PARS アーキテクチャの詳細設計に関する一考察

谷川一哉† 吉田哲生† 児島彰‡ 弘中哲夫‡ 吉田典可‡
E-mail: kazuya@csys.ce.hiroshima-cu.ac.jp

† 広島市立大学大学院 情報科学研究科
‡ 広島市立大学 情報科学部

本研究では 1 サイクル毎にハードウェアの構成を変更できる再構成型コンピュータとして PARS アーキテクチャを提案している．その PARS アーキテクチャを評価するために，PARS アーキテクチャのプロトタイプマシンを Verilog-HDL を使用して設計した．本稿ではその詳細について述べる．今回設計したプロトタイプマシンは 8 ビットの演算を実行する再構成型演算ユニットを 72 個搭載し，それらを再構成するのに必要な構成情報は 4096 ビットであった．また本稿では，設計したプロトタイプマシンの Verilog-HDL の記述を使用し，論理シミュレーションによって性能を評価した．その結果 1 サイクルあたりの再構成型演算ユニットの平均使用率は 35% であった．

The Detailed Design of the PARS Architecture

Kazuya Tanigawa† Tetsuo Yoshida†
Akira Kojima‡ Tetsuo Hironaka‡ Noriyoshi Yoshida‡

†Graduate School of Information Sciences, Hiroshima City University
‡Faculty of Information Sciences, Hiroshima City University

We have proposed the PARS Architecture as a reconfigurable computer which enables single-cycle reconfiguration. To evaluate this architecture, we design a prototype machine of the architecture with Verilog-HDL. This paper describes its details. The prototype machine comprises 72 reconfigurable execution units which execute 8 bits operations, and the code size for reconfiguration is 4096 bits. Also, this paper describes performance evaluation of the prototype machine by gate-level simulation with Verilog-HDL. The result shows the utilization of reconfigurable execution units at 1 cycle is about 35 % of the number of all units.

1 はじめに

近年，アプリケーションの実行速度を向上させるコンピュータとして再構成型コンピュータが注目されている．しかし，これまでに提案された再構成型コンピュータのプログラミングモデルは，1) 固有のプロセッサに依存したプログラミングモデルになっており使いにくい，または，2) C 言語のような逐次実行を前提とするプログラミングモデルを使用しているため並列性が抽出しにくい，といった問題がある．

そこで上述の問題に対処する手法として，本研究では PARS(PARallel Structure) プログラミングモデルを提案している [1]．PARS プログラミングモデルは並列性を損なうことなくアルゴリズムをハードウェアで実行できるようなプログラミングモデルを目指す．

このような PARS プログラミングモデルでは再構成型コンピュータをベースとしたプログラミングモデルとしている．また，PARS プログラミングモデルに最適化された再構成型コンピュータの 1 つとして，PARS アーキテクチャを提案している [2]．

PARS アーキテクチャではサイズに指定がない再構成情報を効率よく構成できるような再構成型コンピュータを目指している．本稿ではそのような特徴を持つアーキテクチャを実際に設計し評価するために，PARS アーキテクチャのプロトタイプマシンとして「tempo」を設計し，その詳細について説明する．

本稿の構成を以下に示す．2 節では PARS アーキテクチャの構成について説明する．3 節では tempo の各構成要素について説明し，4 節で tempo を論理設計した結果について述べる．5 節では評価結果について述べ，6 節でまとめる．

2 PARS アーキテクチャ

PARS アーキテクチャは，動的にハードウェアの構成を変更することによって，実ハードウェアよりサイズの大きい再構成情報を構成できる再構成型コンピュータである．PARS アーキテクチャでは，そのような再構成情報を固定の大きさに分割し，そ

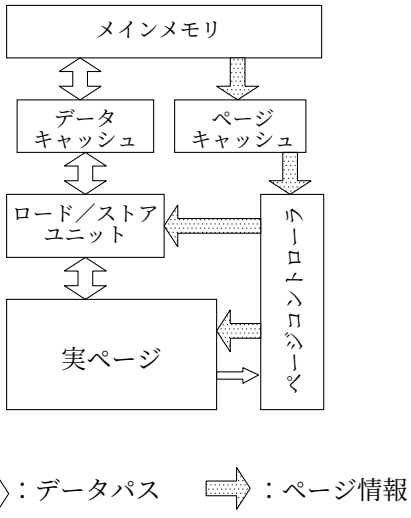

図 1: PARS アーキテクチャの構成図

れらを逐次的に実行する．PARS アーキテクチャではその固定の大きさに分割されたものをページと呼ぶ．そのページへの分割はコンパイラなどを使用して静的に行われる．

PARS アーキテクチャの特徴を以下に列挙する．

1. PARS アーキテクチャは演算器をベースとした再構成型演算ユニットを持つ再構成型コンピュータである．
2. PARS アーキテクチャは全ての再構成型演算ユニットを 1 サイクルで再構成する．

1 つ目の特徴の利点は LUT(LookUp Table) ベースの再構成型演算ユニットより再構成情報を大幅に削減できることである．この 2 つ目の特徴は，1 つ目の特徴による再構成情報の削減により，実現しやすくなっている．2 つ目の特徴によって PARS アーキテクチャは効率的に動的再構成をし，実ハードウェアより大きな再構成情報を効率良く構成できる．

図 1 に PARS アーキテクチャの構成図を示す．同図において白抜きの矢印はデータパスを表す．網掛けの矢印はページによる情報の流れを表す．PARS アーキテクチャの各構成要素については次節で説明する．

3 プロトタイプ tempo

本節では PARS アーキテクチャのプロトタイプマシン tempo について説明する．

3.1 プログラムの構造と実行方式

図 2 に tempo におけるプログラムの構造と実行方式を示す．tempo で実行されるプログラムは複数

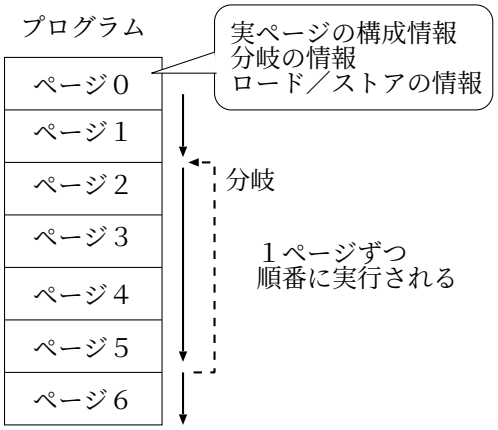

図 2: プログラムの構造と実行方式

のページから構成される．tempo はプログラムを構成する各ページを逐次的に実ページに割り当てることによりプログラムを実行する．ページで指定される情報は実ページの構成情報，分岐の情報，ロード／ストアの情報である．

tempo では分岐によって実行するページを変更することもできる．この分岐先のページを指定するために tempo のプログラムではページ毎にページ番号が割り振られている．

3.2 tempo の構成

tempo の構成図は図 1 で示した PARS アーキテクチャの構成図と同じである．表 1 に tempo の諸元をまとめ，以下で各構成要素について説明する．

メインメモリ プログラムとプログラムで使用するデータが格納される．

データキャッシュ プログラムで使用するデータ用のキャッシュである．

ページキャッシュ ページ用のキャッシュである．

ページコントローラ ページの実行を制御する．

実ページ 主に再構成型演算ユニットと配線資源から構成される．PARS アーキテクチャでは再構成型演算ユニットをセルと呼ぶ．

ロード／ストアユニット プログラムの実行に必要なデータのロードとストアを実行する．

以下で，ページコントローラ，実ページ，ロード／ストアユニットの詳細について説明する．

表 1: tempo の諸元

ページキャッシュサイズ	16K バイト
データキャッシュサイズ	16K バイト
ページキャッシュバンド幅	4096 ビット
データキャッシュバンド幅	32 ビット × 4 ポート
演算器数	72 セル
演算のデータ幅	8 ビット

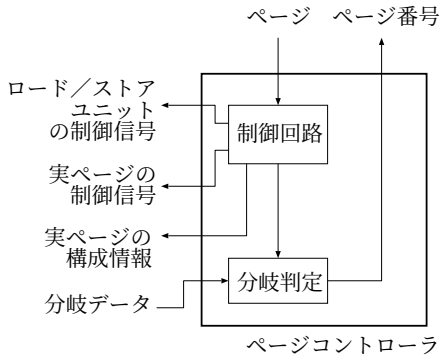

図 3: ページコントローラの構成図

3.3 各構成ユニットの詳細

3.3.1 ページコントローラ

図 3 にページコントローラの構成図を示す．以下，ページコントローラを構成するユニットについて説明する．

制御回路 ページキャッシュからフェッチしてきたページの情報を基にロード／ストアユニットと実ページにデータを送信する．またページに含まれる分岐に関する情報を抽出し，分岐判定回路にデータを送信する．

分岐判定 実ページから送信される分岐データを基に分岐の判定をし，次に実行するページ番号を決定する．分岐判定ユニットは次に実行するページ番号をページキャッシュへ送信する．

3.3.2 実ページ

図 4 に実ページの構成図を示す．

実ページは，2 次元アレイ状に配置されたセルアレイとロード／ストアのためのエリア選択回路により構成される．

エリア選択回路 ロード／ストアユニットと実ページ間にある 4 つのデータパスはセルがメモリにアクセスするためにある．しかしセルで扱うデータ幅は 8 ビットであり，8 ビット単位でメモリアクセスするのは効率が悪い．そこで PARS アーキテクチャでは，4 つのセルを 1 つのグループとするエリアを用意し，このエリア単位でメモリアクセスを行う．エリア選択回路は，ロード／ストアユニットと実ページ間にある各データパスに対してロード／ストアでアクセスするセルを特定する回路である．

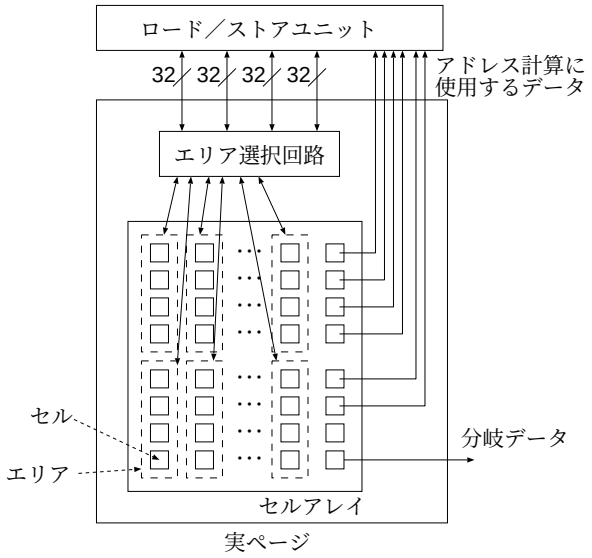

図 4: 実ページの構成図

セルアレイ 図 5 にセルアレイの構造を示す．セルアレイはセルとセル間の通信を担う GRM(Global Routing Matrix) から構成される．同図において正方形のブロックはセルを表し，セルより少し大きめのブロックが GRM を表す．GRM 同士をつなぐ直線は配線である．直線の双方向矢印はセルと GRM 間のデータ転送を表す．点線の単方向矢印はページコントローラやロード／ストアユニットで使用するデータである．

セルと GRM は図 5 のように 9x8 の 2 次元アレイ状に配置される．セルは 8 ビットの演算を実行する．GRM はセル間のデータ転送を実現するため，1 つのセルと隣接する他の GRM に接続されている．

一番右端の列にあるセルは主にメモリアクセスの際に必要となるメモリアドレスの生成や分岐判定データを生成するために使用される．オフセットアドレス，ベースアドレス用データがメモリアクセスの際にどのように使用されるかについては後述する．分岐判定データは前述のページコントローラで分岐の判定に使用するデータである．

制御データを任意のセルから出力できるようにす

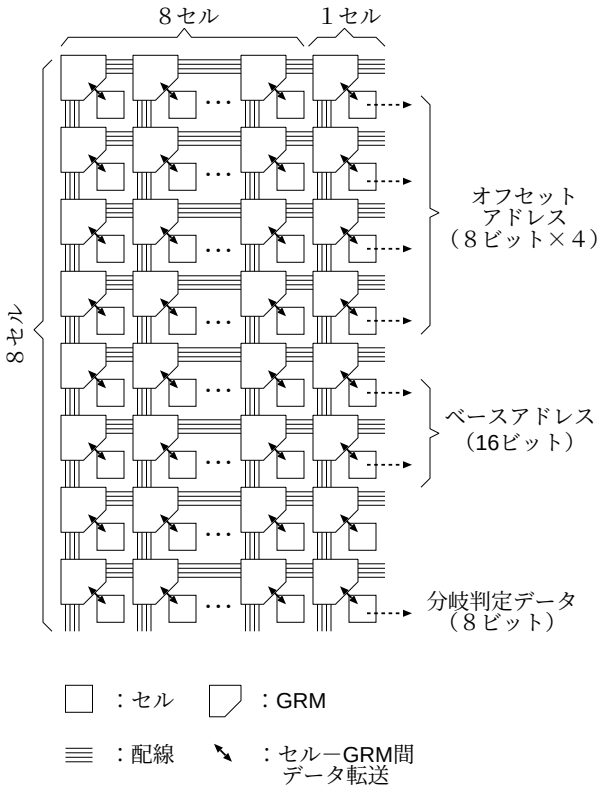

図 5: セルアレイの構造

ることも考えられるが，制御データを出力するセルを固定することにより，必要なハードウェアを簡単化することができる．

GRM は東西南北の各方向にそれぞれ 4 本の出力用の配線を持つ．同様に東西南北の各方向から入力される配線数も 4 本である．1 本の配線のビット幅は 8 ビットとしているため，GRM から各方向に出力できる最大のデータ幅は 32 ビットとなる．

GRM の配線パターンについて説明する．GRM に入力される配線は入力方向以外の全ての方向に出力することができる．例えば東方向から入力された配線は，北方向，西方向，南方向の GRM とセルにデータを転送することができる．ただし，東西南北の各方向で出力として選択できる配線は 1 本のみであり，他の 3 本は固定的な配線となっている．

セル　セルの構造を図 6 に示す．同図において直線は配線を表し，ビット幅が指定されていない信号線は 8 ビットとする．

セルは FU(Function Unit) と 4 つのレジスタから構成される．FU は 8 ビットの演算を実行する．FU の機能は算術演算，論理演算，シフト演算，マルチプレクサである．

FU の出力は必ず一度レジスタに格納される．そ

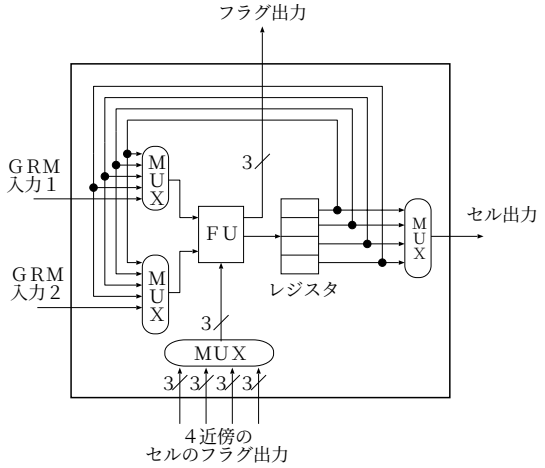

図 6: セルの構造

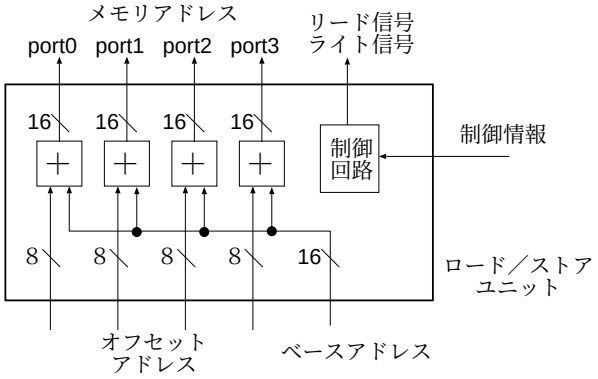

図 7: ロード／ストアユニットの構成図

のため 1 クロックの間に，ある FU の出力を別の FU の入力として使用することはできない．これを言い換えると FU で演算が実行された後，同期をとっていると言える．演算実行後に必ず同期させることにより，コンパイラによる演算のスケジューリングが容易になる．

ただし，FU 間でキャリーを伝搬することはできる．そのキャリーの情報はフラグに含まれる．

3.3.3　ロード／ストアユニット

ロード／ストアユニットは実ページにあるデータとメモリシステムにあるデータの受渡しをする．図 7 にロード／ストアユニットの構成図を示す．

メモリアドレスの計算では，ベースアドレスとオフセットアドレスが加算されることにより，メモリアドレスが計算される．ベースアドレス，オフセットアドレスは図 5 で示した一番右端にあるセルのレジスタの値が使用される．これはプログラムの演算で使用するデータをアドレス計算にも利用できるようにするためである．

表 2: tempo の論理合成結果

	面積 (ゲート数)
ページコントローラ	約 500 ゲート (0.3%)
実ページ	約 16 万ゲート (94%)
ロード／ストアユニット	約 8000 ゲート (5%)
合計	約 17 万ゲート

メモリアドレスの計算では 16 ビットのベースアドレスと 8 ビットのオフセットアドレスが加算されることにより 16 ビットのメモリアドレスが計算される．オフセットアドレスは各メモリポート毎に独立したデータが使用できる．ベースアドレスは 4 つのメモリポートで共通のデータが使用される．

このように tempo ではベースアドレスを共有することでアドレス計算に必要なセル数を削減している．仮に 4 つのメモリポートで共有しない場合，必要なセル数は 8 であり，現在の tempo の場合，必要なセル数は 6 である．

4 tempo の論理設計

本稿では tempo のページコントローラ，実ページ，ロード／ストアユニットの部分を Verilog-HDL を使用して論理設計した．使用したツールは Synopsys Design Compiler の Version 2000.05 である．使用したライブラリは 0.35μm 東大版 EXD 社製ライブラリである．

表 2 に tempo を論理合成した結果を示す．同表において面積は NAND 換算のゲート数で表し，各構成要素が tempo 全体の面積に占める割合を () 内に表す．

5 評価

本節では tempo の有効性を示すため，論理合成後の Verilog-HDL の記述を使用してシミュレーションを行い，性能を評価する．

5.1 評価環境

今回の評価では 4 節で設計した論理合成後の Verilog-HDL 記述を使用して，論理シミュレーションを行う．シミュレーションに使用したツールは Cadence Verilog-XL 3.0 である．

5.2 評価アプリケーション

今回の評価で使用したアプリケーションは DCT の処理である．今回使用した DCT の処理は 1 次元 DCT のアルゴリズムを繰り返し使用する 2 次元 DCT である [3]．本稿では 1 次元 DCT のアルゴリズムを 8 回実行したものを 1 回の DCT の処理と呼ぶ．1 次元 DCT のアルゴリズムは乗算の演算を使用しないアルゴリズムを使用した．DCT の処理はある一定のアルゴリズムに従って手動でマッピングした．

5.3 評価内容

今回の評価では論理シミュレーションにより，DCT の処理を tempo で実行した場合の実行時間を計測する．この時，ページキャッシュ，データキャッシュでキャッシュミスヒットは発生しないと仮定した．また，UltraSPARC-II 360 MHz で同じ処理を実行した場合の実行時間を計測し，tempo の実行時間と比較する．

まず，今回の評価で使用した DCT の処理を C 言語で記述し，その C 言語の記述を UltraSPARC-II 360MHz 上でコンパイルした．コンパイルして得られた実行ファイルを UltraSPARC-II 360MHz 上で実行し，実行時間を計測した．ここでコンパイラには Sun WorkShop Compiler 4.2 を使用し，最適化オプションとして-fast を指定した．

また UltraSPARC-II 360MHz 上の評価ではキャッシュミスによる性能低下が起こる可能性がある．しかし，今回の評価で使用した UltraSPARC-II 360MHz の実行ファイルとデータは搭載されているキャッシュ容量よりも十分に小さかったので，DCT のプログラムの実行中にキャッシュミスが発生する可能性は少ない．

5.4 評価結果

表 3 に UltraSPARC-II 360MHz と tempo の評価結果を示す．tempo 上で DCT の処理を実行した場合の最大動作周波数は 125MHz であった．相対性能は UltraSPARC-II 360MHz を 1 とした場合を示している．

表 3 において面積は NAND 換算のゲート数で UltraSPARC-II 360MHz と tempo の面積を表している．UltraSPARC-II 360MHz の面積は参考文献 [4] に記載してあるトランジスタ数を 2 入力 NAND のトランジスタ数 4 で除算することで求め

表 3: tempo と UltraSPARC-II の評価結果の比較

	実行時間 (ns)	相対性能	面積 (ゲート数)
Ultra SPARC-II 360MHz	660	1	130 万
tempo 125MHz	408	1.6	57 万

た．tempo の面積は 4 節の tempo の論理合成結果と，キャッシュメモリの面積を約 40 万ゲートと見積り，それらを加算することによって面積を求めた．キャッシュメモリの面積の見積り方法はキャッシュメモリに SRAM を使用することを想定し，1 ビットあたり 6 トランジスタ分の面積を消費すると仮定した．

表 3 より tempo の性能向上は UltraSPARC-II 360MHz の 1.6 倍である．UltraSPARC-II 360MHz は 4 命令同時発行のスーパスカラマシンである．それに対し tempo のハードウェアでは演算専用のセルが 64 個あるので，tempo では 8 ビットの演算が最大 64 個並列に動作する．これらのことから，tempo は UltraSPARC-II 360MHz に比べ $64/4 = 16$ 倍の並列性を持っており，その値と比較すると tempo の相対性能が 1.6 倍という数値は小さいと言える．

このような結果になった主な原因として 1 ページあたりの平均セル使用率が低かったことが挙げられる．ここでセル使用率とは実ページ内に存在するセル数のうち，有効な演算をしているセルの割合を表し，1 ページあたりの平均セル使用率とはプログラムに含まれる各ページのセル使用率の平均をとったものである．tempo が DCT の処理を実行した時の 1 ページあたりの平均セル使用率は約 35%であった．

また，UltraSPARC-II 360MHz と tempo の面積を比較すると tempo は UltraSPARC-II360MHz の約 32%の面積である．UltraSPARC-II 360MHz と tempo の性能が同程度で tempo の方が面積が小さい点から，今回評価に用いた DCT の処理においては tempo の方が効率的にハードウェアを使用していると言える．

6 まとめと今後の課題

本稿では PARS アーキテクチャのプロトタイプマシンである tempo の詳細設計について述べた．tempo はページと呼ぶサイズの小さな再構成情報を逐次的に実行することで，実ハードウェアよりサイズの大きい再構成情報を実ハードウェアに構成することができる．

本稿では tempo を論理設計し，DCT の処理を使用して tempo の性能を評価した．その結果，DCT の処理を tempo 上で実行した場合の動作周波数は 125MHz であり，1 サイクルあたりの再構成型演算ユニットの使用率は約 35%であった．

この結果から 1 サイクルあたりの再構成型演算ユニットの使用率をさらに高めることができる分割アルゴリズム，および，アプリケーションの探索を行うことが今後の課題として挙げられる．また今回の設計においては，キャッシュメモリの設計を行っていないので，それを行うことも今後の課題である．

謝辞 本研究の一部は，平成 13 年度文部省科学研究費補助金 (基盤研究 (A)) 継続課題番号 12780238，および (株) 半導体理工学研究センターとの共同研究によるものである．本稿での評価は東京大学大規模集積システム設計教育研究センターの協力で行われた．

参考文献

[1] 谷川一哉，弘中哲夫，吉田典可,「PARS プログラミングモデルと PARS アーキテクチャの提案」，情報処理学会研究報告 2000-ARC-140, pp.37-42, 2000.
[2] 谷川一哉，弘中哲夫，吉田典可,「PARS アーキテクチャに基づくリコンフィギュアラブルコンピュータ」，第 4 回システム LSI 琵琶湖ワークショップ, pp.211-pp.214, 2000.
[3] J. Liang and T. D. Tran, “Fast multiplierless approximation of the DCT with the lifting scheme,” Proc. SPIE Applications of Digital Image Processing XXIII, pp.384-395, 2000.
[4] http://www.sun.com/microelectronics/UltraSPARC-II/specs.html