Continuation Based C による Tchnology Mapping のサポート

河野 真治　　佐渡山 陽
科学技術振興事業団さきがけ研究 21(機能と構成)　琉球大学大学院理工学研究科

概要
継続を基本とした C の下位言語である Continuation Based C (CbC) を使って、テクノロジー・マッピングの問題を記述する方法について考察する。ここでは、PS2 に内蔵された DSP の Vector Unit (VU) の記述を行い、その有効性を示す。CbC で記述された VU の記述は、実際に PS2 上で動作させることが出来る。実際の実行時間は、内蔵ハードウェアの 1/1700 程度であった。

Thecnology Mapping using Continuation Based C

Shinji Kono　　Akira Sadoyama
Japan Science and Technology Corporation　University of the Ryukyu

Outline
We are demonstrating Technology Mapping Representation using Continuation based C (CbC), which is a low level language of C. We demonstrate its usefulness by showing a description of PS2's VU which is embedded DSP. The CbC description actually run on PS2 without using real VU. Actual execution speed is 1/1700 of the real hardware.

1 CbCによるテクノロジー・マッピング

技術の進歩は、半導体、LSI、I/O デバイスの様々な分野で急速に進んでいる。が、同じアプリケーションでも、使用される技術に対応してソフトウェアも変更される必要がある。例えば、Intel の CPU が新しくなるたびに、新しい命令が増え、それをサポートするコンパイラを開発する必要がある。特定のアプリケーションに対して、使用される技術との整合性を取ることがテクノロジー・マッピングである。

ここでは、通常のプログラミング言語よりも下位に位置する言語、Continuation Based C (以下 CbC)[1, 2] を用いて、テクノロジー・マッピング自体を記述する方法を提案する。ターゲットとしては、SCEI の PlayStation2 (以下 PS2) を採用する。

CbC とは、状態遷移単位での処理の記述を行う C の下位互換言語であるこの言語には、関数呼び出し、および、for 文や while 文などのループ命令はない。状態遷移を記述するのには、code 単位と、それを相互に移動する goto を用いる。CbC の code は、コンパイラの内部表現である基本ブロック [5] に相当する。CbC の中でスタックを明示することにより、通常の C を CbC にコンパイルすることが可能である [4]。

CbC の code の引数を Interface と呼ぶ。それを実装するデバイスのレジスタや配線などを対応させることにより、CbC 上でのテクノロジー・マッピングが行われる。この対応を含めて、Interface が定義される。

ストリーム演算命令などのマッピングには ストリーム演算命令の動作を記述する code を用意する。これは、通常は、それほど難しくない。特定の Interface を持つ code をストリーム演算命令の列に手動または自動で変換する。変換後の CbC プログラムは、ストリーム演算命令の code の混ざったものとなる。CbC コンパイラは、その code の部分を実際の機械語表現に置き換えれば良い。置き換えなければ、生成されるコードは、ストリーム演算をサポートしない実装上で、そのまま動作するものとなる。

2 CbC による テクノロジー・マッピングの使用法

[アセンブラをなくす] CbC による表現を使う

ことにより、新しい機械語やハードウェアに対して、アセンブラやハードウェアレジスタ設定を含むコンパイラなどを定義する必要がなくなる。

[新しいハードウェアのエミュレーション] CbC による機能記述があるために、実際のハードウェアが存在しなくても、現存するハードウェア上でのエミュレーションが可能となる。正確なクロックなどが知りたい場合は、クロックそのものを CbC 記述に含めれば良い。

[動作検証] エミュレーションにより、変換した CbC プログラムが元のプログラムと同じ動作をするかどうかを検証することが可能となる。Interface に含まれる状態が記号的に有限であれば、理論的にプログラムの同等性を証明することも可能である。

3 PS2 への応用

PS2 では、Linux を走らせることができ、ユーザによるプログラミングが可能になっている。PS2 は、MIPS CPU を中心として、VU と呼ばれる二つの DSP ユニットと描画エンジンである GS、その他、旧 PlayStation との互換性をとるための IO プロセッサなどを持つ極めて複雑なハードウェアとなっている [3]。

一つの VU は、さらに、Upper Execution Unit と Lower Execution Unit の二つに分かれ、それぞれの動作を 1 命令の中で独立に指定できる。Upper 命令では、浮動小数点の加算、減算、乗算、積和計算が行われる。Lower 命令は、より普通の CPU に近く、レジスタ間転送、ジャンプなどの制御命令を含む。例えば、Lower Unit の LQI 命令の実行は、以下のような interface を持つ CbC の code セグメントとして記述される。

```
code lower_LQI(REGISTERS *regs,
  PARAMETERS *params, HAZARD_CHECK *hazard);
```

hazard は、パイプラインハザードチェック用の状態である。

4 実行時間と実行クロック数

シミュレーションにかかる時間を、gettimeofday 関数を用いて計測した。計測結果は、100 回実行した平均である。同様のシミュレーションを、1.6GHz の CPU を持つ汎用 PC で行った。

実行したクロック数	823
平均実行時間	0.004371 秒
1 クロックの実行時間	5.311 μ sec
PlayStation2 との速度倍率	1/1700 倍

汎用 PC

実行したクロック数	823
平均実行時間	0.000337 秒
1 クロックの実行時間	0.409 μ sec
PlayStation2 との速度倍率	1/130 倍

5 まとめ

CbC は、状態遷移記述に適しており、その用途は幅広い。本研究では、CbC による VU アセンブラプログラムにより、テクノロジー・マッピングを記述する可能性を示した。

参考文献

[1] 河野真治:継続を持つ C の下位言語によるシステム記述, 日本ソフトウェア科学会第 17 回大会論文集, September, 2000(in Japanese)

[2] 河野真治, 島袋仁:C with Continuation と、その PlayStation への応用, 情報処理学会システムソフトウェアとオペレーティング・システム研究会 (OS), May, 2000 (in Japanese)

[3] PlaySation2 Linux Kit
付属マニュアル

[4] 言語の Continuation based C への変換, 河野真治 (琉球大/科学技術振興事業団), 揚 挺 (琉球大), SWoPP 2001, Okinawa, July , 2001

[5] J.D. Ullman, Compiler

[6] Norman Ramsey and Simon Peyton Jones. A single intermediate language that supports multiple implementations of exceptions. In *ACM SIGPLAN 2000 Conference on Programming Language Design and Implementation*, June 2000.