

ジオメトリプロセサ Procyon

4 F-10

—ソフトウェア開発環境—

西本 晴子[†] 岩田 靖[†] 安里 彰[†] 中山 寛[‡] 木村 康則[†][†] (株) 富士通研究所[‡] 富士通株式会社

1 はじめに

近年、プロセッサの開発では、ハードウェアとソフトウェアの開発を並行して行なう場合が多い。そのため、ハードウェアの設計・製作段階からターゲットとなるソフトウェアのテストやデバッグのできる環境を構築する必要がある。

Procyon を開発するにあたっては、設計期間の短縮を第一に考え、シンプルなハードウェア構成にした。そのため、本来ハードウェアで行なっていた演算結果のバイパス制御をソフトウェアによって行なうというように、ソフトウェア開発者への負担が大きくなった。

そこで我々は、デバッグ用のハードウェアシミュレータ、プログラム作成用エディタ、並列プログラム作成用コンバージョンツールなどの、プログラム開発支援ツールの作成を行なった。本論文ではそれらの Procyon ソフトウェア開発環境について述べる。

2 ハードウェアシミュレータ

ハードウェアの設計やソフトウェアの先行開発支援のために、Procyon のコアユニットシミュレータ (Gpsim) の開発を行なった。Gpsim はハードウェアデバッグ用のテストプログラム開発から、コンパイラ開発、アセンブラプログラム開発に至るまで、幅広く利用されている。

図 1 に、Gpsim の基本構成とプログラム開発の流れを示す。Gpsim は、コンパイルまたはアセンブルされた実行形式のオブジェクトコードをシミュレータ内部のメモリ領域に読み込んだ後、コマンド処理部を介してユーザからのインタラクティブな指示を受け取り、1 サイクル単位で、Procyon の各演算スロット内部やリ

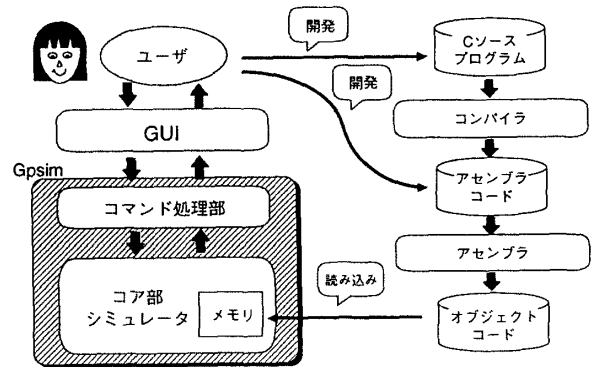


図 1: Gpsim の基本構成

ソースの状態をシミュレートする。

Gpsim は通常のデバッガが持つ、ステップ実行、詳細なブレイクポイント設定等に加えて、

- 各スロットの各パイプラインステージにおける、命令コードやバイパス情報、パイプラインレジスタの状態を 1 サイクルごとに表示する機能
- ホストからの描画入力データパケットを受け取り、ジオメトリ演算をした後、描画プロセサへ出力データパケットを送出するといった、コアユニットと外部とのデータやりとりをシミュレートする機能
- 圧縮された命令コードを元の状態の 4 並列の VLIW 命令列に復元する機能。また圧縮されたコードをディスアセンブルして表示する機能

などの、インタフェース部分を強化した。

また Gpsim のグラフィカルユーザインタフェースを開発した。これはリアルタイムに Procyon 内部のレジスタやメモリの状態を可視化するものである (図 2 参照)。

Gpsim の GUI 化により、パイプラインの動作を視覚的にとらえることができ、デバッグ作業を容易に行うことができる。

Geometry processor Procyon

- Software development environment -

Haruko NISHIMOTO[†], Yasushi IWATA[†], Akira ASATO[†], Hiroshi NAKAYAMA[‡] and Yasunori KIMURA[†]

[†]FUJITSU LABORATORIES LTD.

[‡]FUJITSU LTD.

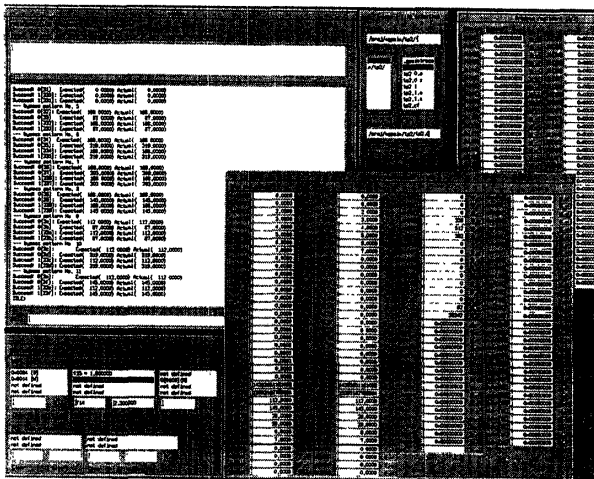


図 2: Gpsim GUI 画面の例

3 ソフトウェア開発環境

3.1 VLIW エディタ

4 並列の VLIW 命令に対して、各スロットごとに独立な編集ができるエディタを作成した。通常のスクリーンエディタを用いると、例えば A スロットにある 1 命令エレメントを削除する際、削除対象になる命令以降に書かれている命令を全て手作業で一段づつ上方に修正していかなければならない。しかしこのエディタを用いれば、スロット毎にバッファが切り分けられているためスロット単位での命令の削除や挿入などが容易に行なうことができ、プログラム編集作業の効率が高くなった。

3.2 コンバージョンツール

並列プログラミングに慣れていないプログラマが、初めから 4 並列をフルに活用した VLIW 用プログラムを書くことは難しい。しかも、Procyon では複雑なバイパス指定をすることもプログラマの役割である。それらの問題を解決するために、独立した 2 つの逐次プログラムを、適切な nop 命令とバイパス情報を付加した 4 並列の VLIW プログラムに変換するツールを作成した (Gpconv)。

A,B スロットにおくことができる命令の書かれたファイルを「B ファイル」と呼び、C,D スロットにおくことのできる命令の書かれたファイルを「D ファイル」と呼ぶ。B ファイルと D ファイルは、浮動小数点レジスタや RAM に関して別々のリソースを使用するため、独立にプログラムを組むことができる。ただし、整数

レジスタは両スロットで共通に使われるリソースなので、B,D ファイル間の整数レジスタの依存関係に対して、ユーザが陽に同期をとる機能を有している。

図 3 に Gpconv の処理の流れを示す。Gpconv は B,D ファイルの文法チェックを行なった後、同期機構を用いて 2 つのファイルのシリアライズを行ない命令列が 1 列に並んだ状態にする。それを元にスケジューリングを行なって、適切な nop 命令やバイパス情報を付加しながら、4 並列の VLIW 命令を形成する。図 4 は、実際に Gpconv でコンバートした例である。

このツールにより、プログラマ自身はバイパス情報を付加することや並列性を気にせずに、プログラムを組むことが可能となった。

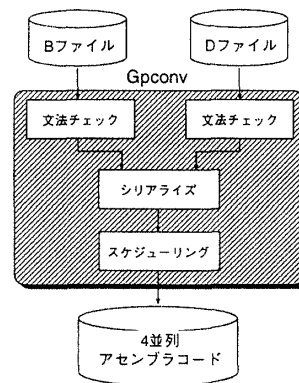


図 3: Gpconv の処理の流れ

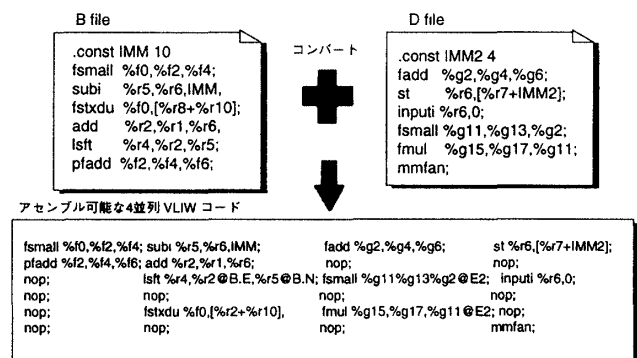


図 4: コンバージョンの例

4 まとめ

Procyon のハードウェアデバッグや、ソフトウェア開発を支援するツールを作成した。これらのツールにより、プログラマのソフトウェア開発の負担を軽減し、ハードウェアとソフトウェアの開発を並行して行なうことができた。