計算機アーキテクチャ 120－1
（1996．10．31）

Memory String Architecture
― メモリウォールを越えて ―

松本 尚 平木 敬

本稿では、まず高速シリアルリンクをプロセッサ・メモリ間インタフェースに使用することでメモリウォール問題が緩和されることを示す。そして、複数の高速シリアルリンクを入出力に持つ Multi-ported Serial-Access Memory チップ（MSAM）と MSAM を内蔵したメモリ／ロジック混載 LSI で作成される Multi-ported Serial-access Processor チップ（MSP）を提案する。最後に、MSP ならびに MSAM で構成される単一および並列システムである Memory String アーキテクチャに関して論じる。

Memory String Architecture
― Beyond the Memory Wall ―

TAKASHI MATSUMOTO and KEI HIRAKI

We adopt high-speed serial links for processor-memory connections in order to solve the memory wall problem. Therefor we propose two novel devices that have high-speed serial interfaces: the Multi-ported Serial-Access Memory (MSAM), which is an extended DRAM chip, and the Multi-ported Serial-access Processor (MSP) that includes MSAM using combined DRAM and logic technology. Finally, the Memory String Architecture that consists of MSP chips and MSAM chips is introduced and discussed.

1. はじめに

プロセッサチップのメモリバンド幅の不足によるボトルネック問題である「メモリウォール」が昨今注目を浴びている。今回のメモリウォールブームの火付け役となった最初の文献 [1] はプロセッサおよびメモリの性能向上が過去とまったく同じ勢いで続くと仮定した非常にナイーブな議論であった。この仮定の下では近い将来メモリのアクセス速度がボトルネックになってシステムのスピードアップが不可能になると論じている。その後、精密なシミュレーション結果から、高性能プロセッサではメモリウォールはすでにボトルネックとなり始めているという研究結果が報告された [4] [5]。そして、プロセッサとメモリ（主記憶）の混載チップの採用をメモリウォールの解決策として掲げている。

本稿では、複数の高速のシリアル通信を外部メモリ用として利用すれば、プロセッサチップのメモリバンド幅を大幅に広げることが可能となりメモリウォールの問題の緩和が可能なことを示す。次に、複数の高速シリアル通信をメモリインタフェースとするのに適したメモリデバイスである Multi-ported Serial-Access Memory（MSAM）チップを提案し、これと組み合わせて使用するプロセッサと MSAM を混載した Multi-ported Serial-access Processor（MSP）チップを提案する。MSP では内部大容量メモリをページ単位キャッシュとして保持するが、流行のプロセッサ／メモリ混載チップとは異なり外部に主記憶 (MSAM) を接続可能である。最後に、MSP と MSAM を組み合わせて構成する Memory String アーキテクチャに関して議論する。

2. メモリウォール問題

本節では、昨今のメモリウォールに関する議論を簡単にまとめる。

前述したように 1995 年 3 月に Wm.A.Wulf と S.A.McKee が現在のプロセッサのスピードアップとメモリ（主記憶の DRAM）のスピードアップの比率が将来にわたって続くとすると、キャッシュミスヒット率を十分小さく仮定しても、近年中にシステムのスピードアップはメモリアクセスの相対的な遅さのために難しくなると指摘した [1]。プロセッサのスピードアップ率、メモリのスピードアップ率、キャッシュのミスヒット率を固定した議論であり、議論自身は非常に単純かつナイーブなものであった。

これに対して、M.V.Wilkes は CMOS テクノロジの限界が迫っているので、現在のプロセッサやメモリのスピードアップ率が将来も有効であると考えることに

† 東京大学 大学院理学系研究科 情報科学専攻
Department of Information Science, University of Tokyo

は無理があると指摘した [2]。また、E.E.Johnson は Wilkes と同様にテクノロジの限界の問題を指摘するとともに、キャッシュが大きくなりワーキングセットが完全にキャッシュ上に乗る場合には、長時間走行するプログラムのキャッシュミスヒット率は限りなくゼロに近付くことを指摘した。さらに、将来のシステムのスピードアップが並列処理でなされるのであれば、メモリウォール問題は解消されてしまうとも述べている [3]。

Wulf らのメモリウォールに関する問題意識を支持する論文として、Burger らの論文 [4] と Saulsbury らの論文 [5] がある。

Burger らは最近のレイテンシ隠蔽テクニックがすべてプロセッサ・メモリバンド幅を増大させる傾向があることを示し、メモリアクセスレイテンシによるプロセッサストールを純粋なレイテンシによるものとバンド幅不足によるものに分けてかなり精密なシミュレーションを行っている。その結果、アプリケーションとレイテンシ隠蔽技法に依存するが、バンド幅不足によるストールが実行時間の 3 割に達するものが存在する。このシミュレーションは現在のプロセッサやメモリ技術をベースにしたパラメータで行っており、Wulf 同様の方法で将来はずっとバンド幅不足が深刻になることを指摘している。さらに、Burger らはキャッシュが有効に使われているかどうかについてもシミュレーション実験をしており、その結果キャッシュを最適に制御することで 1 桁から 2 桁ぐらい主記憶へのメモリアクセスを減らすことが可能であるという結果を出している。結論として、近い将来的にはプロセッサチップ上のオンチップキャッシュを最適化制御することで外部メモリアクセスバンド幅を有効に使い、将来的にはプロセッサ／メモリ（DRAM）混載チップで主記憶をプロセッサと同一チップに閉じ込めてしまうことを提案している。

Saulsbury らはメモリウォール問題を解決する方法として、プロセッサ／メモリ（DRAM による主記憶）混載チップの構成をかなり具体的な形で提案している。まず、論理合成プログラムのような大きなワーキングセットを持つアプリケーションでは、ローエンドマシンである SS-5 が同時代のハイエンドマシン SS-10/61 よりも速いことを引合に出して、メモリとプロセッサの密な結合の必要性を強調している☆。そして、自分達の提案しているプロセッサ／DRAM 混載チップのメモリ構成が通常のキャッシュを含むメモリ階層を持つシステムに見劣りがしないことをシミュレーションで示している。Saulsbury らのチップ内には通常の方式の SRAM ベースのキャッシュは含まれず、DRAM のセル☆☆に付随する row 幅のバッファ（データ用 2 本、命令用 1 本）がキャッシュの役割を兼ねる方式になっている。このため、ラインサイズが極めて大きい（論文では 512 バイト）が、このラインサイズのメモリフェッチが 1 内部 DRAM サイクルで済む。また、同一ラインのキャッシュリプレースによる性能低下を防ぐために小容量の victim cache [6] がチップ内に内蔵されている。シングルプロセッサとしてのメモリシステムの評価の後、マルチプロセッサとして DASH をベースとした CC-NUMA [7] と比較して評価を行っている。

なお、メモリウォール問題と直接関連させてはいないようであるが、日本においてもプロセッサ／主記憶混載チップの開発 [8] や提案 [9] がなされている。

☆ SS-5 はメモリコントローラがプロセッサに内蔵され、メインメモリのアクセスレイテンシは SS-10 の 5 分の 1 である。

☆☆ 16 程度の分割されたセルで大容量 DRAM が構成されることを前提としている。

3. メモリウォール問題およびプロセッサ／主記憶混載に関する疑問点

前節でも述べたように、すでに Wilkes や Johnson によっていくつかの疑問や反論がなされている。これらの点には筆者らも同様に問題を感じているが、本節では他の疑問点について議論する。

まず、第一に Burger や Saulsbury が主張するようにチップのメモリバンド幅の問題が深刻さに関する現状認識に関する疑問である。詳しくは次節で議論するが彼らの前提としたパラレルバスインタフェースではなく高速シリアルインタフェースを採用すれば、ピン当たりのバンド幅は 1 桁以上現在の技術で向上可能である。とすれば、シミュレーションはパラメータを変更して再検討の必要があり、メモリバンド幅の不足に悩む前に Wilkes の言うように CMOS 技術の限界をクリアしなければならない可能性が高い。そして、ベースになるテクノロジの変化が判明する前に、テクノロジ変化後のチップを議論してもナンセンスである。

筆者らは以前より物理的な近傍性や局所性等の高速化に有利になる条件は可能な限り利用すべきである [10] [11] という立場であるので、プロセッサチップ内に大容量メモリが搭載可能になれば当然活用すべきであると考えている。しかし、主記憶を全部チップ内に閉じ込めるという発想は論理的に飛躍がある。つまり、Burger らや Saulsbury らが外部メモリインタフェースを付けることを放棄した論理には納得がいかない。

組み込み用等のローエンド用のプロセッサ／主記憶混載チップに関しては、我々もなんら疑問を持っていない。しかし、汎用の高性能プロセッサチップに関しても主記憶全部を含むべきであると結論付けるためには以下の条件が必要なはずである。

(1) メモリ不足を混載チップを増やして並列処理することで補おうとしているため、複雑な処理（例えば論理合成）でも簡単に効率良い並列処理プログラムが開発可能である必要がある。

(2) チップ内に搭載可能となった大容量メモリをキャッシュとして利用した場合でも、チップ外メモリバンド幅が不足することを示す必要がある。少なくとも、Burger らのシミュレーションでは 4Mbyte 程度のキャッシュがあれば、かなり状況が改善されることを彼らのグラフは示している。

(3) チップ内の大容量メモリへのデータの出し入れが

ボトルネックにならないことを示す必要がある。

(4) 現実問題として、汎用高性能プロセッサメーカとメモリメーカが分離している状態で、汎用のプロセッサ／メモリ混載チップが量産効果をそのまま反映した低価格で生産可能であることを示す必要がある。

筆者らはこれらの条件が満たされているとは考えていないため、チップ内で利用可能になった大容量メモリは大容量キャッシュ☆として使用する方が優れていると考えている。

4. シリアルリンクによるメモリインタフェース

Saulsbury らはチップ外と通信するために自分達の提案するチップに 2.5Gbit/s のシリアルリンクを複数本持たせている。このリンクの媒体は光ファイバであるが、CMOS 技術で直接 2.5Gbit/s のスピードで光インタフェースをドライブしている。チップ上に占める面積も 0.65μm のルールで 1.6mm^2 程度である。この例では光インタフェースのコストが問題となるが、最近ツイステッドペアケーブルもしくは基板上の銅配線による、4.0Gbit/s の高速通信の可能性が Dally らによって示された [12]。Dally らは CMOS 技術のみを使い、チップ内の five-tap の FIR フィルタで出力波形を整形することで高速通信を可能にしている。論文発表の時点では、まだ実装が完成していないが、送信回路のチップ上の大きさは 0.6μm のルールで 0.5mm^2 程度である。

Dally らの主張通り超高速シリアルリンクが小さなコスト（チップ上の面積）のみで実現可能であれば、プロセッサ間通信やワークステーション間通信のみではなく、プロセッサ・メモリ間のチップ間接続にこの技術を応用しない手はない。プロセッサチップの外部ピン 1 本当たり、4Gbit/s（ないし 2Gbit/s）の入出力性能があることになり、Burger がシミュレーションに使った 0.1Gbit/s の値とは大きな開きがあることになる。一次キャッシュ（や必要なら二次キャッシュ）は大容量メモリ混載ロジック LSI が実現可能であれば当然チップ内に内蔵されるので、Dally らも主張している通り数本の外部シリアルリンクがあれば現状のメモリバンド幅を賄うことが可能である☆☆。超高速シリアルリンクをメモリインタフェースに使用すれば、少なくとも Burger や Saulsbury が予想したよりはメモリウォールの問題が深刻化するのを遅らせることが可能であることは間違いない。そして、CMOS の技術限界や発熱といった他の性能の上限を決定する要因と、メモリウォール問題の深刻化とどちらが最初に訪れるかは、現時点では筆者らには判断不可能だと考えている。

☆ 必ずしも HW のみでキャッシュ機能が実現されていることを意味しない。

☆☆ Burger らのシミュレーションでは彼らの設定した現状のパラメータでさえ、メモリバンド幅はボトルネックになっていない。彼らは現在の技術や手法をそのまま外挿して将来の不安を語ってるだけである。

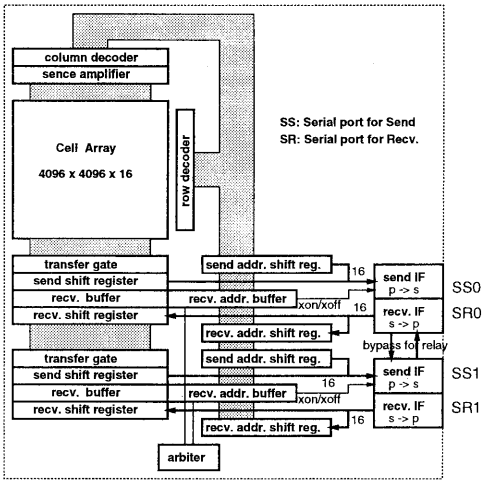

図1 MSAM チップの構造

5. Multi-ported Serial-Access Memory

シリアルアクセスが可能な一般メモリに VRAM（または VDRAM）がある。VRAM は DRAM のメモリセルが 1row 分のアクセスが 1 度に可能なことを利用して、1row 分のデータをシリアル入力から受け取るごとに実際のメモリセルに書き込み、1row 分のデータを 1 度にバッファに読み出しシリアル出力から送出する。シリアルアクセスのタイミングクロックはチップ外部から入力し、row アドレスの指定はパラレル入出力用のアドレスピンと共用している。

この VRAM を超高速シリアルリンクを複数持った **Multi-ported Serial-Access Memory (MSAM)** に拡張する☆☆☆。Litaize らによるボードレベル（従来メモリチップを用いたマルチチップ構成）で複数のシリアルインタフェースを持ったメモリシステム [13] が提案されており、これが今回の拡張に参考になる。このボードレベルのメモリシステムとの最大の差異は VRAM のようにメモリセルの 1row 分のデータを 1 度にアクセスするチップ内のバンド幅の広さが活用できる点にある。また、Litaize らは双方向のシリアルリンクの採用を許し、グローバルクロックの存在を仮定しているが、超高速シリアルリンクを採用する我々は単方向のシリアルリンクのみを採用し、位相のそろったグローバルクロックは仮定しない。図 1 に MSAM の内部ブロック図を示す。我々の MSAM ではクロックはチップの受信インタフェース部分でシリアル通信自体から復元され、デバイスのクロックと同期を取ってデータを受け渡す。リンクのフロー制御にフロー制御専用の外部ピンを割り当てることも可能であるが、メモリチップ自体のピン数の削減ならびにチップ間接続の配線数および中継スイッチ用チップのピン数の削減のために、シリアルリンクの接続は入出力共に各 1 本の point-to-point

☆☆☆ ただし、パラレルポートは不用になる。

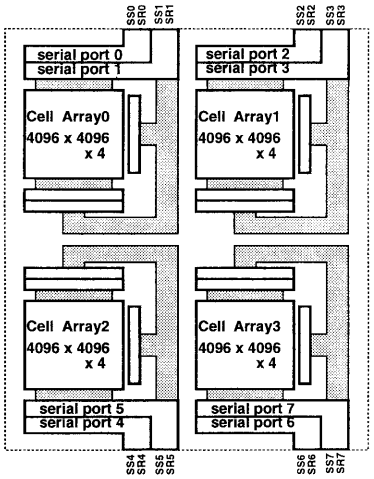

図2　内部4バンク構造のMSAM

接続で他のデバイスと接続する。そして、フロー制御はXON/XOFFメッセージを多重化して転送する☆ことで行う。他に、MSAMがVRAMのシリアルポートと異なる点はアドレスやコマンド情報もシリアルリンクで送られてくる点がある。また、最近のVRAMのシリアルポートは任意のアドレスからのシリアルアクセスが可能になっているが、この機能がクリティカルパスおよびチップ面積の浪費につながる可能性があるため、MSAMではrow境界もしくはその公約数のうち大きな数の境界☆☆でしかアクセスを認めない。

ちなみに、16Mbitのメモリセル16枚からなる256Mbit MSAMを考えるとする。一度に4096 × 16 bit = 8192byteの読み書きがチップ内で可能である。これを4Gbit/s,8B10Bエンコードのシリアルリンクで送ったとすると、実質400Mbyte/sの転送が可能であり、約20μsで8192byteのデータの転送が可能である。レイテンシを小さく抑えるために256byte単位の転送をシリアルリンクでサポートすれば、約600nsで転送が可能となる。メモリセルのサイクルタイムを100nsとすると、8192byte単位転送であれば約200本のシリアルリンクがサポート可能である。256byte単位転送の場合は、メモリセル1枚のrowデータ幅で足りるので、複数のメモリバンクが1チップに内蔵された構成（図2参照）にすれば、600 ／ 100 × 16で約100本のシリアルリンクを接続することができる。シリアルリンクのハードウェア量や発熱を無視して、メモリセルがサポートできる最大数のリンクを接続した場合を考えると、この最大構成のMSAMチップは1チップ当たり、8192byte転送時で80Gbyte/s、256byte転送時で40Gbyte/sのデータ転送能力がある。

MSAMは単に標準化された超高速シリアルリンクを

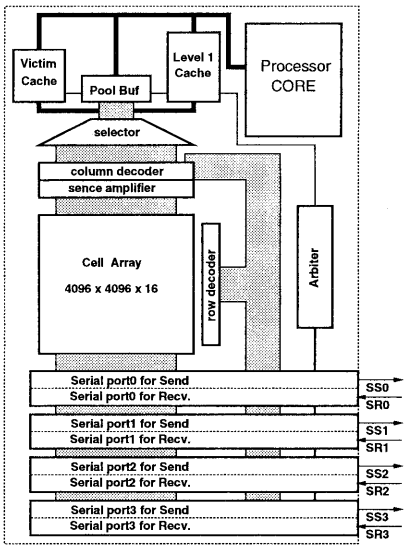

図3　MSPチップの構造

複数持ったメモリデバイスであり、プロセッサ内蔵型のメモリデバイスよりも標準化の障壁はかなり低いと考えられる☆☆☆。

6. Multi-ported Serial-access Processor

前節で複数の高速シリアルリンクを持ったメモリデバイスであるMSAMを提案した。これに対応してプロセッサチップ側にも高速シリアルリンクのインタフェースを持つ必要がある。大容量のメモリ／ロジック混載LSIの技術を仮定しているので、もっとも簡単な方法はMSAMとプロセッサコアとSRAMベースのキャッシュを1チップに混載してしまう方法である。この内蔵MSAMには、SRAMベースのプロセッサキャッシュ用のパラレルポートが通常のMSAMのシリアルポートの他に存在する。このタイプのプロセッサチップを**Multi-ported Serial-access Processor (MSP)**と呼び、構成例を図3に示す。

図3ではDRAMベースの内蔵MSAMは二次キャッシュとして機能させ、SRAM量削減と高速動作のためにに一次キャッシュはダイレクトマップとし、それを補うフルアソシアティブ（または4ウェイ以上のアソシアティビティ）で小容量のvictim cache [6]ならびに一次キャッシュのブロックサイズの4倍程度の幅の（ただしrow幅以下）の小容量pool buffer [14]を実装してある。一次キャッシュのブロックサイズは実装依存であるが、pool bufferが存在するため32byte以下の小さなブロックサイズを採用する。本例では一次キャッシュはライトバック方式のキャッシュを想定しているが、高速

☆ シリアルリンク上では4B5B等のエンコードがなされたデータを転送するため、少数の制御メッセージを他のデータと多重化して送ることが可能である。

☆☆ どの単位でアクセスを認めるかは実装依存。

☆☆☆ 汎用品としてのプロセッサ／メモリ混載チップを否定しているわけでは決してなく、筆者らの提案しているMemory-Based Processor [10]などは、可能であればメモリチップ内に実装すべきロジック回路である。

実装が難しい場合はライトスルー方式として一次キャッシュよりブロック幅の広く容量の大きいライトバック方式のSRAMベースの二次キャッシュを用意する。この場合、一次キャッシュのpool bufferの幅は二次キャッシュのブロックサイズ以下であり、内蔵MSAMは三次キャッシュとして機能する。

二次キャッシュである内蔵MSAMの制御方式は色々考えられるが、本例ではページベースで管理するキャッシュを採用する。ページサイズが4Kbyte程度の場合、大容量の内蔵MSAM全体をカバーするTLBをMSP内に作ることは速度的にもコスト的にも難しい。そこで、ページ存在には以下の4種類の場合が考えられる。

(1) TLBにヒットして内蔵MSAMにも存在
(2) TLBにミスしたが内蔵MSAMには存在
(3) TLBにミスして内蔵MSAMにも存在しないが外付けMSAM（主記憶）に存在
(4) TLBにミスして内蔵MSAMにも存在せず外付けMSAMにも存在しない

TLBヒットは当然ハードウェアのみでメモリアクセスが処理されるが、他のケースについてどこまでハードウェアで処理して、どこから割り込みベースのソフトウェアで処理するかは実装依存である。(3)のケースにMSPはシリアルリンクを通して外づけのMSAMにアクセス要求を発行する。また、レイテンシを考慮してデータの転送単位をページの大きさより小さくした場合は、内蔵MSAMのデータ転送単位ごとにvalid tagが付随し、ページ転送は目的アドレスを含む部位から行われる。

MSPの内部をマルチプロセッサ構成にすることが可能であり、SRAMベースのキャッシュはプロセッサ単位に設け、内蔵MSAMによるページ単位キャッシュは共有キャッシュとする。このメモリ／プロセッサ混載のオンチップマルチプロセッサを**Multi-ported Serial-access MultiProcesor (MSMP)**と呼ぶことにする。MSMP内の各プロセッサ固有のSRAMベースキャッシュのコヒーレンス維持方式は基本的に共有バスによるスヌープ方式を用いる。ただし、アドレスとオペレーションのみが共有バスで転送されれば十分であり、具体的なスヌーププロトコルは実装依存である。

超高速リンクは外部メモリアクセス（ページ入出力）のみに使用されるわけではなく、MSP間の通信や外部入出力にも使用される。これらの用途に使用可能なようにメッセージ受信時に割り込みをプロセッサに発行する機能が受信インタフェースに存在する。I/Oデバイス側にもMSAMと同じ高速シリアルインタフェースが設けられている。実装的には各I/OデバイスにはMSPベースのコントローラが付随しており、バッファリングされたデータが高速シリアル通信で送られることが設計等が共通化できてメリットがある。MSP間通信に関しては次節で述べる。

MSPの内蔵MSAMで容量が足りる用途には当然外づけのMSAMは不要であり、MSPチップ単体で高速のシリアルリンクによる外部入出力を持つ小規模コンピュータシステムが構成できる。内蔵MSAMでは容量

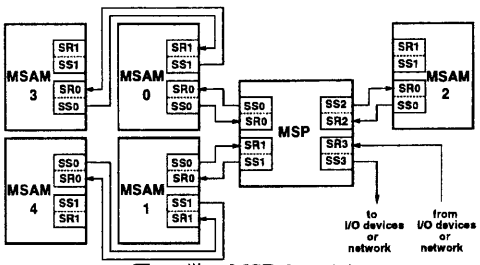

図4 単一MSPシステム

が不足する場合はMSAMをシリアルリンクで図4のように接続する。MSPの複数本のリンクはそれに連なる複数のMSAMから構成されるメモリバンクに対応し、同一バンクへのアクセスがシリアルリンク上で競合しなければ、複数のリンクと複数のメッセージを使って同時に複数の外部メモリアクセスが可能である。よって、MSPのプロセッサ／キャッシュは基本的にメモリアクセスの多重発行、命令のout-of-order実行、緩和されたメモリモデルをサポートする。2組（4本）以上のシリアルリンクを持つMSAMは中継機能を行うように設定可能になっており、同一バンク内のMSAMを図4のように接続可能である。中継ノードではMSPからの要求を無条件に後続のMSAMにフォワードし、各MSAMでは要求をデコードして予め設定された自分の担当アドレス以外の要求は無視される。MSAMからMSPへの返答はヴァーチャルカットスルー方式☆で転送される。MSPから遠い位置にあるMSAMがレイテンシ的には不利であるが、転送路が混雑していない場合はそのコスト増は小さい。公平にバンク接続をするために高速シリアルリンクのスイッチを介してMSPのシリアルリンクに接続する方法も考えられるが、コスト増になることは避けられない。MSAMが直接または間接的にシリアル結合して、システムが構成されるため、このシステム構成法を**Memory String アーキテクチャ (MSA)**と呼ぶ。

7. Memory Stringによる並列アーキテクチャ

MSPおよびMSAMで構成される並列計算環境では、共有メモリモデルに基づいた並列処理を行う[15]。そして、共有アドレスとしてプロセッサの論理アドレスを使用するため論理アドレスは64bit長程度有効であることが望ましい。

少数のMSPはMSP台数分のシリアルリンクを持つ外部MSAMで共有メモリ接続可能である。図5に4MSPと2MSAMからなるシステムを示す。さらに大規模な構成には相互結合網の構成要素としてシリアルリンクのスイッチが必要となる。1台以上のMSPとそれに付随する外づけMSAM（場合によっては存在しな

☆ シフトレジスタの任意アドレス境界からの入出力を認めていないので、厳密に言えばバーチャルカットスルーではなく、受信時に転送路確保に失敗すると次のアドレス境界までは必ずバッファリングされる。

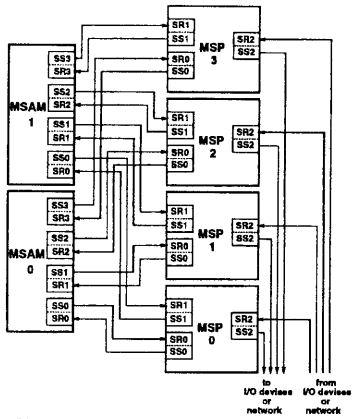

図5 4MSP+2MSAM 並列システム

い）がクラスタを構成し、クラスタ間はMSP間の接続としてスイッチを介したシリアルリンクで接続される。スイッチ間のリンク接続距離が長い場合はMSAMと異なる転送方式および転送媒体を使用することがある。

MSP間の高速かつ仮想化され保護されたユーザレベル通信をサポートするために、メモリベース通信 [16] [17] のためのサポートハードウェアをMSPは持つ。メモリベース通信のためのハードウェア化の度合は実装依存である☆。

MSP間の分散共有メモリの実現方法は実装依存であるが、付加的なハードウェアコストがなく効率の良い実現法の一つである非対称分散共有メモリ [17] もMSAにおける分散共有メモリ実装法の有力候補である。

8. おわりに

昨今話題になっているメモリウォールに関する過去の議論に対する問題点を指摘し、高速シリアルリンクをメモリインタフェースに採用することで大幅にメモリウォール問題が緩和されることを示した。複数の高速シリアルリンクを入出力に持つMulti-ported Serial-Access Memoryチップ（MSAM）とMSAMを内蔵したメモリ／ロジック混載LSIで作成されるMulti-ported Serial-access Processorチップ（MSP）を提案した。MSPならびにMSAMで構成される単一プロセッサシステムおよび並列計算機システムに関して論じた。これらのシステムのアーキテクチャ的な特徴は構成部品がメモリを介したシリアルリンクで接続されている点にあり、Memory Stringアーキテクチャと命名した。主記憶をすべてプロセッサチップに閉じ込めるアプローチとは異なり、プロセッサ当たりの主記憶容量を外づけMSAMの数で自由に調整可能である。MSAMは規格化された高速シリアルリンクインタフェースを通常のDRAMに付加したチップであるため、プロセッサを搭載したメモリチップよりも汎用構成部品としての地位を築き易いと考えている。

☆ ハードウェア寄りの実装としてMBPのhardwired logic回路が例に挙げられ、ソフトウェアよりの実装としてSSS-CORE [15] 上のメモリベース通信が挙げられる。

参考文献

1) Wm. A. Wulf and S. A. McKee: Hitting the Memory Wall: Implications of the Obvious. *ACM Computer Architecture News*, Vol.23, No.1, pp.20–24 (March 1995).
2) M. V. Wilkes: The Memory Wall and the CMOS End-Point. *ACM Computer Architecture News*, Vol.23, No.4, pp.4–6 (September 1995).
3) E. E. Johnson: Graffiti on "The Memory Wall". *ACM Computer Architecture News*, Vol.23, No.4, pp.7–8 (September 1995).
4) D. Burger, J. R. Goodman, and A. Kagi: Memory Bandwidth Limitations of Future Microprocessors. *Proc. 23rd Int. Symp. on Computer Architecture*, pp.78–89 (May 1996).
5) A. Saulsbury, F. Pong, and A. Nowatzyk: Missing the Memory Wall: The Case for Processor/Memory Integration. *Proc. 23rd Int. Symp. on Computer Architecture*, pp.90–101 (May 1996).
6) N. Jouppi: Improving Direct-Mapped Cache Performance by Addition of a Small Fully-Associative Cache and Prefetch Buffer. *Proc. 17th Int. Symp. on Computer Architecture*, pp.364–373 (May 1990).
7) D. Lenoski: The Design and Analysis of DASH: A Scalable Directory-Based Multiprocessor. PhD Dissertation, Stanford Univ. (December 1991).
8) Shimizu, et. al.: A Multimedia 32b RISC Microprocessor with 16Mb DRAM. *Int. Solid-State-Circuits Conf. 1996*, pp.216–217 (February 1996).
9) 村上, 吉井, 岩下: 21世紀に向けた新しい汎用機能部品PPRAMの提案. 情報処理学会研究報告 94-ARC-108, pp.49–56 (October 1994).
10) 松本 尚, 平木 敬: 超並列計算機上の共有メモリアーキテクチャ. 信技報, CPSY 92-26, pp.47–55 (August 1992).
11) 松本 尚: Memory-Based Processorを使用した汎用超並列計算機の基本アーキテクチャ. 並列処理シンポジウムJSPP '94 論文集, pp.409–418 (May 1994).
12) W. J. Dally and J. Poulton: Transmitter Equalization for 4Gb/s Signaling. *Proc. Hot Interconnects IV*, Palo Alto, CA (August 1996).
13) D. Litaize, et. al.: Multiprocessor with a Serial Multiport Memory and a Pseudo Crossbar of Serial Links Used as a Processor-Memory Switch. *ACM Computer Architecture News*, Vol.17, No.6, pp.8–21 (December 1989).
14) L. Yang and J. Torrellas: Optimizing Primary Data Caches for Parallel Scientific Applications: The Pool Buffer Approach. *Proc. 1996 Int. Conf. on Supercomputing*, pp.141–148 (May 1996).
15) 松本 尚 他: 汎用超並列オペレーティングシステム: SSS-CORE — ワークステーションクラスタにおける実現 —. 情報処理学会研究報告 96-OS-73, 情報処理学会, Vol.96, No.79, pp.115–120 (August 1996).
16) 松本 尚, 平木 敬: 汎用超並列オペレーティングシステムSSS-COREのメモリベース通信機能. 第53回情報処理学会全国大会講演論文集, 第1分冊, pp.37–38 (September 1996).
17) 松本, 駒嵐, 渦原, 平木: メモリベース通信による非対称分散共有メモリ. コンピュータシステムシンポジウム論文集, 情報処理学会, 発表予定 (November 1996).