データフロー・プロセッサＲＡＰＩＤの並列デバッグ環境

4B－6

坪田浩乃　田村俊之　小守伸史　久間和生
三菱電機（株）　半導体基礎研究所
岩田誠　寺田浩詔
大阪大学　工学部　情報システム工学科

１．はじめに

我々は、最大性能50MFLOPSのデータフロープロセッサRAPID (Ring Architecture Pipeline Intensive Dataflow processor)を開発し、評価用のマルチプロセッサシステムを構築した(1)。本稿では、RAPIDの並列デバッグ環境について述べる。

データフロー方式はプログラムに内在する並列度を利用して、パイプラインを容易に充足することができる。これは反面、実際の命令実行が、ソースプログラム上out-of-order に起こることを示し、デバッグの難しさを招いている。また、マルチプロセッサシステムでは同時並列にプログラムは実行されるので処理すべきデバッグ情報の量も膨大なものとなる。このため、高速で快適なデバッグ環境を提供するためには、ソフトウェアとハードウェアの融合による支援が必要となる。

RAPIDマルチプロセッサシステムでは、RAPIDチップに実装されているデバッグ支援機能を利用することで高速な情報獲得はハードウェアで実現し、獲得した多くの情報の中から必要な情報の選択やわかりやすい表示等はソフトウェアで柔軟に実現するという方針でデバッグ環境を構築中である。

以下では、まず、RAPIDチップに実装されているデバッグ支援機能について説明した後、現状実現されているデバッグ環境について述べる。また、今後、実現を目指している並列処理向けのソースレベルのデバッガについて提案する。

２．RAPIDチップのデバッグ支援機能

RAPIDチップには、３種類のデバッグ支援機能が用意されている。（図１）

①強制分岐機能

RAPIDチップ内部のパケットを外部に取り出すことにより、トレース機能、スパイ機能を実現する。実行状態のパケットを強制的にチップ外に分岐出力し、デバッグ担当のプロセッサや、ホスト計算機に送り、内容の表示や値の変更等の処理を行ってから、元のプロセッサに戻すことで処理を継続させる。この機能を強制分岐機能と呼び、強制分岐の要因によって以下の３種類が準備されている。強制分岐パケットには、行き先PE＃と出力元PE＃がうめこまれて出力される。

・ブレイク命令による強制分岐（このパケットだけを出力）
・強制分岐パケットが分岐部を通過すると以降全てのパケットを強制分岐出力する
・ハードウェアによる強制分岐（すべてのパケットを強制分岐出力）

②スキャンパスによるトレース機能

RAPIDの動作を停止させ、マッチングメモリへの入力部のパケットをスキャンパスレジスタにコピーした後、ビットシリアルに読み出す。１パケット分のトレースが完了すると当該パケットを通過させ、次のパケットをトレースすることができる。内部フラグを含めて１２８ビットのRAPIDのパケットをすべてトレースすることが可能。RAPIDのパイプラインリングは追い越しを許さないため、チップ内部のパケットの順序は、実際の動作時と同一であること

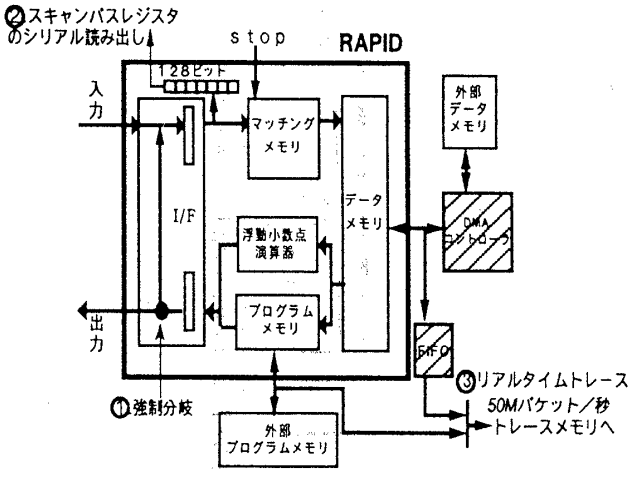

図1．RAPIDチップのデバッグ機能

Debugging Environment for Multiprocessor System of Dataflow processor "RAPID",
Hirono TSUBOTA*, Toshiyuki TAMURA*, Shinji KOMORI*, Kazuo KYUMA*, Makoto IWATA**, Hiroaki TERADA**
*Mitsubishi Electric Corporation Semiconductor Research Laboratry
**Department of Information Systems Engineering, Osaka University

が保証される。

③リアルタイムトレース

外部メモリポート経由で、実行状態の通過パケットを全てトレースできる。最終的な実行順序に起因するバグの発見に有効となる。50MFLOPSで動作するRAPIDをリアルタイムでトレースするため、大量のデータを高速に比較し、格納するためのハードウェアが必要となる。評価システムでは、この機能には対応していない。

3．RAPIDの並列デバッグ環境

現在のRAPIDの評価システムでは、強制分岐機能およびスキャンパスによるトレース機能を使用してアセンブラレベルのデバッグ環境を提供している。

例えば、図２は、スキャンパスによるトレース実行の様子を示している。１秒間に各ＰＥごとに約150個のパケットをトレースすることができる。トレース実行は、トレース回数、ノード番号、カラー番号、データ値の指定でのブレイクポイントの設定、あるいはターミナルからのCtrl-C入力で中断可能で、中断直前の最大1024パケットまでを表示できる。また、強制分岐機能を用いて、指定した変数をホストコンピュータに回収し、値の再代入を実現することができる。

4．並列シンボリッデバッガの実現

RAPIDには、アセンブラに加えＣコンパイラが開発されており、マルチプロセッサにロードするプログラムを生成することができる(2)。RAPID用のＣコンパイラは、デバッグ情報として、各変数更新に対応するノードに対して、ソースファイル名、行番号、変数名の情報が生成されている。該当するノード番号の命令をブレイク命令に変更し、ホスト計算機へ回収することで、プログラムの実行が観測可能となる。ホスト計算機で各ソース行での変数更新を確認することで、どの行の実行が完了したかがわかる。

```
CMND> step 20

                  I De
PE#       PKT#  N st -- Control_Code Opc  Node#(Dec.)  Col VC 1st Oprd 2nd Oprd
-------------------------------------------------------------------------------
PE0(#1)      1 : 0 00 00 0010000011100 FMUL 000000(    0) 000 00 00000000 00000000
PE1(#0)        : NO PACKET !!
PE2(#2)        : NO PACKET !!
PE3(#3)        : NO PACKET !!

PE0(#1)      2 : 0 00 00 0010000011100 FADD 000000(    0) 001 00 00000004 00000000
PE1(#0)      1 : 0 00 00 0010000000000 FMUL 000000(    0) 000 00 00000000 00000000
PE2(#2)        : NO PACKET !!
PE3(#3)        : NO PACKET !!

PE0(#1)      3 : 0 00 00 0010000011100 DMR  000000(    0) 001 00 00000008 00000000
PE1(#0)      2 : 1 00 00 0011000000000 FSUB 000001(    1) 000 00 00000000 00000000
PE2(#2)        : NO PACKET !!
PE3(#3)        : NO PACKET !!

PE0(#1)      4 : 0 00 00 0010000011100 FMUL 000000(    0) 003 00 0000000c 00000000
PE1(#0)      3 : 0 00 00 0010000000000 FSUB 000000(    0) 001 00 00000004 00000000
PE2(#2)        : NO PACKET !!
PE3(#3)        : NO PACKET !!
```

図2．トレース実行の様子

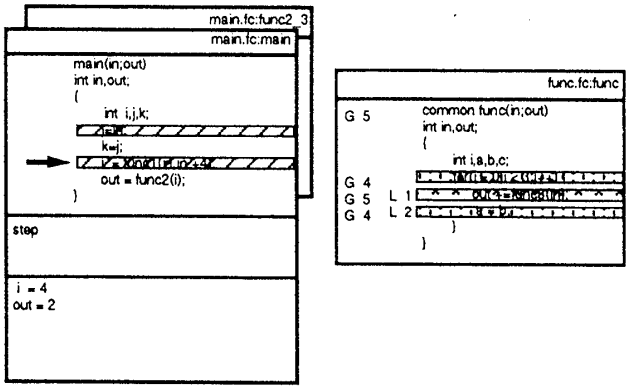

図3．RAPIDシステムにおけるシンボリックデバッガ

前述したように、RAPIDのプログラム実行は、一般にソースファイル行の順番とは異なる。複数のソースファイルに記述された複数の関数が同時に実行される。かつ、同一関数内でも、実行完了行が不連続に現れる。そのため、各ファイルのウィンドウ表示や実行完了行の反転表示、また、ループや共有関数などソース行が複数回実行される場合には、図３に示すように、ループの実行が何回完了したか（Ｌ#)、また、どのカラーの実行が完了したか（Ｇ#)を表示する必要がある。

5．おわりに

今後、RAPIDマルチプロセッサ評価システム上で、様々なアプリケーションの実行を行い。よりよいデバッグ環境を実現していく。

謝辞　当社産業システム研究所の嶋憲司氏、山崎哲男氏をはじめとし、本研究をご指導、ご支援頂いた各位に厚く感謝致します。

参考文献

1)田村，坪田，小守，久間，岩田，寺田．データフロー・プロセッサRAPIDのマルチプロセッサ構成　情報処理学会大48回全国大会論文集，4B-5，(1994)

2)T.Yamasaki, K.Munakata, K.Shima, S.Yoshida, H.Terada, "An implementation of a high-level language for a data-driven processor," IEEE Proc. 25th Asilomar conference on signals, systems & computers(Nov. 1991).