# FPGAによる天体物理学計算の高速化

中里直人[†] 濱田剛[†]

天体物理学では重力多体問題専用計算機 GRAPE が非常に大きな成果をあげてきた．本論文では，演算機能が固定されているという GRAPE 計算機の欠点を解消すべく，Field Programmable Gate Array（FPGA）を利用した計算機上で，浮動小数点演算を実行し天体物理学計算の高速化を行った．我々が開発した FPGA に浮動小数点演算による演算回路を実装するためのソフトウエア PGR を使用し，重力多体問題を大幅に高速化できることだけでなく，世界で初めて SPH 法の専用計算機による高速化に成功した．本論文の結果は，FPGA による浮動小数点演算の実用性を実証している．

# Acceleration of Astrophysical Simulations Using a FPGA Board

Naohito Nakasato[†] and Tsuyoshi Hamada[†]

In astrophysical simulations, GRAPE system has been widely used by many researchers. However, in the GRAPE systems, its function is completely fixed because a specially developed LSI is used as a computing engine. Instead of using such LSI, we are developing a special purpose computing system using Field Programmable Gate Array (FPGA) chips as a computing engine. Using our developed programming system PGR, we have implemented computing pipelines for gravity and the Smoothed Particle Hydrodynamics (SPH) method on our PROGRAPE-3 system. For the first time, we have shown that SPH simulations can be accelerated using our system.

## 1. はじめに

天体物理学では，GRAPE 計算機[1),2)] とよばれる重力多体シミュレーションに特化して高速に計算を行う専用計算機が大きな成果をあげている．重力多体シミュレーションでは，銀河や球状星団などの天体を質点の集団として扱い，個々の質点が互いに重力を及ぼしあうとして，それらの軌道を数値積分することで系の進化を調べる．このとき，質点間の重力相互作用計算は質点の数 $N$ の二乗の計算量となり，この計算がシミュレーションのボトルネックとなる．GRAPE 計算機では，重力相互作用計算の部分に専用に設計された集積回路（LSI）を使うことで，重力多体シミュレーションに限ると大型並列計算機をはるかにしのぐ計算速度と価格対性能比を実現している．これは，一般の計算機では様々な集積回路資源が汎用性やキャッシュメモリのために費やされ，数値演算に使用される資源は回路の 1 割にも満たないのに対し，GRAPE 計算機では計算可能な系を重力相互作用する質点系と限定したうえで，集積回路資源のほぼすべてを演算のために費やすことで実現されている．またそのアーキテクチャ上の特徴は，並列な演算回路にデータをブロードキャストするという構成により，メモリ帯域のボトルネックを軽減しているということにある（以下，このような計算機の構成を GRAPE 型計算機と呼ぶ）．非常に成功してきた GRAPE 計算機には，以下のような改良すべき点がある．第 1 に，重力相互作用のみに特化しているので，他の相互作用（たとえば分子間力や核力など）には直接的には適用できないことである．他の相互作用に対応するには LSI の再設計が必要になる（たとえば分子間力相互作用専用計算機の MD-GRAPE[3)]）．第 2 に，仮に研究者が重力以外の相互作用の関わるシミュレーションを高速に実行するために，GRAPE 型計算機と似たアプローチをとろうとしても，GRAPE 計算機に使われるような LSI の設計には数値シミュレーション手法とはまったく違う分野の専門知識（電子工学，回路設計）が膨大に必要であり，容易には実行できないことである．そのため，天体物理学における専用計算機の使用は重力相互作用する系にほぼ限られてきた．

天体物理学でよく使用されている流体シミュレーション手法に，Smoothed Particle Hydrodynamics（SPH）法[4),5)] がある．SPH 法では，流体を粒子に

† 独立行政法人理化学研究所戎崎計算宇宙物理研究室
The Institute of Physical and Chemical Research (RIKEN)

よって表現し，粒子間には圧力勾配による相互作用が働くとして，それらの軌道を数値積分することで系の進化を調べる．SPH 法では，粒子間の圧力勾配による力は，重力多体シミュレーションと同じように，粒子間相互作用の累算で表されるため，Yokono ら[6] は SPH 法に特化した専用 LSI の開発を試みたが，SPH シミュレーションの高速化にはいたらなかった．我々は，SPH 法に特化した専用 LSI を開発するのではなく，ある程度汎用に使用できる Field Programmable Gate Array（FPGA）を使った GRAPE 型計算機[7] の開発およびその上での天体物理学計算の高速化を行っている．FPGA とは，任意の論理演算が実行可能かつその内部演算を再構成可能（プログラミング）な LSI である．ここ数年の半導体技術の進歩により FPGA が大規模化してきたため，これまでは困難であった浮動小数点演算を FPGA で実行できるようになりつつある．我々は，演算用 FPGA が 4 個搭載された演算ボード（Bioler-3 ボード）上で，浮動小数点を使った演算を行うために，仮数部および指数部に任意のビット長が指定可能な FPGA 用浮動小数点演算ライブラリを作成した．さらに，それらの演算ライブラリを組み合わせて GRAPE 型計算機の演算回路を実装するためのソフトウエアである，Processor Generator for Reconfigurable systems（PGR）を開発した[8]．PGR を使うことで，FPGA の搭載された計算ボードで以下の式で表されるような，任意長のベクトル $\boldsymbol{X}_i$ と $\boldsymbol{X}_j$ の関数 $F$ （相互作用関数）の累算を計算するための演算回路を容易に実装できるようになる．

$$F^i = \sum_{j=1}^{N} F(\boldsymbol{X}_i, \quad \boldsymbol{X}_j). \tag{1}$$

式 (1) は，$\boldsymbol{X}_i$ を粒子の位置や速度とすると，粒子シミュレーション一般に現れる加速度の式に帰着する．Bioler-3 ボードは，**図 1** に示されているように，PCI バスでホスト計算機と接続して使用する（PROGRAPE-3 システム）．実際の計算では，ボード上の FPGA で粒子間相互作用の累算（$F^i$）を計算し，ホスト計算機では得られた加速度を使って運動方程式の積分をする，というように役割を分担する．GRAPE 計算機ではその適用範囲は限られていたが，PROGRAPE-3 と PGR を組み合わせた我々のシステムでは，研究者が必要に応じて様々な粒子間相互作用を実装し，彼らのシミュレーションに適用するということが可能になる．本論文では，PROGRAPE-3 システム上での重力相互作用計算と SPH 法相互作用計算の詳細について報告する．

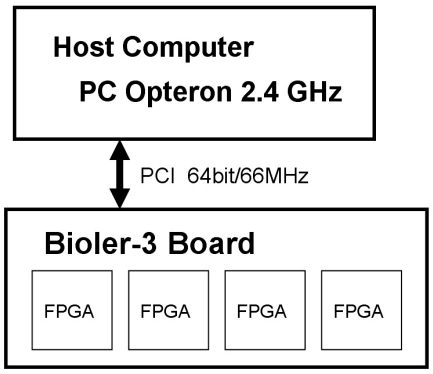



**図 1** PROGRAE-3 システムの概念図
Fig. 1 A schematic view of PROGRAPE-3 system. It consists of a host PC and the Bioler-3 board.

## 2. PGR の概要

PROGRAPE などの FPGA を使った専用計算機で動作する演算回路（演算パイプライン）の設計では，膨大な量のハードウェア回路とソフトウェアを設計および実装しなければならない．我々は，このような状況を改善するために，上記のような設計実装過程を自動化するためのソフトウエアとして PGR を開発している[8]．PGR では，必要な演算機能を PGR 用の言語 PGDL（PGR Description Language）によって記述し，それを PGR によって処理することで，必要なプログラムやソースコードが自動的に生成される．具体的には，(1) 演算パイプラインのソフトウエアエミュレータ，(2) FPGA で動作する演算パイプラインの HDL コード，(3) アプリケーションと結合するためのライブラリ，(4) インタフェースソフト，以上のソースコードが自動的に生成される．たとえば，次章で説明する PGDL による重力相互作用パイプラインの記述は約 30 行であるが，その記述から，PGR によって総計で 5000 行を超える様々なソースコードが自動的に生成される．また，PGR の大きな特徴は，自動的にデータの同期化処理を行い，パイプライン化された演算回路の HDL コードを生成することである．つまり，PGR を使うことによってユーザの労力を大幅に軽減することができる．

PGR では以下のような演算形式を利用できる：(1) 単精度までの任意ビット長の浮動小数点演算，(2) GRAPE などの実装に利用される対数形式による演算，(3) 任意ビット長の固定小数点形式演算．ただし現時点での PGR では，プログラミング可能な問題を式 (1) で表されるような，粒子シミュレーションの加速度計算に限定している．我々はこれまで，以下の章

で説明するように，PGR を使って重力相互作用および SPH 用粒子間相互作用のパイプラインを実装した．ほかにも，分子動力学シミュレーションにおける van der Waars 力や，様々なタイプの多体力も実装可能であると考えている．つまり，PROGRAPE-3 システムと PGR の組合せによって，世界で初めて準汎用的な GRAPE 型計算機が実現できた．

## 3. 重力相互作用パイプラインの実装

ここでは重力相互作用パイプラインを例にとって，PGR におけるプログラミングについて概説する．重力相互作用では，式 (1) における $\boldsymbol{X}_i$ は，粒子 $i$ の質量と座標を $m_i$，$\boldsymbol{r}_i$ とすると，

$$\boldsymbol{X}_i = (m_i, \boldsymbol{r}_i) \tag{2}$$

となる．また，この場合の相互作用関数 $F$ は万有引力の法則となり，具体的に書き下すと，

$$F(\boldsymbol{X}_i, \quad \boldsymbol{X}_j) = -m_j \frac{\boldsymbol{r}_i - \boldsymbol{r}_j}{(|\boldsymbol{r}_i - \boldsymbol{r}_j|^2 + \epsilon^2)^{3/2}}, \tag{3}$$

となる．ここで，$\epsilon$ はゼロ除算による計算結果の発散を防ぐためのパラメータである．結果 $F^i$ は粒子 $i$ に働く重力加速度となる．GRPAE 計算機では，以上のような粒子間相互作用とその累算を計算するパイプラインが専用に開発された LSI チップで並列に動作する．一方で，PROGRAPE-3 システムでは，PGR を使って生成された演算パイプラインが FPGA チップで並列に動作する．GRAPE 型計算機では，各演算パイプラインは粒子 $i$ のデータをその内部レジスタ（$i$–粒子メモリ）に保持し，粒子 $j$ のデータが外部のメモリモジュール（$j$–粒子メモリ）から毎クロック，ブロードキャストにより供給される．GRPAE 計算機では $j$–粒子メモリは独立した SRAM チップ上に実装されているが，PROGRAPE-3 システムでは，$j$–粒子メモリは FPGA 内部の SRAM 上に実装される．以下，演算パイプラインに保持される粒子データを $i$–粒子，演算パイプラインに供給される粒子データを $j$–粒子と呼ぶ．

PGDL による重力パイプライン記述を**図 2** にしめす．ここでは，比較のために専用 LSI に実装されている GRAPE-5 システム[9] と同精度の演算パイプラインの記述例をしめす．GRAPE-5 のパイプラインは，固定小数点形式と対数形式の演算によって構成されている．ここで大文字の NPOS などはマクロ定数として定義されているが，その定義部分は省略した．1 行目の “/NPIPE” ディレクティブは，FPGA チップあたりの演算の並列度を指定している．そして，4–8 行では，アプリケーションプログラムとのインタフェースを定義している．パイプライン自体の定義は 12 行以降に記述されている．各々の行は，1 つの演算あるいは型変換を定義している．たとえば，12 行目はベクトル変数 `xi[3]` と `xj[3]` のそれぞれの成分の差をとり結果を `xij[3]` に代入する，という演算を定義している．これは，式 (3) の分母における粒子間の相対ベクトルを求める操作に対応している．また，`pg_fix_addsub` はこの演算が固定小数点形式の加減算であること示す．同様に，13 行以降に式 (3) の演算式が続き，27 行が式 (1) における結果の固定小数点形式による累算に対応する．なお，13 行と 26 行がそれぞれ固定小数点形式から対数へと対数から固定小数点形式への変換に対応している．つまり，PGDL でのパイプラインの定義は，粒子間相互作用演算を加減乗除算まで分解して 1 行 1 行に並べたものになる．これは，いわゆるアセンブラ記述と類似したものとなっている．この PGDL による記述を PGR によって処理することで，HDL による演算回路のソースコードや，アプリケーションプログラムで利用するライブラリが自動生成される．

```
 1  /NPIPE 16;
 2  /NVMP 1;
 3
 4  /MEM xj[3] <= x[][] : fix(NPOS);
 5  /MEM mj    <= m[]   : log(NLOG,NMAN);
 6  /REG xi[3] <= x[][] : fix(NPOS);
 7  /REG ieps2 <= eps2  : log(NLOG,NMAN);
 8  /REG sx[3] => a[][] : fix(NACC);
 9
10  /SCALE sx[3] : fscale;
11
12  pg_fix_addsub(SUB,xi,xj,xij,NPOS,D_FSU);
13  pg_conv_ftol(xij,dx,NPOS,NLOG,NMAN,D_FTL);
14  pg_log_shift(1,dx,x2,NLOG);
15
16  pg_log_unsigned_add_itp(x2[0],x2[1],x2y2,NLOG,NMAN,D_LAD,NCUT);
17  pg_log_unsigned_add_itp(x2[2],ieps2,z2e2,NLOG,NMAN,D_LAD,NCUT);
18  pg_log_unsigned_add_itp(x2y2,z2e2,r2, NLOG,NMAN,D_LAD,NCUT);
19
20  pg_log_shift(-1,r2,r1,NLOG);
21  pg_log_muldiv(MUL,r2,r1,r3,NLOG,D_LMU);
22  pg_log_muldiv(SDIV,mj,r3,mf,NLOG,D_LMU);
23  pg_log_muldiv(MUL,mf,dx,fx,NLOG,D_LMU);
24  pg_log_expadd(fx,fxs,NEXAD,NLOG,NMAN,D_LMU);
25
26  pg_conv_ltof(fxs,ffx,NLOG,NMAN,NFOR,D_LTF);
27  pg_fix_accum(ffx,sx,NFOR,NACC,D_FAC);
```

**図 2** 重力パイプラインの PGDL による記述例
Fig. 2 PGDL source code example for a gravitational pipeline.

PGR では，演算精度を任意に変更可能であるが，GRAPE-5（理論性能 48.6 GFLOPS）と同精度の演算回路を実装した場合の性能は，以下のような結果となった．この場合，PGR によって PGDL を処理することで，(1) 約 5,500 行の VHDL による演算パイプラインと制御回路のソースコード，(2) 約 1400 行の C 言語によるソフトウエアエミュレータとインタフェースライブラリのソースコードが生成される．生成され

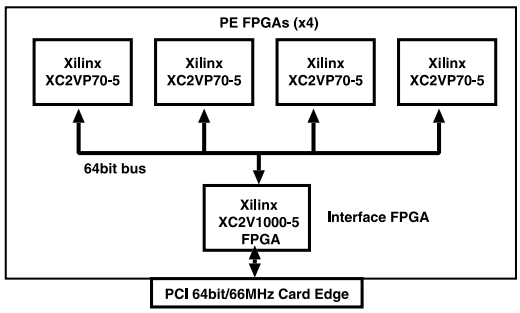



**図 3** Bioler-3 の概略図

Fig. 3 A schematic view of the Bioler-3 board. It has four FPGA chips for computation and one FPGA chip for PCI interface.

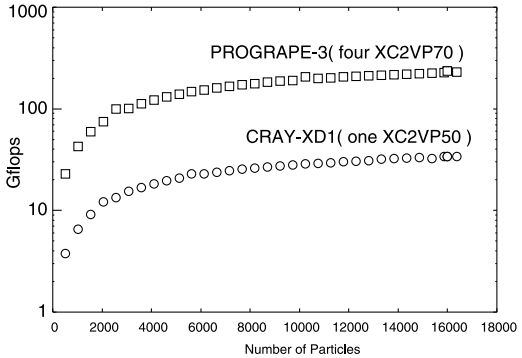



**図 4** GRAPE-5 と同等の演算パイプラインの PROGRAPE-3 上と CRAY XD1 での実行性能

Fig. 4 The performance of the GRAPE-5 pipeline as a function of $N$. The upper squares show the performance with PROGRAPE-3 system. Also, lower circles show the results with a different FPGA system (CRAY XD1).

た VHDL コードは，別途 FPGA 用の論理合成ソフトウエアおよび配置配線ソフトウエアにより処理し，PROGRAPE-3 システムで使用可能となる．生成された VHDL コードを Synplify Pro 7.3.1 で論理合成すると，“/NPIPE 1” の場合 Bioler-3 で使用している FPGA チップ（Xilinx XC2VP70-5）の約 4%の論理回路資源を消費した．この結果から実装可能なパイプラインの本数をある程度見積もることができる．ただし，論理資源の消費量が 80〜90%に達すると，配置配線に非常に時間がかかるようになるか配置配線が不可能になるため，実用上は 70〜80%程度の消費になるように “/NPIPE” ディレクティブを指定する．

以下，ここでは “/NPIPE 16” とした場合の結果を示す．この場合 Synplify Pro 7.3.1 で論理合成すると，約 73%の論理回路資源を消費した．PROGRAPE-3 システムの 1 ボード（**図 3**）あたりでは，64 本のパイプラインが動作する．動作周波数は可変であり，現時点では最高で 133.3 MHz まで正しく動作している．重力パイプライン 1 個あたりの演算数は 38 浮動小数点演算に相当する．よって，1 ボードの PROGRAPE-3 システムを 133.3 MHz で動作させたときの理論性能は，$38 \times 16 \times 4 \times 133.3 \times 10^6 = 324$ GFLOPS 相当となる．GRAPE-5 と比較すると，1 ボードあたりの理論性能では 6.6 倍の性能比が得られた．実際には，ホスト計算機とボード間のデータ転送に要する時間（通信時間）により，実効性能は理論性能より低下する．粒子数（$N$）を変えて測定した実効性能を**図 4** にしめす．図 4 は，相互作用をすべて求める直接計算のアルゴリズムを用いた場合（$i$–粒子と $j$–粒子の数が等しく $N$）の結果である．$N = 16{,}000$ で 234 GFLOPS の演算性能が得られた．この場合，約 80%が FPGA 上での計算時間で，残り約 20%が通信に費やされている．FPGA 上ので計算時間は $O(N^2)$ なのに対し，通信時間は $O(N)$ であるため，$N$ が増えるに従って FPGA 上での計算時間の割合が増加することにより実効性能が改善される．実効性能を GRAPE-5 と比較すると，$N = 10{,}000$ での GRAPE-5 の 1 ボードあたりの実効性能は 27 GFLOPS であり，PROGRAPE-3 の場合には 200 GFLOPS（7.4 倍）となった．実効性能比が理論性能比を上回っているのは，ホストとボード間の実効転送速度の差が要因であり，GRAPE-5 システムでは約 30 MB $\mathrm{sec}^{-1}$ に対して，PROGRAPE-3 では約 100 MB $\mathrm{sec}^{-1}$ である．まとめると，PROGRAPE-3 の理論性能が GRAPE-5 より大幅に向上したのは，1 ボードあたりの動作パイプラインの本数が増えてさらに動作周波数が向上したためであり，実効性能比が理論性能比を上回ったのは，ホスト計算機とボード間の接続に，より高性能のインタフェースである PCI 64 bit/66 MHz を採用したことによる．

PGR では PROGRAPE-3 システムだけでなく，他の FPGA を利用した演算ボードに対応することをめざしている．一例として，CRAY XD1 における重力パイプラインの実効性能を図 4 に示す．この場合，FPGA を 1 チップ（Xilinx XC2VP50-7）の利用で，$N = 16{,}000$ で約 30 GFLOPS の実効性能を得た．つまり，PGR を使うことで，様々な FPGA を利用した演算ボードを活用し，天体物理シミュレーションの高速化が可能になる．

## 4. SPH 法粒子間相互作用パイプラインの実装

前章では，PGR を使うことで，PROGRAPE-3 シ

ステムによって，既存の専用計算機 GRAPE-5 をはるかにしのぐ演算性能が得られることを示した．PROGRAPE の一番大きな利点は，FPGA のパイプラインをプログラミングすることで，様々な用途に転用可能な点である．この章では，SPH 法という天体物理学などで広く応用例がある粒子シミュレーション手法を，PGR によって PROGRAPE-3 システムに実装した結果についてしめす．

SPH 法では，流体力学による相互作用や物理量の評価は式 (1) のような粒子間相互作用の累算に帰着し，粒子の運動方程式を数値積分することで系の進化を計算する．SPH 法には様々な定式化があるが，本研究では以下に示すような定式化を採用した[10]．以下，関数 $W(\boldsymbol{r};h)$ は，$h$ 程度の広がりを持つカーネル関数を表し，本研究では広く使われているスプラインカーネルを採用している．また，$P$ は粒子の圧力を表す．

**密　度：$\rho$**

$$\rho_i = \sum m_j W(\boldsymbol{r}_{ij}). \tag{4}$$

**速度の発散：$\nabla\cdot\boldsymbol{v}$**

$$\rho_i(\nabla\cdot\boldsymbol{v})_i = \sum m_j(\boldsymbol{v}_{ij})\cdot\nabla W(\boldsymbol{r}_{ij}). \tag{5}$$

**速度の回転：$\nabla\times\boldsymbol{v}$**

$$\rho_i(\nabla\times\boldsymbol{v})_i = \sum m_j(\boldsymbol{v}_{ij})\times\nabla W(\boldsymbol{r}_{ij}). \tag{6}$$

**近接粒子数の評価：**

$$\mathrm{n}_i = \sum W_{\mathrm{neighbor}}(\boldsymbol{r}_{ij}). \tag{7}$$

**加速度：$\dot{\boldsymbol{v}}$**

$$\dot{\boldsymbol{v}}_i = -\sum m_j\left(\frac{P_i}{\rho_i^2}+\frac{P_j}{\rho_j^2}+\Pi_{ij}\right)\nabla W(\boldsymbol{r}_{ij}). \tag{8}$$

**エネルギー変化率：$\dot{u}$**

$$\dot{u}_i = \sum m_j\left(\frac{P_i}{\rho_i^2}+\frac{1}{2}\Pi_{ij}\right)(\boldsymbol{v}_{ij})\cdot\nabla W(\boldsymbol{r}_{ij}). \tag{9}$$

**人工粘性項：$\Pi$**

$$\Pi_{ij} = f_{ij}\frac{-\alpha c_{ij}\mu_{ij}+\beta\mu_{ij}^2}{\rho_{ij}}, \tag{10}$$

$$f_{ij} = \frac{f_i+f_j}{2}, \tag{11}$$

$$f_i = \frac{|\nabla\cdot\boldsymbol{v}|}{|\nabla\cdot\boldsymbol{v}|+\nabla\times\boldsymbol{v}+10^{-4}c_i/h_i} \tag{12}$$

$$\mu_{ij} = \frac{h_{ij}\boldsymbol{v}_{ij}\cdot\boldsymbol{r}_{ij}}{(\boldsymbol{r}_{ij})^2+(0.1h_{ij})^2}. \tag{13}$$

ここで，$\boldsymbol{r}_{ij}=\boldsymbol{r}_i-\boldsymbol{r}_j$，$\boldsymbol{v}_{ij}=\boldsymbol{v}_i-\boldsymbol{v}_j$ であり，$c$ は粒子における音速を表し，$h_{ij}$，$c_{ij}$ と $\rho_{ij}$ はそれぞれ，$i$–粒子と $j$–粒子による，$h$，音速と密度の平均値（$h_{ij}=(h_i+h_j)/2$ etc.）を表す．また式 (7) は，Thacker ら[11] の手法に従って近接粒子数の更新に使用される．$W_{\mathrm{neighbor}}$ はその評価に使われるカーネル関数である．

以上のような SPH 法粒子間相互作用を，以下の 2 つのステージに分割して実装した．

- 第 1 ステージ
  式 (4)，(5)，(6) と (7) を計算．
- 第 2 ステージ
  式 (8) と (9) を計算．その際に (10)–(13) も計算される．

これは第 2 ステージの $\dot{\boldsymbol{v}}$ と $\dot{u}$ が，現タイムステップでの $\rho$，$\nabla\cdot\boldsymbol{v}$ と $\nabla\times\boldsymbol{v}$ に依存しているため，以上のように分割する必要がある．各ステージでの浮動小数点演算数は，それぞれ 70，73 演算となる．重力パイプラインは固定小数点形式演算および対数形式演算で構成したが，SPH パイプラインは，浮動小数点演算によって構成した．各ステージに対応する PGDL の記述量はそれぞれ約 200 行であった．PGR で処理した結果，それぞれのステージに対して約 8,600 行の VHDL ソースコードが生成された．また，約 3,300 行の C 言語によるソフトウエアエミュレータとインタフェースライブラリのソースコードが生成された．式 (3) と式 (4)–(13) までを比較すると明らかなように，SPH 法の定式化は重力に比べて格段に複雑であり，GRAPE 計算機のように専用 LSI 上に実装することは，開発コストと時間のうえで困難であった．PGR を使うことで，SPH 法程度に複雑な浮動小数点演算回路を，200 行程度の PGDL を書くだけで容易に実装可能になった．

### 4.1 演算精度の評価

一般に，浮動小数点演算を LSI に実装する場合，仮数部のビット幅を 2 倍にすると必要な回路資源は約 4 倍になる．これは FPGA に実装する場合でも同様であり，FPGA では資源の制約がより厳しいため，必要な精度を見極めたうえで，仮数部のビット幅はできるだけ小さいほうが効率が良い．PGR における浮動小数点演算では，仮数部は単精度（23 ビット）までの任意の精度を選ぶことができる．ここでは，必要な精度を見積もるため，テスト問題を様々な演算精度で解くことで，仮数部のビット幅が演算結果にどのような影響を及ぼすか調べた．テスト問題としては，流体計算でよく使われる Sod の衝撃波管問題を選んだ．**図 5** にその結果をしめす．この図は，Sod の衝撃波管問題を，仮数部のビット幅を変えた SPH パイプラインによって，$t=0.15$ まで進化させた場合の密度の

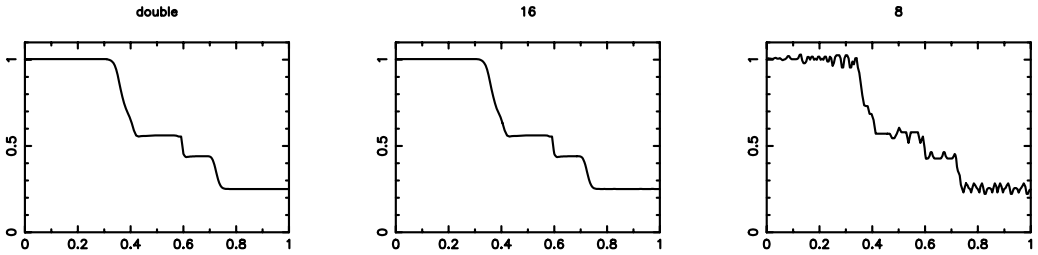



**図 5** Sod の衝撃波管問題の $t = 0.15$ における密度のスナップショット．左のパネルに，ホストにおける倍精度演算による結果をしめす．中央と右のパネルはそれぞれ仮数部を 16，8 ビットにした場合の結果をしめす

Fig. 5 Density snapshots at $t = 0.15$ for the Sod's shock tube test. The left panel shows a result with double double precision calculation on the host computer. The center and right panels show results with mantissa bit width of 16-bit, and 8-bit, respectively.

スナップショットである．左のパネルはホスト計算機で倍精度演算により計算した結果である．真ん中のパネルは仮数部のビット幅を 16 ビットとした場合，右のパネルは 8 ビットとした場合の結果で，これらは PGR によって生成されたソフトウエアエミュレータによって計算した．このとき，いずれの場合にも，指数部のビット幅は 8 ビットに固定している．図 5 から，仮数部が 8 ビットではあきらかに演算精度が不足しており，人工的な振動がみられることが分かる．一方で，16 ビットの場合にはそのようなことはなく，倍精度の場合と比べてもほぼ遜色ない．実際，16 ビットでの結果を倍精度での結果と比べた場合の誤差は，粒子の位置，密度，内部エネルギーそれぞれに対し，$2.98 \times 10^{-6}$，$1.33 \times 10^{-4}$，$3.77 \times 10^{-5}$ となった．また，衝撃波管問題の厳密解を使って誤差を評価したところ，密度と内部エネルギーについて，倍精度演算では $2.556 \times 10^{-2}$ と $1.003 \times 10^{-2}$ であり，仮数部 16 ビットでは $2.548 \times 10^{-2}$ と $9.995 \times 10^{-3}$ となった．つまり，実際上は，以上のように，SPH 法のスキーム自体による誤差が大きいという結果になった．以下，本研究では SPH パイプラインの実装に仮数部 16 ビット，指数部 8 ビットの浮動小数点演算を採用することとした．

### 4.2 実装の詳細

PGR で生成され PROGRAPE-3 システムで実行する SPH パイプラインと，重力パイプラインとの大きな違いは，重力相互作用は長距離力であるのに対して，SPH による相互作用は短距離力であるという点である．本研究で採用しているカーネル関数は $r > 2h$ ではゼロになるため，$i$–粒子を中心に半径 $2h$ の球内（3 次元の場合）に含まれる $j$–粒子（近接粒子）だけが，$i$–粒子に関する式 (4)–(9) における累算に寄与する．そのため，汎用計算機での SPH 法コードでは，各粒子に対する近接粒子検索をどのように実装するかでコードの速度が大きく異なってくる．よって，PROGRAPE-3 システム上の SPH パイプラインを使う場合には，以下のような 3 種類の実装が考えられる．

**直接計算アルゴリズム**

(1) 初期設定.
(2) 第 1 ステージ用の全粒子のデータをボードへ送る.
(3) 全粒子に関して第 1 ステージパイプラインを使って計算をし結果を回収.
(4) 状態方程式の計算．さらに，第 2 ステージ用データの準備.
(5) 第 2 ステージ用の全粒子のデータをボードへ送る.
(6) 全粒子に関して第 2 ステージパイプラインを使って計算をし結果を回収.
(7) 後処理.

$N$ が SPH パイプラインが保持できる粒子数（$N \sim 8{,}000$）と同程度の場合には，式 (4)–(9) における累算を，重力と同じように全粒子対全粒子で計算する．この場合，$i$–粒子と $j$–粒子の数は等しく $N$ であり，通信時間は $O(2N)$，演算時間は $O(N^2)$ となる．

**近接粒子アルゴリズム A**

不必要な演算をできるだけ省略するため，ホスト計算機で近接粒子探索を行ってから，FPGA ボードによって計算する．

(1) 初期設定.
(2) $i = 1$ から $N$（すべての粒子）に対して，以下を実行する.
  (a) $i$–粒子の近接粒子を探索.
  (b) 得られた $i$–粒子の近接粒子データをボー

ドへ送る．
(c) $i$–粒子に関して第 1 ステージパイプラインを使って計算をし結果を回収．
(3) 状態方程式の計算．さらに，第 2 ステージ用データの準備．
(4) $i = 1$ から $N$ に対して，以下を実行する．
(a) 第 2 ステージ用の近接粒子のデータをボードへ送る．
(b) $i$–粒子に関して第 2 ステージパイプラインを使って計算をし結果を回収．
(5) 後処理．

この場合，第 1 ステージでの近接粒子探索の結果は保存しておき，第 2 ステージの際に再利用する．このアルゴリズムを使えば，原理的には $N$ に関する制限はなくなる．なお，本研究では近接粒子探索には Tree アルゴリズムを採用している．この場合には，$i$–粒子の数が $N$ である一方で，$j$–粒子の数は $aN$ となる．ここで $a$ は平均的な近接粒子数で，3 次元 SPH シミュレーションの場合典型的には $a \sim 30 - 100$ である．よって，単純には通信時間は $O(N + aN)$，演算時間は $O(aN)$ となる．パイプラインの利用効率を上げるために，近接粒子どうしで近接粒子リストを共有するなどの最適化をすれば，通信時間は $O(N + bN)$ ($b < a$ で典型的には $3 - 10$ 程度）に減少する．

**近接粒子アルゴリズム B**

“近接粒子アルゴリズム A” では，個々の粒子ごとに近接粒子探索を行うが，GRAPE を使った Tree 法[12] と同じように，個々の粒子ではなく “粒子の集団” に対して近接粒子探索を行い，近接粒子探索の回数を減らしたほうが計算時間が短くなる場合がある．この場合のアルゴリズムは以下のようになる．

(1) 初期設定．
(2) 近接粒子探索用の Tree を構築．
(3) Tree を使い，粒子の集団のリストを作成．
(4) すべての粒子の集団に対して，以下を実行する．
(a) “粒子の集団” の近接粒子を探索．
(b) 第 1 ステージ用の近接粒子のデータをボードへ送る．
(c) “粒子の集団” に含まれる粒子に関して，第 1 ステージパイプラインを使って計算をし結果を回収．
(5) 状態方程式の計算．さらに，第 2 ステージ用データの準備．
(6) すべての粒子の集団に対して，以下を実行する．
(a) “粒子の集団” の近接粒子を探索．
(b) 第 2 ステージ用の近接粒子のデータをボードへ送る．
(c) “粒子の集団” に含まれる粒子に関して，第 2 ステージパイプラインを使って計算をし結果を回収．
(7) 後処理．

“近接粒子アルゴリズム A” では，近接粒子探索の回数は $N$ 回である．一方で，近接粒子アルゴリズム B では，粒子の集団に含まれる粒子数の平均値を $n_{\mathrm{group}}$ とすると，近接粒子探索の回数は $N/n_{\mathrm{group}}$ 回となり，ホスト計算機で必要な演算量が減る．ただし，FPGA ボードでの演算量は “近接粒子アルゴリズム A” にくらべて増える．これは以下のような理由による．アルゴリズム A と B との大きな違いは，A では各粒子ごとに近接粒子探索をするのに対し，B では粒子をまとめて，その「粒子の集団」ごとに近接粒子探索をする点である．こうすることで，近接粒子探索の回数を減らすことができる．一方で,「粒子の集団」に対して近接粒子探索をする場合には，粒子の集団を完全に含む球を求め，その球の中心と半径を使って，近接粒子探索を行うため，近接粒子数の総計が A の場合より大きくなる．$n_{\mathrm{group}}$ は，ホスト計算機の速度やボードとの通信速度，SPH パイプラインの速度に応じて変更すべきパラメータであり，我々のシステムでは $n_{\mathrm{group}} = 100 - 200$ が最適値であった．この場合には，最適化された近接粒子アルゴリズム A と同様に，通信時間は $O(N + bN)$，演算時間は $O(aN)$ になる．

### 4.3 性能評価

生成された VHDL コードを Synplify Pro 7.3.1 で論理合成すると，“/NPIPE 1” の場合，第 1 ステージの演算パイプラインでは約 44%，第 2 ステージの演算パイプラインでは約 40%の論理回路資源を消費した．よって，以下，“/NPIPE 2” とした場合の結果を示す．

PGR を利用して我々が FPGA に実装した SPH パイプラインは，すべての演算がパイプライン演算方式で計算される．そのため，我々が実装した SPH 法第 1 ステージの演算パイプラインの演算性能は，1 クロックあたり 80 浮動小数点演算相当となる．FPGA の動作周波数は可変であり，最高 133.3 MHz まで正しく動作した．よって，この場合のボードあたりの理論性能は，$80 \times 2 \times 4 \times 133.3 \times 10^6 = 85$ GFLOPS 相当となる．実際には，以下の要因により演算性能が若干低下することになる：(1) ホスト計算機とボード間のデータ転送による待ち時間，(2) SPH 法では近接粒子間とのみ相互作用するため，ホスト計算機での近似的

な近接粒子探索にかかる時間.

前節の “直接計算アルゴリズム” で実効性能を測定した結果を**図 6** にしめす．これは，第 1 ステージの演算パイプラインを 1 ボードあたり 8 パイプラインを使用し，$N$ 粒子に対する加速度の演算にかかった時間を $N$ の関数としてプロットした．破線が動作周波数を 66.6 MHz とした場合，実線は動作周波数を 133.3 MHz とした場合の結果である．図から明らかなように，いずれの場合にも，$N > 6{,}000$ では，理論性能の約 80%の実効性能が得られている．$N = 8{,}000$ で動作周波数 133.3 MHz の場合，通信時間，FPGA での計算時間の割合は，それぞれ 14%，86%であった．一方で，ホスト計算機（Opteron 2.4 GHz）の倍精度演算における理論性能は 4.8 GFLOPS であり，SPH 法コードによる実効性能は約 1 GFLOPS であった．つまり，PROGRAPE-3 システムによる SPH 法の演算性能は，ホスト計算機と比べて数十倍以上速いということになる．

実用上は “直接計算アルゴリズム” では無駄な演算が多くなるため，より現実的な性能評価として，“近接粒子アルゴリズム B” の場合の結果を**図 7** にしめす．この場合 Bioler-3 上の 2 つの FPGA チップで第 1 ステージのパイプライン（計 4 パイプライン）を動作させ，残りの 2 チップでは第 2 ステージのパイプライン（計 4 パイプライン）を動作させている．このテストでのパイプラインの動作周波数は 133.3 MHz である．実線は，FPGA ボードを使った場合の，加速度の演算にかかった時間を $N$ の関数としてプロットした．破線は，FPGA ボードを使わずにホスト計算機のみで同じ計算をした場合の結果である．$N = 500{,}000$ で 1 ステップの計算時間で比べると，ホスト計算機のみでは約 38 秒に対し，PROGRAPE-3 システムでは，ホストでの近接相互作用検索を含めて約 8 秒となった．よって，十分 $N$ が十分大きいならば（$N > 100{,}000$），PROGRAPE-3 システムを使うことで，SPH シミュレーションを約 5 倍高速に実行できるということになる．ここで，“近接粒子アルゴリズム B” では，$N = 500{,}000$ のとき，通信時間，FPGA での計算時間，ホストでの計算時間の割合は，それぞれ 61%，22%，17%であった．よって，より大きな加速率を得るためのおおまかな指針は，ホスト計算機と FPGA ボード間の通信速度をより高速にすることである．また，“近接粒子アルゴリズム B” の場合にはホストにおける近接粒子探索のさらなる最適化も必要である．

### 4.4 考察と今後の展望

“近接粒子アルゴリズム B” では，前節にあるように，ホスト計算機から Bioler-3 ボードへのデータ転送時間の割合が約 60%と大きな割合を占めている．SPH パイプラインの場合，ホストからボードへの実効転送速度（倍精度から内部浮動小数点形式への変換作業を含む）は，145 MB $\mathrm{sec}^{-1}$ であり，ボードからホストへの実効転送速度（同上）は 37 MB $\mathrm{sec}^{-1}$ である．これら結果は，PCI 64 bit/66 MHz インタフェースの理論転送速度である 512 MB $\mathrm{sec}^{-1}$ を大きく下回っている．

ここで，PROGRAPE-3 システムの，浮動小数点形式の変換を含まない最大のデータ転送速度は，ホストからボードへの転送（以下，書き込み）の場合約

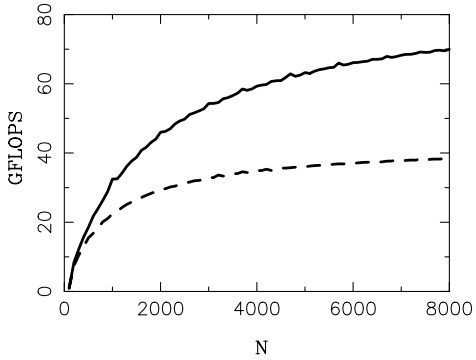



**図 6** 第 1 ステージパイプラインの実効性能（GFLOPS）．破線がクロックを 66.6 MHz とした場合，実線はクロックを 133.3 MHz とした場合の結果を示す

Fig. 6 Obtained performance (in GFLOPS) of SPH first stage pipelines as a function of $N$ is shown in the solid line (clock speed at 66.6 MHz) and the dashed line (133.33 MHz).

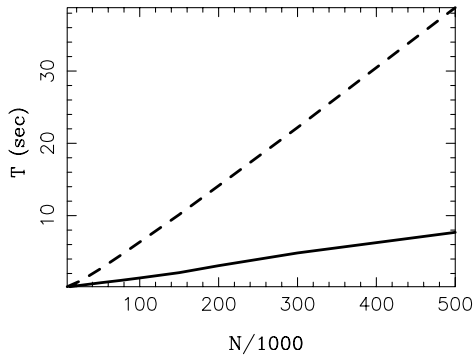



**図 7** “近接粒子アルゴリズム B” による演算性能．実線が PROGRAPE-3 システムを利用した場合，破線がホスト計算機で計算した場合の 1 ステップあたりの計算時間を示す

Fig. 7 The solid line shows calculation time per one step with the neighbor algorithm B as a function of $N$. As a comparison, the dashed line shows calculation time for the same calculation adobe but with the host computer.

220 MB $sec^{-1}$ であり，ボードからホストへの転送（以下，読み出し）の場合約 400 MB $sec^{-1}$（転送サイズ 64 K バイトの場合）であった．現システムでは，ホストからボードへは Programmed I/O（PIO）転送を利用し，ボードからホストへはボードがマスタの DMA 転送を利用している．現システムの場合，DMA 転送速度は転送サイズに強く依存しており，SPH パイプラインの場合は，転送サイズは 512 バイトである．このとき，浮動小数点形式の変換を含まないデータ転送速度は約 110 MB $sec^{-1}$ に低下する．そのため，読み出しの転送サイズを大きくするような最適化の余地もある．まとめると，PROGRAPE-3 で動作する SPH パイプラインの場合，書き込みの速度はほぼハード的な限界程度まで引き出しているが，読み出しの速度に向上の余地が大きくあるということになる．

今後新たな FPGA ボードで SPH パイプラインを動作させる場合には，以下のように書き込みと読み出しの速度がどちらも向上するハードウェアを設計することが望ましい．“近接粒子アルゴリズム B” の場合，書き込みデータ量は $O(bN)$ であるのに対して，読み出しデータ量は $O(N)$ である．ここで，$b$ は典型的には 3 – 10 の数であり，今の場合は $b \sim 4$ であった．一方で，PROGRAPE-3 での “近接粒子アルゴリズム B” によるデータ転送時間の書き込みと読み出しの内訳は，書き込みが 63%であり，読み出しは 37%であった．大まかには，書き込みデータ量が数倍以上多いにもかかわらず，読み出し時間が約 40%を占めていることとなる．よって，将来のハードウェアおよびデータ転送回路の実装の際には，読み出しの高速化や最適化だけでなく，書き込みが高速であることが肝要となる．その際，PCI-X インタフェースや PCI-Express インタフェースの採用が必要となると思われる．また，現システムのホスト計算機では，書き込みに DMA 転送を利用すると転送速度が遅いため，書き込みに PIO 転送，読み出しに DMA 転送を使用しているが，本来は書き込み読み出しともボードがマスターとなる DMA 転送によって実装することが望ましい．

## 5. ま と め

本論文では，FPGA を利用した専用計算機である PROGRAPE-3 システムにおける天体物理学計算の高速化について報告した．天体物理学では重力多体問題専用計算機 GRAPE が非常に大きな成功と成果をあげてきた．PROGRAPE-3 システムでは，GRAPE 計算機と同様の構成をとりその利点を生かしながら，専用 LSI に演算パイプラインを実装していた GRAPE とは異なり，FPGA に演算パイプラインを実装する．そのため，原理的にはどのような演算でも実行可能である．ただし，FPGA のプログラミングには通常のプログラミングとはまったく異なる手法が必要なため，我々は，研究者が FPGA に任意の粒子間相互作用パイプラインを実装するためのソフトウエア PGR を開発した．PGR を使うことで，FPGA に演算パイプラインを実装する労力は大幅に軽減される．

本論文では，PGR の応用例として，まず重力相互作用演算パイプラインの実装について述べた．PGR で実装された重力パイプラインを PROGRAPE-3 システムで動作させると，GPAPE-5 計算機と比べて理論性能で 6.6 倍，実効性能ではそれ以上の優れた性能を得ることができた．PGR は PROGRAPE-3 システムだけではなく，様々な FPGA を使った計算ボード（たとえば Cray XD1）用の演算パイプラインを生成することが可能である．また，PGR を使うことで，重力相互作用より格段に複雑な SPH 法演算パイプラインを実装することができた．PGR によって自動生成されたソフトウエアエミュレータを使い，衝撃波管問題を，仮数部のビット幅を変化させて解くことで，SPH パイプラインに必要な演算精度を見積もった．我々の SPH パイプラインの理論性能は，ボード 1 枚あたり約 85 GFLOPS に相当する．一方で，現実的な設定でホスト計算機と比較した場合の加速率は約 5 倍に達した．SPH 法専用計算機の試みはこれまでいくつかあったが[6),13)]，実用的な SPH 法専用計算機の開発は本論文の結果が世界初である．

我々のこれまでの研究結果は，FPGA で浮動小数点演算を実行し，粒子シミュレーションを高速化可能であることを実証した．また，これにより PROGRAPE-3 システムと PGR の組合せによって，初めて準汎用的な GRAPE 型計算機が実現できた．今後 PGR を使うことで，PROGRAPE-3 システムだけでなく，Cray XD1 や他の FPGA を搭載した計算ボードを使って，様々なシミュレーションを高速化できるようになる．我々は，このような FPGA による浮動小数点演算の高速化が，新しい計算パラダイムとして広がっていくであろうと考えている．

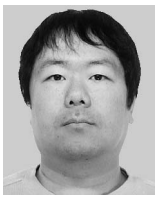
**謝辞** 本研究遂行にあたっての，理化学研究所中央研究所の戎崎俊一主任研究員，千葉大学工学部の伊藤智義教授，および東京大学大学院総合文化研究科の福重俊幸助手からの協力，助言と議論に感謝します．本研究の一部は，情報処理推進機構による 2004 年度上期および 2005 年度上期未踏ソフトウエア創造事業の支援を受けている．また本研究の一部は，中里が理化

学研究所基礎科学特別研究員として遂行した．

## 参 考 文 献


1) Sugimoto, D., Chikada, Y., Makino, J., Ito, T., Ebisuzaki, T. and Umemura, M.: A Special-Purpose Computer for Gravitational Many-Body Problems, *Nature*, Vol.345, pp.33–35 (1990).
2) Makino, J. and Taiji, M.: *Scientific Simulations with Special-Purpose Computers — The GRAPE Systems*, John Wiley and Sons, New York (1998).
3) Fukushige, T., Taiji, M., Makino, J., Ebisuzaki, T. and Sugimoto, D.: A Highly Parallelized Special-Purpose Computer for Many-Body Simulations with an Arbitrary Central Force: MD-GRAPE, *Astrophysical Journal*, Vol.468, pp.51–61 (1996).
4) Lucy, L.: A numerical approach to the testing of the fission hypothesis, *Astronomical Journal*, Vol.82, pp.1013–1024 (1977).
5) Gingold, G. and Monaghan, J.J.: Smoothed particle hydrodynamics — Theory and application to non-spherical stars, *Monthly Notice of Royal Astronomical Society*, Vol.181, pp.375–389 (1977).
6) Yokono, Y., Ogasawara, R., Inutsuka, Y., Chikada, S. and Miyama, S.: Development of a Special Purpose Computer for Cosmic Hydrodynamics Using the SPH Method, *Proc. International Conference on Numerical Astrophysics 1998*, pp.429–430 (1999).
7) Hamada, T., Fukushige, T., Kawai, A. and Makino, J.: PROGRAPE-1: A Programmable, Multi-Purpose Computer for Many-Body Simulations, *Publication of Astronomical Society of Japan*, Vol.52, pp.943–954 (2000).
8) Hamada, T. and Nakasato, N.: PGR: A Software Package for Reconfiguable Super-Computing, *Proc. FPL 2005*, pp.366–373 (2005).
9) Kawai, A., Fukushige, T., Makino, J. and Taiji, M.: *Publication of Astronomical Society of Japan*, Vol.52, p.659 (2000).
10) Navarro, J.F. and White, S.D.M.: Simulations of Dissipative Galaxy Formation in Hierarchically Clustering Universes — Part One — Tests of the Code, *Monthly Notice of Royal Astronomical Society*, Vol.271, pp.271–300 (1993).
11) Thacker, R.J., Tittley, E.R., Pearce, F.R., Couchman, H.M.P. and Thomas, P.A.: Smoothed Particle Hydrodynamics in cosmology: a comparative study of implementations, *Monthly Notice of Royal Astronomical Society*, Vol.319, pp.619–648 (2000).
12) Makino, J.: Treecode with a Special-Purpose Processor, *Publication of Astronomical Society of Japan*, Vol.43, pp.621–638 (1991).
13) Lienhart, G., Kugel, A. and Männer, R.: Using Floating Point Arithmetic on FPGAs for Accelerating Scientific N-Body Simulations, *Proc. IEEE FCCM 2002 Symposium* (2002).


(平成 17 年 10 月 4 日受付)
(平成 18 年 1 月 20 日採録)



**中里 直人**（正会員）

昭和 47 年生．平成 12 年東京大学大学院理学系研究科天文学専攻博士課程修了．平成 13 年から 16 年にかけて日本学術振興会特別研究員．現在理化学研究所基礎科学特別研究員．数値シミュレーションによる天体物理学計算について研究．理学博士．日本天文学会，IAU 各会員．

**濱田 剛**（正会員）

昭和 49 年生．平成 16 年東京大学大学院総合文化研究科広域科学専攻博士課程単位取得退学．同年理化学研究所勤務．FPGA を用いた計算エンジンの研究開発に従事．教養博士．日本天文学会会員．