社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2009-SLDM-138 (20)
2009 / 1 / 30

仕様から自動生成されたプロパティによるプロトコル変換機の形式的検証手法

高　飛[†]　西原　佑[†]　松本　剛史[††]　藤田　昌宏[††]

† 東京大学大学院工学系研究科電子工学専攻　〒113–8656 東京都文京区本郷 7–3–1
†† 東京大学大規模集積システム設計教育研究センター　〒113–0032 東京都文京区弥生 2–11–16
E-mail: highman160@cad.t.u-tokyo.ac.jp, tasuku@cad.t.u-tokyo.ac.jp, matsumoto@cad.t.u-tokyo.ac.jp, fujita@ee.t.u-tokyo.ac.jp

あらまし　近年、設計期間を短縮するために設計資産の再利用がよく行われている。その際、異なるインタフェースを持つ設計同士を接続する場合にはプロトコル変換器を設計し挿入する必要がある。そのため、設計資産再利用を行う設計において、プロトコル変換器の設計と検証は重要である。そこで、本研究では、異なるプロトコルで通信するモジュール間を接続するプロトコル変換器の形式的検証手法を提案する。提案手法では、通信する双方のプロトコルの仕様の積を取ることにより、プロトコル変換器設計の仕様を得る。その後、シミュレーションによって、その仕様のうち設計において実装されている部分を抽出し、そこから設計が満たすべきプロパティを作成する。最終的に、このプロパティによって形式的検証を行い、全てのプロパティを満たせば設計の正しさを証明することができる。実験として、提案手法による AMBA と OCP のプロトコル変換器に対する検証結果を示す。
キーワード　プロトコル変換器、形式的検証、プロパティ検証、仕様

Formal Verification Method for Protocol Transducer Using Automatically Generated Properties from Specification

Fei GAO[†], Tasuku NISHIHARA[†], Takeshi MATSUMOTO[††], and Masahiro FUJITA[††]

† Department of Electronics Engineering, University of Tokyo
7–3–1 Hongo, Bunkyo-ku, Tokyo, 113–8656 JAPAN
†† VLSI Design and Education Center, University of Tokyo
2–11–16 Yayoi, Bunkyo-ku, Tokyo, 113–0032 JAPAN
E-mail: highman160@cad.t.u-tokyo.ac.jp, tasuku@cad.t.u-tokyo.ac.jp, matsumoto@cad.t.u-tokyo.ac.jp, fujita@ee.t.u-tokyo.ac.jp

Abstract　IP-reuse design is widely applied in order to reduce design period by utilizing already designed and well verified modules. When two IPs have different interfaces, it is necessary to design a protocol transducer to transduce the different protocols between them. In this paper, we propose a method to formally verify protocol transducers which transduce the different protocols. In our method, firstly, we generate the specification of the transducer under verification from the specification of the protocols. Next, a subset of the specification which is implemented in the transducer design is extracted by random simulation. Then, a set of properties that should be satisfied in the design is generated. If all of those properties are satisfied, the correctness of the protocol transducer design is proved. We also show the results of the experiments for transducer designs connecting AMBA and OCP protocols.
Key words　Protocol transcuder, Formal verification, Property checking, Specification

1. はじめに

近年、設計期間を短縮するために、新規に SoC(System-on-Chip) やシステム LSI を設計する際に、既存の設計資産を用いる IP 再利用設計が広く適用されている。IP 再利用設計では十分に検証された IP が用いられることが多いが、IP ごとに通信

プロトコルが異なる場合がある。その場合、プロトコル変換器を設計して IP 間に挿入する必要があるため、新しく設計されたプロトコル変換器に設計誤りがないかを検証することが重要となる。本研究では異なるプロトコルで通信する IP 間を接続するプロトコル変換器の検証を目的とする。

通常、プロトコルの仕様は自然言語やタイミングチャートなどで表現されるが、自動検証には数学的な仕様が必要不可欠である。本研究では、Watanabe らの手法 [4] を利用し、有限状態機械として表現された 2 つのプロトコル仕様の積を取ることによって、プロトコル変換器の仕様を導出する。文献 [4] の手法では、その仕様を満たす 1 つの実装を選ぶことによってプロトコル変換器を合成しているが、本研究では、その仕様を利用してプロトコル変換器の設計が正しいかどうかを検証する。

そこで、本研究では、異なるプロトコルで通信する IP 間の接続に必要なプロトコル変換器を検証する問題を、Watanabe らの手法を利用して生成した変換器の仕様が、検証対象である変換器設計を包含することをモデル検査で検証する問題に帰着する。モデル検査は、設計を網羅的に検証できる形式的検証手法の 1 つであり、設計者の意図によって LTL や CTL などの時相論理記述で与えられたプロパティが、状態遷移機械に展開された設計において成立することを網羅的に検証する手法である。しかし、モデル検査での検証では、いくつのプロパティを検証すれば設計が正しいと言うことができるか、が大きな問題であるが、それを判断する明確な指標は存在していない。そのため、モデル検査による検証では、許される時間内にできるだけ多くのプロパティを検証することが行われている。本研究では、積機械を利用して包含関係を検証するためのプロパティの完全性を定義し、それに沿った完全なプロパティ集合を自動生成する手法を提案する。生成されたプロパティを全てモデル検査することによって、積機械と変換器設計の包含関係を検査することができる。

本研究では、通信する双方のプロトコルの仕様から、積機械を作成する。シミュレーションを用いて積機械と検証対象である変換器の対応を取ることによって、与えられた変換器の動作と無関係のことを積機械から削除することによって、検証に十分なプロパティを効率的に生成する。最終的に、このプロパティによって形式的検証を行う。実験として、提案手法による AMBA [1] と OCP [2] の両プロトコルを接続する変換器の検証結果を示す。

本稿の構成は以下のようになっている。第 2 節において関連研究について説明する。第 3 節において、本研究の提案した積機械と変換器設計の包含関係を検証する手法を説明する。第 4 節では、AMBA と OCP に対するプロトコル変換器を実験対象として行った検証の実験結果を示す。第 5 節では、まとめと今後の予定を述べる。

2. 関連研究

2.1 積機械

Passerone らは文献 [3] において、有限状態機械として記述された 2 つのプロトコルの仕様記述を入力とし、そのプロトコル間の通信を行うプロトコル変換器の動作を有限状態機械として合成する手法を提案している。また、Watanabe らは文献 [4] において、この手法を拡張し、SingleRead や SingleWrite といったシーケンスレベルごとに変換器の動作を表す有限状態機械を合成し、それらを組み合わせることによって、より複雑なプロトコル変換器を合成可能な手法を提案している。

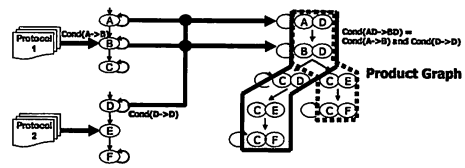

図 1　プロトコル試用からの積機械生成

文献 [3], [4] における変換器合成の様子を示したのが、図 1 である。入力となる 2 つのプロトコルは、入力や出力の信号で発生するイベントの遷移順序を規定した正規表現として記述されていることを前提としている。それぞれのプロトコル仕様は、この正規表現から、等価な有限状態機械に変換される。そして、この 2 つの有限状態機械から積機械を生成する。この積機械の状態遷移条件は、その状態を構成する元のそれぞれの状態における遷移条件の論理積を取ったものになる。この積機械を作る過程で、依存関係が違反する状態 (変換器として到達してはいけない状態) とそこへ到達する可能性のある状態については積機械から除く。そのため、得られた積機械は、プロトコル変換器の仕様として見ることができる。それらの合成手法では、この仕様から開始状態から終了状態に到達する経路のレイテンシが最小となるような部分機械を選ぶことにより、変換器の動作となる有限状態機械を得ることができる。

図 1 において、プロトコル 1 の状態 A と状態 B、プロトコル 2 の状態 D から、積機械において状態 AD と状態 BD が生成される。このとき、状態 AD から状態 BD への遷移条件 $Cond(AD \rightarrow BD)$ は、元のプロトコル 1 の有限状態機械における状態 A から状態 B への遷移条件 $Cond(A \rightarrow B)$ と、プロトコル 2 の有限状態機械の状態 D から状態 D への遷移条件 $Cond(D \rightarrow D)$ を論理積を取ったものになっていることが分かる。

なお、この手法によって生成される積機械では、1 つの状態から同じ遷移条件で異なる状態へ遷移することがあり得る。そこで、積機械を生成する際は、そのような遷移を 1 つにまとめることにより、同じ遷移条件で遷移する先の状態が一意に定まるような有限状態機械としている。

2.2 プロパティの完全性

Bormann らは文献 [5] において、ある設計を検証するための完全なプロパティの集合とは、その設計のある状態における全ての入力パターンがプロパティによってカバーされており、かつ、そのプロパティはどの時刻 (クロックサイクル) でも全ての出力を一意に決められることである、と定義している。その完全性を満たすプロパティによって、設計を十分に検証することができると主張している。しかし、このようなプロパティの生

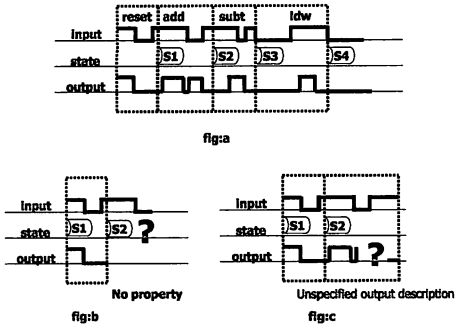

図 2　プロパティの完全性

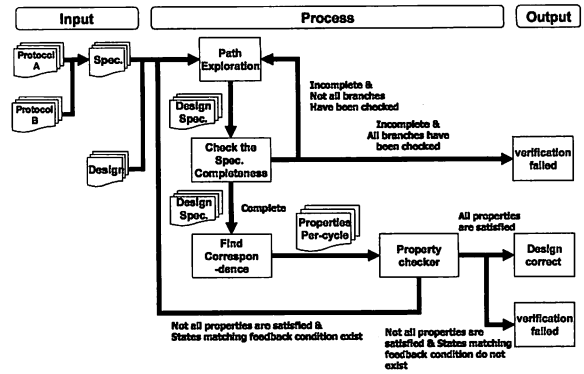

図 3　提案する検証フロー

成は、設計と同程度の労力が必要である。

文献 [5] では、プロセッサの 1 命令ごとにプロパティを生成した例が示されている。ここでは、図 2 を利用して、その例におけるプロパティの完全性について説明を行う。点線の四角形で囲まれている部分は、各命令に対するプロパティを表している。ここで、連続した命令に対するプロパティにおいては、前のプロパティの結果となっている状態 (の集合) と、後のプロパティにおいて前提となっている状態 (の集合) が同じであること、どの時刻においても出力変数が定まっていることが必要である。図の (a) は完全性が満たされている例である。一方、図の (b) と (c) は、プロパティが不足しているため、また、出力変数がプロパティ中で定義されていないために、完全ではない例となっている。本研究では、Bormann らのプロパティ完全性の定義が、プロトコル変換器設計の 1 クロックサイクルごとに生成されたプロパティによって満たされるようにすることにより、完全なプロパティ検証を実現する手法を提案する。

2.3　シミュレーション結果を利用したプロパティ生成

文献 [6], [7] において、設計を正しく動作させる (設計誤りがエラーとして結果に現れない) テストパタンの集合から、設計の満たすべきプロパティを自動生成する手法が提案されている。これらの手法では、一般的に、シミュレーションがより網羅的であれば、生成されたプロパティによってより多くのことが検証できると言える。これらの手法の問題点の 1 つは、シミュレーション結果から生成されたプロパティが設計者にとって理解が難しく、また、限られたシミュレーション結果からプロパティを導出するため、生成されたプロパティが本当に設計によって満たされるべきである保証がないことである。本研究では、2 つのプロトコル仕様から生成された有限状態機会である積機械上で生成したテストパタンによって仕様と設計間の対応を取り、プロパティを生成する。そのため、検証結果が正しい場合には、設計は必ず仕様を満たしていると言える。

3.　積機械と設計の包含関係を検証する手法

本研究では実装であるプロトコル変換器の設計が、仕様となるプロトコルの積機械 (2.1 節参照) に包含されているかをモデル検査により判定する。提案する検証フローを図 3 に示す。

フローの入力は、変換器にて変換が行われる 2 つのプロトコルをそれぞれ表す 2 つの有限状態機械と、検証対象のプロトコル変換器設計である。まず、プリプロセスとして、2.1 節で紹介した Watanabe らの手法を利用して積機械を合成する。これが、プロトコル変換器の設計が満たすべき仕様となる。

次に、積機械から、設計に対応する部分を抽出するため、シミュレーションによる探索を行う。この時、抽出された部分機械が独立のプロトコル変換器として正しく動作するか (全ての入力パタンに対する動作が規定されている) を判定し、不十分な場合は再度シミュレーションを行い、抽出する部分を広げることを繰り返す。

続いて、シミュレーションにより、抽出された部分機械における状態と設計の内部状態 (レジスタ値) との対応を取る。これにより、部分機械における各状態遷移について独立にプロパティを生成することができるようになるため、プロパティが監視するサイクル数が短くなり、大規模な設計であっても短時間で検証が可能となる。ただし、有限状態機械の遷移にループがある場合は、ループを繰り返す回数によっては正しく対応が取れない場合があるため、一定回数を上限として、正しく検証が終了するまでループを辿る回数を変化させる。生成されたプロパティは既存のモデル検査器を用いて検証する。

本節では、以降、各ステップについて詳細に説明する。

3.1　入力となる積機械

図 4(a) に、入力となる積機械の例を示す。積機械は、状態と状態遷移によって構成され、各状態遷移に対して遷移条件が存在する。遷移条件は、入出力信号、定数値、および比較演算子により構成される。例えば、状態遷移 $s_1 \rightarrow s_1$ の遷移条件 T_1 において、in_1 は入力信号、out_1 は出力信号である。ある 1 つの状態からの遷移における遷移条件は、互いに排他的となっている。

積機械には通常、複数のプロトコル変換器の実装が含まれている。そのような場合は、ある状態からの状態遷移の遷移条件において、入力信号による条件が同一であるものが存在する。先に述べたように遷移条件は排他的であるため、この場合は出力信号による条件が異なる。すなわち、条件を満たす入力信号を入れた際に、正しいプロトコル変換器の実装は、どちらかの条件を満たす出力信号を返すということになる。

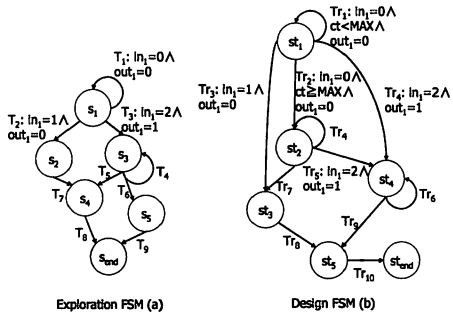

図 4　積機械とプロトコル変換器実装の例

3.2　シミュレーションのテストパターンの生成

変換器の設計が積機械によって規定された動作に含まれているかどうかを検証する際に、積機械の全ての動作をプロパティとして表現すると、一般的に、非常に複雑で大規模なプロパティとなり、検証時間も長くなってしまう。そのため、本研究では、変換器の設計の動作に対応するプロパティのみを生成して検証を行う。そのためには、積機械から検証対象の設計に対応する部分を抽出する必要がある。この抽出作業は、積機械のある状態遷移列の遷移条件を満足する入力パタンを作成し、それを用いて設計をシミュレーションした場合の出力信号の値が、その遷移条件を満足するかどうかを判定することにより行われる。

例えば、図 4(a) において、$s_1 \rightarrow s_2$ の遷移条件を満たす入力パタンは $in_1 = 1$ である。したがって、検証対象の設計をその入力パタンでシミュレーションし、出力が $out_1 = 0$ となる場合、その遷移条件を満たしているため、その設計に状態遷移列 $s_1 \rightarrow s_2$ に対応する動作が存在することが分かる。このようなシミュレーションを積機械の各状態遷移列に対して行い、設計に存在すると判定された状態遷移列の和が、その設計に対応する部分機械となる。

しかし、図 4(a) の例のようにループが存在する積機械においては、ループを辿る回数により状態遷移列が異なるため、状態遷移列の数は無限となり、全ての状態遷移列をシミュレーションすることは不可能となる。そこで、ループを辿る回数の上限としてユーザーが指定し、その範囲内で判定を行う。この時、指定した回数が不十分な場合は、設計で実装されている動作に対応する部分が抽出できない場合がある。例えば、図 4(b) は、(a) の積機械による仕様を実装した設計例である。設計の 2 つの状態 st_1, st_2 が、積機械の状態 s_1 に対応している。ここで、出力信号 ct は、$st_1 \rightarrow st_1$ の状態遷移を辿った回数を示すカウンターの値を表している。すなわち、この設計の場合は、ct_1 が MAX 以上、すなわち $st_1 \rightarrow st1$ の状態遷移を MAX 回辿って初めて st_4 ((a) の積機械における状態 s_3 に対応) に遷移する。したがって、実際には (a) の積機械における $s_1 \rightarrow s_3$ に対応する状態遷移 ($st_2 \rightarrow st_4$) が存在しているものの、それを正しく判定するためには、$s_1 \rightarrow s_1$ のループを MAX 回辿る入力パタンを生成する必要がある。さらに、このようなループが複数個存在する設計では、それぞれのループを辿る場合の数の積だけ、シミュレーションを繰り返す必要があるため、計算量は状態数に対して指数的に増大する。そこで、提案手法では、計算量を抑えるために、その時点で抽出した部分機械がプロトコル変換器として十分であれば、その時点でシミュレーションを止める。

3.3　部分機械が十分である条件

前節で述べた通り、提案手法では、ある時点で抽出した部分機械がプロトコル変換器として十分であれば、その時点でシミュレーションを止める。ある部分機械がプロトコル変換器として十分である条件とは、全ての有効な入力パタンに対しての動作が規定されていることである。有効な入力パタンは、積機械上の状態遷移における遷移条件を満たす入力パタンであるから、先の条件は次のように言い換えることができる。"部分機械上のどの状態においても、対応する積機械上の状態における任意の状態遷移の条件を満たす状態遷移が存在する" 数式で表現すると、以下のようになる。

積機械上の状態の集合を S^p、部分機械上の状態の集合を S^s とする。二つの集合には、$S^s \subseteqq S^p$ の関係がある。次に、積機械上のある状態を与えた時にその状態からの全ての状態遷移における遷移条件の論理和を返す関数 $Cond^p$、部分機械上のある状態を与えた時にその状態からの全ての状態遷移における遷移条件の論理和を返す関数 $Cond^s$ を、それぞれ次のように定義する

$$Cond^p : s^p \rightarrow ci^p$$

$$Cond^s : s^s \rightarrow ci^s$$

ここで、$s^p \in S^p$、$s^s \in S^s$ であり、ci^p, ci^s は入力信号に関する条件式である。これらの関数を用いて、先に述べた部分機械がプロトコル変換器として十分な条件は下記のようになる。

$$\forall s \in S^s (Cond^p(s) \rightarrow Cond^s(s))$$

3.4　状態遷移の存在が確認された場合のシミュレーションの省略

本節では、設計が抽出された部分機械に対して満たすべき性質に着目して、シミュレーションを省略する手法について述べる。

設計と抽出された部分機械が正しく対応している場合、部分機械における任意の状態遷移に対して、それに対応する状態遷移が設計中に存在することになる。したがって、積機械におけるある状態遷移列の存在が確認された場合、その状態遷移列中の状態遷移についての存在が保証されることになる。したがって、存在が確認された状態遷移の組み合わせによる状態遷移列については、既に全ての状態遷移の存在が確認されているため、改めてシミュレーションを行う必要がない。

例えば、図 4(a) の例において、$s_1 \rightarrow s_1 \rightarrow s_2$ という状態遷移列の存在が確認された場合は、$s_1 \rightarrow s_1 \rightarrow s_1 \rightarrow s_2$ という状態遷移列についてのシミュレーションを省略することができる。

また、ある状態と入力信号に関する遷移条件について、その状態からの状態遷移で、その遷移条件を満たすものが一つしか存在しない場合、その状態に到達するまでのある状態遷移列の

存在が確認済みであるならば、その状態遷移列にその状態遷移を連結した状態遷移列についてのシミュレーションの結果のみから、その状態遷移の存在を判定することができる。

例えば、図 4(a) の例において $s_1 \rightarrow s_3$ の入力信号に関する遷移条件は $in_1 = 2$ であり、s_1 からの他の状態遷移において、同じ遷移条件を持つものはない。したがって、状態 s_1 において $in_1 = 0$ という入力がされた場合、必ず出力は $out_1 = 1$ となり、状態 $s2$ に遷移する。設計中にこの状態遷移に対応する遷移が存在する場合、必ず同じ挙動をしなくてはならない。したがって、$in_1 = 2$ という入力パタンでシミュレーションを行った結果、$out_1 = 1$ という出力を返さない場合は、設計中には $s_1 \rightarrow s_3$ に対応する遷移は存在しないことが確定するため、$s_1 \rightarrow s_3$ を含む他の状態遷移による確認を省略できる。

以上の手法により、シミュレーションの回数を少なく抑えることが可能となる。

3.5 レジスタの対応

モデル検査の検証時間を短くするためには、プロパティが監視するサイクル数を小さくすることが重要となる。なぜならば、一般にモデル検査器では、監視するサイクル数分だけ設計が展開されるため、検証するプロパティのサイクル数に対して計算量が指数のオーダーとなるからである。

提案手法では抽出された部分機械からプロパティを生成するが、そのプロパティの監視するサイクル数を小さく抑えるためには、抽出された部分機械における状態と設計の内部状態 (レジスタ値) との対応が必要となる。なぜならば、その対応情報が無い場合、現在いる部分機械の状態は、過去の入力および出力パタンから判定するしかない。例えば、図 4(a) の例において状態 s_4 にいるかを判定するためには、T_2, T_7 を順に満たすような入力および出力パタンが確認する必要がある。この場合、状態 s_4 において検証するプロパティに現在 s_4 にいるかどうかの判定を組み込むと、監視するサイクル数は 2 サイクル増大することになる。大規模な機械においてはこの監視するサイクル数が膨大な数になることが考えられるため、検証時間に致命的な影響を与えることが予想できる。

提案手法では、抽出された部分機械における状態と設計の内部状態 (レジスタ値) との対応情報は以下のようにして得る。

（1） 設計中の状態変数に対応するレジスタをユーザーが指定する

（2） 部分機械から、それぞれの状態に到達するような入力パタンを生成する

（3） その入力パタンを用いて設計をシミュレーションし、その状態に到達した時の、指定されたレジスタの値を記録する

先の例では、T_2, T_7 を順に満たすような入力パタンを生成し、それを用いてシミュレーションをした結果のレジスタ値が、状態 s_4 に対応するレジスタ値となる。

ここでユーザーは、状態変数に対応するレジスタを指定する必要があるが、この指定が間違っている場合は正しく検証を行うことができない。

3.6 プロパティの生成

a） コントロールに関するプロパティ

前節のプロセスで得た部分機械上の状態と設計のレジスタ値との対応関係を用いて、部分機械の各状態遷移が、設計において正しく反映されているかを判定するプロパティを生成する。状態変数に対応する設計中のレジスタ (複数ビット) の値を r、ある状態 $s \in S_s$ に対応するレジスタ値を r_s、s からのある状態遷移の遷移条件を c、遷移先の状態 s' に対応するレジスタ値を r'_s とすると、その状態遷移に対応するプロパティの LTL 表記は以下のようになる。

$$(r = r_s \wedge c) \Longrightarrow X(r = r'_s)$$

例えば、図 4(a) の例において、設計の状態変数に対応する設計中のレジスタを reg、状態 s_1 に対応するレジスタ値を 0 とした場合、$s_1 \rightarrow s_1$ に対応するプロパティは下記のようになる

$$(reg = 0 \wedge in\_1 = 0 \wedge out\_1 = 0) \Longrightarrow X(reg = 0)$$

ここで生成されたプロパティおよびその集合の特徴は下記の二点である。

- 一つ一つのプロパティの監視するサイクル数は 1 サイクルであり小さいため、大規模な設計に対しても高速な判定が可能である
- 状態変数に対応するレジスタについて、プロパティの集合は 2.2 節で紹介した完全性を満たしているため、全てのプロパティを満足する場合は、部分機械が設計を包含していることが保証される。

b） データに関するプロパティ

正しいプロトコル変換器の動作を保証するためには、変換器の両サイドの動作がそれぞれのプロトコルを満たしているかどうか以外に、転送されるアドレスやデータが両サイドにおいて変化していないかどうかを検証する必要がある。アドレスがデータが入出力されるタイミングは状態により決定されるため、その状態に対応するレジスタ値を用いてプロパティを記述する。あるデータ (アドレス) が入力されるポートを i_d、入力される状態に対応するレジスタ値を r_i、そのデータ (アドレス) が出力されるポートを o_d、出力される状態に対応するレジスタ値を r_o、プロパティにおける変数を x とすると、以下のようにプロパティを書くことができる。"$r = r_i \wedge i_d = x$ が成立するならば、いつか $r = r_o \wedge o_d = x$ が成立する" このプロパティは変数の定義が必要であるため LTL では記述することができないが、PSL や SVA では記述することができる。

ただし、このプロパティは大きなサイクル数を監視する必要がある可能性があるため、大規模な設計に対しては適用できない可能性がある。その場合は、[5] にあるように、設計からデータに関する仕様を作り、それを元に監視するサイクル数が小さいプロパティを生成する必要がある。

3.7 手法の適用可能範囲

本節では、本手法の扱えるプロトコルの範囲について述べる。

プロトコルの有限状態機械に同一の状態および初期状態以外へのループが存在する場合に、2.1 節で紹介した Watanabe ら

表1 実験結果

	仕様規模 (状態数)	テストパタン	プロパティ	検証時間 (秒)
OCP ⇒ AMBA プロトコル変換器				
Single Read	11	29	11	6.1
Single Write	10	26	13	6.1
Burst Read	62	255	115	121.6
AMBA ⇒ OCP プロトコル変換器				
Single Read	11	29	11	2.9
Single Write	10	26	13	3.0

の手法では、網羅的な合成は行われない。したがって、その場合、正しい設計であっても、その設計が正しいという判定ができないものが存在する可能性がある。

また、データ (プロトコルの有限状態機械に記述されている信号以外のもの) やその送受信の順序に変更が加えられる場合、仕様にそれらの情報が記述されていないため、正しい設計であっても、その設計が正しいという判定ができないものが存在する可能性がある。この場合は、そのような情報に対応するプロパティを別途検証する必要がある。例えば、Out of Order のプロトコルについては、データのタグに関するプロパティを追加する必要がある。また、バースト転送についても、変換器内部にカウンタや fifo が存在する場合のプロパティを追加する必要がある。

4. 実験結果

OCP と AMBA のプロトコルを変換する変換器を例題として提案手法による実験を行った。用意した変換器は、マスターが OCP・スレーブが AMBA AXI プロトコルで動作する変換器 (OCP ⇒ AMBA) と、マスターが AMBA AXI・スレーブが OCP プロトコルで動作する変換器 (AMBA ⇒ OCP) の 2 つである。それぞれの回路規模は、Cadence 社 PKS と Rohm 0.18um スタンダードセルライブラリを用いて論理合成をして求めたところ、

- **OCP ⇒ AMBA** 3391 ゲート、264FF
- **AMBA ⇒ OCP** 2982 ゲート、276FF

であった。

積グラフの生成は、文献 [4] の手法を実装したツールを用いている。実験環境は以下の通りである。

- 実験環境: Intel Xeon 3.2GHz Dual, 4GB Memory, Fedora Core 4
- プロパティ検証ツール: Synopsis 社 Magellan

本実験はコントロール信号についてのプロパティだけ検証を行った。

OCP ⇒ AMBA 変換器において、1 ワード読み込み (Single Read) と 1 ワード書き込み (Single Write)、さらに、バースト読み込み (Burst Read) の各機能が正しく動作するかどうかの検証を行った。Burst Read の検証では、実験は仕様である積機械から部分機械を抽出する部分は手動で行っている。今回の例題は、仕様規模が小さいため、シミュレーションを用いた部分機械の抽出は 0.1 秒以内で行うことができた。表 1 に示すように、各通信の正しさの検証時間はそれぞれ 6.1、6.1、121.6 秒であった。AMBA ⇒ OCP 変換器においては、Single Read と Single Write の各機能が正しく動作するかどうかを検証し、3 秒程度で検証を行うことができた。また、変換器に意図的に誤りを挿入したところ、仕様である積機械のうち設計で実装されている部分をシミュレーションによって抽出する際に、仕様上の状態遷移を満たすような動作が設計で見つからなかった。

5. まとめと今後の課題

本稿では、二つのプロトコルの有限状態機械の積機械を仕様として完全性を満たすプロパティを自動生成し、プロトコル変換器設計を検証する手法を提案した。その際に、設計に対応する部分機械を抽出し、さらに部分機械の各状態と設計の内部状態との対応を取ることにより、監視するサイクル数が小さい必要最低限のプロパティの集合を作成できる。実験を通して、このプロパティの集合により、短時間で実設計の検証ができることを示した。

今後の課題としては、以下の項目が挙げられる。

- 本稿では、OCP と AMBA プロトコルに含まれる SingleRead などの一部のシーケンスについての検証を行ったが、今後はプロトコル全体について、もしくは他のプロトコルについても実験を行い、手法の実用性を確認したい。
- 提案手法では設計の状態変数に対応するレジスタをユーザーが指定する必要があるが、それの自動検出、もしくはユーザーの指定の補助についても研究を進めたい。
- 本稿ではデータに関するプロパティの生成および特殊なプロトコルに対応するための追加プロパティについては深く言及しなかったが、今後研究を進めて汎用的に適用できる手法としたい。

文　献

[1] *AMBA AXI* **プロトコル仕様書** *v1.0*, ARM Limited, 2003.

[2] *Open Core Protocol Specification version 2.1*, OCP International Partnership, 2005.

[3] R. Passerone, J.A. Rowson, and A. Sangiovanni-Vincentelli, "Automatic Transducer Synthesis of Interfaces between Incompatible Protocols," *Proc. of Design Automation Conference*, pp.8-13, 1998.

[4] S. Watanabe, K. Seto, Y. Ishikawa, S. Komatsu, and M. Fujita, "Protocol Transducer Synthesis using Divide and Conquer approach," *Proc. of the 12th Asia and South Pacific Design Automation Conference*, pp.280-285, 2007.

[5] J. Bormann, S. Beyer, A. Maggiore, M. Siegel, S. Skalberg, T. Blackmore, and F. Bruno, "Complete Formal Verification of TriCore2 and Other Processors," *Proc. of DVcon 2007*, 2007.

[6] B. Isaksen, V. Bertacco, "Verification Through the Principle of Least Astonishment," *Proc. of International Conference on Computer-Aided Design 2006*, pp.860-867, 2006.

[7] F. Rogin, T. Klotz, G. Fey, R. Drechsler, S. Rulke, "Automatic Generation of Complex Properties for Hardware Designs," *Proc. of Design, Automation and Test in Europe 2008*, pp.545-548, March 2008.