大規模プログラムを用いたＦＬＥＮＧプログラムの特性

1D-8

山内　宗、　中屋　雄一郎、　田中　英彦
（東大　工学部）

1．はじめに

現在我々は、高並列推論エンジンＰＩＥの開発を進めている。並列推論マシンのアーキテクチャ、処理方式などを検討するためには、実用規模の応用プログラムの様々な特性を知ることが重要である。本稿では、ストリーム並列言語ＦＬＥＮＧによって記述された、ある程度大きな応用プログラムを高並列推論エンジンＰＩＥのシミュレータ上で走らせて、得られた結果について検討する。

2．ＰＩＥのシミュレーション・モデル

ＰＩＥのシミュレーション・モデルを図１に示す。

一つのインファレンス・ユニット（ＩＵ）は、ユニファイ・プロセッサ（ＵＰ）、メモリ・モジュール（ＭＭ）、共有メモリ（ＳＭ）で構成されている。そして、各ＩＵは共有メモリ網（ＳＭＮ）、負荷分配網（ＤＮ）の二つのネットワークで結合されている。共有メモリへのアクセスとゴール分配の際の通信特性が大きく異っていることが、二種のネットワークに分けた理由である。

ＳＭＮとＤＮは、どちらも４×４のクロスバー・スイッチング・ユニット（ＳＵ）を用いた多段ネットワーク（オメガ網）である。特に、ＤＮは、自動負荷分散機能[1][2]を持った、ネットワークとなっている。

3．シミュレーションに用いたＦＬＥＮＧプログラム

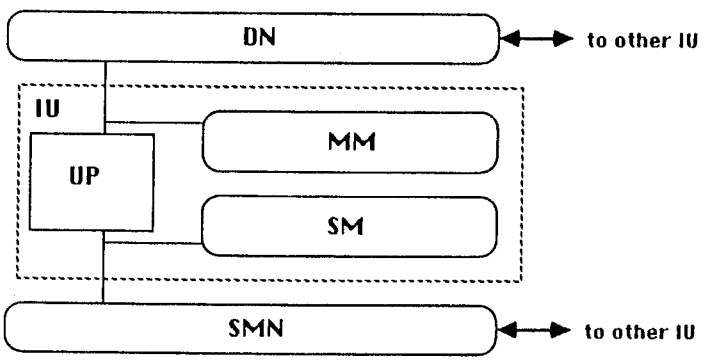
今まで高並列推論エンジンＰＩＥの動作を解析するのに用いられていた論理型言語の応用プログラムとしては、８クイーン、クイック・ソート、素数生成、ナイーブ・リバース等のように比較的小規模のものが多かった。しかし、数十台、数百台規模の並列推論マシンの特性を調べるのに、その様なプログラムで、データをとるだけで十分であるとは言えない。そこで、大規模応用プログラムの必要性が問われるわけである。

では、どの様なプログラムが適しているのであろうか？　ＧＨＣやＦＬＥＮＧのように、動的にプロセスを生成するプログラム言語では、プログラムの行数が長ければ大きなプログラムであるとは言い難い。また、全くシーケンシャルにしか動かないのか、あるいは、次々とプロセスを生成して凄い並列度を有するプログラムなのかによっても、推論マシンの挙動は大きく異なる。従って大きな応用プログラムという場合には、何をもって大きいと判断したのかをはっきりとさせ、どの様なデータを得ることが目的なのかを明確にする必要がある。本稿では、高並列推論エンジンＰＩＥ上でストリーム並列言語ＦＬＥＮＧを実行させた場合の負荷分散の方式を検討することを目的として応用プログラムを選択した。

ストリーム並列言語には、共有変数による値の受渡し、単一化時の同期機構、ガード等によるＯＲ並列性の制限等の特徴がある。そして、この様な言語を並列推論マシンで効率良く実行するためには、次の点に注意をする必要がある。

①共有変数のアクセス競合と負荷分散
②ゴールのサスペンドに伴うオーバーヘッド

今回シミュレーションに用いたプログラムは、論理回路シミュレーションをするものである。なぜ、論理回路シミュレーションを応用プログラムとして選んだかというと、このプログラムは、比較的簡単に並列度を高めることが可能であり、また、プロセス間の通信も頻繁に生じるために、相互結合網の特性を評価するのにも適していると考えられたからである。



IU: Inference Unit
UP: Unify Processor
MM: Memory Module
SM: Shared Memory
DN: Distribution Network
SMN: Shared Memory Network

Fig.1 Inference Unit Model

The characteristics analysis of a large FLENG program on PIE
Tsukasa YAMAUCHI and Hidehiko TANAKA
University of Tokyo

4. 論理回路シミュレーション

今回、論理回路シミュレーションの対象として選んだ回路は、高並列推論エンジンＰＩＥの相互結合網（ＳＭＮ、ＤＮ）に用いられているスイッチング・ユニット（ＳＵ）の中のアービタである。図２にアービタの回路の概略を示す。

論理回路シミュレーションのプログラムには、次のような特徴がある。

①非常に並列度が高い。
②ゴールのアリティの数が多い。
③ＯＲ候補の数が多い。
④サスペンド、アクティベイトが、頻繁に起こる。
⑤ボディ部のゴールの数が多い。
⑥ジェネレータ、コンシューマの関係が複雑である。

①についてであるが、従来のシミュレーション・プログラムでは、数十台の推論ユニットで構成された並列推論マシンのシミュレーションを行っても、そのほとんどが動いていないことが多く、負荷分散の方式を検討するのには、あまり適さなかった。また、ネットワークのブロッキングや、共有メモリのアクセス競合も少なく、ネットワーク構成法の検討にも不向きであった。従って、負荷分散方式や、ネットワークの構成法を検討するのには、論理回路シミュレーションは適していると考えられる。

②については、ＰＩＥでは、ゴール・フレームという単位で処理しているのであるが、アリティが大きいとそのゴール・フレームが長くなり、ユニフィケーションやリダクションにかかる時間が長くなる。

③については、ＯＲ候補が多いとゴール・フレームのマルチ・キャストをしない限り、ヘッド・ユニフィケーションの処理に時間がかかることになる。

④については、素数生成のプログラムの様にどれがジェネレータで、どれがコンシューマかがはっきりわかっているプログラムならば、サスペンドの頻度を減らすのも容易であるが、順序回路のシミュレーションの様に複雑なフィードバックがある場合は、それほど単純ではない。

⑤について。ＰＩＥは、ヘッド・ユニフィケーションが終わった後で定義節に従ってボディ部の新しいゴールを生成し、この処理をリダクションと呼んでいるのであるが、このリダクションという作業はボディ部の複数のゴールに対して逐次的に行われるので、ボディ部のゴールが多いプログラムに対しては、処理速度の低下をもたらす。

⑥についてであるが、これは主に負荷分散に影響を与える。素数生成のプログラムの様にジェネレータとコンシューマが１対１になっていれば、ジェネレータを先にリダクションして少しでも速くジェネレータが走ることができるようにすることによって、コンシューマ・プロセスのサスペンドが生ずる頻度を抑えることも可能になるが、論理回路シミュレーションの場合は、ジェネレータ、コンシューマの関係が、１対多、多対１と多彩であり、しかも、どちらを先に実行したほうが、サスペンドの頻度が減るのかも不明確な場合が多い。

5. おわりに

ストリーム並列言語を指向した並列推論マシンの処理方式等を検討するためには、従来のＯＲ並列向きの例題を高速に実行することばかり考えるよりは、別の例題が必要である。本稿では、特に負荷分散方式や、相互結合網の構成法を検討するのに適していると考えられる応用プログラム「論理回路シミュレーション」について検討してみた。今後は、より多くの例題プログラムが、どの様に動くかを検討して、より詳細なシミュレーションをすることにより、処理方式を決定していく必要がある。

〈参考文献〉

[1] 山内、小池、野田、田中、”高並列推論エンジン実験環境ＰＩＥＥＥ　－自動負荷分散ネットワーク－”、第34回情処全大、4P-3、1987.
[2] 坂井、小池、田中、元岡、”動的負荷分散を行う相互結合網の構成”、情処論、Vol.27、No.5、1986.
[3] 垂井、田中、”並列推論マシンにおけるストリーム並列言語の実行方式の評価”、Proc. of Logic Programming Conference '87、ICOT、1987.
[4] 山内、田中、”ＰＩＥにおけるストリーム並列言語を指向したネットワーク構成法”、第35回情処全大、3C-8、1987.

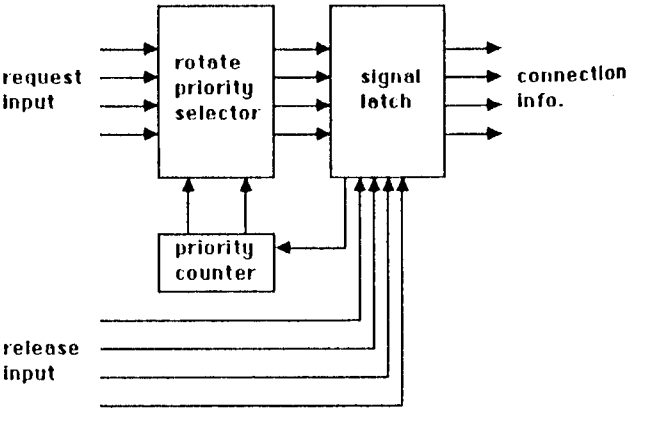


Fig.2 Arbiter Model