2G-1

オブジェクト指向言語による LSIレイアウト設計の協調的処理

阿部 明典、 石塚 満
東京大学

1. はじめに

システムが大規模化するにつれて、その中にある処理を単発的に行なっていたのでは非効率的、時には悪くすらなってきている。システムに於て全ての処理をお互いを関連付けながら協調的に進めて行くことが好ましい場合が増えてきている。LSIのパターンレイアウトの場合もその例外ではなく、全ての部品同士の配置が“或るルール以上ないといけないという拘束”で縛られており、それぞれの部品同士の関係を見ながら協調的にレイアウトしていくと効率であり、好ましいものであると思われる。

以下に示すシステム[1][2]は、オブジェクト指向言語Smalltalkで書かれており、処理は並列でこそないが、オブジェクト指向言語の特徴を生かして全ての処理を関連ある処理と協調的に行なうものである。

2. レイアウトに於ける協調的処理

2.1 オブジェクト指向言語と協調処理

オブジェクト指向言語は全ての処理を、オブジェクトに分け、或るオブジェクトから他のオブジェクトにメッセージを送り、メッセージを受けたオブジェクトがそのメッセージに対する返答をすることで行なう。

従って、処理を細かいオブジェクト（モジュール）に分け、それらのオブジェクトより送られるメッセージを全て関連付けながら処理を進めることにより協調処理を容易に行なうことが出来るし、その概念もモジュールの概念、メッセージのやりとりの概念を持たない他の様式の言語と較べると考え易い。

2.2 本システムに於ける協調的処理

本システムではマルチプロセッサ間のアクセスこそ出来ないが、協調的処理を行い易いオブジェクト指向言語Smalltalkによりシステムの構築を行なっている。従って、シングルプロセッサなので処理、プログラムは直列になってしまうが、処理を細かく分けてモジュール（＝オブジェクト）化しオブジェクトへのメッセージの送付を並列であるかの如くにみなし、それらを相互監視することで協調的処理を行なう。

ここでのLSIパターンレイアウトに於ける協調処理の最終目標は部品を置きながら配線もする、つまり、部品、配線がそれぞれ他の部品の位置などを考慮しながら自分の位置を決めていくという処理をすることである。

しかし、このようなことは、例えばANDを二つ置き、その間を配線するという簡単なレイアウトの場合でもANDを配置するオブジェクト二つ、配線をするオブジェクト、コモンメモリの役も果たすコントローラ等と少なくとも四つのプロセッサが必要となる。本システムでは上記のような事は不可能なので、例えば上記のようなレイアウトを行なう場合は、オブジェクトは四つ（実際にはオブジェクト指向言語ではANDを置くオブジェクトは再利用可能なので三つ）用意しておき、或る程度の事前処理を施してから協調的処理を行なう。実行の実際は、以下の如くとなる。（図1）

①先ず、出力側のANDを置く。この配置は後に置くAND、配線には殆ど影響は与えないと思われるので、ANDを置くべき所の周りの状況－左側にある部品、上部にある部品－をコントローラに調べてもらい、その位置にANDを置く。

②次に、入力側のANDを置く。この配置は配線に可成り影響されるので、本来なら配線と同時に協調的に配置を決めるべきであるが、それはできないので恰も配線が存在するかの如くに配置を決める。実際には、入力側のANDの入力端子数がNなら左側の部品との間にN本の配線が普通、通るはずであるからその分空けながらANDを配置している。この処理は勿論、コントローラが行なう。

③最後に配線を行なう。本来ならANDの配置と同時に行なわれているはずで、本システムの処理に於いても②に示すように仮想的に配線は部品を置く際に引かれている。従って、大抵の場合はその仮想配線路に従って配線すれば良いということになる。

④配線などのため、その処理を行なう前に存在していた部品に悪影響を与えること－その部品の上を配線が通ってしまう等－があった場合は該当の部品を移動して影響が無いようにするが、その際、他の部品もその移動に合わせて移動させる。詳しくは［3］に譲るが、この処理は仮説推論的に行なわれる。つまり、現在の事実が使えなくなったらその事実を使うのを諦め、現状にあう事実を過去には誤っていた項（仮説）に調べに行き、それを新たに事実として採用す

Co-operative Procedure for LSI-Layouting by Object-oriented Language

Akinori Abe , Mitsuru Ishizuka
Univ. of Tokyo

る。

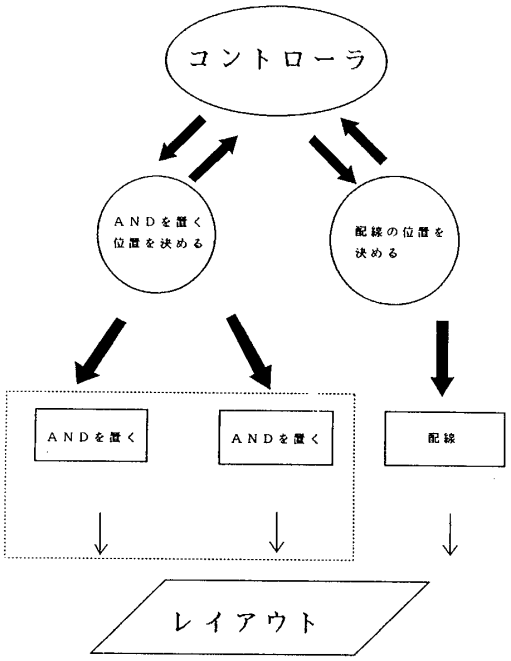

図1　協調処理

3.　まとめ

2節で示したようなような協調的処理をレイアウトの各段階で行なっているが、オブジェクト指向言語を用いているとはいえどもシングルプロセッサ上で行なっているので、完全な協調分散処理というのは難しい。しかし、形だけでも協調的な処理が行えるので、部品の配置決定は周りの状況に対して柔軟に行えるようになっている。

又、本システムでは2節の④で述べたような領域を広げることは容易に行えるが、その反対のことは未だ自動的には行えない。これは配線が通れる領域を増やすのは単純にその配線分だけの領域を追加すれば良いのに対して、領域を減らすのは、例えば、AとBの間の領域を減らすことを考えた場合、先ず、AとBの間に何本配線があったかを記録しておき、コンパクション毎にそれを参照に行かなくてはいけなくなる。この処理は大変オーバーヘッドが大きくなるので、現在のシステムでは行なっていない。

＊＊＊　参考文献　＊＊＊

1］阿部、石塚：“知識型ＬＳＩ－ＣＡＤのためのＳｍａｌｌｔａｌｋによるパターン設計システム”、
人工知能学会研資、
ＳＩＧ－ＫＢＳ－８７０１－５、１９８７

2］阿部、石塚：”ＬＳＩ－ＣＡＤとオブジェクト指向言語”、
Ｃｏｍｐｕｔｅｒ　Ｔｏｄａｙ
９．１９８７、ｐｐ．２３－２８

3］阿部、石塚：“知識型ＬＳＩ－ＣＡＤにおける試行錯誤的レイアウト法の一方法”、
人工知能学会全国大会１９８８（予定）

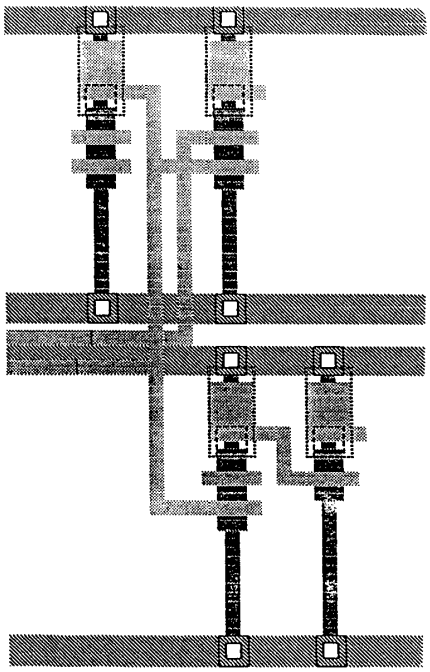

図2－a　協調的レイアウトに於ける④の例
（処理前）

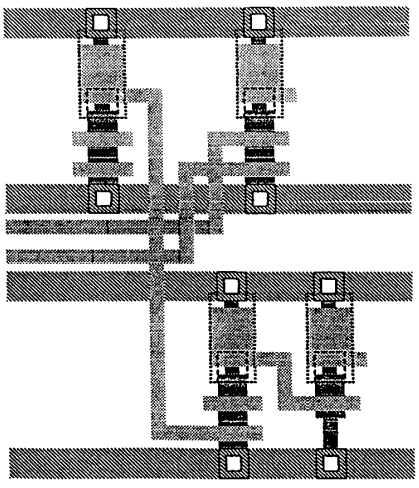

図2－b　協調的レイアウトに於ける④の例
（処理後）