GRAPE-DR とその先、その他の話題

牧野淳一郎

今日の話の構成

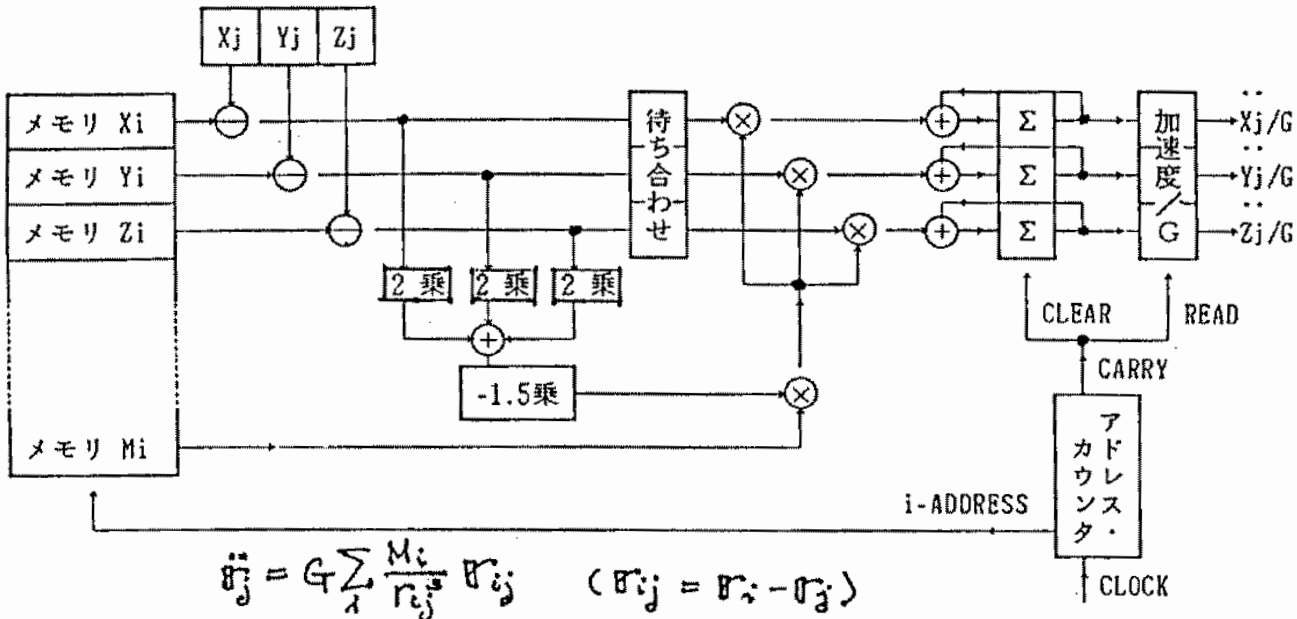
- これまでの GRAPE
- GRAPE-DR の考え方
- 現在の SIMD 超並列チップに共通の問題点
- 解決策
- 性能予測: QCD 計算、多次元 FFT を例に
- まとめ

GRAPE の考え方

- 重力多体問題 (強結合プラズマ、分子動力学): 粒子間相互作用の計算が計算量のほとんど全部
- 効率のよい計算法 (Barnes-Hut tree, FMM, Particle-Mesh Ewald(PPPM) ...) 粒子間相互作用の計算を速くするだけでかなり加速できる
- そこだけ速くする電子回路を作る (「計算機」というようなものではない)

近田提案

1988年、天文・天体物理夏の学校



$$\ddot{\mathbf{r}}_j = G\sum_i \frac{M_i}{r_{ij}^3}\mathbf{r}_{ij} \qquad (\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j)$$

+, −, ×, 2乗は1 operation, -1.5乗は多項式近似でやるとして10operation 位に相当する.
総計24operation.
各operation の後にはレジスタがあって, 全体がpipelineになっているものとする.
「待ち合わせ」は2乗してMと掛け算する間の時間ズレを補正するためのFIFO(First-In First-Out memory).
「Σ」は足し込み用のレジスタ. N回足した後結果を右のレジスタに転送する.

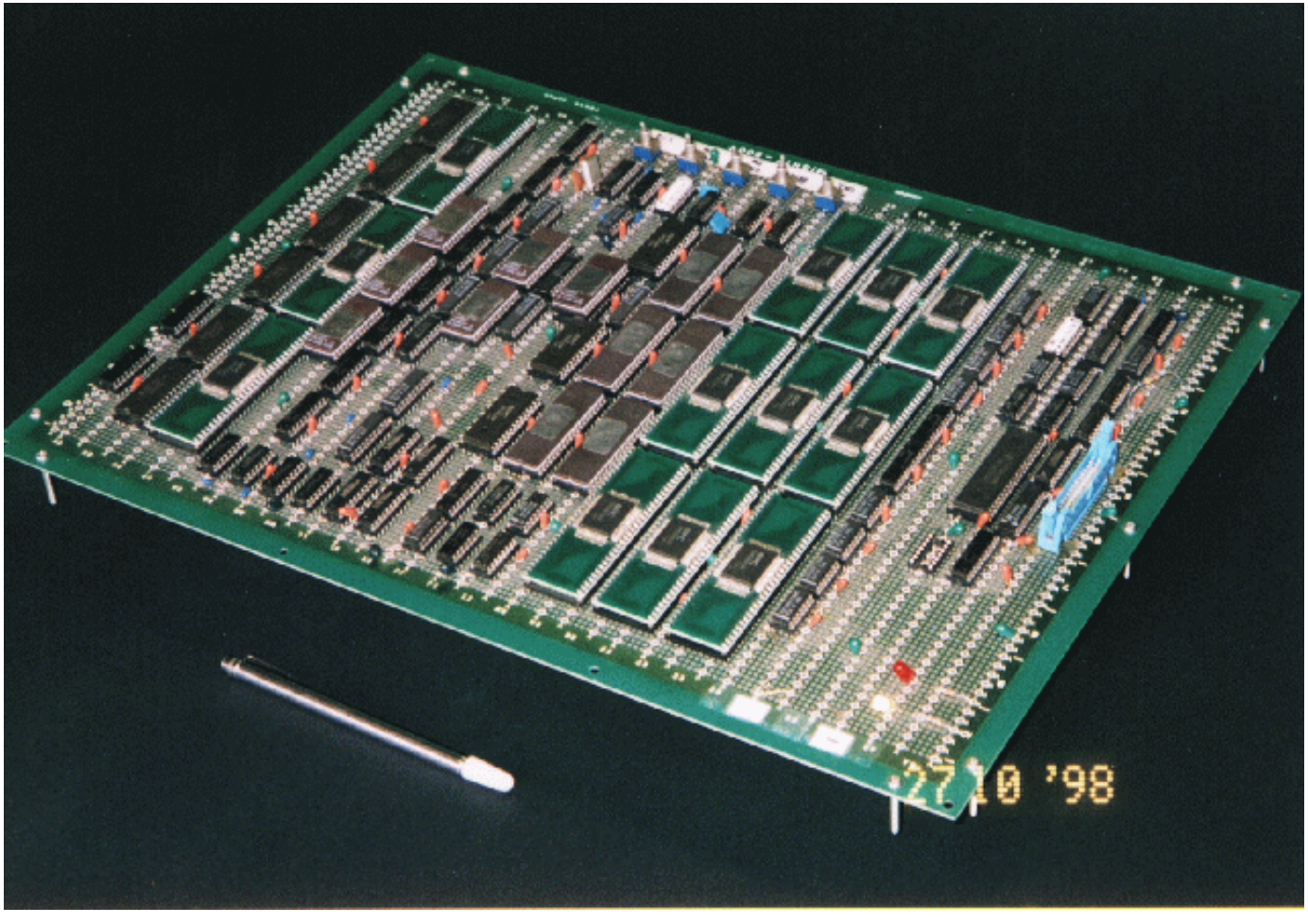
図2. N体問題のj-体に働く重力加速度を計算する回路の概念図.

近田さんによる見積もり

- 32 ビット固定小数点
- IC 200 個
- 体積 $0.1\mathrm{m}^3$
- コスト 400万

「但し、近田電子製作所の見積もりは甘いという声もあることを付け加えます」

GRAPE-1(1989)

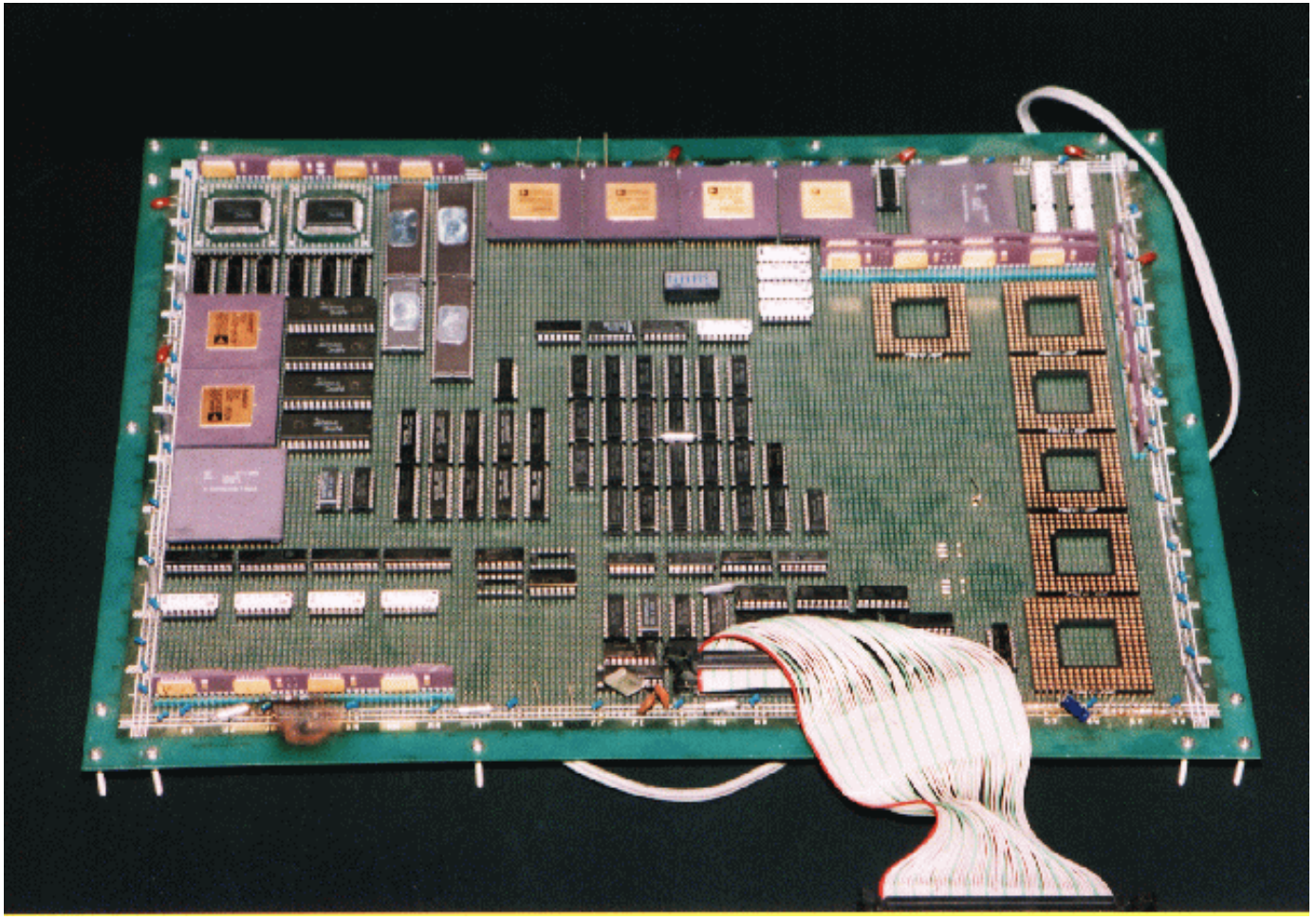


GRAPE-1 の中身

- 演算毎に語長指定。固定 16–対数 8–固定 32–固定 48
- IC 100 個
- コスト 20 万
- 240Mflops 相当
- 伊藤、牧野、戎崎

朴さんには大変お世話になりました。

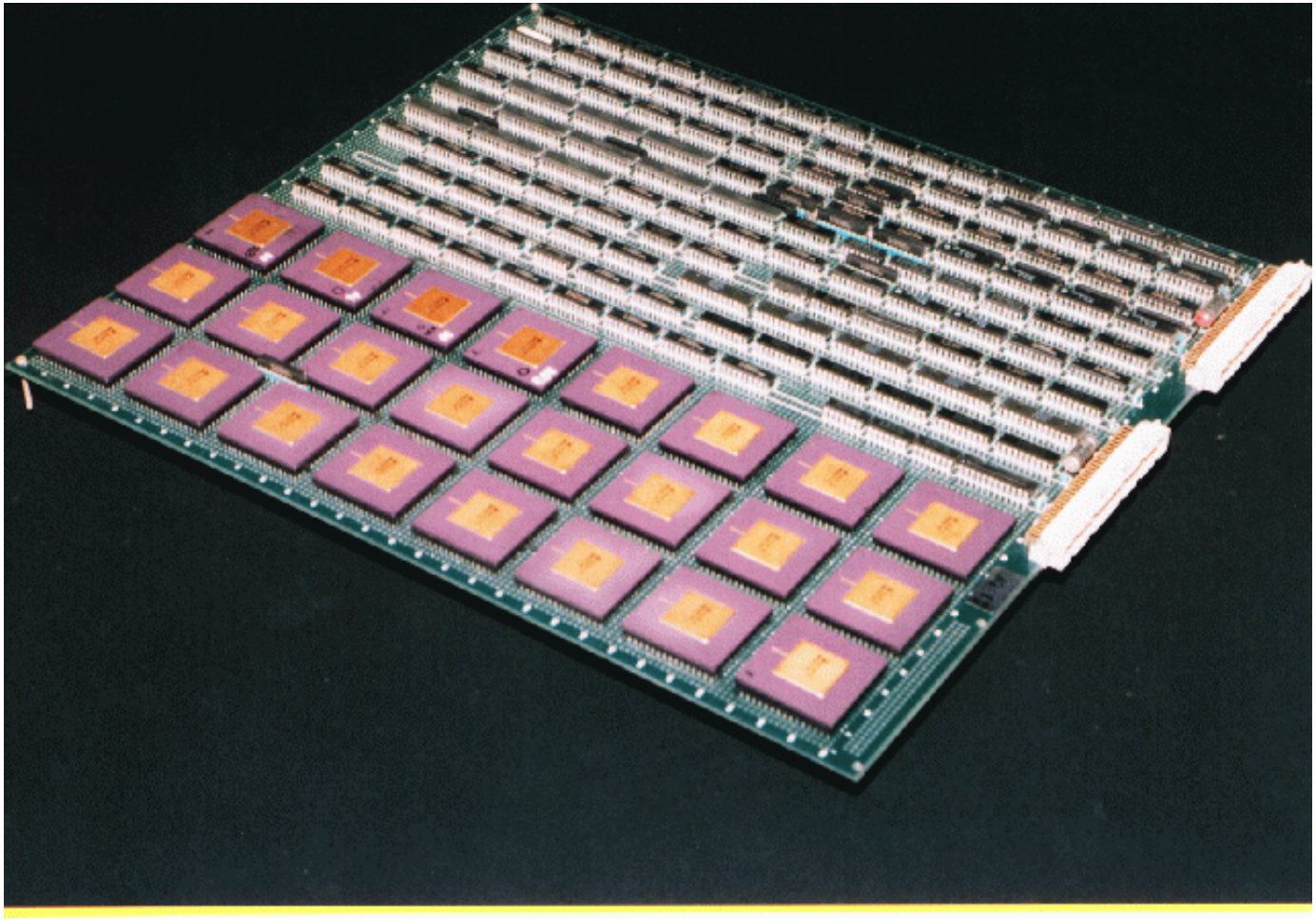
GRAPE-2(1990)



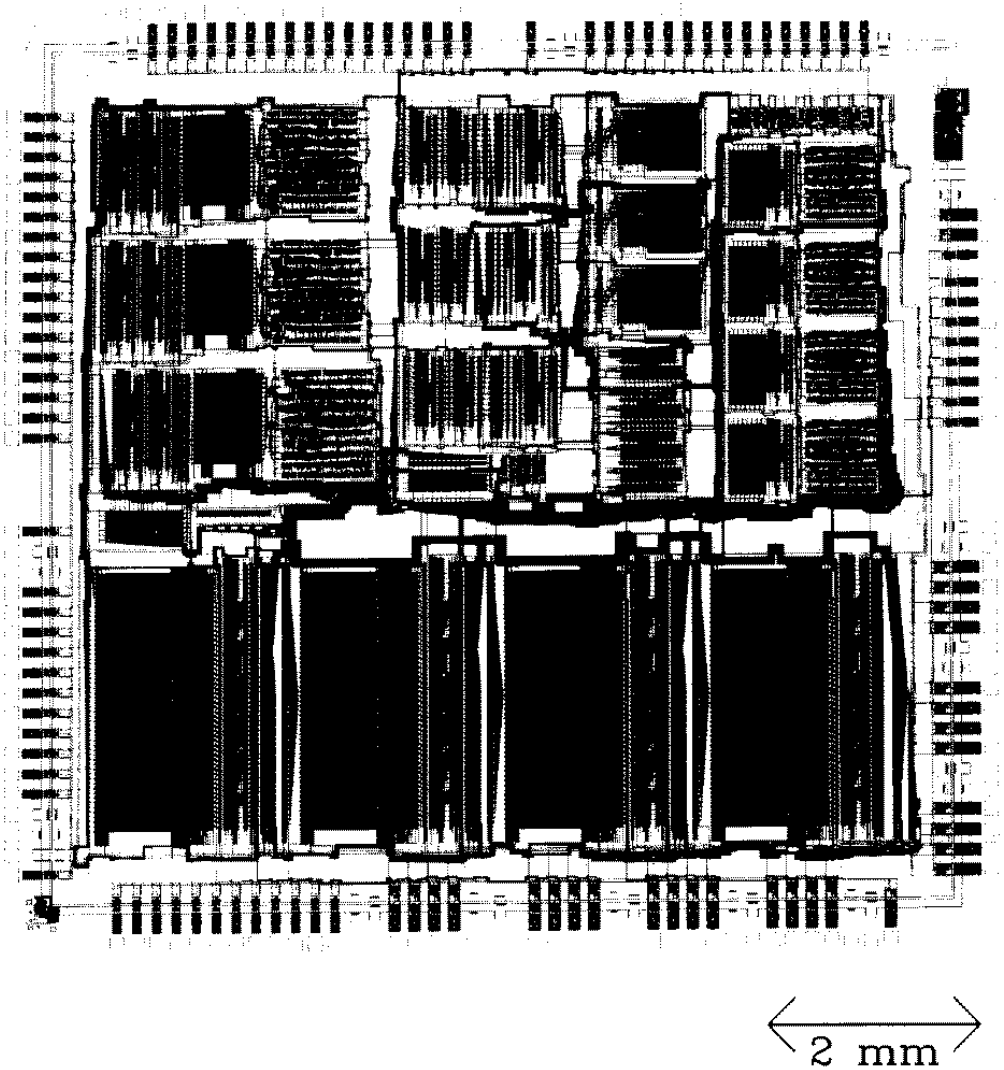
GRAPE-2 の中身

- 8 ビット演算とかは止めて普通に浮動小数点演算 (倍精度は最初と最後だけ)
- 40Mflops
- 戎崎、伊藤、牧野

GRAPE-3(1991)



GRAPE-3 チップ

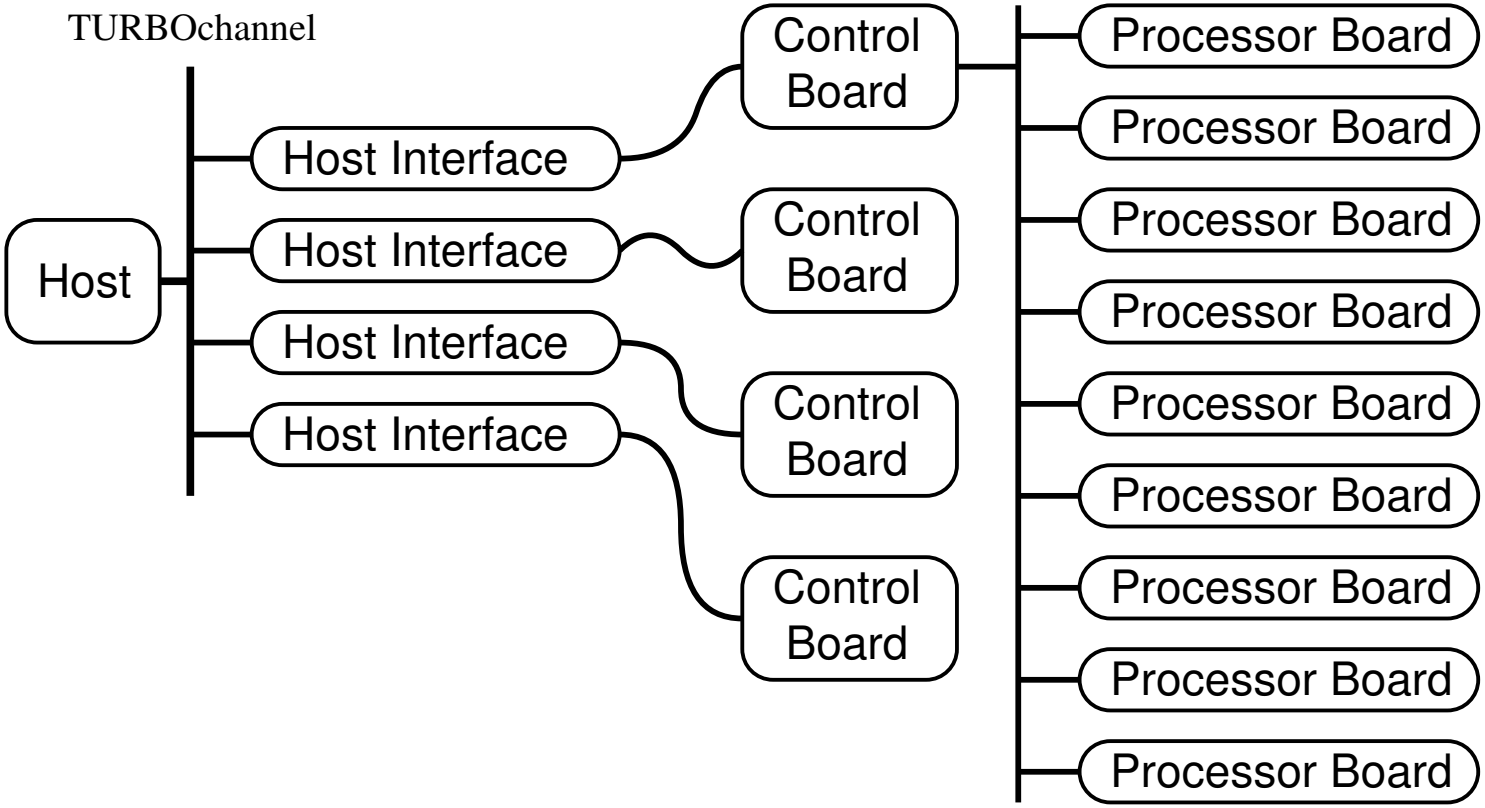


GRAPE-3

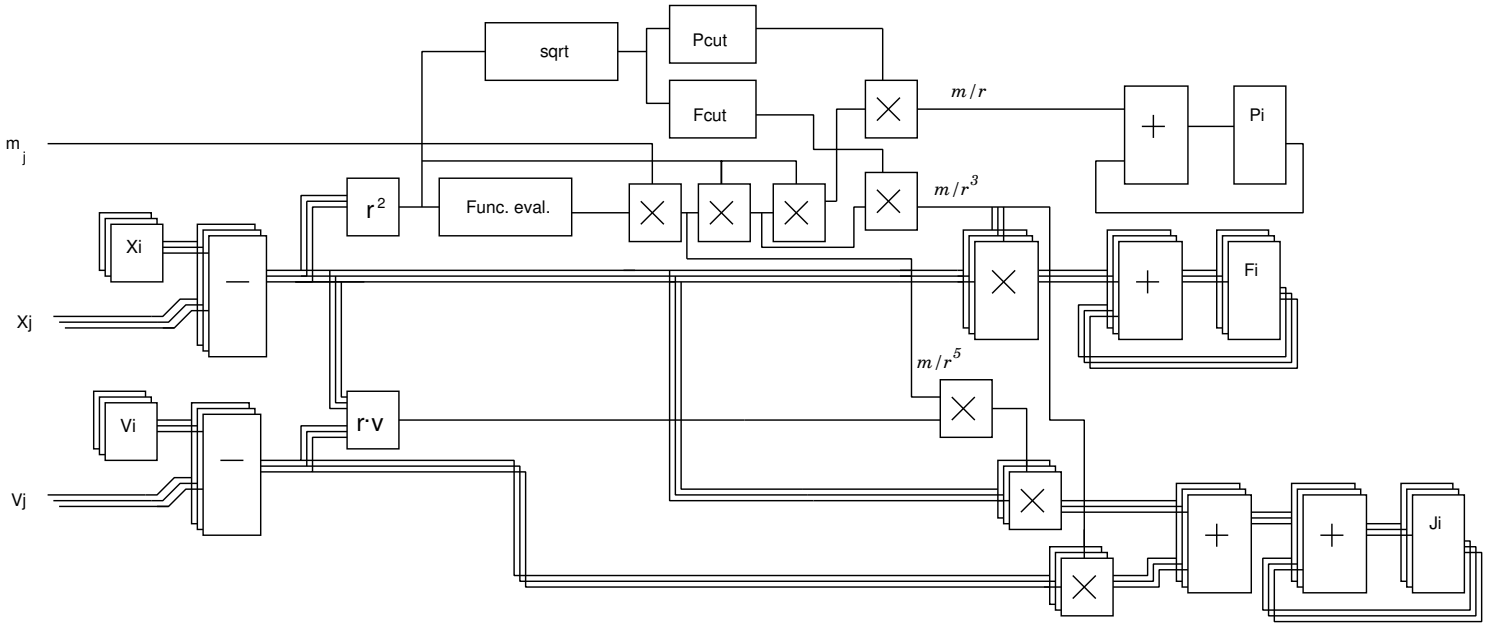
- 仕様決定、シミュレータ (C で記述) は牧野
- 論理設計以降は富士ゼロックス (橋本、富田)
- SCS Genesil で設計
- ファブは NS. 1μm
- ボードは奥村 (現在国立天文台)

GRAPE-4(1995)

GRAPE-4 の構成

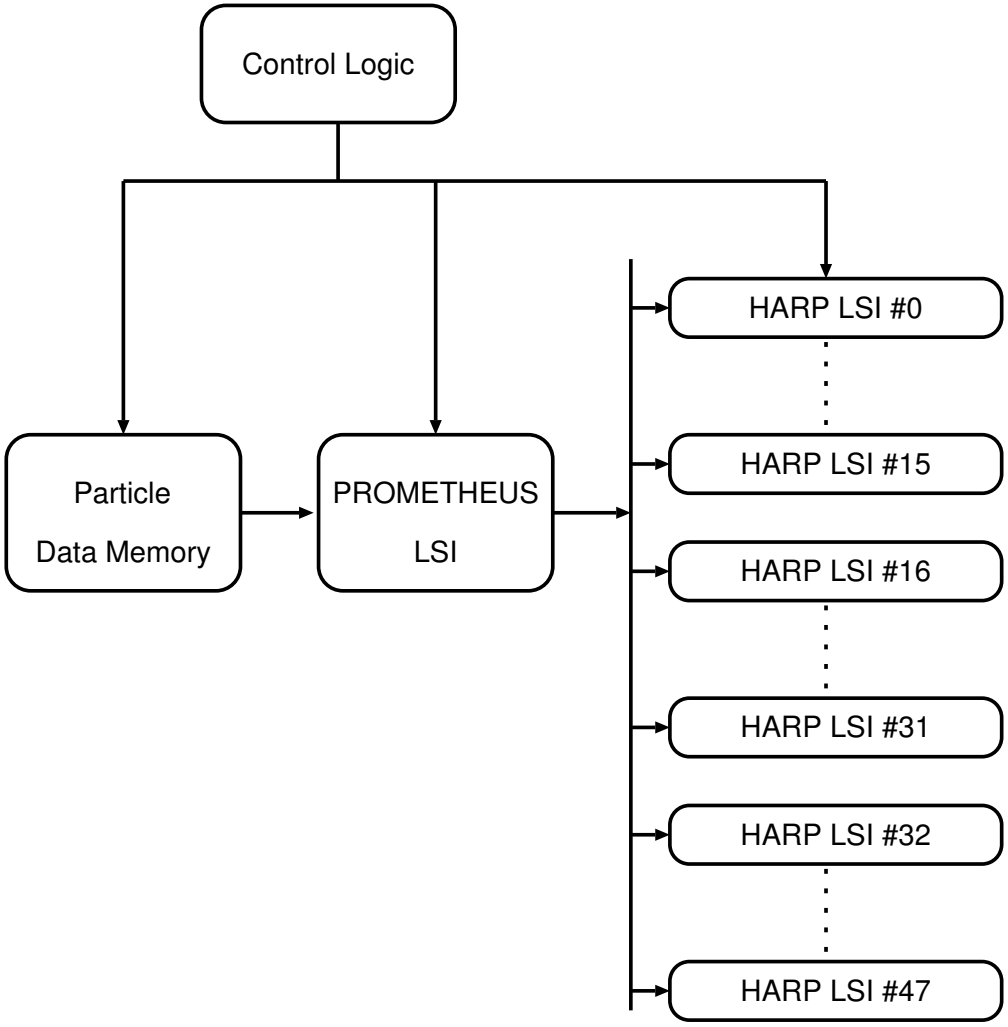

TURBOchannel
Host
Host Interface
Host Interface
Host Interface
Host Interface
Control Board
Control Board
Control Board
Control Board
Processor Board
Processor Board
Processor Board
Processor Board
Processor Board
Processor Board
Processor Board
Processor Board
Processor Board

GRAPE-4 パイプライン



設計:泰地

GRAPE-4 演算ボード



多数の パイプライン LSI がメモリユニットを共有

↓

ボードが単純になり、集積度をあげられる

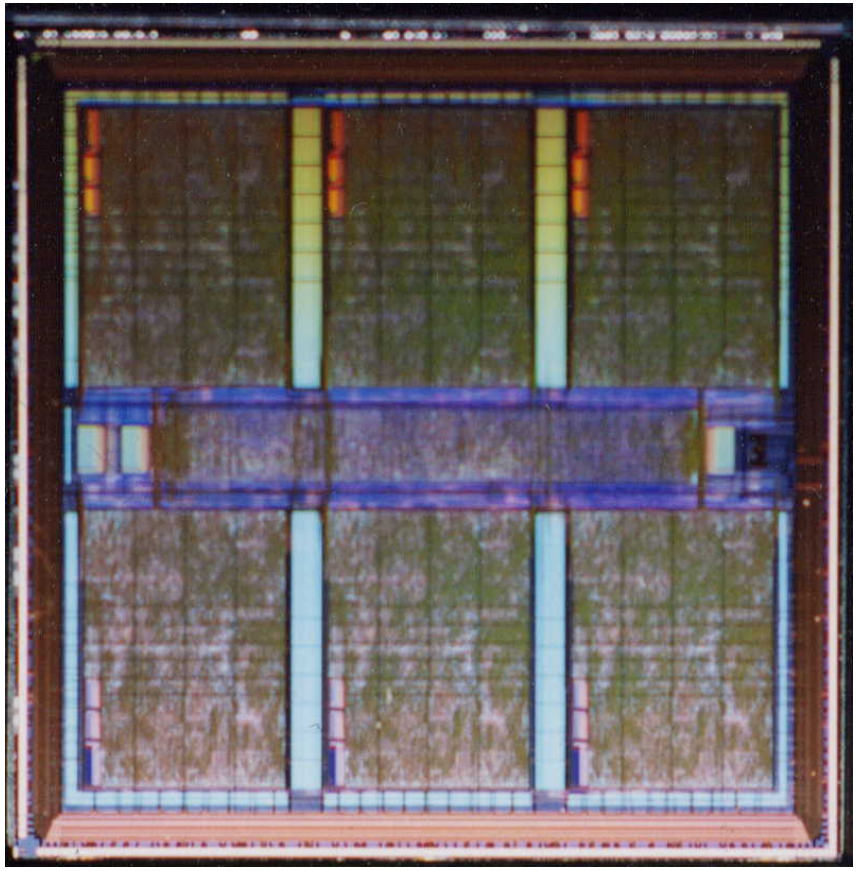
GRAPE-4 概要

- 48 個プロセッサチップがのったボード 36 枚
- 1 チップ 20 演算、32 MHz 動作で 640 Mflops
- LSI logic 1μm スタンダードセル
- プロセッサチップは泰地、「予測子」チップ牧野

GRAPE-6(2001)

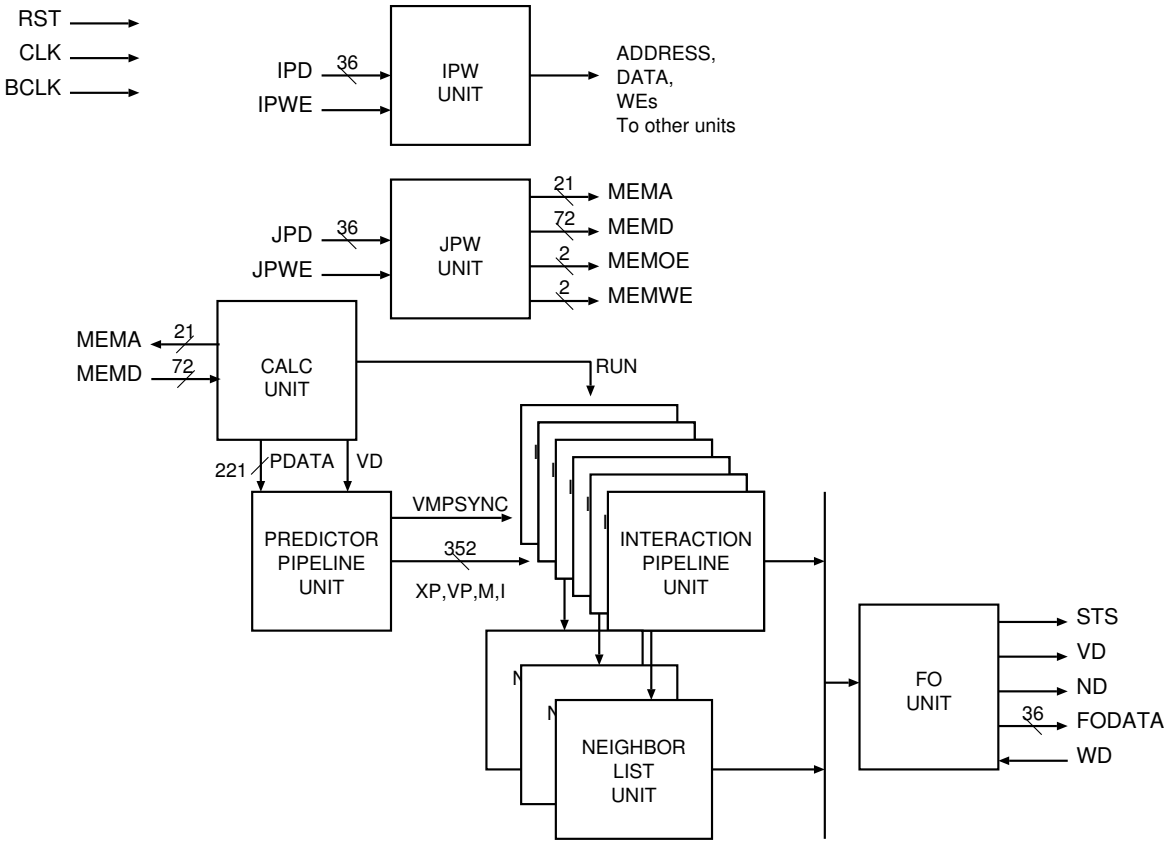
- プロセッサチップ
- プロセッサモジュール
- プロセッサボード
- ネットワークボード
- 全体

パイプライン LSI



- 0.25 μm ルール (東芝 TC-240, 1.8M ゲート)
- 90 MHz 動作
- 6 パイプラインを集積
- チップあたり 31 Gflops

パイプライン LSI詳細

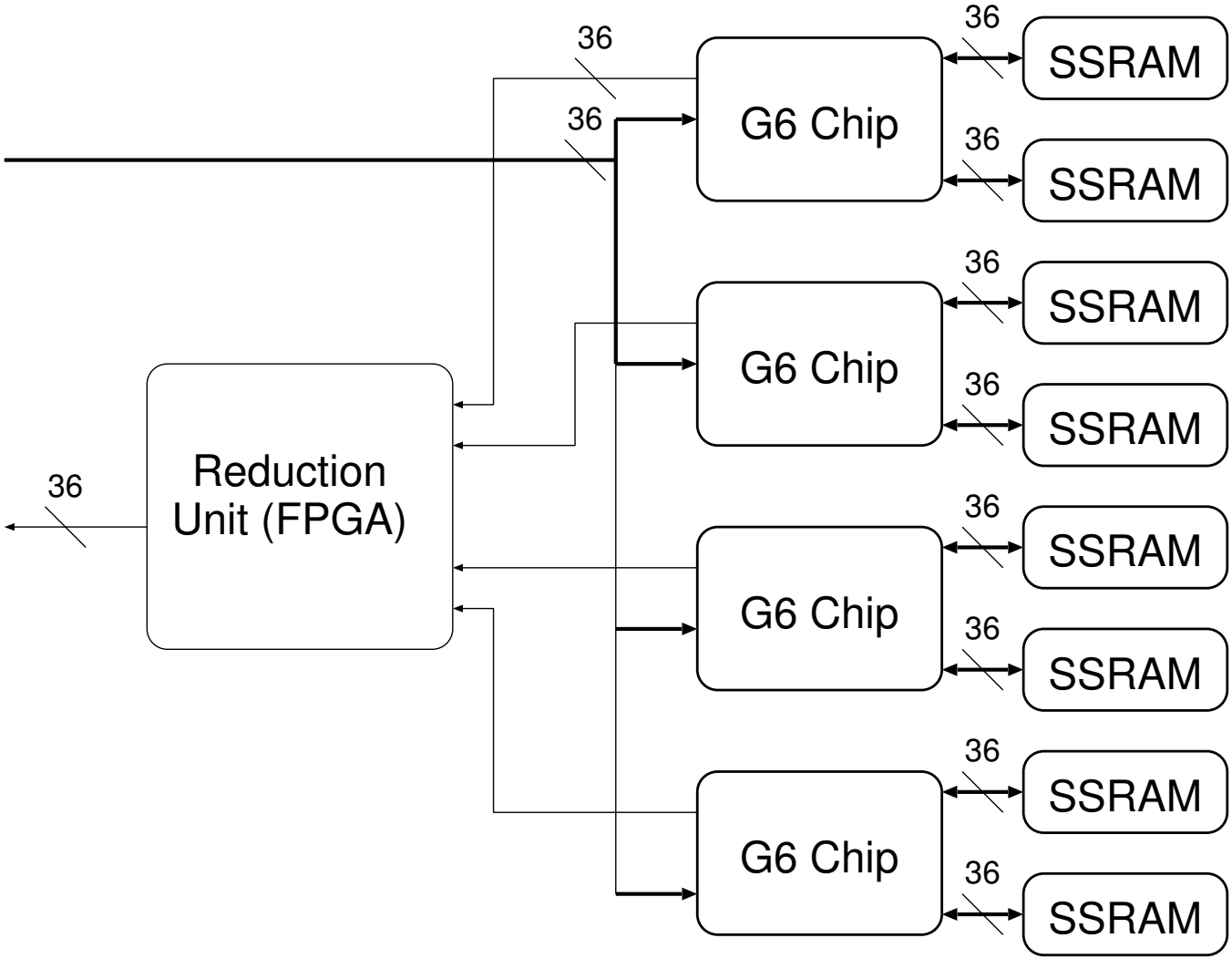


GRAPE-4 プロセッサボードの全機能を集積

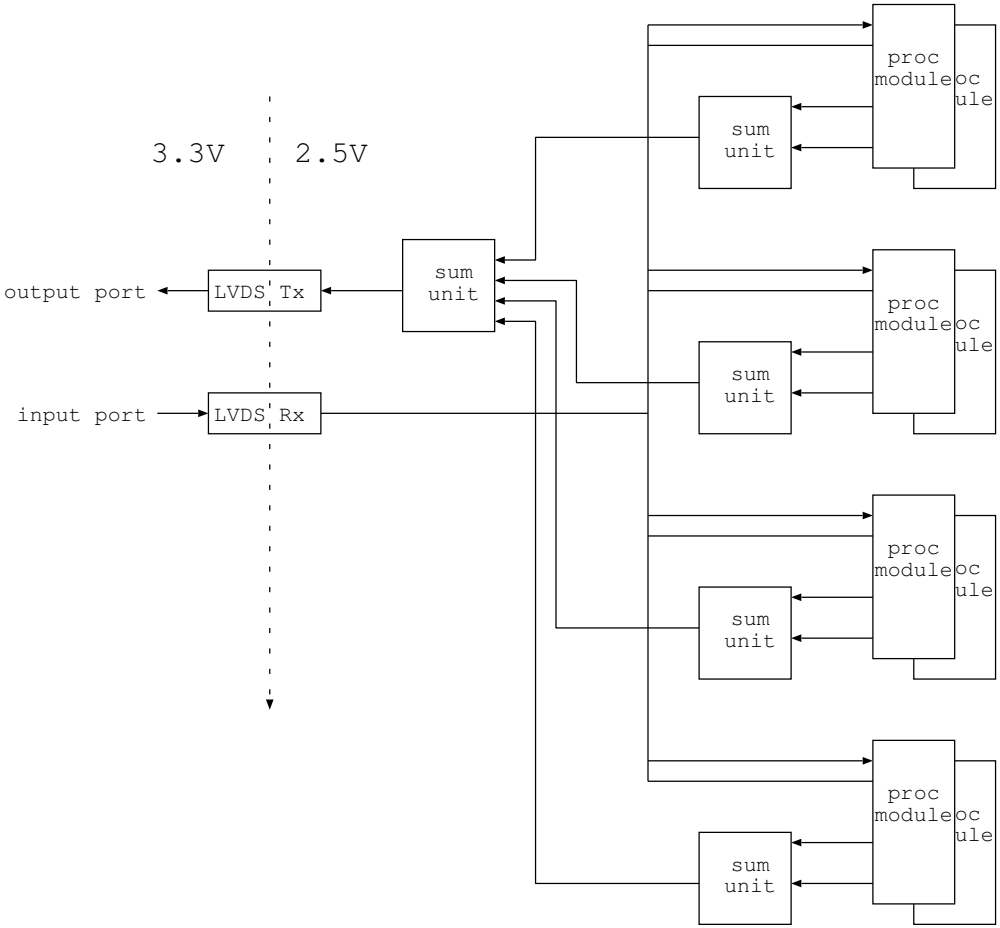
- ホスト IF
- メモリ IF
- パイプライン本体
- 制御回路

GRAPE-6 processor module

GRAPE-6 processor module

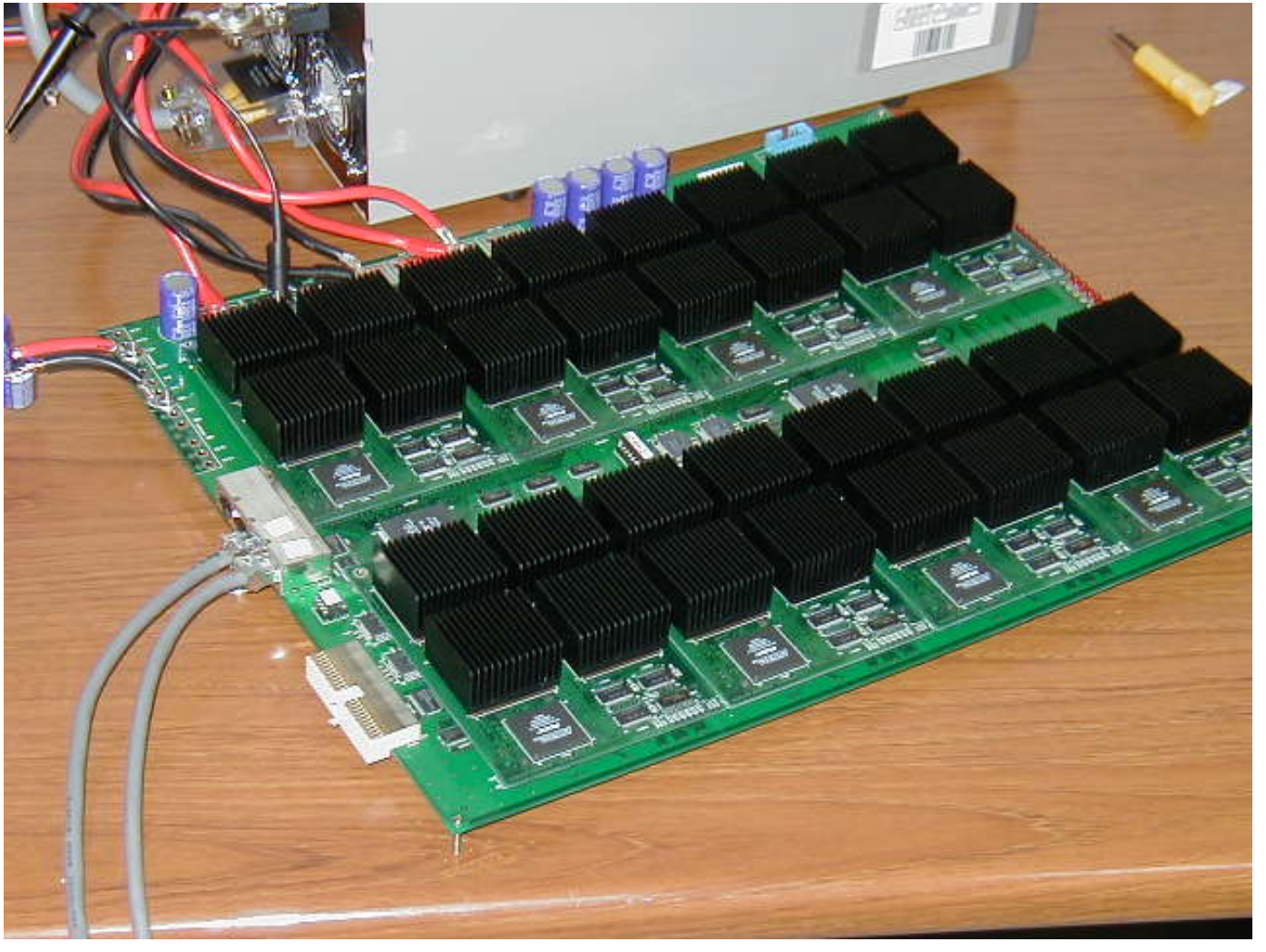


GRAPE-6 Processor board

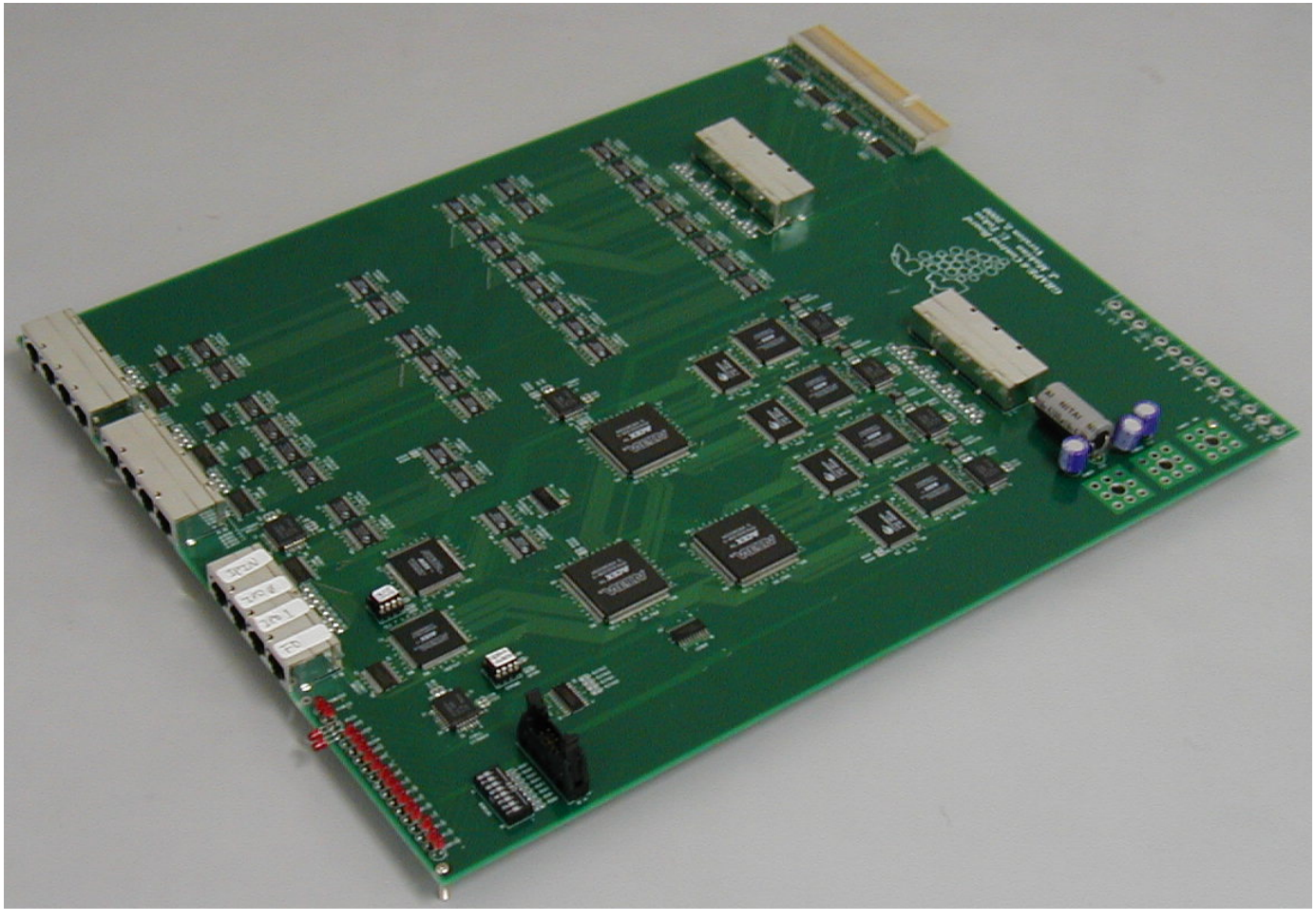


- ボード 1 枚に 32 チップ
- セミシリアル (LVDS) インターフェース (350MHz clock, 4 wires)

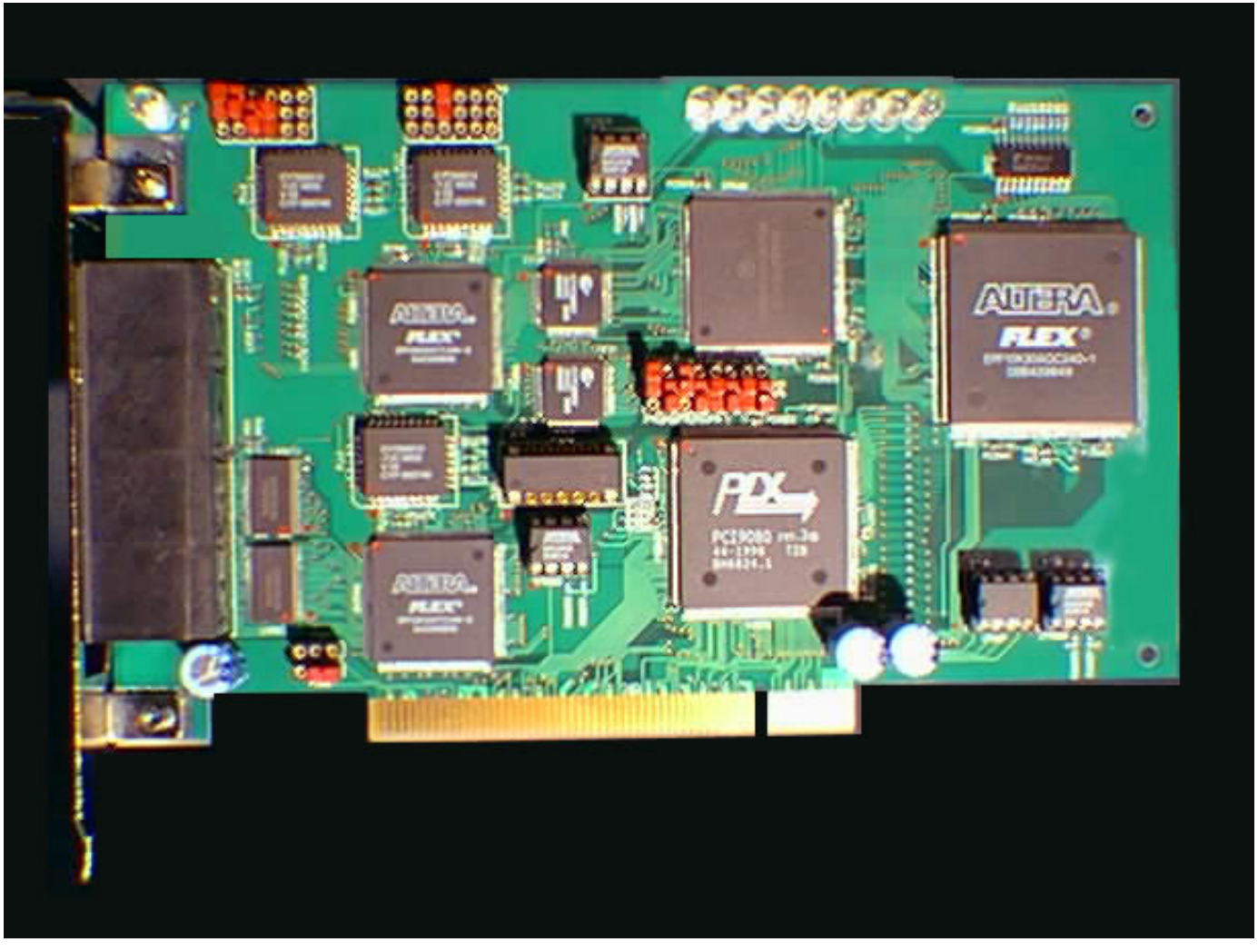
GRAPE-6 processor board



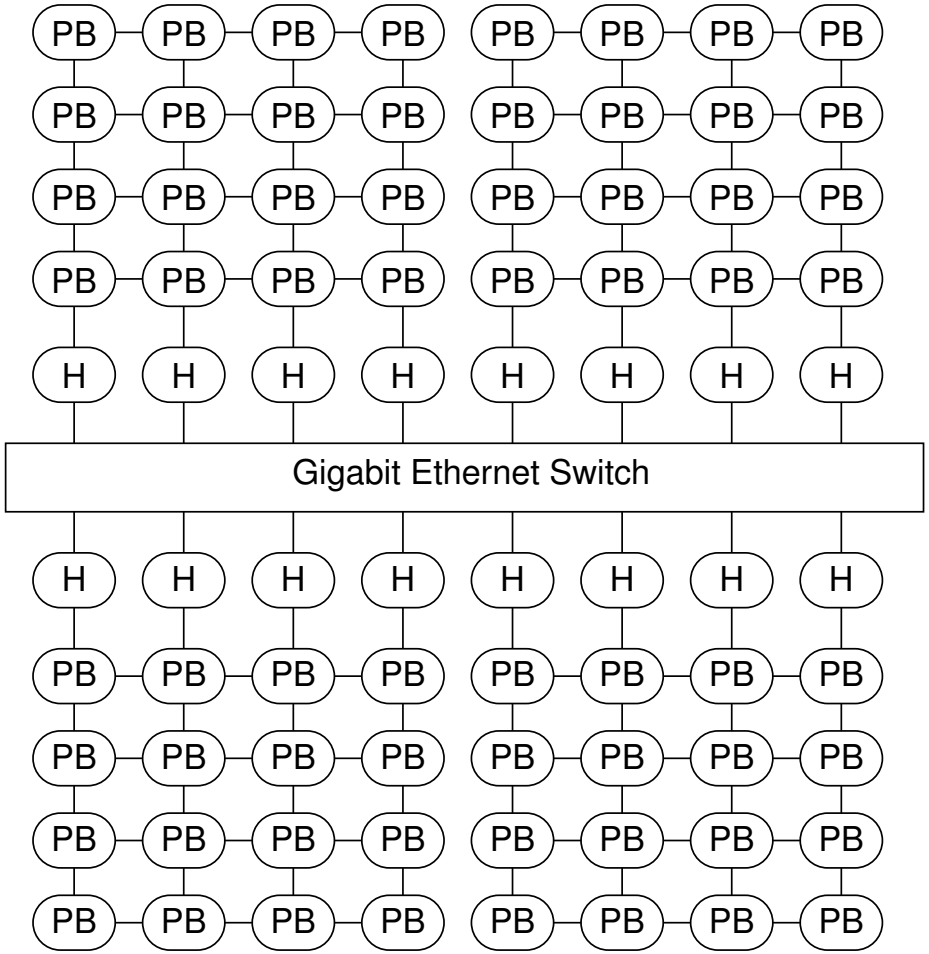
GRAPE-6 network board



GRAPE-6 host interface board



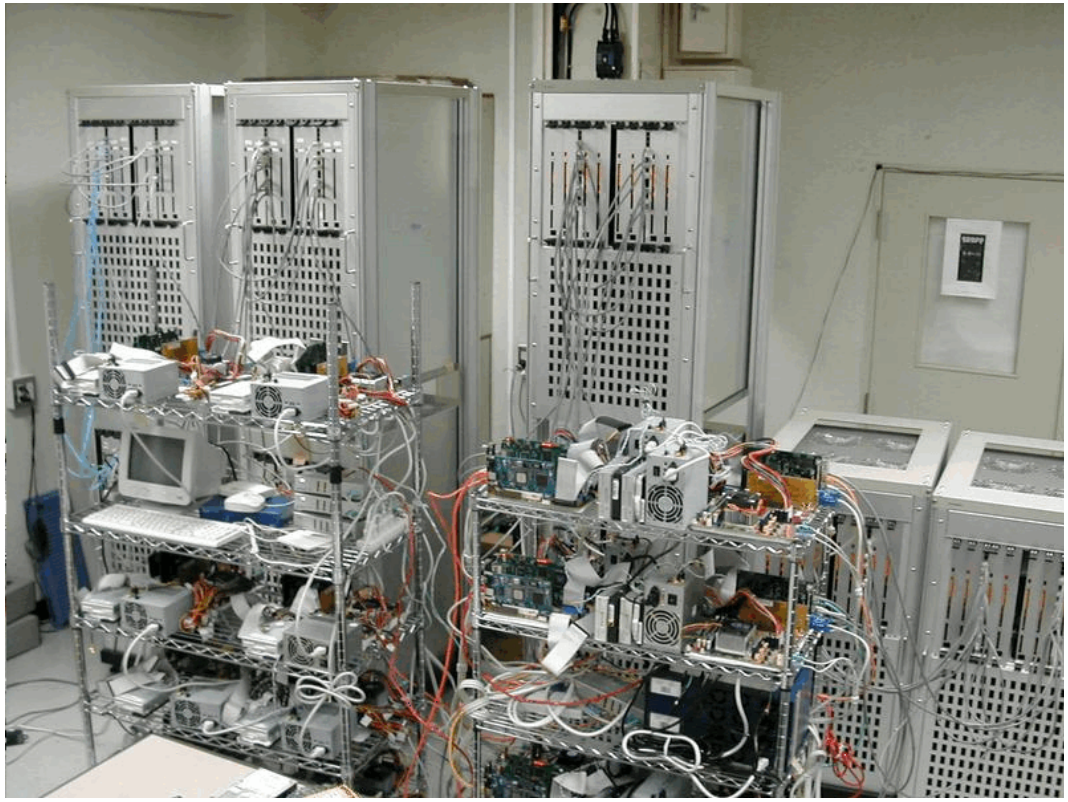
The full 64 Tflops GRAPE-6 system



- ホスト 4 台、プロセッサボード 16 枚のブロック
- 4 ブロックで全体システムを構成。結合は GbE

専用ネットワークと汎用ネットワークの組み合わせ。

The 64-Tflops GRAPE-6 system



2002年現在の 64 Tflops システム

4 ブロック
16 ホスト
64 プロセッサボード

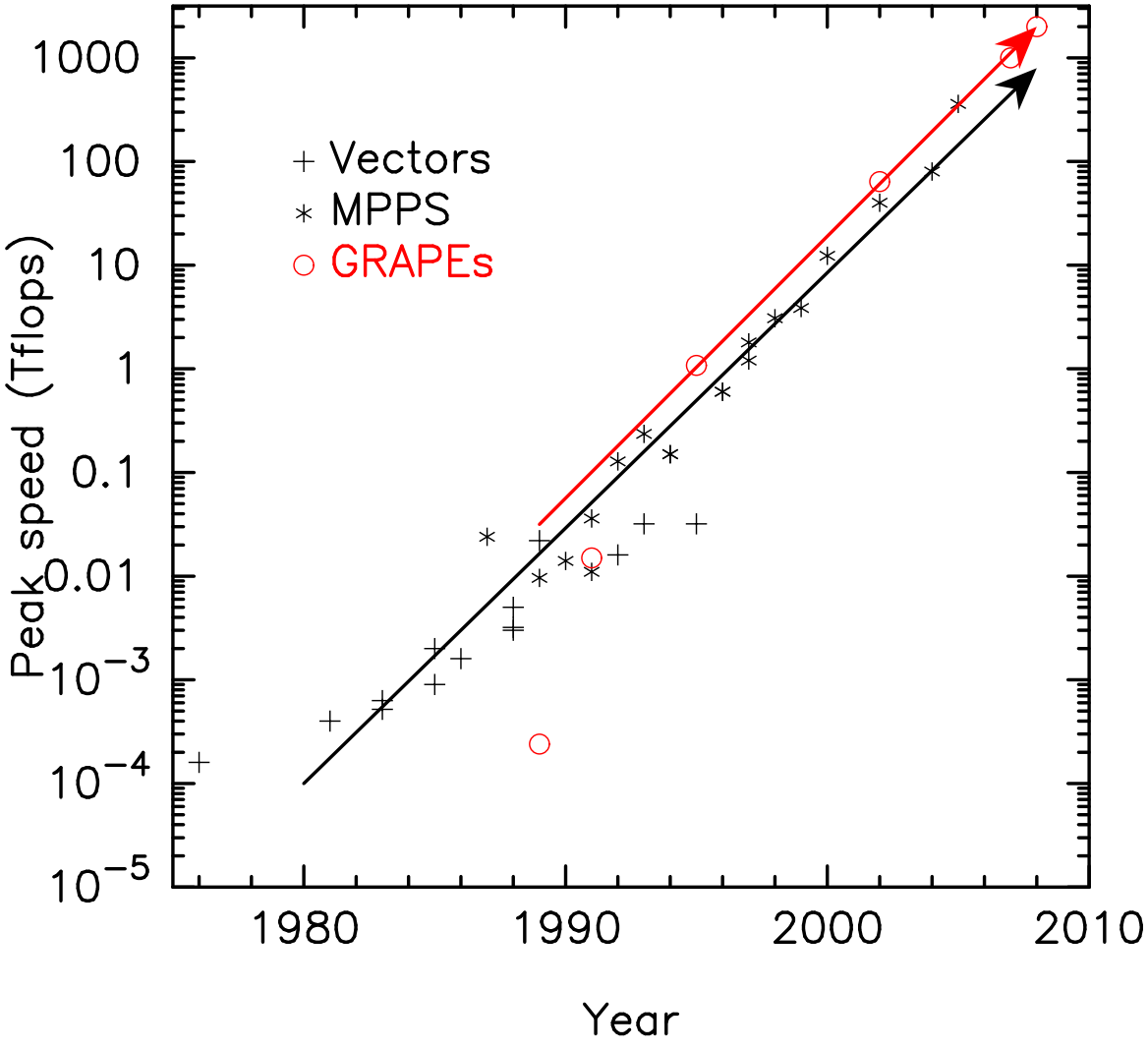
他の GRAPE 型機械

- 1992 GRAPE-2A : 最初の MD 用 GRAPE
  設計:伊藤・福重
- 1996 MD-GRAPE: 最初の MD 用 GRAPE チップ
  チップ設計:泰地
- 2001 MDM (MDG2): 理研・戎崎グループによる。 75T
- 2006 PE (MDG3): 理研・泰地グループによる。 1P
- 1992- MD-Engine: 富士ゼロックス、大正製薬 (現在 NEC)

他の GRAPE 関係専用計算機

- 1991 DREAM: 大規模偏微分方程式計算「専用」計算機。
  ハードディスクを主記憶に、という発想。
  ディスク 1 台の試作程度まで。大野、牧野、戎崎。
- 1993 ZEBRA : Radiocity 法専用計算機。成見、戎崎
- 1995 General: LU 分解専用計算機。市販チップで構成。
  清木、戎崎、泰地、牧野。
- 2002? MACE: LU 分解専用計算機。泰地、戎崎

ピーク性能の進歩



GRAPE-4 以降、完成した時点で世界最高速を実現

商業版 GRAPE

- GRAPE-3 から商業版
- GRAPE-6 を購入した機関

American Museum of Natural History
Drexel University
Indiana University Rochester Institute of Technology
Rutgers University
Rochester Institute of Technology
University of Michigan
University of California
McMaster University
The United Kingdom The University of Cambridge
University of Edinburgh
Observatoire Astronomique Marseille-Provence(OAMP)
Astronomisches Rechen-Institute (ARI)
Ludwing-Maximillans University
Max-Planck-Institute fur Astronomic

GRAPE-6 を購入した機関(つづき)

University of Bonn
University of Mannheim
Holland University of Amsterdam
Nanjing Univesity
Citec Co., Ltd
Gunma Astronomical Observatory
Hokkai-Gakuen University
Kansai University
Kyoto University
National Institute for Fusion Science (NIFS)
National Astronomical Observatory of Japan
Osaka University
The University of Tokyo
Tokyo Institute of Technology
University of Tsukuba

約30機関、 60Tflops。MDGRAPE-2 も同様

GRAPE-6 の次は？

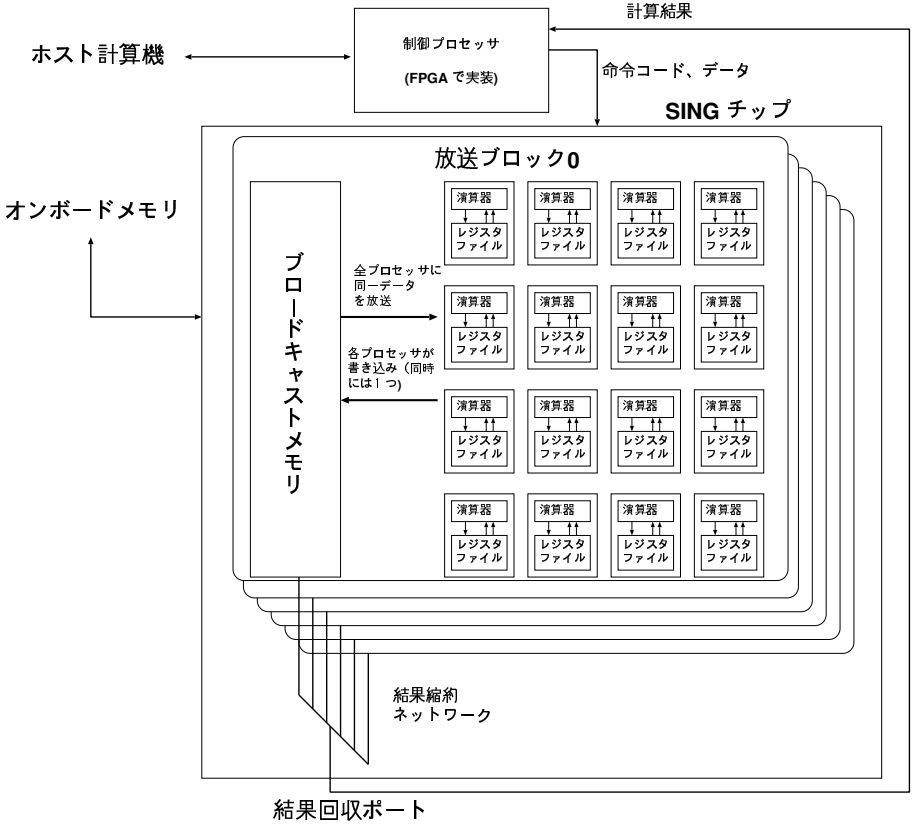
MDGRAPE-3 の次: MDGRAPE-4, 20Pflops@2010

そもそも MDGRAPE-3 にあたるものは? → GRAPE-DR

GRAPE-DR のアプローチ

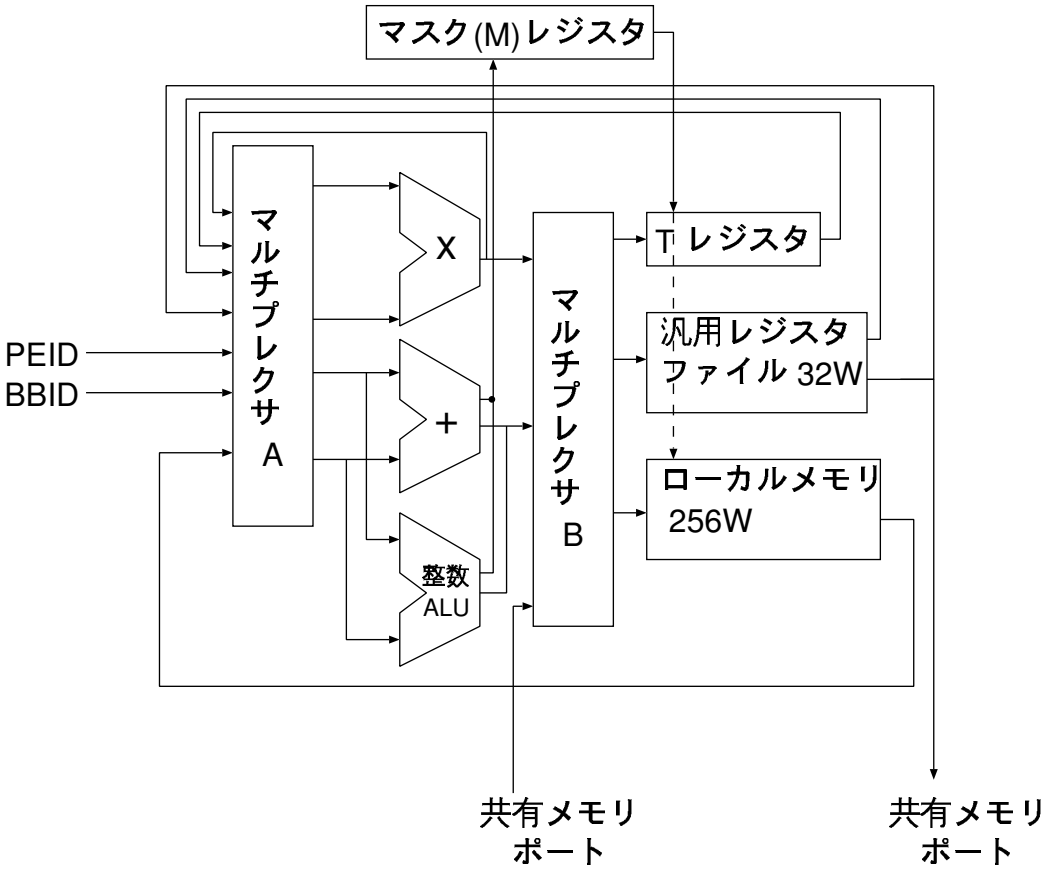
Greatly Reduced Array of Processor Element with Data Reduction

- 沢山の単純なプロセッサを 1 チップに集積
- メモリバンド幅はコストに効かない程度に抑える
- プロセッサを 2 階層でグループ化
- グループにまたがった縮約のためのネットワーク



(従来の GRAPE に比べて) 色々なアプリケーションに使える

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。

アプリケーションに対する考え方

- Memory Wall が問題にならないような
  アプリケーションのみを対象にする
- 3 つの型に特化
  - 散乱実験型
  - 粒子間相互作用型
  - 密行列型
- 可能ならばアプリケーションを書き換える

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

GRAPE-DR の開発状況

チップ評価ボードができたところ。これから動作チェック
作ってるのは福重君で私じゃない、、、

GRAPE-DR の「弱点」

- 非常に通信、メモリ転送の多い計算
  - 疎行列の反復解法
  - 流体、波動方程式の陽解法
  - FFT

要するに、

- ノード内バンド幅
  - オンボードメモリのバンド幅、容量
  - ホストとの転送バンド幅
- ノード間バンド幅

が少ない、ということ。

メモリ/ホストバンド幅

実は GRAPE-DR に限った問題ではない。 SIMD 並列で演算器を増やすと必ず起こる。

- ハイエンド GPU
- Clearspeed

どちらも外部メモリ/ホスト通信バンド幅で性能がリミット

GRAPE-DR の場合:
縮約ネットワークが有効な問題では高い性能

それ以外の問題では？

メモリバンド幅を増やす

- 単純に外付メモリを速くする
  - 50GB/s くらい(今のハイエンド GPU 並) ならできる
  - 計算:メモリ速度で 100:1 くらい
  - FFT・流体・QCD で性能出す: 10:1 が必要
  - 500GB/s は大変。 3Gbit/s の線でも 1000 本以上
- チップ積層とかでないと大変

オンチップメモリではどうか？

- 速度はどうにでもなる
- 電気は食うかもしれないけどしれてる
- 容量は問題？
- ファブは？ IBM, N, S, T? 任天堂はN？

eDRAM の使用例

BG/L	130nm	4MB	11mm by 11mm
QCDOC	180nm	4MB	13mm by 13mm
PSP	90nm?	4MB	?
Xbox 360	90nm?	10MB	?

- BG/L の場合: 1.2Mbits/mm^2
- 65nm, 256Mbits なら 50mm^2
  もとが大きなチップなら OK!

32MB 載せると

- 1 チップ 1 Tflops (GRAPE-DR の 4 倍、結構無理)
- 20,000 チップ (全体ピーク 20Pflops)

とすると、メモリ 640 GB — これは結構大きい。

- QCD 計算: 200^4 くらいの格子なら入る？
  - ノード間通信のことは後で
  - チップ内部構成の話は今日は省略
- 流体計算 2000^3 くらい？
  - 圧縮性流体 (陽解法) なら領域分割で大きな計算も可能
  - 陰解法ではポアソン方程式が問題。これも後で

ノード間通信は？

- 流体計算:あまり問題にならない
  チップ内に入るメッシュが大きいため
- 自由度が小さい計算 (QCD 等) では問題

参考: QCDOC のバランス
1 Gflops : 500Mbits/s ×8 (4D トーラス)
= 0.5 bytes/flops

1 Tflops の場合でも 0.25 bytes/flops くらい欲しい
(扱う格子のサイズはあまり変わらない)

- APE、筑波の数字はもうちょっと小さい?

チップ 1 つが Tflops とすると、これは 250GB/s。
非現実的？

現実的なネットワーク

今あるものの例:

Cray XT3: 7.6GB/s 3次元トーラス。
ノード当りの売り値 100万くらい

- 計算機メーカーなら4年後にこの5倍くらいできる？
- 我々では無理

XT3 並の 10GB/s 3D トーラスを作ったとすると:

- 通信速度は必要量の 1/4
- QCD 計算での効率は 20% くらいはなんとか
- 3次元 FFT でもその程度の効率。 4000^3 ができる
- 実効性能では 4Pflops

作れそうか？

- チップから信号出すのは可能。 1Gbit/s で 1,000本
- ルータ: 我々には技術、経験ない
  → やるならどこかと共同開発

「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
- メッシュネットワークを組めるだけの信号線を出す
- メッシュネットワークを使う構成では BG/L, XT3 のような高密度実装
- 使わないなら 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)
- FFT、QCD(メッシュネットワーク作った場合)
  効率 10–20% (2–4 Pflops)

本当は、

- 価格あたりの性能
- 消費電力あたりの性能

だけが問題。いわゆる「実行効率」は無意味な指標。

(但し、性能推定には実行効率は有用)

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

まとめ

- GRAPE-DR 的な 1 チップ超並列計算機に
  - そこそこの量のオンチップメモリ
  - そこそこの速度のネットワーク

  をつけることで、GRAPE-DR の弱点である
  - 陽解法流体
  - QCD 計算
  - 3D FFT

  でかなり良い実効性能がでるようにできる
- ネットワークはそれでもコストに響く
- ただし、チップ数が多いアプローチより有利
- ネットワーク部分はどこかとの技術協力が必要

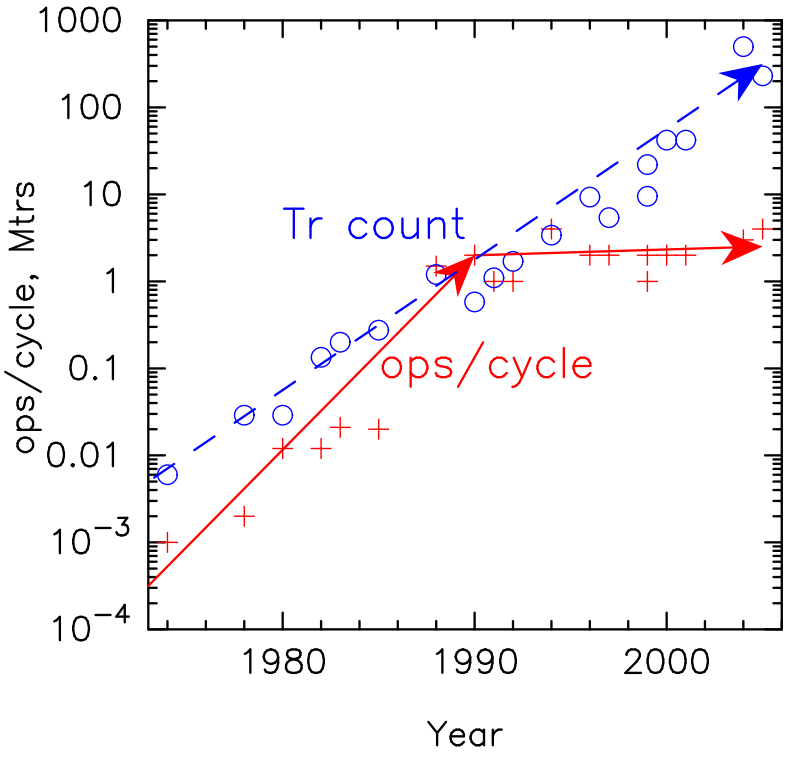
予備資料

Memory Wall への対応法

- ベクトルプロセッサ
  - 正攻法で高速、多数のメモリ配線を用意
  - 利点: 多くのアプリケーションプログラムで高い実効性能
  - 欠点: 高価
- スカラープロセッサ
  - 多階層のキャッシュ
  - 利点: 安価
  - 欠点:
    - 設計が複雑
    - アプリケーションを性能がでるように書くのは困難
      - キャッシュの振る舞いを間接的に制御する必要あり
      - プリフェッチでは不十分

これでいいのか？

- プロセッサチップでのトランジスタの利用効率
  - 1990 年から一貫して低下を続けてきた
- スカラープロセッサ
  - ほとんどの面積がキャッシュとその制御回路
- ベクトルプロセッサ
  - ほとんどの面積が I/O とベクトルレジスタ



トランジスタの有効利用法を考え直す時期

システム構成の考え方

- アプリケーション実行はホスト計算機との連携で行う
- できるだけ強力なホスト計算機・ネットワークを使いたい
- 以下に示すシステム構成は最低線のもの

ベースライン構成

- ホスト計算機: x86 サーバベースのクラスタ
- ネットワーク: InfiniBand ベースの fat tree
- V-GRAPE: PCI-Express ベースの
  拡張カードとして実現
- ノード計算機については始めから予備を用意

計算ノード

VGRAPE カード:	PCI-Express Gen2 x16
VGRAPE チップ:	4 個
消費電力:	350W
メモリ:	128MB
演算ピーク性能:	2.8Tflops
ホストとの通信性能:	8GB/s 双方向
トータルオンチップメモリバンド幅:	1TB/s
ホスト計算機:	通常の x86 Dual CPU
メモリ:	32GB
ネットワーク:	双方向 2GB/s (DDR IB

ラック

- 16(+2) ノード
  - 2 ノードは予備
- ピーク性能
  - VG 部: 44.8 Tflops
  - ホスト部: 1.6Tflops
- メモリ量: 512GB (G8 部 16GB)
- 消費電力: 約 10KW

システム全体

- **計算ノードラック数:** 448
- **計算ノード総数:** 7168(+896)
- **ピーク性能** (Pflops): 20.07+0.72

LINPACK で性能を出すための十分条件

1. 部分軸選択にかかる時間 (主に最大値選択) が計算時間より短い
2. 分解したパネル 1 枚を転送する時間がパネルの行列乗算の時間より短い
3. メモリ階層の各レベルでのメモリサイズに対してバンド幅がブロック化した行列乗算の要求を満たす

(1), (2): ホスト計算ノードの主記憶サイズと通信バンド幅、レイテンシの関係で決まる

(3): V-GRAPE チップのメモリ階層 (PE メモリ、オンボードメモリ、ホストメモリ) とバンド幅の関係で決まる

検討結果の要約

- 計算の詳細は提案書に
  - ボトムライン: ピークの 50% を出すのは難しくない。
  - 参考: 50Gflops ピークの Clearspeed CSX600 で 30Gflops
    - PCI-X の通信ネック、チップ内縮約が高速にはできないことなどから V-GRAPE より不利

プログラミング環境

- 加速ボードのメーカーが必ずいうこと:
  - アプリケーションプログラムから加速ボード側で実行できる部分を自動検出、変換できます
  - ソースコードに手を加えずに性能向上可能です
- 30 年前から同じことをいわれてきた
  - 今更騙される人はいない
- より現実的、効果的なアプローチ
  - 加速ボードでは単純なカーネルルーチンだけを実行
  - 他の部分はホスト上のプログラム。これは「基本的には」改変なし

V-GRAPE で提供する環境

- カーネルルーチン
  - 行列乗算、対角化等の行列演算ルーチン
    - BLAS, LAPACK のサブルーチンの形に
  - 粒子間相互作用計算ルーチン
    - 単純な粒子間相互作用程度はアセンブラでも書ける
  - 二電子積分、交換積分などグランドチャレンジに必要なルーチン
- アセンブラ
  - 昔はみんなこれで書いていた
- PE 上でのコードに対して、「高級言語」を提供
  - PGDL (理研 濱田、中里、FPGA 再構成計算用コンパイラ)
  - SPH 法のための専用パイプライン (演算器150) の合成に初めて成功

京速計算機システムでの位置づけ

- ホスト計算機
  - プログラム全体を実行
- V-GRAPE
  - 計算負荷が大きな部分を汎用的なシステムからオフロードするためのサブシステム

ソフトウェアの継承性

カーネルルーチンだけに限る:

- それ以外の部分は計算機に依存しない
- その部分だけ移植すればよい
- 実はもっと汎用的な計算機でもその部分だけチューニングすればよい

アプリケーションのうち、マシン依存で変更するべき部分を特定することと同等

普通の並列計算機と何が違うか？

並列計算機の古典的な分類 (M. Flynn)
SISD/SIMD/MISD/MIMD

同時に処理される

- **命令列が一つ** (SI) **か複数** (MI) **か**
- **データ列が一つ** (SD) **か複数** (MD) **か**

この分類に入れるなら SIMD**。過去の** SIMD **機とは色々違う。**

- **上のレベル** (**並列ホスト**) **では** MIMD
- **接続ネットワークを相互作用計算に最適化**

接続ネットワーク

何故接続ネットワークが問題か？

SIMD 並列計算機を GRAPE として使う (粒子間相互作用の計算に使う) ことを考える。
単純な使いかた: 1000 個プロセッサがあれば、1000 個の粒子への力を計算。
利点:

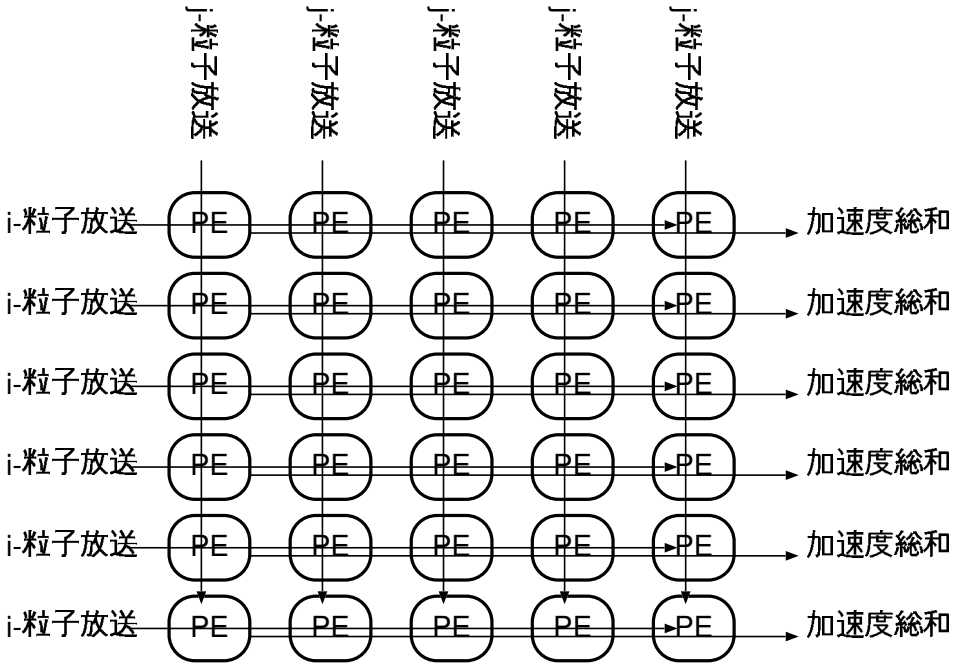
- 単純なネットワークでいい (放送とランダムアクセス)
- メモリもプロセッサあたり少しだけあればいい

問題点:

- 独立時間刻みでは 1000 は多すぎる (複数チップ/ホストでもっと悪くなる)
- チップの高速動作が困難になる。

みかけ上の並列度を下げる

複数のプロセッサ (PE) が同じ粒子への別の粒子からの力を計算、後で合計 (j-並列)



- 合計はチップ内でする必要あり (GRAPE-6 でボード上でやっているのと同じ)
- 2 種類の「放送」が必要:論理的にプロセッサは 2 次元配列、縦、横両方の放送

実際のチップ内ネットワーク

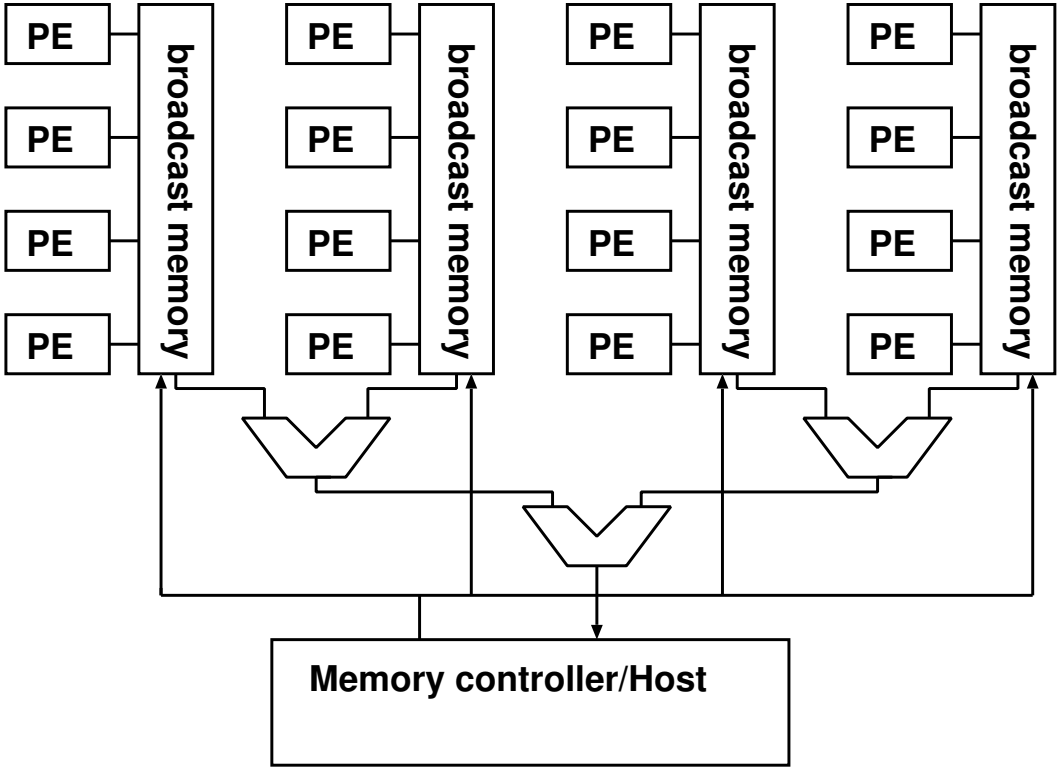
PE は放送メモリとだけ接続

放送メモリからそれにつながった PE には

- 放送
- ランダム読み書き

チップ外ポートから放送メモリには

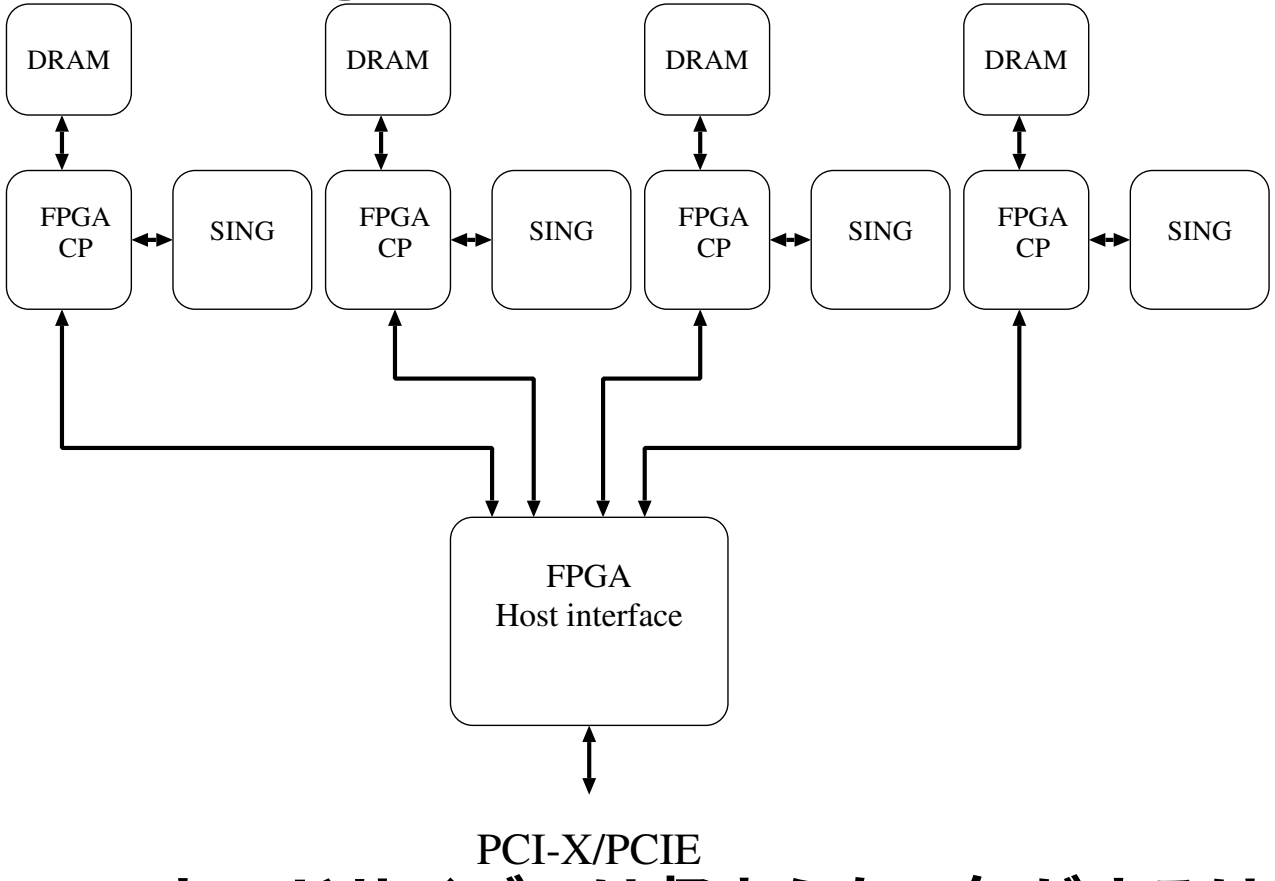
- 放送
- 縮約 (総和など) しながら読み出し
- ランダム読み書き



これで必要なことは全てできる。

ハードウェアのイメージ

SING: Sing Is Not GRAPE、チップの開発コードネーム



PCI カードサイズでは収まらない気がするけど、、、

どうやってプログラムするか？という話

- 今までの GRAPE とは全然違う: プログラムを書く必要がある
- 古典的な SIMD マシンともプログラミングモデルが違う
  - GDR の部分は「付加プロセッサ」:そこでプログラム全体が走るわけではない
  - 通信モデルが違う
  - 制御プロセッサはハードウェアレベルで変更可能(FPGA で実装)

制御プロセッサの中身を決める = プログラミングモデルを決める

アセンブラ定義の概要

- 放送メモリ、 PE ローカルメモリに変数を置くことができる
- PE レジスタはアドレスで直接指定。(変数を置けるように変えるかも)
- ベクトル変数とスカラー変数がある
- 並列に実行する命令は明示的に指定する。

アセンブラの例

単純な重力計算の例

```
var vector long xi     hlt  flt64to72
var vector long yi     hlt  flt64to72
var vector long zi     hlt  flt64to72
var vector short idxi  hlt  fix32to36ru
bvar long xj           elt  flt64to72
bvar long yj           elt  flt64to72
bvar long zj           elt  flt64to72
bvar long vxj xj
bvar short mj        elt  flt64to36
bvar short eps2      elt  flt64to36
bvar short idxj      elt  fix32to36ru
var short  lmj
var short  leps2
var short  lidxj
var vector long accx rrn  flt72to64 fadd
var vector long accy rrn  flt72to64 fadd
var vector long accz rrn  flt72to64 fadd
var vector long pot  rrn  flt72to64 fadd
```

ここまでで変数宣言。 hlt, elt, rrn は通信タイプ。そのあとのは変換タイプ

アセンブラの例(続き)

```
loop initialization
vlen 4
uxor $t $t $t
upassa $ti $ti $lr40v
upassa $t $t   $lr48v
upassa $t $t   $lr56v
upassa $t $t   pot
loop body
vlen 3
bm vxj $lr0v
vlen 1
bm mj lmj
bm eps2 leps2
bm idxj lidxj
```

初期化(ループ前に実行)とループ本体の一部(放送メモリからの転送)

アセンブラの例(続き)

```
vlen 4
nop
upassa idxi  idxi  $t
uxor  $ti lidxj $t
moi 2
ulnot $ti $ti $t  # mreg 1 indicates i != j
moi 0
nop
fsub $lr0 xi $r6v  $t
fsub $lr2 yi $r10v ;  fmul $ti $ti $t
fsub $lr4 zi $r14v
fmul $r10v $r10v $r18v ; fadd $t leps2 $t
fmul $r14v $r14v ;  fadd $fb $ti  $t
fadd $fb $ti  $r18v $t # rsq is now in r18 t, dx, dy,dz are in
```

自己相互作用のチェック、座標の引き算と距離の 2 乗の計算

アセンブラの例(続き)

```
ulsr $ti      il"60"  $t $lr22v
ulsr $ti      il"1"   $t
uadd $ti      $lr22v  $t
usub hl"9fd" $ti      $t        # $lr8v は指数の1.5倍
ulsl $ti      il"60"  $lr30v
moi 1
uand il"1" $lr22v
moi 0
uand $r18v h"000ffffff" $t
uor  $ti h"3ff000000" $t
fmul $ti f"0.57" $t
fsub f"1.57" $ti $t
mi 1
fmul f"1.414" $ti $t
mi 0
nop
fmul $t $lr30v $t $r22v # Here the result is the initial guess
```

r^{-3} の初期推定値。これは手抜きな1次式

アセンブラの例(続き)

```
 fmul $r18v $r18v $r26v $t
 fmul $r18v $ti   $r26v $t
 fmul $ti f"0.5" $r26v # r26v is a**3/2
 fmul $r22v $r22v $t
 fmul $ti $r26v $t
 fsub f"1.5" $ti $t
 fmul $r22v $ti $t $r22v
 fmul $ti $ti $t
 fmul $ti $r26v $t
(反復ちょっと省略)
 fsub f"1.5" $ti $t
 fmul $r22v $ti $t $r22v
 fmul $ti $ti $t
 fmul $ti $r26v $t
 fsub f"0.5" $ti $t
 fmul $r22v $ti $t
 fadd $r22v $ti $t
 fmul lmj $ti $t $r22v
```

ニュートン反復

アセンブラの例(続き)

```
mi 2
fmul $r6v $ti ; upassa pot pot $lr0v
fmul $r10v $t ;  fadd $fb $lr40v  $lr40v accx
fmul $r14v $t ; fadd $fb $lr48v  $lr48v accy
fmul $r18v $t ; fadd $fb $lr56v $lr56v accz
fadd $fb $lr0v  pot
```

ポテンシャルと加速度の積算
この記述から、インターフェース関数とシミュレータを動かすのに必要なデータを生成する。

インターフェース関数

中身はアセンブラ記述から自動生成される。

```
int SING_send_j_particle(struct grape_j_particle_struct *jp,
                         int index_in_EM);
int SING_send_i_particle(struct grape_i_particle_struct *ip,
                         int n);
int SING_get_result(struct grape_result_struct *rp);
void SING_grape_init();
int SING_grape_run(int n);
```

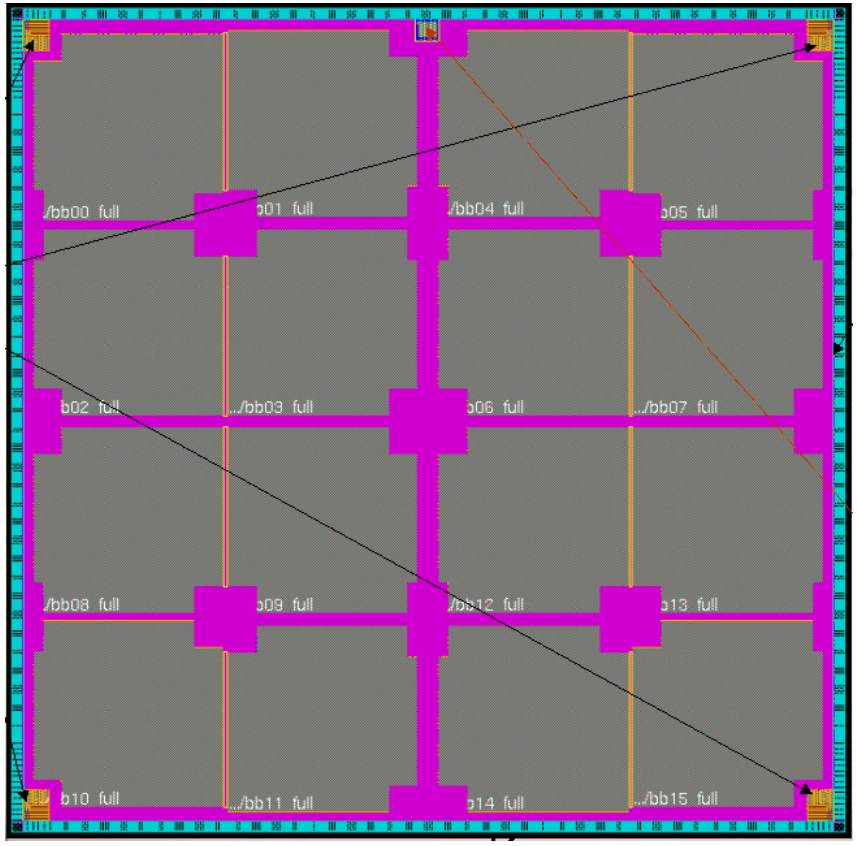
これにもう一層かぶせれば GRAPE-3/5 互換インターフェースはできる。

インターフェース構造体

これももちろん自動生成される。

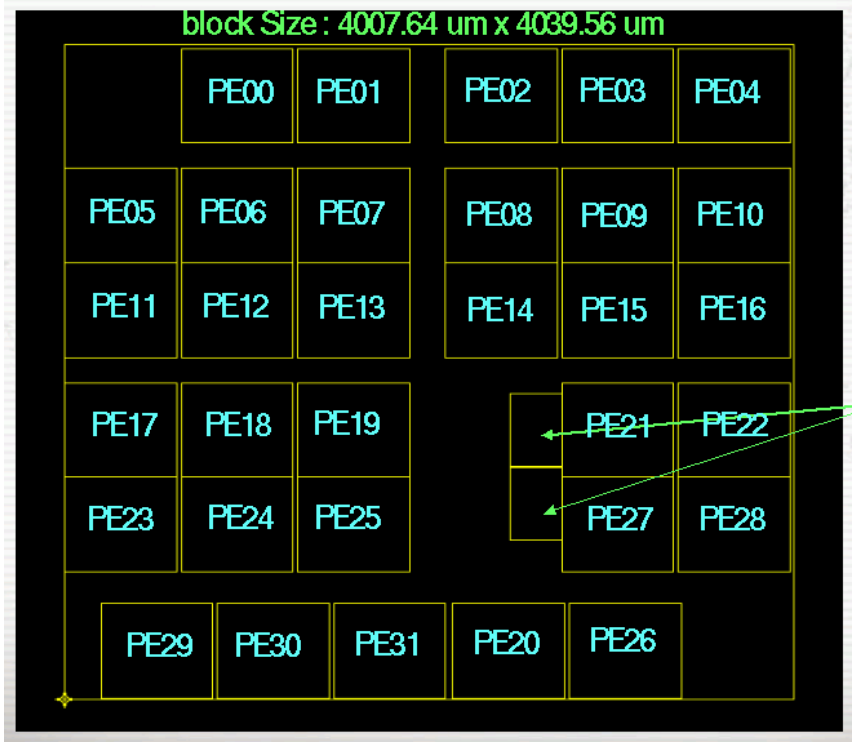
```
struct grape_j_particle_struct{
  double xj;
  double yj;
  double zj;
  double mj;
  double eps2;
  UINT32 idxj;
};
struct grape_i_particle_struct{
  double xi;
  double yi;
  double zi;
  UINT32 idxi;
};
struct grape_result_struct{
  double accx;
  double accy;
  double accz;
  double pot;
};
```

チップ全体フロアプラン



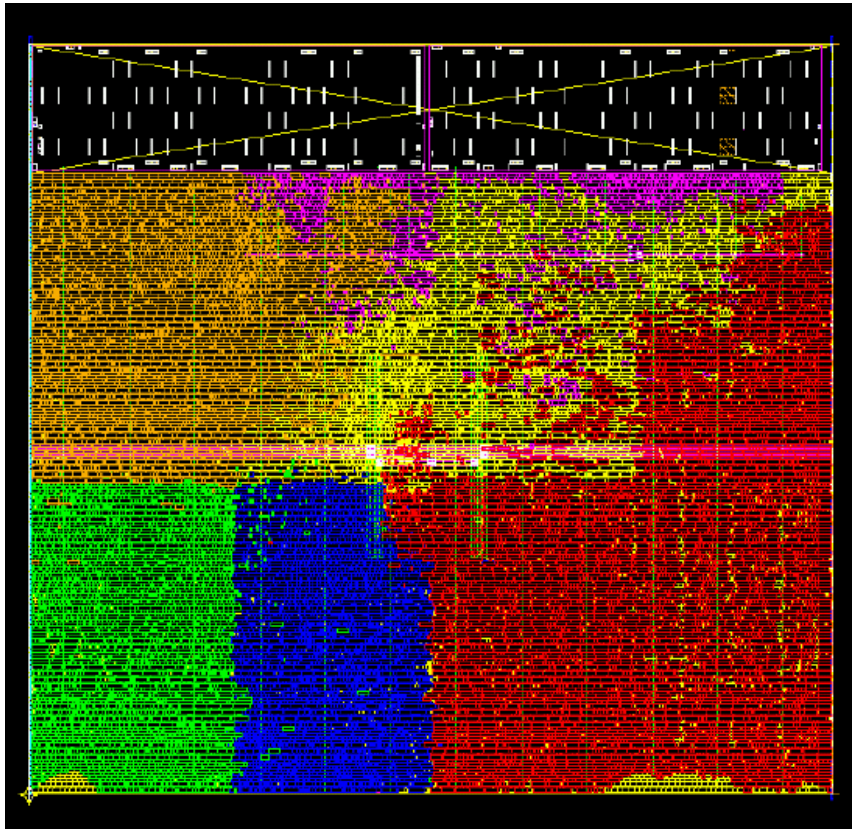
- 特におかしいとこなし
- サイズは大きい。
  17mm 角

放送ブロックフロアプラン



- 特におかしいとこなし

PE フロアプラン



Module Name	Color
grf0	red
fmul	orange
fadd	green
alu	blue
lm	magenta
Others	yellow