人 工 知 能 77－7
（1991. 7. 17）

協調型論理設計エキスパートシステム co-LODEX

箕田依子　澤田秀穂　滝沢ユカ　丸山文宏　川戸信明

富士通株式会社

エージェントと呼ぶ自律的に動作する実行単位を設けて、部分回路を並列に、回路全体に関する規模と遅延時間の制約条件を満たすように設計する、論理設計システム co-LODEX について報告する。エージェントは、部分回路の属性値または制約条件違反情報を通信し合い、自分自身の設計範囲を制限した再設計を繰り返す並列協調機構により、全体に対する制約条件を満足する部分回路を設計する。システムを試作し実験した結果、ある程度以上の規模の回路では、エージェント数程度の台数効果が得られた。

co-LODEX: Cooperative Logic Design Expert System on a Multiprocessor

Yoriko Minoda　Shuho Sawada　Yuka Takizawa　Fumihiro Maruyama　Nobuaki Kawato

Fujitsu Limited

1015, Kamikodanaka Nakahara-ku, Kawasaki 211, Japan

co-LODEX divides the whole circuit to be designed into sub-circuits in advance and designs each sub-circuit on each agent to take advantage of parallel processing. It can design a circuit that satisfies a given maximum gate count and a maximum delay. Agents improve their quality of the design with their cooperation. Global redesign takes place by agents exchanging design results in terms of gate counts and delays or justifications for constraint violations. co-LODEX has been implemented in KL1 on a Multi-PSI. Speedup of about the number of agents has been obtained for a large circuit in preliminary experiments.

1．はじめに

我々は、ハードウェアの機能レベルの仕様から、回路規模と遅延時間に関する制約を満たす回路を設計する論理設計エキスパートシステムco-LODEX（cooperative Logic Design EXpert system）を開発している。まず、与えられた制約条件に対して設計結果を評価し、違反が検出されると、制約条件を満たすよう設計のやり直し（再設計）を行なう方式の研究を行なった。この研究において、論理式で表現する制約条件違反情報（NJ:Nogood Justification）を考案し、これを利用して再設計を実行するシステムを試作し、NJが探索空間の絞り込みに効果があることを確認した［1］。現在は、並列協調方式を用いて、このシステムを並列処理システムに拡張している［2］［3］。その特徴は、エージェントと呼ぶ自律的に動作する実行単位を設けて、部分回路を並列に、全体に対する制約条件を満足するように詳細化することである。

本稿では、まず、co-LODEXの概要として設計の流れとNJを利用した再設計機構を紹介し、並列処理システムに拡張するための並列協調方式について説明する。次に、Multi-PSI上にKL1言語を用いて実装した試作システムについて述べ、負荷分散方式の検討のための実験とその結果について報告する。

2．co-LODEXの概要

2．1　論理設計の流れ

co-LODEXにおける設計の流れを図1に示す。入力は、機能レベルの動作仕様と、レジスタ、演算器などの機能ブロックと機能ブロック間のデータ

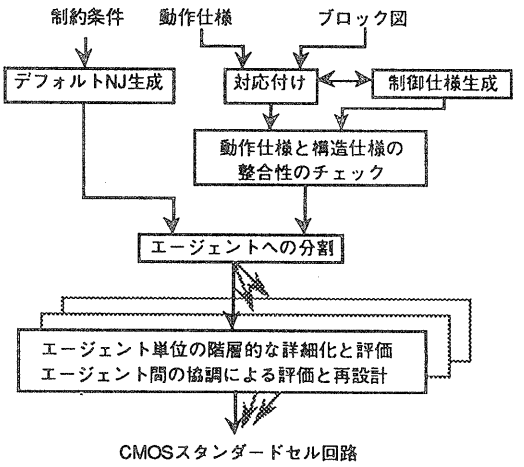

図1　設計の流れ

の流れを示すブロック図、生成した回路が満たすべき制約条件（ゲート数と遅延時間の上限値）の3種類である。動作仕様は、状態遷移表現に基づくレジスタ転送レベルの仕様記述言語で記述したものであり、これだけでも仕様として十分であるが、設計者が具体的な機能ブロックをイメージしている場合は、ブロック図を入力することにより、設計者の意図を反映させることができる。

初めに、動作仕様から機能レベルのオペレーションを抽出し、これらのすべてがブロック図で表されたデータパス上で実行可能なことを検証する。この過程において、機能ブロックとその接続関係、および制御回路の仕様を決定する。これと平行して、制約条件を、デフォルトNJと呼ぶ、制約条件と等価な不等式に書き換える。

次に、機能ブロック（制御回路も制御ブロックと呼ぶ機能ブロックのひとつである）とその接続情報に書き換えられた仕様を分割してエージェントに割り当てる。すなわち、担当する機能ブロックの仕様に対応する部分仕様を生成する。

仕様の分割が終了した後、部分仕様とデフォルトNJを各エージェントに通知する。エージェントは協調して、部分仕様の詳細化と評価を繰り返すことにより、制約条件を満たす回路を設計する。制約条件違反を起こした構成方法にはNJを記録しておく。出力は、CMOSスタンダードセルの接続情報である。もし、制約条件を満足する回路を生成することができなかった場合は、違反した制約条件を提示する。ユーザが制約条件を変更することによって、再設計を開始することができる。制約条件の変更による再設計においては、蓄積された設計結果（機能ブロックの属性値）とNJを再利用する。

2．2　階層的な評価と再設計

詳細化と評価・再設計は、設計の階層と同じ階層構造上に表現する。このとき、NJを階層上のノードに記録することによって、評価・再設計を制御する。

デフォルトNJは、制約条件に関係する属性値を示す変数の和を左辺、制約値を右辺とする、制約条件と等価な不等式である。この不等式が成立することが制約条件違反の必要十分条件になる。デフォルトNJを成立させる構成方法には、その構成方法の属性値でデフォルトNJの対応する変数を具体化したNJを記録することで、組み合わせを禁止する。設計中に生成されるNJは、デフォルトNJを起源とし、その成立が制約条件違反の十分条件になる論理式である。

評価と再設計の制御は、「NJの展開」と「NJの合成」と呼ぶ機構に基づく。例を用いて評価・再設計を説明する。

1. 3つの機能ブロックA、B、Cからなる回路を評価したところ、デフォルトNJ(a)が成立した。

   Aの面積＋Bの面積＋Cの面積＞面積制約値･･･(a)

2. 成立したNJ(a)の左辺の変数から、Aを再設計要素として選ぶ。
3. 再設計要素Aを構成要素A1とA2で置き換える。この操作を「NJの展開」と呼ぶ。

   （以下“の面積”は省略）

   A1＋A2＋B＋C＞面積制約値･･･(b)

4. NJ(b)の左辺の変数から、A1を再設計要素として選んだところセルで実現されているため、NJ(b)のA1を属性値（ゲート数）で置き換えたNJ(c)を構成方法に記録する。

   80＋A2＋B＋C＞面積制約値･･･(c)

5. A1の他の構成方法（90ゲート）でも違反するので、NJ(d)を違反した構成方法に記録する。

   90＋A2＋B＋C＞面積制約値･･･(d)

6. A1の構成方法を変えても制約条件を満足できない。すべての構成方法で違反したとき、構成方法に記録されたNJの論理積を機能ブロックに記録する。この操作を「NJの合成」と呼ぶ。NJ(c)と同一のNJ(e)をA1に記録する。
7. NJ(e)について1.以降と同様の手順で再設計を行なう。

NJを成立させなくなった時点で、途中すべての構成方法を禁止した機能ブロックからNJを成立させないような構成方法を選ぶ作業を開始する。すべての機能ブロックでNJを成立させてしまうときは、すべての変数が数字で具体化されたNJが生成されているはずである。この値は、設計可能な最小値である。

3．並列協調方式

並列協調アルゴリズムは、以下の二つの要件を満たす並列協調の定義から出発した。

(1) 与えられたゴールをサブゴールに分割し、サブゴールを複数のプロセッサで並列に達成する。

(2) 全体のゴールが達成されないとき（例えば、あるサブゴールが達成できないために全体のゴールが達成できない場合）、各プロセッサにおいて自発的に軌道修正を行なう。

ゴールGをサブゴールg_1, $\cdots g_n$に分解するには、計算量f(G)に関して、$f(G) >> f(g_i)$となる分解を選ぶ。つまり、規模の縮小による高速化を図る。一般的にゴールGを満たすためには、サブゴールの調整・再試行の繰り返しが必要である。サブゴールを一括して管理・調整する方式は、管理部に負荷がかかりネックになることが考えられるから、得策でない。従って、協調における各エージェントの自律的な軌道修正が必要であると考える。

この並列協調を論理設計システムに適用すると、次のようになる。

(1) ゴールは与えられた制約条件を満足する回路を設計すること。設計すべき仕様を分割し、各部分回路を並列に設計する。部分回路の設計モジュールをエージェントと呼ぶ。

(2) 全体の制約条件が満足されていないとき、各エージェントが設定した制約条件を変更して再設計する。

回路面積と遅延時間に関して与えられた制約条件は、部分回路に分割して設定することはできない。そこで、部分回路の設計結果（属性値）とNJを交換することにより、他のエージェントの設計値、または、設計不可能な範囲を考慮することで、自身の設計範囲を制限する。すなわち、自身の設計範囲を制限することを、エージェントの制約条件の変更とする。

3．1　並列協調アルゴリズム

図2に沿って説明する。並列協調アルゴリズムは、エージェントに部分仕様とデフォルトNJが通知されたとき始まる。以下の処理は、各エージェントが独立に（並列に）行なう。

【1】蓄積しているNJ（初めはデフォルトNJだけ）を成り立たせないように、部分回路を設計する。他のエージェントが担当している部分の属性値は0として扱う。

【2】少なくともひとつのエージェントが設計に失敗すると、制約を満たす回路は存在しない。すべてが成功したことを確認する。

【3】各エージェントの設計結果（部分回路の属性値）を通知し合う。

【4】ひとつのエージェントでも前回（【4】）と異なる設計結果を出していることを確認する。

【5】通知された設計結果を代入しても、デフォルトNJが成立しなければエージェントは成功。失敗したエージェントがないことを確認する。

【6】蓄積しているNJを成り立たせないように、部分回路を設計する。他のエージェントが担当している部分は【3】で通知された値を使用する。

【7】少なくともひとつのエージェントがNJを成立させない結果を得られたならば、成功した結果で置き換えて、【3】へ。これより、違反が

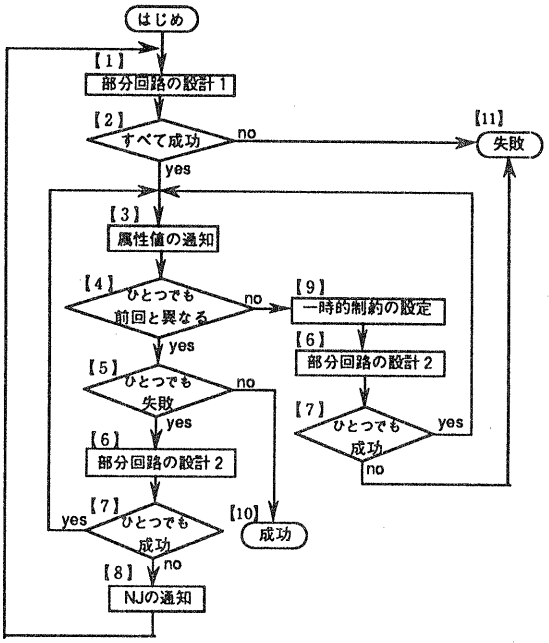

図2　並列協調アルゴリズム

生じている制約の数は減少する。失敗したエージェントには、担当する部分回路に相当する変数を含まないNJが生成されている。

【8】NJを関係するエージェントに通知する。通知されたNJは、「NJの組み合わせ」を行ない、新しいNJとして追加する。

【9】違反している制約のうち、関係するエージェントが最少のものをひとつ選ぶ。この制約を厳しくした一時的制約を与えることにより、強制的に異なる設計結果を導きだす。

【10】部分回路を結合する。

【11】現在の制約条件では設計不可能である。制約条件を緩和することにより、蓄積されたNJのいくつかが成立しなくなり、再設計を開始できる。

エージェント間の同期は、属性値の通知とNJの通知によって実現されている。例えば、【5】では、自分自身（エージェント）が成功したとき、他の（少なくてもひとつの）エージェントが失敗するか、あるいは、他のすべてのエージェントが成功するまで処理を待つ。

3．2　NJの組み合わせ

NJの組み合わせは、他のエージェントの設計不可能範囲から、自分自身の設計範囲を制限するNJを生成する処理である。組み合わせたNJを追加することにより、他のエージェントが担当している部分を0として扱う部分回路の設計（図2の【1】）でも、他のエージェントの設計不可能範囲を考慮できるようになる。

組み合わせの例を示す。3つのエージェントに関係する制約があり、すべてのエージェントで再設計不可能であるとする。このとき、次のNJが生成されている。A1、A2、A3はそれぞれエージェントの属性値に対応する変数である。

エージェント2：A1＋10＋A3＞制約値

エージェント3：A1＋A2＋8＞制約値

エージェント1は、これらのNJからA1＋18＞制約値というNJを生成する。このNJからA1を除く部分に最低でも18必要なことがわかる。

4．試作システム

試作システムは、co-LODEXのエージェントの並列協調方式と、エージェントが独立に（並列に）行なう設計―評価―再設計の過程を実現したものである。入力は、分割されるべき回路の全体仕様と、エージェントへの割り当て、制約条件と等価なデフォルトNJである。本システムは、並列論理

型言語KL1［4］で記述し、Multi-PSI［5］上に実装した。Multi-PSIはMIMD型マシンで、要素プロセッサ（以下、PE）が最大で６４台、２次元メッシュ状のネットワークで結合されている。

Multi-PSI上のKL1言語の特徴を考慮して、設計の階層構造に現われる各要素と、NJの合成・組み合わせなどのデータ管理モジュールをプロセスとして実現した。各プロセスはメッセージ通信をしながら処理を進める。メッセージの受信により処理が励起される永久プロセスの表現を使うと、設計の上限値が通知されると評価・再設計を開始する機能ブロックなどが自然に表現できる。PE間のメッセージ通信のコストは、PE内の処理に比べて非常に高い。そこで、エージェントにおいては他のエージェントの詳細な情報を参照しないこと、各機能ブロックには（NJをそのまま与えるのではなく）制約値と他のエージェントの属性値から算出した上限値を与えることにする。並列に実行できる過程は、エージェントにおける部分回路の設計、その中でも各機能ブロックの設計と複数の経路の遅延時間の計算である。逐次的になる過程は、エージェントとコンポーネント設計のオルタナティブにおける上限値の算出と、NJまたは上限値に対する下位の機能ブロックの属性値の評価である。

設計の階層を構成するプロセスの概念図を図３に示す。楕円はプロセスを表し、矢印は入力／出力ストリームを表す。図３は、設計が進行したある状況を表すものであり、設計の初期状態では、入出力のプロセスだけが存在する。入出力以外のプロセスは、設計の進行に伴って生成する。

入出力プロセスは、入力された回路の仕様を分

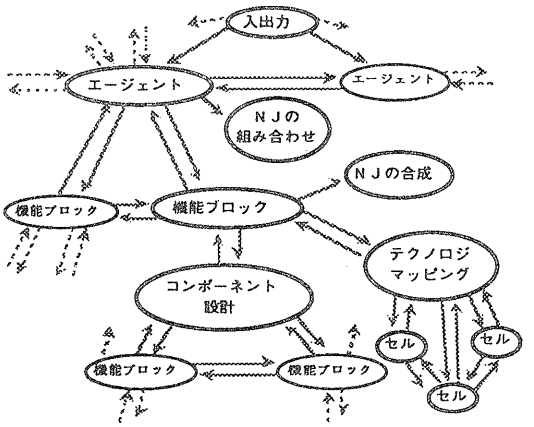

図３　プロセスの概念図

割し、分割した回路の部分仕様とデフォルトNJを各エージェントに通知する。

エージェントプロセスは並列協調方式に従う。部分仕様が通知されると、制約条件を考慮しない詳細化を開始する。詳細化のために、機能ブロックプロセスを生成する。デフォルトNJが通知されると、エージェントの属性に関する上限値を示す、エージェントの制約条件（担当する機能ブロックに関する局所的な制約条件）を生成する。設計の進行に伴って、他のエージェントから属性値またはNJが通知されたときも、同様な制約条件を生成する。エージェントの制約条件をもとに、機能ブロックの属性に関する上限値を算出し、担当する機能ブロックへ一斉に通知する。上限値は各機能ブロックの設計値に応じて変更する。

NJの組み合わせプロセスは、他のエージェントから通知されたNJを組み合わせたNJを生成する。NJの組み合わせプロセスはエージェント毎に生成する。

機能ブロックプロセスは、設計ルールを参照し、詳細化することにより設計を行なう。すなわち、

ひとつ下位のオルタナティブ（コンポーネント設計／テクノロジマッピング）のプロセスを生成し、1レベル詳細化した機能ブロック／セルの仕様をオルタナティブプロセスに通知する。上限値が通知されたときは、下位のオルタナティブに上限値を越えない設計を依頼する。設計に成功したときは結果を上位のプロセスに返し、このオルタナティブをIN状態として記録する。失敗したオルタナティブもOUT状態として記録しておく。すべてのオルタナティブが上限値を越えたとき、NJの合成プロセスにより、これ以上厳しい条件のときの再設計を禁止する情報を生成し、設計可能な最小の属性値を返す。NJの合成プロセスは機能ブロック毎に生成する。

コンポーネント設計のオルタナティブプロセスは、機能ブロックを1レベル詳細化した構成方法に対応する。上位の機能ブロックプロセスから与えられた上限値を越えないような、下位の機能ブロックの上限値を算出し、機能ブロックへ一斉に通知する。設計の進行に伴って、生成する上限値は下位の機能ブロックの設計値に応じて変更する。

テクノロジマッピングのオルタナティブのプロセスは、セルへのマッピングの方法に対応する。構成要素のセルが通知されると、セルのプロセスを生成する。与えられた上限値を越えるか否かの結果と、設計結果（ゲート数と遅延時間）を上位の機能ブロックプロセスに返す。セルプロセスへの遅延時間の計算の依頼は、パスの始点となるすべてのセルへ一斉に通知する。セルプロセスには、セルの特性値、接続関係を記録する。

表1　対象回路

回路規模	回路1	回路2
データパスに現われる機能ブロック数（個）	28	16
最大ゲート数	2660	1638
最小ゲート数	2436	1224
階層の最大深さ（階層）	4	2
階層の最大広がり（倍）	183	2

5．実験と評価

並列協調の効果を得るためには、エージェントの処理がなるべく均等になるような部分仕様への分割と、PEを有効に使用できるマッピングが必要である。そこで、（1）エージェントの数と（2）PEへのマッピングを変更したときの設計時間を計測した。実験に用いた例題は、「2階微分方程式を解く回路（以下、回路1）」と「2点間の距離を整数で近似する回路（回路2）」である。二つの例題の、データパスに現われる機能ブロック数、ゲート数、設計の階層の深さ、データパスに現われる機能ブロックに対する最下位の機能ブロックの総数の最大値を表1に示す。

5．1　実験（1）

多くのエージェントを設け、個々のエージェントが担当する機能ブロックの数を減らすことは、エージェントの設計作業を小さくし、並列実行の効果が期待できる。しかし、属性値の通知、NJの通知といったエージェント間通信が全体として増加するので、並列実行の効果を打ち消す要因となる。エージェントへの分割において、高い台数効果を得るためには、エージェント間通信を抑える必要がある。

遅延時間制約は1クロックで信号が転送される、データパス上のいくつかの機能ブロックを通るパスに関する制約である。なるべく少ないエージェ

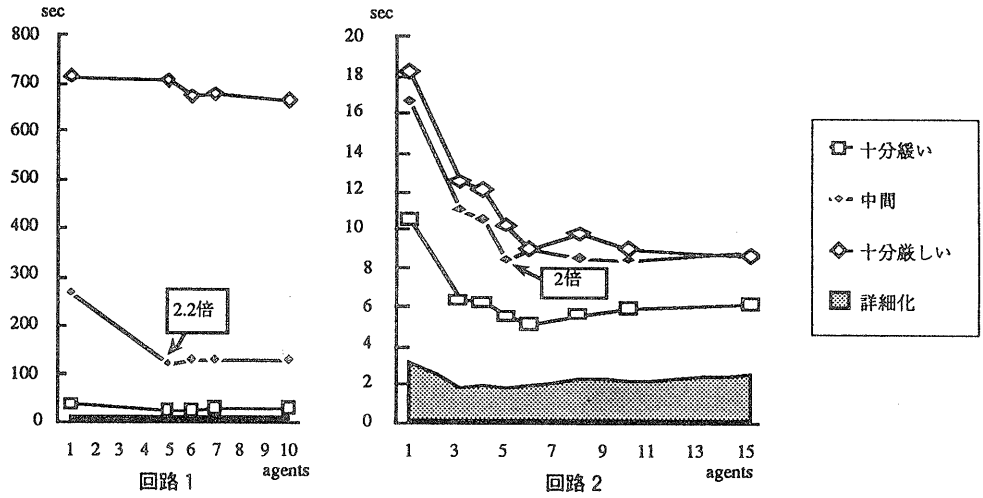

図4　エージェント数と設計時間

ントに分割すると、ひとつの遅延時間制約に関係するエージェントが少なくなるから、エージェント間通信を抑制できると考えられる。そこで、我々はエージェントへの分割の方針を、『クリティカル・パス（の候補）上の機能ブロックはなるべく少ないエージェントに分ける』とする。この分割の方針において、設定するエージェント数が問題となる。

3種類の面積制約について、エージェント数を変えたときの設計時間を図4に示す。使用したPE数は、各エージェントで1台と入出力プロセスが使用する1台である。制約が十分緩いとは、一度の設計で制約を充足する回路が得られるような場合であり、このとき再設計は行なわれない。制約が十分厳しいとは、制約を充足する回路が存在しない制約を与えた場合であり、すべてのエージェントで再設計を行なった後、NJを出力する。中間は、再設計を行なうことによって制約を充足する回路が得られるような場合の一例である。下部の網かけの部分は、仕様を分割してエージェントに通知し、各エージェントが最初の構成方法でセルに詳細化するまでの時間である。

この結果から、次のことがいえる。

- 設計時間を数分の1に短縮できるようなエージェント数が存在すること
- 設計時間の違いは詳細化ではない部分の影響を受けていること

詳細化は、各エージェントが並列に（独立に）行なえる処理であるが、エージェント数を増やすことによって、エージェントの負荷が減少する以上にエージェントを生成するためのオーバヘッド（他のエージェントとのストリーム、接続情報の保持など）が増大していると考えられる。これに対して評価・再設計は、エージェント間の協調に比べてエージェント内の処理の負担が大きいために、エージェントの担当する回路規模が小さいほど時間が短縮されていると考えられる。

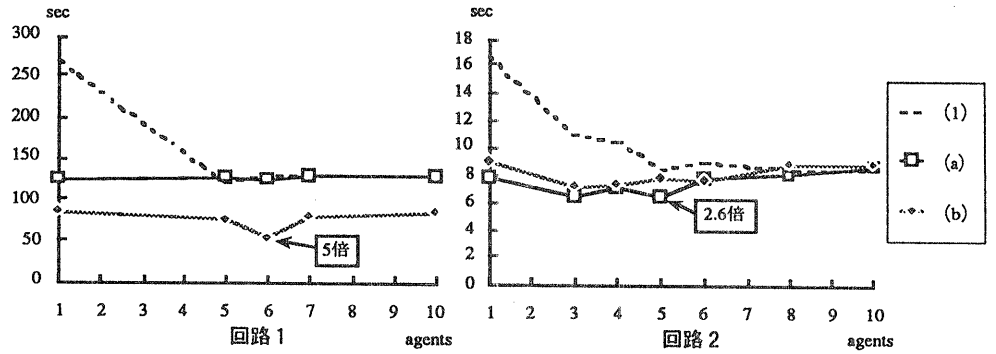

図5　機能ブロックの分散と設計時間

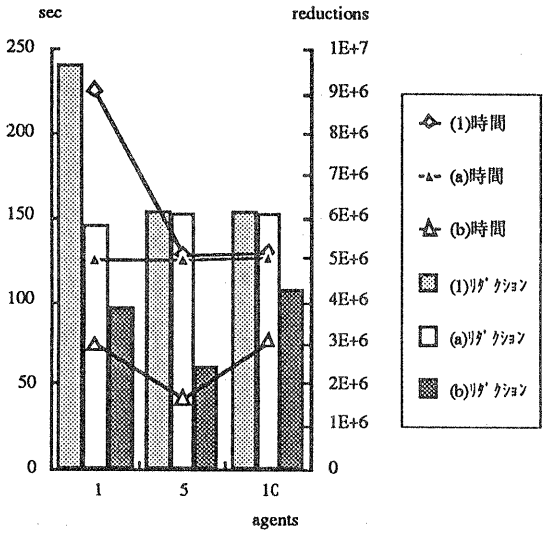

図6　リダクション数と設計時間

5．2　実験（2）

エージェントの処理の分散を考える。エージェントの処理のなかで分散の効果が期待できる部分は、並列に実行できる機能ブロックの詳細化と遅延時間の計算である。そこで、機能ブロックの分散を試みる。図5にこの結果を示す。(a)はデータパス上の機能ブロックだけを分散させた場合（詳細化により生成されるプロセスは親プロセスと同一のPEを使用）、(b)は階層上のすべての機能ブロックを分散させた場合である。使用したPEは16台である。破線はエージェントだけを分散させた実験（1）の結果である。再設計を行なうことによって制約を充足する中間の制約を与えている。

この図は、エージェントを1PEで実行した場合に比べて設計時間が短縮できることを示している。この傾向は回路規模が大きいほど顕著である。また、実験（1）で良い結果を得られたエージェント数はエージェント内の分散方法を変えても良い結果を示している。

回路1では(b)、回路2では(a)という異なる分散方式で高い台数効果を得ている。これは、階層木の形と関係が深いと考えられる。回路1では3階層以降で枝の広がりが183になるような機能ブロックがあるのに対して、回路2は階層の深さ・広がりが小さいためであろう。

最後に、分散方法を変えたときの総計算量が大きく変わる例を紹介する。図6は、回路1の中間の制約を与えたときのリダクション数と実行時間を示すものである。5エージェントのときの分散方法(b)のリダクション数が極端に少なくなっている。これは、制約を満足する組み合わせを早く見つけるために、エージェント内の再設計回数が少

表2　繰り返し数

面積制約	逐次	並列協調
1400	3	1
1300	13	2
1000	17	2

ないことによると考えられる。エージェント内での評価・再設計は逐次的な処理を含んでいるが、この順序を決めているのは並列に行なう詳細化の応答時間である。並列に行なえる部分が処理待ちにならないように分散されることによって、再設計要素の選択が適切な順序になり、総計算量が減少しているということができるのではないだろうか。

5．3　逐次方式との比較

NJを利用した再設計機構により逐次的に設計したときの繰り返し数（再設計回数）と、並列協調における繰り返し数（並列協調アルゴリズムのループ回数）を表2に示す。表2は、回路2で遅延時間制約が十分緩いときの実験結果である。部分回路に分割した並列協調機構により、探索空間の絞り込みが適切に行なわれることを示す一例である。

6．おわりに

適切なエージェントへの分割によって、ある程度以上の規模の回路では、エージェント数程度の台数効果が得られることを確認した。さらに、PEを有効に利用することにより、もっと速く処理できる可能性がある。そのために、エージェント内の処理の並列化や、論理設計における効果的な負荷分散方式を実現していく予定である。

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なわれているものであり、御支援頂いたICOTの生駒研究部長代理（現在、NTTデータ通信株式会社）、新田第七研究室長に深く感謝いたします。

参考文献

[1] 丸山他「評価・再設計機構を備えた論理設計支援システム」信学論　A Vol. J72-A No.8 pp.1172-1180, 1989.8

[2] 箕田他「協調型論理設計エキスパートシステムco-LODEX—概要—」情全国大会 1991.3

[3] 澤田他「協調型論理設計エキスパートシステムco-LODEX—試作—」情全国大会 1991.3

[4] K.Ueda, T.Chikayama, Design of the Kernel Language for the Parallel Inference Machine, The Computer Journal, Vol.33, No.6, 1990, pp.494-500

[5] K.Taki, The parallel software reseach and development tool: Multi-PSI system, Programming of Future Generation Computers, North-Holland, 1988, pp.411-426