平成 20 年度戦略的基盤技術高度化支援事業

「次世代動画像圧縮標準規格に対応する組込システム開発支援ツールの研究開発」

研究開発成果等報告書

平成２１年３月

委託者　東北経済産業局

委託先　地方独立行政法人岩手県工業技術センター

目　次

第１章　研究開発の概要

１－１　研究開発の背景・研究目的及び目標

１）背景

近年、車載カメラやDVDレコーダの画像の高精細化に伴い、その画像データ量が膨大に増え続け、画像の品質を落とさずデータ量を減らす圧縮技術が求められている。従来規格のMPEG2では能力が不十分となってきており、MPEG2の2倍以上の高圧縮率を可能とする新規格の H.264／AVC 対応製品の開発が各メーカの急務となっている。

しかし、H.264／AVCは規格そのものが複雑・難解で、さらに規格化されていないが高圧縮率を達成するために必須の機能も存在するなど、組み込みソフトのプログラム開発は困難を極め、開発期間の短縮化が課題となっている。また、現状のH.264規格に関する開発ツールは約1億円と非常に高額であり、中小企業の開発参入の大きな障壁となっている。

２）目的及び目標

本事業は車載カメラ／情報家電メーカの要望が高い「動画像再生（デコーダ)」に的を絞った低価格の「H.264用開発支援ツール」を開発する。各メーカの開発者は、「H.264用開発支援ツール」に含まれる基本的な動画再生プログラムといくつかの付加機能を選択し、さらに開発者オリジナルの新機能を組み合わせてプログラム作成を行い製品に組み込む。既存のプログラムを組み合わせること、また新機能の開発においても「H.264 用開発支援ツール」に含まれる「フローチャート」により規格を理解し「公開ソースコード」を修正することでプログラムの開発期間の大幅な短縮が可能となる。

また、画像データの処理には専用LSIの開発が必要だが、「H.264用開発支援ツール」では開発したプログラムをFPGAと呼ばれる書き換え可能なLSIに書き込み、実際の製品を想定したシミュレーションや実機での動作検証を可能にすることで、画像処理専用LSIの開発期間短縮及び開発費の削減を図ることができる。

本ツールは、携帯電話のワンセグ放送や、次世代高精細テレビ放送、高精細テレビ電話、ブルーレイ方式のレコーダ等、次世代の高精細画像機器への応用範囲が広くニーズも高いが、競合する開発も予想されるため、以下の目標を定め開発を進める。

１－２　研究体制

研究組織及び管理体制

有限会社エボテック

株式会社イーアールアイ

地方独立行政法人岩手県工業技術センター（事業管理者兼研究実施）

１－３　成果概要

試作開発した「H.264 用開発支援ツール」は、「FPGA 開発支援ボード（ハードウェア)」と、このボード上で稼働する「専用ソフトウェア」でシステム全体を構築した。「FPGA 開発支援ボード」は、異なる 2 種類の FPGA を搭載し、並行動作や協調動作の開発を可能とした。特徴は画像を取り扱うための大容量メモリ 512MB を搭載し、周辺デバイスに DVI 出力、USB、JTAG、SSRAM、FLASH を装備し柔軟な開発が可能となっている。これによって、小規模から大規模までのアプリケーション開発を、１つのボードで行えるハードウェア環境を実現した。

「専用ソフトウェア」は、FPGA に H.264 データ供給するための「CPU ボード・ソフトウェア」と、FPGA 上で稼働する「H.264 再生コア・ソフトウェア」を試作し、H.264 コード解析が容易な開発環境を実現した。次に述べる３種類の機能が本ツールの特徴である。

①解析用ツールとして、低解像度(QCIF:176×144)から高解像度(HD：1920×1080P)の H.264 フル規格の解析を実現した。

②低解像度開発向け(CIF:320×240)に、必要な機能に絞り込んだ機能限定版を実現した。

③H.264 再生機能のコア部分の FPGA ハード回路化を行い、約 7～12%の高速化を実現した。

この FPGA ハード化に際し、プログラム内容が理解しやすい C 言語を、FPGA 回路へ自動変換するツール（Codeveloper）を使った技術を確立した。Codeveloper の C 言語による H.264 の FPGA 実装は世界初（2009 年 3 月時点）である。

１－４　当該研究開発の連絡窓口

地方独立行政法人岩手県工業技術センター
〒020-0852　岩手県盛岡市飯岡新田 3-35-2
電子情報技術部　上席専門研究員　　長谷川　辰雄

第２章　本論

本章では H.264 開発支援ツール開発の全体内容について述べる。

２－１　開発方法

H.264 動画圧縮・再生用の組込み機器開発は PC で稼働するソフト開発とは異なり、省電力・省メモリという厳しい制約下で動画処理プログラムを動作させる必要がある。試作開発したハードウェア（FPGA 開発支援ボード）は、組込み動画処理開発で一般的な動作クロック１００MHz、外部メモリが 512MByte という条件で開発した。FPGA 回路プログラムの作成は、C 言語を Verilog 言語への自動変換するソフトウェア Codeveloper を用いて記述したプログラムと、一部 Verilog 言語で作成したプログラムを組み合わせて実装した。表２－１に開発概要を示す。

表２－１　開発条件

FPGA 動作周波数	100MHz
外部メモリ(DDR2)	512Mbyte
FPGA 回路プログラム	Codeveloper 用 C 言語及び Verilog 言語

２－２　FPGA 開発支援ボード（ハードウェア）の構成

FPGA 開発支援ボード(図２－１)は、組込み動画機器実装で、開発者が試したい様々な回路ロジックに柔軟に対応できるように、大規模用途、小規模用途及び、並列処理用途の開発が可能となる設計とした。具体的には２種類の FPGA とそれらと接続される周辺デバイスおよび、外部ホストと接続するためのホスト・インタフェースで構成した。以下に本ボードの特徴を述べる。

- 32bit／33MHz PCI、32bit／33MHz MPX の２種類の外部 I/F を持ち、何れか片方を外部ホストとして使用できる。
- StratixII、CycloneII２種類の FPGA を搭載しており、実装するデコーダの大きさ、速度などにより、何れか片方を選択して動作させる事ができる。また、２種類の FPGA 間にブリッジ I/F を持ち、２つの FPGA の平行動作または、協調動作などを行うことも可能である。
- ２種類の FPGA それぞれに DVI ドライバ／ポートが接続されており、何れか片方、または両方のポートから RGB 出力ができる。
- ２種類の FPGA それぞれに様々なメモリーデバイスが接続されており、システムを自由に構築できる。
- ALTERA Daughter Card ポートを持ち、様々な ALTARA Daughter Card の中から選択して接続することにより、本ボードの機能を拡張できる。
- ロジックアナライザを接続できる Debug Port を持ち、FPGA にロジックアナライザ I/F を実装することにより、ロジックアナライザでのデバッグが可能である。

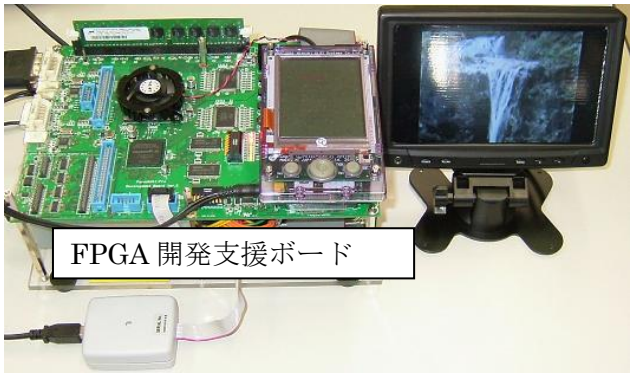


図２－１　FPGA 開発支援ボード外観図

本ボードのシステム構成を表２－２に示す。

表２－２　デバイス内容

デバイス名	説明
StratixII FPGA	大規模（9 万ゲート）、高速ロジック用 FPGA
CycloneII FPGA	小規模（3 万ゲート）、中速ロジック用 FPGA
DDR2	StratixII FPGA の外部メモリで、外部クロック 200MHz で動作する DDR2。3.2GByte/s(400MHz×8Byte)のバス帯域を持つ。プログラム格納領域、または実行領域として使用。
SDR	CycloneII FPGA の外部メモリで、外部クロック 100MHz で動作する SDR。400MByte/s（100MHz×4Byte）のバス帯域を持つ。プログラム格納領域、または実行領域として使用。
SSRAM	StratixII、CycloneII FPGA それぞれに接続される外部メモリで、外部クロック 100MHz で動作する SSRAM。プログラム格納領域、または実行領域として使用。
Flash	StratixII、CycloneII FPGA それぞれに接続される外部メモリ。プログラム格納領域として使用する。
Configuration ROM ／Configuration Port	StratixII、CycloneII FPGA それぞれをコンフィギュレーションするためのシリアル ROM。
JTAG Port	StratixII、CycloneII FPGA が JTAG チェーンで接続されている。FPGA デバッグ時に使用する。
DVI Transmitter／DVI Port	StratixII、CycloneII FPGA それぞれに接続される、各 8bit の RGB 信号を出力する DVI トランスミッタ。LCD モニタを接続してデコード画像を確認出来る。
HOST I/F／T-Engine (VR5500)	32bit、33MHz の PCI I/F。外部ホストから本ボードの制御を行う。
HOST I/F／T-Engine (SH4)	32bit、33MHz の MPX I/F。外部ホストから本ボードの制御を行う。
Daughter Card	ALTERA Daughter Card 接続用コネクタ。
Debug Port	ロジックアナライザで FPGA をデバッグする時に使用する。

２－３　H.264 デコード解析用ソフトウェアの構成

２－３－１　ソフトウェア構成

難解と言われる H.264 の組込み機器開発を容易に解析・評価するために、デコード方法を３種類に分けて評価できるソフトウェアを開発した。一つ目は H.264 規格の参照ソフトウェアである JM(Joint Model)14.0 版（以下 JM）をベースにした解析ソフトであり、H.264 フル規格の評価を目的としたものである。二つ目は H.264 規格の最小構成であるベースライン・プロファイルをさらに機能限定し、小規模の開発を目的としたものである。三つ目はデコードの一部分の機能を FPGA 上に電子回路化し、高速化の評価を目的とするものである。下記に３種類の内容を説明する。

①フル規格版

H.264 規格の全てのデコード機能について、処理の流れや計算方法、データ構造を解析・評価することを目的とした組込みソフトウェアである。ベースライン・プロファイルからハイ４：４：４プロファイルまでの対応が可能（図２－２）。FPGA 上の NIOSⅡ CPU で稼働する。

②機能限定版

監視カメラや車載カメラシステム等、ある特定の小型アプリケーションの用途に合わせ、これに必要な機能に限定し解析・評価することを目的とした組込みソフトウェア。ベース・ライン・プロファイル中の中核機能のみに限定したツールであり、H.264 デコード構造の解析入門用としても活用できる（図２－３）。FPGA 上の NIOSⅡ CPU で稼働する。

③高速化版

H.264 デコードの主要機能のうち、「動き補償」や「イントラ予測」、「デブロッキングフィルタ」を部分的に FPGA 回路化し高速化を図った組み込みソフトウェア。動き補償では８～１０％の高速化を実現した。この FPGA 回路化に際し、ソフトウェア開発で一般的な C 言語でアルゴリズムを開発し、それを FPGA 回路コード（Verilog）へ自動変換するソフト「Codeveloper」を使って実現した。Codeveloper を使用した H.264 のコーデックは世界初である（2009 年 3 月時点）。NIOSⅡ　CPU と FPGA 回路の協調で稼働する。

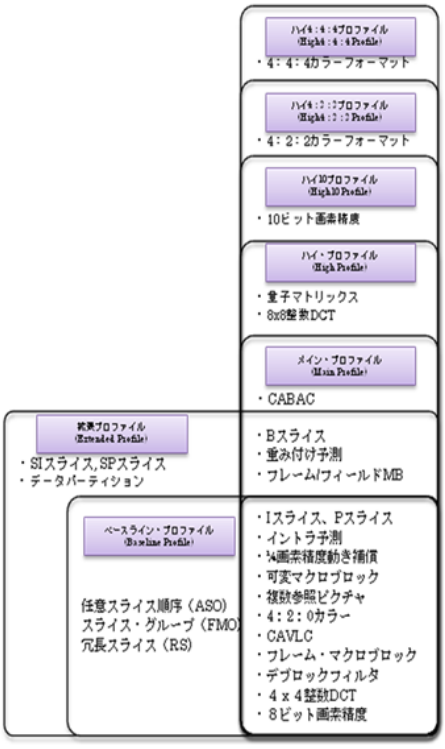


図２－２　フル規格版の機能図

図２－３　機能限定版の機能図

２－３－２　ソフトウェアの特徴

H.264 開発支援ツールのソフトウェアの特徴は、FPGA にデータ供給するための「CPU ソフトウェア」と、FPGA 上で稼働する「H.264 デコード（再生）ソフトウェア」であり、H.264 コード解析が容易なソフトウェア開発環境を実現した。特徴点を以下に記述する。

◆複雑な処理の流れをデータストリーム構造で分かりやすく提供（図２－４）

◆C 言語を VerilogHDL 言語に自動変換するためのノウハウを提供することで、C 言語での FPGA 開発を可能とした。

・ハード化のための専用記述（レジスタ、ストリーム、配列）

・C 言語での動作シミュレーション

・ソフトウェアとハードウェア間のデータ通信設計方法

・モジュール間の接続を視覚的に表示（図２－５）

◆T-Engine と FPGA の協調システム

・汎用的なシステム開発にも利用可能

◆フル規格版は、JM に準拠し全プロファイルに対応

◆機能限定版は、機能限定版で解析入門用

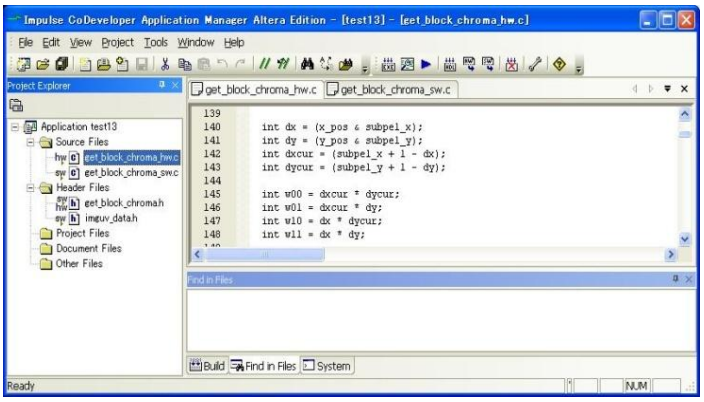


図２－４　Codeveloper での C 言語ソースコード

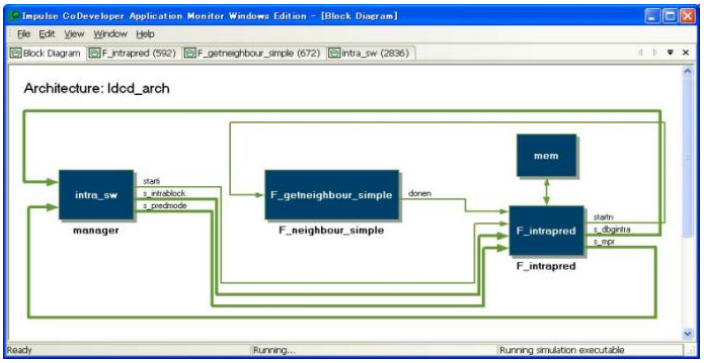
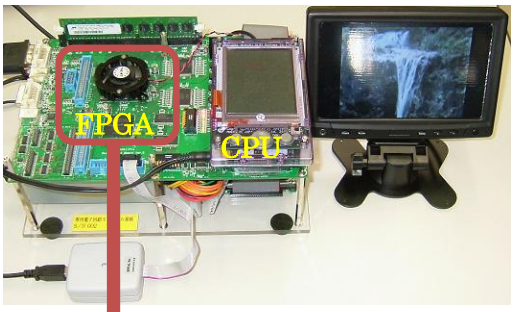


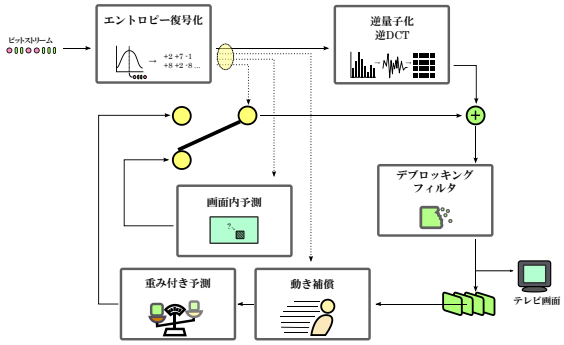
図２－５　モジュール間通信の視覚化

２－４　全体システム構成

本研究で試作したH.264用開発支援ツールは２－２のハードウェアに２－３のソフトウェアを組み込んで動作するシステム構成である（図２－６）。動作の流れは、ハードディスクに圧縮記録されたH.264規格ファイルをCPUが読込み、FPGAボードへ転送する。FPGAは転送された圧縮データをデコード（再生）プログラムによって元画像への復元を行いディスプレイに表示する。CPUボードは再生開始、停止、一時停止、繰り返し表示の命令がボタン、矢印キーで可能となっている。



FPGA開発支援ボード（ハードウェア）



H.264再生ソフトウェア

図２－６　全体機能概略図

２－５　H.264デコードの実験結果

H.264映像のデコード（再生）の実験では、機能限定版（NIOSⅡCPUのソフトウェア処理）と動き補償（輝度値）の一部をFPGA回路化した高速化版の比較実験を行った。実験に使用したH.264ファイルは独自に撮影しH.264でエンコードしたものである。表示速度の結果を表２－３に示す。また、この実験における画像表示の結果を図２－７に示す。

表２－３　表示実行結果

画像サイズ(共通)：320×240、NIOSⅡキャッシュ 64kb(MAX)

ファイル種類	全フレーム数	フレームレート(FPS) ソフト＋FPGAハード	フレームレート(FPS) ソフト処理のみ	高速化率
fall26	254	14.2	14.0	1.5%
fall31	281	12.1	11.1	9.0%
ETCrayfish	199	10.0	9.3	7.5%
fish38	315	10.2	9.0	13.0%
Monkey	89	9.4	8.7	8.0%

図２－７(a)　fall26 連続画像

図２－７(b)　fish38 連続画像

図２－７(c)　Monkey 連続画像

図２－７　表示結果画像（抜粋）

２－６　まとめ

H.264 デコーダのフル規格に準拠した JM ソフトウェアを FPGA で稼働する環境をフル規格版として実現した。これによって、低解像度（180×144）からハイビジョン（1920×1080）の高解像度まで､デコードの流れやデータ値を解析できる組込み開発環境の提供が可能となった｡ただし、JM コードは理解しやすい反面、無駄なコードや資源が使われており効率面では劣っている。そこで、効率化を目指し部分的にハード回路化することで、ソフト処理に対して 10%の高速化を達成することができた（高速処理版)。

３章　FPGA 実機速度計測

本章では H.264 デコードを FPGA 実機上で稼働させたときの各機能の処理速度について述べる。

３－１　目的

効率よくパフォーマンスを向上させるためには、処理時間のかかっている部分からハードウェア化をすすめることが必要である。ここでは、処理時間の計測を行い、FPGA ボード上での JM の各関数にかかる時間と、各処理時間の全処理時間に対する割合を求め、ハードウェア化の対象となる関数がいずれであるかということを判断するための材料を得ることを目的とする。

３－２　計測方法

FPGA にタイマーモジュールを搭載し、各関数の処理時間を 10nsec(1 億分の 1 秒)単位で計測した。

３－３　計測環境

・テストデータ

動画像圧縮の国際標準化機関の JVT(Joint Video Team)で公開されているテストシーケンス・ファイルを使用。

・ NIOSⅡ用ソフトウェア

速度計測のために、JM を改造し NIOS II IDE によりビルドしたものを使用。以下に改造箇所を示す。

- NALU_t 構造体を削除し、NALU 切り出しモジュールをハードウェア化。
- DPB 処理部分の効率化。
- T-Engine は未接続なので FPGA 上のバッファに H.264/AVC データを格納し、読み込む。

NIOS II IDE ビルド時最適化レベル・・・3(MAX)

NIOS II CPU クロック・・・・・・・・・100MHz

３－４　計測結果

計測結果から、主に処理時間の多かった関数を抜粋した計測結果を表３－１に示す。

/* 表の説明 */

※StartMB・・・・・・ImageParameters 構造体のマクロブロック・メンバの初期化

※readMB・・・・・・圧縮符号化されたマクロブロックデータを展開

※DecodeMB・・・・・展開されたマクロブロックのデータをデコード
（イントラ予測、動き補償）

※compute_colocated・・・・動きベクトルの情報などを扱う

※Deblock　・・・・・・デブロックピクチャ処理

※store_picture_in_dpb ・デコードピクチャを参照などの為に dpb に格納

表３－１　計測結果

	camp_mot_fld0_full	cvmp_mot_fld0_full_B	camp_mot_frm0_full	cvmp_mot_frm0_full_B
FILE SIZE	194	395	396	383
ENTOROPY	CABAC	CAVLC	CABAC	CAVLC
VIDEO FRAMES	30	30	30	30
REF-FRAME	IBBP	IBBP	IBBP	IBBP
FIELD/FRAME	FIELD	FIELD	FRAME	FRAME
RESOLUTION	720x480	720x480	720x480	720x480
StartMB (%)	1.904	2.201	1.924	2.332
ReadMB (%)	29.650	23.415	28.114	24.176
DecodeMB (%)	28.215	30.409	41.005	40.050
compute_colocated(%)	4.324	5.454	0.285	0.639
Deblock (%)	22.963	26.051	22.335	26.029
store_picture_in_dpb(%)	7.966	8.604	1.784	2.097

３－５　考察

・Deblock 処理は、Sharp_MP_PAFF_2 のようにエンコードパラメータによってデブロッキングフィルタをかけないと設定されたものは 0.158%となっており、ファイルによってかなりの差があることが分かった。

・ readMB 処理は大きいもので 54.462%(CANLMA2_Sony_C)、小さいものでも 13.784%(src21_420-1)と、かなりの割合を占めることがわかった。特に CABAC によって符号化されているファイルは割合、処理時間ともに大きくなる傾向にある。

・decodeMB 関数では全体的に想定していたよりも低い数字となったが、B フレームのあるファイルは割合、処理時間ともに大きくなる傾向にあることがわかった。

・startMB、compute_colocated はファイルによってばらつきがあり、割合、処理時間ともに上記の 3 つの処理よりもかなり小さい値なので、ハードウェア化の優先度としては低いと考えられる。

第４章　FPGA 回路モジュールの開発

本章では FPGA 開発支援ボードで稼働する FPGA モジュールの開発について述べる。

４－１　FPGA の全体モジュール構成

- FPGA の全体構成は、ALTERA 社から MegaFunction として提供されている NiosII CPU と、各種ペリフェラルコントローラ、及び H.264 のデコーダとしての機能を持つ新規に開発したモジュールで構成した。
- 各モジュールは、32bit／100MHz の Avalon bus で接続されている。
- 各モジュールの、Avalon-Master ポートは他のモジュールの Avalon-Slave ポートへと接続され、Avalon-Master ポートから Avalon-Slave ポートへライト／リードリクエストが発行される。また、Avalon-Slave ポートから Avalon-Master ポートへは、ウェイトリクエストを発行して Avalon-Master からのリクエストをウェイトできる構成となっている。
- H.264 のデコーダとして開発したモジュールは以下の通りである。
  - DAT：外部ホスト CPU と FPGA ボード間で、ホスト I/F を通して H.264 ストリームデータや、デコードデータの入出力を行う。
  - VGA:フレームメモリー上のデコード後の YUV データを RGB 変換して DVI ポートから出力する。
  - GBL：H.264 デコード処理の一部である動き補償（輝度）の予測処理を行う。

４－２　Verilog 言語による動き補償（輝度）の開発

（１）開発方法

H.264 デコーダ機能の中で、メモリアクセスが多く演算時間のかかっている動き補償の一部（輝度）について FPGA 回路モジュールを Verilog 言語で開発した。これを GBL モジュールと呼ぶ。

- GBL モジュールは、動き補償（輝度）処理の、６タップ・フィルタ計算部と、クリッピング処理部及び、そのブロックへのデータの入出力制御部より構成される（図４－１）。
- 輝度の画面内予測処理は、処理時に与えられるパラメータにより、全部で 10 種類の処理方法に分類される。
- GBL モジュールは、処理開始時、処理に必要なパラメータを任意のメモリ領域からリードし、そのパラメータにより 10 種類の処理の中から一つを自動で選択し実行する。

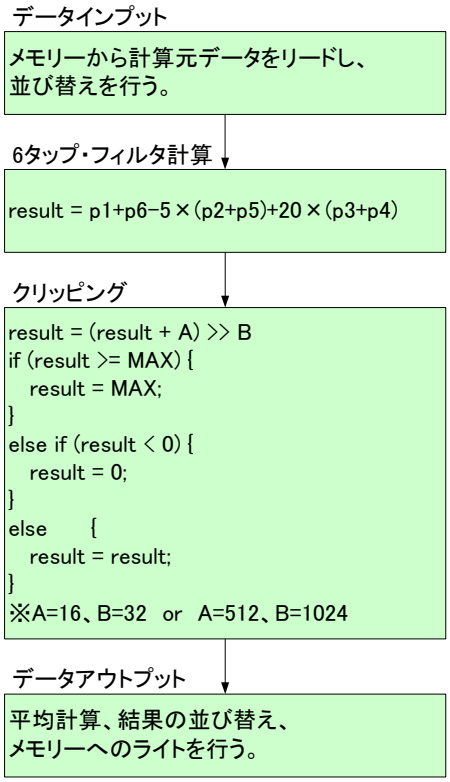


図４－１　GBL モジュールのフロー図

（２）実験結果

動き補償（輝度）の処理を GBL モジュールで行った時と、ソフトウェアで行った時のデコード時間を比較した。GBL モジュールで処理を行った時、ソフトウェアに比べて表４－１のように高速化した。測定条件は表４－２のとおりである。NiosII のデータキャッシュは none～64KByte の間で変更することが出来る。GBL は処理に必要なデータを外部メモリからリードするため、NiosII のデータキャッシュが none 以外の時、GBL を起動する前に、キャッシュの内容を外部メモリにライトする（キャッシュフラッシュ）必要がある。データキャッシュが none と、none 以外の時を比較すると、none 以外の時は、キャッシュフラッシュの時間が余計にかかるので、高速化の割合が小さくなる。

表４－１　高速化結果

NiosII Data cash	Software	Hardware	高速化の割合
none	0.464 (fps)	0.55 (fps)	19 (%)
64KByte	1.368 (fps)	1.473 (fps)	7.68 (%)

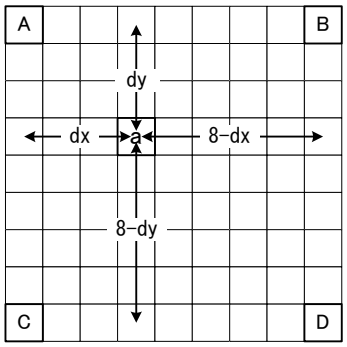
表４－２　測定条件

測定条件	
NiosII Instruction cash	64KByte
Program memory	SSRAM
FPGA 動作周波数	100Mhz
H.264 ファイル	Train 320x240

４－３　C 言語による動き補償（色差）の開発

（１）動き補償（色差）処理概要

JM における動き補償処理は､輝度に関する動き補償処理を行う get_block_luma 関数と色差に関する動き補償処理を行う get_block_chroma 関数の２つに分けられる。ここでは、get_block_chroma 関数について記述する。色差信号の予測補間画像値は、図４－２のように整数画素値（A、B、C、D）を１／８画素精度で線形補間して生成する。線形補間とは、元の信号からの距離に比例した係数を掛けて補完を行うことである。



□ A、B、C、D　　：整数画素
□ dy、dx　　：整数画素Aからの1/8画素単位の距離

□ 予測補間画像値：

a = [(8-dx)(8-dy)A+dx(8-dy)B+(8-dx)dyC+dxdyD]/64

図４－２　予測補間画像値の求め方

（２）iTransform 処理概要

iTransform 関数は、動き補償処理によって得られた予測補間信号（輝度信号 Y、および色差信号 U、V）と係数を用いて逆変換を行い、元の画像信号（以降、画素値）に復元し、適切な領域に画素値を書き込む。iTransform は、マクロブロック毎に条件判断を行い以下の３種類のいずれかの処理を行う。

①マクロブロックを４×４のブロックに区切りながら、予測補間信号（輝度信号 Y、または色差信号 U、または色差信号 V）と係数を用いて、４×４の逆変換を行い、画素値を適切な領域に書き込む

②マクロブロックを８×８のブロックに区切りながら、予測補間信号（輝度信号 Y、または色差信号 U、または色差信号 V）を用いて、８×８の逆変換を行い、画素値を適切な領域に書き込む

③逆変換は行わず、画素値を適切な領域に書き込む

また、色差信号のカラーフォーマットが YUV 4:0:0 および YUV 4:4:4 以外ならば、以下の３種類のいずれかの処理を行う。

④マクロブロックを４×４のブロックに区切りながら、色差信号（U、V）と係数を用いて、４×４の逆変換を行い、画素値を適切な領域に書き込む

⑤サブマクロブロック用４×４の逆変換を行い、画素値を適切な領域に書き込む

⑥逆変換は行わず、画素値を適切な領域に書き込む

（３）get_block_chroma 基本構成

動き補償（色差）関数である get_block_chroma を１つのハードウェアモジュールとし、図４－３に示す構成とした。

①get_block_chroma 構成

get_block_chroma_hw_func：get_block_chroma()関数をハードウェア化したモジュール。

get_block_chroma_sw_func：get_block_chroma_hw_func ハードウェアモジュールをコントロールするソフトウェアモジュール。

aaa_mem：参照データの領域（DDR2)。

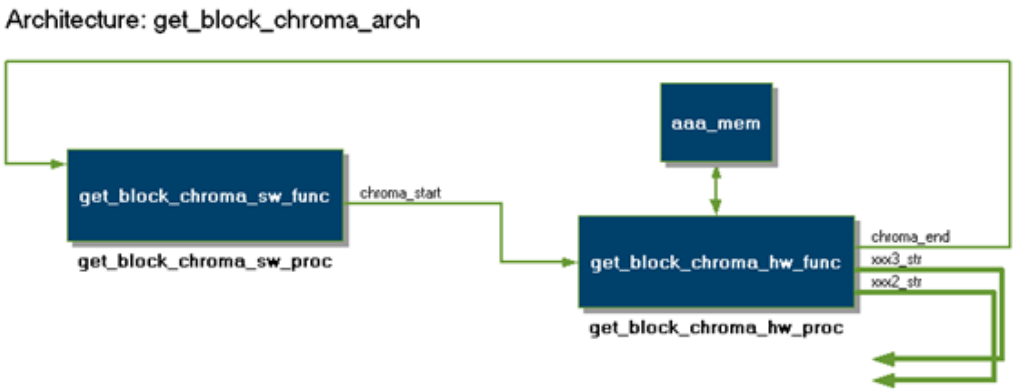


図４－３　get_block_chroma のモジュール構成図

（４）iTransformi 基本構成

iTransform 関数を１つのハードウェアモジュールとし、図４－４に示す構成とした。

①iTransformi 構成

iTransformiHwFunc：iTransform()関数をハードウェア化したモジュール。

iTransformiSwFunc:iTransformiHwFunc ハードウェアモジュールをコントロールするソフトウェアモジュール。

bbb_mem：参照データおよび画素値の書き込み領域（DDR2)。

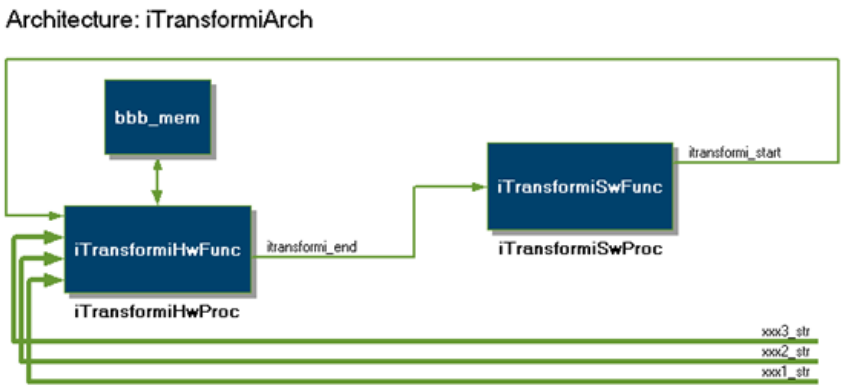


図４－４　iTransform のモジュール構成図

（５）実験結果

表４－３にH.264デコードの実行結果を示す。ソフトウェアのみのデコード（再生）結果と、get_block_chroma_hw_func＋iTransformiHwFuncのハードウェアモジュールを組み込んだデコード結果である。

表４－３　実験結果

ldecod構成	データサイズ	sec	fps
ソフトウェアのみ	320×240×30	17.954	1.671
ソフトウェア＋get_block_chroma_hw_func＋iTransformiHwFunc	320×240×30	17.848	1.681

（６）考察・まとめ

実行結果から、今回試みた構成で処理速度の向上が見受けられた。このことからハードウェア化の範囲を拡げることによって、全体の処理速度の向上が図れると考えられる。

但し、次の課題が残った。Quartus IIのCompilation Reportを見ると、Logic utilization が59%、DSP blockが60%と、FPGAの許容量の約６割を占めていた。他のハードウェアモジュールが入らなくなる可能性がある。さらにコードの見直しによる、コードの削減、効率の良いコードへの変更等を行わなければならない。また、処理速度の向上を行う上でもCoDeveloperに適したコーディング方法の技術開発が必要である。

第５章　C 言語記述による FPGA モジュール化

本章では動き補償（輝度）及びイントラ予測のハードウェア・モジュール化について述べる。

５－１　動き補償（輝度）の概要

動き補償処理は、get_block_luma（輝度）と get_block_chroma（色差）の２つに分けられる。ここでは、get_block_luma について述べる。get_block_luma 処理はマクロブロック単位で行われる。1/2 画素精度の予測画像を作成する際に６タップ・フィルタ処理を行う。このとき、マクロブロックサイズが１６×１６の場合、任意地点を起点とする１６×１６～２１×２１画素のデータが必要になる。６タップ・フィルタ処理は(A-5B+20C+20D-5E+F)/32 で表される積和演算であり、この処理を１マクロブロックにつき２５６～５９２回行う。また、1/4 画素精度の予測画像を作成する際にはさらに２５６回の 2 値の平均処理を行う。このように、get_block_luma 処理はメモリへの頻繁なランダムアクセスと、大量の計算処理が必要になるためこの部分をハードウェア化し高速化を図る。以下、get_block_luma 処理をハードウェア化の詳細について述べる。

５－２　Codeveloper のによる動き補償（輝度）のハード化

動き補償（輝度）の関数である get_block_luma()を１つのモジュールとしてハードウェアの開発を行った。ハードウェアの起動についてはシグナルインタフェースを用い、ソフトウェアから起動し、処理が終了したらソフトウェアに対して終了を知らせる。処理で必要になるデータの受け渡しは、主にレジスタインタフェースを用いるが、参照ピクチャの画素値についてはデータ量が多いためメモリインタフェースを用いる。計算結果についてはメモリインタフェースを用いて直接 DDR2 上に書き込む。図５－１に構成を示す。

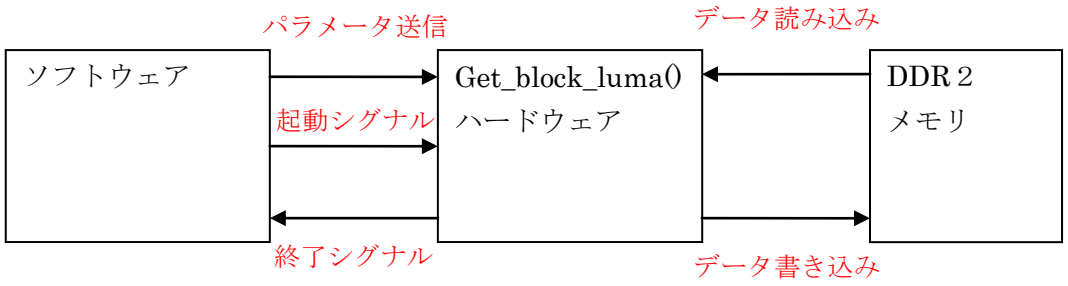


図５－１　get_block_luma ハードウェア化における基本構成

５－３　動き補償（輝度）モジュールの性能向上

５－２で示したハードウェアは、CoDeveloper が生成するメモリインタフェースを用いるためメモリアクセスに時間がかかり全体性能を低下させてしまった。また、get_block_luma ハードウェアの構成としても、分岐が 18 ありそれぞれに６タップ・フィルタなどの演算処理が記述されているため DSP を多く消費してしまうという問題がある。これらに関する改善を以下に示す。

（１）CDC モジュールの利用

CDC(CoDeveloper Control)モジュールは、DDR2 メモリと Codeveloper で作成した get_block_luma モジュール間で高速にアクセス（データ書込み／読込み）ができるように開発したモジュールである。CDC モジュールはアヴァロンバスに接続され get_block_luma ハードウ

ェアとはレジスタインタフェースとストリームインタフェースで接続される。これにより CDC モジュールは get_block_luma ハードウェアにおけるメモリインタフェースとレジスタインタフェースの役割を果たす。

（２）重複したデータ読み込みの回避

1/4 画素精度の計算を行う場合、参照するデータは 1/2 画素精度の計算を行う際に必要になるデータの中に確実に含まれている。そのため、2 回データを読み込むことをせず、CDC が 1 回だけデータを読み込み必要なデータについては再利用する。

また、unsafe position の計算の場合は、実際に参照する領域のデータを一括して読み込み、データを増やしてからストリームでデータを送る。モジュール構成を図５－２に示す。

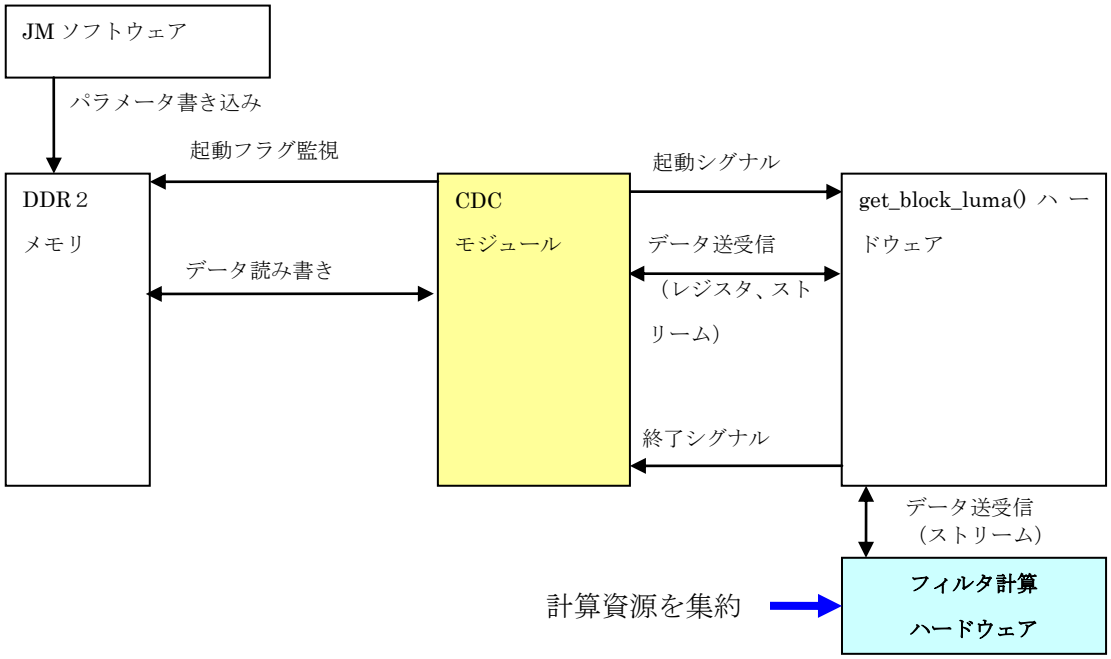


図５－２　CDC を用いた場合のモジュール構成

（３）実験結果

CDC モジュールを使用して、横方向（dy=0）の動き補償(輝度)計算の FPGA ハード化を行い、速度計測の比較実験を行った。この結果を表５－１に示す。ソフト処理に対して約 160 倍の高速化を実現した。これによって、メモリアクセスの高速化及びデータの流れをストリームとレジスタ・インタフェースで処理する CDC モジュールと、Codeveloper で作成した C 言語によるハード化モジュールによって、高速化手法の成果が得られた。

表５－１　速度計測の実験結果

試験条件	ソフト処理平均(μ s)	ハード処理平均(μ s)	ソフト/ハード
サイズ:320x240 周波数:100Mz 測定回数:261 回 データキャッシュ:無 動き方向:横のみ(dy=0)	984.5	5.8	168.7

５－４　イントラ予測のハードウェア・プログラム開発

（１）概要

H.264/AVC には、画像デコード処理の一つとして、イントラ(画面内)予測と呼ばれる画像データの複合処理が存在する。以下処理概要を記述する。

符号化されたデータから画像を生成する際､同一画面内にて予測対象となるマクロブロック(縦 16x 横 16 画素を 1 マクロブロックとする)の周辺にある予測済画素値を参照して新たに画像を生成する方法をイントラ予測という。予測画素算出の後、変換係数を参照して逆離散コサイン変換(IDCT)を行い、最終的な画素値を得る。

（２）開発方法

輝度値のイントラ予測では、1 回の処理で予測を行う画素の単位が 4x4、16x16、8x8 と 3 通り存在する。4x4、8x8 については９つのモードのいずれかで予測が行われ、16x16 では Vertical、Horizontal、DC の 3 モードに Plane を加えた全 4 モードで予測を行う。色差については、8x8 単位で予測を行い、予測モードは輝度 16x16 と同様の 4 モードで予測を行う。これらの各モードについて、Codeveloper 用 C 言語でコードを作成し、FPGA 上で動作を確認した。

（３）実験結果

今回ハード化（FPGA 回路化）を行ったイントラ予測処理範囲に対し、ソフトウェア上で同等の処理を行う範囲との処理時間の比較のため、1 ピクチャ分をデコード処理した際の両者の処理時間の比較を表５－２に示す。イントラ 16x16 では 1.11 倍、イントラ 4x4 で 8.33 倍、イントラ 8x8 とイントラ chroma では 9.09 倍と、すべてのモジュールで処理速度が上がる結果となった。

表５－２ イントラ予測処理ソフト・ハード処理時間比較

ハードモジュール名	ソフトウェア実行時間	ハードウェア実行時間	比較
イントラ 4x4 (107 回処理)	228.186 ミリ秒	27.565 ミリ秒	-200.621 ミリ秒 (12%、8.33 倍)
イントラ 16x16 (77 回処理)	9.700 ミリ秒	9.187 ミリ秒	-0.513 ミリ秒 (90%、1.11 倍)
イントラ 8x8 (116 回処理)	176.129 ミリ秒	19.422 ミリ秒	-156.707 ミリ秒 (11%、9.09 倍)
イントラ chroma(U/V) (300 回処理)	142.485 ミリ秒	16.171 ミリ秒	-126.314 ミリ秒 (11%、9.09 倍)

（４）イントラ予測ハードウェア・プログラムの考察

作成したハードウェア・プログラムの実行結果から、プログラムの容量が FPGA の許容量の約 8 割を消費するため、他のハードプログラムが入らないという課題が挙げられた。今後、容量を節約するコーディング方法の実施、冗長なロジックの見直し等を行う予定である。さらに、ハードモジュール内のメモリアクセス方法の変更、高速化のためのロジック集約を行うことで、更なる性能向上が期待できる。

５－５　まとめ

（１）動き補償（輝度）のハードウェア化（C 言語による FPGA 実装）

Codeveloper を使って、C 言語による動き補償（輝度）の FPGA ハード化を行ったが、処理速度が低下してしまった。これは Codeveloper のメモリアクセス関数に原因があった。そこでメモリアクセス部を新規に開発し、ソフトウェア処理に対し約 160 倍の高速化を実現した。

（２）イントラ予測のハードウェア化（C 言語による FPGA 実装）

Codeveloper を使って、C 言語によるイントラ予測の FPGA ハード化を行った。この結果、ソフト処理に対し約 9 倍の速度向上を実現した。

第６章　全体総括

本研究では、「FPGA 開発支援ボード（ハードウェア）」と、このボード上で稼働する「専用ソフトウェア」から構成される「H.264 用開発支援ツール」を開発した。「FPGA 開発支援ボード」は、異なる 2 種類の FPGA を搭載し、並行動作や協調動作の開発を可能とした。また画像を取り扱うための大容量メモリ 512MB や、周辺インタフェースに DVI や USB 等を装備し、開発自由度の高いシステムとした。H.264 デコード（再生）の「専用ソフトウェア」は、フル規格版、機能限定版、高速処理版の３つの解析ツールとして実現した。これらの試作品を製品化するための取り組みとして、とうほく自動車関連技術展示商談会（H20 年 11 月 17～18 日、刈谷市）と組込み総合技術展 ET2008（H20 年 11 月 19～21 日、横浜市）への出展及び、県内・県外の企業訪問による評価（H21 年 2～3 月）を行った。その結果、C 言語の資産を FPGA で生かしたいとの要望が強く、「C 言語からハード言語への自動変換技術」が最も注目され、本ツールの最大の強みとなっていることが分かった。ただし、C 言語をそのままハード言語（HDL）へ変換できるわけではなく、ハード言語化用の C 言語記述方法が必要となる。本研究ではこの C 言語記述方法を開発し H.264 デコード・ソフトウェアとして実現した。

一方で、ハードウェアによる高速化が部分的であるため、ユーザからは処理全体の速度向上の要望が挙げられた。今後は、製品化の段階で必要となるデコード処理の高速化を達成するために、デブロッキングフィルタなどの FPGA ハード回路化が未達成な部分についてのハード化を進めていく予定である。