推論プロセッサ UNIREDII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 UNIRED、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 UNIRED 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(UNIREDII) 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 UNIREDII のアーキテクチャを説明する。

2 UNIREDIIアーキテクチャ

UNIREDII は並列推論マシンの要素プロセッサとして Committed Chioce 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クストからの命令を受け付ける多重コンテクスト処理 [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 / リプ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。

UNIREDII 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 UNIREDII命令の特徴

UNIREDII 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: UNIREDIIの命令セット

Dereference	derf	dereference
	dicl	dereference and check list
	dcll	dereference and check list and load car
	dcv	dereference and check vector
	dcvl	dereference and check vector and load top
	dcc	dereference and check constant
Execute	exec	execute
	exel	execute on list
	exll	execute on list and load car
	exev	execute on vector
	exvl	execute on vector and load top
	exct	execute on constant
Manipulate Structure	cpir	copy if remote
	cprr	copy if remote with register
	cvtp	check vector top
	cvtr	check vector top with register
Load / Store	ld	load
	tgld	tagged load
	ldst	load and store
	tldo	tagged load and store
	st	store
	undf	store undefind code
	stim	store immediate
Active Unification	bind	bind variable
	bdim	bind with immediate
	cvos	check variable order and swap
Heap Allocation	allc	allocate
	alst	allocate and store
Control	jump	jump
	call	call
	jcmp	jump on compare
	jncp	jump on not compare
	jtag	jump on tag
	jntg	jump on not tag
	jon	jump on flag state on
	joff	jump on flag state off
	stop	stop
Coprocessor	cpcm	send coprocessor command
	susp	send suspend command
	fail	send fail command
	fork	fork new goal
	succ	send success command
Set	setc	set constant
	sett	set mark and tag
	seta	set alternative pointer
Arithmetic and Logical	add	add
	sub	subtract
	and	bit-wise and
	or	bit-wise or
	xor	bit-wise exclusive or
	not	bit-wise not
	shl	shift left
	shr	shift right
	asr	arithmetic shift right
Comparison	grt	greater than
	geq	greater than or equal
	less	less than
	leq	less than or equal
	equl	equal
	neq	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

UNIREDII において まず特徴的な命令として、変数の参照の手繰り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

The Inference Processor UNIREDII: Instruction Set Overview
Kentaro SHIMADA, Takeshi SHIMOYAMA, Takeshi SHIMIZU, Hanpei KOIKE, and Hidehiko TANAKA
University of Tokyo

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(tgld)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、これをアクティベイトするために管理プロセッサSPARCに対しアクティベイ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              ; [H |
$2:     tlds    [r1 + 1], [gtp + 2], r1 ; T]
        allc    8, 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