9M−1

専用プロセッサ設計支援システム(SYARDS)における大規模システム処理系の設計

雨坪 孝尚　上田 穣　吉田 裕　白井 克彦
早稲田大学

1　はじめに

近年、研究が活発になっているハイレベルシンセシスは、高レベルの設計工程において、高級言語による動作記述から、回路の動作や構造等の正しい定義付けを自動的に行い、設計者の意図と下位工程の橋渡しを容易にする技術である[1]。現在、このハイレベルシンセシスのように、高位仕様記述を入力として、プロセッサの設計を支援する、システムの構築に取り組んでいる[2]。このシステムは、高級言語によるアルゴリズム記述をもとに、高速で効率のよい専用プロセッサの設計支援を行うことを目的としている。今回は、デジタル信号処理の実規模レベルのアルゴリズムを対象とし、シミュレーションによって、プロセッサ内におけるデータや演算器のビット構成を、決定する手法について研究した。

2　システムの概要

システムは、アルゴリズム記述が与えられると、その記述にコンパイラ的な処理を施して、シミュレーションと解析のための中間情報に変換する。シミュレータでは、アルゴリズムの確認と、設計の具体化に有効な情報を得るための解析が行われる。

2.1　アルゴリズム記述言語

システムでは、変数の宣言と処理アルゴリズムの記述を中心とする、Pascalに似た仕様記述言語を用いている。例として、J. Machinの公式による円周率の計算を記述したものを図1に示す。

```
program pi(oport);
 output oport: float;
 var    k: integer;
        p,t,last: float;
begin                          k:=1;
 k:=1;                         t:=castfloat(4)
 p:=0;                           /castfloat(239);
 t:=castfloat(16)              last:=1;
   /castfloat(5);              while p<>last do
 last:=1;                       begin
 while p<>last do                last:=p;
  begin                          p:=p-t/castfloat(k);
   last:=p;                      t:=t/(-castfloat(239)
   p:=p+t/castfloat(k);                *castfloat(239));
   t:=t/(-castfloat(5)           k:=k+2
         *castfloat(5));        end;
   k:=k+2                       oport:=p
  end;                         end.
```

図1: アルゴリズム記述

変数の型宣言は、あくまでも変数の属するグループを指定するだけであって、それぞれのビット構成の指定は、シミュレーション時に行う。

ビット構成の異なる2変数の演算を扱うようなハードウェアは、存在しないものとしたので、データ型の異なる2変数の演算は許していない。このため、型の変換に用いるキャスト命令(castfloatなど)を導入している。これは型の変換だけでなく、イミディエイトデータのデータ型を明確にすることにも使われる。このキャスト命令によって、与えられるビット構成に忠実なシミュレーションを行えるようにすることができる。

2.2　中間情報の生成

高級言語で表されたアルゴリズム記述を、単純な3番地文に展開した後に、次の2つの処理を行う。

第一に、3番地文の形成に伴って生じる、一時変数のデータ型の指定をする。定数どうしの演算結果の代入先など、型の特定ができない一時変数には、暫定的にimmediate型が与えられるが、これは後の、定数の計算の畳み込みによって消滅する。

第二に、プログラム内の解析をし、ループ不変式の移動、定数の計算の畳み込み、共通部分式の削除、不要な代入文の削除、算術恒等式の利用といった最適化処理を行う。

以上の処理を経て得られた中間情報には、変数の宣言と3番地文の列が含まれている。

2.3　シミュレータ

シミュレータは、中間情報の3番地文の列に基づき、サンプルデータを用いて演算を行う。この際の演算は、アルゴリズム記述とは別に利用者から与えられた、各データ型に対するビット構成の指定に従って行われる。

2.3.1　指数型数値の形式

本シミュレータにおける、指数型数値の内部表現形式について述べる。仮数部は、最上位ビットとその1つ下位のビットの間に小数点があるとし、指数部の底は2であるとして、いずれも2の補数で表すことにする。シミュレータ内部では、仮数部と指数部をいずれも整数と見立てて、この組をリストで表したものを1つの指数型数値として扱っている。例えば、数値1を仮数部8ビット、指数部4ビットで表すとした場合の、2進数表記とシミュレータ内部で扱っている形式は次の通りである。

```
0 1 0 0 0 0 0 0 0 0 0 0 0 1
          ↓
        (64 1)
```

この形式を用いて、各演算を整数で扱えば、ハードウェア上で演算する場合と同様の精度が得られる。本システムの処理系であるCommon Lispにおいては、整数の大きさについては制限がないので、ビット幅の大きな数値でも取り扱うことができる。演算が複雑な三角関数や平方根などは、通常の数値の形式に一旦変換してから、Common Lispの関数による演算を行い、その結果を再びリストの形式に変換するという方法を、やむをえず採用することにした。固定小数点型の小数点は最下位ビットの下にあ

A design method for large scale special purpose processors in SYARDS
Takanao AMATSUBO, Yutaka UEDA,
Yutaka YOSHIDA, and Katsuhiko SHIRAI
WASEDA University

るとして、符号つき整数のみを表すものとしており、内部では通常の数値の形式で扱っている。

2.3.2 オーバーフロー等の検出

このシミュレータは、オーバーフローがあった時には止めるかどうか、そしてアンダーフローがあった時には0にするかどうかを、オプションで設定することができる。通常は、オーバーフローやアンダーフローを適用せずにシミュレーションを行う。この場合は、浮動小数点型の仮数部のビット幅だけを指定すればよい。そして、シミュレーション終了時に、固定小数点の型に対しては、その型の変数の値の変動幅が、浮動小数点の型に対しては、その型の変数の指数部分の値の変動幅が表示される。同時に、それらに耐えられるビット幅も表示される。利用者はこれを参考にして、異なったビット構成の設定によるシミュレーションをするか、ビット幅を確定するかを選択する。また、使われている各命令の実行頻度を、実行ステップを単位として表示する。

3 適用例

音声高能率符号化アルゴリズムである、LD-CELPを取り上げた。エンコーダは、8kHzでサンプリングされた16bitの音声データの、4サンプル毎に10bitのコードを生成することで、16kbit/sの符号化をする。デコーダは、コードブックをもとにして音声波形を再生する。

異なるビット幅によって、シミュレーションを実行すると、それぞれ符号化されたコードが出力されるが、それらの評価は難しい。そこで、エンコーダの内部に含まれているデコーダ部分を利用し、そこで復号化された波形を評価することで、プロセッサのビット構成の決定を試みた。

エンコーダのアルゴリズムはかなり大きなものなので、中間情報への変換を容易にするために、1つのメインモジュールと34のサブモジュールに分割して記述した。この際、固定小数点のデータ型としてinteger型、浮動小数点のデータ型としてfloat型を用いた。その記述は、全モジュールの合計が968行で、中間情報は2245行の3番地文となった。

1680サンプル（0.21秒に相当）の音声データに対し、仮数部が12, 14, 18, 22, 26, 53ビットのそれぞれの場合についてシミュレーションを行なった。システムの設定は、オーバーフローがあっても止まらず、またアンダーフローがあっても0にはしないということにした。

出力波形を評価する尺度として、C言語による計算結果と原音声をそれぞれ対象とする、ケプストラム距離を算出した。その結果を図2に示す。

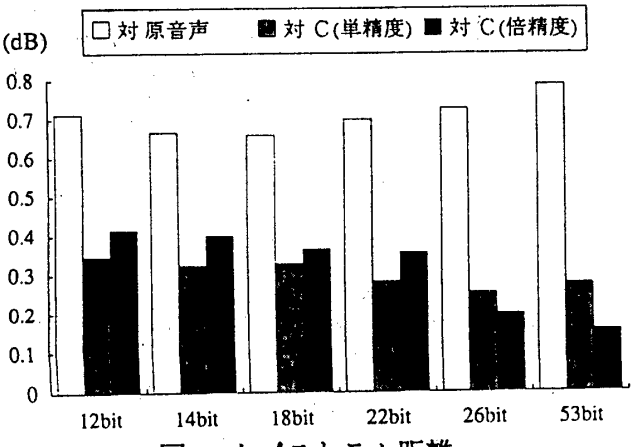

図 2: ケプストラム距離

原音声との距離を比較すると、ビット幅を大きくとればかえってかけ離れる傾向があることがわかる。対C言語では、22ビットと26ビットの間で、単精度と倍精度に対する距離の大小関係が逆転している。これらのことから、仮数部のビット幅は22ビットから26ビットの間に設定するのが適当であると考えることもできる。

ビット幅に関する解析結果を表1に示す。このシミュレーションに関する限りでは、float型の仮数部のビット幅がいかなる場合でも、その指数部には7ビット、integer型には15ビットを確保すればよいことがわかる。

命令の種類別の頻度解析のうち、仮数部が22ビットのときの結果を表2に示す。この表からは、アルゴリズムの実行に要するステップ数がわかる。今回の0.21秒のデータの処理に、これだけのステップがかかるということから、実時間で実現させるためには、約61MHzのクロック速度を要することになる。

また、演算の種類別の頻度解析のうち、同様に22ビットのときの結果を表3に示す。この表からは、ハードウェアとして用意すべき演算器の種類が、データ型ごとにわかる。

表 1: ビット幅解析

float			integer	
仮数部	指数部			
bit 幅	変動幅	bit	変動幅	bit
12	38 -23	7	12064 -8440	15
14	36 -26	7	11360 -9496	15
18	32 -31	7	11352 -8824	15
22	34 -32	7	10272 -8456	15
26	34 -26	7	10928 -8416	15
53	38 -25	7	10920 -8408	15

表 2: 頻度解析（命令種別）

命令型	ステップ
A = B	321775
A = B op C	5100371
A = op B	18683
A = B[C]	3219588
A[B] = C	971711
GOTO X	1528940
IF GOTO X	1715577
CALL A	11011
合計	12887656

表 3: 頻度解析（演算種別）

演算子	ステップ
boolean 型	90570
AND	(90570)
integer 型	4039535
+	(1318538)
-	(1128219)
*	(56112)
DIV	(7125)
=	(1466736)
>	(420)
>=	(54096)
<=	(1680)
CASTFLOAT	(6609)
float 型	2704526
+	(914241)
-	(171696)
-(単項)	(8378)
*	(1316602)
/	(6362)
INT	(1680)
LOG	(336)
POWER	(336)
>	(116914)
<	(74262)
<=	(91706)
<>	(333)
CASTINTEGER	(1680)
合計	6834631

4 おわりに

プロセッサ内で使用するデータや演算器のビット構成を、シミュレーションによって決定できるシステムを構築した。この際、ハードウェアとして合成されたものと、同様の精度の演算でシミュレーションできるように、ビット構成を考慮して演算する方式を導入した。このシステムを用いると、ハードウェア構成の具体化に、有効な情報が得られることがわかった。今後は、アルゴリズム内で特に高い精度が要求される所と、そうでない所を見きわめ、各変数に対するデータ型の割当てを、より的確なものにする技術が望まれる。

参考文献

[1] 高橋隆一, 吉村猛: 「ハイレベルシンセシスの動向」, 電子情報通信学会論文誌A, Vol. J74-A, No. 2, pp. 143-151 (1991).

[2] 池永剛, 白井克彦: 「高級言語により記述されたアルゴリズムを実現する専用プロセッサ設計支援システム」, 情報処理学会論文誌, Vol. 32, No. 11, pp. 1445-1456 (1991).