社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－ＡＶＭ－57（５）
2007／7／19

組込みマルチコアプロセッサ向け H.264 ビデオデコーダの並列化

西原康介　幡生敦史　森吉達治

NEC システム IP コア研究所 〒211-8666 川崎市中原区下沼部 1753

E-mail: k-nishihara@az.jp.nec.com, a-hatabu@bc.jp.nec.com, moriyosi@ce.jp.nec.com

あらまし H.264 は携帯機器向けビデオコーデックとして広く利用されてきているが，高負荷の演算処理を要求するためソフトウェア処理をするには高性能なプロセッサが必要である．一方，組込み機器向けの高性能プロセッサとして，低消費電力でかつ大幅な性能向上が期待されるマルチコアプロセッサが注目されている．しかし，マルチコアプロセッサの性能を引き出すためには，プロセッサコア間の演算負荷の不均等やメモリアクセス競合への対応が不可欠である．そこで，H.264 デコーダのマルチコアプロセッサ上での高速並列動作を目的とした，H.264 デコード処理の特性に応じ分割方法を適用する同期コストを抑えた負荷分散法と，同期待ちをしているアイドルコアにおいてプリロード量を適応的に調節することで待機時間を有効に使いメモリアクセス競合を低減するプリロード手法を提案した．マルチコアプロセッサ MPCore を用いた評価では，1 コアでのデコード性能が 217Mcycle per second (cps)であったのに対し，4 コア 4 並列動作では 96Mcps を実現し，約 2.25 倍の高速化を達成した．デコード性能の向上により動作周波数および電源電圧を下げることができ，推定ではマルチコアを用いることによりシングルコアに比べ 79%の消費電力で H.264 ビットストリームのデコードを実現することが可能となった．

キーワード H.264, マルチコアプロセッサ, プリロード

Parallelization of H.264 Video Decoder for Embedded Multicore Processor

西原康介　幡生敦史　森吉達治

NEC System IP Core Research Laboratories

1753 Shimonumabe, Nakahara-Ku, Kawasaki, Kanagawa, 211-8666 Japan

E-mail: k-nishihara@az.jp.nec.com, a-hatabu@bc.jp.nec.com, moriyosi@ce.jp.nec.com

Abstract H.264 video codec, which is widely-used for mobile devices, requires higher workload than previous standards. Thus a high-performance processor is needed to implement an H.264 decoder in software. Multicore processors are remarkable because of its high performance with low power consumption. To effectively utilize the multicore processor, the load balancing method and the reduction method of memory access contention are required. In this paper, two methods are proposed to enhance parallelized H.264 decoder performance on the multicore processor. One is a workload partitioning which can balance the load by applying appropriate parallelization policy to each H.264 decoding function. And another is the preloading method with adaptive control of the amount of data to preload, which can reduce memory access contention without an additional load. By using both proposed methods on the multicore processor MPCore, the computational load of the decoder is reduced from 217Mcps to 96Mcps and consequently the parallelized H.264 decoder obtains 2.25 times higher processing speed than non-parallel one.

Keyword H.264, Multicore Processor, Preload

1. はじめに

近年，携帯機器向けビデオコーデックとして H.264/MPEG-4 AVC(以下 H.264 と表記)が広く利用されてきている．H.264 は従来のコーデックより高い符号化効率を実現するが，高負荷な演算処理を要求し，ソフトウェア処理には高性能なプロセッサが必要であった．一般に，シングルコアプロセッサの性能改善では，動作周波数の向上による消費電力増加を招くという問題点があり，低消費電力化が要求される組込み向けプロセッサへの適応は困難である．このため，1 チップに複数コアを集積することで，動作周波数を低く抑えたまま，低消費電力でかつ大幅な性能向上が期待されるマルチコアプロセッサが注目されている．

しかし，マルチコアプロセッサの性能を引き出すためには，逐次処理にはなかったプロセッサコア間の演算負荷の不均等や，メモリアクセス競合などへの対応が不可欠であり，並列処理に適したアルゴリズムが必要になる．従来の H.264 コーデックの並列処理はいず

れもマクロブロック単位で並列処理するものであり[1]-[4]，同期を頻繁に取る必要があるため同期処理のオーバーヘッドが問題になっている．また，一般的なマルチコアプロセッサにおけるメモリ共有型のアークテクチャでは，複数コアからのメモリアクセスが同時に起こった場合，1 つのコアしかバスを占有できないためメモリアクセス競合が発生し，性能劣化が顕著になるという問題がある．

そこで，本稿では H.264 デコーダのマルチコアプロセッサ上での高速並列動作を実現するため，H.264 デコード処理特性を考慮し，同期コストを抑えた負荷分散方法を提案する．また，アイドル状態のコアで適応的にプリロードすることで，待機時間を有効に使いメモリアクセス競合を低減する手法を検討し，並列処理に適した H.264 デコーダの開発を行った．

2. H.264 デコーダの並列化

メモリ共有型マルチコアプロセッサを用いて並列化を行うための高速化の課題として，一般的に次の 3 つが挙げられる．

(1) コア間の演算負荷を均等化する処理分割
(2) コア間の同期処理のオーバーヘッド削減
(3) メモリアクセス競合の低減

本稿では，(1)，(2)に関して，H.264 のデコード処理の依存関係等の特性に応じた分割方法を適用し，同期コストを抑えた負荷均等化ができる負荷分散方法を提案する．また，(3)に関しては，アイドル状態のコアで同期待ちの待機時間に合わせ適応的にプリロードすることで，待機時間を有効に使いメモリアクセス競合を低減する手法を提案する．

2.1. デコード処理の特性に応じた負荷分散

H.264 デコード処理は，可変長符号復号（VLD: Variable Length Decoding），動き予測（MC: Motion Compensation）またはイントラ予測（IP: Intra Prediction），逆整数変換，逆量子化，そしてループフィルタ（LF: Loop Filter）を順に行う（本デコーダでは，逆変換および逆量子化は，MC または IP とセットで行うため，今後 MC＋逆変換＋逆量子化，IP＋逆変換＋逆量子化の処理を，それぞれ MC, IP と代表して呼ぶ）．

デコード処理の最小単位は，16×16 画素で構成されるマクロブロック（MB）と呼ばれるブロックであり，並列処理のための処理分割として MB 単位で負荷を分散する方法が最も容易である．ただし，H.264 デコード処理の特性として，IP や LF は図 1に示される A～D の隣接する MB の情報を用いる可能性があるため，各 MB を独立に並列処理することはできず，ある MB を処理する際に依存関係のある MB の処理の終了を確認する必要がある．

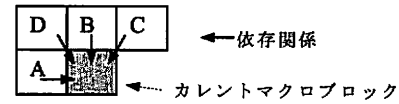

図 1　マクロブロックの依存関係

MB 単位の並列方法の例として，Wave-Front 分割と呼ばれる分割法が知られている[3]．Wave-Front 分割とは，図 1 に示す依存関係が解決している MB から各プロセッサコアに処理を割り当てる方法である．Wave-Front 分割の簡単な実装では，前述した依存関係を解決するために，図 2に示すように右上 MB の処理終了を確認しながら処理を進める．この結果，デコード処理はフレームの左上から波面のように進行していく．この方法は，MB 単位で負荷を分散でき，分散効率は高い．しかし，MB 毎に同期処理を行う必要があり，同期動作自体の処理時間が大きくなるという問題がある．さらに，ある MB が同期待ちをした場合，その左下の MB は処理を進められず待機してしまい，1 つの MB の遅延が全体の処理を遅らせてしまう．このように，MB 単位の並列処理は，同期処理に伴うオーバーヘッドが大きくなるという問題がある．

本稿では，同期によるオーバーヘッドを抑え負荷分散の効率を高める目的で，MB 単位の並列処理は行わず，H.264 デコード処理を構成する各部の依存関係等の特性に応じた負荷分散方法を提案する．Wave-Front 分割は MB 単位の粒度の細かい分割が原因で，同期のオーバーヘッドが問題になった．このため，できるだけ分割の粒度を大きくできるよう，VLD, MC, IP, LF の各処理をフレーム単位で順に行い，それぞれの特性に合わせ分割した．

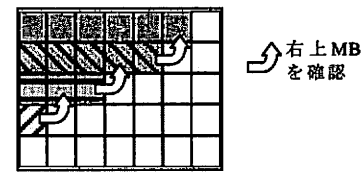

図 2　Wave-Front 分割

表 1　H.264 デコード処理特性と並列処理方針

処理名	処理特性	並列処理方針
VLD	逐次実行が必要，分割処理はできない	1 フレーム先行実行，前フレーム LF と並列実行
MC	MB 間に依存がなく並列化が容易，デコード処理に占める割合大	空間分割，分割割当変更で負荷均等化
IP	隣接 MB に依存（図 1 の A~D），並列化には依存解決が必要，デコードでの割合小	MB ライン単位の Wave-Front 分割
LF	依存（図 1の A,B）はあるが，その影響は複数 MB にわたり伝播せず独立に並列処理可能	MB ライン単位で空間分割し，分割境界 1MB ライン分を再処理

デコード各処理の特性と並列化の方針を表 1に示した．まず，MC および LF はフレームを空間分割し，同期を取らず独立に処理を行った．MC は MB 間に依存関係はないため，並列化は容易であり MB 単位で独立に処理することができる．

LF については，依存関係は存在するが，その影響は境界にのみ現れ複数 MB にわたり伝播しない．このため，LF を適用した後に境界 MB を再処理することで独立処理が可能になる[5]．境界処理負荷低減のため，縦方向に MB ライン単位で分割数分空間的に分割する．例えば，QVGA サイズ(320×240pixel)を 3 分割する場合，5MB ラインずつ各コアに割り当て，さらに 2 つのコアに 1 MB ライン分の境界処理を加える．ただし，この分割法では他コアの境界処理終了を待つコアが発生する．次節において，このアイドルコアの待機時間を有効利用するプリロード法について説明する．

次に，VLD は逐次実行が必要で分割処理はできないため，前フレームの LF と並列に実行させることで，VLD が性能のボトルネックとなることを回避した．

通常 VLD と LF が同時に終了することはなく，終了時間がばらつくことになる．このばらつきを MC の処理の割り当てを適応的に変化させることで吸収させた．一般に MC はデコード処理に占める割合が高く，また，MB 間の依存が無いという特徴がある．これより割り当て領域を動的に変更でき，分配された各 MC の終了をそろえることで分散効率を上げることができた．

一方，IP は隣接 MB との依存があり，他処理とは異なり分割領域間で同期を取りながら並列実行する必要がある．本デコーダでは，MB ラインを単位とした Wave-Front 分割と同様の並列方法を採用している．ただし，一般に，IP はデコード処理に占める割合が小さく，たとえ処理がばらついたとしてもデコード全体に与える影響は小さい．

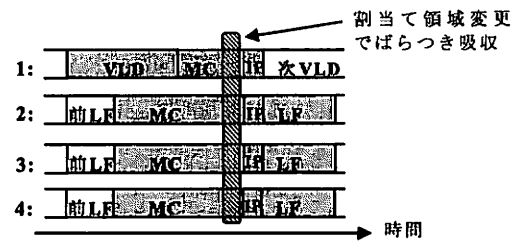

図 3　1 フレーム処理スケジューリング

上記のデコード処理の分割，分散方法により，同期コストを抑えつつ負荷の均等分散が実現できた．4 並列動作の場合における，1 フレームのデコード処理の各コアへのスケジューリングを図 3 に示す．

2.2. アイドル時間を有効利用する適応的プリロード

メモリ共有型のアークテクチャでは，複数のコアからのメモリアクセスが同時に起こった場合，1 つのコアしかバスを占有できず，他のコアからの要求は待たされてしまう．一般に，外部メモリへのアクセスには多くのサイクルが必要になるため，キャッシュミスが起こった場合，アクセス競合による処理速度低下が顕著になる．本稿では，キャッシュを各コアが共有するアーキテクチャを想定し，キャッシュと外部メモリの間のアクセス競合を低減する方法を検討した．

本デコード処理では，特に，MC で作成した予測画像と差分画像の加算部でメモリアクセスが頻発し，この箇所を並列処理する場合にアクセス競合が顕著に観測された．これは，作成した画像を新しいフレームバッファに書き込むため，キャッシュミスが発生し外部メモリに頻繁にアクセスするからである(図 4)．一方，LF では直前に書き込まれたフレームバッファの画像に対しフィルタ処理を行うため，メモリアクセスがキャッシュにヒットしやすい．外部メモリへのアクセスの少ない LF 処理時に MC で用いるフレームバッファ領域のプリロードを行うことで，プリロードと他処理との間の競合を抑えながら使用領域をキャッシュに載せることができる．

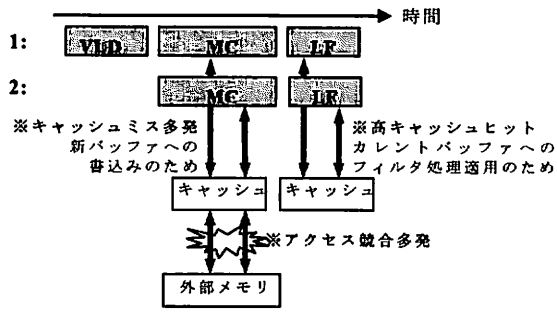

図 4　メモリアクセスの概要

筆者らは，MC でのメモリアクセス競合を低減する目的でプリロード手法の検討を行ってきた[9][10]．これらの手法はプリロード処理をコアに専用に割当てており，本来必要では無い負荷が発生していた．並列処理での依存関係の結果他のコアを待ち合わせているアイドル状態のコアでプリロードを行うことができれば，負荷を増やすことなくさらに効率的にプリロードを行うことができる．

2.1節で述べた負荷分散法では，LF と MC の間で同期待ちのためにアイドル状態が発生する．図 3 では，分割された 3 つの LF は同時に終了するように表現しているが，分割数から 1 引いた数分の境界処理が必要であり，いずれか 1 つのコアは他コアの境界処理終了まで待つ必要がある(図 5)．図 5では，第 2，第 3 のコアが境界処理を担当し，第 4 のコアは境界処理終了を待ち合わせている状態を示した．この待機時間をプリロードに充てることで，余計な負荷の増加なくプリロードが可能になる．

ただし，割り当てられた LF の処理終了後に，MC で必要なフレームバッファ領域を全て連続にプリロードした場合，待機時間内にプリロードが終了しないことや，アクセス集中による競合が原因で，逆にデコード所要サイクルを延ばしてしまうことがありえる．このため，フレームバッファ領域をすべてプリロードするのではなく，他コアの LF と同時に終了できる割合だけプリロードを行う．図 6は，プリロードする割合を変えた場合の，ある入力ストリームのデコード所要サイクルを示している．この例では，全領域の 4 割のプリロード量が最適であることを示している．

LF の所要サイクル数は入力ストリームに依存し，他コアと同時に終了するプリロードの最適な割合を事前に決めておくことはできない．そこで，実行中に最適割合を決定するため，始めに初期状態を決めた後，フレーム終了時にコアの進行状況を評価し割合を更新する方法を採った．具体的には，フレーム領域を 10 等分し，プリロードを行うコアが他コアの LF より時間がかかった場合にはプリロードの割合を減らし，逆に時間が余った場合は割合を増やすように制御を行った．

また，プリロードによるメモリアクセスの集中を避けるため，プリロードを LF 後にまとめて行うのではなく，複数に分割し LF を 1 MB ライン処理した後に 1 分割分適用した．図 5では，第 4 のコアでプリロードを行い，LF+プリロードの処理が第 2，第 3 のコアの LF と同時に終了するように制御する例を示した．これにより，他コアとの競合をできるだけ抑え，入力ストリームに依存することなくアイドルコアの待機時間を有効利用したプリロードが可能になった．

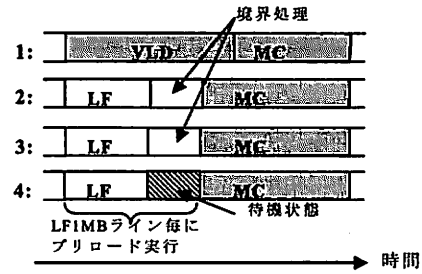

図 5　プリロード例

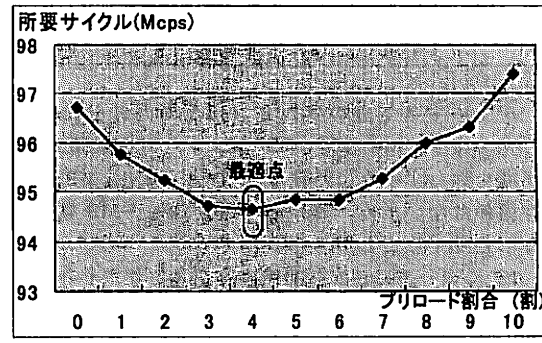

図 6　プリロード割合によるデコード所要サイクル

3. 並列化 H.264 デコーダの性能評価

ARM11 を 4 つ搭載するメモリ共有型のマルチコアプロセッサである MPCore[6]を用いて，並列化デコーダの性能評価を行った．MPCore は，図 7に示すように L1 キャッシュを各コアが保有し，L2 キャッシュを共有する構成になっている．評価に用いたテストボード[7]の概要を表 2に示した．

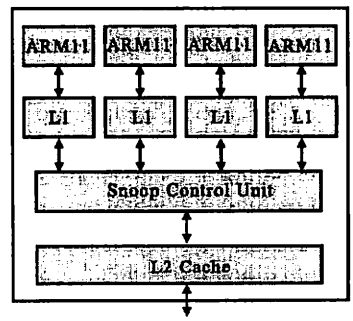

図 7　MPCore ブロック図

表 2　テストボードの概要

プロセッサ	ARM11MPCore
1 次キャッシュ	命令: 32 KB データ: 32 KB
2 次キャッシュ	1 MB
コア動作周波数	210MHz
バス動作周波数	30MHz

2.1節で行った並列化を適用した場合の，並列度を変えた際の並列度 1 に対する性能比（左目盛り）とデコード所要サイクル数（右目盛り）を図 8に示した．評価には，ITE 標準ストリーム 9 種を JM エンコーダ[8]を用い QVGA サイズ，30fps，512kbps，N=60 の条件でエンコードしたものを使い，その平均を取っている．図 9には，4 並列動作時のデコード処理の内訳を示す．LF の並列処理においてアイドル状態になっている箇所があるが，この部分が他コアの境界処理終了を待ち合わせている所である．図 8より 4 並列動作での所要サイクル数は 99Mcps(cycles per second)，並列度 1 の場合は 217Mcps であり，4 並列動作でも並列度 1 に対する性能が 2.19 倍とどまっていることがわかる．このように性能が並列度に対し向上しない主な要因は，メモリアクセス競合によるものであると考えられる．

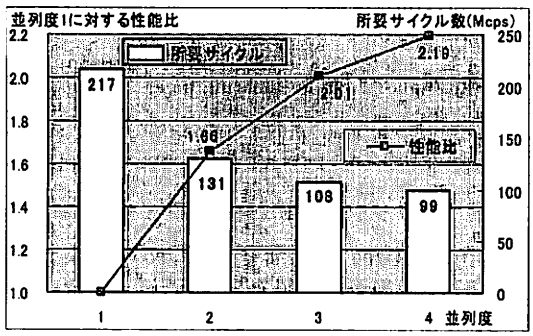

図 8　並列化 H.264 デコーダの性能

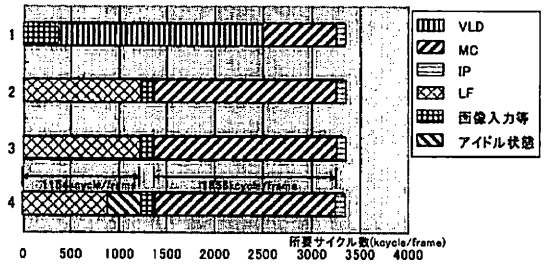

図 9　デコード処理の内訳(並列化のみ)

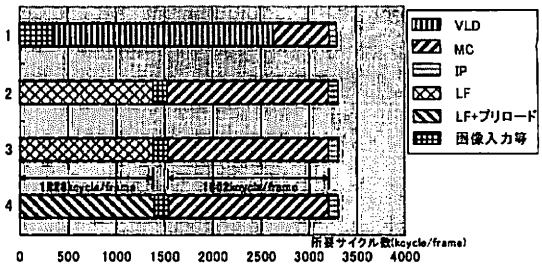

図 10　デコード処理の内訳(プリロード適用)

表 3　プリロードの効果

	LF 所要サイクル数	MC 所要サイクル数	合計
プリロード適用前 (並列化のみ)	1154	1858	3012
プリロード適用後	1223	1662	2885
差分	69	-196	-127

単位 Kcycle/frame

そこで，2.2節で提案した適応制御プリロードを適用したときのデコード処理の内訳を図 10に示す．また，プリロードによる 1 コア当たりの MC と LF の所要サイクル数の変化を表 3に示す．表より，プリロードを行うことにより LF の所要サイクル数は増えるが，MC 所要サイクル数の減少がそれより大きいことがわかる．4 並列動作時の，9 種のストリームでの評価結果を図 11に示す．適応制御プリロードを用いた場合は平均で 96Mcps のデコード性能が得られており，プリロードを行わない場合より 3%の性能改善になっている．並列度 1 に対する性能比も 2.25 倍に向上しており，本稿で提案した負荷分割とプリロードを併用した手法が有効であることを示している．

さらに，デコード所要サイクル数の低減は，システムのコア動作周波数を下げられることを意味している．一般に，CMOS LSI 回路の動作時の消費電力は，動作周波数と電源電圧の二乗の積に比例することが知られており，また，動作周波数の低減は電源電圧の低減を容易にする[11]．ここで仮に，シングルコアと 4 コアのマルチコアの動作周波数が 217MHz，96MHz，および電源電圧が 1.8V，1.2V であるシステムを想定した場合(表 4)，マルチコアの消費電力はシングルコアに比べ 79%に抑えることができる．このようにマルチコアプロセッサを用いることで，電源の低減幅がたとえ小さい場合でも，H.264 ビットストリームのデコードを低消費電力で実現することが可能となる．

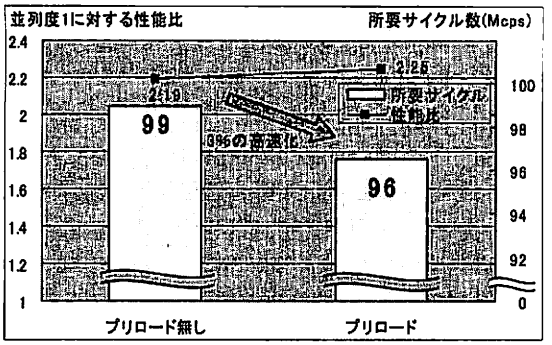

図 11 プリロードによる性能向上

表 4 消費電力推定

	動作周波数	電源電圧
シングルコア	217 MHz	1.8V
マルチコア	96MHz	1.2V
消費電力比 (マルチ/シングル)	0.79	

4. おわりに

本稿では，H.264 デコーダのマルチコアプロセッサ上での高速並列動作を目的とした負荷分散方法とプリロード手法を提案した．まず，H.264 デコード処理の依存関係等の特性に応じ分割方法を適用することで同期処理のオーバーヘッドを抑える負荷分散方法を提案し，さらに，コア間の依存関係のために待機しているアイドル状態のコアで，その待機時間に合わせ適応的にプリロード量を調節することで負荷の増加を抑えメモリアクセス競合を低減するプリロード手法を提案した．

QVGA 512kbps 30fps のビットストリームを用い，前記負荷分散を適用した並列化デコーダの性能を測定した結果，並列化しない場合(所要サイクル 217Mcps)に比べ 4 並列動作(同 99Mcps)させることにより，2.19 倍のデコード性能を達成した．さらに，前記プリロードを適用することで，所要サイクル 96Mcps のデコード性能が得られ，並列化しない場合に比べ 2.25 倍まで性能を高めることができた．また，デコード性能の向上により動作周波数および電源電圧を下げることができ，推定ではマルチコアを用いることによりシングルコアに比べ 79%の消費電力で H.264 ビットストリームのデコードを実現することが可能となった．

本稿では，キャッシュと外部メモリの間のアクセス競合に注目したが，各コアとキャッシュとの間の競合も性能低下の原因であり，今後検討を行いたい．

文 献

[1] E. B. van der Tol, E. G. T. Jaspers, and R. H. Gelderblom, "Mapping of H.264 Decoding on a Multiprocessor Architecture," SPIE Conference on Image and Video Communications and Processing, vol. 5022 pp. 707-718, Jan. 2003.

[2] To-Wei Chen, Yu-Wen Huang, Tung-Chien Chen, Yu-Han Chen, Chuan-Yung Tsai, and Liang-Gee Chen, "Architecture design of H.264/AVC decoder with hybrid task pipelining for high definition videos," Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on, Vol. 3, pp. 2931-2934, May 2005.

[3] Zhuo Zhao, Ping Liang, "Data partition for wavefront parallelization of H.264 video encoder," in Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on, pp. 2669-2672, May 2006.

[4] YK Chen, EQ Li, X Zhou, and S Ge, "Implementation of H.264 encoder and decoder on personal computers," J. VCIR, 2006.

[5] 森吉遠治，宮崎孝，"ソフトウェア H.264 エンコーダの並列化，" 2006 年電子情報通信学会総合大会，D-11-33.

[6] ARM Limited, "ARM11 MPCore Processor," DDI 0360B.

[7] ARM Limited, "Core Tile for ARM11 MPCore," HBI-0146,

ARM Limited, "RealView Emulation BaseBoard," HBI-0140,

ARM Limited,"Application Note, Using a CT11MPCore with the RealView Emulation Board," DAI 0152B.

[8] JVT Reference Software, version 10.2, http://iphome.hhi.de/suehring/tml/.

[9] 西原康介，蟠生敦史，森吉遠治，"組込みマルチコアプロセッサ向け H.264 ビデオデコーダの並列化，" 2007 年電子情報通信学会総合大会，D-11-26.

[10] 西原康介，蟠生敦史，森吉遠治，"組込みマルチコアプロセッサ向け H.264 ビデオデコーダ開発，" マルチメディア，分散，協調とモバイル (DICOMO2007)シンポジウム，2007 年 7 月.

[11] 桜井貴康編著，"低消費電力，高速 LSI 技術，" リアライズ社，1998.2.