Cell Broadband Engine を用いた 超解像度ソフトウェアのリアルタイム実行

田中 康之† 田中 明良† 山下 道生† 今田 敬† 高橋 幸恵† 加藤 宣弘†
株式会社 東芝 コアテクノロジーセンター‡

1．はじめに

アナログ放送映像やDVDフォーマット映像などで使用されるSD(Standard Definition)画像を、ハイビジョン放送などで使用される HD(High Definition)画像へ変換する際に、原画像を綺麗に拡大する超解像度技術が注目されている。

超解像度技術のひとつである再構成方式は、同じ輝度変化を異なる領域から標本化し画素値を算出する方式で、被写体本来の輝度を再現できる特長がある。再構成方式には複数フレーム方式と単一フレーム方式がある。複数フレーム方式では、前後のフレームから動きを検出し、参照したフレームの画素値を処理フレームの標本値として処理を行う。一方、単一フレーム方式は同一フレーム内の画素値を標本化して処理する方式である。

超解像度技術では、画素単位の演算が必要なため処理負荷が大きく、映像処理で重要なリアルタイム実行を実現するために高い演算処理能力が必要となる。

高性能なマルチコアプロセッサである Cell Broadband Engine(以下、Cell)は高い演算処理能力を持ち、ソフトウェアを最適に設計することにより、その能力を最大限に引き出すことができる。今回、超解像度ソフトウェアを Cell 向けに最適化設計することによって、リアルタイム実行を実現できたので、その手法について述べる。

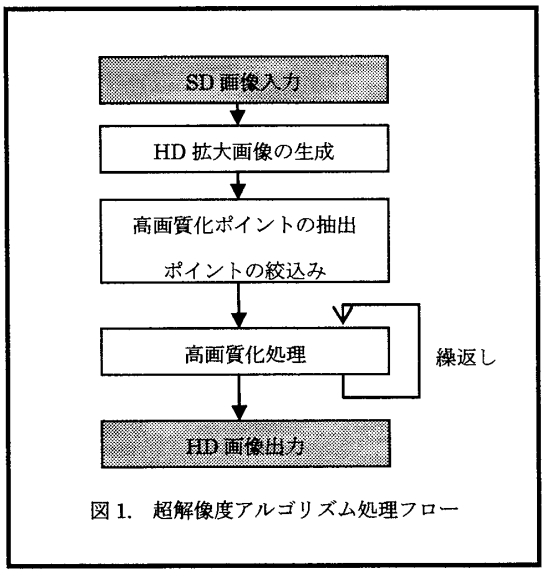

図 1. 超解像度アルゴリズム処理フロー

Real time processing of super resolution software on Cell Broadband Engine
†Yasuyuki Tanaka, Akira Tanaka, Michio Yamashita, Kei Imada, Yukie Takahashi, Nobuhiro Kato
‡Core Technology Center, TOSHIBA Corporation

2．超解像度ソフトウェアアルゴリズム

本論文では単一フレーム方式の超解像度変換技術[1]を採用している。その理由として、複数のフレーム方式に比べて処理負荷が軽い、データを保持するためのメモリ量が少なくすむ、そして各フレームごとに独立して処理を行うことができる、という点が挙げられる。これらの特長は Cell上でのソフトウェア実装に向いていると言える。

図 1 に超解像度アルゴリズムの処理フローを示す。まず入力された 1 フレームのSD画像 (720x480)を通常の拡大フィルタを用いてHD画像(1920x1080)に拡大する[2]。次に、同一フレーム内の別の点の画素値を基に高画質化ポイントの抽出を行う。ここでは、さらに高画質化の効果が高いポイントを優先して処理するようなポイントの絞込みを行う。そして、抽出された画素の情報から高画質化処理を繰り返し行う。繰り返し処理を行うことで効果を強く反映させることができる。これにより、綺麗に拡大したHD画像を生成することができる。

3．Cellプラットホームと超解像度モジュール概要[3]

Cell は 1 個の Power Processor Elements (PPE)コアと 8 個の Synergistic Processor Elements (SPE)コアを持ったマルチコアプロセッサである。PPE では OS が動作し、画像ストリームデータの外部入出力と SPE プログラムの起動制御を行う。SPE は演算処理部 (SPU) と DMA 機能を持つメモリフローコントロール (MFC) および 256KB のローカルストアメモリ (LS) から構成される。SPU は、非対称な 2 本の命令パイプライン(even/odd pipeline)を持っているため、命令にレジスタ間の依存関係がなければ 2 命令同時発行が可能である。また、SPU 命令実行と DMA 転送は並行して動作することが可能で、データ転送を隠蔽した命令実行が可能となる。

超解像度プログラムモジュール本体は SPE 上で動作させる。

4．Cell 上での超解像度モジュール最適化手法

超解像度モジュールは処理負荷が大きく、リアルタイム化のためには処理時間を大きく削減しなければならない。本章では、超解像度ソフトウェアの Cell 向け最適化手法について述べる。

4．1 SPE 向け演算処理部の最適化

超解像度の負荷の高い部分についてSPE演算命令のSIMD化を行う[3]。SPUではfloat型ならば 4 データ、byte 型なら 16 データ同時に演算することができ、spu_shffule 命令を使用することでレジスタ上に連続したデータとして割り付けることができる。

また、超解像度モジュールは 1 フレーム処理するためには SD 画像、HD 画像分のデータ領域が必要になる。LS ではメモリサイズが不足するため、メインメモリと LS メ

モリ間でのデータ転送が必要となる。ここでは図2で示すように、SD画像およびHD画像を縦15分割、横60分割した900ブロックに対してデータ転送、超解像度処理を行う。

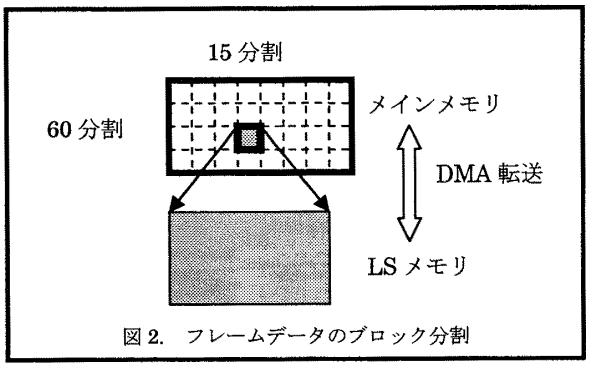

図2. フレームデータのブロック分割

4．2　データ転送隠蔽

4.1で述べたように、SPE上での演算処理には必ずメインメモリとLSメモリ間でのデータ転送が必要となる。しかし、データ転送に大きな時間がかかるため、演算処理部分とデータ転送処理をオーバーラップさせ、データ転送の隠蔽を行う。特に、「HD拡大画像の生成」では、演算処理結果に依存関係がないため、データ転送のための待ち時間を短縮することができる。

4．3　パイプラインストール削減

「高画質化処理」ではアルゴリズムの性質上、ロード・ストア待ちのために生じるパイプラインストールが発生する。これは、注目画素に対して高画質化処理した結果を再び次の処理に使用するため、演算処理が完了した後、次の処理を行うまでの間にストア、ロードが完了しておらず命令発行に待ちが生じているためである。

ここでは、データ転送を隠蔽することができないため、図3のように3点分並列に行うように変更を行う。この変更によりパイプラインストールを削減し高速化を図ることができる。

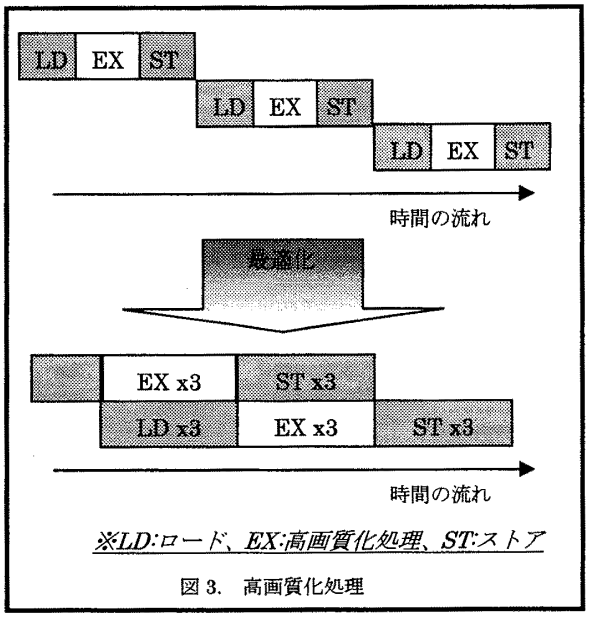

図3. 高画質化処理

5．評価

超解像度ソフトウェアを複数のSPEで並列実行する場合には、2通りの方式が考えられる。1つは1フレームを分割し、各SPEに分割したデータを割り付ける方式であり、もう1つは1フレームを各SPEコアに割り付ける方式である。

今回、1フレーム単位での分割方式を採用した。1フレームデータを各SPEに分割する方式の場合、データ分割・結合のためのオーバヘッドが発生してしまうのに対し、1フレーム単位でSPEに割付ける方式では、SPEの数をフレキシブルに変更することが可能となり、拡張性が高くなるからである。

図4.にSPEにおける超解像度処理の流れを示す。SPE0は全体制御し、残りのSPEが各フレームに対する超解像度処理を実行する。SPE0は他のSPEに対して処理の開始通知とフレームデータの割付、終了後にそれぞれの処理完了通知を受け取ることよりSPE全体の同期制御をしている。

4章で述べた最適化手法を実施することにより、超解像度モジュールの処理時間を最適化前の約1/3に削減することができ、リアルタイム実行が可能になった。

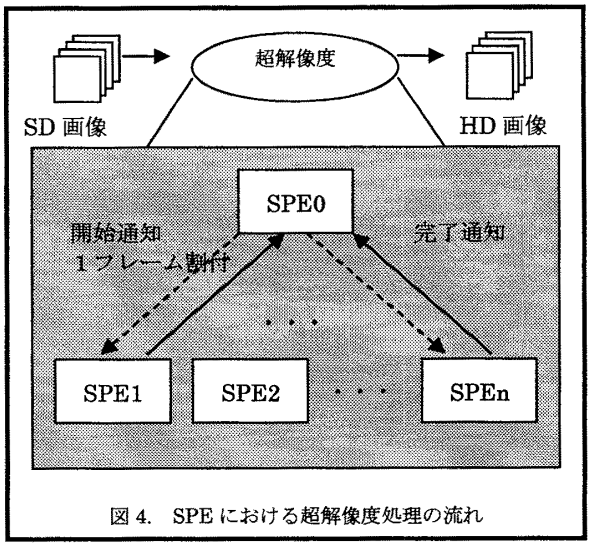

図4. SPEにおける超解像度処理の流れ

6．おわりに

SD画像をHD画像へ変換する超解像処理についてCellの高い演算処理能力を活用し、ソフトウェアをCell向けに最適化することでリアルタイム実行を実現できた。

今後、超解像度のアルゴリズム改善による高速化、高画質化を行う予定である。

参考文献

[1] 井田, 松本, 五十川, "画像の自己合同性を利用した再構成型超解像" 電子情報通信学会技術報告, CS2007-52, IE2007-135, pp.135-140, Dec. 2007.
[2] 高橋, 今田, 高山, 山下, 境, "Cubic Convolutionフィルタの Cell/SPE 向け最適化実装" 情報処理学会全国大会　講演論文集 (2) ,3B-2, pp2-15- 2-16, Mar.2006
[3] http://cell.scei.co.jp/j_download.html
CBE_Architecture_v101_j.pdf
SPU_language_extensions_v23_j.pdf