A High Performance, Low Power Chip Multiprocessor for Large Scale Molecular Orbital Calculation

Kenta NAKAMURA⁎, Hiroaki HONDA⁎‡, Koji INOUE⁎, Hisao SATO†, Masamitsu UEHARA†, Harunobu KOMATSU†, Hiroaki UMEDA§, Yuichi INADOMI§, Kengo ARAKI§, Toru SASAKI§, Shigeru OBARA∥, Umpei NAGASHIMA∥ and Kazuaki MURAKAMI⁎

⁎Department of Informatics, Kyushu University
†Seiko Epson Corp.
‡Mizuho Information & Research Institute, Inc.
§National Institute of Advanced Industrial Science and Technology
∥A Priori Microsystems Inc.

10mm × 5mm; the operating frequency is 200MHz; the estimated power consumption is 2.1W.

Abstract—Ab initio molecular orbital (MO) calculation is very important for solving many challenging problems concerning the development of new drugs, chemicals, polymers, materials, and so on. However, large-scale MO calculations have at least two difficulties. The first is a considerable number of Electron Repulsion Integral (ERI) computations. The second is the computational complexity of single ERI. We are constructing special purpose processors, and a high performance and compact parallel computing system embedding them for a large-scale MO calculation, named EHPC (Embedded High Performance Computing) system. The numerous ERI calculations can be performed efficiently by the hierarchical parallel computing system architecture, and the special processors accelerate the computation of single ERI. For the parallel computing system, high-performance, low power consumption, and high densely integratable special purpose processor is strongly required. So we developed the ERI calculation specific chip-multiprocessor (CMP) architecture, named ERIC. This is the first implementation of an application specific processor architecture for MO calculation.

The ERIC processor architecture has following unique features: heterogeneous CMP architecture optimized with ERI calculation algorithm, microengine-based chip-multiprocessor (μCMP) architecture, and special floating-point operating units, especially inverse-square-root unit, exponential function unit and error function unit for calculating Taylor expansion. We have developed the actual ERIC processor chip with TSMC 0.13μm CMOS technology. The die size is 10mm × 5mm; the operating frequency is 200MHz; the estimated power consumption is 2.1W.

I. INTRODUCTION

Ab initio molecular orbital (MO) method is very important for calculating electronic properties and structures of molecules. Hence, it is indispensable today to interpret experimentally observed phenomena and to predict various physical and chemical properties of known or unknown molecules to design functional materials by the MO method. It can also present information on inter-atomic interaction in molecular assemblies, and is a basis of all molecular simulation approaches.

However, large-scale MO calculation has at least two difficulties: numerous numbers of electron repulsion integral (ERI) calculations and computational complexity of single ERI. As mentioned in next section, \(O(N^4)\) numbers of electron repulsion integral (ERI) calculations are required, where \(N\) corresponds to the computational size of the target molecular system. Moreover, the computation of single ERI is laborious task, almost all arithmetics are consist of double precision floating-point operations, it needs to compute division, inverse square root, exponential function, error function, and a large number of double precision floating-point multiply-and-add operations.

In recent years, many high performance computer systems have been developed as high speed vector computer systems and huge (PC) cluster computer systems, and large scale MO calculations have been actually possible. However, it’s difficult for almost all chemists to use such large computer systems. They are expensive, hard to maintain, require high power consumption, and inappropriate for trial calculations of some ideas. Furthermore, it is difficult to keep the target chemical compounds secret from the other researchers. Therefore, development of the personal computer system, which is high performance, low cost, and easy to use is strongly desired.

We have been therefore developing EHPC (Embedded High Performance Computing) system with following special features in answer to the difficulties and the requirement:

- The total system with hierarchical parallel computing architecture which enables numerous ERI computations efficiently,
- ERI specific LSI which enables fast computation of the single integral,
- Low power consumption and easy to use as the personal computer system.

For realization of those features, high-performance, low power consumption, and high densely integratable special purpose processor is indispensable. So we investigate the ERI computation algorithm, and develop the ERI calculation specific processor architecture, named ERIC.

The rest of the paper is organized as follows: section II
describes the outline of MO calculation. Section III shows the overview of the EHPC system and the ERIC processor architecture, and section IV describes the ERIC processor architecture in detail. Then we show the specifications of the ERIC processor chip, and evaluate the attainable performance of the ERIC processor in section V. Section VI concludes the paper.

II. CHARACTERISTICS OF THE MO CALCULATION

A. General MO Calculation Scheme and Most Time Consuming Step

Ab initio MO calculation is performed by solving following Hartree-Fock equation [1] (Eq. 1):

\[ \sum_{i=1}^{N} \{ F^{(1)}_{ij} + F^{(2)}_{ij} \} C_{ia} = \sum_{i=1}^{N} S_{ij} C_{ia} \epsilon_{a} \]  

(1)

\[ F^{(2)}_{ij} = \sum_{k=1}^{N} \sum_{l=1}^{N} P_{kl} \{(ij, kl) - \frac{1}{2} (ik, jl)\} \]  

(2)

\[ P_{ij} = 2 \sum_{\alpha \in occ} C_{i\alpha} C_{j\alpha} \]  

(3)

\[(ij, kl) = \int \chi_{i}(r_{1}) \chi_{j}(r_{2}) \frac{1}{|r_{1} - r_{2}|} \chi_{k}(r_{2}) \chi_{l}(r_{2}) dr_{1} dr_{2} \]  

here, \( F^{(1)}_{ij}, (ij, kl), F^{(2)}_{ij}, \) and \( P_{ij} \) denote a one-electron Fock matrix, a electron repulsion integral (ERI), a two-electron Fock matrix element, and a density matrix element, respectively. A set \( \{ \chi \} \) is called basis function set. \( N \) is the number of basis functions and is regarded as the computational size of the target molecular system. The total electronic wave function and energy value, and various electronic properties of target molecule are obtained from the resultant set \( \{ C \} \), which is regarded as the coefficients of MO \( (\phi_{a} = \sum_{i=1}^{N} C_{ia} \chi_{i}) \). Since \( F^{(2)} \) matrix depends on the answer \( C \) through \( P \) matrix, above Hartree-Fock equation (Eq. 1) is solved by the iterative procedure called as the SCF (Self-Consistent Field) method.

Whole SCF computational procedure can be outlined as follows: 1. General Setup. 2. Input initial parameters, and generate initial data such as coordinates, basis function set, and so on. 3. Calculate one electron integrals (ONE-EIs). 4. Generate \( F^{(2)} \) matrix with ERI computations. (3 ~ 4 steps are iterative.) 5. Diagonalize the Fock matrix, Judge the convergence, and Update the density matrix. 7. Calculate various properties (when density matrix is converged).

Table I shows the ab initio MO computation time for five sample molecules, and summarizes the time distribution for each step in the SCF procedure. In the SCF procedure mentioned above, the step of \( F^{(2)} \) matrix generation consumes more than 98\% of the total computation time. Furthermore, this percentage increases over 99\% for larger scale calculation. If it is possible to shorten this computation time to a great extent, total computation time would be decreased drastically.

B. Fock Matrix Generation Algorithm

The integral driven calculation algorithm [2], shown in Figure 1, is widely used for the step of \( F^{(2)} \) matrix generation in almost all SCF programs currently available. As the algorithm indicates, the number of ERI calculations is proportional to \( N^{4} \) and increases rapidly as \( N \) becomes larger.

As shown from this algorithm, a calculation of a set \( i, j, k, \) and \( l \) doesn’t depend on the other set. Therefore, there are a lot of parallelism between each ERI calculation.

C. Algorithm for the ERI Calculation by the Obara method

The computation of single ERI is a laborious task. There are several algorithms for the ERI calculation. Among such algorithms, we selected the Obara method which is based on the general recurrence formula for contracted Gaussian function. This method have the good features of fast calculation, numerically high accuracy, and can be applied to the various integral calculations other than the ERI. Figure 2 shows this Obara algorithm for the ERI Calculation[3], [4].

The most characteristic point of the Obara algorithm is its process. An initial integrals are calculated with input data. Then this result is used to perform the recursive calculation for a certain ERI. Therefore, the Obara algorithm can be roughly divided into two segments: initial integral calculation and recursive calculation as shown in Figure 2.

1) Initial Integral Calculation: The initial integral calculation has a four-fold loop structure. In the initial integral calculation, there is a small number of operations per iteration, each with a small amount of instruction-level parallelism (ILP). To make matters worse, the initial integral calculation contains complex double precision floating-point operations such as division, inverse square root, exponential function for

![Fig. 1. Integral Driven Algorithm for the Fock Matrix Generation](image-url)
(Generation of Initial Integrals)
for $a = 1, M_i$
for $b = 1, M_j$
for $c = 1, M_k$
for $d = 1, M_t$
end loop

error function:
inverse-square root, division,
exponential function,
and Taylor expansion calculations
end loop
end loop
end loop

(Recursive Calculation from Initial Integrals)
recursive calculation to the target $(ij, kl)$:
many multiply-and-add calculations

Next, $F_m(T) \sim F_{m-1}(T)$ are obtained by the following relation:

$$F_m(T) = \frac{1}{(2m + 1)} \{ \exp(-T) + F_{m+1}(T) \}$$  \hspace{1cm} \text{(8)}$$

2) Recursive Calculation: Recursive calculation computes a objective ERI after the processing of initial calculation is completed. The recursive calculation part consists of many hierarchical structured sub-functions, which consist of a large number of double precision floating-point multiply-and-add operations and can be processed in parallel.

III. EHPC PLATFORM AND ERIC PROCESSOR ARCHITECTURE

The goal of our project is to build a high-performance parallel computing system which implements many embedded ERI processors, named Embedded High Performance Computing (EHPC) system. We introduce the overview of the EHPC platform and the ERIC processor architecture in this section.

A. EHPC Platform Architecture

1) EHPC Platform Overview: As shown in Figure 4, the total EHPC platform architecture has three-level hierarchical
network structure: conventional PC, housekeeping SH-4 processor and worker ERIC processor in order of descending levels, optimized with the Fock matrix generation algorithm by the integral driven method (Figure 1). One chassis system consists of a general purpose CompactPCI CPU board as a top layer conventional PC board and up to seven custom computation processor boards based on CompactPCI standard, and those boards are connected by CompactPCI bus each other. The ERIC processors are mounted on the board with a Hitachi SH-4 processor. The SH-4 processor on each board has a role in housekeeping and distributes segmentalized Fock matrix computation to the worker ERIC processors. The EHPC system can be used as PC cluster system by connecting chassis systems through 100Mbps Ethernet.

Since low power consumption is one of the most important requirements to the ERIC processor design for implementing the EHPC system, we decide to adopt S1C33 family which is an embedded use RISC-type CPU as a base processor. With this limitation, we investigate the ERI calculation algorithm shown in section II. The Obara algorithm consists of two segments whose characteristics are quite different from each other: an initial integral calculation (IIC) and a recursive calculation (RC), as shown in Figure 2. So we decided the design philosophy for coping with the two calculation parts, the IIC part and the RC part as follows:

- The IIC part including some complex floating-point operations accelerates processing by installing special functional units to a processor.
- The RC part having rich ILP accelerates by installing many functional units to a processor and by parallel processing.

However, implementing the both functions with one processor is not efficient for two reasons: the IIC part does not need many functional units for the sake of its low parallelism and the RC part does not need special functional units for lack of complex floating-point operation. Therefore we would not optimize the ERIC processor with the whole part of the Obara algorithm, and decided to install a co-processor which accelerates a series of floating-point multiply-and-add operations in the ERIC processor. Namely, we adopted a heterogeneous chip-multiprocessor (CMP) architecture as the ERIC processor design, and divided the ERIC processor into two engines which work concurrently: an IIC engine as a main engine and an RC engine as the co-processor for calculating floating-point multiply-and-add operations.

Figure 5 shows the overview of ERIC processor. ERIC processor is mainly composed of the following five modules:

- Host Interface for communication with Host SH4 processor
- SDRAM Interface for accessing External Memory
- Internal Data/Program Memory for storing the calculation results and programs
- IIC Engine for the initial integral calculation
- RC Engine for the recursive calculation

This processor only supports IEEE 754 double-precision floating-point numbers. Therefore, all the functional unit, memory and interfaces are 64-bit width, and single precision floating-point numbers are not supported.

IV. ERIC: ERI-CALCULATION SPECIFIC PROCESSOR ARCHITECTURE

A. IIC Engine

The initial integral calculation (IIC) consists of various operations such as arithmetic, floating-point addition, multiplication, division, inverse square root, exponential function, function call and conditional branch, and lacks in instruction-level parallelism (ILP). Therefore, the design like a general purpose RISC processor is suitable for the architecture of the initial integral calculation engine. In the ERIC processor, the
IIC engine is a customized processor of S1C33 family (Seiko Epson’s original 32-bit RISC-type CPU). For the IIC part of the Obara algorithm, we added five special floating-point operating units, multiply-and-add, division, inverse square root, exponential function and error function to the base processor.

1) S1C33: 32-bit RISC Microprocessor Family: S1C33 family is Seiko Epson’s original 32-bit RISC-type CPU. This CPU was developed for high-performance embedded applications such as peripheral equipment for personal computers, portable equipment and other products which need high-speed data processing with low power consumption. The S1C33 instruction set contains 61 basic instructions (105 instructions in all), but no floating-point instruction. Since the ERIC processor must calculate double-precision floating-point numbers for processing MO computation, we added some floating-point operating units written in the following section to the IIC engine as additional components.

2) Floating-Point Operating Units: As mentioned above, S1C33 family does not have a floating-point operating unit, so we decided to add five double-precision floating-point operating units as follows:

- Floating-point multiply-and-add unit
- Floating-point division and inverse square-root unit
- Floating-point exponential function unit
- Floating-point error function unit which calculate Taylor expansion (Eq. 7)
- 64-bit width load-store unit

Execution clock cycles of the double precision floating-point arithmetic operation are listed on the Table II. These various floating-point arithmetic operations are important for the ERI calculation. For example, the execution clock cycles of the error function unit is 16, while it takes 32 instructions and 146 clock cycles by using the Taylor expansion method measured on the typical MIPS processor. The error function is performed many times at the deepest level of total Fock matrix generation with ERI calculation loop structure. Similarly, execution clock cycles by using the exponential function, the square root, and inverse-square root unit can also shorten the execution clock cycles compared with the software evaluation. Therefore it is expected that the development of ERIC processor will be very effective for ERI calculations.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Cycles</th>
<th>Parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>6</td>
<td>pipelined</td>
</tr>
<tr>
<td>Multiplication</td>
<td>6</td>
<td>pipelined</td>
</tr>
<tr>
<td>Division</td>
<td>23</td>
<td>unpipelined</td>
</tr>
<tr>
<td>Square root</td>
<td>31</td>
<td>unpipelined</td>
</tr>
<tr>
<td>Exponential</td>
<td>20</td>
<td>unpipelined</td>
</tr>
<tr>
<td>Error function</td>
<td>16</td>
<td>unpipelined</td>
</tr>
</tbody>
</table>

B. RC Engine

Recurrence calculation (RC) engine is a co-processor which is specialized to the RC part of the Obara algorithm. As described in the section II, the RC part consists of a large number of function calls, and each sub-function contains only a series of substantial floating-point multiply-and-add operations without branches. Since there is a lot of parallelism between those multiply-and-add operations, we can execute many operations simultaneously. So we decide to install plural multiply-and-add operating units to the RC engine architecture, and investigate the optimal organization for efficient parallel execution of the numerous multiply-and-add operations.

1) µCMP Architecture: For effective utilization of the multiply-and-add operating units, we analyze the characteristics of the RC part and focus attention on an inter-subfunction parallelism. As described in section II-C.2, since there is parallelism between subfunctions and each subfunction contains only floating-point multiply-and-add operation, load / store operation and integer addition / subtraction for address calculations, we proposed a microengine-based chip-multiprocessor (µCMP) architecture [7], [8]. The µCMP architecture has plural quite small specialized processors only for computing floating-point multiply-and-add operation, named Micro-Engines, and each subfunction is computed in parallel by allotted to each Micro-Engines.

2) Design Considerations: We kicked the organization of the Micro-Engines around and experimented with the several types of the RC engine design to find out the optimal architecture. In this experimentation, we assumed that the maximum number of multiply-and-add units and load-store units we can implement is four, and compared the execution cycles of the RC part on each RC engine design changing the number of the Micro-Engines and functional units in each Micro-Engine.

Figure 6 shows the result of the experimentation. It shows the speed-up ratio of the alternative RC engine designs over the base model RC1, 4, 4. Each model notation of the graph means the number of Micro-Engines and functional units as shown in Table III. From the result we can draw the conclusion

---

1Epson Electronic Devices: http://www.epsondevice.com/
that we can get better performance when increasing the number of Micro-Engines than when increasing the functional units in each Micro-Engine.

**TABLE III**

<table>
<thead>
<tr>
<th>Model</th>
<th># of ME</th>
<th># of Units</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC1, 4, 4</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>RC2, 1, 1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>RC2, 2, 1</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>RC2, 2, 2</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>RC3, 1, 1</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>RC4, 1, 1</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

ME: Micro-Engines M&A: Multiply-and-add units LSU: Load-store units

From the result of the experiment, we finally decided to put four Micro-Engines on the RC engine and the components of each Micro-Engine as follows: a register file, a load-store unit, an arithmetic unit and a floating-point multiply-and-add unit. And each Micro-Engine has a basic instruction-set containing only 17 types of instruction such as integer addition/subtraction, load/store, floating-point multiply-and-add, some branch and system control operations.

V. CHIP IMPLEMENTATION

A. Major Specifications

We designed the ERIC processor and fabricated its actual chip with 0.13-micron processes CMOS technology by TSMC. Figure 7 shows the photograph of the ERIC processor chip, and its major specifications are listed on the Table IV with one exception that only power consumption is estimated value because we have not measured it yet.

The estimated power consumption of the ERIC processor is 2.1W, and maximum power consumption is 4.2W. The power consumption is low enough to implement a large number of the processors in the EHPC (Embedded High Performance Computing) platform as we expect.

**TABLE IV**

<table>
<thead>
<tr>
<th>Major Specifications of the ERIC Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die Size</td>
</tr>
<tr>
<td>Logic</td>
</tr>
<tr>
<td>Memory</td>
</tr>
<tr>
<td>Frequency</td>
</tr>
<tr>
<td>Power† (Max.)</td>
</tr>
</tbody>
</table>

†Estimated power consumption

B. Performance Evaluation

We can not show the exact performance evaluation of the entire EHPC system at the moment, because the EHPC system has not been completed and we are currently working on setting up the system which still has a lot of problems to be solved. However, we have the actual ERIC processor chips and experimental boards for testing them, and can evaluate the performance of the single ERIC processor chip. In this section we show side-by-side performance comparison between the ERIC processor and a single PC. The evaluation condition for each is as follows:

ERIC: • Actual execution time of the single ERIC processor with 704Kbytes on-chip SRAM and 256Mbytes off-chip SDRAM (The operating frequency is 200MHz.)
• Programs written in assembly and C language, compiled by S1C33 compiler which does not support floating point optimization

PC: • Actual execution time of a 3.2GHz Pentium 4 processor with 1Mbytes L2 cache and 2Gbytes main memory
• All programs written in C language and compiled by gcc -O2 option
1) IIC Engine: Table V shows the performance evaluation of the IIC engine compared with the Pentium 4 processor for calculations of a \((pp, pp)\) type initial integral and the error function. This error function calculations are the most time consuming step in innermost iteration of the IIC computation 4-fold loop structure as shown in Figure 2. For the \((pp, pp)\) calculation, the IIC engine consumes 41 times execution time compared to the Pentium 4 processor. Now, we are developing the ERIC specific compiler, so this \((pp, pp)\) code optimizations are far from complete at the present stage. For the error function part, the assembly codes have already been fully optimized for ERIC processor. This part of calculation shows calculations of the Pentium 4 processor is 16 times faster than the IIC engine. By the code optimization and utilizing advantages of double precision floating point calculations and special arithmetic units, it is expected that \((pp, pp)\) execution times decrease drastically as this error function level.

Datasheet shows the power consumption of Pentium 4 is 103.0W (TD: Thermal Design Power) [9], while for the ERIC it is 4.2W (Max.). From the performance-per-power \(\left(\frac{\text{Exec. Time}}{\text{Power}}\right)\) point of view, the result shows that the IIC engine has almost 1.5 times higher efficiency than the Pentium 4. However, it is not so efficient as we expected.

2) RC Engine: Table VI shows the execution time (sec.) of the RC part for five peptide molecules, G, GA, GAQ, GAQM and GAQMY. It indicates that the RC engine requires nearly 36 times more execution time than Pentium 4 on average, regardless of the inputs.

C. Improving the Performance of RC Engine

These evaluations in the previous section show that the ERIC processor has no advantage in both computation performance and power efficiency over Pentium 4. It is a disappointing result. So we are now examining why our implementation does not work well, for example, we are examining on memory architecture, functional units design, utilization rate for each functional unit. In this section we show several experimental result.

As a cause of the low performance of the RC engine, we focused attention on the low IPC (Instructions Per Clock cycle), investigated on the memory architecture of the ERIC processor using a cycle-accurate simulation model based on the Verilog-HDL description of the RC engine. Although we adopt a simple request/acknowledge protocol as the on-chip bus protocol of the ERIC processor for both the program memory and the data memory, this bus bandwidth becomes extremely low because of shortening design periods. Specifically, the minimum memory access cycle to on-chip SRAM in all data types (char, short, int and double) is 6. Since these buses are not pipelined, its memory bandwidth is 1 data / 6 clock cycles (30ns) for each engine. In other words, each RC engine can fetch only one instruction for every 6 cycles. SDRAM controller connected with the main memory also has the bandwidth problem. The Figure 8 shows the SDRAM access cycles of each RC engine. The graph shows that some memory accesses, 19.65% on RC[2] and 14.18% on RC[3], take more than 100 cycles, and the worst case takes 32,218 cycles to get a 64bits-width data.

As the result, the total IPC of four RC engines is approximately 0.28, while the average IPC of single RC engine is 0.07. This indicates that the each RC engine can issue only one instruction for every 14 clock cycles.

The Figure 9 compares execution time and power efficiency of Pentium 4 and the ERIC processor chip with those of ERIC simulation models. Here, ERIC-Sim is the simulation model of the ERIC processor with the ideal memory, and each memory (including SDRAM) access can be completed in 6 cycles, and its bus protocol is the same as that of the ERIC processor chip. ERIC-Sim(p) adopts pipelining technique as the bus protocol on the ERIC-Sim for improving the memory bandwidth, and its memory access cycle are also 6.

As shown in Figure 9, we can recognize that the execution time can be shortened to two fifths of the current one if the SDRAM access can be reduced from the ERIC chip. Furthermore, improvement of the bus protocol can make the execution time shortened to one third of the above, which shows ERIC processor possibly has the competitive advantage over Pentium 4 in power efficiency. These results show the
**VI. CONCLUSION**

*Ab initio* molecular orbital (MO) method is very important for calculating electronic structures and properties of molecules, and the personal type of high performance computer is still strongly desired for MO calculation. In the MO computation, electron repulsion integrals (ERIs) consume the most and tremendous calculation time.

In this work we can satisfy the power requirement to assemble many ERIC processors into the compact EHPC system. Unfortunately, we have not achieved enough performance in our research phase yet, however, by improving the memory architecture, the ERIC processor has a potential to obtain 4.58 times higher efficiency over the Pentium Processor.

**ACKNOWLEDGMENT**

The research is granted by Japanese Ministry of Education, Culture, Sports, Science and Technology.

**REFERENCES**


