4E-4

パイプラインプロセッサのマイクロ動作を対象とした形式的機能検証システムの試作

安部公章† 栗田雄一†† 北嶋暁† 武内良典† 今井正治†

† 大阪大学大学院基礎工学研究科情報数理系専攻 †† 奈良先端科学技術大学院大学情報科学研究科

1 はじめに

特定用途向けプロセッサの設計では，設計期間の短縮が重視されるため，設計したプロセッサが仕様を正しく実現していることを短期間で保証することが重要である．しかし，従来のRTレベルの設計においては，設計そのものが複雑になってきており，それに伴い，設計したプロセッサのHDL記述が各命令の意味を正しく実現していることを確かめる機能検証が複雑になってきている[1].

筆者らの所属する研究グループでは，特定用途向けプロセッサ設計システムPEAS-III[2]を用いて，抽象度の高い記述からプロセッサを自動生成する方法を研究している．プロセッサの各命令の動作をパイプラインステージごとのマイクロ動作として記述すれば，その記述からHDL記述を自動生成することができる．

本研究では，PEAS-IIIの入力記述であるマイクロ動作記述を対象とした形式的検証システムを試作した．検証システムでは，各命令の意味を，1命令実行したときの各レジスタ値の変化で表した記述に対し，マイクロ動作記述がその命令の意味を正しく実現していることを判定する．判定には，項書換えによる式変形や論理式の恒真性判定手続きを用いる．PEAS-IIIでのHDL記述の自動生成が正しく行われていると仮定すれば，マイクロ動作記述の正しさを保証することで，生成されるプロセッサのHDL記述が正しいことを保証したことになる．

本手法の有効性を，R3000互換プロセッサを対象とした実験により確認した．従来RTレベルのシミュレーションを行う時点でしか発見が困難であった誤りが，本システムにより，マイクロ動作記述を行った時点で発見できた．また，本システムにより，マイクロ動作記述が正しいことを保証できた．

Prototype Formal Verification System for Micro-Operation Description of Pipelined Processors

Masaaki Abe†, Yuichi Kurita††, Akira Kitajima†, Yoshinori Takeuchi†, and Masaharu Imai†

†Department of Informatics and Computer Science, Graduate School of Engineering Science, Osaka University

††Graduate School of Information Science, Nara Institute of Science and Technology

2 検証システムによるマイクロ動作記述の検証

本システムは，パイプラインプロセッサのマイクロ動作記述，命令の意味記述，基本関数対応表の3つを入力とし，与えられたマイクロ動作記述が命令の意味記述通りに実現されているかどうかを判定する．

2.1 命令の意味記述

各命令の1命令実行時の動作を表現した記述を新たに定義した．この記述を「命令の意味記述」と呼ぶ．具体的には，使用するレジスタ，命令タイプ，命令の動作を記述する．命令の動作は，1命令実行後のレジスタ値の変化の内容を代入文，または条件付の代入文を用いて記述する．また，演算は関数の形で表現する．

例として，加算命令ADDの意味記述を次に示す．

```
"ADD": "R1type" {
  PC  <- inc(PC);
  GPR <- reg_write(GPR, rd,
     add(read(GPR, rs), read(GPR, rt)));}
```

この例では，まず，ADD命令の命令タイプがR1typeであることを示しており，rs，rd，rtというオペランドが用いられる．ADD命令の実行により，プログラムカウンタPCは1増加し，レジスタファイルGPRはフィールドrsとrtで指定されたレジスタの値の和がrdに格納される．

2.2 マイクロ動作記述

マイクロ動作記述では，命令の動作を，各命令ごとに独立に，パイプラインステージごとに分割して記述する．各演算は，どの演算器を用いるかを指定して記述する．本稿では，記述に用いられるモジュールを「リソース」と呼ぶ．

加算命令ADDのマイクロ動作記述を次に示す．

```
"ADD" {
 clk(1) {"IR := IMEM[PC]; PC.inc();"},
 clk(2) {"DECODE(IR); $rs := GPR.read0(rs);
          $rt := GPR.read1(rt);"},
 clk(3) {"($result, $flag)
          := ALU0.add($rs, $rt);"},
 clk(4) {""},
 clk(5) {"GPR[rd] := $result;"}}
```

上の例において，$で始まる名前は変数を表している．

2.3 基本関数とリソースの機能との対応

命令の意味記述で使用する演算の関数と，マイクロ動作記述内で用いる各リソースの機能では，抽象度が異なる．このため，これらの間の対応が必要となる．

2.2節で述べたリソースは，FHM-DB[3]と呼ばれる設計データベースから得る．このデータベースから，データベース中のモジュールの機能とその基本関数の対応を得ることが可能である．

マイクロ動作記述で用いられる全リソースの各機能に対する基本関数表現を等式で表した記述を「基本関数対応表」と呼ぶ．

例として，基本関数対応表の一部を以下に示す．左辺がリソースの機能，右辺がその基本関数表現を表す．

```
PC.inc()               = inc(PC);
ALU0.add_OUT1($A, $B) = add($A, $B);
GPR.write($A, $B)      = reg_write(GPR, $A, $B);
GPR.read0($A)          = read(GPR, $A)
```

2.4 検証の内容

本システムでは，マイクロ動作記述が命令の意味記述で示された通りに動作するかを，各命令について，1命令実行による各レジスタ値の変化が両記述で任意の場合に等しいことを調べる．

3 検証アルゴリズムの概要

2章で述べたマイクロ動作記述，命令の意味記述，基本関数対応表から，マイクロ動作記述の検証を以下の手順で行う．

1. 各ステージのレジスタおよび変数に対する代入式を抜きだし，代入元を前ステージの式で置き換える．これを繰り返すことで，命令実行後の各レジスタの値を表す式を作る．
2. 前項で得られたレジスタ内容を表す式で使用されているリソース機能を，基本関数対応表に基づいて関数表現に変換する．
3. 前項で得られた式と意味記述で与えられた式が等価であるかどうかを，論理式の恒真性判定手続きを用いて判定する．

全命令について，各レジスタについての式が項3で等しいと判定されれば，マイクロ動作記述が命令の意味記述を正しく実現していると判定する．

4 評価実験

評価実験として，MIPS R3000準拠の命令セットアーキテクチャを実現したマイクロ動作が正しく記述されていることを検証した．この命令セットは48命令を持つ．パイプラインの段数は5段である．

本システムで検証可能な誤りを以下に示す．

- 0または1のビット定数，およびビット長の誤り
- フィールド名の誤り
- レジスタ名の誤り
- 書き込みや読み込みのポート指定などリソース機能の把握誤り

これらの誤りは，マイクロ動作記述の構文を調べるだけでは検出できず，従来は，HDL記述を生成する段階で検出したり，生成により得たHDL記述をシミュレータで確認することで検出していた．

実験では，本システムによりいくつか誤りが検出されたが，それらの誤りを訂正した結果，本システムにより，マイクロ動作記述が命令の意味記述を満たすと判定された．なお，検証に要したCPU時間は1回につき約1秒であった．

5 おわりに

本稿では，パイプラインプロセッサのマイクロ動作記述を対象とした形式的機能検証システムを試作した．評価実験としてMIPS R3000互換プロセッサのマイクロ動作記述を検証し，本手法の有効性を確認した．従来シミュレータを通さなければ検出できなかった誤りの検出が抽象度の高いマイクロ動作記述の段階で行えることが確認できた．

今後の課題として，PEAS-IIIで出力されたHDL記述が正しいことを保証するための，パイプラインハザードの検出法や割り込み動作の検証法を考案することが挙げられる．

参考文献

[1] Miroslav N. Velev and Randal E. Bryant. Formal verification of superscalar microprocessors with multicycle functional units, exceptions, and branch prediction. In *37th Design Automation Conference Proceedings 2000 (DAC 2000)*, pp. 112–117, Los Angeles, CA, June 2000. ACM.

[2] Makiko Itoh, Shigeaki Higaki, Jun Sato, Akichika Shiomi, Yoshinori Takeuchi, Akira Kitajima, and Masaharu Imai. PEAS-III: An ASIP design environment. In *International Conference on Computer Design (ICCD2000)*, September 2000. to appear.

[3] Masaharu Imai, Yoshinori Takeuchi, Takafumi Morifuji, and Eiichiro Shigehara. Flexible hardware model: A new paradigm for design reuse. In *APCHDL '98*, Seoul, Korea, July 1998. Invited Talk.