GRAPE-DR 開発の現状と GDR-2 の展望

牧野淳一郎

概要

- GRAPE-DR 開発の現状
  - 開発日程
  - チップ論理設計について
  - チップ物理設計について
- GDR-2 の展望
  - どこからそんな予算がふってくるか？
  - 作るとすればどんなもの？

GRAPE-DR 開発の現状

- 開発日程
- チップ論理設計について
- チップ物理設計について

GRAPE-DR 開発日程

今年度の計画

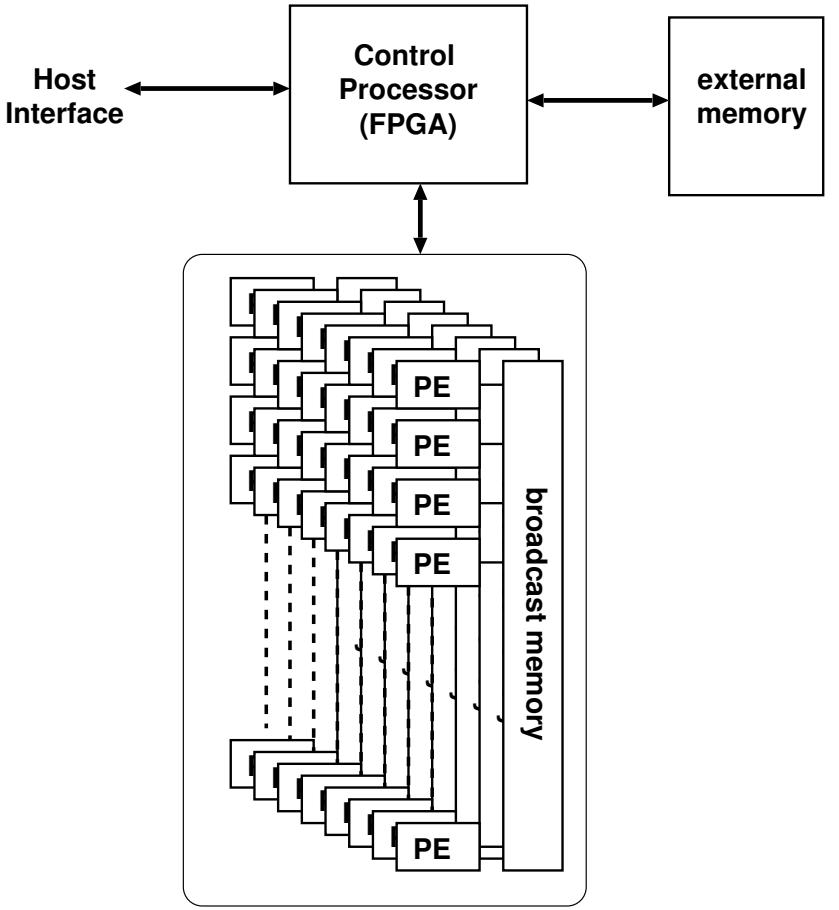
- 今年度どっかくらいにチップサンプル
- チップ 1 個載せるボードもなんとか

チップのほうのもうちょっと細かい計画

	予定	実際
論理設計 FIX	8/5	8/19
テープアウト	10/	12/?
サンプルチップ	06/1	06/3?

論理設計 fix が遅れた理由: 元々 8/5 は無理だった
テープアウトが遅れる理由: 論理設計 fix の後大きな変更をした
どんな変更か、というのは後で詳しく

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

実際のチップ内ネットワーク

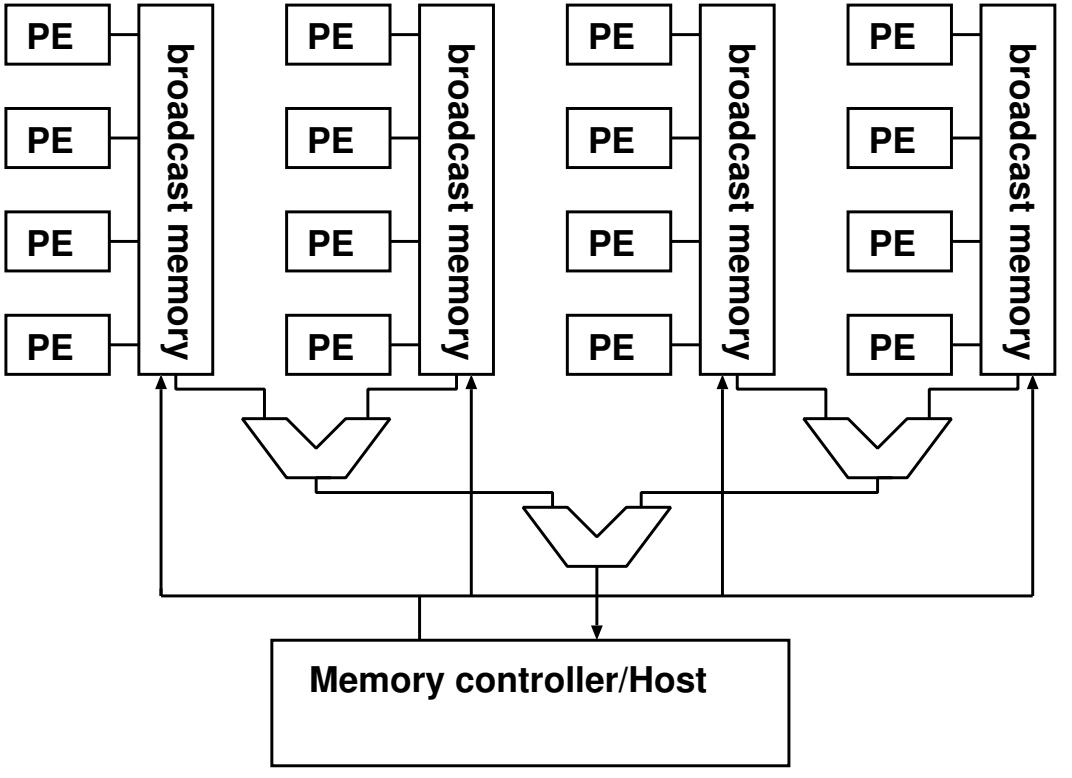
PE は放送メモリとだけ接続

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

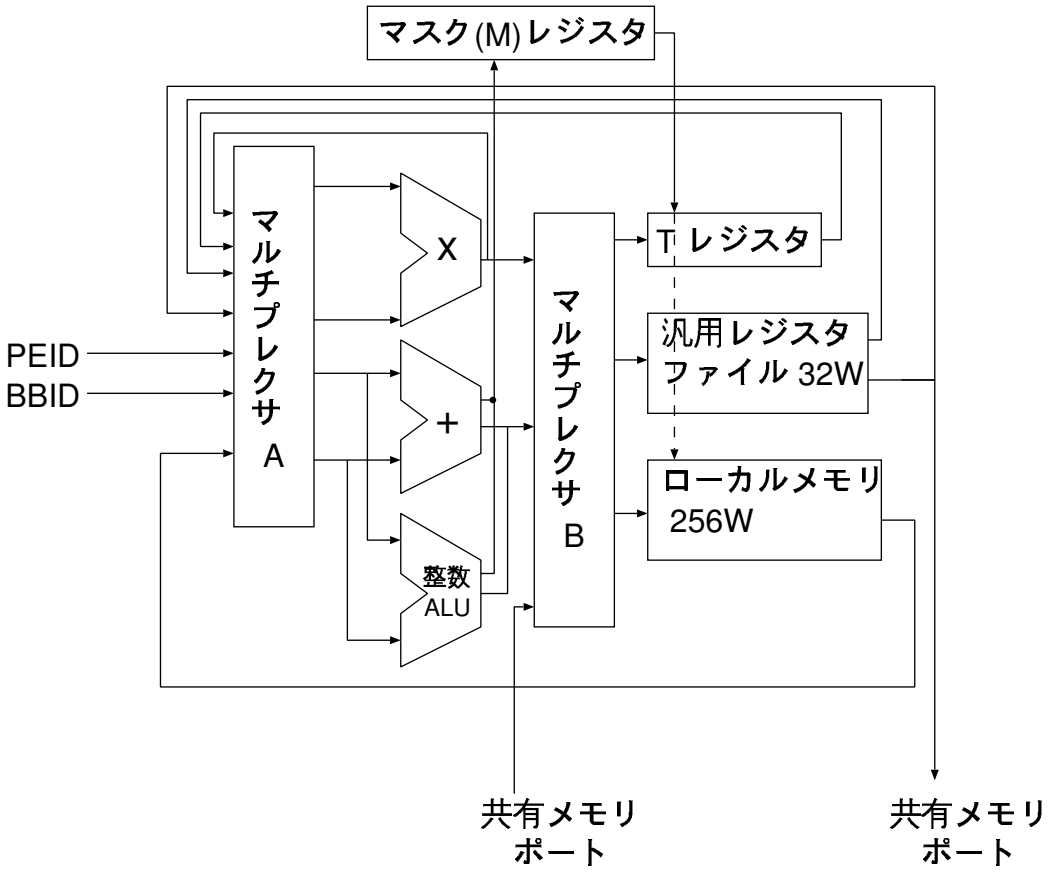
チップ外ポートから放送メモリには

- 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)

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 レジスタについて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(間接アクセスが可能)

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。

チップの細かい設計

行列乗算でハードウェアとしては理論ピークがでるように細かい最適化を一杯した。

- **データ入出力の速度**
- **データ入力、出力、計算のコンカレント動作**
- **出力の裏サイクルでの書き込み**

行列乗算

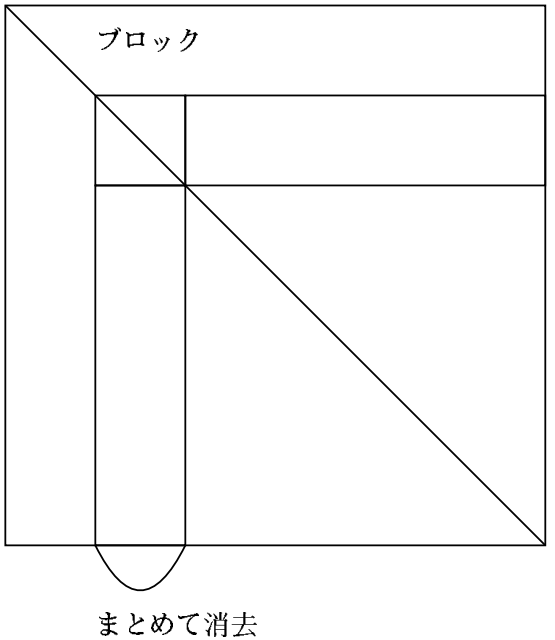
何故行列乗算したいか？
基本的には大抵の線形計算のアルゴリズムはブロック化とか色々して行列乗算が計算量のほとんどになるようにできるから。

具体的例: LU 分解
Gustavson (1996) が「再帰的ブロック化」を発明

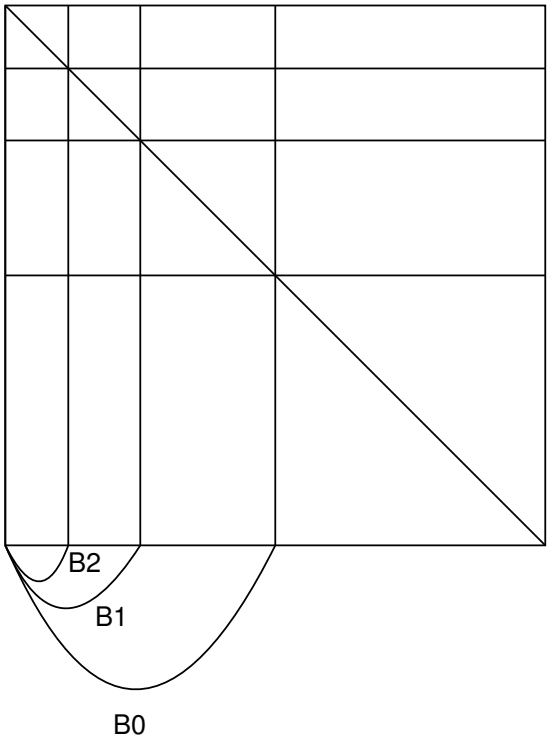
- **まず可能な最大サイズでブロック化**
- **ブロック化できてない残りをさらに半分のサイズでブロック化**
- **ブロック化しても速くならないところまでこれを繰り返す**

Gustavson のアルゴリズム

普通のブロック化



再帰的ブロック化



ブロックサイズが小さいところの計算量は小さい:
サイズに比例して効率が落ちても計算時間そんなに増えない

VPM での行列乗算

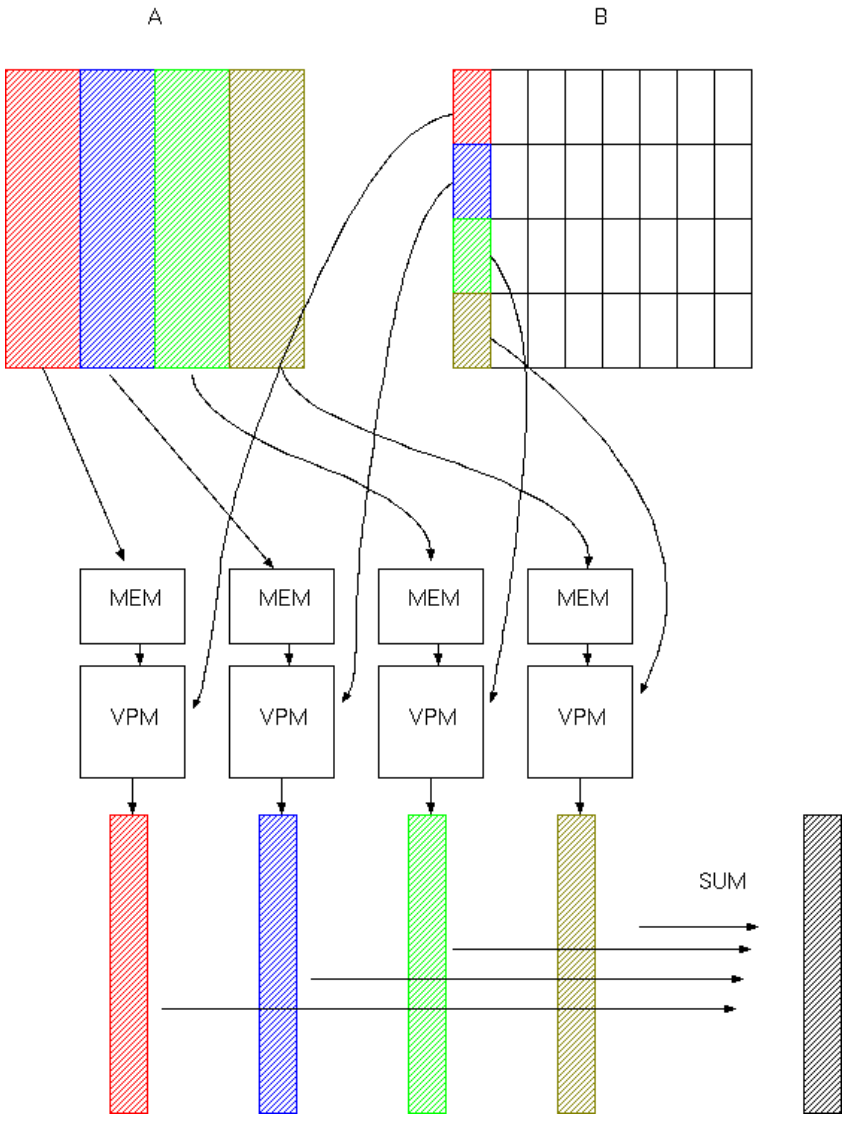
- PE, BB, チップ全体、ボード、ホストの各レベルでブロック化
- 発想は単純だが手順は複雑

以下、ちょっと詳しくトップダウンで説明する。ボードは4チップからなるとする。

行列乗算手順

1. もとの行列積を 2048^2 の部分行列の積に分解。 VPM ボードではこの 2048^2 の行列の積だけを考える。
2. $C = A \times B$ をするとして、 A を縦に 4 等分して各チップの外付メモリに格納
3. B の最初の 256 行を横に 4 等分して各チップのローカルメモリに格納
4. A の 1 列を送って B との積を取る。4 チップで求まったものを合計すると C の 1 行目の最初の 256 個が求める。
5. 上を A の全列に対して繰り返す
6. 上を B の残りに対して同様に繰り返す

行列乗算手順



- A はメモリに。チップで 1 列計算している間に次の列を転送
- B は 1 つ計算している間に次のをメモリまでは転送
- C は計算している間に前のを回収
- チップ内でも同様な分割 (512×256 を 32×8 に)
- シミュレータ、 FPGA 実装で一応動作した。

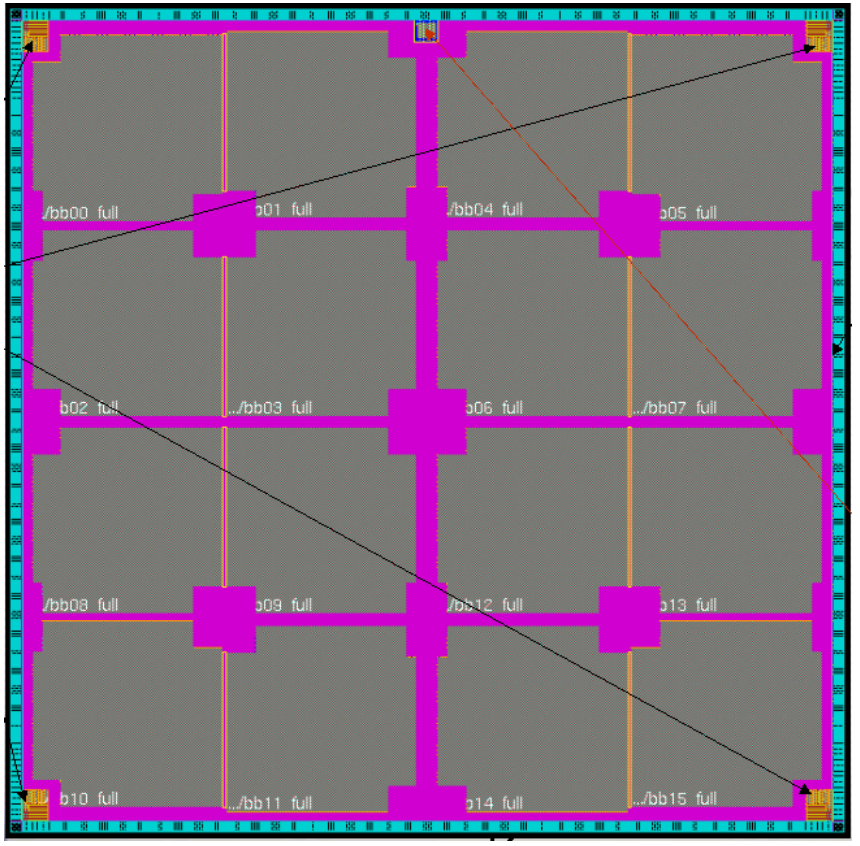
計算上はピークの 90% 以上の性能がでる。

物理設計

することはなんか一杯ある。基本的には設計業者 (Alchip) **の仕事**

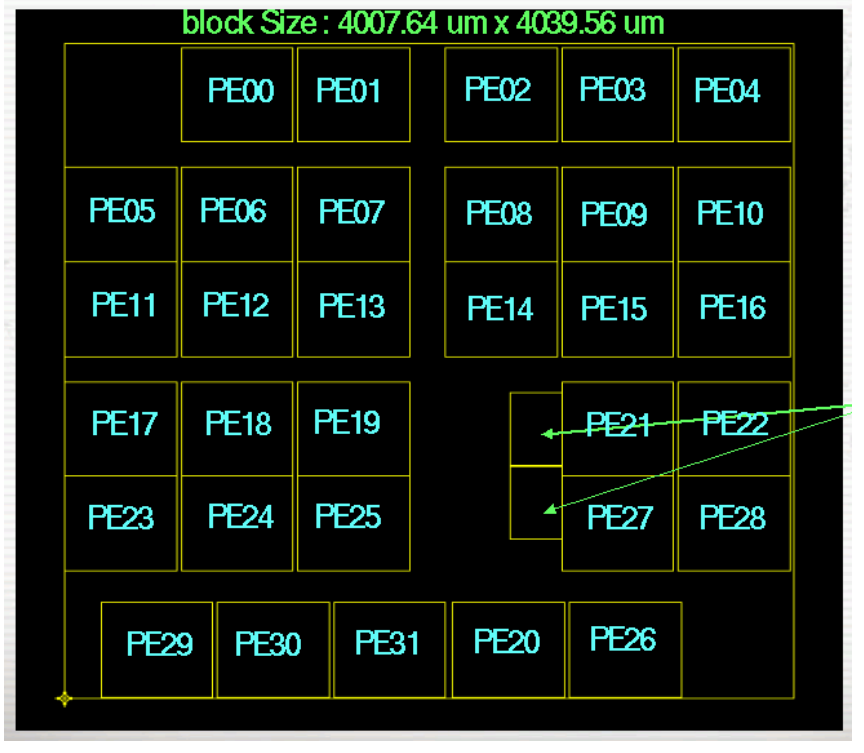
- **フロアプラン**
- **タイミング調整**
- **クロックツリー作成**
- I/O **パッド**
- **電源グリッド**

チップ全体フロアプラン



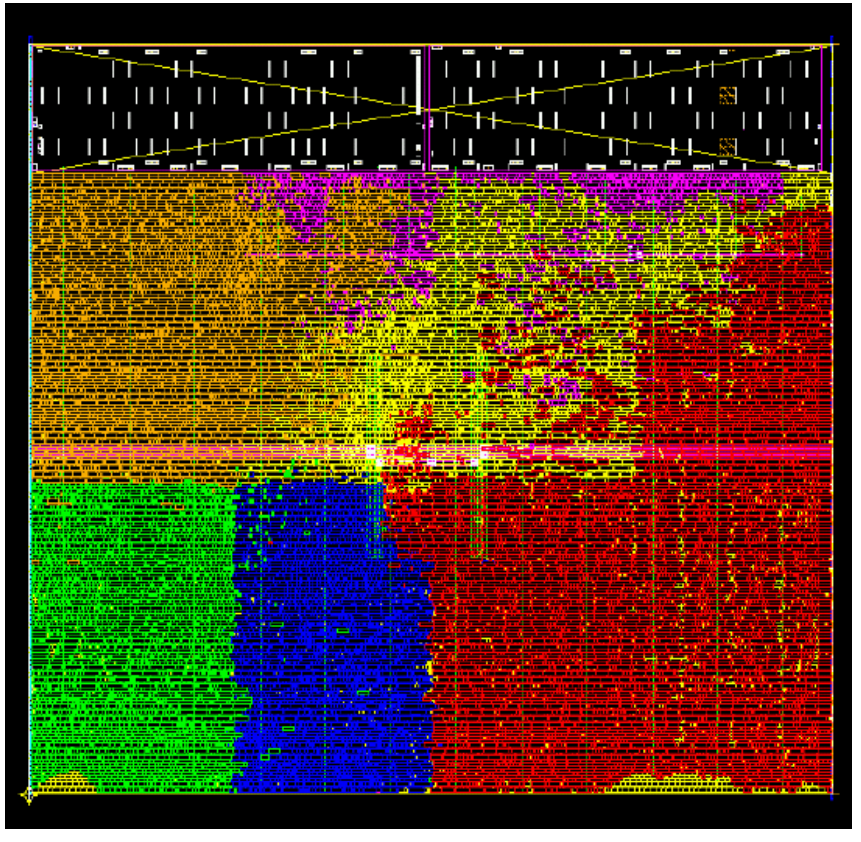
- 特におかしいとこなし
- サイズは大きい。
  17mm 角

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

- **レジスタファイル**がやけに大きい、、、

あれ？

何が問題であったか？

設計業者のほうでレジスタファイルにコンパイルしたライブラリではなくてただのフリップフロップの山を使っていた。

- サイズ
- 消費電力

どちらもとても大きい

何故 8 月まで誰も気がつかなかったか？が問題ではある。

対策

- まともな IP (ライブラリ) を捜す
- フリップフロップをレベルセンシティブラッチに変える

結局ターゲットプロセスで使えて小さくなるレジスタファイル IP はなかった (IBM ならあるのに、、、) ので、ラッチに変えた。

- サイズは大きくは変わらない
- 消費電力はかなり減るはず
- この修正のため 1.5 ヶ月遅れ

GRAPE-DR 現状まとめ

- 予定から 1.5 ヶ月遅れで進行中。
- 遅れの原因はレジスタ実装の変更。
- まあ、3 月にはチップサンプルがくると思う

GDR-2

GRAPE-DR (GDR) がまだできてもいないのに次の話？

事情:

京速計算機=ポスト地球シミュレータ

京速計算機とは？

以下、公開の情報のみ。まだ文部科学省から概算要求がいっただけなので、このまま予算が通るというわけではない。

- 2006 年度開発スタート
- 総費用 (アプリケーションソフトウェア開発とかまで全部こみこみ) 1100 億
- ベクタ、スカラ並列、「準汎用」の 3 種からなる「複合型」
- 「準汎用」だけ 2011 年 3 月完成。この時点で LINPACK 10Pflops が目標。
- 残りは 2013 年

つまり、 LINPACK 10Pflops でる機械を 5 年でででっち上げると、、、

LINPACK とは

本来は密行列演算のライブラリ。最近は LINPACK ベンチマーク、特に High Performance Linpack Benchmark を指す。

- **まともなガウス消去 (ブロック化とかはしていいけどストラッセンのアルゴリズムとかで計算量を減らすのはなし)**
- **行列サイズは制限なし**
- **精度要求はあり**
- **いわゆる Top500 で使っている性能基準**

2011年に 10P ってどれくらい無理か？

過去 10 年の数字

1996	367GF	???	CP-PACS
1997	1.1TF	50億？	ASCI-Red
2001	7.2TF	200億？	ASCI-White
2002	36TF	400億？	地球シミュレータ
2005	136TF	100億？	BlueGene/L

9年で 400倍。

5年で100倍は普通では無理。技術トレンドの 3-4倍上

GRAPE-DR でのLINPACK

- 2009年3月に 600 Tflops くらい。
- この数字自体はそんなに大きくないが、コストが競合の 1/10 くらい。
- 電力消費もやっぱりそれくらい小さい

LINPACK で高い性能を出すことが目標なら魅力的。
他の応用も一杯ある「はず」。

GDR-2 で 10P？

発想は力任せ

- **65nm でチップ作り直す。**
- **10 倍並べる**
- **フルサイズの倍精度乗算器を載せる**
- **細かいチューニング。レジスタファイルちゃんと作るとか。**

これくらいで 20 倍の性能はいけるはず。できればチップ数は 10 倍 ではなくて 5 倍くらいにしたい。

まあ、あくまでも話としては。
「候補の 1 つ」という段階 (全体計画が予算通ったとして)