ハードウェア機能の性能評価システムの開発

5V-7

藤原真二　柴山潔　萩原宏
京都大学　工学部

1．はじめに

ハードウェア機能を設計する過程は、1)方式設計、2)機能設計、3)論理設計、4)実装設計の４つの段階に分類できる。現在、設計したハードウェアを評価するためのシステムとしてシミュレータがあるが、多くのものは論理回路シミュレータであるため、3)の論理設計が完了しないと適用できない。我々は、1)方式設計、2)機能設計の段階において設計したハードウェア機能を評価するシステムが必要であると考え、その開発を進めている。

本稿では、トレース・マッピング法を用いた、方式・機能設計段階におけるハードウェア機能の性能評価システムの構成方式と性能評価方式について述べる。

2．トレース・マッピング法による性能評価方式

本システムは、設計の初期段階におけるハードウェア機能の性能評価を対象とする。すなわち、設計したハードウェア上で、適用対象となる応用プログラムを実行させたときのハードウェア機能の性能を評価するシステムである。

機能設計段階において、設計したハードウェア機能の性能評価を行う場合には、通常、その専用シミュレータを作成して応用プログラムをシミュレートし、性能評価データを得ている。機能設計段階においては機械命令セットがまだ決まっていないのが普通であり、ハードウェア機能も確定していない。したがって、ハードウェア機能の設計を変更する度に、専用シミュレータを変更する必要があった。そこで我々はトレース・マッピング法による、設計するハードウェア機能に依存しない性能評価システムの開発を進めている。

トレース・マッピング法とは、機械命令に依存しない高級言語で記述された応用プログラムの実行を、ハードウェア機能に依存しないＲＴ系列(Register Transfer系列)のトレース・データに変換し、設計したハードウェア機能のＲＴレベル・ブロック図上にマッピングすることにより性能を評価する方式である。本方式では評価対象の応用プログラムの実行時間やハードウェア機能の各ファシリティ（部品およびデータバス）の使用頻度などの性能評価を得ることができる。

本システムでは、応用プログラムを記述する高級言語としてLispを用いており、また、本システムそのものや各種ライブラリもLispを用いて記述している。Lispを採用した理由としては、一般の手続型言語の機能を含んでおり、プログラムの実行をインタプリトできるので実行結果のトレース・データが容易にとれること、各種ライブラリの記述が容易であることがあげられる。

3．システム構成

本システムの構成図を図１に示す。システムの入力となるものは、(A)評価対象のハードウェア上で実行したい応用プログラム、(B)評価対象となるハードウェアのＲＴレベル・ブロック図、(C)組み込み関数展開ライブラリ、(D)ハードウェア機能として用意している部品の動作定義ライブラリである。システム本体はRT-Tracer、RT-Optimizer、組み込み関数展開、ＲＴ系列マッピングの４つの部分で構成される。

3.1 RT-Tracer

RT-Tracerは、(A)の対象となる応用プログラムの実行過程をインタプリティブにトレースして、ＲＴ系列で記述されるトレース・データを生成する（図２参照）。一つのＲＴ系列は関数名、結果格納先リスト、オペランド・リストからなる。コンパイルされている関数はハード

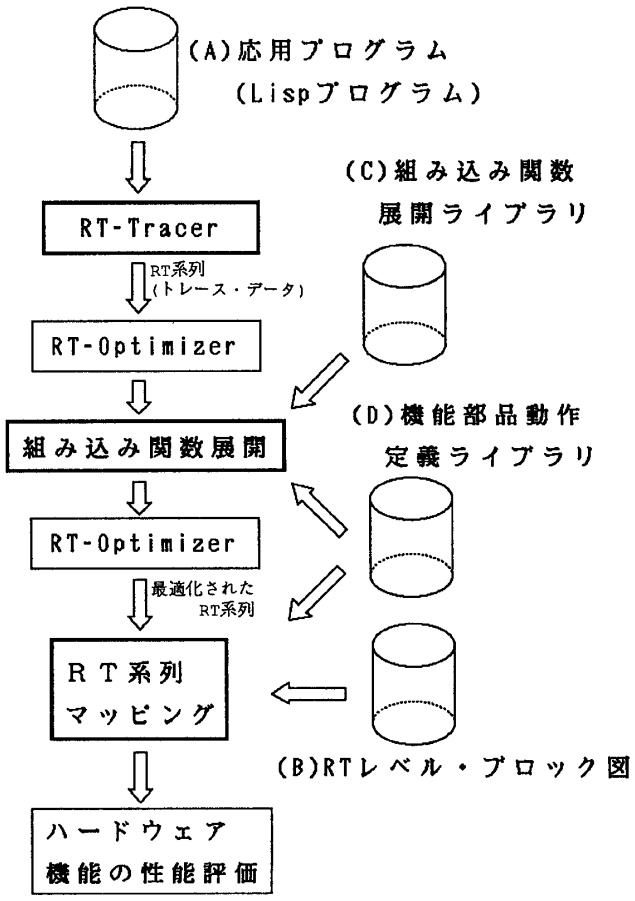

図１　システム構成図

Development of a Performance Evaluation System for designing Hardware Functions
Shinji FUJIWARA,Kiyoshi SHIBAYAMA,Hiroshi HAGIWARA
Kyoto University

```
;; (fact 3)の実行トレース
(ID (TMP49) 3)
(%CALL FACT X)
(ID (X-50) TMP49)
(ID (TMP52) 1)
(ID (TMP53) X-50)
(%CALL (LAMBDA (R XX)..))
(ID (R-54) TMP52)
(ID (XX-55) TMP53)
(ID (TMP58) XX-55)
(ID (TMP59) 1)
(< (TMP60) TMP58 TMP59)
(%IF TMP60)
(ID (TMP63) XX-55)
(ID (TMP64) R-54)
(* (TMP65) TMP63 TMP64)
(ID (R-54) TMP65)
    .....
```

```
;; 最適化されたＲＴ系列
(%CALL FACT X)
(ID (X-50) 3)
(%CALL (LAMBDA (R XX)..))
(ID (R-54)  1)
(ID (XX-55) X-50)
(< (TMP60) XX-55 1)
(%IF TMP60)
(* (R-54) XX-55 R-54)
    .....

;; factの定義
(defun fact (x)
  (do ((r 1)
       (xx x (- xx 1)))
      ((< xx 1) r)
      (setq r (* xx r))))
```

図２　ＲＴ系列の例

ウェア機能として実行可能であるとみなし、一つのＲＴ系列に変換する。したがって、高機能部品を使用する場合にはその機能をLisp関数で記述してコンパイルしておけばよい。

3.2 RT-Optimizer

RT-Optimizerは、生成されたＲＴ系列のトレース・データの冗長な部分を最適化する。ここではおもにパラメータを引き渡すためだけに生成された一時変数の消去を行う。

3.3 組み込み関数展開

組み込み関数展開では、Lispの組み込み関数の中で一つのハードウェア機能部品で実行できないものを、一つの機能部品で実行可能な関数（プリミティブ関数）に展開する働きをする。プリミティブ関数は、(B)のハードウェアのＲＴレベル・ブロック図と(D)の機能部品の動作定義ライブラリを参照することにより決定できる。

展開は、ＲＴ系列中に存在するすべての関数がプリミティブ関数になるまで(C)の組み込み関数展開ライブラリの関数定義にしたがって、再帰的に実行される。特殊な展開を要求する場合には、このライブラリに展開方法をLispの関数として定義する必要がある。

組み込み関数を展開したトレースは、再びRT-Optimizerにかけて、展開時に生成された冗長な部分を削除する。

3.4 ＲＴ系列マッピング

ＲＴ系列マッピングでは、最適化されたＲＴ系列のトレース・データを評価対象となるハードウェアのＲＴレベル・ブロック図（図３参照）上にマッピングする。まず、最適化されたトレース・データをシミュレートして変数の参照関係と型および値の解析を行う。この情報をもとにして、ＲＴ系列中の変数や定数を設計したハードウェア機能を構成するレジスタ、レジスタ・ファイル、メモリなどの部品に最適化しながらマッピングする。ここでは、設計したハードウェア機能の持つ並列処理能力を引き出すように、ＲＴ系列の同時実行可能性も考慮しながらマッピングをする。

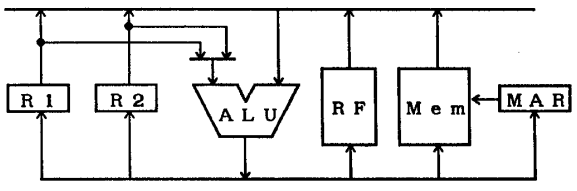

図３　ＲＴレベル・ブロック図の例

```
(defparts GR
  :class register
  :bit 16
  :optional-function
    (incf decf))

(defparts ALU
  :class functional
  :pin
   ((l (signed-byte 16))
    (r (signed-byte 16))
    (out (signed-byte 16)))
  :function
   ((+ l r) (- l r) r))
```

```
(defparts Mem
  :class memory
  :bit 16
  :size 4048
  :a-time 120)

(defparts RF
  :class register-file
  :bit 16
  :size 16)
```

図４　機能部品動作定義の例

4．機能部品動作定義ライブラリ

機能部品としては値を記憶するための記憶要素クラスの部品と各種演算をするための演算要素クラスの部品の動作をライブラリとして用意する（図４に例を示す）。

4.1 記憶要素クラス

このクラスの部品はその動作により、以下の３つのサブクラスに分類できる。

1)レジスタ：READとWRITEを１マシン・サイクルの間に実行できる記憶要素のプリミティブである。ただし、WRITEしてからREADはできない。

2)レジスタファイル：レジスタの集合である。１マシン・サイクルでREADまたはWRITEができる。同時にアクセスできるレジスタ数は通常１つである。レジスタのアドレスは制御回路から直接与えられる。

3)メモリ：大容量の記憶要素である。メモリのアドレスを外部の記憶要素から与える必要があるので、READやWRITEは１マシン・サイクル以上かかるのが普通である。データの入出力ピンの他にアドレスの入力ピンがある。配列やリストなどのデータはメモリにしか割り付けることができない。

4.2 演算要素クラス

各種演算を実行する機能を持つ部品のクラスである。演算の動作は副作用のないLispプログラムで記述する。関数の引数名が部品の入力ピン名となる。演算結果はOUTピンに出力される。また、各ピンの入出力値の型の宣言もできる。宣言がない場合にはいかなる型でも受け付けるものとする。

複数の機能を実現できる場合には各機能について同様に記述する。この場合、関数の引数を同じにするとピンを共有することができる。各演算にかかる時間は、引数の値を参照して記述することができる。

5．おわりに

現在開発中である、ハードウェアの機能設計段階における性能評価システムの構成と、性能評価方式について述べた。本システムはHP9000/370CH上のCommon Lispを用いて開発している。現在は特にＲＴ系列マッピングの方式について検討を進め、システムの実現に向けた研究を行っている。