システムＬＳＩ設計技術　96－7
（2000．5．12）

# 汎用エンジンRM-Vを対象とする高位合成システム

室屋 友和†, 橋本 匡史††, 高林 宏忠†††, 黒木 修隆†, 沼 昌宏†

†神戸大学, ††三菱電機(株), †††(株)PFU
†〒657-8501 神戸市灘区六甲台町 1-1
E-mail: {muroya, numa, kuroki}@cas.eedept.kobe-u.ac.jp

あらまし　内部の論理を電気的に書替え可能な LSI である FPGA とメモリから構成された汎用エンジン RM-V（Reconfigurable Machine-V）を対象とする，高位合成システム RMAC-V（Reconfigurable Machine Application Compiler for RM-V）を開発した。本システムでは，C 言語による動作記述を入力し，RT レベルの VHDL 記述を合成する。アプリケーションのメモリ・アクセスに要するクロック数を削減するために，マルチクロック・スケジューリングと行アドレスの先行入力を提案している。WTE（Wavelet Transform Engine）を用いた実験の結果，従来手法と比較して総クロック数が 33％削減される効果が確認された。

キーワード　高位合成，C 言語，汎用エンジン，VHDL，FPGA，SDRAM

# A High-Level Synthesis System for Reconfigurable Machine: RM-V

Tomokazu Muroya†, Koji Hashimoto††, Hirotada Takabayashi†††,
Nobutaka Kuroki† and Masahiro Numa†

†Kobe University, ††Mitsubishi Electric Corporation, †††PFU Limited
†1-1, Rokkodai, Nada, Kobe, Hyogo 657-8501, Japan
E-mail: {muroya, numa, kuroki}@cas.eedept.kobe-u.ac.jp

**Abstract**　This paper presents a high-level synthesis system, called RMAC-V (Reconfigurable Machine Application Compiler for RM-V), for applications using SDRAMs implemented on the flexible architecture of RM-V (Reconfigurable Machine-V) combining FPGAs and memories. Given an application program written in the C language, RMAC-V produces an RT-level hardware description in VHDL. To reduce the clock counts needed to access memories, RMAC-V introduces two techniques: multi-clock scheduling and preloading row address. Experimental results on Wavelet Transform Engine (WTE) have shown 33 % fewer total clock counts than those with conventional method.

**key words**　High-level synthesis, C language, Reconfigurable Machine, VHDL, FPGA, SDRAM

## 1. はじめに

大規模・複雑化した論理回路を対象とする CAD や大きなデータ量を扱う画像処理の分野においては，計算機が扱うデータ量の増加にともない，処理時間の増加が問題となっている。処理時間を短縮する方法の一つとして，対象とするアルゴリズムのハードウェア化が挙げられる。その一方で，従来の専用ハードウェアによる高速化手法 [1], [2] には，柔軟性・汎用性に欠ける点に問題があった。

そこで，内部の論理を電気的に書替え可能な LSI である FPGA(Field Programmable Gate Array) [3], [4] とメモリから構成される汎用エンジン [5] の概念が提案され，これまでに汎用エンジン RM(Reconfigurable Machine)-I [6], -II [7], -III [8], -IV [9], -V [10] の 5 台のプロトタイプが実現されてきた。最新の RM-V では，新たに 64 Mbit SDRAM(Synchronous Dynamic RAM) を最大 32 個実装してメモリ容量を飛躍的に拡大するとともに，130 K ゲート規模の FPGA を最大 5 個搭載することで大規模な回路を実現可能とした。

CAD や画像処理など計算量の多い処理を汎用エンジン上で実現するためには，そのアルゴリズムを RT (Register Transfer)レベルで記述するアプリケーション設計を行う必要がある。論理合成ツールの進歩によって設計効率は改善されたが，大規模で複雑な回路を短時間に設計するためには，機能設計の自動化が望まれる。そのため，より高位の動作記述から合成を行う高位合成 [11]-[13] に関する研究が進められている。

その一方で，RM-V で新たに実装された SDRAM の制御は，それまでの汎用エンジンにも実装されてきた SRAM(Static RAM)と比べて複雑であるため，人手による RT レベル設計には数週間程度の時間を要していた。最近では，SDRAM のアクセス・モデルを考慮した高位合成手法 [13] も提案されているが，複数のメモリを扱えない点に問題がある。したがって，汎用エンジンのように，複数のメモリ・バンクへの並列アクセスによって高速化を図るシステムには対応できない。

さらに，SDRAM は SRAM よりもレイテンシ(latency)，すなわちアクセスに要するクロック数が多いため，動作周波数を向上させることで，メモリ・アクセスに要する時間を削減する手法が必要である。そこで，以下の 2 点を目的として，C 言語の動作記述をもとに RT レベルの VHDL(VHSIC Hardware Description Language)記述を自動的に生成する高位合成システム RMAC-V (Reconfigurable Machine Application Compiler for RM-V)を実現した。

i) 複数の SDRAM に対応したスケジューリングの実現による，RM-V のアプリケーション設計時間短縮
ii) メモリ・アクセスに必要なクロック数の削減による，アプリケーション性能の向上

以下，第 2 章では，本システムの仕様と構成を示し，第 3 章では，本システムに適用した SDRAM の特性を考慮したスケジューリング手法およびメモリ割付け手法を提案する。第 4 章では本システムを適用した実験結果を示し，第 5 章でまとめを行う。

表 1 RMAC-V の入力記述

| 項 目 | 内 容 |
|---|---|
| 変数の型 | int(16 ビット)，信号線を表すデータ型 Sig |
| 式の演算子 | & \| ~ ^ + - ++ -- << >> += -= <<= >>= \|= &= && \|\| != == <= >= < > * |
| 制御文 | ANSI C に含まれるすべての制御文が使用可能:<br>・ if ～ else ～ 文<br>・ for 文, while 文, do ～ while 文<br>・ switch ～ case 文<br>・ goto 文 |
| 配列 | 任意のグローバル 1 次元配列が使用可能 |

## 2. 高位合成システム RMAC-V

高位合成システム RMAC-V は，C 言語をベースとした動作記述を入力して，RT レベルの VHDL 記述を出力する。以下，入力記述に関する仕様と本システムの構成について述べる。

### 2.1 RMAC-V の入力記述

本システムにおける入力記述に用いられる演算子，予約語(キーワード)を表 1 に示す。記述に関して，基本的には ANSI C の記述を用いるが，ANSI C とは以下の点で異なる。

i) 構造体(struct)と共用体(union)，列挙型(enum)，およびポインタには未対応
ii) 変数の型として，int(16 ビット)，ビット指定可能な型 Sig を利用可能
iii) 配列は 1 次元のみ対応し，外部変数として扱う
iv) 除算，剰余には未対応

また，SDRAM を用いたアプリケーションの動作記述

において，配列の連続アドレスに対するアクセスが存在した場合，合成時のオプション指定によりバースト転送を適用することが可能である。

### 2.2 RMAC-V の処理概要

図 1 に示すように，本システムは以下のような複数のプログラムから構成されている。プログラム間のデータの受渡しは，中間ファイルを介して行う。

#### (1) C パーサ RCC [14]

本システムは，パーサとして RCC（Retargetable C Compiler）と呼ばれるコンパイラを使用している。RCC は，各アーキテクチャのコードを生成する前の中間形式のデータであるシンボル・データを出力する機能をもっている。本システムでは，このシンボル・データを利用する。

#### (2) ビット・パーサ

ビット・パーサにより，動作記述中で定義されている各変数，配列のビット幅を読み取り，ビット・ファイルとして出力する。

#### (3) CDFG ジェネレータ

C パーサ，ビット・パーサが生成したシンボル・ファイル，ビット・ファイルを読み取り，そこから本システムが使用する内部形式である CDFG（Control Data Flow Graph）のデータを生成する。

#### (4) スケジューラ，アロケータ

スケジューラは，CDFG データをもとに，各演算をどの演算ステップへ割り付けるかを決定するスケジューリングを行う。このとき，アプリケーションにおけるメモリのアクセス動作を考慮する。また，CDFG データから動作記述中で使用されている配列のデータ量と依存関係を抽出し，それらをもとに各配列をメモリに割り付ける。

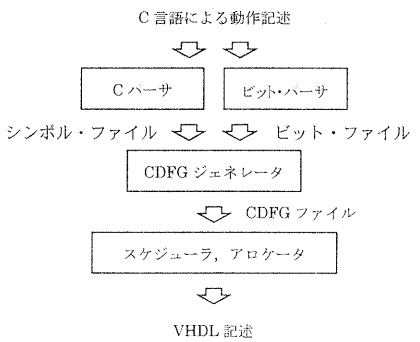


図 1 RMAC-V の構成

アロケータは，スケジューラが出力する CDFG ファイルを読み取り，各演算，各変数をそれぞれ演算器，レジスタに割り付けるデータパス・アロケーションを行う。この処理と並行して，制御論理を合成する。最後に，合成した結果を RT レベルの VHDL 記述で出力する。

## 3. SDRAM に対応した高位合成手法

高位合成システム RMAC-V は，通常の SRAM のみならず，汎用エンジン RM-V が搭載する SDRAM を用いたアプリケーションにも対応する。メモリ・アクセスをいかに効率よく行うかが，アプリケーションの性能を大きく左右する。そこで，SDRAM のタイミング・モデルを考慮したスケジューリングを行うことで，メモリ・アクセスに必要なクロック数を削減する手法を提案する。さらに，メモリ割付けを考慮したスケジューリング手法も考案し，図 1 に示したスケジューラに組み込んだ。

### 3.1 マルチクロック・スケジューリング

SRAM を用いたアプリケーションでは，メモリ・データの読出し，書込みは 1 クロックで完了する。それに対して SDRAM では，メモリ・アクセスの際に，行アドレスと列アドレスの 2 回に分けてアドレスを与える必要がある。さらに，列アドレス入力からデータ出力までに，CAS（Column Address Strobe）レイテンシと呼ばれるクロック数分の遅延が生じる。汎用エンジンのようなメモリと FPGA を組み合わせたシステムでは，メモリ・アクセスの高速化が処理性能向上の鍵となるので，この問題を解決する必要がある。

RM-V において，SRAM へのアクセスのみ行う単純動作については約 40 MHz での動作が確認されたが，SRAM を用いたアプリケーションの最高動作周波数は 20 MHz 程度である。その一方で，SDRAM へのアクセスについては，59 MHz での動作が確認されている。

したがって，SDRAM に入力するクロック周波数をアプリケーション部の整数倍にすることで，SDRAM のスループットを向上させる効果が期待できる。本稿では，このように SDRAM に対してアプリケーション部と異なるクロックを与えることをマルチクロック方式と呼び，この方式を想定して行うスケジューリングをマルチクロック・

表 2 ランダム・アクセスに必要なクロック数

| メモリ | SDRAM | | SRAM |
|---|---|---|---|
| マルチクロック | 非適用 | 適用 | — |
| READ | 5 | 3 | 1 |
| WRITE | 3 | 2 | 1 |

スケジューリングと呼ぶ。SDRAMの動作確認結果から，倍率は2倍，もしくは3倍にすることが考えられが，現状では2倍に固定している。

以下，ランダム・アクセスと連続アクセスの2種類について，マルチクロック・スケジューリングの効果を示す。

### 3.1.1 ランダム・アクセス

非連続的なアドレスに対するメモリ・アクセス，すなわちランダム・アクセスの場合は，行アドレスと列アドレスをそれぞれ設定する必要がある。たとえば，CAS レイテンシを 2 とすると，データを受け取るまでに 5 クロックを要する。

ランダム・アクセスの読出しに対して，マルチクロック・スケジューリングを適用した場合の SDRAM アクセス動作を図 2 に示す。ここで，CLK はアプリケーション部に与えるクロック，SDCLK は SDRAM に与えるクロック，ADR はアドレスを表す。SDCLK の周波数を CLK の 2 倍とすることで，行アドレスおよび列アドレスをより早いタイミングでメモリに与えることが可能となる。結果として，マルチクロック・スケジューリングを適用しない場合よりも，データを 2 クロック早く読み出すことができる。

表 2 に，マルチクロック・スケジューリングを適用したときのランダム・アクセスに必要なクロック数を，SRAM と比較して示す。SRAM には及ばないが，読出し，書込みともに 2 クロックの削減効果が示されている。

### 3.1.2 連続アクセス

一方，連続アドレスへのアクセス，すなわち連続アクセスに対しては，バースト転送を利用することでデータの入出力の高速化が可能である。バースト転送は SDRAM のデータ転送モードの一種である。先頭のアドレスを 1 回指定するだけで，バースト長で設定した個数のデータを連続的に転送できるため，高いスループットが得られる。

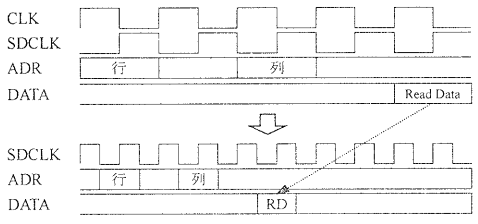


図 2 マルチクロック・スケジューリングの例

表 3 連続アクセスに必要なクロック数

| メモリ | SDRAM | | | | SRAM | |
|---|---|---|---|---|---|---|
| 連続するデータ数 | 4 | | 8 | | 4 | 8 |
| マルチクロック | 非適用 | 適用 | 非適用 | 適用 | — | — |
| READ | 8 | 5 | 12 | 7 | 4 | 8 |
| WRITE | 6 | 4 | 10 | 6 | 4 | 8 |

このバースト転送にマルチクロック方式を組み合わせることで，さらなる高速化が期待できる。表 3 に，CAS レイテンシを 2 に設定したときの 4 個，8 個のデータに対する連続アクセスについて，マルチクロック方式の適用／非適用によるクロック数の違いを，SRAM との比較を交えて示す。マルチクロック方式の適用によって，クロック数が 30 % から 40 % 程度削減されている。また，連続データ数が 8 個の場合には，SRAM よりも高速なアクセスが可能となる。

## 3.2 行アドレスの先行入力

SDRAM では，行アドレスと列アドレスの 2 回に分けてアドレスを入力する。そのために，前節に述べたマルチクロック・スケジューリングを行ったとしても，CAS レイテンシが 2 のとき，ランダム・アドレスに対する読出しでは，データの出力に 3 クロックを必要とする。

動作記述内で条件分岐のない部分については，メモリ・アクセスを行う一つ前のステートで，次にどの配列にアクセスするかが判明している。RMAC-V は行アドレスを配列番号，すなわち個々の配列を識別する番号に対応づける。よって，図 3 に示すように配列番号に対応

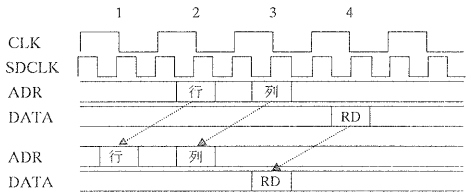


図 3 行アドレスの先行入力の例

した行アドレスを1クロック先に入力することで，アクセス時間を1クロック短縮できる。これを行アドレスの先行入力と呼ぶ。

行アドレスの先行入力を行った場合，アクセスに必要なクロック数は表2，表3に示した値よりも，さらに1クロック少なくなる。その結果，8個の連続データに対するアクセスを例にとると，SRAMと比べて読出しでは2クロック，書込みでは3クロック削減できる。

このように，行アドレスの先行入力をマルチクロック・スケジューリングと組み合わせることで，クロック数の削減効果が期待できる。

### 3.3 メモリ割付けを考慮したスケジューリング

RM-Vには，130Kゲートの大規模FPGAを実装している。そこでRMAC-Vでは，リソースに制限がないときに有効とされるASAP(as soon as possible)スケジューリング法を採用した。図4に処理概要を示す。

スケジューリング対象とするCDFGの各ノードに，それを実行すべきステートSTを属性としてもたせる。このSTを順次決定することが，スケジューリングの主な目的である。具体的には，加減算やシフトなどの1クロックで実行可能な演算については，入力側のノード集合(fanin)の各要素がもつSTの最大値+1を，そのノードのSTとして設定する。ただし，乗算のように複数クロックを必要とすることがある演算については，合成時にパラメータとして指定される乗算レイテンシ(ML)を加える。一方，配列のノードに対してはメモリ・アクセスを考慮する必要があるため，以下に述べるメモリ割付けスケジューリングを行う。

SDRAMを用いたアプリケーションを対象とする場合，メモリ・アクセスに必要なクロック数が転送モード等によって変化するため，スケジューリングの前に最適なメモリ割付けを行うことは難しい。そこで，スケジューリングと同時にメモリ割付けを行う，メモリ割付けスケジューリングの手法を考案した。図5に示すように，メモリ割付けスケジューリングは以下の順で行われる。

#### (1) 暫定ステートの決定

まず，暫定的なステートを決定してSTに設定する。加減算やシフトなどの演算子ノードと同様に，入力先のノード集合で最大となるステートに1を加えた値が暫定ステートになる。

#### (2) 割付け確認

着目する配列が，メモリに割り付け済みかどうかを確認する。割り付け済みの場合は，当該ステートにおけるアクセス競合の有無を調べ，もし競合があれば暫定ステートの修正を行う。そして，連続アドレス・アクセスかランダム・アクセスかによって適切な転送モードを選択するとともに，ステートを決定する。

まだ配列がメモリに割り付けられていない場合は，以下で述べるメモリ・バンク割付けを行う。

#### (3) メモリ・バンク割付け

割付け対象とする配列に対してアクセスを行う各ステートについて，同時アクセスが必要となる他の配列とは異なるメモリ・バンクに割り付ける必要がある。全メモリ・

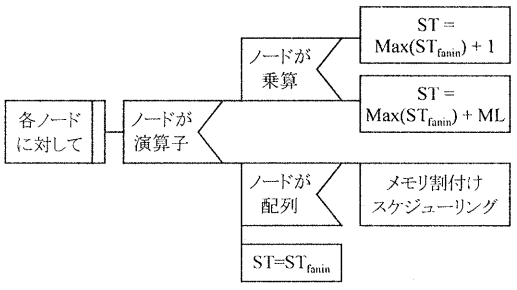


図4 ASAPスケジューリング

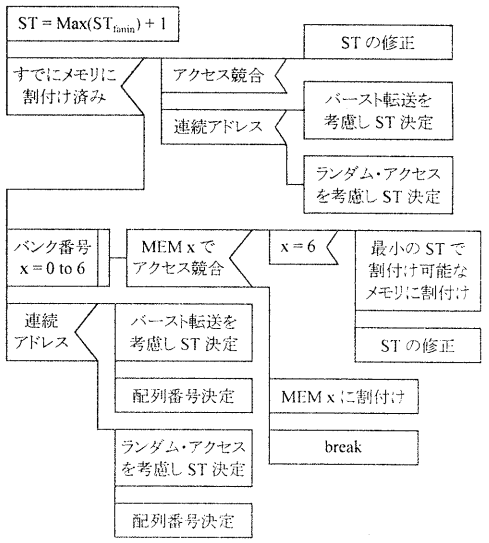


図5 メモリ割付けスケジューリング

表 4 バースト転送適用時の状態ステップ数・総クロック数

| 方式 | | (a) | (b) | (c) |
|---|---|---|---|---|
| マルチクロック・スケジューリング | | — | — | ○ |
| 行アドレスの先行入力 | | — | ○ | ○ |
| 状態ステップ数 | | 20 | 15 | 14 |
| 総クロック数 | 非並列化 | 294,914 | 212,994 | 196,610 |
| | 並列化 | 147,458 | 106,498 | 98,306 |

バンクにおいてアクセス競合が生じる場合，ステート数の増加が最小となるメモリ・バンクを探し，暫定ステートの修正を行う。

### (4) ステートと配列番号の決定

連続アドレス・アクセスの場合はバースト転送の適用を前提とし，またランダム・アクセスの場合はそれを考慮して，最終的なステートを決定する。また，個々の配列を識別する配列番号も決定する。

## 4. 実験結果

提案手法に基づく高位合成システム RMAC-V を計算機上に実現した。手法の有効性を評価するために行った実験の結果について述べる。具体的には，ウェーブレット変換処理について，マルチクロック・スケジューリングと行アドレスの先行入力を適用した合成結果から，提案手法の有効性を評価する。

ウェーブレット変換 [15] は，信号処理の分野で広く用いられている周波数解析法である。高周波成分については時間分解能が高く，低周波成分については周波数分解能が高いという特徴をもっており，信号の局所的性質と大域的性質を同時に解析することができる。実験には，256×256 画素，256 階調の入力画像に対して，ハール・ウェーブレット変換 [15] を適用した。

まず，ウェーブレット変換エンジン（Wavelet Transform Engine : WTE）の動作を C 言語で記述した後，それを入力として高位合成を行った。RMAC-V によるメモリ割付けの結果，WTE は RM-V 上に 7 バンクある SDRAM のうち，3 バンクのみを利用していた。そこで，メモリ・バンクを有効に利用するために，二つの回路で並列処理を行うように動作記述を変更した後，再度高位合成を行った。

表 4 に，(a) バースト転送のみ適用，(b) バースト転送とマルチクロック・スケジューリングを適用，(c) バースト転送，マルチクロック・スケジューリング，行アドレスの先行入力のすべてを適用，の 3 種類について，状態ステップ数と総クロック数の比較結果を示す。マルチクロック・スケジューリングと行アドレスの先行入力をともに適用することによって，メモリ・アクセスに必要なクロック数が削減され，状態ステップ数が 6 減少する効果があった。その結果，処理に必要な総クロック数が 33 % 削減された。この結果から，提案手法が総クロック数の削減に有効であることが示された。

表 5 WTE の設計結果

| | |
|---|---|
| 使用メモリバンク数 | 6 |
| 状態ステップ数 | 14 |
| 総クロック数 | 98,306 |
| 最高動作周波数 | 22.2 MHz* |

* クリティカルパスの遅延時間から算出

クロック数削減効果が顕著であった，マルチクロック・スケジューリングと行アドレスの先行入力を適用した並列処理回路について，シミュレーションを行った。結果を表 5 に示す。この実験では入力画像として図 6 に示す SIDBA 標準画像“GIRL”(256×256 画素，モノクロ 256 階調)を使用した。また WTE により変換された出力画像を，図 7 に示す。

マルチクロック・スケジューリングと行アドレスの先行入力を適用することで，5 クロックで 4 画素を読み出すことが可能となった。さらに，各配列が少ないメモリバンク数で割り付けられるため，二つの回路を並列に動作させることが可能となった。その結果，5 クロックで 8 画素

図 6 入力画像

の読出しが可能となり，総クロック数が削減された。SDRAMコントローラ部に若干の問題点が残っているため，まだ実機での完全な動作確認にはいたっていないが，クリティカル・パスの遅延情報に基づくアプリケーション部の動作周波数は22.2 MHzと評価された。

実験結果から，とくにバースト転送を適用できる場合には，メモリ・アクセスに必要なクロック数が大きく削減され，処理速度が向上することが明らかとなった。

さらに，従来はSDRAMを用いたアプリケーションの設計には数週間程度の時間を要していたが，RMAC-Vを利用することで数秒に短縮され，設計時間短縮に大きな効果が認められた。

## 5. まとめ

本稿では，汎用エンジンRM-Vのアプリケーション設計時間の短縮とその性能向上を目的として，SDRAMのメモリ・アクセスを高速化するマルチクロック・スケジューリングと行アドレスの先行入力の手法を提案した。さらに，これらの手法に基づく高位合成システムRMAC-Vを実現した。ウェーブレット変換に適用した実験の結果，設計時間を大きく短縮できる効果とともに，提案手法による性能向上効果が確認された。

今後の課題としては，パイプライン処理への対応，ランダム・アクセスの高速化などが挙げられる。

### 参考文献


[1] T. Blank, "A survey of hardware accelerators used in computer-aided design," IEEE Design and Test of Computers, vol. 1, no. 3, pp. 21-39, 1984.

[2] G. F. Pfister, "The Yorktown simulation engine: introduction," Proc. 19th DAC, pp. 55-59, 1982.

[3] The XC4000 Data Book, Xilinx, 1991.

[4] FLEX 10K Embedded Programmable logic family data sheet, Altera Corporation, 1996.

[5] 菅沼直昭，村田之広，冨田昌宏，平野浩太郎，"汎用エンジンの開発と論理診断への応用"，DAシンポジウム'92, pp. 89-92, 1992.

[6] 冨田昌宏，村田之広，菅沼直昭，平野浩太郎，"汎用エンジンRM-Iの開発"，情報処理学会第45回全国大会講演論文集, vol. 6, pp. 155-156, 1992.

[7] 冨田昌宏，菅沼直昭，澄川文徳，平野浩太郎，"汎用エンジンRM-IIの構成"，JSPP'93, pp. 151-158, 1993.

[8] 澄川文徳，垣原雅已，菅沼直昭，冨田昌宏，平野浩太郎，"配線変更可能なボードを用いた汎用エンジンRM-III"，情報処理学会第48回全国大会講演論文集, vol. 6, pp. 119-120, 1994.

[9] 井上真一，奥田知史，高瀬 幹，沼 昌宏，平野浩太郎，"汎用エンジンRM-IVとその応用"，DAシンポジウム'96, pp. 99-104, 1996.

[10] 山口卓也，安西伸介，水谷 敦，黒木修隆，沼 昌宏，"汎用エンジンRM-Vの開発"，電気関係学会関西支部連合大会論文集, 1999.

[11] M. C. McFarland, A. C. Parker and R. Camposano, "Tutorial on high-level synthesis," Proc. 25th DAC, pp. 330-335, 1988.

[12] A. Nicolau and R. Potasman, "Incremental tree height reduction for high-level synthesis," Proc. 28th DAC, pp. 770-774, 1991.

[13] A. Khare, P. R. Panda, N. D. Dutt and A. Nicolau, "High-level synthesis with synchronous and RAMBUS DRAMs," SASIMI'98, pp. 186-193, 1998.

[14] C. W. Fraser, D. Hanson, A Retargetable C Compiler for ANSI C, ACM SIGPLAN Notices, 1991.

[15] O. Rioul and M. Vetterli, "Wavelet and signal processing," IEEE SP., vol. 8, no. 4, pp. 14-38, 1991.


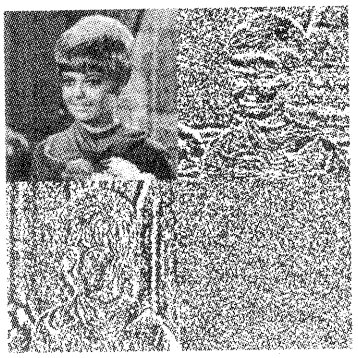

図7　出力画像