Out-of-order スーパスカラ・プロセッサ RSD と車載向け SIMT 型アクセラレータ HORNET について

塩谷 亮太
shioya@ci.i.u-tokyo.ac.jp

塩谷 亮太

1. 東京大学大学院 情報理工学系研究科 創造情報学専攻 准教授
   - ◇ コンピュータ・システム一般の研究
2. 株式会社ティアフォー 技術アドバイザー（兼務）
   - ◇ 自動運転システムの開発

RISC-V プロセッサへの取り組み

- RISC-V の塩谷が感じる利点：
  - ◇ 最少の努力で，最新の環境が使える
- 事例：
  1. OoO スーパスカラ・プロセッサ RSD
  2. 自動運転向け SIMT アクセラレータ HORNET

もくじ

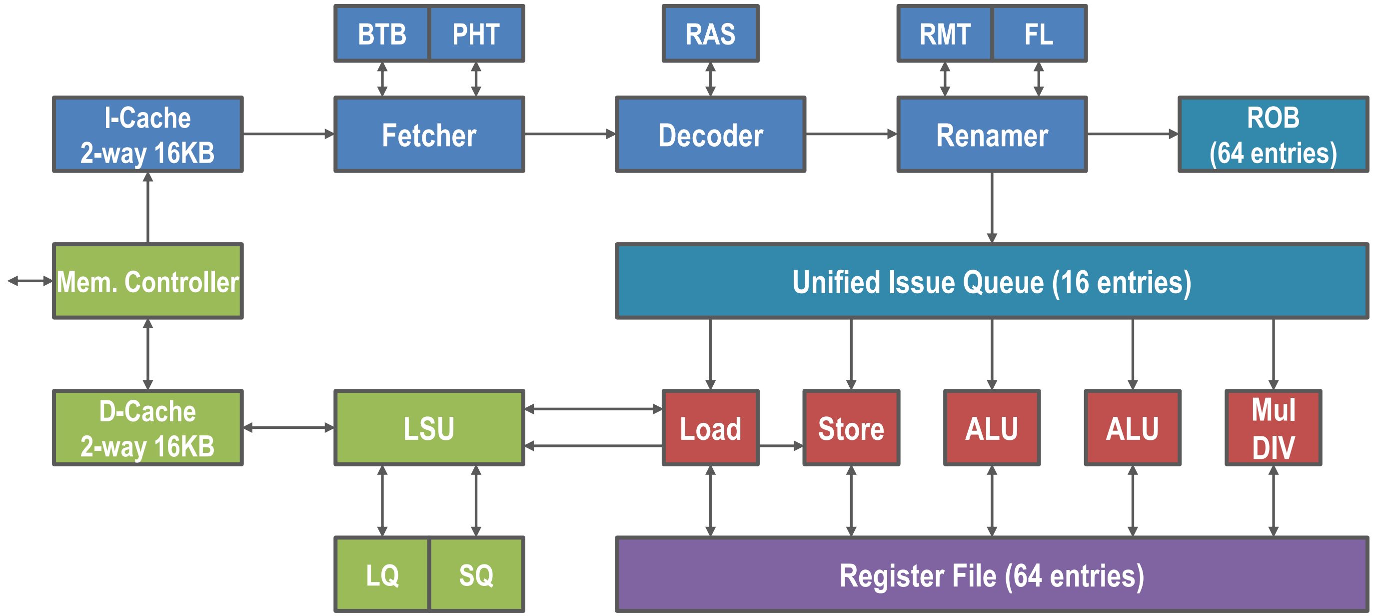
1. **プロセッサ RSD**
2. アクセラレータ HORNET

プロセッサ RSD

- コンパクトかつ高速な RISC-V OoO スーパスカラ・プロセッサ
  - RISC-V RV32IM をサポート
  - SystemVerilog で 1 万行に達しないぐらいの規模
  - FPGA 上で BOOMv2 の 1/3，MicroBlaze の 5倍ぐらいの資源量
  - Dhrystone 実行時のクロックあたり性能で，BOOMv2 の 2.5 倍ぐらい

- 論文とリポジトリ
  - Susumu Mashimo et al., : An Open Source FPGA-Optimized Out-of-Order RISC-V Soft Processor, *IEEE Intl. Conf. on Field-Programmable Technology (FPT)*, 2019
  - https://github.com/rsd-devel/rsd

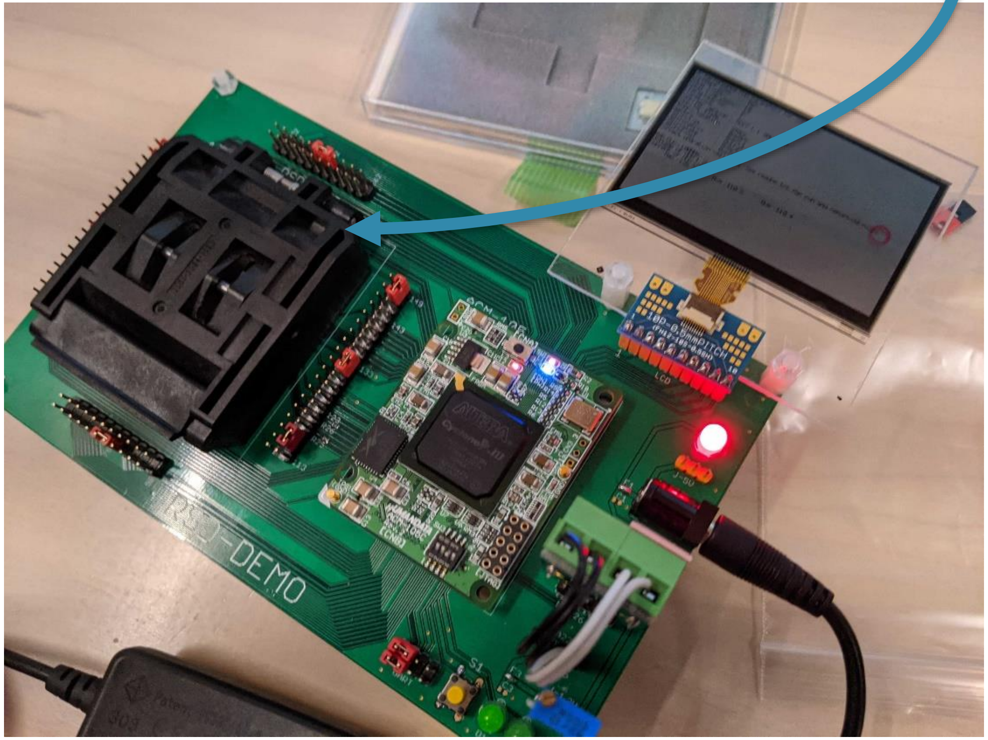
マイクロアーキテクチャ

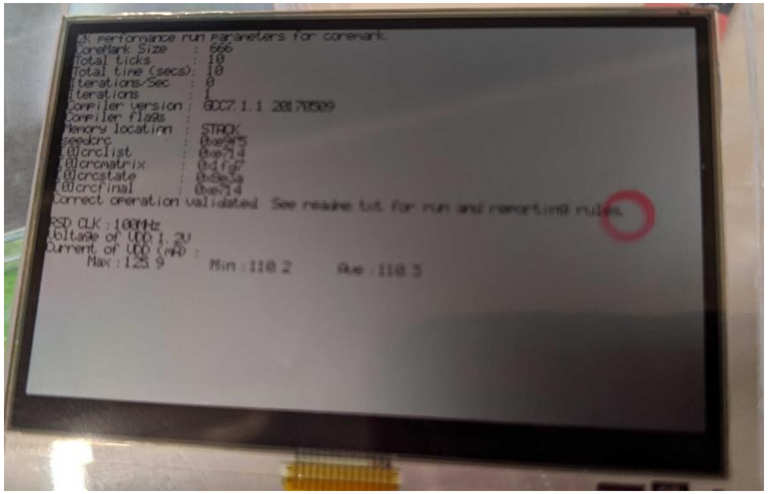


- マイクロアーキテクチャ（以下はデフォルトで，柔軟に変更可能）
  - ◇ 最大 64 命令まで out-of-order スケジューリング可能
  - ◇ 5 命令同時発行可能（2-fetch/5-issue/2-commit pipeline）
  - ◇ 2-way 16KB L1D/I caches
  - ◇ g-share 分岐予測器
  - ◇ 投機的な load/store 実行

65nm プロセスでの ASIC 実装

- ルネサス 65nm で実装（2020）
  - ◇ ASIC 設計が初めてなため，動作させることを目的に単純化＆低速化
  - ◇ 100 MHz/130mw で無事動作（左が RSD，右は IO 用 FPGA）





- 協力：VDEC，株式会社ロジックリサーチ，天野先生，奥原さん

さらに進んだプロセスでの高速な ASIC 実装

- 65nm 版はとにかく各部を単純化
  - ◇ クロック周波数は度外視
  - ◇ チップ外からのクロックやメモリ・バスがコアに直結
- 現在新しく試作中のものでは，がんばれば一桁程度は高い動作周波数が実現可能な手応え

ASIC 試作で得られた知見（1）

- クリティカルパス
  - Out-of-order 実行を行うバックエンド・パイプラインは遅延に余裕がある
  - フロントエンド・パイプラインがクリティカル
    - 分岐予測や命令キャッシュ
    - ここが問題になるとは予想していなかった
- コアとしてはかなり小型
  - L1 キャッシュの面積が支配的

ASIC 試作で得られた知見（2）

- SystemVerilog の機能を使い倒しても問題はない
  - ◇ 問題ない ＝ 問題なく高速な回路が合成できる
  - ◇ （Synopsys の DC や Synplify の場合

- RSD の記述スタイル
  - ◇ interface や struct の使用
  - ◇ assign は基本使わない
  - ◇ 論理は always_comb の中で手続き的に記述

```
    IntOpSubInfo intSubInfo[ INT_ISSUE_WIDTH ];
    BrOpSubInfo  brSubInfo[ INT_ISSUE_WIDTH ];

    always_comb begin
        stall = ctrl.backEnd.stall;
        clear = ctrl.backEnd.clear;

        for ( int i = 0; i < INT_ISSUE_WIDTH; i++ ) begin
            iqData[i] = pipeReg[i].intQueueData;
            intOpInfo[i] = iqData[i].intOpInfo;
            intSubInfo[i] = intOpInfo[i].intSubInfo;
            brSubInfo[i] = intOpInfo[i].brSubInfo;
            pc[i] = ToAddrFromPC(iqData[i].pc);
            flush[i] = SelectiveFlushDetector(
                        recovery.toRecoveryPhase,
                        recovery.flushRangeHeadPtr,
                        recovery.flushRangeTailPtr,
                        iqData[i].activeListPtr
                        );
```

RSD での RISC-V 採用

- 経緯：
  - ◇ 元々は ARM のサブセットとして開発
  - ◇ 途中で RISC-V に移行
- RISC-V 採用の利点
  1. 公開ができる
  2. 大幅な簡単化

利点１　公開ができる

- ARM 互換時は，オープンソースとしての公開はできなかった
  - 当時の塩谷は問題ないと思い込んで ARM を選択してしまった
    - QEMU 等のエミュレータのソースが公開されていたため
  - ハードの設計は勝手に公開できない
    - 特許等で訴えられる
  - 内々で使うものになっていた
- RISC-V 採用により，大手を振って公開可能に

利点２　大幅な簡単化

- ARM の命令
  - ◇ 様々な機能が直交して使用できる
- その結果,
  - ◇ ロード命令であり,
  - ◇ アドレス計算の結果にさらに加算して並列にレジスタに書き戻し,
  - ◇ さらに条件付き実行でもあり,
  - ◇ 間接分岐命令（PC を書き換える）でもある
    ･･･のような動作が１つの命令で実行されうる

利点２　大幅な簡単化

- スーパスカラのパイプラインではまともに処理できない
  - ◇ マイクロ命令への分解ではサポートしきれない
  - ◇ アセンブリコードの時点で複数の単純な ARM 命令に分解して無理矢理対応
- RISC-V 化により
  - ◇ マイクロ命令への分解自体が不要に
  - ◇ 命令デコーダが実装 4000 行から 1300 行に縮小
  - ◇ コンパイラ出力を無加工で実行可能に
- 最少の努力で，最新の環境が使える

もくじ

HORNET

RISC-V Based SIMT-Style Localization Accelerator for Autoware

TierIV

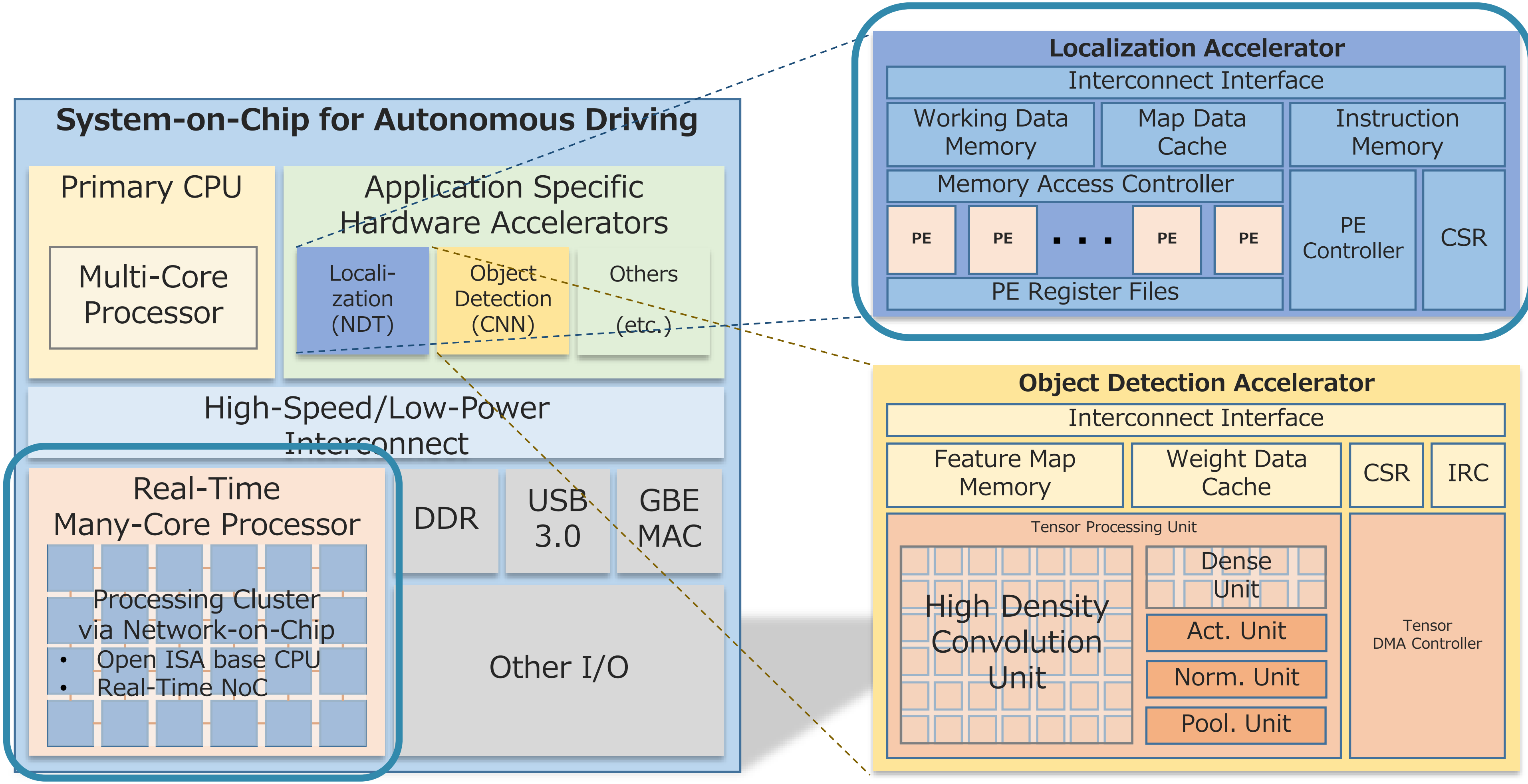
アクセラレータ HORNET

- 自動運転向けのアクセラレータとして開発
  - ◇ 株式会社アクセル，株式会社ティアフォーにより開発中
  - ◇ 塩谷はマイクロアーキテクチャに関与
- 現在の状況
  - ◇ 現状はFPGAで検証中
  - ◇ 2022年にテストチップの予定

背景：自動運転アルゴリズムと消費電力

- アルゴリズムの発展と，消費電力の肥大化
  - 現状でも高性能な CPU や GPU を使って処理する必要がある
  - 今後もさらに処理量が肥大化し続ける見込み
- 実際に売る車に搭載しようと思うと，相当に消費電力を削減していく必要がある
- アプローチ：アクセラレータで処理して消費電力を減らす
  - そのための SoC を研究・開発中

Autoware-Defined AI Accelerator Design Concept



◇ 太枠部分が RISC-V ベース

◇ 今回は右上の Localization Accelerator が主題

自動運転の処理

1. Sensing
   - ◇ センサー出力の前処理（フォーマット変換やノイズ除去など）

2. Perception
   - ◇ 現在の障害物の検知と将来の予測

3. **Localization ← 今回着目する部分**
   - ◇ LiDAR（レーザー照射による距離測定器）出力の点群と地図データのマッチングによる自己位置の計算

4. Planning
   - ◇ 上記を元に自分の動きをきめる

Localization の処理

1. NDT (Normal Distribution Transform) Scan Matching
    1. LiDAR から得られた点群から地図データ内の最近傍点を求める → ツリーのトラバース
    2. 評価値の計算と更新 → ある程度小さな行列演算

2. Position/Pose Estimation
    - ◇ Kalman Filter → そこそこ大きい行列演算

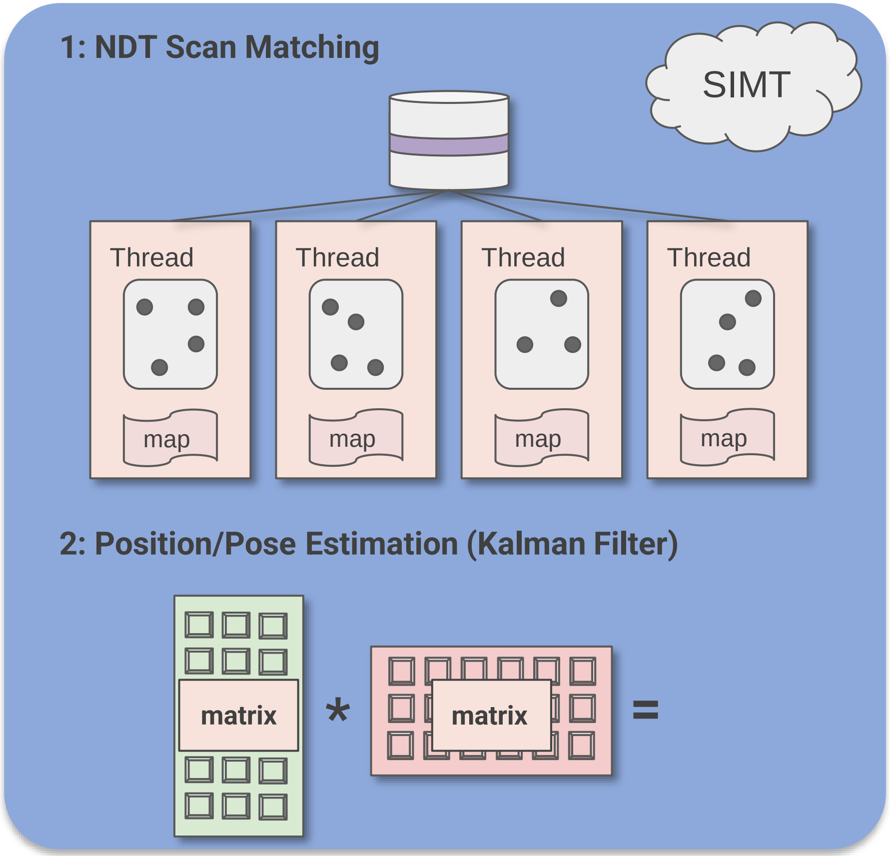
LiDAR の高精度化による出力点数の増加により処理量が肥大化
→ 自動運転の処理全体に占める割合がかなり大きい

~600,000 **~1,200,000** **~2,400,000**

https://velodynelidar.com/products/

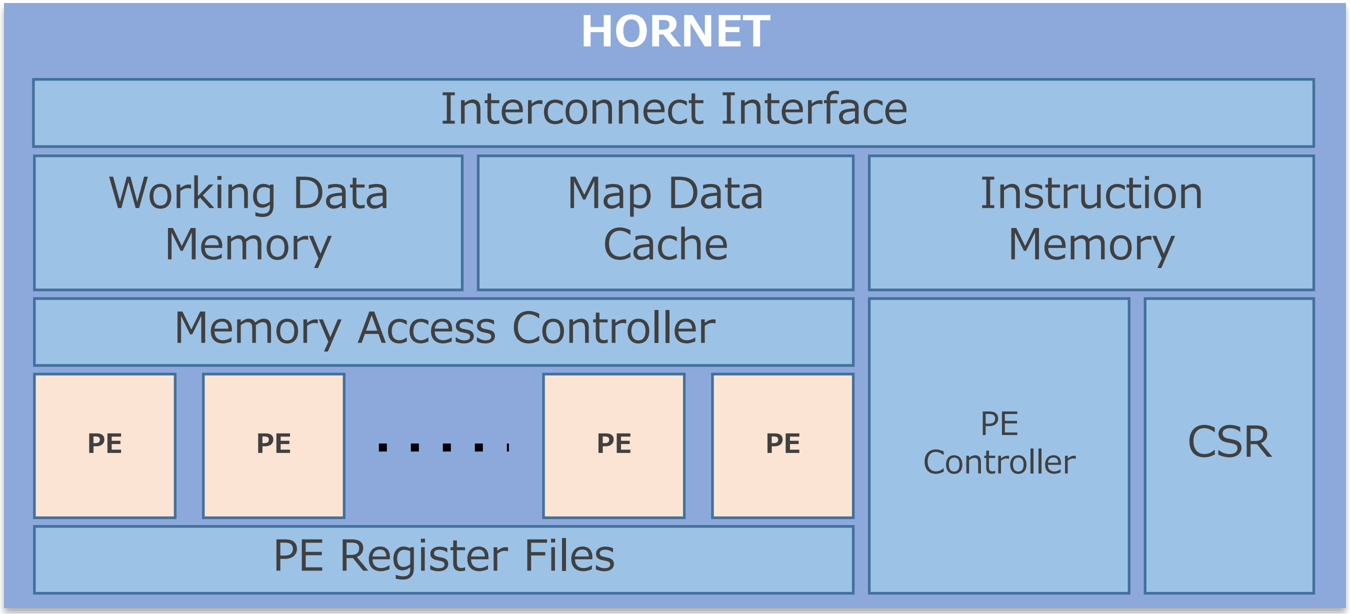
HORNET アーキテクチャ

- SIMT (Single Instruction Multiple Thread) ベースのアーキテクチャを採用
  - 単一命令で複数スレッドを駆動（複数データを処理）
  - 実行パスが分岐した場合は時分割でそれぞれ処理
- 狙い：
  - トラバースをマルチスレッドで効率良く処理
  - 行列は SIMD として効率良く処理
  - 専用回路ではなくプロセッサとすることで汎用性を持たせる



HORNET アーキテクチャ

- 比較的大きな読み出し専用キャッシュを搭載
  - ◇ 地図データなどの格納
- 汎用的な処理ができるため，その他の細かい前処理等も一括で処理可能
  - ◇ データ転送のオーバーヘッドを削減



アーキテクチャの特色

- ターゲットとする問題は，ある程度 GPU でも処理できるが･･･
  - ◇ GPU も SIMT アーキテクチャ
  - ◇ 既存の GPU では無駄が多い & 小回りが効かない
  - ◇ 分岐に弱い
- HORNET の特色：
  - ◇ ターゲットを絞って，シンプルなパイプライン構造で効率的に実現
    - □ GPU と比べて余計な機能を持たず，
      複雑なレイテンシ隠蔽機構などももたない
  - ◇ 分岐に耐性を持たせるために幅を狭く & cond. move 等の拡張

プログラミングモデルとツールチェイン

- 命令セットとして RISC-V を採用
  - ◇ SIMT 向けに拡張
- SIMT 用の API 拡張をした C 言語でプログラミング
  - ◇ Open CL の様なスキームでプログラミングが可能
- 既存の RSIC-V LLVM バックエンドを拡張して実装
  - ◇ コンディショナル・ムーブの追加など
  - ◇ 同期関係は atomic 拡張のものをそのまま使用

プログラミングモデルとツールチェイン

- LLVM バックエンドへの RISC-V からの変更量はかなり少ない
  - SIMT は通常の命令セットとはかなり実行モデルが異なる･･･
    ような印象があった（塩谷個人の印象です）
  - かなり異なるコード生成器が必要だと考えていた
  - 実際に必要な変更はロード・ストアのアドレッシング関係ぐらい
    - 各レーンに暗黙に異なるオフセットが足し込まれるなど
- 最少の努力で，最新の環境が使える

まとめ

- RISC-V にかかわる事例の紹介
  - RSD，HORNET
- RISC-V の研究上での利点：
  - オープンである
  - 最少の努力で，最新の環境が使える
- 個人的によかったこと
  - プロセッサを自分で作ってもよい空気にしてくれた
  - 大企業のみが作るものから，みんなが作るものに

謝辞

- 本研究は一部，東京大学ＶＤＥＣ活動を通して，シノプシス株式会社，メンター株式会社，日本ケイデンス株式会社の協力で行われたものである．