C-015

Bidirectional な SIMD をもつキュープロセッサ

Bidirectional SIMD Queue Processor

大日向 大地 †,††　曽和 将容 †
Daichi OBINATA　Masahiro SOWA

1 はじめに

計算に FIFO の記憶を用いるキュー計算モデル [1, 2] では，プログラムは高い並列性を持つことが知られており，キューレジスタを用いたスーパスカラプロセッサが実現されている [3, 4].

本論文ではキュー計算モデルの並列性に対して SIMD を適用する過程で，この並列性が持つ動的なふるまいに着目した Bidirectional SIMD と呼ぶ SIMD を提案し，その命令を実行可能なプロセッサを論じる.

2 キュー計算モデルへの SIMD の適用

キュー計算モデルにおける SIMD 演算には，キューへのアクセス方法から 2 種類の演算が定義できる．ひとつは，キューをベクトルレジスタとするキューベクトル演算，もうひとつは連続するスカラ演算をひとつの命令で表現するキューマルチスカラ演算である.

2.1 キューベクトル演算

キューはデータが境界なく一列に並ぶため，どのようなデータ列も単純な一次元のデータ配列として平坦に記憶され，命令によって意味付けされることになる．この性質を利用し，一次元配列の一部をベクトルデータとして扱うベクトル演算 (Fig. 1) が定義でき，この方式を**キューベクトル演算**と呼ぶ.

ベクトルレジスタに FIFO を用いる方式はすでにある [5] が，キューベクトル演算では完全に一本の FIFO を使用しており，以下のような特徴がある.

1. **ベクトル長を制限しない** — キューサイズの物理的限界までベクトル長を伸ばすことができる.
2. **記憶資源の効率化** — 複数のベクトルデータ，スカラデータを一本のキューに配置するので，記憶資源が無駄なく使用できる.

2.2 キューマルチスカラ演算

キュー計算モデルの特徴のひとつは，演算命令がオペランドフリーであることである．これは，キューがひとつで，そのアクセス方法が命令によって一意に定まるためで，LIFO のスタック計算モデルでも同様である.

命令にオペランド指定が伴わないことで，プログラム中に同じ命令が連続するケースが発生する．このとき，同じ命令を何重にも連続して列挙するより，ひとつの命令とそれが繰り返される回数を記すほうが，プログラムの情報量としては有利である.

それに従い，あるスカラ演算を与えられた回数繰り返す演算方式をマルチスカラ演算，特にキュー計算モデルでの方式 (Fig. 2) を**キューマルチスカラ演算**と呼ぶ.

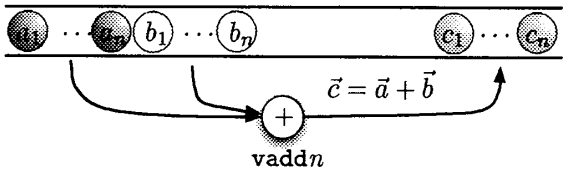

Fig. 1 Vectorized Queue Computing

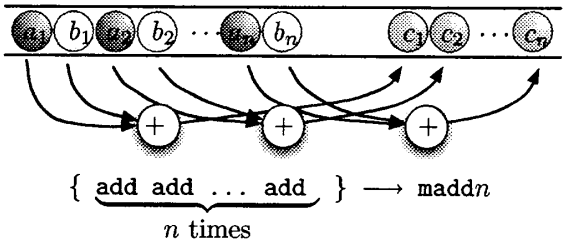

Fig. 2 Multi-scalarized Queue Computing

3 キュー計算モデル SIMD の 2 方向性

キューベクトル演算はデータが完全に並列で横の関係のみを持ち，並列度は命令によって固定化される．対してキューマルチスカラ演算では同じ命令の繰り返しであるため，横の並列性のほか，依存関係を持つレベルを超えた縦の従属性を命令内部に持たせることが可能であり，このときの並列度の幅と従属性の深さはキューに並ぶデータの数によって命令実行時に動的に決定される．例えば，mop6 という命令 (ここで，prefix の m はマルチスカラ命令，op は 1 入力 1 出力の任意の演算，suffix の 6 は繰り返し回数を示す) の場合，キューに並ぶデータが 3 つであれば 3 並列の op 演算が 2 度行われ，キューに並ぶデータが 2 つのときは 2 並列の op 演算が 3 度行われることになる (Fig. 3). また，これは Fig. 4 に示す演算のように，異なるレベル間で並列度が同じである必要はなく，利用の幅は広い.

この，ひとつの命令が横方向の並列性と縦方向の従属性の 2 つを備え，キューに並ぶデータの数に従って縦横両方向に動的に伸縮することのできるこの演算方式を Bidirectional SIMD (**2 方向性** SIMD) と呼ぶ.

Bidirectional な性質は，オペランドフリーでかつ記憶に FIFO を用いるキュー計算モデルにおいて現れ，LIFO を用いるスタック計算モデルでは同じオペランドフリーでもこの性質は現れない.

4 Bidirectional SIMD 計算機の構成

4.1 マシンモデル

Bidirectional SIMD を備えるキューマシンの基本構成モデルを Fig. 5 に示す．このマシンモデルは，主記憶のキュー Queue，演算器 Ex，演算器への入力データ

† 電気通信大学大学院情報システム学研究科
†† obinata@sowa.is.uec.ac.jp

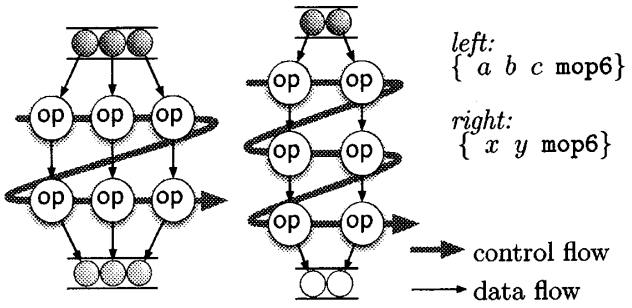

Fig. 3 mop6 Bidirectionalism — 6 scalar op operations

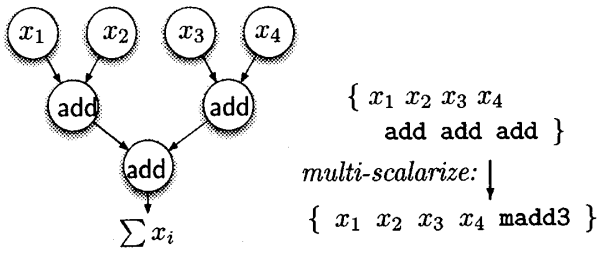

Fig. 4 Summation with Bidirectional SIMD

のキュー $\mathrm{SrcQ}_{\{1,2\}}$，演算器からの出力データのキュー $\mathrm{DestQ}_{\{1,2\}}$，演算の命令のキュー OpQ から成り，次のうな流れで動作する．

1. 入力された命令は，ベクトル命令であればベクトル長分，マルチスカラ命令であれば回数分の数のスカラ命令列に展開され，OpQ にキューイングされる (この数を N とする)．
2. 演算のソースデータは，Queue から $\mathrm{SrcQ}_{\{1,2\}}$ にバッファリングされた後に演算器に渡される．ベクトル命令であれば，Queue の先頭から N 個を取り出して SrcQ_1 にキューイングし，さらに Queue から N 個を取り出して SrcQ_2 にキューイングする．マルチスカラ命令のときは，Queue の先頭からデータを取り出し，SrcQ_1，SrcQ_2 に交互に書き込む手続きを N 回行う．
3. Ex は OpQ にキューイングされている命令を読み，命令の動作規則に従って SrcQ_1，SrcQ_2 から値を取り出し，結果を DestQ_1，DestQ_2 に書き込む．
4. 演算の結果は $\mathrm{DestQ}_{\{1,2\}}$ から Queue に書き込まれる．ベクトル命令であれば，DestQ_1 から N 個取り出して Queue へ書き込み，さらに DestQ_2 から N 個取り出して Queue へ書き込む．スカラ命令であれば，DestQ_1，DestQ_2 から交互に取り出して Queue に書き込む手続きを N 回行う．

この一連の繰り返しにより，ひとつの計算機上でキューベクトル演算，キューマルチスカラ演算の両方の演算方式が実行できる．

4.2 SIMD キュープロセッサ

実物のプロセッサとして実現するには，前述のスーパースカラプロセッサをベースとし，ベクトル命令・マルチスカラ命令を展開するパイプラインユニットを増設し，演算で使用するキューの割当てを行うキュー計算ユニットに機能拡張を行ことで実現できる．これは，マシンモデルの $\mathrm{SrcQ}_{\{1,2\}}$，$\mathrm{DestQ}_{\{1,2\}}$ を Queue の一部に

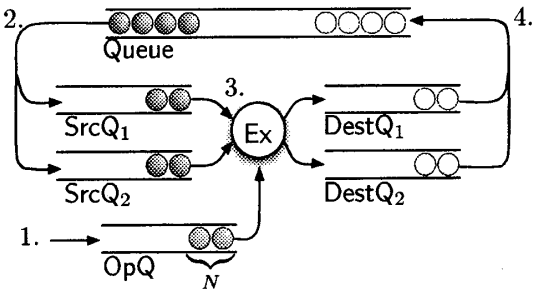

Fig. 5 Bidirectional SIMD Queue Machine Model

マッピングすることでバッファリングのオーバヘッドを省くことを狙うものである．

こうして実現されるプロセッサは，

- ベクトル長の制限が大幅に緩和される
- 連続する同一の命令がひとつの命令にまとめられる

などの理由により，高度な多元行列演算などにおいて有用であると考えられる．また，このプロセッサではひとつの演算命令をプロセッサ内部で展開および繰り返し実行するため，並列度あるいは繰り返し回数が演算器の数に比べ十分に大きくなると，命令フェッチが長時間遊休状態となる．このとき，フェッチユニットや外部バスを有効利用することで，高性能化が期待できる．

5 おわりに

キュー計算モデルにおける 2 種類の SIMD 演算の方式と，それを備える計算機の可能性を論じた．キューベクトル演算では，ベクトル長制限の緩和と記憶資源の有効利用が期待できる．また，キューマルチスカラ演算では Bidirectional SIMD により，依存関係を含む命令列をひとつの命令にすることができる．今後，さらにこの Bidirectional な性質がもつ可能性を考察し，また計算機構の設計を行っていく．

参考文献

[1] Bruno R. Preiss and Carla V. Hamacher. Data Flow on Queue Machine. In *12th Int, IEEE Symposium on Computer Architecture*, pp. 342–351, Boston, MA., Aug 1985.

[2] Herman Schmit, Benjamin Levine, and Benjamin Ylvisaker. Queue Machines: Hardware Compilation in Hardware. In *IEEE FCCM'02*, 2002.

[3] S. Okamoto, H. Suzuki, A. Maeda, and M. Sowa. Design on a Superscalar Processor Based on Queue Machine Computation Model. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 151–154, 1999.

[4] M. Sowa, Ben A. Abderazek, and T. Yoshinaga. Parallel Queue Processor Architecture Based on Produced Order Computation Model. *Supercomputing, HPC*, Vol. 32, No. 3, pp. 217–229, 6 2005.

[5] 弘中哲夫, 岡崎恵三, 村上和彰, 富田眞治. ストリーム FIFO 方式に基づくベクトルプロセッサ「順風」. 情報処理学会論文誌, Vol. 32, No. 7, pp. 828–837, 1991.