設計自動化　45—8
(1988. 12. 16)

論理合成システムLODESの概要と評価

Outline and Evaluation of the Logic Synthesis System LODES

植田 雅彦、松中 雅彦、角田 学、西山 保

Masahiko UEDA, Masahiko MATSUNAKA, Manabu SUMIDA, Tamotsu NISHIYAMA

松下電器産業(株)　半導体研究センター

Matsushita Electric Industrial Co., Ltd.　Semiconductor Research Center

あらまし

VLSI設計工程における論理設計の自動化を目的として論理合成システムLODESを開発している。本システムはハードウェア記述言語、論理式、真理値表、機能図等の機能仕様を入力とし、抽象的な論理のレベルからテクノロジーを考慮したレベルまで階層的に論理合成を行い、最終的に特定のテクノロジーの制約を満たす最適化された論理回路を出力する。本システムはルールベースで実現しているため、設計者が持っている知識を柔軟に取り込んだり、テクノロジーの変更に容易に対応することができる。本稿ではLODESの概要とそこで用いられている論理最適化手法について説明した後、本システムを実際のVLSI設計に適用して評価した結果を報告する。

Abstruct

In order to automate the time consuming logic design step in the VLSI design flow, we are developping an automatic logic synthesis system LODES. This system accepts functional specifications such as hardware description language, boolean expressions, truth tables, functional diagrams etc. and synthesizes them hiearchically from abstruct level to technology dependent level, and finally outputs optimized logic circuits which satisfy constraints of specified technology. This system was realized as an rule-base system so that we can flexibly take in the designers' know-how and easily change the used technology. In this paper, we describe the outline and the logic otimization method of LODES, and then report its evaluation in practical VLSI design.

1. はじめに

LSI回路規模の増大に伴って、設計期間短縮を目的とした設計自動化ツールの開発が急務となっている。しかしながら、現状ではセルの自動配置配線システム等レイアウト設計の一部に自動化が実現されているのみであり、論理設計より上流の設計はほとんどが人手で行われている。我々は従来人手で行われてきた論理設計の自動化を目的として論理合成システムLODES(LOgic Design Expert System)を開発している。[1][2][3][4][5]

本システムはハードウェア記述言語、論理式、真理値表、機能図等の機能仕様を入力とし、抽象的な論理のレベルからテクノロジーを考慮したレベルまで階層的に論理合成を行い、最終的に特定のテクノロジーの制約を満たす最適化された論理回路を出力する。設計者が持っている知識を柔軟に取り込んだり、テクノロジーの変更を容易にするために本システムはエキスパートシステムとして実現している。

以下、第2章ではLODESの概要と特徴について述べ、第3章ではLODESで用いている論理最適化手法について説明した後、第4章で本システムを実際のVLSI設計に適用して評価した結果を報告する。

2. 概要および特徴

2.1システムの特徴

(1)多様な設計スタイルのサポート

設計者が論理設計を行う状況は、機能記述からそれを満たす論理を設計する場合以外に、論理式から設計する場合等、様々である。

本システムではこのような多様な論理設計スタイルに対応してハードウェア記述言語による入力以外に論理式や真理値表入力や機能図入力をサポートし、ユーザーの便宜を図っている。

(2)階層的論理合成

設計者が論理設計を行う場合、抽象的な論理のレベルからテクノロジーを考慮したレベルへと階層的に設計している。

本システムはこのような階層的論理設計の流れに沿って合成を進める。合成の上位レベルにおいても最適化を行うことにより、下位レベルでの最適化処理の負担を軽減すると共に、下位レベルでは除去困難な大域的冗長性を取り除くことができる。

(3)テクノロジーに対する柔軟性

テクノロジーに依存する知識は合成の最終段階であるテクノロジーマッピングに集められ、それより上流のプロセスはテクノロジー非依存になっているので、テクノロジーマッピングの知識を入れかえるだけで容易にテクノロジーを変更することができる。

(4)標準言語をサポート

標準言語を記述言語として採用することにより、ユーザーの言語習得の負担を軽減するとともに、それに基づく各種CADツールを利用できるという便宜が生まれる。現在、LSI設計用記述言語標準化委員会が中心になってHSL-FXをベースに記述言語の標準化が検討されている。

(5)言語非依存性[1]

現在は入力言語としてHSL-FXをサポートしているが、トランスレータに含まれる言語依存知識を入替えることにより、容易に他の言語にも対応することができる。

(6)問合せ機能[3]

言語によって全ての設計情報を記述することはできない。本システムではこのような情報をユーザーに問合せることによって高品質な合成を行っている。

(7)知識ベースとアルゴリズムの共存[2]

本システムは人手設計並の高品質な回路を合成するために、設計者の持っている知識を柔軟に取込めるフレームをデータ構造とするプロダクションシステムで実現しているが、論理最適化処理のようにアルゴリズミックな手法が適した処理に対しては手続き的プログラムで実現することにより高速化している。

(8)ルール作成支援機能[4]

設計者がルールを作成するのを容易にしたり、テクノロジーの変更に対してルールを作り直す手間を軽減するために、セル割付けルールをセルライブラリから自動的に生成する機能を持っている。

(9)論理図自動生成機能[5]

合成結果から論理図を自動生成することが可能である。論理図は外部の論理シミュレーションシステムの図面データになっているので、設計者は論理図を基にシミュレーションを行って合成結果のタイミング検証を行い、必要であればマニュアルで回路修正を行うことが可能である。

詳細なタイミング制約を満たす合成が技術的に困難な現状では、このような検証と修正は不可欠である。

2.2システム構成

本システムはトランスレータ、機能マクロ展開部、テクノロジーマッピング部、論理式･真理値表入力部、機能図入力部、論理図生成部、等のモジュールから構成される。

このうちトランスレータ、機能マクロ展開部、テクノロジーマッピング部がルールベースに基づいて処理される。

2.3処理の概要

本システムはハードウェア記述言語、論理式、真理値表、機能図等の機能仕様を入力とし、最終的に特定のテクノロジーの制約を満たす最適化された論理回路を出力する。本システムの処理フローをFig 1に示す。以下、モジュール別に処理の概要を説明する。

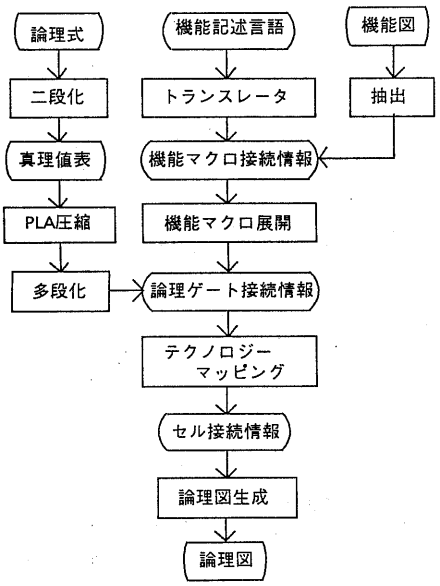

Fig 1 LODES 処理フロー

(1)トランスレータ

HSL-FXで記述された機能記述を解析し、構文レベルでの最適化処理を行った後、デコーダやセレクタ等の機能マクロの接続情報に変換する。

従来の機械的なトランスレーションとは異なり、構文レベルでの最適化を行うことによって記述の仕方による合成回路の品質の劣化を防ぐとともに、言語で表現されない設計情報をユーザーに問合せることによって高品質な合成を行っている。HSL-FXによる記述例をFig.2(a)に示す。

(2)機能マクロ展開

言語から合成されたり、機能図として入力された機能マクロの接続情報に対して大域的最適化処理を行った後、各機能マクロを分析して最適なルールを選択し、テクノロジーに依存しない抽象的な論理に展開する。

言語による機能記述には言語に特有の冗長性が含まれており、これは大域的な性質を持つので後の局所変換手法で取り除くことは困難である。ここでは第3章で述べる論理最適化手法を用いることにより記述の仕方によらずに高品質な回路を合成することが可能である。

入力された機能図の例をFig.2(b)に示す。(これはFig.2(a)の記述からトランスレートした結果と等価である。) また、現在サポートしている機能マクロの一覧をFig.3に示す。

(3)テクノロジーマッピング

機能マクロ展開部や論理式･真理値表入力部で生成された抽象的な論理ゲートの接続情報に対して局所変換手法によって論理最適化を行った後、特定のテクノロジーのセルへの割付けを行う。

一般にルールベースシステムではルールの条件部を満たすデータの検索に要する処理時間の増大が問題になるが、ここではルールの構造化や反転論理の自動生成等の手法[2]を用いることによって処理時間を抑え、その結果、複合ゲート等の複雑なセルを含むセルライブラリーへの効率的な割付けを実現している。

また、遅延制約を付加することが可能で、クリティカルパスを検出して最大論理段数の制約のもとで最小の回路を合成することができる。

Fig.2(a)の記述から合成された最終結果の論理図をFig.2(c)に示す。

(4) 論理式･真理値表入力

入力された論理式はブール代数の定理を使って積和標準形に展開され、真理値表に変換される。得られた真理値表はPLA論理圧縮プログラムESPRESSO[6]によって最適化された後、共通因子の抽出[7]等の手法を使って多段化され、テクノロジーマッピング部に入力される。ここで用いた論理最適化手法については第3章で詳細に説明する。

Fig.4に入力される論理式の例を示すが、本システムでは論理式と共に論理変数の間に成り立つ制約条件を入力することが可能になっている。この制約条件は論理最適化処理の中ではdon't care条件として扱われ、より最適な回路を生成するのに利用される。階層設計のもとでは論理変数は必ずしも独立であるとは限らないので、このような制約条件を付加することは有効である。

(5) 機能図入力

本システムは機能マクロを構成要素として、テクノロジーの制約等を考慮せずに抽象的なレベルで論理設計を行うことを可能とする機能図入力もサポートしている。入力された機能図から機能マクロとその属性及び接続情報が抽出され、機能マクロ展開部に入力される。

設計者は必ずしも記述言語による設計に慣れているとは限らず、そのような場合には従来の論理図入力に基づく論理設計の延長線上にあるこの機能図入力が有効である。

また機能図は直接回路構造を記述するために、記述言語と比較して論理合成結果を細かく制御することが可能である。

```
NAME : RUN_LENGTH_DECODER ;
LEVEL : AUTOMATON ;

EXT : NRESET, MCK, SEL, RUNIN<1:0>, RUNDATA<1:0> ;
INPUTS : .RUNIN , .SEL ;
OUTPUTS : .RUNDATA ;
CLOCK : .MCK ;
RESET : .NRESET ;

REGISTER : ROUT<1:0> , COUNT<1:0> ;

BEHAVIOR-SECTION ;
AUTOMATON:R1_D:^.NRESET:.MCK;
S0:     BEGIN
          ROUT := .RUNIN ;
          IF .SEL & .RUNIN<1> THEN -> S1
                              ELSE -> S0 END-IF
        END ;
S1:     CASE .RUNIN OF
        #0      -> S3
        #1      -> S0
        #2      BEGIN COUNT := 1 ; -> S2 END
        #3      BEGIN COUNT := 2 ; -> S2 END
        END-CASE ;
S2:     IF COUNT = 0 THEN -> S0
                     ELSE COUNT := DEC(COUNT);
                     -> S2 END-IF ;
S3:     -> S3 ;
END-AUTO ;
.RUNDATA := ROUT
END-SECTION;
END ;
```

Fig.2(a)　HSL-FX記述の例

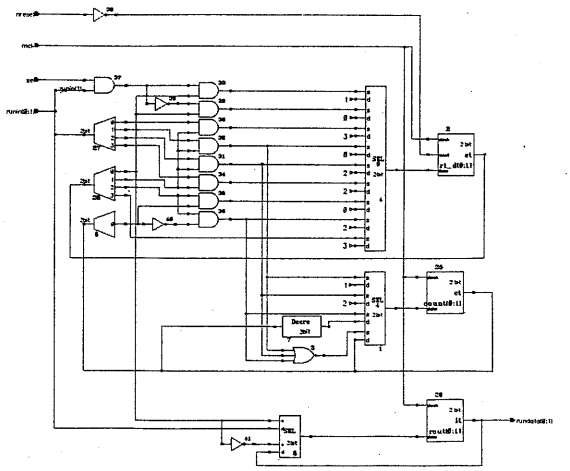

Fig.2(b)　機能図の例

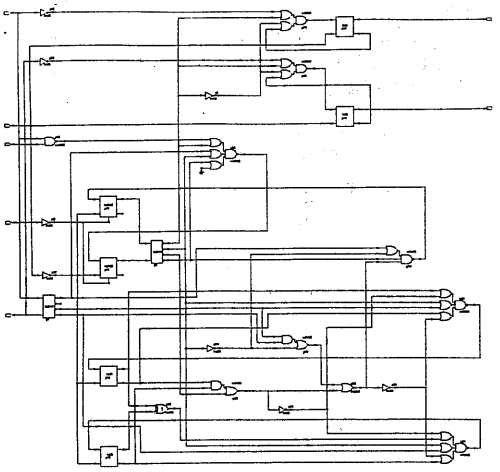

Fig.2(c)　合成結果の例

ゲートマクロ	演算マクロ
AND	加算器
OR	減算器
EXOR	インクリメンタ
インバータ	デクリメンタ
3ステートバッファ	比較器
	シフタ
制御マクロ	
デコーダ	記憶マクロ
エンコーダ	レジスタ
プライオリティ	カウンタ
エンコーダ	RAM
セレクタ	ROM
マルチプレクサ	PLA
ブール式	

Fig.3　機能マクロの一覧

```
X = B1*S1+¯B1*B2*S1+¯B1*¯B2*S0
      +B2*¯S1+¯B2*¯S0        (論理式)
B1*B2 = B1                   (制約条件)
```

↓

```
制約条件なし
   X = ¯S0+S1+¯B1+B2
制約条件あり
   X = 1
```

Fig.4　制約条件付き論理式入力

3. 論理最適化手法

本システムの実設計への適用と評価を通して、論理合成の要素技術としての論理最適化手法の重要性を再認識した。実際の設計において熟練設計者並の回路を合成するには優れた論理最適化手法が不可欠である。

このような認識に基づき、本システムでは論理最適化手法の再検討を行った。ここで用いた方法は、論理合成結果を熟練設計者が設計した回路と比較して不足している機能を抽出し、それを実現する処理を追加するというものである。

3.1従来の論理最適化手法

従来、論理合成手法としては大きく分けて

(a)アルゴリズムによる方法[6]

(b)ルールベースに基づく方法[7]

の2つの手法が検討されてきた。

(a)は2段論理の最小化アルゴリズムをベースに、共通因子の抽出等の手法を用いて多段化する方法であり、真理値表や論理式を使って効率的に表現できるランダム論理的な回路であれば実用的な回路を合成することが可能である。

ただしアルゴリズムベースであるためテクノロジーを考慮した最適化は困難であり、合成は抽象的なレベルに止まる。またデータパス系の回路のように真理値表や論理式では効率的に表現できないような回路では有効ではなく、対象回路は比較的小規模な制御系の回路に制限される。

(b)は局所変換手法に基づく最適化をベースとした方法であり、いくつかの局所的な論理最適化変換をルールとして持ち、入力された回路に繰り返し適用することによって最適化された回路を得る。記号処理に基づいているため対象回路の制約が少なく、またテクノロジーの特性を考慮した最適化や遅延制約を付加した最適化等を比較的容易に実現することが出来る。

ただし局所変換手法に基づいているため大域的最適化が困難であり、回路が大域的な冗長性を含んでいる場合には十分に最適化することが出来ない。その結果、合成結果を見ながら人手で機能記述を修正する等の処理が必要になる。

以上述べたようにそれぞれに一長一短があり普通(a)と(b)を組合せて用いることが多い。[8]

この場合の典型的な手法は、まず入力された回路を真理値表や論理式の形式に変換してアルゴリズミックな手法で最適化した後、ルールベースの局所変換手法でテクノロジーに依存した最適化を行う。

これによって大域的最適化とテクノロジーの特性を考慮した最適化を共に実現することができるが真理値表や論理式を使って効率的に表現できなければならないという対象回路の制約は依然として残っている。

3.2機能記述に含まれる冗長性

言語による機能記述にはFig.5に例を示すような特有の冗長性がしばしば含まれている。

これらの冗長性は、次のような性質を持つ。

```
if COND1 then
    if COND2 then
            ・
            ・
else
    if COND3 then
            ・
            ・
end-if;
if COND4 then
            ・
            ・
```

典型的な機能記述では
(1)独立したif文の条件
　COND1とCOND4
(2)thenとelseに含まれるif文の条件
　COND2とCOND3
に冗長性が含まれることが多い。

Fig.5 機能記述に含まれる冗長性の例

(a)ソフトウェア言語ではあまり問題にならない。

(b)回路全体にかかわる大域的な性質を持つ。

(c)同じ機能を表していても記述の仕方によって合成結果は大きく変わる。

例えば記述をもとに機能シミュレーションを行うような場合にはあまり問題にならないが、この記述をもとに論理合成を行うような場合には合成された回路に大きな冗長性をもたらす。

これらの冗長性はその大域的な性質のために局所変換手法では取り除くことが困難である。例えばFig.5の例で極端な場合を言えば、2つの条件で場合分けするような時にはCOND2とCOND3が全く同じ論理式になるがシステムはそれぞれに対応する回路を合成してしまう。

このような場合に、従来は合成した結果を見ながら人手によって記述を修正するか、もしくは最初から論理合成を意識して冗長性を除いた機能記述を行う必要があった。

しかしながらこのようなハードウェアを意識した低いレベルで記述を行うことはそれ自身わずらわしい上に、機能記述の判読性を損なうという欠点がある。

3.3本システムのアプローチ

本システムにおいても従来と同じく論理最小化アルゴリズムと局所最適化手法を併用している。すなわち論理式や真理値表で入力された回路はアルゴリズミックな手法を用いて最小化された後、ルールによる局所最適化が行われる。

しかし記述言語で入力された回路に対しては従来のように記述全体を論理式や真理値表表現に変換して論理最小化を行うのではなく、前節で述べたような冗長性を含む可能性のある部分に対して選択的に論理最小化アルゴリズムを適用する。

このような論理最適化手法を構文レベルの最適化や機能マクロレベルの最適化と組み合せることによって、本システムでは対象回路を制限することなく大域的に最適化された回路を合成することが可能である。また、論理最小化アルゴリズムは回路規模の増大とともに処理時間が急速に大きくなり実質的に扱える回路規模に制限があるがこのように必要な部分のみ選択することにより、より大規模な回路を扱うことが可能である。

3.4論理最小化アルゴリズム

ここでは本手法で用いている論理最小化アルゴリズムについて説明する。ここで用いている論理最小化アルゴリズムは従来のものと同じく2段論理の最小化と多段化をベースにしたものであるが、実回路を使った評価を通して見つかったいくつかの処理を追加している。処理フローをFig.6に示す。

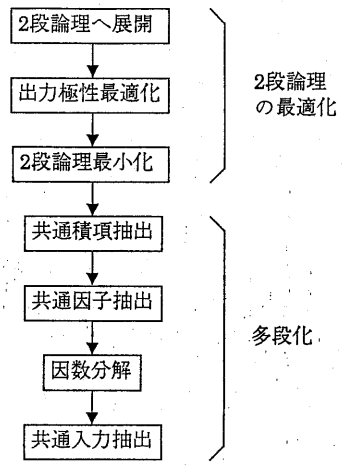

Fig.6 論理最小化の処理フロー

以下、順に処理の概要を説明する。

(1)2段論理へ展開

論理式はブール代数の公式を用いてAND-ORの2段論理へ展開される。この時入力間に論理的制約があればこれも展開しdon't careとして扱う。

(2)出力極性最適化

論理関数の中にはそのまま実現するよりも、その否定を実現する方が容易なものがある。そこで2段論理最小化に先立ち出力位相の最適化を行っている。

アルゴリズムは2重極性特性関数を用いる手法[9]を採用しているが、積項数を最小にする戦略では多段化した後必ずしも回路規模が小さくなるとは限らないため、接続数を最小にする戦略を採った。すなわち[9]の手法で求めたいくつかの候補について接続数を計算し、それが最小になるものを最適出力極性とした。

(3)2段論理最小化

2段論理を(2)で求めた出力極性のもとで最小化を行う。アルゴリズムはESPRESSO[6]をそのまま用いている。

(4)共通積項抽出

多段化の第一ステップとして、2段論理の各出力間で共通の積項があればそれをくくり出す。

(5)共通因子抽出

このステップと次のステップは因子をくくり出すことによって論理の最小化を行う。通常くくり出せる因子は複数存在するのでその中で最適なものを求める必要がある。複数の論理関数に共通の因子があればそれを優先してくくり出すのが本ステップの共通因子抽出である。

なお共通因子でなくともFig.7(a)に示すように、ある論理関数の因子の否定が別の論理関数の因子かキューブになっていれば、その因子も優先してくくり出される必要がある。ここでは各因子の否定を求めることにより、このような共通因子の抽出も行っている。

(6)因数分解

共通因子でなくとも論理関数の因子をくくり出すことは論理を最小化する上で有効である。ここでも最適因子を求めることが重要である。

通常セルライブラリはEXORゲートを含んでおり、EXORの抽出は回路のコストを下げるのに効果がある。ここではFig.7(b)に示すように候補である因子の中からEXORになるものを探し出し、それを優先してくくり出している。

(7)共通入力抽出

最後にいくつかの積項に共通の入力があればそれをくくり出す。

3.5新たに追加した機能

本システムは、論理合成結果の人手設計結果との比較を通して見つかった、いくつかの不足機能を追加している。それらの機能を以下に示す。

(a)出力極性最適化機能
(b)因子とその否定の抽出
(c)EXOR抽出

これらの機能は常に必要であるわけではないが、従来の論理最小化アルゴリズムに追加することで、より最小化された論理を合成することができる。

4. 本システムの評価

実際の設計データをもとに本システムで合成された回路の品質と処理時間を評価した。なお、評価はSUN4(10MIPS)上で行った。

4.1論理最適化手法の評価

前章で説明した本システムの論理最適化手法の評価を行った。ここでは、3.5で説明した機能を入れた場合(新手法)と入れない場合(旧手法)で、論理式・真理値表から合成された結果の回路規模(トランジスタ数)の比較を行った。Table1にこれらの機能を入れることによってどれだけ回路削減に効果があるかを、多段化処理の後(a)とテクノロジーマッピングの後(b)で評価した結果を示す。

この結果から次のことがわかる。

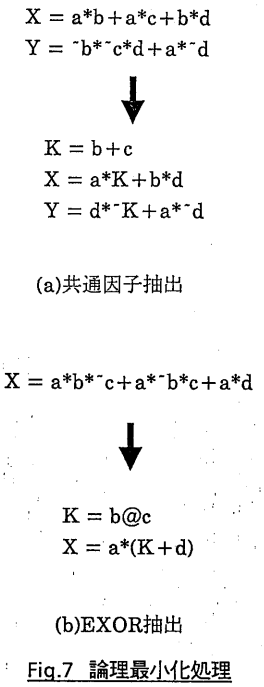

Fig.7 論理最小化処理

Table 1 論理最適化手法の評価

回路	旧手法		新手法		新/旧	
	a	b	a	b	a	b
E1	234	175	212	158	0.91	0.90
E2	404	248	352	214	0.87	0.86
E3	424	319	384	227	0.91	0.71
E4	710	550	574	429	0.81	0.78

(1)新たに追加した機能は有効に働き、回路削減にかなりの効果がある。

(2)その効果はテクノロジーマッピングを行った後でも依然として残っている。

(a)と(b)の回路規模を比較してわかるように、テクノロジーマッピングでの局所変換によってかなりの最適化が行われている。しかしながら(2)の結果は、これらの機能がもたらした回路削減効果は、このような局所変換による最適化では実現されていなかったタイプのものであることを示している。

4.2回路の品質の評価

実際の設計データを使って本システムで合成した結果と人手で設計した結果の比較を行った。比較項目は、回路規模(トランジスタ数)、ゲート段数(遅延の目安)、使用されているセル種類数である。

(1)論理式･真理値表入力

まず論理式･真理値表をもとに設計した場合の結果をTable2に示す。ここでは回路規模とゲート段数の比較を行った。回路中C1~C4は制御回路であり、D1とD2は演算器の一部である。

Table 2　LODESと人手設計の比較
(論理式･真理値表入力)

回路	LODES		manual		LOD /man	
	TR	LEVEL	TR	LEVEL	TR	LEVEL
C1	434	8	503	7	0.86	1.14
C2	114	6	110	5	1.04	1.2
C3	150	8	259	5	0.58	1.6
C4	40	5	58	4	0.69	1.25
D1	112	5	84	9	1.33	0.55
D2	429	8	389	12	1.10	0.66

この表から次のようなことがわかる。

(a)制御系の回路ではLODESの結果は人手設計と比較して回路規模では小さくゲート段数では大きくなっている。

(b)逆に演算器系の回路ではLODESの結果は人手設計と比較して回路規模では大きくゲート段数では小さくなっている。

この結果から、制御系の回路ではLODESは設計者を上回る高品質な回路を合成しているが、演算器系の回路の設計ではまだ設計者のレベルには達していないことがわかる。

これは、演算器では同じ回路が繰り返して使われるため面積効率への要求が厳しく、設計者は時間をかけて十分な最適化を行うのに対して、制御系の回路では限られた設計時間内に時間のかかる最適化を十分に行うことは困難であるためである。

(2)記述言語入力

次に記述言語をもとに設計した場合の結果をTable3に示す。ここでは回路規模とセル種類数の比較を行った。回路L1~L6は全て制御系の回路である。

この結果から次のことがわかる。

(a)制御系の回路に対しては記述言語で入力した場合でも人手設計に対して優位な結果を得ている。これは第3章で説明した本システムの論理最適化手法が有効に働いているためである。

Table 3　LODESと人手設計の比較
(記述言語入力)

回路	LODES		manual		LOD /man	
	TR	セル種	TR	セル種	TR	セル種
L1	571	18	578	18	0.99	1.0
L2	319	16	392	17	0.81	0.94
L3	392	14	512	12	0.77	1.17
L4	108	8	124	9	0.87	0.89
L5	302	8	308	5	0.98	1.6
L6	556	16	572	14	0.97	1.14

(b)セルの種類数の比較からLODESは人手設計と同じ程度の種類のセルを用いていることがわかる。このことは本システムがテクノロジーの特性を十分に利用した最適化を行っていることを示している。

4.3処理時間の分析

次に実際の設計データにもとづいて本システムの処理時間の分析を行った。分析項目は、回路規模(トランジスタ数)と処理時間の関係、および言語記述の場合の記述量(行数)と処理時間の関係である。

(1)回路規模と処理時間の関係

Fig.8に回路規模と処理時間の関係を、論理式･真理値表入力の場合と記述言語入力の場合で区別して示す。

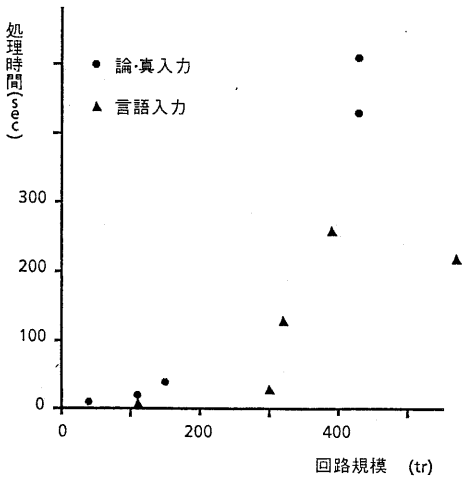

Fig.8　回路規模 vs 処理時間

この結果から次のことがわかる。

(a)回路規模と処理時間の関係は、論理式･真理値表で入力したか記述言語で入力したかにはかかわらない。これは論理合成の処理時間のほとんどが最終ステップのテクノロジーマッピングで費やされていることに原因がある。

(b)処理時間はだいたい回路規模の2乗に比例しており、より大規模な回路を実用的に扱うには、なんらかの高速化手法を導入する必要がある。

(2)記述量と処理時間の関係

次に言語入力の場合の記述量(行数)と処理時間の関係をFig.9に示す。

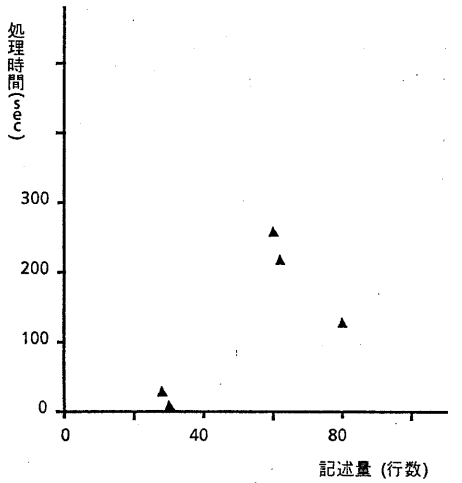

Fig.9 記述量 vs 処理時間

ここでも記述量が増加するにつれて処理時間も増大していく傾向にある。これは用いた評価データが制御系回路であるために、記述量が合成結果の回路規模と比例に近い関係にあるためである。データパス系の回路では合成される回路規模は記述量よりもデータのビット幅に依存するので記述量と処理時間はあまり関係しないと思われる。

5. まとめと今後の課題

LODESの概要と特にそこで用いられている論理最適化手法、および実際の設計データを使って評価した結果について報告した。

本システムは本文中で説明した様々な特徴や高機能の論理最適化手法によって、記述言語や機能図、論理式、真理値表等の機能記述から高品質に論理合成を行うことができる。

評価した結果、制御系の回路に関しては論理式･真理値表入力であるか記述言語入力であるかにかかわらず、多くの場合に人手設計を上回る合成結果を得た。なお、ここで用いた回路は規模は小さいが全て実際の設計データであり、本システムの実用性を保証している。

しかしながら、演算器系の回路に関しては未だ設計者のレベルには到達していないことがわかった。

処理時間について言えば、今回の評価からはだいたい合成結果の回路規模の2乗に比例するという傾向がわかった。これは、より大規模な回路の合成を考えた場合に問題になることが明らかであり、何らかの高速化の検討が必要である。

ただし、ここで評価に用いた回路は全てランダム論理的な制御系回路である。現在の典型的な論理合成手法では、制御系回路は回路規模のわりには最も処理時間を必要としている。

今後の課題として、より大規模な回路に適用して評価を行い、不足している知識の獲得と問題点の洗い出しを行うことによって、システムの実用性を高めていく予定である。

その他の重要な検討課題としては、

(1)データパス系回路の合成

(2)高速化

等があげられる。

謝辞

最後に本研究に際し、日頃御指導や貴重な御助言を頂きました当社デバイス開発研究所所長 石原健氏、ならびに本研究開発に惜しみない御協力を頂いた松本典子、泉野祥吾の各氏に深く感謝致します。

参考文献

[1]松中 他：言語非依存型論理合成の一手法
情報処理学会第35回全国大会(1987)
[2]植田 他：論理合成システムにおける論理最適化手法の検討、情報処理学会第35回全国大会(1987)
[3]松中 他：論理合成エキスパートシステムLODESートランスレータにおける問合せ機能、情報処理学会第36回全国大会(1988)
[4]松本 他：論理合成エキスパートシステムLODESー回路変換における知識表現ー、情報処理学会第36回全国大会(1988)
[5]泉野 他：論理図自動生成の一手法
情報処理学会第37回全国大会(1988)
[6]R.Braton et al. : Logic Minimization Algorithms forVLSI Synthesis, Kluwer Academic Publishers, 1984
[7]R.Braton et al. : The Decomposition and Factorization of Boolean Expressions, ISCAS-82, 1982
[8]A.deGeus et al. : Optimization of Combinational Logic using a rule based expert system, Journal of IEEE Design and Test, 1985
[9]T.Sasao : Input Variable Assignment and Output Phase Optimization of PLA's, IEEE Trans. on Computers. Vol. C-33, 1984