「演算加速機構を持つ 将来のHPCIシステムに関する調査研究」 進捗報告

主管事業実施機関： 筑波大学
計算科学研究センター

共同事業参画機関：東京工業大学、理化学研究所、
会津大学、日立製作所
協力機関：東京大学、広島大学、
高エネルギー加速器研究機構

「演算加速機構を持つ将来のHPCIシステムに関する調査研究」

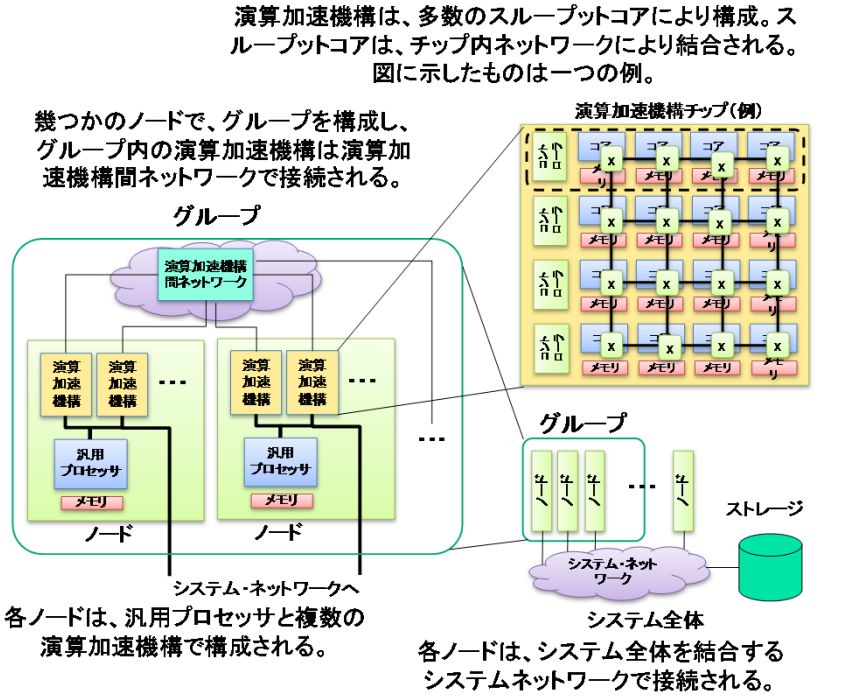
- ナノテクやライフサイエンスの進歩、気候気象予測や地震・防災への対処には計算科学は不可欠かつ有効な手段
  - そのためにはさらなる計算能力が要請されている。
  - 設置面積、消費電力等の制限からノード数の増加による並列システムの性能向上には限界
- ライフサイエンスの分子シミュレーション等、多様な分野で比較的小さい一定サイズの問題の高速化が望まれている(いわゆる強スケーリング)
  - 対応した研究開発の例: ANTON, MDGRAPE-4

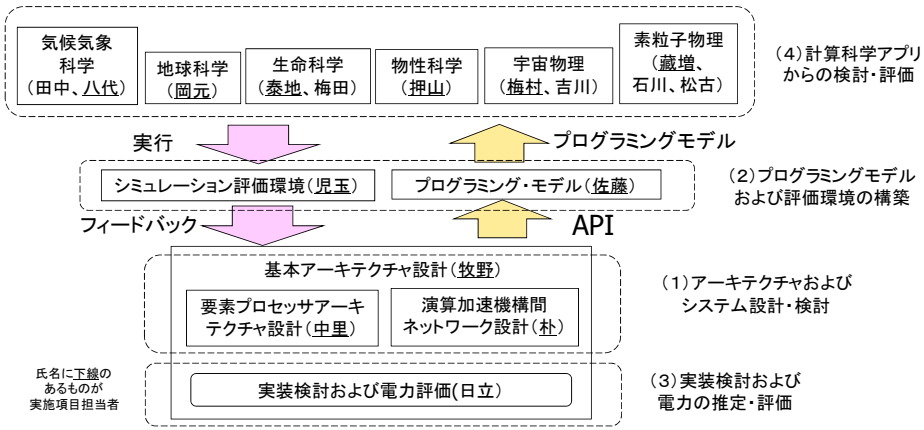
電力効率の大幅な効率化と**強スケーリング問題の高速化**による新たな計算科学の展開を目指して、演算加速機構による並列大規模システムについて調査研究を行う。

平成23年度文部科学省アプリケーション＆コンピュータアーキテクチャ・コンパイラ・システムソフトウェア合同作業部会において、まとめられた「今後のHPCI技術開発に関する報告書」の中で、分類されたシステム構成のうち**「メモリ容量削減」**および**「演算重視」**のシステムを主な調査研究の対象とする。

- 主管事業実施機関: 筑波大学 計算科学研究センター
- 共同事業参画機関: 東京工業大学、理化学研究所、会津大学、日立製作所
- 協力機関: 東京大学、広島大学、高エネルギー加速器研究機構
- 調査研究を、以下の4つの項目に分けて実施
  - (1)アーキテクチャおよびシステムの設計・検討
  - (2)プログラミング・モデルおよび評価環境の構築
  - (3)実装検討および電力の推定・評価
  - (4)計算科学アプリからの検討・評価

多数の演算コアを内蔵したチップによる演算加速機構が汎用プロセッサで構成された並列システムの各ノードに接続もしくは内蔵されているヘテロジーニアスな並列システムを想定





もくじ

- PACS-GのStraw man（たたき台）アーキテクチャと性能概算
- 電力、面積等からのアーキテクチャの見直し
- プログラミングモデル、性能評価環境の進捗
  - PACS-Gソフト開発用シミュレータ
  - サイクルベースシミュレータ

評価対象アプリケーション

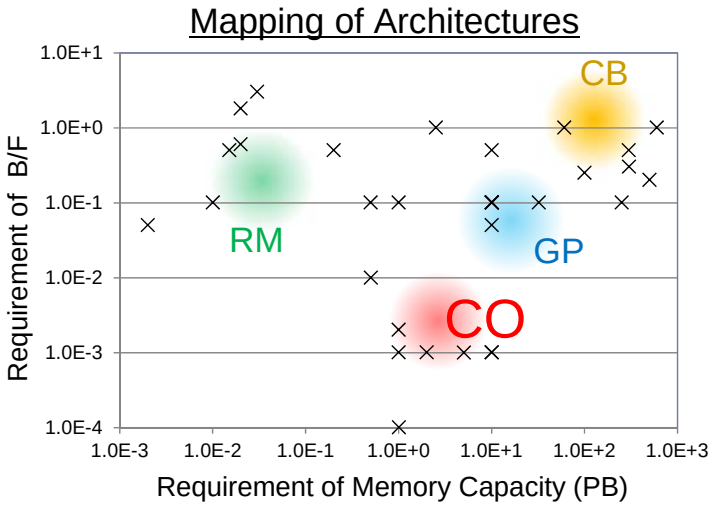
- 計算科学に対する**社会的・科学的課題の達成のために必要なアプリケーション**のうち、本調査研究で対象とするメモリ削減型(RM)および演算重視型(CO)で、ある程度の実行効率が期待できるものの洗い出しを進めている。
  - 生命科学、物性科学における分子動力学計算、生命科学、物性科学、ものづくり分野における第一原理計算、素粒子物理における格子QCD、原子核物理における様々な手法、宇宙物理における粒子シミュレーション、流体計算等(合同作業部会報告より)

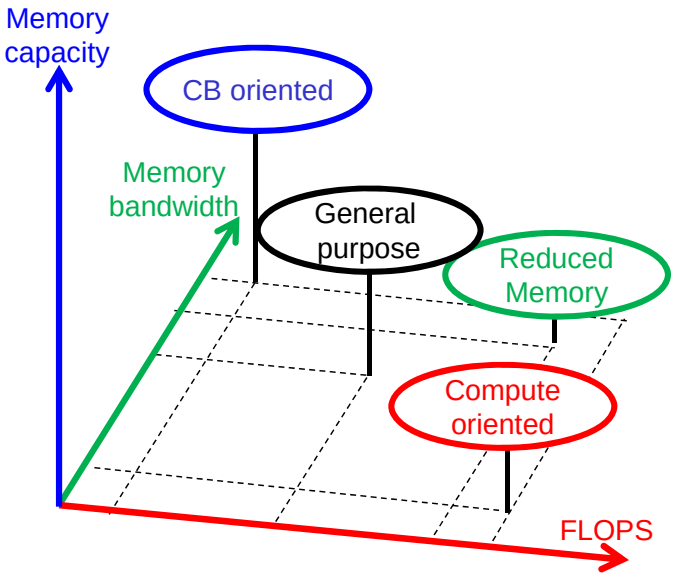
- 本調査研究では科学技術計算における**典型的な計算の一つであるステンシル型の並列アプリケーション**について適用可能の検討を進めている
  - 地震シミュレーション　地震波計算コード　FDM
  - 気象シミュレーション NICAM

- 以下の5つのアプリのカーネルをターゲットとしてアーキテクチャとのco-designを進めている。
  - 格子QCD　（素粒子分野）
  - 重力多体計算　N-body（宇宙物理分野）
  - 磁気流体コード HMD　（宇宙物理分野）
  - 分子動力学　MD（生命科学）
  - 地震波計算コード（地球物理）

- 現在、検討中：NICAM（気象）、RS-DFT（物性）、FMO（化学）

（合同作業部会報告より抜粋）





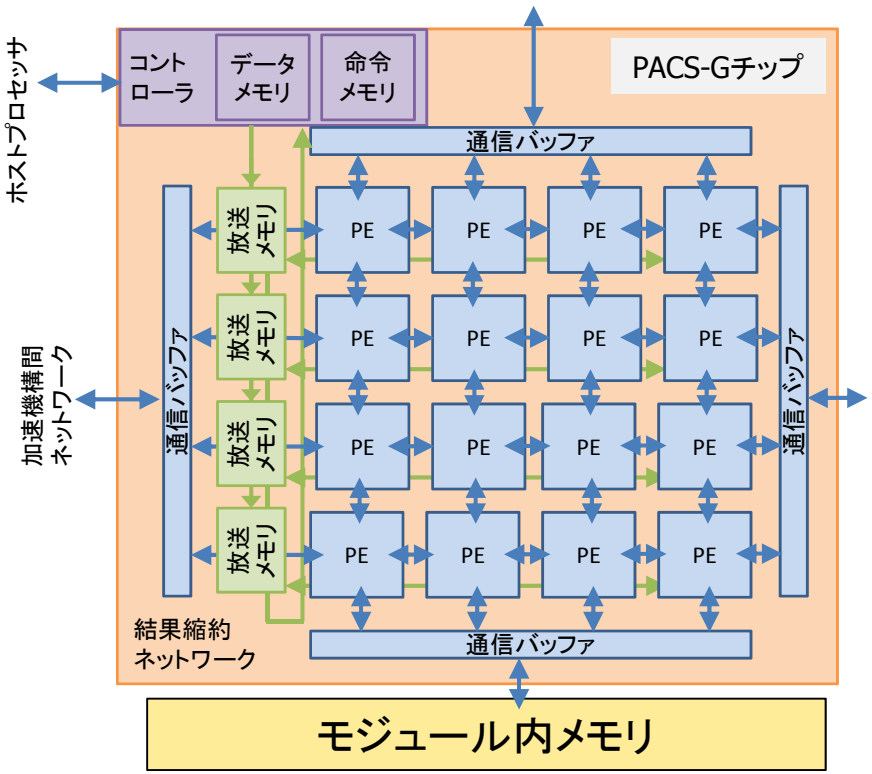
PACS-G アーキテクチャの概要： ノード（チップ）

- 以下のアーキテクチャを、**Straw man（たたき台）アーキテクチャ**として設定

- 演算集約型とメモリ削減型のステンシル計算を両立させるアーキテクチャ（プロセッサ、ネットワーク）をターゲットに設定
- 2018～2020年のLSIテクノロジーとして、14nmを想定。チップサイズを20mm^2 として、メモリ(SRAM)換算で1GB/チップを想定
- チップの基本アーキテクチャは、SIMD
- チップ内は、2次元のメッシュ・ネットワークを（当面）想定 （＋ブロードキャスト・リダクションネットワークを検討）、コア間 16GB/s （双方向）

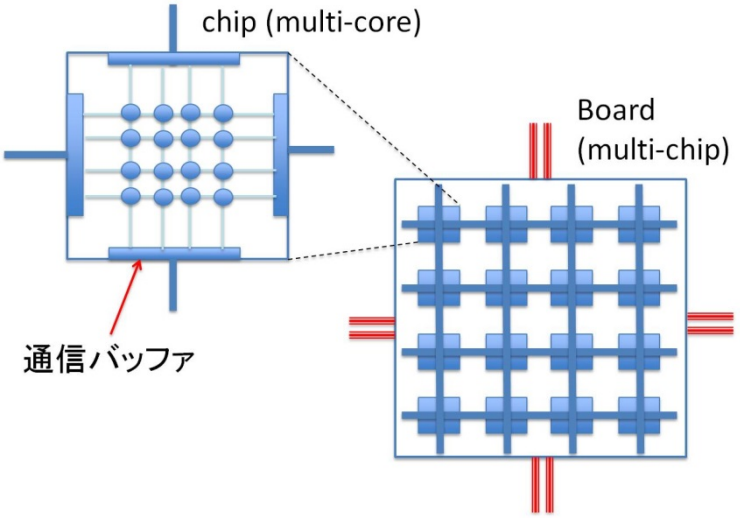
- コアとメモリの比を1:1 として、チップあたり4096 コア（PE） = 64 x 64
- チップ内メモリ 512MB/チップ, 128KB/コア
- コアの基本性能は2FMA@1GHz, したがって、4GFlopsx4096 = 16TFlops/チップ (64Kチップ/1EF)
- TSV 2.5次元実装によるモジュール内メモリを想定。HMC もしくはWide IO DRAM で、
  - バンド幅は1000-1500GB/s
  - サイズは、16-32GB/chip程度
- チップ外付けメモリ(DDR/DIM)は、想定しない

- 電力は250W/チップを目標 （16MW/1EF）
- 2048 チップ/group, Group内のチップは演算加速機構ネットワークで結合



PACS-G アーキテクチャの概要： ノード間、ホスト間

- 1024～2048チップ（グループ）ごとに演算加速機構間ネットワークで結合
- チップの2次元メッシュネットワークをボード上のチップ間ネットワークに展開する際、ボード上のチップ（例えば4x4=16個）を同様に2次元メッシュ結合すると、隣接チップ間（数cm～10cm程度）接続のバンド幅は、チップ内隣接ネットワークの20～40%程度で実現可能（電気配線）
- さらに、ボード間ネットワークまでも2次元（あるいはより高位の多次元）メッシュ展開とすると、インタフェースチップからの光コネクションが使えればチップ間バンド幅と同等（=チップ内ネットの40%程度）が実現可能。

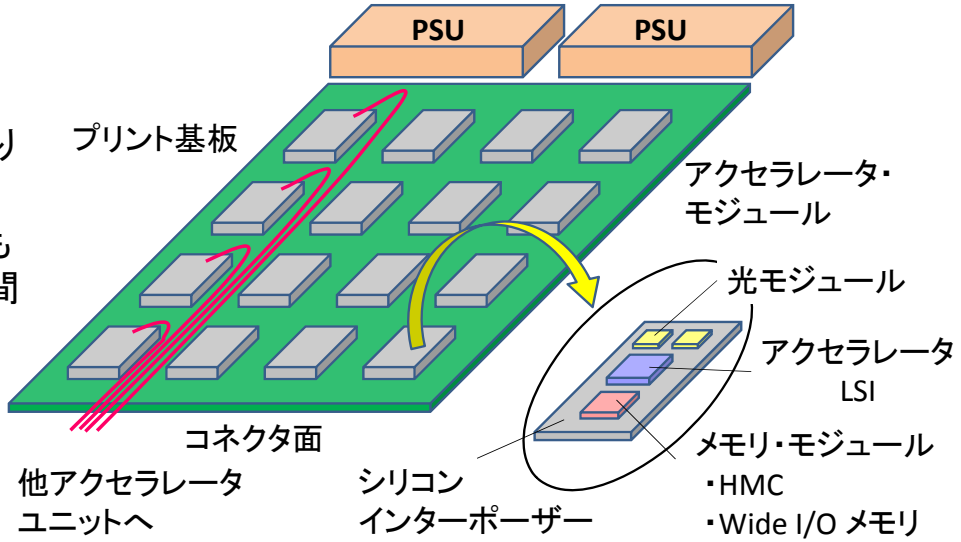


チップ間ネットワークの検討例

チップ内コア間：
16GB/s (= 1GHz x 8B x 2(双方向)@コア間)→1024GB/s (= 16GB/s x 64コア)
チップ間（ボード内、ボード外）：
200～400GB/s (= 32ch. x 25～50Gbps x 2(双方向))

- QCDのような隣接通信のアプリであれば演算のB/F値よりラックサイズのシステムまではメッシュのままで対応可能
- 当面、2次元メッシュで考えるが、もう少し高次元の実装も検討。また、メッシュをトーラスに変更することは、チップ間配線の実装で対応可

- 検討しているシステムは、汎用CPUを基本とした超並列システムにアタッチされることを想定
- ホストとのインタフェースは、PCI Express Gen4 x 16相当の性能を期待



システムの実装イメージの検討例

評価アプリによる性能概算

- 計算の1ステップでの演算、メモリアクセス、通信のパターンと発生量を分析し、想定したプロセッサアーキテクチャ、ネットワークアーキテクチャから期待できる性能を推定。現時点では、グループ単位(～2048チップ,32PF)に限定。
  - 全部データがオンチップに乗る場合:演算・メモリ性能　4 B/F, メモリ1TB/group
  - データが積層メモリに乗る場合:演算・メモリ性能　0.05B/F, メモリ32TB/group

アプリケーション	想定問題サイズ	効率・性能	コメント・比較
格子QCD (素粒子物理)	物理体積$(12fm)^4$ ハドロン多体系 128^4格子	12%～53% 7.9～34.7PF　2048チップ(単精度peak 65.5 PF)	• 評価対象アルゴリズム:領域分割前処理単精度クォークソルバー(ウィルソンクォーク型、BiCGStab法) • オンチップのメモリのみを利用 • 通信レイテンシパラメータの範囲で性能に幅がでる。 • 京では、効率が26%,　32768ノード、1.1 PF
磁気流体コード (宇宙物理)	セル数　1984^3	1.89 PF, 22.5% 512チップ(8PF)	• HLL近似リーマン解法、磁場をflux-CT法よる有限体積法。時間積分を2次精度のTVD Runge-Kutta法 • グローバルメモリを利用 • 210-220ms/step,　Intel Core i7 4096コアで4.5s/step
重力多体計算 (宇宙物理)	814G interaction/sec/chip (単精度、無衝突系)		• 重力計算を演算加速機構で加速。粒子の軌道計算はホスト計算機で行う。オンチップメモリのみ使用 • Intel Xeon E5-2670 の 66.7倍
分子動力学 (MD)カーネル (生命科学)	1セル/コア、1セル$(5Å)^3$, カットオフ半径12Åを仮定 2580原子/コア	3.67PF、 最大15M原子/256チップ, 784.4us/ステップ	• 近距離相互作用の直接和計算を計算。遠距離相互作用計算、結合力計算は未評価 • セルインデックス法(空間座標分割)とハーフシェルスキームを仮定 • 通信ネックになっていないため小規模問題ではさらに高速化可能? • 京では、全ノードで500M原子、4.6PF, 114ms/ステップ
地震波 計算コード (地球科学)	格子サイズ 2048x2048x512	3.5 PF /1024チップ	• 3次元時間領域差分法(FDTD)、空間差分4次精度、時間差分2次精度、弾性体、速度と応力を変数とするスキーム • オンチップのみ。今後、グローバルメモリも検討。 • 格子間隔 50 m, 最小横波速度 300 m/s を想定した場合 (100×100×25 km, Δt～0.001s), 実時間の10倍程度の速さ • 現状のGPUクラスタでは実時間の1/20 程度の計算速度(1/200)

仕様の見直し

- アプリの性能および電力とのトレードオフで通信バンド幅の見直しを行った
  - 演算加速コプロセッサのチップ間のネットワークのバンド幅が226GB/sと過大なものになっていたが、内部ネットワークからの延長としてmaxで考えていたもの
  - 結論：28GbpsのSERDESを用いて、8レーン／ポート、ポートあたり22.3GB/s, 3次元トーラスのために6ポートで、十分な性能が得られる
    - 地震波シミュレーション（差分法）：strong-scaleを見込むオンチップメモリだけの場合は50GB/sの場合、効率23.3%, 15GB/sの場合効率14.3%。モジュールメモリを使う場合は、計算律速。
    - 宇宙磁気流体コードHMD：50GB/sの場合、効率21.5%, 15GB/sの場合効率18.5%
    - 分子動力学(MD)コード: バンド幅は20GB/sぐらいで十分。むしろ、レーテンシが重要
    - QCD:バンド幅は20GB/sぐらいで十分。むしろ、レーテンシが重要

電力の見積もり(1/2)

- 10nmのプロセスが使えることが前提
- コア:750MHz、倍精度 倍精度1mul+1add=2flops/cycle, 1.5 Gflops/core、0.094W/core
  - 根拠:FSでの試作では28nmプロセスを使った試作コアの電力は、500MHz, 倍精度1mul+1add=2flops/cycle, 1 Gflops/coreで、0.025W/core (typical)、28nm->10nmプロセスで、電力効率は1/4(汎用と同等)1GHzでは、電力効率が悪くなるため、750MHzぐらいが適当との見込み。500MHzから750MHzについては、電力最適化込で、1.5倍と見積もり
- メモリ:2.5DのHBMの場合は、メモリ1個あたりの容量は8GB、メモリバンド幅384GB/s、8W,これを4個搭載することにより、1.5TB/s, 32W
  - もしも、HBMが使えない場合、HMCの場合は、メモリ1個あたりの容量は16GB、メモリバンド幅　224GB/s、15W,これを4個搭載することにより、896GB/s, 60W
- ネットワーク:28GbpsのSERDESを用いて、8レーン/ポート、ポートあたり22.3GB/s, 3次元トーラスのために6ポートSERDESの電力を0.3Wとした時にネットワークの電力は、14.4W程度
- AC/DC, DC/DC等、オーバーヘッドは25%

電力の見積もり(2/2)

- ①試算した結果、2.5Dを使える場合は、70GF/W以上、HMCの場合、60GF/W以上の電力性能の見込み
- ②ネットワークについては、アプリケーションの性能概算の見直しを行い、22.4 GB/s (8レーン) x 6 ポート
- ③コアの周波数は1GHzよりも電力効率を上げるためには750MHzが適当であろう。チップあたりのコア数については、チップの冷却方式など考慮して検討が必要
  - 750MHz, 1FMA/.core, 10240コア(1.53PF) 、2.5D HBM 32GBの場合、183W
  - 750MHz, 1FMA/.core, 10240コア(1.53PF) 、HMC 64GBの場合、213W
- ④ 2.5Dを使える場合は、1.5TB/s, HMCの場合は0.9TB/s

チップ面積の見積もり

- 10nmプロセスを前提
- 現在の試作チップ倍精度1mul, 1 add, 単精度はその2倍で、ロジック部分が 0.4x0.08mm
- メモリ (1kW) とレジスタ(1R1Wx64Wx2) がそれぞれ 0.3mm x 0.06mm 程度
- 28nmから10nmプロセスになった場合、面積が1/4(= x 0.25)から、0.4 の幅で検討。

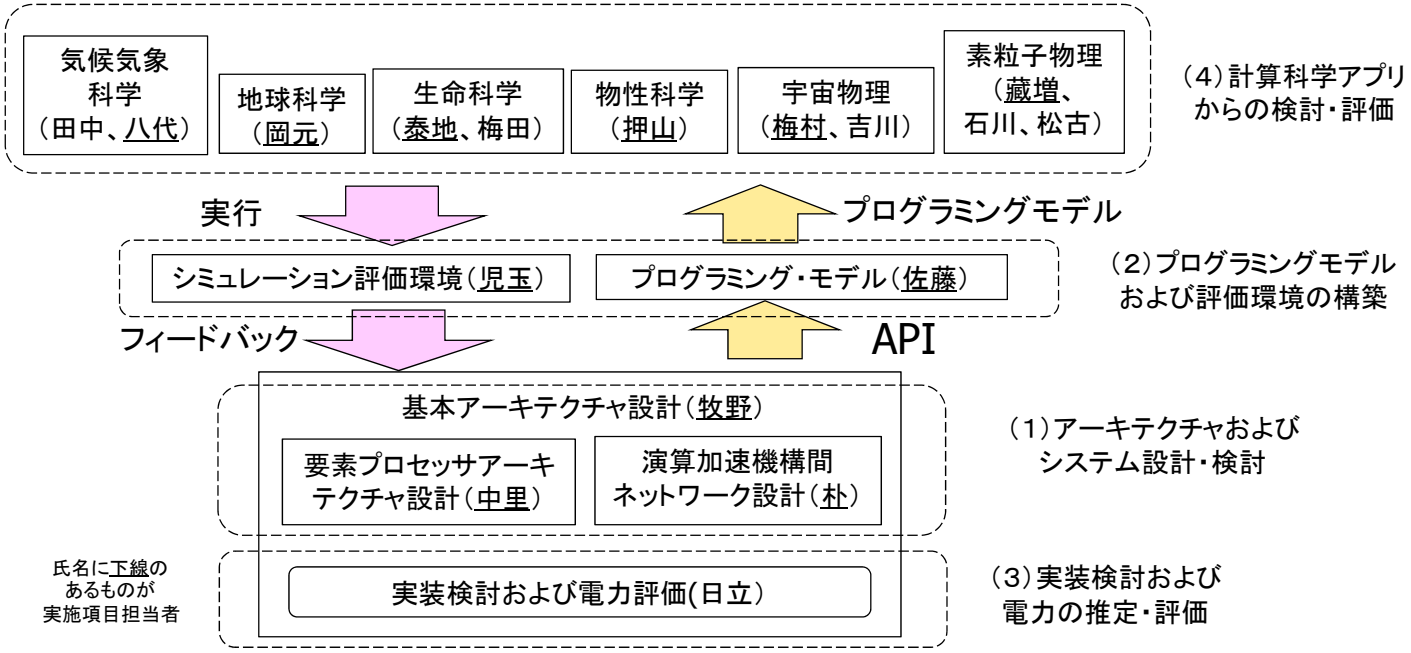
	28nm	10nm (x 0.4)	10nm (x 0.25)
メモリ	0.124	0.0496	0.031
コア	0.032	0.0128	0.008
レジスタ	0.018	0.0072	0.0045
合計	0.174	0.0696	0.0435

- 試作では、2 FLOPS/演算器なので、コアは想定の半分にしたもの、すなわち64KB コアメモリ、8192コアで見積もる。
- 92コアでは、560～357 mm^2となる。下限では20mm x 20 mmで入る見込み。
- しかし、上に述べたように駆動周波数が750MHzになると、性能が12TF/chipになるが、目標性能の16TF／chipを達成するためには、コアを増やすためには、チップサイズを大きくする必要がある可能性がある。
- 面積を大きくすることができない場合には、オンチップメモリ等の削減等を検討する必要がある。

プログラミングモデル、開発・評価環境

- PACS-Gソフト開発用シミュレータ
  - 命令セットの検討
  - プログラミングモデルの検討

- サイクルベース・シミュレータ
  - チップ内の命令実行をクロックサイクル精度で評価
  - PEアーキテクチャは、GRAPE-DRの構成をベースに拡張



PAGS-Gの基本アーキテクチャ

- 大規模メニーコア
  SIMDアーキテクチャ

  GRAPE-DR

  ＋

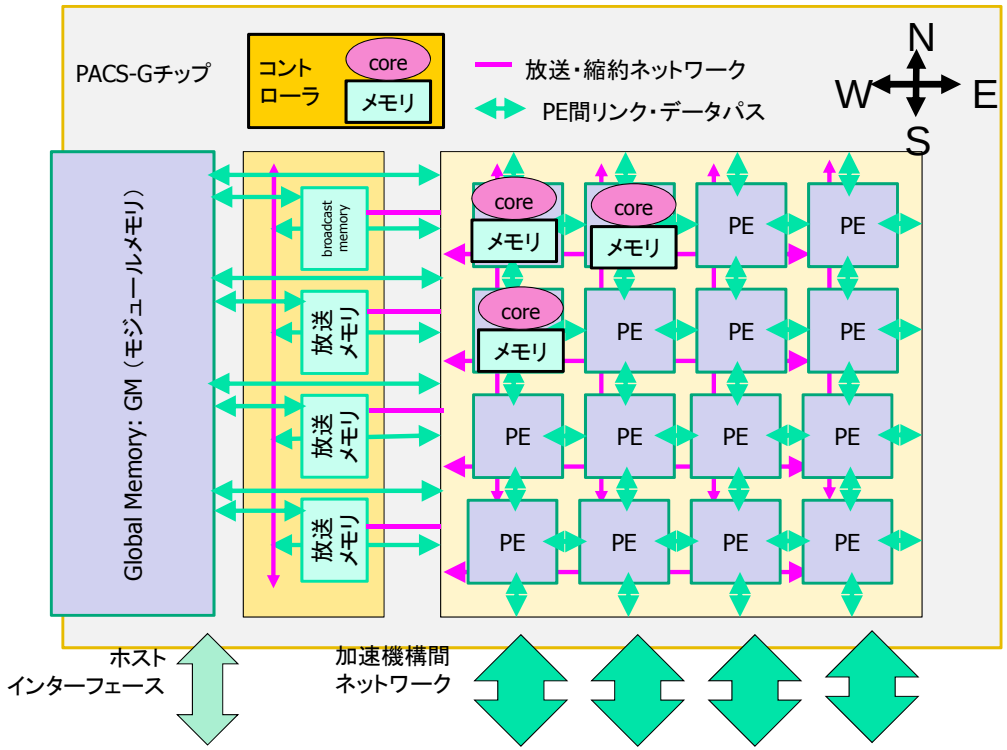
  コア間ネットワーク通信(ステンシル)

  ≒

  CM (Connection Machine)

CM-2 (1987?)

PACS-Gソフト開発用シミュレータ：PACS-Gの基本構成



- PE　プロセッサエレメント。CPUコアとメモリからなる。PE間は、2次元メッシュネットワーク（現在のところ2次元メッシュを仮定）で結合されており、N*M個接続されている。PEから、他のプロセッサにデータを送ることができる。全体がコントローラから制御され、SIMDで動作する。
- コントローラ　PEと同じくCPUコアとメモリからなる。システム全体の命令の制御、SIMD命令のissueを行う。
- 基本命令（演算命令等）は、コントローラとコアで同じ命令が動くものと仮定。基本命令セットは、MIPS-1相当。レジスタは、64ビット長で、256個（これは、コンパイラを楽にするため）。
- Broadcast Memory (BM) PEの各列に接続されているオンチップメモリ。各列のPEにデータを転送したり、同じデータをブロードキャストする。また、各PEのデータを集約処理をして、書き込む。
- Global Memory (GM) チップに接続されているメモリ（積層メモリを想定）。PEにデータを読み込んだり、PEからデータを書き込むことができる。また、GMからBMにデータを転送することもできる。

命令フォーマットとシミュレータの構成

- 命令フォーマット （命令セットはMIPS-1相当）

  [prefix:] opcode operand1, operand2, operand3

  - prefixは、命令を選択実行するマスク8ビットを指定する。prefixが、*の場合は0と同等であり、すべてのPEで実行される。prefixが省略された場合には、コントローラのみで実行される。
  - prefixが指定された時には、上位4ビットm 下位4ビットnに対し、プロセッサの条件レジスタc（setc命令で設定）として、(c & m) ==nの時に、命令が実行される。
  - 命令の記述の中で、*は、don't careを示す。オペランドが省略された場合には、0が指定されたものとみなす。

- PEの通信について
  - PEの通信については、PEの隣接ノードへの通信、BM/GMからのデータ通信、等があるが、PEに特定のレジスタ(Cレジスタ)に格納されるものとしている。

命令セット(1/2)

- **PE命令セット**
  - 各PEのコアとコントローラのコアで実行できる。SIMD実行で、各コアで、実行するかどうかは、指定されたマスクで制御する。
  - 1. 整数演算命令
  - 2. ロードストア命令
  - 3. 浮動小数点命令
  - 4. 実行マスク制御命令・その他
    - PE命令は、アセンブラ上はprefix付の命令であり、prefixで指定されたマスクで指定される条件が満たされた場合にPEのコアで実行されることになる
  - 5. PE間通信命令
    - PE間の転送命令は、2次元メッシュを仮定しているため、隣接へのsend命令だけ。
    - プログラム上、隣接のノードのデータを読むのは隣接ノードからのsend命令になる
    - PEの転送命令で転送されたデータは64ビット。特別の通信レジスタ（Cレジスタ）に入り、これを使うかは各PEで選択。

命令セット(2/2)

- **コントローラ制御命令**
  - これらの命令は、コントローラのみで実行される。
  - 条件分岐、jump命令やcall命令等

- **BM・PE間転送命令**
  - Broadcastメモリ(BM)とPE間のデータ転送・リダクション演算命令
  - BMは、PEのE-W方向に1づつ接続されている。この転送命令は全体で実行する。
  - PE間の転送と同様に、PEへのBcast/moveのデータはCレジスタに入る。

- **GM・PE間転送命令**
  - Globalメモリ(GM)とPE間のデータ転送・演算命令。この転送命令は全体で実行する
  - BMとGM間等のデータ転送については、メモリ上のアドレス指定。これらの命令は、基本的にコントローラで実行されるものと仮定。

- **コントローラ、BM・GM間の転送命令**
  - コントローラ、Globalメモリ(GM)とBM間のデータ転送・演算命令。この転送命令は全体で実行する。

例

- 現在のところ、簡単なCもどきの言語(tiny-C)を使って、コントローラの部分を記述してある。

```
int S;
main(){
    // print pd indx
    asm {
        *: peidx 0xFA,*,* ;   // 各Peで自分のPE座標を取得
        *: peidy 0xFB,*,* ;
        *: prtln "pe=(%d,%d)" ;   // デバック用のプリント機能
    }
    // global sum
    asm {
          *: addi 1,0,1 ;   // レジスタ1に1をセット
       *: stcr 1,*,* ;   // Cレジスタに、レジスタ1の内容を転送
        rdp2bw 0,*,* ;   // BWのアドレス0にリダクション
        addi 1,*,S  ;    // コントローラのレジスタ1に変数Sのアドレスをセット
           rdb2cw 1,0,* ;   /// BWのアドレス0から、変数Sにリダクション
    }
    println("sum=%d",S);
}
```

```
int i;
int S;

// test global memory
main(){

    asm { addi  10,*,100 } // reg 10 =
    for(i = 0; i < 256; i = i + 1){
        asm {
            ldw 1,0,i ;  // load i
            stgw 1,10,0 ;
            addi 10,10,4 ; // incremer
        }
    }

    asm {
        addi  11,*,100  ; // start adc
        mvg2pw 11,*,*   ; // load PE
    }

    // global sum
    asm {
        rdp2bw 0,*,* ;
          addi 1,*,S  ;
        rdb2cw 1,0,* ;
    }

    println("sum=%d",S);
}
```

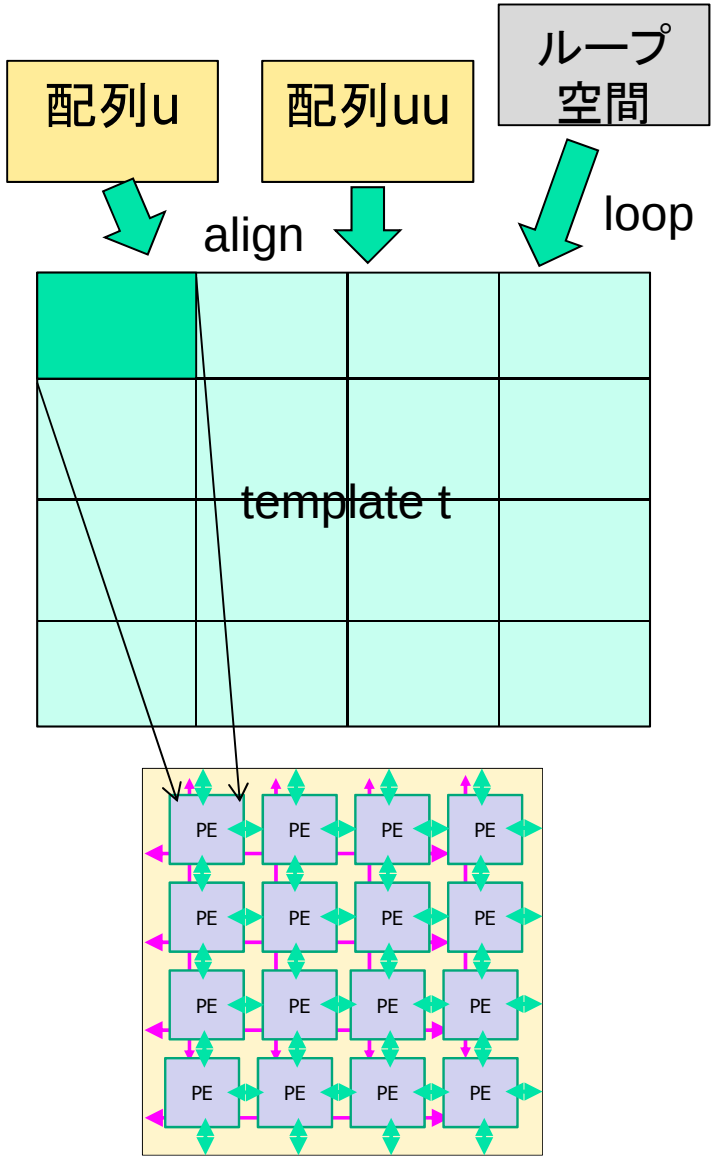
（とりあえず）低レベル・プログラミングモデル

- full setのCコンパイラ (F77も?)
- データ変数に拡張storage classを導入して、アドレスマップを作れるようにする
  - default（何もつけなかった場合）は、PEとコントローラに割り付け。すなわち、PEとコントローラは、同じアドレスマップを持つ（これでOK？）
  - _global_ で、グローバルメモリへ割り当て
  - _bm_ で、BMの割り当て
- PEにSIMDブロックを導入。
  - _pe_block_ { ... }
  - ブロックのなかでは、演算命令かif文のみ。
  - これを、SIMD命令に展開
- レジスタ割り当ての品質を上げることは難しいので、変数のstorage classに指定
  - _reg_<レジスタ番号>
  - 指定された変数はレジスタに割り当て
- 転送命令やその他の命令について、intrinsic関数を準備

XcalableMPによるプログラミングの検討

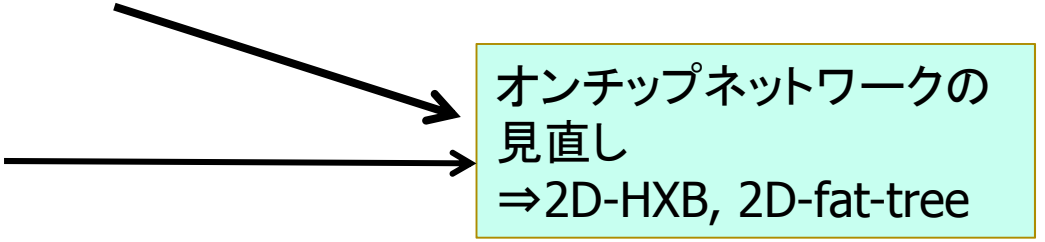
- laplace方程式のソルバーのステンシルのコード
- template指示文で、コア上のtemplate(仮想index空間)を定義
- align指示文で、配列をtemplateに整合させて配置
- ループもloop指示文を使うことにより、iteration空間をtemplateに分割して実行
- block分割しかないため、distribute指示文はなし
- 隣接PEのアクセスは行えるためのでshadow指示文は不要（だだし、チップ間の場合は必要）

```
#pragma xmp template t(0:XSIZE+2, 0:YSIZE+2)

double u[XSIZE+2][YSIZE+2],uu[XSIZE+2][YSIZE+2];
#pragma xmp align u[i][j] with t(i,j)
#pragma xmp align uu[i][j] with t(i,j)

#pragma xmp loop on t(x,y)
  for(x = 1; x <= XSIZE; x++)
    for(y = 1; y <= YSIZE; y++)
       u[x][y] = (uu[x-1][y] + uu[x+1][y]
              + uu[x][y-1] + uu[x][y+1])/4.0;
sum = 0.0;
#pragma xmp loop on t(x,y) reduction(+:sum)
  for(x = 1; x <= XSIZE; x++)
    for(y = 1; y <= YSIZE; y++)
        sum += (uu[x][y]-u[x][y]);
```



プログラミングモデルからのアーキテクチャの検討

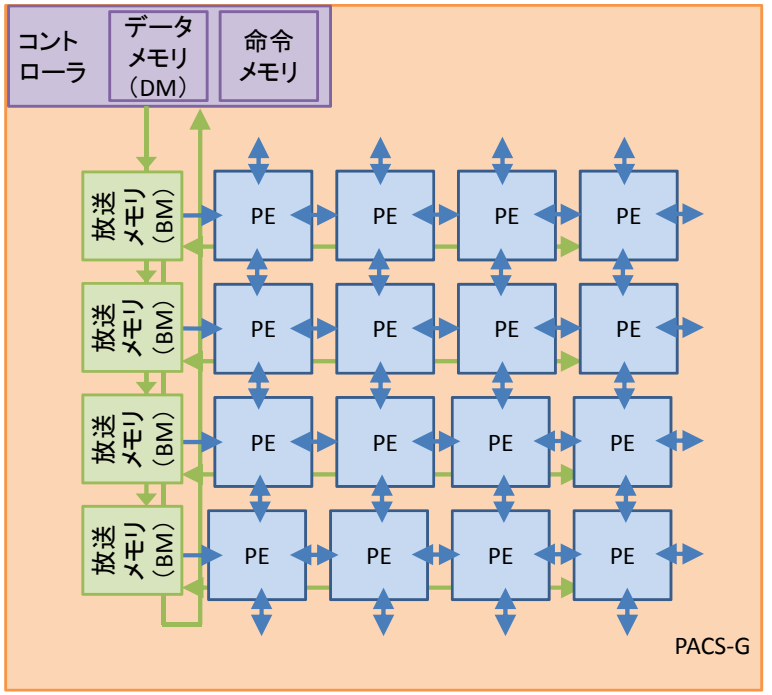
- 他のPEのメモリアクセス(get/put)
  - CM-2ではあった。
  - オンチップネットワークでルーティングが必要
  - SIMDなので、全部が終わるまで待たなくてはならない（PEのなかの計算とはオーバーラップができるが）
- 相対ルーティング
  - 多次元の隣接のアクセス



- チップ間のプログラミングモデル
  - チップ間は、バッファリングして通信する。低レベルは、通信ライブラリ
  - 言語レベルでは、ユニフォームになるようにサポート
- ホストとのプログラミングインタフェース
- OpenCLのサポート
  - OpenCLのグローバルメモリをどこに置くか。
  - 案1）モジュールメモリ(GM)へのランダムアクセスを許す
  - 案2）PEのメモリをグローバルメモリとしてアクセスできるようにする（グローバルなアドレス付を行う）

PACS-Gサイクルベースシミュレータ

- チップ内の命令実行をクロックサイクル精度で評価
  - チップ間通信機構は未実装
  - PEアーキテクチャはGRAPE-DRを拡張
  - PE間隣接通信はPE間の4方向の専用レジスタへの読み書きとして実現
  - GRAPE-DRと同様の放送メモリ
  - ホストとのデータ通信や各PEの命令発行・フロー制御を行うコントローラ

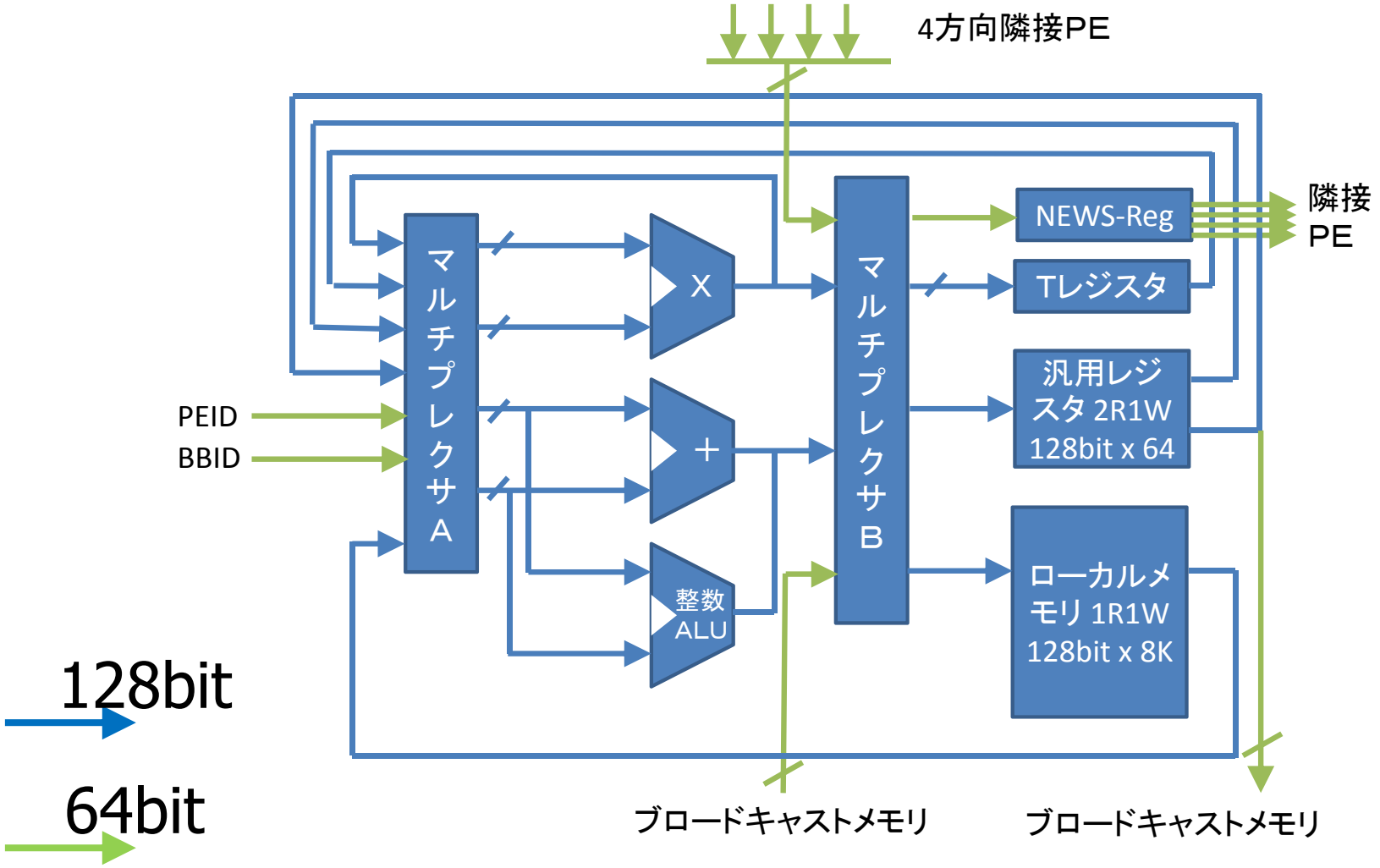


● **HPC-13 GPU・メニーコアコンピューティングIII　　8/2 金13:30 ～ 15:00**

....

(38) 大規模SIMD 型アクセラレータの検討
児玉祐悦, 山口佳樹(筑波大), 中里直人(会津大学), 牧野淳一郎(東工大), 朴泰祐, 佐藤三久(筑波大)

PEのアーキテクチャ

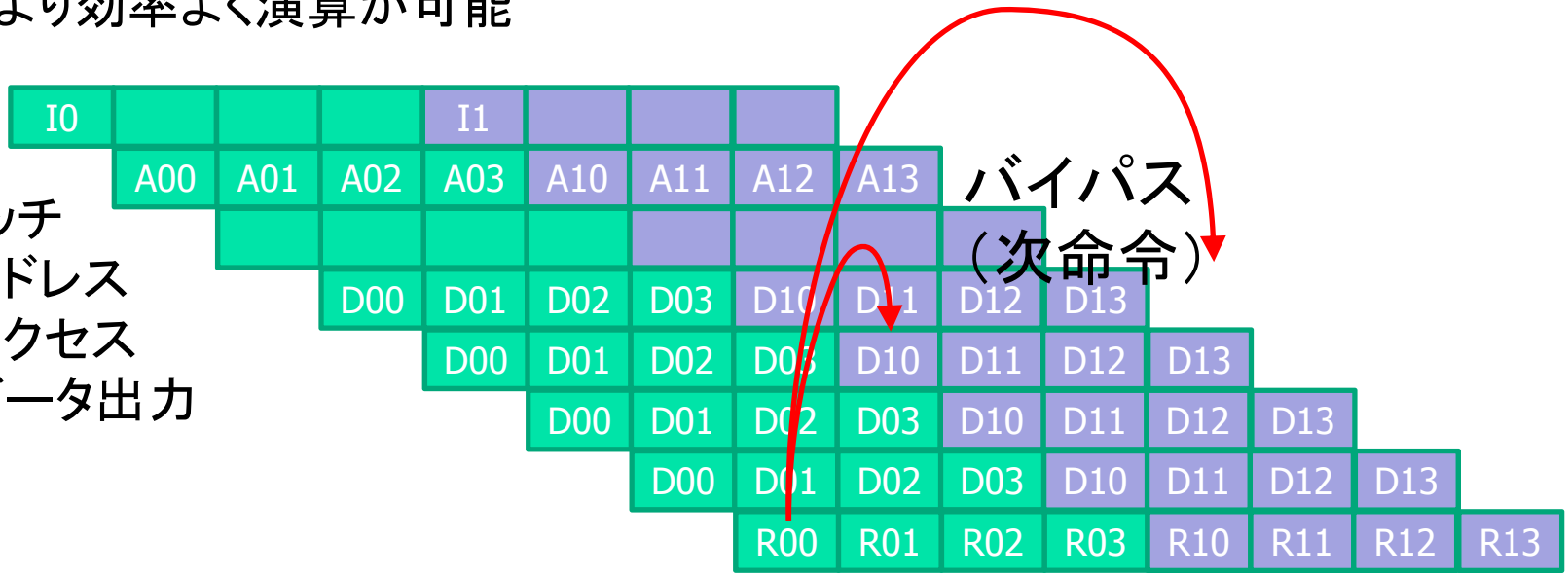

4方向隣接PE
隣接
PE
NEWS-Reg
Tレジスタ
汎用レジスタ 2R1W
128bit x 64
ローカルメモリ 1R1W
128bit x 8K
マルチプレクサA
マルチプレクサB
X
+
整数
ALU
PEID
BBID
128bit
64bit
ブロードキャストメモリ
ブロードキャストメモリ

演算パイプライン

1命令で4サイクルのベクトル演算

- 演算器のレイテンシを隠ぺい
- 倍精度で8演算、単精度で16演算まとめることにより効率よく演算が可能

0:命令ラッチ
1:メモリアドレス
2:メモリアクセス
3:メモリデータ出力
4:演算1
5:演算2
6:演算3
7:結果書込



命令セット

- コントローラで実行されるフロー制御命令
  - DATA i 10 // データ確保
  - LOAD 0 pos // メモリロード
  - DEC 0 // レジスタ演算
  - BNE LOOP // 条件分岐
  - IDP 1 cnst // 放送メモリへのDMA起動

- 各PEで実行される演算命令
  - fmul 命令; fadd/iadd 命令; bm/mv 命令 // 3命令同時実行（レジスタ2read/1write, ローカルメモリ 1read/1writeの制約を満たしていれば）
  - 4ビットの条件フラグから1ビットを指定。通常は1に固定されているフラグが指定され全PEが動作。

サイクルベースシミュレータのアセンブラプログラムによる評価

- FDTD法による差分スキーム
  - 5.2TFLOPS（単精度）
- 姫野ベンチマーク(Middle:128x128x256)
  - 7.4TFLOPS（単精度）
- 行列乗算
  - Kernel部: 14.9TFLOPS (倍精度）
  - 放送メモリへのデータ転送時間を含めると1TFLOPSに低下
  - 512GB/sのバンド幅を持つスタックメモリを仮定すると12.5TFLOPSに向上

サイクルベースシミュレータの今後の評価

- チップ間ネットワークを含めた複数チップでの評価
- チップ内ネットワークの検討で、2次元隣接以外のネットワーク、例えば4次元ハイパークロスバなどの検討を行っており、それに対応したシミュレータによる評価
- 1命令4サイクルのベクトル演算を、通常の1命令1サイクルの命令とした場合の評価
- 計算と通信をオーバラップしやすくする工夫や、その場合の評価

- ソフトウエア開発用シミュレータとのすり合わせ（マージ？）

- CM-2やMasParといったかつてのSIMD型MPPを再評価し、改良の参考にする

耐故障性、信頼性のための検討

- 基本方針として、初期設定で障害コアを検出・回避、運用時には最低限エラーの検出は行えるようにする。一般に大規模SIMDのエラーリカバリは難しいので、チェックポイント等のソフトウエア的な対処が必要

- Defect, プロセスのばらつきへの対処
  - 予備コアの列と行を用意しておき、障害コアが検出された時には、障害コアの列または列をスルーモードとする。（2次元メッシュの場合）
  - 予備コアを導入しやすいオンチップネットワークの検討
  - 大規模SIMD向けのクロック、同期の検討

- 実行時のエラーへの対処
  - 演算回路でのエラー検出
  - メモリ信頼向上として、ECCを付加するほか、ネットワークについても、ECCを付加して信頼性向上を図る。
  - エラーが起きた場合には、チェックポイントによりリカバリを図る

- システムは、ある程度の規模で演算加速機構チップをネットワークで接続する構成になる（現在のところ、2048チップを1グループとして結合）。障害が起きた場合は、このグループごとに切り離して、運用

まとめ・課題

- エクサスケール（エクサフロップス）システムを実現するには、演算に特化した演算加速機構が必要。（電力性能50GF/W以上、汎用では難しい）
- 次世代の計算科学を展開するには、strong scalingが必要。
- ポイントは、2018-2020で可能になる、14nm-10nmテクノロジーを用いて、オンチップの計算がどのくらいできるか（16TF/chipの場合のmemory wall問題、...）

- 次のステップはより詳細なプロセッサアーキテクチャの決定、命令レベルシミュレータの開発と、それによるアプリケーション（システム全体）の性能評価である
  - ネットワークトポロジーの検討（現在は、２次元メッシュ）
  - ある程度のプロセッサを演算加速機構に付加することも検討
  - ネットワークのルーティング機能の検討
  - プログラミングモデルとコンパイラの実装
  - 実際のコードを用いた定量的評価、詳細な電力評価 (アプリFS mini-app)
  - NICAM（気象）、RS-DFT（物性）、FMO（化学）

- 全体システムの検討
  - ホストとの接続形態の検討
  - システム全体としての耐故障機能
  - I/O等