FPGAによる高性能計算

- FPGA-based High-Performance Computing -

東北大学大学院情報科学研究科
計算数理科学分野
佐野　健太郎

謝辞

東京工業大学学術国際情報センター
青木　尊之　教授,　　西川　武志　准教授

東北大学大学院情報科学研究科
山本　悟　教授,　　王　陸洲　氏,　　初田　義明　氏

エクソンモービル
飯塚　尊則　氏

Imperial College, London
Prof. Wayne Luk

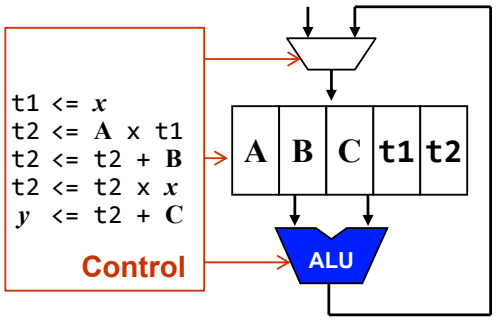
Maxeler Technology, Inc.
CEO Dr. Oskar Mencer,　　Mr. Oliver Pell

内容

- FPGAによる高性能計算とは
- 事例1　ストリームアーキテクチャに基づく流体計算専用計算機
- 事例2　シストリックアーキテクチャに基づく差分法専用計算機
- まとめ

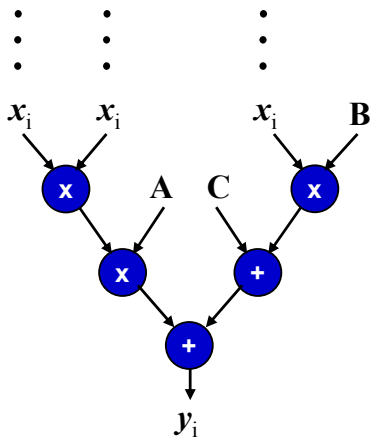
はじめに

$$y_i = A x_i^2 + B x_i + C$$



汎用μプロセッサ

逐次演算 ⇒ スループットの限界



専用計算回路

並列演算 ⇒ 高スループット

内在する**並列性・規則性をHW化**することにより
効率よく高性能計算を実現できる問題がある

専用計算回路を構築するには・・・

- **ASIC/カスタムLSI**

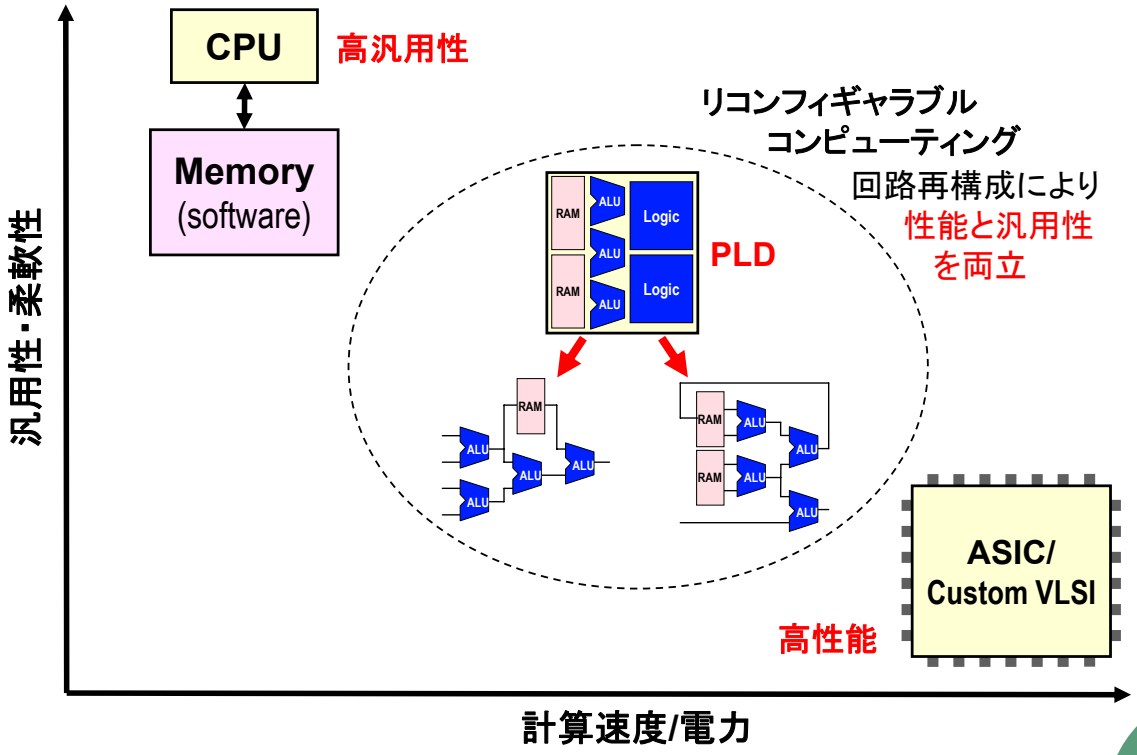
良
- ✓ 最高性能 （膨大な数のTr.を効率よく利用）
- ✓ 大量生産により安価に

悪
- ✓ 高い初期コスト ⇒ 少量生産に不向き
- ✓ 長い開発期間
- ✓ 柔軟性に欠く ⇒ 様々な計算問題への対応は困難
  ⇒ 設計ミスが命取りに

- **柔軟性（汎用性）を損なわない方法**

プログラマブルロジックデバイス（PLD）による
リコンフィギャラブルコンピューティング

リコンフィギャラブルコンピューティング



FPGA（Field-Programmable Gate Array）

- プログラマブルロジックデバイスの一つ
  - ✓ エンド・ユーザが回路を書き換え可能
    - ➢ VLSIの回路エミュレーション
    - ➢ 少量生産品の電子回路
  - ✓ 基盤に実装された状態で書き換え可能
    - ➢ 惑星探査機，携帯の基地局（通信方式の更新）

- 使用例

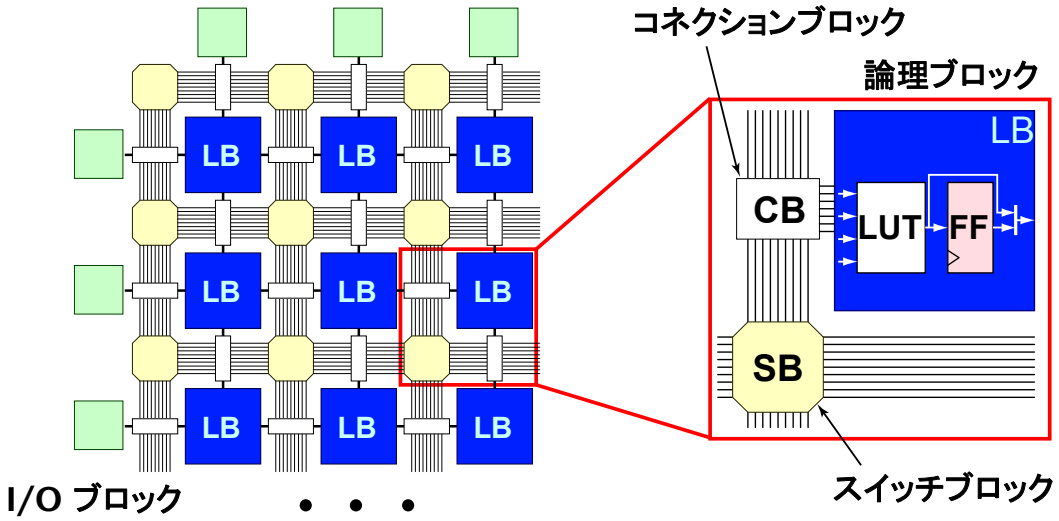

高級ディジタルAV


惑星探査機


携帯の基地局

問題に特化した回路に書き換えて計算も可 ⇒ どのような回路?

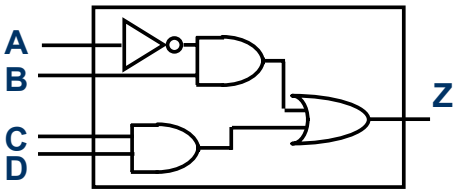
FPGAの基本構造



FPGA = 論理ブロック ⇒ 順序回路（LUT & FF）
+ I/Oブロック ⇒ ピンの入出力機能を設定
+ スイッチと配線 ⇒ ブロック間を接続
（プログラマブル要素）

Look-up Table は万能ゲート

- 組合せ論理回路は Look-Up Tables (LUTs)で実現可
- LUTのサイズは入力のビット幅のみに依存
- LUT遅延は論理の複雑さに依存せず



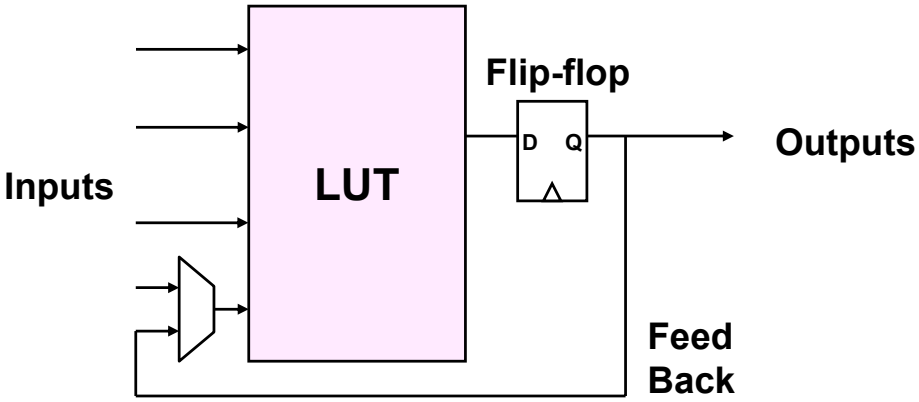
組み合わせ論理回路

address				data
A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
	.	.	.	
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

SRAMで実現可
（⇒ LUT）

真理値表

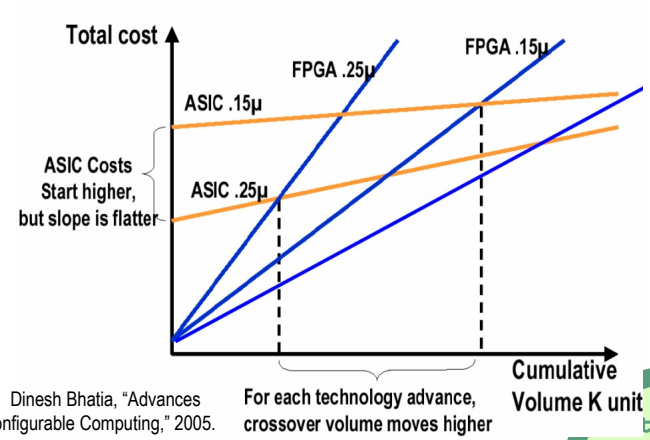
順序回路



LUTs (組合せ論理回路) & **Flip-flops** ⇒ **順序回路**

FPGAは脇役から主役へ

- **初期のFPGA**　**Glue Logic** の置き換え
  （複数のLSIを接続する外付け論理回路）
- **近年のFPGA**　**SoC**（System-on-Chip）として
  - ✓ 大規模化　多数のLogic Element
  - ✓ 複合化　DSP, RAM, μP ⇒ システムそのものがFPGA上に
  - ✓ 高速化　500～MHz動作（DSP, BRAM）
  - ✓ CADの進歩　低いNRE*コスト　短いTTM**

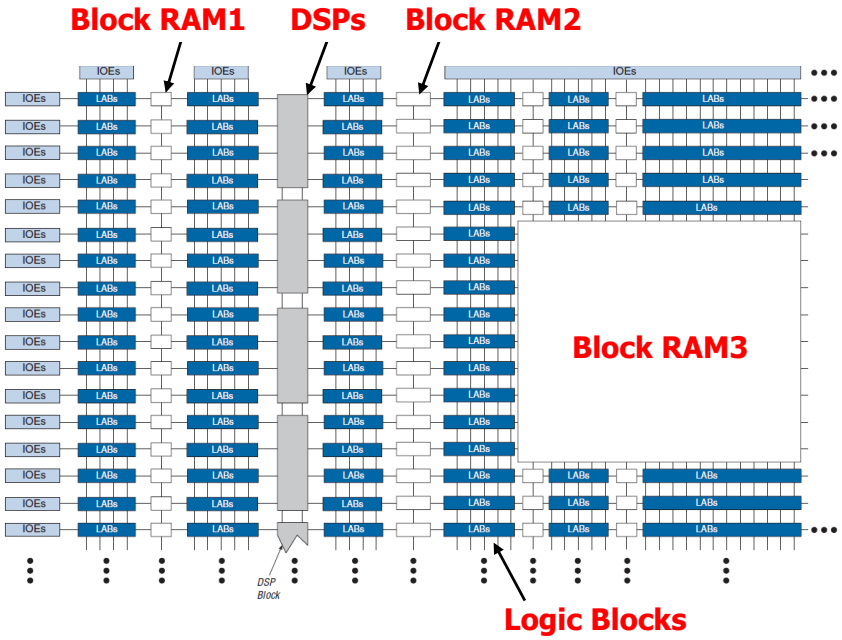


* Non-Recurring Engineering
** Time-To-Market

出展：Dinesh Bhatia, "Advances in Reconfigurable Computing," 2005.

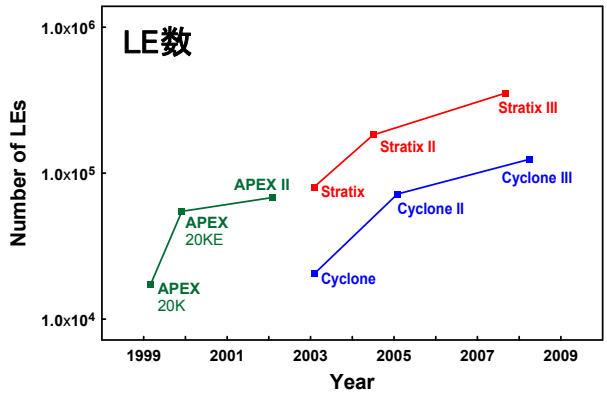
近年の大規模FPGA　ALTERA Stratix-II

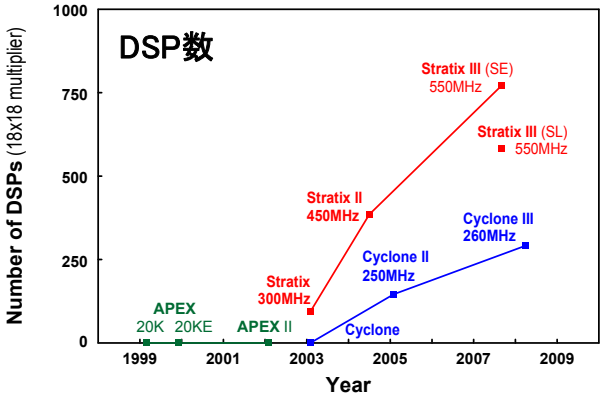
- ✓ **多数のロジック**
  (～ 143520 LUTs)
- ✓ **多数の整数乗算器**
  (～ 96　36x36 DSPs)
- ✓ **内蔵RAM**
  (～ 8.9Mbits BRAMs)
- ✓ **LVDSやメモリI/F**

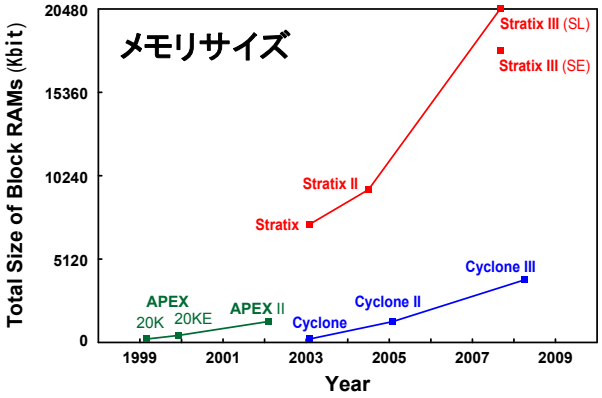


浮動小数点演算器　⇒　DSPブロックを用いて構築

FPGAの進歩（ALTERA FPGAs）

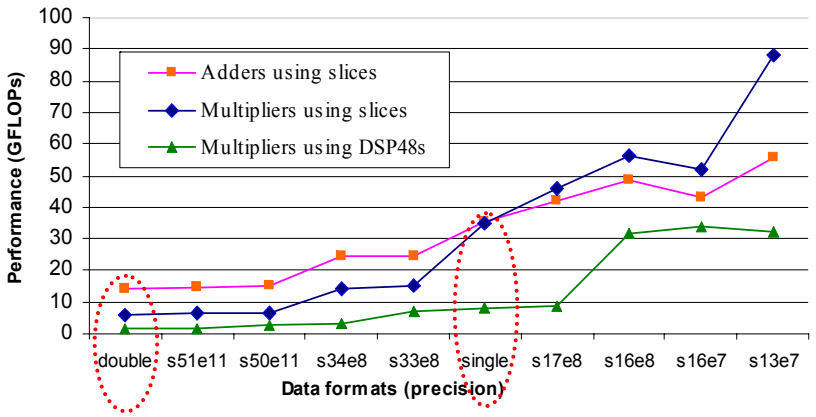






FPGAによる高性能計算

- 整数演算・ビット操作はμPの数百～数千倍の性能
- 浮動小数点演算は?
  - ✓ μPの数倍～の潜在性能（単精度・倍精度），特殊精度はさらに?
    [Underwood, FPGA2004] “FPGA vs. CPUs: Trends in peak floating-point performance”
  - ✓ 4倍精度は？
    μPでは
    倍精度の
    20分の1程度…
  - ✓ 電力あたりは？

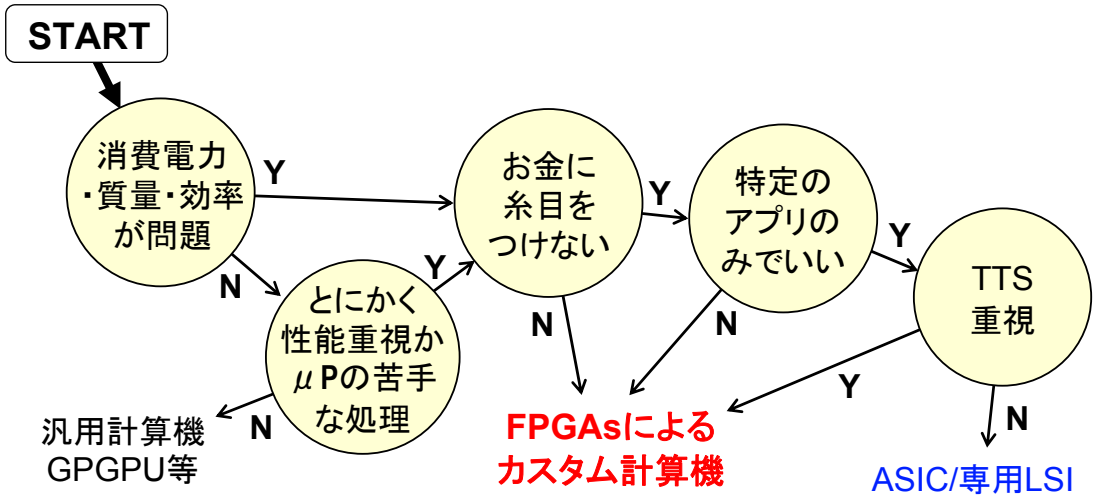


Virtex-II Pro におけるFP精度と性能
（性能 = 演算器数 x 動作周波数）

Junqing Sun, “Obtaining High Performance via Lower-Precision FPGA Floating Point Units,” SC07.

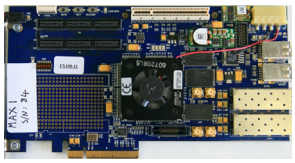
FPGAが適する問題とは？

HW化に向く計算を前提



FPGAを利用可能なシステム

- **FPGAボード**
  - ✓ 主要ベンダのFPGA
  - ✓ I/F（PCI, PCI-Express）



- **FPGAコプロセッサ**（FSB接続）
  - ✓ XtremeData, Inc. XD1000, XD2000F

http://www.xtremedatainc.com/

- **FPGAクラスタ**
  - ✓ SGI Altix RASC Blade, Virtex-II 6000
    SGI Altix RC100 Blade, Virtex-4 LX200

http://www.sgi.co.jp/products/altix/index.html

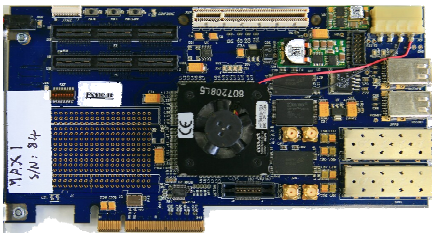
  - ✓ CRAY-XD1, Virtex-II Pro
    CRAY-XR1 “Reconfigurable Processing Blade”, Virtex-4 LX200

http://www.cray.com/global_pages/japan01.html

  - ✓ SRC-6, SRC-7

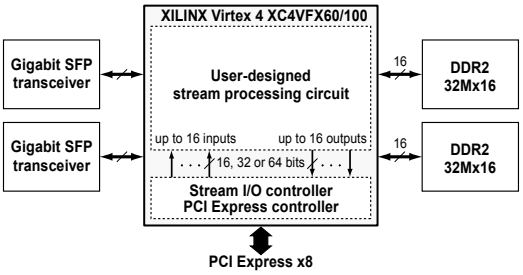
http://www.srccomp.com/index.asp

FPGA搭載ボードの例

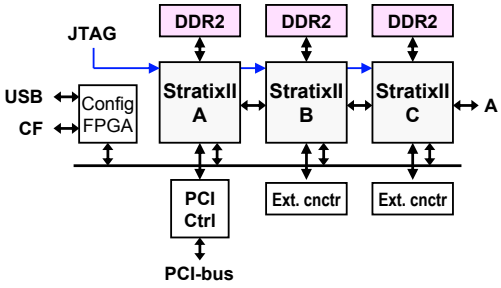


MAX-1
http://www.maxeler.com/content/

DN7000K10PCI
http://www.dinigroup.com/DN7000k10pci.php



PCIe I/FはFPGA内に実装
JTAG・PCIe経由によるコンフィグ

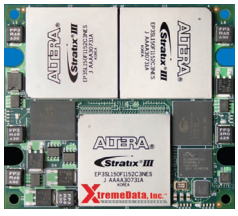


PCI I/Fは別チップ
ローカルメモリ, FPGA間接続

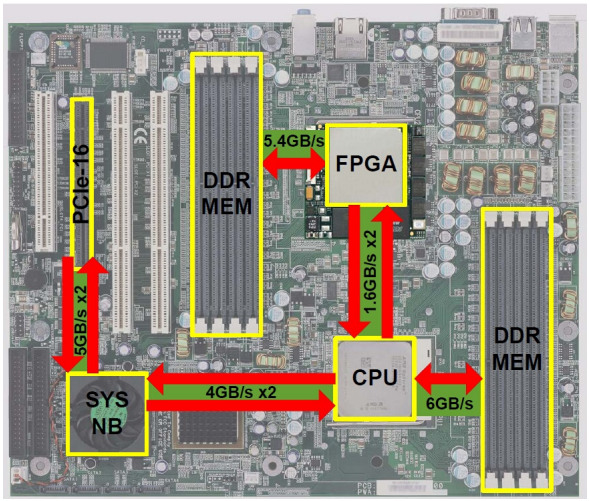
FPGAコプロセッサ（XtremeData, Inc.）

CPU ソケットにFPGAを接続
FSB/Hyper Transport Link経由での
広帯域メモリアクセス

XD2000F
Altera StratixII
FPGAs for AMD
Opteron Socket



XD2000i
Altera StratixIII
FPGAs for
Intel Socket
(to be released in 2008 Q2-)

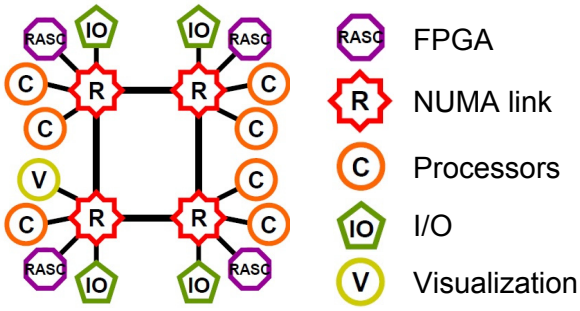


http://www.xtremedatainc.com/

FPGAクラスタの例（SGI Altix RASC）

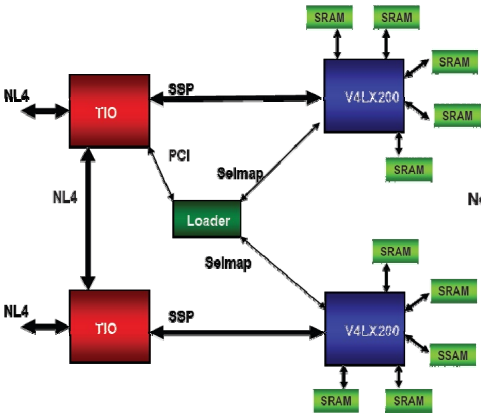
RASC RC100

2x Xilinx Virtex-4 LX200
80MB QDR SRAM or 20GB DDR2 SDRAM
Dual NUMAlink 4 ports



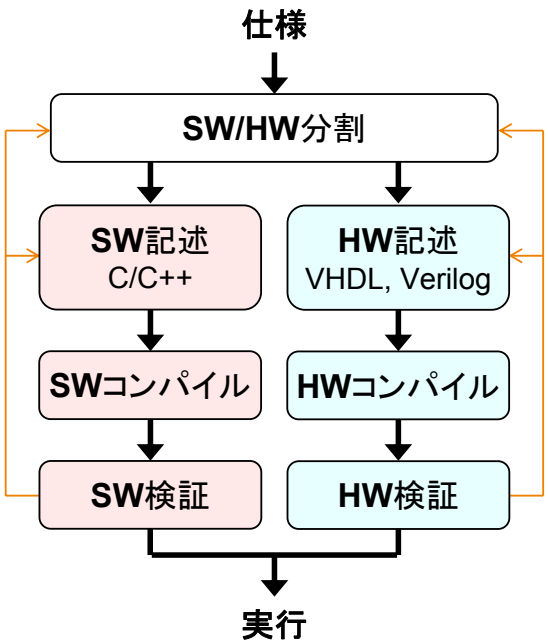
Altix System Architecture

http://www.sgi.co.jp/products/altix/index.html

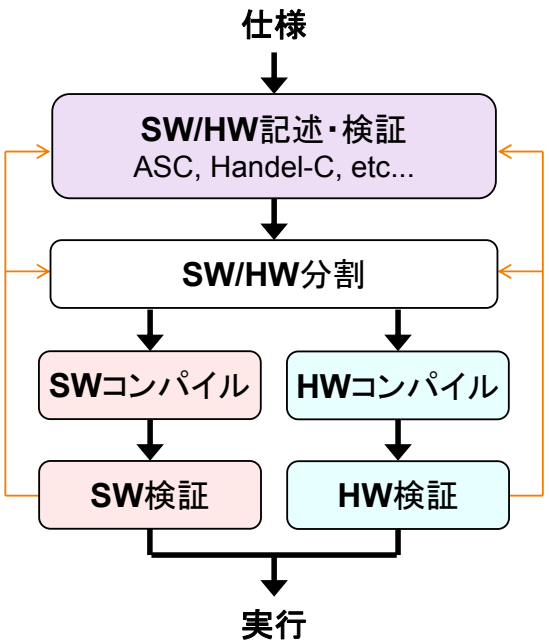


RASC (Reconfigurable Application Specific Computing) **Node**

FPGAのためのSW/HW開発の流れ

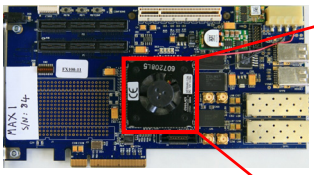


従来の方法

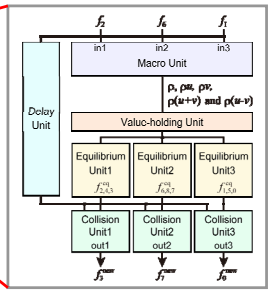


新しい方法

ストリーム型
格子ボルツマン法専用計算機



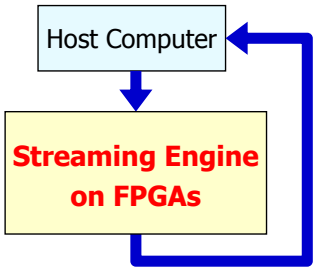
Xilinx Virtex4 FPGA x 1



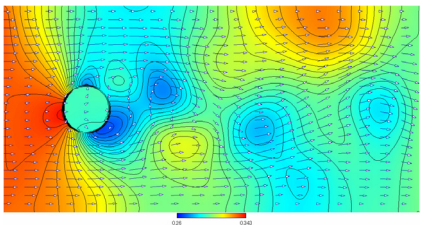
ストリーミング型プロセッサ

ストリーム型計算に基づく専用計算機

- **ストリーム処理**
  - ✓ 異なるデータに対する同様な規則的計算
  - ✓ 計算エンジンにデータ流（ストリーム）を流す
  - ✓ パイプラインにより高スループット
  - ✓ 連続アクセスにより広帯域を確保



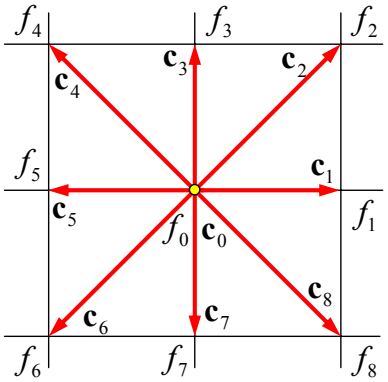
- **格子ボルツマン法による流体計算**
  - ✓ 各格子点の計算が殆ど独立 ⇒ ストリーム処理可能
  - ✓ 計算が規則的 条件分岐殆ど無 ⇒ パイプライン化に適



ストリーム処理に基づく浮動小数点格子ボルツマン法専用計算機

格子ボルツマン法(LBM, Lattice Boltzmann Method)

格子上に離散表現された仮想粒子による流体の数値計算手法



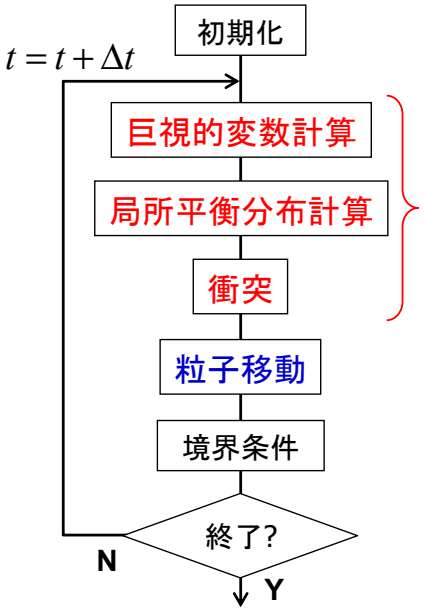
格子点上に定義される仮想粒子の離散速度分布
(9 速度; 2D9Vモデル)

i 方向 (= 0,1,...,8)

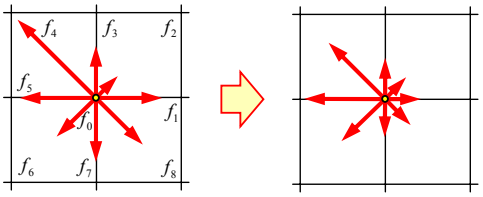
$\mathbf{c}_i$ 方向 i の粒子速度ベクトル

f_i 粒子分布関数
(速度iを持つ粒子の数密度)

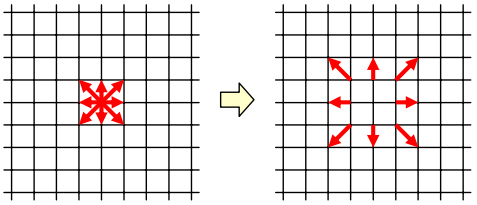
格子ボルツマン法のアルゴリズム



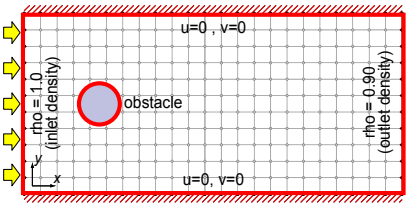
平衡状態への緩和



粒子数密度の移動



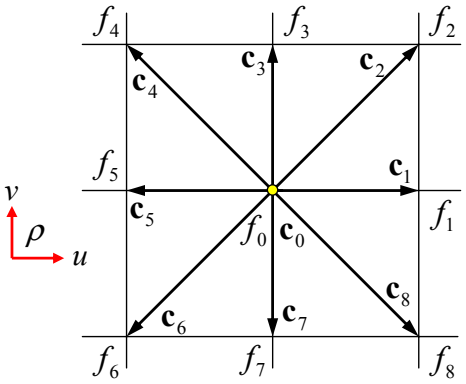
境界処理



格子ボルツマン法のアルゴリズム

巨視的変数計算

各格子点において
巨視的変数量(密度,運動量)を計算



密度

$$\rho = \sum_i f_i = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8$$

運動量 (流速: $\mathbf{v} = (u, v)$)

$$\rho \begin{pmatrix} u \\ v \end{pmatrix} = \sum_i f_i \cdot \mathbf{c}_i = \sum_i f_i \begin{pmatrix} c_{xi} \\ c_{yi} \end{pmatrix} = \begin{pmatrix} f_1 - f_3 + f_5 - f_6 - f_7 + f_8 \\ f_2 - f_4 + f_5 + f_6 - f_7 - f_8 \end{pmatrix}$$

局所平衡分布計算と衝突

- **密度,流速(ρ, $\mathbf{v} = (u, v)$)に対する局所平衡分布関数**

$$f_0^{eq} = \frac{4}{9}\rho\left[1 - \frac{3}{2}\mathbf{v}^2\right]$$

$$f_i^{eq} = \frac{1}{9}\rho\left[1 + 3(\mathbf{c}_i \cdot \mathbf{v}) + \frac{9}{2}(\mathbf{c}_i \cdot \mathbf{v})^2 - \frac{3}{2}\mathbf{v}^2\right] \qquad (i = 1, 2, 3, 4)$$

$$f_i^{eq} = \frac{1}{36}\rho\left[1 + 3(\mathbf{c}_i \cdot \mathbf{v}) + \frac{9}{2}(\mathbf{c}_i \cdot \mathbf{v})^2 - \frac{3}{2}\mathbf{v}^2\right] \qquad (i = 5, 6, 7, 8)$$

- **衝突**: 粒子分布を局所平衡状態へ近づける

$$f_i^{new} = f_i - \frac{1}{\tau}\left\{ f_i - f_i^{eq} \right\}$$

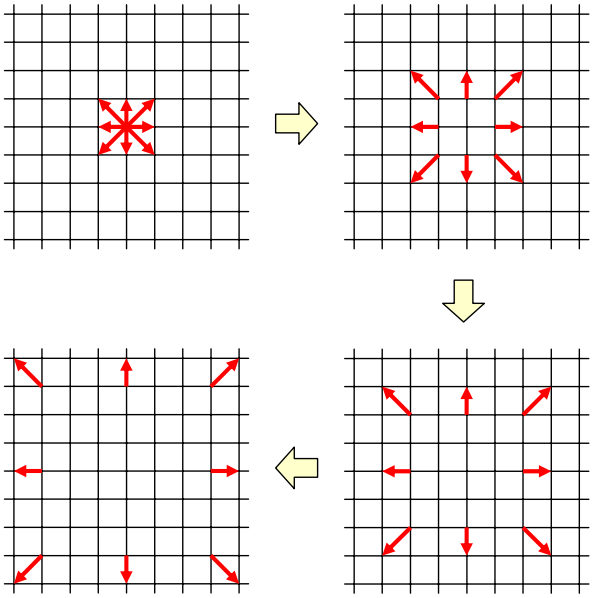
緩和係数 τ : 局所平衡状態に達する時間 $= \tau \Delta t$

粒子の移動（並進）

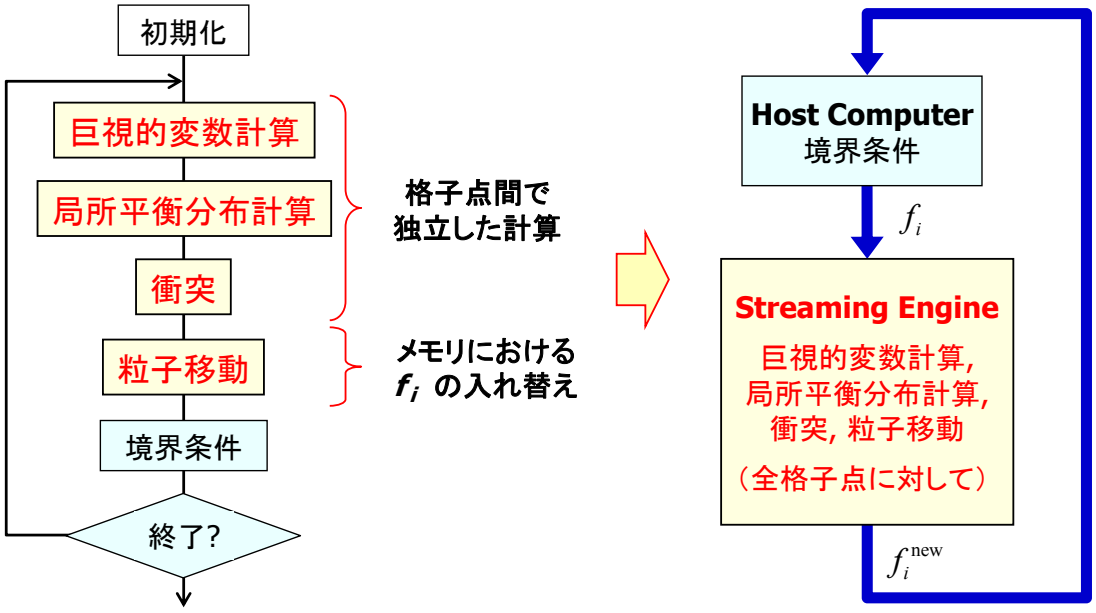
時間刻みΔtにおける粒子の移動

- ✓ 方向 i の粒子数密度 f_i
  ⇒ 方向 i の隣接格子に移動
- ✓ 各格子点における
  新たな粒子分布関数 f_i
  ⇒ 時刻 (t + Δt)における
  新しい状態

実際にはメモリ上のデータ
f_i の並べ替え



格子ボルツマン法のストリーム処理



格子ボルツマン法のアルゴリズム　　ストリーム処理

ストリーミング計算の詳細

粒子分布関数　$f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8$

巨視的変数計算

$$\rho = \sum_i f_i = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8$$

$$\begin{pmatrix} \rho u \\ \rho v \end{pmatrix} = \sum_i f_i \cdot \mathbf{c}_i = \sum_i f_i \begin{pmatrix} c_{xi} \\ c_{yi} \end{pmatrix} = \begin{pmatrix} f_1 - f_3 + f_5 - f_6 - f_7 + f_8 \\ f_2 - f_4 + f_5 + f_6 - f_7 - f_8 \end{pmatrix}$$

局所平衡分布計算

$$f_i^{\mathrm{eq}} = \mathrm{A}\rho + \mathrm{B}\rho(\mathbf{c}_i \cdot \mathbf{v}) + \mathrm{C}\rho(\mathbf{c}_i \cdot \mathbf{v})^2 - \mathrm{D}\rho\mathbf{v}^2$$

$\mathbf{c}_i$ = (0,0), (0,±1), (±1,0), or (±1, ±1)

衝突

$$f_i^{\mathrm{new}} = f_i - \mathrm{E}\{ f_i - f_i^{\mathrm{eq}} \}$$

並進

$$f_i^{\mathrm{new}}(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) = f^{\mathrm{new}}{}_i(\mathbf{x}, t)$$

メモリ上の並び替え

粒子分布関数　$f_0^{\mathrm{new}}, f_1^{\mathrm{new}}, f_2^{\mathrm{new}}, f_3^{\mathrm{new}}, f_4^{\mathrm{new}}, f_5^{\mathrm{new}}, f_6^{\mathrm{new}}, f_7^{\mathrm{new}}, f_8^{\mathrm{new}}$

計算の最適化

浮動小数点演算器の削減

巨視的変数計算

密度　$\rho = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8$

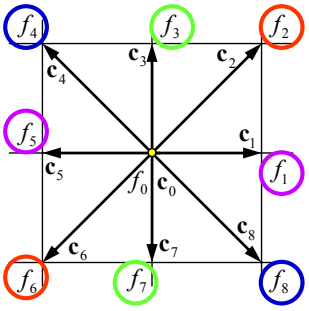
運動量

$$\rho u = (f_2 - f_6) - (f_4 - f_8) + (f_1 - f_5)$$

$$\rho v = (f_2 - f_6) + (f_4 - f_8) + (f_3 - f_7)$$

$$\rho(u+v) = (f_1 - f_5) + (f_3 - f_7) + 2(f_2 - f_6)$$

$$\rho(u-v) = (f_1 - f_5) + (f_3 - f_7) - 2(f_4 - f_8)$$



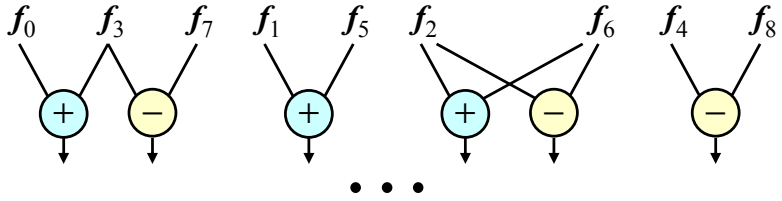
局所平衡分布計算

$$f_i^{\mathrm{eq}} = \mathrm{A}\rho + \mathrm{B}\begin{pmatrix} \rho u, \\ \rho v, \\ \rho(u+v), \text{or} \\ \rho(u-v) \end{pmatrix} + \frac{1}{\rho}\left\{ \mathrm{C}\begin{pmatrix} \rho u, \\ \rho v, \\ \rho(u+v), \text{or} \\ \rho(u-v) \end{pmatrix}^2 - \mathrm{D}\left[(\rho u)^2 + (\rho v)^2\right] \right\}$$

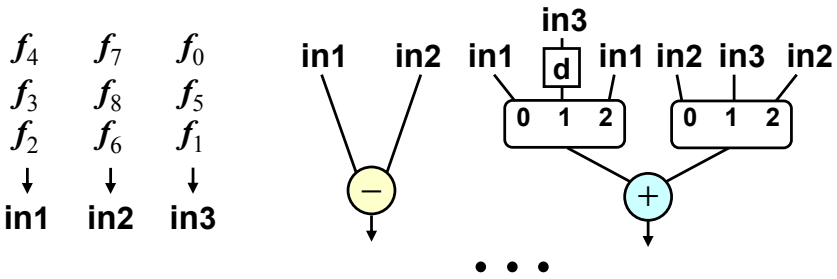
ベクトルの内積 ⇒ スカラ値の選択

同一演算器の時分割利用

9変数を同時入力 ： 空間並列 ⇒ 演算器数多 & 広帯域

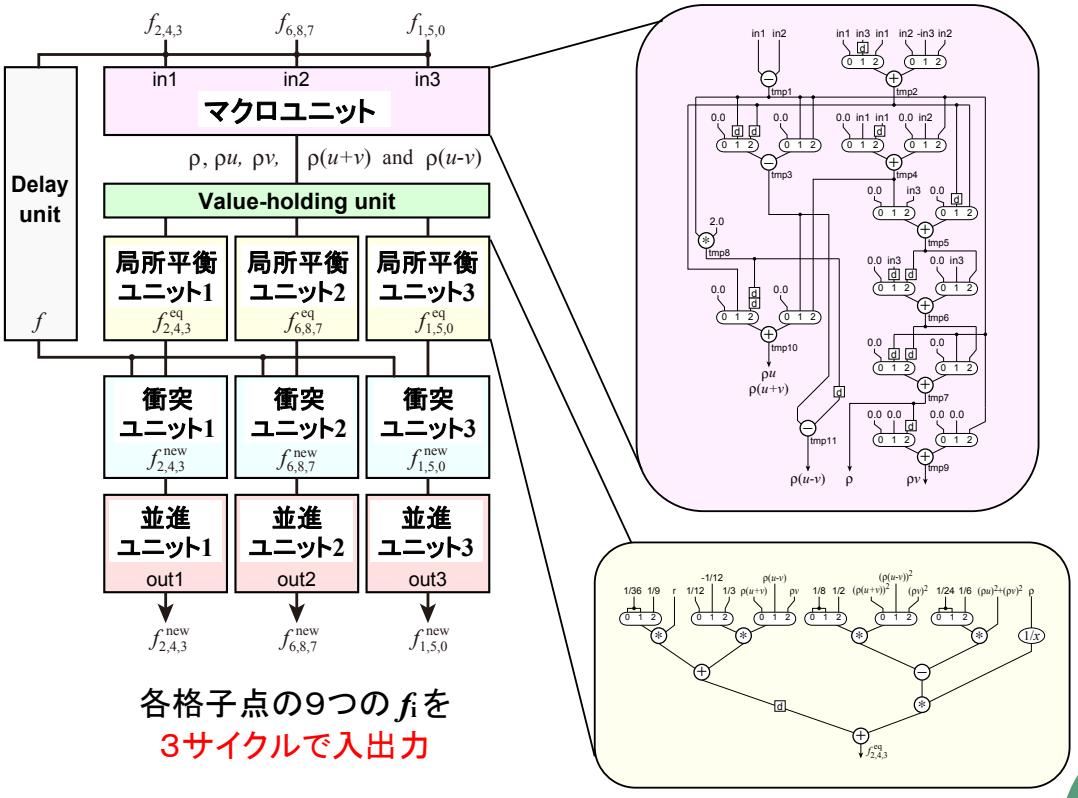


9変数を複数サイクル入力 ： 空間・時間並列 ⇒ 演算器数・帯域を削減可



実装可能演算器数, 動作周波数, I/O帯域 を考慮し **3変数3サイクル入力**

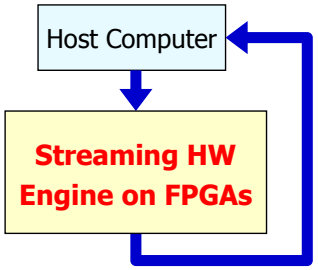
ストリーミング回路の概要



各格子点の9つの f_i を
3サイクルで入出力

ストリームコンパイラASC

- **ストリーム処理**
  - ✓ 異なるデータに対する同様な規則的計算
  - ✓ 計算エンジンにデータ流（ストリーム）を流す
  - ✓ 連続アクセスにより広帯域を確保



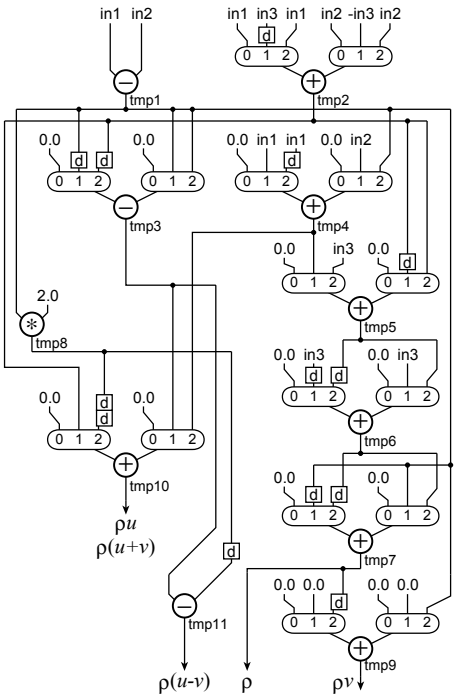
ストリームアーキテクチャ

- **ストリームコンパイラ (ASC)** [Oskar, 2006]
  - ✓ C/C++によるストリーム計算記述
  - ✓ C/C++コードからパイプライン回路を自動生成
    （演算器等設計の必要無し）

Stream processing code in C/C++ → (ASC) → Circuits for FPGA → (FPGA tool) → Stream engine on FPGA

easy like software!

ASCのためのストリーミングHW記述



マクロユニットのデータパス
（巨視的変数計算）

```
int main(int argc, char **argv) {

  STREAM_START; // ASC code start
  STREAM_OPTIMIZE=THROUGHPUT;

  HWfloat  n1(IN, HWFLOAT_TYPE);
  HWfloat  in2(IN, HWFLOAT_TYPE);
  HWfloat  in3(IN, HWFLOAT_TYPE);
  HWfloat  tmp1(TMP, HWFLOAT_TYPE);  // temporary1
  HWfloat  tmp2(TMP, HWFLOAT_TYPE);  // temporary2
  HWfloat  tmp2(TMP, HWFLOAT_TYPE);  // temporary3
  HWint  cyc (REGISTER, 3, UNSIGNED);  // for cycle switching
  cyc = 1;
  cyc = IF(cyc[0]==1, 2, IF(cyc[1]==1, 4, 1));
  // cyc=1, 2, 4  for the 1st, 2nd , 3rd cycles
  ...

  tmp1 = in1 – in2;
  tmp2 = IF(cyc[1], prev(in3, 1), in1)  +  IF(cyc[1], -in3, in2);
  tmp3 = IF(cyc[0], 0.0, IF(cyc[1], prev(tmp1, 1), prev(tmp2, 1)))
         - IF(cyc[0], 0.0, tmp1);
  ...

  STREAM_END;
  return 0;
}
```

入力変数

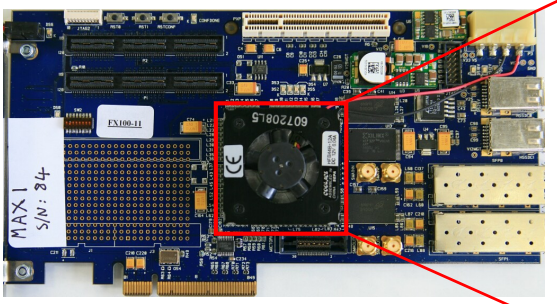
計算用テンポラリ変数

サイクル切り替え用カウンタ

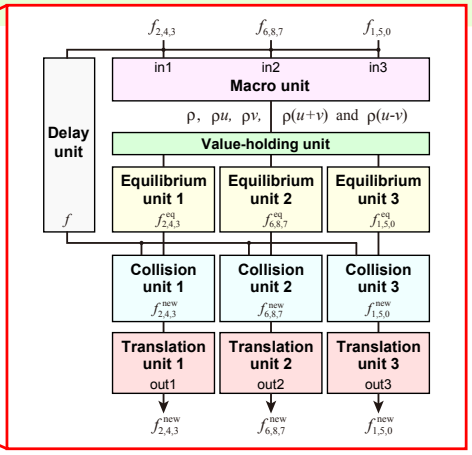
条件演算子 IF(条件, 真の値, 偽の値)

ASCクラスライブラリを用いたC++コード

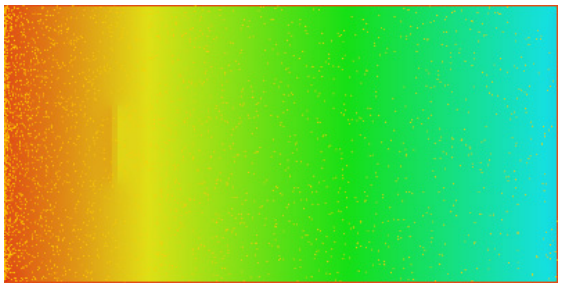
実装と性能評価



Xilinx Virtex4 FPGA x 1



LBMストリーミング回路
125MHz動作のパイプライン



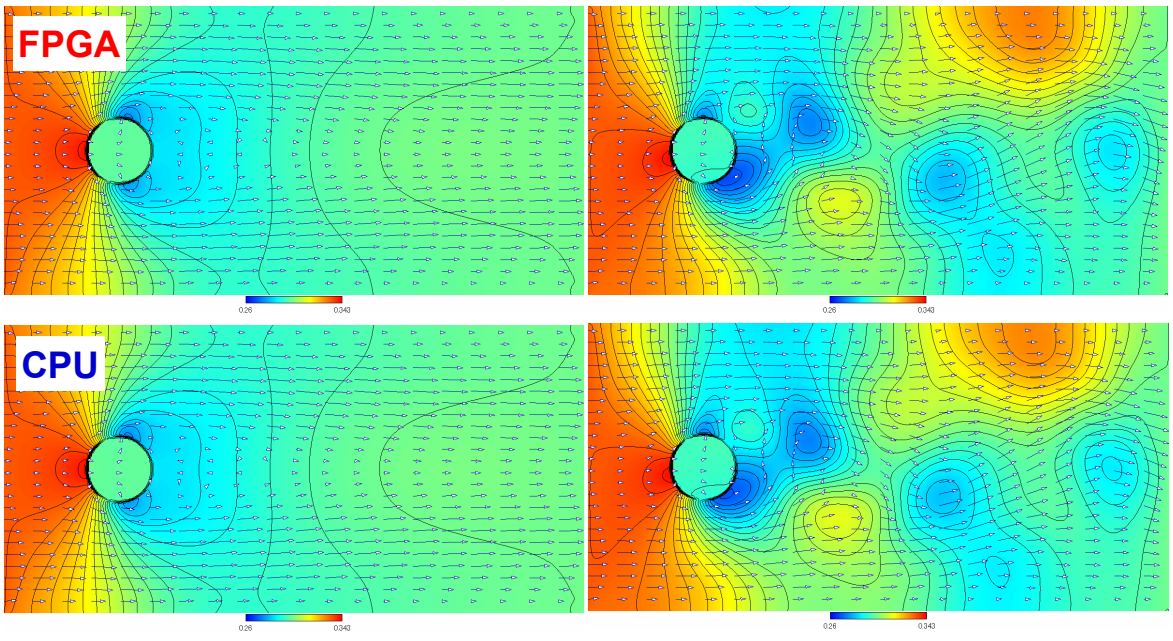
2次元非定常流計算

浮動小数点演算器
（単精度相当31bit: 指数8bit, 仮数22bit）

加算器　27　　乗算器　17

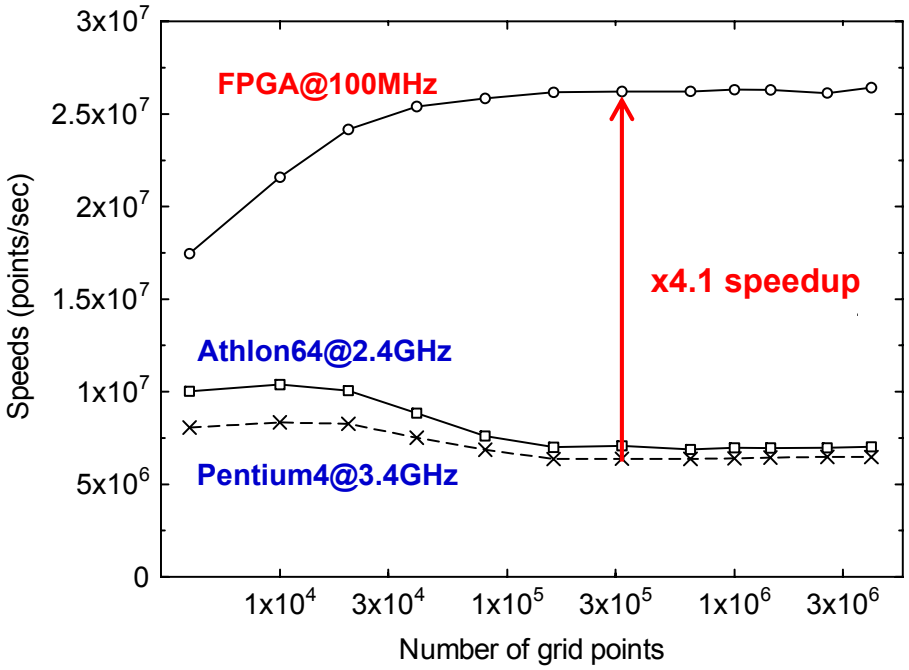
除算器　1

計算結果の比較 (Circle, RE=300)



T=10000　　T=50000

問題サイズと速度向上率

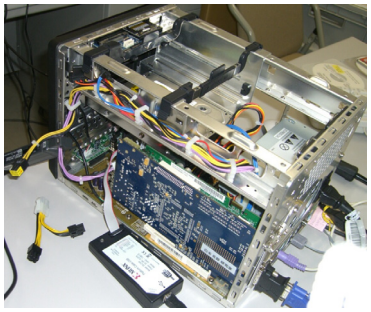


現状ではPCI-Express x8の半分の帯域（1.8GB） ⇒ 性能向上の余地有り

電力・エネルギ評価（AC）

- **使用機器**
  メモリハイコーダ　HIOKI 8855
- **計測方法**
  ATX電源 AC 100Vの電圧・電流を測定
  ⇒　電力算出
- **条件**
  ホストPC
  CPU　　Athlon64 400+ 2.4GHz
  FPGA　　ストリーム型LBM専用計算機

  格子ボルツマン法（2D19V）計算
  400x200格子, 3200タイムステップ
  1600x800格子, 800タイムステップ



ACの電圧・電流計測クリップ

平均消費電力・総エネルギの比較

400x200計算の消費電力

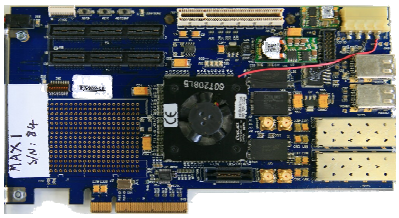
		FPGA(対CPU*比)
400x200 3200 steps	平均電力 (W)	9割程度
	エネルギ (J)	3割程度
1600x800 800 steps	平均電力 (W)	9割程度
	エネルギ (J)	3割程度

* Athlon64 2.4GHz　　** システム全体のAC電源

ストリーム型LBM専用計算機のまとめ

ストリーム型の計算 ⇒ 高スループット & 広帯域の連続メモリアクセス

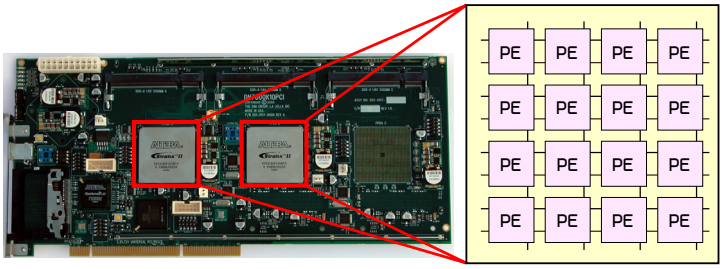
- **格子ボルツマン法による流体力学計算機**
  - ✓ 並列性 & 規則性　⇒ パイプライン
  - ✓ 演算器の時分割利用　⇒ 省HW使用, I/O帯域と計算速度のバランス
- **FPGAによる実装**
  - ✓ ASCによるC++コード ⇒ パイプライン自動生成
  - ✓ 45の単精度浮動小数点演算器, 125MHz動作
  - ✓ PCI-Express x8のバンド幅がネック
    しかし、CPUとの協調計算が可
  - ✓ 3〜4倍の高速計算, 低消費電力・低エネルギ



Kentaro Sano, Oskar Mencer and Wayne Luk, "FPGA-based Acceleration of the Lattice Boltzmann Method," Procs of the International Conference on Parallel Computational Fluid Dynamics (ParCFD2007), CDROM(#041, 5 pages), May, 2007.

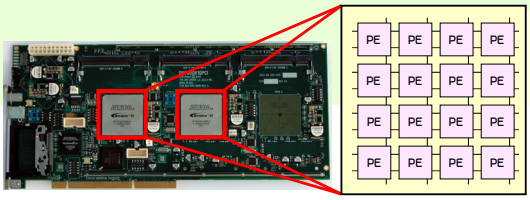
Kentaro Sano, Oliver Pell, Wayne Luk and Satoru Yamamoto, "FPGA-based Streaming Computation for Lattice Boltzmann Method," Procs of the International Conference on Field-Programmable Technology (ICFPT2007), pp.233-236, Dec, 2007.

シストリックアレイ型差分法専用計算機



ALTERA Stratix-II FPGA　　アレイ型プロセッサ

設計コンセプト

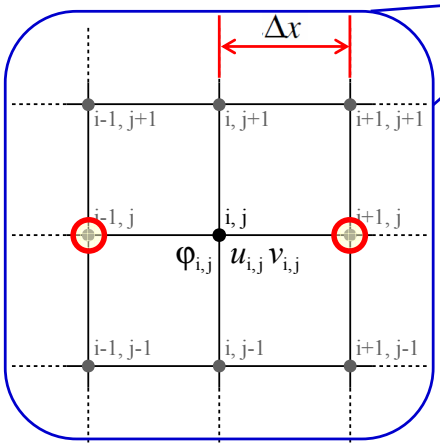


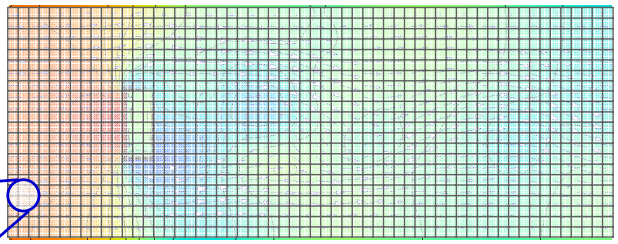
- **あるカテゴリの計算問題**
- **再コンフィギュレーションせずとも可変の計算**
  - ✓ プログラマブルの基本構造 & 計算問題に特化した構造
- **計算全体の単独実行**
  - ✓ ホスト・FPGA間の帯域制約を回避
  - ✓ FPGA側のみで拡張可能 ⇒ ホスト & スケーラブル大規模FPGAシステム

今回の設計・実装

✓ 対象問題	差分法計算 （流体計算, 伝熱計算, 電磁波計算, etc）
✓ アーキテクチャ	シストリックアレイ, 計算メモリ, 簡素な計算要素
	（FPGA内蔵メモリのみを使用 ⇒ FPGAの最大性能）
✓ 拡張性	LVDSによるFPGA間通信
	（FPGAアレイのスケーラビリティを実証）

対象問題： 差分法計算





例）差分法に基づく数値流体力学計算

$$\frac{du}{dx} \simeq \frac{u_{i+1} - u_{i-1}}{2\Delta x}$$

$$\frac{d^2u}{dx^2} \simeq \frac{u_{i-1} - 2u_i + u_{i+1}}{(\Delta x)^2}$$

2次元・3次元直交格子
⇒ 規則性

中心差分法による近似
⇒ 局所性・並列性

計算問題の例： フラクショナルステップ法

[Kim et. al, 1985]

- **非圧縮粘性流体の代表的計算手法**

$$\nabla V = 0$$ 連続の式

$$\frac{\partial V}{\partial t} + (V \cdot \nabla)V = -\nabla\varphi + \nu\nabla^2 V$$ 運動方程式

V 速度 $= (u, v)$
P 圧力
ρ 密度
$\nu \equiv \mu/\rho$ 動粘性係数
$\varphi \equiv P/\rho$

- **計算手順**

1. 圧力を考慮せずに運動方程式の時間微分項から仮の速度 V^* を求める

$$V^* = V^n + \Delta t\left\{-(V^n \cdot \nabla)V^n + \nu\nabla^2 V^n\right\} \quad (1)$$

2. 反復法等によりポアソン方程式を解き、仮の速度V^* から圧力 ϕ^{n+1} を求める

$$\nabla^2\varphi^{n+1} = \frac{\nabla \cdot V^*}{\Delta t} \quad (2)$$

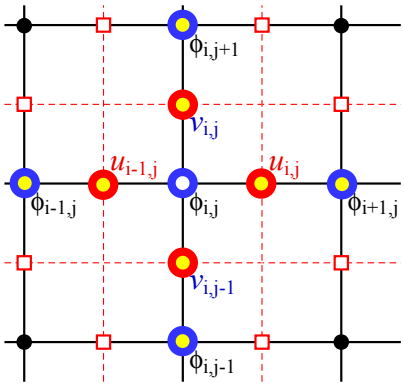
3. V^* とϕから運動方程式より次の時間ステップの速度 V^{n+1} を求める

$$V^{n+1} = V^* - \Delta t\nabla\varphi^{n+1} \quad (3)$$

ヤコビ法によるポワソン方程式の反復解法

手順2のポワソン方程式

$$\nabla^2 \varphi^{n+1} = \frac{\nabla \cdot V^*}{\Delta t} \qquad (\ V^* = (u^*, v^*)\ )$$



スタガード（食違い）格子

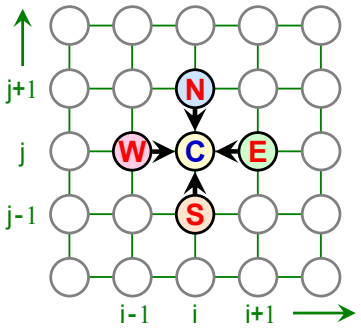
中心差分による離散化

$$\varphi'_{i,j} = \alpha \left(\frac{\varphi_{i+1,j} + \varphi_{i-1,j}}{\Delta x^2} + \frac{\varphi_{i,j+1} + \varphi_{i,j-1}}{\Delta y^2} - D_{i,j} \right)$$

ただし

$$\alpha = \frac{\Delta x^2 \Delta y^2}{2(\Delta x^2 + \Delta y^2)} \qquad D_{i,j} = \frac{1}{\Delta t} \left(\frac{u^*_{i,j} - u^*_{i-1,j}}{\Delta x} + \frac{v^*_{i,j} - v^*_{i,j-1}}{\Delta y} \right)$$

差分法に基づく計算における一般式



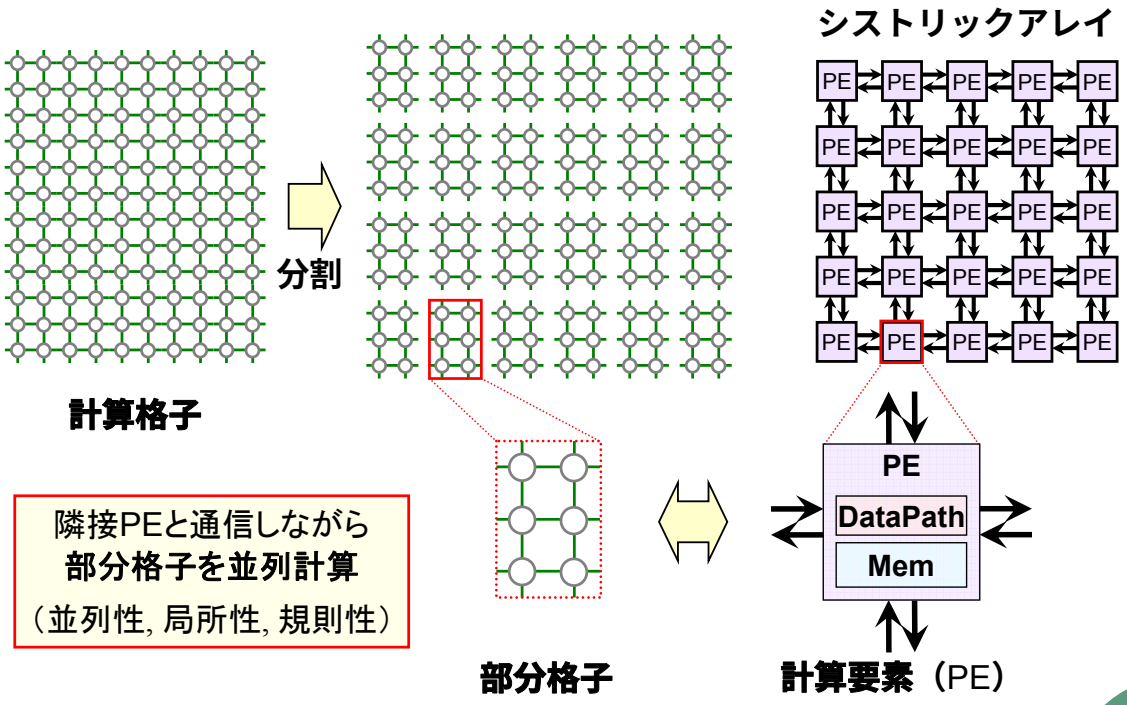
一般式

$$Q^{n+1}_{i,j} = C_0 + C_C\, Q^{n}_{i,j} + C_E\, Q^{n}_{i+1,j} + C_W\, Q^{n}_{i-1,j} + C_S\, Q^{n}_{i,j-1} + C_N\, Q^{n}_{i,j+1}$$

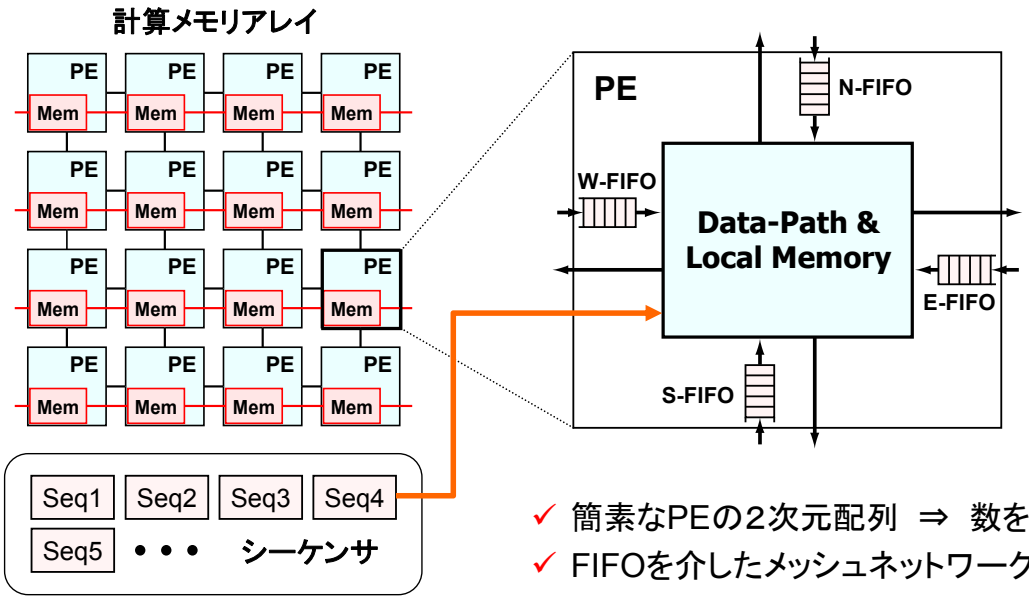
$Q_{i,j}$ 各格子点の変数　　C_* 各格子点(i,j)のみで求まる係数

⇒ 各格子点の累算を並行して行うためのアーキテクチャ

データ並列計算モデルとシストリックアーキテクチャ

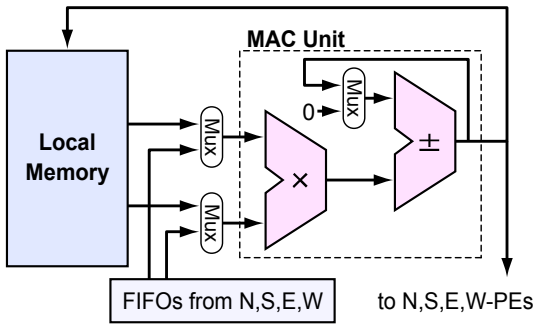


シストリックアレイ型差分法専用計算機



- ✓ 簡素なPEの2次元配列 ⇒ 数を稼ぐ
- ✓ FIFOを介したメッシュネットワーク
- ✓ 複数のシーケンサ
  各シーケンサ ⇒ 複数のPEを制御
  （境界条件により異なる計算用）

計算要素(PE)のデータパスと命令セット



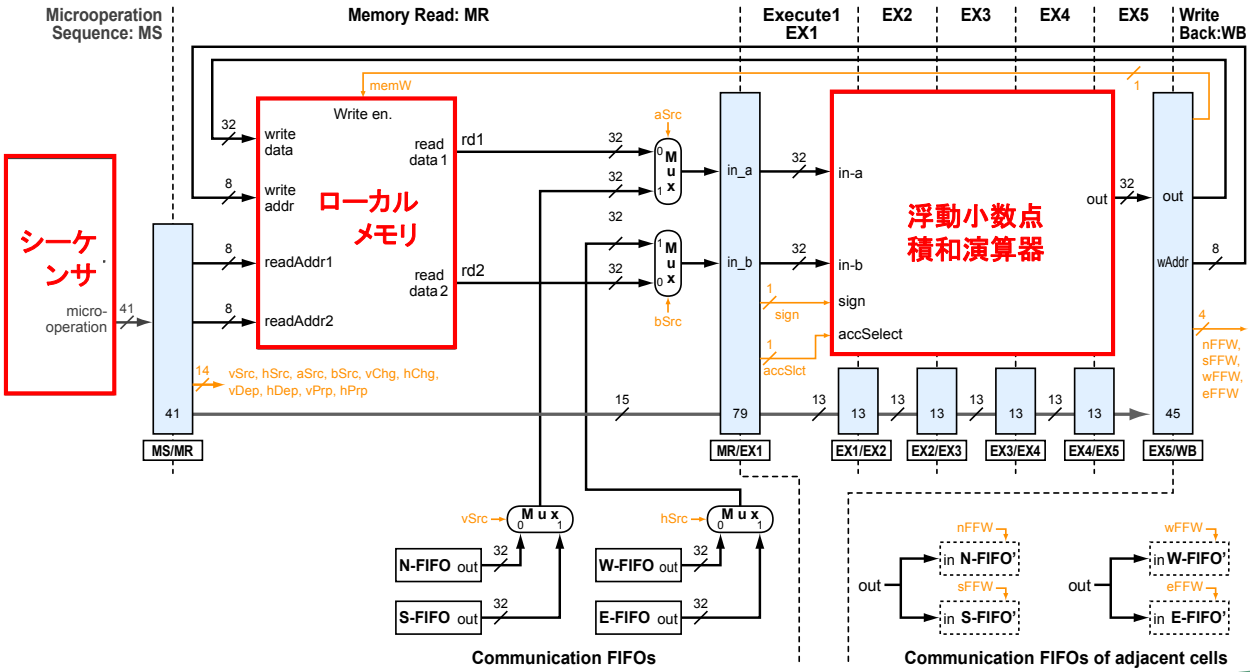
PEのデータパスの概要

- ✓ シンプルな演算命令
- ✓ メモリ・FIFOの参照
- ✓ 隣接PEへの演算結果書込み
- ✓ ループ専用命令

※あくまで固定機能の時分割実装が目的(必要に応じて拡張)

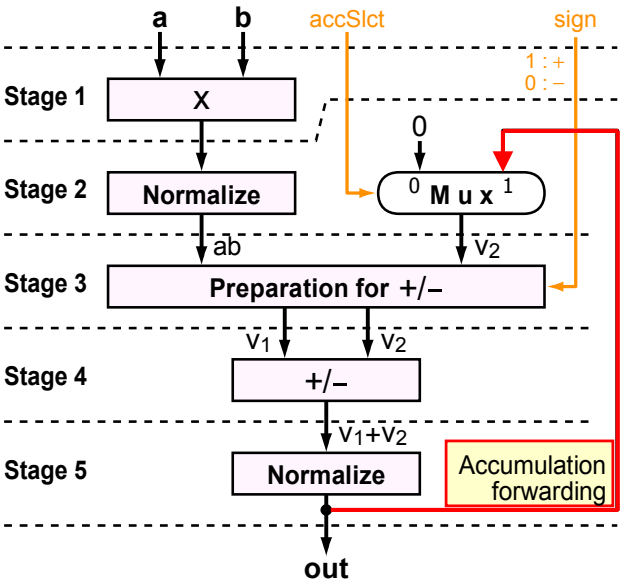
```
//      Op.     dst1, dst2,     src1,       src2
        mulp    NE,   -,        u[0;0],     one
        mulp    E,    -,        u[1;0],     one
        mulp    SE,   -,        u[2;0],     one
        mulp    NW,   -,        u[0;1],     one
        mulp    W,    -,        u[1;1],     one
        ...                                  ← W-PEへの送信

// Main Loop Begin
        lset    500,  Main      ← ループ専用命令

Main:   mulp    -,    -,        c,          one
        mulp    -,    -,        c,          one
        mulp    -,    -,        c,          one
        accp    -,    B[0;0],   u[0;0],     b
        accp    -,    B[1;0],   u[1;0],     b
        accp    -,    B[2;0],   u[2;0],     b
        mulp    -,    -,        c,          one
        mulp    -,    -,        c,          one
        mulp    -,    -,        c,          one
        accp    -,    B[0;1],   u[0;1],     b
        accp    -,    B[1;1],   u[1;1],     b
        accp    -,    B[2;1],   u[2;1],     b
        ...
        accp    NW,   v[0;1],   SFIFO,      b
        accpbne W,    v[1;1],   ph[0;1],    b
        accp    SW,   v[2;1],   ph[1;1],    b
```


計算要素のデータパス(8段パイプライン)



隣接セルとの通信FIFO

積和演算器（5段パイプライン）

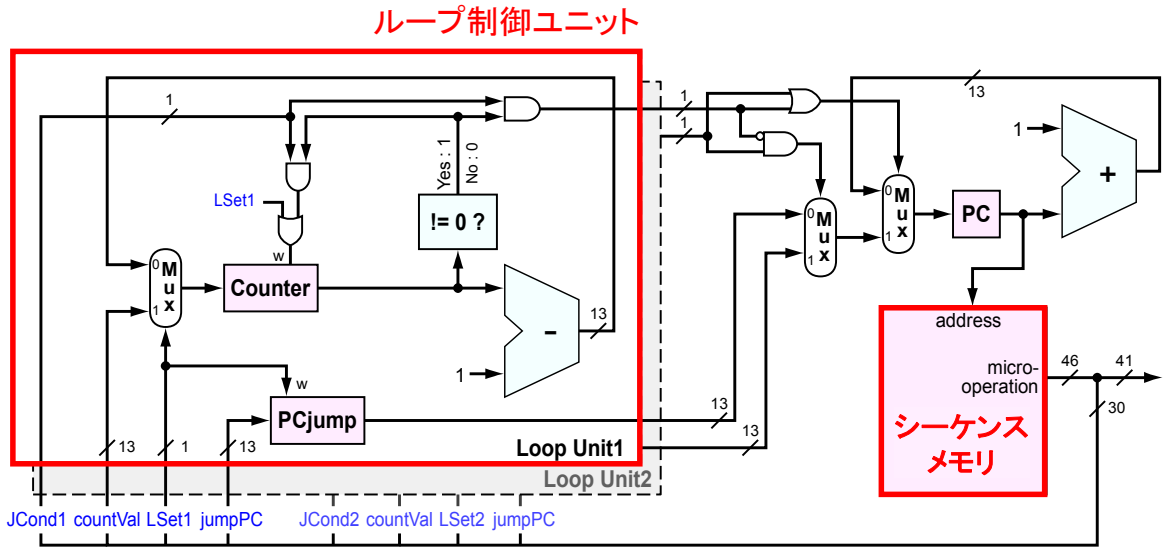


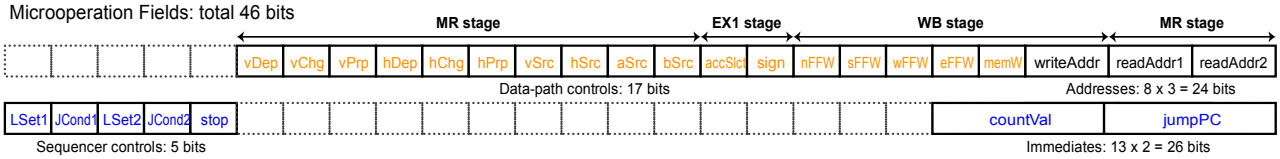
3ステージ前への
フォワーディング

⇒ 3サイクル毎の
入力を累算可能

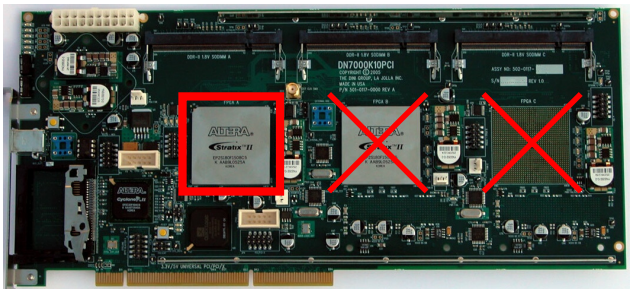
$$OUT = a_1b_1 + a_2b_2 + a_3b_3 + a_4b_4 + \ldots$$

シーケンサ

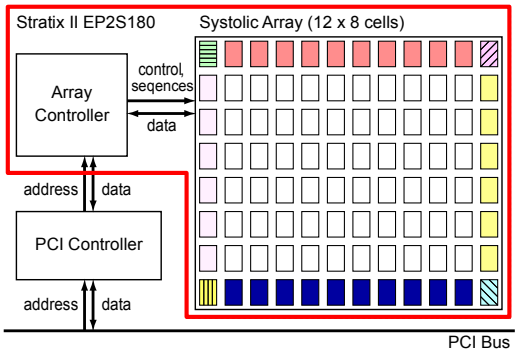




単一FPGAによる試作実装



FPGAボード: DN7000k10PCI



12 x 8 PE 計算メモリアレイ

ALTERA StratixII FPGA

- ✓ 最大規模 (EP2S180)
- ✓ 143520 ALUTs
- ✓ 8.9 Mbits のBlockRAM
- ✓ 96 36x36-乗算器

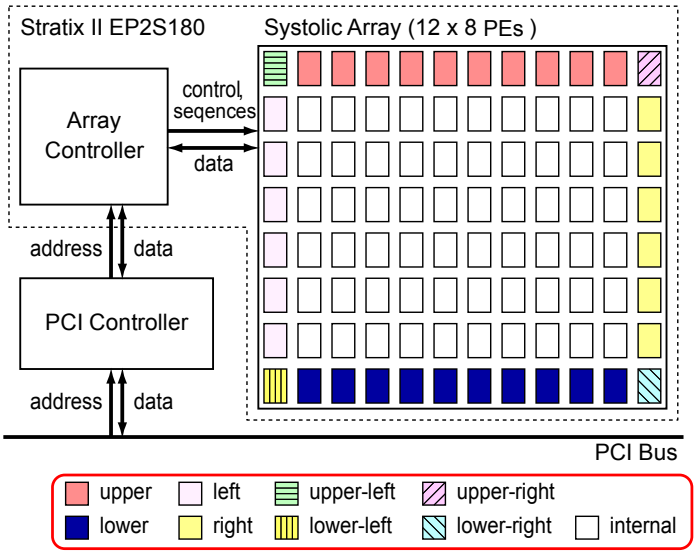
実装結果（verilog-HDL）

PE数　96
動作周波数　96 MHz
ピーク性能　18.4 GFlops（単精度）

PCIコントローラ, アレイコントローラ
シーケンサ x 9 を実装

境界PEと9つのシーケンサ

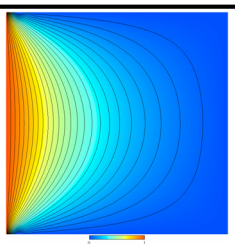
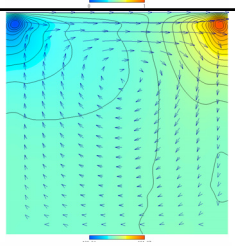
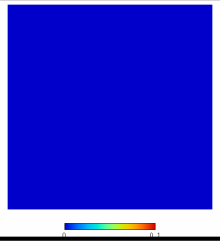
異なる境界条件
⇩
異なる計算
⇩
異なるシーケンサ



9シーケンサの内訳

内部	60 PE	internal
縁	4 PE	right, left, upper, lower
角	4 PE	upper-left, lower-left, upper-right, upper-left

計算速度 （単一FPGA 96MHz）

計算問題		実効性能（効率）	Pentium4 3.2GHz比
ラプラス方程式の数値解法（ヤコビ法）		15.1 GFlops （81.75%）	x 11.15
正方キャビティ強制対流（フラクショナルステップ法）		16.5 GFlops （89.45%）	x 14.12
電磁波伝播（FDTD法）		15.1 GFlops （81.8%）	x 12.26

電力・エネルギ評価（DC）

- **使用機器**
  メモリハイコーダ HIOKI 8855
- **計測方法**
  ATX電源の電圧・電流を測定
  （**DC：+3.3V, +5V, +12V**） ⇒ DC電力算出
- **条件**
  ホストPC （HP ML310）
  CPU **Celeron 2.66GHz**
  FPGA 単一FPGA, 96MHz

DC電源測定のクリップ等

計算問題
ヤコビ法計算 単精度, 2D格子(96x96), 3.3×10^5反復
FDTD法計算 単精度, 2D格子(72x72), 8.0×10^5タイムステップ

平均消費電力・総エネルギの比較

		FPGA**（対CPU*比）
ヤコビ法計算 96x96 3.3x10^5 反復	平均電力	70% 程度
	エネルギ	5% 程度
FDTD法計算 72x72 8.0x10^5 steps	平均電力	70% 程度
	エネルギ	6% 程度

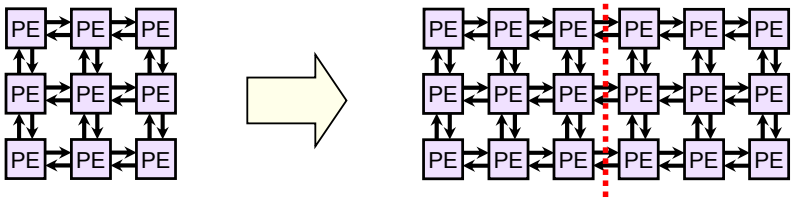
* Celeron 2.66GHz

** HDD, 電源ファンを除くシステム全体のDC電源

アレイサイズに対するスケーラビリティ

PE間の通信量はアレイサイズに無関係

⇒ **PE間要求帯域一定のままアレイを拡張可能**



計算速度 ∝ PE数

局所メモリの総容量∝ PE数

局所メモリの総帯域∝ PE数

規模に対してスケーラブルなシステム

差分法専用計算機のまとめ

差分法計算 ⇒ 規則性・局所性・並列性 （累算計算）

- **シストリックアレイによる専用計算機**
  - ✓ プログラマブルの基本構造 & 問題に特化した構造
    （簡素なPE, ネットワーク, シーケンサ）

- **FPGAによる実装**
  - ✓ FPGA単独で全計算を実行
  - ✓ 96の単精度浮動小数点演算器, 96MHz動作
  - ✓ 10～倍の高速計算, 低消費電力・低エネルギ
  - ✓ オンチップメモリの限界 ⇒ 外部メモリ利用／FPGAアレイ構築の予定

Kentaro Sano, Takanori Iizuka and Satoru Yamamoto, "Massively Parallel Processor based on Systolic Architecture for High-Performance Computation of Difference Schemes," Proceedings of the International Conference on Parallel Computational Fluid Dynamics (ParCFD2006), pp.174-177, May, 2006.

Kentaro Sano, Takanori Iizuka and Satoru Yamamoto, "Systolic Architecture for Computational Fluid Dynamics on FPGAs," Proceedings of the 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM2007), pp.107-116, April, 2007.

佐野 健太郎, 王 陸洲, 初田 義明, 山本 悟, “FPGAによる数値流体力学専用計算の設計と評価,” 第21回数値流体力学シンポジウム講演論文集, CDROM(No.G6-2, 4 pages), 2007.

総括

- **大規模FPGAによる高性能計算**
  - ✓ 潜在性能（現在）
    - ◎ ビット・整数演算, 他精度(4倍精度?)
    - ○ 単精度
    - △ 倍精度
  - ✓ 低消費電力
  - ✓ 複数のFPGAによるスケーラブルなシステム
- **アプリケーションの選択が鍵**
  - ✓ HW化に適した問題　　並列性・規則性
  - ✓ 実はデータ入替え等の計算以外の処理で優位 （μPが苦手とする処理）
- **今後の課題**
  - ✓ FPGAの潜在的性能向上を具現化するために
    - ➢ アプリケーション ⇒ HWのマッピング（アーキテクチャ）
      I/O帯域, 内部メモリ, 演算性能のバランス
  - ✓ 設計・実装の容易化, 規格化