計算機アーキテクチャ 64－3
（1987. 1. 27）

ＰＩＥの構造メモリ試作ハードウェアのソフトウェア全体について

猪股 宏文， 平田 圭二， 田中 英彦

（ 東京大学 工学部 ）

我々は高並列エンジンＰＩＥの構造メモリ試作ハードウェアの開発を行っている。構造メモリ試作ハードウェアとは、ＰＩＥで構造データ共有方式を支援するための構造メモリと、その性能の評価に必要な推論ユニットから構成されている実験システムである。ハードウェアの開発及び評価に伴い、色々なソフトウェアを開発した。それらのソフトウェアには、ハードウェアモニタ、マイクロアセンブラ、専用ハードウェアの制御のためのマイクロプログラムや推論ユニットシミュレータ等がある。以下では、それらのソフトウェアについて報告する。

THE WHOLE SOFTWARE FOR A STRUCTURE MEMORY PILOT MACHINE

Hirofumi INOMATA, Keiji HIRATA and Hidehiko TANAKA

Faculty of Engineering, University of Tokyo, 3-1 Hongo 7-chyoume, Bunkyo-ku, Tokyo 113, Japan

We have developed a Structure Memory Pilot Machine for a Parallel Inference Engine -PIE-. This pilot machine is an experimental system consisting of a Strucure Memory and an Inference Unit for evaluating the Structure Memory. This paper presents the whole software developed for this system. These software include a monitor for debugging the hardware, a translator for microprogramming language, microprograms for the Structure Memory and part of the Inference Unit, a simulator for the Inference Unit, etc.

１．はじめに

現在、我々は高並列エンジン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ム言語を高速かつＯＲ並列に実行するマシンである。

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することがソフトウェアシミュレーションによって確認されている[2]。ＰＩＥにおいて構造データ共有方式を支援するには、Ground Instance 格納用の共有メモリとして構造メモリ(Structure Memory)の導入が不可欠であり[3]、現在試作が進められている[4,5,6,7]。本稿では、構造メモリ試作ハードウェアにおけるソフトウェアについて報告する。

２．構造メモリ試作ハードウェアの概要

構造メモリ試作ハードウェアは、ＰＩＥに構造データ共有方式を導入したときの現実的なデータを測定するための実験システムである。各部の構成を図１で示す。

構造メモリ試作ハードウェアは、大きく分けて構造メモリ(SM)、推論ユニット(IU)とホストコンピュータ(HOST)の３ブロックから構成されている。ＳＭとＩＵは、Lazy Fetch Network(LFN) でむすばれている。また、ＳＭとＩＵはホストコンピュータから直接制御できるようにそれぞれがホストコンピュータにつながれている。

ＳＭは、全てが専用ハードウェアで組まれ、SMLFとSMMan とMemoryの３つのブロックから構成されている。SMLFはＩＵからのLazy Fetch(LF)の処理を専用に行う。SMMan は、RCの更新、GIの格納と空アドレス(EA)の管理を行う。Mem はＳＭのメインメモリを示していて、ここにGIが格納される。

ＩＵは、専用ハードウェアと市販されている６８Ｋボードコンピュータから構成されている。実際のＰＩＥのＩＵは、単一化プロセッサ(UP)とメモリモジュール(MM)と定義節メモリ(DM)とアクティビティコントローラ(AC)から構成されている。本システムでは、構造メモリを支援するのに必要な部分（ＩＵの一部）を専用ハードウェアで構成し、その他の部

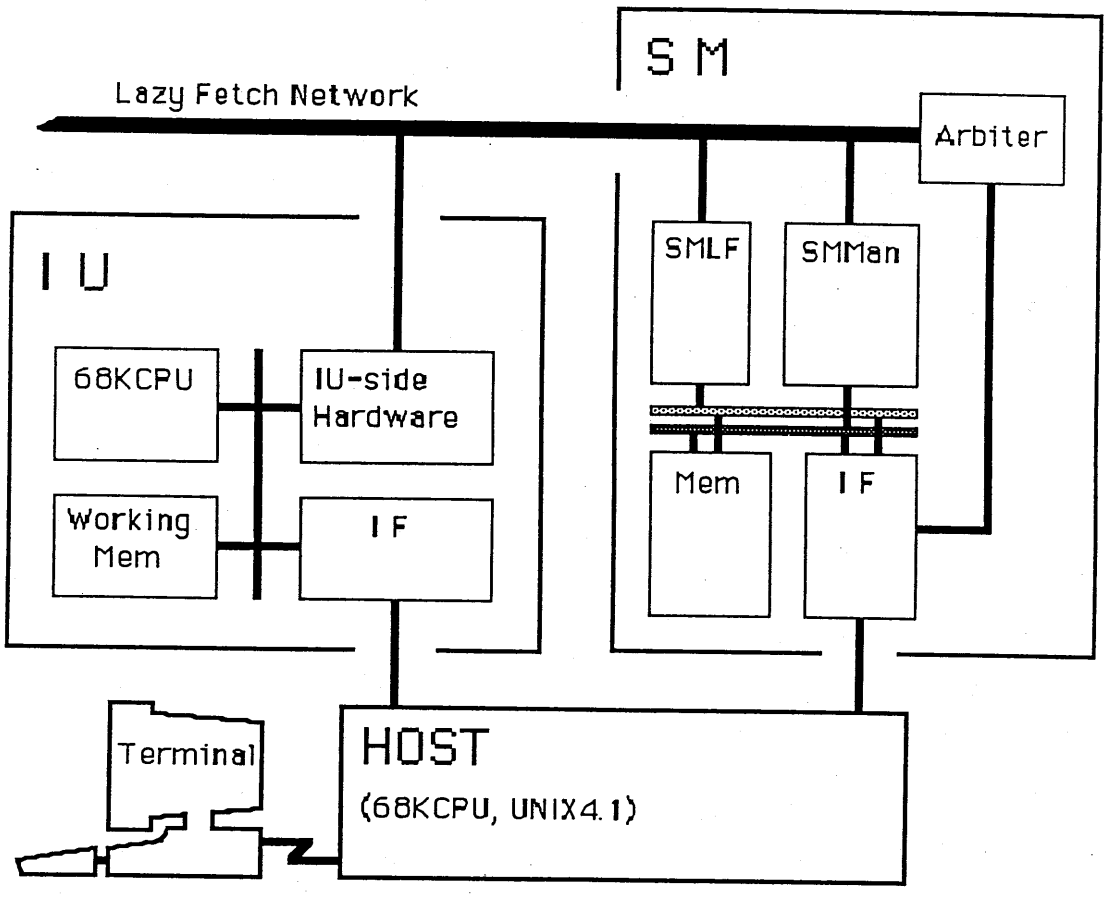

図１．構造メモリ試作ハードウェアの全体構成

分を６８Ｋボードコンピュータでシミュレーションする。また、ＳＭとＩＵにあるIFはＨＯＳＴとインタフェスするためのハードウェアである。ＳＭとＩＵの専用ハードウェアはそれぞれマイクロプログラム制御方式を採用し、シーケンサとしてAm2910を用いている。マイクロ命令コードフォーマットは水平型に近いものである。また、それぞれの専用ハードウェアには演算ユニットとしてAm29116 を用いている。 本試作ハードウェアに必要とされるソフトウェアは、ＳＭとＩＵの専用ハードウェアの制御のためのマイクロプログラムとＩＵの６８Ｋボードコンピュータで走らせるための推論ユニットシミュレータ[7] と更にこれらを開発するためのツールとしてのソフトウェアがである。

3．ソフトウェア開発環境

我々は、ソフトウェアの開発が円滑に行えるように環境を整えている。ホストコンピュータのＣＰＵは 68010であり、OSはUNIX4.1BSDである。全てのソフトウェアはUNIXの上で操作管理されていて、マイクロプログラム以外のソフトウェアは全てＣ言語とUNIXユーティリティによって開発されている。特に、UNIX4.2BSD以降支援されているMMAPシステムコール[8] を本ハードウェアを制御するためにUNIX4.1BSDに移植している。以上のようなホストコンピュータの上で (1)ハードウェアモニタ、 (2)マイクロアセンブラ[6] 、 (3)ローダ、 (4)マイクロシミュレータの４つのソフトウェアをツールとして開発した。以下で、それぞれのソフトウェアの簡単な紹介をする。

3．1．ハードウェアモニタ

このソフトウェアは、構造メモリ試作ハードウェア全体の資源つまりメモリ、データレジスタ、制御レジスタ、状態レジスタ等に端末からデータが読み書きできるようにするためのものである。アクセス可能な資源はホストコンピュータからみると１つのアドレス空間にマッピングされていて、１つのアドレスを与えると目的の資源をアクセスすることができる。このソフトウェアを開発するに当って、MMAPシステムコールを用いている。このソフトウェアは、特にハードウェアのデバッグに用いられている。

3．2．マイクロアセンブラ

このソフトウェアは、UNIXのLEX/YACCユーティリティ等を用いて開発した。マイクロプログラムの必要な部分はＳＭに２種類とＩＵに１種類ある。それらの制御するハードウェアはそれぞれ異なっているので、マイクロアセンブラも３種必要である。我々は、マイクロプログラムの開発が円滑に行われるように、マイクロアセンブラのニーモニックは共通した形式に統一している。マイクロアセンブラのニーモニックとしては、マイクロアセンブラ自身の開発の容易性と見安さの点からProlog-like な形式を採用している。

１ステップのマイクロ命令コードに相当するマイクロアセンブラのステートメントは以下のようである。

$OP_1(ARG_1), OP_2(ARG_2), \cdots, OP_N(ARG_N);$
OP_I: 操作名
ARG_I: アーギュメント (I, N=1, 2, 3, ・・・)

操作名に対しては純粋な操作の名称のみではなく、その操作が対象となる資源の名称も用いられる。以下では、幾つかのニーモニックについて説明する。

(1) データ転送の制御

movea(SRC, DST_1, DST_2, ・・・, DST_N),
moved(SRC, DST_1, DST_2, ・・・, DST_N),
SRC: Source Storage
DST_I: Destination Storage(I, N=1, 2, ・・・)

ＩＵとＳＭのどれも２つのバスを柱として構成されているので、それらにそれぞれＡとＢという名称を付け、それらを仲介してデータを転送するときにそれぞれmovea とmoved を用いる。各データレジスタは SRCと DSTに同一のレジスタを指定することによってリードモディファイライトの指定が可能である。

(2) シーケンサの制御

seq(Ins, ・・・),
Ins: Instruction

シーケンサの命令の名称をAm2910a のマニアル[9] によっている。これの役割は、主に条件分岐命令、無条件分岐命令、サブルーチン分岐命令等の指定を行う。

(3) 連想メモリの制御

cam(Sel, Ins, ・・・),
Sel = LFBT or RCB
Ins: Instruction

連想メモリはＮＴＴ厚木通研で開発されたAPL1を

用いている[10]。

(4) その他

以上説明したニーモニックの他にそれぞれのハードウェアに特有のものもある。ＩＵの専用ハードウェアでは、６８Ｋボードコンピュータとインタフェースするために、６８Ｋからのリクエストフラグ等をリセットするためのニーモニックがある。

本マイクロアセンブラでは分岐先アドレス等にラベルを用いることができる。また、ラベルを数値のマクロとして用いることもできる。

また、シーケンサには多分岐命令があり、それを支援するために多分岐アドレステーブル（メモリ）にあらかじめ分岐先アドレスを設定する必要がある。この設定をするための疑似命令がある。

実際にアセンブラから出力されるオブジェクトはダンプリストのようなＡＳＣＩＩコードの数字列の形にした。これは、マイクロアセンブラ自身のバグの発見を迅速に行うためである。以上説明したマイクロアセンブラは、制御用マイクロプログラムをアセンブルする以外に、ハードウェアのデバック用のマイクロプログラム等にも用いられる。

３．３．その他

我々は、上の２つのソフトウェア以外にローダとマイクロシミュレータを製作した。

ローダは、マイクロアセンブラの出力したオブジェクトを実際の専用ハードウェアのＷＣＳにロードするためのソフトウェアである。また、ローダに入力するファイルにデータを加えることによって、マイクロプログラムのオブジェクトのロードと同時に各レジスタやメモリ等にデータを書き込ませることもできる。

マイクロシミュレータは、マイクロプログラムのオブジェクトを入力すると、各専用ハードウェアの動作をシミュレーションし、マイクロ命令コードの簡単な逆アセンブラをする機能も有しているソフトウェアである。これは、ハードウェアがある程度動作するようになるまで、マイクロプログラム等をデバックするために製作された。

４．制御用マイクロプログラム

ＳＭはSMLFと SMManの２つのブロックから構成されていて２種の制御用マイクロプログラムが必要であるが、それぞれのブロックは機能別に分割されているので、機能（サブコマンド）別に制御用マイクロプログラムのアルゴリズムを簡単に説明する。ＳＭの２つのブロックのうちSMLFは Lazy Fetch を専門に処理するブロックである。また、ＩＵの制御用マイクロプログラムについても以下で機能別にアルゴリズムを説明する。機能としては、ＳＭとＩＵのそれぞれに (1)LazyFetch(LF) 、(2) 参照カウント(RC)の管理、(3) GIの格納と(4) 空アドレス(EA)の管理に相当する４つのものがそれぞれある。RCはガーベジコレクションのために用いられる。

４．１．構造メモリのマイクロプログラム

ＳＭは、ＩＵからのコマンドパケットとＬＦＮアービタからの指示によってどのサブコマンドを実行するかを決定する。サブコマンドは、(1) ＬＦ、(2) ＲＣの更新、(3) ＧＩの格納、(4) ＥＡの管理の４つである。

(1) Lazy Fetch(LF)

SMLFは、busy wait でＩＵからLFNBufにデータが到着しているのを知らせる条件分岐信号がactiveになるのを待つ。LFNBuf内のデータのタグ部を見てそれがリストであるかベクタであるかの検出をして、各々の処理ルーチンへ実行を移す。ＩＵに返すデータはメモリに格納されていてそれを指すポインタはインクリメンタブルな専用レジスタ(LFR) に格納されているので、連続したデータの読み出しは１語１ステップで行える。リストデータの読み出しには、９ステップ、サイズｎのベクトデータでは（ｎ＋８）ステップかかる。

(2) 参照カウンタ(RC)の更新

SMMan は、Reference Count Queue(RCQ)にバッファリングされているRC命令を１つ処理する毎にＩＵからコマンドパケットが到着しているかを検査する。コマンドパケットが到着している場合はコマンドパケットの処理を行い、到着してない場合はRC命令の処理を続行する。ここでコマンドパケットの処理とは、それを各サブコマンドの処理できるような形に展開することであり、RC更新のサブコマンドに対してはRC命令のコマンドパケットをRCQ にバッファリングする操作に相当する。

また、RCの処理に対してその処理結果をＩＵに折り返して確認することを行わないので、ＩＵはRC命令の送出のタイミングに注意する必要がある。任意のタイミングで行うと、ＳＭとＩＵの間のRCの一貫性が崩れる場合がある。これについては、5.で説明する。

(3) Ground Instance(GI) の格納

ＳＭに格納されるGIは空アドレスを伴ったコマンドパケットとして送られてくる。空アドレスは、Address Translation Table(ATT)のアドレスである。

実際には、GIは ATTに登録されているアドレスに書き込まれる。そして、そのGIに対するRC命令を RCQにpushして処理を終える。従って、ベクタデータに対しては、それが送られてきた時点でVari-size Cell Memory(VCM)領域に動的に割付られる。VCM の空領域はサイズ毎の自由リストで管理されている。また、リストデータはList Memory(LM) 領域に格納され、その空領域は自由リストで管理されている。リストデータは6ステップ、サイズnのベクトデータは（n＋14）ステップで処理が終了する。ATTとVCM とLMはメインメモリの上に割付られている。

(4) 空アドレス(EA)の管理

ＩＵから空アドレスの要求があった場合、リスト型かベクト型かに従って空アドレスバッファ(EAOB)から一定個数のEAを取り出しパケットの形でそのＩＵに返送する。EAOBには、ガーベジコレクションによって回収されたEAの一部を常にある個数以上バッファリングする。

以上の４つのサブコマンドをＳＭは管理している。また、アルゴリズムからの処理速度の概算によるとＩＵが１６台ののときでも性能の低下が起きない[6]。

4．2．推論ユニット(IU)のマイクロプログラム

ＩＵの専用ハードウェアは、６８Ｋボードコンピュータからの指示によってその動作を決定する。専用ハードウェアの機能は、(1) ＬＦ、(2) ＲＣの管理、(3) ＧＩの格納、(4) ＥＡの管理の４つである。

(1) Lazy Fetch(LF)

ＬＦの要求を受けると、要求されているGIが Lazy Fetch Buffer (LFB)にバッファリングされているかをLazy Fetch Buffer Table(LFBT) で検索する。LFBTには連想メモリが用いられ、最大１２８個までのGIノードを登録できる。バッファリングされていれば、それが格納されている LFBのアドレスを６８ＫＣＰＵに返して処理を終了する。バッファリングされていなければ、ＬＦの要求をＳＭに送出し、要求したGIの到着を待つ。ＳＭからGIが到着すると、それをLFB にバッファリングし、LFBTに登録し、それを格納した LFBのアドレスを６８ＫＣＰＵに返して処理を終了する。

(2) 参照カウンタの(RC)管理

ＩＵからRC命令の指示を受けると、ＳＭのアドレス毎にReference Count Buffer(RCB) にそのRC命令をバッファリングする。RCB には連想メモリが用いられ、１２８個までのGIノードに対するRC命令をバッファリングできる。もし、RCB に自由領域がなくなると、あるサイズ以上の自由領域をつくるために、バッファリングされているRC命令をコマンドパケットの形にして、Command Output Buffer(COB)にＳＭへの送出するためにバッファリングする。実際のコマンドパケットのＳＭへの送出は６８ＫＣＰＵからの指示によって起動する。

(3) Ground Instance(GI) の格納

GIの格納の要求を受けると、Empty Address Buffer (EAB) からGIの空アドレス(EA)を取り６８ＫＣＰＵに返す。そして、Ground Instance Buffer(GIB) に格納されているGIとそのEAを組にしたコマンドパケットの形で COBにバッファリングする。GIB には切り分けられたGIをあらかじめ６８ＫＣＰＵがバッファリングしておく。このコマンドパケットはRC命令のコマンドパケットと一緒に６８ＫＣＰＵの指示で COB からＳＭに送出される。

(4) 空アドレス(EA)の管理

空アドレスは常に一定以上の個数 EABにバッファリングされている。EAB にバッファリングされているEAがある個数を下回るとEAの要求命令のコマンドパケットを COBにバッファリングする。ＳＭからEAがコマンドパケットの形で返送されてくると、それを展開して EABにEAをバッファリングする。EAの量の検査は、EAB からEAを取り出すときに行う。

また、EAには格納するGIがリスト型かベクト型かで２種類ある。

4．3．その他

以上の４つの機能をマイクロプログラムで実現している。実際の６８ＫＣＰＵとのインタフェスでは(1) ＬＦ、(2) RC命令、(3) GIの格納、(4) コマンドパケットの送出というの４つのサブコマンドを用ている。また、RC命令に関しては１つのGIノードに対するRCのインクリメントとデクリメントの２種類のみを用いている。これはUPの処理によっていて、その処理は5.で説明される。

以上の制御用マイクロプログラム以外にハードウェアをデバッグするために幾つかの簡単なマイクロプログラムを製作している。これには、シーケンサや演算ユニットや連想メモリのようにインストラクションコードによって動作するものに対して個別に色々な命令をを送って動作を確認するためのものなどがある。実際の動作確認は、マイクロプログラムをステップ動作させハードウェアモニタでそれぞれのレジスタやメモリの内容をモニタすることや、ロジックアナライザやシンクロスコープによってそれぞれの信号を直に検査することによっている。

５．推論ユニット(IU)シミュレータ

以下では、ＩＵの６８Ｋボードコンピュータのためのプログラムについて説明する。このシミュレータを開発するに当って１つのハードウェアイメージを仮定している。このプログラムはホストコンピュータの上で開発し、そのオブジェクトを６８Ｋボードコンピュータにロードして実行する。現在はデバッグを兼ねてホストコンピュータ上で実行している。

５．１．推論ユニットのハードウェアイメージ

図２にシミュレーションするハードウェアイメージを示す。ＰＩＥのＩＵは、単一化プロセッサ(UP)とメモリモジュール(MM)と定義節メモリ(DM)とアクティビティコントローラ(AC)から構成されている。１点鎖線で囲まれた部分がUPであり、特に点線で囲まれた部分は専用ハードウェアをであり、残りの部分がソフトウェアでシミュレートされる部分[11]である。６８ＫＣＰＵと専用ハードウェアの間のデータの受け渡しには、レジスタではg_cellとRC_reg、メモリでは LFBと GIBが用いられ、制御レジスタで専用ハードウェアに指示を与える。MMは２つの動作モードを持っていて、(1) FIFOの動作、(2) 最後に入力したゴール節を最初の出力するゴール節単位のLIFOの動作のどちらかを選択することができる。これは、広さ優先の実行と深さ優先の実行との切り換えができるようにするためである。DMについては、定義節単位に読み込めるようなメモリを仮定している。ACについては特にハードウェアの仮定はしてない。

５．２．シミュレータのアルゴリズム

以下では全体的な処理の流れとして広さ優先実行のアルゴリズムについて簡単に説明し、ユニフィケイション（単一化）とリダクション（縮退）のそれぞれのアルゴリズムについて説明する。

5.2.1.全体的な処理の流れ

リーダがPrologのプログラムをDMに書き込み、初期

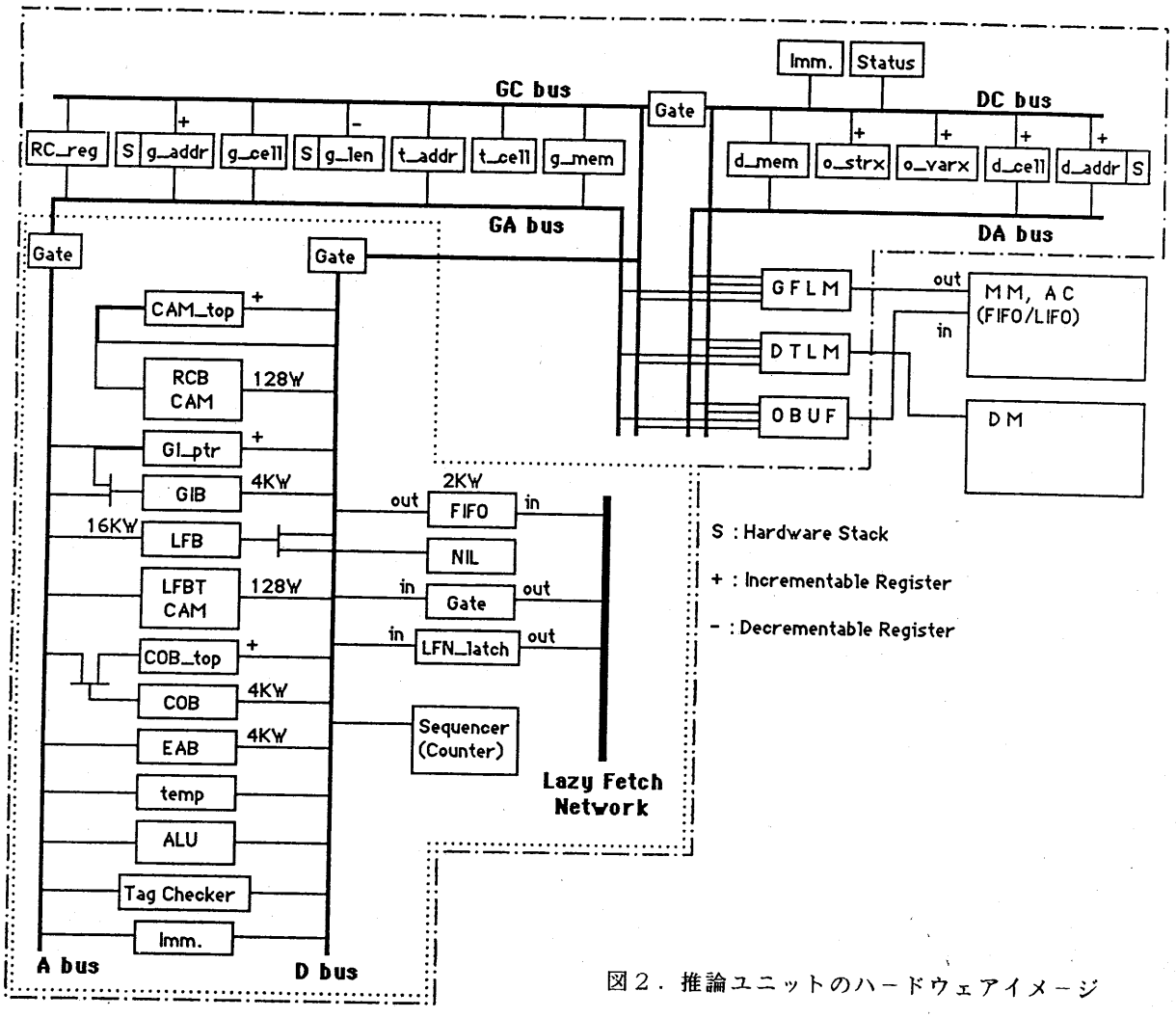

図２．推論ユニットのハードウェアイメージ

ゴールをMMに書き込むと実行が起動される。

まず、MMからGFLMにゴール節をコピーし、DMからDTLMに定義節をコピーする。

次に、UPが単一化を行う。単一化が失敗した場合は、次の定義節をDMからコピーしてきて再度単一化を行う。全ての定義節について単一化を試みてしまった場合は、新たなゴール節をMMからコピーしてきて再度初めの定義節から単一化を行う。単一化が成功した場合は、UPは縮退を行う。新たに生成されたゴール節があればMMに出力する。

そして、次の定義節をDMからコピーしてくるか、全ての定義節について単一化を試みてしまった場合は新たなゴール節をMMからコピーしてきて再度初めの定義節をコピーしてきて単一化の操作からMMにゴール節が無くなるまで繰り返す。

ＳＭへのコマンドパケットの送出はそのタイミングに注意が必要である。もし任意にコマンドパケットをＳＭへ送出すると、ＳＭ上のまだ必要とされるGIに対するRCが途中で零になるようなことが起きる可能性がある。このシミュレータでは、単一化の失敗の後と新ゴール節のMMへの出力の後という推論サイクルの終了時毎にコマンドパケットがＳＭへ送出される。こうすると、ＳＭ上RCが途中で零になるようなことは起きない。

5.2.2.ユニフィケイション（単一化）

従来のアルゴリズム[12]からＳＭを支援するために付け加えた部分について説明する。

ほとんど変更は無く、ゴール節のセルとそれに対応する定義節のセルを比較（ユニファイ）するときにそれらのセルがＳＭのアドレスであれば、Lazy Fetch(LF) の要求を専用ハードウェアに対して行う。その比較には専用のタグ検出回路を用いて、その結果で多条件分岐命令を行い各処理を行うと仮定している。また、もし最後の定義節との単一化であればそれらのＳＭのアドレスに対する参照カウンタ(RC)のディクリメント命令を専用ハードウェアに送る。これは、最後の定義節との単一化が終了すると元のゴール節は消滅するので、その中に含まれているＳＭのアドレスに対してその参照の数が減ったことをＳＭへ知らせることである。

処理速度は、ＬＦのために専用ハードウェアが必要とする処理ステップ数を従来の単一化の処理ステップ数に加算したものである。従って、ＬＦが発生しない場合従来の処理速度と全く等しくなる。これは、ゴール節と定義節おそれぞれのセルを比較して行う多条件分岐命令の飛び先を従来のものより増やすことによって実現される。

5.2.3.リダクション（縮退）

これについても、単一化の説明と同様に追加点を重点に説明する。この操作には(1) RCのインクリメント命令発行、(2) GIの切り分けの操作の２つが追加される。

RCのインクリメント命令の発行は元のゴール節から新しく生成されるゴール節にＳＭのアドレスがコピーされたときに起きる。

GIの切り分けの操作は、新ゴール節の生成と同時に行われる。元のゴール節と定義節からセルデータが新ゴール節にコピーされて操作が進むが、構造データに関しては、コピー先を新ゴールの形成場所であるOBUFと GIBの２箇所にマルチキャストしておき、それがGIであるか否かが判断できたところで、２箇所にマルチキャストした内のどちらかを元に戻すという操作を行う。もしGIであった場合、OBUFに既に書き込まれてしまったGIを消去して専用ハードウェアにGIの格納の要求を送り、その結果としてEAを受け取る。受け取ったEAはOBUFの消去されたGIの代わりにOBUFに出力する。GIでなかった場合は、GIB に既に書き込まれてしまった構造データを消去する。構造データの消去にはOBUFと GIBのどちら場合も、消去すべきデータの先頭アドレスを前以てレジスタに格納させておいて、それを自由領域先頭アドレスを格納するレジスタに転送するだけで良い。縮退の実行は構造データの頭から深さ優先に新ゴール節にコピーしていくので、GIとしては構造データのボトムの部分が深さ１段のGIとして検出され切り分けられる。以上の実行を現実するためにも、単一化のとき同様各多条件分岐命令の飛び先を増やしている。

以上のような操作を従来に縮退のアルゴリズムに加えた。処理速度に関しては、UPの本体と専用ハードウェアの部分はゲートによって接続されているという仮定をしているので、もし専用ハードウェアで前に要求した命令の処理が終了する前に何か要求を発生させると、それがUPの処理をウエイトさせる原因となる。

５．３．シミュレータの実行

このシミュレータの実行速度はUPのクロック周波数が５MHz としたときに、ホストコンピュータ上の実行で約1000倍遅くなる。これに、各種専用ハードウェアへの要求の発生間隔等のデータをファイルに出力させると、4000倍から 10000倍ぐらいの実行時間を必要とする。現在、ホストコンピュータ上で解以外のデータを出力させない状態で８ＱＵＥＥＮＳの例題の規模まで実行できることは確かめている。

また、このシミュレータはＳＭのあるときの実行と無いときの実行が両方できるようになっているの

で、ＳＭによる性能の向上の測定はこのシミュレータ１つで行うことができる。

６．おわりに

現在、ＳＭとＩＵの専用ハードウェアの一部とＩＵの制御用マイクロプログラムがデバッグ中である。それらのデバッグが終了しだい、ＳＭとＩＵの通信のトラフィック量等のデータや全体的な性能の測定を行う予定である。データを取る場合、走らせるPrologのプログラムに結果が相当依存すると思われるので、それについて現在考慮中である。

謝辞

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さまの御厚意で、連想メモリの使用ができ、その使用に際してはいろいろとアドバイスを頂き、深く感謝致します。また、日頃から貴重なアドバイスを頂くＰＩＥの研究プロジェクト“ＳＩＧＩＥ”のメンバー諸氏に感謝致します。

＜参考文献＞

[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ﾌﾟｲﾝｼﾞｬﾊﾟﾝ'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回情報全大，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回情報全大，1R 2.

[6] 平田他，“ＰＩＥの構造メモリ試作ハードウェアのマイクロプログラム”，第33回情報全大，5B-2.

[7] 猪股他，“ＰＩＥの構造メモリ試作ハードウェアの推論ユニット側ソフトウェア”，第33回情報全大，5B-3.

[8] UNIX4.2BSD System Interface Manual -System Calls, MMAP(2)

[9] "Am2900 Family 1985 Data Book", Advanced Micro Device, INC. (1985)

[10] 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.

[11] 小池他，“ＰＩＥの単一化プロセッサ ～システム構成～”，第28回情報全大，6F-6.

[12] 湯原他，“高並列推論エンジンＰＩＥの単一化プロセッサと縮退アルゴリズム”，信学技報，EC83-30.