計算機アーキテクチャ 110−11
設 計 自 動 化 73−11
（1995. 1. 20）

パイプライン計算機における性能設計支援手法

田村恭久、西田隆一、長尾聰行、伊藤 潔
上智大学 理工学部

パイプライン計算機における性能評価・性能改善手法PIPEDIETSを提案する。パイプラインの論理を与えられた時に、遅延時間のボトルネックを判定し、セルの移動やパイプラインステージの増減などの性能改善を実施する際のアルゴリズムを考案した。セル移動した場合の機能の等価性を保証するため、まとめて移動する必要のあるセルを判定する。セル移動では2種類の手法を考案した。1つは全ステージの遅延時間の状況を把握し、大局的に性能改善を行なう。もう1つは最も著しいボトルネックから順次改善を行なう。これらは対象とするアーキテクチャの論理規模に応じて使い分けるのが好ましい。

Performance Design Assistance for Pipelined Architecture

Yasuhisa Tamura, Ryuuichi Nishida, Toshiyuki Nagao, Kiyoshi Itoh
Faculty of Science and Technology, Sophia University
Kioi-cho 7-1, Chiyoda-ku Tokyo 102 JAPAN
{ytamura, itohkiyo}@sophia.ac.jp

This paper proposes a performance design assist method for pipelined architecture, named PIPEDIETS (PIpelined architecture PErformance Diagnosis / Improvement ExperT System). PIPEDIETS diagnoses performance bottlenecks of a given pipelined logic, and suggests appropriate improving plans of logic cell movement or pipeline stage increase / decrease. In order to assure the functional consistency for the cell movement, PIPEDIETS identifies cells that should be moved together. PIPEDIETS supports two types of plans for cell movement, the former diagnoses the target of whole stages and suggests comprehensive improving plans, while the latter focuses the worst bottleneck and suggests plans of microscopic view. These plans will be used alternatively according to the scale of the target logic.

1．はじめに

ソフトウェアの多機能化やマルチメディアへの対応にともない、より高性能な計算機ハードウェアが市場で求められている。これに応えるため、パイプライン方式などハードウェアの論理方式レベルでの高性能化方式が多数提案されている。

こういったハードウェアの高性能化方式やそれらの性能評価方式を設計の現場で適用する場合には、単にそれらの方式を実現するだけではなく、性能評価の結果をもとに、設計者が持つ経験的な知識を適用して性能を改善し、ハードウェア論理を変更していく作業が行なわれている。この一連の性能改善のプロセスは、高性能な製品を開発するためには重要であるにかかわらず、設計現場のノウハウや開発環境に依存する割合が大きいため、研究テーマとして取り上げられなかったのが実情である。

本稿の対象とするパイプライン計算機は、機械語命令の実行に必要な機能を複数のステージに分割し、各々のステージが命令をバケツリレーのように順次実行・受け渡しをすることによって高速化を図っている。このパイプライン計算機の性能改善を行なう場合、性能上のあい路は各ステージの中、特定のステージなど随所に存在する可能性があり、またステージ間の性能バランスが悪いとこれが性能上のあい路になる可能性もある。このように、パイプライン計算機の性能改善は複雑な作業となるため、人手による作業を行なうためには相当年数の設計経験が必要である。

本稿では、高性能な計算機ハードウェアで頻繁に利用されるパイプライン計算機を対象とし、この性能評価ならびに性能改善を行なうプロセスを明らかにし、このプロセスに好適な設計支援環境PIPEDIETS (PIpelined architecture PErformance Diagnosis / Improvement ExperT System)を提案する。PIPEDIETSによる性能改善は次のステップを経て行なわれる。

(1)VHDL記述から性能評価に必要なパラメータを抽出

パイプライン計算機の性能評価にはガントチャートが好適である。VHDL記述からガントチャートを生成するため、論理回路からなる信号の経路を数え上げ、その全てについて信号遅延時間を計算する。信号の経路は小規模のLSIでも数万本にのぼるので、この作業は人手では不可能である。そこでVHDL記述から自動的に経路と遅延時間を計算する。

(2) 性能評価シミュレーション

ガントチャートをもとに性能評価シミュレーションを行ない、性能上の不具合やあい路を検出するための総合的なデータを収集する。

(3)性能上の不具合やあい路などの問題点を検出

性能評価シミュレーションをもとに、性能上の不具合やあい路を検出する。パイプライン計算機における性能上のあい路は、前述したように随所に存在し、またパイプラインステージ間の性能アンバランスもあい路として検出する。これを自動化することにより、性能改善全体に必要な時間を大幅に圧縮することができる。

(4) 不具合やあい路に対する改善案を提示

性能上の問題を、エキスパートシステムで診断・改善が可能な形式に変換し、性能改善の案を提示させる。改善の一般的な指針は、論理設計の専門家が経験的な知識として持っている。本研究では、それらの知識をエキスパートシステム中に取り入れ、実際の不具合やあい路に対して適用する段階でエキスパートシステムの推論エンジンを利用する。この推論過程においては、ある変更が他の箇所に悪影響を及ぼしたり、また改善が全く効果を示さないことを検出する必要もある。そこで、問題のない改善案を提示するため、適用後どうなるかを想定した性能シミュレーションも行なう。

(5) 改善案に従って論理記述を変更

性能改善はガントチャートのレベルで適用されるだけでなく、VHDL記述に反映する必要がある。このため、改善に基づいたVHDL書き換えの規則を開発し、これをもとにVHDLを書き換える。

以上のプロセスを実際のハードウェア設計に適用することにより、従来人手により多大な時間をかけて行なっていた性能改善を、自動あるいは半自動で短時間に実行可能になる。これは、計算機ハードウェアの設計時間の短縮に大きく寄与する。

PIPEDIETSでは、改善のプロセスに応じてハードウェア論理記述を次の3側面、すなわち(A)ハードウェアの論理記述の側面、(B)性能評価の側面、(C)エキスパートシステムの対象となるデータの側面、から捉えている。ソフトウェアやシステムの開発技法として最近注目を浴びつつあるドメイン分析手法[1]では、このように解きたい問題の種類に応じたモデル化が有効である、という主張に立っている。このドメイン分析の立脚点をハードウェア開発に適用することによって、ハードウェアの性能設計を効率的に行なえる。前述の性能改善プロセスを、この3側面の記述の変換として捉えたものが、図1である。

本稿における計算機ハードウェアの論理記述は、近年標準化が進んでいるハードウェア論理記述言語であるVHDL (Very high speed integrated circuit Hardware Description Language) を採用

する。従来の絵や図式によるハードウェア記述は組織や事業所によるローカルな規則が多く、一般的な記述法を定めにくい。VHDLは標準化が設計現場にまで浸透しつつあり、ユーザ数が近年増加している。

改善案に従って論理記述を変更するプロセスでは、自動あるいは半自動でVHDLを書き換える。設計者が論理不良などを修正する場合もVHDLの書き換えを行なうことになるので、VHDLの記述という共通のプラットフォームを設計者と性能改善ツールが共有できる。これにより、設計における円滑な作業が可能となる。

以下では、性能改善の手法に焦点をあて、特に2種類のセル移動の手法を中心に説明する。また、論理の等価性を保証するため同時に移動させる必要のあるセルを判定する方法について言及する。

2．パイプライン計算機とその性能モデル

パイプライン計算機は、CPUなどが実行する一連の処理を複数のステージに分割し、それらのステージが並行動作することで高性能化を達成するハードウェア及びその方式である。通常の計算機ハードウェアを同じクロック周波数で5段のパイプライン構造に変更した場合、各々のステージが連続した5個の命令を並列に処理するため、理想的な状況では処理能力は5倍に増加する。各々のステージは、クロックのタイミングと同期して信号を次ステージに送るラッチによって区切られている。各々のステージは、様々な論理回路によって構成されているが、本稿ではこの論理回路の最小構成単位をセルと置く。セルはトランジスタ数個～数十個から成り、AND、OR、1ビット加算などの基本的な機能を実現する。各々のステージは、ステージの開始ラッチから終了ラッチ（これは次ステージの開始ラッチでもある）までがセル群によって接続されており、これらの接続により多数のパスが構成される。

信号がセルの入り口から出口までを通過する際に、遅延時間が生ずる。この遅延時間はセルによって異なり、数値はセルライブラリ中にパラメータとして与えられるものとする。

パイプライン計算機では、その動作周波数の逆数として各ステージのクロックタイムが決定される。このクロックタイムと各ステージの信号遅延時間の関係により、パイプライン計算機の性能上の問題は、以下の3種類に分類される。

(P1)あるステージの遅延時間が過大で、動作不良を起こす。

(P2)全ステージ・全パスの遅延時間はクロックタイム内に収まっているが、あるパスの遅延時間ががほぼクロックタイムに等しく、他のステージのパスの遅延時間には余裕があるのに、クロックタイムを短縮できない。

(P3)全てのステージの遅延時間がクロックタイムに対し余裕がある。

(P1)は動作不良の原因なので、商用のシステムで起こると顧客のデータ破壊などの事故を引き起こす。よって、(P1)は絶対に起こらないよう改善を施す必要がある。これに対し、(P2)と(P3)は正しく動作するので、問題としては軽微である。しかしまだ性能を向上させる余地が残されている。

3．性能改善の方法と手順

3-1. 改善手法

本稿で扱う改善は、以下の6種類（ただしセルの移動は2種類あり厳密には7種類）である。ここで挙げた改善は、アーキテクチャ設計の現場で経験的に行なわれているものであり、必ずしも網羅的ではない。また、実装テクノロジによって適用不可能な改善も含まれている。実際の改善に当たっては、個々の改善の適用可能性を検討する必要がある。

(S1) セルの移動

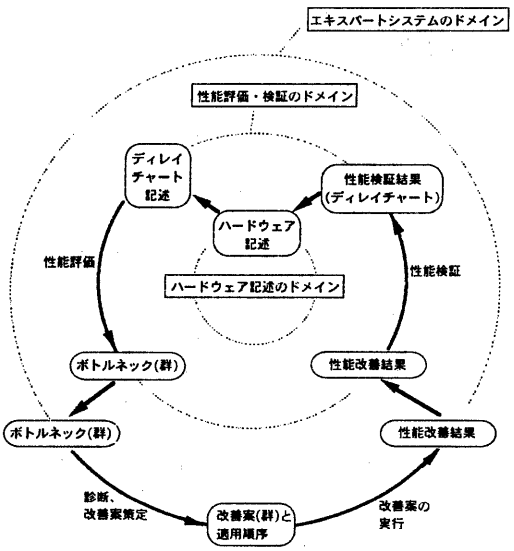

図1 計算機ハードウェアの性能改善サイクル

(S1A) パス改善方式

方法：問題のステージ中の最大遅延時間のパスに着目し、そのパス中のセルを他のステージに移動する。場合によってはステージラッチを追加する。

制約：移動前と移動後で同一の機能となるよう、セル間の接続関係を維持しながら移動を行なう（サブパスによって保証：後述）。また、移動に伴って他のパスが性能上問題にならないことを保証しなければならない。

(S1B) 全ステージ改善方式

方法：全ステージに共通する遅延時間の目標値を設定することによって、全ステージの全パスの遅延時間がその目標値以内に収まるようにセルを移動する。

制約：A方式と同様に移動前と移動後で同一の機能となるよう、セル間の接続関係を維持しながら移動を行なう。しかし移動に伴って他のパスが性能上問題になることを無視することができる場合がある。

(S2) ステージの分割

方法：問題のステージを二つのステージに分割する。あるステージで問題となるパスの数が多い場合には有効である。

制約：パイプライン構造が変化するので、分岐時などのステージ抑止の条件が変化する。本システムでチェックする範囲を超えた場合にはアーキテクチャの設計者に変更の可否を問い合わせる。

(S3) 複数ステージの統合

方法：隣接する2ステージにおける全てのパスの遅延時間がクロック時間に対して1/2程度である場合、当該の2ステージを1ステージに統合できる。

制約：統合した結果、パスの遅延時間がクロック時間を超過する場合もありえる。これは再度(S1)などの改善を施す必要がある。また、(S2)と同様パイプライン構造が変化するので、ステージ抑止条件をチェックする必要がある。

(S4) ステージのクロックタイミング調整

方法：あるステージでパス遅延時間がクロック時間をオーバし、かつ後続のステージで全てのパスの遅延時間がクロック時間に対して余裕を持っている場合、当該2ステージを隔てるラッチのクロックタイミングを遅らせる。これにより、前のステージではクロック時間が伸びた形になるので、遅延による動作不良は防げる。

制約：ASICでは、このような微調整の機能は持たないものが多い。適用可能か否かは、設計のプラットフォームに依存する。

(S5) クロック周波数アップ

方法：全てのステージにおいて、全てのパス遅延時間がクロック時間に対して余裕がある場合は、クロック周波数を上げる（クロック時間を短縮する）ことができる。この改善は論理変更を伴わないので、論理設計者にとっては性能向上のためには最も労力の少ない改善手法である。

制約：特になし。

(S6) クロック周波数ダウン

方法：多くのステージにおいて、パス遅延時間がクロック時間を超過している場合には、クロック周波数を下げる（クロック時間を伸長する）ことで対策できる。

制約：この改善は、常に性能向上を求められている現在の計算機ハードウェアのマーケットでは若干受け入れられない面を持つ。この方法以外適用可能なものがない場合、致し方なく適用する方法であろう。

これらの改善の中で、(S1)～(S3) は論理変更によるもの、(S4)～(S6) はクロック操作によるものである。2節で述べた問題と、ここで列挙した改善は表1のように対応する。

3-2. 改善方法の具体的な手順

前項で挙げた改善手法の中でその手順が特に複雑な(S1)、(S2)、(S4)に関して具体的に手順を述べる。

(S1)セルの移動

(S1A) パス改善方式

セルの移動A方式は問題のステージ中最も遅延時間が長いパス（このパスをPとする。）を基準として行う。まずP中のセルを問題のステージの前方ステージ、後方ステージのどちら側のステージに移動させるか決定する。一度になるべく多くのセルを移動させることによって改善をおこないたいので方法として次のように問題のステージの両隣のステージのパスを比較する。

表1　性能問題と対策の対応

	S1A	S1B	S2	S3	S4	S5	S6
P1	○	○	○	−	○	−	○
P2	○	○	○	−	−	−	−
P3	−	−	−	○	−	○	−

パスPにつながる両ステージのパスを検出し、前方側のステージ中最も遅延時間が長いパスを選び出す。同様に後方ステージ側から最も遅延時間が長いパスを選び出す。選び出された2つのパスを比較し遅延時間の短いパス（このパスをQとする。）が存在するステージにセルの移動を行う。

次にP中の移動させたいセルを決定する。セルの移動先のステージと移動元のステージの遅延時間のバランスを保つためにつぎのような式を考える。

$r = p - (p + q) \div 2$

(p：Pの遅延時間、q：Qの遅延時間、r：P中の移動させたいセルの遅延時間の合計)

全てのセルについてセルを通過する遅延時間が既知であるので、上式のrよりP中の移動させたいセルの数nが決定する。

次に移動後の論理の等価性を考えるために図2を示す。決定したn個のP中のセルのみを移動させると、C1、C2、C3やCSu、CSv、CSwといったセルによって構成されているパスがとぎれてしまう。これを防ぐためにP中の移動させたいn個のセルそれぞれから移動先ステージに向かってラッチを終点とするP以外の経路を探索する。この経路をサブパスと呼ぶ（図2の例としてLB3-CS1-CS2-C3C3やLB2-C1)。P中の移動させたいn個のセルとサブパス中に含まれるセルを移動させることによって、セル間の接続関係保ちつつ改善がおこなえる。

最後にセルの移動によって他パスへの悪影響が及ばないことを図3のアルゴリズムによって示す。図3の式(f)は、セル移動によってセルの移動先に新たにできるパスがセル移動後のパスPより必ず小さくなることを保証している。これによって移動先のステージの新たにできるパスは性能上の問題にならない

(S1B) 全ステージ改善方式

セルの移動B方式では3-1で述べたように全

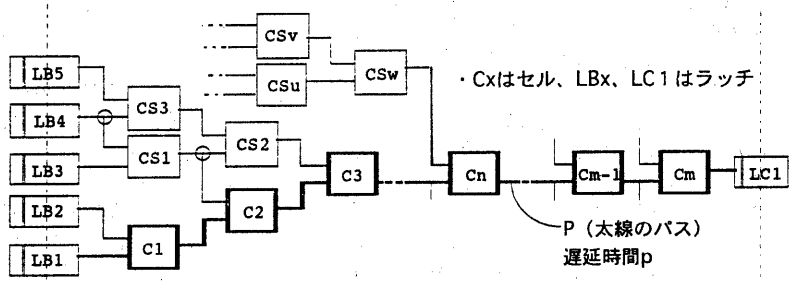

図2　サブパスの例

```
for( k=n; k=1; k--)      /*  k：ﾊﾟｽPのk番目のｾﾙ  */
{
   for( j;;j++)          /*  j：ｾﾙkのｻﾌﾞﾊﾟｽ#j  */
   {
      Sj = ｻﾌﾞﾊﾟｽjの遅延時間

      T = max(Ti)       /*  Tiはｾﾙkのｻﾌﾞﾊﾟｽ#jに繋がる移動ｽﾃｰｼﾞ先の  */
                        /*                  各ﾊﾟｽの遅延時間  */
                        /*  Tは Ti中で最大の遅延時間          */

      for( x=k; x=n; m++)  /* xは実際に移動させようとするP中のセルの個数  */
      {
         if ( r-x< ((x-k)+1) + Sj + T )          ―――― (f)
              (1)                  (2)
         {
            A(k,j)=x-1
            break;
         }
      }
   }
}
A=min(A(k,j))  /* Aは実際に移動させるP中のセルの個数  */
```

PのA番目までのｾﾙ及びA番目までのｾﾙのｻﾌﾞﾊﾟｽ上のｾﾙを移動させる
(1)はセルx個移動後のパスPの遅延時間
(2)はセルx個移動後のセル移動先ステージにできる新たなパスの遅延時間

図3　セルの移動による他パスへの悪影響阻止の為のアルゴリズム

ステージ
に共通する遅延時間の目標値を設定する。目標値の設定には次の３つのがある。
(1)パイプライン計算機が正常に動作するクロック時間（性能上の問題(P1)の解消）
(2)設計者の要求するクロック時間
(3)改善を行なうパイプライン計算機の限界クロック時間
(1)の目標値で改善不可能な場合、この改善手法は問題(P1)に対し適用できない。(2)、(3)の場合、目標値が改善可能な値であるかどうかはこの改善手法が全て終了しないと分からないため、改善可能な目標値を予想し検討することが必要となる。目標値の予想を誤ると改善不可能、または更に改善の余地が残され（(3)では改善失敗）、目標値再設定の必要が生じ効率が下がる。目標値予想の手段として、全パスの平均を取る、全体を通しての最長パスをステージ数で割る、全パスの最頻値をとる事により得られると考える。

目標値が決定したら実際にセルの移動を行なう。手順としてある１つのステージに着目し、そのステージ中の全てのパスの遅延時間が目標値以下になるまでセルの移動を繰り返す。１つのステージでの処理が完了すると隣接する次のステージの処理に移行する。セルの移動先は原則として次に処理を行なうステージに移動させる。開始するステージはセル移動先ステージが絞られるため、両端いずれかに最も近いステージから改善を行い、１つずつ内側のステージに処理対象を移し、反対側の端に到達した時点で目標値オーバパスが無くなっていれば改善終了である(図４)。反対側の端に到達した時点で目標値オーバパスが存在する場合は目標値を変更し再度初期状態から始める。

(S2)ステージの分割

問題のステージを２つに分割するということは言い替えると、問題のステージ中の全てのパスに１つのみのステージ分割ラッチを設けるということである。このとき全ての分割するステージ中のパスが均等に分割されかつあるパス中に２つ以上のステージ分割ラッチを設けないために次のような操作を行う。

ステージの負荷を均等に分けるため、問題のステージ中の最大遅延時間のパスを選びだしそのパスの中間地点にあるセルを基準セルとする。そのセルの出力直後に分割ラッチを設ける。次にその基準セルの前ステージ方向へ辿った全ての経路中のセル（前サブパスセルとよぶ）および後ろステージ方向へ辿った全ての経路中のセル（後サブパスセルとよぶ）を検出し列挙する。そして基準セルを含むパスを問題のステージのガントチャートから削除する。

次にまた問題のステージ（前段階までの基準セルを含むパスは除外されている）中から最大遅延時間のパスを選び出し、このパスの中間地点にあるセルを新たな基準セルとするのであるが、このとき中間地点にあるセルが前サブパスセルになる場合にはそれよりも後ステージ側にあるセルを基準セルとし、後サブパスセルになる場合にはそれよりも前ステージ側にあるセルを基準セルとする。この操作によって全てのパス中にステージ分割ラッチが１つのみ設けられることを保証している。以下は先ほどと同様の操作を行なう。以上のような操作を繰り返しガントチャートから全てのパスが削除されると終了となる。

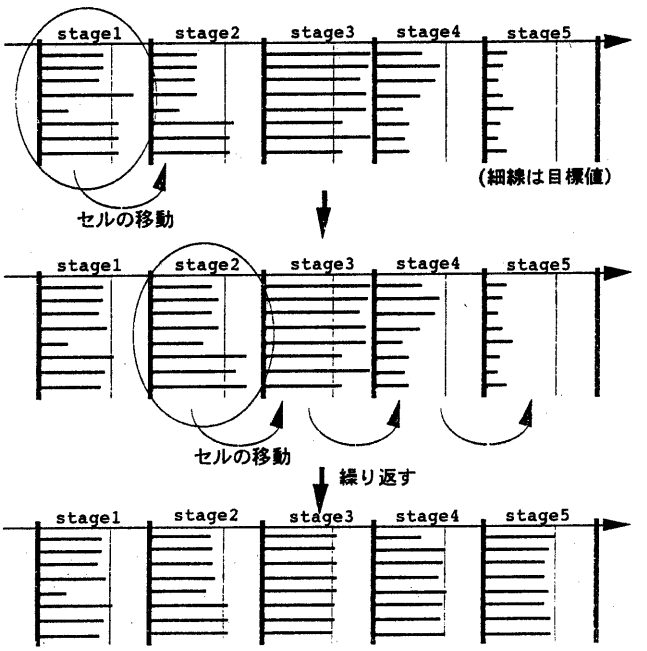

図４　セルの移動(全ステージ改善方式)

4．改善例

性能改善の検証を行なうため、Alliance[6]添付のビットスライスプロセッサAMD2901をパイプライン化し(約600セルで構成)、この性能改善を試みた。以下では説明のため、図5のパイプラインを取り上げ、セル移動による改善を行なう。

改善の目標は、n個のセルを移動させることによってボトルネックバスPを改善することである。ここで、C1,C2,C3,…,C9およびCS1,CS2,CS3はそれぞれ1ゲートで構成されたセルであり、p=9,q=3より

r=9-(9+3)/2=3

となり上記の条件よりn=3となる。また、サブバスに繋がる前ステージの最多ゲート数通過バスのゲート数Tはあらかじめわかっているとする。

改善を行なうバスPが最前ステージもしくは最後ステージある場合、セルを移動させるステージは一意に決まるが、そうでない場合はセルの移動先ステージを決める必要がある。

ここでPに繋がる前後ステージのバスのゲート通過数を比較しているのは、ボトルネックバスであるPのディレイをなるべく短くしたいのでセルをなるべく多く移動できるようなステージを選ぶためである。

前ステージにセルを移動させる場合はラッチからの入力を直接前ステージから受け取るために、後ステージにセルを移動させる場合はラッチへの出力を直接後ステージに送るために行なうか、あるいは、新しくラッチを設けることによってセルを完全にもとのステージから移動先のステージに移すことができる。

図5ではセルC1、C2、C3を移動させたいが、C2はCS1の入力がありC3はCS2の入力があるのでC1、C2、C3のみを移動させると信号の流れを変えてしまう。

これを防ぐためにPの移動させたいセル（例ではC1、C2、C3）から移動先ステージまでのP以外の経路、すなわちサブバスを検索する。

この例でサブバスは、LB2-C1, LB3-CS1-C1, LB4-CS1-C2, LB3-CS1-CS2-C3, LB4-CS1-CS2-C3, LB5-CS3-CS2-C3 の6つのバスである。信号の流れを変えないためにサブバスに現われたセル（CS1,CS2,CS3)も移動させたい。

次に、上記のことからP中のセルC1,C2,C3とサブバス中のセルCS1,CS2,CS3を移動させるとする(図6)。すなわち、ラッチLB5に接続している移動先ステージのあるバスをRとし、このバスRは6ゲート通過する。

CS2,CS3のセルを移動させたならば図7のようなラッチLA5からセルC3までを通過する1つのバスができる。この新たにできるバスは9ゲート通過するが、これはPが通過するゲート数であるのでこ

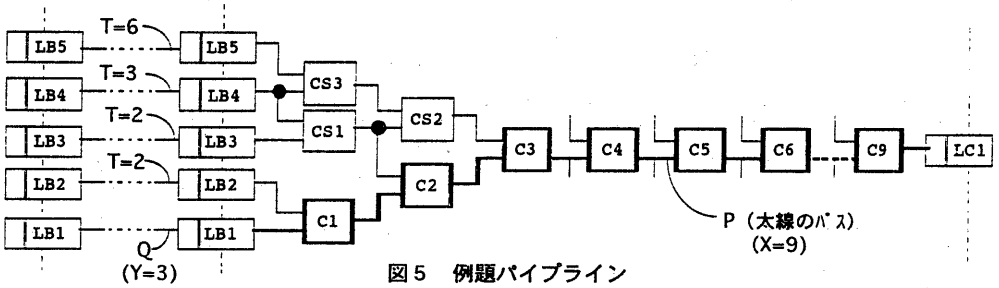

図5　例題パイプライン

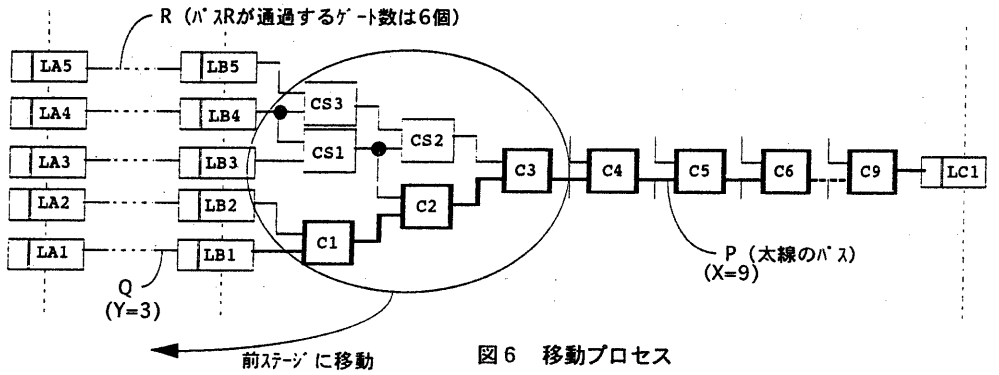

図6　移動プロセス

のパスはボトルネックパスとなる。

このようにセルの移動によって新しいボトルネックパスを生じさせてしまう恐れがある。これを防ぐために図３のアルゴリズムを考える。これに従って移動するセルを決定した場合、前述した移動先のステージのパスと改善を行なうパスのゲート段数のバランスを保てない可能性がある。しかし、どのようなボトルネックパスであっても前述の問題点を回避することはできる。

図３のアルゴリズムに従って図６の移動パスを決定するプロセスを表２に示す。

５．おわりに

パイプライン計算機を対象とした性能改善手法PIPEDIETSを提案した。アーキテクチャ設計者はPIPEDIETを対話的に利用しながら、少ない労力で性能設計や性能改善を行なうことができる。

今後、高い並列性を求められているVLIWなどのアーキテクチャに対してPIPEDIETSの手法を適用することを検討していきたい。

参考文献

(1)田村恭久、伊藤　潔、杵嶋修三、ドメイン分析・モデリング技術の現状と課題、情報処理　Vol.35, No.10, pp.952-961 (Oct.1994).

(2)Xing-Jian Xu, M.Ishizuka, An Algorithm for Designing Pipeline Processing Circuit, Proc. IPSJ DA Symposium '92, pp.1-4, August 1992.

(3)Hennessy,J.L., Patterson,D.A., Computer Architecture : A Quantitative Approach, Morgan Kaufmann 1990.

(4)Armstrong,J.R., Gray,F.G., Structured Logic Design with VHDL, Prentice Hall 1993.

(5) Navabi,Z., VHDL : Analysis and Modeling of Digital Systems, McGrawHill 1993.

(6)Greiner,A., Alliance : A Complete Set of CAD Tools for teaching VLSI Design, included in Alliance Software Set.

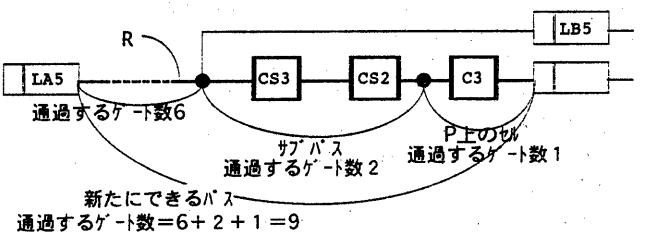

図７　ｾﾙの移動による新しいﾊﾟｽ

表２　図１２の例題に対するセル移動のプロセス

k	j	ｻﾌﾞﾊﾟｽ	Sj	T	x	条件式(f) (p=9)	説明
k=3	j=1	LB3 - CS1- CS2- C3	S1=2	T=2	x=3	9-3=6＜((3-3)+1)+2+2)=5	条件式が成り立たないのでj= 2 となり次のｻﾌﾞﾊﾟｽを検索
	j=2	LB4 - CS1- CS2- C3	S2=2	T=3	x=3	9-3=6＜((3-3)+1)+2+3)=6	条件式が成り立たないのでj=3 となり次のｻﾌﾞﾊﾟｽを検索
	j=3	LB5 - CS3- CS2- C3	S2=3	T=6	x=3	9-3=6＜((3-3)+1)+2+6)=9	条件式が成り立つので A(3,3) = 3-1 =2 となりﾙｰﾌﾟの外に出る。C3のサブパスは３つなのでk= 2
k=2	j=1	LB3 - CS1- C2	S1=1	T=2	x=2	9-2=7＜((2-2)+1)+1+2)=4	条件式が成り立たないのでx=3
					x=3	9-2=7＜((2-2)+1)+1+2)=4	条件式が成り立たないのでj= 2
	j=2	LB4 - CS1- C2	S2=1	T=3	x=2	9-2=7＜((2-2)+1)+1+3)=5	条件式が成り立たないのでx=3
					x=3	9-3=6＜((3-2)+1)+1+3)=6	条件式が成り立たないのでj=3 となり次のｻﾌﾞﾊﾟｽを検索。 しかしｾﾙC2のｻﾌﾞﾊﾟｽは2つしかないのでk=1
k=1	j=1	LB2 - C1	S1=0	T=2	x=1	9-1=8＜((1-1)+1)+0+2)=3	条件式が成り立たないのでx= 2
					x=2	9-2=7＜((2-1)+1)+0+2)=4	条件式が成り立たないのでx=3
					x=3	9-3=6＜((3-1)+1)+0+2)=5	条件式が成り立たないのでj=2 となり次のｻﾌﾞﾊﾟｽを検索。 しかしｾﾙC1のｻﾌﾞﾊﾟｽは１つしかないので終了