3R-4 回路自動設計のための アルゴリズム記述とフロー解析

上野 聡 竹沢寿幸 白井克彦
(早稲田大学理工学部)

1. はじめに

高級言語による仕様記述をもとに知識工学的手法を用いて回路の機能設計を支援するためのシステム[1]を開発している。レジスタ・トランスファレベルの状態遷移記述から回路を合成する方法は、信号処理を対象とするユーザにとっては馴染みの薄いものである。こうしたユーザの立場からは、通常用いている高級言語によるアルゴリズム記述をもとに回路合成が行われることが望まれる。一方、機能設計において記憶要素や演算器、内部接続ユニットを割り付けるための入力としては3番地文の列が適している。しかし、高級言語によるアルゴリズム記述をそのまま3番地文に変換すると無駄が多いため、フロー解析を行い最適化を行う必要がある。

本稿ではPASCAL風の仕様記述言語の特徴と、その記述から3番地文を生成する際のフロー解析と最適化の方法について述べる。

2. 設計仕様記述言語

仕様記述の対象は、信号処理アルゴリズムに限定する。なぜなら、信号処理アルゴリズムでは入出力の仕様を明確に記述することが出来、処理内容も主に四則演算を用いた繰り返しが比較的多いためである。

言語仕様は標準PASCALのサブセットにいくつかの変更および拡張を行ったものである。

データ構造の基本型としてbit型を導入する。本来高級言語の目的の一つは抽象化したデータ型を用いることにより、ハードウェア構成に依らずにデータ処理を行う事である。しかし、信号処理アルゴリズムを記述する場合にはデータのbit幅を常に意識する必要があり、これを基本型とすればデータを正確に記述することができる。bit型には符号付き(signed)と符号無し(unsigned)があり、符号付きの場合はMSBが符号ビットになる。実数型についてはハードウェアによる実現方法が様々であるうえに、信号処理においては演算精度が重要であるため、必要な精度のみを記述する。また、構造を持つデータ型としては一次元の配列型が使用可能とする。

制御構造については、標準PASCALと同様にし、if文、case文、while文、repeat文、およびfor文が使用可能である。

実際の仕様記述は次のように行う。プログラムの頭書きにはプログラム名と入出力変数名を記述する。続いて、入出力変数、ラベル、定数、型、変数を記述する。これらの順序は任意である。この後にbegin-endで囲まれた、一つの処理アルゴリズムが記述できる。一つのアルゴリズムを一つのＬＳＩとして設計するため、関数や手続きは記述できない。アルゴリズムは四則演算、論理演算、およびハードウェアで実現可能ないくつかの関数によって記述されなくてはならない。

```
program parcor-filter (s, rk, e);
input   s  : array [1..n] of signed bit (12);
        rk : array [1..m+1] of signed bit (8);
output  e  : array [1..n] of signed bit (12);

const   m = 12;
        n = 128;
type    integer = singed bit (16);
var     ft, gto : array [1..m+1] of integer;
        i, j, m1 : integer;

begin
  for i := 1 to 16 do begin
      gto[i] := 0;
      ft[i]  := 0
  end;
  e[1] := s[1];
  m1 := m + 1;
  for j := 2 to n do begin
      ft[1]  := s[j];
      gto[1] := s[j-1];
      for i := 2 to m1 do begin
          ft[i]  := ft[i-1] - rk[i-1] * gto[i-1];
          gto[i] := gto[i-1] - rk[i-1] * ft[i-1];
      end;
      e[j] := ft[m1]
  end
end.
```

図1. PARCOR格子型フィルタの仕様

Specification Description and Flow Analysis
for an Architecture Design System
Satoshi UENO, Toshiyuki TAKEZAWA, Katsuhiko SHIRAI
Waseda University

3．フロー解析とコード最適化

仕様記述を基にフロー解析とコード最適化を行い、機能設計を行うための３番地文の列を作成する。コード最適化の目的は、効率のよいコードにより速い実行速度を得ることと、ハードウェア構成要素の数を減らしコストを低くすることである。

最適化は、次の手順で行う。

まず、与えられたソースプログラムをそのまま３番地文の列に変換する。３番地文の列は、さらにいくつかの基本ブロックに分割される。基本ブロックは最適化を行うための基本単位であり、これに入るには必ず先頭の文を実行しなければならず、かつブロックの途中から制御が分岐することがないように分割する。

次に変数のライフタイム解析を行い、それらの情報をもとにして、コードの移動や、帰納変数の消去などの主なループ最適化を行う。コードの移動とは、ループ内で不変な式を検出し、ループのヘッダ前ブロックに移動することにより、処理を高速化することである。帰納変数とはループ内で等差数列となっているか、あるいはその線形関数となっている変数であり、これらのうち不要なものを消去することにより、処理速度が向上する。

次に、大域的共通部分式の消去や、定数のたたみ込みなどの大域的最適化を行い、さらにに局所的共通部分式の消去や、不要な複写の消去などの、基本ブロックの最適化を行う。

最後に、各基本ブロック内で演算の相互依存性から並列性を抽出し、また基本ブロック間の制御フローグラフも同時に作成する。通常のコンパイラとは異なり、回路を設計するためには特に並列性の情報は重要である。

以上により作成された中間表現は３番地文の２次元リストになる。最適化の手法は通常のコンパイラと同様であるが、基本ブロック間の制御フローや並列性の抽出、変数のライフタイム解析などは続く設計過程で利用するための情報である。それらの情報をいかに設計に生かすかが問題である。

4．例

図１にPARCOR格子型フィルタの仕様を記述した例を示す。これにフロー解析と最適化を行った結果の３番地文の列を図２に示す。各ブロック間の制御フローを図３に示す。これらをもとに実際の回路設計を行った例は文献[1]で述べる。

5．むすび

回路設計のための仕様記述言語およびフロー解析について報告した。ここで述べた言語仕様は試験的なものであり、制約条件の記述等のユーザの要求が記述できる必要がある。またフロー解析の結果を演算速度や演算精度等の性能評価に利用したいと考えている。

参考文献

[1]竹沢，上野，白井：“アーキテクチャ設計支援システムにおける並列処理とパイプライン処理”，情報処理学会第33回全国大会，3R-3，(1986-10)．

```
B1: S1 := 2,            T2 := addr(gto)-2, T4 := addr(ft)-2

B2: if S1>32 goto B4

B3: T2[S1] := 0,        T4[S1] := 0
    S1 := S1+2
    goto B2

B4: T6 := addr(s)-2,    T9 := addr(e)-2,    S2 := 4
    T7 := T6[2]
    T9[2] := T7

B5: if S2>256 end

B6: T12 := T6[2],       T16 := S2-2,        S3 := 4
    T4[S2] := T12,      T18 := T6[T16]
    T23 := addr(rk)-2, T2[2] := T18

B7: if S3>26 goto B9

B8: T22 := S3 - 2
    T24 := T23[T22],    T28 := T2[T22],     T40 := T23[T22],    T44 := T4[T22]
    T29 := T24*T28,     T33 := T4[T22],     T45 := T40*T44,     T49 := T2[T22]
    T34 := T33-T29,                         T50 := T49-T45
    T4[S3] := T34,                          T2[S3] := T50
    S3 := S3+2
    goto B7

B9: T55 := T4[26]
    T9[S2] := T55
    S2 := S2+2
    goto B5
```

図２．フロー解析結果

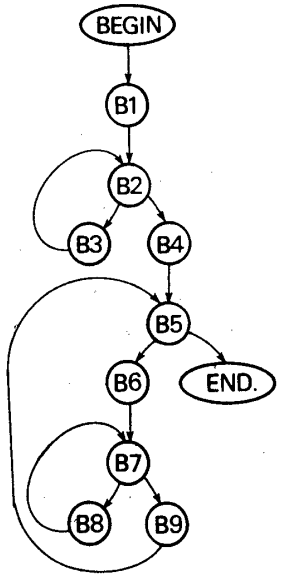

図３．ブロック間の制御フロー