4D－4

条件分岐を含むループの最適なソフトウェア・パイプライニング

糸賀 裕弥 (筑波大学) †　秋川 友宏 (筑波大学) †
山下 義行 (筑波大学) ††　中田 育男 (図書館情報大学)†††

1. はじめに

ソフトウェア・パイプライニング[1]は命令レベル並列化方法として実用段階である。しかしループに条件分岐が含まれる場合のそれは、一般的とはなっていない。

我々は Enhanced Modulo Scheduling(EMS と略す)[2]に注目し、EMS をさらに改良した方法 (改良 EMS と略す)[3]を提案した。これらは、条件分岐の状態に応じて複数のコードを準備する方法である。実行効率は高いが、複数のコード間の遷移が複雑であるため、レジスタ割付や命令スケジューリングの方法が従来よりも複雑になると考えられる。

本稿では、通常の RISC アーキテクチャ及びスライドウィンドウ・アーキテクチャ[4]を対象とし、改良 EMS によるコードに対しての、干渉グラフによるレジスタ割付方法を提案する。

2. 改良 EMS による命令スケジューリング

図 1 のプログラムに、改良 EMS を適用し命令スケジューリングを行なう。紙面の都合により改良 EMS についての詳しい説明は省く。具体的なスケジューリングの手順については該当論文[2][3]を参照されたい。

本稿で想定するプロセッサは、同時発行命令数 1、パイプライン実行可能。命令のレイテンシが load/mult・add・その他の命令がそれぞれ 3・2・1mc であるとし、遅延分岐は考えない。定数 1～4 はレジスタ R1～R4

```
DO I=1, 1000
    IF ( A(I) > 1.0 ) THEN
        C(I) = 2*( A(I) + 1 )**2 + 3
    ELSE
        C(I) = A(I)**2 + 4
```

図 1 例題プログラム

```
L_TT: Check_Loop_Index   L_TF: Check_Loop_Index
      BEQ   TT_Epilogue        BEQ   TF_Epilogue
 >>   MULT  Rq,R2,Rr      >>   MULT  Rq,R2,Rr
      ADD   Ra,R1,Rp           MULT  Ra,Ra,Rb
      LOAD  A(I),Ra            LOAD  A(I),Ra
      ADD   Rr,R3,Rs           ADD   Rr,R3,Rs
      MULT  Rp,Rp,Rq           ADD   Rb,R4,Rc
 >>   STORE Rs,C(I-2)     >>   STORE Rs,C(I-2)
      CMPR  Ra,R1              CMPR  Ra,R1
      BGT   L_TT               BGT   L_FT
      BRA   L_TF               BRA   L_FF

L_FT: Check_Loop_Index   L_FF: Check_Loop_Index
      BEQ   FT_Epilogue        BEQ   FF_Epilogue
      ADD   Ra,R1,Rp           MULT  Ra,Ra,Rb
      LOAD  A(I),Ra            LOAD  A(I),Ra
      STORE Rc,C(I-2)          STORE Rc,C(I-2)
      MULT  Rp,Rp,Rq           ADD   Rb,R4,Rc
      CMPR  Ra,R1              CMPR  Ra,R1
      BGT   L_TT BGT           L_FT
      BRA   L_TF BRA           L_FF
```

図 2 改良 EMS でのスケジューリング

にあらかじめ設定されているとする。

命令スケジューリングの結果を図 2 に示す。これは、ベクトル計算機のマスクビットのような機構を持たないアーキテクチャで実行させるために、逆 IF 変換[5]したものであり、パイプラインのステージごとの真偽の状態を 4 種類のコードで表している。実際には条件分岐の真偽の状態によって、繰り返しごとに 4 種類のうちいずれか 1 つのコードが適宜選択され実行される。

>>で示された命令がリソース予約テーブル外にスケジューリングされた命令である。例題プログラムの else 部 (図 2 の下の 2 状態のコード) は、EMS の場合よりも Initiation Interval が小さくなっている。

3. 干渉グラフによるレジスタ割付

レジスタ割付問題には、干渉グラフとその彩色アルゴリズムが良く利用される。コード中の変数をグラフ上の頂点として表現し、同時に存在する変数の頂点同士を干渉枝で結ぶ。このグラフに彩色を行ない、色ごとに異なるレジスタを割付る。

図 3 は、4 種類の状態のコードそれぞれに対する干渉グラフである。ここで、Ra～のように～が変数の右側に付された頂点は、頂点の存在する状態でその変数が定義され、別の状態でその変数が使用されること

Register allocation methods for Optimal Software Pipelining for Loops with Conditional Branches

† Hiroya ITOGA, Tomohiro HARAIKAWA, Doctoral Program in Engineering, Univ. of Tsukuba
†† Yoshiyuki YAMASHITA, Inst. of Info. Sci. and Elec., Univ. of Tsukuba
††† Ikuo NAKATA, Univ. of Library and Info. Sci.

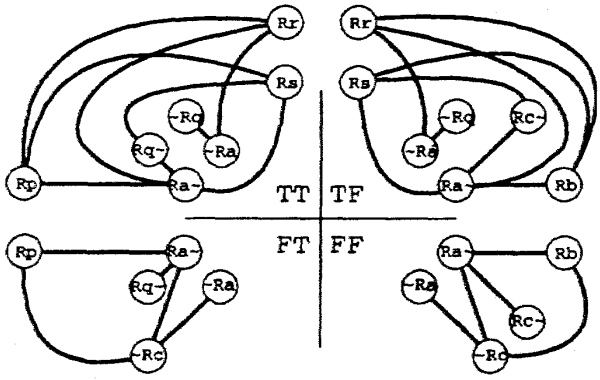

図3　状態別干渉グラフ

を表現している。~Raはその逆である。

通常のRISCアーキテクチャでは、Ra~と~Raには同じレジスタを割付なければならないが、問題の性質を明らかにするため、あえて別の頂点として表す。

3.1　制約付き彩色法

制約付き彩色法は、Ra~のように状態の遷移にまたがる頂点に対して、同名の頂点であれば同一のレジスタに割付けるという制約を設け、4種類の干渉グラフそれぞれに独立して彩色を行なう方法である。

この方法は一般的なレジスタ彩色法と本質的には変わらない。しかし、パイプラインステージ数や処理の内容に関係して、コードの状態の種類や変数の数が増えるに従い、頂点数も増えてしまうという欠点がある。

3.2　重ね合わせによる彩色法

図3の4つの状態のコードは、全く異なる干渉グラフを与えるわけではない。改良EMSなどの命令スケジューリング方法の性質から、4つの状態のグラフには互いに共通する部分グラフがしばしば含まれる。

この性質を利用し、4つの状態のコードそれぞれにおける同名の頂点全てを、同一の頂点として考え、図4のようにグラフを重ね合わせる方法が考えられる。これが重ね合わせによる彩色法である。

彩色の複雑さを決める要因としては、干渉枝の数よりも頂点の数が大きく影響していると考えられる。つまり、図4は複雑に見えるが、図3よりも彩色に関しては容易であろうと推定できる。

ただし彩色が最適である場合においては、制約付き彩色法と重ね合わせによる彩色法を比較したとき、後者の場合に必要な彩色数、すなわち必要レジスタ数の増える場合が存在することが判明している。しかし、実際には彩色の最適化は実現が困難であるため、多くの場合において後者が有利であると我々は考えている。

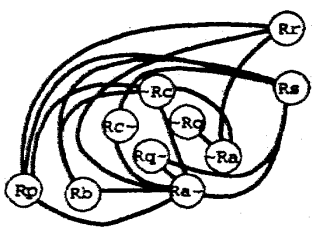

図4　状態別干渉グラフを重ね合わせる

3.3　スライドウィンドウにおける彩色法

スライドレジスタ干渉グラフを用いたレジスタ割付方法[6]と同様に、本稿におけるRa~と~Raにレジスタ名の変化情報による制約を与えて彩色を行なうことで、以上の議論は、スライドウィンドウ・アーキテクチャ[4]に対してもそのまま適用可能である。

特に重ね合わせによる彩色法を適用した場合には、通常のアーキテクチャで彩色が最適である場合に起こりうる、必要な彩色数が重ね合わせによって増えてしまう問題が、その原理上生じにくいことが判明している。この詳細は別途報告する予定である。

4.　まとめと今後の課題

これまで、条件分岐を含むループのソフトウェア・パイプライン化のレジスタ割付方法はほとんど知られていなかった。だが、今回の方法を用いることで効率的に可能であるということが判明した。

今後は各種ベンチマークなどを用いて、改良EMSのスケジューリング及びレジスタ割付方法について、より詳しい調査を重ねていく。

参　考　文　献

1) Allan, V. H., Jones, R. B., Lee, R. M. and Allan, S.J.: Software Pipelining, *ACM Computing Surveys*, Vol. 27, No. 3, pp. 367–432 (1995).

2) Warter, N.J., Haab, G.E. and Bockhaus, J.W.: Enhanced Modulo Scheduling for Loops with Conditional Branches, *Proc. of the 25th Annu. Int. Sym. on Microarchitecture(MICRO-25)*, pp. 170–192 (1992).

3) 山下義行, 中田育男: ループ中に条件分岐を含む場合の最適なソフトウェア・パイプライニング, 並列処理シンポジウム JSPP'94, pp. 17–24 (1994).

4) 位守弘允, 中村宏, 朴泰佑, 中澤喜三郎: スライドウィンドウ方式による疑似ベクトルプロセッサ, 情報処理学会論文誌, Vol.34, No.12, pp.2612–2623 (1993).

5) Warter, N. J., Mahlke, S. A., Hwu, W. W. and Rau, B. R.: Reverse If-Conversion, *Proc. of the ACM SIGPLAN 1993 Conf. on Prog. Lang. Desi. and Impl.(PLDI)*, pp. 290–299 (1993).

6) 添野元秀, 山下義行, 中田育男: スライドウインドウを考慮したレジスタ割り付け, 情報処理学会第52回全国大会講演論文集 (5), pp. 13–14 (1996).