システムロバスト性向上のための モデル検査による検証手法の提案と 宇宙機搭載 FPGA への適用事例

梅田浩貴[†1]　倉林翔[†2]　小口一浩[†1]　植田泰士[†1]　片平真史[†1]
早水公二[†3]　森崎修司[†4]

概要：宇宙機等のクリティカルシステムやそのソフトウェアの開発では，想定外の事象にも対応できるようロバスト性を高めるために，異常値の入力やタイミングのずれといった，全てが仕様に規定されている範囲ではない「仕様外」も含めた検証が重要である．形式手法の 1 つであるモデル検査は，状態遷移系で記述された振る舞いに対する検証が可能である．一方，モデル検査は大規模な状態遷移系になると状態爆発が生じてしまう．その対策として，状態遷移系上で「抽象化や絞り込み」が行われるが，実際のシステムの振る舞いと不一致が発生し，偽反例の出力やシステムに影響がない反例等によって，開発活動上の検証価値を低下させてしまう．よって，下流工程で検出困難な「仕様外」やシステム影響度の高い事象に特化してモデル検査を用いた検証手法を提案する．本論文では，GSN を用いたゴール指向アプローチで検査式の生成や絞り込みを行う方法の構築方法と，宇宙機搭載 FPGA の上流設計に対し提案手法で構築した検証手法の有効性評価の結果を報告する．

キーワード：検査式，モデル検査，GSN，反例 ，ソフトエラー

1. はじめに

宇宙機等のクリティカルシステムやソフトウェア[1]（SW）の開発では，複雑化や大規模化が進み，想定外の事象にも対応できるロバスト性が重要である．例えば，システム及び SW の国際規格 SQuaRE[2]では，利用時品質のリスク回避性があり，その具体的な指標として仕様外のリスクシナリオを考慮している件数[3]がある．システムの稼働時には発生するが仕様外となる対象として，異常入力やタイミング等があり，再現性の低い不具合となってしまう．再現性の低い不具合の検出方法として，形式手法の 1 つであるモデル検査[4]がある．モデル検査では，検査したい対象を有限状態機械として捉えモデルを作成し，検査式として表現された性質がモデルにある状態空間を満たすかを検査する．モデル検査を設計検証として適用する際，検査式及びモデル作成の追加コストや，計算資源からの制約による状態爆発問題に対応するため抽象化等のノウハウが必要になる．そのため，モデル検査技術に精通した技術者による開発と並行しての適用や，モデル検査技術を隠蔽した支援ツールによって開発者自身による適用，といった取り組みが行われていた[5][8]．本論文では，従来のアプローチに加えて，モデル検査による追加コストを負担することに相応する検証価値を保持するため，発生頻度が低く検出困難且つシステムへの影響度が高い不具合の検出を目的とした手法を提案する．

発生頻度が低い不具合の検出には，検証の論理的な網羅性の向上が効果的である．一方，論理的な網羅性の向上のみでは，検証ケースの爆発が発生してしまうため，検証価値がある検証ケースに絞り込むためには設計エキスパートのドメイン知識と論理的な網羅性の融合も必要である．特にシステムロバスト性を向上するために必要なドメイン知識は，設計エキスパートが仕様にはないが実際の振る舞いとして考慮している「仕様外に関するドメイン知識」が重要となる．そこで提案手法は，設計エキスパートが保有するドメイン知識をゴール指向記法である GSN（Goal Structuring Notation）[12]により知見として体系化し，因果関係のあるシステムからその構成要素であるコンポーネントやまでの前提条件を組み合わせることで，

・システム影響度と仕様外を考慮した検査式の作成
・制約や前提条件の抽出による偽反例の防止
・検証価値があるシステム内外の特定状況による絞り込み

を実現する方法とした．なお GSN は，ゴール毎に前提条件を提示し，上位と下位のゴールを接続することで因果関係を表現できる特徴と，ゴール間を接続するストラテジーによって論理的な網羅性を向上できる特徴を有するため提案手法では採用し，設計エキスパートのレビュー観点の体系化として活用している．

また，本手法を基に，宇宙機搭載の FPGA（field-programmable gate array）の論理設計に対する検証手法を構築し，設計経験が浅くモデル検査の知識を保有しない設計者に対して，RQ1：仕様外を含めた検証価値が高い検査式が作成できるか，RQ2：影響度が高く検出が困難な問題点を検出できるか，について有効性を評価した結果を示す．

†1 国立研究開発法人　宇宙航空研究開発機構
Japan Aerospace Exploration Agency (JAXA)
†2 株式会社エイ・イー・エス Advanced Engineering Services Co., Ltd.
†3 株式会社フォーマルテック
†4 国立大学法人 名古屋大学　Nagoya University

2. 関連研究

2.1 モデル検査の開発現場への適用

形式手法の 1 つであるモデル検査は，並行動作するシステムの振る舞いに関する正しさを保証するための技術として使われている[6]. モデル検査を開発現場に適用する際のプロセス[7]では，計算，知識及び作業リソースの制約から下記の課題が発生する（図 1）.

・モデル，検査式等の作成時に専門知識と工数が必要
・モデル検査器の実行時に状態爆発問題
・反例解析時に偽反例の発生

モデル検査を開発現場へ適用するアプローチは，自動化やモデル検査知識の隠蔽によって適用コストを下げていく方法と，レビューやテスト等の既存検証活動と差別化する方法がある. 前者では, 図 1 にある「検証対象モデルの設計」,「検証記述の実装」,「検証の実行とフィードバックに焦点」を当て，UML（Unified Modeling Language）等の既存の設計成果物からの自動変換[8]，反例解析の支援[9]等，多くの提案がされている.

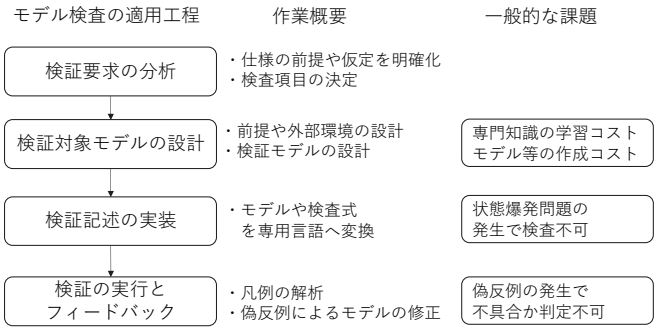

図 1:モデル検査プロセスと適用課題

提案手法は，後者に主軸を置き，図 1 にある「検証要求の分析」に焦点を当て，ドメイン知識の抽出と体系化をゴール指向アプローチで行い，工程 2 以降の検査式の生成やモデルの絞り込み等に活用することで，開発活動上の検証価値を向上する方法である．先行研究として，ドメイン知識を用いた検査式の作成は，規格等の公知の情報から FTA（Fault Tree Analysis）を用いた事象の詳細化[10]があるが，システムのロバスト性向上のために仕様外となるドメイン知識の抽出をどのように行うべきか言及されていない．またゴール指向アプローチとモデル検査の組み合わせとして，外部環境のモデルを識別するための方法[11]はあるが，システムの内部構成要素が実現するロバスト性向上策に関して，検出が困難でシステム影響度の高い問題点を検出するために重要となる前提条件をどのように組み合わせるべきかについて言及されていない.

2.2 ゴール構造化記法（GSN）

GSN とは，議論を可視化するために考案された記法[12]であり，主に安全性の論証で必要なアシュアランスケース[13]でも使われている．国内では，アシュアランスケースの 1 つである D-Case として知られている[14]．GSN の記法としてのルールを示す図 2 は，ゴール（主張）で構成された，類似のツリー型である FTA との最大の違いは，前提（コンテキスト）と網羅である視点（ストラテジー）を明示しており，ゴールの抜けに関する議論が容易になる点である. ソフトウェア FMEA（Failure Mode and Effect Analysis）の故障モードを分析する際にも応用している[15].

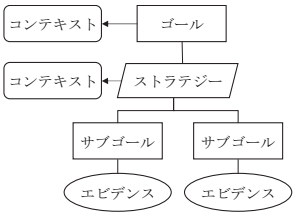

図 2：GSN モデリングルール

2.3 FPGA の設計とソフトエラー

システムとしてのロバスト性が必要とされ，且つ，宇宙機への搭載によって「仕様外」を具体的に考慮しなければならない FPGA システムとしてバス制御 FPGA システムがある. バス制御 FPGA システムは，コンポーネントとして複数のデバイス（センサー，CPU，メモリ等）が搭載され，有機的なシステムとして振る舞っている[16]．また，FPGA は多数の入出力信号を並列な制御と，素子の多段配置でタイミングの制御であるため，再現性の低い不具合が発生しやすい[17]. さらに，宇宙機搭載の FPGA は，放射線耐性があるデバイスを採用するが，一般の FPGA を搭載することも増えており，放射線によるデータが一時的な異常値となる事象（ソフトエラー）[18]，つまり各デバイスの仕様やプロトコル規約に定義されていない,「仕様外」を考慮する必要がある. バス制御 FPGA システムの上流設計では，回路ブロック図（FPGA 論理部と各デバイスとの入出力関係を示したシステム入出力構造図），ステートマシン（入出力の条件）で設計している（図 3）．その際，通信先装置や異なるプロトコルへの切り替え[19]といった重要なタイミング（コーナーケースと呼ぶ）を FPGA タイミングチャートの作成で検証している[20]. 信号の衝突等の問題がある場合, 再度, 回路ブロック図や FPGA ステートマシンを修正し，再帰的に設計を行っている．コーナーケースの検出には，仕様外がありうるデバイス特性と要件を組合せる知識が必要であり，属人性が高い.

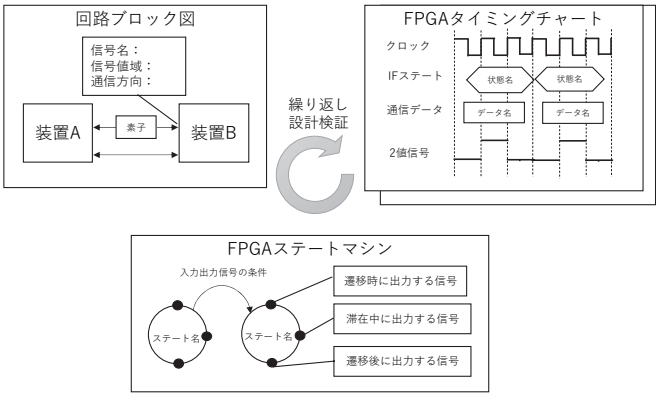

図 3：FPGA 論理部外部インターフェース設計

3. 提案手法

図 4 は提案手法の全体の入出力関係を示している．仕様外情報を含むレビュー結果や不具合情報等から前提付き構造型知見及び検査式の作成方法（3.1 項），絞り込みを行うためのモデルの作成方法（3.2 項），反例解析ビューの設計方法（3.3 項）を提案する．

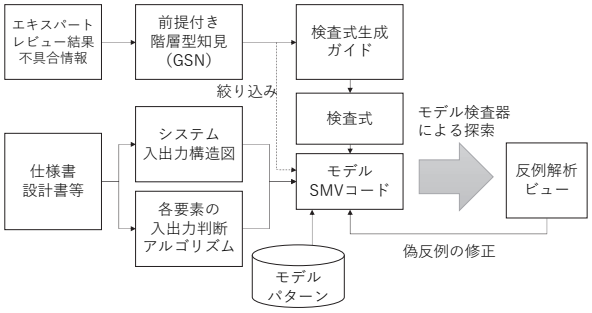

図 4：提案手法の入出力関係

3.1 検査式の作成方法

従来の検証活動に加えて，モデル検査による追加コストを計上しても検証価値が認められるためには，後工程で検証することができない，システムへの影響度が高い検査項目に焦点を当てる．そのため，仕様外となるドメイン知識を保有する設計エキスパートの知見を分析する．本項では，ドメイン知識の抽出と知見の体系化（3.1.1 項），最適化（3.1.2 項），適用（3.1.3 項）をする方法を述べる．

3.1.1 レビュー結果等から前提付き階層型知見の作成

本論文では，仕様情報とはシステム個別の要求要件やプロトコル等の規格，ハードウェアを起因とする制約等で設計者に与えられた要件を指す．設計情報とは，上記の要件を満たすために設計者が考案した設計対象の振る舞い（状態遷移図やフローチャート等で表現）を指す．よって，仕様外となるドメイン知識を分析するために，仕様情報に対するレビューコメント，又は，仕様情報と設計情報の対比で設計情報にしかない条件文等から，仕様外として考慮されたドメイン知識を抽出する．モデル検査で検証できる特性は状態遷移系で表現できる「振る舞い」であるため，設計情報の 2 者間比較（矛盾や不整合）のレビュー結果は除外する．

また，設計エキスパートに対して過去の経験をヒアリング等で知見として体系化する際，抽象的な表現に集約すると他者が具体化できない，具体化し過ぎると知見の量が多くなり過ぎて他者が探し出すことができない課題が発生する．よって，抽出すべきドメイン知識は，その目的とその有効条件が明確となるよう抽出する．レビュー結果や不具合情報から抽出するドメイン知識は，レビューの目的として設計対象が「懸念」となる状態およびシステムへの最終影響となる「懸念」，ドメイン知識の有効条件として過去の類似となる設計対象との差異である「特徴」となる．そのため，レビュー結果や不具合情報から，設計者やレビューアが考える懸念と設計対象の振る舞いが読み取れない場合，別途，分析フレームワークで補正する必要がある．[15]

また，開発の後工程で検出されにくい問題点（例：複数条件によって発生する振る舞い）や，システムや外部への波及する影響度の高い問題点を扱っているドメイン知識限定するため，下記の観点を満たすドメイン知識のみ抽出する．図 5 は，各観点の関係性を示している．

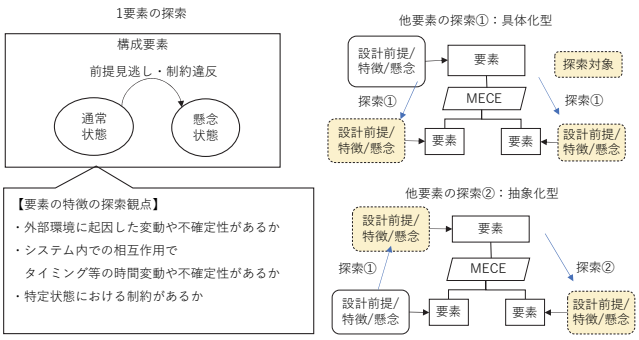

図 5：知見抽出観点（1 要素と要素間関係）

1：設計対象の特徴（違い）はあるか．（要素特定）
2：設計対象は懸念される状態は明確であるか．又は，設計対象が特定の状態になることでシステムへ影響があるか．（状態特定）
3：該当箇所の懸念される状態の発生条件（制約違反）は明確であるか．（条件特定）

次に抽出されたドメイン知識を体系化し「知見」とすることで，システムとシステム構成要素（コンポーネントや SW）の因果関係の整理によって，不足しているドメイン知識を抽出する．記法は，GSN を用いて要素を具体化および抽象化する 2 つの探索方法で行う．（図 5）

GSN では，ゴールにシステム，コンポーネント，SW の

振る舞い等といった設計対象を，コンテキストに（前提）として設計要素に付随した特徴，制約違反で発生する懸念を記載する. ゴールを分解するストラテジーは, 構成要素，状態，ガイドワードのいずれかで設計対象の分解する視点を，エビデンスは本手法では使用しない．本ルールで表現された GSN を「前提付き階層型知見」と呼ぶ．前提付き階層型知見 は下記のルールで作成する（図 6）

＜上位下位ゴール間の因果関係を満たした構築＞

上位ゴールにあるコンテキスト（設計前提）は，下位ゴールにあるコンテキストを満たすよう構築する．図 5 は，ゴールの抽象度によって新たなコンテキストを抽出して指針を示している．ゴールにある要素が抽象的である場合，自ゴールにあるコンテキストから因果関係のある下位ゴールのコンテキストを作成する（他要素の探索①:具体化型）．ゴールにある要素が具体的である場合，自ゴールにあるコンテキストから前提となる上位ゴールのコンテキストを導出する．（他要素の探索②：抽象化型）

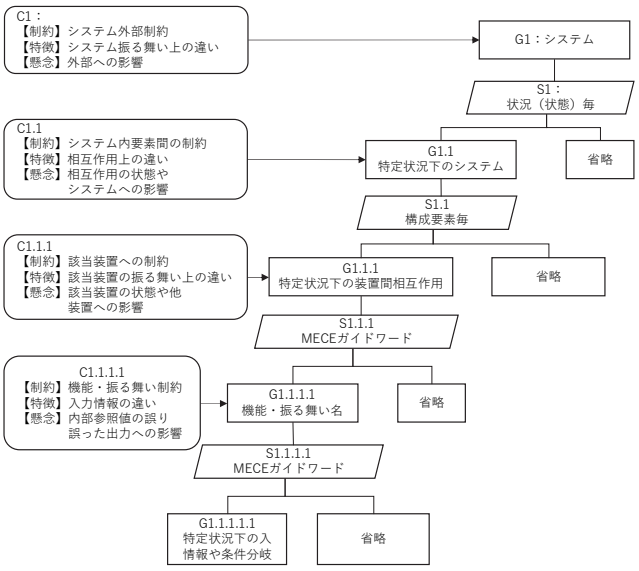

図 6：前提付き階層型知見

各階層のゴールの展開順序は，ゴールが「構成要素」のストラテジーで展開された場合は，上位から下位アーキテクチャへの展開とし，ゴールが「状態」で展開された場合は，時間が進む順序とする（時間が戻る「例：要因毎」の展開はしない）．また，原則，構成要素と振る舞いの交互に展開する．構成要素の展開が連続すると状態が抽出されず検査式の作成が困難になるため，非推奨としている．

＜同位ゴール間の網羅性を確保して構築＞

ストラテジーは，ストラテジーの展開パターンが推奨されている[21]. 本手法では, 設計対象の要素と状態を網羅的に抽出することを目的としているため，ストラテジーは，MECE（Mutually Exclusive, Collectively Exhaustive）であること, 設計対象の 1 つの要素に適用していること, とする．よって，対義語（例：最大と最小，双方向と一方向等）はストラテジーとしてそのまま適用できる．HAZOP(Hazard and operability studies)[22]によるガイドワードを用いる場合，ゴールにある設計対象に合わせて選択する必要がある．また， STAMP/STPA （System Theoretic Accident Model and Processes/ System-Theoretic Process Analysis）[23]にある「操作」に対するガイドワード（早過ぎる，遅すぎる，誤る，来ない）は，ゴールにある設計対象が制御上の「操作」であればそのまま適用できる．

3.1.2 前提付き階層型知見から検査対象への最適化

検出困難な事象及びシステム影響度が高い事象を検出できる検査式は，検査対象であるシステム内の複数の要素とその状態を含み，且つ，システムの状況とその状況下で影響度大となる出力情報が含まれている必要がある．

また，反例解析時に問題となるかどうかは，入力側の情報が実際の振る舞いとしてありうるか判断しているため，検査式の前提条件が明確であることも重要である．

上記を満たす検査式を作成するためには，システムから SW まで同じ因果にある設計情報を抜粋する必要がある．但し，V 字モデルの開発プロセス等において，システム，コンポーネント，アルゴリズム等の前提条件は，別々の文書に記載されている，又は暗黙となっていることが多く，1 つの因果関係がある検査式を作成する難易度は高い．一方，3.1.1 項で構築した前提付き階層型知見は，下位層にあるゴールは，上位ゴールにある前提条件を満たして作成しているため，上位から下位のゴールまでルートは 1 つの因果関係がある．そのため，最上位ゴールから末端ゴールまでのコンテキスト情報を組み合わせた結果（コンテキスト継承）は，システムから SW までの 1 つの因果関係があることになる．また，前提付き階層型知見のストラテジーが MECE となるガイドワードで作成されるため，コンテキスト継承となる因果関係に重複はなく，末端ゴールノードの数だけ因果関係を抽出できる．コンテキスト継承は，複数の制約を含んだ因果関係である「仕様外」に相当し，検証価値が高い検査項目である．しかし，前提付き階層型知見のコンテキスト情報は，実際の検査対象と前提が不一致等で有効ではない知見が含まれていることや，検査対象から状態遷移系を特定する単位に書かれていないこと，といった課題が発生する．よって下記の工程で検査対象に最適化した「検査式作成ガイド」を生成する．（図 7）

・工程 1：有効な知見の識別
・工程 2：知見を検査対象の情報単位に分解
・工程 3：1 つの因果関係を確保しつつ合成

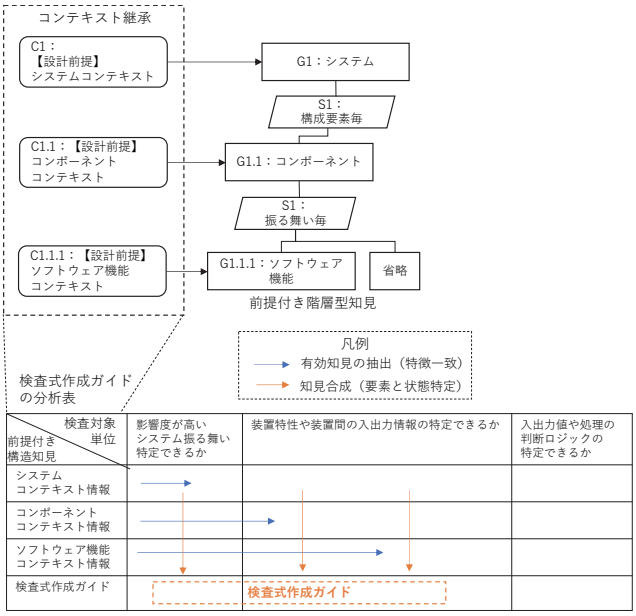

図 7：コンテキスト継承単位で検査式作成ガイドの作成

工程 1 の有効な知見の識別は，前提付き階層型知見のコンテキスト情報にある特徴と検査対象の特徴が一致しているかによって確認する．

工程 2 の知見を検査対象の情報単位を分解する場合，下記の視点で分解する．

・システム影響度が高い状況の特定できる情報か
・入出力構造図上にある要素及び状態を特定できる情報か
・入出力構造図上にある入出力情報を特定できる情報か
・入出力判断ロジック上の条件式を特定できる情報か

工程 3 の 1 つの因果関係を確保しつつ合成は，コンテキスト継承の単位で検査式作成ガイドを作成で達成できる．

3.1.3 検査式作成ガイドを用いた検査式の生成と絞り込み

3.1.2 項で作成した検査式作成ガイドを用いて，検査対象のうち該当する入出力値の組合せや判断ロジックの条件値を具体的に抽出し，検査式を作成する．本方法で作成される検査式は，特定状況下での複数のコンポーネントの不具合となりうる状態値であるため，検査式の基本表現は，いつか特定状態（EF）になる，と，AND 条件が基本となる．

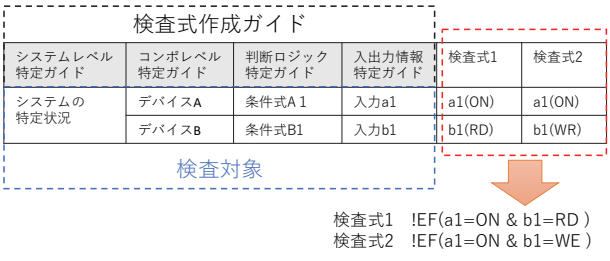

図 8：段階的な検査式作成ガイドの適用

前提付き階層型知見及び検査式の作成ガイドは，下記の観点で，状態爆発問題を防ぐモデルの絞り込みに活用する．

・絞り込み観点 1（シナリオ上の独立性）：GSN 上位層
システム外部の始点からシナリオとして該当入力が同時に発生しない等の前提で，開発上の検証価値が低下しない場合，該当のシナリオの入力や条件式等で区切り一度に探索する範囲を絞る．

・絞り込み観点 2（コーナーケースの指定）：GSN 下位層
システム内の要素間で並列性，非同期，判断条件が変わる等といった場合，該当する条件分岐群や入出力の相互作用等のみに絞る．

また，本手法は，源泉（レビュー結果や不具合情報）から検査式までトレーサビリティが確保され，検査式の意図や背景の把握が可能となるため，下記の観点で議論することでより検証価値が高められる．

・仕様外の抽出（ドメイン知識保有者と議論）
・検査対象の特性把握（現在の設計者と議論）
・検査式への変換箇所（モデル検査エキスパートと議論）

3.2 モデルの作成方法

システムへの影響度が高い問題点を検出するため，システムの入出力関係を表現するシステム入出力構造図と，重要な入出力を判断している判断フローから，モデルを作成する方法を構築する．図 9 にモデル作成時のインプットと工程を示す．

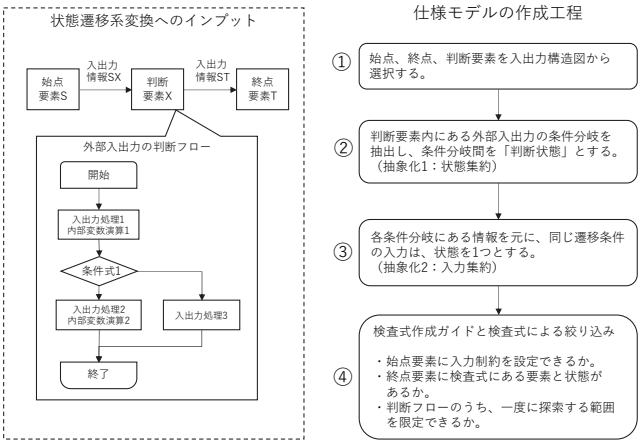

図 9：モデル生成時のインプットと作成プロセス

工程①では，システムの入出力構造図上で，始点要素，判断要素，終点要素の 3 つに分類する．

始点要素とは，モデル検査で保証したい前提を表現する要素であり，外部環境を模擬したセンサーの取り得る値，ユーザからの入力等が該当する．始点要素は，外部異常を模擬するため状態遷移をランダムにする．始点要素は，，工程④で偽反例を修正するため複数要素の入力制約を追加する対象である．終点要素とは，モデル検査で検出したい結果を表現する要素であり，アクチュエータへの出力値や排

他性のあるリソース等が該当する．終点要素の状態値は，検査式に入っているか照合し，検査式の範囲とモデルの範囲が合っているか確認する．

判断要素とは，始点要素から終点要素までの入出力情報を伝達する上で中間にある要素であり，入力に対して出力の判断を行っている．検査対象となるシステムは，複数の判断要素が直列及び並列に多段の構成となっており，システム全体をモデル化すると状態爆発が発生しやすくなる．また設計時に検証したい判断要素（検証要素）と設計前提となっている判断要素(前提要素)に分けることができる．

工程②と③でモデル検査の実行時に状態爆発を防ぐため，検査対象を抽象化する場合，設計者がモデル化された範囲が検証の目的に合致しているか，つまり抽象化によって実際の振る舞いとの乖離している内容を把握できる必要がある．また，システム影響度の高い問題点の検出を対象とする本手法では，システムの外部と内部や内部要素間の相互作用に着目しているため，下記の 2 つの方式とする．（図 10)

・前提要素は，その出力（検証要素への入力）のうち，同じ遷移条件と見なして複数の出力を 1 つの出力に集約してモデル化する（入力集約）

・検証要素は，フローチャート上にある全ての処理ではなく，入出力の判断が変わる条件式に関する処理や該当する条件式の区間を状態として集約しモデル化する.(処理集約)

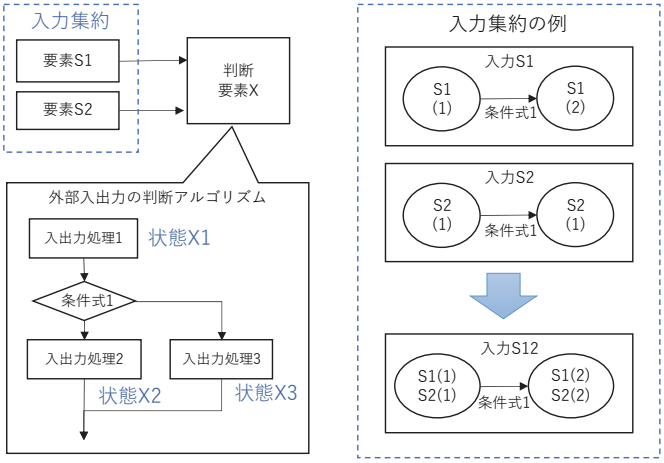

図 10：抽象化による集約パターンの概説

3.3 反例解析ビューの作成方法

3.3.1 反例解析ビューのモデリングルール生成

モデル検査器から出力された結果はモデル化された変数の推移が文字列として表示されるため，設計者の読み取り負荷が高い．そのため，モデル検査器の出力結果から．設計者が設計対象の修正が必要かどうか判断しやすいビューへ変換するモデリングルールを構築する．反例解析ビューのモデリングルールの生成は,構造図上の入出力関係と，修正すべき問題であるかどうかの判断を，要素種別毎にどのような状態値で判断しているのか，次にどのような要素間の関係で判断しているのか，によって確定させる．

まず，設計者が要素の種別毎にどのような状態情報で問題点の有無を判断しているか，確定させる．（例：2 値信号は ON/OFF/不定値，データ信号はビット列等.）

次に設計者がどのような要素間の関係で問題点の有無を判断している場合，入出力構造図上の入出力制約のパターンに分けて反例解析ビューの基本的なモデリングルールを決める．（図 11）

入出力制約のパターン 1 は，検出すべき制約違反が特定の要素に限定できる場合である（例：排他性のあるリソースの競合）．問題点であるかの判断は，入力元の要素が該当のリソースを使用することが必須であるか判断する必要があることから処理内容まで明らかにする必要があるため，入力要素の内部処理を反例解析ビューに含める．

入出力制約のパターン 2 は，要素間が直列関係であり入力順序で制約違反が判定可能な場合である．(例：複数要素間の相互作用）問題点であるかの判断は，予め規定された入力順序や特定状態での入力で判断できるため，入力順序を中心として必要最小限の状態を反例解析ビューに含める．

入出力制約のパターン 3 は，要素間が並列関係であり，各要素の状態で制約違反であるか判定可能な場合である．（例:特定の入出力の値が同時に成立してはならない場合）．問題点であるかの判断は，全ての要素と状態値を反例解析ビューに含める．

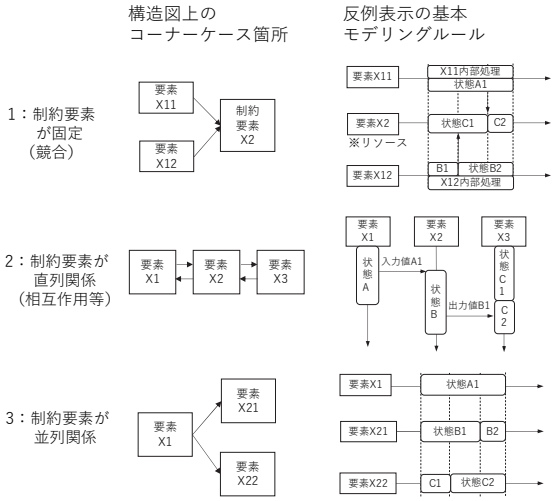

図 11：反例解析ビューの基本的なモデリングルール

3.3.2 偽反例に対するモデルの修正（制約の抽出）

実際の振る舞いと一致しない「偽反例」の検出要因は，入力や処理を集約し過ぎてしまう（モデル上の状態漏れ）場合と，入力上の暗黙的な制約が設計成果物から読み取れない場合（モデル上の遷移条件漏れ）がある．（例：読み込み信号と書き込み信号は同時に受信しない等.）

前者である入力や処理を抽象化することで集約し過ぎて発生した偽反例は，3.2 項で示すフローチャート上にある入力要素や判断要素を具体化すればよい．

後者の暗黙的な制約は，設計者が入力間の制約を新たに定義できる仕組みが必要である．暗黙的な制約定義は，新たなビューの学習といった技術者への負担を軽減するため，構築済の反例解析ビューのモデリングルールに付加情報を加える方法を検討する．暗黙的な制約を反例解析ビューへ反映する観点は，「制約パターンの抽出観点」と呼ぶ．制約パターンの抽出観点は，制約を課す要素（制約元要素 A）と制約を受ける要素（制約先要素 B）と，各要素の 3 つの遷移条件（1：状態への到達時，2：状態の保持，3：状態からの遷移）とした場合，制約先要素と制約元の各に 3 つの遷移条件を組み合わせて 9 通りの制約パターンの抽出観点を得る．（例：A1-B1）例えば，反例解析ビューが入出力制約パターン 1 だった場合，制約先要素 B は入力要素 X11 と X12 であり，制約元要素 A はリソースである X2 となり，制約パターンの抽出観点は，「X2 が制約違反（リソース競合時）に X11 や X12 は必ず遷移しなければならない（暗黙的な）状態はあるか？」となる．

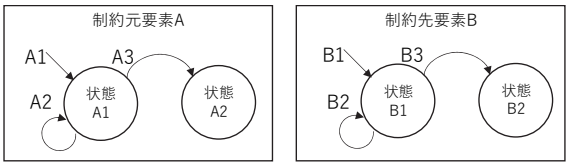

図 12：偽反例に対する制約抽出観点の対象

表 1．偽反例に対する制約抽出観点の例

制約パターンの抽出観点			入出力制約パターン1（例：リソース競合）に対する制約抽出パターンの適用例
No	制約元要素	制約先要素	
A1-B1	該当状態へ到達時	遷移すべき状態	リソースの競合が発生時、要素Bが必ず遷移しなければならない状態は何か？
A2-B3	該当状態を維持	該当状態以外へ遷移	リソースが競合中、要素Bは現在の状態以外へ遷移しなければならないか？
A3-B2	該当状態から遷移時	該当状態を維持	リソースの競合が解消時、要素Bは現在の状態を維持しなければならないか？

4. 適用事例

4.1 バス制御の FPGA システムの概説

提案手法が妥当かどうかを実際のシステムを対象に確かめることを目的として，バス制御 FPGA システム（図 13）の適用事例を以下に示す．設計対象は，FPGA 論理部であり，外部装置と内部装置のインターフェースを担っている．その際，各装置との通信によって共通バス内で信号の衝突が発生しないよう，タイミング設計を行っている．その際，2.3 項に示すように，複数のプロトコル切り替え，異なる特性の装置の切り替え等のコーナーケースやソフトエラー対策等によって「仕様外」に対する検証が発生する．

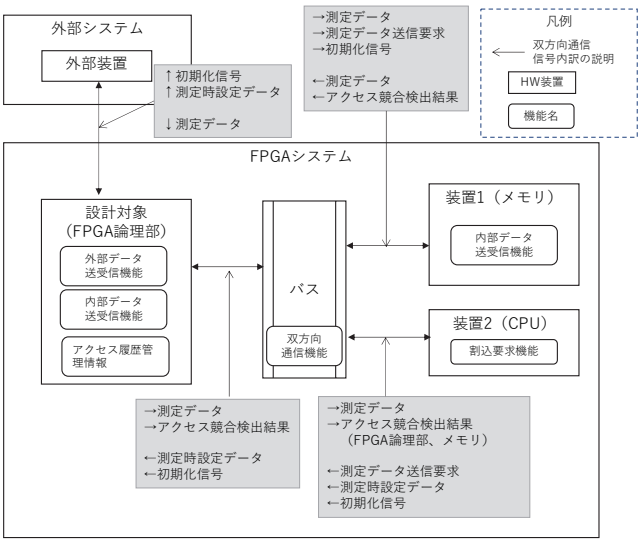

図 13：適用先の FPGA システム例

4.2 検査式の作成方法の例

設計のレビュー結果に対して，知見抽出観点（3.1.1 項）を適用し，前提付き構造型知見の例を図 14 に示す．

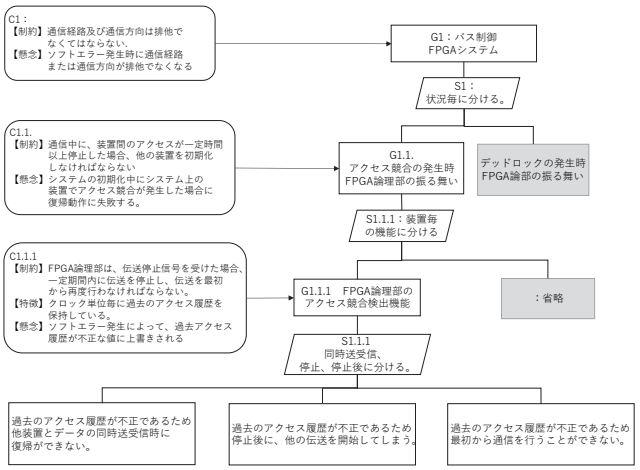

図 14：前提付き構造型知見の抜粋

次に前提付き構造型知見からコンテキスト継承の単位で検査式作成のガイドを作成する．図 15 は，検査対象に合わせた検査式作成ガイドで検査式を作成した例である．

	システムレベル	コンポレベル	判断ロジック	入出力情報	検査式1	検査式2
検査式作成ガイド	システムの初期化中に動作する通信は？	双方向通信のアクセス履歴を保管するデバイスであるか？	中断時、アクセス順序を決める処理であるか？	アクセス履歴を管理する信号であるか？		
検査対象	X通信によるハングアップ	アクセス履歴を初期化するCPU	全装置間のアクセスが停止し、かつ各装置が正常なアクセスを再開する条件文	各装置からのアクセス競合検出信号(_INT_ERR)	ON	
				・CPUから各装置への初期化信号(_RESET)	ON	ON
		アクセス履歴保持する装置	アクセス履歴が上書きを判断する条件文	アクセス履歴信号(_STATE_PAST)	RD	WR
				FPGA論理部のアクセス状態(_STATE_AHBL_IF_STATE)	RD	WR

検査式1　!EF(INT_ERR=ON & RESET=ON & STATE_PAST=RD & STATE_AHBL_IF_STATE=RD)

検査式2　!EF(RESET=ON & STATE_PAST=WR & STATE_AHBL_IF_STATE=WR)

図 15：検査式の作成例

4.3 モデルの作成方法の例

モデル作成時に追加の工数を発生させないようにするため，3.2 節で提案したモデル作成手段に基づき，図 16 に示す既存の設計成果物（FPGA バス通信システムの入出力関係を示す回路ブロック図や FPGA 入出力 IF ステートマシン等）の情報から，SMV コードで表現されたモデルを自動生成する仕組みを構築した．

設計対象である FPGA 論理部の振る舞いは，ステートチャートとして表現されるため，そのままモデルへ変換できる．一方，外部の入出力は回路ブロック図から論理素子の振る舞いや入出力関係を結線から読み取る必要があるため，予め論理素子の種類に応じたモデルパターンを準備し，回路ブロック図から，デバイスの種類，入出力関係の識別を行うことでモデルの自動生成を実現している．

また，技術者が，状態爆発防止のために実際の動作と比較して何を省略しているか，把握できることが重要である．そのため，抽象化は 3.2 項で示す処理集約を行う（入出力の判断を行っている条件分岐の境界値の単位でモデル上の変数を定義する）．また，3.1.3 項で示す前提付き構造知見を用いた絞り込みを行うため，フローチャート形式上で，モデルを生成する区間を指定できるようにしている．

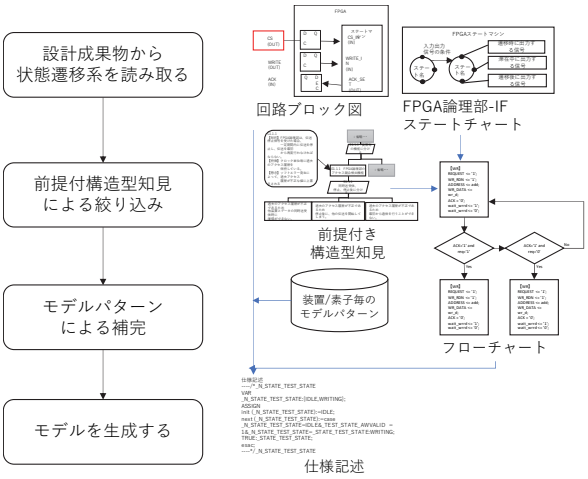

図 16：設計成果物からモデルの自動生成

4.4 反例の検出方法と偽反例防止方法の例

技術者がモデル検査の出力結果を参照し，設計上の問題点であるか判断しやすくするため，3.3.1 項に示す要素種別毎の状態情報を図 17 に示す属性と属性値とした．入出力構造上の要素間は並列関係であり，特定区間における各要素の状態で制約違反であるか判定するため，入出力制約のパターン 3 (図 11)とした．なお，FPGA 論理部はクロックに応じて各信号等の属性値が変化するため，モデル検査器の出力結果を直接，図 18 のように変換する方式とした．

属性	属性値	タイミングチャート上 1タイムスロットの表記
クロック	N/A	
信号	信号あり （上線で表現）	
	信号がONへ変化 （同期信号）	
	信号がONへ変化 （非同期信号）	
FPGA-IF ステート マシン	ステートマシン名	状態名
	次のステートへ 遷移開始	状態名

図 17：要素種別毎における状態値の定義例

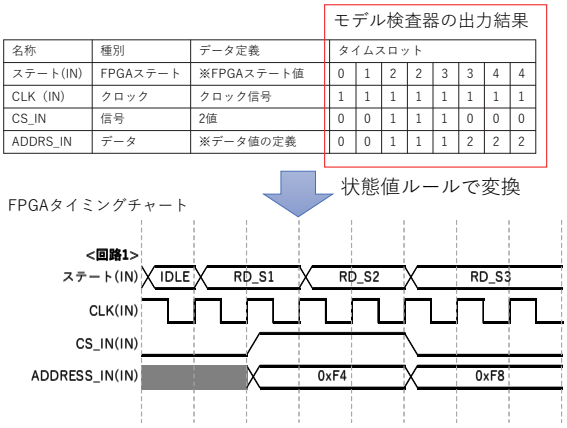

図 18：反例解析ビューの実装例

また，図 18 で出力された反例が偽反例となっている場合，モデルの始点要素（3.2 項）の遷移条件に追加することで偽反例の検出を防ぐ．本事例では，始点要素は回路ブロックス上にあるシステム外から入力される信号が該当する．追加する遷移条件は，設計者が検査結果であるタイミングチャート上で偽反例である入力信号の関係を制約として特定する．本事例における入力信号の制約は，表 1 に示す「偽反例に対する制約抽出観点」を FPGA システムに対する 2 つ入力信号を基準信号と従属信号として適用し，制約パターンを定義した（図 19）．

No	抽出観点	偽反例防止の制約パターン
1	A2-B2	基準信号のアサート期間中は、従属信号がアサートでなければならない
2	A3-A3	基準信号のアサート/ネゲートにより、従属信号をアサート/ネゲートする

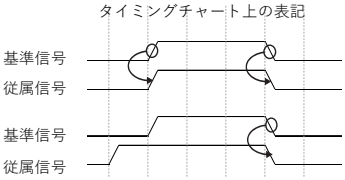

図 19：偽反例防止による制約抽出の例

5. 有効性評価と考察

4.2 項に示す本手法の有効性を 4.1 項に示す FPGA 論理設計に適用し，下記の RQ と指標で評価を行った．RQ1：仕様外を含めた検証価値が高い検査式が作成できるか，RQ2：検査式は，影響度が高く検出が困難な問題点を検出

できるか．評価の条件は，FPGA 設計者として経験が浅く上位の FPGA 技術者の指導を必要としている技術者，かつモデル検査技術を保有していない技術者（検査式の作成や SMV コードの実装が未経験）が，まず提案手法なしで検査式の作成を行い，その後，同じ技術者が同じ回路に対して提案手法を用いて検査式の作成を行った．そのため，提案手法で作成された検査式や検出した問題点は，方法なしの検査式や問題点に対して全て新規追加となる．

RQ1 : 仕様外を含め検証価値が高い検査式が作成できるか，

従来の検証活動（設計時にタイミングチャートの作成による検証と試験ケースの設定のこと）において，検査式は検証価値があるか評価するため，表 2 に示す評価指標を設定した．検出難易度の水準が 2 以上であれば，タイミングチャートの作成や試験ケースの設定は高い属人性が発生し，検証漏れによる不具合が発生しやすくなるため，検証価値がある検査式と言える．図 20 はその計測結果である．

表 2．RQ1 に対する評価指標

水準	設計時の検出難易度	試験時の検出難易度
1	単一仕様を分析したタイミングチャートで検出可	試験ケースとして設定可
2	複数仕様を分析したタイミングチャートで検出可	試験ケースとして設定できるか，ケース数が膨大
3	仕様外を想定し，タイミングチャートで検出可	試験ケースを設定できない

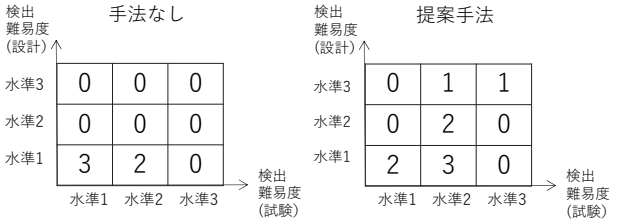

図 20：RQ1 に対する計測結果

RQ2：影響度が高く検出が困難な問題点を検出できるか，

RQ1 で作成した検査式に対して，基本設計中である実際の設計成果物に実行し，表 3 に示す評価指標で検出難易度と影響度で評価を行った．図 21 はその計測結果である．

表 3：RQ2 に対する評価指標

水準	検出難易度	影響度
1	システム内部の単体装置の振る舞いで検出可	システムに影響はない
2	システム内部の複数装置の振る舞いで検出可	システムの一時的な障害になる
3	システム外部と内部の振る舞いで検出可	システムの永続的な障害になる

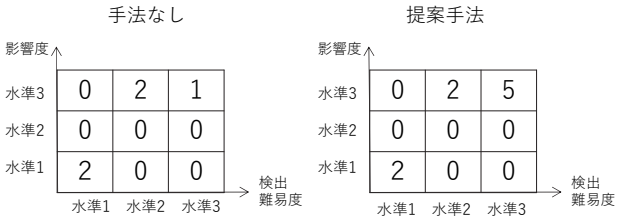

図 21：RQ2 に対する計測結果

手法ありと手法なしで導出された検査式の特性に対して，システムの前提条件（外乱や利用条件等）が特定されているか，装置間の相互作用が特定されているか，装置の制約が特定されているか，の 3 点で評価した結果を表 4 に示す．

表 4：検査式の特性

評価指標	手法なし	提案手法
システムの前提条件が特定されているか	0	2
装置間の相互作用が特定されているか	3	7
装置の制約が特定されているか	1	4

仕様外を考慮した検査式は，システムの前提条件が特定されている検査式に該当しており，且つ，検出した問題点はシステムの永続的な障害になる問題点であり，前提付構造型知見にあるシステムから設計対象までの前提条件を組み合わせによって検査式を作成することが効果的であること示している．

また，4.3 項と 4.4 項に示すモデルや反例解析の作成支援ツールによって，技術者はモデル検査技術の習得することなく，設計上の問題点まで提示できた．また，従来行っている設計検証としてのタイミングチャートの作成工数に対して，優位である．なお，従来の設計検証とは，同じ回路に対してより上位の設計者が作成した際の平均時間である．よって，モデルや反例解析の作成支援ツールは効果的であると考える．

手法なしと提案手法による工数比較では，提案手法の工数は 15%増加しており，前提付き階層型知見の読み取り負荷が加わっている．一方，従来の設計検証に対しての削減率として，方法なしが 71%減，提案手法が 64%減となり，一定の削減効果は確保できている．

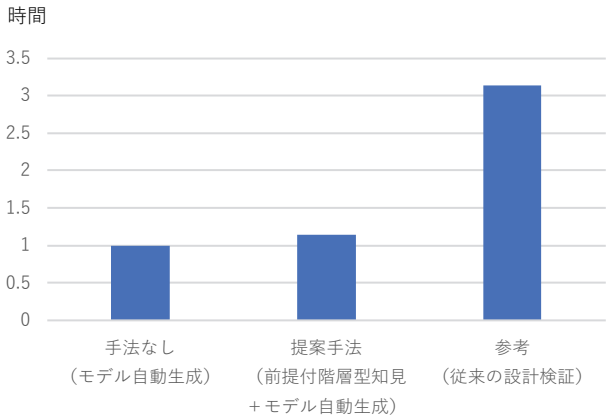

図 21：1 タイミングチャート当たりの作成工数

但し，モデル検査による追加コストを負担することに相応する検証価値を保持できているか，は，システム全体の検証コストと比較したデータが必要であり，今後の課題である．

また，提案手法の適用対象とした FPGA 論理設計は，論理素子として動作するためパターン化されたアルゴリズムと設計者が定義する入出力の判断アルゴリズムが明確であるため，設計情報からのモデルの読み取りやモデルパターンを構築しやすい．また反例解析ビューについても，CPU クロックを基準に対して各要素の状態を定義すればよいので，構築しやすい側面がある．

今後の課題として，提案手法は，FPGA の上流設計のみの適用に留まっており，他の特性を持つシステムにも提案手法でモデル検査を用いた検証方法を構築し，モデル検査による追加コストを負担することに相応する検証価値を保持できているか，もしくは高めることができるか，実証していく．

6. 結論

本論文では，モデル検査を用いた検証方法の検証価値を高めるため，検出困難且つ影響度の高い不具合検出を支援する方法を提案した．提案手法を用いて FPGA 論理設計向けの検証方法を構築し，非熟練者による有効性評価を行い，有効性の確認はできた．今後，他の特性があるシステムに対して検証手法の構築と有効性の評価を行う．

謝辞 本研究の実証実験に終始，ご協力およびご助言を頂いた 株式会社 第一システムエンジニアリング 第 2 技術部 杉本雄茂様，伊藤誠様，新井利彦様，光田和弘様，岩瀬裕昭様へ謹んで感謝の意を表する．

参考文献

[1] Oxford University Press, A Dictionary of Computing, 2004.
[2] IEEE Systems and software Quality Requirements and Evaluation (SQuaRE) − System and software quality model, ISO/IEC 25000, 2014.
[3] 独立行政法人情報処理推進機構，つながる世界のソフトウェア品質ガイド，第一版, p. 208, 2015.
[4] E.M.Clarke, O. Grumberg, D. Kroening, D. Peled, H. Veith, Model Checking, MIT Press, 2000.
[5] 独立行政法人情報処理推進機構，独立検証機関による形式手法を用いた第三者検証のコスト評価実施報告書, p. 60, 2013.
[6] 吉岡信和，田辺良則．田原康之，長谷川哲夫，磯部祥尚，モデル検査による設計検証，コンピュータソフトウェア vol. 31, no. 4, p. 40-65, 2014.
[7] 本位田真一，萩谷昌己，吉岡信和，青木利晃，田原康之, SPIN による設計モデル検証．近代科学社, p.226, 2008.
[8] 後藤隼弐，吉岡信和，シーケンス図を用いたモデル検査支援ツール．コンピュータソフトウェア, vol. 32, No. 4, pp.50-73, 2015.
[9] 佐藤直人，芹沢 一，前田 浩光，高橋 伸明，前川 義之，四野見 秀明，村尾 直哉，大島 豊，田代 敦，黒川 勇，八木沢 育哉，高田 豊，清永 章彦，大橋 正己，偽反例の出力を抑制するモデル検査支援環境，ソフトウェアエンジニアリングシンポジウム 2016, p. 211-218, 2016.
[10] 加藤淳，狼嘉彰，モデル検査における安全性プロパティの系統的な導出手法，情報処理学会研究報告, Vol.2011-SE-171 No.25, 2011.
[11] 乾道孝，吉岡信和，落水浩一郎，ゴール指向分析に基づくモデル検査のための外部環境の抽象化手法，情報処理学会誌 vol. 52, no. 12, p. 3205-3220, 2011.
[12] Goal Structuring Notation Working Group, GSN Community Standard Version 1, 2011.
[13] T. Kelly, R. Weave, The Goal Structuring Notation: A Safety Argument Notation, Proc. of 2004 Workshop on Assurance Cases the Dependable Systems and Networks, p. 6, 2004.
[14] DEOS 協会 D-Case 部会, D-Case 構文定義書, D-Case, http://www.dcase.jp/
[15] 梅田浩貴，波平晃佑，祖川和弘，植田泰士，片平真史, GSN 及び ESD モデルを用いたソフトウェア FEMA の提案，ソフトウェア品質シンポジウム 2018 報文集, 2018 .
[16] XILINX 社, AXI リファレンスガイド(UG1037(v3.0)), p.27-p.42, XILINX 社, 2015.
[17] 電子情報通信学会 知識ベース, 6 群(コンピュータ・基礎理論とハードウェア)-- 7 編(ディペンダブルコンピューティング)-- 3 章 設計検証/テスト(ver.1), p. 13 - p.16, 2010.
[18] ITU-T, K.131-Soft Error Measures of Field Programmable Gate Arrays, https://www.itu.int/rec/T-REC-K.Sup11/en, 2018.
[19] ARM ホールディングス, ARM. AMBA AXI and ACE Protocol Specification(IHI 0022D(ID102711)), p.19-p.100, ARM 社, 2011.
[20] M. Fujita, I. Ghosh, M. Prasad, Verification Techniques for System Level Design, Morgan Kaufmann Publishers, p.163-p.230, 2007.
[21] S.Yamamoto and Y.Matsuno, An evaluation of argument patterns to reduce pitfalls of applying assurance case, 2013 1st International Workshop on Assurance Cases for Software-Intensive Systems (ASSURE), 2013
[22] IEC 61882, Hazard and operability studies (HAZOP) studies application guide, 2016.
[23] N. Leveson, Engineering a Safer World: Systems Thinking Applied to Safety, MIT Press, 2011.