Vol.2010-ARC-187 No.21
Vol.2010-EMB-15 No.21
2010/1/29

メディア処理向けカスタムプロセッサにおける復号処理命令拡張の検討

國武勇次[†1] 久村孝寛[†2,†3] 安浦寛人[†1,†4]

ビデオやオーディオなどのメディア処理の効率化を図るために，SIMD や VLIW などのデータ並列性を利用するアーキテクチャがこれまで盛んに研究されてきた．その結果，データ並列性のある DCT や動き探索などの処理は非常に高速に処理できるようになった．その一方で，データ並列性のない逐次的な処理は高速化しにくいため，メディア処理における逐次処理にかかる実行時間の割合が大きくなっている．逐次的な処理の代表例として可変長符号処理がある．従来は，可変長符号処理を専用回路で実装することが多かったが，近年はソフトウェアによる柔軟性を重視して専用演算器をもったプロセッサによる実装例もある．本稿では，ソフトによる柔軟性を備えた後者の実装例に着目し，ツール生成環境を利用した可変長復号化に適したプロセッサの命令セット検討について述べる．ベースとなる組み込みプロセッサ V850E に対して，可変長復号化に適した命令を 2 つ追加することで，可変長復号化にかかる命令数を約 55 %まで削減できることを確認した．

Extended Study of Decoding Instructions in a Custom Processor for Media Processing

Yuji Kunitake,[†1] Takahiro Kumura[†2,†3] and Hiroto Yasuura[†1,†4]

In order to execute efficiently video and audio cording, many architectures using data parallelism like SIMD and VLIW have been studied so far. As a result, signal processing tasks having inherent data parallelism in their algorithms such as discrete cosine transform (DCT) and motion compensation (MC) have been handled efficiently on those architectures. On the other hand, serial tasks which have no data parallelism are difficult to speed up. It makes the portion of the processing time taken for serial tasks more significant compared with that of tasks having data parallelism. Variable length coding and decoding is one of such serial tasks and its implementation can be classified into two categories: dedicated circuits and application-specific processor with dedicated functional units. The latter method has been used recently to exploit the flexibility of its software implementation. In this paper, we present a case study on instruction set extension of a processor for variable length decoding (VLD). In this case study a tool generator which generates software toolchain is used to explore the instruction set enhancement. Through this case study, we show that the number of instructions required for VLD can be reduced to 55% by using dedicated instructions for VLD.

1. はじめに

デジタル TV やテレビ会議などのマルチメデイア処理を効率良く実現するために SIMD や VLIW などのデータ並列性を利用するプロセッサアーキテクチャがこれまで盛んに研究されてきた．その結果，データ並列性のある DCT や動き探索などの処理は非常に高速に処理できるようになった．その一方で，データ並列性のない逐次的な処理は高速化しにくいため，メディア処理における逐次処理にかかる実行時間の割合が大きくなっている．

逐次的な処理の代表例として可変長符号処理がある．従来は，可変長符号処理を専用回路で実装することが多かったが，近年はソフトウェアによる柔軟性を重視して専用演算器をもったプロセッサによる実装例もある．これは専用回路に比べてコスト，性能，消費電力の面で劣っていたカスタムプロセッサでもトランジスタの集積度の向上によって十分に低コストかつ高性能，低消費電力で実現可能になったためである．従って本稿ではカスタムプロセッサを用いたソフトウェアでの実装を対象としてマルチメディア処理の高速化を検討する．

複数のプロセッサで処理を分担する場合には，逐次処理である可変長符号処理が特にボトルネックになりやすい．可変長復号化によって得られたデータは逆 DCT などの処理に使われる．可変長復号化の処理性能が低い場合，これらの処理を並列に実行しても逆 DCT を行うデータが効率良く供給されないため VLIW や SIMD による性能向上効果を効率良く引き出せない．従って，逐次処理である可変長復号処理の高速化が重要となる．

本稿ではベースとなる組込みプロセッサ V850E[1)] を対象として，可変長復号化に適した

†1 九州大学
Kyushu University
†2 日本電気株式会社 システム IP コア研究所
System IP Core Research Laboratories, NEC Corporation
†3 大阪大学
Osaka University
†4 独立行政法人科学技術進行振興機構，CREST
Japan Science and Technology Agency, CREST

© 2010 Information Processing Society of Japan

Vol.2010-ARC-187 No.21
Vol.2010-EMB-15 No.21
2010/1/29

表 1 MPEG2 の復号処理における可変長復号処理の割合

	実行命令数	割合
可変長復号処理	49,262,489	6.9%
GetBits 関数	22,072,689	3.1%
復号処理全体	709,440,977	-

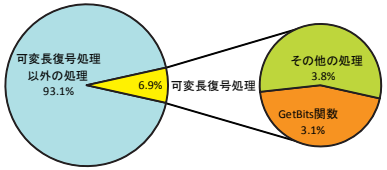

図 1 MPEG2 の復号処理における可変長符号処理の割合と内訳

プロセッサの命令セットを検討する．可変長復号処理のビットストリーム読み込み処理と符号語同定処理に着目し，各処理に対して 2 つの拡張命令を検討する．符号語を同定するためのアルゴリズムには大きく分けてツリー型とテーブル参照型[2),3)] の 2 つがある．本稿では符号語同定処理に対してテーブル参照型である Group-based 法[2)] のアルゴリズムを基に，階層的に符号語を同定することによりハードウェアコストを抑えたアーキテクチャを提案する．

2. 可変長復号処理

可変長復号処理は，ビットストリームの先頭ビットから逐次的に対応する符号語を取り出し，符号語に対応するシンボルを特定する処理である．可変長復号処理を大きくわけると以下の 3 つの処理にわけることが出来る．可変長復号処理では，復号化するビットストリームが無くなるまでこれらの処理が繰り返し行われる．

(1) ビットストリームの読み込み
(2) 符号語の同定
(3) 符号語に対するシンボルの取得

表 1 と図 1 に復号処理全体 (逆 DCT，逆量子化，動き探索などを含む復号処理) に対する可変長復号処理の実行命令数の割合とその内訳を示す．これは，MPEG2 の復号処理を行うサンプルプログラムを PC 上で実行した場合の割合を示している．これより可変長復号処理の割合は復号処理全体の約 6.9% であることが分かる．この解析結果から可変長復号処理にかかる命令を専用命令に置き換えても最大でも 6.9% 以上の削減は見込めない．しかし，本研究ではハードウェアの構成を，逐次処理である可変長符号処理のみ V850E で実行し，逆 DCT や動き探索などの処理はデータ並列性を利用できる異なるプロセッサで実行することを想定している．従って，専用命令命令の適用により可変長復号処理を高速化できれば復号処理全体を大きく高速化できると考えられる．従って以降，V850E では逐次処理である可変長符号処理部分のみ実行する構成を前提に議論する．

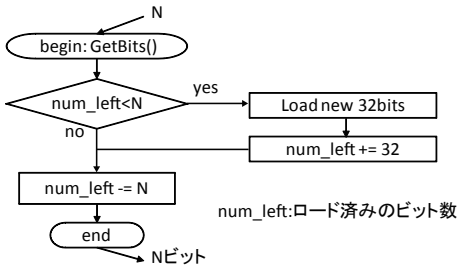

図 2 GetBits 関数の動作フロー

2.1 ビットストリームの読み込み

ビットストリームの読み込み処理は，任意長のビット列をメモリに格納されているビットストリームから必要なビット数だけ読み出す処理である．図 2 にビットストリームの読み出し処理を行う関数 GetBits() の例を示す．この例で GetBits() は，任意長 N(32bit 以下) のビット列を読み出す関数として実装されている．メモリからロードされたビットストリームは，一旦小容量のバッファに格納される．バッファに格納されているビットの MSB(Most Significant Bit) 側から N ビットの値が返される．図 2 中の num_left はメモリからロード済みのビット数を表している．既にロード済みのビット数が N より大きい場合 (num_left $\geq$ N)，バッファの MSB 側の N ビットが取り出され GetBits() の返り値として返される．ロード済みのビット数が N よりも小さい場合 (num_left $<$ N)， メモリより 32bit のデータがロードされバッファ内のデータの LSB(Least Significant Bit) 側に連結される．その後バッファの MSB 側から N ビット分ビットストリームが取り出される．GetBits() の処理は非常に単純である．しかし同定された任意の符号語長だけビットストリームを読み込むため，逐次的にしか処理が出来ない．GetBits() はロード済みのデータ長によって処理が分岐するため，条件分岐命令や条件実行命令などが使われると考えられる．条件分岐命令が使われると分岐レイテンシが発生し，条件実行命令が使われると条件成立/非成立の両方の命令が必要となるため GetBits() の処理が可変長復号処理に占める実行サイクル数の割合が大きくなる．また，MPEG2 の復号処理のサンプルプログラムを PC 上で実行した場合，可変長復号処理に占めるビットストリーム読み出し処理の実行命令数の割合は約 45%であった．従って，GetBits() を一命令で実現できれば，可変長復号処理にかかる実行サイクル数，実行命令数ともに削減できると考えられる．

© 2010 Information Processing Society of Japan

Vol.2010-ARC-187 No.21
Vol.2010-EMB-15 No.21
2010/1/29

2.2 符号語の同定

次に符号の同定について示す．符号語の同定とは，ビットストリームの先頭ビット列がどの符号語と一致しているか特定する処理のことを言う．符号語の同定アルゴリズムは数多く提案されている．大きく分けると，ツリー型とテーブル参照型のアルゴリズムに分けられる．ツリー型のアルゴリズムはビットストリームの先頭から順に 1 ビットづつバイナリツリーを葉までたどることで符号語を同定する．従って，1bit づつしか復号化できないため符号語ごとに復号化にかかる実行命令数が異なる．さらにソフトウェアで実装した場合，符号語長と同じ回数メモリを参照しなければならない．このためリアルタイム性が求められるアプリケーションをソフトウェアで実装するにはツリー型のアルゴリズムは向いていない．

一方，テーブル参照型のアルゴリズムは符号語の特性からテーブルのオフセットを求め，符号語とシンボルが対応づけられたテーブルを参照することで復号化する．従って，どの符号語に対しても復号化にかかる命令数はほぼ同じであり，メモリ参照回数はテーブルを参照する 1 回程度でよい．このためリアルタイム性が求められるアプリケーションをソフトウェアで実装する場合テーブル参照型のアルゴリズムが適している．テーブル参照型のアルゴリズムは数多く提案されている．ここで，いくつかのアルゴリズムとその特徴を紹介する．

シンプルテーブルルックアップ法

シンプルテーブルルックアップ法は最もシンプルな方法で，全ての符号語を一つの LUT(Look Up Table) に登録しておく．ビットストリームの先頭のビット列から最長符号語長と同じビット数のビット列を LUT のオフセットとして利用する．この方法では，符号語が可変長なため，長さの短い符号語は重複して LUT に登録されることとなり，LUT のサイズが大きくなってしまう．

リーディングサインカウント法

リーディングサインカウント法では，MSB と一致する符号語の接頭ビット数を利用して符号語のグループわけを行う．これによりシングルテーブルルックアップ法より LUT サイズを小さくすることが出来る．しかしながらグループ内の符号語長が複数存在する場合 LUT のサイズは最小とはならない．

プリフィックスグルーピング法

プリフィックスグルーピング法は符号語の prefix(接頭ビット) と suffix(接尾ビット) の特徴を利用する．可変長符号の特徴として符号語長の同じ符号は prefix が共通している．これを利用してビットストリームと prefix のマッチングをとることで同定する符号語の符号語長を特定する．符号語長が得られると，符号語が同定されるため参照する LUT のオフセットが同定した符号語の prefix と suffix により求まる．符号語長ごとにグループ分けされるため，LUT に無駄なエントリが存在しない．従って LUT サイズを最小に抑えることが可能である．

Group-based 法[2)]

Group-based 法では，各符号語を最長の符号語と同じ符号語長であるとみなして扱う．このとき，短い符号語の下位ビットは不定値として扱われる．各符号語を不定値まで含めた値で昇順にソートし，実際の符号語長ごとにグループ分けを行う．これにより重複するエントリが LUT 内に存在しないためサイズを小さく抑えることが出来る．各グループの境界条件とビットストリームの値を大小比較を行うことによりグループを特定することが出来，グループ毎に符号語長は決まっているため符号語を同定することが出来る．

Two Step Group Matching 法[3)]

Two Step Group Matching 法は，リーディングサインカウント法と Group-based 法を組み合わせて二段階で LUT のオフセットを求める．まず，リーディングサインカウント法でグループを特定する．次にグループ内に Branch と呼ばれる符号語長ごとに分けられたグループを持っており，グループ内の符号語の suffix と大小比較を行うことで Branch を特定し LUT のオフセットを求める．

このように，様々なアルゴリズムが提案されているが，アルゴリズムが複雑になるとソフトウェアで実装する場合にオフセットを得るための処理に多くのサイクルを費すこととなる．

2.3 符号語に対するシンボルの取得

符号語に対するシンボルの取得の処理では，符号語の同定により求めたオフセットを基に LUT を参照することでシンボルを得る．従って，可変長復号処理に占める符号語に対するシンボルの取得処理の割合は大きくない．本研究では，ビットストリームの読み込みと符号語の同定処理に対する専用命令を検討することで可変長復号処理の高性能化を図る．

3. 可変長復号処理に適した拡張命令

本節では，表 2 に示すビットストリーム読み込み命令と符号語同定命令を提案する．

3.1 ビットストリーム読み込み命令

前節で示した関数 Getbits() は，繰り返し実行されるため可変長復号処理に占める割合が大きい．そこで Getbits() の処理を 1 命令で実行できるよう GETBITS 命令を検討する．表 2 に示すように GETBITS 命令のオペランドは Rd，Imm，Raddr の 3 つである．Rd は出力レジスタでビットストリームの先頭 32bit を格納しているレジスタを指定する．Imm は

© 2010 Information Processing Society of Japan

Vol.2010-ARC-187 No.21
Vol.2010-EMB-15 No.21
2010/1/29

表 2 提案する拡張命令

ニモニック	オペランド	説明
GETBITS	Rd, Imm, Raddr	指定された N(Imm)bit のビットストリームを読み込み Rd に返す. Raddr はビットストリームの先頭アドレスを示す.
CODEIDENT	Rd, Rin	先頭ビットストリーム Rin を入力とし，Group-based 法を基にした階層的な符号語同定回路により LUT のオフセット値を算出し Rd に返す.

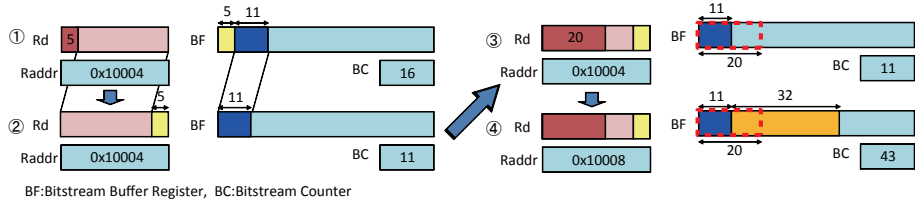

図 3 GetBits 命令の動作例

読み込むデータ長 N を即値で与える．Raddr はメモリに格納されているビットストリームの先頭アドレスを指定する．

GETBITS 命令を実装するにあたって，出力レジスタ用の 32bit シフタ，メモリからロードしてきたビット列を一時格納するための 64bit のバッファ(Bitstream Buffer Register:BF) とそれ専用の 64bit シフタが必要となる．またバッファ内の有効ビット数を保持するためのレジスタとして 6bit のレジスタ (Bitstream Counter:BC) が必要となる．

図 3 に GETBITS 命令の動作例を示す．① は初期状態を表している．BF 内に 16bit のビットストリームがロード済みの状態である．今 5bit の符号語が復号され，5bit 分のデータを Rd に補充するために GETBITS 命令が実行されたとする．まず Rd 内の値が 5bit 左シフトされる．このとき，BC は 16 を示しているので，BF 内には有効ビット数が 5bit 以上存在する．従って BF 内の先頭をそのまま Rd の LSB 側へ補充し，BF 内のビット列を 5bit 左シフトして BC を 11 に更新する．これにより状態 ② となる．次に状態 ② のとき 20bit の符号語が復号されたとする．この場合，③ の BF に示されるように，点線で囲まれた後ろ 9bit 分読み込むデータが不足している．そこで，Raddr に示されるアドレス 0x10004 からデータをロードし BF に補充する．すると，状態 ④ となり，Raddr はアドレスが 4 インクリメントされ，BC は 43 に更新される．その後 BF 内の有効ビットが読み出すビット数以上の場合と同じ操作が行われる．

3.2 符号語同定命令

前節では，複数の符号語同定アルゴリズムを示して符号語の同定方法の問題点について言及した．ソフトウェアで実装する場合，符号語とビットストリームのパターンマッチを行うための LUT のサイズと LUT のオフセットを求めるための処理性能が重要となる．そこで，LUT のサイズが最小となるような符号語同定アルゴリズムを採用し，複雑なオフセット値の計算を一命令で実行できる符号語同定命令 (CODEIDENT 命令) を検討する．

表 3 Group-based 法によるグループ分けの例

Group	Code	Boundary	Code_len	GPoffset_len	GPoffset
G1	100xxx	-	3	1	0
G2	1010xx	101000	4	2	10
	1011xx				11
	1100xx				00
G3	11010x	110100	5	1	0
	11011x				1
G4	111000	111000	6	2	00
	111001				01
	111010				10
	111011				11

3.2.1 符号語同定アルゴリズム

符号語の同定アルゴリズムには，符号語の大小関係を利用する Group-based 法[2] を利用する．表 3 に最長符号語 6bit の符号語を Group-based 法でグループ分けした例を示す．まず符号語長の短い符号語は最長符号語と比較して短いビット数分下位ビットを不定値として扱い，全て同じ符号語長とみなして昇順にソートされる．次に実際の符号語長が同じものを一つのグループとして定義する．表 3 の Boundary は各グループの境界条件を示しており各符号語の最小値をとる．Code_len と GPoffset_len は各グループの実際の符号語長とグループ内のオフセットの長さを表している．GPoffset は各符号語のグループ内でのオフセットを示しており，実際の符号語の suffix と一致する．

符号語を同定する場合，まずビットストリームの先頭ビット列 6bit を各グループの境界条件と大小比較を行う．例えばビットストリームの先頭 6bit が 110111(2) だとする．各境界の値と大小比較を行うと，(Boundary(G3)<110111(2)<Boundary(G4)) であることから同定すべき符号語は G3 内に存在することが分かる．表 3 から G3 に属する符号語の符号語長は 5 かつ GP 内オフセットの長さは 1 であることが分かる．従って，110111(2) から同定する符号語は 11011(2) であることが分かり，GP 内オフセットは符号語の LSB 側から 1bit 分の値，つまりこの例では 1(2) であることが分かる．

3.2.2 Group-based 法を用いた階層的な符号語同定回路

図 4 に Group-based 法を利用した符号語同定命令を実装した場合の回路構成を示す．

© 2010 Information Processing Society of Japan

Vol.2010-ARC-187 No.21
Vol.2010-EMB-15 No.21
2010/1/29

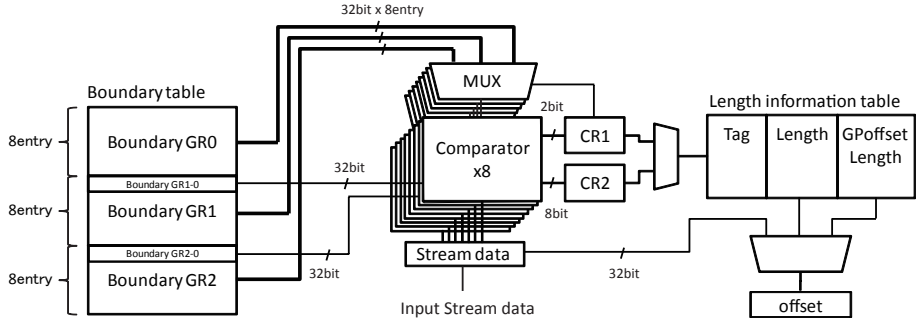

図 4 Group-based 法を用いた階層的な符号語同定回路

Boundary table(Btable) は各グループの境界条件を格納するテーブルである．エントリ数は 24 で最大 25 のグループ分けが可能である．Comparater は境界条件と先頭のビットストリームの大小比較を行うための比較器である．一回の比較でグループを特定するには 24 個の比較器が必要となる．そこでハードウェアコストを抑えるために，比較器の数を 8 個に削減し二段階に分けて比較することでグループを特定する．CR1 と CR2 はそれぞれ一回目と二回目の大小比較を行った結果を格納するためのレジスタである．それぞれのレジスタのサイズは CR1 が 2bit，CR2 が 8bit である．Length information table(LItable) は各グループ毎にエントリがあり，各グループを構成する符号語の長さとグループ内オフセットの長さが格納される．

階層的な符号語同定回路の動作について以下に示す．階層的に比較を行うために，Btable はさらに 3 つのグループに分けられている．まず，3 つのグループの境界条件である 2 つのエントリ (GR1-0,GR2-0) と先頭のビットストリームを比較し結果が CR1 に格納される．ビットストリームの方が境界条件より大きい場合に各境界条件に対応するビットが 1 にセットされる．例えば (GR1-0<bit_stream<GR2-0) のとき CR1=01(2) となる．これにより 3 つの内どのグループに属するかが特定される．次に特定されたグループの 8 個の境界条件と先頭のビットストリームを比較し結果が CR2 に格納される．CR2 も CR1 と同様に結果が格納される．次に，CR1 と CR2 の値から LItable のインデックスを求め，同定する符号語の属するグループの符号語長 CL とグループ内オフセットの長さ GOL を得る．符号語長 CL とグループ内オフセット長 GOL の値から式 (1) によりグループ内オフセットが求まる．

$$GPoffset = (bit\_stream << (CL - GOL)) >> (32 - CL) \quad (1)$$

各グループのエントリ数は $2^{GOL(GPn)}$ で求められるため，シンボルを取得するための LUT のオフセットは式 (2) で求まる．

$$Offset = 2^{GOL(GP0)} + 2^{GOL(GP1)} + \cdots + 2^{GOL(GPn-1)} + GPoffset(GPn) \quad (2)$$

ここでグループの最大数について言及する．実験に利用した符号語は MPEG2 のインターマクロブロックの DCT 係数復号と逆量子化の処理に利用されている符号語であり，これをグループ分けする場合 25 グループで十分であった．しかし文献[3] によると現在マルチメディア処理に用いられる符号語テーブルは 6-34 個のグループに分けられるとある．Two Step Group Matching 法では，prefix 毎にグループを分けた後，符号語長毎に更にグループを細分化するため，符号語長毎にグループ分けを行う Group-based 法とのグループ数に大きな違いはないと考えられる．従って想定した 25 グループよりグループ数が多くなる場合，提案する回路を拡張する必要がある．しかし比較器の数は拡張する必要は無く，Btable のエントリ数のみ増やせばよい．8 個の比較器を持つ場合，最大で $8 \times 8 + 1$ 個のグループを扱うことが可能であるため，一段目で特定するグループの候補が 8 以下であれば比較器を拡張する必要はない．

4. 拡張命令適用による効果

4.1 評価環境

提案する 2 つの命令の適用による効果を評価するために，V850E[1] をベースプロセッサとするツール生成環境を利用して，コンパイラ，アセンブラおよびシミュレータを生成しサンプルプログラムへ組み込むことで評価を行った．V850E は一命令発行の 5 段パイプラインで構成されている．命令セットは RISC 型命令セットをサポートしており，基本命令は 1 サイクルで実行することが可能である．サンプルプログラムには MPEG2 のインターマクロブロックの DCT 係数復号と逆量子化を行うプログラムを使用した．8×8 のブロック 6 個の画素を復号するサンプルプログラで，復号化する画素データはランダムで生成される．このプログラムで使用されている符号語の総数は 113 個である．Group-based 法を基にグループ分けを行うと 14 個のグループに分割することができる．

4.2 実行命令数

図 5 に拡張命令の適用前の実行命令数で適用後の値を正規化したグラフを示す．各棒グラフは左から適用前 (org)，GETBITS 命令を適用した場合 (GETBITS)，GETBITS 命令と CODEIDENT 命令を適用した場合 (GETBITS&CODEIDENT) の実行命令数をそれぞれ表している． GETBITS 命令を適用した場合，約 13%実行命令を削減することができた．

© 2010 Information Processing Society of Japan

Vol.2010-ARC-187 No.21
Vol.2010-EMB-15 No.21
2010/1/29

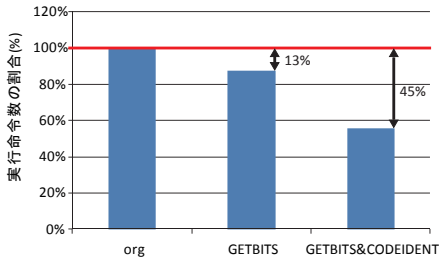

図 5 インターマクロブロックの DCT 係数復号化および逆量子化における実行命令数

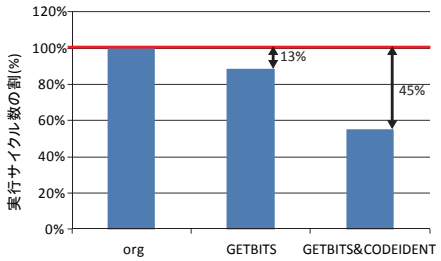

図 6 インターマクロブロックの DCT 係数の復号化および逆量子化における実行サイクル数

さらに GETBITS 命令と CODEIDENT 命令の両方を適用した場合，約 45%の実行命令を削減することができた．

4.3 実行サイクル数

提案する符号語同定命令は実行に 3 サイクルかかる．そこで，図 6 に実行サイクル数の見積りを評価した．グラフは拡張命令を適用する前の実行サイクル数で正規化されている．サンプルプログラムにおいて実行された命令の各実行サイクル数は V850E の仕様から，条件分岐命令 Bcond を 2 サイクル，無条件分岐命令 JR と JARL を 2 サイクル，JMP を 3 サイクルとした．GETBITS 命令を適用した場合，約 13%実行サイクル数を削減できた．さらに GETBITS 命令と CODEIDENT 命令を適用した場合，約 45%の実行サイクル数削減することができた．この結果は，ほぼ実行命令数の結果と同じである．従って，提案する符号語同定命令の実行に 3 サイクル必要でも十分な性能改善効果が得られると考えられる．可変長復号処理はインターマクロブロック以外にイントラマクロブロックやヘッダ部分の復号処理がありこれらにもインターマクロブロックと同様の効果が期待できる．このため可変長復号処理全体として 40-50%程度の実行サイクル数の削減できると考えられる．

4.4 テーブルサイズの比較

表 4，5 に符号語を同定するのに必要なテーブルサイズを示す．サンプルプログラムに使用した MPEG2 のインターマクロブロックの DCT 係数復号と逆量子化では，リーディングサインカウント法が採用されており，符号語長を格納した LUT とシンボルである run, level を格納した LUT の 2 つが必要となる．一方，提案する CODEIDENT 命令では，一つの LUT に符号語長とシンボルである run，level をまとめて格納している．Btable のエントリサイズは 32bit，LItable のエントリサイズは 16bit とし，これらのエントリ数は最大グループ数である 25 とする．符号語同定命令の適用前と適用後を比較すると，適用後は約半分のサイズまで削減されている．これは，テーブルを一つにまとめたことによる効果と Group-based 法によりグループを符号語長ごとに分けて LUT を構成したためテーブルの無駄を削減した効果であると考えられる．

表 4 リーディングサインカウント法に必要なテーブルサイズ

符号語長テーブル	640 byte
run,level テーブル	1,472 byte
合計	2,112 byte

表 5 符号語同定命令に必要なテーブルサイズ

符号語長，シンボル LUT	928 byte
Boundary table	100 byte
Length infomation table	50 byte
合計	1,078 byte

5. ま と め

本稿では，データ並列性のある DCT や動き探索の処理と逐次処理である可変長符号処理を異なるコアで実行する構成を想定し，高速化が困難な可変長復号処理の高速化を目指した．可変長復号処理の中でも割合の大きいビットストリーム読み込み処理と符号語同定処理を一命令で実行出来る GETBITS 命令と CODEIDENT 命令を提案した．インターマクロブロックの DCT 係数復号と逆量子化の処理に対して GETBITS 命令を適用した場合約 13%，さらに符号語同定命令を適用した場合，約 45%の実行命令を削減することができた．符号語同定命令ではハードウェアコストを抑えるために Group-based 法を基に階層的な同定を行った．これにより命令の実行サイクルが 3 サイクルとなるが，処理全体の実行サイクルにほとんど影響をあたえないことが確認された．これらの結果より，提案する 2 つの命令を採用することにより可変長復号処理の実行サイクル数を効果的に削減できる．

謝辞 本研究の一部は，科学技術振興機構 CREST プロジェクトの支援によるものである．

参 考 文 献

1) NECElectronics Corp. User's manual: V850 family for architectur. Document No. U10243EJ7V0UM00,http://www.necel.com/, March 2001.
2) B.J. Shieh, Y.S. Lee, and C.Y. Lee. A new approach of group-based vlc codec system with full table programmability. *Circuits and Systems for Video Technology, IEEE Transactions on*, Vol.11, No.2, pp. 210–221, Feb 2001.
3) S.Wisayataksin, T.Isshiki, and H.Kunieda. Entropy decoding processor for modern multimedia applications. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer*, Vol. E92-A, No.12, pp. 3248–3257, 2009.

© 2010 Information Processing Society of Japan