計算機アーキテクチャ 146－１６
（２００２．２．１）

メディアプロセッサ MAPCA のデータ転送方式

細木 浩二†，東嶋 重樹†，田代 卓††，川口 敦生†，西岡 清和†
日立製作所 システム開発研究所†
日立製作所 デバイス開発センタ††

メディアプロセッサ MAPCA において、データキャッシュを含むメモリ領域間のデータ転送を行うデータ転送エンジンを設けることにより、データキャッシュへのプリフェッチ及び吐出し処理と、演算器によるメディア処理を並列に実行する。このデータ転送エンジンは、プログラマブルなアドレス生成器と内部メモリを持ち、画像データなどの 2 次元構造のデータを 1 次元化、及び逆変換を処理しながら、データ転送を行う。本方式により、HD 解像度画像の横成分を 2 分の 1 にダウンサンプリングする実時間ビデオデコーダにおいて、100% のデータキャッシュヒット率を達成し、640M サイクル毎秒要した処理を 263M サイクル毎秒にて実現した。

A data transfer implementation on media processor MAPCA

Koji Hosogi† , Shigeki Higashijima† , Takashi Tashiro†† , Atsuo Kawaguchi† , Kiyokazu Nishioka†
Systems Development Laboratory, Hitachi, Ltd.†
Device Development Center, Hitachi, Ltd.††

This paper describes a data transfer architecture for a media-processor. Our approach is to have a data transfer engine which can transfer data between memory and data cache.It can handle a data cache pre-fetch and write-back from data cache in parallel with media processing.

On a HD video down decoder, this approach achieved no data cache miss and succeeded real-time decoding in 263MHz frequency.

１まえがき

デジタルTVや高品質ビデオ会議などのマルチメディア処理には数十GOPSを超える処理性能が要求されるため、アーキテクチャ面から様々なアプローチが試みられてきた。従来から使われてきた機能固定の ASIC を用いた手法では、標準規格の多様化、新規格の登場、ユーザ要求の複雑化などの流れに対して、迅速な対応が困難であった。そこで、近年、メディアプロセッサが注目されている。

メディアプロセッサは、多様なメディア演算器を持ち、C 言語などの高級言語による開発が可能なため、新規機能への迅速な対応と、画像処理や音声処理など、異なった機能の対応などの利点を持つ。

マルチメディア処理は、高い演算性能を必要とすると共に、高いデータ転送性能を必要とする。特に MPEG-2 実時間デコード処理など、数 M バイトにわたる大容量のデータに対し、順次処理を行うメディア処理では、そのデータ構造に空間的局所性はあるが、時間的局所性はない。一般的に、プロセッサの性能を決める 1 要因であるデータキャッシュは、空間的且つ時間的局所性のあるデータ構造に対し有効であるが、特にデータキャッシュ容量に対して使用メモリ容量が大きい場合、キャッシュとしての性能を十分に引き出すことは困難である。

我々はこの問題を、メディア演算器とは独立に動作する、プログラマブルなデータ転送エンジン（データ・ストリーマ [DataStreamer^{TM}]）を用い、メディア演算とデータ転送を並列実行可能なアーキテクチャにより解決し、メディアプロセッサ MAPCA[1] を開発した。

本稿では、MAPCA の概要と、本アーキテクチャによるデータ転送方式、及び、デジタルビデオ向けの HD（1080i）フォーマット（MPEG-2 フォーマットによる 1920*1080 画素、30 フレーム毎秒のインタレースモード）[2] のダウンデコードによる性能評価について述べる。

2 メディアプロセッサ MAPCA

メディアプロセッサMAPCAのブロック図を図 1 に示す。メディア演算器は 136 ビット命令長の VLIW (Very Long Instruction Word) アーキテクチャ [3] を使用し、300MHz 動作時、30GOPS の演算性能を持つ。メディア演算器は命令キャッシュ (32KB)、データキャッシュ (32KB) と接続する。また、可変長符号処理に特化した可変長符号コプロセッサ(VLx)を持つマルチプロセッサ構造を採る。

データストリーマは、異なるメモリ領域（主記憶や I/O デバイス、VLx など）間のデータ転送を行う、プログラマブルなデータ転送エンジンである。特に、本メモリ領域として、データキャッシュを指定可能な点が特徴である。

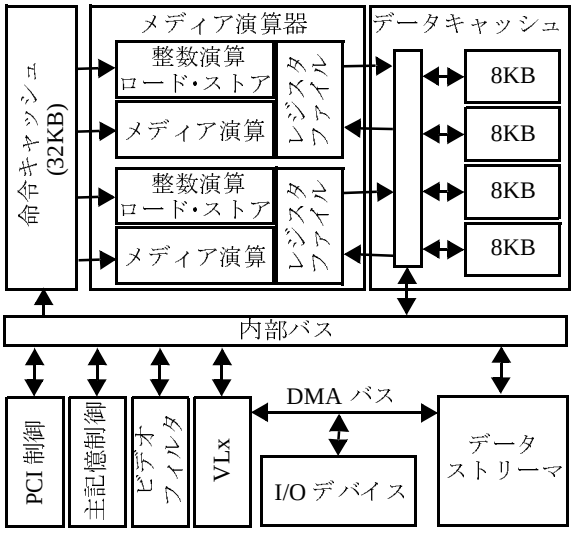


図 1　MAPCA ブロック図

DataStreamer^{TM} はイクエーターテクノロジ - 社の商標です。

更に、周辺回路には、ビデオやオーディオ、PCI などの多彩な入出力と、画像の拡大・縮小を行うビデオ・フィルタ等の搭載により、様々なアプリケーションに対応可能である。

3 データ・ストリーマ

3.1 Gather-Scatter 型データ転送

データ･ストリーマは、プログラマブルなデータ転送エンジンで、メモリ－メモリ間転送、及び I/O デバイス－メモリ間のデータ転送を行う。データ･ストリーマは64チャンネルのメモリ転送用チャネルを持ち、その情報を元に、アドレス生成し、メモリと内部バッファ間にてデータ転送を行う (図 2)。

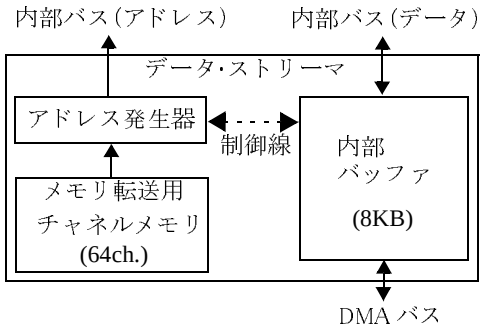


図 2 データ・ストリーマのブロック図

全てのメモリ転送は 2 つのチャネルを使用し、以下のステップによりデータ転送を行う。

step1) 入力側チャネルにて、メモリ ->内部バッファ転送を行う。

step2) step1 終了を検出後、出力側チャネルにて、内部バッファ -> メモリの転送を行う。

すなわち、入力側チャネルは、メモリから内部バッファにデータを集め (Gather チャネル)、出力側チャネルにより、内部バッファのデータを振分ける (Scatter 型チャネル)。このように、1 つのメモリ－メモリ転送に Gather チャネルと Scatter チャネルを使用する為、64 チャネル持つデータ･ストリーマは、同時に 32 種のデータ転送を可能とする。

各チャネルは、デスクリプタと呼ばれる、データ転送を記述した、一種のユーザ･プログラムにより、そのデータ転送を記述できる。また、各チャネルには、同一のバッファ領域を示すバッ

ファ ID を設定し、スコアボーディング [4] 機能を用いて、同期化する。

一方、データの供給先及び供給元が I/O デバイスの場合、各 I/O デバイスがマスタとなり、本バッファに対してデータのリード・ライトを行う。I/O デバイス - メモリ転送の場合、バッファ -> メモリ転送のみにチャネルを使用し、その逆は、メモリ -> バッファ転送のみにチャネルを使用することにより、メモリ－メモリ転送と同様の制御によって DMA 転送を可能とする。

3.2 アドレス変換

一般に 1 画面あたり数 M バイトの画像データは、2 次元配列のデータ形式をとり、マクロブロックなどの単位で処理を実行する場合、その規則的な配置により、特に低容量で連想度の低いキャッシュメモリを使用すると、確実にブロックの衝突が発生し、スラッシングが激増する。

この問題を考慮し、データ・ストリーマによるデータ転送では、ある配列のデータ構造を別の次元に変換しながら実行できる。例えば、主記憶上に格納されたマクロブロックの場合、2 次元配列構造をとる。まず、データ・ストリーマの転送により 1 次元の連続データに変換し、メディア演算器は、この変換された連続データをアクセスすることで、ダイレクトマップのキャッシュメモリを使用しても、スラッシングは発生しない。同様に、メディア演算器が生成した連続データを、データ・ストリーマにより 2 次元に変換しながら転送することで、主記憶上には、2 次元のフレーム構造を形成できる (図 3)。

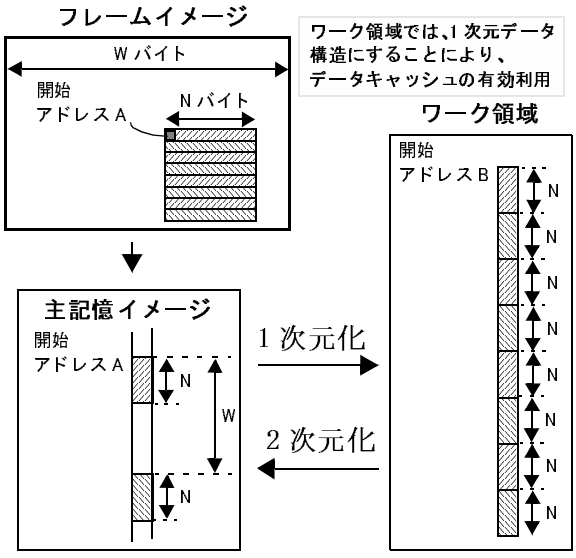


図 3 データ・ストリーマのアドレス生成

このような配列変換を有するアドレス生成を行う為、デスクリプタには、転送開始アドレス、データ幅、ピッチ、繰返し数などを設定する事により、任意サイズのブロック転送を可能とする。

3.3 チェイン型ディスクリプタ

1 画面分の画像データなど、データ転送中に対象となるデータ領域の構造 (開始アドレス、転送量など) が変化しない場合、1 ペアの Gather-Scatter チャネルを使用することにより、データ転送を可能とする。

しかしながら、演算器がマクロブロック単位で逐次処理をするような場合、1 ペアのチャネルでは表現できないため、複数のデスクリプタを使用する。更に、動き補償処理など、演算結果によって、そのアドレス情報が生成されるような場合、その度にデスクリプタを生成し、データ・ストリーマとの同期をとりながら、データ転送を起動しなければならない。

この問題を回避する為、デスクリプタには、次のデスクリプタが格納されているアドレスを設定するフィールドを設け、現在の転送が終了後、次のデスクリプタを自動的にフェッチする機能を設けた。これにより、ソフトウェアにて、次のデスクリプタを生成し、メモリ領域にライトするのみで、データ・ストリーマの内部情報をポーリングすることなく、データ・ストリーマは自動的に次のデータ転送を行う事が可能で、ポーリング等の同期処理による性能低下を回避できる。

3.4 コヒーレント・モード

データ・ストリーマは、3.2 項に示したデータ転送を可能とするが、主記憶上の画像データを配列変換し、再度、データキャッシュのスヌープを用いながら主記憶に転送すると、主記憶容量を 2 倍必要とし、内部バスの負荷が増加する。我々は、表 1 に示すコヒーレント・モード [5] を用い、データキャッシュ自身を、データ転送元、及び転送先に直接指定可能とすることで、この問題を解決した。

Coherently Allocate モードによるデータキャッシュへのライトアクセスは、キャッシュミスが発生した場合でも、そのブロックをデータキャッシュ上にアロケートするため、データキャッシュへのプリフェッチとして動作し、本データ転送後

に、メディア演算器がロード命令を発行する場合、確実にキャッシュヒットする。

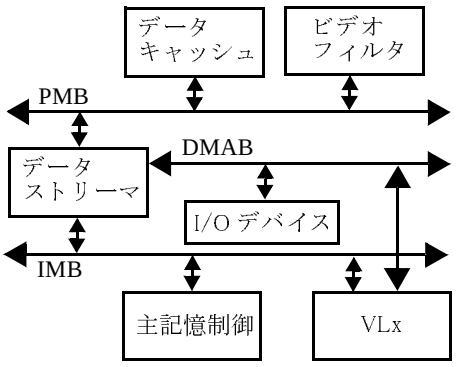
一方、Coherently (No-)Allocate モードによるデータキャッシュからのリードアクセスは、データキャッシュからのデータの吐出しとして動作し、例えば、メディア演算器が生成した画像データを、演算器の介入なしに、主記憶上に配置されたフレーム・バッファなどに転送できる。

表 1：コヒーレント・モード

コヒーレントモード	動　作
Coherently Allocate	データキャッシュに対するキャッシャブルなアクセス。キャッシュミス時、データキャッシュ上にそのブロックをアロケートする。
Coherently No-Allocate	通常のスヌープと同様に振舞い、データキャッシュヒットの場合、データキャッシュに対しリード・ライトを行う（ライト時は主記憶も更新）。データキャッシュミスの場合、主記憶に対してリード・ライトを実行し、データキャッシュのアロケートは行わない。
Non-Coherently	スヌープ処理を伴わないデータ転送。

4 内部バス・トポロジ

このように、データ・ストリーマは Gather-Scatter型データ転送エンジンであり、一つのデータ転送に、内部バスを 2 回使用する。マルチメディア処理などの大容量データ転送が必要な場合、余分なデータ転送量を増やすことは、トラフィック・ネックに繋がる。この問題に対し、複数のバスを用い、データ転送を分散化することにより解決した。図 4 に MAPCA の内部バス・トポロジを示す。



PMB: Processor Memory Bus
IMB: Internal Memory Bus
DMAB : Direct Memory Access Bus

図 4 内部バス・トポロジ

バス分散の基本方針は、ペアの Gather 転送と Scatter転送では同一バスを使用しない事により、バス使用率の平均化を行う事である。従って、アプリケーションのデータフローを検討した結果、主記憶 - データキャッシュ間、主記憶 - ビデオ・フィルタ間、データキャッシュ -VLx 間などを別のバスに配置し、データ・ストリーマ内のデータパスを 2 つのバスに接続する形態とした。本トポロジにより、同時に 2 種のデータ転送が可能となり、バスのトラフィック・ネックが回避できる。また、チップの実装面において、フロアプランの最適化により、配線長、及び負荷数を低減し、クリティカルパス対策も実現している。

5 ソフトウェア・パイプライン

次に、MAPCA アーキテクチャに適した、メディア演算器のソフトウェア・モデルについて述べる。一般に、メディア処理を高速実行するには、処理を複数に分割し、それぞれの処理を、複数の演算器にて並列に実行する。

しかしながら、分割された処理同士では、互いに依存関係を有し、処理間のデータ転送が生じる。データ転送に要するレイテンシが大きい場合、性能低下が発生する。従って、データ転送も並列実行することにより、高い処理性能を得ることが可能である。

我々のデータ・ストリーマによるアプローチでは、メディア演算器とデータ転送の並列処理が可能なため、これを実現できる。

また、メディア処理で必要な数Mバイトといった大容量データを、メモリ・アロケーション管理したデータキャッシュ内の同一の小容量領域に、繰返しプリフェッチとデータの吐出しを行い、更に、データキャッシュをダブルバッファとして使用することにより、効率的なデータ転送を可能とする。

すなわち、現在のデータが格納された、ダブルバッファの一方のセットには、メディア演算器がアクセスし、これと同時に、他方のセットにデータ・ストリーマによって、次のデータをプリフェッチする、もしくは、過去に生成された演算結果をデータキャッシュから吐出すといった、データキャッシュの排他的な並列使用によって、データ転送によるオーバヘッドを隠すことが可能である（図 5）。

従って、メディア演算器の処理とデータ転送処理を、ソフトウェア・レベルにて、完全にパイ

プライン化することにより、全ての処理が並列実行可能となる。

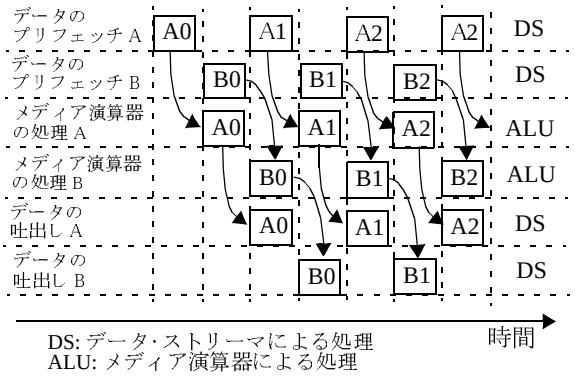


図 5 ソフトウェア・パイプライン

6 性能評価

本アーキテクチャを使用したメディアプロセッサ MAPCA の性能評価を行った。評価に使用したプログラムは、HD(1080i) フォーマットのビットストリーム画像の横方向を 2 分の 1 にダウンサンプリングし、従来のアナログテレビに出力するダウンデコーダ [6] で、ハフマン符号化、逆離散コサイン変換、動き補償、画像生成、及び DRC 出力などに伴うデータ転送を、データ・ストリーマにより実行する (図 6)。

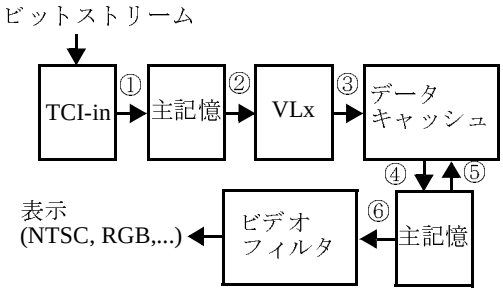
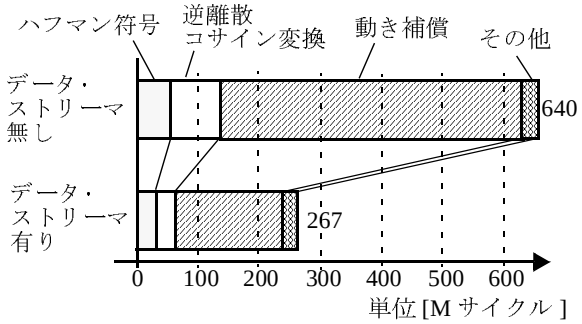


図 6 ダウンデコーダのデータフロー

この結果、640M サイクル毎秒要していた実時間デコードのプログラムが、データ・ストリーマの使用により、263M サイクル毎秒で実現できた。この時、データキャッシュミスは、初期設定時に 65 回発生し、他の全てのキャッシュアクセスは、キャッシュヒットし、キャッシュヒット率 100% を実現した (図 7)。



(1 秒あたり I フレーム 2 枚、P フレーム 8 枚、B フレーム 20 枚の場合)

データキャッシュヒット率 **100%** を実現

図 7 ダウンデコードの必要サイクル数

7 まとめ

マルチメディア処理のデータ転送を考慮した、メディアプロセッサ MAPCA を開発し、評価用の全 HD(1080i) フォーマットの画像を、周波数 270MHz にて実時間ダウンデコードできた。

今後は、より性能を必要とする HD フルデコードや、高精彩 TV 出力実現のため、より効率化なデータ転送方式の提案を行う。また、内蔵 DRAM や専用エンジンなど、近年のデバイスやアーキテクチャのトレンドを踏まえ、メディアプロセッサのアーキテクチャ開発への反映を行う。

参考文献

[1] 小島他 :“メディアプロセッサ (MAP) のビジョンとアーキテクチャ”, 第 59 回情報処理学会全国大会 ,IC-1,1999

[2] ISO/IEC: “Information technology-Generic coding of moving pictures and associated audio information”,ISO/IEC 13818, 1996

[3] Mike Johnson: “Superscalar Microprocessor Design” PTR Prentice Hall, Inc., 1991

[4] John L. Hennessy, David A. Patterson: ″Computer Architecture : A Quantitative Approach”,Morgan Kaufmann Publishers, Inc., 1990

[5] Daniel E. Lenoski, Wolf-Dietrich Weber: “Scalable Shared-Memory Multiprocessing”, Morgan Kaufmann Publishers, Inc., 1995

[6] 廣井他 :” VLIW 型メディアプロセッサを用いた MPEG-2 ビデオデコーダ”, 映像情報メディア学会技術報告 ,Vol.25, No.21, PP.25 ～ 30”,2001