GRAPE-DR ボードの開発状況

牧野淳一郎 (国立天文台)
福重俊幸 (K & F Computing Research)
藤野 健 (東京大学)

今日の話の構成

- GRAPE-DR について
- 開発計画
- 開発状況
- まとめ

要点:
PCI-Express 評価ボードできた。量産型ボード設計大体終わった。年内に動作すると嬉しいな。

GRAPE-DR 計画とは何か？

「基本的には」次期 GRAPE 計画

- 2004 年度から 5 年計画
- 目標ピーク性能: 2 Petaflops
- チップ数 4096
- 単体チップ性能 0.5Tflops

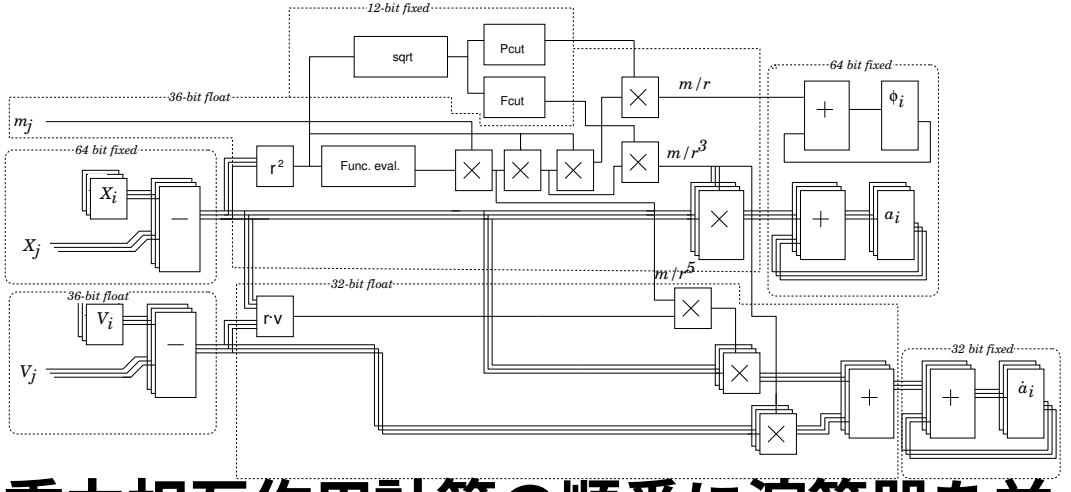
と、これだけなら今までの GRAPE が速くなっただけ。

実際のアーキテクチャ: 今までの GRAPE とは全然違う

- なぜ違うか
- それで何ができるか

GRAPE とはどんなものだったか？

プロセッサアーキテクチャ



重力相互作用計算の順番に演算器を並べたパイプライン

- シリコンの利用効率は極めて高い
- 動作クロックも上げやすい

- アプリケーション限られる。多種類作るのはリソースがかかり過ぎる

「次期 GRAPE」の実際的な問題

天文だけ (しかも理論だけ (しかも N 体だけ)) でもらうにはチップ開発コストが大き過ぎる

チップ開発費

1990	1μm	1500 万円
1997	0.25μm	1 億円
2004	90nm	3 億円以上？
2006	65nm	10 億円以上

ある程度広い応用を持つものでないと予算獲得が難しい

GRAPE-DR の基本的な考え

- なんらかの方法でプログラム可能であるということにして予算を獲得する。
- でも、簡単にできることしかやらない。
- それでできるようなアプリケーション・アルゴリズムを考える。

もうちょっとそれらしく言うと:

- 応用に特化し、多数の演算器を１チップに集積、並列動作させて高い性能を得た専用計算機の特徴を生かす
- しかし広い応用範囲を実現する

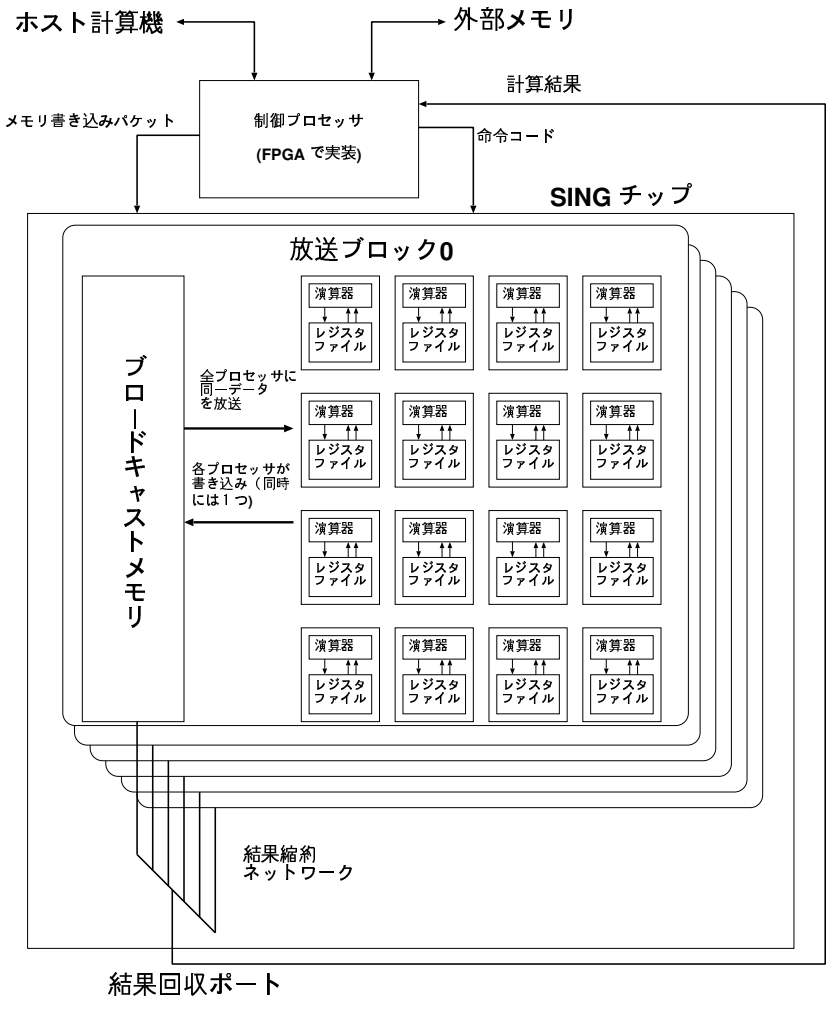
SIMD 並列処理

- パイプラインプロセッサをやめにして、「プログラム可能なプロセッサ」を沢山載せる。
- SIMD (Single Instruction Multiple Data): 全プロセッサが同じ命令を実行

計算機科学の専門家から見ると、「失敗した過去の技術」

- 牧野が作文したのでは予算とれなかった

GRAPE-DR におけるSIMD 並列処理



- 非常に多数のプロセッサエレメント (PE) を 1 チップに集積
- PE = 演算器 + レジスタファイル (メモリをもたない)
- チップ内に小規模な共有メモリ (PE にデータをブロードキャスト)。これを共有する PE をブロードキャストブロック (BB) と呼ぶ。
- 制御プロセッサ、外部メモリへのインターフェースを持つ

何ができるか？

- 今までの GRAPE の代わり
  - 重力計算
  - 分子間力
  - 離散フーリエ変換 (Ewald 法)
  - SPH
- もっと他のこと (現在開発進行中のもの)
  - 密行列計算
  - 量子化学計算
  - 多倍長計算
  - メッシュでの流体等？
  - それ以外

プログラム可能なので、結構なんでもできる。

開発計画

- 2004 年度: プロセッサ論理設計
- 2005 年度: プロセッサ物理設計、試作
- 2006 年度: 評価ボードによる動作確認、量産ボード試作
- 2007, 8 年度: 量産、アプリケーションチューニング

量産ボード試作ちょっと遅れた。安い FPGA がでるのを待ってたため。

開発状況

PCI-Express x8、GRAPE-DR 1チップ。
一応動いている
FPGA はまだ高価なもの。

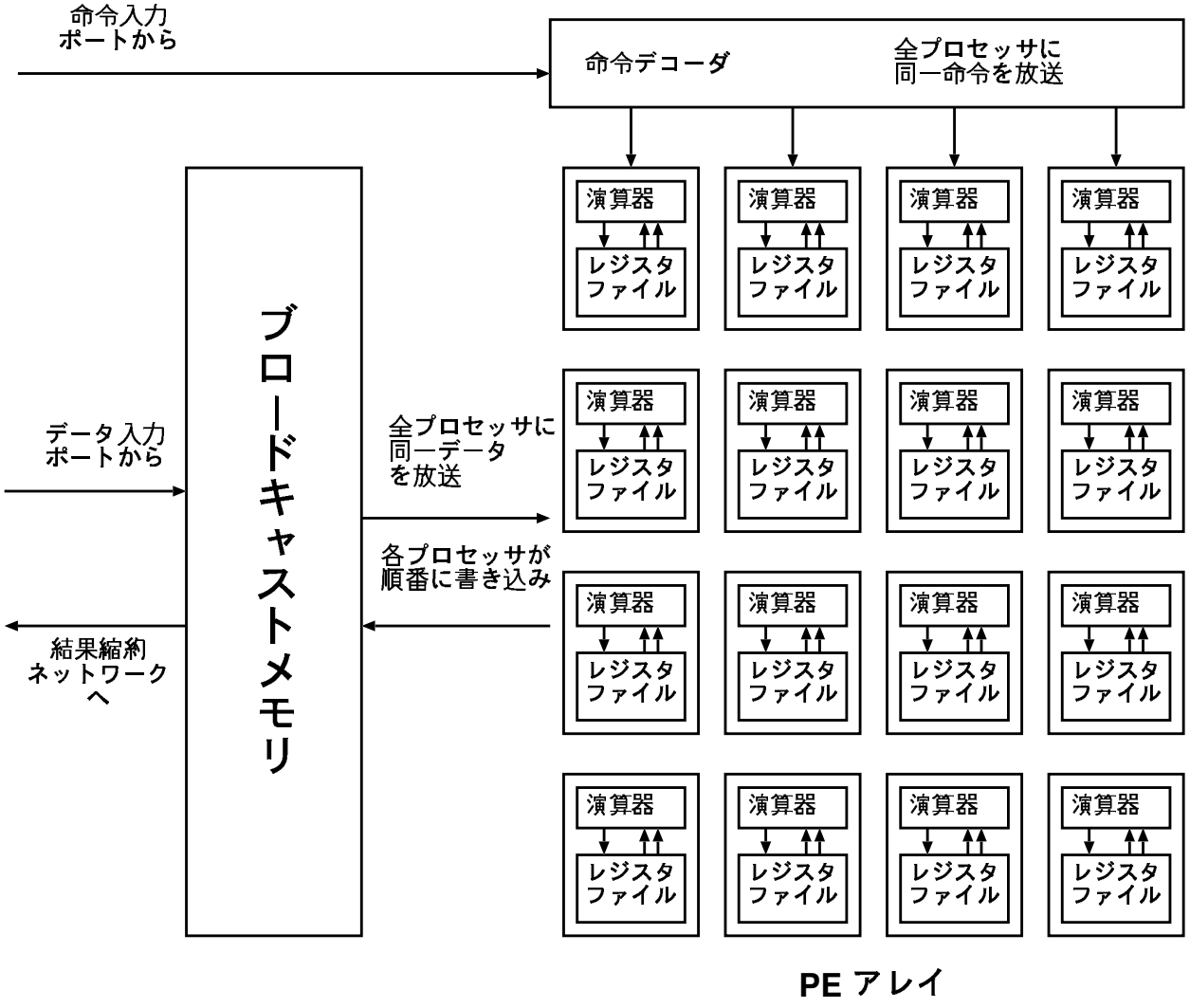
量産型ボード

- 制御プロセッサに安い FPGA (Altera Arria GX) を使用。
- 1ボードに GDR チップ+FPGA 4組。
- それぞれから PCI-Express x4
- それらを PCI-Express スイッチチップで x16 にまとめてホストに接続
- 回路設計終わった。現在基板設計中。
- ピーク性能 2Tflops、通信速度ピーク 4GB/s x 2

まとめ

- GRAPE-DR は 2008 年度完成、単精度ピーク 2Pflops を目指す次世代 GRAPE 計画
- 従来の GRAPE と違ってプログラム可能なので重力計算以外にも色々できる
- 量産型ボードがもうすぐできる

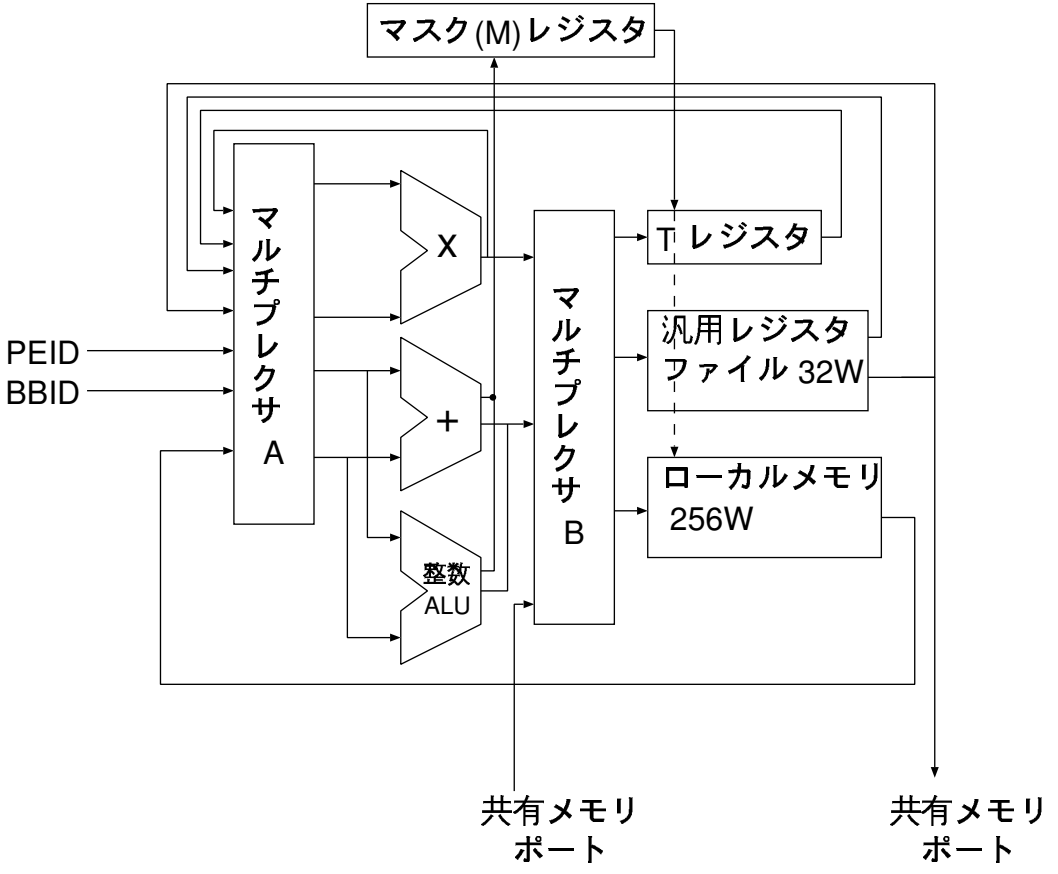
ブロードキャストブロック



各プロセッサは同じデータをブロードキャストメモリから受け取る

このブロックがチップ内には沢山ある。答はブロックにまたがって合計することもできる。

PE の構造



- 浮動小数点演算器
- 整数演算器
- レジスタ
- メモリ (256 語),K とか M ではない。

PEの詳細

データ形式

単精度浮動小数点: 36 ビット (符号1、指数11、仮数24)
倍精度浮動小数点: 72 ビット (符号1、指数11、仮数60)
36/72ビット固定小数点数

演算命令

乗算は単精度のみ (倍精度のための部分積をサポート) 倍精度乗算を 2 サイクルでするために 25×50 ビットの乗算器
整数演算、加減算は倍精度のみ (メモリ/レジスタからの読出し/格納時に単・倍変換ができる)
特殊な浮動小数点命令: 仮数を正規化しないまま演算を続ける。これにより、演算順序によらないで結果が同じになることを保証する (GRAPE-6 の積算と同様)
普通に正規化もできる (こっちがデフォルト)

PEの詳細(続き)

- パイプラインは8ステージ。
- 基本命令は4データに対するベクトル命令。4サイクルに1回しか命令ははいらない。
- T レジスタのみ直前の命令の実行結果を利用可能。
- T レジスタはアドレスレジスタになる(間接アクセス)

サポートする命令等は基本的には昔の SIMD 計算機、例えば CM-2, MasPar MP-1 なんかとあまり変わらない。但し、PE がはるかに強力になっている。

散乱実験型

- 多数の PE が、独立にイベントを計算
  - イベント間の相互作用はない、または非常に少ない
    - レイトレース計算：光学部品（レンズ、導光版）設計
    - 放射線伝播のモンテカルロ計算：検出器設計
    - 3 体問題:連星と単独星の遭遇、微惑星同士の遭遇
- “Embarassingly Parallel” とほぼ対応
- 古典的 SIMD 機と同様の振る舞い:
  - Goodyear MPP, ICL DAP, TMC CM-1/2, Maspar MP-1/2
  - 極端に少ないメモリ
  - PE 間通信が遅い
- 計算速度と通信速度の比:
  - 散乱実験の計算がどれだけ複雑かで決まる

粒子間相互作用型

$$f_i = \sum_j f(x_i, x_j)$$

- 他の「粒子」との「相互作用」を縮約。
  - 全ての相互作用を並列に計算可能
  - 同じ「粒子」のための計算結果を高速に縮約する必要
- 計算手順
  - PE に相互作用を受ける粒子をロード
  - 相互作用を及ぼす粒子をロード
  - 計算機終了したら結果を縮約しながら回収
  - 計算速度とチップ外への通信速度の比:
    相互作用を及ぼす粒子数に比例

密行列型

$$c_{ij} = \sum_{k} a_{ik} b_{kj}$$

- 計算手順
  - 行列が PE に収まるところまで分割。それから
  - 行列 A の部分行列を PE にロード
  - B の1列を分解して各グループにロード
  - 各 PE で B の部分列と A の部分行列の積を計算
  - 計算が終わったものから順次回収。グループ間で合計
- 計算速度・通信速度の比はチップ全体にロードできる行列のサイズに依存
  - メモリサイズの平方根に比例して通信速度を落とせる

計算・通信比のまとめ

- 散乱実験型: アプリケーション依存
- 粒子間相互作用型: 粒子数依存
- 密行列型: オンチップメモリサイズ依存
- 設計におけるトレードオフ:
  - なるべくアプリケーション範囲を広く
    - メモリを多く、バンド幅を広く $\rightarrow$ コスト増
  - コストを圧迫しないようにバランスを考える必要あり
- 実際の設計では密行列型の要求がもっとも厳しい

How do you use it?

- GRAPE: We'll write the necessary software. Move from GRAPE-6 will be less painful than move from GRAPE-4 to GRAPE-6.
- Matrix etc ... RIKEN/NAOJ will do something
- New applications:
  - Primitive Compiler available
  - For high performance, you need to write the kernel code in assembly language

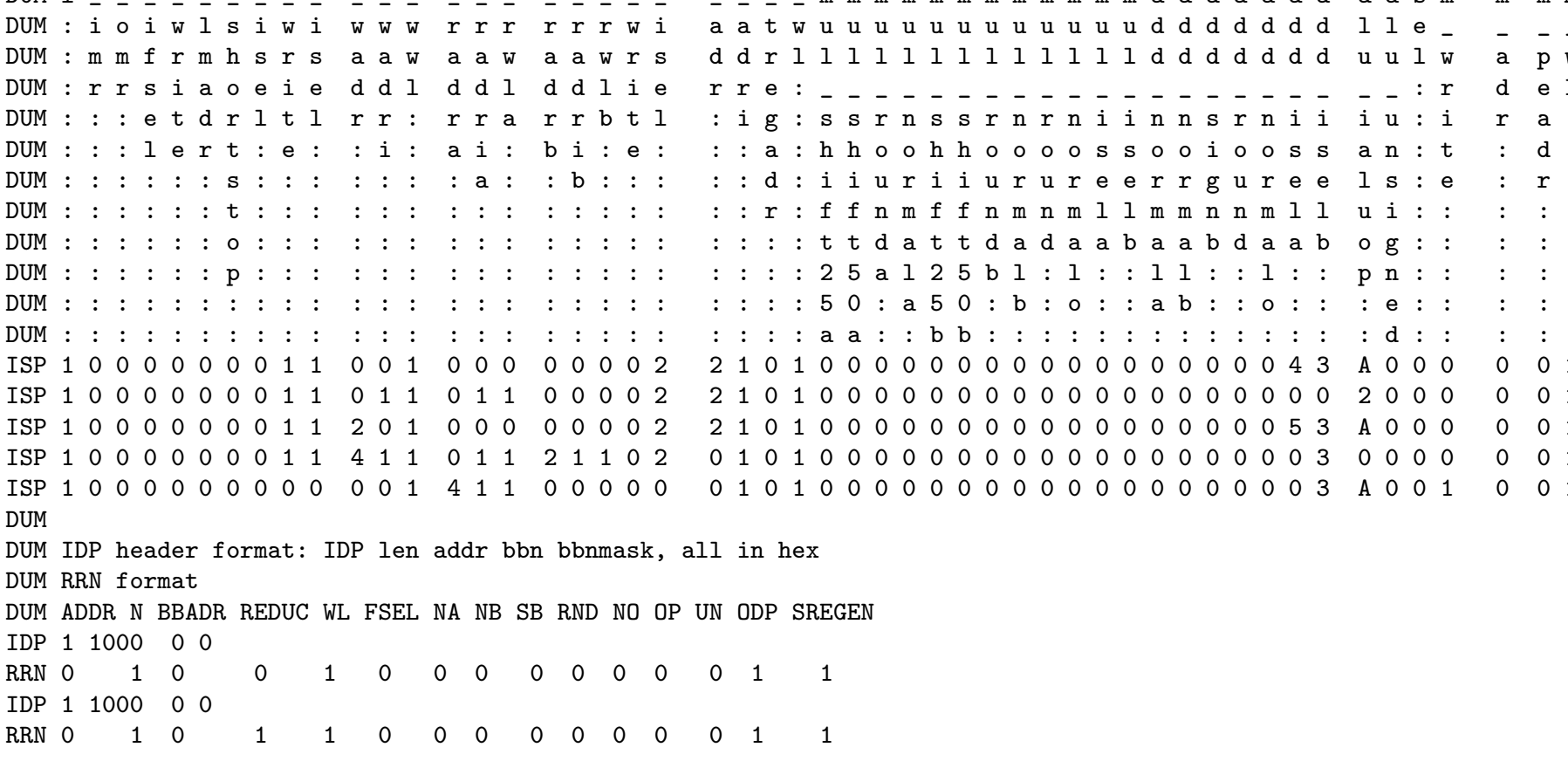
How do you really use it?

Machine language: 108 bits horizontal microcode

```
DUM l m m m t t t t r r  r r r  r r r  r r r l l   l l l l f f f f f f f f f f f f f f f f f f  i i f b   b  b l
DUM l _ _ _ _ _ _ _ _ _  _ _ _  _ _ _  _ _ _ _ _   _ _ _ _ m m m m m m m m m m m m a a a a a a a a  a a s m   m  m r
DUM : i o i w l s i w i  w w w  r r r  r r r w i   a a t w u u u u u u u u u u u u u d d d d d d d d  l l e _   _  _ _
DUM : m m f r m h s r s  a a w  a a w  a a w r s   d d r l l l l l l l l l l l l l d d d d d d d d  u u l w   a  p w
DUM : r r s i a o e i e  d d l  d d l  d d l i e   r r e : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _  _ _ : r   d  e l
DUM : : : e t d r l t l  r r :  r r a  r r b t l   : i g : s s r n s s r n r n i i n n s r n i i  i u : i   r  a
DUM : : : l e r t : e :  : i :  a i :  b i : e :   : : a : h h o o h h o o o o s s o o i o o s s  a n : t   :  d
DUM : : : : : : s : : :  : : :  : a :  : b : : :   : : d : i i u r i i u r u r e e r r g u r e e  l s : e   :  r
DUM : : : : : : t : : :  : : :  : : :  : : : : :   : : r : f f n m f f n m n m l l m m n n m l l  u i : :   :  :
DUM : : : : : : o : : :  : : :  : : :  : : : : :   : : : : t t d a t t d a d a a b a a b d a a b  o g : :   :  :
DUM : : : : : : p : : :  : : :  : : :  : : : : :   : : : : 2 5 a l 2 5 b l : l : : l l : : l : :  p n : :   :  :
DUM : : : : : : : : : :  : : :  : : :  : : : : :   : : : : 5 0 : a 5 0 : b : o : : a b : : o : :  : e : :   :  :
DUM : : : : : : : : : :  : : :  : : :  : : : : :   : : : : a a : : b b : : : : : : : : : : : :  : d : :   :  :
ISP 1 0 0 0 0 0 0 0 1 1  0 0 1  0 0 0  0 0 0 0 2   2 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  4 3  A 0 0 0   0  0 1
ISP 1 0 0 0 0 0 0 0 1 1  0 1 1  0 1 1  0 0 0 0 2   2 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0  2 0 0 0   0  0 1
ISP 1 0 0 0 0 0 0 0 1 1  2 0 1  0 0 0  0 0 0 0 2   2 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  5 3  A 0 0 0   0  0 1
ISP 1 0 0 0 0 0 0 0 1 1  4 1 1  0 1 1  2 1 1 0 2   0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 3  0 0 0 0   0  0 1
ISP 1 0 0 0 0 0 0 0 0 0  0 0 1  4 1 1  0 0 0 0 0   0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 3  A 0 0 1   0  0 1
DUM
DUM IDP header format: IDP len addr bbn bbnmask, all in hex
DUM RRN format
DUM ADDR N BBADR REDUC WL FSEL NA NB SB RND NO OP UN ODP SREGEN
IDP 1 1000  0 0
RRN 0    1  0     0    1   0   0  0   0  0  0  0   0  1    1
IDP 1 1000  0 0
RRN 0    1  0     1    1   0   0  0   0  0  0  0   0  1    1
```

How do you really use it?

Machine language: 108 bits horizontal microcode



```
DUM l m m m t t t t r r  r r r  r r r  r r r l l  l l l l f f f f f f f f f f f f f f f f  i i f b  b  b
DUM l _ _ _ _ _ _ _ _ _  _ _ _  _ _ _  _ _ _ _ _  _ _ _ _ m m m m m m m m m m m m a a a a a a a a  a a s m  m  m
DUM : i o i w l s i w i  w w w  r r r  r r r w i  a a t w u u u u u u u u u u u u u d d d d d d d d  l l e _  _  _
DUM : m m f r m h s r s  a a w  a a w  a a w r s  d d r l l l l l l l l l l l l l l d d d d d d d d  u u l w  a  p
DUM : r r s i a o e i e  d d l  d d l  d d l i e  r r e : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _  _ _ : r  d  e
DUM : : : e t d r l t l  r r :  r r a  r r b t l  : i g : s s r n s s r n r n i i n n s r n i i  i u : i  r  a
DUM : : : l e r t : e :  : i :  a i :  b i : e :  : : a : h h o o h h o o o o o s s o o i o o s s  a n : t  :  d
DUM : : : : : : s : : :  : : :  : a :  : b : : :  : : d : i i u r i i u r u r e e r r g u r e e  l s : e  :  r
DUM : : : : : : t : : :  : : :  : : :  : : : : :  : : r : f f n m f f n m n m l l m m n n m l l  u i : :  :  :
DUM : : : : : : o : : :  : : :  : : :  : : : : :  : : : : t t d a t t d a d a a b a a b d a a b  o g : :  :  :
DUM : : : : : : p : : :  : : :  : : :  : : : : :  : : : : 2 5 a l 2 5 b l : l : : l l : : l : :  p n : :  :  :
DUM : : : : : : : : : :  : : :  : : :  : : : : :  : : : : 5 0 : a 5 0 : b : o : : a b : : o : :  : e : :  :  :
DUM : : : : : : : : : :  : : :  : : :  : : : : :  : : : : a a : : b b : : : : : : : : : : : :  : d : :  :  :
ISP 1 0 0 0 0 0 0 0 1 1  0 0 1  0 0 0  0 0 0 0 2  2 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  4 3  A 0 0 0  0  0
ISP 1 0 0 0 0 0 0 0 1 1  0 1 1  0 1 1  0 0 0 0 2  2 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0  2 0 0 0  0  0
ISP 1 0 0 0 0 0 0 0 1 1  2 0 1  0 0 0  0 0 0 0 2  2 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  5 3  A 0 0 0  0  0
ISP 1 0 0 0 0 0 0 0 1 1  4 1 1  0 1 1  2 1 1 0 2  0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 3  0 0 0 0  0  0
ISP 1 0 0 0 0 0 0 0 0 0  0 0 1  4 1 1  0 0 0 0 0  0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 3  A 0 0 1  0  0
DUM
DUM IDP header format: IDP len addr bbn bbnmask, all in hex
DUM RRN format
DUM ADDR N BBADR REDUC WL FSEL NA NB SB RND NO OP UN ODP SREGEN
IDP 1 1000  0 0
RRN 0    1  0     0    1   0   0  0   0  0  0  0   0  1    1
IDP 1 1000  0 0
RRN 0    1  0     1    1   0   0  0   0  0  0  0   0  1    1
```

Probably not the way you want to program.

What do we need to specify?

Consider a GRAPE-like model:

We calculate $\sum_j f(x_i, x_j)$, for many "i"s and many "j"s

1. **First send all "j"s**
2. **Repeat the following**
3. **send "j"s to the local memory of PEs**
4. **Let the chip calculate the interaction $f(x_i, x_j)$ and accumurate**
5. **retrieve the result**

This model, with some extention, seems to be able to express most of the things we want to do on GRAPE-DR.

Interface definitions

We need ways to specify

- actual PE operations to calculate f
- procedure and data type for
  - "j"s
  - "i"s
  - results

原始的なコンパイラ

(中里 2006)

```
/VARI  xi, yi, zi, e2;
/VARJ  xj, yj, zj, mj;
/VARF  fx, fy, fz;
dx = xi - xj;
dy = yi - yj;
dz = zi - zj;
r2 = dx*dx + dy*dy + dz*dz + e2;
r3i= powm32(r2);
ff = mj*r3i;
fx += ff*dx;
fy += ff*dy;
fz += ff*dz;
```

これから GRAPE 並のことをするアセンブラ、インターフェースライブラリを生成。
基本的なアイデアは PGR (FPGA を使ったPROGRAPE用コンパイラ、濱田D論2006) と同様

GRAPE-DR の次

- 「京速計算機」あるいは「次世代スーパーコンピュータプロジェクト」
  - 2006 年度から 5ヵ年計画
  - 10 Pflops
  - 総予算 1100 億？
- GRAPE-DR のようなものが一部入るかもしれない。まだ開発実施本部の方針が定まらない。
- もしも入るとしたら総予算の 10% 以下程度。

「V-GRAPE 構想」

GRAPE-DR、 ClearSpeed との違い

- かなり大きなオンチップ DRAM をつける
- ある程度いろんなことを 高速にできそう
  - 陽解法流体計算
  - 構造解析
- PC クラスタ+加速ボード構成

“Versatile-GRAPE”

大雑把な性能推定

ピーク 20Pflops として

- 重力多体、分子動力学、第一原理計算
  効率 30–50% (6–10 Pflops)
- 陽解法流体、有限要素法
  効率 10–20% (2–4 Pflops)

他のアプローチとの比較

筑波大学の将来プラン
48Gflops、 15GB/s、400,000 チップで 20PF

ピーク性能を同じにして比べると

- **チップ数** 1/20
- bisection bandwidth 1/2
- **ノード間接続総本数** 1/5
- **総メモリ** 1/3000

トータルコスト、信頼性、消費電力にかなりの差？

メモリ沢山いる計算: 遅いメモリをうまく使う必要あり