

スーパースケラプロセサの構成と、その性能評価

斎藤 光男 井上 淳 武田 謙治
東芝総合研究所

スーパースケラプロセサにおける様々なアーキテクチャパラメータを変化させた場合における性能を評価した。スーパースケラプロセサを設計、開発する際には、同時命令発行数、演算ユニット数、個々の演算器性能といった様々なアーキテクチャパラメータを考慮することが必要である。また、実アプリケーションでの性能を推定する際には、命令列並べ替えやループ最適化といったソフトウェアによる最適化効果も考慮して性能予測を行う必要がある。

我々は、様々な構成のスーパースケラプロセサに対して最適化されたコードを生成する最適化コンパイラと可変なプロセサ構造に対応するシミュレータを開発し、これを用いて各アーキテクチャパラメータがプロセサ性能に与える影響を評価した。

評価は、同時命令発行数、演算器構成、命令実行速度を変化させたものについてベンチマークプログラムにより行った。その結果、命令発行数、演算器構成については4命令発行、2ALU、2メモリポートで性能が飽和することと、浮動小数点ユニットに関しては、演算器数より演算命令速度の効果が大きいことを検証した。

Configuration of a Superscalar Processors and its Performance Evaluation

Mitsuo Saito, Atsushi Inoue, Kenji Takeda
Research and Development Center, Toshiba Corporation
1 Komukai-Toshiba-cho Saiwai-ku Kawasaki, Kanagawa 210

We present the evaluation results for analyzing the architectural design issues of a superscalar processor. We have developed general tools for evaluating a superscalar processor, which consists of a retargetable C compiler and a reconfigurable hardware simulator, both of which can accept various configuration of processors.

Using these tools, we generate optimal code for each configuration and simulate it. We have evaluated the following items; (1) number of instruction issued (2) Configuration of functional units (3) Speed of each instruction, with some benchmark programs. The results shows that, as for the instruction issue rate and the configuration of functional units, the processor model which issues 4 instructions per cycle and has 2 ALUs and 2 memory ports attains saturated performance, and as for the configuration of FPU, it is more necessary to reduce the latency of instruction than to have more operation units.

1 はじめに

近年提唱されている高性能プロセサの多くは命令レベルの並列性を重視したスーパースケラ・プロセサである。これらのスーパースケラ・プロセサを開発するには、同時実行命令数や演算ユニット数、ユニットの同時実行性など様々なアーキテクチャパラメータを決定することが必要となる。我々はこのような基礎評価を行うためのツール群を開発し、これらを用いてスーパースケラ・アーキテクチャの評価を行った。

スーパースケラ・プロセサの基本モデルを設定し、そのモデルを実現する可変構造ハードウェアシミュレータを用いて評価を行なう。評価対象となるスーパースケラ・プロセサモデルについて説明し、これをシミュレートするハードウェアシミュレータの特徴を述べる。

スーパースケラ・アーキテクチャを有効に使用するためには、コンパイラによる最適化が必須となる。本稿ではスーパースケラ・プロセサ用の最適化コンパイラ [1] を用いて、各テストプログラムの並列性を十分に引き出した状況で評価を行う。本コンパイラは、評価対象プロセサのアーキテクチャ特性をコンパイルオプションとして受けとり、そのアーキテクチャの下で命令スケジューリング、ループ最適化を行った最適コードを生成する。

最後に、(1) 同時実行命令数、(2) 整数ユニット構成、(3) 浮動小数点ユニットの構成、の3点について評価を行った結果を示し、スーパースケラ・プロセサの実性能に影響するアーキテクチャ項目を探る。

2 プロセサモデル

本節では評価対象となるスーパースケラ・プロセサのモデルについて述べる。

本プロセサは複数命令同時発行が可能な in-order issue、out-of-order completion のスーパースケラ・プロセサを基本モデルとする。本プロセサは、ハードウェア構成を簡単にし、クロック高速化による処理能力の向上を目指している。

4 命令発行モデルの基本構成を図 1 に示す。

演算器としては、2*組の 32 ビット整数 ALU、整数乗除算器 1 個、分岐器 1 個、浮動小数点加減算器 1*個、浮動小数点乗除算器 1*個を持つ。レジスタ構成は整数が 32 ビット長で 32 本であり、浮動小数点が 64 ビット長で 32 本の構成である。レジスタポートは各々 6* read, 4* write である。データキャッシュは 2*ポート構成で各ポートはそれぞれ 64 ビット長である。分岐予

*可変パラメータ

測機構としては 64 エントリの Branch Target Buffer を持ち、動的に分岐予測を行なう。予測成功時は 0 ペナルティであり、予測失敗時は 2 ペナルティである。命令セットは MIPS R3000 のスーパーセットである。

命令は最大 4 命令が同時に発行される。並列発行に際してはデータ依存解析とリソースコンフリクト解析に基づいて行われる。データ依存解析はスコアボードを用いて行なう。

各演算命令のレイテンシ*を表 1 に示す。

パイプライン構成は 5 段であり、各ステージの機能は以下の通りである。

1. F ステージ：命令を命令キャッシュから複数個（4 命令であれば 128 ビット分）を読み込む
2. D ステージ：複数命令を同時にデコードする。依存解析とリソース解析を行い、各命令が発行可能かどうかを調べる。同時発行可能な命令を E ステージに渡す。
3. E ステージ：演算を実行する。load/store 命令の場合、アドレス計算は E ステージで ALU を用いて行う。
4. M ステージ：メモリアクセスを行う。
5. W ステージ：結果をレジスタファイルに格納する。

浮動小数点演算は E ステージ以降が異なる。E ステージは E1, E2, E3 ステージの 3 段パイプライン構成になっており、除算を除いて毎サイクル、命令を投入することができる。乗算と除算は並列処理が可能である。バイパスは 2 段目以降から可能である。浮動小数点演算では M ステージがなく E3 ステージの次は W ステージとなる（図 2）。

3 評価ツール

本節では、開発した評価ツール（シミュレータ、言語処理系）の特徴を説明する。

3.1 スーパースケラシミュレータ

2 で説明したプロセサモデルのパイプラインを忠実に再現するパイプラインシミュレータを作成して評価に用いた [2]。本シミュレータは以下の特徴を持つ。

- a.out 形式のオブジェクトコードを直接実行

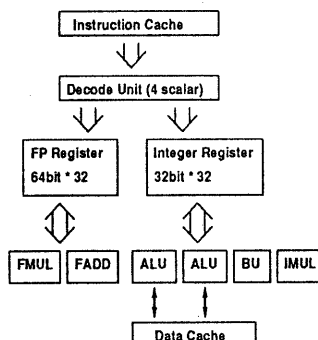


図 1: アーキテクチャの基本構成

表 1: 命令のレイテンシ

命令	latency
整数演算	0
整数ロード	1
FP ロード	2
整数・FP 間転送	2
整数乗算	12
整数除算	34
FP 絶対値, 符号反転	1
FP 型変換	2
FP 加減乗算	2
FP 除算 (単精度)	9
FP 除算 (倍精度)	16

- 実機と同一のクロック数（パイプラインを忠実に再現しているので、ベンチマークプログラムを実機と同一のクロック数で正確なシミュレーションが可能である）
- 大規模なベンチマークを実行可能
- 高速シミュレーション（クロック 25MHz の WS 上で最大 45KIPS の性能を達成した）
- 可変アーキテクチャ下でのシミュレーション可能
可変パラメータとして以下のものをサポート
 - 同時発行命令数
 - 演算器数（ALU, FP 加算器, FP 乗算器）
 - メモリポート数
 - 命令のレイテンシ

本シミュレータは以下の 3 個の構成要素からなる。

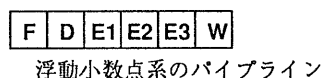
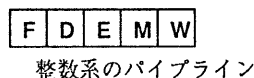


図 2: パイプライン構成

1. システムコールサポート部：本シミュレータでは、ベンチマークプログラム中のシステムコールは、制御をホスト計算機に移して行う方式とした。従って、システムコールのシミュレーションは行っていない。しかし本方式ではベンチマークプログラムを全く書き換えずに忠実な実行が可能という特徴を持つ。
2. シミュレータ部：プロセッサ本体が C 言語を用いて記述されている。MMU や Exception Handler 等のベンチマークの動作に不要な機能は省いている。
3. モニタ部：オブジェクトコードのロード、デバッグ機能等を有する。

3.2 最適化コンパイラ

今回の評価にはスーパースケラ・プロセッサ用に開発した最適化コンパイラ [1] を用いた。本コンパイラのスーパースケラ用最適化を行うバックエンド部分の構成を図 3 に示す。このバックエンド部は大きく 2 つに分かれ、各々以下の処理を行う。

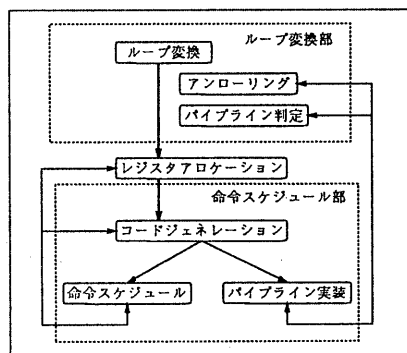


図 3: コンパイラ構成

- ループ変換部

- 大域的なループ構造の抽出、依存関係のリスト
- アンローリングの段数推定、実現
- パイプライン実現予測
- ループ交換

• 命令スケジュール部

- リストスケジュールによる局所的命令スケジュール
- パイプラインスケジュールによる大域的命令スケジュール
- レジスタリネーミング
- 命令列のアラインメント処理
- 命令コンパクション

ループ変換部では、プログラム内のループ構造を抽出し依存解析を行う。この結果はスケジュール部でのソフトウェアパイプライン変換の可否判定に使用するほか、ループ交換処理もこの解析に基づいて実行する。また、ループ内の演算内容、使用レジスタ数を調べ、最適な展開段数を推定しループアンローリングを行う。

命令スケジュール部では、リストスケジューリングによる基本ブロック内スケジューリングとループ部分のソフトウェアパイプライン変換を行う。命令スケジュール部は対象となるプロセッサの特性パラメータを用いて任意のスーパースケラ構成に対する最適なスケジューリングを生成できる。パラメータとしては、表2に示すものをコンパイル時に指定することができる。

例えば、4命令発行、ALU2個、メモリポート2個、FP加算器1個、FP乗算器1個、FP演算レイテンシ2、FPロードレイテンシ2のプロセッサパラメータを与えて、図4のプログラムをスケジューリングした結果を図5に示す。同じプログラムを2命令発行、ALU、メモリポート1個に制限した場合のスケジューリング結果は図6のようになる。ALU数の制約から複数のload/store命令、アドレス加算命令が逐次化されてスケジューリングされていることが判る。

今回の評価では、この可変パラメータスケジューリングを用いて、各アーキテクチャに対する最適コードを生成し評価を行った。

```

1.  ld    f6,0(r4)
2.  subd   f7,f6,f2
3.  sd     f7,0(r5)
4.  ld     f8,8(r4)
5.  subd   f9,f8,f6
6.  sd     f9,8(r5)
7.  addiu  r4,r4,16
8.  addiu  r5,r5,16
9.  movd   f2,f8
10. bne    5,2,loop

```

図 4: スケジューリング前のコード

```

loop:
  ld f6,0(r4)      ld f8,8(r4)      .      .
  -
  subd f7,f6,f2    .      .      .
  subd f9,f8,f6    .      .      .
  -
  sd f7,0(r5)      addiu r4,r4,16    movd f2,f8    .
  sd f9,8(r5)      addiu r5,r5,16    bne 4,2,loop .

```

図 5: 4 命令発行プロセッサに対するスケジューリング結果

4 性能評価

評価用プログラムとしては Stanford ベンチマーク、Livermore ループ、SPEC の一部 (espresso/eqntott/matrix300/tomcatv) を使用する。

評価は次の3項目について行った。

1. 同時命令発行数の効果を調べる。最大同時発行命令数、演算ユニット構成を変えた3種類のプロセッサを対象に並列化効果を評価する。浮動小数点系プログラムについては、命令実行速度 (レイテンシ) を変えた場合の影響も調べる
2. 整数系プログラムについて命令発行数を固定し、整数ユニット構成要素 (ALU 数、メモリポート数) を変えた場合の高速化効果を調べる
3. 浮動小数点系プログラムについて命令発行数、ALU 数を固定し、FP 演算器数、メモリポート数、FP 演算命令レイテンシ、FP ロード命令レイテンシの効果を調べる。

```

loop:
  ld f6,0(r4)      .
  ld f8,8(r4)      .
  -
  subd f7,f6,f2     .
  subd f9,f8,f6     .
  addiu r4,r4,16    .
  sd f7,0(r5)       .
  sd f9,8(r5)       .      movd f2,f8
                           addiu r5,r5,16
  bne 4,2,loop

```

図 6: 2 命令発行プロセッサに対するスケジューリング結果

表 2: スケジューリング・パラメータ

項目	意味	default
inst_per_cycle	1 サイクル命令発行数	4
alu	ALU 数	2
mul	整数乗算器数	1
mem	メモリポート数	2
load	1 サイクルの load 命令数	2
store	1 サイクルの store 命令数	2
fadd	FP 加算器数	1
fmul	FP 乗算器数	1
bra	分岐ユニット数	1
iuld_latency	整数 load のレイテンシ	1
fpuld_latency	FP-load のレイテンシ	2
fadd_latency	FP 加算、乗算のレイテンシ	2

4.1 同時命令発行数の評価

同時命令発行数の効果を評価するため、次の 3 種類のプロセッサ構成についてシミュレータ上での実行速度を調べた。

2inst: 最大 2 命令同時発行、ALU 数 2、メモリポート数 1、FP 加算器 1、FP 乗算器 1

4inst: 最大 4 命令同時発行、ALU 数 4、メモリポート数 2、FP 加算器 2、FP 乗算器 2

8inst: 最大 8 命令同時発行、ALU 数 8、メモリポート数 4、FP 加算器 2、FP 乗算器 2

ここで、メモリポート数は 1 サイクルに同時実行できる load/store 命令数の上限を示す。FP 演算命令は、各プロセッサが持つ演算器数の制限を越えない範囲で任意の組合せで同時使用できる。

整数系プログラム (Stanford, iSPEC) の実行結果を図 7 に示す。各プログラムについて 1 命令発行プロセッサモデル¹に対する相対性能を示している。各プロ

¹ 整数ユニットと浮動小数点ユニットは並行に動作する

グラムとも、2inst モデルと 4inst モデルの間では一定の性能向上が見られるが 8inst モデルにしても性能向上は得られず、1.15 倍から最大で 1.6 倍程度しか高速化できない。これは個々のプログラムの基本ブロック長が短く、(現在の処理系ではブロックを越えるスケジューリングを行っていないため) 必要以上のハードウェアを持っても、それらを十分に使用するコードが生成できないためと考えられる。一般に整数系プログラムを局所命令スケジューリングのみで高速化する場合の発行命令数は 4 で十分であることがいえる。[†]

浮動小数点系プログラム (Livermore, fSPEC) の実行結果を図 8 に示す。3 種類の命令発行数 (FP 演算、FP ロードのレイテンシは 2 とする) について、同様に 1 命令発行モデルに対する相対性能を示す。

Livermore ループで個々のループに対する性能を見た場合、命令発行数を増やしても高速化できないループ (L01,L03) と命令発行数の効果が顕著に現れるループ (L12,L21) に大きく別れることが判る。大規模なプログラムでは、このような異なる性質のループが複数存在して、それらの効果が加重された形で並列化効果が現れると考えられる。図の LIVALL は Livermore ループの 1 から 14 ままで初期化ループを含めて実行したものである。この場合は命令発行数が増えるにつれて徐々に並列化効果が現れていることが判るし、fSPEC についても (高速化の程度にばらつきはあるが) 命令発行数の効果が現れている。

同じ評価を、各モデルの FP 演算命令、FP ロード命令レイテンシを 1 とした場合についてまとめたものを図 9 に示す。命令発行数の効果の小さいループ (L01,L03) について見ると 2 レイテンシの場合に比べて各モデルとも大きく性能が上っている。特に 4,8 命令発行時の性能が大きくなっている。これらのループはデータ依存による待ちサイクルが多く発行命令数の効果は一定値で頭打ちになるが、命令レイテンシを減少することでループ全体のクリティカルパスが短くなり、性能が大幅に向上する。特に、命令発行数の効果の大きいループ群 (L12,L21) は、レイテンシを減らすと、命令発行数が 8 の時に高速化が著しい。

4.2 整数ユニット構成の評価

次に整数系プログラムについて、整数ユニット (ALU 数、メモリポート数) の構成を変えた場合の効果を調べた。この評価では 4.1 の結果を考慮し、命令発行数は 4 に固定した場合の最適な ALU 数、メモリポート数を

[†] eqntott は今回使用したデータセットでは CPU 性能が支配的な部分の効果が大きく、4inst 以上で 3 倍以上高速化される

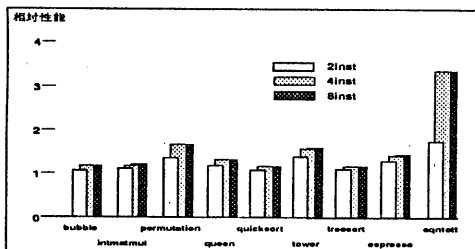


図 7: 命令発行数の効果 (整数系プログラム)

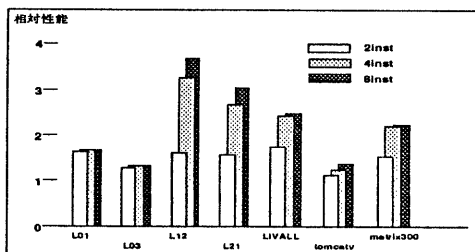


図 8: 命令発行数の効果 (FP レイテンシ 2)

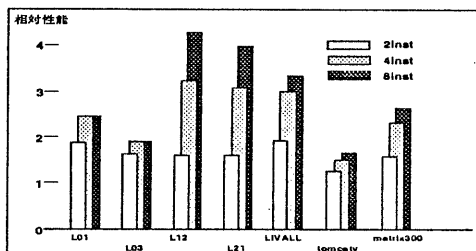


図 9: 命令発行数の効果 (FP レイテンシ 1)

調べる。

メモリポート数、ALU 数をそれぞれ 1～3 と変えた場合の実行結果 (1ALU、1メモリポート結果との相対性能) を図 10 に示す。大部分のプログラムでは、ALU、メモリポート数を各々 2 以上増やしてもそれ以上は高速化できず、整数系のプログラムにおいては、整数ユニット構成としては ALU、メモリポート数はそれぞれ 2 個持てば十分でありハードウェアコストとのトレードオフに帰着することが判る。

4.3 浮動小数点ユニット構成の評価

最後に浮動小数点ユニットの各構成要素を変化させた場合の性能を評価する。評価は、8 命令同時発行のプロセッサで、評価対象となる項目以外はプログラム

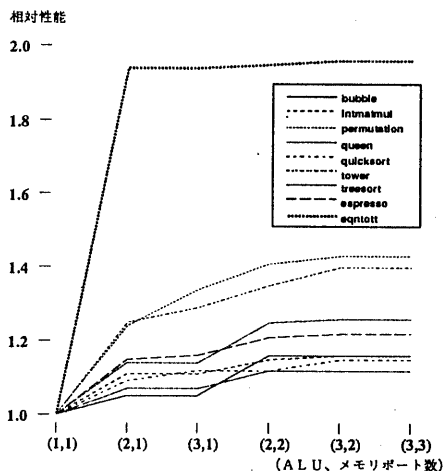


図 10: ALU、メモリポート数の効果

実行を制限しないことを想定し、各項目の制約条件を変化させた実行性能を求める。例えば FP ユニット数を評価する際には、ALU、メモリポートは 8 個分をフルに使用できるとし、命令レイテンシ評価の場合はすべての演算ユニットが 8 個まで使用できるとする。

評価対象の項目は、

- FP 演算ユニット (加算、乗算器) 数
- メモリポート数 (FP ロードストアは IU 上で実行されるため)
- FP 演算命令のレイテンシ
- FP ロード命令のレイテンシ

として、Livermore 1～24 ループを最速段数でアンローリングした場合の性能を求めた。

FP 演算ユニット数を変えた場合の実行結果を図 11 に示す。各ループに対し、1FPU と比較した相対性能を示している。全 28 ループ中、7 ループのみが FPU 数変更の効果を示し、他のループでは効果は小さい。また FPU 数の効果が見られたループについても、#12 を除いては 2FPU 以上の変化はない。この結果から、一般には FPU 数は 2 が最適でそれ以上の FPU を有効に使用できるケースはまれであるといえる。

データロード、ストアの影響を調べるため、メモリポート数を 1～4 と変えた場合の実行結果を図 12 に示す。同様に 1 メモリポート実行との相対性能を示す。この場合も、効果の見られるのは 4 ループのみで、一部のデータ初期化ループ (#14-1) 以外は利得は小さいことが判る。

以上の結果を見ると、Livermore ループに関しては一部の例外を除いてはプロセッサ構成を変化しても多くの効果は得られないといえる。これは、

- ループ構造が単純であっても、データ依存の影響で演算実行待ちになるケースでは余分な演算を実行できないことがある。
- 分岐を含むループでは、基本ブロック自体が小さいため並列化対象命令が少ない

などの理由によると考えられる。

次に命令レイテンシを変えた場合の効果調べる。FP 演算命令 (fadd, fmul) レイテンシを 1~5 と変えた場合の実行結果を図 13 に示す。グラフは横軸に演算の実行サイクル数 (1 レイテンシの場合 2 サイクル実行) を縦軸に 1 レイテンシ実行との相対性能を log スケールでプロットしてある。太線は演算レイテンシのみが性能劣化に影響したと仮定した下限値 (命令実行に 2 倍かかった場合、性能が半減する) カーブを示す。

各ループともほぼ直線に乗った傾向を示し、多くのループでレイテンシ増加による劣化の程度は非常に大きい。特に、ループ内の演算実行系列が決まっている場合 (データ依存がある場合も含めて)、命令スケジューリングで最適配置を行った結果、演算実行待ちサイクルがループ全体の性能を決定するようになる。このため演算レイテンシを短縮することが直接ループの実行性能に効くようになる。

この実験でも内部に分岐を含むループは変化が小さいが、これは演算実行待ちより分岐によるパイプライン途切れの影響がでたものと考えられる。

同様に FP ロード命令レイテンシを 1~10 まで変えた場合の結果を図 14 に示す。FP 演算レイテンシの場合と比べると、ロード命令レイテンシが増加した場合の劣化は小さいことが判る。これはループアンローリングとスケジューリングを併用する場合、各ロード命令は別の演算と並列に実行できるので、ロードレイテンシが増加する影響はループの先頭の演算準備段階でしか影響がないためと考えられる。しかし一部のループでは、レイテンシが一定値を越えた場合にロードレイテンシサイクルに他の命令を埋め込むことができなくなり、劣化の程度が大きくなる場合もある (グラフの折れ曲がっているもの)。

この評価では内部に分岐を含むループの劣化が大きい。複数の分割された基本ブロックの先頭では新たな演算系列に対するロードが単独で実行されるため、そ

(FPU 1 個に対する)
性能比

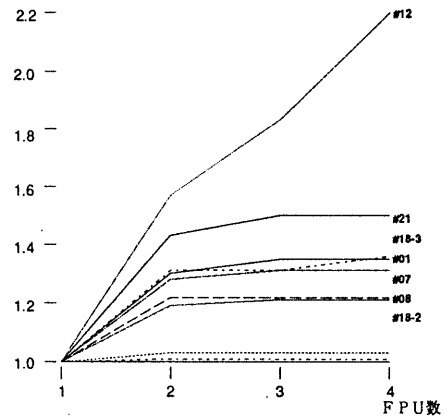


図 11: FPU 数の効果

のレイテンシが大きく影響するものと考えられ、これは一般に非数値演算の場合も同様と考えられる。

5 おわりに

スーパースケラ・プロセッサにおける基礎評価のためのツールとして開発した可変構造ハードウェアシミュレータ、最適化コンパイラの構成と、これを用いた評価結果を説明した。本評価は任意に設定したプロセッサモデルに対し最適なおブジェクトコードを生成して対応するシミュレータ上で実行評価を行うもので、最適で、より実稼働状態に近い状態の評価を行うことが可能である。

本稿では、(1) 同時発行命令数、(2) 整数ユニット構成、(3) 浮動小数点ユニットの構成の 3 点について評価を行った。

命令発行数に関しては、整数系プログラムの大部分は 4 命令発行段階で最適な性能を示し、それ以上の命令発行の効果は小さいことが判った。浮動小数点系プログラムは命令発行数の効果のほとんどない場合と、命令発行数が大きな効果を持つ場合があり、大規模なプログラムでは 2,4,8 と発行数を増やすにつれて徐々に性能が上がるが判った。

整数ユニット構成に関しては、ALU、メモリポート数を変えた場合の実行性能を評価し、整数系の大部分のプログラムで (4 命令発行) 2ALU, 2メモリポートで性能が飽和することが判った。最適な個数はハードウェアコストとのトレードオフの問題に帰着する。

浮動小数点ユニット構成については、FP ユニット

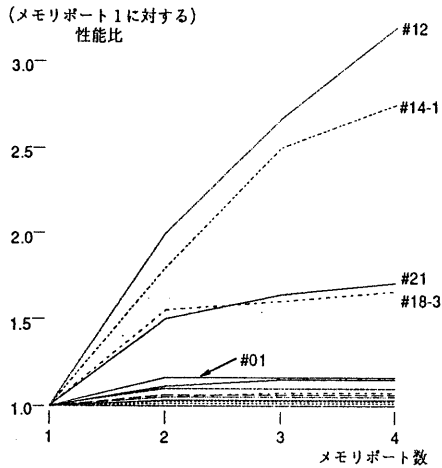


図 12: メモリポート数の効果

数、メモリポート数、FP 演算レイテンシ、FP ロード命令レイテンシの4点について評価を行った。Livermore ループを対象とした評価結果を見ると、FP ユニット、メモリポートなどの演算器構成を増やしても、最適化コードではさほど性能は向上しないこと、反面、FP 演算レイテンシの増加は多くのプログラムで大幅な性能劣化をもたらすこと、FP ロード命令レイテンシの増加は、一定の範囲を越えなければ FP 演算ほどの影響はないが、分岐の多いプログラムでは性能劣化が大きいこと、などが判った。

参考文献

- [1] 白川 健治、井上 淳. “スーパースカラプロセッサにおけるループ並列化の検討”. 情報処理学会研究報告, 91-ARC-91-3, pp. 21-28, Nov. 1991.
- [2] 武田 謙治、井上 淳、白川 健治. “スーパースカラプロセッサの高速シミュレータによる評価”. 情報処理学会第 44 回全国大会 2D-1, Mar. 1992.

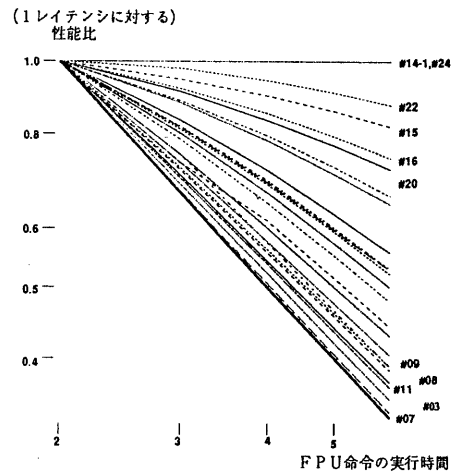


図 13: FP 演算レイテンシの効果

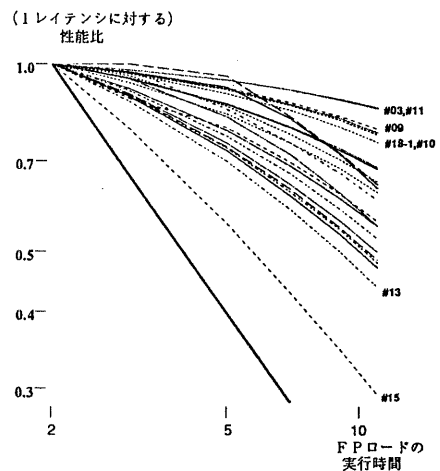


図 14: FP ロードレイテンシの効果