In-place 処理に基づく離散ウェーブレット変換の CUDA による高速化の検討

生澤 拓也 [1]　伊野 文彦 [2]　萩原 兼一 [2]

[1] 大阪大学基礎工学部情報科学科，[2] 大阪大学大学院情報科学研究科

1　はじめに

離散ウェーブレット変換（DWT）は周波数解析手法の 1 つであり，ノイズ除去やデータ圧縮などに応用されている．これらの応用は実時間処理を求めていて，DWT の高速化が必要である．そこで，高いメモリ帯域幅を持つ GPU による高速化がなされている．

Laan ら [1] は，計算量の少ないリフティング方式による DWT を統合開発環境 CUDA 上に実装している．リフティング方式では，不連続なメモリ領域のみを 1 度に更新できるため，単純な実装はメモリコアレッシングを実現できず性能が低い．そこで，この既存手法は連続領域を更新できるようデータを並び替えたのちに，グローバルメモリに書き出す．しかし，入力領域および出力領域を区別する必要があり，リフティング方式の特長である in-place 処理を実現できない．

そこで本研究では，DWT の高速化を目的として，in-place 処理を実現できるリフティング方式の CUDA 実装を検討する．リフティング方式を構成する一連の処理のうち，in-place 処理を妨げるデータの並び替えについてメモリコアレッシングを実現する．

2　ウェーブレット・リフティング

n 個の要素からなる入力を $x = x_0, x_1, \ldots, x_{n-1}$ とすると，DWT が出力する低域成分 $c = c_0, c_1, \ldots, c_{n/2-1}$ および高域成分 $d = d_0, d_1, \ldots, d_{n/2-1}$ は次式で与えられる．

$$d_t = x_{2t+1} - \sum_{i=t-k}^{t+k-1} p_i x_{2i} \quad (1)$$

$$c_t = x_{2t} + \sum_{i=t-k+1}^{t+k} u_i d_i \quad (2)$$

ここで，$0 \le t \le n/2-1$ であり，定数 p，u および k は基底関数により定まる．

リフティング方式は in-place 処理を実現するために，x_{2t+1} および x_{2t} の領域をそれぞれ d_t および c_t として上書きする．したがって，c および d の各要素は交互に配置される．

$$c_0, d_0, c_1, d_1, \ldots, c_{n/2-1}, d_{n/2-1} \quad (3)$$

この結果，単純には入力および出力の双方に対するメモリコアレッシングは両立できない．

3　提案手法

提案手法は式 (3) の要素列を in-place 処理にて以下のように並び替える．

$$c_0, c_1, \ldots, c_{n/2-1}, d_0, d_1, \ldots, d_{n/2-1} \quad (4)$$

さらに，この並び替えにおいてメモリコアレッシングを実現することにより実行効率の高い DWT 実装を実現する．並び替えは以下に挙げる 2 つのフェーズにより処理する．

フェーズ 1　入力データ x を m 個に分割し，各々をスレッドブロックに割り当てる．各スレッドブロックは共有メモリ上で低域成分の集合 C_i および高域成分の集合 D_i を生成する．ここで，C_i および D_i はそれぞれ i 番目のスレッドブロックが生成した集合を表す．メモリコアレッシングがなされるように，C_i および D_i をグローバルメモリに書き出す．つまり，k を自然数として各集合は $32k$ 個の要素を内包する．

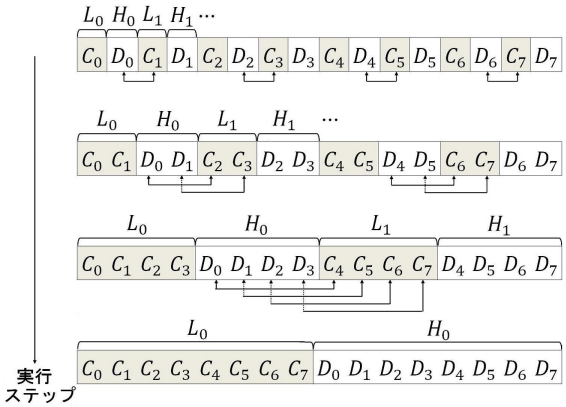

図 1: フェーズ 2 の動作例

フェーズ 2　グローバルメモリに書き出した低域成分および高域成分のうち，連続領域に存在する集合の集合を先頭からそれぞれ L_j および H_j $(j = 0, 1, \ldots, m-1)$ とする（図 1）．フェーズ 2 の初期状態では，$L_j = C_j$ および $H_j = D_j$ である．以下の処理を $m = 1$ になるまで繰り返す．

1. H_{2t} および L_{2t+1} $(0 \le t \le m/2-1)$ を交換する．各スレッドブロックは H_{2t} および L_{2t+1} が内包する集合のうち 1 対の交換を担当する．フェーズ 1 にて示した通り，これらの集合に対する読み書きはメモリコアレッシングの条件を満たせる．
2. $L_t = L_{2t} \cup L_{2t+1}$，$H_t = H_{2t} \cup H_{2t+1}$ $(0 \le t \le m/2-1)$
3. $m = m/2$

図 1 はフェーズ 2 の動作例を $m = 8$ として示している．図中の矢印は，1 個のスレッドブロックが担当する交換を表す．

4　実験

提案手法の性能を評価するために，実行時間に関して既存手法と比較した．実験に使用した GPU は NVIDIA GeForce GTX 470 であり，CUDA バージョンは 4.2 である．使用したデータは 8192 × 8192 画素からなる 2 次元画像であり，ウェーブレットは Daslauriers-Dubuc(13,7) である．

提案手法は既存手法と比較しておよそ 1/4 の性能であり，実行時間は 17.6 ミリ秒から 64.9 ミリ秒に増加した．性能が低下した原因は並び替え処理のオーバヘッドにあり，グローバルメモリの参照量が 5.3 倍に増えたことによる．なお，メモリコアレッシングのおかげでメモリ帯域幅に関する実行効率は 83.4%に達しているため，参照量を削減することが性能改善のために必要である．

今後の課題としては，グローバルメモリの参照量を削減できる並び替えの実現が挙げられる．

参考文献

[1] W.J. van der Laan, A.C. Jalba, and J.B.T.M. Roerdink, “Accelerating Wavelet Lifting on Graphics Hardware Using CUDA,” IEEE Trans. Parallel and Distributed Systems, 22(1):132–146, Jan. 2011.

ⓒ 2013 Information Processing Society of Japan