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

2005－SLDM－118（1）
2005／1／25

ALU間接続を制限したALUアレイによるリコンフィギュラブルプロセッサの検討

岡田 誠† 平松 達夫† 中島 洋† 小曽根 真† 平瀬 勝典† 木村 晋二‡

† 三洋電機株式会社 デジタルシステム技術開発センター 〒574-8534 大阪府大東市三洋町 1-1
‡ 早稲田大学大学院 情報生産システム研究科 〒808-0135 福岡県北九州市ひびきの 2-7

E-mail: † {m_okada, t_hiramatsu, hi_nakajima, m_ozone, k_hirase}@rd.sanyo.co.jp, ‡ shinji_kimura@waseda.jp

あらまし 本稿では，小型で処理効率の良いALUアレイベースのダイナミックリコンフィギュラブルプロセッサを提案する．ALU間の接続に制限を加えることで，接続部分の回路規模を約63%に削減できることを示すと同時に，独自開発したデータフローのマッピング手法を用いることにより，接続制限を行わないアーキテクチャと同程度の処理性能を達成できることを示す．本アーキテクチャの有用性を示すため，MPEG-4デコード処理を本アーキテクチャにマッピングして評価を行った．ASIC回路との比較を行った結果，同等の性能を約2.4倍の回路規模で達成できることを示している．これはリコンフィギュラブルプロセッサの汎用性を考えると適当であると考えられる．

キーワード リコンフィギュラブルプロセッサ，ALUアレイ，データフローグラフ

A Reconfigurable Processor Based on ALU Array Architecture with Limitation on the Interconnection

Makoto OKADA† Tatsuo HIRAMATSU† Hiroshi NAKAJIMA† Makoto OZONE†
Katsunori HIRASE† and Shinji KIMURA‡

† Digital Systems Development Center, SANYO Electric Co., Ltd 1-1 Sanyo-cho, Daito-shi, Osaka, 574-8534 Japan
‡ Graduate School of Information, Production and Systems, Waseda University 2-7 Hibikino, Kitakyusyu-shi, Fukuoka, 808-0135 Japan

E-mail: † {m_okada, t_hiramatsu, hi_nakajima, m_ozone, k_hirase}@rd.sanyo.co.jp, ‡ shinji_kimura@waseda.jp

Abstract Dynamic reconfigurable processor based on ALU array architecture for consumer appliances is introduced. We propose the ALU array architecture with the limitation on the interconnection for area reduction. With the proposed architecture, we can reduce gate size by 63% on interconnections. In addition, we show that the performance of the proposed architecture is almost the same as one without limitations. The proposed processor is a parallel processing processor that consists of a sequencer and an ALU array, adopted multi threading technology. We develop compilation tools from source codes written in C language for the proposed processor. We demonstrate the software model of the processor using MPEG-4 video decoding application.

Keyword Reconfigurable Processor, ALU array, Data Flow Graph

1. はじめに

近年，LSIの集積度の向上が進んでいる．集積度の向上により，大規模回路の実現が可能となり，機器の高機能化，小型化をもたらしたが，それと共にLSIの初期回路開発のコストも非常に高価となった．また様々な規格や仕様に対応する必要から，機器の寿命も短くなってきており，短期間でタイミングの良いLSI開発が求められている．その一方で，性能に対する要求は強く，高速処理が要求されるアプリケーションに対して，マイクロプロセッサやDSPでは性能要求にこたえることは現状では困難である．リコンフィギュラブルデバイスは，これらの課題を解決可能な新しいアプローチを提供するデバイスとして注目されている．

従来から書き換え可能なデバイスとして，FPGAとDSPが存在する．FPGAは，図1のように，1-4ビットのロジックエレメントとそれらを接続する配線スイッ

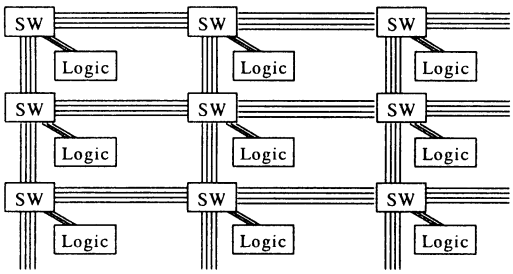


図 1 FPGA の構成

チから構成される．ところが，FPGA 上でシステムを実装した場合，ASIC で実装した場合に比べてかなり実装サイズが大きい．FPGA のリソースのうち，80-90%がスイッチと配線で占められ，残りの領域の多くは，コンフィギュレーション用メモリとして使われている．実際に演算を行うロジックに対する，この膨大な配線リソースが ASIC に対する面積増大の原因と考えられる．一方，DSP においては，1 つないし，数個の ALU で処理を実行するため，ASIC で要求される性能を満足することはできない．従って，FPGA よりも小型で，DSP よりも処理性能の良いリコンフィギュラブルデバイスが求められる．

我々は，ALU アレイアーキテクチャと呼ばれるアーキテクチャをベースに，より小型で処理性能の高いリコンフィギュラブルプロセッサを提案する．本リコンフィギュラブルプロセッサは，省面積のため配線数を制限した ALU アレイ，シーケンサ，及びマルチスレッドで構成されている．本プロセッサでは，柔軟に再構成が可能であるので，様々な機能変更を，処理を停止することなく，実現することができ，更に回路規模は従来のリコンフィギュラブルデバイスに比べて小さい．

本稿では，本プロセッサのアーキテクチャ，及び本プロセッサ上で MPEG-4 デコード処理を実現した場合の設計について述べる．また ALU アレイの動作をシミュレーションするために開発したソフトウェアモデルと，ASIC に対する回路規模比較を示す．

2. アーキテクチャモデル

2.1. 提案するアーキテクチャ

図 2 に我々が提案するリコンフィギュラブルプロセッサのアーキテクチャを示す．図のように，本アーキテクチャでは，シーケンサ，ALU アレイ，コマンドメモリと呼ばれるコマンドデータ格納用メモリ，及び出力メモリから構成される．ALU アレイは，ALU を縦 3 段，横 6 列に配置した構成を採用している．

処理実行時，シーケンサはコマンドメモリに対して，ALU アレイに割り当てるコマンドデータの格納先のアドレスを送信し，指定されたコマンドデータが ALU アレイに割り当てられた後，ALU アレイは処理を開始する．この割り当て処理は数クロックで終了する．ALU アレイは実行中に，シーケンサコマンドと呼ばれるデータを送信する．シーケンサはそのデータにより，次に ALU アレイに対して割り当てるコマンドデータのアドレスを決定する．この詳細については，次の 2.2 において述べる．

コマンドデータには，ALU に割り当てる命令種別，ALU 間の接続情報，入出力データの格納アドレスが示されている．

2.2. シーケンサ

シーケンサは， ALU アレイに割り当てるコマンドデータを決定するための重要な機能である．シーケンサのモデル図を図 3 に示す．シーケンサは，シーケンサリストと呼ばれるコマンドデータ格納アドレスのリストとアドレス演算器から構成される．シーケンサリストにおいて，シーケンサアドレスはシーケンサリスト内の位置を示すアドレスであり，コマンドアドレス

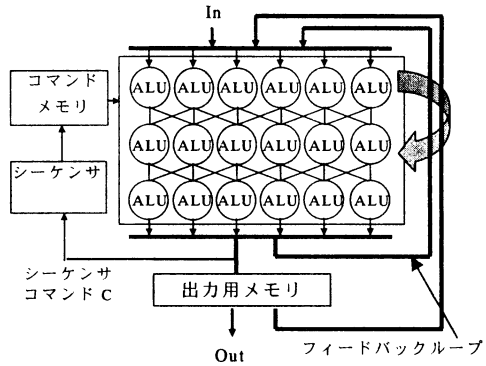


図 2 ALU アレイアーキテクチャ

はコマンドメモリ内に格納されたコマンドデータの先

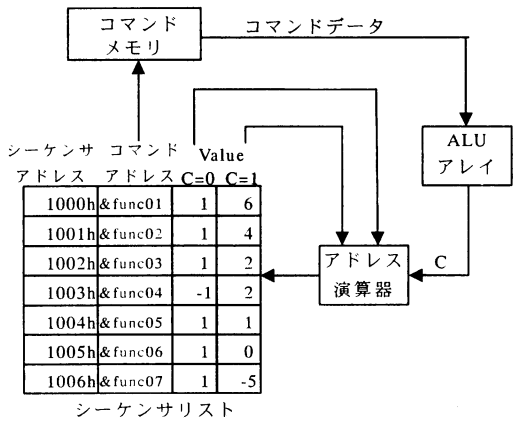


図 3 シーケンサモデル

頭位置を示すアドレスである．Value は，現在のシーケンサアドレスに加えられる値で，もし ALU アレイで演算されたシーケンサコマンド C が 0 である場合，"C=0"列の値が加算され，C が 1 である場合，"C=1" 列の値が加算される．例えば，現在 ALU アレイに割り当てられているコマンドデータが&func01 で示されるアドレスのデータで，ALU アレイがシーケンサコマンド C が 1 を出力した場合，1000h に 6 が加算され，1006h となるので，コマンドアドレス&func07 のコマンドデータが次に ALU アレイに割り当てられる．もし Value が負か 0 であるならば，ループ処理実行となる．FPGA 上でループ処理を実装する場合，展開が必要であるが，本アーキテクチャではシーケンサを使うことでコマンドメモリを削減することができる．

本アーキテクチャでは，条件分岐に対して 2 つの処理方法を備えている．図 4 に，その処理分岐方法を示す．1 つは ALU アレイ上に直接条件分岐命令を割り当てる方法で，もう 1 つは上述したシーケンサで処理する方法である．2 つの方法が利用可能である利点として，図 4(a)のように，処理対象が比較的小さい場合，分岐ノードを用いることでシーケンサ処理のオーバヘッドを減らすことができる．また図 4(b)のように，処理対象が比較的大きい場合，分岐で選択されなかった側の処理を実行する必要がないため，無駄を省くことができる．

2.3. ALU アレイ

本アーキテクチャでは，18 個の ALU，上段の ALU からデータを選択するためのスイッチ，フィードバックループを持つ ALU アレイを採用した．それぞれの ALU は独立して動作するので，18 命令が同時実行可能である．ALU は，通常の算術命令に加えて条件判定，データ選択，1，2，4，8 ビットの算術シフト，及び乗算を展開処理するための専用命令を備えている．本アーキテクチャの ALU アレイの特長として以下の点がある．

(1) 図 5 に示すように，ALU 間の接続を，1 つの ALU は隣接する 3 つの ALU のみに接続するように制限している．従来の 1 つの ALU から下段の ALU 全てに接続した構成に対して，接続線数は 72 本から 32 本に削減されている．

(2) ALU アレイの最下段からの出力データを上段に 1 クロックでフィードバックできるようにしているので，3 段毎にシーケンサでマッピングし直す必要がない．これにより後述するデータフローグラフをそのまま ALU アレイに割り当てることができ，効率的な処理が可能となる．

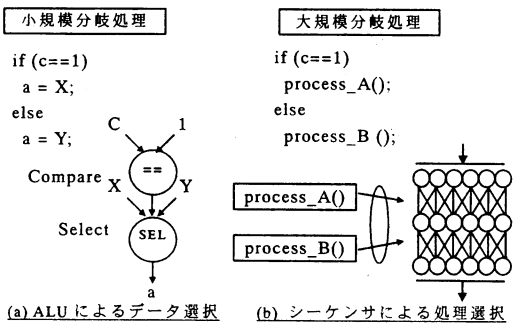
接続制限の有無に対する回路規模を評価するため，Verilog-HDL で設計した ALU の入力スイッチ回路のゲートサイズを表 1 に示す．また 10 のサンプルソースを，我々が開発したコンパイラによりマッピングし，その場合の処理に必要な段数の合計も合わせて示す．この段数が大きくなると，コマンドデータ量と実行サイクル数が増加する．表 1 より，ゲートサイズが約 63%削減できているのに対して，段数は約 6%の増加となっている．この段数の増加は，接続制限に合わせて DFG を変形しているためであるが，許容可能な範囲である．

2.4. マルチスレッド

通常の ALU アレイアーキテクチャでは，ハードウェアをマッピングするときに，実行されるすべてのハードウェアを ALU にマッピングする必要がある．このため，マッピングしたいハードウェアのサイズが実ハードウェアを超えた場合は実現できないという問題を有していた．

これに対して，提案アーキテクチャでは，毎時刻 ALU の機能および接続を切り替える方式を採用しているので，限られた量のハードウェアで，大規模な回路機能を効率的に実現できる．

マッピング手法は以下のように行う．まず，与えられた DFG を 3×6 の部分回路に分ける．そしてこの 3×6 の部分回路を ALU アレイにマッピングする．なお，各行の機能は 1 クロックで実現することとしているので，部分回路の実行には 3 クロック必要である．



(a) ALU によるデータ選択　(b) シーケンサによる処理選択

図 4 条件分岐処理方法

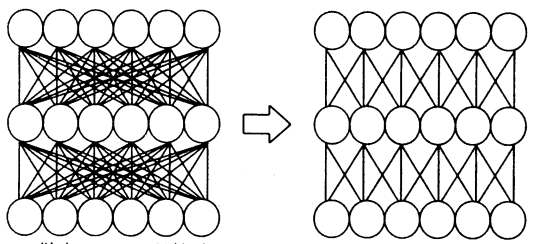

(1) 従来の ALU 間接続　(2) 提案する ALU 間接続

図 5 接続制限型 ALU アレイ

表 1 スイッチにおけるゲートサイズと処理段数

	従来の接続構成	提案する接続構成
Gate size	7487 gates	4717 gates
Rows of processing	467	494

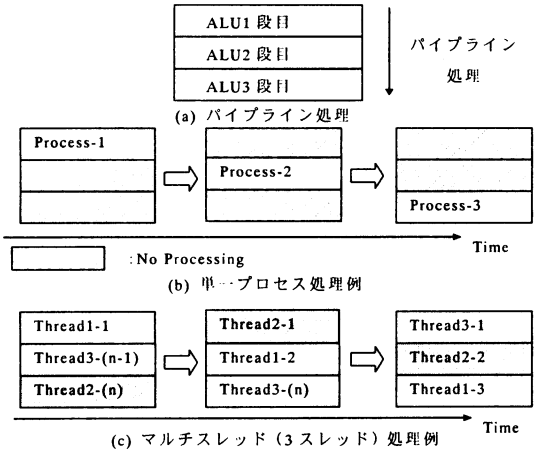


図 6 マルチスレッド

図 6(a) に 3 段分の実行の図を示す.

基本的には，この部分回路の実行を繰り返すことで全体の機能を実現できる．しかし，これでは，図 6 (b) に示すように，斜線で示した部分の ALU は実際には動作しておらず，ALU の利用効率は低い．

そこで，複数の独立な処理を並列に実行することで，ALU の稼働率を上げる手法を提案する．それは 3 つの独立なスレッドを同時に実行していることに対応するので，マルチスレッド処理と呼ぶ．実行状況を図 6 (c) に示す．3 つのスレッドが同時に横 1 行分の ALU を用いることで，ALU の空きを少なくすることができる．

本方式は，段数を増やしても適用することが可能である．現在では 3 段の ALU アレイを用いているので 3 スレッドの並列実行でよいが，n 段の場合も n 個（以上）のスレッドの並列実行を行うことで，ALU アレイの処理効率を上げることが可能である．

3×6 のアレイの行数の 3 はスレッドの数に，6 は並列度に影響を与える．

3. コンパイラ

3.1. コンパイラ動作フロー

我々は，C 言語で記述されたソースから，コマンドデータとシーケンサリストにおける設定情報を含むコンフィギュレーションデータを生成するためのコンパイラを開発した．コンパイラは，本アーキテクチャの ALU 間接続制限に合わせて C ソースを DFG に変換する．現在開発中のコンパイラでは，マルチスレッドへの自動分割には対応していないため，ユーザは 3 つのソースに分割する必要がある．この分割の自動化については，今後の課題と考えている．コンパイラの処理手順は図 7 に示すように，(1)C ソースを解析し，幾つかの仮想コードへ分割し，(2)更に接続制限にあった DFG を生成，(3)仮想コードの処理順を示すシーケンサリストを生成し，バイナリ制御コードに変換する．

3.2. シーケンサ分割と乗算展開処理

図 8 に，ユーザがシーケンサ分割を指定する場合，及び乗算展開処理におけるビット幅指定する場合の C ソース例を示す．図 8(1)のように，C ソース中にシーケンサコメントと呼ばれる/*__SEQUENCER__*/を記述することで，上述の図 7 のように，異なるデータフローグラフとして分割され，シーケンサリストに追加される．表 2 に IDCT の C ソースにおける処理の段数，コマンドデータの格納に必要なコマンドメモリのサイズ，及び ALU の使用効率を示す．シーケンサコメントを使用することで，処理段数は 79%増加するが，コマンドメモリサイズは 24%に削減されている．この処理段数とコマンドメモリ使用量はトレードオフの関係にあるが，適宜シーケンサコメントを挿入することで調整可能である．

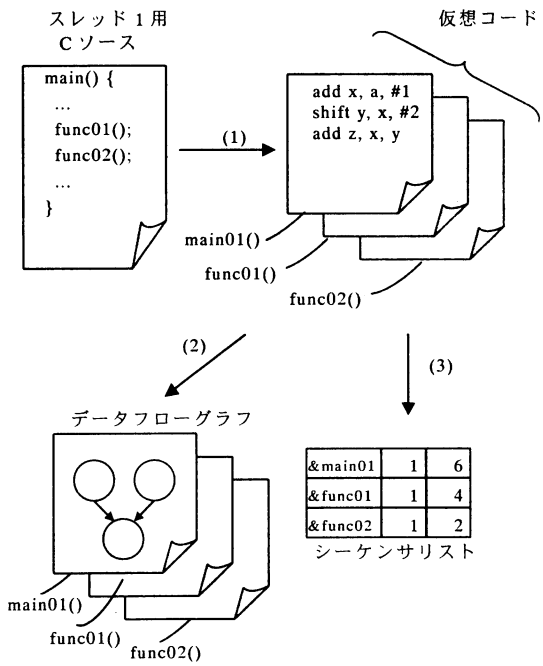


図 7 コンパイラ動作フロー

乗算命令は，2つの入力のうち，どちらかが定数であれば，シフト演算に変換され，2入力共に変数であれば，コンパイラによりデータ幅に依存した筆算処理に展開される．このデータ幅は，図8(2)のように，Cソース中にビット幅指定用コメントを記述することが可能である．図9に，データ幅を入力データ3ビットとして指定した場合の乗算処理のDFGを示す．もしコメント指定が無い場合，データ幅は16bitとして乗算展開されるため，DFGの処理段数は19段となるが，コメントでの指定により，6段となっている．尚，乗算展開時の命令multは，図10の命令リストのように，シフト，条件分岐，及び加算の組み合わせた専用命令である．

```
for/*__SEQUENCER__*/(; Blk <= V_BLOCK; Blk++)
{
    decodeBlockTexture(buf, Blk, pos);
    if(Blk == U_BLOCK) {
      DcScalar = DcScalarS; }          } (1)
    ...
}
...
//#mul_bit=3*3;  ------------------>  (2)
z = x * y;
```

図8 分割指定，乗算指定したCソース例

表2 IDCTでの処理段数とコマンドメモリのサイズ及びALUの使用効率

	シーケンサコメント未使用	シーケンサコメント使用
Row of processing	825	1,473
Memory Size (Kbytes)	23.2	5.5
ALU Usage Rate (%)	78.3	49.2

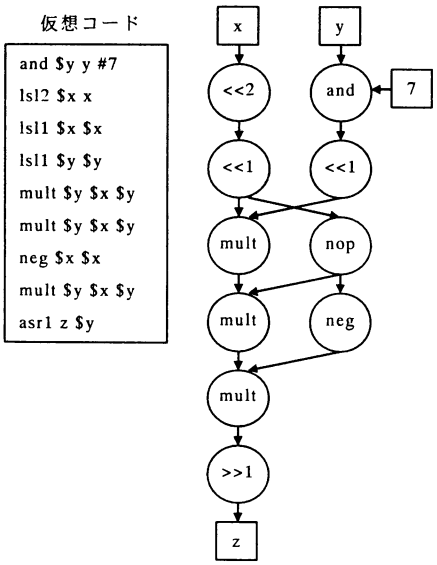


図9 乗算処理のDFG (x, y: 3bit, z: 6bit)

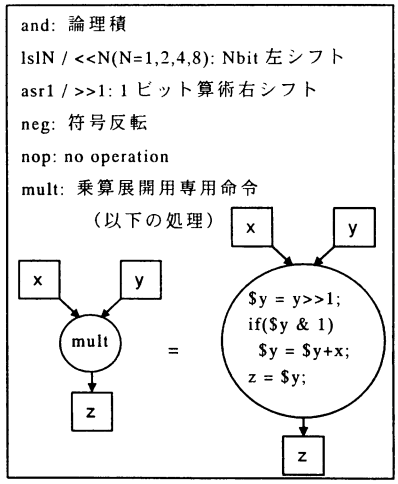


図10 命令リスト

4. MPEG-4デコーダの設計と評価

4.1. MPEG-4デコードアルゴリズムのスレッド分割

本プロセッサにおいて，MPEG-4デコード処理を設計し，評価を行った．対応する仕様はSimple Profile(SIF resolution, 15fps)である．図11にMPEG-4デコード処理のブロック図を示す．今回，図における点線で囲む後半部分について設計を行った．上述の通り，各スレッドに対してCソースを用意する必要があるため，表3のように処理ブロックをスレッドに分割した．この際，各スレッドの処理量ができるだけ平均化され，スレッド間で共有されるデータが少なくなるように分割している．

4.2. シミュレーションと回路規模評価

図12に，本プロセッサの動作検証を行うためのシミュレーションの処理フロー図を示す．我々は，本プロセッサをVerilog-HDL記述により設計したが，ALUアレイにおける処理をシミュレーションし，視覚化するため，PC上で動作するソフトウェアモデルと呼ばれるツールも開発した．図12のように，ソフトウェアモデルとRTL記述の回路は，コンパイラから同じコンフィギュレーションデータを入力とし，全く同じタイミングで動作するため，PC上でALUアレイの動作解析を行うことができる．それに加え，ソフトウェアモデルは，MPEG-4デコードのビューアと連動させており，図13のようなデコード途中結果を視認することができる．

本プロセッサのRTL記述の回路の規模は，当社開発のASIC回路に対しては，約2.4倍の規模で実現できた．

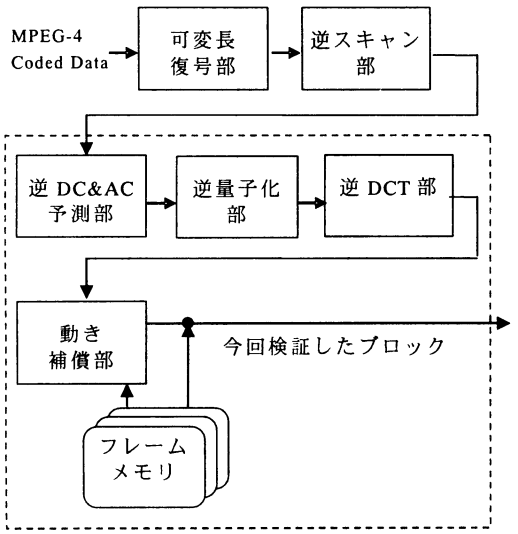


図 11 MPEG-4 デコードのブロック図

表 3 各スレッドに対する機能ブロックの割り当て

C ソース	MPEG-4 機能ブロック
スレッド 1	逆 DC & AC 予測
スレッド 2	逆量子化，逆 DCT
スレッド 3	動き補償，フレームメモリアクセス

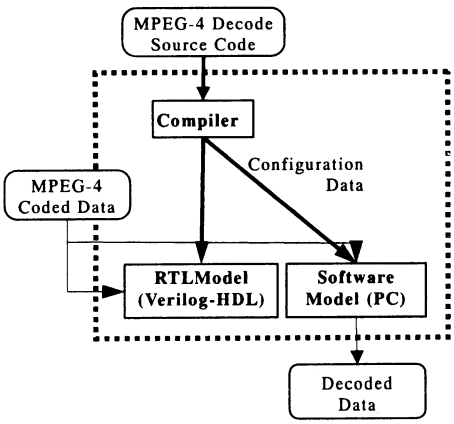


図 12 シミュレーションの処理フロー図

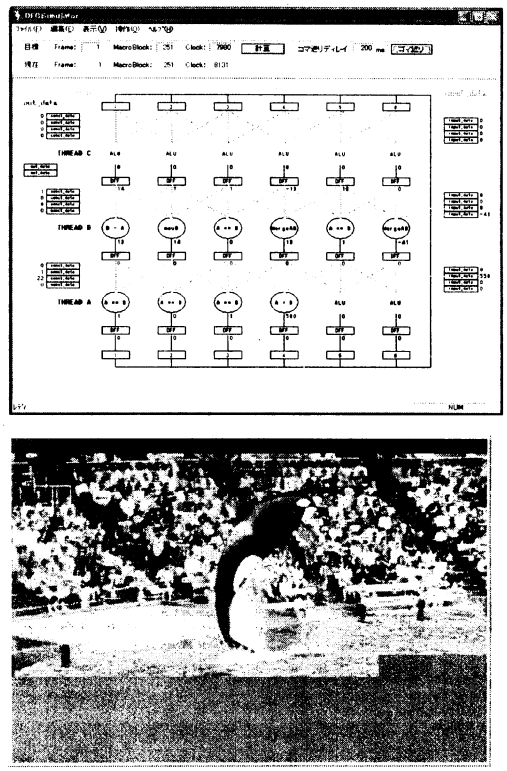


図 13 ソフトウェアモデルによる MPEG-4
デコード処理のシミュレーション

5. 結論

小型かつ処理性能の高いリコンフィギュラブルプロセッサを開発するため，省面積用に接続制限した ALU アレイアーキテクチャを提案し，MPEG-4 デコードアルゴリズムを用いた動作検証を行った．その結果，提案したプロセッサは，MPEG-4 デコーダの ASIC 回路に対し，およそ 2.4 倍の規模であった．本プロセッサ上で，3 個のアプリケーションを搭載すれば，回路規模面での有利性が出るので，現実的に実現可能なサイズであることが分かった．

文　献

[1] あべ松 竜盛，“データ駆動型 ALU アレイを使用した動的再構成可能計算機の設計と評価”，奈良先端大修士論文，NAIST-IS-MT9751004, 1999

[2] 天野英晴，“ダイナミックリコンフィギャラブルプロセッサの研究開発動向”，情報処理学会研究報告，vol.2003,no.105,pp.139-144, 2003