協調型論理設計エキスパートシステムにおける仮説推論に基づく再設計機構

Redesign Mechanism Based on Assumption-Based Reasoning for Cooperative Logic Design Expert System

角田多苗子　丸山文宏　松永裕介　箕田依子　川戸信明

Taeko Kakuda, Fumihiro Maruyama, Yusuke Matsunaga, Yoriko Minoda, and Nobuaki Kawato

富士通株式会社

FUJITSU LIMITED

We are developing a cooperative expert system for VLSI logic design, in which functionally distributed agents cooperate iterating the refine-evaluate-redesign cycle under the constraints of area and time. In this paper, we propose a redesign mechanism based on assumption-based reasoning to automate inevitable redesign process.

We consider design decisions as assumptions, and constraint violation as contradiction. When a constraint violation is detected, the redesign mechanism is invoked. Justifications for constraint violations, called nogood justifications(NJs), play a central role. Our system implements redesigns by expanding and generating NJs in a hierarchy that represents the circuit being designed. Assumption-based reasoning is more important when cooperation is involved, because decisions must be retracted.

1. はじめに

ＶＬＳＩ技術の急速な進歩に伴い，高品質な回路を短期間で設計できるような支援システムの開発が望まれている．筆者らは，設計過程全体を統一的に管理し，より高度な支援を行うことを目標にして，協調型論理設計エキスパートシステムの研究を行っている．このシステムは，論理設計における機能分散に注目することによって，機能的に分散した複数のエージェント（データパスを設計するエージェント，制御回路を設計するエージェント）を協調的に動作させて問題解決を行うものである．エージェント内部だけでなく，協調機構による他エージェントからの要請や，制約条件自体の変更に基づいて起こりうる再設計を自動化し，かつこれを効率的に行うために，仮説推論に基づく再設計機構を備えている．

設計過程は，仕様からターゲットとする（例えばＣＭＯＳ）回路までの段階的な詳細化の過程であり，その各段階では，例えば機能ブロックの構成等を選択していくことになる．求める回路は，最終的な回路レベルで，制約条件（例えば，遅延時間，全体のセル数）を満たす必要があるが，各選択の時点では，内部構造が完全には詳細化されていなかったり，他の部分の詳細化が必ずしも完了しているとは限らないため，その結果が直ちに判明するわけではない．すなわち，途中の選択段階では，遅延時間等の制約条件を評価できるとは限らないため，妥当と思われるものを選択して設計を進めていき，結果的に条件にあわないことがわかるとやり直すという，詳細化・評価・再設計の繰り返し過程が不可避である．

一方，仮説推論は，問題解決の際に，真偽が不明であるデータあるいは知識を，とりあえず真として推論を進める推論方式である．推論が進み新しいデータが生成されるにつれて，以前に真とした仮説が誤っていたこと（矛盾をおこすこと）が判明し，矛盾を解消する必要がでてくる．これを設計過程と対比してみると，設計の途中段階で採用する選択肢は仮説に，制約条件違反は矛盾に対応し，再設計は矛盾を解消することと考えられる．

そこで，仮説推論に基づいた再設計機構を実現することとした．

2. 設計問題における仮説推論

一般に仮説推論を検討する上で，まず何を仮説，及び矛盾とするか，ということを明確にする必要がある．また，仮説推論を実現するためには，矛盾の検出，及び矛盾の解消，という2つの課題に取り組まなければならない．

設計問題における仮説推論として，ここでは，複数の選択肢それぞれ（例えば，加算器を設計する選択肢としてキャリールックアヘッド・アダーとリップルキャリー・アダー）を仮説とする．また，仮説推論の結果生じる矛盾としては，設計された回路が与えられた制約条件に違反することであると考える．したがって，矛盾の検出，解消，という問題は，設計においては，制約条件に対する違反の検出，及び効率的な再設計の実現，という問題となる．

一般に，仮説推論を実現するためには，あるデータが何を根拠に出てきたのか，あるいは，どのような理由で矛盾をおこしたのか，という情報を記憶しておくことが必要である考えられ，従来ＡＴＭＳ等で，仮説と理由付け（justification)に基づく管理方法が考えられている．しかし，これらの手法には以下のような問題点があるために，ここで対象とする設計の問題には適用できない．まず，ＡＴＭＳでは，全ての仮説を予め数え上げ，それらの組み合わせ全てを管理の対象とするが，設計ではある仮説が意味をもつかどうかはそれ以前の仮説選択に依存するため，すべての仮説の組み合わせに意味があるわけではない．また，ＡＴＭＳではある部分集合で矛盾が見つかった場合，そのスーパーセットも矛盾を含むとして探索範囲を絞れる．しかし，設計では全体の詳細化が完了しないと矛盾を検出できないことも多く，小さい部分集合から探索範囲の絞り込みを行うことは一般にはできない．そこで設計向きの仮説推論を実現するために，「失敗の理由付け（NJ：nogood justification）」に基づく再設計機構を考案し，これにより上記の問題の解決を図ることとした．以下では，データパス設計エージェントを例として，再設計機構について述べる．

3. NJに基づく再設計機構

NJは，制約条件，選択肢，セル数や遅延時間に関する条件式等の論理積で表現される，常に偽でなければならない論理式である．

NJに基づく再設計機構のポイントは以下の点である．

①ある選択肢をとることを禁止する条件すべてをNJの形で蓄積する．制約条件に対しても，それに相当するNJ（デフォルトNJ）を生成し，統一的に扱う．

②設計データ（階層設計されるデータパス構造を表現する）を階層データ構造で構成し，その中に，各選択に対応した"オルタナティブノード"を導入する．

③NJを，関連するオルタナティブノードに分散して格納し，再設計の管理に用いる．

これによって，前節の問題のうち，まず制約条件に対する違反の検出については，デフォルトNJの真偽のチェックにより行う．再設計に関しては，まずそれぞれの選択を禁止する条件がNJとして管理されているために，やり直しに関係する部分を特定し，無関係なデータの取消を避けることができる．また，再設計箇所の決定は，NJに基づいて階層データ構造上を移動しながら行うが，この際のNJの操作やチェックは設計データを表現する階層データ構造に対応して行えばよく，これらの操作やチェックを効率的に行える．さらに，いったん選ばれた選択肢の設計結果は，オルタナティブノードとして蓄えられているため，環境の変化によっては，内部を再度設計することなく，生き返らせることが可能である．

以下，まず設計データを表現する階層データ構造とその上でのNJの生成操作について説明し，最後に再設計処理の概要を示す．

3.1 階層データ構造

図1に設計対象を表現する階層データ構造を示す．この構造は，2種類のノード，すなわちコンポーネントノードとオルタナティブノードによって構成される．コンポーネントノードは，加算器，レジスタ等の回路要素（コンポーネント）を表し，

そのトップレベルをデータパスノードと呼ぶ．システムは，データパス部を階層的に設計し，この結果，コンポーネントはいくつかの部分コンポーネントから構成されることになる．コンポーネントの構成法には選択肢があり，オルタナティブノードは，コンポーネントの一つ一つの構成法を示し，これは設計における個々の仮説に対応している．例えば，図1に示すように，コンポーネントcomp1 をcomp11とcomp12から構成するという選択肢があった場合，comp1 がcomp11とcomp12から構成されることを表現するオルタナティブノードalt1が作られる．

コンポーネントのある選択肢を採用したことにより矛盾が発生し，再設計によって新たな選択が行われると，新しいオルタナティブノードが生成され，もとのオルタナティブノードはその時点で無効となる．各時点で有効なオルタナティブノードは一つのコンポーネントに対して高々一つで，これをINの状態にあるといい，無効になっているオルタナティブノードはOUT の状態にあるという．INオルタナティブノードだけが，その時点での回路の階層構造を表している．OUT の状態のものは，まとめてそのコンポーネントノードに保存され，環境が変化すれば生き返らせることができる．

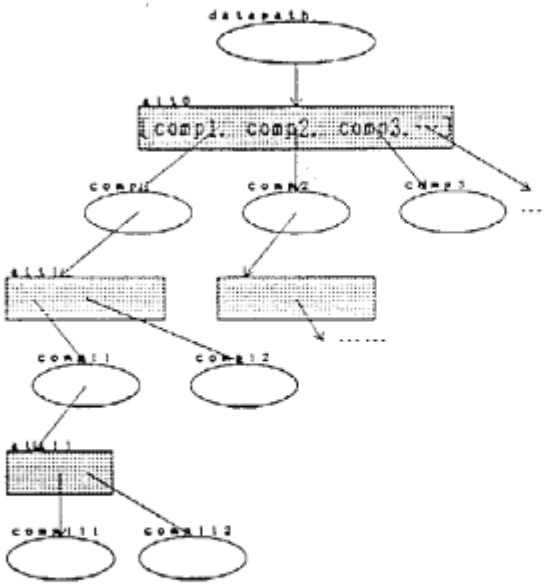

図1　設計データを表現する階層データ構造
（網かけしたものがオルタナティブノード）

システムはこの構造上で，制約条件の評価，及び再設計を行う．次節で述べるように，NJをこの構造上に分散して配置でき，再設計箇所の特定等に必要なNJに対する操作もこの構造に対応付けて定義できる．

3.2　NJの生成

NJは，与えられた制約条件に相当するデフォルトNJを基として，展開，及び合成という2つの操作を施すことによって，新たに生成され，オルタナティブノードに格納される．

(1)デフォルトNJ

デフォルトNJは，回路が満たすべき制約条件を等価なNJの形に変換したもので，評価の拠点となるコンポーネントノード直下のオルタナティブノードに書き込まれる．例えば，データパスのセル数制約が与えられていた場合に生成されるデフォルトNJは，次のようになり，図1のオルタナティブノードalt0に書き込まれる．

［datapathの構成はalt0］
　∧［Σ_i alt0!comp$_i$!cell ＞
　　CNSTR(datapath) ］…①

但し，alt0! comp$_i$!cell は，alt0を構成する各コンポーネントcomp$_i$ のセル数，CNSTR(datapath) はdatapathのセル数制約を示す．

実際は，①中の［datapathの構成はalt0］という項は，alt0というオルタナティブノード自体が表しているので，alt0ノードには，①中の不等式の項だけを格納しておけばよい．

(2)NJの展開

あるオルタナティブノードのNJをその子コンポーネントノードAに関して展開するとは，

(a) Aにおける IN状態のオルタナティブノードを加え，

(b) Aが含まれる条件式におけるAの部分を，その構成要素に関する条件に置き換える

ことである．例えば，①のNJがcomp1(セル数400とする）について展開されると，②が生成される．前と同様に構成に関する項はそのNJが書き込まれるオルタナティブ自体が表しているため，不等式の項だけがalt1に書き込まれる．

[datapathの構成はalt0]

∧ [compl の構成はalt1] (*)

∧ [$\sum_{i=1}^{n}$ alt0!comp$_i$!cell

+ Σ alt1!compl$_i$!cell (*)

> , CNSTR(datapath)] …②

NJの展開は，あるコンポーネントをさらに詳細に見ることによって，階層構造上を下へと移動し，その内部の変更部分（新たな選択肢を探すべきコンポーネント）を決定するために行われる．

compl を再設計するとは，alt1に書き込まれた，compl について展開したNJ②を偽にするように，compl の内部構造を変更することである．変更できるのは，自分の構成要素だけであるため，comp1 を構成する compl1 または compl2 のどちらかを選んで，そのノードに対して展開操作を繰り返す．これ以上展開できなくなると，そのオルタナティブ自身が否定されOUT 状態になり，親コンポーネントが変更される．

(3)NJの合成

NJの合成とは，あるコンポーネントノードに関する選択肢がすべて否定されたときに，個々の選択肢を含んだNJを合成して，そのノードの項を削除したNJを生成することである．例えば，あるコンポーネントXに３つの選択肢α，β，γしかないとして，それぞれのオルタナティブノードにそれらを禁止するNJ，$A \wedge \alpha$，$B \wedge \beta$，$C \wedge \gamma$ があったとすると，それらからXに関する項が削除されたNJ，$A \wedge B \wedge C$ が合成され，階層構造上を上に移動して，コンポーネントXを構成要素として含むオルタナティブノードに格納される．これは，その時点での環境のもとではコンポーネントXが再設計不可能なことを示すものである．これにより，X以外のコンポーネントを選択し，再設計することになる．

3.3　再設計処理の概要

図２に再設計の処理フローの概略を示す．

デフォルトNJが真になると，そのデフォルトNJを持つオルタナティブノードを出発点として再設計が始まる．NJの展開，或いは合成操作を行いながら階層データ構造上を移動して再設計すべきコンポーネントを決定し，そのINオルタナティブをOUT にし，OUT の中から一つのオルタナティブを選び（または新たにオルタナティブを生成し），INにする．すべてのコンポーネントノードにINオルタナティブが存在し，かつすべてのNJをも真にしないとき，設計が完了する．

NJを階層構造中の関連する部分に分散して格納しているため，矛盾の解消に関連する部分を局所化し，探索空間を絞ることが可能となる．階層構造を下へ移動する際，複数の子ノードのうちどれを選ぶかは，設計領域に依存する知識を用いて決定する．

4.　おわりに

協調型論理設計エキスパートシステムにおける，仮説推論に基づく効率的な再設計機構について，設計データを表現する階層データ構造とその上でのNJの管理方法を中心に述べた．現在，この機構を含むシステム全体の試作をＩＣＯＴで開発された逐次型推論マシンＰＳ１上で進めている．

謝辞

本研究は，第五世代コンピュータプロジェクトの一環として行われたものであり，御支援頂いたＩＣＯＴ第五研究室藤井室長に感謝致します．

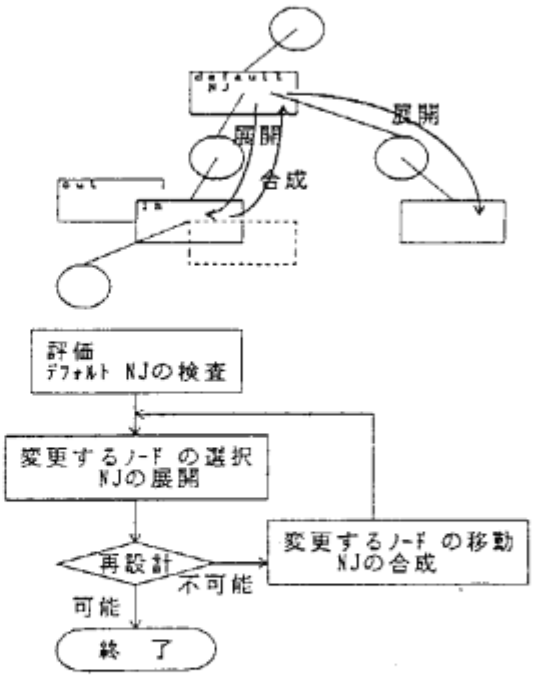

図２　再設計の処理フローの概略