GRAPE Project
— 専用計算機による計算科学

牧野淳一郎
東京大学理学系研究科天文学専攻

あるいは

世界一速いコンピュータ(の作り方)

(と講義予定には書いてあるような気がする)

牧野淳一郎
東京大学理学系研究科天文学専攻

この講義の大雑把な予定

1. 計算科学と計算「機」科学
2. 天文学における多体シミュレーションの役割
3. 多体シミュレーションのアルゴリズム
4. 何故専用計算機か？
5. GRAPE プロジェクトの歴史
6. 実際にどんなふうに作ってきたのか
7. これからどうするか

計算科学と計算機科学

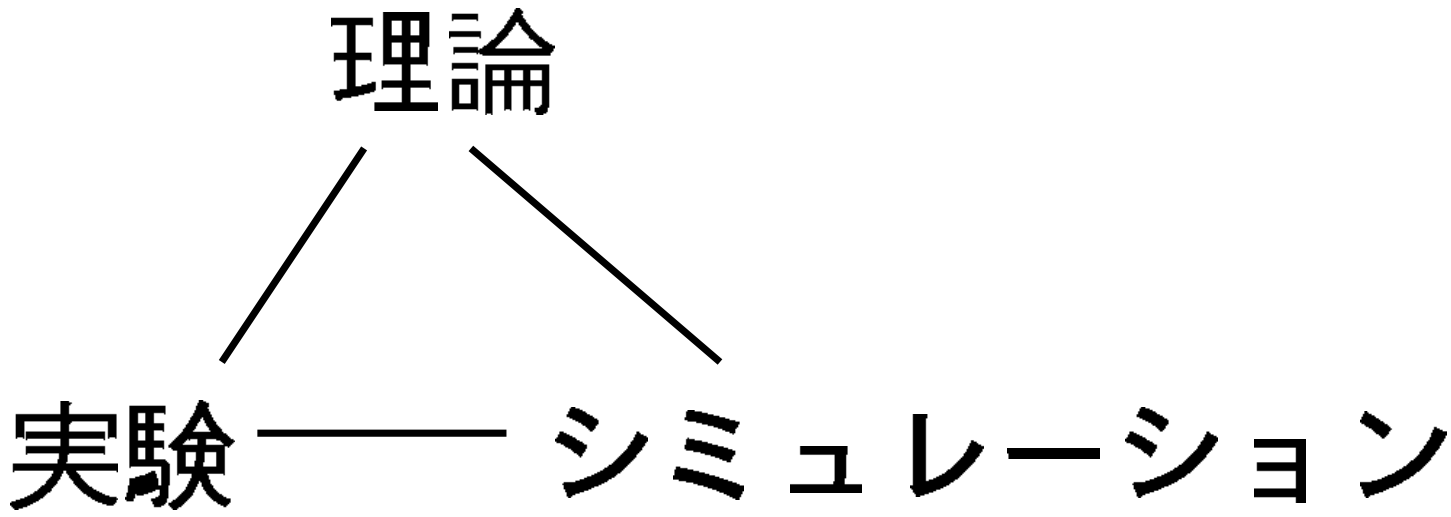
というか、計算科学って何？

- 計算機を使う科学

今どき計算機を使わない科学なんかない。

- たてまえ
- 現実

たてまえ



(紙と鉛筆による) 理論と実験のどちらとも違う科学の方法論

と言われても今一つよくわからない。

たてまえつづき

筑波大学計算機科学研究センター長挨拶

計算科学とは、**超高速計算機の先端的な適用**により、自然の現象の背後に潜む法則を発見・理解し、その帰結を探り、またさらにこれを技術的応用に展開させるディシプリンです。

普通の計算機では計算科学ではない？何故？

計算科学とはでは何か？

原理的には計算機シミュレーションで役に立つことがわかるはずのことで

しかし現在の普通の計算機で普通の計算法でやったのでは役に立たないことを

普通でない計算機をなんとかしたり普通でない計算法をなんとかすることで役に立つようにしよう

という学問。

まあ、こういう問題はいくらでもある。

計算機科学との関係

計算機科学との関係

良く知らない。

計算機科学との関係

良く知らない。

普通に考えると

- **計算機を作る：計算機科学 (工学) の一部**
- **計算法：応用数学**
- **それらを使って計算する：個別の科学**

こういう分業を無視する、というのが今日の話。

天文学におけるシミュレーションの役割

- 観測の解釈 — 理論＋モデル計算
  知られている物理法則から天体現象を理解する
- もうちょっと原理的な問題
  カオス
  熱力学

もうちょっと原理的な問題

重力多体系:
重力だけで相互作用する質点の集まりには一体なにが起きるのか？

粒子数 $N = 2$: ニュートンが解決
$N > 2$ ポアンカレが「一般には解析解はない」ことを示した

球状星団：$N \sim 10^6$
銀河: $N \sim 10^{11}$
なにが起きるか？
数値計算なら、初期条件を与えれば答はでる。

計算すればなんでもわかるか？

原理的には答は YES。
現実の問題としては NO。

- 計算機の能力
- 素過程の理解

この 2 つのどちらが大きな問題かは分野によって違う。
一方で他方をカバーしよう、というのももちろんある。

例:銀河の進化

典型的な銀河: 1000 億個くらいの星、それと同程度の質量のガス、その数倍の質量の「ダークマター」からなる (ということになっている)

- 1000 億体問題は今の計算機では全く無理
- 1000 億体問題が解けても、それだけでは駄目
  - ガスから星ができる過程
  - 超新星爆発等で星がガスに戻る過程
  - ダークマターはどうするか

とはいえ、今日の話は主に 1000 億体問題、あるいは 100 万体問題をどうやって解くかという話。

重力多体系の基礎方程式

もとの方程式自体はもちろん、各粒子の運動方程式

$$\frac{d^2 \boldsymbol{x}_i}{dt^2} = \sum_{j \neq i} G m_j \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{|\boldsymbol{x}_j - \boldsymbol{x}_i|^3}, \tag{1}$$

数値計算は基本的にはこれを使う

数値計算の方法等

この種の問題：基本的に

- **より大粒子数で**
- **より正確な**

計算をすることで、「新しいことがわかる」（こともある）。

↓

どうやって今までより「良い（大きく、正確な）」計算ができるようにするかが問題

「良い」計算をする方法

基本的な事実: 速く計算できればそれだけ良い計算が可能になる。

- **計算法を改良する**
- **速い計算機を買う**
- **速い計算機を作る**

以下、それぞれの話をする。

計算法

原理的には、多体シミュレーションはとっても単純：
運動方程式

$$\frac{d^2 x_i}{dt^2} = \sum_{j \neq i} G m_j \frac{x_j - x_i}{|x_j - x_i|^3}, \tag{2}$$

を数値積分するだけ。

右辺を計算するプログラム： 2 重ループで 10 行くらい
時間積分： なにかルンゲクッタとか適当なものを使えばいい

というだけで話が済めばいいけれど、もちろん世の中はそんなに簡単ではない。

何が問題か？

- 計算精度の問題： 2粒子の近接散乱、自己重力による構造形成 — 時間刻みをどんどん短くしないとちゃんと計算できなくなる。
  積分時間が長いので高精度の公式を使いたい。
- 計算量の問題： 右辺の計算量が $O(N^2)$ — N が少し大きくなるとすぐに計算時間が現実的ではなくなる

というわけで、以下、時間領域での計算法と空間領域のそれを概観する。

計算法 — 時間領域

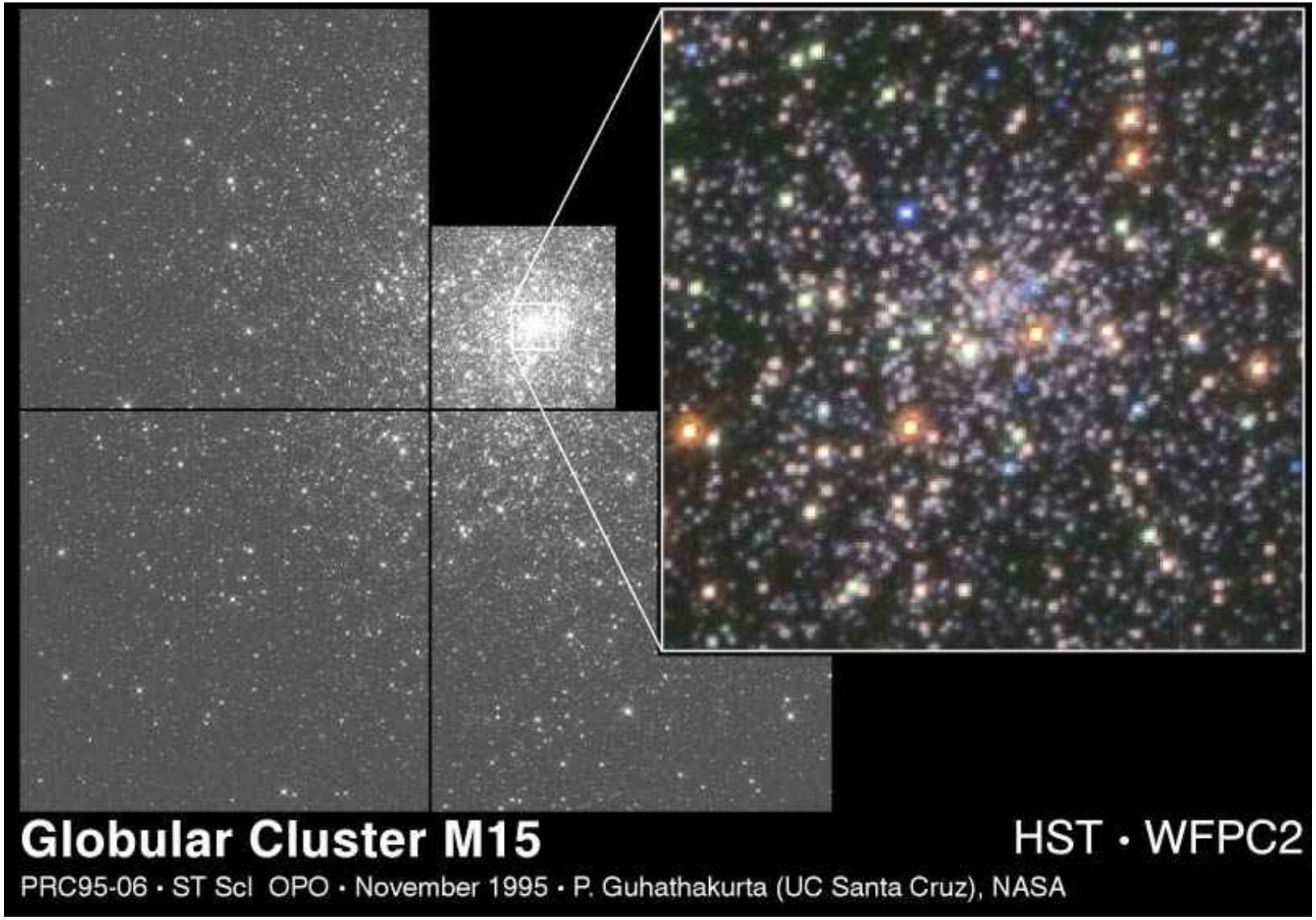
算数としては、単に常微分方程式の初期値問題の数値解。
ナイーブに考えると、いろんな公式がライブラリであるので、
それを使えば済みそうな気がする。
それだけでは済まないのが問題。
済まない理由:

- 粒子によって非常に大きく軌道のタイムスケールが違うことがある
- 連星とかそういったものができる

軌道タイムスケールの問題

- 構造形成による効果
- 一様な系でも起きる問題

構造形成による効果


Globular Cluster M15
HST • WFPC2
PRC95-06 • ST ScI OPO • November 1995 • P. Guhathakurta (UC Santa Cruz), NASA

球状星団 M15

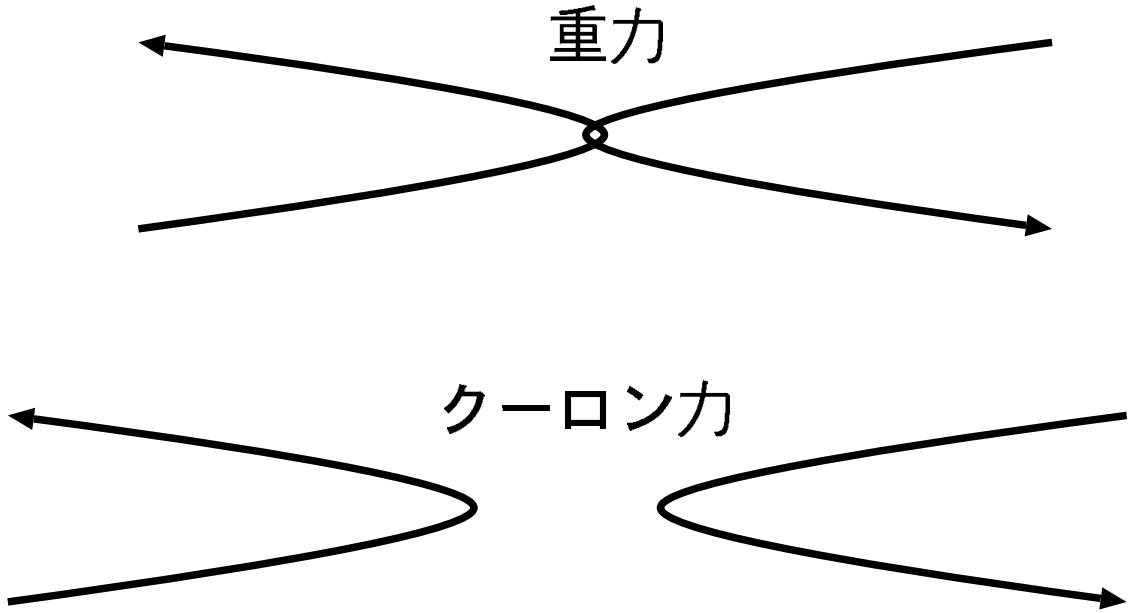
- 「コア崩壊型」星団 — 星の数密度が中心までべき (半径の -1.8 乗) で増加
- 中心にブラックホールがある？

一様な系でも起きる問題

重力が引力であるために2つの粒子が非常に近付くことがある

非常に短い時間刻みが必要

重力多体系特有の問題、分子動力学計算ではこういう問題はおこらない。



計算量への影響

単純な可変時間刻みでは計算量が大きくなる。

理由: どちらの場合も、タイムステップの分布がべき乗的なテイルをもつようになる。

粒子数 N が増えるに従って、タイムステップが短くなる。
構造形成の効果: 最悪 $O(N^{1.3})$
2 体衝突の効果: $O(N^{1/3})$ 程度

対応:

- **粒子毎に時間刻みを変化させる。(独立時間刻み)**
- **2 体衝突、連星は座標変換して扱う。**

独立時間刻みの原理

粒子毎にばらばらの時刻 t_i と時間ステップ Δt_i を与える

1. **$t_i + \Delta t_i$ が最小の粒子を選ぶ。**
2. **その粒子の軌道を新しい時刻まで積分する。**
3. **その粒子の新しい時間刻みを決める。**
4. **ステップ 1 に戻る。**

ある粒子の時刻 $t_i + \Delta t_i$ で他の粒子の位置が高精度で必要:
予測子・修正子型の公式を使う。

計算法 — 空間領域

運動方程式の右辺をどうやって評価するか？という問題。

以下、 独立時間刻みのことはとりあえず棚上げにして話をすすめる。

広く使われている方法： Barnes-Hut treecode

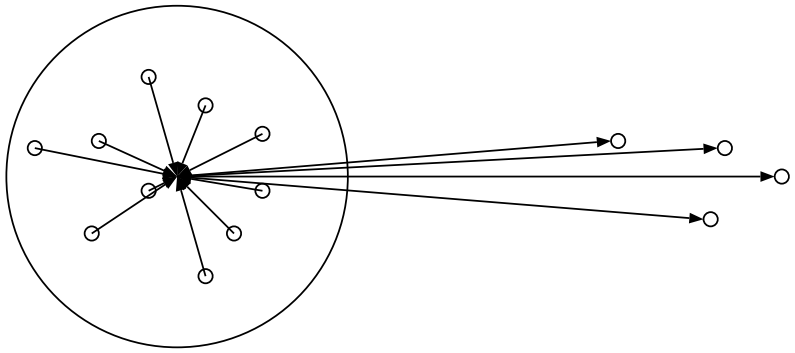
有名な方法： 高速多重極法

ツリー法、FMMの基本的発想

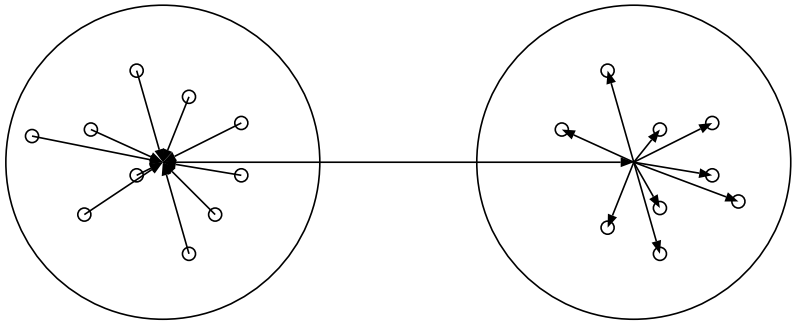
遠くの粒子からの力は弱い

↓

まとめて計算できないか？



Tree



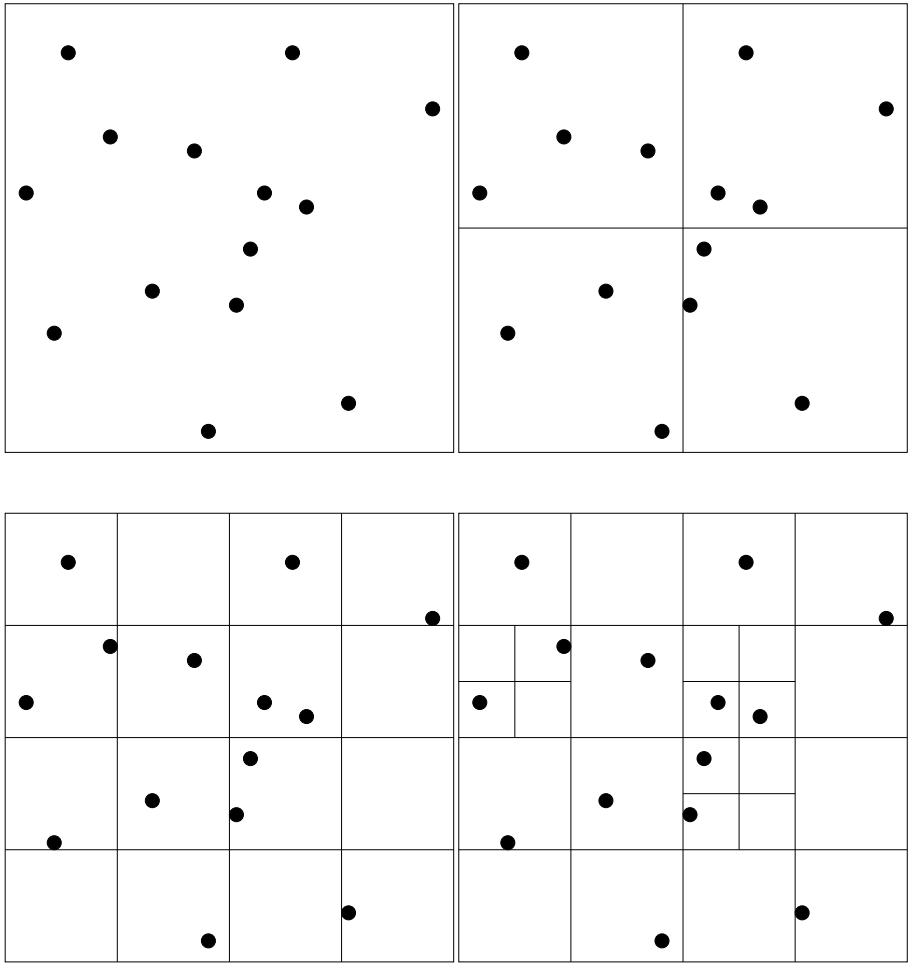
FMM

- **ツリー：力を及ぼすほうだけをまとめて評価**
- **FMM：力を受けるほうもまとめて評価**

どうやってまとめるか? — ツリー法の場合

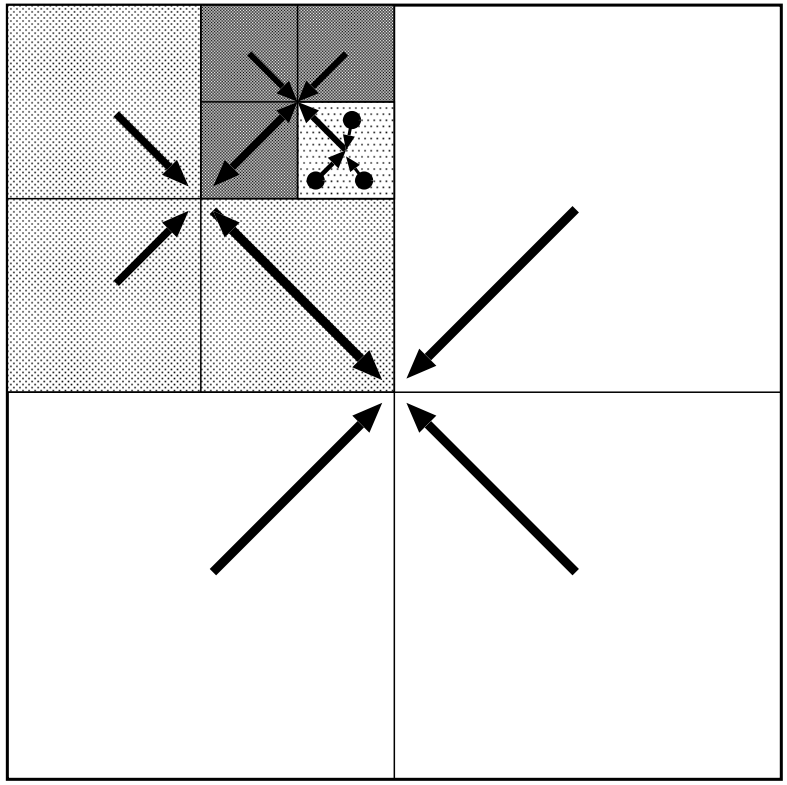
階層的なツリー構造を使う。

- まず、全体が入るセルを作る
- それを再帰的に 8 (2 次元なら 4) 分割する
- 中の粒子がある数以下になったら止める（上の例では 1 個）



多重極展開の構成

まず、ツリーの各セルのなかの粒子がつくるポテンシャルの多重極展開を計算する。

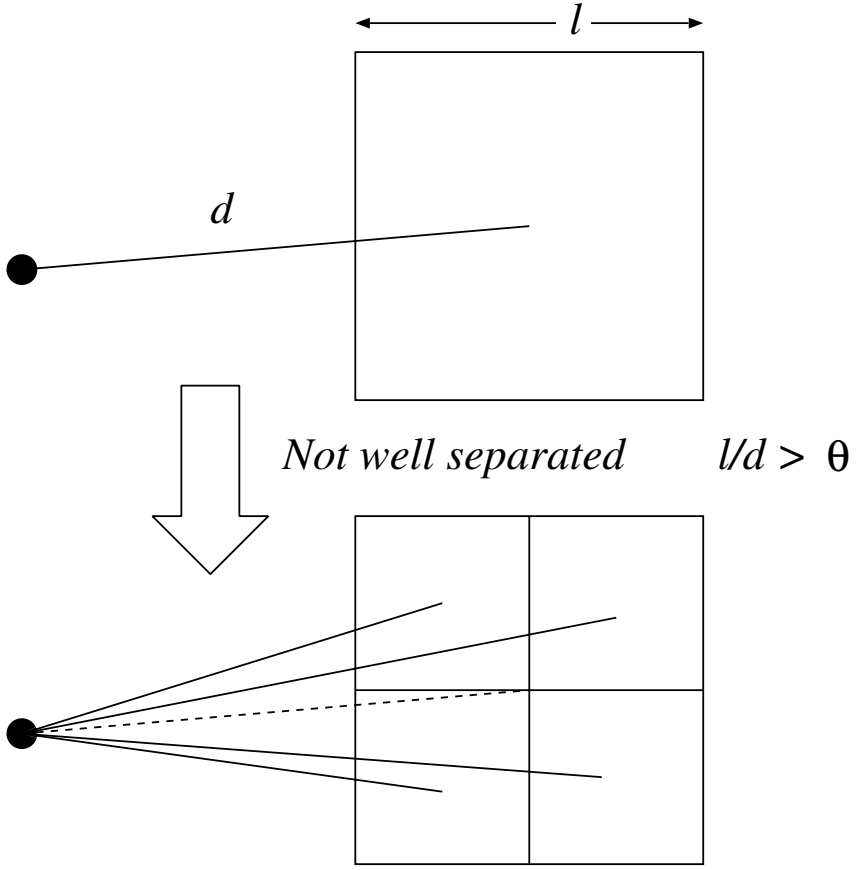


- 最下層のセル:そのなかの粒子が作るポテンシャルを多重極展開
- それ以外：子セルの多重極展開の展開中心をシフトして加算

下から順に計算していけばよい。計算量は $O(N)$。展開をシフトする式はかなり複雑。

ツリー法での力の計算

再帰的な形に表現すると格好がいい。



- **十分に離れている:重心（あるいは多重極展開）からの力**
- **そうでない：子ノードからの力の合計**

系全体からの力 $=$ ルートからの力

ツリー法の計算量と計算精度

簡単なオーダー見積り:

$$\text{誤差} \propto \theta^{(p+1)}$$

$$\text{計算量} \propto \theta^{-3} p^2 N \log N$$

p:多重極展開の次数

$\log N$: ツリーの階層数

θ^3 ある階層で、相互作用を計算するセルの数

実際には、、、

実際の振舞い：複雑。

- **精度は上の見積りよりよい（θ のべきが大きい）**
- **計算量はそれほど増えない**

ことが多い。（粒子の分布による）

高速化の第二の方法: 速い計算機を使う

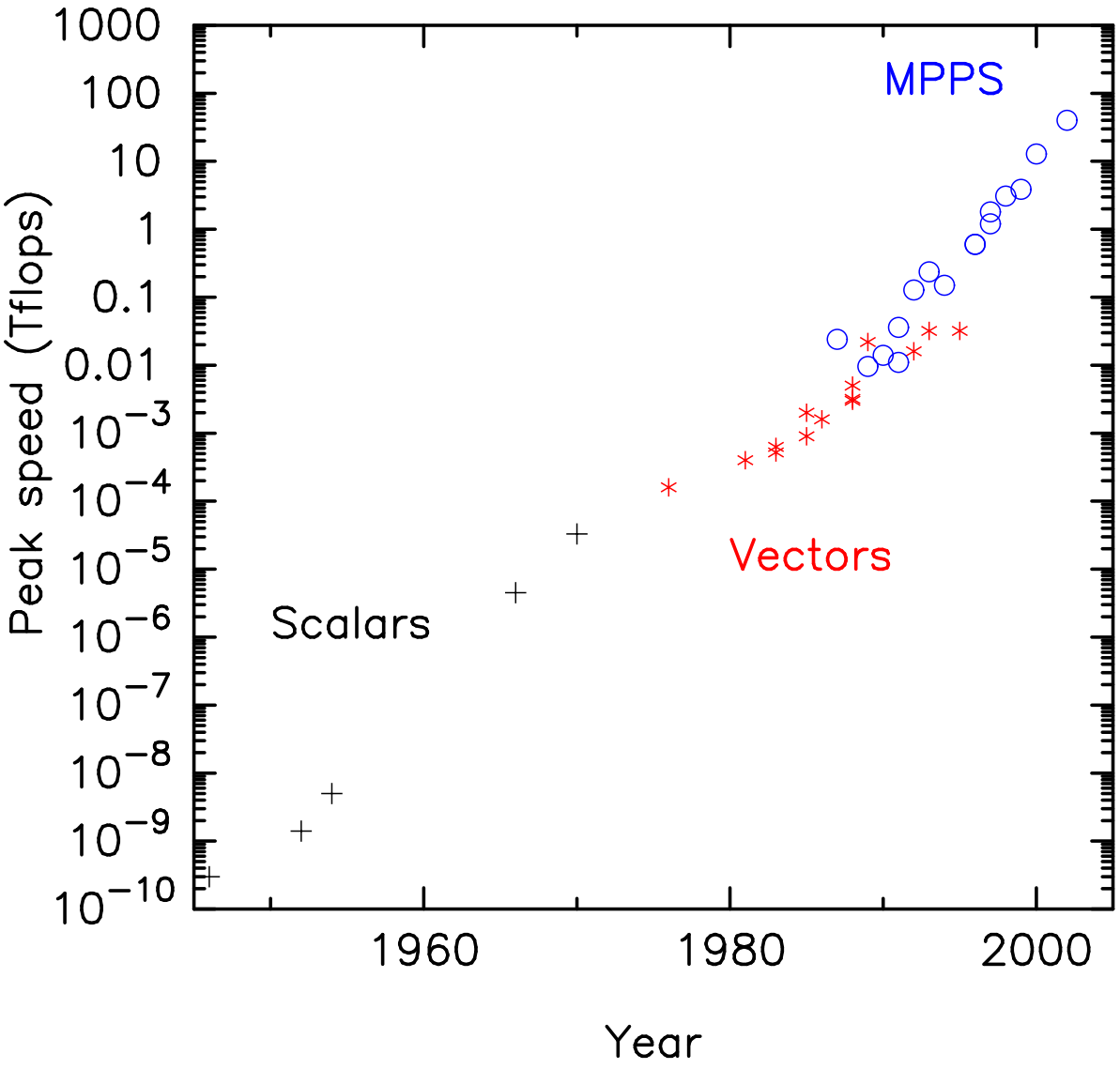
速い計算機を使えば速く計算できるというのはなんかトートロジーみたいで当たり前という気もするが、実はそうでもない。

根本的な理由:
最近 30 年間の計算機の「発展」のおかげで、あるアルゴリズムを速い計算機でちゃんと性能がでるように動かすのはどんどん大変になってきた。

計算機の発展

速度の進化: 50 年で 10^{10} 倍

ほぼ時間の指数関数



指数関数的進化を可能にしたもの

1. ムーアの法則:トランジスタの大きさ: 3 年で半分
   - トランジスタの数: 4 倍
   - 速度: 2 倍
2. 計算アーキテクチャの革命
   スカラー計算機 $\rightarrow$ ベクトル $\rightarrow$ 並列

というわけで、現状での課題:(割合ネットワークが遅い並列計算機上での) 並列化

多体シミュレーションの並列化

- 独立時間刻み
- ツリー・FMM

独立時間刻みの並列化

実はそれ以前の問題: 「全粒子から全粒子への力の計算」の並列化

計算量: $O(N^2)$、あるいはタイムステップが増えることも入れると $O(N^{3.5})$ くらい。

$O(N)$ 以上のプロセッサを使いたい。

$O(N)$ でも係数が小さいと使えるプロセッサ数は結構小さいことがある

並列化の 2 方法

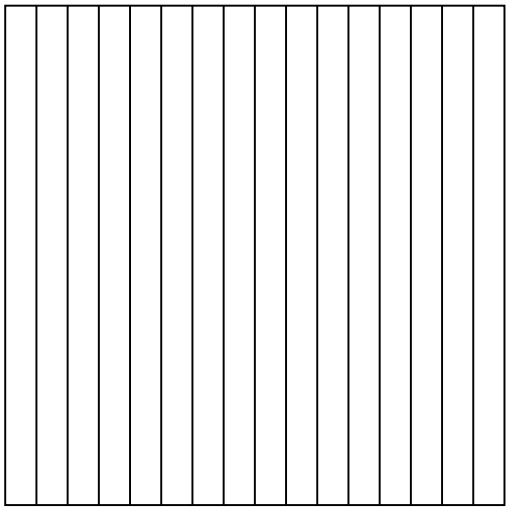
- 各ノードが全粒子のコピーを持って、重力計算、時間積分は分担する
- 各ノードが自分の分担の分だけを持つ。重力計算のためには他のノードから粒子を貰ってくる。

どちらにしても、各ノードが全粒子の情報を使って力を計算する。

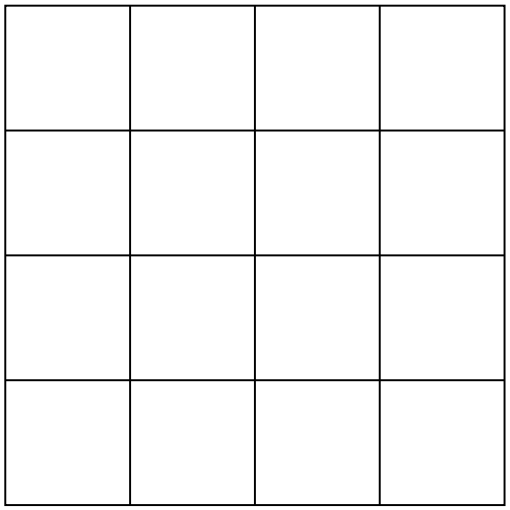
ノード一つの通信バンド幅でスケーラビリティが制限される。

もうすこし違う考え方はないか？

近接相互作用の時（2次元の例）



1-D decomposition



2-D decomposition

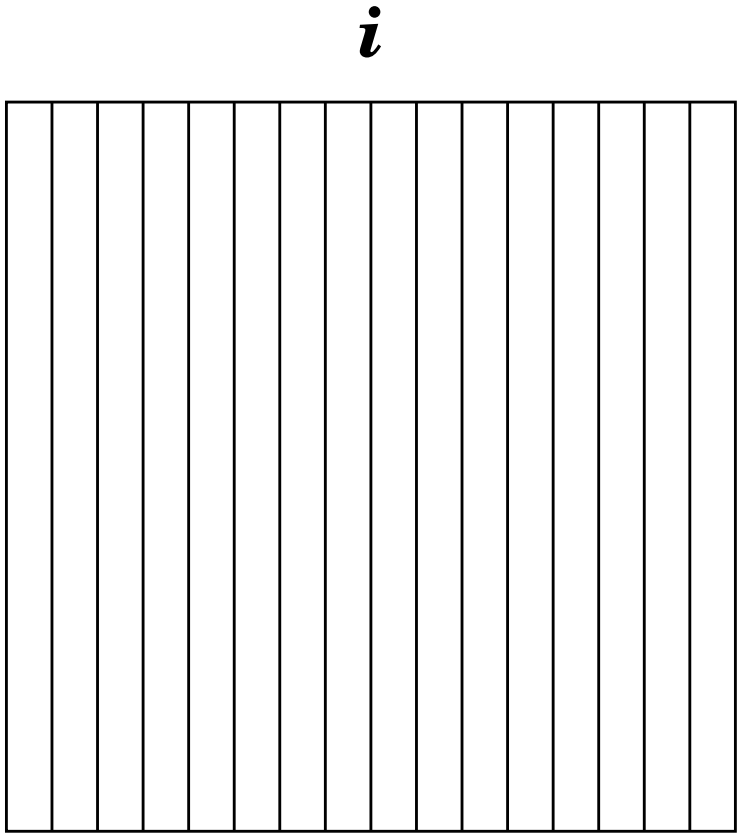
1次元空間分割：スケーラブルでない

ノード数 $\propto$ 1次元方向のメッシュ数

2次元空間分割：スケーラブル

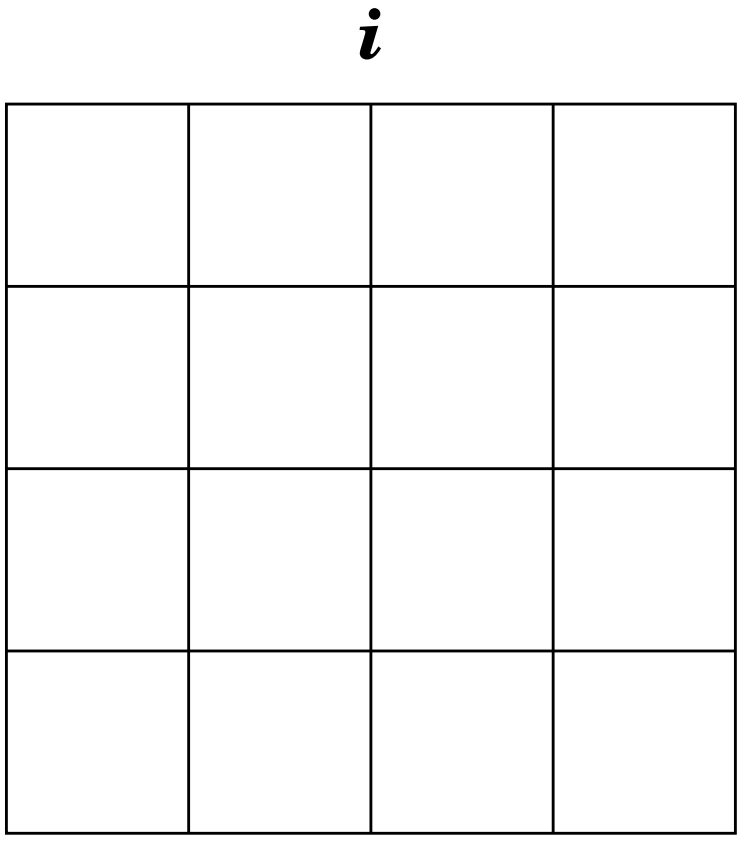
ノード数 $\propto$ 全体のメッシュ数

遠距離相互作用 —— 相互作用行列を考える



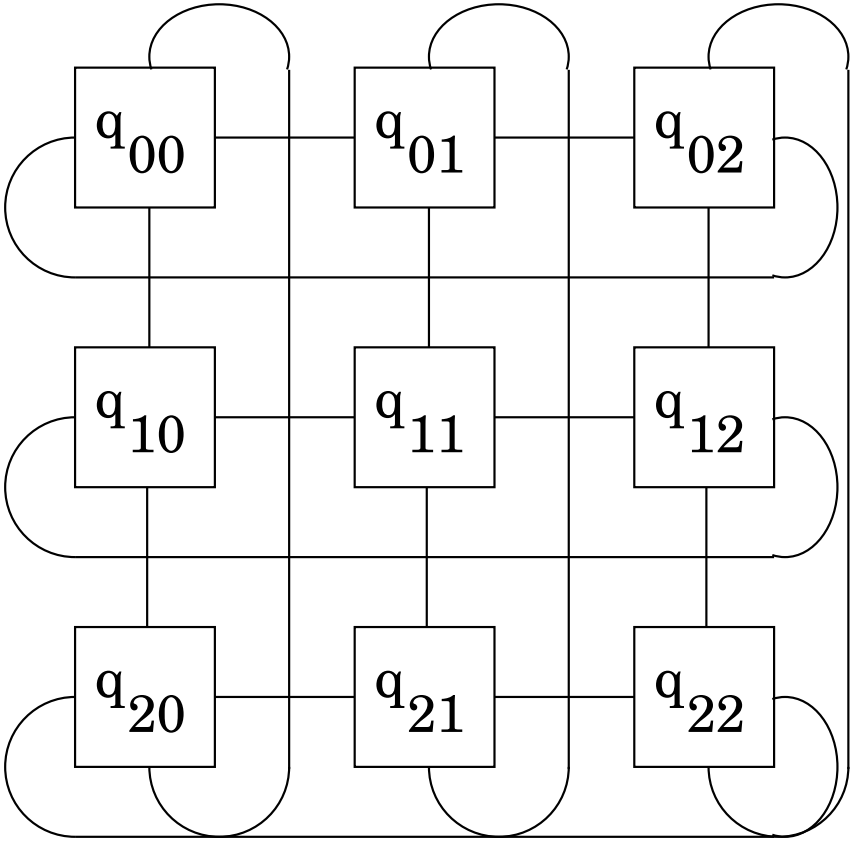
- 相互作用の行列を見ると、普通のアルゴリズムは1次元分割になっている。
- これがスケーラブルでない理由
- 2次元分割は可能か？

相互作用の2次元分割



- 形式的には左のような絵を書ける。
- 実際にはこれはなにを意味しているのか？

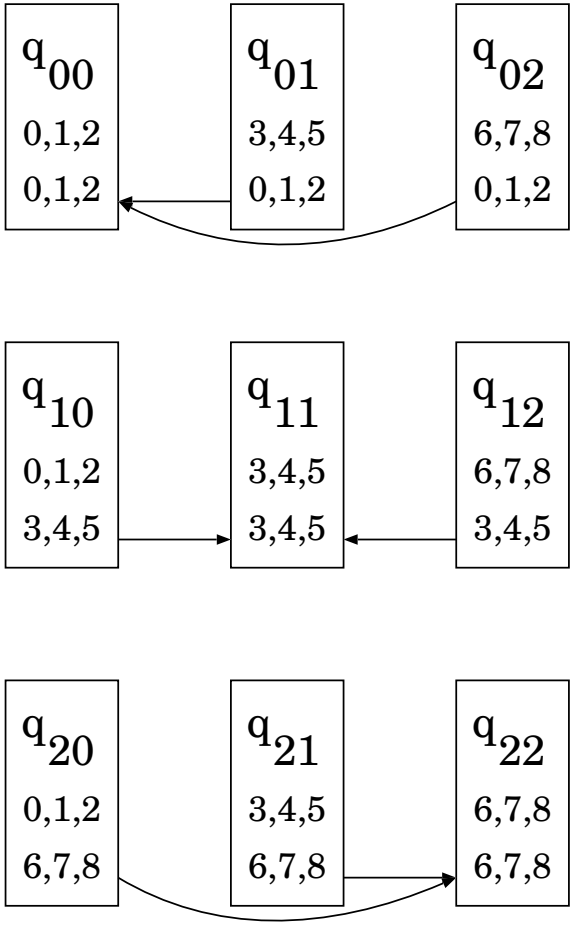
ハードウェアとしては、、、



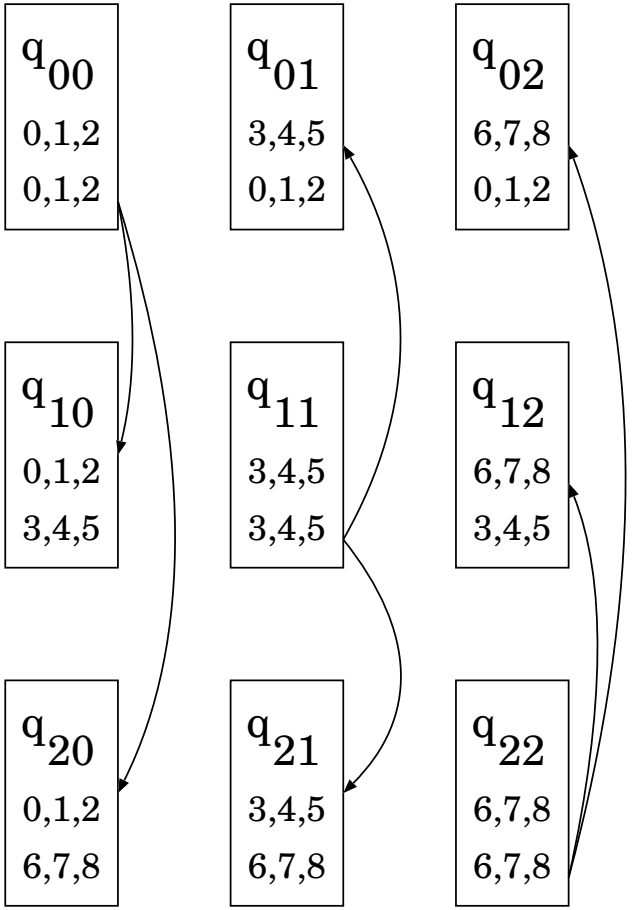
- 分割した相互作用行列にプロセッサを対応させる。
- トポロジ：2D トーラスなり「ハイパークロスバー」なり、、、

計算手順

Step 1



Step 2



横に総和、縦に放送

使えるプロセッサ数

- 効率が $50\ \%$ になるプロセッサ数

$$p_{\mathrm{half,2Dbcast}} \sim (NC_f/2C_c)^2 \tag{3}$$

(通信のスタートアップが無視できる場合) ここから先でも速度は向上する（プロセッサ数の平方根に比例）

ツリー法のベクトル化、並列化

並列化の技法はツリー法も FMM も同じ。

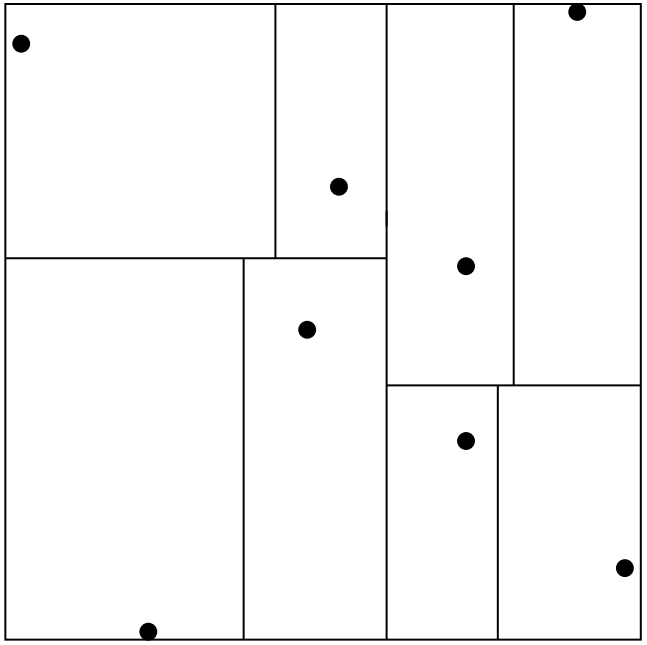
- **共有メモリの機械では「容易」（アルゴリズムの並列度は実際上 N）**
  - **Splash2 ベンチマークとかで効率がでないのはツリー生成のアルゴリズムが無能なせい**
- **メッセージパッシングの機械：効率の良い空間分割が必要。Salmon & Warren が様々な方法をテストした。プログラムは面倒だが、作ってしまえば効率はよい。**

従来の並列化アルゴリズム

実用になっている並列化法は以下の 2 つ。どちらももと Caltech Hypercube のグループの Salmon と Warren によるもの

- Orthogonal Recursive Bysection (ORB)
- Hashed Oct Tree (HOT)

ORB

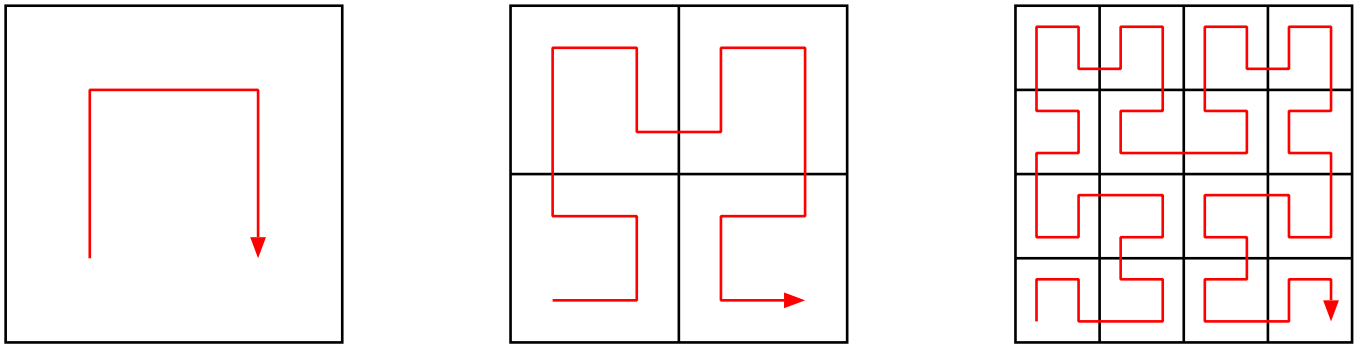


- まず、粒子が半分ずつにわかれるように x 軸に垂直な平面で切る
- 切ったそれぞれについて、また半分になるように y 軸に垂直な平面で切る 3 次元なら次に z、2 次元なら次は x 軸になる

というのを、プロセッサ数になるまで繰り返す

HOT

ペアノ曲線：N 次元空間を 1 次元にマッピング



- **粒子をペアノ曲線上の順番で並べる**
- **各プロセッサに連続した粒子を割り当てる**

HOT の例

ただしこれはペアノ曲線ではなくて Morton Ordering

別のアプローチ — 計算機を作る

速い計算機を買ってきて動かすというのもなかなか大変である。

- 10 年もたつと計算機アーキテクチャが変わる (かもしれない) ので昔頑張って作ったプログラムが水の泡になる (かもしれない)
- 最近考えないといけないことが増えた
  - 分散メモリでの並列化
  - キャッシュの有効利用による高速化
  - その他なんだか分からないテクニック

もうちょっと違う人生はないか?

一つの考え方: 計算機を自分で作る

人が作った計算機を苦労して使おうと思うから面倒。
ハードウェアから好きなように作ればむしろ簡単にならないか?

なぜ専用計算機を考えるのか

基本的には、汎用計算機に比べて、「速く、安く」できる（かもしれない）から。

なぜ「速く、安い」か？

- 問題自体の特性
- 技術的な要因
- 歴史的、経済的な要因

問題自体の特性

粒子系シミュレーションの特徴:一つの粒子が多数の粒子と相互作用する

- 計算量が多い（メモリ必要量に比べて）
- 計算が比較的単純な繰り返しである
- 通信パターンが規則的である

（独立時間刻み、ツリー法では細かい考慮が必要にはなる）

対象システムの分類

連続系（流体等）：規則的、近傍通信、計算量小

粒子系：規則的（ $N \times N$ 通信）、計算量大

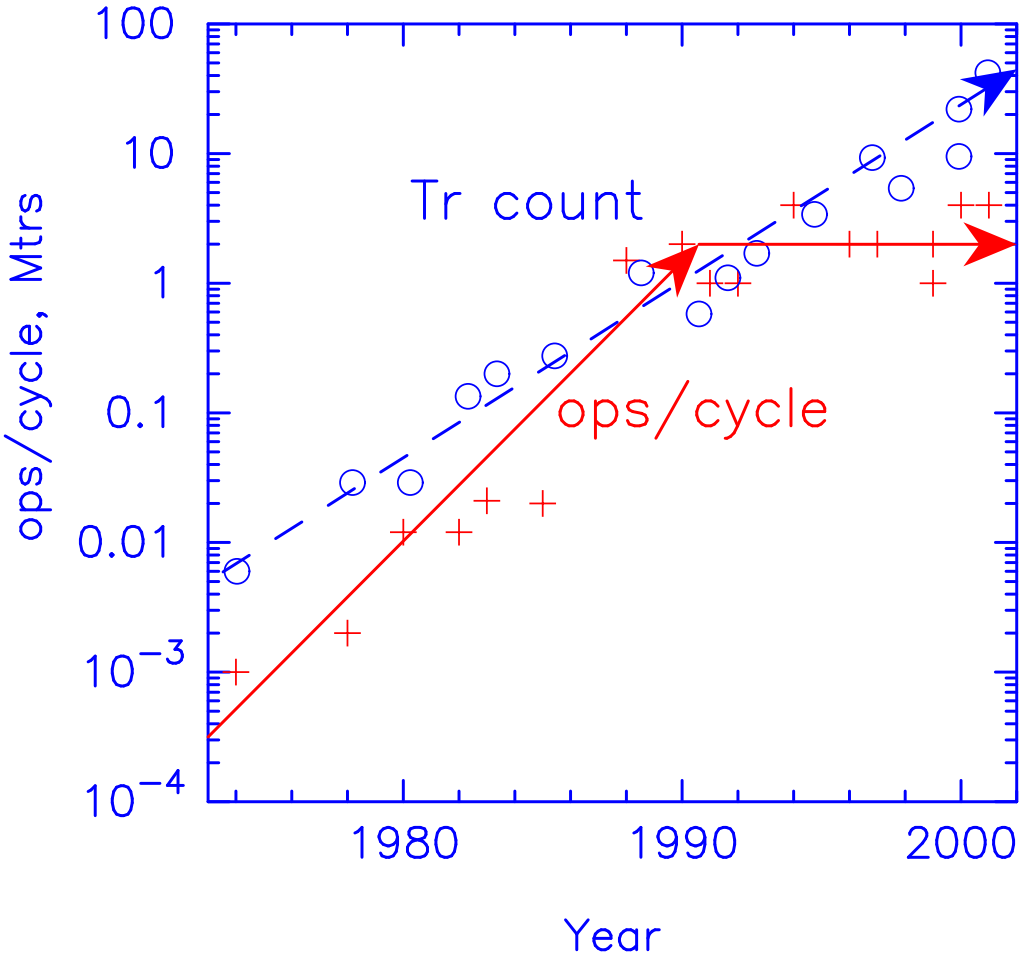
その他（離散不規則系）：回路シミュレーション等？

規則的で計算量が大きい $=$ 専用ハードウェアに適している

技術的な要因

- 半導体製造技術の進歩 $=$ 多数の演算回路を集積する大規模回路が実現可能
- 汎用計算機の設計技術の限界 $(?)$ $=$ トランジスタ利用効率の急速な低下

マイクロプロセッサの「進化」

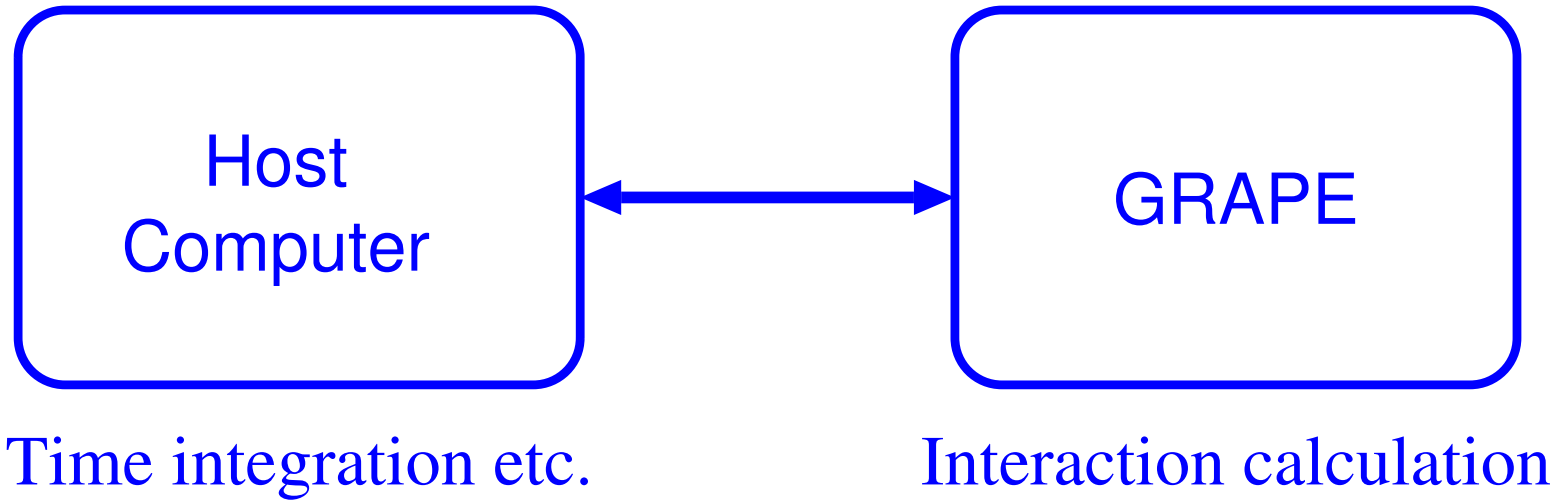


代表的なマイクロプロセッサの、サイクル当たりの浮動小数点演算数

1 を超えてから、ほとんど止まっている ＝ 汎用の並列処理は難しい

ここがつけめ。

GRAPE の基本的考え



専用ハード: 相互作用の計算

汎用ホスト: 他のすべての計算

専用ハードウェア

- 相互作用計算のための専用パイプラインプロセッサ
  - 多数の演算器を集積可能
  - すべての演算器が常時並列動作

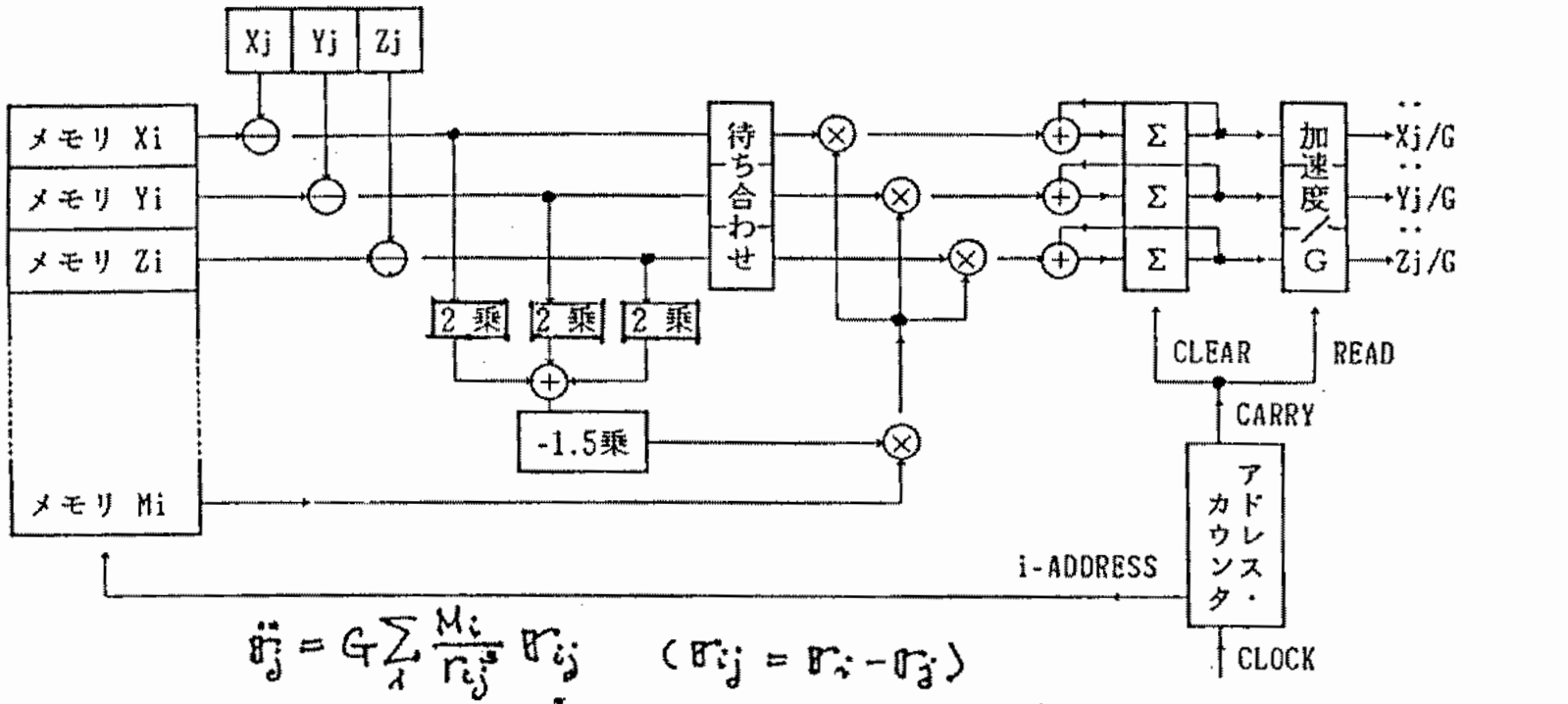
  → 非常に高い性能

重要な条件: Memory wall **の回避**

汎用ホスト計算機

- 高レベル言語 (Fortran, C, C++...)
- すでにあるプログラムが少しの変更で使える。
- 独立時間刻み、ツリー法等も使える

GRAPE パイプライン



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

+, −, ×, 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 1988）

Memory wall の回避

あるいは通信量の削減

ホスト — GRAPE 間: N 個の粒子、 N^2 の計算

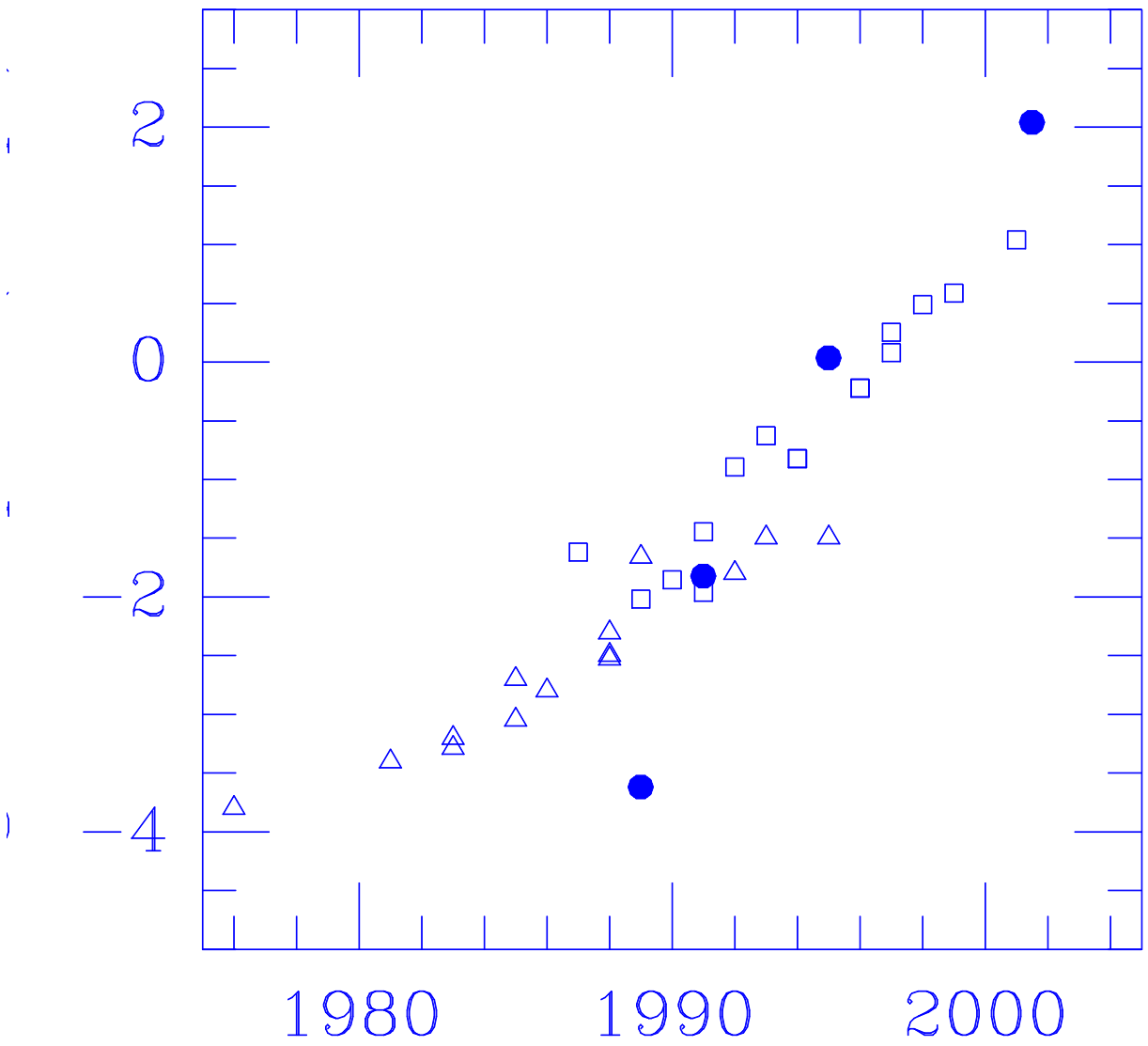
ボード/チップレベル:

- **i 並列 (複数のパイプラインがメモリ 1 つから同じデータを受け取って、別の粒子への力を計算)**
- **仮想マルチパイプライン**

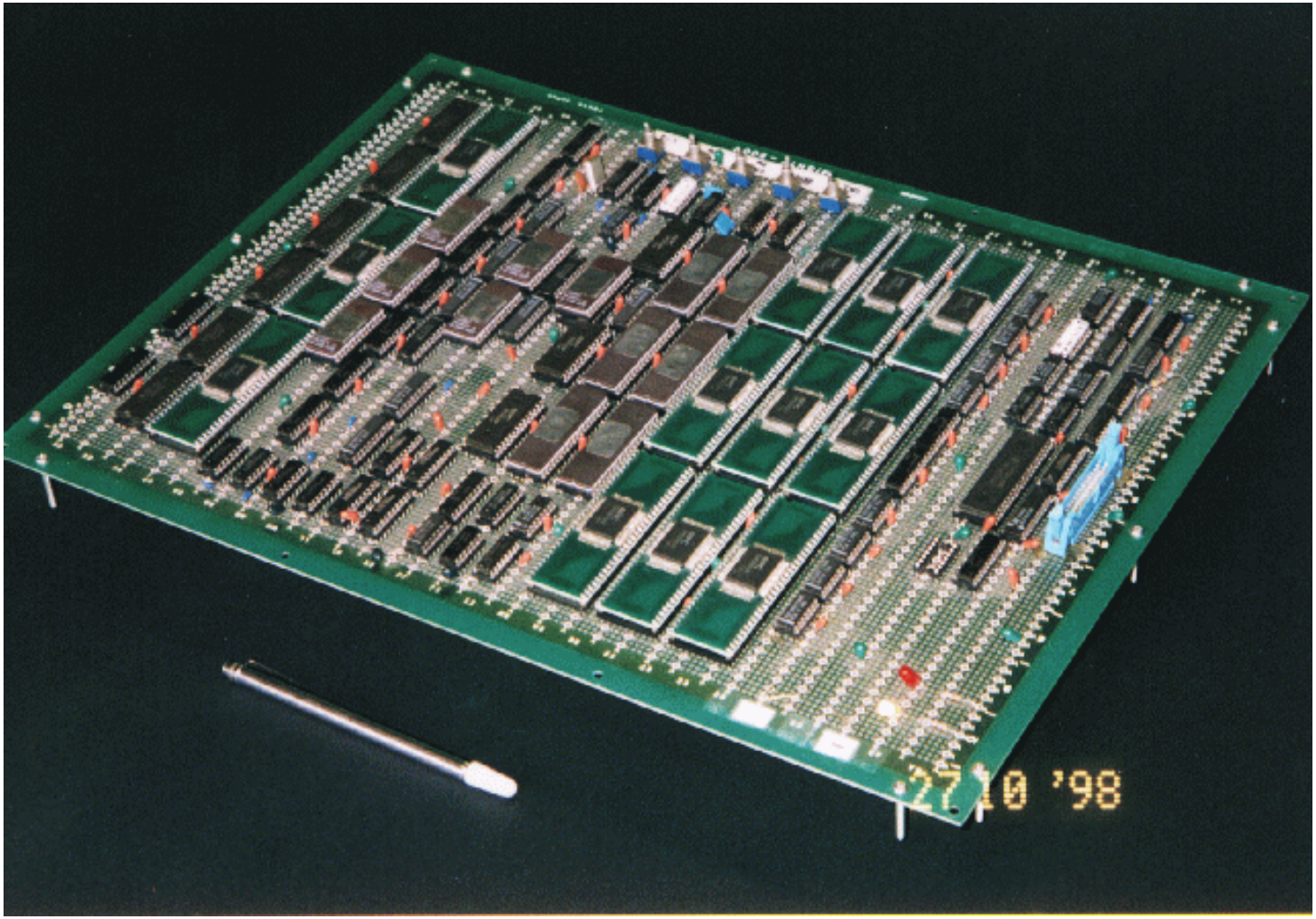
GRAPE アーキテクチャの発展

1989 GRAPE-1 低精度、EPROM で演算
1990 GRAPE-2 高精度、浮動小数点演算 LSI
1991 GRAPE-3 低精度、カスタム LSI
1995 GRAPE-4 高精度、カスタム LSI 、超並列
1998 GRAPE-5 低精度、複数パイプラインを集積
2001 GRAPE-6 高精度、複数パイプラインを集積

計算速度の発展



GRAPE-1



GRAPE-1 の中身

「初めてのデジタル回路」

当初の目標

- とりあえず演算パイプラインのようなものを作る
- ホスト計算機につないで動かす
- 意味がある計算ができるかどうかはあまり気にしない

最初の構成

- 演算は ROM テーブルルックアップ。データは 8 ビット
- 通信は GPIB

計算/通信比: 粒子数1万くらいで性能がでればいいことにすると
動作クロック 5 MHz くらい = 1粒子のデータのやりとりに 1ms くらいは使っていい
データ量が数十バイトくらい → データ転送速度は100KB/s くらい欲しい

シリアルでは不足
SCSI は難しくて良く分からなかった

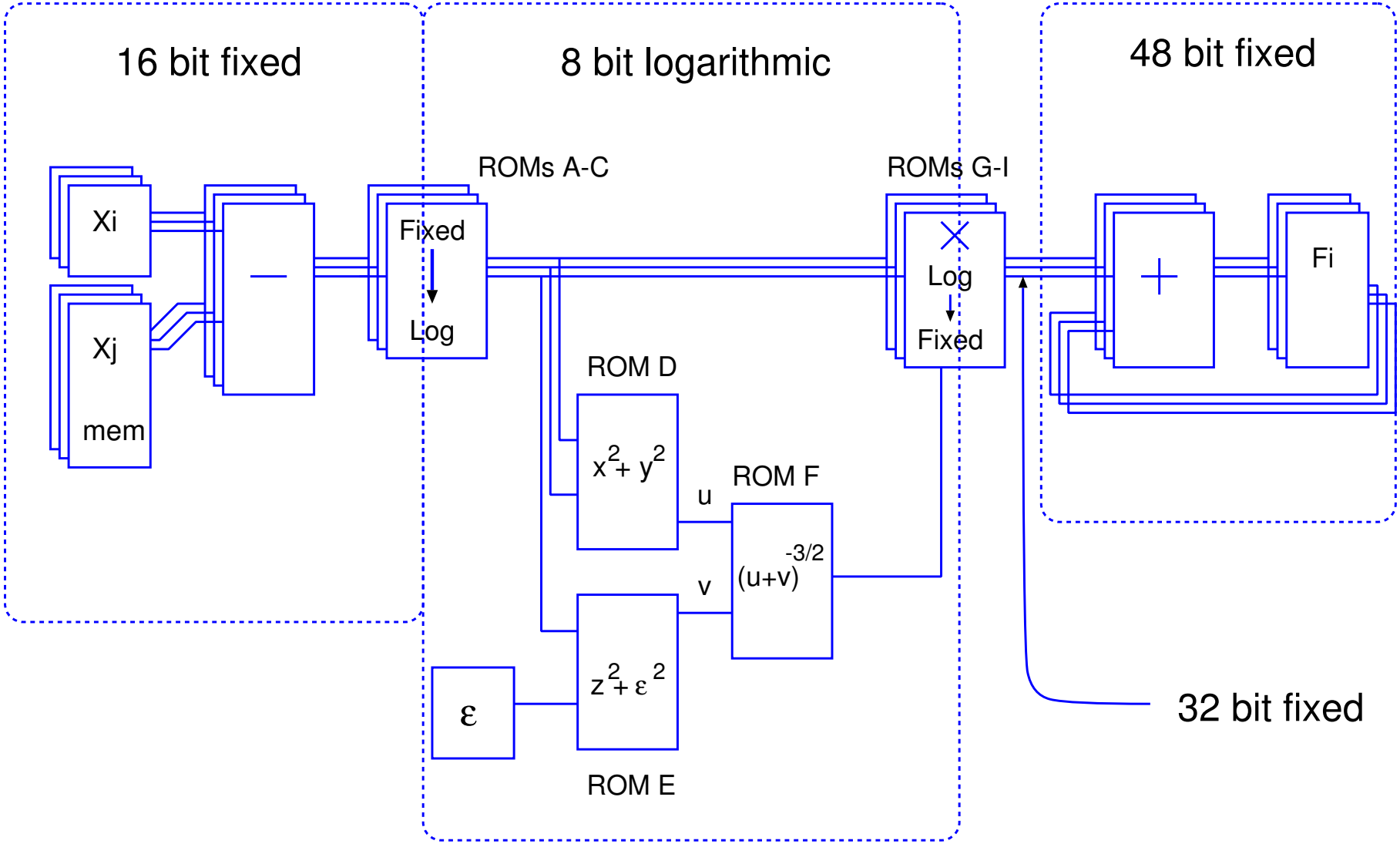
実用になるものへ

途中の計算 (2 粒子間の力の相対精度) は低くても、

- **最初の座標の引き算**
- **最後の足し上げ**

だけを (そこそこ) 高い精度でやれば、問題によっては十分使える (ツリーコードに比べれば高い精度になる)
ということに気が付いたので、そっちに変更
最初の座標の引き算: 16bit **固定小数点、最後の足し上げ:**48bit **固定小数点**
とする。

GRAPE-1 パイプライン



開発中のトラブル

ハードは割合順調に出来たらしい (実は伊藤君が担当だったので良く知らない)

できてから、性能がでない、、、

当初: PC-98 ホスト、どこかの GP-IB ボード (TI の GP-IB インターフェースチップがのってるだけのもの)。
通信速度は問題なし

ホストが遅い、メモリものらない、、、、というので何故か GP-IB インターフェースカードがあった Sony NEWS-830 につなぐと、、、、

通信オーバーヘッドというもの

NEWS につなぐと全くまともな性能が出ない。

原因: GPIB を使った通信は read/write システムコールを使う。で、これが 1 回呼ぶと 1ms くらいかかる。

GRAPE-1: 1 粒子もらって、それへの力を計算して、結局返す。

NEWS 830 は I/O プロセッサ付き: オーバーヘッドは一層大きい

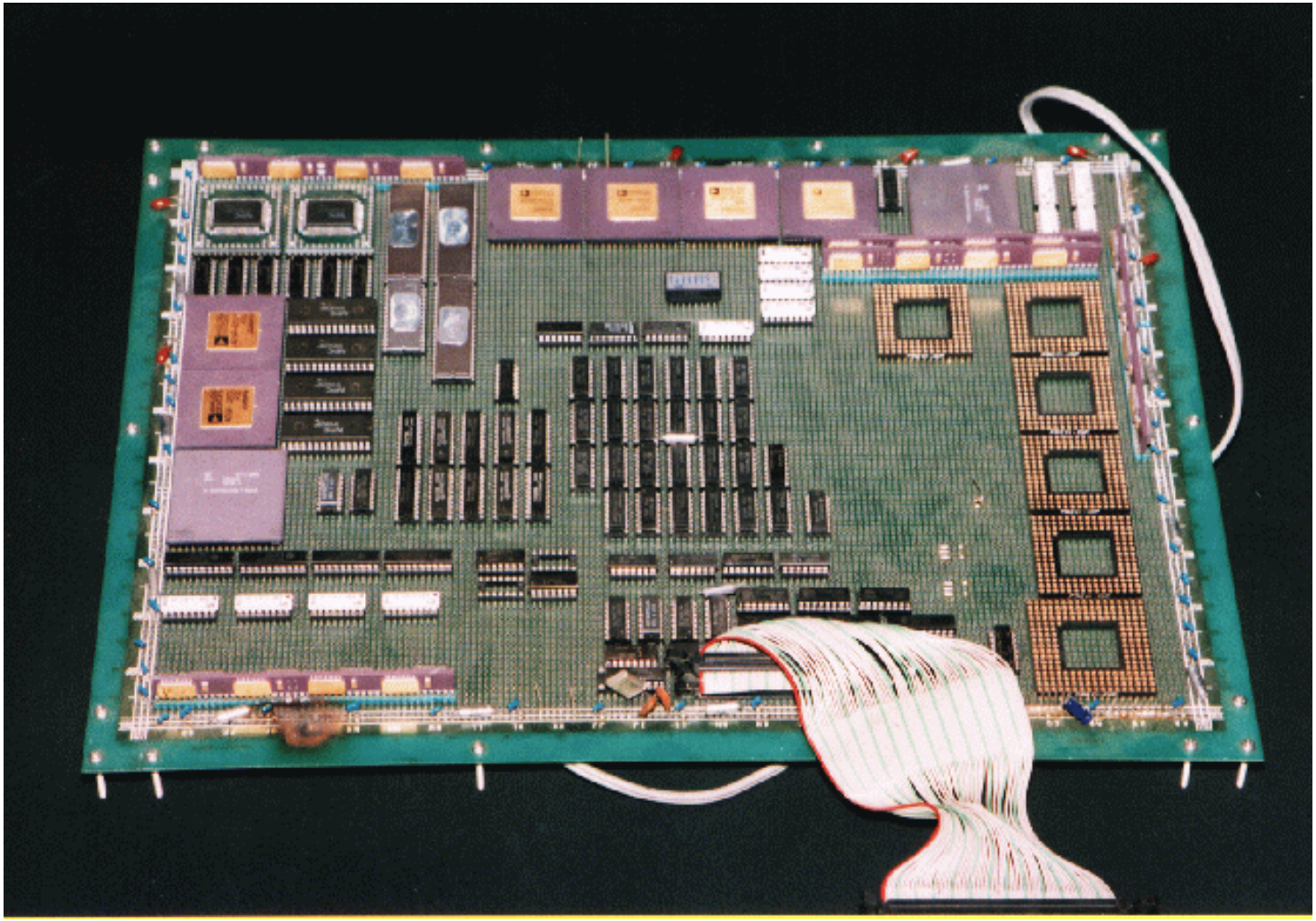
最初の対応: PC-98 をバッファリングに使う。

次の対応: I/O プロセッサのない Pop NEWS にして、GP-IB コントローラのレジスタをユーザープロセスがいじりまわす。これで 100KB/s くらいでるようになった。

教訓

- 通信ソフトウェアは厄介である
- 教科書的に「良い」方法が良い結果になるとは限らない
- 結果が良ければ「邪道な」ことでもなんでもかまわない

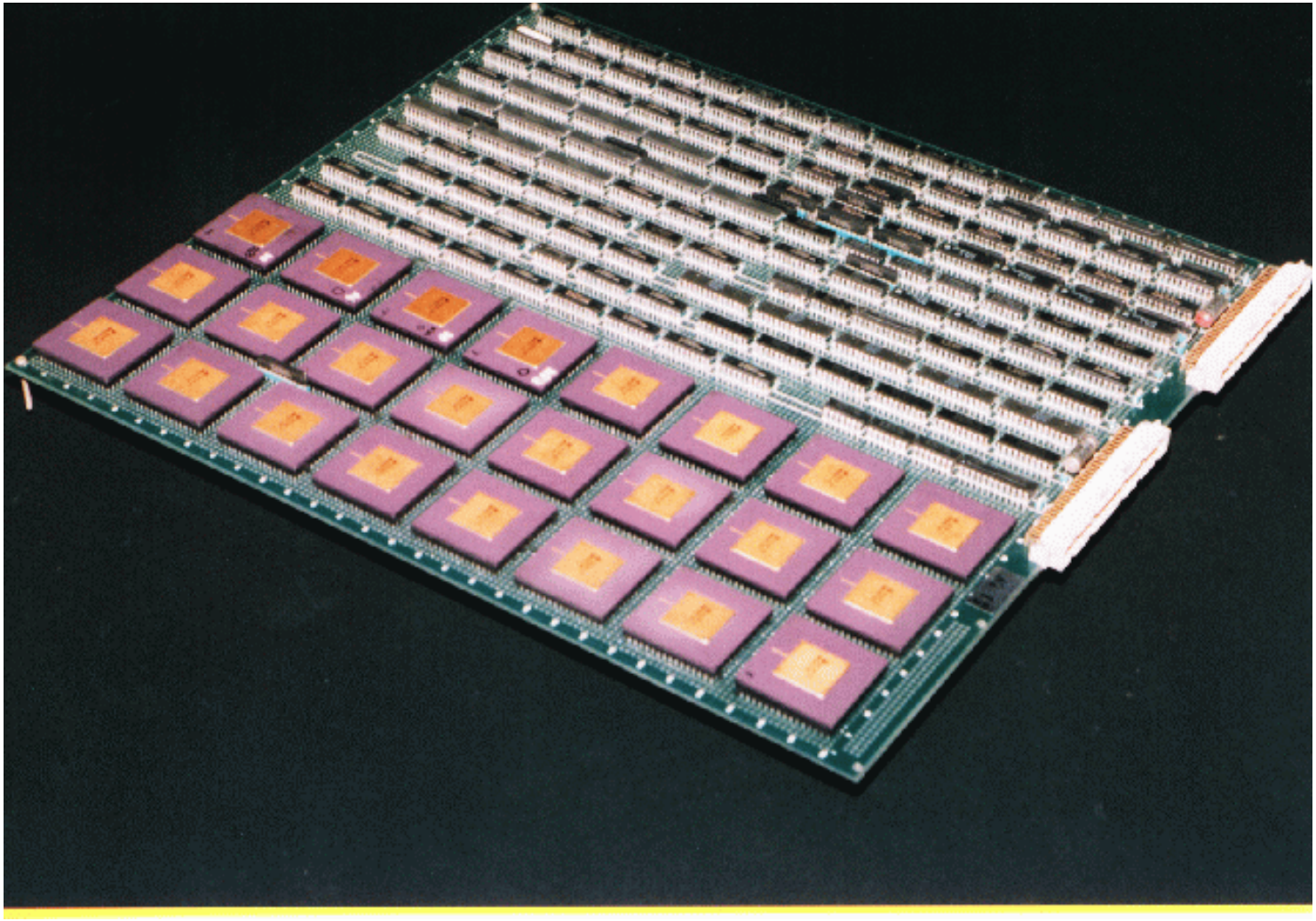
GRAPE-2



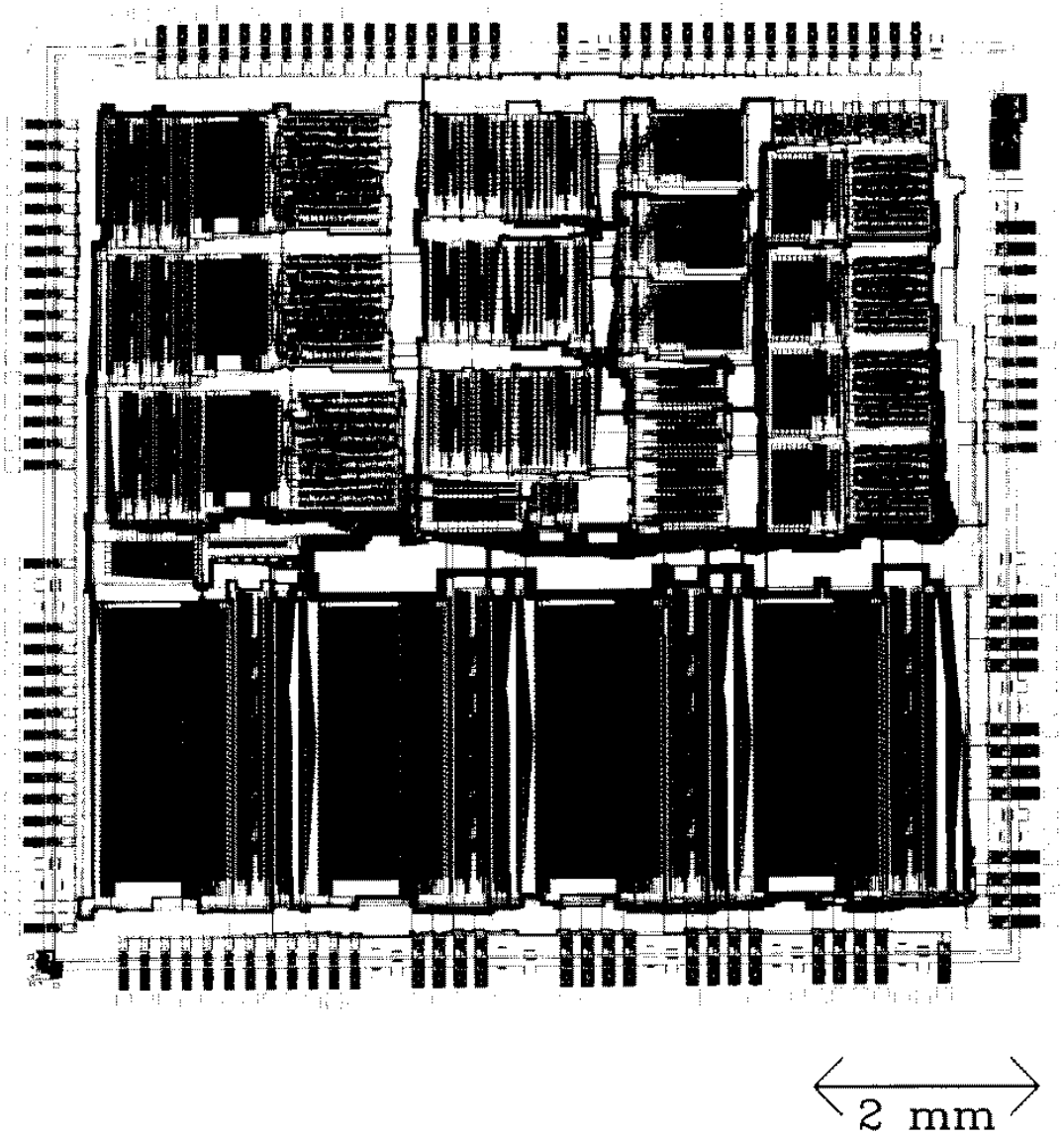
GRAPE-2 のまとめ

- 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)
- 通信は VME バス
- ホストは VME バスに PIO アクセス。 DMA は使わない (GRAPE はマスターにならない)

GRAPE-3



GRAPE-3 チップ



GRAPE-3 チップ

- 仕様決定、シミュレータ (C で記述) は牧野がやった
- 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

チップを作るのはどれくらい心臓に悪いか?ということ

2回チップを作る予算があったことはない=一発勝負

普通の責任分担

- **テストベクタが通らなければ向こうのせい**
- **テストベクタが通ればこっちのせい**

「理論上は」「完璧な」テストベクタを作ればこちらのせいである問題は起こりえない。

実際はそうはいかないけど、、、

普通のチップとの違い

GRAPE の場合、マイクロプロセッサとかに比べるとはるかに楽

「内部状態」というものがほとんどない。計算中/停止の違いくらい。これはアキュムレータの制御だけ。

演算回路の仕様自体が問題。小規模なシミュレーションでは動いても、実際にハードウェアができて初めて可能になる大規模な計算で問題なく動くかどうかはわからない。

その他、予想もしない問題は起きる。

GRAPE のチップはどんなふうに作るか

1. まずパイプラインの仕様を決める。
   - 何を計算するか
   - 各演算器の演算精度
   - 目標性能
   - メモリインターフェース、ホストインターフェースの物理仕様 (ビット幅、速度)
2. パイプラインの機能検証: 動作 (計算結果) をソフトウェアでシミュレーションするプログラムを書く。その動作が理屈に合っていて、実際の多体計算プログラムにいれても動くことを確認する。
3. あとは普通に RTL 設計

CPU の設計との違い: 物理設計の検証は動作記述と合わせる。サイクルレベルの設計は HDL 記述しかない。

GRAPE-4

GRAPE-4 における通信

演算性能:GRAPE-3 の 100 倍

通信:単一ホストでできるベストに近いところで済ませられればそうしたい。

とすると 100 MB/s くらい。

1 粒子データ 200 バイトくらい

とすれば、 10^5 粒子くらいでまあまあの性能がでる計算になる。

まあ、賞でも取ろうという時以外は機械を分けて使うので、これくらいの見積りで十分。

100MB/s をどうやって出すか? (1993年に)

- DMA は必須
- ホストは、、、、 DEC Alpha、バスは TurboChannel

オーバーヘッドを避けるためにはDMA のたびにカーネルにいったりしては困る。
というわけで
ユーザープロセスのバッファに使うページの物理アドレスをカーネルの内部関数をなんかして調べて、それをユーザープロセスがインターフェースカードの DMA アドレスレジスタに直接書く。

最近なら？

最近のまともな OS では、カーネルが確保したバッファをユーザー空間にマップしてくれるみたいなので同様のことがもうちょっと安全に実現できる。

GRAPE-6 について

- 設計思想
- プロセッサチップ
- プロセッサボード
- ネットワークボード
- 全体

設計思想

プロジェクト目標 (お金をもらう時の公約)

粒子系専用でいいけど世界一の速度を達成する

「当然」の目標:

サイエンスとして、研究にちゃんと使える機械にする。おもちゃとは違う。

境界条件

- 総予算: 5 億 (参考:地球シミュレータ 400 億、ASCI Q 250 億？)

で、初期目標: 200Tflops (地球シミュレータ 40T)

境界条件

- 総予算: 5 億 (参考:地球シミュレータ 400 億、ASCI Q 250 億？)

で、初期目標: 200Tflops (地球シミュレータ 40T)

「常識」では考えられない

予算獲得時には結構問題

境界条件

- 総予算: 5 億 (参考:地球シミュレータ 400 億、ASCI Q 250 億？)

で、初期目標: 200Tflops (地球シミュレータ 40T)

「常識」では考えられない

実際のところできる話かどうか？

だいたいできたから結果論だけど

GRAPE-6 の性能予測

演算性能の予測: 基本的は GRAPE-4 から外挿しただけ

	G4	G6 (予測)	G6 (実際)
設計ルール	1μm	0.25μm	0.25μm
動作クロック	32 MHz	125 MHz	90MHz
パイプライン本数	1/3	5-10	6
演算性能	600Mflops	36-72 Gflops	31 Gflops
開発費	2500万	7000万くらい	1億以上
チップ単価	8000円	1-2万	3万

楽観的には 3000 チップで 200 Tflops。チップ以外はボード当り 100 万くらい (これは大体合ってた)

チップについては結構予測が甘かった、、、 200Tflops が 64 Tflops に落ちた理由

全体アーキテクチャの制限

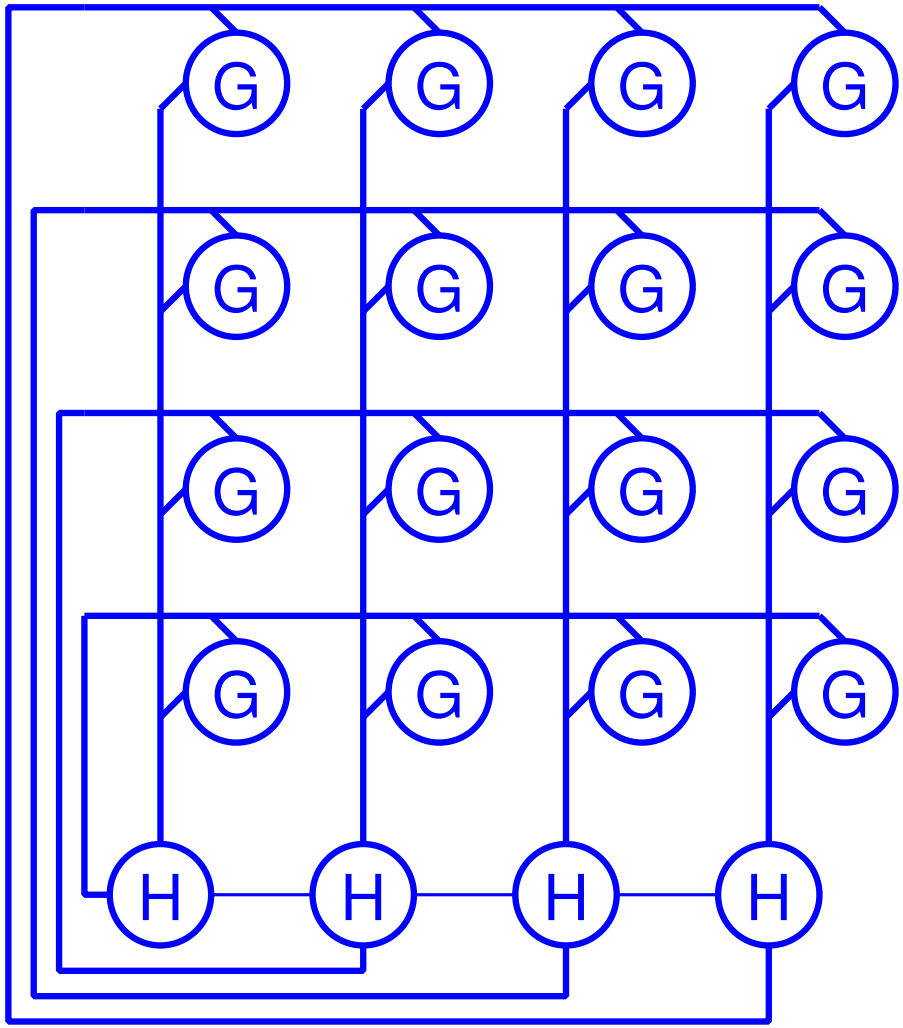
もっとも厄介な制約: ホスト計算機との通信速度

GRAPE-4 の実効速度: 50 MB/s 程度

理想的には 100 倍、でなくても 20 倍くらいは欲しい。

- **並列ホストは必須**
- **単純に並列にホストと GRAPE をつなぐのでは駄目**

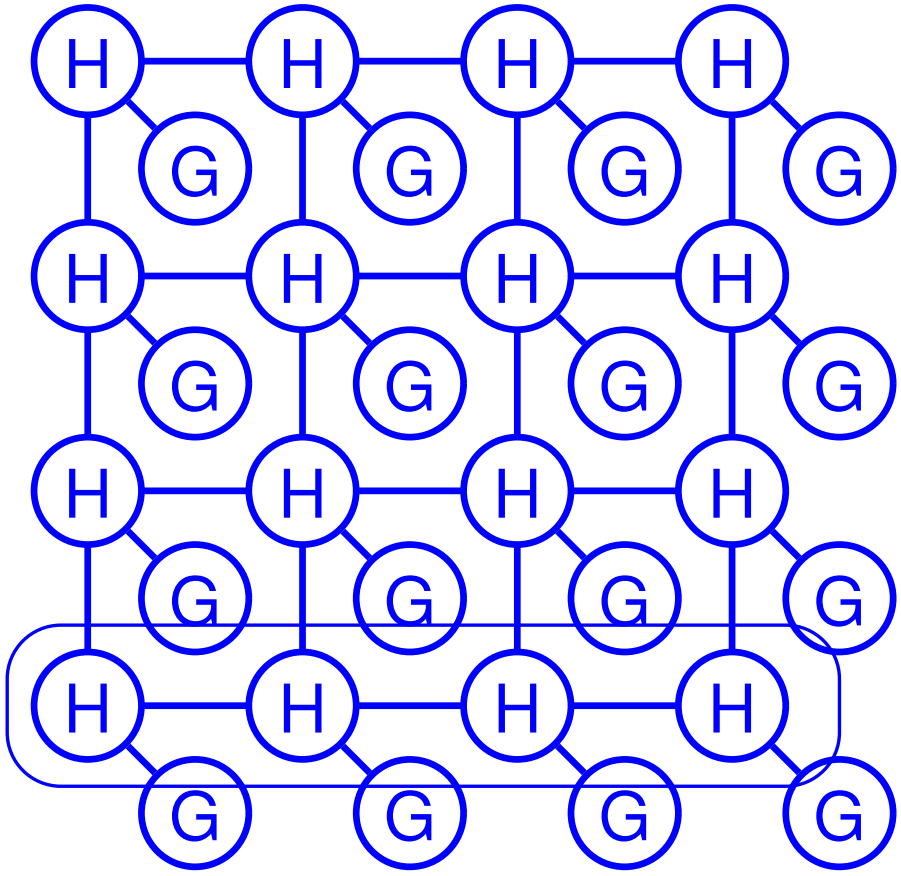
GRAPE-6 の2次元ネットワーク



GRAPE 側で2次元アルゴリズムを実現

実際のネットワークはもうちょっとややこしい

別の解



今 (2002 年以降) ならこれ。
1996 年の時点では採用は現実的ではなかった (と思う)

- 速いネットワークは高価だった (GbE が本当に安くなったのは 2003 年)
- 速いホスト計算機も高価だった (Alpha, HP-PA は x86 の数倍の速度があった)

チップアーキテクチャの制限

GRAPE-4 ボード 1 枚分くらいがチップに入る。

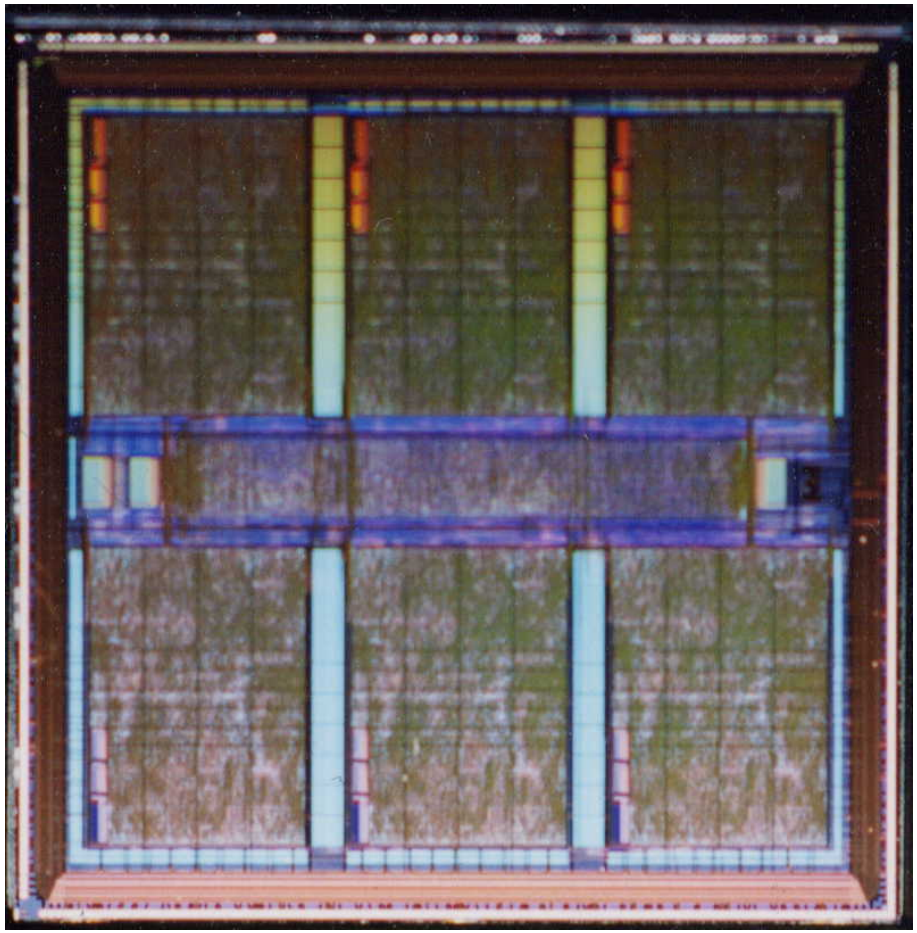
独立時間刻みを使うことを考えると、メモリを共有できるチップ数に制限がつく。

チップ毎にメモリを持つことで、この問題を回避。

複数のチップが同じ粒子への、別の粒子からの力を計算、ボード上で合計してホストに戻す。

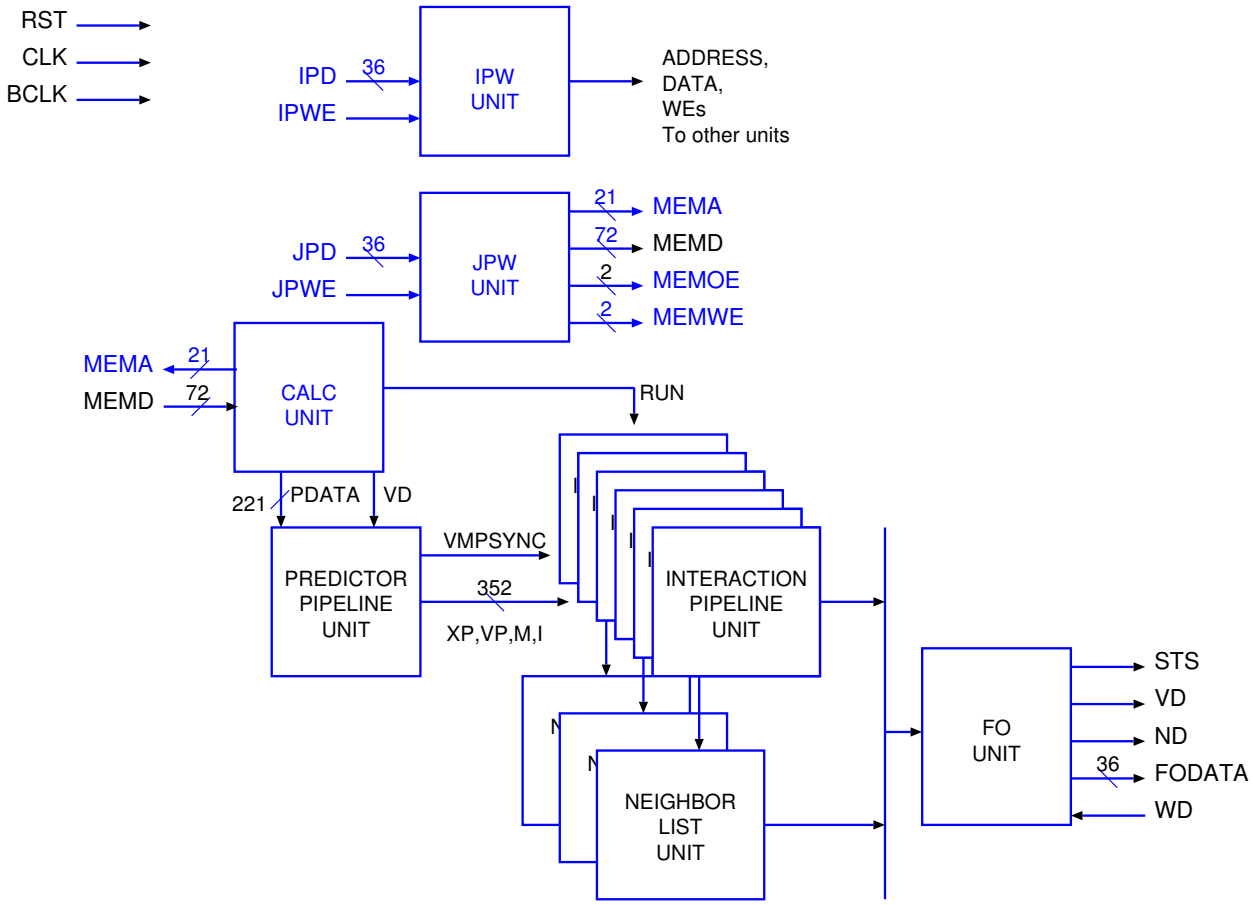
指数を固定して加算することで、結果が加算順序に依存しないようにする。

パイプライン チップ



- 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

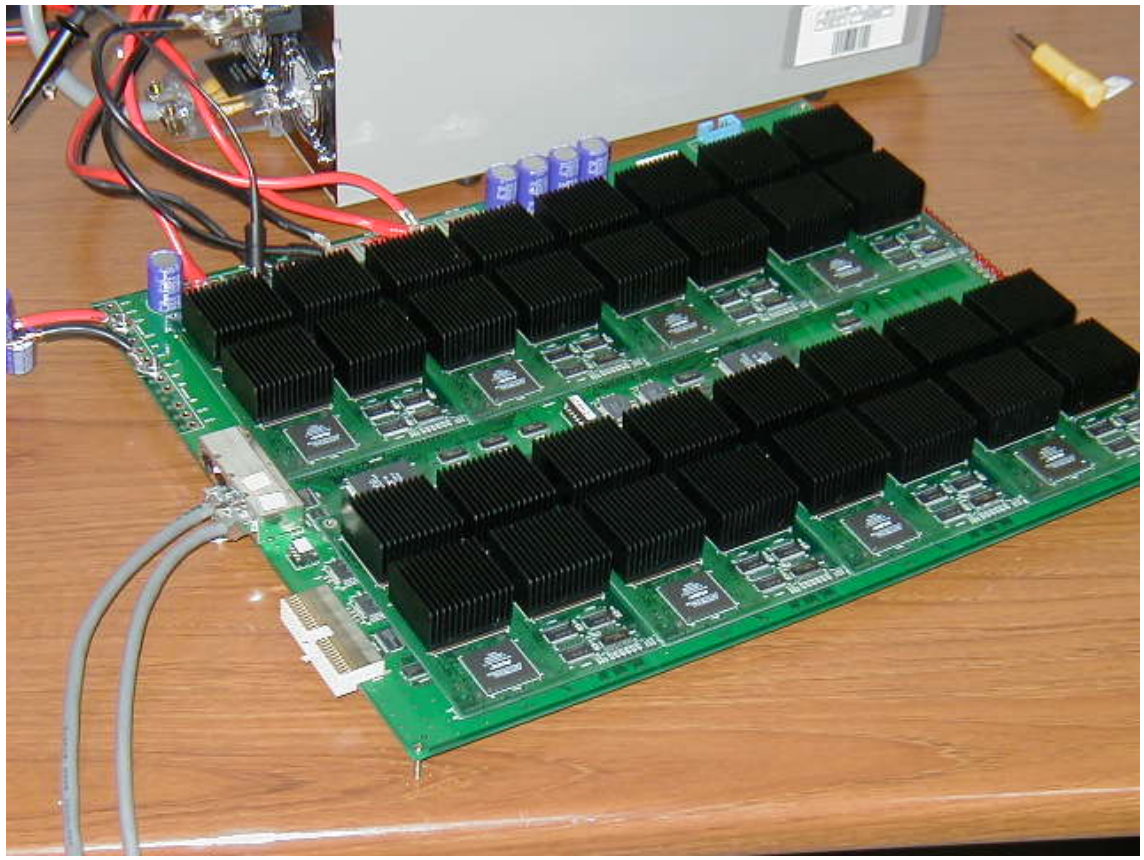
パイプライン LSI詳細



GRAPE-4 プロセッサボードの全機能を集積

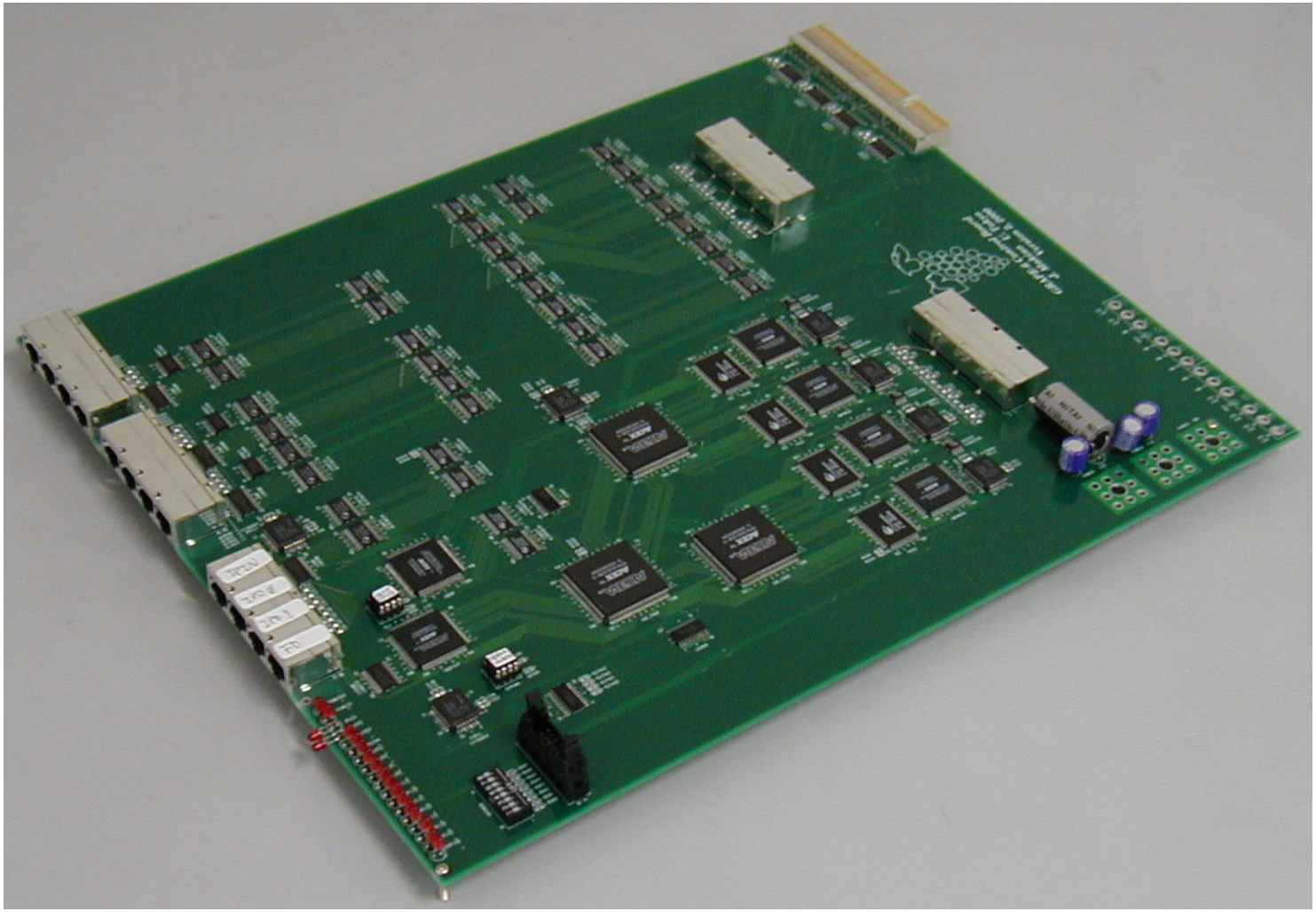
- ホストIF
- メモリIF
- パイプライン本体
- 制御回路

GRAPE-6 processor board

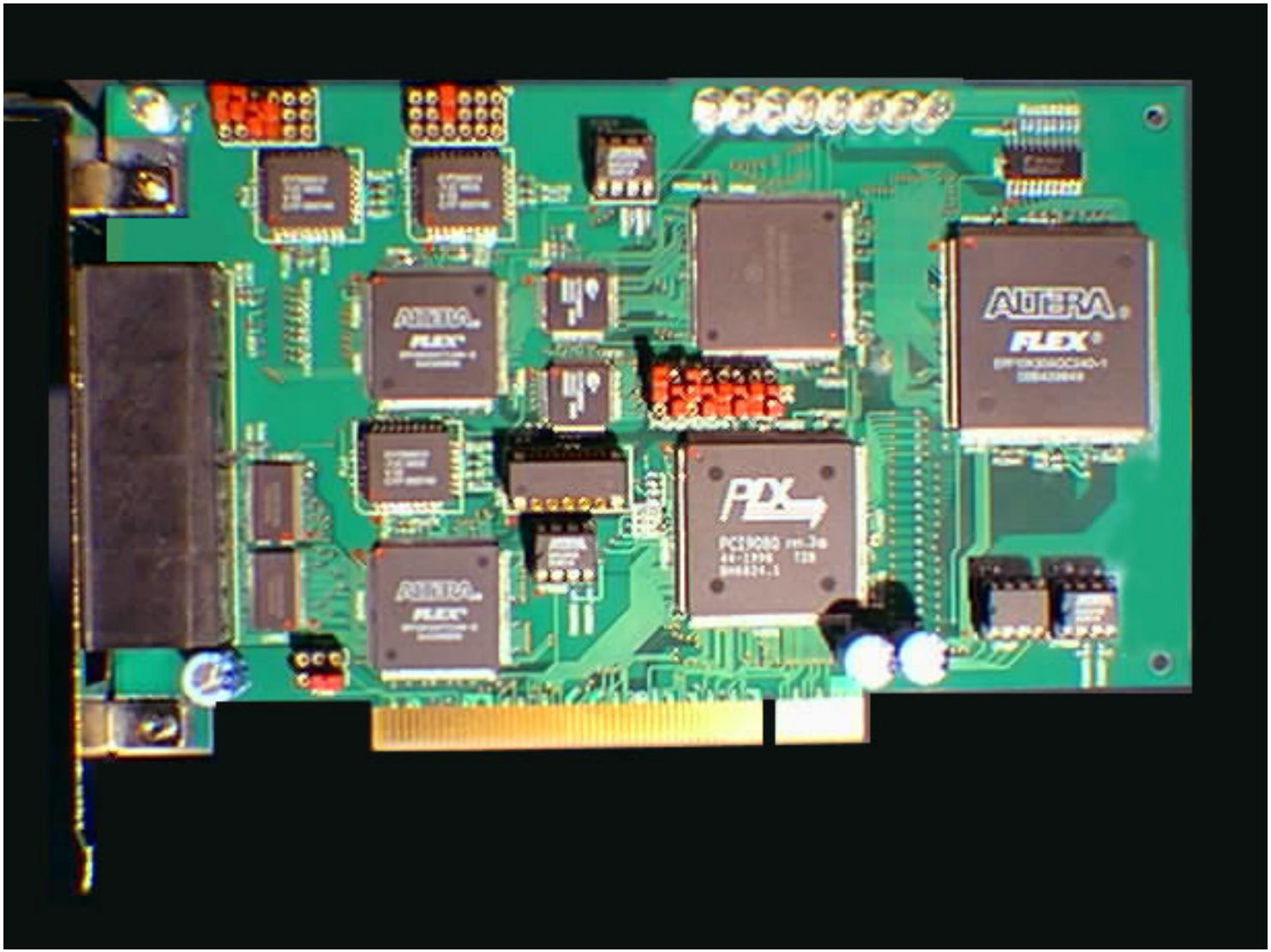


- ボード 1 枚に 32 チップ
- セミシリアル (LVDS) インターフェース (350MHz clock, 4 wires)

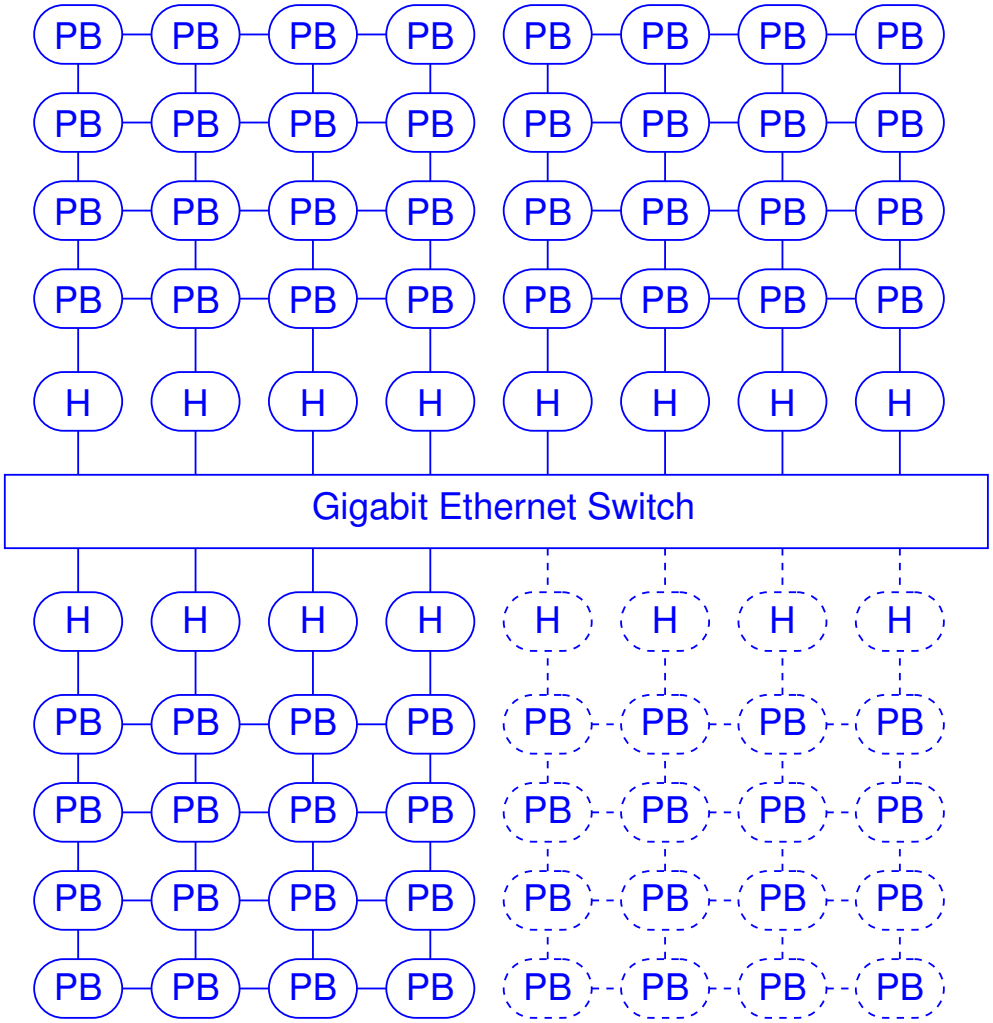
GRAPE-6 network board



GRAPE-6 host interface board



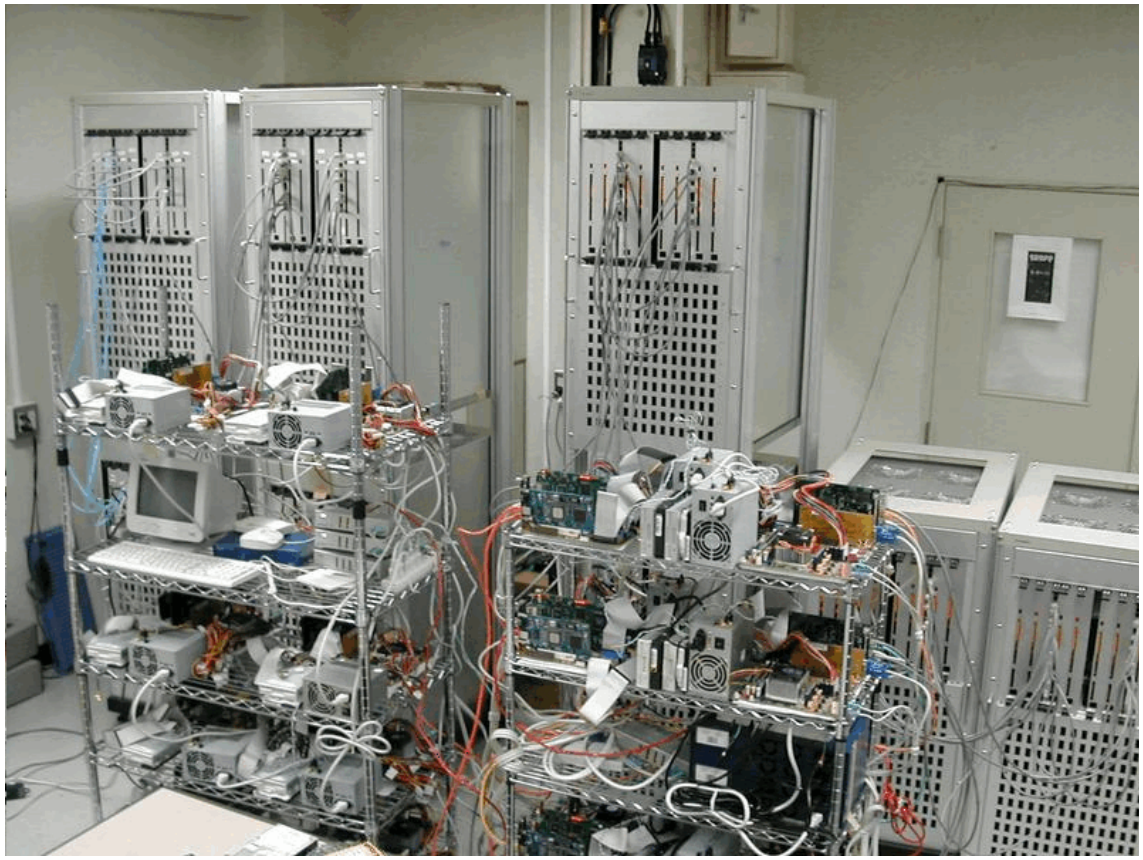
The full 64 Tflops GRAPE-6 system



- 4-host, 16-board "block" with dedicated network
- 4 (currently 3) "blocks" connected through GbE network

Combination of host network solution and dedicated network solution.

The 64-Tflops GRAPE-6 system



Present 64-Tflops system.

4 blocks with 16 host computers.

個人的な感想

GRAPE のようなものを「うまく作る」のは結構大変である。

- 対象となる問題
- アルゴリズム・要求精度
- 計算機アーキテクチャ
- 論理設計
- 実装
- OS、デバイスドライバ等のソフトウェア

の全般についてまあまあの理解が統合されていないと全体設計が出来ない。

普通の計算機を作るのに比べると

普通の計算機では

- 対象となる問題、アルゴリズム、要求精度

「必要ない」（本当？）

- それ以外

「まあまあ」ではまあまあの計算機しか出来ない。

他の誰かと同じようなことをすれば済むという面もある（1番になれるか？という問題はある）

専用計算機は難しい？

「成功」といえるようなものはそんなに多くない。

多体系に話を限るとアプローチは2つ。どちらも結構いろいろある。

専用パイプライン

- DMDP (Delft)
- FASTRUN
- GRAPE
- MD-Engine
- MDM

並列計算機

- Digital Orrery
- Transputer-based projects...
- HaMM
- **他にも沢山**

あんまりうまくいかないのはなぜ？

どれが成功でどれが失敗かというのはさておき、、、
基本的な理由は 2 つ。

1. 出来たものが実は使えなかった。
2. 出来た時には速度が汎用計算機より速くなかった。

理由 (1) は専用計算機に固有だが、例は少ない（公表されていないのがあるのかもしれないが）
理由 (2) は普通の計算機と同じ。開発に時間がかかるとムーアの法則の分相対的に遅くなる。

開発期間の問題

大抵の人は（私を含めて）見積もりが甘いというのは確か。

本質的な問題：
専用計算機の場合、開発期間が多少（1-2年）延びたくらいで意味がなくなるようなプロジェクトは、そもそもあまり意味がない（出来てからのハードウェアの寿命がどうせ短い）。
大雑把には、計画時点で

- 価格性能比が 1000倍 — 問題なし
- 価格性能比が 100倍 — ちょっと苦しい
- 価格性能比が 10倍 — やめたほうがいい

開発に5年、マシンの稼働期間が5年とするとこれくらい。

価格性能比 1000 倍は可能か？

あくまでも「計画時点」。とはいえ、、、

半導体技術は 1-2 年先を見込める: 1.5-2 倍得

大量生産されるマイクロプロセッサほどクロックはあげられない: 3-5 倍損

結局、

- 演算数が同じならチップ当りの値段を 1/1000 にする
- 値段が同じなら演算数を 1000 倍にする

必要がある。

チップ当りの値段を下げる

ある程度は可能

PC クラスタに使うような計算機のノード単価: 50-100 万

ASIC の量産単価 1-10 万

ASIC の値段が半分以上を占めるようにシステムを設計できれば 5-50 倍は下がる。(開発費があるので実際には 10 倍以上は難しい)

メモリ・ネットワーク周りに手を抜けないと値段は下がらない。

チップ当りの演算数を増やす

マイクロプロセッサはクロックあたり 1 演算程度しかしない

$\longrightarrow$ 100 個以上の演算器が 1 チップに入れば OK。(GRAPE-6 の時。今だと 1000 個いれないと駄目)

入れるだけならなんということはない。 10^7 トランジスタもあれば十分。

どうやって使うか

- チップ外への通信速度
- 演算器間の通信速度
- 内部メモリ、レジスタと演算器の間の通信速度

GRAPE の場合

- 実／仮想マルチパイプラインによるチップ外通信速度の削減
- ハードワイヤドパイプラインによる最小コストでの演算器間接続
- 同じくハードワイヤドのため内部メモリ不要

通常 on-chip multiprocessor で起きる問題をほぼ完全に回避している。
基本的にはそういうことが出来る問題だから。
逆にいえば、そういうことが出来る問題（アルゴリズム）でないと専用計算機はうまくいかない。

専用計算機の今後

GRAPE アーキテクチャ：汎用計算機に比べた相対的優位はだんだん大きくなる

ムーアの法則でトランジスタは増える

汎用計算機では増えたトランジスタの全部を演算器には使ってない

GRAPE は全部使える

チップの初期コストが上がるのが問題：予算をどうやって取る？

次期 GRAPE

振興調整費 GRAPE-DR プロジェクト 2004-2008

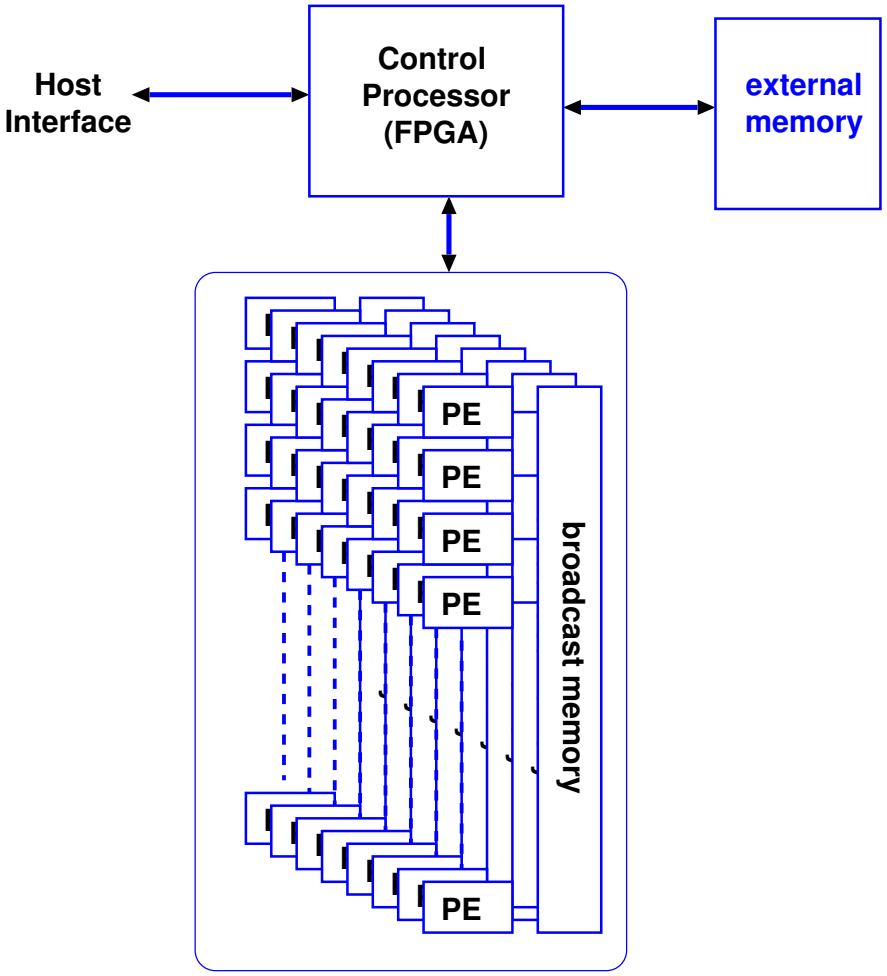
基本的な考え

- **チップに演算器を 1-2000 個くらい入れる**
- **それを (GRAPE が得意なタイプの問題に対しては) ある程度のプログラム可能性をもった形で使う。 GRAPE のようなハードワイヤードなパイプラインにはしない。**

アーキテクチャを変える理由

- 同じようなのを 3 回も作るのは飽きる
- Top 500 に載りたい
- 予算が欲しい
- ハードワイヤードなパイプラインでなくても汎用計算機にかなり余裕で勝てる (まあ、私の見積もりは甘いわけだが)

GRAPE-DR の アーキテクチャ



- 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)
- PE はプログラムによって並列動作する
- 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ユニット (BU) 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

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 として使う時:
単純な方法: 1000 プロセッサが 1000 個の粒子への力を計算。
利点:

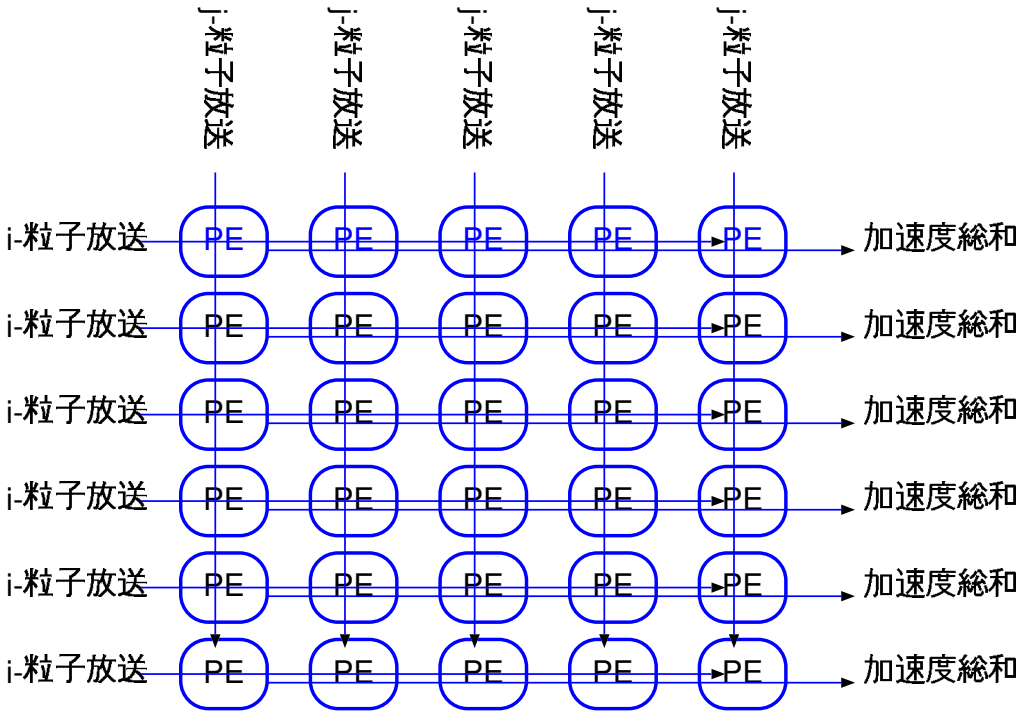
- 単純なネットワークでいい (放送とランダムアクセス)
- メモリもプロセッサあたり少しだけあればいい

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

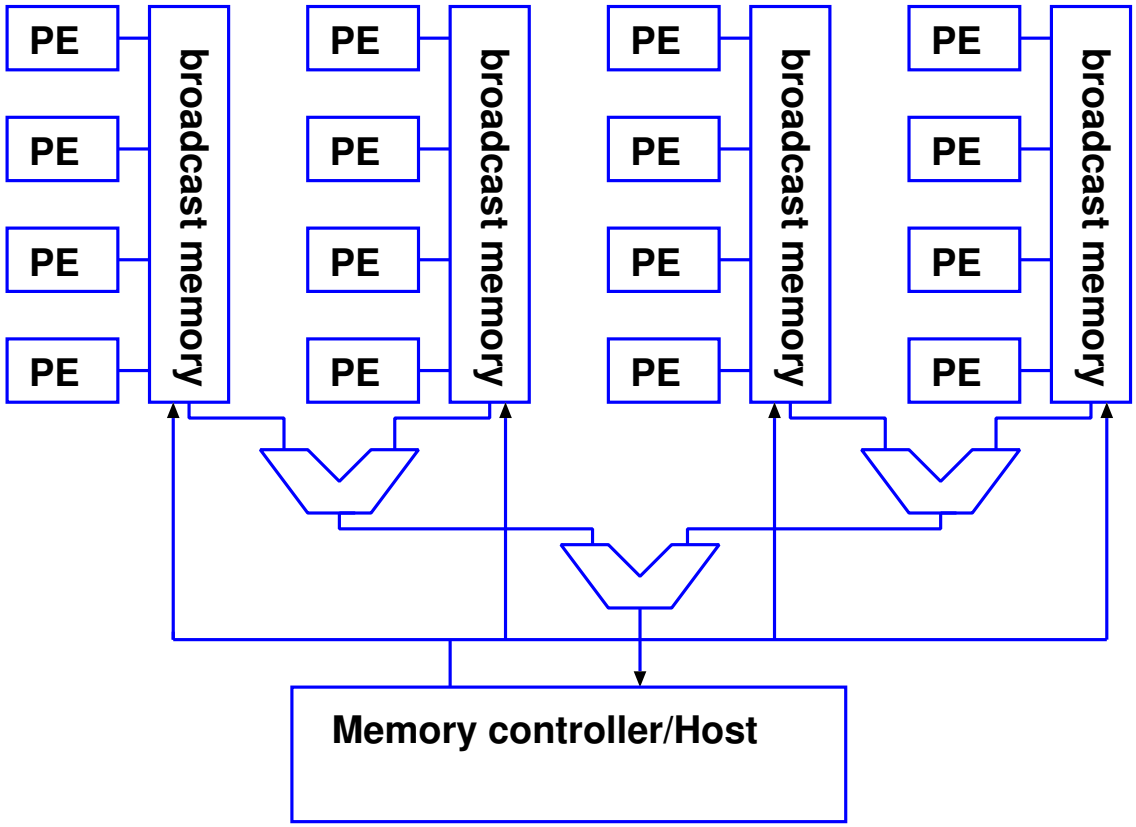
PE は放送メモリとだけ接続

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

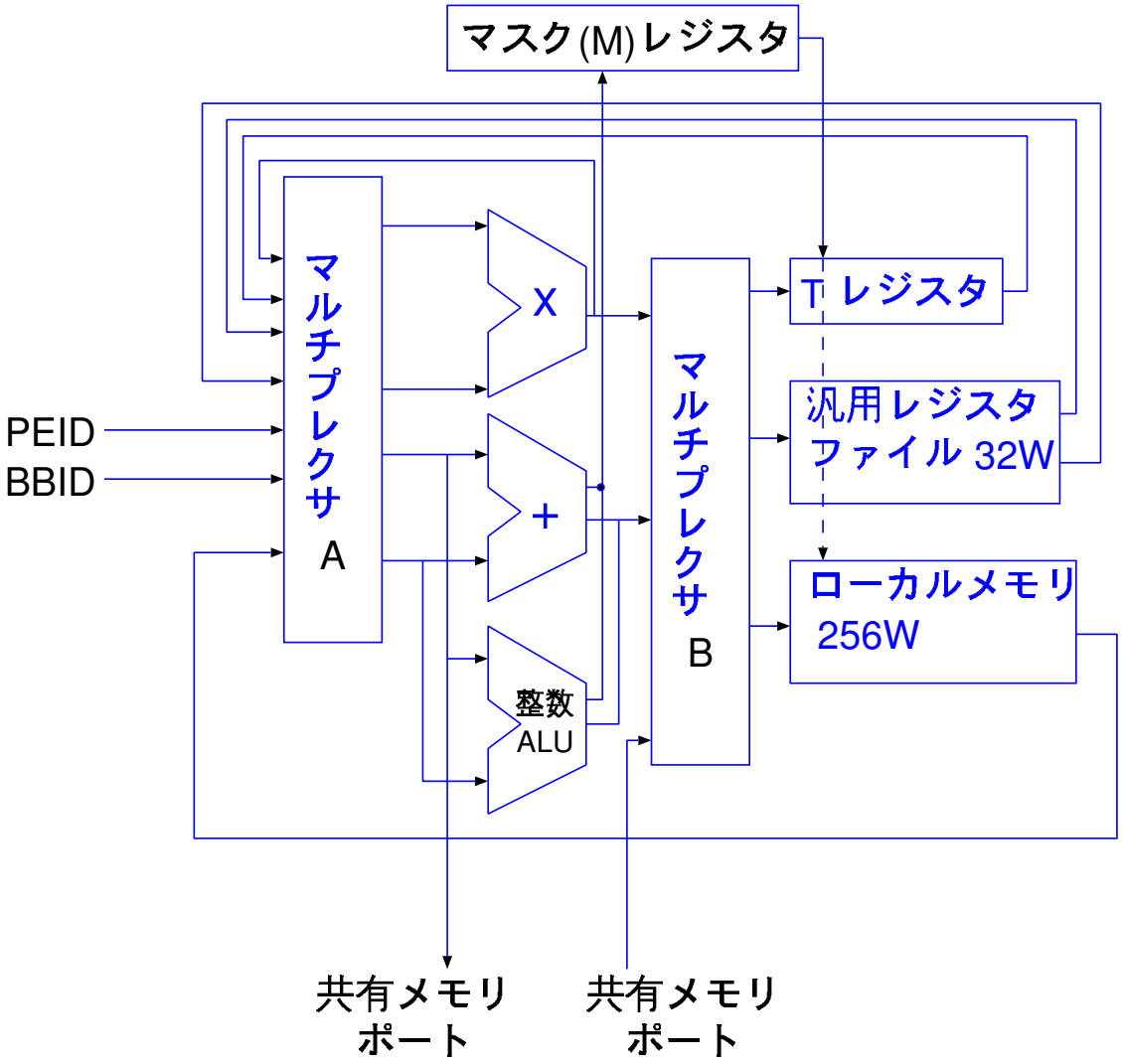
チップ外ポートから放送メモリには

- 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。

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について

サポートする命令等は基本的には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。

GRAPE として以外にどんなことができるか

これから考える。
ソフトウェア: 一緒にやる人募集中。詳しくは平木さんにどうぞ。

専用計算機についてのまとめ

- GRAPE では、もっとも演算量の多い相互作用の計算だけを専用化し、他の計算は汎用計算機に回すことで高い性能と柔軟性を両立させている。
- この分担のためにハード開発は比較的容易である。
- このような分担をうまく機能させるためには、採用するアルゴリズムがそういうふうにできている必要がある。可能ならアルゴリズムの変更も考えるべき。
- とはいえ、次世代プロジェクトは専用性が薄くなるかも。

Case Study： GRAPE-6 の場合

演算方式、論理設計： GRAPE-4 のものを多少改良／変更。
大きな問題はなかった。
実装：テクノロジ微細化、低電圧化のためにいろいろ問題が起きた。

- レイアウトが収束しない（タイミング要求を満たせない）。
- チップ内で電源電圧が下がる。
- 負荷変動に電源が対応できない。

レイアウトが収束しない

（最近はもっと大変らしいけど）ディープサブミクロン設計でありがちな問題。

配線遅延 ＞＞ スイッチング遅延

→ レイアウトしないと遅延時間がわからない

論理設計段階では仮配線長を仮定するやりかたでは駄目。

レイアウトと論理設計を並行して進めるような手法が必要

GRAPE での配線遅延

GRAPE では

- 長い配線はパイプラインとインターフェース回路の間くらいしかない
- そこのタイミング要求はあんまり厳しくない（マルチサイクルでいいのが多い）

ので比較的楽で、大幅なレイアウト修正なしでなんとか誤魔化していた、、、

実際、ほぼ同じ時期に同じ会社が作っていた高速マイクロプロセッサに比べれば半分以下の期間でレイアウトが終った。

チップ内で電源電圧が下がる。

最初のチップでは

- パッケージの電源ピンからダイのパッドまでの配線抵抗
- ダイ上の電源供給ネットの抵抗

が大きかったために、内部で電圧が下がってまともに動かなかった。

計算中と非動作時では2倍程消費電力が違うので、電圧もその分変わる。

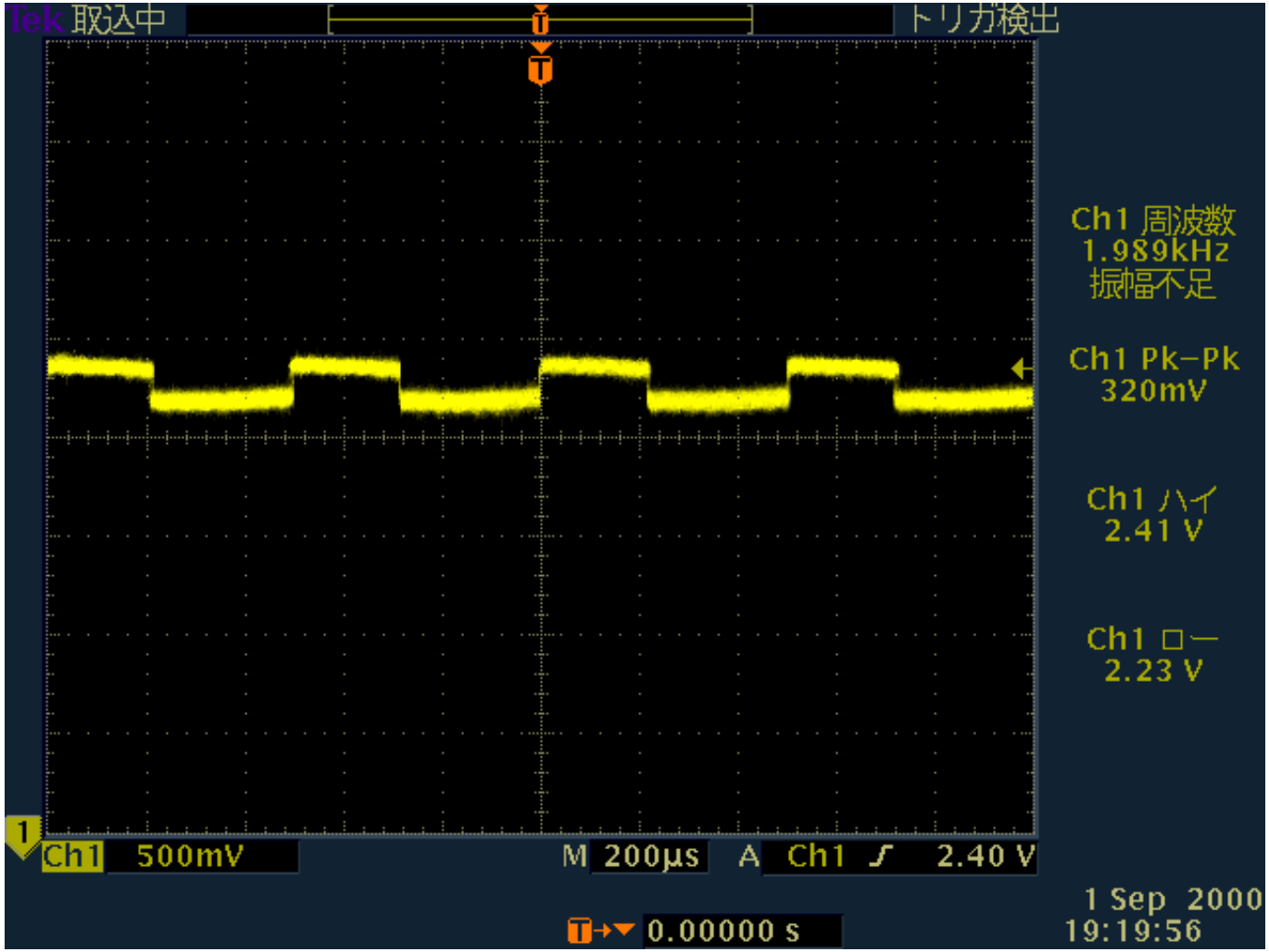
形式的な原因：チップ製造を担当した会社の設計ガイドラインに、コアロジックの消費電力から電源ピン数等への制約がなかった（らしい）

負荷変動に電源が対応できない。

チップが動作時と非動作時で2倍くらい消費電力が違う。動作時間は$10\mu s$ から $1ms$ 程度。ボード1枚では $70A$ と $140A$ の間で変わる

- **普通のスイッチング電源では電流変化に応答できない**
- **大容量の電解コンデンサをつけてもほとんど気休めにしかならない**

電源波形


Tek 取込中
トリガ検出
Ch1 周波数
1.989kHz
振幅不足
Ch1 Pk-Pk
320mV
Ch1 ハイ
2.41 V
Ch1 ロー
2.23 V
Ch1 500mV
M 200µs A Ch1 2.40 V
0.00000 s
1 Sep 2000
19:19:56

負荷変動への対応

- スイッチング電源をフィードフォワード制御する？
- 非動作時でも消費電力が下がらないように設計する？

現在の対応

- ある程度応答の速い電源に変える
- 内部抵抗の小さい電解コンデンサを大量につける

最近の PC 用の電源回路は素晴らしく応答が速いのであまり問題ではないかも。

Case Study からの教訓

結局は：

専用計算機も大きくなると大規模システムで起きるようなあらゆる問題が起きる。

というだけ。

あとは、、、

担当エンジニアのいうことは正しいとは限らない

↓

ボードやチップは専門家にまかせておけば出来るというものでもない

とはいえ、これはあまり専用計算機固有の問題というわけではない。