Vol.2014-ARC-208 No.14
2014/1/23

ソフトプロセッサ向けの SIMD 整数演算ユニットの設計と実装

藤枝 直輝[1,a)]　宇山 和輝[1,b)]　市川 周一[1,c)]

概要：本稿では，我々が The 1st IPSJ SIG-ARC High-Performance Processor Design Contest に向けて提出した計算機システムの設計と実装について述べる．特に，課題プログラムの一部の処理を高速化するために追加した，SIMD 整数演算ユニットのアーキテクチャについて解説する．公開のデータセットを用いて評価した結果，行列積で 4.89 倍，ステンシル計算で 5.53 倍の高速化を達成した．また，その他の改善を含めたリファレンス設計からの高速化率は行列積で 31.1 倍，ステンシル計算で 30.6 倍であった．

Design and Implementation of SIMD Integer Arithmetic Unit for Soft Processors

Naoki Fujieda[1,a)]　Kazuki Uyama[1,b)]　Shuichi Ichikawa[1,c)]

1. はじめに

計算機システムの応用範囲の広まりにより，また，FPGA などの再構成可能論理技術の普及により，特定の応用プログラムに特化した計算機システム，すなわち専用計算システムの重要度が高まっている．本稿では，こうした背景を踏まえて実施された “The 1st IPSJ SIG-ARC High-Performance Processor Design Contest” [1] のプロフェッショナル部門に向け，我々のチーム “CCS Platinum” [*1] が提出した計算機システムの設計と実装について述べる．

我々が実施した改善は以下の 5 点にまとめられる．

(1) 動作周波数の検討
(2) メモリコントローラの改良およびキャッシュの追加
(3) シリアル出力回路への送信バッファの追加
(4) SIMD 整数演算ユニットの追加
(5) プログラムのソフトウェア面の高速化

このうち本デザインの最も重要な改善点が (4) である．本ユニットは，課題プログラムの一部である行列積とステンシル計算の処理の高速化を目的にして設計されたものである．本稿では上記の改善のうち，2 節でプログラムの大きな変更を要しない改善である (1) から (3) を，3 節でプログラムの大幅な修正を含む改善である (4) と (5) について述べる．4 節で評価を行い，これらの改善の有効性を確認する．

2. 動作周波数・メモリシステムの改善

本稿における全ての改善は，コンテスト実行委員会から提供されたリファレンス設計 [1] を基に行った．この設計は Xilinx Spartan-6 XC6SLX75 を搭載した，Digilent Atlys ボードを対象としている．

はじめに，リファレンス設計のプロセッサコアが動作する周波数を検討し，以降の改善でそれが著しく低下することがないように，目標の動作周波数を定める．検討の結果，リファレンス設計のコアは最大で 40 MHz をわずかに上回る周波数で動作することがわかった．しかしながら，乗算命令が大きなボトルネックとなることが確認されたため，これを 2 サイクルかけて実行するものとする．この変更により，最大の動作周波数は 44 MHz に向上した．以上の結果から，以降の改善における目標周波数を 40 MHz と

[1] 豊橋技術科学大学
Toyohashi University of Technology
[a)] fujieda@tut.ac.jp
[b)] uyama@ccs.ee.tut.ac.jp
[c)] ichikawa@tut.jp
[*1] CCS は Custom Computing Systems の略である．

定める．

次に，メモリコントローラのポート幅を拡大するとともに，FPGA 内部のブロック RAM 領域を用いたキャッシュを構成する．Atlys ボードには DDR2 SDRAM チップが搭載されており，FPGA 内部のメモリコントローラを通してアクセスできる．リファレンス設計では 32 ビット毎の転送を行うが，コントローラの生成に用いる Xilinx Memory Interface Generator の設定を変更し，またバースト転送を利用することにより，512 ビットを一度に転送するように変更する．構成したキャッシュでは，この 512 ビットを単位にアクセスを行う．キャッシュの容量は，ブロック RAM 領域の全体容量を考慮して 128 KiB と定める．キャッシュの構造については，複雑なものをとればその分性能は向上すると考えられるが，実装やデバッグに要する期間が長期化するリスクがある．そのため，比較的シンプルな構造で良好な性能を示す，2-way Skewed-Associative Cache [2] を用いることとする．これは，ウェイごとに異なる計算式でインデックスを計算することで，インデックスの競合が頻発すること（スラッシング）によるキャッシュ性能低下を防ぐものである．

更に，シリアル通信モジュールの設計を改善し，送信バッファを追加して複数文字を送信する際の待ち時間を削減する．送信バッファは，ブロック RAM を用いた 2 KiB の FIFO として実現する．課題プログラムで連続送信される文字列は高々 1.2 KiB 程度であるから，これにより送信時のソフトウェアによるウェイトが一切不要となる．

3. SIMD ユニットのアーキテクチャ

前節に挙げた 3 つの改善は，ソフトウェアの面では変更が不要であるか，あるいはライブラリレベルでの変更で済むものである．本節では更なる高速化を達成するために，プログラムの修正を伴う改善を行う．特に，課題プログラムのうち行列積とステンシル計算の 2 つを大幅に高速化するため，SIMD 整数演算ユニットを追加する．

行列積では，積和演算により行および列の各要素の積の総和を求める必要がある．また，課題プログラムのステンシル計算では，自身とそれに隣接する 8 つの要素との平均を求める必要がある．これらを実現するためには，SIMD ユニットは少なくとも SIMD アキュムレータと 2 つの SIMD レジスタをもち，以下の機能をサポートする必要がある．ただし行列積では，行列を生成する前処理の段階で一方の行列を転置しておくものとする．

(a) レジスタにメモリから値を読み出し，レジスタの値をメモリに書き込む．
(b) アキュムレータの各要素を 0 でリセットする．
(c) 2 つのレジスタの各要素の積をアキュムレータに加算する．
(d) アキュムレータの各要素の総和を求める．

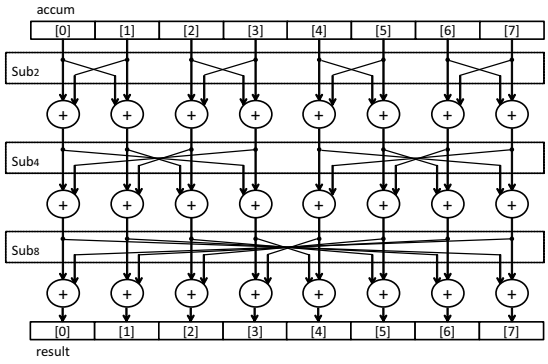

図 1 アキュムレータの各要素の総和を求める演算．

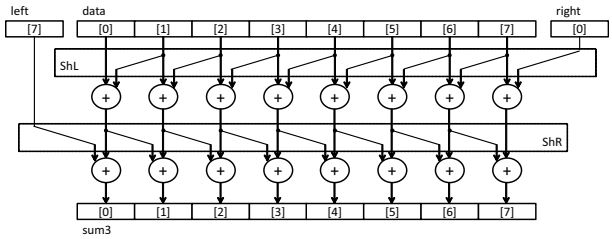

図 2 左右に隣接した要素との和を求める演算．

(e) レジスタの各要素と，その左右に隣接した要素との和を求める．
(f) レジスタの各要素の値をアキュムレータに加算する．
(g) アキュムレータの各要素を定数で除算する．

このうち (a) と (b) は両方のプログラムに必要な基本機能であり，(c) と (d) は行列積で，(e) から (g) はステンシル計算で用いる．SIMD レジスタの大きさは，FPGA の DSP（Digital Signal Processing）ユニットの搭載数と，1 つの 32 ビット乗算器につき 4 つの DSP ユニットを使用することを考慮して，8 ワード（256 ビット）に定める．

行列積においては (c) を繰り返し用いることで 8 つの部分和が得られる．これらの部分和から総和を求める (d) の演算を**図 1** に示す．この演算には，3 種類の SIMD レジスタ内要素の入替え（Sub_2，Sub_4，Sub_8）と，3 回の SIMD 加算を含む．

ステンシル計算においては，初めに行方向の隣接要素との和を計算しておき，次にそれらと列方向で隣接したもの同士との和をとることで，隣接 8 要素との和を求める．その上で，これらを定数 9 で除算して，平均を求める．行方向の隣接要素との和を求める演算を**図 2** に示す．この演算は，SIMD レジスタ要素間の左シフト（ShL）・右シフト（ShR）と，2 回の SIMD 加算を含む．また，一般に定数による除算は，定数による乗算とシフト・加算に変換できる [3]．具体的には，9 による除算の商は，0x38e38e39 で乗算した積を 33 ビット右シフトし，結果が負数であれば 1 を加算することによって得られる．このとき，最初の定数乗算を列方向の和を取る直前に行うことにすれば，対応する (f) の機能は，レジスタの各要素の値と定数との積を

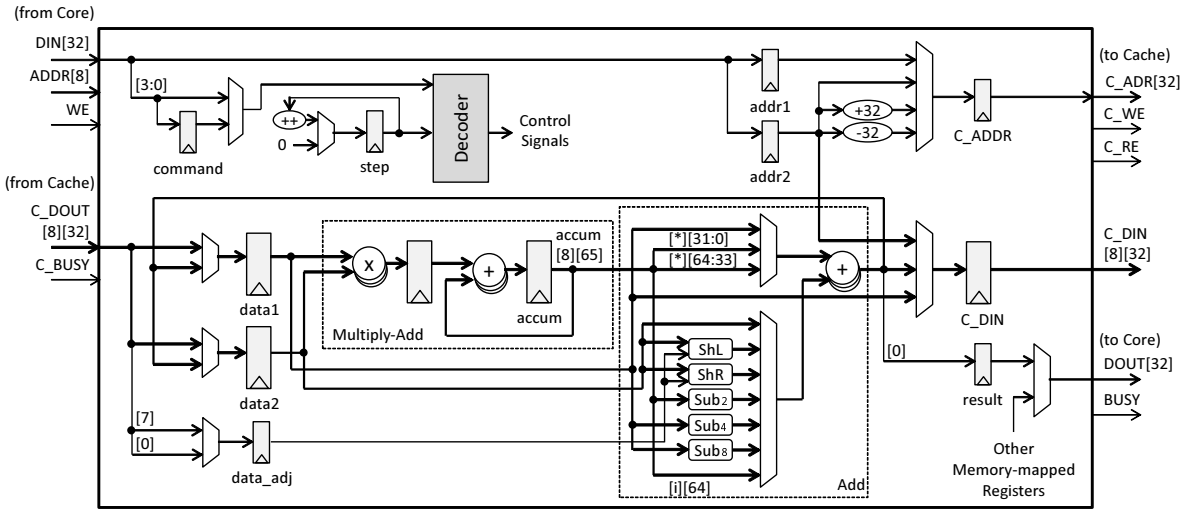

図 3 SIMD ユニットの全体構成.

アキュムレータに加算する機能と読み替えられる．すなわち，予めレジスタの各要素に定数を書き込んでおくことで，(f) の機能は (c) の機能へと統合できる．そして，(g) の機能は，アキュムレータの各要素について 33 ビット右シフトしたものと符号ビットとの和を求める機能と読み替えられる．ゆえに，SIMD ユニットが必要とする SIMD 算術演算は，アキュムレータに対する積和演算と，入出力を切り替え可能な加算とにまとめられる．

以上をもとに，SIMD 積和演算と加算，レジスタ要素間の入替えやシフトを用いて，必要な機能群をサポートするユニットとして，SIMD ユニットを設計する．**図 3** に，SIMD ユニットの全体構成を示す．プロセッサコアはメモリマップされた 3 つの入力レジスタ（command，addr1，addr2）と 1 つの出力レジスタ（result）を介して SIMD ユニットにアクセスする．コアが command レジスタへの書き込みを行うと，SIMD ユニットは動作を開始する．各コマンドはいくつかのステップに分割されており，デコーダ（Decoder）はコマンド番号とステップ番号をもとに各種制御信号を生成する．

例えば，図 1 に示した総和を求める演算は，3 つのステップから構成される．最初のステップでは，アキュムレータの各要素の下位 32 ビットとそれらに対し第 1 の要素入替え（Sub_2）を適用したものとをそれぞれ加算し，data1 レジスタへと格納する．第 2 のステップでは，data1 レジスタの各要素と，data1 レジスタに第 2 の要素入替え（Sub_4）を適用したものとを加算し，再び data1 レジスタへと格納する．最後のステップでは，data1 とその第 3 の要素入替え（Sub_8）との加算の結果の 1 要素を取り出して，result レジスタへと格納する．このように，先に示した SIMD ユニットが必要とする機能群は，こうした小さなステップの組み合わせによって表現される．

SIMD ユニットの追加にともない，行列積とステンシル計算のプログラムを SIMD ユニットを利用したプログラムへと変更する．また，これらを含めた全てのプログラムで，ソフトウェア面の高速化を行う．これらの高速化には，コンパイルオプションの検討，手動による不要なメモリ書き込みの削除やループアンローリング，配列の添字のポインタによる置き換えなどを含む．

4. 評価

本節では我々が実施した 5 種類の改善について，Atlys ボード上で，コンテスト実行委員会が公開しているツールキットに付属のデータセット [1] を用いて評価する．論理合成と FPGA への実装には Xilinx ISE 14.5 を使用し，オプションは Planahead のデフォルト設定を用いる．各プログラムの実行ファイルは gcc 4.7.3，および binutils 2.21 を用いて生成する．また，ソフトウェア面の高速化における開発補助と性能見積もりのために，SIMD ユニットに相当する機能を独自に追加した SimMips 0.7.5 [4] を利用する．

リファレンス設計における動作周波数を 30 MHz，改善後の各設計における動作周波数を 40 MHz とする．改善の指標として，シリアル通信により全ての入力データを受け取ってから，計算を完了して全てのデータが出力されるするまでの時間（以下計算時間という）を用いる．すなわち，コンテストにおいて定義される実行時間から，入力データのシリアル通信に要する 5.24288 秒（$= 2^{19}/100000$）を減じたものを指標とする．

表 1 に，それぞれの改善における計算時間と，改善前後の，またはリファレンス設計（表中では Ref で表す）との性能向上比を示す．一番左の列の (1) から (5) は，その番号までの全ての改善を施したシステムを表す．時間の列は秒単位の計算時間を示す．対改善前の列はその番号の改善を施す前と比較した性能向上比を，対 Ref の列はリファレンス設計と比較した性能向上比を表す．これらを 4 つのプ

表 1 各改善後における計算時間と性能向上比.

	310: 整数ソート			320: 行列積			330: ステンシル計算			340: 最短経路問題		
改善	時間 [s]	対改善前	対 Ref	時間 [s]	対改善前	対 Ref	時間 [s]	対改善前	対 Ref	時間 [s]	対改善前	対 Ref
Ref	13.319		1.00	16.309		1.00	14.352		1.00	26.898		1.00
(1)	9.792	1.36	1.36	12.166	1.34	1.34	10.468	1.37	1.37	19.695	1.37	1.37
(2)	3.559	2.75	3.74	3.082	3.95	5.29	2.923	3.58	4.91	4.909	4.01	5.48
(3)	3.234	1.10	4.12	2.956	1.04	5.52	2.679	1.09	5.36	4.887	1.01	5.50
(4)	3.234	1.00	4.12	0.604	4.89	27.00	0.485	5.53	29.61	4.887	1.00	5.50
(5)	2.987	1.08	4.46	0.524	1.15	31.11	0.469	1.03	30.60	2.809	1.74	9.58

ログラムそれぞれについてまとめている.

(1) の動作周波数の変更の結果，周波数比 4/3 よりも少し大きな改善がみられた．これは DRAM のリフレッシュ頻度が相対的に減少し，リフレッシュを待つレイテンシが改善されたためであると考えられる.

(2) のキャッシュの追加による性能向上は最大で 4 倍となった．付属のデータセットは比較的小さいと予想されるので，キャッシュの恩恵をより受けやすいと考えられる．そのため，決勝問題のデータセットを用いた場合，キャッシュによる改善はより小さくなる可能性がある.

(3) の出力バッファによる絶対的な計算時間の短縮は，出力すべき文字数に比例することが予想されるが，評価からもそのことが確認された．性能向上比では，ほとんど出力のない最短経路問題では 1%未満の改善にとどまったが，比較的多くの文字を出力するクイックソートでは 10%程度の改善を得た.

(4) の SIMD ユニットの追加による恩恵を受けられるのは行列積とステンシル計算のみであるものの，これら 2 つのプログラムでは非常に大きな性能改善を得た．SIMD ユニット追加前後の性能比は，行列積で 4.89 倍，ステンシル計算で 5.53 倍となった．特にステンシル計算で大きな効果を得られたのは，改善前の実行ファイルが定数除算に低速な除算命令を使うようコンパイルされていたことが原因の 1 つと考えられる.

(5) のソフトウェア面の高速化では，特に最短経路問題において効果が大きく，1.74 倍の性能向上を得た．この向上に最も関係する改善点は，配列の添字を格納する変数がループ内で頻繁に利用されていたことに注目し，その変数に対し直接ポインタを格納するように変更したことである．これによりアドレス計算に関係する命令が削減できた.

以上全ての改善点を組み合わせた結果，リファレンス設計に対して行列積で 31.1 倍，ステンシル計算で 30.6 倍の性能向上が達成された．また，SIMD ユニットが適用できない整数ソートや最短経路問題についても，それ以外の改善を通してそれぞれ 4.46 倍，9.58 倍の高速化に成功した.

5. おわりに

本稿では，我々がプロセッサ設計コンテスト向けに設計した高性能計算機システムについて，特に，その中で最も重要な改善点である SIMD 整数演算ユニットについて述べた．評価の結果，全ての改善を適用した場合，SIMD ユニットが適用可能であった 2 つのプログラムについては，いずれもリファレンスに対して 30 倍を超える高い性能向上を達成した.

決勝デザインの提出に向けては，例えば SIMD ユニットでは処理の分割方法や同期のタイミングなどといった，細かい最適化の余地が残されている．また，決勝問題ではより大きなデータセットが想定されるので，それを踏まえたメモリシステムの調整も必要と考えられる．今後はより高い性能向上を目指し，可能な限り修正・調整を進めていく.

参考文献

[1] プロセッサ設計コンテスト実行委員会：The 1st IPSJ SIG-ARC High-Performance Processor Design Contest，情報処理学会 (online)，available from ⟨http://www.arch.cs.titech.ac.jp/contest/⟩ (accessed 2013-11-28).

[2] Seznec, A.: A Case for Two-way Skewed-associative Caches, *Proceedings of the 20th Annual International Symposium on Computer Architecture*, pp. 169–178 (1993).

[3] Granlund, T. and Montgomery, P. L.: Division by Invariant Integers Using Multiplication, *Proceedings of the ACM SIGPLAN 1994 Conference on Programming Language Design and Implementation*, pp. 61–72 (1994).

[4] 藤枝直輝，渡邉伸平，吉瀬謙二：教育・研究に有用な MIPS システムシミュレータ SimMips，情報処理学会論文誌，Vol. 50, No. 11, pp. 2665–2676 (2009).