解　説　高速プロセッシングデータバス技術

5. RISC超並列スーパーコンピュータのデータバス技術

Data Bus Technology of RISC-based Massively Parallel Supercomputer by Hideo SAWAMOTO, Osamu ISHIHARA (General Purpose Computer Division, Hitachi Ltd.) and Masamori KASHIYAMA (Ofice Systems Division, Hitachi, Ltd).

澤本英雄[1]　石原　修[1]　柏山正守[1]

1 (株)日立製作所

1. はじめに

1.1 なぜRISC並列コンピュータか？

1970年代のベクトル型計算機の出現以来，急速に発展してきた計算物理学や，大規模な気象／環境シミュレーションなど，コンピュータによる数値実験が急速に発展してきている．これらの分野では，問題の規模の拡大や，精度向上を図ろうとすると格子点が増加し，“メモリ量”と“演算量”が急激に増加する．たとえば格子点が100とすると，空間と時間の4次元が必要になり，$8B \times 100^4 = 800MB$ ものメモリが1変数だけで必要になってしまう．そのため“その時代の最高性能の計算機”を求めることになり，現在では，100GFLOPS(Giga FLoating Operations Per Second)以上の演算性能，100GB(Giga Bytes)以上のメモリ容量が要求される[1),2)]．

一方，これらの計算は空間的・時間的変化が局所的で，並列計算が可能であることが多い．そして，計算機ハードウェアからみると，従来のベクトル方式より分散メモリの並列方式の方が，同じコスト，同じ技術力であれば，より高いシステム演算性能(GFLOPS)とより大きなシステムメモリ量が実現できる．さらに，最近のCMOS半導体技術の進歩により高性能RISCプロセッサは，ベクトル型計算機に匹敵するクロック周波数になり，1チップでありながら高い演算性能と低消費電力の両立が可能になってきた．

これら諸条件から，最初に述べたような大規模科学技術計算の要求に答える“その時代の最高性能の計算機”としてはRISC並列コンピュータがふさわしい．

1.2 大規模科学技術計算におけるデータバス

前述のように大規模科学技術計算では膨大な“メモリ量”と“演算量”を要求するため，科学技術計算用計算機の指標として“GB”と演算のスループットである“GFLOPS”とが重要である．そしてこの“メモリ”と“演算”をつなぐのが“データバス”であり，この性能指標“メモリ・スループット(GB/s)”が実効性能には重要である．なぜなら膨大な“メモリ・データ”を速く大量に“演算”に供給できなければ，データ待ちのために演算が止まってしまい，ピーク演算性能は出せないからである．

本稿では，高い実効性能を出すため日立製作所が開発したSR2201のデータバス技術について解説する．まず最初にプロセッサ内の実効性能向上策について述べ，次にプロセッサ間のデータバスであるネットワークにふれた後，データバスの構成，それを実現するためのハードウェアを紹介し，最後に性能について述べる．

2. プロセッサ内のデータバス

2.1 マイクロプロセッサの長所と短所

近年の高性能マイクロプロセッサの性能向上には，目を見張るものがある．(性能)＝(周波数)×(方式性能)であるが，(周波数)は主に半導体デバイスの性能向上により2～3年で2倍になっており，(方式性能)も1チップあたりの搭載可能なトランジスタ数が増加したことから，スーパスカラによる同時実行命令数の増加，アウトオブオーダ実行によるペナルティの削減，キャッシュメモリ容量の増加によるキャッシュ・ミス・ペナルティの削減，などにより向上しており，(周波数)の向上と(方式性能)の向上の相乗効果

により，性能は飛躍的に向上している．

しかし，キャッシュにデータが入りきらないような大規模科学計算では深刻な問題がある．RISCの育ての親ともいうべきHennessyとPattersonは，いみじくも彼らの有名な教科書[3]の中で，次のように述べている．

“高速なパイプライン・プロセッサは，大規模な科学技術計算分野においてとくに有用となる．このようなプロセッサでは一般に，キャッシュを用いて実効的なメモリ・レイテンシの低減化に努めている．しかし，プログラム・サイズが大きく，かつ，実行時間が長い科学技術計算プログラムにおいては，非常に大きなデータへのアクセス，しかも参照の局所性の低いアクセスを行う場合が多い．参照の局所性が低いと記憶の階層化の効果が薄れ，結果としてキャッシュが役に立たないという事態を招く．この問題に対処するには，メモリ・アクセス・パターンがあらかじめ決まっていて，かつ，メモリ・アクセスが効果的にパイプライン化できるようなデータについてはキャッシングを行わないようにすればよい．コンパイラ(のアシスタンス)によって，将来このような装置も可能となるかもしれない．”（7.1章“なぜベクトル・マシンか”より）

換言すれば，たとえピークGFLOPSが高くても，あるいはSPEC値のようなほとんどのデータがキャッシュにある場合の性能(これをイン・キャッシュ性能と呼ぶことにする)が高くても，大規模科学技術計算においては，十分なメモリ・スループットがないかぎりピークGFLOPSもSPEC値もほとんど意味をもたない．

高いイン・キャッシュ性能というマイクロプロセッサの長所を保ちつつ，上記のような短所を解決し，大規模科学技術計算においてもイン・キャッシュ性能と同等の性能を達成するためには，レジスタ-主記憶間の(1)メモリ・レイテンシの隠蔽，と(2)レジスタ-キャッシュ間と同じメモリ・スループット，がなければならない．このため，

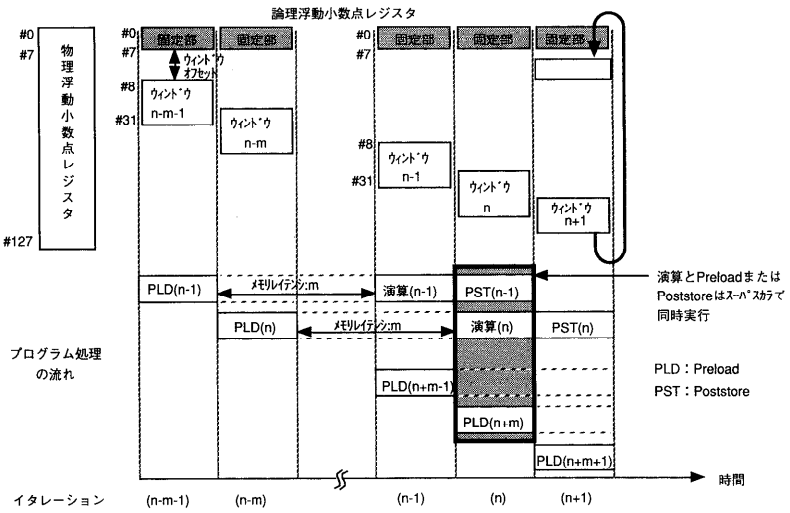


図-1 SR2201擬似ベクトル処理

SR2201では新たに擬似ベクトル処理(PVP: Pseudo Vector Processing)機構[4]を導入した．

2.2 擬似ベクトル処理

2.2.1 メモリ・レイテンシの隠蔽 ー Preloadと FPR本数の拡張

基本的なプログラムの処理は，(A)演算に用いるオペランド・データのメモリからレジスタへの“ロード”，(B)“演算”，(C)演算結果のメモリへの“ストア”，の3ステップに分けられ，ベクトルや行列計算ではこの処理がDO LOOPになって繰り返される．そして各ステップは前のステップが完了していないと実行できず，完了待ちのストールが発生する．

そこで，DO LOOPで繰り返されるという特徴と，同時に複数の命令が実行できるスーパスカラを利用し，1つのループ内で(B)“演算”と，メモリ・レイテンシ以上離れた将来のループで用いるオペランド・データの(A)“ロード”と，そして演算レイテンシ以上以前のループで演算した結果の(C)“ストア”を，1つのループ内で実行するようにコンパイラで命令スケジュール(ソフトウェア・パイプライニング)することで，メモリ・レイテンシと演算レイテンシを隠蔽する．これを可能にするために(A)“ロード”にPreload，(C)“ストア”にPoststoreという新命令を導入した．また，FPR(Floating Point Register)本数を，ベースとしたRISCアーキテクチャにおける本数の32本から128本に拡張した．

図-1に示すように，演算命令と，Preload命令

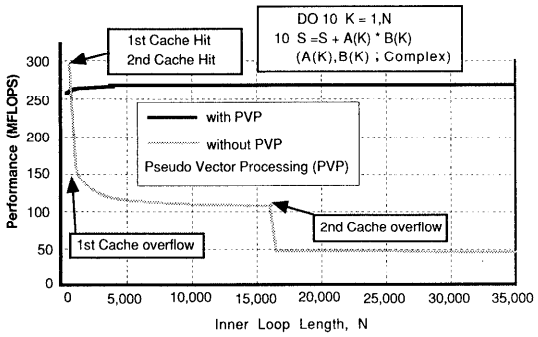


図-2 ベクトル内積計算の推定性能

またはPoststore命令はスーパスカラで同時に動作する．しかもPreload命令は，キャッシュ・ミスしても，ストールすることなく，キャッシュ・ヒットする通常のFLD(Floating Load)命令と同様に1サイクル・ピッチで発行でき(ノンブロッキング)，かつ主記憶から8Bのデータをキャッシュをバイパスして直接FPRに書き込む．メモリ・アクセスはパイプライン化されており，かつ主記憶-FPR間のメモリ・スループットが，1次キャッシュ-FPR間と同じ8B/cycleであるため，あたかもすべてのデータがキャッシュにあるイン・キャッシュの場合と同じ性能で動作する．また，Preload命令は，拡張した128本のFPRすべてをターゲットレジスタとして指定できる．そこで，たとえば1サイクルで1本のFPRを消費する(8B/cycle)プログラムにおいて，演算で使用しているFPR番号より，M離れたFPR番号をPreloadのターゲットにすれば，Mサイクル分のメモリ・レイテンシを隠蔽できることになる．

2.2.2 基本命令セットとの互換 ―スライドウィンドウの導入

ベースのRISCアーキテクチャでは，FPRは32本である(これを論理FPRと呼ぶことにする)．そこで，128本の物理FPRのうちの32本を，ウィンドウを通して論理FPRにマッピングする．そしてこのマッピング(レジスタ番号変換)をハードウェアで自動的に行うことにより，ベースのRISCアーキテクチャの命令セットが，そのまま動作するようにした．またウィンドウの始点のオフセットをセットするウィンドウ切り換え命令も新設した．この命令を使って，コンパイラは，ウィンドウを移動させることができる．そこで，これをスライドウィンドウと呼ぶ．

スライドウィンドウを使うと，同じ論理FPR番号でループしても，ループの最後でウィンドウをスライドさせることで，実際には異なる物理FPRを使うことができる．つまりコンパイラのアシストによるレジスタ・リネーミングである．新たに論理FPRにマッピングされた物理FPRには，以前に発行されたPreload命令によって，すでにメモリ・データが到着しているため，次のイタレーションでもメモリ・データ待ちのストールは発生せず，イン・キャッシュの場合と同じ性能で動作する．そして，データサイズが大きくなって，ループ長がいくら長くなっても，この動作を繰り返すため，性能低下はまったく起こさない．**図-2**にベクトル内積計算の机上推定性能を示す．なお実際にはループ長が短くキャッシュを使った方が性能がよい場合は，コンパイラがwithout PVPのコードに変えることができるため，各ループ長での最高性能を得られる．

図-1に示すように，論理FPRは“グローバル・レジスタ”と“ローカル・レジスタ”の2つの部分に分けた．

グローバル・レジスタはウィンドウを切り換えても共通で，本数は8，12，16と変えることができ，常に物理FPRの0からにマッピングされる．このため，定数や演算の中間結果の格納に使用できる．

ローカル・レジスタは128本からグローバル・レジスタ部を除いた物理FPR上を移動できる．たとえば，グローバル・レジスタ本数を8本とすると，物理FPRの#0 ～ #7がグローバル・レジスタになり，物理FPRの#8 ～ #127のうちの24本がローカル・レジスタとして，スライドする．そして，#127の次は#8にラップアラウンドするので，あたかも無限の本数があるかのようにみえる．

これらSR2201の動作をベクトル・プロセッサと比較すると，(A)Preload，(B)スカラ演算，(C)Poststore，といった都合N個のスカラ命令が，ベクトル長Nのベクトル・ロード，ベクトル演算，ベクトル・ストアに対応し，本数を拡張したスライドウィンドウつきFPRが，ベクトル・レジスタに対応すると考えられる．そしてPreloadは，ベクトル・ロードと同様に，(キャッシュ・ミスしたときは)キャッシュは用いない，

レジスタ-主記憶間直接転送である．そこで，我々はこの処理を“擬似ベクトル処理”と呼ぶことにした．

2.3 Preload と Prefetch の比較

いくつかの高性能 RISC プロセッサでは，Preload ではなく Prefetch をサポートしており，SR2201 でも，Preload とは別に Prefetch をサポートしている．そして Prefetch もまた，キャッシュ・ミス時のメモリ・レイテンシを隠蔽しキャッシュ・ミス・ペナルティを削減する目的であるため，Preload と Prefetch は混同されやすい．しかし場合によっては下記(1)から(4)の点では Preload が優れている（図-3 参照）．

(1) Prefetch(図-3 ： Case1)では，FLD 命令のほかに Prefetch 命令が余分に必要となり命令数が増加する．一方 Preload では，通常の FLD 命令を置換するだけであるため命令数は増加しない．

(2) Prefetch(図-3 ： Case1)では，2 次キャッシュまたは主記憶からのフェッチ・データの 1 次キャッシュ書き込み中，キャッシュはロード命令による読み出しができない．なぜなら，通常はキャッシュは 1 ポートの RAM で，リードとライトは同時にはできないからである．結果として，主記憶から FPR までの実効メモリ・スループットが低下してしまう．

ここで注意したいことは，RISC においては演算のオペランド・データは，レジスタになければならないため，実効性能を規定するのは主記憶-キャッシュ間(プロセッサ・バス)ではなく，主記憶-レジスタ間のメモリ・スループットであり，さきに述べたようにメモリ・スループットを主記憶-レジスタ間でみると Prefetch では主記憶-キャッシュ間より悪くなることがあるのである．

一方 Preload では，主記憶のデータを直接 FPR に格納するため，Prefetch での主記憶-レジスタ間のメモリ・スループットの低下は発生しない．

(3) Prefetch(図-3 ： Case2)では 1 回のメモリ・リクエストでキャッシュ・ライン単位(SR2201 では，主記憶からは 128B)のデータをフェッチする．このため，リスト・ベクトルのような離散アドレスのデータでは Prefetch してくるキャッシュ・ラインの一部だけが有効データである．たとえば要素間のアドレスが 128B 以上離れていると，128B のキャッシュ・ラインのうち 8B だけが有効なデータであり，実効メモリ・スループットは 8/128 ＝ 1/16 に低下してしまう．

一方，Preload では 8B 単位に主記憶に対してリクエストを発行するため，たとえランダムなアドレスであっても上記のような実効メモリ・スループットの低下はない．

(4) Prefetch(図-3 ： Case2)は，キャッシュへのデータ・ロードであるため，キャッシュでのスラッシングが発生することがある．また，リスト・ベクトルのリスト・アドレスなどのように，キャッシュに置いておいて，何度も再利用したいデータが，あるとき突然 Prefetch データによって上書きされる恐れもある(キャッシュ・ポルーション)．一方，Preload ではキャッシュを使わないため，上記のような問題は発生しない．

以上 Preload の利点を述べた．しかし Preload は，浮動小数点データに対してしか適応できない．一方 Prefetch は，目的のデータが固定小数点であれ浮動小数点であれ有効である．したがって，固定小数点データは Prefetch でキャッシュに，浮動小数点データは Preload で

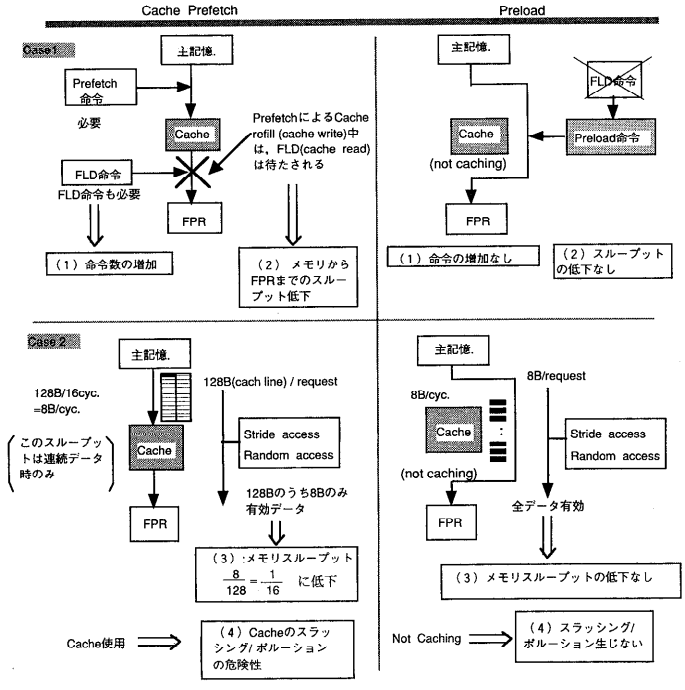


図-3 Preload と Prefetch の比較

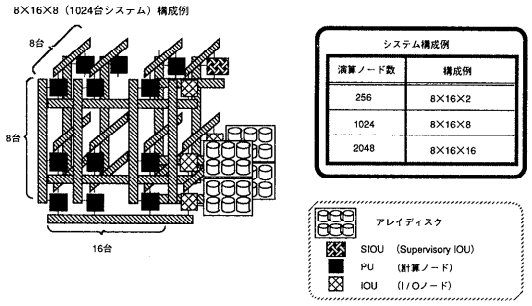


図-4 3次元クロスバネットワーク

FPR に先取りできる．たとえば前述のリスト・ベクトルのリスト・アドレスは Prefetch してキャッシュに置いておいて，リスト・ベクトル自身は Preload で FPR に先取りすることができる[5]．

3. プロセッサ間のデータバスーネットワーク

ほかの PE（Processor Element ：要素プロセッサ）の主記憶（ローカルメモリ）とのデータバスであるネットワークも，並列コンピュータとしての性能にとって重要である．ネットワークは，PE を 3 次元に配置し，各次元方向の一列にクロスバスイッチを配置した 3 次元クロスバ構成をとる．各次元のクロスバスイッチは入力ポートと出力ポートの組合せが異なれば，すべての入力ポートから同時に経路の衝突なしに転送が可能であり，最大 3 回クロスバスイッチを乗り換えることにより任意の PU および IOU と転送が可能である．

図-4 に 1,024PU（Processor Unit ：演算プロセッサ）＋ 64IOU（IO Unit ： IO プロセッサ）を接続する 3 次元クロスバネットワークを示す．なお，ハードウェアとしては，PU も IOU も同じ PE である．

4. SR2201 のデータバス構造

図-5 に CPU のブロック図を示す．FPR-1 次キャッシュ（D-cache）間（FLD 命令，Preload 命令によるデータ転送），1 次キャッシュ-2 次キャッシュ間（BT:Block Transfer），1 次／2 次キャッシュ-主記憶間（LT:Line Transfer），そして FPR-主記憶間（1 次キャッシュ・ミスの Preload 命令）すべてメモリ・スループットは 1.2GB/s（8B × 150MHz または 16B × 75MHz）になるように設計してある．

図-6 に 1PE ノードと，各 PE 間を結合するクロスバの LSI 構成とインターコネクトを示す．

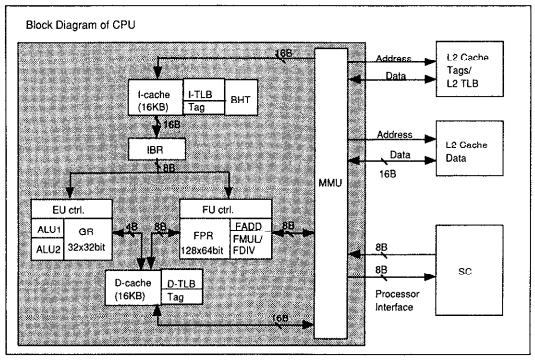


図-5 CPU ブロック図

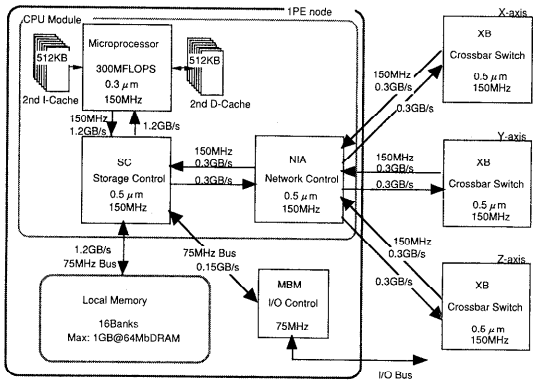


図-6 PE の構成

PE は RISC プロセッサの CPU，SC（Storage Control unit），ローカルメモリ，ほかの PU とのデータ転送をするネットワーク制御の NIA（Network Interface Adapter）からなる．そして最大 2,048PU ＋ 128IOU を結合する 3 次元クロスバスイッチ LSI が XB である．

CPU-SC 間はアドレス／データ兼用の 8B 幅，150MHz，片方向のパス，2 組（上りと下り）で接続されている．このため，Preload 連続発行の場合，CPU からみて下りのパスを使って，SC に毎サイクル Preload のアドレス，あるいはストア・アドレスとストア・データを送りながら，同時に上りのパスを使って以前に発行した Preload のフェッチ・データの 8B を毎サイクル受け取ることができる．すなわち，実効メモリ・スループットとして 8B × 150MHz ＝ 1.2GB/s を実現している．

SC-ローカルメモリ間は 75MHz であるが，バス幅を 8B × 2 にすることで 1.2GB/s のメモリ・スループットを達成している．したがって CPU

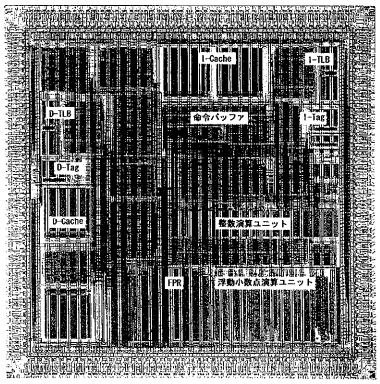


図-7 CPUチップ写真

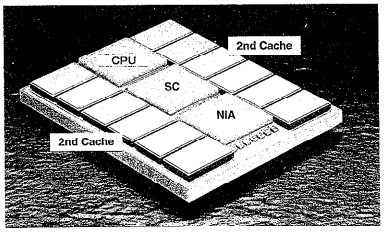


図-8 プロセッサ・モジュール

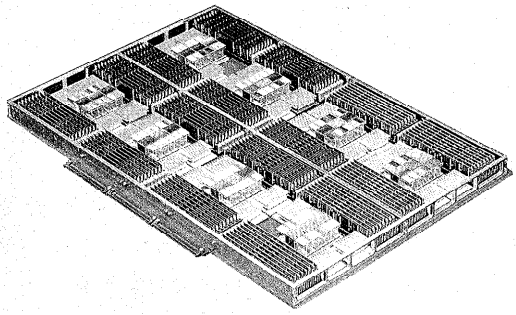


図-9 8PEパッケージ

のレジスタからローカルメモリまで，メモリ・スループットは，すべて1.2GB/sを実現している．

PE間のデータパスであるネットワークは，SC-NIA間，NIA-XB間すべて片方向2B×150MHz＝300MB/sのスループットを実現している．3次元クロスバネットワークは，ソースシンクロナス・ディープ・ウェーブパイプライン技術で150MHzサイクルのデータ転送を行っている．

5. SR2201のハードウェア技術

SR2201のデータバスの高メモリ・スループットを支えるハードウェア技術について紹介する．

表-1 CPU LSI諸元

プロセス技術	0.3 μm CMOS
周波数	150MHz
チップサイズ	15.7mm × 15.7mm
トランジスタ数	4.5M
LSIパッケージ	1672pin CCB，(信号520)
電源電圧	2.5V
消費電力	10W@150MHz
Peak MFLOPS/W	30MFLOPS/W

(1)プロセッサ・エレメント(PE)

CPUのチップ写真を**図-7**に，LSI諸元を**表-1**に示す[6)~8)]．CPUチップは，0.3 μ mCMOSフルカスタムLSIである．

図-6で，CPU Moduleとして示したチップセットを搭載したプロセッサ・モジュールの写真を**図-8**に示す．プロセッサ・モジュールに搭載しているLSIは，中央上段から順にCPU，SC，NIAとなっている．LSIの周辺に搭載している12個のチップは，2nd cache用シンクロナスSRAMで1MBある．約6cm×7cmのモジュール上に，これらのLSIを高密度，コンパクトに実装にすることで，LSI間の150MHz，1サイクルピッチのデータ転送を実現している．

(2)8PEパッケージ

8個のPEを収納する8PEパッケージ(約40cm×60cm)を**図-9**に示す．各々のPEは，最大1GBのローカルメモリを搭載可能である．各々のPEのサイズは，約15cm×20cmとコンパクトであり，このことにより低いローカルメモリ・レイテンシを達成している．

6. 性　　能

6.1 性能測定結果

(1)ベクトル内積計算

図-10にシングルプロセッサにおけるベクトル内積計算の実測結果を示す．擬似ベクトル処理の有無による性能差がよくわかる．シングルプロセッサのピーク性能300MFLOPSに達していないのは，擬似ベクトル処理ではスカラ命令を繰り返すために，演算ループの最後に分岐命令とスライドウィンドウ切り換え命令が必要になるためである．このオーバヘッドは，ループ・アンローリングにより低減できる．

(2)LINPACK

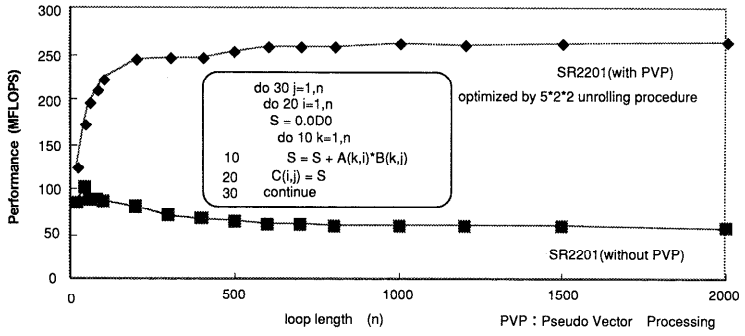


図-10 ベクトル内積計算の実測結果

Result of LINPACK Benchmarks(Comparison of Achievement)

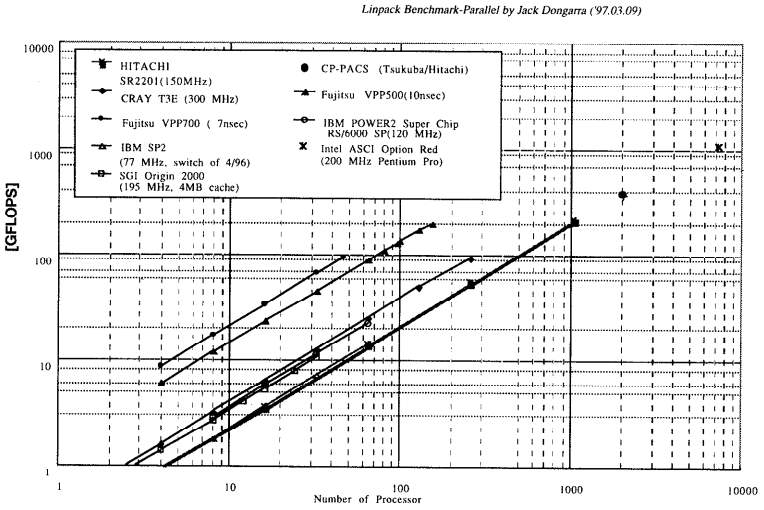


図-11 LINPACK ベンチマーク結果[9]

ベンチマーク・テスト結果として，LINPACK[9]の結果を，**図-11** に示す．1,024 プロセッサで，約 220GFLOPS を達成し，実働している商用機として世界最高性能である．また擬似ベクトル処理，3 次元クロスバスイッチ，そして高密度ハードウェア実装技術により，効率は 72 ～ 75 % と 1,024 プロセッサまでほとんど低下せずフラットで，8PU から 1,024PU までスケーラブルに性能が向上している．

7. おわりに—今後の展開

おわりにあたって，再度 Hennessy と Patterson の文章[3]を引用させていただく．

“将来，現在のベクトル型スーパコンピュータのクロック周波数に匹敵するクロック周波数の高性能スカラ・マシンが登場する可能性がある．ただ，複数のメモリ・パイプラインが使用でき，かつ，ピーク性能に近い性能が出せる非常に長いベクトルを多用するような問題に対しては，ベクトル・マシンはまだ性能の優位性を保てるはずである．

1990 年代には，廉価（なぜなら，ピーク性能およびハードウェア量がずっと小さいから）なスカラ・マシンがより大量の命令レベル並列性を活用するようになり，多くのアプリケーションにおいてベクトル・マシンと大差ない性能を発揮し出すようになろう．”（7.9 章“おわりに”より）

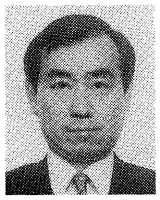
1 チップ CMOS RISC プロセッサは最先端半導体技術の進歩を素早く享受し，その半導体技術の最高性能を引き出しやすい．そして，周波数と演算器の数だけで決まるピーク性能は，この半導体技術の進歩の恩恵を直接受けることができるため，今後も RISC 並列コンピュータが，その時代の最高性能のコンピュータとして発展していくだろう．しかし，本稿で述べたような実効性能を決めるデータバスの高速化は，半導体だけではなく，総合的なハードウェア技術が必要であり，ピーク性能の飛躍的向上に追従することがますます困難になっていくと思われる．

今後このピーク性能と実効性能の乖離を縮めるべく，ハードウェア開発が進められるだろうが，一方で，コンパイラだけではなく，ソースプログラムレベルでのメモリ・アクセス頻度の低減やメッセージパッシングによる並列化などでメモリデータバスの負担を軽減することが，実効性能向上にますます重要になってくると思われる．

参考文献

1) 岩崎洋一，宇川 彰，梅村雅之：計算物理学とCP-PACS計画，情報処理，Vol.37，No.1，pp.11-17(Jan. 1996)．
2) 川添良幸，水関博志，八重樫育生：並列計算機は実際に超大規模計算に役立つのか？，bit，Vol.29，No.4，pp.85-93(1997)．
3) Hennessy, J. L. and Patterson, D. A.(富田真治他訳)：コンピュータ・アーキテクチャ，日経BP社(1992)．
4) 位守弘充，中村 宏，朴 泰祐，中澤喜三郎：スライドウィンドウ方式による擬似ベクトルプロセッサ，情報処理学会論文誌，Vol.34，No.12，pp.2612-2623(Dec. 1993)．
5) 広野 哲，中村 宏，朴 泰祐，中澤喜三郎：擬似ベクトルプロセッサにおける高速リストベクトル処理，情報処理学会論文誌，Vol.37，No.10(Oct. 1996)．
6) Saito, K. et al.: A 150MHz Superscalar RISC Processor with Pseudo Vector Processing Feature, Hot Chips VII, 6.4(1995)．
7) 斎藤拡二他：擬似ベクトル機構を有する並列コンピュータ向けRISCプロセッサ，信学技報 TECHNICAL REPORT OF IEICE, DSP95-104, ICD95-153(Oct. 1995)．
8) Shimamura, K. et al.: A Superscalar RISC Processor with Pseudo Vector Processing Feature, Proc. International Conference on Computer Design(ICCD '95), pp102-109(1995)．
9) Dongarra, J.: Performance of Various Computers Using Standard Linear Equations Software (March 1997)．(最新版は http://www.netlib.org/benchmark/performance.ps)．

(平成9年4月16日受付)



澤本 英雄

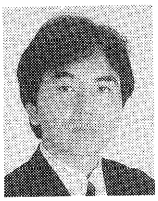
1953年生．1976年京都大学工学部電気系学科卒業．1978年同大学院工学研究科修士課程電子工学専攻修了．1985年Yale大学Master of Science(Electrical Engineering専攻)，1978年(株)日立製作所入社．現在，汎用コンピュータ事業部主任技師．汎用大型計算機の開発を経て，SR2201のRISCプロセッサ開発に従事．RISCプロセッサおよびRISC並列コンピュータに興味をもつ．IEEE会員．

e-mail:hsawamo@kanagawa.hitachi.co.jp

石原 修（正会員）

1965年生．1989年日本大学理工学部物理学科卒業．1991年同大学院理工学研究科修士課程物理学専攻修了．同年(株)日立製作所入社．現在，汎用コンピュータ事業部技師．スーパーコンピュータの開発に従事．科学技術計算，並列プログラミングに興味をもつ．IEEE会員．

e-mail:oishiha@kanagawa.hitachi.co.jp



柏山 正守（正会員）

1959年生．1983年日本大学理工学部物理学科卒業．1983年(株)日立製作所入社．現在，オフィスシステム事業部主任技師．スーパーコンピュータおよびRISCプロセッサの開発に従事．パーソナルウルトラコンピュータに興味をもつ．IEEE会員．

e-mail:masamori@ebina.hitachi.co.jp