

3C-5

ブロック分割記憶管理法による
パイプラインマージソータの機能拡張

楊維康, 伏見信也, 喜連川優*, 田中英彦, 元岡達
 東京大学工学部 *東京大学生産技術研究所

1. はじめに

我々はハードウェアパイプラインマージソータのアルゴリズムの単純性と高速性を維持しながら, その実装, 及びレコード長, レコード数, キー長等の変化に対して柔軟且つ効率的に対処可能なように, 機能拡張についての研究を進めている(〔1〕〔2〕参照). ソートプロセッサの記憶管理方式としてはリスト構造による方式, ブロック構造による方式等が挙げられるが, 我々はリストによる記憶管理及びLength Tuning 機構を有するソートプロセッサを試作し(〔3〕参照), 更にブロック分割記憶管理法について検討した(〔4〕参照).

ブロック分割記憶管理法では, メモリ領域の使用, 空き領域の回収等がレコード長と独立に, ブロック単位で行われる為, パイプラインマージソータにおいて, マルチストリームソート, 可変長レコードソートの実現に於ける記憶管理の問題を解決することができる. 今回はブロック分割記憶管理法を用いたソータにマルチストリームソートと可変長レコードソートを実現する上での問題点について検討し, 又これら種々の機能拡張を容易に実現可能なソートプロセッサの新しい構成法について考察する.

2. マルチストリームソートと可変長レコードソート

マルチストリームソートとは, 連続して入力される複数のストリームを, 入力を中断させることなくパイプライン的にソート処理する機能を指す(図1参照). ここでストリームとは, ソートすべきあるまとまったレコードの列である. 又, 可変長レコードソートとは, 長さの異なるレコードからなるストリームをソートする機能を指す.

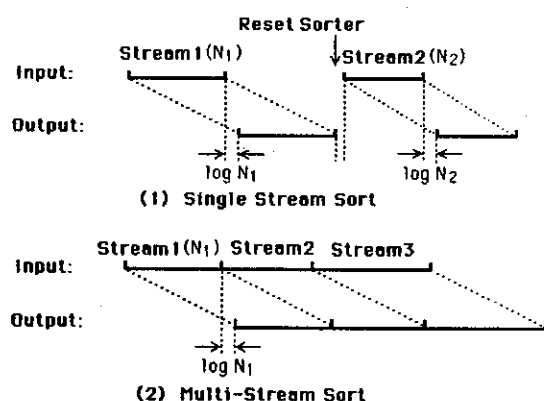


図1. マルチストリームソート

3. マルチストリームソートと可変長レコードソート

の問題点とその解決法

パイプラインマージソートの基本アルゴリズムでは(2-Way), 各ソートプロセッサが第一ストリングをメモリにロードした後, 続いて入力されてくるマージペアとしての第二ストリングとマージして出力を行う(ここで言うストリングは, ソートされたレコードの順序列で, ストリームと区別する). この際パイプラインを維持する為, マージ出力すると同時にデータ入力を続け, 次のマージフェーズ用のデータの準備を行う.

プロセッサは, メモリにロードされたストリングの状態を把握する為, メモリに存在するストリングの数, ストリングの先頭位置, ストリングのレコード数等を管理する. メモリに存在するストリングの数が少ない場合, これら管理情報をストリング毎に保持する制御レジスタ群を用いた制御構成が可能である.

しかし, 可変長レコードのデータや複数のストリームがパイプライン上で流れる時, 一般的な状態を考えると, あるプロセッサのメモリ上に存在するデータはストリング数, ストリング長等が様々で, 従来の様にストリング毎にレジスタを持って管理することは困難である.

そこで一つの解決法としては, データの入力時に, ストリームに関する初期化データ, ストリングの附加情報としてのフラグ等を, データと一緒にメモリに格納することが考えられる. この様にすることによって, メモリに存在するデータはそれ自身に関する情報を保持し, 処理される時点で当該情報がプロセッサによって利用されることになる. 即ち, プロセッサはフラグ等のストリングに関する情報を検出することによって, 処理の状態を把握できる. この様に, プロセッサは同時にメモリ中存在するすべてのストリングの制御情報を管理しておく必要はなく, 現在マージ処理の対象となっている2本のストリングに関する情報のみを持つだけで十分である.

ここでデータのフォーマットは各レコードにストリングの先頭, ストリームの終了等を示すフラグが付加され, 又データにはレコードの境界を示す1ビットのタグが付加される.

4. ソートプロセッサの構成

以上の如く, 入力データは一旦メモリにロードした後処理される. このことから制御を簡単にする為, データの

入力と処理出力を独立にし、この二つの操作を機能的に分離した機能モジュールによって実行するソートプロセッサの構成が考えられる。ここで図2に示す様にデータ入力を司るデータ入力部とマージソート及びデータ出力を司る処理出力部からなるソートプロセッサの構成を提案する。以下にその各部について説明する。

入力部：入力データに処理を加えずにメモリに書き込む。

処理出力部：NSPQの情報により有効データの有無を判断し、あればそれをマージして出力するとともに、データ処理によって生じた空き領域の回収等、メモリ管理を行う。

NSPQ (Next String Pointer Queue)：これは入力部と処理出力部間オートマトン間メッセージ通信の役割を果たすFIFOのキュー構造のハードウェアで、キューへの入力は入力部によって行われ、出力は処理出力部によって行われる。

処理出力部は2本のstringのマージソートを行う際、2つのstringの先頭位置が必要であるが、その内、第一stringの位置はプロセッサの初期状態又は前の処理状態から得られるが、マージ対象となる第二stringの位置はNSPQで管理される。データ入力時、入力部は入力データを監視して、マージ対象の第二stringに当たるstringが入力されると、その格納位置のポインタをNSPQに入れる。処理出力部はマージ処理を始める前にNSPQをアクセスして、第二stringの位置を得る。又、キューが空きである状態はデータが到着していないことを示す。

図2にプロセッサの構成を示す。

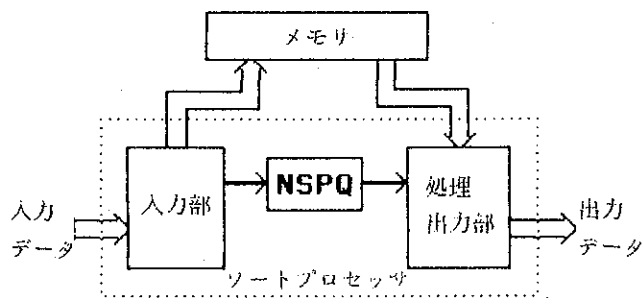


図2. ソートプロセッサの構成

5. 入力部と処理出力部の状態遷移

入力部の状態遷移：入力部は入力データを監視して、データのフラグから現在入力中のstringが第一stringか第二stringかを判断する。第一stringをロードする状態から第二stringをロードする状態に移行する時、第二stringの先頭アドレスをNSPQに入れる（図3）。

処理出力部の状態遷移：処理出力部の状態は大きくデータ待ち状態、データ処理状態、ブロック管理状態と分けることができる（図4）。プロセッサの初期状態はデータ待ち状態、この状態でNSPQをチェックして、NSPQが空でないならば、NSPQから第二stringへのポインタを得て、

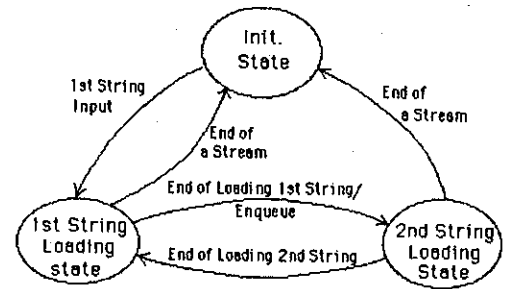


図3. 入力部の状態遷移図

データ処理状態に移移してマージ処理を始める。マージ終了後、又NSPQをチェックして、次マージのデータの有無を判断する。又、データ処理状態に於て、処理が進むにつれ、次ブロックポインタの検出、又は空きブロック回収の要求がある時、随時ブロック管理状態に入って、適当な処置をする。

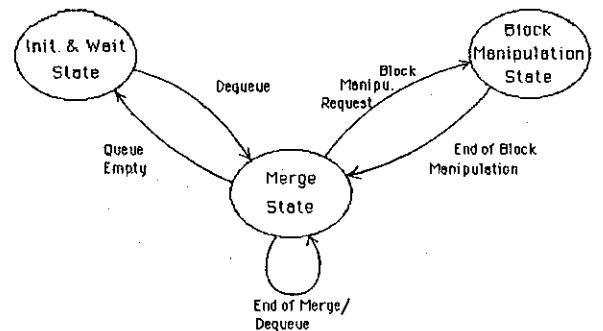


図4. 処理出力部の状態遷移図

6. 結び

ブロック分割記憶管理法に基づき、マルチストリームソートと可変長レコードソートが可能なハードウェアマージソータの新しいプロセッサ構成法を提案した。レジスタランスタフェレベルのアルゴリズムを記述し、シミュレーションを行って、アルゴリズムの正当性を確認している。又、可変長レコードソート時のLength Tuning アルゴリズムについても検討を終えており、これらの詳細については稿を改めて発表したい。

<参考文献>

- (1) 伏見他：GRACE に於けるソーティングユニットの機能拡張 第24回情処全大, 4G-4 (1982)
- (2) 林他：パイプラインマージソートアルゴリズムの改良によるソート時間の短縮 第29回情処全大, 3F-8 (1984)
- (3) 楊 他：Length Tuning 機構を有するハードウェアマージソータの設計 第30回情処全大, 1D-8 (1985)
- (4) 楊 他：ブロック分割記憶管理法によるパイプラインマージソータ 第31回情処全大, 1B-9 (1985)