Vol.2013-SLDM-161 No.6
2013/5/16

マイクロアーキテクチャの CTL演繹体系を用いた検証

山田 雄二，富岡 涼太，高橋 隆一
広島市立大学　情報科学研究科　情報工学専攻

設計の大規模化や複雑化に伴い検証とテストが重要な課題となっている．本研究ではテストパターンに依存せずに網羅的な検証が行える形式的検証に注目した．形式的検証には定理証明型とモデル検査がある．定理証明型では設計が満たすべき定理を実際の設計が満たしている公理から導けるか調べることで検証を行う．定理と公理の記述は定理証明に用いる処理系の制約を受ける．モデル検査では設計をクリプキ構造で表し，設計が満たすべき性質を CTL で記述して検証を行う．状態数が多くなるという問題がある．本研究では定理と公理の両者を CTL 式で表し，CTL 演繹体系を用いて公理から定理を導くことによる検証を試みた．簡単な 4 段パイプライン・コンピュータを設計して，パイプライン化で問題になるハザードを検証対象とした．

Microarchitecture Verification using CTL Deduction system

Yuji Yamada，Ryota Tomioka and Ryuichi Takahashi
Faculty of Information Sciences，Hiroshima City University

Verification and test have become critically important as the scale of the design grows. This paper investigates the formal verification which could be done without using the test patterns. Theorem proving method and the model checking are two prior methods for the formal verification. In the theorem proving method, the verification is accomplished by trying to prove the theorem, which should be satisfied in the design, by using the axioms which represent the implementation. The description style for the theorems and the axioms depend on the theorem prover available. In the model checking, designs are expressed in the Kripke structure, which is used to verify the properties in CTL, which are expected to be satisfied on the structure. The number of the states tends to become too many. In this paper, we tried to express the design in CTL and tried to deduct the properties in CTL by using CTL deduction system. We investigated the hazards on a small scale pipelined computer having 4 stages.

1 はじめに

設計技術の進歩により，ディジタルシステムは大規模化，複雑化している．検証やテストにかかる時間が増大している．開発期間に多くの割合を占めるのは検証作業である．検証作業の効率化，設計品質の向上が重要な課題となっている．検証にはアサーションベース検証 [1]，[2] と形式的検証 [3] がある．本研究ではテストパターンに依存せずに網羅的な検証が行える形式的検証に注目した．形式的検証には定理証明型とモデル検査がある．定理証明型は設計を表す公理と，設計の満たすべき性質を表す定理を用いて，公理の下で定理が成り立っているかを定理証明支援システムによって証明する手法である．定理証明型で扱う対象は定理証明を行う処理系の制約を受ける．モデル検査は設計をクリプキ構造として表し，設計が満たすべき性質を CTL で記述し，クリプキ構造として表された設計において，CTL 式で表現された性質が成り立っているかの検証を行う．モデル検査ではモデルの状態数が多くなる傾向があり，実用にはなりにくい．本研究では検証に使う公理を CTL 式として抽出し，別途，設計の満たすべき定理も CTL 式で記述しておき，公理から定理が演繹できるかを CTL 演繹体系を用いて検証する手法を模索した．

検証対象は Verilog HDL[4] で設計した簡単な 4 段パイプライン・コンピュータとした．Verilog 記述から公理を CTL 式として抽出し，別途，定理を CTL 式で記述した．CTL 演繹体系を用いて公理から定理を導いて検証を行うものとした．

2 従来手法

機能や性能が仕様を満たしているかを調べることを検証という．検証にはシミュレーションによって妥当性検証を行うアサーションベース検証と，形式的仕様記述やプロパティなどを用いてシステムの妥当性を数学的に証明する形式的検証がある．

2.1 アサーションベース検証

アサーションベース検証 (assertion-based verification)，略して ABV では，アサーションで記述した性質をシミュレーションによって検証する．アサーションとは正しい設計として満たされるべき，または起こり得ない，性質などの記述である．設計内部の関心のある動作を検証対象とすることで観測性の向上が見込める．バグが起こり得る箇所にアサーションを書く必要がある．アサーションの記述が設計者依存になる場合は重大なバグを見落とす可能性がある．実行可能なすべての入力に対してシミュレーションを行うことは困難であるため限られた動作でしか検証できない．どの程度網羅できるかはテストベンチに依存する．

2.2 形式的検証

形式的検証では，仕様を形式的に記述し，設計が仕様を満たしているかを数学的に調べる．検証に成功すれば設計が仕様を完全に満たしているという結論が得られる．形式的検証には定理証明型とモデル検査がある．

定理証明型は，定理証明システムを用いて設計を検証する手法である．検証したい性質を定理として記述し，設計が満たしている公理を用いて検証する．定理証明型では定理証明を行う処理系の制約を受ける．

モデル検査とは検証対象であるモデル S において性質 p が成り立つかを判定する手法である．検証対象をクリプキ構造 S にモデル化し，様相論理 p で検証すべき性質を記述する．様相論理の論理式を解釈するための構造をクリプキ構造という．モデル検査の対象となるクリプキ構造 S=(W，R，V) においてWは可能世界の集合．R は到達可能性関係．V は命題記述に対する真偽の付値を意味している．クリプキ構造の例を図 1 に示す．

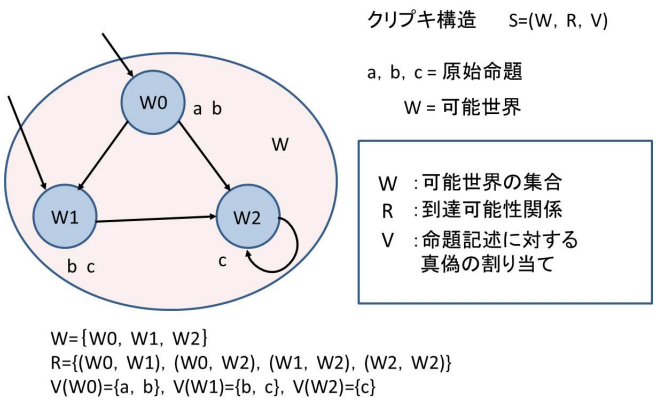

図 1: クリプキ構造

時相論理とは時間とともに状態を変化させるコンピュータの性質を記述するために用いられる様相論理である．与えられた状態からの全ての経路を扱う場合を分岐時間時相論理といい，計算木論理 (computation tree logic) が代表的である．CTL と略称される．

モデル検査ではクリプキ構造の構築と探索が大きな負担となる．実用化されている対象はバスプロトコルの検証などに限られている．

3　計算機論理 (CTL)

CTL 式は以下のように定義される．

- 原始命題は CTL 式である
- CTL 式に論理和 (∨)，論理積 (∧)，否定 (¬)，合意 (→) を用いた式も CTL 式である
- CTL 式である p，q に対し，EX p，EFp，EGp，E(pUq)，AXp，AFp，AGp，A(pUq) も CTL 式である

CTL 式は経路限定子Ｅ，Ａと時相演算子Ｘ，Ｕ，Ｆ，Ｇの組み合わせによって記述される．Ｅは「現在の状態から始まる経路が存在 (exists) すること」を意味し，Ａは「現在の状態から任意 (all) の経路に対して」を意味する．Ｘは「次（next）の状態」で，ｐ Ｕ ｑ は「ｑ が成り立つまで（untill）ｐが成り立つ」，Ｆは「経路上やがて（finally）」，Ｇは「経路上の至る所（globally）」を意味する．例えばＡＸ¬ ｐは「現在の状態から遷移可能なすべての状態でｐが成り立たない」を意味する．CTL 式は計算木上で真か偽の値をとる．

3.1　シーケント

$A_1, \cdots, A_m$ と $B_1, \cdots, B_n$ を論理式の並びとしたとき，下記をシーケント (sequent) という [5]．

$$A_1, \cdots, A_m \rightarrow B_1, \cdots, B_n$$

$A_1, \cdots, A_m$ を前提部，$B_1, \cdots, B_n$ を結論部といい，$A_1 \wedge \cdots \wedge \ A_m \Rightarrow \ B_1 \vee \cdots \vee \ B_n$ という論理式が対応する．$m = 0$ または $n = 0$ でもよく，$m = 0$ のときは $A_1 \wedge \cdots \wedge \ A_m = \top$，$n = 0$ のときは $B_1, \cdots, B_n = \bot$ とする．$m = n = 0$ のときは $\top \rightarrow \bot$ であって，$\bot$ と同値になる．

$A_1 \wedge \cdots \wedge \ A_m \Rightarrow \ B_1 \vee \cdots \vee \ B_n$ がトートロジであるとき，シーケント $A_1, \cdots, A_m \rightarrow B_1, \cdots, B_n$ もトートロジであるという．命題記号をどのように解釈しても $A_1, \cdots, A_m$ のどれかが偽になるか，$B_1, \cdots, B_n$ のどれかが真になるかであることを意味する．トートロジでないとは $A_1, \cdots, A_m$ のすべてが真に解釈され，$B_1, \cdots, B_n$ のすべてが偽に解釈されることを意味する．

前提部と結論部に同じ命題記号が現れるとき初期シーケントという．P,Q,R が命題記号であるとき P,Q,R → S,Q は初期シーケントになる．初期シーケントには一般の論理式を許してよい．初期シーケントはトートロジである．

3.2　命題論理演繹体系

証明に用いる命題論理の演繹体系を図 2 に例示する．

$$\frac{\Gamma \rightarrow \Delta}{\Gamma' \rightarrow \Delta'}\ (\text{並び替え}) \qquad \frac{\Gamma \rightarrow \Delta, \phi \quad \Gamma \rightarrow \Delta, \Psi}{\Gamma \rightarrow \Delta, \phi \wedge \Psi}\ (\wedge\text{右})$$

$$\frac{\Gamma \rightarrow \Sigma, \phi \quad \phi, \Theta \rightarrow \Delta}{\Gamma, \Theta \rightarrow \Sigma, \Delta}\ (\text{切断}) \qquad \frac{\phi, \Gamma \rightarrow \Delta, \Psi}{\Gamma \rightarrow \Delta, \phi \Rightarrow \Psi}\ (\Rightarrow\text{右})$$

$$\frac{\Gamma \rightarrow \Delta}{\Gamma, \Theta \rightarrow \Delta}\ (\text{W左})$$

図 2: 命題論理の演繹体系

図 2 において「W 左」は「weakening 左」を表している．

3.3 CTL 演繹体系

CTL の書き換え規則を図 3 に示す．

AXϕ ≡ ¬(EX¬ϕ)
EXϕ ≡ ¬(AX¬ϕ)
AFϕ ≡ A(trueUϕ)
EFϕ ≡ E(trueUϕ)
AGϕ ≡ ¬(EF¬ϕ)≡A(trueU¬ϕ)
EGϕ ≡ ¬(AF¬ϕ)≡E(trueU¬ϕ)

図 3: CTL 書き換え規則

$AX\ \phi$ は全ての経路に対して次の状態で ϕ が成り立つという意味となる．$\neg\ (EX \neg \phi\ )$ は「ある経路に対して次の状態で ϕ が成り立たないことはない」という意味となる．$AX\ \phi$ は $\neg\ (EX \neg \phi\ )$ と書き換えることができる．他の 5 つも同様である．

時相論理における CTL 演繹体系を図 4 に例示する．1 つ目の式は「現在の状態で $\Gamma \rightarrow \phi$ が真なら次の状態でも $\Gamma \rightarrow \phi$ が真となる」という意味となる．他の 4 つも同様である．

$$\frac{\Gamma\rightarrow\phi}{AX(\Gamma)\rightarrow AX(\phi)}\ (AX) \qquad \frac{\Gamma\rightarrow\Psi,\phi,\Delta \quad \Gamma\rightarrow\Psi,AXA(\phi U\Psi),\Delta}{\Gamma\rightarrow A(\phi U\Psi),\Delta}\ (AU右)$$

$$\frac{\Psi,\Gamma\rightarrow\Delta \quad \phi\ ,AXA(\phi U\Psi),\Gamma\rightarrow\Delta}{A(\phi U\Psi),\Gamma\rightarrow\Delta}\ (AU左)$$

$$\frac{\Gamma\rightarrow\Psi,\phi,\Delta \quad \Gamma\rightarrow\Psi,EXE(\phi U\Psi),\Delta}{\Gamma\rightarrow E(\phi U\Psi),\Delta}\ (EU右)$$

$$\frac{\Psi,\Gamma\rightarrow\Delta \quad \phi\ ,EXE(\phi U\Psi),\Gamma\rightarrow\Delta}{E(\phi U\Psi),\Gamma\rightarrow\Delta}\ (EU左)$$

図 4: CTL 演繹体系

図 4 に書かれていない EX ϕ，AF ϕ，EF ϕ，EG ϕ，AG ϕ 等の CTL 式も書き換え規則を用いた演繹により導出，証明ができる．

4 CTL 演繹体系を用いた検証

本研究では公理と定理を CTL 式で記述し，CTL 演繹体系を用いて公理⇒定理の証明を行う手法を模索した．CTL 式で記述可能な定理を扱うことができる．従来のモデル検査のようにクリプキ構造の状態遷移を追う必要もない．

4.1 検証対象

本研究での検証対象は簡単な 4 段パイプライン・コンピュータとした．図 5 に検証対象となるマイクロアーキテクチャのブロック図を示す．

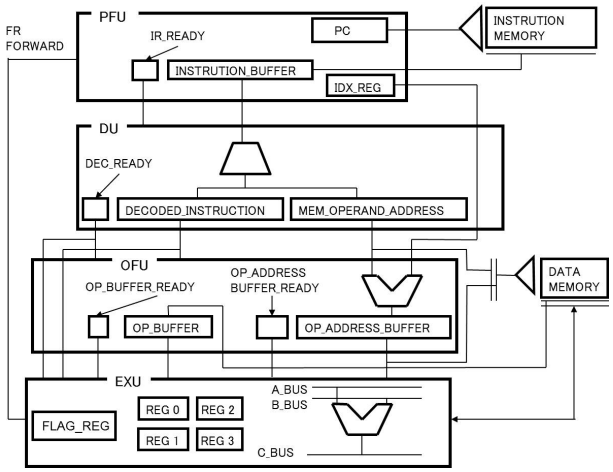

図 5: 検証対象のブロック図

設計したコンピュータは PFU，DU，OFU，EXU からなる 4 段パイプラインとなっている．条件付き分岐に関わるフラグレジスタの値のフォワーディングを実装している．制御ハザードが問題になる．

相対アドレッシングのためのインデックスレジスタを実装している．インデックスレジスタの値のセットはプリフェッチユニットで行われている．相対アドレスの計算はオペランドフェッチユニットで行われている．WAR ハザードが問題になる．

条件付き分岐の制御ハザードはインターロックとフォワーディングにより回避した．相対アドレッシングによるデータハザードはインターロックにより回避した．

4.2 CTL 演繹体系による検証

記述から抜き出した公理，満たすべき性質の定理の両者を CTL 式で記述し，公理⇒定理の検証を行った．

図 6 にフラグレジスタの値に関わる制御ハザードに関する公理を示す．図 6 の公理はフォワーディングとインターロックに関わる CTL 式を設計の記述から抜き出したものである．図 6 において##1 は SystemVerilog[1] での記述である．##m は m クロック後を表す．

```
公理
WAIT2→##1 WAIT1
WAIT1→##1FETCH
CPL, ##1 C-JMP→##2 WAIT2
CPL→(##3 FR_FWD = ##4FR)
```

図 6: 制御ハザードの公理

公理の抽出に用いた Verilog 記述の一部を図 7 に示す．

```
always @(posedge CLOCK1 or posedge RESET)
  begin
      if (RESET)
        FR = 2'b00;
    else
     begin
      if(FC_CPL)
       begin
            if (A_BUS >  B_BUS)  FR=2'b00;
        else if (A_BUS == B_BUS)  FR=2'b01;
        else if (A_BUS <  B_BUS)  FR=2'b10;
       end
     end

 always @(A_BUS or B_BUS or FLAG_REGISTER or
FLAG_FORWARD_R)
        if(FC_CPL)
        begin
            if (A_BUS >  B_BUS)  FLAG_FORWARD_R=2'b00;
        else if (A_BUS == B_BUS)  FLAG_FORWARD_R=2'b01;
        else if (A_BUS <  B_BUS)  FLAG_FORWARD_R=2'b10;
        else FLAG_FORWARD_R=FLAG_REGISTER;
       end
```

図 7: フラグレジスタに関わる Verilog 記述

図 6 の公理の 1 つ目は WAIT2 が立ったら次のクロックで WAIT1 が立つという記述である．2 つ目は WAIT1 が立ったら次のクロックで FETCH が立つという記述である．3 つ目は CPL が立って，次のクロックで C-JMP が立ったら，CPL が立った 2 クロック後に WAIT2 が立つという記述である．4 つ目は CPL が立ったら 3 クロック後の FR_FWD の値と 4 クロック後の FR の値が同じという記述である．WAIT や FETCH はインターロックに関わる．FR_FWD が実際にフォワーディングを行っている信号線である．

図 7 のフラグレジスタ，FR は実行ユニットに置かれ，クロックに同期して履歴を持つ．FR_FWD は組合せ回路として記述した．

図 8 に FR の値に関わる制御ハザードに関する定理を示す．

```
定理
CPL,##1 C-JMP→##4 FETCH∧(##3 FR_FWD=##4 FR)
```

図 8: 制御ハザードの定理

図 8 の定理は CPL が立ち，次のクロックで C-JMP が立ったら，4 クロック後に FETCH が立ち，3 クロック後の FR_FWD の値と 4 クロック後の FR の値が等しいことを表している．

図 9 に CTL 式で記述した公理から演繹体系を用いて定理を導いた例を示す．

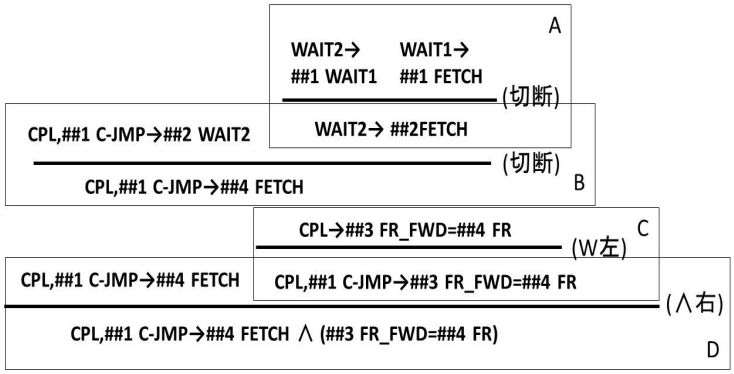

図 9: フラグレジスタに関する証明

図 9 の A では上段の 2 つの式より WAIT2 の 2 クロック後には FETCH が立っているということを導いている．同様に B では CPL の立った 1 クロック後に C-JMP が立ったら，CPL の立った 4 クロック後に FETCH が立つということを導いている．C では CPL が立った 1 クロック後に C-JMP が立ったら，CPL が立った 3 クロッ

ク後の FR_FWD の値と 4 クロック後の FR の値が同じになるということを導いている．D では CPL が立った 1 クロック後に C-JMP が立ったら，4 クロック後に FETCH が立って，3 クロック後の FR_FWD の値と 4 クロック後の FR の値が同じになることを導いている．最後に導出された式は図 8 の定理になっている．

CTL 演繹体系と命題論理演繹体系の規則を用いて公理⇒定理の証明を行うことで定理として表した性質の検証ができた．

5 まとめと今後の課題

本研究では設計が満たすべき定理と，実際の設計が満たしている公理の両者を CTL 式で記述し，CTL 演繹体系と命題論理演繹体系を用いることにより公理から定理が導けることを検証する手法を模索した．

証明に必要な公理の抽出方法の確立，証明に用いる CTL 演繹体系の拡充などが今後の課題である．

本研究は東京大学大規模集積システム設計教育研究センターを通しケイデンス株式会社の協力で行われたものである．

参考文献

[1] Ben Cohen， Srinivasan Venkataramanan，Ajeetha Kumari 著，三橋明城男，朽木順一，茂木幸夫，小笠原敦，明石貴昭　訳：SystemVerilog アサーション・ハンドブック，丸善 (2006)

[2] Harry D. Foster，Adam C. Krolnik，David J. Lacey 著，東野輝夫，岡野浩三，中田明夫　訳：アサーションベース設計　原書 2 版，丸善 (2004)

[3] 米田友洋，梶原誠司，土屋達弘，ディペンダブルシステム-高信頼システム実現のための耐故障・検証・テストの技術，共立出版（2005）.

[4] 高橋隆一 : Verilog HDL によるシステム開発と設計，共立出版 (2008)

[5] 萩谷昌己，西崎真也，論理と計算のしくみ，岩波書店，(2007)