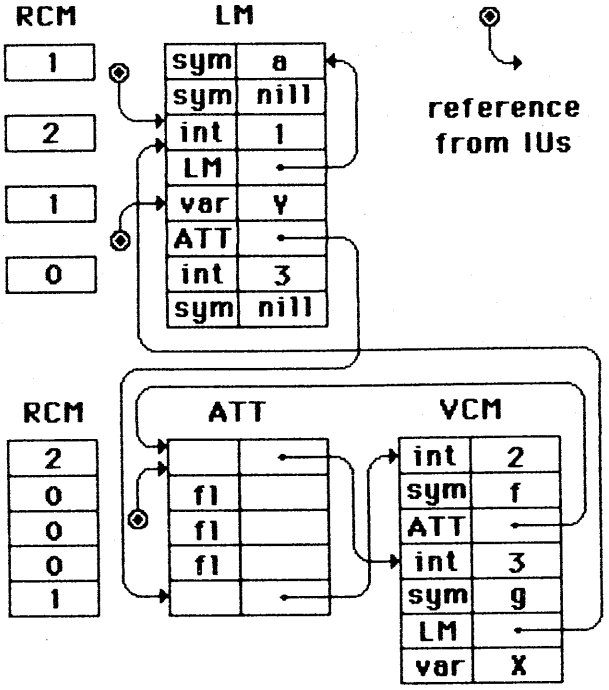
Fig.5 Management of Empty Address

これら４操作は複数のＩＵやＳＭで独立に実行されるので、ＳＭへのアクセス競合が生じることもあり得る。

２．２ 構造メモリの論理構造

ＳＭでは、１語でシンボルや整数等を表現し、２語でリスト１ノードを表現し、複合項は引数の個数＋２語で表現することとした。従ってガーベジコレクション（ＧＣ）の際、可変長ノードが存在するため、メモリ領域の圧縮が必要となる。例えばＧＦがメモリ上のノードのアドレスを直接指している場合は、メモリの圧縮を行う度にＧＦ中の全ポインタを書き換えねばならない。これを避けるためにアドレス変換テーブル（Address Translation Table：ＡＴＴ）を導入し、ノードの実体は可変長セルメモリ（Vari-sized Cell Memory ：ＶＣＭ）に格納する（図６）。ＧＦからＧＩノードへは必ずＡＴＴを介してアクセスし、ＶＣＭからの参照も全てＡＴＴを介して行う。これにより、ＶＣＭの圧縮を行ってもＡＴＴだけ書き換えれば良く、ＳＭ内だけの処理で済む。

この方式では常にＡＴＴを参照する手間が必要であるが、リストノードの場合、ゴミノードは自由リストとして管理すればメモリ領域を圧縮せずに済み、ＡＴＴを介する必要がなくなる。従ってリストノードに関する限り、リストノード専用のメモリ（Lis



LM : List Memory
ATT : Address Translation Table
VCM : Vari-sized Cell Memory
RCM : Reference Count Memory

fl : free list tag

Fig.6 Logical Image of Structure Memory

t Memory ：LM）に格納し、GFから直接参照する。また複合項の場合でも、ATTを介するアクセスを高速化するために、可変長ノードの長さに関する情報を含むセル（PIEの場合は先頭のセル）だけATT上に置く方法が考えられるが、本試作機ではハードウェア上の制約からこの方法は見送った。

2．3　構造メモリのガーベジコレクション

LMとATTのGCは参照カウンタ方式［8］で行い、ゴミのノードは自由リストで管理する。LMとATTにはエントリに対応した参照カウントメモリ（Reference Count Memory ：RCM）が設けてある。参照カウンタ更新によるゼロの伝搬はBreadth Firstの順に行われる。空き番地の要求があった時に、自由リストをたぐり、予め決められた個数分の空き番地を収集する。GIの切り分け・格納は末端の1ノードに対してのみ行われるので、SM中に循環構造が出現することは無い。

VCM上のノードがゴミになるのは、それを指すATTのセルがゴミになった時である。ゴミになったVCMのノードはサイズ毎の自由リストにつながれ管理される［6］。VCM上にノードを割り付ける時は通常、適合したサイズの自由リストから取り出して来るか、又はより大きなサイズのノードから切り出して来るかのいずれかである。しかし、VCMが細分化されすぎると大きなノードが割り付けられなくなってしまう。この場合は新しいVCMをもう1面用意して、そこにコピー・圧縮を行い、大きな空き領域を作らねばならない。この時はATT上のセルとそれに対応するRCMをスキャンしながら、生きているノードのみ新VCMにコピーして行く。この場合、ATTのみ書き換えれば良い。このようにサイズ毎の自由リストで可変長ノードを管理することで、VCMの割り付けが高速に実行でき、VCMのコピー・圧縮というオーバーヘッドの大きい処理をできるだけ遅らせることが可能となる。

3．　構造メモリ試作ハードウェアの全体構成と各部の機能

3．1　全体構成

本試作ハードウェアはIU1台、SM1台という構成を持ち、上記4操作を効率良く実行するための専用ハードウェアをIU側、SM側に各々持つ（図7）。

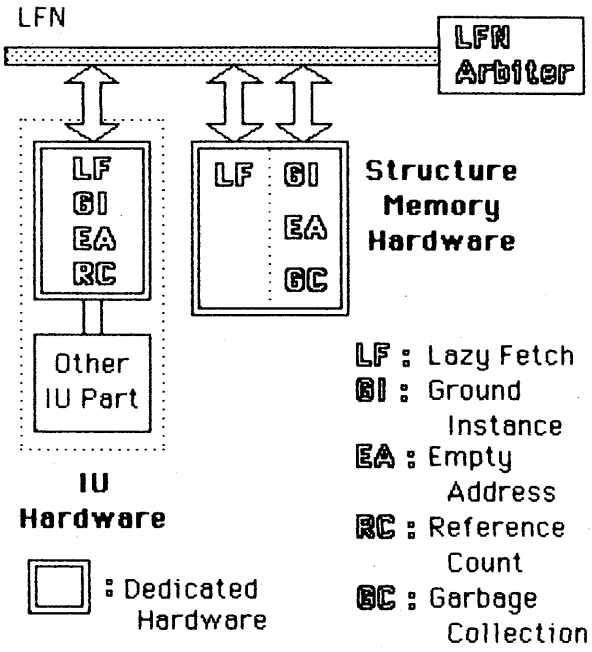


Fig.7 Block Diagram of SM Hardware Simulator

IU側でハードウェア化したのは構造データ共有方式を支援する基本的な部分のみである。即ち単一化プロセッサ（Unify Processor：UP）の内、LF、GI格納、EA管理、RC命令生成を行う部分である。UPの他の部分、定義節メモリ（Definition Memory ：DM）、メモリモジュール（Memory Module ：MM）、アクティビティコントローラ

（Activity Controller ：ＡＣ）はマイクロプロセッサで実装した。ＵＰではＭＭから取り出したＧＦの単一化・縮退を次々と行い、新しく生成されたＧＦを再びＭＭに戻す。例えば単一化の最中にＬＦを行う時は、ＩＵ側の専用ハードウェア内の特定のレジスタに読み出したいＧＩノードのアドレスを書き込み、ＬＦバッファ（ＬＦＢ）に答えが書き込まれるまで待てば良い。ＬＦＢはＳＭのキャッシュのような役割を果たし、読み出して来たＧＩノードを適当なタイミングまで保持する。従って同じＧＩノードを２回以上ＳＭまで読み出しに行く必要はない。

ＳＭにはＬＦを高速に行うためにシーケンサを２つ設けた。各々をＳＭＬＦ側とＳＭＭＡＮ側と呼ぶ。ＳＭＬＦ側ではＬＦ処理のみを行い、ＳＭＭＡＮ側では２．１節の②～④に対応する処理を行う。この２つのシーケンサは同一クロックのもとで独立に動作する。従って、ＩＵからＬＦＮを通してＳＭに転送されるコマンドには、ＳＭＬＦかＳＭＭＡＮのどちらかの宛て先が必要である。

ＬＦＮはバスによって実現した。ＬＦＮアービタに対する要求は次の３通りある。

① ＬＦ要求（ＩＵから）
② コマンドパケット転送要求（ＩＵから）
③ ＥＡパケット転送要求（ＳＭから）

優先度に関しては、①が１番高く、②③は同じである。①の場合、１回の要求で、ＩＵ→ＳＭのＧＩノードアドレスの転送、及びＳＭ→ＩＵのＧＩ１ノードの返送が行われ、その間ＬＦＮは占有され続ける。②③の場合には、１回の要求でＩＵ→ＳＭ又はＳＭ→ＩＵのパケット転送が生じる。②のコマンドパケットは、ＧＩ格納コマンド、ＲＣ命令コマンド、ＥＡ要求コマンドから成り、ＩＵ側から適当なタイミングで送出される。③はＩＵからのＥＡ要求に対して、ＳＭがＥＡパケットを返送する時に生じるものである。

３．２　ハードウェア構成

３．２．１　ＩＵ

本試作機のハードウェア構成を図８に示す。図８において、ＩＵハードウェアは約５００個のＴＴＬＩＣ（殆どＡＬＳタイプ）と２８Ｋ語（１語＝４バイト）のメモリ、ＡＬＵ、２個の連想メモリ（ＮＴＴ厚木通研製ＡＰＬ１Ｂ［７］）等から成り、ボード７枚により実装される。内部バスはバス変換回路を通してマルチバスに接続されており、さらにこのマルチバスには市販の６８Ｋボード、５１２Ｋメモリボードも接続されており、これらでＩＵ１台分をシミュレートする。

３．２．２　ＳＭ

ＳＭハードウェアについては、ＳＭＬＦでは約２００個のＴＴＬＩＣを２枚のボードに、ＳＭＭＡＮでは約４００個のＴＴＬＩＣを５枚のボードに実装

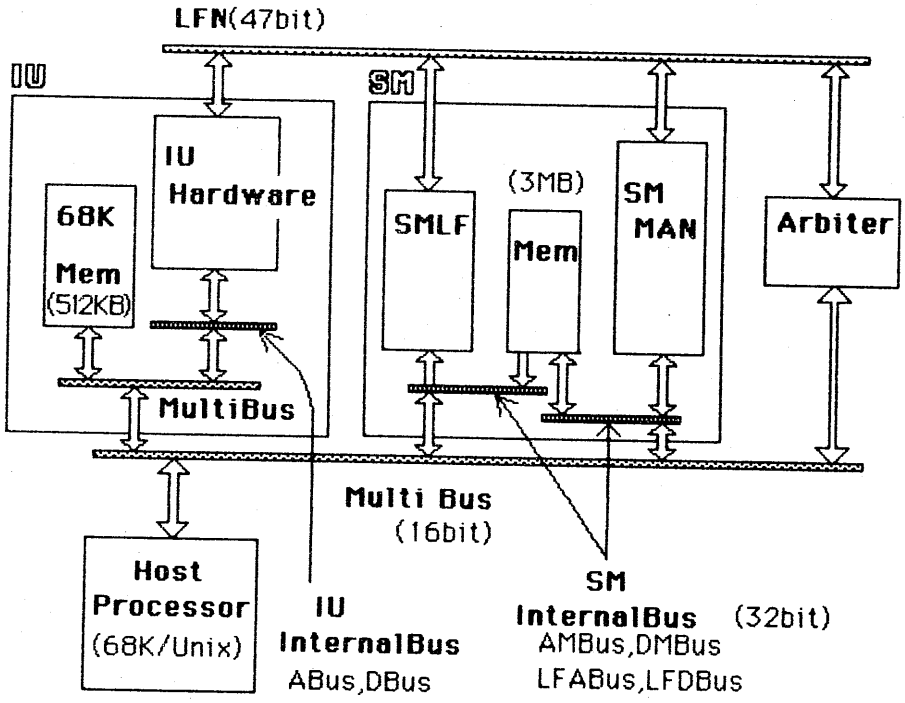


Fig.8 Simulator Hardware Configuration

した（ともにＩＣは殆どＡＬＳタイプを使用）。３．１節でも述べたように、ＳＭＬＦとＳＭＭＡＮは独立に動作し、２種類の内部バスから１つの共有メモリにアクセスを行う。ＳＭＬＦは読み出す一方であり、ＳＭＭＡＮは読み書き両方を行う。この共有メモリの容量は７６８Ｋ語（３Ｍバイト）であり、アクセス時間５５ｎＳのメモリＩＣを用い、４枚のボードで実装した。共有メモリに対するアクセス競合はＳＭＭＡＮ側がハードウェア的に検出し、１マイクロサイクル待つようにした。従ってＳＭＬＦ側には共有メモリのアクセスに関してオーバーヘッドは全く無い。またＳＭとＩＵは非同期に動作する。

３．２．３　ネットワーク

ネットワークアービタはリングカウンタを基本とした回路を用いた。優先度が２段階あるため、２つのリングカウンタを外付回路により結合し所定の機能を実現した。リングカウンタのクロック周波数だけ２０MHz としたので高速なアービトレーションが行える。ネットワークのバス幅は４７ビットであり、その内訳は次のようになっている。

ａ．データ線　３２ビット：
　ワードシリアルな伝送が可能である。
ｂ．送信相手ユニット番号　６ビット：
　ＳＭＬＦ、ＳＭＭＡＮ、ＩＵの識別番号が割り振られる。
ｃ．ネットワークステータス　４ビット：
　ネットワークの動作モードや、転送されているデータの種類等を示す。
ｄ．制御線　５ビット：
　ストローブ線、受信側ＦＩＦＯのfull／empty のフラグ等がある。

ＬＦＮの転送速度は５Ｍ語／秒（２０Ｍバイト／秒）である。

３．２．４　テスト及びデバッグ環境

本試作機では、プログラムの開発、変更から実行までが一貫して容易に行えるよう、デバッグやテストをハードウェア的に支援した部分がいくつかある。それは、

▷ＩＵ、ＳＭ、ＬＦＮアービタに含まれるハードウェア資源がすべてホスト計算機からアクセスできるようにした、
▷アービタ内に、ネットワーク上にテスト用のパケットを送出できるようなハードウェアを設けた、
▷ＡＬＵや連想メモリだけに命令を与え、任意に単独動作させることができる、

等である。従って高速なマイクロプログラムの転送及びロード、ハードウェア診断、ユニットの単体動作確認等が簡便に行える。

３．２．５　その他

本試作機にはシーケンサＩＣがＩＵ１台当たり１個、ＳＭ１台当たり２個の計３個使用されている。いずれもＡＭＤ社のＡｍ２９１０Ａであり、クロック５MHz で動作する。各ハードウェアのマイクロ命令長はＩＵが１６８ビット、ＳＭＬＦが９２ビット、ＳＭＭＡＮが１２８ビットである。シーケンサ、ＡＬＵ等には同じＬＳＩを用いている。どの場合についても、シーケンサの命令（条件分岐に関するフィールドも含む）に２０～４０ビット、ＡＬＵの命令に５０ビット、バスとのゲート開閉用信号に１０～３０ビット程度要している。マイクロ命令はほぼ水平型に近い構造をしている。これは本シミュレータが試作機であり、将来、マイクロ命令体系の変更等に柔軟に対処する必要があるためである。従って命令のコード化、重畳化はなるべく抑えた。

コマンドの種類による分岐、タグの種類による分岐等の多重分岐に対応して、マルチウェイジャンプの機能を設けた。４WayのマルチウェイジャンプがＩＵ側で４か所、ＳＭ側で６４か所使える。

試作機内では１語が３２ビットから成り、上８ビットがタグ部、下２４ビットがデータ部として使われる。特にＳＭの共有メモリにアクセスする時はデータ部のサブフィールドがボード番号等の意味を持つ（図９）。

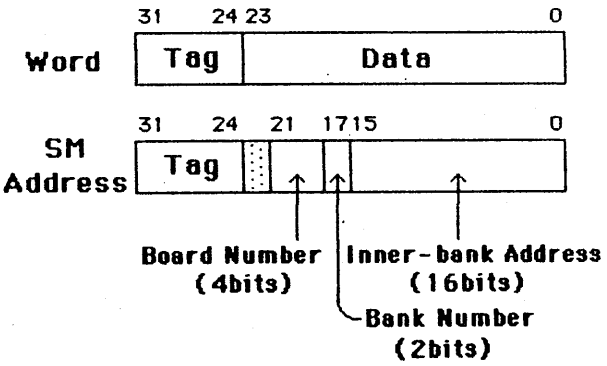


Fig.9 Word Configuration

４．　ＩＵハードウェアの内部構成

４．１　ＩＵハードウェアの概略

ＩＵハードウェアはＬＦ、ＧＩ格納、ＲＣ命令のバッファリング及び最適化、ＥＡのバッファリング、コマンドパケットの生成を行う。図１０にその内部構成を示す。ＡＢus、ＤＢusの幅、レジスタ長は３２ビットである。ＲＣreg 、ＧＩＢ、ＬＦＢ、ＥＡＢ、g-cellは６８Ｋマイクロプロセッサから読み書きされるレジスタであり、データの受け渡しに使われる。ＣＡＭtop 、ＧＩptr 、ＣＯＢtop は自動イ

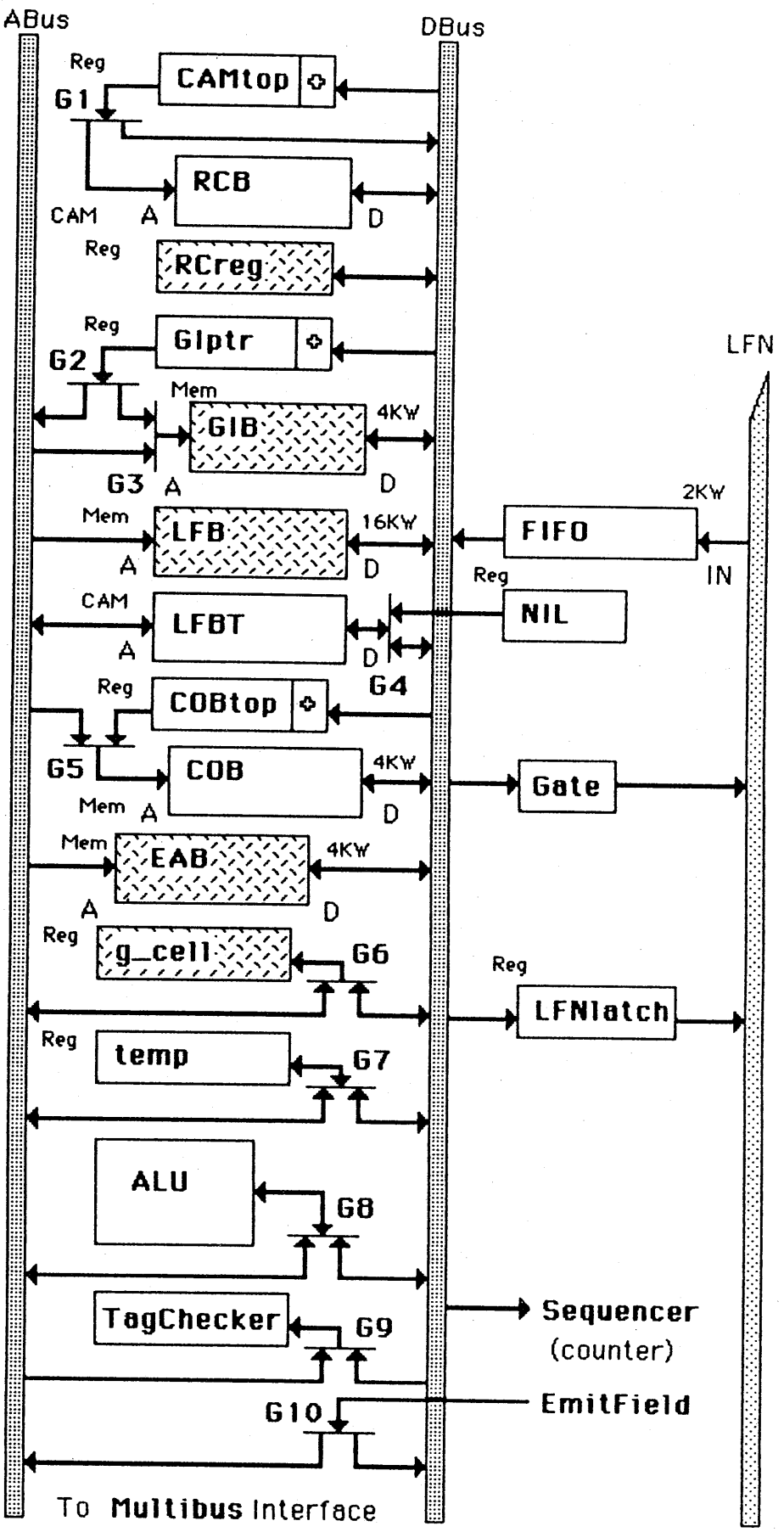


Fig.10 IU hardware Block Diagram

ンクリメントが可能である。ＡＬＵには多機能なＡｍ２９１１６（１６ビット幅）２個を縦続に接続し３２ビット幅として使い、上位と下位で全く独立に動作させることができる。またＡＬＵ中には３２語のレジスタファイルがあり、定数の保持や通常のレジスタとして用いるのには十分なので、他にレジスタは設けなかった。ＩＵハードウェアは通常アイドルの状態にあり、レジスタやバッファは６８Ｋマイクロプロセッサから自由にアクセスできる。ＬＦ等の処理を行う時には６８Ｋマイクロプロセッサが所定のレジスタに値を設定し、シーケンサに割り込みをかける。

４．２　ＬＦ

ＬＦに関連するレジスタ及びバッファはg-cell、ＬＦＢＴ、ＬＦＢ等である。

▷g-cell：６８Ｋマイクロプロセッサからの指示でＬＦしたいＳＭの番地が書き込まれる。そしてＬＦの処理が終了すると、答えとしてＬＦしたＧＩノードの格納されているＬＦＢの番地が入る。

▷ＬＦＢＴ（ＬＦＢテーブル）：ＬＦＢ中にあるＧＩノードのＳＭでの番地を持つ。ＬＦＢＴのエントリ位置とＬＦＢのエントリ位置は対応しており、ＬＦＢＴにおいてエントリされていないセルにはnilが書き込まれる（図１１）。連想メモリで実装されているので表検索は１μステップ（２００ｎＳ）で完了する。ＬＦＢＴの大きさは１２８語である。親ＧＦがすべての子ＧＦを作り終えた時点でクリアされる。

▷ＬＦＢ：ＬＦされたＧＩノードの実体が格納される。小さなＧＩは先頭から１２８セル以内の領域に格納されるのに対し、大きなＧＩについてはそこに間接ポインタが埋め込まれ、実体は１２８番地以降に格納される。

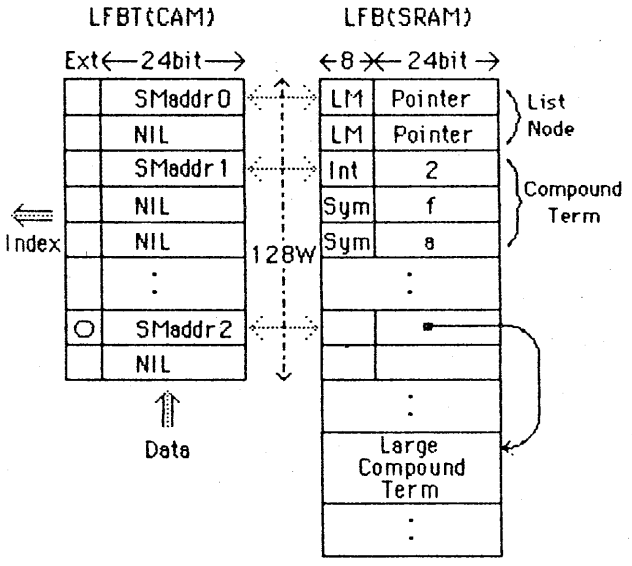


Fig.11 Operation on LFBT and LFB

ＬＦは以下の手順で実行される。

①g-cellに書き込まれたＧＩのＳＭ中の番地がＬＦＢＴ中にあるか否かを検索する。

②もし該当番地が存在したら、その番地を格納しているＬＦＢＴセルのインデックスを答えとしてg-cellに書き込む（終了）。

③もし該当番地が無い場合は、ＬＦＮlatch にＳＭの番地を書き込みＬＦＮの要求を行う。ＳＭからの返送を待つ間、ＬＦＢＴ、g-cellに値の書き込みを済ませておく。

④ＳＭから送られて来たＧＩはＦＩＦＯに入るので、それをＬＦＢの所定の位置に格納する（終了）。

４．３　ＧＩの格納

ＧＩの格納に関連するレジスタ及びバッファはＧＩＢ、ＧＩptr 、ＣＯＢ、ＣＯＢtop 、ＥＡＢ等である。

▷ＧＩＢ（ＧＩバッファ）、ＧＩptr ：ＧＩＢは６８Ｋマイクロプロセッサから格納したいＧＩを受け渡されるバッファである。ＧＩptr は読み書きの際のインデックスレジスタとして用いられる。

▷ＣＯＢ（Command　Output Buffer ）、ＣＯＢtop ：ＧＩＢ中のＧＩノードはＥＡＢ（ＥＡバッファ）から取り出されたＥＡと組にしてＣＯＢに転送される。

▷ＥＡＢ：ＳＭから渡されたＥＡを格納するバッファである。ＥＡはＧＩを格納する時に取り出され、ＧＦに埋め込まれるのと同時に、ＧＩと組にしてＣＯＢに転送される。

４．４　参照カウント命令の管理

参照（ＲＣ）カウント命令に関連するレジスタ及びバッファはＲＣreg 、ＲＣＢ、ＣＡＭtop 等である。

▷ＲＣreg ：６８Ｋマイクロプロセッサから縮退時にＳＭアドレスと±１が書き込まれる。

▷ＲＣＢ（ＲＣバッファ）：上８ビットが参照カウンタを更新する値、下２４ビットがＳＭの番地を表す。連想メモリで実装されているので、ＳＭ番地の検索は１μステップで完了する。

▷ＣＡＭtop ：ＲＣＢ内で次にどのセルに書き込むかを表わすレジスタである。

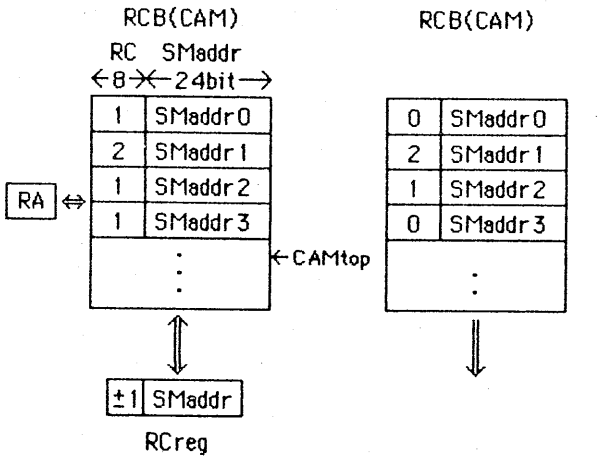
ＲＣＢの更新は以下の手順で実行される（図１２（ａ））。

①ＲＣreg に書き込まれたＳＭの番地がＲＣＢ中にあるか否かを検索する。この時あらかじめ上８ビットが＋１か－１かで分岐しておく（図中Ａlgor

ithm１、２行目）。
②もし該当番地があった場合は、そのインデックスをＲＡに読み出し、同時に該当セルをＲＣreg に読み出す（３行目）。
③①の分岐が＋１か－１かに従い、ＲＣreg の上８ビットを更新する（４行目）。
④ＲＡで示されるＲＣＢの中のセルにＲＣreg の内容を書き込む（５行目）（終了）。
⑤①でもし該当番地が無い場合は、ＣＡＭtop で指されるセルにＲＣreg を書き込みＣＡＭtop を＋１する（終了）。

ＲＣＢからＣＯＢにＲＣ命令を転送する場合、参照カウンタの更新値が零の命令はＳＭに送る必要が無い。連想メモリを使用したために、非常に少いオーバーヘッドでこのような命令を取り除くことができた。以下ＲＣ命令の転送手順を述べる（図１２（ｂ））。

①上８ビットが零であるようなセルを検索する（図中Ａlgorithm１～３行目）。
②該当セルを全てinactiveにする（４行目）。
③残りのactiveなセル全てを選択する（５、６行目）。
④選択されたセル全てを読み出す（７行目）（終了）。



Algorithm
1 WID RCreg
2 SRD
(Pexout == 1)
3 RDBN
4 increment RC
5 WAB RA

(a) Update

Algorithm
1 WMD 0x00ffffff
2 WID 0x0
3 SRD
4 ESST
5 WMD 0xffffffff
6 SRD
7 while(Pexout==1)
RDBY

(b) Read Out

Fig.12 Updating and Reading Out RCB

４．５　構造メモリの空き番地の管理

空き番地（ＥＡ）の管理に関連したバッファはＥＡＢである。

▷ＥＡＢ：リストのＧＩノード又は複合項のＧＩノードを格納するために用いられるＬＭのＥＡとＡＴＴのＥＡが格納されており、必要に応じて読み出される。ＥＡＢ中のＥＡがある一定個数以下になると、ＳＭに対してＥＡの要求を行う。

４．６　その他

ネットワークからのデータ受信はシーケンサと非同期に行われる。ＦＩＦＯにそのデータが格納されると条件フラグの１つがＯＮになる。ネットワークへデータを送信する時は、アービタに要求を出しＡckが返って来た後、ＤＢus上に送信したいデータを出力し、Gate を開ける。

ＴagCheckerは１ステップでタグ部（語の上８ビット）の判定ができるので、その出力は条件分岐する場合に用いられる。

シーケンサ内のカウンタを用いて条件分岐することも可能にするため、ＤＢusからシーケンサへの経路を設けた。

５．　ＳＭハードウェアの内部構成

５．１　ＳＭハードウェアの概略

ＳＭハードウェアの内部構成図を図１３に示す。３．１節で述べたように、ＳＭハードウェアには同一クロックで独立に動作する２つのシーケンサがある。共有メモリはバンク分けされており（１バンク＝６４Ｋセル）、ＭＡＮ側とＬＦ側が同一バンクにアクセスを行わない限りは全く競合は起こらない。もし同一バンクに対してアクセス競合が生じた場合は、ハードウェアでそれを検出し、ＭＡＮ側を１μステップ待たせる。

ＬＦＲは自動インクリメント機能を持ち、ＷＭＲとＭＡＲは自動インクリメント・デクリメント機能を持つ。ＬＭ、ＡＴＴ、ＶＣＭ、ＲＣＭはすべて共有メモリ上に割り付けられるが、そのメモリ管理法としては、最初から各々に決められた領域を割り振っておく方法と、ＯＳのように要求があった時点で逐一適当なブロックを割り付けて行く方法の２通りを考えており、どちらでも実現可能な構成になっている。ＳＭＬＦ、ＳＭＭＡＮ共に、シーケンサ内のカウンタを使用する場合を考慮し、ＩＵ同様バスからシーケンサへの経路を設けた。２つのＡＬＵともＩＵハードウェアと同じものを使用した。このＡＬＵは非常に多くの機能を持ち、レジスタファイルの容量も十分なため、外部には特にレジスタを設けなかった。

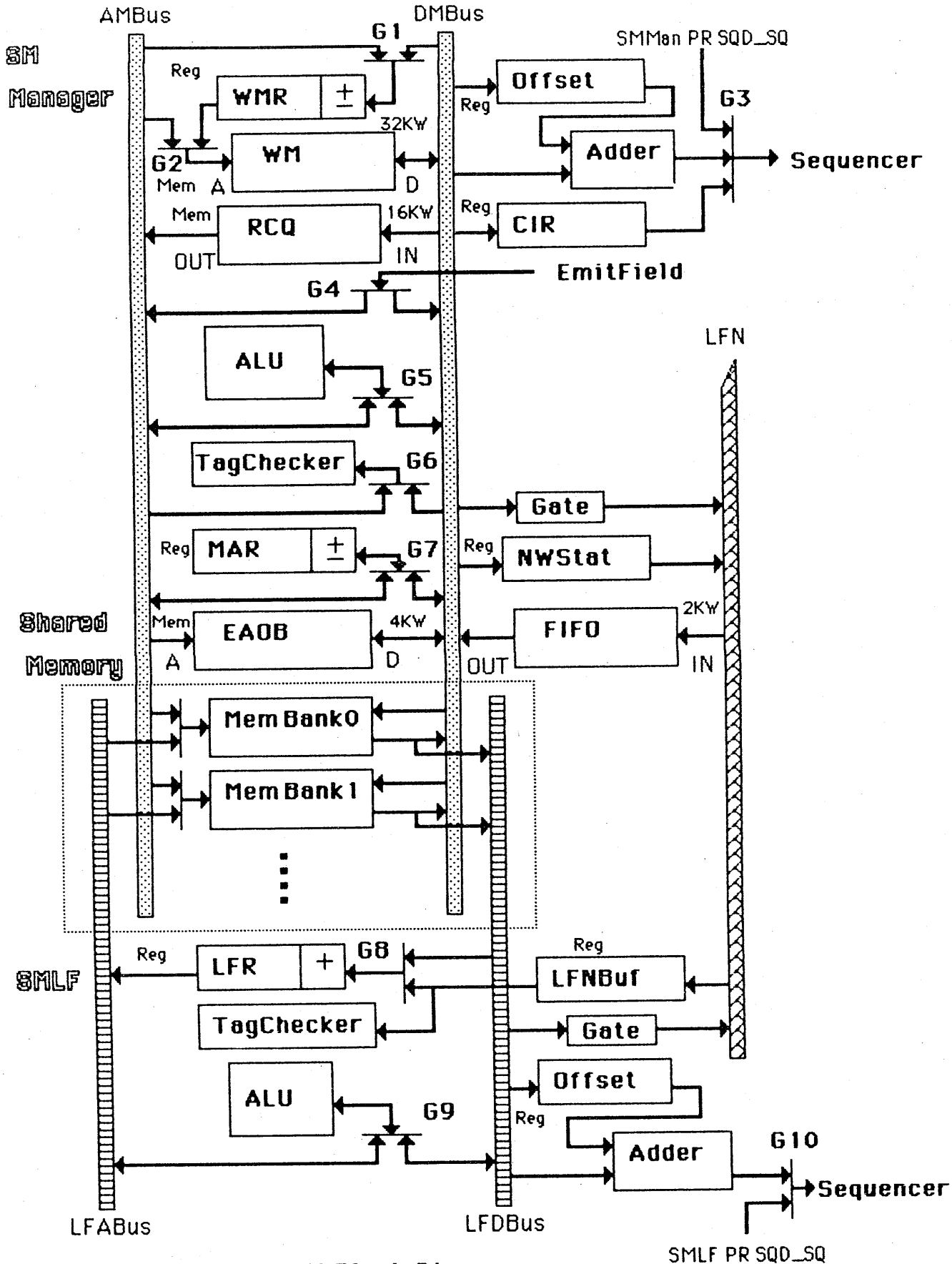


Fig.13 SM Block Diagram

5．2　SMLF側

SMLFは普段アイドルの状態であり、LFNBufからの割り込みを受け付けてLF処理を開始する。LFNBufには、IUの要求しているGIノードを指すポインタとそのタグが入る。タグ部はIU側と同じTagCheckerにより、リストか複合項かの判定に使われる。ポインタ（SMのアドレス）はLFR（Lazy Fetch Register）に入り、LFABusに送出される。リスト、複合項いずれの場合でも次に読み出すべき共有メモリのアドレスは常にLFRに格納されているので、共有メモリの読み出しは1セル／1μステップで行える。LFの最中はIU側がアービタに要求を出し続けているので、SMLFからGIノードを返送する際、SMLFがアービタに要求を出す必要は無い。

5．3　SMMAN側

SMMAN側では、GIの格納、RC命令処理、EAのバッファリング、ガーベジコレクション（GC）を行う。SMMANは通常RC命令の処理を行っており、1ノード処理する度にFIFOにコマンドパケットが到着しているか否かをチェックする。もし到着していればコマンドパケットの処理を行い、そうでない時はRC命令の処理を続行する。IUはGI格納、RC更新、EA要求のコマンドを1つのパケットにまとめて、LFNを通してSMMANのFIFOに転送する。SMMANはパケットの中から個々のコマンドを取り出し、コマンドの種類を判定し、CIR（Command Instruction Register）に各コマンドの処理を行うマイクロルーチンの先頭番地をセットしジャンプする。

▷GIの格納

リストノードを格納する場合は、ただ単にLM上の指定されたアドレスにGIノードを格納し対応するRCMを更新すれば良い。複合項を格納する場合は、ATTに関してはリストノードと同様であるが、VCMに関しては適合するサイズの空きノードを自由リストから取り出して来て、そこに書き込みを行う。

▷RC命令

実行しなければならないRC命令は全てRCQ（Reference Count Queue）に格納する。従ってコマンドパケット中のRC命令は即座に処理されるのではなく、全て一旦RCQに格納されてから処理される。RCQはFIFOで管理されているので、零の伝搬はBreadth Firstに行われる。尚、RCQに対して同時に入力と出力を行うことはできない。

▷EAの管理

LMのEAとATTのEAはEAOB（Empty Address Output Buffer）に蓄えられており、EAの要求があった時は即座に応答する。EAOBへのEAの補充は、GCの最中に適宜行われる。

▷GC

RCQからRC更新命令を1つ取り出し、対応するノードの参照カウンタを更新する。もし参照カウンタが零にならなければ何もせず、零になった場合は、そのノードに含まれるポインタを－1という更新値とともにRCQに挿入する。ゴミのノードは各々の自由リストに接続する。複合項をVCMに格納する際適当な大きさの領域が見つからない場合は、新しいVCMの領域を確保し、VCMのコピー・圧縮を開始する。以降は新VCMの方にGIを割り付ける。この時、旧VCMの内容全部を1度にコピー・圧縮するのではなく、RC更新命令の処理と共に、一定個数のノードを徐々にコピー・圧縮して行く。

5．4　その他

統計情報の記録等、様々な用途に使うためのメモリとしてWM（Working Memory）と、WM専用のインデックスレジスタとしてWMR（Working Memory Register）を設けた。また共有メモリのアクセスは連続した番地に対して行うことが多いので、自動インクリメント・デクリメント機能付きのレジスタMAR（Memory Address Register）を設けた。

6．　構造メモリ試作ハードウェアの特徴

以下に本シミュレータの特徴を列挙する。

▷LFを高速に行うハードウェアを持つ

UPで行われる単一化操作の速度が直接マシン性能に反映されるので、LFに関連した部分は全てハードウェア化し、できるだけ高速に実行するようにした。IU側においてはLFBTに連想メモリを用い、SM側ではLF専用のシーケンサを設けた。UPはLFNを通してDMAのようにSMをアクセスする方式をとり、アービタを20MHz のクロックで動作させ、高速なアービトレーションを実現した。

▷様々なアルゴリズムがテストできる

LF、GI格納、RC方式によるGC、SMのメモリ管理等について、できるだけ多くのハードウェアアルゴリズムをテストするには、ハードウェア自体が柔軟かつ多機能でなくてはならない。そのために、十分なレジスタファイルを含む多機能なALUやワーキングメモリを装備し、RCMを共有メモリ上に置き24ビットの幅を持たせ、マルチウェイジャンプ機能を取り入れた。

▷デバッグやデータ収集が容易である

作業時間を有効に使うためには、ハードウェアやソフトウェアのデバッグ機能が充実しており、データ収集の手間ができるだけ少い方が望ましい。従ってホスト計算機からはWCS等も含む全てのハードウェア資源にアクセスできるようにし、ブレークポイントを設定するため、WCSのアドレスを監視できるようにした。これによりマイクロプログラムのロードやデバッグ、統計情報の収集が高速に行える。また連想メモリやALUの周辺にデバッグ用のレジスタをいくつか設け、LFNにテスト用パケットを送出できるようにした。

▷拡張可能である

これは様々なアルゴリズムが実現可能であることと多少関連するが、将来さらにPIEの試作を進めて行く上で、本ハードウェアが全体としても部分としても活用されることが望ましい。例えば、どんな大きなGIでも送受信・格納できるように、LFBを十分大きくとり、LFNにFIFOのfull／empty 線を設けたので、現在とは異なるSMの使用法も考えられる。ネットワークステータスや送信相手ユニット番号に余裕を持たせ、SMの共有メモリボードが通常のメモリボードとしても使用できる。

7．おわりに

現在のSMが格納するのはGIであるが、将来は若干の機能追加でGI以外の構造データも格納し、集合演算、高階機能、ストリームによるプロセス間通信等の言語機能を支援することを考えている。また本試作機の高速性は連想メモリに負う所も大きい。

本試作機は現在製作中であり、並行してマイクロプログラム等のソフトウェアも作成している。デバッグ終了後は種々のデータ収集を行い、構造データ共有方式の有効性を実証する。また構造データ共有方式が定義節に含まれるGIにも有効か否かの検討や、LFNのトラヒックに関するデータに基づくレベル1システムの最適IU台数の決定等も行う予定である。

謝辞

連想メモリチップを使用するに当たり、NTT厚木通信研究所集積回路応用研究室の小倉武氏他の皆様に御世話になり感謝致します。常日頃から熱心に討論し、貴重なコメントをして頂くPIE研究プロジェクト“SIGIE”のメンバー諸氏に感謝いたします。

＜ 文献 ＞

[1] T. Moto-oka et al, “The Architecture of A Parallel Inference Engine - PIE -” FGCS´84, ICOT.

[2] 平田他, “高並列推論エンジンPIEにおける構造データの効率的な処理方式について”, 信学技報, EC83－38.

[3] 平田他, “PIEにおける構造メモリの構成について”, アーキテクチャワークショップインジャパン´84, 情報処理学会.

[4] 平田他, “PIEの構造メモリ試作ハードウェア～全体構成～”, 第32回情処全大, 1R－1.

[5] 猪股他, “PIEの構造メモリ試作ハードウェア～推論ユニット～”, 第32回情処全大, 1R－2.

[6] A. Goldberg and D. Robson, “Small talk－80：The Language and Its Implementation”, Addison－Wesley, 1983.

[7] 小倉他, “大容量連想メモリLSIの構成とその応用手法”, 信学技報, CAS84－192.

[8] L. P. Deutsch and D. G. Bobrow, “An Efficient, Incremental, Automatic Garbage Collector”, CACM, 19, 9, (Sept. 1976).