C言語からの設計システムBachを用いたMPEG-4 IPコアの開発

MPEG-4 IP Core Development Using C-Based Synthesis System, Bach

山田晃久*1　櫻井涼二*1　山口雅之*1　神戸尚志*1
Akihisa Yamada　Ryoji Sakurai　Masayuki Yamaguchi　Takashi Kambe

堅田裕之*2　Andrew Kay*3　Paul Boca*3　Vince Zammit*3
Hiroyuki Katata

要　旨

低ビットレート向けの動画圧縮方式であるMPEG-4は，次世代移動通信システム（IMT-2000）やディジタルラジオ放送に採用されるなど，今後幅広く利用されると見込まれている。MPEG-4ビデオコーデックをIPコア化できれば，携帯電話，動画カメラなどに搭載されるシステムLSIの開発期間を短縮できる。しかし，MPEG-4の規格は自由度が大きいため，IPコアは用途に応じて柔軟に変更できる必要がある。このような要求を満たすために，C言語を用いた回路合成システムBachを用いて，アプリケーションに応じて容易にカスタマイズできるMPEG-4ビデオコーデックIPコアを開発した。本稿では，本設計で用いた回路合成システムの概要を紹介し，本システムを用いたMPEG-4の設計手順について述べる。

MPEG-4 is a standard of video coding method for low bit rate applications, which is expected to be widely used in next generation mobile communication system, IMT-2000, and digital radio broadcasting. If the MPEG-4 video codec is made as an IP core, we should be able to accelerate development of system LSI for mobile phones, video cameras, etc. It is also essential that the core is flexible enough, because the MPEG-4 standard allows various customizations depending on parameters of applications such as transfer rate and image size. Using C-based synthesis system, Bach, we have developed an MPEG-4 video codec IP core that meets these requirements. In this report, we summarize the Bach system and describe the design flow of how the MPEG-4 core has been designed using Bach.

*1 IC開発本部　設計技術開発研究所　第1開発室
*2 情報家電開発本部　マルチメディア開発研究所　NB1PT
*3 Sharp Laboratries of Europe, Ltd.

まえがき

インターネットが普及し，電子メールやWWW上で画像データが頻繁にやりとりされるようになってきた。このような中，動画の圧縮方式として注目されている技術としてMPEG-4（Moving Picture Experts Group Phase 4）がある。MPEG-4は，ISO／IEC JTC1／SC29／WG11という国際標準化組織で標準化を行なっている移動体通信やインターネットに向けた高能率のデータ圧縮方式であり，1999年9月にバージョン1が標準化された。このMPEG-4は，次世代移動通信システムIMT-2000やディジタルラジオ放送の動画圧縮方式として採用されるなど，今後移動体通信やインターネット関連機器で広く使われることになると見られている[1)]。このような商品をターゲットとして，各社はLSI開発を進めているが，バージョン1に続いてバージョン2の標準化が進められており，規格変更，追加の可能性があること，MPEG-4の規格そのものに自由度が大きいことなどから，多くの場合DSPとソフトウェアにより実現されている。しかしながら主な応用商品である携帯機器への搭載を考えた場合には，低消費電力化することが望ましく，演算量の多い部分は布線論理（ハードウェア）として実現することにより動作周波数を下げるような方策が必要となる。

我々はこの要求に応えるために，C言語ベースの設計システムBach[2)]を用いて，カスタム化が容易なMPEG-4（シンプルプロファイル）IPコアを開発した。Bachはシャープヨーロッパ研究所とIC開発本部が共同で開発した設計システムで，C言語をもとに回路の動作を記述しやすいように拡張したBach-Cで書かれ

たアルゴリズム記述から，RT（レジスタ転送）レベルの回路を自動で合成することができる。従って，従来のHDL（ハードウェア記述言語）を使った場合に比べ設計期間を大幅に短縮できるだけでなく，設計したIP コアを再利用する場合に，仕様変更や機能追加などもアルゴリズム記述を変更すればよいため短期間で行うことができる。

本稿では，まずC 言語ベースの設計システム Bach の概要を説明し，このツールを用いたMPEG-4 IPコアの設計について述べる。

１．Bachシステムの概要

１･１　Bach-C言語

我々は，アルゴリズムから回路を自動合成するBachシステムの入力言語としてANSI C言語を選択した。これは，社内のアルゴリズム開発者の多くが利用しており，設計資産も活用できると考えたからである。ただし，C言語の枠内だけでハードウェアのアルゴリズムを記述しようとすると記述が繁雑になってしまい，設計者にとっては記述しづらくなってしまうため，直観的に記述できるように文法を拡張した。

拡張した点は，

（１）任意のビット幅演算を実現するためのデータ型

（２）ハードウェアの並列動作を明示するためのpar構文

（３）並列に動作する処理間の通信

である[2)]。**図１**に Bach-C 記述の例を示す。

（１）任意ビット幅のデータ型

Bach-C 言語では，int（signed）型，unsigned 型において，ビット幅を指定することができる。設計者は演算精度を確保するのに必要なビット幅を明示することにより，冗長のない回路を生成することができる。例えば，**図１**中 14 行目では変数 i が４ビット幅の unsigned 型として宣言され，15 行目では変数 x が８ビット幅の int 型として宣言されている。ビット幅を明示していない場合は，デフォルトのビット幅とみなされる。なお，デフォルトのビット幅は設定ファイルにより，任意の自然数に設定可能である。

（２）並列構文

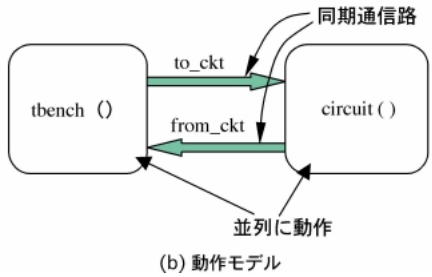
並列に動作するハードウェアを明示的に記述できるように，Bach-C 言語には並列構文（par）が追加されている。**図１**の５－８行目では，関数circuit()とtbench() が並列に動作することが指定されている。

（３）通信

Bach-C 言語では，並列に動作する処理間でデータをやり取りするための通信を記述することができる。

```
1  void main(void)
2  {
3    chan int #8 to_ckt, form_ckt;
4
5    par {
6      circuit(to_ckt, from_ckt);
7      tbench(to_ckt, from_ckt);
8    }
9  }
10
11 void circuit (chan int#8 to_ckt,
12                 chan int#8 from_ckt)
13 {
14   unsigned #4 i;
15   int #8 x;
16
17   for (i = 0; i < 19; i++) {
18     x = receive(to_ckt);
19     send(from_ckt, i * x);
20   }
21 }
22
23 void tbench(chan int #8 to_ckt,
24             chan int #8 from_ckt)
25 {
26   unsigned #4 i;
27
28   for (i = 0; i < 10; i++) {
29     send(to_ckt, i);
30     putint(stdout, 10, 0, receive(to_ckt));
31   }
32 }
```

(a) Bach-C記述



(b) 動作モデル

図１　Bach-C 記述の例

Fig. 1　An example of Bach-C description.

図１の18行目と29行目では並列に動作するtbench ()からcircuit（）へのデータ転送が記述されており，19行目と 30 行目ではその逆方向のデータ転送が記述されている。

データ転送に用いられる通信路には同期通信路と非同期通信路がある。同期通信路では，送信側と受信側の双方が準備できるのを待ってデータ転送が行なわれ，データ転送が完了するまで次の処理は行なわれない。一方，非同期通信路では，通信相手が準備できていなくても送信，または受信が実行され，ただちに次の処理が行なわれる。**図１**の３行目では，同期通信路（chan型）としてto_cktとfrom_cktが宣言されている。

１･２　ツール構成

Bach はツールとして，コンパイラ，会話型デバッガ，コンパイル型シミュレータ，プロファイラを持ち，これらのツールの起動，結果の解析等はGUIベースで行える（**図２**）。

（１）コンパイラ

コンパイラはハードウェアの動作を記述したBach-

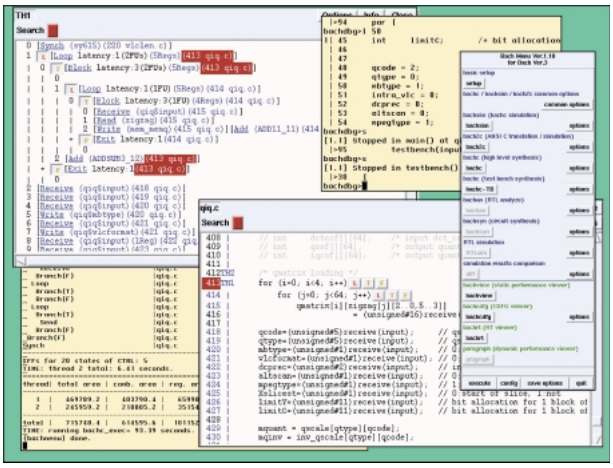


図2　Bachシステムの GUI
Fig. 2　GUI of the Bach system.

C記述を入力し，動作合成を行う。動作合成では，記述中のデータの流れや制御の流れを解析して動作の並列性を抽出し，演算の実行順序や回路中に使用される演算器，レジスタの個数を最適化して，RTレベルの回路を生成する[3]。出力は論理合成可能なRTレベルの記述である。

このコンパイラでは並列構文の各分岐は1つの階層となるように合成する[4]。動作合成では記述中の並列動作を自動で抽出できるが，回路が大規模になった場合に，合成した回路が階層的な構造になっていた方がFPGAへの回路の分割等が行い易いため，設計者が意図的に全体の回路を分割できるようになっている。

（2）シミュレータ／デバッガ

デバッガはBach-C記述上でブレークポイントを設定したり，ステップ実行することが可能で，並列動作を含む回路動作を会話的にデバッグすることができる。コンパイル型シミュレータは，Bach-CをANSI Cに変換し，Cコンパイラで実行モジュールを生成するため，並列動作を含む回路動作を高速に検証することが可能である。

（3）プロファイラ

プロファイラはシミュレーション中に記述中の各構文が実行される回数をカウントする。実行回数が多い箇所を改善する場合などに必要な情報を得ることができる。

2．C言語を用いたMPEG-4 IPコアの設計

今回の設計では，まず計算機上で動作検証が済んだC言語記述からはじめて**図3**のような工程で設計を行った。本節では，特にハードウェア設計に着目しMPEG-4 IPコアの設計手順について述べる。

Bachシステムを用いた設計では，回路（ハードウェア）のアルゴリズムをBach-C言語で記述すれば，自動でRTレベルの回路が合成され，VHDL記述が出力される。以降の設計工程は従来の設計工程と同じである。以下では，

（1）　アーキテクチャ設計（ハードウェア／ソフトウェア分割）
（2）　ハードウェアアルゴリズム設計
（3）　Bach-C記述の作成
（4）　Bachコンパイラによる合成

について詳しく述べる。

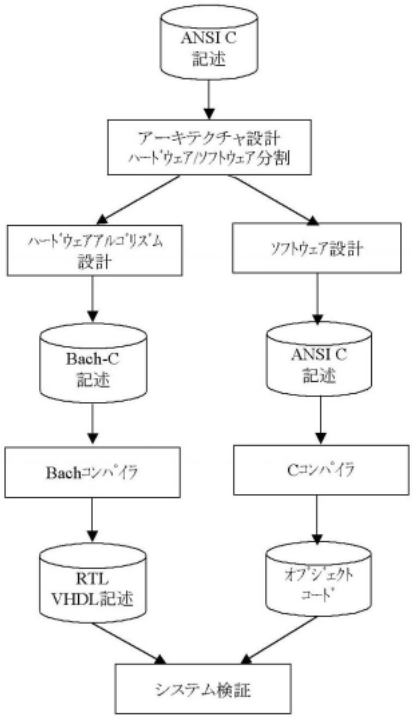


図3　Bachシステムを用いた設計フロー
Fig. 3　Design flow using the Bach system.

2・1　アーキテクチャ設計（ハードウェア／ソフトウェア分割）

MPEG-4のエンコード処理はデコード処理よりも計算量が多く，又エンコード時にデコード処理も行っていることから，エンコード処理を解析してアーキテクチャを決定した。

エンコード処理は，16×16の画素からなるマクロブロック（MB）を単位として，（a）現画像／参照画像入力，（b）動きベクトル検出，（c）差分画像生成，（d）DCT（離散コサイン変換），（e）量子化，（f）逆量子化，（g）IDCT（逆離散コサイン変換），（h）参照画像生成，（i）DC/AC予測，（j）ジグザグスキャン，（k）VLC（可変長符号化），（l）ビットストリーム生成の処理からなっている（**図4**）。さらに応用商品によって

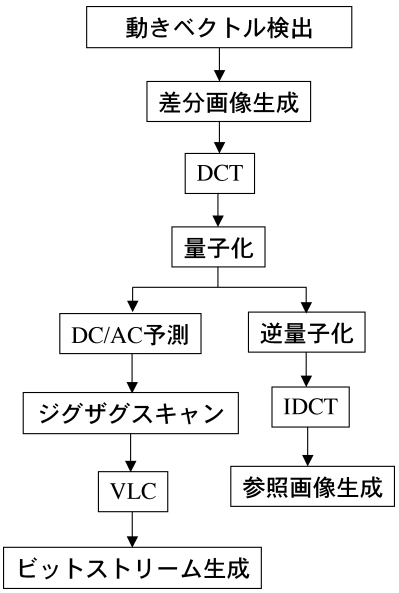


図4　MB 内のエンコード処理
Fig. 4　Encoding flow for MB.

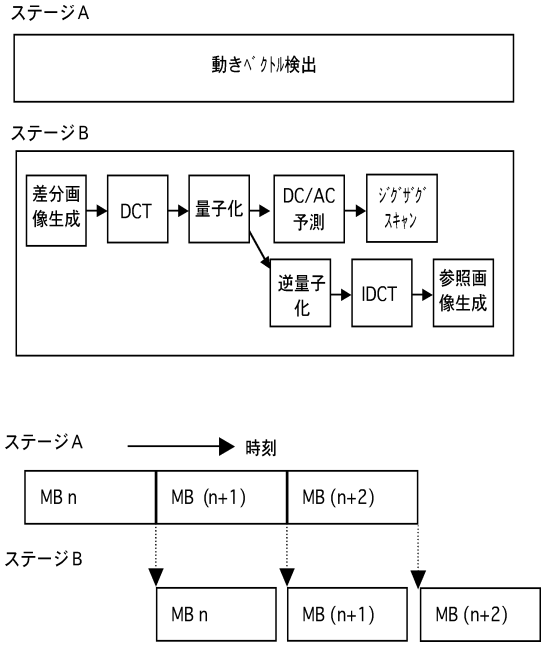


図5　パイプラインアーキテクチャ
Fig. 5　Pipeline architecture.

要求が異なるエラー耐性やレート制御などがある[5]。これらの処理を, 規格変更や仕様変更に柔軟にし, かつ低消費電力化のために動作周波数を低く抑えることを重視し，CPU によるソフトウェア処理と布線論理によるハードウェア処理を組み合わせて実現することにした。

ビットストリーム生成は, 計算量も少なく今後拡張される可能性もあるため, ソフトウェアによる処理が必要と考えた。また, レート制御, エラー耐性は応用商品毎に異なる処理になる可能性が高く, エラー耐性処理と深く関連する VLC も合わせてソフトウェア処理とした。一方処理データ量を考慮すると, 大量の画素データを処理する動きベクトル検出, DCT, 量子化などはハードウェアでの処理が必要なため, (a) から (h) の処理はハードウェア処理とした。さらにハードウェア処理とソフトウェア処理間でのデータ転送量を検討すると，(j) のジグザグスキャンまでをハードウェアで行い, ジグザグスキャン後のデータをソフト処理側へ転送することで, データ転送量を抑えられることが分かった。

このようにして，MPEG-4 アルゴリズムをハードウェア／ソフトウェア処理に分割し, 高性能かつ規格変更に柔軟な IP コアとした。

2・2　ハードウェアアルゴリズム設計

今回の IP コアは，アルゴリズムの機能を確認するために開発されたC記述をベースに開発した。この記述を直接 Bach-C に書き直しただけでは大域的な並列化が得られず, 必ずしも性能の高い回路を得ることができない。そこで Bach-C 記述を作成する前にハードウェアアルゴリズム設計を行った。

ハードウェアの基本アーキテクチャとしては, 動作の並列度を上げるためにMB毎に各処理をパイプライン的に処理するパイプラインアーキテクチャとした。パイプライン段数, 各パイプラインステージの内容を決定するために, 各処理のステップ数を見積もり, 各処理間でのデータ依存, 処理依存関係を抽出したところ, 概算見積で動きベクトル検出のステップ数と, その他の処理のステップ数の和が同程度であった。そこで（A）動きベクトル検出,（B）差分画像生成～ジグザグスキャン, 参照画像生成の 2 ステージ構成とした（**図5**）。ステージ B では DC/AC 予測～ジグザグスキャンの処理と，逆量子化～参照画像生成の処理はデータ依存関係および処理依存関係がないため並列に実行することにした。このようにパイプラインステージに分割することで，各 MB の処理は概算で 4000～6000 サイクルのスループットで処理できるようになる。つまり QCIF 15 フレーム／秒の動画であれば，約 9MHz の動作周波数でエンコード処理が可能である。

2・3　Bach-C記述の作成

2・2で検討したハードウェアアルゴリズムに基づき，元の C 記述を Bach-C 記述へと変更していく。主な変更点としては,

（1）各変数のビット幅を指定する。
（2）繰り返し処理を効率よく実行できるように記

述する。

（3）並列に実行する処理を並列構文（par）を用いて記述する

である。

各変数のビット幅は，Bach-C記述をシミュレータで検証しながら演算精度を確保できる範囲最小になるように決定していった。繰り返し処理は，演算器，レジスタ数，スループットに大きな影響を与えるため，Bachコンパイラで合成を行い，必要なステップ数，回路規模（演算器数，レジスタ数）を確認しながら記述を改善した。

```
 1 chan to_stage_B;
 2
 3 par {
 4   while (1) { // Stage A
 5     Motion_Prediction( ) ; // 動きベクトル検出
 6     send(to_stage_B, 1) ;
 7   }
 8   while (1) { // Stage B
 9     recieve(to_stage_B) ;
10     MC( ) ;    // 差分画像生成
11     DCT( ) ;    // DCT
12     Quant( );   //量子化
13     par {
14       {
15         DC_AC( );  // DC/AC予測
16         ZIGZAG( ); // ジグザグスキャン
17       }
18       {
19         DeQuant( );  /逆量子化
20        IDCT( );     // IDCT
21        MKREF( );    //参照画像生成
22       }
23     }
24   }
25 }
```

図6　MPEG-4のBach-C記述

Fig. 6　Bach-C description for MPEG-4.

図5のパイプライン処理は**図6**のような記述になる。ステージAでは動きベクトル検出が終わるとステージBへの起動信号を同期通信路to_stage_Bを通じて送り，ステージBがこの起動信号を受け取った後に次のMBの動きベクトル検出の処理に進む。ステージBではステージAからの起動信号を待ち，受け取った後にステージ内の処理が実行される。13行目のparは，ステージB内でDC/AC予測～ジグザグスキャンと逆量子化～参照画像生成が並列に実行されることを意味する。

2・4　Bachコンパイラによる合成

動作周波数とセルライブラリを指定して合成する。コンパイラは，セルライブラリから遅延，面積情報を算出するので，所望の動作周波数で動作するRTレベルの回路を生成する。合成後に各処理に必要なサイクル数や面積の見積値も出力されるため，これらの情報を元に記述の改善を行ない，回路の最適化を行った。

むすび

Bach-Cの検証は，RTレベルの検証と比べて100倍程度高速であり，短時間でビットアキュレートな検証を行なえる。今回の設計の場合，QCIFサイズの画像100フレーム分をRTレベルで検証しようとすると約1日かかってしまうが，Bach-Cでの検証時間は約3分で完了する。したがって，Bach-Cのレベルで確実に機能を検証した後に合成をすることにより，手戻りのない回路設計が行えた。

今回設計したMPEG-4 IPコアは，低消費電力化を図るために計算量の多い部分を布線論理化することにより，DSPを用いてソフトウェアだけで処理した場合に比べ約1/8の動作周波数で同程度の性能を得ることができた。この時，Bach-C言語を用いて布線論理の部分を設計することにより，従来のRTレベルからの設計に比べ容易に拡張や変更ができるようになっている。例えばデコーダ機能が不要な場合に，エンコーダ機能のみを実現することもソフトウェア部だけでなく，ハードウェア部もCレベルの変更だけで済む。

今後は，このIPコアを用いて応用商品向けLSIの開発を行うと共に，今回有効性が確認されたBachシステムを用いて新規IPコアの開発を行っていく。

謝辞

本開発を進めるにあたり，ご指導ならびにご協力頂いた情報家電開発本部マルチメディア開発研究所草尾寛チーフ，伊藤典男主任，内海端主任に深く感謝致します。また，今回使用したBachシステムの開発に際しご討論頂いたシャープヨーロッパ研究所の野村俊夫チーフ，ならびにBachシステムの開発に従事された設計技術開発研究所第1開発室の関係各位に深く感謝致します。

参考文献

1)　加藤，堀切，“MPEG4で動画メディアが大増殖”，日経エレクトロニクス11月1日号，no. 756，pp. 99-117，日経BP社（1999）.

2)　A. Yamada, K. Nishida, R. Sakurai, A. Kay, T. Nomura, T. Kambe,“ Hardware synthesis with the Bach system ”, in Proc. of IEEE ISCAS 99, Vol. VI, pp. 366-369（1999）.

3)　西田，岡田，大西，A. Kay, P. Boca，山田，神戸，“ハードウェアコンパイラBachの動作合成系”，情報処理学会DAシンポジウム99，pp. 95-100（1999）.

4)　高橋，石浦，山田，神戸，“ハードウェアコンパイラBachにおけるスレッド分割手法”，電子情報通信学会第12回回路とシステム（軽井沢）ワークショップ，pp. 103-108（1999）.

5)　三木，“MPEG-4のすべて”，工業調査会（1998）.

（2000年9月21日受理）