計算機アーキテクチャ 110−12
設 計 自 動 化 73−12
（1995. 1. 20）

ハイパースカラ・プロセッサ『中洲1号』の開発環境および開発状況

宮嶋 浩志‡　白川 暁†　斎藤 靖彦‡　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

† 九州大学 工学部 情報工学科

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

あらまし

ハイパースカラ・プロセッサ・アーキテクチャは，現在主流となっているスーパースカラ方式の次世代プロセッサ・アーキテクチャの1アプローチとして筆者らが提案している手法である．現在，筆者らは本方式の有効性を示すことを目標にハイパースカラ・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持つ32ビット・マイクロプロセッサである．本稿では，現在開発中のハイパースカラ・プロセッサ『中洲1号』の概要，設計環境，および，その開発状況について述べている．

Development Environment and Progress of Hyperscalar Processor : NAKASU-1

Hiroshi MIYAJIMA‡　Akira SHIRAKAWA†
Yasuhiro SAITOH‡　Kazuaki MURAKAMI‡

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

† Department of Computer Science and Communication Engineering
Faculty of Engineering
Kyushu University

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

Abstract

We are proposing a post-superscalar processor architecture, called *hyperscalar processor architecture*. We have been developing a hyperscalar processor : NAKASU-1. This paper reportsthe environment and status of development of 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(*Hyper-scalar 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].

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 [3]. 現在，

- 「中洲 1 号」のハードウェア開発
- 最適化コンパイラの開発
- 性能評価

を同時進行で進めている.

本稿では，まず 2 章でハイパースカラ方式，および，現在開発中の「中洲 1 号」の概要に関して述べる. 次に，3 章で「中洲 1 号」の開発環境について紹介する. 4，5，および，6 章において，現在行っているハードウェアの開発状況，コンパイラの開発状況，および，性能評価に関して述べる. 7 章で今後の課題について述べ本稿のまとめとする.

2 「中洲 1 号」の概要

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.「中洲 1 号」では，命令長は 32 ビット固定長で命令フェッチ幅は 2 としている. 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.

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.「中洲 1 号」では並列動作可能な *FU* は 5 で，各 *FU* 当たりの *IR* 数は 32 である. よって，計 5 × 32 本の *IR* を装備する.

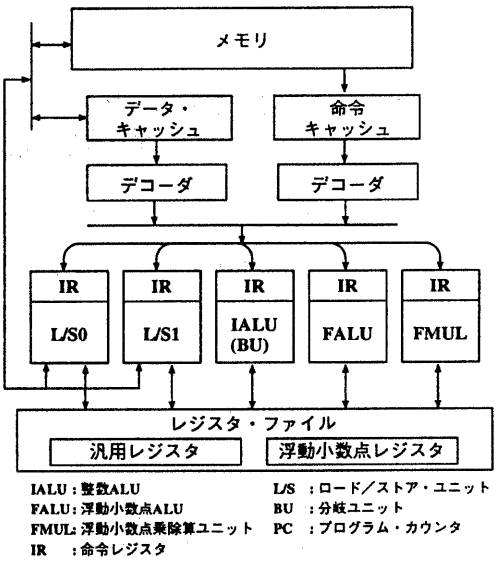

図 1:「中洲 1 号」の全体構成

IR が構成するアドレス空間は，5 × 32 の 2 次元配列となる. このとき，各行はあたかも，5 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 32 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 32 命令まで格納する.

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 (「中洲 1 号」の場合，スーパースカラ度：2，ハイパースカラ度：5 となる).

2.1 命令セット概要

「中洲 1 号」の命令セットは，32 ビット長 RISC プロセッサ *DLX*[9] 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 [3]. 命令形式は 32 ビット固定長である. 演算はすべてレジスタ-レジスタ間，あるいは，レジスタ-即値間のみで行う (ロード／ストア・アーキテクチャ).「中洲 1 号」の命令セットは，次のように分類できる.

- 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／ジャンプ命令は，通常モードと加速モードでは動作が異なる．ハイパースカラ方式特有の命令として，通常モードと加速モードとの間を遷移させる命令，および，*IR* を制御する命令が加わる．

- FLUSH(*FLUSH instruction registers*) 命令：すべての *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．すなわち，命令キャッシュからの命令フェッチを停止し，変わりに IRPC で指定された *IR* からの命令フェッチを開始する．通常モードで有効である．
- TQUIT(*Turbo QUIT*) 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 TFR(*True/False Register*) 値が 0 か否かを判定する．条件成立時は，指定した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．

「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

また，以下のような幾分ハイパースカラ特有の拡張を行っている．

- 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 [2].
- 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 [3].

2.2 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．
- 汎用レジスタ・ファイル：32 ビット長 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・ファイル：64 ビット長 32 個，読出しポート×6，書込みポート×4.
- ステータス・レジスタ：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：浮動小数点比較命令の結果を保持する．
- True/False レジスタ：1 ビット長 32 個．比較命令の結果を保持する．分岐は TFR 値に基づいて決定する．
- 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*.

2.3 命令パイプライン処理過程

図 2 に「中洲 1 号」のパイプライン構成の概略図を示す (文献 [3] におけるアーキテクチャから若干の変更をなしている)．

各ステージの動作は以下の通りである．

- 通常モード：4 ステージ構成
  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)．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．
- 加速モード：3 ステージ構成
  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)．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．
  2. EX：各 *FU*(BU を除く) における命令実行．
  3. 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)．

なお「中洲 1 号」では ID ステージにおいて，データ・ハザードを動的に解消する．

また，「中洲 1 号」では通常モードにおいて，BTB(*Branch Target Buffer*) を用いた分岐予測を行う．基本的に文献 [4] の分岐予測方法を採用している．IF ステージにおいて，命令をフェッチすると同時に対応する BTB のエントリの読出しを行う．エントリが有効であれば，フェッチしたブロック内の分岐命令を分岐成立と予測し，次サイクルの命令フェッチは BTB エントリに登録されていた分岐先アドレスを用いて行う．なお，分岐予測は静的に行う．加速モードでは分岐ペナルティが無いため分岐予測は行わない．

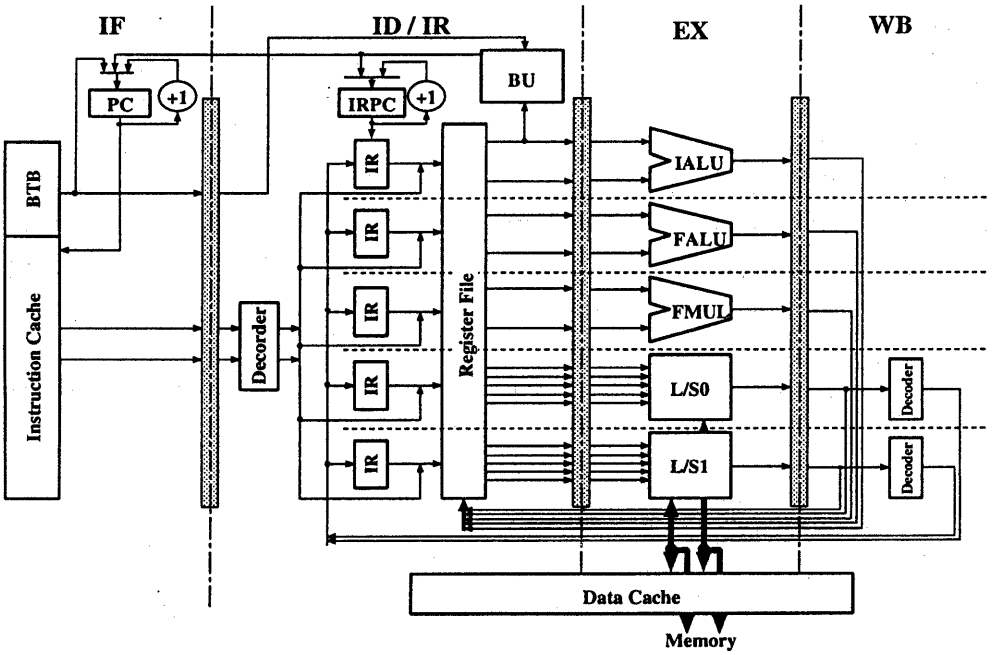

図 2: 「中洲 1 号」のパイプライン構成

2.4 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しており，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 である．

3 『中洲 1 号』の開発環境

現在，図 3に示すように，以下の面から「中洲 1 号」の開発を進めている．

- 「中洲 1 号」のハードウェア開発
- 最適化コンパイラの開発
- 性能評価

このうち，ハードウェア開発に関しては，以下の CAD ベンダのツールを我々は使用可能である．

- Mentor Graphics 社 (以下，Mentor)
- Cadence Design Systems 社 (以下，Cadence)

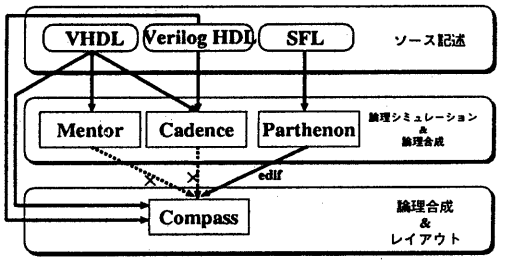

図 4: 使用可能な CAD の相関関係

- Compass Design Automation 社 (以下，Compass)
- NTT 社 (以下，Parthenon)

表 1に，これら我々の所有する CAD ツールで可能な開発工程を示す．表 1に示すように，レイアウトが可能なツールは Compass のみであるが，Compass の論理シミュレータを所有していないため，全開発工程を Compass のツールで行うことができない．また，現在，Mentor および Cadence は，ネットリスト・トランスレータを所有していないため，Compass へのネットリスト渡しができない．Parthenon から Compass へのネットリスト渡しは可能である．これら CAD ツール間の相関関係を図 4に示す．図 4中の矢印はソース記述からの選択可能な開発工程パスを示す．点線はネットリスト渡しが不可能であることを示す．

「中洲 1 号」のファブリケーション・メーカが未定であるため，使用するライブラリも現在未定である．

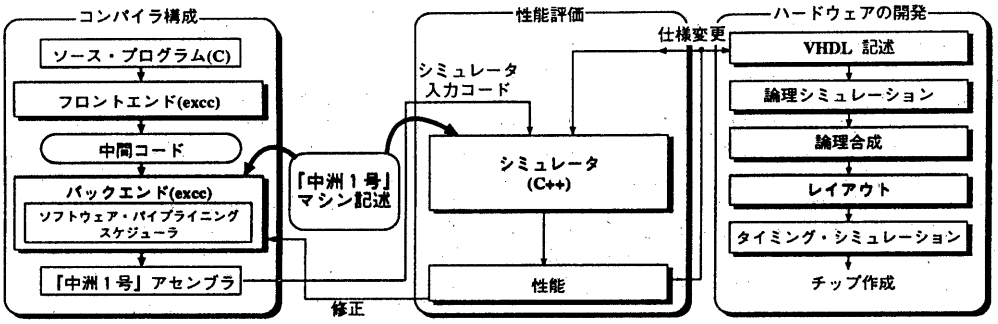

図 3: 「中洲 1 号」の開発工程

	Mentor	Cadence	PARTHENON	Compass
HDL	VHDL	Verilog & VHDL	SFL	Verilog & VHDL
論理シミュレーション	○	○	○	×
論理合成	○	○	○	○
レイアウト	×	×	×	○

表 1: 所有するツールで可能なハードウェア開発工程

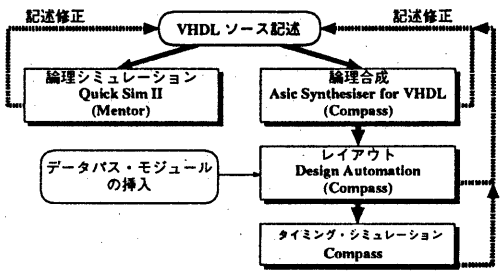

図 5: ハードウェア開発工程

4 ハードウェアの開発

図 5に示すように，Mentor の QuickSimII を用いて論理シミュレーションを行い，論理合成およびレイアウトは Compass のツールを用いて行う．

VHDL を入力言語とする合成系は，ツールによって論理合成可能な記述が若干異なり，HDL 記述を行う際には，ツール固有の記述上の制限を考慮して記述する必要がある．このため，予め Compass のツールで論理合成可能な制限内の記述法で記述し，その VHDL 記述を QuickSim II で論理シミュレーションしている．

4.1 VHDL 記述

図 6に「中洲 1 号」のチップ・フロアプラン [3] を示す．VHDL 記述は図 6に示すようにモジュール分割し，RT レベルで行っている．現時点で，論理合成の対象として記述を行っているのは，図 6中の斜線部分を除く部分，および，斜線部分中のメモリ・セル部分を除く部分である．斜線部中のメモリ・セル部分は RAM の挿入を予定している．レジスタ・ファイルおよび浮動小数点演算器に関しては 4.2節および 4.3節にて後述する．

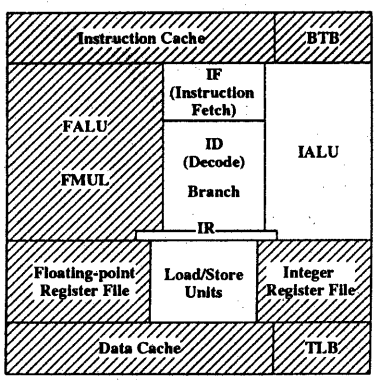

図 6: チップ・フロアプラン

現在，

- 命令レジスタ (IR)
- 整数レジスタ・ファイル
- 浮動小数点レジスタ・ファイル
- IF ステージ
- ID ステージ

の VHDL 記述を完了しており，論理シミュレーションを行うと共に，残りの部分の記述を行っている．今後，各モジュールごとに論理合成，レイアウトと進める予定である．

4.2 マルチポート・レジスタ・ファイル

「中洲 1 号」は，2.2節で示したようにマルチポート・レジスタ・ファイルを装備する．通常，レジスタ・ファイルはフルカスタムでレイアウト設計したものをマクロセルとして挿入する．HDL でレジスタ・ファイルを設計，論理合成した場合，レジスタ・ファイル・ポート

をマルチプレクサで制御するように回路を生成してしまう．それをスタンダード・セル方式でレイアウトすると，その面積はフルカスタム設計のレジスタ・ファイルの面積と比べ巨大なものとなる．

我々の研究室で開発した *QP-DLX*[5]，および，『中洲1号』のレジスタ・ファイルの，トランジスタ数およびレイアウト面積の比較を行った．論理合成ツールはParthenon，レイアウト・ツールはCompassを用いた．図2に比較結果を示す．*QP-DLX*は，32ビット汎用レジスタ32本の3ポート・レジスタ・ファイル(読出し×2，書込み×1)を持つ．比較モデルの設計工程は以下の通りである．

- *QP-SC*：*QP-DLX*のレジスタ・ファイルをSFLで設計および論理合成，Compassの Chip Compilerを用いてスタンダード・セル方式でレイアウト．
- *QP-DP*：*QP-DLX*のレジスタ・ファイルをCompassのデータパス・コンパイラ(繰り返しパターンからなるモジュールを生成するツール)で設計．
- *NAKASU-int-regfile*：『中洲1号』の整数レジスタ・ファイルをSFLで設計および論理合成．
- *NAKASU-fp-regfile*：『中洲1号』の浮動小数点レジスタ・ファイルをSFLで設計および論理合成．

*QP-SC*のレイアウト面積は，*QP-DP*の10倍以上である．現在最先端の商用プロセッサのチップ・サイズから，$15 \times 15mm^2$程度を想定して『中洲1号』の設計を行っている．しかし，『中洲1号』のレジスタ・ファイルよりもポート数が少ない*QP-SC*でさえ，その8分の1近くの面積を消費することになってしまう．今，我々の必要とするレジスタ・ファイル*NAKASU-int-regfile*，および，*NAKASU-fp-regfile*は，トランジスタ数にしてそれぞれ2倍，3倍以上であるので，*QP-SC*よりもはるかに大きなレイアウト面積を必要とする．このため，VHDLでレジスタ・ファイルを記述，論理合成，レイアウトする方針は，多大なハードウェア・コストを要するため選択できない．しかし，データパス・コンパイラは，我々の必要とするレジスタ・ファイルのポート数をサポートしていない(3ポートまでを許容)．

現在の我々の環境ではフルカスタムでマルチポート・レジスタ・ファイルを設計するのは不可能である．このため，レジスタ・ファイルはVHDLの記述は行っているが，実装時にはマクロセル・ライブラリを挿入する予定である．

4.3 機能ユニット

機能ユニットに関しても，前節にて述べたレジスタ・ファイルと同様の問題が発生する．HDLで設計およびレイアウトを行った機能ユニットは，フルカスタム設計された機能ユニットに比べ動作周波数が遅い上，レイアウト面積を多く必要とする．

また，整数ALUおよび整数MULは設計可能であるが，演算パイプライン化された浮動小数点演算器の開発は困難である．

このため，現在，機能ユニットのVHDL記述は行っているが論理合成の対象としていない．実装時にはマクロセル・ライブラリを挿入する予定である．

5 コンパイラの開発

ハイパースカラ・プロセッサは，実行するプログラムの命令レベル並列度の高い部分を命令レジスタ(*IR*)に格納し高並列度で実行する．*IR*に格納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．このとき，当該ループ部分の処理形態としては，擬似ベクトル処理およびソフトウェア・パイプライニング処理が可能である．擬似ベクトル処理の適用範囲がベクトル化可能なループに限られるのに対して，ソフトウェア・パイプライニング処理はベクトル化不可能なループにも適用可能である．つまり，ハイパースカラ・プロセッサにとってソフトウェア・パイプライニングは必要不可欠なコンパイラ技術と言える．

5.1 ソフトウェア・パイプライニング

ソフトウェア・パイプライニング・アルゴリズムでは，命令間の依存関係および使用可能な資源の数の制約を満足しつつ，ループのイタレーション開始間隔(*III*: *Iteration Initiation Interval*)を最小にすることを目指す．*III*の最小化を阻害する要因としては，命令間依存関係(データ依存関係および制御依存関係)および資源(*FU*およびレジスタ)競合が挙げられる．これらのうち，レジスタ値のライフタイムが長いことが原因でイタレーション間に生じる逆依存関係(一種のレジスタ競合)の影響が大きい．この逆依存関係を解消するため，従来からベクトル・レジスタ，および，モジュロ変数拡張といった手法が用いられている．しかし，ベクトル・レジスタはハードウェア・コストが増加するため，コスト/パフォーマンスの点から慎重を期す必要がある．また，ハイパースカラ・プロセッサでは，*IR*段数が限られていることから，コードサイズが大きくなるモジュロ変数拡張は必ずしも万全ではない．そこで，我々は，逆依存関係の解消の第3の手法として，ステージ・バランシング[2]を提案した．

図7に，ステージ・バランシングの概念図を示す．ステージ・バランシングとは，いま目標としている*III*が与えられている時，その*III*よりライフタイムの長いレジスタ値が存在する場合，当該レジスタ値を適宜異なるレジスタにコピーすることで，レジスタ値のライフタイムを常に*III*未満に保とうという手法である．

5.2 スケジューラの開発

図8に，予定している『中洲1号』コンパイラの構成を示す．現在，バックエンドにて行うコード・スケジューリングに関して研究を進めている．具体的には，5.1節で示した，逆依存解決手法ステージ・バランシングを用いたソフトウェア・パイプライニング・スケジューラを作成している．スケジューラの入力は『中洲1号』アセンブラ・コードである．プログラム中から抽出した最内ループ部分のオブジェクト・コードにソフトウェア・パイプライニングを施す．当面はスケジューラの出力を次節にて述べるシュミレータの入力コードとして使用する予定である．

コンパイラのフロントエンドおよびバックエンドには現在dlxccを用いている．dlxccは，gcc1.37.1にDLX[9]のマシン記述を入力したものである．このため，現状ではdlxccによって得られたオブジェクト・コードを

比較モデル	トランジスタ数	$Height(mm) \times Width(mm)$	面積 (mm^2)
QP-DP	—	1.169 × 2.107	2.463
QP-SC	70135	7.300 × 3.531	25.78
NAKASU-int-regfile	142020	—	—
NAKASU-fp-regfile	254968	—	—

表 2: レジスタ・ファイル面積比較

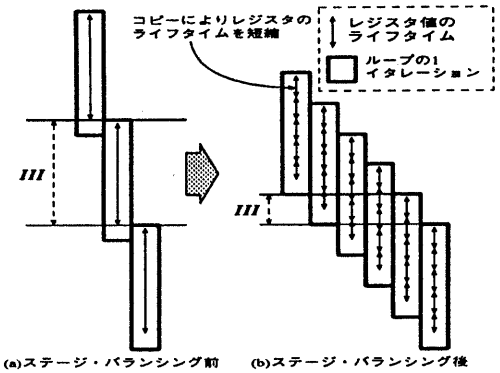

図 7: ステージ・バランシング

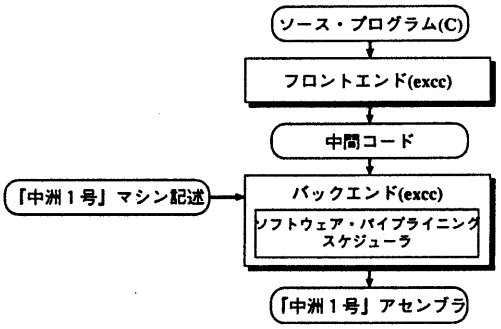

図 8: コンパイラ構成 (予定)

手作業で『中洲 1 号』オブジェクト・コードに変換している．また，gcc のバージョンが古いため，生成されたオブジェクト・コードの質が悪いといった問題も抱えている．そこで，コンパイラのフロントエンドおよびバックエンドに，電総研で開発が行われている並列計算機 EM-X 用 C コンパイラ excc[7] の採用を現在検討中である．先に説明したソフトウェア・パイプライニング・スケジューラのバックエンドへの接続も検討中である．

6 シミュレータの開発

6.1 予備性能評価結果

我々は今までにハイパースカラ・プロセッサの予備性能評価を行った [2]．予備性能評価はハイパースカラ方式と競合するアーキテクチャであるスーパースカラ方式，VLIW 方式，および，ベクトル処理方式の評価モデルを設定し，比較検討を行った．各評価モデルのオブジェクト・コードを gcc により生成し，その最内ループに対して，文献 [8] のモジュロ・スケジューリング・アルゴリズムに従って人手でソフトウェア・パイプライニングを施した．評価データとして，実行サイクル数および浮動小数点演算数を収集した．ベンチマーク・プログラムは *LFK*(*Livermore Fortran Kernels*)1〜14 を用い，評価指標として FLOPC(*floating-point operations per clock cycle*) を用いた．

その結果，ハイパースカラ・プロセッサ (スーパースカラ度 2，ハイパースカラ度 5) は，スーパースカラ・プロセッサ (スーパースカラ度 2) の 1.32 倍，従来型ベクトル・プロセッサの 1.09 倍，VLIW プロセッサ (スーパースカラ度 5) の 0.99 倍の性能が達成可能なことを確認した．我々はこの結果より，ハイパースカラ・プロセッサよりも大きな命令供給バンド幅，および，命令キャッシュを必要とする VLIW 方式よりもコスト・パフォーマンスが良いのではないかと考えている．

しかし，予備性能評価では以下の点に関して簡略化していた．

- メモリ・レイテンシを一定，かつ，小さなもの (3cycle) とした．
- キャッシュ容量を無限大とし，ヒット率 100%とした．
- ベンチマーク・プログラムは，*LFK*(*Livermore Fortran Kernels*)1〜14 のみを用いた．
- 演算器構成，命令発行多重度等，評価モデルのハードウェア構成を一定とした．

このため，予備性能評価では十分な性能評価を行ったとは言えない．よって，上記の項目を考慮に入れ，再評価を行う必要がある．

現在，C++を用いて性能評価用ソフトウェア・シミュレータを作成中である．設計中の VHDL 記述を用いて QuickSim II 上でのシミュレーションは検証目的以外行わない．行う予定の性能評価は，『中洲 1 号』のチューニング目的に留まらずハイパースカラ・プロセッサ自身の性能評価である．このため，シミュレータは以下のようなパラメータを指定可能とする．

- スーパースカラ度
- ハイパースカラ度
- 機能ユニット構成
- メモリ・アクセス・レイテンシ
- キャッシュの構成，容量，および，有無
- 命令レジスタ段数

ベンチマーク・プログラムは，一般的にマイクロプロセッサの性能評価に用いられている，

- LFK(*Livermore Fortran Kernels*)
- Linpack

- SPEC Benchmark

等を用いる予定である．

なお，シミュレータの入力コードは，5.2節にて述べた開発中のスケジューラの出力コードを用いる．また，競合するアーキテクチャとの比較を行うために，他アーキテクチャのシミュレーション・モデルも作成する予定である．

7 おわりに

以上，ハイパースカラ・プロセッサ『中洲1号』の開発環境および開発状況に関して述べた．現在，我々は『中洲1号』の開発として，

- ハードウェアの開発：VHDL記述および論理検証
- コンパイラの開発：ソフトウェア・パイプライニング・スケジューラの作成
- 性能評価：ソフトウェア・シミュレータの作成

を同時進行で進めている．今後，

- ハードウェアの開発：VHDL記述の論理合成およびレイアウト
- コンパイラの開発：フロントエンドおよびバックエンド部の検討
- 性能評価

へと作業を進めていく予定である．

ハイパースカラ方式は，現在主流であるスーパースカラ方式の次世代プロセッサ・アーキテクチャとしての1アプローチであり，そのアプローチは有効であると我々は考えている．現在，『中洲1号』のファウンダリ・メーカを探している．

謝辞

CADツールの利用に関して便宜を図って頂いているメンター・グラフィックス・ジャパン(株)，ケイデンス・デザイン・システムズ(株)，および，(株)ソリトン・システムズ，ならびに，CADツールParthenonを御提供頂いているNTTに感謝致します．

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 助手に感謝致します．また，シミュレータに関して御討論頂いた，CADグループの赤星博輝 氏をはじめとする安浦研究室の諸氏に感謝致します．

コンパイラに関して御討論頂いた，電総研の佐藤三久博士に感謝致します．

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] 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.

[3] 宮嶋浩志，村上和彰，"ハイパースカラ・プロセッサ『中洲1号』のアーキテクチャ," 情処研報，ARC–107–4, 1994年7月.

[4] 原哲也，久我守弘，村上和彰，富田眞治，"DSN型スーパースカラ・プロセッサ・プロトタイプの分岐パイプライン," 情処研報，ARC–86–3, 1991年1月.

[5] 中川智水，岩井原瑞穂，村上和彰，安浦寛人，"教育用32ビットマイクロプロセッサQP-DLXの設計におけるレイアウト," 情報処理学会第48回全国大会講演論文集(6)，pp.113–114, 1994年3月.

[6] 赤星 博輝，安浦寛人 "情報抽出技術を用いたアーキテクチャ評価用シミュレーション・モデルの生成," 情処研報，DA-??–?, 1995年1月.

[7] 佐藤三久，私信，1994年11月.

[8] Lam, M. S., *A Systolic Array Optimizing Compiler*, Kluwer Academic Publishers, pp.83–124, 1989.

[9] Hennessy, J. L., Patterson, D. A.,*Computer Architecture : A Quantitaive Approach*, Morgan Kaufman Publishers,Inc.,1990.