2T-3 データ駆動形プロセッサにおける相対アドレス方式の検討

坪田浩乃、小守伸史、高田英裕、田村俊之、浅井文康、徳田健　(三菱電機㈱ＬＳＩ研究所)
山崎哲男、嶋憲司　(三菱電機㈱産業システム研究所)
西川博昭、寺田浩詔　(大阪大学工学部)

1．はじめに

データ駆動形プロセッサを実用化する上において、プログラム記憶のハードウェア規模の大きいことが一つの障害となっている。

本報告では、プログラム記憶中の行き先ノード番号を相対アドレスで表現することにより、プログラム記憶サイズが大幅に圧縮可能であり、また、容易にマルチタスク環境が実現できることを示す。さらに、この方式を採用することによる、性能に対する影響についても評価を行った。

2．プログラム記憶サイズの圧縮

データ駆動形プロセッサのプログラムは、命令ノード群と、これらの間におけるデータ依存関係を示すアークからなる有向グラフとして定義される。物理メモリ上では、図1に示す通り、命令ノードの情報として制御フラグおよび命令コード、また、アークの情報として行き先ノード番号が格納されている。

例えば、制御フラグが8ビット、命令コードが8ビット、行き先ノード番号を格納する領域が32ビットと仮定すると、1命令当りのビット幅は48ビットとなり、ノイマン形のプロセッサと比較した場合の短所となっている。

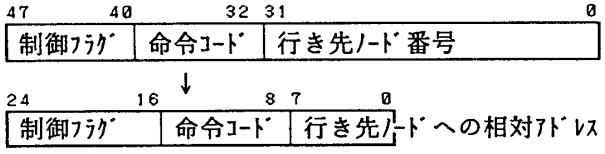

図1　プログラム記憶の情報

そこで、プログラム記憶のサイズを圧縮するために、行き先ノード番号を、現在の番地と行き先ノードの格納番地とのオフセット値、即ち相対アドレスで表現する方式を提案する。本方式は、命令ノード間の接続の大半は局所的であり、行き先ノードへの相対アドレスが大きな値をとることは稀である、という性質に基づいている。

A Study on Relative Address Architechture for Data-Driven Processors.
H. Tsubota, S. Komori, H. Takata, T. Tamura, F. Asai, T. Tokuda (Mitsubishi Elec. Corp. LSI R&D Labo.)
T. Yamasaki, K. Shima (Mitsubishi Elec. Corp. Industrial Electronics & Systems R&D Labo.)
H. Nishikawa, H. Terada (Osaka University)

3．マルチタスクの実行

従来のデータ駆動形プロセッサでは、プログラムのロードモジュールが生成された時点でアドレスの絶対値が決定されていた。このため、複数のプログラムを同時に実行するためには、ロードモジュール生成以前に、別々のアドレス領域を指定しておく必要があった。

本方式によれば、プログラム上のすべてのノードが相対アドレスで表現されるので、プログラムはプログラム記憶の任意のアドレスにロードして実行することができる。すなわち、一つのタスクを実行中に、プログラム記憶の空き領域に、他のタスクを動的にロードし、マルチタスクの同時並行処理を行うことが可能であり、特別なハードウェアを付加することなく、マルチタスク環境を容易に実現することができる。

4．相対アドレス方式を用いた実行制御

本方式を実現するための、巡回パイプライン構成の一般的なデータ駆動形プロセッサの概略図を図2に示す。

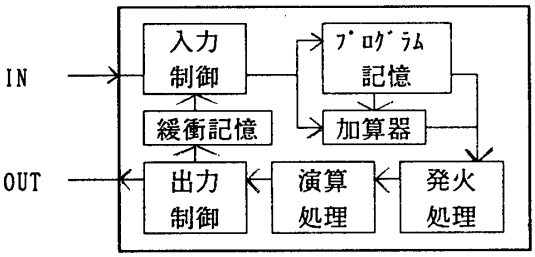

図2　データ駆動形プロセッサ

外部入力パケットあるいは、演算結果パケットをプログラム記憶部が受け取ると、パケット中の（絶対）行き先ノード番号をアドレスとしてメモリから次に実行すべき命令コードと、相対アドレスが読み出され、加算器によって行き先番地の加算が行われ、次の（絶対）行き先ノード番号が得られる。発火処理部では、（絶対）行き先ノード番号をＫＥＹとして、演算相手データとの待ち合わせを行う。待ち合わせの完了したパケットは、即座に演算実行される。

プログラム記憶には、図1下部に示した構成の命令ノード群が格納されている。但し、行き先ノードへの相対アドレスが8ビットで表現できない場合、すなわち、$-128\ (=-2^7)$ ～$127\ (=2^7-1)$ の範囲を越える場合は、溢れた桁は拡張アドレスとして次のアド

レスに格納し、また、制御フラグに1ビットを追加し、拡張フラグに当てるものとする。

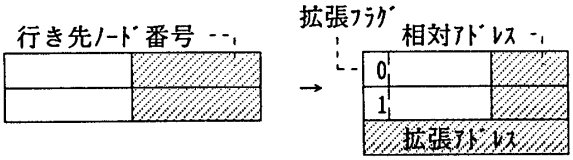

図3　拡張アドレスの発生

5．ノードアドレスの再割り当て

図3に示したような拡張アドレスの発生に伴って、言語処理系の中で、プログラム記憶内における各ノードのアドレスの付け換えを行う必要がある。アドレス再割り当てのアルゴリズムを図4に示す。

図中の配列NODE[]の各要素は、各ノードに対応しており、初期状態では、拡張アドレス発生以前のアドレスが格納されている。最終的に各ノード情報がプログラム記憶内にロードされるときのアドレスが格納される。また、Ext_cntには、新たに発生した拡張アドレス数が格納される。

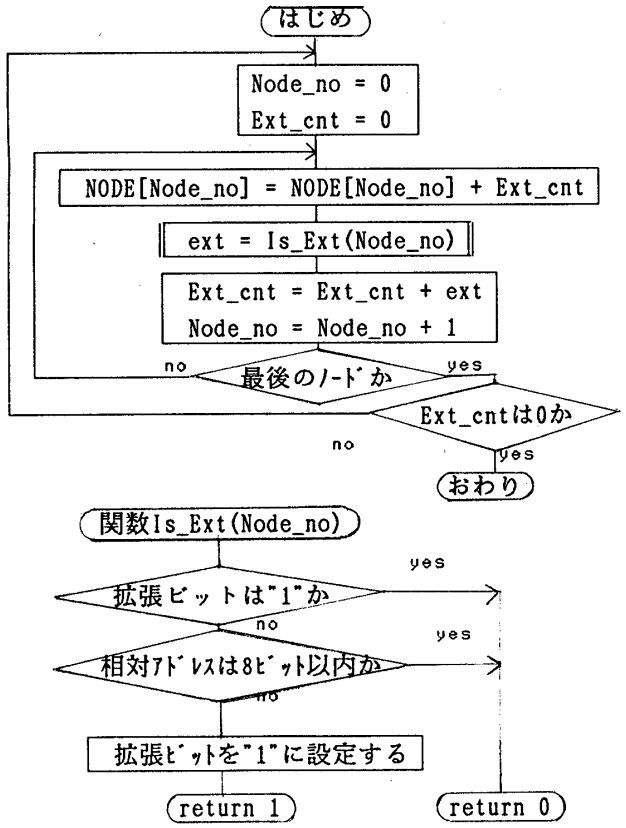

図4　ノードアドレス再割り当てのアルゴリズム

このアルゴリズムは、全ノードをチェックする2重ループになっている。外側のループは、拡張アドレスの挿入に起因する新たな拡張アドレスの発生に対応するために設けられており、ノード接続の局所性を仮定すれば、全ノード数にかかわらず、ほぼ一定の繰り返し回数になる。したがって、処理時間は近似的に全ノード数 " n " のオーダーに抑えることができる。

6．評価結果

拡張アドレスが発生した場合には、1パケットに対するプログラム記憶アクセス回数が2回となり、プログラム記憶からの出力が1回分休止する。このため、パイプライン中に1サイクル分の空きが生じ、パイプライン処理の実効的な効率が低下する。低下率は、静的な見積りとしては、拡張アドレスの発生頻度に比例すると考えられる。

そこで、拡張アドレスの発生頻度について実験を行った。今回は、特に相対アドレスのための最適化は考慮せず、入力に近いノードから順にノード番号を与えるようにノード番号を割り付けられたオブジェクトプログラムを対象として、拡張アドレス領域を必要とするノードが何パーセント存在するかを調べた。

評価に用いたプログラムは以下の3種類であり、各々約1000ノードのプログラムである。

①5つのデータ入力をもつ5並列のプログラムであり、アークは互いにクロスしながら2入力の演算を行わせるもの。
②上記①のプログラムをループさせたもの。
③上記①のプログラムを2重にループさせたもの。

表1　評価結果

	拡張が必要なﾉｰﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ記憶の圧縮率
①	0 %	52 %
②	2 %	53 %
③	4 %	54 %

以上の結果から、単純な構造のプログラムでは、プログラム記憶サイズは60%以下に、また、性能低下は数%以内に抑えられることがわかった。

7．おわりに

データ駆動形プロセッサへの相対アドレス方式の導入の検討を行った。この結果、この方式によりデータ駆動形プロセッサ上のプログラム記憶サイズを現在の60%以下に減らすことが可能であり、一方、このとき性能に与える影響は数%以下であることがわかった。

また、本方式により、マルチタスク環境が容易に実現できることを示した。今回は、試みにきわめて単純な構造のプログラムについて評価を実施した。今後、さらに複雑な構造のプログラム、また、大規模な応用プログラムについても評価を行い、本方式の有効性を検証していく予定である。

謝辞

いつも有益な議論をいただくシャープ㈱の宮田宗一博士、岡本俊弥氏、内藤裕幹氏をはじめとする諸氏に厚く感謝します。