計算機アーキテクチャ　98－15
設　計　自　動　化　65－15
(1993. 1. 22)

アーキテクチャ評価用ワークベンチ
－コンパイラの自動生成－

赤星 博輝　安浦 寛人
九州大学大学院総合理工学研究科

アーキテクチャ評価用ワークベンチ (以後，ワークベンチ) は，論理合成を行うだけでなく，性能評価を行うことを目的としている．我々の性能評価の手法は，1) ハードウェア記述言語による設計記述から，コンパイラの自動生成を行い，実際のプログラムを用いた性能評価，2) 実際に論理合成を行い，そのハードウェアの性能評価を同時に行うことが特長である．

本論文では，スーパースカラ用のアーキテクチャ評価用ワークベンチのプロトタイプの実現について述べて，簡単な設計について適用した結果について報告する．

Evaluation Workbench for Superscalar Processor Architecture
– Compiler Generator–

Hiroki AKABOSHI and Hiroto YASUURA

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

E-mail:{akaboshi,yasuura}@is.kyushu-u.ac.jp

Evaluation Workbench for Superscalar Processor Architecture performs behavior synthesis and supports performance evaluation. There are two important ways of performance evaluation . 1) Evaluation Workbench generates a compiler from hardware description language(HDL) and simulates the design with compiled program by behavior level simulator. 2) Evaluation Workbench synthesizes the HDL description into VLSI circuit and simulates the design in the circuit level simulator.

This paper describes the compiler generator, and shows our evaluation methodology by a simple example.

1 はじめに

近年の半導体技術の進歩によって大規模な設計が可能となり，数百万ゲート規模の集積回路が実現されている．このように大規模な集積回路は，計算機の支援無くしては設計することはできない．現実に，ハードウェア設計に対する計算機による設計支援 (CAD:Computer Aided Design) 技術が進歩し，レイアウトや論理合成の自動化については，ほぼ完成された技術となっている．現在では，簡単な RISC チップなら数日で設計が完了するような，より高位からの CAD を利用することができる．しかし，次世代の設計支援技術として，何が必要とされているかという点は，はっきりしていない (図 1参照)．

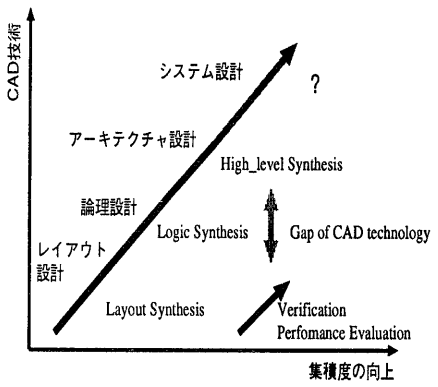

図 1: CADtechnology

次世代の設計支援技術を探るため，アーキテクチャ評価用ワークベンチ (以後ワークベンチ) の開発を行っている．これは設計の規模の増大また複雑化に対応するため，確立された下位の CAD 技術を組み合わせて，設計の初期の段階における精度の高い性能評価，検証，論理合成といった総合的な設計支援を行うことを最終目的としている．しかし，このような総合的な CAD に関しては，どのような要素技術が必要で，どのような問題点があるかまだ明らかでない．そこで我々は，並列処理システムの性能評価を行うアーキテクチャ評価用ワークベンチを開発し，これらの問題点を明らかにすることを目指している．

ワークベンチでは設計者はハードウェア記述言語 (HDL:Hardware Description Language) によって設計を行う．その記述からコンパイラに必要な情報を抽出し，コンパイラの自動生成を行う．生成したコンパイラを用いることで，実際に対象とする計算機システム上で使用するプログラムをコンパイルし，マシンコードを生成する．このコードを使って，より実際に近い条件でシミュレーションを行うことで，設計の早期からシステムの評価を精度良く行うことができる．また，論理合成ツールを使用することで，HDL から実チップを生成するためのデータを作成することができる．

本論文では，2 章でアーキテクチャ評価用ワークベンチについて述べて，3 章でコンパイラ・ジェネレータ，4 章で本手法の適用例を示す．

2 アーキテクチャ評価用ワークベンチ

2.1 目的

性能評価などを含めた総合的な CAD システムを構築する上で，問題点も明確ではなく，問題の定式化は，まだ行われていない．アーキテクチャ評価用ワークベンチの目的は，問題の定式化，および，性能評価を含めた総合的な CAD システムに必要な基礎技術の開発を行うことである．

我々は，以下の点が重要と考えている．

- 並列システム (ハードウェア/ソフトウェア) の記述 および 設計法
- アーキテクチャ設計早期段階における性能評価
- ハードウェア/ソフトウェアの協調設計 (Hardware/ Software CoDesign)

アーキテクチャ評価用ワークベンチは，実際の設計支援環境を構築を行うことで，これらの点について考察を行っていく．

2.2 ターゲットハードウェア

アーキテクチャ評価用ワークベンチのターゲットとするハードウェアは，汎用の並列処理システムである．現在作成しているアーキテクチャ評価用ワークベンチのプロトタイプは，ターゲットをスーパースカラ・プロセッサに限定している．これは，基礎技術を開発していく上で，ターゲットをある程度限定していく必要があると考えたから

である．スーパースカラ・プロセッサは，プログラムから複数の命令を同時にフェッチした後，命令解読，オペランドフェッチ，実行を行い，命令を並列に処理していくプロセッサであり，これをターゲットとした理由は，1プロセッサで並列処理が完結していること，また，並列処理システムの持つ開発の難しさを持っていると考えたからである．

2.3 性能評価

特定用途向けプロセッサでなく，汎用のプロセッサをターゲットとしている．汎用であるために，与えられた特定のプログラムの動きを解析して，ハードウェア/ソフトウェア双方を最適化することはできない．汎用プロセッサを，設計段階でどのように性能評価，そしてハードウェアの最適化を行っていくかが問題である．

実際に性能評価を行うためには，ベンチマークプログラムを利用する方法が現段階では最適と考えられる．しかし，性能評価に用いるベンチマークプログラムの数が少ないと，十分な性能評価ができない場合がある．性能評価の精度を上げるためには，より多くのベンチマークプログラムを実行させる必要がある．

今までのプロセッサと命令互換である場合は，既存のコンパイラが利用できるが，新しい命令セットやアーキテクチャを持ったプロセッサは，既存のコンパイラがない．そのため，プログラムをハンドコンパイルする必要があるために，評価に利用できるプログラムの数やプログラムの長さに制限があった．

現在，プロセッサの性能評価は，C言語などでシミュレータを作成し，その上で評価を行っている．そのために，ハードウェアの評価(クロック周波数やチップ面積)と，ベンチマークによるソフトウェアでの性能評価が別々に行われることが多い．

そこで，アーキテクチャ評価用ワークベンチでは，汎用プロセッサを設計するために，ハードウェアの評価とソフトウェアによる評価がともにできる環境を実現する．

2.4 構成

実際にアーキテクチャを設計していく上では，ハードウェア/ソフトウェアの情報は不可欠である．ワークベンチの構成は，図2のようになっており，ハードウェア/ソフトウェアに関しての情報を同時に生成できるようになっている．ハードウェアの評価としては，クロック周波数やチップ面積など，ソフトウェアによる情報としては，設計ハードウェア上でのプログラムの実行ステップ数，演算器の使用率などがあげられる．

ハードウェアの評価には，入力であるハードウェアの設計記述から，論理合成，そして，レイアウト合成を行って，論理シミュレーションを行う必要がある．それに対して，ソフトウェアによる評価に対しては，アーキテクチャレベルでシミュレーションを行う動作シミュレータ，ベンチマーク・プログラム，そして，ベンチマーク・プログラムをコンパイルするコンパイラが必要となる．論理合成ツール，レイアウト合成ツール，シミュレータに関しては，現段階でほぼ完成された技術となっている．これに対して，新しいアーキテクチャや命令セットを持った計算機では，コンパイラが無いために，シミュレーションに制約が生じる．そのために，ハードウェアに適したコンパイラを自動生成する技術によって，ハードウェアの評価と，ソフトウェアによる評価を同時に行うことが可能となる．

現段階では，ハードウェアの設計記述からコンパイラの自動生成についての研究を行っている．アーキテクチャ評価用ワークベンチでは，コンパイラ・ジェネレータによってコンパイラの自動生成を行っている．

2.5 性能評価用コンパイラ

アーキテクチャ評価用ワークベンチで生成するコンパイラは，性能評価を行うためのもので，実機の上で使用するものではない．これから，アーキテクチャ評価用ワークベンチで生成するコンパイラを，性能評価用コンパイラと呼ぶ．ここで，性能評価用コンパイラの要件としては，

- ハードウェア構成に応じて最適化を行い，プログラムをコンパイルすること．
- ハードウェア構成に応じた性能評価用コンパイラが，実際のコンパイラを作る場合より，容易でかつ短時間で作成できること．

が重要である．

今までは，さまざまな構成について，各々シミュレーションを行うことが難しかったが，それぞれについて性能評価用コンパイラを自動的に生成す

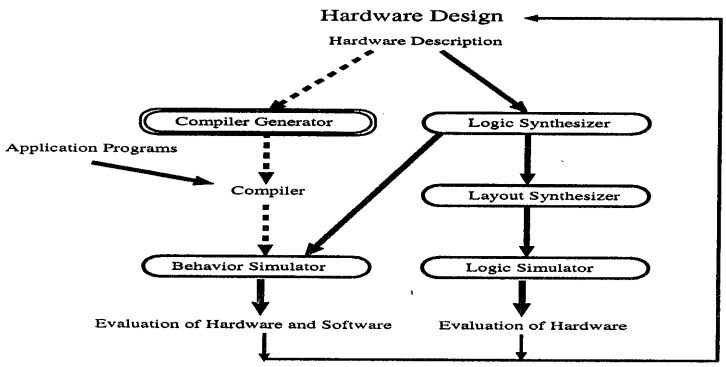

図 2: アーキテクチャ評価用ワークベンチ

ることでソフトウェアによる評価で問題となっていた，ベンチマークプログラムのコンパイルに関する問題が解決できる．

2.6 コンパイラの自動生成

今まで，特定用途向き (ASIC) プロセッサのためのソフトウェア開発環境，様々な構成の並列計算機に対しての並列プログラミング環境の構築の中で，いくつかのコンパイラ生成の研究が行われている．

- PEAS(Practical Environment for ASIP Development)[5]: PEAS というシステムは，ASIP(Application Specific Integrated Processor) を開発するためのシステムである．処理としては，ある応用プログラムの集合を解析し，その解析の結果から特定用途向け CPU のハードウェア構成，命令セットなどを決定する．この決定されたアーキテクチャ情報から，C コンパイラ，アセンブラ，シミュレータなどのソフトウエア開発環境の生成，CPU のコアを生成する．特に，ソフトウェア開発環境としては，アーキテクチャに制限を加えることで実現している．生成する CPU の制限としては，1) ハーバードアーキテクチャ,2) 命令長は 32 ビット,3)Load/Store アーキテクチャ,4)3 アドレス形式,5) 汎用レジスタ方式である．また，可変である項目は，1) レジスタの数，2) 命令セットの実現方法，3) アドレスバスの幅 である．特に，命令セットを，
  1. 絶対に必要でハードウェアで直接インプリメントされる原始ファンクショナリティ
  2. ハードウェアでも実現できるが，原始ファンクショナリティを組み合わせても実現できる基本ファンクショナリティ
  3. 実行頻度が高いライブラリ関数およびユーザ定義の関数を拡張ファンクショナリティ

  の 3 種類に分類し，それらの間で命令セットを自由に変更できる．
- **統合型並列化コンパイラ** [4]: 統合型並列化コンパイラの設計方針は，1)Multilingual Compiler 2)Multitargetable Compiler 3)Multilevel Optimizing Compiler 4)Multilevel Pararellizing Compiler の 4 つである．基本的に，コンパイラを，ターゲットに依存しない部分，依存はするがパラメータ化可能な部分，完全にターゲットに依存しパラメータ化不可能な部分の 3 つに分けて構成することでマルチターゲット・コンパイラを実現しようとしている．
- **スーパースカラ向けコンパイラ**: スーパースカラ・プロセッサの性能評価で用いているコンパイラは，スーパースカラでないパイプラインプロセッサと命令互換である場合では，既存のコンパイラの出力するコードに最適化を行うことによって実現している場合が多い [2, 3].

アーキテクチャ評価用ワークベンチでは，以下のような条件のハードウェアの性能評価用コンパイラを生成しようとしている．

- ハーバードアーキテクチャ
- 3アドレス方式
- Load/Store アーキテクチャ
- 汎用レジスタ方式

その上で，変更可能な項目は以下の通りである．

- 命令セット
- レジスタ数
- 演算器などのハードウェア構成

3 コンパイラ・ジェネレータ

性能評価用コンパイラを自動生成するコンパイラ・ジェネレータは，コンパイラ生成用データの抽出部，コンパイラ作成部の2モジュールで構成する．

コンパイラ生成用データの抽出部 ハードウェアの設計記述から，コンパイラを生成するために必要な情報を自動抽出する．現段階では，記述レベルはハードウェア記述言語による動作記述とし，実際の設計には，記述上の制約が必要となる．

コンパイラ生成部 コンパイラ生成用データの抽出部が抽出したデータから，性能評価用コンパイラを生成する．ターゲットとするアーキテクチャが，スーパースカラ・プロセッサであるために，実際に性能評価を行うためには，並列性を有効に抽出/利用する必要がある．そして，要件を満たすために，図3のような構成となる．

今回は，コンパイラ生成部について報告を行う．

3.1 コンパイラ生成部

ハードウェア記述から抽出されたデータから，性能評価用コンパイラを生成する．基本的な流れは，図4のようになっている．

3.1.1 広域最適化部

広域最適化部では，ループ変換やコード移動を行ってソースプログラムの中の並列に実行できる命令を増加させる．現段階では，広域最適化についてはインプリメントしていない．将来的には，ループ展開やソフトウェア・パイプライニング，パーコレーション・スケジューリングやトレース・スケジューリングなどのアルゴリズムについて考察し，実現する必要がある．広域最適化部に与えるデータとしては，並列に実行することができる演算器の数，および，レジスタの数などを与える．それらのパラメータとソースプログラムを入力として与え，広域最適化を行い，最適化後のプログラムを出力とする関数として実現する．

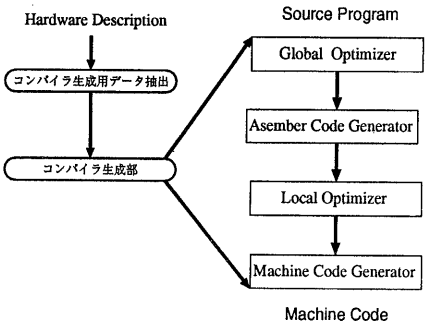

図 3: コンパイラ・ジェネレータ

3.1.2 アセンブラコード生成部

広域最適化を行ったソースプログラムからアセンブラコードを出力する．この部分ではGNU C(GCC)[8] を利用した．GCC を使って，ターゲットマシン用のコンパイラを生成するには，GCC の中間言語である RTL とターゲットマシンの対応の記述である “md” ファイルと，マシンの構造を記述するマシン記述マクロ (tm.h) を記述を行う必要がある．

アーキテクチャ評価用ワークベンチでは，命令セットの変更を，仮想的な命令セットを利用することで実現している．現段階で実際に持っている，仮想的な命令セットは，図5の通りである．設計者が実際の命令と仮想的な命令の対応を記述すると，それから，実際の命令セットによる “md” ファイルを生成する．これは，必ずしも1対1の置換えを行うのではなく，実際の命令を組み合わせて実現することも自動的に行うようにしている．

マシンの構造を記述するマシン記述マクロに関しては，現段階では，自動的に生成せずに，テンプレートを用意している．これによって，レジスタの数などの変更を行う．これは，レジスタの用

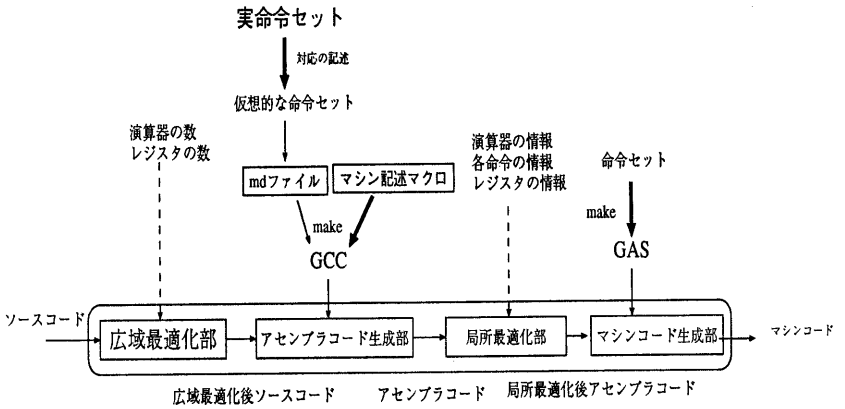

図 4: コンパイラ生成部

データ転送: LB, LBU, SB, LH, LHU, SH, LW, SW, LF, LD, SF, SD, MOVI2S, MOVS2I, MOVF, MOVD, MOVFP2I, MOVI2FP

整数演算: ADD, ADDI, ADDU, ADDUI, SUB, SUBI, SUBU, SUBUI, MULT, MULTU, DIV, DIVU, AND, ANDI, OR, ORI, XOR, XORI, NEG, LHI, SLL, SRL, SRA, SLLI, SRLI, SRAI, SLT, SGT, SLE, SGE, SEQ, SNE, SLTI, SGTI, SLEI, SGEI, SEQI, SNEI

コントロール: BEQZ, BNEZ, BFPT, BFPF, J, JR, JAL, JALR, TRAP, RFE

浮動小数点演算: ADDD, ADDF, SUBD, SUBF, MULTD, MULTF, DIVD, DIVF, CVTF2D, CVTF2I, CVTD2F, CVTD2I, CVTI2F, CVTI2D, LTD, GTD, LED, GED, EQD, NED, LTF, GTF, LEF, GEF, EQF, NEF

その他: NOP

図 5: サポートしている命令

途や関数呼び出しの方などで自動的には決定ができないものがあり，そのために，自動的な生成は行っていない．

この “md” ファイルとマシン記述マクロの 2 つを用意して，GCC を make してコンパイラの生成を行っている．

3.1.3 局所最適化部

局所最適化部では，出力されたアセンブラコードから，実行時間が短くなるよう命令の並び変えを行う．アルゴリズムとしては，リスト・スケジューリング [6] を利用している．この局所最適化部で，演算器などのハードウェア構成の変更に対応している．ハードウェアのパラメータを与えて，アセンブラコードを入力とし，実行時間を最短になるようにスケジューリングしたアセンブラを出力とする関数として実現している．

3.1.4 マシンコード生成部

最終的なマシンコードを生成するために，与えられた命令セットによるアセンブラを作成する．これは，GAS に命令セットなどを与えてコンパイルを行うことで，実現する．

3.2 コンパイラ生成用データ

評価用コンパイラを生成するために必要なデータを，コンパイラ生成のフェーズで分類すると，以下のようになる．

ソース言語からアセンブラ・コード 命令セットを変更するための “md” ファイルの生成の情報としては，図 6 である．このように，アセンブラで記述を行う．

マシン記述マクロを必要とするので，以下のデータを与える必要がある．今のところアーキテクチャ評価用ワークベンチでは，テンプレート方式にしており，これらの情報は人手によって与えている．今後は，この部分についても自動化を考えている．

- メモリに関する情報
- レジスタに関する情報

```
INST[LB ][ASM]="LDB";
INST[LBU ][ASM]="LDBU";
INST[SB ][ASM]="STB";
INST[LH ][ASM]="LDH";
INST[LHU ][ASM]="LDHU";
INST[SH ][ASM]="STH";
INST[LW ][ASM]="LDW ";
INST[SW ][ASM]="STW";
INST[LF ][ASM]="LDFS";
INST[LD ][ASM]="LDFD";
...
```

図 6: 中間言語とアセンブラコードの対応

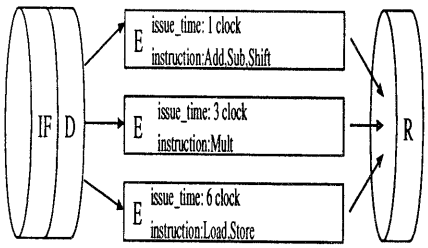

図 7: 最適化データ

- 関数呼び出しに関する情報

最適化 最適化は，現段階では局所最適化についてインプリメントが完了している．この段階で与えるデータは，各演算器毎に命令の発行間隔と，どの演算器でどの命令を実行するか，また，どの演算器を同時に実行できるかというデータである (図 7参照)．

マシンコード生成 マシンコード生成では，実際の命令コード表を与える．

4 アーキテクチャ設計例

今回は，ワークベンチでの設計の有効性を示すために，アーキテクチャ設計例として簡単なスーパースカラ・プロセッサを設計した．命令長は，32 ビット，ハーバードアーキテクチャ，3 オペランド方式，汎用レジスタ方式である．

アーキテクチャ評価用ワークベンチでは，実際に論理合成およびレイアウト合成を行うことで，実際のハードウェアについての情報を求める．また，実際に仕様を満すかという点については，実際のプログラムによるシミュレーションを行うことによって性能評価を行う．これらについて示す．

算術演算 ADD, SUB
データ転送 LD,LDI, ST
コントロール BEQZ, J
命令セット

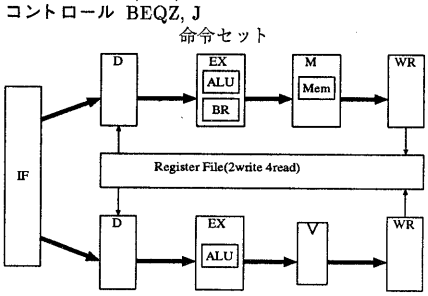

図 8: 設計したプロセッサ

表 1: ハードウェアの情報

	スーパースカラ	パイプライン
周波数(標準値)	32 MHz	30 MHz
ゲート数	45449	24678

スーパースカラ・プロセッサは ASIC Design System(Tsutsuji)[7] によって設計を行った．ハードウェアの設計結果としては，表 1の通りである．これらのデータは富士通の CG21 にマッピングした場合である．また，各部品毎に出力される遅延情報から，今後修正すべき点などを見つけることができる．

これと同時に，ハードウェアの情報とともに，実際のアプリケーションによる性能評価を行う．以下が，プログラム例である．

```
main() {
  int i,x;
  for  (i =1; i < N ; i++ )
      x = x +1;
}
```

表 2から，ALU を 1 つ増やしただけでは，ループ回数 10 の場合で性能の向上は 7.2%，ループ回数 2 の場合で 7.6%であった．それに対して，ハードウェア量としては，パイプラインプロセッサに比べて，84%増加している．

このように，プログラムの実際の動作ステップの情報，また，ハードウェア自体の情報というように，ハードウェア/ソフトウェアの双方の情報から，性能評価を行うことが有効であり，設計時のトレードオフについてのさまざまな評価が行うことが容易になる．

表 2: ソフトウェアからの情報

N	スーパースカラ	パイプライン
ループ回数10	137 Steps	147 Steps
ループ回数2	26 Steps	28 Steps

5 考察

5.1 コンパイラ・ジェネレータ

今回の評価では，設計したプロセッサの命令が少な過ぎたために，コンパイラを完全に生成することはできなかった．これは，今回作成したプロセッサの命令が少なくて，C言語のフルセットをコンパイルすることができないためである．しかし，現状のコンパイラ・ジェネレータでは，実際にはC言語のフルセットをコンパイル可能な場合でも，インプリメントの方法上からコンパイルが不可能な事が生じ得る．今後，基本的な命令の増加も含めて，インプリメントの方法についても，考える必要がある．

中間言語から，実マシンのアセンブラコードに変換する部分についての研究を行っている．

5.2 性能評価

今現在のアーキテクチャ評価用ワークベンチでは，動作シミュレーションや論理シミュレーションによって，ハードウェアの情報やソフトウェアによる情報をすべて出力してくる．今回の設計例は，比較的簡単であったため，必要な情報を容易に発見できた．複雑な設計で，ボトルネックの発見などを行う場合には，CADによって情報を加工して設計者に解りやすい形で出力する必要がある．

6 おわりに

実際にプロトタイプをアーキテクチャ評価用ワークベンチとして，コンパイラ生成部のインプリメントを行った．インプリメントを行うことで，いくつかの問題が新たな問題点も生じてきたため，今後，これらの問題点について解決をしていきたい．

性能評価の方法としては，ハードウェアとソフトウェアの両面から性能評価を行えるために，例として簡単な例ではあったが有効であると考えている．

今後，コンパイラを生成するためのデータを抽出する部分の実現を行うことで，コンパイラの完全な自動生成を行っていきたい．

謝辞

ASIC Design System(Tsutsuji) の提供など御協力いただく大鶴祥介所長をはじめとするYHPシステム技術研究所の皆様に深く感謝致します．日頃討論いただく九州大学大学院総合理工学研究科の村上和彰講師，および，安浦研究室の諸氏に感謝致します．

この研究は一部，京都高度技術研究所の新産学交流事業EAGLの支援による．

参考文献

[1] 赤星ほか: “スーパースカラアーキテクチャ評価用ワークベンチ,” DA シンポジウム'92論文集,pp.77-80,1992年8月．

[2] 岡本ほか: “SPARC アーキテクチャに基づくスーパースケーラマイクロプロセッサ“FLARE” の性能評価,” 情処研報，ARC-96-4,1992年10月．

[3] 中西ほか: “スーパスカラ・プロセッサ-SARCH- のコードスケジューラ,” 情処研報，ARC-96-5,1992年10月．

[4] 音成ほか: “統合型並列化コンパイラ・システム -概要，中間コードおよび解析手法-,” 情処研報，ARC-87-7,1991年1月．

[5] 博多ほか: “ASIC CPU 向きソフトウェア開発環境生成系の実現,” 信学技法，VLD92-25,pp.43-48,1992年．

[6] 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．

[7] “ASIC Design System Users's Manual,” 株式会社YHP システム技術研究所．

[8] Richard M.Stallman: “Using and Porting GNU CC for versin 1.37.1,” Free Software Foundation,INC, Feb. 1990.