ＩＸＭにおける連想メモリ上の並列演算性能

7U-3

国分明男、樋口哲也、古谷立美

電子技術総合研究所

１．はじめに

近年、半導体ＬＳＩ技術の進歩によって、大容量の連想メモリが作られるようになってきており [1-3]、今後の進展も期待できる状況にある。意味記憶システムＩＸのハードウェアプロトタイプＩＸＭ [4]には、大容量の連想メモリ[3] システムが備えられている。一方、連想プロセッサの観点から、連想メモリ上での基本論理演算レベルにおける並列処理アルゴリズムの提案が既になされている [5-6]。意味ネットワーク処理等のＡＩ応用においては、論理操作のみならず演算操作も含まれる場合が多いと考えられるので、本論文では、ＩＸＭにおける連想メモリ上の、それら各種並列演算アルゴリズムの性能を試算する。

２．並列演算アルゴリズム

プログラミングモデルは、次の通りである。連想メモリはワード構成であり、各ワードに一致を示す検索フラグビットが付属している。同一ワードにおける新たな検索結果と、検索フラグビットの内容または上位ワードの検索フラグビットの内容との間でＡＮＤ／ＯＲ操作が可能である。検索データ用マスクレジスタ、書き込みデータ用マスクレジスタが備えられている。通常のメモリと同様にアドレス指定で書き込みおよび読み出しができる他に、図１に示す基本操作が可能である。

同一ワード上の二つのフィールドに置かれた値を並列加算するには、図２に示すように、キャリーフィールドを別に用意して、対象フィールドの各ビット毎に、以下の操作を繰り返すことによって行う。加算前のビットパターンのうち加算後にビットパターンが変化するものが４種類存在する。それらを順次、検索してフラグを立て、フラグが立ったワードに変更ビットパターンを並列書き込みする。並列書き込み後に、変更ビットパターンを再び検索しないような順序にする。同一ビットパターンを２回続けて並列書き込みする場合、ＡＭ．ＯＲ．ＳＥＡＲＣＨ操作を用いて、検索フラグのＯＲを取り、並列書き込みを一回減らすようにする。

連想メモリの同一ワード上の二つのフィールドに置かれた値を並列掛算するには、図３に示すように、キャリーフィールドと結果を格納する積フィールドを用意して、乗数フィールドの各ビットにもとづいて被乗数フィールドの内容と積フィールドの内容との間で、シフト加算を繰り返すことによって行う。シフト加算は、シフトして加算する操作であり、上記の並列加算においてたしこむ際に、積フィールドの検索および書き込み対象のビット位置を、ずらして操作することによって実現される。

種々の応用において、ワード間で演算を行う必要性がある場合も存在している。連想メモリ上でワード間加算を行うためにカウントレスポンダを用いる方式が既に提案されているが[5] 、ＩＸＭの連想メモリにはカウントレスポンダが備えられていないので、上位ワードの検索フラグビットの内容との間のＡＮＤ検索によって、隣接ワード間で加算を繰り返すことによって実現する。その際に、隣接ワード間で加算を走査的に繰り返す方法と二進木的に繰り返す方法が考えられるが、両者に大きなステップ数の差はない。カウントレスポンダを用いる方式と比べると、加算が逐次的になる欠点はあるが、異なるワードグループで並列的に演算を行えるという利点がある。

３．連想メモリ上の並列演算性能

連想メモリはＩＸＭのＰＥボード当たり、４０ビット×４０９６ワード構成である。検索データ用マスクレジスタのビット幅は４０ビット、書き込みデータ用マスクレジスタのビット幅は４バイトデータの各バイトについて１ビットとタグが８ビットの合計１２ビットである。このため、ビット毎の並列書き込みを伴う並列演算アルゴリズムでは、最大１２ビット幅までの処理となる。図４に、上記の各アルゴリズムの並列演算性能を示す。実際の処理時間は、トランスピュータ上のＯＣＣＡＭで記述されたプログラムのオーバーヘッドを含むので、連想メモリに対する操作時間５００ナノ秒×処理ステップ数に基づく推定の数倍（執筆時点の実測値では８倍程度）になる。

５．おわりに

連想メモリによるフィールド間並列演算は、対象となるワード数が多く、フイールドあたりのビット数が少ないほど有利である。このことは、意味ネットワーク処理、ニューラルネットワーク処理のような応用において、大規模化を実現しようとする際に、連想メモリによる処理の検討を行うべきであることを示している。

Performance of Arithmetic Algorithms on an Associative Memory on the IX Machine

A.kokubu, T.Higuchi and T.Furuya

Electrotechnical Laboratory

謝辞

連想メモリについて援助して頂いたＮＴＴ－ＬＳＩ研究所小倉武氏、本研究の機会を与えられた当研究所棟上情報アーキテクチャ部長に感謝する。

参考文献

[1] T. Ogura, S. Yamada and T. Nikaido, "A 4-kbit Associative Memory LSI", IEEE Journal of Solid-State Circuits, Vol. SC-20, No. 6, pp.1277-1282, 1985.

[2] H. Kadota, J. Miyake, Y. Nishimichi, H. Kudo and K. Kagawa, "An 8-kbit Content-Addressable and Reentrant Memory", IEEE Journal of Solid-State Circuits, Vol. SC-20, No. 5, pp. 951-957, 1985.

[3] 小倉武、山田慎一郎、山田順三、長沼次郎、丹野雅明、"20Kb CAM (Content Addressable Memory)"，電子情報通信学会技術研究報告、Vol.87, No.349, pp. 31-37, CPSY87-33, 1988.

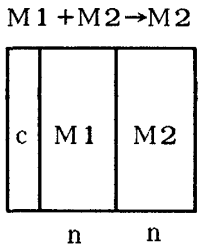
[4] 樋口哲也、古谷立美、楠本博之、半田剣一、国分明男、"意味記憶システムＩＸ（イックス）のプロトタイプについて"、情報処理学会研究報告、Vol.87, No.63, 知識工学と人工知能 54-8、1987.

[5] C. C. Foster, "Content Addressable Parallel Processors", Van Nostrand Reinhold Company, New York, 1976.

[6] S. Ilgen and I. D. Scherson, "Parallel Processing on VLSI Associative Memory", Proceedings of the International Conference on Parallel Processing, pp.50-53, 1987.

基本操作	意味
AM.SEARCH	検索し、一致ワードにフラグを立てる
AM.AND.SEARCH	検索し、フラグとのＡＮＤをとる
AM.OR.SEARCH	検索し、フラグとのＯＲをとる
AM.LAND.SEARCH	検索し、上位フラグとのＡＮＤをとる
AM.LOR.SEARCH	検索し、上位フラグとのＯＲをとる
AM.PAR.WRITE	フラグがオンのワードに並列に書込む
AM.SEARCH.MASK	検索マスクパターンをセットする
AM.WRITE.MASK	書き込みマスクパターンをセットする

図１　連想メモリに対する基本操作

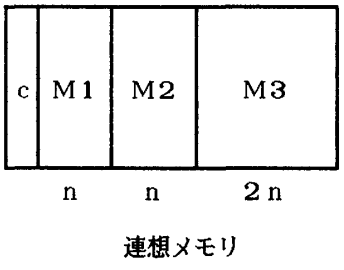


ｍ１とｍ２のビット位置を変えながら、以下の操作をｎ回行う。

```
AM.SEARCH.MASK
AM.WRITE.MASK
AM.SEARCH(c=1,m1=0,m2=0)
AM.PAR.WRITE(c=0,m2=1)
AM.SEARCH(c=1,m1=0,m2=1)
AM.OR.SEARCH(c=0,m1=1,m2=1)
AM.PAR.WRITE(c=1,m2=0)
AM.SEARCH(c=0,m1=1,m2=0)
AM.PAR.WRITE(c=0,m2=1)
```

図２　フィールド間並列加算アルゴリズム

ここで、ｃはキャリー用の１ビット、ｍ１、ｍ２は各々Ｍ１、Ｍ２フィールド中の対応する１ビット。



Ｍ２フィールド中の１ビットｍ２にもとづいて、Ｍ１フィールドの内容とＭ３フィールドの内容との間で、並列シフト加算をｎ回行う。

図３　フィールド間並列掛算アルゴリズム

	ステップ数	並列演算時間（μS）	
		４ビット	８ビット
並列加算	９ｎ	１８	３６
並列掛算	９n^2	７２	－－

図４　連想メモリ上の並列演算性能

ここで、ｎは演算対象ビット数、ステップサイクル時間 500ns。演算時間はワード数に依存しない。