2V-6

ＡＤＬによる 並列処理プロセッサ設計支援手法

林　喜弘、芳野　泰成、小田原豪太郎
(東京大学工学部)

1．はじめに

近年、ＡＳＩＣの普及はめざましいものがあるが、扱うデータ量や処理速度への要求はますます高まっている。これに対して、並列処理方式は１つの有効な解決法である。しかし、ＡＳＩＣのユーザであるシステム設計者にとっては、その設計は非常に困難である。なぜなら、並列処理プロセッサの特にアーキテクチャの設計には、多くのハードウェアの知識を必要とし、また考慮すべき事項も多いからである。

本稿では、システム設計者でも容易に並列処理プロセッサを設計可能とする設計支援環境について述べる。並列アルゴリズム記述言語ＡＤＬ（Algorithm Description Language）およびアルゴリズム及びアーキテクチャの設計の支援手法を提案する。なお、本稿で扱う並列処理プロセッサは、複数のオートマトンのモデルとして扱っている。また、本支援環境はオートマトン間のアーキテクチャの設計を主たる対象としているため、オートマトン自身は汎用のＣＰＵ（コア）で実現するものとしている。そのため、全ての演算はオートマトン内ではシーケンシャルに実行される。すなわち、本システムの目的は並列プロセスの実現法の設計支援であって、演算レベルの並列（クロック同期パイプライン、シストリックアレイ等）は設計の対象としていない。

2．設計フロー

従来、特定用途向けの並列処理プロセッサの設計では、アルゴリズムの設計とそれを実現するアーキテクチャの設計を同時に考慮しなければならなかった。このため、システム設計者には非常に負担となっていた。

これに対して我々が提案する手法は、アルゴリズムの設計とアーキテクチャの設計を独立させる点が特徴である。図１に設計フローを示す。

（１）並列処理アルゴリズムの設計

まず設計者は、実現したい処理アルゴリズムを、そのアーキテクチャを意識せずに自由に設計する。その際、そのアルゴリズムは逐次処理であろうと並列処理であろうと問題はない。ただ、８クィーン問題のように本質的に並列な問題も存在するので、並列処理記述が可能な記法を提供しておく。本設計環境では、図形的なアルゴリズム記述言語ＡＤＬを提供している。ＡＤＬは、従来我々が提案していたＡＤＬ[1] に変更を加えた、PASCALに似たソフトウェアに近い言語である。処理を論理的なかたまりであるプロシージャ単位で記述する。また、引数、型宣言などもサポートしている。並列／逐次の区別は、'V'マークにより行う。

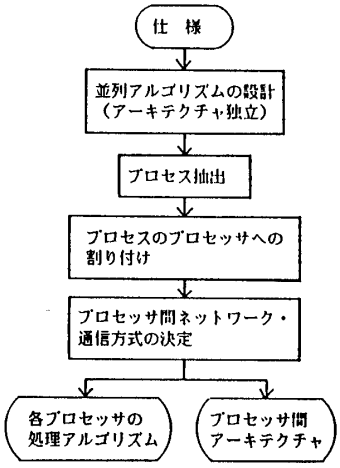


図１　設計フロー

（２）プロセスの抽出

次に、(1)で設計された処理アルゴリズムから並列プロセスを抽出する。この設計を支援するために、次の２つのアプローチを持つ。

a）逐次処理から並列部を抽出

設計者が設計した処理アルゴリズムの中で、そのデータの依存性より並列実行可能な部分を抽出し、設計者に提示する。これには、２つの考え方がある。

i）負荷分散

大量のデータに対して同じ処理を繰り返し、かつ、それぞれのデータが独立である場合、同じ処理を行うプロセスを複数発生させ、データを振り分けて処理を行ったほうが効率がよい場合がある。その際、転送やメモリ競合のオーバーヘッドを考慮した速度向上率で判断することが必要である。システムは、図2a)に示したような部分を抽出し、単一プロセスで実現した場

A Design Support Technique for Parallel Processors using ADL
Yoshihiro HAYASHI, Taisei YOSHINO, Gotaro ODAWARA
University of Tokyo

合と、複数のプロセスで実現した場合の推定処理時間（マシンサイクル）を提示する。また、入出力データを解析し、通信にかかる時間を推定する。ここで、通信方式として、1個1個のデータのシーケンシャル通信や、DMA通信など、いくつかの通信方式を仮定して、また、そのデータの内部での使われ方を参照して、推定を行う。

ii）機能分散

大量のデータに対して同じ処理を繰り返し、かつ、その処理が複数の独立したものである場合には、それぞれの処理を1つのプロセスに割り付ける（図3）。この場合も、i)と同様に、通信などをオーバーヘッドを考慮する必要がある。

b）並列処理の圧縮

設計者が設計した(1)のアルゴリズム中には、処理効率の面から無駄な並列部が有ることもある。現在、並列処理によってその並列プロセスの数分のプロセッサが必要となるため、ハードウェア量の増分に見合っただけの効率が得られない場合は、かえって逐次処理として1つのプロセッサで実現した方がよい。システムは、並列部分を抽出し、その繰り返し回数や1つの処理の時間を推定し、設計者に提示、判断を仰ぐ。

（3）プロセスのプロセッサへの割り付け

(2)で決定されたプロセスを1つのオートマトン（プロセッサ）に割り付ける処理を行う。判断の基準は以下のとうりである。

a）時間的にずれているプロセスは、1つのプロセッサで実現する。

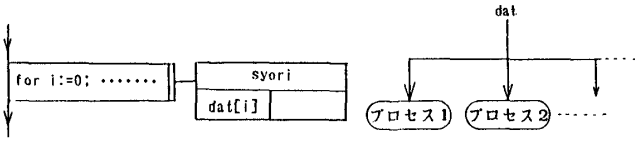
b）同一の処理を行うプロセスは、単独で1つのプロセッサで実現する。

c）時間的に相互関係の規定がない並列プロセスは、CPUの割り込み機能を使う。

システムは、以上の判断基準を示すパラメータを設計者に提示する。

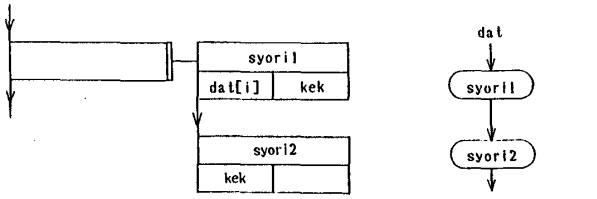
（4）プロセッサ間のネットワーク、通信方式の決定

(3)で決定されたプロセッサ間のネットワークを、そのデータの依存関係より決定する。これと同時に、プロセッサ間の通信方式も決定し、各プロセッサの処理に通信動作を付加する。(3)により、各プロセッサの入出力データが分かっている。そこで、各データの各プロセッサ内での使われ方、データのタイプにより、データの持ち方、通信方法が決定される。例えば、図4a)のように、2つのプロセッサ間で使用される配列データdat1が、プロセッサAで出力され、プロセッサBに入力される場合を考える。プロセッサAで逐次dat1が加工されるならば、同図b)のように同期をとって、AとBが通信するような形



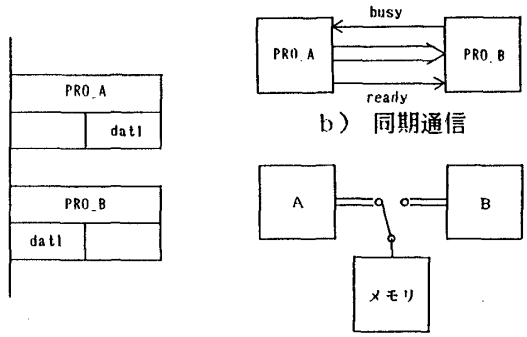
a） ADL記述　　b） 並列プロセス

図2　負荷分散



a）ADL記述　　b） 並列プロセス

図3　機能分散



a）並列プロセスの例　　c） 共有メモリの切り替え

b） 同期通信

図4　アーキテクチャの決定

をとる。一方、dat1に対するAでの処理が完全に終わってBでの処理が始まるならば、同図c)のように共有メモリの切り替え、あるいは、DMA等でデータの転送を行う。システムはこれらの判断材料となるパラメータを提示し設計者の判断の手助けを行う。

以上により、各プロセッサ内で実現すべきソフトウェア処理とプロセッサ間のアーキテクチャを得ることができる。

3．おわりに

本稿では、並列処理プロセッサのアルゴリズム設計、アーキテクチャ設計を支援する手法について述べた。このレベルの設計は、システムの性能を規定する重要な部分であり、また設計者が一番工夫する部分であることから、基本的に判断は設計者に委ねられ、システムはその判断材料を提示する形を取っている。現在、プロトタイプを作成し、実際の設計例に対して適用し、その有効性の確認と問題点の抽出を行っている。

参考文献

[1] 芳野他：「図形的ハードウェア記述言語を用いたシリコンコンパイラの設計環境」，情処研報88-DA-41