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

2007－ＳＬＤＭ－128（14）
2007／1／18

即時通信を行うハードウェアのサイクル精度動作記述モジュール群に対するモデル検査の一手法

藤田 裕久† 濱田 雅彦† 谷本 匡亮† 中田 明夫† 東野 輝夫†

† 大阪大学大学院情報科学研究科
〒 565-0871 大阪府吹田市山田丘 1-5
E-mail: †{h-fujita,tanimoto,nakata,higashino}@ist.osaka-u.ac.jp, ††m-hamada@ics.es.osaka-u.ac.jp

あらまし ハードウェアには配線遅延があるためクロック周波数の向上に限界がある．そこで，実行時間削減を目的としてクロックサイクルを消費しない即時通信がしばしば用いられる．そのような通信を行うサイクル精度動作記述には，組み合わせ回路としての閉路が発生することに起因する値の発振や発散の可能性といった難しさがある．そこで，本論文では wire による即時通信を行うサイクル精度動作記述モジュール群に発振や発散の問題が生じるか否か等を，モデル検査によって検証する手法を提案する．実験では上述した問題の発生を含んだ簡単なテストケース群に対して提案手法を適用し，その有効性を確認した．

キーワード 設計検証，サイクル精度，即時通信，組み合わせ閉路，モデル検査

Model Checking of Cycle Accurate Hardware Behavior Models with Instantaneous Communication

Hirohisa FUJITA†, Masahiko HAMADA†, Tadaaki TANIMOTO†, Akio NAKATA†, and Teruo HIGASHINO†

† Dept. of Information Science and Technology, Osaka University
Suita, Osaka 565–0871
E-mail: †{h-fujita,tanimoto,nakata,higashino}@ist.osaka-u.ac.jp, ††m-hamada@ics.es.osaka-u.ac.jp

Abstract Wiring delay imposes a limitation on increase of clock frequency. Therefore, instantaneous communications consuming no clock cycles are sometimes used. Cycle accurate behavior models allowing such communications have a problem of oscillation and divergence in variable values caused by combinational loops. In this paper, we propose a model checking method of cycle accurate behavior models including instantaneous communications of wire. Proposed method can detect the occurrence of oscillation, divergence and other problems like access conflict. In the experiments, we confirmed the effectiveness of our method using several simple test cases.

Key words instantaneous communication, constructive analysis, combinational loops, cycle accurate behavior model, model checking

1. はじめに

ハードウェアにはゲート遅延や配線遅延があり，実行時間削減を目的としたクロック周波数の向上には限界がある．また，個々の処理にかかるサイクル数を減らすためには並列度を上げる必要があり，面積増加などの問題が生じる．そこで，通信にかかるサイクル数の削減がしばしば行われる．通信サイクル削減に有効な通信機構の一つに，クロックサイクルを消費しない即時通信がある．

クロックサイクルを消費する通信としては，SystemC 等で用いられる Signal 通信がある．Signal 通信とは，サイクル精度レベルの記述において，信号を送出したサイクルの次サイクル以降で受信可能となる通信機構である．一方，即時通信は信号を送出したサイクルから受信可能な通信機構である．即時通信が記述可能な言語として Esterel [2] が挙げられる．Esterel の即時通信では，信号を送出したサイクルのみでそれが受信可能である．ハードウェアの設計において，Signal 通信と Esterel の即時通信の両方の特徴を取り入れた通信機構が使われること

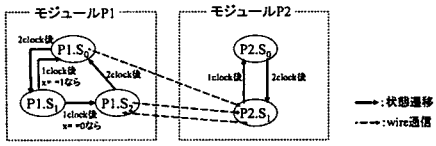

図 1　モジュールの動作と wire 通信

がある．この通信機構では，信号送出を行ったサイクルと同一またはそれ以降のサイクルで受信可能である．したがって，クロックサイクルを消費せずに通信できるが受信のタイミングを送信に合わせる必要がないという長所を持つ．以降では，この通信機構を wire 通信 [11], [12] と呼ぶ．wire 通信は概念的にはモジュール間共有変数である wire 変数への読み書きを通じて実現される．

即時通信の難しさとして，組み合わせ閉路が存在することに起因する値の発振や発散の可能性といったことが挙げられる．そのような問題の解析手法は，主にゲートレベルを対象にして研究されてきた．[9] が組み合わせ回路における発振の検出手法を初めて提案し，その後，順序回路への拡張手法 [10] やより効率の良いアルゴリズム [4] が考案されてきた．また，他のアプローチとして，組み合わせ閉路を持つ回路を，それと等価でかつ組み合わせ閉路を持たない回路に変換する手法 [6] もある．しかし，これらのようにサーキットレベルを対象とした手法では解析を行う前に，ハードウェアの動作記述をネットリストへ変換するという高コストな手順が要求される．そこで，[8] ではブール関数レベル記述に対して出力が安定するか否かを判定する手法を提案している．しかし，この手法はビット単位のモデル化が前提である．

同期式順序回路間で即時通信を行うようなサイクル精度動作記述モジュール群では，値が不定になるか否かが通信先モジュールの状態にも依存する．例えば，図 1 ではモジュール P1 が $P1.S_2$ に，モジュール P2 が $P2.S_1$ にいる時のみ組み合わせ閉路が発生し，値が不定になる可能性が生じる．特に省電力などのために利用されるマルチクロックシステムにおいては，モジュール群の状態遷移の実行系列と即時通信の動作を共に解析する必要があり，従来手法を適用することは難しい．

そこで，本研究ではモジュールをサイクル精度レベルで動作記述された同期式順序回路とし，モジュール間で wire 通信を行うようなハードウェアを対象とした検証手法を提案する．提案手法ではマルチクロックシステムの検証も可能である．なお，素子の遅延は全て 0 秒と仮定する．

提案手法では，そのような解析を実現するためにモデル検査 [5] を利用する．提案手法は，モジュール群のサイクル精度動作記述と CTL (Computation Tree Logic) [7] 等によって記述された検証性質を入力として受ける．検証性質の具体例としては「データ変数 x に対して，やがて変数 x==1 が成立する実行系列がある」といったことが挙げられる．出力としては，検証性質を満たすか否かを，また満たさない場合にはその反例も返す．提案手法を用いることにより，出力の発振や発散の検出

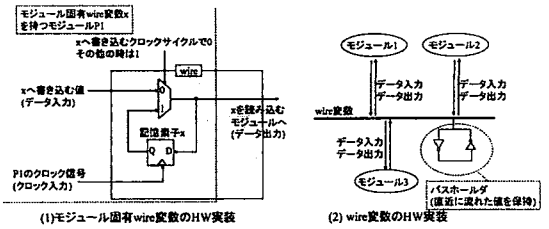

図 2　モジュール固有 wire 変数と wire 変数の実装

のみでなく，モジュールへのアクセス信号の状態を検査することでアクセス競合の検出などが可能となる．また，ビット単位のブール演算に変換することなく，ワード単位の演算を扱うことによって，変換および変換後の検証のコストを削減できる可能性がある．

モデル検査による検証を実現するためにはシステムの動作を状態遷移モデル群で記述する必要がある．そこで我々は，素子の遅延を 0 秒と仮定した場合，出力が定まるまで 0 秒の間に wire 変数の値の更新を繰り返すというモデルで wire 通信の動作を表せることに着目した記述手法を考案した．この手法は，検証対象ハードウェアで出力の発振が発生するときは状態遷移モデル群において zeno condition [3] が発生するという特徴を持つ．zeno condition とは時間が進まないまま無限回遷移が実行され続けるという状況を意味する．提案手法においては，zeno condition を検出できるような検証性質を与えることで出力の発振・発散を検出が可能である．

実験では，上述した問題の発生を含んだ簡単なテストケース群に対して提案手法を適用し，それが正しく動作することを確認した．

本稿の構成は以下の通りである．2. で wire 通信の動作について，3. で wire 通信を行う検証対象ハードウェアについてそれぞれ説明する．4. では提案手法の概要を述べる．5. では検証対象ハードウェアのモデル化手法を説明する．6. では実験結果を示す．最後に 7. でまとめと今後の課題に関する検討を行う．

2. wire 通信

本章では wire 通信について述べる．wire 通信とは，モジュール間を繋ぐ配線として動作し，かつ値を保持することもできるような素子を通じたモジュール間通信である．ここで，モジュールを同期式順序回路で構成されると仮定する．

まず，内部レジスタ変数を定義し，次に wire 通信で用いられる wire 変数を定義する．

[Definition 1]　**内部レジスタ変数**とは，データ入力，データ出力そしてクロック入力を持つ記憶素子である．**内部レジスタ変数を持つモジュール**とは，その内部レジスタ変数にクロック入力とデータを与え，データ出力を受けるモジュールである．□

内部レジスタ変数は，それを持つモジュールによってのみ値を読み書きされる．また，内部レジスタ変数の値の更新はクロック入力のエッジにおいて行われる．したがって，サイクル精度

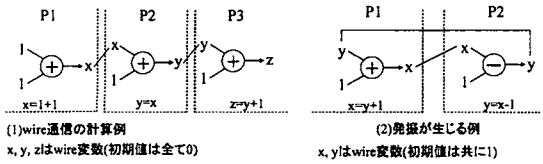

図 3　wire 通信の動作

レベルの記述において，内部レジスタ変数の値を更新はその次サイクル以降で反映される．内部レジスタ変数はモジュールの内部変数として動作する．

次に，wire 変数の定義のため，1 モジュールのみ書き込み可能であるという制限を持つ wire 変数 (モジュール固有 wire 変数と呼ぶ) について述べる．

[Definition 2]　**モジュール固有 wire 変数**とは，データ入力，データ出力，そしてクロック入力を持つ記憶素子である．**モジュール固有 wire 変数を持つモジュール**とは，それにクロック入力を与えるモジュールである．モジュール固有 wire 変数の動作は図 2(1) (注1)のように実装されるハードウェアの動作で定義される．　□

モジュール固有 wire 変数は，それを持つモジュールによってのみ値を書き込まれるが，全モジュールがデータ出力の値を読み込むことができる．内部レジスタ変数と同様に，モジュール固有 wire 変数が保持する値の更新はクロック入力のエッジにおいて行われる．ただし，読み込むモジュールは記憶素子に保持される前の書き込む値を常に取得できる．また，モジュール固有 wire 変数に値が書き込まれないサイクルにおいては，データ出力には記憶素子が保持している値が出力される．したがって，モジュール固有 wire 変数はサイクル精度レベルの記述において，値を更新するサイクルと同一サイクルで更新が反映され，それ以降のサイクルでも値が保存される．

最後に，一般の wire 変数を定義する．

[Definition 3]　**wire 変数**は任意のモジュールに対してデータ入力とデータ出力を与える記憶素子である．wire 変数の動作は図 2(2) のように実装されるハードウェアの動作で定義される．　□

モジュール固有 wire 変数とは異なり，wire 変数では全モジュールが書き込み可能となる．また，wire 変数はモジュールが書き込んだ瞬間からその値を保持する．

wire 通信では wire 変数の更新にクロックサイクルを消費しない．そのため，各モジュールは常に，内部変数の値と現在受け取っている wire 変数の値から wire 変数への出力を出し続ける．したがって，同時刻において複数モジュールから同一 wire 変数への書き込みと読み込みが同時に発生した場合，各モジュールは wire 変数の値が収束するまで更新を繰り返すことになる．その例を図 3(1) に示す．この例では，モジュール P1 (x=1+1 を計算)，P2 (y=x+1 を計算) と P3 (z=y+1 を計算) の存在

(注1)：図 2. は動作を表す一実装例であり，必ずしもこのように実装される必要はない

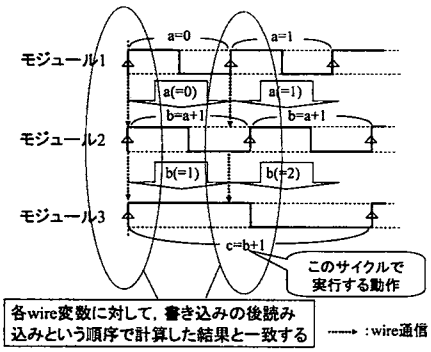

図 4　wire 通信のタイミングチャート

を仮定している．x, y, z の初期値は 0 とする．wire 通信では，読み込む wire 変数の現在の値から計算結果を出力するため，x, y, z の初期値を受け取っている間は P1 が x=2，P2 が y=1 そして P3 が z=1 を出力する．この更新を受けて，x の値が 2 となっている間 P2 は y=3 を出力する．同様にして，次に P3 が z=4 を出力する．その結果，x=2, y=3, z=4 で計算結果が収束し，これらの値を出力し続ける．

wire 通信の動作の特徴は，収束するまで上記のような更新を繰り返すことにより，必ず読み込みより書き込みが先行するという計算結果に収束することである．以上より，マルチクロックシステムにおける wire 通信のタイミングチャートは図 4 のようになる．

最後に，wire 通信では出力値が定まらないことがあることを図 3(2) の例を用いて説明する．この例において x, y の初期値は共に 1 とする．初期値を受け取っている間，P1 と P2 はそれぞれ x=2, y=0 を出力する．この更新を受けて，P1 と P2 は x=2, y=0 を受け取っている間それぞれ x=1, y=1 を出力し，初期値を受け取っている間の動作に戻る．これらの動作を繰り返すことで値が不定となる．

3.　検証対象ハードウェア

検証の対象とするハードウェアは，モジュールが同期式順序回路として動作し，それらの間で wire 通信を行うものとする(注2)．モジュールはクロックサイクル内では組み合わせ回路として動作するものとする．また，各モジュールは一般に異なるクロックを持つものとする．

次に，モジュールの動作仕様の記述クラスを説明する．モジュールの動作仕様は，C などの高級言語のハードウェア合成可能なサブクラスによってサイクル精度レベルで記述される．動作仕様が持つ変数には，内部レジスタ変数と wire 変数 の二つのみを仮定する(注3)．各変数の取り得る値の範囲は有限であ

(注2)：wire 通信が扱えれば，Signal 通信や Esterel の即時通信への拡張は容易に可能

(注3)：サイクル内で更新され，同一サイクルで更新された値を参照される内部変数は，それを含まない記述に等価変換できるため，ないものとして一般性を失わない

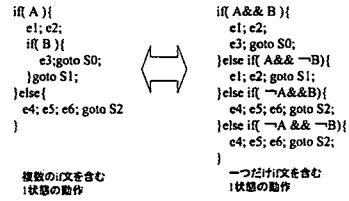

図 5　複数の if 文を一つの if 文へ変換する例

るとする．次に，動作仕様の 1 状態に記述することのできる動作を以下で定義する．

[Definition 4]　**モジュールの動作仕様の 1 状態に記述できる動作**は，任意の個数の条件分岐と任意の個数の代入文である．ここで代入文の右辺には四則演算からなる式のみが書けるとする．また，遷移先である次状態も指定できる．　□

本稿では簡単のため 1 状態に記述することのできる条件分岐として一つの if 文 (else if, else を含む) のみを考える．ここで，if 文を一つしか記述できないことは制限ではないことを説明する．まず，1 状態内で同じ変数へ 2 回書き込むという記述は無いものとする(注4)．このとき，1 状態で実行される代入文の集合は動作仕様において逐次的に記述されるが，実行は並列に行われる (同時処理文)．このとき，任意の個数の条件分岐を一つにまとめることで，等価な一つの if 文へと機械的に変換できる．その例を図 5 に示す．図 5 において，A, B は if 文の分岐における分岐条件式を，e1, e2,...,e6 は代入文をそれぞれ表す．

次に，1 サイクル内における代入文実行の動作を説明する．検証対象ハードウェアでは，1 サイクルの間，右辺に現れる変数の値からの演算結果を左辺に現れる変数に出力し続ける．2. で述べたように，wire 通信ではクロックサイクル内であっても wire 変数の値が変化する可能性がある．したがって，右辺に表れる変数の値が 1 サイクル内で変化した場合は，その変化に応じて新しい演算結果を出力し続ける．なお，演算は 0 秒で行われるものとする．

最後に，if 文の分岐評価が wire 変数の値に依存するときの動作を説明する．分岐評価が wire 変数の値に依存するときは，1 サイクルの途中で実行する代入文が変化する可能性がある．その例を図 6 に示す．図 6 では，モジュール P1 のクロックサイクルタイムを 1 秒，モジュール P2 のクロックサイクルタイムを 2 秒としている．P1 と P2 がそれぞれ状態 1 と状態 3 に同時に入ったとすると，その 1 秒後に P1 は状態遷移を行い a の値を 0 から 1 へ変化させる．その結果，P2 は 1 サイクル内の初めの 1 秒間と残りの 1 秒間で実行する代入文が変化する．

ここで問題となるのが，初めの 1 秒間で#2 を実行しているため wire 変数 c に 0 が保存され，後半 1 秒間も c=0 を出力してしまうことである．この場合，モジュールがクロックサイクル内で順序回路として動作してしまう．そこで，1 サイクルの途中で if 文の分岐が変化した場合は，それ以前に実行していた代

(注4)：任意の記述をこの制限を満たす記述に等価変換可能であるので，このように制限しても一般性を失わない．

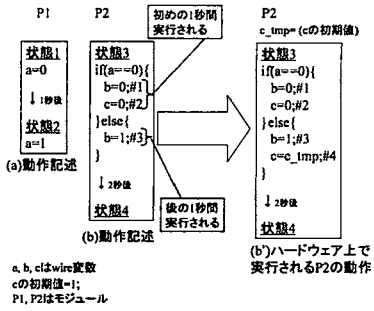

図 6　if 文のサイクル精度動作記述とハードウェア上でのその動作

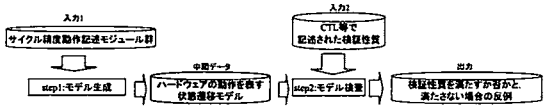

図 7　提案する検証手法の概要

入文の計算結果を自動的に**訂正**するものとする．ここで**訂正**とは，サイクルの途中で更新が行われた wire 変数の値を，クロックエッジにおいて保持されていた値に戻す動作である．図 6 では (b') の#4 がその訂正処理に該当する．

4.　提案する検証手法

4.1　問題定義

wire 通信を行うサイクル精度記述検証問題を，入力として与えられるサイクル精度動作記述と検証性質から，検証結果を出力する問題であると定義する．以下で入出力を詳細に説明する．

入力 1:サイクル精度動作記述　最初の入力として，検証対象となるハードウェアのサイクル精度動作記述を与える．その動作仕様は 3. で説明したものである．

入力 2:検証性質　もう一つの入力は，CTL (Computation Tree Logic) 等によって記述された検証性質である．検証性質の記述クラスは利用するモデル検査ツールに依存する．

出力:検証結果　出力は，入力 1 で与えたハードウェアの動作が入力 2 で与えた検証性質を満たすか否かと，満たさない場合の反例である．ここで反例とは，検証性質を満たさないような状態遷移の実行系列を意味する．

4.2　提案手法の概要

提案する検証手法の概要を図 7 に示す．提案手法は以下の二つのステップからなる．

step1　入力 1 として与えられる検証対象ハードウェアの動作を表す状態遷移モデルを生成する．生成された状態遷移モデルは提案手法の中間データとして働き，step2 へと渡される．

step2　モデル検査を実行する．提案手法で用いるモデル検査は，システムの動作を記述した状態遷移モデルと検証性質を入力として受け，状態遷移モデルが検証性質を満たすか否かと，満たさない場合の反例を返す．したがって，モデル検査ツールに step1 から渡された中間データと入力 2 の検証性質を与え，出力を得る．

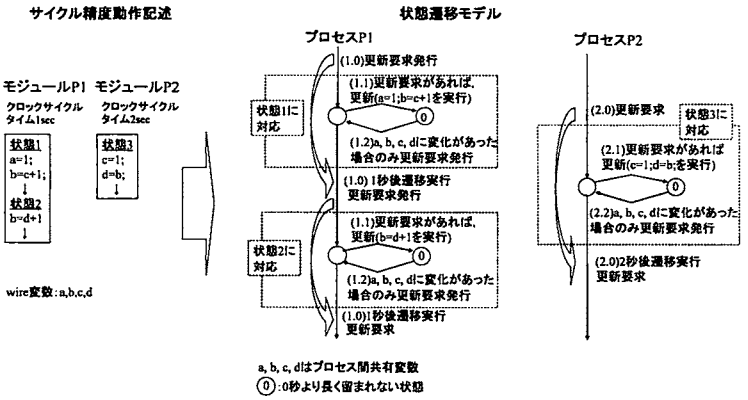

図 8　代入文のみからなるサイクル精度動作記述のモデル化手法

5. 検証対象ハードウェアのモデル化手法

本章では検証対象ハードウェアのモデル化手法を説明する．検証対象となるサイクル精度動作記述には if 文が含まれるが，まず，代入文のみを考えたモデル化手法を述べる．次に，if 文も含まれた場合のモデル化手法の説明を行う．なお，本章では簡潔な説明のために変数記述付きの時間オートマトンを利用するが，提案するモデルは整数のみを扱うため有限状態機械のクラスで記述可能である．また，本章ではモジュール P の動作を表す時間オートマトンをプロセス P と書く．

2. で述べたように，wire 通信では現在の wire 変数の値を受けて計算結果を出力するという動作を常に行っている．提案手法ではこの動作を，値が収束するまで計算を繰り返すというモデルで表す．ここで，図 8 に if 文を含まないサイクル精度動作記述を状態遷移モデルで表す手法の概要を示す．

この例では，モジュール P1, P2 はそれぞれ以下の処理を行う．

- P1 は状態 1(a=1;b=c+1;) に 1 秒間留まった後，状態 2 (b=d+1;) に遷移しそこに 1 秒間留まる．
- P2 は状態 3 (c=1;d=b;) に 2 秒間留まる．

サイクル精度動作記述における各状態は，図 8 のプロセスにおいて点線で囲まれた部分に対応している．そして，それらの点線部をつなぐ遷移 (1.0), (2.0) はサイクル精度動作記述における状態遷移に対応している．つまり，(1.0), (2.0) はクロックエッジの動作を表す．したがって，状態遷移モデルではクロックサイクルタイムの間点線部に滞在した後，クロックエッジを表す遷移を実行する．

次に，クロックサイクル内の動作を表す点線部について説明する．まず，更新要求について述べる．更新要求は，プロセスの動作を制御するためにモデル化時に新しく追加するプロセス間共有信号である．代入文の実行が必要なプロセスが一つでもあれば，そのプロセスに更新要求を発行させる．そして，それを全プロセスに受けさせ，代入文を実行させる．ここで，代入文の実行が必要なとき，つまり更新要求が発行されるときを以下の二つに分ける．

(a)　サイクル精度動作記述上の新しい状態に移るとき．このとき，そのサイクルで実行する代入文をまず実行させる．これは図 8 の (1.0), (2.0) に対応している．

(b)　計算結果の値が収束していないとき．収束するまで更新を繰り返すというモデルであるため，再び代入文を実行させる．これは図 8 の (1.2), (2.2) に対応している．

最後に，繰り返しの計算過程を説明する．まず，いずれかのプロセスがクロックエッジを表す遷移を実行すると更新要求を発行する．これを全プロセスが受けて，更新を実行する遷移 (1.1), (2.1) によって代入文を実行する．このとき，更新を実行する遷移は全てのプロセスで同期して実行する．次に，更新を実行する遷移の前と後で，全ての wire 変数の値が同じに保たれているか否かによって，計算結果の値が収束しているか否かを判定する．判定の結果，収束していなければ更新要求を発行し，それを全プロセスが受けて更新を実行する遷移を再び行う．以上の処理を全てのプロセスの計算結果が収束して更新要求が発行されなくなるまで繰り返す．この繰り返しは時間 0 で行われる．したがって，値が収束しない場合は時間 0 で繰り返しを続けることになりモデル上で時間がそれ以上進まなくなる (zeno condition)．そこで，モデル検査実行時に zeno condition を検出するような検証性質を与えることによって計算結果の発振・発散を検出することが可能となる(注5)．

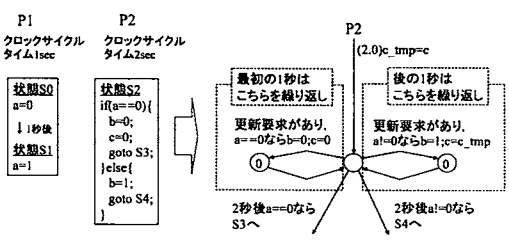

図 9　if 文を含むときのモデル化手法

(注5)：変数の取り得る値の範囲が有限であるため得られる状態モデルが有限状

検証を行った項目	項目の説明	与えた検証性質
1.代入文計算結果の正しさ	書き込みの後読み込みという順序で計算した結果と一致するか	常に正しい計算結果になっている
2.if文計算結果の正しさ	例：図6	常に正しい計算結果になっている
3.代入文のみから生じる発振検出	例：図3.(b)	zeno conditionが満たされることはない
4.if文評価による発振検出	例：P0: if(a==0){b=1;}else{b=0;} P1: if(b==0){a=0;}else{a=1;} このとき実行する代入文を変化させ続ける	zeno conditionが満たされることはない
5.wire変数への同時書き込みによる値の不定検出	例：P1:a=0, P2:a=1というように同時にwire変数に書き込むと値が不定になる	zeno conditionが満たされることはない
6.計算結果の発散検出	例 P1：a=b+1，P2:b=a+1のとき$a, b \to \infty$	整数変数の上限か下限に達することはない

図 10 実験項目

なお，図 8 には示していないが内部変数へ書き込みを行う代入文はハードウェア上ではクロックエッジにおいて行われる．したがって，そのような代入文が存在するときはクロックエッジ表す遷移 (1.0), (2.0) で実行させる．

次に，if 文を含むときのモデル化手法の例を図 9 に示す．if 文がある場合は分岐の数だけ繰り返しのパターンを増やす．また，3. で説明したように，更新実行時に訂正処理を追加する．そのためにクロックエッジを表す遷移 (2.0) においてに wire 変数の値を保持し，訂正 (c=c_tmp) に利用している．

6. 実　　験

提案手法の有効性を確認するために，いくつかの簡単なテストケースによる実験を行った．なお，モデル検査ツールには UPPAAL [1] を用いた．UPPAAL は状態遷移モデルとして時間オートマトンを，検証性質として CTL のサブクラスをサポートしている．

実験内容を図 10 に示す．項目 1.2. は提案するモデル上で wire 通信の計算が正しく行われていることの確認をするためのものである．項目 3.〜6. は wire 通信が持つ問題をそれぞれ含んだテストケースとなっており，提案手法がそれを検出できることの確認を目的としている．

実験を行った結果，項目 1.2. から提案手法が行う wire 通信の計算結果は正しいものであることがわかった．項目 3.〜6. では「問題が発生しない」という検証性質を与えて，それが偽となる (問題が発生する) 実行系列を反例として求めることができた．反例である実行系列から状態遷移モデル上のどこで問題が発生したかがわかる．5. で述べたように，状態遷移モデルの各部分はサイクル精度動作記述の各状態に対応している．これにより提案手法を用いることで問題が発生した場所も正しく求められた．

また，これらの実験において検証にかかった計算時間は数秒程度であり，小規模の例題に対しては実用的な時間で検証を行えることを確認した．

7. む す び

本稿では，サイクル精度レベルで動作記述された wire 通信を行うモジュールからなるハードウェアを対象とした検証手法を提案した．提案手法ではモデル検査を用いることにより wire 通信の動作とモジュールの状態遷移を共に解析することを実現した．また，実験では小規模な例題に対して有効性を確認した．

今後は産業界で実際に使われているハードウェアへ提案手法を適用してその実用性を確認する予定である．また，即時通信における配線遅延の揺らぎ（ジッタ）などを考慮した解析（ジッタ解析）も可能であると考えており，その拡張も今後の課題である．

態で納まる．したがってこの検証問題は決定可能となる

文　　献

[1] J. Bengtsson, K.G. Larsen, F. Larsson, P. Pettersson, and W. Yi, "UPPAAL: a tool suite for automatic verification of real-time systems," Proc. of Hybrid Systems III, Lecture Notes in Computer Science, vol.1066, pp.232–243, Springer-Verlag, 1996.

[2] G. Berry and G. Gonthier, "The esterel synchronous programming language: Design, semantics, implementation," Science of Computer Programming, vol.19, no.2, pp.87–152, 1992.

[3] H. Bowman and R. Gómez, "How to stop time stopping," Form. Asp. Comput., vol.18, no.4, pp.459–493, 2006.

[4] K. Claessen, "Safety property verification of cyclic synchronous circuits," Proc. of Workshop on Synchronous Languages Applications and Programs (SLAP), Electronic Notes on Theoretical Computer Science (ENTCS), vol.88, Elsevier, 2003.

[5] E.M. Clarke, O. Grumberg, and D.A. Peled, Model Checking, MIT Press, 1999.

[6] S.A. Edwards, "Making cyclic circuits acyclic," DAC '03: Proceedings of the 40th conference on Design automation, New York, NY, USA, pp.159–162, ACM Press, 2003.

[7] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, "Symbolic model checking for real-time systems," Inf. Comput., vol.111, no.2, pp.193–244, 1994.

[8] J.H.R. Jiang, A. Mishchenko, and R.K. Brayton, "On breakable cyclic definitions," ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design, Washington, DC, USA, pp.411–418, IEEE Computer Society, 2004.

[9] S. Malik, "Analysis of cyclic combinational circuits," ICCAD '93: Proceedings of the 1993 IEEE/ACM international conference on Computer-aided design, Los Alamitos, CA, USA, pp.618–625, IEEE Computer Society Press, 1993.

[10] T.R. Shiple, G. Berry, and H. Touati, "Constructive analysis of cyclic circuits," EDTC '96: Proceedings of the 1996 European conference on Design and Test, Washington, DC, USA, p.328, IEEE Computer Society, 1996.

[11] Y Explorations, Inc, HY-C LRM 1.2 Rev1.1, Nov. 2004. http://www.yxi.com/Library/HY-C_LRM_1.2_Rev_1.1.pdf.

[12] S. Zhao and D.D. Gajski, "Defining an enhanced rtl semantics," DATE '05: Proceedings of the conference on Design, Automation and Test in Europe, Washington, DC, USA, pp.548–553, IEEE Computer Society, 2005.