3L-2 Datarolプロセッサの インスタンススワップ管理と負荷制御について

薗田浩二　上田哲郎　谷口倫一郎　雨宮真人
九州大学総合理工学研究科

1. はじめに

我々は、データフローアーキテクチャの問題点を解決するアーキテクチャとしてメモリの概念を取り入れたDatarolプロセッサアーキテクチャを提案してきた [1] [2]。これまでの設計では、1つのプロセッサエレメント(PE)はオペランドメモリ(OM)部に1k枚のレジスタファイルを持つようになっているが、実装の面からみるとこれはあまり現実的ではない。また、プログラム実行時に、同時に処理が可能な状態でいる関数活性体(インスタンス)の数はほとんどの場合あまり多くない。そこで、メモリを階層化し、処理の中断しているインスタンスは必要に応じてPE外部のメモリに待避させる方法が有効であると考える。また、データフローでは爆発的な並列性の増加のために資源の枯渇という問題が生じてくる。この問題を回避するためには、インスタンス生成の制御を行い資源を効率的に使用することが必要となる。ここでは、Datarolプロセッサのインスタンスのスワップ管理、及びインスタンス生成の制御の方法について考察する。

2. メモリ階層化

Datarolプロセッサは各インスタンス毎にレジスタファイルと呼ぶ作業領域を割り当てる。これは一定の容量(32～64words)を持つメモリブロックであり、現在までの設計では、1PE当り1k枚のレジスタファイルを持つようになっている。しかし、実行時において各インスタンスは関数呼び出しの命令を実行するとその関数からの値が返って来るまで次の処理に移ることができず休止(Suspend)状態になっていることが多い。従って短い時間内でみれば同時に処理が可能であるインスタンスはそれほど多くなく、レジスタファイルは1PEに少数(32～64枚)用意すれば十分と考える。あるインスタンスの為のレジスタファイルが必要になった場合はSuspendしているインスタンスをPE外部のメモリに退避し、そのレジスタファイルを割り付けてやれば良い。

2.1 Suspend状態の検出

Suspend状態を検出する方法としては、インスタンス内のトークンの増減を監視しておいて、トークンの数が0になった時をSuspend状態とするものが提案されている [3]。しかし、あるインスタンスがSuspend状態になるプログラム上の位置は、コンパイル時に解析可能であるため、それらのノード数をあらかじめ登録しておき、全てが実行された時をSuspend状態とすれば良い。インスタンスがSuspend状態となるプログラム上の位置は、

1. link,rlink命令のノード、
2. rlink命令の子孫のノードへリンクしているノードの中で、rlink命令の子孫でないノード、

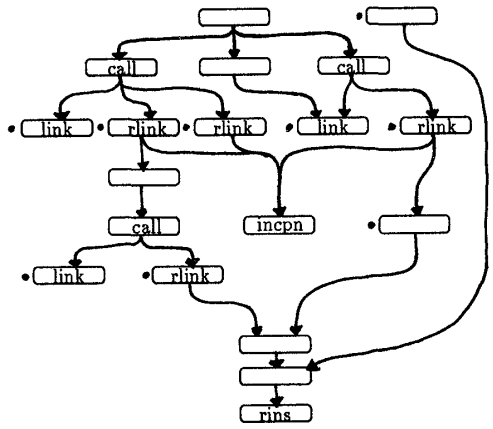


図1: incpn命令を埋め込んだデータロールグラフ

である。したがって、1と2のノード数の和(PN)をインスタンス割付の時(call命令実行時)にあらかじめ登録しておき、それらの命令が実行される度にPNを減少させて、0になった時をSuspendした状態とする。プログラムで関数適用命令が直列におかれていた場合、次の関数適用命令のためにPNの数を増加させておかなければならない。このためrlink命令の後には、PNを増加させる命令(incpn)を置いておく。図1はincpnをプログラム中に埋め込んだものである。図中で・印のあるノードが、PNを減少させるノードであり、incpn命令へのリンクは最初にトークンが流れてきたもののみ有効となる。

2.2 インスタンス制御ユニット

インスタンスの状態の管理、Suspendしたインスタンスの外部メモリへの待避(Swap-Out)または復帰(Swap-In)等の処理はかなり時間がかかると思われるのでこれらが他の処理になるべく影響を及ぼさないようにFU内部においてインスタンス制御ユニット(ICU)は演算部と並列に配置する(図2)。ICUには、インスタンスの情報やPE外部のメモリに退避したときのセグメント番号、Suspend検出用カウンタ(PN)、インスタンスに割り付けられているレジスタファイル番号などを登録しておくInstance-Table(IT)、どのインスタンスにも割り付けられていないレジスタファイル番号をプッシュしておくFree-Register-Stack(FRS)、レジスタファイルとインスタンスとの対応を記述しておくRegister-Table(RT)、Suspendしたインスタンスの名前を入れておくSuspended-Instance-

---

Control of Instance swapping and Load balancing of Datarol Processor

Kouji SONODA, Tetsuro UEDA, Rin-ichiro TANIGUCHI, and Makoto AMAMIYA
Graduate School of Engineering Sciences, Kyushu University

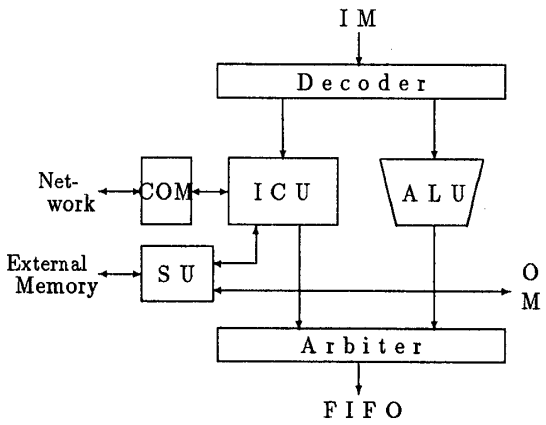
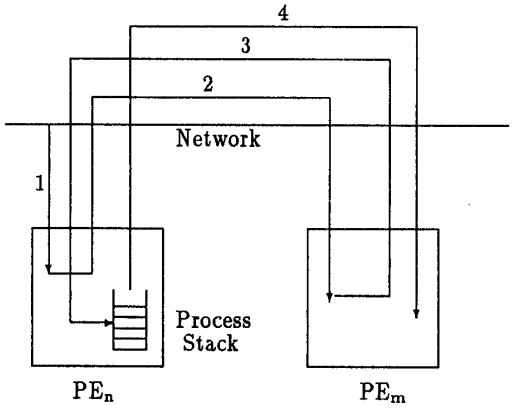


図 2: FU の構成



1 負荷情報、2 PE 要求パケット、
3 プロセス要求パケット、4 プロセス割り付け
図 3: 負荷分散メカニズム

Queue(SIQ)、などがある。また、レジスタファイルと外部メモリ間のスワッピングを管理する Swapping-Unit(SU) も別に設け、そこにはまだ使用されていないメモリバンクの番号をプッシュしておく Memory-Segment-Number-Stack(MSNS) を用意しておく。 関数適用は ICU において以下のようにおこなわれる。call,rcall 命令はインスタンステーブルの使用可能なエントリ（インスタンス名となる）を決定すると同時にそのインスタンスに割り当てるレジスタファイルを決定する。フリーのレジスタファイルがあればそれを割り付けるが、ない場合には SIQ から SUSPENDED 状態インスタンスのインスタンス名を取り出してきて、レジスタファイルデータを外部メモリに Swap-Out した後に、そのレジスタファイルを割り付ける。call,rcall命令はその値としてレジスタファイルナンバーを次命令に渡す。rlink 命令では RFT から自分のインスタンス名を取り出し呼び出すインスタンスへと送る。return 命令が値を返すとき返し先インスタンスが SWAP-OUT 状態の場合は、call 命令の時と同様にしてレジスタファイルを確保し、そこへインスタンスを Swap-In する。これらの処理は RUN 状態のインスタンスの処理と並列して裏で行うことができるため、大きなオーバーヘッドとはならないと考える。

3. プロセス制御

データフローは、プログラムに内在する並列性を自然に引き出して実行するという利点を持っている。しかし、物理的な計算機資源には限りがあるため、再帰型のプログラムでは、資源の枯渇のために実行できなくなる可能性があるという欠点も合わせもつ。したがって何等かのプロセス制御を行う必要があり、再帰型のプログラムの場合 Depth first で実行するように制御するのが良い。ここでは簡単の為に計算機が 1 クラスタで構成されている（全ての PE が同じプログラムコードを保持している）と仮定する。プロセス制御用の機構としては、スタックを用いる。スタックは本質的に Depth first の制御を行う性質を備えている。1 つのインスタンス内に再帰呼び出しが複数ある場合は、1 つの再帰呼び出しを除いてすべてプロセッサ内のプロセススタックに積む。そして、負荷の軽いプロセッサが存在する場合その中で最も負荷の軽いプロセッサに対してプロセスを割り付ける。Datarol プロセッサは PE 間結合ネットワークとしてオメガ網を採用しており、ネットワークスイッチはその出力につながっている PE（スイッチ）の中で最小負荷の PE（スイッチ）の負荷の値を自分の負荷として保持している [2] ため、PE はその入力につながっているスイッチの負荷を調べることで軽負荷の PE がクラスタ内に存在するかどうかを判断することができる (図 3 に PE 間でプロセスを割り付けるときのメカニズムを示す)。従って、全ての PE の負荷が重いときにはネットワークを通じてプロセスが生成されることがない。各 PE は、自 PE 内のいくつかの RUN 状態のインスタンスが一つのインスタンスの生成と自インスタンスの Suspend を繰り返し、RUN 状態のインスタンス数を増やすことがないので資源の枯渇問題を招くことなく再帰呼び出しを速やかに終結させることができる。また、PE は自分の負荷が軽くなった場合にスタックにパケットが積まれているとそこからパケットをポップしてきて自 PE で実行する。

4. おわりに

Datarol プロセッサのメモリ階層化、及びインスタンス生成の制御の方法について考察した。インスタンスの状態を管理し、Suspend したインスタンスを PE 外部のメモリに待避させることで PE 内のレジスタファイル数を必要最小限に抑えることができる。またインスタンス生成の制御をスタックを用いて行うことで資源の枯渇問題を回避することができる。今後は、これらの機構を取り入れることによるオーバーヘッド、プロセススタックの容量等を明らかにしていきこの方法の有効性を検証していく。

参考文献

[1] 雨宮真人、“超並列多重処理のためのプロセッサアーキテクチャ”、「コンピュータアーキテクチャ」シンポジウム、昭和 63 年 5 月

[2] 上田哲郎等、“Datarol プロセッサの設計とその性能評価”、信学技報 CPSY 89-15(1989-08)

[3] 武末勝 “データフロー型計算機の負荷制御方式”、電子通信学会、CAS86-134(1986-11)