Vol.2016-SLDM-176 No.6
2016/5/11

# SystemVerilogを用いた専用プロセッサのマトロイドによる検証

平川 昌和，高橋 隆一

広島市立大学　情報科学研究科　情報工学専攻

設計の大規模化に伴い検証とテストが重要になっている．1990 年代には論理合成技術が普及した．検証技術はこれに続く技術である．本研究では設計の広範囲な内容を一括して扱うために，実現しているアルゴリズムにおいて普遍的に成り立つ性質に注目する検証を模索した．クラスカルのアルゴリズムを実行できる 3 段パイプラインの専用プロセッサを設計し，枝を付加しても閉路が作られないというアサーションを構成した．閉路がないという性質はマトロイドとして満たすべき性質になっていた．木であることと連結で節点数は枝数より 1 多いという性質は同値であることに注目した．簡単なモデル検査で得られる性質から高度な性質を演繹することも考えられる．

# Verification of ASIPs on matroid by using SystemVerilog

Masakazu Hirakawa and Ryuichi Takahashi

Faculty of Information Sciences，Hiroshima City University

Verification and test have become important with increasing scale of the design. Logic synthesis spread in 1990 ’s. Verification is the next step. We investigated an assertion based verification considering a property generally held in the algorithm on the system. We designed an application specific instruction set processor for Kruskal’s algorithm. Nonexistence of loop was the property that should be held as matroid. To be a tree is equivalent to the property that the number of the vertexes is equals to the number of the edges plus 1 in a connected graph. A sophisticated property could be deducted from simple properties proved by model checking.

# 1 はじめに

設計の大規模化に伴い検証とテストが重要な課題になっている．1990 年代にはハードウェア記述言語（HDL）によって仕様を記述し，回路は論理合成で実現する設計スタイルが普及した．検証技術はこれに続く技術である．HDL としては 1980 年代すでに業界標準として広く用いられ，1995 年に IEEE1364 として標準化された Verilog HDL[1] が代表的である．SystemVerilog[2] は Verilog HDL の完全上位互換の言語であり，アサーションを書くことができる．アサーションとは設計が満たすべき性質の記述である．

本研究では設計チームと検証チームが別箇に存在する開発体制を想定している．検証チームは仕様書をもとに網羅的な検証項目を準備して待機する．検証はアサーションベース検証 (ABV)[2]，[3] によって行われる．検証チームに渡す設計の品質を高めるために，設計チームも ABV を使うことが考えられる．設計チームが用いるアサーションは広範囲な内容を一括して対象と出来れば効率が良い．本研究では対象とするアルゴリズムにおいて普遍的に成り立っている性質に注目する検証を模索した．

クラスカルのアルゴリズムを実行できる 3 段パイプラインの専用プロセッサを設計した．枝を付け加える命令を実装している．枝を付加しても閉路が作られないというアサーションを構成した．連結で閉路がないという性質がマトロイド [4] として満たすべき性質になっていた．木であることと連結で節点数は枝数より 1 多いという性質は同値であることに注目した．

ABV の他に形式的検証 [5] がある．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．モデル検査ではモデル上で設計対象が満たすべき性質が満たされるかどうかを数学的に検証する．性質は様相論理によって記述される．モデル検査ではモデルの状態数が多くなる傾向がある．本研究では，様相論理で記述された性質を比較的容易なモデル検査で得られる性質から演繹することを考えた．

# 2 設計検証

設計で得られた機能や性能が仕様を満たしているか調べることを設計検証という．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，形式的検証がある．

## 2.1 アサーションベース検証（ABV）

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．アサーションとは設計で満たされるべき，または起こってはならない性質の記述である．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．ABV によって設計品質の向上と設計期間の短縮が期待される．図 1 に ABV を用いることによる設計品質の向上の概念図 [6] を示す．同じ品質であれば ABV を用いた方が短い期間で得られる．同じ期間であれば ABV を用いた方が高い品質が得られる．

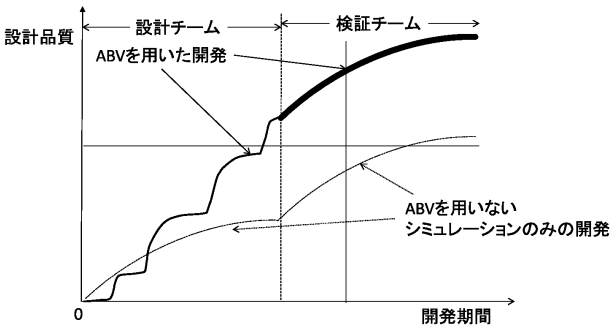


図 1: ABV を用いることによる設計品質の向上

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 [5].

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．様相論理は可能世界においてどの命題が成り立つかを扱う論理である．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．

## 3 設計した ASIP

特定の応用分野に適した命令セットを持つプロセッサを application specific instruction set processor，略して ASIP という．

節点の集合 V と枝の集合 E からなるグラフ G(V,E) において，すべての節点を含む部分グラフ S(V,T) が木であるなら， S(V,T) のことをグラフ G(V,E) の全域木であるという．各枝に重みがある場合，重みの総和が最大となる全域木を最大全域木という．重みの総和が最小となる全域木を最小全域木という．本研究では最大全域木あるいは最小全域木を求める ASIP を設計した．最大全域木を求めるためクラスカルのアルゴリズムを用いた．

### 3.1 クラスカルのアルゴリズム

クラスカルのアルゴリズム [7] は一般には貪欲法と呼ばれる手法になっている．以下にクラスカルのアルゴリズムの手順を示す．ここでは最大全域木を求めるアルゴリズムを示す．$\{e_i\}$ を枝の集合，N をノード数とする．$\omega(e_i)$ によって枝 $e_i$ の重みを表す．

Step 1 枝を重みの降順に並べる:$\omega(e_1) \geqq \omega(e_2) \geqq \cdots \omega(e_m)$

Step 2 i=1，辺の集合 X=$\emptyset$ とする．

Step 3 X $\cup$ $\{e_i\}$ が閉路を含まないなら X $\leftarrow$ X $\cup$ $\{e_i\}$ とする．

Step 4 $|X|$=N-1 なら X が定める木を全域木として終了する．

Step 5 i $\leftarrow$ i+1 として Step3 へ

### 3.2 全域木専用プロセッサ

図 2 に本研究で設計した 3 段パイプラインの ASIP のブロック図を示す．

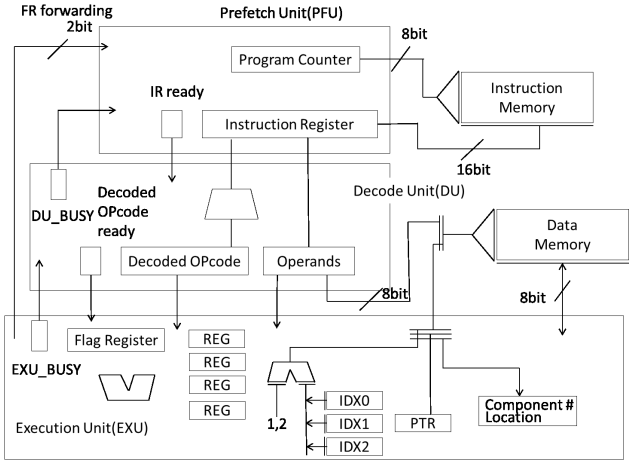


図 2: ブロック図

プリフェッチユニット（PFU），デコードユニット（DU），実行ユニット（EXU）からなる 3 段パイプラインとなっている．クロックは単相である．フラグのフォワーディングを行っている．クラスカルのアルゴリズムを実行するため枝を付け加える命令を実装している．静的オートマトン記述（SSMD）[1] で実現している．ここにオートマトン記述は各状態で行われるレジスタ間転送と状態遷移を状態毎に列挙した記述を意味する．SSMD は Verilog HDL によるオートマトン記述である．

クラスカルのアルゴリズムを実行するため，Union-Find[7] を実装している．Union-Find を用いることで，枝の追加を行う際に閉路ができるかの判定を行っている．ノードが連結しているならば同じグループに属している．ノードが

非連結ならば異なるグループに属している．枝を追加する際に，枝の両端のノードが異なるグループならば閉路は出来ない．

設計したプロセッサはハーバードアーキテクチャである．命令とデータで物理的に異なる記憶装置を用いている．完全なインターロックを搭載している．インターロックとはハザードを検出して回避する機構である [1].

## 4 マトロイド

貪欲法によって最適解が得られる根拠はマトロイドにある．

### 4.1 マトロイドの定義

有限集合 S とそのべき集合の部分集合 I$\subseteq 2^S$ が与えられていて

$$\emptyset \in I \tag{1}$$

$$X_1 \subseteq X_2, X_2 \in I \Rightarrow X_1 \in I \tag{2}$$

$$X_1, X_2 \in I, |X_1| < |X_2|$$

$$\Rightarrow \exists e \in X_2 - X_1 \quad [X_1 \cup \{e\} \in I] \tag{3}$$

が成り立つとき S と I の対 M(S，I) をマトロイドという．クラスカルのアルゴリズムは枝の集合に閉路がないという性質がマトロイドであることに注目した貪欲法になっている．

### 4.2 全域木の満たす性質

連結で閉路がないグラフ G を木という．クラスカルのアルゴリズムにおいては枝を追加しても閉路が作られないという性質を満たしている必要がある．

グラフ G に関して次の条件は同値である．

(i) G は木である．

(ii) G は連結であってどの枝を除いても非連結になる．

(iii) G のどの 2 つのノード間にも基本道が 1 つだけ存在する．

(iv) G は連結でノードの数が枝の数より 1 大きい．

枝を追加する命令に対応させて，各連結成分において常にノードの数が枝の数より 1 大きいという条件でアサーションを記述した．

## 5 ABV による検証

設計チームが用いるアサーションは広範囲な内容を一括して対象と出来れば効率が良い．本研究では普遍的に成り立つ性質としてマトロイドが満たすべき性質に注目した．常にノードの数が枝の数より 1 大きいという条件を用いてアサーションを構成した．枝を追加しても常に閉路がないということを確認できるアサーションとなっている．図 3 に本研究で用いたアサーションを例示する．

```
always@(posedge CLOCK1)
  begin
    for(a=0,NODE=IDX3,V1=0;a<n;a=a+1,NODE=NODE+4)
      begin
        if(MEM[NODE+1]==1) V=V+1;
end
  end
always@(posedge CLOCK1)
    begin
        for(b=0,EDGE=IDX2,E1=0;b<m;b=b+1,EDGE=EDGE+5)
          begin
            if(MEM[EDGE+3]==1) E=E+1;
end
  end

core_memory1: assert property
    (
 @(posedge CLOCK1) ADD_EDGE && V && E |=>  (V==E+1));
```

図 3: アサーション

図 3 は枝を追加した時には，常にノードの数が枝の数より 1 大きいことを確認するアサーションの例である．連結であるノードと枝の数を数えて，V と E に加算している．SystemVerilog は Verilog HDL の完全上位互換な言語であり，加算のループも記述できる．

本研究ではデータ構造にリンクリストを用いている．アサーションの記述内の NODE と EDGE は，それぞれ連結しているノードと枝のリンクリストの先頭を指している．アサーションの記述内の NODE+4 と EDGE+5 は次のノード，次の枝へのポインタとなっている．常に成立していることが確認できた．枝を誤って追加し，閉路ができてしまう場合にアサーションは不成立となることも確認できた．

## 6 モデル検査と演繹

形式的検証においては満たすべき性質を時相論理で記述する．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．

### 6.1 CTL と LTL

与えられた状態からのすべての経路を扱う場合を分岐時間時相論理という．計算木論理 (computation tree logic) が代表的である．CTL と略称される．

CTL の論理式は，複数の経路や経路の間の関係を記述することができる．BNF 記法により以下のように定義される [8].

$$A ::= P|\neg A|A \vee A$$
$$\mathbf{EX}A|\mathbf{E}(A\mathbf{U}A)|\mathbf{EF}A|\mathbf{EG}A$$
$$\mathbf{AX}A|\mathbf{A}(A\mathbf{U}A)|\mathbf{AF}A|\mathbf{AG}A$$

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．

線形時相論理（Linear Temporal Logic）を略して，LTL という．LTL は個々の経路を扱う．LTL には経路限定子 **E**，**A** は表れない．LTL 式は時相演算子と原子命題で表される．例えば **FG** p は「経路上やがて経路上のいたるところで p が成り立つ」を意味する．

### 6.2 時相論理の式を演繹することによる検証

図 4 にゲンツェン流の自然演繹 [9] の例を示す．A は運動会が行われたことを表し，B は朝に雨が降っていれば運動会は行われないことを表している．⊥は矛盾を表す．

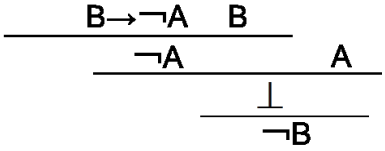


図 4: ゲンツェンの自然演繹の例

運動会が行われたなら朝に雨は降っていなかったことが導かれている．導出図が存在することを表す記号⊢ を用いると

$$A \vdash \neg B$$

と表される．

筆者らは通常の論理式ではなく，時相論理の式を演繹して検証に用いている [10].

図 5 に時相論理式を対象とする推論規則の例 [11] を示す．図 5 で f は連結で $|V| = |E| + 1$ であることを表し，g は連結で閉路がないことを表している．V は節点の集合，E は枝の集合である．

$\vdash f \Rightarrow g$ は連結で $|V| = |E| + 1$ ならば連結で閉路がないという定理を表している．$\vdash f \Rightarrow Xf$ は f が成り立つならば次の状態でも f が成り立つという意味である．$\vdash f \Rightarrow Xf$ はモデル検査によって証明される性質とする．

$$\frac{\vdash f \Rightarrow g \qquad \vdash f \Rightarrow Xf}{\vdash f \Rightarrow Gg}$$

図 5: 推論規則

図 5 の推論規則によって，連結で $|V| = |E|+1$ であることもモデル検査によって証明されていれば，いたるところで連結で閉路がないことが導出される．

## 7 まとめと今後の課題

設計チームが広範囲な内容を一括して扱うために，設計している専用プロセッサが実現するアルゴリズムで普遍的に成り立っている性質に

注目したアサーションを用いることを提案した．クラスカルのアルゴリズムを実現する専用プロセッサでは連結で閉路を持たないという性質がマトロイドとして満たすべき性質になっていた．証明やモデル検査によって比較的容易に得られる性質を前提とすることで，満たされるべき性質を表す時相論理の式を推論によって導出することによって形式的検証が行える可能性を示した．時相論理の式を対象とする推論規則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] 高橋隆一 : Verilog HDL によるシステム開発と設計，共立出版 (2008)

[2] 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)

[3] 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)

[4] 上野修一，高橋篤司 : 情報とアルゴリズム，森北出版株式会社（2005）.

[5] 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）

[6] 高橋隆一 : 電子機器設計に活かすディジタル代数学，科学情報出版 (2013)

[7] 平田富夫 : C によるアルゴリズムとデータ構造，科学技術出版（2002）

[8] 萩谷昌己，西崎真也 : 論理と計算のしくみ，岩波書店（2007）

[9] 鹿島亮 : 数理論理学，朝倉書店 (2009)

[10] 山田雄二，富岡涼太，高橋隆一 :“ マイクロアーキテクチャの CTL 演繹体系を用いた検証, ” 情処研報 Vol.2013-SLDM-161 No.6，信学技報 Vol.113, No.30, pp.31-36(2013).

[11] Ming-Hsien Tsai，Bow-yaw Wang :” Formalization of CTL* in Calculus of Inductive Constructions，”ASIAN 2006，LNCS 4435，pp．316-330（2007）