2F−1

並列コンパイル方式

間藤隆一、　荒木均、　加藤等、　野島晋二
松下電器産業株式会社　情報通信東京研究所

1 はじめに

近年、マルチメディア処理をはじめとして、大量の計算を高速に処理する必要性が高まっている。本報告では、特定用途として使用し、均一ではない並列性を含む数値計算の命令レベル並列処理を目的としたマルチプロセッサ・システムのアーキテクチャを提案し、シミュレーティッド・アニーリング[Kirkpatrick 83]を利用したコンパイル方式について報告する。

2 処理対象

我々は、pascal言語からのLSI設計つまりHLS(high level synthesis)の研究を行なっている[舘野 91]。従来のHLSでは、1つのLSIを設計の対象としており、格納できる機能ユニットの数の制約から、並列処理性能に限界がある。そこで、複数のLSIを協調的に動作させるような論理合成技術が必要であるが、複数のLSIを製作することは、コスト高と設計時間の増加を招いてしまう。そのため、あらかじめ機能ユニットの異なるプロセッサ群を用意し、応用分野に応じてそれを適当に組合せ、徹底したプログラムの最適化を行なうことによって、複数のLSIを新たに製作する場合に近い性能を発揮することを目的としている。そのため、本プロセッサは、以下にあげる処理を対象とする。

- 並列性の高い数値計算処理
- 特定の用途向けの処理またはコプロセッサとして実現したい処理
- プログラムサイズが小さい処理

3 命令レベル並列処理

上記の目的にあったものとして、汎用CPUの命令レベル並列処理のアーキテクチャ[村上 91]があり、様々な方式が提案されている。本システムでは、特に、均一でない並列性を含む数値計算を対象とした並列処理および機能ユニットを自由に組み込める点からVLIWアーキテクチャを採用し、プロセッサ間の同期に、データ駆動方式を採用した。

4 データ駆動型VLIW

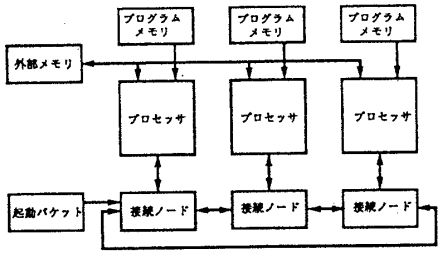

図1　全体構成

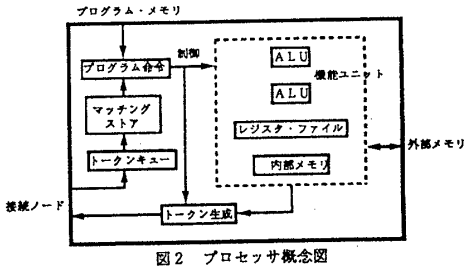

図2　プロセッサ概念図

図1がデータ駆動型VLIWの全体構成図である。複数のプロセッサが密結合し、全体として単一VLIW方式と同等の動作を行なう。各プロセッサ間の通信は、接続ノードを経由したパケットにより実施される。接続ノードは、パケットに含まれるプロセッサ番号に従って、自分のプロセッサ番号ならば自分のプロセッサに転送し、他のプロセッサ番号であれば、次の接続ノードに転送される。プロセッサと接続ノード間および接続ノード間のデータは、並列データとして高速に転送される。図2がプロセッサの構成図である。接続ノードから入力したパケットから余分な情報を除いたトークンが生成され、トークン・キューに転送される。マッチングストアは、タグを識別子として、対応するトークンを見つける。そのトークンが持つ命令をプログラムメモリからロードし、その命令を実行する。命令は、水平型マイクロ命令であり、複数の機能ユニットを並列に実行することができる。命令は、通常、複数のマイクロ命令から構成されており、他のプロセッサと関係のない処理を実行し、再び待ち状態に入る。他のプロセッサへのパケットは、内部のデータとマイクロ命令が合成し、接続ノードを介して転送される。

Scheduling method for multiprocessor
Ryuichi Mato, Hitoshi Araki, Hitoshi Kato, Shinji Nojima
Matsushita Electric Industrial Co., LTD.

5 コンパイル方式

このように並列度が高いプロセッサでは、プログラムをどのようにスケジューリングするかが重要な問題となる。現在、機能ユニットが同じで、制御命令を含まない数値計算式のプログラムをコンパイルの対象としている。入力された命令のハイトリダクションを行い、2項演算式に展開する。図3は、演算式を適当に割りつけた初期のスケジュール表を示している。縦方向が時間ステップ、横方向が各ALUが行なう演算を示している。2本の縦線が各プロセッサの境界を示し、1本の縦線がALUの境界を示している。問題は、スケジュール表の時間ステップ（コスト）を最小にするスケジュールを求めることである。コストは、以下に示す事項に基づいて計算される。

データ転送時間 とは、あるプロセッサ内に必要とするデータがないとき、他のプロセッサから転送する時間である。

データ衝突時間 とは、接続ノードにおいて、データ転送が双方向からきたとき、データを退避する時間である。

レジスタ内データ格納 とは、データが、レジスタファイル内で収められるとき、コストが小さくなる。

外部メモリ・アクセス競合 とは、外部メモリに対して、各プロセッサが同時にアクセスするとコストが高くなる。

アニーリングのアルゴリズムを以下に示す。

```
T = 初期温度
X = 初期スケジュール表
while(コスト変化がある。) {
        for(一定回数) {
        J = ある演算式をランダムに他の場所に
            移動したスケジュール表
        if(accept(コスト(J), コスト(X),T)) {
        X = J;
        }
    }
    T = T * 0.9
}
```

```
accept(コスト(J), コスト(I),T) {
    コストの変化 = コスト(I) − コスト(J);
    if(コストの変化 < 0) return(真);
    else {
        Y = exp(− コストの変化 /T);
        R = random(0,1);
        if(R < Y) return(真);
        else return(偽);
    }
}
```

6 実行環境

コンパイラは、Solbourn上のLISPでインプリメントされている。入力となるプログラムは、自動的にランダムな命令を生成するプログラムで生成し、コンパイルし、実験を行なっている。

7 まとめ

マルチプロセッサ構成のVLIW方式により、並列の粒度は粗くなるが、VLIW方式の特徴である命令レベル並列度を高くできるアーキテクチャとそのアーキテクチャ上で効果的なコンパイル方式を提案した。今後は、制御命令の追加と異なる機能ユニットを含むCPU上で最適化可能なコンパイラを製作する予定である。

参考文献

[Kirkpatrick 83] S.Kirkpatrick, C.D.Gelatt, M.P.Vecc *"Optimizaton by Simulated Annealing"* in *Science*, 13 May 1983, Vol.220, No.4598, pp671-680

[舘野 91] 舘野他 *"Rule-based Annealing"* in 情処研報, 12 Sep. 1991, 91-AI-78

[村上 91] 村上和彰 "ハイパースカラ・プロセッサ・アーキテクチャ" in 情処研報, 11 March 1991, 91-ARC-87

processor1		processor2		processor3	
alu1	alu2	alu1	alu2	alu1	alu2
Thbp=Iysi-Iysi					
Thiq=Thbp-Thbp					
Tgck=Thbp-Thiq					
Thcn=Thiq-Tgck	Tmbi=Thbp-Tgck	Tlvf=Tgck*Tgck			
Twde=Thcn*Thiq	Towy=Thcn/Icjq	Taxp=Thbp-Tmbi	Thqo=Tgck*Thcn		
Tffq=Twde+Tmbi	Tice=Twde/Tmbi	Tvke=Twde/Thcn	Teja=Towy/Tlvf	Tfgw=Towy+Tlvf	Tuci=Towy-Taxp
Tdan=Tffq-Tice	Twdf=Tffq*Tffq	Tkzb=Teja+Thbp	Txjh=Twde/Tffq	Tkcc=Tice*Tvke	Ttfm=Tgck/Tffq
Tzxn=Tice+Twdf	Tmsx=Ifgp-Txjh	Tqln=Twdf/Towy	Tdnc=Txjh+Tice	Tipv=Ttfm/Tice	Tfty=Tlvf/Tkcc
Tnrk=Tzxn*Taxp	Tocj=Tdan-Tzxn	Tynx=Thiq-Tzxn	Tnyh=Tgck/Tzxn	Tenl=Tmsx-Tvke	Tviv=Tice/Tkcc
Osof=Tnrk/Txjh	Twgi=Tdan-Tnrk	Tbcl=Tvke-Tnrk	Tmwb=Tfgw+Tnrk	Ttll=Txjh+Twde	Trmu=Tfty+Tuci
Tlyv=Taxp-Twgi	Tnwj=Tocj*Tuci	Turz=Taxp+Tnyh	Tzug=Tkzb+Tviv	Touh=Tfgw*Tdan	Okjy=Tzxn*Tzxn
Ohbx=Tlyv+Tfty	Ozhk=Tnrk-Tlyv	Onic=Thcn*Tnwj		Oneu=Thqo+Tffq	

図3: スケジュール表