

推論プロセッサ UNIREDIの命令セットの概要

4W-3

島田 健太郎, 下山 健, 清水 剛, 小池 汎平, 田中 英彦

{shimada,ken,shimizu,koike,tanaka}@mtl.t.u-tokyo.ac.jp

東京大学 工学部

1 初めに

我々は現在並列推論マシン PIE64[1]の製作を進めている。PIE64は64台の推論ユニット(Inference Unit: IU)を2系統の回線交換ネットワークで結合した構成を採っており、対象言語としてCommitted Choice型言語FLENGを実行する。各IUにはネットワークと接続しFLENG向きの高度な通信機能を提供するNIP(Network Interface Processor)、IU内でのFLENGの実行処理を行なう推論プロセッサUNIRE、及び全体の管理を行なう汎用プロセッサSPARCがある。我々は先にこの推論プロセッサUNIREについて概要設計を行なったが[4]、更に検討を重ねた結果、先の設計はより一層の効率化が可能であることがわかった。そこで今回言わば第2版の概要設計を行なった[3]。この第2版のUNIRED(UNIREDI)は、FLENGを独自の機械命令にコンパイルしたものを実行する。ここでは、その命令セットを中心にUNIREDIのアーキテクチャを説明する。

2 UNIREDIアーキテクチャ

UNIREDIは並列推論マシンの要素プロセッサとしてCommitted Choice型言語FLENGを効率良く実行するために設計された推論プロセッサである[3]。主な特徴としては、次のような点が挙げられる。

- 記号処理向きの機能としてタグ・アーキテクチャを採用、FLENGの処理を効率化している。
- 命令バス、メモリ読み出しバス、メモリ書き込みバスの三つの分離したメモリバスを持ち、バンド幅の広い並列メモリアクセスを行なう。
- 同時に複数のコンテキストからの命令を受け付ける多重コンテキスト処理[3]の機能を持つ。これによって動的なパイプライン充足率の向上を行なう。
- コプロセッサ・コマンドバスを持ち、NIP、SPARCとの間でのコマンド/リブライの通信プロトコルをサポートしており、これらのプロセッサと協調動作を行なう。

UNIREDIは効率化のため、内部でパイプライン構成が採られている。命令バスの他にメモリ読み出しバス、メモリ書き込みバスを独立して持っていることにより、FLENGのような論理型言語の処理では頻出するRead-Modify-Write型のメモリ操作をパイプラインの1スロットで1クロックで行なうことが可能である。

3 命令セット

3.1 UNIREDI命令の特徴

UNIREDIの命令セットを表1に掲げる。論理型言語の実装によく用いられるWAM型命令[2]との違いは主に、

表1: UNIREDIの命令セット

Dereference	derr dfer dcll dfcv dclv dfcc	dereference dereference and check list dereference and check list and load ca dereference and check vector dereference and check vector and load top dereference and check constant
Execute	exec exel exll excv exvl exct	execute execute on list execute on list and load ca execute on vector execute on vector and load top execute on constant
Manipulate Structure	cpir cpr cvtp cvtr	copy if remote copy if remote with register. check vector top check vector top with register
Load / Store	ld lgld ldst lids st sufd stlm	load tagged load load and store tagged load and store store store undefind code store immediate
Active Unification	bind bdim evos	bind variable bind with immediate check variable order and swap
Heap Allocation	allc alst	allocate allocate and store
Control	jump call jcmp jncp jtag jnig jon joff stop	jump call jump on compare jump on not compare jump on tag jump on not tag jump on flag state on jump on flag state off stop
Coprocessor	cpem susp fall fork	send coprocessor command send suspend command send fall command fork new goal
Set	succ setc seti seta	send success command set constant set mark and tag set alternative pointer
Arithmetic and Logical	add sub and or xor not shl shr asr	add subtract bit-wise and bit-wise or bit-wise exclusive or bit-wise not shift left shift right arithmetic shift right
Comparison	gt geq less leq equal neq	greater than greater than or equal less than less than or equal equal not equal

- FLENGがCommitted Choice型言語であることによる命令機構の簡素化。
- 並列実行に対応した命令仕様。
- パイプラインの構成を意識し、これを有効利用して最適化を行なうための複合命令。
- 逆にハードウェアのコストを下げるためにすべて1命令1ワード構成とし、1クロックで実行するための命令の分解。

等の点である。

3.2 主な命令

3.2.1 デレファレンス命令

UNIREDIに於いてまず特徴的な命令として、変数の参照の手続きを行なうデレファレンス命令がある。この命令は、指定されたレジスタの内容が変数への参照であればその参照先の

メモリ・ワードを読み出して元のレジスタに入れ、自分自身に Jump を行なう。そして再実行された時、先に読み出したものがなお変数への参照であれば再び同じ動作を行なう。読み出した変数が未束縛であれば、オペランドまたは選択点レジスタで指定される飛び先へ Jump を行なう。この時サスペンド処理を高速化するため、未束縛であった変数へのポインタをサスペンション・レジスタへ格納しておく。

このデレファレンス命令には他に、更に変数でない時にリスト等へのポインタであるかどうか調べる命令等があり、その時にはリストでなかった時の飛び先を未束縛変数であった時の飛び先と分けることが可能であり、サスペンド検査の処理を効率化している。

### 3.2.2 構造データ操作命令

リスト、ベクタなどの構造データ操作命令には、幾つかの基本機能を実現する命令及び先のデレファレンス命令との複合命令がある。基本命令としては構造データの要素を取り出すロード (lgld) 命令や、構造データの長さを検査する check vector top 命令の他、リモートな構造データであるかどうか検査して NIP にコマンドを発行しローカル・コピーを作る copy if remote 命令等がある。デレファレンス命令との複合命令では、例えばデレファレンスした結果がリストへのポインタであったら、そこから car 部を読み出してくる dereference and check list and load car 命令等があり、これらは 1 クロックで実行される。

### 3.2.3 バインド命令

UNIREDII では変数が自 IU 内のローカルである場合に限りそのバインド処理を行なう。他 IU のリモートな変数に対しては NIP にコマンドを発行して処理を依頼する。従って、この bind 命令ではまず指定された変数がリモートであるかどうか検査し、リモートな場合には NIP にコマンドを発行する。変数がローカルであるとロックを掛けて読み出しを行ない、まだ未束縛であるかどうか調べて未束縛ならば値をバインドするという操作を不可分に行なう。読み出した変数の値は第 3 オペランドで指定されたレジスタに送る。更に値をバインドした際、その変数でサスペンドしているゴールがあれば、これをアクティベートするために管理プロセッサ SPARC に対しアクティベートコマンドを発行する。

### 3.2.4 コプロセッサ間通信命令

NIP、SPARC と協調動作を行なうためのコプロセッサ・コマンドは上の bind 命令命令のように暗黙に発行される場合と、コプロセッサ間通信命令によって陽に指定されて発行される場合がある。暗黙のコマンド発行によっては、他にデレファレンス命令やロード命令に於いても参照がリモートであると自動的に NIP へのコマンドに変換されて発行され、ネットワークを介したメモリ・アクセスを統一的に実現している。陽に指定されるコプロセッサ間通信命令には、負荷分散の契機となる新しい子ゴールの生成を管理プロセッサ SPARC に通知する fork 命令や、ゴールの実行の終了を通知する send success command 命令、ゴールの実行のサスペンド / フェイルを知らせる send suspend/fail command 命令等がある。

```
append([H | T], X, Y) :- Y = [H | Z], append(T, X, Z).
append([], X, Y) :- X = Y.
```

append/3:

```
seta    $suspend, ap
dcll    r1, $1, r4
$2:     tlds    [r1 + 1], [gtp + 2], r1 ; T]
allc    S, LST, 2, r5 ; [
st      r4, [r5] ; H |
sudf    [r5 + 1], r6 ; Z]
bind    r5, r3, r7 ; Y = [H | Z]
jntg    r7, UDF, $check
st      r6, [gtp + 4], r3 ; Z)
exll    r1, $2, r4 ; [H |
$1:     dfcc    r1, nil, $fail ; []
derf    r2, $3 ; X = Y
$3:     derf    r3, $4
$4:     cvos    r2, r3, r2, r3
bind    r2, r3, r4
jntg    r4, UDF, $check
succ    gtp, mp
```

図 1: append のコンパイル例

### 3.3 コンパイル例

最後に具体的なコンパイル例として、append の例を図 1 に掲げる。この例では、append が再帰的に実行される場合、そのループを構成する命令数は 8 である。

### 4 終りに

UNIREDII は現在詳細設計の段階である。最終的には CMOS ゲートアレイとして実現し、クロックは 10MHz とする予定である。これからの課題としてはシミュレーション等も含めたプロセッサ・アーキテクチャの評価、実際に PIE64 に組み込んだ実機での 64 台の並列動作の評価などが挙げられる。

### 参考文献

- [1] Koike, H. and Tanaka, H.: "Multi-Context Processing and Data Balancing Mechanism of the Parallel Inference Machine PIE64" Proc. of Fifth Generation Computer Systems, Tokyo, Japan, November 1988.
- [2] Warren, D.H.D.: "An Abstract Prolog Instruction Set" Technical Note 309, Artificial Intelligence Center, SRI, 1983.
- [3] 島田, 下山, 清水, 小池, 田中: "推論プロセッサ UNIREDII のアーキテクチャ" 情報処理学会計算機アーキテクチャ研究会 77-2, 1989 年 7 月
- [4] 島田, 清水, 小池, 田中: "並列推論マシン PIE64 の推論プロセッサ UNIRED の概要" 情報処理学会第 38 回全国大会 5U-9, 1989 年 3 月