

協調型論理設計エキスパートシステム

角田多苗子, 丸山文宏, 松永裕介, 川戸信明

富士通株式会社

機能的に分散されたエージェントが設計データと制約条件を交換しながら協調動作する、協調型論理設計エキスパートシステムの枠組みを提案する。第一段階として、構造記述に基づいたスタティック回路設計と、動作記述に基づいた制御回路設計に対する2つの分散されたエージェントを考える。一つの例に沿って、このシステムにおける設計の流れを示す。面積と時間についての制約で回路を評価し、もし設計結果が制約を満たさないものならば、再設計のプロセスが起動される。最後に、3つのプロセッサから構成される実験的システムを提案し、課題について触れる。

Cooperative Expert System for Logic Design

Taeko KAKUDA, Fumihiro MARUYAMA, Yusuke MATSUNAGA, Nobuaki KAWATO

Fujitsu LTD.

Kawasaki 1015, Kamikodanaka Nakahara-ku, Kawasaki 211, Japan

We propose a framework of cooperative expert systems for logic design, in which distributed agents cooperate by exchanging design data and constraints. As the first step, we consider two distributed agents, one for designing static circuits based on structural description, the other for designing control circuits based on behavioral description. An example is examined, where the circuit is evaluated against constraints about area and time. If the result is unacceptable, the redesign process is invoked. Finally, we propose an experimental system consisting of three processors and touch upon some of the things that have to be done.

1. はじめに

VLSI技術の急速な進歩は、様々な分野において、製品の短寿命化と新製品開発競争を激化させており、この状況は今後ともとどまることはないと考えられる。このため、高品質な製品を短期間で設計することのできる高度な設計支援システムの実現が強く望まれている。このようなシステムを構築するためには、従来技術では取り扱い困難な熟練設計者の持つ優れた問題解決能力を対象とした研究を進め、これをコンピュータ上に実現していかなければならない。そのためには、このような問題解決能力の発揮に必要な基本的枠組みを究明し、これを備えたCADシステムを構築することが重要であると考えられる。

熟練設計者は、VLSI設計に関する各種の知識を利用/補完し、また、詳細化・評価・再設計の繰り返しを見通しよく行い、高品質な設計結果を得ていると考えられる。従って、各種の知識ベースあるいは設計ツールを統合し、設計過程全体を統一的に管理する設計環境の実現が必要である。

米国カーネギーメロン大学では、既存の各CADツールを知識源とした黑板モデルに基づく設計環境 ULYSSESシステムを開発している [Bushnell1986]。これは、CADツールの蓄積を有効に利用した現実的なアプローチと考えられるが、評価・再設計の繰り返しを直接サポートできるようにツール内部を変更することは不可能である。

イリノイ大学では、設計過程における詳細化を上位レベルの機能仕様から下位レベルの構造仕様への変換とみなし、構造仕様を次の機能仕様としてトップダウンに設計を進めるエキスパートシステムを提案している [Brewer 1986]。このシステムでは、下位レベルから上位レベル

への失敗の報告という形で再設計を起動し、詳細化・評価・再設計のループをサポートしているが、仕様および制約条件が上位から下位へ一方向に流れるというトップダウン設計に限定されており、設計手法としては柔軟性に欠ける。

本稿では、上記の問題を解決するため、VLSIの論理設計における各種の機能分散に注目し、各機能専門のエージェントを設け、それらを協調させながら全体の論理設計を行う論理設計エキスパートシステムco-LODEX(cooperative logic design expert system)を提案する。第一ステップとして、構造記述に基づくスタティック回路設計と動作記述に基づく制御部設計を異なるエージェントに振り分け、これらを協調させるシステムについて検討する。

2. システムの概要

2.1 システムの位置づけ

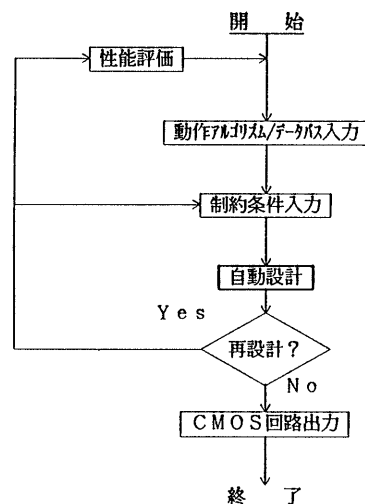


図1 本システムを用いた論理設計の手順

図1に、このシステムを用いた論理設計の手順を示す。論理回路の設計においては、その基礎となる大まかなデータパスが非常に重要であ

り、不適当なデータベースから出発した場合、いくら細部の修正を積み重ねても高品質な結果は得られない。従って、初期入力として、動作アルゴリズムと共にデータベースを設計者に入力してもらう。

各種の設計条件（例えば、回路規模、時間、テスト容易性）を満たす設計に導くためには制約条件（設計条件を具体的に表現したもので、例えば、「ゲート数が1200以下」）を利用する。設計者は、やはり初期入力として、制約条件を入力する。ここでシステムは自動設計を開始する。設計者は、出力された設計結果に対して、制約条件を強化あるいは緩和して再設計を行わせ、別の可能性を探ることができる。この際、システムは零からやり直すのではなく、強化／緩和された制約条件に無関係なところはそのまま採用することにより、再設計のループのターンアラウンドを短くする。設計者はこのループを繰り返して設計条件を満足する結果を得る。動作アルゴリズムあるいはデータベースの大幅な変更が必要となる場合には、設計者は性能評価用ツールを利用し、その結果に基づき最終的な判断を下す。

2.2 設計のプロセス／データ管理

上で述べたループ以外に、設計では各段階において詳細化・評価・再設計のループが現れる。設計プログラムとは独立した評価プログラムを用いるアプローチでは、評価の結果制約条件を満たしていないことがわかった場合、設計プログラムを最初から再実行するといった形になりやすい。また、制約条件が変わる度に全体をやり直す必要がある。さらに、全体についての条件から決まってくる部分についての条件（ある演算器のゲート数等）を扱いにくい（部分についての条件を扱うためには動的な設計／評価の制御が必要だから）ために、制約条件で設計プ

ロセスを巧みにガイドすることが難しい。

熟練設計者は、効率的に設計を行うため、制約条件を常に意識し、それを評価できるようになった時点からなるべく早い機会に評価を行っている。従って、本システムには、設計データと制約条件を密着させ、評価の結果制約条件を満たしていないことがわかった時には、自動的にやり直しプロセスを起動するメカニズムが必要である。また、設計のやり直しにもかかわらず、設計データが矛盾なく一貫して管理されることが必要である。しかも、設計データを無駄に取り消すこと（やり直した箇所に無関係な情報をも取り消してしまうこと）を最小限に抑えることが必要である。TMS [Doyle 1979] や ATMS [de Kleer 1986] で提案されているように、理由付け（justification）を付加／管理することにより、設計データを矛盾なく一貫して管理することを提案する。やり直しに無関係かどうかはこのような情報から判断する。

2.3 分散協調の枠組み

[Smith, 1985]によれば、分散システムを検討する理由には次のようなものがある。

- (a)問題の性質・特徴から見ると、空間的あるいは機能的分散が自然である。あるいは、グローバル・ゴールがない。
- (b)より速く、より信頼性の高い、そしてより拡張性のある強力な問題解決システム構築に有望である。

論理設計の場合、従来から認識されてきた、構造設計に基づくスタティック回路設計と動作記述に基づく制御設計の切り分け、さらに、テスト設計（テスト容易化設計及びテスト生成）といった機能分散がある。そして、設計の品質は、互いに矛盾する複数の尺度、つまり、回路の面積、時間、テスト容易性によって評価され、グローバルなゴールはない。言い換えると、与

えられた仕様を満たす回路を一次元に並べることはできず、尺度に対応する複数次元の空間内の許容される領域に含まれる回路の中から解をひとつ選ぶことになる。この選択（許容領域に含まれる複数の候補回路からひとつを選ぶこと：簡単なのは最初に見つかった回路を取ること）はケース・バイ・ケースで行われる。

以上の考察から、機能分散によってそれぞれかなり独自に得られた部分解を統合して、最終的な許容領域に含まれる候補回路を求める方式が考えられる。「統合」という意味は、それぞれ単一あるいは少数の尺度のもとで得られた部分解を組み合わせても、その結果が最終的な許容領域に含まれる保証はないため、部分解を調整する必要があるということである。本システムでは、この調整を制約条件を利用して行うこととする。ある尺度に関してもっと優れた回路が要求される場合には、その尺度に関するより強い制約条件のもとでベターな部分解を求める。この結果、他の部分解の手直しが必要な場合もある。このようにして、高品質な自動設計を達成することができる。さらに、物理的に異なるプロセッサ上で機能分散を実現すれば、設計時間の短縮にも寄与することができる。

我々は、第一ステップとして、構造記述に基づくスタティック回路設計と動作記述に基づく制御部設計を切り分け、異なるエージェントに振り分けるシステムを検討する。さらに、設計者を分散協調の枠組みに取り込むために、設計者を一つのKS（知識源）とみなすことにより、統一的なインタフェースで設計者の高度な知識が利用できるようにする。必要なら、あたかもルール次の候補を呼び出すように、設計者に別解を要求する。

3. システムの動作例

以下の検討では、制御回路設計エージェント

は、ステートマシン設計KS、オペレーション変換KS、トランスレーションKS、状態割り当てKS、遅延評価KSをもち、またスタティック回路設計エージェントは、コンポーネント設計KS、テクノロジマッピングKS、ゲート数評価KSをもつものとする。この他に、各エージェントには、再設計のためのKSがある。以下、例題に沿ってこのシステムの動作を示す。

3.1 初期入力

```
FUNCTION:main:clk:
  idle::
    STOP(RST=0),X<-XI,Y<-YI,GOTO main1.
  main1::
    IF (X=Y) THEN(OU:=X,GOTO idle)
    ELSE (IF (X<Y) THEN(Y<-Y-X),
          ELSE(X<-X-Y),
          GOTO main1).
FEND;
```

図2 動作記述

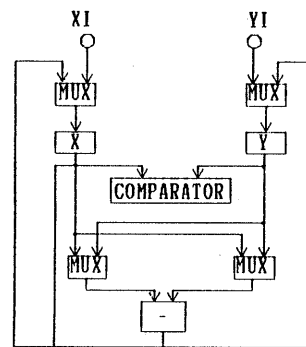


図3 データパスの記述

図2は、2つの正数の最大公約数を求めるハードウェア [Camposano 1087] の動作記述、図3は、そのデータパスの記述を示している。

動作記述は、富士通研究所で開発された言語、UHD L (Unified Hardware Description Language) で与える。この言語は、レジスタトランスファレベルのハードウェア記述言語 DDL を時相論理 [Moszkowski 1986] を用いて拡張したもので、連続した状態の集まり (長さが 1 サイクルとは限らない) であるインターバルを単位として動作を記述する。

構造記述はデータパスの構造を特定するもので、設計者は頭に描いているデータパスをコンポーネント (レジスタ、カウンタ、加算器等の機能ブロック、ユーザが定義することもできる) とその接続で表現する。

設計者は初期入力として以上の動作、構造記述の他に制約条件を与える。この例では、クロック周期が 200ns 以下、全ゲート数 1200 個以下とする。

3.2 オペレーションとパスの対応付け

動作と構造は、同じハードウェアを別の側面から見たもので、互いに関連しあったものである。そこで、動作記述中のオペレーションと、そのオペレーションが実行される時に通るデータパス上の道筋に対応をつける。また、ユーザ定義コンポーネントとオペレーションとの対応も必要となる。これらの作業は、オペレーションに対応するパスや、ユーザ定義コンポーネントを画面上で設計者に指示してもらうことにより行い、システムとしては、使用されないコンポーネントがあるか否か、等のチェックを行う。この対応付けにより、各コンポーネントには、どのオペレーションでどういうパス上で用いられるかの情報が記録される。これは、後にパスに沿う遅延を計算するとき使用される。

これ以後、制御回路設計エージェントとスタティック回路設計エージェントは、それぞれに独立に設計を実行するが、他方の設計結果 (評

価値等) が必要な場合は、待ち合わせを行い、協調的に設計を進める。

3.3 制御回路設計 (動作側)

・ステートマシン設計 K S

UHD L のインターバルをステートで実現する。

・遅延評価 K S

オペレーションとパスとの対応関係を参照して、クリティカルパスと考えられるパスに沿う遅延時間を計算する。この計算は、必要となるコンポーネントが設計されるのを待って実行される。この例では、最大遅延が 130ns と計算され、条件を満たしている。

3.4 スタティック回路設計 (構造側)

・コンポーネント設計 K S

入力されたコンポーネントを、ユーザ定義コンポーネントや演算器から順に設計する。この例では、まず減算回路の設計にとりかかる。コンポーネントの実現の方法は何通りかあるが、最初は最も一般的と思われるものを選ぶことにすると、減算回路の場合、減数の補数を加算する方法がとられる。この例では、データパスとオペレーションの対応付けから結果が負になることはない ($X > Y$) ことが判断されるため、結果が負になる場合に必要な出力側の補数回路を省略した図 4 のような構成で、設計する。

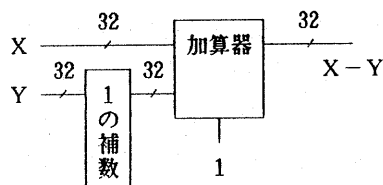


図4 減算回路の構成

・テクノロジマッピング K S

減算回路を構成する加算器、1の補数回路、および初期入力された32ビット比較器、32ビットレジスタ X、Y、32ビットの2入力マルチプレクサ 4個をスタンダードセルで構成する。

・評価 K S

全体のゲート数を計算する。表1に最初の設計でのゲート数を示す。各評価値には、それがどの方法によってできたものであるか、という理由付けが付加されている。この結果、制約の1200個を272 ゲート超過していることが検出され、再設計 K S が起動される。

表1 全ゲート数

コンポーネント	ゲート数
減算回路 (加算器+補数回路)	432 (400+32)
比較器	336
レジスタ2個	320
マルチプレクサ4個	384
計	1472

3.5 設計変更

・再設計 K S (構造側)

全ゲート数の理由付けをみて、それを構成する各コンポーネントのゲート数を減らそうとし、ゲート数減少の効果が大きいもの(ゲート数の大きいもの)から再設計を試みる。

全ゲート数の理由付けは、
(減算回路は432ゲート) ∧ (比較器は336ゲート)

∧ (レジスタは320ゲート)

∧ (マルチプレクサは384ゲート)

∧ (432+336+320+384=1472)

であり、さらに(減算回路は432ゲート)は、

(加算器は400ゲート) ∧ (補数回路は32ゲート)
∧ (400+32=432)

という理由付けをもつ。ここでは、減算回路中の加算器が選ばれる。

・コンポーネント設計 K S

加算器を、4ビットアダーセル8個から2ビットアダーセル16個に変更する。この際、再設計に関係するのは加算器だけで、その他のコンポーネントには影響しない。この結果遅延は長くなるがゲート数が144 個減少する。しかしまだ 128個超過している。

・再設計 K S (動作側)

コンポーネントの再設計というローカルな手段では条件を満足できないので、構造側の再設計 K S は動作自体の変更を動作側に要求する。動作側の再設計 K S は、それをうけて入力の U H D L 記述を変換するために、オペレーション変換 K S を起動する。

・オペレーション変換 K S

オペレーション変換 K S は、

「比較の対象が定数の場合回路が簡単になる可能性がある」、

「A=B は A-B=0と等価」、

「A>B は A-B>0と等価」、

「B-A は -(A-B) と等価」、

などの単純化の知識を使って図2の U H D L を図5のように変換する。

```

      ⋮
      ⋮
main1::
  IF (Y-X=0) THEN (OU:=X, GOTO idle)
  ELSE (IF (Y-X>0) THEN (Y<-Y-X)
        ELSE (X<- -(Y-X)),
        GOTO main1).
```

図5 動作記述の変形

・再設計K S（動作側）

変換したU HD Lを設計者に示し、減算がどのオペレーションでも常に $Y-X$ であるとか、比較器が簡単化できるのではないかと示唆して、データパスの変更を要求する。設計者は、それによって、データパスを図6のように変更する。すなわち、比較器を削除し、新たに2の補数回路と、0判定回路を導入する。なお、この変更は自動化できる可能性があり、今後検討する。

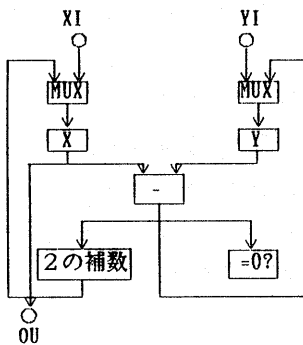


図6 変更したデータパス

構造側のコンポーネント設計K Sがこれに対応して、あらたに加わった2の補数回路と0判定回路を設計する。再びゲート数の評価を行うと、最終的に表2のようになり、全体が1008ゲートとなり条件が満たされる。動作側でもU HD Lの変更に伴って影響をうける箇所を変更し、オペレーションの遅延を再評価すると、最大遅延 188nsで制約が満足されているので、この動作に基づいて制御回路を再設計する。

表2 設計変更後の全ゲート数

コンポーネント	ゲート数
減算回路 (加算器+補数回路)	288 (256+32)
2の補数回路	184
0との等価性回路	24
レジスタ2個	320
マルチプレクサ2個	192
計	1008

4. 今後の課題

以上のような検討を踏まえて、我々は図7に示す構成のシステムを試作・評価する。各エージェント（スタティック回路設計のエージェント、制御部設計のエージェント、設計者）は、I C O Tで開発された逐次型推論マシンP S I上に実現する。設計者以外のエージェントは複数のK Sを持ったエキスパートシステムである。

すでにエキスパートシステムの基本部分はP S I上に論理型オブジェクト指向言語E S Pで実現した〔角田 1987, 箕田 1987〕。今後の課題は、

- (1)制約条件評価機構
- (2)自動再設計機構
- (3)設計データ管理機構
- (4)エージェント間通信機構

を実現することである。また、動作アルゴリズムとデータパスの対応付けの自動化も検討したい。

5. おわりに

V L S Iの論理設計における機能分散に着目し、各機能専門のエージェントを協調させながら全体の設計を行う論理設計エキスパートシス

テムを提案した。機能分散としては、構造記述に基づくスタティック回路設計及び動作記述に基づく制御部設計を取り上げ、検討を行ったが、テスト容易化設計及びテスト生成を専門に行うエージェントを加えたシステム等も考えられる。

この研究は、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I C O T第五研究室藤井室長に感謝致します。

参考文献

Bushnell, M. L., et al. "VLSI CAD Tool Integration Using the ULYSSES Environment" Proc. of 23rd Design Automation Conf., pp. 51-61 (1986).

Brewer, F. D. "An Expert-System Paradigm for Design" Proc. of 23rd Design Automation Conf., pp. 62-68 (1986).

Doyle J. "A Truth Maintenance System" Ar-

tificial Intelligence 24 (1979).

de Kleer, J. "An Assumption-based Truth Maintenance System" Artificial Intelligence 28 (1986).

Smith, R. G. "Report on The 1984 Distributed Artificial Intelligence" AI Magazine Fall, 1985.

Camposano, P. "Structural Synthesis in The Yorktown Silicon Compiler" Proc. of VLSI '87, pp. 29-40 (1987).

Moszkowski, B., "Executing Temporal Logic Programs" Cambridge University Press (1986)

角田他 "論理設計エキスパートシステム L O D E X の概要" 情報処理学会第35回全国大会

箕田他 "論理設計エキスパートシステム L O D E X における推論エンジンの開発" 情報処理学会第35回全国大会

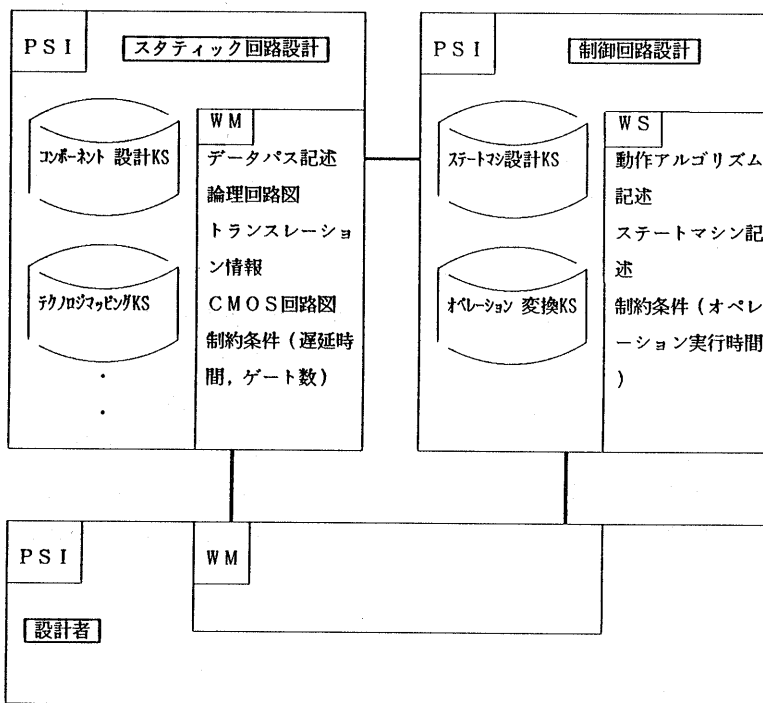


図7 システム構成