2007－DPS－133 (8)
2007／11／26

組込みタイルプロセッサ-TEMP-の設計とマルチメディア演算の実装

田路 真也† 森 秀樹† 上原 稔†

†東洋大学大学院工学研究科情報システム専攻

概要

組込みＣＰＵのための新たなマルチメディア処理用サブプロセッサを提案する．アーキテクチャに近年のプロセッサの微細化に対応できるよう配線遅延を回避できるタイルプロセッサを参考にした．またタイルアーキテクチャの持つ高い並列性を持つためメディアプロセッサに最適であると考えた．本稿においてはタイルプロセッサにバタフライ演算を実装させ命令速度の高速化を図った．

Design of Tile-based Embedded Multimedia Processor -TEMP- and Implementation of Multimedia Operation

Shinya Touji† Hideki Mori† Minoru Uehara†

†Dept. of Open Information Systems,
Graduate School of Engineering, Toyo University

Abstract

We suggest an assistant processor for new multimedia processing for main processor. We referred to the tile-based processor which could evade a wiring delay to be able to support tininess of a recent processor about the architecture. In addition, We thought that We were most suitable for media processor to have the high concurrency which lasted of the tile architecture. We let a tile processor implement butterfly operation in this report and planned speedup of order speed.

1．はじめに

20 世紀後半から始まったインターネットの普及による急速なマルチメディア化はCPUに画像処理、音声処理など他の処理よりも比較的処理時間のかかる処理をさせることが多くなった．そこでこれらの処理を早くすることを目的としたマルチメディアプロセッサが必要になってきた．この流れはパソコンに限ったことではなく組み込みCPU の世界、特に携帯電話や DVD プレイヤーなどのCPUにも高速な音声処理や画像処理が求められるようになってきている．

また近年プロセッサの微細化は 1 クロックで通信できる物理的距離による配線遅延という問題も懸念される．

本稿ではこの二つの問題を解決するために配線遅延を回避できるタイルプロセッサを使いマルチメディア処理を実行するマルチメディア処理専用の組込み用サブプロセッサ TEMP を設計することを目的とする．今回は FFT などのマルチメディア処理において基本となるバタフライ演算を実行させることにした．

2．関連技術

2.1 バタフライ演算

バタフライ演算とは FFT で行なう演算でNをサンプル数、$W_N^k = \exp(-j2\pi k/N)$とすると以下の式で表せる．

$$X(k) = x(k) + W_N^k \times x(k+\frac{N}{2})$$

$$X(k+\frac{N}{2}) = x(k) - W_N^k \times x(k+\frac{N}{2})$$

バタフライ演算は入力がたすき掛けになっており図 2 に示すように図で表すと蝶の羽のような形になることからバタフライ演算と名付けられた．

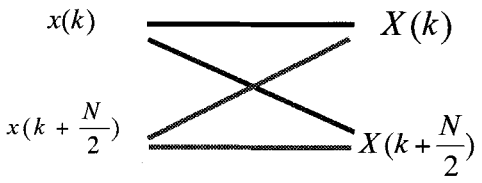

図 2　バタフライ演算のシグナルフロー

$N=8$の場合このバタフライ演算を 12 回行なうことによって FFT の結果を得ることができる．1個のバタフライ演算はW_N^kがあらかじめ用意されているとすると 1 回の複素数乗算と 2 回の複素数加減算からなっており計算を$N/2\log_2 N$減らすことができる．

2.2 タイルプロセッサ TRIPS

現在はそこまで問題にはされていないが今後、プロセッサが微細化していくと 1 クロックでチップの中を伝わる信号の距離は小さくなってくる．将来 32nm ノードぐらいまで微細化していくと 1 クロックで伝播できる信号はチップの面積に比べて 1%以下になってしまう．そこで出てくるのが配線遅延の問題である．物理的に遠い間での通信では数十クロックかかってしまうこともある．この配線遅延を回避するために考えられたのがタイルアーキテクチャである[1]．

タイルアーキテクチャでは配線遅延が問題とならない小さなサイズの演算タイルを規則的に敷き詰めて物理的に近いところに配置されたタイル間でのみデータの受け渡しを行うことでタイル間の配線遅延を回避する．

タイルプロセッサにデータフローの考え方を加えたものが現在テキサス大学で研究されているTRIPS と呼ばれるタイルプロセッサである[2]．TRIPS も演算タイル自体は整数演算器、浮動小数点演算器、インプットポート、アウトプットポート、オペランドバッファ、ルータ、64 の命令バッファからなる単純なものでこのタイルを 4×4 の 16 個、タイル状に並べその周りに命令キ

ャッシュとデータキャッシュ、レジスタを配置した形となっている．タイルは隣接するタイルと通信を行う．

TRIPS のコンパイラは各タイルに静的に命令を配置する．各タイルはデータが到着し準備ができ次第命令を実行していく．この部分がデータフローマシンのアーキテクチャに似ている．

TRIPS では命令やデータをタイルの周囲のキャッシュに配置しているため、命令やデータをフェッチするために遅延が発生する．この遅延を隠蔽するために一度に複数の命令をフェッチしている．4×4 の TRIPS プロセッサの構成図を図 1 に示す．

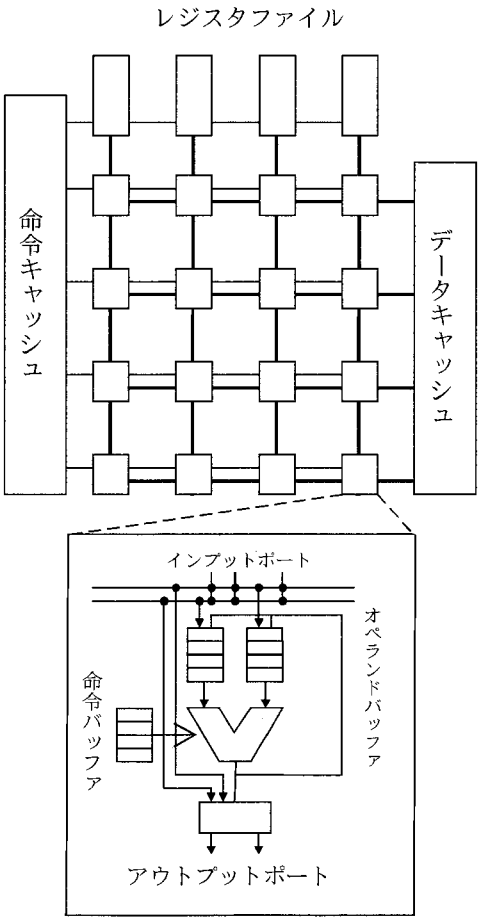

図 1　4×4TRIPS プロセッサとタイルの構成図

2.3　データフローマシン

データフロー方式は非ノイマン型と呼ばれる方式のひとつで 1974 年に Dennis が提案した並列処理を容易に記述できる計算モデルに基づいている．このモデルは関数的言語との親和性が高く関数的言語を使用すればプログラマは並列性を明示しなくても並列性が最大限に引き出されるという優れた特徴を持つ．しかし一方命令セットも異なることからノイマン型のソフトウェア資産が活用できないので現代の社会においてはなかなか普及していないのが現状であるが、スーパースカラプロセッサにおけるアウトオブオーダー処理はこのデータフローモデルから影響を受けたといわれている．

データフロー方式における計算はデータフローグラフと呼ばれるグラフで表現される[3]．データフローグラフでのノードは加算、減算などの演算を表しアークはリンクと呼ばれるデータの入出力線を示す．データフローグラフにおける計算では入力リンクにデータを表すトークンを置くことによって始まる．その後以下の規則に従って計算を実行していく．実際の計算の例を図 2 に示す．

- 入力リンクに全てのトークンがそろったらノードを発火する(演算を実行する)．
- ノードは発火したら入力リンクからトークンを取り除き、演算結果のトークンを出力リンクに置く．

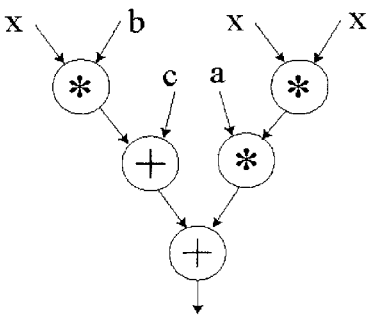

図 2　$ax^2 + bx + c$ のデータフローグラフ

TRIPS ではデータフローグラフでの各ノードが各演算タイルであるといえる.

データフローモデルにはもうひとつ規則がある.それは「1つのリンク上には1度に1個のトークンしか存在できない」というもの.これはデータフローグラフを 1 回しか使用しない場合は気にしなくてもよいが無駄ができ実用的ではない.そこでトークンにタグをつける動的計算モデルが考え出された.タグをつけたために動的計算モデルではひとつのリンク上に 1 個以上のトークンを置くことを許す.そして発火の規則を以下のように変更している.

規則1:ノードは同じタグを持つトークンが入力リンクにそろったら発火する.

規則2:ノードは発火した後にその入力トークンを取り除き、出力トークンに新たなタグをつけて出力リンクに置く.

TRIPS では各タイルのオペランドバッファには複数のデータが格納でき、必要なデータがそろったら演算を開始する.

3.プロセッサの設計

3.1 プロセッサ仕様の決定

今回作成するプロセッサ TEMP の仕様を以下のように決めた.

■TEMP は前述の TRIPS を参考に基本的にマルチメディア処理のみを行う補助タイルプロセッサとし、別のメインプロセッサとバスでつながっているものとする.命令実行時はメインプロセッサから共通のレジスタを介してデータ、命令を受けてマルチメディア処理を実行する.結果はデータキャッシュからメインプロセッサへ送られる.

■マルチメディア命令を受けた TEMP はマルチメディア命令を簡単な命令に分割した後、各タイルに命令を渡す.また計算を始めるために必要な最初のデータはこのとき共有レジスタから一番上の列の演算タイルのレジスタに格納しておく.演算タイルはこの命令を受け演算をした後、次のタイルに渡す.この作業を繰り返すことにより最終的に全体としてマルチメディア命令を実現できるようにする.

■TEMP のデザインは図3のように TRIPS を参考にして、ALU に入力と命令それぞれを入れるレジスタとルータで構成された演算タイルを 4×5 の計 20 個配置する.各タイルの接続は TRIPS と違いさらに右斜め下、左斜め下のタイルとも接続している.この斜めの接続は配線遅延の回避という点ではデメリットだがより柔軟に命令を配置することができ効率よくタイルに命令を割り当てられると考え今回は斜めの接続を追加することに決めた.

■今回の実装するバタフライ演算では複素数を扱うためデータ長は実部 8bit 虚部 8bit の 16bit とした.

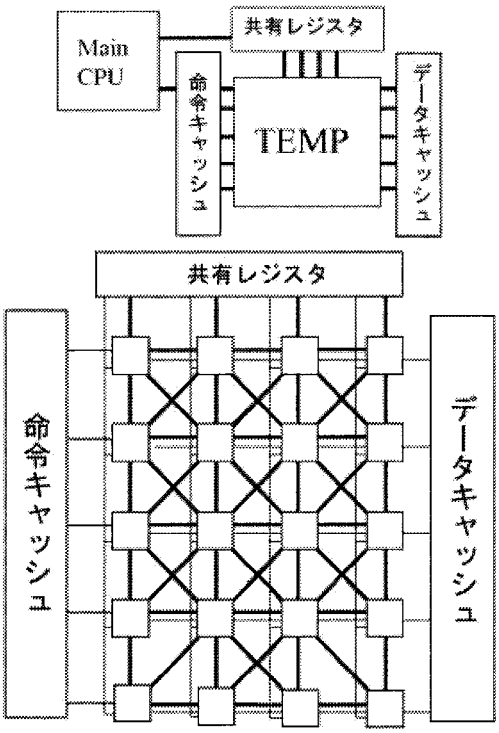

図3 TEMP のデザイン

3.2 命令の決定

バタフライ演算を実装するために演算タイル自体の命令を以下のように決定した.

- タイルの実行する命令は基本的な演算命令に加え、即値命令と複素数乗算用命令を追加する.
- 複素数乗算は実部と虚部の計算を同時に 1 クロックで行なう命令にする.
- 命令長は命令を決めるオペコード 4bit と出力先オペランド 4bit、さらに即値用に 16bit(8bit×2)用意する

今回演算タイルに実装した命令表を表 1 に示す.

表 1 演算タイル命令表

オペコード	命令名	動作
0000	THR	A を次のタイルに出力
0001	AND	A と B の論理積を出力
0010	OR	A と B の論理和を出力
0011	EOR	A と B の排他的論理和を出力
0100	NOT	A の否定を出力
0101	ADD	A と B の和を出力
0110	SUB	A と B の差を出力
0111	MUL	A の下位 8bit と B の積を出力
1000	ADDI	A と即値の和を出力
1001	SUBI	A と即値の差を出力
1010	MULI	A と即値の積を出力
1011	未使用	
1100	ST	A をデータキャッシュに格納
1101	SL	A を 1bit 左シフトして出力
1110	SRL	A を 1bit 論理右シフトして出力
1111	CPML	A と即値を複素数乗算

加減算は基本的に複素数加減算であるが実部と虚部を別々に計算するため整数の加減算を並列に行なうことも可能である.

4.タイルプロセッサのシミュレーションとマルチメディア命令の実装

前章で決めた仕様に基づいて演算タイル-図 4-を Verilog にて設計し論理合成、シミュレーションを行なった.この演算タイルは実部と虚部の計算のために 8bitALU を 2 つ持っている.

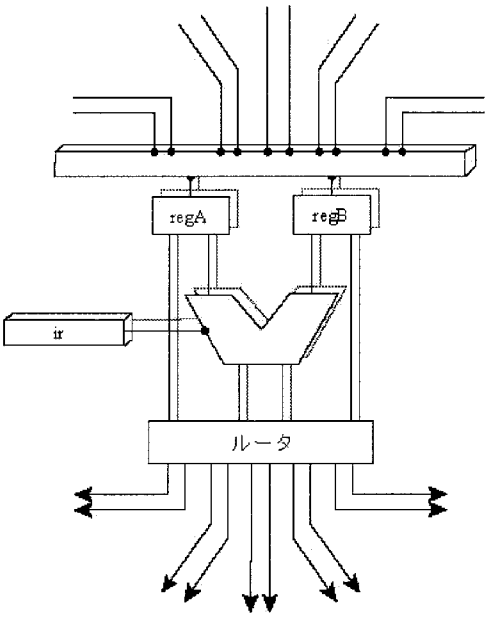

図 4 演算タイル

この演算タイルを 20 個格子状に並べたタイルプロセッサを設計し、バタフライ命令を実装させた.このタイルプロセッサは単純に 20 個格子状に並べ送った命令に応じて各タイルに命令を割り当てる疑似タイルプロセッサとなっている.

各タイルにバタフライ演算実装のための命令の割り当て方を図 5 に示す.

今回実装したバタフライ演算命令は 2 個のバタフライ演算を平行に実行し、その出力を元にもう一度バタフライ演算を実行する.つまり最終的に一回の割り当てで計 4 回のバタフライ演算を実行することができる.

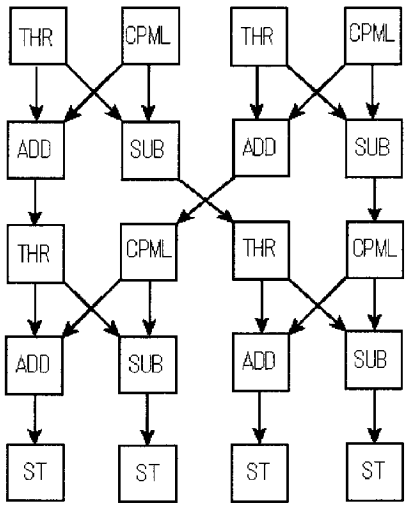

図5　演算タイルへの命令割り当て

今回作成したTEMPとシンプルな命令だけを持つプロセッサのバタフライ演算を 4 回行なう際のステップ数を表2に示す.

表2バタフライ演算のステップ数

	TEMP	汎用プロセッサ
バタフライ演算	5ステップ	44ステップ

表にあるように急激なステップ数の減少がみられた.要因としてはバタフライ演算を並列に行なっているのと複雑な複素数乗算をひとつの命令にまとめて処理していること、実部と虚部を同時に計算していることなどが挙げられる.

次にTEMPとTRUPSとの違いである斜め方向への接続の効果について実験した.同じ命令を持つが斜めへの接続を持たないTEMPにおいてバタフライ演算を実行するための命令の割り当てを考えた.しかしどんな割り当てをしても斜めがあるときのようにうまく配置することができず使わないタイルや逆に何度も使うタイル、が出てきてしまった.また迂回するために無駄にタイルを使ってしまうことも分かった. TEMP のタイルの数を増やしても同様にうまく配置することはできなかった. 結果今回の実装に関しては斜め接続が効果的であるといえる。

5.まとめ

本稿ではマルチメディア処理用のサブプロセッサとしてタイルプロセッサにバタフライ演算を実装させた.プロセッサのデザイン、命令を決定し、効率的なタイルへの割り当て方を提案した.同時に複数のバタフライ演算を行なうことにより結果としてステップ数の大幅な減少が見られた. 今後はタイルプロセッサのデザインの考察、TEMPを使った具体的なFFTの計算方法、またそれに伴うメモリアドレッシング方法などを決めていくことが課題として挙げられる.そして最終的にタイルプロセッサを組み込む前後での演算処理速度の向上率、既存のマルチメディアプロセッサとの処理速度の比較などを行なう予定である.またタイルプロセッサの並列性を生かしフォルトトレランス性を持たせることも今後の課題として挙げることができる.

参考文献

[1]吉瀬謙二 ：新世代マイクロプロセッサアーキテクチャ（前編）:1.アーキテクチャ基盤技術 6.タイルプロセッサ, 情報処理学会誌「情報処理」2005年10月.

[2] K. Sankaralingam, R. Nagarajan, H. Liu, J. Huh, C.K. Kim D. Burger, S.W. Keckler, and C.R. Moore. :Exploiting ILP, TLP, and DLP Using Polymorphism in the TRIPS Architecture, 30th Annual International Symposium on Computer Architecture (ISCA), pp. 422-433, June 2003.

[3] Sharp, J. A. : Data Flow Computing, Ellis Horwood Limited, Chichester, England 1985