ループ展開・ソフトウエアパイプライニングの新手法

2F－3

境隆二 ， 竹内陽一郎

(株) 東芝　情報処理・機器技術研究所

1．はじめに

ＶＬＩＷ計算機のコンパイラは、命令の並列性を抽出し、スケジューリングする、という並列最適化を行う。このときより並列度の高いコードであるほど、プログラムの実行速度が速くなる。本稿では、最適化の最も効果の大きい、ループの最適化として、ループアンロールとソフトウェアパイプライニングのそれぞれの利点を生かした方法を提案する。

2　アンロールとパイプライニングの問題点

ループアンロールとは、ループを何回か展開することで、並列実行可能な命令数を増やし、並列度を高める手法である（図４，図５）。またソフトウェアパイプライニングとは、図６に示すように、ループの各繰り返しをハードウェアのパイプラインのように、次々と実行させるための手法である。

ループアンロールでは、アンロールしたループ（以下大ループと呼ぶ）の境界ごとに、命令の並列度が低下する（図５）。このため、ｎ回ループアンロールした場合と、完全にソフトウェアパイプライン化した場合とでは、実行サイクルで、ｎ＋ｋ：ｎ（ｋ＞０は定数）の差がある。この差を小さくするためには、ｎを大きくすればよいが、コードサイズが大きくなりすぎるという問題がある。さらに、アンロールする事で命令を広範囲に配置できるようになる反面、レジスタのliveレンジが広くなりすぎて、レジスタスピルコードを挟まなくてはいけなくなるということも起こり得る。

一方、図４に示すように、ループアンロールすることで、ＤＡＧが大きくなるので、ハイトリダクションが適用出来る可能性がある（図７）。ソフトウェアパイプライニングでは、各繰り返しが全く同じ形のコードでなければならないという制約のために、ハイトリダクションの機会を失うことになる。また、ＩＦ－ＴＨＥＮ－ＥＬＳＥ等の処理がループ内にある場合、それらを効率的にかつ効果的にパイプライン化するアルゴリズムはまだ現実的ではない。

```
for (i=0; i<Loop; i++)
 {
  t = (c[i] + d[i]) * t;
  x[i] = t + a[i];
 }
```

図１ ソースプログラム

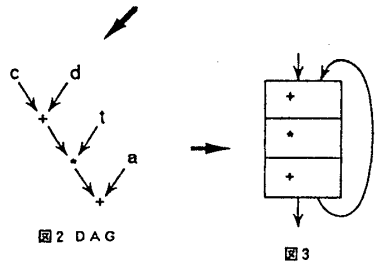


図２ ＤＡＧ

図３

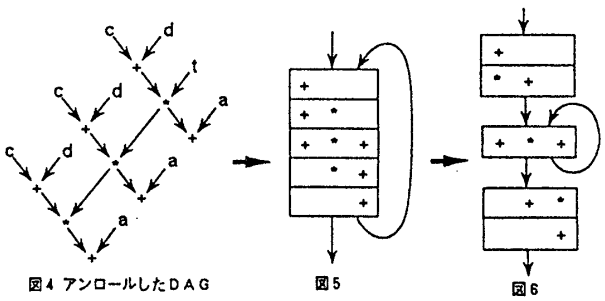


図４ アンロールしたＤＡＧ

図５

図６

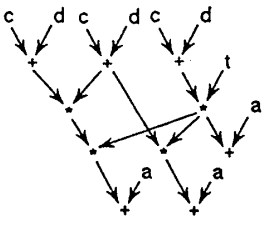


図７ ハイトリダクションした ＤＡＧ

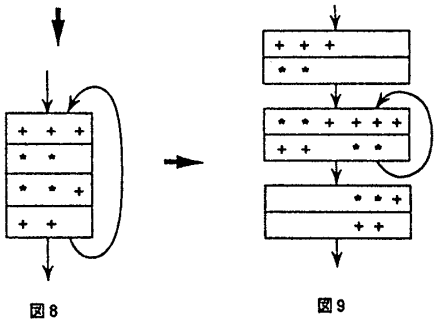


図８

図９

New method in software pipelining

Ryuji SAKAI ,　Yoichiro TAKEUCHI

TOSHIBA CORPORATION

3　効率的なパイプライン化のアルゴリズム

従来のループパイプライン化のアルゴリズムには、ループの各iterationを投入する間隔を、最初に決めてスケジュールするモジュロスケジューリング法と、ループを数回展開してから、同じパターンを見つけて１つにまとめるパーフェクトパイプライニング等がある．どちらも、命令の並列スケジュールとパイプライニングが非独立であるために、１．並列スケジュールの最適化を十分行えない　２．アルゴリズムが複雑になる　という問題がある．ここでは、より柔軟な最適化が出来るようにＲＩＳＣコンパイラのディレイドスロット最適化の拡張である以下の方法を説明する．

ｽﾃｯﾌﾟ１：ループボディのリストスケジュール

ループボディの命令を出来るだけ上へ押し上げるようにして並列スケジュールする

ｽﾃｯﾌﾟ２：ｌｉｖｅレンジの縮小

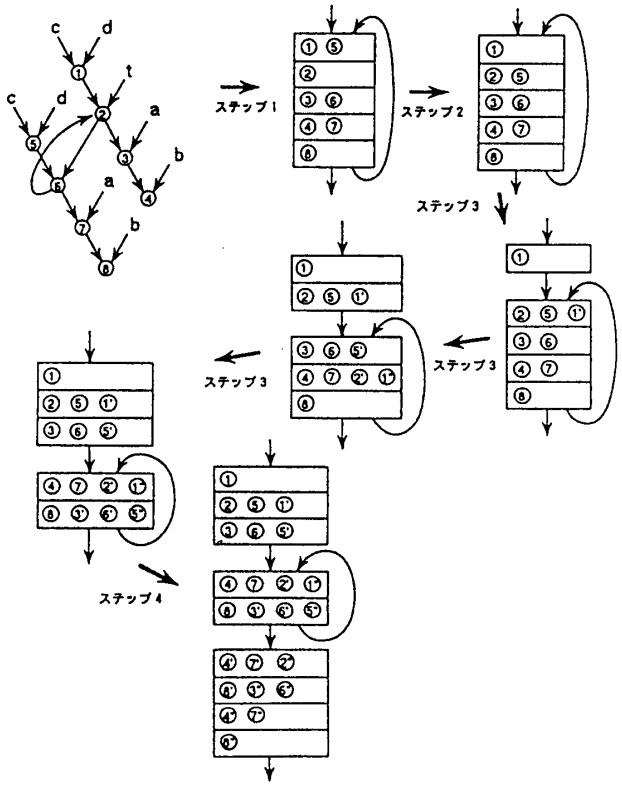
ｽﾃｯﾌﾟ１で押し上げ過ぎた命令（ｸﾘﾃｨｶﾙﾊﾟｽにない命令）を押し下げる

ｽﾃｯﾌﾟ３：パイプライン化

ループの先頭のＶＬＩＷ命令を単位命令に分解しループの底から押し上げる．

ループの先頭を次の命令にする押し上げようとする命令が、ループの底からあふれるまでｽﾃｯﾌﾟ３を繰り返す

ｽﾃｯﾌﾟ４：ループの後処理をつけ加える



4　ループ最適化の基本戦略

ループアンロール、ソフトウェアパイプライニングの問題点の解決策として、数回アンロールしたループを、１つのループと見なしてパイプライン化する、という方法を考える．すなわちループアンロールの欠点であった、大ループ境界の並列度低下を、大ループに３のアルゴリズムを適用することにより、回避しようとするものである．この方法だと、最初に数回ループアンロールするので、ループの繰り返しをまたいだ最適化（ﾊｲﾄﾘﾀﾞｸｼｮﾝなど）を行うことが出来る．さらに、３で述べたアルゴリズムは大ループの境界のみでなく、各iterationをオーバーラップさせることができるので、より並列度の高い、高速なコードを生成する事が出来る．図１のプログラムをこの方法に従って最適化したものを図８に示す．これは、アンロールせずにそのままソフトウェアパイプライニングする（図６）よりも１．５倍高速なコードである．

5　まとめ

ＶＬＩＷ方式では、ハードウェアのコストの削減という意図から、命令語の各フィールドに配置できる命令の種類を制限することがある．３で示したアルゴリズムを使うと、このハードウェアの制約を容易にを組み込むことか出来る．またループアンロールの問題点は、コードサイズの増大とループ境界の並列度の低下であった．４で述べた方式を採用すると、コードサイズの増加は一定量で抑えられ、ハイトリダクションや、ＩＦ－ＴＨＥＮ－ＥＬＳＥを含んだ処理の最適化に有効であり、さらに３のパイプライン化を行うことで、ループ境界を張り合わせることが出来てループ処理の並列度が向上する．

参考文献

（１）A.Aiken and A.Nicolau "Perfect Piplining: A New Loop Paralleliz-ation Technique", In Proceedings of the 1988 European symposium on Prog-raming Technique, pp.221-235

（２）B.Ramakrishna Rau, David W.L.Yen, Wei Yen, and Ross A.Towle "TheCydra 5 Departmental Supercomputer Design Philosophies, Decisions, and Trade-offs",IEEE Computer,Vol.22,No.1,Jan.1989, pp.12-35

（３）中谷：" ＶＬＩＷ計算機のためのコンパイラ技術" 情報処理，Vol.31,No.6,pp.763-772 [1990]