パイプラインマージソータのＬＳＩ化

4F-4

林　隆史　喜連川　優　伏見　信也　田中　英彦　元岡　達
（東京大学　工学部）

1.　はじめに

我々はパイプライン化されたマージソートアルゴリズムによるO（N）ハードウェアソータのＬＳＩ化を行なっているが、ＬＳＩ化に於いてはチップの結合が重要となる。今回はＬＳＩソータを複数個用いた大容量ソータの構成法、並びにレコード長に対する制御方式についてその実装技法を検討する。

2.　ソートチップの結合方式

現在は１つのプロセッサのＬＳＩ化を行なっているが、将来集積度が向上すれば複数のプロセッサからなるソータを１チップにすることが可能となる。１チップ化されたソータを複数台結合して任意の規模のソータを容易構成できることが好ましい。今、ソータのソート可能なレコード数をM、入力ストリーム長をYレコードとすると、Y＞Mの場合ソータを［（Y－１）／（M－１）］台用意し、一次元状に接続する。複数台ソータの連結はソータレベルでのマージ操作を行なえばよいが単なるマージ（Merge　Connection）では各ソータにランダムに空エリアが生ずることになる。パイプラインマージソートはソータの容量を単位とするパイプライン処理が可能な点に特徴があるが、先の結合ではその処理は容易ではない。

ここでは、Stream Driven Connection なる結合方式を提案する。本方式では図１に示される如く空エリアは先頭のソータから順次生成されて行き、ソータレベルでのパイプライン処理が可能である。結合処理は各ソータにM－１個のレコードを貯えるSort Phase と各々のソータを一斉に結合するConnection Phaseに分かれる。

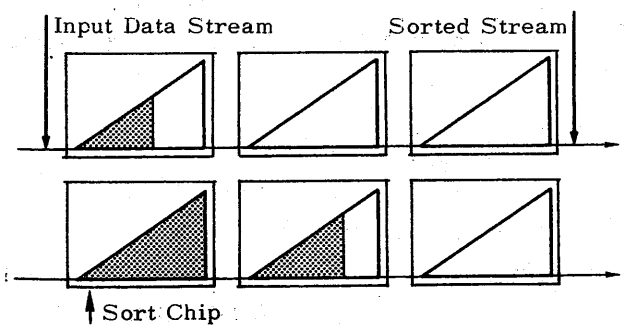

Fig.1 Stream Driven Connection of Sort Chips

2.1 Sort Phase

各ソータはM－１個のレコードを貯えるとwait状態に入いり、それ以後のレコードは次ソータへバイパスする。これはＲＮＣなるカウンタを設け入力レコード数を計数することにより制御される（図２）。

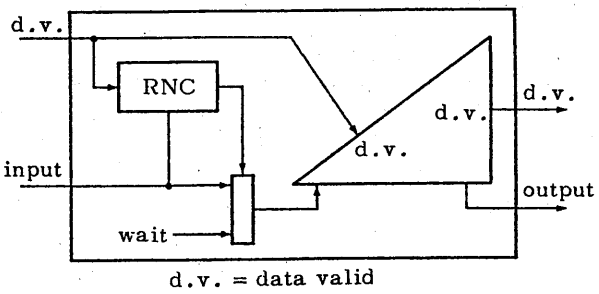

Fig.2 Structure of Sort chip

2.2 Connection Phase

本Phaseでは各ソータのソートストリングのマージを行なう。最終レコードの先頭には eos（end of string）フラグが付けられ（図３）、マージの契機はこのフラグの入力により与えられる。eos は最終ストリングの先頭に付加される様制御される。この制御はプロセッサPi がeos の入力された時点でPi 内のフラグeosfを立て（eosf＝１）、マージ出力を開始する時点でeosf＝１ならばeos をフラグ領域に付加して出力することにより実現できる。又、ストリームの最後にはその区切りを示すeor フラグが設けられる。プロセッサはeor が入力されると、現在マージ中のストリングを出力後にeor を付加して処理を終える。尚、これらフラグの処理は従来のポインタ操作時間内に埋め込むことが出来、時間的オーバヘッドは増加させずにすむ。

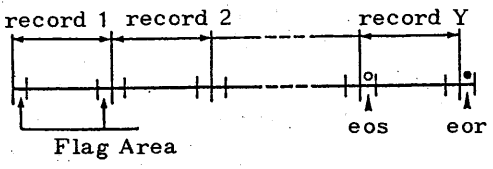

Fig.3 Format of Input Data Stream

2.3 ストリームバイパス機構

一般に最終段のソータに対するレコード数YはMより少なくなる。この場合ダミーレコードを付加し

Y＝Mとしてソートを行なうことも可能であるが、不要なプロセッサをバイパスすることで高速化が可能である。図４に示す如くソート時間は

$$Y+M+\log_K M-1$$

から

$$Y+K^{\lceil \log_K Y \rceil -1}\cdot\left\lceil \frac{Y}{K^{\lceil \log_K Y \rceil -1}} \right\rceil +\log_K M-1$$

に改善できる。このバイパス機構は入力ストリームの先頭レコードでeos を検出したプロセッサがストリームを素通りさせることで実現できる。

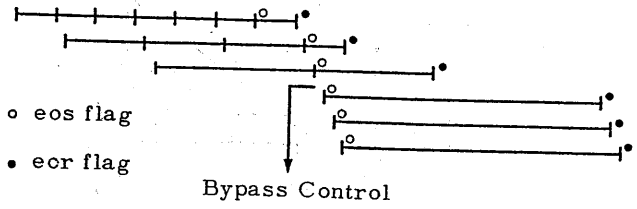

Fig.4 eos & eor Flag Manipulation

3. レコード長の変動に対する制御

試作ソータでは比較器入力レジスタMTRは１レコード全体が格納される様になっているが、MTRにはキー部のみを格納するようにすれば、非キー部に対する大きさの制限は解消される。今、MTRの長さをLバイト、入力ストリームのレコード長をXバイトとし、レコードのキー部を含む先頭のLバイトを改めてキー部と呼ぶこととすると、レコードの配置法としてキー部をMTRに非キー部（X－Lバイト）をメモリに置くことが考えられる。このときキー部はMTR上に存在しておりメモリへのアクセス回数は増加せず１バイト／２クロックの処理レートを維持できるが、比較終了以前は入力レコードを格納するメモリアドレスが定まらない。そこで以下のリソースを追加しメモリ制御を行なう。

STPX…入力されてくるレコードの為にwrite buffer領域をとる。そのアドレス保持を行なうレジスタ。

MARW…入力レコードをwrite bufferに書込むための書込み専用アドレスレジスタ。

MARR…メモリからの読み出しに用いる読み出し専用アドレスレジスタ。

その他に非キー部の先頭アドレス（STP＋L）を計算するための加算器が必要である。メモリ制御部の構造は図５のようになる。そして、ストリング管理を行なうポインタ操作の例を図６に示す。

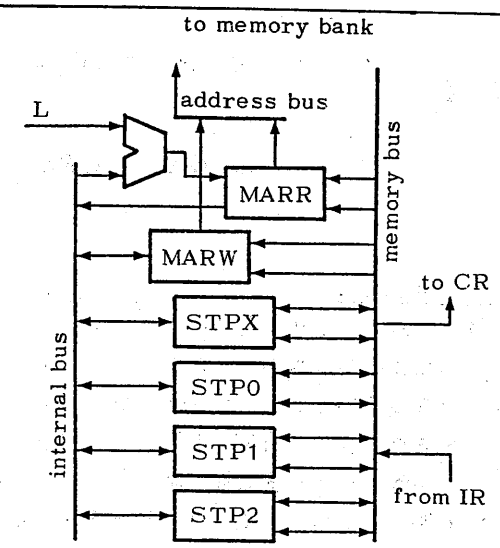

Fig.5 Organization of the Memory Control Block

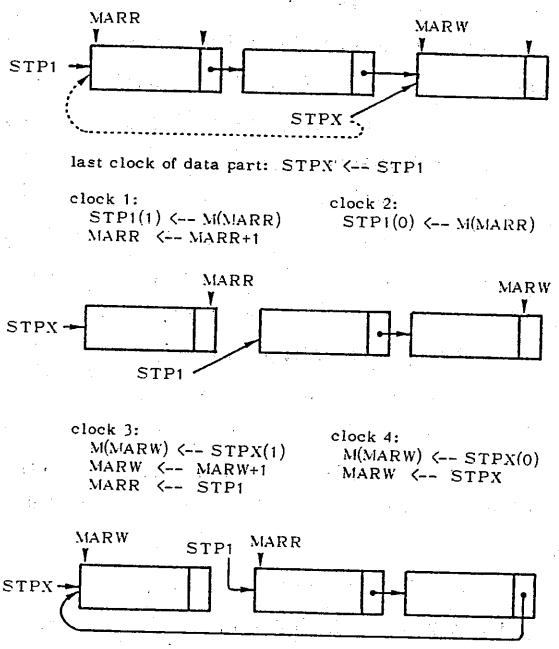

Fig.6 An Example of the Pointer Manipulation

4. おわりに

現時点での集積度の制約から開発中のソータでは１チップに１つのプロセッサを入れた形になっている。今後、拡張機能を順次実装していく予定である。

『 参考文献 』

［１］喜連川他：
パイプラインマージソータの構成
信学技報 EC82－32 1982

［２］林他： パイプラインマージソータの構成
＝ＬＳＩ化に関する一検討＝
情報処理学会第25回全国大会 4P-4 1982