計算機アーキテクチャ 132−15
システムソフトウェアとオペレーティング・システム 80−15
ハイパフォーマンスコンピューティング 75−15
（1999. 3. 5）

オンチップメモリを用いたHPCプロセッサの検討

近藤正章[†] 坂井修一[††]
朴泰祐[†] 中村宏[†††]

本稿では，CPUとメモリをシングルチップ上に融合させたプロセッサ・メモリ混載型LSIについて，特にHPC分野をターゲットにしたアーキテクチャの検討を行なう．プロセッサ・メモリ混載型LSIでは，オンチップメモリに対するアクセスが低レーテンシかつ高バンド幅であるため，性能向上が期待されるが，HPCではワーキングセットが大きく，オンチップメモリにそのすべてが収まりきらないことが多い．ここでは，最初に，HPC用VLSIアーキテクチャとしてオンチップメモリ，オフチップメモリの両者を持つアーキテクチャを考え，その命令セットおよびハードウェア構成の概略を提案する．次に，オンチップ・オフチップ両メモリのスループット，浮動小数点演算器数などをパラメータとして，いくつかのプログラムについての予備実験を行ない，アーキテクチャの諸元を設定する．さらに，Linpackベンチマークプログラムにおける性能予測では，オンチップメモリを用いたブロッキングを行なうことにより，プロセッサ・メモリ混載型LSIで高性能が得られることを確認した．

A study of HPC processor using On-Chip Memory

MASAAKI KONDO,[†] SHUICHI SAKAI,[††] TAISUKE BOKU[†]
and HIROSHI NAKAMURA[†††]

In this paper, we describe our study of processor-memory integrated LSI architecture aiming at performance improvement of HPC applications. The memory-integrated processor has low latency and high bandwidth in respect of access to the on-chip memory. In the HPC applications, however, their working sets are too large to fit into the on-chip memory. Therefore we discuss VLSI architecture with both on-chip and off-chip memories, and we propose an outline of extended instructions and hardware. In addition, to decide architectural factors, the performance of some programs are evaluated with on-chip/off-chip memory throughput and number of floating point pipeline units being parameterized. And Linpack benchmark examination shows that memory-integrated processor achieves high performance taking on-chip memory blocking algorithm.

1. はじめに

近年，トランジスタ集積度の高成長により，CPUとメモリをシングルチップ上に融合させたプロセッサ・メモリ混載型LSIが現実のものとなってきている．プロセッサ・メモリ混載型LSIでは，ロジック部とオンチップメモリ間の高いバンド幅や低い通信レーテンシを活用できるため，システムとしての高性能化が期待されている．そのため九州大学を中心としたPPRAM[1]やカリフォルニア大学バークレー校のIRAM[2]など，DRAMとロジックを一体化させたアーキテクチャの研究がなされている．我々はその高性能化を，大規模科学技術計算に代表されるハイパフォーマンスコンピューティング(以下HPCと呼ぶ)分野のアプリケーションに適用することを検討している．

一方，高バンド幅や低通信レーテンシといった特徴を活用するためには，データがオンチップメモリ上にある必要があるが，HPCのアプリケーションにおいては，基本的にワーキングセットが大きく，またそのサイズはCPU能力にほぼ比例する．例えば現在筑波大学で稼働中のCP-PACS[3]の例から，2004年頃に必要なメモリ容量を概算してみる．CP-PACSのPUあたりのメモリ量は256MB程度であり，クロック周波数は150MHzである．2004年に内部クロックが2.1GHz，すなわち現在のCP-PACSの14倍になったとすると，約3.6GByteのメモリが必要になると考えられる．一方，STARC[4]の予測によると，2004年のプロセッサ

† 筑波大学 電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba
†† 東京大学 工学研究科 電気工学専攻
Department of Electrical Engineering, The University of Tokyo
††† 東京大学 先端科学技術研究センター
Research Center for Advanced Science and Technology, University of Tokyo

で混載可能なDRAM容量は1Gbit程度であり，単純にプロセッサとメモリを混載しただけでは，オンチップメモリ容量の不足のため高性能化が得られない．

我々は，この点から従来のプロセッサアーキテクチャを考え直し，大容量の主記憶を外付けDRAMで実現する一方，プロセッサ内部には中規模容量のメモリを持つ新たなプロセッサアーキテクチャを提案する．外部のDRAMはバンク分けをして高スループットを実現し，内部メモリは，(1) レーテンシ隠蔽，(2) 中間結果の格納と高速読み出し，の2つの目的で使用する．

本稿では，HPC向けのプロセッサ・メモリ混載型LSIについて，オンチップメモリを考慮した命令，およびハードウェア構成を検討し，予備的性能評価を行なうことを目的とする．次章ではプロセッサ・メモリ混載型LSIについて概要やその背景，また我々の提案するアーキテクチャの基本的アイデアを示す．3章では具体的に拡張命令，拡張ハードウェアについて検討し，4章においてLinpackベンチマークプログラムによる性能予測を示す．5章でまとめと今後の課題について述べる．

2. メモリ混載型アーキテクチャ

2.1 背　　景

現在のコンピュータ・システムでは，半導体技術の進歩を背景に，クロック速度の向上，スーパスカラやVLIWによるプロセッサ内並列化，投機的実行といった技術により性能向上が著しく，主記憶との性能格差の問題，すなわちフォン・ノイマン・ボトルネックの問題が深刻化してきている．また，この性能格差は今後さらに拡大する傾向にある．現在ではメモリ階層を多段化し，キャッシュメモリを設けることによりこの現状に対処している．しかしHPC分野のアプリケーションでは，ワーキングセットが大きく，キャッシュミスが頻発するため，性能低下が著しい．

そこで，キャッシュミスが起きたときのペナルティを低減する方法として，(a) キャッシュプリフェッチ，またキャッシュミスそのものを削減する方法として，(b) キャッシュブロッキング，などが考えられてきた．しかしそれらの技法にはいくつかの問題点がある．

まず(a) キャッシュプリフェッチにおいては，キャッシュ階層を多段化することが通常となる現在では，キャッシュラインサイズの違いから，どのキャッシュに対してのプリフェッチを行なうかでその粒度が変わる．個々のキャッシュにとって最適な粒度でプリフェッチを行なうことができないため，プリフェッチにより無駄なデータ転送が生じることがある．

また，(b) キャッシュブロッキングにおいては，データのキャッシュへのマッピングが制御できず，うまくキャッシュに乗らない場合がある．例えば配列サイズが2の巾乗のときself-interferenceによるキャッシュミスが生じる．これに対しては小さなブロックサイズを選ぶ[5]，配列サイズを変更してself-interferenceを減らす[6]，などが提案されているが，いずれもデータサイズ，キャッシュ構成に応じてプログラムを変更する必要があり，またプログラムも複雑になる．さらに複数の配列をキャッシュに載せる必要がある場合の相互干渉，すなわちcross-interferenceを排除することは難しい．

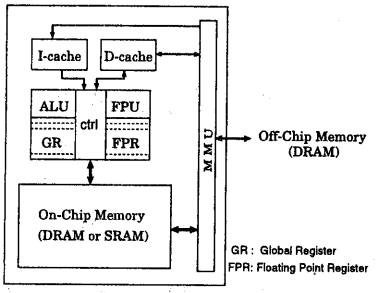


図1　ブロック図

2.2 明示的に制御できるオンチップメモリ

前節で述べた問題点は，本来ハードウェアで暗黙的に制御されるキャッシュをユーザ側で制御しようとするために生じる．そのため我々は，キャッシュに代わりユーザ(コンパイラ)がプログラムで明示的に制御できるメモリをプロセッサ内部に搭載することで，その問題点に対処することを検討している．HPCのアプリケーションではメモリアクセスが定型的な場合が多く，ユーザレベルでアクセスを制御できると考える．

図1に我々の提案するアーキテクチャのブロック図を示す．プロセッサチップ内にはロジック部と1次キャッシュの他にオンチップメモリを搭載する．ユーザレベルでは読み切れないアクセスがあることを考慮し，1次キャッシュも搭載している．したがって，アクセスが定型的でないものは，

register ↔ cache ↔ Off-Chip RAM

のように従来のキャッシュを利用したデータアクセスを行ない，アクセスが定型的なものは，

register ↔ On-Chip RAM ↔ Off-Chip RAM

のように混載したメモリを利用したアクセスを行なう．

ここで，オンチップメモリとしてはDRAMだけでなくSRAMを用いることも想定している．これは，前述のようにHPCのアプリケーションにとってオンチップメモリの容量は絶対的に不足するため，大容量であるがレーテンシの大きなDRAMにするか，DRAMに比べ小容量ではあるがレーテンシの小さなSRAMにするかで，どちらの性能が良いか現時点では判断し難いためである．オンチップSRAMは一見キャッシュのように見えるが，ハードウェアによる自動リプレース等がないため，定型的なアクセスに対するデータのアロケーションとスケジューリングはかえってやり易

表1　ハードウェアの仮定

仮定	内部 clock	bus clock	data bus pin 数	Off-Chip DRAM レーテンシ
HW1	2 [GHz]	500 [MHz]	512 [pin]	20 [ns]
HW2	2 [GHz]	1 [GHz]	512 [pin]	20 [ns]
HW3	2 [GHz]	1.6 [GHz]	1024 [pin]	20 [ns]

くなると考えられる．

上記構成のもと，主に次の方法で性能向上を狙う．

- オフチップ・オンチップメモリ間では，まとまった量のデータの prefetch，poststore を行なうことによりオフチップメモリのレーテンシを隠蔽する
- オンチップメモリ・レジスタ間でも preload / poststore を行なう
- レジスタ・オンチップメモリ間での preload / poststore と，オンチップ・オフチップメモリ間での prefetch / poststore をオーバーラップさせる
- オンチップメモリを用いたブロッキング手法などを用い，再利用性を活かすことで，オフチップメモリアクセスの低減を図る一方，内部メモリの低レーテンシ，高バンド幅を活かしたデータアクセスにより性能向上を図る

さらに，上記の方法でデータ供給系がボトルネックとならなければ，スーパスカラや VLIW などの手法で，現在より浮動小数点演算器数を増やし，命令レベル並列度を上げることで，絶対性能をも向上させることができると考える．

3. 命令およびハードウェアの仕様

3.1 拡張命令

以下に本稿で提案するアーキテクチャにおける，追加命令および拡張命令について詳述する．

extend-load, extend-store　レジスタとオンチップメモリ間のデータ転送を行なう拡張命令: オンチップメモリの高バンド幅を利用し，複数レジスタに対する複数ワードの同時 preload/poststore が行なえる．次節以降では，n extend-load/store とし，n は連続何ワードを転送できるか表し，n をパラメータとして評価する．

page-load, page-store　オンチップメモリとオフチップメモリ間のデータ転送行なう追加命令: 転送はページ単位で行ない前後のページに対する計算とをオーバーラップさせることによりレーテンシを隠蔽する．またページサイズは可変である．

これら命令のフォーマットなどはまだ厳密には定義していないが，両命令ともソースアドレスとディスティネーションアドレス/レジスタを指定する必要がある．

3.2 予備的性能評価

本節では，前述の命令をもとに最適なハードウェア仕様を検討するため，2003 年頃に可能となるであろうハードウェアを仮定し，いくつかのカーネルループに対する予備的性能評価を行なう．

以下に仮定したレーテンシおよび条件，また表 1には仮定したハードウェアを示す．

- 命令のレーテンシ
  - extend-load/store 命令: 5 [clock]
  - multiply 命令: 2 [clock]
  - multiply and add 命令: 4 [clock]
- 仮定する条件
  - すべての浮動小数点演算器 (パイプライン) で multiply and add 命令が実行可能
  - page-load/store の page サイズ: 4KByte 固定
  - extend-load/store は倍精度単位で行なえる
  - オンチップメモリサイズ: 2MByte

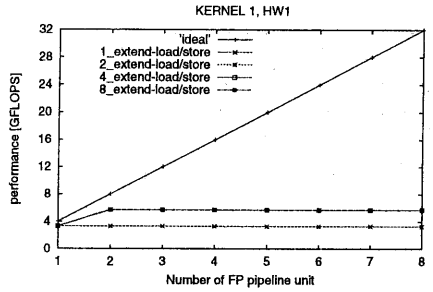
表 1の HW3 については米国半導体工業会 (SIA) の 2003 年における予測 [7] をもとに仮定した．

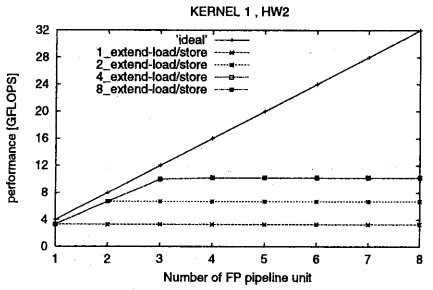
これら条件のもと，HW1-3 とオフチップメモリスループット/アクセスタイムを変化させ，さらに extend-load/store ワード数 n をパラメータとしてオンチップメモリのスループットを変化させた．評価に用いるプログラムは livemore kernel より "KERNEL 1"，"KERNEL 7"，および "KERNEL 21" を多少変更した "行列積" のプログラムを用いた．それらのプログラムを以下に示す．なお，評価は机上計算によるものである．

```
----- livemore kernel 1 -----
  DO 1 k = 1,1000
1   X(k)= Q + Y(k)*(R*ZX(k+10) + T*ZX(k+11))
----- livemore kernel 7 -----
  DO 7 k = 1,1000
    X(k)=  U(k  ) + R*( Z(k  ) + R*Y(k  )) +
.      T*( U(k+3) + R*( U(k+2) + R*U(k+1)) +
.      T*( U(k+6) + R*( U(k+5) + R*U(k+4))))
 7 CONTINUE
----- 行列積  -----
  DO 21 k= 1,25
    DO 21 j= 1,25
      DO 21 i= 1,1000
        PX(i,j)= PX(i,j) +VY(i,k) * CX(k,j)
21 CONTINUE
```

livemore kernel 1

図 2に KERNEL 1 の評価結果を示す．KERNEL 1 は 2load，1store に対して 5 演算を行なうカーネルループであり，レジスタの再利用性がほとんどない．したがって，extend-load/store ワード数の増加に伴い性能が向上する．しかし，あるところで exted-load/store ワード数の増加に対し性能が頭打ちになる．これは，オンチップ・オフチップメモリ間のスループットがボトルネックとなるためである．このプログラムではオン





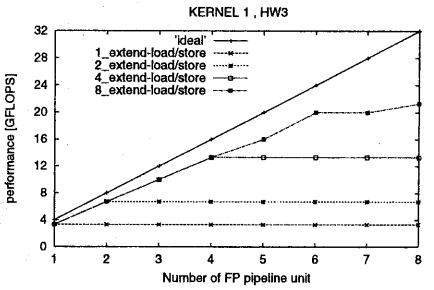
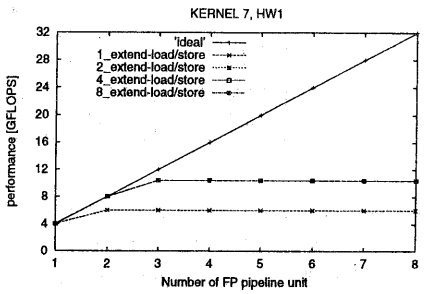
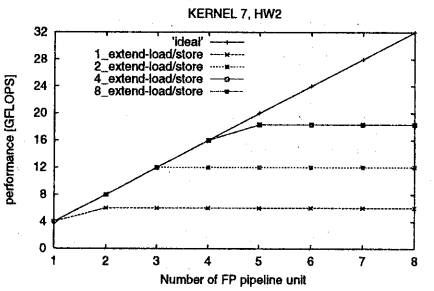


図 2　livermore kernel "KERNEL 1" の性能評価結果





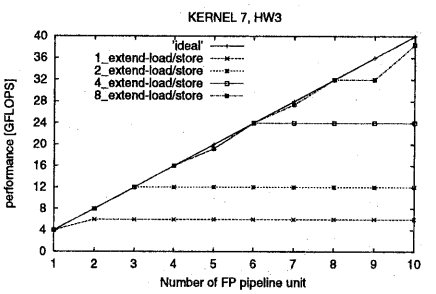


図 3　livermore kernel "KERNEL 7" の性能評価結果

チップメモリの再利用性がないため，仮定ハードウェアの HW1 から HW3 とスループットの高くなるにつれ，頭打ちになる限界のラインの性能が向上している．

livemore kernel 7

図 3に KERNEL 7 の評価結果を示す．KERNEL 7 は 3load，1store に対して 12 演算を行なうカーネルループであり，レジスタの再利用性が非常に高い．したがって extend-load/store より演算がボトルネックになり易く，浮動小数点演算パイプラインの増加に伴い，性能が向上していくのがわかる．

性能が頭打ちになるのは KERNEL 1 と同じくオンチップメモリに再利用性がないためであるが，HW3 のようにオフチップ・オンチップメモリ間のスループットが非常に高い場合，浮動小数点演算パイプライン数が 10 以上になっても，データ供給が間に合うため，さらに性能が向上している．

行　列　積

図 4に行列積の評価結果を示す．このプログラムは 2load，1store に対し 1 演算を行なう．このプログラムではオンチップメモリの再利用性があり，最終的に配列すべてをオンチップメモリに収めることが可能なため，オンチップ・オフチップメモリ間のスループットによる性能差がほとんどなくなる．したがって，HW1 から HW3 ともにすべて同じ結果となる．

また，このプログラムはレジスタの再利用性が全くないため，extend-load/store がボトルネックとなる．そのため同時 extend-load/store 数の増加に従い性能向上が顕著に現われる．

3.3　ハードウェア仕様

以上，いくつかのカーネルプログラムに対して性能評価を行なったが，それぞれの評価結果より，コストも含め最適なハードウェア仕様を検討する．

まず，オフチップメモリとオンチップメモリ間のスループットであるが，これは表 1の I/O clock とデータバスのピン数により決定される．当然 HW1 から HW3 へとスループットの向上に伴いシステムの性能

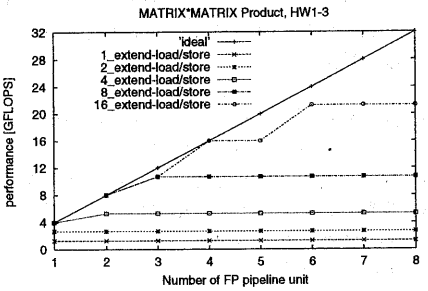


図4 "行列積" の性能評価結果

も向上するが，コストも非常に増加する．実現容易性も含め考えて，仮定ハードウェアの HW2 を取り挙げて評価を進める．

次に浮動小数点演算パイプラインの数，および同時 extend-load/store ワード数 n について検討する．各々の評価結果において HW2 のグラフを見ると，8 extend-load/store で浮動小数点演算パイプライン数が 3 から 5 のあたりで性能が頭打ちになっている．2003 年頃における半導体技術を仮定した場合，レジスタ・オンチップメモリ間に 8 × double precision word，すなわち 512bit のデータパスを引くことは，難しくないと考えられる．したがって，本研究では 8extend-load/store，4 × 浮動小数点パイプライン構成が最適ではないかと考える．以後この構成をもとに検討を行なう．

上記のハードウェア構成を以下にまとめる．また図5にデータ転送を含めたシステム全体の構成図を示す．

- internal clock: 2 [GHz]
- I/O clock: 1 [GHz]
- number of pins(data bus): 512 [pin]
- 浮動小数点演算器 (パイプライン) 数: 4 [本]
- 同時 extend-load/store 数: 倍精度語 × 8
- 最大性能: 16 GFLOPS/PU

4. linpack による性能予測

4.1 性能評価

本章では前節で検討したハードウェア仕様をもとに，実際に Linpack ベンチマークプログラムを用いた性能評価を行なう．

検討したのは一番処理の重いガウスの消去法を行う部分についてであり，pivoting などの時間は含めない．また行列サイズは 1000×1000 を対象とし，倍精度長で演算を行なう．すべての行列がオンチップメモリに乗るためには約 7.6MByte 必要となる．

以下に検討する上での条件を示す．

- 命令のレーテンシは考えない．
- page-load/store の page サイズ: 4KByte 固定
- オンチップメモリサイズ: 2MByte

オンチップメモリを 2MByte と仮定するため，行列すべてを収めることができない．そのため，高性能を得るためにはブロッキング手法により，オフチップメモリアクセスの低減を図る必要がある．

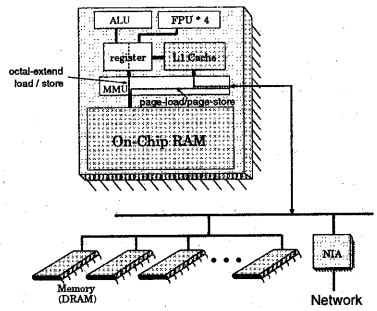


図5 システム全体の構成図

評価には 3 種類のアルゴリズムを用いた．

(1) Linpack ベンチマーク default のアルゴリズム (外積型ガウスの消去法)

(2) オンチップメモリブロッキングを行なったアルゴリズム

(3) (2) に加えレジスタブロッキングを行なったアルゴリズム

(1) および (2) では，すべての行列がオンチップメモリに乗った場合の性能は 10.6[GFLOPS] となり，事実上これが目標性能となる．これは extend-load/store 命令と浮動小数点演算パイプライン数および実際の演算のバランス (2load，1store に対して 2 演算) で決まる．また，(3) のアルゴリズムの場合は，オンチップメモリ上にすべての行列が乗った場合，約 15.6[GFLOPS] の性能が得られる．

(1) のアルゴリズムに対しては，いくつかの場合を評価した．評価結果を表 2 に示す．表中の CASE はそれぞれ次に対応する．

- case1: (1) のアルゴリズムで，常に次に計算する列を prefetch し，計算し終えた列を poststore する
- case2: (1) のアルゴリズムで，最終的にオンチップメモリに行列が収まるまで計算が進んだら，その中で計算する
- case3: (1) のアルゴリズムで，最初からオンチップメモリの再利用性を考慮したアクセスを行なう
- case4: (2) のアルゴリズム

表 2 検討結果

CASE	Performance
case1	4.56 [GFLOPS]
case2	5.03 [GFLOPS]
case3	7.20 [GFLOPS]
case4	10.6 [GFLOPS]
case5	15.6 [GFLOPS]

- case5: (3) のアルゴリズム

4.2 考　　察

評価結果より，Linpack ベンチマークプログラムの default のアルゴリズム (1) では，オンチップメモリの再利用性に限界があり，目標性能の 10.6[GFLOPS] には遠く及ばない．case1 はオフチップメモリのスループットで決まるため，仮定しているシステム構成での最低性能と言える．case2 はほぼキャッシュの動作に等く，2MByte の従来通りのキャッシュがある場合の性能と考えられる．また case3 については，アルゴリズムの変更なしに多少の性能向上が見られる．しかし，これは今回の評価で行列サイズの 4 分の 1 をオンチップメモリに乗せることができたためであり，行列サイズが 10000×10000 のように規模が大きくなると，その性能は case1 程度まで落ちてしまう．

一方，オンチップメモリブロッキングを行なった case4，それに加えレジスタブロッキングを行なった case5 とも，ブロック内，あるいはブロック同士の演算をする間に，page-load/store 命令を用いたオンチップ・オフチップメモリ間の prefetch/poststore が完全にオーバラップして行なえる．したがって，オンチップ・オフチップメモリ間のスループットがボトルネックとならないため，それぞれの目標性能に達することができ，プロセッサ・メモリ混載型 LSI でのオンチップメモリの有効性を示す結果となった．

このようにブロッキングアルゴリズムを用い，またオンチップメモリ上のデータをうまく制御することができれば，システムのピーク性能に近い性能が得られることがわかった．他のプログラムにおいても，Linpack ベンチマーク同様オンチップメモリを効果的に用いることで高性能が得られると考えられる．

5. まとめと今後の課題

本稿では，プロセッサ・メモリ混載型 LSI について，主に HPC 分野のアプリケーションをターゲットとしたアーキテクチャの検討，およびその性能評価を行なってきた．

プロセッサ・メモリ混載型 LSI では，オンチップメモリ上のデータに対するアクセスが低レーテンシかつ高バンド幅であるため，高性能化が期待されるが，オンチップメモリに乗りきらないようなワーキングセットの大きなアプリケーションでも，ブロッキングなどの手法により高性能化が可能である．またその際，従来のキャッシュでの問題点も，オンチップ・オフチップメモリ間のデータアクセスを明示的に制御することで解決可能であると考えられる．

性能評価では，2003 年頃に可能となるであろうハードウェアをいくつか仮定し，評価を行ない，さらにハードウェア構成について検討した．また，Linpack ベンチマークによる性能評価では，ブロッキングアルゴリズムによりオンチップメモリの再利用性を活用することで，高い性能を得られることが確認された．

今後の課題としては，

- 他のアプリケーションでの性能評価
- シミュレータを用いた，詳細な性能評価
- アーキテクチャの詳細な検討
- 追加・拡張命令のためのコンパイラの作成
- 提案するアーキテクチャのゲートレベルの設計
- 並列計算機への適用

などが挙げられる．

特に現在は，本研究で検討しているアーキテクチャのクロックレベルシミュレータを作成中であり，シミュレーションにより他の様々なアプリケーションプログラムについての性能を示していく予定である．

謝辞　本研究を行なうにあたり，御助言，御討論頂いた筑波大学計算物理学研究センターの関係者各位に感謝致します．なお，本研究は日本学術振興会未来開拓学術研究推進事業「計算科学」のプロジェクトの一つとして行なわれている．

参 考 文 献

1) 村上 和彰, 岩下 茂信, 宮嶋 浩志, 白川 暁, 吉井 卓： メモリーマルチプロセッサー体型 ASSP（Application-Specific Standard Product）アーキテクチャ：*PPRAM*, 信学技報, ICD96-13, CPSY96-13, FTS96-13, 1996 年 4 月

2) David Patterson, Thomas Anderson, Neal Cardwell, Richard Fromm, Kimberly Keeton, Christoforos Kozyrakis, Randi Thomas and Katherine Yelick: A Case for Intelligent RAM: IRAM, IEEE Micro, April 1997

3) 岩崎 洋一, 中澤 喜三郎 ほか: 計算物理学と超並列計算機 – CP-PACS 計画 –, 情報処理, Vol.37, No.1, pp.10-42 1996 年

4) URL <http://www.starc.or.jp>

5) Monica S. Lam, Edward E. Rothberg and Michael E. Wolf: The cache performance and optimizations of Blocked Algorithms: Proc. of ASPLOS-IV, pp.63-74, April 1991

6) Preeti Ranjan Panda, Hiroshi Nakamura, Nikil D. Dutt and Alexandru Nicolau: A Data Alignment Technique for Improving Cache Performance: Proc. of ICCD'97, pp.587-592, October 1997

7) The National Technology Roadmap for Semiconductors 1997 Edition:Semiconductor Industry Assosiation (SIA), 1997 年