設計向き並列協調問題解決システムの提案

小野 昌之　　横山 孝典　　和田 正覚

(財) 新世代コンピュータ技術開発機構

1　はじめに

LSI 設計の上流工程である方式設計の自動化は、方式設計の設計フローが曖昧で且つ宣言的知識が多い為、従来のアルゴリズム手法では取り扱いが難しく知識処理による対応が期待されてきた。しかし良い成果は得られていない。これは方式設計自体が複雑な問題を持ち、且つ設計に対し非常に高度な知識と創造性の発揮が要求される為で、従来の一つの問題解決方式によるアプローチでは十分とはいえない。その為、方式設計問題を部分問題へと分割化することで問題を小さくし、個々の問題に適した推論要素技術を利用するハイブリッドな問題解決機構が有効な一手法であると考えている。

そこで方式設計問題の解決の為にまず方式設計を分析し、方式設計向きの問題解決機構として、協調問題解決の枠組を提案する。次にこの枠組を用い、試作中の並列協調問題解決システムの概要を紹介し、更にシステムに対する評価を加える。

2　方式設計と設計モデル

2.1　方式設計の設計プロセス

対象としている方式設計は、命令セット、機能仕様及び性能仕様を入力とし、基本的な機能ブロック構成とそのブロック仕様及び RTL レベルの動作仕様の出力を定めるフェーズである。ここでは分析例としてデータパス構造上での命令実行動作の設計を示す。

命令動作設計では、まずハードウェアに依存しない上位レベルでの命令実行動作を設計し、それを基に不完全なデータパス構造が設計される。次に、そのデータパスの詳細化レベルに合わせて命令実行動作を具体化していく。その際動作を具体化した毎にそのレベルにおける評価を行い、それが不適当な解であれば改良又は再設計を行う。この繰り返しにより設計解へと収束していく。

このような方式設計の設計プロセスを分析した結果をまとめ、図 1 に示す。特に改良、再設計については最適解を得る為のものである。また、解生成段階は詳細化、最適化を行うフェーズであり、ある値が決定されその値を基に更に他の値を決定するデータ駆動の問題解決を行っていると考えられる。この部分に関しては設計作業が曖昧な為、厳密な設計フローとして定義できない。

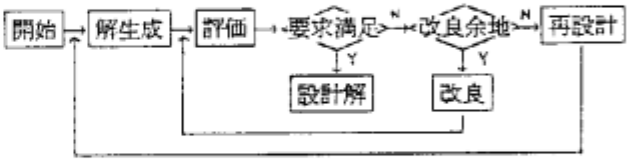

図 1: 方式設計の設計プロセス

2.2　問題分析

設計では対象を様々な角度から眺めた方が良質の解が得られる。方式設計の設計プロセスは再設計を除くと抽象レベルから繰り返し具体化, 収束, 改良を行っていると考えられ、この部分は協調動作そのものである。しかし問題が複雑で大きいので、単一の問題として扱えない、一種類の問題解決方式では十分とはいえない、問題が大き過ぎるので部分問題として分けた場合その部分問題間に依存関係が存在するため他の問題と完全に独立に解けないといった問題が存在する。

そこで、方式設計は性質の異なる部分問題を持つ局面の集合であると見なし、これらの局面に対応する実際の設計内容と問題の性質を表 1 のように分類した。

表 1: 方式設計の分類とその性質

分類	設計内容	問題の性質
動作設計	命令動作定義 命令語の構成	計画問題
構造設計	ブロック抽出 データパス構造決定	組合せ問題 最適化問題
タイミング設計	動作シーケンス定義 パスサイクル定義	制約充足問題 最適化問題

2.3　設計モデル化

設計作業の汎用モデル化は、一部では行われているが定式化困難である。しかし対象を限定すれば、設計自動化の為のモデル化は可能である。

そこで問題分析結果より、方式設計は動作面、構造面、タイミング面の三つの独立した見方で設計が行なわれ、それらが協調して設計解へと収束する協調モデルであると定義した。このモデルを基に並列処理向きの問題解決方式を検討する。

Parallel and Cooperative Problem Solving System for Architectural Design

Masayuki ONO,Takanori YOKOYAMA,Masahiro WADA

ICOT

3 並列協調問題解決システム

これまで既にいつくかの協調問題解決方式が提案されているが、設計問題は部分問題間で同一の設計対象を共有することや、要求仕様や設計方針によって設計項目や設計手順を動的に変更できる柔軟な構成とする必要があることから、協調問題解決にはブラックボードモデルが適している。しかし、従来のブラックボードをそのまま適用したのでは、問題解決機能、および実行効率のふたつの面で問題が生じる。そこで定義した設計モデルを基に、疎結合並列マシン上で並列処理を取り入れた並列協調問題解決方式の一手法を提案する。

図2に試作中の問題解決アーキテクチャを示す。

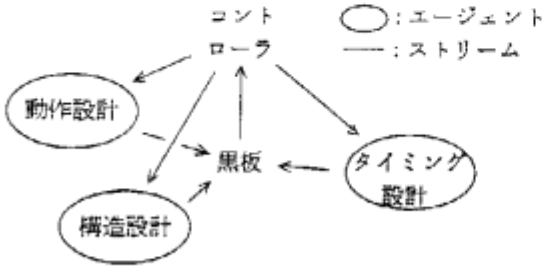

図2: 協調問題解決アーキテクチャ

3.1 ブラックボードモデル

従来のブラックボードが静的な共有メモリにすぎなかったのに対して、本ブラックボードは制約充足、協調動作を積極的に支援するものである[1]。機能として以下のものを持つ。

(a) 設計対象の制約と要求仕様としての制約を満足する状態に保つことにより設計解生成の効率化と枝刈りを行う。

(b) 並列に推論されるエージェントからの提案をコンテクスト管理する。

(c) ブラックボードはコンパイラによりエージェント毎の参照する情報を知っており、その情報が具体化された時点で能動的にその情報をエージェントへ送信する。

3.2 並列協調問題解決アーキテクチャ

ブラックボードは設計対象に関する知識を参照しながら、複数のエージェントにより生成された設計データを全体の整合性を保つように合成する。又、設計対象に関する知識はブラックボード上に記述する。これにより、部分問題間にまたがる設計対象に関する制約条件を利用した、自然な協調処理を実現する方式を持つ。

又、本アーキテクチャでは設計方針が変わる改良はないことを前提に知識をモジュール化している為、設計方針毎に独立した協調推論が可能で、再設計時の後戻りは行わず、設計方針毎に協調推論を並列に実行する。この結果、推論時間の少ないものから順に各設計方針毎に要求仕様を満たす解候補を得られ、その都度解候補との比較を行うことで最適解を導く。

3.3 並列処理

ブラックボードモデルでは各エージェントは並列に実行することができる。しかし、その並列度はエージェントの数に制限される。また、ブラックボードモデルでは全てのエージェントが一つのブラックボードを共有する為、アクセスがボトルネックになり、大きな並列度は期待できない。更に、マルチPSIやPIMのような共有メモリを持たないマシン上で効率良く動かすことは難しい。そこでブラックボードを複数のプロセスで表現し分散させ、ボトルネックになることを回避し、各エージェントのスムーズな並列実行を可能にした。

又、並列論理型言語KL1の特徴を生かすためパイプラインを基本とした処理とする。すなわち、エージェントとブラックボードをストリームで接続し、各エージェントはブラックボードから入力データを入力し、部分的な設計を行い、出力データをブラックボードに出力するというデータフロー的な処理を行うものとする。本問題解決アーキテクチャでは設計方針毎に解候補が多数得られるので、これをパイプライン処理することにより効率化が図れる。

4 本システムの評価

ブラックボードでのプロセス表現は、各プロセスと各エージェントの間における通信量が疎のもの密のものとに分類することができ、疎結合マシン上の負荷分散時にプロセッサマッピングがし易くなるという利点がある。この方式は並列処理に適した方式と言える。

方式設計の部分問題へのモジュール化は知識の追加や保守が容易であるが、この部分問題間の依存関係は大変大きく、厳密な知識の分類は難しい。その為、各モジュールからの提案を単に合成する問題として扱えなくなる恐れがある。この点に関しては推論メカニズムと共に知識の分割法を見直す必要がある。

5 終わりに

以上、本設計システムの大きな枠組である並列協調問題解決方式について述べた。本方式は問題を部分問題にモジュール化し、並列協調問題解決の枠組(知識処理)と並列処理の導入で解決するものである。

各部分問題の問題解決方式として計画問題、組み合わせ最適化問題、制約充足問題等が必要であると分析したが、更に個々の詳細な問題分析が必要である。これらは今後の課題である。

参考文献

[1] 横山孝典ら "並列協調問題解決のための対象モデル表現方式", 情報処理学会第41回全国大会予稿集,(1990)