FPGAにおける計算科学アプリケーションの部分オフローディング

藤田 典久[1]

研究協力: 大畠 佑真[2], 小林 諒平[1,2], 朴 泰祐[1,2]

1: 筑波大学 計算科学研究センター

2: 筑波大学 システム情報工学研究科

背景と目的

- PPX, PACS-Xに向けて、FPGAを用いるアプリケーション開発手法の検討が必要

- FPGAにおける従来の開発手法では、HPCアプリケーション on FPGAを開発することが難しい
  - カーネル開発にかかるコスト
  - 計算カーネル以外の回路開発が必要 (メモリ・PCIeなど)
  - FPGAにおける専門知識が必要

背景と目的

- Altera社が提供しているOpenCL to FPGAコンパイラを利用
  - OpenCLを用いた場合のFPGAの基礎評価を行う
  - 記述性、開発効率、実効性能
  - どのような要素が最適化をする上で重要なのかを明らかにする

- 将来的にはAccelerator in SwitchをOpenCLで記述したい
  - 計算部の性能 (Accelerator)
  - 加えて、通信機構に対する操作をOpenCLで記述できるか (Switch)

HDLとHLS

- Hardware Description Language (HDL)
  - Verilog
  - VHDL
- 1クロック、1ビット単位の動作を記述
- 大規模な回路を作成する場合の開発コストが問題
- ソフトの世界におけるアセンブラに相当

HDLとHLS

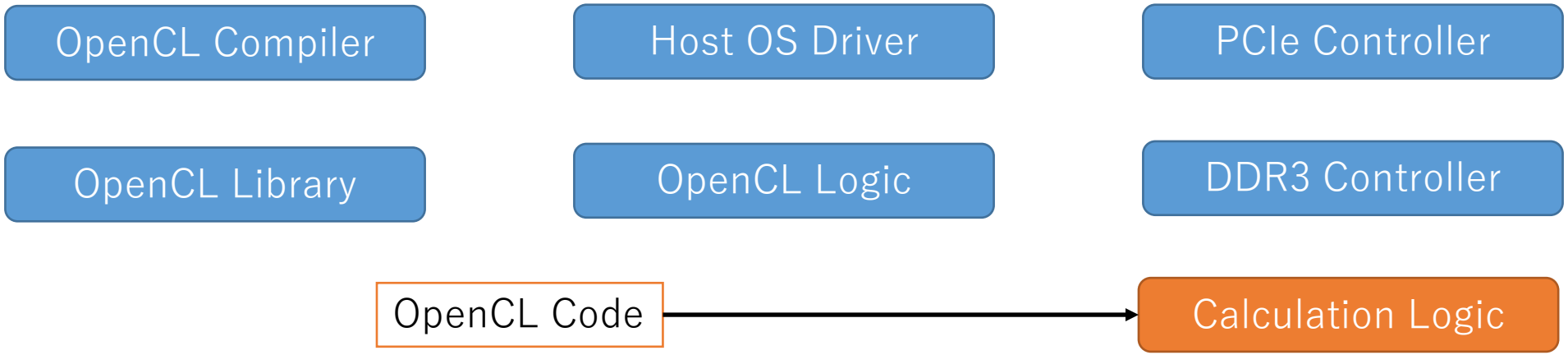
- High Level Synthesis (HLS, 高位合成)
  - Altera OpenCL
  - Xilinx Vivado HLS (C)
- ソフトウエア開発で用いられている言語でFPGAプログラミングが可能
- 大規模な開発にも耐えられる
- 高位言語→HDL→FPGA回路

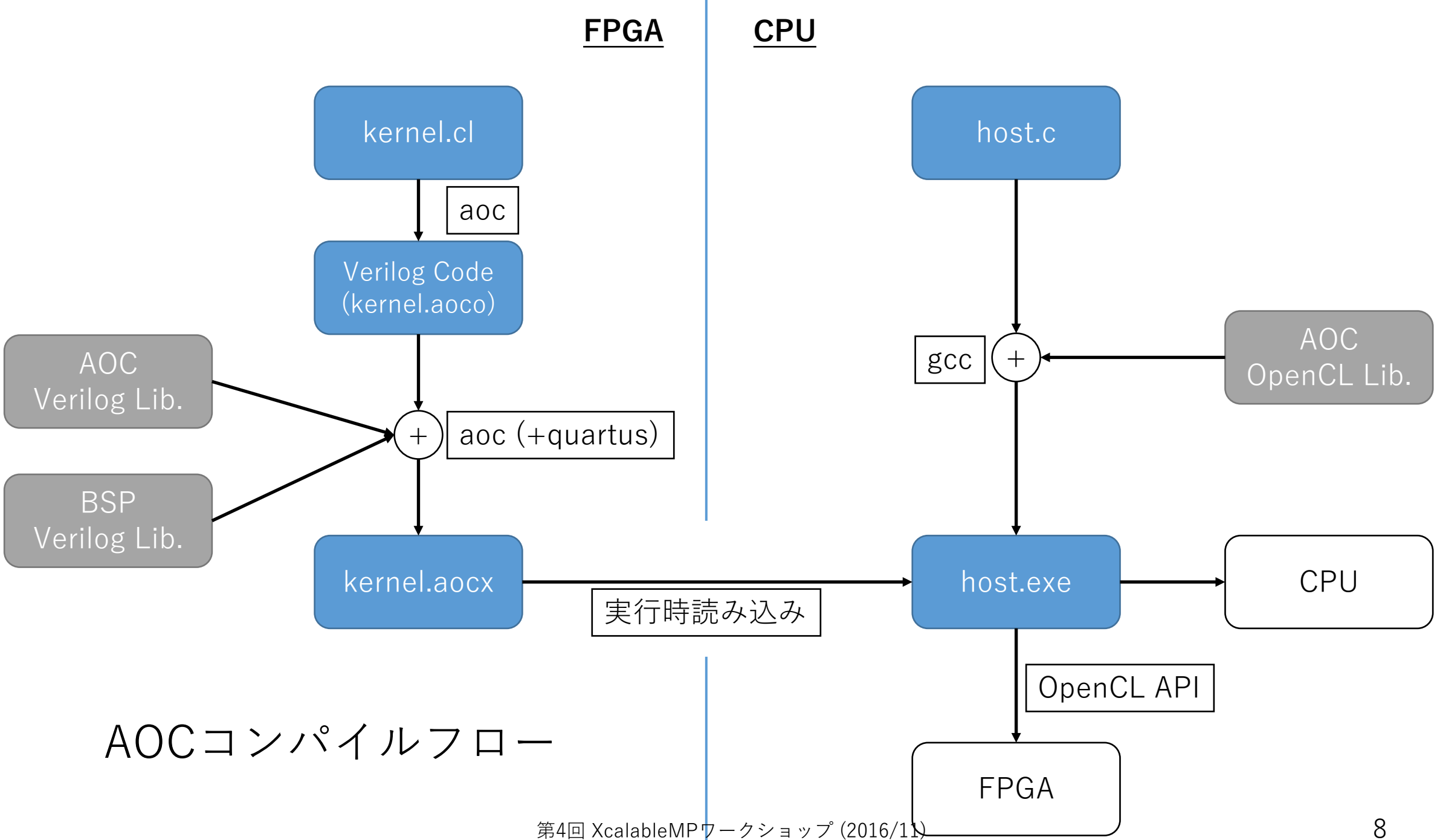
Altera Offline Compiler (AOC)

- Altera社が自社FPGA向けに提供しているOpenCLコンパイラ
- OpenCLコードを元にして、FPGAの回路を生成可能
- FPGAの回路だけでなく、制御用のドライバやライブラリも提供される
  - ハードだけでなく、ソフト環境も一式あり、AOC開発環境のみでFPGAの利用を開始できる
- ARM Linux, x86-64 Linux, Windows向けに開発環境が提供されている
  - ARM向けはARM CPUとAltera FPGAを同一チップにしたSoC型製品があるため

Altera Offline Compiler (AOC)

- FPGAをアクセラレータとして利用するための環境一式が構築できる
- OpenCLの標準APIを用いて、ホストCPUからFPGAを制御可能





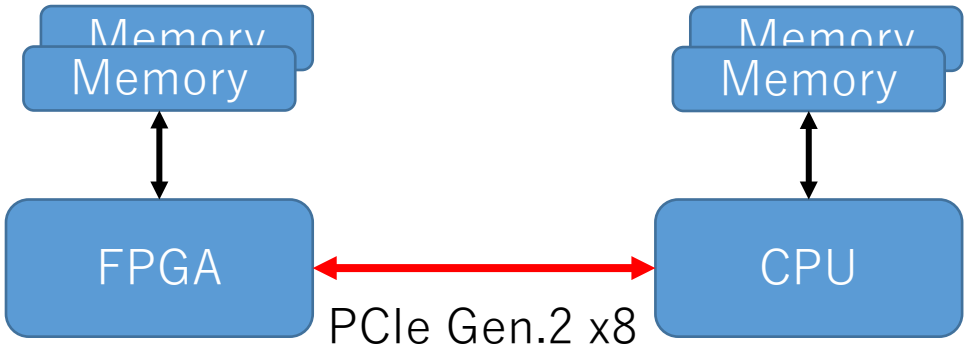
AOCコンパイルフロー

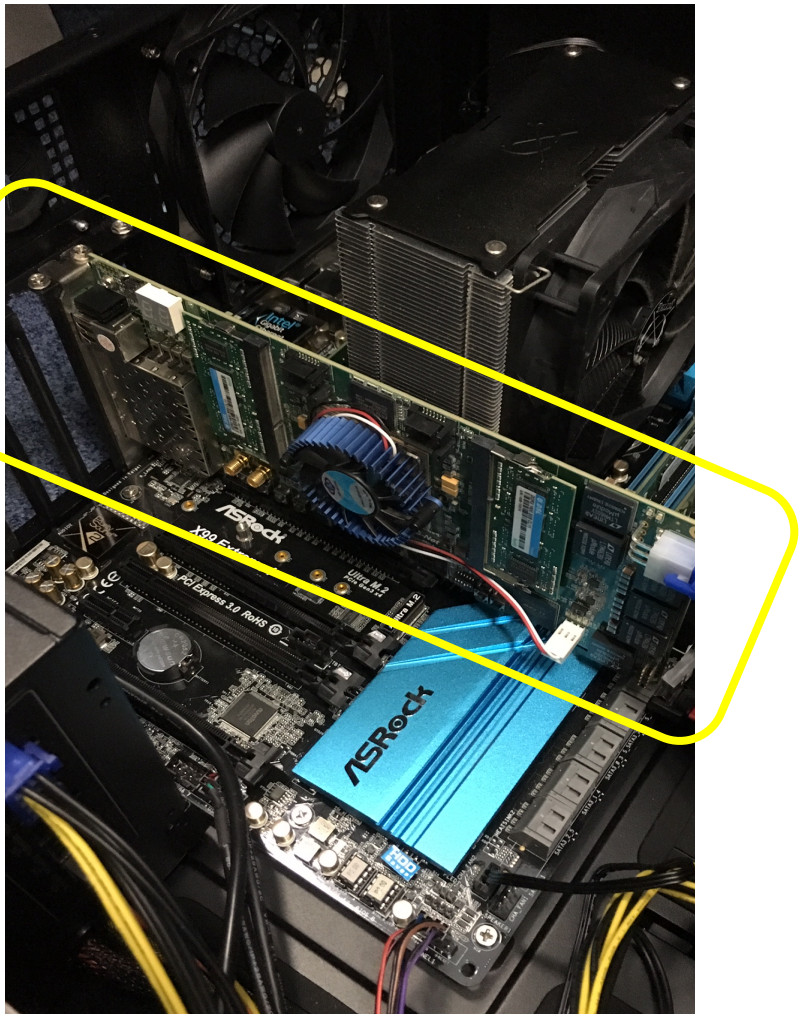
AOC Board Support Package (BSP)

- CPUやGPUとは異なり、FPGAチップがOpenCLに対応しているだけではOpenCLは使えない
  - FPGAの場合、ボード毎に構成が違う (周辺回路・配線・メモリなど)
- OpenCL用のBoard Support Package (BSP)が必要
  - 周辺回路のHDL、ボード固有の各種パラメータ、Linux/Windowsドライバ
- BSPが提供されているボードを購入するのが一般的
  - Terasic社, Bittware社など

Terasic DE5-Net FPGAボード

Terasic DE5-Net	
FPGA Chip	Altera Stratix V 5SGXEA7N2F45C2
FPGA External Memory	DDR3 SDRAM 1GB x2, 1600MHz (25.6GB/s)
PCIe (to Host)	PCIe Gen.2 x8





OpenCLコード例

- STREAM Triadコード
- 2パターンの実装例
- 上:シングルスレッドのような実装
  - single work-item
  - ループをアンロール
- 下: GPUのような実装
  - ループアンロールと同じような効果
  - カーネル回路が8個複製されるわけではない

```
__kernel void triad(
    __global float* restrict c,
    __global float const* restrict a,
    __global float const* restrict b,
    float scalar,
    uint n) {
  #pragma unroll 4
  for (uint i = 0; i < n; i++)
    c[i] = a[i] + scalar * b[i];
```

```
__kernel__
__attribute__((reqd_work_group_size(8,1,1)))
void triad(
    __global float* restrict c,
    __global float const* restrict a,
    __global float const* restrict b,
    float scalar,
    uint n) {
  uint i = get_global_id(0);
  c[i] = a[i] + scalar * b[i];
}
```

AOC/OpenCLまとめ

- FPGA/HDLの知識が全くなしにFPGAを利用可能
  - 高い記述性
  - aocコマンドによって、OpenCLコードからFPGAに書き込み可能な形式まで自動で生成される
  - ボード毎の対応はBSPによってなされる
- OpenCL APIを用いたFPGA制御ができる
  - GPUの同様なアクセラレータモデル
- FPGAへの回路データのロード(program)もOpenCLからできる
  - clCreateProgramWithBinary関数でaocxファイルを読み込む

CG法(conjugate gradient method)

- 対称正定値行列を係数とする連立一次方程式を解くためのアルゴリズム
- CG法の処理時間の大部分を右図4行目のSpMV計算(Sparse Matrix Vector)が占める
- 疎行列の圧縮にはCRS(Compressed Row Storage)を使用

```
1.   r = b − Ax
2.   p = r
3.  loop
4.     ap = Ap
5.     rr1 =  r^t ・ r
6.     pap = p^t ・ ap
7.     alpha = rr1 / pap
8.     x += alpha * p
9.     r -= alpha * ap
10.    If(|r| < e) break
11.    rr2 = r^t ・ r
12.    beta = rr2 / rr1
13.    p = r + beta * p
14. end
```

$$\begin{pmatrix} 5 & 0 & 0 & -1 & 0 \\ 0 & 3 & 9 & 0 & 0 \\ 0 & 0 & 1 & 0 & 8 \\ 0 & -2 & 0 & 5 & 0 \\ -3 & 0 & 0 & 2 & -1 \end{pmatrix} \xrightarrow{\text{CRS}}$$

VAL = {5 -1 3 9 1 8 -2 5 -3 2 -1}

COL_IND = {0 3 1 2 2 4 1 3 0 3 4}

ROW_PTR = {0 2 4 6 8 11}

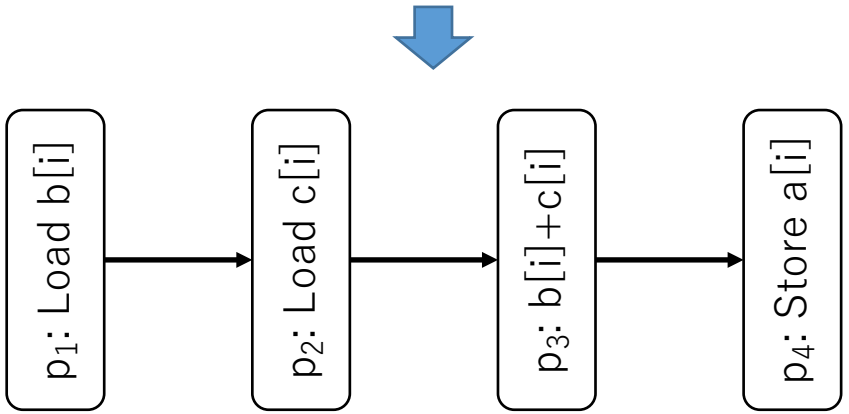
FPGA向け最適化

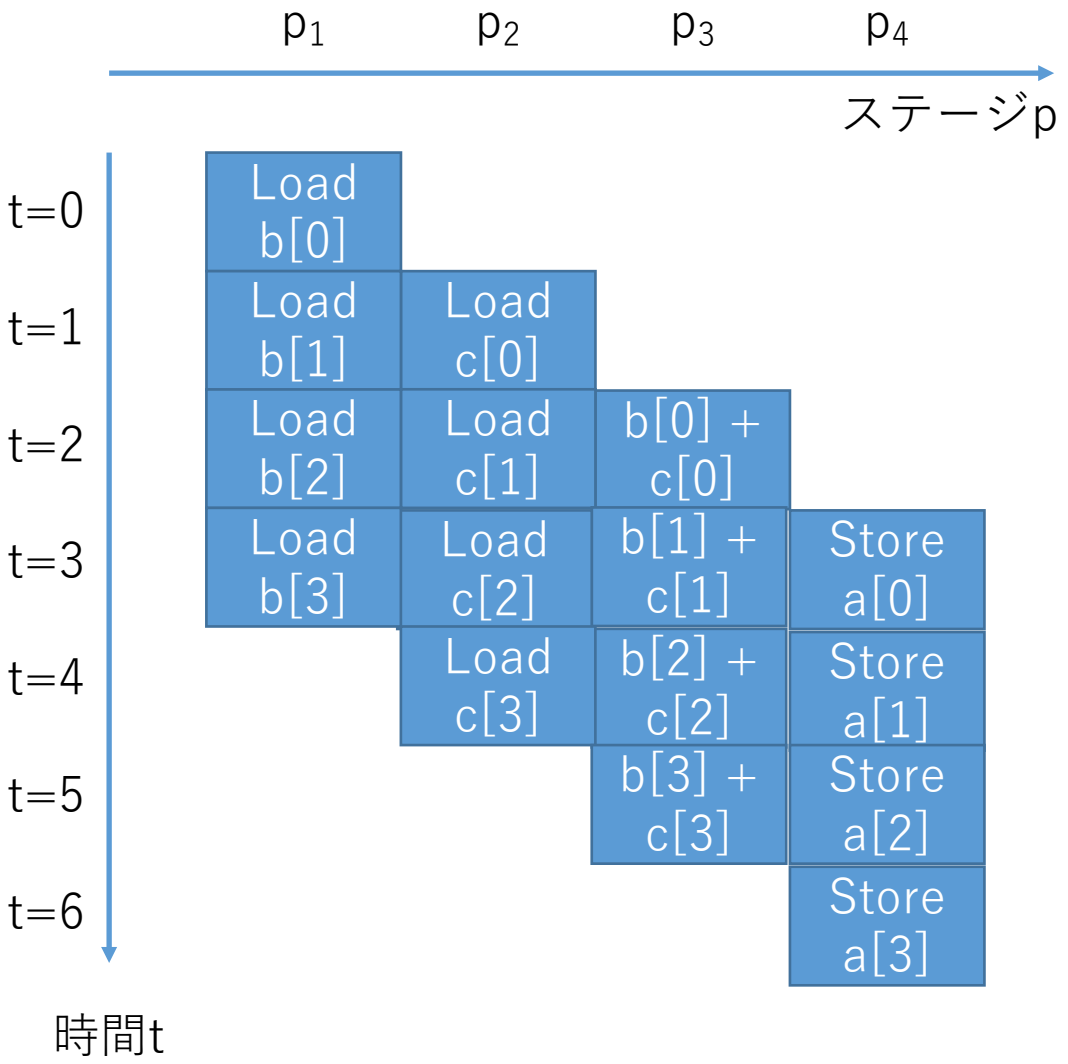
- 実行モデルがCPUやGPUと大きく異る
  - CPU: SIMD
  - GPU: CUDA, SIMT
  - FPGA: Pipeline
  - 例えば、GPU向けのOpenCLコードを、
    そのままFPGAで高速に実行できるわけではない
- 最適化の方向性も異なる
  - パイプラインを止めずに動作させ続ける事が重要
  - ループアンローリング等

パイプライン実行

- パイプライン
  - FPGAにおける基本の実行モデル
  - ループ構造はパイプラインに変換される

```
for (int i = 0; i < 4; i++) {
  a[i] = b[i] + c[i];
}
```





測定

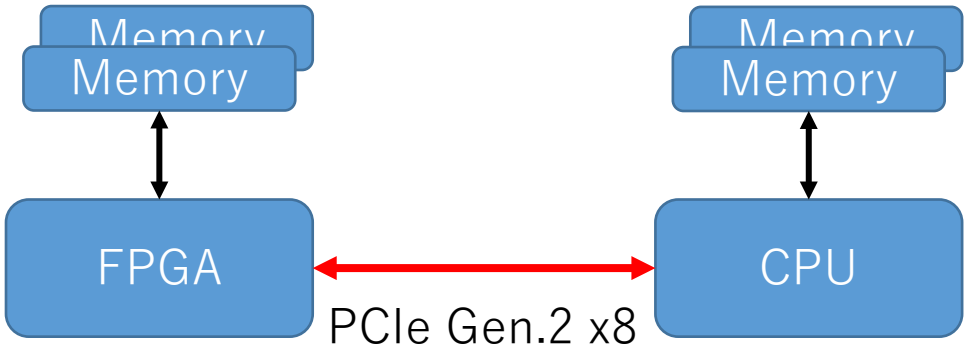
- 浮動小数点数演算性能(FLOPS)の導出には以下の式を用いる(Nは問題サイズ)

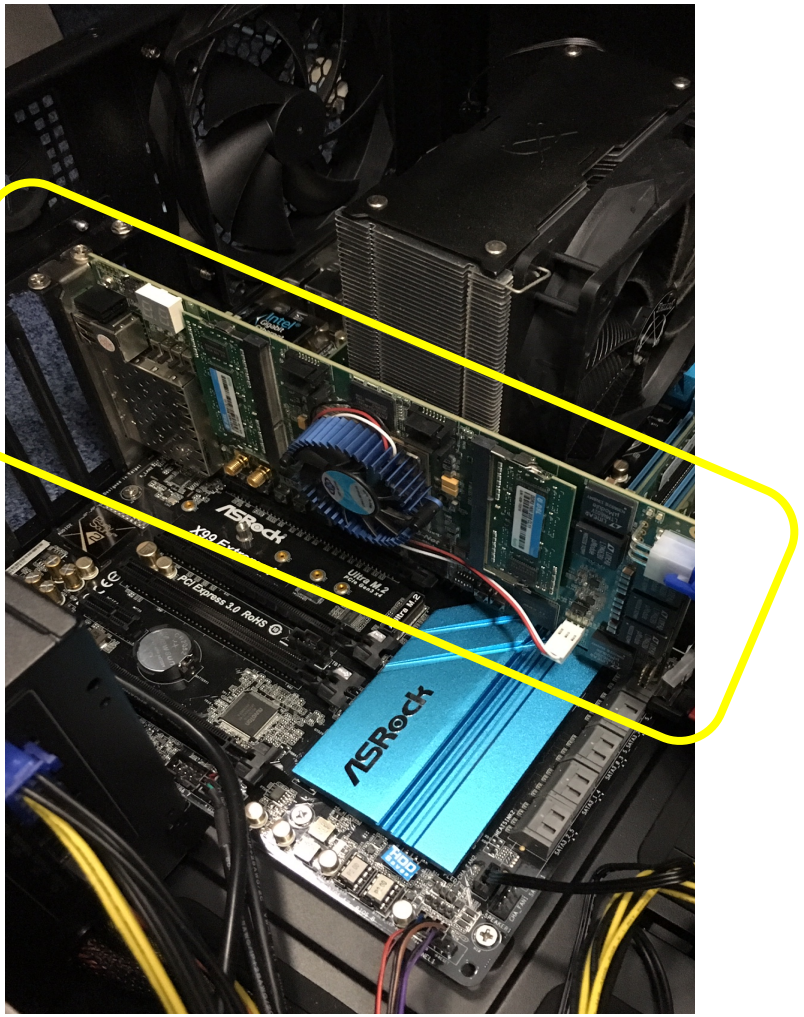
$$FLOPS = \frac{2 \times (N \times 5 + \text{行列の非零要素数}) \times \text{反復回数}}{\text{実行時間}}$$

- N = 1,200、CG法の反復回数を1000回に固定し、1行あたりの非零要素数を変化させ実行時間を測定
- 計算中に用いた実数は全て単精度(float型)である
- ループ展開数16以上から回路規模がFPGAを超えた
  - 12までの測定

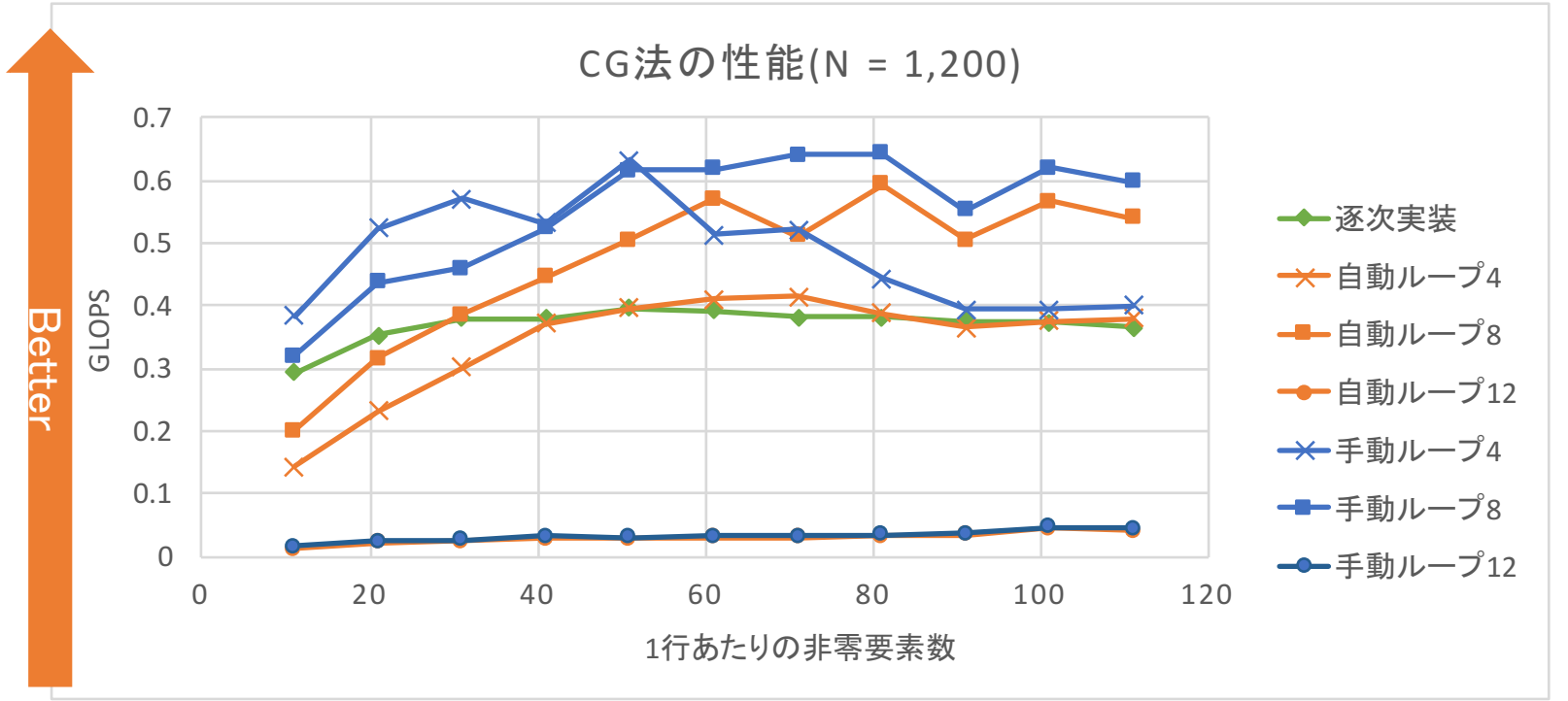
Terasic DE5-Net FPGAボード (再掲)

Terasic DE5-Net	
FPGA Chip	Altera Stratix V 5SGXEA7N2F45C2
FPGA External Memory	DDR3 SDRAM 1GB x2, 1600MHz (25.6GB/s)
PCIe (to Host)	PCIe Gen.2 x8





性能測定

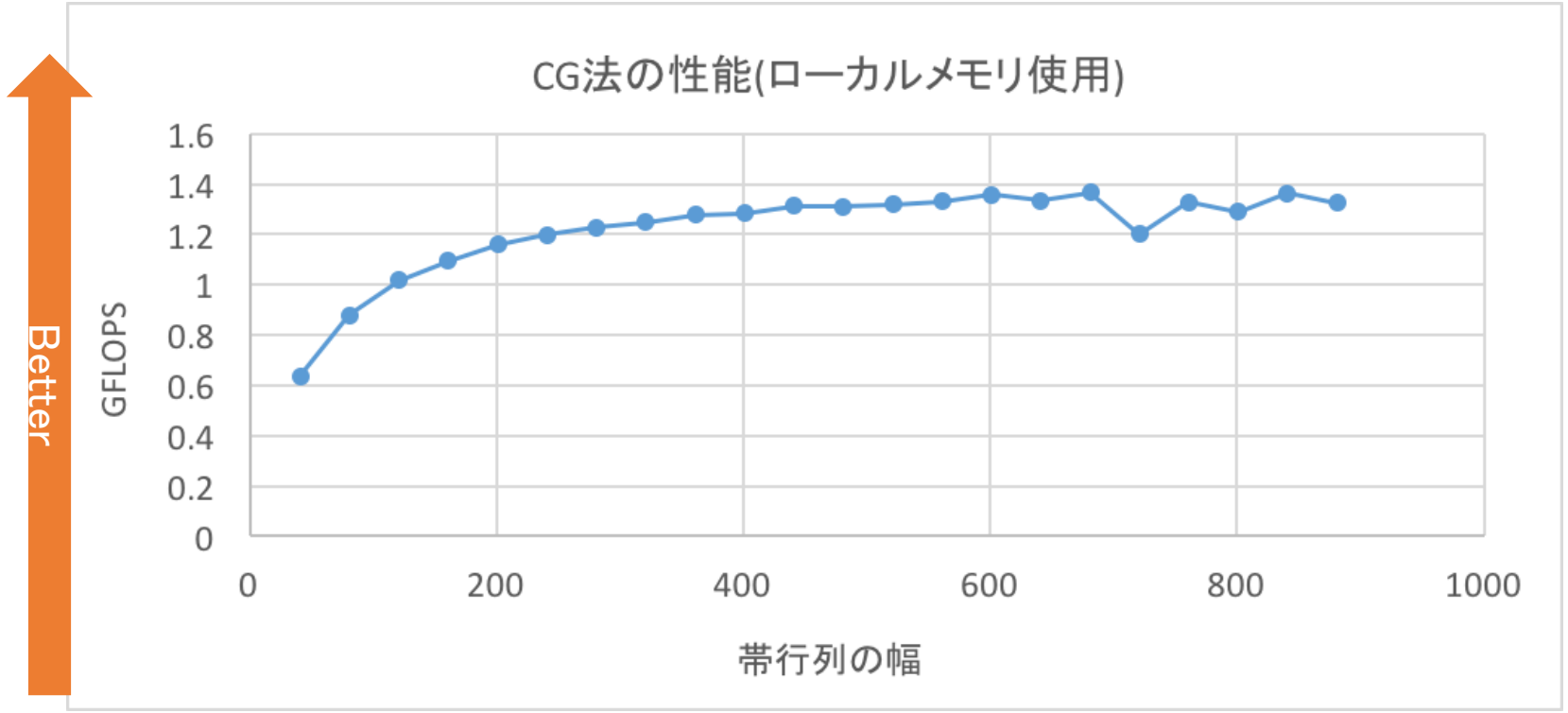


- ピーク性能は約0.65 GFLOPS
- 手動でループ展開した場合が最大となった
- ループ展開の方法によって性能が変わる
- ループ展開数12の性能が非常に悪い

ローカルメモリ利用最適化

- ベクトルPをFPGAのローカルメモリで読み書きした場合
  - OpenCLで__localと宣言。FPGA内にあるSRAMが割りてられる
- ローカルメモリの容量はDRAMと比べると非常に小さい
- 演算できる問題サイズが制限される
  - このFPGAではN = 100,000まで扱えることを確認した
  - チップによる容量上限は 6.2MB (N=1,550,000)
  - 手動でループを8展開したプログラムを使用
- N = 10,000、CG法の反復回数を10,000回に固定し、 1行あたりの非零要素数を変化させた

測定結果（ローカルメモリ）



- ピーク性能が約1.4 GFLOPS に向上した
- ベクトルPの配列をローカルメモリ上で読み書きしたことにより実効バンド幅が増えたため

まとめ

- CG法をFPGA上でOpenCLを用いて実装を行った
  - 1.4 GFLOPSの性能を達成
  - また、記述性も非常に高い
  - しかしながら、SpMVはメモリ帯域がボトルネックになる
  - DDR3 (25.6GB/s)のメモリ帯域では不足

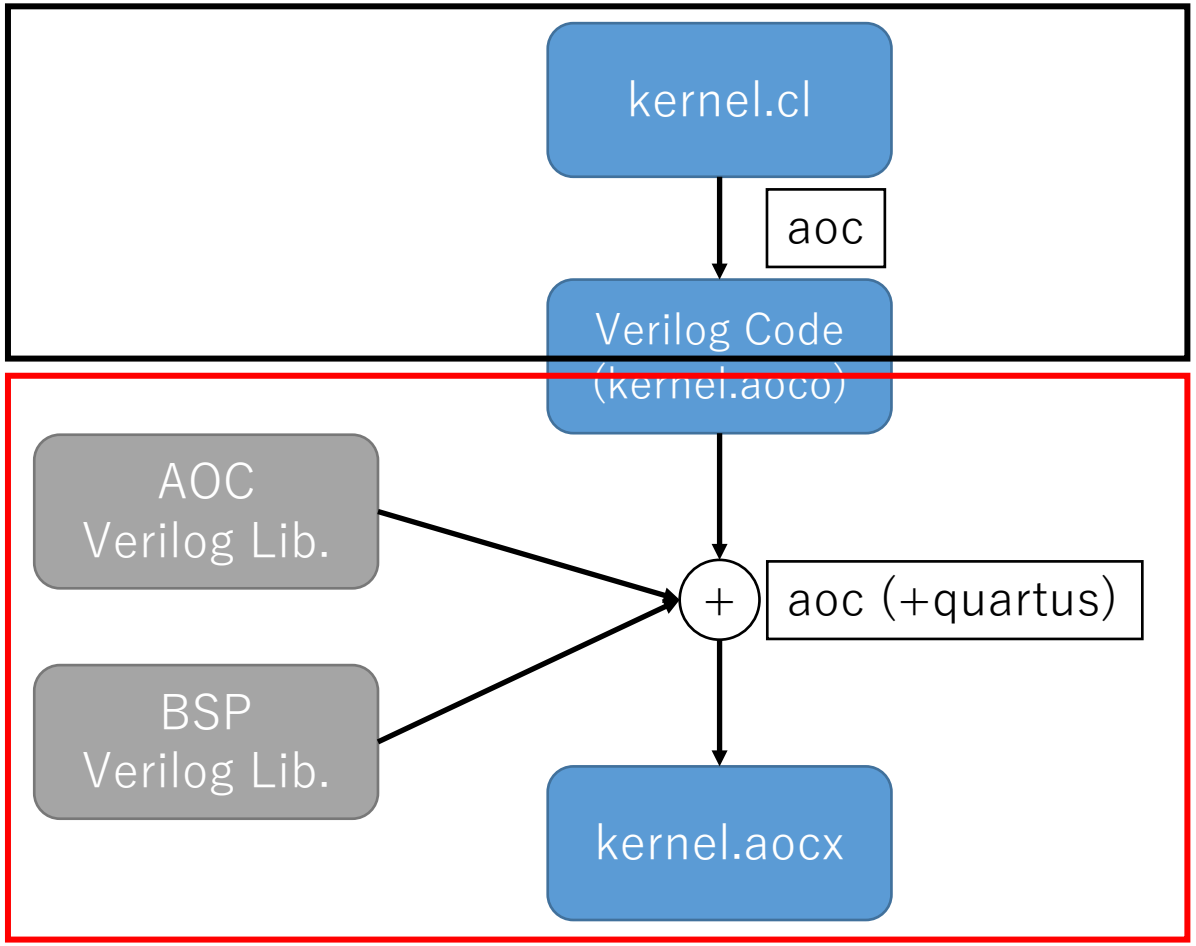
- ループのアンロール展開数を適切に設定することが重要
  - FPGA内部はパイプライン構造が生成される
  - パイプラインのストールを少なくすることが重要

まとめ

- GPUがフィットする問題で、GPUと勝負して性能で上回るのは難しい
  - ex. 行列積
  - 演算性能: 数百GFLOPS vs. 10.6 TFLOPS (NVIDIA P100, float)
  - メモリ帯域: 25.6GB/s vs. 732GB/s (NVIDIA P100)
- GPUではうまくできない事で勝負する必要がある
  - 細粒度(スレッド単位)の分岐
  - 自律動作・直接I/Oが可能
    - Accelerator in Switch
  - 問題に特化した回路生成
    - 可変精度の固定・浮動小数点数

今後の課題

- OpenCLコードをFPGA回路に変換する工程に時間がかかる
  - 単純な回路でも90分
  - 複雑な回路では～6時間程度
  - 将来のより大規模なFPGAを利用する際の不安要素
- CPU上でのOpenCLエミュレーションはあるが
  - エミュレーション速度が遅い
  - 性能評価・比較などを行う際は実機動作が必要
  - 複数のバージョンを並行コンパイルするなどの工夫が必要



今後の課題

- FPGAにおけるOpenCL最適化の検証を進める
  - 最適なメモリアクセスパターン
  - Work Group/Itemの割り当て方法
  - ループの記述方法 (特に多重なもの)

- 最新FPGAのArria 10を評価する
  - OpenCL BSPが未整備でOpenCLではまだ試せていない
  - IEEE754準拠の単精度浮動小数点数演算機がハードロジックとして搭載
  - 依然として外部メモリ帯域は弱い (DDR4)

- 以下予備スライド

Stratix VとArria 10の比較

	Stratix V	Arria 10
半導体プロセス	28nm	20nm
Logic Element (LE)	622k	1,150k
DSP Blocks	256	1,518 (IEEE-754 単精度)
onchip SRAM容量	50Mb	53Mb
浮動小数点数演算性能 (公称値)	n/a	1,366 GFLOPS
外部メモリ規格	DDR3 SDRAM	DDR4 SDRAM

*) Terasic DE5-Net、DE5a-Netボードに搭載されているFPGA同士の比較