1C－3

専用プロセッサ設計システム(SYARDS)と論理合成システムとの接続

樋渡仁　上田穣　吉田裕　白井克彦

早稲田大学 理工学部

1 はじめに

近年、VHDL等の専用プロセッサ設計環境の充実や汎用プロセッサの高速化などにより、ソフトウェアで対処できる可能性も広がりつつあり、両者の利点を生かしたプロセッサ設計が探求されている。我々はこのような背景から、幅広い分野のユーザを対象とし、アルゴリズムによる動作記述を入力とした専用プロセッサ設計システム(SYARDS)の提案と研究を行なってきた[1],[2]。しかしながら、以前より現実的なシステムをめざす意味からも設計された専用プロセッサの回路規模を見積もりたいという要求が指摘されていた。

そこで本稿では、専用プロセッサ設計から得られたプロセッサ情報をテンプレート法と呼ばれる方法によりデータパス割付や制御割付を行ない、最終的に論理合成システム(PARTHENON)を用いてゲート数や実行時間を見積もる。さらに、代表的なアルゴリズムに関してSYARDSを適用し、割付アルゴリズムとしてのテンプレート法を評価する。

2 テンプレート法

高位合成においては、スケジューリングされたCDFG(control/data flow graph)を構造記述に変換する過程を割付と呼ぶが、最終的に設計された専用プロセッサの性能を決定するのは、この割付によってどのようなデータパスと制御ユニットを生成するかに大きく依存する。そこで、本稿では図1に示したテンプレート法と呼ばれる割付法を用いた。この方法では、対象とするプロセッサの構造をある程度限定し、命令セットや記憶要素などのアルゴリズム実行に必要最小限の資源を求めて、それらの要素を統合することによって回路合成する。

テンプレート法におけるデータパス割付は、次のように実行される。まず、CDFGレベルで必要なマイクロ操作は全て実行可能なデータパスを雛型として用意し、解析系から得られた必要最小限の命令セットをもとに割付段階において実行に不必要な演算器は合成しない。例えば、乗算を使用しないアルゴリズムのデータパス割付を行なう場合には、実際の実行に不要な乗算器は合成しない。

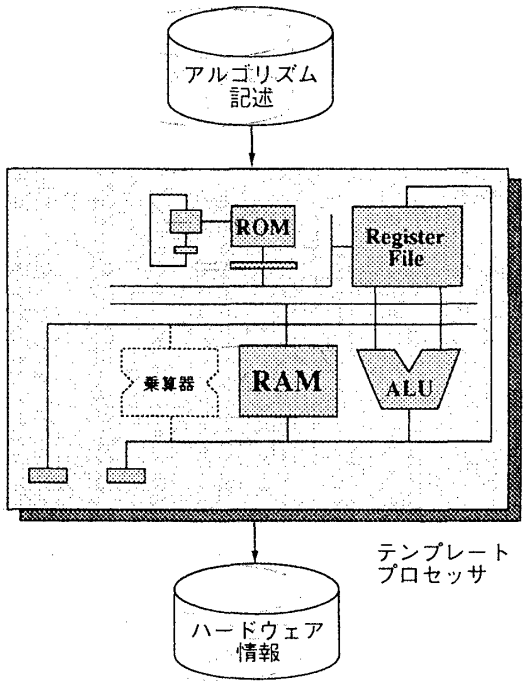

図1: テンプレート法の概念

一方、記憶要素はそれぞれ対応する解析の結果を用いて、最小限のレジスタやRAMの容量をもとめ、実行に必要な記憶要素の割付を行なう。また、レジスタをデータパス内の任意の位置に配置するような割付法も存在するが、テンプレート法ではレジスタファイルを用意して配置する場所も固定で対処している。

また、テンプレート法では、解析系から得られた命令セットを用いてマイクロコードを生成し、このコードによってデータパスの制御を行なう。この際、命令セットにもある種のテンプレートを用意して、あらかじめ解析結果として抽出した必要最小限のレジスタやRAMの容量も考慮して、マイクロコードとそれを実行する制御ユニットを合成する。

Combination of Design System for Special Purpose,Processors (SYARDS) with Logic Synthesis Tool

Jin HIWATASHI, Yutaka UEDA, Yutaka YOSHIDA and Katsuhiko SHIRAI

Department of Electrical Engineering, Waseda University

3 適用例

SYARDS を用いた専用プロセッサ設計と評価の目的で、代表的な信号処理アルゴリズムとソートアルゴリズムを取り上げる。

○ バブルソート (BS)
○ クイックソート (QS)
○ PARCOR 格子型フィルタ (PAR)
○ 自己相関関数 (COR)

この 4 種のアルゴリズムに対して SYARDS を適用し、テンプレート法を用いて割付を行ない、最終的に命令実行時間やゲート数を見積もった。4 種のアルゴリズムに関して、実行時間とゲート数、静的ステップ数を図示したものを図 2 に示す。

4 評価

従来の SYARDS のみを用いた場合には、設計された専用プロセッサの実行ステップ数しか評価できなかったが、本稿では割付法としてテンプレート法を採用し、割付結果をハードウェア記述言語 (SFL) に変換して、論理合成システム (PARTHENON) を利用して論理合成を行ない、得られたネットリストから回路規模や実行時間等の情報抽出を可能とした (図 3)。

このように見積もった情報は、様々な設計条件に左右されやすく正確とはいえないが、設計の初期段階からより詳細な情報を得ることができれば、それ以降の設計過程に反映させることが可能である。特に、SYARDS が主な設計対象としている信号処理アルゴリズムを実現する場合には、リアルタイム性を満たしているかの情報を SYARDS にフィードバックできる点は極めて重要である。

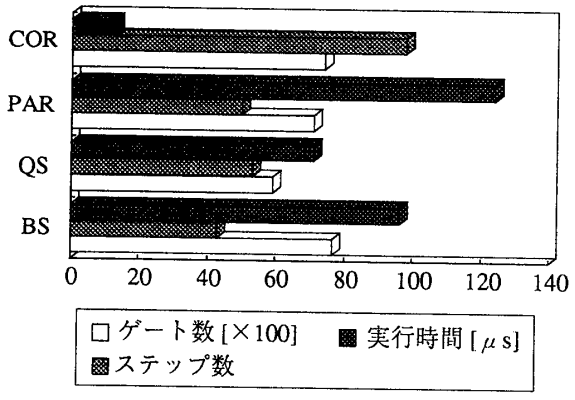

図 2: 適用例の回路規模・実行時間

```
module parcor {
  stage exu {
    state_name ex0 ;
    first_state ex0 ;
    state ex0 par {
      alt {
        edec<0> | edec<1> : par {
          d1 = rfiles.rd1(mona.do(ir2).r1).a_out ;
          d2 = 0b0000000000000001 ;
          alt {
            edec<0> : a1 := alu.sub(d1,d2).out ;
            edec<1> : a1 := alu.add(d1,d2).out ;
          }
        }
        edec<2> : par {
          alt {
            move.do(ir2).i1 : d1 = move.im1 ;
            else : d1 = rfiles.rd1(move.r1).a_out ;
          }
```

図 3: ハードウェア記述言語 (SFL) による記述

次に、本稿で用いた割付法であるテンプレート法の評価を行なう。一般的な高位合成システムで行なわれている割付法では、資源に制約条件などを与えないで合成を行なうと、対象とするシステムの複雑化にともない回路規模が増加する。しかし、図 2 に見られるようにテンプレート法では、複雑度 (CDFG の静的ステップ数) が増加しても、ゲート数等の回路規模がそれほど増加しないことが示された。すなわち、この評価により大規模な回路の割付にはテンプレート法が有効であることが示された。

5 おわりに

SYARDS と既存の論理合成ツールの接続を通して、プロセッサに関するより詳細な情報が得られ、より現実的な設計が行なえることを示した。また、割付法としてはテンプレート法を用い、同時にテンプレート法が大規模な回路の割付に有効であることを示した。将来的には、テンプレート法をより自由な配置や資源量を可能にした割付法へと拡張を行なう予定である。

参考文献

[1] 雨坪, 上田, 吉田, 白井:“ビット幅を考慮した大規模システム処理系の設計手法について,” 情報処理学会設計自動化研究会,67-2 (1993-6)

[2] 池永, 白井:“複数のアルゴリズムを実行する専用プロセッサのアーキテクチャ設計,” 情報処理学会設計自動化研究会,51-9 (1990-2)