Vol.2014-ARC-208 No.4
2014/1/23

チーム iKOMA-TypeR のプロセッサデザインおよび評価用アプリケーションプログラムの最適化

紅林修斗† 清水怜†

本稿では，情報処理学会計算機アーキテクチャ研究会 200 回記念研究会のイベントであるプロセッサ設計コンテスト（The 1st IPSJ SIG-ARC High-Performance Processor Design Contest ）用のプロセッサデザインおよび評価用アプリケーションプログラムの最適化について述べる．

The Processor Design of team iKOMA-TypeR and The Optimization of Evaluation Application Programs

SHUTO KUREBAYASHI† RYO SHIMIZU†

In this paper, we describe a processor design and the optimization of evaluation application programs for The 1st IPSJ SIG-ARC High-Performance Processor Design Contest.

1. はじめに

本稿では情報処理学会計算機アーキテクチャ研究会 200 回記念研究会のイベントであるプロセッサ設計コンテスト（The 1st IPSJ SIG-ARC High-Performance Processor Design Contest）用のプロセッサデザインついて述べる．

2 章でまず今回実装したプロセッサについて述べる．次に 3 章ではコンテストで使用する評価用アプリケーションプログラムに対して行った最適化手法ついて述べる．最後に 4 章で評価実験結果を示す．

2. プロセッサデザイン

今回は画像処理や科学計算向けプロセッサである，線形型演算器アレイ型アクセラレータ LAPP（Linear Array Pipeline Processor ）[1]を評価用 FPGA ボードに移植した．図 1 に LAPP のモジュール構成を示す．LAPP は大きく初段の VLIW stage と 2 段目以降の Array stages に分かれており，今回は評価用 FPGA のロジック数の上限から VLIW stage のみを使用している．図 2 に最終的に実装したモジュール構成を示す．

図 1 の元の LAPP から図 2 のプロセッサコンテスト用に変更するにあたって主に次の 2 つの課程を経ている．

(1) **Array stages の削減**

前述した通りロジック数の上限から Array stage を削る必要があった．Array stages を削ると同時に VLIW stage でデコードした命令を各 Array stage に割り当てる MAP モジュールも削った．またホスト PC と CMD IF 間を図 2 の UART および PLoader に仲介させて評価用 FPGA ボードに対応させた．LAPP ではデータキャッシュおよび命令キャッシュは 4Kbyte×4way だがコンテスト用では 4Kbyte×1way とした．

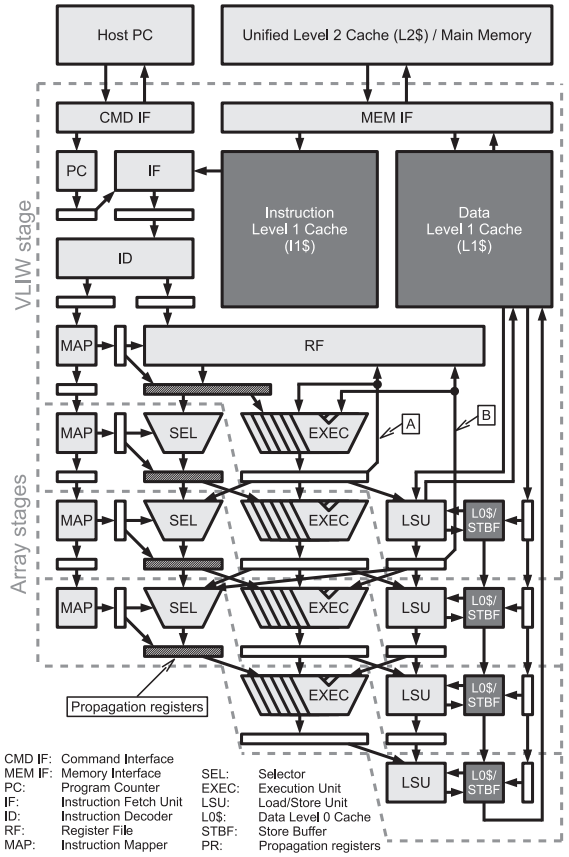

図 1 LAPP のモジュール構成[1]

† 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

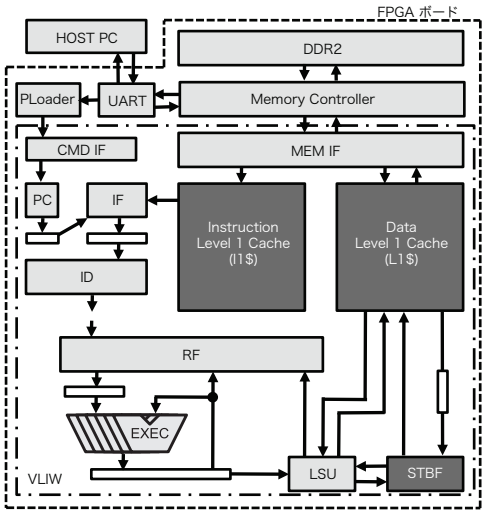

図 2　プロセッサコンテスト版 VLIW stage

(2) MEM IF 周りの変更

LAPP の開発資産として図 3(a)および(b)があり，今回はこの(b)を基にコンテスト用の VLIW プロセッサを構築した．また 4 章の評価実験ではこれを用いて評価した．

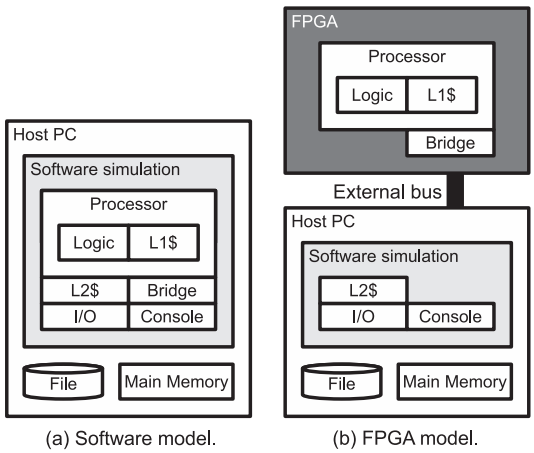

図 3　LAPP の開発資産[1]

図 3(b)に示す様に，FPGA 上での開発を容易化するため Main Memory と L2 キャッシュがホスト PC 側に存在していた．コンテスト用ではまず L2 キャッシュを排し，Main Memory を評価用 FPGA ボード上の DRAM に移動させた．またホスト PC とのインターフェイスである Bridge，I/O，Console 等の機能を，図 2 の UART や Memory Controller に差し替えた．

最終的な仕様を表 1 に示す. Contest Processor が今回コンテスト用に構築したもので，LAPP が基になった改変前のものである．コンテスト用のプロセッサは動作周波数が 30MHz，用途の異なる演算機を複数備え，最大 4 命令を同時に処理可能となっている．命令セットに FR-V を用いているが，面積の制約からメディア命令は実装できなかった

表 1　プロセッサ仕様

	Contest Processor	LAPP
Brach predictor	gshare	gshare
Return address stack	8-entry	8-entry
Decode width (instruction/cycle)	4	8
General register file	32-entry, 11R5W	32-entry, 11R5W
Media register file	none	32-entry, 7R4W
Function Unit	3 ALU 1 BRC 1 EAG 1 LSU	3 ALU 1 BRC 1 EAG 1 LSU 4 MEDIA
Data transfer speed (byte/cycle)	8	8
Instruction level 1 cache	1 way, 4KB	4 way, 16KB
Data level 1 cache	1 way, 4KB	4 way, 16KB
I1$ and L1$ cache line size	64 byte	64 byte

3. 評価用アプリケーションプログラム

今回，コンテストで用いられる 4 つの評価用アプリケーションプログラム（310_sort，320_mm，330_stencil，340_spath）の内，特に 320_mm に改善の余地が多く残っていた．この章ではこの評価用アプリケーションプログラムに対して行った最適化手法について述べる．

320_mm ではまず，データシード（init_data）から 2 つの n×n 行列である int 型配列 a[n*n]と b[n*n]を生成し，次にその内積として配列 c[n*n]を計算する．

改善前の実装を以下に示す．

```
/* ----- Initialize ----- */
    int d = 0;
    for (y=0; y<n; y++) {
        for (x=0; x<n; x++) {
            a[n*y+x] = init_data[d      % (64*1024)];
            b[n*y+x] = init_data[(d+1) % (64*1024)];
            c[n*y+x] = 0;
            d +=2;
        }
    }
/* -----   main kernel ----- */
    for (y = 0; y < n; y++) {
        for (x = 0; x < n; x++) {
            for (i = 0; i < n; i++){
                c[n*y+x] += a[n*y+i] * b[n*i+x];
            }
        }
    }
```

改善前の実装では内積計算時（main kernel）の配列 b へのアクセスが n 要素毎であり，シーケンシャルアクセスとなっておらずデータ量が大きい場合，キャッシュミスが多発してしまう問題があった．そこでデータ生成の段階で配

列 b を転置行列としておくことで，内積計算時のキャッシュミスを削減した．改善後の実装を以下に示す．

```
/* ----- Initialize ----- */
        int d = 0;
        for (y=0; y<n; y++) {
            for (x=0; x<n; x++) {
                b[n*x+y] = init_data[(d+1) % (64*1024)];
                c[n*y+x] = 0;
                d +=2;
            }
        }

/* -----   main kernel ----- */
        for (y = 0; y < n; y++) {
            for (x = 0; x < n; x++) {
                for (i = 0; i < n; i++){
                    c[n*y+x] += init_data[(2*(n*y+x))%(64*1024)]
                                * b[n*x+i];
                }
            }
        }
```

改善後では最内ループ内の配列 b へのアクセスがシーケンシャルになっている．また配列 a はデータシード（init_data）よりデータ量が大きい．よって配列 a の全要素を最初に生成するのではなく，内積計算時に必要な要素だけを逐一生成し，データ量の小さいデータシードのまま保持することでキャッシュのヒット率を向上させている．

4. 性能評価

前章で行った改善策の効果を確認するために，性能評価実験を行った．2 章の図 3 で示したシミュレーション環境を用いて IPC（Instruction per Cycle）と L1 データキャッシュのヒット率を調べた．

図 4 に IPC を示す．データサイズは n=32, 64, 128, 256, 512 で実験を行った．320_mm には 3 章で述べた 2 種類の最適化手法（転置行列，配列 a の逐次データ生成）に加えてループ展開も用いた．図 4 の各項目は Level0 が何も改善を行っていない初期の実装，Level1 が転置行列化，Level2 が Level1+逐次データ生成，Level3 が Level2+ループ展開を行った値となっている．

データサイズが大きい場合，特に n=128 以降では 3 つ全ての手法を適用した Level3 は何も適用していない初期の実装に比べての約 5 倍の IPC を実現した．

次に図 5 にデータキャッシュのヒット率を示す．Level3 ではデータサイズが大きい場合でも高いヒット率を維持できている．Level3 の IPC が Level0 のそれと比べて伸びているのはキャッシュミスによるペナルティが少ないためだと予測できる．

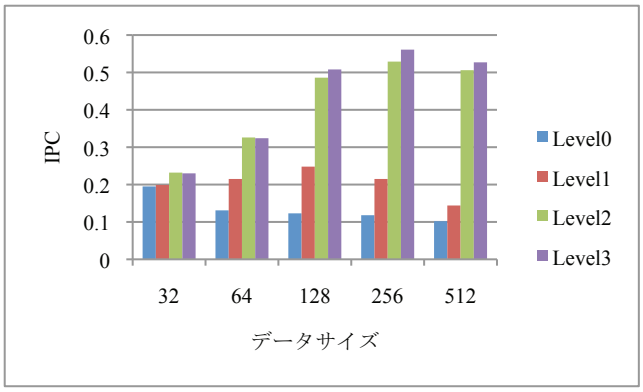

図 4　320_mm 評価実験結果 IPC

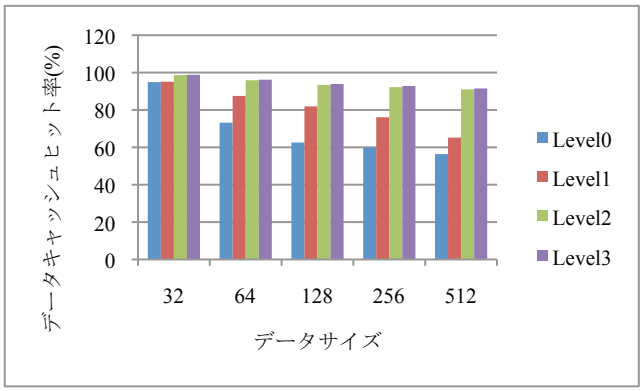

図 5　320_mm 評価実験結果データキャッシュヒット率

5. 謝辞

今回コンテストの参加に際して，多くのサポートをしていただきました実行委員の皆様，また貴重なご意見とご助力を賜りました奈良先端科学技術大学院大学情報科学研究科，中島康彦先生と姚駿先生に心からお礼申し上げます．

参考文献

1) 齊藤光俊，下岡俊介，Devisetti Venkata Rama Naveen，大上俊，吉村和浩，姚駿，中田尚，中島康彦: 線形演算器アレイ型アクセラレータを備えた高電力効率プロセッサの開発，電子情報通信学会論文誌. D，情報・システム J95-D(9)，pp.1729-1737 (2012).