4K−1

代数的手法を用いた順序回路設計支援システムにおける検証支援機能と検証手順

森岡 澄夫　北道 淳司　東野 輝夫　谷口 健一

大阪大学 基礎工学部 情報工学科

1．まえがき

回路設計において設計の正しさを検証することは重要な課題であり，形式的手法として高階論理や時相論理等の適用が行われている[1]．筆者らの研究グループでは代数的言語 ASL による同期式順序回路の記述法を定め，形式的な設計検証を行うことのできる段階的詳細化による設計法を提案し[2]，これらに基づく回路設計支援システムを作成してきた．本稿では提案してきた設計法のもとで設計検証を形式的に行う方法，本設計支援システムにおける検証支援機能の概要，および本機能の使用例について報告する．

2．代数的手法を用いた順序回路の記述法

同期式順序回路を設計対象とし，代数的言語 ASL を用いて仕様記述を行う．回路動作を状態遷移の実行とみなし，遷移の実行制御の記述 C と各遷移の動作内容の記述 D を公理を用いて記述する(公理の書き方などの詳細は文献[2]参照)．ASL では述語や関数を用いて動作内容を抽象的に記述することができる．

例題として，レジスタ X, Y に入っている値の最大公約数を求め，結果を X に格納する動作を行う回路の仕様記述を図 1 に示す(ただし動作内容の記述 D のみ)．正数 a,b の最大公約数を表す基本関数 GCD(a,b) を用い，ある状態 s から遷移 gcd を行なった後の X の値 X(gcd(s)) が状態 s における X, Y の値の最大公約数であることが記述されている．この仕様には最大公約数を求めるアルゴリズムは記述されていない．

```
(((X(s) > 0) and (Y(s) > 0)) imply
        (X(gcd(s)) = GCD(X(s),Y(s)))) == TRUE;
```

図 1: 最大公約数を求める回路の仕様記述（レベル 1）

3．段階的詳細化

設計者は抽象的な動作内容に対し具体的な実現方法(動作アルゴリズムや抽象データタイプ等の実現法等)を考案し，設計を進めていく．設計者は，次の手順に従って正しさの保証された下位レベルの仕様 <C',D'> を合成し，これを繰り返して要求仕様より論理設計レベル(使用する部品やデータパスを決定したレベル)の仕様(レジスタの制御入力の論理式など)を得る．

（1）下位レベルで用いる動作を決める(動作内容を定める公理 D' を記述する)．（2）上位レベルの一動作を実現する下位レベルの動作系列を考案する．これを対応関係 M と呼び，公理の形で記述する．（3）詳細化の正しさを検証支援機能を用いて証明する．（4）上位レベルの実行制御の記述 C と対応関係 M から，下位レベルの実行制御の記述 C'(必要なら C' の簡単化も行う)を合成する．

論理設計レベルの仕様が得られると，マイクロプログラムまたは状態レジスタの入力論理式の合成を機械的に行う．以下，図 1 の仕様における最大公約数の計算をユークリッドの互除法により実現する場合を示す．設計者は遷移 gcd の動作を，図 2 のような動作を行う遷移 sub, exchange, terminate を図 3 のように実行して実現することを考え，上位レベルの遷移 gcd をこの遷移系列で置き換えて下位レベルの仕様を得る．

Facility to Support Verification in Hardware Design System using Algebraic Method
Sumio MORIOKA, Junji KITAMICHI, Teruo HIGASHINO, Kenichi TANIGUCHI
Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University
Toyonaka-shi, 560 Japan

```
X(sub(s))        == (X(s) - Y(s));
Y(sub(s))        == Y(s);
X(exchange(s))   == Y(s);
Y(exchange(s))   == X(s);
X(terminate(s)) == X(s);
Y(terminate(s)) == Y(s);
```

図 2: 最大公約数を求める回路の仕様記述（レベル 2）

```
gcd(s)  == loop(s);
loop(s) == if (X(s) = Y(s)) then terminate(s)
           else if (X(s) > Y(s)) then loop(sub(s))
           else loop(exchange(s));
```

図 3: レベル 1 ・ 2 間の対応関係の記述

4．段階的詳細化の正しさの検証法

論理設計レベルの仕様が要求仕様を満たしていることは，3．における詳細化の手順（3）により保証される．詳細化が正しいとは，上位レベルの一遷移によるレジスタの値の変化が，下位レベルでも対応関係 M に従った遷移の実行により実現されるということである．これは代数的には，対応関係記述の公理および下位レベルでの動作内容記述の公理を用いることにより，上位レベルの各公理がそれぞれ定理となることに相当する[3]．この証明は，上位レベルの公理 $\alpha == \beta$ に対する式 $\alpha = \beta$ が恒真である(ただし α, β の型は整数あるいは論理型)ことを以下のような手法を適当に組み合わせて示すことにより行われる．

帰納法の適用：詳細化に用いる遷移系列に閉路が存在する，すなわち下位レベルでのある動作が繰返し実行される場合，検証者は閉路上の適当な途中状態に対し，その状態で成り立つべき不変式を設定する．また，始状態に証明すべき式の前提条件，終状態には証明すべき式を設定する．始状態・終状態も含め，これらの設定を行なった状態を**不変式付き状態**と呼ぶ．終状態に設定された式が成り立つことは次の方法により示される．

> 任意の 2 つの不変式付き状態 S_1, S_2 に対し，S_1 から S_2 に至る遷移系列で，その途中状態に不変式付き状態を含まないものが存在するなら，そのような全ての遷移系列について「状態 S_1 での不変式 R_1 が成り立ち，その遷移系列の実行条件が成立して状態 S_1 から状態 S_2 に到達したとき，状態 S_2 での不変式 R_2 が成り立つ」ことを証明する．

以下，1 つの遷移系列についての証明を**不変式保存証明ステップ**と呼ぶことにする．

項書き換えと場合分け：対応関係の公理，下位レベルの動作記述公理，および基本的な演算に関する公理を書換え規則と見なし証明すべき式を書き換える．その結果に if 文がある場合，条件部による場合分けを行う．場合分け後，項書き換えを行ない，その結果に if 文が出現すればさらに場合分けを行う．

必要な補題の追加：数学的に正しいと分かっている性質などを適当に補題 A として導入し，証明すべき式 B について「A ならば B」を表す式 C を作成して，この恒真性を判定する．

プレスブルガー文真偽判定アルゴリズムの適用：プレスブルガー文真偽判定アルゴリズムは，型として整数及び論理型，演算として $+$, $>$, $=$, $\wedge$, $\vee$, *not* 等で構成されるクラスの式の恒真性を判定するものである．

5．検証支援機能

我々が作成している順序回路設計支援システムにおける検証支援機能は上位・下位レベルの動作内容の公理と対応関係の公理を入力とし，以下のことが行える．

詳細化に用いられた遷移系列の図示： 対応関係の公理を状態図として図示する．この際，簡単な図形レイアウトを行うが，検証者が後に修正を加えることもできる．

不変式編集の支援： 対応関係中の状態図における閉路の始点を検出し図示する．検証者はその状態に対し必ず不変式を設定しなければならない．また，検証者は編集作業用のウィンドウを開いて，任意の途中状態に対して不変式の設定・修正・削除を行うことができる．設定された不変式は表示用ウィンドウに常時示される．

証明実行の管理： 各不変式保存証明ステップを任意の順に証明でき，また，各ステップが未証明あるいは証明済であるかを常時管理する．また，証明済のステップに含まれる遷移は実線，未証明のステップに含まれるものは点線で表示される．

各不変式保存証明ステップの証明に用いる式の自動生成： 検証者がどのステップを証明するか指定することにより，恒真性を示すべき式を自動生成する．

不変式修正による再証明の支援： 不変式を修正した場合，すでに証明された不変式保存証明ステップのうち，再証明すべきものを自動的に未証明の扱いにする．

項書き換えの支援： 証明すべき項に対する項書き換えを自動で行うことができる．また，手動で項書き換えを行う場合についても，適用可能な規則を検証者に提示するなどの支援を行う．

場合分けの管理： 場合分けに用いる条件式 (一般に複数) を検証者が指定した後，各条件式に対する真偽の割当てを (どの割当てが未検証あるいは検証済であるか) 管理する．各真偽割当て毎に，各条件式が真または偽であることを表す公理が追加される（そのもとで項書き換え等を行なうことができる）．

プレスブルガー文真偽判定： プレスブルガー文真偽判定を行う．

6．検証例

上述の機能の使用例として，2．における最大公約数の例題の検証について説明する．検証者が上位・下位レベルの仕様と対応関係を記述 (図 1・2・3) して本設計支援システムに与えると，対応関係が図示される (図 4)．また，状態 1 に不変式が必要であることが検出され，四角枠で丸を囲った記号で表示される．そこで検証者は状態 1 に設定する不変式を考案し (図 5)，不変式編集ウィンドウを用いて設定を行う．この不変式は「レジスタ X, Y の値の最大公約数が状態 0 におけるレジスタ X, Y の最大公約数と同じである」ことを表している．設定がなされると状態 1 の記号が二重丸に変化し不変式の内容を常時表示するウィンドウが開く (図 6)．

この例題では，以下の 4 つの不変式保存証明ステップについて証明を行なう必要がある．

- 状態 0 から状態 1 への遷移（帰納法の初期段階）
- 状態 1 から状態 1 への遷移 exchange（帰納法の帰納段階）
- 状態 1 から状態 1 への遷移 sub（帰納法の帰納段階）
- 状態 1 から状態 2 への遷移 terminate（帰納法の終了段階）

図 6 の段階ではいずれのステップも未証明なので，全ての遷移が点線で表示されている．検証者がいずれのステップを証明するか指定すると，本機能は恒真性を示すべき式を自動生成する．例えば遷移 exchange を含むステップについては「状態 1 において不変式と遷移 exchange の実行条件が両方とも成立するとき，遷移 exchange を実行した後の状態 1 でもその不変式が成立する」ことを表す式が生成される．検証者がこの式に対し自動項書き換えを行い，関数 GCD に関する数学的な性質 (GCD(a,b) と GCD(b,a) は等しいこと) を表す公理を補題として追加し，プレスブルガー文真偽判定アルゴリズムを適用したところ，恒真であると判定された．よってこのステップの証明が成功し，遷移 exchange の表示が実線に変化した (図 7)．さらに検証者が他のステップについても同様に証明を行った結果，全段階の証明が成功した．これにより遷移の表示は全て実線となり (図 8)，検証作業が終了した．

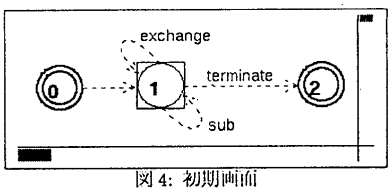

図 4: 初期画面

```
(((X(s0) > 0) and (Y(s0) > 0)) imply (X(s) > 0 and
Y(s) > 0 and GCD(X(s0),Y(s0)) = GCD(X(s),Y(s))))
```
図 5: 状態 1 に設定された不変式

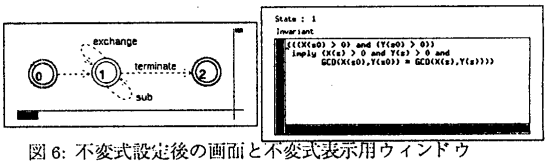

図 6: 不変式設定後の画面と不変式表示用ウィンドウ

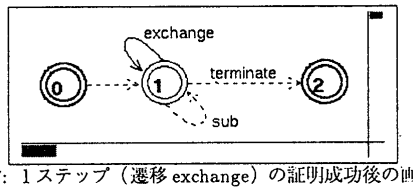

図 7: 1 ステップ（遷移 exchange）の証明成功後の画面

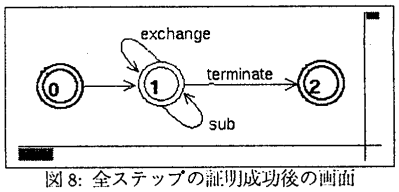

図 8: 全ステップの証明成功後の画面

7．あとがき

以上，代数的手法を用いて詳細化が正しいことを形式的に検証する方法と，我々の設計支援システムにおける検証支援機能の概略、およびその使用例について述べた．今後の課題として，実用規模の例題を用いての本支援機能の評価や，全正当性の検証支援の方法の検討およびその検証支援のための機能の追加等があげられる．

参考文献

[1] J.J.Joyce:''Formal Verification and Implementation of a Microprocessor'', VLSI Specification, Verification and Synthesis, Kluwer Academic Publishers, pp.129-157(1988)
[2] 杉山裕二，北道淳司，谷口健一:''代数的手法を用いた順序回路の記述とその詳細化について'', 信学技報, SS88-7(昭 63-06).
[3] 北道淳司，谷口健一:''代数的手法を用いた順序回路の詳細化と正しさの証明'', 1989 年信学会秋期全国大会, SD-10-6(1989).