計算機アーキテクチャ 149－5
（2002．8．22）

バイナリ互換性を持つ粗粒度再構成型アーキテクチャの提案

長山賀与† 谷川一哉† 児島彰‡ 弘中哲夫‡

†広島市立大学大学院 情報科学研究科 情報工学専攻
‡広島市立大学 情報科学部

我々は再構成型アーキテクチャにおけるソフトウェア資産の構築を目標としてPARSアーキテクチャを提案している．しかし従来のPARSアーキテクチャではソースプログラムレベルでのソフトウェア資産の構築は実現しているが，バイナリプログラムレベルの互換性までは達成されていない．そこで本稿では，ハードウェアの仮想化をハードウェアでサポートすることにより，バイナリ互換性を持つPARSアーキテクチャを提案する．さらにハードウェアの仮想化を効率良く実現するためのレジスタ方式を提案し，その有効性を奇遇変換ソートにより検証した．

The Coarse Grain Reconfigurable Architecture with Binary Level Compatibility

Kayo Nagayama†, Kazuya Tanigawa†, Akira Kojima‡, Tetsuo Hironaka‡

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

We have proposed a PARS architecture that aims to reuse the software assets on reconfigurable architecture. The proposed PARS architectures can share the software assets written by high-level programming language, but can not share binary programs, because they do not have binary compatibility. In this paper, we propose the novel PARS architecture with the binary level compatibility by hardware virtualization mechanism in the processor. Further, we propose novel register management methods for realization of hardware virtualization efficiently, and verified the feasibility of the proposed methods by using one parallel sorting algorithm.

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[1]．著者らは再構成型コンピュータを汎用コンピュータとして使用することを目的としソフトウェア資産の構築を可能にするため，I-PARS実行モデル[2]，PARSアーキテクチャ[3]を提案・研究している．

しかし現在のPARSアーキテクチャでは，コンパイラによりサイズに制限のないプログラムを再構成部の大きさに合わせて分割し，それらを逐次的に再構成部に割り当てて実行していくため，プログラムレベルでのソフトウェア資産の構築はできるが，バイナリプログラムレベルの互換性は達成されていない．すなわち，過去のプロセッサ用に作成されたソフトウェアを新たなプロセッサ上で実行しようとした場合に再コンパイルが必要となり，過去のソフトウェア資産をそのまま再利用することができない．

そこで上述の問題に対処する手法として，本研究では構成情報にバイナリ互換性を持たせることにより，再構成部の大きさが異なるプロセッサ間でも同一の構成情報を利用して実行できるPARSアーキテクチャを提案する．またこのバイナリ互換性を持つPARSアーキテクチャを実現するためには，プログラムを限られたレジスタ数で効率良く実行するレジスタ方式が必要となる．そこで本稿では限られたレジスタ数で効率良く実行するレジスタ方式を提案する．

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とその問題点について説明する．3節ではバイナリ互換性を持つPARSアーキテクチャの構成について説明する．4節では限られたレジスタ数で効率良く実行するレジスタ方式の説明をする．5節では奇偶変換ソートを用いて各レジスタ方式において性能評価を行い，その評価結果について考察し，最後に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アーキテクチャは，セルと呼ばれる演算器をベースとした再構成型演算ユニットを持ち，このセルを1サイクルで再構成できるという特徴を持つ．本節では，PARSアーキテクチャの構成と問題点について説明する．

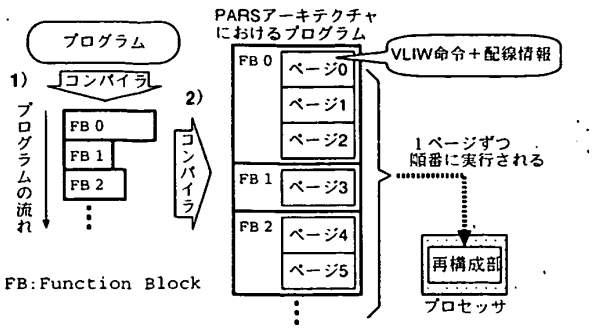

図 1: プログラムの構造と実行方式

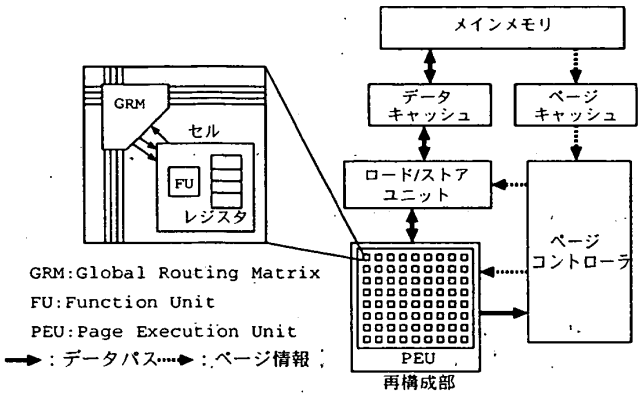

図 2: PARS アーキテクチャの構成図

2.1 プログラムの構造と実行方式

図 1 に PARS アーキテクチャにおけるプログラムの構造と実行方式を示す.

PARS アーキテクチャにおけるプログラムは同図に示すように, 1) コンパイラによってプログラム内の並列に実行可能なすべての演算を機能ブロック (FB:Function Block) に割り当てる. 機能ブロックの大きさに制限はなく, 並列に実行可能な演算の数によって可変である. 2) さらにコンパイラによって各機能ブロックを各プロセッサの再構成部の大きさに合わせて分割する. PARS アーキテクチャでは, この各プロセッサの再構成部の大きさに合わせて分割したものをページと呼ぶ. ページは従来のコンピュータでの命令と配線情報からなる. PARS アーキテクチャでは順にページを実行していくが, 分岐命令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, ページ毎にページ番号が割り振られている.

このプログラムを構成する各ページを逐次的にプロセッサ内の再構成部に割り当てることによりプログラムを実行する.

2.2 PARS アーキテクチャの構成

図 2 に PARS アーキテクチャの構成図を示す.

同図において実線の矢印はデータの流れを, 点線の矢印はページの情報の流れを表す. 以下, 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.

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

再構成部 1 つの PEU から構成される.

- **PEU(Page Execution Unit)** セルと GRM が 2 次元アレイ状に並んでおり, この部分でハードウェアの再構成を行い実行する. PEU の大きさは可変である.
- **セル** FU と 4 つのレジスタから構成される.
- **FU(Function Unit)** 8bit の演算を行う.
- **GRM(Global Routing Unit)** セル間の通信を行う. セル間の通信を行うため, 1 つのセルと隣接する他の GRM に接続されている.

2.3 問題点

2.1 節で説明したように PARS アーキテクチャでは, プログラムを各プロセッサの PEU の大きさに合わせてページに分割するため, 生成されたページは各プロセッサの再構成部の大きさに依存したものとなる. そのため, PEU の大きさが異なるプロセッサ間で同一のページを利用して実行することができず, 各プロセッサの PEU の大きさが異なる度に同じプログラムでもコンパイルし直さなければならないといった問題がある.

これでは, 過去のプロセッサ用に作成したソフトウェアを新たなプロセッサ上で実行しようとした場合に再コンパイルが必要となり, 過去に作成したソフトウェアをそのまま再利用することができない.

そこでこの問題に対処する手法として, バイナリ互換性を持つ PARS アーキテクチャを提案する.

3 バイナリ互換性を持つ PARS アーキテクチャの提案

バイナリ互換性を持つ PARS アーキテクチャでは, どんな大きさの再構成部を持つプロセッサに対しても同じバイナリプログラムで実行することができるアーキテクチャを目指している. 本節では, バイナリ互換性を持つ PARS アーキテクチャの構成と予備評価, 限られたレジスタを効率良く使用して実行するレジスタ方式について説明する.

3.1 プログラムの構造と実行方式

図 3 にバイナリ互換性を持つ PARS アーキテクチャにおけるプログラムの構造と実行方式を示す. バイナリ互換性を持つ PARS アーキテクチャにおけるプログラムは同図に示すように, 1) コンパイ

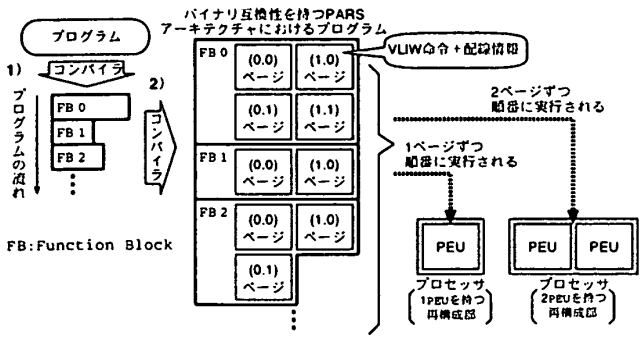

図 3: プログラムの構造と実行方式

ラによってプログラムを機能ブロックに割り当てた後，2) その各機能ブロックを固定の大きさに分割する．バイナリ互換性を持つ PARS アーキテクチャでは，この固定の大きさに分割したものをページと呼ぶ．このページにはページ毎に座標が割り振られている．

このプログラムを構成する各ページを各プロセッサ内の再構成部の大きさに合わせてフェッチし，逐次的に再構成部に割り当てることによりプログラムを実行する．以下，PARS アーキテクチャとバイナリ互換性を持つ PARS アーキテクチャの相違点について説明する．

相違点

バイナリ互換性を持つ PARS アーキテクチャでは、コンパイラによってプログラムを機能ブロックに割り当てた後、その各機能ブロックを各プロセッサの再構成部の大きさに合わせて分割するのではなく，固定の大きさのページに分割し，その各ページを各プロセッサ内の再構成部の大きさに合わせてフェッチし実行することで，再構成部の大きさが異なるプロセッサ間でも同一のページを利用して実行することができる．このようにしてバイナリ互換性を実現する．

3.2 バイナリ互換性を持つ PARS アーキテクチャの構成

バイナリ互換性を持つ PARS アーキテクチャの構成は図 2 に示した PARS アーキテクチャの構成とほぼ同様である．以下，PARS アーキテクチャとバイナリ互換性を持つ PARS アーキテクチャの相違点について説明する．

相違点

PARS アーキテクチャと異なる点は，再構成部と PEU の概念である．PARS アーキテクチャでは PEU の大きさは可変であり，再構成部は 1 つの PEU から構成されるというものであった．バイナリ互換性を持つ PARS アーキテクチャでは，PEU の大きさを固定し，その固定の大きさを持つ PEU を 2 次元アレイ状に並べたものを再構成部とする．つまり，バイナリ互換性を持つ PARS アーキテクチャでは，再

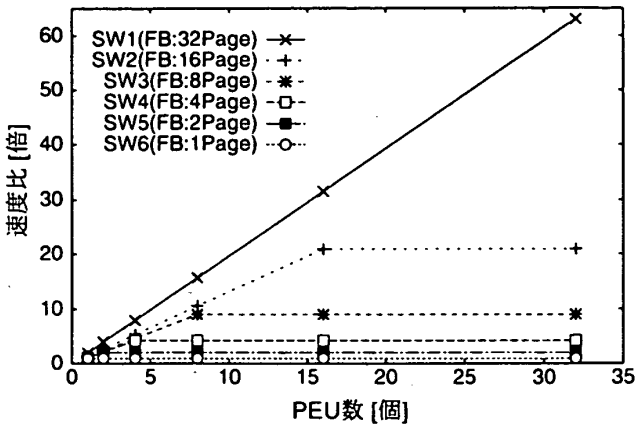

図 4: FB と PEU 数を変化させたときの速度比

構成部は複数の PEU から構成される．1 つの PEU は 1 つのページを実行する．つまり，図 3 に示すように各プロセッサ内の再構成部に存在する PEU 数に合わせてページをフェッチし実行する．

3.3 予備評価 1

評価内容

予備評価 1 では，プロセッサ内に充分多くのレジスタが存在する場合において，機能ブロックの大きさとプロセッサ内の PEU 数を変化させたときの処理速度について評価する．今回の予備評価で使用したアプリケーションは並列ソートアルゴリズムの奇偶変換ソート [4] である．ソートの入力データ数は 1024 個とし，すでにレジスタにロードされていることを前提とする．ページは 8 × 8 のセルからなり，それぞれのページは 1 サイクルで実行されるものとする．レジスタは 1024 個のデータのソートを実行するのに充分な程あるものとする．このような評価環境において，奇偶変換ソートのアルゴリズムに従って手動でマッピングを行い，バイナリ互換性を持つ PARS アーキテクチャで実行した場合のサイクル数を試算する．

評価結果

図 4 に評価結果を示す．同図において横軸は再構成部で実行できるページ数 (PEU 数) を，縦軸は 1 個のページからなる機能ブロックを 1 ページずつ実行したときの速度を 1 とした場合の速度比を表す．また，SW1，SW2，SW3，SW4，SW5，SW6 はそれぞれ機能ブロックが 32，16，8，4，2，1 個のページからなるということを表す．同図より，PEU 数が大きくても機能ブロック内のページ数が少ない部分では性能が一定になっていることから，PEU に無駄なくページを割り当てることが重要であるということがわかる．また同図よりレジスタが充分にある場合では，PEU 数に比例して性能向上しており性能のスケーラビリティが高いといえる．しかし大きな機能ブロックを実行するためには多くのページを実行する必要があり，この多くのページを実行するためには通常各ページの実行結果を保存しておく

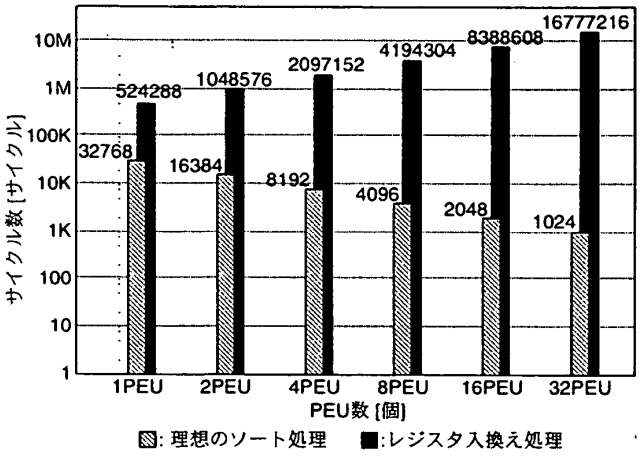

図 5: レジスタの入換えを行う場合

ためにページ数に比例した数のレジスタが必要となる．つまり大きな機能ブロックを実行するためには，膨大な数のレジスタが必要になる．しかし，ハードウェアでは無制限に多くのレジスタを用意することは不可能である．そこで，大きな機能ブロックを実行するためにはページの入換えと同時にレジスタの入換えが必要になる．

3.4 予備評価 2

評価内容

そこで予備評価 2 では，実行するページが変わる度に各ページ用のレジスタをすべて入換えるという場合において，PEU 数を変化させたときの実行にかかるサイクル数について評価を行う．評価内容は予備評価 1 と同様である．ただし，レジスタは 1 つのセルに対して 4 個とし，メモリバンド幅は 32 ビットのポートが 4 つあるものとする．これはつまり，1 サイクルで 8 ビットのデータを 16 個ロード/ストアできることを示す．

評価結果

図 5 に評価結果を示す．同図は機能ブロックが 32 個のページからなる場合においての評価結果である．同図において縦軸は実行にかかるサイクル数を log scale で，横軸はプロセッサ内の PEU 数を表す．また，斜線で示した棒グラフはレジスタが充分にあるという理想の状態でソートを行った場合のサイクル数を，黒で示した棒グラフは実行するページが変わる度に各ページ用のレジスタの入換えにかかるサイクル数を表す．また，レジスタが充分にあるという理想の状態では PEU 数に比例して性能向上しているが，実行するページが変わる度に各ページ用のレジスタを入換える場合では，PEU 数が増加するにつれて性能が急激に低下してることがわかる．これは PEU 数が増加するにつれて入換えるレジスタの数が増加し，メモリアクセスが頻繁に発生するためである．これでは高い性能のスケーラビリティを実現することができない．

そこで限られたレジスタを効率良く使用して実行するレジスタ方式が必要となる．

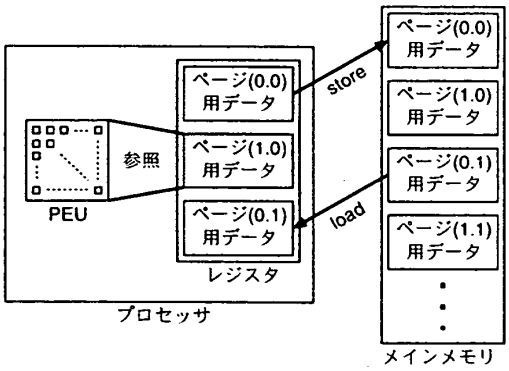

図 6: 総入換えレジスタ方式

4 レジスタ方式の提案

限られたレジスタを効率良く使用して実行するレジスタ方式として，総入換えレジスタ方式と部分入換えレジスタ方式を提案する．以下，各レジスタ方式について説明する．

4.1 総入換えレジスタ方式

図 6 に総入換えレジスタ方式の構成を示す．総入換えレジスタ方式は，実行するページが変わる度にレジスタの内容をメモリとやり取りし入れ換えていく方式である．これは予備評価 2 で行った実行するページが変わる度にレジスタを入れ換えるという方法と似ているが，総入換えレジスタ方式ではレジスタを複数セット用意した上で演算の実行を行いながらレジスタの内容を入れ換えていく．こうすることにより，レジスタの入れ換えに必要なサイクル数を隠蔽することができる．図 6 は，1PEU からなる再構成部を持つプロセッサにおける総入換えレジスタ方式の動作を表している．再構成部では，1 サイクル前でレジスタにロードされたページ (1.0) 用のデータを参照してページ (1.0) の実行を行いながら，ページ (0.0) 用のデータをストアしページ (0.1) 用のデータをレジスタにロードする．

総入換えレジスタ方式は再構成部内の PEU 数が多い場合には，多くのレジスタが必要になり，また多くのレジスタのデータを 1 サイクルで入換えなければならないため大きなメモリバンド幅が必要となる．

メリット・デメリット

総入換えレジスタ方式は，単純にページが変わる度にレジスタの内容を入れ換えるため構造が単純である．しかし，逆にページが変わる度にレジスタの内容を入換えなければならないためメモリ転送が多くなる．また，異なるページ間でレジスタを参照しようとした場合，目的のレジスタが格納されているメモリアドレスを計算し，メモリから目的のデータをロードする必要があるため，異なるページ間でのレジスタ参照は困難である．また，再構成部内の

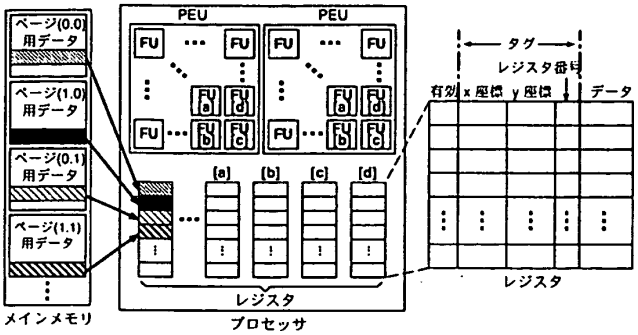

図 7: 部分入換えレジスタ方式

PEU 数が多い場合には，多くのレジスタと大きなメモリバンド幅が必要である．

4.2 部分入換えレジスタ方式

図 7 に部分入換えレジスタ方式の構成を示す．部分入換えレジスタ方式は，よく使用するレジスタをプロセッサ内のレジスタに置いておくキャッシュのような方式である．部分入換えレジスタ方式では，異なるページ間のデータ転送をレジスタを介して行うため，特定の FU がレジスタを共有するという構造になっている．図 7 に示される FU[a] はレジスタ [a] を，FU[b] はレジスタ [b] を，FU[c] はレジスタ [c] を，FU[d] はレジスタ [d] を共有する．他に示されていない FU に関しても同様に特定の FU がレジスタを共有する．PARS アーキテクチャでは 1 つの FU と 1 セットのレジスタが対応していたが，この部分入換えレジスタ方式では，複数の FU が 1 セットのレジスタを共有している．また，複数のページ用のデータをプロセッサ内のレジスタに保持しているため，各レジスタにはページの座標，レジスタ番号をタグとしてデータに付加して格納する．

部分入換えレジスタ方式では，データをレジスタに格納するとき空きのレジスタがない場合や参照したいデータがアクセスしたレジスタ内に存在しない場合にミスが発生する．ミスが発生すると，まずマシン全体をストップさせどこのレジスタでミスが発生したか判定する．ミスが発生したレジスタを発見すると LRU によって置き換えるレジスタを選択し，選択されたレジスタのデータを退避する．次にメモリアドレスを計算し選択されたレジスタにデータを読み出し，実行を再開する．このミスが多発する場合にはミスの処理に時間がかかり性能低下に繋がる．

メリット・デメリット

部分入換えレジスタ方式は，複数ページ用のデータをプロセッサ内のレジスタに保持しているため，参照しようとするデータがプロセッサ内のレジスタに存在していれば，メモリ転送は少なる．また異なるページ間でのレジスタ参照も簡単になる．しかしミスが多発する場合にはミスの処理に時間がかかり性能低下する．また，部分入換えレジスタ方式はタグによる管理が必要になるため構造が複雑である．

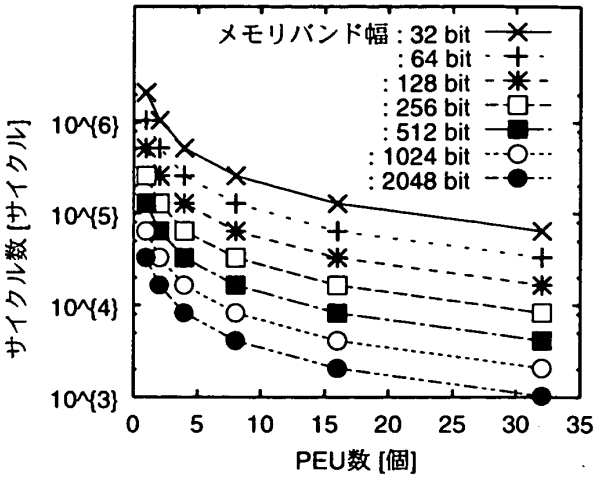

図 8: PEU 数とメモリバンド幅を変化させたときのサイクル数

5 評価

本節では，各レジスタ方式の有効性を評価するために性能評価を行う．

5.1 総入換えレジスタ方式

評価内容

総入換えレジスタ方式では，レジスタの内容をメモリとやり取りするためのポートのメモリバンド幅とプロセッサ内の PEU 数を変化させたときの実行にかかるサイクル数について評価を行う．また，総入換えレジスタ方式を実現するためにはプロセッサ内の PEU 数を変化させた場合に必要なレジスタ数について評価を行う．今回の評価で使用したアプリケーションは予備評価と同様に奇偶変換ソートである．ソートの入力データ数は 1024 個とし，すでにレジスタにロードされていることを前堤とする．ページは 8 × 8 のセルからなり，それぞれのページは 1 サイクルで実行されるものとする．レジスタは 1 つの FU に対して 4 個とする．

評価結果

図 8 にメモリバンド幅と PEU 数を変化させたときの実行にかかるサイクル数の評価結果を，表 1 に必要なレジスタ数の評価結果を示す．図 8 において縦軸は実行にかかるサイクル数を，横軸はプロセッサ内の PEU 数を表す．同図では各メモリバンド幅において実行にかかるサイクル数を示す．図 8 と表 1 から，総入換えレジスタ方式では，多くのレジスタと大きなメモリバンド幅を用意することが性能向上の必要条件である．メモリバンド幅が小さいとレジスタの内容の入換えに時間がかかり性能が低下する．

5.2 部分入換えレジスタ方式

評価内容

部分入換えレジスタ方式では，レジスタ数を変化させたときのミスの発生回数について評価を行う．

表 1: 必要となるレジスタ数

PEU数	必要なレジスタ数
1	768
2	1536
4	3072
8	6144
16	12288
32	24576

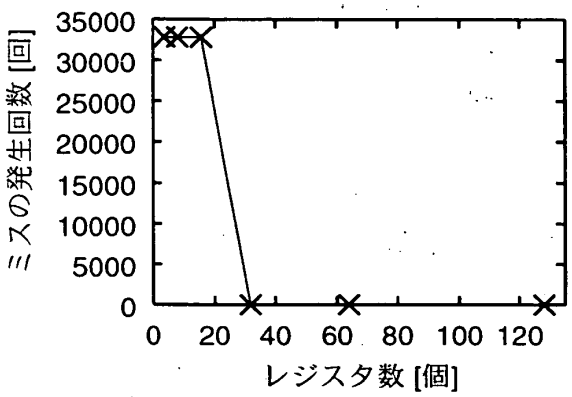

図 9: レジスタ数を変化させたときのミスの発生回数

評価に使用したアプリケーション，入力データ数や評価環境は総入換えレジスタ方式と同様である．ただし，レジスタ数は複数の FU に対して 4，8，16，32，64，128 個とする．

評価結果

図 9 にレジスタ数を変化させたときのミスの発生回数の評価結果を示す．同図において縦軸はミスの発生回数を，横軸は複数の FU に対するレジスタ数を表す．プロセッサ内の PEU 数を変化させてもミスの発生回数は変化せず，ミスの発生回数は複数の FU に対するレジスタ数に左右される．同図から，アプリケーションを実行するのに十分なレジスタが存在する場合には，ミスは発生せず理想の状態に近づくことがわかる．またレジスタが不足する場合には，ミスが連続的に発生してしまい性能低下が大きいことがわかる．

各レジスタ方式において理想の状態と同じ性能を達成している場合の必要なレジスタ数を比較する．総入換えレジスタ方式では 24576 個のレジスタが必要であり，部分入換えレジスタ方式では 1 ページのセル数×レジスタ数＝ 8 × 8 × 32 ＝ 2048 個のレジスタが必要である．よって，部分入換えレジスタ方式では総入れ換えレジスタ方式と比較して 2048/24576 ＝ 1/12 のレジスタ数で理想の状態と同じ性能を達成していることがわかる．

6 まとめと今後の課題

本稿では，バイナリプログラムレベルでのソフトウェア資産の構築を可能とするバイナリ互換性を持つ PARS アーキテクチャを提案した．またこれを実現するために単純にレジスタの入換えを行う総入換えレジスタ方式と効率的にレジスタの入換えを行う部分入換えレジスタ方式を提案し，その有効性を評価するため奇偶変換ソートを用いて性能評価を行った．

その結果，部分入換えレジスタ方式では総入れ換えレジスタ方式と比較して 1/12 のレジスタ数で理想の状態と同じ性能を達成していることがわかった．

今後の課題としては，さらに各レジスタ方式の有効性を評価するために DCT や FEAL など他アプリケーションについても評価を行い，Verilog-HDL を用いたバイナリ互換性を持つ PARS アーキテクチャの詳細設計を行う．

謝辞

本研究を進めるにあたり，貴重な御意見を頂いた広島市立大学情報科学部の北村俊明教授に感謝致します．本研究の一部は武田計測先端知財団の協力のもとで行われた．

参考文献

[1] 末吉敏則：“Reconfigurable Computing System の現状と課題-Computer Evolution へ向けて-”，信学技報，CPSY96-91, pp. 111–118, 1996.

[2] Kazuya Tanigawa,Testuo Hironaka, Akira Kojima,and Noriyoshi Yoshida : “A Generalized Execution Model for Programming on Reconfigurable Architectures and an Architecture Supporting the Model”, http://www.lirmm.fr/fpl2002/home.html, September 2002 (in printing)

[3] 谷川一哉，吉田哲生，児島彰，弘中哲夫，吉田典可：“PARS アーキテクチャの詳細設計に関する一考察”，情処研報，2001-ARC-144, pp. 31–36, 2001.

[4] S. アクル著，阿江忠，山下雅史，相原玲二訳：“並列ソーティング・アルゴリズム”，啓学出版，pp.46–49，1988.