PNコンピュータのパイプラインストールの要因分析と対策*

3K－6

小林 勇志† 岡本 秀輔‡ 曽和 将容§
電気通信大学大学院 情報システム学研究科¶

1 はじめに

我々の研究室で数年来、研究してきたパイプラインPNプロセッサ[1]を今回試作した。

本報告では今回設計したPNコンピュータの特に演算ユニット(AU)についてのパイプラインストールの起きる要因と起きた時の対処について述べる。

2 PNコンピュータの構成

試作するにあたりピン数などの制限により、従来のPNコンピュータにあったレジスタ間比較命令を実行する分岐制御ユニット(BU)をなくし、代わりに新たに自命令のみならず、他のユニットの命令をメモリからフェッチしてきて、命令キューに格納するフェッチユニット(FU)と命令キューを付加したPNコンピュータをパルテノンシステムを用いて設計した。構成を図1に示す。

今回設計したPNコンピュータは、演算ユニット(AU)、転送ユニット(TU)、フェッチユニット(FU)の3ユニットと、AU用とTU用の2つの命令キューで構成される。

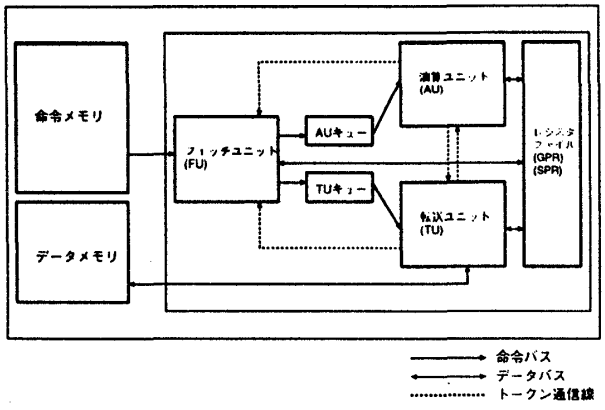

図1: PNコンピュータの構成

また、ユニット間のデータの授受は各ユニットが同時にアクセス可能なマルチポートレジスタの、レジスタファイルを介して行なわれる。

3 パイプライン構成

AU、TUのパイプライン構成は以下の4ステージである。

- AU(演算ユニット)、TU(転送ユニット)
  IF 命令フェッチ(キューから)
  ID 命令デコード、レジスタ読み込み、トークン受信
  EX 命令実行
  WB レジスタ書き込み、トークン送信

- FU(フェッチユニット)
  厳密にはパイプライン処理は行なわれていない

4 ストール要因

AU、TUのパイプラインストールの要因としては以下のものが挙げられる。

1. 命令キューがempty
2. 他のユニットのトークンカウンタがfull
3. トークン待ち
4. ハザード(RAW)

1、2の検出は毎クロックごと命令キューおよび他のユニットで行なわれ、起こった場合は制御入力端子によって知らされフェッチを止める。

3、4の検出はパイプラインのデコードステージで行なわれ、起こった場合は制御内部端子によって自分自身から知らされフェッチを止める。

つまり1、2に関してのは検出は他のユニットが行なうので、対処としてはただ命令キューがemptyでなくなり、他のユニットのトークンカウンタがfullでなくなるのを待つだけである。だから他の要因と同時に起こったとしても、これといって特別の対処を行なう必要はない。

しかし3、4に関しての検出は自ユニットが両方ともデコードステージで行なうので、3、4が同時に起こっ

*An analysis of pipeline stall on Parallel Neumann computer
†Kobayashi Yuuji
‡Okamoto Syusuke
§Sowa Masahiro
¶Graduate School of Information Systems,The University of Electoro-Communications

た時などパイプラインの状態やトークンカウンタの状態などで対処の仕方が変わるので大変である。

あと、ここで特に3について述べておく、PNコンピュータの各ユニットは独立に動作しているので、ユニット間で実行する命令に依存関係がある時はトークンを送受信することで、その依存関係を保っている。だから、パイプラインストールの要因にトークン待ちがあるのはPNコンピュータの特徴である。

5 検出と対処

ここでは特にAUについての検出と対処について述べる。AUのブロック図は図2に示す。

検出の仕方は、トークン待ちの場合はフェッチしてきた命令の受信タグがあるかどうかと同時にトークンカウンタ(TC)が0でないかを調べることで行なう。つまり、受信タグがありTCが0でない時のみトークン待ちである。ハザード(RAW)の場合はデコードステージの命令のソースレジスタと実行ステージの命令のデスティネーションレジスタが同じかどうか調べることで行なう。しかし、ソースレジスタのレジスタ番号が0の時は比較をしない。なぜなら、レジスタ0の値は常に0だからである。また、今回のPNコンピュータで使用しているレジスタファイルでは、読み書きが同じレジスタに同時に起こったときはデータのバイパスが行なわれるので、レジスタ読み込みが行なわれるデコードステージの命令のソースレジスタと、レジスタ書き込みが行なわれるライトバックステージの命令のデスティネーションレジスタの比較をする必要はない。

基本的な対処の仕方としては、トークン待ちの場合はトークンが送られてTCが1になるまでデコードステージで待つ、トークンが送られてきたら(TCが1になったら)、TCを1デクリメントして次の実行ステージへ進む。ハザード(RAW)の場合はただ単にデコードステージで1クロック待って、実行ステージの命令のデスティネーションレジスタのレジスタ番号を記憶しているレジスタ(op_e_dest)の値を0に書き換え、次の実行ステージへ進む。

対処の仕方で大きな違いは待ち時間にある。トークン待ちの場合はトークンが送られてきていなければ、トークンが送られてくるまでひたすらデコードステージで待つ。つまり、トークン待ちの場合は待ち時間が一定でない。それに比べてハザード(RAW)の場合はデコードステージで一定時間(1クロック)待ちさえすれば必ずハザード(RAW)は回避できる。

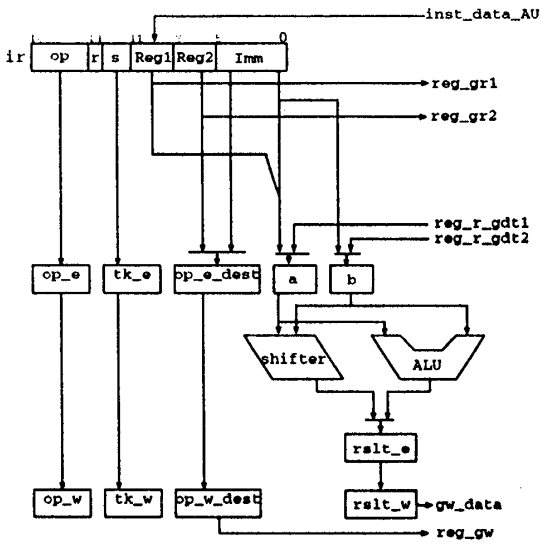

図2: AUブロック図

6 おわりに

今回設計したAUにおいて、トークン待ちをした命令はハザードを起こさない。また、ハードウェア的には、AU内にバイパス機構を設ければハザードは起こらない。しかし、PNコンピュータに特有のトークン待ちをなくすことはできない。だから、トークン待ちの時間をできるだけ短くするとか、待っている時間をどう有効に使うかとかが今後の問題であろう。

パルテノンシステムを開発し貸与して下さった、NTTコミュニケーション科学研究所の方々にお礼を申し上げます。

参考文献

[1] 笠田 洋和 "パルテノンシステムを用いたパイプライン PN プロセッサの設計" 名古屋工業大学 平成4年度卒業論文

[2] 壷井 彰久 "PN スーパースカラプロセッサのパルテノンシステムによる LSI 設計" 情報処理学会第48回全国大会

[3] John L. Hennessy & David A. Patterson "Computer Architecutre" Morgan Kaufmann Publishers 1990