計算機アーキテクチャ　107－4
（１９９４．　７．　２１）

ハイパースカラ・プロセッサ『中洲１号』のアーキテクチャ

宮嶋 浩志　　村上和彰

九州大学 大学院総合理工学研究科 情報システム学専攻

〒816 福岡県春日市春日公園 6-1

E-mail: {miyajima, murakami}@is.kyushu-u.ac.jp

筆者らは現在，ハイパースカラ・プロセッサ「中洲１号」の開発を行っている．

「中洲１号」は，通常のRISCプロセッサの命令セットに，ハイパースカラ方式特有の変更および拡張を施した命令セットをもつ32ビット・マイクロプロセッサである．「中洲１号」は基本的にスーパースカラ・プロセッサと同じ構成で，通常２命令同時に実行する．また，並列動作可能な機能ユニット対応に設けた命令レジスタから，予めロードしておいた命令を５命令同時に発行し実行する．

本稿では，ハイパースカラ・プロセッサ「中洲１号」の命令セット・アーキテクチャ，ハードウェア構成，および，チップ・フロアプランについて述べている．

The Architecture of the Hyperscalar Processor: NAKASU-1

Hiroshi MIYAJIMA　Kazuaki MURAKAMI

Department of Information Systems
Interdisciplinary Graduate School of Engineering Sciences
Kyushu University
6–1 Kasuga-koen, Kasuga-shi, Fukuoka 816 Japan

E-mail: {miyajima, murakami}@is.kyushu-u.ac.jp

We have been developing a hyperscalar processor, called NAKASU-1.

NAKASU-1 is a 32bit microprocessor that has an extended RISC instruction set. NAKASU-1 has a superscalar organization that evables two instructions to be excuted in parallel. Also, NAKASU-1 has instruction registers for each concurrently executable function unit. Five of instructions preloaded to those instruction registers can be executed in parallel.

This paper describes the instruction set architecture, hardware organization, and floor plan, of the hyperscalar processor NAKASU-1.

1 はじめに

ハイパースカラ・プロセッサ・アーキテクチャ(*Hyperscalar processor architecture*: 以下，ハイパースカラ方式) [1] とは，従来の命令レベル処理方式である，命令パイプライン処理方式，スーパースカラ方式，超長形式機械命令 (VLIW) 方式，および，ベクトル処理方式の短所を排し長所を包括した命令レベル処理方式である [1].

ハイパースカラ方式とは，簡約すれば，

- 命令長および命令フェッチ巾は，スーパースカラ方式と同程度に抑える，
- 機能ユニット (*FU*) 対応に (1 個以上の) ユーザ可視の命令レジスタ (*IR*) を設けて，それに (解読済みの) 命令をロードすることで VLIW プログラムをプロセッサ内部に形成し，あたかも VLIW プロセッサの如く動作させる，
- さらに，用途に応じて必要ならベクトル・レジスタを設け，命令レジスタ内に形成した VLIW 命令のループにより，ベクトル・データに対して，**擬似ベクトル処理** (ベクトル命令の動作をスカラ/VLIW 命令のループで模擬する)，あるいは，**ソフトウェア・パイプライニング処理** (ソフトウェア・パイプライン化されたスカラ/VLIW 命令のループで処理する) を施す，

方式である [1].

よって，その対象とするアプリケーション分野は，

- スーパースカラ・プロセッサが得意とする「命令レベル並列度がさほど高くない非科学技術計算分野 (ビジネス・アプリケーション)」から，
- VLIW プロセッサおよびベクトル・プロセッサが得意とする「命令レベル並列度の高い科学技術計算分野」まで，

と，ほぼ全方位にわたる [1].

しかもハードウェア・コストは，命令レジスタおよびベクトル・レジスタ分を除くと，スーパースカラ・プロセッサ並みであり，VLIW プロセッサやベクトル・プロセッサよりかなり小さい [2]. また，ハイパースカラ方式を用いたプロセッサが取り得る設計空間 (デザイン・スペース) は，かなり広いものになる [4].

現在我々は，

- ハイパースカラ方式の有効性を示す，
- 設計上の課題を明確にする，

ことを目的として，ハイパースカラ・プロセッサ「中洲 1 号」の開発を行っている [5]. また，ハイパースカラ用最適化コンパイラの開発も同時に進めている [3, 6, 7].

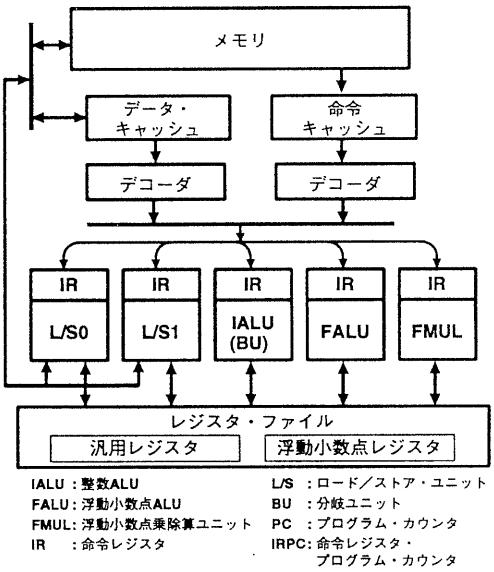


図 1: 「中洲 1 号」の全体構成

本稿では，まず 2 章でハイパースカラ方式について，3 章で現在開発中の「中洲 1 号」のアーキテクチャについて述べる．次に，4 章で「中洲 1 号」のハードウェア構成に関して述べる．最後に，5 章で今後の課題について述べ本稿のまとめとする．

2 ハイパースカラ方式

図 1に，ハイパースカラ・プロセッサの基本構成例として，現在開発中の「中洲 1 号」の全体構成を示す．

ハイパースカラ・プロセッサの構成は，基本的には通常のスカラ・プロセッサ，あるいは，スーパースカラ・プロセッサと変わらない．命令長および命令フェッチ巾は，スーパースカラ・プロセッサと同程度とする．例えば，命令長は 32 ビットで命令フェッチ巾は 1〜4 命令程度で構わない．これにより，VLIW 方式の短所であるコード・サイズの増加および命令キャッシュの低使用効率といった問題を解決する．

2.1 命令レジスタ

ハイパースカラ方式のスーパースカラ方式に対する本質的な相異点は，並列動作可能な機能ユニット (*FU*) ごとに，複数個のユーザ可視の命令レジスタ

(*IR*) を設けた点である．*FU* 数を f，各 *FU* 当たりの *IR* 数を r とすると，計 $f \times r$ 個の *IR* が存在する．

IR が構成するアドレス空間は，$f \times r$ の 2 次元配列となる．このとき，各行はあたかも，f 個のフィールドから成る 1 個の VLIW 命令のように見える．また，*IR* 全体では，最大 r 命令から成る 1 個の VLIW プログラムのように見える．各 *IR* は，対応する *FU* にディスパッチする (解読済み) 命令を r 命令まで格納する．*IR* へ命令を格納するためのロード命令が，命令セットに加わる．

2.2 動作原理

ハイパースカラ・プロセッサの動作モードとしては，少なくとも次の 2 モードが定義可能である．

- **通常** (*normal*) モード：一般のスーパースカラ方式と同様，命令キャッシュからフェッチしてきた命令をデコードして，対応する *FU* にディスパッチする．*IR* は使用しない．
- **加速** (*turbo*) モード：命令キャッシュからではなく，*IR* から (解読済み) 命令をフェッチして，一意に対応する *FU* にディスパッチする．

通常モードおよび加速モードにおいて 1 クロック・サイクル当たり同時に実行開始可能な命令数をそれぞれ，**スーパースカラ度**および**ハイパースカラ度**と呼ぶ．

命令セットとしては，通常モードと加速モードとの間で制御を遷移させる以下の命令が加わる．

- ***IR* ディスパッチ開始命令**: 通常モードで実行可能で，通常モードから加速モードへ制御を遷移させる．すなわち，命令キャッシュからの命令フェッチを停止し，代わりに指定された *IR* から命令フェッチを開始する．*IR* 内の VLIW プログラムを「手続き」と見なせば，本命令は「手続き呼出し」命令に相当する．
- ***IR* ディスパッチ終了命令**: 加速モードで実行可能で，*IR* ディスパッチ終了条件が成立するか否かを判定し，成立する場合は加速モードから通常モードへ制御を遷移させる．すなわち，
  - *IR* ディスパッチ終了条件が成立する場合：*IR* からの命令フェッチを停止し，命令キャッシュの指定されたアドレスから命令フェッチを再開する．
  - *IR* ディスパッチ終了条件が成立しない場合：加速モードのままで，指定された *IR* から命令フェッチを続ける．

  本命令は，「手続き復帰」命令および「ループ戻り分岐」命令に相当する．

以上から，*IR* の典型的な利用法として，次のものが可能となる．

① 命令レベル並列度が小さい部分では，通常モードを用い通常のスーパースカラ・プロセッサ同様にプログラムを実行する．

② ループのように命令レベル並列度が高い部分を実行する場合は，まず，当該プログラム部分を *IR* ロード命令により *IR* に予めロードする．

③ *IR* へのロードが完了したら，*IR* ディスパッチ命令により加速モードに遷移し，*IR* からの命令ディスパッチを開始する．

④ *IR* ディスパッチ終了条件が成立するまで，*IR* からの命令ディスパッチを続ける．

⑤ *IR* ディスパッチ終了命令により，*IR* ディスパッチ終了条件が成立したら通常モードに遷移する．

⑥ 再び，通常モードで通常のスーパースカラ・プロセッサ同様のプログラム実行を再開する．

IR に格納するプログラム部分，つまり，加速モードで実行するプログラム部分としては，命令レベル並列度の高いループ部分が有力な候補である．ループ部分の実行形態としては，次の 2 つの手法が考えられる [1, 8]．

- ソフトウェア・パイプライニング処理
- 擬似ベクトル処理

上記の手法の詳細については，文献 [1] を参考されたい．

3 命令セット・アーキテクチャ

3.1 概要

「中洲 1 号」の命令セットは，通常の 32 ビット長 RISC プロセッサの命令セットを基に，後述するハイパースカラ方式固有の変更および拡張を施したものである [5]．

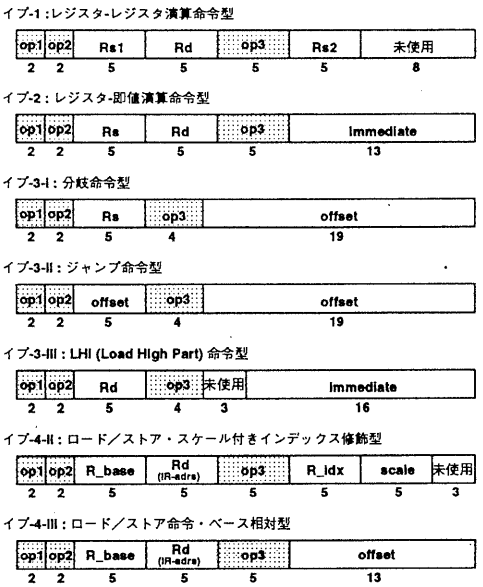
図 2 に「中洲 1 号」の命令形式を示す．すべての命令は 32 ビット固定長である．命令形式は大きく 4 種類に分類される．

整数演算および浮動小数点演算ともに，演算はすべてレジスタ-レジスタ間，あるいは，レジスタ-即値間で行う (ロード／ストア・アーキテクチャ)．

3.2 レジスタ

「中洲 1 号」は以下のレジスタを備える．

- プログラム・カウンタ (PC)：
  32 ビット長 1 個．加速モードでは機能しない．



OP1,OP2,OP3：命令操作コード・フィールド
Rs,Rs1,Rs2：ソース・レジスタ指定フィールド
Rd：デスティネーション・レジスタ指定フィールド

図 2: 命令形式

- 汎用レジスタ (R0～R31)：
  32 ビット長 32 個.
- 浮動小数点レジスタ (F0～F31)：
  64 ビット長 32 個.
- ステータス・レジスタ (SR)：
  プロセッサ状態を保持する．モード状態フラグ，スーパーバイザ・フラグ，等.
- 浮動小数点状態レジスタ (FSR)：
  浮動小数点比較命令の結果を保持する.
- 命令レジスタ・プログラム・カウンタ (IRPC)：
  5 ビット長 1 個．通常モードでは機能しない．加速モード時に，*FU* にディスパッチすべき命令を指定する.
- 命令レジスタ (IR)：
  (32 個/*FU*) × 5*FU*.

3.3 アドレシング・モード

「中洲 1 号」は，以下のアドレシング・モードを備える.

- PC(プログラム・カウンタ) 相対アドレシング：分岐/ジャンプ命令用
- ベース相対アドレシング：ロード/ストア命令用
- スケール付きインデックス修飾アドレシング：ロード/ストア命令用

加速モードにおいて，配列アクセスを行う際には配列要素のアドレス計算がボトルネックとなる可能性がある．これを回避するために，アドレシング・モードとして，ベース相対アドレシング・モードに加えて，スケール付きインデックス修飾アドレシング・モードを定義している [5].

3.4 命令の概要

「中洲 1 号」の命令セットは，次のように分類される.

- 整数演算命令
- 浮動小数点演算命令
- 分岐／ジャンプ命令
- ロード／ストア命令
- ハイパースカラ方式特有命令

整数演算命令，浮動小数点演算命令，および，ロード／ストア命令は，通常モードと加速モードとの間で動作に差異はない．一方，分岐／ジャンプ命令，および，ハイパースカラ方式特有命令には，

- 通常モードと加速モードとの間で動作が異なる命令，
- 通常モードあるいは加速モードのどちらか一方でしか有効でない命令，

がある.

3.4.1 命令の変更および拡張点

- 加速モードにおける分岐命令は，通常のプログラム・カウンタ (PC) ではなく，命令レジスタ・プログラム・カウンタ (IRPC:*Instruction Register Program Counter*) を用いる.
- いずれの *FU* でもレジスタ間転送を行えるように，各種のレジスタ間転送命令を定義する．これは，ソフトウェア・パイプライニング処理を阻害する要因であるイタレーション間の依存関係の解決するために，レジスタ間転送を多用するからである [3, 6].
- データ・キャッシュをバイパスして，必ずメモリに対してアクセスを行うメモリ直接アクセス命令を設ける．これは，キャッシュが効かないアプリケーションへの対処，ならびに，メモリ・アクセス・レイテンシを一定にしてソフトウェア・パイプライニングを容易にすることを目的にしたものである [5].

3.4.2 ハイパースカラ方式特有命令

- FLUSH (*FLUSH instruction registers*) 命令：
  全ての *IR* に no-op をロードする．通常モードで有効である.

- LDIR (*LoaD Instruction Register*) 命令：指定した *IR* へ指定したメモリ・アドレスから命令をロードする．通常モードで有効である．
- JALR2T (*Jump And Link Register to Turbo mode*) 命令：*IR* ディスパッチ開始命令．次PC値を指定した汎用レジスタに退避し，IRPCに即値をセットして，加速モードに遷移する．通常モードで有効である．
- TQEQZ (*Turbo mode Quit EQual Zero*) 命令およびTQNEZ (*Turbo mode Quit Not Equal Zero*) 命令：*IR* ディスパッチ終了命令．*IR* ディスパッチ終了条件が成立するか否か，すなわち，指定した汎用レジスタの値が0か否かを判定する．条件成立時は，指定した別の汎用レジスタの値をPCにセットして，通常モードに遷移する．条件不成立時は，IRPCに即値をセットして，加速モードをそのまま続行する．加速モードで有効である．

4 ハードウェア構成

図3に「中洲1号」のパイプライン構成の概略図を示す．

現在，最先端の商用マイクロプロセッサの集積度が300万トランジスタ程度であることを踏まえ，数百万トランジスタ程度のトランジスタの使用を前提として設計を行っている．

4.1 機能ユニット構成

「中洲1号」は，以下に示す6個の独立に動作可能な機能ユニット (*FU*) を装備する．

- 整数演算ユニット (IALU)×1
- 分岐ユニット (BU)×1
- 浮動小数点加減算ユニット (FALU)×1
- 浮動小数点乗除算ユニット (FMUL)×1
- ロード／ストア・ユニット (L/S)×2

ただし，IALUとBUは命令パイプラインとしては統合化され，これらに対して同時に命令をディスパッチすることはできない．すなわち，加速モード時，IALUとBUのどちらか一方，FALU，FMUL，L/S_0，および，L/S_1の計5個の *FU* に対して同時に命令がディスパッチ可能である．よって，ハイパースカラ度は5である．また，スーパースカラ度は2とした．

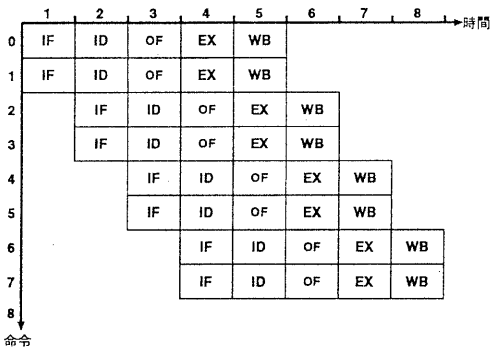


図 4: 命令パイプライン処理過程 (通常モード)

4.2 命令パイプライン処理過程

図4および図5に，通常モードおよび加速モードにおける命令パイプライン処理過程をそれぞれ示す．各ステージの動作は以下の通りである．

- 通常モード：5ステージ構成 (図4参照)
  - ① IF：命令キャッシュから命令フェッチ．
  - ② ID：命令デコード，および，各 *FU* への命令ディスパッチ (最大2命令)．
  - ③ OF：レジスタ・ファイル読出し，および，BUにおける分岐命令実行．
  - ④ EX：各 *FU*(BUを除く) における命令実行．
  - ⑤ WB：レジスタ・ファイル書込み．
- 加速モード：4ステージ構成 (図5参照)
  - ① IR：命令レジスタからデコード済みの命令フェッチ，および各 *FU* への命令ディスパッチ (最大5命令)．
  - ② OF：レジスタ・ファイル読出し，および，BUにおける分岐命令実行．
  - ③ EX：各 *FU*(BUを除く) における命令実行．
  - ④ WB：レジスタ・ファイル書込み．

分岐ユニット (BU) を除く全 *FU* は演算パイプライン化されており，毎サイクル新しい命令を実行可能である (BUはシングル・サイクル命令)．各 *FU* の命令実行所要サイクルを表1に示す．コピー命令は，各 *FU* とも1サイクルで実行する．

なお「中洲1号」では，レジスタ・オペランドに関するデータ・ハザードの解消は，静的に行うことを前提としており，動的にハザード解消は行わない．

フォワーディング　後続命令に対してフォワーディングを行う．すなわち，各 *FU* のEX(実行) ステー

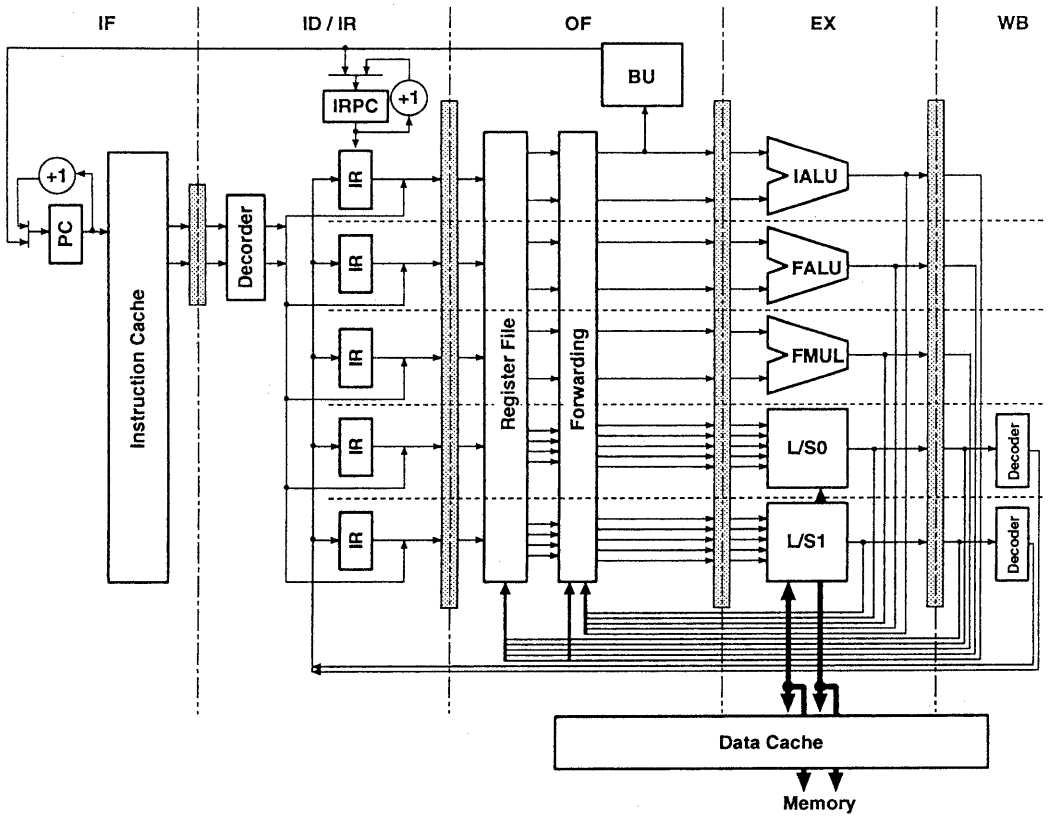


図 3: 「中洲１号」のパイプライン構成

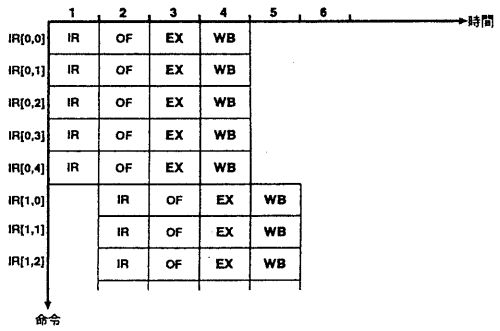


図 5: 命令パイプライン処理過程 (加速モード)

表 1: 各機能ユニットの実行所要サイクル

機能ユニット	実行所要サイクル数
IALU	1(乗算以外)
	2(乗算)
FALU	4(コピー命令以外)
	1(コピー命令)
FMUL	4(コピー命令以外)
	1(コピー命令)
L/S	1 + メモリ・アクセス・レイテンシ†
	2‡
	1(コピー命令)

†: メモリ直接アクセス命令
‡: メモリ直接アクセス命令以外

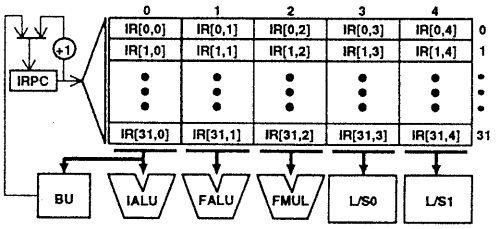


図 6: 「中洲 1 号」の命令レジスタ構成

ジ，および，WB(レジスタ書込み) ステージの出力を OF(オペランド・フェッチ) ステージへ送る (図 3参照).

4.3 命令レジスタ構成

命令レジスタは，図 6に示すように各 FU に 32 段ずつ設ける.

命令レジスタの段数は，文献 [5, 6] での検討結果に基づいて決定した．各命令レジスタはデコード済みの命令を格納し，レジスタ長は約 32 ビットである．したがって，命令レジスタの全容量は，約 32 ビット×32 段×$5FU = 5K$ ビットとなる.

4.4 レジスタ・ファイル構成

図 7に，「中洲 1 号」のレジスタ・ファイルの構成を示す.

レジスタ・ファイルの構成に関しては，さまざまな選択肢が存在する [2]．たとえば，論理的に構成に関しては，

- スカラ・レジスタのみを備える.
- ベクトル・レジスタのみを備える.
- スカラ・レジスタとベクトル・レジスタの両方を備える.

といった選択肢が可能である.

文献 [3, 6] での検討の結果，ベクトル・レジスタを備えることの効果は確認できたが，ハードウェア・コストの増加に見合うだけの性能向上は見られなかった．そこで，「中洲 1 号」ではベクトル・レジスタを設けず，以下のスカラ・レジスタのみを設けることにした.

- 汎用レジスタ：32 ビット × 32 個．読出しポート×8，書込みポート×3.
- 浮動小数点レジスタ：64 ビット × 32 個．読出しポート×6，書込みポート×4.

上記のようにポート数が多い場合 (汎用レジスタ・ファイルが 11 ポート，浮動小数点レジスタ・ファイルが 10 ポート)，レジスタ・ファイルの物理的にどう構成するかが問題となる．これには，

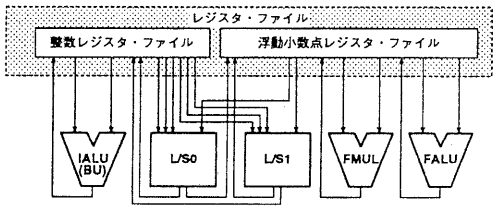


図 7: レジスタ・ファイル構成

- マルチポート・レジスタ構成
- マルチバンク・メモリ構成

の 2 つの選択肢が存在する．マルチバンク・メモリ構成の場合，バンク数<ポート数 とすれば，ハードウェア・コストはマルチポート・レジスタ構成より小さくなる．しかし，逆にバンク・コンフリクトが発生し，レジスタ・アクセス時間が一定にならないという短所がある.

「中洲 1 号」では，マルチバンク・メモリ構成の短所は許容できないことから，マルチポート・レジスタ構成を採用した.

4.5 チップ・フロアプラン

図 8に，「中洲 1 号」のチップ・フロアプランを示す.

配線ルール $0.8\mu m$，チップ面積 $15mm \times 15mm$ 程度を想定している．チップ内の各モジュールの面積は，現在の最先端の商用プロセッサ [9, 10, 11, 12] を基に見積もっている．ハイパースカラ・プロセッサは，同一スーパースカラ度および同一機能ユニット構成のスーパースカラ・プロセッサに比べて，少なくとも

- 命令レジスタ，および，
- レジスタ・ファイルに追加したポート

分だけハードウェア・コストが増加する．命令レジスタのメモリ容量は，「中洲 1 号」の場合 $5K$ ビット相当で，仮に命令キャッシュとデータ・キャッシュを合わせて $16K$ バイト (= $128K$ ビット) 備えるとすると，命令レジスタによるメモリ・コストの増加は約 4%程度で問題にならない.

一方，レジスタ・ファイルのポート数が増加すると，ほぼポート数の 2 乗に比例してレジスタ・ファイルの面積が増加するものと思われる．スーパースカラ度 2 で，整数演算と浮動小数点演算を並列実行可能なスーパースカラ・プロセッサの場合，汎用レジスタ・ファイルおよび浮動小数点レジスタ・ファイルのポート数はいずれも 3 程度である．「中洲 1 号」の場合，それぞれ 11 ポートおよび 10 ポートであるから，レジスタ・ファイル面積を 10 倍にし

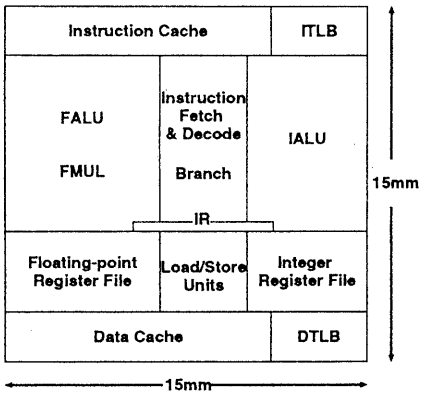


図 8: 「中洲 1 号」チップ・フロアプラン

たものではなく，*IWarp* のレジスタ・ファイル (15 ポート)[9, 10] の面積を基に見積もっている．

I/O パッドは，アドレス・バス (32 ビット × 3)，命令バス (32 ビット × 1)，および，データ・バス (64 ビット × 2) で少なくとも 256 個用いる．この他に，電源および制御信号のパッドが加わる．

5 おわりに

以上，現在開発中のハイパースカラ・プロセッサ「中洲 1 号」の命令セット・アーキテクチャ，および，ハードウェア構成について述べた．

現在，我々はハードウェア記述言語 SFL を用いたハードウェア記述および機能シミュレーションを行っている．今後は，VHDL あるいは VerilogHDL を用いたハードウェア記述，および，論理合成，そして，レイアウトへと進める予定である．

また，ハイパースカラ用最適化コンパイラも，同時進行で開発中である [3, 6, 7]．

謝辞

日頃から御討論頂く，九州大学 大学院総合理工学研究科 安浦寛人 教授，岩井原瑞穂 助手，および，広島市立大学 情報科学部 弘中哲夫 助教授に感謝致します．また，ハイパースカラ研究グループの斎藤靖彦 氏，白川暁 氏をはじめとする安浦研究室の諸氏に感謝致します．

参考文献

[1] 村上和彰, "ハイパースカラ・プロセッサ・アーキテクチャ — 命令レベル並列処理への第 5 のアプローチ —," 並列処理シンポジウム *JSPP'91* 論文集, pp.133–140, 1991 年 5 月.

[2] 斎藤靖彦, 村上和彰, "ハイパースカラ・プロセッサ・アーキテクチャ — 実現上の課題 —," 情処研報, ARC–101–12, 1993 年 8 月.

[3] 弘中哲夫, 斎藤靖彦, 村上和彰, "ハイパースカラ・プロセッサ・アーキテクチャ — ソフトウェア・パイプライニング処理に関する評価 —," 信学技報, VLD93–89, 1993 年 12 月.

[4] 村上和彰, "ハイパースカラ・プロセッサ・アーキテクチャ — Soft-Core Processor としての適応性 —," 信学技報, VLD93–96, 1993 年 12 月.

[5] 宮嶋浩志, 斎藤靖彦, 弘中哲夫, 村上和彰, "ハイパースカラ・プロセッサ・アーキテクチャ — ハイパフォーマンス・プロトタイプ・プロセッサの設計および予備性能評価 —," 情処研報,ARC–105–7, 1994 年 3 月.

[6] 弘中哲夫, 斎藤靖彦, 宮嶋浩志, 村上和彰,"ハイパースカラ・プロセッサ・アーキテクチャ — プロトタイプの設計および性能評価 —," 並列処理シンポジウム *JSPP'94* 論文集, pp.9–16, 1994 年 5 月.

[7] 斎藤靖彦, 村上和彰, "ハイパースカラ用最適化コンパイラの開発—ステージ・バランシングを用いたソフトウェア・パイプライニング—," 信学技報, CPSY–94–25, 1994 年 7 月.

[8] 村上和彰,"スーパースカラ・プロセッサの性能を最大限に引き出すコンパイラ技術", 日経エレクトロニクス, no.521, pp.165–185, 1991 年 3 月.

[9] Borkar, S., et al., "Supporting Systoic and Memory Communication in iWarp,"*Proc. 17th Ann Int'l. Symp. Computer Architecture*, IEEE CS Press, pp,70–81, Jun.1990.

[10] Peterson, C., Sutton, J., and Wiley, P., "iWarp : A 100-MOPS, LIW Microprocessor for Multicomputers,"*IEEE Micro*, vol.13, no.3, pp.26–87, Jun.1991.

[11] Alpert, D., and Avnon, D., "Architecture of the Pentium Microprocessor,"*IEEE Micro*, vol.13, no.3, pp.11–21, Jun.1993.

[12] McLellan, E., "The Alpha AXP Architecture and 21064 Processor,"*IEEE Micro*, vol.13, no.3, pp.36–47, Jun.1993.