計算機アーキテクチャ 152－23
（２００３．３．１１）

Eric(二電子積分計算専用プロセッサ)LSI の開発

原田宗幸[†1] 中村健太[†1] 桑山庸史[†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[†4] 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専用並列計算機システムに搭載する，二電子積分計算専用プロセッサ，Eric 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，Eric の全体構成および動作について述べる．そして，現段階での Eric の性能評価を行う．

Development of Special Purpose Processor for Molecular Orbital Calculations

Muneyuki HARADA ,[†1] Kenta NAKAMURA ,[†1]
Youji Kuwayama ,[†1] Masamitsu UEHARA ,[†2] Hisao SATO ,[†2]
Shigeru OBARA ,[†3] Hiroaki HONDA ,[†3]
Umpei NAGASHIMA ,[†4] Yuichi INADOMI [†4]
and Kazuaki MURAKAMI[†1]

We are developing Eric which is a special-purpose processor built in special-purpose computer system for *ab initio* Molecular orbital calculations to reduce the calculation time. Using characterization of two-electron integrals in the "Obara method," it is possible to reduce the calculation time. In this paper ,we describe outline and operation of Eric for Molecular orbital Calculations. Also, we evaluate the performance of Eric.

1. はじめに

近年の計算機技術の飛躍的な発達にもかかわらず，解を導くために必要な時間が現実的でないものがある．その 1 つに非経験的分子軌道法による分子軌道計算がある．現在の非経験的分子軌道計算にかかる時間では，非経験的分子軌道法を生命あるいは科学現象の分子論的解明，創薬の分野へ適用することは難しい．

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，EHPC(Embedded High Performance Computing) 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処理する，専用並列計算機システム (EHPC マシン) の開発に取り組んでいる[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，二電子積分計算とフォック行列を生成する部分である．これらの計算にかかる時間が，全体の計算時間の 95%を占める[1)]．したがって，この部分を高速に処理することが，非経験的分子軌道計算の高速化につながる．本稿で紹介する Eric(二電子積分計算専用プロセッサ) は，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．

二電子積分計算のアルゴリズムには，小原のアルゴリズム[3)]を採用する．小原のアルゴリズムを適用することにより，二電子積分計算は漸化計算で表され，複数の積和演算を並列的に計算することで高速化できる．小原のアルゴリズムは，大きく分けて初期積分計算と漸化計算の 2 つから構成される．初期積分計算は，命令レベル並列度が低く，計算に時間のかかる浮動小数点除算，開平逆数演算，指数関数演算をクリティカル

†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
Email: ehpc-lsi@star.fuji-ric.co.jp

パス上に含む．漸化計算は，全ての計算が積和演算で構成され，命令レベル並列度が高い．

そこで，本稿で紹介するEricは，小原のアルゴリズムの特徴に合わせて，初期積分計算を専用に処理する初期積分計算エンジンと，漸化計算を専用に処理する漸化計算エンジンを搭載する．つまり，Ericは，2つの異なるエンジンから構成されるCMP(Chip Multiprocessor) アーキテクチャを採用する．また，漸化計算エンジンは，漸化計算の並列性を十分に利用するため，同じ性質を持つ複数のエンジン (以下，マイクロエンジンと呼ぶ) から構成されるμCMPアーキテクチャを採用する．

本稿では，CMP+μCMPアーキテクチャを採用しているEricの概要および，動作を中心に述べる．以下，2章では，小原のアルゴリズムの特徴について述べる．3章で，Ericアーキテクチャの概要を述べ，4章で，Ericの動作原理について述べる．5章で，Ericの性能評価を行い，6章で本稿をまとめる．

2. 小原のアルゴリズム

二電子積分計算のアルゴリズムとして，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は，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) 型の二電子積分を求めたあと，目的の二電子積分まで漸化計算により求める．したがって，小原のアルゴリズムは図1に示すように，初期積分計算と漸化計算の2つに大きく分けられる．

2.1 初期積分計算

初期積分計算は，図1に示すように，4重のループ構造をとる．このループ1回あたりの演算数は非常に少なく，命令レベル並列度は低い．また，演算数は少ないものの，クリティカル・パス上には，計算に時間がかかる浮動小数点除算，開平逆数演算，指数関数演算，および，誤差関数計算を含む．したがって，初期積分計算を高速に処理するには，浮動小数点除算，開平逆数演算，指数関数演算，および，誤差関数計算を高速に処理する必要がある[4),5)]．

☆ 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]．

```
for (I=0; I<Nshell; I++)
  for (J=0; J<I; J++)
    for (K=0; K<J; K++)
      for (L=0; L<K; L++)
        for (i=0; i< N_i; i++)
          for (j=0; j< N_j; j++)
            for (k=0; k< N_k; k++)
              for (l=0; l< N_l; l++)
                < s_i,s_j,s_k,s_l >の計算
              forend
            forend
          forend
        forend
        < a_I,b_J,c_K,d_L >の計算 (漸化計算)
      forend
    forend
  forend
forend
```

（初期積分計算）

図1 小原のアルゴリズムのプログラム構造

2.2 漸化計算

漸化計算は，図1に示すように，初期積分計算の結果をもとに目的の積分タイプの二電子積分の値を求める．ただし，積分タイプ (ss,ss) に関しては，初期積分計算で積分タイプ (ss,ss) の計算を完了しているので漸化計算は無い．図2に漸化計算の構造を示す．図2からわかるように，漸化計算は積分タイプごとに，51種類の関数を複数回呼び出すことで構成されている．また，関数は，1回から複数回の積和演算のみで構成されており，初期積分計算に含まれる浮動小数点除算などの複雑な演算は漸化計算には含まれない．漸化計算は，関数内の積和演算の並列度が高く，さらに関数間においても並列度が高い．したがって，漸化計算を高速に処理するには，関数内と関数間の並列性を十分に活かす必要がある．

3. Ericアーキテクチャの概要

3.1 設計方針

2章で述べたように，小原のアルゴリズムは特徴の大きく異なる，初期積分計算と漸化計算の大きく2つから構成される．そこで，それぞれの計算の高速化に向けて，

- 初期積分計算は浮動小数点除算，開平逆数演算，指数関数演算，および，誤差関数計算を高速に処理する専用演算器を設けて，初期積分計算を高速化する
- 漸化計算は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，漸化計算を高速化する

という設計方針でEricのアーキテクチャを検討して

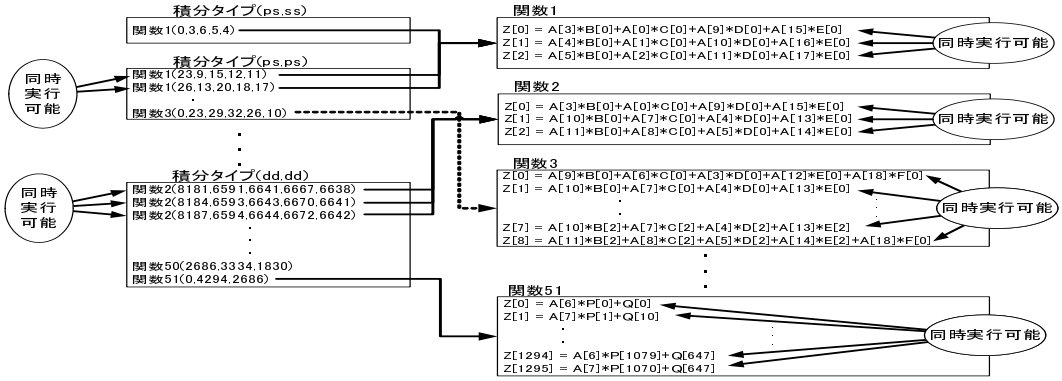

図 2　漸化計算のプログラム構造

きた[4),5)].

しかし，2.1節で述べたように，初期積分計算の命令レベル並列度は低く，2.2節で述べたように，初期積分計算に含まれる浮動小数点除算などの複雑な演算は含まない．そこで，初期積分計算を専用に処理する初期積分計算エンジンと，漸化計算を専用に処理する漸化計算エンジンの 2 つのエンジンによる CMP アーキテクチャで，Eric を構成することを方針に加えた[5)].

さらに，漸化計算エンジンについては，内在する並列性を最大限に活用するための構成を文献 5) で検討した．漸化計算エンジンの構成法として，手法 FS，FP，SP を提案したが，最も良い性能を示した構成は，手法 FP であった．手法 FP とは，関数間の並列性を活かす手法である．概要を以下に示す．

手 法 FP

漸化計算エンジンを複数のマイクロエンジンで構成する．関数呼出しを行うマクロ命令を外部メモリに保持し，関数内部の処理をマイクロ命令化してを内部メモリに準備する．そして，それぞれのマイクロエンジンが並列にマクロ命令を処理する．この処理を漸化計算が終了するまで繰り返す．

これより，漸化計算エンジンを手法 FP，つまり複数のマイクロエンジンによるμCMP アーキテクチャで構成することを方針に加えた．

なお，フォック行列作成は漸化計算が終了したあとに，初期積分計算エンジンで行う．

3.2 全体構成

3.1で示した方針のもとで設計した Eric アーキテクチャのブロック図を図 3に示す．Eric は大きく分けて

- 初期積分計算エンジン (IIC Engine)
- 漸化計算エンジン (RC Engine)
- 内部データメモリ (Internal Data Mem.)
- 対ホストインタフェース (HOST I/F)
- 対外部メモリインタフェース (SDRAM I/F)

から成る．対象とする浮動小数点演算は，全て倍精度の浮動小数点演算であり，単精度はサポートしない．

3.2.1 初期積分計算エンジン (IIC Engine)

小原のアルゴリズムの初期積分計算を処理するエンジンである．また，漸化計算が終了したあと，フォック行列作成も処理する．初期積分計算エンジンは，

- 浮動小数点除算，開平逆数演算専用演算器 (FDIV&1/$\sqrt{}$)
- 指数関数演算, 誤差関数計算専用演算器 (EXP&ERF)
- 浮動小数点積和演算器 (FMUL&ADD)
- ロード・ストアユニット (LSU)
- 整数演算ユニット (IALU)
- IIC Engine 用プログラムメモリ (IIC Program Mem.)
- 誤差関数表 (ERF Table)

などから成る．

3.2.2 漸化計算エンジン (RC Engine)

小原のアルゴリズムの漸化計算を処理する計算エンジンであり，

- μエンジン (μEngine)×4
- μプログラムメモリ (μ Program Mem.)

などから成る．μエンジンの個数は，面積の制約により 4 個とする．

μエンジンは，漸化計算の関数を処理する計算エンジンであり，

- 浮動小数点積和演算器 (FMUL&ADD)
- ロード・ストアユニット (LSU)
- 整数演算ユニット (IALU)

などから成る．

μプログラムメモリは，マイクロ命令を保持するためのメモリである．4 個のμエンジンは，同時に動作

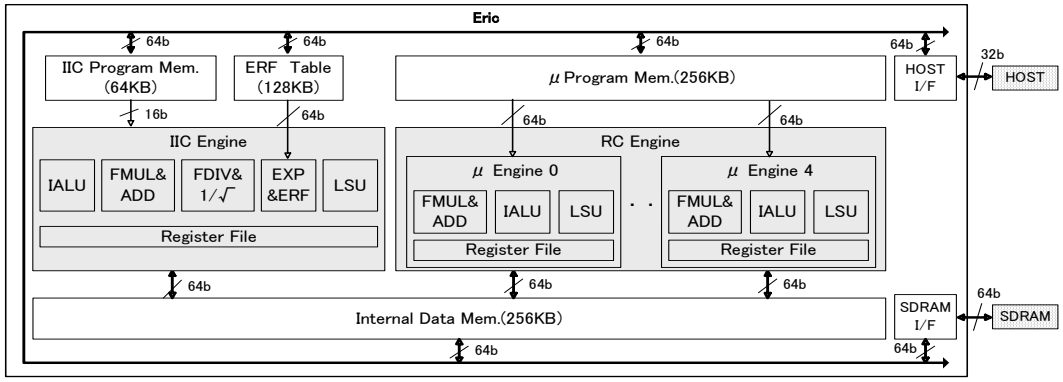

図 3　Eric のブロック図

し，かつμプログラムメモリを共有するため，同時に複数のマイクロエンジンからアクセスされる．したがって，1 ポートでは同時に複数のμエンジンが動作できない．そこで，μプログラムメモリをバンク分けする．バンク分けとは，メモリを複数のバンクに分けて構成するもので，複数のμエンジンは，同時に 1 つのバンクにアクセスすることはできないが，別々のバンクに対してはアクセスすることができる．

3.2.3　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の保持と，ホストとのデータの授受を行う．内部データメモリも，漸化計算エンジンのμプログラムメモリの場合と同様に，初期積分計算エンジンと 4 個のμエンジンが，同時にアクセスする場合がある．そこで，内部データメモリもバンク分けし，複数のエンジンが同じバンクにアクセスする場合を除いて，各エンジンが同時にアクセスするのを可能にする．

4. 処 理 手 順

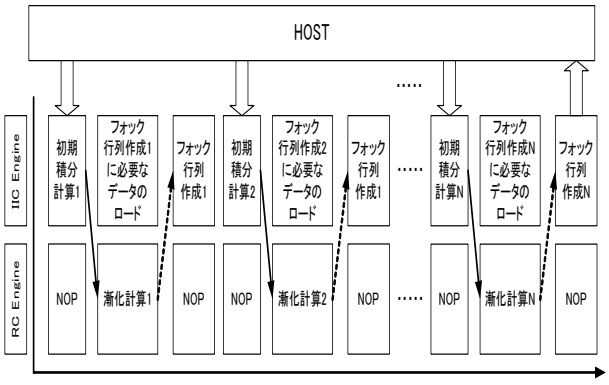

図 4　Eric の動作原理

4.1　Eric の処理手順

Eric の処理の流れは以下のようになる (図 4参照).

1. ホストが初期データ (積分タイプなど) および処理開始信号を Eric に送る．データを受け取るまで Eric は待つ．
2. Eric がホストから開始信号を受け取ったら，IIC Engine は初期積分計算を行う．一方，RC Engine は計算終了まで待つ．
3. IIC Engine が初期積分計算を終了したら，RC Engine が漸化計算を行う．一方，IIC Engine は，フォック行列に必要なデータを外部メモリから内部データメモリにコピーする．
4. RC Engine が漸化計算を終了し，IIC Engine がフォック行列に必要なデータのコピーを終了すると，IIC Engine は，フォック行列作成の処理を開始する．RC Engine は，次の二電子積分計算の初期積分計算が終了するまで待つ．
5. IIC Engine がフォック行列作成の処理を終了したら，Eric はホストに終了を知らせる．
6. ホストは，次の二電子積分計算が存在する場合は 1 に戻る．無い場合は，計算結果を Eric より受け取り，全体の処理を終了する．

4.2　IIC Engine の処理手順

IIC Engine の処理の流れは以下のようになる．

1. Eric がホストから開始信号を受け取ると，初期積分計算を行う．
2. 初期積分計算を終了したら，フォック行列に必要なデータを外部メモリより内部データメモリにコピーする．
3. フォック行列に必要なデータのコピーが終了したら，漸化計算が終了するのを待って，フォック行列作成の処理を開始する．
4. フォック行列作成の処理が終了したら，ホストに

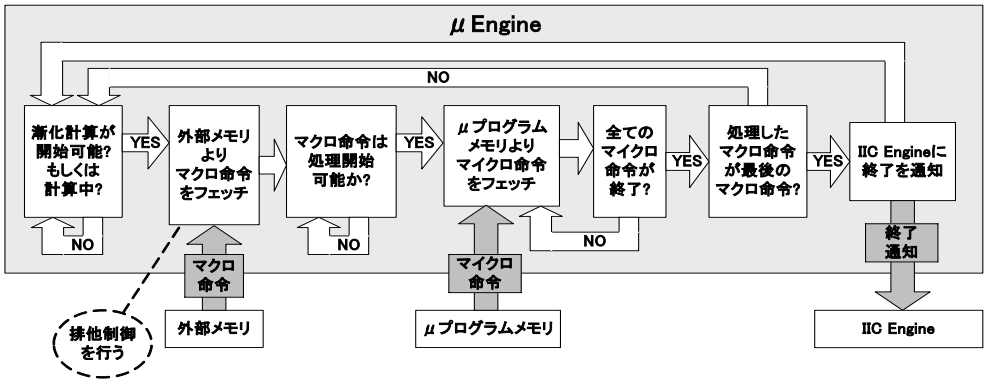

図 5　RC Engine の動作原理

終了を知らせ，次の二電子積分計算の初期積分計算開始まで待つ．

4.3　RC Engine の処理手順

RC Engine は，4 個のμEngine それぞれが，以下の処理手順で動作して漸化計算を処理する (図 5参照)．なお，手順 2 は 1 つのμEngine だけが動作できるような排他制御を行う．

1. 漸化計算を計算可能な状態であれば，2 に進む．それ以外の場合は，計算可能になるまで待つ．
2. マクロ命令を外部メモリよりフェッチする．
3. フェッチしてきたマクロ命令の処理を開始可能か判定する．開始可能なら 4 に進むが，不可能なら可能になるまで待つ．
4. マクロ命令に対応するマイクロ命令を，μプログラムメモリより毎サイクルフェッチしてきて処理を行う．
5. 全てのマイクロ命令が終了したあと，処理したマクロ命令が最終のマクロ命令であった場合は，IIC Engine に知らせて，1 に戻る．それ以外の場合は，何もせずに 1 に戻る．

5. 性 能 評 価

5.1　評 価 方 法

評価は，漸化計算にかかる実行時間で行った．しかし，Eric については，まだ実際の LSI はできていないため，実行時間を測定できない．そこで，シミュレーションで実行サイクル数を求めて，Eric のクロックサイクル時間と実行サイクル数の積をとって実行時間を求めた．なお，クロックサイクル時間は，動作周波数を 200，250，300MHz として求め，実行サイクル数は，メモリへのアクセスは，毎サイクル同時アクセス可能，つまり同じバンクには複数のμEngine がアクセスしない，と仮定して求めた．また，ベースモデルには，800MHz の Pentium Ⅲ と 512MB のメモリを搭載した PC を用いた．

5.2　評価結果および考察

評価結果を図 6に示す．なお，性能向上率は以下の式より求めた．

$$\text{性能向上率} = \frac{\text{Pentium の実行時間} - \text{Eric の実行時間}}{\text{Pentium の実行時間}} \times 100(\%)$$

図 6に示すように，動作周波数が 200MHz の Eric は，(dd,dd) などの大きい積分タイプにおいては，約 280%の性能向上が見られる．しかし，ほとんどの積分タイプにおいて Pentium に劣っており，最大で約 75%の性能低下が見られる．一方，動作周波数が 250MHz や 300MHz になると，(ps,ss) などの小さい積分タイプ以外を除けば，Eric の方が性能が良くなる．(dd,dd) においては，250MHz で約 370%，300MHz では，約 470%もの性能向上が見られる．

図 7にペプチド分子 GAQMY における，漸化計算の総実行時間を示す．なお，総実行時間は，各積分タイプの漸化計算にかかる実行時間と積分タイプの実行回数の積より求めた．

図 7より分かるように，200MHz の動作周波数における Eric は，総実行時間で比較した場合でも 36 秒ほど Pentium に劣っている．しかし，250MHz 以上になると，逆に圧倒的に Eric の方が性能が良くなる．250MHz では実行時間の差は，640 秒ほどとなり，300MHz では約 1100 秒にも及ぶ．

これらの結果より，現在の実行サイクル数のもとでは，Eric は最低でも，250MHz 以上の動作周波数で動作する必要があると言える．ところが，今回の評価は，メモリアクセスに関しては，同じバンクに対する

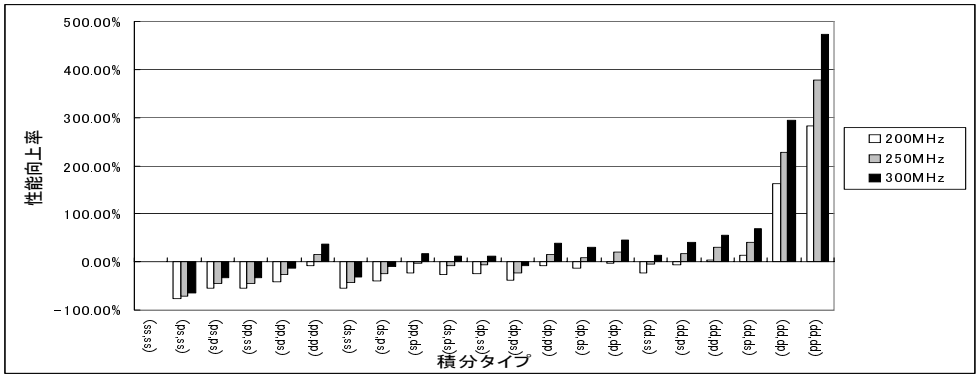

図 6　Pentium の実行時間に対する Eric の実行時間

アクセスは無いと仮定したものであるため，実際の実行時間は，さらに増える可能性が高い．したがって，Eric は 300MHz の動作周波数を目指す必要がある．

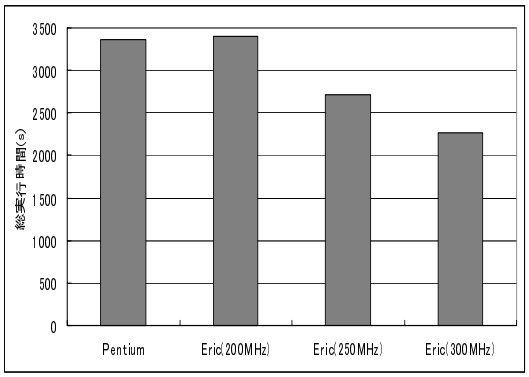

図 7　ペプチド分子 GAQMY における総実行時間

6. おわりに

本稿では，Eric が対象とする二電子積分計算のアルゴリズムとして採用している小原のアルゴリズムの概要について述べた．そして，小原のアルゴリズムの特徴に合わせた Eric アーキテクチャの概要と，動作について述べた．また，漸化計算にかかる実行時間より評価を行い，Eric の目標動作周波数を示すことができた．だが，実行時間削減のためには，動作周波数を向上させる以外に，実行サイクル数を削減することも考えられる．そこで，今後の課題としては，目標動作周波数で設計することに加えて，命令スケジューリングなどの最適化手法の改良などを行い，実行サイクル数を削減することも挙げられる．

なお，本稿で述べたアーキテクチャに基づいて，Eric の設計を行い，平成 15 年中には，テープアウトする予定である．

謝辞　本研究は一部，平成 14 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) 中村健太, 波多江秀典, 原田宗幸, 上原正光, 佐藤比佐夫, 小原繁, 本田宏明, 長嶋雲兵, 稲富雄一, 村上和彰, “二電子積分計算専用プロセッサ・アーキテクチャ,” HPC89-13, 2002 年 3 月.
6) 木原寛, 内田希, 生田茂, “分子軌道法”, 講談社, 1994 年.