計算機アーキテクチャ　147−13
ハイパフォーマンスコンピューティング　89−13
（2002．3．7）

二電子積分計算専用プロセッサ・アーキテクチャ

中村健太[†1]　波多江秀典[†1]　原田宗幸[†1]
上原正光[†2]　佐藤比佐夫[†2]　小原繁[†3]
本田宏明[†3]　長嶋雲兵[†4]
稲富雄一[†5]　村上和彰[†1]

筆者らは，非経験的分子軌道計算を高速に処理することを目的とする，二電子積分計算専用プロセッサの開発を行っている．二電子積分計算のアルゴリズムとして採用する小原のアルゴリズムの特徴を活かすことで高速処理が可能となる．本稿では，二電子積分計算専用プロセッサ・アーキテクチャの概要と全体構成について述べ，そのうち漸化計算エンジンの構成法の検討と性能の評価を行う．

Processor Architecture for Molecular Orbital Calculation

Kenta NAKAMURA ,[†1] Hidenori HATAE ,[†1]
Muneyuki HARADA ,[†1] Masamitsu UEHARA ,[†2]
Hisao SATO ,[†2] Shigeru OBARA ,[†3] Hiroaki HONDA ,[†3]
Unpei NAGASHIMA ,[†4] Yuichi INADOMI [†5]
and Kazuaki MURAKAMI[†1]

We are developing a custom processor for *ab initio* Molecular Orbital Calculation to reduce the calculation time. Using characterization of two-electron integrals in the "Obara method," it is possible to reduce the calculation time. This paper describes an outline of processor architecture for Molecular Orbital Calculation. We propose and evaluate organizing method for Recursive Function Engine which a part of custom processor.

1. はじめに

非経験的分子軌道法は，一番近似の粗いハートリー・フォック法を用いた場合でも，用いる基底関数の 4 乗に比例する演算量と補助記憶量を必要とする[1)]．そこで，我々「科学技術計算専用ロジック組込型プラットフォーム・アーキテクチャに関する研究」グループでは，非経験的分子軌道法による分子軌道計算を高速に実行する専用計算機システムの開発に現在取り組んでいる[2)]．

非経験的分子軌道計算において最も時間を要するのが，二電子積分を計算し，フォック行列を生成する部分であるため，大規模計算を高速化する際にはこの部分の並列化，高速化が重要である．現在アーキテクチャの検討を行っている二電子積分計算専用プロセッサでは，非経験的分子軌道法の全計算時間の 90%以上を占める，この二電子積分計算を高速に処理することを目的とする．

二電子積分計算は小原のアルゴリズム[3)] を適用することにより漸化計算で表され，複数の積和演算を並列的に計算することにより高速化できる．この小原のアルゴリズムは大きく分けて，命令レベル並列性が少なく，浮動小数点除算，開平逆数演算，指数関数演算がクリティカル・パスとなっている初期積分計算部分と，多くの命令レベル並列性を持つ漸化計算部分との 2 つの部分に分けられる[4)]．専用プロセッサは，この性質の大きく異なる両方の部分に最適化され，その両方を高速に処理できることが求められる．

そこで本稿では，小原のアルゴリズムの特徴に合わせて，初期積分計算部分を専門に処理する初期積分計算エンジンと，漸化計算部分を専門に処理する漸化計算エンジンの 2 つのエンジンから成るプロセッサ・

†1 九州大学　Kyushu University
†2 セイコーエプソン 株式会社　Seiko Epson Corp.
†3 北海道教育大学　Hokkaido University of Education
†4 産業技術総合研究所
National Institute of AdvancedIndustrial Science and Technology
†5 株式会社 富士総合研究所　Fuji Research Institute Corp.

Email: ehpc-lsi@star.fuji-ric.co.jp

アーキテクチャを提唱し，その上で各エンジンの構成法についての検討を行う．以下，第 2 章では非経験的分子軌道計算と小原のアルゴリズムの特徴について述べる．次に第 3 章では 2 つのエンジンから成る専用プロセッサ・アーキテクチャの概要について述べる．第 4 章では漸化計算エンジンの構成の検討とその評価を行う．

2. 二電子積分計算アルゴリズム

非経験的分子軌道法による分子軌道計算において最も計算時間を費やす二電子積分計算は，小原のアルゴリズムにより計算する．本章では小原のアルゴリズムの特徴について述べる．

2.1 非経験的分子軌道法

非経験的分子軌道法の解法には，ハートリー・フォック法（HF 法）を用いる[5]．HF 法を用いた非経験的分子軌道計算において，計算時間の 95〜99%を占めるのが二電子積分計算を求め，フォック行列を生成するステップである[1]．また二電子積分計算は，基底関数の数 N の 4 乗に比例する計算量が必要であるため，N が多くなる（系のサイズが大きくなる，もしくは精度の高い基底関数を使う）に従って計算時間の割合が大きくなる傾向にある．したがって非経験的分子軌道法による分子軌道計算の高速化を実現するには，いかに二電子積分計算を並列で高速に処理することが重要である．

2.2 小原のアルゴリズムの特徴

二電子積分計算のアルゴリズムはいくつか知られているが，本プロジェクトでは小原のアルゴリズム[3]を採用する．小原のアルゴリズムの特徴は，まず入力データからいくつかの初期パラメータを計算し，積分タイプ☆(ss,ss) 型の二電子積分（初期積分）を求めたあと，それらを用いて目的の二電子積分まで漸化計算により求める点にある．したがって小原のアルゴリズムは図 1 に示すように，初期積分計算部分と漸化計算部分の 2 つに大きく分けられる．

2.2.1 初期積分計算部分

初期積分計算は図 1 に示すように 4 重のループ構造

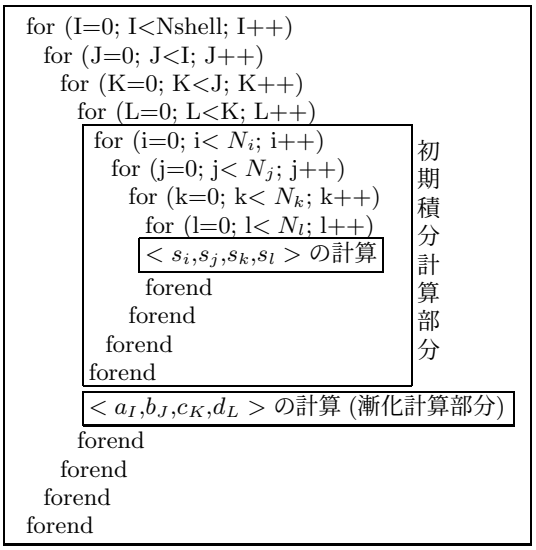

図 1 小原のアルゴリズムのプログラム構造

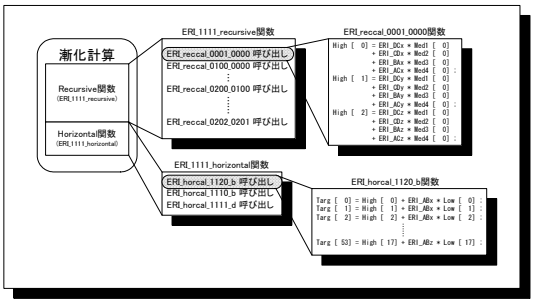

図 2 漸化計算部分の例（積分タイプ (pp,pp)）

となる．このループ 1 回あたりの演算数は非常に少なく，命令レベル並列性は低い．さらに，浮動小数点除算，開平逆数演算，指数関数演算という複雑な算術演算を含んでおり，かつそれらの演算がクリティカル・パス上に存在する[4]．

2.2.2 漸化計算部分

漸化計算部分は図 1 から見て取れるように，初期積分計算部分が完了したのち，初期積分の計算結果をもとにして最終的に求める二電子積分の値を計算するステップである．そして漸化計算部分は図 2 に示すようには漸化計算関数（ERI_recursive 関数）と水平移行関数（ERI_horizontal 関数）から成る．またそれぞれの関数が積和演算の集合である ERI_reccal 関数または，ERI_horcal 関数（サブ関数）を呼び出す階層構造を持っている．

3. 専用プロセッサ・アーキテクチャの概要

第 2.2 節で述べたように，二電子積分計算は大きく

☆ 二電子積分を決定する 4 つのガウス型関数それぞれの軌道量子ベクトルを a, b, c, d とする．ある軌道量子ベクトル μ において，ベクトル 3 成分 (x 成分，y 成分，z 成分) の和を軌道量子数と呼ぶ．また，軌道量子数が 0，1，2，$\cdots$，の軌道をそれぞれ s 軌道，p 軌道，d 軌道，$\cdots$，と呼ぶ．二電子積分計算はその計算を決定する 4 つの軌道量子ベクトル a，b，c，d ごとの軌道量子数が表す軌道を用いて (pp,ss)，(dd,ss) のように表記される．このように軌道量子数により識別した積分のことを積分タイプと呼ぶ[4]．

性質の異なる初期積分計算と漸化計算からなり， 専用プロセッサはこの両方を高速に処理することが求められる．本章では，二電子積分計算を高速に処理するプロセッサ・アーキテクチャの概要について述べる．

3.1 戸川の評価

文献 4) の中で戸川が行った評価では，専用プロセッサが初期積分計算部分と漸化計算部分の両方を単一の VLIW（Very Long Instruction Word）プロセッサで処理することを計画していた．その上で，

- 浮動小数点除算，開平逆数演算，指数関数演算を高速に処理する専用演算器を設けることで初期積分計算部分を高速化する
- 積和演算器を多数搭載し，内在する並列性を活用することで漸化計算部分を高速化する

という方針に基づき，専用演算器と多数の積和演算器，および，全ての演算器に共有されたマルチポート・レジスタファイルから成る VLIW アーキテクチャのプロセッサモデルについての評価を行っている．

しかしながら，第 2.2 節で述べたように，対象とする二電子積分計算では初期積分計算と漸化計算の性質が大きく異なり，特に 2.2.1 節で述べたように初期積分計算部分の命令レベル並列性が低いため，演算器を多数搭載しても十分な性能向上が得られない．また，単一の VLIW プロセッサで処理するには以下のような VLIW 特有の問題がある．

- レジスタファイルに非常に多くのポートが必要となり，マルチポート・レジスタファイルの面積および遅延により専用プロセッサの性能の上限が抑えられる．
- 命令語長が長いことから命令供給が問題となる．
- コードサイズが大きくなることから，内部メモリに命令を持たせるのに適していない．

これらの問題を解消し，目標とする高い処理能力を得ることができるプロセッサ・アーキテクチャの検討を行った．

3.2 設 計 方 針

専用プロセッサが汎用プロセッサと違う点は言うまでもなく，処理する対象が限定されており，プロセッサ・アーキテクチャを計算対象の特徴に合わせて最適化できる点である．

第 2 章で述べたように，小原のアルゴリズムは初期積分計算部分と漸化計算部分という 2 つの部分に分けることができる．専用プロセッサが二電子積分計算を高速に処理するために

- 初期積分計算部分には専用演算器を用いる
- 漸化計算部分は内在する並列性を活用する

という基本方針には変わりない．

しかし単一の VLIW プロセッサで処理を行う場合には，第 3.1 節で述べたような問題があり十分な性能向上が望めないことから，専用プロセッサのアーキテクチャを図 3 に示すように，初期積分計算エンジンと漸化計算エンジンという 2 つのエンジンに分割した構成とし，初期積分計算部分と漸化計算部分をそれぞれ専用エンジンで実行することで 2 つの計算を同時にオーバーラップさせて実行することを検討している．

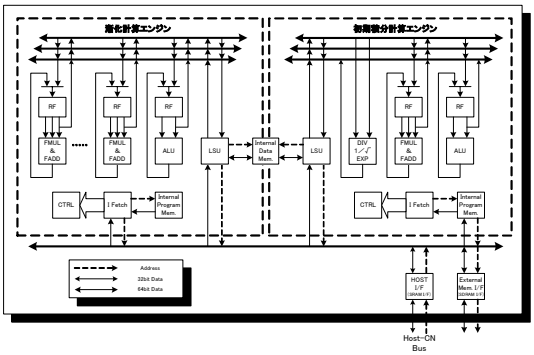

図 3 専用プロセッサ・アーキテクチャのブロック図

3.3 全 体 構 成

現在検討中のアーキテクチャは大きく分けて

- 初期積分計算エンジン
- 漸化計算エンジン
- 内部データメモリ（Internal Data Mem.）
- 対ホスト間インタフェース（HOST I/F）
- 対メモリ間インタフェース（External Mem. I/F）

から成る．対象とする演算は倍精度の浮動小数点とし，各演算器，メモリ，インタフェースもそれぞれ 64 ビット幅となる．また単精度の浮動小数点はサポートしない．

3.3.1 初期積分計算エンジン

小原のアルゴリズムの初期計算部分を専用に処理する計算エンジンであり，

- 浮動小数点除算，開平逆数演算，指数関数演算を高速に処理する専用演算器（DIV, $1/\sqrt{\ }$, exp）
- 浮動小数点積和演算器（FMUL & FADD）
- ロード・ストアユニット（LSU）
- 内部プログラムメモリ（Internal Program Mem.）

などから成る．

3.3.2 漸化計算エンジン

漸化計算部分を専用に処理する計算エンジンであり，

- 複数の浮動小数点積和演算器（FMUL & FAD）
- ロード・ストアユニット（LSU）

● 内部プログラムメモリ（Internal Program Mem.）などから成る．

3.3.3 内部データメモリ

初期積分計算および漸化計算で必要となるデータを保持するデータメモリ．2 バンク構成となっており，初期積分計算エンジンと漸化計算エンジン間で共有される．また，ダブルバッファ処理を行い，両エンジン間のデータの授受を行う役割も兼ねる．

3.3.4 動 作 原 理

図 1 から見て取れるように，小原のアルゴリズムでは初期積分計算が完了したのち漸化計算が行われる．また，それぞれの二電子積分計算間でのデータ依存は存在しないことから，個々の二電子積分計算は並列に実行可能である．この特徴を活かして，専用プロセッサは図 4 のように初期積分計算と漸化計算をオーバーラップさせて処理する．

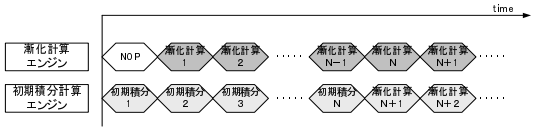

図 4 初期積分計算部と漸化計算部の実行の様子

4. 漸化計算エンジンにおける計算手法の検討

第 3 章では，小原のアルゴリズムの特徴を活かすプロセッサ・アーキテクチャの全体構成について述べた．しかしながら第 3.1 節で述べた問題点のうち，VLIW プロセッサ特有の課題は依然として残っている．

本章ではこの課題を解消しつつ，漸化計算部分に内在する並列性を活かす漸化計算エンジンの構成について評価を行う．

4.1 従来の手法および新手法の提案

戸川の評価における漸化計算部分の処理は，多数のサブ関数呼出しから成る漸化計算をインライン展開し，命令レベル並列性を高めることにより高速化を図っていた．この手法では命令レベル並列性の高い，d 軌道を含むような積分タイプを処理する場合には高い処理能力を得ることができる．一方で漸化計算部分が持つ命令レベル並列性は積分タイプによって異なり，s 軌道と p 軌道のみから成る命令レベル並列性の低い積分タイプを処理する場合には演算器を十分活用できない場合がでてくる．また，命令供給やコードサイズの問題という VLIW プロセッサ特有の課題を抱えている．

そこで，漸化計算部分が図 2 に示したように，関数呼出しの階層構造を持つことを利用し，マイクロ命令を使ったコード圧縮を行う手法を提案する．また，その手法が活用する並列性の違いにより以下の 3 手法に分類し，評価を行った．

手法 FS　命令レベル並列性のみ
手法 FP　関数間 ＋ 命令レベル並列性
手法 SP　ステートメント間 ＋ 命令レベル並列性

4.1.1 サブ関数間逐次処理（手法 FS）

漸化計算部分ではサブ関数を繰り返し呼び出す構造をしている．このサブ関数の種類は ERI_reccal 関数が 26 種類，ERI_horcal 関数も 25 種類程度と多くないことから，サブ関数内部の処理をマイクロ命令化し内部メモリに保持することが可能である．その上で，マイクロ命令化されたサブ関数を呼び出す漸化計算関数と水平移行関数のみをマクロ命令とし，外部メモリから逐次的に供給することで，先述した命令供給とコードサイズの課題を解消できる．これを “手法 FS” と呼ぶ．

本手法では，1 つのサブ関数内の命令レベル並列性のみしか活用しないことから，演算器の利用効率が上がらないという，従来の手法からの問題点が残る．また，マルチポート・レジスタファイルが抱える問題も未解決である．

4.1.2 サブ関数間並列処理（手法 FP）

漸化計算部分に内在する並列性は命令レベル並列性ばかりではない．サブ関数間にもデータ依存関係が無く，並列実行可能なものが多数存在する．このサブ関数間並列性を活用するのが “手法 FP” である．

手法 FP では関数呼出しを行うマクロ命令を並列に処理し，対応するマイクロ命令を各クラスタに処理させる．ここで，クラスタとは 1 個以上の積和演算器と，1 個以上のロード・ストアユニット，およびそれらのユニットのみからアクセスできるレジスタファイルから構成される制御単位である．

この手法では，1 つのサブ関数を割り当てる演算器の個数が手法 FS よりも少なくなるため，演算器の利用効率が向上することが期待できる．また，レジスタファイルのポート数も手法 FS と較べ削減することができる．

4.1.3 ステートメント間並列処理（手法 SP）

手法 FP ではサブ関数間の並列性を利用したが，サブ関数は多数のステートメントの集合からなり，そのステートメント間にも並列性が存在する．そこで，ステートメントの出現パターンから規則性を抽出し，16 通りにグループ分けしたものをマイクロ命令とする手法が “手法 SP” である．

この手法も FP と同様に演算器の利用効率向上や，

レジスタファイルのポート数削減が期待できる．さらにFPと較べ，マイクロ命令1個あたりの命令サイズが小さく，種類も少ないため，マイクロ命令を保持するのに必要な内部メモリサイズを小さくすることができる．

4.2 各手法の性能比較評価

前節で述べた各手法に対しての評価をおこなう．また，その結果より，漸化計算エンジンの最適な構成法を考察する．

4.2.1 評価方法

評価方法に関しては，漸化計算部分の実行サイクル数が最小となるように，クリティカル・パス法を拡張したリスト・スケジューリング[6]を行い，その結果による実行サイクル数を求めた．

4.2.2 評価モデル

評価を行う漸化計算エンジンのモデルとして，表1に示すようなレイテンシを持つ積和演算器（M&A）と，ロード・ストアユニット（LSU），およびレジスタファイルから成る計算エンジンについて考える．先に述べた手法について表2に示す数の機能ユニットを有する各モデルについて評価する．ただし，表中のModelは先頭の要素から，手法名，クラスタ数，1クラスタあたりの積和演算器数とロード・ストアユニット数を表すものとする．

表1 各機能ユニットのレイテンシ

	M&A	LSU
Issue Latency	1	1
Result Latency	6	5

表2 機能ユニットの個数

Model	Cluster	M&A	LSU	Total	
				M&A	LSU
並列性の違いによる評価					
$FS_{1,6,6}$	1	6	6		
$FP_{2,3,3}$	2	3	3		
$FP_{3,2,2}$	3	2	2		
$FP_{6,1,1}$	6	1	1	6	6
$SP_{2,3,3}$	2	3	3		
$SP_{3,2,2}$	3	2	2		
$SP_{6,1,1}$	6	1	1		
FP におけるクラスタ構成の違いによる評価					
$FP_{2,1,1}$	2	1	1	2	2
$FP_{2,2,1}$	2	2	1	4	2
$FP_{2,3,1}$	2	3	1	6	2
$FP_{3,1,1}$	3	1	1	3	3

4.2.3 評価結果および考察

4.2.3.1 並列性の違いによる評価

積和演算器の個数および，ロード・ストアユニットの個数をの個数をそれぞれ6に固定したときの，全ての手法における全ての構成について，性能の違いを評価したのが図5である．ただし手法SPについては，1クラスタあたりの積和演算器の個数が増加しても性能向上が得られなかったことから，$SP_{6,1,1}$ の場合のみグラフに示した．また，グラフは $FS_{1,6,6}$ について正規化しており，$FS_{1,6,6}$ に対する，他の手法の性能向上率を示している．

このグラフが示しているように，ほとんどの積分タイプにおいて $FP_{6,1,1}$ が最も高い性能向上比を示している．また，手法FPについてのみ着目すると，クラスタあたりの機能ユニット数を減らしても，クラスタ数を増やすことが性能向上につながることがわかる．

4.2.3.2 手法FPにおけるクラスタ構成の違いによる評価

手法FPについて $FP_{2,1,1}$ をベースとした場合に，積和演算器を増加させた $FP_{2,2,1}$，$FP_{2,3,1}$ の構成と，クラスタ数を増加させた $FP_{3,1,1}$ をの構成を比較したのが図6である．

このグラフが示しているように，$FP_{3,1,1}$ が高い性能を示している．総積和演算器数では2倍の $FP_{2,3,1}$ と較べてもほとんどの積分タイプで2倍以上の性能を示している．このことから，クラスタあたりの積和演算器の数を増加させるより，クラスタの数を増やすことが性能向上につながると言える．

4.2.3.3 考　　察

これら2つの結果から，手法FPが他の手法と較べ優れており，またFPの構成法としては最小構成（$M\&A = 1, LSU = 1$）のクラスタを多数搭載する構成が最適であると言える．

5. ま と め

以上の検討から，専用プロセッサを初期積分計算エンジンと漸化計算エンジンから成るアーキテクチャとする全体構成を決定した．また，そのうち漸化計算エンジンについては，少数の積和演算器から成るクラスタを多数搭載することで漸化計算部分を高速に処理できることが明らかとなり，今後のプロセッサ・アーキテクチャの仕様を決める上での有用な指針を得た．

今後は面積や消費電力といった制約条件をより詳細に見積もった上で，制約条件の範囲内で最高の性能を得ることができる初期積分計算エンジンの構成，漸化計算エンジンのクラスタ数の決定などを行う．また，そのほかのメモリ構成やホスト間との連携についても詰めていき，2002年度中の論理設計の完了を目標に仕様決定を行っていく．

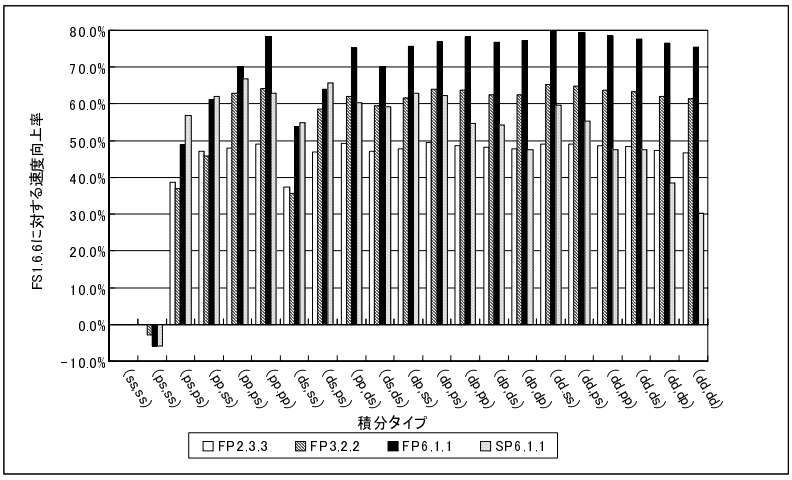

図 5　活用する並列性の違いによる速度向上率比較

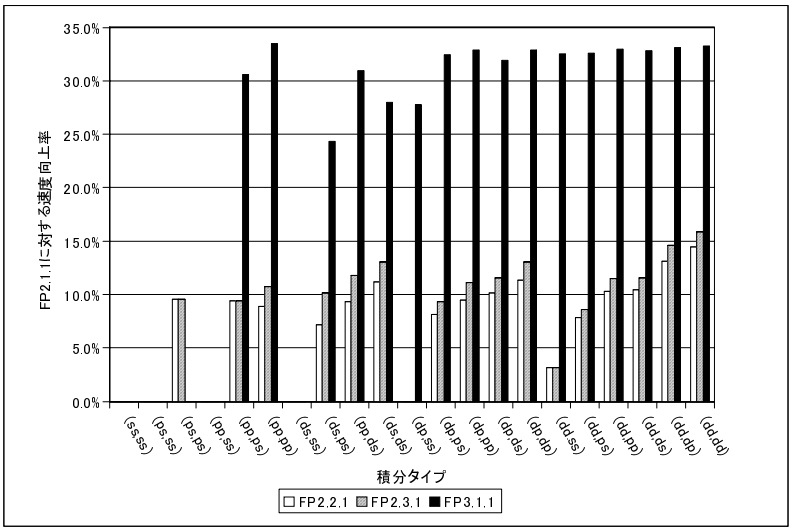

図 6　クラスタ構成の違いによるの速度向上率比較

謝辞　本研究は一部，平成 12 年度科学技術振興調整費総合研究「科学技術計算専用ロジック組込型プラットフォーム・アーキテクチャに関する研究」に依る．

参 考 文 献

1) 高島一, 山田想, 小原繁, 北村一泰, 稲畑深二郎, 宮川宣明, 田辺和俊, 長嶋雲兵, “分散メモリ型並列計算機に適した新しい大規模フォック行列生成アルゴリズム ―積分カットオフとの関連―,” J. Chem. Software, Vol. 6, No. 3, p. 85-104, January 2000.

2) 村上和彰, 稲垣祐一郎, 上原正光, 大谷康昭, 小原繁, 小関史朗, 佐々木徹, 棚橋隆彦, 中馬寛, 塚田捷, 長嶋雲兵, 中野達也, “科学技術計算専用ロジック組込み型プラットフォーム・アーキテクチャの開発 –プロジェクト全体像–,” HPC82-1, 2000 年 8 月.

3) S. Obara and A. Saika, “General recurrence formulas for molecular integrals over Cartesian Gaussian function,” J. Chem. Phys. Vol98 no.3, August 1988.

4) 戸川勝巳, 小原繁, 上原正光, 佐藤比佐夫, 波多江秀典, 中村健太, 村上和彰, “新「小原のアルゴリズム」に基づく二電子積分計算専用 LSI について,” HPC85-5, 2001 年 3 月.

5) 藤永茂, “入門分子軌道法,” 講談社, 1990 年.

6) 村上和彰, “スーパースカラ・プロセッサの性能を最大限に引き出すコンパイラ技術,” 日経エレクトロニクス, No.521:165-185, 1991 年 3 月.