並列論理型言語 FGHC の VLIW 計算機での実行方式

7F−5

坂本 真理子　　木村 康則

(株) 富士通研究所

1 はじめに

本稿では，並列論理型言語 FGHC を VLIW(Very Long Instruction Word) 方式の並列計算機で効率良く実行する方式について論じる．

従来 VLIW 計算機は，Fortran で記述された数値計算プログラムを高速に実行するための並列計算機として位置付けられてきた．一方，FGHC は記号処理用の並列言語である．主にゴールを並列の単位とした実行方式が研究されており，VLIW 計算機のような命令レベルの並列をサポートする計算機での実現例はない．本稿では FGHC の言語の特徴を利用し，FGHC プログラムを VLIW 命令にコンパイルし実行する方式について述べる．

2 VLIW 計算機

VLIW 計算機では，複数の演算 (以下オペレーション) を同時に実行することによりプログラム実行の高速化を狙う．ここで，同時に実行が開始される複数のオペレーションを一つの VLIW 命令と呼ぶ．オペレーションを組み合わせて VLIW 命令を作るのはコンパイラが行う．すなわち，命令スケジューリングが静的 (コンパイル時) に行われるのが，VLIW 方式の特徴である．[5][6]

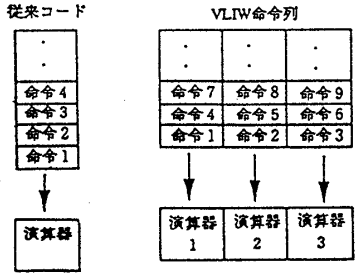

図 1: 従来方式と VLIW 方式の命令実行

図 1で次に実行される VLIW 命令は命令 1, 命令 2, 命令 3 で，これらは同時に実行開始となる．

3 並列論理型言語 FGHC

並列論理型言語 FGHC は，ICOT で設計された言語であり [4]，次のようなシンタックスを持つクローズの集合として表される．

$$H : -G_1, \cdots, G_m | B_1, \cdots, B_n. \ (m \geq 0, n \geq 0)$$

ここで，H, G_i, B_i は，各々，クローズヘッド，ガードゴール，ボディゴールと呼ばれる．オペレータ '|' は，コミットメントバーと呼ばれ，クローズ中でこれに先立つ部分を受動部 (またはガード部)，これに続く部分を能動部 (またはボディ部) と呼ぶ．ここで，ガードゴールとしては，組込述語しか書けない．

ゴール H が与えられると実行が始まる．クローズ群 (候補節と呼ぶ) のうち，H と同じクローズヘッド H を持つもののガード部の実行が試され，成功した場合にそのボディ部のゴール群が実行される．ガード部の実行によって，ゴール H の引数変数を具体化してはならない．具体化するような実行は中断され，他のゴールがその変数を具体化するまで待たされる．実行すべきゴールが無くなると実行が終了する．

4 FGHC 言語の特徴と VLIW 方式

本節では FGHC の処理を VLIW 計算機で並列に実行することを考える．本稿ではゴールと 1 つのクローズとの間の選択実行の各々を並列実行の単位と考える．
並列の可能性は，以下の 2 つである．

1. FGHC のガード部の実行では，実行環境を変えることがない．すなわち，ガード部の実行に失敗して分岐が発生した場合に環境を戻す (トレーススケジューリングの場合の辻褄合わせ) 必要がない．この性質を利用して，ガード部のユニフィケーションを出来るだけ並列に行うことを考える．

2. ボディ部の実行では，ボディゴールを生成するが，このゴール間に直接の依存関係はない．これよりボディゴールの生成を並列にすることを考える．具体的には，ボディゴールの引数の並列準備である．引数にリストなどの構造体が現れる場合には，その生成も並列に行う．

図 2 にガード部のユニフィケーションを並列に行うためのコンパイル例を示す．この図では，四つのゴール引数のユニフィケーションが四つのオペレーションフィールドから構成される VLIW 命令の各フィールドに置かれて，並列に実行されている．ここで，code(Hi) はクローズ H の第 i 引数 Hi のユニフィケーションの処理コードである．また，図 3 にボディ部のユニフィケーションを並列に行うためのコンパイル例を示す．この図では，三つのボディゴールを生成する処理と，ボディゴールの引数として与えられるリスト構造の生成が並列に行われている．code(bi) は，各々ボディ部のゴール $b1, b2, b3$ を生成する

Implementing FGHC on VLIW machine
Mariko Sakamoto(mariko@flab.fujitsu.co.jp),
Yasunori Kimura(ykimura@flab.fujitsu.co.jp)
Fujitsu Laboratories Ltd.

実行するゴール： ?- (A1,A2,A3,A4).
プログラム： g(H1,H2,H3,H4):- guard | body.
コンパイルコード：

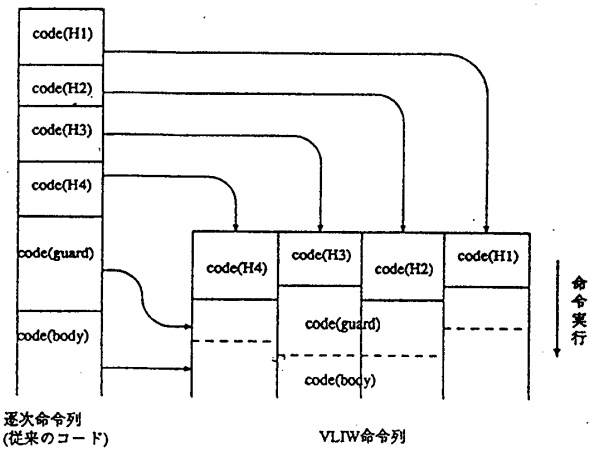

図 2: ガード部の並列実行

ためのコードである．$b2$ の第2引数はリスト構造で，この構造は独自に生成されゴール $b2$ 生成時に使用される．

5 効果

本方式の効果を調べるために，簡単なベンチマークプログラムについて，VLIW 命令列を生成した場合と，逐次実行のコードで，静的なサイズを比較した．結果を表 1 に示す．この計算は，プログラムを一旦，KL1B[3] と呼ぶ抽象命令セットにコンパイルし，そのコード列を Trace 計算機 [1] の命令列に変換するという手順で行った．

表 1: コードサイズ減少率

プログラム	コードサイズの減少率
qsort	5
queen	15

減少率が小さいのは，使用したプログラムのゴールの引数個数とボディ部のゴール数が少ないためである．例えば，各々のプログラムの平均引数個数は queen が 5.4，qsort が 3 であるのに対し，このプログラム内の引数個数 7 のクローズでは 39% の減少となっている．引数個数とゴール数の多いプログラムに対してはより高い効果が期待できる．

6 おわりに

FGHC の言語の特徴を利用して VLIW 計算機で効率良く実行する方式について概略を述べた．本方式では，トレーススケジューリング [2] におけるトレースの選択，'辻褄あわせ' のコードの生成などが不要になり，コンパイラの処理が軽くなっているのが特徴である．なお，本稿ではデリファレンス処理については述べなかった．デリファレンスの段数は一般には予測不可能であるので，本方式ではガード部の実行時に不都合が生じる可能性がある．これを解決するデリファレンス処理の実現手法の詳細については，別の機会に報告する予定である．

実行するゴール： ?- (A1,A2,A3).
プログラム： g(H1,H2,H3):- guard | b1(H1),b2(H2,[a,b,c]),b3(H3).
コンパイルコード：

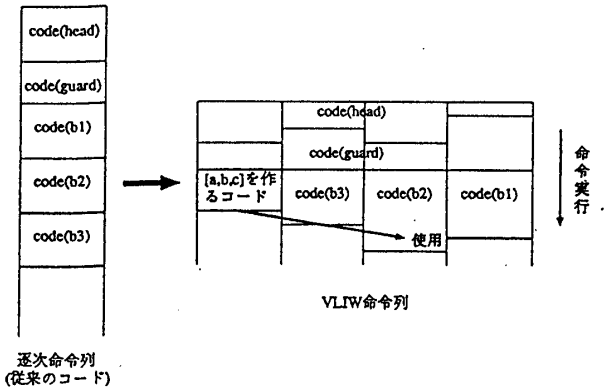

図 3: ボディ部の並列実行

謝辞

日頃ご指導頂く，川戸部長，服部部長代理，篠木室長，熱心に討論して頂いた人工知能三研の皆様に感謝します．

参考文献

[1] Colwell,R.P., Nix, R.P., O'Donnell,J.J., Papworth,D.I and Rodman,P.K. : A VLIW Architecture for a Trace Scheduling Compiler. IEEE Trans. on Computer, Vol.37, No.8, pp.967-979, Aug., 1988.

[2] Fisher,J.A. : Trace Scheduling : A Technique for Global Microcode Compaction. IEEE Trans. on Computer, Vol.C-30, No.7, pp.478-490, July, 1981.

[3] Kimura,Y., and Chikayama,T. : An Abstract KL1 Machine and its Instruction Set. Proc. *Symp. on Logic Prog. '87*, pp.468-477, Aug. 1987.

[4] Ueda,K. : Guarded Horn Clauses. Ph.D. Thesis, Information Engineering Course, Univ. of Tokyo, 1986.

[5] 中田 登志男： VLIW 計算機のためのコンパイラ技術. 情報処理学会誌 Vol.31, No.6, pp.763-772, June, 1990.

[6] 村上 和彰： スーパスカラプロセサの性能を最大限に引き出すコンパイラ技術. 日経エレクトロニクス 1991 年 3 月 4 日号 (No.521), pp.165-185.